Cloudera Runtime 7.3.3

Storing Data Using Ozone

Date published: 2020-07-28
Date modified: 2026-12-13



https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Upgrading Ozone overview 8
Preparing Ozone for upgrade 8
Backing up Ozone 8
Upgrading Ozone parcels 9

Ozone S3 Multitenancy overview (Technical Preview) 9
Prerequisites to enable S3 Multitenancy 9
Enabling S3 Multi-Tenancy 10
Tenant Commands 10

Multi Protocol Aware System overview 11
Upgrading this feature 12
Files and Objects together 12
Bucket Layout 12
Ozone FS namespace optimization with prefix 14
OBS as Pure Object Store 17
Configuration to create bucket with default layout 17
Performing Bucket Layout operations in Apache Ozone using CLI 18
FSO operations 18
Object Store operations using AWS client 19
Ozone Ranger policy 20

Ozone S3 Security Token Service overview 20
Enabling Ozone S3 Security Token Service 21
Additional configurable properties 23
IAM Session Policy to Ranger permission or action mapping 24
Ranger policy setup 26
Creating a user in Ranger 26
Creating a role in Ranger 28
Creating a resource policy for Assume_Role in Ranger 30
Creating resource policies for volume, bucket, and key permissions 33
How to use Ozone S3 Security Token Service 43
Creating an Ozone bucket and installing AWS CLI 43
Creating a user in Ozone 44
Obtaining S3 credentials in Ozone 44
Generating the Ozone S3 Security Token Service using the AssumeRole API 45
Applying the conflicting action conditions to AssumeRole requests using the API 50

Ozone Ranger Integration 51
Configuring a resource-based policy using Ranger 51

Snapshot support in Ozone 54
Cluster and hardware configuration in snapshot deployment 55

Erasure Coding overview.....55
Enabling EC replication configuration cluster-wide.....56
Enabling EC replication configuration on bucket.....56
Enabling EC replication configuration on keys or files.....57

Master node decommissioning in Ozone.....57
SCM decommissioning.....57
Decommissioning SCM.....57
OM decommissioning.....59
Decommissioning OM Node.....59
Adding new Ozone Manager node.....61

Ozone recon heatmap.....62
Accessing Ozone Recon Web UI.....62
Ozone recon heatmap.....63

Container Balancer overview.....67
Container balancer CLI commands.....67
Determining the threshold.....67
Choosing an appropriate value for the threshold.....68
Configuring container balancer service.....69
Activating container balancer using Cloudera Manager.....70

Disk Balancer overview.....70
Architecture and how it works.....71
Authentication and authorization.....72
Configuring disk balancer.....73
Disk balancer CLI commands.....74

Ozone Replication Manager overview.....78
Ozone Replication Manager throttling of tasks.....78
Replicate container commands.....78
Delete container replica commands.....79
EC reconstruction commands.....79
Configurations for throttling of tasks.....80

Storage container reconciliation overview.....81
Reconciling storage containers.....82

Managing Ozone quota.....84
Understanding quota.....84
Storage Space level quota considerations.....84
Namespace quota considerations.....85
Additional quota considerations.....85
Commands for managing volumes and buckets.....85
Commands for managing volumes.....86
Commands for managing buckets.....89

Managing storage elements by using the command line interface.....92
Commands for managing volumes.....92
Assigning administrator privileges to users.....95
Commands for managing buckets.....96
Commands for managing keys.....99

Using Ozone S3 Gateway to work with storage elements.....101
Configuration to expose buckets under non-default volumes.....101
REST endpoints supported on Ozone S3 Gateway.....102
Configuring Ozone to work as a pure object store.....102
Access Ozone S3 Gateway using the S3A filesystem.....103
Accessing Ozone S3 using S3A FileSystem.....103
Examples of using the S3A filesystem with Ozone S3 Gateway.....106
Access Ozone S3 Gateway from Cloudera components using S3A filesystem.....106
Using the AWS CLI with Ozone S3 Gateway.....107
Configuring an https endpoint in Ozone S3 Gateway to work with AWS CLI.....107
Examples of using the AWS CLI for Ozone S3 Gateway.....107
Accessing Ozone object store with Amazon Boto3 client.....109
Obtaining resources to Ozone.....109
Obtaining client to Ozone through session.....109
List of APIs verified.....109

Ozone HttpFS gateway.....112
Supported operations.....112
Adding Ozone HttpFS gateway.....114
Configuring TLS/SSL encryption manually for Ozone HttpFS gateway.....116
Configuring HttpFS proxy users.....117

Working with Ozone File System (ofs).....118
Setting up ofs.....118
Volume and bucket management using ofs.....119
Key management using ofs.....120

Working with Ozone File System (o3fs).....121
Setting up o3fs.....121

Ozone configuration options to work with Cloudera components.....121
Configuration options for Spark to work with Ozone File System (ofs).....122
Configuration options to store Hive managed tables on Ozone.....122
Configuration options for Impala to work with Ozone File System.....122
Configuration options for Oozie to work with Ozone storage.....123
Configuration options for Cloudera components to work with Ozone S3.....123
Setting up an external account for Ozone S3 access.....123
Enabling Hive and Impala to use the credentials.....124
Setting up S3 connector.....127
Configuring Hive access for Ozone using S3A.....128
Configuring Impala access for Ozone using S3A.....130
Configuring Spark access for Ozone using S3A.....131

Overview of the Ozone Manager in High Availability..... 133
Considerations for configuring High Availability on the Ozone Manager....133
Ozone Manager nodes in High Availability....133
Read and write requests with Ozone Manager in High Availability....133

Overview of Storage Container Manager in High Availability.....134
Considerations for configuring High Availability on Storage Container Manager....134
Storage Container Manager operations in High Availability....135

Offloading Application Logs to Ozone.....135

Removing Ozone DataNodes from the cluster.....136
Decommissioning Ozone DataNodes....136
Placing Ozone DataNodes in offline mode....137
Configuring the number of storage container copies for a DataNode....137
Recommissioning an Ozone DataNode....138
Handling datanode disk failure....138

Multi-Raft configuration for efficient write performances..... 138

Working with the Recon web user interface..... 139
Access the Recon web user interface....139
Elements of the Recon web user interface....140
Overview page....140
DataNodes page....140
Pipelines page....141
Missing Containers page....142

Configuring Ozone to work with Prometheus.....143

Ozone trash overview.....145

Configuring the Ozone trash checkpoint values.....145

Ozone topology awareness.....146
Topology hierarchy....146
RATIS/THREE Data....147
Erasure Coding data....148
Customizing Ozone network topology....149
Verifying Ozone network topology using the CLI....150

Ozone Placement Policy.....153
Placement Policy for Ratis Containers....153

Placement Policy for Erasure Coded Containers....154

Ozone volume scanner....154

Ozone OMDBInsights....155
Accessing Recon Web UI....155
OMDBInsights....156

Cluster Capacity....162
Ozone cluster (physical) capacity....162
Service capacity....162
Pending deletions....163
Datanode insights....164

Upgrading Ozone overview

You must understand the overview of the upgrade feature. By upgrading the Cloudera Runtime parcels using Cloudera Manager, Ozone is also upgraded. This Ozone upgrade provides you with new features that are made available with the Cloudera Runtime release.

This feature helps you upgrade or downgrade Ozone. Ozone upgrade from Cloudera Manager is managed by upgrading the Cloudera parcels. Before upgrading the Cloudera parcels, as a pre-upgrade step, you must take a backup of OM metadata and SCM metadata. Ozone will be brought to a read-only state before the upgrade and this helps the OMs to synchronize before the upgrade. The upgrade is completely managed by Cloudera Manager.

When the new version of Ozone starts, new features are not yet available. This allows you to downgrade to an older version. In case you wish to downgrade, then the older version of Ozone is restored. However, data written in the newer version is still readable by the older version of Ozone.

If you wish to finalize the upgrade and enable the new features, you must run the Finalize Upgrade command. This updates the metadata layout of Ozone services, persists the changes required for the new version, and enables the new features.

Note: After running the Finalize Upgrade command, it is not possible to downgrade.

Preparing Ozone for upgrade

The Ozone upgrade procedure has three steps: first, prepare Ozone for the upgrade, second, backup OM and SCM metadata, and lastly, upgrade Cloudera parcels.

About this task

Ozone will be brought to a read-only state before the upgrade so the OMs can synchronise before the upgrade.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Click Actions
5. Click Prepare for Upgrade.

Note: You must run the Prepare for Upgrade option for Ozone before backing up OM and SCM metadata.

Backing up Ozone

You must shutdown the cluster and take the backup of Ozone Manager (OM) and Storage Container Manager (SCM) metadata.

About this task

Note: To locate the hostnames required to backup OM and SCM, open the Cloudera Admin Console, go to the Ozone service, and click the Instances tab.

Procedure

1. On each OM, copy the directories indicated by the ozone.om.db.dirs and ozone.om.ratis.storage.dir configuration keys to the backup location by running the command `cp -r <config_directory> <backup_directory>`.
2. On each SCM, copy the directories indicated by the ozone.scm.db.dirs and ozone.scm.ha.ratis.storage.dir config keys to the backup location by running the command `cp -r <config_directory> <backup_directory>`.



Note: You must take a backup of ozone.scm.ha.ratis.storage.dir only if ozone.scm.ratis.enable is set to *TRUE*.

Upgrading Ozone parcels

You must upgrade the Cloudera parcels which in turn updates Ozone. To complete the upgrade of Ozone services, you must finalize the upgrade. This allows you to access the latest features available for this release.

Procedure

1. Using Cloudera Manager, upgrade the Cloudera parcels. This upgrades Ozone as well.
2. Click the Finish Upgrade option. In this state you can read and write to ensure that the Ozone cluster is working as expected. At this stage, you can decide to finalize the upgrade or downgrade to the previous version.
3. Click the Finalize Upgrade option. At this stage, the cluster will be in the read-only state. After sufficient DataNodes finalize to serve writes, the cluster will leave the read-only state.



Note: After running the Finalize Upgrade command, it is not possible to downgrade.

Ozone S3 Multitenancy overview (Technical Preview)

Apache Ozone now supports the multi-tenancy feature. This feature enables Ozone to compartmentalize the resources and create multiple tenants.

Technical Preview: This is a technical preview feature and considered under development. Do not use this in your production systems. To share your feedback, contact Cloudera Support by logging a case on our Cloudera Support Portal. Technical preview features are not guaranteed troubleshooting guidance and fixes.

You can access multiple S3-accessible Ozone volumes available over AWS S3 using CLI or APIs. You can control each of these volumes with Ozone administrator privileges or tenant administrator privileges. You can use Apache Ranger to control the volume, bucket and key access.

Each tenant by default has a volume assigned. An administrator can provide the volume access to a user. An Access ID & Secret Key pair is generated for every user to access the volume. An Ozone administrator can then assign one or more tenant users with the tenant administrator privilege in a tenant, so these tenant administrators can assign and revoke users from the tenant without involving the Ozone administrators.

Prerequisites to enable S3 Multitenancy

Before you proceed to enable the feature, you must understand the mandatory prerequisites.

- To have a secure cluster, you must enable Kerberos Authentication. For more information, see Securing the cluster using Kerberos.

- You must perform a one-time configuration change in Ranger UserSync to add an om user or short username that Ozone Manager is using for Kerberos authentication to the ranger.usersync.whitelist.users.role.assignment.rules configuration.

  Note: This is no longer necessary after Ranger allows service admin users to create, update, and delete Ranger roles.

- You must have a minimum of one S3 Gateway setup in order to access the tenant buckets with S3 API. For more information, see Using Ozone S3 gateway.
- You can create additional Ozone policies using Ranger, For more information, see Configure a resource-based policy.

Enabling S3 Multi-Tenancy

You must perform the following steps to enable the S3 multi-tenancy feature.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Go to Configurations
5. Search for Enable Ozone S3 Multi-Tenancy and select the checkbox
6. Click Save Changes
7. Restart the Ozone service

Tenant Commands

After setting up the S3 Multi-Tenancy feature, you can create and list tenants, assign users to tenants and also assign admin privileges, revoke admin access or even delete a tenant, and so on.

The following commands assume that the cluster is Kerberized and Ranger enabled.

Note: If you have enabled Ozone Manager HA on the Ozone service, then you must append `--om-service-id=` to the commands.

Creating a tenant

To create a new tenant in the Ozone cluster, you must have cluster admin privileges defined in ozone.administrators configuration. When you create a tenant, a volume with the same name is created. However, tenant name and volume name must be the same and tenant volume cannot be changed after the tenant is created.

To create a new tenant, run `ozone tenant [--verbose] create <TENANT_NAME>`

Listing a tenant

To list all tenants in an Ozone cluster, run `ozone tenant list [--json]`

Assiging a user to a tenant

Only an Ozone cluster administrator can assign the first user to a tenant. After the first user gets admin privileges, the first user can create and assign new users. A user can be assigned multiple tenants.

To assign a user to a tenant, run `ozone tenant [--verbose] user assign <USER_NAME> --tenant=<TENANT_NAME>`

Assigning a user as a tenant admin

Only an Ozone cluster administrator can assign the first user to a tenant along with access key ID/secret pair. After the first user gets admin privileges, the first user can create and assign new users. A user can be assigned multiple tenants.

Both delegated and non-delegated tenant admins can assign and revoke tenant users from their tenant. However, only a delegated admin can assign and revoke the tenant admins from a tenant.

You can be a tenant admin in multiple tenants. However, you will be assigned different access IDs under each tenant.

To assign a user as a tenant admin, the current logged-in user must have Ozone cluster administrator or tenant delegated administrator privilege, run `ozone tenant user assignadmin <ACCESS_ID> [-d|--delegated] --tenant=<TENANT_NAME>`

Listing users in a tenant

To list users in a tenant, run `ozone tenant user list [--json] <TENANT_NAME>`

Getting tenant user info

To get tenant user's information, run `ozone tenant user info [--json] <USER_NAME>`

Revoking a tenant admin

To revoke a tenant admin, run `ozone tenant [--verbose] user revokeadmin <ACCESS_ID>`

Revoking user access from a tenant

To revoke the user access from a tenant, run `ozone tenant [--verbose] user revoke <ACCESS_ID>`

Deleting a tenant

The tenant must be empty and all admin user access revoked before deleting a tenant. This is a safety design to ensure that even if a tenant is deleted, the volume created for the tenant is intact.

To delete a tenant, run `ozone tenant [--verbose] delete <TENANT_NAME>`



Note: After a successful tenant delete command, the tenant information is removed from the Ozone Manager database and default tenant policies are removed from Ranger. However, the volume itself along with its data is not removed. An administrator can delete a volume manually using CLI.

Multi Protocol Aware System overview

The overview helps you to understand Ozone file system support, differences between flat namespace and hierarchical namespace, different bucket layouts, and their use cases.

Ozone natively provides Amazon S3 and Hadoop Filesystem compatible endpoints and is designed to work seamlessly with enterprise scale Data Warehousing, Batch Analytics, Machine Learning, Streaming Workloads, and so on. The prominent use cases based on the integration with storage service are mentioned below:

- Ozone as a pure S3 object store semantics
- Ozone as a replacement filesystem for HDFS to solve the scalability issues
- Ozone as a Hadoop Compatible File System (HCFS) with limited S3 compatibility. For example, for key paths with "/" in it, intermediate directories will be created.
- Multiprotocol access - Interoperability of the same data for various workloads.

Upgrading this feature

You must first upgrade Ozone and perform the pre and post finalization steps to use this feature.

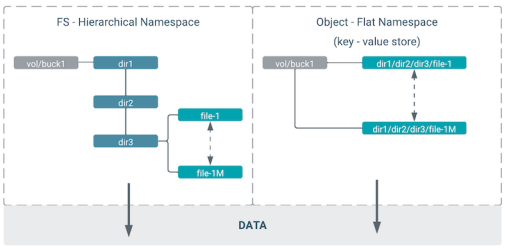
Procedure

1. Pre-Finalization Phase: You can only create buckets with a LEGACY layout. If any client (old or new) tries to create a new bucket with OBJECT_STORE or FILE_SYSTEM_OPTIMIZED layout, this request is blocked.
2. Post-Finalization Phase:
   a) New Clients: Full Bucket Layout feature is available.
   b) Old Clients: You cannot interact with any buckets that are not in LEGACY layout. This means they cannot talk to FSO or OBS buckets. Ozone displays the UNSUPPORTED_OPERATION exception in all such cases. For example, attempts to create directories and keys, list status, read bucket info, and so on will also display an UNSUPPORTED_OPERATION exception.

Files and Objects together

Bucket Layout concept is now introduced in Ozone that helps you with the unified design representing files, directories, and objects stored in a single system.

A single unified design represents files, directories, and objects stored in a single system. Ozone performs this by introducing the bucket layout concept in the metadata namespace server. With this, a single Ozone cluster with the capabilities of both Hadoop Compatible File System (HCFS) and Object Store (like Amazon S3) features by storing files, directories, objects and buckets efficiently. Also, the same data can be accessed using various protocols.



Bucket Layout

Apache Ozone now supports bucket layout feature. This helps you in categorising different Ozone buckets like FSO, OBS, and Legacy.

Apache Ozone object store now supports a multi-protocol aware Bucket Layout. The purpose is to categorize Ozone Bucket based on the prominent use cases:

- FILE_SYSTEM_OPTIMIZED (FSO) Bucket
  - Hierarchical FileSystem namespace view with directories and files similar to HDFS.
  - Provides high performance namespace metadata operations similar to HDFS.
  - Provides capabilities to read/write using Amazon S3.
- OBJECT_STORE (OBS) Bucket - Provides a flat namespace (key-value) similar to Amazon S3.
- LEGACY Bucket - Represents existing pre-created buckets for smooth upgrades from previous Ozone version to the new Ozone version



You can create FSO/OBS/LEGACY buckets using following shell commands. You can specify the bucket type in the layout parameter.

- `$ ozone sh bucket create --layout FILE_SYSTEM_OPTIMIZED /s3v/fso-bucket`
- `$ ozone sh bucket create --layout OBJECT_STORE /s3v/obs-bucket`
- `$ ozone sh bucket create --layout LEGACY /s3v/bucket`

This table explains the differences between Bucket Type and Client Interface

Bucket Type	**S3 Compatible Interface**	**ofs**	**o3fs (Deprecated, not recommended)**
	URL Scheme: http://bucket.host:9878/	URL Scheme: ofs://om-id/volume/bucket/key	URL Scheme: o3fs://bucket.volume.om-id/key
FSO	Supports Read, Write, and Delete operations	Supports Read, Write, and Delete operations	Supports Read, Write, and Delete operations
OBS	Supports Read, Write, and Delete operations	Unsupported	Unsupported

Note: FSO and OBS are accessible only on Cloudera Base on premises 7.1.8 and higher.

Ozone FS namespace optimization with prefix

Ozone now supports FS namespace optimization with prefix that provides atomicity and consistency in renaming and deleting files and subdirectories under a directory. Ozone now handles partial failures and performance is now deterministic.

FSO feature helps in performing the rename or delete metadata operations for the directories which have large sub-trees or sub-paths. With this feature, Ozone handles partial failures and provides atomicity and consistency in renaming and deleting each and every file and subdirectory under a directory. Performance is deterministic now and is similar to HDFS, especially for the delete and rename metadata operations, and if you are running the Spark or Hive like big data queries.

Highlights of this feature:

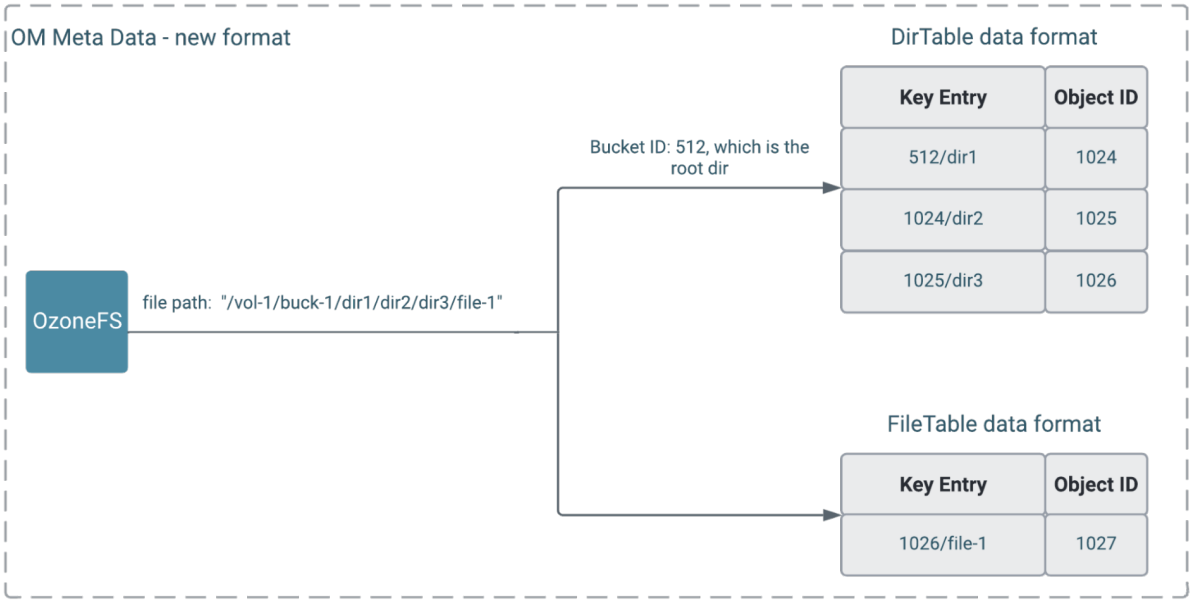
- Provide an efficient Hierarchical FileSystem Namespace view with intermediate directories similar to HDFS.
- Support for Atomic Rename and Deletes. This helps Hive, Impala, and Spark for job and task commits.
- Strong consistency guarantees without any partial results in case of directory rename or delete failures.
- Rename, move, and recursive directory delete operations should have deterministic performance numbers irrespective of the large set of subpaths (directories/files) contained within it.

Changes you can observe by using this feature:

- Apache Hive drop table query, recursive directory deletion, and directory moving operations becomes faster and consistent without any partial results in case of any failure.
- Dropping a managed Impala table should be efficient without requiring O(n) RPC calls where n is the number of file system objects for the table.
- Job Committers of Hive, Impala, and Spark often rename their temporary output files to a final output location at the end of the job. The performance of the job is directly impacted by how quickly the rename operation is completed.
- ACL support through Apache Ranger.
- For more information on understanding the performance capabilities between Apache Ozone and S3 API and how to natively integrate workloads, see High Performance Object Store for Cloudera on premises and High Performance Object Store for Cloudera on premises.

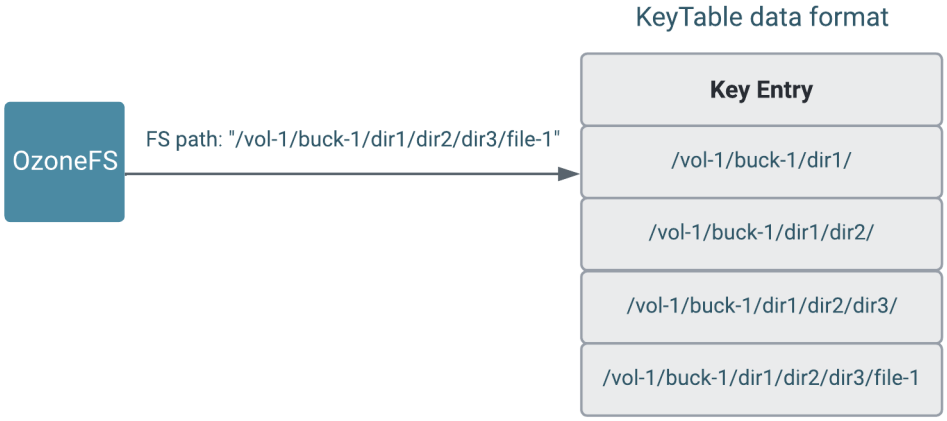
Metadata layout format

In the File System Optimized (FSO) buckets, OM metadata format stores intermediate directories into DirectoryTable and files into FileTable as shown in the below image. The key to the table is the name of a directory or a file prefixed by the unique identifier of its parent directory <parent unique-id>/<filename>



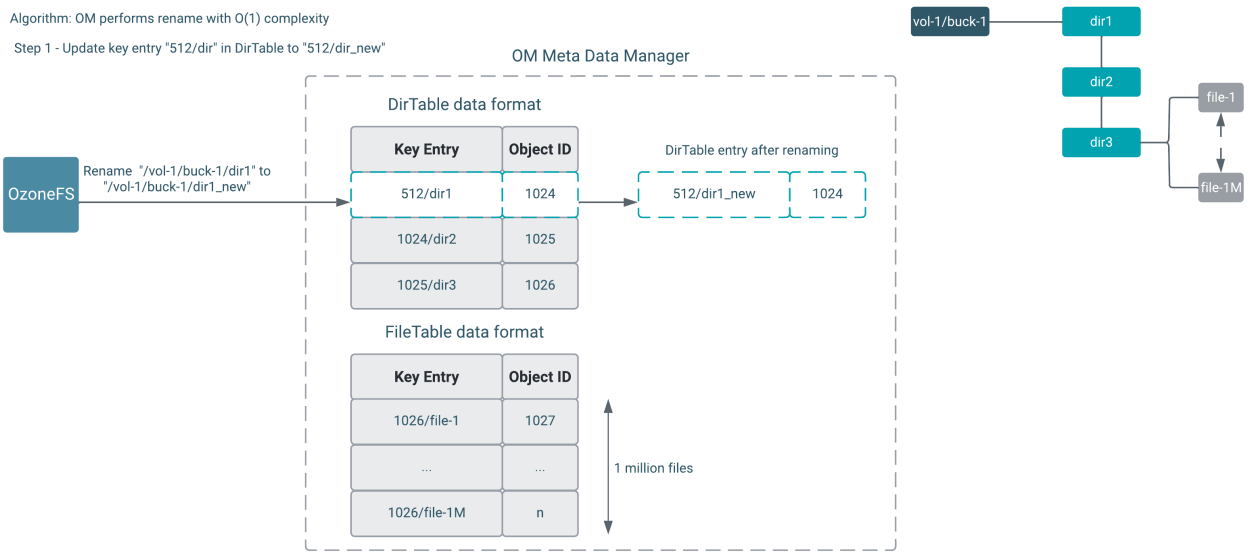
Delete and Rename Operation

Currently, in the Legacy Ozone file system to delete or rename dir 1, you have to delete or rename dir 1 in all the rows. This is expensive, time consuming, and not scalable.

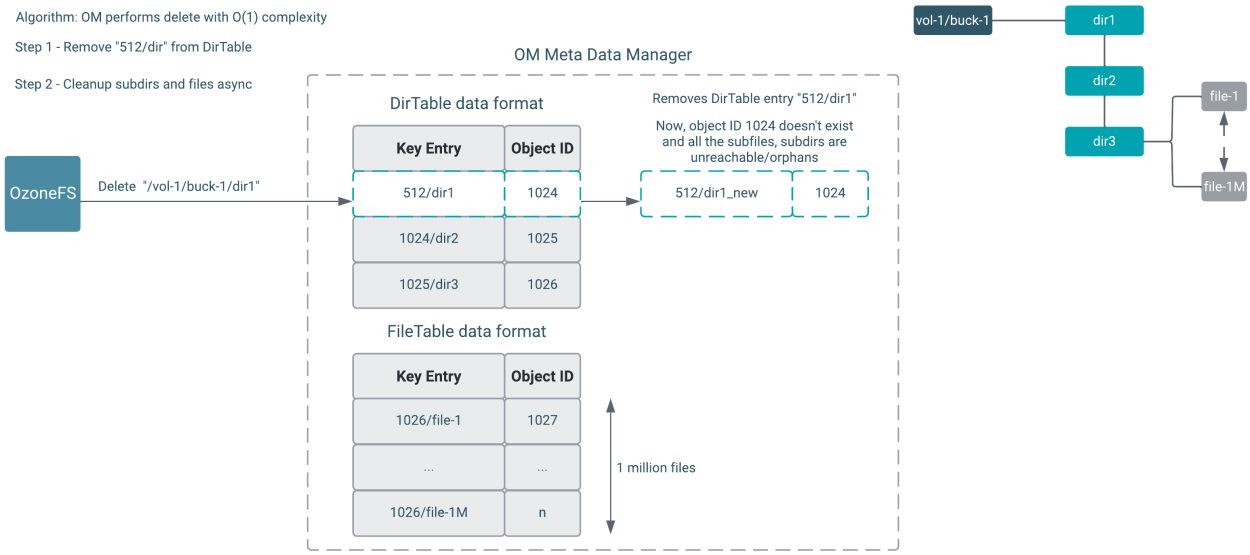


Now, there is an object ID allocated for every path created. Deleting or renaming dir 1 now updates all rows based on the Object ID. The images below explain how rename and delete operations work.

Rename



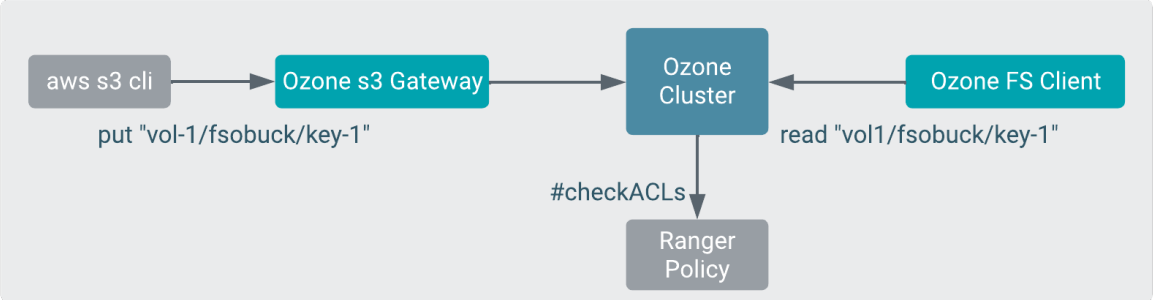
Deletes

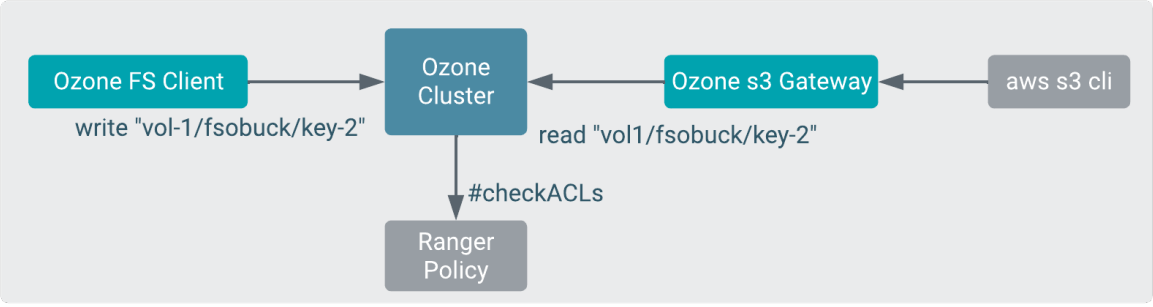


Interoperability Between S3 and FS APIs

FSO Bucket Layout supports interoperability of data for various use cases. For example, You can create an FSO Bucket type and ingest data into Apache Ozone using FileSystem API. The same data can be accessed through the Ozone S3 API (Amazon S3 implementation of the S3 API protocol) and vice versa.

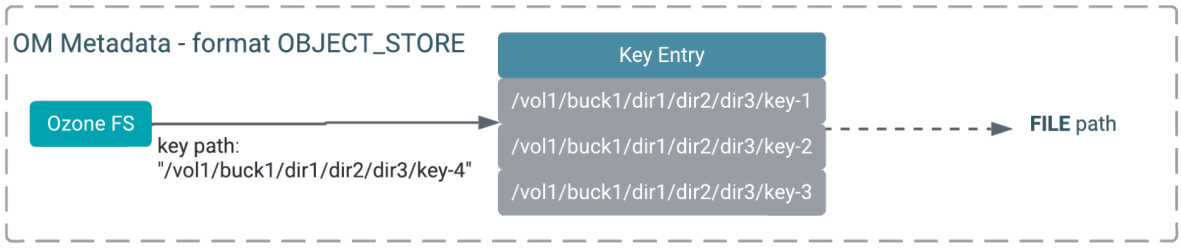
Multi-protocol client access - read/write operations using Ozone S3 and Ozone FS client.





OBS as Pure Object Store

OBS is the existing Ozone Manager metadata format which stores key entries with full path names, where the common prefix paths will be duplicated for keys like shown in the below diagram.



Configuration to create bucket with default layout

You must set the following configuration in Cloudera Manager to create a bucket with default layout.

About this task

In Cloudera Manager, you must configure ozone-site.xml to define the default value for bucket layout during bucket creation if the client has not specified the bucket layout argument. Supported values are OBJECT_STORE, FILE_SYSTEM_OPTIMIZED, and LEGACY.

By default, this configuration value is empty. Ozone will default to FILE_SYSTEM_OPTIMIZED bucket layout if it finds an empty configuration value. You must add the below property and provide the value.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Go to Configurations
5. Search for the ozone.default.bucket.layout configuration
   a) Select the bucket type
   b) Click Save Changes
   c) Restart the Ozone service

Performing Bucket Layout operations in Apache Ozone using CLI

Run the below commands to get an understanding of the basic operations like create, list, move, read, write, and delete.

Procedure

1. Creating FSO and OBS buckets using Ozone Shell:
   a) `ozone sh bucket create --layout FILE_SYSTEM_OPTIMIZED /s3v/fso-bucket`
   b) `ozone sh bucket create --layout OBJECT_STORE /s3v/obs-bucket`

   Note: The server-side default layout will be used for bucket creation if the --layout is not specified. For more information, see Configuration to create bucket with default layout documentation.
2. Bucket Info:
   a) `ozone sh bucket info /s3v/fso-bucket`
   b) `ozone sh bucket info /s3v/obs-bucket`

FSO operations

You can run the following commands for performing FSO operations.

Procedure

1. Creating directories inside FSO buckets:
   a) `ozone fs -mkdir -p ofs://ozone1/s3v/fso-bucket/dir1/dir2/dir3/`
   b) `ozone fs -mkdir -p ofs://ozone1/s3v/fso-bucket//aa///bb//cc///`

   In FSO, bucket paths will be normalized
2. Listing FSO bucket `ozone fs -ls -R ofs://ozone1/s3v/fso-bucket/`
3. Creating some files inside FSO buckets
   a) `ozone fs -touch ofs://ozone1/s3v/fso-bucket/dir1/dir2/dir3/file1`
   b) `ozone fs -touch ofs://ozone1/s3v/fso-bucket/dir1/dir2/dir3/file2`
   c) `ozone fs -touch ofs://ozone1/s3v/fso-bucket/aa/bb/cc/abc_file3`
4. Listing FSO bucket `ozone fs -ls -R ofs://ozone1/s3v/fso-bucket/`
5. Move command on a file, which Ozone internally does a renaming operation `ozone fs -mv ofs://ozone1/s3v/fso-bucket/dir1/dir2/dir3/file2 ofs://ozone1/s3v/fso-bucket/dir1/dir2/`
6. Move command on a directory, which internally does a renaming operation `ozone fs -mv ofs://ozone1/s3v/fso-bucket/aa/ ofs://ozone1/s3v/fso-bucket/new_aa`
7. Listing FSO bucket `ozone fs -ls -R ofs://ozone1/s3v/fso-bucket/`

Multi Protocol Access operations using AWS Client

You can run the following commands to run multi protocol access operations using the AWS client.

Procedure

1. Writing a new file to FSO bucket
   a) `aws s3 cp --endpoint-url http://0.0.0.0:9878 ozone.txt s3://fso-bucket/dir1/dir2/dir3/awsfile1`
   b) `aws s3 cp --endpoint-url http://0.0.0.0:9878 ozone.txt s3://fso-bucket/dir1/dir2/dir3/awsfile2`
2. Reading from Ozone Bucket
   a) `aws s3api --endpoint http://0.0.0.0:9878 get-object --bucket fso-bucket --key /dir1/dir2/dir3/awsfile1 ./ozone_doc`
   b) `cat ./ozone_doc`
3. Listing Bucket Objects `aws s3api --endpoint-url http://0.0.0.0:9878 list-objects --bucket fso-bucket`
4. Deleting a file `aws s3 rm --endpoint-url http://0.0.0.0:9878 s3://fso-bucket/dir1/dir2/dir3/awsfile1`
5. Following operations using Ozone FS commands
   a) Listing directory `ozone fs -ls -R ofs://ozone1/s3v/fso-bucket/`
   b) Displaying the content of file `ozone fs -cat ofs://ozone1/s3v/fso-bucket/dir1/dir2/dir3/awsfile2`
6. Following operations using Ozone Shell commands
   a) Listing Keys `ozone sh key list /s3v/fso-bucket/`
7. Following operations using Ozone FS commands
   a) Deleting a file `ozone fs -rm -skipTrash ofs://ozone1/s3v/fso-bucket/dir1/dir2/dir3/awsfile2`
   b) Deleting a directory `ozone fs -rm -R -skipTrash ofs://ozone1/s3v/fso-bucket/dir1/`
   c) Listing directories (dir1 should not exist) `ozone fs -ls -R ofs://ozone1/s3v/fso-bucket/`

Object Store operations using AWS client

You can run the following commands for performing OBS operations using AWS Client.

Procedure

1. Creating a bucket `aws s3api --endpoint-url http://0.0.0.0:9878 create-bucket --bucket=obs-s3bucket`

   Note: The aws create-bucket API will create an OBS bucket.

2. Bucket Info `ozone sh bucket info /s3v/obs-s3bucket`
3. Writing a file to bucket
   a) `aws s3 cp --endpoint-url http://0.0.0.0:9878 ozone.txt s3://obs-s3bucket/dir1/dir2/dir3/##awsfile1`
   b) `aws s3 cp --endpoint-url http://0.0.0.0:9878 ozone.txt s3://obs-s3bucket/dir1/dir2/dir3/##awsfile2`
4. Reading the above file from bucket
   a) `rm -rf /tmp/sample.txt`
   b) `ozone sh key get /s3v/obs-s3bucket/dir1/dir2/dir3/##awsfile1 /tmp/sample.txt`
   c) `cat /tmp/sample.txt`

5. Listing bucket object `aws s3api --endpoint-url http://0.0.0.0:9878 list-objects --bucket obs-s3bucket`
6. Deleting a key `aws s3 rm --endpoint-url http://0.0.0.0:9878 s3://obs-s3bucket/dir1/dir2/dir3/awsfile1`

Ozone Ranger policy

Using the Ozone Ranger policy integration, you can set new Ozone Ranger policies.

Procedure

1. Ranger permissions for OBJECT_STORE buckets
   a) You must configure policy on a resource key path
   b) Wild-Card: Keys starting with key path keyRoot/key*
2. Ranger permissions for FILE_SYSTEM_OPTIMIZED buckets
   a) Configure policy on resource path, which can be at the level of a specific file or directory path component.
   b) Wild Cards: You can configure a policy for all sub-directories or sub-files using wildcards. For example, /root/app*. For more information, see performance optimized authorization approach for rename and recursive delete operations in the Ranger Ozone plugin.



Note: For more information on setting Ozone Ranger policies, see Ranger Ozone Integration.

Ozone S3 Security Token Service overview

The Ozone Security Token Service (STS) provides temporary, short-lived S3 credentials (AccessKeyId, SecretAccess Key, SessionToken) through the AWS-compatible AssumeRole API. It enables secure, fine-grained access to Ozone buckets without exposing long-term principal credentials or Kerberos keytabs.

Typically, Ozone S3 credentials are tied to a Kerberos identity. STS introduces a programmatic method to obtain temporary credentials, which include an access key, a secret key, and a session token. These credentials inherit permissions from a Ranger role and can optionally be further restricted by an inline IAM session policy.

Key functionalities include:

- **Dedicated endpoint**: Runs on a dedicated port (separate from the S3 Gateway port).
- **API support**: Implements the AWS AssumeRole specification.
- **Authorization**: Relies on Apache Ranger for role delegation and permission checking.

How Ozone STS works

The temporary credential generation and usage flow operates through the following steps:

1. **Request**: A user with long-term credentials calls AssumeRole API on the STS endpoint.
2. **Validation**: Ozone validates the request signature and queries Apache Ranger to confirm the caller is permitted to assume the target role.
3. **Policy evaluation**: If an optional inline IAM session policy is provided, Ozone intersects its permissions with the target Ranger role when the token is used.
4. **Credential issuance**: Ozone returns the following short-lived credentials:
   - AccessKeyId - Starting with the ASIA access key prefix
   - SecretAccessKey
   - SessionToken

5. **Access**: The client performs standard S3 operations using the temporary keys and including the SessionToken in the x-amz-security-token header.

Important: Temporary credentials have a lifespan ranging from 15 minutes to 12 hours (900 to 43,200 seconds). If the DurationSeconds parameter is not specified, the token's expiration defaults to 3600 seconds (1 hour), aligning with the AWS specification.

Enabling Ozone S3 Security Token Service

Learn about the prerequisites and the end-to-end procedure required to enable the Ozone Security Token Service (STS).

Before you begin

Before enabling and using STS, ensure your environment meets the following requirements:

1. A functioning, secure Ozone cluster must be available with both Kerberos and Apache Ranger enabled.
2. The Ranger Ozone plugin must be installed on the Ozone Manager (OM) and S3 Gateway (S3G) with the Rang erOzoneAuthorizer configured.
3. The calling user (that is, the service principal) must possess permanent S3 credentials. You can obtain these using the following command:

```
kinit -kt /cdep/keytabs/[***YOUR_S3_USERNAME***].keytab [***HOSTNAME***]
 && ozone s3 getsecret
```

4. Ozone volumes, buckets, and keys are created as needed for STS token access.
5. The Ranger Administrator must configure the following:
   - Create a Ranger user corresponding to the service principal in Ozone.
   - Create a Ranger role for each role that can be assumed using STS.
   - Grant the calling user the assume_role permission on that role.
   - Grant the role the necessary resource permissions and actions (volume/bucket/key) using Ranger resource policies.

   Important: Ranger configuration changes may take up to 30 seconds to take effect in Ozone.
6. The role name provided in the RoleArn during an API call must exactly match the Ranger role name. For example:

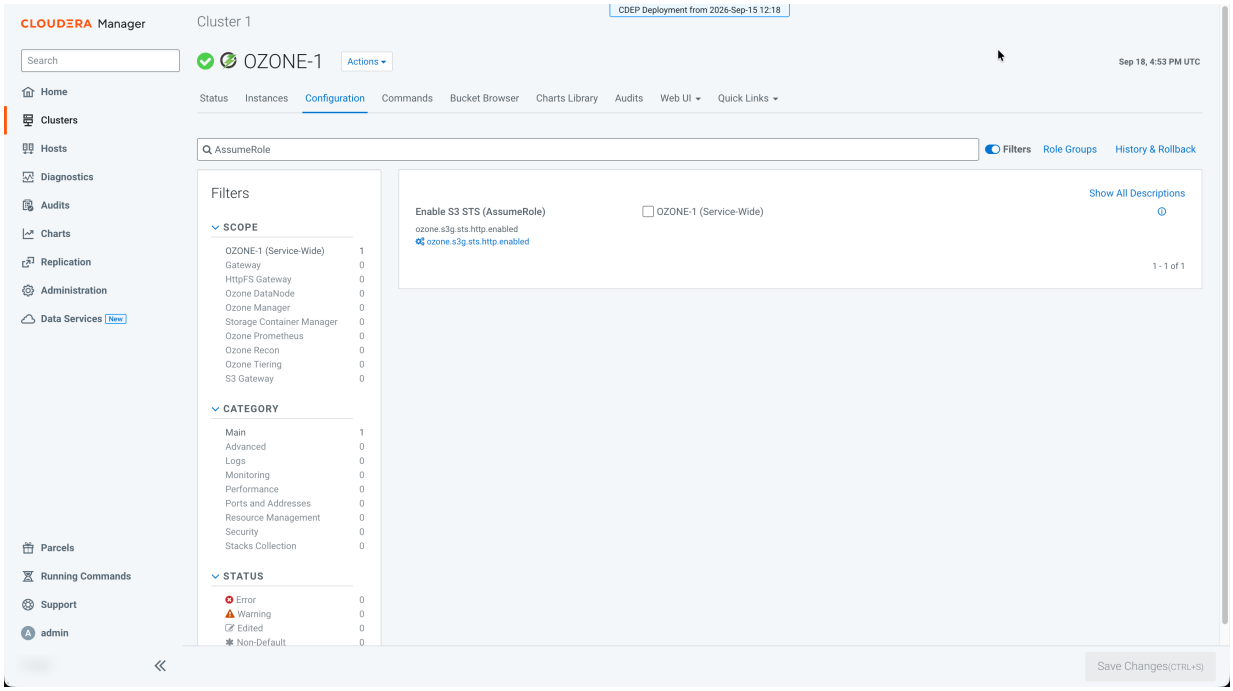
```
arn:aws:iam::123456789012:role/my-data-reader-role
                            ^^^^^^^^^^^^^^^^^^^^
                            Ranger role name
```

7. To configure Ranger policies with curl, set these environment variables:
   - RANGER_URL: Ranger Admin base URL (for example, http s://*[***RANGER_HOST***]*:*[***PORT_NUMBER***]*)
   - RANGER_SERVICE_NAME: Ozone Ranger service name (example, cm_ozone)
   - RANGER_ADMIN_USERNAME: Ranger Admin username
   - RANGER_ADMIN_PASSWORD: Ranger Admin password

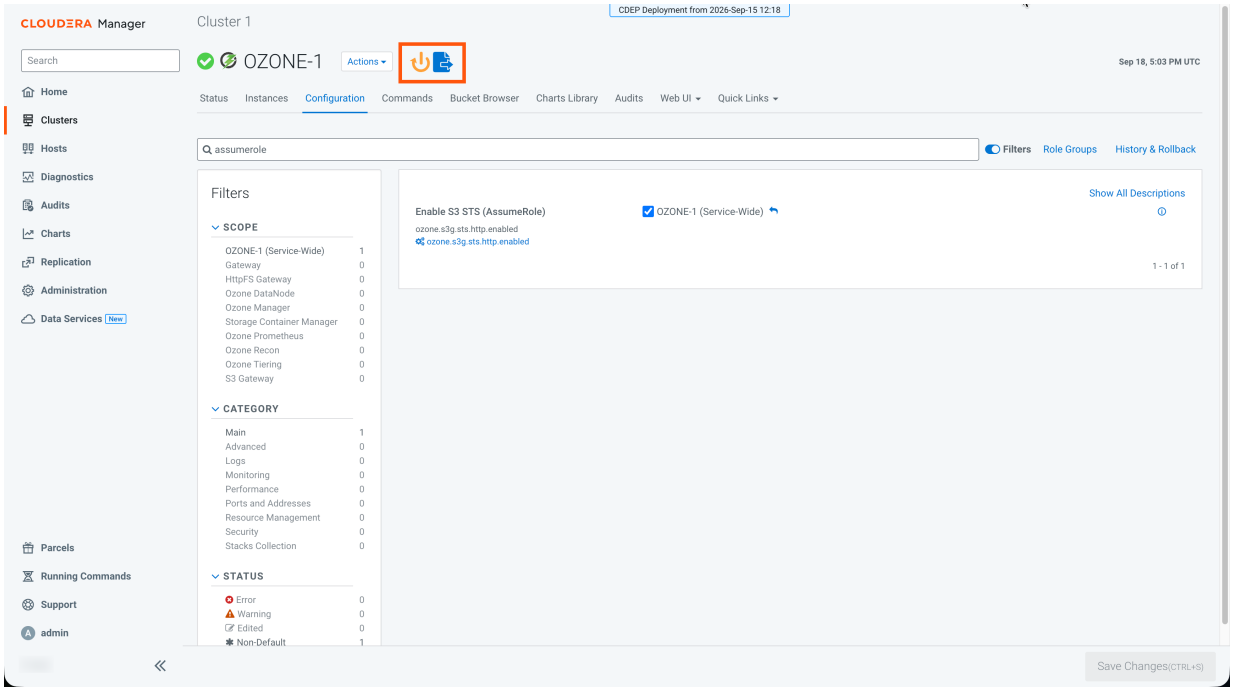
Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the Ozone service.
4. Click the Configuration tab.

5. Search for assumeRole.
6. In the Enable S3 STS (AssumeRole) configuration, select the checkbox next to OZONE-1 (Service-Wide).



7. Click Save Changes.
8. Wait about 30 seconds for the Stale Configuration icon  to appear next to the Actions dropdown (near the OZONE-1 label at the top of the page).



9. Click the Stale Configuration: Restart needed icon  .
10. In the **Stale Configurations** page, click Restart Stale Services.
11. In the **Review Changes** page, select the Regular Restart option.
12. Click Continue.
13. Click **Finish** once the restart completes.

Additional configurable properties

Learn about the advanced configuration parameters available for tuning the Ozone S3 Gateway Security Token Service (STS) endpoint. These properties allow administrators to customize network bindings and token revocation cleanup settings within ozone-site.xml to meet specific requirements.

The following properties can be adjusted in ozone-site.xml to customize the behavior of the STS endpoint on the S3 Gateway:

Table 1: Additional configurations

Property	Default value	Description
ozone.s3g.sts.http.enabled	false	Feature flag for the STS endpoint.
ozone.s3g.sts.http-port	9880	HTTP port for STS.
ozone.s3g.sts.https-port	9881	HTTPS port for STS.
ozone.s3g.sts.http-bind-host	0.0.0.0	HTTP bind host.
ozone.s3g.sts.https-bind-host	0.0.0.0	HTTPS bind host
ozone.om.sts.token.cleanup.service.interval	3h	Interval for cleaning revoked token entries older than 12 hours.
ozone.om.sts.token.cleanup.service.timeout	15m	Timeout for a cleanup run.

After enabling, the STS endpoint is available at one or more of the following URLs, depending on the ozone.http.policy value (HTTP_ONLY, HTTPS_ONLY, or HTTP_AND_HTTPS) and whether TLS is configured. By default, the value is HTTPS_ONLY:

```
http://[***S3G_HOST***]:[***HTTP_PORT***]
https://[***S3G_HOST***]:[***HTTPS_PORT***]
```

where,

- *[***HTTP_PORT***]* is the ozone.s3g.sts.http-port value.
- *[***HTTPS_PORT***]* is the ozone.s3g.sts.https-port value.

Example,

```
http://[***S3G_HOST***]:9880
https://[***S3G_HOST***]:9881
```

The S3 object API remains on the existing S3G port (typically 9879).

IAM Session Policy to Ranger permission or action mapping

When a caller passes a Policy parameter to AssumeRole, Ozone parses the AWS IAM session policy JSON and maps each supported S3 action to Ranger permissions and actions at the volume, bucket, and key levels. Ranger then authorizes access based on the intersection of the permissions and actions defined in the role and those defined in the session policy.

Supported session policy subset

Table 2: Supported session policy subset

IAM Element	Supported	Notes
Effect	Allow only	Deny and other effects are rejected.
Action	Supported S3 actions and wildcards: • s3:* • s3:Get* • s3:Put* • s3:List* • s3:Create* • s3:Delete*	Unknown actions are silently ignored, similar to AWS behavior (AssumeRole still succeeds, but credentials simply will not grant that action).
Resource	arn:aws:s3::: prefix or *	Other ARN prefixes are rejected.
Condition	StringEquals or StringLike with key s3:prefix	Only one condition operator is allowed per statement; it only applies to s3:ListBucket.
Policy Size	# 2048 characters	Matches the AWS AssumeRole limit.

S3 action to Ranger permissions and actions mapping

The following table outlines the required Ranger permissions and actions at each resource level for each S3 action when it appears in a session policy.

Note: Each applicable resource level must also include a matching action-matches condition for that S3 action (for example, PutObject at the volume, bucket, and key levels for object-scoped actions). If action-matches is omitted at a level, the ACL permission applies to every S3 action that requires that ACL at that level. For example, granting key-level READ without specifying action-matches authorizes both GetObject and GetObjectTagging. To maintain security, always specify the action at every applicable level.

Table 3: Ranger mapping table

S3 Action	Scope	Volume	Bucket	Key
s3:AbortMultipartUpload	Object	• Permission: READ • Action: AbortMultipartUpload	• Permission: READ • Action: AbortMultipartUpload	• Permission: WRITE • Action: AbortMultipartUpload
s3:CreateBucket	Bucket	• Permission: READ • Action: CreateBucket	• Permission: CREATE • Action: CreateBucket	—
s3:DeleteBucket	Bucket	• Permission: READ • Action: DeleteBucket	• Permission: DELETE • Action: DeleteBucket	—
s3:DeleteObject	Object	• Permission: READ • Action: DeleteObject	• Permission: READ • Action: DeleteObject	• Permission: DELETE • Action: DeleteObject
s3:DeleteObjectTagging	Object	• Permission: READ • Action: DeleteObjectTagging	• Permission: READ • Action: DeleteObjectTagging	• Permission: WRITE • Action: DeleteObjectTagging

S3 Action	Scope	Volume	Bucket	Key
s3:GetBucketAcl	Bucket	• Permission: READ • Action: GetBucketAcl	• Permission: READ, READ_ACL • Action: GetBucketAcl	—
s3:GetObject	Object	• Permission: READ • Action: GetObject	• Permission: READ • Action: GetObject	• Permission: READ • Action: GetObject
s3:GetObjectTagging	Object	• Permission: READ • Action: GetObjectTagging	• Permission: READ • Action: GetObjectTagging	• Permission: READ • Action: GetObjectTagging
s3:ListAllMyBuckets	Volume	• Permission: READ, LIST • Action: ListAllMyBuckets	—	—
s3:ListBucket	Bucket	• Permission: READ • Action: ListBucket	• Permission: READ, LIST • Action: ListBucket	• Permission: READ • Action: ListBucket
s3:ListBucketMultipartUploads	Bucket	• Permission: READ • Action: ListBucketMultipartUploads	• Permission: READ, LIST • Action: ListBucketMultipartUploads	—
s3:ListMultipartUploadParts	Object	• Permission: READ • Action: ListMultipartUploadParts	• Permission: READ • Action: ListMultipartUploadParts	• Permission: READ • Action: ListMultipartUploadParts
s3:PutBucketAcl	Bucket	• Permission: READ • Action: PutBucketAcl	• Permission: READ, READ_ACL, WRITE_ACL • Action: PutBucketAcl	—
s3:PutObject	Object	• Permission: READ • Action: PutObject	• Permission: READ • Action: PutObject	• Permission: CREATE, WRITE • Action: PutObject
s3:PutObjectTagging	Object	• Permission: READ • Action: PutObjectTagging	• Permission: READ • Action: PutObjectTagging	• Permission: WRITE • Action: PutObjectTagging

For s3:ListBucket, key-level READ is granted on the listed prefix (or * if no s3:prefix condition is present).

Simplifying bucket and key policies with All

You can often simplify bucket and key policies by granting All instead of individual access types such as write or delete. The action-matches condition restricts the specific S3 action to which the permission applies, meaning All with Action: PutObject will not broadly authorize unrelated operations at that resource level.

At the volume level, use specific access types such as read and list rather than All. Volume-level permissions apply even when a request has no mapped S3 action (similar to bucket-level and key-level permissions). Therefore, granting All on a volume could unintentionally allow destructive operations, such as volume deletion, when the role is used outside the S3 context.

Example: IAM permission policy mapped to Ranger policies

Suppose a role should support this IAM permission policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::reports/*"
  }]
}
```

The corresponding Ranger role needs matching policies at each level:

Level	Resource	Permission	Action-matches Policy Condition
Volume	s3v	READ	GetObject
Bucket	reports	READ or All	GetObject
Key	* (under reports)	READ or All	GetObject

When using All at bucket or key level, the action-matches: GetObject condition is what prevents unrelated S3 actions from being authorized at that level.

Examples:

- Ranger policy item for the key level (specific read permission):

```
{
  "accesses": [{ "type": "read", "isAllowed": true }],
  "roles": ["my-data-reader-role"],
  "conditions": [{ "type": "action-matches", "values": ["GetObject"] }]
}
```

- Equivalent simplified key-level policy using all:

```
{
  "accesses": [{ "type": "all", "isAllowed": true }],
  "roles": ["my-data-reader-role"],
  "conditions": [{ "type": "action-matches", "values": ["GetObject"] }]
}
```

Note: Without action-matches on the volume, bucket, and key policies, the same key-level read grant also authorizes GetObjectTagging and any other read-scoped S3 action at that path.

s3:prefix condition (ListBucket only)

If an IAM session policy or role permission policy statement includes a Condition, only s3:ListBucket from that statement takes effect (similar to AWS behavior).

Operator	Behavior
StringEquals	Exact prefix match; wildcard values in s3:prefix are ignored.
StringLike	Prefix pattern match; supports * wildcards

The caller supplies a matching prefix (and optionally a delimiter) for ListObjects or ListObjectsV2, or access is denied.

Ranger policy setup

Using Ranger, you can setup new Ozone policies that will help you to set right access permissions to Ozone S3 Security Token Service (STS). Learn how to configure Ranger and set up Ranger policy for STS.

Creating a user in Ranger

Learn how to create a service principal user in Ranger using the web console or CLI.

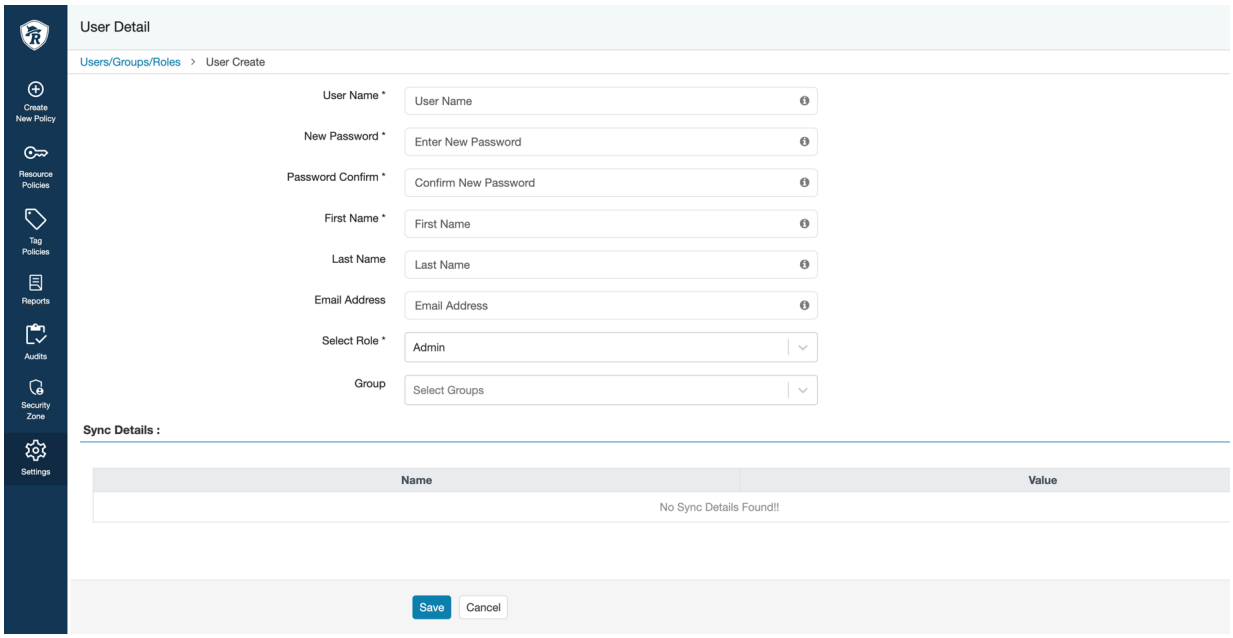
Using UI

Learn how to create a user in Ranger using UI.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the relevant Ranger cluster.
4. Click the Web UI dropdown menu and click one of the Ranger Admin Web UI. The Ranger log in page displays.
5. Sign in to Ranger. The Ranger home page displays.
6. In the left navigation, click Settings, and then click Users.
7. In the Users tab, click Add New User button. The **User Detail** page displays.

Figure 1: Create user in Ranger



8. Provide the following information:
   a. User Name: Enter the unique identifier as a user name. For example, svc-iceberg-rest-catalog.
   b. New Password: Enter the password.
   c. Password Confirm: Enter the same password that you have entered in the preceding field.
   d. First Name: Enter the first name of the user. For example, Iceberg REST.
   e. (Optional) Last Name: Enter the last name of the user. For example, Catalog.
   f. (Optional) Email Address: Enter the email id of the user. The email ID must be in *[***USER_NAME***]@[***YOUR_DOMAIN***]*.com. For example, svc-iceberg-rest-catalog@example.com.
   g. Select Role: Click the Select Role dropdown button and select User from the list.
   h. (Optional) Group: Click Group dropdown button and select the required group from the list.
9. Click Save.

Using CLI

Learn how to create a user in Ranger using CLI.

Before you begin

Ensure that you have the Administrator user name and password for Ranger.

Procedure

1. Open a terminal on your device.

2. Run the following `curl` command to create a user in Ranger:

```
curl -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "loginId": "[***UNIQUE_USERNAME_FOR_THE_NEW_USER***]",
    "name": "[***NAME_OF_THE_USER***]",
    "password": "[***PASSWORD_FOR_THE_USER***]",
    "firstName": "[***FIRST_NAME_OF_THE_USER***]",
    "lastName": "[***LAST_NAME_OF_THE_USER***]",
    "emailAddress":
 "[***UNIQUE_USERNAME_FOR_THE_NEW_USER***]@[***DOMAIN_NAME***].com",
    "userRoleList": ["ROLE_USER"],
    "userPermList": [
      { "moduleId": 1, "isAllowed": 1 },
      { "moduleId": 3, "isAllowed": 1 },
      { "moduleId": 7, "isAllowed": 1 }
    ]
  }' \
  "[***RANGER_URL***]/service/xusers/secure/users"
```

Example,

```
curl -u admin:Admin123 \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "loginId": "svc-iceberg-rest-catalog",
    "name": "svc-iceberg-rest-catalog",
    "password": "Password123!",
    "firstName": "Iceberg REST",
    "lastName": "Catalog",
    "emailAddress": "svc-iceberg-rest-catalog@example.com",
    "userRoleList": ["ROLE_USER"],
    "userPermList": [
      { "moduleId": 1, "isAllowed": 1 },
      { "moduleId": 3, "isAllowed": 1 },
      { "moduleId": 7, "isAllowed": 1 }
    ]
  }' \
  "https://ranger-host.example.com:6182/service/xusers/secure/users"
```

Creating a role in Ranger

Learn how to create a role in Ranger using the web console or CLI.

Using UI

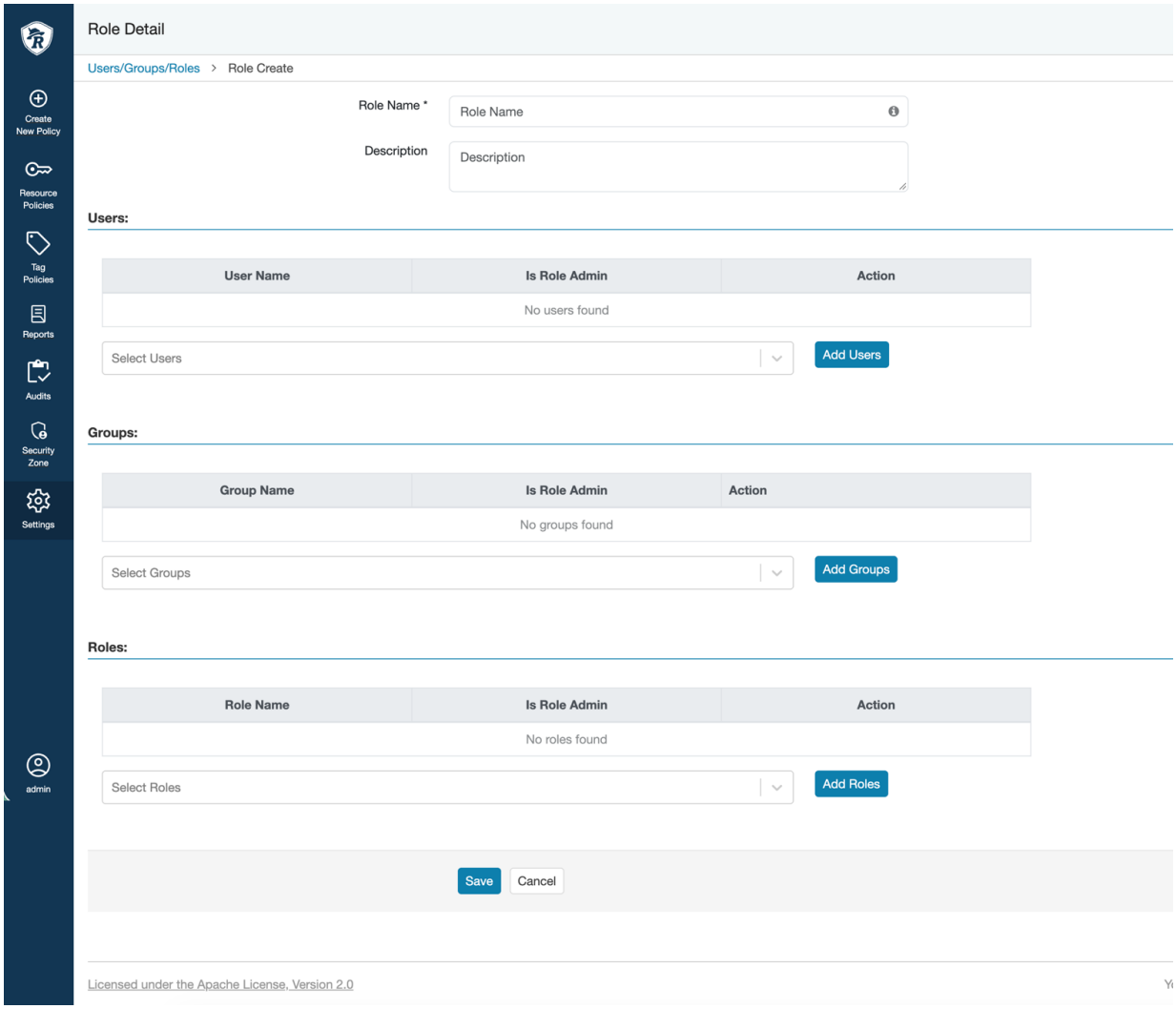
Learn how to create a role in Ranger using UI.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the relevant Ranger cluster.
4. Click the Web UI dropdown menu and click one of the Ranger Admin Web UI. The Ranger log in page displays.

5. Sign in to Ranger. The Ranger home page displays.
6. In the left navigation, click Settings, and then click Roles.
7. In the Roles tab, click the Add New Role button. The **Role Detail** page displays.

Figure 2: Create role in Ranger



8. Provide the following information:
   a. Role Name: Enter the unique identifier as a role name. For example, iceberg-data-all-access.
   b. Description : Provide a brief summary of the role's intended permissions or scope.
   c. (Optional) Group: Click Group dropdown button and select the required group from the list.
9. Click Save.

Using CLI

Learn how to create a role in Ranger using CLI.

Before you begin

Ensure that you have the Administrator user name and password for Ranger.

Procedure

1. Open a terminal on your device.

2. Run the following `curl` command to create a role in Ranger:

```
curl -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "name": "[***ROLE_NAME***]",
    "description": "[***ROLE_DESCRIPTION***]"
  }' \
  "[***RANGER_URL***]/service/roles/roles"
```

Example,

```
curl -u admin:Admin123 \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "name": "iceberg-data-all-access",
    "description": "Iceberg data all access"
  }' \
  "https://ranger-host.example.com:6182/service/roles/roles"
```

Creating a resource policy for Assume_Role in Ranger

Learn how to create a resource policy in Ranger to grant the Assume_Role permissions for the user.

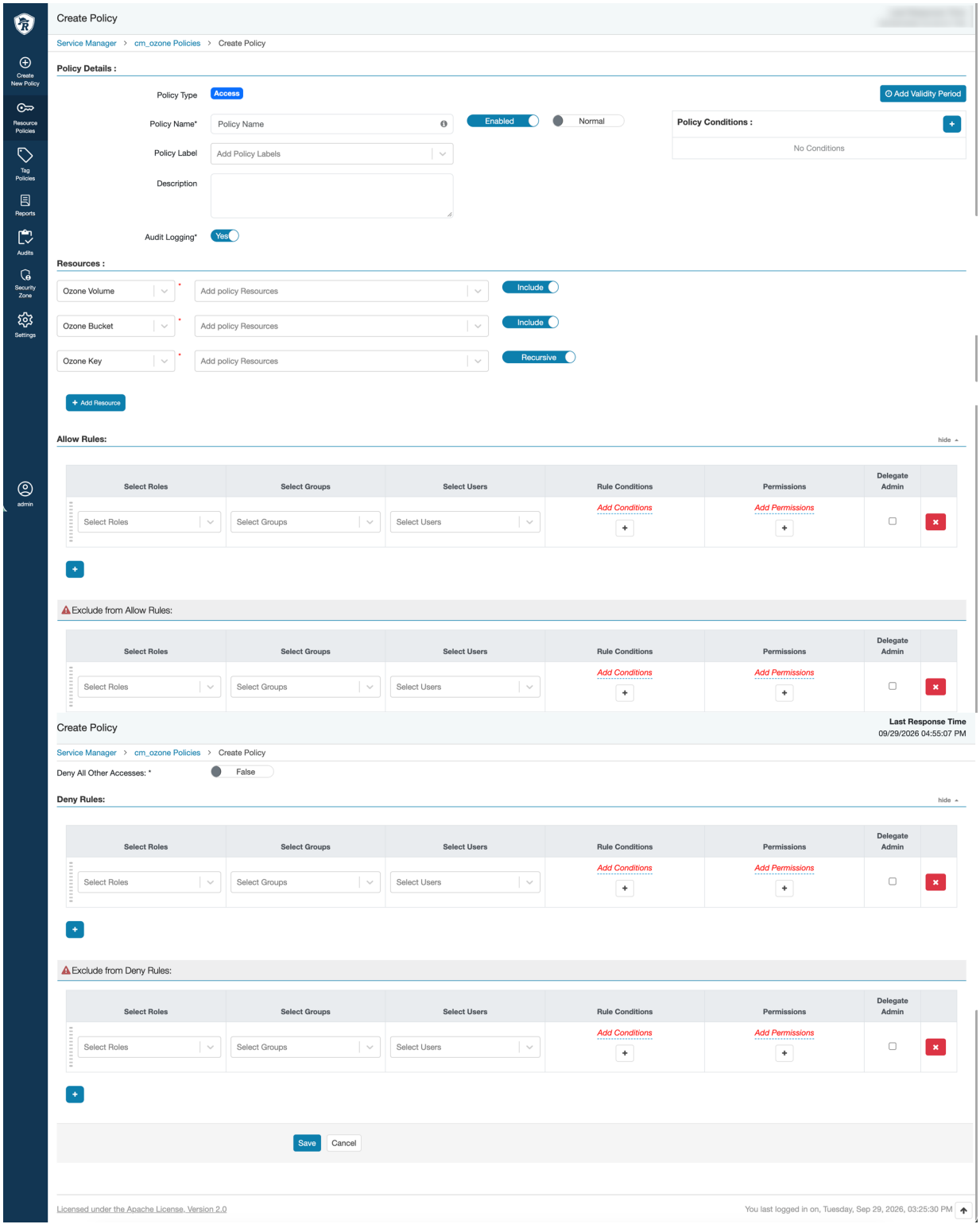
Using UI

Learn how to create a resource policy in Ranger to grant the Assume_Role permissions for the user using UI.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the relevant Ranger cluster.
4. Click the Web UI dropdown menu and click one of the Ranger Admin Web UI. The Ranger log in page displays.
5. Sign in to Ranger. The Ranger home page displays.
6. In the left navigation pane, select Resource Policies, and then select cm_ozone.
7. Click Add New Policy button. The **Create Policy** page displays.

Figure 3: Create new policy in Ranger



8. In the Policy Name box, enter a descriptive name for the policy. For example, iceberg assume role policy.
9. In the Resources section, click the Ozone Volume drop-down menu and select Role.
10. In the Add policy Resources text box, enter the role name (for example, iceberg-data-all-access), wait for the list to update, and select Create *[***ROLE_NAME***]*.
11. In the Allow Rules section, perform the following actions:
    a) Under Select Users, select the service user (for example, svc-iceberg-rest-catalog).

b) Under Permissions, select the plus sign (+).

c) Select the Assume_Role check box, and then select the  check mark at the bottom of the dialog box.

12. Click Save.

Using CLI

Learn how to create a resource policy in Ranger to grant the Assume_Role permissions for the user using CLI.

Before you begin

Ensure that you have the Administrator user name and password for Ranger.

Procedure

1. Open a terminal on your device.
2. Run the following `curl` command to create a resource policy in Ranger to grant the Assume_Role permissions for the user:

```
curl -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "isEnabled": true,
    "service": "[***RANGER_SERVICE_NAME***]",
    "name": "[***POLICY_NAME***]",
    "policyType": 0,
    "policyPriority": 0,
    "isAuditEnabled": true,
    "resources": {
      "role": {
        "values": ["[***TARGET_ROLE_NAME***]"],
        "isExcludes": false,
        "isRecursive": false
      }
    },
    "policyItems": [
      {
        "accesses": [
          { "type": "assume_role", "isAllowed": true }
        ],
        "users": ["[***SERVICE_USER_NAME***]"],
        "delegateAdmin": false
      }
    ],
    "serviceType": "ozone",
    "isDenyAllElse": false
  }' \
  "[***RANGER_URL***]/service/public/v2/api/policy"
```

Example,

```
curl -u admin:Admin123 \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
    --data '{
    "isEnabled": true,
    "service": "cm_ozone",
```

```
    "name": "iceberg assume role policy",
    "policyType": 0,
    "policyPriority": 0,
    "isAuditEnabled": true,
    "resources": {
      "role": {
        "values": [
          "iceberg-data-all-access"
        ],
        "isExcludes": false,
        "isRecursive": false
      }
    },
    "policyItems": [
      {
        "accesses": [
          {
            "type": "assume_role",
            "isAllowed": true
          }
        ],
        "users": [
          "svc-iceberg-rest-catalog"
        ],
        "delegateAdmin": false
      }
    ],
    "serviceType": "ozone",
    "isDenyAllElse": false
  }' \
  "https://ranger-host.example.com:6182/service/public/v2/api/policy"
```

Creating resource policies for volume, bucket, and key permissions

Learn how to create resource policies in Ranger to grant the permissions for the volume, bucket, and key.

Creating a resource policy for a volume using UI

Learn how to create a resource policy in Ranger to grant the permissions for a volume using UI.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the relevant Ranger cluster.
4. Click the Web UI dropdown menu and click one of the Ranger Admin Web UI. The Ranger log in page displays.
5. Sign in to Ranger. The Ranger home page displays.
6. In the left navigation pane, select Resource Policies, and then select cm_ozone.
7. Use one of the following methods to locate the policy associated with your target volume:
   - If a volume policy already exists (for example, S3_VOLUME_POLICY_FOR_OZONE_CANARY), enter the policy name in the search box and click the link in the Policy ID column to edit it.
   - If no policy exists for the volume,
     a. Click Add New Policy.
     b. Enter a Policy Name.
     c. In the Resources section, click the Ozone Volume drop-down menu and select Ozone Volume.
     d. In the Add policy Resources text box, enter the volume name (for example, s3v).
     e. Wait for few seconds and then select create *[***VOLUME_NAME***]* (for example, create s3v).
8. In the Allow Rules section, perform the following actions:

a) (Optional) Click the plus (+) button to add a new row.
b) Under Select Roles, click the Select Roles dropdown menu.
c) Select your target access role (for example, iceberg-data-all-access).
d) Under Permissions, click the plus sign (+).
e) Select the check boxes next to Read and List (optional), and then select the  check mark at the bottom of the dialog box.
f) Locate the row assigned to your target user or role (for example, iceberg-data-all-access).
g) Under Rule Conditions, click the plus (+) button to add a new condition.
h) Under Actions, select GetObject and PutObject.
i) Click Save in the dialog box.

9. Click Save.

Creating a resource policy for a volume using CLI

Learn how to create a resource policy in Ranger to grant the permissions for a volume using CLI.

Before you begin

Ensure that you have the Administrator user name and password for Ranger.

Procedure

1. Open a terminal on your device.

2. Fetch current policy definition:

```
curl -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
  --fail -H "accept: application/json" \
  "[***RANGER_URL***]/service/public/v2/api/servi
ce/[***RANGER_SERVICE_NAME***]/policy/[***POLICY_NAME***]" \
  > policy.json
```

Example,

```
curl -u admin:Admin123 \
  --fail -H "accept: application/json" \
  "https://ranger-host.example.com:6182/service/public/v2/api/service/cm_o
zone/policy/S3_VOLUME_POLICY_FOR_OZONE_CANARY" \
  > policy.json
```

3. Append new role access rule to the policy JSON:

```
jq '.policyItems += [{
  "accesses": [
    {"type": "read", "isAllowed": true},
    {"type": "list", "isAllowed": true}
  ],
  "roles": ["[***ROLE_NAME***]"],
  "users": [],
  "groups": [],
  "conditions": [{ "type": "action-matches", "values": ["GetObject", "Put
Object"] }],
  "delegateAdmin": false
}]' policy.json > policy.updated.json
```

Example,

```
jq '.policyItems += [{
  "accesses": [
```

```
    {"type": "read", "isAllowed": true},
    {"type": "list", "isAllowed": true}
  ],
  "roles": ["iceberg-data-all-access"],
  "users": [],
  "groups": [],
  "conditions": [{ "type": "action-matches", "values": ["GetObject", "Pu
tObject"] }],
  "delegateAdmin": false
}]' policy.json > policy.updated.json
```

4. Apply updated policy definition:

```
curl   -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
  --fail -H "Content-Type: application/json" \
  --request PUT \
  -H "accept: application/json" \
  --data @policy.updated.json \
  "[***RANGER_URL***]/service/public/v2/api/servi
ce/[***RANGER_SERVICE_NAME***]/policy/[***POLICY_NAME***]"
```

Example,

```
curl -u admin:Admin123 \
  --fail -H "Content-Type: application/json" \
  -H "accept: application/json" \
  --request PUT \
  --data @policy.updated.json \
  "https://ranger-host.example.com:6182/service/public/v2/api/service/cm
_ozone/policy/S3_VOLUME_POLICY_FOR_OZONE_CANARY"
```

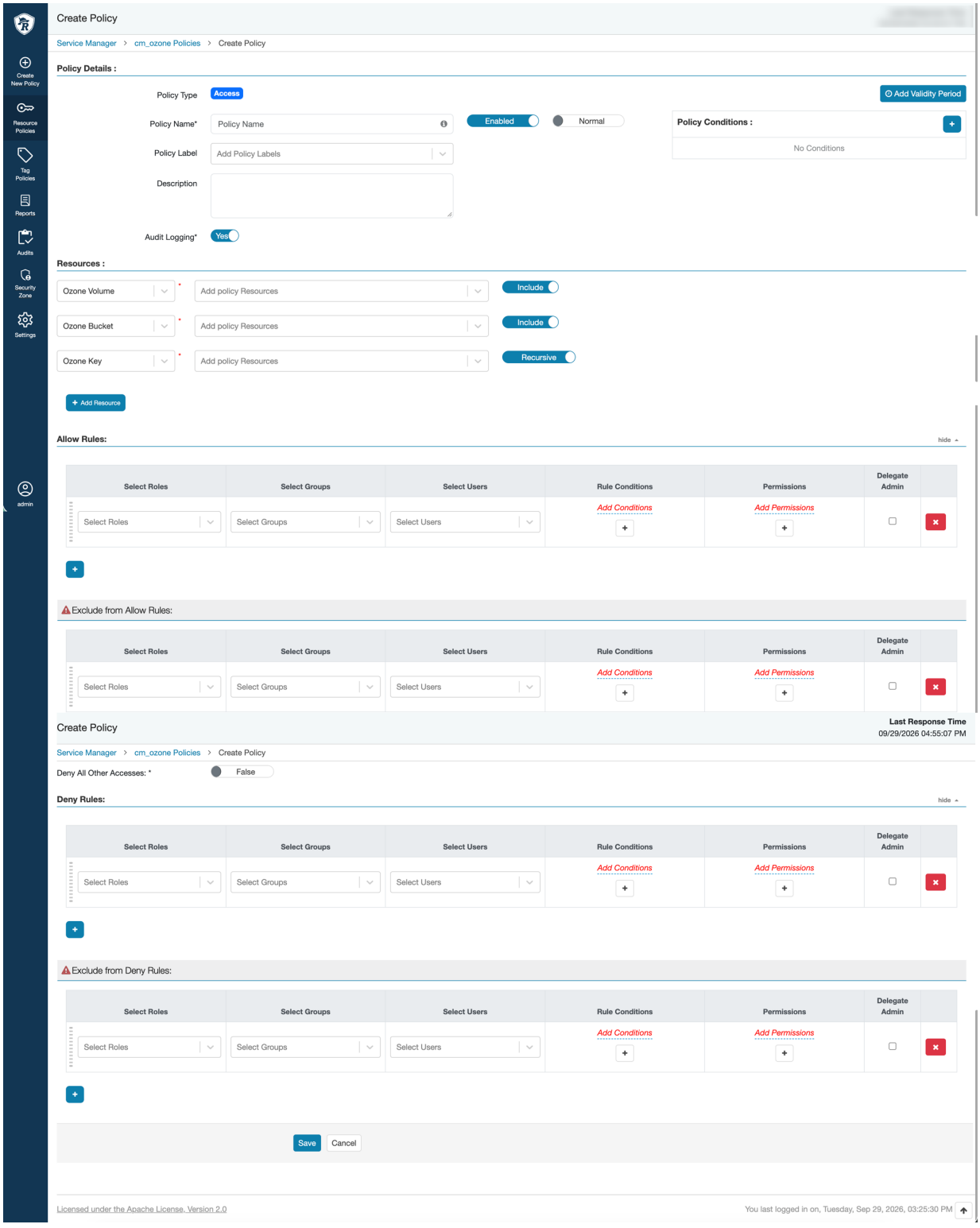
Creating a resource policy for a bucket using UI

Learn how to create a resource policy in Ranger to grant the permissions for a bucket using UI.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the relevant Ranger cluster.
4. Click the Web UI dropdown menu and click one of the Ranger Admin Web UI. The Ranger log in page displays.
5. Sign in to Ranger. The Ranger home page displays.
6. In the left navigation pane, select Resource Policies, and then select cm_ozone.
7. Click Add New Policy button. The **Create Policy** page displays.

Figure 4: Create new policy in Ranger



8. In the Policy Name box, enter a descriptive name for the policy. For example, iceberg bucket access.
9. In the Resources section, perform the following steps:
   a) Make sure that the Ozone Volume drop-down menu value is set to Ozone Volume.
   b) In the Add policy Resources text box, enter the volume name (for example, s3v).
   c) Wait for few seconds and then select create *[***VOLUME_NAME***]* (for example, create s3v).
   d) Make sure that the Ozone Bucket drop-down menu value is set to Ozone Bucket.

e) In the Add policy Resources text box, enter the bucket name (for example, iceberg).
f) Wait for few seconds and then select create *[***BUCKET_NAME***]* (for example, create iceberg).
g) Click the Ozone Key drop-down menu and select None. The Add policy Resources text box disables.

10. In the Allow Rules section, perform the following actions:

a) Under Select Roles, click the Select Roles dropdown menu,
b) Select your target access role (for example, iceberg-data-all-access).
c) Under Permissions, click the plus sign (+).
d) Select the check box next to All, and then select the  check mark at the bottom of the dialog box.
e) Locate the row assigned to your target user or role (for example, iceberg-data-all-access).
f) Under Rule Conditions, click the plus (+) button to add a new condition.
g) Under Actions, select GetObject and PutObject.
h) Click Save in the dialog box.

11. Click Save.

Creating a resource policy for a bucket using CLI

Learn how to create a resource policy in Ranger to grant the permissions for a bucket using CLI.

Before you begin

Ensure that you have the Administrator user name and password for Ranger.

Procedure

1. Open a terminal on your device.
2. Run the following `curl` command to create a resource policy in Ranger to grant the permissions for the bucket:

```
curl -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "isEnabled": true,
    "service": "[***RANGER_SERVICE_NAME***]",
    "name": "[***RANGER_POLICY_NAME***]",
    "policyType": 0,
    "policyPriority": 0,
    "isAuditEnabled": true,
    "resources": {
      "volume": {
        "values": ["[***VOLUME_NAME***]"],
        "isExcludes": false,
        "isRecursive": false
      },
      "bucket": {
        "values": ["[***BUCKET_NAME***]}"],
        "isExcludes": false,
        "isRecursive": false
      }
    },
    "policyItems": [
      {
        "accesses": [
          { "type": "all", "isAllowed": true }
        ],
        "roles": ["[***ROLE_NAME***]"],
```

```
        "conditions": [{ "type": "action-matches", "values": ["GetObject",
 "PutObject"] }],
        "delegateAdmin": false
      }
    ],
    "serviceType": "ozone",
    "isDenyAllElse": false
  }' \
  "[***RANGER_URL***]/service/public/v2/api/policy"
```

Example,

```
curl   -u admin:Admin123 \
  --location \
  --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "isEnabled": true,
    "service": "cm_ozone",
    "name": "iceberg bucket access",
    "policyType": 0,
    "policyPriority": 0,
    "isAuditEnabled": true,
    "resources": {
      "volume": {
        "values": ["s3v"],
        "isExcludes": false,
        "isRecursive": false
      },
      "bucket": {
        "values": ["iceberg"],
        "isExcludes": false,
        "isRecursive": false
      }
    },
    "policyItems": [
      {
        "accesses": [
          {
            "type": "all",
            "isAllowed": true
          }
        ],
        "roles": ["iceberg-data-all-access"],
        "conditions": [{ "type": "action-matches", "values": ["GetObject",
 "PutObject"] }],
        "delegateAdmin": false
      }
    ],
    "serviceType": "ozone",
    "isDenyAllElse": false
  }' \
  "https://ranger-host.example.com:6182/service/public/v2/api/policy"
```

Creating a resource policy for all keys in a bucket using UI

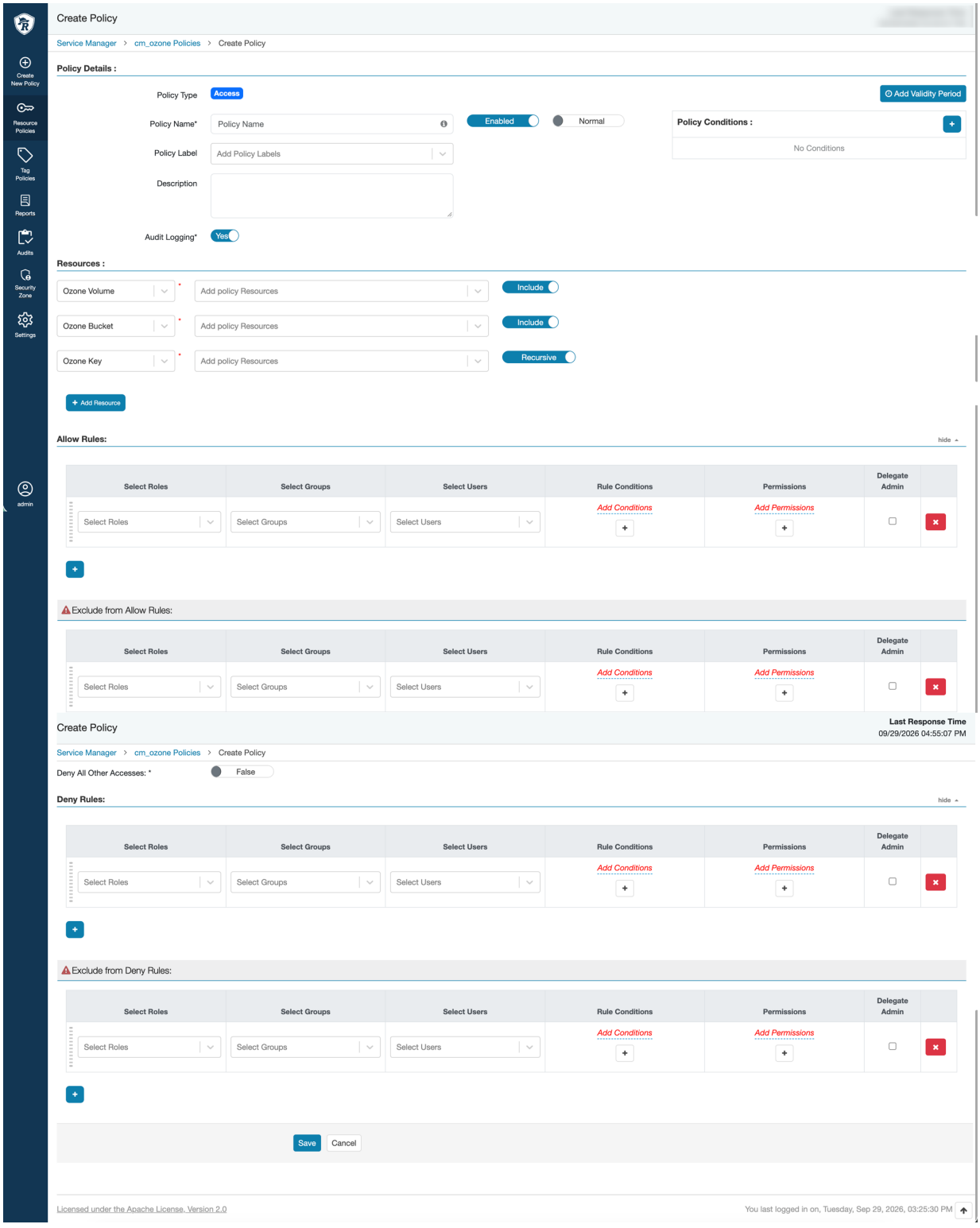
Learn how to create a resource policy in Ranger to grant the permissions for all keys in a bucket using UI.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.

3. Select the relevant Ranger cluster.
4. Click the Web UI dropdown menu and click one of the Ranger Admin Web UI. The Ranger log in page displays.
5. Sign in to Ranger. The Ranger home page displays.
6. In the left navigation pane, select Resource Policies, and then select cm_ozone.
7. Click Add New Policy button. The **Create Policy** page displays.

Figure 5: Create new policy in Ranger



8. In the Policy Name box, enter a descriptive name for the policy. For example, iceberg table access.
9. In the Resources section, perform the following steps:
   a) Make sure that the Ozone Volume drop-down menu value is set to Ozone Volume.
   b) In the Add policy Resources text box, enter the volume name (for example, s3v).
   c) Wait for few seconds and then select create *[***VOLUME_NAME***]* (for example, create s3v).
   d) Make sure that the Ozone Bucket drop-down menu value is set to Ozone Bucket.

e) In the Add policy Resources text box, enter the bucket name (for example, iceberg).
f) Wait for few seconds and then select create *[***BUCKET_NAME***]* (for example, create iceberg).
g) Make sure that the Ozone Key drop-down menu value is set to Ozone Key.
h) In the Add policy Resources text box, enter *.
i) Wait for few seconds and then select create *.

10. In the Allow Rules section, perform the following actions:

a) Under Select Roles, click the Select Roles dropdown menu,
b) Select your target access role (for example, iceberg-data-all-access).
c) Under Permissions, click the plus sign (+).
d) Select the check box next to All, and then select the  check mark at the bottom of the dialog box.
e) Locate the row assigned to your target user or role (for example, iceberg-data-all-access).
f) Under Rule Conditions, click the plus (+) button to add a new condition.
g) Under Actions, select GetObject and PutObject.
h) Click Save in the dialog box.

11. Click Save.

Creating a resource policy for all keys in a bucket using CLI

Learn how to create a resource policy in Ranger to grant the permissions for all keys in a bucket using CLI.

Before you begin

Ensure that you have the Administrator user name and password for Ranger.

Procedure

1. Open a terminal on your device.

2. Run the following `curl` command to create a resource policy in Ranger to grant the permissions for all the keys in the bucket:

```
curl -u [***RANGER_ADMIN_USERNAME***]:[***RANGER_ADMIN_PASSWORD***] \
 --include --location \
 --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "isEnabled": true,
    "service": "[***RANGER_SERVICE_NAME***]",
    "name": "[***RANGER_POLICY_NAME***]",
    "policyType": 0,
    "policyPriority": 0,
    "isAuditEnabled": true,
    "resources": {
      "volume": {
        "values": ["[***VOLUME_NAME***]"],
        "isExcludes": false,
        "isRecursive": false
      },
      "bucket": {
        "values": ["[***BUCKET_NAME***]}"],
        "isExcludes": false,
        "isRecursive": false
      },
      "key": {
        "values": ["*"],
        "isExcludes": false,
        "isRecursive": true
```

```
      }
    },
    "policyItems": [
      {
        "accesses": [
          {
            "type": "all",
            "isAllowed": true
          }
        ],
        "roles": ["[***ROLE_NAME***]"],
        "conditions": [{ "type": "action-matches", "values": ["GetObject",
 "PutObject"] }],
        "delegateAdmin": false
      }
    ],
    "serviceType": "ozone",
    "isDenyAllElse": false
  }' \
  "[***RANGER_URL***]/service/public/v2/api/policy"
```

Example,

```
curl   -u admin:Admin123 \
  --location \
  --request POST \
  --header "Content-Type: application/json" \
  --header "accept: application/json" \
  --data '{
    "isEnabled": true,
    "service": "cm_ozone",
    "name": "iceberg table access",
    "policyType": 0,
    "policyPriority": 0,
    "isAuditEnabled": true,
    "resources": {
      "volume": {
        "values": ["s3v"],
        "isExcludes": false,
        "isRecursive": false
      },
      "bucket": {
        "values": ["iceberg"],
        "isExcludes": false,
        "isRecursive": false
      },
      "key": {
        "values": ["*"],
        "isExcludes": false,
        "isRecursive": true
      }
    },
    "policyItems": [
      {
        "accesses": [
          {
            "type": "all",
            "isAllowed": true
          }
        ],
        "roles": ["iceberg-data-all-access"],
        "conditions": [{ "type": "action-matches", "values": ["GetObject",
 "PutObject"] }],
```

```
        "delegateAdmin": false
      }
    ],
    "serviceType": "ozone",
    "isDenyAllElse": false
  }' \
  "https://ranger-host.example.com:6182/service/public/v2/api/policy"
```

How to use Ozone S3 Security Token Service

Learn how to use Ozone S3 Security Token Service.

Creating an Ozone bucket and installing AWS CLI

Learn how to create an Ozone bucket under the s3v volume, upload a sample text file, and install the AWS CLI utility.

Before you begin

- Ensure you have SSH access to the target server with root or other required permissions.
- Valid Kerberos keytab file for the hdfs principal located at /cdep/keytabs/hdfs.keytab.

Procedure

1. Open a terminal and connect to the cluster host using SSH:

   ```
   ssh root@[***CLOUDERA_MANAGER_HOST_NAME***]
   ```

   Example,

   ```
   ssh root@cm_hostname_1.example.com
   ```

2. When prompted, enter the password.
3. Authenticate as the hdfs user in Kerberos using the keytab:

   ```
   kinit -kt /cdep/keytabs/hdfs.keytab hdfs
   ```

4. Create a bucket:

   ```
   ozone sh bucket create /[***VOLUME_NAME***]/[***BUCKET_NAME***]
   ```

   Example,

   ```
   ozone sh bucket create /s3v/iceberg
   ```

5. Create a sample text file for testing:

   ```
   echo "string for demo" > ./[***FILE_NAME***].txt
   ```

   Example,

   ```
   echo "string for demo" > ./demotest.txt
   ```

6. Upload the test file to the bucket:

   ```
   ozone sh key put
    /[***VOLUME_NAME***]/[***BUCKET_NAME***]/[***FILE_NAME***].txt
    ./[***FILE_NAME***].txt
   ```

Example,

```
ozone sh key put /s3v/iceberg/demotest.txt ./demotest.txt
```

7. Download and install the AWS CLI package on the server:

```
curl "https://awscli.amazonaws.com/[***AWS_PACKAGE_NAME***]" -o "awscli
v2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Example for a Linux server:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awsc
liv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Creating a user in Ozone

Learn how to create a user in Ozone.

Before you begin

An existing user must be already configured with the ASSUME_ROLE permission in Ranger. For more details about creating a user with the ASSUME_ROLE permission, see Creating a user in Ranger.

Procedure

1. Sign in to the host server as the root user or other user with required permissions.
2. Run the following command to create the Kerberos user:

```
kadmin.local -q "addprinc -randkey [***PRINCIPAL***]" && \
kadmin.local -q "xst -k /tmp/[***SERVICE_USER***].key
tab [***PRINCIPAL***]" && \
cp "/tmp/$[***SERVICE_USER***].keytab"
 "[***KEYTAB_DIR***]/[***SERVICE_USER***].keytab"
```

Example,

```
kadmin.local -q "addprinc -randkey svc-iceberg-rest-catalog/icebergclien
t@example.com" && \
kadmin.local -q "xst -k /tmp/svc-iceberg-rest-catalog.keytab svc-iceberg-
rest-catalog/icebergclient@example.com" && \
cp /tmp/svc-iceberg-rest-catalog.keytab /cdep/keytabs/svc-iceberg-rest-cat
alog.keytab
```

Obtaining S3 credentials in Ozone

Learn how to authenticate an Ozone cluster using Kerberos, generating S3-compatible access keys, and configuring the AWS CLI to interact directly with Ozone storage.

Before you begin

- Access to an Ozone environment with Kerberos authentication enabled.
- AWS CLI installed on your workstation.
- A valid Kerberos keytab file or principal password.

Procedure

1. Open a terminal on your device.
2. Perform the following command to get the permanent credentials:

```
kinit -kt [***PATH_TO_KEYTAB_FILE***] [***HOSTNAME***] && ozone s3 getsecr
et
```

Example,

```
kinit -kt /cdep/keytabs/svc-iceberg-rest-catalog.keytab svc-iceberg-rest-
catalog/icebergclient@example.com && ozone s3 getsecret
```

3. Note down the awsAccessKey and awsSecret values displayed in the terminal output.
4. Configure the AWS client:

```
aws configure
```

5. When prompted, provide the following information:
   - AWS Access Key ID: Enter the awsAccessKey obtained earlier.
   - AWS Secret Access Key: Enter the awsSecret obtained earlier.
   - Default region name: Enter the region that's appropriate for your installation. For example, us-east-1.
   - Default output format: Leave it blank.
6. Press Enter.

Generating the Ozone S3 Security Token Service using the AssumeRole API

Learn how to generate and validate Ozone S3 Security Token Service (STS) in different scenarios using the AssumeRole API.

Generating limited-scope STS credentials Using AssumeRole

Learn how to request short-term Ozone S3 Security Token Service (STS) credentials scoped with limited permissions (for example, inline read-only **s3:GetObject** permissions).

Before you begin

Access to an environment configured with a STS service endpoint and a target Ranger role.

Procedure

1. Open a terminal on your device.



Note: This is the same server used in Creating an Ozone bucket and installing AWS CLI.

2. Run the `assume-role` command with an inline session policy to restrict permissions to read-only (s3:GetObject):

```
aws sts assume-role \
  --role-arn arn:aws:iam::[***ACCOUNT_ID***]:role/[***ROLE_NAME***] \
  --role-session-name "[***SESSION_NAME***]" \
 --policy '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::[***BUCKET_NAME***]/*"
      }
    ]
```

```
  }' \
  --duration-seconds 3600 \
  --endpoint-url "https://[***OZONE_HOST***]:[***OZONE_PORT***]"
```

where,

- *[***ACCOUNT_ID***]* is a dummy ID that must match the valid length of an AWS account ID.
- *[***ROLE_NAME***]* must be the same role that was created in Ranger.
- *[***SESSION_NAME***]* is a dummy name that must match the valid length as per AWS specifications.
- *[***BUCKET_NAME***]* must be the bucket name that was created in Ozone and for which corresponding resource policies were created in Ranger.

Example,

```
aws sts assume-role \
  --role-arn "arn:aws:iam::123456789012:role/iceberg-data-all-access" \
  --role-session-name "iceberg-session-with-policy" \
  --policy '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::iceberg/*"
      }
    ]
  }' \
  --duration-seconds 3600 \
  --endpoint-url "https://ozone-host.example.com:9881"
```

The command returns a JSON object containing the temporary STS credentials:

- AccessKeyId
- SecretAccessKey
- SessionToken

Record these temporary STS credentials.

3. Open a new terminal and SSH into the cluster server :



Note: This server must be different from the one used in Creating an Ozone bucket and installing AWS CLI.

```
ssh root@[***CLOUDERA_MANAGER_HOST_NAME***]
```

Example,

```
ssh root@cm_hostname_2.example.com
```

4. When prompted, enter the password.
5. Download and install the AWS CLI (if not already installed):

```
curl https://awscli.amazonaws.com/[***AWS_PACKAGE_NAME***] -o awscliv2.zip
unzip awscliv2.zip
sudo ./aws/install
```

Example for a Linux server:

```
curl https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip -o awscliv
2.zip
unzip awscliv2.zip
```

```
sudo ./aws/install
```

6. Configure the AWS client to use the STS credentials that were generated in the step 2 on page 45 in the previous cluster server:

```
CRED
S='[***PASTE_THE_JSON_RESPONSE_COPIED_IN_STEP_2_WHILE_INVOKING_ASSUMEROLE_API***]'
```

7. Parse the credentials and export the environment variables along with your S3 endpoint URL:

```
export AWS_ACCESS_KEY_ID=$(echo "$CREDS" | jq -r .Credentials.AccessKeyId)
export AWS_SECRET_ACCESS_KEY=$(echo "$CREDS" | jq -r .Credentials.SecretA
ccessKey)
export AWS_SESSION_TOKEN=$(echo "$CREDS" | jq -r .Credentials.SessionToken
)
export AWS_ENDPOINT_URL_S3=htt
ps://[***OZONE_SSL_S3_GATEWAY_HOST***]:[***OZONE_PORT***]
```

Example,

```
export AWS_ACCESS_KEY_ID=$(echo "$CREDS" | jq -r .Credentials.AccessKeyId)
export AWS_SECRET_ACCESS_KEY=$(echo "$CREDS" | jq -r .Credentials.SecretA
ccessKey)
export AWS_SESSION_TOKEN=$(echo "$CREDS" | jq -r .Credentials.SessionToken
)
export AWS_ENDPOINT_URL_S3="https://ozone-ssl-s3-gateway-host.example.com:
9879"
```

8. Attempt to create a bucket to verify that the command fails due to insufficient permissions:



Note: This command should fail because GetObject in the session policy does not permit bucket creation.

```
aws s3api create-bucket --bucket [***TEST_BUCKET_NAME***]> --regi
on [***REGION***]
```

Example,

```
aws s3api create-bucket --bucket my-sts-created-bucket-001 --region us-e
ast-1
```

9. Retrieve the test object and this command will be executed successfully:

```
aws s3api get-object --bucket [***BUCKET_NAME***] --key [***OBJECT_KEY***]
 ./[***TEST_FILE_NAME***]
```

Example,

```
aws s3api get-object --bucket iceberg --key demotest.txt ./test.txt
```

10. Validate the content of the retrieved file:

```
cat ./[***TEST_FILE_NAME***]
```

Example,

```
cat ./test.txt
```

11. Attempt to overwrite the object to verify that the command fails due to insufficient permissions:

Note: This command should fail because GetObject in the session policy does not permit uploading or overwriting objects.

```
aws s3api put-object --bucket [***BUCKET_NAME***] --key [***OBJECT_KEY***]
 --body ./[***TEST_FILE_NAME***]
```

Example,

```
aws s3api put-object --bucket iceberg --key demotest.txt --body ./test.txt
```

12. Attempt to retrieve the object tags to verify that the command fails due to insufficient permissions:

Note: This command should fail because GetObject in the session policy does not permit retrieving object tags.

```
aws s3api get-object-tagging --bucket [***BUCKET_NAME***] --k
ey [***OBJECT_KEY***]
```

Example,

```
aws s3api get-object-tagging --bucket iceberg --key demotest.txt
```

Generating role-scoped STS credentials Using AssumeRole

Learn how to request short-term AWS Security Token Service (STS) credentials using the full, unrestricted permissions of a target role (without inline session policy constraints). It also details how to apply these temporary credentials in a client environment and verify that the expected role-level permissions are active.

Procedure

1. Open a terminal on your device.
2. To obtain role-scoped STS credentials, run the following `assume-role` command in the Cloudera Manager Host 1 where you configured AWS:

Note: This is the same server used in Creating an Ozone bucket and installing AWS CLI.

```
aws sts assume-role \
  --role-arn arn:aws:iam::[***ACCOUNT_ID***]:role/[***ROLE_NAME***] \
  --role-session-name "[***SESSION_NAME***]" \
  --duration-seconds 3600 \
  --endpoint-url "https://[***OZONE_HOST***]:[***OZONE_PORT***]"
```

where,

- *[***ACCOUNT_ID***]* is a dummy ID that must match the valid length of an AWS account ID.
- *[***ROLE_NAME***]* must be the same role that was created in Ranger.
- *[***SESSION_NAME***]* is a dummy name that must match the valid length as per AWS specifications.

Example,

```
aws sts assume-role \
  --role-arn "arn:aws:iam::123456789012:role/iceberg-data-all-access" \
  --role-session-name "iceberg-session-no-policy" \
  --duration-seconds 3600 \
  --endpoint-url "https://ozone-host.example.com:9881"
```

The command returns a JSON object containing the temporary STS credentials:

- AccessKeyId
- SecretAccessKey
- SessionToken

Record these temporary STS credentials.

3. Open a new terminal and connect to the new cluster server using SSH:



Note: This server must be different from the one used in Creating an Ozone bucket and installing AWS CLI.

```
ssh root@[***CLOUDERA_MANAGER_HOST_NAME***]
```

Example,

```
ssh root@cm_hostname_2.example.com
```

4. When prompted, enter the password.
5. Download and install the AWS CLI (if not already installed):

```
curl https://awscli.amazonaws.com/[***AWS_PACKAGE_NAME***] -o awscliv2.zip
unzip awscliv2.zip
sudo ./aws/install
```

Example for a Linux server:

```
curl https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip -o awscliv
2.zip
unzip awscliv2.zip
sudo ./aws/install
```

6. Configure the AWS client to use the STS credentials that were generated in the step 2 on page 48 in the previous cluster server:

```
CRED
S='[***PASTE_THE_JSON_RESPONSE_COPIED_IN_STEP_2_WHILE_INVOKING_ASSUMEROLE_API***]'
```

7. Parse the credentials and export the environment variables along with your S3 endpoint URL:

```
export AWS_ACCESS_KEY_ID=$(echo "$CREDS" | jq -r .Credentials.AccessKeyId)
export AWS_SECRET_ACCESS_KEY=$(echo "$CREDS" | jq -r .Credentials.SecretA
ccessKey)
export AWS_SESSION_TOKEN=$(echo "$CREDS" | jq -r .Credentials.SessionToken
)
export AWS_ENDPOINT_URL_S3=htt
ps://[***OZONE_SSL_S3_GATEWAY_HOST***]:[***OZONE_PORT***]
```

Example,

```
export AWS_ACCESS_KEY_ID=$(echo "$CREDS" | jq -r .Credentials.AccessKeyId)
export AWS_SECRET_ACCESS_KEY=$(echo "$CREDS" | jq -r .Credentials.SecretA
ccessKey)
export AWS_SESSION_TOKEN=$(echo "$CREDS" | jq -r .Credentials.SessionToken
)
export AWS_ENDPOINT_URL_S3="https://ozone-ssl-s3-gateway-host.example.com:
9879"
```

8. Attempt to create a bucket and this command must fail due to limited permissions:

Note: This command should fail because the assumed role only grants GetObject and PutObject actions, neither of which permits bucket creation.

```
aws s3api create-bucket --bucket [***TEST_BUCKET_NAME***] --regio
n [***REGION***]
```

Example,

```
aws s3api create-bucket --bucket my-sts-created-bucket-001 --region us-e
ast-1
```

9. Retrieve the test object and this command must be executed successfully:

```
aws s3api get-object --bucket [***BUCKET_NAME***] --key [***OBJECT_KEY***]
 ./[***TEST-FILE-NAME***]
```

Example,

```
aws s3api get-object --bucket iceberg --key demotest.txt ./test.txt
```

10. Overwrite the object and this command must be executed successfully:

```
aws s3api put-object --bucket [***BUCKET_NAME***] --key [***OBJECT_KEY***]
 --body ./[***TEST-FILE-NAME***]
```

Example,

```
aws s3api put-object --bucket iceberg --key demotest.txt --body ./test.txt
```

11. Attempt to retrieve the object tags to verify that the command fails due to insufficient permissions:

Note: This command should fail because the assumed role only grants GetObject and PutObject actions, neither of which permits retrieving object tags.

```
aws s3api get-object-tagging --bucket [***BUCKET_NAME***] --k
ey [***OBJECT_KEY***]
```

Example,

```
aws s3api get-object-tagging --bucket iceberg --key demotest.txt
```

Applying the conflicting action conditions to AssumeRole requests using the API

Learn how to generate temporary AWS Security Token Service (STS) credentials using an inline session policy that conflicts with the underlying Ranger policy restrictions. Use this procedure to verify that conflicting session policy actions result in access denial due to a lack of permission overlap.

Procedure

1. Open a terminal on your device.
2. To invoke the assume-role API with a session policy specifying an action (for example, s3:GetObjectTagging) that conflicts with your target role's configured Ranger policy (for example, GetObject and PutObject), run the following assume-role command in the Cloudera Manager Host 1 where you configured AWS:

```
aws sts assume-role \
  --role-arn arn:aws:iam::[***ACCOUNT_ID***]:role/[***ROLE_NAME***] \
  --role-session-name "[***SESSION_NAME***]" \
 --policy '{
```

```
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:GetObjectTagging",
        "Resource": "arn:aws:s3:::[***BUCKET_NAME***]/*"
      }
    ]
  }' \
  --duration-seconds 3600 \
  --endpoint-url "https://[***OZONE_HOST***]:[***OZONE_PORT***]"
```

where,

- *[***ACCOUNT_ID***]* is a dummy ID that must match the valid length of an AWS account ID.
- *[***ROLE_NAME***]* must be the same role that was created in Ranger.
- *[***SESSION_NAME***]* is a dummy name that must match the valid length as per AWS specifications.
- *[***BUCKET_NAME***]* must be the bucket name that was created in Ranger.

Example,

```
aws sts assume-role \
  --role-arn "arn:aws:iam::123456789012:role/iceberg-data-all-access" \
  --role-session-name "iceberg-session-with-policy" \
  --policy '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "s3:GetObjectTagging",
        "Resource": "arn:aws:s3:::iceberg/*"
      }
    ]
  }' \
  --duration-seconds 3600 \
  --endpoint-url "https://ozone-host.example.com:9881"
```



Important: Because access control evaluates the intersection of the STS session policy and the underlying Ranger resource policy, requesting an action that does not overlap with the role's allowed actions results in a session with no effective permissions. All S3 requests made with this token will be denied.

Ozone Ranger Integration

Set up policies in Ranger for the users to have the right access permissions to the various Ozone objects such as buckets and volumes.

When using Ranger to provide a particular user with read/write permissions to a specific bucket, you must configure a separate policy for the user to have read access to the volume in addition to policies configured for the bucket.

Configuring a resource-based policy using Ranger

Using Ranger, you can setup new Ozone policies that will help you to set right access permissions to various Ozone objects like volumes and buckets.

About this task

Through configuration, Apache Ranger enables both Ranger policies and Ozone permissions to be checked for a user request. When the Ozone Manager receives a user request, the Ranger plugin checks for policies set through the Ranger Service Manager. If there are no policies, the Ranger plugin checks for permissions set in Ozone.

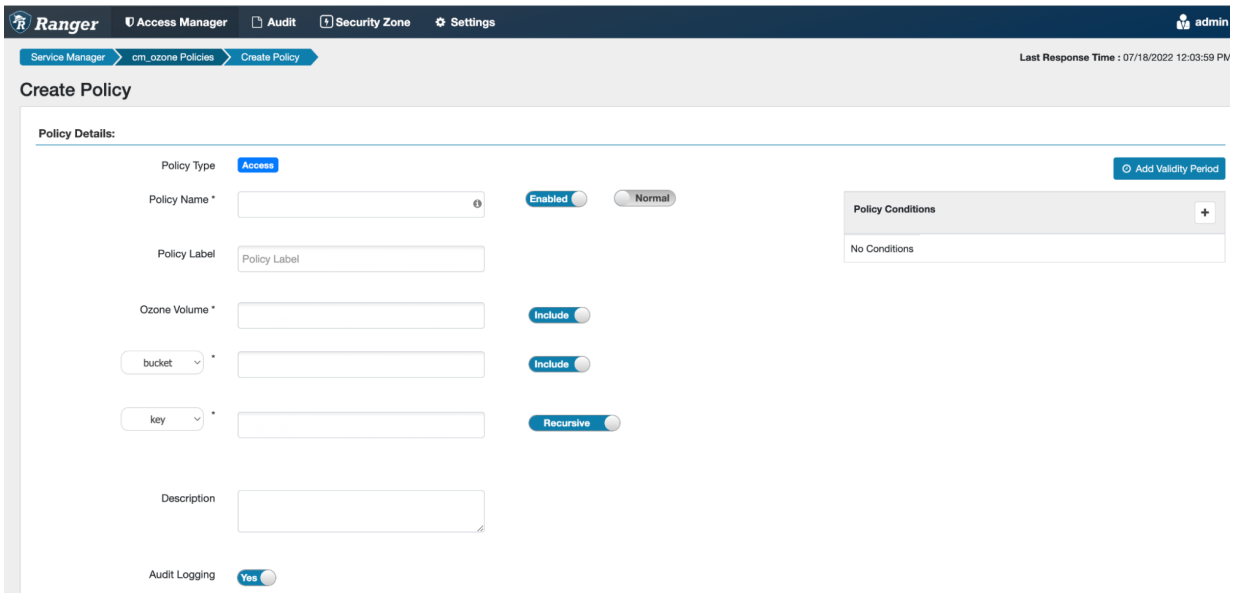
Cloudera recommends you to create permissions at the Ranger Service Manager, and to have restrictive permissions at the Ozone level.

Note:

- For more information on how Ranger policies work on Ozone resources, see Ranger policies with Ozone resources.
- By default, any changes in the Ranger policy will take 30 seconds to take effect in Ozone.

Procedure

1. On the Service Manager page, select an existing Ozone service. The List of Policies page appears.
2. Click Add New Policy. The Create Policy page appears.



3. Set the configurations on **Create Policy** page:

Field	Description
Policy Name	Enter a unique name for this policy. The name cannot be duplicated anywhere in the system.
Normal or Override	Enables you to specify an override policy. When override is selected, the access permissions in the policy override the access permissions in existing policies. This feature can be used with Add Validity Period to create temporary access policies that override existing policies.
Add Validity Period	Specify a start and end time for the policy.
Policy Label	(Optional) Specify a label for this policy. You can search reports and filter policies based on these labels.
Ozone Volume	Specify volumes that can be accessed. Ensure that the Ozone volume key is set to Include.

Field	Description
	If you want to deny access at the volume level, then disable this option by turning off using the Ozone Volume key to Exclude.
Bucket	Specify buckets that can be accessed. Ensure that the Ozone Bucket key is set to Include. If you want to deny access at the bucket level, then disable this option by turning off using the bucket key. Or, select None from the Bucket drop-down.
Key	Provide the Key
Recursive or Non Recursive	Recursive or Non recursive function
Description	(Optional) Describe the purpose of the policy.
Audit Logging	Specify whether this policy is audited. To disable auditing, turn off the Audit Logging key.

4. Allow Conditions

Label	Description
Select Role	Specify the roles to which this policy applies. To designate a role as an Administrator, select the Delegate Admin check box. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Select Group	Specify the groups to which this policy applies. To designate a group as an Administrator, select the Delegate Admin check box. Administrators can edit or delete the policy, and can also create child policies based on the original policy. The public group contains all users, so granting access to the public group grants access to all users.
Select User	Specify the users to which this policy applies. To designate a user as an Administrator, select the Delegate Admin check box. Administrators can edit or delete the policy, and can also create child policies based on the original policy.
Policy Conditions	Provide the IP address range
Permissions	Add or edit permissions: All, Read, Write, Create, List, Delete, Read_ACL, Write_ACL, and Select All. **Note:** If you select specific permissions, then you must not select the All option as the All option implicitly considers all permissions overriding the permissions you have specifically selected.

Label	Description
Delegate Admin	You can use Delegate Admin to assign administrator privileges to the roles, groups, or users specified in the policy. Administrators can edit or delete the policy, and can also create child policies based on the original policy.

Note: You can use the Plus (+) symbol to add additional conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on. Similarly, you can also exclude certain Allow Conditions by adding them to the Exclude from Allow Conditions list.

5. You can use the Deny All Other Accesses toggle key to deny access to all other users, groups, and roles other than those specified in the allow conditions for the policy.
6. If you wish to deny access to a few or specific users, groups, or roles, then use must set Deny Conditions.

   You can use the Plus (+) symbol to add deny conditions. Conditions are evaluated in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on. Similarly, you can also exclude certain Deny Conditions by adding them to the Exclude from Deny Conditions list.
7. Click Add.

Snapshot support in Ozone

Learn about different scenarios where you can use snapshots, the snapshot APIs that are available for use, and the snapshot architecture.

Snapshot feature for Apache Ozone object store enables you to take point-in-time consistent image of a given bucket. Snapshot feature enables you to handle various use cases, including:

- Backup and restore

  Create hourly, daily, weekly, or monthly snapshots for backup and recovery.
- Archival and compliance

  Take snapshots for compliance purpose and archive them.
- Replication and disaster recovery (DR)

  Snapshots provide frozen immutable images of the bucket on the source Ozone cluster. Snapshots can be used for replicating these immutable bucket images to remote DR sites.
- Incremental replication

  DistCp with SnapshotDiff offers an efficient way to incrementally sync up source and destination buckets.

Snapshot APIs

Snapshot feature is available through ozone fs and ozone sh CLI. This feature can also be programmatically accessed from Ozone ObjectStore Java client. The feature provides following functionalities:

- Create an instantenous snapshot for a given bucket `ozone sh snapshot create [-hV] <bucket> [<snapshotName>]`.
- List all snapshots of a given bucket `ozone sh snapshot list [-hV] <bucket>`.
- Delete a specific snapshot for a given bucket `ozone sh snapshot delete [-hV] <bucket> <snapshotName>`.
- Given two snapshots, list all the keys that are different between them - SnapshotDiff `ozone sh snapshot diff [-chV] [-p=<pageSize>] [-t=<continuation-token>] <bucket> <fromSnapshot> <toSnapshot>`.

The SnapshotDiff functionality in CLI/API is asynchronous. The first time the API is invoked, Ozone Manager (OM) starts a background thread to calculate the SnapshotDiff, and returns Retry with suggested duration for the retry operation. After the SnapshotDiff is computed, this API returns the differences in multiple pages. Within each Snap shotDiff response, OM also returns a continuation token for the client to continue from the last batch of SnapshotDiff results. This API is safe to be called multiple times for a given snapshot source and destination pair. Internally, each OM computes SnapshotDiff only once and stores it for future invocations of the same SnapshotDiff API.

Snapshot architecture

Ozone snapshot architecture leverages the fact that data blocks once written, remain immutable in their lifetime. These data blocks are reclaimed only when the object key metadata that references them, is deleted from the Ozone namespace. All of the Ozone metadata is stored on the OM nodes in the Ozone cluster. When you take a snapshot of an Ozone bucket, internally the system takes snapshot of the Ozone metadata in OM nodes. Since Ozone does not allow updates to DataNode blocks, integrity of data blocks referenced by Ozone metadata snapshot in OM nodes remains intact. Ozone key deletion service is also aware of Ozone snapshots. Key deletion service does not reclaim any key as long as it is referenced by the active object store bucket or any of its snapshot. When the snapshots are deleted, a background garbage collection service reclaims any key that is not part of any snapshot or active object store. Ozone also provides the `SnapshotDiff` API. Whenever a user issues a SnapshotDiff between two snapshots, it efficiently calculates all the keys that are different between these two snapshots and returns paginated SnapshotDiff list result.

Cluster and hardware configuration in snapshot deployment

Learn about CPU, memory, and storage requirements for the snapshot feature.

Snapshot feature places additional demands on the cluster in terms of CPU, memory, and storage. Cluster nodes running Ozone Managers (OM) and Ozone DataNodes (OD) should be configured with extra storage capacity depending on the number of active snapshots that you want to keep. Ozone snapshots consume incremental amount of space per snapshot. For example, if the active object store contains 100 GB data (before replication) and a snapshot is taken, then the 100 GB of space is locked in that snapshot. If the active object store consumes another 10 GB of space (before replication) subsequently, then the overall space requirement is 100 GB + 10 GB = 110 GB in total (before replication). This is because common keys between Ozone snapshots and the active object store share the storage space.

Similarly, nodes running OM should be configured with extra memory depending on how many snapshots are concurrently read from. This also depends on how many concurrent SnapshotDiff jobs are expected in the cluster. By default, an OM allows 10 concurrent SnapshotDiff jobs at a time, which can be increased in configurations.

Erasure Coding overview

The Ozone Erasure Coding (EC) feature provides data durability and fault-tolerance along with reduced storage space and ensures data durability similar to Ratis THREE replication approach.

The Ozone default replication scheme Ratis THREE has 200% overhead storage space including other resources. Using EC in place of replication helps in reducing storage cost as the overhead storage space is only 50%. For example, if you replicate 6 blocks of data, you need 18 blocks of disk space in Ratis. However, if you use EC with Ozone, you need 6 blocks plus 3 parity totalling to 9 blocks of disk space.

Write and read using EC

When a client requests write, OM allocates a block group (data and parity) number of nodes from the pipeline to the client. Client writes d number of chunks to d number of nodes. Parity chunks(p) are created and transferred to the remaining p number of nodes. After this process is completed, the client can request for a new block group after the writing of the current block group is finished.

For reads, OM provides the node location information. If the key is erasure coded, the client reads the data in the EC way.

Note:

- Cloudera recommends you to use the Erasure Coding feature at the bucket level so that EC is applied on all the keys created in a bucket. However, you can configure EC at key level as well.
- If you do not want EC to be configured for specific keys, you can explicitly specify the replication configuration for those keys.
- If replication configuration is not defined specifically for both buckets and keys, then cluster-wide or global default configuration is applied.
- Ozone does not support ISA-L and has no plans to add support. ISA-L is relevant only when EC is enabled. Additionally, if native EC libraries are unavailable, the system automatically falls back to Java libraries. This fallback does not impact write operations.

Enabling EC replication configuration cluster-wide

You can set cluster-wide default Replication configuration with EC by using the configuration keys ozone.server.default.replication.type and ozone.server.default.replication.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Go to Configurations
5. Search for ozone.server.default.replication and ozone.server.default.replication.type
   a) Click Add
   b) Click View as XML
   c) For ozone.server.default.replication property, copy and paste: <property> <name>ozone.server.default.replication</name> <value>RS-X-Y-1024k</value> </property>

      Note: RS-X-Y-1024k is an example where RS is the codec type, X is the number of data blocks, Y is the parity and 1024k is the size of the EC chunk size. For example, if you have 6 data blocks of 1024k size and you need 3 parity blocks, this is the value RS-6-3-1024k
   d) For ozone.server.default.replication.type property, copy and paste: <property> <name>ozone.server.default.replication.type</name> <value>EC</value> </property>
   e) Click View Editor. You must provide the values for the properties

Property	Value
ozone.server.default.replication	Supported EC options are RS-3-2-1024K, RS-6-3-1024K, and RS-10-4-1024K
ozone.server.default.replication.type	EC

   f) Click Save Changes
   g) Restart the Ozone service

Enabling EC replication configuration on bucket

You can enable EC replication configuration at bucket level.

Procedure

1. You can set the bucket level EC Replication configuration through CLI by running `ozone sh bucket create <bucket path> --type EC --replication rs-6-3-1024k`
2. To reset the EC Replication configuration,running `ozone sh bucket set-replication-config <bucket path> --type EC --replication rs-3-2-1024k`

Note:

- The new configuration applies only to the keys created after resetting the EC Replication configuration. Keys created before resetting the EC Replication configuration will have the older configuration.
- If you set EC Replication configuration on RATIS (while writing at a bucket level) and you are using Ozone File System (ofs/o3fs), there is a timeout of 10 minutes where you will continue to write on RATIS as a caching process in place at the bucket level. For fresh buckets, if you set EC Replication configuration on RATIS, the new configuration is immediately available for the bucket. However, for ofs/o3fs/s3 you can only use bucket level settings as you cannot allow EC setting on key creation.

Enabling EC replication configuration on keys or files

You can enable EC configuration replication at key level.

Procedure

You can set the key level EC Replication configuration command through CLI while creating the keys irrespective of bucket Replication configuration `ozone sh key put <Ozone Key Object Path> <Local File> --type EC --replication rs-6-3-1024k`

Note: If you have already configured the default EC Replication configuration for a bucket, you do not have to configure the EC Replication configuration while creating a key.

Master node decommissioning in Ozone

This feature helps you to decommission Ozone Manager (OM) and Storage Container Manager (SCM).

SCM decommissioning

Storage Container Manager (SCM) decommissioning is the process in which you can remove one of the SCM from the SCM HA Ring.

This section provides you the steps to decommission SCM and primordial SCM.

Decommissioning SCM

Learn how to decommission Storage Container Manager (SCM).

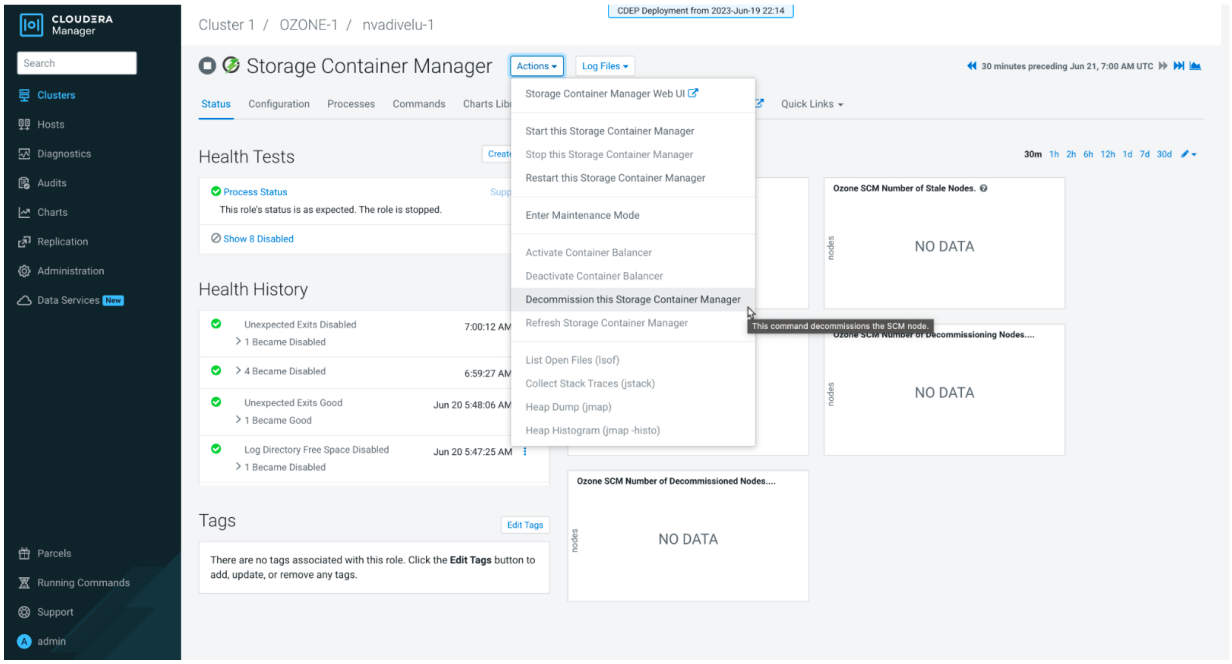
About this task

To decommission SCM, perform the following steps.

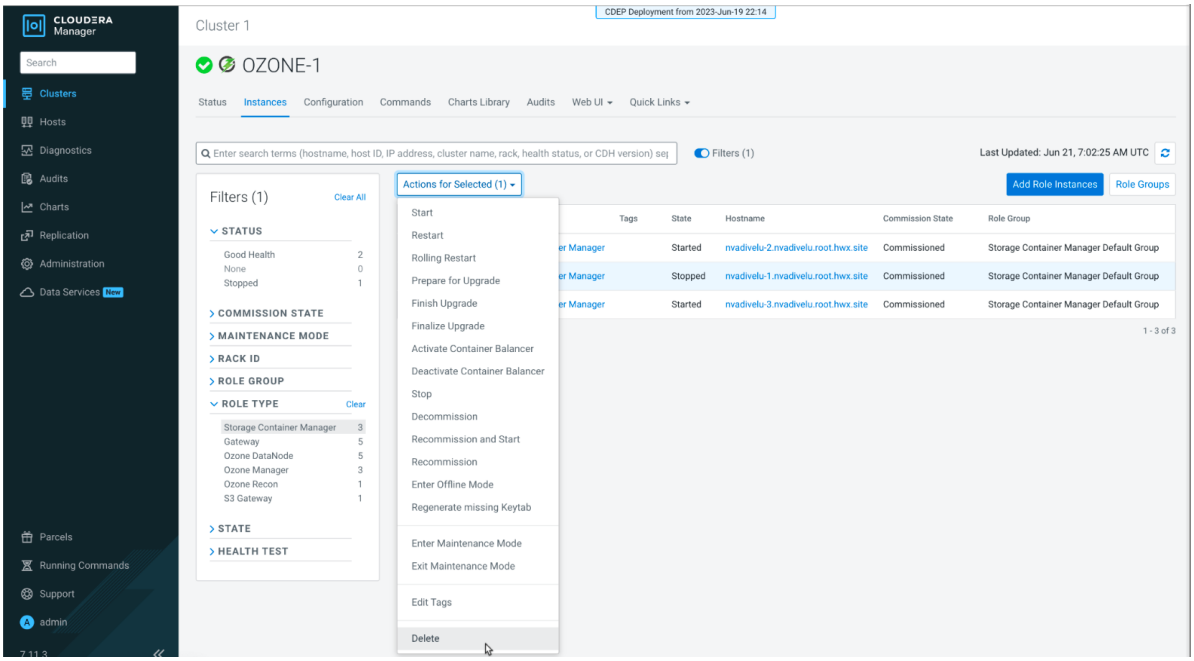
Procedure

1. Log in to Cloudera Manager.
2. Navigate to Clusters.
3. Select the Ozone service.

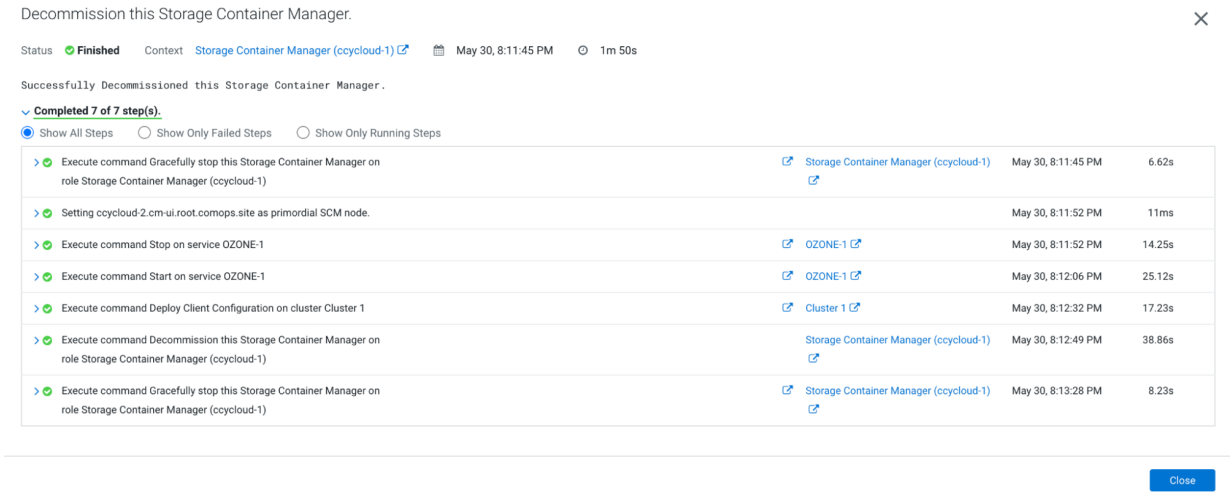
4. Click Instances.
5. Click Storage Container Manager (node that has to be decommissioned).
6. Click Actions.
7. Click Decommission this Storage Container Manager.



8. After the SCM is decommissioned, you must delete the SCM.
   a) Navigate to Clusters.
   b) Select the Ozone service.
   c) Click Instances.
   d) Click Select the Storage Container Manager that is decommissioned.
   e) Click Actions for Selected.
   f) Click Delete.



9. Decommissioning a primordial SCM will also have the same process. However, during the decommissioning process, primordial SCM property will be automatically set.



OM decommissioning

Ozone Manager (OM) decommissioning is the process in which you remove one of the OM from the OM HA Ring.

This section provides you the steps to decommission OM node, run the decommissioning command from Cloudera Manager UI, delete the decommissioned OM, and add new OM node.

Decommissioning OM Node

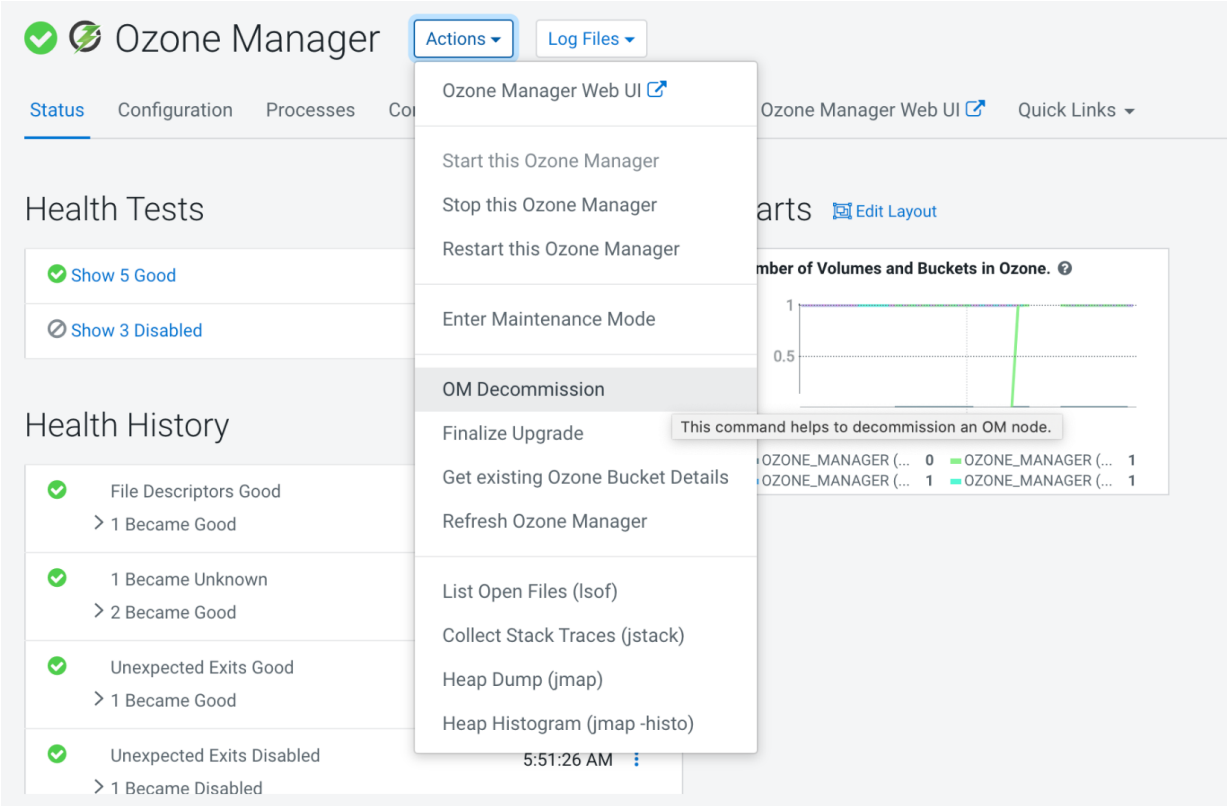
Add the Ozone Manager (OM) NodeId of the decommissioning node to the decommissioning property ozone.om.decommissioned.nodes.[ozone_service_id] in ozone-site.xml of all nodes.

About this task

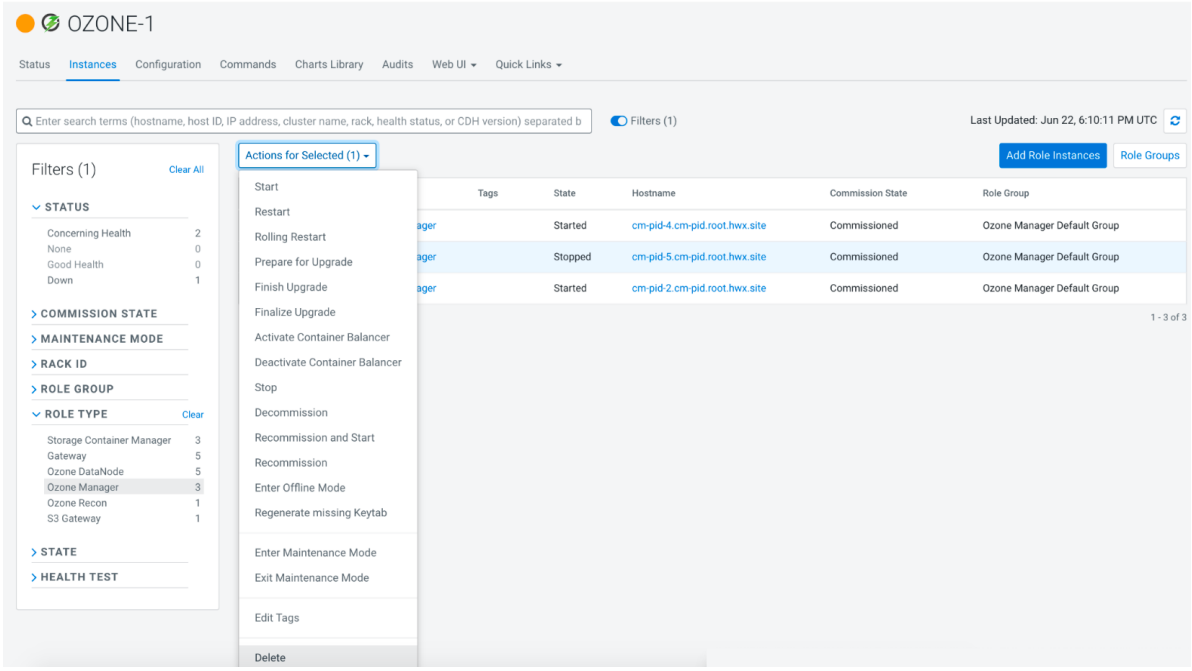
To decommission OM, perform the following steps.

Procedure

1. Log in to Cloudera Manager.
2. Navigate to Clusters.
3. Select the Ozone service.
4. Click Ozone Manager.
5. Click Actions.
6. Click OM Decommission.



7. After the OM is decommissioned, you must delete the OM.
   a) Navigate to Clusters.
   b) Select the Ozone service.
   c) Click Instances.
   d) Click the Ozone Manager that is decommissioned.
   e) Click Actions for Selected.
   f) Click Delete.



Adding new Ozone Manager node

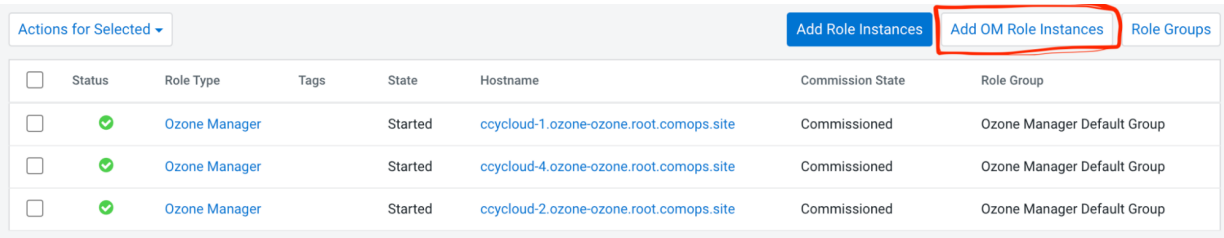
Learn how to add new Ozone Manager (OM) node.

About this task

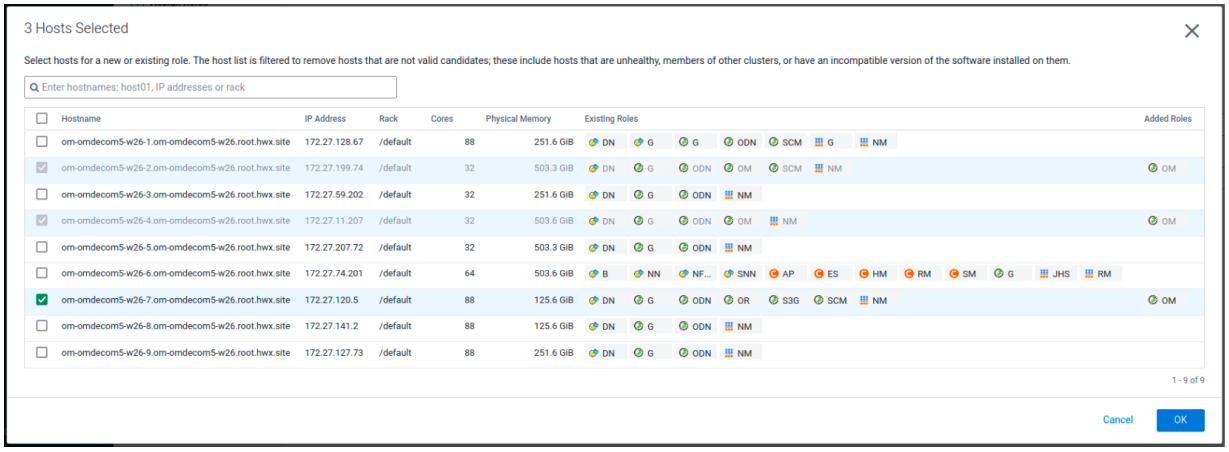
To add a new OM node, perform the following steps.

Procedure

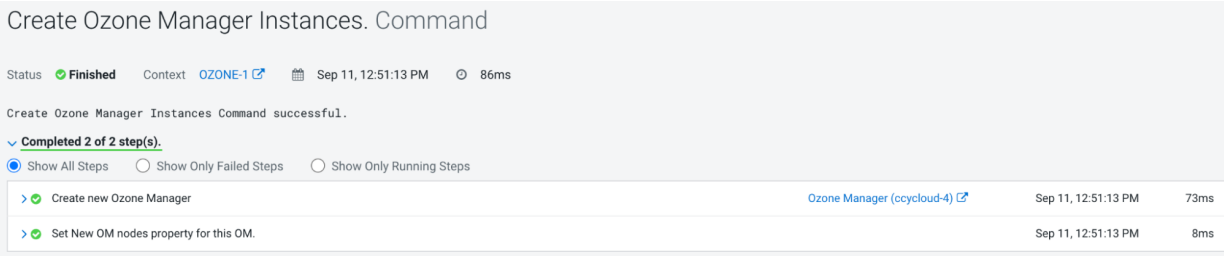
1. Navigate to Clusters.
2. Select the Ozone service.
3. Click Instances.
4. Click Add OM Role Instance.



5. Navigate to Ozone Manager.
6. Select the Hosts.



7. Click Continue.



New Ozone Manager Nodes property will be automatically set.

8. Click Finish.

Ozone recon heatmap

Review the conceptual information on Heatmap and access the heatmap feature as an administrator to read or view the most accessed volumes, buckets, and top 100 keys across Apache Ozone.

Accessing Ozone Recon Web UI

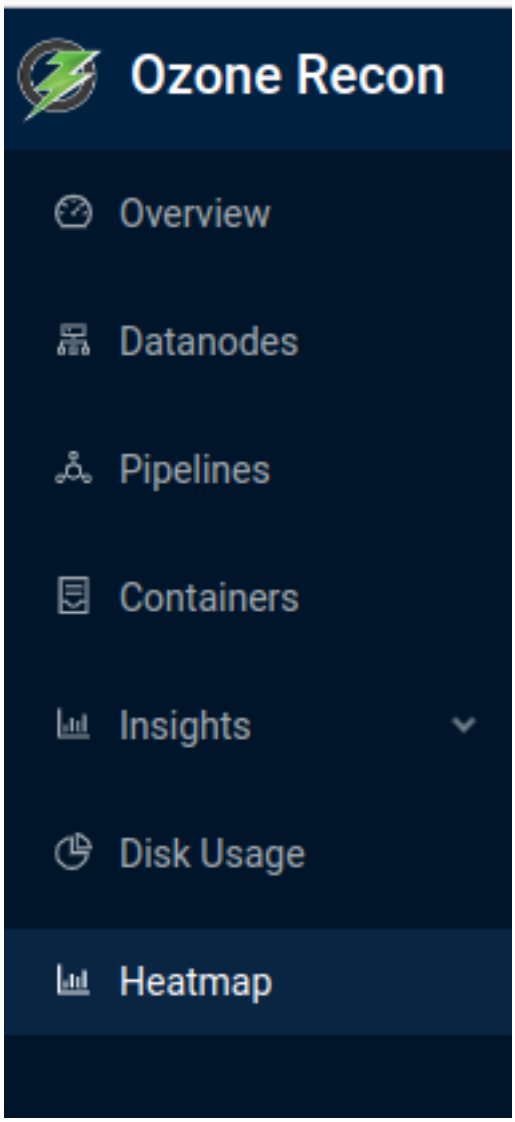
To access the Ozone Recon Web UI, perform the following steps.

About this task

To access Ozone Recon Web UI, perform the following steps.

Procedure

1. Log in to Cloudera Manager.
2. Navigate to Clusters.
3. Select the Ozone service.
4. Click Recon Web UI.
5. On the Ozone Recon left navigation pane, click Heatmap



Ozone recon heatmap

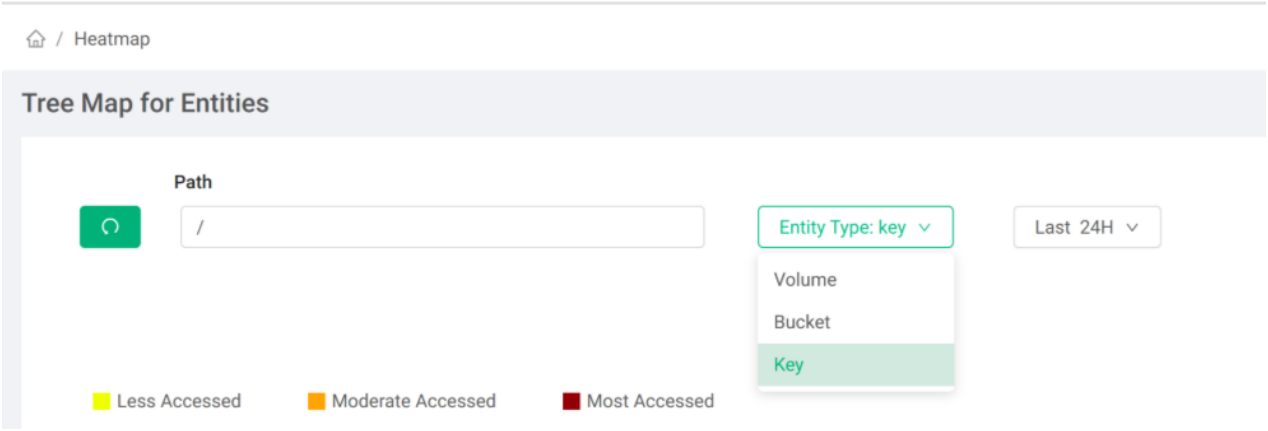
This feature helps the administrator with a capability in Ozone Recon UI to read or view the most accessed volumes, buckets, and top 100 keys across Apache Ozone.

To enable or disable the Heatmap feature, set the ozone.recon.heatmap.enable parameter to true or false respectively. By default, the ozone.recon.heatmap.enable parameter is set to true.

The size of each block is based on the size of the file and color is based on the access count of the file. The Most frequently accessed files are shown in dark red shade as they are the most heated and this shade gets lighter as the access count decreases.

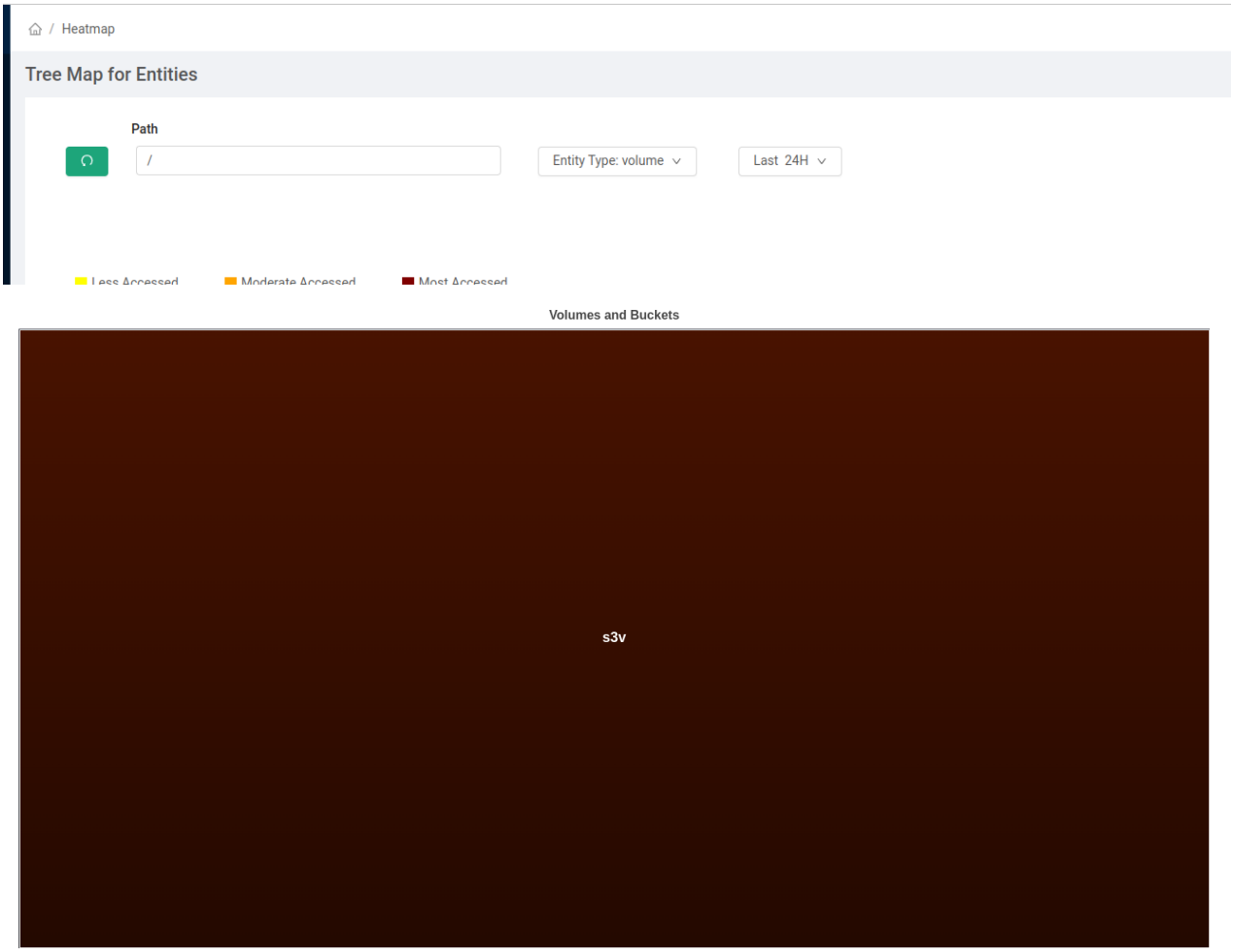


There are three entityTypes present: Volume, Bucket, and Key. You can select any entityType. The default entity type is Key and the default duration is 24 hours. The default value represents the heatmap on read access metadata of the Ozone keys for the last 24 hours.



Volume

Heatmap at volume level is displayed.

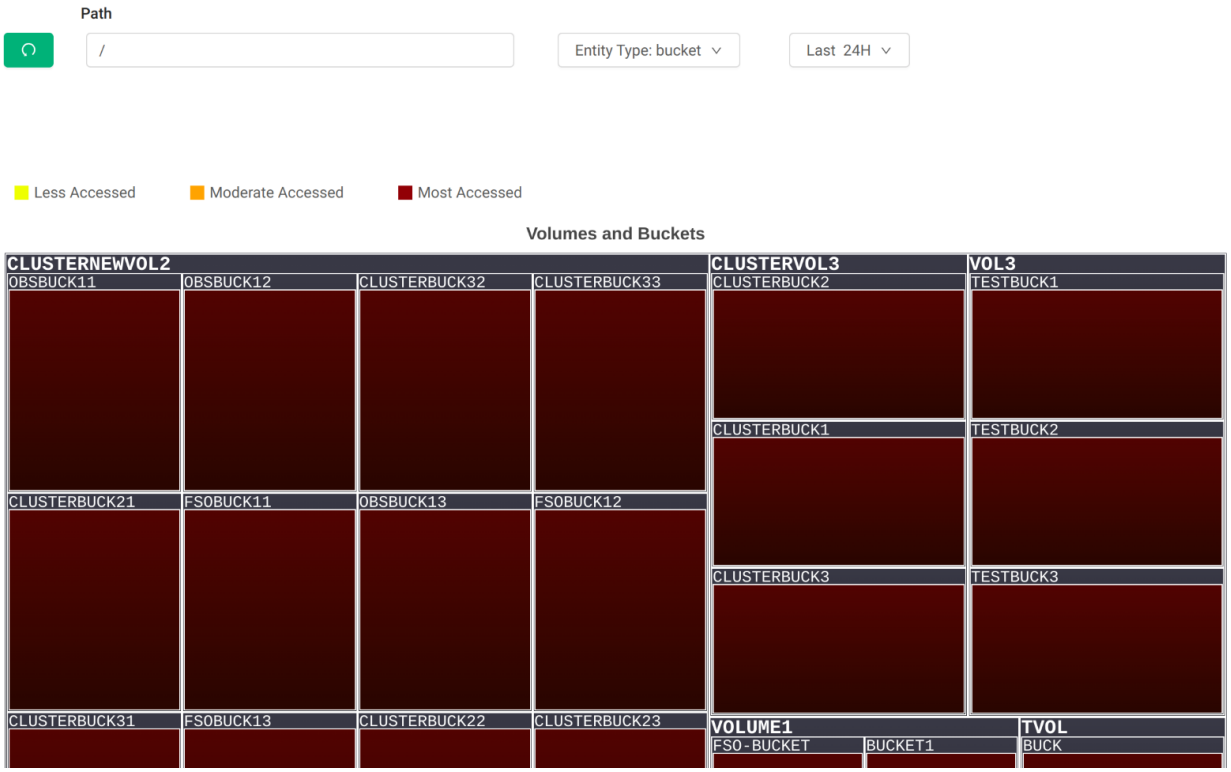


Bucket

Heatmap at bucket level is displayed.

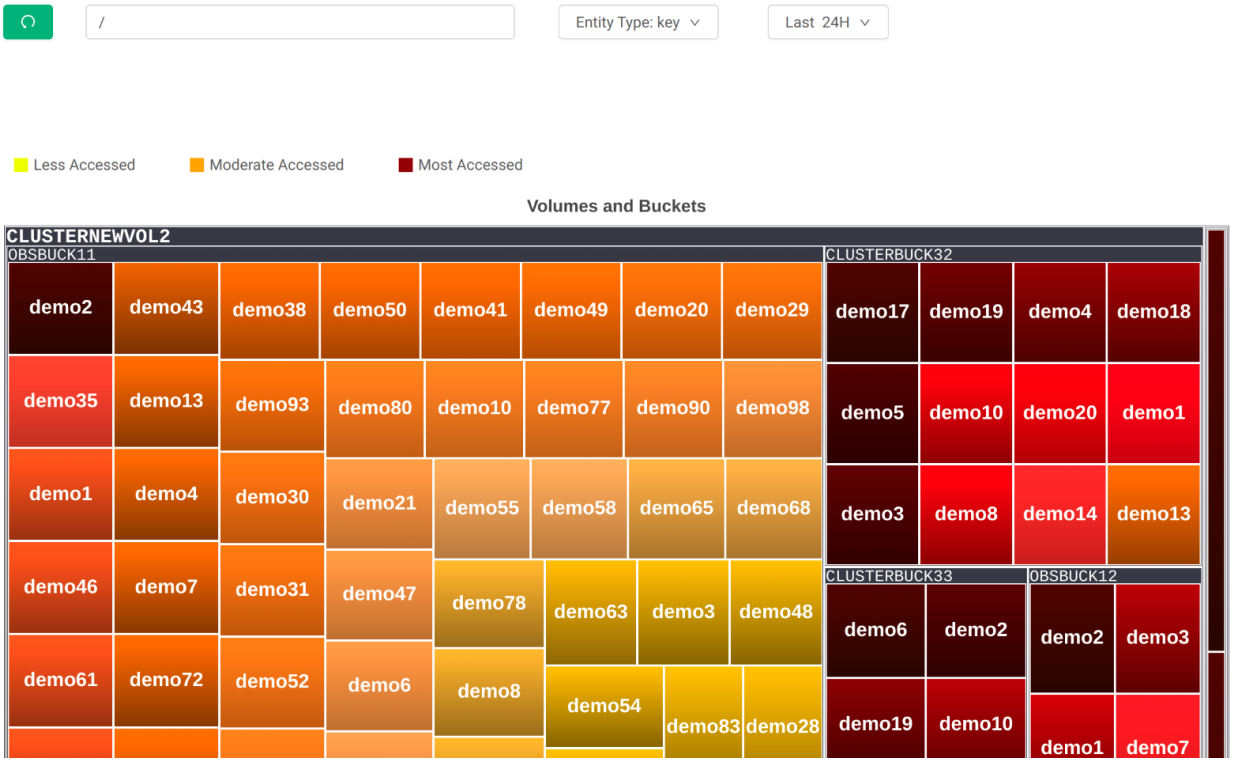
If you use Legacy buckets, you can view the buckets for heatmap representation of keys.

If you use FSO buckets, you can view the buckets for heatmap representation of directories/sub-directories and files.



Key

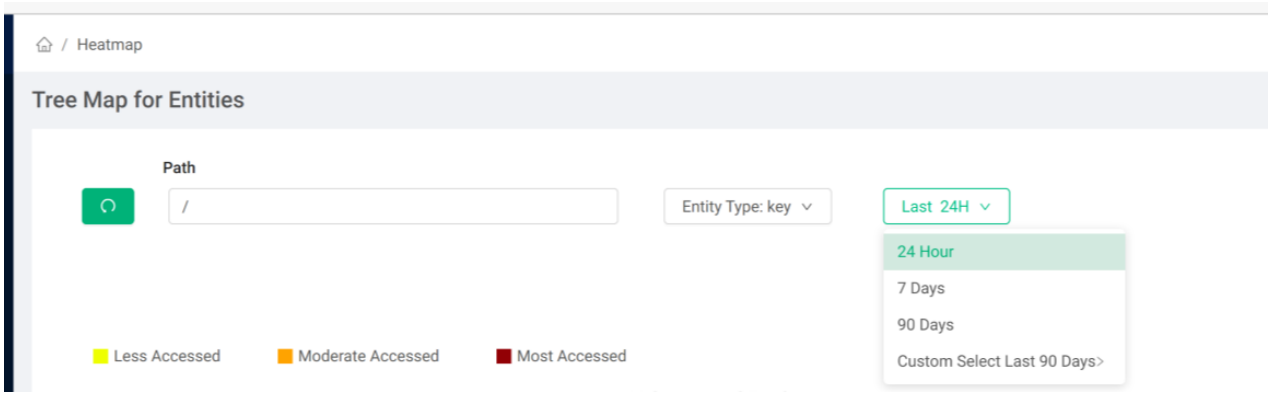
Heatmap at key level is displayed.



Time

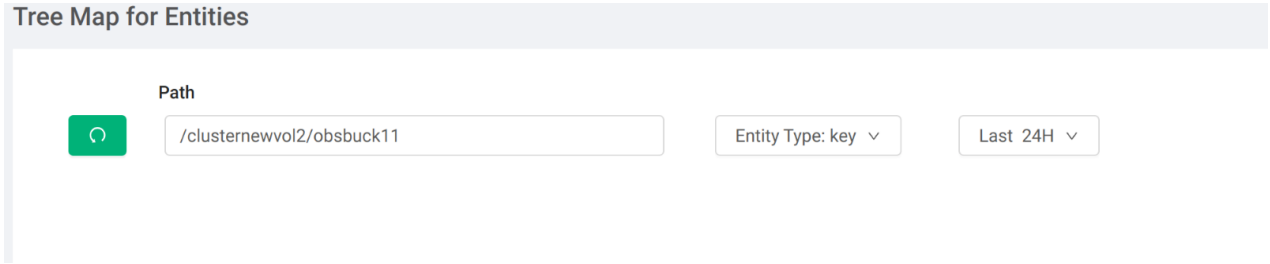
Viewing the volume, bucket, or key based on a timeline.

Heatmap representation is available based on the last 24 hours, 7 Days, and 90 Days read access metadata or based on read access metadata for any custom date < 90 days. The default value is 24 hours.



When you select Key as the entity type and provide a path in the search box, you will see the heatmap representation at the specified path level. This means the application builds a heatmap based on the top 100 keys (most accessed or most heated keys) for the path provided. The path can contain numbers, letters, and forward slash.

The default value in the Path search box is /.



When you hover over an entity in the heatmap, a tooltip is displayed. Entity information like Access count, file name and Max Access Count is displayed.



Container Balancer overview

Container Balancer is a service in Storage Container Manager that balances the utilization of datanodes in an Ozone cluster.

A cluster is considered balanced if for each datanode, the utilization of the datanode (used space to capacity ratio) differs from the utilization of the cluster (used space to capacity ratio of the entire cluster) no more than the threshold. This service balances the cluster by moving containers among over-utilized and under-utilized datanodes.

Note:

1. Container Balancer has a command line interface for administrators. You can run the `ozone admin containerbalancer -h` help command for commands related to Container Balancer.
2. Container Balancer supports both Ratis and Erasure Coded closed containers.

Container balancer CLI commands

You can run the following commands in the cluster.

- To start the service, run the following command `ozone admin containerbalancer start`
- To stop the service, run the following command `ozone admin containerbalancer stop`
- To check the status of the service, run the following command `ozone admin containerbalancer status`

Determining the threshold

Container Balancer balances the utilization of DataNodes in a cluster using the Threshold. Learn how to determine the threshold value before configuring the required parameters.

Ozone's Container Balancer tries to bring the utilization of DataNodes closer to the cluster's average utilization. Utilization is defined as used space divided by capacity. Container Balancer uses the

hdds.container.balancer.utilization.threshold property, also known as *threshold*, to decide which DataNodes are unbalanced. The threshold is a percentage in the range of 0 to 100. The default value is 10 %.

If you set the threshold value to a lower value, say 1 %, Container Balancer tries to bring the utilization of DataNodes close to 1 % of the average cluster utilization. This means moving more containers and having to run for a longer time. At a higher threshold value, say 20 %, the Container Balancer tries to bring the utilization of DataNodes within 20 % of the average cluster utilization. This will move fewer containers, and hence take less time.

Cloudera recommends lowering the threshold if you want the balancer to act more frequently.

If you have a 90-node cluster with 18 PB capacity out of which Ozone, other processes, and files have used 14PB. You added 10 more nodes with a total capacity of 2PB to the cluster. You want to run the container balancer with the default threshold at 10%.

The utilization average of this cluster is Total capacity used in the cluster (14PB) / Total capacity of the cluster (18PB + 2PB) * 100 = 70%

Container Balancer tries to move the containers between over-utilized and under-utilized nodes over multiple iterations to get individual datanodes utilization closer to the cluster's average utilization.

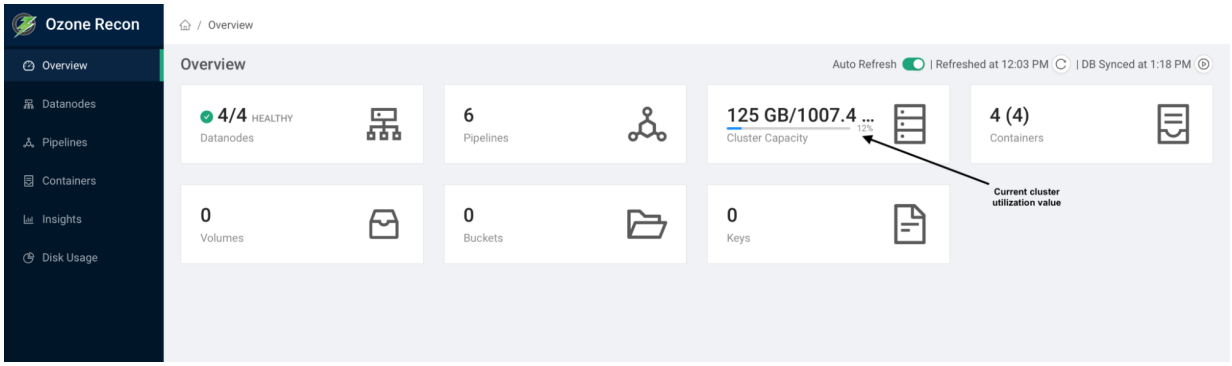
- Over-utilized nodes have utilization greater than the average cluster utilization by the threshold percentage. For example, since the threshold is 10%, the utilization of nodes >80% is over-utilized.
- Under-utilized nodes have utilization less than the average cluster utilization by the threshold percentage. For example, since the threshold is 10%, the utilization of nodes <60% is under-utilized.
- The newly added host with a utilization of 0% must be part of the under-utilized nodes.

Choosing an appropriate value for the threshold

To select an appropriate threshold value based on your cluster utilization, perform the following steps:

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Open Recon Web UI
5. Go to the Overview page. The Overview page displays the cluster's current used space and total capacity. The cluster utilization is displayed as a percentage value.



6. Click Datanodes on the left navigation pane. The utilization value of each Datanode is displayed in the Storage Capacity column.
7. Determine an appropriate threshold value using your cluster's utilization and the utilization of the Datanodes.

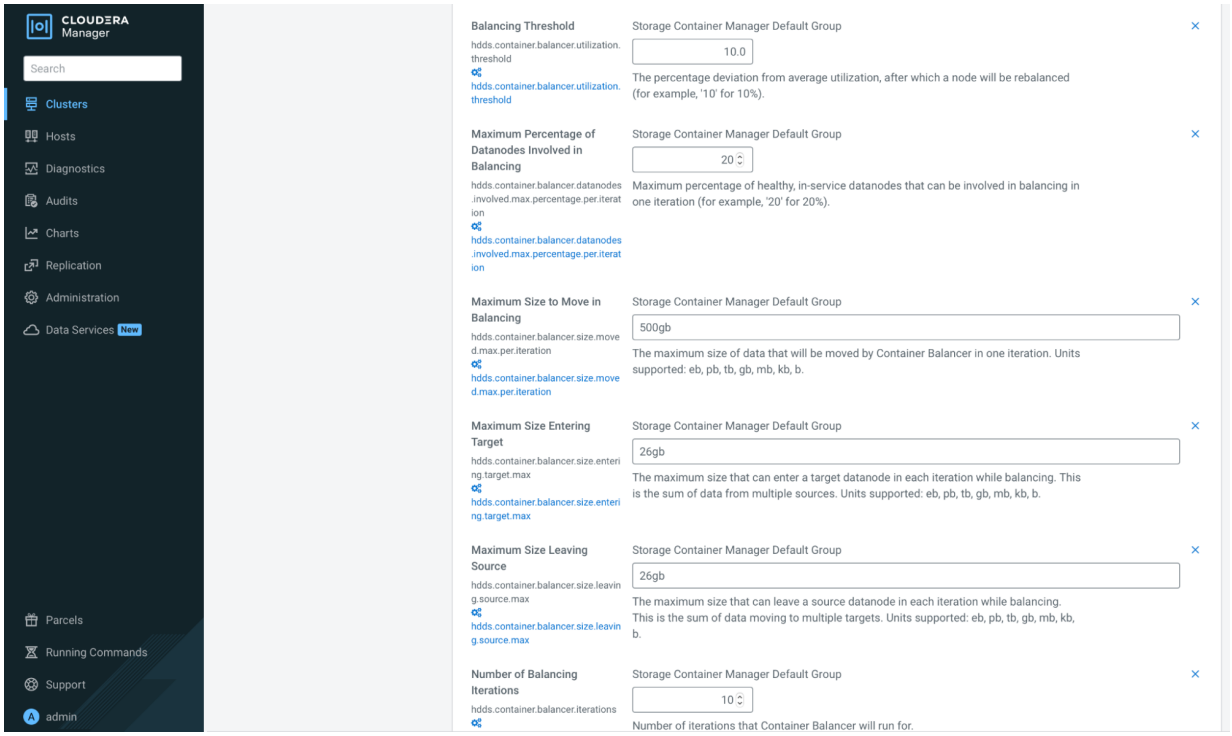
   For example, if your cluster utilization is at 70%, some over-utilized Datanodes have utilizations around 95%, and other Datanodes are around 60%, then you can set the threshold value to 1%.
8. Set the threshold value in Cloudera Manager. For more information, see Configuring container balancer service.

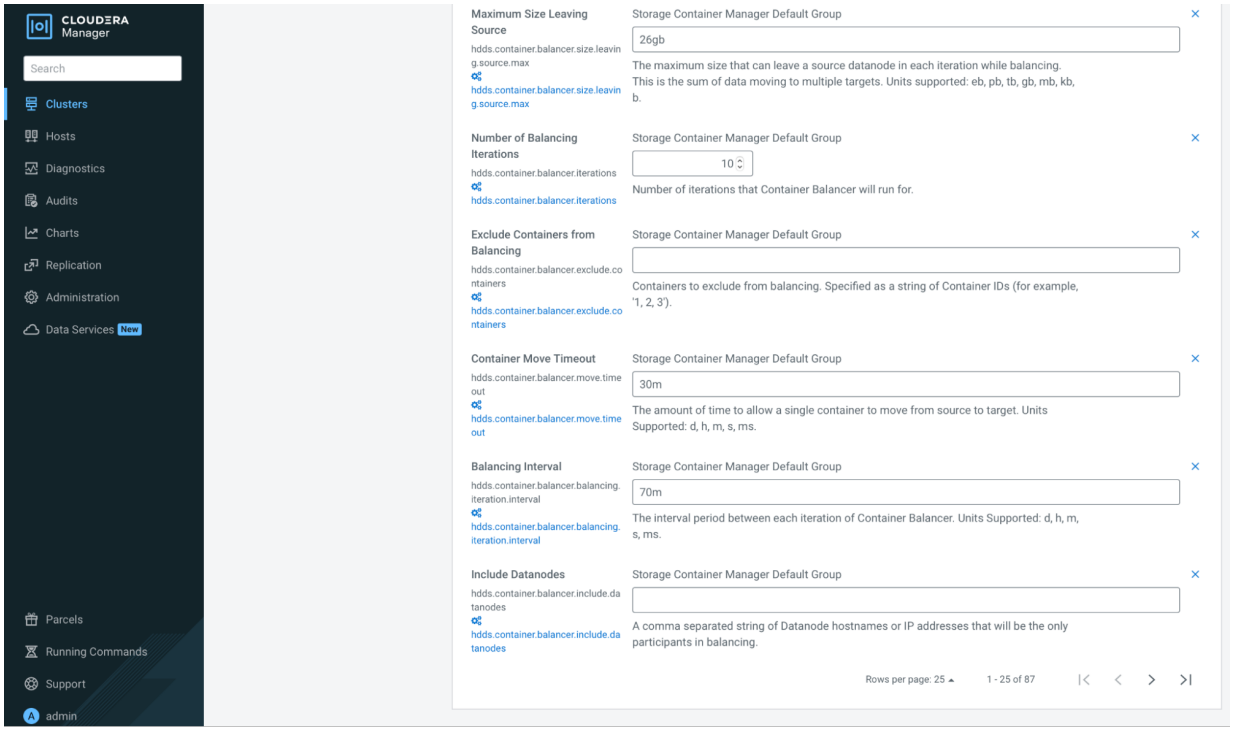
Configuring container balancer service

To use container balancer using Cloudera Manager, perform the following steps.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Go to Configurations
5. You can filter configurations for Container Balancer by selecting the Scope as Storage Container Manager or search for hdds.container.balancer





6. You can now set the values for the Container Balancer configurations.

Activating container balancer using Cloudera Manager

You can activate and deactivate container balancer feature using Cloudera Manager. Perform the following steps to activate or deactivate the feature.

Activating container balancer through Cloudera Manager

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Click Actions
5. Click Activate Container Balancer

Deactivating container balancer through Cloudera Manager

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Click Actions
5. Click Deactivate Container Balancer

Disk Balancer overview

Ozone disk balancer is a tool that evenly spreads data across different storage disks inside a single DataNode.

Apache Ozone effectively distributes containers across multiple DataNodes in a cluster, ensuring cluster-wide balancing. However, within a single DataNode, individual disks can still drift out of balance over their operational lifetime. Disk imbalance generally occurs due to:

- Adding new disks to expand a DataNode's storage capacity.
- Replacing broken disks with new, empty drives.
- Massive deletions of blocks or replicas that disproportionately empty certain disks.

Uneven disk utilization creates performance bottlenecks. Over-utilized disks become I/O hotspots, degrading the DataNode's overall throughput and increasing the risk of failure. To resolve this, Ozone features an intra-node Disk Balancer that automatically detects imbalances and evenly distributes data across all volumes within a single DataNode.

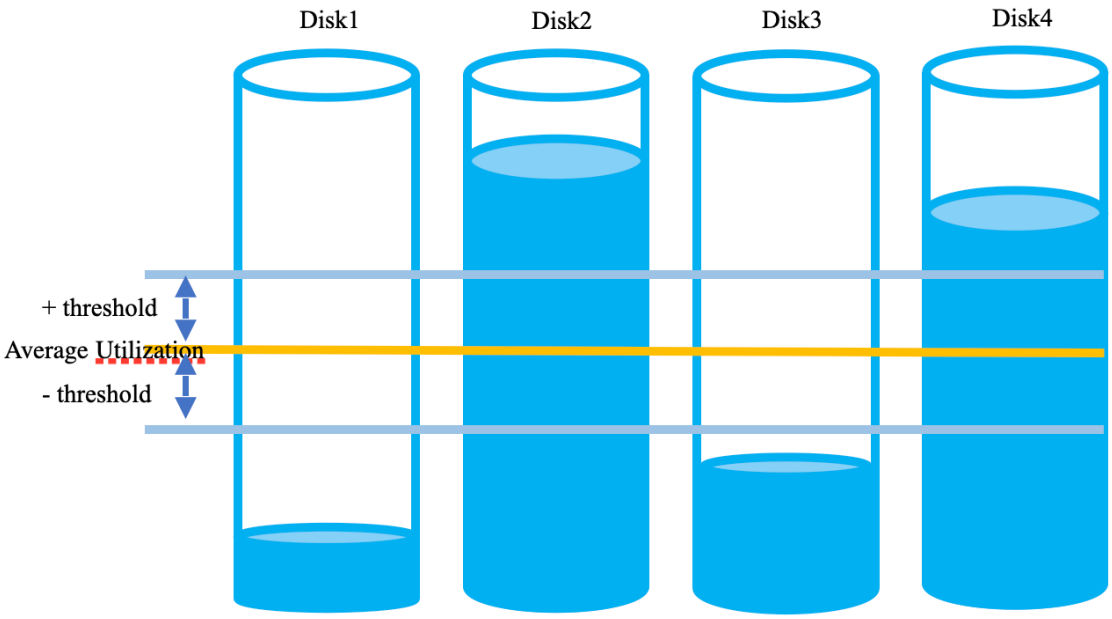
Architecture and how it works

The Disk Balancer works entirely within the local Datanode to securely shift data from heavily utilized disks to under-utilized ones without impacting cluster-wide replication or involving the Storage Container Manager (SCM).

Volume data density

Disk Balancer detects imbalance using a metric called Volume Data Density, which measures how far a disk's individual utilization deviates from the node's average utilization.

Figure 6: Uneven data spread across different disks



The metrics are calculated as follows:

- Average utilization = (Total capacity - Total free space) / Total capacity
- Volume utilization = (Disk capacity - Disk free space) / Disk capacity
- Volume Data Density = Volume utilization - average Utilization

A disk is marked as a candidate for balancing if its Volume Data Density exceeds a configurable threshold (default is 10.0%). The Disk Balancer then moves the containers from most utilised disk to the least utilised disk. Disk Balancer can be triggered manually by CLI commands.

Container move process

To ensure data is always consistent and production I/O isn't interrupted, Disk Balancer uses a strictly safe moving process:

1. **Select volumes**: Identifies the most over-utilized volume as the source and the most under-utilized volume with sufficient space as the destination.
2. **Select containers**: Picks eligible containers from the source volume. By default, only CLOSED and QUASI_CLOSED containers are eligible for moving.
3. **Copy and import**: A temporary copy of the container is created and imported onto the destination disk. The copy transitions into a RECOVERING state and is imported as a new container.
4. **Lazy deletion**: After a successful move and metadata update, the original container is scheduled for lazy deletion (by default, 5 minutes later).

Authentication and authorization

Authentication verifies client identities during direct Remote Procedure Call (RPC) communication with DataNodes, while authorization controls which users are permitted to perform Disk Balancer administrative operations.

Disk Balancer CLI commands communicate directly with DataNodes through RPC. SCM is used to discover active datanodes, but does not participate in the actual Disk Balancer control operations.

Authentication

In secure clusters using Kerberos, the DataNode principal must be properly configured for RPC authentication in ozone-site.xml so the client can verify the DataNode's identity:

```
<property>
  <name>hdds.datanode.kerberos.principal</name>
  <value>dn/[***_HOST***]@REALM.TLD</value>
</property>
```

where, *[***_HOST***]* is the DataNode service principal. Each DataNode will substitute *[***_HOST***]* with its own fully qualified hostname at startup. The *[***_HOST***]* placeholder allows using the same configuration setting on all DataNodes.

Important: Without this configuration, Disk Balancer commands will fail with authentication errors in secure clusters. The client uses this principal to verify the DataNode's identity when establishing RPC connections.

Authorization

The Datanode performs authorization checks based on the ozone.administrators property.

- **Admin operations (start, stop, and update)**: Require the authenticated user to be listed in ozone.administrators property or a group inside ozone.administrators.groups property. By default (if unconfigured), only the OS user running the Datanode daemon possesses administrative access.
- **Read-only operations (status and report)**: Do not require administrator privileges. Any authenticated user can monitor Disk Balancer status and generate reports.

Important:

- By default, if ozone.administrators property is not configured, only the user who launched the DataNode service can start, stop, or update DiskBalancer. This means that in a typical deployment where the DataNode runs as user dn, only that user has administrator privileges for Disk Balancer operations.
- After updating the ozone.administrators configuration, restart the DataNode service for the changes to take effect.

Enabling authorization for additional users:

To allow other users to perform Disk Balancer administrator operations (start, stop, update), configure the ozone.administrators property in ozone-site.xml.

Examples:

- For authorizing single user,

```
<property>
  <name>ozone.administrators</name>
  <value>scm</value>
</property>
```

- For authorizing multiple single users,

```
<property>
  <name>ozone.administrators</name>
  <value>scm,hdfs</value>
</property>
```

- For authorizing groups,

```
<property>
  <name>ozone.administrators.groups</name>
  <value>ozone-admins,cluster-operators</value>
</property>
```

Configuring disk balancer

Configure Disk Balancer feature flags, operational settings, and security permissions in ozone-site.xml to safely manage intra-node data migration across Datanode volumes.

The Disk Balancer is enabled by default. If you need to turn the feature off, set the hdds.datanode.disk.balancer.enabled property to false in your ozone-site.xml file on your Datanodes.

```
<property>
  <name>hdds.datanode.disk.balancer.enabled</name>
  <value>false</value>
</property>
```

Important:

- When disabled, the Disk Balancer service will not initialize, and CLI commands will be rejected.
- Although the Disk Balancer service is enabled by default and runs within the DataNode, it does not rebalance data automatically. The service remains idle until a user explicitly initiates the data movement using the Disk Balancer CLI command.

You can tune the Disk Balancer using the following configuration properties:

Table 4: Disk Balancer configurations

Property	Default value	Description
hdds.datanode.disk.balancer.enabled	true	Enables or disables the Disk Balancer feature on the Datanode.
hdds.datanode.disk.balancer.volume.density.threshold.percent	10.0	The deviation percentage from average utilization required to trigger balancing.
hdds.datanode.disk.balancer.max.disk.throughputInMBPerSec	10	The maximum bandwidth (MB/s) the Disk Balancer is allowed to consume, ensuring no disruption to production I/O.
hdds.datanode.disk.balancer.parallel.thread	5	The maximum number of threads dedicated to concurrent container moves.
hdds.datanode.disk.balancer.service.interval	60s	Specifies the time interval at which the Disk Balancer service checks for volume imbalance and reloads configuration settings.

Property	Default value	Description
hdds.datanode.disk.balancer.stop.after.disk.even	true	Controls whether the Disk Balancer service automatically stops after all volume data densities drop within the threshold.
hdds.datanode.disk.balancer.replica.deletion.delay	5m	The delay time before the source container is deleted after a successful move.
hdds.datanode.disk.balancer.container.states	CLOSED, QUASI_CLOSED	A comma-separated list of container states eligible for moving.
hdds.datanode.disk.balancer.container.choosing.policy	org.apache.hadoop. ozone.container. diskbalancer.policy. DefaultContainerChoosingPolicy	The policy for selecting source and destination volumes and eligible containers to move.
hdds.datanode.disk.balancer.service.timeout	300s	The maximum timeout duration allowed for Disk Balancer service operations.
hdds.datanode.disk.balancer.should.run.default	false	Specifies whether the Disk Balancer service should run by default.

Example configuration:

```
<configuration>
  <!-- Feature Activation -->
  <property>
    <name>hdds.datanode.disk.balancer.enabled</name>
    <value>true</value>
  </property>

  <!-- Threshold and Operational Limits -->
  <property>
    <name>hdds.datanode.disk.balancer.volume.density.threshold</name>
    <value>10.0</value>
  </property>
  <property>
    <name>hdds.datanode.disk.balancer.bandwidth.per.volume.in.mb</name>
    <value>20</value>
  </property>

  <property>
    <name>hdds.datanode.disk.balancer.parallel.threads</name>
    <value>5</value>
  </property>

  <property>
    <name>hdds.datanode.disk.balancer.container.states</name>
    <value>CLOSED,QUASI_CLOSED</value>
  </property>

  <property>
    <name>hdds.datanode.disk.balancer.stop-after-disk-even</name>
    <value>true</value>
  </property>
</configuration>
```

Disk balancer CLI commands

The disk balancer operations are manually triggered and managed using the `ozone admin datanode diskbalancer` command.

Important: Disk Balancer is enabled by default on Datanodes. To disable the service and block CLI operations, set hdds.datanode.disk.balancer.enabled property to false in ozone-site.xml.

Use the following CLI commands to manage and monitor the Disk Balancer service:

- To start the service:

```
ozone admin datanode diskbalancer start
```

- To stop the service:

```
ozone admin datanode diskbalancer stop
```

- To update the configuration:

```
ozone admin datanode diskbalancer update
```

- To check the status of the service:

```
ozone admin datanode diskbalancer status
```

Running the `status` command displays the following information:

Field name	Description
Datanode	Name or IP address of the target node.
Status	Current execution state of the process (example, RUNNING, STOPPED, or PAUSED).
Threshold(%)	Configured imbalance threshold percentage for triggering disk balancing.
BandwidthInMB	Maximum bandwidth allocated to the operation in megabytes per second.
Threads	Number of active concurrent worker threads assigned to the process (Number of threads = Number of containers moved parallely).
StopAfterDiskEven	Boolean flag indicating if execution halts once disk usage balances evenly (true or false).
ContainerStates	Displays which container is eligible for the move process. (by default, CLOSED or QUASI_CLOSED).
SuccessMove	Total count of successfully relocated containers or data blocks.
FailureMove	Total count of failed container or block relocation attempts.
BytesMoved(MB)	Volume of data successfully transferred so far, measured in megabytes.
EstBytesToMove(MB)	Remaining volume of data estimated to move, measured in megabytes.
EstTimeLeft(min)	Calculated time remaining until job completion, measured in minutes.

- To get the report of the service:

```
ozone admin datanode diskbalancer report
```

The `report` command output displays the volume details for each DataNode.

Sample output:

```
Datanode: 59c14bfa-1ccd-45e4-83e6-8c2c3a5de873
Aggregate VolumeDataDensity: 0.00%
IdealUsage: 8.00% | Threshold: 10.00% | ThresholdRange: (0.00%, 18.00%)

Volume Details:

StorageID                                | StoragePath     | OzoneCapacit
y | OzoneAvailable | OzoneUsed | ContainerPreAllocatedSpace | EffectiveU
sedSpace | Utilization | VolumeDensity
DS-57df847c-49a9-4b3e-84a5-b851bc58e2be   /data/hdds/hdds   1006.75 GB
   926.22 GB        4.18 MB     0 B                          80.53 GB
        8.00%         0.00%
```

The following table details the command-line options, descriptions, and usage examples available for Disk Balancer CLI operations.

Table 5: Disk Balancer command options

Option	Description	Example
*[***DATANODE-ADDRESS***]*	One or more Datanode addresses as positional arguments. Addresses can be: • Hostname (example, DN-1): uses default CLIENT_RPC port (19864) • Hostname with port (example, DN-1:19864) • IP address (example, 192.168.1.10) • IP address with port (example, 192.168.1.10:19864) • Stdin (-): Reads DataNode addresses from standard input, one per line	DN-1 DN-1:19864 192.168.1.10 -
--in-service-datanodes	Queries SCM for all IN_SERVICE DataNodes and executes the command on all of them.	--in-service-datanodes
--json	Formats command output as JSON.	--json
-t or --threshold-percentage	Volume density threshold percentage (default: 10.0). Used with `start` and `update` commands.	-t 5 --threshold-percentage 5.0
-b or --bandwidth-in-mb	Maximum disk bandwidth in MB/s (default: 10). Used with `start` and `update` commands.	-b 20 --bandwidth-in-mb 50
-p or --parallel-thread	Number of parallel threads (default: 5). Used with `start` and `update` commands.	-p 5 --parallel-thread 10
-s or --stop-after-disk-even	Automatically stops the balancer after disks are balanced (default: true). Used with start and update commands.	-s false --stop-after-disk-even true
-c or --container-states	Comma-separated list of container lifecycle state names eligible to be moved between disks. Used with start and update commands.	-c CLOSED,QUASI_CLOSED --container-states CLOSED

Important:

- When Storage Container Manager (SCM) is offline, do not use the --in-service-datanodes option. Instead, explicitly specify the target DataNode address to perform disk balancer operations.
- Before starting Disk Balancer, review the disk balancer report to determine the optimal configuration parameters for best results.

Example commands:

- Start disk balancer:
  - Start disk balancer on multiple DataNodes:

    ```
    ozone admin datanode diskbalancer start DN-1 DN-2 DN-3
    ```

  - Start disk balancer on all IN_SERVICE and HEALTHY DataNodes:

    ```
    ozone admin datanode diskbalancer start --in-service-datanodes
    ```

  - Start disk balancer with configuration parameters:

    ```
    ozone admin datanode diskbalancer start DN-1 -t 5 -b 20 -p 5
    ```

Or using the long form:

```
ozone admin datanode diskbalancer start DN-1 --threshold-percentage 5 -b
 20 -p 5
```

- Read datanode addresses from stdin:

```
echo -e "DN-1\nDN-2" | ozone admin datanode diskbalancer start -
```

- Start disk balancer with json output:

```
ozone admin datanode diskbalancer start DN-1 --json
```

- Stop disk balancer:
  - Stop disk balancer on multiple DataNodes:

```
ozone admin datanode diskbalancer stop DN-1 DN-2 DN-3
```

  - Stop disk balancer on all IN_SERVICE and HEALTHY DataNodes:

```
ozone admin datanode diskbalancer stop --in-service-datanodes
```

  - Stop disk balancer with JSON output:

```
ozone admin datanode diskbalancer stop DN-1 --json
```

- Update disk balancer configuration:
  - Update multiple parameters:

```
ozone admin datanode diskbalancer update DN-1 -t 5 -b 50 -p 10
```

  - Update on all IN_SERVICE and HEALTHY DataNodes:

```
ozone admin datanode diskbalancer update --in-service-datanodes -t 5
```

    Or using the long form:

```
ozone admin datanode diskbalancer update --in-service-datanodes --thresh
old-percentage 5
```

  - Update with JSON output:

```
ozone admin datanode diskbalancer update DN-1 -b 50 --json
```

- Get disk balancer status:
  - Get status from multiple DataNodes:
    - ozone admin datanode diskbalancer status DN-1 DN-2 DN-3
  - Get status from all IN_SERVICE and HEALTHY DataNodes:

```
ozone admin datanode diskbalancer status --in-service-datanodes
```

  - Get status in JSON format:

```
ozone admin datanode diskbalancer status --in-service-datanodes --json
```

- Get disk balancer report:

- Get report from multiple DataNodes:

```
ozone admin datanode diskbalancer report DN-1 DN-2 DN-3
```

- Get report from all IN_SERVICE and HEALTHY DataNodes:

```
ozone admin datanode diskbalancer report --in-service-datanodes
```

- Get report in JSON format:

```
ozone admin datanode diskbalancer report --in-service-datanodes --json
```

Ozone Replication Manager overview

Learn about Ozone's Replication Manager (RM), how it performs the throttling of tasks, and the configurations you can use to control the throttling tasks.

The RM is a service which runs inside the leader Storage Container Manager (SCM) daemon in a cluster. Its role is to make both RATIS and Erasure Coded data durable. It does this by periodically checking the health of all the containers in the cluster, and taking actions for any containers which are not healthy. Those actions can be creating new replicas of RATIS containers, reconstructing Erasure Coded data, closing replicas, removing any unnecessary replicas, and so on.

The RM process is split into stages. First, it checks containers and identifies those with problems. Second, it takes actions on the problematic containers.

The thread that checks each container in the first stage runs periodically. You can configure its interval by using the hdds.scm.replication.thread.interval configuration. Default is 5 minutes.

The threads that take action on problematic containers in the second stage also run periodically with a default of 30 seconds. You can configure them by using the hdds.scm.replication.under.replicated.interval and hdds.scm.replication.over.replicated.interval configurations.

Ozone Replication Manager throttling of tasks

Learn about the different commands Ozone's Replication Manager (RM) schedules for throttling of tasks.

To protect the cluster from being overloaded with the RM tasks, it is important that a limited number of these tasks run on the cluster at any time. The load on datanodes can change over time and this impacts their speed at processing the tasks. In addition to throttling concurrent work, it is important that the RM, running inside SCM, does not queue too many tasks on the datanodes.

RM schedules the following types of commands which it throttles:

- Replicate container commands

  Creates additional copies of a container to resolve replication issues and to allow nodes to be decommissioned safely.
- Delete container replica commands

  Resolves over replication and also deletes containers in unexpected states.
- EC reconstruction commands

  Recovers lost EC replicas. These are the most expensive commands.

Replicate container commands

Learn about replicate container command, types of this command, and the configuration that you can use to control it, and

Ozone's Replication Manager (RM) sends this command to a datanode that contains a replica of the container, instructing the datanode to replicate this to another target datanode. The replicate container command is sent to the datanode with the least commands queued.

If all sources have too many commands queued, the container cannot be replicated, and the command is re-queued to be tried again later.

There are two types of replication commands; simple replication and EC reconstruction. The simple replication and EC reconstruction commands share the same datanode queue and worker thread pool on the datanode, and hence they have a single combined limit. As EC reconstruction commands are more expensive to process than simple replication commands, the EC reconstruction commands are given a weighting so that queuing one command counts 1 * weight to the limit. The default weight is currently 3. For more information, see *EC reconstruction commands*.

Use the hdds.scm.replication.datanode.replication.limit configuration to adjust the limit of the number of simple replication and EC reconstruction commands that can be queued on a datanode.

The balancer and low priority commands

The Container Balancer service also sends replicate commands through the RM API to balance the utilization of datanodes in an Ozone cluster.

The Container Balancer can create a large number of replicate container commands on the datanode, so that the Container Balancer does not impact the more important work performed by RM. The Replicate Container commands can be sent with two priorities; normal and low. The Container Balancer always sends low priority replicate container commands, while the RM always sends normal priority commands. Low priority commands do not count towards the command limit configured by the hdds.scm.replication.datanode.replication.limit configuration. If the datanode has normal priority commands queued, the low priority commands are not processed. That way, if there is a large amount of Balancer work scheduled, and some essential replication work is required, replication work gets priority.

The Container Balancer also schedules commands with a larger timeout, to give time for its work to complete and also to cater for any higher priority commands which might slow its progress.

Related Information

EC reconstruction commands

Delete container replica commands

Learn about the delete container replica commands, its priroty, and the configuration to control it.

The delete container commands are throttled in much the same way as for the replicate container commands. When a delete command is attempted, the current command count is checked and if the datanode is overloaded, another replica is tried or the container is re-queued and attempted again later.

Use the hdds.scm.replication.datanode.delete.container.limit configuration to adjust the limit on the number of delete commands that can be queued on a datanode.

The balancer and delete commands

Unlike the replication commands, there is no priority ordering for delete container replica commands scheduled by the Container Balancer, for several reasons:

- Delete is less important than replication, as a delayed delete cannot result in data loss.
- The balancer delete commands are triggered by the completion of a replicate command and this rate of completion naturally throttles the delete.
- Delete commands are less resource intensive, and hence the datanode should be able to deal with a large number quickly.

EC reconstruction commands

Learn about EC reconstruction commands, the configurations you can use to control it, and how the Replication Manager (RM) replicates data from a decommissioning host to other datanodes.

The EC reconstruction commands are the most expensive commands scheduled on datanodes. A reconstruction command recovers between 1 and the EC scheme parity number replicas, and reads from the EC scheme data number replicas.

The EC reconstruction commands share the same limits on the datanodes as replication commands, but reconstruction commands are given a weighting (default 3 at the current time) as they are more expensive for the coordinator node to run.

Use the hdds.scm.replication.datanode.reconstruction.weight configuration to configure the weight given to EC reconstruction commands, and the hdds.scm.replication.datanode.replication.limit configuration to adjust the total limit of replication and EC reconstruction commands.

EC and decommissioning

RM is responsible for replicating data from a decommissioning host to other datanodes so that the decommissioning task can be completed.

When a node hosting a RATIS container is decommissioned, there are generally 3 sources available for the container replicas. One on the decommissioning host, and then 2 others on somewhat random nodes across the cluster. This allows the decommissioning load and the speed of decommission to be shared across many more nodes.

For an EC container, the decommissioning host is likely the only source of the replica which needs to be copied and hence the decommission will be slower.

A host which is decommissioning is generally not used for RATIS reads unless there are no other nodes available, but it would still be used for EC reads to avoid online reconstruction. As decommission progresses on the node, and new copies are formed, the read load declines over time. Furthermore, decommissioning nodes are not used for writes, so they should be under less load than other cluster nodes.

Due to the reduced load on a decommissioning host, it is possible to increase the number of commands queued on a decommissioning host and also increase the number of commands allowed to run in parallel.

When a datanode switches to a decommissioning state, it adjusts the size of the replication supervisor thread pool higher, and if the node returns to the in-service state, then it returns to the lower thread pool limit.

Similarly, RM increases the limit for the number of replication commands that can be queued on a datanode that is decommissioning or entering maintenance. SCM can allocate more commands to the decommissioning host, as it should process them more quickly due to the lower load and increased threadpool.

You can use the hdds.datanode.replication.outofservice.limit.factor configuration to increase the size of a decommissioning datanode’s thread pool and the number of replication commands that can be queued on it.

Configurations for throttling of tasks

Learn about different parameters that you can use to control Replication Manager's (RM) throttling of tasks.

hdds.scm.replication.datanode.replication.limit

Total number of replication commands that can be queued on a datanode. The limit is made up of (number_of_replication_commands + reconstruction_weight * number_of_reconstruction_commands). The default value is 20.

hdds.scm.replication.datanode.reconstruction.weight

The weight to apply to multiple reconstruction commands before adding to the datanode.replication.limit. The default value is 3.

hdds.scm.replication.datanode.delete.container.limit

The total number of delete container commands to be queued on a given datanode. The default value is 40.

hdds.scm.replication.inflight.limit.factor

The overall replication task limit on a cluster is the number of healthy nodes multiplied by the datanode.replication.limit. The hdds.scm.replication.inflight.limit.factor configuration, which should be between 0 and 1, scales that limit down to reduce the overall number of replicas pending creation on the cluster. A setting of 0 disables global limit checking. A setting of 1 effectively disables it by making the limit equal to the above equation. However, if there are many decommissioning nodes on the cluster, the decommissioning nodes will have a higher than normal limit, so the setting of 1 might still provide some limit in extreme circumstances. The default value is 0.75.

hdds.datanode.replication.outofservice.limit.factor

When a datanode is decommissioning its replication thread pool, the hdds.datanode.replication.streams.limit configuration, whose default value is 10, is multiplied by this factor to allocate more threads for replication. On SCM, the limit for any datanode which is not in-service (for example, decommissioning or entering maintenance) is also increased by the same factor. This allows the node to dedicate more resources to replication as it will not be used for writes and will be reduced in priority for reads. The default value is 2.0.

hdds.datanode.replication.streams.limit

The maximum number of simultaneous replication related commands that can run on a single datanode at a time, either by pushing data to a new target, or by coordinating an EC container reconstruction. The default value is 10.

hdds.scm.replication.event.timeout

The amount of time SCM allows for a task scheduled on a datanode to complete. After this duration, the datanode discards the command and SCM assumes that it is lost and schedules another if still relevant. The throttling applied by SCM when scheduling commands should prevent too many commands from being scheduled that can be completed in this interval. The default value is 300 seconds.

Global replication command limit

You can configure a global replication limit by limiting the number of inflight containers pending creation.

The global replication limit is defined by the hdds.scm.replication.inflight.limit.factor configuration. The default value is 0.75.

Global delete limit

Container replica deletes tend to be targeted to a single node, and the datanode already has a thread pool to handle them, which limits the number of deletes running concurrently. There is also no network impact when deleting a container. Therefore, there is no global command limit for delete commands.

Current replication state

To get a view of the overall state of the cluster, the `ozone admin container report` command can be run by an administrator. It returns details from the last RM run, indicating the number of containers with various problem states. This report is cached by each run of the RM check thread. Hence, it can be up to 5 minutes in stale mode.

In addition, the report values are exposed through metrics on the leader SCM process, and various other metrics detailing the current under and over replication queue sizes, number of inflight commands and many other details that can give insight to the throttling and completion of commands on the cluster.

Storage container reconciliation overview

Container reconciliation identifies corrupted or diverged container replicas and repairs them to ensure consistency and identical data across all replicas.

By design, container replicas are identical and contain the same data. Container reconciliation is a repair mechanism that allows administrators to detect and repair container replicas that are corrupted or have diverged from each other. The data within each replica is summarized as a single data checksum. If the replica data diverges, these checksums will differ. After reconciliation,, the replicas will contain the same data with matching checksums. This checksum remains constant even as data is deleted from the container in the background.

he system summarizes the data within each replica as a single data checksum. If the replica data diverges, these checksums will differ. After reconciliation, the replicas are restored to an identical state with matching checksums. This checksum remains constant even as data is deleted from the container in the background.

Restrictions and limitations

Consider the following constraints and behavioral limitations before you use container reconciliation:

- Reconciliation is not supported for containers that are in the OPEN, DELETING, or DELETED states. Reconciling containers in these states fails immediately without any modifications. Reconciliation can only be used to repair data in UNHEALTHY replicas by updating the replica data and corresponding checksums, but not their state.
- Reconciliation is supported for Ratis-replicated containers only. Reconciling an Erasure Coded (EC) container fails immediately without any modifications.
- Containers with a replication factor of 1 cannot be reconciled because no peer replicas exist to compare against. Similarly, containers with a replication factor of 3 that have zero or one replica available cannot be reconciled.
- After running the ozone admin container reconcile command on a CLOSED RATIS/THREE container, checksums might match, but the replica state does not change, even if the container was UNHEALTHY. If the container is corrupted again later and the checksums diverge once more, you can run the ozone admin container re concile command multiple times until the container is repaired.
- Storage Container Manager (SCM) can only use replication to automatically recover from failures. SCM will not automatically schedule reconciliation. Since replication requires at least one source replica not in the UNHE ALTHY state, reconciliation will need to be run manually if all replicas are UNHEALTHY.

Reconciling storage containers

Reconcile storage containers by detecting, repairing and, monitoring corrupted or inconsistent container replicas to ensure data consistency and uniformity across all replicas.

Detecting corrupted or diverged containers

- **Using Recon web UI**:
  1. Sign in to Cloudera Manager.
  2. Click Clusters in the left navigation.
  3. Select the Ozone service.
  4. In the Web UI drop-down menu, select Recon Web UI. The Recon UI page is displayed.
  5. Click Containers in the left navigation.
  6. Click the Mismatched Replicas tab. The Mismatched Replicas tab displays all containers whose replicas have different data checksums. Containers displayed in this tab are eligible for reconciliation.
- **Using the command line**:
  1. Open the Terminal on your device.
  2. Run the following command to see the reconciliation status of specific containers:

     ```
     ozone admin container reconcile --status <CONTAINER_ID> [<CONTAINER_ID>
      ...]
     ```

     This command displays the following information about each container and its replicas:

     - Container state

- Replication configuration
- Data checksum for each replica
- Whether the data checksums of all the container's replicas match
- DataNode details such as ID, hostname, and IP address for each replica

Triggering container reconciliation

1. Open the Terminal on your device.
2. Run the following command to trigger reconciliation for one or more containers:

```
ozone admin container reconcile <***CONTAINER-ID***> [<***CONTAINER-ID***>
 ...]
```

Result: When you trigger reconciliation, the system performs following actions:

1. The Storage Container Manager sends a reconciliation command to all the DataNodes hosting replicas of the container.
2. Each DataNode compares its container data checksums with those of the peer replicas.
3. The DataNodes automatically repair the missing or the corrupt data by performing the following actions:
   - Copying missing blocks from peer replicas
   - Retrieving missing chunks within blocks
   - Repairing chunks with incorrect checksums
4. The Data Nodes scan the container data in the background to verify the integrity of the repaired data.



Important: Reconciling containers might repair their data, but it does not change their state.

Monitoring reconciliation progress

- **Using Recon web UI**:
  1. Sign in to Cloudera Manager.
  2. Click Clusters in the left navigation.
  3. Select the Ozone service.
  4. In the Web UI drop-down menu, select Recon Web UI. The Recon UI is displayed.
  5. Click Containers in the left navigation.
  6. Click the Mismatched Replicas tab.
  7. Verify that the container is no longer displayed in the Mismatched Replicas tab.
- **Using the command line**:
  1. Open the Terminal on your device.
  2. Run the following command to see the reconciliation status of specific containers:

     ```
     ozone admin container reconcile --status <***CONTAINER-ID***>
      [<***CONTAINER-ID***> ...]
     ```

     This command displays the following information about each container and its replicas:

     - Container state
     - Replication configuration
     - Data checksum for each replica
     - Whether the data checksums of all the container replicas match
     - DataNode details such as ID, hostname, and IP address for each replica
- **Review DataNode logs** for the reconciliation progress messages.

Managing Ozone quota

The Ozone shell is the primary command line interface for managing the quota of volumes and buckets. Understand what Ozone quotas are, how to define and manage Storage space level and Namespace quota, and the commands available to you that will hel you to manage volumes and buckets.

Apache Ozone provides a resource management feature enabling you to define and manage quota for using space and namespace at volume and bucket levels. Resource usages for space and namespace are captured by default for volumes and buckets. This feature allows you to:

- Define the space quota of the volume and bucket (storage space of volume and bucket).
- Define the namespace quota of the volume (the number of buckets in the volume).
- Define namespace quota for the bucket (number of files or keys or directories recursively in the bucket).

Understanding quota

As an administrator, understand the Storage space level and Namespace quota that you can define and manage for the volumes and buckets .

To use the quota feature, you must be an administrator.

Storage Space level quota

- Volume storage space quota: Defines the overall space usage limit for the buckets under a volume.
- Bucket storage space quota: Defines the maximum space usage limit by keys and files recursively under a bucket.

Namespace quota

- Volume namespace quota: Defines the number of buckets which can be present in a volume.
- Bucket namespace quota: The namespace quota for the bucket is the number of files or keys or directories inside the bucket recursively.

Storage Space level quota considerations

As an administrator, understand how you can define and manage space level quota for volumes and buckets.

- Only administrators can define the volume and bucket storage space.
- By default, the quota for volume and bucket is not enabled and the size is unrestricted. To enable quota, you need to set quota limits for the volumes and buckets using CLI options.
- After you enable the volume quota, the total bucket quota cannot exceed the volume quota.
- You can enable the bucket quota without enabling the volume quota. The size of the bucket quota is unrestricted as the volume quota is not set.
- Volume quota comes into effect only when the bucket quota is set. You need to set the bucket quota before setting the volume quota for the volume quota to come into effect. This is because Ozone can only check the usedBytes of the bucket when we write the key and volume quota can monitor the bucket for usedBytes.
- You cannot disable the bucket quota until you enable the volume quota.
- When a volume has linked (for example, symlink) buckets, the linked bucket space usage is not restricted by this volume. Linked bucket space usage is restricted by source volume quota and monitored by source volume and source bucket quota. For example, adding keys and files to the bucket (symlink) linked to the volume does not impact the linked volume.
- A quota value with -2 for the volume and buckets represents old volume and buckets where space usage is not captured. The quota feature does not support such volumes and buckets. This needs a recalculation of space usage to support the quota feature during the upgrade. Space usage recalculation is required during the upgrade as enabled by default to rectify incorrect values as the quota feature was not supported previously.

Namespace quota considerations

As an administrator, understand how you can define and manage namespace quota for volumes and buckets.

- Administrators can define the namespace quota of volume and bucket.
- By default, the namespace quota for volume and bucket is not enabled and hence have unlimited quota.
- After enabling the volume namespace quota, the buckets under the volume cannot exceed the volume namespace quota.
- After enabling the bucket namespace quota, the keys under the bucket cannot exceed the bucket namespace quota.
- When a volume has linked (for example, symlink) buckets, the linked bucket is counted as volume namespace quota by the volume having this bucket as linked. A linked bucket does not define a separate namespace quota, it refers to the namespace quota of the source bucket for keys inside the linked bucket.
- A quota value with -2 for the volume and buckets represents old volume and buckets where space usage is not captured. The quota feature does not support these volumes and buckets This needs recalculation of namespace usages to support the quota feature during the upgrade. Space usage recalculation is required during the upgrade as enabled by default to rectify incorrect values as the quota feature was not supported previously.
- For the File System Optimized (FSO) bucket, while files and directories are moving to trash, the trash consumes extra namespace for the below cases:
  - For internal directory, the path of trash in the bucket is /.trash/<user>/<current or timestamp>
  - For the extra path created while moving a file or a directory, the trash is present at certain directory hierarchy.

    The example for the extra path created at the source is /<vol>/<bucket>/dir1/dir2/file.txt

    Scenario 1:

    - Move file.txt to trash while performing the delete operation
    - Trash created with dir1 and dir2 as extra namespace to have same path as source in trash: /<vol>/<bucket>/.trash/<user>/current/dir1/dir2/file.txt
    - This consumes extra name space of 2

    Scenaro 2

    - Move dir2 to trash while perforning the delete operation
    - Trash created with dir1 as extra namespace /<vol>/<bucket>/.trash/<user>/current/dir1/dir2/file.txt
    - This consumes extra namespace of 1 for dir1

    Scenario 3

    - Move dir1 to trash while performing the delete operation. In this case, no extra namespace is required /<vol>/<bucket>/.trash/<user>/current/dir1/dir2/file.txt

Additional quota considerations

Review certain considerations related to FSO buckets and quota limits before proceeding with the commands for managing quota.

- For the File System Optimized (FSO) bucket with recursive deletion of the directory, the release of quota happens asynchronously after subdirectories and files are removed. When a directory is removed, recursive deletion can be in progress in the background.
- When the quota is about to reach the limit, the Ozone clients (in parallel) commit the file, then the file commit is successful for those files meeting the quota and verification will be in the order of first come basis at the backend.

Commands for managing volumes and buckets

As an administrator, you must understand the quota commands that you can use to manage the volumes and buckets.

Commands for managing volumes

Depending on whether you are an administrator or an individual user, the Ozone shell commands enable you to create, delete, view, list, and update volumes. Before running these commands, you must have configured the Ozone Service ID for your cluster from the Configuration tab of the Ozone service on Cloudera Manager.

Creating volume and specifying quota

Only an administrator can create a volume and assign it to a user. You must assign administrator privileges to users before they can create volumes.

Creating a volume:	
Command Syntax	`ozone sh volume create [--namespace-quota=<quotaInNamespace>] [--space-quota=<quotaInBytes>] <uri>`

Command Usage	• Create volume with only space quota:
	`ozone sh volume create --space-quota=<quotaInBytes> <uri>`
	• Create volume with only namespace quota:
	`ozone sh volume create --namespace-quota=<quotaInNamespace> <uri>`
	• Create volume with both space and namespace quota:
	`ozone sh volume create --namespace-quota=<quotaInNamespace> --space-quota=<quotaInBytes> <uri>`

Purpose	Creates a volume with the quota.
Arguments	• --namespace-quota: Specify the number of buckets in a volume. This is an optional parameter. • --space-quota: Specifies the maximum size the volume can occupy in the cluster. This is an optional parameter. • uri: The name of the volume to create in the <prefix>://<Service ID>/<volumename> format.

Examples	`ozone sh volume create --space-quota=2TB o3://ozone1/vol1`
	This command creates a 2 TB volume named vol1. Here, ozone1 is the Ozone Service ID.
	`ozone sh volume create --namespace-quota=100 o3://ozone1/vol1`
	This command sets the namespace quota of vol1 to 100.

Checking namespace and space quota for volume

Command Syntax	`ozone sh volume info <uri>`
Purpose	• Get the quota value and usedNamespace information of the volume.

Arguments	uri: The name of the volume you want to view in the <prefix>://<Service ID>/<volumename> format.

Example	`ozone sh volume info o3://ozone1/vol1`
	• This command gets the quota value and usedNamespace of vol1.
	`{ "metadata" : { }, "name" : "vol1", "admin" : "user1", "owner" : "user1", "quotaInBytes" : 2199023255552, "quotaInNamespace" : 100, "usedNamespace" : 0, "creationTime" : "2023-07-10T06:04:44.284Z", "modificationTime" : "2023-07-10T06:05:58.505Z", "acls" : [{ "type" : "USER", "name" : "user1", "aclScope" : "ACCESS", "aclList" : ["ALL"] }, { "type" : "GROUP", "name" : "staff", "aclScope" : "ACCESS", "aclList" : ["ALL"] }], "refCount" : 0 }`

Updating volume quota

Command Syntax	`ozone sh volume setquota [-hV] [--namespace-quota=<quotaInNamespace>] [--space-quota=<quotaInBytes>] <uri>`

Command Usage	• Update volume with only space quota:
	`ozone sh volume setquota --space-quota=<quotaInBytes> <uri>`
	• Update volume with only namespace quota:
	`ozone sh volume setquota --namespace-quota=<quotaInNamespace> <uri>`
	• Update volume with both space and namespace quota:
	`ozone sh volume setquota --namespace-quota=<quotaInNamespace> --space-quota=<quotaInBytes> <uri>`

Purpose	Updates the quota of the specific volume.

Arguments	• --namespace-quota: Updates the maximum number of buckets this volume can have. • --space-quota: Updates the maximum size the volume can occupy in the cluster. • uri: The name of the volume to update in the <prefix>://<Service ID>/<volumename> format.  **Note:** • You cannot set the space quota of volumes and buckets in decimals. For example, 1.5 TB. • Ensure that the minimum space quota is the defa ult block size * replication factor. If you set the value lesser than the default block size * replication factor, while writing the data (key put) operation, an operation error is displayed.
Example	`ozone sh volume setquota --namespace-quota=1000 --space-quota=10GB o3://ozone1/vol1` This command sets the *VOL1* namespace quota to 1000 and the space quota to 10 GB.

Clearing volume quota

Namespace	
Command Syntax	`ozone sh volume clrquota [-hV] [--namespace-quota] [--space-quota] <uri>`

Command Usage	• Clear volume with only space quota:
	`ozone sh volume clrquota --space-quota <uri>`
	• Clear volume with only namespace quota:
	`ozone sh volume clrquota --namespace-quota <uri>`
	• Clear volume with both space and namespace quota:
	`ozone sh volume clrquota --space-quota --namespace-quota <uri>`

Purpose	• Clear the namespace and space quota of the volume.
Arguments	• --namespace-quota: Clears the namespace quota of a volume. • --space-quota: Clears the space quota of a volume.
Example	• `ozone sh volume clrquota --namespace-quota o3://ozone1/vol1` This command clears the namespace quota of vol1. • `ozone sh volume clrquota --space-quota o3://ozone1/vol1` This command clears the space quota of vol1.

Commands for managing buckets

The Ozone shell commands enable you to create, delete, view, and list buckets. Before running these commands, you must have configured the Ozone Service ID for your cluster from the Configuration tab of the Ozone service on Cloudera Manager.

Creating bucket and specifying quota

Command Syntax	`ozone sh bucket create [--namespace-quota=<quotaInNamespace>] [--space-quota=<quotaInBytes>] <uri>`

Command Usage	• Create volume with only space quota:
	`ozone sh bucket create --space-quota=<quotaInBytes> <uri>`
	• Create volume with only namespace quota:
	`ozone sh bucket create --namespace-quota=<quotaInNamespace> <uri>`
	• Create volume with both space and namespace quota:
	`ozone sh bucket create --namespace-quota=<quotaInNamespace> --space-quota=<quotaInBytes> <uri>`

Purpose	Creates a bucket with the quota.
Arguments	• uri: The name of the bucket to create in the <prefix>://<Service ID>/<volumename>/<bucketname> format. • --namespace-quota: Specify the number of keys, files, and directories in a bucket. This is an optional parameter. • --space-quota: Specifies the maximum size the bucket can occupy in the cluster. This is an optional parameter.

Example	`ozone sh bucket create --space-quota=2TB o3://ozone1/vol1/buck1`
	This command creates a 2 TB bucket named buck1. Here, ozone1 is the Ozone Service ID.
	`ozone sh bucket create --namespace-quota=100 o3://ozone1/vol1/buck1`
	This command sets the namespace quota of buck1 to 100.

Checking Namespace and space quota for bucket

Command Syntax	`ozone sh bucket info <uri>`
Purpose	Get the quota value, usedNamespace, and usedBytes info of the bucket.

Arguments	uri: The name of the bucket you want to view in the <prefix>://<Service ID>/<volumename>/<bucketname> format.

Example	`ozone sh bucket info o3://ozone1/vol1/buck1`
	This command gets the quota value, usedNamespace, and usedBytes of buck1.
	`{ "metadata" : { }, "volumeName" : "vol1", "name" : "buck1", "storageType" : "DISK", "versioning" : false, "usedBytes" : 0, "usedNamespace" : 0, "creationTime" : "2023-07-10T06:12:47.270Z", "modificationTime" : "2023-07-10T06:12:47.270Z", "sourcePathExist" : true, "quotaInBytes" : 1099511627776, "quotaInNamespace" : 100, "bucketLayout" : "LEGACY", "owner" : "user1", "link" : false }`

Setting namespace and space level quota for bucket

Namespace quota is a number that represents how many unique names can be used. This number cannot be greater than LONG.MAX_VALUE in Java.

Command Syntax	`ozone sh bucket setquota [-hV] [--namespace-quota=<quotaInNamespace>][--space-quota=<quotaInBytes>] <uri>`

Command Usage	• Update bucket with only space quota:
	`ozone sh bucket setquota --space-quota=<quotaInBytes> <uri>`
	• Update bucket with only namespace quota:
	`ozone sh bucket setquota --namespace-quota=<quotaInNamespace> <uri>`
	• Update bucket with both space and namespace quota:
	`ozone sh bucket setquota --namespace-quota=<quotaInNamespace> --space-quota=<quotaInBytes> <uri>`

Purpose	• Manage the namespace and space quota of a bucket.

Arguments	• uri: The name of the bucket to create in the <prefix>://<Service ID>/<volumename>/<bucketname> format. • --namespace-quota: Update the number of keys, files, and directories in a bucket. This is an optional parameter. • --space-quota: Update the maximum size the bucket can occupy in the cluster. This is an optional parameter. • **Note:** • You cannot set the space quota of volumes and buckets in decimals. For example, 1.5 TB. • Ensure that the minimum space quota is the defa ult block size * replication factor. If you set the value lesser than the default block size * replication factor, while writing the data (key put) operation, an operation error is displayed.
Example	• `ozone sh bucket create --namespace-quota=100 o3://ozone1/vol1/buck1` This command sets the namespace quota of buck1 to 100. • `ozone sh bucket setquota --space-quota=10GB o3://ozone1/vol1/buck1` This command sets *VOL1/BUCK1* space quota to 10 GB.

Clearing namespace and space quota for bucket

Command Syntax	`ozone sh bucket clrquota [-hV] [--namespace-quota] [--space-quota] <uri>`

Command Usage	• Clear bucket with only space quota:
	`ozone sh bucket clrquota --space-quota <uri>`
	• Clear bucket with only namespace quota:
	`ozone sh bucket clrquota --namespace-quota <uri>`
	• Clear bucket with both space and namespace quota:
	`ozone sh bucket clrquota --space-quota --namespace-quota <uri>`

Purpose	• Clear the namespace quota and space quota of the bucket.

Arguments	• --namespace-quota: Clear the namespace quota of a bucket. • --space-quota: Clear the space quota of a bucket.
Example	• `ozone sh bucket clrquota --namespace-quota o3://ozone1/vol1/buck1` This command clears the namespace quota of buck1. • `ozone sh bucket clrquota --space-quota o3://ozone1/vol1/buck1` This command clears the space quota of buck1.

Managing storage elements by using the command line interface

The Ozone shell is the primary command line interface for managing storage elements such as volumes, buckets, and keys.

Note: Ensure that the valid length for bucket or volume name is 3-63 characters.

For more information about the various Ozone command line tools and the Ozone shell, see Apache Ozone Command Line Interface documentation.

Commands for managing volumes

Depending on whether you are an administrator or an individual user, the Ozone shell commands enable you to create, delete, view, list, and update volumes. Before running these commands, you must have configured the Ozone Service ID for your cluster from the Configuration tab of the Ozone service on Cloudera Manager.

Creating a volume

Only an administrator can create a volume and assign it to a user. You must assign administrator privileges to users before they can create volumes. For more information, see Assigning administrator privileges to users.

Command Syntax	`ozone sh volume create --quota=<volumecapacity> --user=<username> URI`
Purpose	Creates a volume and assigns it to a user.
Arguments	• -q, quota: Specifies the maximum size the volume can occupy in the cluster. This is an optional parameter. • -u, user: The name of the user who can use the volume. The designated user can create buckets and keys inside the particular volume. This is a mandatory parameter. • URI: The name of the volume to create in the <prefix>://<Service ID>/<volumename> format.
Example	`ozone sh volume create --quota=2TB --user=usr1 o3://ozone1/vol1` This command creates a 2 TB volume named vol1 for user usr1. Here, ozone1 is the Ozone Service ID.

Deleting a volume

Command Syntax	`ozone sh volume delete URI`
Purpose	Deletes the specified volume which must be empty. To delete the volume that is not empty, you must use `-r` in the delete command.
Arguments	URI: The name of the volume to delete in the <prefix>://<Service ID>/<volumename> format.
Example	`ozone sh volume delete o3://ozone1/vol2` This command deletes the empty volume vol2. Here, ozone1 is the Ozone Service ID.

Viewing volume information

Command Syntax	`ozone sh volume info URI`
Purpose	Provides information about the specified volume.
Arguments	URI: The name of the volume whose want to view in the <prefix>://<Service ID>/<volumename> format.
Example	`ozone sh volume info o3://ozone1/vol3` This command provides information about the volume vol3. Here, ozone1 is the Ozone Service ID.

Listing volumes

Command Syntax	`ozone sh volume list --user <username> URI`
Purpose	Lists all the volumes owned by the specified user.
Arguments	• -u, user: The name of the user whose volumes you want to list. • URI: The Service ID of the cluster in the <prefix>://<Service ID>/ format.
Example	`ozone sh volume list --user usr2 o3://ozone1/` This command lists the volumes owned by user usr2. Here, ozone1 is the Ozone Service ID.

Updating a volume

Command Syntax	`ozone sh volume setquota --namespace-quota <namespacecapacity> --space-quota <volumecapacity> URI`
Purpose	Updates the quota of the specific volume.

Arguments	• --namespace-quota <namespacecapacity>: Specifies the maximum number of buckets this volume can have. • --space-quota <volumecapacity>: Specifies the maximum size the volume can occupy in the cluster. • URI: The name of the volume to update in the <prefix>://<Service ID>/<volumename> format.
Example	`ozone sh volume setquota --namespace-quota 1000 --space-quota 10GB /volume1` This command sets *VOLUME1* namespace quota to 1000 and space quota to 10 GB.

Setting volume Space level quota

Command Syntax	`ozone sh volume setquota --space-quota <volumecapacity> /volume`
Purpose	Manage the quota of the specific volume.

Arguments	• --space-quota <volumecapacity>: Specifies the maximum size the volume can occupy in the cluster.
	• **Note:**
	  • You cannot set the quota of volumes and buckets in decimals. For example, 1.5 TB.
	  • Ensure that the minimum storage quota is default block size * replication factor. If you set the value lesser than the default block size * replication factor, while writing the data (key put) operation, an operation error is displayed.

Example	`ozone sh volume setquota --space-quota 10GB /volume1` This command sets *VOLUME1* space quota to 10 GB.

Clearing volume Space level quota

Command Syntax	`ozone sh volume clrquota --space-quota /volume`
Purpose	Clear the space quota of the volume.
Arguments	--space-quota: Specifies the space quota of a Volume.
Example	`ozone sh volume clrquota --space-quota /volume1` This command clears the space quota of Volume1.

Check space level quota for volume

Command Syntax	`ozone sh volume info /volume`
Purpose	Get the quota value and usedBytes information of the volume.

Arguments	-
Example	`ozone sh volume info /volume1` This command gets the quota value and usedBytes information of volume1.

Setting Namespace quota

Namespace quota is a number that represents how many unique names can be used. This number cannot be greater than LONG.MAX_VALUE in Java.

Command Syntax	`ozone sh volume create --namespace-quota <volumecapacity> /volume`
Purpose	Manage the quota of the namespace volume.
Arguments	--namespace-quota <volumecapacity>: Specifies the namespace quota of a Volume.
Example	`ozone sh volume create --namespace-quota 100 /volume1` This command sets the namespace quota of Volume1 to 100.

Clearing volume spacelvel quota

Command Syntax	`ozone sh volume clrquota --namespace-quota /volume`
Purpose	Clear the namespace quota of the volume.
Arguments	--namespace-quota: Specifies the namespace quota of a Volume.
Example	`ozone sh volume clrquota --namespace-quota /volume1` This command clears the namespace quota of Volume1.

Check Namespace level quota for volume

Command Syntax	`ozone sh volume info /volume`
Purpose	Get the quota value and usedNamespace information of the volume.
Arguments	-
Example	`ozone sh volume info /volume1` This command gets the quota value and usedNamespace information of volume1.

Assigning administrator privileges to users

You must assign administrator privileges to users before they can create Ozone volumes. You can use Cloudera Manager to assign the administrative privileges.

About this task

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Click the Configuration tab.
5. Search for the Ozone Service Advanced Configuration Snippet (Safety Valve) for ozone-conf/ozone-site.xml property.

   Specify values for the selected properties as follows:

   - Name: Enter ozone.administrators.
   - Value: Enter the ID of the user that you want as an administrator. In case of multiple users, specify a comma-separated list of users.
   - Description: Specify a description for the property. This is an optional value.
6. Enter a Reason for Change, and then click Save Changes to commit the change.

Commands for managing buckets

The Ozone shell commands enable you to create, delete, view, and list buckets. Before running these commands, you must have configured the Ozone Service ID for your cluster from the Configuration tab of the Ozone service on Cloudera Manager.



Note: Ensure that the valid length for bucket or volume name is 3-63 characters.

Creating a bucket

Command Syntax	`ozone sh bucket create URI`
Purpose	Creates a bucket in the specified volume.
Arguments	URI: The name of the bucket to create in the <prefix>://<Service ID>/<volumename>/<bucketname> format.
Example	`ozone sh bucket create o3://ozone1/vol1/buck1` This command creates a bucket buck1 in the volume vol1. Here, ozone1 is the Ozone Service ID.

Deleting a bucket

Command Syntax	`ozone sh bucket delete URI`
Purpose	Deletes the specified bucket which must be empty. To delete the bucket that is not empty, you must use `-r` in the delete command.

Arguments	URI: The name of the bucket to delete in the <prefix>://<Service ID>/<volumename>/<bucketname> format.
Example	`ozone sh bucket create o3://ozone1/vol1/buck2` This command deletes the empty bucket buck2. Here, ozone1 is the Ozone Service ID.

Viewing bucket information

Command Syntax	`ozone sh bucket info URI`
Purpose	Provides information about the specified bucket.
Arguments	URI: The name of the bucket whose details you want to view, in the <prefix>://<Service ID>/<volumename>/<bucketname> format.
Example	`ozone sh bucket info o3://ozone1/vol1/buck3` This command provides information about bucket buck3. Here, ozone1 is the Ozone Service ID.

Listing buckets

Command Syntax	`ozone sh bucket list URI --length=<number_of_buckets> --prefix=<bucket_prefix> --start=<starting_bucket>`
Purpose	Lists all the buckets in a specified volume.
Arguments	• -l, length: Specifies the maximum number of results to return. The default is 100. • -p, prefix: Lists bucket names that match the specified prefix. • -s, start: Returns results starting with the bucket *after* the specified value. • URI: The name of the volume whose buckets you want to list, in the <prefix>://<Service ID>/<volumename>/ format.
Example	`ozone sh bucket list o3://ozone1/vol2 --length=100 --prefix=buck --start=buck` This command lists 100 buckets belonging to volume vol2 and names starting with the prefix buck. Here, ozone1 is the Ozone Service ID.

Setting bucket Space level quota

Command Syntax	`ozone sh bucket setquota  --space-quota <volumecapacity> /volume/bucket`
Purpose	Manage the quota of the specific bucket under a bucket.

Arguments	• --space-quota <volumecapacity>: Specifies the maximum size the volume can occupy in the cluster.
	• **Note:**
	  • You cannot set the quota of volumes and buckets in decimals. For example, 1.5 TB.
	  • Ensure that the minimum storage quota is default block size * replication factor. If you set the value lesser than the default block size * replication factor, while writing the data (key put) operation, an operation error is displayed.

Example	`ozone sh bucket setquota --space-quota 10GB /volume1/bucket1` This command sets *VOLUME1/BUCKET1* space quota to 10GB.

Clearing bucket Space level quota

Command Syntax	`ozone sh bucket clrquota --space-quota /volume/bucket`
Purpose	Clear the space quota of the bucket.
Arguments	--space-quota: Specifies the space quota of a bucket.
Example	`ozone sh bucket clrquota --space-quota /volume1/bucket1` This command clears the space quota of bucket1.

Check space level quota for bucket

Command Syntax	`ozone sh bucket info /volume/bucket`
Purpose	Get the quota value and usedBytes information of the bucket.
Arguments	-
Example	`ozone sh bucket info /volume1/bucket1` This command gets the quota value and usedBytes information of bucket1.

Setting Namespace quota

Namespace quota is a number that represents how many unique names can be used. This number cannot be greater than LONG.MAX_VALUE in Java.

Command Syntax	`ozone sh bucket create --namespace-quota <volumecapacity> /volume/bucket`
Purpose	Manage the quota of the namespace bucket.

Arguments	--namespace-quota <volumecapacity>: Specifies the namespace quota of a bucket.
Example	`ozone sh bucket create --namespace-quota 100 /volume1/bucket1` This command sets the namespace quota of bucket1 to 100.

Clearing bucket spacelvel quota

Command Syntax	`ozone sh bucket clrquota --namespace-quota /volume/bucket`
Purpose	Clear the namespace quota of the bucket.
Arguments	--namespace-quota: Specifies the namespace quota of a bucket.
Example	`ozone sh bucket clrquota --namespace-quota /volume1/bucket1` This command clears the namespace quota of bucket1.

Check Namespace level quota for bucket

Command Syntax	`ozone sh bucket info /bucket`
Purpose	Get the quota value and usedNamespace information of the bucket.
Arguments	-
Example	`ozone sh bucket info /bucket1` This command gets the quota value and usedNamespace information of bucket1.

Commands for managing keys

The Ozone shell commands enable you to upload, download, view, delete, and list keys. Before running these commands, you must have configured the Ozone Service ID for your cluster from the Configuration tab of the Ozone service on Cloudera Manager.

Downloading a key from a bucket

Command Syntax	`ozone sh key get URI <local_file name>`
Purpose	Downloads the specified key from a bucket in the Ozone cluster to the local file system.

Arguments	• URI: The name of the key to download in the <prefix>://<Service ID>/<volumename>/<bucketname>/<keyname> format. • filename: The name of the file to which you want to write the key.
Example	`ozone sh key get o3://ozone1/hive/jun/sales.orc sales_jun.orc` This command downloads the sales.orc file from the /hive/jun bucket and writes to the sales_jun.orc file present in the local file system. Here, ozone1 is the Ozone Service ID.

Uploading a key to a bucket

Command Syntax	`ozone sh key put URI <filename>`
Purpose	Uploads a file from the local file system to the specified bucket in the Ozone cluster.
Arguments	• URI: The name of the key to upload in the <prefix>://<Service ID>/<volumename>/<bucketname>/<keyname> format. • filename: The name of the local file that you want to upload. • -r, --replication: The number of copies of the file that you want to upload.
Example	`ozone sh key put o3://ozone1/hive/year/sales.orc sales_corrected.orc` This command adds the sales_corrected.orc file from the local file system as key to /hive/year/sales.orc on the Ozone cluster. Here, ozone1 is the Ozone Service ID.

Deleting a key

Command Syntax	`ozone sh key delete URI`
Purpose	Deletes the specified key from the Ozone cluster.
Arguments	URI: The name of the key to delete in the <prefix>://<Service ID>/<volumename>/<bucketname>/<keyname> format.
Example	`ozone sh key delete o3://ozone1/hive/jun/sales_duplicate.orc` This command deletes the sales_duplicate.orc key. Here, ozone1 is the Ozone Service ID.

Viewing key information

Command Syntax	`ozone sh key info URI`
Purpose	Provides information about the specified key.

Arguments	URI: The name of the key you want to view in the <prefix>://<Service ID>/<volumename>/<bucketname>/<keyname> format.
Example	`ozone sh key info o3://ozone1/hive/jun/sales_jun.orc` This command provides information about the sales_jun.orc key. Here, ozone1 is the Ozone Service ID.

Listing keys

Command Syntax	`ozone sh key list URI --length=<number_of_keys> --prefix=<key_prefix> --start=<starting_key>`
Purpose	Lists the keys in a specified bucket.
Arguments	• -l, length: Specifies the maximum number of results to return. The default is 100. • -p, prefix: Returns keys that match the specified prefix. • -s, start: Returns results starting with the key *after* the specified value. • URI: The name of the bucket you want to list in the <prefix>://<Service ID>/<volumename>/<bucketname>/ format.
Example	`ozone sh key list o3://ozone1/hive/year/ --length=100 --prefix=<key_prefix> --start=day1` This command lists 100 keys belonging to the volume /hive/year/ and names starting with the prefix day, but listed after the value day1. Here, ozone1 is the Ozone Service ID.

Using Ozone S3 Gateway to work with storage elements

Ozone provides S3 Gateway, a REST interface that is compatible with the Amazon S3 API. You can use S3 Gateway to work with the Ozone storage elements.

In addition, you can use the Amazon Web Services CLI to use S3 Gateway.

After starting Ozone S3 Gateway, you can access it from the following link:

```
http://localhost:9878
```

Note: For the users or client applications that use S3 Gateway to access Ozone buckets on a secure cluster, Ozone provides the AWS access key ID and AWS secret key. See the Ozone security documentation for more information.

Configuration to expose buckets under non-default volumes

Ozone S3 Gateway allows access to all the buckets under the default /s3v volume. To access the buckets under a non-default volume, you must create a symbolic link to that bucket.

Consider a non-default volume /vol1 that has a bucket /bucket1 in the following example:

```
ozone sh volume create /s3v
```

```
ozone sh volume create /vol1

ozone sh bucket create /vol1/bucket1
ozone sh bucket link /vol1/bucket1 /s3v/common-bucket
```

As shown in the example, you can expose /bucket1 as an s3-compatible /common-bucket bucket through the Ozone S3 Gateway.

REST endpoints supported on Ozone S3 Gateway

In addition to the GET service operation, Ozone S3 Gateway supports various bucket and object operations that the Amazon S3 API provides.

The following table lists the supported Amazon S3 operations:

Operations on S3 Gateway

- GET service

Bucket operations

- GET Bucket (List Objects) Version 2
- HEAD Bucket
- DELETE Bucket
- PUT Bucket
- Delete multiple objects (POST)

Object operations

- PUT Object
- COPY Object
- GET Object
- DELETE Object
- HEAD Object
- Multipart Upload

Configuring Ozone to work as a pure object store

Depending on your requirement, you can configure Ozone to use the Amazon S3 APIs and perform the various volume and bucket operations.

About this task

You must modify the ozone.om.enable.filesystem.paths property in ozone-site.xml by using Cloudera Manager to configure Ozone as an object store.

Procedure

1. Open the Cloudera Admin Console.
2. Go to the Ozone service.
3. Click the Configuration tab.
4. Select Category > Advanced.
5. Configure the Ozone Service Advanced Configuration Snippet (Safety Valve) for ozone-conf/ozone-site.xml property as specified.
   - Name: ozone.om.enable.filesystem.paths
   - Value: False

6. Enter a Reason for Change and then click Save Changes.
7. Restart the Ozone service.

What to do next

You must also configure the client applications accessing Ozone to reflect the Ozone configuration changes. If you have applications in Spark, Hive or other services interacting with Ozone through the S3A interface, then you must make specific configuration changes in the applications.

Access Ozone S3 Gateway using the S3A filesystem

If you want to run Ozone S3 Gateway from the S3A filesystem, you must import the required CA certificate into the default Java truststore location on all the client nodes for running shell commands or jobs. This is a prerequisite when the S3 Gateway is configured with TLS.

About this task

S3A relies on the hadoop-aws connector, which uses the built-in Java truststore ($JAVA_HOME/jre/lib/security/cacerts). To override this truststore, you must create another truststore named jssecacerts in the same folder as cacerts on all the cluster nodes. When using Ozone S3 Gateway, you can import the CA certificate used to set up TLS into cacerts or jssecacerts on all the client nodes for running shell commands or jobs. Importing the certificate is important because the CA certificate used to set up TLS is not available in the default Java truststore, while the hadoop-aws connector library trusts only those certificates that are present in the built-in Java truststore.

Note: S3A is not supported when the File System Optimization (FSO) ozone.om.enable.filesystem.paths is enabled for Ozone Managers. Note that FSO is enabled by default. Therefore, to use S3A, you must override or set the ozone.om.enable.filesystem.paths property to false in the Cloudera Manager Clusters Ozone service Configuration Ozone Service Advanced Configuration Snippet (Safety Valve) for ozone-conf/ozone-site.xml property. After you save the configuration, restart all Ozone Managers for the configuration to take affect.

Important: Cloudera recommends that you use ofs:// to denote the Ozone storage path instead of *s3a://* wherever applicable. For example, use ofs://ozone1/vol1/bucket1/dir1/key1 instead of s3a://bucket1/dir1/key1.

Procedure

- Create a truststore named jssecacerts at $JAVA_HOME/jre/lib/security/ on all the cluster nodes configured for S3 Gateway, as specified.
  a) Run keytool to view the associated CA certificate and determine the srcalias from the output of the command.

```
/usr/java/default/bin/keytool -list -v -keys
tore [***SSL.CLIENT.TRUSTSTORE.LOCATION***]
```

  b) Import the CA certificate to all the hosts configured for S3 Gateway.

```
/usr/java/default/bin/keytool -importkeystore -destk
eystore $JAVA_HOME/jre/lib/security/jssecacerts -srckeys
tore [***SSL.CLIENT.TRUSTSTORE.LOCATION***] -srcalias [***ALIAS***]
```

Accessing Ozone S3 using S3A FileSystem

If the Ozone S3 gateway is configured with TLS (HTTPs), you must import the CA certificate to Java truststore. This is because the CA certificate that is used to set up TLS is not available in the default Java truststore; however, the hadoop-aws connector library only trusts the built-in Java truststore certificates.

To override the default Java truststore, create a truststore named jssecacerts in the same directory ($JAVA_HOME/lib/security/jssecacerts) on all cluster nodes where the user intends to run jobs or shell commands against Ozone S3. You can find the Ozone S3 gateway truststore location from the ozone-site.xml file which is normally located in the /

etc/ozone/conf.cloudera.OZONE-1 directory. From the ozone-site.xml file, you can find ssl.client.truststore.location and ssl.client.truststore.password.

List entries in the store

- `/usr/java/default/bin/keytool -list -v -keystore <<ssl.client.truststore.location>>`

```
[root@vc0833 ~]# /usr/java/default/bin/keytool -list -v -keystore /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks
Enter keystore password:
Keystore type: jks
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: cmrootca-0
Creation date: Feb 22, 2022
Entry type: trustedCertEntry
```

From the command output, you can find out the srcalias value which is shown as Alias name. In this example, the Alias name is cmrootca-0. Import the CA certificate (in this example, the certificate is imported to jssecacerts truststore). `/usr/java/default/bin/keytool -importkeystore -destkeystore $JAVA_HOME/lib/security/jssecacerts -srckeystore <<ssl.client.truststore.location>> -srcalias <<alias>>`

```
[root@vc0802 ~]# /usr/java/default/bin/keytool -importkeystore -destkeystore ./jssecacerts -srckeystore /var/lib/
cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks -srcalias cmrootca-0
Importing keystore /var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_truststore.jks to ./jssecacerts...
Enter destination keystore password:
Re-enter new password:
Enter source keystore password:
```

```
[root@ve1331 security]# pwd
/usr/lib/jvm/java-openjdk-11/lib/security
[root@ve1331 security]# ls -al
total 436
drwxrwxr-x 2 500 500     98 Apr  5  2019 .
drwxrwxr-x 6 500 500   4096 Apr  5  2019 ..
-rw-rw-r-- 1 500 500   1253 Apr  5  2019 blacklisted.certs
-rw-rw-r-- 1 500 500 184401 Dec  7 08:52 cacerts
-rw-rw-r-- 1 500 500   8815 Apr  5  2019 default.policy
-rw-rw-r-- 1 500 500 234959 Apr  5  2019 public_suffix_list.dat
[root@ve1331 security]#
```

Note: Depending on the installed JAVA version on your cluster, the jssecacerts truststore directory path might be different from what the command line and screenshot show.

- Enter the destination password as changeit and the source password as it is configured in the cluster.
- Obtain awsAccessKey and awsSecret using the ozone s3 getsecret command `ozone s3 getsecret --om-service-id=<<ozone service id>>`
- Ozone S3 properties need to be either passed in from command line or stored as cluster-wide Safety Valve in core-site.xml file. To do this is, add the Safety Valve to core-site.xml through HDFS configuration from Cloudera Manager.

```
fs.s3a.impl org.apache.hadoop.fs.s3a.S3AFileSystem
fs.s3a.access.key  <<accessKey>>
fs.s3a.secret.key  <<secret>>
fs.s3a.endpoint    <<Ozone S3 endpoint Url>>
fs.s3a.bucket.probe 0
fs.s3a.change.detection.version.required false
```

```
fs.s3a.change.detection.mode none
fs.s3a.path.style.access true
fs.s3a.directory.marker.retention keep
```

In the configurations, replace <<accessKey>> and <<secret>> with awsAccessKey and awsSecret obtained using the Ozone S3 getsecret command accordingly and <<Ozone S3 endpoint URL>> with Ozone S3 gateway URL from the cluster.

If you do not store the Ozone S3 properties as cluster-wide Safety Valve in core-site.xml file, you can run the command:

Create a directory dir1/dir2 in testbucket `hadoop fs -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.key=<<accesskey>> -Dfs.s3a.secret.key=<<secret>> -Dfs.s3a.endpoint=<<s3 endpoint url>> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem -mkdir -p s3a://testbucket/dir1/dir2`

S3 properties are stored as safety valves in the HDFS core-site.xml file in the following sample shell commands:

- Create a directory dir1/dir2 in testbucket `hadoop fs -mkdir -p s3a://testbucket/dir1/dir2`

```
[root@vc0802 ~]# hadoop fs -mkdir -p s3a://testbucket/dir1/dir2
22/07/19 10:27:12 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties
,hadoop-metrics2.properties
22/07/19 10:27:12 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
22/07/19 10:27:12 INFO impl.MetricsSystemImpl: s3a-file-system metrics system started
22/07/19 10:27:13 INFO Configuration.deprecation: No unit for fs.s3a.connection.request.timeout(0) assuming SECONDS
22/07/19 10:27:14 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics system...
22/07/19 10:27:14 INFO impl.MetricsSystemImpl: s3a-file-system metrics system stopped.
22/07/19 10:27:14 INFO impl.MetricsSystemImpl: s3a-file-system metrics system shutdown complete.
[root@vc0802 ~]# hadoop fs -ls -R s3a://testbucket/dir1/
22/07/19 10:27:29 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties
,hadoop-metrics2.properties
22/07/19 10:27:29 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
22/07/19 10:27:29 INFO impl.MetricsSystemImpl: s3a-file-system metrics system started
22/07/19 10:27:30 INFO Configuration.deprecation: No unit for fs.s3a.connection.request.timeout(0) assuming SECONDS
drwxrwxrwx   - systest systest          0 2022-07-19 10:27 s3a://testbucket/dir1/dir2
22/07/19 10:27:31 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics system...
22/07/19 10:27:31 INFO impl.MetricsSystemImpl: s3a-file-system metrics system stopped.
22/07/19 10:27:31 INFO impl.MetricsSystemImpl: s3a-file-system metrics system shutdown complete.
```

- Place a file named key1 in the dir1/dir2 directory in testbucket `hadoop fs -put /tmp/key1 s3a://testbucket/dir1/dir2/key1`

```
[root@vc0802 ~]# hadoop fs -put /tmp/key1 s3a://testbucket/dir1/dir2/key1
22/07/19 15:38:53 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties
,hadoop-metrics2.properties
22/07/19 15:38:53 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
22/07/19 15:38:53 INFO impl.MetricsSystemImpl: s3a-file-system metrics system started
22/07/19 15:38:54 INFO Configuration.deprecation: No unit for fs.s3a.connection.request.timeout(0) assuming SECONDS
22/07/19 15:38:58 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics system...
22/07/19 15:38:58 INFO impl.MetricsSystemImpl: s3a-file-system metrics system stopped.
22/07/19 15:38:58 INFO impl.MetricsSystemImpl: s3a-file-system metrics system shutdown complete.
[root@vc0802 ~]# hadoop fs -ls -R s3a://testbucket/dir1/
22/07/19 15:39:02 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties
,hadoop-metrics2.properties
22/07/19 15:39:02 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
22/07/19 15:39:02 INFO impl.MetricsSystemImpl: s3a-file-system metrics system started
22/07/19 15:39:03 INFO Configuration.deprecation: No unit for fs.s3a.connection.request.timeout(0) assuming SECONDS
drwxrwxrwx   - systest systest          0 2022-07-19 15:39 s3a://testbucket/dir1/dir2
-rw-rw-rw-   1 systest systest       8557 2022-07-19 15:38 s3a://testbucket/dir1/dir2/key1
22/07/19 15:39:04 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics system...
22/07/19 15:39:04 INFO impl.MetricsSystemImpl: s3a-file-system metrics system stopped.
22/07/19 15:39:04 INFO impl.MetricsSystemImpl: s3a-file-system metrics system shutdown complete.
```

- List files/directories under testbucket

```
[root@vc0802 ~]# hadoop fs -ls -R s3a://testbucket/
22/07/19 15:56:29 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-s3a-file-system.properties,
hadoop-metrics2.properties
22/07/19 15:56:29 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
22/07/19 15:56:29 INFO impl.MetricsSystemImpl: s3a-file-system metrics system started
22/07/19 15:56:29 INFO Configuration.deprecation: No unit for fs.s3a.connection.request.timeout(0) assuming SECONDS
drwxrwxrwx   - systest systest          0 2022-07-19 15:56 s3a://testbucket/dir1
drwxrwxrwx   - systest systest          0 2022-07-19 15:56 s3a://testbucket/dir1/dir2
-rw-rw-rw-   1 systest systest       8557 2022-07-19 15:38 s3a://testbucket/dir1/dir2/key1
22/07/19 15:56:30 INFO impl.MetricsSystemImpl: Stopping s3a-file-system metrics system...
22/07/19 15:56:30 INFO impl.MetricsSystemImpl: s3a-file-system metrics system stopped.
22/07/19 15:56:30 INFO impl.MetricsSystemImpl: s3a-file-system metrics system shutdown complete.
```

Examples of using the S3A filesystem with Ozone S3 Gateway

You can use the S3A filesystem with Ozone S3 Gateway to perform different Ozone operations.

The following examples show how you can use the S3A filesystem with Ozone.

Note: In the following examples, replace the values of access key and secret from the output of the ozone s3 getsecret --om-service-id=<*OZONE SERVICE ID*> command and replace the Ozone S3 endpoint URL with the S3 Gateway URL of the Ozone cluster.

Creating a directory in a bucket

The following example shows how you can create a directory /dir1/dir2 within a bucket named testbucket `hadoop fs -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.key=<accesskey> -Dfs.s3a.secret.key=<secret> -Dfs.s3a.endpoint=<s3 endpoint url> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem -mkdir -p s3a://testbucket/dir1/dir2`

Adding a key to a directory

The following example shows how you can add a key to the dir2 directory created in the previous example `hadoop fs -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.key=<accesskey> -Dfs.s3a.secret.key=<secret> -Dfs.s3a.endpoint=<s3 endpoint url> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem -put /tmp/key1 s3a://testbucket/dir1/dir2/key1`

Listing files or directories in a bucket

The following example shows how you can list the contents of a bucket named testbucket `hadoop fs -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version.required=false -Dfs.s3a.change.detection.mode=none Dfs.s3a.access.key=<accesskey> -Dfs.s3a.secret.key=<secret> -Dfs.s3a.endpoint=<s3 endpoint url> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem -ls -R s3a://testbucket/`

Access Ozone S3 Gateway from Cloudera components using S3A filesystem

Learn about how to access Ozone S3 Gateway from Cloudera components using S3A filesystem.

1. To access Ozone from Hive using S3A, see Configuring Hive access for Ozone using S3A.
2. To access Ozone from Impala using S3A, see Configuring Impala access for Ozone using S3A.
3. To access Ozone from Spark using S3A, see Configuring Spark access for Ozone using S3A.

Using the AWS CLI with Ozone S3 Gateway

You can use the Amazon Web Services (AWS) command line interface (CLI) to interact with S3 Gateway and work with various Ozone storage elements.

Configuring an https endpoint in Ozone S3 Gateway to work with AWS CLI

For Ozone S3 Gateway to work with Amazon Web Services (AWS) command line interface (CLI), you must perform specific configurations, especially if the S3 Gateway has https endpoints.

About this task

You must export the CA certificate required on all the client nodes for running the shell commands, and convert the certificate to PEM format because the Python SSL supported with AWS CLI honors certificates in the PEM format.

Procedure

1. Run keytool to view the associated CA certificate and determine the srcalias from the output of the command.

```
/usr/java/default/bin/keytool -list -v -keys
tore <SSL.CLIENT.TRUSTSTORE.LOCATION>
```

2. Export the CA certificate to PEM format.

```
keytool -export -alias <ALIAS> -file <S3G-CA.CRT> -keysto
re <SSL.CLIENT.TRUSTSTORE.LOCATION>
```

```
openssl x509 -inform DER -outform PEM -in <S3G-CA.CRT> -out /tmp/s3gca.pem
```

3. Run the ozone s3 getsecret command for the values of the access key and secret key.

```
ozone s3 getsecret --om-service-id=<OZONE SERVICE ID>
```

4. Run the aws configure command to configure the access key and the secret key.

```
aws configure accesskey/secret
```

What to do next

You can pass the certificate in PEM file format to the aws s3api command and perform various tasks, such as managing buckets, keys, and so on. The following example shows how you can create a bucket using the aws s3api command:

```
aws s3api --endpoint https://bv-hoz-1.bv-hoz.abc.site:9879 --ca-bundle "/tmp
/s3gca.pem" create-bucket --bucket=wordcount
```

Examples of using the AWS CLI for Ozone S3 Gateway

You can use the Amazon Web Services (AWS) command line interface (CLI) to interact with S3 Gateway and work with various Ozone storage elements.

Defining an alias for the S3 Gateway endpoint

Defining an alias for the S3 Gateway endpoint helps you in using a simplified form of the AWS CLI. The following examples show how you can define an alias for the S3 Gateway endpoint URL:

```
alias ozones3api='aws s3api --ca-bundle --endpoint https://localhost:9879'
alias ozones3api='aws s3api --ca-bundle --endpoint http://localhost:9878'
```

Examples of using the AWS CLI to work with the Ozone storage elements

The following examples show how you can use the AWS CLI to perform various operations on the Ozone storage elements. All the examples specify the alias ozones3api:

Operations	Examples
Creating a bucket	`ozones3api create-bucket --bucket buck1` This command creates a bucket buck1.
Adding objects to a bucket	`ozones3api put-object --bucket buck1 --key Doc1 --body ./Doc1.md` This command adds the key Doc1 containing data from Doc1.md to the bucket buck1.

Listing objects in a bucket	`ozones3api list-objects --bucket buck1`
	This command lists the objects in the bucket buck1. An example output of the command is as follows:
	`{ "Contents": [{ "LastModified": "2018-11-02T21:57:40.875Z", "ETag": "1541195860875", "StorageClass": "STANDARD", "Key": "Doc1", "Size": 2845 }, { "LastModified": "2018-11-02T22:36:23.358Z", "ETag": "1541198183358", "StorageClass": "STANDARD", "Key": "Doc2", "Size": 5615 }, { "LastModified": "2018-11-02T21:56:47.370Z", "ETag": "1541195807370", "StorageClass": "STANDARD", "Key": "Doc3", "Size": 1780 }] }`

Downloading an object from a bucket	`ozones3api get-object --bucket buck1 --key Doc1 ./Dpc1`
	This command downloads the key Doc1 from the bucket buck1 as a file Dpc1. An example output of the command is as follows:
	`{ "ContentType": "application/oc`

Accessing Ozone object store with Amazon Boto3 client

Boto3 is an AWS SDK for Python. It provides object-oriented API services and low-level services to the AWS services. It allows users to create, and manage AWS services such as EC2 and S3. Understand how to access the Ozone object store with Amazon Boto3 client.

Note: Cloudera supports S3 only.

Prerequisites

Ensure you have installed a higher version of Python 3 for your Boto3 client. For information on Boto3 documentation, see Boto3 documentation.

Obtaining resources to Ozone

You must obtain the required resources to create the required S3 resources.

For more information, see Amazon Boto3 documentation.

```
s3 = boto3.resource('s3',
                    endpoint_url='http://localhost:9878',
                    aws_access_key_id='testuser/scm@EXAMPLE.COM',
                    aws_secret_access_key='c261b6ecabf7d37d5f9ded654b1c724
adac9bd9f13e247a235e567e8296d2999'
)
'endpoint_url' is pointing to Ozone s3 endpoint.
```

Obtaining client to Ozone through session

You must obtain a client to Ozone through a session.

For more information, see Amazon Boto3 documentation.

```
Create a session
    session = boto3.session.Session()

Obtain s3 client to Ozone via session:

    s3_client = session.client(
        service_name='s3',
        aws_access_key_id='testuser/scm@EXAMPLE.COM',
        aws_secret_access_key='c261b6ecabf7d37d5f9ded654b1c724adac9bd9f13e2
47a235e567e8296d2999',
        endpoint_url='http://localhost:9878',
    )
    'endpoint_url' is pointing to Ozone s3 endpoint.
In our code sample below, we're demonstrating the usage of both s3 and s3_c
lient.
```

There are multiple ways to configure Boto3 client credentials if you're connecting to a secured cluster. In these cases, the above lines of passing 'aws_access_key_id' and 'aws_secret_access_key' when creating Ozone s3 client shall be skipped.

For more information, see Boto3 documentation.

List of APIs verified

Understand the list of APIs that were verified.

Following APIs were verified:

- Create bucket

- List bucket
- Head bucket
- Delete bucket
- Upload file
- Download file
- Delete objects(keys)
- Head object
- Multipart upload



Note: Ensure that the valid length for bucket or volume name is 3-63 characters.

Create a bucket

Use the following code snippet to create a bucket.

```
response = s3_client.create_bucket(Bucket='bucket1')
print(response)
```

This will create a bucket ‘bucket1’ in Ozone volume ‘s3v’.

List buckets

Use the following code snippet to list buckets.

```
response = s3_client.list_buckets()
print('Existing buckets:')
for bucket in response['Buckets']:
    print(f'  {bucket["Name"]}')
```

This will list all buckets in Ozone volume ‘s3v’.

Head a bucket

Use the following code snippet to head a bucket.

```
response = s3_client.head_bucket(Bucket='bucket1')
print(response)
```

This will head bucket ‘bucket1’ in Ozone volume ‘s3v’.

Delete a bucket

Use the following code snippet to delete a bucket.

```
response = s3_client.delete_bucket(Bucket='bucket1')
print(response)
```

This will delete the bucket ‘bucket1’ from Ozone volume ‘s3v’.

Upload a file

Use the following code snippet to upload a bucket.

```
response = s3.Bucket('bucket1').upload_file('./README.md','README.md')
print(response)
```

This will upload ‘README.md’ to Ozone creates a key ‘README.md’ in volume ‘s3v’.

Download a file

Use the following code snippet to download a file.

```
response = s3.Bucket('bucket1').download_file('README.md', 'download.md')
```

```
print(response)
```

This will download ‘README.md’ from Ozone volume ‘s3v’ to local and create a file with name ‘download.md’.

Head an object

Use the following code snippet to head an object.

```
response = s3_client.head_object(Bucket='bucket1', Key='README.md')
print(response)
```

This will head object ‘README.md’ from Ozone volume ‘s3v’ in the bucket ‘bucket1’.

Delete Objects

Use the following code snippet to delete objects.

```
response = s3_client.delete_objects(
    Bucket='bucket1',
    Delete={
        'Objects': [
            {
                'Key': 'README4.md',
            },
            {
                'Key': 'README3.md',
            },
        ],
        'Quiet': False,
    },
)
```

This will delete objects ‘README3.md’ and ‘README4.md’ from Ozone volume ‘s3v’ in bucket ‘bucket1’.

Multipart upload

Use the following code snippet to use ‘maven.gz’ and ‘maven1.gz’ as copy source from Ozone volume ‘s3v’ and to create a new object ‘key1’ in Ozone volume ‘s3v’.

```
response = s3_client.create_multipart_upload(Bucket='bucket1', Key='key1')
print(response)
uid=response['UploadId']
print(uid)

response = s3_client.upload_part_copy(
    Bucket='bucket1',
    CopySource='/bucket1/maven.gz',
    Key='key1',
    PartNumber=1,
    UploadId=str(uid)
)
print(response)
etag1=response.get('CopyPartResult').get('ETag')
print(etag1)

response = s3_client.upload_part_copy(
    Bucket='bucket1',
    CopySource='/bucket1/maven1.gz',
    Key='key1',
    PartNumber=2,
    UploadId=str(uid)
)
print(response)
etag2=response.get('CopyPartResult').get('ETag')
print(etag2)
```

```
response = s3_client.complete_multipart_upload(
    Bucket='bucket1',
    Key='key1',
    MultipartUpload={
        'Parts': [
            {
                'ETag': str(etag1),
                'PartNumber': 1,
            },
            {
                'ETag': str(etag2),
                'PartNumber': 2,
            },
        ],
    },
    UploadId=str(uid),
)
print(response)
```

Note: 'ETag's is required and important for the call.

Ozone HttpFS gateway

Ozone now supports HttpFS gateway. This allows Ozone to integrate with other tools through REST APIs. HttpFS service is a Jetty based web application that uses the Hadoop FileSystem API to talk to the cluster. HttpFS service is an optional role available in an Ozone cluster.

You can use HttpFS as a gateway to access data on an Ozone cluster behind a firewall. HttpFS is the only system that is allowed to cross the firewall into the Ozone cluster. You can use HttpFS to access data in Ozone using:

- HTTP utilities such as curl and wget
- HTTP libraries Perl from languages other than Java

HttpFS service provides REST HTTP gateway supporting File System operations such as read and write. HttpFS service is interoperable with the webhdfs REST HTTP API. You can use the webhdfs client FileSystem implementation to access HttpFS using:

- Ozone filesystem CLI (ozone fs)
- Java applications using the Hadoop FileSystem Java API.

HttpFS has built-in security that supports the following:

- Hadoop pseudo authentication
- Kerberos SPNEGO
- Other pluggable authentication mechanisms
- Hadoop proxy user support

Supported operations

Review the list of WebHDFS REST APIs and their state of support in Ozone.

File and directory operations

Operation	Support
Create and Write to a File	Supported

Operation	Support
Append to a File	Not implemented in Ozone
Concat Files	Not implemented in Ozone
Open and Read a File	Supported
Make a Directory	Supported
Create a Symbolic Link	Not implemented in Ozone
Rename a File or Directory	Supported with limitations
Delete a File or Directory	Supported
Truncate a File	Not implemented in Ozone
Status of a File or Directory	Supported
List a Directory	Supported
List a File	Supported
Iteratively List a Directory	Supported

Other file system operations

Operation	Support
Get Content Summary of a Directory	Supported
Get Quota Usage of a Directory	Supported
Set Quota	Not implemented in Ozone FileSystem API
Set Quota By Storage Type	Not implemented in Ozone
Get File Checksum	Unsupported
Get Home Directory	Unsupported
Get Trash Root	Unsupported
Set Permission	Not implemented in Ozone FileSystem API
Set Owner	Not implemented in Ozone FileSystem API
Set Replication Factor	Not implemented in Ozone FileSystem API
Set Access or Modification Time	Not implemented in Ozone FileSystem API
Modify ACL Entries	Not implemented in Ozone FileSystem API
Remove ACL Entries	Not implemented in Ozone FileSystem API
Remove Default ACL	Not implemented in Ozone FileSystem API
Remove ACL	Not implemented in Ozone FileSystem API

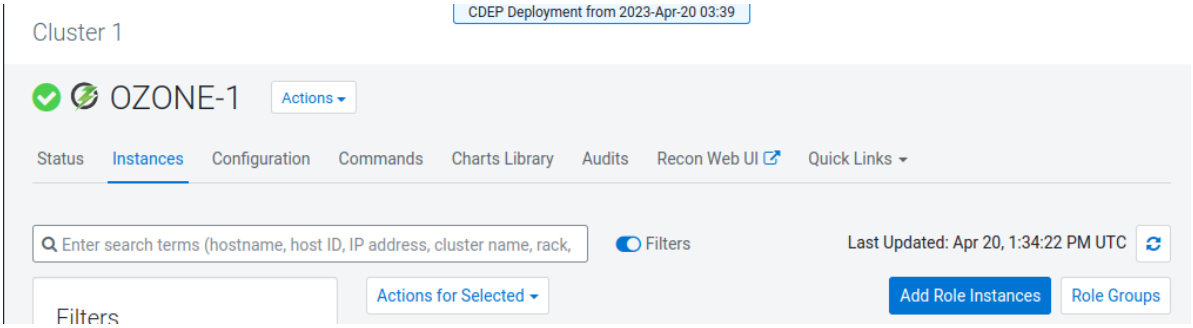
Operation	Support
Set ACL	Not implemented in Ozone FileSystem API
Get ACL Status	Not implemented in Ozone FileSystem API
Check access	Not implemented in Ozone FileSystem API

Adding Ozone HttpFS gateway

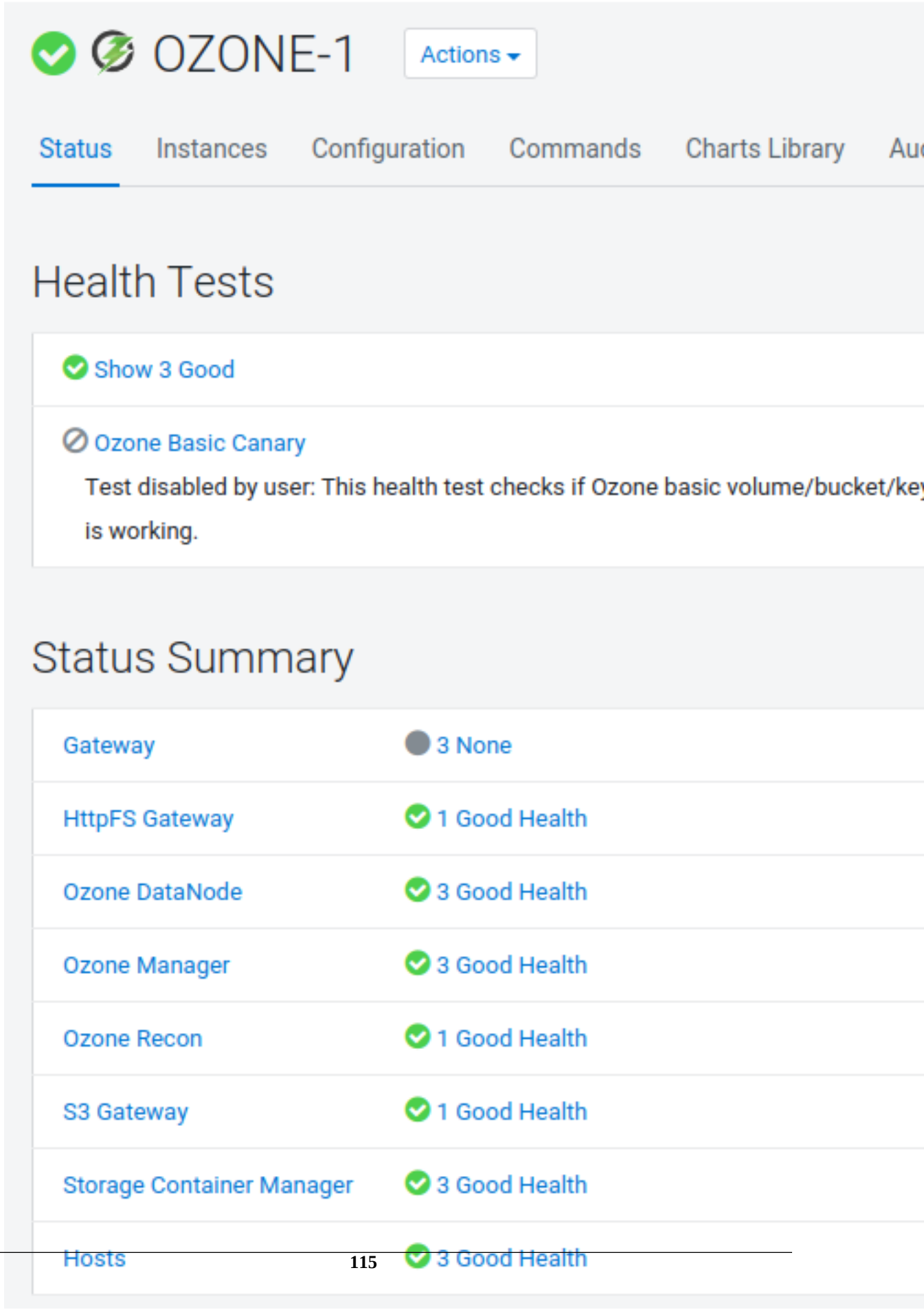
Ozone now supports HttpFS gateway. This allows Ozone to integrate with other tools through REST APIs. HttpFS service is a Jetty based web application that uses the Hadoop FileSystem API to talk to the cluster. HttpFS service is an optional role available in an Ozone cluster.

Procedure

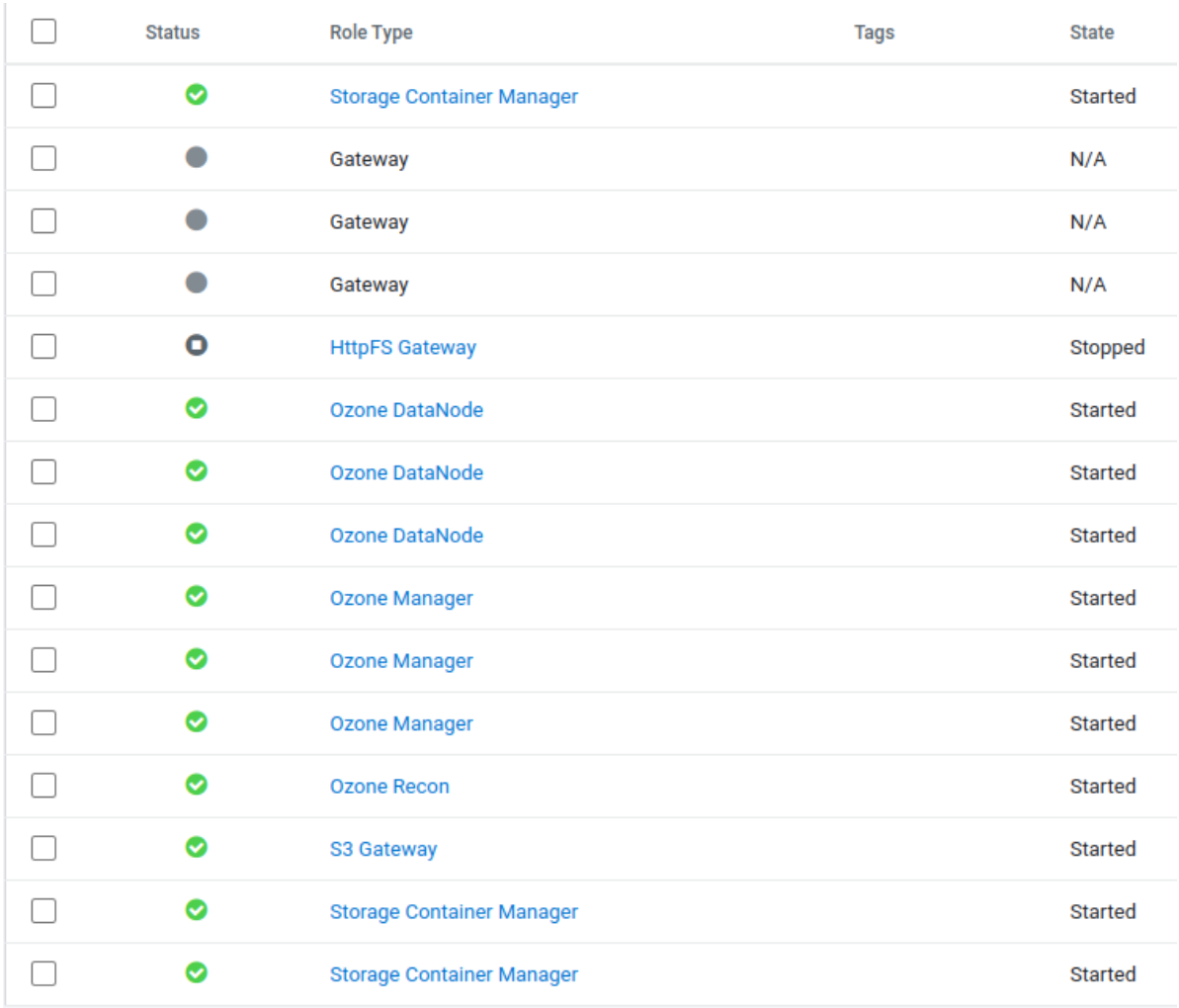
1. Restart the CDP Private Cloud Base cluster to ensure dependent components use the correct version of Ozone.
2. Assign the HttpFS Gateway role to one of the instances.
   a) Log in to Cloudera Manager
   b) Go to Clusters Ozone-1 Instances
   c) Click Add Role Instances



   d) Select one of the hosts for the HttpFS Gateway role and Save
   You should see the following roles:


OZONE-1
Actions
Status
Instances
Configuration
Commands
Charts Library
Health Tests
Show 3 Good
Ozone Basic Canary
Test disabled by user: This health test checks if Ozone basic volume/bucket/key
is working.
Status Summary
Gateway
3 None
HttpFS Gateway
1 Good Health
Ozone DataNode
3 Good Health
Ozone Manager
3 Good Health
Ozone Recon
1 Good Health
S3 Gateway
1 Good Health
Storage Container Manager
3 Good Health
Hosts
3 Good Health

e) Start the HttpFS Gateway role.



Status	Role Type	Tags	State
✓	Storage Container Manager		Started
●	Gateway		N/A
●	Gateway		N/A
●	Gateway		N/A
■	HttpFS Gateway		Stopped
✓	Ozone DataNode		Started
✓	Ozone DataNode		Started
✓	Ozone DataNode		Started
✓	Ozone Manager		Started
✓	Ozone Manager		Started
✓	Ozone Manager		Started
✓	Ozone Recon		Started
✓	S3 Gateway		Started
✓	Storage Container Manager		Started
✓	Storage Container Manager		Started

Configuring TLS/SSL encryption manually for Ozone HttpFS gateway

You can configure TLS/SSL encryption manually for Ozone HttpFS gateway using Cloudera Manager.

Before you begin

- The keystores containing certificates that are bound to the proper domain names are accessible on all the hosts on which at least one Ozone role instance is running.
- The hdfs user has read permissions to the keystore files for Ozone.
- You must specify absolute paths to the keystore file. These settings apply to all hosts on which the various Ozone role instances run. Therefore, the paths that you specify must be valid on all the hosts. In addition, the keystore file names for Ozone must be the same on all hosts.

Procedure

1. In Cloudera Manager, go to Ozone Configuration .
2. Search for *tls/ssl*.
3. Enter the TLS/SSL properties for the various Ozone roles.

   Provide the values for the properties corresponding to the following fields on the Ozone configuration page:

   - Enable TLS/SSL for HttpFS Gateway

- HttpFS Gateway TLS/SSL Server JKS Keystore File Location
- HttpFS Gateway TLS/SSL Server JKS Keystore File Password
- HttpFS Gateway TLS/SSL Server JKS Keystore Key Password
- **Note:** Ensure that you configure the ozone.prometheus.ca.file property when configuring TLS/SSL encryption for Ozone. If the location of the root CA certificate is not specified, the location of the auto-TLS CA certificate is considered as the default value for this property.

4. Click Save Changes and restart the Ozone service.

Configuring HttpFS proxy users

HttpFS proxy user functionality enables you to perform operations on behalf of another user. In Cloudera Base on premises, configure this user impersonation through Cloudera Manager.

Procedure

1. Sign in to Cloudera Manager.
2. In the left navigation, click Clusters.
3. Select the Ozone service.
4. Click Configuration.
5. Under Scope, click HttpFS Gateway.
6. In the search bar, type proxy user.
   The following HttpFS proxy user settings for Data Explorer and Hive are displayed.

Table 6: HttpFS proxy user configurations

Property	Description	Allowed values
httpfs.proxyuser.hue.hosts	Specifies the hosts from which Data Explorer can perform impersonation operations.	Comma-separated hostnames, IP addresses, or IP address ranges in CIDR format. Use * to allow any host.
httpfs.proxyuser.hue.groups	Specifies the groups whose members Data Explorer can impersonate.	Comma-separated group names. Use * to allow any group.
httpfs.proxyuser.hive.hosts	Specifies the hosts from which Hive can perform impersonation operations.	Comma-separated hostnames, IP addresses, or IP address ranges in CIDR format. Use * to allow any host.
httpfs.proxyuser.hive.groups	Specifies the groups whose members Hive can impersonate.	Comma-separated group names. Use * to allow any group.

7. Configure both the hosts and groups properties for the service account that performs impersonation.

   Important:

   - For production environments, Cloudera recommends that you specify authorized hosts and groups instead of using *.
   - Cloudera Manager generates httpfs-site.xml. Do not edit this file directly because Cloudera Manager replaces it when the role restarts.

8. Click Save Changes.
9. Restart the affected HttpFS Gateway role instances after you change these properties.

Results

HttpFS proxy user is configured and the user can perform operations on behalf of another user.

- The following error indicates that the target user is not allowed by the proxy user's group configuration:

```
User: user/host@REALM is not allowed to impersonate user01
```

To resolve this error:

1. Verify that the Kerberos principal maps to the expected short user name through hadoop.security.auth_to_local.
2. Verify that the corresponding httpfs.proxyuser.hue.groups or httpfs.proxyuser.hive.groups property includes a group to which the target user belongs.

- The following error indicates that the request originated from a host that is not allowed:

```
Unauthorized connection for super-user: user/host@REALM from IP 192.0.2.10
```

Verify that the corresponding httpfs.proxyuser.hue.hosts or httpfs.proxyuser.hive.hosts property includes the source hostname, IP address, or IP address range.

Working with Ozone File System (ofs)

The ofs file system is a flat layout file system that allows Ozone clients to access all the volumes and buckets under a single root. Client applications such as Hive, Spark, YARN, and MapReduce run natively on ofs without any modifications.

Setting up ofs

Select the Ozone bucket to configure ofs.

Procedure

1. Select the Ozone bucket on which you want ofs to reside.

   If you do not have a designated volume or bucket for ofs, create them using the required commands:

   ```
   ozone sh volume create /volume
   ozone sh bucket create /volume/bucket
   ```

   Note: Cloudera Manager currently does not support Ozone as the default file system.

2. The ofs file system implementation classpath is added to Cloudera services by default. But if an application is unable to instantiate the ofs file system class, add the ozone-filesystem-hadoop3.jar to the classpath.

   ```
   export HADOOP_CLASSPATH=/opt/ozone/share/ozonefs/lib/hadoop-ozone-filesystem-hadoop3-*.jar:$HADOOP_CLASSPATH
   ```

3. After setting up ofs, you can run HDFS dfs CLI commands such as the following on Ozone FS: (Assuming Ozone Service ID is “omservice1”)

   `hdfs dfs -ls ofs://omservice1/volume/bucket` and `hdfs dfs -mkdir ofs://omservice1/volume/bucket/users`

   Now, applications such as Hive and Spark can run on this file system after some basic configuration changes.

   Note: Any keys that are created or deleted in the bucket using methods other than ofs are displayed as directories and files in o3fs.

Related Information

Configuration options for Spark to work with Ozone File System (ofs)

Volume and bucket management using ofs

When using ofs, Ozone administrators and users can perform various volume and bucket operations with the help of the Hadoop shell commands such as creating volumes and buckets and using ACLs on the volumes and buckets.

Creating volumes and buckets

Ozone administrators can create directories under the root and first-level directories using the Hadoop shell. Creating a directory under the root is equivalent to creating an Ozone volume. Creating a directory under a first-level directory is equivalent to creating a bucket. In addition, Ozone users can create buckets under volumes to which they have the write access.

In the following example, you create a volume named volume1 using the `-mkdir` command of the Hadoop shell:

```
ozone fs -mkdir ofs://ozservice1/volume1/
```

The equivalent Ozone command to create a volume is as follows:

```
ozone sh volume create o3://ozservice1/volume1/
```

Similarly, the Hadoop shell command for creating a bucket is as follows:

```
ozone fs -mkdir ofs://ozservice1/volume1/bucket1/
```



Note:

- OFS does not support recursive volume delete operation. Recursive volume delete operation is supported using the Ozone shell command.
- If you use the `-mkdir -p` command to create volumes and buckets that do not exist, Ozone creates the specified volumes and buckets.

Using the /tmp directory

The ofs root contains a special tmp volume mount for backward compatibility with legacy Hadoop applications that use the /tmp/ directory. To use the volume mount, the Ozone administrator must first create a tmp volume and set its Access Control List (ACL) to ALL. This administrator needs to perform this process once for every cluster.

The following example shows how to create the tmp volume and assign it the required ACLs:

```
ozone sh volume create tmp
ozone sh volume setacl tmp -al world::a
```

After the administrator has created the tmp volume, each user must initialize their respective tmp bucket once. The following example shows how to initialize the tmp bucket.

```
ozone fs -mkdir ofs://ozservice1/tmp/
```

The user can then write to the /tmp/ bucket just as they would to a regular bucket.

Using ACLs on volumes and buckets

You must consider the following when setting Access Control Lists (ACLs) on Ozone volumes and buckets:

- Setting ACLs on a first-level directory except /tmp/ is the same as setting ACLs on a volume.
- Setting ACLs on a second-level directory is the same as setting ACLs on a bucket.
- The ACLs on the /tmp/ directory are the same as those on the bucket from which the /tmp/ directory is mapped.

  For example, if you map ofs:///tmp/ from ofs:///tmp/<tmp-bucket-for-current-user>/, the ACLs on ofs:///tmp/<tmp-bucket-for-current-user>/ are the same as those on ofs:///tmp/bucket1/.

Note: By default, the name of a user's temp bucket under the /tmp/ volume is the MD5 hash of the username.

- You cannot set ACLs on the root (/) because it is only a logical root.

Using _ in naming volume and bucket

If you prefer to use _ in naming the volumes and buckets, then you must configure the following parameter on the Ozone server:

<property> <name>ozone.om.namespace.s3.strict</name> <value>false</value> </property>

Warning: Using _ in naming the volumes and buckets will break S3 complaince.

Renaming volumes and buckets

The ofs file system does not support renaming of volumes and buckets. Any attempt to rename a volume or a bucket results in an exception. You can only rename directories inside a bucket.

For example, ofs supports renaming of ofs:///volume1/bucket1/dir1 to ofs:///volume1/bucket1/dir2.

Key management using ofs

When using ofs, Ozone administrators and users can perform various operations on Ozone keys with the help of the Hadoop shell commands such as creating keys, recursively listing keys, and renaming keys in a bucket.

Creating keys

You must consider the following when creating Ozone keys using ofs:

- You cannot create files (keys) under the root or the first-level directory (volume) except in the /tmp/ directory.
- You can add keys to the second-level directory (bucket) or lower-level directories.

Recursively listing keys

You must consider the following when using the ls -R command to recursively list Ozone keys under volumes and buckets:

Running the ls -R command	Recursively lists the following
For a bucket	All the keys that belong to the particular bucket
For a volume	All the buckets that belong to the specified volume and the keys that belong to each bucket
At the root	All the volumes under the root, all the buckets that belong to each volume, and all the keys that belong to each bucket

Renaming keys

You can rename only the keys that belong to a bucket. The ofs file system does not allow you to rename the keys across volumes or buckets.

For example, ofs allows renaming of the key ofs:///volume1/bucket1/key1.txt to ofs:///volume1/bucket1/key2.txt. However, ofs:///volume1/bucket1/key1.txt cannot be renamed to ofs:///volume1/bucket2/key11.txt.

Working with Ozone File System (o3fs)

The Ozone File System (o3fs) is a Hadoop-compatible file system. Applications such as Hive, Spark, YARN, and MapReduce run natively on o3fs without any modifications.

The Ozone File System resides on a bucket in the Ozone cluster. All the files created through o3fs are stored as keys in that bucket. Any keys created in the particular bucket without using the file system commands are shown as files or directories on o3fs.

Note: o3fs is deprecated. Cloudera recommends you to use ofs instead. For information, see Working with Ozone File System (ofs).

Setting up o3fs

Select the Ozone bucket to configure o3fs.

Procedure

1. Select the Ozone bucket on which you want o3fs to reside.

   If you do not have a designated volume or bucket for o3fs, create them using the commands

   ```
   ozone sh volume create /volume
   ozone sh bucket create /volume/bucket
   ```

   Note: Cloudera Manager currently does not support Ozone as the default file system.

2. The o3fs file system implementation classpath is added to Cloudera services by default. But if an application is unable to instantiate the o3fs file system class, add the ozone-filesystem-hadoop3.jar to the classpath.

   ```
   export HADOOP_CLASSPATH=/opt/ozone/share/ozonefs/lib/hadoop-ozone-filesy
   stem-hadoop3-*.jar:$HADOOP_CLASSPATH
   ```

3. After setting up o3fs, you can run HDFS commands such as the following on Ozone: (Assuming Ozone Service ID is ozone)

   `hdfs dfs -ls o3fs://bucket.volume.ozone/` and `hdfs dfs -mkdir o3fs://bucket.volume.ozone/users`

   Now, applications such as Hive and Spark can run on this file system after some basic configuration changes.

   Note:

   - Any keys that are created or deleted in the bucket using methods other than o3fs are displayed as directories and files in o3fs.
   - o3fs is deprecated. Cloudera recommends you to use ofs instead. For information, see Setting up ofs.

Ozone configuration options to work with Cloudera components

There are specific options that you must configure to ensure that other Cloudera components such as Spark and Hive work with Ozone.

In the case of Spark, you must update a specific configuration property to run Spark jobs with o3fs on a secure Kerberos-enabled cluster. Similarly, for Hive, you must configure the values of specific properties to store Hive managed tables on Ozone.

Configuration options for Spark to work with Ozone File System (ofs)

After setting up ofs, you can make configuration updates specific to components such as Spark to ensure that they work with Ozone.

To run Spark jobs with ofs on a secure Kerberos-enabled cluster, ensure that you assign a valid URI by setting the value of the Spark Client Advanced Configuration Snippet (Safety Valve) property for the spark.conf or the spark-defaults.conf file through the Cloudera Manager web UI.

For example:

```
spark.yarn.access.hadoopFileSystems=ofs://service1/vol1/bucket1/
```

Related Information

Setting up ofs

Configuration options to store Hive managed tables on Ozone

If you want to store Hive managed tables with ACID properties on Ozone, you must configure specific properties in hive-site.xml.

You can consider either of the following options to store Hive managed tables with ACID support on Ozone:

- Set the value of the hive.metastore.warehouse.dir property to point to the path of the Ozone directory where you want to store the Hive tables.
- Set the value of the metastore.warehouse.tenant.colocation property to true. You can then set the MANAGEDLOCATION of your Hive database to point to an Ozone directory so that the Hive tables can reside at the specified location.



Note: Dynamic partitioning in Hive with the default settings can generate an unexpected load on the filesystem when bulk loading data into tables because Hive creates a number of files for every partition. To avoid this issue, consider updating the following properties and tuning them further based on your requirements: hive.optimize.sort.dynamic.partition and hive.optimize.sort.dynamic.partition.threshold.

From a file system perspective, the recommended values are as follows:

- hive.optimize.sort.dynamic.partition = true
- hive.optimize.sort.dynamic.partition.threshold = 0

If you notice that some queries are taking a longer time to complete or failing entirely (usually noticed in large clusters), you can choose to revert the value of hive.optimize.sort.dynamic.partition.threshold to "-1". The performance issue is related to HIVE-26283.

Configuration options for Impala to work with Ozone File System

Learn how Ozone can work with Impala.

You can use Impala to query data files that reside on Apache Ozone distributed storage, rather than in HDFS.

The Ozone Erasure Coding (EC) provides data durability and fault-tolerance along with reduced storage space and ensures data durability similar to the Ratis THREE replication approach. Impala query engine supports Ozone EC.

Related Information

Impala with Ozone

Erasure Coding Overview

Configuration options for Oozie to work with Ozone storage

Oozie supports Ozone storage along with HDFS.

Apache Ozone is a highly scalable next-gen object store available on the Cloudera Base on premises cluster which enables you to optimize storage for big data workloads.

For Oozie to integrate with Ozone, you need to perform the following steps:

1. Upload Oozie ShareLib to Ozone.
2. Enable Oozie workflows that access Ozone storage.

For more details, see *Using Oozie with Ozone*.

Related Information

Using Oozie with Ozone

Configuration options for Cloudera components to work with Ozone S3

Learn about how to set up an external account for Ozone S3 access and configure S3 with the Cloudera components.

Setting up an external account for Ozone S3 access

Learn about how to set up an external account for Ozone S3 access.

Before you begin

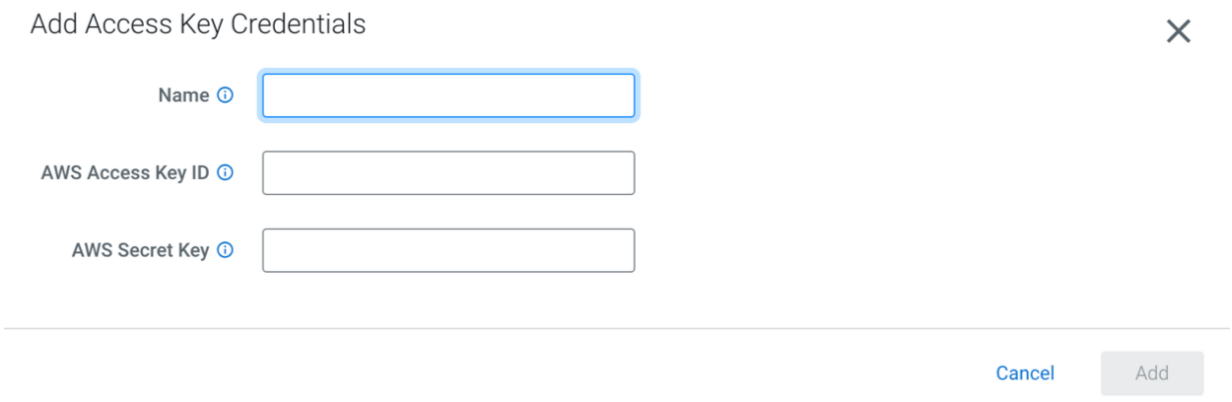
You must have S3 credentials for Hive.

Procedure

1. Run the following commands on CLI to get the S3 credentials for Hive:

```
kinit hive
ozone s3 getsecret --om-service-id=[***OZONE-SERVICE-ID***]
```

2. Sign in to Cloudera Manager.
3. Navigate to Administration External Accounts AWS Credentials Add Access Key Credentials .
4. Enter the S3 credentials for Hive.



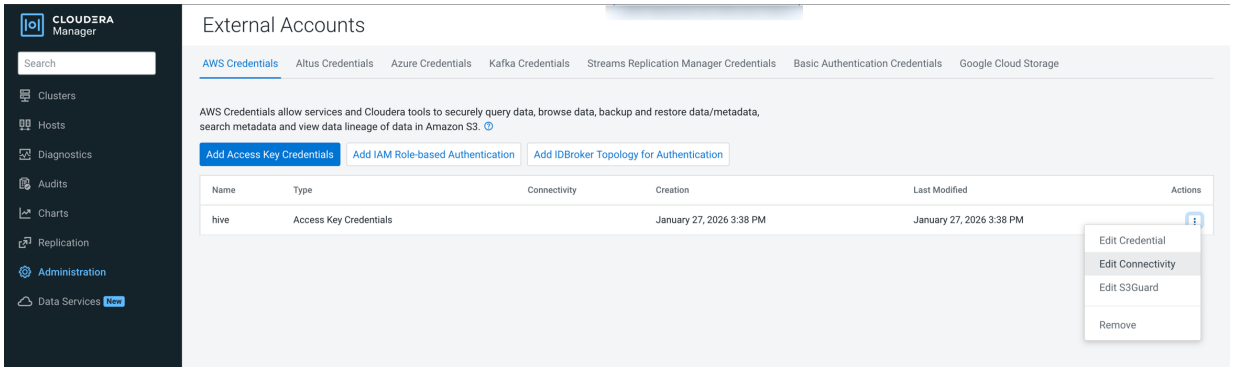
5. Click Add.

Enabling Hive and Impala to use the credentials

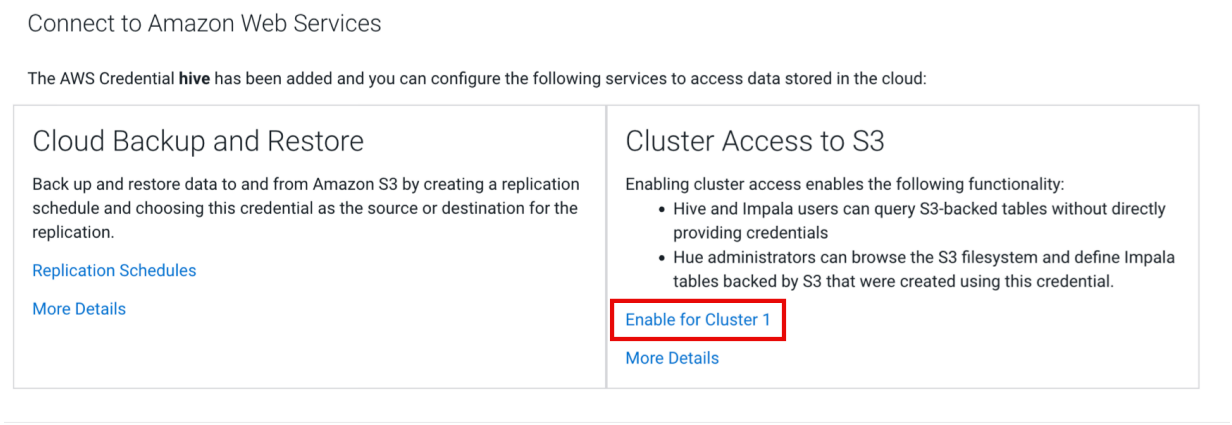
After setting up the external account for Ozone S3 access, you must enable Hive and Impala to use the credentials.

Procedure

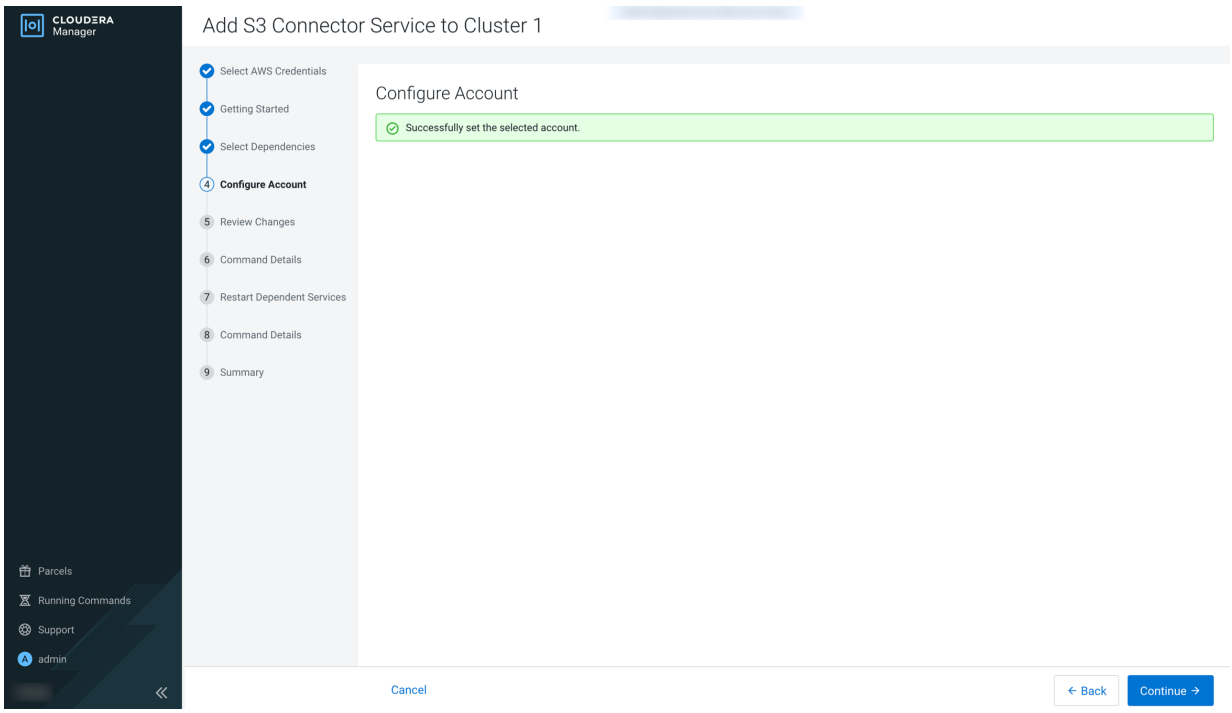
1. Sign in to Cloudera Manager.
2. Navigate to Administration External Accounts AWS Credentials .
3. Click on the three dots next to the Hive credentials and then click Edit Connectivity.



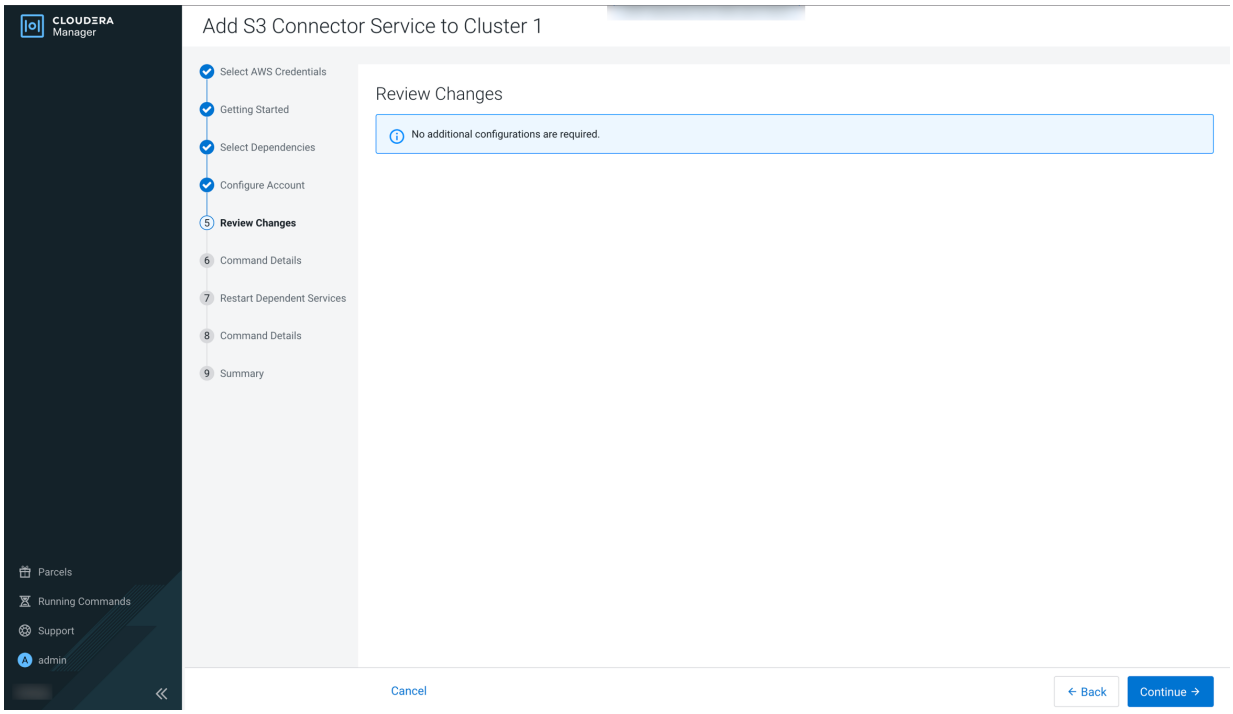
4. Under Cluster Access to S3 section, click Enable for *[***CLUSTER-NAME***]*.



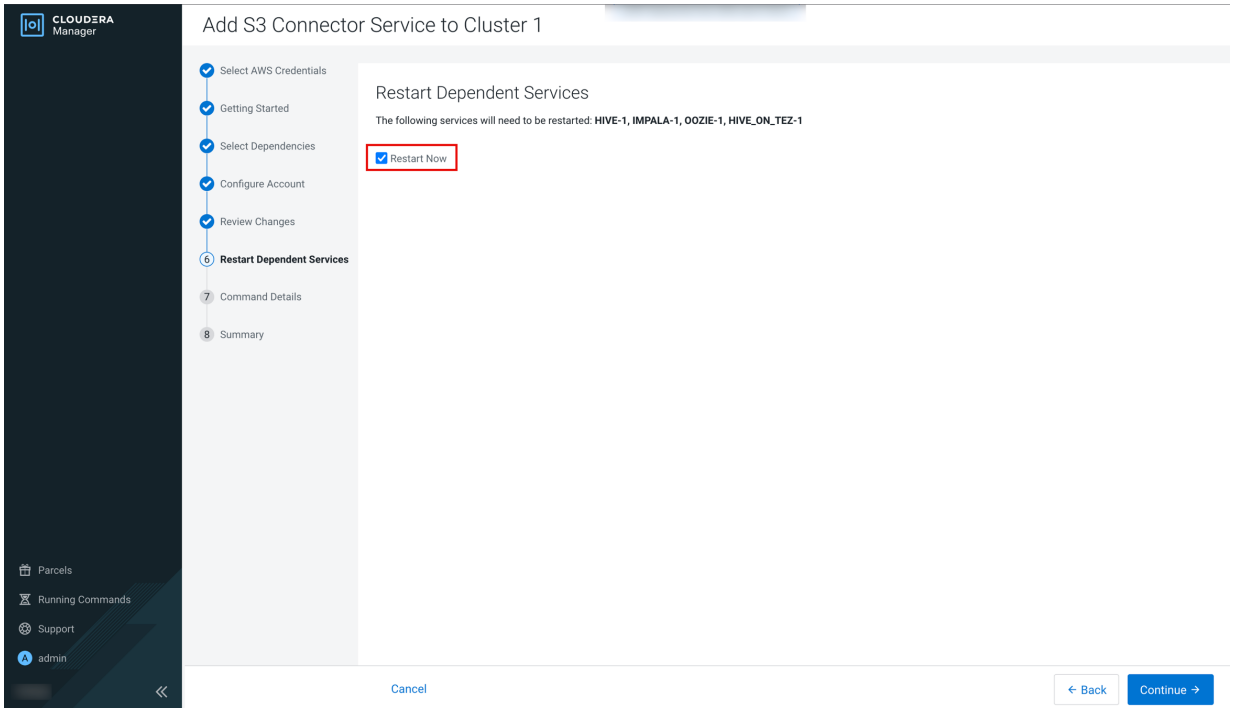
5. After enabling your cluster, configure the account on the Configure Account page. Click Continue.



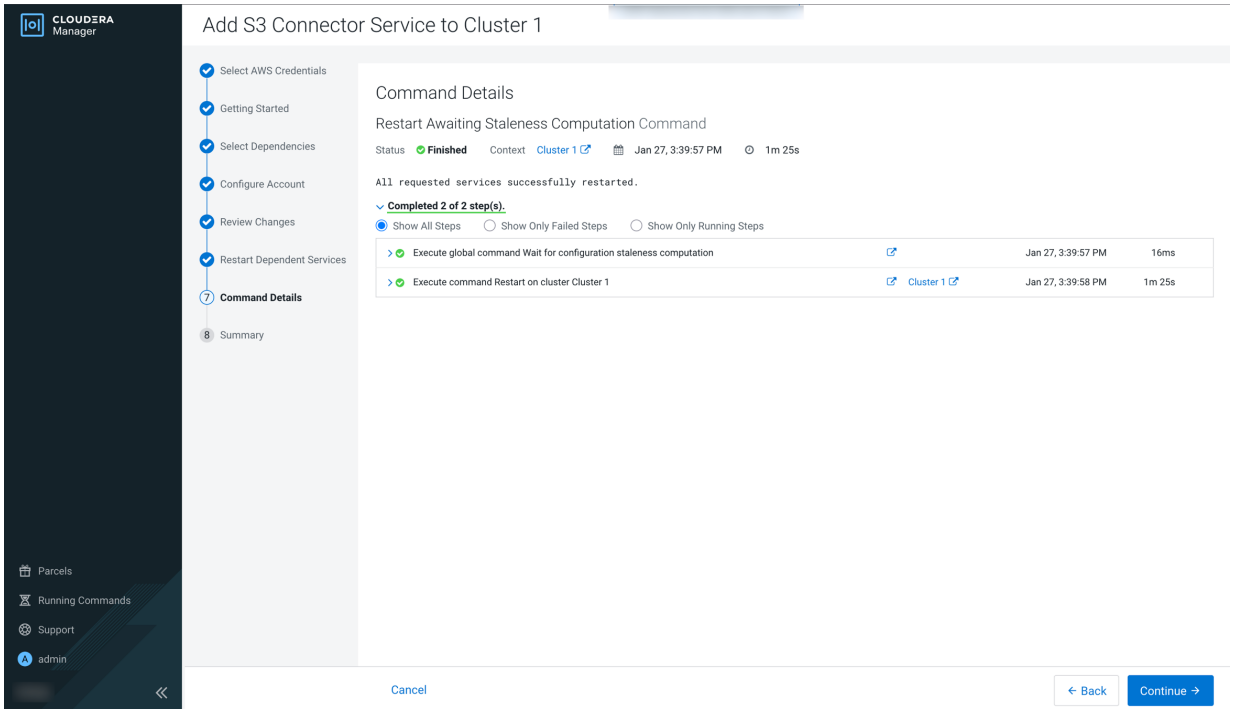
6. You can review the changes on the Review Changes page. Click Continue.



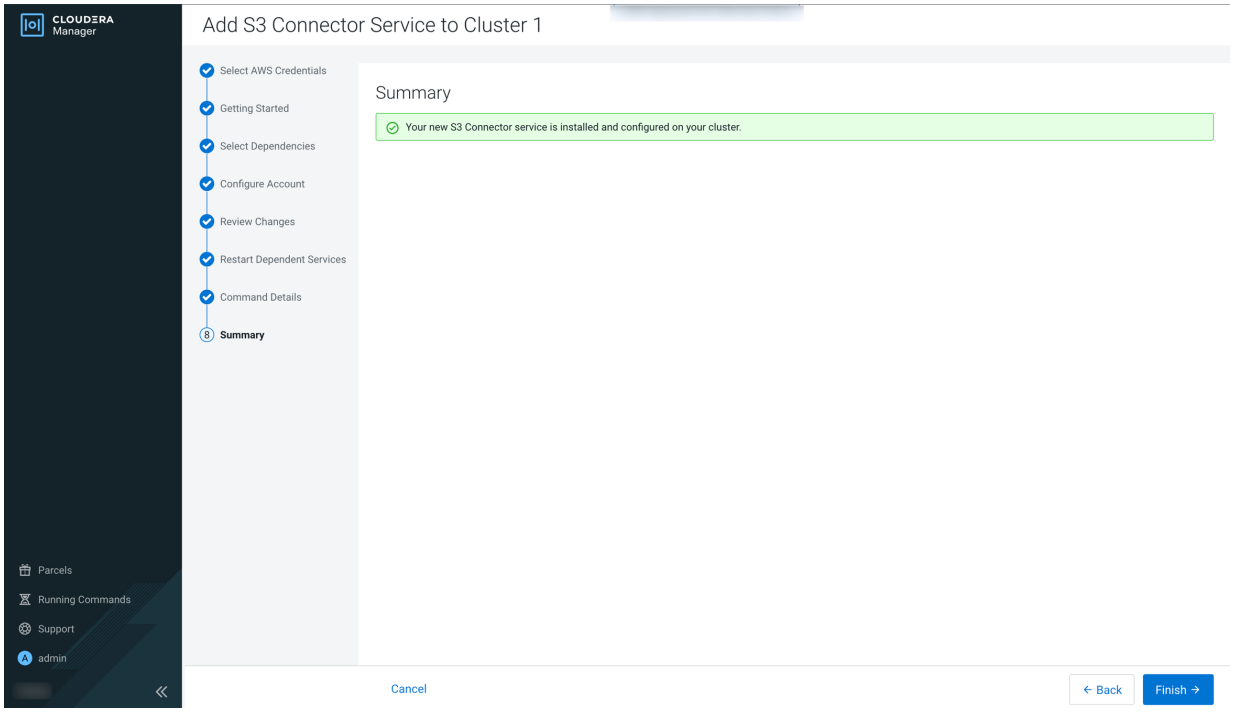
7. On the Restart Dependent Services page, select the Restart Now checkbox and click Continue.



8. On the Command Details page, you can monitor the progress of restarting the dependent services. After the restart process is complete, click Continue.



9. On the Summary page, click Finish.

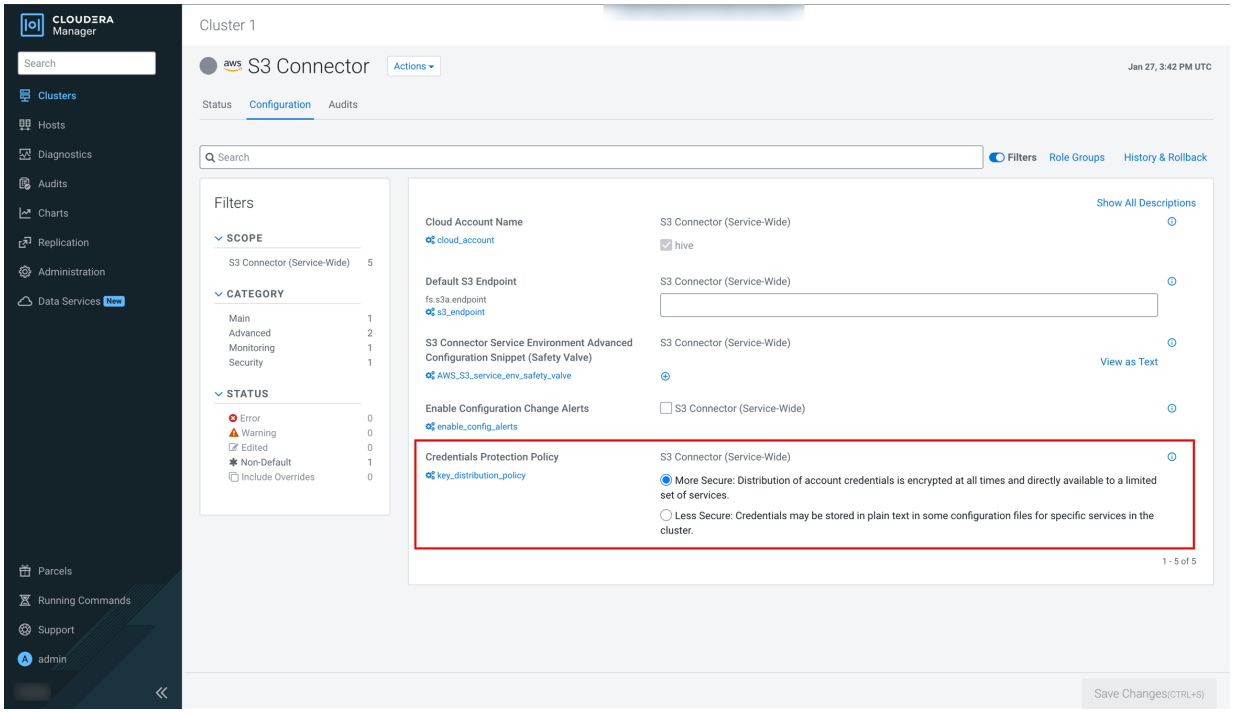


Setting up S3 connector

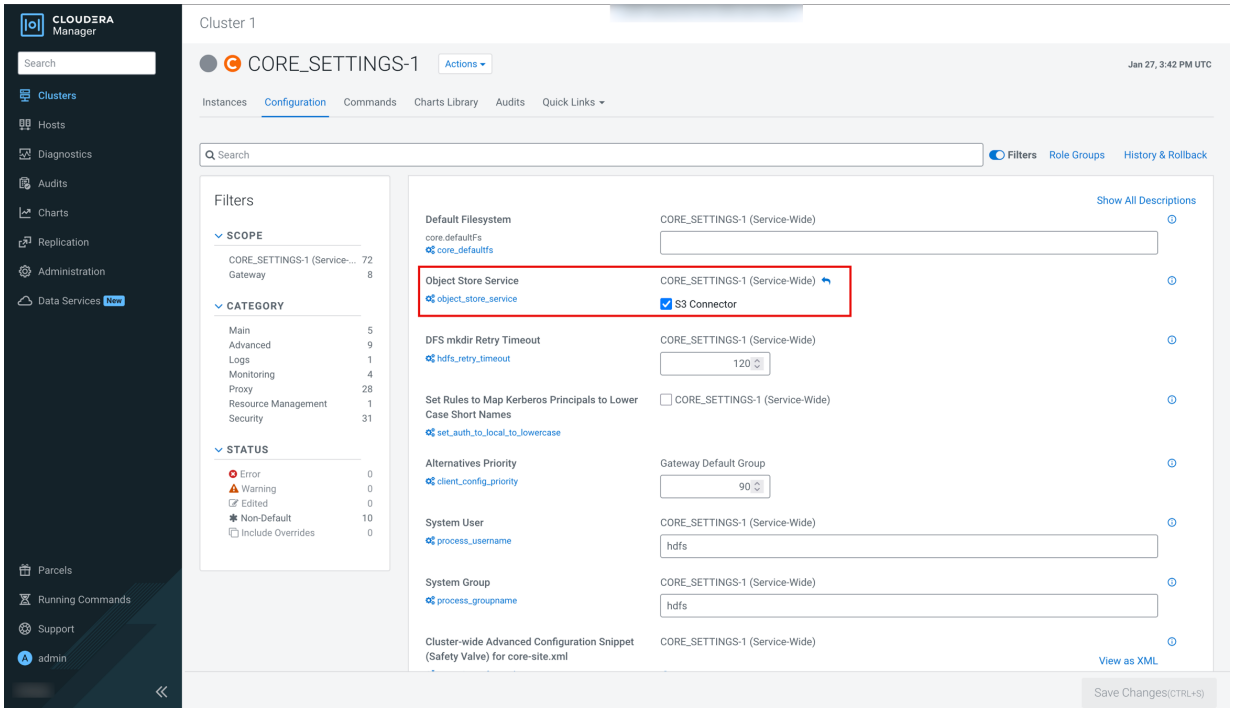
Learn about how to set up the Ozone S3 connector.

Procedure

1. Sign in to Cloudera Manager.
2. To add S3 connector service, navigate to Clusters Add Service S3 Connector .
3. Navigate to Clusters S3 Connector Configuration Credentials Protection Policy , select More Secure option and click Save Changes.



4. To setup the S3 credentials for authentication to Ozone Object Store, navigate to Clusters Configurations > Object Store Service , select S3 connector check box and click Save Changes.



Note:

- These credentials are not distributed to any other services other than Hive and/or Impala.
- The credentials are encrypted and obfuscated.
- An additional service S3 connector is added to the cluster and is linked to CORE_SETTINGS. You can unlink the S3 connector to undistribute the credentials from Hive and Impala.

Configuring Hive access for Ozone using S3A

Learn about how to configure Hive access for Ozone using S3A.

Before you begin

- Set up an external account for Ozone S3 access.
- Enable Hive and Impala to use the credentials.
- Set up the S3 connector.
- You must import the CA certificate to run Accessing Ozone S3 using S3A FileSystem.
- Perform the following steps to add the following configurations to S3 configurations for Ozone access:

Important: Make sure you do not add any S3 credentials.

1. Sign in to Cloudera Manager.
2. Navigate to Clusters Hive Configurations Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml .
3. Add the following properties:

```
fs.s3a.endpoint = <Ozone S3 endpoint URL>
fs.s3a.bucket.probe = 0
fs.s3a.change.detection.version.required = false
fs.s3a.path.style.access = true
fs.s3a.change.detection.mode = none
```

4. Navigate to Clusters Hive_On_Tez Configurations Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml .
5. Add the following properties:

```
fs.s3a.endpoint = <Ozone S3 endpoint URL>
fs.s3a.bucket.probe = 0
fs.s3a.change.detection.version.required = false
fs.s3a.path.style.access = true
fs.s3a.change.detection.mode = none
```

6. Restart the Stale services on the Clusters.
7. Navigate to Hive Service Actions and click Deploy client configuration.

- Provide the required permissions in Ranger to the user running the queries.

  Example: Provide a user with all the permissions:

  - Assign the user with all permissions to the Database, table/udf, and URL resources in a HadoopSQL resource-based policy.
  - Assign the user with S3_VOLUME_POLICY in an Ozone policy.

  You can change the permissions based on your requirements.

About this task

Perform the following steps to sign in to the Hive shell, create a Hive table using S3A, add data to the table, and view the added data. You can perform the same procedure by signing on to Cloudera Data Explorer (Hue) using the Hive or Beeline shell.

Procedure

1. Create an Ozone bucket by running the following command:

```
ozone sh bucket create /s3v/[***BUCKET-NAME***]
```

   Example: The following example shows how you can create a bucket named s3hive:

```
ozone sh bucket create /s3v/s3hive
```

2. Sign in to the Hive shell and perform the following steps:
   a) Create a table on Ozone using S3A.

```
jdbc:hive2://> create external table mytable1(key string, value int) loc
ation 's3a://s3hive/mytable1';
```

   b) Add the data to the table.

```
jdbc:hive2://> insert into mytable1 values("cldr",1);
jdbc:hive2://> insert into mytable1 values("cldr-cdp",1);
```

   c) View the data added to the table.

```
jdbc:hive2://> select * from mytable1;
```

   Example,

```
+-------------------+----------------------+
| mytable1.key  | mytable1.value  |
+-------------------+----------------------+
| cldr              |     1                |
+-------------------+----------------------+
| cldr-cdp          |     1                |
```

```
+-----------------+---------------------+
```

Configuring Impala access for Ozone using S3A

Learn about how to configure Impala access for Ozone using S3A.

Before you begin

- Set up an external account for Ozone S3 access.
- Enable Hive and Impala to use the credentials.
- Set up the S3 connector.
- You must import the CA certificate to run Accessing Ozone S3 using S3A FileSystem.
- Perform the following steps to add the following configurations to S3 configurations for Ozone access:

  Important: Make sure you do not add any S3 credentials.

  1. Sign in to Cloudera Manager.
  2. Navigate to Clusters Impala Configurations Impala Catalog Server Advanced Configuration Snippet (Safety Valve) for core-site.xml .
  3. Add the following properties:

     ```
     fs.s3a.endpoint = <Ozone S3 endpoint URL>
     fs.s3a.bucket.probe = 0
     fs.s3a.change.detection.version.required = false
     fs.s3a.path.style.access = true
     fs.s3a.change.detection.mode = none
     ```

  4. Navigate to Clusters Impala Configurations Impala Daemon Advanced Configuration Snippet (Safety Valve) for core-site.xml .
  5. Add the following properties:

     ```
     fs.s3a.endpoint = <Ozone S3 endpoint URL>
     fs.s3a.bucket.probe = 0
     fs.s3a.change.detection.version.required = false
     fs.s3a.path.style.access = true
     fs.s3a.change.detection.mode = none
     ```

- Provide the required permissions in Ranger to the user running the queries.

  Example: Provide a user with all the permissions:

  - Assign the user with all permissions to the Database, table/udf, and URL resources in a HadoopSQL resource-based policy.
  - Assign the user with S3_VOLUME_POLICY in an Ozone policy.

  You can change the permissions based on your requirements.

Procedure

1. Create an Ozone bucket.

   ```
   ozone sh bucket create /s3v/[***BUCKET-NAME***]
   ```

   Example: The following example shows how you can create a bucket named s3impala:

   ```
   ozone sh bucket create /s3v/s3impala
   ```

2. Sign in to the Impala shell and perform the following steps:

a) Create a table on Ozone using S3A.

```
> create external table mytable2(key string, value int) location 's3a://
s3impala/mytable2';
```

b) Add the data to the table.

```
> insert into mytable2 values("cldr",1);
> insert into mytable2 values("cldr-cdp",1);
```

c) View the data added to the table.

```
> select * from mytable2;
```

Example,

```
+----------+-------+
| key      | value |
+----------+-------+
| cldr     | 1       |
+----------+-------+
+----------+-------+
| cldr-cdp | 1       |
+----------+-------+
```

Configuring Spark access for Ozone using S3A

You must configure specific properties for client applications such as Spark to access the Ozone data store using S3A.

Before you begin

- You must import the CA certificate to run Ozone S3 Gateway from the S3A filesystem.
- You must create an ozone-s3.properties file with the following configuration to run the Spark word count program:

```
spark.hadoop.fs.s3a.impl = org.apache.hadoop.fs.s3a.S3AFileSystem
spark.hadoop.fs.s3a.access.key = <ACCESS KEY>
spark.hadoop.fs.s3a.secret.key = <SECRET>
spark.hadoop.fs.s3a.endpoint = <OZONE S3 ENDPOINT URL>
spark.hadoop.fs.s3a.bucket.probe = 0
spark.hadoop.fs.s3a.change.detection.version.required = false
spark.hadoop.fs.s3a.change.detection.mode = none
spark.hadoop.fs.s3a.path.style.access = true
```

Note: In the list of configurations, replace the values of access key and secret from the output of ozone s3 getsecret --om-service-id=<*OZONE SERVICE ID*> and replace the Ozone S3 endpoint URL with the S3 Gateway URL of the Ozone cluster.

About this task

The following procedure explains how you can configure Spark access to Ozone using S3A and run a word count program from the Spark shell.

Procedure

1. Create an Ozone bucket.

   The following example shows how you can create a bucket named sparkbucket:

   ```
   ozone sh bucket create /s3v/sparkbucket
   ```

2. Add data to the bucket.

   The following example shows how you can add data to the sparkbucket bucket:

```
hadoop fs  -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version
.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.
key=<ACCESSKEY> -Dfs.s3a.secret.key=<SECRET> -Dfs.s3a.endpoint=<S3
 ENDPOINT URL> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.ha
doop.fs.s3a.S3AFileSystem -mkdir -p s3a://sparkbucket/input
```

```
hadoop fs  -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version
.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.
key=<ACCESSKEY> -Dfs.s3a.secret.key=<SECRET> -Dfs.s3a.endpoint=<S3
 ENDPOINT URL> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.ha
doop.fs.s3a.S3AFileSystem -put /tmp/key1 s3a://sparkbucket/input/key1
```

3. Start the Spark shell and wait for the prompt to appear.

```
spark-shell --properties-file <OZONE-S3.PROPERTIES>
```

4. Create a Resilient Distributed Dataset (RDD) from an Ozone file and enter the specified command on the Spark shell.

```
var lines = sc.textFile("s3a://sparkbucket/input/key1")
```

5. Convert each record in the file to a word.

```
var words = lines.flatMap(_.split(" "))
```

6. Convert each word to a key-value pair.

```
var wordsKv = words.map((_, 1))
```

7. Group each key-value pair by key and perform aggregation on each key.

```
var wordCounts = wordsKv.reduceByKey(_ + _ )
```

8. Save the results of the grouping and aggregation operations to Ozone.

```
wordCounts.saveAsTextFile("s3a://sparkbucket/output")
```

9. Exit the spark shell and view the results through S3A.

```
hadoop fs  -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version
.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.
key=<ACCESSKEY> -Dfs.s3a.secret.key=<SECRET> -Dfs.s3a.endpoint=<OZONE S3
 ENDPOINT URL> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.ha
doop.fs.s3a.S3AFileSystem -ls -R s3a://sparkbucket/
```

```
hadoop fs  -Dfs.s3a.bucket.probe=0 -Dfs.s3a.change.detection.version
.required=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.
key=<ACCESSKEY> -Dfs.s3a.secret.key=<SECRET> -Dfs.s3a.endpoint=<OZONE S3
 ENDPOINT URL> -Dfs.s3a.path.style.access=true -Dfs.s3a.impl=org.apache.ha
doop.fs.s3a.S3AFileSystem -cat s3a://sparkbucket/output/part-00000
```

Overview of the Ozone Manager in High Availability

Configuring High Availability (HA) for the Ozone Manager (OM) enables you to run redundant Ozone Managers on your Ozone cluster and prevents the occurrence of a single point of failure in the cluster from the perspective of namespace management. In addition, Ozone Manager HA ensures continued interactions with the client applications for read and write operations.

Ozone Manager HA involves a leader OM that handles read and write requests from the client applications, and at least two follower OMs, one of which can take over as the leader in situations such as the following:

- Unplanned events such as a crash involving the node that contains the leader OM.
- Planned events such as a hardware or software upgrade on the node that contains the leader OM.



Important: Cloudera recommends enabling HDFS HA before creating Iceberg tables because enabling HDFS HA makes existing Iceberg tables unreadable. If HDFS is running without HA, export existing Iceberg tables to a non-Iceberg format before enabling HA, then re-create the Iceberg tables and reload the exported data. For more information, see Create table feature using Apache Iceberg.

Considerations for configuring High Availability on the Ozone Manager

There are various factors that you must consider when configuring High Availability (HA) for the Ozone Manager (OM).

- OM HA is automatically enabled when you set up Ozone on a Cloudera cluster with at least three nodes as OM hosts.
- You must define the OM on at least three nodes so that one OM node is the leader and the remaining nodes are the followers. The OM nodes automatically elect a leader.

  The following command lists the OM leader node and the follower nodes:

  ```
  ozone admin om getserviceroles -id=<ozone service id>
  ```

Ozone Manager nodes in High Availability

A High Availability (HA) configuration of the Ozone Manager (OM) involves one leader OM node and two or more follower nodes. The leader node services read and write requests from the client. The follower nodes closely keep track of the updates made by the leader so that in the event of a failure, one of the follower nodes can take over the operations of the leader.

The leader commits a transaction only after at least one of the followers acknowledges to have received the transaction.

Read and write requests with Ozone Manager in High Availability

Read requests from the client applications are directed to the leader Ozone Manager (OM) node. After receiving an acknowledgement to its request, the client caches the details of the leader OM node, and routes subsequent requests to this node.

If repeated requests to the designated leader OM node start failing or fail with a *NonLeaderException*, it could mean that the particular node is no longer the leader. In this situation, the client must identify the correct leader OM node and reroute the requests accordingly.

The following command lists the OM leader node and the follower nodes:

```
ozone admin om getserviceroles -id=<ozone service id>
```

In the case of write requests from clients, the OM leader services the request after receiving a quorum of acknowledgements from the follower.

Note: The read and write requests from clients could fail in situations such as a failover event or network failure. In such situations, the client can retry the requests.

Overview of Storage Container Manager in High Availability

Configuring High Availability (HA) for the Storage Container Manager (SCM) prevents the occurrence of a single point of failure in an Ozone cluster to manage the various types of storage metadata, and ensures continued interactions of the SCM with the Ozone Manager (OM) and the DataNodes.

SCM HA involves the following:

- A leader SCM that interacts with the OM for block allocations, and works with the DataNodes to maintain the replication levels required by the Ozone cluster.
- At least two follower SCMs that closely keep track of the updates made by the leader so that in the event of a failure, one of the follower nodes can take over the operations from the leader.

Important: Cloudera recommends enabling HDFS HA before creating Iceberg tables because enabling HDFS HA makes existing Iceberg tables unreadable. If HDFS is running without HA, export existing Iceberg tables to a non-Iceberg format before enabling HA, then re-create the Iceberg tables and reload the exported data. For more information, see Create table feature using Apache Iceberg.

Considerations for configuring High Availability on Storage Container Manager

Similar to configuring High Availability (HA) for the Ozone Manager (OM), there are various factors that you must consider when configuring HA for the Storage Container Manager (SCM).

- SCM HA is supported only on *new* Cloudera *cluster deployments starting with* Cloudera *7.1.7*. You can configure SCM HA when adding Ozone as a service through Cloudera Manager.

  Note: For information about adding and deleting services using Cloudera Manager, see the following:

  - Adding a service
  - Deleting services
- To configure SCM HA, you require at least three nodes as SCM hosts so that one SCM node is the leader and the remaining nodes are the followers. The SCM nodes automatically elect a leader.

  Note: The ozone admin scm roles command lists the leader and follower SCM nodes in an Ozone cluster.
- A primordial SCM node generates the cluster ID and distributes it across Ozone Manager and DataNodes in an Ozone cluster. A primordial SCM must be running for other SCMs in SCM HA setup to bootstrap initially. If there is an existing SCM instance running and you want to add a new SCM instance, the primordial node configuration needs to be the existing SCM instance only.

  Note: If the primordial SCM is not chosen correctly, the Ozone cluster can encounter issues from OMs and DNs can crash into SCMs.
- You must specify one of the three SCM host names as the primordial node using the ozone.scm.primordial.node.id property. In addition, you must specify the SCM service ID using the ozone.scm.service.id property.
- If a primordial SCM node is inaccessible, new SCM nodes *cannot* join an HA configuration.

Note:

- You can now configure Ozone Service ID through the Cloudera Manager setup wizard. However, you must not change the Ozone Service ID after the setup is complete. If you change the Ozone Service ID, Ozone Manager fails to start up.
- You can also configure primordial ID in Cloudera Manager while setting up Ozone as a service.
- Currently, Cloudera Manager does not have an option to disable parameters that can be configured after the setup.

- After you have configured the HA cluster, ensure that you *do not change* the SCM Ratis port number (9894).

Note: For information about the various Ozone ports, see Ports Used by Cloudera Runtime Components.

Storage Container Manager operations in High Availability

When an Ozone cluster has Storage Container Manager (SCM) High Availability (HA) configured, the important SCM operations; for example, managing client requests such as allocating containers, local operations such as destroying pipelines, and processing DataNode updates, are handled differently from a non-HA Ozone cluster.

Client request management

SCM clients are the different Ozone elements that interact with the SCM such as DataNodes, the Ozone Manager (OM) and so on.

On receiving client requests such as allocating a container or a pipeline, the leader SCM performs all the required operations. The leader also performs metadata changes that result from executing the client request, and accordingly updates Ratis. The changes are replicated to the followers through Ratis.

Performing operations local to the SCM

SCM performs local operations such as destroying pipelines, deleting stale DataNodes, and so on, when it stops receiving heartbeats or reports from DataNodes. The followers log the occurrences of these operations and their results. The leader performs any metadata changes that result from these local operations, and accordingly updates Ratis. The changes are replicated to the followers through Ratis.

Processing DataNode updates

The DataNodes send heartbeats and reports to all the SCMs so that they maintain consistent information about the health of the DataNodes. If the SCMs require to interact with the DataNodes, only the leader sends the required information while the followers update their states accordingly.

Note: Because newer heartbeats and reports can overwrite the existing information, the SCMs are eventually consistent with the DataNode heartbeats and reports.

Offloading Application Logs to Ozone

In order to offload logs from HDFS to Ozone, a small Ozone cluster can be deployed through Cloudera Manager and then by configuring Remote App Log Directory in YARN configs, customers can offload logs from HDFS to Ozone. This will help in freeing up space in HDFS metadata for more user data. After changing the configuration, logs for all YARN, Hive on Tez and Spark on Yarn are automatically redirected to Ozone.

Note: This does not require any changes in the application.

HDFS is used as the primary storage engine by most of the Big Data applications like Hive, YARN, Spark, and so on. Currently, it stores both the important data like Hive and Spark tables and logs generated by these applications.

The YARN Log Aggregation feature enables you to move local log files of any application onto HDFS or Apache Ozone depending on the cluster configuration. Fore more information, see YARN Log Aggregation Overview

Changing configuration

- In Cloudera Manager, select the service.
- Click the Configuration tab. Search for remote app log.
- In the Filters pane, under Scope, select NodeManager.
- In the Remote App Log Directory (yarn.nodemanager.remote-app-log-dir) field, add the following:

```
ofs://ozone1/logvol/logbuck/temp_log_dir
```

Removing Ozone DataNodes from the cluster

You can remove Ozone DataNodes from the Cloudera cluster in a controlled manner using Cloudera Manager for performing maintenance operations. If you want to remove the DataNodes permanently or for an unknown duration, you can decommission them. If you want to make the DataNodes unavailable for a short period of time, such as, for a few days or hours, you can place them in offline mode.

When you initiate the process of decommissioning a DataNode, Ozone automatically ensures that all the storage containers on that DataNode have an additional copy created on another DataNode before the decommission completes. Similarly, when you initiate the process of placing a DataNode in offline mode, Ozone ensures that at least two copies of the DataNode's storage containers are present on other nodes before the particular DataNode enters offline mode.

Note:

- Before a DataNode enters offline mode, you can reduce the minimum number of online copies of the storage container from two to one. This process reduces the time to complete the offline mode operation. However, the process increases the risk of data becoming temporarily unavailable if another DataNode fails.

  For details on how to reduce the minimum number of storage container copies, see Configuring the number of storage container copies for a DataNode.
- Ozone does not specify any upper limit on the number of DataNodes you can simultaneously decommission or place in offline mode. However, there must be enough space on the cluster to hold the additional storage containers. Otherwise, the DataNodes cannot complete the decommissioning or offline mode processes.

You can also recommission a DataNode that is already decommissioned or placed in offline mode. When you recommission such a DataNode, Ozone automatically removes any excess containers created during the decommission or offline process.

Decommissioning Ozone DataNodes

You can remove Ozone DataNodes from the cluster by decommissioning the DataNode instances using Cloudera Manager.

Before you begin

Ensure that the cluster has sufficient space to hold the additional storage containers of the DataNodes that you are decommissioning.

Procedure

1. In Cloudera Manager, go to the Ozone service.
2. Click Instances.
3. Select the DataNode instances that you want to decommission.
4. Select Actions for Selected>Decommission.
5. Click Decommission to confirm.

   Ozone initiates decommissioning of the selected DataNodes. The process takes time depending on the number of storage containers to replicate. After the process is complete, the Instances page shows the Commissioned State of the selected DataNodes as Decommissioned.

Placing Ozone DataNodes in offline mode

You can temporarily remove Ozone DataNodes from the cluster by placing the DataNode instances in offline mode using Cloudera Manager.

Before you begin

Ensure that the cluster has sufficient space to hold the additional storage container copies belonging to the DataNode that you are placing in offline mode.

Procedure

1. In Cloudera Manager, go to the Ozone service.
2. Click Instances.
3. Select the DataNode instances that you want to place in offline mode.
4. Select Actions for Selected>Enter Offline mode.
5. Click Enter Offline mode to confirm.

   Ozone starts preparing the selected DataNodes for offline mode. The process takes time depending on the number of storage containers to replicate. After the process is complete, the DataNode is stopped, and the Instances page shows the Commissioned State of the selected DataNodes as Offlined.

Configuring the number of storage container copies for a DataNode

By default, Ozone ensures that at least two copies of any container stored on a DataNode entering the offline mode are available on other nodes in the cluster. To reduce the time for nodes to enter offline mode, you can reduce the number of copies to one.

Procedure

1. In Cloudera Manager, go to the Ozone service.
2. Click Configuration.
3. Search for the Storage Container Manager Advanced Configuration Snippet (Safety Valve) for ozone-conf/ozone-site.xml property, and specify the following values:
   - Name: hdds.scm.replication.maintenance.replica.minimum
   - Value: 1
4. Click Save.
5. Restart the Storage Container Manager (SCM) instances from the Instances page.

Recommissioning an Ozone DataNode

You can add an Ozone DataNode that is already decommissioned or in offline mode back to the cluster using Cloudera Manager.

About this task

After decommissioning and deleting a DataNode instance, if you try adding the DataNode instance to the same host as before, Cloudera Manager considers the newly added DataNode instance as Commissioned. However, the Storage Container Manager recognizes the DataNode ID and treats the newly added DataNode as Decommissioned. To address this discrepancy, you must recommission the DataNode.

Procedure

1. In Cloudera Manager, go to the Ozone service.
2. Click Instances.
3. Select the DataNode instances that you want to recommission.
4. Select Actions for Selected>Recommission and Start.
5. Click Recommission and Start to confirm.

   The selected DataNodes rejoin the cluster and Instances page shows the Commissioned State of the DataNodes as Commissioned.

Handling datanode disk failure

If there is a disk failure on a datanode, you must place the node in offline mode, stop the node, replace the disk, start the node, and recommission the node to remove it from offline mode. Perform the following steps:

Procedure

1. Log in to Cloudera Manager.
2. Navigate to Clusters.
3. Select the Ozone service
4. Place the datanode in offline mode. See Placing Ozone DataNodes in offline mode.
5. Stop the node.
6. Replace the failed disk(s). If the new disk is mounted to a different location than the old disk, you will need to update the configurations accordingly.
   a) Go to Configurations
   b) To update the path to a Ratis storage disk, update the corresponding entry in dfs.container.ratis.datanode.storage.dir to point to the new disk's mount point.
   c) To update the path to a data storage disk, update the corresponding entry in hdds.datanode.dir to point to the new disk's mount point.
7. Restart the node.
8. Recommission the Datanode to remove it from offline mode. See Recommissioning an Ozone DataNode.



Note: In the event of complete node failure, you must decommission the node. For more information on decommissioning the node, see Decommissioning Ozone DataNodes.

Multi-Raft configuration for efficient write performances

Multi-Raft configuration improves write performances in Ozone by including DataNodes in multiple pipelines.

Ozone uses the Raft protocol for replicating data across the cluster and providing consistent write performances among the DataNodes. Ozone stores data in a number of containers throughout the cluster and each container allocates data blocks on DataNodes. In addition, Ozone creates pipelines, as logic groups, to assemble containers from several DataNodes for redundancy purposes. Each pipeline consists of DataNodes in a Raft group such that there are three DataNodes with one of them being the leader. Through the pipeline, data is written to the leader, which replicates the same to the followers. A pipeline uses RaftLog to spread the write load on disks.

In a single-Raft configuration, a DataNode can join only one pipeline. This can slow down the write performance to the DataNodes and impact the efficiency with which disks are used on various DataNodes. Starting with Cloudera Runtime 7.1.6, a multi-Raft configuration is available on Ozone by default. A multi-Raft configuration is beneficial because it increases the write efficiency by providing for more containers and pipelines without increasing the number of nodes or disks. Further, multi-Raft configuration also leads to more efficient use of DataNodes and disks in spreading the writes throughout the cluster.

Configuration properties for pipeline limits

To configure the pipeline limits for a multi-Raft configuration, you can set up certain properties as advanced configuration snippets using the Ozone Service Advanced Configuration Snippet (Safety Valve) for ozone-conf/ozone-site.xml property.

Property	Description	Default value
ozone.scm.datanode.pipeline.limit	Limit for the number of 3-node pipelines that a DataNode can join.	2 If you want to increase the number of pipelines based on the number of RaftLog disks, then you can set this value to 0.
ozone.scm.pipeline.per.metadata.disk	Number of pipelines to create for a Raft-log disk.	2

Working with the Recon web user interface

Recon is a centralized monitoring and management service within an Ozone cluster that provides information about the metadata maintained by different Ozone components such as the Ozone Manager (OM) and the Storage Container Manager (SCM).

Recon keeps track of the metadata as the cluster is operational, and displays the relevant information through a dashboard and different views on the Recon web user interface. This information helps in understanding the overall state of the Ozone cluster.

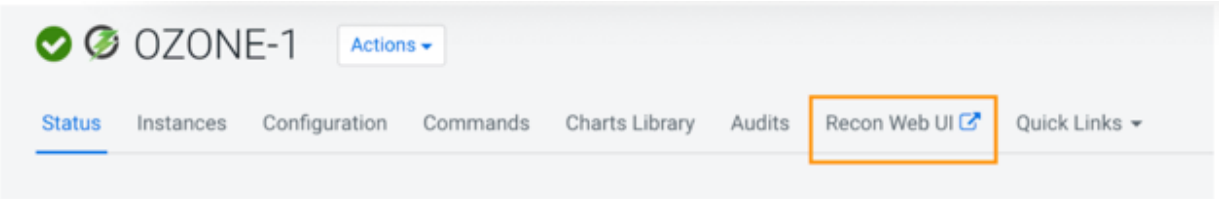
The metadata that components such as the OM and the SCM maintain are quite different from one another. For example, the OM maintains the mapping between keys and containers in an Ozone cluster while the SCM maintains information about containers, DataNodes, and pipelines. The Recon web user interface provides a consolidated view of all these elements.

Access the Recon web user interface

You can launch the Recon web user interface from Cloudera Manager. Recon starts its HTTP server over port 9888 by default. The default port is 9889 when auto-TLS is enabled.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Click Recon Web UI.



The Recon web user interface loads in a new browser window.

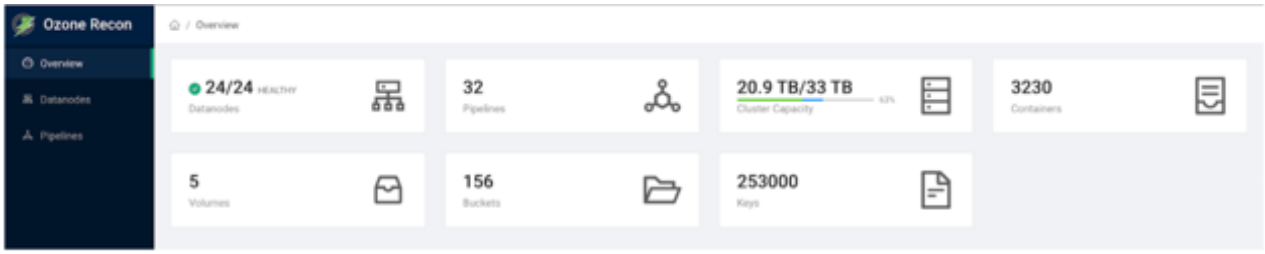
Elements of the Recon web user interface

The Recon web user interface displays information about the Ozone cluster on the following pages: Overview, DataNodes, and Pipelines. In addition, a separate page displays information about any missing storage containers.

Overview page

The Overview page displays information about different elements on the Ozone cluster in the form of a consolidated dashboard. This page loads by default when you launch the Recon web user interface.

Note: Recon interacts with the Storage Container Manager (SCM), the DataNodes, and the Ozone Manager (OM) at specific intervals to update its databases and reflect the state of the Ozone cluster, and then populates the **Overview** page. Therefore, the information displayed on the **Overview** page might occasionally not be in synchronization with the current state of the Ozone cluster because of a time lag. However, Recon ensures that the information is eventually consistent with that of the cluster.



Recon displays the following information from the SCM and the DataNodes on the **Overview** page in the form of cards:

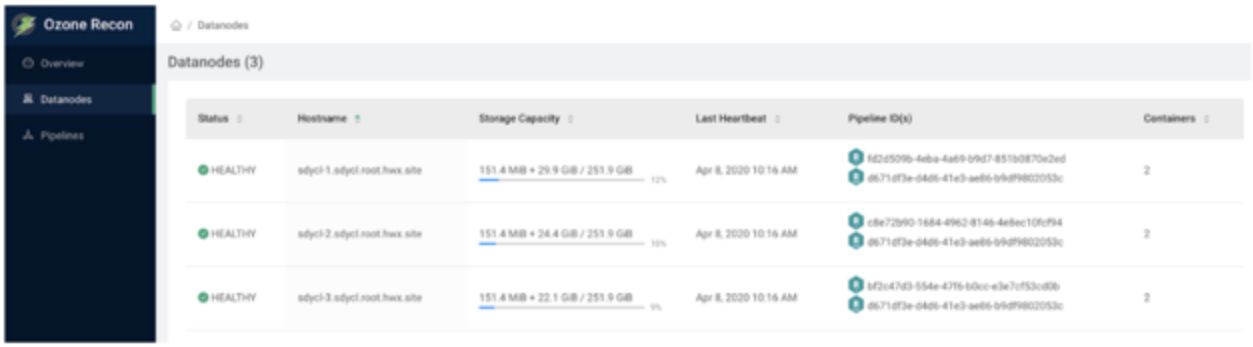
- Health of the DataNodes in the cluster. Clicking this card loads the DataNodes page.
- Number of pipelines involved in data replication. Clicking this card loads the Pipelines page.
- Capacity of the cluster. The capacity includes the amount of storage used by Ozone, by services other than Ozone, and any remaining storage capacity of the cluster.
- Number of storage containers in the SCM. If there are any missing containers reported, the Containers card is highlighted with a red border. You can then click the card to view more information about the missing containers on a separate page.

Recon displays following information from the Ozone Manager (OM) on the **Overview** page:

- Number of volumes in the cluster
- Total number of buckets for all the volumes in the cluster
- Total number of keys for all the buckets in the cluster

DataNodes page

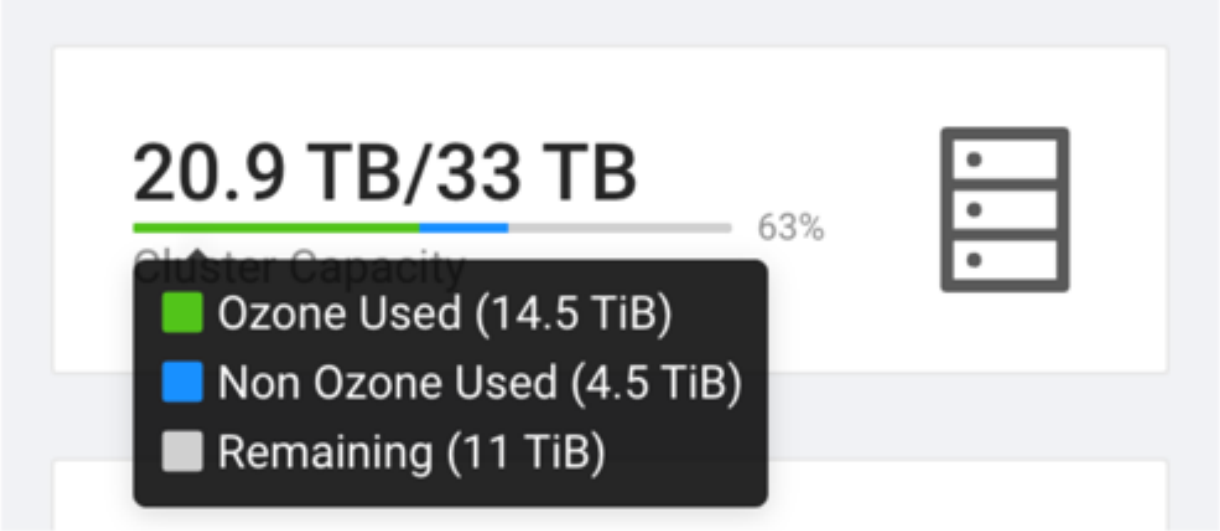
The DataNodes page displays information about the state of the DataNodes in a tabular format. You can load this page either by clicking the DataNodes tab on the left pane or the DataNodes card on the **Overview** page.



The following columns of the table provide details of the DataNodes:

- Status: The health status of the particular DataNode. The status can be either of the following:
  - HEALTHY: Indicates a normal functional DataNode.
  - STALE: Indicates that the SCM has not received a heartbeat from the DataNode for a certain period of time after the previous heartbeat.
  - DEAD: Indicates that the SCM has not received a heartbeat beyond a certain period of time since receiving the previous heartbeat. The time period beyond which the DataNode can be categorized as DEAD is configurable. The default value is five minutes. Until this threshold is reached, the DataNode is in a STALE state.
  - DECOMMISSIONING: Indicates that the DataNode is being decommissioned.
- Hostname: The cluster host that contains the particular DataNode.
- Storage Capacity: The storage capacity of the particular DataNode. The capacity information includes the amount of storage used by Ozone, by services other than Ozone, and any remaining storage capacity of the host.

  Hovering your mouse pointer over a particular entry displays the detailed capacity information as a tool tip.

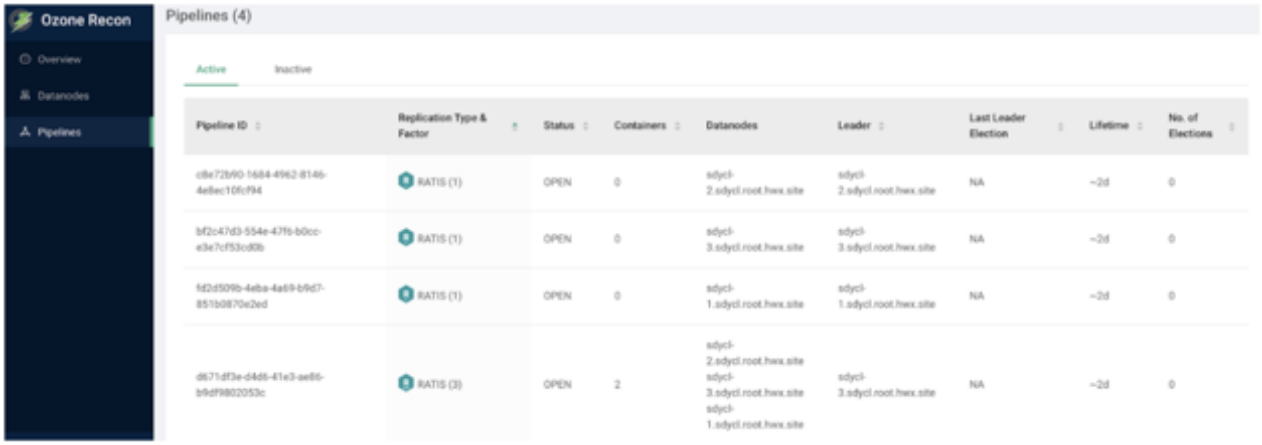


- Last Heartbeat: The timestamp of the last heartbeat sent by the particular DataNode to the SCM.
- Pipeline ID(s): The IDs of the pipelines to which the particular DataNode belongs.
- Containers: The number of storage containers inside the particular DataNode.

Pipelines page

The Pipelines page displays information about active pipelines including their IDs, the corresponding replication factors and the associated DataNodes. The page does not display any inactive pipelines.

An active pipeline is one that continues to participate in the replication process. In contrast, an inactive pipeline contains DataNodes that are dead or inaccessible, leading to the removal of its metadata from the Recon database, and eventually the destruction of the pipeline itself.



The page displays Pipeline information in a tabular format. The following columns provide the required information:

- Pipeline ID(s): The ID of a particular pipeline.
- Replication Type & Factor: The type of replication and the corresponding replication factor associated with a particular pipeline. The replication types are Standalone and Ratis. Accordingly, the default replication factor is three for Ratis and one for Standalone.
- Status: Specifies whether the particular pipeline is open or closed.
- DataNodes: The DataNodes that are a part of the particular pipeline.
- Leader: The DataNode that is elected as the Ratis leader for the write operations associated with the particular pipeline.
- Lifetime: The period of time for which the particular pipeline is open.
- Last Leader Election: The timestamp of the last election of the leader DataNode associated with this pipeline.

  Note: This field does not show any data for the current release.

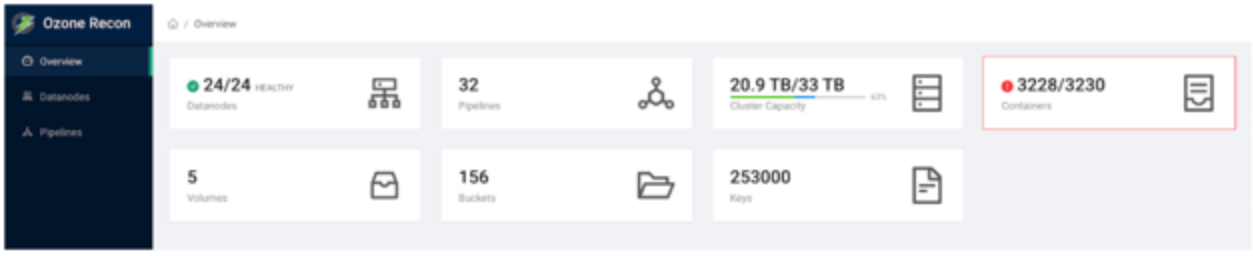
- No. of Elections: The number of times the DataNodes associated with the pipeline have elected a leader.

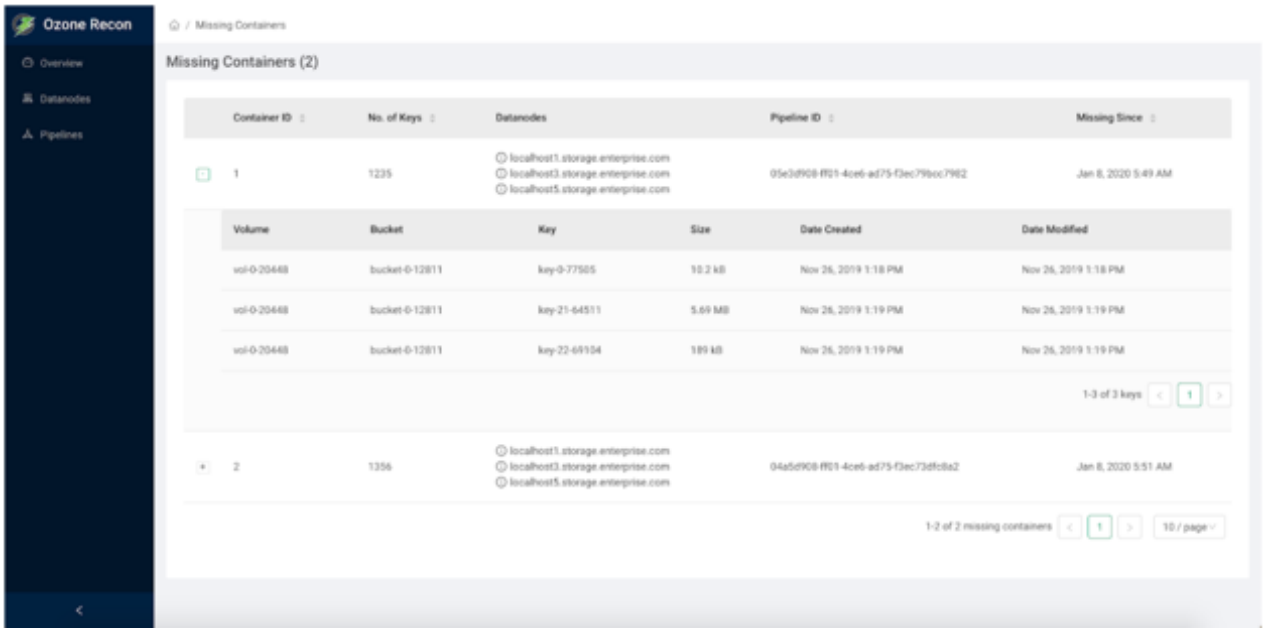
  Note: This field does not show any data for the current release.

Missing Containers page

There can be situations when a storage container or its replicas are not reported in any of the DataNode reports to the SCM. Such containers are flagged as missing containers to Recon. Ozone clients cannot read any blocks that are present in a missing container.

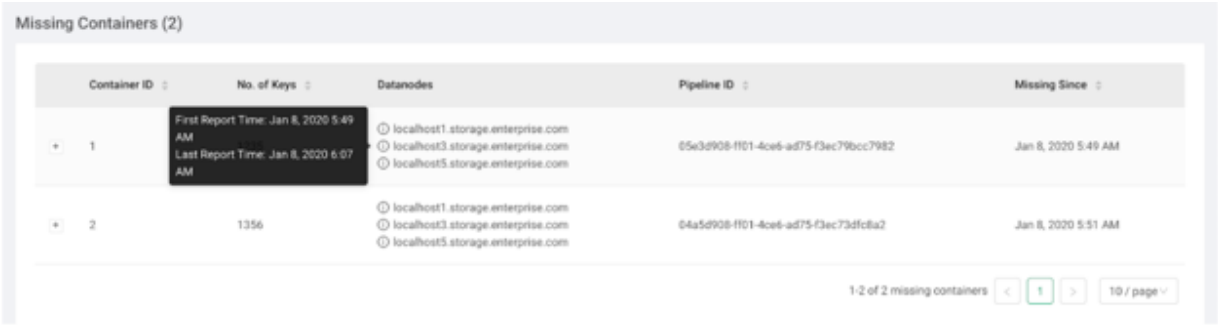
The Containers card on the Overview page of the Recon web user interface is highlighted with a red border in the case of missing containers. Clicking the card loads the **Missing Containers** page.





The page displays information about missing containers in a tabular format. The following columns provide the required information:

- Container ID: The ID of the storage container that is reported as missing due to the unavailability of the container and its replicas. Expanding the + sign next to a Container ID displays the following additional information:
  - Volume: The name of the volume to which the particular key belongs.
  - Bucket: The name of the bucket to which the particular key belongs.
  - Key: The name of the key.
  - Size: The size of the key.
  - Date Created: The date of creation of the key.
  - Date Modified: The date of modification of the key.
- No of Keys: The number of keys that were a part of the particular missing container.
- DataNodes: A list of DataNodes that had a replica of the missing storage container. Hovering your mouse pointer on the information icon shows a tool tip with the timestamp when the container replica was first and last reported on the DataNode.



Configuring Ozone to work with Prometheus

You can configure your Ozone cluster to enable Prometheus for real time monitoring of the cluster.

About this task

To enable Prometheus to work on your Ozone cluster, use Cloudera Manager to add the Ozone Prometheus role instance.

Note: The Prometheus binary is not available in Cloudera Base on premises 7.1.9 for the Ubuntu operating system, you can install Prometheus separately and specify the path to the parent directory, for example /usr/bin, in the prometheus.location parameter in Ozone.

Procedure

1. Log in to Cloudera Manager
2. Navigate to Clusters
3. Select the Ozone service
4. Add the Ozone Prometheus role instance to the Ozone service.

   For more information about adding role instances using Cloudera Manager, see Adding a role instance.

   Note: If you do not see Ozone Prometheus in the list of role instances to configure, it means that the role instance is not configured correctly. In this situation, the Prometheus logs (/var/log/hadoop-ozone/ozone-prometheus.log) on the Prometheus instance host show a FileNotFound error.
5. Start the Ozone Prometheus role instance.

   For information about starting role instances using Cloudera Manager, see Starting, stopping, and restarting role instances.

   After starting the role instance, the Prometheus Web UI quick link is added to the Ozone Prometheus page on Cloudera Manager.
6. Click the Prometheus Web UI quick link to launch the web user interface on a separate browser window.

   The metrics drop-down list displays various metrics from the Ozone daemons.
7. Select any metric from the drop-down list or enter the name of a metric and click Execute.

   Click the Graph or Console tab to view further details.

Important: Ozone Prometheus inherits its TLS protocol and cipher configuration from Ozone's settings. However, because Prometheus is a Go-based application, Go imposes additional limitations and permits only the following cipher suites:

- TLS 1.2 Ciphers:
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA
  - TLS_RSA_WITH_AES_128_GCM_SHA256
  - TLS_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  - TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS 1.3 Ciphers:
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256

Ozone Prometheus disregards all other cipher suites configured for the Ozone service and enforces only the supported suites listed above for each protocol.

Ozone trash overview

The Ozone trash feature helps prevent accidental deletion of files and directories.

When you delete a file in Ozone, the file is not immediately removed from Ozone. The deleted files are first moved to the /user/<username>/.Trash/Current directory, with their original filesystem path being preserved. After a user-configurable period of time (fs.trash.interval), a process known as trash checkpointing renames the Current directory to the current timestamp, that is, /user/<username>/.Trash/<timestamp>. The checkpointing process also checks the rest of the .Trash directory for any existing timestamp directories and removes them from Ozone permanently. You can restore files and directories in the trash simply by moving them to a location outside the .Trash directory.

Configuring the Ozone trash checkpoint values

You can use Cloudera Manager to configure the time period after which an Ozone trash checkpoint directory is deleted and the time interval between the creation of trash checkpoint directories.

Before you begin

You must ensure that you have set the Filesystem Trash Interval property. For details, see Setting the trash interval.

Procedure

1. Log in to Cloudera Manager.
2. Navigate to Clusters.

3. Select the Ozone service.
4. Click Configuration.
5. Search for the Ozone Filesystem Trash Interval property and set its value.

   This property specifies the time period after which an Ozone trash checkpoint directory is deleted. The default value is 1 day. Setting the value to 0 disables the trash feature.
6. Search for the Ozone Filesystem Trash Checkpoint Interval property and set its value.

   This specifies the time period between trash checkpoints, and its value must be less than the value of the Ozone Filesystem Trash Interval property. Setting the value to 0 implies that the value of Ozone Filesystem Trash Interval is used to determine the interval between trash checkpoints.

Ozone topology awareness

Ozone can use topology-related information like rack placement to optimize read and write pipelines.

To get a full rack-aware cluster, Ozone requires the following configurations:

- Ozone-configured topology hierarchy information.
- Write path configuration: When Ozone chooses 3 datanodes for a specific pipeline for the open container (WRITE), topology information can be used.
- When Ozone reads a Key, Ozone should prefer to read from the closest node.
- Container replication configuration: When Ozone Container Balancer or Replication Manager wants to move or replicate a replica of a closed container, topology information can be used.

For more information on Open vs Closed containers, see the Containers documentation.

Caution: You must not use / in the rack name and node name. / is the layer separator in the topology hierarchy. For example, if there are three layers of hierarchy and you use / in the rack name, Ozone interprets it as four layers.

Topology hierarchy

Learn about how to configure the topology hierarchy, static list, dynamic list, write path, read path and container replication.

To configure the topology hierarchy, use the net.topology.node.switch.mapping.impl configuration key. This configuration defines the implementation of org.apache.hadoop.net.CachedDNSToSwitchMapping.

As this configuration belongs to the Hadoop class, the configuration is exactly the same as the Hadoop configuration.

Static list

To configure the static list, use the TableMapping mentioned below:

```
<property>
   <name>net.topology.node.switch.mapping.impl</name>
   <value>org.apache.hadoop.net.TableMapping</value>
</property>
<property>
   <name>net.topology.table.file.name</name>
   <value>/opt/hadoop/compose/ozone-topology/network-config</value>
</property>
```

The net.topology.table.file.name property option must point to a text file. The file format is a two-column text file with columns separated by whitespace. The first column is the IP address and the second column specifies the rack where the address maps. If there is no entry for a host in the cluster, then /default-rack is assumed.

Dynamic list

To identify the Rack information, use the following external script:

```
<property>
   <name>net.topology.node.switch.mapping.impl</name>
   <value>org.apache.hadoop.net.ScriptBasedMapping</value>
</property>
<property>
   <name>net.topology.script.file.name</name>
   <value>/usr/local/bin/rack.sh</value>
</property>
```

If you are implementing an external script, the script is specified with the net.topology.script.file.name parameter in the configuration files. The external topology script is not included with the Ozone distribution and is provided by the administrator.Ozone sends multiple IP addresses to ARGV when forking the topology script.

Ozone sends multiple IP addresses to ARGV when forking the topology script. The number of IP addresses sent to the topology script is controlled with the net.topology.script.number.args property. The default value is 100. For example, if you change the net.topology.script.number.args property value to 1, then for each IP submitted, one topology script gets forked.

RATIS/THREE Data

The following write path, read path, and container replication applies only to Ozone RATIS/THREE data.

Write path

- You can configure the placement of open containers using the ozone.scm.pipeline.placement.impl configuration key.
- The pipeline placement policy is available in the org.apache.hdds.scm.pipeline package.
  - By default, the PipelinePlacementPolicy is used for topology awareness. Currently, this is the only pipeline placement policy implemented in Ozone.
  - To change to a user-customized implementation, use the following property

```
<property>
                  <name>ozone.scm.pipeline.placement.impl</name>
                  <value>full_class_name_of_the_customized_implementat
ion</value>
                  </property>
```

This placement policy complies with the algorithm used in HDFS. With the default three replicas, two replicas will be on the same rack and the third replica will be on a different rack.



Note:

- The PipelinePlacementPolicy policy applies to network topology like /rack/node.
- Cloudera recommends you to have a customized placement policy implementation if the network topology has more layers than /rack/node.

Read path

Configure the read path to read the data from the closest pipeline using the following property:

```
<property>
         <name>ozone.network.topology.aware.read</name>
         <value>true</value>
         </property>
```

Container replication

- You can configure the placement of closed three replica containers using the ozone.scm.container.placement.impl configuration key.
- The container placement policies are available in the org.apache.hdds.scm.container.placement.algorithms package.
  - By default, the SCMContainerPlacementRackAware is used for topology awareness. This placement policy complies with the algorithm used in HDFS. With the default three replicas, two replicas will be on the same rack and the third replica will be on a different rack.
  - To change the placement of closed three replica containers, using the property ozone.scm.container.placement.impl, following is an example

```
<property>
          <name>ozone.scm.container.placement.impl</name>
          <value>org.apache.hadoop.hdds.scm.container.placement.algorithm
s.SCMContainerPlacementCapacity
          </value>
          </property>
```



Note:

- The SCMContainerPlacementRackAware policy applies to network topology like /rack/node.
- Cloudera recommends you not use the SCMContainerPlacementRackAware policy if the network topology has more layers than /rack/node.

Erasure Coding data

The following write path, read path, and container replication applies to Ozone EC data.

Write path

- You can configure the placement of open EC containers using the ozone.scm.container.placement.ec.impl configuration key.
- The pipeline placement policy is available in the org.apache.hdds.scm.container.placement.algorithms package.
  - By default, the SCMContainerPlacementRackScatter is used for topology awareness. Currently, this is the only pipeline placement policy implemented for EC in Ozone.
  - To change to a user-customized implementation, use the following property

```
<property>
   <name>ozone.scm.container.placement.ec.impl</name>
   <value>full_class_name_of_the_customized_implementation</value>
</property>
```

This SCMContainerPlacementRackScatter placement policy will try to distribute the replicas of an EC container on datanodes on as many racks as possible. For example, if the EC policy used is RS-3-2-1024k, then this policy will try to distribute the 5(3+2) replicas of an EC container to 5 datanodes, each under a different rack, as much as possible.



Note:

- The SCMContainerPlacementRackScatter policy applies to network topology like /rack/node.
- Cloudera recommends you to have a customized placement policy implementation if the network topology has more layers than /rack/node.

Read path

For an EC container, each replica contains different pieces of data. Data is read as requested. There is no topology configuration here.

Container replication

Currently, closed EC containers' replication and balance use the same placement policy described in the Write Path section. That is, the property ozone.scm.container.placement.ec.impl with default implementation SCMContainerPlacementRackScatter applies to both open containers write and closed containers replication and balance.

Customizing Ozone network topology

Learn about how to customize the Ozone network topology.

About this task

By default, Ozone supports a three layer network topology. For example, "/rack/node", where "/", "rack", and "node" are three layers of the network topology. To customize the network topology, do the following steps:

Procedure

1. Define network topology schema file:
   a) Prepare an Ozone network topology schema XML file.

      Ozone's default network topology schema file is network-topology-default.xml. It consists of three layers which includes Root, an InnerNode Rack, and a LeafNode.

      To define a four layer network topology, see network-topology-nodegroup.xml. For example, the following sample schema file ozone-customized-network-topology.xml consists of four layers which includes ROOT, two InnerNodes dc and rack, and one Leaf node.

```
<?xml version="1.0"?>
<configuration>
      <layoutversion>1</layoutversion>
      <layers>
     <layer id="cluster">
           <prefix></prefix>
           <cost>1</cost>
           <type>Root</type>
     </layer>
     <layer id="dc">
           <prefix>dc</prefix>
           <cost>1</cost>
           <type>InnerNode</type>
           <default>/default-dc</default>
     </layer>
     <layer id="rack">
            <prefix>rack</prefix>
                   <cost>1</cost>
            <type>InnerNode</type>
            <default>/default-rack</default>
     </layer>
     <layer id="node">
            <prefix></prefix>
            <cost>0</cost>
            <type>Leaf</type>
     </layer>
      </layers>
      <topology>
     <path>/cluster/dc/rack/node</path>
     <enforceprefix>false</enforceprefix>
      </topology>
</configuration>
```

b) After the schema file is ready, copy it to every node in the cluster, under some directory, for example, /opt/cloudera.
c) Make sure that the SCM, Recon, and OM services have the permission to access the customized Ozone network topology schema file.

2. Update host's network path.

   To update the rack information of each node, see Editing rack assignments for hosts. For example, in the preceeding example four layer schema file, you can change the rack information from /rack1 to /dc1/rack1.
3. Set the ozone.scm.network.topology.schema.file property to the file path of the customized Ozone network topology schema file for the SCM, Recon, and OM services.
4. Save the changes.
5. Restart the SCM, Recon, and OM services.

Verifying Ozone network topology using the CLI

After configuring topology awareness and starting the cluster, run `ozone admin printTopology` command to verify that the Storage Container Manager (SCM) has applied the correct rack assignments to each DataNode. This check ensures no DataNode is incorrectly assigned to /default-rack due to mapping errors.

Command syntax

```
ozone admin printTopology [***OPTIONS***]
```

Command options

Table 7: ozone admin printTopology command options

Option	Description
-o or --order	Sorts and groups output by network location (rack path).
-f or --full	Includes full node information of the topology in the output.
--operational-state *[***STATE***]*	Filters DataNodes by operational state such as IN_SERVICE, DECOMMISSIONING, DECOMMISSIONED, ENTERING_MAINTENANCE or IN_MAINTENANCE.
--node-state *[***STATE***]*	Filters DataNodes by health state such as HEALTHY, STALE, or DEAD.
--scm *[***HOST:PORT***]*	Connects to a specific SCM instance (example, scm-host.example.com:9860).
-ID or --service-id *[***ID***]*	Specifies the service ID of the SCM HA cluster.
--json	Formats the command output as a JSON object. The JSON output creates an independent array for each node health state (HEALTHY, STALE, DEAD). Each array element is a JSON object representing an individual DataNode. The fields populated in each object depend on whether default formatting or additional flags (such as --full) are applied.

Output examples

Following are the output samples for the `ozone admin printTopology` with different options:

- Default output of the command: Without options, the command lists DataNodes grouped by health state such as HEALTHY, STALE, or DEAD. Each state with at least one DataNode displays a state header followed by one line per node. Each entry displays the IP address, hostname, port mapping, operational state, and rack location.

Command:

```
ozone admin printTopology
```

Sample output:

```
State = HEALTHY
172.18.0.12(ozone-ha-datanode-3.ozone-ha_default):HTTP=9882,CLIENT_RPC=1
9864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATI
S_DATASTREAM=9855,STANDALONE=9859    IN_SERVICE    /rack1
172.18.0.15(ozone-ha-datanode-1.ozone-ha_default):HTTP=9882,CLIENT_RPC=198
64,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATIS_
DATASTREAM=9855,STANDALONE=9859    DECOMMISSIONED    /rack1
172.18.0.14(ozone-ha-datanode-5.ozone-ha_default):HTTP=9882,CLIENT_RPC=1
9864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATI
S_DATASTREAM=9855,STANDALONE=9859    IN_SERVICE    /rack2
172.18.0.10(ozone-ha-datanode-2.ozone-ha_default):HTTP=9882,CLIENT_RPC=198
64,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATIS_
DATASTREAM=9855,STANDALONE=9859    IN_MAINTENANCE    /rack2
State = STALE
172.18.0.13(ozone-ha-datanode-4.ozone-ha_default):HTTP=9882,CLIENT_RPC=198
64,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATIS_
DATASTREAM=9855,STANDALONE=9859    IN_SERVICE    /rack2
```

- Ordered output (with the --order option): Groups DataNodes first by health state, then alphabetically by rack path. Each DataNode under a rack displays its IP address, hostname, and operational state.

  Command:

  ```
  ozone admin printTopology --order
  ```

  Sample output:

  ```
  State = HEALTHY
  Location: /rack1
   172.18.0.12(ozone-ha-datanode-3.ozone-ha_default) IN_SERVICE
   172.18.0.15(ozone-ha-datanode-1.ozone-ha_default) DECOMMISSIONED
  Location: /rack2
   172.18.0.14(ozone-ha-datanode-5.ozone-ha_default) IN_SERVICE
   172.18.0.10(ozone-ha-datanode-2.ozone-ha_default) IN_MAINTENANCE
  State = STALE
  Location: /rack2
   172.18.0.13(ozone-ha-datanode-4.ozone-ha_default) IN_SERVICE
  ```

- Detailed output (with the --full option): When you run the command with the --full option, the output expands to include the UUID, IP address, hostname, port mapping, operational state, and rack location for each DataNode, grouped by health state.

  Command:

  ```
  ozone admin printTopology --full
  ```

  Sample output:

  ```
  State = HEALTHY
  d669581f-5726-44c0-b73f-f3c2b8a82946/172.18.0.12(ozone-ha-datanode-3.ozo
  ne-ha_default):HTTP=9882,CLIENT_RPC=19864,REPLICATION=9886,RATIS=9858,RA
  TIS_ADMIN=9857,RATIS_SERVER=9856,RATIS_DATASTREAM=9855,STANDALONE=9859
   IN_SERVICE    /rack1
  d8773b49-5b76-480d-a745-738cc341917f/172.18.0.15(ozone-ha-datanode-1.ozon
  e-ha_default):HTTP=9882,CLIENT_RPC=19864,REPLICATION=9886,RATIS=9858,RAT
  ```

```
IS_ADMIN=9857,RATIS_SERVER=9856,RATIS_DATASTREAM=9855,STANDALONE=9859
 DECOMMISSIONED    /rack1
20c5a433-58c7-4722-b7ae-9d0d88986958/172.18.0.14(ozone-ha-datanode-5.ozone
-ha_default):HTTP=9882,CLIENT_RPC=19864,REPLICATION=9886,RATIS=9858,RATI
S_ADMIN=9857,RATIS_SERVER=9856,RATIS_DATASTREAM=9855,STANDALONE=9859
IN_SERVICE    /rack2
52549926-53e2-4e2c-b542-8d9a25e815d0/172.18.0.10(ozone-ha-datanode-2.oz
one-ha_default):HTTP=9882,CLIENT_RPC=19864,REPLICATION=9886,RATIS=9858,R
ATIS_ADMIN=9857,RATIS_SERVER=9856,RATIS_DATASTREAM=9855,STANDALONE=9859
 IN_MAINTENANCE    /rack2
State = STALE
2dc9a4d1-64d0-4615-bc2e-36ea661af0e0/172.18.0.13(ozone-ha-datanode-4.ozone
-ha_default):HTTP=9882,CLIENT_RPC=19864,REPLICATION=9886,RATIS=9858,RATI
S_ADMIN=9857,RATIS_SERVER=9856,RATIS_DATASTREAM=9855,STANDALONE=9859
IN_SERVICE    /rack2
```

Important: If --full and --order options are used simultaneously, --orderoption takes precedence and overrides the full node output.

- Filtering output: Filter the topology display by combining or using --node-state and --operational-state options independently.
  - **Command**:

    ```
    ozone admin printTopology --node-state HEALTHY
    ```

    Sample output:

    ```
    State = HEALTHY
    172.18.0.12(ozone-ha-datanode-3.ozone-ha_default):HTTP=9882,CLIENT_RPC=1
    9864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATI
    S_DATASTREAM=9855,STANDALONE=9859    IN_SERVICE    /rack1
    172.18.0.15(ozone-ha-datanode-1.ozone-ha_default):HTTP=9882,CLIENT_RPC
    =19864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RA
    TIS_DATASTREAM=9855,STANDALONE=9859    DECOMMISSIONED    /rack1
    172.18.0.14(ozone-ha-datanode-5.ozone-ha_default):HTTP=9882,CLIENT_RPC=1
    9864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATI
    S_DATASTREAM=9855,STANDALONE=9859    IN_SERVICE    /rack2
    172.18.0.10(ozone-ha-datanode-2.ozone-ha_default):HTTP=9882,CLIENT_RPC
    =19864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RA
    TIS_DATASTREAM=9855,STANDALONE=9859    IN_MAINTENANCE    /rack2
    State = STALE
    ```

  - **Command**:

    ```
    ozone admin printTopology --operational-state DECOMMISSIONED
    ```

    Sample output:

    ```
    State = HEALTHY
    172.18.0.15(ozone-ha-datanode-1.ozone-ha_default):HTTP=9882,CLIENT_RPC=1
    9864,REPLICATION=9886,RATIS=9858,RATIS_ADMIN=9857,RATIS_SERVER=9856,RATI
    S_DATASTREAM=9855,STANDALONE=9859    DECOMMISSIONED    /rack1
    State = STALE
    ```

Note: If the cluster contains nodes in other health states, their State = <HEALTH_STATE> headers may still appear without any DataNodes listed beneath them, even if those states do not match the applied filter.

Ozone Placement Policy

Ozone uses Placement Policies to decide how to distribute Ratis and Erasure Coded containers among DataNodes. This document provides an overview of the available policies and describes how to configure them.

Uses of Placement Policy:

1. Ozone creates pipelines of DataNodes for Ratis or Erasure Coded "Open" containers. Ozone selects DataNodes based on load balancing and network topology.
2. Ozone selects the DataNodes that must get the replicas of a "Closed" Ratis or Erasure Coded container. This logic is used when resolving under replication or over replication of a container, for example. This is different from the above point because Closed containers are not part of write pipelines. You can configure this using ozone.scm.container.placement.impl for Ratis containers.

This document discusses the placement policies available for configuring ozone.scm.container.placement.impl.

Placement Policy for Ratis Containers

The default placement policy for Ratis containers is SCMContainerPlacementRackAware.



Note: Cloudera recommends you use only one placement policy. Switching placement policies can cause containers to be mis-replicated and moved among DataNodes. The size of data that will move cannot be predicted.

SCMContainerPlacementRackAware

This policy requires container replicas to be present on at least two racks or one rack if only one is available. It provides rack failure tolerance. If the replication factor is three, this policy places two replicas on one rack and the third on a different rack. It is also valid for all three replicas to be present on different racks. Cloudera recommends this policy when you are using a network topology with racks.



Warning: The rack failure tolerance on a cluster is not guaranteed if this placement policy is not used.

SCMContainerPlacementCapacity

If the cluster’s network topology is not using multiple racks, Cloudera recommends you use SCMContainerPlacementCapacity. This placement policy leads to an I/O pattern where the lower utilized nodes are favoured more than the higher utilized nodes for placing containers. However, a part of the I/O will still go to the higher-utilized nodes.

SCMContainerPlacementRandom

This placement policy randomly selects healthy DataNodes without considering racks or node utilizations.

Procedure to configure for Ratis Containers

1. Log in to Cloudera Manager.
2. Navigate to Clusters.
3. Select the Ozone service.
4. Go to Configurations.
5. Search for Storage Container Manager Advanced Configuration Snippet (Safety Valve) for ozone-conf/ozone-site.xml.
6. Define the Placement Policy for ozone.scm.container.placement.impl. For example, org.apache.hadoop.hdds.scm.container.placement.algorithms.SCMContainerPlacementRackAware.

7. Click Save.

Placement Policy for Erasure Coded Containers

The default placement policy for Erasure Coded (EC) containers is SCMContainerPlacementRackScatter. This is currently the only supported Placement Policy for Erasure Coded containers. This tries to spread container replicas across as many racks as possible.

For an EC 3-2 container, there will be one replica per rack, for a total of five racks. However, if less than five racks are present in the cluster, then it is still valid to spread out the container replicas among the available racks. For example, if only two racks are available, then one rack will hold three replicas and the other will hold two.

Ozone volume scanner

The Ozone Volume Scanner feature enables to detect any disk failures on the DataNodes. Learn how you can configure the frequency of volume scans that can detect disk failures and how to handle volume failures.

The volume scanner scans each data volume configured by hdds.datanode.dir and each metadata volume configured by dfs.container.ratis.datanode.storage.dir.

Various events in the DataNode can trigger volume scans. Each volume scan consists of multiple checks.



Note: If you want to change the default values, you must change them through Cloudera Manager safety valve.

Background volume scan

Datanodes scan every volume once per hour. The frequency of this check is configured with hdds.datanode.periodic.disk.check.interval.minutes. This property defines the minimum frequency of scans for a volume.

On-demand volume scan

Any error reading from or writing to a volume during regular datanode operation triggers a scan of that volume. To prevent frequent scanning of the same volume, the hdds.datanode.disk.check.min.gap configuration, which defaults to 10 minutes, specifies the minimum time to wait between two consecutive scans of the same volume.

- Directory check

  This checks that each directory configured in hdds.datanode.dir and dfs.container.ratis.datanode.storage.dir exists and has read, write, and execute permissions by the datanode process. If this check fails, the volume is marked as failed.
- I/O check

  This checks that the underlying disk is present and functioning properly. The I/O check writes data to a small temporary file, synchronizes it to ensure it touches the hardware, reads the data, and then deletes the file. To account for intermittent errors, this check must fail multiple times before the volume is failed. The specifics of this check can be changed with the following configurations:

  - hdds.datanode.disk.check.io.file.size

    The size in bytes of the file to write for disk checking. During the check, the content of this file is stored in memory. The default value is 100 bytes.
  - hdds.datanode.disk.check.io.test.count

    The number of volume scan results in determining if the volume should be failed based on the I/O failures. The default value is 3.
  - hdds.datanode.disk.check.io.failures.tolerated

The number of I/O failures that can occur out of the last hdds.datanode.disk.check.io.test.count scans without the volume marked unhealthy. The default value is 1.

An example of using the default values:

Consider the I/O check passed on two out of the last three volume scans but failed on the latest volume scan. The volume remains healthy because one out of the last three I/O checks failed and hdds.datanode.disk.check.io.failures.tolerated is set to 1. If a fourth volume scan is run and the I/O check fails, the volume is failed, because out of the last three volume scans two failed due to I/O checks.

- Time check

  Both of the above checks must finish within a certain amount of time. Otherwise, the volume fails. This time limit is configured with the hdds.datanode.disk.check.timeout parameter. The default value is 10 minutes.

Handling Volume Failures

When a volume is marked failed, Ozone no longer uses it and triggers replication of the data from existing copies on other datanodes. After the issue on the failed volume is corrected, restart the datanode to detect the new volume.

Datanodes continue to run until they have no healthy data volumes or metadata volumes remaining. If you want the datanodes to shut down after a specified number of volume failures, set hdds.datanode.failed.data.volumes.tolerated or hdds.datanode.failed.metadata.volumes.tolerated to a positive number. If the set number of volume failures is crossed, the datanode shuts down automatically.

Ozone OMDBInsights

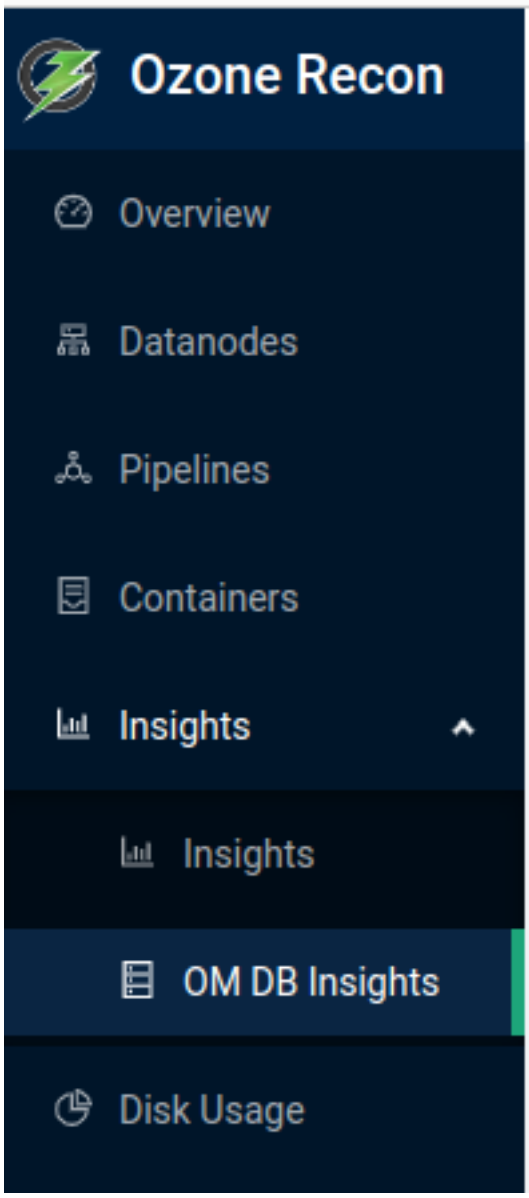
The Ozone Manager Database Insights feature helps you view the container mismatch information, open keys, keys pending for deletion, and deleted container keys.

Accessing Recon Web UI

To access Ozone Recon Web UI, perform the following steps.

Procedure

1. Log in to Cloudera Manager.
2. Navigate to Clusters.
3. Select the Ozone service.
4. Click Recon Web UI.
5. On the Ozone Recon left navigation pane, click Insights



6. Click OM DB Insights

OMDBInsights

The Ozone Manager Database Insights feature helps you view the container mismatch information, open keys, keys pending for deletion, and deleted container keys. These are accessible to administrators and helpful for diagnostic purposes in running clusters.

There are four tabs available:

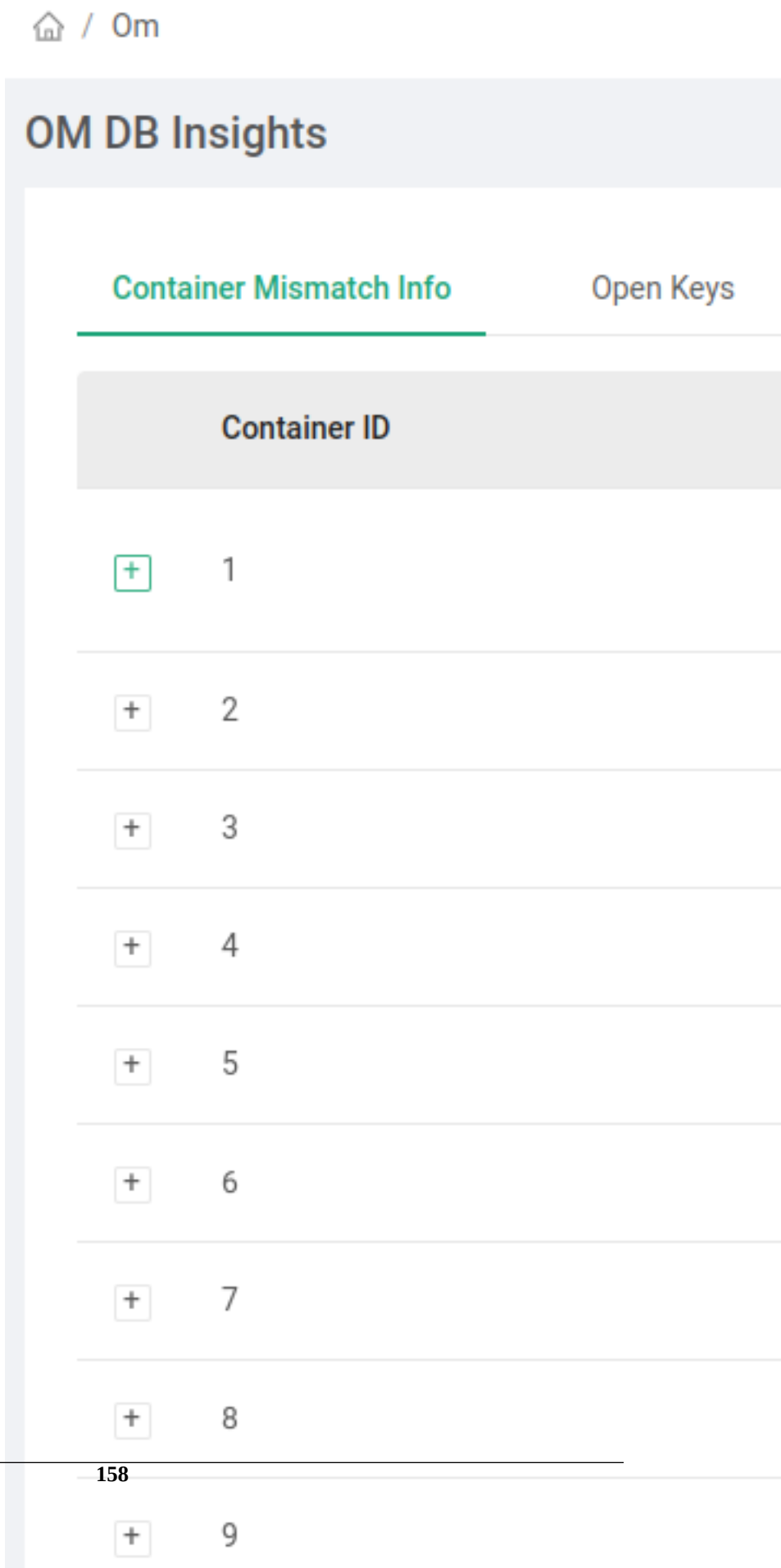
1. Container Mismatch Info
2. Open Keys
3. Keys Pending for Deletion
4. Deleted Container keys

Container Mismatch Info

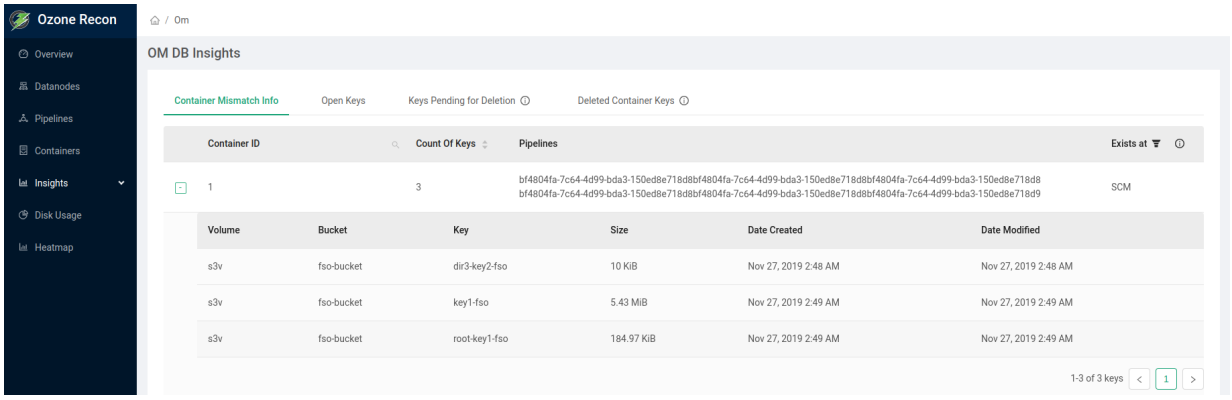
- This tab displays container-level information showing a mismatch between SCM and OM.

- If any container is deleted in SCM but referred by files and keys in OM and vice versa, you can use the Exists at filter to view such information.
  1. Exists at OM: Container is present in OM but not in SCM. There is data loss.
  2. Exists at SCM: Container is present in SCM but not in OM.

- If the container is present in SCM but not in OM, the API path is /containers/mismatch?



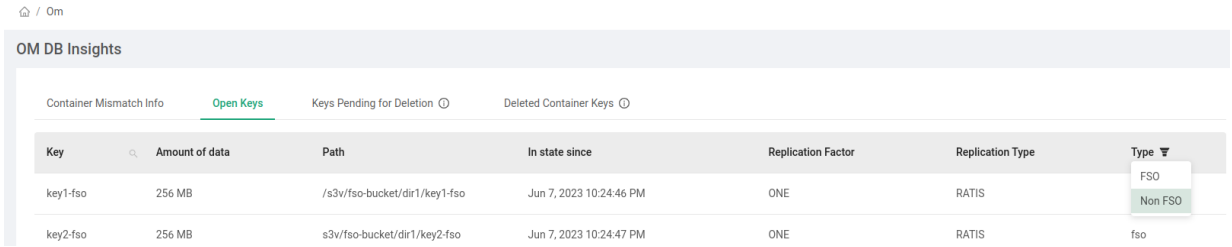
- The Recon Web UI then queries /api/v1/containers/1/keys to get details of keys mapped to such containers and this information is displayed in the expanded table of each row.



Open Keys

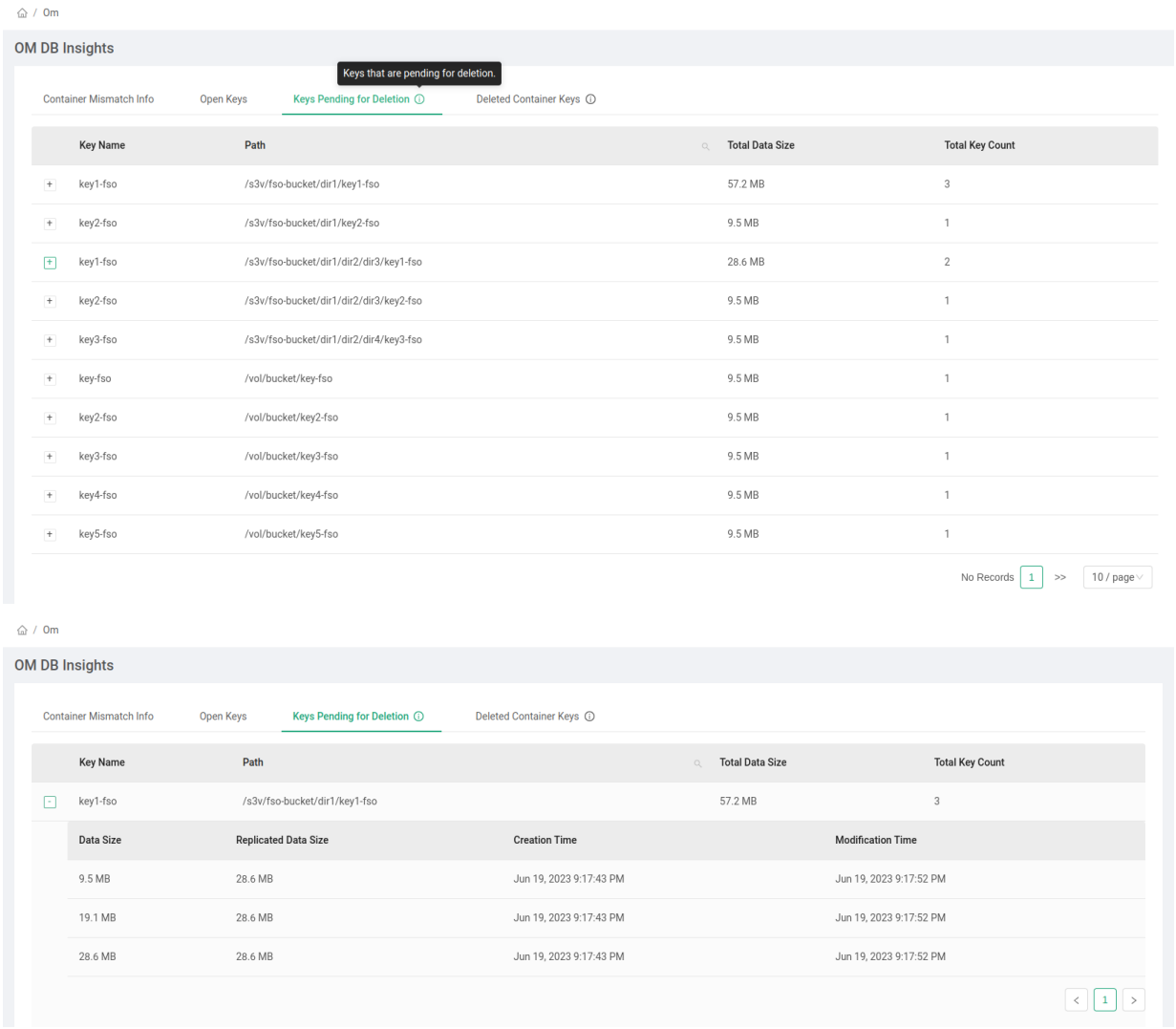
- This tab displays key-level information showing open keys and the amount of data mapped to open keys.
- The Type filter allows you to filter the information for FSO and Non-FSO keys.
- Based on the Type filter selection, you will have either the FSO or Non FSO open keys displayed in the UI.

1. If you select FSO from the Type filter, then the following API gets called and all FSO Open keys are displayed on the Recon UI. /api/v1/keys/open?includeFso=true&includeNonFso=false&limit=10&prevKey
2. If you select Non FSO from the Type filter, then the following API gets called and all Non FSO Open keys are displayed on the Recon UI. /api/v1/keys/open?includeFso=false&includeNonFso=true&limit=10&prevKey



Keys Pending for deletion

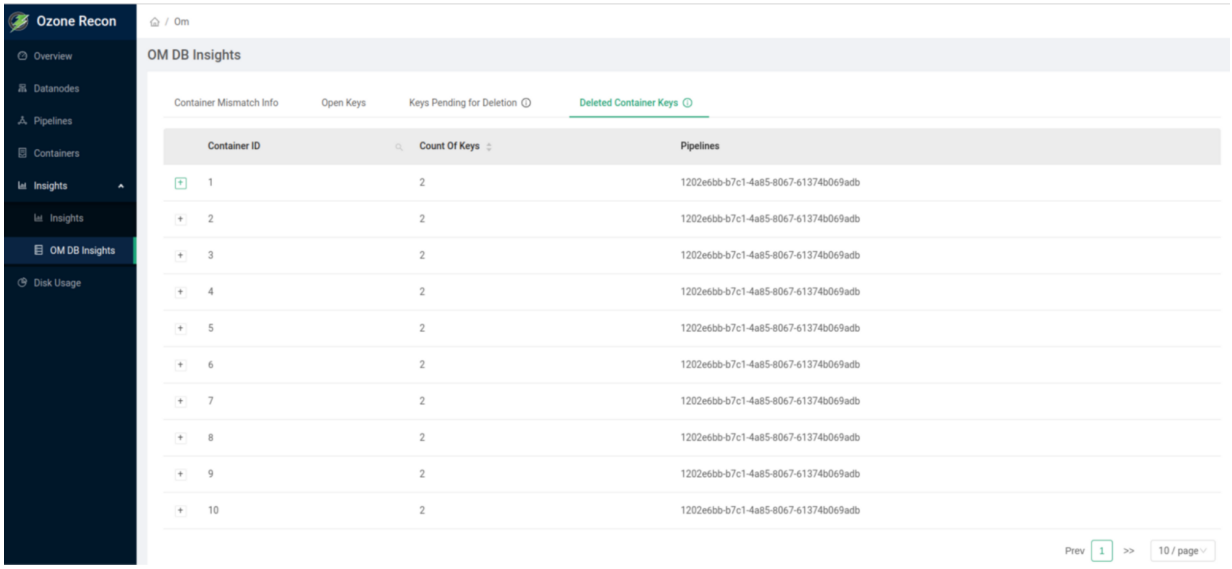
- This tab displays Keys that are pending for deletion.
- End Point /api/v1/keys/deletePending?limit=10&prevKey

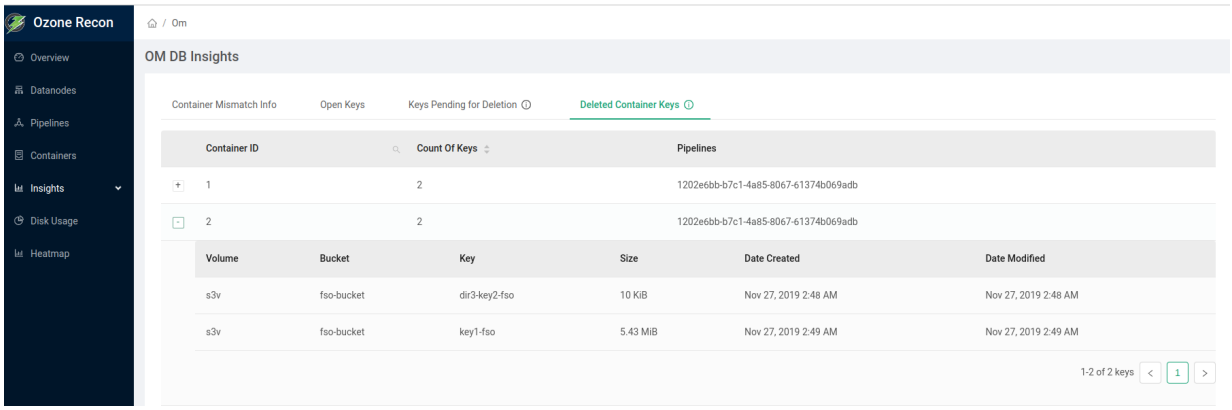


- Multiple keys can have the same name but different sizes, creation time, and modification times. For example, there can be two objects of key information present in the omKeyInfo list and these keys can have the same name (key1-fso)and the same Path (/s3v/fso-bucket/dir1/key1-fso) but have different sizes like 100 bytes and 200 bytes. Since the omKeyInfoList contains two objects with the same keyName and path, the dataSize attribute of all objects in the omKeyInfoList calculates to TotalDataSize=300 Bytes.

Deleted Container Keys

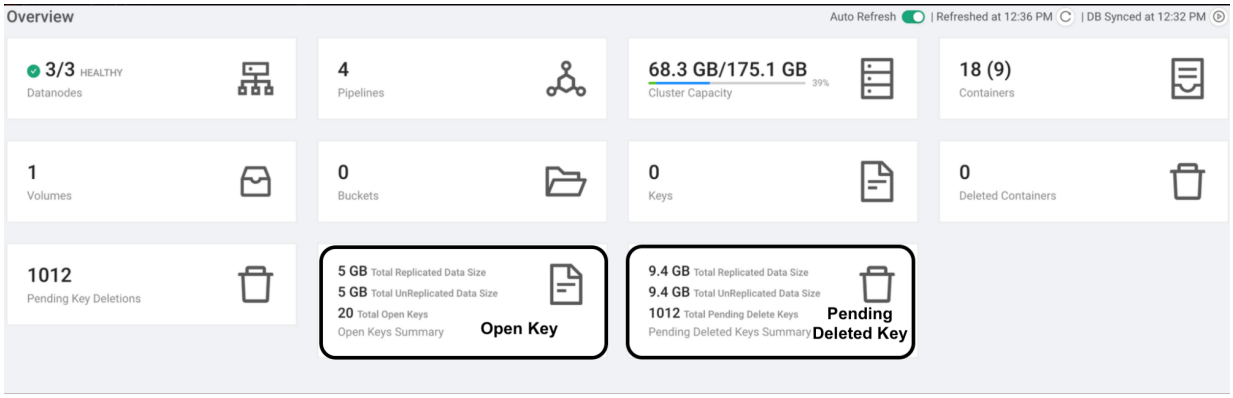
- This tab displays the information of keys that are mapped to containers in the DELETED state in SCM
- End Point /api/v1/containers/mismatch/deleted?limit=10&prevKey=0





Summary APIs

- The endpoint for Open keys Summary API is /api/v1/keys/open?limit=0. This API helps you to know the replicated and unreplicated data size. The total Open key count is also displayed on the UI.
- The endpoint for Pending Deleted Keys Summary API is /api/v1/keys/deletePending?limit=1. This API helps you to know the replicated and unreplicated data size. The total Pending deleted key count is also displayed on the UI.



Cluster Capacity

The Cluster Capacity dashboard in Ozone Recon provides a centralized view of storage utilization across your entire cluster. Moving from high-level physical capacity to individual DataNode diagnostics, it tracks usable space, logical data distribution, pre-allocated storage, and pending deletion lifecycles to help you monitor storage health and balance capacity effectively.

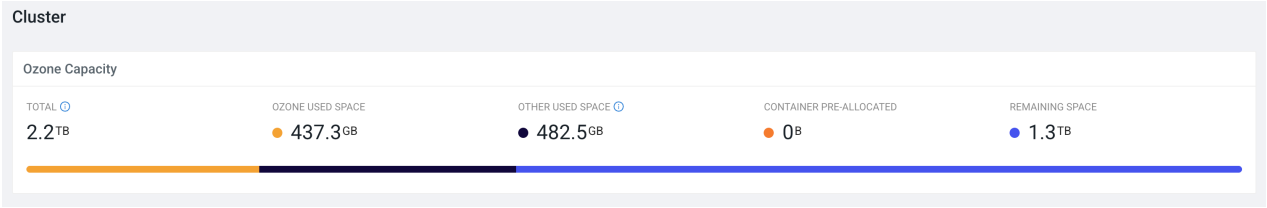
The **Cluster Capacity** page is organized from top to bottom, moving from broad cluster metrics to granular node details:

- **Cluster summary**: Provides overall physical storage capacity details for the Ozone cluster.
- **Service summary**: Displays logical storage usage and details how Ozone space is consumed.
- **Pending deletion lifecycle**: Tracks how pending deletions move across the Ozone Manager (OM), Storage Container Manager (SCM), and DataNodes.
- **Datanode insights**: Offers detailed metrics on individual DataNode storage distribution and performance.

Ozone cluster (physical) capacity

The **Cluster** widget provides a high-level summary of the total physical storage managed by Ozone DataNodes. It helps you distinguish between space used by Ozone versus space consumed by other processes on the underlying hardware.

Figure 7: Ozone cluster physcial capacity



The **Cluster** widget displays the following metrics:

- **Total capacity**: The combined usable storage space across all configured directories on live DataNodes, calculated after subtracting reserved file system space.
- **Ozone used space**: The physical disk space currently occupied by replicated Ozone blocks.

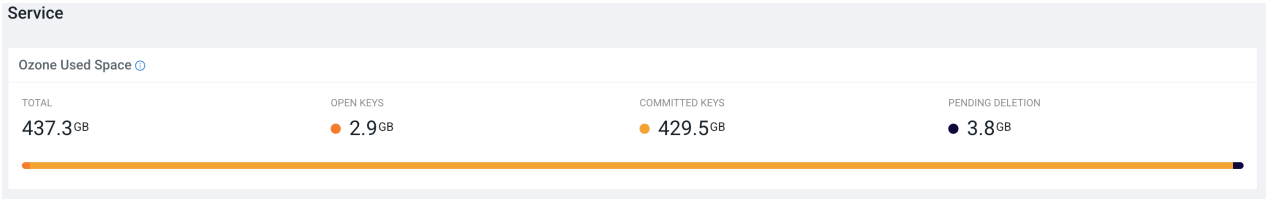
  Note: This Ozone used space accounts for your replication factor configuration. For example, a 100 GB key with 3x replication consumes 300 GB of physical space.

- **Other used space**: Storage consumed by internal Ozone system files rather than actual user data. This includes logs, configuration files, RocksDB files and so on.
- **Container pre-allocated**: Space reserved temporarily for open containers that have been allocated to clients but not yet fully written to. This reserved space decreases as data is written, and any remaining unused space is released when the container is closed.
- **Remaining space**: The actual amount of unused physical disk space currently available for new Ozone data.

Service capacity

The **Service** widget transitions from the physical hardware view to a logical architecture view. It breaks down the Ozone Used Space based on the current state of the keys within the system.

Figure 8: Ozone service capacity



This **Service** widget breaks down total Ozone used space across the following key states:

- **Total**: The sum of all Ozone data currently tracked in the system across all states. This exactly matches the physical OZONE USED SPACE in the **Cluster** widget.
- **Open keys**: Temporary data in keys that are currently being written to by clients or have not yet been committed to the system.
- **Committed keys**: Finalized, immutable data successfully stored and accessible by users.
- **Pending deletion**: Data from keys logically deleted by users but not yet physically scrubbed from the DataNodes. This is the combined total size of data pending deletion across OM, SCM, and Datanodes. This space will eventually be reclaimed.

Pending deletions

Because Ozone handles deletion asynchronously, this **Pending Deletion** widget provides transparency into the multi-stage data deletion pipeline. It tracks how deletion requests flow from the Ozone Manager (OM) down to physical removal on the DataNodes.

Figure 9: Pending deletions



The **Pending Deletion** widget breaks down pending data deletions across three system components:

- **Ozone Manager (OM)**: Keys or directories deleted by the client, but whose underlying blocks have not yet been processed by the Storage Container Manager (SCM).
- **Storage Container Manager**: Blocks that SCM has identified as ready for deletion and is actively commanding DataNodes to remove.
- **DataNodes**: Blocks currently queued on individual DataNodes waiting for physical disk deletion.

Important:

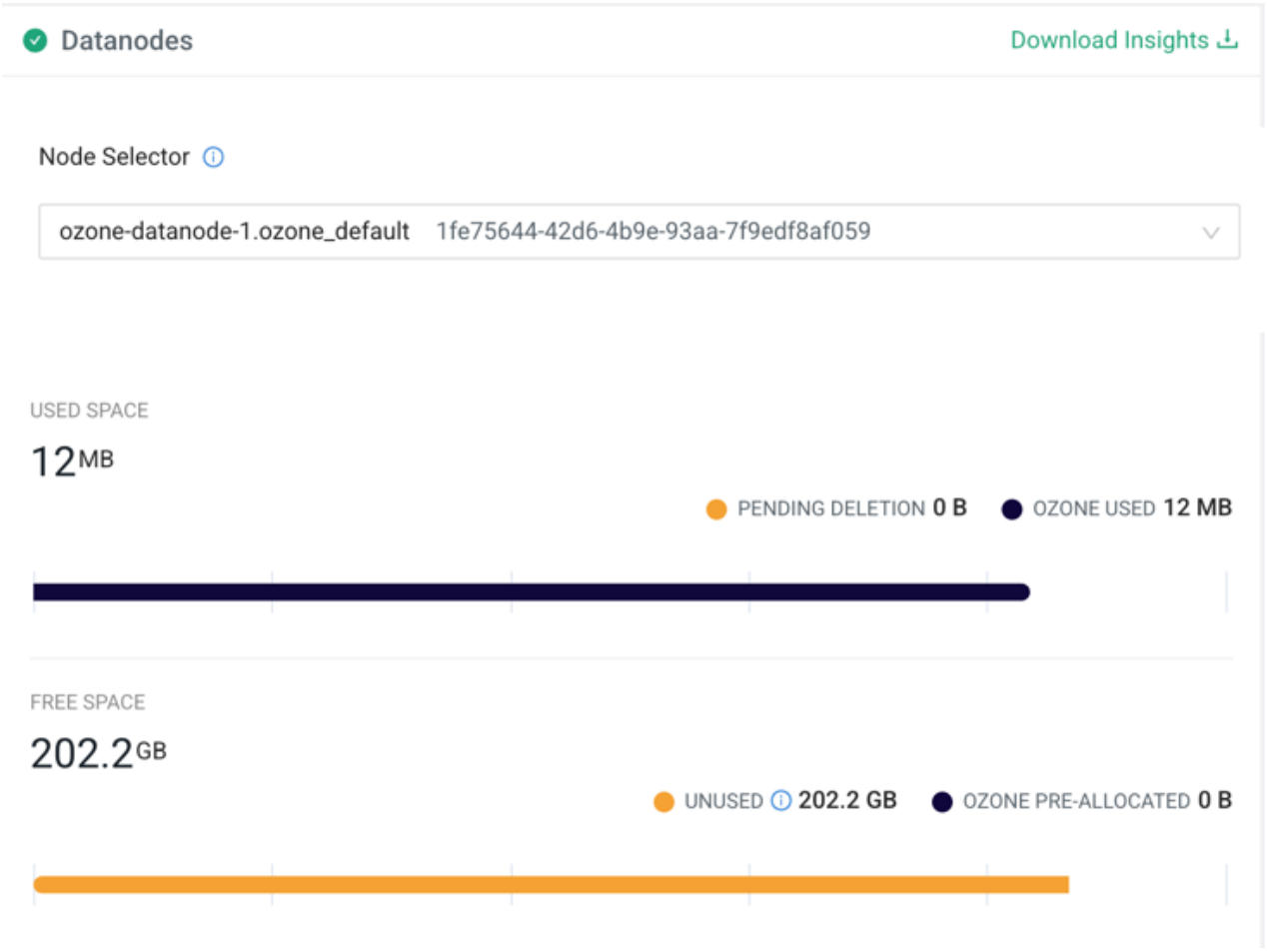
- If active database tracking shows a large variance from reported disk usage, or you notice a persistently high Pending Deletion value, it typically points to un-reclaimed deleted block backlogs or delayed asynchronous deletion pipelines.

  Example: If SCM shows a large amount of pending deletion (example, 1 TB) but the DataNodes stage shows 0 B, SCM may be failing to communicate deletion commands to the nodes.
- Ozone processes deletions asynchronously, which can cause temporary overlaps in pending deletion metrics across distributed components:
  - Ozone Manager (OM) to Storage Container Manager (SCM): When OM deletes data (example, 10 GB), it issues a block-deletion request to SCM. During initial processing, both OM and SCM may simultaneously report 10 GB pending deletion. After OM updates its state, OM reports 0 GB while SCM continues to report 10 GB until execution completes.
  - SCM to DataNodes: A similar propagation delay occurs while SCM dispatches block-deletion commands to the underlying DataNodes.

Datanode insights

The **Datanodes** widget shifts focus to individual node performance. This is crucial for identifying storage imbalances, failing disks, or nodes that are filling up faster than their peers.

Figure 10: Datanodes insights



Using the Datanode Inspector:

- Node Selector: Pick a specific Datanode using the searchable dropdown, which displays both the hostname and the unique UUID (example, ozone-datanode-1.ozone_defa).
- Download Insights: Export a snapshot report of all DataNode storage distributions in CSV format.

Once a Datanode is selected, you can view its specific storage distribution:

- **Used space (Datanode level)**: Total storage distribution on the selected node.
- **Pending deletion**: Space occupied by blocks queued for immediate physical deletion on this node.
- **Ozone used**: Physical space used by active, replicated Ozone blocks on this specific node.
- **Ozone pre-allocated**: Space reserved specifically for open Ozone containers on this node.
- **Free or unused space**: Remaining unused physical capacity available on this node.

Important: Toggle between multiple nodes using the Node Selector. Large discrepancies in Ozone Used space across different nodes can be a strong indicator of data balancing issues within your cluster.