Experiment 5: HIVE

Apache Hive Introduction

Apache Hive is an enterprise data warehouse system used to query, manage, and analyse data stored in the Hadoop Distributed File System

The Hive Query Language (HiveQL) facilitates queries in a Hive command-line interface shell. Hadoop can use HiveQL as a bridge to communicate with relational database management systems and perform tasks based on SQL-like commands.

Install Apache Hive on Ubuntu 20.04.

Prerequisites

Apache Hive is based on Hadoop and requires a fully functional Hadoop framework.

5.1 Install Apache Hive on Ubuntu

To configure Apache Hive, first you need to download and unzip Hive. Then you need to customize the following files and settings:

- Edit **.bashrc** file
- Edit **hive-config.sh** file
- Create **Hive directories** in HDFS
- Configure **hive-site.xml** file
- Initiate **Derby database**

Step 1: Download and Untar Hive

Access your Ubuntu command line and download the compressed Hive files using and the wget command followed by the download path:

$ wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz

```
hdoop@phoenixnap:~$ wget https://downloads.apache.org/hive/hive-3.1.2/apache-hi
ve-3.1.2-bin.tar.gz
--2020-06-01 08:11:30--  https://downloads.apache.org/hive/hive-3.1.2/apache-hi
ve-3.1.2-bin.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 2a01:4f8
:10a:201a::2
Connecting to downloads.apache.org (downloads.apache.org)|88.99.95.219|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 278813748 (266M) [application/x-gzip]
Saving to: 'apache-hive-3.1.2-bin.tar.gz'

apache-hive-3.1.2-b 100%[===================>] 265.90M  10.9MB/s    in 25s

2020-06-01 08:11:55 (10.7 MB/s) - 'apache-hive-3.1.2-bin.tar.gz' saved [2788137
48/278813748]
```

Once the download process is complete, untar the compressed Hive package:

Step 2: Configure Hive Environment Variables (bashrc)

The **$HIVE_HOME** environment variable needs to direct the client shell to the *apache-hive-3.1.2-bin* directory. Edit the *.bashrc* shell configuration file using a text editor of your choice (we will be using nano):

$ sudo nano .bashrc

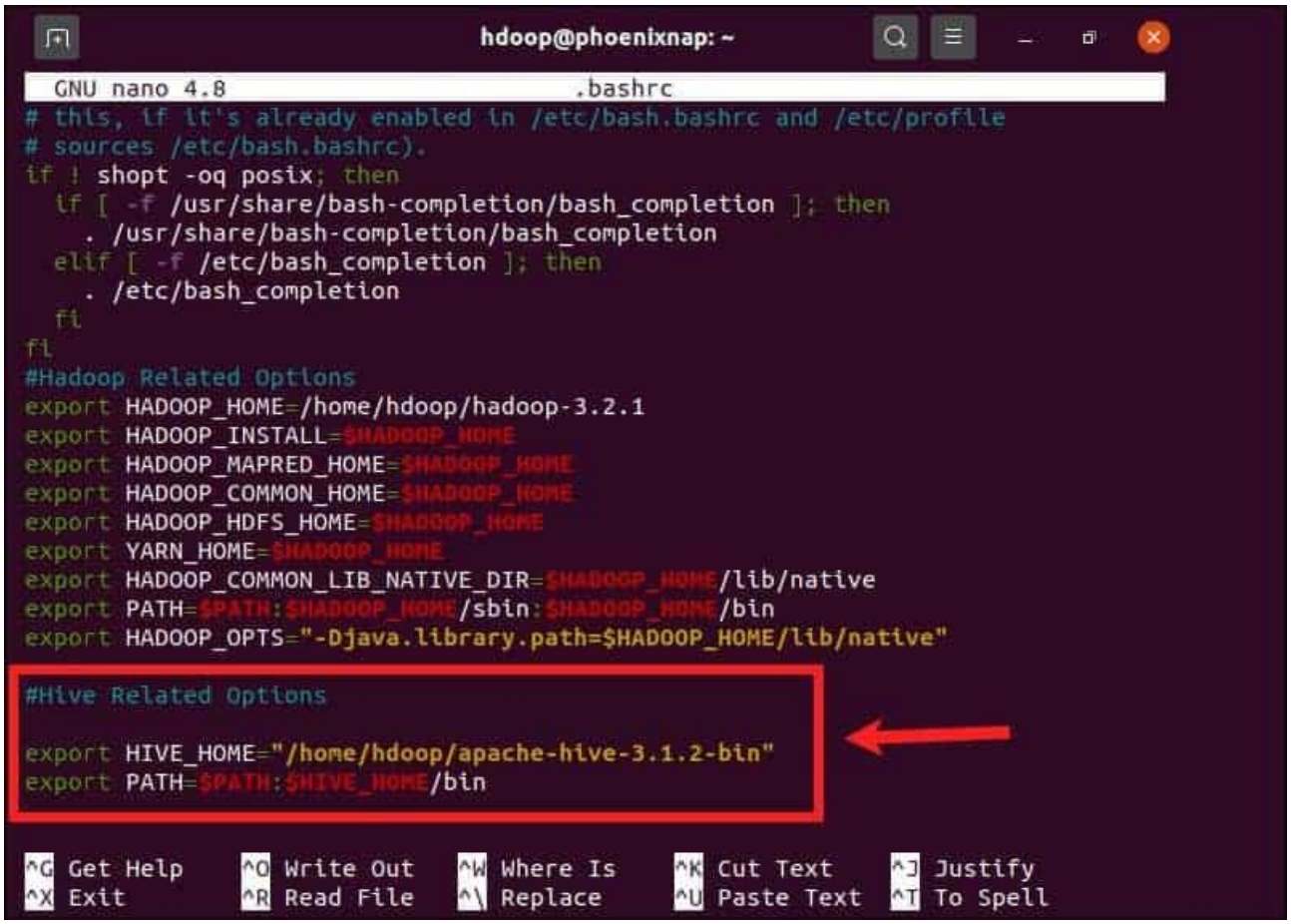
Append the following Hive environment variables to the *.bashrc* file:

export HIVE_HOME= "home/hdoop/apache-hive-3.1.2-bin"

export PATH=$PATH:$HIVE_HOME/bin

The Hadoop environment variables are located within the same file.

$ tar xzf apache-hive-3.1.2-bin.tar.gz



The Hive binary files are now located in the *apache-hive-3.1.2-bin* directory.

Save and exit the *.bashrc* file once you add the Hive variables. Apply the changes to the current environment with the following command:

$ source ~/.bashrc

Step 3: Edit hive-config.sh file

Apache Hive needs to be able to interact with the Hadoop Distributed File System. Access the *hive-config.sh* file using the previously created **$HIVE_HOME** variable:

$ sudo nano $HIVE_HOME/bin/hive-config.sh

Add the **HADOOP_HOME** variable and the full path to your Hadoop directory:

export HADOOP_HOME=/home/hdoop/hadoop-3.2.1

```
# Allow alternate conf dir location.
HIVE_CONF_DIR="${HIVE_CONF_DIR:-$HIVE_HOME/conf}"

export HIVE_CONF_DIR="/home/hdoop/apache-hive-3.1.2-bin/conf"
export HADOOP_HOME=/home/hdoop/hadoop-3.2.1
```

Save the edits and exit the *hive-config.sh* file.

Step 4: Create Hive Directories in HDFS

Create two separate directories to store data in the HDFS layer:

- The temporary, *tmp* directory is going to store the intermediate results of Hive processes.
- The *warehouse* directory is going to store the Hive related tables.

Create tmp Directory

Create a *tmp* directory within the HDFS storage layer. This directory is going to store the intermediary data Hive sends to the HDFS:

$ hdfs dfs -mkdir /tmp

Add write and execute permissions to tmp group members:

$ hdfs dfs -chmod g+w /tmp

Check if the permissions were added correctly:

$ hdfs dfs -ls /

The output confirms that users now have write and execute permissions.

```
hdoop@phoenixnap:~$ hdfs dfs -ls /
Found 4 items
drwxr-xr-x   - hdoop supergroup          0 2020-06-02 02:37 /Exampledir
drwxrwxr-x   - hdoop supergroup          0 2020-06-02 07:26 /tmp
```

Create warehouse Directory

Create the *warehouse* directory within the */user/hive/* parent directory:

$ hdfs dfs -mkdir -p /user/hive/warehouse

Add **write** and **execute** permissions to *warehouse* group members:

$ hdfs dfs -chmod g+w /user/hive/warehouse

Check if the permissions were added correctly:

$ hdfs dfs -ls /user/hive

The output confirms that users now have write and execute permissions.

```
hdoop@phoenixnap:~$ hdfs dfs -ls /user/hive
Found 1 items
drwxrwxr-x   - hdoop supergroup          0 2020-06-02 09:06 /user/hive/warehous
e
```

Step 5: Configure hive-site.xml File (Optional)

Apache Hive distributions contain template configuration files by default. The template files are located within the Hive *conf* directory and outline default Hive settings.

Use the following command to locate the correct file:

$ cd $HIVE_HOME/conf

List the files contained in the folder using the **ls** command.

```
hdoop@phoenixnap:~$ cd $HIVE_HOME/conf
hdoop@phoenixnap:~/apache-hive-3.1.2-bin/conf$ ls
beeline-log4j2.properties.template
hive-default.xml.template          ivysettings.xml
hive-env.sh.template               llap-cli-log4j2.properties.template
hive-exec-log4j2.properties.template  llap-daemon-log4j2.properties.template
hive-log4j2.properties.template    parquet-logging.properties
```

Use the *hive-default.xml.template* to create the *hive-site.xml* file:

$ cp hive-default.xml.template hive-site.xml

Access the *hive-site.xml* file using the nano text editor:

$ sudo nano hive-site.xml

Using Hive in a stand-alone mode rather than in a real-life Apache Hadoop cluster is a safe option for newcomers. You can configure the system to use your local storage rather than the HDFS layer by setting the *hive.metastore.warehouse.dir* parameter value to the location of your Hive *warehouse* directory.

```
GNU nano 4.8                    hive-site.xml                    Modified
  <property>
    <name>hive.metastore.db.type</name>
    <value>DERBY</value>
    <description>
      Expects one of [derby, oracle, mysql, mssql, postgres].
      Type of database used by the metastore. Information schema &amp;amp; JDBCSto>
    </description>
  </property>
  <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive/warehouse</value>
    <description>location of default database for the warehouse</description>
  </property>
  <property>
    <name>hive.metastore.warehouse.external.dir</name>
    <value/>
    <description>Default location for external tables created in the warehouse>
  </property>
  <property>
    <name>hive.metastore.uris</name>
    <value/>
    <description>Thrift URI for the remote metastore. Used by metastore client>
  </property>
  <property>
    <name>hive.metastore.uri.selection</name>
```

Step 6: Initiate Derby Database

Apache Hive uses the Derby database to store metadata. Initiate the Derby database, from the Hive *bin* directory using the **schematool** command:

$ HIVE_HOME/bin/schematool –initSchema –dbType derby

The process can take a few moments to complete.

```
Initialization script completed
schemaTool completed
hdoop@phoenixnap:~/apache-hive-3.1.2-bin/bin$
```

Derby is the default metadata store for Hive. If you plan to use a different database solution, such as MySQL or PostgreSQL, you can specify a database type in the *hive-site.xml* file.

How to Fix guava Incompatibility Error in Hive

If the Derby database does not successfully initiate, you might receive an error with the following content:

"Exception in thread "main" java.lang.NoSuchMethodError: com.google.common.base.Preconditions.checkArgument(ZLjava/lang/String;Ljava/lang/Object;)V"

This error indicates that there is most likely an incompatibility issue between Hadoop and Hive *guava* versions.

Locate the **guava jar** file in the Hive *lib* directory:

$ ls $HIVE_HOME/lib

Locate the **guava jar** file in the Hadoop *lib* directory as well:

$ ls $HADOOP_HOME/share/hadoop/hdfs/lib

The two listed versions are not compatible and are causing the error. Remove the existing **guava** file from the Hive *lib* directory:

$ rm $HIVE_HOME/lib/guava-19.0.jar

Copy the **guava** file from the Hadoop *lib* directory to the Hive *lib* directory:

$ cp $HADOOP_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE_HOME/lib/

Use the **schematool** command once again to initiate the Derby database:

$ HIVE_HOME/bin/schematool –initSchema –dbType derby