LanguageManual Cli

Hive CLI

$HIVE_HOME/bin/hive is a shell utility which can be used to run Hive queries in either interactive or batch mode.

Deprecation in favor of Beeline CLI

HiveServer2 (introduced in Hive 0.11) has its own CLI called Beeline, which is a JDBC client based on SQLLine. Due to new development being focused on HiveServer2, Hive CLI will soon be deprecated in favor of Beeline (HIVE-10511).

See Replacing the Implementation of Hive CLI Using Beeline and Beeline – New Command Line Shell in the HiveServer2 documentation.

Hive Command Line Options

To get help, run "`hive -H`" or "`hive --help`".
Usage (as it is in Hive 0.9.0):

```
usage: hive
 -d,--define <key=value>          Variable substitution to apply to Hive
                                  commands. e.g. -d A=B or --define A=B
 -e <quoted-query-string>         SQL from command line
 -f <filename>                    SQL from files
 -H,--help                        Print help information
 -h <hostname>                    Connecting to Hive Server on remote host
    --hiveconf <property=value>   Use value for given property
    --hivevar <key=value>         Variable substitution to apply to hive
                                  commands. e.g. --hivevar A=B
 -i <filename>                    Initialization SQL file
 -p <port>                        Connecting to Hive Server on port number
 -S,--silent                      Silent mode in interactive shell
 -v,--verbose                     Verbose mode (echo executed SQL to the
                                  console)
```

Version information

As of Hive 0.10.0 there is one additional command line option:

```
--database <dbname>      Specify the database to use
```

Note: The variant "`-hiveconf`" is supported as well as "`--hiveconf`".

Examples

See Variable Substitution for examples of using the `hiveconf` option.

- Example of running a query from the command line

```
$HIVE_HOME/bin/hive -e 'select a.col from tab1 a'
```

- Example of setting Hive configuration variables

```
$HIVE_HOME/bin/hive -e 'select a.col from tab1 a' --hiveconf hive.exec.scratchdir=/home/my/hive_scratch
--hiveconf mapred.reduce.tasks=32
```

- Example of dumping data out from a query into a file using silent mode

```
$HIVE_HOME/bin/hive -S -e 'select a.col from tab1 a' > a.txt
```

- Example of running a script non-interactively from local disk

```
$HIVE_HOME/bin/hive -f /home/my/hive-script.sql
```

- Example of running a script non-interactively from a Hadoop supported filesystem (starting in Hive 0.14)

```
$HIVE_HOME/bin/hive -f hdfs://<namenode>:<port>/hive-script.sql
$HIVE_HOME/bin/hive -f s3://mys3bucket/s3-script.sql
```

- Example of running an initialization script before entering interactive mode

```
$HIVE_HOME/bin/hive -i /home/my/hive-init.sql
```

The hiverc File

The CLI when invoked without the `-i` option will attempt to load $HIVE_HOME/bin/.hiverc and $HOME/.hiverc as initialization files.

Logging

Hive uses log4j for logging. These logs are not emitted to the standard output by default but are instead captured to a log file specified by Hive's log4j properties file. By default Hive will use `hive-log4j.default` in the `conf/` directory of the Hive installation which writes out logs to `/tmp/<userid>/hive.log` and uses the `WARN` level.

It is often desirable to emit the logs to the standard output and/or change the logging level for debugging purposes. These can be done from the command line as follows:

```
$HIVE_HOME/bin/hive --hiveconf hive.root.logger=INFO,console
```

`hive.root.logger` specifies the logging level as well as the log destination. Specifying `console` as the target sends the logs to the standard error (instead of the log file).

See Hive Logging in Getting Started for more information.

Tool to Clear Dangling Scratch Directories

See Scratch Directory Management in Setting Up HiveServer2 for information about scratch directories and a command-line tool for removing dangling scratch directories that can be used in the Hive CLI as well as HiveServer2.

Hive Batch Mode Commands

When `$HIVE_HOME/bin/hive` is run with the `-e` or `-f` option, it executes SQL commands in batch mode.

- `hive -e '<query-string>'` executes the query string.
- `hive -f <filepath>` executes one or more SQL queries from a file.

> **Version 0.14**
>
> As of Hive 0.14, <filepath> can be from one of the Hadoop supported filesystems (HDFS, S3, etc.) as well.
>
> ```
> $HIVE_HOME/bin/hive -f hdfs://<namenode>:<port>/hive-script.sql
> $HIVE_HOME/bin/hive -f s3://mys3bucket/s3-script.sql
> ```
>
> See HIVE-7136 for more details.

Hive Interactive Shell Commands

When `$HIVE_HOME/bin/hive` is run without either the `-e` or `-f` option, it enters interactive shell mode.

Use ";" (semicolon) to terminate commands. Comments in scripts can be specified using the "--" prefix.

Command	Description
quit exit	Use quit or exit to leave the interactive shell.
reset	Resets the configuration to the default values (as of Hive 0.10: see HIVE-3202).
set <key>=<value>	Sets the value of a particular configuration variable (key). **Note:** If you misspell the variable name, the CLI will not show an error.
set	Prints a list of configuration variables that are overridden by the user or Hive.
set -v	Prints all Hadoop and Hive configuration variables.
add FILE[S] <filepath> <filepath>* add JAR[S] <filepath> <filepath>* add ARCHIVE[S] <filepath> <filepath>*	Adds one or more files, jars, or archives to the list of resources in the distributed cache. See Hive Resources below for more information.
add FILE[S] <ivyurl> <ivyurl>* add JAR[S] <ivyurl> <ivyurl>* add ARCHIVE[S] <ivyurl> <ivyurl>*	As of Hive 1.2.0, adds one or more files, jars or archives to the list of resources in the distributed cache using an Ivy URL of the form ivy://group:module:version?query_string. See Hive Resources below for more information.
list FILE[S] list JAR[S] list ARCHIVE[S]	Lists the resources already added to the distributed cache. See Hive Resources below for more information.
list FILE[S] <filepath>* list JAR[S] <filepath>* list ARCHIVE[S] <filepath>*	Checks whether the given resources are already added to the distributed cache or not. See Hive Resources below for more information.
delete FILE[S] <filepath>* delete JAR[S] <filepath>* delete ARCHIVE[S] <filepath>*	Removes the resource(s) from the distributed cache.
delete FILE[S] <ivyurl> <ivyurl>* delete JAR[S] <ivyurl> <ivyurl>* delete ARCHIVE[S] <ivyurl> <ivyurl>*	As of Hive 1.2.0, removes the resource(s) which were added using the <ivyurl> from the distributed cache. See Hive Resources below for more information.
! <command>	Executes a shell command from the Hive shell.
dfs <dfs command>	Executes a dfs command from the Hive shell.
<query string>	Executes a Hive query and prints results to standard output.
source <filepath>	Executes a script file inside the CLI.

Sample Usage:

```
hive> set mapred.reduce.tasks=32;
hive> set;
hive> select a.* from tab1;
hive> !ls;
hive> dfs -ls;
```

Hive Resources

Hive can manage the addition of resources to a session where those resources need to be made available at query execution time. The resources can be files, jars, or archives. Any locally accessible file can be added to the session.

Once a resource is added to a session, Hive queries can refer to it by its name (in map/reduce/transform clauses) and the resource is available locally at execution time on the entire Hadoop cluster. Hive uses Hadoop's Distributed Cache to distribute the added resources to all the machines in the cluster at query execution time.

Usage:

```
ADD { FILE[S] | JAR[S] | ARCHIVE[S] } <filepath1> [<filepath2>]*
LIST { FILE[S] | JAR[S] | ARCHIVE[S] } [<filepath1> <filepath2> ..]
DELETE { FILE[S] | JAR[S] | ARCHIVE[S] } [<filepath1> <filepath2> ..]
```

- FILE resources are just added to the distributed cache. Typically, this might be something like a transform script to be executed.
- JAR resources are also added to the Java classpath. This is required in order to reference objects they contain such as UDFs. See Hive Plugins for more information about custom UDFs.
- ARCHIVE resources are automatically unarchived as part of distributing them.

Example:

```
hive> add FILE /tmp/tt.py;
hive> list FILES;
/tmp/tt.py
hive> select from networks a
            MAP a.networkid
            USING 'python tt.py' as nn where a.ds = '2009-01-04' limit 10;
```

Version 1.2.0

As of Hive 1.2.0, resources can be added and deleted using Ivy URLs of the form ivy://group:module:version?query_string.

- *group* – Which module group the module comes from. Translates directly to a Maven groupId or an Ivy Organization.
- *module* – The name of the module to load. Translates directly to a Maven artifactId or an Ivy artifact.
- *version* – The version of the module to use. Any version or * (for latest) or an Ivy Range can be used.

Various parameters can be passed in the *query_string* to configure how and which jars are added to the artifactory. The parameters are in the form of key value pairs separated by '&'.

Usage:

```
ADD { FILE[S] | JAR[S] | ARCHIVE[S] } <ivy://org:module:version?key=value&key=value&...> <ivy://org:
module:version?key=value&key1=value1&...>*
DELETE { FILE[S] | JAR[S] | ARCHIVE[S] } <ivy://org:module:version> <ivy://org:module:version>*
```

Also, we can mix <ivyurl> and <filepath> in the same ADD and DELETE commands.

```
ADD { FILE[S] | JAR[S] | ARCHIVE[S] } { <ivyurl> | <filepath> } <ivyurl>* <filepath>*
DELETE { FILE[S] | JAR[S] | ARCHIVE[S] } { <ivyurl> | <filepath> } <ivyurl>* <filepath>*
```

The different parameters that can be passed are:

1. *exclude*: Takes a comma separated value of the form org:module.
2. *transitive*: Takes values true or false. Defaults to true. When transitive = true, all the transitive dependencies are downloaded and added to the classpath.
3. *ext*: The extension of the file to add. 'jar' by default.
4. *classifier*: The maven classifier to resolve by.

Examples:

```
hive>ADD JAR ivy://org.apache.pig:pig:0.10.0?exclude=org.apache.hadoop:avro;
hive>ADD JAR ivy://org.apache.pig:pig:0.10.0?exclude=org.apache.hadoop:avro&transitive=false;
```

The DELETE command will delete the resource and all its transitive dependencies unless some dependencies are shared by other resources. If two resources share a set of transitive dependencies and one of the resources is deleted using the DELETE syntax, then all the transitive dependencies will be deleted for the resource except the ones which are shared.

Examples:

```
hive>ADD JAR ivy://org.apache.pig:pig:0.10.0
hive>ADD JAR ivy://org.apache.pig:pig:0.11.1.15
hive>DELETE JAR ivy://org.apache.pig:pig:0.10.0
```

If A is the set containing the transitive dependencies of pig-0.10.0 and B is the set containing the transitive dependencies of pig-0.11.1.15, then after executing the above commands, A-(A intersection B) will be deleted.

See HIVE-9664 for more details.

It is not neccessary to add files to the session if the files used in a transform script are already available on all machines in the Hadoop cluster using the same path name. For example:

- `... MAP a.networkid USING 'wc -l' ...`
  Here `wc` is an executable available on all machines.
- `... MAP a.networkid USING '/home/nfsserv1/hadoopscripts/tt.py' ...`
  Here `tt.py` may be accessible via an NFS mount point that's configured identically on all the cluster nodes.

Note that Hive configuration parameters can also specify jars, files, and archives. See Configuration Variables for more information.

HCatalog CLI

Version

HCatalog is installed with Hive, starting with Hive release 0.11.0.

Many (but not all) `hcat` commands can be issued as `hive` commands, and vice versa. See the HCatalog Command Line Interface document in the HCatalog manual for more information.