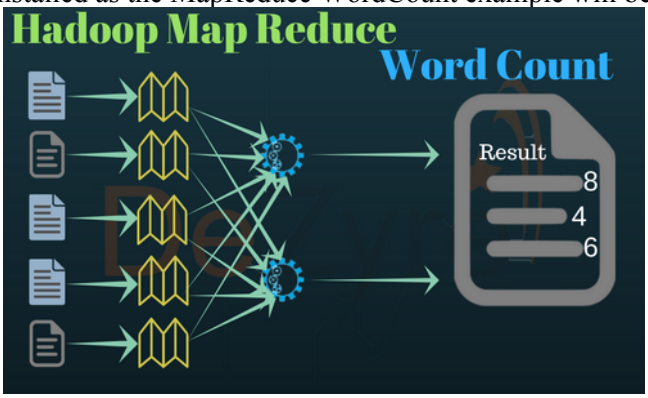
Pre-requisites to follow this Hadoop WordCount Example Tutorial

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)

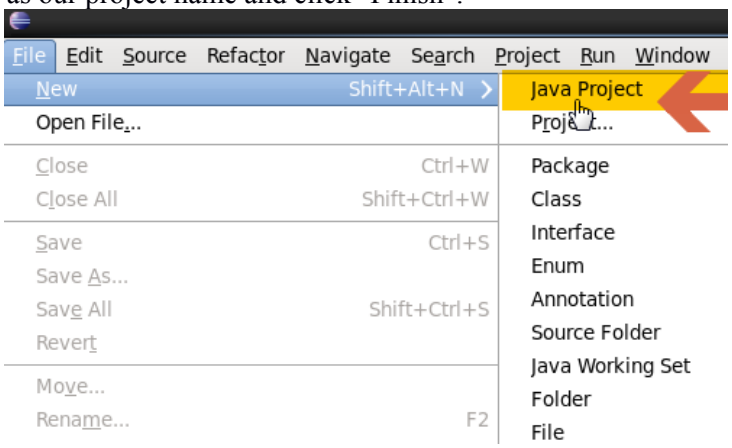
This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file. An important point to note during the execution of the WordCount example is that the mapper class in the WordCount program will execute completely on the entire input file and 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.

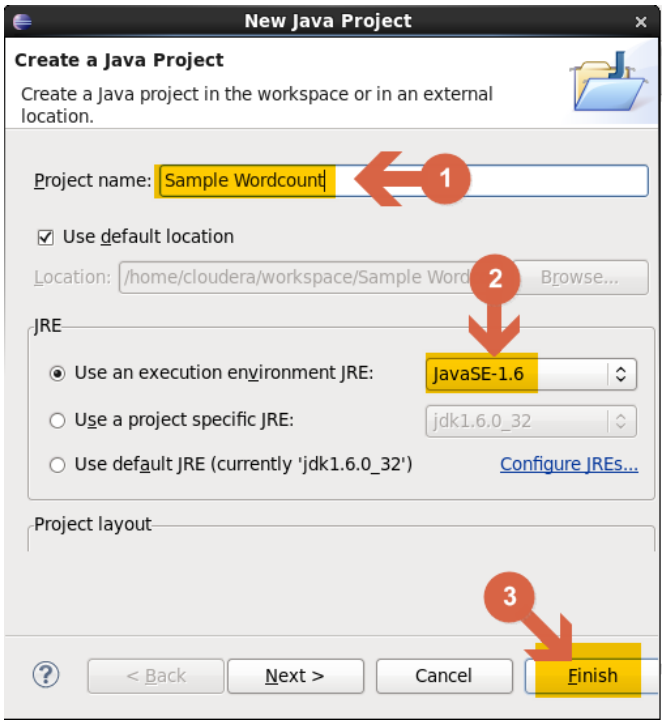
Running the WordCount Example in Hadoop MapReduce using Java Project with Eclipse

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 -

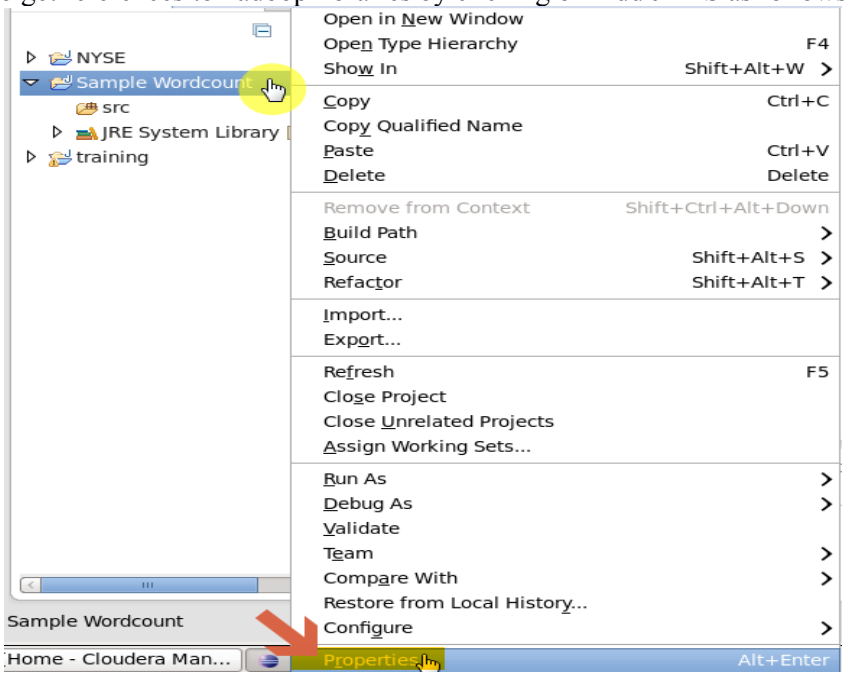
Let's create the java project with the name "Sample WordCount" as shown below -
File > New > Project > Java Project > Next.
"Sample WordCount" as our project name and click "Finish":

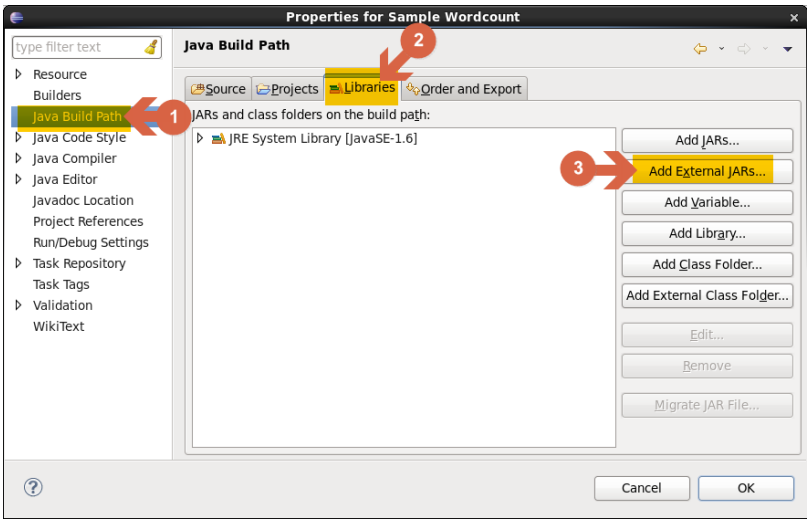




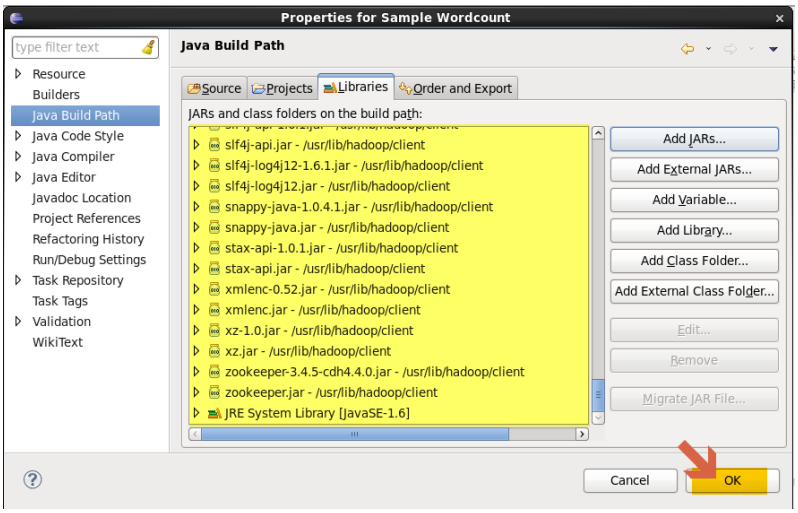
Step 2 -

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



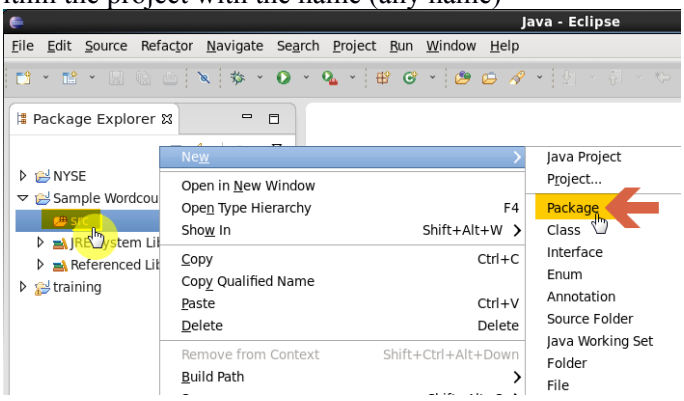


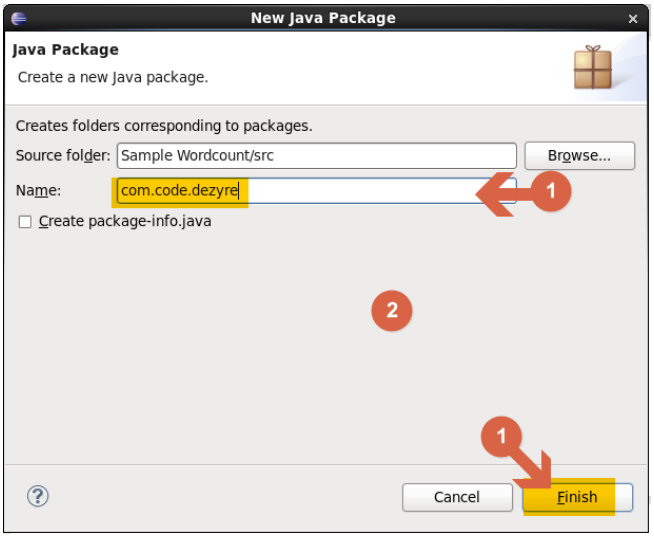
Go to the folder where hadoop is installed and add the jars.



Step 3 -

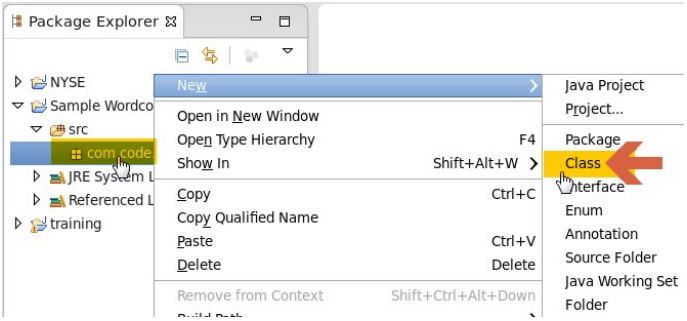
Create a new package within the project with the name (any name)

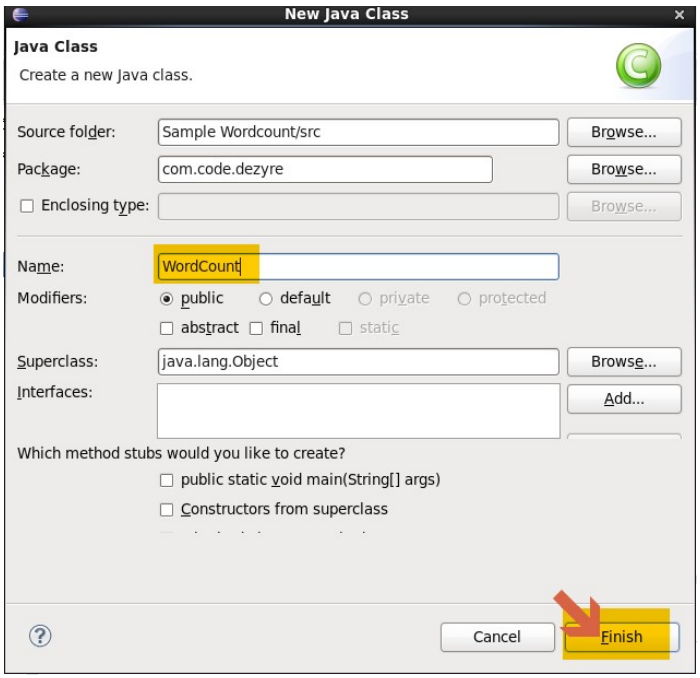




Step 4 -

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





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 -

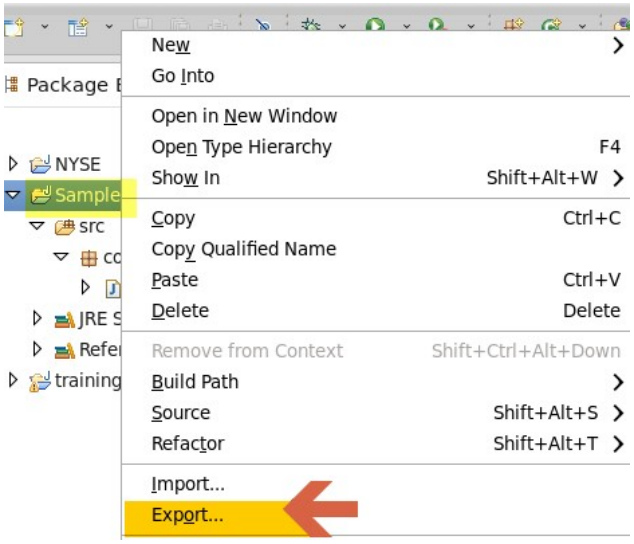
i. OutputKeyClass
ii. OutputValueClass
iii. Mapper Class
iv. Reducer Class
v. InputFormat
vi. OutputFormat
vii. InputFilePath
viii. OutputFolderPath

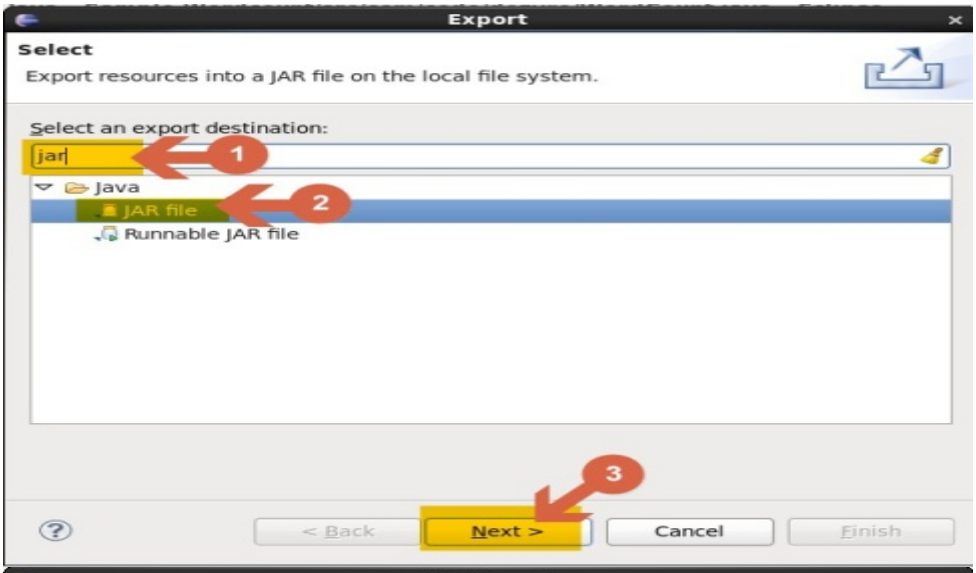
```
        public static void main(String[] args) throws Exception {
```

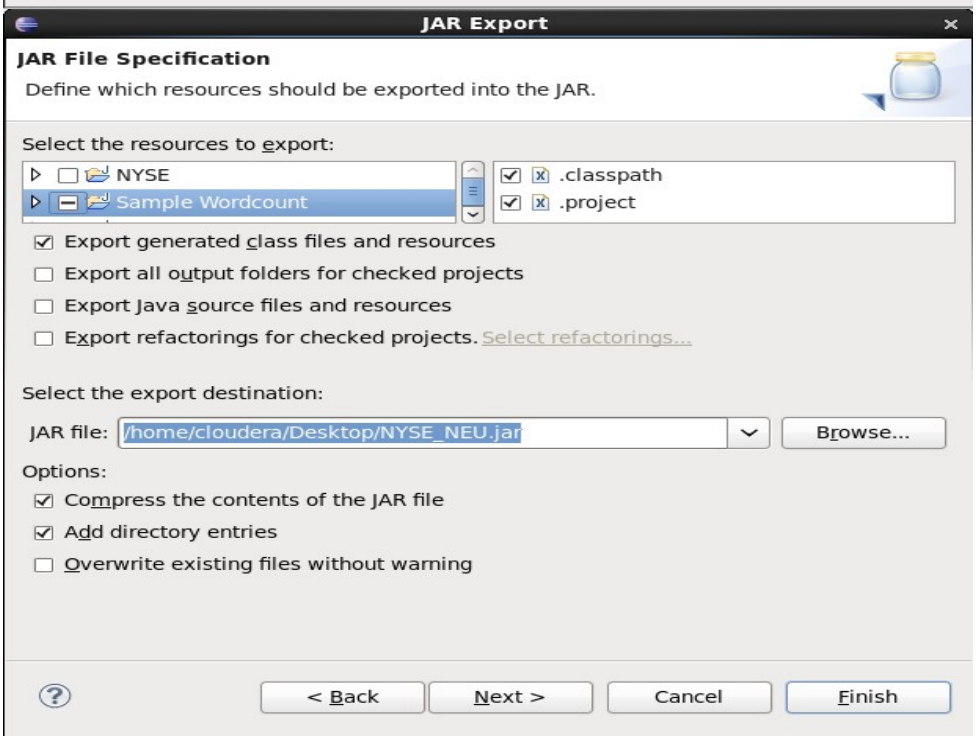
```
        JobConf conf = new JobConf(WordCount.class);
        conf.setJobName("WordCount");

        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);
    }
}
```

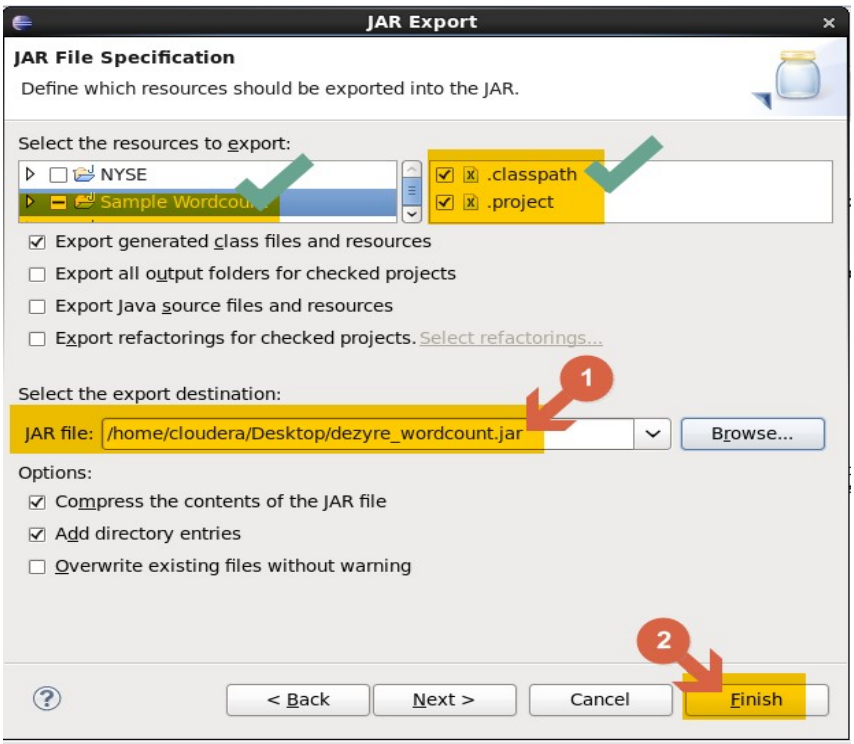
Step 8 -

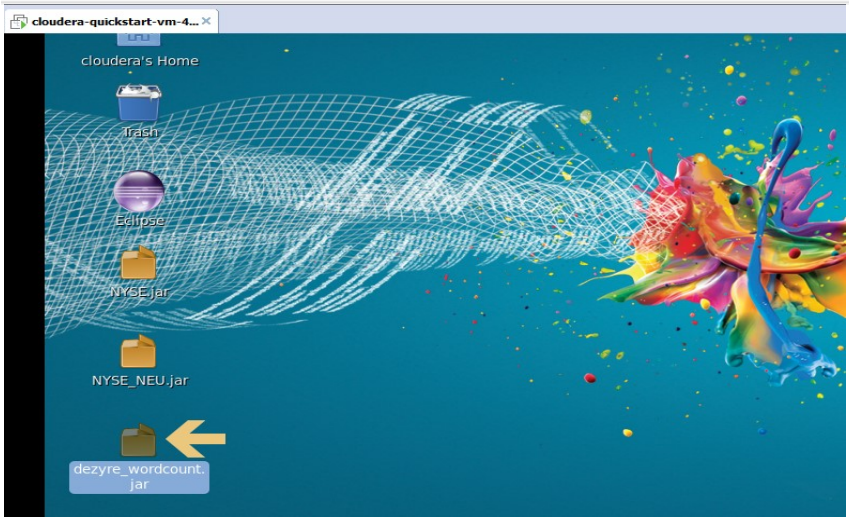
Create the JAR file for the wordcount class -



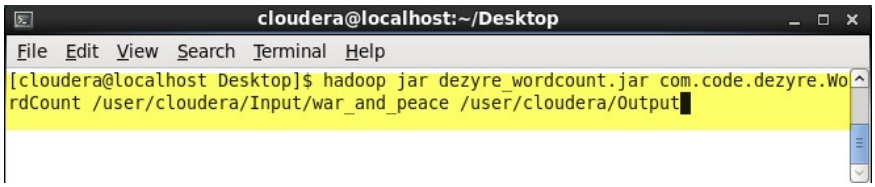

Export
Select
Export resources into a JAR file on the local file system.
Select an export destination:
jar
1
Java
JAR file
2
Runnable JAR file
3
< Back
Next >
Cancel
Finish


JAR Export
JAR File Specification
Define which resources should be exported into the JAR.
Select the resources to export:
NYSE
Sample Wordcount
.classpath
.project
Export generated class files and resources
Export all output folders for checked projects
Export Java source files and resources
Export refactorings for checked projects. Select refactorings...
Select the export destination:
JAR file: /home/cloudera/Desktop/NYSE_NEU.jar
Browse...
Options:
Compress the contents of the JAR file
Add directory entries
Overwrite existing files without warning
< Back
Next >
Cancel
Finish

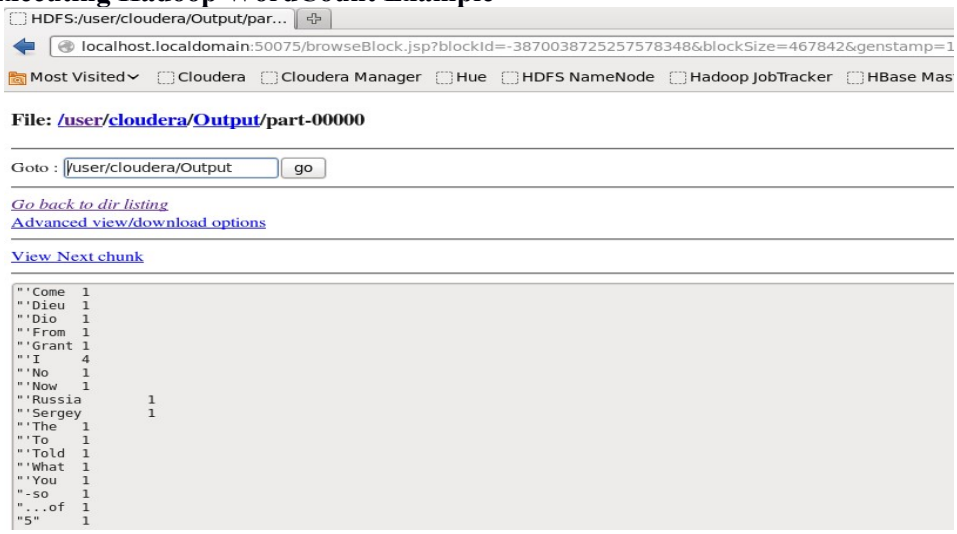




How to execute the Hadoop MapReduce WordCount program ?



Output of Executing Hadoop WordCount Example -



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

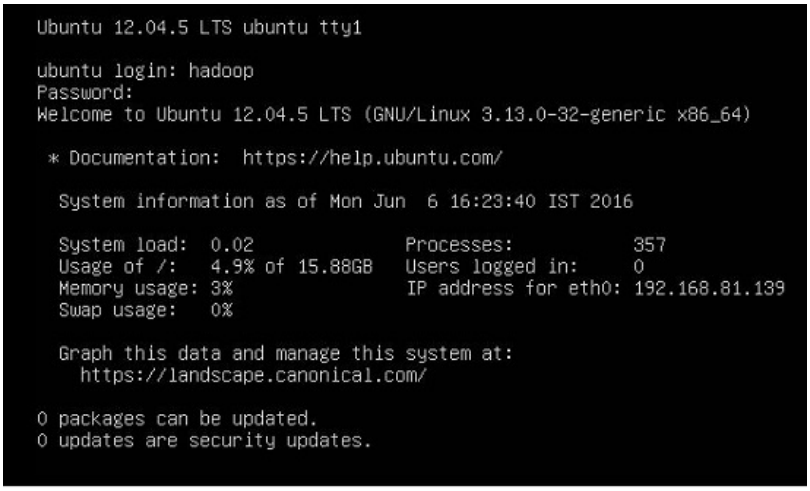
This big data hadoop tutorial will cover the pre-installation environment setup to install hadoop on Ubuntu and detail out the steps for hadoop single node setup so that you perform basic data analysis operations on HDFS and Hadoop MapReduce.

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

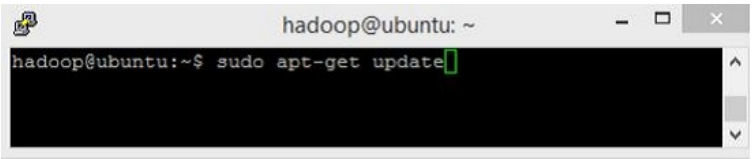
In this hadoop tutorial , we are using Ubuntu Server 12.04.5 LTS (64 bit). You can download it from this link



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-

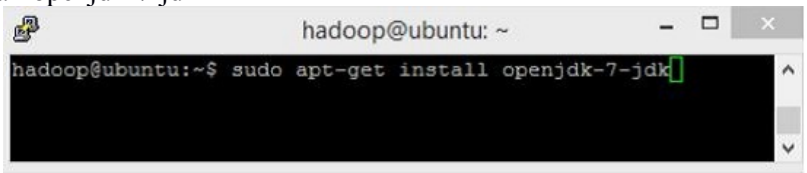
$ sudo apt -get update



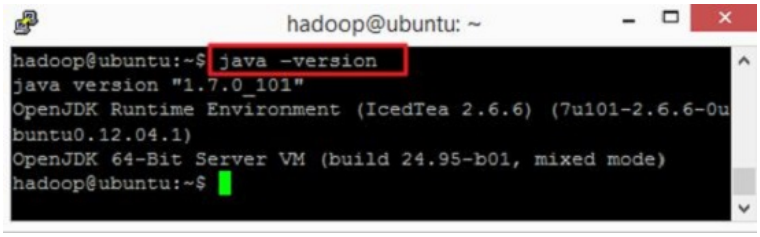
Install Java

Java is the main pre-requsiite 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 installe don 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



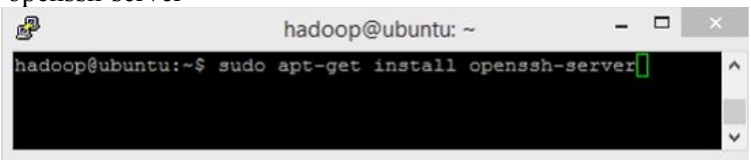
You can check the java version installed on your machine by using the command - java -version



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



Adding a dedicated Hadoop User Account

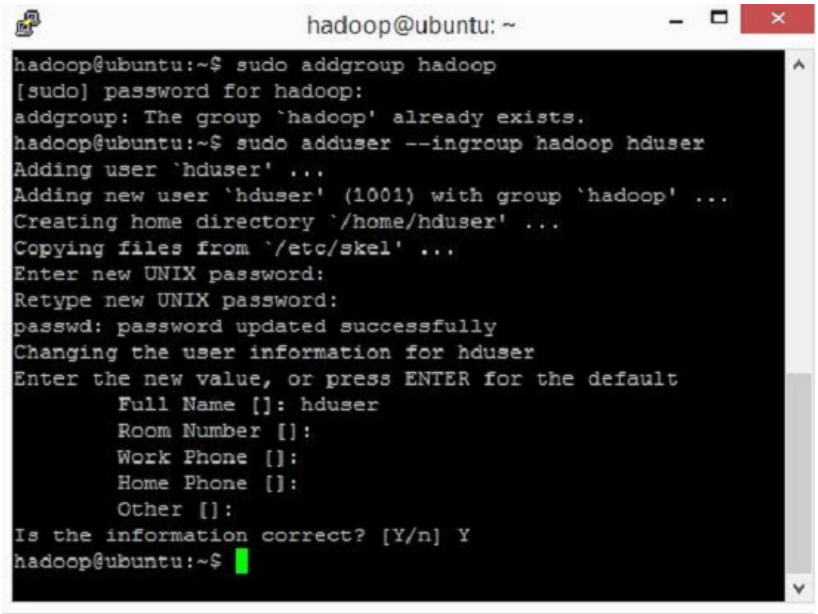
Creating a dedicated hadoop user helps separate HDFS from UNIX file system. We can begin by creating a "Hadoop" group as follows -

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 -



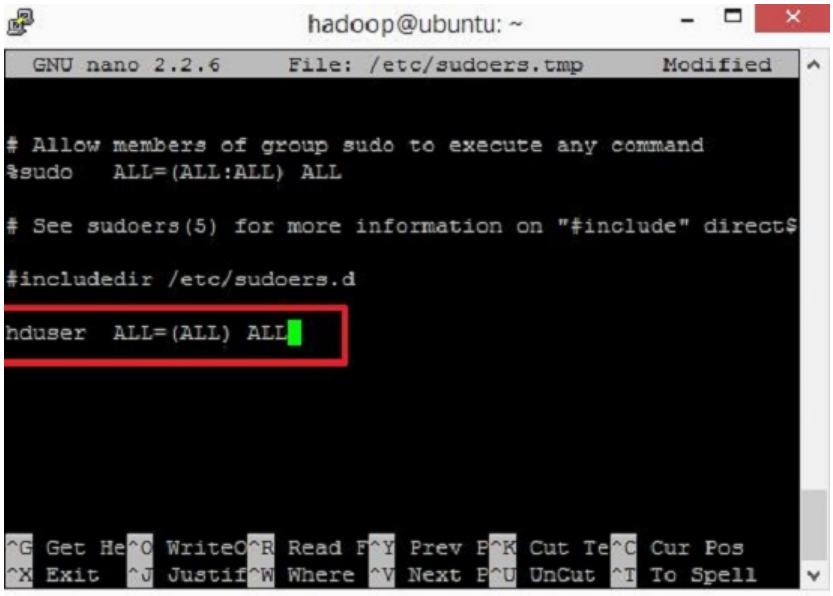
Grant All Permissions to the Hadoop User "hduser created in the above Step

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



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

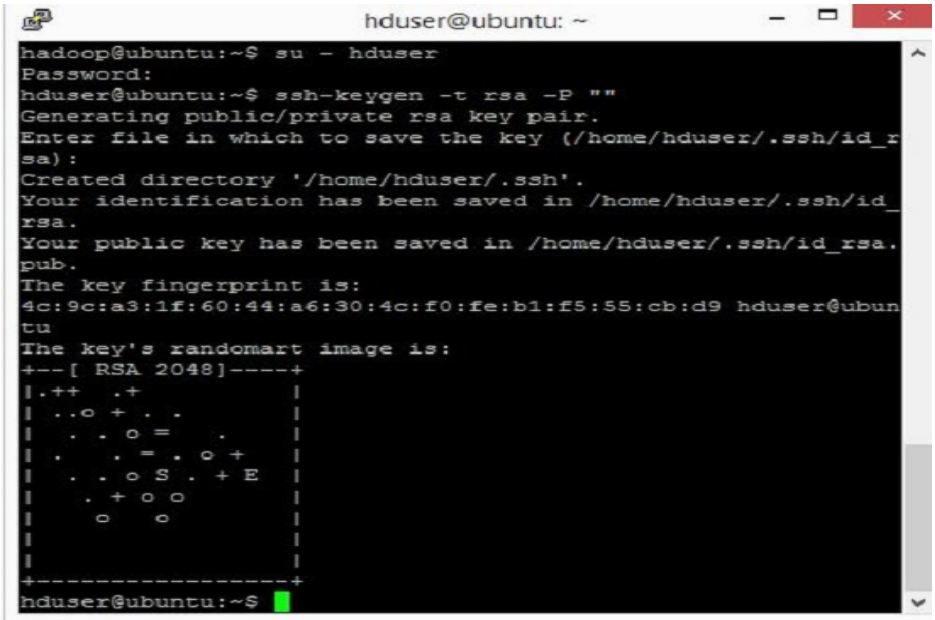
SSH Setup is needed to perform various operations on the hadoop cluster so that the master node can login to the slave nodes to start or stop them.SSH must be setup even on the secondary NameNode listed in the master's file so that it can be started from the Namenode using the command ./start-dfs.sh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

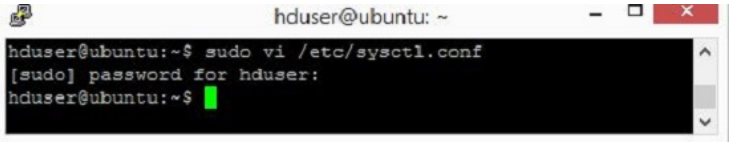


In the above screenshot, the command **hduser@ubuntu:~$ ssh-keygen -t rsa -P ""** command will create an empty password RSA key pair. It is not suggested to use an empty password, however, if you do not

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

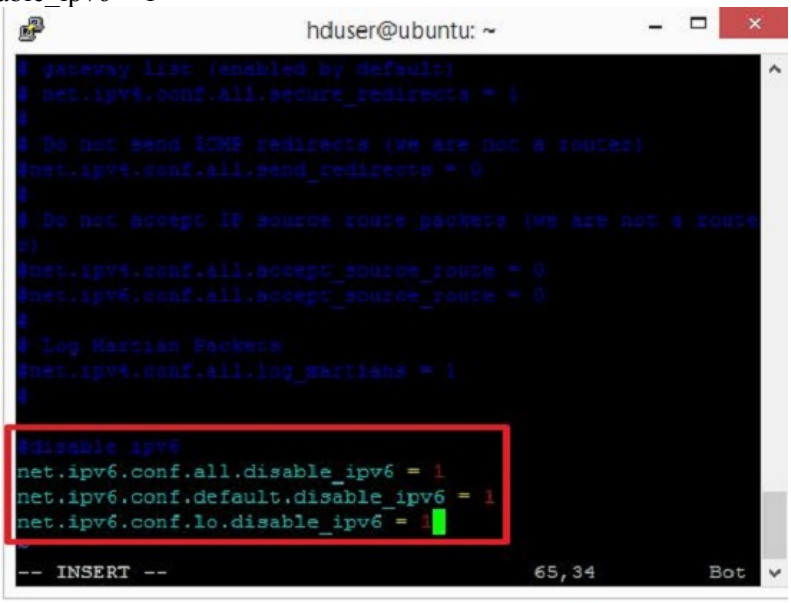
Disabling IPv6

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



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
```



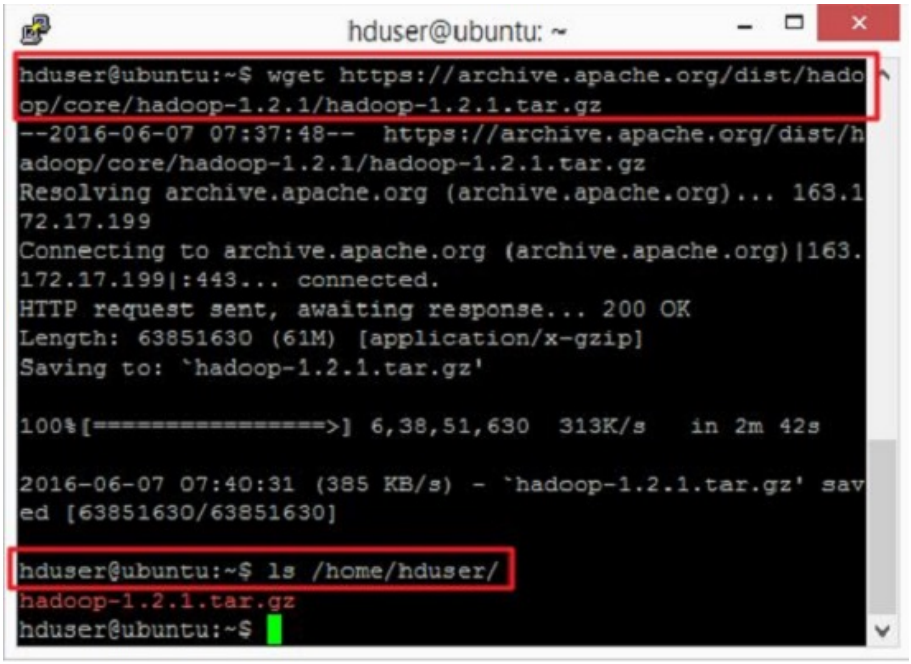
To ensure that IPv6 has been disabled , run the following command -
$cat /proc/sys/net/ipv6/conf/all/disable_ipv6
Having disabled IPv6, it is suggested that you power off the machine and restart it again using the below command -
$ sudo reboot now
Hurray, you have completed the environment setup to install 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



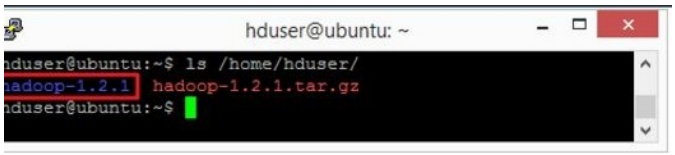
The compressed hadoop file needs to be unzipped as follows -

$ tar -xvzf /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/



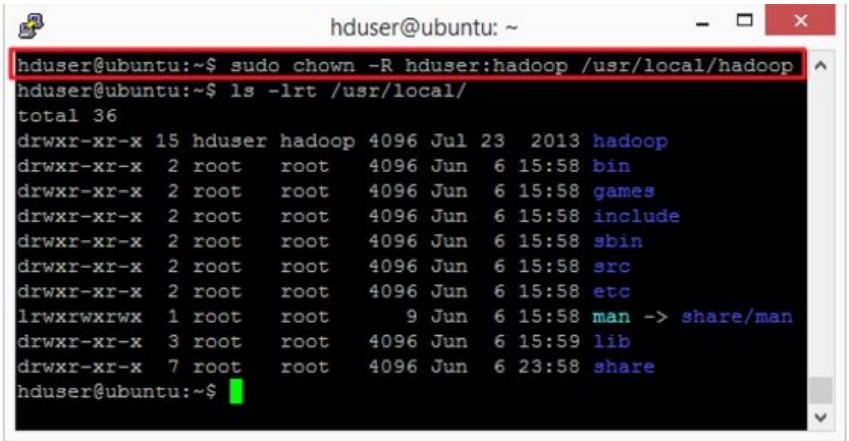
Listing the contents of /home/hduser/ shows that hadoop has been installed. You can now move the contents of the directory to the location of your choice. Let/s move hadoop directory to /usr/local/

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

Assign Ownership of Hadoop to hduser

Ensure that you change thw ownership of all files to “hduser” and the “hadoop” group.

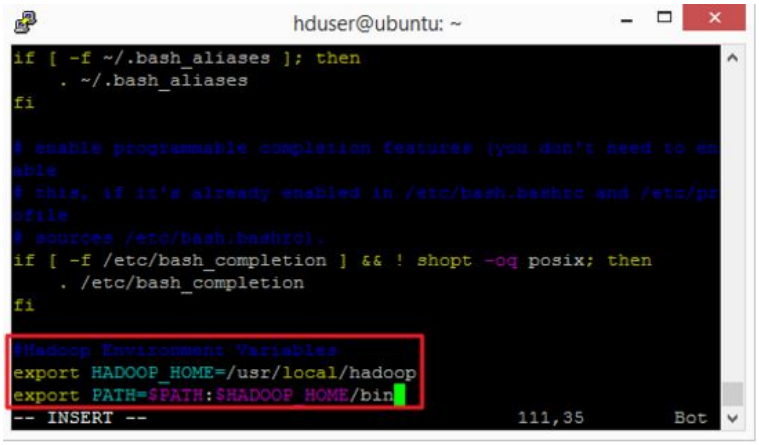
$ sudo chown -R hduser:hadoop /usr/local/hadoop



Hadoop Configuration

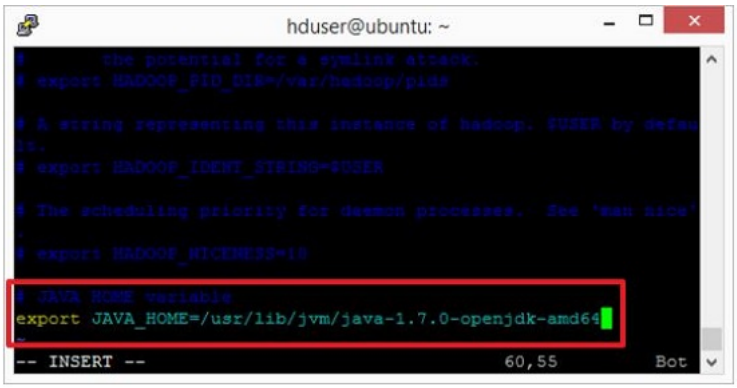
Before hadoop is up and running, you need to configure the hadoop environment.

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



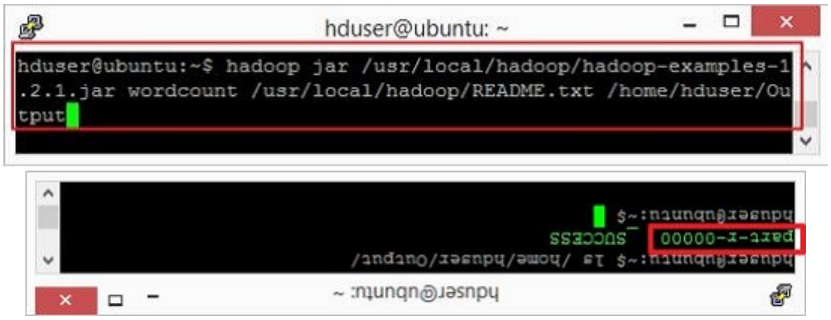
Enter the following command to update bashrc:
$ exec bash

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



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



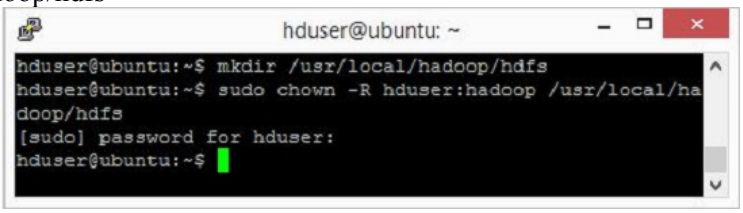
Hadoop Single Node Setup - Pseudo Distributed Mode

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



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.

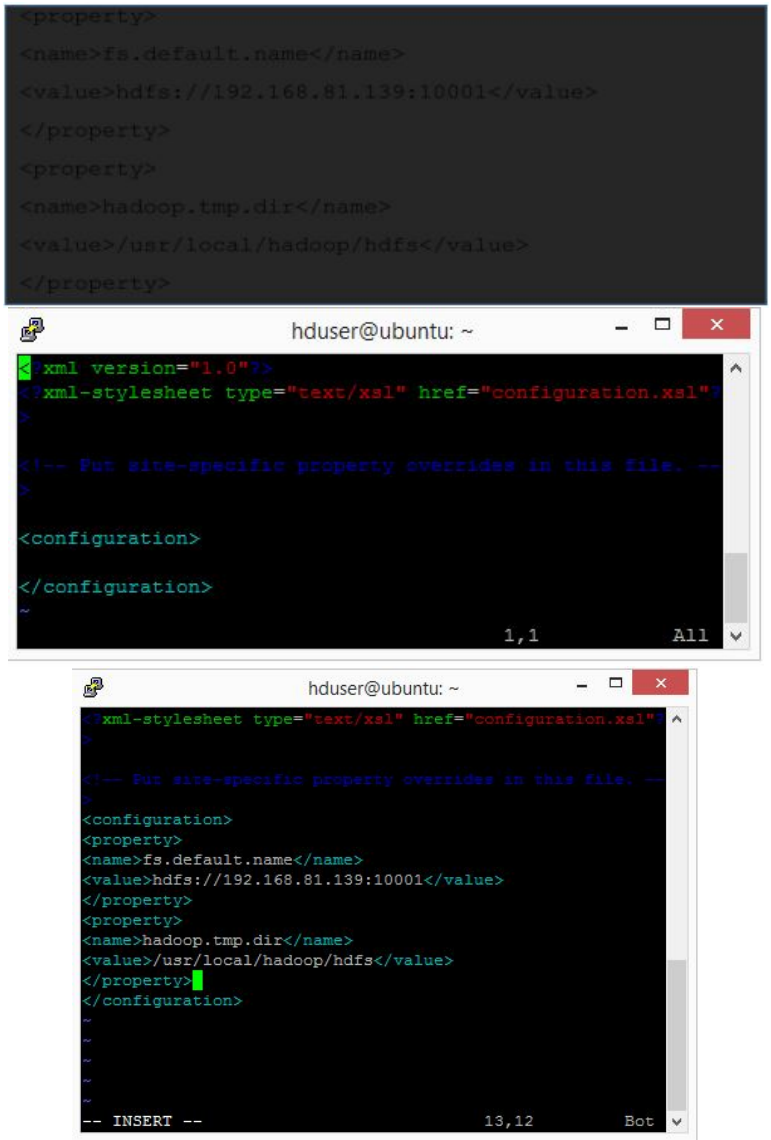
Open the core-site.xml file using the vi editor -

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

Enter the following lines between the configuration tab:

fs.default.name

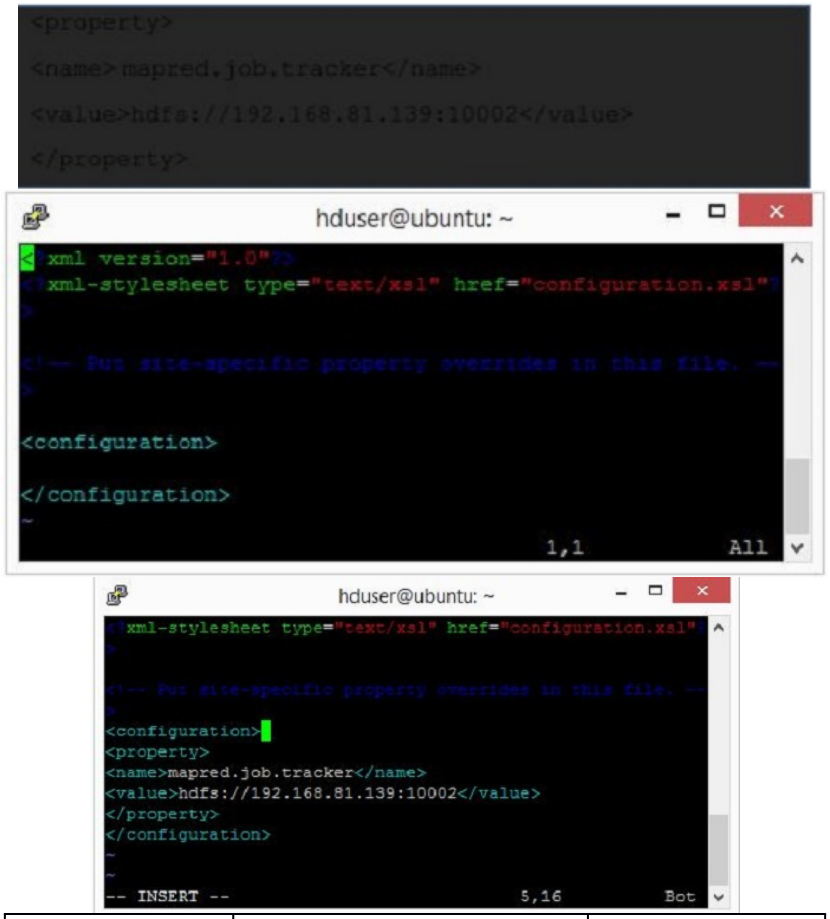
hdfs://192.168.81.139:10001
hadoop.tmp.dir
/usr/local/hadoop/hdfs



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

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



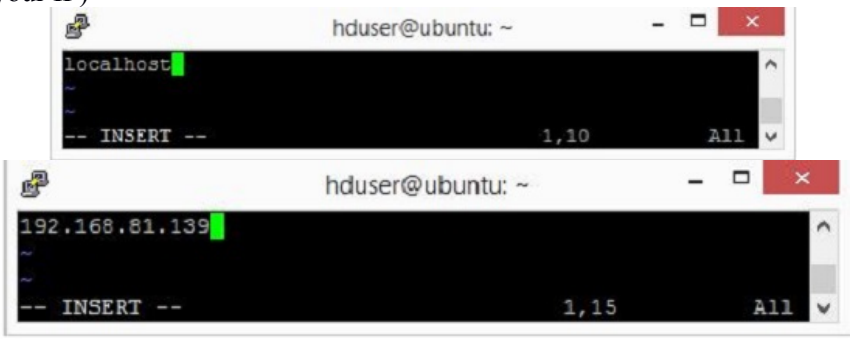
Parameter	Value	Notes
mapred.job.tracker	Host or IP and port of JobTracker.	*hdfs://host:port* pair.

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:

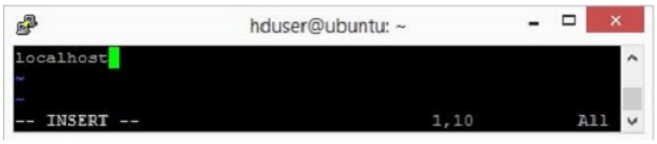
$ sudo nano /usr/local/hadoop/conf/masters

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

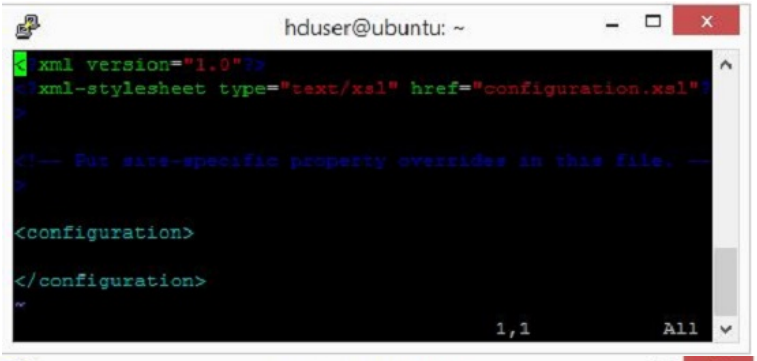


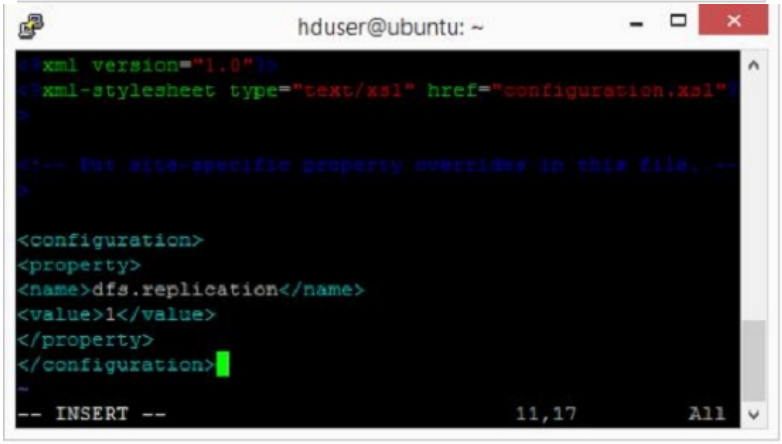
Configuring the Slaves

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



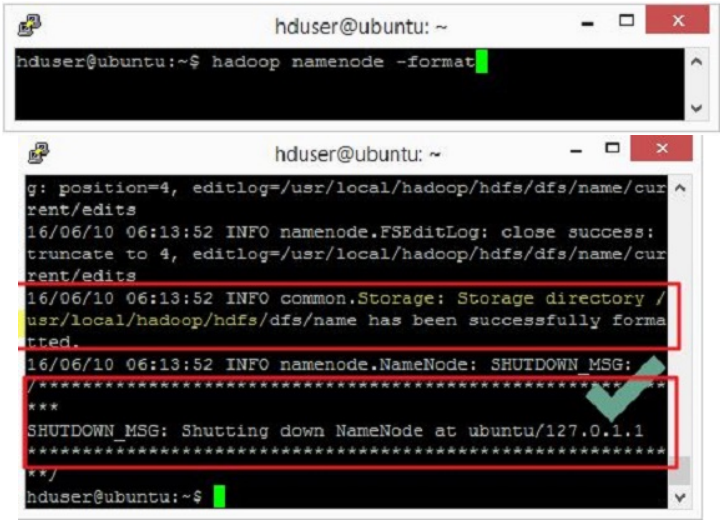


Parameter	Value	Notes
dfs-replication	Value in positive integer	*No. of replicate/duplicate block*

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)

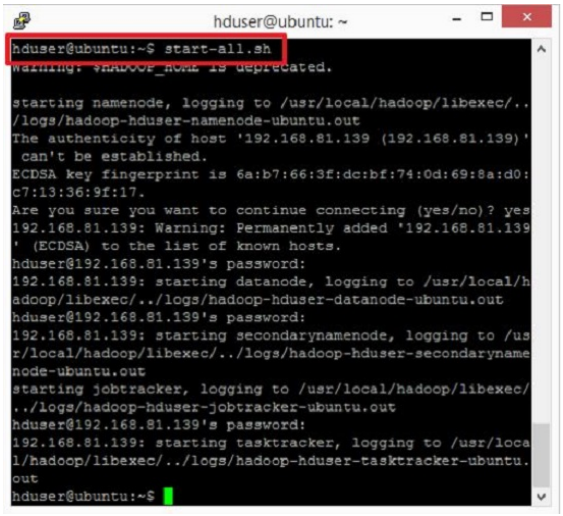
Formatting the NameNode

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

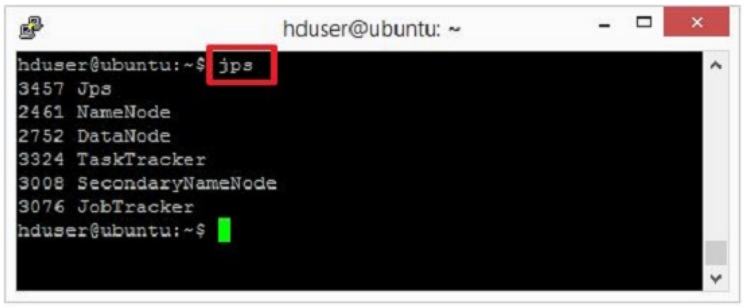


Starting the Hadoop Cluster

Start the Hadoop server by executing the following command:
$ start-all.sh



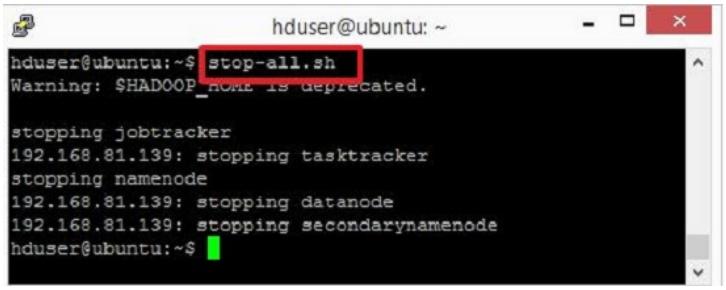
To verify if all the services are up and running, execute the below command-
$ jps



Stop the hadoop servers

Stop the Hadoop server by executing the following command:

$ stop-all.sh

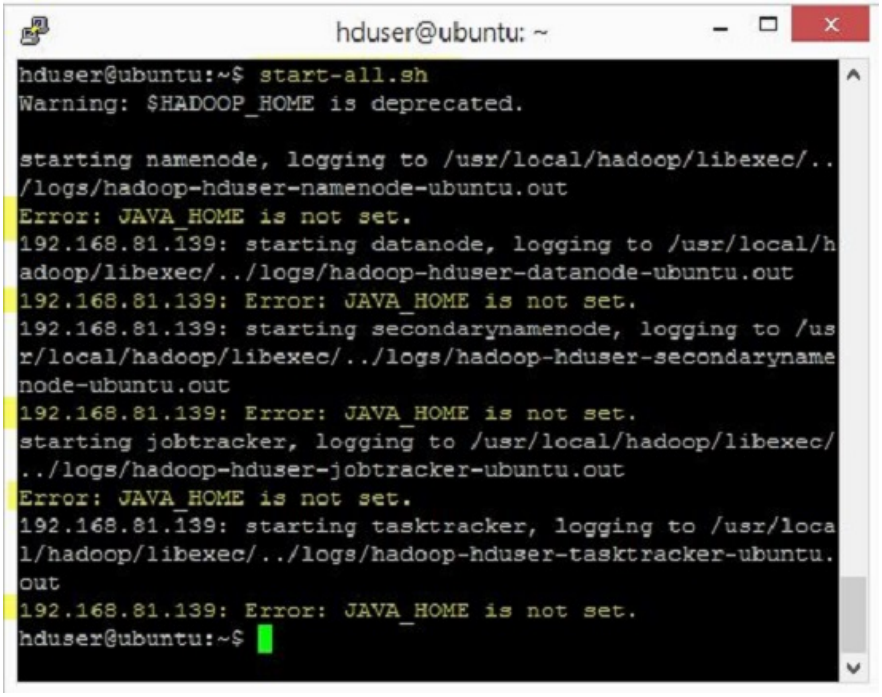


Troubleshooting

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

Common Errors Encountered during Hadoop Installation

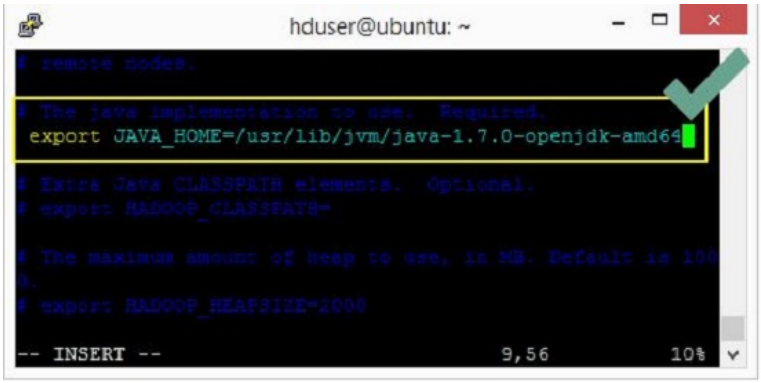
Error :JAVA_HOME is not set.



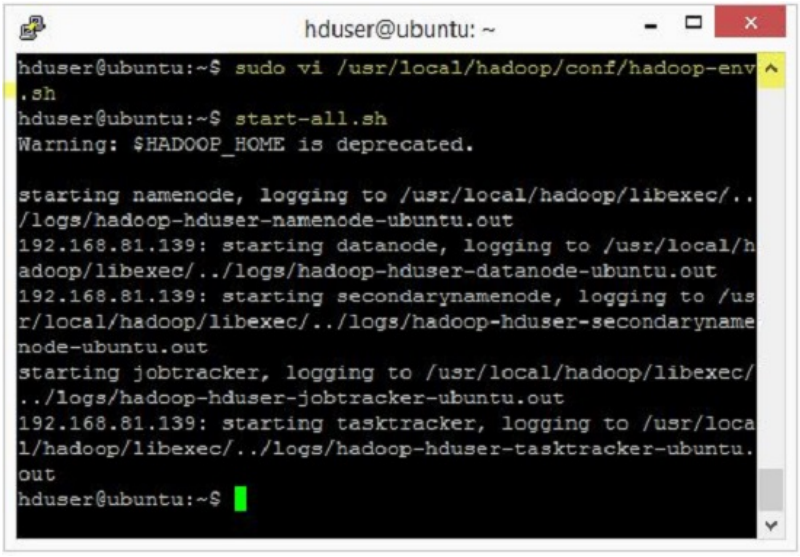
Solution to JAVA_HOME is not set error -

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

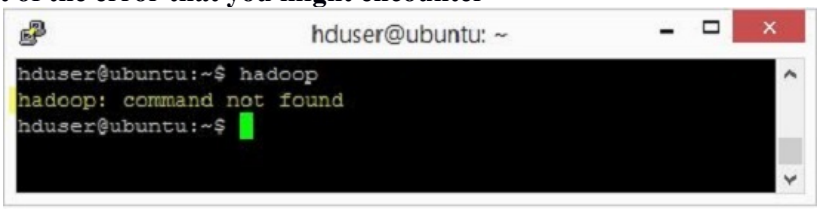
$ sudo vi $HADOOP_HOME/conf/hadoop-env.sh



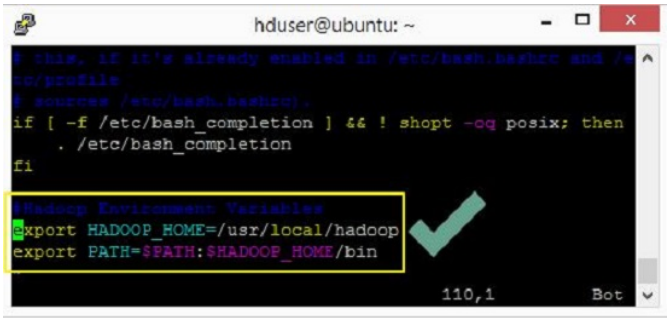
Just to ensure that the error is resolved, try starting the hadoop server again using the hadoop command start-all.sh.



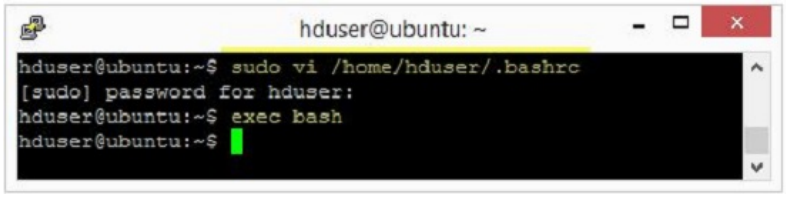
Error: hadoop: command not found
Here's a snapshot of the error that you might encounter -



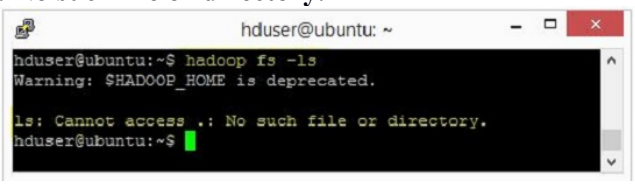
To resolve this issue, open the .bachrc fileusing vi editor and define the path for HADOOP_HOME/bin -
$ sudo vi /home/hduser/.bashrc



Execute the below command to update the .bashrc -

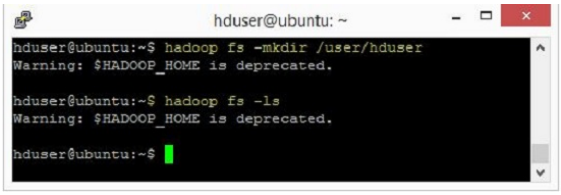


Error: ls cannot access .: No such file or directory.

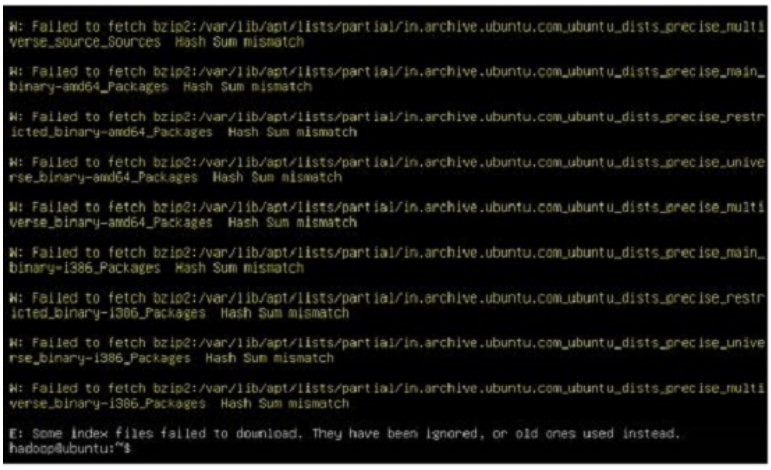


When you do not mention the path after the ls command, it takes the default path . /user/. To resolve this issue, create a folder under user/ i.e. in this tutorial it is user/hduser.

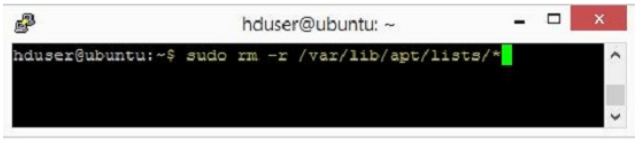
$ hadoop fs -mkdir /user/hduse



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



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



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.