Big Data Lab

Hive Installation

Hadoop Setup For Hive

- Check your java version, if it is not Java8, then you need to install Java8 and set it as the default version.

```
java -version
```

- Steps to install java 8

```
sudo apt update
sudo apt install openjdk-8-jdk -y
```

- Select java 8 version as the default version from the existing versions.

```
sudo update-alternatives --config java
(Type the selection number corresponding to java 8)
```

- -Create a java.sh file and export the JAVA_HOME and PATH variables

```
vi java8.sh
```

- Type the below under java8.sh file

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-
amd64/
export PATH=$PATH:$JAVA_HOME
```

- Bash the file

```
bash java8.sh
```

- Export java path

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

- Setting up a Non-Root User for Hadoop Environment. Therefore, Install the OpenSSH server and client using the following command:

```
sudo apt install openssh-server openssh-client -y
```

- Generate an SSH key pair and define the location is is to be stored in:

```
ssh-keygen -t rsa -P " -f ~/.ssh/id_rsa
```

- Use the cat command to store the public key as authorized_keys in the ssh directory:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- Set the permissions for your user with the chmod command:

```
chmod 0600 ~/.ssh/authorized_keys
```

- The new user is now able to SSH without needing to enter a password every time. Verify everything is set up correctly by using the hdoop user to SSH to localhost:

```
ssh localhost
```

- Open and Edit the .bashrc shell configuration file

```
sudo vi .bashrc
```

- Once you add the variables, save and exit the .bashrc file. It is vital to apply the changes to the current running environment by using the following command :-

```
#Hadoop Related Options
export HADOOP_HOME=/home/msrit/Downloads/hadoop-
3.2.2 export HADOOP_INSTALL=$HADOOP_HOME export
HADOOP_MAPRED_HOME=$HADOOP_HOME export
HADOOP_COMMON_HOME=$HADOOP_HOME export
HADOOP_HDFS_HOME=$HADOOP_HOME export
YARN_HOME=$HADOOP_HOME export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOM
E/lib/native export
PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/b
in export HADOOP_OPTS="-
Djava.library.path=$HADOOP_HOME/lib/native" export
HIVE_HOME=/home/msrit/Downloads/apache-hive-3.1.2-bin
export PATH=$PATH:$HIVE_HOME/bin
```

```
source ~/.bashrc
```

- Open the hadoop-env.sh file and make few changes:

```
sudo vi $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

- Uncomment the $JAVA_HOME variable (i.e., remove the # sign) and add the full path to the OpenJDK installation on your system.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

- Open the core-site.xml file

```
sudo vi $HADOOP_HOME/etc/hadoop/core-site.xml
```

- add the following configuration in between <configuration> and </configuration> to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting:

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/msrit/Downloads/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

- Use the following command to open the hdfs-site.xml file for editing:

```
sudo vi $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

- Add the following configuration in between <configuration> and </configuration>

```
<property>
<name>dfs.data.dir</name>
<value>/home/msrit/Downloads/dfsdata/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>/home/msrit/Downloads/dfsdata/datanode</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
```

- Use the following command to access the mapred-site.xml file and define MapReduce values:

```
sudo vi $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

- Add the following configuration in between <configuration> and </configuration> to change the default MapReduce framework name value to yarn:

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

- Open the yarn-site.xml file in a text editor:

```
sudo vi $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

- Add the following configuration in between <configuration> and </configuration>

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF
_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_H
OME</value>
</property>
```

- Format the NameNode before starting Hadoop services for the first time

```
hdfs namenode -format
```

- cd to sbin directory of hadoop

```
cd Downloads/hadoop-3.2.2/sbin
```

- execute the following commands to start the NameNode and DataNode:

```
./start-dfs.sh
```

- Once the namenode, datanodes, and secondary namenode are up and running, start the YARN resource and nodemanagers by typing:

```
./start-yarn.sh
```

- Type this simple command to check if all the daemons are active and running as Java processes:

```
jps
```

- cd to the main directory

```
cd ..
cd ..
cd ..
```

Hive Installation

- Open terminal and cd into Downloads folder.

- Download Hive source code tarball using the followingcommand in the terminal.

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

Or download the source code tarball directly from the website.

- Unzip the file and go to home directory.

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

```
cd ..
```

- Set Hive globally by updating the system bash file.

```
gedit ~/.bashrc
```

- Paste these 2 lines at the end of the file

```
export HIVE_HOME=/home/msrit/Downloads/apache-hive-3.1.2-bin

export PATH=$PATH:$HIVE_HOME/bin
```

- Save and close the bashrc file and run the source command toload and save the new variables globally.

```
source ~/.bashrc
```

- Access the *hive-config.sh* file using the previously created **$HIVE_HOME** variable:

```
sudo gedit $HIVE_HOME/bin/hive-config.sh
```

- Add the **HADOOP_HOME** variable and the full path to your Hadoop directory in *hive-config.sh* file. Save the edits and exit the hive-config.sh file.

```
export HADOOP_HOME=/home/msrit/hadoop-3.2.1
```

- 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 /
```

- 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
```

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
```

- 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 gedit hive-site.xml
```

- 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.

Check the below lines are there in *hive-site.xml* file.

```
<property>
<name>hive.metastore.warehouse.dir</name>

 <value>/user/hive/warehouse</value>
<description>location of default database for the warehouse

</description>

</property>
```

- Paste these lines to *hive-site.xml*:

```
<property>
<name>system:java.io.tmpdir</name>
<value>/tmp/hive/java</value>
</property>
<property>
<name>system:user.name</name>
<value>${user.name}</value>
</property>
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:derby:/opt/hive-3.1.2-bin/metastore_db;databaseName=metastore_db;create=true</value>
</property>
```

- Change the lines in *hive-site.xml* to given below lines:

```
<property>
<name>hive.txn.xlock.iow</name>
<value>true</value>
<description>
Ensures commands with OVERWRITE (such as INSERT OVERWRITE) acquire Exclusive locks for transactional tables. This ensures that inserts (w/o overwrite) running concurrently
are not hidden by the INSERT OVERWRITE.
</description>
</property>
```

- 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
```

- 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/
```

- Run the given command:

```
rm -rf metastore_db
```

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

```
$HIVE_HOME/bin/schematool -dbType derby -initSchema
```

Steps To Run Hive

- Start the Hive command-line interface using the following commands:

```
cd $HIVE_HOME/bin
```

```
hive
```