Cloudera Runtime 7.2.11

# Working with Apache Hive metastore

**Date published: 2019-08-21**
**Date modified: 2021-09-08**

**https://docs.cloudera.com/**

# Legal Notice

# Contents

# HMS table storage

You need to understand how the Hive metastore (HMS) stores Hive tables when you run a CREATE TABLE statement or migrate a table to Cloudera Data Platform. The success or failure of the statement, the resulting table type, and the table location depends on a number of factors.

## HMS table transformations

The HMS includes the following Hive metadata about tables that you create:

- A table definition
- Column names
- Data types
- Comments in a central schema repository

When you use the EXTERNAL keyword in the CREATE TABLE statement, HMS stores the table as an external table. When you omit the EXTERNAL keyword and create a managed table, or ingest a managed table, HMS might translate the table into an external table or the table creation can fail, depending on the table properties. An important table property that affects table transformations is the ACID or Non-ACID table type:

**Non-ACID**

Table properties do not contain any ACID related properties set to true. For example, the table does not contain such properties transactional=true or insert_only=true.

**ACID**

Table properties do contain one or more ACID properties set to true.

**Full ACID**

Table properties contain transactional=true but not insert_only=true

**Insert-only ACID**

Table properties contain insert_only=true.

The following matrix shows the table type and whether or not the location property is supported.

| ACID | Managed | Location Property | Comments | Action |
|---|---|---|---|---|
| Non-ACID | Yes | Yes | Migrated to CDP, for example from an HDP or CDH cluster. | Table stored as external |
| Non-ACID | Yes | No | Table location is null | Table stored in subdirectory of external warehouse* |

* metastore.warehouse.external.dir

HMS detects type of client for interacting with HMS, for example Hive or Spark, and compares the capabilities of the client with the table requirement. HMS performs the following actions, depending on the results of the comparison:

| Table requirement | Client meets requirements | Managed Table | ACID table type | Action |
|---|---|---|---|---|
| Client can write to any type of ACID table | No | Yes | Yes | CREATE TABLE fails |
| Client can write to full ACID table | No | Yes | insert_only=true | CREATE TABLE fails |
| Client can write to insert-only ACID table | No | Yes | insert_only=true | CREATE TABLE fails |

If, for example, a Spark client does not have the capabilities required, the following type of error message appears:

```
Spark has no access to table `mytable`. Clients can access this table only i
f
they have the following capabilities: CONNECTORREAD,HIVEFULLACIDREAD, HIVE
FULLACIDWRITE,
HIVEMANAGESTATS, HIVECACHEINVALIDATE, . . .
```

**Related Information**

Secure Hive Metastore

# HWC authorization

The way you configure Hive Warehouse Connector (HWC) affects the query authorization process and your security. There are a number of ways to access Hive through HWC, and not all operations go through HiveServer (HS2). Some operations, such as Spark Direct Reader and Hive Streaming, go to Hive directly through HMS where storage-based permissions generally apply.

As a client user, you must be logged in using kerberos before using HWC. You need appropriate storage permissions to write to destination partition or table location. You need to configure an HWC read option. HWC read configuration options are shown in the following table:

**Table 1:**

| Capabilities | JDBC mode | Spark Direct Reader mode | Secure Access mode |
|---|---|---|---|
| Ranger integration with fine-grained access control | # | N/A | # |
| Hive ACID reads | # | # | # |
| Workloads handled | Non-production workloads, small datasets | Production workloads, ETL without fine-grained access control | Large workloads with fine-grained access control, row filtering, and column masking |

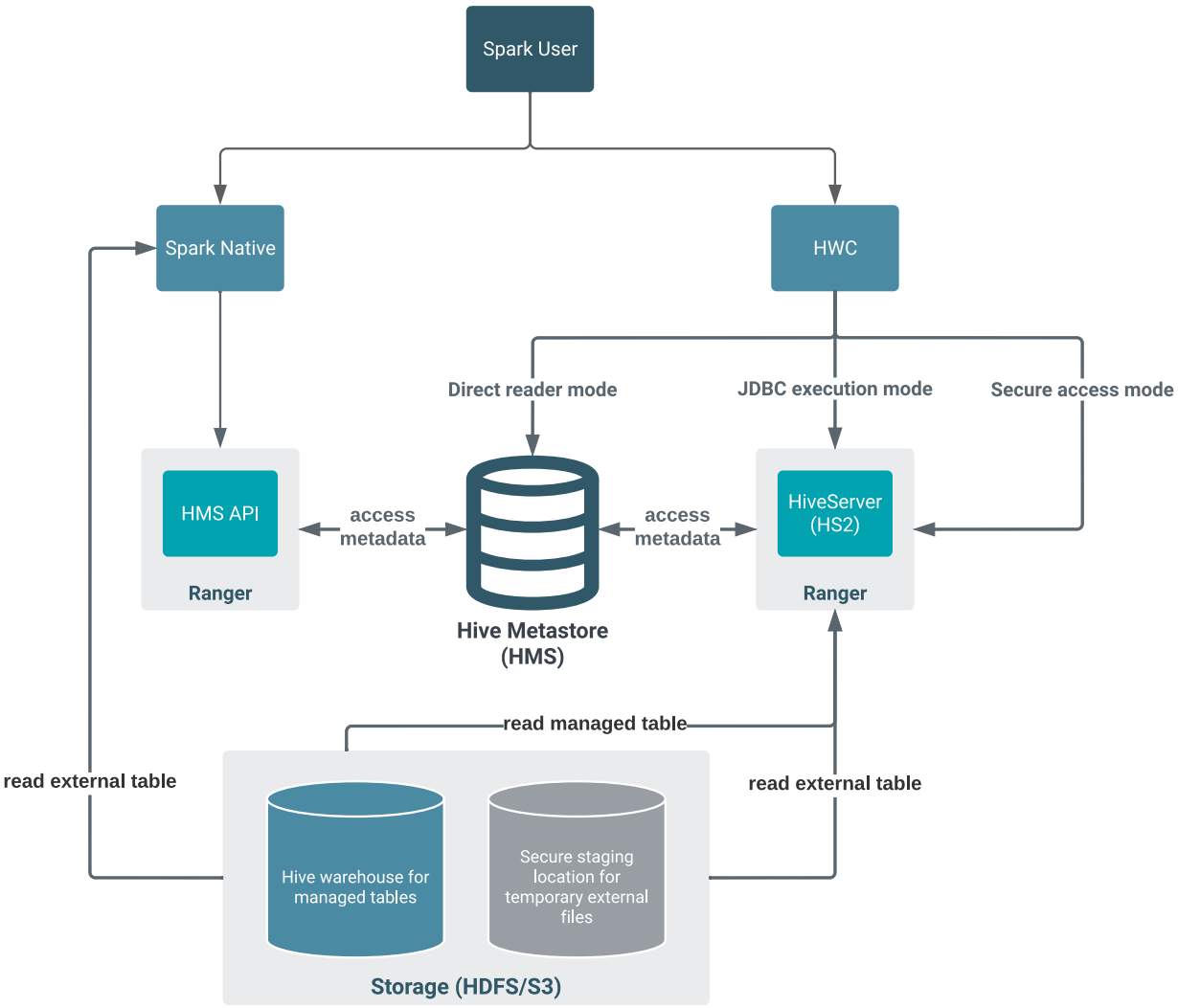
These read configuration options require connections to different Hive components:

- Direct Reader configuration: Connects to Hive Metastore (HMS)
- JDBC configuration: Connects to HiveServer (HS2)
- Secure Access configuration: Connects to HiveServer (HS2)

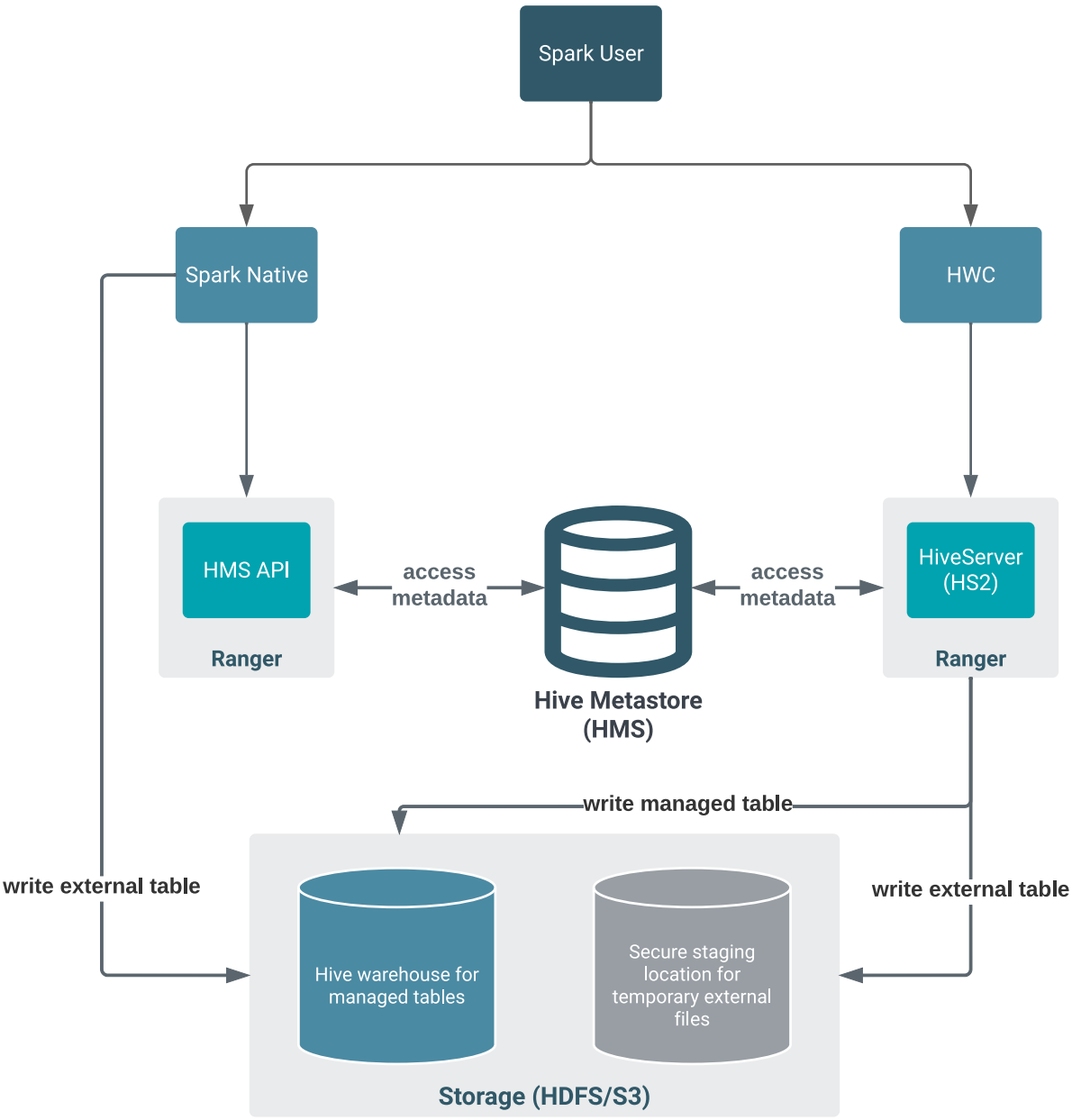
Ranger authorizes access to Hive tables from Spark through HiveServer (HS2) or the Hive metastore API (HMS API).

To write ACID managed tables from Spark to Hive, you must use HWC. To write external tables from Spark to Hive, you can use native Spark or HWC.

The following diagram shows the typical read authorization process:




The following diagram shows the typical write authorization process:




When writing, HWC always enforces authorization through HiveServer (HS2). Reading managed tables in JDBC mode or Secure access mode enforces Ranger authorization, including fine-grained features, such as row masking and column mapping. In Direct Reader mode, the Ranger and HMS integration provides authorization.

External table queries go through the HMS API, which is also integrated with Ranger. If you do not use HWC, the Hive metastore (HMS) API, integrated with Ranger, authorizes external table access. HMS API-Ranger integration enforces the Ranger Hive ACL in this case. When you use HWC, queries such as DROP TABLE affect file system data as well as metadata in HMS.

Using the Direct Reader option, SparkSQL queries read managed table metadata directly from the HMS, but only if you have permission to access files on the file system. You cannot write to managed tables using the Direct Reader option.

Using the Secure access mode, you can enable fine-grained access control (FGAC) column masking and row filtering to secure managed (ACID), or even external, Hive table data that you read from Spark.

Managed table authorization

A Spark job impersonates the end user when attempting to access an Apache Hive managed table. As an end user, you do not have permission to access, managed files in the Hive warehouse. Managed tables have default file system permissions that disallow end user access, including Spark user access.

As Administrator, you set permissions in Ranger to access the managed tables when you configure HWC for JDBC mode or Secure access mode reads. You can fine-tune Ranger to protect specific data. For example, you can mask data in certain columns, or set up tag-based access control.

When you configure HWC for Direct Reader mode, you cannot use Ranger in this way. You must set read access to the file system location for managed tables. You must have Read and Execute permissions on the Hive warehouse location (hive.metastore.warehouse.dir).

External table authorization

Ranger authorization of external table reads and writes is supported. You need to configure a few properties in Cloudera Manager for authorization of external table writes. You must be granted file system permissions on external table files to allow Spark direct access to the actual table data instead of just the table metadata.

### Direct Reader Authorization Limitation

As Spark allows users to run arbitrary code, Ranger fine grained access control, such as row level filtering or column level masking, is not possible within Spark itself. This limitation extends to data read using Direct Reader.

To restrict data access at a fine-grained level, use a read option that supports Ranger. Only consider using the Direct Reader option to read Hive data from Spark if you do not require fine-grained access. For example, use Direct Reader for ETL use cases.

# Authorizing external tables

As Administrator, you need to know how to authorize users to read and write to Apache Hive external tables, which includes accessing tables using Spark SQL, Hue, and Beeline. You also need to configure file level permissions on tables for users.

### About this task

You set the following properties and values for HMS API-Ranger integration:

**hive.metastore.pre.event.listeners**

Value:

```
org.apache.hadoop.hive.ql.security.authorization.plugin.metastor
e.HiveMetaStoreAuthorizer
```

Configures HMS writes.

**hive.security.authenticator.manager**

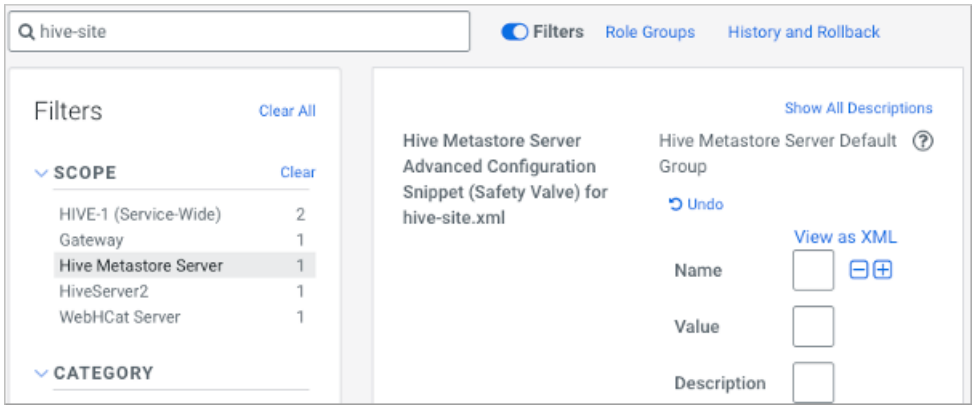
Value: org.apache.hadoop.hive.ql.security.SessionStateUserAuthenticator

Add properties to hive-site.xml using the Cloudera Manager Safety Valve as described in the next section.

### Procedure

1. In Cloudera Manager, to configure Hive Metastore properties click Clusters Hive-1 Configuration .
2. Search for hive-site.

3. In Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click +.




4. Add a property name and value.
5. Repeat steps to add other properties.
6. Save changes.
7. Configure file level permissions on tables for users.

   Only users who have file level permissions on external tables can access external tables.

# Configure HMS properties for authorization

As Administrator, if you have any problem with query authorization, you might need to set up Apache Hive metastore (HMS) authorization through Ranger. For example, if you configured storage-based authorization of Hive queries, and then you want to switch to authorization through Ranger, you must setup authorization through Ranger. You configure HMS properties to make this switch.

## About this task

To integrate the HMS API and Ranger for authorizing queries, you need to add the following HMS properties and values to hive-site.xml using Cloudera Manager:

**hive.metastore.pre.event.listeners**

Value:

```
org.apache.hadoop.hive.ql.security.authorization. \
  plugin.metastore.HiveMetaStoreAuthorizer
```

Configures HMS writes.

**hive.security.authenticator.manager**

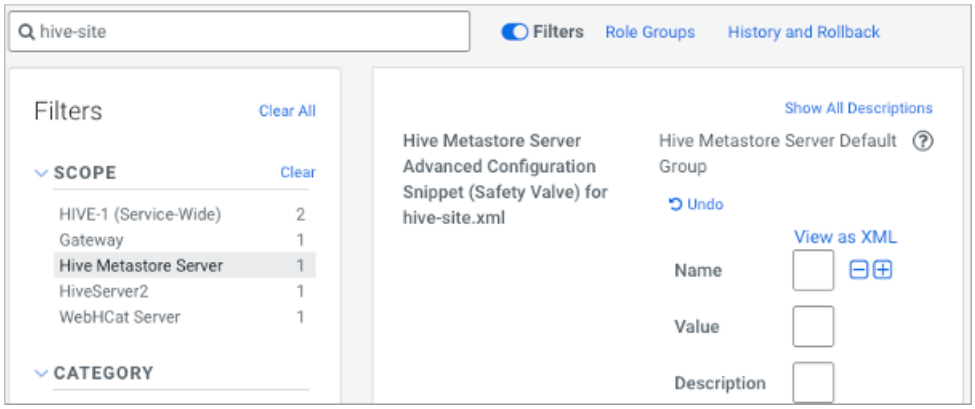
Value: org.apache.hadoop.hive.ql.security.SessionStateUserAuthenticator

Add properties to hive-site.xml using the Cloudera Manager Safety Valve as described in the next section.

## Procedure

1. In Cloudera Manager, to configure Hive Metastore properties click Clusters Hive-1 Configuration .
2. Search for hive-site.

3. In Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click +.




4. Add a property name and value.
5. Repeat steps to add other properties.
6. Save changes.

### Related Information

Secure Hive Metastore

# Filter HMS results

After you write an API to filter data from Hive, you need to know how to set up the API, enable it, and disable it. You need to add few Hive metastore (HMS) properties hive-site.xml using the Cloudera Manager Safety Valve.

### About this task

HMS can perform server-side filtering of data returned by a read operation. Filtering is enabled by default, which shows the results of statements, such as SHOW TABLES or SHOW DATABASES, based on who the user is. You can disable filtering by setting a boolean flag and hook. The hook identifies the class name that implements the filtering. You add the following properties and values in hive-site.xml for HMS API-Ranger integration:

**metastore.server.filter.enabled**

Value: true (to do filtering) or false (no filtering)

**metastore.filter.hook**

Value: org.apache.hadoop.hive.ql.security.authorization.plugin.metastore.HiveMetaStoreAuthorizer

Add properties as described in the previous section.

### Related Information

Secure Hive Metastore

# Setting up the metastore database

You need to know how to set up a backend database for Hive metastore (HMS) if you have an on-prem cluster. Set up consists of installing a supported database, configuring properties, specifying the metastore location. You can also configure optional connection parameters.

You need to install a supported database for Hive metastore (HMS) to store the metadata. You configure Hive metastore by modifying hive-site.xml. You use the Cloudera Manager Safety Valve feature instead of hive set key =value on the command line.

### Related Information

Apache Wiki: Hive Metastore Administration
Example of using the Cloudera Manager Safety Valve
Custom Configuration (about Cloudera Manager Safety Valve)

## Setting up the backend Hive metastore database

In CDP Private Cloud Base you need to install, start, and configure a backend database for Hive metastore.

### About this task

In this procedure, you install a database on a different node/cluster from the HiveServer for sharing the Hive metastore (HMS) with Hive, Impala, Spark, and other components. Do not put HiveServer and the database on the same node. You can have one or more HMS instances in your cluster that can take over in the event of a problem.

### Procedure

Install a supported database.

- MariaDB/MySQL
- PostgreSQL
- Oracle

**Note:** CDP does not support Percona for MySQL as a backend database for HMS.

## Set up MariaDB or MySQL database

You install a MariaDB or MySQL database to serve as the backend database for the Hive metastore. You also install the MySQL driver on your cluster, and then configure the database.

### Procedure

1. Install MySQL from the command line of a node in your cluster.
   - RHEL:
     ```
     sudo yum install mysql-server
     ```
   - SLES:
     ```
     sudo zypper install mysql
     sudo zypper install libmysqlclient_r17
     ```
   - Ubuntu:
     ```
     sudo apt-get install mysql-server
     ```
2. Start the mysql daemon.
   - RHEL:
     ```
     sudo service mysqld start
     ```
   - SLES or Ubuntu:
     ```
     sudo service mysql start
     ```
3. Install the latest MySQL JDBC driver on the Hive metastore server node in your cluster.

4. Set the MySQL root password.

```
sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
[...]
Set root password? [Y/n] y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] N
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

5. Configure the MySQL server to start when the cluster starts up.
   - RHEL

     ```
     sudo /sbin/chkconfig mysqld on
     sudo /sbin/chkconfig --list mysqld
     ```

   - SLES:sudo chkconfig --add mysql
   - Ubuntu: sudo chkconfig mysql on
6. Create the initial database schema.

```
mysql -u root -p
Enter password:
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE /usr/lib/hive/scripts/metastore/upgrade/mysql/hive-schema-
n.n.n.mysql.sql;
```

   If the metastore service runs on the host where the database is installed, replace 'metastorehost' in the CREATE USER example with 'localhost'.

7. Create a MySQL user account to access the metastore.

```
mysql> CREATE USER 'hive'@'metastorehost' IDENTIFIED BY 'mypassword';
...
mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'hive'@'metastorehost';
mysql> GRANT ALL PRIVILEGES ON metastore.* TO 'hive'@'metastorehost';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

## Set up a PostgreSQL database

You install a Postgres database to serve as the backend database for the Hive metastore. You also install the Postgres driver on your cluster, and then configure the database.

### Procedure

1. Install the database from the command line of a node in your cluster.
   - RHEL:
     ```
     sudo yum install postgresql-server
     ```
   - SLES:
     ```
     sudo zypper install postgresql-server
     ```
   - Ubuntu:
     ```
     sudo apt-get install postgresql
     ```
2. If your system is RHEL, initialize database files; otherwise, skip this step.
   ```
   sudo service postgresql initdb
   ```
3. Set the listen_addresses property to * and set the standard_conforming_strings property off.
   - RHEL and SLES:
     ```
     sudo cat /var/lib/pgsql/data/postgresql.conf  | grep -e listen -e standa
     rd_conforming_strings
     listen_addresses = '*'
     standard_conforming_strings = off
     ```
   - Ubuntu:
     ```
     cat /etc/postgresql/9.1/main/postgresql.conf | grep -e listen -e standar
     d_conforming_strings
     listen_addresses = '*'
     standard_conforming_strings = off
     ```
4. Add a new line into pg_hba.con to ensure that the Postgres user can access the server remotely.
   For example, to allow all users to connect from all hosts to all your databases:
   ```
   host  <database>  <user>  <network address>  <mask>   md5
   ```
   This configuration is applicable only for a network listener. Using this configuration does not open all your databases to the entire world; you must provide a password to authenticate yourself, and privilege restrictions configured in PostgreSQL are effective.
5. Start the Postgres server.
   ```
   sudo service postgresql start
   ```
6. Set Postgres to start when the cluster comes up, and check the configuration.
   ```
   chkconfig postgresql on
   chkconfig --list postgresql
   ```
7. On the Hive metastore server node in your cluster, install the latest Postgres JDBC driver (the postgresql-jdbc package), and create symbolic link to the /usr/lib/hive/lib/ directory.
   - RHEL
     ```
     sudo yum install postgresql-jdbc
     ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-
     jdbc.jar
     ```

- SLES

```
sudo zypper install postgresql-jdbc
ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-j
dbc.jar
```

- Ubuntu

```
sudo apt-get install libpostgresql-jdbc-java
ln -s /usr/share/java/postgresql-jdbc4.jar /usr/lib/hive/lib/postgresql-
jdbc4.jar
```

8. Create a metastore database and user account to access the metastore, using the script for your Hive version represented by n.n.n.

```
sudo -u postgres psql
postgres=# CREATE USER hiveuser WITH PASSWORD 'mypassword';
postgres=# CREATE DATABASE metastore;
postgres=# \c metastore;
You are now connected to database 'metastore'.
postgres=# \i /usr/lib/hive/scripts/metastore/upgrade/postgres/hive-sche
ma-n.n.n.postgres.sql
SET
SET
...
```

## Set up an Oracle database

You install an Oracle database to serve as the backend database for the Hive metastore. You can use the free Express edition. You also install the Oracle driver on your cluster, and then configure the database.

### Procedure

1. Install Oracle on a node in your cluster.
2. Download the Oracle JDBC driver from the Oracle web site, and put the JDBC JAR into /usr/lib/hive/lib/.

```
sudo mv ojdbc<version_number>.jar /usr/lib/hive/lib/
```

3. Connect to the Oracle database as administrator, and create a user account to access the metastore.

```
sqlplus "sys as sysdba"
SQL> create user hiveuser identified by mypassword;
SQL> grant connect to hiveuser;
SQL> grant all privileges to hiveuser;
```

4. Connect as the hiveuser, and load the initial schema, using the script for your release n.n.n.

```
sqlplus hiveuser
SQL> @/usr/lib/hive/scripts/metastore/upgrade/oracle/hive-schema-n.n.n.
oracle.sql
```

5. Connect to the Oracle database as administrator and remove the power privileges from user hiveuser.

```
sqlplus "sys as sysdba"
SQL> revoke all privileges from hiveuser;
```

6. Grant limited access to all the tables.

```
SQL> BEGIN
FOR R IN (SELECT owner, table_name FROM all_tables WHERE owner='HIVEUSER')
 LOOP
```

```
  EXECUTE IMMEDIATE 'grant  SELECT,INSERT,UPDATE,DELETE on '||R.owner||'.
'||R.table_name||' to hiveuser';
  END LOOP;
END;

/
```

# Configuring metastore database properties

In CDP Private Cloud Base, you configure Hive and Hive metastore by modifying hive-site.xml indirectly using the Cloudera Manager Safety Valve feature. A step-by-step procedure shows you how to set a number of property names and values in lieu of using hive set key=value on the command line, which is not supported.

## About this task

This task assumes the database is running on myhost, the user account is hiveuser, and the password is mypassword. Substitute the following connection URLs and driver names depending on the your database type.

- MySQL connection URL: jdbc:mysql://myhost/metastore

  MySQL driver name: com.mysql.jdbc.Driver
- Postgres connection URL: jdbc:postgresql://myhost/metastore

  Postgres driver name: org.postgresql.Driver
- Oracle connection URL: jdbc:oracle:thin:@//myhost/xe

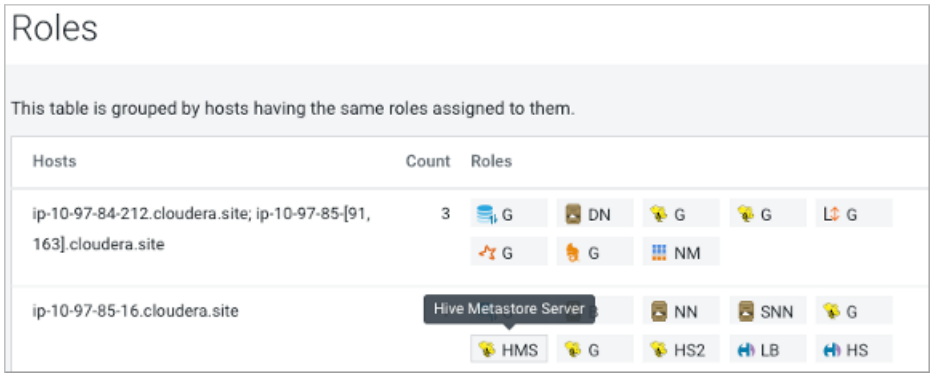
  Oracle driver name: oracle.jdbc.OracleDriver

## Before you begin

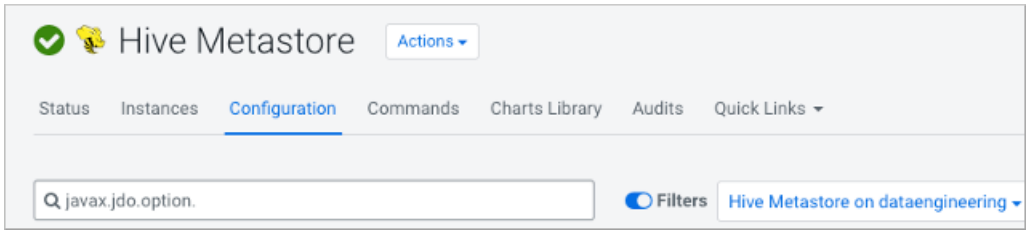
- The following components are running:
  - HiveServer
  - Hive Metastore
  - A database for the metastore, such as the default MySQL Server
  - Hive clients
- Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

## Procedure

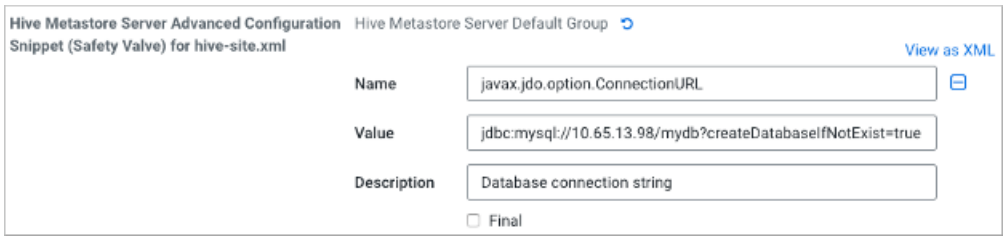
1. Find the fully qualified domain name or IP address of Hive metastore by navigating to Cloudera Manager Hosts Role(s) and looking through the list of roles to find Hive Metastore Server.



2. Navigate to the metastore host configuration in Clusters Hive Metastore Configuration , and search for jav ax.jdo.option.ConnectionURL.




3. In Value, specify the database connection string using the following syntax: <connection protocol>://<metastore host>/<metastore database>?createDatabaseIfNotExist=true
For example:




In View as XML, the XML configuration snippet appears.

```
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://10.65.13.98/mydb?createDatabaseIfNotExist=true</value>
<description>Database connection string</description>
</property>
```

4. Optionally, repeat the previous steps on all hosts in the cluster.
5. In the same manner, specify other required connection properties on the metastore host (required), or on all hosts (optional) as shown in the following example.

```
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>(your driver name)</value>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>hive</value>
</property>

<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>mypassword</value>
</property>

<property>
<name>datanucleus.autoCreateSchema</name>
<value>false</value>
</property>

<property>
<name>datanucleus.fixedDatastore</name>
<value>true</value>
</property>

<property>
<name>datanucleus.autoStartMechanism</name>
<value>SchemaTable</value>
```

```
</property>

<property>
<name>hive.metastore.schema.verification</name>
<value>true</value>
</property>
```

### Related Information

Example of using the Cloudera Manager Safety Valve

Custom Configuration (about Cloudera Manager Safety Valve)

## Configuring metastore location and HTTP mode

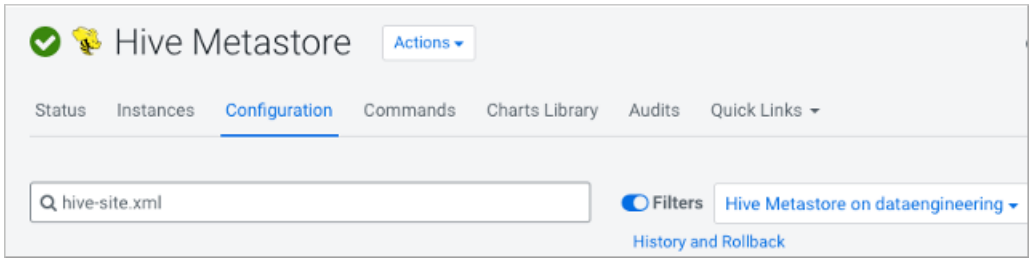
In addition to the database properties you need to set in CDP Private Cloud Base, you must configure the metastore URI property. This property defines one or more metastore locations.

### Before you begin

- The following components are running:
  - HiveServer
  - Hive Metastore
  - A database for the metastore, such as the default MySQL Server
  - Hive clients
- Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

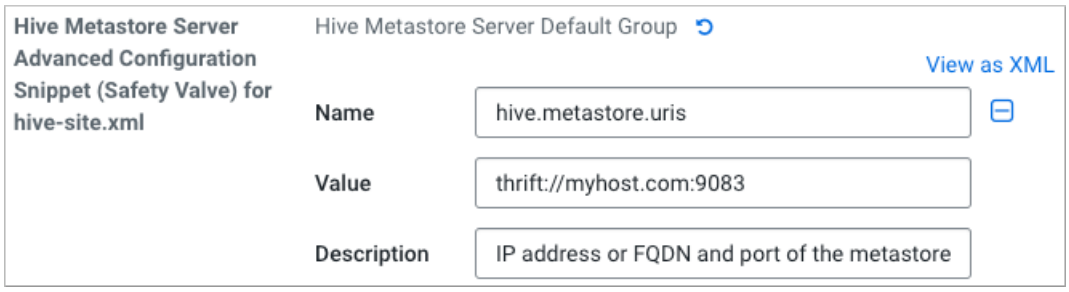
### Procedure

1. In Clusters Hive Metastore Configuration , search for hive-site.xml.




2. In the Hive Metastore Server Advanced Configuration Safety Valve, which you use to change properties in hive-site.xml, click + and add the hive.metastore.uris property using the following syntax: thrift://<n.n.n.n>:9083

   Substitute for <n.n.n.n> an IP address or fully qualified domain name (FQDN) of the metastore host.




   Only the Hive Metastore Server Default Group in hive-site.xml should define this property.

### Related Information

Secure Hive Metastore

# Setting up a JDBC URL connection override

You can configure fine-grained tuning of the HMS database connection. You specify a JDBC URL override, which depends on your database, for establishing a connection to the Hive metastore database.

## About this task

This task is intended for advanced database users only. When using this override, the following properties are overwritten

- Hive Metastore Database Name
- Hive Metastore Database Host
- Hive Metastore Database Port
- Enable TLS/SSL to the Hive Metastore Database

## Before you begin

- The required default user role is Configurator.
- You know the values for setting the following properties:
  - Hive Metastore Database Type
  - Hive Metastore Database User
  - Hive Metastore Database Password

## Procedure

1. Set the value of the Hive Metastore Database JDBC URL Override property according to your cluster configuration.
   - MySQL

     ```
     jdbc:mysql://<host>:<port>/<metastore_db>?key=value
     ```
   - PostgreSQL

     ```
     jdbc:postgresql://<host>:<port>/<metastore_db>?key=value
     ```
   - Oracle JDBC Thin using a Service Name

     ```
     jdbc:oracle:thin:@//<host>:<port>/<service_name>
     ```
   - Oracle JDBC Thin using SID

     ```
     jdbc:oracle:thin:@<host>:<port>:<SID>
     ```
   - Oracle JDBC Thin using TNSName

     ```
     jdbc:oracle:thin:@<TNSName>
     ```
2. Click Save.
3. Click  Actions Deploy Client Configuration .
4. Restart Hive Metastore.

## Related Information

Secure Hive Metastore

# Tuning the metastore

Similar to other tuning procedures, general metastore tuning involves tweaking and testing until you discover the combination of changes that improves metastore performance. Tuning suggestions include hardware and software changes.

Generally, you need to limit concurrent connections to Hive metastore. As the number of open connections increases, so does latency. Issues with the backend database, improper Hive use, such as extremely complex queries, a connection leak, and other factors can affect performance.

## General Metastore Tuning

Try making the following changes to tune HMS performance:

- Ensure that a single query accesses no more than 10,000 table partitions. If the query joins tables, calculate the combined partition count accessed across all tables.
- Tune the backend (the RDBMS). HiveServer connects to HMS, and only HMS connects to the RDBMS. The longer the backend takes, the more memory the HMS needs to respond to the same requests. Limit the number of connections in the backend database.

  MySQL: For example, in /etc/my.cnf:

  ```
  [mysqld]
          datadir=/var/lib/mysql
          max_connections=8192
          . . .
  ```

  MariaDB: For example, in /etc/systemd/system/mariadb.service.d/limits.conf:

  ```
  [Service]
          LimitNOFILE=24000
          . . .
  ```

- Use default thrift properties (8K):

  ```
  hive.server2.async.exec.threads 8192
  hive.server2.async.exec.wait.queue.size 8192
  hive.server2.thrift.max.worker.threads 8192
  ```

- Set datanucleus.connectionPool.maxPoolSize for your applications. For example, if poolSize = 100, with 3 HMS instances (one dedicated to compaction), and with 4 pools per server, you can accommodate 1200 connections.

## Related Information

Secure Hive Metastore