Cloudera Operational Database ..

Top Tasks

Date published: 2022-09-27
Date modified: 2023-01-12

https://docs.cloudera.com/

Legal Notice

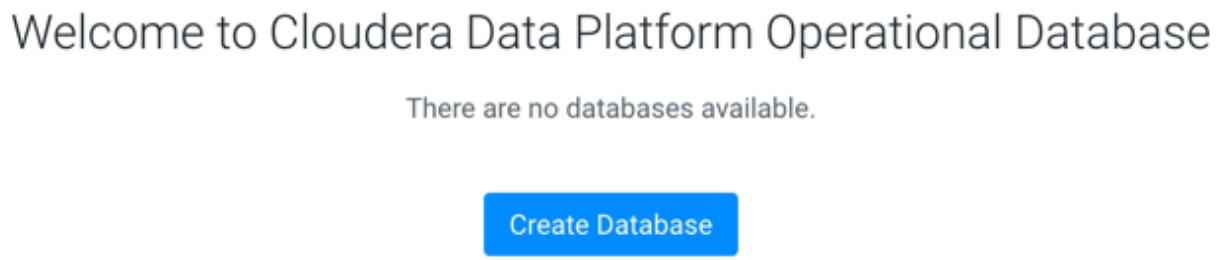
Contents

Creating a database using COD

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

About this task

Required role: You must be logged into the COD as an ODAdmin.



Before you begin

- Understand the CDP environment and user management. For more information, see *User Management* and *CDP 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 COD 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 CDP 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. COD 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 with Caching and Data Tiering, 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.
   - The storage type Cloud Storage with Caching and Data Tiering resembles cloud storage with 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 <cite>HBase Time-based Data Tiering using Persistent BucketCache</cite>.

     You must have the COD_DATATIERING entitlement to be able to use this storage type.
   - 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.
5. 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.
6. 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-7218 --database-name tes
   t --scale-type LIGHT --storage-type CLOUD_WITH_EPHEMERAL --auto-scaling-
   parameters '{"minWorkersForDatabase":5, "maxWorkersForDatabase":100}' --
   num-edge-nodes 0 --java-version 8
   ```
7. 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

COD Edge Node Overview
COD User Management
CDP Environments
COD Auto Scaling
COD CLI command reference GitHub repository
CDP CLI BETA command reference GitHub repository

Creating a database using CDP CLI

You can create an operational database in your registered environment using Cloudera Operational Database (COD) CLI Beta.

Before you begin

- You must download and install the latest CDP CLI beta version.
- Understand CDP environment and user management. For more information, see *User Management* and *CDP Environments* topics.
- Set up an environment that gives you credential and cloud storage.
- Ensure you are authorized to create a database.

About this task

- Required role: You must be logged into the COD as an ODAdmin.
- You can create a COD database in an AWS, GCP, or Azure environment.

Procedure

1. Log in to the CDP CLI beta tool.
2. Enter the following command:

   cdp opdb create-database --environment-name cod7215 --database-name test123

   Pass the --use-hdfs flag to create a database for COD using HDFS storage.

   If you pass --no-use-hdfs flag or do not pass any flag, the database for COD is created using cloud storage. For example, Amazon S3, GCS, or ABFS.

Results

Your new database is ready for use.

Related Information

COD User Management

CDP Environments

Deploying applications on Cloudera Operational Database

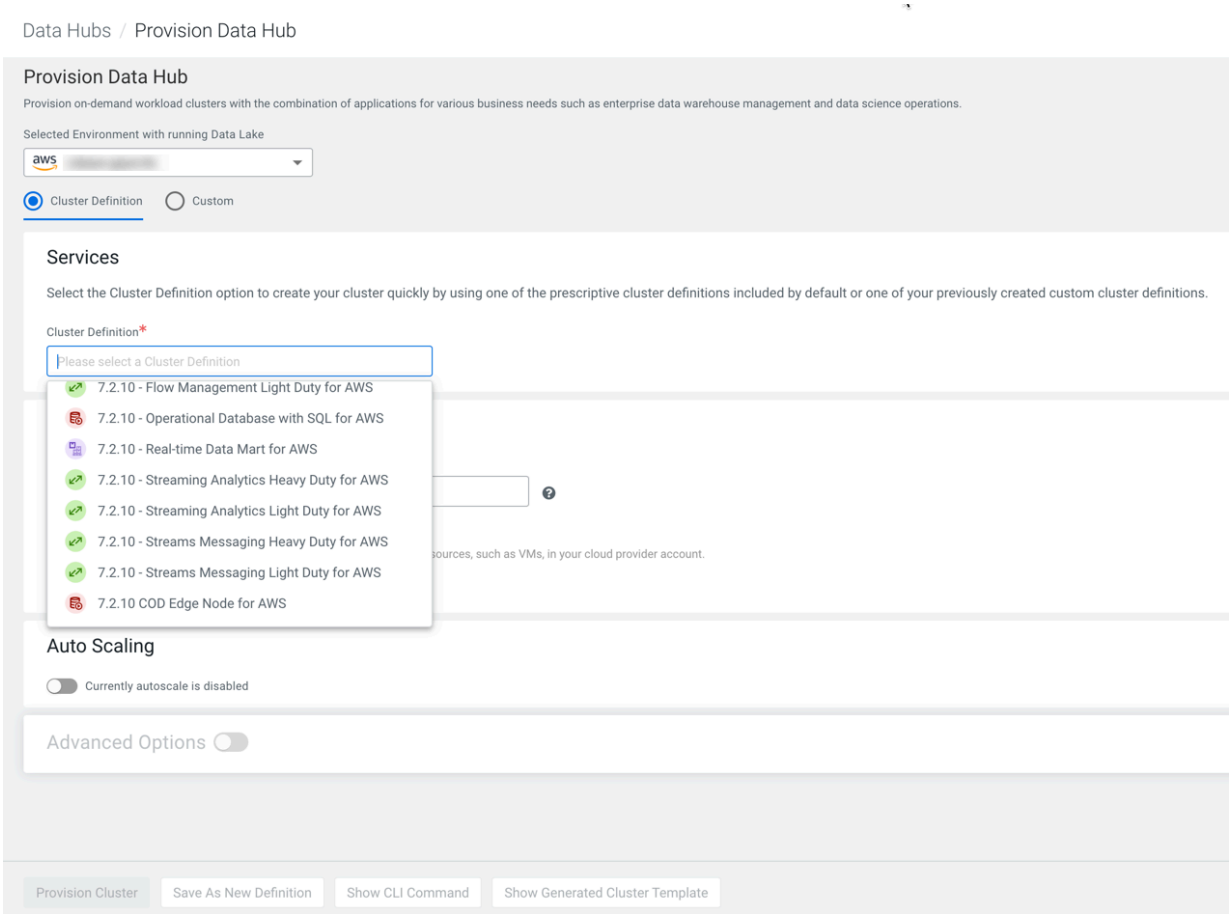
The edge node is a dedicated Data Hub cluster that enables you to communicate with your Cloudera Operational Database (COD) instance and your applications. You can deploy a cluster that works as an edge node to access your COD instance. Deploy the edge node cluster in the same environment as the COD instance to ensure that the security groups and data ingress rules that apply to the COD instance must also apply to the edge node cluster.

Procedure

1. From the Cloudera Management Console, click Data Hub Clusters.
2. Click Create Data Hub.
3. In the Selected Environment with running Data Lake drop-down list, select the same environment used by your COD instance.
4. Select the Cluster Definition.

5. In the Cluster Definition drop-down list, select the [***RUNTIME VERSION***] COD Edge Node for [***CLOUD PROVIDER NAME***].

   For example, select the 7.2.10 COD Edge Node for AWS cluster template.



6. In the Cluster Name field, provide a cluster name that you can identify later as an edge node of a specific COD instance.
7. Click Provision Cluster.

What to do next

After you deploy the edge node, you can run your applications on this edge node using the Client connectivity information. See how to compile applications for COD in Compile an application for your database.

Importing and restoring data into COD database

You can import your data into your Cloudera Operational Database (COD) database by restoring your HBase table into COD.

Before you begin

- Enable HBase replication on your COD 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
  ```

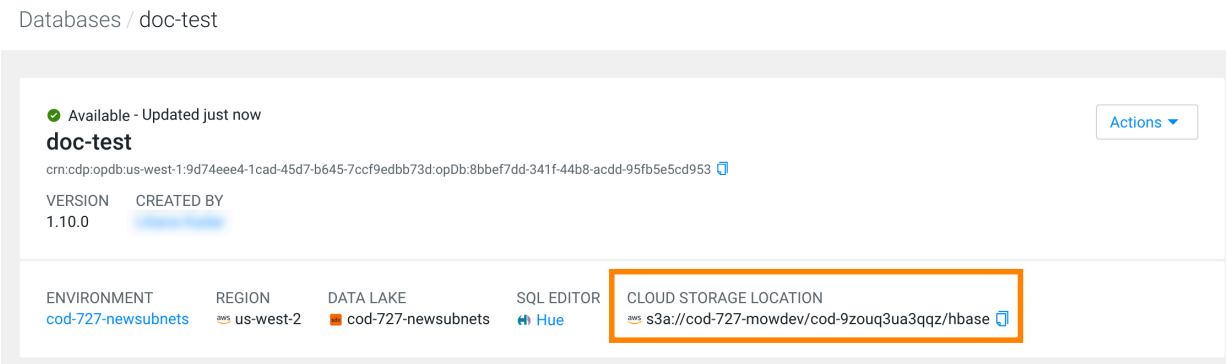
  For example, the data-from-onprem snapshot can be exported into s3a://cod-external-bucket/hbase:

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

- Have and 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 COD database using the COD web interface.
   s3a://my_cod_bucket/cod-12345/hbase

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 COD 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.

Example: run an application for the Apache HBase client

Checking the example of how to run a Maven application for the HBase client gives you better understanding about how to run your own application for the HBase client.

Before you begin

The requiredKerberos option is set to true for the Apache HBase client. The application must be run for this client from a computer which:

- Has internal network access to the public cloud in which the database is deployed
- Can resolve the internal hostnames of the database
- Can obtain a Kerberos ticket from the database's Kerberos Key Distribution Center (KDC)

One way to run the Apache HBase client applications is to launch an edge node in your cloud provider which meets the above requirements.

Ensure that you download the directory containing the Apache HBase client configuration files from clientConfig urationUrl provided in describe-client-connectivity response. This is a protected endpoint, and you have to use your CDP workload credentials to access this endpoint. You can also get the Apache HBase client configuration as a client tarball from the COD database web user interface Connect tab.

Procedure

1. Use clientConfigurationURL from the describe-client-connectivity response to obtain the necessary configuration details to communicate with Apache HBase:

   ```
   $ cdp opdb describe-client-connectivity --database-name [***YOUR DATABASE
    NAME***] --environment-name [***YOUR ENVIRONMENT NAME***]  | jq '.connect
   ors[] | select(.name == "hbase") | .configuration.clientConfigurationUrl'
   "https://client_Configuration_URL/.../services/hbase/clientConfig"
   $ curl -k -o clientConfig.zip -u '[***USERNAME***]:[***CDP WORKLOAD
    PASSWORD***]' https://client_Configuration_URL/.../services/hbase/client
   Config
   ```

   You can build the application in your local machine and copy the JAR files to the remote node using the following commands:

   ```
   $ scp -r target ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws
   .com:
   $ scp clientConfig.zip ec2-user@my-ec2-bastion-host.us-west-2.compute.amaz
   onaws.com:
   $ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "sudo
   yum install -y java-1.8.0-openjdk"
   $ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "unzip
    clientConfig.zip"
   ```

2. Ensure that you have a Kerberos ticket:

```
$ kinit [***USERNAME***] Password: [***PASSWORD***]
$ java -cp target/nosql-libs/*:target/nosql-exemplar-0.0.1-SNAPSHOT.jar
:hbase-conf com.cloudera.odx.nosql.Client
```

3. Run your application.

Example: run an application for the Apache Phoenix thick client

Checking the example of how to run a maven application for the Phoenix thick client gives you better understanding about how to run your own application for the Phoenix thick client.

Before you begin

The requiredKerberos option is set to true for the Phoenix thick client. This means that the application must be run for this client from a computer which:

- Has internal network access to the public cloud in which the database is deployed
- Can resolve the internal hostnames of the database
- Can obtain a Kerberos ticket from the database's Kerberos Key Distribution Center (KDC)

One way to run an Apache Phoenix thick client application is to launch an edge node in your cloud provider which meets the above requirements.

Procedure

1. Use the JDBC URL from the describe-client-connectivity command to run the example.

   One method is to build on your local machine and copy the JAR files to the remote node:

```
$ scp -r target ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws
.com:
$ scp clientConfig.zip ec2-user@my-ec2-bastion-host.us-west-2.compute.amaz
onaws.com:
$ ssh ec2-user@my-ec2-bastion-host.us-west-2.compute.amazonaws.com "sudo
yum install -y java-1.8.0-openjdk"
```

2. Ensure that you have a Kerberos ticket and run your application for the Phoenix thick client

```
 kinit username
$ java -cp target/sql-libs/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar:hbase-
conf com.cloudera.odx.sql.Client "[***PHOENIX THICK JDBC URL***]"
```

3. Run your application.

Example to run application using the Phoenix thin client

Checking the example of how to run a maven application for the Phoenix thin client gives you better understanding about how to run your own application for the Phoenix thin client.

The requiredKerberos options set to false for the Phoenix thin client which means that it can be used from virtually any node. In this example, the client runs from the local machine.

For the Apache Phoenix thin client the describe-client-connectivity call returns a base JDBC URL . You must append the following attributes to the URL which are specific to your identity:

- avatica_user: your CDP username (required)
- avatica_password: your CDP worload password (required)
- truststore: a truststore for your CDP Knox gateway (optional)

- truststore_password: the password for the truststore file (optional)

You can use Maven to ease launching this application, but a standalone Java program is similarly launched:

```
$ mvn exec:exec -Dexec.executable=java -Dexec.args='-cp target/sql-libs/*:ta
rget/sql-exemplar-0.0.1-SNAPSHOT.jar com.cloudera.odx.sql.ThinClient "jdbc:p
hoenix:thin:url=[***PHOENIX THIN JDBC URL;serialization=PROTOBUF;authenti
cation=BASIC;avatica_user=[***USERNAME***];avatica_password=[***PASSWORD***]
;truststore=[***CDP-TRUSTSTORE.JKS***];truststore_password=[***TRUSTSTORE
 PASSWORD***]"'
```

Or, you can launch the application without the help of Maven:

```
$ java -cp target/sql-libs/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar com.clou
dera.odx.sql.ThinClient "jdbc:phoenix:thin:url=[***PHOENIX THIN JDBC URL;ser
ialization=PROTOBUF;authentication=BASIC;avatica_user=[***USERNAME***];ava
tica_password=[***PASSWORD***];truststore=[***CDP-TRUSTSTORE.JKS***];truststo
re_password=[***TRUSTSTORE PASSWORD***]"
```

Compile an application for your COD database

Once you have created your application and a database using Cloudera Operational Database (COD), you have to compile your application for your database.

Before you begin

- Set your CDP workload password. For more information, see the *Setting the workload password* documentation in the related information section.
- Grant the ODUser role to the machine user using the Management Console Actions Manage Acces page for their CDP environment. By setting the ODUser role you can grant a number of rights in the system that allows you to access the COD using the machine user's workload password.
- Add synchronized users from User Management Service in the CDP Control Plane into the environment in which your COD database is running.

Procedure

1. Get the Maven repository location to fetch JAR files.

   There are two ways to get the necessary information:

   - In command line: Using the cdp opdb describe-client-connectivity --database-name <*your_database*> --environment-name <*your_environment*> command.
   - In the COD user interface: Clicking the Connect bar and selecting the applicable client.

   You have to use the version and the mavenURL attributes in your Maven project and configuration.

   The following is an example about how to fetch the required HBase information to build your application:

   ```
   $ cdp opdb describe-client-connectivity --database-name <your_database> --
   environment-name <your_environment>  | jq '.connectors[] | select(.name
    == "hbase")'
   {
     "name": "hbase",
     "version": "2.2.3.7.2.0.0-219",
     "kind": "LIBRARY",
     "dependencies": {
       "mavenUrl": "https://repository.cloudera.com/artifactory/cloudera-
   repos"
     },
   ```

```
  "configuration": {
    "clientConfigurationUrl": "http://client_Configuration_URL/.../
services/hbase/clientConfig"
  },
  "requiresKerberos": true
```

The following example fetch the required Phoenix information to build your application:

```
 cdp opdb describe-client-connectivity --database-name <your_database> --e
nvironment-name <your_environment> | jq ".connectors[] | select(.name == \
"phoenix-thick-jdbc\") | .dependencies.mavenUrl"
  cdp opdb describe-client-connectivity --database-name <your_database> --
environment-name <your_environment> | jq ".connectors[] | select(.name ==
 \"phoenix-thin-jdbc\") | .version"

Phoenix-thick
"https://repository.cloudera.com/artifactory/cloudera-repos"
"5.0.0.7.2.0.0-128"
Phoenix-thin
"https://repository.cloudera.com/artifactory/cloudera-repos"
"5.0.0.7.2.0.0-128"
```

2. Modify your application's settings.

   Ensure your application's settings.xml file uses the correct URL and version for your COD database.

   An example when using NoSQL client:

```
<project>
  <dependencies>
    <!-- NoSQL client for COD -->
    <dependency>
      <groupId>org.apache.hbase</groupId>
      <artifactId>hbase-shaded-client</artifactId>
      <version>2.2.3.7.2.0.0-219</version>
    </dependency>
  </dependencies>
  ...
  <repositories>
    <!-- Define our COD repository; this would be given to us by COD itsel
f -->
    <repository>
      <id>nosql-odx</id>
<url>https://maven_URL/cdp-proxy/hbase/jars</url>
      <name>Cloudera NoSQL COD Repository</name>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

   An example when using SQL client:

```
<project>
  <dependencies>
    <!-- SQL client for ODX -->
    <dependency>
      <groupId>org.apache.phoenix</groupId>
      <artifactId>phoenix-client</artifactId>
      <version>5.0.0.7.2.0.0-128</version>
    </dependency>
    <dependency>
      <groupId>org.apache.phoenix</groupId>
```

```
      <artifactId>phoenix-queryserver-client</artifactId>
      <version>5.0.0.7.2.0.0-128</version>
    </dependency>
  </dependencies>
  ...
  <repositories>
    <!-- Define our COD repository -->
    <repository>
      <id>sql-cod</id>
<url>https://maven_URL/cdp-proxy-api/avatica/maven</url>
      <name>Cloudera SQL COD Repository</name>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
    </repository>
  </repositories>
</project>
```

3. Build your application.
4. Run your application for the applicable client.