

## Storing Data Using Ozone

Date published: 2020-04-24

Date modified: 2021-08-05



# Legal Notice

© Cloudera Inc. 2024. 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

|   |           |
|---|-----------|
| <b>Managing storage elements by using the command-line interface.....</b> | <b>5</b>  |
| Commands for managing volumes.....  | 5         |
| Assigning administrator privileges to users.....                          | 7         |
| Commands for managing buckets.....  | 7         |
| Commands for managing keys.....   | 9         |
| <b>Using Ozone S3 Gateway to work with storage elements.....</b>          | <b>11</b> |
| Configuration to expose buckets under non-default volumes.....            | 11        |
| REST endpoints supported on Ozone S3 Gateway.....                         | 11        |
| Configuring Ozone to work as a pure object store.....                     | 12        |
| Access Ozone S3 Gateway using the S3A filesystem.....                     | 12        |
| Examples of using the S3A filesystem with Ozone S3 Gateway.....           | 13        |
| Configuring Spark access for S3A.....                                     | 14        |
| Configuring Hive access for S3A.....                                      | 15        |
| Configuring Impala access for S3A.....                                    | 16        |
| Using the AWS CLI with Ozone S3 Gateway.....                              | 17        |
| Configuring https endpoints in Ozone S3 Gateway to work with AWS CLI..... | 17        |
| Examples of using the AWS CLI for Ozone S3 Gateway.....                   | 18        |
| <b>Accessing Ozone object store with Amazon Boto3 client.....</b>         | <b>20</b> |
| Obtaining resources to Ozone.....   | 20        |
| Obtaining client to Ozone through session.....                            | 20        |
| List of APIs verified.....  | 21        |
| Create a bucket.....  | 21        |
| List buckets.....   | 21        |
| Head a bucket.....  | 21        |
| Delete a bucket.....  | 21        |
| Upload a file.....  | 22        |
| Download a file.....  | 22        |
| Head an object.....   | 22        |
| Delete Objects.....   | 22        |
| Multipart upload.....   | 22        |
| <b>Working with Ozone File System (o3fs).....</b>                         | <b>23</b> |
| Setting up o3fs.....  | 23        |
| <b>Working with ofs.....</b>  | <b>24</b> |
| Volume and bucket management using ofs.....                               | 24        |
| Key management using ofs.....   | 26        |
| <b>Ozone configuration options to work with CDP components.....</b>       | <b>26</b> |
| Configuration options for Spark to work with o3fs.....                    | 26        |
| Configuration options to store Hive managed tables on Ozone.....          | 27        |

|  |           |
|--|-----------|
| <b>Overview of the Ozone Manager in High Availability.....</b>                     | <b>27</b> |
| Considerations for configuring High Availability on the Ozone Manager.....         | 27        |
| Ozone Manager nodes in High Availability.....                                      | 28        |
| Read and write requests with Ozone Manager in High Availability.....               | 28        |
| <b>Overview of Storage Container Manager in High Availability.....</b>             | <b>28</b> |
| Considerations for configuring High Availability on Storage Container Manager..... | 29        |
| Storage Container Manager operations in High Availability.....                     | 29        |
| <b>Offloading Application Logs to Ozone.....</b>                                   | <b>30</b> |
| <b>Removing Ozone DataNodes from the cluster.....</b>                              | <b>30</b> |
| Decommissioning Ozone DataNodes.....   | 31        |
| Placing Ozone DataNodes in offline mode.....                                       | 31        |
| Configuring the number of storage container copies for a DataNode.....             | 32        |
| Recommissioning an Ozone DataNode.....   | 32        |
| <b>Multi-Raft configuration for efficient write performances.....</b>              | <b>32</b> |
| <b>Working with the Recon web user interface.....</b>                              | <b>33</b> |
| Access the Recon web user interface.....   | 33        |
| Elements of the Recon web user interface.....                                      | 34        |
| Overview page.....   | 34        |
| DataNodes page.....  | 34        |
| Pipelines page.....  | 35        |
| Missing Containers page.....   | 36        |
| <b>Configuring Ozone to work with Prometheus.....</b>                              | <b>37</b> |
| <b>Ozone trash overview.....</b>   | <b>38</b> |
| <b>Configuring the Ozone trash checkpoint values.....</b>                          | <b>38</b> |

# 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 <https://hadoop.apache.org/ozone/docs/1.0.0/interface/cli.html>.

## 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 | <pre>ozone sh volume create --quota=&lt;volume capacity&gt; --user=&lt;username&gt; URI</pre>  |
| Purpose        | Creates a volume and assigns it to a user.   |
| Arguments      | <ul style="list-style-type: none"> <li>-q, quota: Specifies the maximum size the volume can occupy in the cluster. This is an optional parameter.</li> <li>-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.</li> <li>URI: The name of the volume to create in the &lt;prefix&gt;://&lt;Service ID&gt;/&lt;volumename&gt; format.</li> </ul> |
| Example        | <pre>ozone sh volume create --quota=2TB - -user=usr1 o3://ozone1/vol1</pre> <p>This command creates a 2-TB volume named vol1 for user usr1. Here, ozone1 is the Ozone Service ID.</p>  |

### Deleting a volume

|                |  |
|----------------|--|
| Command Syntax | <pre>ozone sh volume delete URI</pre>              |
| Purpose        | Deletes the specified volume, which must be empty. |

|           |   |
|-----------|---|
| Arguments | URI: The name of the volume to delete in the <prefix>://<Service ID>/<volumename> format.   |
| Example   | <pre>ozone sh volume delete o3://ozone1/vol2</pre> <p>This command deletes the empty volume vol2. Here, ozone1 is the Ozone Service ID.</p> |

### Viewing volume information

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

### Listing volumes

|                |  |
|----------------|--|
| Command Syntax | <pre>ozone sh volume list --user &lt;username&gt; URI</pre>  |
| Purpose        | Lists all the volumes owned by the specified user.   |
| Arguments      | <ul style="list-style-type: none"> <li>-u, user: The name of the user whose volumes you want to list.</li> <li>URI: The Service ID of the cluster in the &lt;prefix&gt;://&lt;Service ID&gt;/ format.</li> </ul> |
| Example        | <pre>ozone sh volume list --user usr2 o3://ozone1/</pre> <p>This command lists the volumes owned by user usr2. Here, ozone1 is the Ozone Service ID.</p>   |

### Updating a volume

|                |  |
|----------------|--|
| Command Syntax | <pre>ozone sh volume setquota --namespace-quota &lt;namespacecapacity&gt; --space-quota &lt;volumecapacity&gt; URI</pre> |
| Purpose        | Updates the quota of the specific volume.  |

|           |  |
|-----------|--|
| Arguments | <ul style="list-style-type: none"> <li><code>--namespace-quota &lt;namespacecapacity&gt;</code>: Specifies the maximum number of buckets this volume can have.</li> <li><code>--space-quota &lt;volumecapacity&gt;</code>: Specifies the maximum size the volume can occupy in the cluster.</li> <li>URI: The name of the volume to update in the <code>&lt;prefix&gt;://&lt;Service ID&gt;/&lt;volumename&gt;</code> format.</li> </ul> |
| Example   | <pre>ozone sh volume setquota --namespace-quota 1000 --space-quota 10GB /volume1</pre> <p>This command sets <i>volume1</i> namespace quota to 1000 and space quota to 10GB.</p>  |

## 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. On Cloudera Manager, go to the Ozone service.
2. Click the Configuration tab.
3. 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.

4. 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 | <pre>ozone sh bucket create URI</pre>     |
| 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   | <pre>ozone sh bucket create o3://ozone1/vol1/buck1</pre> <p>This command creates a bucket buck1 in the volume vol1. Here, ozone1 is the Ozone Service ID.</p> |

### Deleting a bucket

|                |  |
|----------------|--|
| Command Syntax | <pre>ozone sh bucket delete URI</pre>  |
| Purpose        | Deletes the specified bucket, which must be empty.   |
| Arguments      | URI: The name of the bucket to delete in the <prefix>://<Service ID>/<volumename>/<bucketname> format.   |
| Example        | <pre>ozone sh bucket delete o3://ozone1/vol1/buck2</pre> <p>This command deletes the empty bucket buck2. Here, ozone1 is the Ozone Service ID.</p> |

### Viewing bucket information

|                |   |
|----------------|---|
| Command Syntax | <pre>ozone sh bucket info URI</pre>   |
| 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        | <pre>ozone sh bucket info o3://ozone1/vol1/buck3</pre> <p>This command provides information about bucket buck3. Here, ozone1 is the Ozone Service ID.</p> |

### Listing buckets

|                |   |
|----------------|---|
| Command Syntax | <pre>ozone sh bucket list URI --length=&lt;number_of_buckets&gt; --prefix=&lt;bucket_prefix&gt; --start=&lt;starting_bucket&gt;</pre> |
| Purpose        | Lists all the buckets in a specified volume.  |

|           |  |
|-----------|--|
| Arguments | <ul style="list-style-type: none"> <li>-l, length: Specifies the maximum number of results to return. The default is 100.</li> <li>-p, prefix: Lists bucket names that match the specified prefix.</li> <li>-s, start: Returns results starting with the bucket <i>after</i> the specified value.</li> <li>URI: The name of the volume whose buckets you want to list, in the &lt;prefix&gt;://&lt;Service ID&gt;/&lt;volumename&gt;/ format.</li> </ul> |
| Example   | <pre>ozone sh bucket list o3://ozone1/vol2 --length=100 --prefix=buck --start=buck</pre> <p>This command lists 100 buckets belonging to volume vol2 and names starting with the prefix buck. Here, ozone1 is the Ozone Service ID.</p>   |

## 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 | <pre>ozone sh key get URI &lt;local_file name&gt;</pre>   |
| Purpose        | Downloads the specified key from a bucket in the Ozone cluster to the local file system.  |
| Arguments      | <ul style="list-style-type: none"> <li>URI: The name of the key to download in the &lt;prefix&gt;://&lt;Service ID&gt;/&lt;volumename&gt;/&lt;bucketname&gt;/&lt;keyname&gt; format.</li> <li>filename: The name of the file to which you want to write the key.</li> </ul> |
| Example        | <pre>ozone sh key get o3://ozone1/hive/jun/sales.orc sales_jun.orc</pre> <p>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.</p>            |

### Uploading a key to a bucket

|                |   |
|----------------|---|
| Command Syntax | <pre>ozone sh key put URI &lt;filename&gt;</pre>  |
| Purpose        | Uploads a file from the local file system to the specified bucket in the Ozone cluster. |

|           |  |
|-----------|--|
| Arguments | <ul style="list-style-type: none"> <li>URI: The name of the key to upload in the &lt;prefix&gt;://&lt;Service ID&gt;/&lt;volumename&gt;/&lt;bucketname&gt;/&lt;keyname&gt; format.</li> <li>filename: The name of the local file that you want to upload.</li> <li>-r, --replication: The number of copies of the file that you want to upload.</li> </ul> |
| Example   | <pre>ozone sh key put o3://ozonel/hive/year/sales.orc sales_corrected.orc</pre> <p>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, ozonel is the Ozone Service ID.</p>  |

### Deleting a key

|                |  |
|----------------|--|
| Command Syntax | <pre>ozone sh key delete URI</pre>   |
| 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        | <pre>ozone sh key delete o3://ozonel/hive/jun/sales_duplicate.orc</pre> <p>This command deletes the sales_duplicate.orc key. Here, ozonel is the Ozone Service ID.</p> |

### Viewing key information

|                |   |
|----------------|---|
| Command Syntax | <pre>ozone sh key info URI</pre>  |
| Purpose        | Provides information about the specified key.   |
| Arguments      | URI: The name of the key whose details you want to view, in the <prefix>://<Service ID>/<volumename>/<bucketname>/<keyname> format.   |
| Example        | <pre>ozone sh key info o3://ozonel/hive/jun/sales_jun.orc</pre> <p>This command provides information about the sales_jun.orc key. Here, ozonel is the Ozone Service ID.</p> |

### Listing keys

|                |   |
|----------------|---|
| Command Syntax | <pre>ozone sh key list URI --length=&lt;number_of_keys&gt; --prefix=&lt;key_prefix&gt; --start=&lt;starting_key&gt;</pre> |
| Purpose        | Lists the keys in a specified bucket.   |

|           |   |
|-----------|---|
| Arguments | <ul style="list-style-type: none"> <li>• -l, length: Specifies the maximum number of results to return. The default is 100.</li> <li>• -p, prefix: Returns keys that match the specified prefix.</li> <li>• -s, start: Returns results starting with the key <i>after</i> the specified value.</li> <li>• URI: The name of the bucket whose keys you want to list, in the &lt;prefix&gt;://&lt;Service ID&gt;/&lt;volumename&gt;/&lt;bucketname&gt;/ format.</li> </ul> |
| Example   | <pre>ozone sh key list o3://ozone1/hive/year/ --length=100 --prefix=key_prefix --start=day1</pre> <p>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.</p>   |

## 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 Manager 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:

- Ozone S3 Gateway currently does not support ETags and versioning. Therefore, you must disable any configuration related to them when using S3A with Ozone S3 Gateway.
- 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:** It is recommended 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 -keystore [***ssl.client.truststore.location***]
```

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

```
/usr/java/default/bin/keytool -importkeystore -destkeystore $JAVA_HOME/jre/lib/security/jssecacerts -srckeystore [***ssl.client.truststore.location***] -srcalias [***alias***]
```

## 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.require
d=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.key=<accesskey>
-Dfs.s3a.secret.key=<secret> -Dfs.s3a.endpoint=<s3 endpoint url> -Dfs.s3
a.path.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSyste
m -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.require
d=false -Dfs.s3a.change.detection.mode=none -Dfs.s3a.access.key=<accesskey> -
Dfs.s3a.secret.key=<secret> -Dfs.s3a.endpoint=<s3 endpoint url> -Dfs.s3
a.path.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSyste
m -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.require
d=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.pa
th.style.access=true -Dfs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem -l
s -R s3a://testbucket/
```

## Configuring Spark access for 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
```

## Configuring Hive access for S3A

You must configure specific properties for client applications such as Hive 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 configure the following Hive properties using the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml:

```
fs.s3a.bucket.<<bucketname>>.access.key = <accesskey>
```

```
fs.s3a.bucket.<<bucketname>>.secret.key = <secret>
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
```



**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.

- You must provide the required permissions in Ranger to the user running the queries. Consider the following example of providing a user with all permissions. You can change the permissions based on your requirements.
  - 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.

### About this task

The following procedure explains how you can log on 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 logging on to Hue using the Hive or Beeline shell.

### Procedure

1. Create an Ozone bucket.

The following example shows how you can create a bucket named `s3hive`:

```
ozone sh bucket create /s3v/s3hive
```

2. Log on to the Hive shell and perform the specified steps.

- a) Create a table on Ozone using S3A.

```
jdbc:hive2://bv-hoz-1.bv-hoz.abc.site> create external table mytable1(
key string, value int) location 's3a://s3hive/mytable1';
```

- b) Add data to the table.

```
jdbc:hive2://bv-hoz-1.bv-hoz.abc.site> insert into mytable1 values("cldr",1);
jdbc:hive2://bv-hoz-1.bv-hoz.abc.site> insert into mytable1 values("cldr-cdp",1);
```

- c) View the data added to the table.

```
jdbc:hive2://bv-hoz-1.bv-hoz.abc.site> select * from mytable1;
```

## Configuring Impala access for S3A

You must configure specific properties for client applications such as Impala 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 configure the following Impala properties using the Cluster-wide Advanced Configuration Snippet (Safety Valve) for `core-site.xml`:

```
fs.s3a.bucket.<<bucketname>>.access.key = <accesskey>
fs.s3a.bucket.<<bucketname>>.secret.key = <secret>
```

```
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
```



**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.

- You must provide the required permissions in Ranger to the user running the queries. Consider the following example of providing a user with all permissions. You can change the permissions based on your requirements.
  - 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.

## Procedure

1. Create an Ozone bucket.

The following example shows how you can create a bucket named `s3impala`:

```
ozone sh bucket create /s3v/s3impala
```

2. Log on to the Impala shell and perform the specified steps.

- a) Create a table on Ozone using S3A.

```
bv-hoz-1.bv-hoz.abc.site:25003> create external table mytable2(key string, value int) location 's3a://s3impala/mytable1';
```

- b) Add data to the table.

```
bv-hoz-1.bv-hoz.abc.site:25003> insert into mytable2 values("cldr",1);
bv-hoz-1.bv-hoz.abc.site:25003> insert into mytable2 values("cldr-cdp",1);
```

- c) View the data added to the table.

```
bv-hoz-1.bv-hoz.abc.site:25003> select * from mytable2;
```

## 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 https endpoints 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 -keystore <ssl.client.truststore.location>
```

2. Export the CA certificate to PEM format.

```
keytool -export -alias <alias> -file <s3g-ca.crt> -keystore <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                   | <pre>ozones3api create-bucket --bucket buck1</pre> <p>This command creates a bucket buck1.</p>   |
| Adding objects to a bucket          | <pre>ozones3api put-object --bucket buck1 --key Doc1 --body ./Doc1.md</pre> <p>This command adds the key Doc1 containing data from Doc1.md to the bucket buck1.</p>  |
| Listing objects in a bucket         | <pre>ozones3api list-objects --bucket buck1</pre> <p>This command lists the objects in the bucket buck1. An example output of the command is as follows:</p> <pre>{   "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     }   ] }</pre> |
| Downloading an object from a bucket | <pre>ozones3api get-object --bucket buck1 --key Doc1 ./Dp1</pre> <p>This command downloads the key Doc1 from the bucket buck1 as a file Dp1. An example output of the command is as follows:</p> <pre>{   "Content-Type": "text/plain" }</pre>   |

## 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:** In the Cloudera environment only S3 is supported.

### 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

Understand the available documentation to create the required S3 resources.

See Amazon Boto3 documentation to create the required S3 resources: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/resources.html>

```
s3 = boto3.resource('s3',
                    endpoint_url='http://localhost:9878',
                    aws_access_key_id='testuser/scm@EXAMPLE.COM',
                    aws_secret_access_key='c261b6ecabf7d37d5f9ded654b1c724adac9bd9f13e247a235e567e8296d2999'
)
'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='c261b6ecabf7d37d5f9ded654b1c724adac9bd9f13e247a235e567e8296d2999',
    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_client.
```

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.

## 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.

### Setting up o3fs

Select the Ozone bucket to configure o3fs and add specific properties to core-site.xml.

#### 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 required commands:

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

```
ozone sh bucket create /volume/bucket
```

2. Add the properties `fs.o3fs.impl` and `fs.default.name` to `core-site.xml`.

Adding these properties makes the bucket as the default file system for HDFS `dfs` commands and registers the `o3fs` file system type.

```
<property>
  <name>fs.o3fs.impl</name>
  <value>org.apache.hadoop.fs.ozone.OzoneFileSystem</value>
</property>
<property>
  <name>fs.defaultFS</name>
  <value>o3fs://bucket.volume.OzoneServiceId</value>
</property>
```

3. Add the `ozone-file-system-hadoop3.jar` to the classpath.

```
export HADOOP_CLASSPATH=/opt/ozone/share/ozonefs/lib/hadoop-ozone-file-system-hadoop3-*.jar:$HADOOP_CLASSPATH
```



**Note:** With Hadoop 2.x, use the `hadoop-ozone-file-system-hadoop2-*.jar`.

After setting up `o3fs`, you can run `hdfs` commands such as the following on Ozone:

- `hdfs dfs -ls /`
- `hdfs dfs -mkdir /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`.

### Related Information

[Configuration options for Spark to work with o3fs](#)

## Working with 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.

## 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:** 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 `o3fs:///tmp/<tmp-bucket-for-current-user>/`, the ACLs on `ofs:///tmp/<tmp-bucket-for-current-user>/` are the same as those on `o3fs:///tmp/bucket1/`.



**Note:** The name of a user's 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.

### 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 <code>ls -R</code> 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`.

## Ozone configuration options to work with CDP components

There are specific options that you must configure to ensure that other CDP 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 o3fs

After setting up o3fs, you can make configuration updates specific to components such as Spark to ensure that they work with Ozone.

To run Spark jobs with o3fs 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=o3fs://bucket1.vol1.securehost1.example.com:9862
```

### Related Information

[Setting up o3fs](#)

## 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 filesystem 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](#).

## 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.

## 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 CDP 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.

## 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 CDP cluster deployments starting with CDP 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 DN's 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.
- 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. For 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 one of the following:

```
ofs://ozone1/logvol/logbuck/temp_log_dir
```

## Removing Ozone DataNodes from the cluster

You can remove Ozone DataNodes from the CDP 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.

## 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 CDP 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  |
|---|--|--|
| <code>ozone.scm.datanode.pipeline.limit</code>    | Limit for the number of 3-node pipelines that a DataNode can join. | 2<br><br>If you want to increase the number of pipelines based on the number of RaftLog disks, then you can set this value to 0. |
| <code>ozone.scm.pipeline.per.metadata.disk</code> | 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.

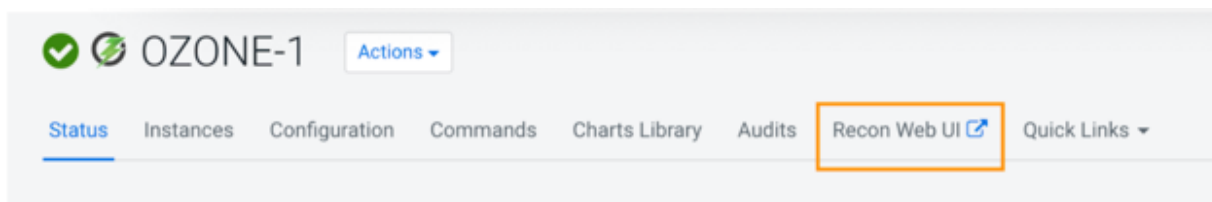
### Access the Recon web user interface

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. Go to the Ozone service.

2. Click Recon Web UI.



The Recon web user interface loads in a new browser window.

## Elements of the Recon web user interface

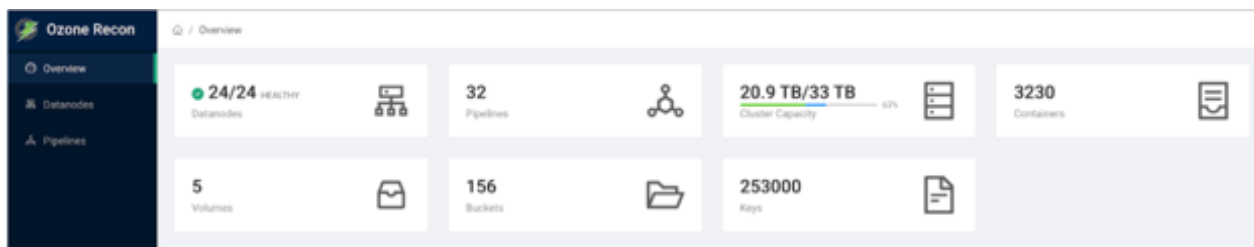
The Recon web user interface displays information about the Ozone cluster on the following pages: Overview, DataNodes, and Pipelines. In addition, a separate page displays information about any missing storage containers.

### 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

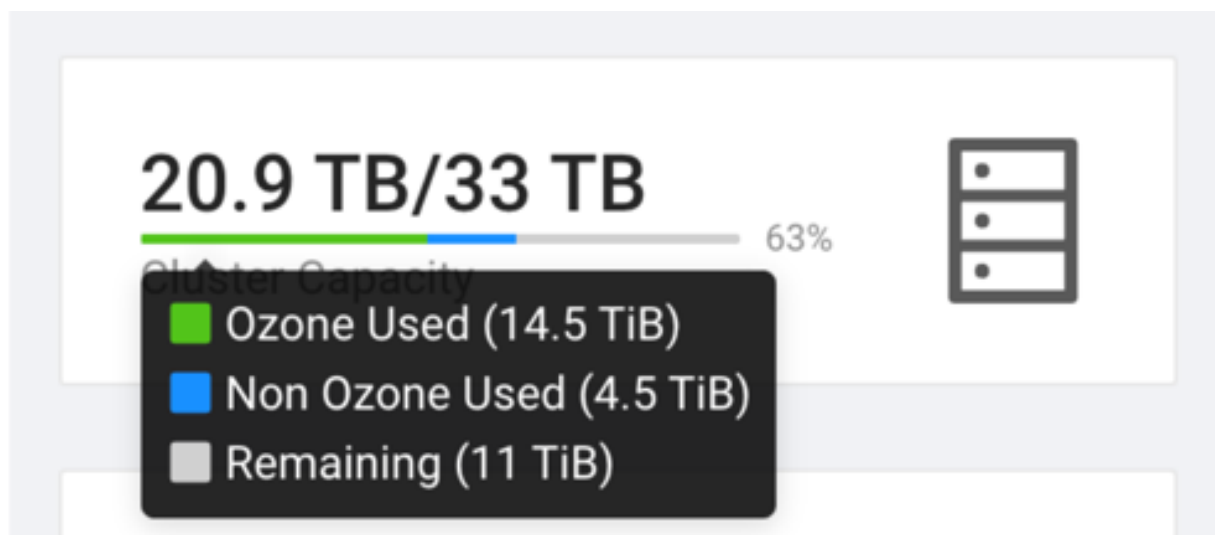
The DataNodes page displays information about the state of the DataNodes in a tabular format. You can load this page either by clicking the DataNodes tab on the left pane or the DataNodes card on the Overview page.

| Status  | Hostname                    | Storage Capacity              | Last Heartbeat       | Pipeline ID(s)   | Containers |
|---------|-----------------------------|-------------------------------|----------------------|--|------------|
| HEALTHY | adycl-1.adycl.root.hwx.site | 151.4 MB / 29.9 GB / 251.9 GB | Apr 8, 2020 10:16 AM | 1fd2509b-4aba-4ae9-99d7-851b0870a2ed<br>8671df3e-d486-41e3-ae86-b9d9902053c  | 2          |
| HEALTHY | adycl-2.adycl.root.hwx.site | 151.4 MB / 24.4 GB / 251.9 GB | Apr 8, 2020 10:16 AM | c3e72990-1684-4962-8146-4efec10f094<br>8671df3e-d486-41e3-ae86-b9d9902053c   | 2          |
| HEALTHY | adycl-3.adycl.root.hwx.site | 151.4 MB / 22.1 GB / 251.9 GB | Apr 8, 2020 10:16 AM | 1f2c47d3-554e-4716-b0cc-e3e7c7f53cd0b<br>8671df3e-d486-41e3-ae86-b9d9902053c | 2          |

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.

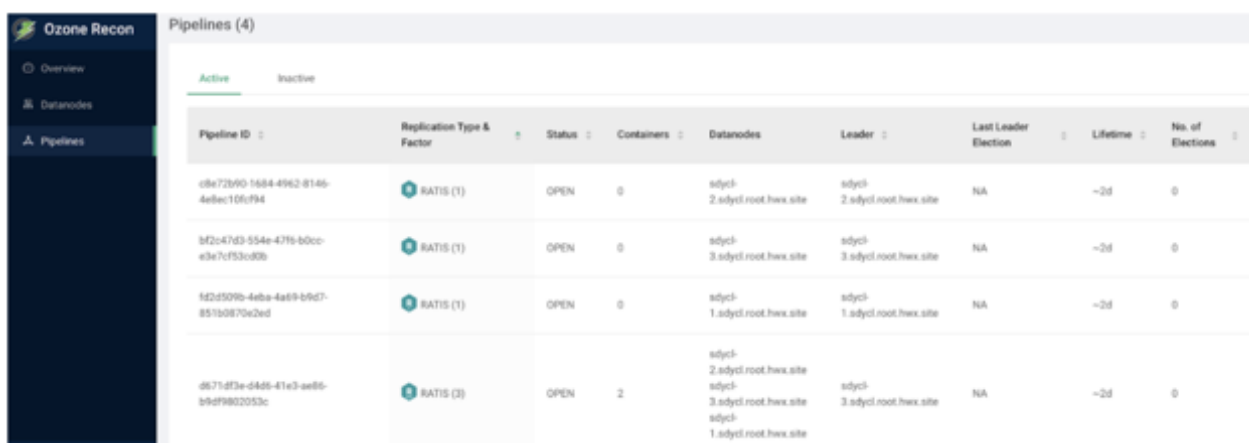


- **Last Heartbeat:** The timestamp of the last heartbeat sent by the particular DataNode to the SCM.
- **Pipeline ID(s):** The IDs of the pipelines to which the particular DataNode belongs.
- **Containers:** The number of storage containers inside the particular DataNode.

## 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 screenshot shows the 'Pipelines (4)' page in the Ozone Recon web interface. The left sidebar has 'Overview', 'Datanodes', and 'Pipelines' (selected). The main area shows a table of pipeline information with columns: Pipeline ID, Replication Type & Factor, Status, Containers, Datanodes, Leader, Last Leader Election, Lifetime, and No. of Elections. There are four rows of pipeline data, all with a status of 'OPEN'.

| Pipeline ID                          | Replication Type & Factor | Status | Containers | Datanodes   | Leader                      | Last Leader Election | Lifetime | No. of Elections |
|--------------------------------------|---------------------------|--------|------------|---|-----------------------------|----------------------|----------|------------------|
| c8e72b90-1684-4962-8146-4e8ec10fc934 | RATIS (1)                 | OPEN   | 0          | sdycl-2.sdycl.root.hwx.site   | sdycl-2.sdycl.root.hwx.site | NA                   | -3d      | 0                |
| 3f2c47e3-554e-4716-b0cc-e3e7cf53c80b | RATIS (1)                 | OPEN   | 0          | sdycl-3.sdycl.root.hwx.site   | sdycl-3.sdycl.root.hwx.site | NA                   | -3d      | 0                |
| 1d2d509b-44ba-4a69-b9d7-851b0870a2ed | RATIS (1)                 | OPEN   | 0          | sdycl-1.sdycl.root.hwx.site   | sdycl-1.sdycl.root.hwx.site | NA                   | -3d      | 0                |
| d671d73e-d4d5-41e3-ae85-b9d9802053c  | RATIS (3)                 | OPEN   | 2          | sdycl-2.sdycl.root.hwx.site<br>sdycl-3.sdycl.root.hwx.site<br>sdycl-1.sdycl.root.hwx.site | sdycl-3.sdycl.root.hwx.site | NA                   | -3d      | 0                |

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.

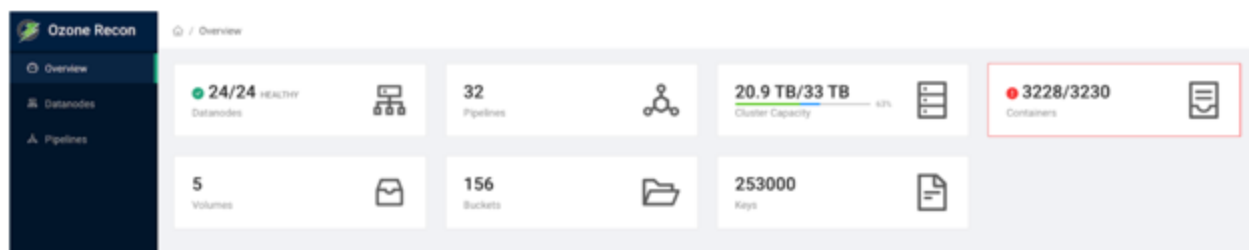


**Note:** This field does not show any data for the current release.

## 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.



Overview

Datanodes

Pipelines

Missing Containers

Missing Containers (2)

| Container ID                       | No. of Keys | Datanodes   | Pipeline ID                           | Missing Since       |
|------------------------------------|-------------|---|---------------------------------------|---------------------|
| <div><div></div><div>1</div></div> | 1235        | <div><div></div>localhost1.storage.enterprise.com</div> <div><div></div>localhost3.storage.enterprise.com</div> <div><div></div>localhost5.storage.enterprise.com</div> | 05e3d908-ff01-4ce6-ad75-f3ec79bdc7982 | Jan 8, 2020 5:49 AM |

| Volume      | Bucket         | Key          | Size    | Date Created         | Date Modified        |
|-------------|----------------|--------------|---------|----------------------|----------------------|
| vol-0-20448 | bucket-0-12811 | key-0-77505  | 10.2 kB | Nov 26, 2019 1:18 PM | Nov 26, 2019 1:18 PM |
| vol-0-20448 | bucket-0-12811 | key-21-64511 | 5.69 MB | Nov 26, 2019 1:19 PM | Nov 26, 2019 1:19 PM |
| vol-0-20448 | bucket-0-12811 | key-22-68104 | 189 kB  | Nov 26, 2019 1:19 PM | Nov 26, 2019 1:19 PM |

1-3 of 3 keys

<

1

>

|                                    |      |   |                                      |                     |
|------------------------------------|------|---|--------------------------------------|---------------------|
| <div><div></div><div>2</div></div> | 1356 | <div><div></div>localhost1.storage.enterprise.com</div> <div><div></div>localhost3.storage.enterprise.com</div> <div><div></div>localhost5.storage.enterprise.com</div> | 04a5d908-ff01-4ce6-ad75-f3ec73dfc8a2 | Jan 8, 2020 5:51 AM |
|------------------------------------|------|---|--------------------------------------|---------------------|

1-2 of 2 missing containers

<

1

>

10 / 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.

Missing Containers (2)

| Container ID | No. of Keys | DataNodes   | Pipeline ID                           | Missing Since       |
|--------------|-------------|---|---------------------------------------|---------------------|
| + 1          |             | localhost1.storage.enterprise.com<br>localhost3.storage.enterprise.com<br>localhost5.storage.enterprise.com | 05e3d908-ff01-4ce6-ad75-f3ec79bdc7982 | Jan 8, 2020 5:49 AM |
| + 2          | 1356        | localhost1.storage.enterprise.com<br>localhost3.storage.enterprise.com<br>localhost5.storage.enterprise.com | 04a5d908-ff01-4ce6-ad75-f3ec73dfc8a2  | Jan 8, 2020 5:51 AM |

1-2 of 2 missing containers

## 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.

### Procedure

1. In Cloudera Manager, go to the Ozone service.
2. 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 FileNotFoundException error.

3. 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.

4. 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.
5. 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.

## 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. In Cloudera Manager, go to the Ozone service.
2. Click Configuration.
3. 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.

4. 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.