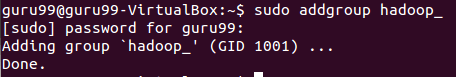
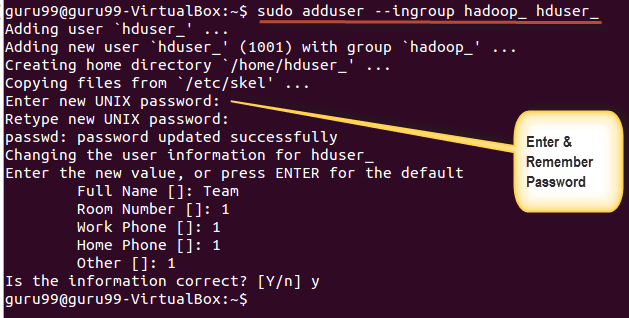
**Part 1) Download and Install Hadoop**

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

sudo addgroup hadoop\_

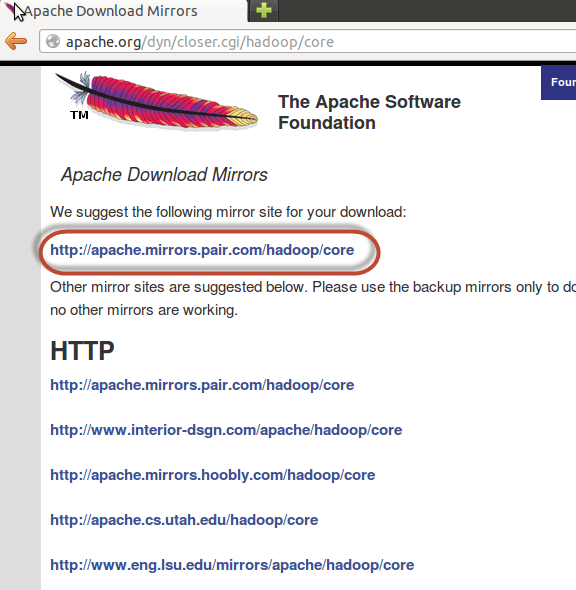


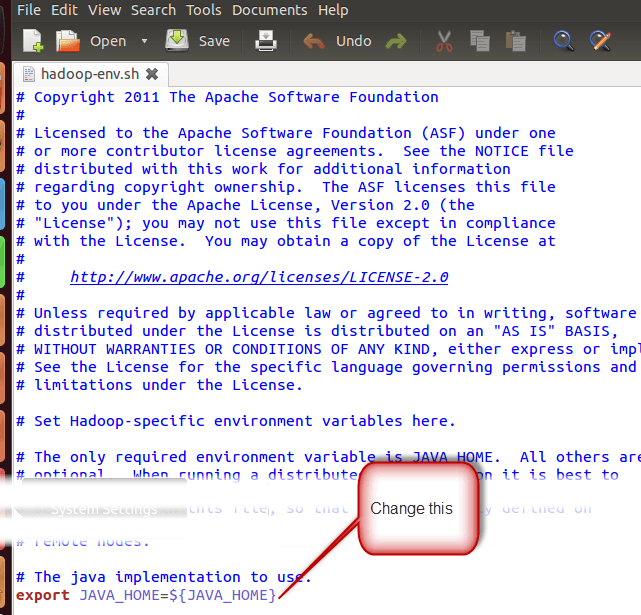
sudo adduser --ingroup hadoop\_ hduser\_



Enter your password, name and other details.

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





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

**1.** **‘hadoop.tmp.dir’ –**Used to specify a directory which will be used by Hadoop to store its data files.

**2. ‘fs.default.name’ –**This specifies the default file system.

To set these parameters, open core-site.xml

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

Hadoop Setup Tutorial - Installation & Configuration

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

<property>

<name>hadoop.tmp.dir</name>

<value>/app/hadoop/tmp</value>

<description>Parent directory for other temporary directories.</description>

</property>

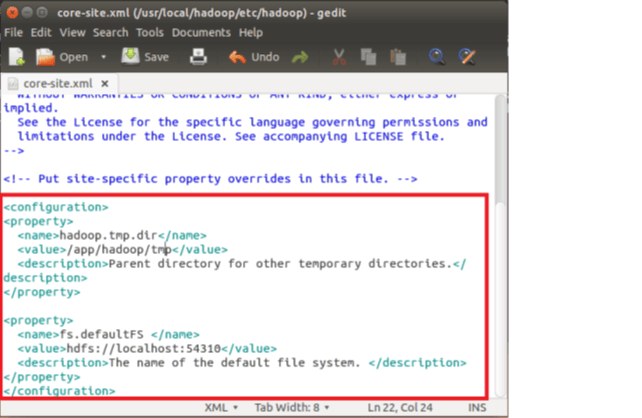
<property>

<name>fs.defaultFS </name>

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. </description>

</property>



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

Hadoop Setup Tutorial - Installation & Configuration

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

sudo mkdir -p <Path of Directory used in above setting>

Hadoop Setup Tutorial - Installation & Configuration

Grant permissions to the directory

sudo chown -R hduser\_:Hadoop\_ <Path of Directory created in above step>

Hadoop Setup Tutorial - Installation & Configuration

sudo chmod 750 <Path of Directory created in above step>

Hadoop Setup Tutorial - Installation & Configuration

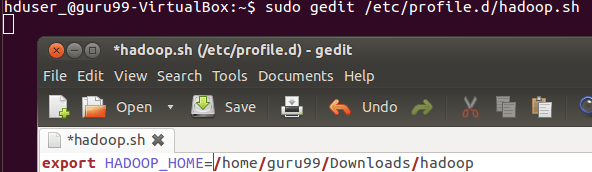
**Step 3)** Map Reduce Configuration

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

sudo gedit /etc/profile.d/hadoop.sh

And Enter

export HADOOP\_HOME=/home/guru99/Downloads/Hadoop



Next enter

sudo chmod +x /etc/profile.d/hadoop.sh

Hadoop Setup Tutorial - Installation & Configuration

Exit the Terminal and restart again

Type echo $HADOOP\_HOME. To verify the path

Hadoop Setup Tutorial - Installation & Configuration

Now copy files

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

Hadoop Setup Tutorial - Installation & Configuration

Open the **mapred-site.xml** file

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

Hadoop Setup Tutorial - Installation & Configuration

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

<property>

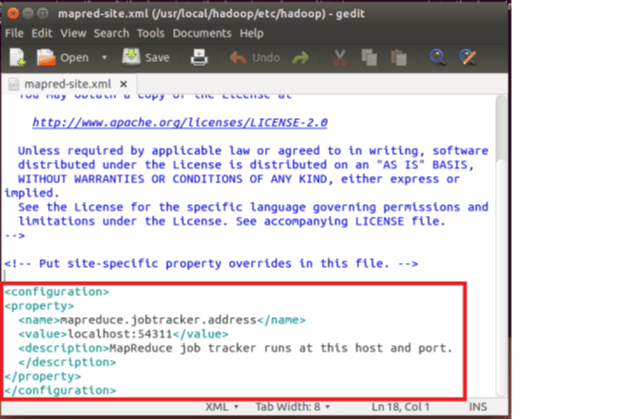
<name>mapreduce.jobtracker.address</name>

<value>localhost:54311</value>

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

</description>

</property>



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

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

Hadoop Setup Tutorial - Installation & Configuration

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

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.</description>

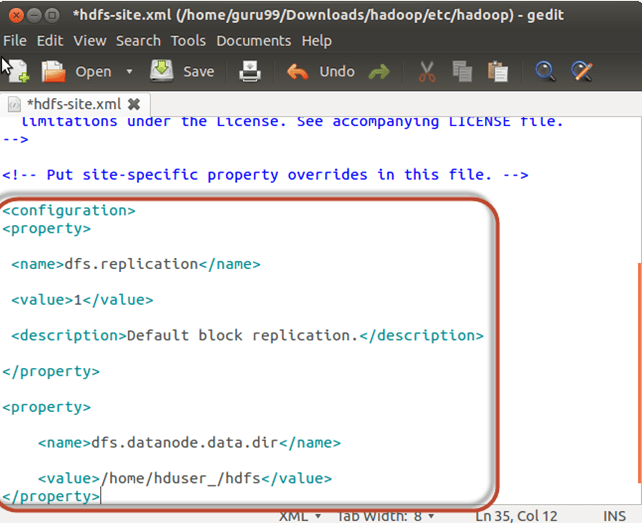
</property>

<property>

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

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

</property>



Create a directory specified in above setting-

sudo mkdir -p <Path of Directory used in above setting>

sudo mkdir -p /home/hduser\_/hdfs

Hadoop Setup Tutorial - Installation & Configuration

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

sudo chown -R hduser\_:hadoop\_ /home/hduser\_/hdfs

Hadoop Setup Tutorial - Installation & Configuration

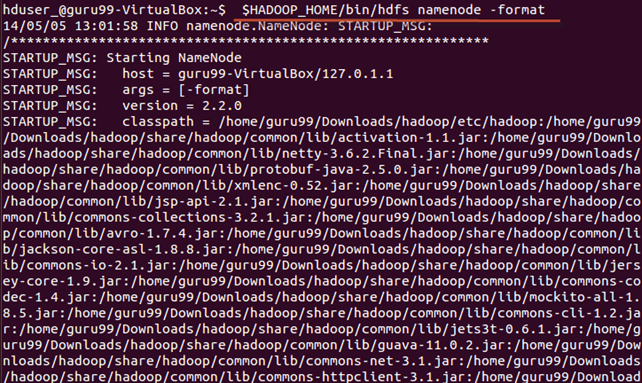
sudo chmod 750 <Path of Directory created in above step>

sudo chmod 750 /home/hduser\_/hdfs

Hadoop Setup Tutorial - Installation & Configuration

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

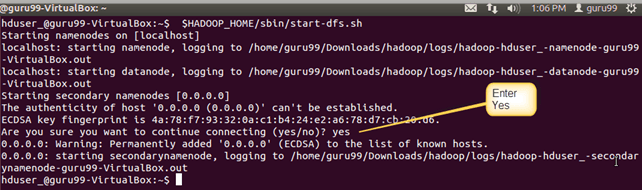
$HADOOP\_HOME/bin/hdfs namenode -format



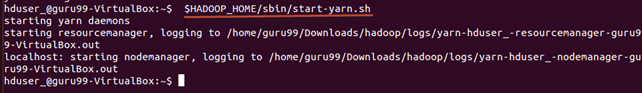
**Step 5)**Start Hadoop single node cluster using below command

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

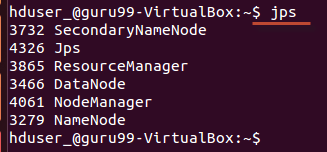
An output of above command



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



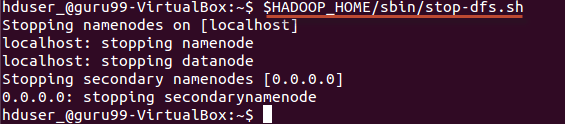
Using **‘jps’** tool/command, verify whether all the Hadoop related processes are running or not.



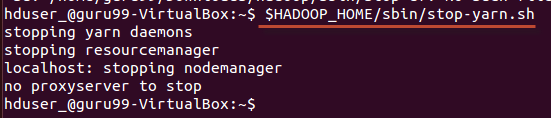
If Hadoop has started successfully then an output of jps should show NameNode, NodeManager, ResourceManager, SecondaryNameNode, DataNode.

**Step 6)**Stopping Hadoop

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



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



Map Reduce

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

**Usage**

Assuming environment variables are set as follows:

export JAVA\_HOME=/usr/java/default

export PATH=${JAVA\_HOME}/bin:${PATH}

export HADOOP\_CLASSPATH=${JAVA\_HOME}/lib/tools.jar

Compile WordCount.java and create a jar:

$ bin/hadoop com.sun.tools.javac.Main WordCount.java

$ jar cf wc.jar WordCount\*.class

Assuming that:

* /user/joe/wordcount/input - input directory in HDFS
* /user/joe/wordcount/output - output directory in HDFS

Sample text-files as input:

$ bin/hadoop fs -ls /user/joe/wordcount/input/

/user/joe/wordcount/input/file01

/user/joe/wordcount/input/file02

$ bin/hadoop fs -cat /user/joe/wordcount/input/file01

Hello World Bye World

$ bin/hadoop fs -cat /user/joe/wordcount/input/file02

Hello Hadoop Goodbye Hadoop

Run the application:

$ bin/hadoop jar wc.jar WordCount /user/joe/wordcount/input /user/joe/wordcount/output

Output:

$ bin/hadoop fs -cat /user/joe/wordcount/output/part-r-00000

Bye 1

Goodbye 1

Hadoop 2

Hello 2

World 2