Configuring the Hive Metastore for CDH (#topic_18_4)

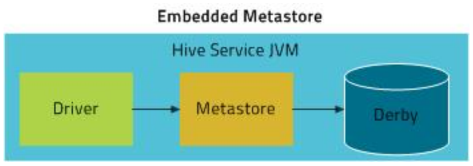
The HMS service stores the metadata for Hive tables and partitions in a relational database, and provides clients (including Hive) access to this information using the metastore service API. This page explains deployment options and provides instructions for setting up a database in a recommended configuration.

Continue reading:

Metastore Deployment Modes (#topic_18_4_1)

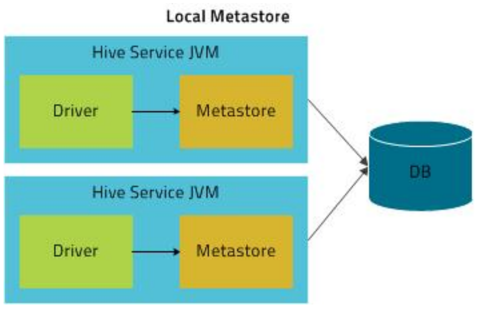
Embedded Mode (#topic_18_4_1__section_zzx_cvv_ls)

Cloudera recommends using this mode for experimental purposes only.



Embedded mode is the default metastore deployment mode for CDH. In this mode, the metastore uses a Derby database, and both the database and the metastore service are embedded in the main HiveServer2 process. Both are started for you when you start the HiveServer2 process. This mode requires the least amount of effort to configure, but it can support only one active user at a time and is not certified for production use.

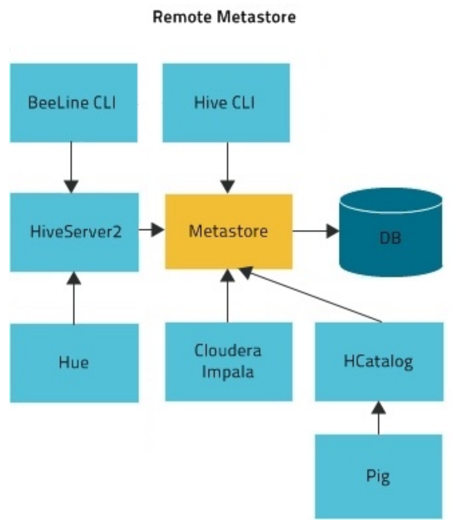
Local Mode (#topic_18_4_1__section_k2y_cvv_ls)



In Local mode, the Hive metastore service runs in the same process as the main HiveServer2 process, but the metastore database runs in a separate process, and can be on a separate host. The embedded metastore service communicates with the metastore database over JDBC.

Remote Mode (#topic_18_4_1__title_508)

Cloudera recommends that you use this mode.



In Remote mode, the Hive metastore service runs in its own JVM process. HiveServer2, HCatalog, Impala, and other processes communicate with it using the Thrift network API (configured using the `hive.metastore.uris` property). The metastore service communicates with the metastore database over JDBC (configured using the `javax.jdo.option.ConnectionURL` property). The database, the HiveServer2 process, and the metastore service can all be on the same host, but running the HiveServer2 process on a separate host provides better availability and scalability.

The main advantage of Remote mode over Local mode is that Remote mode does not require the administrator to share JDBC login information for the metastore database with each Hive user. HCatalog requires this mode.

Supported Metastore Databases [(#topic_18_4_2)](#topic_18_4_2)

For up-to-date information, see **[Database Requirements (rg_database_requirements.html#cdh_cm_supported_db)](rg_database_requirements.html#cdh_cm_supported_db)**. Cloudera strongly encourages you to use MySQL because it is the most popular with the rest of the Hive user community, and, hence, receives more testing than the other options. For installation information, see:

- **[Install and Configure MariaDB for Cloudera Software (install_cm_mariadb.html#install_cm_mariadb)](install_cm_mariadb.html#install_cm_mariadb)**
- **[Install and Configure MySQL for Cloudera Software (cm_ig_mysql.html#cmig_topic_5_5)](cm_ig_mysql.html#cmig_topic_5_5)**
- **[Install and Configure PostgreSQL for Cloudera Software (cm_ig_extrnl_pstgrs.html#cmig_topic_5_6)](cm_ig_extrnl_pstgrs.html#cmig_topic_5_6)**
- **[Install and Configure Oracle Database for Cloudera Software (cm_ig_oracle.html#cmig_topic_5_8)](cm_ig_oracle.html#cmig_topic_5_8)**

Metastore Memory and Hardware Requirements [(#concept_jsw_bnc_rp)](#concept_jsw_bnc_rp)

Component	**Java Heap**		**CPU**	**Disk**
HiveServer 2	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required for the following: • HiveServer2 log files • `stdout` and `stderr` output files • Configuration files • Operation logs stored in the `operation_logs_dir` directory, which is configurable • Any temporary files that might be created by local map tasks under the `/tmp` directory
	2-10 connections	4-6 GB		
	11-20 connections	6-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	Cloudera recommends splitting HiveServer2 into multiple instances and load balancing them once you start allocating more than 16 GB to HiveServer2. The objective is to adjust the size to reduce the impact of Java garbage collection on active processing by the service.			

Component	Java Heap		CPU	Disk
	Set this value using the Java Heap Size of HiveServer2 in Bytes Hive configuration property.			
Hive Metastore	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required so that the Hive metastore can store the following artifacts: • Logs • Configuration files • Backend database that is used to store metadata if the database server is also hosted on the same node
	2-10 connections	4-10 GB		
	11-20 connections	10-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	Set this value using the Java Heap Size of Hive Metastore Server in Bytes Hive configuration property.			
Beeline CLI	Minimum: 2 GB		N/A	N/A

Important: These numbers are general guidance only, and can be affected by factors such as number of columns, partitions, complex joins, and client activity. Based on your anticipated deployment, refine through testing to arrive at the best values for your environment.

For information on configuring heap for the Hive metastore, as well as HiveServer2 and Hive clients, see **Tuning Apache Hive in CDH (admin_hive_tuning.html#concept_u51_lkv_cv)** .

General Metastore Tuning Recommendations (#tune_metastore_recommendations)

Generally, you need to limit concurrent connections to Hive metastore. A large number of open connections affects performance as does issues with the backend database, improper Hive use, such as extremely complex queries, a connection leak, and other issues. Try making the following changes:

- Buy an SSD for one or more Hive metastores.
- Cloudera recommends that a single query access 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.

Configuring the Metastore Database (#configure_metastore_db)

This section describes how to configure Hive to use a remote database, with examples for MySQL (cdh_ig_hive_metastore_configure.html#configure_mysql_db_hive_metastore), PostgreSQL (cdh_ig_hive_metastore_configure.html#configure_postgresql_db_hive_metastore), and Oracle (cdh_ig_hive_metastore_configure.html#configure_oracle_db_hive_metastore).

The configuration properties for the Hive metastore are documented in the Hive Metastore Administration documentation (https://cwiki.apache.org/confluence/display/Hive/AdminManual+Metastore+Administration) on the Apache wiki.

Note: For information about additional configuration that may be needed in a secure cluster, see Hive Authentication (cdh_sg_hive_security.html#topic_9).

Configuring a Remote MySQL Database for the Hive Metastore (#configure_mysql_db_hive_metastore)

Cloudera recommends you configure a database for the metastore on one or more remote servers that reside on a host or hosts separate from the HiveServer2 process. MySQL is the most popular database to use. Use the following steps to configure a remote metastore. If you are planning to use a cloud service database, such as Amazon Relational Database Service (RDS), see Configuring a Shared Amazon RDS as an HMS for CDH (admin_hive_config_amzn_shared_rds.html#config_hive_amz_shared_rds) for information about how to set up a shared Amazon RDS as your Hive metastore.

1. **Install and start MySQL if you have not already done so**
   To install MySQL on a RHEL system:

   ```
   $ sudo yum install mysql-server
   ```

   To install MySQL on a SLES system:

   ```
   $ sudo zypper install mysql
   $ sudo zypper install libmysqlclient_r17
   ```

   To install MySQL on a Debian/Ubuntu system:

   ```
   $ sudo apt-get install mysql-server
   ```

   After using the command to install MySQL, you may need to respond to prompts to confirm that you do want to complete the installation. After installation completes, start the `mysql` daemon.
   On RHEL systems

   ```
   $ sudo service mysqld start
   ```

   On SLES and Debian/Ubuntu systems

   ```
   $ sudo service mysql start
   ```

2. **Configure the MySQL service and JDBC driver**

   Before you can run the Hive metastore with a remote MySQL database, you must install the MySQL JDBC driver, set up the initial database schema, and configure the MySQL user account for the Hive user.

   For instructions on installing the MySQL JDBC driver, see Installing the MySQL JDBC Driver (cm_ig_mysql.html#cmig_topic_5_5_3).

   Configure MySQL to use a strong password and to start at boot. Note that in the following procedure, your current `root` password is blank. Press the Enter key when you're prompted for the root password.

   To 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!
```

To make sure the MySQL server starts at boot:

- On RHEL systems:

```
$ sudo /sbin/chkconfig mysqld on
sudo /sbin/chkconfig --list mysqld
mysqld          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

- On SLES systems:

```
$ sudo chkconfig --add mysql
```

- On Debian/Ubuntu systems:

```
$ sudo chkconfig mysql on
```

3. **Create the database and user**

The instructions in this section assume you are using **Remote mode (cdh_ig_hive_metastore_configure.html#topic_18_4_1__title_508)**, and that the MySQL database is installed on a separate host from the metastore service, which is running on a host named `metastorehost` in the example.

Note: If the metastore service will run on the host where the database is installed, replace `'metastorehost'` in the `CREATE USER` example with `'localhost'`. Similarly, the value of `javax.jdo.option.ConnectionURL` in `/etc/hive/conf/hive-site.xml` (discussed in the next step) must be `jdbc:mysql://localhost/metastore`. For more information on adding MySQL users, see **http://dev.mysql.com/doc/refman/5.5/en/adding-users.html (http://dev.mysql.com/doc/refman/5.5/en/adding-users.html)**.

Create the initial database schema. Cloudera recommends using the **Metastore Schema Tool (cdh_ig_hive_schema_tool.html#metastore_schema_tool)** to do this.

If for some reason you decide not to use the schema tool, you can use the `hive-schema-n.n.n.mysql.sql` file instead; that file is located in the `/usr/lib/hive/scripts/metastore/upgrade/mysql/` directory. (*n.n.n* is the current Hive version, for example 1.1.0.) Proceed as follows if you decide to use `hive-schema-n.n.n.mysql.sql`.

Example using hive-schema-*n.n.n*mysql.sql

Note: Do this only if you are not using the Hive schema tool.

```
$ 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;
```

You also need a MySQL user account for Hive to use to access the metastore. It is very important to prevent this user account from creating or altering tables in the metastore database schema.

Important: To prevent users from inadvertently corrupting the metastore schema when they use lower or higher versions of Hive, set the `hive.metastore.schema.verification` property to true in `/usr/lib/hive/conf/hive-site.xml` on the metastore host.

Example

```
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;
```

4. **Configure the metastore service to communicate with the MySQL database**

   This step shows the configuration properties you need to set in `hive-site.xml` (/usr/lib/hive/conf/hive-site.xml) to configure the metastore service to communicate with the MySQL database, and provides sample settings. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer2), `hive.metastore.uris` is the only property that **must** be configured on all of them; the others are used only on the metastore host.

   Given a MySQL database running on `myhost` and the user account `hive` with the password `mypassword`, set the configuration as follows (overwriting any existing values).

   Note: The `hive.metastore.local` property is no longer supported (as of Hive 0.10); setting `hive.metastore.uris` is sufficient to indicate that you are using a remote metastore.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://myhost/metastore</value>
  <description>the URL of the MySQL database</description>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</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.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore host</description>
</property>

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

Configuring a Remote PostgreSQL Database for the Hive Metastore (#configure_postgresql_db_hive_metastore)

Before you can run the Hive metastore with a remote PostgreSQL database, you must configure a connector to the remote PostgreSQL database, set up the initial database schema, and configure the PostgreSQL user account for the Hive user.

1. **Install and start PostgreSQL if you have not already done so**

To install PostgreSQL on a RHEL system:

```
$ sudo yum install postgresql-server
```

To install PostgreSQL on a SLES system:

```
$ sudo zypper install postgresql-server
```

To install PostgreSQL on a Debian/Ubuntu system:

```
$ sudo apt-get install postgresql
```

After using the command to install PostgreSQL, you may need to respond to prompts to confirm that you do want to complete the installation. In order to finish installation on RHEL compatible systems, you need to initialize the database. Please note that this operation is not needed on Ubuntu and SLES systems as it's done automatically on first start:

To initialize database files on RHEL compatible systems

```
$ sudo service postgresql initdb
```

To ensure that your PostgreSQL server will be accessible over the network, you need to do some additional configuration.

First you need to edit the `postgresql.conf` file. Set the `listen_addresses` property to `*`, to make sure that the PostgreSQL server starts listening on all your network interfaces. Also make sure that the `standard_conforming_strings` property is set to `off`.

You can check that you have the correct values as follows:

On Red-Hat-compatible systems:

```
$ sudo cat /var/lib/pgsql/data/postgresql.conf  | grep -e listen -e standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

On SLES systems:

```
$ sudo cat /var/lib/pgsql/data/postgresql.conf  | grep -e listen -e standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

On Ubuntu and Debian systems:

```
$ cat /etc/postgresql/9.1/main/postgresql.conf | grep -e listen -e standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

You also need to configure authentication for your network in `pg_hba.conf`. You need to make sure that the PostgreSQL user that you will create later in this procedure will have access to the server from a remote host. To do this, add a new line into `pg_hba.con` that has the following information:

```
host    <database>         <user>         <network address>         <mask>               md5
```

The following example allows all users to connect from all hosts to all your databases:

```
host    all         all         0.0.0.0         0.0.0.0              md5
```

Note: This configuration is applicable only for a network listener. Using this configuration does not open all your databases to the entire world; the user must still supply a password to authenticate himself, and privilege restrictions configured in PostgreSQL will still be applied.

After completing the installation and configuration, you can start the database server:

Start PostgreSQL Server

```
$ sudo service postgresql start
```

Use `chkconfig` utility to ensure that your PostgreSQL server will start at a boot time. For example:

```
chkconfig postgresql on
```

You can use the `chkconfig` utility to verify that PostgreSQL server will be started at boot time, for example:

```
chkconfig --list postgresql
```

2. **Install the PostgreSQL JDBC driver**

   Before you can run the Hive metastore with a remote PostgreSQL database, you must configure a JDBC driver to the remote PostgreSQL database, set up the initial database schema, and configure the PostgreSQL user account for the Hive user.

   To install the PostgreSQL JDBC Driver on a RHEL 6 system:

   On the Hive metastore server host, install `postgresql-jdbc` package and create symbolic link to the `/usr/lib/hive/lib/` directory. For example:

```
$ sudo yum install postgresql-jdbc
$ ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-jdbc.jar
```

   To install the PostgreSQL JDBC Driver on a SLES system:

   On the Hive metastore server host, install `postgresql-jdbc` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

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

   To install the PostgreSQL JDBC Driver on a Debian/Ubuntu system:

   On the Hive metastore server host, install `libpostgresql-jdbc-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

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

3. **Create the metastore database and user account**

   Proceed as in the following example, using the appropriate script in /usr/lib/hive/scripts/metastore/upgrade/postgres/ *n.n.n* is the current Hive version, for example 1.1.0:

```
$ 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-schema-n.n.n.postgres.sql
SET
SET
...
```

   Now you need to grant permission for all metastore tables to user `hiveuser`. PostgreSQL does not have statements to grant the permissions for all tables at once; you'll need to grant the permissions one table at a time. You could automate the task with the following SQL script: Note: If you are running these commands interactively and are still in the Postgres session initiated at the beginning of this step, you do not need to repeat `sudo -u postgres psql`.

```
bash# sudo -u postgres psql
metastore=# \c metastore
metastore=# \pset tuples_only on
metastore=# \o /tmp/grant-privs
metastore=#   SELECT 'GRANT SELECT,INSERT,UPDATE,DELETE ON "'  || schemaname || '". "' ||tablename ||'" TO hiveuser ;'
metastore-#   FROM pg_tables
metastore-#   WHERE tableowner = CURRENT_USER and schemaname = 'public';
metastore=# \o
metastore=# \pset tuples_only off
metastore=# \i /tmp/grant-privs
```

You can verify the connection from the machine where you'll be running the metastore service as follows:

```
psql -h myhost -U hiveuser -d metastore
metastore=#
```

4. **Configure the metastore service to communicate with the PostgreSQL database**

   This step shows the configuration properties you need to set in `hive-site.xml` (/usr/lib/hive/conf/hive-site.xml) to configure the metastore service to communicate with the PostgreSQL database. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer2), `hive.metastore.uris` is the only property that **must** be configured on all of them; the others are used only on the metastore host.

   Given a PostgreSQL database running on host `myhost` under the user account `hive` with the password `mypassword`, you would set configuration properties as follows.

   Note:

   - The instructions in this section assume you are using **Remote mode (cdh_ig_hive_metastore_configure.html#topic_18_4_1__title_508)**, and that the PostgreSQL database is installed on a separate host from the metastore server.
   - The `hive.metastore.local` property is no longer supported as of Hive 0.10; setting `hive.metastore.uris` is sufficient to indicate that you are using a remote metastore.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://myhost/metastore</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hiveuser</value>
</property>

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

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

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore host</description>
</property>

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

5. **Test connectivity to the metastore**

```
$ hive –e “show tables;”
```

   Note: This will take a while the first time.

Configuring a Remote Oracle Database for the Hive Metastore (#configure_oracle_db_hive_metastore)

Before you can run the Hive metastore with a remote Oracle database, you must configure a connector to the remote Oracle database, set up the initial database schema, and configure the Oracle user account for the Hive user.

1. **Install and start Oracle**

   The Oracle database is not part of any Linux distribution and must be purchased, downloaded and installed separately. You can use the **Express edition (http://www.oracle.com/technetwork/database/database-technologies/express-edition/overview/index.html)** , which can be downloaded free from the Oracle website.

2. **Install the Oracle JDBC Driver**

   You must download the Oracle JDBC Driver from the Oracle website and put the JDBC JAR file into the `/usr/lib/hive/lib/` directory. For example, the version 6 JAR file is named `ojdbc6.jar`. To download the JDBC driver, visit the **Oracle JDBC and UCP Downloads (http://www.oracle.com/technetwork/database/application-development/jdbc/downloads/index.html)** page, and click on the link for your Oracle Database version. Download the `ojdbc6.jar` file (or `ojdbc8.jar`, for Oracle Database 12.2).

   Note: This URLs was correct at the time of publication, but can change.

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

3. **Create the metastore database and user account**

   Connect to your Oracle database as an administrator and create the user that will use the Hive metastore.

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

   Connect as the newly created `hiveuser` user and load the initial schema, as in the following example. Use the appropriate script for the current release (for example hive-schema-1.1.0.oracle.sql) in /usr/lib/hive/scripts/metastore/upgrade/oracle/ :

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

   Connect back as an administrator and remove the power privileges from user `hiveuser`. Then grant limited access to all the tables:

   ```
   $ sqlplus "sys as sysdba"
   SQL> revoke all privileges from hiveuser;
   SQL> BEGIN
     2     FOR R IN (SELECT owner, table_name FROM all_tables WHERE owner='HIVEUSER') LOOP
     3        EXECUTE IMMEDIATE 'grant  SELECT,INSERT,UPDATE,DELETE on '||R.owner||'.'||R.table_name||' to hiveuser';
     4     END LOOP;
     5  END;
     6
     7  /
   ```

4. **Configure the metastore service to Communicate with the Oracle Database**

   This step shows the configuration properties you need to set in `hive-site.xml` (/usr/lib/hive/conf/hive-site.xml) to configure the metastore service to communicate with the Oracle database, and provides sample settings. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer2), `hive.metastore.uris` is the only property that must be configured on all of them; the others are used only on the metastore host.

   Example

   Given an Oracle database running on `myhost` and the user account `hiveuser` with the password `mypassword`, set the configuration as follows (overwriting any existing values):

   ```
   <property>
     <name>javax.jdo.option.ConnectionURL</name>
     <value>jdbc:oracle:thin:@//myhost/xe</value>
   </property>

   <property>
     <name>javax.jdo.option.ConnectionDriverName</name>
     <value>oracle.jdbc.OracleDriver</value>
   </property>

   <property>
     <name>javax.jdo.option.ConnectionUserName</name>
     <value>hiveuser</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>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore host</description>
</property>

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

Specifying a JDBC URL Override for Database Connections (#jdbc_url_override)

In instances where you wish to configure fine-grained tuning of the HMS database connection, you can specify a JDBC URL override to be used when establishing a connection to the HMS database.
Warning:
This configuration setting is intended for advanced database users only. Be aware that when using this override, the following properties are overwritten (in other words, their values will *not* be used):

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

Prerequisites

- The required default user role is Configurator (cm_sg_user_roles.html#concept_wfh_tvy_qp__configurator) .
- When using the **Hive Metastore Database JBC URL Override**, you *must* still provide the following properties to connect to the database:
  - Hive Metastore Database Type
  - Hive Metastore Database User
  - Hive Metastore Database Password

To specify a Hive Metastore JDBC URL Override for database connections:

1. Open the Cloudera Manager Admin Console and go to the **Hive-1 service**.
2. Click the **Configuration** tab.
3. Select Category > Hive Metastore Database.
4. Edit the **Hive Metastore Database JDBC URL Override** property according to your cluster configuration (the default value is Empty `""`):

Database Type	**Hive Metastore Database JDBC URL Override Format**
MySQL	`jdbc:mysql://<host>:<port>/<metastore_db>?key=value`
PostgreSQL	`jdbc:postgresql://<host>:<port>/<metastore_db>?key=value`

Database Type	Hive Metastore Database JDBC URL Override Format
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>`

Important: Formats are dependent on the JDBC driver version that you are using and subject to change between releases. Refer to your database product documentation to confirm JDBC formats for the specific database version you are using.

Categories: Configuring (../categories/hub_configuring.html) | Databases (../categories/hub_databases.html) | Deploying (../categories/hub_deploying.html) | HMS (../categories/hub_hms.html) | Hive (../categories/hub_hive.html) | How To (../categories/hub_how_to.html) | Memory (../categories/hub_memory.html) | Metastore (../categories/hub_metastore.html) | Planning (../categories/hub_planning.html) | Requirements (../categories/hub_requirements.html) | All Categories (../categories/hub.html)