Big Data Lab

Hadoop Installation

1. Hadoop Installation

1.1 From Source :-

- Open terminal and cd into Downloads folder.
- Download Hadoop source code tarball using the following command in the terminal.

```
wget http://apache.mirrors.lucidnetworks.net/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz
```

Or download the source code tarball directly from the website.

- Unzip the file

```
tar -xvf hadoop-3.2.2.tar.gz
```

Note: hadoop_path is the path of where the extracted Hadoop folder is present. It can be found by using the following commands :-

```
cd hadoop-3.2.2
```

```
pwd
```

- Set Hadoop globally by updating the system bash file.

```
gedit ~/.bashrc
```

- Paste these 2 lines at the end of the file

```
export HADOOP_HOME=hadoop_path
```

```
export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.2.2.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-3.2.2.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-3.2.2.jar:$HADOOP_HOME/lib/*"
```

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

```
source ~/.bashrc
```

- Run the following command to set the Hadoop java path

```
echo export JAVA_HOME=/usr/lib/jvm/default-java >>
$HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

- To verify installation run the following command, it will return a long list of commands.

```
$HADOOP_HOME/bin/hadoop
```

1.2. Installation using a shell script (easier method) :-

Running this shell script automates the complete installation process.

- Open the terminal and create a new file named install-hadoop.sh
- Copy paste the below code into the new file and save it.

```
#!/bin/sh

cd ~/Downloads
echo ""
wget "https://dlcdn.apache.org/hadoop/common/hadoop-3.2.2/hadoop-
3.2.2.tar.gz"
echo Downloaded Hadoop-3.2.2 successfully
echo ""
tar -xvf hadoop-3.2.2.tar.gz
echo Unzipped Hadoop-3.2.2 successfully
echo ""
rm hadoop-3.2.2.tar.gz
cd hadoop-3.2.2
HADOOP_HOME=`pwd`
cd ..
echo export HADOOP_HOME=$HADOOP_HOME  >> ~/.bashrc
source ~/.bashrc
echo ""
echo HADOOP_HOME set to $HADOOP_HOME
echo ""
echo ""
echo export JAVA_HOME=/usr/lib/jvm/default-java >>
$HADOOP_HOME/etc/hadoop/hadoop-env.sh
echo Hadoop JAVA_HOME set to /usr/lib/jvm/default-java
echo ""
CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-
core-3.2.2.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-
client-common-3.2.2.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-
3.2.2.jar:$HADOOP_HOME/lib/*"
echo export CLASSPATH=$CLASSPATH >> ~/.bashrc
source ~/.bashrc
echo ""
echo CLASSPATH set to $CLASSPATH
echo ""
echo "Installation completed successfully"
```

After saving the file run the following command.

```
bash install-hadoop.sh
```

2. Steps to run MapReduce programs

- Create a folder for a new program, inside that create 3 files for Mapper, Reducer and Driver classes. Write the relevant business logic in it. In the same folder run the following commands :-

- Compile all java files

```
javac -d . *.java
```

- Set driver class in manifest

```
echo Main-Class: package-name.driver > Manifest.txt
```

- Create an executable jar file

```
jar cfm customname.jar Manifest.txt package-name/*.class
```

- Run the jar file

```
$HADOOP_HOME/bin/hadoop jar customname.jar
inputfile output
```

- View output

```
cat output/*
```

Note: Replace package-name, customname, inputfile with the relevant names and files used in your program.

Big Data Lab

PIG Installation

1. Pig Installation

1.1 From Source :-

- Open terminal and cd into Downloads folder.
- Download Pig source code tarball using the following command in the terminal.

```
wget https://downloads.apache.org/pig/pig-0.17.0/pig-0.17.0.tar.gz
```

Or download the source code tarball directly from the website.

- Unzip the file

```
tar -xvf pig-0.17.0.tar.gz
```

Note: pig_path is the path of where the extracted PIG folder is present.

It can be found by using the following commands.

```
cd pig-0.17.0
```

```
pwd
```

- Set PIG globally by updating the system bash file.

```
gedit ~/.bashrc
```

- Paste the following code at the end of the file

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

```
export PIG_HOME=pig_path

export PATH=$PATH:$PIG_PATH/bin

export PIG_CLASSPATH=$HADOOP_HOME/conf
```

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

```
source ~/.bashrc
```

- To verify installation run the following command, it will return pig version number.

```
pig -version
```

1.2. Installation using a shell script (easier method) :-

Running this shell script automates the complete installation process.

- Open the terminal and create a new file named install-pig.sh
- Copy paste the below code into the new file and save it.

```
#!/bin/sh

cd ~/Downloads
echo ""
wget "https://downloads.apache.org/pig/pig-0.17.0/pig-0.17.0.tar.gz"
echo Downloaded pig-0.17.0 successfully
echo ""
tar -xvf pig-0.17.0.tar.gz
echo Unzipped pig-0.17.0 successfully
echo ""
rm pig-0.17.0.tar.gz
cd pig-0.17.0
PIG_PATH=`pwd`
cd ..
echo export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64 >> ~/.bashrc
source ~/.bashrc
echo ""
echo JAVA_HOME set to /usr/lib/jvm/java-8-openjdk-amd64
echo ""
echo export PIG_HOME=$PIG_PATH >> ~/.bashrc
source ~/.bashrc
echo ""
echo PIG_HOME set to $PIG_PATH
echo ""
echo export PATH=$PATH:$PIG_PATH/bin >> ~/.bashrc
source ~/.bashrc
echo ""
echo PATH set to $PATH:$PIG_PATH/bin
echo ""
echo export PIG_CLASSPATH=$HADOOP_HOME/conf >> ~/.bashrc
source ~/.bashrc
echo ""
echo PIG_CLASSPATH set to $HADOOP_HOME/conf
echo ""
echo "Installation completed successfully"
```

After saving the file run the following command.

```
bash install-pig.sh
```

2. Steps to run Pig programs

- Create a pig script and write the business logic in this file

```
gedit file_name.pig
```

- To run the code

```
pig -x local file_name.pig
```

Big Data Lab

Spark Installation

1. IntelliJ Installation

1.1 From Source :-

- Download IntelliJ **Community Version** from here.
- Unzip the file
- Using the terminal go to the downloaded folder and cd into the bin folder.
- Run the following command to start the IDE

```
./idea.sh
```

1.2. Installation using a shell script :-

Running this shell script automates the complete installation process.

- Open the terminal and create a new file named install-intellij.sh
- Copy paste the below code into the new file and save it.

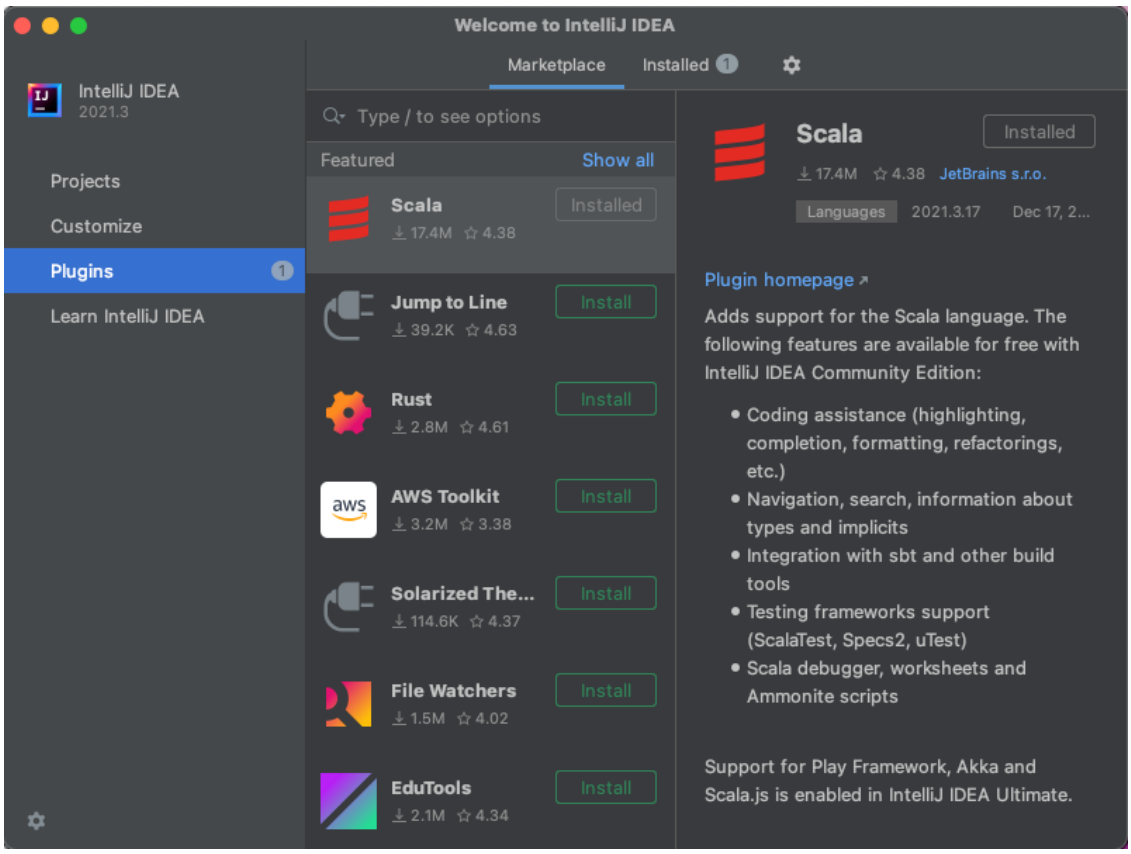
```
#!/bin/sh
cd ~/Downloads
wget "https://download.jetbrains.com/idea/ideaIC-
2021.3.tar.gz?_gl=1*14ggh2a*_ga*MTk0NjM1MjAzNS4xNjQwNTk3NTQz*_ga_V0XZL7QHEB*MTY0MDU
5NzU0My4xLjEuMTY0MDU5NzgxOS4w&_ga=2.226875417.589077562.1640597544-
1946352035.1640597543"
mv "ideaIC-
2021.3.tar.gz?_gl=1*14ggh2a*_ga*MTk0NjM1MjAzNS4xNjQwNTk3NTQz*_ga_V0XZL7QHEB*MTY0MDU
5NzU0My4xLjEuMTY0MDU5NzgxOS4w&_ga=2.226875417.589077562.1640597544-
1946352035.1640597543" "ideaIC-2021.3.tar.gz"
sudo tar -xvzf ideaIC-2021.3.tar.gz
cd ~/Downloads/
cd idea-IC-213.5744.223//bin
./idea.sh
```

After saving the file run the following command.

```
bash install-intellij.sh
```

2. Setup IntelliJ for Spark

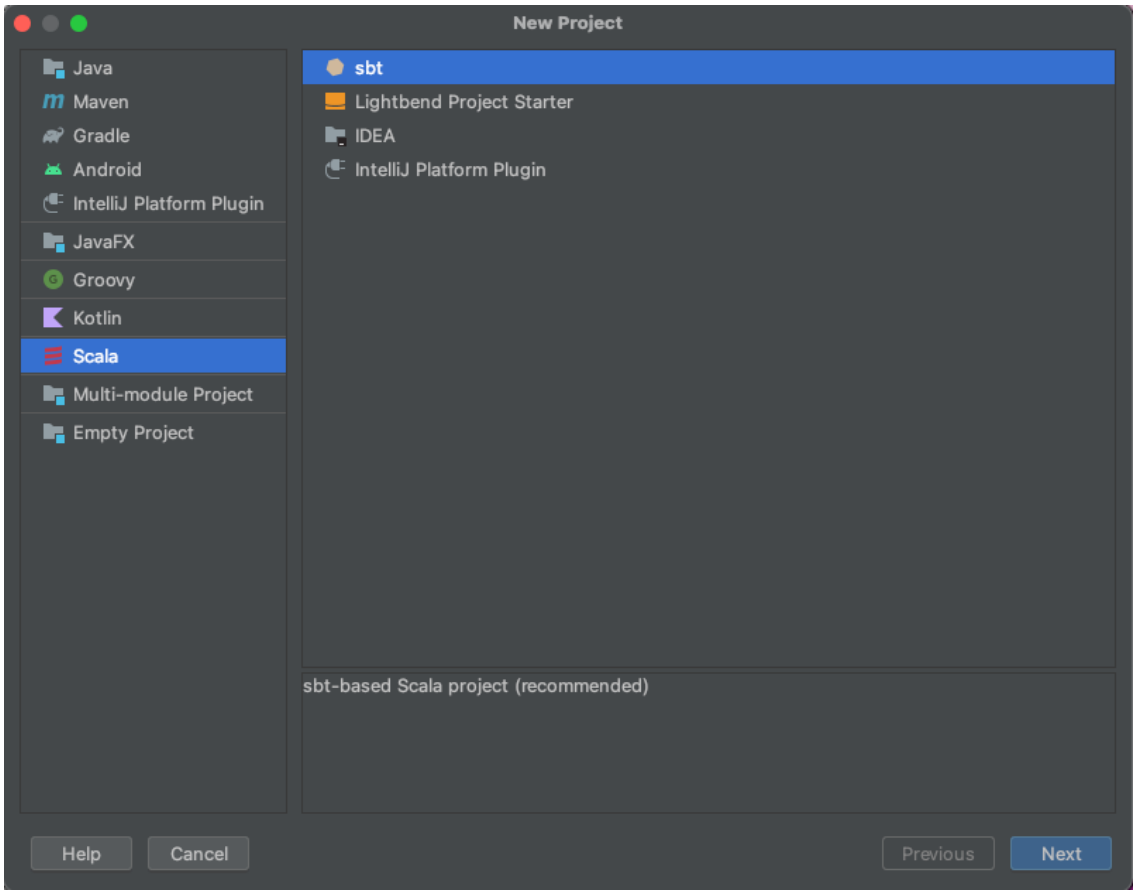
- Navigate to Plugins on the taskbar and install Scala.



- Restart the IDE once the plugin is installed

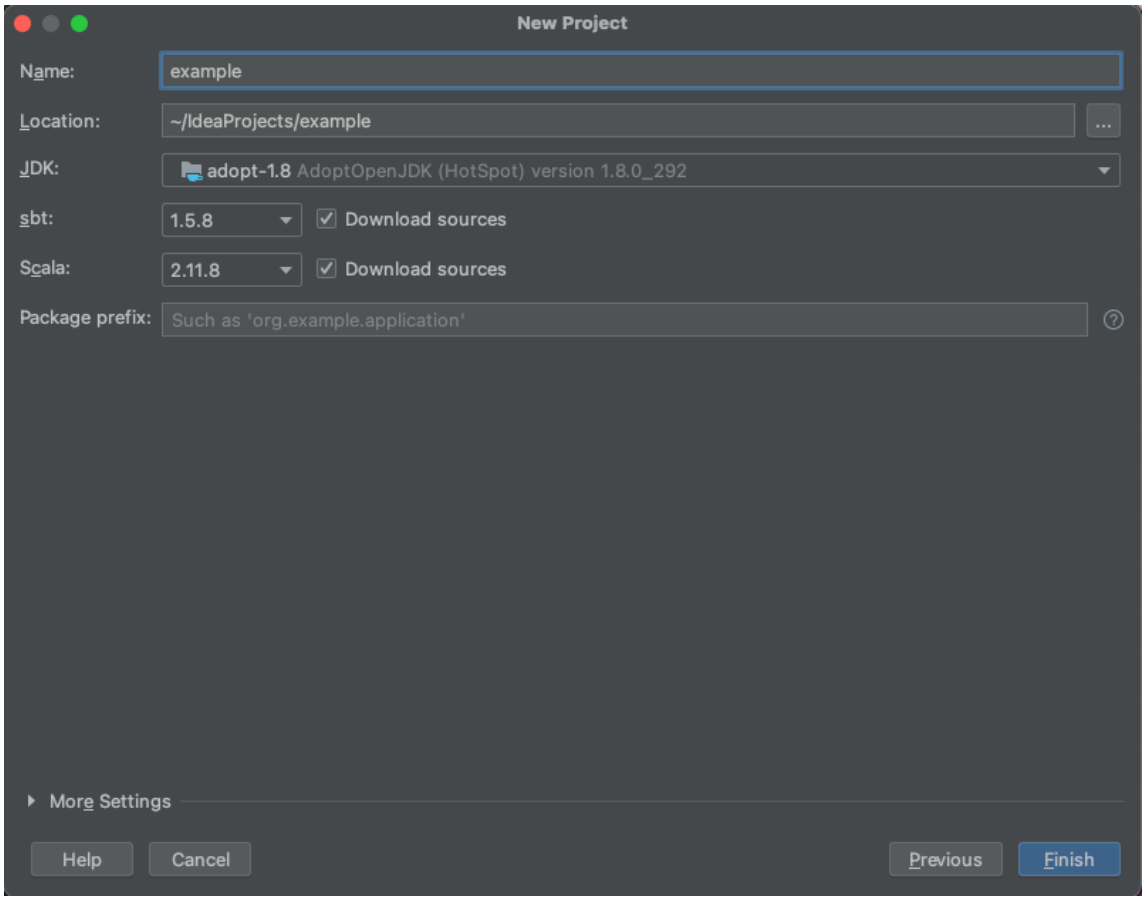
3. Creating Scala Project

- Click on create a project, select scala and use sbt option(default) to create an sbt-based scala project.



- Project settings :-

    - Select JDK : 1.8
    - SBT version : 1.5.8, check download sources
    - Scala version : 2.11.8, check download sources

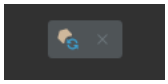


Note: if the versions aren't set properly the programs won't run due to version mismatch errors.

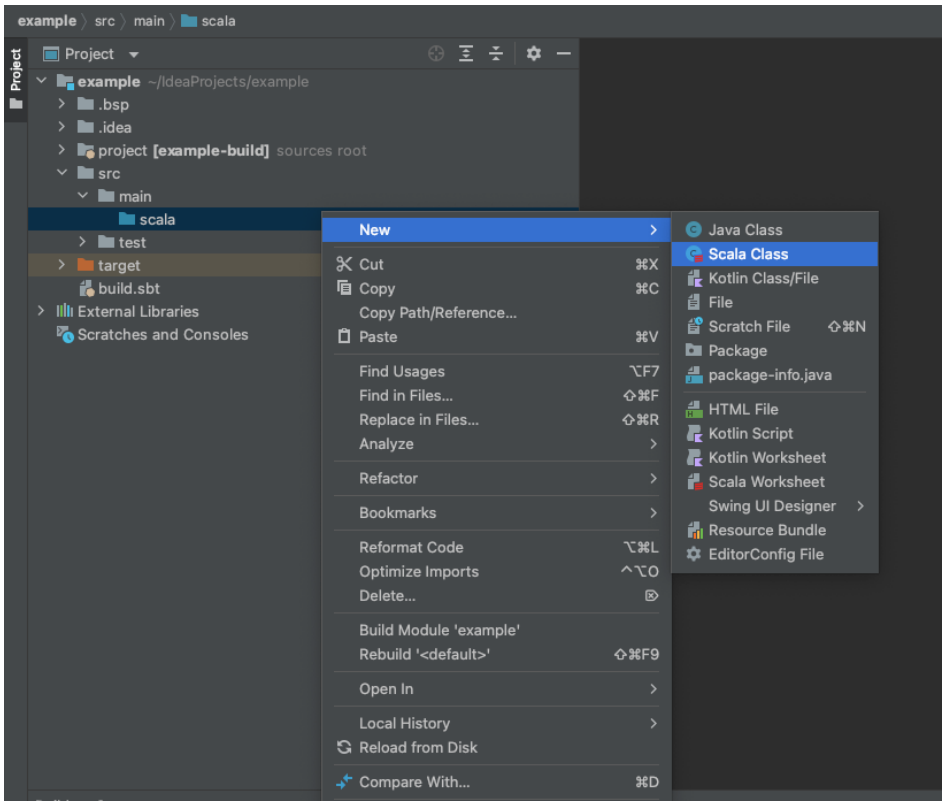
- Open build.sbt file and add the following dependencies

```
val sparkVersion = "2.4.0"

libraryDependencies ++= Seq(
  "org.apache.spark" %% "spark-core" % sparkVersion,
  "org.apache.spark" %% "spark-sql" % sparkVersion,
)
```

-  When this icon appears in your project please click on it to sync your changes made in the build.sbt file.

- Once the build is successful, head over to src/main/scala. Click on *new Scala class*. Give a file name and create a Scala Object .Write the business logic here and run the file.



Note: If you are unable to create a new scala class then the IDE is downloading some important files from the internet, please look at the progress bar found near the bottom right corner of the screen. Once the progress bar finishes all processes and vanishes you will be able to create a new scala class/object.

Big Data Lab

HBase Installation

1. HBase Installation

1. From Source :-

- Open terminal and make sure you are inside the home directory.

- Download HBase source code tarball using the following command in the terminal.

```
wget https://dlcdn.apache.org/hbase/2.4.9/hbase-2.4.9-bin.tar.gz
```

- Unzip the file

```
tar -xvf hbase-2.4.9-bin.tar.gz
```

Note: Make sure Hadoop is installed already.
It can be found by using the following commands.

```
cd ~
```

```
hadoop version
```

Note:

1. Follow the Hadoop installation ppt to set paths in the "bashrc" file.
2. Copy the output of **echo $JAVA_HOME**

- Set HBase paths in bashrc file

```
sudo nano .bashrc
```

- Paste these lines at the end of the file

```
#HBASE_SETUP settings
        #"/home/hdoop" can be different, so,
        #1- cd ~
        #2- pwd
        #3- copy the output and replace the "/home/hdoop" with the output.

        export HBASE_HOME=/home/hdoop/hbase-2.4.9
        export PATH=$PATH:$HBASE_HOME/bin
#JAVA_HOME settings
        # paste the output of "echo $JAVA_HOME"
        export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
        export PATH=$PATH:/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin
```

- Check to confirm all of the below are present

```
#HADOOP_SETUP settings
        #use the output of "echo $HADOOP_HOME"
        export HADOOP_HOME=/home/hdoop/hadoop-3.2.2
        export HADOOP_INSTALL=$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=$HADOOP_HOME/lib/native
        export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
        export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

- Apply the changes in “bashrc” file

```
source ~/.bashrc
```

- create a folder named "zookeeper“ in the home folder.

```
cd ~
mkdir zookeeper
```

- Making changes in the HBase configuration files

```
sudo nano hbase-site.xml
```

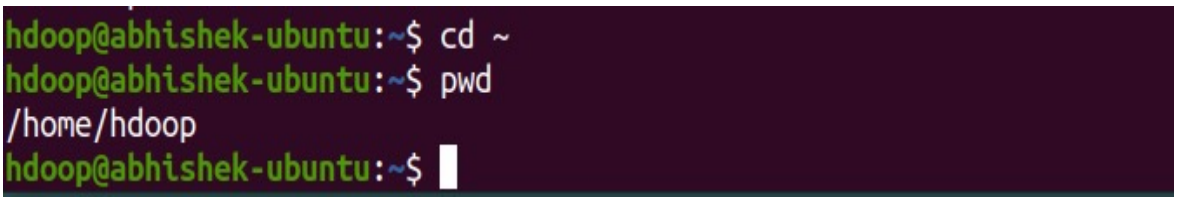
```
cd conf
```

```
cd hbase-2.4.9/
```

- In hbase-site.xml; between <configuration></configuration>, paste the following code and make sure to replace **/home/hdoop** with the output of the path of home directory where **Hbase-2.4.9** is extracted.

```
cd ~
```

```
pwd
```



- Copy the output and replace **/home/hdoop** with the output in the “**hbase-site.xml**”

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
 </property>

 <!--"/home/hdoop" can be different, so,
                1- cd ~
                2- pwd
                3- copy the output and replace the "/home/hdoop" with the output.
  -->
 <property>
    <name>hbase.rootdir</name>
    <value>/home/hdoop/hbase-2.4.9/hbasestorage</value>
 </property>

 <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/hdoop/zookeeper</value>
 </property>

 <property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2181</value>
 </property>

 <property>
  <name>hbase.unsafe.stream.capability.enforce</name>
  <value>false</value>
 </property>
```

- Now, we need to make changes to "**hbase-env.sh**" file

```
sudo nano hbase-env.sh
```

- Paste the following at the end of the file

```
# paste the output of "echo $JAVA_HOME" and $JAVA_PATH
                export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
                export PATH=$PATH:/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin
```

- Uncomment the following code (line)

```
export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"
```

- Start the services

```
sudo hdfs namenode -format
start-dfs.sh
start-yearn.sh
```

- Check the running services

```
jps
```

```
hdoop@abhishek-ubuntu:~$ jps
7137 HQuorumPeer
9617 JarBootstrapMain
13730 SecondaryNameNode
7250 HMaster
13571 DataNode
7894 HRegionServer
13463 NameNode
14247 Jps
```

- Start HBase

```
cd ~/hbase-2.4.9/bin
```

```
./start-hbase.sh
```

```
hdoop@abhishek-ubuntu:~/hbase-2.4.9/bin$ ./start-hbase.sh
127.0.0.1: running zookeeper, logging to /home/hdoop/hbase-2.4.9/bin/../logs/
hbase-hdoop-zookeeper-abhishek-ubuntu.out
running master, logging to /home/hdoop/hbase-2.4.9/logs/hbase-hdoop-master-ab
hishek-ubuntu.out
: running regionserver, logging to /home/hdoop/hbase-2.4.9/logs/hbase-hdoop-r
egionserver-abhishek-ubuntu.out
```

hbase shell

```
hdoop@abhishek-ubuntu:~/hbase-2.4.9/bin$ hbase shell
2022-01-06 14:43:04,845 WARN  [main] util.NativeCodeLoader: Unable to load native-hado
op library for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.4.9, rc49f7f63fca144765bf7c2da41791769286dfccc, Fri Dec 17 19:02:09 PST 2021
Took 0.0023 seconds
hbase:001:0>
```

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_HOME/lib/native export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin export HADOOP_OPTS="-Djava.library.path=$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
```