**Software Development for Cloud Computing**

**Assignment 4**

**Name - Ronak Shah**

**Student Id - 4949773**

**Unit Code - COS80001**

**Title - Programming with Amazon Elastic Map Reduce (Submission for Assignment 4)**

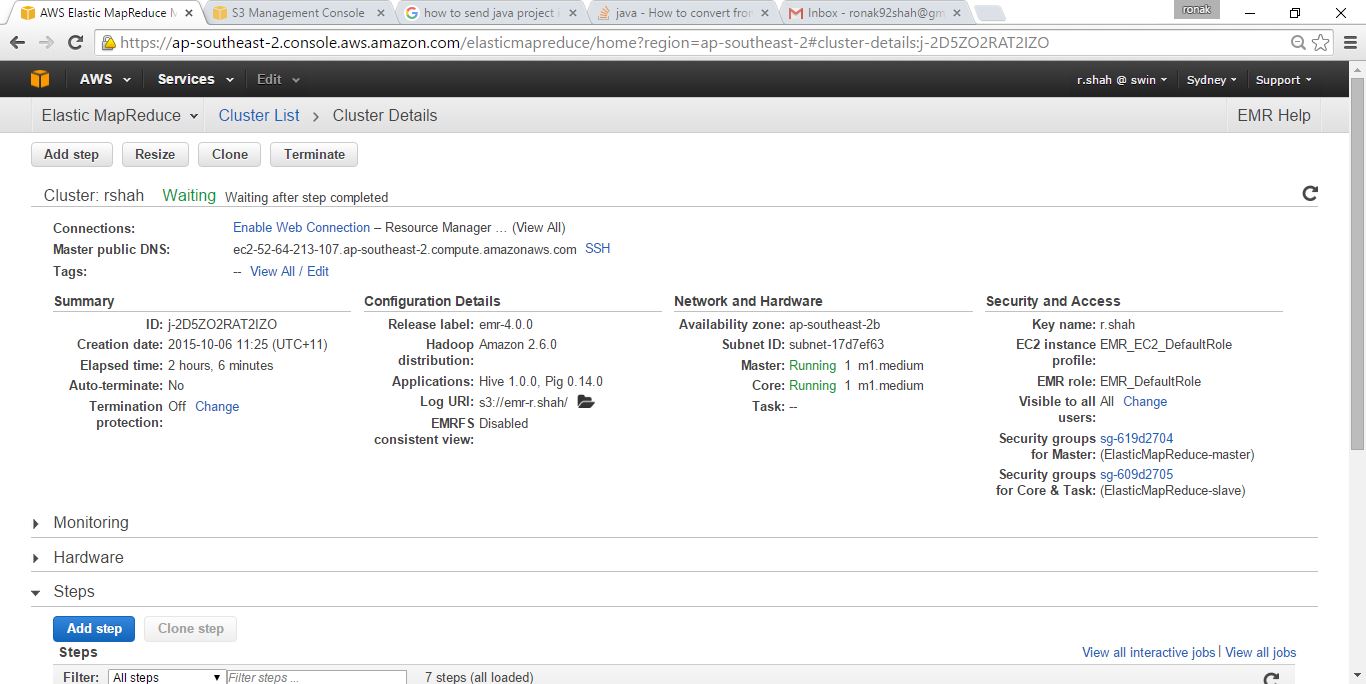
****

Figure : Cluster details

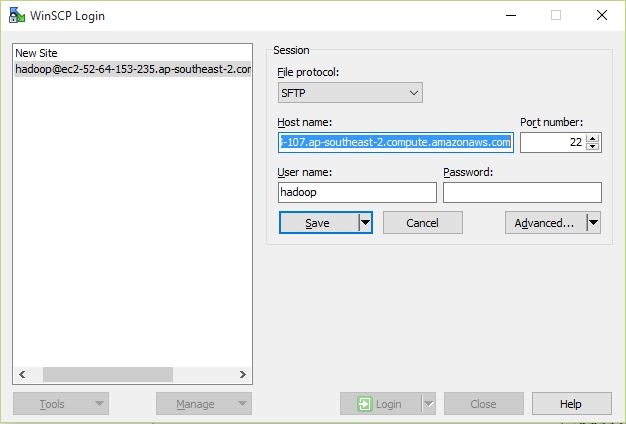
****

Figure : Winscp login

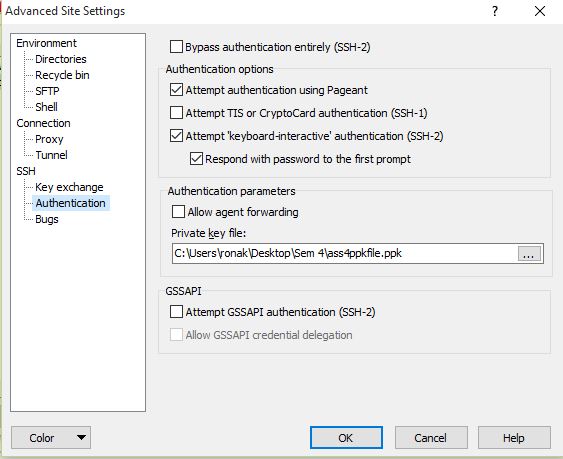
****

Figure : Winscp Authentication .ppk file

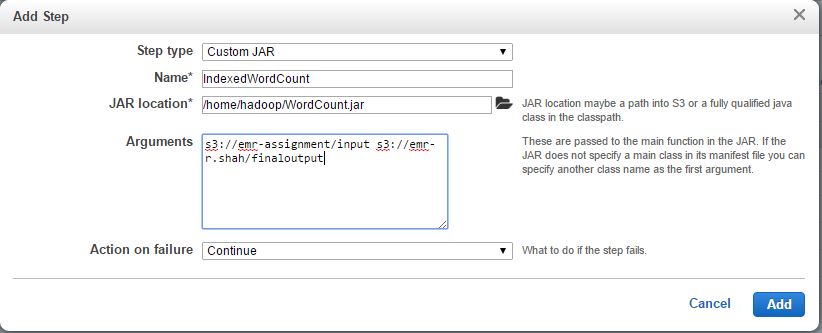
****

Figure : Adding Steps

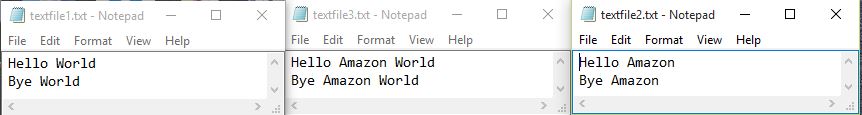
****

Figure : Input file

****

Figure : Output of Word Count

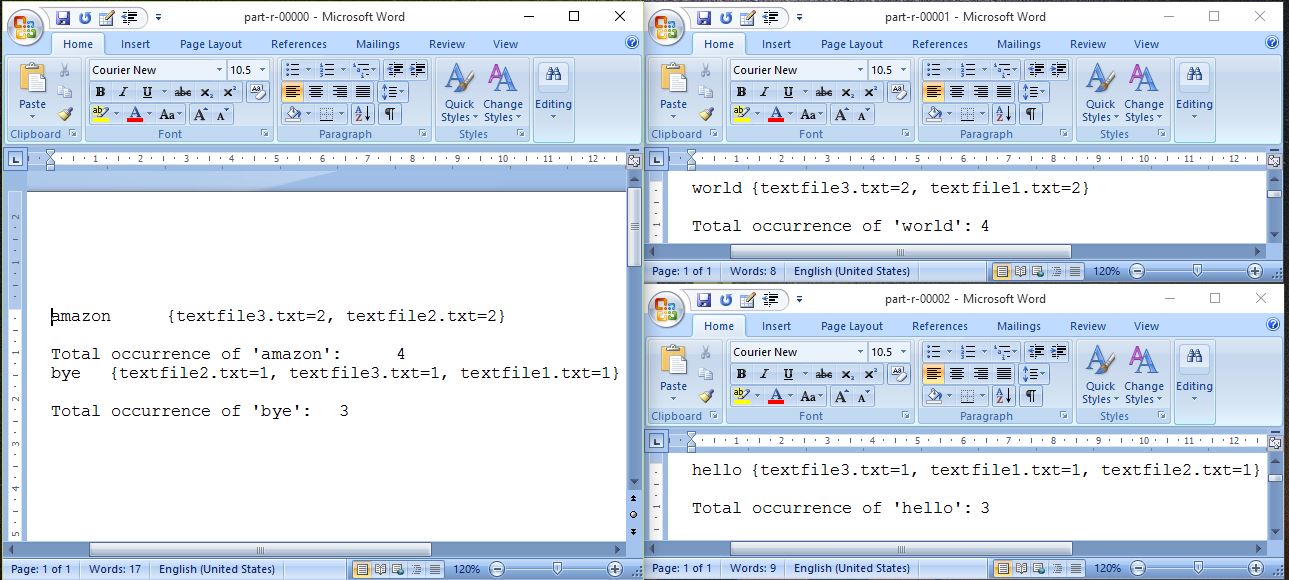
****

Figure : Output of Indexed Word Count

**Source Code:**

**WordCount.java**

import org.apache.hadoop.fs.Path;

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

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

import org.apache.hadoop.mapreduce.\*;

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

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

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

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

public class WordCount {

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

Configuration conf = new Configuration();

Job job = new Job(conf, "WordCount");

job.setJarByClass(WordCount.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setNumReduceTasks(3);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

job.waitForCompletion(true);

}

}

**Map.java**

import java.io.IOException;

import java.util.\*;

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

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

public class Map extends Mapper<LongWritable, Text, Text, Text> {

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

private Text word = new Text();

private String pattern= "^[a-z][a-z0-9]\*$";

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{

//getting filenames

InputSplit inputSplit = context.getInputSplit();

String fileName = ((FileSplit) inputSplit).getPath().getName();

// getting values

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

word.set(tokenizer.nextToken());

String stringWord = word.toString().toLowerCase();

//Matching

if (stringWord.matches(pattern)){

//generating key-value pair

// instead of sending number we are sending the filename associated with the word.

context.write(new Text(stringWord), new Text(fileName));

//context.write(new Text(stringWord), one);

}

}

}

}

**Reduce.java**

**import** java.io.IOException;

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

**import** org.apache.hadoop.mapreduce.\*;

**public** **class** Reduce **extends** Reducer<Text, Text, Text, Text>

{

**public** **void** reduce(Text txt, Iterable<Text> val, Context contxt)

**throws** IOException, InterruptedException

{

**int** count = 0; //to count how many times word appears in the file

**int** total = 0;

**boolean** bolChk=**false**;

String file="";//to store filename we counting

String txtStr="{"; // includes output

**for** (Text value : val)

{

**if**(!bolChk)

{

file=value.toString();

bolChk=**true**;

}

**if** (file.equals(value.toString())){ // chk if similar

count=count+1; //increase the count

total = total + 1;

}

**else**

{

txtStr+=file + "="+count +", "; // store in string

file=value.toString(); // as value is object so to convert in string.

count=1;

total = total + 1;

}

}

txtStr+= file + "="+count +"} \n"; //output pattern

contxt.write(txt, **new** Text(txtStr));

String tot = "" + total; // convert to string from int

//as method write accepts (Text,Text) displaying it and converting to Text

contxt.write(**new** Text ("Total occurrence of '" + txt + "':"), **new** Text (tot));

}

}