Cloudera Runtime 7.3.2

Amazon S3 Extract

Date published: 2020-07-28
Date modified: 2026-03-31

Legal Notice

Contents

Amazon S3 metadata collection

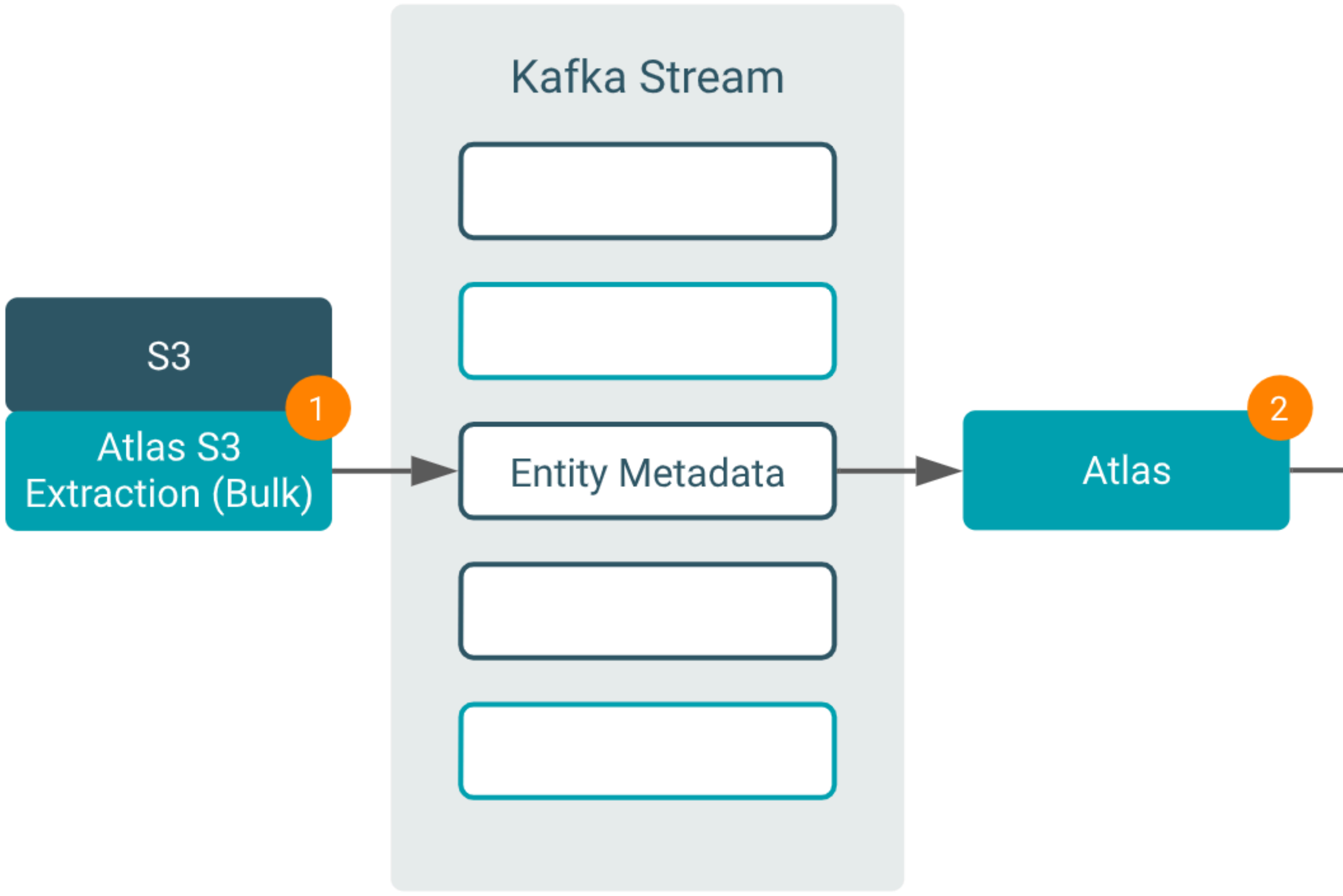
Amazon Simple Storage Service (S3) is a storage solution offered by Amazon Web Services (AWS) that provides highly available storage in the cloud. Clusters deployed in the AWS cloud and in on-premise data centers are using Amazon S3 as persistent storage. Common use cases include BDR (backup and disaster recovery) and persistent storage for transient clusters deployed to the cloud, such as storage for ETL workload input and output.

As with data stored on HDFS and processed using compute engines like Hive and Impala, Atlas can obtain metadata and lineage from Amazon S3 storage. The S3 extraction logic is provided as stand-alone scripts for bulk and incremental metadata collection that you can configure to run through a scheduler.

Typically, you would run the extractor in bulk mode once, then run the extractor in incremental mode on a regular schedule. This method allows Atlas to track changes that occur to data assets; it requires that the bucket publish events to an SQS Queue so the extractor can process all events relevant to the data assets being tracked.

Alternatively, you can run the bulk extraction periodically. With this method, Atlas may miss updates that occur between extractions and will not acknowledge when data assets are deleted.

When running the bulk extractor, Atlas collects metadata for current buckets and objects.



1. The bulk S3 extractor collects metadata from S3 and publishes it on a Kafka topic.
2. Atlas reads the message from the topic and compares it to existing metadata and lineage information.
3. Atlas creates and updates the appropriate entities.

When running the incremental extractor, events in S3 are communicated to Atlas through an SQS Queue:

1. When an action occurs in the S3 bucket.
2. The bucket publishes an event to an SQS queue.
3. The incremental S3 extractor runs and collects the events from the SQS queue and converts them into Atlas entity metadata.
4. The extractor publishes the metadata on a Kafka topic.
5. Atlas reads the message from the topic and compares it to existing metadata and lineage information.
6. Atlas creates and updates the appropriate entities.

The following table lists some differences between object types and supported features offered by Amazon S3 and how those are supported by Atlas:

Feature	Amazon S3	Atlas
System-defined metadata includes properties such as Date, Content-Length, Last-Modified. Some system-defined properties comprise the technical attributes for the object in Atlas.	✅	✅
User-defined metadata consists of custom key-value pairs (in which each key is prefixed with x-amz-meta-) that can be used to describe objects on Amazon S3. Atlas does not collect this kind of metadata; Atlas collects system-defined metadata and tags.	✅	⊘
Versioning. Atlas maintains a single version of a S3 object regardless of how many versions are created in S3. The extractor takes the latest version of an object and replaces previous versions in Atlas. The Atlas audit log will show the updates. If, after metadata is extracted, the latest object's version is deleted, the next extraction will update the Atlas entity metadata with that of the earlier version (which is now the latest).	✅	⊘
Unnamed directories. Unnamed directories are ignored by Atlas.	✅	⊘
Object lifecycle rules. See Object Lifecycle Rules Constraints for more information. Atlas does not support lifecycle rules that remove objects from Amazon S3. For example, an object lifecycle rule that removes objects older than n days deletes the object from Amazon S3 but the event is not tracked by Atlas. This limitation applies to removing objects only.	✅	✓
Amazon Simple Queue Service (SQS). If an SQS queue is configured, Atlas uses Amazon SQS queue for incremental extraction. The queue holds event messages such as notifications of updates or deletes that allow Atlas to track changes for a given object, such as uploading a new version of an object.	✅	✓
Hierarchy in object storage. Atlas uses metadata from S3 to produce a file-system hierarchy.	✓	✅

Feature	Amazon S3	Atlas
Encryption. S3 supports bucket-level and object level encryption. Atlas collects encryption information as metadata. For reading, assuming the AWS credentials are configured such that Atlas can access buckets to collect metadata, the encryption choices are transparent to Atlas.	✓	✓

Accessing AWS

Atlas is integrated with IDBroker for access to the AWS session credential using Kerberos credentials for the atlas service user.

The Atlas S3 extractor authenticates with the service user and receives a delegation token fom IDBroker. The extractor uses the token to request AWS credentials. By default, this credential has a lifetime of seven days, which can be adjusted in Cloudera Manager by modifying the Knox configuration property IDBroker Knox Token TTL.

If IDBroker is not configured, you can provide AWS credentials (AWS access key and secret key or temporary security credential) on the extractor command line or in a configuration file.

AWS objects and inferred hierarchy

Atlas recreates the inferred hierarchy of objects in S3 by creating relationships among objects.

Atlas creates entities for S3 objects following a file system path model. Virtual objects for intermediate directory names are ignored. For example, s3://sampleBucket/testDb/testTable/data.orc and s3://sampleBucket//testDb/testTable/data.orc are treated as the same.

AWS object lifecycle

Atlas can represent the lifecycle of a given S3 object when you use the incremental extractor to collect creation and deletion events from S3.

To collect these events, you need to provide an existing SQS queue or create a new one and to configure your S3 bucket to publish the events to the queue.

Note: Note that when you configure Amazon S3 Lifecycle policies to automatically delete versions of objects or objects after an interval, S3 does not send event notifications and therefore Atlas will not reflect that these objects have been deleted.

AWS configuration

You need the following items to be set up for AWS before you can extract metadata from an S3 bucket:

- AWS credentials
    - The account for this AWS credential must have appropriate permissions to read from S3 buckets and the configured SQS Queue.
    - If you are using Cloudera on cloud, these credentials are configured as part of SDX and there isn't anything extra you need to do.
    - If you are using Cloudera in a data center (installed on-premise or on virtual hardware in the cloud), you need to supply the extractor with AWS credentials. See Configure credentials for Atlas extraction.

- S3 Bucket access policies
  - If you use a bucket other than the one specified by default when creating a Cloudera environment, you need to update the aws-cdp-bucket-access-policy with the additional bucket as a resource. See Minimal setup for cloud storage.
- S3 Bucket encryption
  - See AWS documentation "Amazon S3 default encryption for S3 buckets". Your choices for bucket and object encryption are transparent to Apache Atlas as Atlas accesses the bucket through the authorized credentials you provide.
- S3 Bucket event publication
  - If you intend to collect all activity that occurs on the data assets you are tracking, you should configure the buckets to publish, update and delete events. See To configure an S3 bucket to publish events.
- SQS queue
  - Update an existing queue or configure a new one to pass S3 event information to Atlas. See To configure an SQS queue suitable for Atlas extraction.

To configure an SQS queue suitable for Atlas extraction

Use the following information to configure an SQS queue for Atlas extraction.

Prerequisite: You must add Amazon Resource Names (ARN) of the IAM role for the user / group configured under Cloudera. For more information, see Onboarding Cloudera users and groups for cloud storage.

1. Log in to the AWS Management Console (https://aws.amazon.com/console/) with the AWS account (IAM user).

   Make sure that there are adequate privileges to administer the queue and the buckets involved in metadata extraction.
2. Open the Simple Queue Service setup page and select Services > Application Integration > Simple Queue Service.
3. Click Create New Queue or Get Started Now, if the region has no configured queues.
4. For each region that has Amazon S3 buckets, create a queue as follows:

   a. Click Create Queue. Enter a Queue Name, name it, for example: aws-s3-cdp-atlas-events, click the Standard Queue (not FIFO), and next under configuration you must configure the queue using the following settings:

      Default Visibility Timeout =10 minutes

      Message Retention Period =14 days

      Delivery Delay = 0 seconds

      Receive Message Wait Time = 0 seconds

      Maximum message size = 256 KB

   b. Under Access Policy, choose method Advanced and insert the access policy similar to below:

```
{
"Version": "2008-10-17",
"Id": "__default_policy_ID",
"Statement": [
{
“Xxxx”: "__owner_statement",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::00000000000:root"
},
"Action": "SQS:ReceiveMessage",
"Resource": "arn:aws:sqs:us-west-1:00000000000:aws-s3-cdp-atlas-events"
},
{
"xxxx": "__sender_statement",
"Effect": "Allow",
"Principal": {
"Service": "s3.amazonaws.com"
},
"Action": "SQS:SendMessage",
"Resource": "arn:aws:sqs:us-west-1:00000000000:aws-s3-cdp-atlas-events"
}
]
}
```

Note: The value above in the snippet arn:aws:iam::00000000000:root is the ARN for the AWS role mentioned as indicated in the prerequisite.

c. Configure at rest Encryption to meet your organization standards. In-transit encryption is configured automatically.

Caution: Do NOT change other default configuration settings.

d. Click Create Queue.

5. Repeat this process for each region that has Amazon S3 buckets.

For more information about creating an SQS queue, see Creating an Amazon SQS queue.

To configure an S3 bucket to publish events

Before you configure an S3 bucket, you must make sure that Cloudera can access the S3 bucket.

For more information, see Minimal setup for cloud storage.

1. Log into AWS.
2. Go to S3 Service.
3. Select the bucket you want to update.
4. Go to Properties > Advanced Settings > Events.
5. In the Events dialog box, click Add notification.
6. Enter a name for the event notification, such as Atlas Create/Delete Events.
7. In the Events list, select All object delete events and All object create events.
8. In Send to, choose SQS Queue.
9. In SQS, enter the name of the existing queue or choose "Add SQS queue ARN" to specify an existing queue with its ARN.
10. Click Save.

S3 Extractor configuration

The Atlas metadata extractor for AWS S3 is a stand-alone application that you can run on the Atlas host as needed to provide comprehensive metadata for data assets stored in S3.

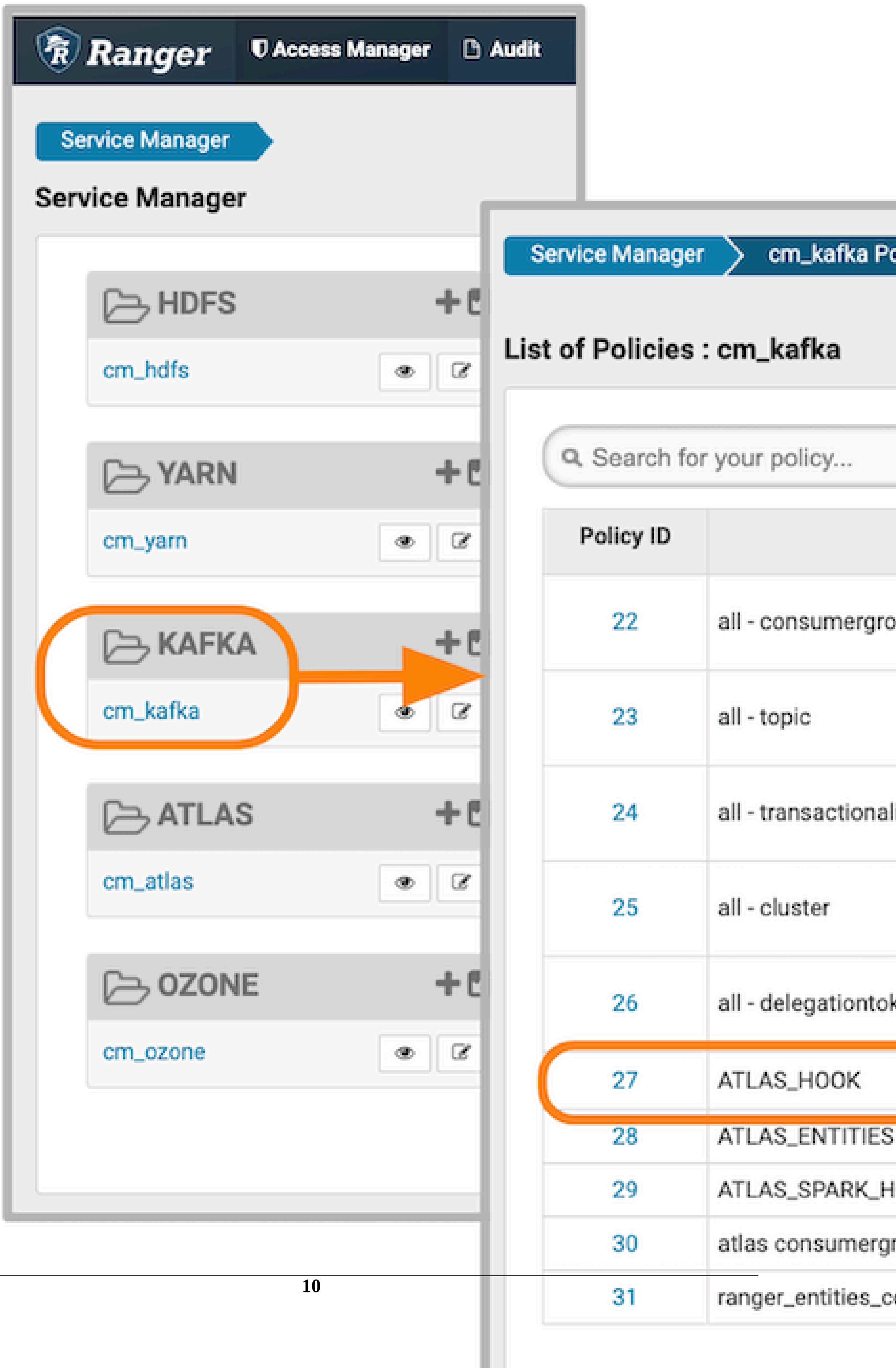
Prerequisites

To setup the extractor follow these steps as part of the prerequisite.

- In Apache Ranger, in the Kafka service policies, add the "Publish" permission for the atlas user in the Ranger Atlas_Hook policy.

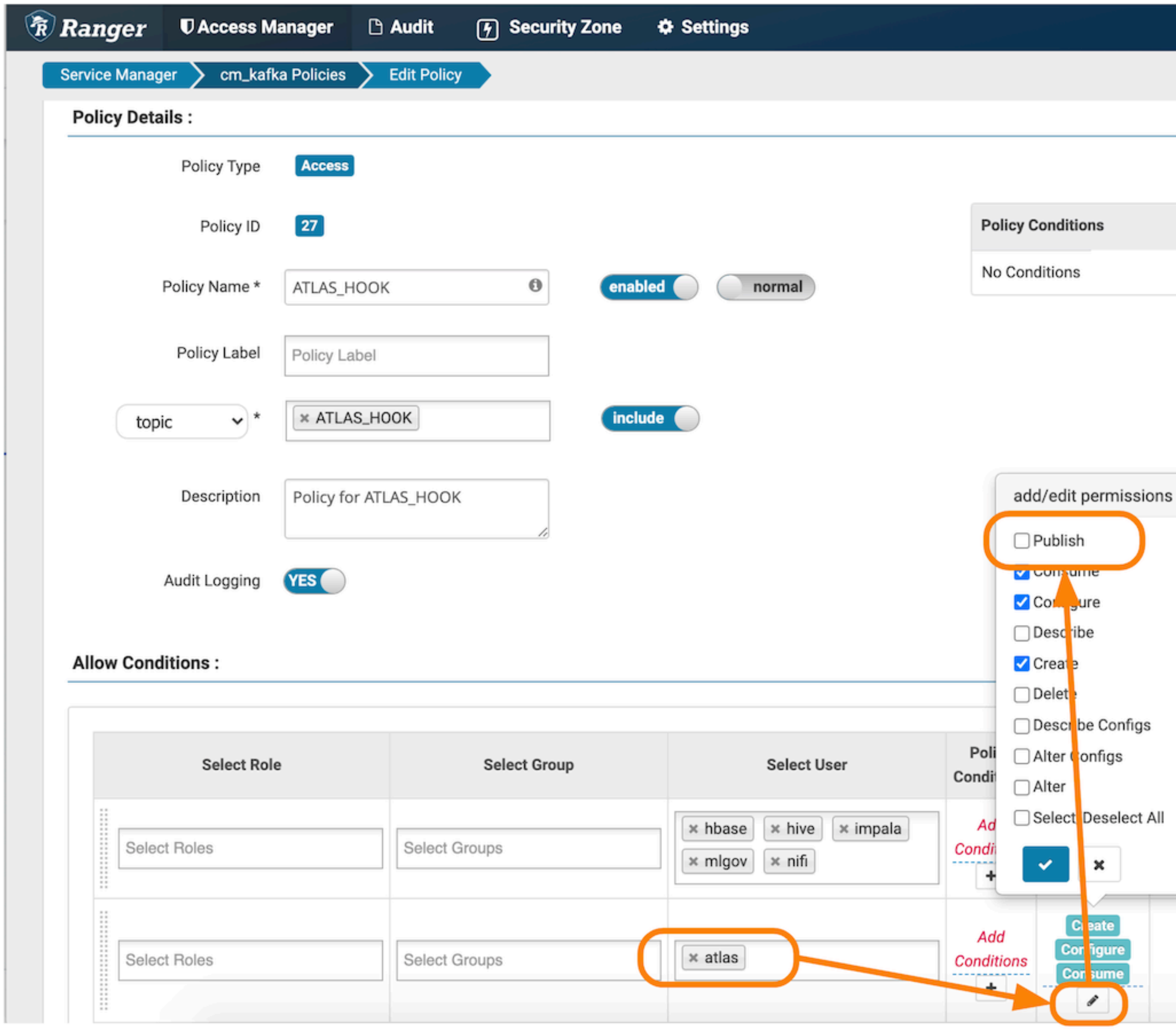
  1. Contact an administrator with privileges to update Ranger policies.
  2. Open **Service Manager** > **Kafka policies**.

3. Open the ATLAS_HOOK



Policy ID	
22	all - consumergro
23	all - topic
24	all - transactional
25	all - cluster
26	all - delegationtok
27	ATLAS_HOOK
28	ATLAS_ENTITIES
29	ATLAS_SPARK_H
30	atlas consumergr
31	ranger_entities_c

4. In the policy details, look for the atlas user and add the Publish permission to that condition.



Policy Details :

Policy Type: Access

Policy ID: 27

Policy Name *: ATLAS_HOOK — enabled — normal

Policy Label: Policy Label

topic *: ATLAS_HOOK — include

Description: Policy for ATLAS_HOOK

Audit Logging: YES

Policy Conditions: No Conditions

Allow Conditions :

Select Role	Select Group	Select User	Policy Conditions	
Select Roles	Select Groups	hbase, hive, impala, mlgov, nifi	Add Conditions	
Select Roles	Select Groups	atlas	Add Conditions	Create, Configure, Consume

add/edit permissions
- [ ] Publish
- [x] Consume
- [x] Configure
- [ ] Describe
- [x] Create
- [ ] Delete
- [ ] Describe Configs
- [ ] Alter Configs
- [ ] Alter
- [ ] Select/Deselect All

- SSH into the Atlas host. You can locate the Atlas host URL through Cloudera Manager.
- Set JAVA_HOME if it is not already set.

Configure credentials for Atlas extraction

When Atlas is running as part of SDX, you can skip this step but follow other instructions.

The extractor is authenticated with AWS through integration with the IDBroker. To authenticate the extractor otherwise, you'll need to provide AWS credentials to the extractor, either through a configuration file or on the command line.

Note: Any changes to the AWS credentials (for example, if you rotate credentials on a regular basis) must be for the same AWS account (IAM user). Changing the AWS credentials to those of a different IAM user results in errors from the Amazon Simple Queue Service (used transparently by Atlas).

Atlas can use either of the following authentication methods:

1. Basic credentials: AWS access key and secret key.
2. Temporary credentials: AWS session token along with access and secret key. See Getting Temporary Credentials with AWS STS.

Both methods require you to specify the region where the authentication is valid.

- If the connection to AWS S3 expires, Atlas will retry X times to re-establish the connection. This property atlas.s3.extraction.max.reconnect.count controls the number of retries, which defaults to 2.

Extraction Command

The extractor script is located in the extractors/bin directory in $ATLAS_HOME.

By default, the location is /opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/aws-s3-extractor.sh

Note: The extractor configuration file includes properties that require AWS S3 configuration values. Be sure to set those values in the configuration file before running the command.

Using the following command to start S3 extraction:

```
$ aws-s3-extractor.sh [--config <config file>]
                      [--extraction <mode>]
                      [--failOnError]
                      [--logdir <log directory>]
```

Command Line Option	Description

`-c <config file>`
`--config <config file>`
	The path of the configuration file containing extraction configuration properties. See Extractor configuration properties on page 12.
	Defaults to $ATLAS_HOME/extractors/conf/aws-s3.conf

`-e <mode>`
`--extraction <mode>`
	The type of extraction to be done, “Incremental” or “Bulk”. If not specified, the extraction mode is "Bulk".

-f --failOnError	Flag to set the extraction to stop when it encounters an exception. Without the flag, the extraction continues after exceptions.

Extractor configuration properties

Some of the extractor configuration properties.

Configuration Parameter (prefixed with atlas.s3.extraction)	Purpose
.aws.region	Required. AWS S3 region for extraction. Cloudera on cloud: provided by IDBroker integration. Cloudera Data Center: the region must be provided.
.access.key .secret.key .session.token	AWS S3 credentials. Cloudera on cloud: provided by IDBroker integration. Cloudera Data Center: • For basic authentication, provide access and secret keys. • For session based authentication, provide all three credentials. Configure credentials for Atlas extraction on page 11.

Configuration Parameter (prefixed with atlas.s3.extraction)	Purpose
.whitelist.paths	Set of data assets to include in metadata extraction. If no paths are included, the extractor extracts metadata for all buckets. Buckets inside the region and accessible by the credential used to run the extractor. For Cloudera on cloud, this is defined in aws-cdp-bucket-access-policy. The list can include buckets or paths; separate multiple values with commas. See Defining what assets to extract metadata for on page 13.
.blacklist.paths	Set of data assets to exclude from metadata extraction. The list can include buckets or paths; separate multiple values with commas. See Defining what assets to extract metadata for on page 13.
.max.object.per.call	Number of objects returned in one call to AWS S3 in bulk extraction. Defaults to 1000.
.resume.from.progress.file	Enable bulk extraction resume. Disabled by default.
.progress.file	Name of the progress file that tracks bulk extraction that will be used if the extraction has to be resumed mid-way after a failure. This file is created in the Atlas log directory. Defaults to extractor_progress_file.log
.max.reconnect.count	Number of times the extractor will try to reconnect to AWS in case of credentials expiring or retryable exception from AWS S3. Defaults to 2.
.fs.scheme	File system scheme used in case of AWS S3. At this time, S3a is the only option.
.incremental.queueName	Required for incremental mode. AWS SQS Queue name configured to get the AWS S3 bucket events.
.incremental.messagesPerRequest	The number of messages the extractor will try to get from the SQS queue in one call. Defaults to 10.
.incremental.requestWaitTime	The interval that the extractor will wait to collect messages so long as the limit set in .incremental.messagesPerRequest hasn't been met. Defaults to 20 seconds.
.incremental.max.retry	Maximum retry count if the SQS queue doesn't respond during an extractor request. Defaults to 20.

Defining what assets to extract metadata for

By default, the Atlas S3 extractor collects metadata for all buckets that are visible through the active IAM policy.

You can specify the objects to include using configuration properties to include buckets, directory trees, and specific objects (whitelist.paths) or exclude buckets, directory trees, and specific objects (blacklist.paths). When you include a path in the whitelist, Atlas extracts metadata for that directory object and all objects underneath it.

The include and exclude path lists are comma-separated lists of paths for any given bucket. Each item in the list can be as general as a bucket name or a directory path, or as specific as a low level directory or even a single object. The path names in the lists are formatted as S3 URIs, so it may be that they include repeated slash characters to represent virtual objects in S3.

If the same path ends up on both lists, the exclude list wins.

The following examples show how to include objects in or exclude objects from metadata collection.

This example includes a single file from the 2018 directory and all files in the data subdirectory for 2019 and 2020:

```
atlas.s3.extraction.whitelist.paths=s3a://acme-finance-set4/year-end/2018/data/storage/only-this-file.json,acme-finance-set4/year-end/2019/data,acme-finance-set4/year-end/2020/data
```

This example includes a single file from one bucket and all contents of a second bucket:

```
atlas.s3.extraction.whitelist.paths=s3a://demo-bucket/demo.txt,   s3a://acme-finance-set4/
```

This example excludes specific files from a directory. If no include paths were given, only these files would be excluded from all the buckets available.

atlas.s3.extraction.blacklist.paths=s3a://acme-finance-set4/year-end/admin/tools-archive.zip,atlas.s3.extraction.blacklist.paths=s3a://acme-finance-set4/year-end/reporting/template.xls

Running bulk extraction

Bulk extraction takes a snapshot of the metadata for the current version of all objects defined by the allowlist as of the initial start time of the extraction.

If the bulk extraction process runs for a few hours, there may be additions or modifications to AWS S3 content after the start of the extraction. The changes after the extraction start time are not included in the extraction.

The first time bulk extraction runs, it creates entities in Atlas for each S3 bucket and object. If you run bulk extraction again, it will:

- Update Atlas entities with new information it finds for buckets or objects whose qualifiedNames match existing entities.
- Create new entities for any new buckets or objects that are not found in Atlas.

The second extractions will NOT:

- Mark entities deleted in Atlas if the object from the first extraction does not exist at the time of the second extraction.
- Collect version information for objects other than the current version.

The following command line example runs the bulk extraction. Assuming the mandatory properties are set in the default configuration file, no parameters are required:

```
/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/aws-s3-extractor.sh
```

Running incremental extraction

Any object added to S3, including new buckets and new versions of existing objects, publishes an event to AWS SQS (as configured by the user).

Incremental extraction consumes the events from SQS queue and creates corresponding Atlas entities.

After running the bulk extraction once, you can run incremental extraction to update Atlas metadata based on the events tracked in S3. This method has the benefit of tracking changes that may not be captured in successive bulk extraction, such as when more than one object version is created between extractions or when objects are deleted.

Note: Before running incremental extraction, you must set up event tracking in the AWS S3 buckets and configure an SQS queue to collect the S3 events.

The incremental extraction operation reads all events produced by S3, then filters the list based on the include and exclude configurations for the extractor. The appropriate messages are passed into the Kafka queue and consumed by Atlas. Because the messages sit in the SQS queue, you don't need to run the incremental extraction operation constantly: when it runs, it will go through all the messages waiting in the queue. Run the extractor often enough that the metadata is not stale by the time it processes, but not so often that it takes longer to run than the time between runs. In addition, note that the SQS Queue does not maintain the events forever. By default, the messages stay for 4 days; you can configure this value to hold the events longer to protect against the chance that the extractor isn't able to process events in time.

In addition to the include and exclude lists described in Defining what assets to extract metadata for on page 13, the configuration properties relevant to incremental extraction control the connection to the SQS Queue and how long the extraction actively waits for events from S3. You must configure the name of the queue that the events are published on. You can control how long the extractor waits to receive messages from the queue and how many messages to wait for. When configuring the number of messages to collect, remember that the extractor collects all events from the queue, not just the events corresponding to objects in the allowlist.

In addition, you can set the timeout or retry time so the extractor doesn't keep running even when it isn't able to connect to the queue.

You must plan whether to set up a repeating job to run incremental extraction. Assuming you have a lot of changes that you want to track:

- Incremental extraction takes more time to process the individual changes than running bulk extraction.
- Incremental extraction maintains a complete history of objects (shows updates and deletes)
- Batch is faster, but only captures current state and leaves deleted entities in Atlas

In conclusion, if you want to capture the history of S3 objects, run bulk extraction once then run incremental extraction on a regular basis. As you have more experience with the volume of changes and the speed of the extraction, you can determine if you should run the incremental extraction more frequently.

The following command line example runs the incremental extraction. Assuming the mandatory properties are set in the default configuration file, only the parameter to enable incremental mode is required:

```
/opt/cloudera/parcels/CDH/lib/atlas/extractors/bin/aws-s3-extractor.sh -e IN
CREMENTAL
```

Logging Extractor Activity

The extractor produces a log file in the same location as the Atlas server log file. While the server log is available through Cloudera Manager, the extractor file is not.

About this task

To locate the extractor log file.

Procedure

1. SSH into the Atlas host.
2. Find the active Atlas configuration:

   $ ps -ef | grep atlas.conf

   In the output, look for the string

   ```
   atlas.conf
   ```

   For example:

   atlas.conf=/var/run/cloudera-scm-agent/process/99-atlas-ATLAS_SERVER/conf
3. In the configuration directory you found in the previous step, read the atlas-log4j.properties file.

   Look for the log.dir entry. For example:

   log.dir=/var/log/atlas
4. In the log directory, you'll see the extractor log file and the progress file:
   - aws-s3-extractor.log
   - extractor_progress_file.log

S3 actions that produce or update Atlas entities

As objects are created in S3, Atlas generates entities to represent those objects. Atlas does not create processes to represent S3 operations.

The following table lists the S3 actions that produce or update metadata in Atlas.

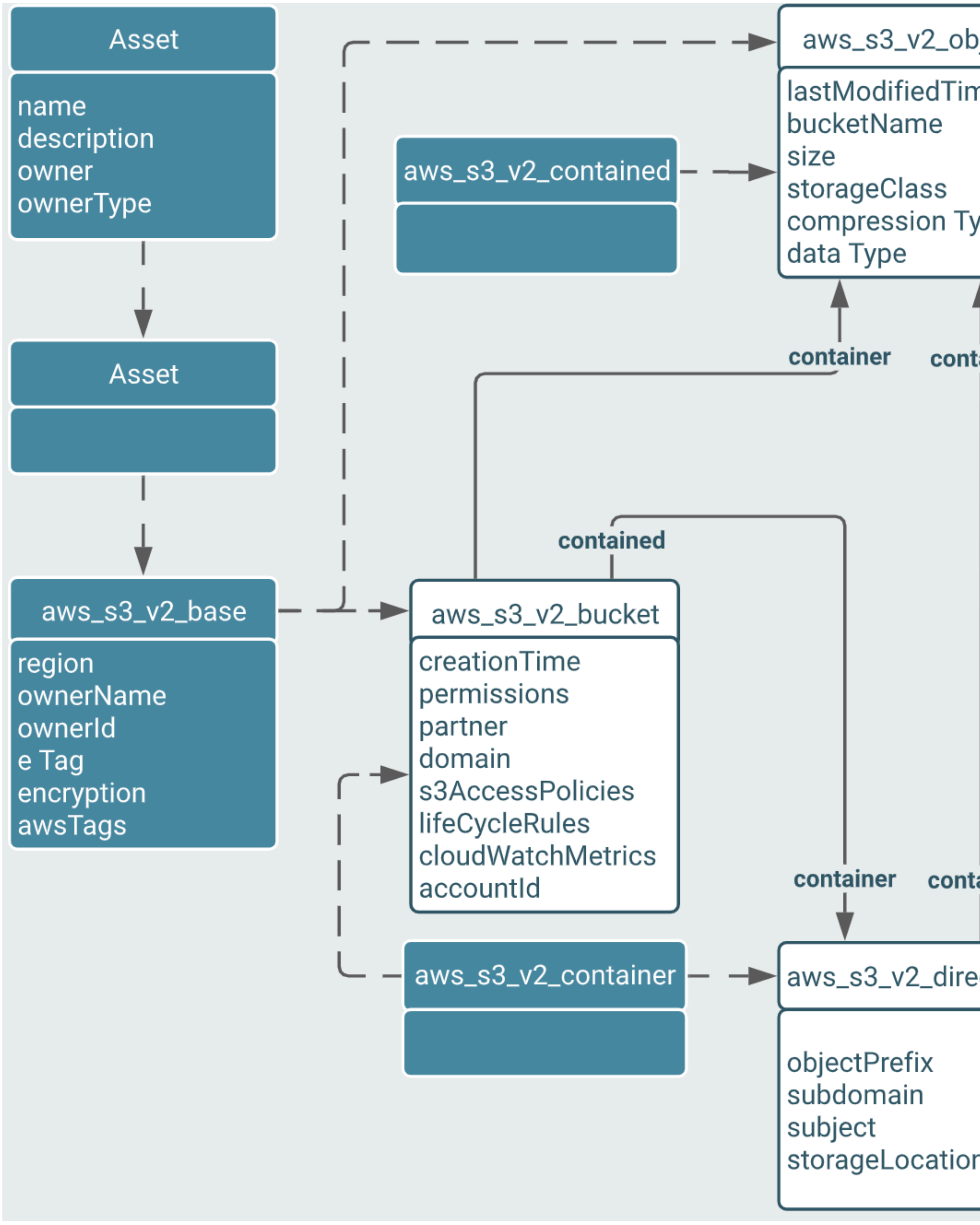
S3 Evemt	Bult Extraction	Incremental Extraction
Bucket		
• creation	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current bucket metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete (No S3 event is produced to indicate that a bucket has been deleted.)	No	No
Object (directory)		
• creation	Yes	Yes
• upload	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#
Object (file)		
• creation	Yes	Yes
• upload (When you upload a "folder" to S3, the objects in the folder are prefixed with the folder name.)	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#

Notable actions in S3 that do NOT produce metadata entities include any actions that affect only data and not metadata. Actions that do not create Atlas entities include:

- Bucket deletion does not produce an event and is ignored by Atlas.
- Events associated with subfolders when a folder is uploaded. When you upload a "folder" to S3, the objects in the folder are prefixed with the folder name, so the folder and any subfolders are not created as objects in S3. Atlas ignores metadata or events for subdirectories included in the uploaded folder; Atlas tracks metadata for the uploaded folder and the data asset objects in the uploaded folder.
- Event notifications produced when a version other than the latest version is deleted.

S3 entities created in Atlas

Note that previous versions of Atlas included other metadata models for S3 data. The version 2 (V2) models are supported as of Cloudera 7.2.0.


Asset
name
description
owner
ownerType
Asset
aws_s3_v2_base
region
ownerName
ownerId
e Tag
encryption
awsTags
aws_s3_v2_contained
aws_s3_v2_ob
lastModifiedTim
bucketName
size
storageClass
compression Ty
data Type
container
cont
contained
aws_s3_v2_bucket
creationTime
permissions
partner
domain
s3AccessPolicies
lifeCycleRules
cloudWatchMetrics
accountId
container
conta
aws_s3_v2_container
aws_s3_v2_dire
objectPrefix
subdomain
subject
storageLocatior

AWS S3 Base

S3 entities that represent S3 buckets and objects inherit attributes from the S3 base entity type. Atlas does not generate entities of this type; it is used only as a supertype for other entity definitions.

Identifier	Description
typeName	AWS_S3_V2_BASE
name	Inherited from Asset supertype.
description	Inherited from Asset supertype.
owner	Inherited from Asset supertype.
ownerType	Inherited from Asset supertype.
qualifiedName	Fully qualified name for each entity at Atlas which is used to create lineage. See the child entity types for details on how the qualified name is generated.
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.
ownerID	Owner ID as reported by S3.
eTag	A hash of an object as reported by S3. This value is updated when a new version of an object is added to a bucket.
encryption	Encryption algorithm as reported by S3.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.

AWS S3 Container

The S3 model includes two entity definitions that generalize the hierarchical relationship among the other entity types. Atlas does not generate entities of this type; it is used only as a supertype for other entity definitions. The "container" entity type includes a relationship that represents the one-to-many relationship between a directory and objects in the directory. The "contained" relationship looks the other way to represent the one-to-one relationship between an object and its parent, either a directory or a bucket.

The entity definitions for aws_s3_v2_bucket and aws_s3_v2_directory inherit the attributes from the container entity definition.

name	AWS_S3_V2_CONTAINER
relationship: contained	The relationship between a bucket or directory and an array of one or more objects, including subdirectories.

AWS S3 Contained

The S3 model includes two entity definitions that generalize the hierarchical relationship among the other entity types. Atlas does not generate entities of this type; it is used only as a supertype for other entity definitions. The "contained" relationship represents the one-to-one relationship between an object and its parent, either a directory or a bucket. The "container" entity type provides the opposite relationship: the one-to-many relationship between a bucket or directory and objects in the directory.

The entity definition for aws_s3_v2_object inherits the attributes from the contained entity definition.

Identifier	Description
name	AWS_S3_V2_CONTAINED

Identifier	Description
relationship: container	The one-to-one relationship between an object and its parent directory or bucket.

AWS S3 Bucket

The Atlas entity type for bucket metadata inherits attributes from the AWS_S3_V2_BASE entity type and includes additional attributes specific to bucket metadata.

Identifier	Description
typeName	AWS_S3_V2_BUCKET
name	Name of the object as reported by S3.
description	Description provided in Atlas metadata. No value is collected from S3.
owner	Atlas user who ran the metadata extractor.
ownerType	Not used.
qualifiedName	Fully qualified name of the bucket. The name includes Atlas's version of the S3 URI followed by @cluster_name. For example: s3a://acme-finance-bucket@cm
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.
ownerID	Owner ID as reported by S3.
eTag	A hash of the bucket as reported by S3.
encryption	Bucket-level encryption algorithm as reported by S3.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.
permission	Permissions at AWS S3
creationTime	Creation data as reported by S3. DIstinct from Atlas entity create time.
partner	Not used. Supports backward compatibility.
domain	Not used. Supports backward compatibility.
s3AccessPolicies	Not used. Supports backward compatibility.
lifeCycleRules	Not used. Supports backward compatibility.
cloudWatchMetrics	Not used. Supports backward compatibility.
accountId	Account ID as reported by S3.
relationship: contained	The relationship between a bucket and an array of one or more objects, including directories.

AWS S3 Object

Though AWS S3 includes both files and directories as objects, the Atlas model creates separate entity types for objects from directories.

The Atlas entity type for object metadata inherits attributes from the AWS_S3_V2_BASE entity type and includes additional attributes specific to object metadata. Note that Atlas makes a distinction between files and directories. Atlas creates entities of this type for files; it creates entity types of AWS_S3_V2_DIRECTORY for directories.

Table 1:

Identifier	Description
typeName	AWS_S3_V2_OBJECT
name	Name of the object as reported by S3.
description	Description provided in Atlas metadata. No value is collected from S3.
owner	
ownerType	Not used.
qualifiedName	Fully qualified name of the object. The name includes Atlas's version of the S3 URI followed by @cluster_name. For example: s3a://acme-finance-bucket/2018directory/specific_file.json@cm
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.
ownerID	Owner ID as reported by S3.
eTag	A hash of an object as reported by S3. This value is updated when a new version of an object is added to a bucket.
encryption	Encryption algorithm as reported by S3. If no object-level encryption is specified, this value is the same as the bucket-level encryption.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.
lastModifiedTime	The last modified time of the object as reported from S3.
bucketName	The name of the bucket containing this object.
size	The size of the object as reported from S3.
storageClass	Storage class used for storing the object as reported from S3.
compressionType	Compression type used for the object as reported from S3.
data Type	Owner of the object
relationship: container	The one-to-one relationship between an object and its parent directory or bucket.

AWS S3 Directory

The Atlas entity type for "directory" metadata inherits attributes from the AWS_S3_V2_OBJECT entity type and includes additional attributes specific to object metadata.

Identifier	Description
typeName	AWS_S3_V2_DIRECTORY
name	Name of the object as reported by S3.
description	Description provided in Atlas metadata. No value is collected from S3.
owner	
ownerType	Not used.
qualifiedName	Fully qualified name of the directory object. The name includes Atlas's version of the S3 URI ending with a slash, followed by @cluster_name. For example: s3a://acme-finance-bucket/2018directory/@cm
region	AWS S3 region as reported by S3.
ownerName	Owner as reported by S3.

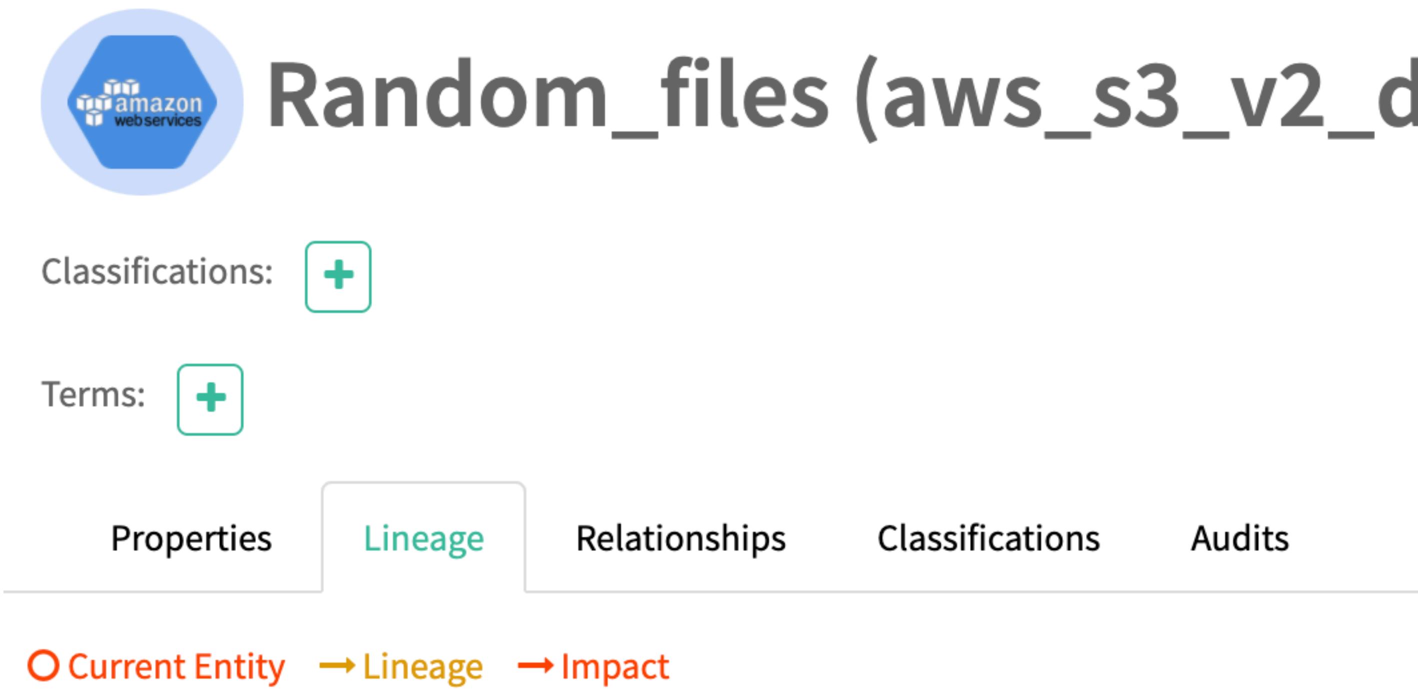
Identifier	Description
ownerID	Owner ID as reported by S3.
eTag	A hash of an object as reported by S3. This value is updated when a new version of an object is added to a bucket.
encryption	Encryption algorithm as reported by S3. If no directory-level encryption is specified, this value is the same as the bucket-level encryption.
awsTags	Object tags defined in S3. Atlas stores S3 tags as an array of key-value pairs.
storageLocation	Not used. Supports backward compatibility.
objectPrefix	Not used. Supports backward compatibility.
subdomain	Not used. Supports backward compatibility.
subject	Not used. Supports backward compatibility.
relationship: contained	The relationship between a directory and an array of one or more objects, including other directories.
relationship: container	The one-to-one relationship between a directory and its parent directory or bucket.

S3 relationships

The S3 model includes relationships that describe the perceived hierarchy among objects in a bucket. These relationships are created through "container" and "contained" relationship attributes.

Example of Atlas S3 Lineage

An example of S3 lineage that includes S3 objects connected to a Hive table.


amazon web services
Random_files (aws_s3_v2_d
Classifications:
Terms:
Properties
Lineage
Relationships
Classifications
Audits
Current Entity
Lineage
Impact

S3 entity audit entries

The number of Atlas audits for S3 entities may vary based on whether or not you are using incremental extraction to collect metadata from S3. For example, if you run bulk extraction against a bucket, then objects in the bucket are updated twice before bulk extraction runs again, Atlas will show the creation audit event followed by a single update audit event. If instead you run bulk extraction against a bucket, then trigger incremental extraction, the Atlas audit will show the creation audit event followed by an update audit event for the first update and an update audit event for the second update.

S3 Evemt	Bult Extraction	Incremental Extraction
Bucket		
• creation	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current bucket metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete (No S3 event is produced to indicate that a bucket has been deleted.)	No	No
Object (directory)		
• creation	Yes	Yes
• upload	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#
Object (file)		
• creation	Yes	Yes
• upload (When you upload a "folder" to S3, the objects in the folder are prefixed with the folder name.)	Yes	Yes
• update	Yes (Bulk extraction updates entities only if the current object metadata differs from the metadata collected in the previous extractor run; interim changes are not captured.)	Yes
• delete	No	Yes#