**Exp No: 9**

### Aim: Working with Hadoop for single node configuration Hadoop single node Configuration using VM

**Purpose**

This document describes how to set up and configure a single-node Hadoop installation so that you can quicklyperform simple operations using Hadoop MapReduce and the Hadoop Distributed File System (HDFS).

### Prerequisites

***Supported Platforms***

GNU/Linux is supported as a development and production platform. Hadoop has been demonstrated onGNU/Linux clusters with 2000 nodes.

Windows is also a supported platform but the followings steps are for Linux only. To set up Hadoop onWindows, see [wiki page](http://wiki.apache.org/hadoop/Hadoop2OnWindows) .

***Required Software***

Required software for Linux include:

1. Java™ must be installed. Recommended Java versions are described at [HadoopJavaVersions](https://cwiki.apache.org/confluence/display/HADOOP/Hadoop%2BJava%2BVersions) .
2. ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons if the optional start and stop scripts are to be used. Additionally, it is recommmended that pdshalso be installed for better ssh resource management.

***Installing Software***

If your cluster doesn’t have the requisite software you will need install it. For example on Ubuntu Linux:

$ sudo apt-get install ssh

$ sudo apt-get install pdsh

### Download

To get a Hadoop distribution, download a recent stable release from one of the [Apache Download](http://www.apache.org/dyn/closer.cgi/hadoop/common/) [Mirrors](http://www.apache.org/dyn/closer.cgi/hadoop/common/)

### Prepare to Start the Hadoop Cluster

Unpack the downloaded Hadoop distribution. In the distribution, edit the file etc/hadoop/hadoop- env.shto definesome parameters as follows:

# set to the root of your Java installation export JAVA\_HOME=/usr/java/latest

Try the following command:

$ bin/hadoop

This will display the usage documentation for the hadoop script.

Now you are ready to start your Hadoop cluster in one of the three supported modes:

Local (Standalone) Mode Pseudo-Distributed Mode Fully-Distributed Mode

### Standalone Operation

By default, Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful fordebugging.

The following example copies the unpacked conf directory to use as input and then finds and displays every match ofthe given regular expression. Output is written to the given output directory.

$ mkdir input

$ cp etc/hadoop/\*.xml input

$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar grep input output

$ cat output/\*

### Pseudo-Distributed Operation

Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in aseparate Java process.

***Configuration***

Use the following: etc/hadoop/core-site.xml:

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

etc/hadoop/hdfs-site.xml:

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

### Setup passphraseless ssh

Now check that you can ssh to the localhost without a passphrase:

$ ssh localhost

If you cannot ssh to localhost without a passphrase, execute the following commands:

$ ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa

$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

$ chmod 0600 ~/.ssh/authorized\_keys

### Execution

The following instructions are to run a MapReduce job locally. If you want to execute a job on YARN, see YARN onSingle Node.

1. Format the filesystem:

$ bin/hdfs namenode -format

1. Start NameNode daemon and DataNode daemon:

$ sbin/start-dfs.sh

The hadoop daemon log output is written to the $HADOOP\_LOG\_DIRdirectory (defaults to $HADOOP\_HOME/logs).

1. Browse the web interface for the NameNode; by default it is available at: NameNode - http://localhost:9870/
2. Make the HDFS directories required to execute MapReduce jobs:

$ bin/hdfs dfs -mkdir /user

$ bin/hdfs dfs -mkdir /user/<username>

1. Copy the input files into the distributed filesystem:

$ bin/hdfs dfs -mkdir input

$ bin/hdfs dfs -put etc/hadoop/\*.xml input

1. Run some of the examples provided:

$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar grep input output

1. Examine the output files: Copy the output files from the distributed filesystem to the

local filesystem andexamine them:

$ bin/hdfs dfs -get output output

or

View the output files on the distributed filesystem:

$ bin/hdfs dfs -cat output/\*

1. When you’re done, stop the daemons with:

$ sbin/stop-dfs.sh

### YARN on a Single Node

You can run a MapReduce job on YARN in a pseudo-distributed mode by setting a few parameters and runningResourceManager daemon and NodeManager daemon in addition.

The following instructions assume that 1. ~ 4. steps of the above instructions are already executed.

1. Configure parameters as follows: etc/hadoop/mapred-site.xml:

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.application.classpath</name>

<value>$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*:$HADOOP\_MAPRED\_HOME/share/hadoop/map

</property>

</configuration>

etc/hadoop/yarn-site.xml:

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.env-whitelist</name>

<value>JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_D

</property>

</configuration>

1. Start ResourceManager daemon and NodeManager daemon:

$ sbin/start-yarn.sh

1. Browse the web interface for the ResourceManager; by default it is available at:ResourceManager - http://localhost:8088/
2. Run a MapReduce job.
3. When you’re done, stop the daemons with:

$ sbin/stop-yarn.sh

### Fully-Distributed Operation

For information on setting up fully-distributed, non-trivial clusters see [Cluster Setup](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html).

**Exp No: 10**

### Aim: Working with Hadoop for multi node configuration Hadoop multi node Configuration using VM

This experiment explains the setup of the Hadoop Multi-Node cluster on a distributed environment.

As the whole cluster cannot be demonstrated, we are explaining the Hadoop cluster environment using three systems (one master and two slaves); given below are their IP addresses.

* Hadoop Master: 192.168.1.15 (hadoop-master)
* Hadoop Slave: 192.168.1.16 (hadoop-slave-l)
* Hadoop Slave: 192.168.1.17 (hadoop-slave-2)

Follow the steps given below to have Hadoop Multi-Node cluster setup.

###### Installing Java

Java is the main prerequisite for Hadoop. First of all, you should verify the existence of java in your system using "java -version". The syntax of java version command is given below.

$ java -version

If everything works fine it will give you the following output. java version "1.7 .e 71"

Java(TM) SE Runtime Environment (build 1.7.0\_71-b13) Java HotSpot(TM) Client VM (build 25.e-b02, mixed mode)

If java is not installed in your system, then follow the given steps for installing java.

###### Step 1

Download java (JDK <latest version> - X64.tar.gz) by visiting the following link [www.oracle.com](http://www.oracle.com/)

Then jdk-7u71-linux-x64.tar.gz will be downloaded into your system.

###### step 2

Generally you will find the downloaded java file in Downloads folder. Verify it and extract the jdk- 7u71-linux-x64.gz file using the following commands.

$ cd Downloads/

$ ls

jdk-7u71-Linux-x64.gz

$ tar zxf jdk-7u71-Linux-x64.gz

$ ls

jdk1.7.0\_71 jdk-7u71-Linux-x64.gz

###### Step 3

To make java available to all the users, you have to move it to the location "/usr/local/". Open the root, and type the following commands.

$ su password :

# mv jdk1.7.0\_71 /usr/local/ # exit

###### Step 4

For setting up PATH and JAVA\_ HOME variables, add the following commands to /.bashrc file. export JAVA\_HOME=/usr/local/jdk1.7.0\_71

export PATH=PATH:$JAVA\_HOME/bin

Now verify the java -version command from the terminal as explained above. Follow the above process and install java in all your cluster nodes.

###### Creating User Account

Create a system user account on both master and slave systems to use the Hadoop installation. # useradd hadoop

# passwd hadoop

###### Mapping the nodes

You have to edit hosts file in /etc/ folder on all nodes, specify the IP address of each system followed by their host names.

# vi /etc/hosts

enter the following lines in the /etc/hosts file.

192.168. 1.109 hadoop-master

192.168. 1.145 hadoop-slave-l

192.168. 56.1 hadoop-s1ave-2

###### Configuring Key Based Login

Setup ssh in every node such that they can communicate with one another without any prompt for password.

# su hadoop

$ ssh-keygen -t rsa

$ ssh-copy-id -i / .ssh/id\_rsa.pub tutorialspoint@hadoop-master

$ ssh-copy-id -i / .ssh/id\_rsa.pub hadoop\_tp1l@hadoop-slave-l

$ ssh-copy-id -i / .ssh/id\_rsa.pub hadoop\_tp2@hadoop-s1ave-2

$ chmod 0600 / . ssh/authorized\_keys

$ exit

###### Installing Hadoop

In the Master server, download and install Hadoop using the following commands. # mkdir /opt/hadoop

# cd /opt/hadoop/

# wget <http://apache.mesi.com.ar/hadoop/common/hadoop-1.2.1/hadoop-1.2.0.tar.gz> # tar -xzf hadoop-1.2.0.tar .gz

# mv hadoop-1.2.0 hadoop

# chown -R hadoop /opt/hadoop # cd /opt/hadoop/hadoop/

###### Configuring Hadoop

You have to configure Hadoop server by making the following changes as given below.

###### core-site.xml

Open the core-site.xml file and edit it as shown below.

<configuration><property>

< name>fs.default.name</name>

< value>hdfs:/ / hadoop-master : 9000/ </value>

</property>

<property>

< name>dfs.permissions</name>

<value>false</value>

</property>

</configuration>

###### hdfs-site.xml

Open the hdfs-site.xml file and edit it as shown below.

<configuration>

<property>

< name>dfs.data.dir</name>

<value>/opt/hadoop/hadoop/dfs/name/data</value>

<final>true</final>

< /property >

<property>

< name>dfs.name.dir</name>

< value>/opt/hadoop/hadoop/dfs/name</value>

<final>true</final>

< / property >

<property>

< name>dfs . replication</name>

<value>l</value>

</property >

</configuration>

###### mapred-site.xml

Open the mapred-site.xml file and edit it as shown below.

<configuration>

< property >

< name>mapred . job. tracker</name>

<va1ue>hadoop-master:9001</va1ue>

< /property>

</configuration>

###### hadoop-env.sh

Open the hadoop-env.sh file and edit JAVA\_ HOME, HADOOP\_CONF\_DIR and HADOOP\_CONF\_ OPTS as shown below

Note - Set the JAVA HOME as per your system configuration. export JAVA\_HOME=/opt/jdk1.7.0.71

export . net . preferIPv4Stack=true export

###### Installing Hadoop on Slave Servers

**Install Hadoop on all the slave servers by following the given commands.**

# su hadoop

$ cd /opt/hadoop

$ scp -r hadoop hadoop-slave-l:/opt/hadoop

$ scp -r hadoop hadoop-s1ave-2:/opt/hadoop

###### Configuring Hadoop on Master Server

Open the master server and configure it by following the given commands. # su hadoop

$ cd /opt/hadoop/hadoop

###### Configuring Master Node

$ vi etc/hadoop/masters hadoop-master

###### Configuring Slave Node

$ vi etc/hadoop/slaves hadoop-slave-l hadoop-s1ave-2

###### Format Name Node on Hadoop Master

# su hadoop

$ cd /opt/hadoop/hadoop

$ bin/hadoop namenode -format

11/10/1410:58:07 INFO namenode.NameNode: STARTUP\_MSG:

STARTUP\_ MSG: Starting NameNode

STARTUP MSG: host = hadoop-master/192.168.1.109 STARTUP MSG: args = [ -format]

STARTUP MSG: version = 1.2. 0

STARTUP MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/bran compiled by 'hortonfo' on Mon May 6 06:59:37 UTC 2013

STARTUP\_MSG: java = 1.7.0\_ 71

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 11/10/14 10: 58:08 INFO util.GSet: Computing capacity for map BlocksMap editlog=/opt/hadoop/hadoop/dfs/name/current/edits

……………………….

……………………….

11/10/14 10:58:08 INFO common .Storage: Storage directory

/opt/hadoop/hadoop/dfs/name has been successfully formatted. 11/10/14 10:58:08 INFO namenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SHUTDOWN\_MSG: Shutting down NameNode at hadoop-master/192.168.1.15

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

###### Starting Hadoop Services

The following command is to start all the Hadoop services on the Hadoop-Master.

$ cd $HADOOP\_HOME/sbin

$ start-all. sh

###### Adding a New DataNode in the Hadoop Cluster

Given below are the steps to be followed for adding new nodes to a Hadoop cluster.

###### Networking

Add new nodes to an existing Hadoop cluster with some appropriate network configuration Assume the following network configuration.

For New node Configuration - IP address : 192.168.1.103 netmask : 255.255.255.0

hostname : slave3. in

###### Adding User and SSH Access Add a User

On a new node, add "hadoop" user and set password of Hadoop user to anything you want by using the following commands.

useradd hadoop passwd hadoop

Setup Password less connectivity from master to new slave.

###### Execute the following on the master

Mkdir -p $HOME/ .ssh Chmod 700 $HOME/ .ssh

ssh-keygen -t rsa -P ‘’ - f $HOME/ .ssh/id\_rsa

cat $HOME/ .ssh/id\_rsa.pub >> $HOME/ .ssh/authorized\_keys chmod 644 $HOME/ .ssh/authorized\_keys

Copy the public key to new slave node in hadoop user $HOME directory scp $HOME/ .ssh/id\_rsa.pub hadoop@192.168.1.103:/home/hadoop/

###### Execute the following on the slaves

Login to hadoop. If not, login to hadoop user. su hadoop ssh -X [hadoop@192.168.1.103](mailto:hadoop@192.168.1.103)

Copy the content of public key into file **"$HOME/.ssh/authorized\_keys**" and then change the permission for the same by executing the following commands.

cd $HOME

mkdir –p $HOME/ .ssh chmod 700 $HOME/ .ssh

cat id\_rsa.pub >>$HOME/ . ssh/authorized\_keys chmod 644 $HOME/ .ssh/authorized\_keys

Check ssh login from the master machine. Now check if you can ssh to the new node without a password from the master.

ssh [hadoop@192.168.1.103](mailto:hadoop@192.168.1.103) or hadoop@s1ave3

###### Set Hostname of New Node

You can set hostname in file **/etc/sysconfig/network**

On new slave3 machine NETWORKING = yes

HOSTNAME = slave3.in

To make the changes effective, either restart the machine or run hostname command to a new machine with the respective hostname (restart is a good option).

On slave3 node machine - hostname slave3.in

Update /etc/hosts on all machines of the cluster with the following lines - 192.168.1.102 slave3.in slave3

Now try to ping the machine with hostnames to check whether it is resolving to IP or not. On new node machine -

ping master.in

###### Start the DataNode on New Node

Start the datanode daemon manually using $HADOOP HOME/bin/hadoop-daemon.sh script. It will automatically contact the master (NameNode) and join the cluster. We should also add the new node to the conflslaves file in the master server. The script-based commands will recognize the new node.

###### Login to new node

su hadoop or ssh -X [hadoop@192.168.1.103](mailto:hadoop@192.168.1.103)

###### Start HDFS on a newly added slave node by using the following command

./bin/hadoop-daemon.sh start datanode

Check the output of jps command on a new node. It looks as follows.

$ jps

7141 DataNode

10312 Jps

###### Removing a DataNode from the Hadoop Cluster

We can remove a node from a cluster on the fly, while it is running, without any data loss. HDFS provides a decommissioning feature, which ensures that removing a node is performed safely.

To use it, follow the steps as given below

###### Step 1 — Login to master

Login to master machine user where Hadoop is installed.

$ su hadoop

###### Step 2 — Change cluster configuration

An exclude file must be configured before starting the cluster. Add a key named dfs.hosts.exclude to our **$HADOOP HOME/etc/hadoop/hdfs-site.xml** file. The value associated with this key provides the full path to a file on the NameNode's local file system which contains a list of machines which are not permitted to connect to HDFS.

For example, add these lines to **etc/hadoop/hdfs-site.xml file**.

<property>

<name>dfs.hosts.exclude</name>

<value>/home/hadoop/hadoop-l.2.l/hdfs\_exclude. txt</value>

<description>DFS exclude</description>< /property>

###### Step 3 — Determine hosts to decommission

Each machine to be decommissioned should be added to the file identified by the hdfs\_exclude.txt, one domain name per line. This will prevent them from connecting to the NameNode. Content of the **"/home/hadoop/hadoop-l .2.1/hdfs\_exclude.txt"** file is shown below, if you want to remove DataNode2.

slave2. in

###### Step 4 — Force configuration reload

Run the command "$HADOOP HOME/bin/hadoop dfsadmin -refreshNodes quotes.

$ "$HADOOP HOME/bin/hadoop dfsadmin -refreshNodes

This will force the NameNode to re-read its configuration, including the newly updated 'excludes' file. It will decommission the nodes over a period of time, allowing time for each node's blocks to be replicated onto machines which are scheduled to remain active.

On **slave2.in**, check the jps command output. After some time, you will see the DataNode process is shutdown automatically.

###### Step 5 — Shutdown nodes

After the decommission process has been completed, the decommissioned hardware can be safely shut down for maintenance. Run the report command to dfsadmin to check the status of decommission. The following command will describe the status of the decommission node and the connected nodes to the cluster.

$ $HADOOP\_HOME/bin/hadoop dfsadmin -report

###### Step 6 — Edit excludes file again

Once the machines have been decommissioned, they can be removed from the 'excludes' file. Running "$**HADOOP HOME/bin/hadoop dfsadmin -refreshNodes"** again will read the excludes file back into the NameNode; allowing the DataNodes to rejoin the cluster after the maintenance has been completed, or additional capacity is needed in the cluster again, etc.

**Special Note** - If the above process is followed and the tasktracker process is still running on the node, it needs to be shut down. One way is to disconnect the machine as we did in the above steps. The Master will recognize the process automatically and will declare as dead. There is no need to follow the same process for removing the tasktracker because it is NOT much crucial as compared to the DataNode. DataNode contains the data that you want to remove safely without any loss of data.

The tasktracker can be run/shutdown on the fly by the following command at any point of time.

$ $HADOOP\_HOME/bin/hadoop-daemon.sh stop tasktracker

$HADOOP\_HOME/bin/hadoop-daemon.sh start tasktracker

**Exp No: 11**

HBase single node Configuration using VM

###### Apache HBase Installation Modes

Apache HBase can be installed in three modes. The features of these modes are mentioned below.

1. Standalone mode installation (No dependency on Hadoop system)
   * This is default mode of HBase
   * It runs against local file system
   * It doesn't use [Hadoop](https://www.guru99.com/bigdata-tutorials.html) HDFS
   * Only HMaster daemon can run
   * Not recommended for production environment
   * Runs in single JVM
2. Pseudo-Distributed mode installation (Single node Hadoop system + HBase installation)
   * It runs on Hadoop HDFS
   * All Daemons run in single node
   * Recommend for production environment
3. Fully Distributed mode installation (MultinodeHadoop environment + HBase installation)
   * It runs on Hadoop HDFS
   * All daemons going to run across all nodes present in the cluster
   * Highly recommended for production environment

# How to Download Hbase tar file stable version

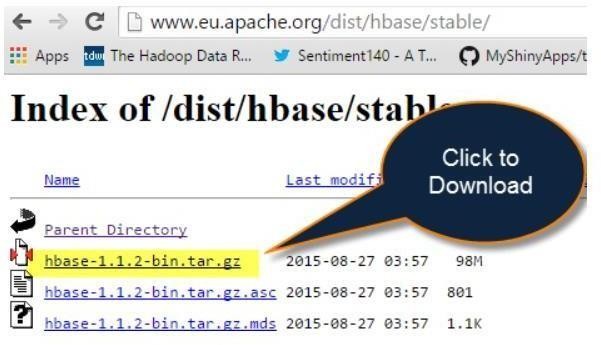
**Step 1)** Go to the link [here](http://www.apache.org/dyn/closer.lua/hbase/) to download HBase. It will open a webpage as shown below.



**Step 2)** Select stable version as shown below 1.1.2 version



**Step 3)** Click on the hbase-1.1.2-bin.tar.gz. It will download tar file. Copy the tar file into an installation location.



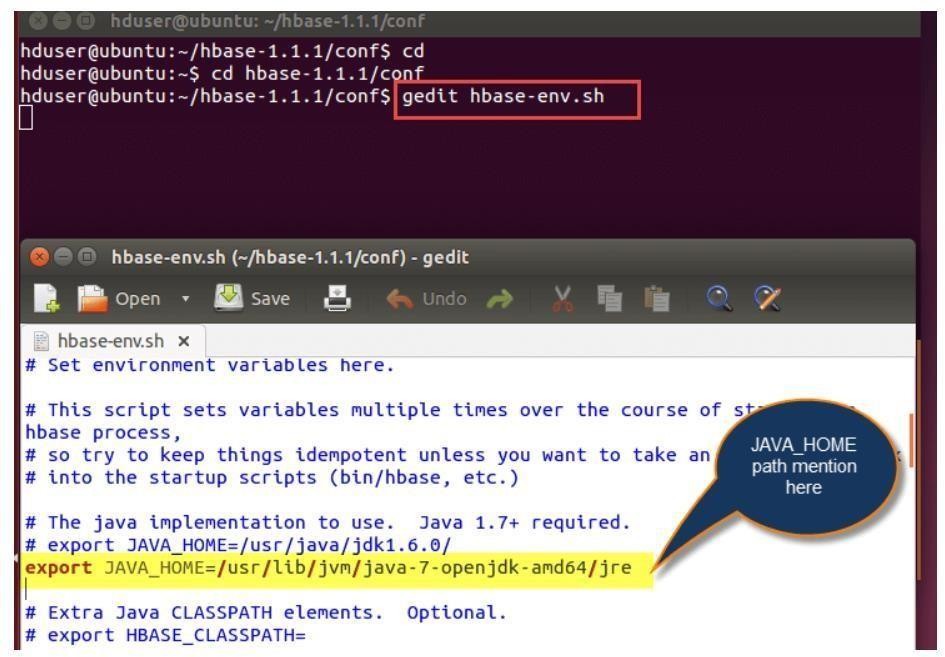
# Hbase - Standalone mode installation:

Installation is performed on Ubuntu with Hadoop already installed.

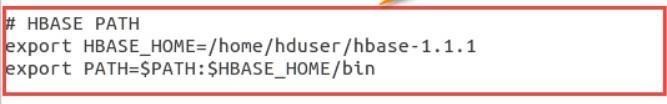
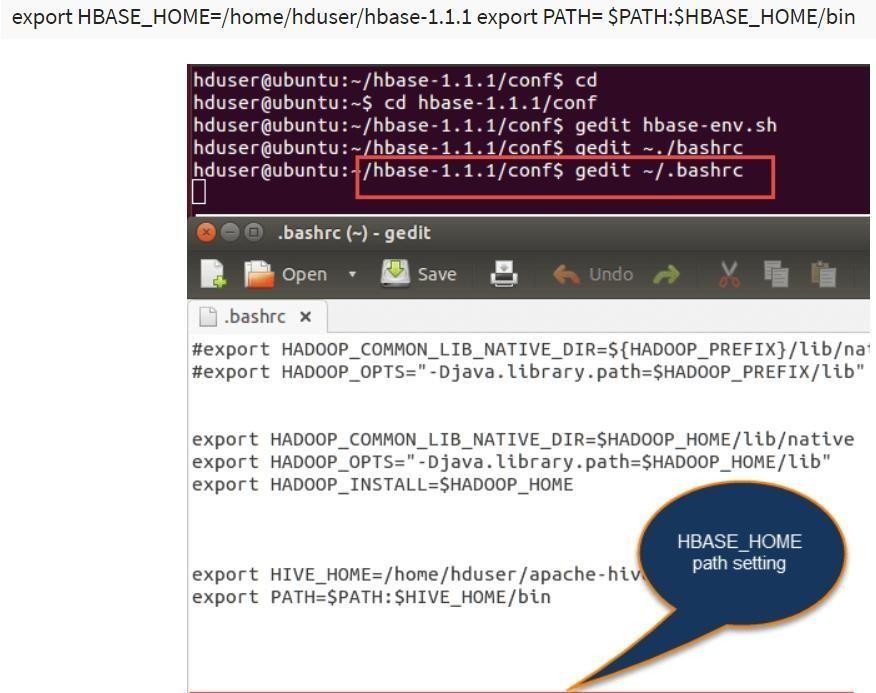
**Step 1)** Place hbase-1.1.2-bin.tar.gz in /home/hduser

**Step 2)** Unzip it by executing command $tar -xvf hbase-1.1.2-bin.tar.gz**.** It will unzip the contents, and it will create hbase-1.1.2 in the location /home/hduser

**Step 3)** Open hbase-env.sh as below and mention JAVA\_HOME path in the location.

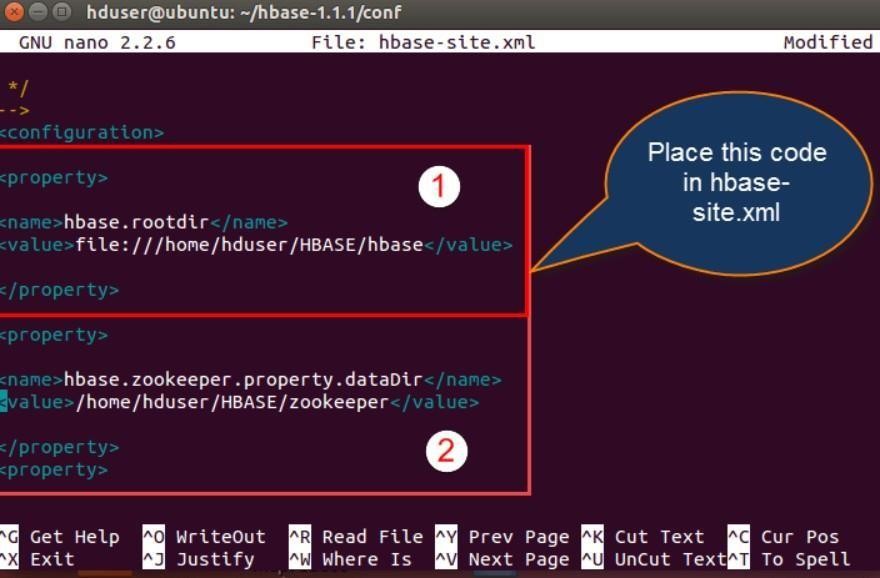


**Step 4)** Open ~/.bashrc file and mention HBASE\_HOME path as shown in below



**Step 5)** Open hbase-site.xml and place the following properties inside the file hduser@ubuntu$ gedit hbase-site.xml(code as below)



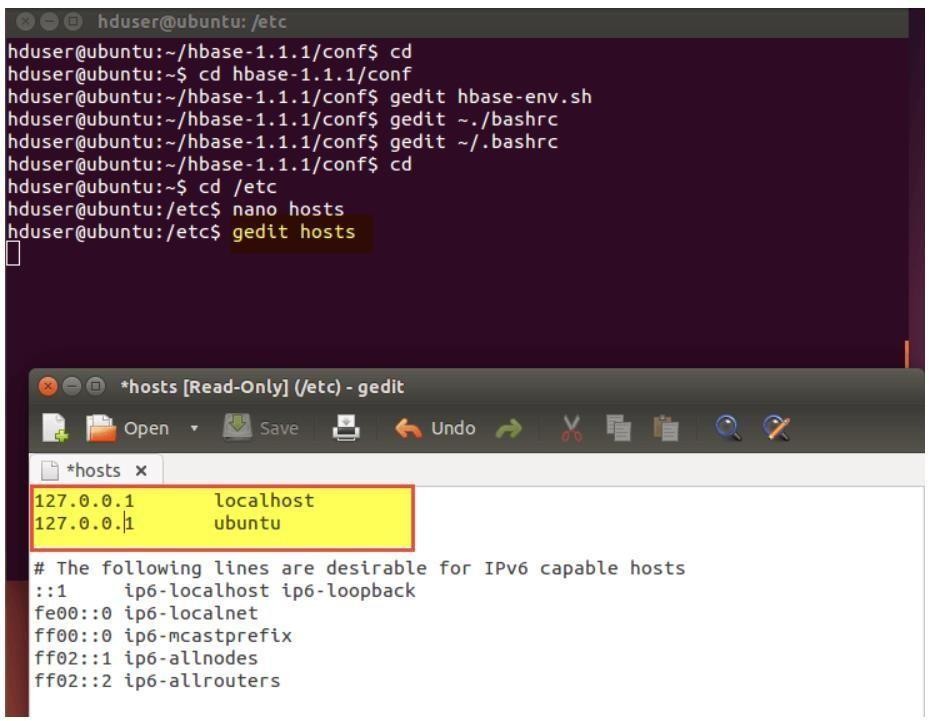


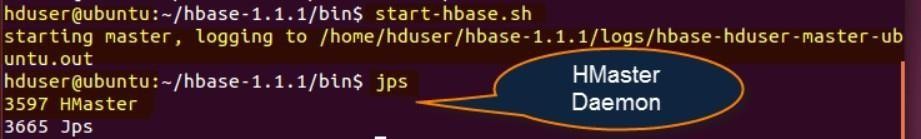
Here we are placing two properties

* + One for HBase root directory and
  + Second one for data directory correspond to ZooKeeper.

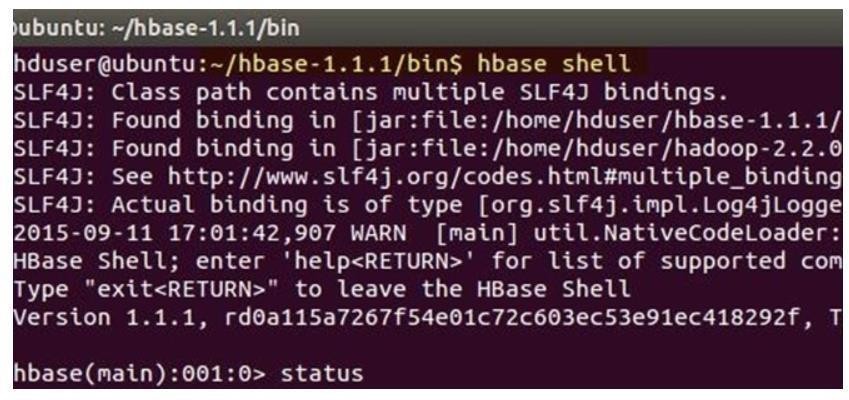
All HMaster and ZooKeeper activities point out to this hbase-site.xml.

**Step 6)** Open hosts file present in /etc. location and mention the IPs as shown in below.



**Step 7)** Now Run Start-hbase.sh in hbase-1.1.1/bin location as shown below. And we can check by jps command to see HMaster is running or not.

**Step8)** HBase shell can start by using "hbase shell" and it will enter into interactive shell mode as shown in below screenshot. Once it enters into shell mode, we can perform all type of commands.



The standalone mode does not require Hadoop daemons to start. HBase can run independently.

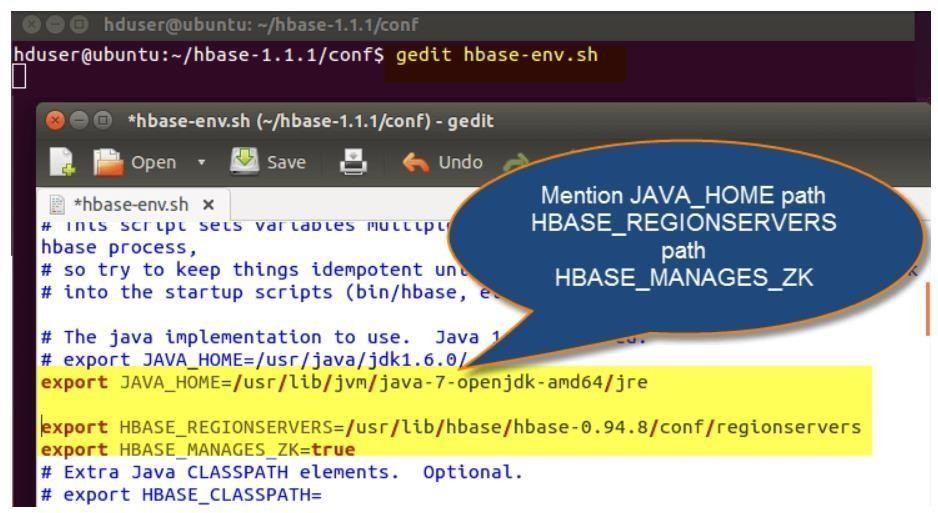
## Hbase - Pseudo Distributed mode of installation:

This is another method for Apache Hbase Installation, known as Pseudo Distributed mode of Installation. Below are the steps to install HBase through this method.

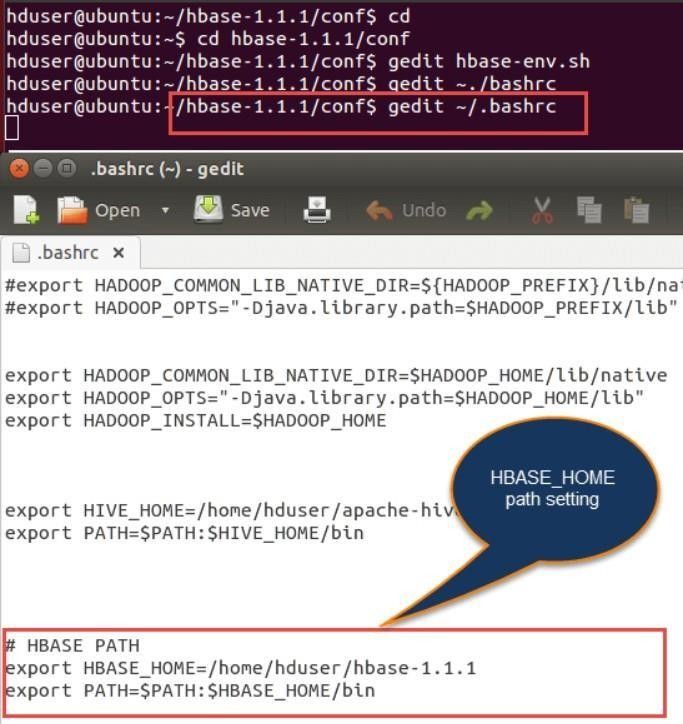
**Step 1)** Place hbase-1.1.2-bin.tar.gz in /home/hduser

**Step 2)** Unzip it by executing command$tar -xvf hbase-1.1.2-bin.tar.gz**.** It will unzip the contents, and it will create hbase-1.1.2 in the location /home/hduser

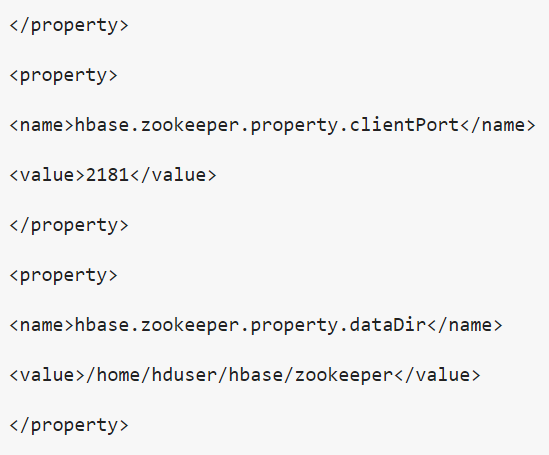
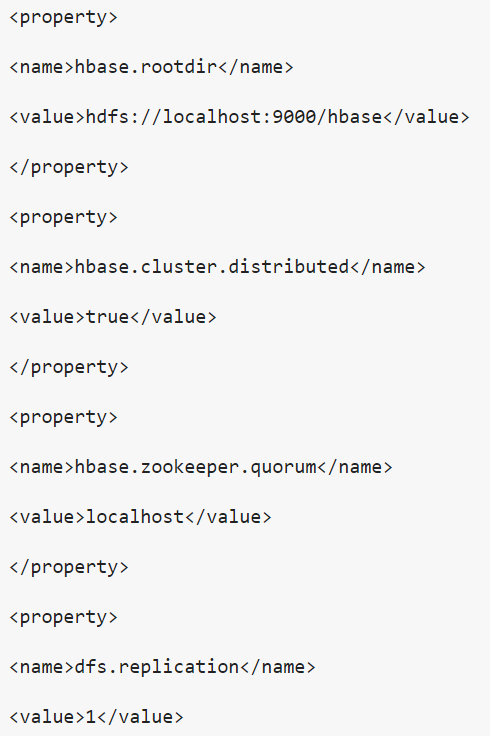
**Step 3)** Open hbase-env.sh as following below and mention JAVA\_HOME path and Region servers' path in the location and export the command as shown

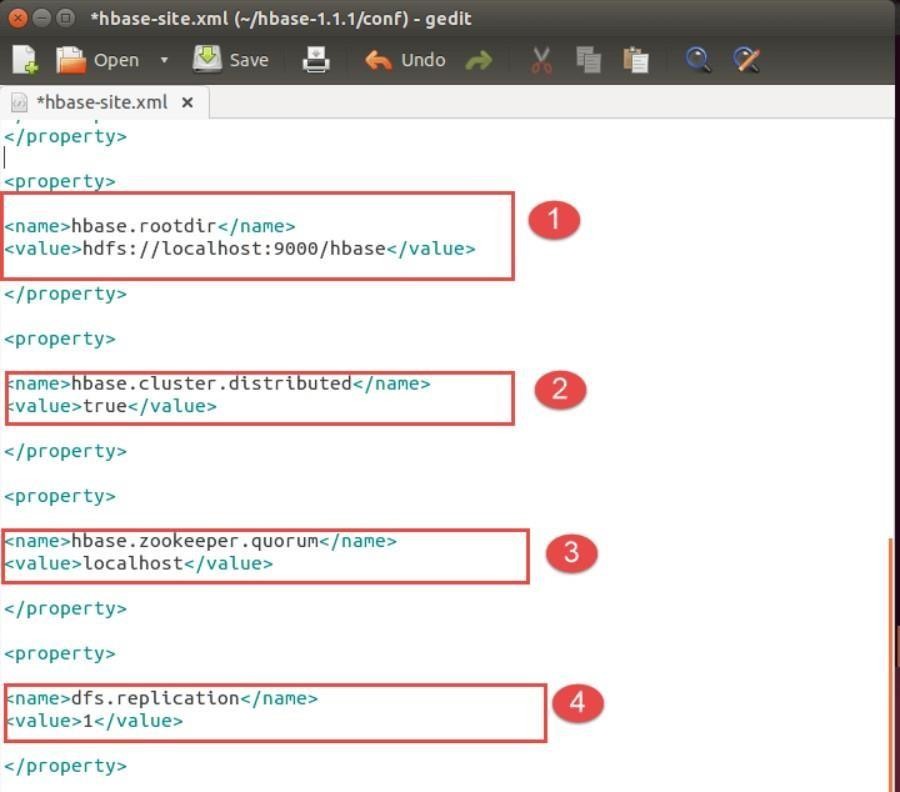


**Step 4)** In this step, we are going to open ~/.bashrc file and mention the HBASE\_HOME path as shown in screen-shot.



**Step 5)** Open HBase-site.xml and mention the below properties in the file.(Code as below)







1. Setting up Hbase root directory in this property
2. For distributed set up we have to set this property
3. ZooKeeper quorum property should be set up here
4. Replication set up done in this property. By default we are placing replication as 1.

In the fully distributed mode, multiple data nodes present so we can increase replication by placing more than 1 value in the dfs.replication property

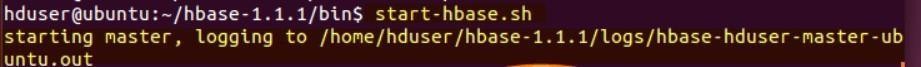
1. Client port should be mentioned in this property
2. ZooKeeper data directory can be mentioned in this property

**Step 6)** Start Hadoop daemons first and after that start HBase daemons as shown below

Here first you have to start Hadoop daemons by using**"./start-all.sh"** command as shown in below.



After starting Hbase daemons by hbase-start.s



Now check jps



# Hbase - Fully Distributed mode installation:-

* + This set up will work in Hadoop cluster mode where multiple nodes spawn across the cluster and running.
  + The installation is same as pseudo distributed mode; the only difference is that it will spawn across multiple nodes.
  + The configurations files mentioned in HBase-site.xml and hbase-env.sh is same as mentioned in pseudo mode.

**HBase Installation Troubleshooting**

**1) Problem Statement:** Master server initializes but region servers not initializes

The Communication between Master and region servers through their IP addresses. Like the way Master is going to listen that region servers are running or having the IP address of 127.0.0.1. The IP address 127.0.0.1 which is the local host and resolves to the master server own local host.

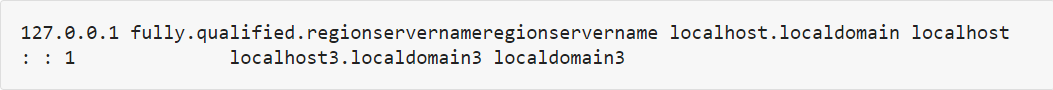
Cause:

In dual communication between region servers and master, region server continuously informs Master server about their IP addresses are 127.0.0.1.

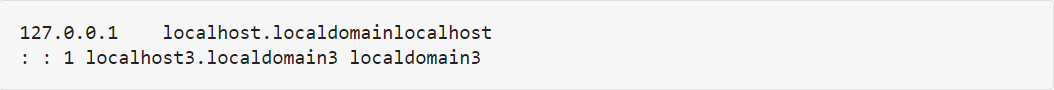
Solution:

* + Have to remove master server name node from local host that is present in hosts file
  + Host file location /etc/hosts What to change:

Open /etc./hosts and go to this location



Modify the above configuration like below (remove region server name as highlighted above)



**2) Problem Statement:** Couldn't find my address: XYZ in list of Zookeeper quorum servers

**Cause:**

* ZooKeeper server was not able to start, and it will throw an error like .xyz in the name of the server.
* HBase attempts to start a ZooKeeper server on some machine but at the same time machine is not able to find itself the quorum configuration i.e. present in **HBase.zookeeper.quorum** configuration file.

**Solution:-**

* Have to replace the host name with a hostname that is presented in the error message
* Suppose we are having DNS server then we can set the below configurations in HBase-site.xml.
  + HBase.zookeeper.dns.interface
  + HBase.zookeeper.dns.nameserver

**3) Problem Statement:** Created Root Directory for HBase through Hadoop DFS

* Master says that you need to run the HBase migrations script.
* Upon running that**, the HBase migrations script respond like no files in root directory**.

**Cause:**

* Creation of new directory for HBase using [Hadoop](https://www.guru99.com/bigdata-tutorials.html) Distributed file system
* Here HBase expects two possibilities

1. Root directory not to exist
2. HBase previous running instance initialized before

**Solution:**

* + Make conformity the HBase root directory does not currently exist or has been initialized by a previous run of HBase instance.
  + As a part of solution, we have to follow steps

**Step 1)** Using Hadoop dfs to delete the HBase root directory

**Step 2)** HBase creates and initializes the directory by itself

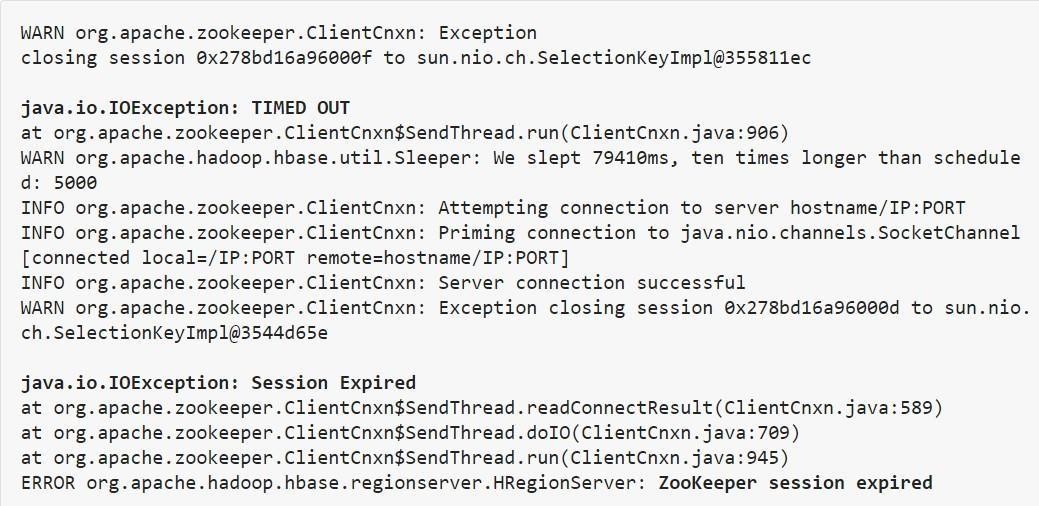
**4) Problem statement:** Zookeeper session expired events

**Cause:**

* + HMaster or HRegion servers shutting down by throwing Exceptions
  + If we observe logs, we can find out the actual exceptions that thrown

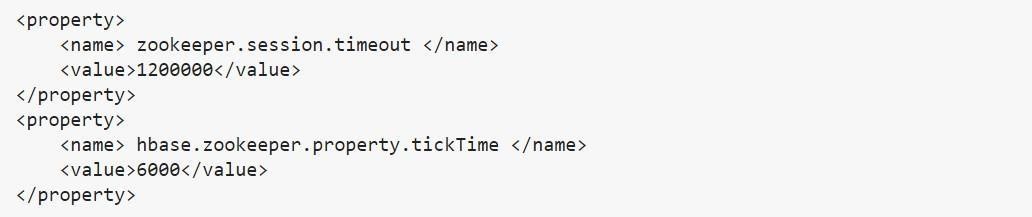
The following shows the exception thrown because of Zookeeper expired event. The highlighted events are some of the exceptions occurred in log file

Log files code as display below:



**Solution:**

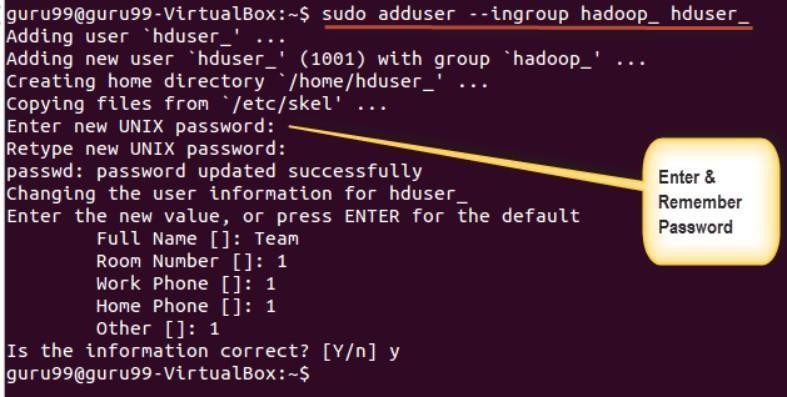
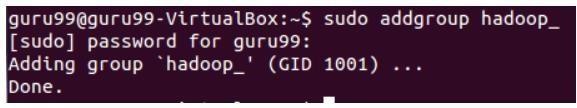
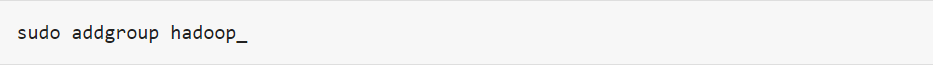
* + The default RAM size is 1 GB. For doing long running imports, we have maintained RAM capacity more than 1 GB.
  + Have to increase the session timeout for Zookeeper.
  + For increasing session time out of Zookeeper, we have to modify the following property in "hbase- site.xml" that present in hbase /conf folder path.
  + The default session timeout is 60 seconds. We can change it to 120 seconds as mentioned below



# How to Install Hadoop with Step by Step Configuration on Linux Ubuntu

## Part 1) Download and Install Hadoop

**Step 1)** Add a Hadoop system user using below command



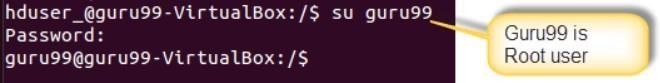
Enter your password, name and other details.

**NOTE:** There is a possibility of below-mentioned error in this setup and installation process.

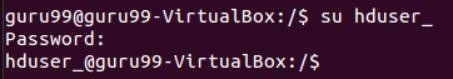
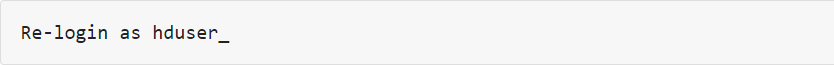
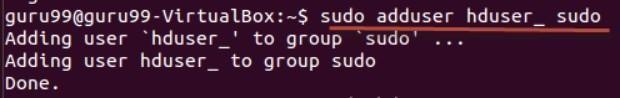
**"hduser is not in the sudoers file. This incident will be reported."**



This error can be resolved by Login as a root user

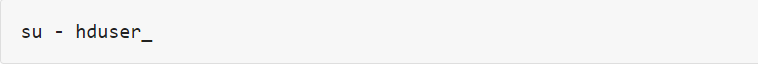


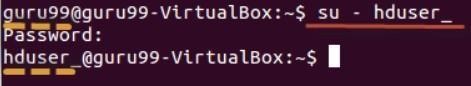
Execute the command



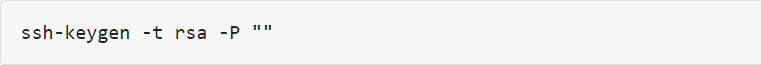
**Step 2)** Configure SSH

In order to manage nodes in a cluster, Hadoop requires SSH access First, switch user, enter the following command



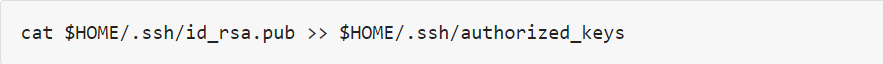


This command will create a new key.

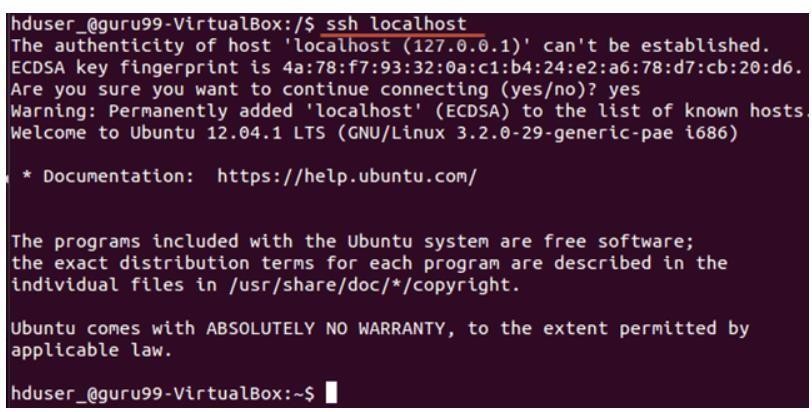




Enable SSH access to local machine using this key.



Now test SSH setup by connecting to localhost as 'hduser' user.



**Note:** Please note, if you see below error in response to 'ssh localhost', then there is a possibility that SSH is not available on this system-



**To resolve this -**

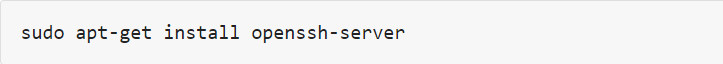
Purge SSH using,



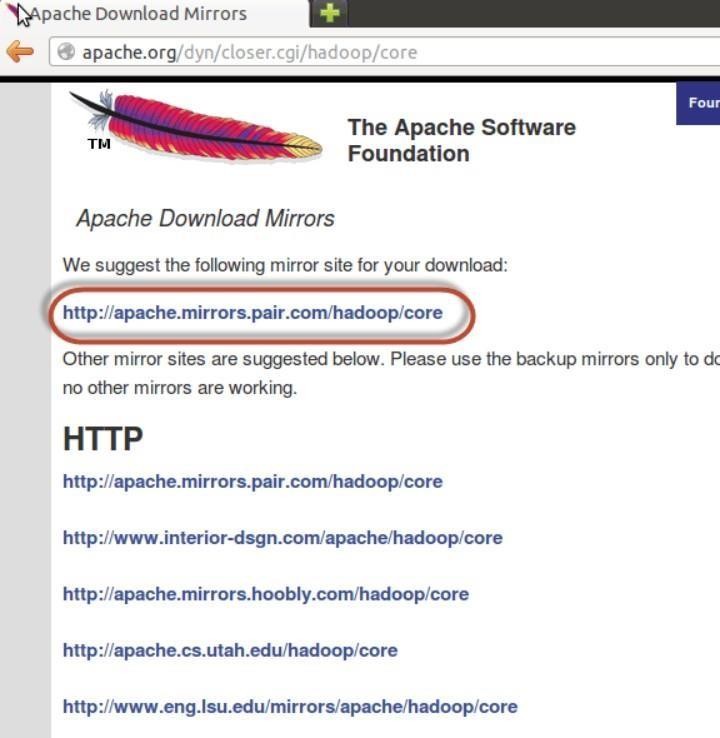
It is good practice to purge before the start of installation



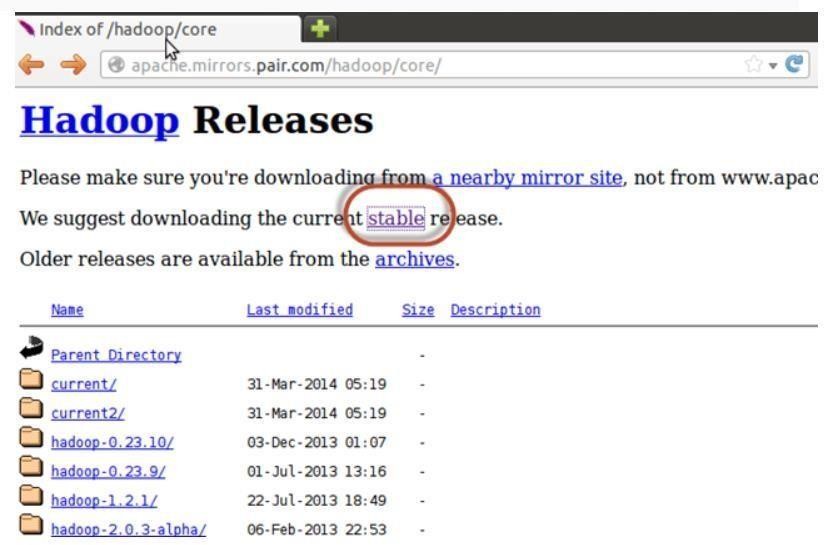
Install SSH using the command-



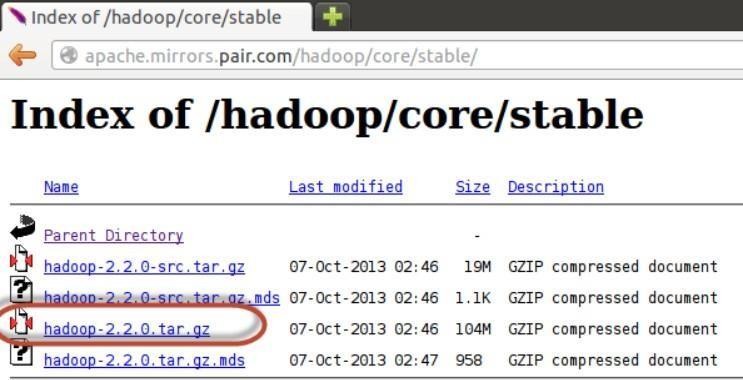
**Step 3)** Next step is to [Download Hadoop](https://www.guru99.com/install-linux.html)



Select Stable



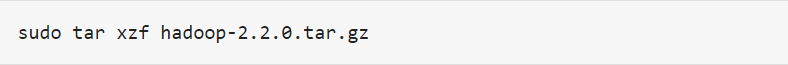
**Select the tar.gz file ( not the file with src)**



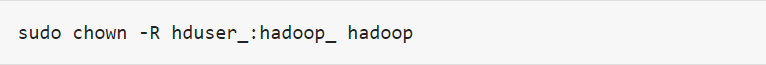
Once a download is complete, navigate to the directory containing the tar file



Enter,



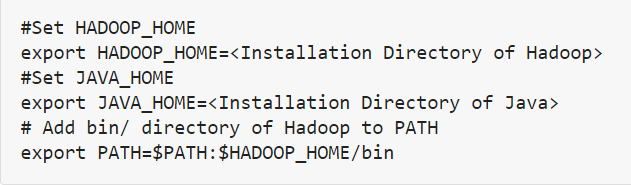
**Now, rename hadoop-2.2.0 as hadoop**

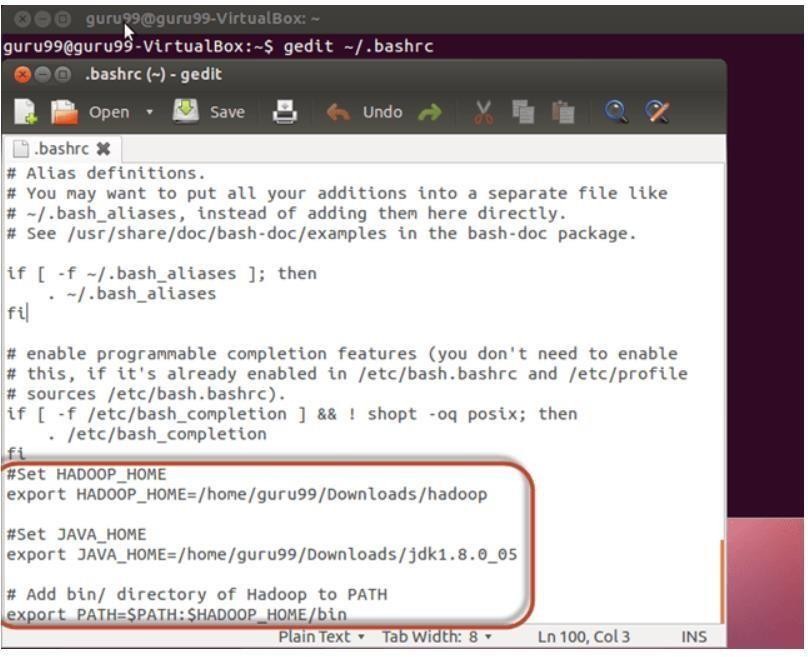


## Part 2) Configure Hadoop

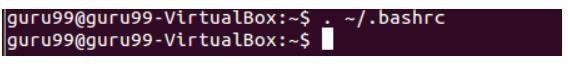
**Step 1)** Modify **~/.bashrc** file

Add following lines to end of file **~/.bashrc**



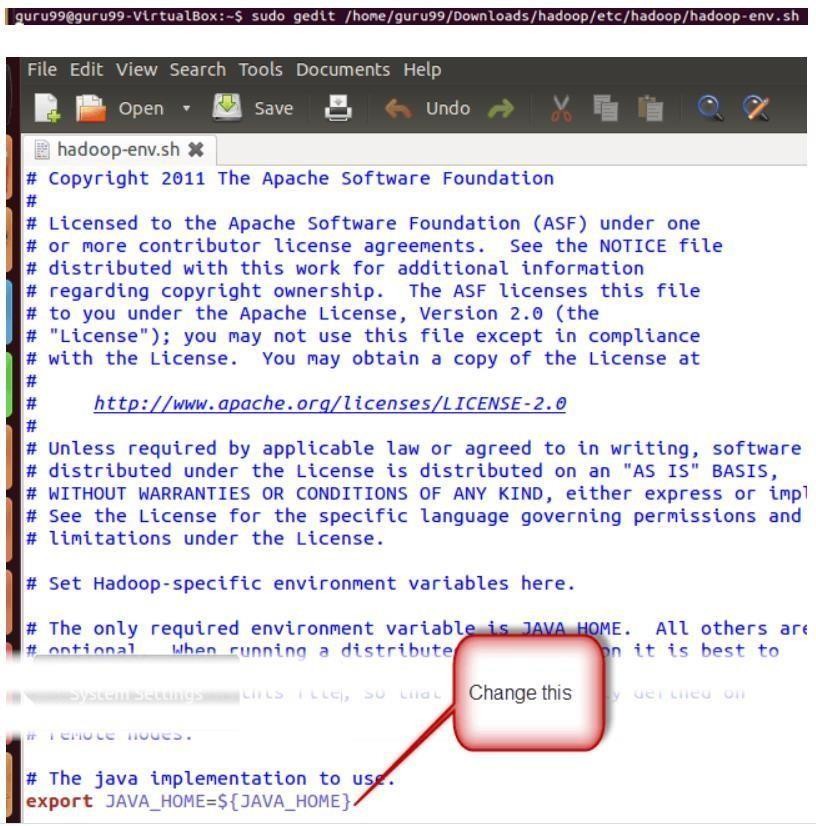


Now, source this environment configuration using below command

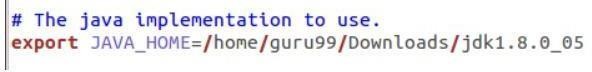


**Step 2)** Configurations related to HDFS

Set **JAVA\_HOME** inside file **$HADOOP\_HOME/etc/hadoop/hadoop-env.sh**



With



There are two parameters in **$HADOOP\_HOME/etc/hadoop/core-site.xml** which need to be set-

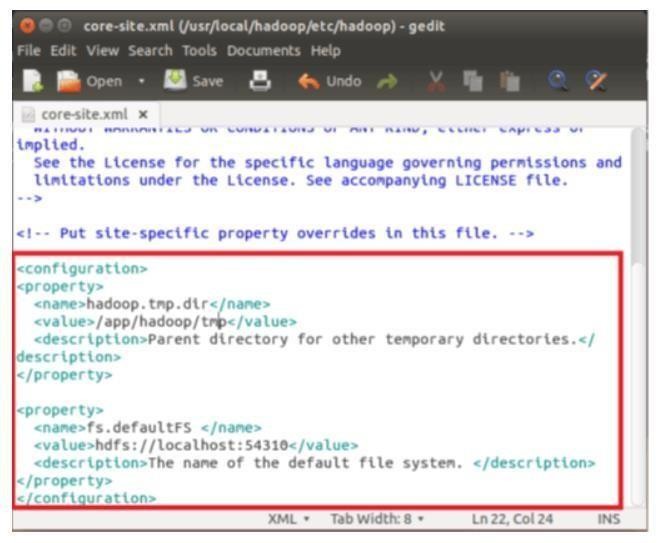
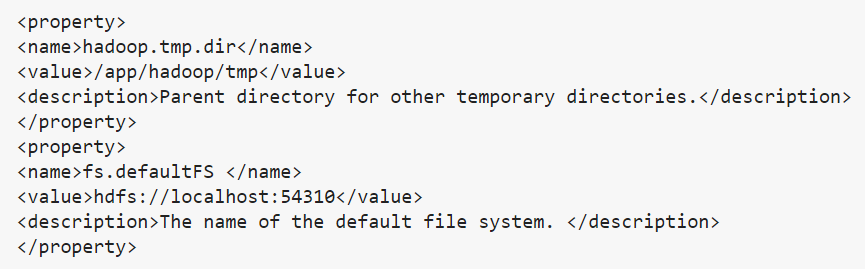
1. **'hadoop.tmp.dir' -** Used to specify a directory which will be used by Hadoop to store its data files.
2. **'fs.default.name' -** This specifies the default file system.

To set these parameters, open core-site.xml





Copy below line in between tags <configuration></configuration>



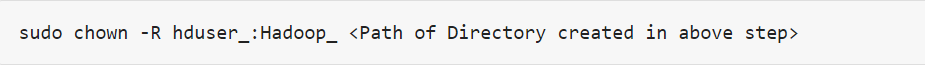
Navigate to the directory **$HADOOP\_HOME/etc/Hadoop**



Now, create the directory mentioned in core-site.xml



Grant permissions to the directory

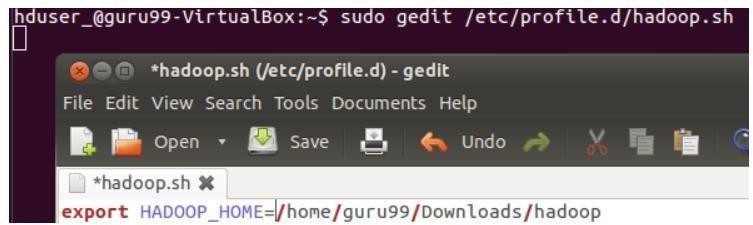


**Step 3)** Map Reduce Configuration

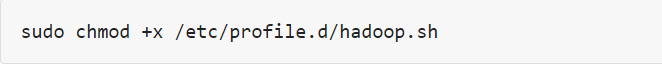
Before you begin with these configurations, lets set HADOOP\_HOME path



And Enter



Next enter



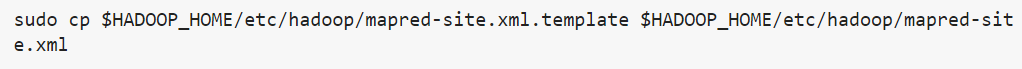


Exit the Terminal and restart again

Type echo $HADOOP\_HOME. To verify the path



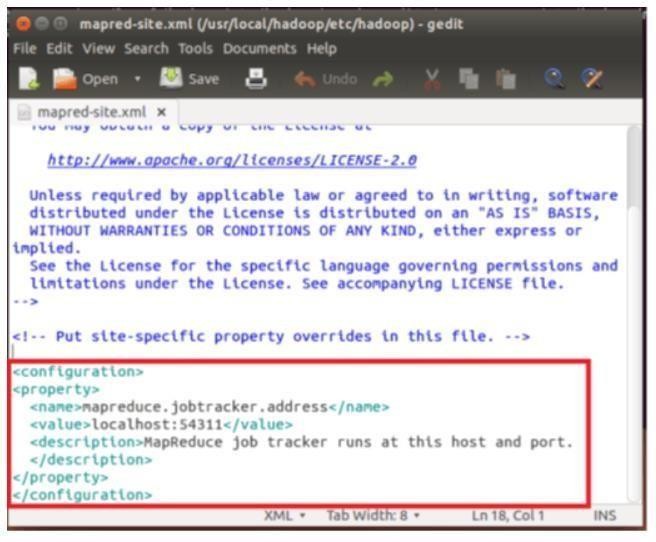
Now copy files



Open the **mapred-site.xml** file

sudo gedit $HADOOP\_HOME/etc/hadoop/mapred-site.xml

Add below lines of setting in between tags <configuration> and </configuration>



<property>

<name>mapreduce.jobtracker.address</name>

<value>localhost:54311</value>

<description>MapReduce job tracker runs at this host and port.

</description>

</property>

Open **$HADOOP\_HOME/etc/hadoop/hdfs-site.xml** as below,

sudo gedit $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

Add below lines of setting between tags <configuration> and </configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.</description>

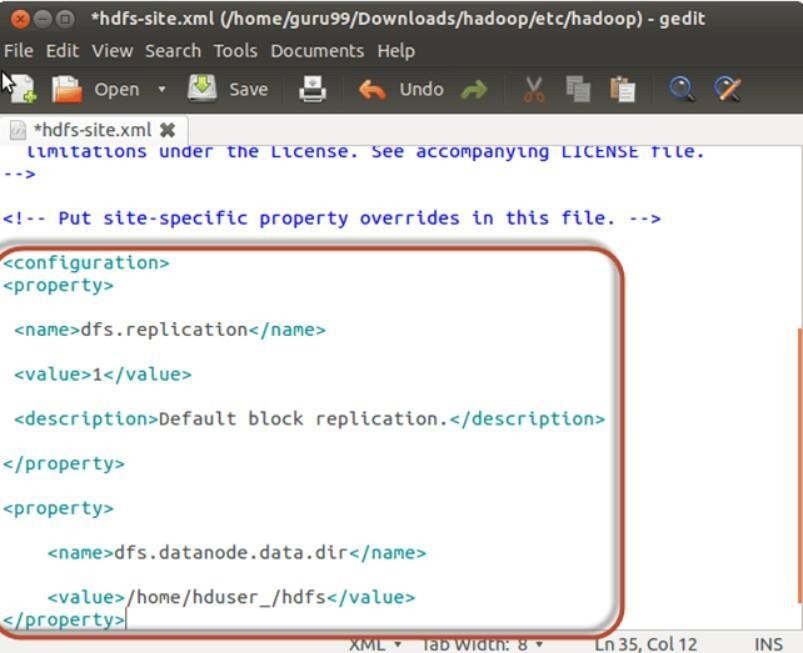
</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/home/hduser\_/hdfs</value>

</property>



Create a directory specified in above setting-

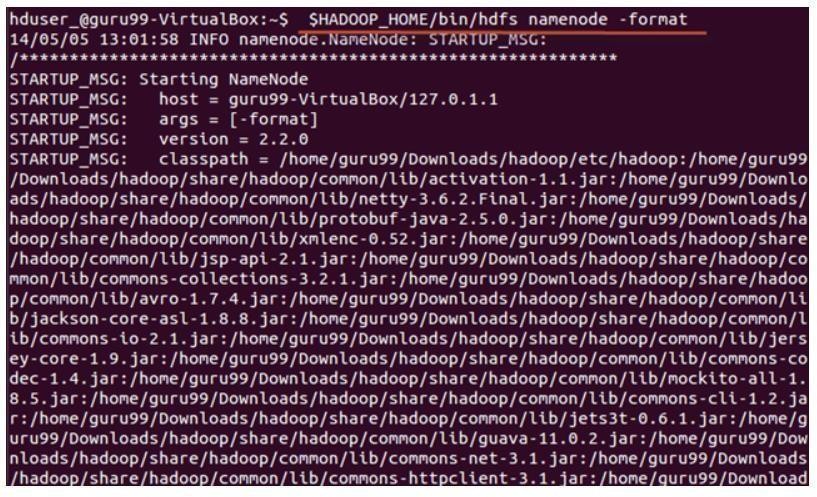
sudo mkdir -p <Path of Directory used in above setting> sudo mkdir -p /home/hduser\_/hdfs

sudo chown -R hduser\_:hadoop\_ <Path of Directory created in above step> sudo chown -R hduser\_:hadoop\_ /home/hduser\_/hdfs



sudo chmod 750 <Path of Directory created in above step> sudo chmod 750 /home/hduser\_/hdfs

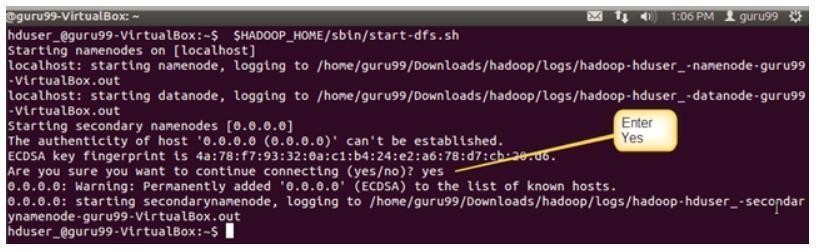
**Step 4)** Before we start Hadoop for the first time, format HDFS using below command

$HADOOP\_HOME/bin/hdfs namenode -format

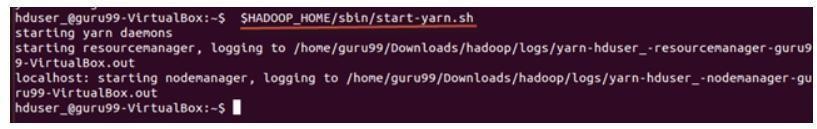
**Step 5)** Start Hadoop single node cluster using below command

$HADOOP\_HOME/sbin/start-dfs.sh

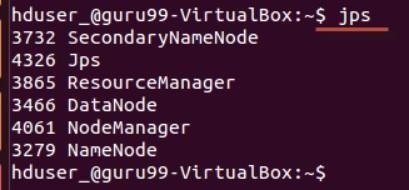
An output of above command



$HADOOP\_HOME/sbin/start-yarn.sh



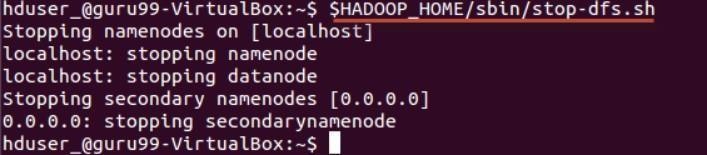
Using **'jps'** tool/command, verify whether all the Hadoop related processes are running or not.



If Hadoop has started successfully then an output of jps should show NameNode, NodeManager, ResourceManager, SecondaryNameNode, DataNode.

**Step 6)** Stopping Hadoop

$HADOOP\_HOME/sbin/stop-dfs.sh



$HADOOP\_HOME/sbin/stop-yarn.sh



**Exp No: 12**

NoSQL Queries using HBase

**Creating a Table using HBase Shell**

You can create a table using the **create** command, here you must specify the table name and the Column Family name. The **syntax** to create a table in HBase shell is shown below.

**create** ‘<table name>’,’<column family>’

###### Example

Given below is a sample schema of a table named emp. It has two column families: “personal data” and “professional data”.

|  |  |  |
| --- | --- | --- |
| **Row key** | **personal data** | **professional data** |
|  |  |  |
|  |  |  |

You can create this table in HBase shell as shown below. hbase(main):002:0> create 'emp', 'personal data', 'professional data' And it will give you the following output.

0 row(s) in 1.1300 seconds

=> Hbase::Table – emp

###### Verification

You can verify whether the table is created using the **list** command as shown below. Here you can observe the created emp table.

hbase(main):002:0> list TABLE

emp

2 row(s) in 0.0340 seconds

**Listing a Table using HBase Shell**

**list** is the command that is used to list all the tables in HBase. Given below is the syntax of the list command.

hbase(main):001:0 > list

When you type this command and execute in HBase prompt, it will display the list of all the tables in HBase as shown below.

hbase(main):001:0> list TABLE

emp

Here you can observe a table named emp.

#### Enabling a Table using HBase Shell

Syntax to enable a table:

enable ‘emp’ Example

Given below is an example to enable a table.

hbase(main):005:0> enable 'emp' 0 row(s) in 0.4580 seconds

###### is\_enabled

This command is used to find whether a table is enabled. Its syntax is as follows: hbase> is\_enabled 'table name'

The following code verifies whether the table named **emp** is enabled. If it is enabled, it will return true and if not, it will return false.

hbase(main):031:0> is\_enabled 'emp' true

0 row(s) in 0.0440 seconds

#### describe

This command returns the description of the table. Its syntax is as follows: hbase> describe 'table name'

Given below is the output of the describe command on the **emp** table.

hbase(main):006:0> describe 'emp' DESCRIPTION

ENABLED

'emp', {NAME ⇒ 'READONLY', DATA\_BLOCK\_ENCODING ⇒ 'NONE', BLOOMFILTER

⇒ 'ROW', REPLICATION\_SCOPE ⇒ '0', COMPRESSION ⇒ 'NONE', VERSIONS ⇒

'1', TTL true

⇒ 'FOREVER', MIN\_VERSIONS ⇒ '0', KEEP\_DELETED\_CELLS ⇒ 'false', BLOCKSIZE ⇒ '65536', IN\_MEMORY ⇒ 'false', BLOCKCACHE ⇒ 'true'}, {NAME

⇒ 'personal

data', DATA\_BLOCK\_ENCODING ⇒ 'NONE', BLOOMFILTER ⇒ 'ROW', REPLICATION\_SCOPE ⇒ '0', VERSIONS ⇒ '5', COMPRESSION ⇒ 'NONE', MIN\_VERSIONS ⇒ '0', TTL

⇒ 'FOREVER', KEEP\_DELETED\_CELLS ⇒ 'false', BLOCKSIZE ⇒ '65536',

IN\_MEMORY ⇒ 'false', BLOCKCACHE ⇒ 'true'}, {NAME ⇒ 'professional data', DATA\_BLO

CK\_ENCODING ⇒ 'NONE', BLOOMFILTER ⇒ 'ROW', REPLICATION\_SCOPE ⇒ '0', VERSIONS ⇒ '1', COMPRESSION ⇒ 'NONE', MIN\_VERSIONS ⇒ '0', TTL ⇒ 'FOREVER', K

EEP\_DELETED\_CELLS ⇒ 'false', BLOCKSIZE ⇒ '65536', IN\_MEMORY ⇒

'false', BLOCKCACHE ⇒ 'true'}, {NAME ⇒ 'table\_att\_unset', DATA\_BLOCK\_ENCODING ⇒ 'NO

NE', BLOOMFILTER ⇒ 'ROW', REPLICATION\_SCOPE ⇒ '0', COMPRESSION ⇒ 'NONE', VERSIONS ⇒ '1', TTL ⇒ 'FOREVER', MIN\_VERSIONS ⇒ '0', KEEP\_DELETED\_CELLS

⇒ 'false', BLOCKSIZE ⇒ '6

#### Alter

##### Alter is the command used to make changes to an existing table. Using this command, you can change the maximum number of cells of a column family, set and delete table scope operators, and delete a column family from a table.

Changing the Maximum Number of Cells of a Column Family

Given below is the syntax to change the maximum number of cells of a column family. hbase> alter 't1', NAME ⇒ 'f1', VERSIONS ⇒ 5

In the following example, the maximum number of cells is set to 5.

hbase(main):003:0> alter 'emp', NAME ⇒ 'personal data', VERSIONS ⇒ 5 Updating all regions with the new schema...

0/1 regions updated. 1/1 regions updated. Done.

0 row(s) in 2.3050 seconds

#### Existence of Table using HBase Shell

You can verify the existence of a table using the **exists** command. The following example shows how to use this command.

hbase(main):024:0> exists 'emp' Table emp does exist

0 row(s) in 0.0750 seconds

==================================================================

hbase(main):015:0> exists 'student' Table student does not exist

0 row(s) in 0.0480 seconds

#### Dropping a Table using HBase Shell

Using the **drop** command, you can delete a table. Before dropping a table, you have to disable it. hbase(main):018:0> disable 'emp'

0 row(s) in 1.4580 seconds

hbase(main):019:0> drop 'emp' 0 row(s) in 0.3060 seconds

Verify whether the table is deleted using the exists command. hbase(main):020:07gt; exists 'emp'

Table emp does not exist

0 row(s) in 0.0730 seconds

*drop\_all*

This command is used to drop the tables matching the “regex” given in the command. Its syntax is as follows:

hbase> drop\_all ‘t.\*’

**Note:** Before dropping a table, you must disable it.

Example

Assume there are tables named raja, rajani, rajendra, rajesh, and raju. hbase(main):017:0> list

TABLE

raja rajani rajendra rajesh

9 row(s) in 0.0270 seconds

All these tables start with the letters **raj**. First of all, let us disable all these tables using the **disable\_all** command as shown below.

hbase(main):002:0> disable\_all 'raj.\*' raja

rajani rajendra rajesh raju

Disable the above 5 tables (y/n)? y

5 tables successfully disabled

Now you can delete all of them using the **drop\_all** command as given below. hbase(main):018:0> drop\_all 'raj.\*'

raja

rajani rajendra rajesh raju

Drop the above 5 tables (y/n)? y

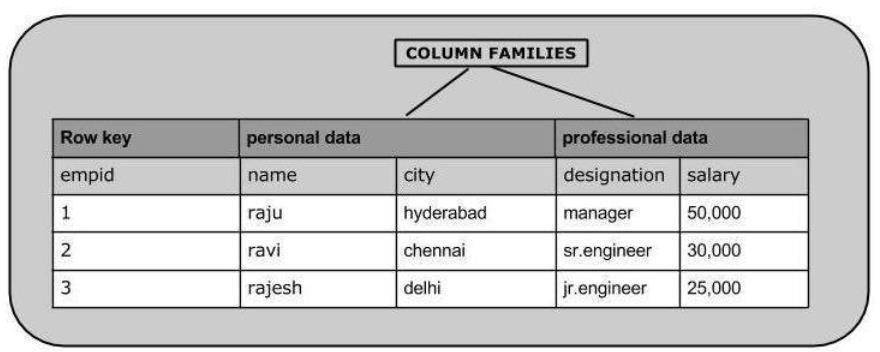
5 tables successfully dropped

#### Inserting Data using HBase Shell

To create data in an HBase table, the following commands and methods are used:

* **put** command,
* **add()** method of **Put** class, and
* **put()** method of **HTable** class.

As an example, we are going to create the following table in HBase.



Using **put** command, you can insert rows into a table. Its syntax is as follows: put ’<table name>’,’row1’,’<colfamily:colname>’,’<value>’

Inserting the First Row

Let us insert the first row values into the emp table as shown below. hbase(main):005:0> put 'emp','1','personal data:name','raju'

0 row(s) in 0.6600 seconds

hbase(main):006:0> put 'emp','1','personal data:city','hyderabad' 0 row(s) in 0.0410 seconds

hbase(main):007:0> put 'emp','1','professional data:designation','manager'

1. row(s) in 0.0240 seconds

hbase(main):007:0> put 'emp','1','professional data:salary','50000' 0 row(s) in 0.0240 seconds

###### Insert the remaining rows using the put command in the same way. If you insert the whole table, you will get the following output.

hbase(main):022:0> scan 'emp'

ROW COLUMN+CELL

1. column=personal data:city, timestamp=1417524216501, value=hyderabad 1 column=personal data:name, timestamp=1417524185058, value=ramu

1 column=professional data:designation, timestamp=1417524232601, value=manager

1. column=professional data:salary, timestamp=1417524244109, value=50000 2 column=personal data:city, timestamp=1417524574905, value=chennai
2. column=personal data:name, timestamp=1417524556125, value=ravi 2 column=professional data:designation, timestamp=1417524592204, value=sr:engg
3. column=professional data:salary, timestamp=1417524604221, value=30000 3 column=personal data:city, timestamp=1417524681780, value=delhi
4. column=personal data:name, timestamp=1417524672067, value=rajesh 3 column=professional data:designation, timestamp=1417524693187, value=jr:engg

value=25000

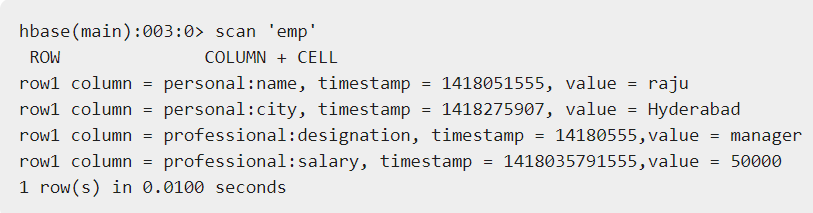
#### Updating Data using HBase Shell

You can update an existing cell value using the **put** command. To do so, just follow the same syntax and mention your new value as shown below.

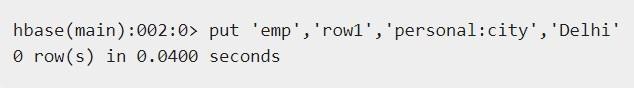
put ‘table name’,’row ’,'Column family:column name',’new value’ The newly given value replaces the existing value, updating the row.

Example

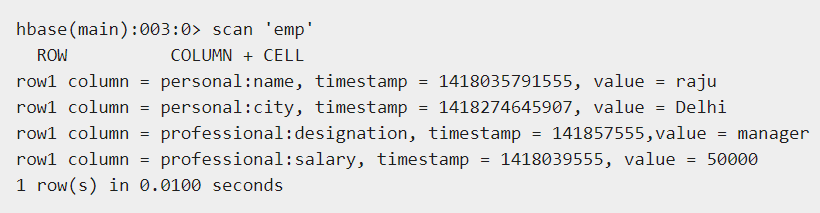
Suppose there is a table in HBase called **emp** with the following data.



The following command will update the city value of the employee named ‘Raju’ to Delhi.



The updated table looks as follows where you can observe the city of Raju has been changed to ‘Delhi’.



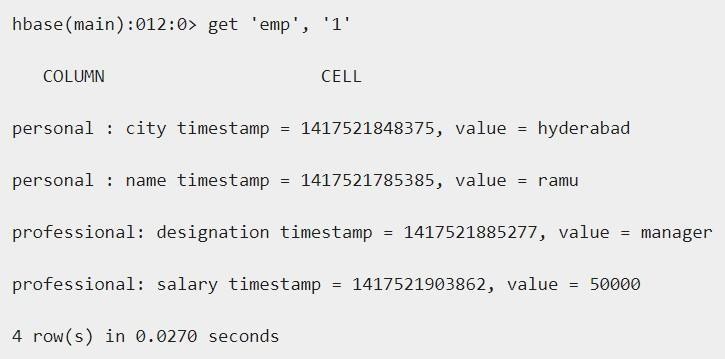
#### Reading Data using HBase Shell

The **get** command and the **get()** method of **HTable** class are used to read data from a table in HBase. Using **get** command, you can get a single row of data at a time. Its syntax is as follows:



Example

The following example shows how to use the get command. Let us scan the first row of the **emp** table.



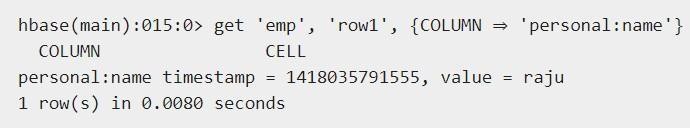
##### Reading a Specific Column

Given below is the syntax to read a specific column using the **get** method.



Example

Given below is the example to read a specific column in HBase table.



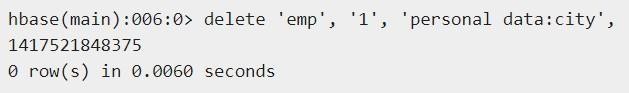
# Deleting a Specific Cell in a Table

Using the **delete** command, you can delete a specific cell in a table. The syntax of **delete** command is as follows:



Example

Here is an example to delete a specific cell. Here we are deleting the salary.



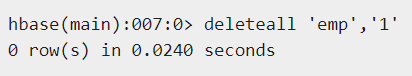
###### Deleting All Cells in a Table

Using the “deleteall” command, you can delete all the cells in a row. Given below is the syntax of deleteall command.



###### Example

Here is an example of “deleteall” command, where we are deleting all the cells of row1 of emp table.



Verify the table using the **scan** command. A snapshot of the table after deleting the table is given below.

