

## Compiling an Application Against COD

Date published: 2020-08-14

Date modified: 2023-01-12



# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

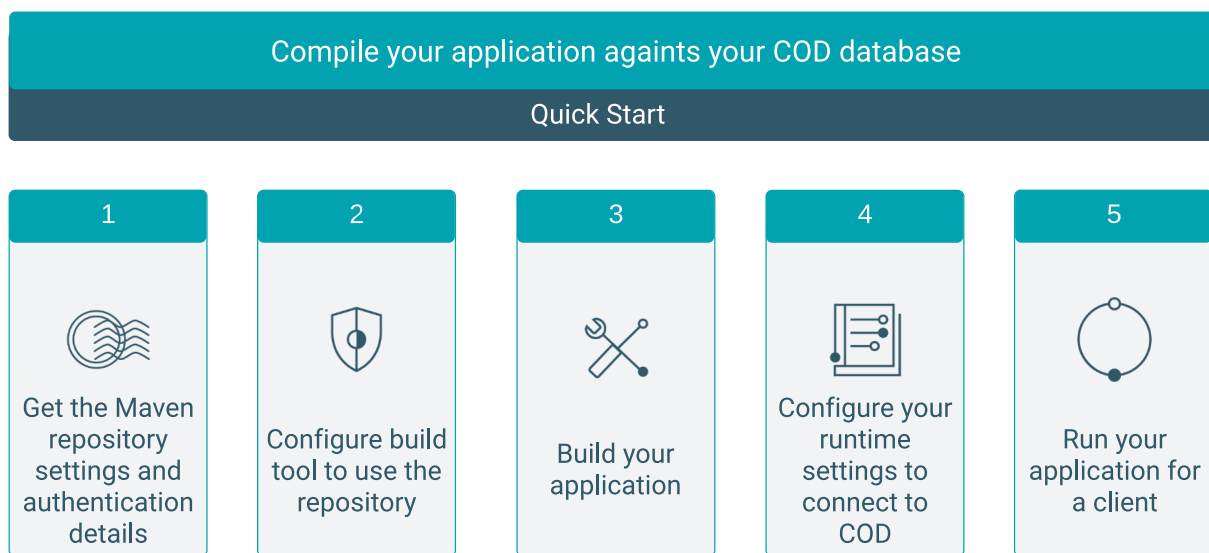
# Contents

<b>Client connectivity information for compiling your application for COD.....</b>	<b>4</b>
<b>Compile an application for your COD database.....</b>	<b>8</b>
<b>Example: run an application for the Apache HBase client.....</b>	<b>10</b>
<b>Example: run an application for the Apache Phoenix thick client.....</b>	<b>11</b>
<b>Example to run application using the Phoenix thin client.....</b>	<b>12</b>

## Client connectivity information for compiling your application for COD

When compiling your application for your Cloudera Operational Database (COD) you need information about the connectivity options that helps your applications interact with COD. You can get that information either using the CLI or the COD user interface.

The following is a graphical representation of the necessary steps to compile an application against COD.



From the Connect tab in the Databases user interface, you can get information about the available client connectivity configuration options in COD:

- HBase: Apache HBase Maven URL, Apache HBase client version, and the Apache HBase client configuration URL used to download a file containing the Apache HBase client configuration
- HBase REST: Apache HBase REST Server version and Apache HBase REST Server URL
- HBase Client Tarball: Apache HBase version, download URL of the client tarball, and the Apache HBase client configuration URL used to download a file containing the Apache HBase client configuration
- Phoenix (Thick): Phoenix thick JDBC driver URL and version number of the client artifacts
- Phoenix (Thin): Phoenix thin JDBC driver URL and version number of the client artifacts
- Phoenix (ODBC): Phoenix ODBC driver URL. You need to authenticate again to access this driver

You can also download the YARN and Kerberos configuration from these tabs where applicable.

### Maven repository containing the HBase client artifacts

Maven repository containing HBase client artifacts is provided in the HBase tab. You can get the following information from this tab:

- HBase Maven URL: Use this URL to access the Maven repository containing HBase client artifacts
- HBase Client Version: You can view the version of HBase client artifacts in the Maven repository
- HBase Client Configuration URL: You can get the endpoint which provides a ZIP file with HBase client configuration files

You cannot download the tarball using a web browser, you must instead use a tool like curl in the CLI to download the tarball. For example,

```
curl -O -u <username> "<download url>.tar.gz"
```

HBase Maven URL ⓘ

[Redacted]



HBase Client Version ⓘ

2.2.6.7.2.6.0-71



HBase Client Configuration URL ⓘ

[Redacted]



> [Kerberos Configuration](#)

Kerberos configuration information is provided in the HBase tab. Only authenticated clients can access a database in COD. You need the following Kerberos information that you can use to connect your application to COD:

- **Kerberos Realm:** You need the Kerberos realm information for connecting your clients requiring Kerberos authentication
- **KDC Host:** You need the Kerberos Key Distribution Center (KDC) hostname for your clients requiring Kerberos authentication
- **Krb5.conf:** You can download a sample krb5.conf file for your clients requiring Kerberos authentication

The /etc/krb5.conf file is the configuration a client uses to access a realm through its configured KDC. The krb5.conf maps the realm to the available servers supporting those realms. It also defines the host-specific configuration rules for how tickets are requested and granted. The krb5.conf file is used to determine the default Kerberos realm and KDC. Add the contents into the /etc/krb5.conf file on your edge node. For more information, see [Configure Kerberos](#).

▼ [Kerberos Configuration](#)

Kerberos Realm ⓘ

[Redacted]



KDC Host ⓘ

[Redacted]



Krb5.conf ⓘ

[Redacted]



## HBase REST

You can use the Apache HBase REST server to interact with Cloudera Operational Database (COD). You can get the following information from this tab:

- **HBase REST Server Version:** You can see the version of HBase REST server
- **HBase REST Server URL:** You can get the URL to the HBase REST Server to connect to your COD

HBase REST Server Version ⓘ

2.2.6.7.2.6.0-71



HBase REST Server URL ⓘ

[Redacted]



## HBase Client Tarball

You can download the 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.

Click **Connect HBase Client Tarball** tab, and then use the Download URL to download the tarball.

You cannot download the tarball using a web browser, you must instead use a tool like curl in the CLI to download the tarball. For example,

```
curl -O -u <username> "<download url>.tar.gz"
```

The screenshot shows the 'Connect' tab in the COD interface. The 'HBase Client Tarball' sub-tab is active. The 'Usage' section explains that the tarball contains JAR files and scripts for connecting to the database. The 'HBase Version' field is empty. The 'Download URL' field is highlighted with an orange arrow. The 'HBase Client Configuration URL' field is also empty. Below these fields are three expandable sections: 'Kerberos Configuration', 'Yarn Configuration', and 'JWT Configuration'.

Kerberos configuration information is provided in the HBase Client Tarball tab. Only authenticated clients can access a database in COD. You need the following Kerberos information that you can use to connect your application to COD:

- **Kerberos Realm:** You need the Kerberos realm information for connecting your clients requiring Kerberos authentication
- **KDC Host:** You need the Kerberos Key Distribution Center (KDC) hostname for your clients requiring Kerberos authentication
- **Krb5.conf:** You can download a sample krb5.conf file for your clients requiring Kerberos authentication

YARN configuration information is provided in the HBase Client Tarball tab. You need this additional configuration information to submit YARN applications that access data stored in COD.

You must create the ssl-client.xml in the downloaded and unzipped YARN Archive if it is not present in the archive folder. You need this file to run YARN workloads against COD.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>ssl.client.truststore.type</name>
    <value>jks</value>
  </property>
  <property>
    <name>ssl.client.truststore.location</name>
```

```

<value>[ [/path/to/truststore.jks]]</value>
</property>
<property>
  <name>ssl.client.truststore.password</name>
  <value>[ [password]]</value>
</property>
<property>
  <name>ssl.client.truststore.reload.interval</name>
  <value>10000</value>
</property>
</configuration>

```

- **YARN Archive:** You can use this URL to download a ZIP archive that contains the necessary YARN client configuration files
- **TLS Truststore:** You can use this command to download a Java KeyStore file that contains the certificate authority necessary to submit YARN jobs
- **TLS Truststore Password:** You can use this command to download the password for the TLS truststore

#### YARN Configuration

Additional configuration required to submit YARN application.

YARN Archive ⓘ

TLS Truststore ⓘ

TLS Truststore Password ⓘ

## Phoenix JDBC thick and thin drivers

Apache Phoenix has two kinds of JDBC drivers, thick and thin. The thick driver communicates directly with Apache ZooKeeper and Apache HBase, and the thin driver does this through the Phoenix Query Server. You can connect your application to the COD instance using the provided connection and configuration URLs.

You can get the URL for the thick and thin JDBC driver using:

- **CDP command-line interface:** Use the `cdp opdb describe-client-connectivity --database-name [***YOUR DATABASE***] --environment-name [***YOUR ENVIRONMENT***]` command
- **COD web user interface:** Click *Databases* *your database* *Connect* tab, and then select the Phoenix (Thick) or Phoenix (Thin) client

Phoenix Maven URL ⓘ

Phoenix (Thick) Client Version ⓘ

Phoenix (Thick) JDBC URL ⓘ

#### > Kerberos Configuration

Phoenix Maven URL ⓘ

Phoenix (Thin) Client Version ⓘ

Phoenix (Thin) JDBC URL ⓘ

## Phoenix ODBC driver

ODBC is one of the most established and widely supported API for connecting to and working with databases. Use the Database Open Database Connectivity (ODBC) interface to access the operational database. The ODBC driver is provided by Cloudera as an additional download, but you can also use ODBC drivers from third-party providers. You can download the ODBC driver for your operating system using the links in the Phoenix (ODBC) tab.

You can get the following information:

- Phoenix ODBC driver download URL: A URL where Phoenix ODBC driver is available. You may need to authenticate again to access this driver
- Phoenix ODBC URL: The basis of a ODBC URL for the Phoenix thin driver to connect to this database

Phoenix ODBC driver download URL ⓘ

[Windows 32-bit](#)

[Windows 64-bit](#)

[Linux 32-bit](#)

[Linux 64-bit](#)

[Mac OS X](#)

Phoenix ODBC URL ⓘ

## Related Information

[Configuring Kerberos](#)

# 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 Access](#) 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.

#### Related Information

[Setting the workload password](#)

## 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 `clientConfigurationUrl` 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 '.connectors[] | 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/clientConfig
```

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.amazonaws.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.amazonaws.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.

### Related Information

[COD edge node overview](#)

## 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 required Kerberos 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 workload 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/*:target/sql-exemplar-0.0.1-SNAPSHOT.jar com.cloudera.odx.sql.ThinClient "jdbc:phoenix:thin:url=[***PHOENIX THIN JDBC URL;serialization=PROTOBUF;authentication=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.cloudera.odx.sql.ThinClient "jdbc:phoenix:thin:url=[***PHOENIX THIN JDBC URL;serialization=PROTOBUF;authentication=BASIC;avatica_user=[***USERNAME***];avatica_password=[***PASSWORD***];truststore=[***CDP-TRUSTSTORE.JKS***];truststore_password=[***TRUSTSTORE PASSWORD***]"
```