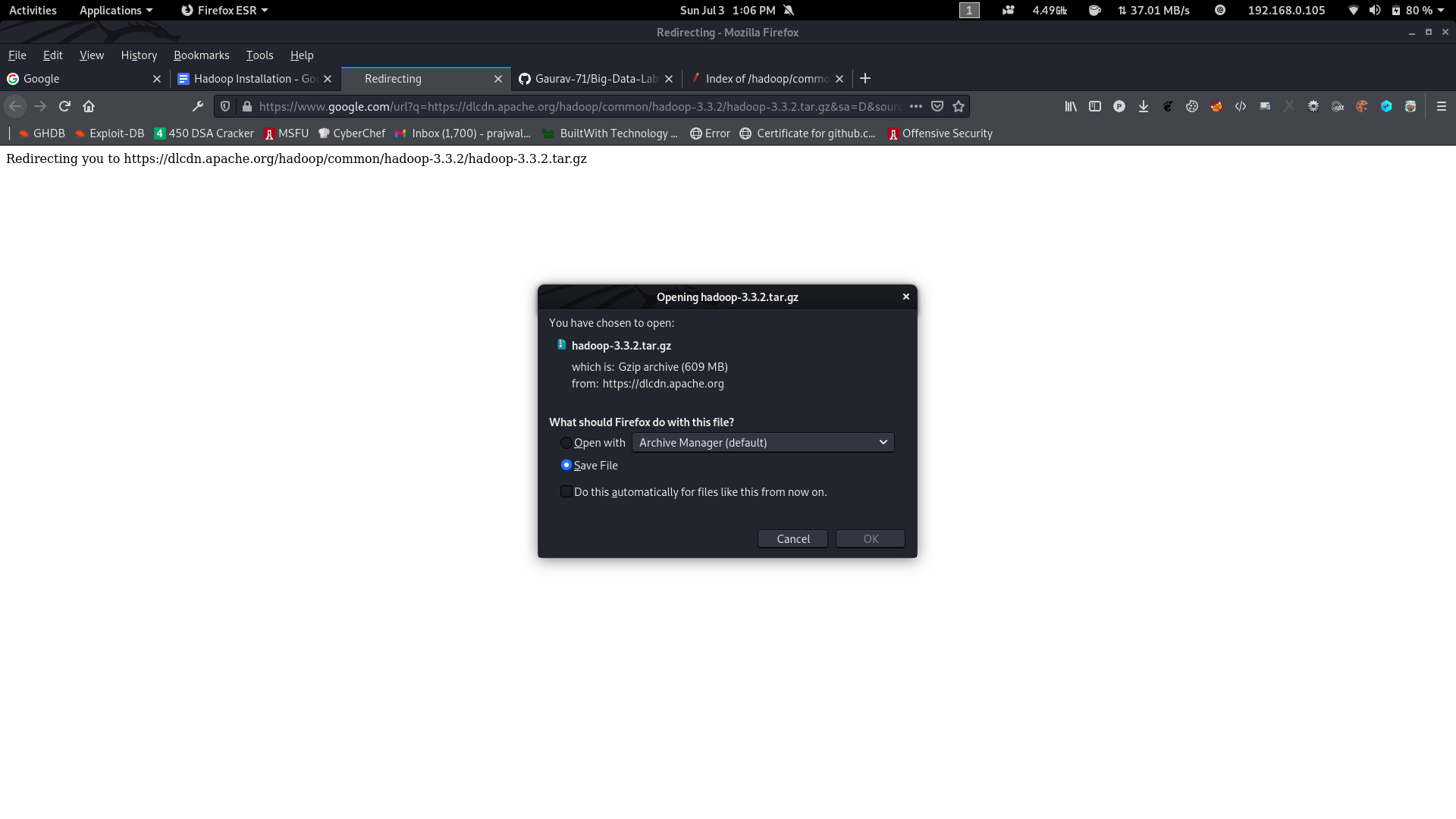
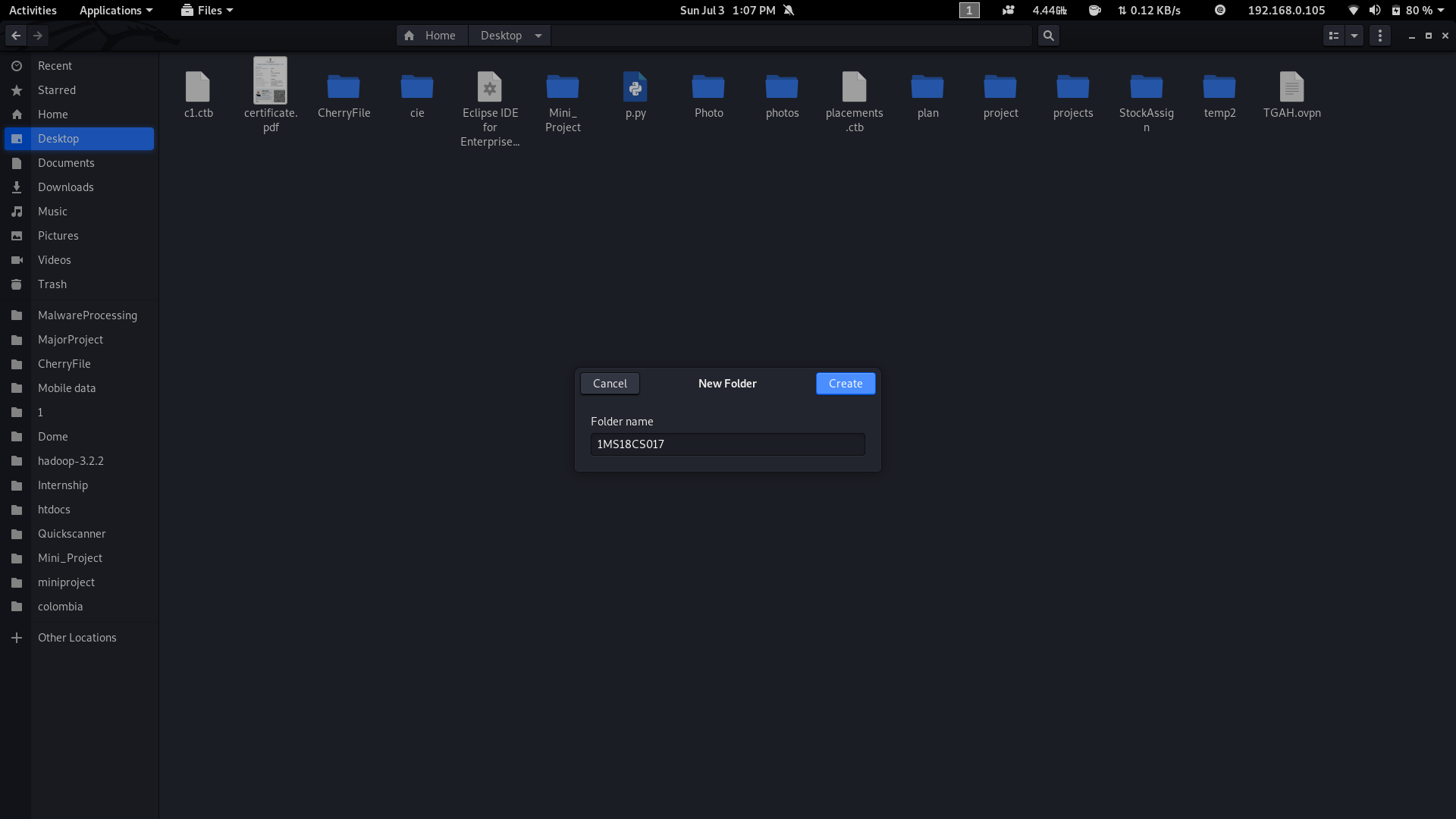
**Hadoop Installation**

1.Download Hadoop file from the following link from any Browser

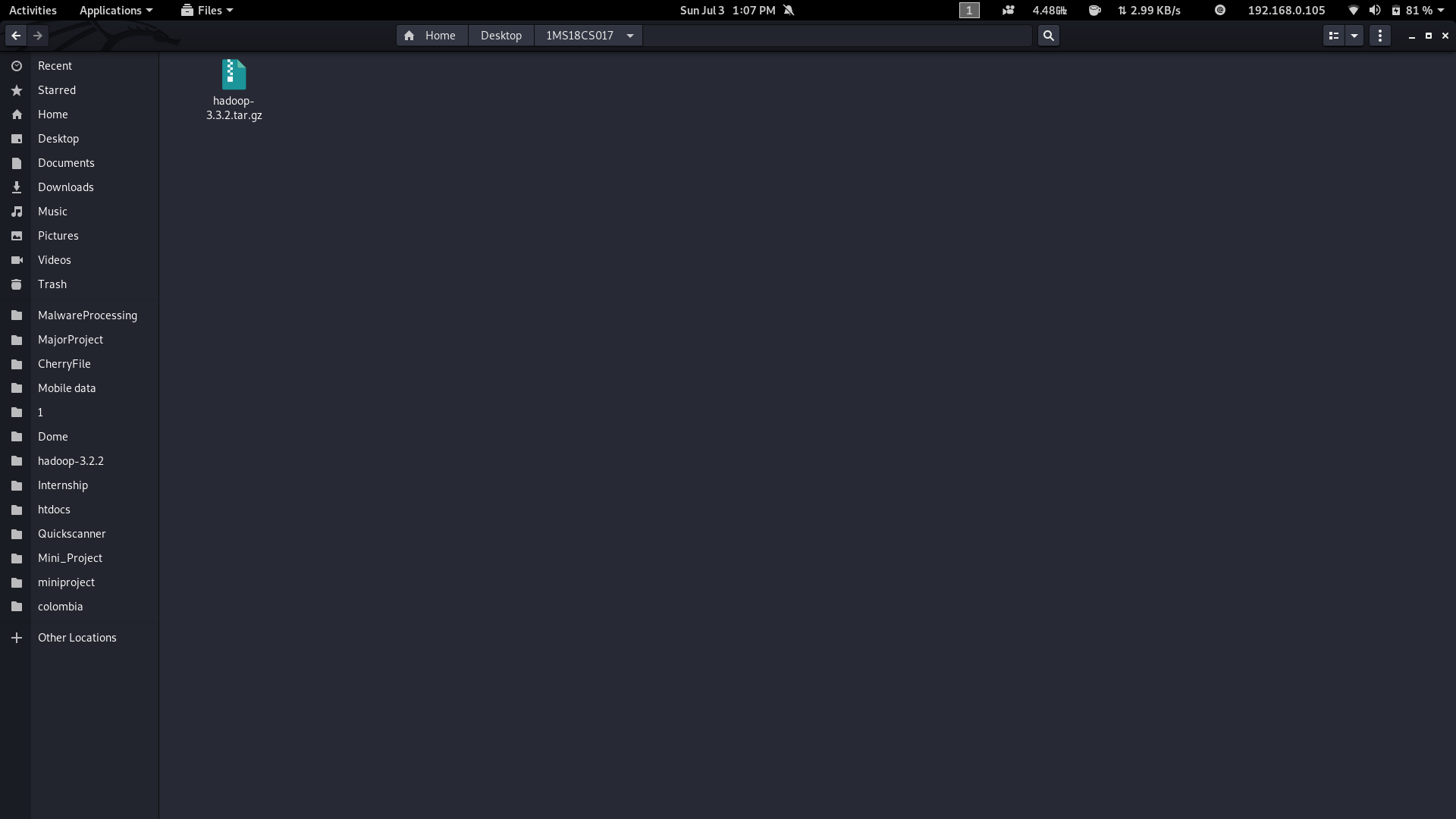
<https://dlcdn.apache.org/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz>



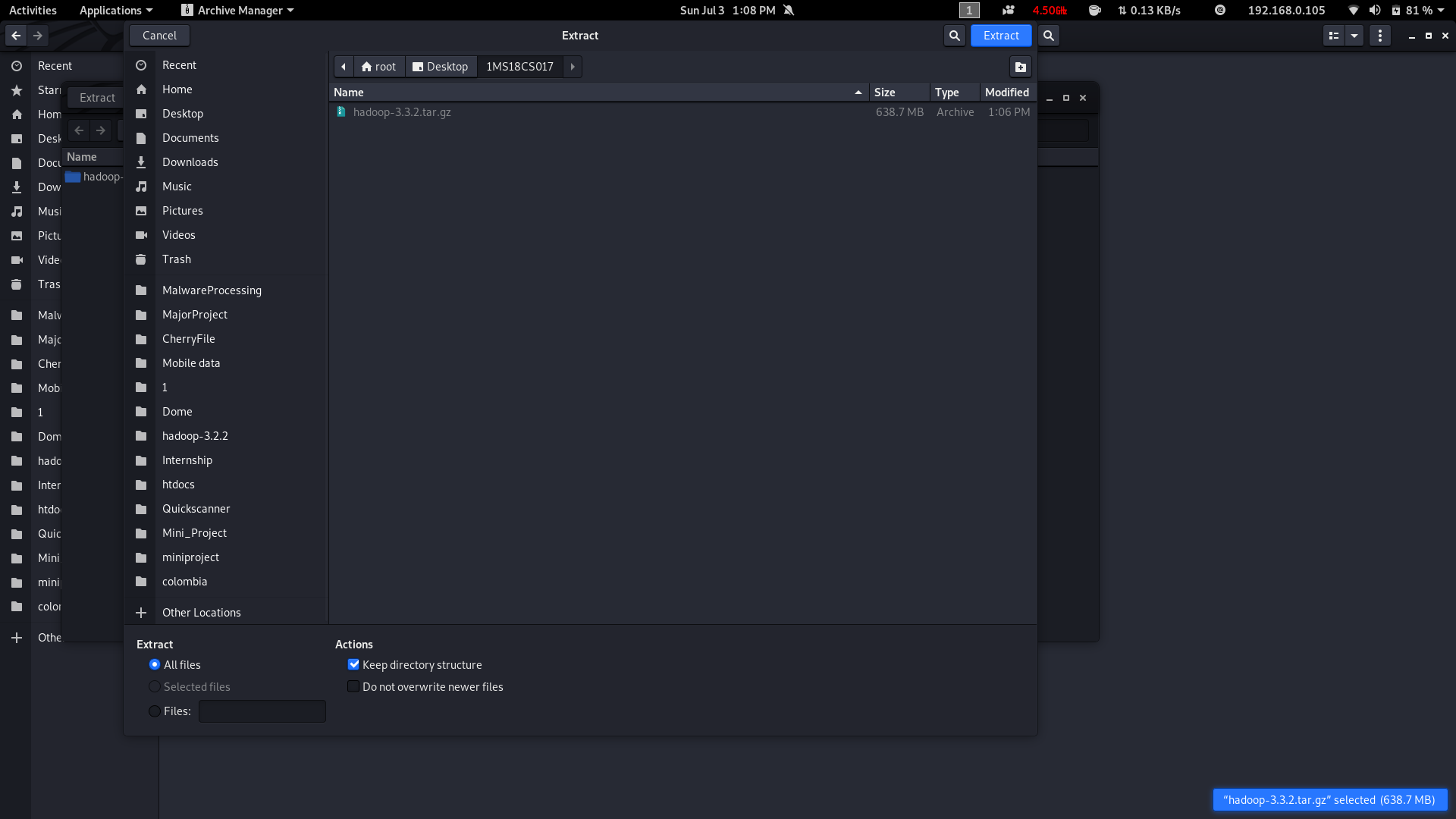
2. Create a new Folder inside Desktop , name the Folder as your USN <1ms18cs017>.



3. Move the Downloaded Hadoop File to USN <1ms18cs017> Folder.

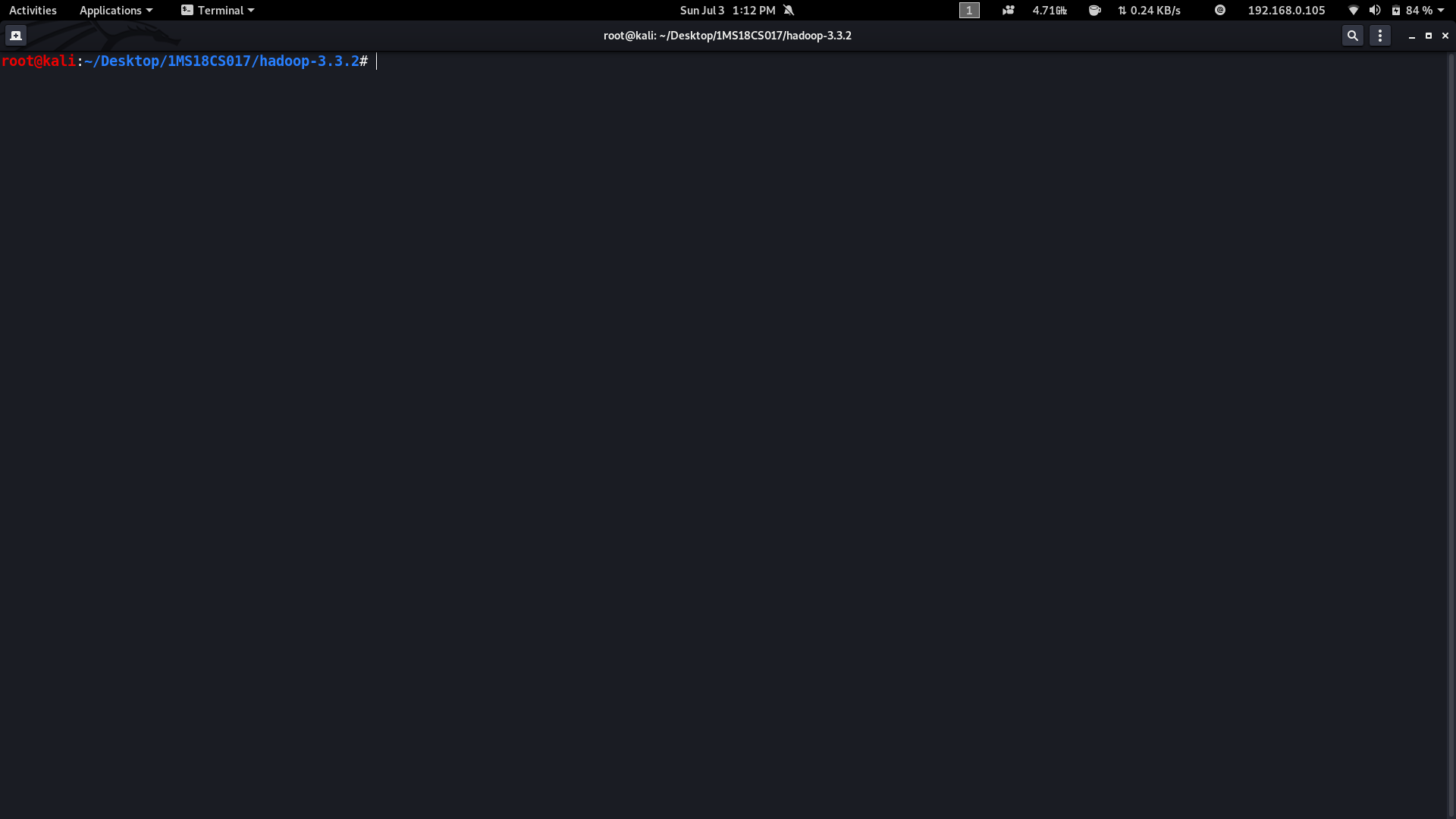


4. Right Click on that File and Extract inside the USN <1ms18cs017> Folder.

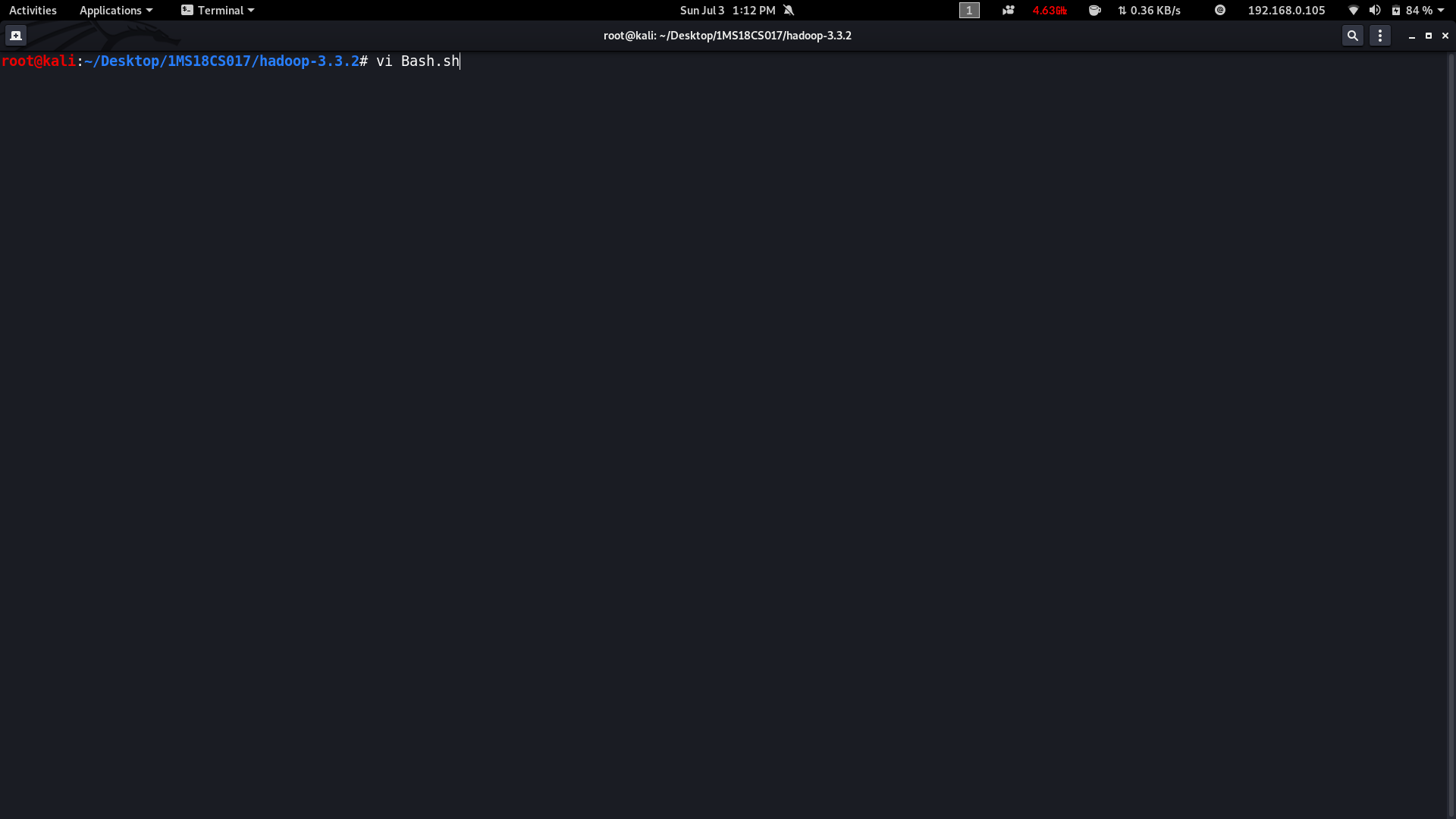


5. Open Terminal

6. Navigate to Extracted Hadoop Folder cd ~/Desktop/<1ms18cs017>/hadoop-3.3.2

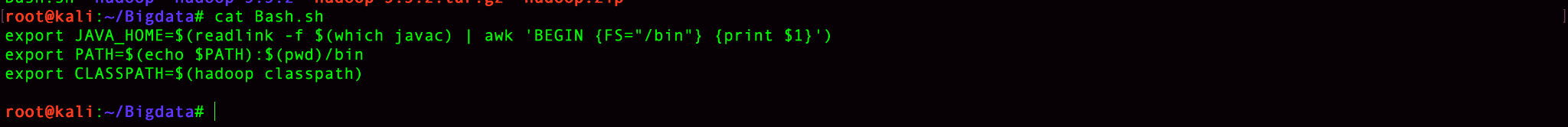


7. Create a New File named Bash.sh



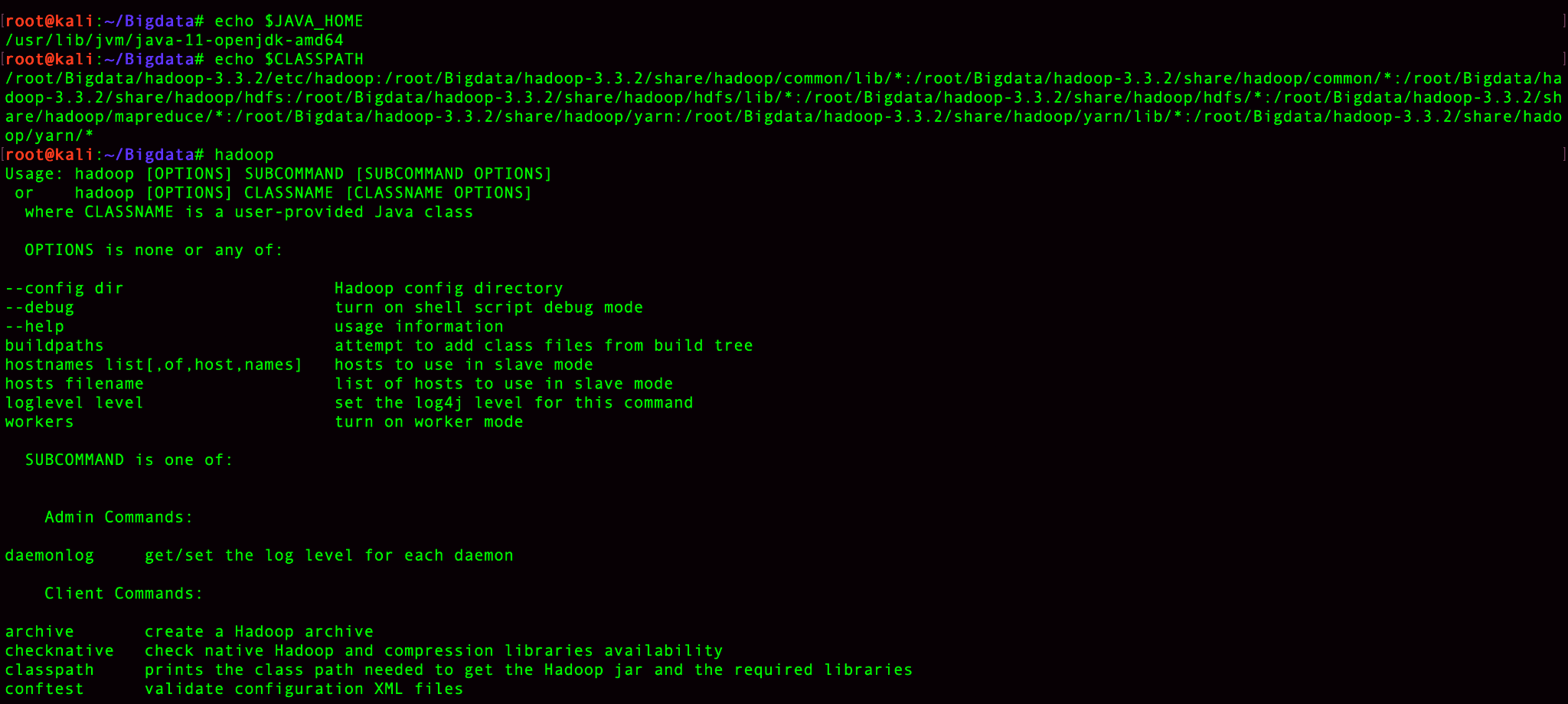
8. Copy the Below code and Paste inside Bash.sh and save that File.

| export JAVA\_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')  export PATH=$(echo $PATH):$(pwd)/bin  export CLASSPATH=$(hadoop classpath) |
| --- |



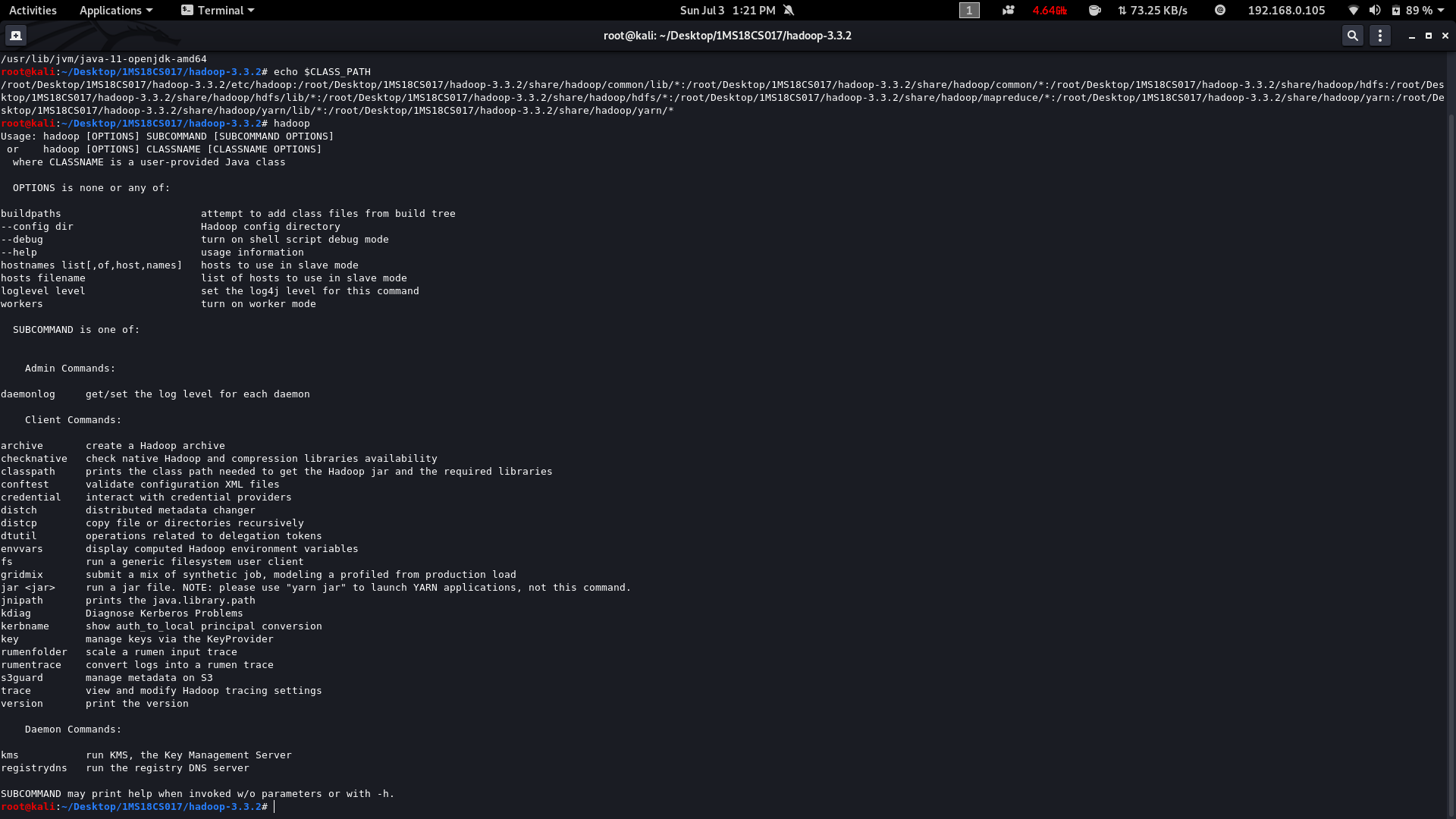
9. Execute the bash.sh File using following command **source Bash.sh.**

**NOTE: Make source before compiling or running hadoop compile this file.**



10. Verify JAVA\_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.

11. Verify Hadoop is Installed or not by executing **hadoop** command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.



**2. Hadoop Programs**

**2.1Odd Even Sum**

We need to create 3 java classes

→ Driver Class

→ Mapper Class

→ Reducer Class

**//driver.java**

package oddeven;

import java.io.\*;

import java.util.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package oddeven;

import java.io.\*;

import java.util.\*;

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>

{

public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable> output,Reporter r) throws IOException

{

String[] line=value.toString().split(" ");

for(String num:line){

int number=Integer.parseInt(num);

if(number%2==0) {

output.collect(new Text("even"),new IntWritable(number));

}

else{

output.collect(new Text("odd"),new IntWritable(number));

}

}

}

}

**//reducer.java**

package oddeven;

import java.io.\*;

import java.util.\*;

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

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

public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>

{

public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable> output ,Reporter r) throws IOException

{

int sum=0,count=0;

while(value.hasNext()){

sum+=value.next().get();

count++;

}

output.collect(new Text("Sum of "+key+" Numbers"),new IntWritable(sum));

output.collect(new Text(key+" Number count"),new IntWritable(count));

}

}

**Input → 1 2 3 4 5 6 7 8 9 10**

**Steps to run MapReduce programs**

**Compile all java files (driver.java mapper.java reducer.java)**

| javac -d . \*.java |
| --- |

The above command compiles all the java files and creates .class files inside the package.

**Set driver class in manifest**

echo Main-class: <packagename.driver> > Manifest.txt

Ex:

| echo Main-Class: oddeven.driver > Manifest.txt |
| --- |

**Create an executable jar file**

jar cfm <jar\_name>.jar Manifest.txt <package-name>/\*.class

Ex:

| jar cfm oddeven.jar Manifest.txt oddeven/\*.class |
| --- |

**Run the jar file**

hadoop jar <jarfilename>.jar <inputfile> <output>

Ex:

| hadoop jar oddeven.jar oe.txt output |
| --- |

Here oe.txt is input file for Oddeven create Input File

echo 1 2 3 4 5 6 7 8 9 10 > oe.txt

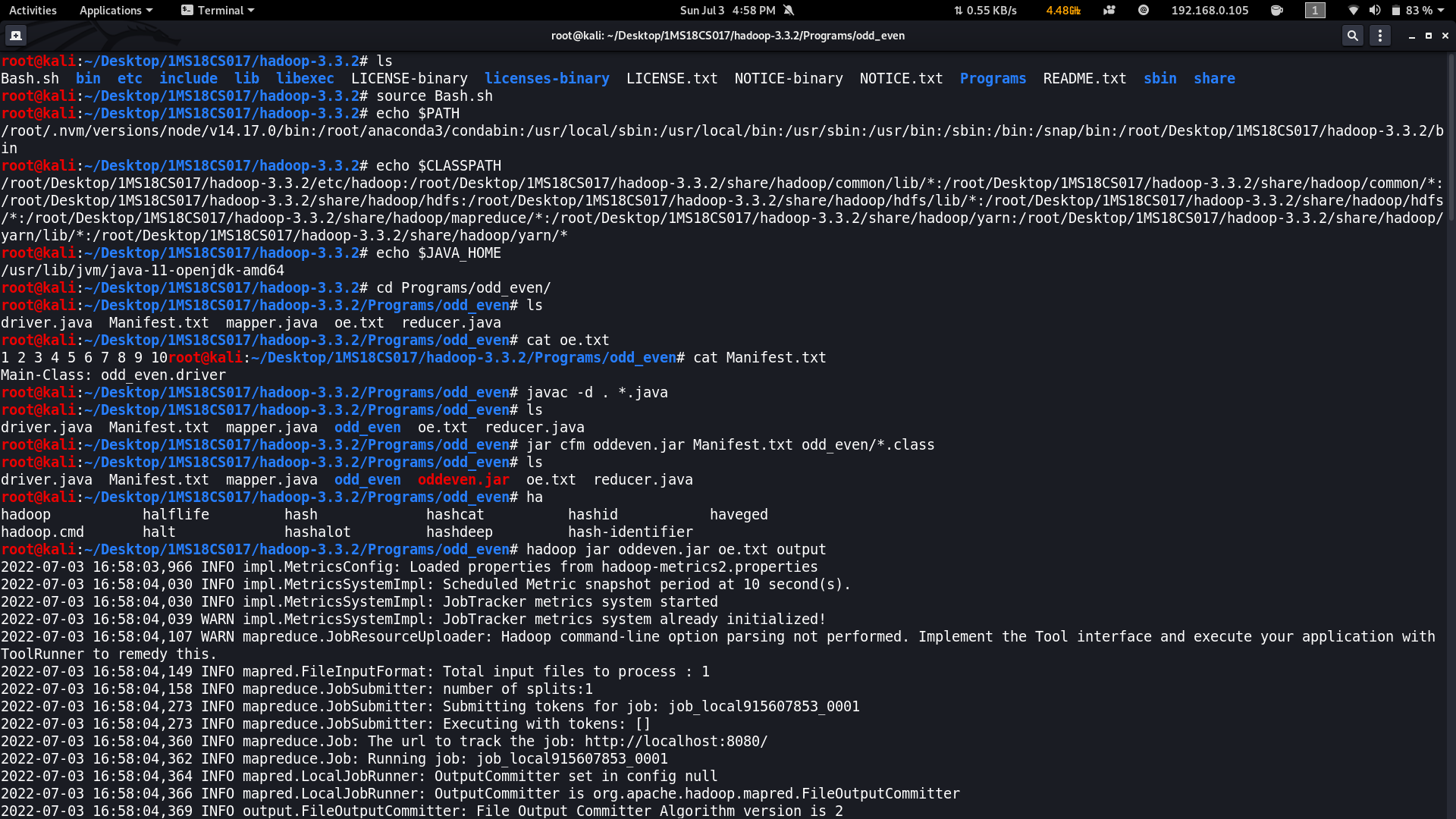
**To see the Output**

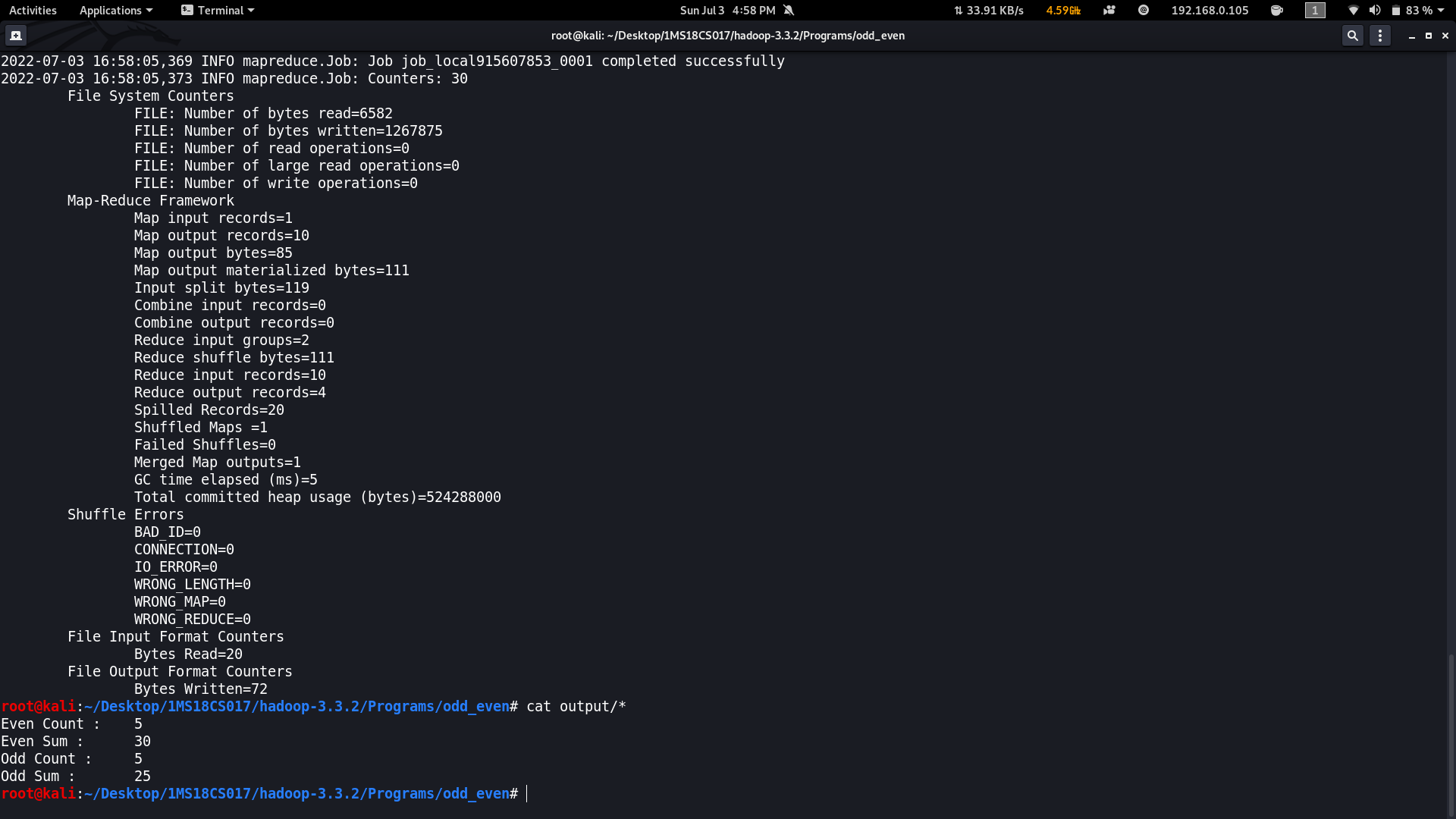
cat <output\_folder>/\*

| cat output/\* |
| --- |

**NOTE:** You Got any errors in import or hadoop not found or Java\_home not set then compile Bash.sh File. **source Bash.sh**

Input Files for All programs [https://drive.google.com/file/d/10qq09DsK6pQ\_I29AqEsq70ExaOyME5 5B/view?usp=sharing](https://drive.google.com/file/d/10qq09DsK6pQ_I29AqEsq70ExaOyME55B/view?usp=sharing)

****

****

**2.2 Weather**

**//driver.java**

package weather;

import java.util.\*;

import java.io.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(DoubleWritable.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package weather;

import java.util.\*;

import java.io.\*;

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable, Text,Text,DoubleWritable>{

public void map(LongWritable key , Text value , OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException

{

String line=value.toString();

String year=line.substring(15,19);

Double temp=Double.parseDouble(line.substring(87,92));

output.collect(new Text(year), new DoubleWritable(temp));

}

}

**//reducer.java**

package weather;

import java.util.\*;

import java.io.\*;

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

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

class reducer extends MapReduceBase implements Reducer<Text,DoubleWritable,Text,DoubleWritable> {

public void reduce(Text key, Iterator<DoubleWritable> value, OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException{

Double max=-9999.0;

Double min=9999.0;

while(value.hasNext()){

Double temp=value.next().get();

max=Math.max(max,temp);

min=Math.min(min,temp);

}

output.collect(new Text("Max temp at "+ key), new DoubleWritable(max));

output.collect(new Text("Min temp at "+ key), new DoubleWritable(min));

}

}

**Input.txt**

0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999999

0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999

0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999

0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999

0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999999

**2.3 Earthquake**

**//driver.java**

package earthquake;

import java.util.\*;

import java.io.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(DoubleWritable.class);

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

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

JobClient.runJob(conf);

}

**//mapper.java**

package earthquake;

import java.util.\*;

import java.io.\*;

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable, Text,Text,DoubleWritable>

{

public void map(LongWritable key , Text value , OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException

{

String[] line=value.toString().split(",");

Double longi=Double.parseDouble(line[7]);

output.collect(new Text(line[11]), new DoubleWritable(longi));

}

}

**//reducer.java**

package earthquake;

import java.util.\*;

import java.io.\*;

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

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

class reducer extends MapReduceBase implements Reducer<Text,DoubleWritable,Text,DoubleWritable> {

public void reduce(Text key, Iterator<DoubleWritable> value, OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException

{

Double max=-9999.0;

while(value.hasNext())

{

Double temp=value.next().get();

max=Math.max(max,temp);

}

output.collect(new Text(key), new DoubleWritable(max));

}

}

**2.4 Insurance**

**//driver.java**

package insurance;

import java.io.\*;

import java.util.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package insurance;

import java.io.\*;

import java.util.\*;

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>

{

public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable> output,Reporter r) throws IOException

{

String[] line=value.toString().split(",");

output.collect(new Text(line[2]),new IntWritable(1));

}

}

**//reducer.java**

package insurance;

import java.io.\*;

import java.util.\*;

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

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

public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>

{

public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable> output ,Reporter r) throws IOException

{

int sum=0;

while(value.hasNext())

{

sum+=value.next().get();

}

output.collect(key,new IntWritable(sum));

}

}

**2.5 Sales**

**//driver.java**

package sales;

import java.io.\*;

import java.util.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package sales;

import java.io.\*;

import java.util.\*;

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>

{

public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable> output,Reporter r) throws IOException

{

String[] line=value.toString().split(",");

int price=Integer.parseInt(line[2]);

String cardtype=line[3];

String Country=line[7];

output.collect(new Text("Country "+Country),new IntWritable(price));

output.collect(new Text("CardType "+cardtype),new IntWritable(1));

}

}

**//reducer.java**

package sales;

import java.io.\*;

import java.util.\*;

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

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

public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>

{

public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable> output ,Reporter r) throws IOException

{

int sum=0;

while(value.hasNext())

{

sum+=value.next().get();

}

output.collect(new Text(key),new IntWritable(sum));

}

}

**2.6 Employee**

**//driver.java**

package employee;

import java.io.\*;

import java.util.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(DoubleWritable.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package employee;

import java.io.\*;

import java.util.\*;

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

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

class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , DoubleWritable> {

public void map(LongWritable key, Text value, OutputCollector<Text,DoubleWritable> output ,Reporter r) throws IOException

{

String[] line=value.toString().split("\\t");

salary=Double.parseDouble(line[8]);

output.collect(new Text(line[3]), new DoubleWritable(salary));

}

}

**//redducer.java**

package employee;

import java.io.\*;

import java.util.\*;

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

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

class reducer extends MapReduceBase implements Reducer<Text,DoubleWritable,Text,DoubleWritable> {

public void reduce(Text key,Iterator<DoubleWritable> value , OutputCollector<Text,DoubleWritable> output ,Reporter r) throws IOException

{

int count=0;

Double sum=0.0;

while(value.hasNext()){

sum+=value.next().get();

count+=1;

}

output.collect(new Text(key+" Average"), new DoubleWritable(sum/count));

output.collect(new Text(key+" Count"), new DoubleWritable(count));

}

}

**2.7 Matrix Multiplication**

**//driver.java**

package matrix;

import java.util.\*;

import java.io.\*;

import org.apache.hadoop.fs.Path;

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

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

public class driver{

public static void main(String args[]) throws IOException

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(Text.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package matrix;

import java.util.\*;

import java.io.\*;

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

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

class mapper extends MapReduceBase implements Mapper<LongWritable, Text, Text,Text>

{

public void map(LongWritable key, Text value, OutputCollector<Text,Text> output, Reporter r) throws IOException

{

String line[]=value.toString().split(",");

Text OutputKey=new Text();

Text OutputValue=new Text();

if(line[0].equals("A"))

{

for(int i=0;i<3;i++)

{

OutputKey.set(line[1]+","+i);

OutputValue.set("A,"+line[2]+","+line[3]);

output.collect(OutputKey,OutputValue);

}

}

else

{

for(int i=0;i<2;i++)

{

OutputKey.set(i+","+line[2]);

OutputValue.set("B,"+line[1]+","+line[3]);

output.collect(OutputKey,OutputValue);

}

}

}

}

**//reducer.java**

package matrix;

import java.util.\*;

import java.io.\*;

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

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

public class reducer extends MapReduceBase implements Reducer<Text,Text,Text,Text>

{

public void reduce(Text key ,Iterator<Text> value , OutputCollector<Text,Text> output,Reporter r) throws IOException

{

HashMap<Integer,Float> a=new HashMap<Integer,Float>();

HashMap<Integer,Float> b=new HashMap<Integer,Float>();

String[] v;

while(value.hasNext())

{

v=value.next().toString().split(",");

if(v[0].equals("A"))

{

a.put(Integer.parseInt(v[1]),Float.parseFloat(v[2]));

}

else

{

b.put(Integer.parseInt(v[1]),Float.parseFloat(v[2]));

}

}

float aij,bij, result=0.0f;

for(int i=0;i<5;i++)

{

aij=a.containsKey(i) ? a.get(i): 0.0f;

bij=b.containsKey(i) ? b.get(i): 0.0f;

result+=aij\*bij;

}

if(result!=0.0f)

{

output.collect(null,new Text(key+","+Float.toString(result)));

}

}

}

**//input.txt**

A,0,0,1.0

A,0,1,1.0

A,0,2,1.0

A,0,3,1.0

A,0,4,1.0

A,1,0,2.0

A,1,1,2.0

A,1,2,2.0

A,1,3,2.0

A,1,4,2.0

B,0,0,1.0

B,0,1,1.0

B,0,2,1.0

B,1,0,1.0

B,1,1,1.0

B,1,2,1.0

B,2,0,1.0

B,2,1,1.0

B,2,2,1.0

B,3,0,1.0

B,3,1,1.0

B,3,2,1.0

B,4,0,1.0

B,4,1,1.0

B,4,2,1.0

**2.8 WordCount**

**//driver.java**

package wordcount;

import java.io.\*;

import java.util.\*;

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

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

import org.apache.hadoop.fs.Path;

public class driver

{

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

{

JobConf conf=new JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

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

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

JobClient.runJob(conf);

}

}

**//mapper.java**

package wordcount;

import java.io.\*;

import java.util.\*;

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>

{

public void map(LongWritable key , Text value, OutputCollector<Text,IntWritable> output, Reporter r) throws IOException

{

String line[]=value.toString().split(" ");

for(String a:line){

output.collect(new Text(a),new IntWritable(1));

}

}

}

**//reducer.java**

package wordcount;

import java.io.\*;

import java.util.\*;

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

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

class reducer extends MapReduceBase implements Reducer<Text , IntWritable , Text , IntWritable>

{

public void reduce(Text key,Iterator<IntWritable> value, OutputCollector<Text,IntWritable> output, Reporter r) throws IOException

{

int count=0;

while(value.hasNext())

{

count+=value.next().get();

}

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

}

}

**//input.txt**

HDFS is a storage unit of Hadoop

MapReduce is a processing tool of Hadoop

**Zip file with all Hadoop codes**

<https://drive.google.com/file/d/1vjqlO-EOT4nsNlqizUMiEnba8e36RuqJ/view?usp=sharing>