Project: calculate Pi

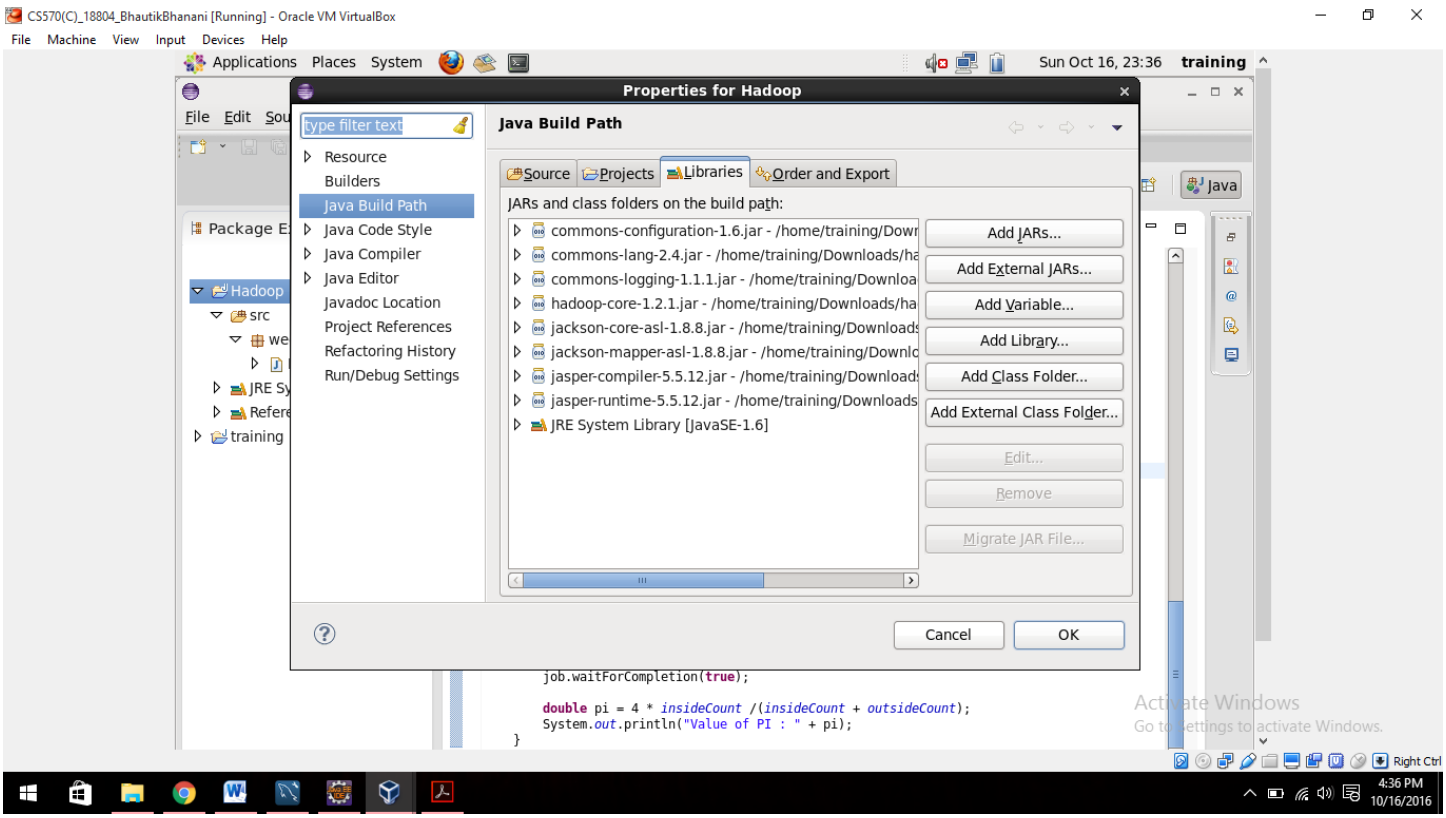
Solution:

Step 1: Open Cloudera and open Eclipse in it. Create one class under Hadoop project and name it ‘Pi’.

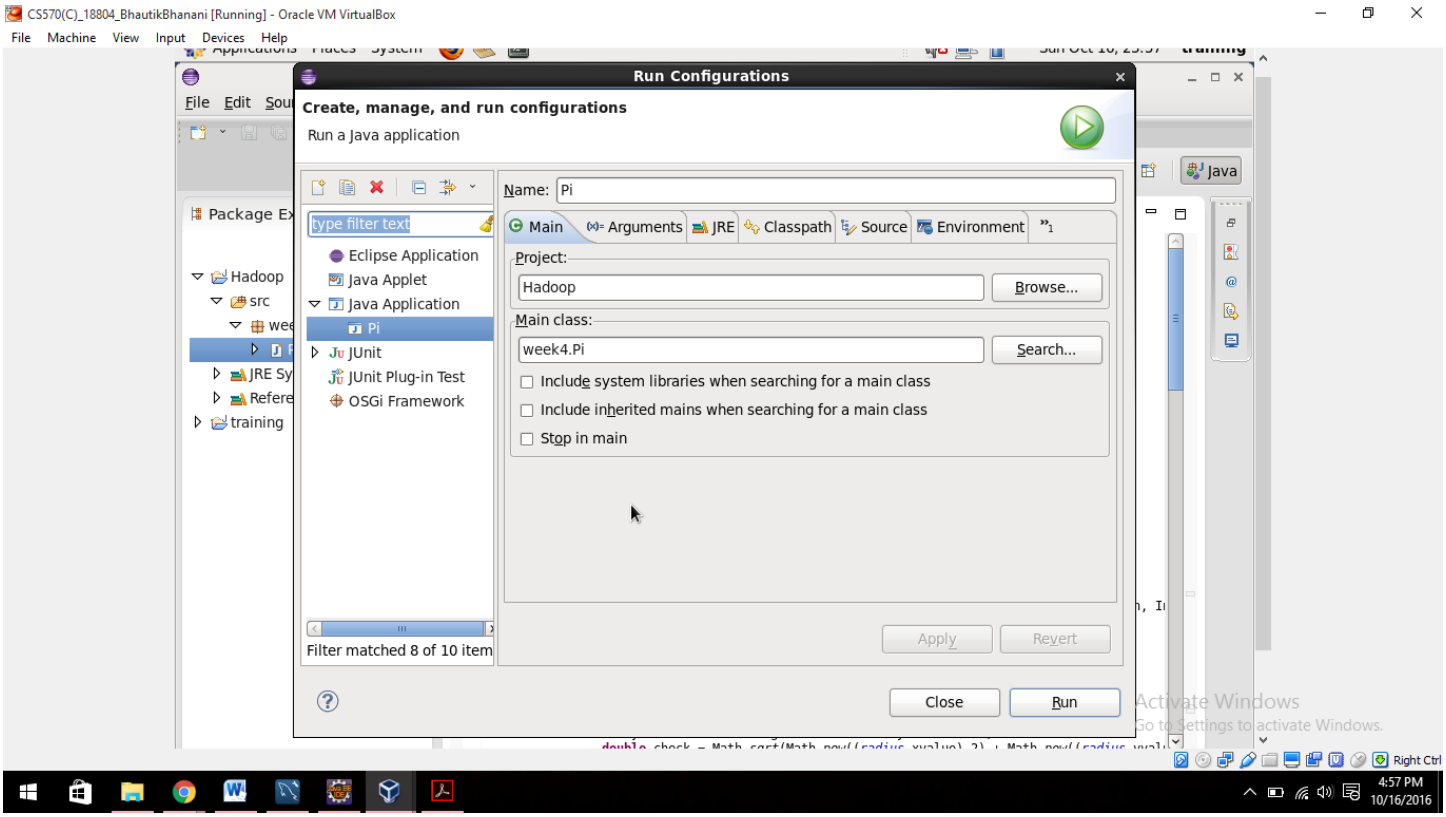
Step 2: Create file Pi.java and write a bellow code to calculate ‘Pi’ value and create driver function and configure job.

```
// copy from repository
```

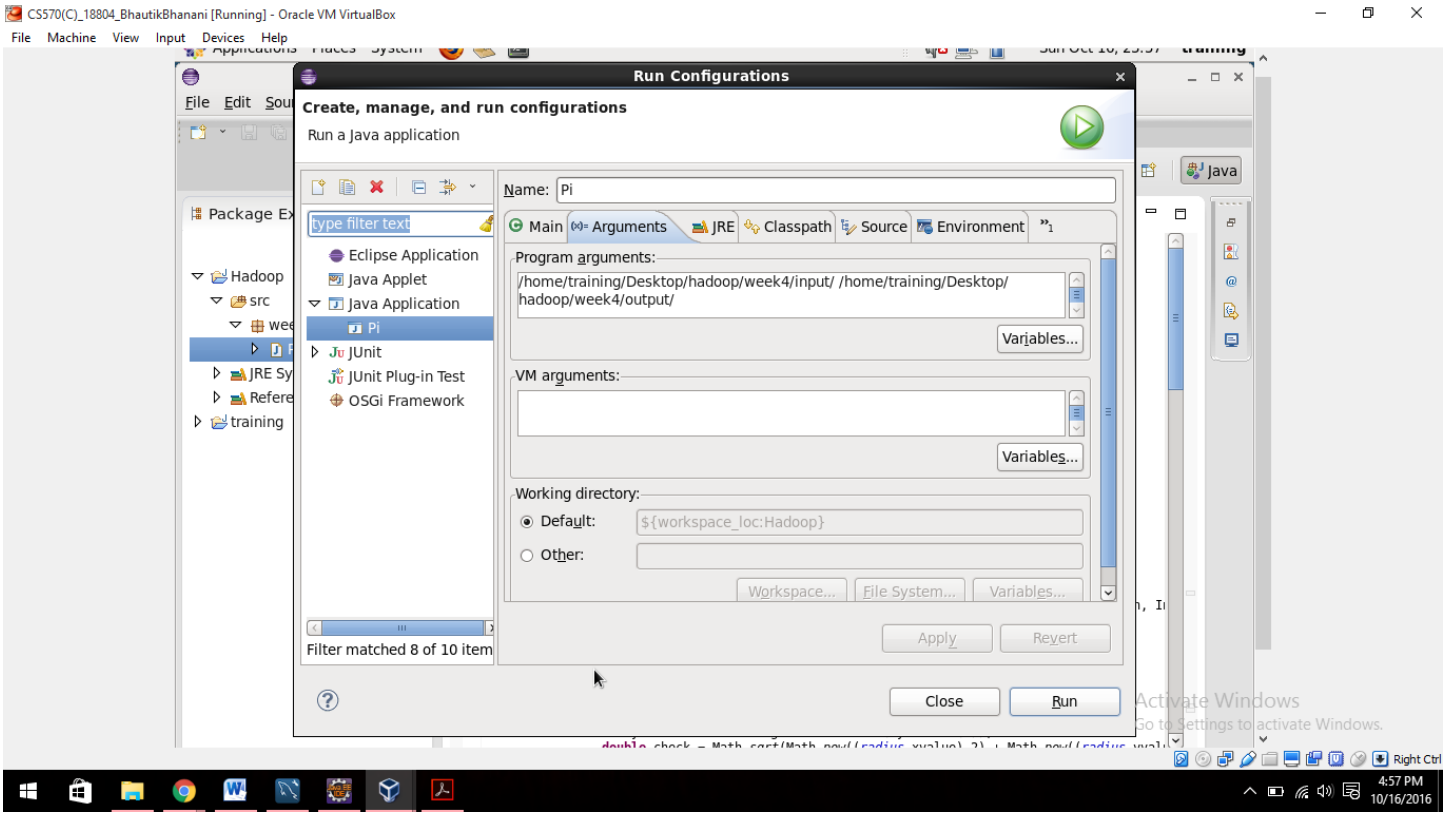
Step 3: Now import necessary hadoop jar files into “Java Build Path” of your project.



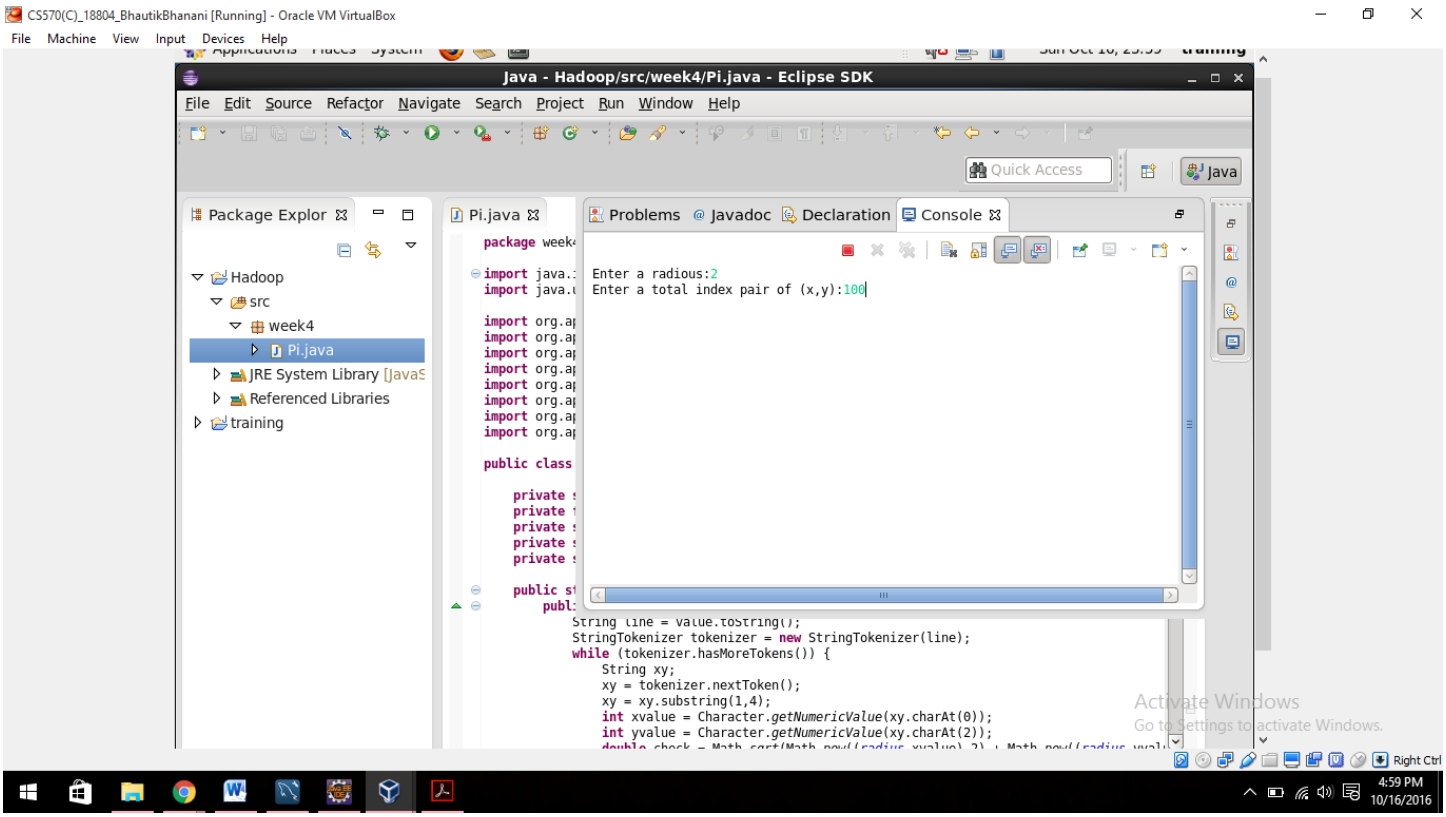
Step 4: Now run this program for once, so this class comes under “Java Application” in Run Configuration window.

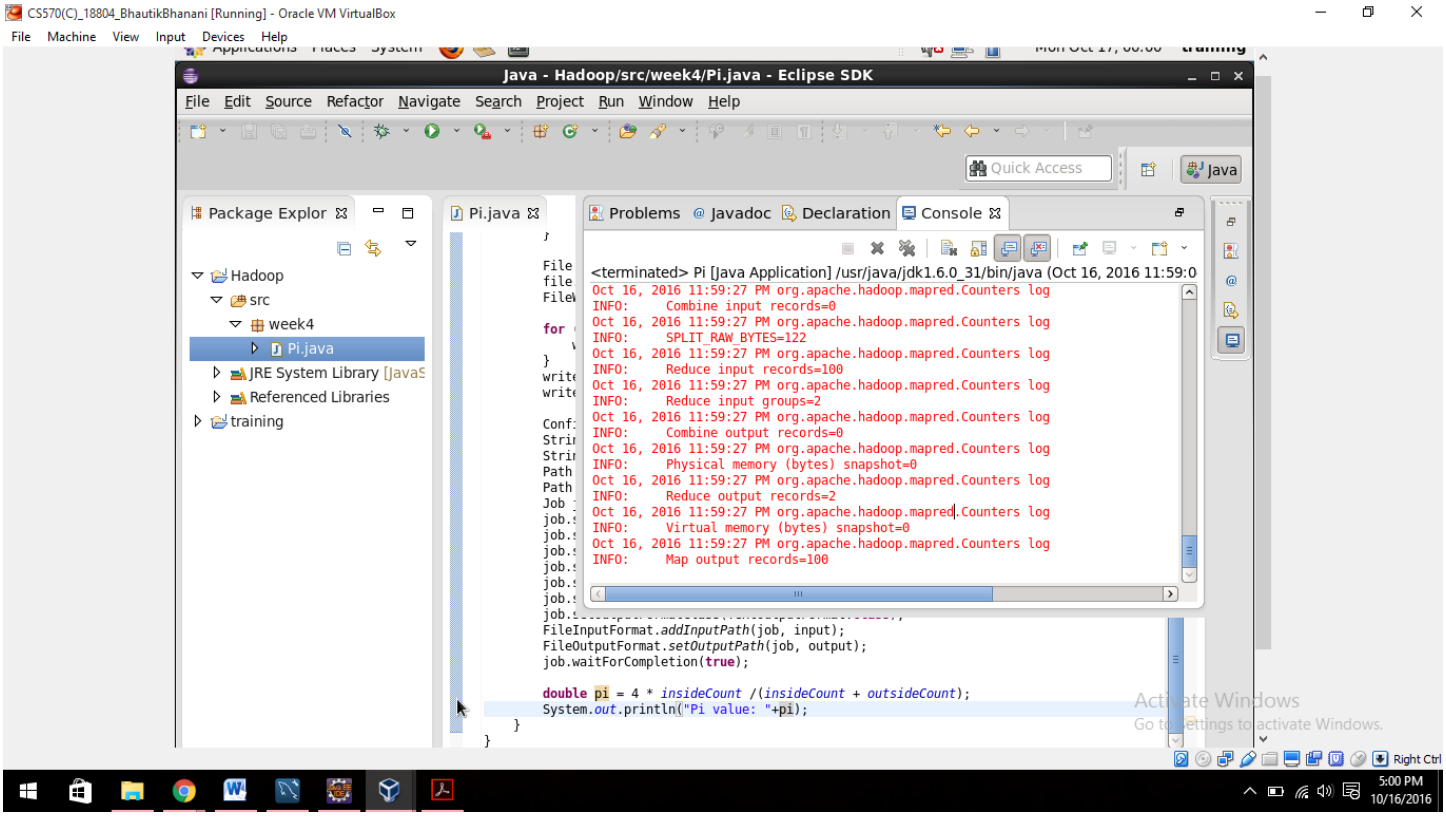


Step 5: Now add argument under “Arguments” window and for first argument give input folder and for second argument, give output folder.

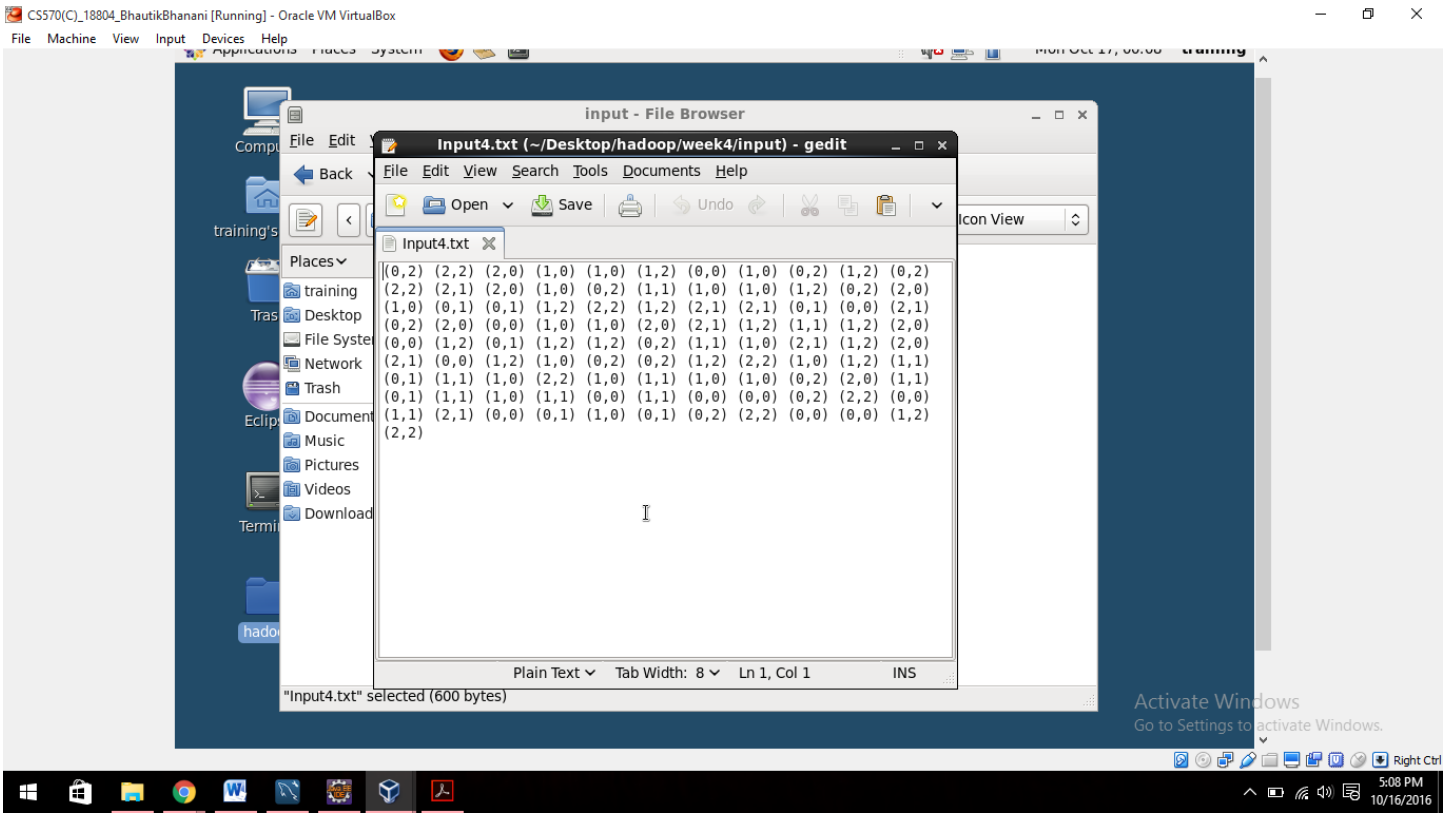


Step 6: Now run the program.

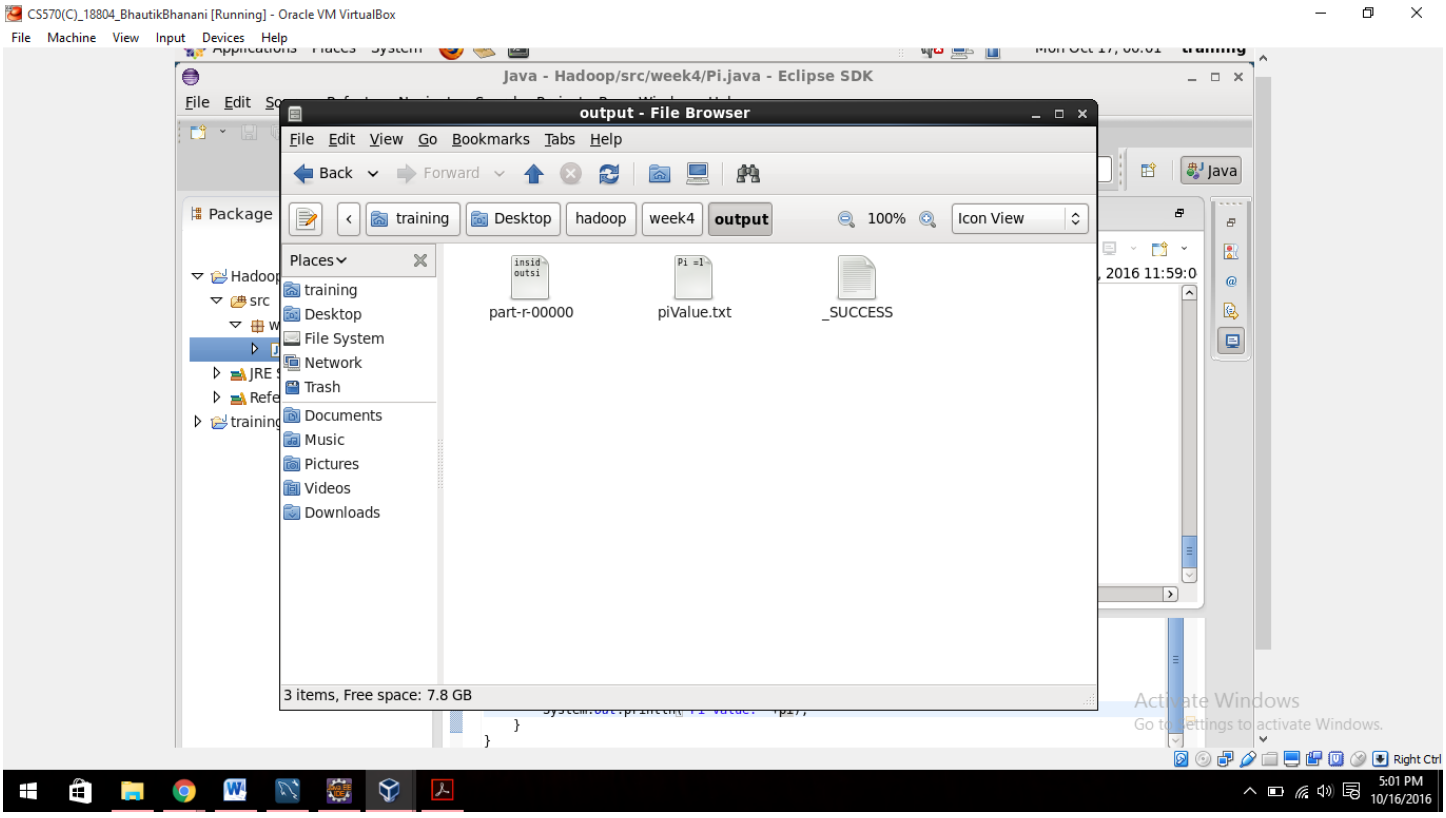




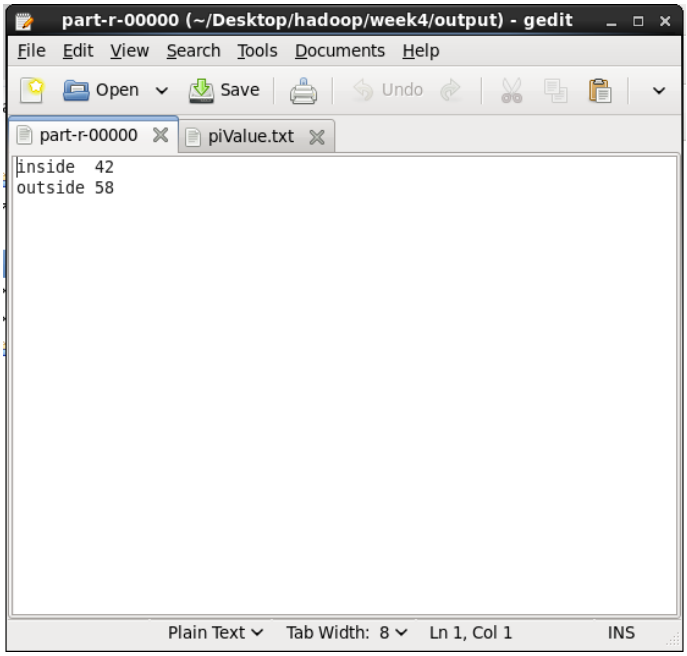
Step 7: Now first check input4.txt file under input folder.

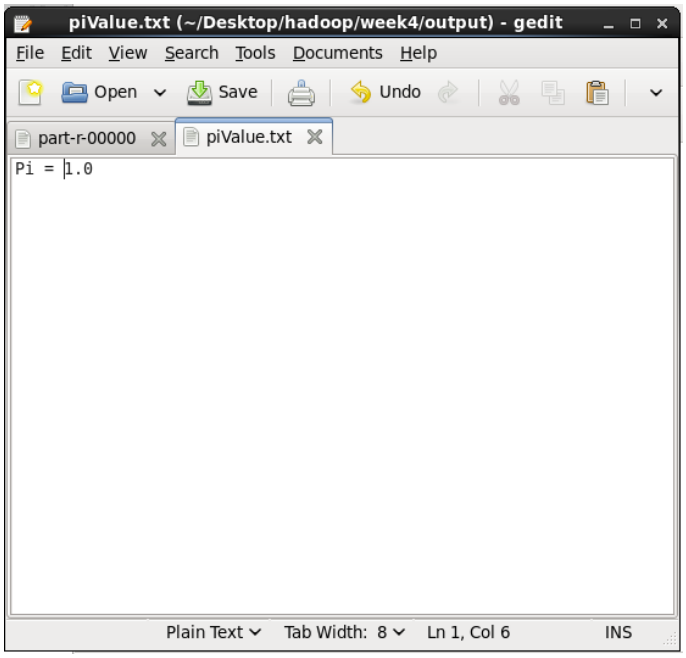


Now check output folder.



Open part-r-0000 and piValue.txt files.



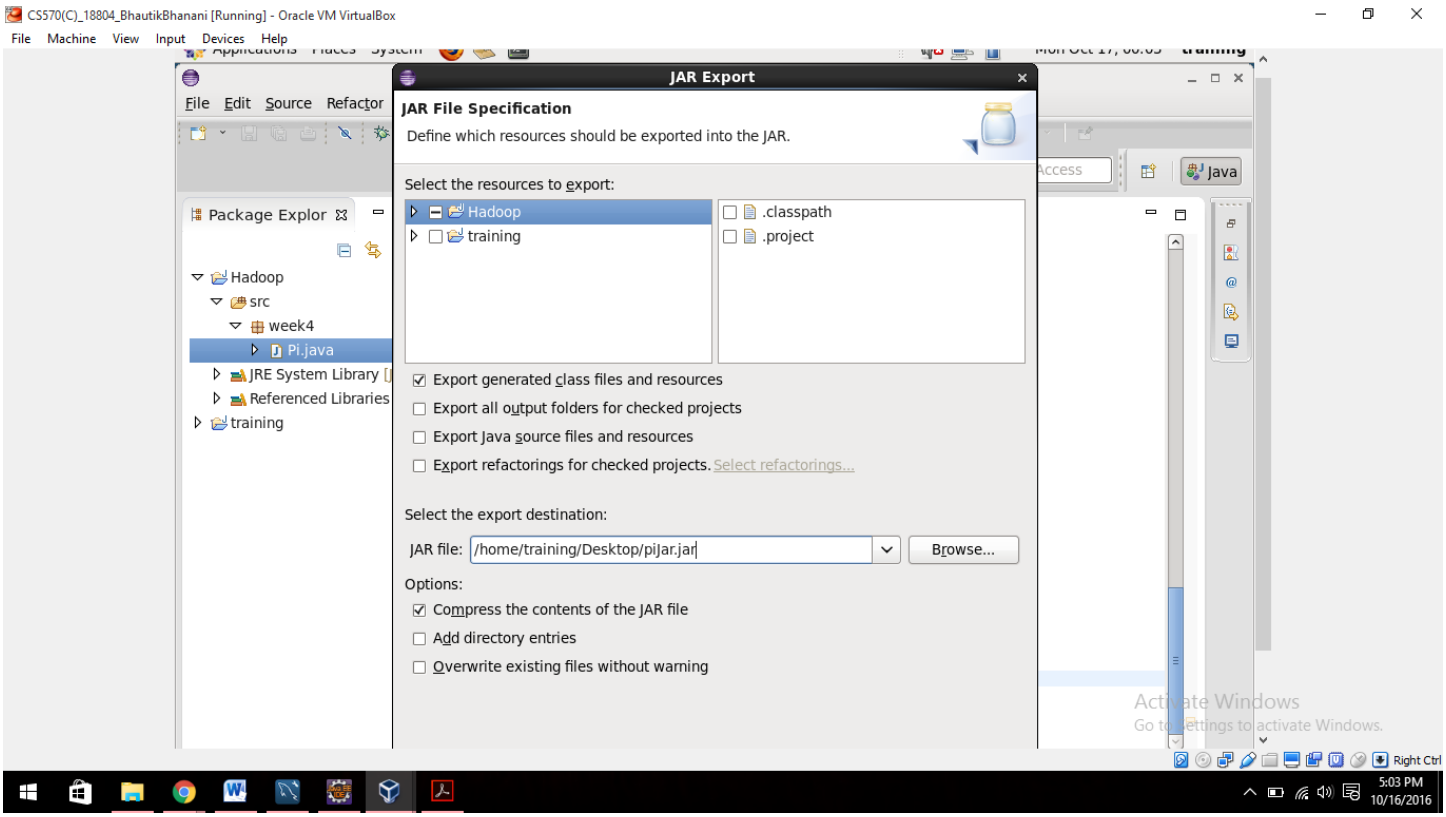


Step 8: Now change driver function as per bellow code. Here I am passing radius and number of (x,y) values in arguments for AWS.

```
public static void main(String[] args) throws Exception{
        radius = Integer.parseInt(args[0]);
        int index = Integer.parseInt(args[1]);

        int numX[] = new int[index];
        int numY[] = new int[index];

        for (int i = 0; i < index; i++){
                numX[i] = (int) (Math.random() * (radius + 1));
                numY[i] = (int) (Math.random() * (radius + 1));
                System.out.println(numX[i] + "," + numY[i]);
        }

        File file = new File(args[2] + "/Input4.txt");
        file.createNewFile();
        FileWriter writer = new FileWriter(file);

        for (int i = 0; i < index; i++){
                writer.write("(" + numX[i] + "," + numY[i] + ") ");
        }
        writer.flush();
        writer.close();
```

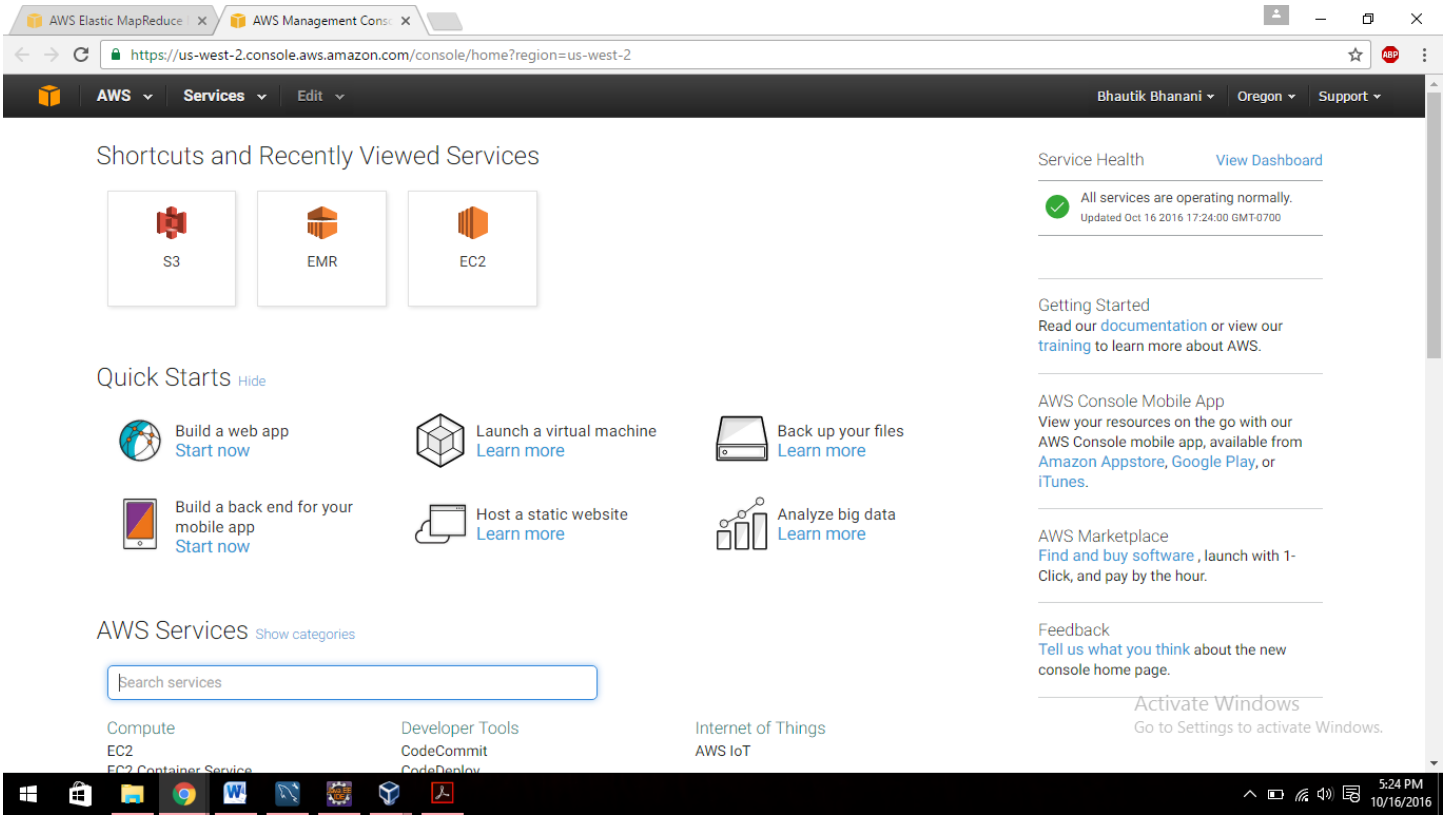
```
        Configuration conf = new Configuration();
        String Input = args[2];
        String Output = args[3];
        Path input = new Path(Input);
        Path output = new Path(Output);
        Job job = new Job(conf, "Pi");
        job.setJarByClass(Pi.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);
        job.setNumReduceTasks(1);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, input);
        FileOutputFormat.setOutputPath(job, output);
        job.waitForCompletion(true);

        double pi = 4 * insideCount /(insideCount + outsideCount);
        File file1 = new File(args[3] +"/piValue.txt");
        FileWritter fw = new FileWritter(file1);
        BufferedWritter bw = new BufferedWritter(fw);
        bw.write(String.valueOf("Pi value = "+pi));
        bw.close();
}
```

Step 9: Now export Pi.class into jar file.

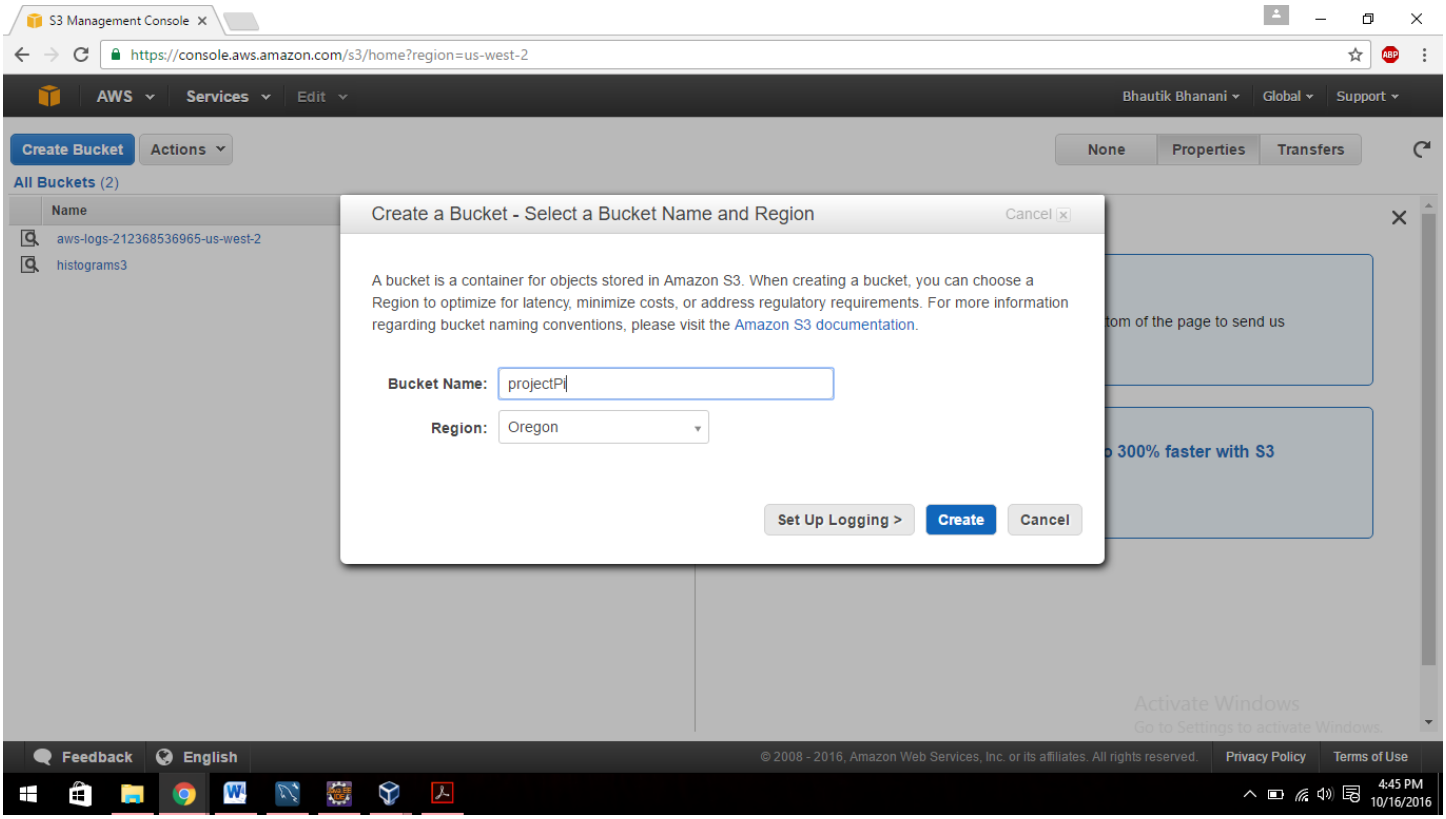


Now run Pi project on AWS

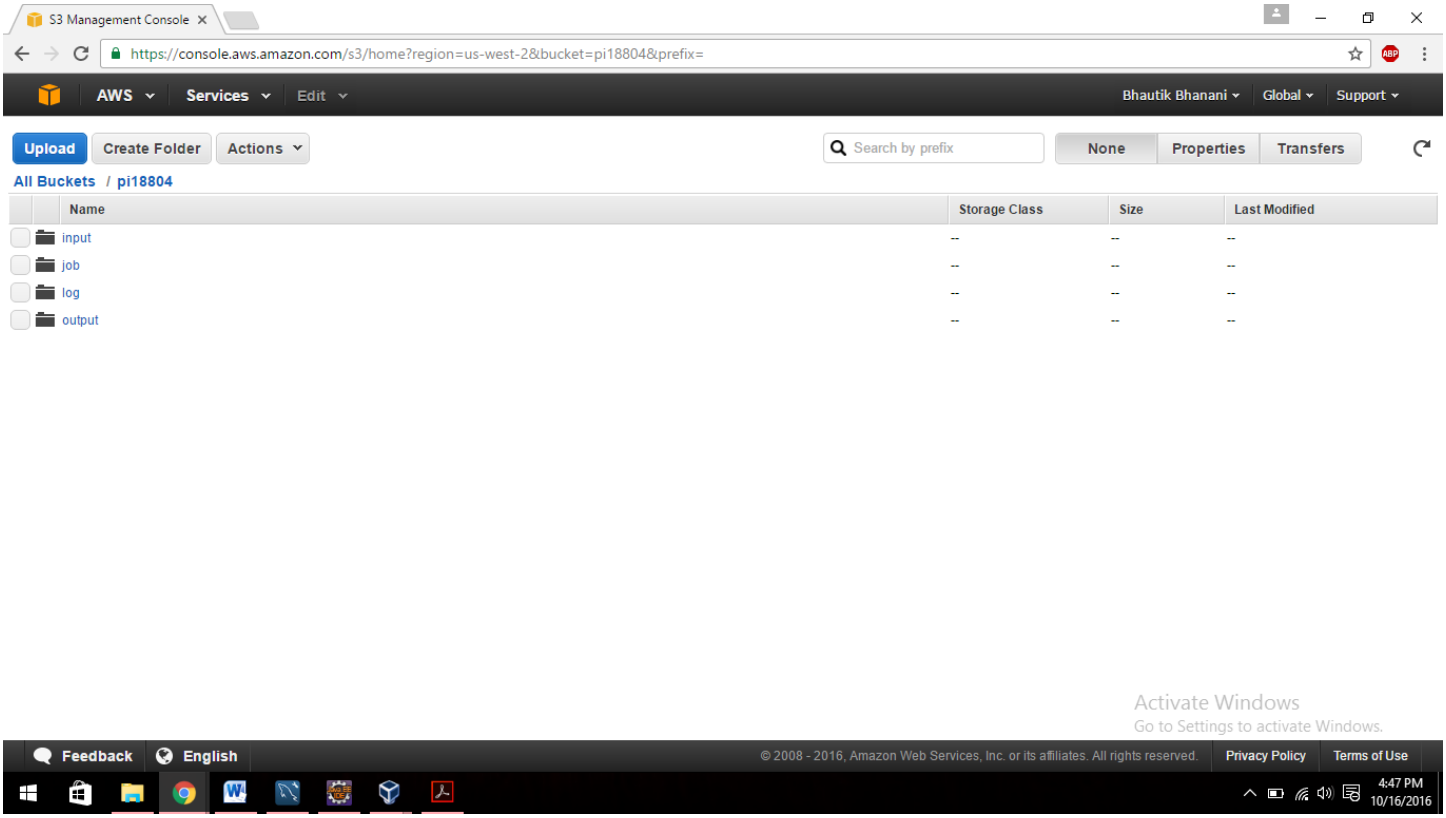
Step 1: Login to your AWS account and AWS dashboard.



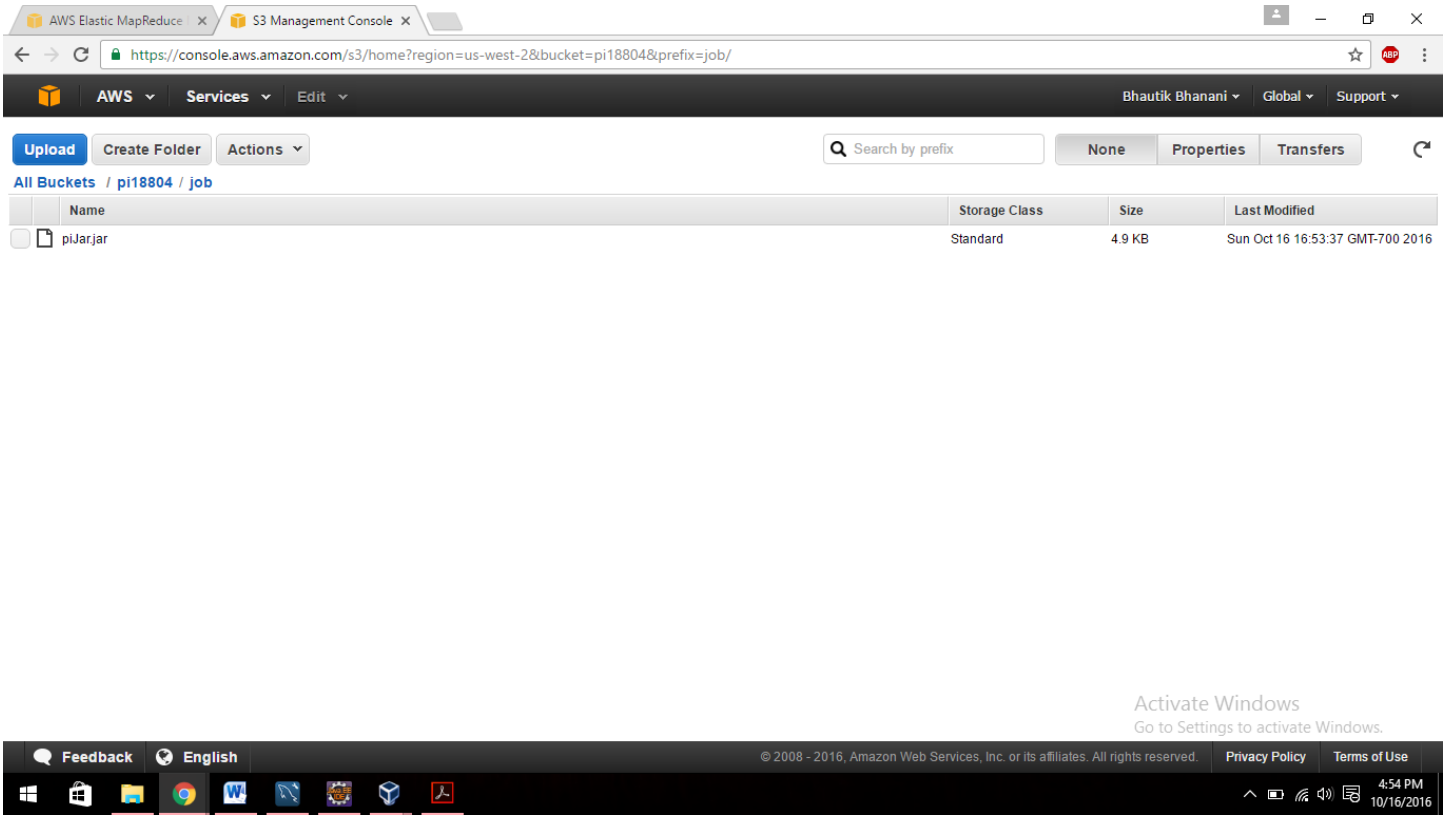
Step 2: Go to **S3**, and create one bucket for Pi project.



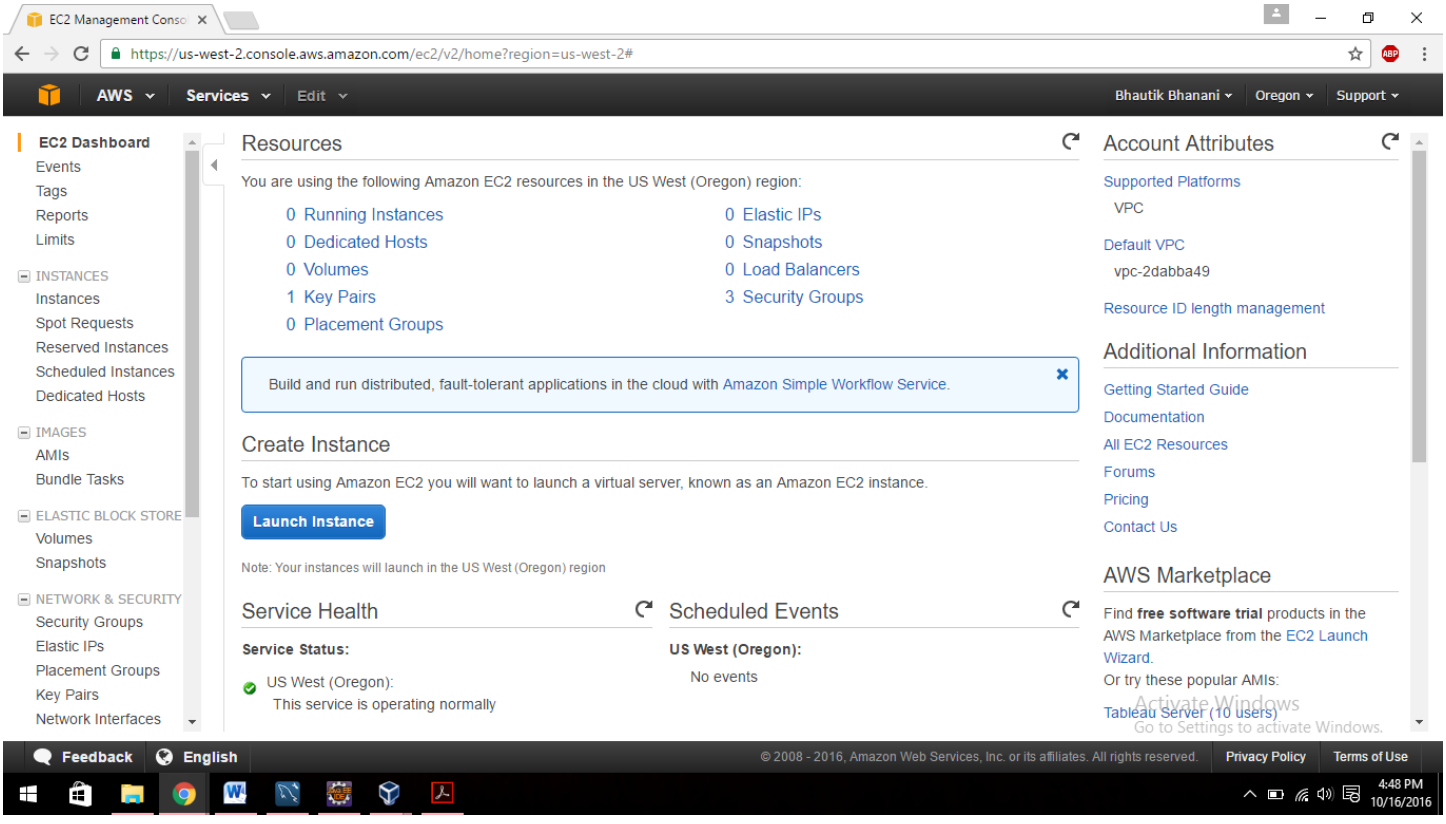
Step 3: Under your newly created bucket, create following folders: input, output, jobs and log.



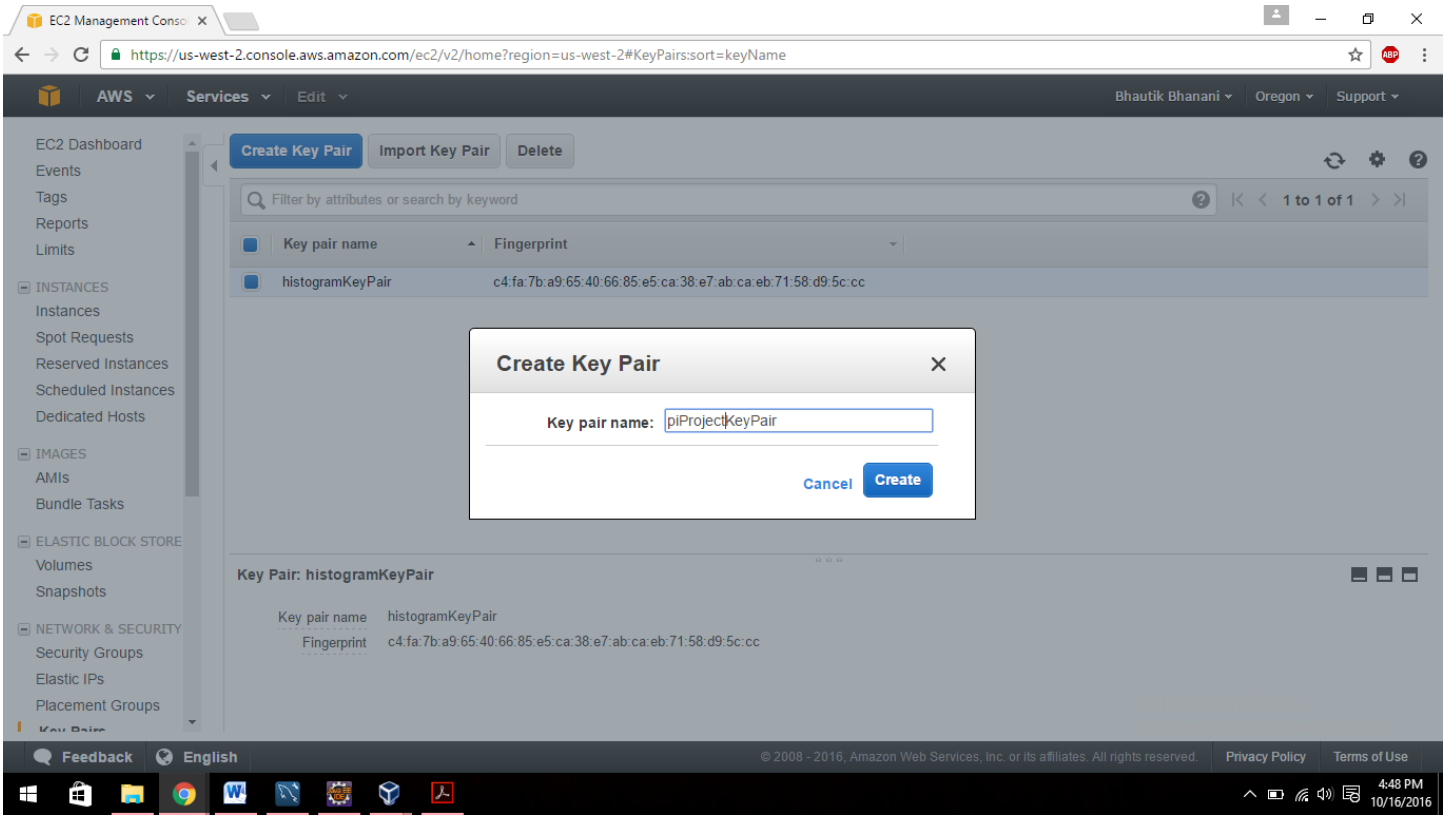
Step 4: Upload your exported jar file under **job** folder.



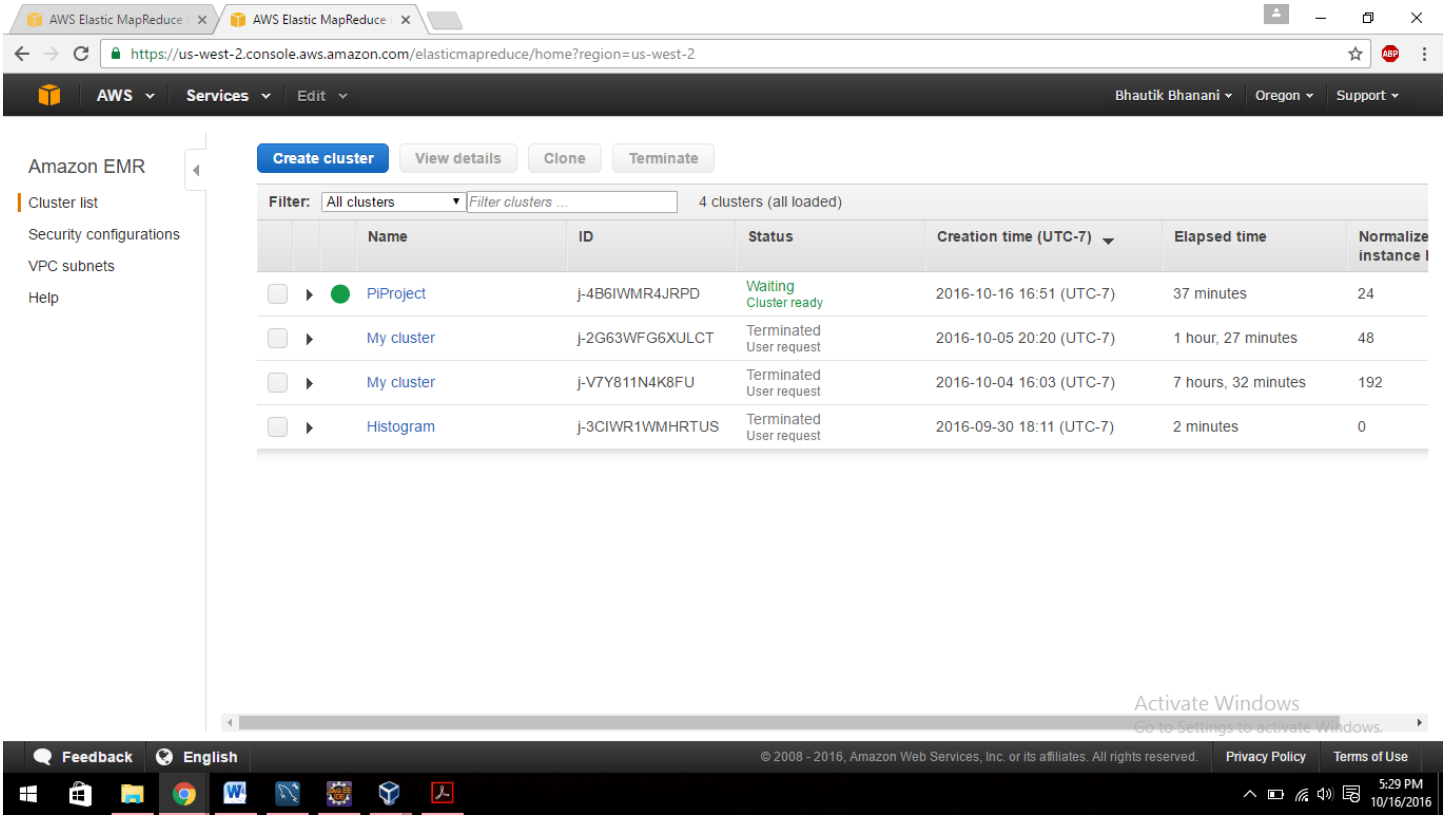
Step 5: Now go to **EC2** from AWS dashboard.



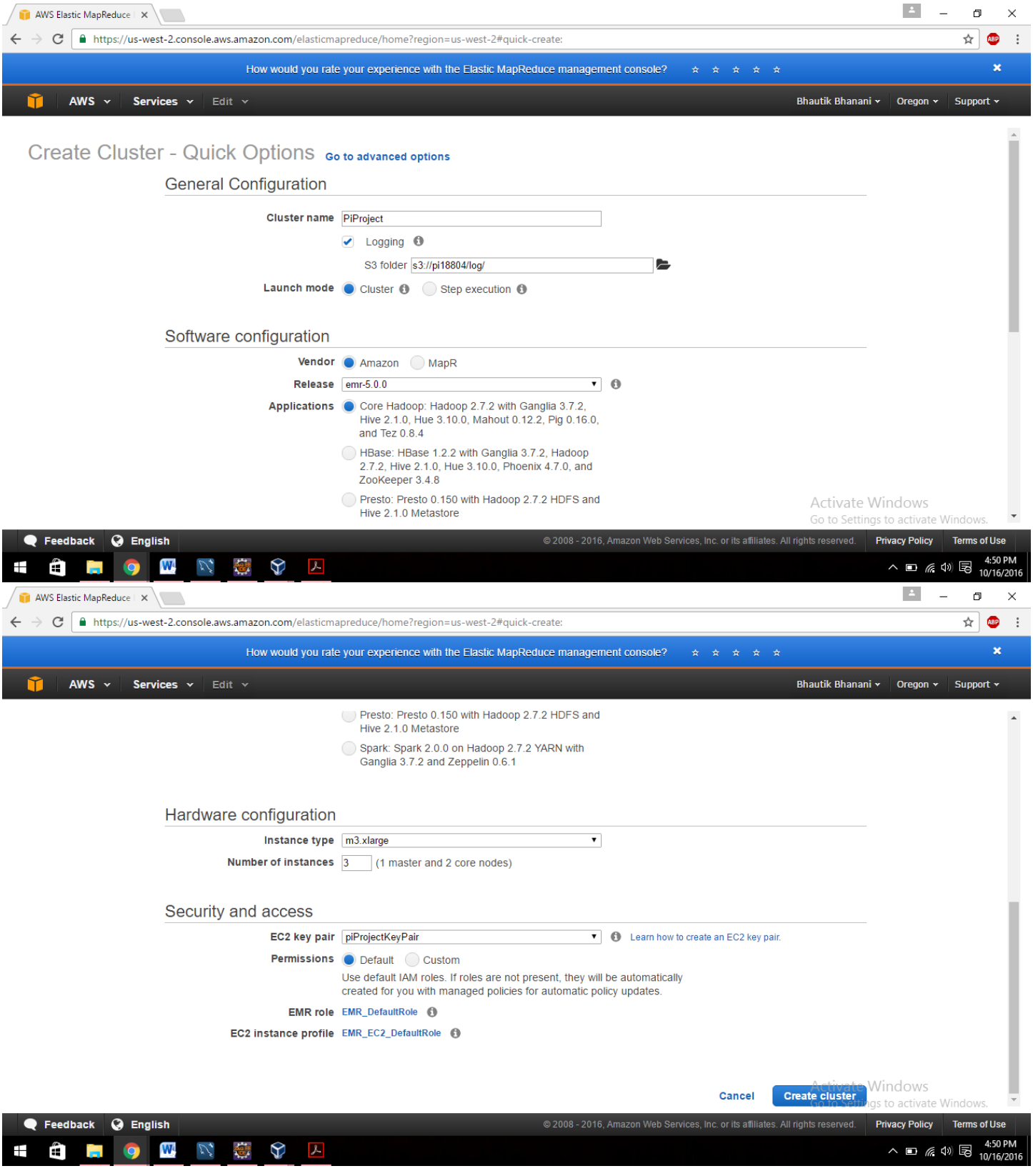
Step 6: Create key pair for your project.

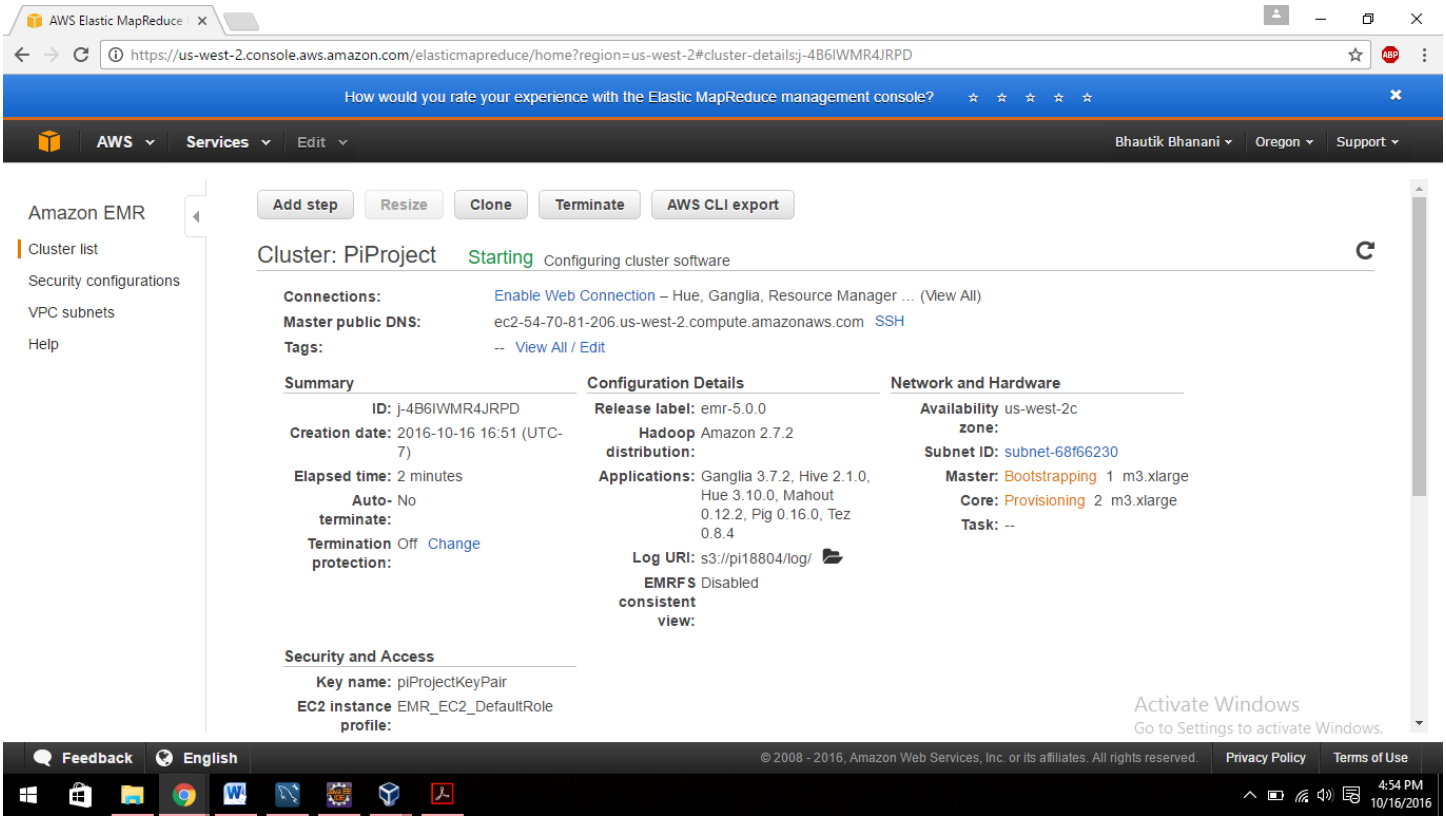


Step 7: Go to **EMR** from AWS dashboard.

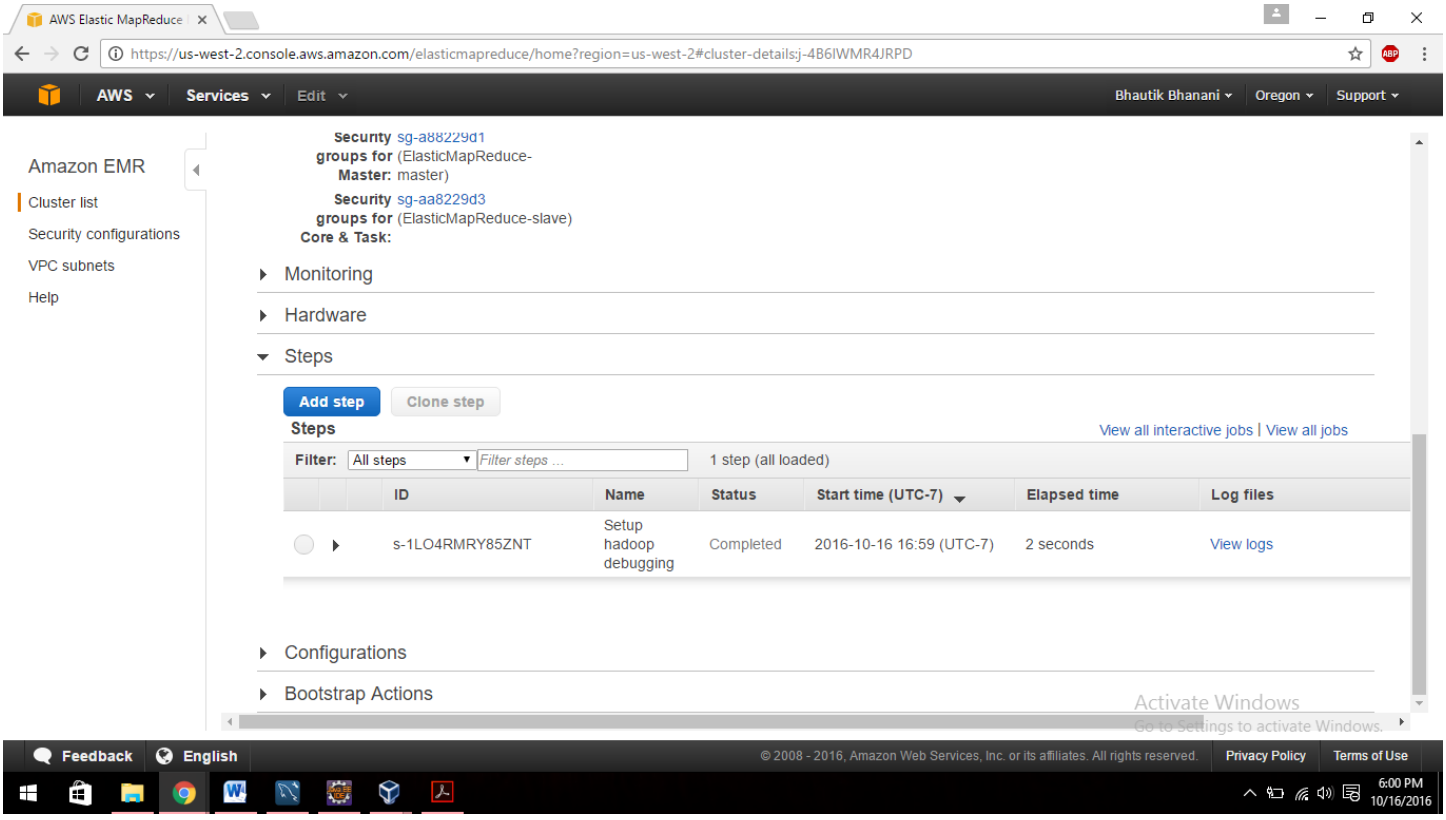


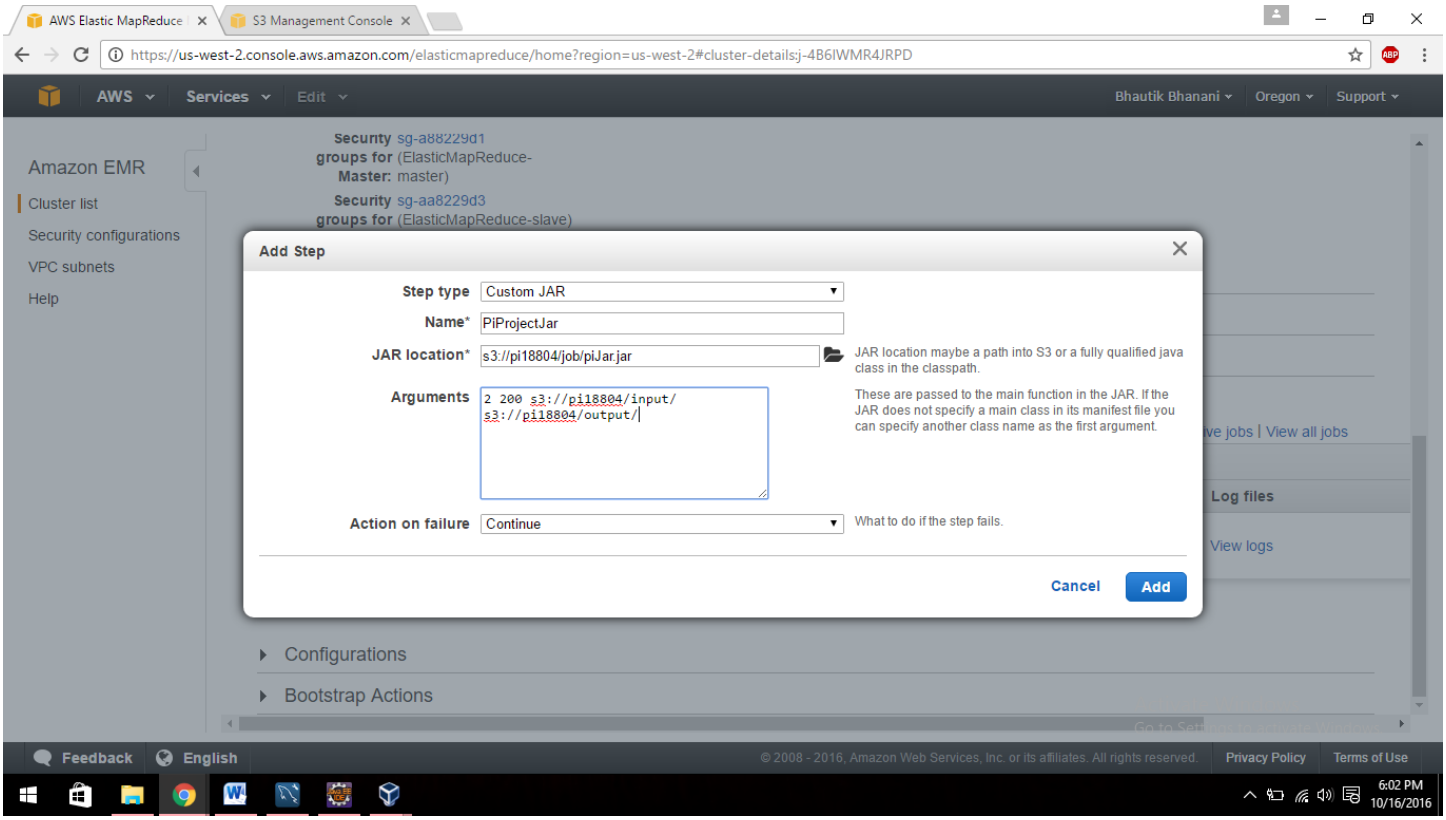
Step 8: Create new cluster of Pi project.



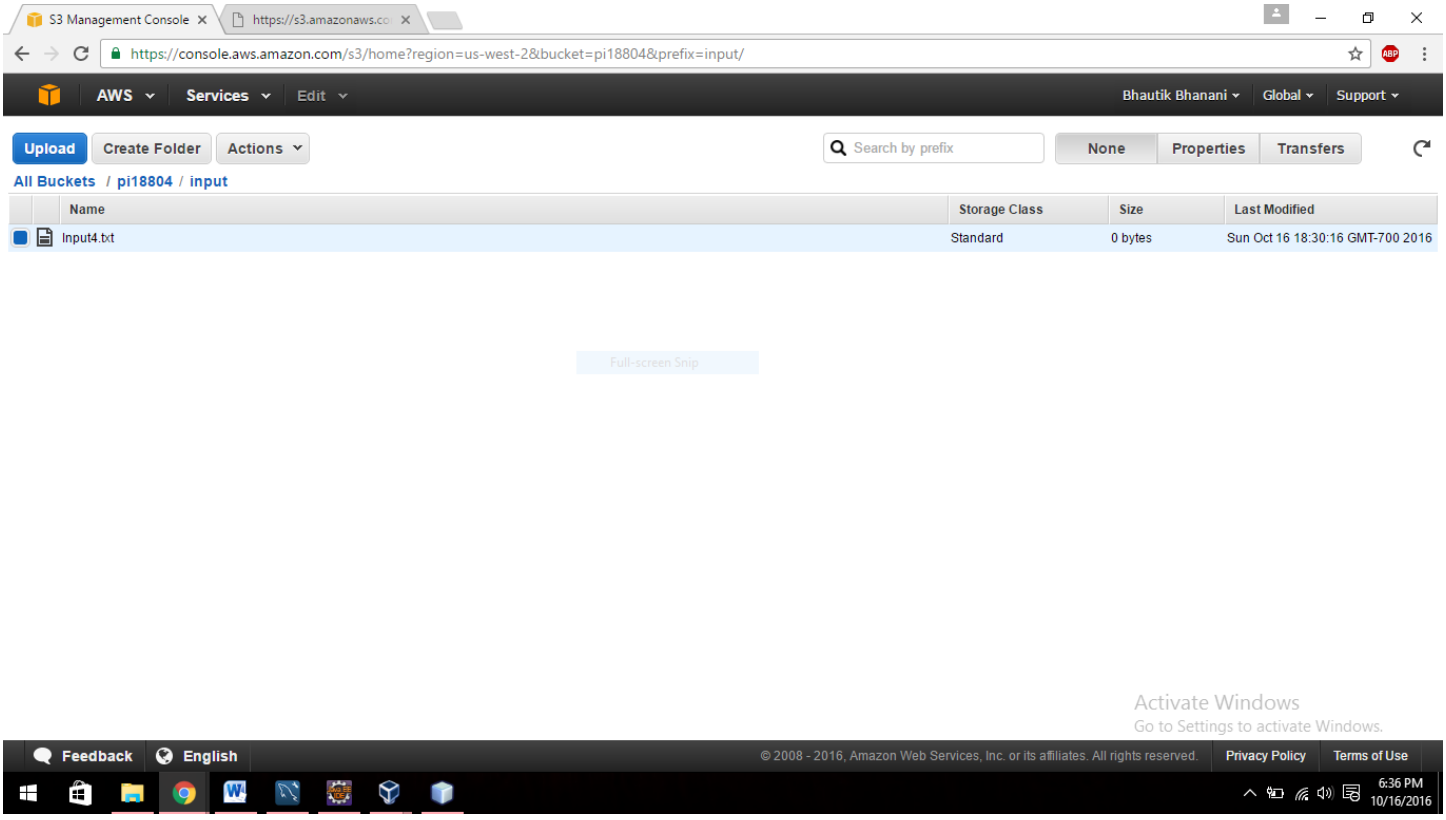


Step 9: When the cluster is in running state, create on step under Steps tab.

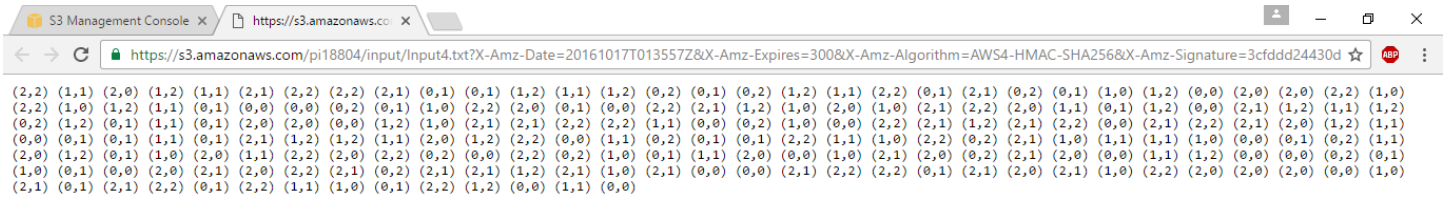




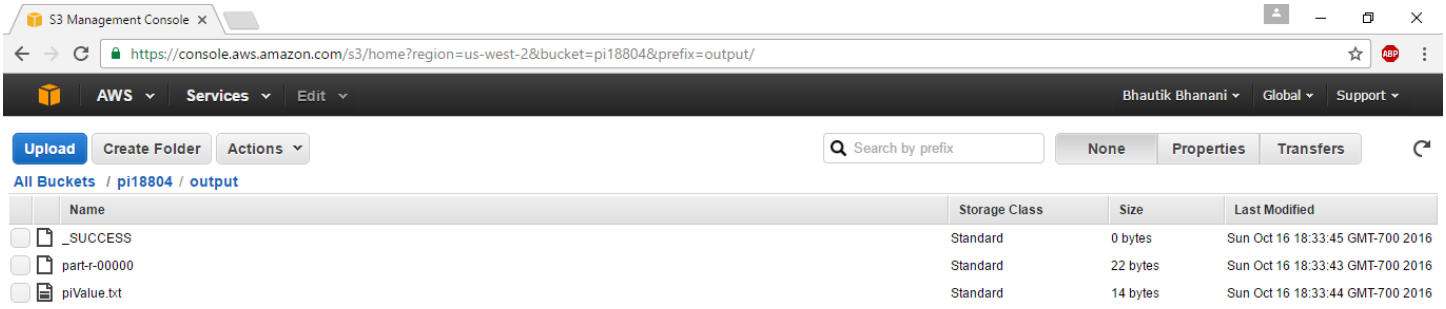
Step 10: When your created step shows status Completed, then go to input folder under **S3**.

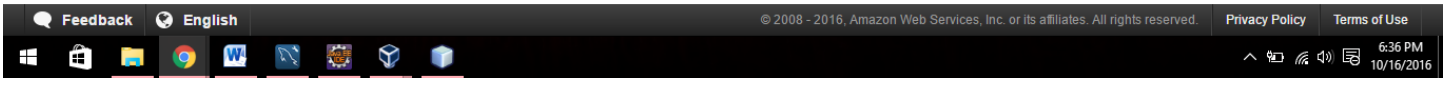


Open Input4.txt

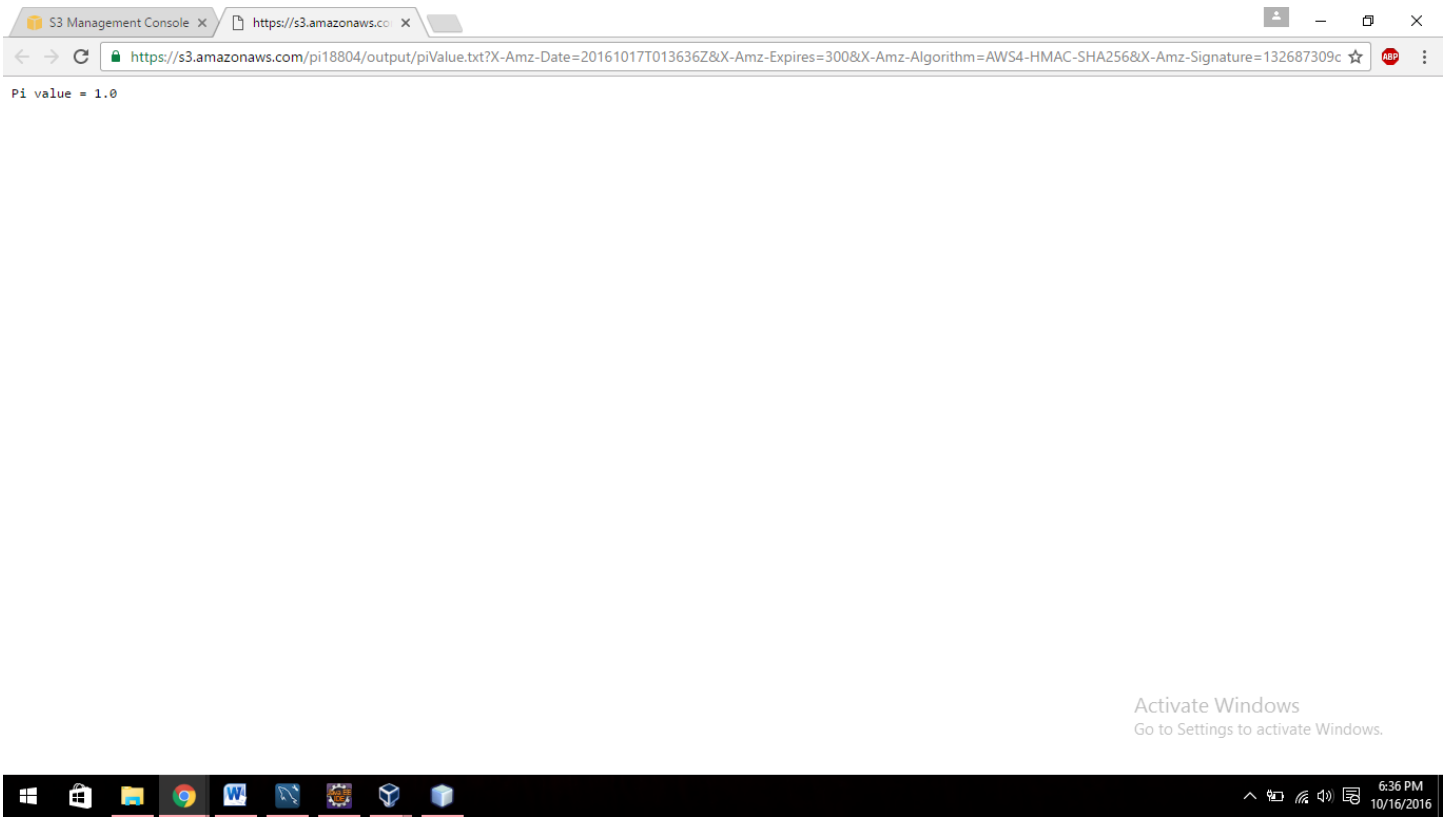


Step 12: Now check output folder.

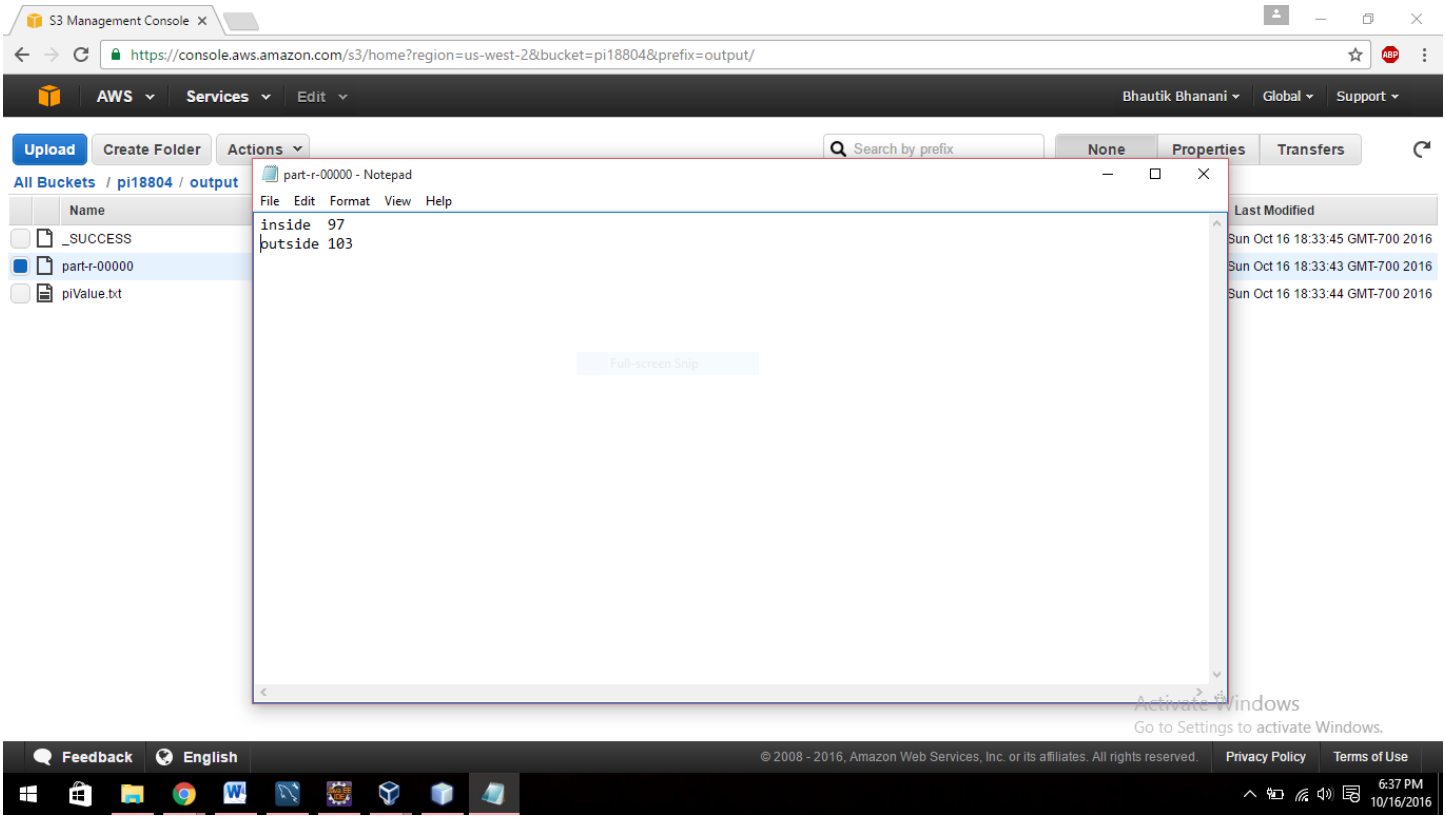




Open piValue.txt



Now open part-r-0000 file



Step 13: After this, terminate your cluster. If it will be running then AWS will charge more for that.