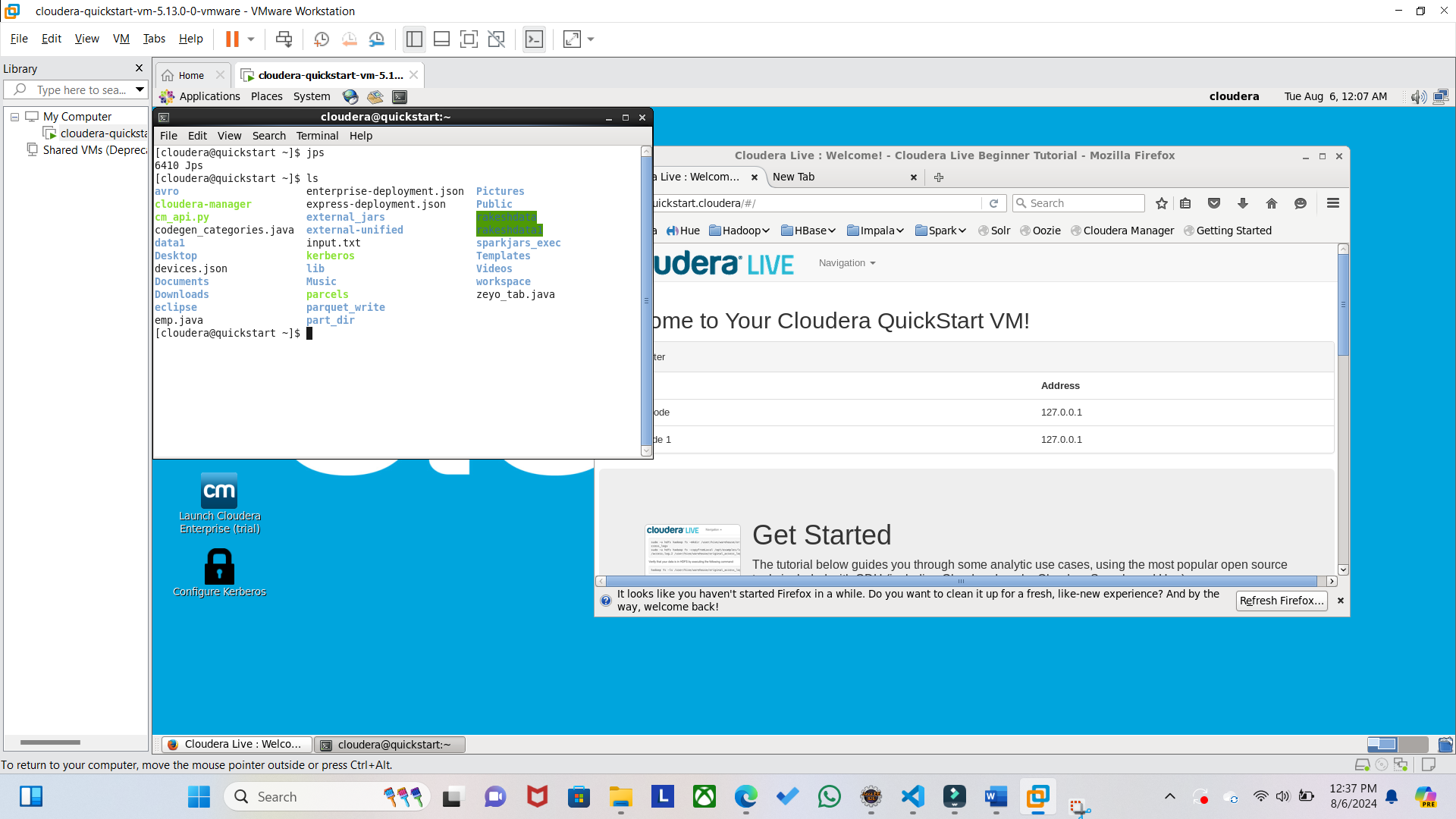
Once the VMWARE workstation and Cloudera are installed and configured, we can practice the below Bigdata/Hadoop concepts.

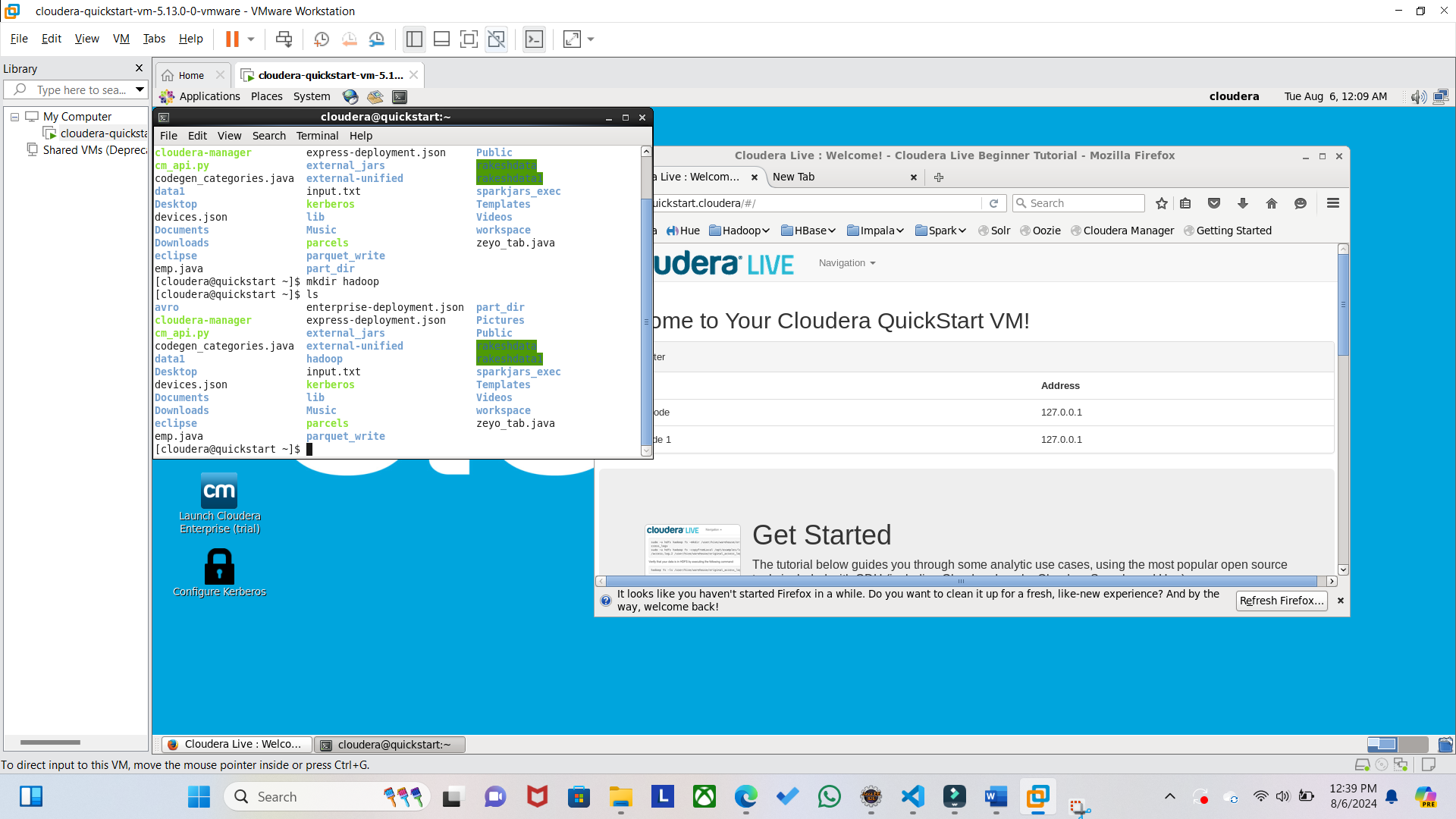
Hadoop:

Basic commands of Hadoop:

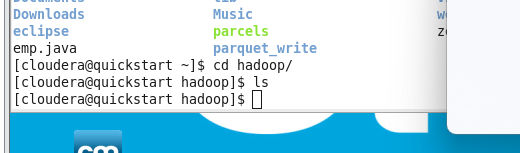
Ls command to list the files/folders



mkdir Hadoop: to create a folder



cd Hadoop/ : to change the directory



Now create a file inside this hadoop folder as below:

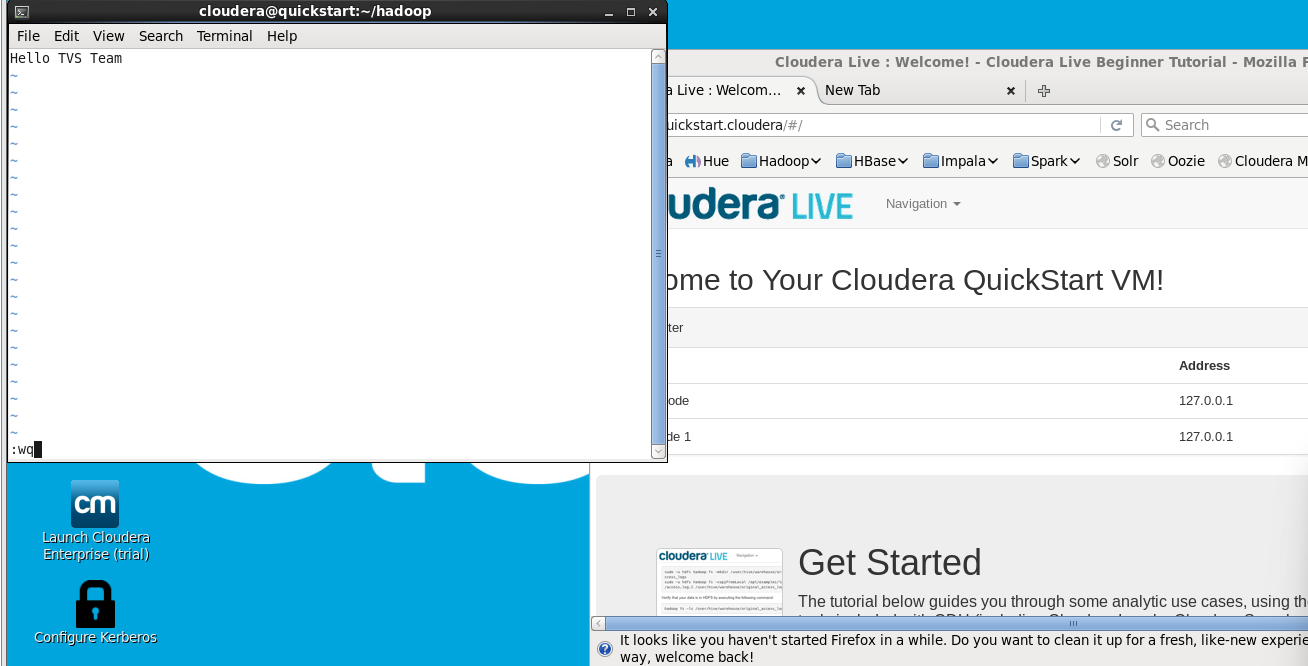
Vi command to create a new file.





Press ‘i’ to go on edit mode in this file and type any content as per your wish.

Once text is typed, press ‘escape’ and then :wq





To see the content of the file do as below: use cat command

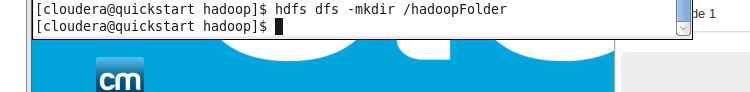


The above are all simple commands.

Now let’s create a folder on our HDFS system (Hadoop file system) and learn the new commands:

hdfs dfs is Hadoop specific commands. (Hadoop Distributed File system)

hdfs dfs -mkdir /hadoopFolder

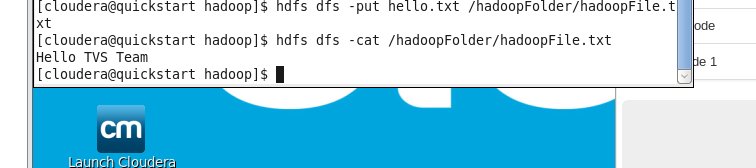


Lets copy the local file hello.txt onto the HDFS system as below:

hdfs dfs -put hello.txt /hadoopFolder/hadoopFile.txt

here we are copying the data of hello.txt onto a new file on HDFS system called hadoopFile.txt

put: command to copy a file from local machine to HDFS system.



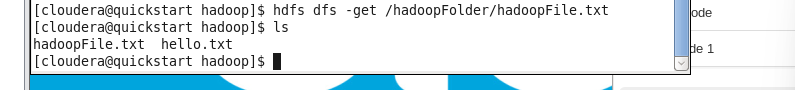
This way we were able to move a local file of our computer onto the HDFS system.

Similarly let’s see how to get a file from a HDFS system onto our computer.

Tranforming files between local and HDFS (vice versa).

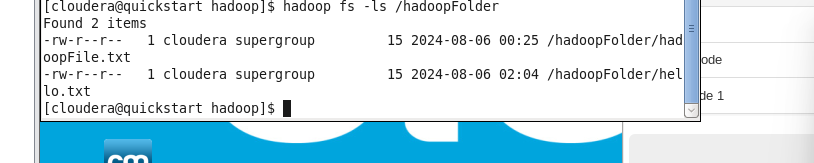
hdfs dfs -get /hadoopFolder/hadoopFile.txt

get command: to move the file from HDFS to local system.



The below commands are similar to that of put command which copies the local file into hdfs system.



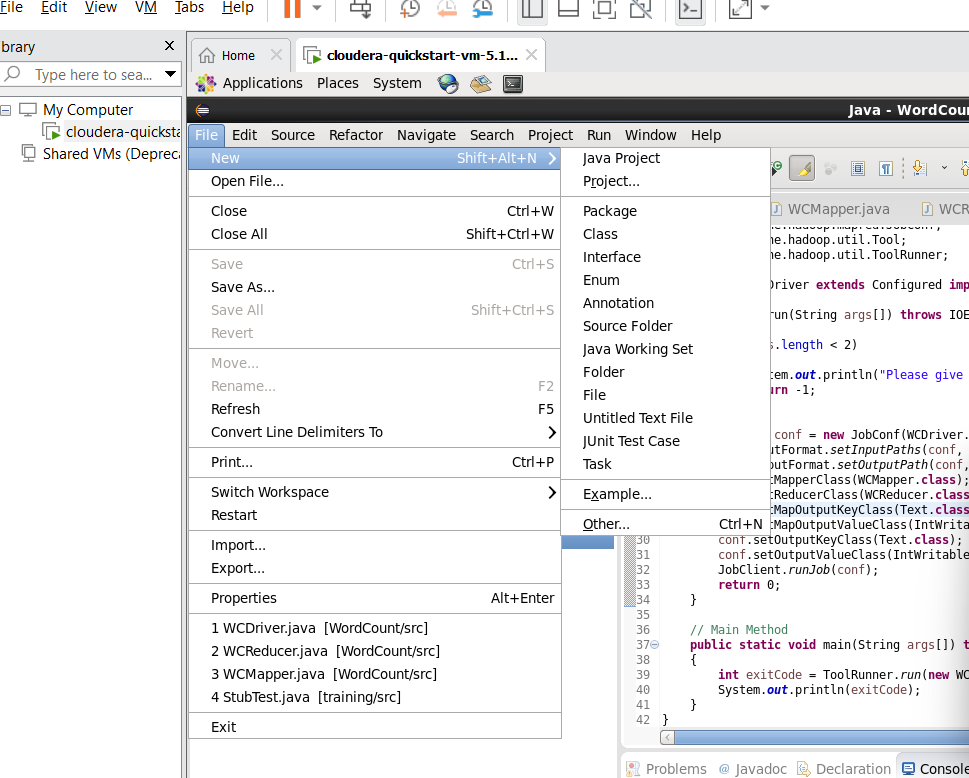


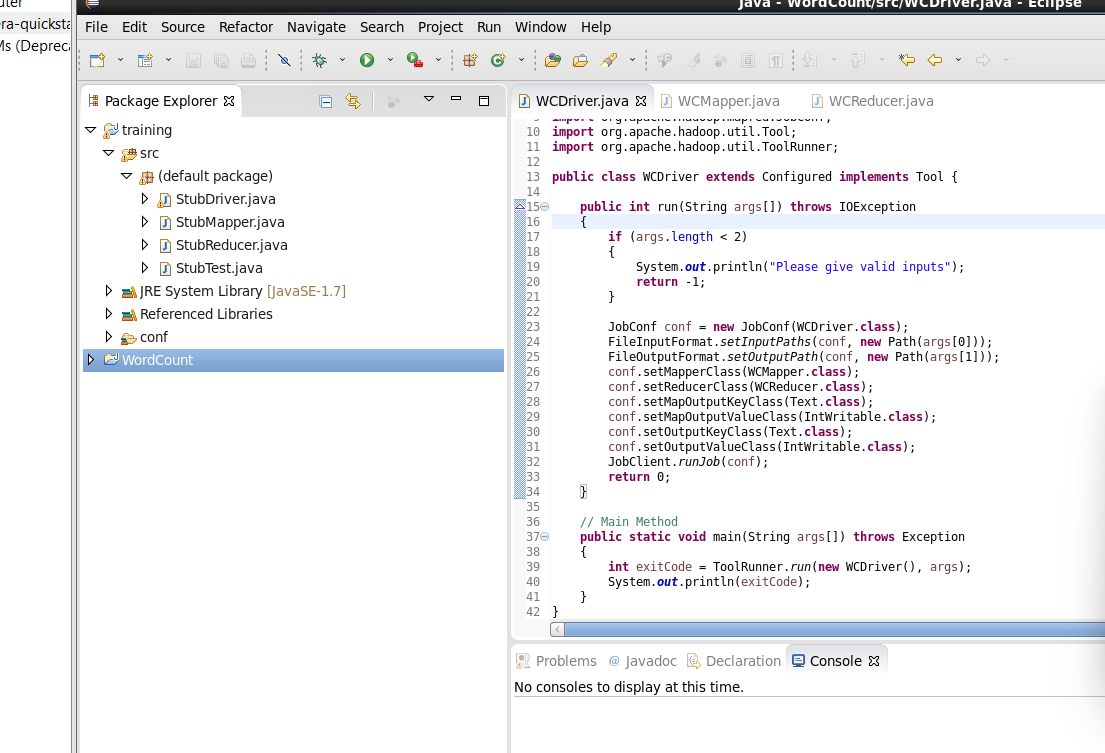
HADOOP MAPREDUCE example using a WordCount program:

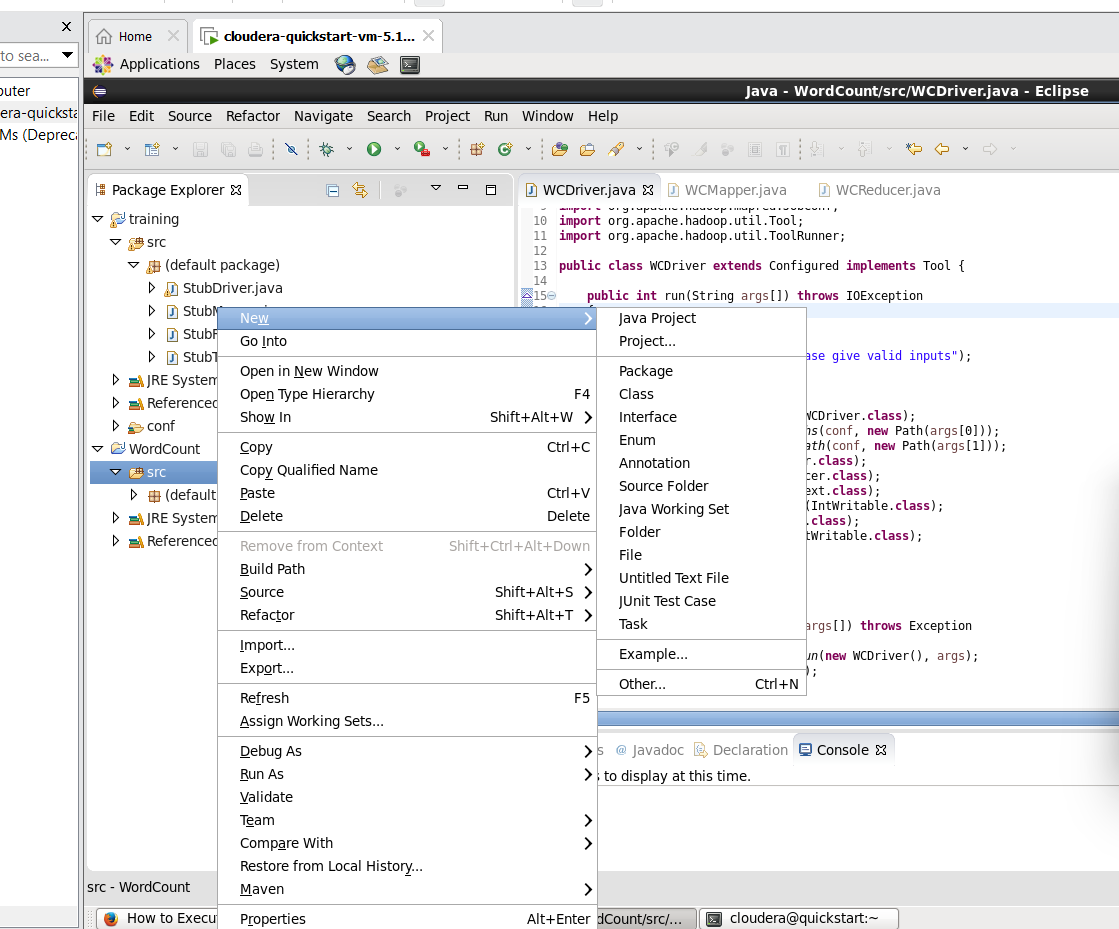
Below steps to be followed in the cloudera machine:

1. Open the eclipse icon which is present on cloudera desktop

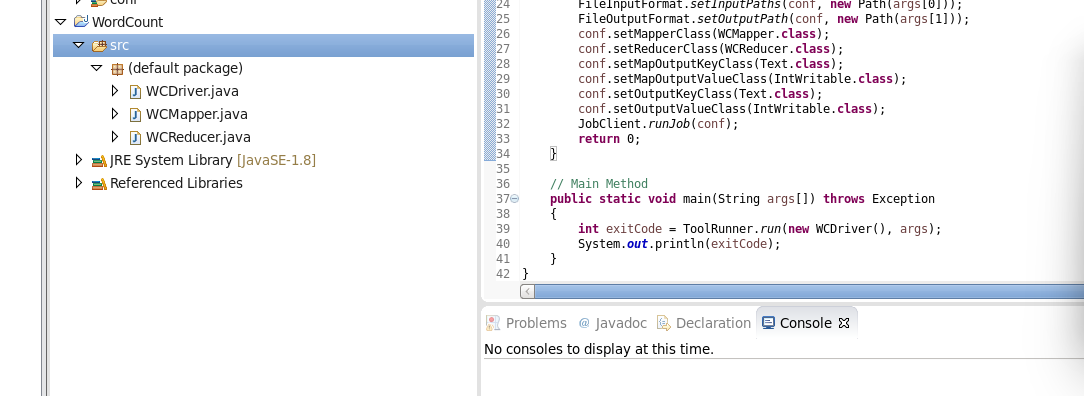
As below select a Java Project and name it as WordCount



1. 
2. Expand the WordCount project and right click on src folder and create 3 java classes as shown below:



1. Name the 3 classes that you create as WCDriver.java, WCMapper.java, WCReducer.java



1. Copy paste the code from here for each of these classes:

**WCMapper.java**

**import** java.io.IOException;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.Mapper;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reporter;

**public** **class** WCMapper **extends** MapReduceBase **implements** Mapper<LongWritable,

Text, Text, IntWritable> {

// Map function

**public** **void** map(LongWritable key, Text value, OutputCollector<Text,

IntWritable> output, Reporter rep) **throws** IOException

{

String line = value.toString();

// Splitting the line on spaces

**for** (String word : line.split(" "))

{

**if** (word.length() > 0)

{

output.collect(**new** Text(word), **new** IntWritable(1));

}

}

}

}

**WCReducer.java**

**import** java.io.IOException;

**import** java.util.Iterator;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reducer;

**import** org.apache.hadoop.mapred.Reporter;

**public** **class** WCReducer **extends** MapReduceBase **implements** Reducer<Text,

IntWritable, Text, IntWritable> {

// Reduce function

**public** **void** reduce(Text key, Iterator<IntWritable> value,

OutputCollector<Text, IntWritable> output,

Reporter rep) **throws** IOException

{

**int** count = 0;

// Counting the frequency of each words

**while** (value.hasNext())

{

IntWritable i = value.next();

count += i.get();

}

output.collect(key, **new** IntWritable(count));

}

}

**WCDriver.java**

**import** java.io.IOException;

**import** org.apache.hadoop.conf.Configured;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.FileInputFormat;

**import** org.apache.hadoop.mapred.FileOutputFormat;

**import** org.apache.hadoop.mapred.JobClient;

**import** org.apache.hadoop.mapred.JobConf;

**import** org.apache.hadoop.util.Tool;

**import** org.apache.hadoop.util.ToolRunner;

**public** **class** WCDriver **extends** Configured **implements** Tool {

**public** **int** run(String args[]) **throws** IOException

{

**if** (args.length < 2)

{

System.***out***.println("Please give valid inputs");

**return** -1;

}

JobConf conf = **new** JobConf(WCDriver.**class**);

FileInputFormat.*setInputPaths*(conf, **new** Path(args[0]));

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

conf.setMapperClass(WCMapper.**class**);

conf.setReducerClass(WCReducer.**class**);

conf.setMapOutputKeyClass(Text.**class**);

conf.setMapOutputValueClass(IntWritable.**class**);

conf.setOutputKeyClass(Text.**class**);

conf.setOutputValueClass(IntWritable.**class**);

JobClient.*runJob*(conf);

**return** 0;

}

// Main Method

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

{

**int** exitCode = ToolRunner.*run*(**new** WCDriver(), args);

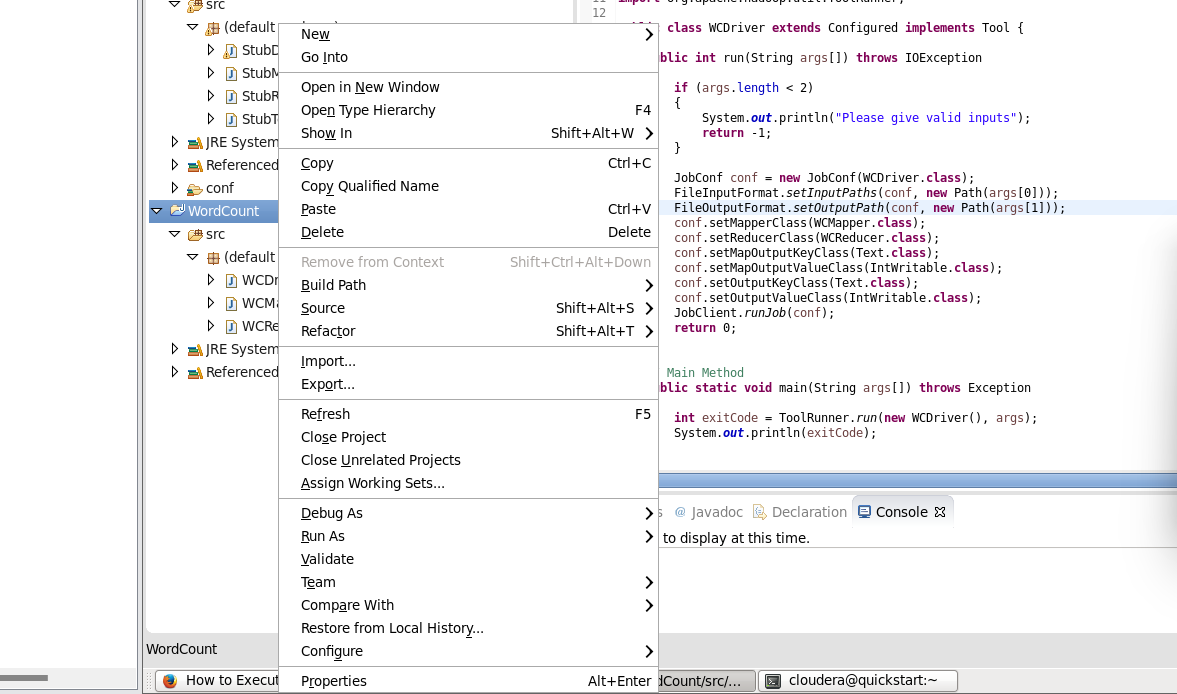
System.***out***.println(exitCode);

}

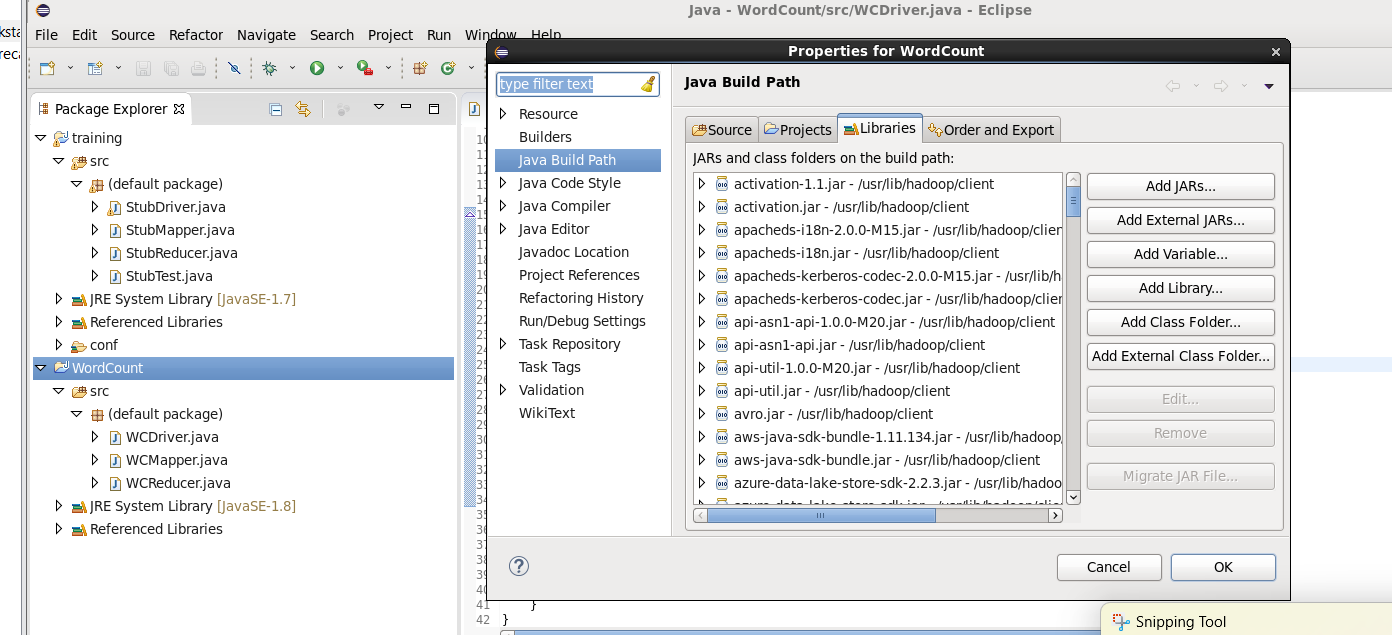
}

1. Its important to add dependent libraries onto the build path as below:

Right click on WordCount project as below and click on **properties**:

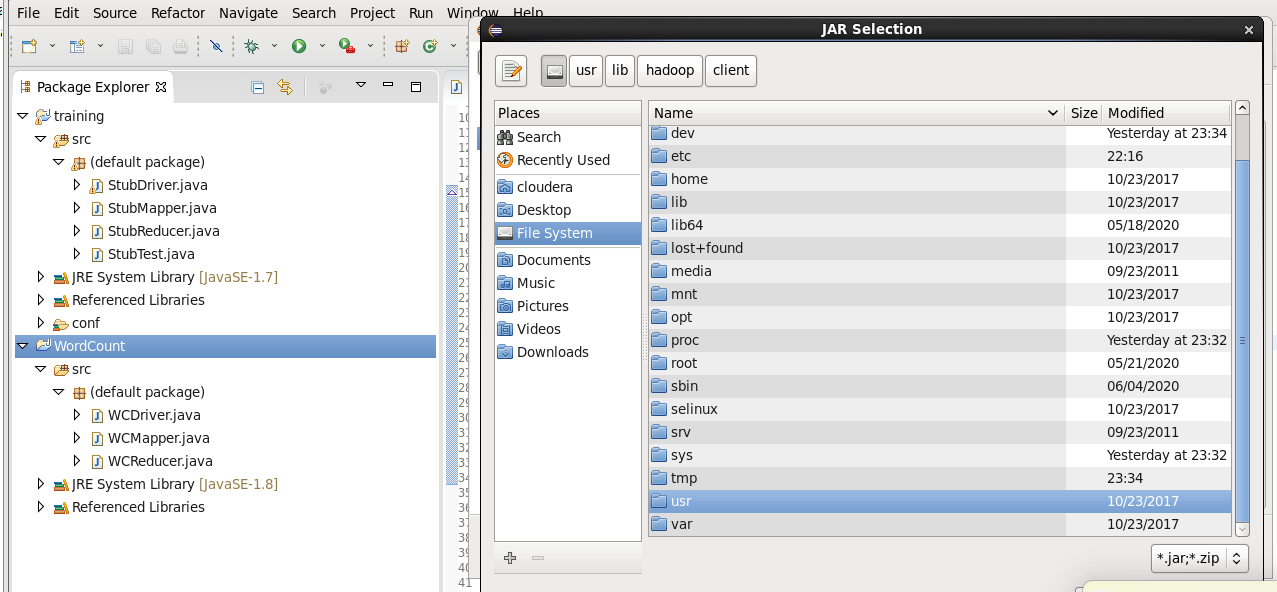


Once clicked on Properties – select – Java Build path – ADD External Jars

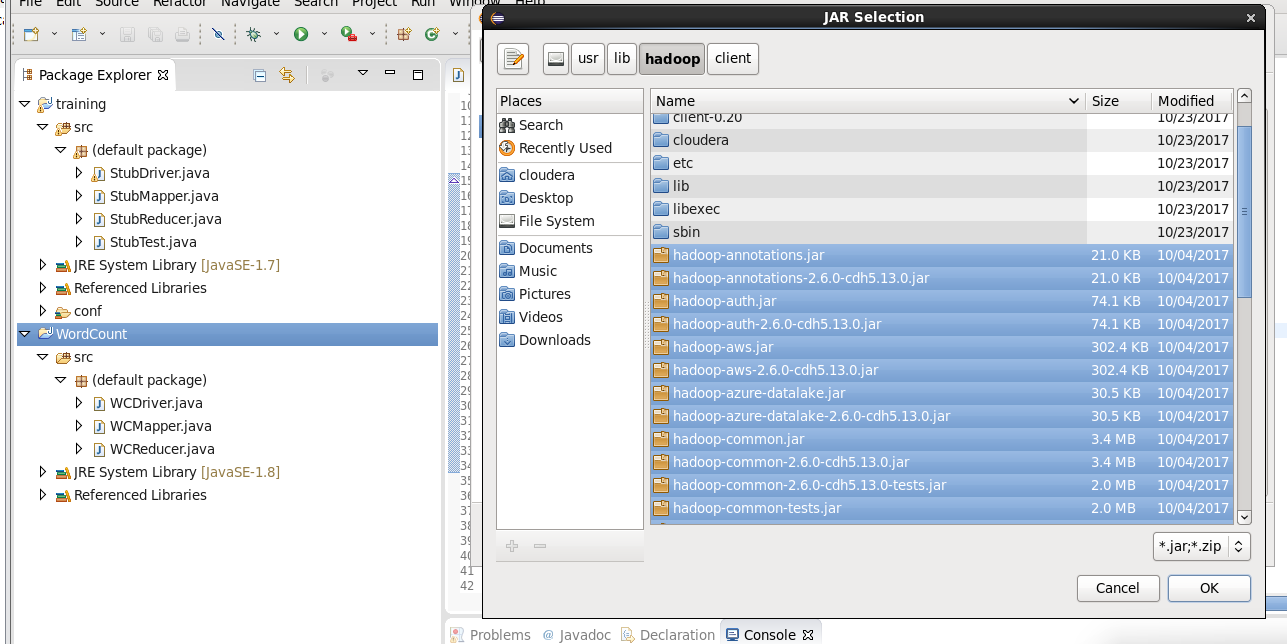


Once clicked on Add external Jars you will see as below:

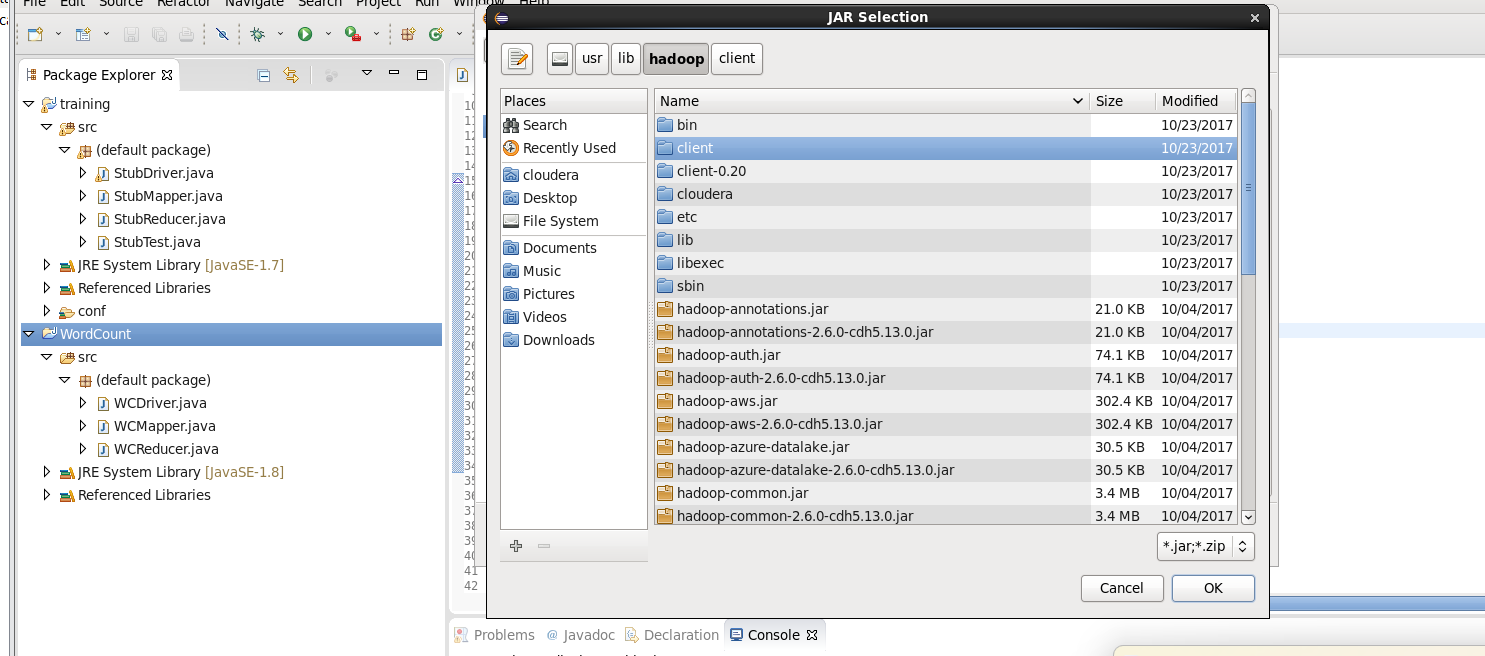
Go to File system – usr – lib – Hadoop

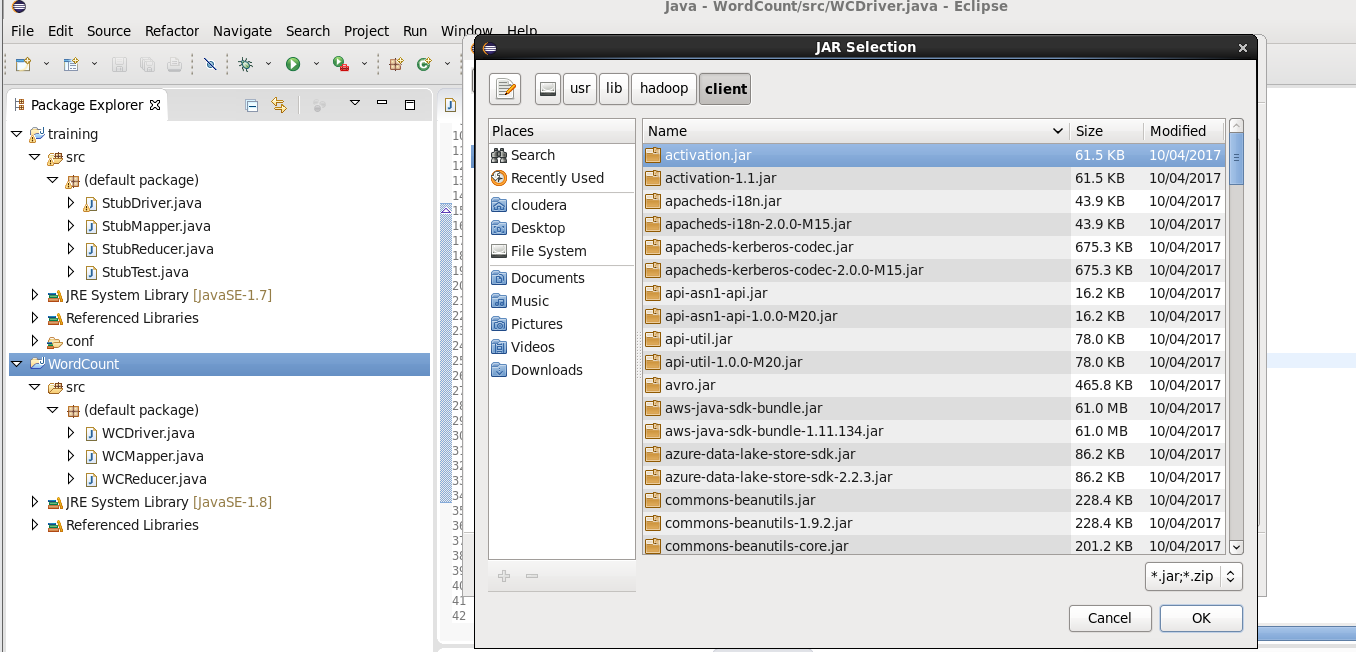


Once landed in Hadoop – select all the jar files as shown below and click ok.



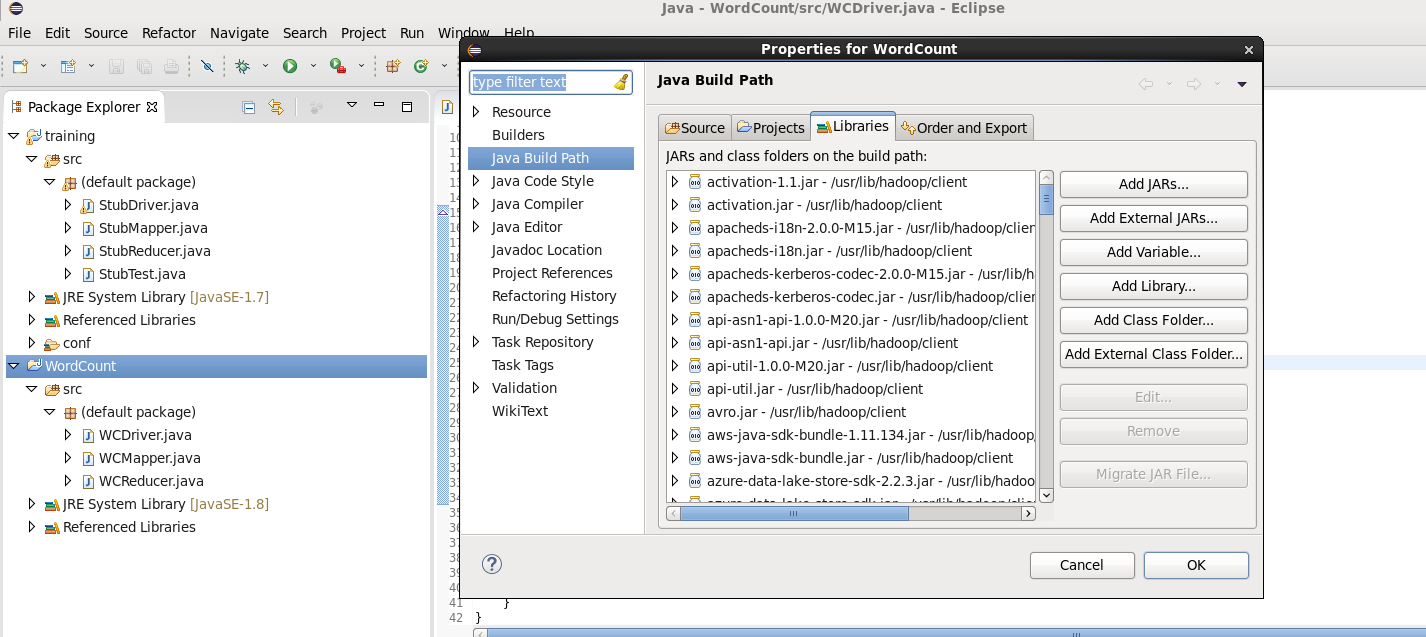
Again click on Add external jars , this time click on client folder under the Hadoop folder as below:





Select all the jar files here and finally click ok.

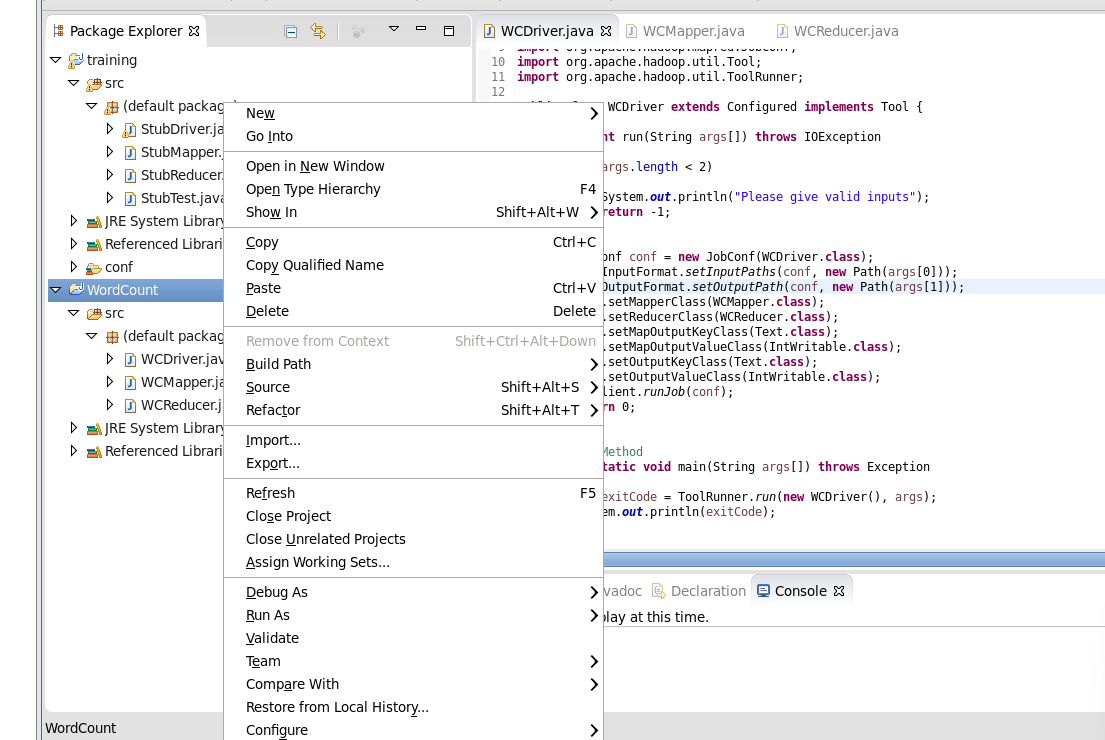
Finally once all the jar files are added as below, click ok. Now you should not see any errors in all those 3 class files (WCMapper, WCReducer, WCDriver)

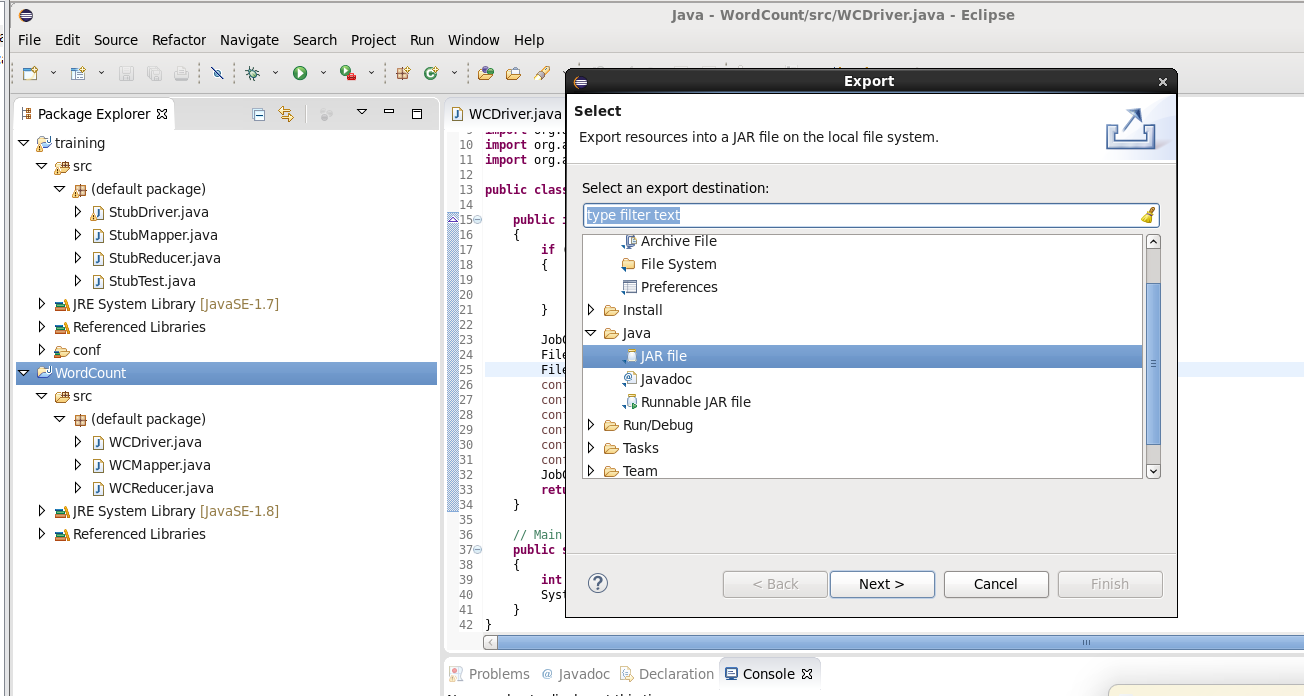


7)

Its time now to export this WordCount project as a jar file following the below steps:

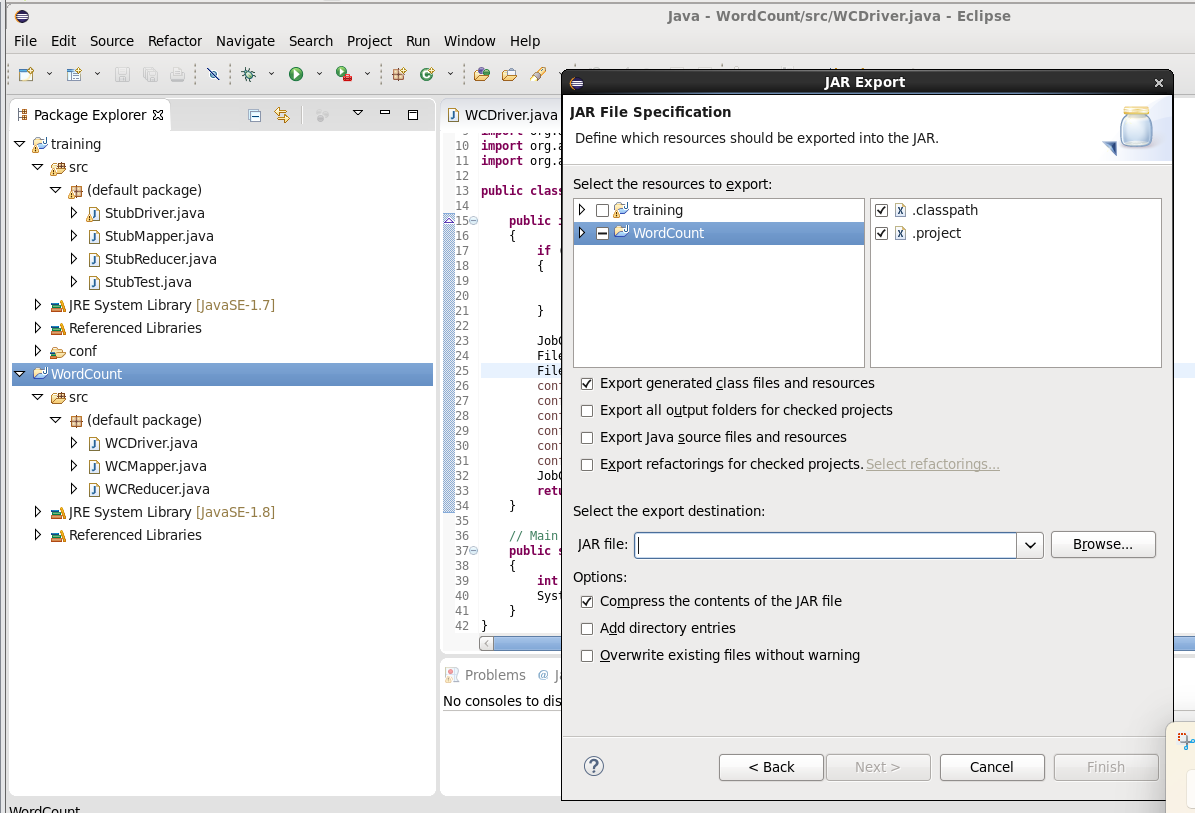
Right click on the WordCount project and select Export option:



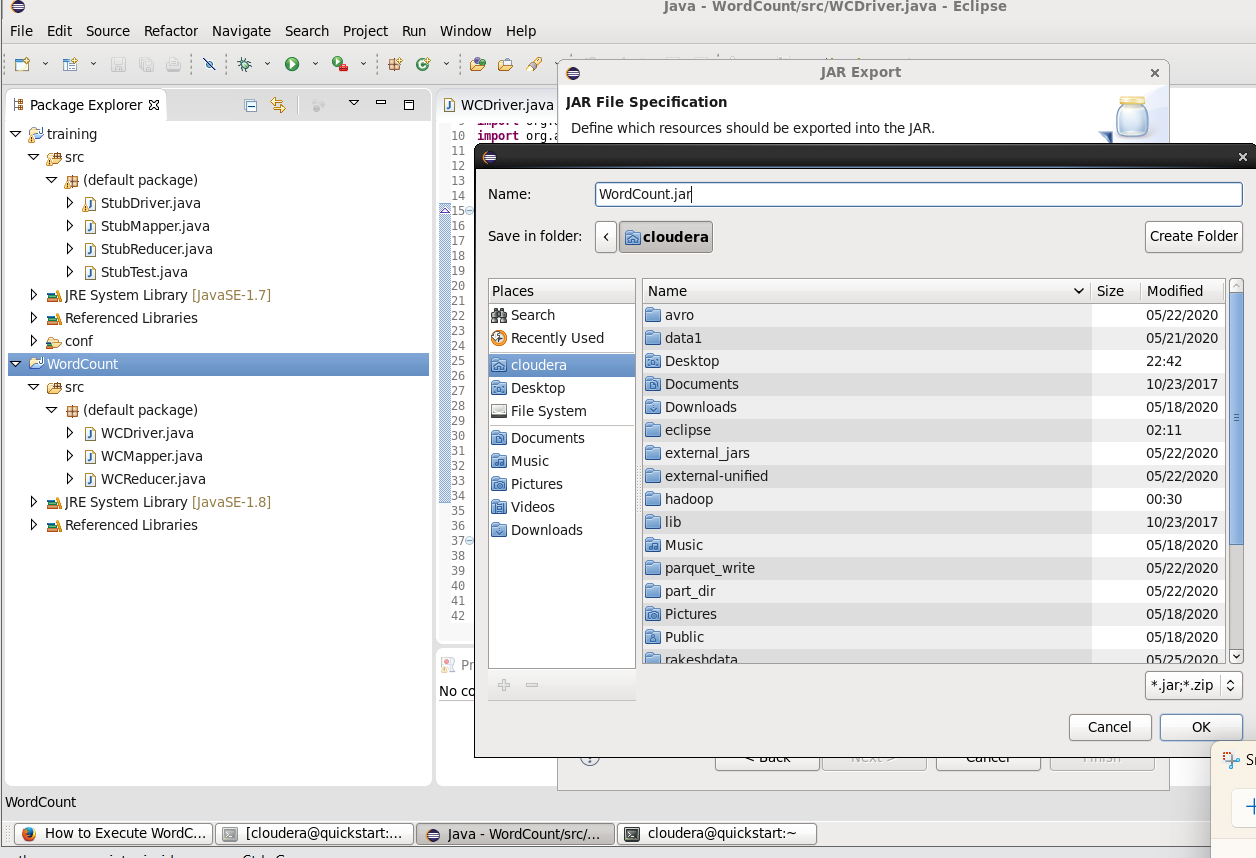


Select the jar option as above, click next:

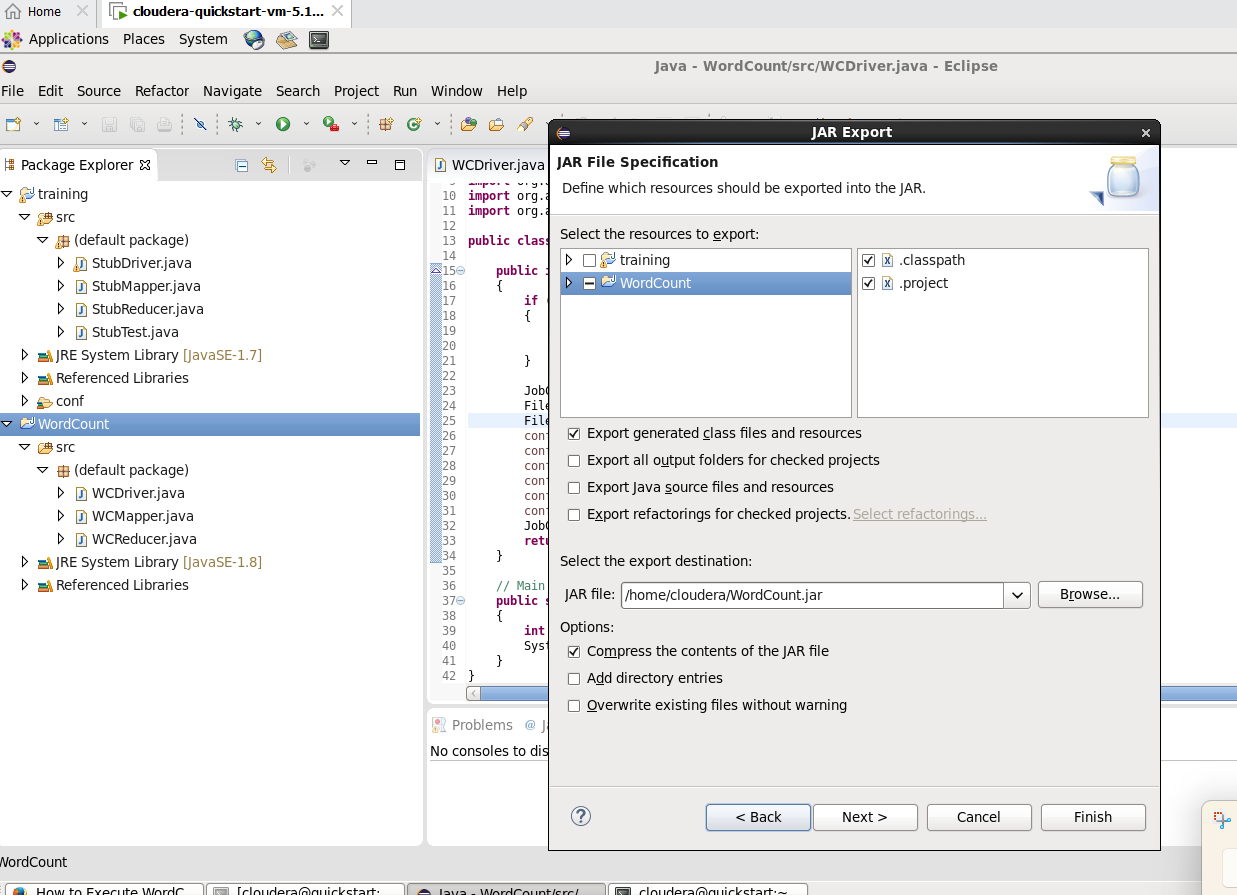
Browse for the destination:



Select cloudera folder and name the file as WordCount.jar and click ok

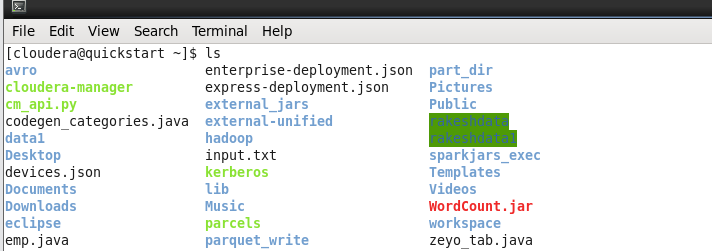


Finally click on finish



8)

Now open the cloudera terminal and check if WordCount.jar is seen:

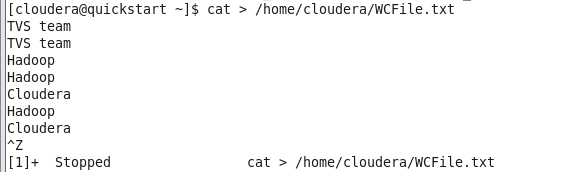


You can see the WordCount.jar!

9) We need to now create a simple text file for Word counting purpose:

Create a WCFile.txt using cat command as below.

Once done with entering the text , press ctrl-z to exit.



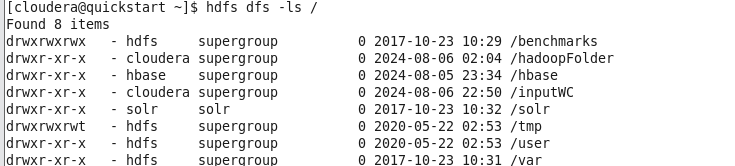
10)

Now lets create a folder in the hdfs system to store this WCFile.txt

Named the folder as inputWC

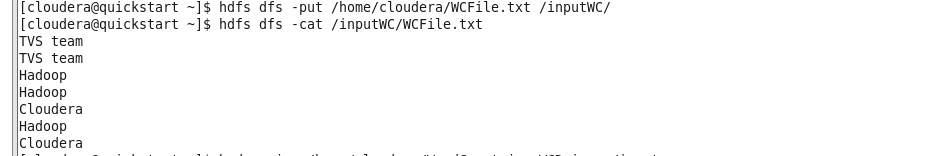


Execute the below command to cross check if the folder is created:



11) Lets put this WCFile.txt onto the HDFS system:

Use the put command as below and also cross check the data using cat command



12) Now lets execute the jar file for wordCounting

The command to execute the jar file

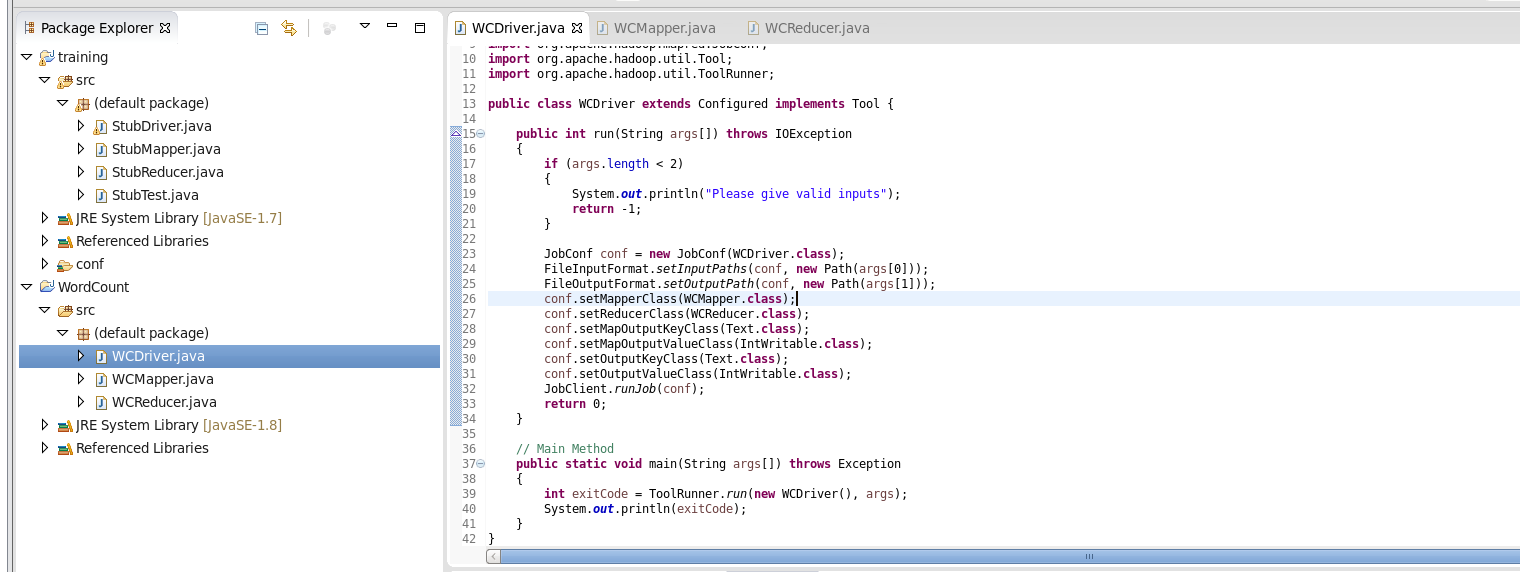
hadoop jar /home/cloudera/WordCount.jar WCDriver /inputWC/WCFile.txt /outWC

Explaining the above command:

Hadoop jar -> this is a command to execute the jar file

/home/cloudera/WordCount.jar -> this is the location where our jar is located. WordCount.jar is the one which had exported from eclipse.

WCDriver -> This is a main java class which will be executed. [Refer the below code in eclipse]

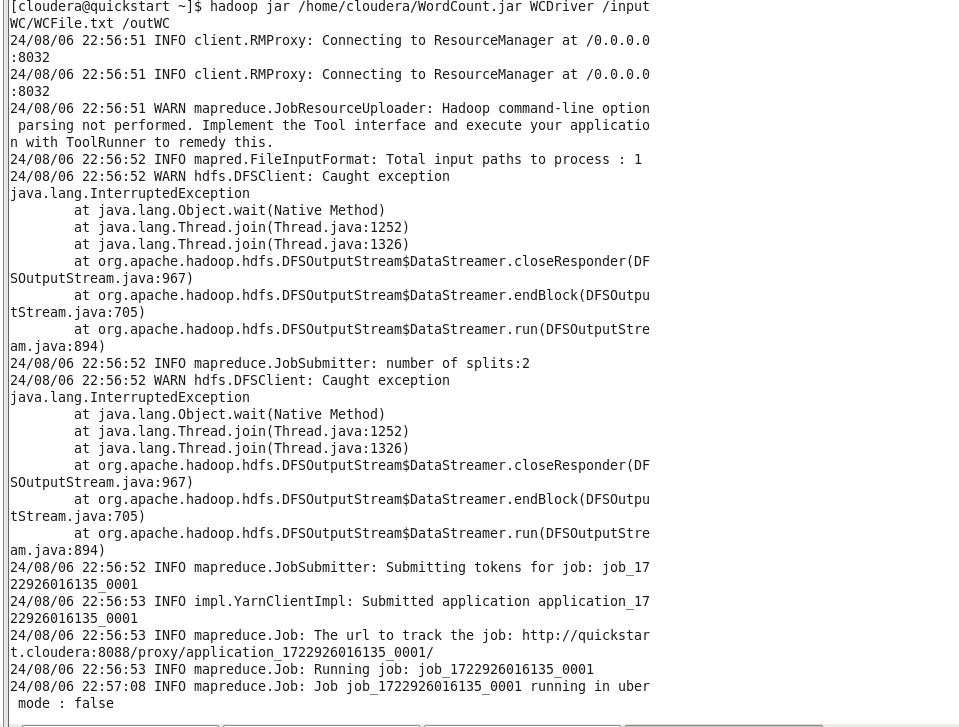


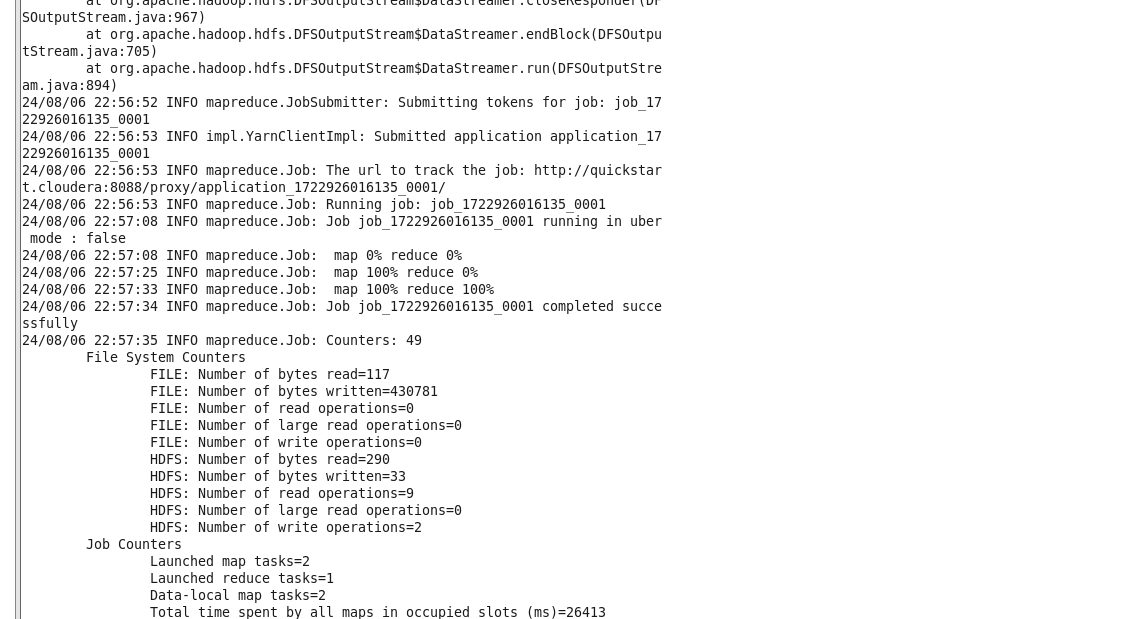
/inputWC/WCFile.txt -> this is the file on hdfs which has the data to be counted.

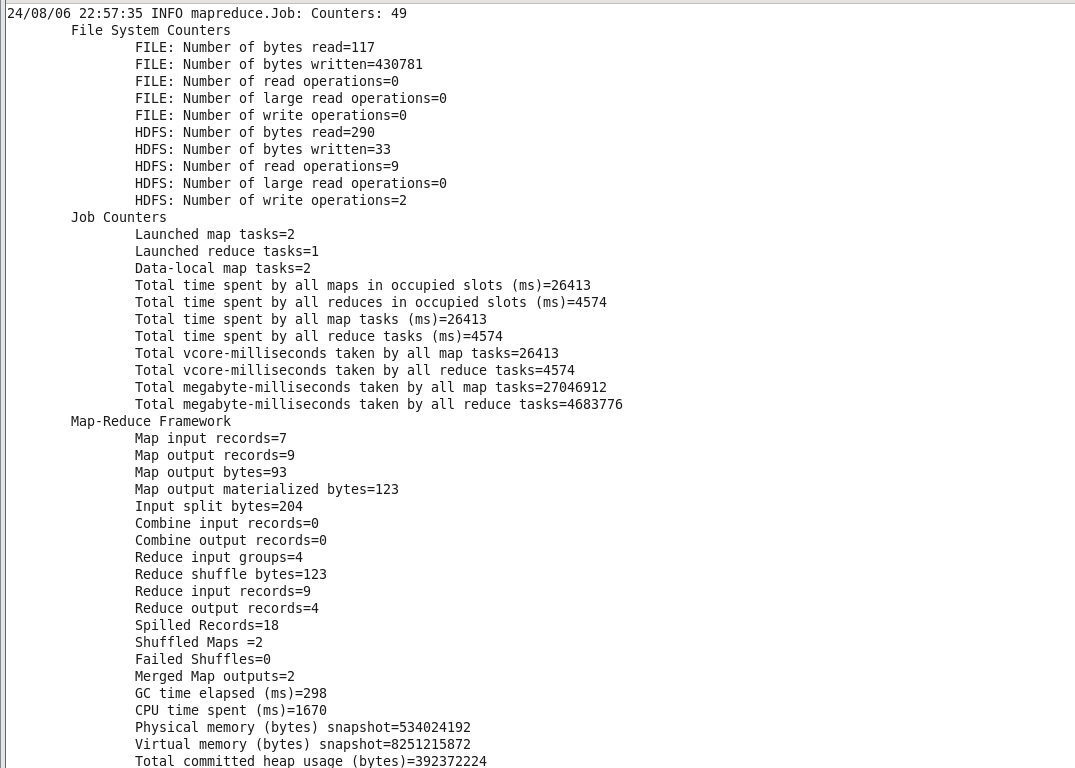
/outWC -> this folder gets created once the below command is processed and stores the output.



This command executes as below:



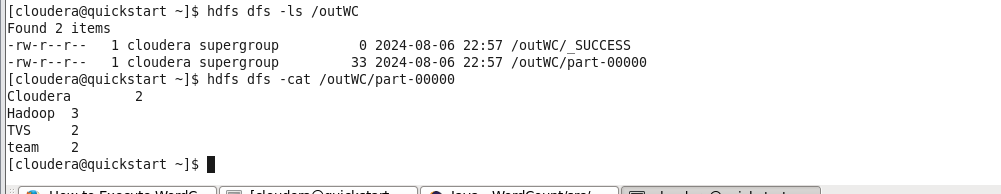






This may take sometime and if you observe the terminal it is working based on the map reduce algorithm.

13) To check the output execute as below



The WordCount.jar has successfully executed and has counted the words as seen above.