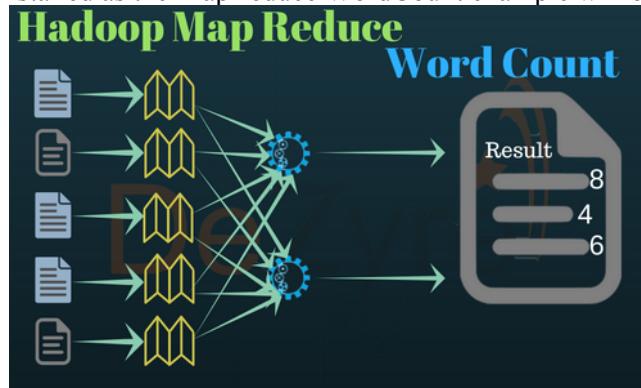


## Pre-requisites to follow this Hadoop WordCount Example Tutorial

- i. **Installation** must be completed successfully.
- ii. Single node hadoop cluster must be configured and running.
- iii. Eclipse must be installed as the MapReduce WordCount example will be run from eclipse IDE.



## Word Count - Hadoop Map Reduce Example -

Hadoop WordCount operation occurs in 3 stages -

- i. Mapper Phase
- ii. Shuffle Phase
- iii. Reducer Phase

### Hadoop WordCount Example- Mapper Phase Execution

The text from the input text file is tokenized into words to form a key value pair with all the words present in the input text file. The key is the word from the input file and value is '1'.

For instance if you consider the sentence “An elephant is an animal”. The mapper phase in the WordCount example will split the string into individual tokens i.e. words. In this case, the entire sentence will be split into 5 tokens (one for each word) with a value 1 as shown below -

Key-Value pairs from Hadoop Map Phase Execution-

(an,1)  
(elephant,1)  
(is,1)  
(an,1)  
(animal,1)

### Hadoop WordCount Example- Shuffle Phase Execution

After the map phase execution is completed successfully, shuffle phase is executed automatically wherein the key-value pairs generated in the map phase are taken as input and then sorted in alphabetical order.

After the shuffle phase is executed from the WordCount example code, the output will look like this -

(an,1)  
(an,1)  
(animal,1)  
(elephant,1)  
(is,1)

### Hadoop WordCount Example- Reducer Phase Execution

In the reduce phase, all the keys are grouped together and the values for similar keys are added up to find the occurrences for a particular word. It is like an aggregation phase for the keys generated by the map phase. The reducer phase takes the output of shuffle phase as input and then reduces the key-value pairs to unique keys with values added up. In our example “An elephant is an animal.” is the only word that

appears twice in the sentence. After the execution of the reduce phase of MapReduce WordCount example program, appears as a key only once but with a count of 2 as shown below -

(an,2)  
(animal,1)  
(elephant,1)  
(is,1)

This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file. An important point to note during the execution of the WordCount example is that the mapper class in the WordCount program will execute completely on the entire input file and not just a single sentence. Suppose if the input file has 15 lines then the mapper class will split the words of all the 15 lines and form initial key value pairs for the entire dataset. The reducer execution will begin only after the mapper phase is executed successfully.

### **Running the WordCount Example in Hadoop MapReduce using Java Project with Eclipse**

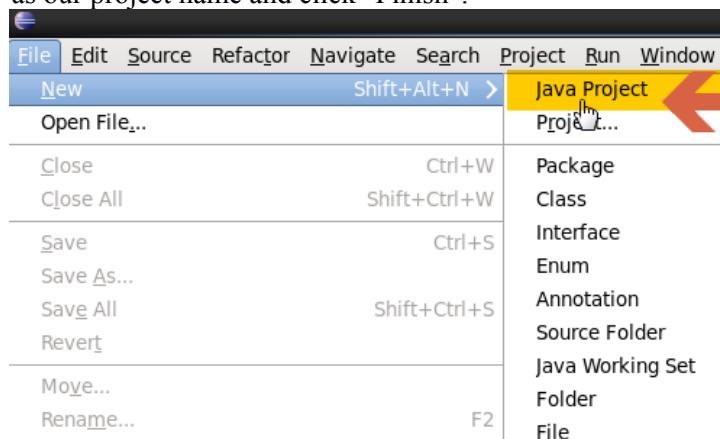
Now, let's create the WordCount java project with eclipse IDE for Hadoop. Even if you are working on Cloudera VM, creating the Java project can be applied to any environment.

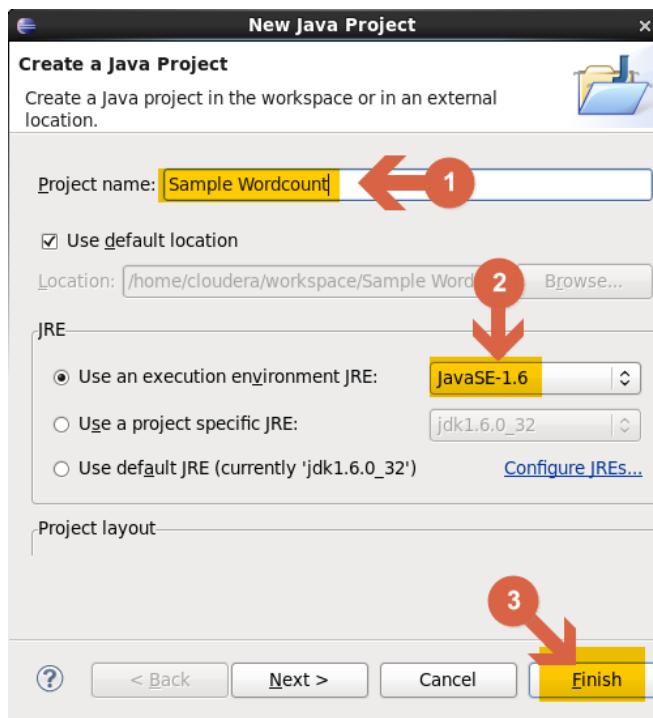
#### **Step 1 -**

Let's create the java project with the name "Sample WordCount" as shown below -

File > New > Project > Java Project > Next.

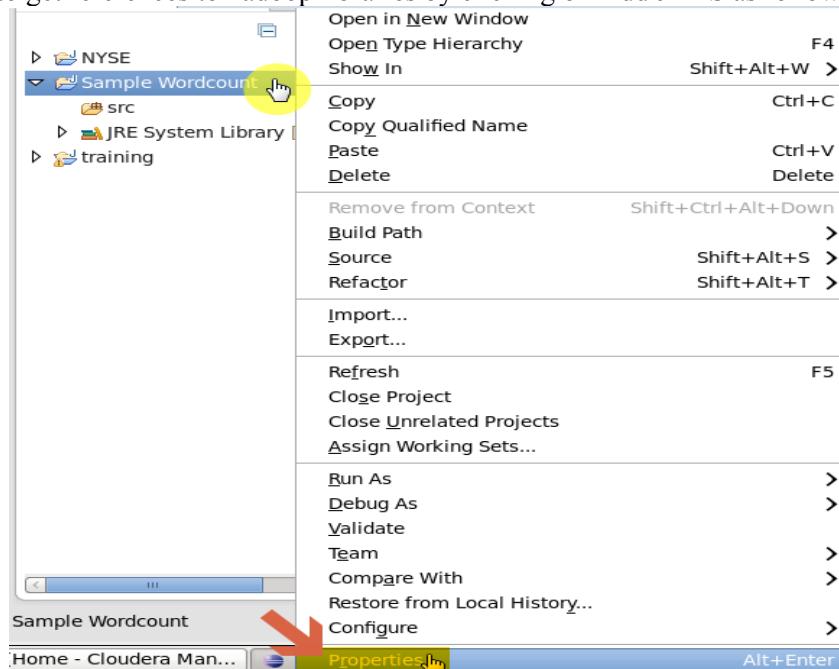
"Sample WordCount" as our project name and click "Finish":

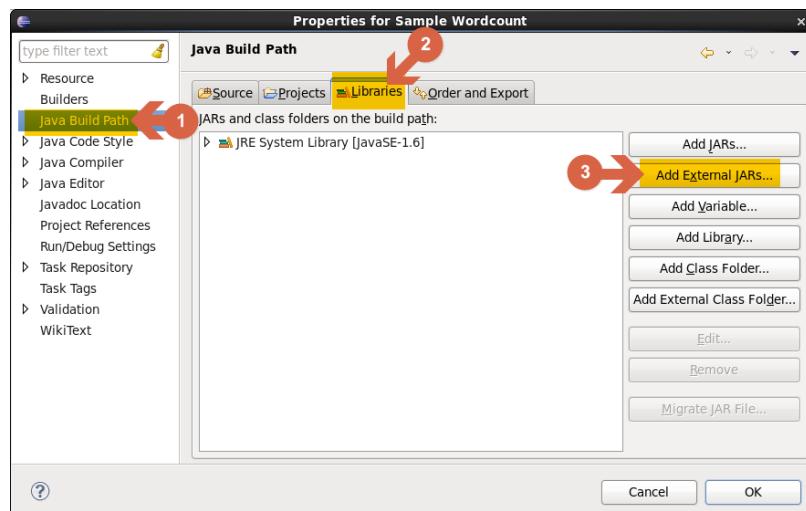




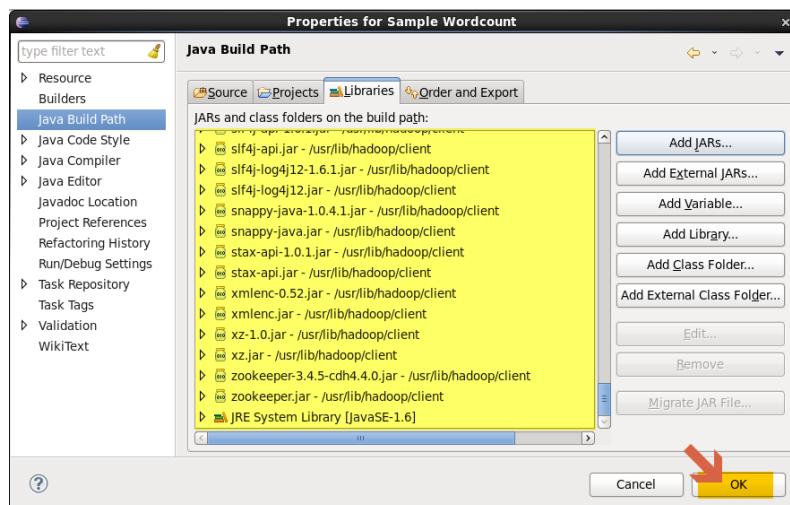
## Step 2 -

The next step is to get references to hadoop libraries by clicking on Add JARS as follows -



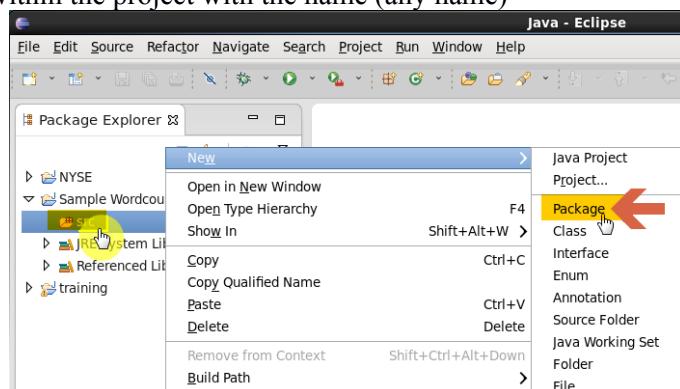


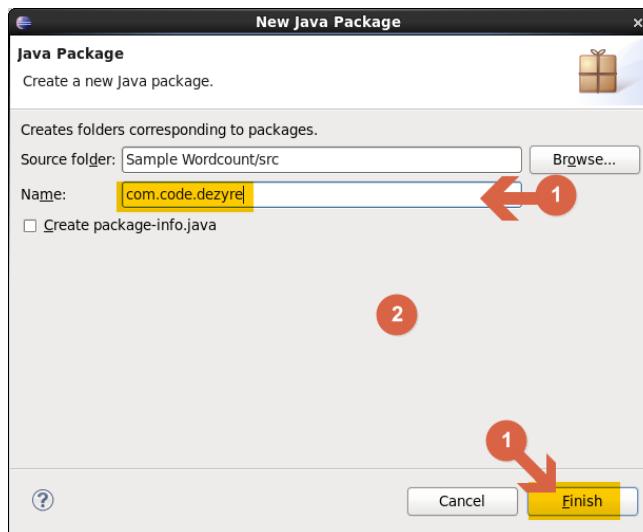
Go to the folder where hadoop is installed and add the jars.



### Step 3 -

Create a new package within the project with the name (any name)





#### Step 4 -

Now let's implement the WordCount example program by creating a WordCount class under the project package name

**New Java Class**

Create a new Java class.

Source folder: Sample Wordcount/src

Package: com.code.dezyre

Name: WordCount

Modifiers: public

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

□ public static void main(String[] args)  
□ Constructors from superclass

Finish

#### Step 5 -

Create a Mapper class within the WordCount class which extends MapReduceBase Class to implement mapper interface. The mapper class will contain -

1. Code to implement "map" method.

2. Code for implementing the mapper-stage business logic should be written within this method.

#### **Mapper Class Code for WordCount Example in Hadoop MapReduce**

```
public static class Map extends MapReduceBase implements Mapper {  
    private final static IntWritable one = new IntWritable(1);  
    private Text word = new Text();  
    public void map(LongWritable key, Text value, OutputCollector output, Reporter  
reporter) throws IOException {  
        String line = value.toString();  
        StringTokenizer tokenizer = new StringTokenizer(line);  
        while (tokenizer.hasMoreTokens()) {  
            word.set(tokenizer.nextToken());  
            output.collect(word, one);  
        }  
    }  
}
```

In the mapper class code, we have used the String Tokenizer class which takes the entire line and breaks into small tokens (string/word).

#### **Step 6 -**

Create a Reducer class within the WordCount class extending MapReduceBase Class to implement reducer interface. The reducer class for the wordcount example in hadoop will contain the -

1. Code to implement "reduce" method

2. Code for implementing the reducer-stage business logic should be written within this method

#### **Reducer Class Code for WordCount Example in Hadoop MapReduce**

```
public static class Reduce extends MapReduceBase implements Reducer {
```

```
    public void reduce(Text key, Iterator values, OutputCollector output,  
                      Reporter reporter) throws IOException {  
        int sum = 0;  
        while (values.hasNext()) {  
            sum += values.next().get();  
        }  
        output.collect(key, new IntWritable(sum));  
    }  
}
```

#### **Step 7 -**

Create main() method within the WordCount class and set the following properties using the JobConf class -

- i. OutputKeyClass
- ii. OutputValueClass
- iii. Mapper Class
- iv. Reducer Class
- v. InputFormat
- vi. OutputFormat
- vii. InputFilePath
- viii. OutputFolderPath

```
    public static void main(String[] args) throws Exception {
```

```

        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("WordCount");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);

        conf.setMapperClass(Map.class);
        //conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

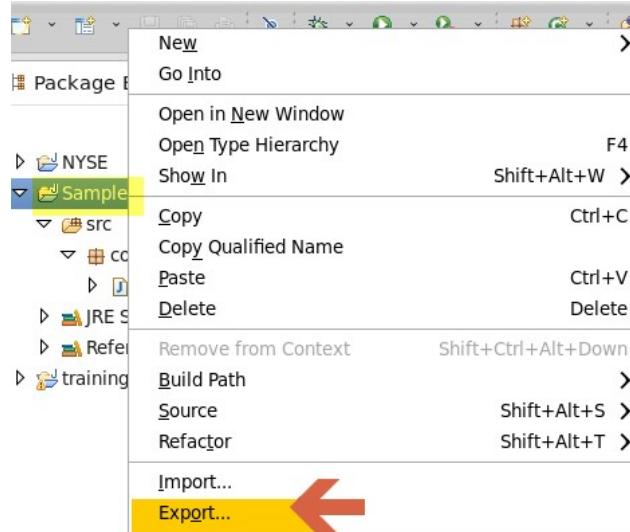
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

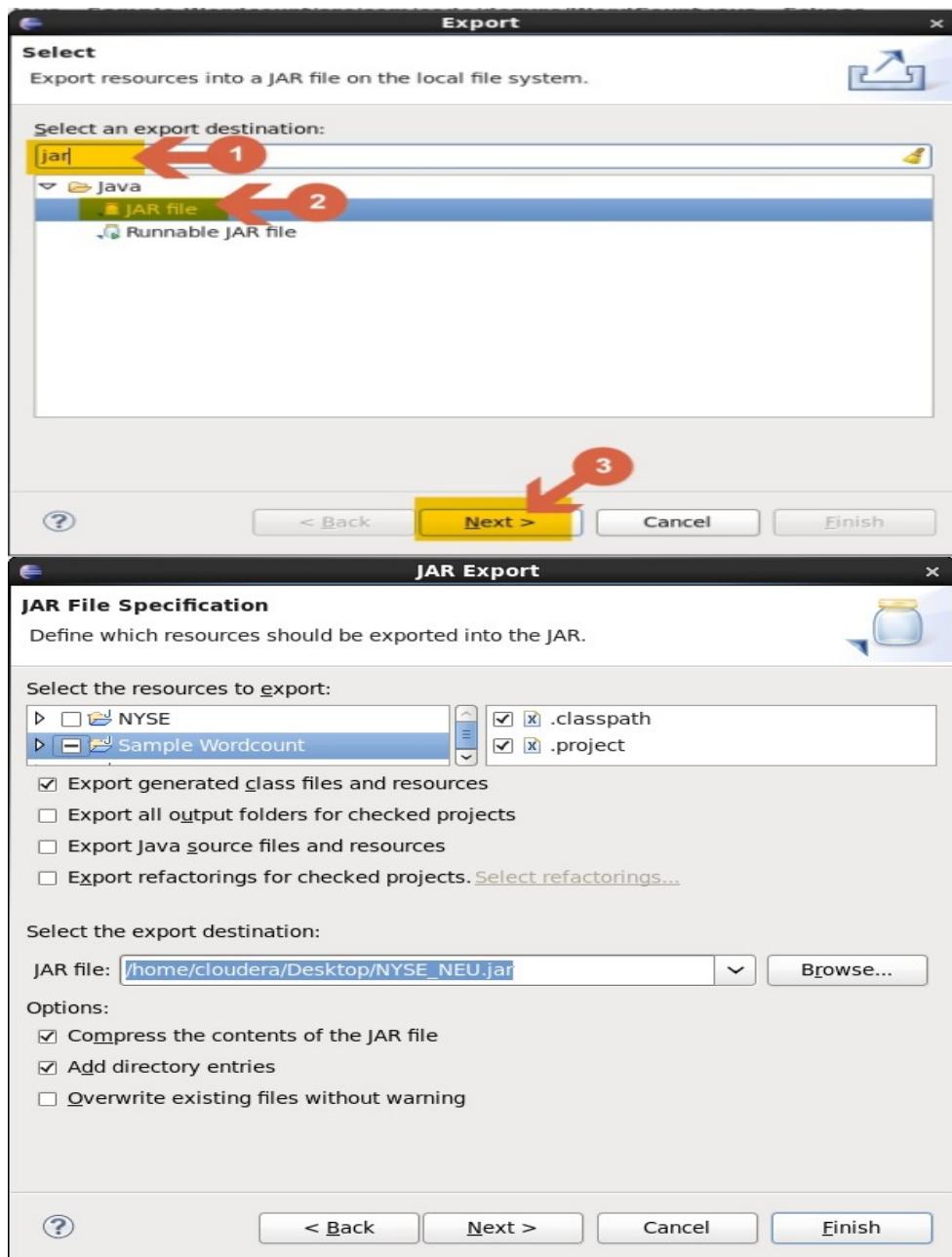
        JobClient.runJob(conf);
    }
}

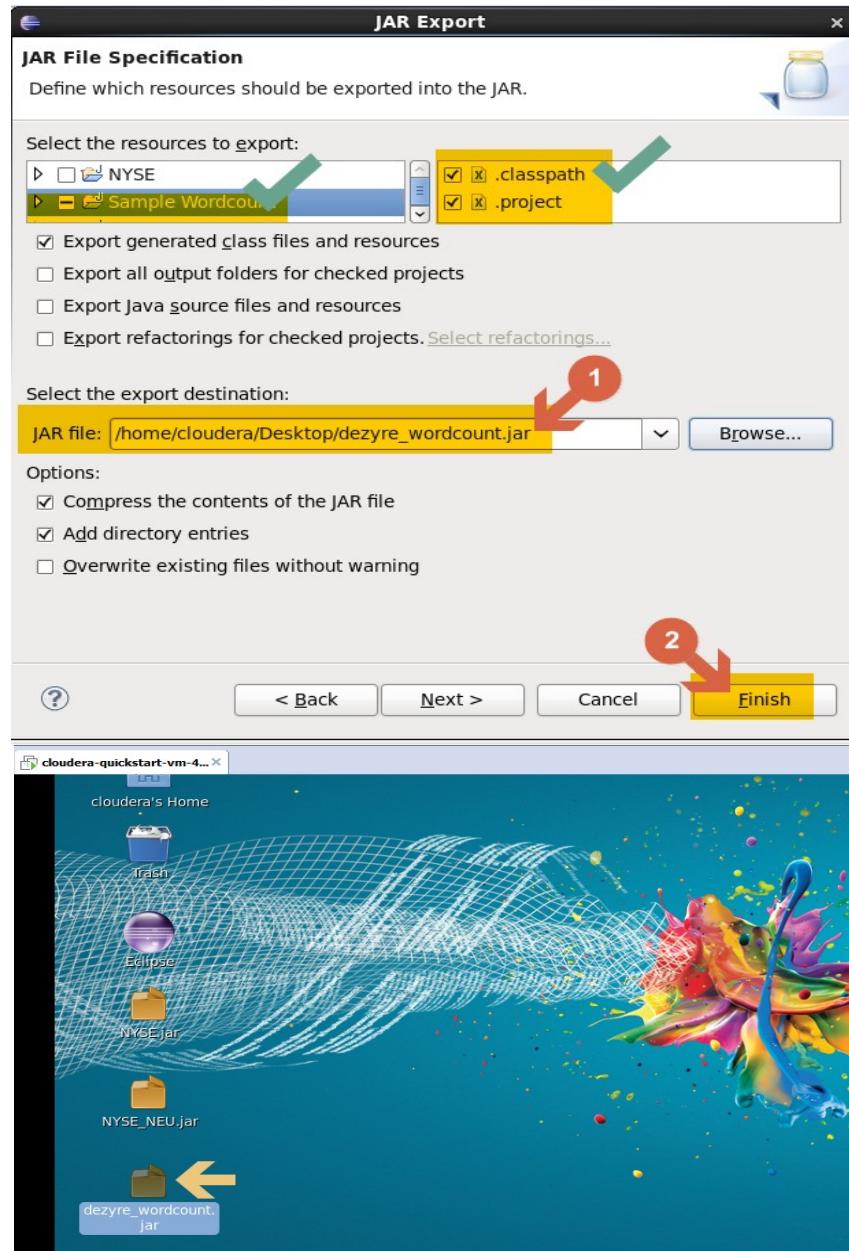
```

### Step 8 -

Create the JAR file for the wordcount class -







How to execute the Hadoop MapReduce WordCount program ?

```
cloudera@localhost:~/Desktop
File Edit View Search Terminal Help
[cloudera@localhost Desktop]$ hadoop jar dezyre_wordcount.jar com.code.dezyre.Wo
rdCount /user/cloudera/Input/war_and_peace /user/cloudera/Output
```

```

cloudera@localhost:~/Desktop
File Edit View Search Terminal Help
[cloudera@localhost Desktop]$ hadoop fs -mkdir /user/cloudera/Input/
[cloudera@localhost Desktop]$ hadoop fs -put war_and_peace /user/cloudera/Input/
[cloudera@localhost Desktop]$
[cloudera@localhost Desktop]$

```

### Output of Executing Hadoop WordCount Example -

File: [/user/cloudera/Output/part-00000](#)

Goto : [/user/cloudera/Output](#) go

[Go back to dir listing](#)  
[Advanced view/download options](#)

[View Next chunk](#)

```

"Come 1
"Dieu 1
"Dio 1
"From 1
"Grant 1
"La 4
"No 1
"Now 1
"Russia 1
"Sergey 1
"The 1
"To 1
"Tell 1
"What 1
"You 1
"-so 1
"...of 1
"5" 1

```

The program is run with the war and peace input file

```

*/
(base) dell-optiplex380@delloptiplex380-OptiPlex-380:~$ su - hduser
Password:
hduser@delloptiplex380-OptiPlex-380:~$ ls
Desktop examples.desktop Music Public Videos
Documents hadoop-1.2.1.tar.gz Output sample.txt Word.jar
Downloads hadoop-1.2.1.tar.gz.1 Pictures Templates

```

```

hduser@delloptiplex380-OptiPlex-380:~$ start-all.sh
Warning: $HADOOP_HOME is deprecated.

```

```

starting namenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-namenode-
delloptiplex380-OptiPlex-380.out
172.16.0.227: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-datanode-
delloptiplex380-OptiPlex-380.out
172.16.0.227: starting secondarynamenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-
secondarynamenode-delloptiplex380-OptiPlex-380.out
starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-jobtracker-delloptiplex380-
OptiPlex-380.out
172.16.0.227: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-tasktracker-
delloptiplex380-OptiPlex-380.out
hduser@delloptiplex380-OptiPlex-380:~$ jps
5058 -- main class information unavailable
4899 NameNode
5203 SecondaryNameNode
5288 JobTracker
5468 Jps

```

5423 TaskTracker

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop fs -mkdir /user/hduser/Input1
Warning: $HADOOP_HOME is deprecated.
```

mkdir: org.apache.hadoop.hdfs.server.namenode.SafeModeException: Cannot create directory  
/user/hduser/Input1. Name node is in safe mode.

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop dfsadmin -safemode leave
Warning: $HADOOP_HOME is deprecated.
```

Safe mode is OFF

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop fs -mkdir /user/hduser/Input1
Warning: $HADOOP_HOME is deprecated.
```

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop fs -put /home/hduser/sample.txt
/user/hduser/Input1/sample
Warning: $HADOOP_HOME is deprecated.
```

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop jar Word.jar packWordCount.WordCount
/user/hduser/Input1/sample /user/hduser/Output1
Warning: $HADOOP_HOME is deprecated.
```

```
19/10/01 11:58:13 INFO input.FileInputFormat: Total input paths to process : 1
19/10/01 11:58:13 INFO util.NativeCodeLoader: Loaded the native-hadoop library
19/10/01 11:58:13 WARN snappy.LoadSnappy: Snappy native library not loaded
19/10/01 11:58:13 INFO mapred.JobClient: Running job: job_201910011155_0001
19/10/01 11:58:14 INFO mapred.JobClient: map 0% reduce 0%
19/10/01 11:58:19 INFO mapred.JobClient: map 100% reduce 0%
19/10/01 11:58:27 INFO mapred.JobClient: map 100% reduce 33%
19/10/01 11:58:28 INFO mapred.JobClient: map 100% reduce 100%
19/10/01 11:58:29 INFO mapred.JobClient: Job complete: job_201910011155_0001
19/10/01 11:58:29 INFO mapred.JobClient: Counters: 29
19/10/01 11:58:29 INFO mapred.JobClient: Map-Reduce Framework
19/10/01 11:58:29 INFO mapred.JobClient: Spilled Records=362
19/10/01 11:58:29 INFO mapred.JobClient: Map output materialized bytes=2570
19/10/01 11:58:29 INFO mapred.JobClient: Reduce input records=181
19/10/01 11:58:29 INFO mapred.JobClient: Virtual memory (bytes) snapshot=3867643904
19/10/01 11:58:29 INFO mapred.JobClient: Map input records=19
19/10/01 11:58:29 INFO mapred.JobClient: SPLIT_RAW_BYTES=116
19/10/01 11:58:29 INFO mapred.JobClient: Map output bytes=2202
19/10/01 11:58:29 INFO mapred.JobClient: Reduce shuffle bytes=2570
19/10/01 11:58:29 INFO mapred.JobClient: Physical memory (bytes) snapshot=279097344
19/10/01 11:58:29 INFO mapred.JobClient: Reduce input groups=125
19/10/01 11:58:29 INFO mapred.JobClient: Combine output records=0
19/10/01 11:58:29 INFO mapred.JobClient: Reduce output records=125
19/10/01 11:58:29 INFO mapred.JobClient: Map output records=181
19/10/01 11:58:29 INFO mapred.JobClient: Combine input records=0
19/10/01 11:58:29 INFO mapred.JobClient: CPU time spent (ms)=1300
19/10/01 11:58:29 INFO mapred.JobClient: Total committed heap usage (bytes)=217055232
19/10/01 11:58:29 INFO mapred.JobClient: File Input Format Counters
19/10/01 11:58:29 INFO mapred.JobClient: Bytes Read=1496
19/10/01 11:58:29 INFO mapred.JobClient: FileSystemCounters
```

```

19/10/01 11:58:29 INFO mapred.JobClient: HDFS_BYTES_READ=1612
19/10/01 11:58:29 INFO mapred.JobClient: FILE_BYTES_WRITTEN=118507
19/10/01 11:58:29 INFO mapred.JobClient: FILE_BYTES_READ=2570
19/10/01 11:58:29 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=1293
19/10/01 11:58:29 INFO mapred.JobClient: Job Counters
19/10/01 11:58:29 INFO mapred.JobClient: Launched map tasks=1
19/10/01 11:58:29 INFO mapred.JobClient: Launched reduce tasks=1
19/10/01 11:58:29 INFO mapred.JobClient: SLOTS_MILLIS_REDUCES=9105
19/10/01 11:58:29 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots
(ms)=0
19/10/01 11:58:29 INFO mapred.JobClient: SLOTS_MILLIS_MAPS=4697
19/10/01 11:58:29 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots
(ms)=0
19/10/01 11:58:29 INFO mapred.JobClient: Data-local map tasks=1
19/10/01 11:58:29 INFO mapred.JobClient: File Output Format Counters
19/10/01 11:58:29 INFO mapred.JobClient: Bytes Written=1293
hduser@delloptiplex380-OptiPlex-380:~$ hadoop fs -ls
Warning: $HADOOP_HOME is deprecated.

```

Found 4 items

drwxr-xr-x	-	hduser	supergroup	0	2019-09-30 16:41	/user/hduser/Input
drwxr-xr-x	-	hduser	supergroup	0	2019-10-01 11:57	/user/hduser/Input1
drwxr-xr-x	-	hduser	supergroup	0	2019-09-30 16:44	/user/hduser/Output
drwxr-xr-x	-	hduser	supergroup	0	2019-10-01 11:58	/user/hduser/Output1

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop fs -ls Output1
```

Warning: \$HADOOP\_HOME is deprecated.

Found 3 items

-rw-r--r--	1	hduser	supergroup	0	2019-10-01 11:58	/user/hduser/Output1/_SUCCESS
drwxr-xr-x	-	hduser	supergroup	0	2019-10-01 11:58	/user/hduser/Output1/_logs
-rw-r--r--	1	hduser	supergroup	1293	2019-10-01 11:58	/user/hduser/Output1/part-r-00000

```
hduser@delloptiplex380-OptiPlex-380:~$ hadoop fs -cat Output/part-r-00000
```

Warning: \$HADOOP\_HOME is deprecated.

"ACCORDING"	1
"ACT"	1
"ADDITIONAL"	1
"AGAINST"	1
"ALL"	1
"ANCIENT"	1
"ANYBODY"	1
"ANYONE"	1
"APPLE"	1
"ARM"	1
"ASIDE"	1
"AWAY"	1
"BAND"	1
"BANK"	2
"BAT"	1
"BELL"	3
"BEST"	2

"BIGGEST"	1
"BIRDS"	2
"BIT"	3
"BLOOD"	1
"BOTH"	1
"BUILDING"	2
"BY"	5
"CALM"	2
"CANNOT"	1
"CAT"	1
"CELL"	1
"CHEMICAL"	3
"CHOSE"	1
"CIRCUS"	2
"CLAWS"	1
"CLOCK"	1
"COLLECT"	1
"CORRECT"	1
"COULD"	1
"COW"	1
"CREAM"	1
"DANCE"	3
"DETERMINE"	2
"DUST"	1
"EQUATOR"	1
"EVENT"	2
"EXPLANATION"	1
"FARTHER"	1
"FAST"	2
"FORWARD"	1
"FRIGHTEN"	1
"FROZEN"	2
"FUNNY"	1
"GARDEN"	2
"GENERAL"	1
"GENTLE"	1
"GENTLY"	1
"GIFT"	1
"GOLDEN"	2
"GREW"	1
"HELP"	1
"HOME"	1
"INTO"	4
"JOB"	2
"JOURNEY"	1
"KNEW"	1
"LABEL"	1
"LEADER"	3
"LONG"	2
"LOVE"	1
"MANNER"	2

"METAL"	1
"MOLECULAR"	1
"MOMENT"	1
"NATION"	1
"NEARBY"	2
"NORTH"	1
"NOTICE"	2
"ONLY"	1
"ORANGE"	1
"OVER"	2
"PAGE"	1
"PAID"	1
"PAIR"	1
"PALACE"	2
"PARTICULAR"	1
"PARTY"	1
"PENCIL"	1
"PERHAPS"	2
"PIG"	1
"PLEASANT"	1
"PLUS"	2
"POT"	3
"PREVENT"	1
"PROVIDE"	2
"RANGE"	1
"REAR"	1
"RIGHT"	1
"ROSE"	1
"SAID"	2
"SCIENTIFIC"	1
"SHAKING"	1
"SLIP"	1
"STEEP"	3
"STRIP"	1
"SUIT"	3
"SWEET"	1
"SYSTEM"	2
"THERE"	1
"THINK"	1
"TOGETHER"	1
"TRACK"	1
"TUNE"	2
"VOYAGE"	1
"WAGON"	3
"WALL"	1
"WARM"	1
"WEAR"	1
"WHEN"	1
"WHEREVER"	2
"WILLING"	1
"WITH"	2

"WON" 3  
"WORE" 2  
"WRAPPED" 1  
"WRITING" 1  
"YOU" 1  
hduser@delloptiplex380-OptiPlex-380:~\$  
"STONE" 2  
"STRIP" 1  
"SUIT" 3  
"SWEET" 1  
"SYSTEM" 2  
"THERE" 1  
"THINK" 1  
"TOGETHER" 1  
"TRACK" 1  
"TUNE" 2  
"VOYAGE" 1  
"WAGON" 3  
"WALL" 1  
"WARM" 1  
"WEAR" 1  
"WHEN" 1  
"WHEREVER" 2  
"WILLING" 1  
"WITH" 2  
"WON" 3  
"WORE" 2  
"WRAPPED" 1  
"WRITING" 1  
hduser@delloptiplex380-OptiPlex-380:~\$  
"YOU" 1  
hduser@delloptiplex380-OptiPlex-380:~\$

## HADOOP INSTALLATION

This big data hadoop tutorial will cover the pre-installation environment setup to install hadoop on Ubuntu and detail out the steps for hadoop single node setup so that you perform basic data analysis operations on HDFS and Hadoop MapReduce.

Apache Hadoop can be installed in 3 different modes of execution -

1. Standalone Mode - Single node hadoop cluster setup
2. Pseudo Distributed Mode - Single node hadoop cluster setup
3. Fully Distributed Mode - Multi-node hadoop cluster setup

### Hadoop Pre-installation Environment Setup

In this hadoop tutorial , we are using Ubuntu Server 12.04.5 LTS (64 bit). You can download it from this [link](#)

```
Ubuntu 12.04.5 LTS ubuntu tty1

ubuntu login: hadoop
Password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation: https://help.ubuntu.com/

System information as of Mon Jun  6 16:23:40 IST 2016

System load:  0.02      Processes:          357
Usage of /:   4.9% of 15.88GB  Users logged in:    0
Memory usage: 3%           IP address for eth0: 192.168.81.139
Swap usage:   0%

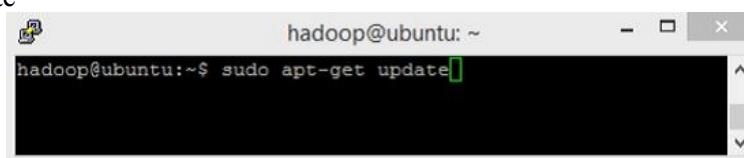
Graph this data and manage this system at:
https://landscape.canonical.com/

0 packages can be updated.
0 updates are security updates.
```

### Check for update or update the source index

Before you begin to install hadoop on Ubuntu , ensure that it is updated with the latest packages from all the repositories and PPA's. Execute the below command to see if there are any updates available-

```
$ sudo apt-get update
```

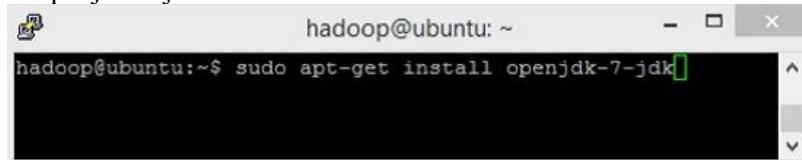


### Install Java

Java is the main pre-requisite software to run hadoop. To run hadoop on Ubuntu, you must have Java installed on your machine preferably Java version 1.6+ from Sun/Oracle or OpenJDK must be installed. You can check if Java is already installed on your machine using the below command-`java -version`.

Java™ 1.6.x or later, preferably from Oracle or Openjdk, must be installed. However, using Java 1.6+ is recommended for this hadoop tutorial. We are using Openjdk-7 to install Java software in this hadoop tutorial-

```
$ sudo apt-get install openjdk-7-jdk
```



You can check the java version installed on your machine by using the command - `java -version`

```
hadoop@ubuntu:~$ java -version
java version "1.7.0_101"
OpenJDK Runtime Environment (IcedTea 2.6.6) (7u101-2.6.6-0u
buntu0.12.04.1)
OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)
hadoop@ubuntu:~$
```

## Install SSH

SSH is required to manage the remote machines and your local machines before using hadoop on it. In this hadoop tutorial, we will use openssh server which can be installed as follows -

```
$ sudo apt-get install openssh-server
```

```
hadoop@ubuntu:~$ sudo apt-get install openssh-server
```

## Adding a dedicated Hadoop User Account

Creating a dedicated hadoop user helps separate HDFS from UNIX file system. We can begin by creating a “Hadoop” group as follows -

The next step is to create a hadoop user named hsuser and add it to the “hadoop” group created in the above step.

```
$ sudo addgroup hadoop
```

```
$ sudo adduser --ingroup hadoop hduser
```

On executing the above command, it will prompt you for the password and other details as shown below

```
hadoop@ubuntu:~$ sudo addgroup hadoop
[sudo] password for hadoop:
addgroup: The group 'hadoop' already exists.
hadoop@ubuntu:~$ sudo adduser --ingroup hadoop hduser
Adding user 'hduser' ...
Adding new user 'hduser' (1001) with group 'hadoop' ...
Creating home directory '/home/hduser' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
      Full Name []: hduser
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] Y
hadoop@ubuntu:~$
```

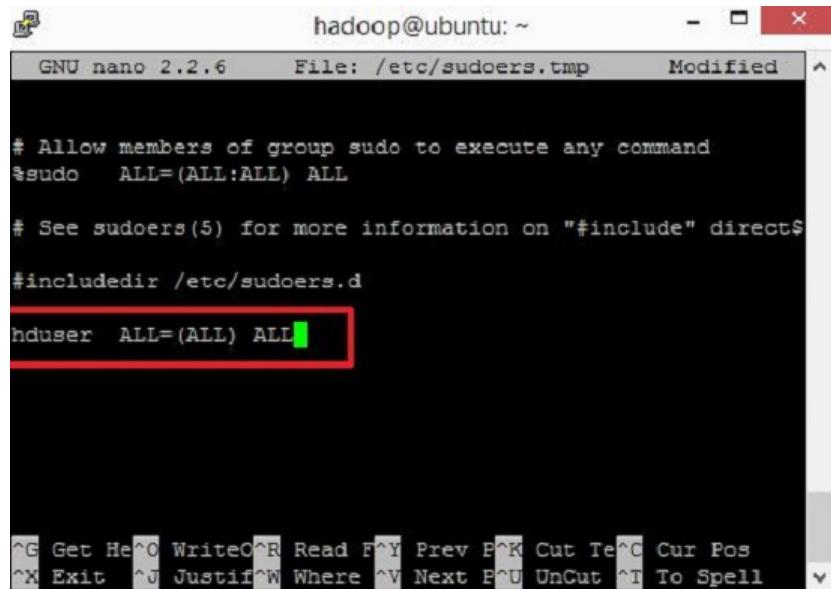
## Grant All Permissions to the Hadoop User “hduser created in the above Step

To grant all permissions to the created hadoop user, you must configure the sudoers file located at /etc/sudoers. However, this file cannot be configured directly and should be done using the visudo command as follows-

```
$ sudo visudo
```

Just type “o” to insert a line at the end with the command to grant all permissions to the hadoop user “hduser” as follows -

```
hduser ALL=(ALL) ALL
```



```
# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives

#include /etc/sudoers.d

hduser ALL=(ALL) ALL
```

Hit Escape to exit the insert mode of the editor and type “:x” to save the changes and exit from the file.

### Configuring SSH Access

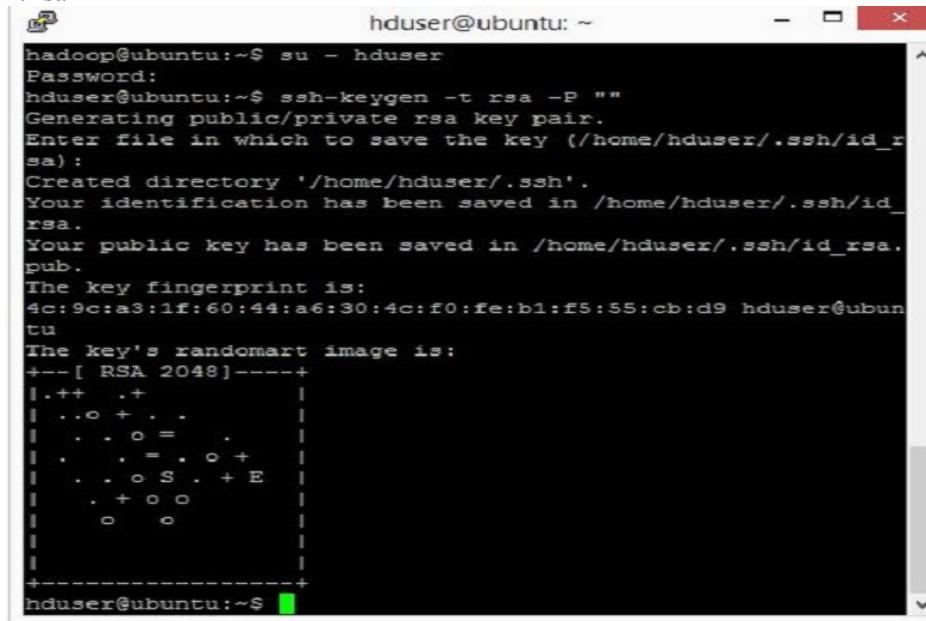
SSH Setup is needed to perform various operations on the hadoop cluster so that the master node can login to the slave nodes to start or stop them. SSH must be setup even on the secondary NameNode listed in the master’s file so that it can be started from the Namenode using the command ./start-dfs.sh and the job tracker node with ./start-mapred.sh.

To configure SSH access, you must login as hadoop user hduser -

```
$ su - hduser
```

The next step is to generate the SSH key for the hadoop user using the following command-

```
$ ssh-keygen -t rsa -P ""
```



```
hduser@ubuntu:~$ su - hduser
Password:
hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
4c:9c:a3:1f:60:44:a6:30:4c:f0:fe:b1:f5:55:cb:d9 hduser@ubuntu
The key's randomart image is:
+--[ RSA 2048]--+
| .++ .+
| ..o + ..
| ... o =
| . . = . o +
| . . o S . + E
| . + o o
| o o
|
+
```

In the above screenshot, the command **hduser@ubuntu:~\$ ssh-keygen -t rsa -P ""** command will create an empty password RSA key pair. It is not suggested to use an empty password, however, if you do not

always want to enter the passphrase whenever hadoop interacts with the nodes then you must give an empty password. This will ensure that hadoop interacts with the nodes without your interaction.

The next step is to enable SSH access with the key generated in the previous step -

```
$ cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
```

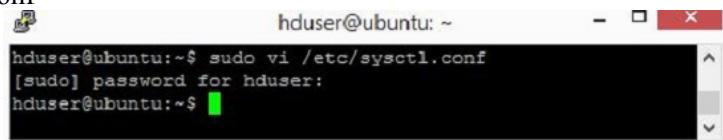
### Disabling IPv6

As Hadoop does not support IPv6 and is tested to work only IPv4 network, it is suggested to disable IPv6.

However, if you are not using IPv6, you can simply skip this step of the hadoop installation process.

To disable IPv6, you need open the file /etc/sysctl.conf using the vi editor as shown below -

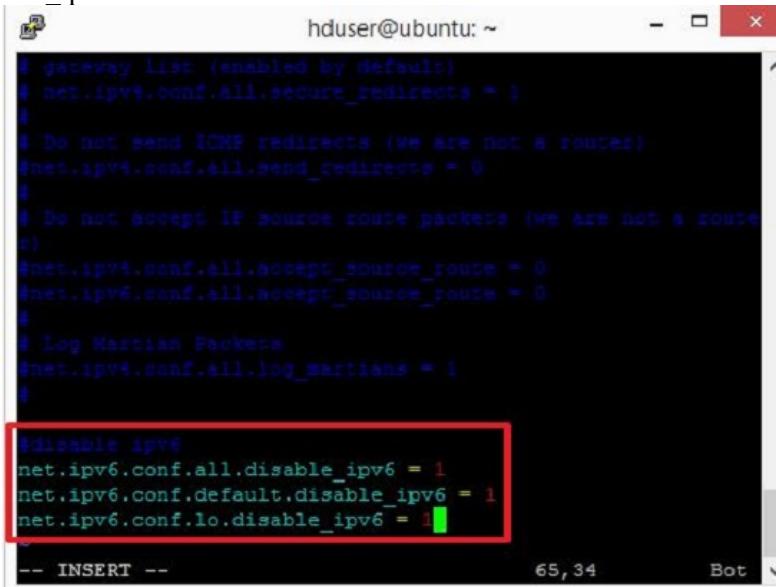
```
$ sudo vi /etc/sysctl.conf
```



```
hduser@ubuntu:~$ sudo vi /etc/sysctl.conf
[sudo] password for hduser:
hduser@ubuntu:~$
```

Copy the below lines of code to disable IPv6 -

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```



```
net.ipv4.conf.all.secure_redirects = 1
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.log_martians = 1

# disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

To ensure that IPv6 has been disabled , run the following command -

```
$cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

Having disabled IPv6, it is suggested that you power off the machine and restart it again using the below command -

```
$ sudo reboot now
```

Hurray, you have completed the environment setup to install hadoop. Now, let's get started with Hadoop installation in standalone mode.

### Hadoop Single Node Setup- Standalone Mode

Hadoop on a single node in standalone mode runs as a single java process. This mode of execution is of great help for debugging purpose. This mode of execution helps you run your MapReduce application on small data before you start running it on a hadoop cluster with big data.

### Download Hadoop

Hadoop can be downloaded using the “wget” command as shown below -

```
$ wget https://archive.apache.org/dist/hadoop/core/hadoop-1.2.1/hadoop-1.2.1.tar.gz
```

```
hduser@ubuntu:~$ wget https://archive.apache.org/dist/hadoop/core/hadoop-1.2.1/hadoop-1.2.1.tar.gz
--2016-06-07 07:37:48-- https://archive.apache.org/dist/hadoop/core/hadoop-1.2.1/hadoop-1.2.1.tar.gz
Resolving archive.apache.org (archive.apache.org)... 163.172.17.199
Connecting to archive.apache.org (archive.apache.org)|163.172.17.199|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 63851630 (61M) [application/x-gzip]
Saving to: `hadoop-1.2.1.tar.gz'

100%[=====] 6,38,51,630 313K/s in 2m 42s

2016-06-07 07:40:31 (385 KB/s) - `hadoop-1.2.1.tar.gz' saved [63851630/63851630]

hduser@ubuntu:~$ ls /home/hduser/
hadoop-1.2.1.tar.gz
hduser@ubuntu:~$
```

The compressed hadoop file needs to be unzipped as follows -

```
$ tar -xvf /home/hduser/hadoop-1.2.1.tar.gz
```

### Verify if Hadoop is installed

To confirm that hadoop has been installed, you can run the following command -

```
$ ls /home/hduser/
```

```
hduser@ubuntu:~$ ls /home/hduser/
hadoop-1.2.1  hadoop-1.2.1.tar.gz
hduser@ubuntu:~$
```

Listing the contents of /home/hduser/ shows that hadoop has been installed. You can now move the contents of the directory to the location of your choice. Let's move hadoop directory to /usr/local/

```
$ sudo mv /home/hduser/hadoop-1.2.1 /usr/local/hadoop
```

### Assign Ownership of Hadoop to hduser

Ensure that you change the ownership of all files to "hduser" and the "hadoop" group.

```
$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

```
hduser@ubuntu:~$ sudo chown -R hduser:hadoop /usr/local/hadoop
hduser@ubuntu:~$ ls -lrt /usr/local/
total 36
drwxr-xr-x 15 hduser hadoop 4096 Jul 23 2013 hadoop
drwxr-xr-x  2 root   root  4096 Jun  6 15:58 bin
drwxr-xr-x  2 root   root  4096 Jun  6 15:58 games
drwxr-xr-x  2 root   root  4096 Jun  6 15:58 include
drwxr-xr-x  2 root   root  4096 Jun  6 15:58 sbin
drwxr-xr-x  2 root   root  4096 Jun  6 15:58 src
drwxr-xr-x  2 root   root  4096 Jun  6 15:58 etc
lrwxrwxrwx  1 root   root    9 Jun  6 15:58 man -> share/man
drwxr-xr-x  3 root   root  4096 Jun  6 15:59 lib
drwxr-xr-x  7 root   root  4096 Jun  6 23:58 share
hduser@ubuntu:~$
```

### Hadoop Configuration

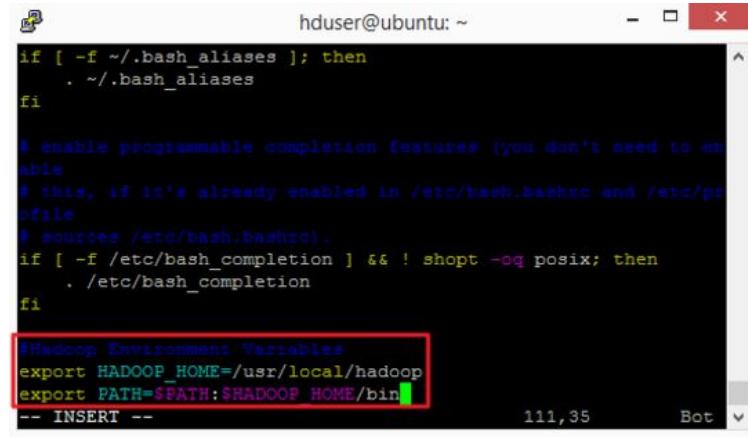
Before hadoop is up and running, you need to configure the hadoop environment.

**Update .bashrc** - We are updating .bashrc with the following Hadoop environment variables:

```
# Set HADOOP_HOME
export HADOOP_HOME=/home/hduser/hadoop
# Add Hadoop bin and sbin directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin;$HADOOP_HOME/sbin
$ sudo nano /home/hduser/.bashrc
```

Enter the following lines at the end of the file:

```
#Hadoop Environment Variables
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```



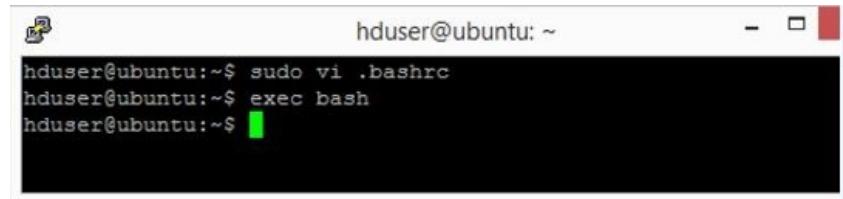
```
hduser@ubuntu: ~
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile.d/
# sources /etc/bash_completion).
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
    . /etc/bash_completion
fi

#Hadoop Environment Variables
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

Enter the following command to update bashrc:

```
$ exec bash
```



```
hduser@ubuntu: ~
hduser@ubuntu:~$ sudo vi .bashrc
hduser@ubuntu:~$ exec bash
hduser@ubuntu:~$
```

Java is a pre-requisite for hadoop to run, so you need to inform hadoop where java is installed by setting the variable JAVA\_HOME in the hadoop-env.sh file.

```
$ sudo nano /usr/local/hadoop/conf/hadoop-env.sh
```

Enter the following lines at the end of the file:

```
#JAVA HOME variable
```

```
Export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

**Note:** If you are trying to install hadoop with Ubuntu server or Ubuntu Desktop (32-bit) then your JAVA\_HOME would be - JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-i386

```

# the potential for a symlink attack.
# export HADOOP_PID_DIR=/var/hadoop/pids

# A string representing this instance of hadoop. $USER by default.
# export HADOOP_IDENT_STRING=$USER

# The scheduling priority for daemon processes. See 'man nice'.
# export HADOOP_NICENESS=10

# JAVA_HOME variable
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

```

Confirm the successful installation of Hadoop in standalone mode

```
$ hadoop jar /usr/local/hadoop/hadoop-examples-1.2.1.jar wordcount /usr/local/hadoop/README.txt /home/hduser/Output
```

```

hduser@ubuntu:~$ hadoop jar /usr/local/hadoop/hadoop-examples-1.2.1.jar wordcount /usr/local/hadoop/README.txt /home/hduser/Output

```

### Hadoop Single Node Setup - Pseudo Distributed Mode

Hadoop is installed on a single machine in this mode of execution also just like standalone mode but in this all the daemons run as separate Java processes i.e. NameNode, DataNode, JobTracker, TaskTracker, Secondary NameNode all run on a single machine.

#### Create data directory for Hadoop - HDFS

Create a data folder HDFS using mkdir and assign all permissions. A hadoop user will have to read or write to these directories , thus it is necessary to change the permissions of the above directories for the corresponding hadoop user.

```
$ mkdir /usr/local/hadoop/hdfs
```

```

hduser@ubuntu:~$ mkdir /usr/local/hadoop/hdfs
hduser@ubuntu:~$ sudo chown -R hduser:hadoop /usr/local/hadoop/hdfs
[sudo] password for hduser:
hduser@ubuntu:~$

```

Hadoop configuration files are present in the HADOOP\_HOME/conf dir in this tutorial the path is /usr/local/hadoop/conf/.

#### Configuring core-site.xml

This XML file contains common properties to HDFS, MapReduce, YARN . Hadoop provides default configuration for these properties in the core-default.xml file.The default properties and their values can be found on the following Github link - <https://github.com/facebookarchive/hadoop-20/blob/master/src/core/core-default.xml>.

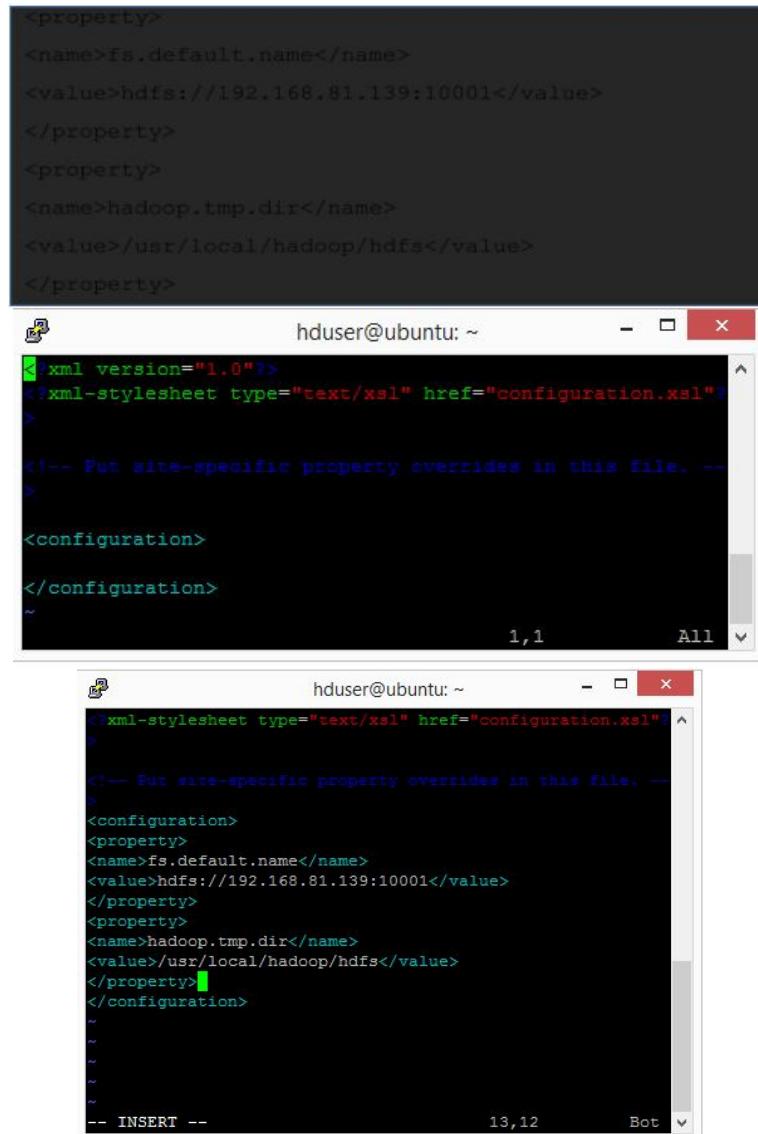
Open the core-site.xml file using the vi editor -

```
$ sudo nano /usr/local/hadoop/conf/core-site.xml
```

Enter the following lines between the configuration tab:

```
fs.default.name
```

```
hdbs://192.168.81.139:10001  
hadoop.tmp.dir  
/usr/local/hadoop/hdfs
```



```
<property>  
  <name>fs.default.name</name>  
  <value>hdbs://192.168.81.139:10001</value>  
</property>  
  
<property>  
  <name>hadoop.tmp.dir</name>  
  <value>/usr/local/hadoop/hdfs</value>  
</property>
```

```
<?xml version="1.0"?>  
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  
>  
  
<!-- Put site-specific property overrides in this file. -->  
>  
  
<configuration>  
  </configuration>  
~
```

```
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  
>  
  
<!-- Put site-specific property overrides in this file. -->  
>  
  
<configuration>  
  <property>  
    <name>fs.default.name</name>  
    <value>hdbs://192.168.81.139:10001</value>  
  </property>  
  <property>  
    <name>hadoop.tmp.dir</name>  
    <value>/usr/local/hadoop/hdfs</value>  
  </property>  
</configuration>  
~  
~  
~  
~  
~  
~  
-- INSERT --
```

Parameter	Value	Notes
fs.default.name	URI of NameNode.	<i>hdbs://hostname:port</i>
hadoop.tmp.dir	Path to hdfs	<i>used as the base for temporary directories locally</i>

### Configuring mapred-site.xml

It contains mapreduce override properties. The default properties and their values here:

<https://github.com/facebookarchive/hadoop-20/blob/master/src/mapred/mapred-default.xml>.

Open the mapred-site.xml using the vi editor -

\$ sudo nano /usr/local/hadoop/conf/mapred-site.xml

Enter the below code in the configuration tab-

mapred.job.tracker  
hdfs://192.168.81.139:10002

The image shows two terminal windows side-by-side. Both windows have a title bar 'hduser@ubuntu: ~'. The top window displays the contents of a file named 'mapred-site.xml' with the following XML:

```
<property>
  <name>mapred.job.tracker</name>
  <value>hdfs://192.168.81.139:10002</value>
</property>
```

The bottom window displays the contents of a file named 'core-site.xml' with the following XML:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->
<configuration>
</configuration>
~
```

Both windows show the status bar at the bottom indicating the current line (e.g., 1,1, 5,16) and the word count (e.g., All).

Parameter	Value	Notes
mapred.job.tracker	Host or IP and port of JobTracker.	<i>hdfs://host:port</i> pair.

### Configuring the masters

It contains IP's or hostname of all secondary NameNode's or checkpoint servers, one per line. Execute the following command to edit masters:

\$ sudo nano /usr/local/hadoop/conf/masters

Enter your checkpoint server or system IP for Pseudo-mode-  
192.168.81.139(your IP)

The image shows two terminal windows side-by-side. Both windows have a title bar 'hduser@ubuntu: ~'. The top window displays the contents of a file named 'masters' with the following text:

```
localhost
```

The bottom window displays the contents of the same 'masters' file with the following text:

```
192.168.81.139
```

Both windows show the status bar at the bottom indicating the current line (e.g., 1,10, 1,15) and the word count (e.g., All).

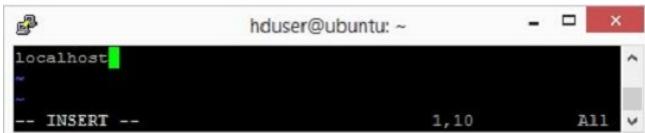
### Configuring the Slaves

It contains IP's or hostname of all datanodes, one per line. Execute the following commands to edit the slaves :

```
$ sudo nano /usr/local/hadoop/conf/slaves
```

Enter your datanode IP or system IP for pseudo-mode:

192.168.81.139(your IP)



### Configuring hdfs-site.xml

It contains HDFS override properties. The default properties and their values for hdfs-site. Can be found here: <https://github.com/facebookarchive/hadoop-20/blob/master/src/hdfs/hdfs-default.xml>.

Execute the following commands to edit the hdfs-site.xml :

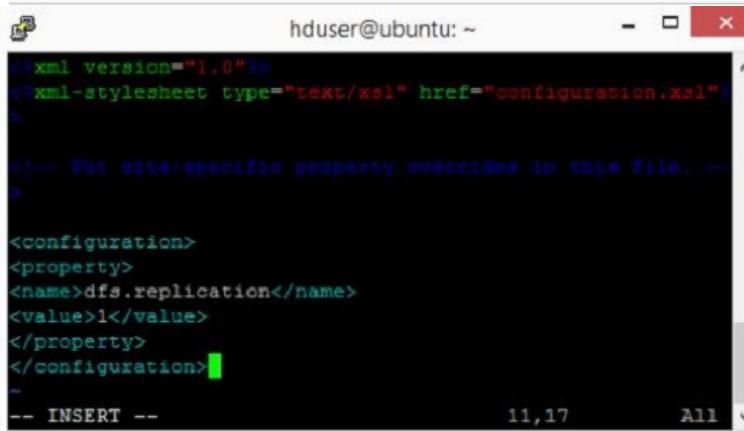
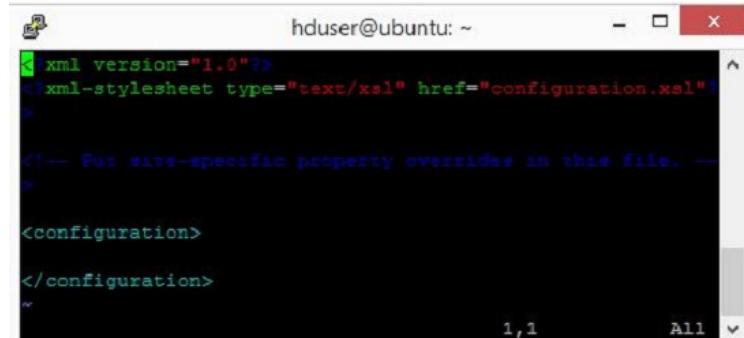
```
$ sudo nano /usr/local/hadoop/conf/hdfs-site.xml
```

Enter the following lines between tab:

```
dfs.replication
```

```
1
```

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



Parameter	Value	Notes
dfs-replication	Value in positive integer	No. of replicate/duplicate block

Here is the formula to calculate replication:

Replication factor = No. of datanodes or less than the no. of datanode (depends on probability of failure of nodes)

### Formatting the NameNode

The foremost step to get hadoop up and running is to format the hadoop distributed file system (HDFS) of your hadoop cluster. NameNode should be formatted when hadoop cluster is setup for the first time. If you format an already running HDFS, you will lose all the data that is present in HDFS for the cluster.

Execute the following command to format the NameNode:

```
$ hadoop namenode -format
```

The screenshot shows two terminal windows. The top window displays the command `hadoop namenode -format`. The bottom window shows the output of this command, which includes log entries about edit logs being closed and truncated, the storage directory being successfully formatted, and the NameNode shutting down. A green checkmark icon is overlaid on the bottom window's terminal area, indicating success.

```
hduser@ubuntu:~$ hadoop namenode -format
g: position=4, editlog=/usr/local/hadoop/hdfs/dfs/name/current/edits
16/06/10 06:13:52 INFO namenode.FSEditLog: close success: truncate to 4, editlog=/usr/local/hadoop/hdfs/dfs/name/current/edits
16/06/10 06:13:52 INFO common.Storage: Storage directory /usr/local/hadoop/hdfs/dfs/name has been successfully formatted.
16/06/10 06:13:52 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
*** SHUTDOWN_MSG: Shutting down NameNode at ubuntu/127.0.1.1 ****
**/
```

### Starting the Hadoop Cluster

Start the Hadoop server by executing the following command:

```
$ start-all.sh
```

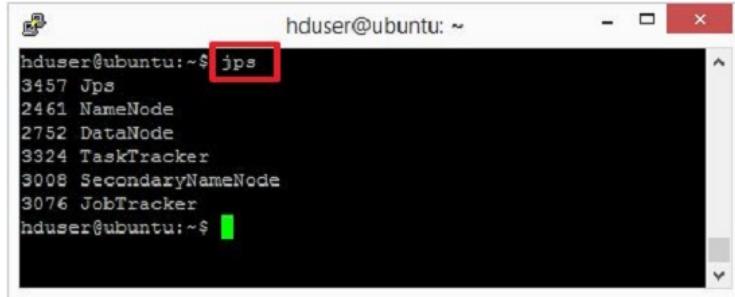
The screenshot shows a terminal window executing the command `start-all.sh`. The output logs the starting of various Hadoop services: Namenode, Datanode, Secondary Namenode, Jobtracker, and Tasktracker. It also shows a warning message about the deprecation of the `HADOOP_HOME` environment variable. A red box highlights the command `start-all.sh`.

```
hduser@ubuntu:~$ start-all.sh
warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-namenode-ubuntu.out
The authenticity of host '192.168.81.139 (192.168.81.139)' can't be established.
ECDSA key fingerprint is 6a:b7:66:3f:de:bf:74:0d:69:8a:d0:c7:13:36:9f:17.
Are you sure you want to continue connecting (yes/no)? yes
192.168.81.139: Warning: Permanently added '192.168.81.139' (ECDSA) to the list of known hosts.
hduser@192.168.81.139's password:
192.168.81.139: starting datanode, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-datanode-ubuntu.out
hduser@192.168.81.139's password:
192.168.81.139: starting secondarynamenode, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-secondarynamenode-ubuntu.out
starting jobtracker, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-jobtracker-ubuntu.out
hduser@192.168.81.139's password:
192.168.81.139: starting tasktracker, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-tasktracker-ubuntu.out
hduser@ubuntu:~$
```

To verify if all the services are up and running, execute the below command-

```
$ jps
```

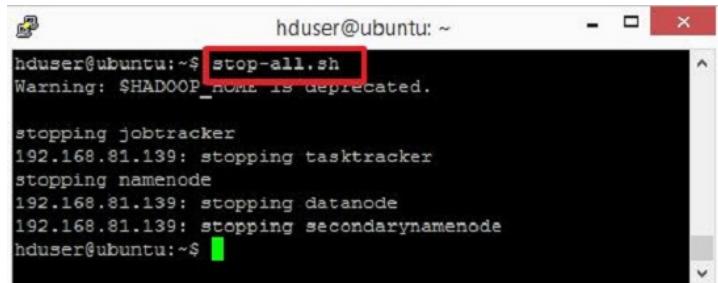


```
hduser@ubuntu:~$ jps
3457 Jps
2461 NameNode
2752 DataNode
3324 TaskTracker
3008 SecondaryNameNode
3076 JobTracker
hduser@ubuntu:~$
```

## Stop the hadoop servers

Stop the Hadoop server by executing the following command:

```
$ stop-all.sh
```



```
hduser@ubuntu:~$ stop-all.sh
Warning: $HADOOP_HOME is deprecated.

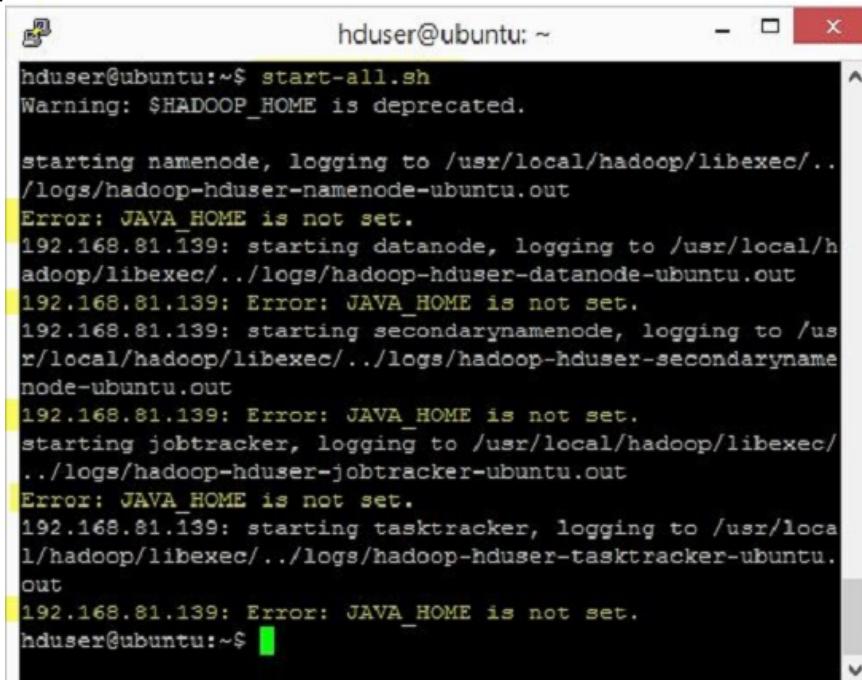
stopping jobtracker
192.168.81.139: stopping tasktracker
stopping namenode
192.168.81.139: stopping datanode
192.168.81.139: stopping secondarynamenode
hduser@ubuntu:~$
```

## Troubleshooting

Apache Hadoop uses \$HADOOP\_HOME/logs directory to maintain all the error logs so whenever you face any issues while installing hadoop on ubuntu then look at the log files.

### Common Errors Encountered during Hadoop Installation

#### Error :JAVA\_HOME is not set.



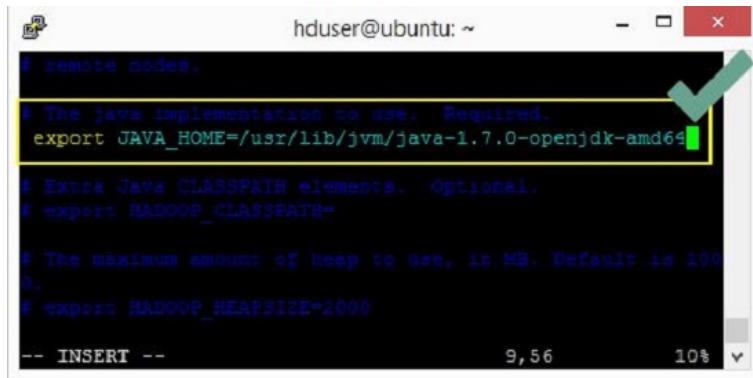
```
hduser@ubuntu:~$ start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/...
/logs/hadoop-hduser-namenode-ubuntu.out
Error: JAVA_HOME is not set.
192.168.81.139: starting datanode, logging to /usr/local/hadoop/libexec/...
/logs/hadoop-hduser-datanode-ubuntu.out
192.168.81.139: Error: JAVA_HOME is not set.
192.168.81.139: starting secondarynamenode, logging to /usr/local/hadoop/libexec/...
/logs/hadoop-hduser-secondaryname-
node-ubuntu.out
192.168.81.139: Error: JAVA_HOME is not set.
starting jobtracker, logging to /usr/local/hadoop/libexec/...
/logs/hadoop-hduser-jobtracker-ubuntu.out
Error: JAVA_HOME is not set.
192.168.81.139: starting tasktracker, logging to /usr/local/hadoop/libexec/...
/logs/hadoop-hduser-tasktracker-ubuntu.out
192.168.81.139: Error: JAVA_HOME is not set.
hduser@ubuntu:~$
```

Solution to JAVA\_HOME is not set error -

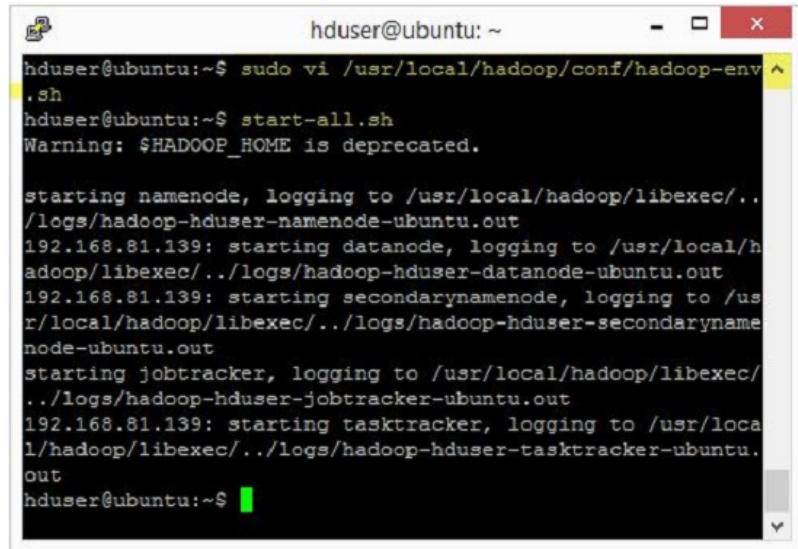
Open the hadoop-env.sh file located in HADOOP\_HOME/conf/ using vi editor and set the Java home path-

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



```
# remote nodes.  
# The java implementation to use. Required.  
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64  
  
# Export Java CLASSPATH elements. Optional.  
# export HADOOP_CLASSPATH=  
  
# The maximum amount of heap to use, in MB. Default is 1000.  
#  
# export HADOOP_HEAPSIZE=1000  
  
-- INSERT --
```

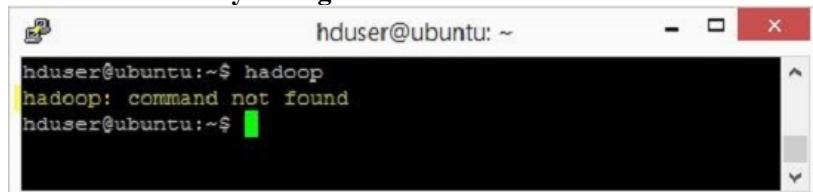
Just to ensure that the error is resolved, try starting the hadoop server again using the hadoop command start-all.sh.



```
hduser@ubuntu:~$ sudo vi /usr/local/hadoop/conf/hadoop-env.sh  
hduser@ubuntu:~$ start-all.sh  
Warning: $HADOOP_HOME is deprecated.  
  
starting namenode, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-namenode-ubuntu.out  
192.168.81.139: starting datanode, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-datanode-ubuntu.out  
192.168.81.139: starting secondarynamenode, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-secondaryname-node-ubuntu.out  
starting jobtracker, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-jobtracker-ubuntu.out  
192.168.81.139: starting tasktracker, logging to /usr/local/hadoop/libexec/..../logs/hadoop-hduser-tasktracker-ubuntu.out  
hduser@ubuntu:~$
```

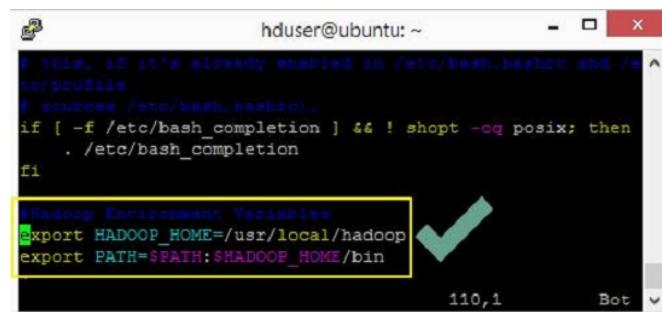
**Error: hadoop: command not found**

Here's a snapshot of the error that you might encounter -



```
hduser@ubuntu:~$ hadoop  
hadoop: command not found  
hduser@ubuntu:~$
```

To resolve this issue, open the .bachrc file using vi editor and define the path for HADOOP\_HOME/bin - \$ sudo vi /home/hduser/.bashrc



```
# this, if it's already defined in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc  
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then  
    . /etc/bash_completion  
fi  
  
#Hadoop Environment Variables  
export HADOOP_HOME=/usr/local/hadoop  
export PATH=$PATH:$HADOOP_HOME/bin
```

Execute the below command to update the .bashrc -

```
hduser@ubuntu:~$ sudo vi /home/hduser/.bashrc
[sudo] password for hduser:
hduser@ubuntu:~$ exec bash
hduser@ubuntu:~$
```

**Error: ls cannot access .: No such file or directory.**

```
hduser@ubuntu:~$ hadoop fs -ls
Warning: $HADOOP_HOME is deprecated.

ls: Cannot access .: No such file or directory.
hduser@ubuntu:~$
```

When you do not mention the path after the ls command, it takes the default path ./user/. To resolve this issue, create a folder under user/ i.e. in this tutorial it is user/hduser.

**\$ hadoop fs -mkdir /user/hduse**

```
hduser@ubuntu:~$ hadoop fs -mkdir /user/hduser
Warning: $HADOOP_HOME is deprecated.

hduser@ubuntu:~$ hadoop fs -ls
Warning: $HADOOP_HOME is deprecated.

hduser@ubuntu:~$
```

**Error : Some index files failed to download or old index file used.**

```
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_multiverse_source_Sources Hash Sum mismatch
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_main_binary-amd64_Packages Hash Sum mismatch
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_restricted_binary-amd64_Packages Hash Sum mismatch
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_multiverse_binary-amd64_Packages Hash Sum mismatch
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_main_binary-i386_Packages Hash Sum mismatch
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_restricted_binary-i386_Packages Hash Sum mismatch
W: Failed to fetch bzr2:/var/lib/apt/lists/partial/in.archive.ubuntu.com_ubuntu_dists_precise_multiverse_binary-i386_Packages Hash Sum mismatch
E: Some index files failed to download. They have been ignored, or old ones used instead.
hadoop@ubuntu:~$
```

This error can be resolved by removing the old index files by running the command- **\$ sudo rm -r /var/lib/apt/lists/\***

```
hduser@ubuntu:~$ sudo rm -r /var/lib/apt/lists/*
hduser@ubuntu:~$
```

If you encounter any “Incompatible NamespaceID’s” exception then to trouble shoot such error you have to do the following -

- Stop all the services
- Delete /tmp/hadoop/dfs/data
- Start all the services again.