

Managing Your Database

Date published: 2020-08-14

Date modified: 2024-12-18

CLOUDERA

Legal Notice

© Cloudera Inc. 2025. 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

Creating a database using Cloudera Operational Database.....	4
Configuring a database.....	6
Configuring worker nodes.....	6
Configuring edge nodes.....	6
Upgrading a database using Cloudera Operational Database.....	7
Changing database status using Cloudera Operational Database.....	8
Launching hbase shell.....	9
Configuring strong meta servers.....	11
Importing and restoring data into Cloudera Operational Database database.....	13
Backing up a Cloudera Operational Database table.....	14
AWS Graviton instances in Cloudera Operational Database.....	17
Prerequisites.....	17
Creating AWS Graviton-based Cloudera Operational Database clusters.....	17
HBase REST server scaling in Cloudera Operational Database.....	19
Configuring the HBase REST server scaling.....	19
Limitations.....	20
Ensure HBase populates pre-existing object storage data.....	21
Managing custom images in Cloudera Operational Database.....	22
Creating a database using a custom image.....	22
Upgrading a database using a custom image.....	23
Switching image catalogs in Cloudera Operational Database.....	23
Enabling HBase region canary.....	24

Creating a database using Cloudera Operational Database

You can create an operational database in your registered Cloudera environment using the Cloudera Operational Database.

About this task

Required role: You must be logged into the Cloudera Operational Database as an OAdmin.

Watch the video at <https://www.youtube.com/watch?v=Zyok2K23XCs>

Before you begin

- Understand the Cloudera environment and user management. For more information, see *User Management* and *Cloudera Environments* topics.
- Set up an environment that gives you credential and cloud storage. For more information, see *Before you create an operational database cluster*.
- Ensure that you are authorized to create a database.

Procedure

1. In the Cloudera Operational Database web interface, click Create Database.
2. Specify the location of the database where you want to store it.
 - a) Provide a name for the database in the Database Name field.
 - b) Select the Cloudera environment from the list in which you want to associate the database.
 - c) Click Next.

If an environment does not exist, you can create one by clicking Create New Environment.

For more information, see *Register your first environment*.

3. Commission your database by defining a scale for your database using a predefined Data Lake template.

The template helps you to structure your database automatically thereby saving your time and cost. Cloudera Operational Database creates the predefined number of LITE or HEAVY gateway and master nodes, a set of worker nodes, and also adds additional functionalists into the new database. In case you need to modify the default number of nodes defined in the template, you can do so after the database creation.

The available templates are Micro Duty, Light Duty, and Heavy Duty. By default, Light Duty is selected.

You can create a small database using the Micro Duty template, which consists of one Gateway node and one Worker node. In a Micro database, the Gateway node carries out the processes involved in the Master or Leader nodes. You can consider using a Micro cluster for your testing and development purposes.

4. Configure your database by selecting the storage type as Cloud Storage with Caching, Cloud Storage, or HDFS.

- The storage type Cloud Storage with Caching is equivalent to using `--storage-type CLOUD_WITH_EPHEMERAL` option on CDP CLI while creating an operational database.

You can also use this storage to configure time-based priority caching, where data within a specified time range is given a higher priority. In contrast, older data are likely to get evicted. For more information on this storage type, see *HBase Time-based Data Tiering using Persistent BucketCache*

You must have the `COD_DATATIERING` entitlement to use time-based priority caching.

- The storage type Cloud Storage, which resembles block storage, is equivalent to using `--storage-type CLOUD` option on CDP CLI while creating an operational database.
- The storage type HDFS is equivalent to using `--storage-type HDFS` option on CDP CLI while creating an operational database.

By default, Cloud Storage with Caching is selected.

Choose a type of your database setup.

- Choose Default for pre-selected and recommended configurations.
- Choose Custom if you need to modify any of the default values.
 - The HDFS Volume Type option appears only if you select HDFS as the storage type.
 - If you disable the Autoscaling option, the Worker Nodes and Compute Nodes options are hidden. Instead, a Node Count option appears for Worker Nodes.

The minimum and maximum number of worker nodes vary for different storage types.

- Micro duty: Minimum node count: 1. Maximum node count: 5.
- Light duty: Minimum node count: 3. Maximum node count: 100.
- Heavy duty: Minimum node count: 3. Maximum node count: 800.

5. Review the details before creating the database.

Click View CLI Command to get the complete command details corresponding to your settings. You can use it to create the database using CDP CLI.

Alternatively, you can use the following sample command to create the database using CDP CLI.

```
cdp opdb create-database --environment-name cod-7218 --database-name test --scale-type LIGHT --storage-type CLOUD_WITH_EPHEMERAL --auto-scaling-parameters '{"minWorkersForDatabase":5, "maxWorkersForDatabase":100}' --num-edge-nodes 0 --java-version 8
```

6. Click Create Database.

Results

An information page is displayed that shows the status of the database. Your new database is ready to be used once its status becomes Available.



Note: Your database starts with a fixed size; however, it scales up and down as the workload applied to the database changes. For more information, see the *Auto Scaling* topic.

Related Information

[Cloudera Operational Database edge node overview](#)

[Cloudera Operational Database User Management](#)

[Cloudera Environments](#)

[Before you create an operational database cluster](#)

[Register your first environment](#)

[Cloudera Operational Database Auto Scaling](#)

[CDP CLI BETA command reference GitHub repository](#)

Configuring a database

Learn how to configure the properties of an operational database in your Cloudera Public Cloud environment.

After you create an operational database with the specified worker and edge nodes, if you want to change the node counts, you can do so for the existing database, without needing to create a new one.

Configuring worker nodes

Know how to configure an operational database's autoscaling and worker node properties in your environment.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.

Procedure

1. In the Cloudera Operational Database web interface, select the Databases tab.
2. Find the database you want to configure.
- 3.



In the row of the selected database, click (Actions) and choose Configure Database.

4. Modify the settings for Autoscaling and Worker Nodes.

Consider these points while setting the worker nodes.

- Ensure that the number of maximum worker nodes is greater than or equal to the minimum number of worker nodes; otherwise, autoscaling is disabled.

If autoscaling is enabled, the operational database automatically adjusts the maximum and minimum number of worker nodes; however, you can set these node counts to remain in the cluster.

- Non-micro clusters must have at least three worker nodes; while micro clusters can have one minimum worker node.
- The supported value range for different clusters is as follows:

Micro: 1 minimum, 5 maximum

Light: 3 minimum, 100 maximum

Heavy: 3 minimum, 800 maximum

5. Click Configure Database.

Results

The new settings are reflected in the existing database immediately after the modification.

Configuring edge nodes

Know how to configure the edge nodes of an existing operational database in your environment.

About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.

Procedure

1. In the Cloudera Operational Database web interface, select the Databases tab.
2. Find the database you want to configure.
- 3.

In the row of the selected database, click  (Actions) and choose Configure Edge Nodes.

4. You can go to each tab Add or Delete to add or delete a specific number of edge nodes.
You can add a maximum of 20 edge nodes to the cluster.
5. Click Add Edge Nodes or Delete Edge Nodes accordingly.

Results

The new settings are reflected in the existing database immediately after the modification.

Upgrading a database using Cloudera Operational Database

Learn how to upgrade your existing Cloudera Operational Database in the Cloudera environment.

About this task

- You need to use the CDP Beta CLI and run the `upgrade-database` command to upgrade your database.
- Required role: You must be logged into the Cloudera Operational Database as an ODAdmin.
- To use Cloudera Operational Database on a GCP environment, you must do it through CDP CLI with `--use-hdfs` flag.



Important: Before upgrading the Runtime version in your existing Cloudera Operational Database clusters to 7.2.16 or higher versions, you might need to perform some additional steps before upgrading. For more information, see *Data Lake upgrade validations for Python dependency* under the Cloudera Management Console what's new topic and *Upgrading Data Hubs*.

Before you begin

- Understand Cloudera environment and user management. For more information, see *User Management* and *Cloudera Environments* topics.
- Ensure you are authorized to upgrade a database.
- You must download and install the latest CDP CLI beta version.
- Cloudera recommends that you run the `prepare-upgrade-database` command before upgrading the cluster. This command performs all validations and downloads all required parcels for the upgrade operation. For more information, see *prepare-upgrade-database*.

Procedure

1. Log in to the CDP CLI tool.

- Run the following command to prepare for the upgrade. This step is optional.

To reduce the chances of upgrade failures, Cloudera Operational Database supports a preparation phase for the runtime upgrades. During this phase, Cloudera Operational Database runs all the validations and downloads the required parcels for the machines.

```
cdp opdb prepare-upgrade-database --environment <ENVIRONMENT-NAME> --
database <DATABASE-NAME> --runtime <RUNTIME-VERSION> [OR --image-id <IMAGE-ID>]
```

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--runtime <value>	The runtime version to upgrade to.
--image-id <imageId>	The image ID to upgrade to.

- Run the following command to upgrade the database.

This command upgrades an operational database in an environment to a given Runtime:

```
cdp opdb upgrade-database --environment <ENVIRONMENT-NAME> --database
<DATABASE-NAME> --runtime <RUNTIME-VERSION> [--os-upgrade-only | --no-os-
upgrade only]
```

Option	Description
--environment (string)	The name or CRN of the environment.
--database <value>	The name or CRN of the database.
--runtime <value>	The runtime version to upgrade to.
[--os-upgrade-only --no-os-upgrade-only]	Controls whether to perform only an Operating System upgrade.

Related Information

[Data Lake upgrade validations for Python dependency](#)

[Service pack upgrades](#)

[Cloudera Operational Database User Management](#)

[Cloudera Environments](#)

[prepare-upgrade-database](#)

[CDP CLI BETA command reference GitHub repository](#)

Changing database status using Cloudera Operational Database

After you create an operational database, you can start, stop, or drop the database using the Cloudera Operational Database user interface.

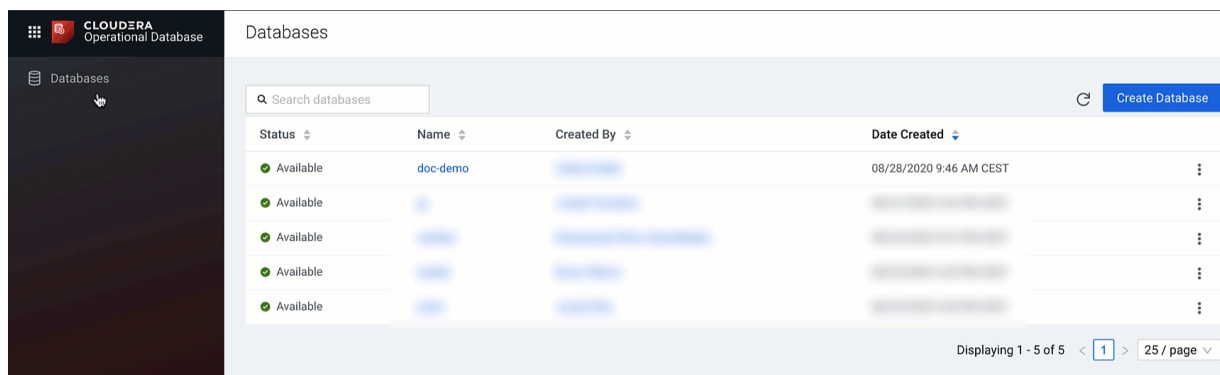
About this task

Required role: You must be logged into the Cloudera Operational Database as an ODAdmin

Procedure

- In the Cloudera Operational Database web interface, select the Databases tab.

2. Find the database you want to manage.
3. In the row of the selected database, click Actions and select one of the following actions:
 - Start Database - Starting the database. When you create a new database, it is initially started by default.
 - Stop Database - Stopping the database and the underlying instances.
 - Drop Database - Dropping the database and deleting all the data that is stored in that database. However, your Cloudera Shared Data Experience Data Lake continues to exist and you can create another database using the same Data Lake.



Related Information

[Start database](#)

[Start database-Beta](#)

[Stop database](#)

[Start database-Beta](#)

[Drop database](#)

[Drop database-Beta](#)

Launching hbase shell

To launch the *hbase shell* and use it with Cloudera Operational Database, you have to create an edge node with a configured HBase client tarball.

Before you begin

Configure an edge node. For more information, see *Cloudera Operational Database edge node overview*.

Procedure

1. In your Cloudera Operational Database web interface navigate to **Connect HBase Client Tarball**.

2. Add your Hbase Client Tarball and HBase Client Configuration.

[Connect](#)
[Charts](#)
[Events](#)
[Diagnostic Bundles](#)
[Snapshots](#)

[HBase](#)
[HBase REST](#)
[HBase Client Tarball](#)
[Phoenix \(Thick\)](#)
[Phoenix \(Thin\)](#)
[Phoenix \(ODBC\)](#)
[Phoenix Python](#)

Usage
You can download the Apache HBase Client Tarball that contains the JAR files used to connect to your database. The HBase Client Tarball contains the necessary scripts and JAR files that you need to connect your database when using interactive tools such as HBase Shell or SQLLine.

HBase Version
2.4.17.7.2.18.800-32

Download URL
<https://cod-1fsu9a0nhmr2-gateway0.alim-mow.xcu2-8y8x.dev.cldr.work/cod-1fsu9a0nhmr2/cdp-proxy-api/hbase/jars/hbase-client-tarball.tar.gz>

HBase Client Configuration URL
`curl -f -o "hbase-config.zip" -u "csso_" "https://cod-1fsu9a0nhmr2-gateway0.alim-mow.xcu2-8y8x.dev.cldr.work/clouderamanager/api/v41/clusters/cod-1fsu9a0nhmr2/`

[> Kerberos Configuration](#)
[> Yarn Configuration](#)
[> JWT Configuration](#)

- Download hbase-client-tarball.tar.gz and extract it to a location on the edge node. This is your *"HBASE_HOME"*.
- Download the client configuration using the curl command.
- Extract the client configuration zip file, and move the contained hbase-conf directory to *"HBASE_HOME"* with the name conf (instead of hbase-conf).

3. Launch the hbase shell.

```
$ kinit [***CDP_WORKLOAD_USER***]
Password: *****
$ export HBASE_HOME=hbase-2...
$ cd $HBASE_HOME
$ ./bin/hbase shell
```

4. Validate you can connect through hbase shell.

Use the list command in hbase shell to list all tables.

5. Add yarn configuration to hbase configuration.

Yarn Configuration

Additional configuration required to submit YARN applications.

YARN Archive

```
curl -f -o "yarn-config.zip" -u "csso_" "https://cod-1fsu9a0nhmr2-gateway0.alim-mow.xcu2-8v8x.dev.cldr.work/clouderamanager/api/v41/clusters/cod-1fsu9a0nhmr"
```

TLS Truststore

```
curl -f -o "cod-truststore.jks" -u "csso_..." "https://cod-1fsu9a0nhmr2-gateway0.alim-mow.xcu2-8y8x.dev.cldr.work/clouderamanager/api/v41/certs/truststore?type=JL"
```

TLS Truststore Password

```
curl -u "csso_ " "https://cod-1fsu9a0nhmr2-gateway0.alim-mow.xcu2-8y8x.dev.cldr.work/clouderamanager/api/v41/certs/truststorePassword"
```

- Navigate to `Connect HBase Client Tarball` and expand the `YARN Configuration` section..
- Download the `yarn-config.zip` file.
- Add the files from `yarn-conf` to `"HBASE_HOME"/conf`.
- Download the TLS truststore and remember where you place it.
- Update the `ssl-client.xml` file with the TLS TRuststore details.
- Ensure that in the `ssl-client.xml` file the `ssl.client.truststore.location` points to the `cod-truststore.jks` you have downloaded in step d.
- Ensure that the `ssl.client.truststore.password` matches the TLS Truststore Password.



Note: If you plan to use the `hbck2` command on Cloudera Operational Database client side, set the `HBAS E CLASSPATH` environment variable pointing to the HBase client tarball lib directory as follows:

```
export HBASE_CLASSPATH=/HBase\_client\_tarball\_location/hbase-client-  
tarball/lib/:
```

Results

You have an edge node with configured HBase client tarball and you can launch the hbase shell.

Related Information

Cloudera Operational Database edge node overview

Configuring strong meta servers

The strong meta feature uses an HBase RegionServerGroup (RSGroup) to provide dedicated servers to the HBase system tables. This prevents customer table regions from being hosted on the same RegionServers as the system regions and prevents the load from these custom tables from affecting system table access and initialization.

Before you begin

- You must have the `COD_STRONG_META_SERVERS` entitlement to use this feature.
- This feature is only supported on AWS environments.
- By default, meta replication is enabled and the replica number is set to 3. Therefore, ensure that each strong meta node contains at least 3 meta regions. This does not require any manual steps.

Procedure

1. Launch the CDP CLI tool.

2. Run the following command to add the desired number of strong meta servers.

```
cdp opdb update-database --environment-name envName --database-name dbName
--num-desired-strong-meta-servers 3
```



Note: The minimum number of strong meta servers to be added is 3 because the meta replication is already enabled in Cloudera Operational Database and the minimum number of supported meta regions is 3.

The above command creates 3 strong meta nodes, adds them to the cluster, and puts a region server on them.

3. To verify if the nodes are healthy, ensure that the logs on these nodes have no errors or exceptions. If data on the cluster and balancer is enabled, these nodes must serve random regions because the balancer moves regions there.
4. Ensure that the RSGroup feature is enabled on the Cloudera Operational Database cluster. To enable this, use the `list_rsgroups` command in the HBase shell.

If the command is not enabled, the following error is displayed.

```
org.apache.hadoop.hbase.exceptions.UnknownProtocolException: No registered
Master Coprocessor Endpoint found for RSGroupAdminService. Has it been en
abled?
```

To enable the RSGroup feature, perform the following steps:

- a) If the `hbase.master.loadbalancer.class` property is defined, copy its value to the `hbase.rsgroup.grouploadbalancer.class` property in the `hbase-site.xml` file.
 - b) Set the value of `hbase.master.loadbalancer.class` property to `org.apache.hadoop.hbase.rsgroup.RSGroupBasedLoadBalancer` in the `hbase-site.xml` file.
 - c) Set the value of `hbase.coprocessor.master.classes` property to `org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint` in the `hbase-site.xml` file.
 - d) Restart the master nodes for these changes to take effect.
 - e) To verify if the setup was successful, use the `list_rsgroups` command in the HBase shell.
5. Create a `RegionServerGroup` and add the strong meta nodes to it. Ensure that the system tables are also added. Execute the following steps in an HBase shell.

```
// verify that the feature works and every worker/strongmeta nodes are v
isible
list_rsgroups
//create the group (please note, that a different name can be also used)
add_rsgroup "strongmeta"
//verify that the group is created
list_rsgroups
//move the strongmeta nodes to the group (node names can be copied from
the list_rsgroups output)
move_servers_rsgroup "strongmeta",["cod--xxx-strongmeta0.cod-7216.xcu2-8y8
x.dev.cldr.work:16020","cod--xxx-strongmeta1.cod-7216.xcu2-8y8x.dev.cldr
.work:16020","cod--xxx-strongmeta2.cod-7216.xcu2-8y8x.dev.cldr.work:1602
0"]
//verify that the nodes are added to the group
list_rsgroups
//move the system tables to the group
move_tables_rsgroup "strongmeta",["hbase:meta"]
move_tables_rsgroup "strongmeta",["hbase:acl"]
move_tables_rsgroup "strongmeta",["hbase:namespace"]
move_tables_rsgroup "strongmeta",["hbase:rsgroup"]
//verify that the tables were moved
list_rsgroups
```

6. If the previous steps are successful, ensure that everything is functioning as expected using the HBase UI.

What to do next

Ensure that each strong meta node now only serves system table regions.

Importing and restoring data into Cloudera Operational Database database

You can import your data into your Cloudera Operational Database by restoring your HBase table into Cloudera Operational Database.

Before you begin

- Enable HBase replication on your Cloudera Operational Database cluster. For more information, see *Cloudera Operational Database data replication*.
- Have a location in cloud storage (for example, S3 or ABFS) with an exported snapshot in it, and have the name of the snapshot.

If you do not already have an exported HBase snapshot, you can export your data to cloud storage using the following command:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snaps
hot [***SNAPSHOT NAME***] -copy-to [***CLOUD STORAGE LOCATION***] -mappers
10
```

For example, the data-from-onprem snapshot can be exported into s3a://cod-external-bucket/hbase:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot data-from-
onprem -copy-to s3a://cod-external-bucket/hbase -mappers 10
```

- Have an edge node with a configured HBase client tarball and know how to launch the hbase shell from it. For more information, see *Launching HBase shell*.

Procedure

1. Get your CLOUD STORAGE LOCATION for your Cloudera Operational Database using the Cloudera Operational Database web interface.

s3a://my_cod_bucket/cod-12345/hbase

Databases / doc-test

The screenshot shows the Cloudera Operational Database web interface for a database named 'doc-test'. The interface includes a status bar at the top indicating 'Available - Updated just now' and an 'Actions' dropdown. Below this, the database's unique identifier is displayed. A table lists various attributes: ENVIRONMENT (cod-727-newsbtnets), REGION (us-west-2), DATA LAKE (cod-727-newsbtnets), SQL EDITOR (Hue), and CLOUD STORAGE LOCATION (s3a://cod-727-mowdev/cod-9zouq3ua3qqz/hbase). The 'CLOUD STORAGE LOCATION' field is highlighted with an orange border.

2. Add your bucket to the IAM policy used by IDBroker.

For more information, see one of the following documentation:

- [AWS Environments: Minimal setup for cloud storage](#)
- [Azure Environment: Minimal setup for cloud storage](#)

3. Launch the hbase shell from the edge node.

For more information, see *Launching HBase shell*.

4. From the edge node, run the ExportSnapshot command. Use the external bucket as the source location and the Cloudera Operational Database cloud storage as the target.

For example:

```
$ cd $HBASE_HOME
$ ./bin/hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot
  "data-from-onprem" --copy-from s3a://cod-external-bucket/hbase --copy-to
  s3a://my_cod_bucket/cod-12345/hbase
```

5. Use the list_snapshots command and verify that your snapshot is listed.

```
$ cd $HBASE_HOME
$ ./bin/hbase shell
$ hbase> list_snapshots
[ 'snapshot_name' ]
```

6. Use the restore_snapshot or the clone_snapshot command to reconstitute the table.



Warning: The restore_snapshot command overwrites an existing table.

- restore_snapshot: Overwrites current table state with that of the snapshot. It means that any data modification applied after the snapshot was taken would be lost. If the table does not exist in the given cluster, the command automatically creates it.
- clone_snapshot: Accepts a new table name for the table in which it restores the table schema and data.

7. Validate that all rows are present in the table using the hbase rowcounter or count command in the hbase shell.

Related Information

[Launching hbase shell](#)

[Cloudera Operational Database data replication](#)

[Using the CldrCopyTable utility to copy data](#)

Backing up a Cloudera Operational Database table

You can backup your Cloudera Operational Database table by creating a snapshot of the table and exporting it to your S3, HDFS, or ABFS bucket.

About this task

Create snapshot command takes a snapshot of the HBase table, exports the snapshot to the specified storage location, and deletes the snapshot from HBase after the export to the specified storage location is successful.

Before you begin

- Ensure that you have access to the location where you want to take the backup of the table.
- You have an S3 or ABFS bucket which you want to use to backup an HBase table in Cloudera Operational Database. For example: s3a://cod-backups/hbase
- You have an edge node configured for your Cloudera Operational Database database. For more information, see *Configure Edge Nodes*.

Procedure

1. Run the following command using `cdplici-beta`.

```
cdp opdb create-snapshot --environment-name ENVIRONMENT_NAME --database-name DATABASE_NAME --table-name TABLE_NAME --snapshot-name SNAPSHOT_NAME --snapshot-location SNAPSHOT_LOCATION
```

For example,

```
cdp opdb create-snapshot --environment-name odx-wgel6j --database-name jpfy --table-name table_6uf7xt --snapshot-name s1 --snapshot-location s3a://cod-7215/cod--yq3p370r6qro/backupdata
```

JSON output:

```
{
  "environmentName": "odx-wgel6j",
  "databaseName": "jpfy",
  "status": "IN_PROGRESS",
  "creationTime": 0,
  "commandID": 1546336350
}
```

2. Validate that the snapshot and data is present using the `list-snapshots` command.

The following command lists the snapshots created in the environment against the Cloudera Operational Database database filters like table name, command ID, and time range.

```
cdp opdb list-snapshots --environment-name ENVIRONMENT_NAME --database-name DATABASE_NAME [--table-name TABLE_NAME] [--command-id COMMAND_ID] [--from-creation-time FROM_CREATION_TIME] [--to-creation-time TO_CREATION_TIME]
```

Examples

Without filters

```
cdp opdb list-snapshots --environment-name odx-wgel6j --database-name jpfy
{
  "environmentName": "odx-wgel6j",
  "databaseName": "jpfy",
  "snapshots": [
    {
      "tableName": "table_tob8vc",
      "snapshotName": "snapshot3_table_tob8vc",
      "creationTime": 1662014265011,
      "status": "SUCCESSFUL",
      "commandID": 1546336058,
      "snapshotLocation": "abfs://qedailynat-filesystem@qedailynatstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
    },
    {
      "tableName": "table_tob8vc",
      "snapshotName": "snapshot2_table_tob8vc",
      "creationTime": 1662014023725,
      "status": "SUCCESSFUL",
      "commandID": 1546336035,
      "snapshotLocation": "abfs://qedailynat-filesystem@qedailynatstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
    },
    {
      "tableName": "table_tob8vc",
      "snapshotName": "snapshot1_table_tob8vc",
      "creationTime": 1662013782370,
```

```

        "status": "DELETED",
        "commandID": 1546336077,
        "snapshotLocation": "abfs://qedailynat-filessystem@qedailyna
tstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
    },
    {
        "tableName": "table_lpirwy",
        "snapshotName": "snapshot_table_lpirwy",
        "creationTime": 1662013568312,
        "status": "DELETED",
        "commandID": 1546336004,
        "snapshotLocation": "abfs://qedailynat-filessystem@qedailyna
tstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase"
    }
]
}

```

With command ID filter

```

$ cdp opdb list-snapshots --environment-name odx-wgel6j --database-name
jpfy --command-id 1546336385
{
  "environmentName": "odx-wgel6j",
  "databaseName": "jpfy",
  "snapshots": [
    {
      "tableName": "table_6uf7xt",
      "snapshotName": "s2",
      "creationTime": 1662035967244,
      "status": "SUCCESSFUL",
      "commandID": 1546336385,
      "snapshotLocation": "abfs://qedailynat-filessystem@qedailyna
tstorageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase/"
    }
  ]
}

```

With table name filter

```

$ cdp opdb list-snapshots --environment-name odx-wgel6j --database-name
jpfy --table-name table_6uf7xt
{
  "environmentName": "odx-wgel6j",
  "databaseName": "jpfy",
  "snapshots": [
    {
      "tableName": "table_6uf7xt",
      "snapshotName": "s2",
      "creationTime": 0,
      "status": "SUCCESSFUL",
      "commandID": 1546336385,
      "snapshotLocation": "abfs://qedailynat-filessystem@qedailynats
torageaccount.dfs.core.windows.net/cod-rtzupxh843ua/hbase/"
    }
  ]
}

```

Related Information

[Launching hbase shell](#)

[Create snapshot](#)

[List snapshot](#)

AWS Graviton instances in Cloudera Operational Database

Know how to create and use AWS Graviton-based clusters in your Cloudera Operational Database environment.

AWS Graviton is a versatile ARM-based processor family designed to deliver exceptional price-to-performance efficiency for cloud workloads hosted on AWS Elastic Compute Cloud (EC2). By leveraging AWS Graviton, businesses can significantly optimize their operational costs.

Considerations for using AWS Graviton in your cluster

- You can create the Graviton-based Cloudera Operational Database clusters only on the AWS environments.
- The supported storage type is S3 with ephemeral cache enabled.
- The supported scale type is HEAVY.
- AWS is still rolling out Graviton, this feature is only available in some regions.
- Deploying Cloudera Operational Database on Graviton might take longer based on the cluster shape and size.

Prerequisites

Learn about the prerequisites for deploying Cloudera Operational Database clusters in AWS Graviton environments.

- You must be logged into the Cloudera Operational Database as an administrator.
- Verify if the selected instance types are available in your preferred regions.
- To use Cloudera Operational Database on Graviton, you need to install beta CDP CLI. For more information, see the *Installing Beta CDP CLI*.

Related Information

[Installing Beta CDP CLI](#)

Creating AWS Graviton-based Cloudera Operational Database clusters

Learn how to create AWS Graviton-based Cloudera Operational Database clusters using the Cloudera Operational Database UI or CDP CLI command.

About this task

You can create AWS Graviton-based Cloudera Operational Database clusters using the Cloudera Operational Database UI or by executing CDP CLI commands.

Procedure

1. In the Cloudera Operational Database web interface, click Create Database.
2. Specify the location of the database where you want to store it.
 - a) Provide a name for the database in the Database Name field.
 - b) Select the Cloudera environment with which you want to associate the database. The environment does not have to be Graviton-based; it must use a Cloudera version that supports Cloudera Operational Database on Graviton.
 - c) Click Next.

If an environment does not exist, you can create one by clicking Create New Environment.

For more information, see *Register your first environment*.

3. Select ARM64 as the Architecture from the drop-down list.

4. Commission your database by defining a scale for your database using a predefined Data Lake template. Select Heavy Duty.

The template helps you to structure your database automatically thereby saving your time and cost. Cloudera Operational Database creates the predefined number of HEAVY gateway and master nodes, a set of worker nodes, and also adds additional functionalists into the new database. In case you need to modify the default number of nodes defined in the template, you can do so after the database creation.

The available templates are Micro Duty, Light Duty, and Heavy Duty. By default, Light Duty is selected.

5. Configure your database by selecting the storage type as Cloud Storage with Caching.

The storage type Cloud Storage with Caching is equivalent to using `--storage-type CLOUD_WITH_EPHEMERAL` option on CDP CLI while creating an operational database.

6. Check or update the settings for your database.

- a) Check all the default settings for your database under the Default tab.
- b) Go to the Advanced tab if you need to modify any of the default values.
 - The HDFS Volume Type option appears under the Advanced tab only if you select HDFS as the storage type in the Configuration step.
 - If you disable the Autoscaling option using the Advanced tab, the Worker Nodes and Compute Nodes options are hidden. Instead, a Node Count option appears.

The minimum and maximum number of worker nodes vary for different storage types.

- Micro duty: Minimum node count: 1. Maximum node count: 5.
- Light duty: Minimum node count: 3. Maximum node count: 100.
- Heavy duty: Minimum node count: 3. Maximum node count: 800.

7. Review the details before creating the database.

Click Show CLI Command to get the complete command details corresponding to your settings. You can use it to create the database using CDP CLI.

Alternatively, you can use the following sample command to create the database using CDP CLI.

```
cdp opdb create-database --environment-name cod-731 --database-name test
--architecture ARM64 --scale-type HEAVY --storage-type CLOUD_WITH_EPHEMERAL
--auto-scaling-parameters '{"minWorkersForDatabase":5, "maxWorkersForDatabase":100}' --num-edge-nodes 0
```

8. Click Create Database.

Alternatively, you can use the CDP CLI to create the Cloudera Operational Database.

- a. Launch the beta CDP CLI tool.
- b. Run the following command to create Graviton-based Cloudera Operational Database clusters.

```
cdp opdb create-database --environment-name [***ENV_NAME***] --database-name
[***DATABASE_NAME***] --architecture [***STRING***] --storage-type CLOUD_WITH_EPHEMERAL
--scale-type HEAVY
```

For example,

```
cdp opdb create-database --environment-name aws_test --database-name cod_7214
--architecture ARM64 --storage-type CLOUD_WITH_EPHEMERAL --scale-type HEAVY
```

On the Cloudera Operational Database UI, an information page is displayed that shows the status of the database. Your new database is ready to be used once its status becomes Available.

Related Information

[Register your first environment](#)

[CDP CLI BETA command reference guide](#)

HBase REST server scaling in Cloudera Operational Database

Using Apache HBase REST API, you can scale up the HBase REST server for better connectivity to Cloudera Operational Database.



Important: To use this feature in your Cloudera environment, you must have the COD_RESTWORKERS entitlement enabled. Contact Cloudera Support or your Cloudera account team if you do not have this entitlement.

Multiple Knox Gateway servers are required to scale up the HBase REST servers and support the increased load, so you must have multiple gateway nodes. When you define multiple gateway nodes, the Apache Knox Gateway instances work in an HA mode, and the load is balanced among the multiple Knox instances. Each gateway node hosts an Apache Knox Gateway instance that provides access to the HBase REST server hosted on a REST worker node. The maximum number of gateway nodes supported is 10.

You can specify the required number of REST worker nodes using the `--restworker-nodes-count` option in the `create-database` command. This optional parameter can only be defined when you specify the `--gateway-nodes-count` option. Using the `update-database` command, you can scale up the number of the REST worker nodes after database creation. The default number of REST worker nodes supported is 0.

The recommended Cloudera Operational Database deployment type is Single-AZ to leverage the scaling of HBase REST servers.

Benefits of using a REST server

- When a direct line of sight between the client and all HBase servers cannot be established, you can connect the client to a Cloudera Operational Database-deployed load balancer backed by Knox and REST servers, allowing seamless connectivity with Cloudera Operational Database.
- This setup can leverage the HTTP authentication proxy, HTTP - HTTPS proxy, and OAuth proxy features of Knox. For example, Knox can act as an HTTP authentication proxy to support JWT for REST servers, with the flexibility to integrate other authentication providers as needed.
- This setup is language-independent, which means you can use any client, regardless of the programming language, to interact with Cloudera Operational Database.

Configuring the HBase REST server scaling

Know how to configure an HBase REST server scaling.

About this task

When you configure an HBase REST server:

- The heap of Knox is increased to 31 GiB on HEAVY and 2 GiB on LIGHT scale Cloudera Operational Database clusters respectively.
- The Knox parameter `gateway.httpsrequestHeaderBuffer` is set to 1 MiB as a safety valve configuration in Cloudera Manager. For more information on this parameter, see the *Apache Knox documentation*.
- The heap size of the HBase REST server is set to 16 GiB on REST worker nodes.



Note: Cloudera recommends exercising caution while using the gateway nodes because these instance types are expensive. If you wish to save costs, use them judiciously.

Before you begin

- To utilize this functionality, you need at least two gateway nodes. The required number of gateway nodes can be specified using the `--gateway-nodes-count` option in the `create-database` command.
- To enable this functionality, you must create at least two gateway nodes.
- You must have the ODAAdmin rights to make changes to the Cloudera Operational Database database.
- You must download and install the latest CDP CLI beta version. For more information, see *Installing Beta CDP CLI*.
- Cloudera recommends using Java 11 to create the clusters. Java 8 has a slow TLS implementation, which significantly reduces the HTTPS throughput of both the REST server and Knox.

Procedure

1. Launch the CDP CLI tool.
2. Create the database with the desired scale type and number of gateway nodes using the `--gateway-nodes-count` option.



Important: In the case of the HEAVY scale type, the Gateway node is order of magnitude stronger than that of the LIGHT scale type (32 vCPU and 256 GiB memory vs 8 vCPU 32 GiB memory), and the allocated heap for the hosted Knox is 31 GiB. Thus the gateway node can handle a much higher load with fewer node numbers. You must consider this while planning the number of gateway nodes.

```
cdp opdb create-database --environment-name <ENV_NAME> --database-name
<DATABASE_NAME> --scale-type HEAVY --gateway-nodes-count INTEGER --disable-multi-az --java-
version 11
```

Optionally, you can specify the number of REST worker nodes using the `--restworker-nodes-count` option.

```
cdp opdb create-database --environment-name <ENV_NAME> --database-name
<DATABASE_NAME> --gateway-nodes-count <INTEGER> --restworker-nodes-count <INTEGER> --
disable-multi-az --java-version 11
```

What to do next

If you prefer to scale up or down an HBase REST server, use the following `update-database` command.



Note: The REST server is CPU-bound, if the CPU utilization is too high, you must increase the number of nodes.

```
cdp opdb update-database --environment-name <ENV_NAME> --database-name
<DATABASE_NAME> --num-desired-restworker-nodes <INTEGER>
```

For example,

```
cdp opdb update-database --environment-name cod-env --database-name cod-db -
-num-desired-restworker-nodes 4
```

Use the following command to scale down an HBase REST server to zero instances.

```
cdp opdb update-database --environment-name <ENV_NAME> --database-name
<DATABASE_NAME> --remove-restworker-nodes
```

Related Information

[Installing Beta CDP CLI](#)

[Apache Knox documentation](#)

Limitations

Know the limitations and known issues while scaling an HBase REST server.

Limitations

- The number of gateway nodes cannot be modified after the database is created, so plan the number of requested gateway nodes based on the estimated load.
- You can scale the HBase REST servers in Cloudera Operational Database clusters deployed in Amazon Web Services (AWS) environments.
- The network address translation (NAT) could interfere with traffic distribution because the current setup relies on a Network Load Balancer for sticky sessions, which may cause all requests to route to a single Knox server. Future Cloudera Operational Database versions plan to use an Application Load Balancer for more effective distribution based on session cookies. Until then, NAT must be avoided if more than one Knox or REST server is in use.
- Knox nodes must be specified at the time of cluster creation, as scaling Knox nodes after deployment is currently not supported.
- A Multi-AZ setup can increase data transfer costs, as data might need to pass through multiple services, such as Knox, REST, and HBase region servers across different availability zones before reaching the client.
- HBase Client Tarball downloads might fail if Knox re-routes the request to the wrong gateway.
- The MICRO scale type does not support multiple Knox instances and scaling of HBase REST servers.
- Changing the Knox Provider Configurations and Descriptors using the Knox Admin UI only changes the provider configurations, descriptors, and the generated topologies of the Knox instance whose Admin UI is used.

Known issues

LB based client configs and tarball URLs in the describe-client-connectivity command output fails intermittently

Due to a known Cloudbreak issue, CB-26030, downloading an HBase client configuration intermittently fails when using multiple Gateway nodes.

Retry the download operation until it succeeds.

describe-database command returns incorrect output.

After creating the Cloudera Operational Database cluster with the required number of gateway and REST worker nodes, when you run the describe-database command, the output always shows 0 number of gateway and REST worker nodes.

Upgrade to the latest Cloudera Operational Database version.

Rolling operating system upgrade does not function correctly

When you try to perform a rolling upgrade on the Cloudera Operational Database cluster, it does not work with multiple gateway and REST worker nodes.

Upgrade to the latest Cloudera Operational Database version.

Ensure HBase populates pre-existing object storage data

Cloudera Operational Database databases store data in cloud storage. The location of the data is stored in cloud storage is defined by the data access cloud storage location in the Cloudera environment and the Cloudera Operational Database database name .

When a Cloudera Operational Database database is dropped, any data stored in the cloud storage is not dropped. You must delete the data separately if you no longer need it. If a Cloudera Operational Database database is created against a cloud storage location which already contains Cloudera Operational Database data from a previous instance of Cloudera Operational Database, Cloudera Operational Database will automatically re-create the tables with the corresponding data in the new Cloudera Operational Database instance. Multiple Cloudera Operational Database databases must not refer to the same cloud storage location as this will cause data loss and data corruption.

If a Cloudera Operational Database database is created using the same name against a cloud data access storage location, which already contains Cloudera Operational Database data from a previous instance of Cloudera

Operational Database, Cloudera Operational Database will automatically re-create the tables with the corresponding data in the new Cloudera Operational Database instance.

Managing custom images in Cloudera Operational Database

You can customize a default image for compliance or security reasons. You can then use the CDP CLI to register a custom image catalog and set the custom image within the custom image catalog. Later, you can use this custom image to create an operational database.

To know more about the custom images, see *Custom images and image catalogs*.

Related Information

[Custom images and image catalogs](#)

Creating a database using a custom image

Cloudera Operational Database allows you to create a database using a custom image for compliance or security purposes. You can inherit pre-installed packages or software libraries from the custom image while creating an operational database.

Before you begin

- You must download and install the latest CDP CLI beta version.
- You must have a working Cloudera environment.
- You must ensure that the custom image resides in your Cloudera environment.

Procedure

1. Run the following command to create a database using a custom image.

```
cdp opdb create-database --environment <ENV_NAME> --database <DATABASE-NAME>
--image <"id": <IMAGE_ID>, "catalog": <CATALOG_NAME>>
```

Parameter value	Description
<env_name>	Name of the environment that holds the new database.
<database-name>	Name of the new database for which you want to use the custom image.
<image_id>	Image ID of the database image.
<catalog_name>	Name of the database image catalog.

For example,

```
cdp opdb create-database --environment-name cod-7215 --database-name tes
tdb --image '{"id": "78aeac7f-4a37-4395-a270-833d655c20fb", "catalog": "cdp-
default"}'
```

2. Verify that the database creation is successful for the same image.

Navigate to the Data Hub Clusters service or to the Cloudera Management Console Data Hub Clusters , and click the Image Details tab of the newly created Cloudera Data Hub cluster.

Upgrading a database using a custom image

You can upgrade an existing database using a custom image. The database inherits the packages and software libraries of the custom image, and is upgraded to the provided database image.

Before you begin

- You must download and install the latest CDP CLI beta version.
- You must have a working Cloudera environment.
- You must ensure that the custom image resides in your Cloudera environment.

Procedure

1. Run the following commands sequentially to upgrade a database using a custom image.

- a. `cdp opdb upgrade-database --environment <ENV_NAME> --database <DATABASE_NAME> --image-id=string`
- b. `cdp opdb upgrade-database --environment <ENV_NAME> --database <DATABASE_NAME> --image-id=string --os-upgrade-only`

Parameter value	Description
<env_name>	Name of the environment that holds the existing database.
<database_name>	Name of the existing database for which you want to use the custom image.
--image-id	Image ID of the database image to which the existing database is upgraded.
--os-upgrade-only	Only perform an Operating System upgrade.

For example,

```
cdp opdb upgrade-database --environment cod-7216-micro3 --database tempd
b12 --image-id a188a3d7-067f-42f1-9b89-16d40faab24c
```

```
cdp opdb upgrade-database --environment cod-7216-micro3 --database tempd
b12 --image-id a188a3d7-067f-42f1-9b89-16d40faab24c --os-upgrade-only
```

2. Verify that the database upgrade is successful for the same image.

Navigate to the Data Hub Clusters service or to the Cloudera Management Console Data Hub Clusters , and click the Image Details tab of the upgraded Cloudera Data Hub cluster.

Switching image catalogs in Cloudera Operational Database

You can switch the image catalog of an existing operational database. You may want to switch the image catalog for a database to restrict which Runtime version can be upgraded to, or to move to custom images for an existing database.

Before you begin

- You must download and install the latest CDP CLI beta version.
- You must have a working Cloudera environment.
- You must ensure that the custom image resides in your Cloudera environment.

Procedure

1. Run the following command to switch an image catalog of an existing database.

```
cdp opdb update-database --environment <ENV_NAME> --database <DATABASE_NAME>
--catalog <CATALOG_NAME>
```

Parameter value	Description
<env_name>	Name of the environment that holds the existing database.
<database_name>	Name of the existing database for which you want to switch the image catalog.
<catalog_name>	Name of the image catalog that you want to update for the database.

For example,

```
cdp opdb update-database --environment-name cod-7216-micro2 --database-name testdb --catalog v3dev-1
```

2. Verify that the image catalog switching is successful for the database.

Navigate to the Data Hub Clusters service or to the Cloudera Management Console Data Hub Clusters , and click the Image Details tab of the updated Cloudera Data Hub cluster.

Enabling HBase region canary

You can enable the HBase canary, which is an optional service, to monitor the health of the HBase RegionServer.

About this task

Cloudera Operational Database provides the CLI option `--enable-region-canary` to add the HBase canaries to monitor the RegionServer health. You can use this option while creating an operational database using CDP CLI.

The HBase canaries are added under the HBase service in the Cloudera Manager. You can view them under Cloudera Manager HBase Service Configuration tab .

Before you begin

Ensure that you have created a Cloudera environment and is ready to use it.

Procedure

1. Log in to the terminal that has the CDP CLI client installed.
2. Use the following command to enable the HBase region canaries.

```
cdp opdb create-database --environment-name ENVIRONMENT_NAME --database-name DATABASE_NAME --enable-region-canary
```

For example,

```
cdp opdb create-database --environment-name cdp_7215 --database-name cod_1 --enable-region-canary
```

Results

The following HBase canaries are added on the Cloudera Manager UI.

- `hbase_region_health_canary_enabled` (HBase Region Health Canary)
- `hbase_region_health_canary_slow_run_alert_enabled` (HBase Region Health Canary Slow Run Alert Enabled)
- `hbase_canary_alert_unhealthy_region_percent_threshold` (HBase Canary Unhealthy Region Percentage Alert Threshold)

Related Information[Create database](#)