Cloudera Runtime 1

Start Hive and run commands

Date of Publish: 2019-04-15

https://docs.hortonworks.com/

Contents

Start the Hive Shell

You use Beeline to launch the Hive shell, Data Analytics Studio (DAS), or another Hive UI. You then query Hive as an end user authorized by Apache Ranger. As administrator, you must first set up the user in the operating system and in Ranger.

Before you begin

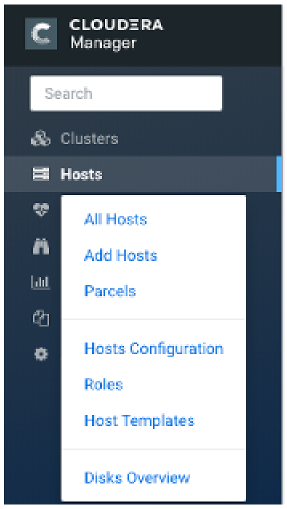
If your cluster is not set up with security, skip these steps.

- Create a user. For example, add a Linux user using the adduser operating system command.
- Add the user to the Ranger list of users.
- In Ranger, add the user name to Hive policies that grant full access to Hive.
- In Ranger, add the user name to the HDFS all-path policy.
- In Cloudera Manager > Hive > Configuration, search for and enable HiveServer2 Enable Impersonation.

  This is equivalent to setting hive.server2.enable.doAs=true in hive-site.xml.

Procedure

1. In Cloudera Manager Home, click Hosts > All Hosts.

2. Make a note of the IP address or host name of a node in your cluster, for example myhost-1.com
3. Use ssh to log into the cluster.
   For example:

```
ssh myhost-1.com
```

4. Get help about starting the Hive shell.
   On the command line, type

```
hive -h
```

   Output is:

```
Connect using simple authentication to HiveServer2 on localhost:10000
beeline -u jdbc:hive2://localhost:10000 username password

Connect using simple authentication to HiveServer2 on hs.local:10000 using
 -n for username and -p for password
beeline -n username -p password -u jdbc:hive2://hs2.local:10012
```

```
Connect using Kerberos authentication with hive/localhost@mydomain.com as
 HiveServer2 principal
beeline -u "jdbc:hive2://hs2.local:10013/default;principal=hive/
localhost@mydomain.com"

Connect using SSL connection to HiveServer2 on localhost at 10000
beeline "jdbc:hive2://localhost:10000/default;ssl=true;sslTrustStore=/usr/
local/truststore;trustStorePassword=mytruststorepassword"

Connect using LDAP authentication
beeline -u jdbc:hive2://hs2.local:10013/default <ldap-username> <ldap-
password>
```

5. Start Hive from the command line. using your user name and Alternatively

   - Use your user name if your cluster security is set up.
   - Use the user name hive and no password.

   Substitute the name or IP address of your HiveServer host.

   beeline -u jdbc:hive2://mycloudhost-3.com:10000 -n <your user name> -p
6. Enter your password at the prompt.

   Depending on the security in your cluster, you might omit the -p password option.
7. Enter a Hive query.

   ```
   SHOW DATABASES;
   ```
8. Create a table in the default database.

   ```
   CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3,2));
   ```
9. Insert data into the table.

   ```
   INSERT INTO TABLE students VALUES ('fred flintstone', 35, 1.28), ('barney
    rubble', 32, 2.32);
   ```

Related Information

Configure a Resource-based Policy: Hive

Run a Hive command

You can run most Hive commands that push configuration variables to Hive SQL scripts from the command line of a node in your cluster. The hive keyword, which launches Beeline in the background, precedes the command.

About this task

Hive supports running Hive commands from the command line using Beeline only. In the task below, you start Beeline in the background and enter the -e flag followed by a Hive set command that lists system variables.

Before you begin

- You set up access to the Data Warehouse from an external machine, such as your laptop.

Procedure

1. From an external machine, connect to the Data Warehouse.

2. On a node in your cluster, enter the hive command to send configuration properties to standard output.

```
> hive -e set
```

Supported commands appear. All obsolete Hive CLI commands are supported on the Beeline command line except set key=value commands that configure Hive Metastore.

The output includes the system variable settings:

```
+----------------------------------------------------------------+
|                             set                                |
+----------------------------------------------------------------+
| _hive.hdfs.session.path=/tmp/hive/hive/91ecb...00a             |
| _hive.local.session.path=/tmp/hive/91ecb...00a                 |
                         |
 ...
```

Convert Hive CLI scripts to Beeline

If you have legacy scripts that execute Hive queries from edge nodes using the Hive CLI, you must solve potential incompatibilities with variable substitution in these scripts. The Data Warehouse supports Beeline instead of Hive CLI. You can use Beeline to run legacy scripts with a few caveats.

About this task

In this task, you resolve incompatibilities in legacy Hive CLI scripts and Beeline:

- Configuration variables
  - Problem: You cannot refer to configuration parameters in scripts using the hiveconf namespace unless allowed.
  - Solution: You include the parameter in the HiveServer whitelist.
- Namespace problems
  - Problem: Beeline does not support the system and env namespaces for variables.
  - Solution: You remove these namespace references from scripts using a conversion technique described in this task.

Procedure

1. Create a conversion script named env_to_hivevar.sh that removes env references in your SQL scripts.

```
#!/usr/bin/env bash

CMD_LINE=""

#Blank conversion of all env scoped values
for I in `env`; do
  CMD_LINE="$CMD_LINE --hivevar env:${I} "
done
echo ${CMD_LINE}
```

2. On the command line of a node in your cluster, define and export a variable named HIVEVAR, for example, and set it to execute the conversion script.

```
export HIVEVAR=`./env_to_hivevar.sh`
```

3. Define and export variables to hold a few variables for testing the conversion.

```
export LOC_TIME_ZONE="US/EASTERN"
export MY_TEST_VAR="TODAY"
```

4. On the command line of a cluster node, test the conversion: Execute a command that references HIVEVAR to parse a SQL statement, remove the incompatible env namespace, and execute the remaining SQL.

```
hive ${HIVEVAR} -e 'select "${env:LOC_TIME_ZONE}";'
```

```
+-------------+
|     _c0     |
+-------------+
| US/EASTERN  |
+-------------+
```

5. Create a text file named init_var.sql to simulate a legacy script that sets two configuration parameters, one in the problematic env namespace.

```
set mylocal.test.var=hello;
set mylocal.test.env.var=${env:MY_TEST_VAR};
```

6. Whitelist these configuration parameters: In Ambari, go to Hive > Configs > Advanced > Custom hiveserver2-site.

7. Add the property key: hive.security.authorization.sqlstd.confwhitelist.append.

8. Provide the property value, or values, to whitelist, for example: mylocal\..*|junk.

This action appends mylocal.test.var and mylocal.test.env.var parameters to the whitelist.

9. Save configuration changes, and restart any components as required.

10. Execute a command that references HIVEVAR to parse a SQL script, removes the incompatible env namespace, and executes the remaining SQL, including the whitelisted configuration parameters identified by hiveconf:.

```
hive -i init_var.sql ${HIVEVAR} -e 'select
 "${hiveconf:mylocal.test.var}","${hiveconf:mylocal.test.env.var}";'
```

```
+--------+--------+
|  _c0   |  _c1   |
+--------+--------+
| hello  | TODAY  |
+--------+--------+
```