Hive 2.3.7 Installation on Google Cloud Platform (GCP) Compute Engine

Step 1: Login as hduser

su hduser

Step 2: Download the Hive tar.gz file (https://hive.apache.org/downloads.html)

sudo wget https://mirrors.estointernet.in/apache/hive/hive-2.3.7/apache-hive-2.3.7-bin.tar.gz

Step 3: Extract the .tar.gz file to /usr/local

sudo tar xvzf apache-hive-2.3.7-bin.tar.gz -C /usr/local

Step 4: Configure environment variables in .bashrc file

sudo vim ~/.bashrc

```
export HIVE_HOME=/usr/local/apache-hive-2.3.7-bin
export HIVE_CONF_DIR=/usr/local/apache-hive-2.3.7-bin/conf
export PATH=$HIVE_HOME/bin:$PATH
export CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/apache-hive-2.3.7-bin/lib/*:.
```

Step 5: Apply the changes to bashrc file.

source ~/.bashrc

Step 6: Creating Hive warehouse directory

Check Hadoop path

echo $HADOOP_INSTALL

check hduser user in supergroup or not

start-all.sh

jps

hdfs dfs -ls /

Step 7: Create Hive directories within HDFS.

The directory **warehouse** is the location to store the table or data related to hive, and the temporary directory **tmp** is the temporary location to store the intermediate result of processing.

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

hdfs dfs -mkdir -p /tmp

Step 8: Set read/write permissions for table.

In this command, we are giving write permission to the group:

hdfs dfs -chmod g+w /tmp

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

check hduser user in supergroup or not

hdfs dfs -ls /

hdfs dfs -ls /user

Configuring Hive

Step 9:

To configure Hive with Hadoop, we need to edit the **hive-env.sh** file, which is placed in the **$HIVE_HOME/conf** directory.

Redirect to conf folder

cd $HIVE_HOME/conf

create hive-env.sh from hive-env.sh. template

sudo cp hive-env.sh.template hive-env.sh

Step 10: Edit the hive-env.sh file by appending the following line

sudo vim hive-env.sh

appending the following line

export HADOOP_INSTALL=/usr/local/hadoop

Step 11: Downloading Apache Derby

Now we need an external database server to configure **Metastore**. We use **Apache Derby** database.

cd /tmp

sudo wget http://archive.apache.org/dist/db/derby/db-derby-10.13.1.1/db-derby-10.13.1.1-bin.tar.gz

sudo tar xvzf db-derby-10.13.1.1-bin.tar.gz -C /usr/local

cd ..

Step 12: set up the Derby environment by appending the following lines to ~/.bashrc file

sudo vim ~/.bashrc

```
export DERBY_HOME=/usr/local/db-derby-10.13.1.1-bin
export PATH=$PATH:$DERBY_HOME/bin
export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
```

Step 12.a: Apply the changes to bashrc file.

source ~/.bashrc

Step 13: To create a directory named data in $DERBY_HOME directory to store Metastore data.

sudo mkdir $DERBY_HOME/data

Configuring Hive Metastore

Configuring **Metastore** - specifying to Hive where the database is stored.

Step 14: Edit the hive-site.xml file, which is in the $HIVE_HOME/conf directory.

cd $HIVE_HOME/conf

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

sudo vim hive-site.xml

In **hive-site.xml** check the following supposed to be there in between the <configuration> and </configuration> tags

```
<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby:;databaseName=metastore_db;create=true</value>
    <description>
      JDBC connect string for a JDBC metastore.
      To use SSL to encrypt/authenticate the connection, provide database-
specific SSL flag in the connection URL.
      For example, jdbc:postgresql://myhost/db?ssl=true for postgres
database.
    </description>
  </property>
```

Step 15: Set permission to Hive folder

cd /usr/local/

sudo chown -R hduser:hadoop apache-hive-2.3.7-bin

Step 16:Metastore schema initialization

From Hive 2.1, we need to run the **schematool** command below as an initialization step.

cd apache-hive-2.3.7-bin/bin

schematool -dbType derby -initSchema

Step 17:Verifying Hive Installation by running Hive CLI

echo $HIVE_HOME

$HIVE_HOME/bin/hive

If you get Error:

cd ..

cd conf

sudo vim **hive-site.xml**

Error:

```
Exception in thread "main" java.lang.RuntimeException: Couldn't create
directory ${system:java.io.tmpdir}/${hive.session.id}_resources
```

edit **hive-site.xml**:

```
<property>
    <name>hive.downloaded.resources.dir</name>
    <!--
    <value>${system:java.io.tmpdir}/${hive.session.id}_resources</value>
    -->
    <value>/home/hduser/hive/tmp/${hive.session.id}_resources</value>
    <description>Temporary local directory for added resources in the remote
file system.</description>
  </property>
```

Error:

```
java.net.URISyntaxException: Relative path in absolute URI:
${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D
<property>
    <name>hive.exec.local.scratchdir</name>
    <!--
    <value>${system:java.io.tmpdir}/${system:user.name}</value>
    -->
    <value>/tmp/mydir</value>
    <description>Local scratch space for Hive jobs</description>
  </property>
```