Cloudera Runtime 7.3.1

Data Migration from HDFS to Ozone

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

Contents

Migrating your data from HDFS to Ozone

You can migrate data from your existing HDFS deployment to Ozone for benefits such as scaling the metadata to billions of files; supporting denser, more cost-effective storage nodes as a part of a strategy to segregate storage and compute; or cold storage and archival storage needs.

This document assumes that you have an existing HDFS cluster from which you want a part or whole data to be moved to Ozone. The Ozone service can exist in the same cluster as HDFS or in a different cluster.

Considerations for transferring data from HDFS to Ozone

There are certain factors that you must consider when transferring data from your HDFS deployment to Ozone. The factors include HDFS dependency on Ozone, roles and sizing considerations, and Ozone namespace concepts.

HDFS dependency on running Ozone

Although Ozone is an independent service and does not use any runtime component of HDFS, Ozone relies on the mechanism of how Cloudera Manager distributes client configurations to different services such as Hive, Spark, YARN, and so on. Because this mechanism of Cloudera Manager does not work without HDFS, Ozone requires HDFS as a dependency when running on a Cloudera cluster.

If applications such as Hive, Spark, YARN, or others require the cluster HDFS client configurations, Ozone client configurations are also bundled along with the HDFS configurations. Therefore, out-of-the-box support for Ozone based storage can be made available in Cloudera Data Warehouse and Cloudera Machine Learning applications. This dependency also implies that changing any Ozone client configuration requires restarting of all the dependent client services.

For more information, see the *Ozone - Steps to enable HDFS dependency on running Ozone* section in the Complete Post-Upgrade steps for upgrades to Cloudera Base on premises documentation.

Roles and sizing considerations for Ozone

You must be aware of the hardware sizing requirements for your Ozone deployment. Further, you must understand the roles for which you need to configure different nodes.

For the Ozone hardware sizing requirements, see Ozone.

The following Ozone roles require different node requirements:

- Six master node instances: Three each of Ozone Manager (OM) and Storage Container Manager (SCM) services. Ozone has no other dependencies for high availability because the three master nodes form a Raft ring and replicate metadata with instantaneous failover without requiring ZooKeeper or Journal Nodes, unlike in HDFS.
- One Recon node and S3 Gateway daemons: 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 OM and SCM.

  S3 Gateways provide REST access to Ozone over HTTP and support the AWS-compatible S3 API. Cloudera recommends deploying a number of gateways behind a load balancer. These gateways are stateless and can also be co-hosted with DataNodes.

Related Information

Ozone architecture

Ozone namespace concepts

The Ozone namespace includes the following conceptual entities: volumes, keys, and buckets. 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.

The three main entities can be explained as follows:

Volume

An Ozone volume is similar to a user account, for example; sales, marketing, engineering, and so on.

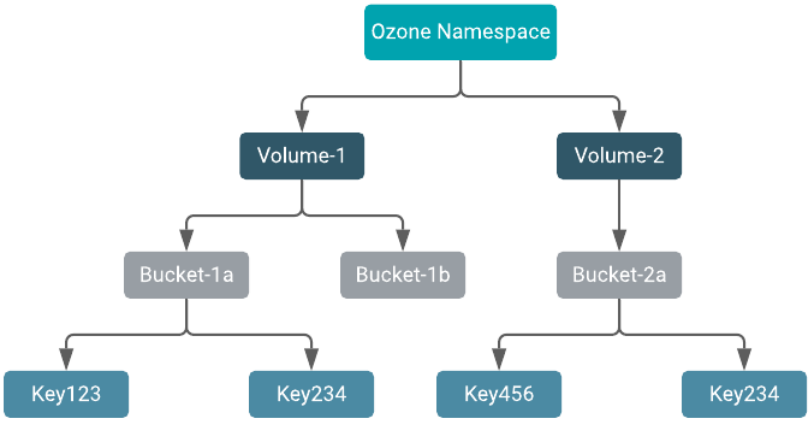
Buckets

An Ozone bucket is similar to a directory or an Amazon S3 bucket. A bucket any number of keys but cannot contain other buckets.

Keys

An Ozone key is similar to any key in an object store or a file in a filesystem. If a key is used as a filesystem, the prefixes are treated as directories.

The following image depicts the relationship between volumes, buckets, and keys in the Ozone namespace:



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. Storage container is an internal Ozone abstraction that represents a collection of blocks and that singularly forms the unit of replication in Ozone.

Related Information

Ozone architecture

Ozone configurations

Ozone's main configuration file is ozone-site.xml that carries the server or the client configurations depending on the component.

The details of the server-side or client-side configuration files are as follows:

- The server-side ozone-site.xml files such as OM, SCM, Recon, and S3 Gateway are present in their respective run directories; for example, /var/run/cloudera-scm-agent/process/45-ozone-OZONE_MANAGER, under ozone-conf.
- The Ozone shell clients or S3 Gateway configurations are available under /etc/ozone/conf on all the hosts.
- In addition to ozone-site.xml, the client-only configurations are also present in core-site.xml used by clients. Therefore, Ozone client applications such as Hive, Spark and so on, typically have configurations in ozone-site.xml and core-site.xml.

The Ozone client configurations contain the location information for the client to know where the OM nodes are. A quorum of OM nodes is identified by a Service ID (ozone.service.id), and therefore, the Service ID usually prefixes any ofs:// path. Using the Service ID to configuration mapping, the Ozone client identifies the Ozone Manager quorum with which to interact.

Note: If the ozone shell or hadoop-based ofs:// commands do not work as expected, it is likely that the Ozone client is trying to find the Ozone Manager in the default local host, and keeps retrying until failure. Therefore, you must ensure that the configuration files have the required information.

Depending on the services configured for the Cloudera cluster, you can also find other general purpose configuration files such as hdfs-site.xml, core-site.xml and ranger-*.xml files in the Ozone conf directories.

Adding Core Configuration service for Ozone

After enabling Kerberos, you can now add the Core Configuration service for Ozone using the Cloudera Manager version 7.4.4 and higher.

Adding Core Configuration service

You must add the Core Configuration service for your cluster using Cloudera Manager. To enable Kerberos on Ozone, HDFS is not required as a prerequisite for Ozone to access core-site.xml.

1. Log in to Cloudera Manager.
2. Navigate to the cluster to which you want to add the Core Configuration service.
3. Click Actions > Add Service. Add Service to Cluster page appears.
4. Select the Core Configuration service and click Continue.
5. In the Assign Roles page, assign the hosts and click Continue.
6. In the Review Changes page, you can provide the values for the parameters available and click Continue.
7. The Command Details page appears. You can monitor the status of the Core Configuration service. Click Continue.
8. In the Summary page, click Finish. The Core Configuration service is now added to your cluster.

Adding Ozone Service

You must add the Ozone service for your cluster using Cloudera Manager.

1. Log in to Cloudera Manager.
2. Navigate to the cluster to which you want to add the Ozone service.
3. Click Actions > Add Service. Add Service to Cluster page appears.
4. Select the Ozone service and click Continue.
5. The Select Dependencies page appears. As optional dependencies, you can either select HDFS or Core Configuration. Click Continue.
6. In the Assign Roles page, assign the hosts and click Continue.
7. In the Review Changes page, you can provide the values for the parameters available and click Continue.
8. The Command Details page appears. You can monitor the status of the Core Configuration service. Click Continue.
9. In the Summary page, click Finish. The Ozone service is now added to your cluster.

Permission models on Ozone

Ozone uses Access Control Lists (ACLs) to provide various access permissions to users. To enable ACLs in ozone, you must set ozone.acl.enabled = true. Ozone supports two types of authorizers to set the access permissions: native authorizer and Ranger authorizer.

Considerations for using the authorizers

You must be aware of specific considerations for using the different authorizers.

- You can configure Ozone to use any one of the authorizers, but not both at the same time.
- If using the Ranger authorizer for Ozone, the only way you can set or change policies is by using the Ranger user interface and the Ranger REST API. Ozone does not enable you to update ACLs using its CLI when managed by Ranger.
- Every volume has an *owner*, who has default permissions to all the entities within the volume. A volume can be created with another user as owner using a CLI flag. Ozone does not support bucket ownership.
- By default, the Ozone Manager service user om is the cluster administrator, and the only user with global access to all the volumes. The policy to provide the cluster administrator with the global access to all the volumes is a part of the default policies available with the Ozone-Ranger plug-in. Therefore, by default, only the om user can

list volumes at the root level of ofs:// (== list all volumes). You can add more users to this default Ozone policy in Ranger for admin access to Ozone.

Ranger authorizer

Cloudera recommends using the Ranger authorizer to manage access permissions for Ozone in Cloudera deployments. The Ozone-Ranger plug-in supports resources at the level of a volume or a bucket or a key, and the supported operations are similar to those of HDFS. You can use the Ozone-Ranger plug-in to configure policies at the level of a specific volume or a bucket or a key.

HDFS namespace mapping to Ozone

Mapping HDFS directories to Ozone volumes, buckets, and keys is extremely important to understand as you migrate the content. Apart from how to perform the mapping, the number of Ozone elements to create for the migration of data is also an important consideration.

Ozone volumes and HDFS mapping

You can follow either of two approaches to map Ozone volumes with HDFS directories. You can create a volume called '/user' in Ozone and create buckets that map to the second-level HDFS directories. This approach ensures that all HDFS existing paths in Hive or Spark applications can migrate to using Ozone by merely adding the ofs://<service-id> prefix.

Alternatively, in deployments where Ozone acts as an archival storage for multiple clusters, you can create a volume for every source HDFS cluster and then create buckets within that for individual directories or applications that use the HDFS cluster.

Note: Ozone supports encryption zones *at the bucket level*. Therefore, when moving encrypted data from HDFS, you must ensure that an encryption zone in HDFS is not split across buckets in Ozone.

Number of Ozone volumes, buckets, and keys for the migration

Although there is no strict limitation on the number of volumes, buckets and keys that can be managed by Ozone, Cloudera recommends the following:

- Volumes in the range of 1-20
- Buckets per volume in the range of 1-1000
- Keys under /volume/bucket in the range of one to a few hundred million

Using OFS to abstract volumes and buckets

The Ozone Filesystem (OFS) connector abstracts out the volume and the bucket from the FS client and works internally with the Ozone back-end to create them. For example, the following command creates the volume, bucket, and then automatically creates the two directories, if you have the required permissions:

```
hadoop fs -mkdir -p ofs://ozone1/spark-volume/spark-bucket/application1/inst
ance1
```

Process of migrating the HDFS data to Ozone

After verifying the prerequisites for HDFS to Ozone data migration process, you can use the distcp command to transfer the data from HDFS to Ozone.

About this task

You need to verify that the cluster satisfies certain prerequisites for migrating data from the HDFS sources and then prepare the Ozone cluster for ingesting data from the HDFS cluster. You can create additional Ranger policies after preparing the Ozone cluster. Then you can migrate the HDFS data by using the distcp command. After the HDFS to Ozone migration process completes, validate the migrated data. You can also configure other services including Hive and Spark to work with Ozone.

Read the topics in this section for information about preparation for the HDFS to Ozone migration process and topics that describe how to complete the HDFS to Ozone migration process.

Verifying the migration prerequisites

You must create the destination Ozone cluster and ensure that the cluster satisfies the prerequisites for migrating data from the HDFS sources.

Procedure

- Before migrating the data, you must ensure that you have managed the prerequisites.
  - Deployed either a new Ozone cluster that is ready for the migration, or have added Ozone to an existing cluster, and restarted all the services on that cluster.
  - Defined the configuration property ozone.service.id, which denotes a quorum of Ozone Managers similar to a Namespace ID in HDFS.
  - Configured the Ranger service on the cluster.
  - Added HDFS and Ranger as dependencies to Ozone before restarting all the services.
  - Verified that Ozone is operational through the following basic shell commands:

```
> kinit as ‘om’ user or a new user with admin privileges for Ozone.
(Use ozone.keytab from Ozone Manager’s run directory)

===> Create Ozone Volume
> ozone sh volume create /test-vol
===> Create Ozone Bucket
> ozone sh bucket create /test-vol/test-bucket


===> Write test key
> ozone sh key put /test-vol/test-bucket/test-key <path_to_key>

===> Test get key info
> ozone sh key info /test-vol/test-bucket/test-key


===> Verify ofs:// working
> hdfs dfs -cat ofs://<ozone.service.id>/test-vol/test-bucket/test-key

===> Delete the created artifacts
> hdfs dfs -rm -r  ofs://<ozone.service.id>/test-vol
```

Preparing Ozone for data ingestion

You must prepare the Ozone cluster for ingesting the data from the HDFS cluster.

About this task

Consider the example of moving an encrypted HDFS directory to Ozone. The following procedure outlines the steps you must perform to create the equivalent bucket and target directory in Ozone.

Procedure

1. Decide on the HDFS to Ozone mapping scheme.

   Consider the example of mapping the HDFS /user directory to an Ozone volume.
2. Create the volume /user as an admin user.

```
> kinit as admin user and create the volume
> ozone sh volume create o3://<ozone.service.id>/user
```

3. Create an encrypted bucket to migrate a user's directory from the HDFS cluster.

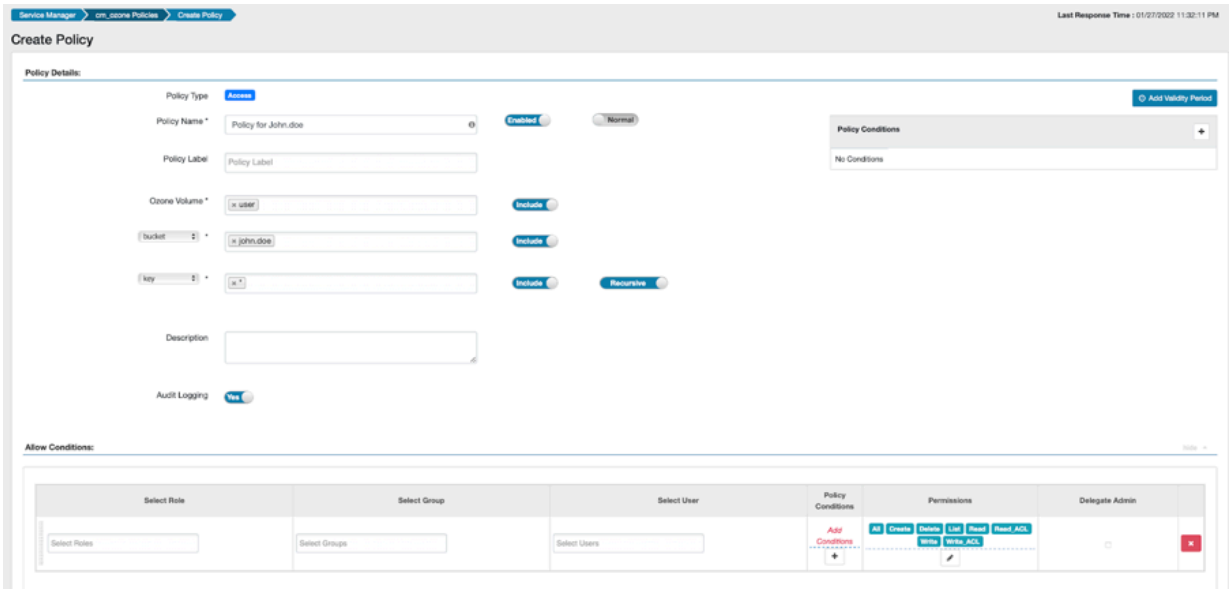
   The following example shows how you can create an encrypted bucket for a user john.doe.

```
> kinit as admin user and create the bucket for john.doe
> hadoop key create xxx_key
> ozone sh bucket create -k xxx_key /user/john.doe
(Optionally create a target dir inside the bucket if needed)
> hdfs dfs -mkdir ofs://<ozone.service.id>/user/john.doe/target_dir
```

4. Provide permissions to the user to access the encrypted bucket.

   Use Ranger to provide the required permissions.

   The following image shows the Ranger policy configured for the user john.doe on the /user/john.doe bucket.



Note: If you run kinit as user john.doe, ensure that john.doe has the permissions to create keys using Ranger KMS. Also, ensure that any user who wants to write data into an encrypted bucket has the Decr ypt EEK permissions through Ranger KMS.

Creating additional Ranger policies for the keyadmin user

After creating the encrypted bucket, add the required policies to allow reads to the bucket.

About this task

Consider the key creation example in Preparing Ozone for data ingestion on page 8. The prefix xxx is an administrator, and therefore, has all the privileges.

Procedure

- Add the keyadmin policy to make xxx a key administrator, and allow john.doe access to the encrypted key.

Moving data from HDFS to Ozone using the distcp command

Use the hadoop distcp command to move the content from the HDFS source cluster.

Before you begin

You must consider the following before running the distcp command:

- Execute the distcp command from the destination cluster.
- Ensure that the distcp user can run a MapReduce job on YARN. Otherwise, you must tweak the following configurations to enable the distcp user:
  - allowed.system.users
  - banned.users
  - min.user.id
- If the source directories have a high file count, you can create a manual copy listing as specified in the following example.

```
> hdfs dfs -ls hdfs://<hdfs-nameservice>/user/john.doe/application1/* >
src_files
```

  The copy listing output file can be read and submitted as input one by one to a distcp job.

Procedure

- Consider the example of a user john.doe whose data is from the /user/john.doe/application1/ directory and you want to transfer to Ozone, run the distcp command as specified.

```
> hadoop distcp -direct hdfs://<hdfs-nameservice>/user/john.doe/applicat
ion1 ofs://<ozone.service.id>/user/john.doe/
```

Note:

- If your bucket type is OBS (object-store), you must use the -direct flag. Ozone supports atomic renames with FSO (file-system optimized) buckets. FSO is the default bucket layout starting OZONE-Parcel 718.2.0 version.
- If using the object store (OBS) bucket type, a rename at a directory level internally means multiple client calls to rename the directory in every subdirectory and key with the required prefix, eventually ending with the top-level directory's renaming. Hence, writing to a temp directory in target and then renaming a top-level directory is a costly operation in Ozone for OBS buckets, unlike FSO buckets.
- Ozone *does not* support file appends, concatenations, and snapshots. Therefore, Ozone also does not support the distcp flags append, -diff, -rdiff, and -blocksperchunk.
- From HDFS storage to DistCp in CDH5 and HDP2 to Apache Ozone in Cloudera, you must set the -skipcrccheck parameter to skip the file checksum comparison. From HDFS storage to DistCp in CDH6, HDP3, and Cloudera to Apache Ozone in Cloudera, you must set the -Ddfs.checksum.combine.mode=COMPOSITE_CRC -Dozone.client.checksum.type=CRC32C parameter to enable the file checksum between the two filesystems because the file-level checksum type (HDFS-13056) is only available in CDH/HDP/Cloudera based on newer Hadoop versions.
- If Kerberos is enabled on the Ozone cluster, add the ozone.om.kerberos.principal.pattern = * parameter.

Example

For example, to distcp from a Kerberized Cloudera HDFS cluster ns1 /tmp directory to a Kerberized Cloudera Ozone cluster ozone1 under volume v1, bucket b1, directory /dst, execute the following command at a destination cluster host:

```
hadoop distcp \
```

```
-Ddfs.checksum.combine.mode=COMPOSITE_CRC \ -Dozone.client.checksum.type=CRC
32C \
-Dozone.om.kerberos.principal.pattern=* \
 hdfs://ns1/tmp/ \
 ofs://ozone1707264383/v1/b1/dst/
```

Related Information

Using Distcp to copy files

Validating the migrated data

You must run specific commands on the source and destination clusters to validate the migrated data from the HDFS cluster.

Procedure

- Considering the example of the user john.doe, run the hdfs du command on the source directory (/user/john.doe/application1) and the target directory (ofs://<ozone.service.id>/user/john.doe/application1).

  Note: The Ozone filesystem (ofs) does not currently return the correct replicated size of a file in the du command response. However, for simple listing of files and raw file sizes, the du output comparison helps in testing the success of disctp.

Cleaning up data on the HDFS source

After validating the data migrated to Ozone, you can clean up the HDFS data depending on your requirements.

(Optional) Configure other services to work with Ozone

If you want services such as Hive and Spark to work with the migrated data in Ozone, you must perform specific configurations.

Procedure

- Hive: To access Hive data on Ozone, you must perform the configurations as explained in the following:
  - Accessing Hive files on Ozone
  - Create an Ozone-based external table
- Spark: To configure Spark to work with Ozone, you must perform the configurations as explained in the following:

  Configuration options for Spark to work with ofs
- Impala: You can use Impala to query data files that reside on Apache Ozone distributed storage, rather than in HDFS

  For more information, see Impala with Ozone