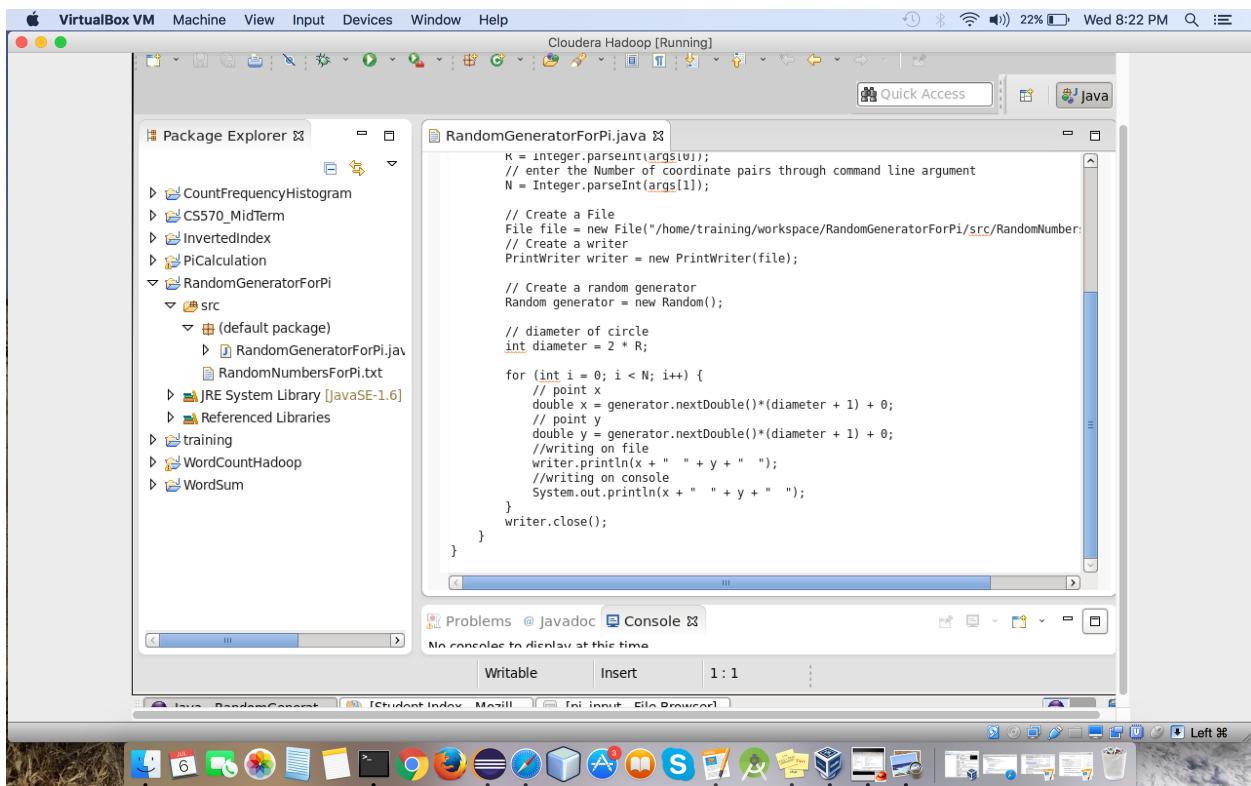
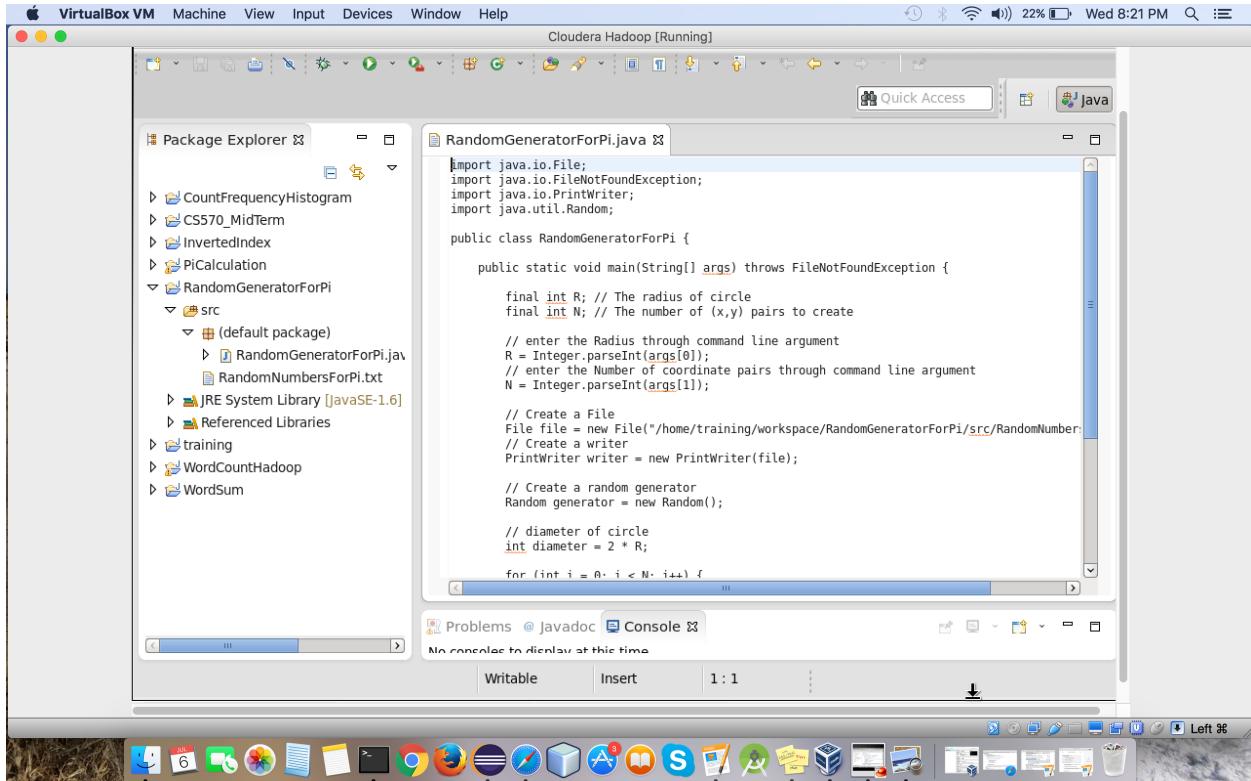


Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

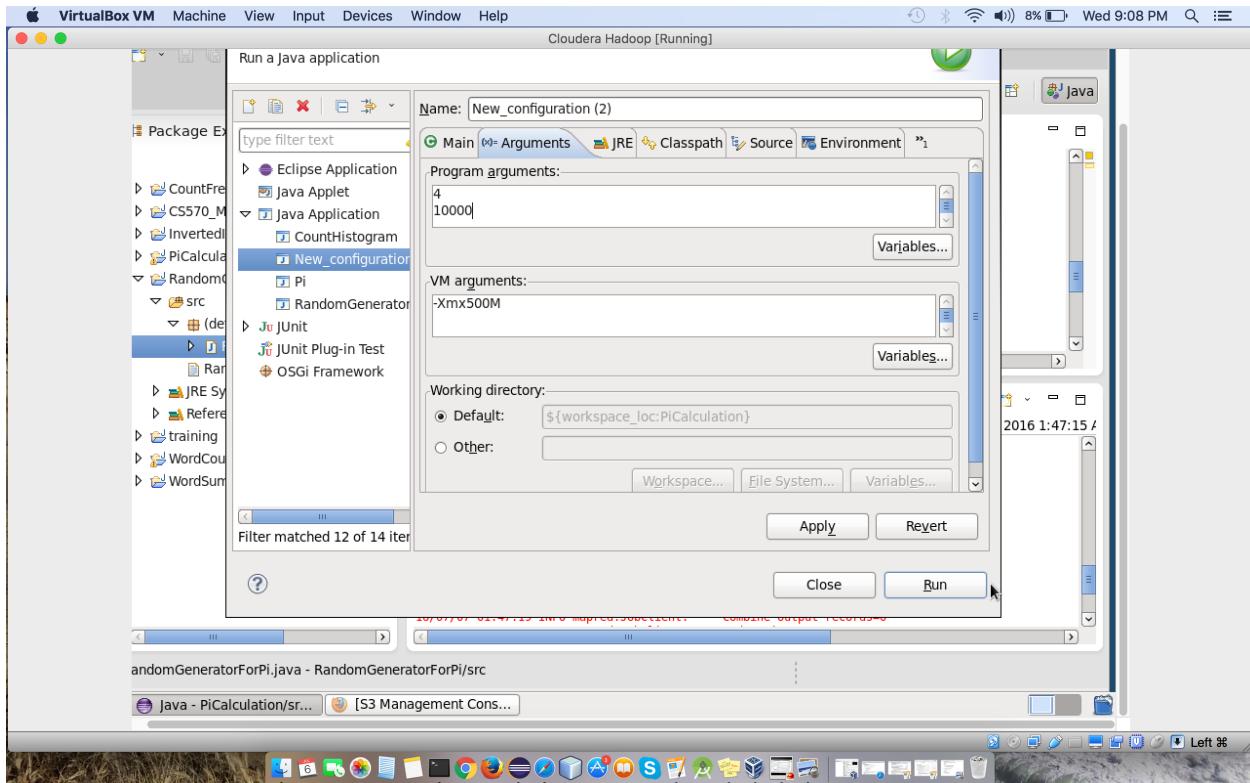
Generating a PI MapReduce program to calculate the PI value on Virtual Machine- Cloudera and then running it on Amazon Web Service: Create RandomGeneratorForPi.java class



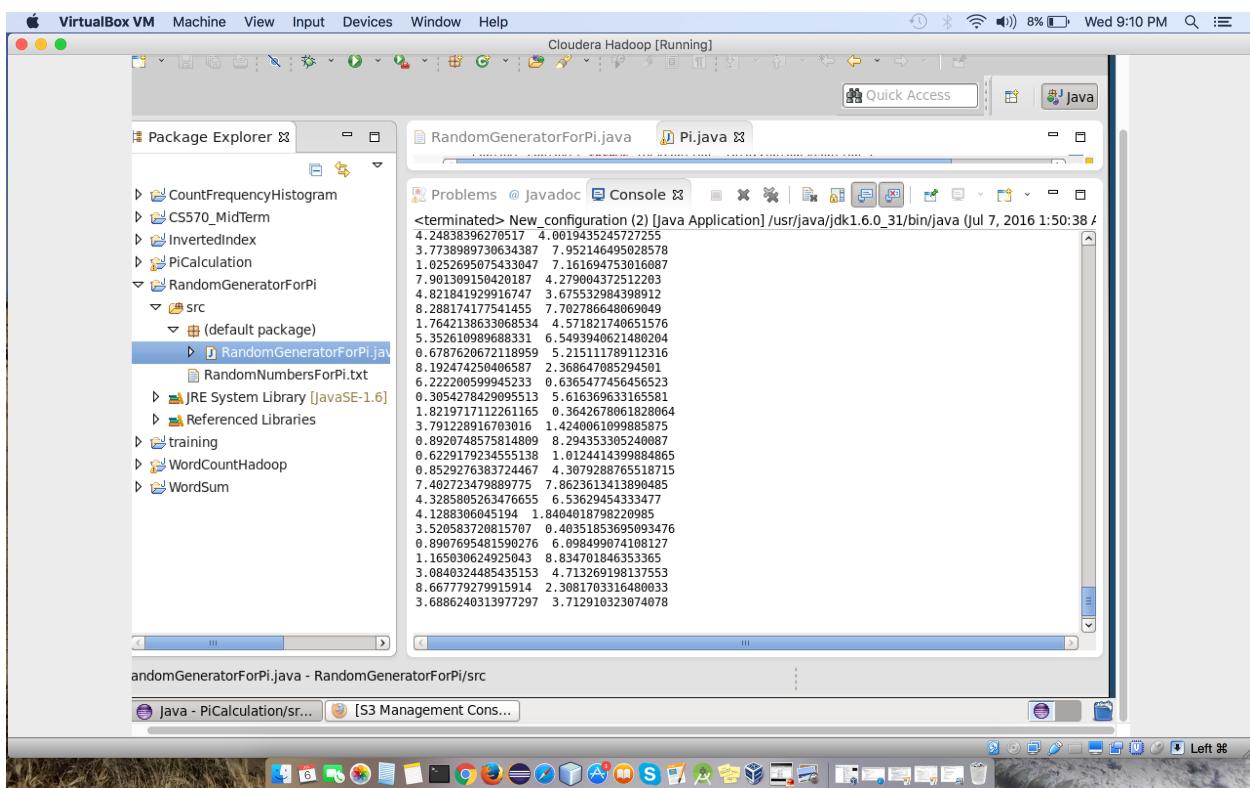
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Enter Arguments as Radius=4 and Number of pairs to generate=10000:



Run the RandomGeneratorForPi.java class in command line and check the output:

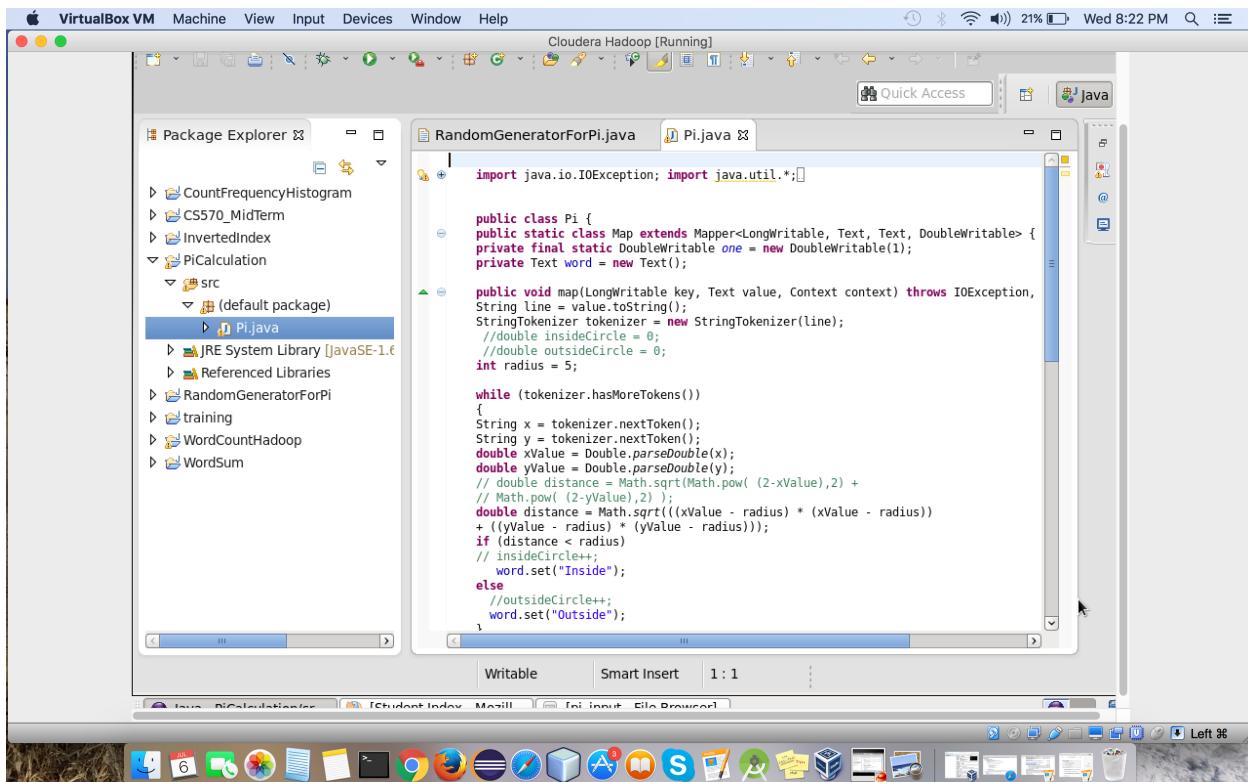


Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

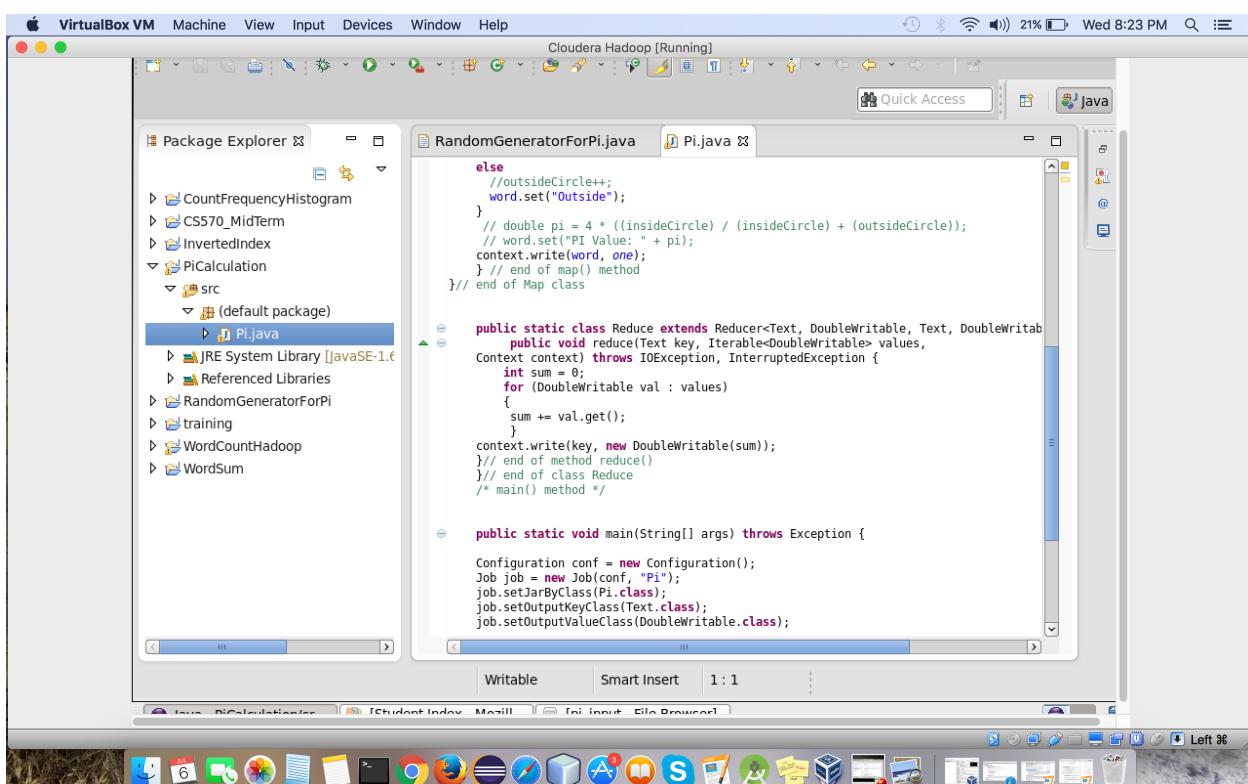
The above output generated is save in RandomGeneratorForPi.txt file.

Create a Pi.java class:



Screenshot of the Eclipse IDE interface showing the RandomGeneratorForPi.java file in the editor. The code implements the map() method for a Mapper. It tokenizes input lines, calculates distances from a center point (0,0) with radius 5, and sets the word to "Inside" if the point is inside the circle or "Outside" if it is outside. The code uses StringTokenizer, Double.parseDouble, and Math.sqrt.

```
import java.io.IOException; import java.util.*;public class Pi { public static class Map extends Mapper<LongWritable, Text, Text, DoubleWritable> { private final static DoubleWritable one = new DoubleWritable(1); private Text word = new Text(); public void map(LongWritable key, Text value, Context context) throws IOException, String line = value.toString(); StringTokenizer tokenizer = new StringTokenizer(line); //double insideCircle = 0; //double outsideCircle = 0; int radius = 5; while (tokenizer.hasMoreTokens()) { String x = tokenizer.nextToken(); String y = tokenizer.nextToken(); double xValue = Double.parseDouble(x); double yValue = Double.parseDouble(y); //double distance = Math.sqrt(Math.pow( (2-xValue),2 ) + // Math.pow( (2-yValue),2 )); double distance = Math.sqrt((xValue - radius) * (xValue - radius) + ((yValue - radius) * (yValue - radius))); if (distance < radius) // insideCircle++; word.set("Inside"); else //outsideCircle++; word.set("Outside"); } } // end of Map class
```



Screenshot of the Eclipse IDE interface showing the RandomGeneratorForPi.java file in the editor. The code continues from the previous screenshot, defining the reduce() method which sums up the values for each category ("Inside" or "Outside") and writes the total count back to the context. It also includes the main() method which configures the job and runs it.

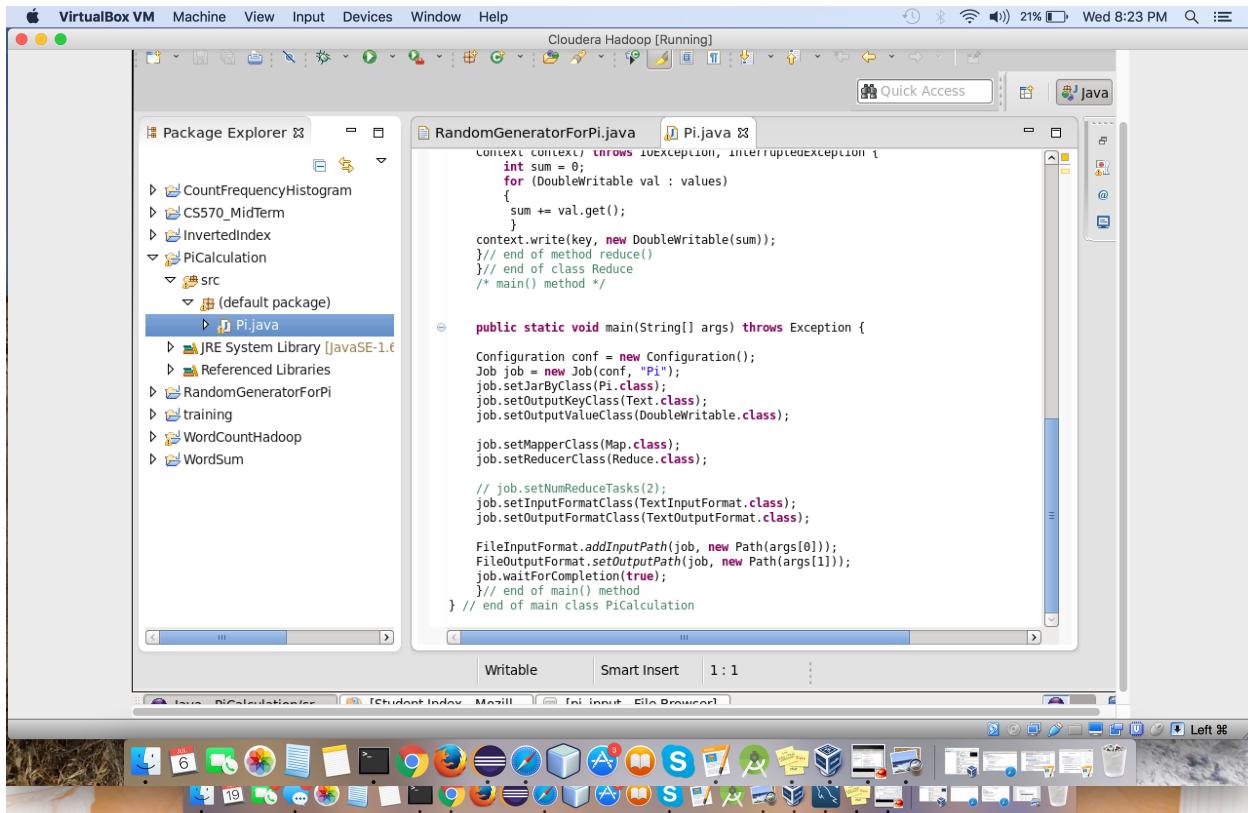
```
else //outsideCircle++; word.set("Outside"); } // double pi = 4 * ((insideCircle) / (insideCircle) + (outsideCircle)); // word.set("PI Value: " + pi); context.write(word, one); } // end of map() method } // end of class Map
```

```
public static class Reduce extends Reducer<Text, DoubleWritable, Text, DoubleWritable> { public void reduce(Text key, Iterable<DoubleWritable> values, Context context) throws IOException, InterruptedException { int sum = 0; for (DoubleWritable val : values) { sum += val.get(); } context.write(key, new DoubleWritable(sum)); } // end of method reduce() } // end of class Reduce /* main() method */
```

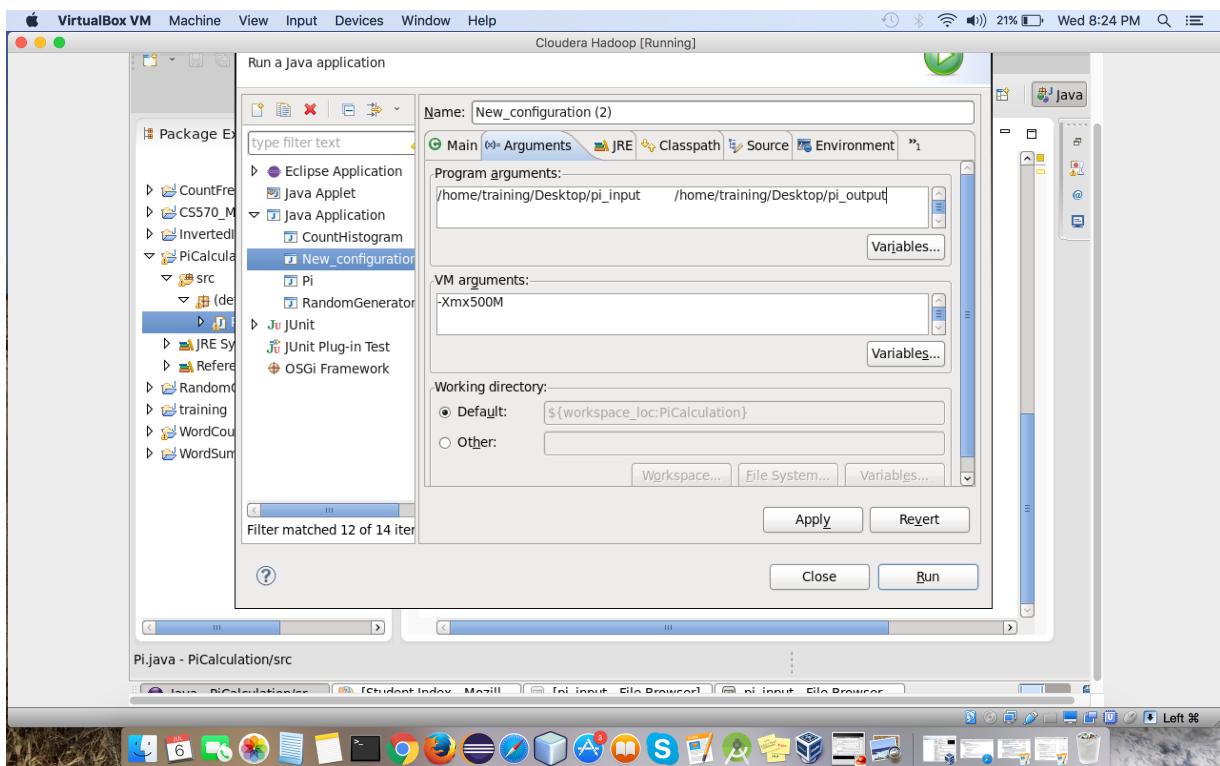
```
public static void main(String[] args) throws Exception { Configuration conf = new Configuration(); Job job = new Job(conf, "Pi"); job.setJarByClass(Pi.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(DoubleWritable.class); job.setMapperClass(Map.class); job.setReducerClass(Reduce.class); job.setMapOutputKeyClass(DoubleWritable.class); job.setMapOutputValueClass(DoubleWritable.class); job.setNumReduceTasks(1); FileInputFormat.addInputPath(new Path(args[0]), job); FileOutputFormat.setOutputPath(new Path(args[1]), job); job.waitForCompletion(true); } }
```

Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com



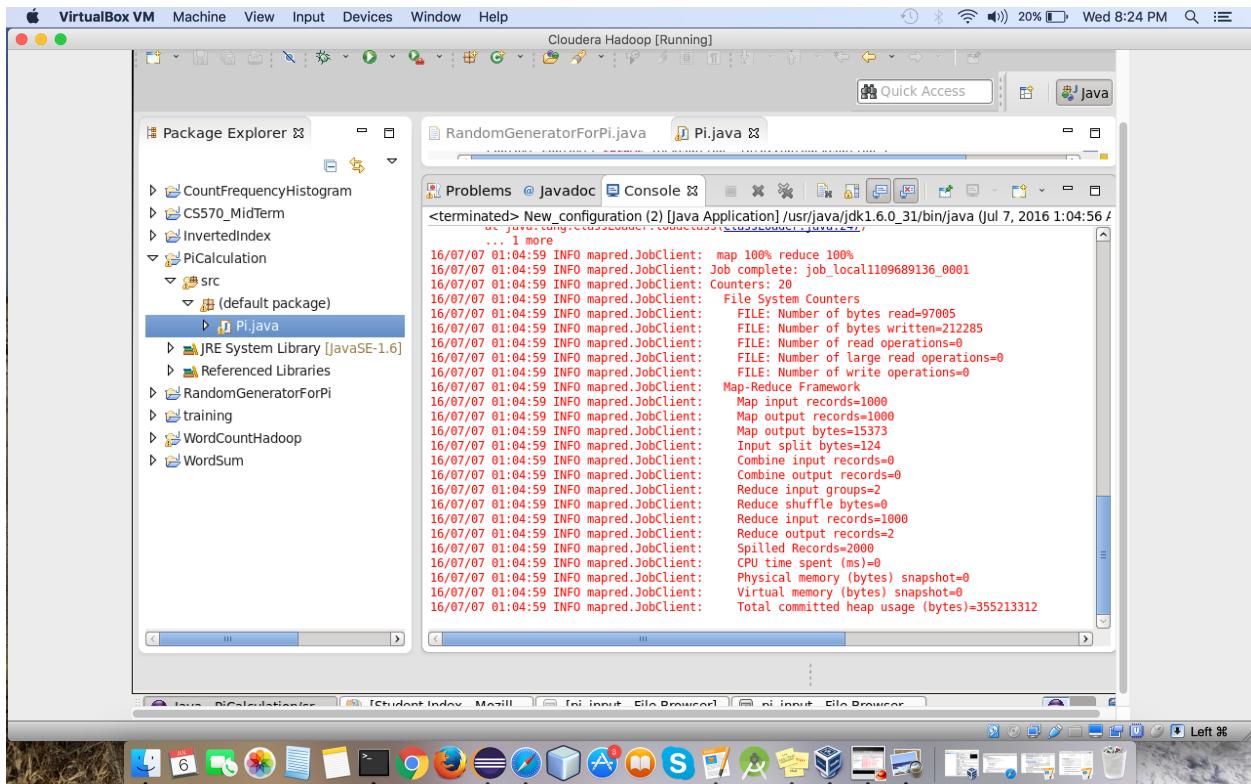
In the argument, provide the input file path where RandomGeneratorForPi.txt is saved and the path where output file will be automatically generated:



Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

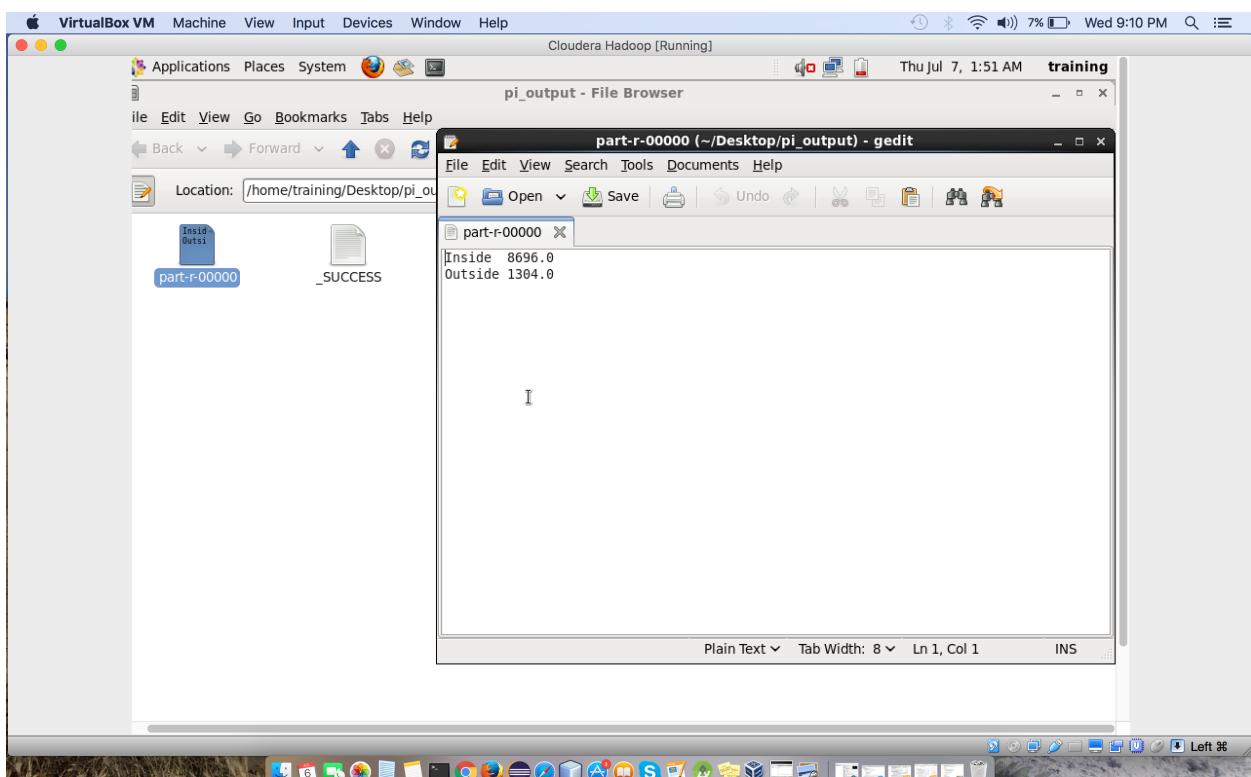
Run the Pi.java class:



The following output is generated in Pi_output file:

Number of darts inside:8696.0

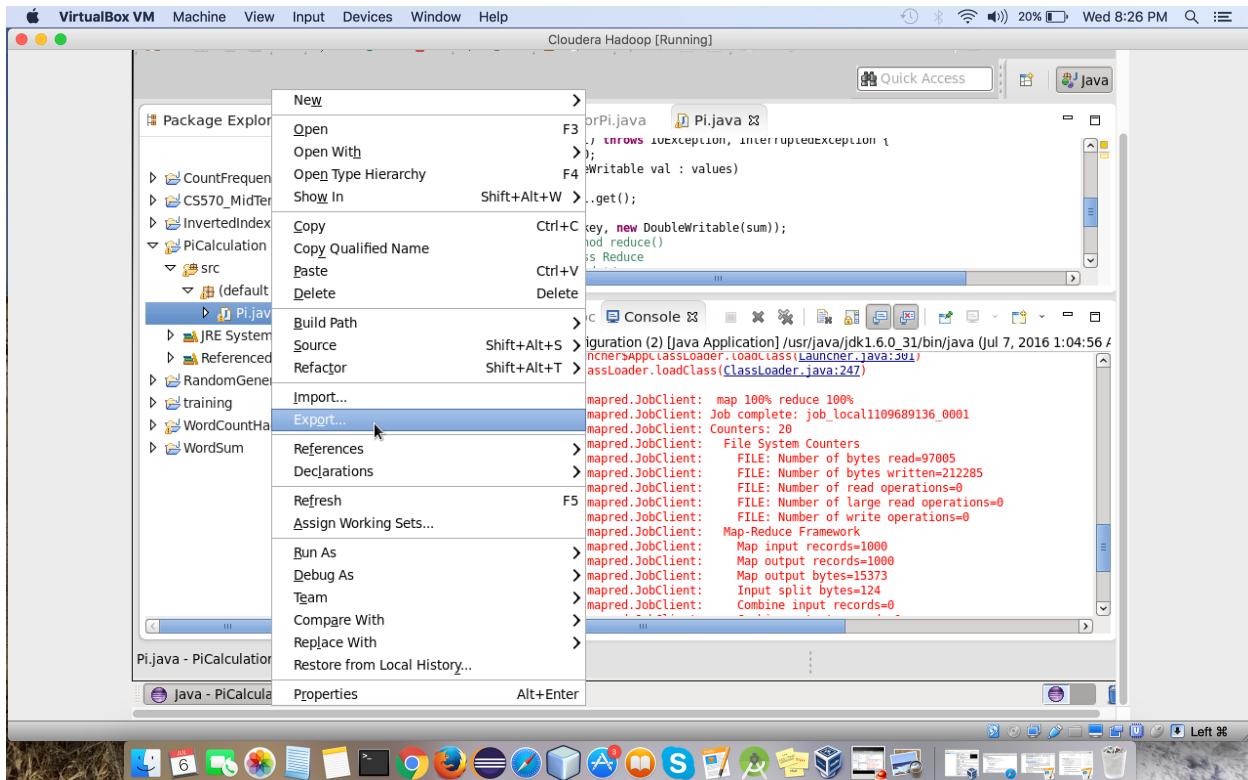
Number of darts outside: 1304.0



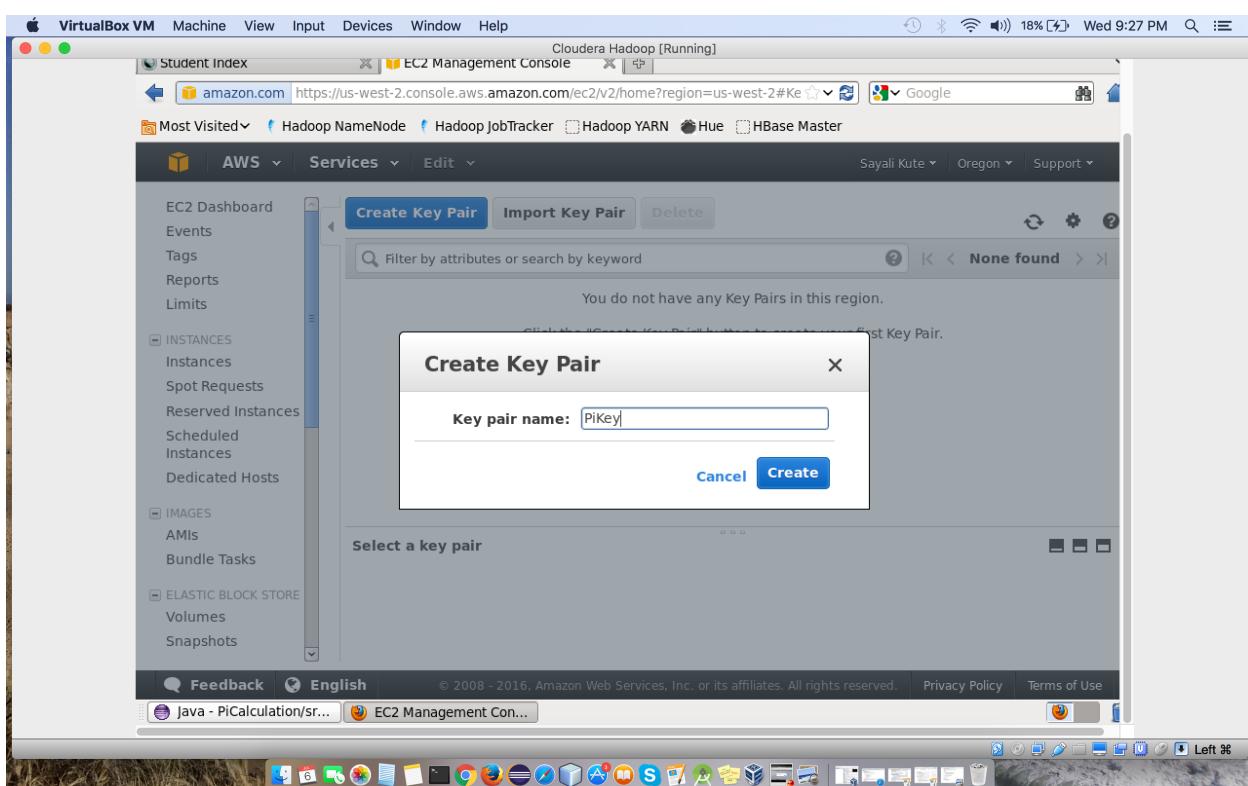
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Export the Pi.java class and save it with .jar extension:

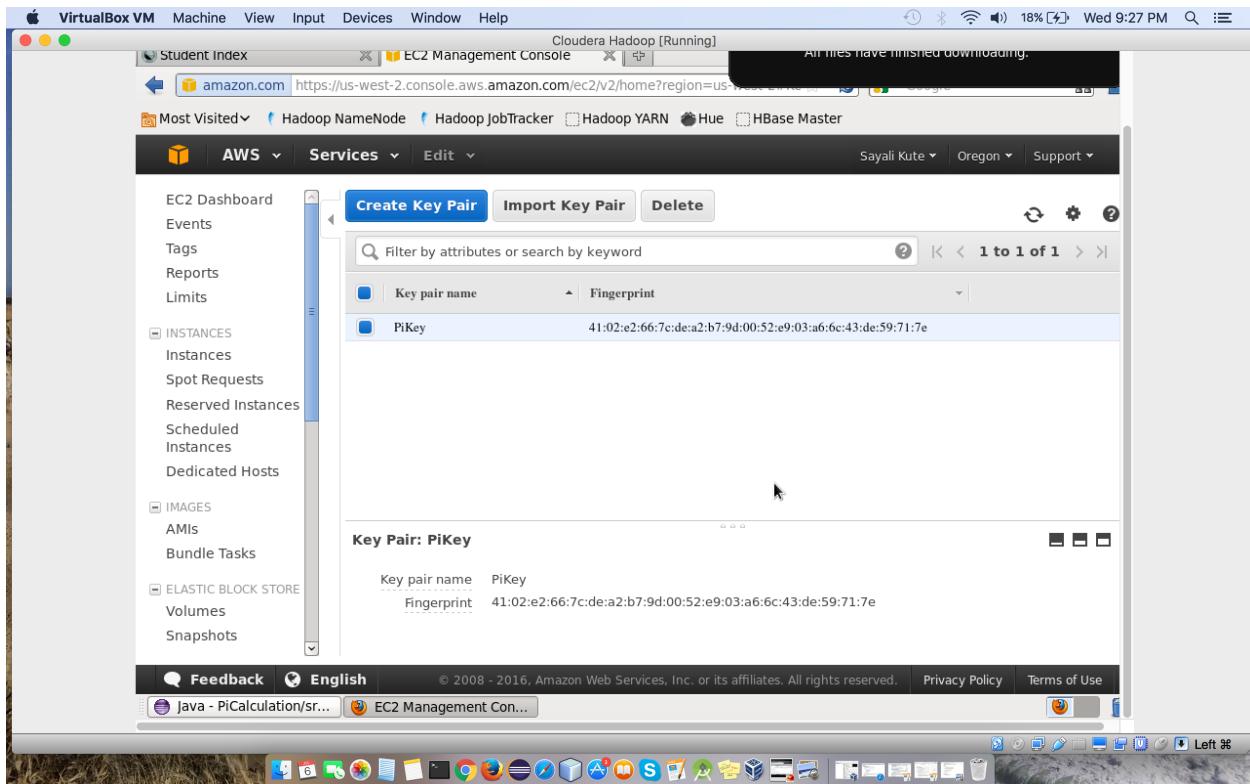


In AWS, create key pair through EC2 service:

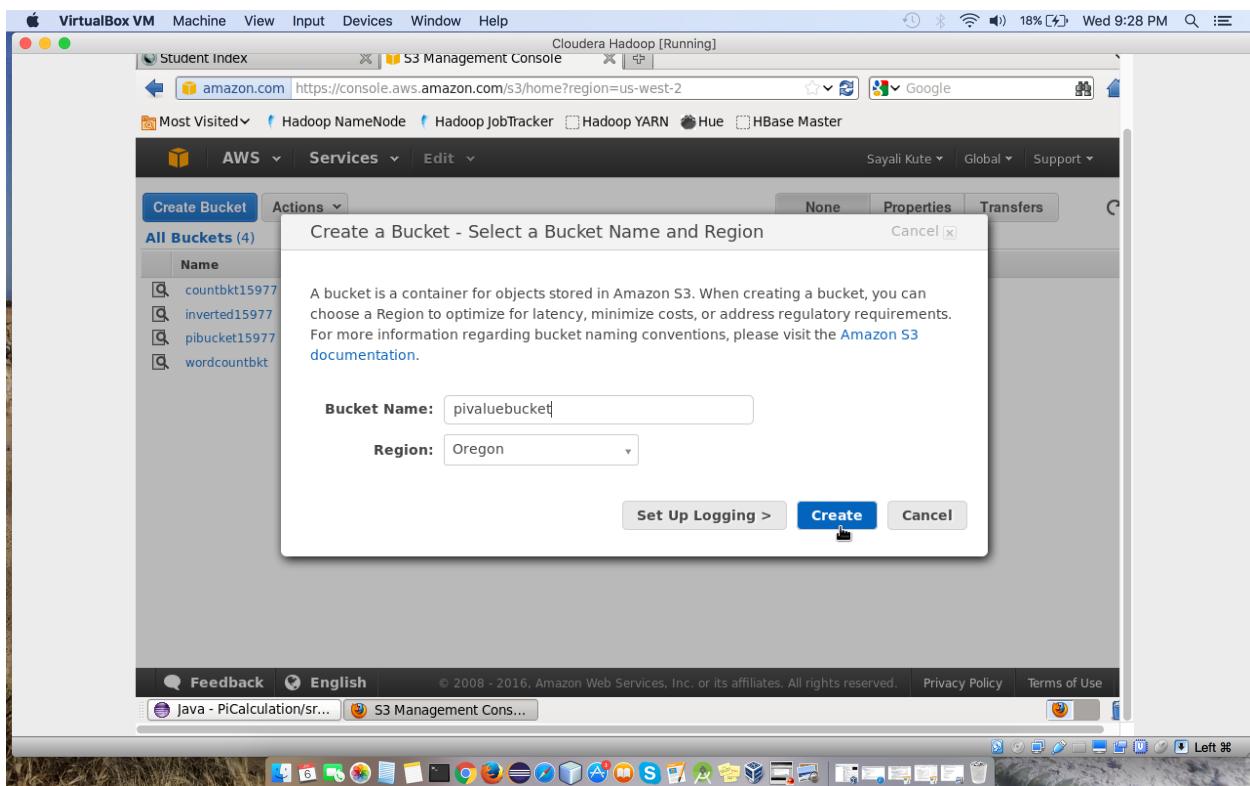


Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com



Create Bucket through S3 service:



Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

The screenshot shows the AWS S3 Management Console interface. At the top, there are tabs for 'Student Index' and 'S3 Management Console'. Below the tabs, the URL is https://console.aws.amazon.com/s3/home?region=us-west-2. The navigation bar includes links for 'Most Visited', 'Hadoop NameNode', 'Hadoop JobTracker', 'Hadoop YARN', 'Hue', and 'HBase Master'. On the left, there's a sidebar with 'AWS Services' and 'Edit' dropdowns, and user information 'Sayali Kute', 'Global', and 'Support'. The main area displays the 'All Buckets / pivaluebucket' view. It has a table with columns: Name, Storage Class, Size, and Last Modified. The table lists four folders: 'data', 'job', 'logs', and 'result', all with 'Standard' storage class and zero size. There are buttons for 'Upload', 'Create Folder', and 'Actions'. A search bar says 'Search by prefix' and filter buttons for 'None', 'Properties', and 'Transfers'. The bottom of the screen shows the Mac OS X desktop with various application icons.

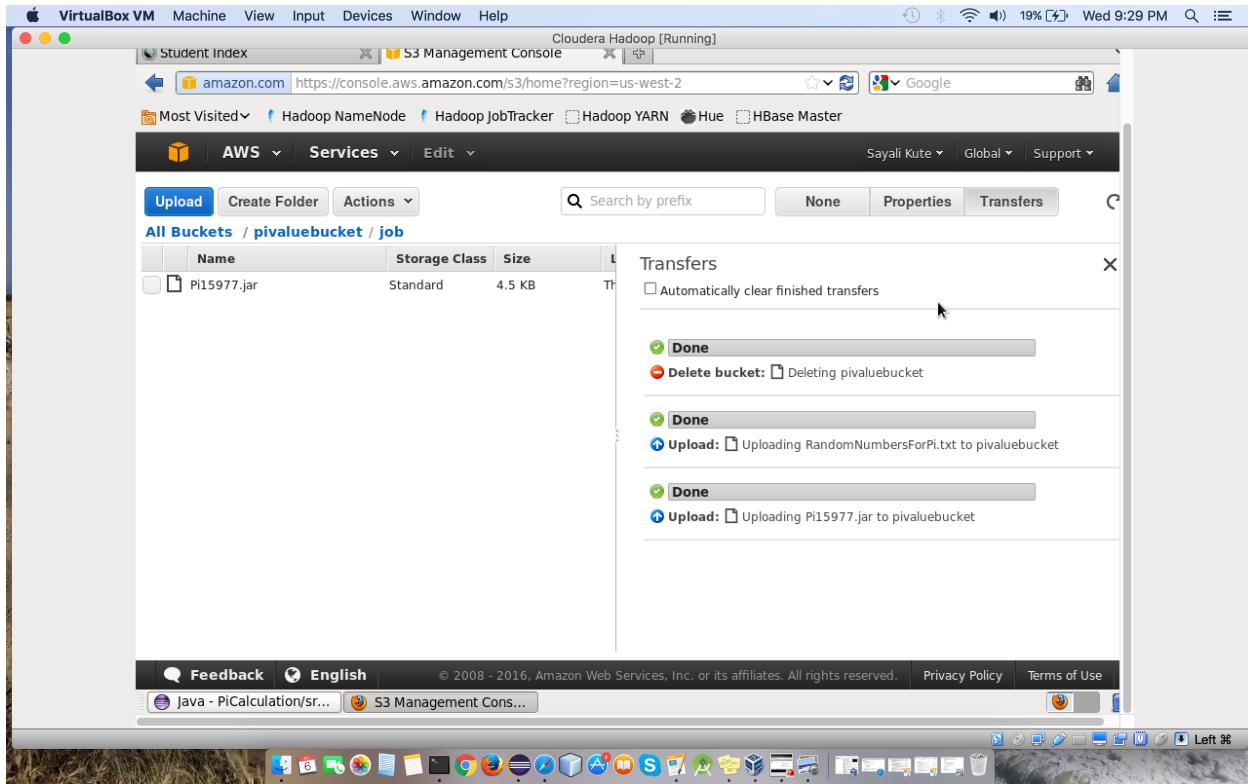
Upload the RandomGeneratorForPi.txt file in ‘data’ folder and exported Pi15977.jar file in the ‘job’ folder.

This screenshot shows the AWS S3 Management Console with the 'pivaluebucket / data' view. The table in the main area shows one file: 'RandomNumbersForPi.txt' with a size of 386.9 KB. To the right, a 'Transfers' panel is open, displaying a list of completed operations. It includes a section for 'Automatically clear finished transfers' with an unchecked checkbox. The transfer history shows two entries: a successful 'Delete bucket' operation for 'pivaluebucket' and a successful 'Upload' operation for 'RandomNumbersForPi.txt' into the 'pivaluebucket'. The bottom of the screen shows the Mac OS X desktop with various application icons.

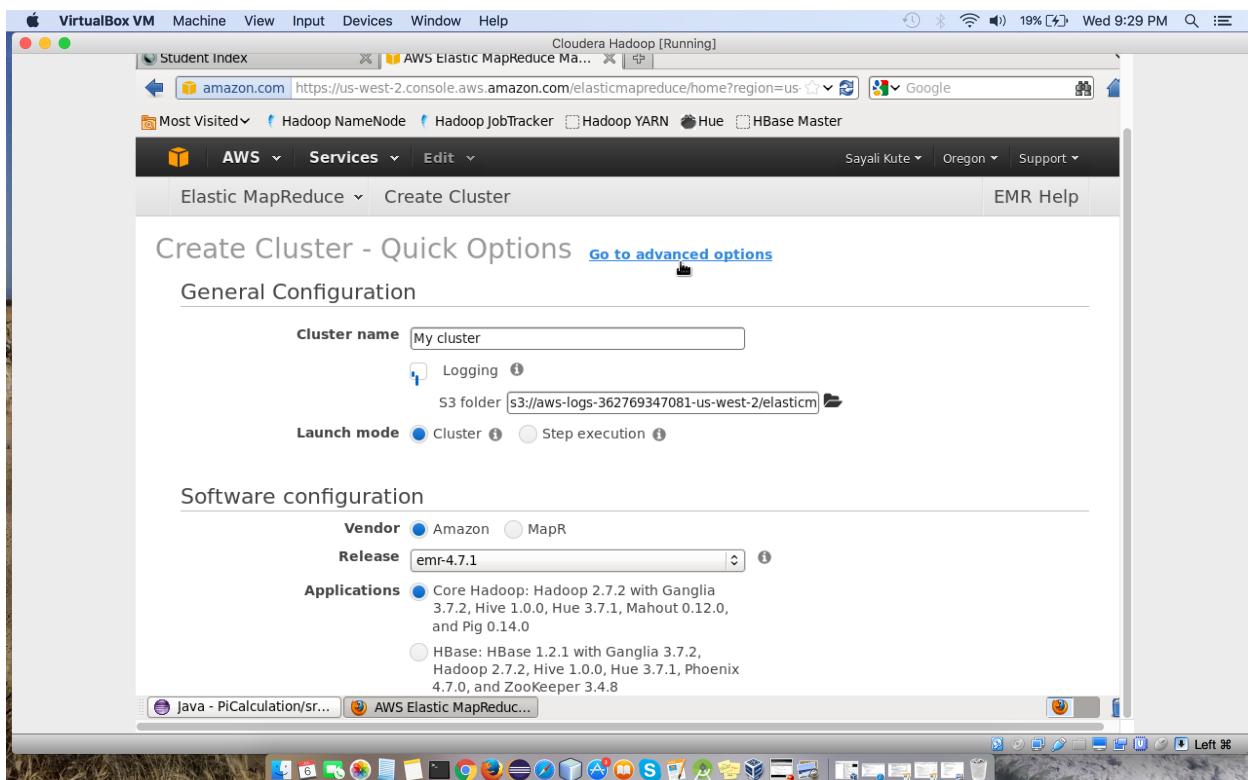
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Create a cluster through EMR service and select all the required



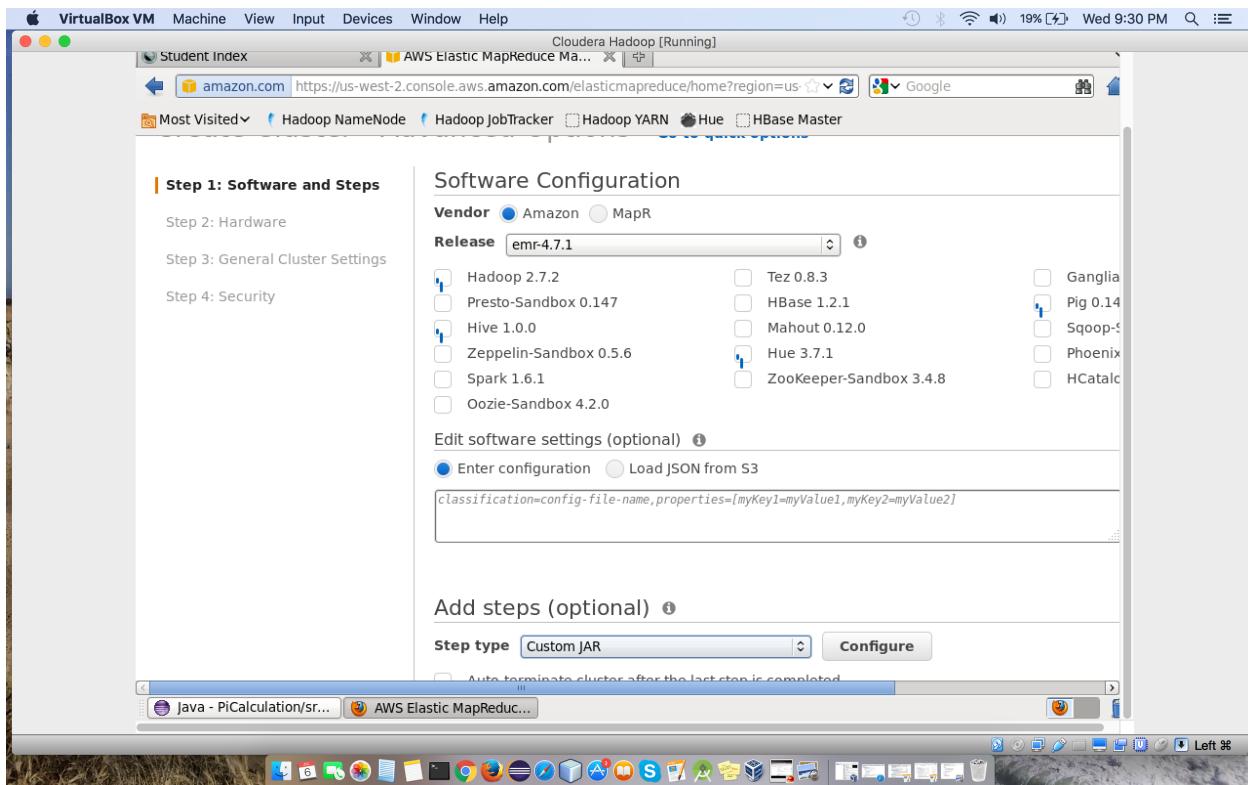
advance choices:



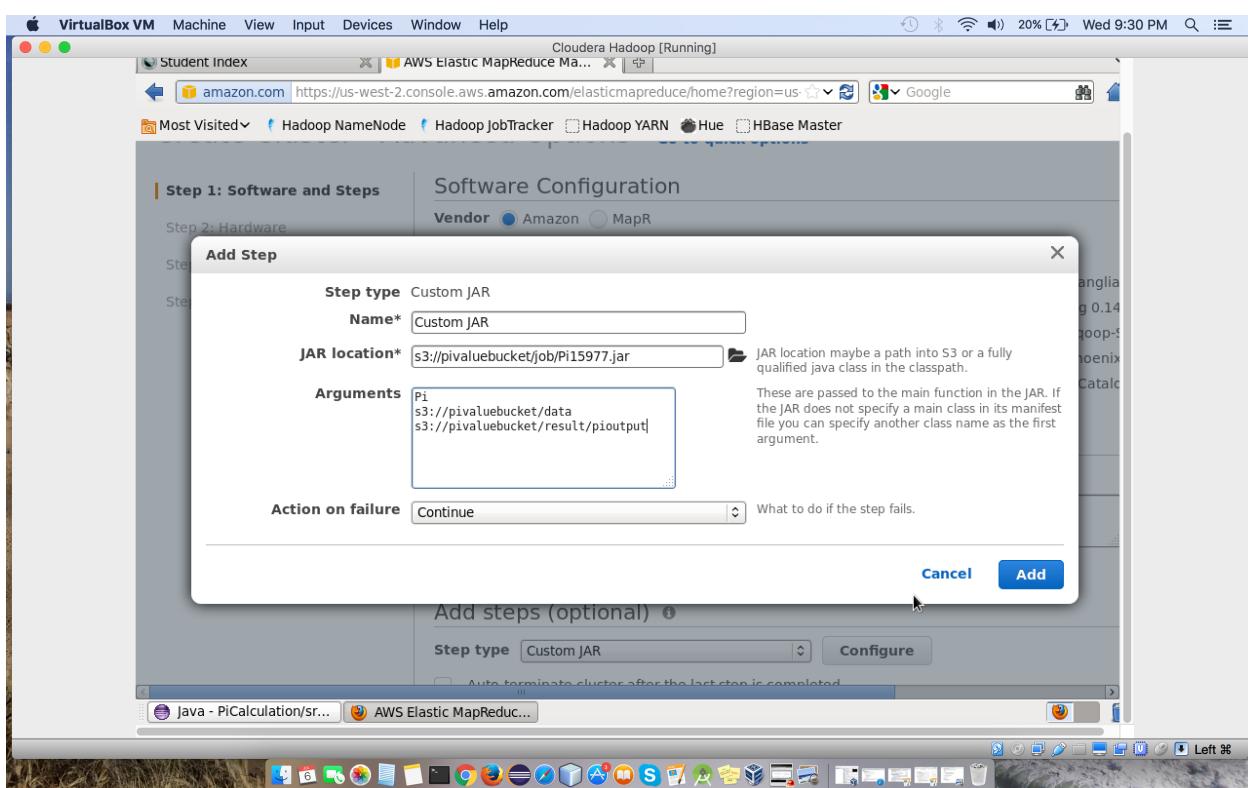
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Select ‘step type’ as ‘Custom JAR’ and configure other settings as follows:



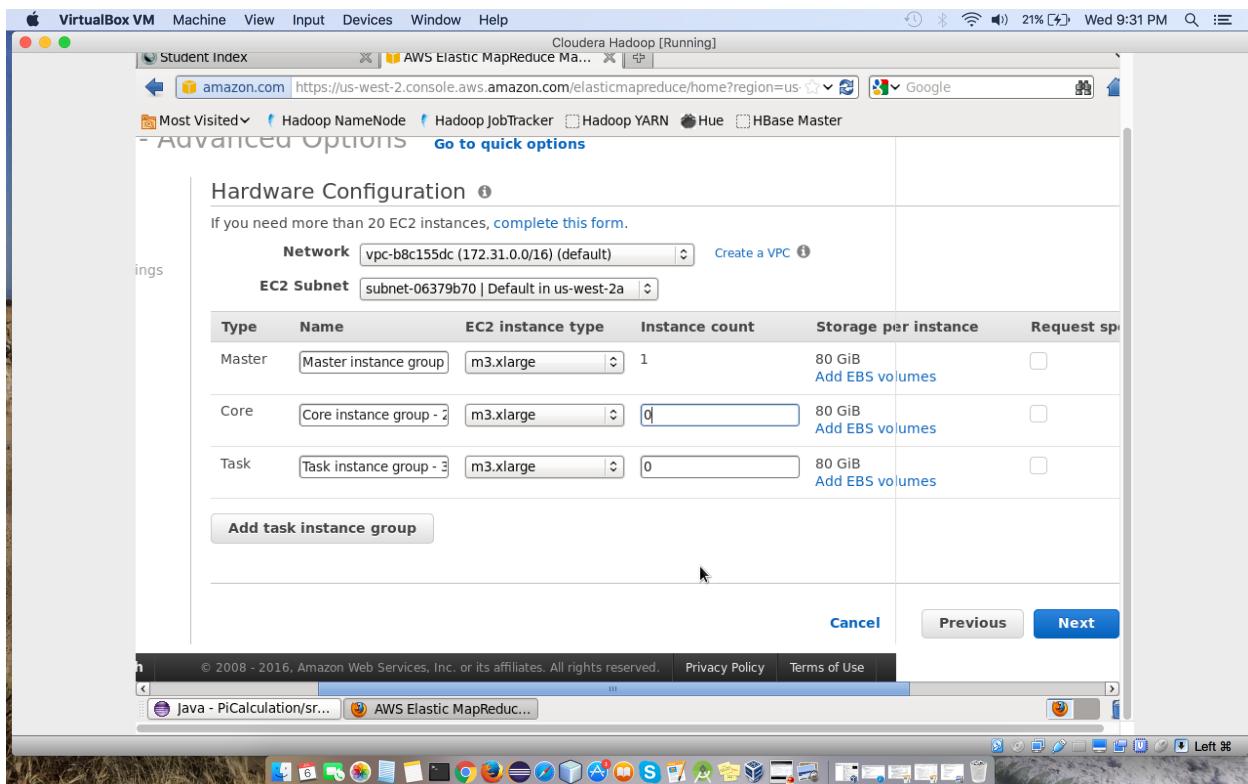
Provide JAR file location path and arguments:



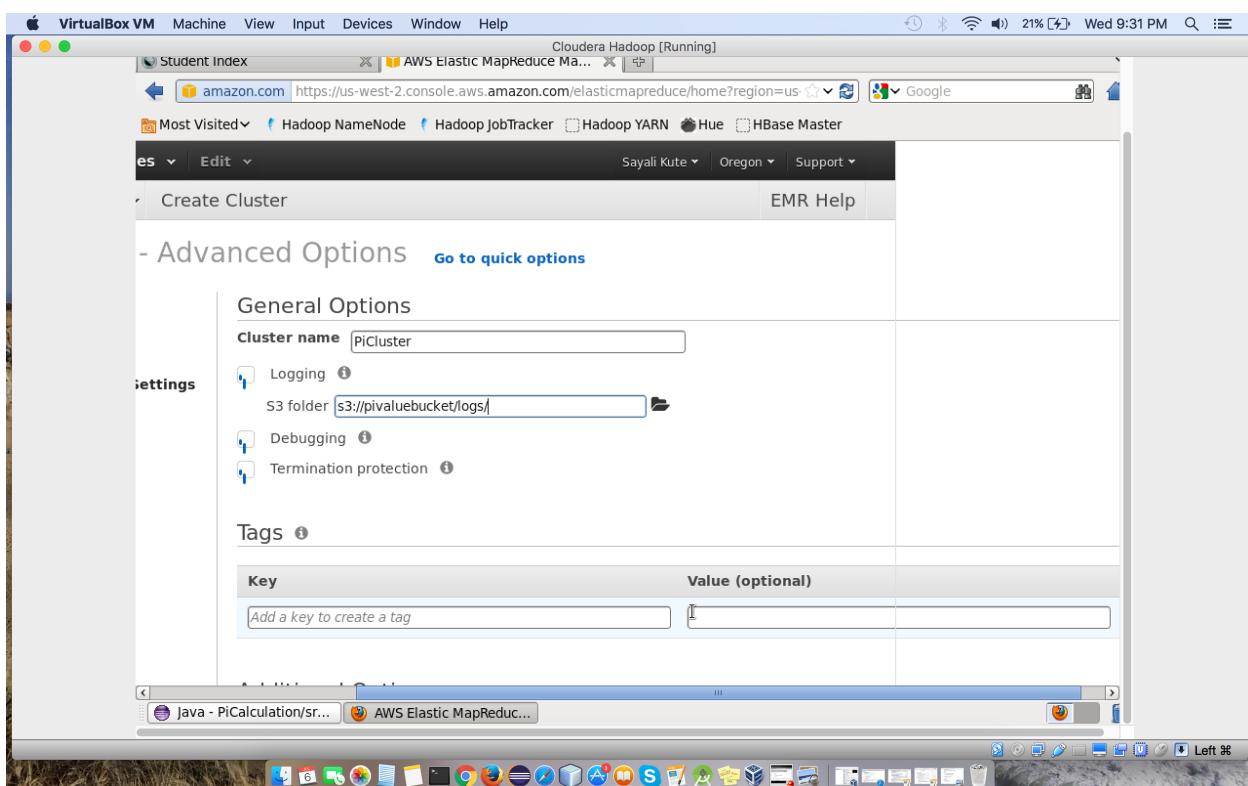
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Make instance count of 'core' as zero:



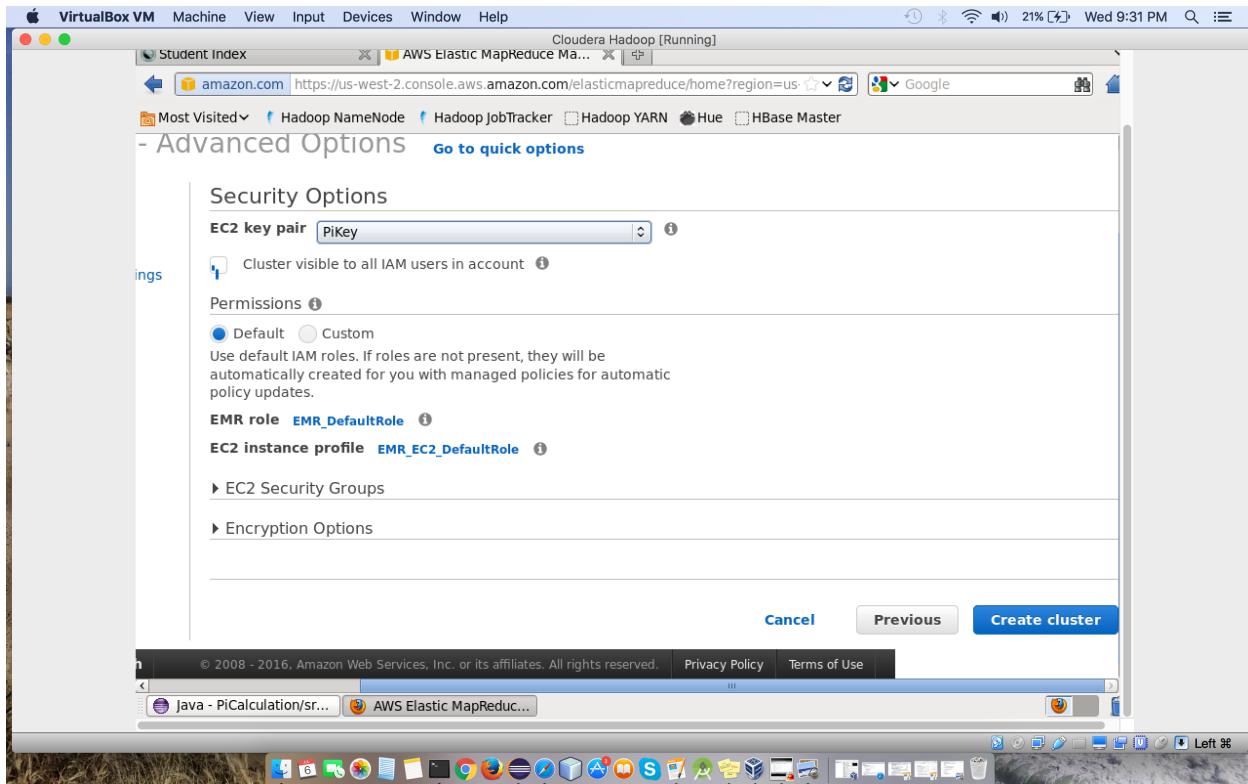
Provide the log folder path present in the S3 Service:



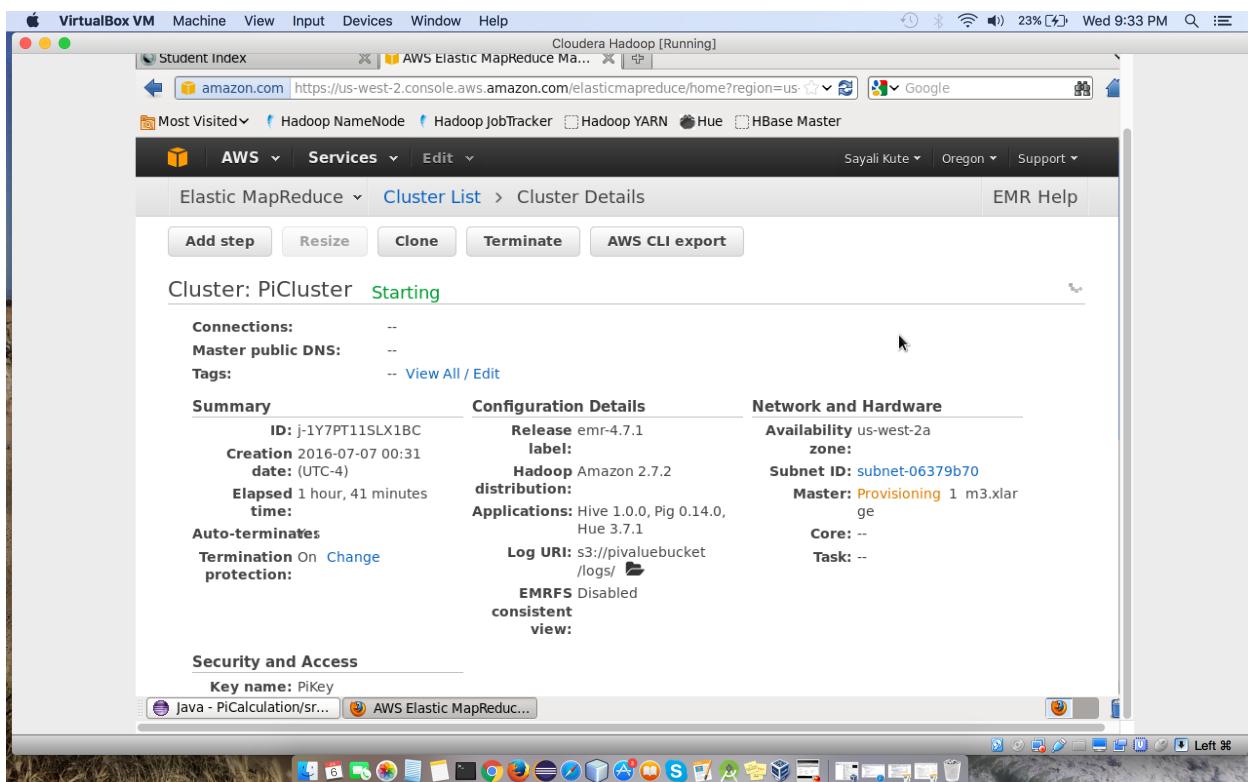
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Select the appropriate EC2 key pair:



Cluster has started working and runs successfully :



Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

The screenshot shows the AWS Elastic MapReduce Cluster Details page. The cluster is named "PiCluster" and is listed as "Terminated". The "Summary" section includes details like ID: j-1Y7PT11SLX1BC, Creation: 2016-07-07 00:31, and End date: 2016-07-07 00:40. The "Configuration Details" section shows Release: emr-4.7.1, Hadoop: Amazon 2.7.2, Applications: Hive 1.0.0, Pig 0.14.0, Hue 3.7.1, and Log URI: s3://pivaluebucket/logs/. The "Network and Hardware" section indicates Availability: us-west-2a, zone: subnet-06379b70, Master: Bootstrapping, and Core: --. The "Security and Access" section shows Key name: PiKey.

The output is generated in the ‘results’ folder of S3 Service:

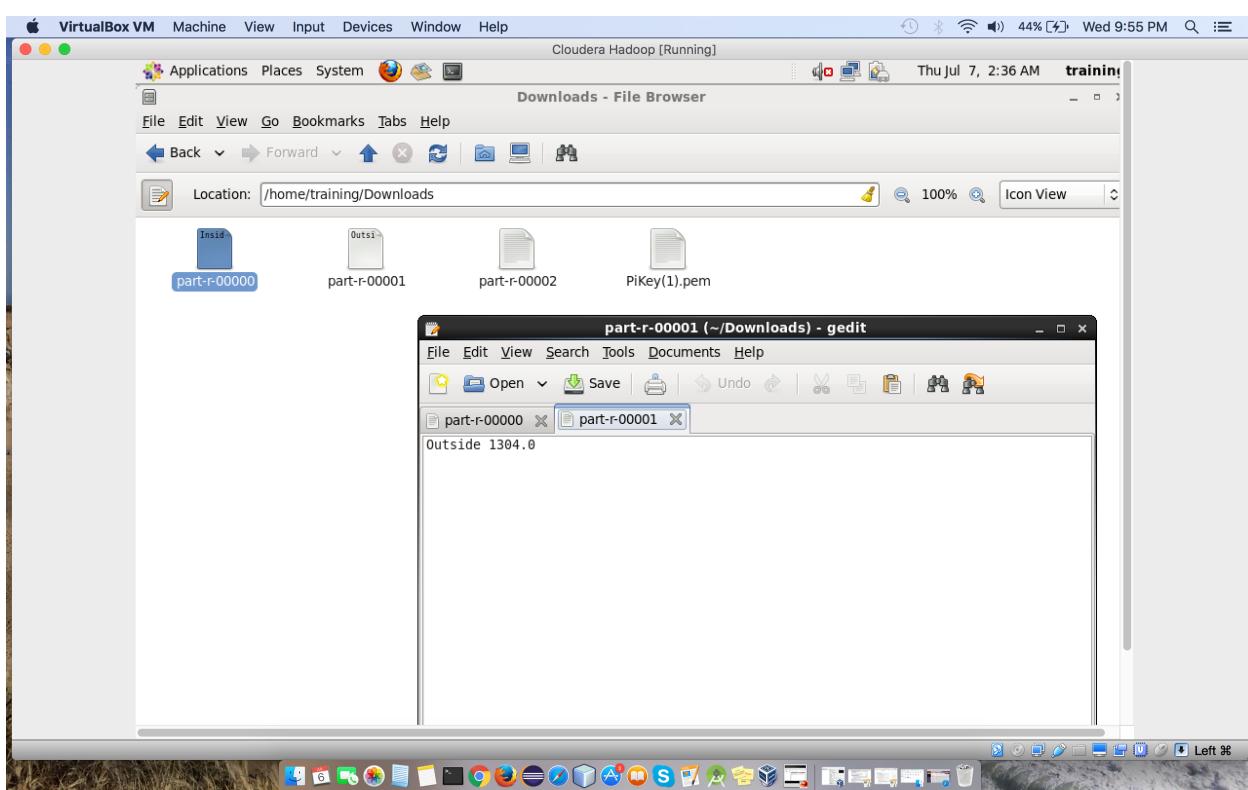
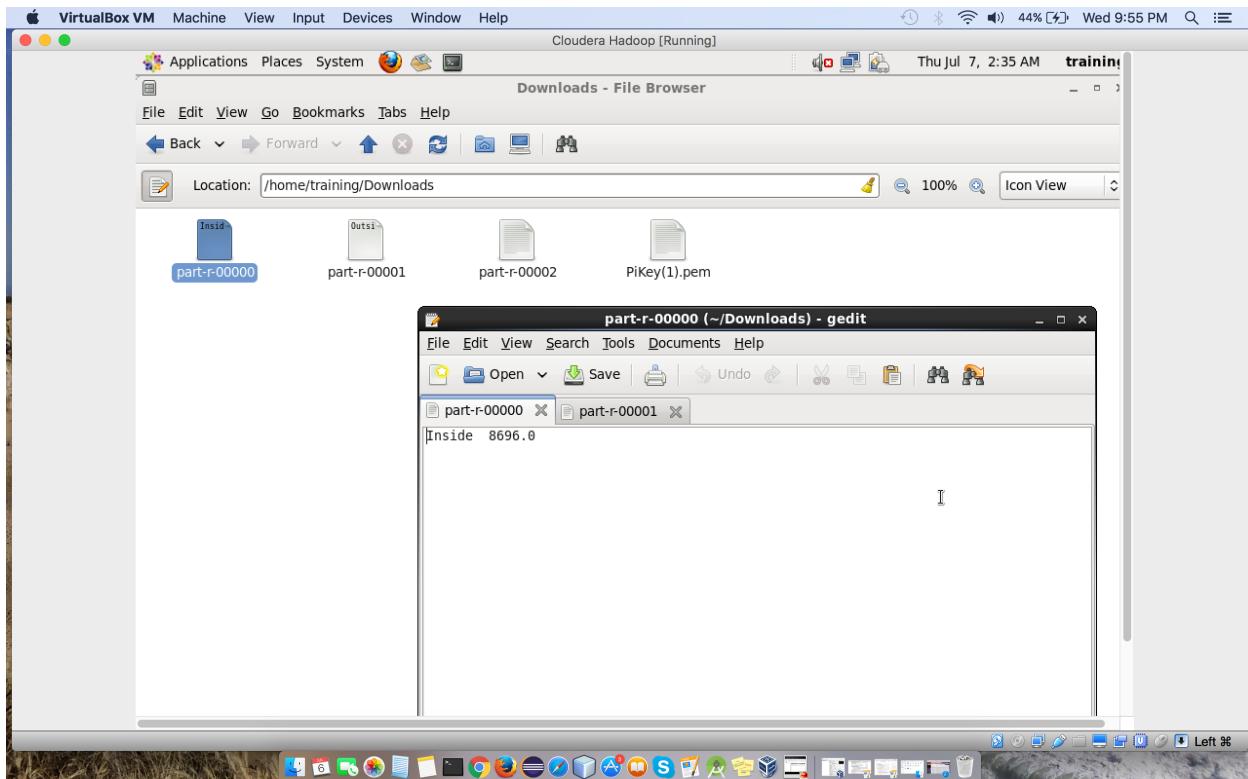
The screenshot shows the AWS S3 Management Console. The user is viewing the contents of the "pioutput" folder within the "result" folder of the "pivaluebucket" bucket. The table lists four objects: "_SUCCESS" (0 bytes, Standard storage class), "part-r-00000" (14 bytes, Standard storage class), "part-r-00001" (15 bytes, Standard storage class), and "part-r-00002" (0 bytes, Standard storage class). The objects were last modified on Thu Jul 07 00:39:42 GMT-400 201.

Name	Storage Class	Size	Last Modified
_SUCCESS	Standard	0 bytes	Thu Jul 07 00:39:42 GMT-400 201
part-r-00000	Standard	14 bytes	Thu Jul 07 00:39:39 GMT-400 201
part-r-00001	Standard	15 bytes	Thu Jul 07 00:39:41 GMT-400 201
part-r-00002	Standard	0 bytes	Thu Jul 07 00:39:42 GMT-400 201

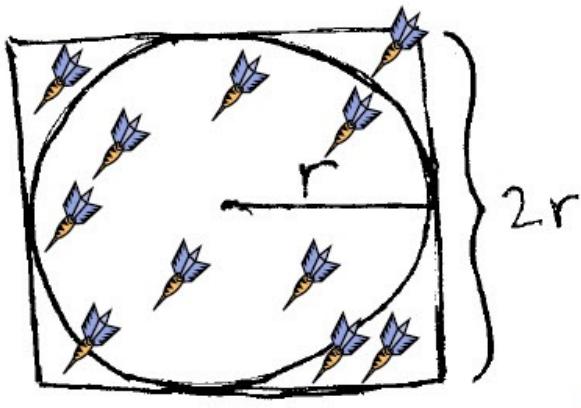
Name: Sayali Vishnu Kute

Email ID: sayalikute2507@gmail.com

Calculation of PI value:



- Throw N darts on the board. Each dart lands at a random position (x,y) on the board.



- Note if each dart landed inside the circle or not

- Check if $x^2 + y^2 < r$

- Take the total number of darts that landed in the circle as S

$$4 \left(\frac{S}{N} \right) = \pi$$

Output File (pioutput): This file contains value listed as:

Inside : 8696.0

Outside : 1304.0

Formula to calculate pi : $4 * (S/N) = \pi$

where, S = Number of darts hit inside the circle

N = Total number of darts thrown

$$\text{Pi} = 4 * (\text{Inside} / (\text{Inside} + \text{Outside}))$$

$$\text{Pi} = 4 * (8696.0 / (8696.0 + 1304.0))$$

$$\text{Pi} = 4 * (8696.0 / 10000)$$

$$\text{Pi} = 4 * 0.8696$$

$$\text{Pi} = 3.4784 \text{ (approximately)}$$

—END—