**Hadoop Hello World Example - Count Word**

In this tutorial, we are going to count word in a text file using Hadoop MapReduce

**System requirements**

- Linux version

- Hadoop installed

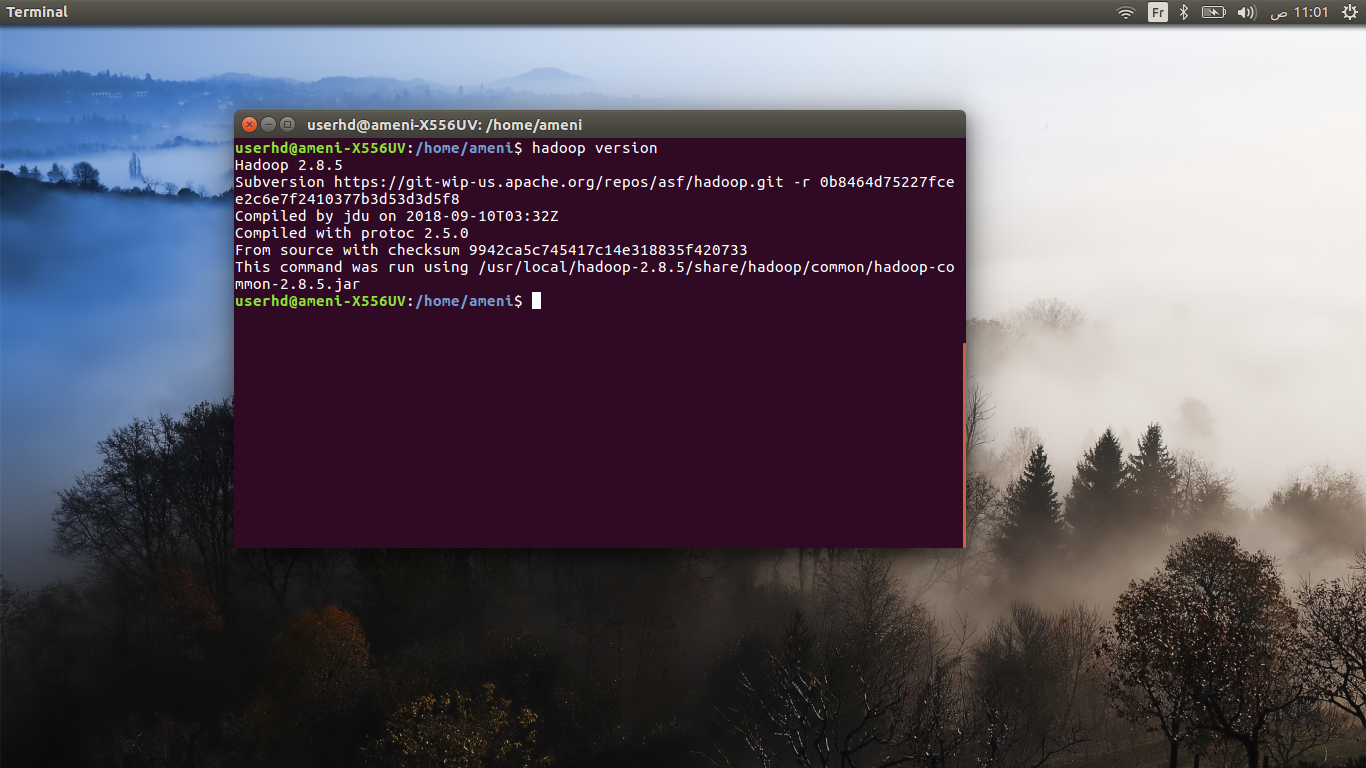
- Java installed

**Step 1 : Check the proper running of Hadoop**

We are going to use the user created previously named **‘userhd’**

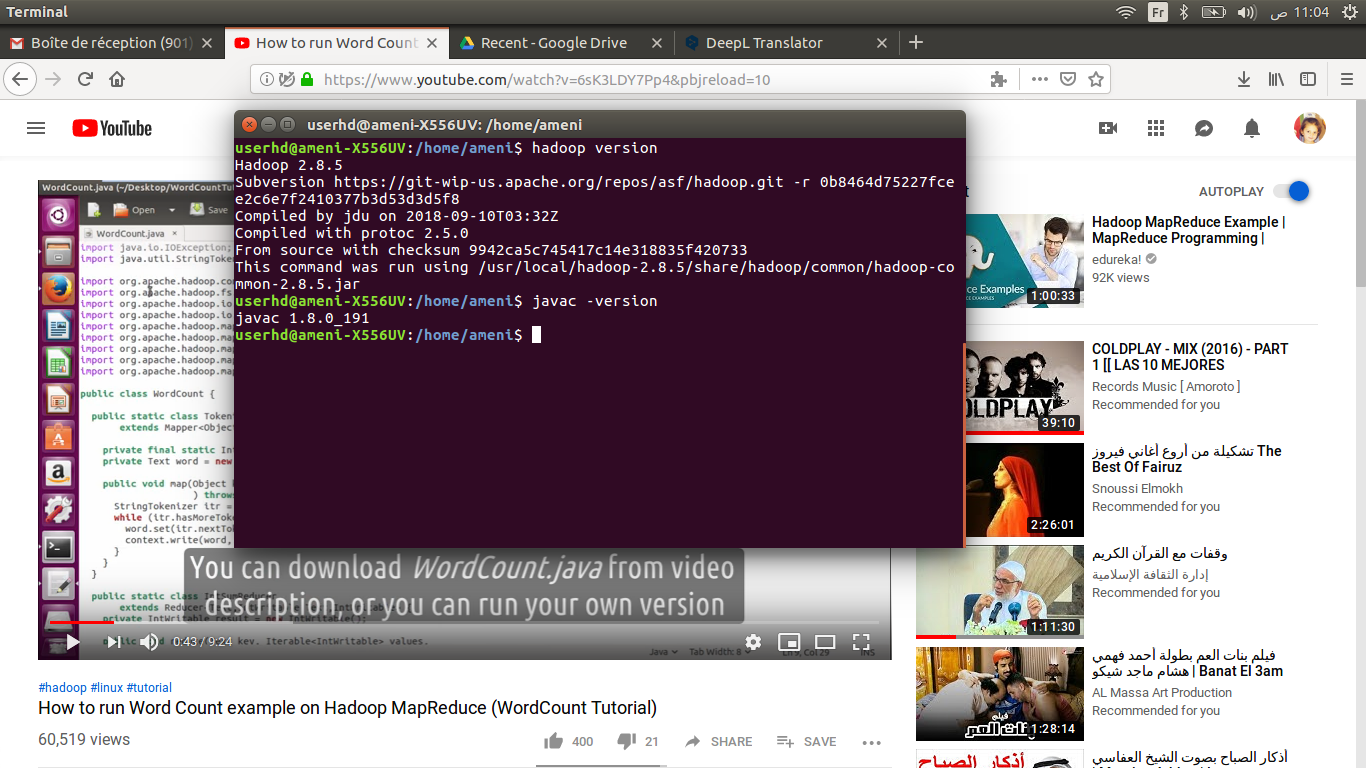
1- Make sure Hadoop is installed & running :

hadoop version



2- Make sure Java is running correctly :

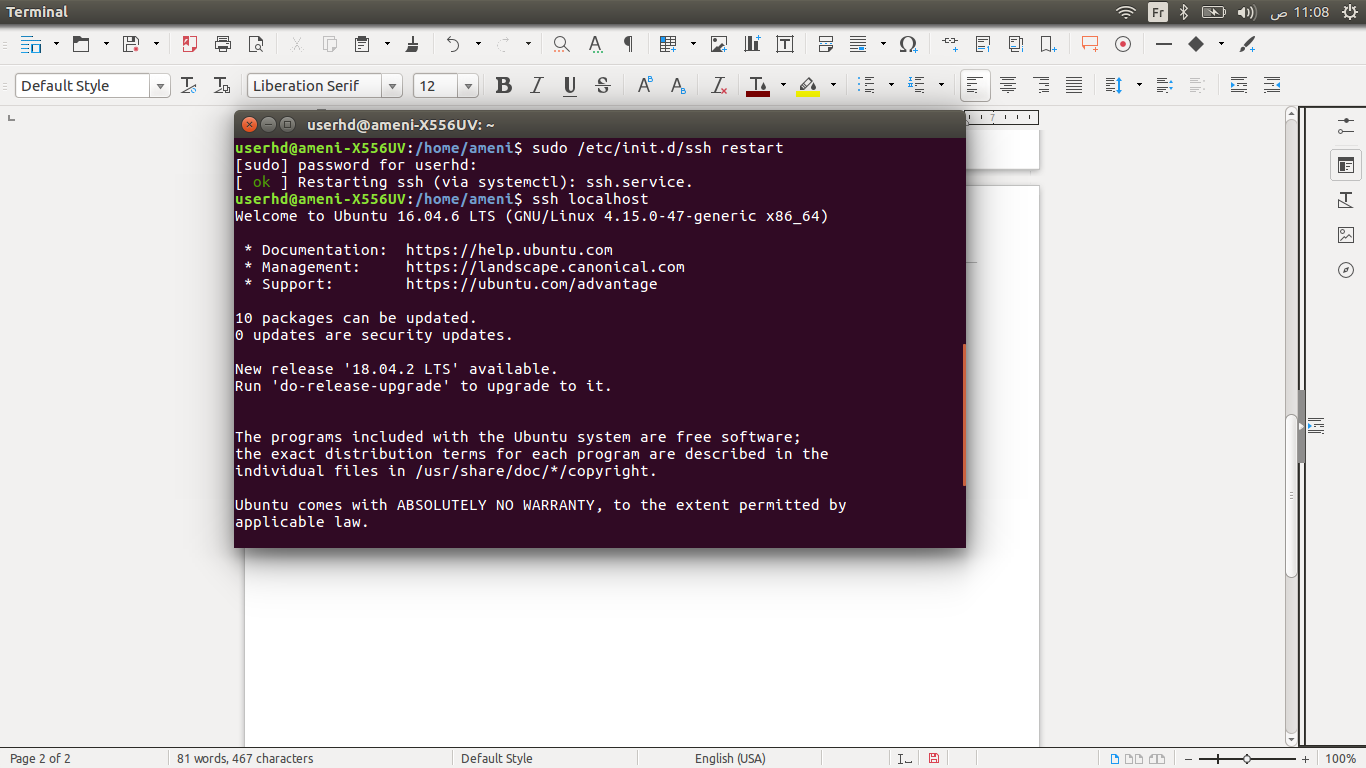
javac -version



3- Start Hadoop via SSH

sudo /etc/init.d/ssh restart

ssh localhost



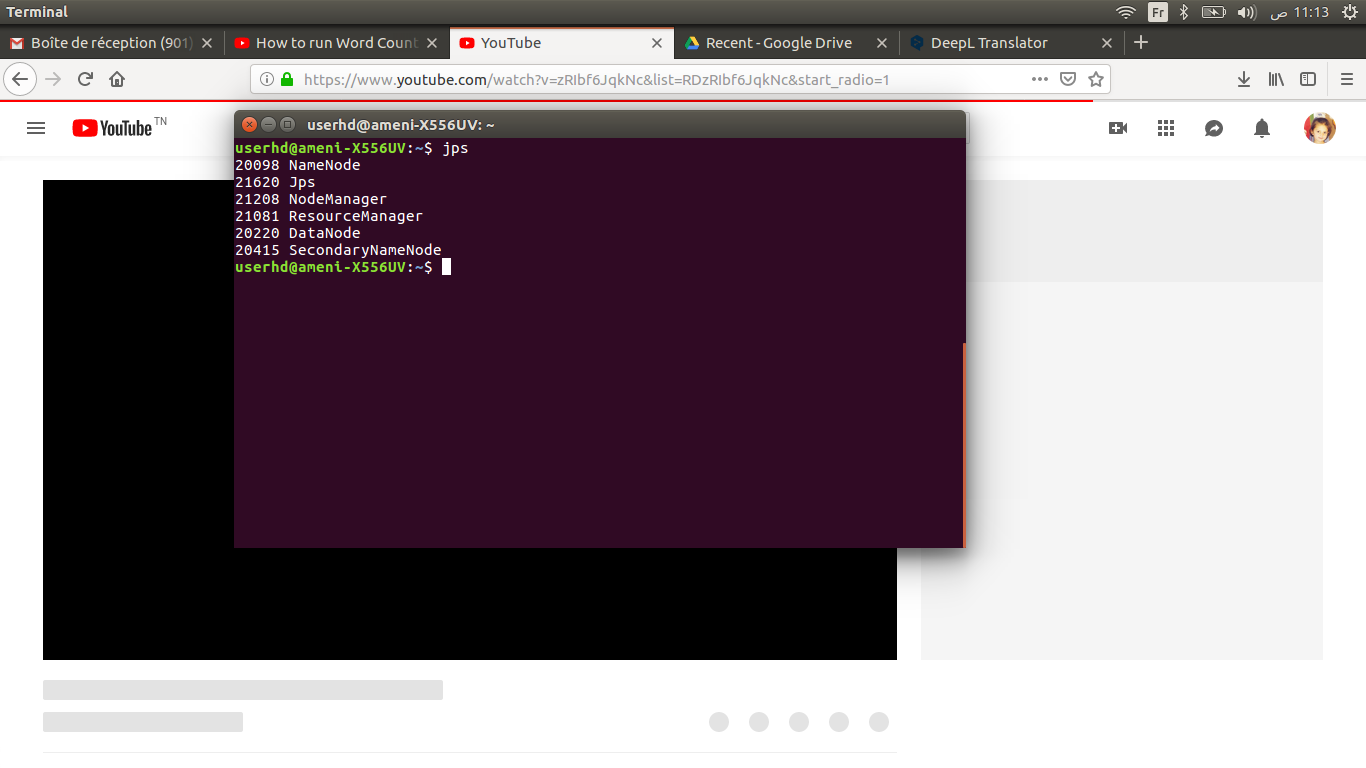
4- Start Hadoop via :

start-dfs.sh

start-all.sh



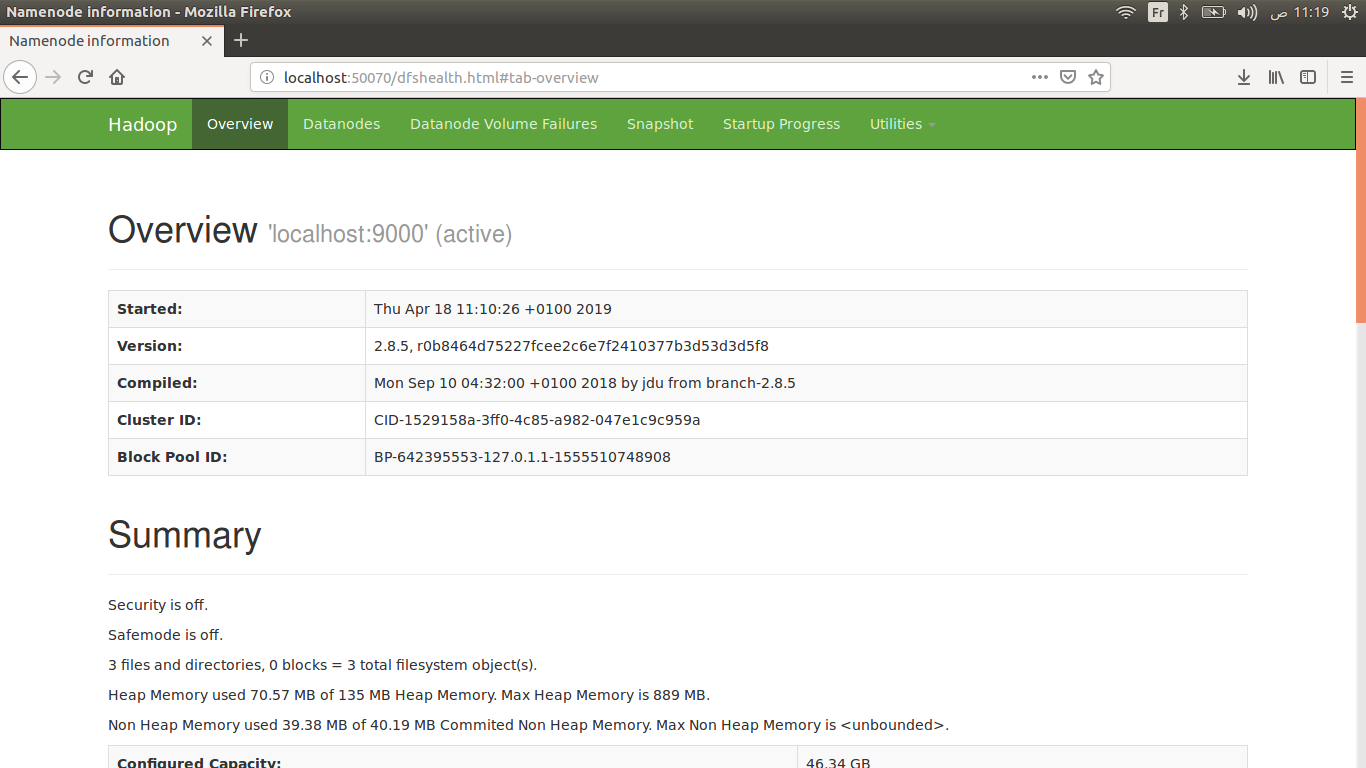
5- Check Hadoop services are running :

 jps

- We must check that the local server is functional while typing the next URL in the browser :

[**http://localhost:50070/**](http://localhost:50070/)

The interface displayed must be as follows :



\*\*\* Forced start of **‘datanode’** :

If you encounter a problem when observing that the **datanode** service is not listed in the list returned by the command **'jps'**, proceed as follows:

hdfs dfsadmin -report

hadoop /usr/local/hadoop/bin datanode

**Step 2 : Hello World Example (Count Word)**

We will now compile the Java example code (Count Word) to verify that the compilation environment is functional, and that Hadoop is properly able to perform tasks, and become familiar with the compilation.

1- Input directory configuration

- Create a directory where Hadoop will hold input data, projects and java classes and give the right permissions.

- Change the owner of these directories to the user : **userhd**, group : **myhadoop** ( created in the previous tutorial ‘Hadoop installation’)

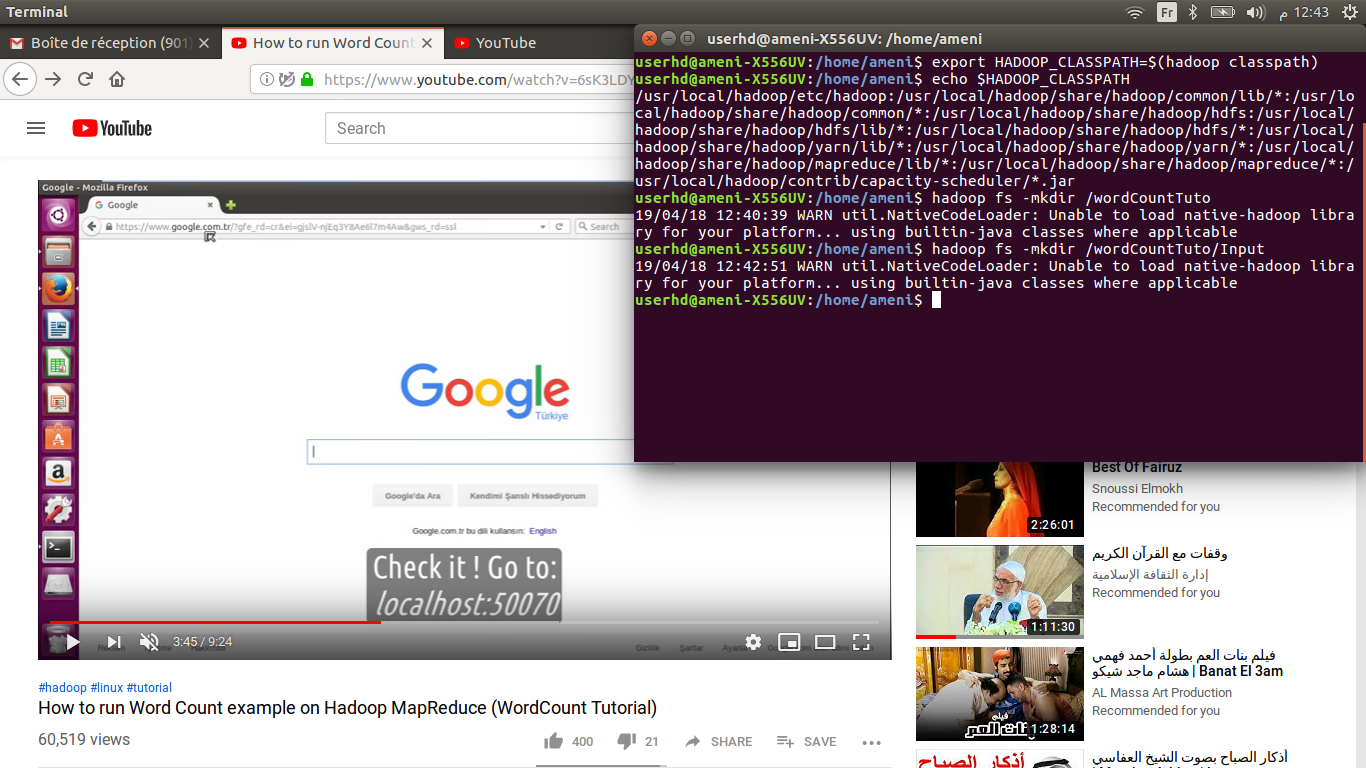
- Create a directory on **HDFS** and set Hadoop classpath :

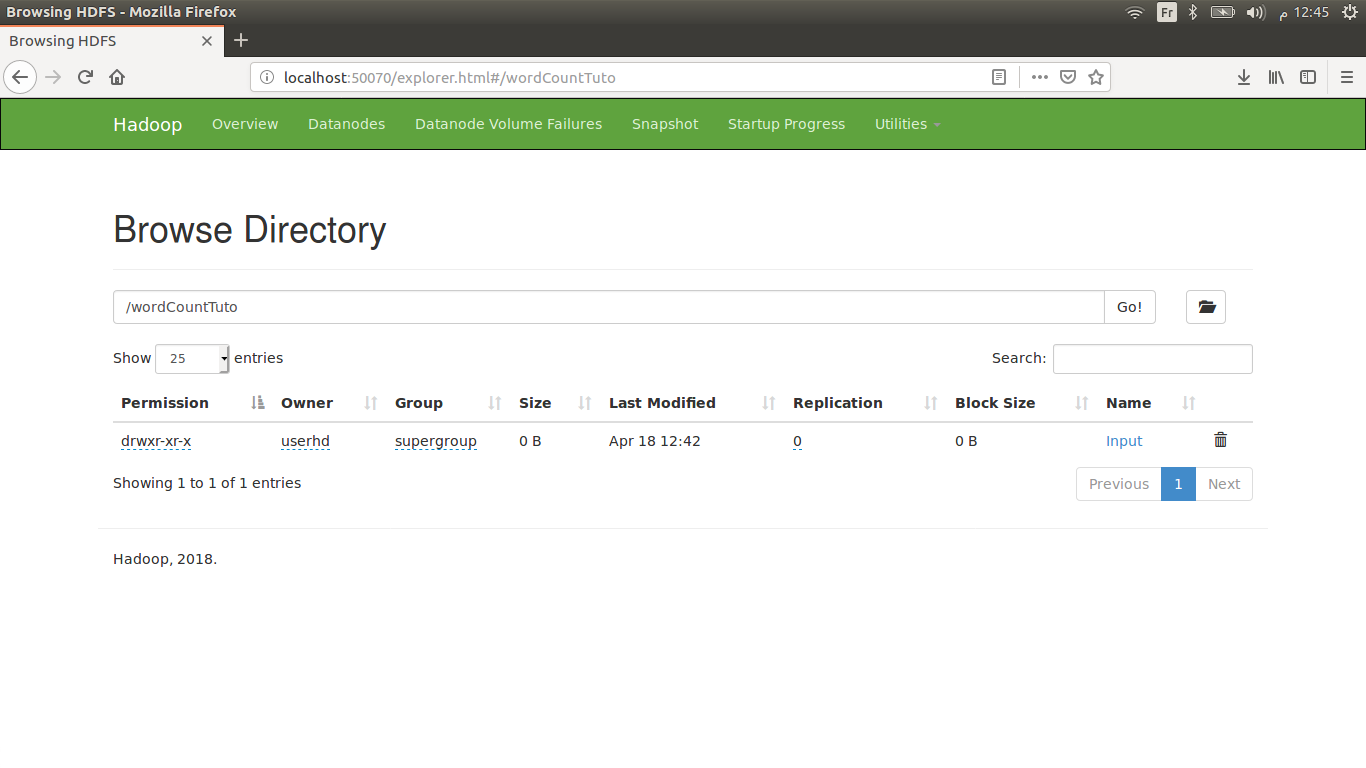
export HADOOP\_CLASSPATH=$(hadoop classpath)

hadoop fs -mkdir <DIRECTORY\_NAME>

- Create a directory inside it for the input :

hadoop fs -mkdir <HDFS\_INPUT\_DIRECTORY>



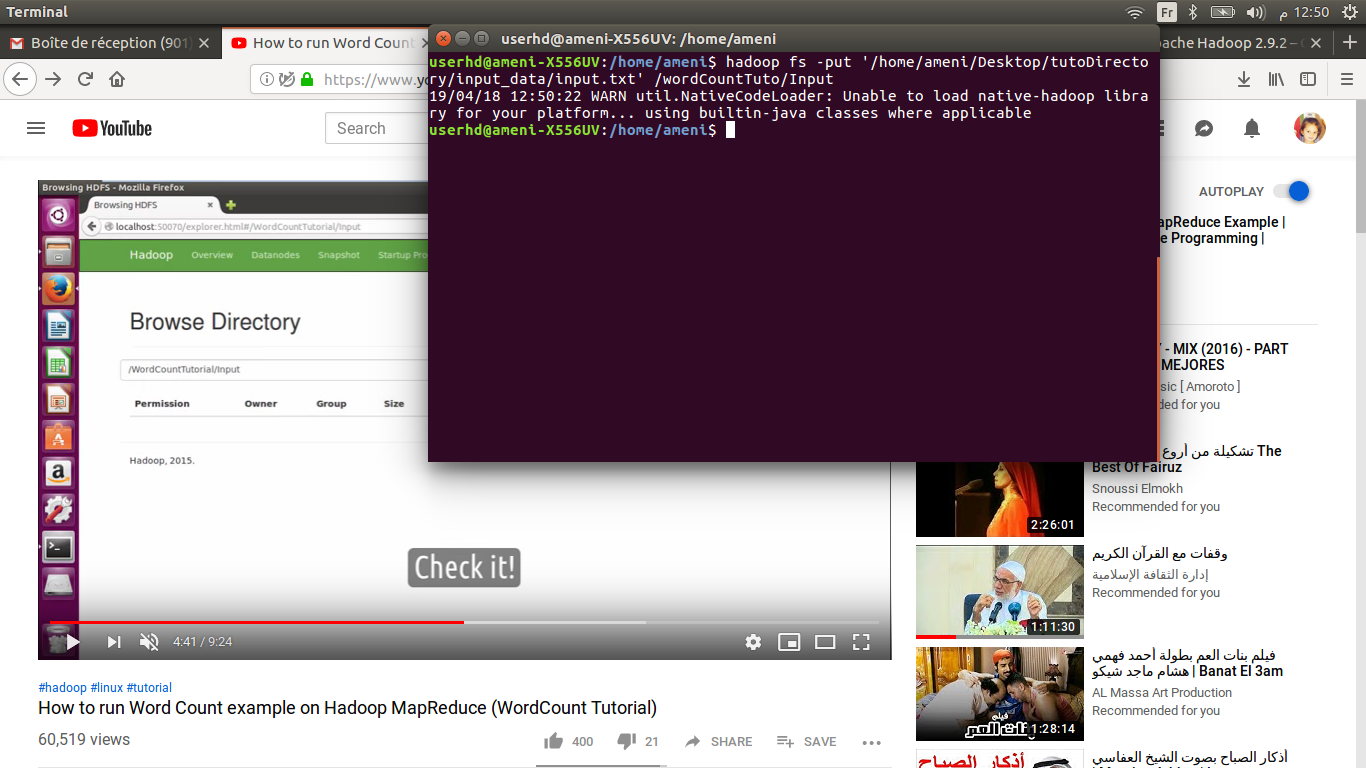
* Check it and go to <http://localhost:50070/>

2- Prepare data

- Create your own text file.

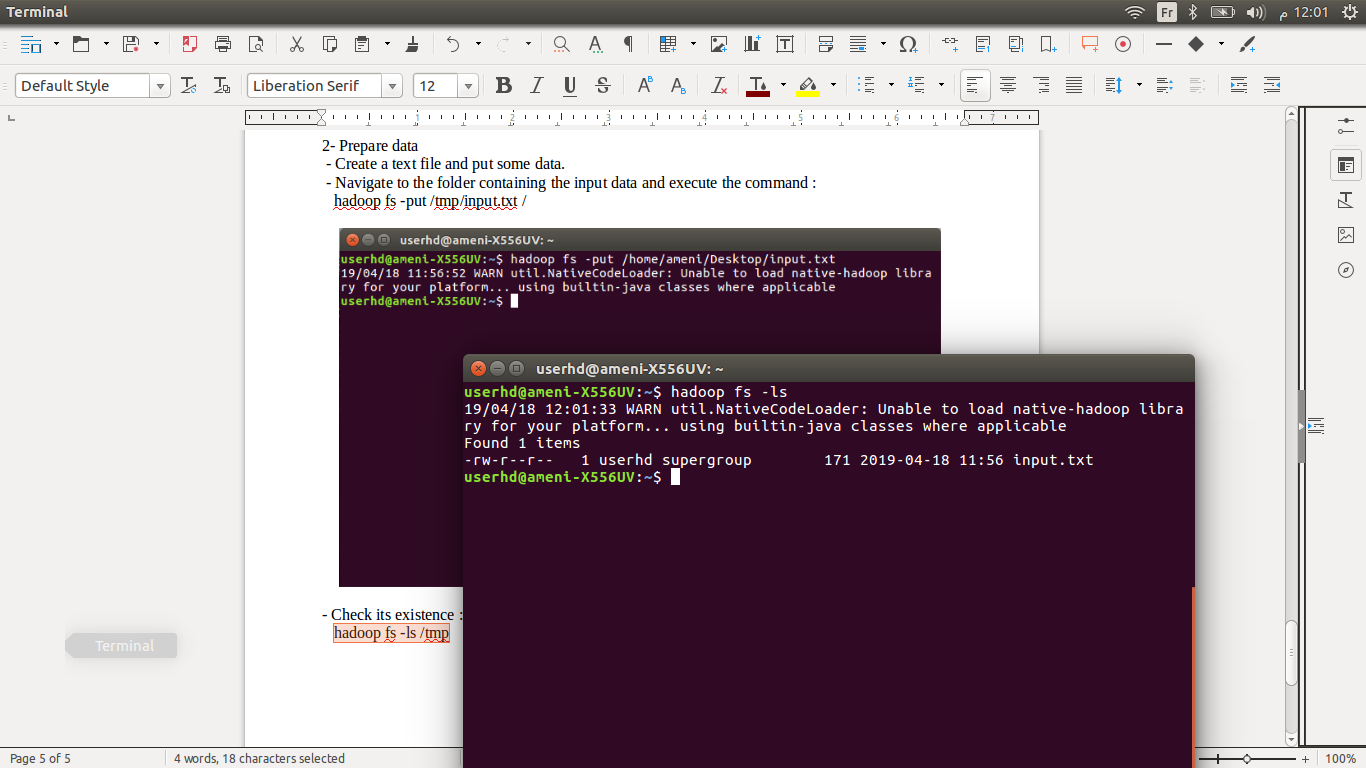
- Navigate to the folder containing the input data and execute the command :

hadoop fs -put <INPUT\_FILE> <HDFS\_INPUT\_DIRECTORY>

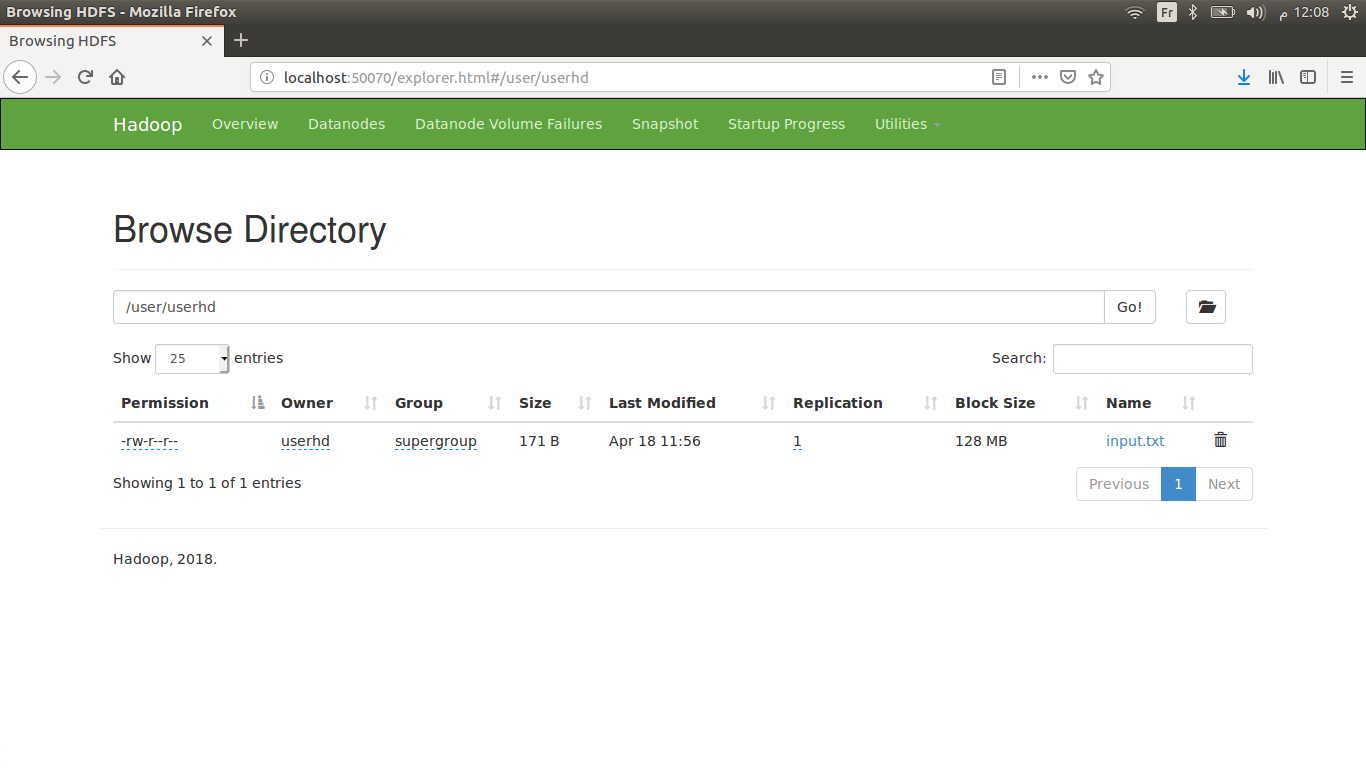


- Check its existence :

hadoop fs -ls



- Check it again and go to **<http://localhost:50070/>**



3- Create a directory **‘tutoDirectory‘** then create a Java class **‘ WordCount.java’**

- Write down the code below :

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]));

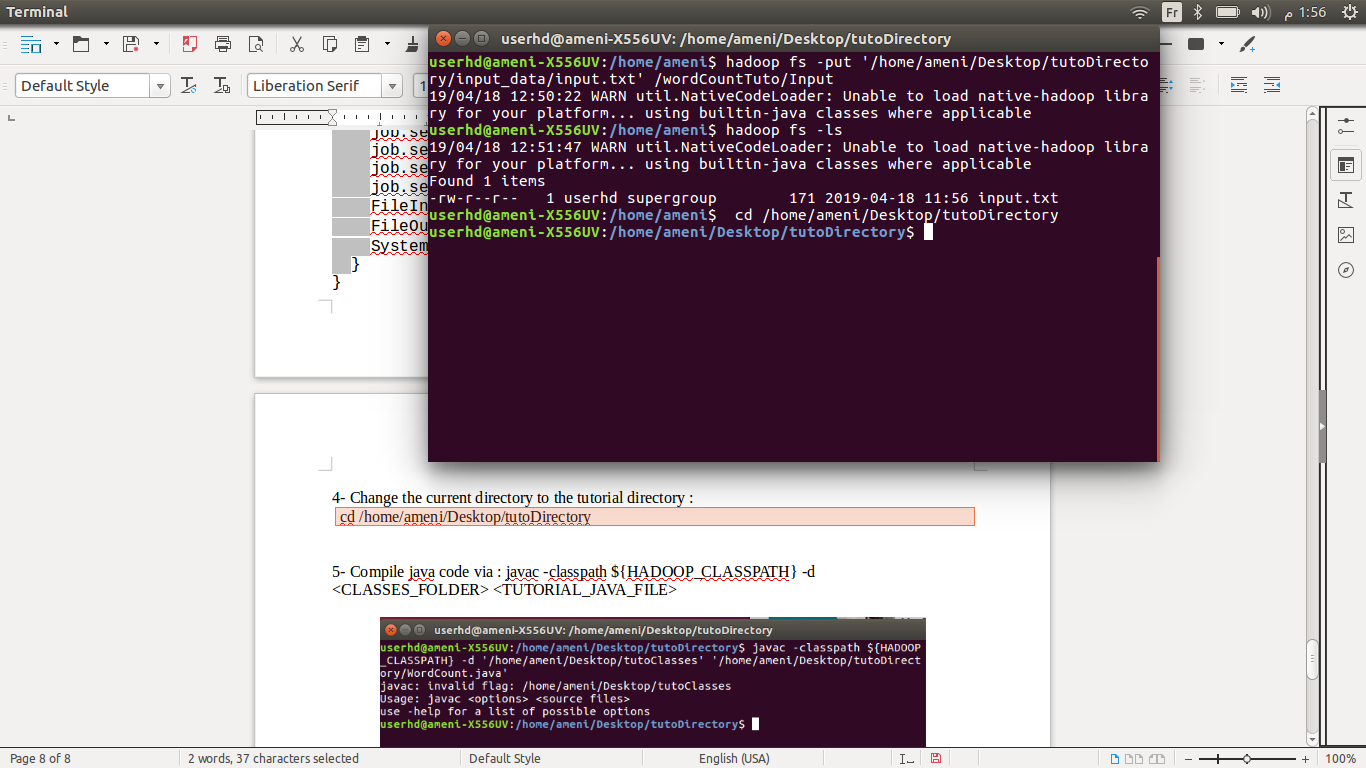
    System.exit(job.waitForCompletion(true) ? 0 : 1);

  }

}

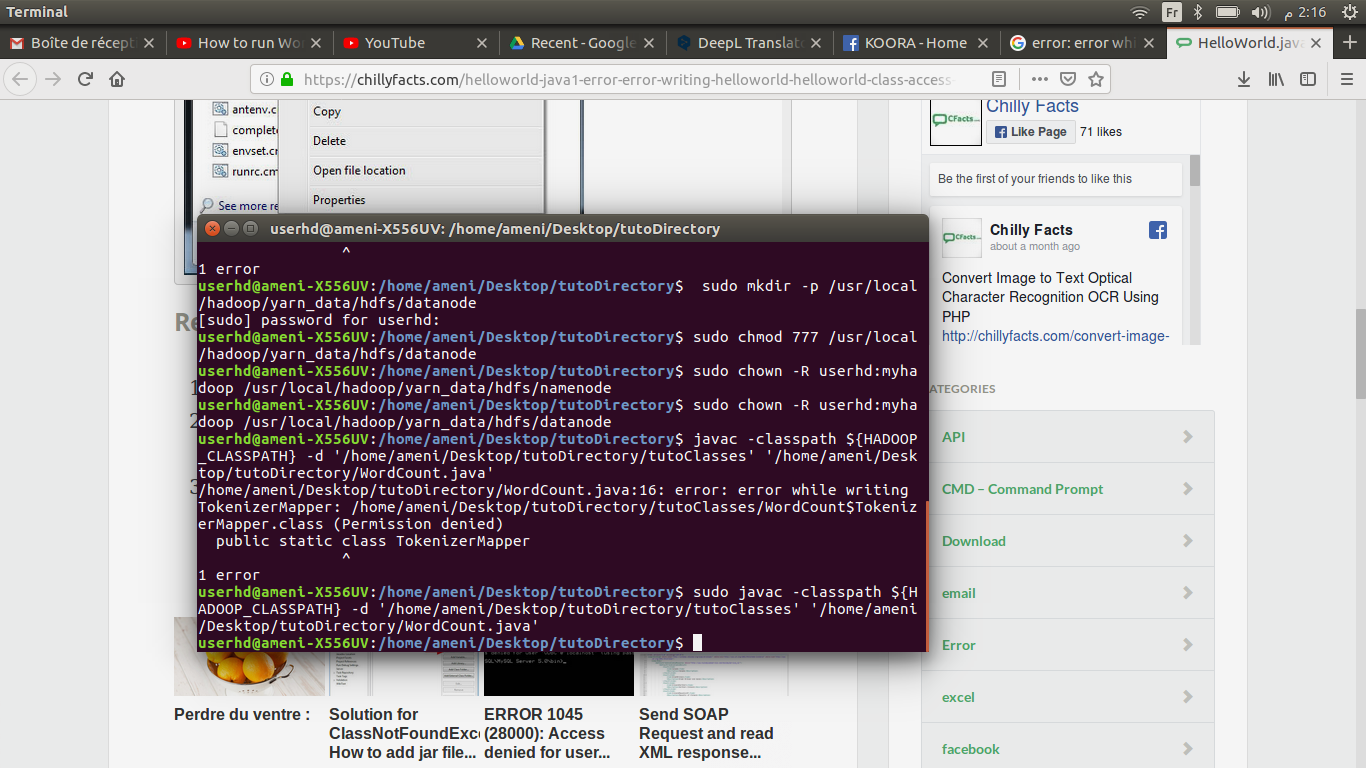
4- Change the current directory to the tutorial directory :

cd <DIRECTORY>

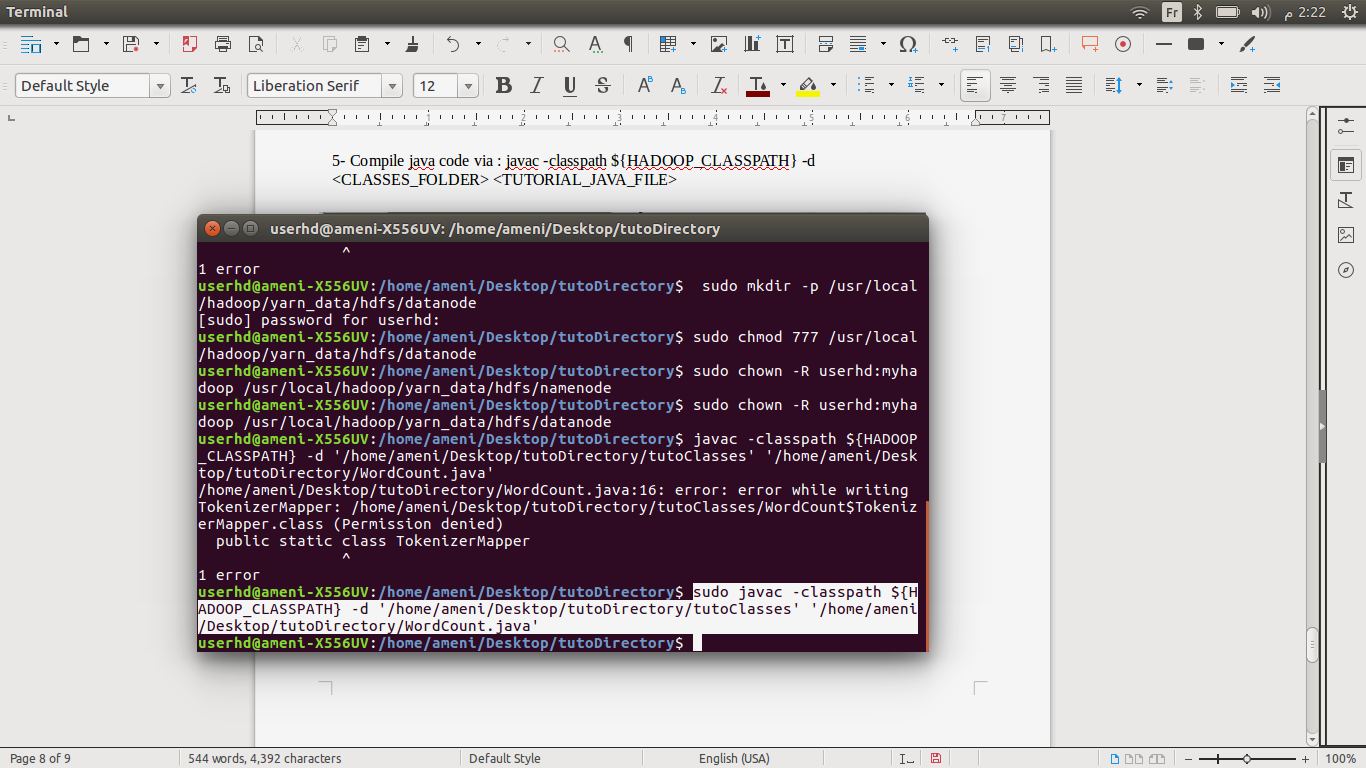


5- Compile java code via :

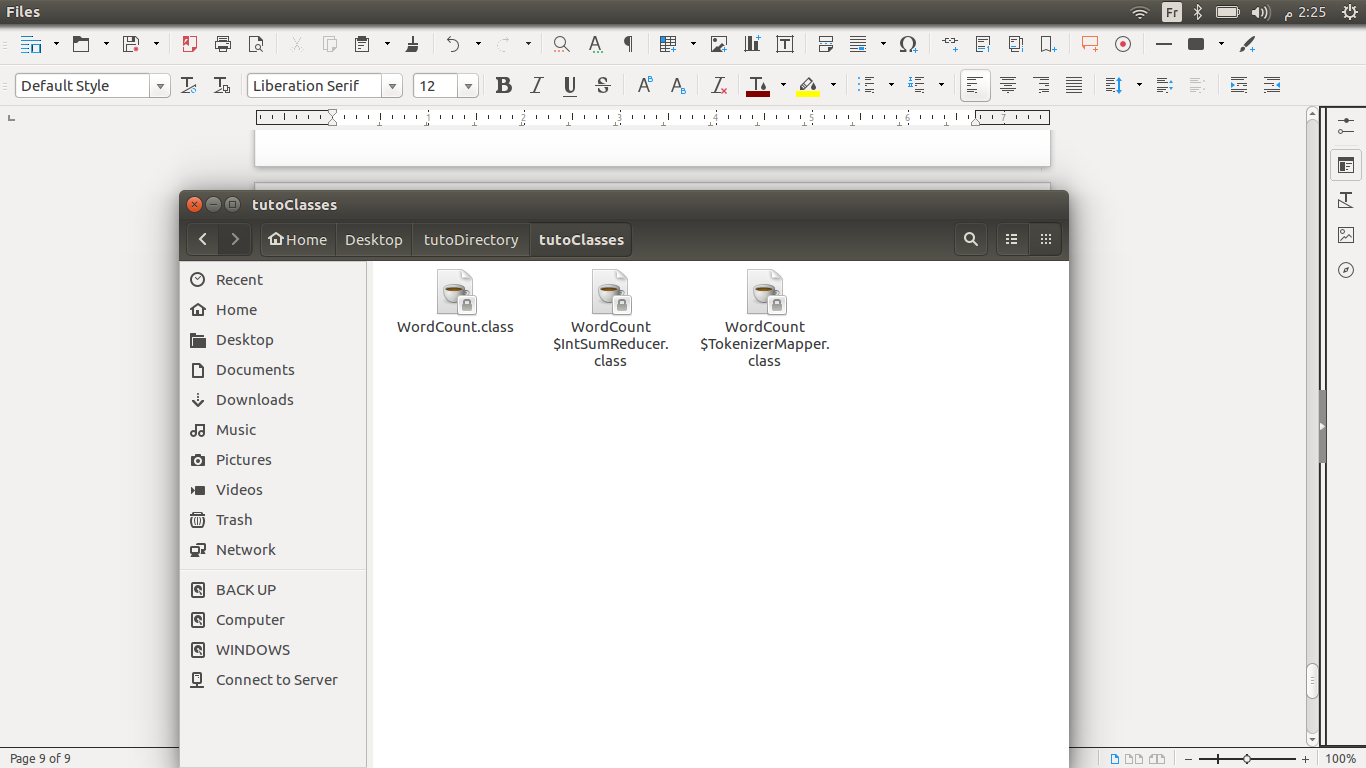
Javac -classpath ${HADOOP\_CLASSPATH} -d <CLASSES\_FOLDER> <TUTORIAL\_JAVA\_FILE>



\*\*\* If you encounter Permission denied error, try with the command sudo

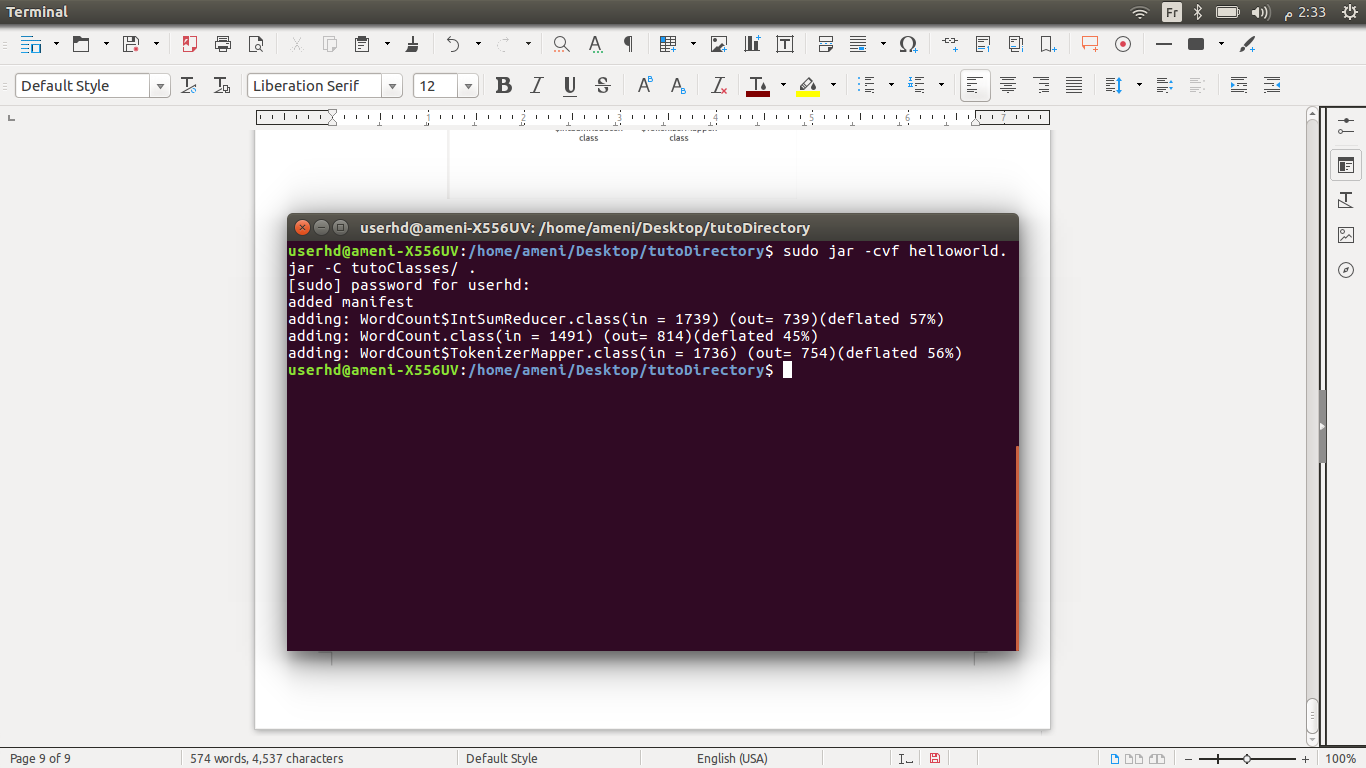


- Now check if the files are generated or not :



6- Put the output files in one jar file :

jar -cvf <JAR\_FILE\_NAME> -C <CLASSES\_FOLDER> .



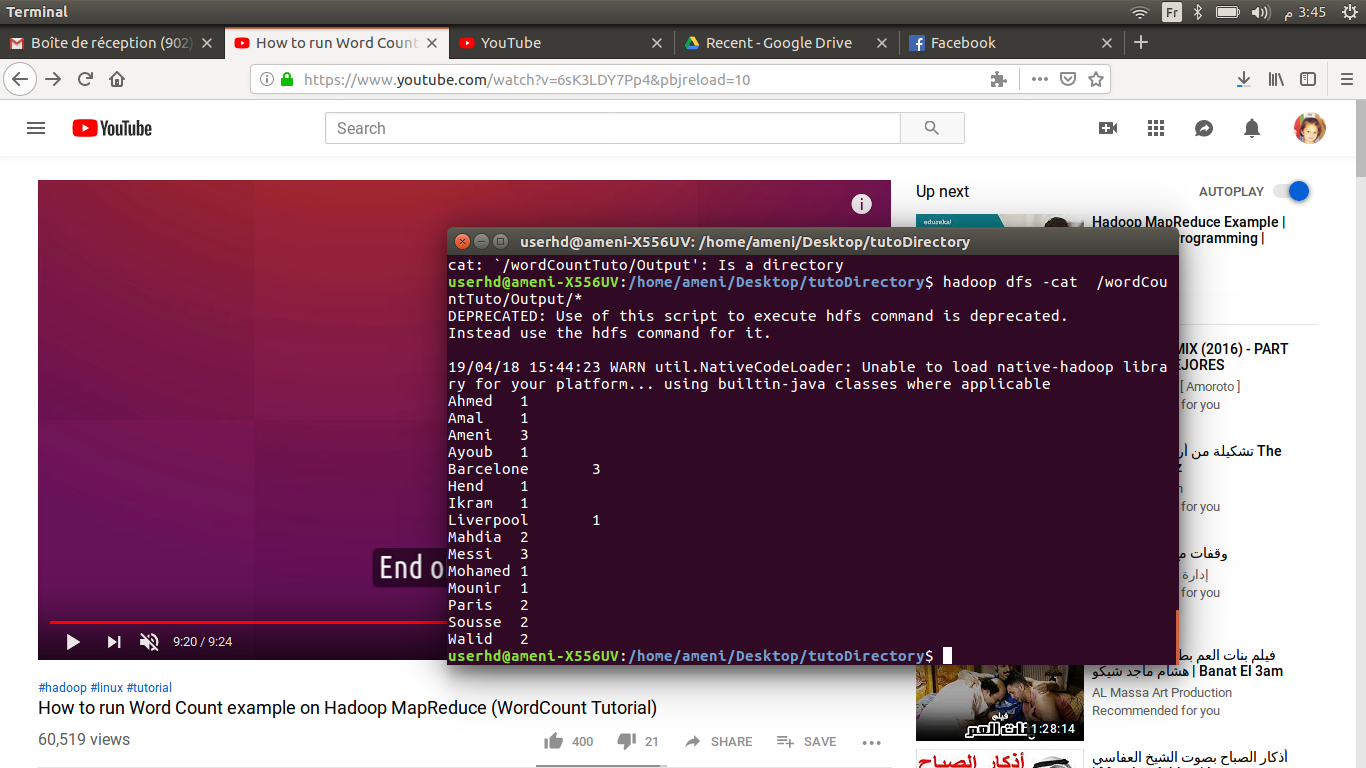
- Now we have the **.jar** file, run it on Hadoop :

hadoop jar <JAR\_FILE> <CLASS\_NAME> <HDFS\_INPUT\_DIRECTORY> <HDFS\_OUTPUT\_DIRECTORY>



- Search for the output : INFO mapreduce.Job: Job job\_1555582256639\_0001 completed successfully

7- Sort the output

 hadoop dfs -cat <HDFS\_OUTPUT\_DIRECTORY>\*

And we are done !!!