

# HBASE - INSTALLATION

[http://www.tutorialspoint.com/hbase/hbase\\_installation.htm](http://www.tutorialspoint.com/hbase/hbase_installation.htm)

Copyright © tutorialspoint.com

This chapter explains how HBase is installed and initially configured. Java and Hadoop are required to proceed with HBase, so you have to download and install java and Hadoop in your system.

## Pre-Installation Setup

Before installing Hadoop into Linux environment, we need to set up Linux using **ssh** *SecureShell*. Follow the steps given below for setting up the Linux environment.

## Creating a User

First of all, it is recommended to create a separate user for Hadoop to isolate the Hadoop file system from the Unix file system. Follow the steps given below to create a user.

- Open the root using the command “su”.
- Create a user from the root account using the command “useradd username”.
- Now you can open an existing user account using the command “su username”.

Open the Linux terminal and type the following commands to create a user.

```
$ su
password:
# useradd hadoop
# passwd hadoop
New passwd:
Retype new passwd
```

## SSH Setup and Key Generation

SSH setup is required to perform different operations on the cluster such as start, stop, and distributed daemon shell operations. To authenticate different users of Hadoop, it is required to provide public/private key pair for a Hadoop user and share it with different users.

The following commands are used to generate a key value pair using SSH. Copy the public keys from id\_rsa.pub to authorized\_keys, and provide owner, read and write permissions to authorized\_keys file respectively.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

## Verify ssh

```
ssh localhost
```

## Installing Java

Java is the main prerequisite for Hadoop and HBase. 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 "1.7.0_71"
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
```

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

## Step 1

Download java *JDK < latestversion > - X64.tar. gz* by visiting the following link [Oracle Java](#).

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 xzf 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 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 apply all the changes into the current running system.

```
$ source ~/.bashrc
```

## Step 5

Use the following commands to configure java alternatives:

```
# alternatives --install /usr/bin/java java usr/local/java/bin/java 2  
# alternatives --install /usr/bin/javac javac usr/local/java/bin/javac 2  
# alternatives --install /usr/bin/jar jar usr/local/java/bin/jar 2  
  
# alternatives --set java usr/local/java/bin/java  
# alternatives --set javac usr/local/java/bin/javac  
# alternatives --set jar usr/local/java/bin/jar
```

Now verify the **java -version** command from the terminal as explained above.

## Downloading Hadoop

After installing java, you have to install Hadoop. First of all, verify the existence of Hadoop using “Hadoop version ” command as shown below.

```
hadoop version
```

If everything works fine, it will give you the following output.

```
Hadoop 2.6.0
Compiled by jenkins on 2014-11-13T21:10Z
Compiled with protoc 2.5.0
From source with checksum 18e43357c8f927c0695f1e9522859d6a
This command was run using
/home/hadoop/hadoop/share/hadoop/common/hadoop-common-2.6.0.jar
```

If your system is unable to locate Hadoop, then download Hadoop in your system. Follow the commands given below to do so.

Download and extract [hadoop-2.6.0](#) from Apache Software Foundation using the following commands.

```
$ su
password:
# cd /usr/local
# wget http://mirrors.advancedhosters.com/apache/hadoop/common/hadoop-
2.6.0/hadoop-2.6.0-src.tar.gz
# tar xzf hadoop-2.6.0-src.tar.gz
# mv hadoop-2.6.0/* hadoop/
# exit
```

## Installing Hadoop

Install Hadoop in any of the required mode. Here, we are demonstrating HBase functionalities in pseudo distributed mode, therefore install Hadoop in pseudo distributed mode.

The following steps are used for installing **Hadoop 2.4.1**.

### Step 1 - Setting up Hadoop

You can set Hadoop environment variables by appending the following commands to `~/.bashrc` file.

```
export HADOOP_HOME=/usr/local/hadoop
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_INSTALL=$HADOOP_HOME
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

### Step 2 - Hadoop Configuration

You can find all the Hadoop configuration files in the location “\$HADOOP\_HOME/etc/hadoop”. You need to make changes in those configuration files according to your Hadoop infrastructure.

```
$ cd $HADOOP_HOME/etc/hadoop
```

In order to develop Hadoop programs in java, you have to reset the java environment variable in **hadoop-env.sh** file by replacing **JAVA\_HOME** value with the location of java in your system.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
```

You will have to edit the following files to configure Hadoop.

### core-site.xml

The **core-site.xml** file contains information such as the port number used for Hadoop instance, memory allocated for file system, memory limit for storing data, and the size of Read/Write buffers.

Open core-site.xml and add the following properties in between the <configuration> and </configuration> tags.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

### hdfs-site.xml

The **hdfs-site.xml** file contains information such as the value of replication data, namenode path, and datanode path of your local file systems, where you want to store the Hadoop infrastructure.

Let us assume the following data.

```
dfs.replication (data replication value) = 1
(In the below given path /hadoop/ is the user name.
hadoopinfra/hdfs/namenode is the directory created by hdfs file system.)

namenode path = //home/hadoop/hadoopinfra/hdfs/namenode
(hadoopinfra/hdfs/datanode is the directory created by hdfs file system.)

datanode path = //home/hadoop/hadoopinfra/hdfs/datanode
```

Open this file and add the following properties in between the <configuration>, </configuration> tags.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/namenode</value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/datanode</value>
  </property>
</configuration>
```

**Note:** In the above file, all the property values are user-defined and you can make changes according to your Hadoop infrastructure.

### yarn-site.xml

This file is used to configure yarn into Hadoop. Open the yarn-site.xml file and add the following property in between the <configuration>;, </configuration>; tags in this file.

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

## mapred-site.xml

This file is used to specify which MapReduce framework we are using. By default, Hadoop contains a template of yarn-site.xml. First of all, it is required to copy the file from **mapred-site.xml.template** to **mapred-site.xml** file using the following command.

```
$ cp mapred-site.xml.template mapred-site.xml
```

Open **mapred-site.xml** file and add the following properties in between the <configuration> and </configuration> tags.

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

## Verifying Hadoop Installation

The following steps are used to verify the Hadoop installation.

### Step 1 - Name Node Setup

Set up the namenode using the command “hdfs namenode -format” as follows.

```
$ cd ~
$ hdfs namenode -format
```

The expected result is as follows.

```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = localhost/192.168.1.11
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 2.4.1
...
...
10/24/14 21:30:56 INFO common.Storage: Storage directory
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully formatted.
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to
retain 1 images with txid >= 0
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11
*****/
```

### Step 2 - Verifying Hadoop dfs

The following command is used to start dfs. Executing this command will start your Hadoop file system.

```
$ start-dfs.sh
```

The expected output is as follows.

```
10/24/14 21:37:56
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hadoop/hadoop-
2.4.1/logs/hadoop-hadoop-namenode-localhost.out
localhost: starting datanode, logging to /home/hadoop/hadoop-
2.4.1/logs/hadoop-hadoop-datanode-localhost.out
Starting secondary namenodes [0.0.0.0]
```

### Step 3 - Verifying Yarn Script

The following command is used to start the yarn script. Executing this command will start your yarn daemons.

```
$ start-yarn.sh
```

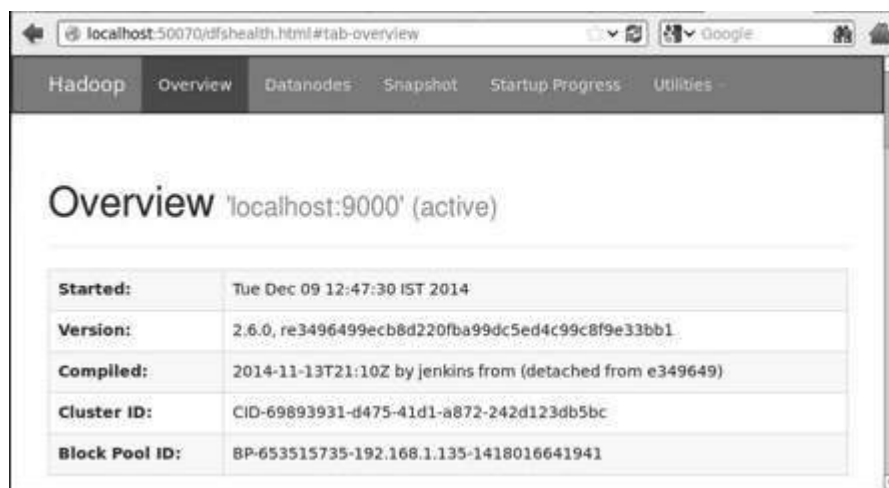
The expected output is as follows.

```
starting yarn daemons
starting resourcemanager, logging to /home/hadoop/hadoop-
2.4.1/logs/yarn-hadoop-resourcemanager-localhost.out
localhost: starting nodemanager, logging to /home/hadoop/hadoop-
2.4.1/logs/yarn-hadoop-nodemanager-localhost.out
```

## Step 4 - Accessing Hadoop on Browser

The default port number to access Hadoop is 50070. Use the following url to get Hadoop services on your browser.

http://localhost:50070



## Step 5 - Verify all Applications of Cluster

The default port number to access all the applications of cluster is 8088. Use the following url to visit this service.

http://localhost:8088/



## Installing HBase

We can install HBase in any of the three modes: Standalone mode, Pseudo Distributed mode, and Fully Distributed mode.

### Installing HBase in Standalone Mode

Download the latest stable version of HBase from <http://www.interior-dsgn.com/apache/hbase/stable/> using “wget” command, and extract it using the tar “zxvf” command. See the following command.

```
$cd usr/local/  
$wget http://www.interior-dsgn.com/apache/hbase/stable/hbase-0.98.8-hadoop2-bin.tar.gz  
$tar -zxvf hbase-0.98.8-hadoop2-bin.tar.gz
```

Shift to super user mode and move the HBase folder to /usr/local as shown below.

```
$su  
$password: enter your password here  
mv hbase-0.99.1/* Hbase/
```

### Configuring HBase in Standalone Mode

Before proceeding with HBase, you have to edit the following files and configure HBase.

#### hbase-env.sh

Set the java Home for HBase and open **hbase-env.sh** file from the conf folder. Edit JAVA\_HOME environment variable and change the existing path to your current JAVA\_HOME variable as shown below.

```
cd /usr/local/Hbase/conf  
gedit hbase-env.sh
```

This will open the env.sh file of HBase. Now replace the existing **JAVA\_HOME** value with your current value as shown below.

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0
```

#### hbase-site.xml

This is the main configuration file of HBase. Set the data directory to an appropriate location by opening the HBase home folder in /usr/local/HBase. Inside the conf folder, you will find several files, open the **hbase-site.xml** file as shown below.

```
#cd /usr/local/HBase/  
#cd conf  
# gedit hbase-site.xml
```

Inside the **hbase-site.xml** file, you will find the <configuration> and </configuration> tags. Within them, set the HBase directory under the property key with the name “hbase.rootdir” as shown below.

```
<configuration>  
  //Here you have to set the path where you want HBase to store its files.  
  <property>  
    <name>hbase.rootdir</name>  
    <value>file:/home/hadoop/HBase/HFiles</value>  
  </property>  
  
  //Here you have to set the path where you want HBase to store its built in zookeeper
```

```
files.  
  <property>  
    <name>hbase.zookeeper.property.dataDir</name>  
    <value>/home/hadoop/zookeeper</value>  
  </property>  
</configuration>
```

With this, the HBase installation and configuration part is successfully complete. We can start HBase by using **start-hbase.sh** script provided in the bin folder of HBase. For that, open HBase Home Folder and run HBase start script as shown below.

```
$cd /usr/local/HBase/bin  
$./start-hbase.sh
```

If everything goes well, when you try to run HBase start script, it will prompt you a message saying that HBase has started.

```
starting master, logging to /usr/local/HBase/bin/../logs/hbase-tpmaster-  
localhost.localdomain.out
```

## Installing HBase in Pseudo-Distributed Mode

Let us now check how HBase is installed in pseudo-distributed mode.

## Configuring HBase

Before proceeding with HBase, configure Hadoop and HDFS on your local system or on a remote system and make sure they are running. Stop HBase if it is running.

### hbase-site.xml

Edit hbase-site.xml file to add the following properties.

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

It will mention in which mode HBase should be run. In the same file from the local file system, change the hbase.rootdir, your HDFS instance address, using the hdfs:/// URI syntax. We are running HDFS on the localhost at port 8030.

```
<property>  
  <name>hbase.rootdir</name>  
  <value>hdfs://localhost:8030/hbase</value>  
</property>
```

## Starting HBase

After configuration is over, browse to HBase home folder and start HBase using the following command.

```
$cd /usr/local/HBase  
$bin/start-hbase.sh
```

**Note:** Before starting HBase, make sure Hadoop is running.

## Checking the HBase Directory in HDFS

HBase creates its directory in HDFS. To see the created directory, browse to Hadoop bin and type the following command.



```
$ ./bin/hadoop fs -ls /hbase
```

If everything goes well, it will give you the following output.

```
Found 7 items
drwxr-xr-x - hbase users 0 2014-06-25 18:58 /hbase/.tmp
drwxr-xr-x - hbase users 0 2014-06-25 21:49 /hbase/WALs
drwxr-xr-x - hbase users 0 2014-06-25 18:48 /hbase/corrupt
drwxr-xr-x - hbase users 0 2014-06-25 18:58 /hbase/data
-rw-r--r-- 3 hbase users 42 2014-06-25 18:41 /hbase/hbase.id
-rw-r--r-- 3 hbase users 7 2014-06-25 18:41 /hbase/hbase.version
drwxr-xr-x - hbase users 0 2014-06-25 21:49 /hbase/oldWALs
```

## Starting and Stopping a Master

Using the “local-master-backup.sh” you can start up to 10 servers. Open the home folder of HBase, master and execute the following command to start it.

```
$ ./bin/local-master-backup.sh 2 4
```

To kill a backup master, you need its process id, which will be stored in a file named “**/tmp/hbase-USER-X-master.pid.**” you can kill the backup master using the following command.

```
$ cat /tmp/hbase-user-1-master.pid |xargs kill -9
```

## Starting and Stopping RegionServers

You can run multiple region servers from a single system using the following command.

```
$ .bin/local-regionservers.sh start 2 3
```

To stop a region server, use the following command.

```
$ .bin/local-regionservers.sh stop 3
```

## Starting HBaseShell

After Installing HBase successfully, you can start HBase Shell. Below given are the sequence of steps that are to be followed to start the HBase shell. Open the terminal, and login as super user.

## Start Hadoop File System

Browse through Hadoop home sbin folder and start Hadoop file system as shown below.

```
$cd $HADOOP_HOME/sbin
$start-all.sh
```

## Start HBase

Browse through the HBase root directory bin folder and start HBase.

```
$cd /usr/local/HBase
$./bin/start-hbase.sh
```

## Start HBase Master Server

This will be the same directory. Start it as shown below.

```
./bin/local-master-backup.sh start 2 (number signifies specific
```

```
server.)
```

## Start Region

Start the region server as shown below.

```
$/bin/./local-regionservers.sh start 3
```

## Start HBase Shell

You can start HBase shell using the following command.

```
$cd bin
$./hbase shell
```

This will give you the HBase Shell Prompt as shown below.

```
2014-12-09 14:24:27,526 INFO [main] Configuration.deprecation:
hadoop.native.lib is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.98.8-hadoop2, r6cfc8d064754251365e070a10a82eb169956d5fe, Fri
Nov 14 18:26:29 PST 2014

hbase(main):001:0>
```

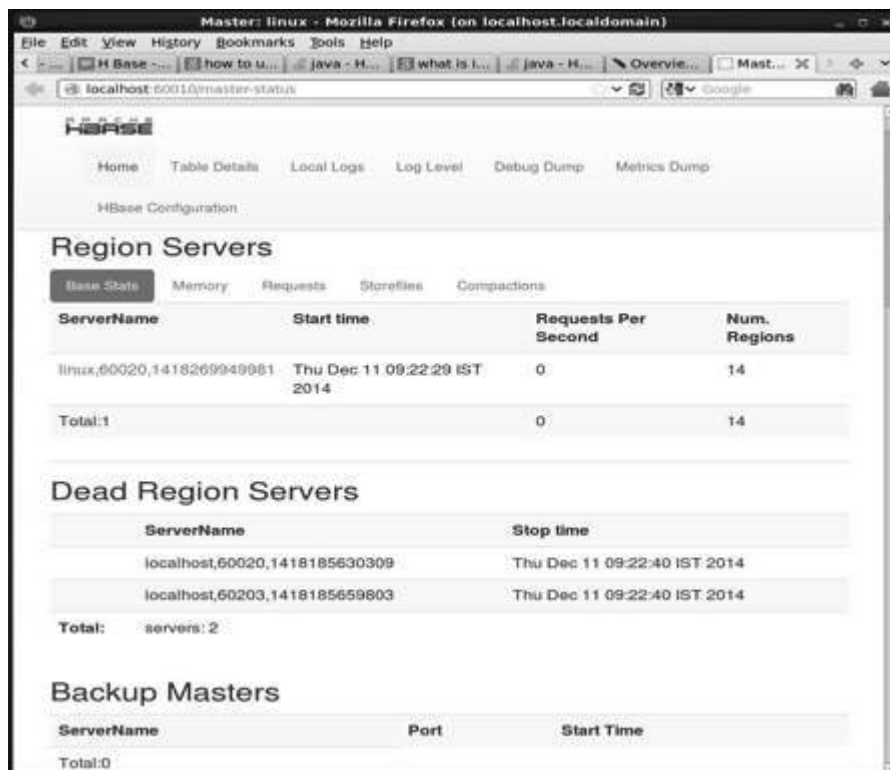
## HBase Web Interface

To access the web interface of HBase, type the following url in the browser.

```
http://localhost:60010
```

This interface lists your currently running Region servers, backup masters and HBase tables.

## HBase Region servers and Backup Masters



The screenshot shows the HBase Web Interface in a Mozilla Firefox browser window. The interface has a navigation bar with links: Home, Table Details, Local Logs, Log Level, Debug Dump, and Metrics Dump. Below the navigation bar is the 'HBase Configuration' section. The main content area is divided into three sections: 'Region Servers', 'Dead Region Servers', and 'Backup Masters'.

**Region Servers**

ServerName	Start time	Requests Per Second	Num. Regions
linux,60020,1418269949981	Thu Dec 11 09:22:29 IST 2014	0	14
Total:1		0	14

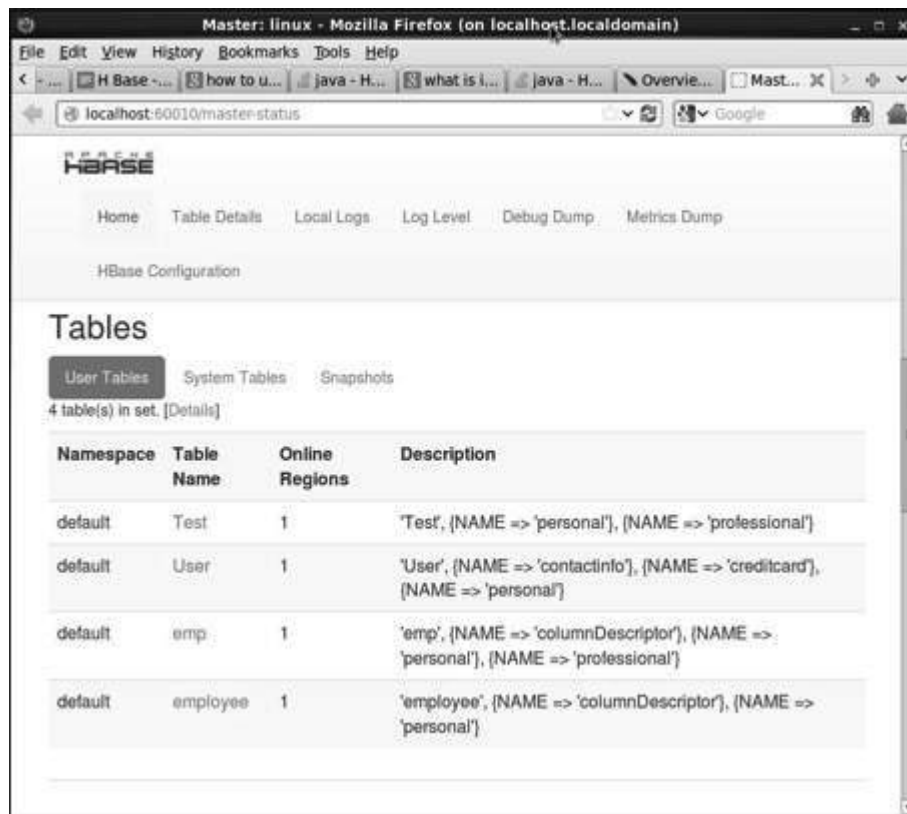
**Dead Region Servers**

ServerName	Stop time
localhost,60020,1418185630309	Thu Dec 11 09:22:40 IST 2014
localhost,60203,1418185659803	Thu Dec 11 09:22:40 IST 2014
Total: servers: 2	

**Backup Masters**

ServerName	Port	Start Time
Total:0		

## HBase Tables



## Setting Java Environment

We can also communicate with HBase using Java libraries, but before accessing HBase using Java API you need to set classpath for those libraries.

## Setting the Classpath

Before proceeding with programming, set the classpath to HBase libraries in **.bashrc** file. Open **.bashrc** in any of the editors as shown below.

```
$ gedit ~/.bashrc
```

Set classpath for HBase libraries *libfolderinHBase* in it as shown below.

```
export CLASSPATH = $CLASSPATH://home/hadoop/hbase/lib/*
```

This is to prevent the “class not found” exception while accessing the HBase using java API.

Loading [Mathjax]/jax/output/HTML-CSS/jax.js