**Install Hadoop: Setting up a Single Node Hadoop Cluster**

There are two ways to install Hadoop, i.e. **Single node** and **Multi node**.

**Single node cluster** means only one DataNode running and setting up all the NameNode, DataNode, ResourceManager and NodeManager on a single machine. This is used for studying and testing purposes. For example, let us consider a sample data set inside a healthcare industry. So, for testing whether the Oozie jobs have scheduled all the processes like collecting, aggregating, storing and processing the data in a proper sequence, we use single node cluster. It can easily and efficiently test the sequential workflow in a smaller environment as compared to large environments which contains terabytes of data distributed across hundreds of machines.

While in a **Multi node cluster**, there are more than one DataNode running and each DataNode is running on different machines. The multi node cluster is practically used in organizations for analyzing Big Data. Considering the above example, in real time when we deal with petabytes of data, it needs to be distributed across hundreds of machines to be processed. Thus, here we use multi node cluster.

In this blog, I will show you how to install Hadoop on a single node cluster.

**Prerequisites**

* *VIRTUAL BOX*: it is used for installing the operating system on it.
* *OPERATING SYSTEM*: You can install Hadoop on Linux based operating systems. Ubuntu and CentOS are very commonly used. In this tutorial, we are using CentOS.
* *JAVA*: You need to install the Java 8 package on your system.
* *HADOOP*: You require Hadoop 2.7.3 package.

**Install Hadoop**

**Step 1:**[Click here](https://goo.gl/ipdJJa) to download the Java 8 Package. Save this file in your home directory.

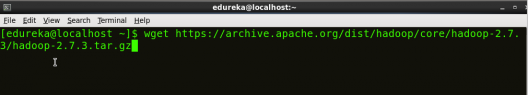
**Step 2:** Extract the Java Tar File.

***Command*:** tar -xvf jdk-8u101-linux-i586.tar.gz

Untar Java - Install Hadoop - Edureka

**Step 3:**Download the Hadoop 2.7.3 Package.

***Command*:** wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz



**Step 4:**Extract the Hadoop tar File.

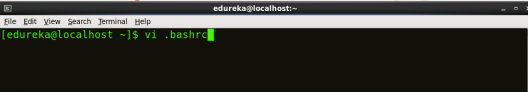
***Command***: tar -xvf hadoop-2.7.3.tar.gz

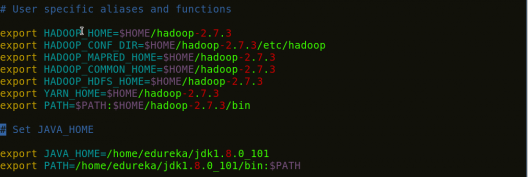
Extract Hadoop Package - Install Hadoop - Edureka

**Step 5:**Add the Hadoop and Java paths in the bash file (.bashrc).

Open**.** **bashrc** file. Now, add Hadoop and Java Path as shown below.

***Command*:**  vi .bashrc





export HADOOP\_HOME=$HOME/hadoop-2.7.3

export HADOOP\_CONF\_DIR=$HOME/hadoop-2.7.3/etc/hadoop

export HADOOP\_MAPRED\_HOME=$HOME/hadoop-2.7.3

export HADOOP\_COMMON\_HOME=$HOME/hadoop-2.7.3

export HADOOP\_HDFS\_HOME=$HOME/hadoop-2.7.3

export YARN\_MAPRED\_HOME=$HOME/hadoop-2.7.3

export PATH=$PATH:$HOME/hadoop-2.7.3/bin

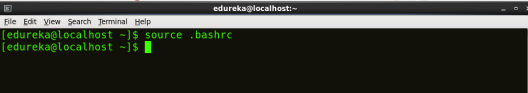
export JAVA\_HOME=/home/a/jdk1.8.0\_121

export PATH=/home/a/jdk1.8.0\_121/bin:$PATH

Then, save the bash file and close it.

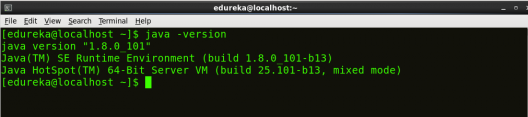
For applying all these changes to the current Terminal, execute the source command.

***Command*:** source .bashrc

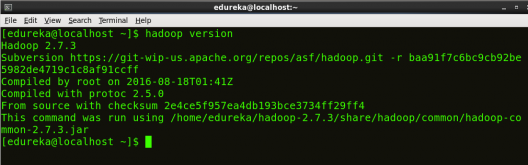


To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.

***Command*:**java -version



***Command*:**hadoop version

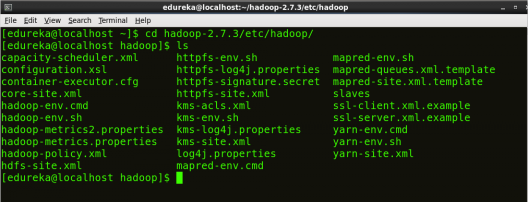


**Step 6:**Edit the [**Hadoop Configuration files**](http://www.edureka.co/blog/explaining-hadoop-configuration/?utm_source=blog&utm_medium=content-link&utm_campaign=install-hadoop).

***Command:*** cd hadoop-2.7.3/etc/hadoop/

***Command:*** ls

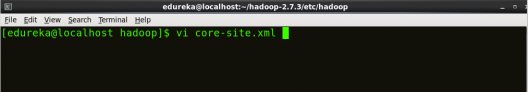
All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can see in the snapshot below:



**Step 7:**Open *core-site.xml* and edit the property mentioned below inside configuration tag:

*core-site.xml* informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.

***Command*:** vi core-site.xml



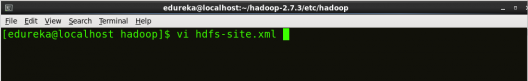


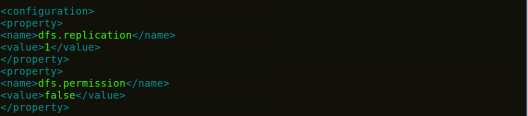
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?xml version="1.0" encoding="UTF-8"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <configuration>  <property>  <name>fs.default.name</name>  <value><hdfs://localhost:9000></value>  </property>  </configuration> |

**Step 8:** Edit *hdfs-site.xml*and edit the property mentioned below inside configuration tag:

*hdfs-site.xml* contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

***Command*:** vi hdfs-site.xml





|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?xml version="1.0" encoding="UTF-8"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <configuration>  <property>  <name>dfs.replication</name>  <value>1</value>  </property>  <property>  <name>dfs.permission</name>  <value>false</value>  </property>  </configuration> |

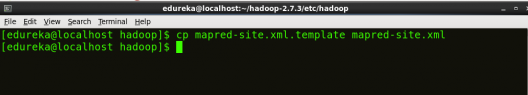
**Step 9:**Edit the *mapred-site.xml* file and edit the property mentioned below inside configuration tag:

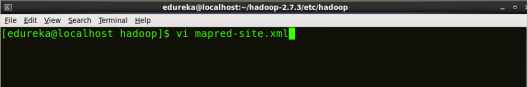
*mapred-site.xml* contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process,  CPU cores available for a process, etc.

In some cases, mapred-site.xml file is not available. So, we have to create the mapred-site.xml file using mapred-site.xml template.

***Command*:** cp mapred-site.xml.template mapred-site.xml

***Command*:**vi mapred-site.xml.





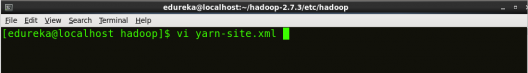


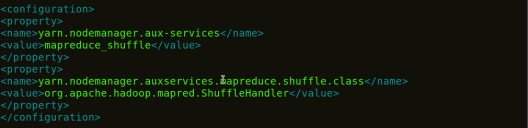
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?xml version="1.0" encoding="UTF-8"?>  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  </configuration> |

**Step 10:** Edit *yarn-site.xml* and edit the property mentioned below inside configuration tag:

*yarn-site.xml* contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

***Command*:** vi yarn-site.xml





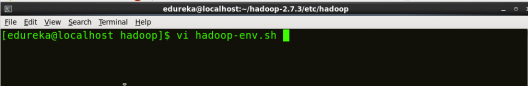
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?xml version="1.0">  <configuration>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>  <value>org.apache.hadoop.mapred.ShuffleHandler</value>  </property>  </configuration> |

**Step 11:**Edit*hadoop-env.sh* and add the Java Path as mentioned below:

*hadoop-env.sh* contains the environment variables that are used in the script to run Hadoop like Java home path, etc.

***Command*:** vi hadoop–env.sh

export JAVA\_HOME=/home/a/jdk1.8.0\_101



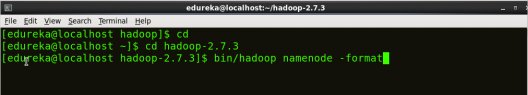
Property of hadoop-env - Install Hadoop - Edureka

**Step 12:** Go to Hadoop home directory and format the NameNode.

***Command*:** cd

***Command*:** cd hadoop-2.7.3

***Command*:** bin/hadoop namenode -format



This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the dfs.name.dir variable.

Never format, up and running Hadoop filesystem. You will lose all your data stored in the HDFS.

**Step 13:** Once the NameNode is formatted, go to hadoop-2.7.3/sbin directory and start all the daemons.

***Command:***cd hadoop-2.7.3/sbin

Either you can start all daemons with a single command or do it individually.

***Command:****./*start-all.sh

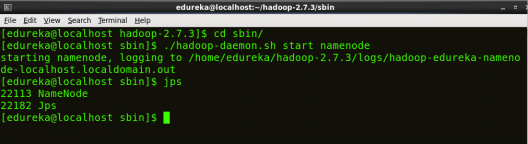
The above command is a combination of***start-dfs.sh, start-yarn.sh*** & ***mr-jobhistory-daemon.sh***

Or you can run all the services individually as below:

**Start NameNode:**

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the file stored across the cluster.

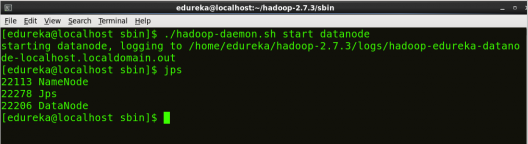
***Command:***./hadoop-daemon.sh start namenode



**Start DataNode:**

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for different operations.

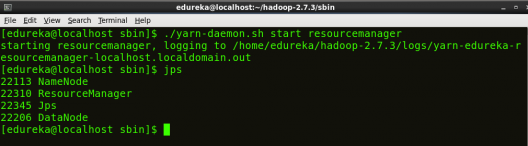
***Command:***./hadoop-daemon.sh start datanode



**Start ResourceManager:**

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and the each application’s ApplicationMaster.

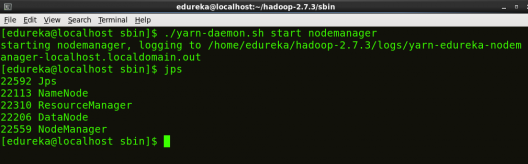
***Command:***./yarn-daemon.sh start resourcemanager



**Start NodeManager:**

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

***Command:***./yarn-daemon.sh start nodemanager



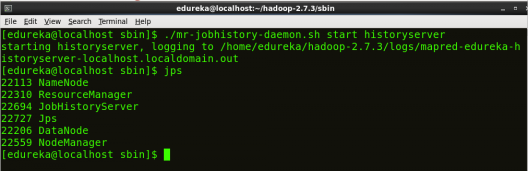
**Start JobHistoryServer:**

JobHistoryServer is responsible for servicing all job history related requests from client.

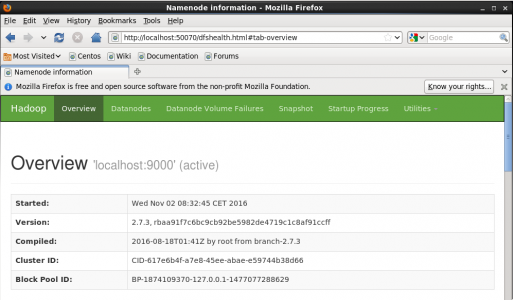
***Command*:** ./mr-jobhistory-daemon.sh start historyserver

**Step 14:** To check that all the Hadoop services are up and running, run the below command.

***Command:***jps



**Step 15:** Now open the Mozilla browser and go to **localhost**:**50070/dfshealth.html** to check the NameNode interface.



Congratulations, you have successfully installed a single node Hadoop cluster in one go.

HIVE

### Downloading Hive

We use hive-0.14.0 in this tutorial. You can download it by visiting the following link [http://apache.petsads.us/hive/hive-0.14.0/.](http://apache.petsads.us/hive/hive-0.14.0/) Let us assume it gets downloaded onto the /Downloads directory. Here, we download Hive archive named “apache-hive-0.14.0-bin.tar.gz” for this tutorial. The following command is used to verify the download:

$ cd Downloads

$ ls

On successful download, you get to see the following response:

apache-hive-0.14.0-bin.tar.gz

## Step 4: Installing Hive

The following steps are required for installing Hive on your system. Let us assume the Hive archive is downloaded onto the /Downloads directory.

### Extracting and verifying Hive Archive

The following command is used to verify the download and extract the hive archive:

$ tar zxvf apache-hive-0.14.0-bin.tar.gz

$ ls

On successful download, you get to see the following response:

apache-hive-0.14.0-bin apache-hive-0.14.0-bin.tar.gz

### Copying files to /usr/local/hive directory

We need to copy the files from the super user “su -”. The following commands are used to copy the files from the extracted directory to the /usr/local/hive” directory.

$ su -

passwd:

# cd /home/user/Download

# mv apache-hive-0.14.0-bin /usr/local/hive

# exit

### Setting up environment for Hive

You can set up the Hive environment by appending the following lines to **~/.bashrc** file:

export HIVE\_HOME=/usr/local/hive

export PATH=$PATH:$HIVE\_HOME/bin

export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/\*:.

export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/\*:.

The following command is used to execute ~/.bashrc file.

$ source ~/.bashrc

## Step 5: Configuring Hive

To configure Hive with Hadoop, you need to edit the **hive-env.sh** file, which is placed in the **$HIVE\_HOME/conf** directory. The following commands redirect to Hive **config** folder and copy the template file:

$ cd $HIVE\_HOME/conf

$ cp hive-env.sh.template hive-env.sh

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

export HADOOP\_HOME=/usr/local/hadoop

Hive installation is completed successfully. Now you require an external database server to configure Metastore. We use Apache Derby database.

## Step 6: Downloading and Installing Apache Derby

Follow the steps given below to download and install Apache Derby:

### Downloading Apache Derby

The following command is used to download Apache Derby. It takes some time to download.

$ cd ~

$ wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz

The following command is used to verify the download:

$ ls

On successful download, you get to see the following response:

db-derby-10.4.2.0-bin.tar.gz

### Extracting and verifying Derby archive

The following commands are used for extracting and verifying the Derby archive:

$ tar zxvf db-derby-10.4.2.0-bin.tar.gz

$ ls

On successful download, you get to see the following response:

db-derby-10.4.2.0-bin db-derby-10.4.2.0-bin.tar.gz

### Copying files to /usr/local/derby directory

We need to copy from the super user “su -”. The following commands are used to copy the files from the extracted directory to the /usr/local/derby directory:

$ su -

passwd:

# cd /home/user

# mv db-derby-10.4.2.0-bin /usr/local/derby

# exit

### Setting up environment for Derby

You can set up the Derby environment by appending the following lines to **~/.bashrc** file:

export DERBY\_HOME=/usr/local/derby

export PATH=$PATH:$DERBY\_HOME/bin

Apache Hive

18

export CLASSPATH=$CLASSPATH:$DERBY\_HOME/lib/derby.jar:$DERBY\_HOME/lib/derbytools.jar

The following command is used to execute **~/.bashrc** file:

$ source ~/.bashrc

### Create a directory to store Metastore

Create a directory named data in $DERBY\_HOME directory to store Metastore data.

$ mkdir $DERBY\_HOME/data

Derby installation and environmental setup is now complete.

## Step 7: Configuring Metastore of Hive

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the hive-site.xml file, which is in the $HIVE\_HOME/conf directory. First of all, copy the template file using the following command:

$ cd $HIVE\_HOME/conf

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

Edit **hive-site.xml** and append the following lines between the <configuration> and </configuration> tags:

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby://localhost:1527/metastore\_db;create=true </value>

<description>JDBC connect string for a JDBC metastore </description>

</property>

Create a file named jpox.properties and add the following lines into it:

javax.jdo.PersistenceManagerFactoryClass =

org.jpox.PersistenceManagerFactoryImpl

org.jpox.autoCreateSchema = false

org.jpox.validateTables = false

org.jpox.validateColumns = false

org.jpox.validateConstraints = false

org.jpox.storeManagerType = rdbms

org.jpox.autoCreateSchema = true

org.jpox.autoStartMechanismMode = checked

org.jpox.transactionIsolation = read\_committed

javax.jdo.option.DetachAllOnCommit = true

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore\_db;create = true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine

## Step 8: Verifying Hive Installation

Before running Hive, you need to create the **/tmp** folder and a separate Hive folder in HDFS. Here, we use the **/user/hive/warehouse** folder. You need to set write permission for these newly created folders as shown below:

chmod g+w

Now set them in HDFS before verifying Hive. Use the following commands:

$ $HADOOP\_HOME/bin/hadoop fs -mkdir /tmp

$ $HADOOP\_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$ $HADOOP\_HOME/bin/hadoop fs -chmod g+w /tmp

$ $HADOOP\_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

The following commands are used to verify Hive installation:

$ cd $HIVE\_HOME

$ bin/hive

On successful installation of Hive, you get to see the following response:

Logging initialized using configuration in jar:file:/home/hadoop/hive-0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties

Hive history file=/tmp/hadoop/hive\_job\_log\_hadoop\_201312121621\_1494929084.txt

………………….

hive>

The following sample command is executed to display all the tables:

hive> show tables;

OK

Time taken: 2.798 seconds

hive

## Step 6: Downloading and Installing Apache Derby

Follow the steps given below to download and install Apache Derby:

### Downloading Apache Derby

The following command is used to download Apache Derby. It takes some time to download.

$ cd ~

$ wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz

The following command is used to verify the download:

$ ls

On successful download, you get to see the following response:

db-derby-10.4.2.0-bin.tar.gz

### Extracting and verifying Derby archive

The following commands are used for extracting and verifying the Derby archive:

$ tar zxvf db-derby-10.4.2.0-bin.tar.gz

$ ls

On successful download, you get to see the following response:

db-derby-10.4.2.0-bin db-derby-10.4.2.0-bin.tar.gz

### Copying files to /usr/local/derby directory

We need to copy from the super user “su -”. The following commands are used to copy the files from the extracted directory to the /usr/local/derby directory:

$ su -

passwd:

# cd /home/user

# mv db-derby-10.4.2.0-bin /usr/local/derby

# exit

### Setting up environment for Derby

You can set up the Derby environment by appending the following lines to **~/.bashrc** file:

export DERBY\_HOME=/usr/local/derby

export PATH=$PATH:$DERBY\_HOME/bin

Apache Hive

18

export CLASSPATH=$CLASSPATH:$DERBY\_HOME/lib/derby.jar:$DERBY\_HOME/lib/derbytools.jar

The following command is used to execute **~/.bashrc** file:

$ source ~/.bashrc

### Create a directory to store Metastore

Create a directory named data in $DERBY\_HOME directory to store Metastore data.

$ mkdir $DERBY\_HOME/data

Derby installation and environmental setup is now complete.

## Step 7: Configuring Metastore of Hive

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the hive-site.xml file, which is in the $HIVE\_HOME/conf directory. First of all, copy the template file using the following command:

$ cd $HIVE\_HOME/conf

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

Edit **hive-site.xml** and append the following lines between the <configuration> and </configuration> tags:

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby://localhost:1527/metastore\_db;create=true </value>

<description>JDBC connect string for a JDBC metastore </description>

</property>

Create a file named jpox.properties and add the following lines into it:

javax.jdo.PersistenceManagerFactoryClass =

org.jpox.PersistenceManagerFactoryImpl

org.jpox.autoCreateSchema = false

org.jpox.validateTables = false

org.jpox.validateColumns = false

org.jpox.validateConstraints = false

org.jpox.storeManagerType = rdbms

org.jpox.autoCreateSchema = true

org.jpox.autoStartMechanismMode = checked

org.jpox.transactionIsolation = read\_committed

javax.jdo.option.DetachAllOnCommit = true

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore\_db;create = true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine

## Step 8: Verifying Hive Installation

Before running Hive, you need to create the **/tmp** folder and a separate Hive folder in HDFS. Here, we use the **/user/hive/warehouse** folder. You need to set write permission for these newly created folders as shown below:

chmod g+w

Now set them in HDFS before verifying Hive. Use the following commands:

$ $HADOOP\_HOME/bin/hadoop fs -mkdir /tmp

$ $HADOOP\_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$ $HADOOP\_HOME/bin/hadoop fs -chmod g+w /tmp

$ $HADOOP\_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

The following commands are used to verify Hive installation:

$ cd $HIVE\_HOME

$ bin/hive

On successful installation of Hive, you get to see the following response:

Logging initialized using configuration in jar:file:/home/hadoop/hive-0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties

Hive history file=/tmp/hadoop/hive\_job\_log\_hadoop\_201312121621\_1494929084.txt

………………….

hive>

The following sample command is executed to display all the tables:

hive> show tables;

OK

Time taken: 2.798 seconds

hive>