Cloudera Runtime 7.0.3

Storing Data Using Ozone

Date published: 2019-08-21
Date modified:

https://docs.cloudera.com/

Legal Notice

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

Introduction to Ozone

Apache Ozone is a scalable, redundant, and distributed object store optimized for big data workloads. Apart from scaling to billions of objects of varying sizes, applications that use frameworks like Apache Spark, Apache YARN and Apache Hive work natively on Ozone without any modifications. Ozone natively supports the S3 API and provides a Hadoop-compatible file system interface. Ozone is typically available in a CDP Private Cloud Base deployment.

⚠ **Important:** Ozone is available for technical preview and considered to be under development. Do not use this component in your production systems. If you have questions regarding Ozone, contact support by logging a case on the Cloudera Support Portal.

Technical preview features are not guaranteed troubleshooting and fixes.

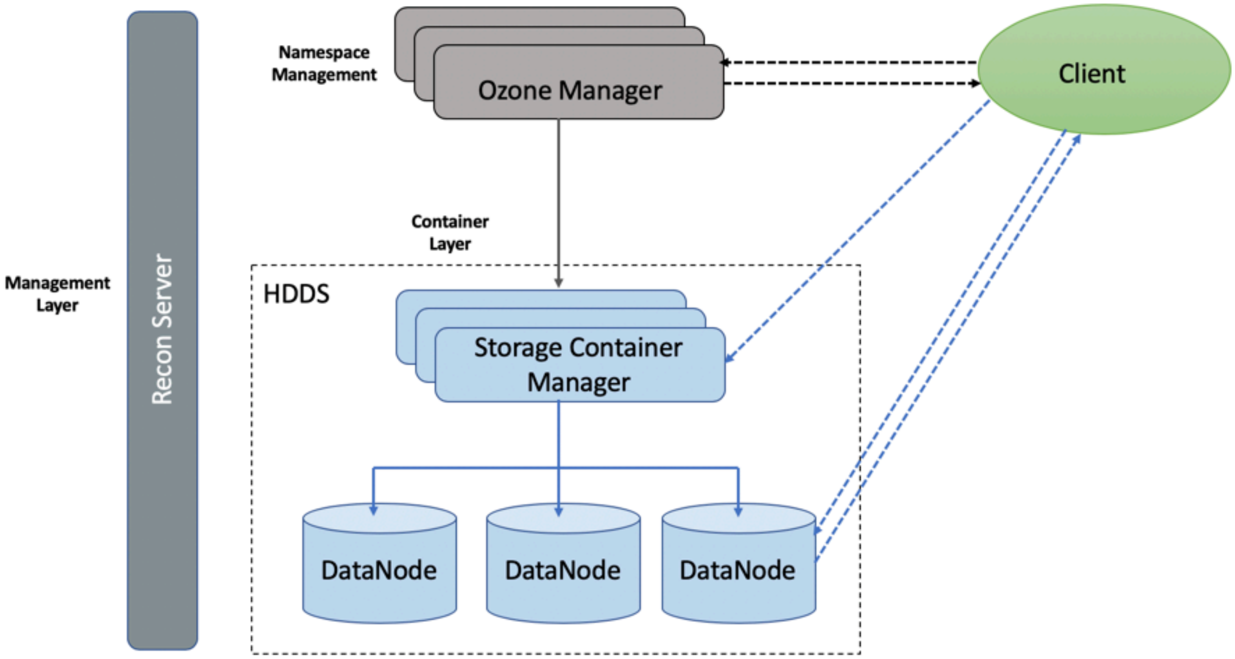
Ozone consists of three important storage elements: volumes, buckets, and keys. Each key is part of a bucket, which, in turn, belongs to a volume. Only an administrator can create volumes. Depending on their requirements, regular users can create buckets in volumes. Ozone stores data as keys inside these buckets.

When a client writes a key, Ozone stores the associated data on DataNodes in chunks called blocks. Therefore, each key is associated with one or more blocks. Within a DataNode, multiple unrelated blocks can reside in a storage container.

Ozone architecture

Ozone can be co-located with HDFS with single security and governance policies for easy data exchange or migration and also offers seamless application portability. Ozone has a scale-out architecture with minimal operational overheads. Ozone separates management of namespaces and storage, helping it to scale effectively. The Ozone Manager (OM) manages the namespaces while the Storage Container Manager (SCM) handles the containers.

The following diagram shows the components that form the basic architecture of Ozone:



Hadoop Distributed Data Store

Ozone is built on a highly available, replicated block storage layer called Hadoop Distributed Data Store (HDDS).

Blocks

Blocks are the basic unit of storage. In Ozone, each block is of 256 MB in size. A collection of blocks forms a storage container. The SCM allocates blocks inside storage containers for the client to store data.

Storage Containers

A storage container is a group of unrelated blocks managed together as a single entity. A container exists in a DataNode and is the basic unit of replication, with a capacity of 2 GB to 16 GB.

DataNodes

DataNodes contain storage containers comprising of data blocks. The SCM monitors DataNodes through heartbeats.

Ozone Manager

The Ozone Manager (OM) is the metadata manager for Ozone. The OM manages the following storage elements:

- The list of volumes for each user
- The list of buckets for each volume
- The list of keys for each bucket

The OM maintains the mappings between keys and their corresponding Block IDs. When a client application requests for keys to perform read and write operations, the OM interacts with the SCM for information about blocks relevant to the read and write operations, and provides this information to the client. In addition, the OM also handles metadata operations from the clients.

Ozone Manager

The Ozone Manager (OM) is a highly available namespace manager for Ozone.

OM manages the metadata for volumes, buckets, and keys. OM maintains the mappings between keys and their corresponding Block IDs. When a client application requests for keys to perform read and write operations, OM interacts with SCM for information about blocks relevant to the read and write operations, and provides this information to the client. In addition, OM also handles metadata operations from the clients.

Storage Container Manager

The Storage Container Manager performs multiple critical functions for an Ozone cluster.

SCM manages the addition and removal of DataNodes, and allocates storage containers and blocks. SCM also manages block collections, ensuring that the blocks maintain the required level of replication. SCM allocates blocks to clients through OM for read and write operations. In addition, SCM executes recovery actions when faced with DataNode or disk failures.

Pipelines

Pipelines determine the replication strategy for the blocks associated with a write operation.

Recon Server

Recon is the management interface for Ozone. Recon provides a unified management API for Ozone.

How Ozone manages read operations

The client requests the block locations corresponding to the key it wants to read. The Ozone Manager (OM) returns the block locations if the client has the required read privileges.

The following steps explain how Ozone manages read operations:

1. The client requests OM for block locations corresponding to the key to read.

2. OM checks the ACLs to confirm whether the client has the required privileges, and returns the block locations and the block token that allows the client to read data from the DataNodes.
3. The client connects to the DataNode associated with the returned Block ID and reads the data blocks.

How Ozone manages write operations

The client requests a block from the Ozone Manager (OM) to write a key. OM returns the Block ID and the corresponding DataNodes for the client to write data.

The following steps explain how Ozone manages write operations:

1. The client requests a block from OM to write a key. The request includes the key, the data to write, the pipeline type, and the replication count.
2. OM finds the blocks that match the request and returns them to the client.

   Note: If security is enabled on the cluster, OM also provides a block token along with the block location to the client. The client uses the block token to connect to the DataNodes and send the command to write chunks of data.
3. The client connects to the DataNodes associated with the returned block information and writes the data.
4. After writing the data, the client updates the block information on OM.
5. OM records the associated key information.

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.

For more information about the various Ozone command-line tools and the Ozone shell, see https://ozone.apache.org/docs/1.3.0/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.

Creating a volume

Only an administrator can create a volume and assign it to a user.

Note: You must assign administrator privileges to users before they can create volumes. For more information, see Assigning administrator privileges to users on page 8.

Command Syntax	`ozone sh volume create --quota=<volumecapacity> --user=<username> /<volumename>`
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. • The name of the volume to create.
Example	`ozone sh volume create --quota=2TB --user=usr1 /vol1` This command creates a 2-TB volume named vol1 for user usr1.

Deleting a volume

Command Syntax	`ozone sh volume delete /<volumename>`
Purpose	Deletes the specified volume, which must be empty.
Arguments	The name of the volume to delete.
Example	`ozone sh volume delete /vol2` This command deletes the empty volume vol2.

Viewing volume information

Command Syntax	`ozone sh volume info /<volumename>`
Purpose	Provides information about the specified volume.
Arguments	The name of the volume whose details you want to view.
Example	`ozone sh volume info /vol3` This command provides information about the volume vol3.

Listing volumes

Command Syntax	`ozone sh volume list --user <username>`
Purpose	Lists all the volumes owned by the specified user.
Arguments	-u, user: The name of the user whose volumes you want to list.
Example	`ozone sh volume list --user usr2` This command lists the volumes owned by user usr2.

Updating a volume

Command Syntax	`ozone sh volume update --quota=<volumecapacity> --user=<username> /<volumename>`
Purpose	Updates the quota or changes the user associated with a specific volume.
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 an optional parameter. • The name of the volume to update.
Example	`ozone sh volume update --quota=10TB /vol4` This command updates the quota of volume vol4 to 10TB.

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.

Creating a bucket

Command Syntax	`ozone sh bucket create /<volumename> /<bucketname>`
Purpose	Creates a bucket in the specified volume.

Arguments	The name of the bucket to create in the /volume/bucket format.
Example	`ozone sh bucket create /vol1/buck1` This command creates a bucket buck1 in the volume vol1.

Deleting a bucket

Command Syntax	`ozone sh bucket delete /<volumename>/<bucketname>`
Purpose	Deletes the specified bucket, which must be empty.
Arguments	The name of the bucket to delete in the /volume/bucket format.
Example	`ozone sh bucket create /vol1/buck2` This command deletes the bucket buck2.

Viewing bucket information

Command Syntax	`ozone sh bucket info /<volumename>/<bucketname>`
Purpose	Provides information about the specified bucket.
Arguments	The name of the bucket whose details you want to view.
Example	`ozone sh bucket info /vol1/buck3` This command provides information about bucket buck3.

Listing buckets

Command Syntax	`ozone sh bucket list /<volumename> --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. • The name of the volume whose buckets you want to list.
Example	`ozone sh bucket list /vol2` This command lists all the buckets in volume vol2.

Commands for managing keys

The Ozone shell commands enable you to upload, download, view, delete, and list keys.

Downloading a key from a bucket

Command Syntax	`ozone sh key get /<volumename>/<bucketname>/<keyname> <local_filename>`
Purpose	Downloads the specified key from a bucket in the Ozone cluster to the local file system.
Arguments	• keyname: The name of the key in /<volumename>/<bucketname>/<keyname> format. • filename: The name of the file to which you want to write the key.
Example	`ozone sh key get /hive/jun/sales.orc sales_jun.orc` This command downloads the sales.orc file from /hive/junbucket and writes to the sales_jun.orc file present in the local file system.

Uploading a key to a bucket

Command Syntax	`ozone sh key put /<volumename>/<bucketname>/<keyname> <filename>`
Purpose	Uploads a file from the local file system to the specified bucket in the Ozone cluster.
Arguments	• keyname: The name of the key in /<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 /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.

Deleting a key

Command Syntax	`ozone sh key delete /<volumename>/<bucketname>/<keyname>`
Purpose	Deletes the specified key from the Ozone cluster.
Arguments	The name of the key in /<volumename>/<bucketname>/<keyname> format.
Example	`ozone sh key delete /hive/jun/sales_duplicate.orc` This command deletes the sales_duplicate.orc key.

Viewing key information

Command Syntax	`ozone sh bucket info /<volumename>/<bucketname>/<keyname>`
Purpose	Provides information about the specified key.
Arguments	The name of the key whose details you want to view.
Example	`ozone sh bucket info /hive/jun/sales_jun.orc` This command provides information about the sales_jun.orc key.

Listing keys

Command Syntax	`ozone sh key list /<bucketname> --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. • The name of the bucket whose keys you want to list.
Example	`ozone sh key list /hive/jun/` This command lists all the keys in the volume /hive/jun.

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.

URL schema for Ozone S3 Gateway

Ozone S3 Gateway supports both the virtual host-style URL s3 bucket addresses (http://bucketname.host:9878) and the path-style addresses (http://host:9878/bucketname). By default, S3 Gateway uses path-style addresses.

If you want to use virtual host-style URLs, you must set the main domain name in ozone-site.xml.

```
<property>
   <name>ozone.s3g.domain.name</name>
   <value>s3g.internal</value>
</property>
```

URL to browse Ozone buckets

You can browse the contents of Ozone buckets by adding ?browser=true to the bucket URL.

For example, you can use the following URL to browse the contents of the Ozone bucket buckoz1:

```
http://localhost:9878/buckoz1?browser=true
```

Important: You cannot use the Ozone S3 Gateway browser to view buckets created using the Ozone shell. Only the buckets created using the Ozone S3 Gateway are visible from the browser.

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
- GET Object
- DELETE Object
- HEAD Object
- Multipart Upload (Except the listing of the current MultiPart Uploads)

Mapping for an Ozone volume in Amazon S3 API

The Amazon S3 REST API does not provide any entity that directly maps to an Ozone volume in the Ozone S3 Gateway. Instead, a bucket name defined in the S3 API maps to its corresponding Ozone volume/bucket location.

Consider the following example of a bucket b1 created using the S3 API:

```
aws s3api --endpoint-url http://localhost:9878 create-bucket --bucket=b1
```

You can view the bucket location using the following command:

```
ozone sh bucket path b1
```

Examples of using the Amazon Web Services command-line interface for 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 example shows how you can define an alias for the S3 Gateway endpoint URL:

```
alias ozones3api='aws s3api --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 b1` This command creates a bucket b1.
Adding objects to a bucket	`ozones3api put-object --bucket b1 --key Doc1 --body ./Doc1.md` This command adds the key Doc1 containing data from Doc1.md to the bucket b1.

Listing objects in a bucket	`ozones3api list-objects --bucket b1`
	This command lists the objects in the bucket b1. 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 b1 --key Doc1 ./Dpc1`
	This command downloads the key Doc1 from the bucket b1 as a file Dpc1. An example output of the command is as follows:
	`{`
	`"ContentType": "application/octet-stream",`
	`"ContentLength": 2845`

Working with Ozone File System

Ozone File System (OzoneFS) is a Hadoop-compatible file system. Applications such as Hive, Spark, YARN, and MapReduce run natively on OzoneFS without any modifications.

OzoneFS resides on a bucket in the Ozone cluster. All the files created through OzoneFS 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 OzoneFS.

Setting up OzoneFS

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

Procedure

1. Select the Ozone bucket on which you want OzoneFS to reside.

If you do not have a designated volume or bucket for OzoneFS, 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</value>
</property>
```

3. Add the ozone-filesystem.jarfile to the classpath.

```
export HADOOP_CLASSPATH=/opt/ozone/share/hadoop/ozonefs/hadoop-ozone-fil
esystem.jar:$HADOOP_CLASSPATH
```

After setting up OzoneFS, 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 OzoneFS show up as directories and files in OzoneFS.

Related Information

Configuration updates for Spark to work with OzoneFS

Configuration updates for Spark to work with OzoneFS

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

Configuration changes for Spark

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 OzoneFS