Cloudera Runtime 7.0.0

Configuring Apache Hive metastore

Date published: 2019-08-21
Date modified:

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

Apache Hive metastore in CDP 4

Configuring HMS for high availability 4

Setting up the metastore database 5

Set up the backend Hive metastore database 6

Set up MariaDB or MySQL database 6

Set up a PostgreSQL database 7

Set up an Oracle database 9

Configure metastore database properties 10

Configure metastore location and HTTP mode 12

Set up a JDBC URL connection override 13

Tuning the metastore 14

HMS table storage 15

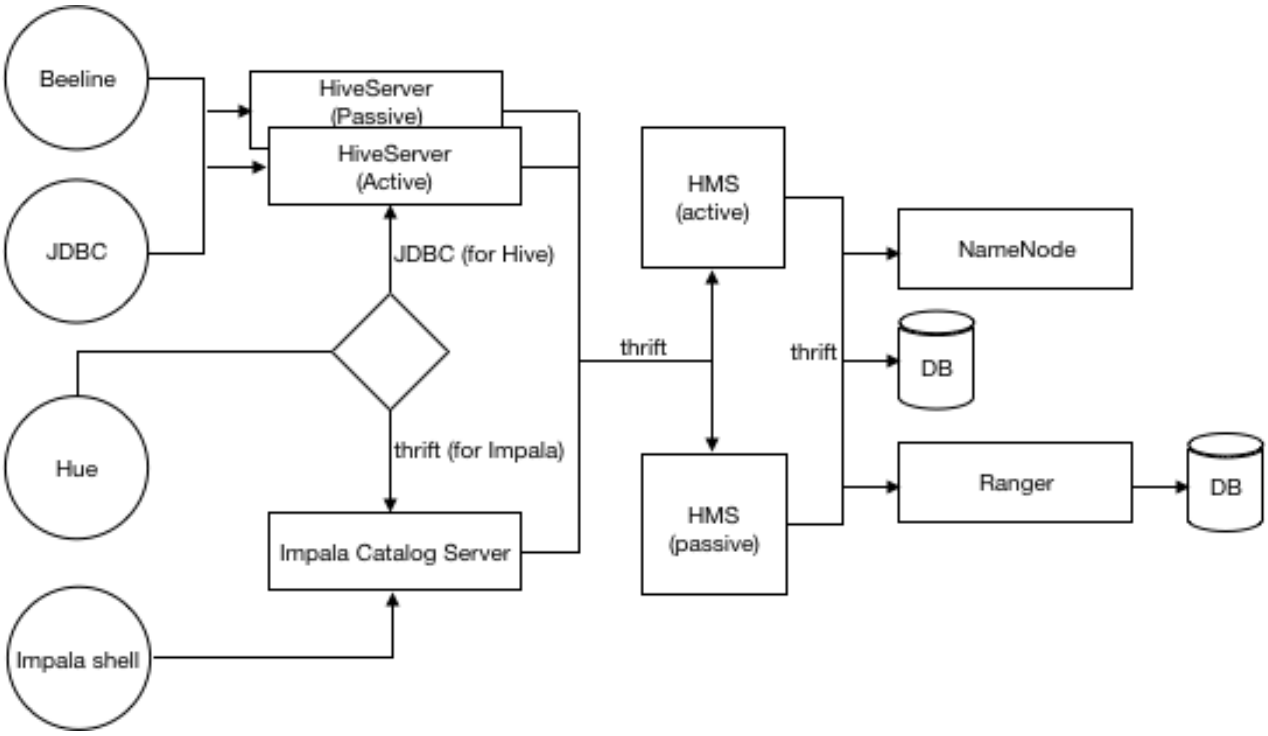
Setting up a shared Amazon RDS as a Hive metastore 16

Set up Amazon RDS as a Hive metastore 16

Apache Hive metastore in CDP

The Hive metastore (HMS) is a separate service, not part of Hive, not even necessarily on the same cluster. HMS stores the metadata on the backend for Hive, Impala, Spark, and other components.

Beeline, Hue, JDBC, and Impala shell clients make requests through thrift or JDBC to HiveServer. The redundant HiveServer is passive and provides fail-over services. The HiveServer instance reads/writes data to Hive metastore. One or more HMS instances on the backend can talk to other services, such as Ranger. The redundant HMS is passive and provides fail-over services. The physical data resides in a backend RDBMS, one for HMS and one for the security service, Ranger for example. All connections are routed to a single RDBMS service at any given time. HMS talks to the NameNode over thrift and functions as a client to HDFS.



Configuring HMS for high availability

To provide failover to a secondary Hive metastore if your primary instance goes down, you need to know how to add a Metastore role in Cloudera Manager and configure a property.

About this task

Multiple HMS instances run in active/active mode. Load balancing is done at the Hive client side (like HiveServer or Spark) as the HMS client picks an HMS instance randomly. By default, the hive.metastore.uri.selection property is set to RANDOM. If that HMS instance is down, then the client randomly picks another instance from the list of HMS instances specified through the hive.metastore.uris property.

Before you begin

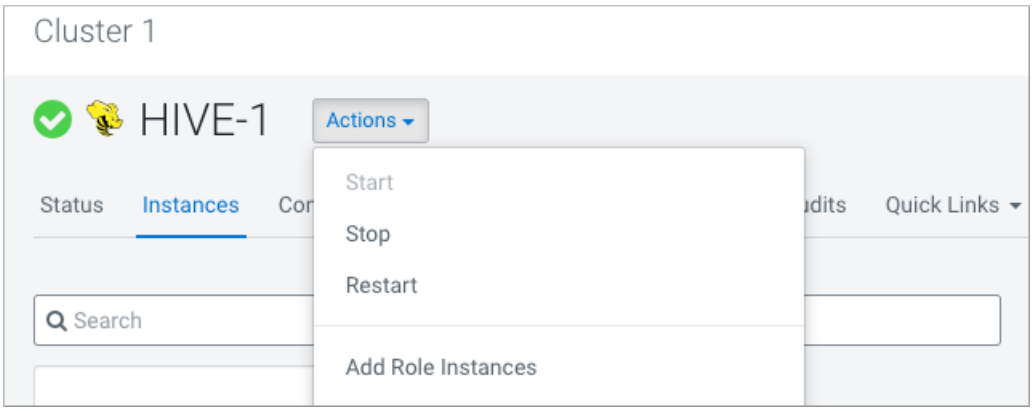
Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

Procedure

1. In Cloudera Manager, click Clusters Hive (aka Hive Metastore) Configuration .

2. Take one of the following actions:
   - If you have a cluster secured by Kerberos, search for Hive Delegation Token Store, which specifies storage for the Kerberos token as described below.
   - If you have an unsecured cluster, skip the next step.
3. Select org.apache.hadoop.hive.thrift.DBTokenStore, and save the change.

   Storage for the Kerberos delegation token is defined by the hive.cluster.delegation.token.store.class property. The available choices are Zookeeper, the Metastore, and memory. Cloudera recommends using the database by setting the org.apache.hadoop.hive.thrift.DBTokenStore property.
4. Click Instances Actions Add Role Instances



5. In Assign Roles, in Metastore Server, click Select Hosts.
6. In Hosts Selected, scroll and select the host that you want to serve as the backup Metastore, and click OK.
7. Click Continue until you exit the wizard.
8. Start the Metastore role on the host from the Actions menu.

   The hive.metastore.uris property is updated automatically. To verify, go to /etc/hive/config directory in your cluster node and look for the updated property in the hive-site.xml file.
9. To check or to change the hive.metastore.uri.selection property, go to Clusters Hive (aka Hive Metastore) Configurations in Cloudera Manager, and search for 'Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml'.
10. Add the property and value (SEQUENTIAL or RANDOM).

Related Information

Example of using the Cloudera Manager Safety Valve

Custom Configuration (about Cloudera Manager Safety Valve)

Setting up the metastore database

In CDP Private Cloud Base, you need to install a supported database for Hive metastore (HMS) to store the metadata. You configure Hive metastore by modifying hive-site.xml using the Cloudera Manager Safety Valve feature, described later, instead of using hive set key=value on the command line.

Related Information

Apache Wiki: Hive Metastore Administration
Example of using the Cloudera Manager Safety Valve

Set up the backend Hive metastore database

CDP Public Cloud supports only PostgreSQL for the backend Hive metastore database, and generally, you do not need to manually perform the installation. In CDP Private Cloud Base you need to install, start, and configure a backend database.

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 componenets. 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

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;
   -- For Package install method:
   mysql> SOURCE /usr/lib/hive/scripts/metastore/upgrade/mysql/hive-schema-
   n.n.n.mysql.sql;
   -- For Parcel install method:
   mysql> SOURCE /opt/cloudera/parcels/CDH/lib/hive/scripts/metastore/upgrad
   e/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
     -- For Package install method:
     ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-j
     dbc.jar
     -- For Parcel install method:
     ```

```
ln -s /usr/share/java/postgresql-jdbc.jar /opt/cloudera/parcels/CDH/lib/
hive/lib/postgresql-jdbc.jar
```

- SLES

```
sudo zypper install postgresql-jdbc
-- For Package install method:
ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-
jdbc.jar
-- For Parcel install method:
ln -s /usr/share/java/postgresql-jdbc.jar /opt/cloudera/parcels/CDH/lib/
hive/lib/postgresql-jdbc.jar
```

- Ubuntu

```
sudo apt-get install libpostgresql-jdbc-java
-- For Package install method:
ln -s /usr/share/java/postgresql-jdbc4.jar /usr/lib/hive/lib/postgresql-
jdbc4.jar
-- For Parcel install method:
ln -s /usr/share/java/postgresql-jdbc4.jar /opt/cloudera/parcels/CDH/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'.
-- For Package install method:
postgres=# \i /usr/lib/hive/scripts/metastore/upgrade/postgres/hive-schem
a-n.n.n.postgres.sql
-- For Parcel install method:
postgres=# \i /opt/cloudera/parcels/CDH/lib/hive/scripts/metastore/upgrad
e/postgres/hive-schema-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/ if you are using a Package install method or place the JAR in /opt/cloudera/parcels/CDH/lib/hive/lib/ if you are using a Parcel install method.

```
-- For Package install method:
sudo mv ojdbc<version_number>.jar /usr/lib/hive/lib/
-- For Parcel install method:
sudo mv ojdbc<version_number>.jar /opt/cloudera/parcels/CDH/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
-- For Package install method:
SQL> @/usr/lib/hive/scripts/metastore/upgrade/oracle/hive-schema-n.n.n.o
racle.sql
-- For Parcel install method:
SQL> @/opt/cloudera/parcels/CDH/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;

/
```

Configure 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. Using hive set key=value on the command line 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: jdbc:postgresql://myhost/metastore
- Oracle connection URL: jdbc:oracle:thin:@//myhost/xe

  Oracle driver name: oracle.jdbc.OracleDriver

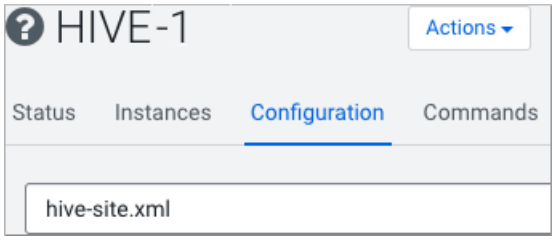
Before you begin

- 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

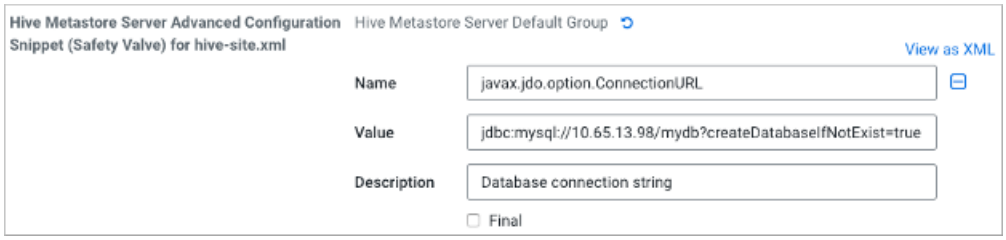
1. Find the fully qualified domain name or IP address of Hive metastore by navigating to Cloudera Manager Hosts Role(s) and looking through the list of roles to find Hive Hive Metastore Server.

2. Navigate to the metastore host configuration in Clusters HIVE-1 Configuration , and search for hive-site.xml.



HIVE-1 is the Hive metastore service.

3. In Hive Metastore Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click +, and add the name of the javax.jdo.option.ConnectionURL property.
4. 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>
```

5. Optionally, repeat the previous steps on all hosts in the cluster.
6. 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

Configure metastore location and HTTP mode

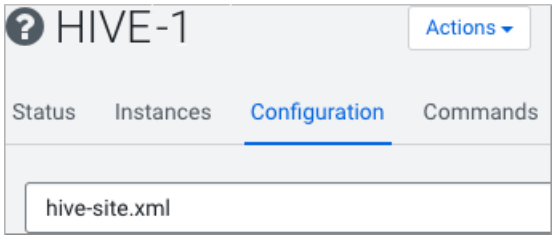
In addition to the database properties you need to set in CDP Private Cloud Base, you must configure the metastore URI property. This property defines one or more metastore locations. In CDP Public Cloud, you need to check the HTTP transport mode property. This property is set to http only for HiveServer and not for the metastore.

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

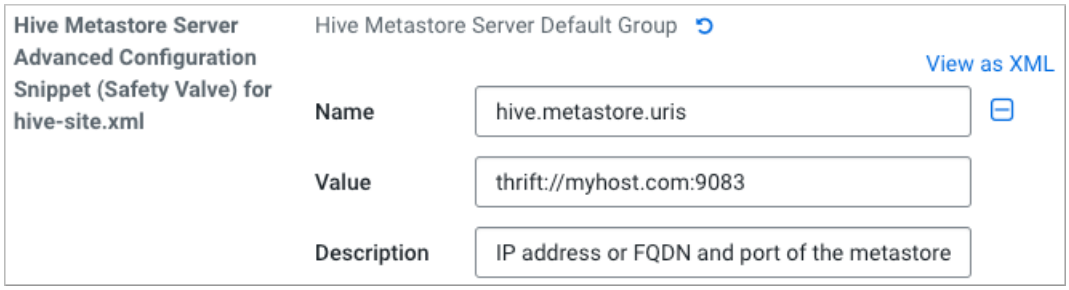
1. In Clusters HIVE-1 Configuration , search for hive-site.xml.



By default, the HIVE-1 includes the Hive metastore service.

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.

Set up a JDBC URL connection override

You can configure fine-grained tuning of the HMS database connection. You specify a JDBC URL override 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-1.

Tuning the metastore

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 in CDP Private Cloud Base

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.

HMS table storage

You need to understand how 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 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 transformatons 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.

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 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

Table requirement	Client meets requirements	Managed Table	ACID table type	Action
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, . . .
```

Setting up a shared Amazon RDS as a Hive metastore

CDP Private Cloud Base can share a single persistent instance of the Amazon Relational Database Service (RDS) as the Hive metastore (HMS) backend database. The persistence can extend beyond a cluster life cycle, elimintating the need for subsequent clusters to regenerate metadata.

Using a shared Amazon RDS server as your HMS backend, you can deploy and share data and metadata across multiple transient, as well as persistent, clusters subject to limitations listed below. For example, you can have multiple transient Hive or Apache Spark clusters writing table data and metadata that you can subsequently query from a persistent Apache Impala cluster. Or, you might have several different transient clusters, each dealing with different types of jobs on different data sets that spin up, read raw data from S3, do the ETL (Extract, Transform, Load) work, write data out to S3, and then spin down. In this scenario, you want each cluster to simply point to a permanent HMS and perform ETL. Using RDS as a shared HMS backend database reduces overhead because you no longer need to recreate the HMS for each cluster, every day, for each transient ETL job that you run.

Limitations

The following limitations apply to the jobs you run when you use an RDS server as a remote backend database for Hive metastore.

- No overlapping data or metadata changes to the same data sets across clusters.
- No reads during data or metadata changes to the same data sets across clusters.
- Overlapping data or metadata changes are defined when multiple clusters concurrently perform the following actions:
  - Make updates to the same table or partitions within the table located on S3.
  - Add or change the same parent schema or database.

Cloudera Support helps you as a licensed customers to repair any unexpected metadata issues. This support does not include root-cause analysis.

Set up Amazon RDS as a Hive metastore

In CDP Private Cloud Base, you can set up RDS as a Hive metastore using MySQL as a backend database. Other clusters that want to use the metastore do so without setting up a backend database.

Procedure

1. Follow instructions in Amazon documentation to create a MySQL instance with Amazon RDS.

2. Set up MySQL as a backend database for Hive metastore using the hostname, username, and password configured during your RDS setup.
3. Configure Hive, Impala, and Spark to use Amazon S3.