PRACTICAL JOURNAL IN

BIG DATA ANALYTICS

MODERN NETWORKING

EMBEDDED SYSTEMS

SUBMITTED BY

VRUSHALI PRAKASH ADAK

PRN NO.: 2020430041

IN PARTIAL FULLFILMENT FOR THE DEGREE OF MASTER OF

SCIENCE IN INFORMATION TECHNOLOGY PART – I
SEMESTER II

ACADEMIC YEAR

2023-2024

B.N. BANDODKAR COLLEGE OF SCIENCE (AUTONOMOUS)
(AFFILIATED TO UNIVERSITY OF MUMBAI)
THANE (W) - 400601, MAHARASHTRA

YEAR: 2023-2024

Vidya Prasarak Mandal's

B. N. BANDODKAR COLLEGE OF SCIENCE (AUTONOMOUS), THANE.

(Affiliated to University of Mumbai)

NAAC REACCREDITED 'A' GRADE

Best College Award, University of Mumbai

माहिती व तंत्रज्ञान विभाग

दूरध्वनी क्र. २५३३ ६५०७

Department of Information Technology

Tel. No. 2533 6507

Email : itbnb@vpmthane.org

CERTIFICATE

This is to certify that

Shri / Kum. _____

of M. Sc. (Information Technology) Part I Semester - II has completed the required number of experiments (Total =) signed herein, in this laboratory during the year 2023 – 2024.

Seal

Incharge
Department of Information Technology

Principal
B. N. Bandodkar College of Science, Thane

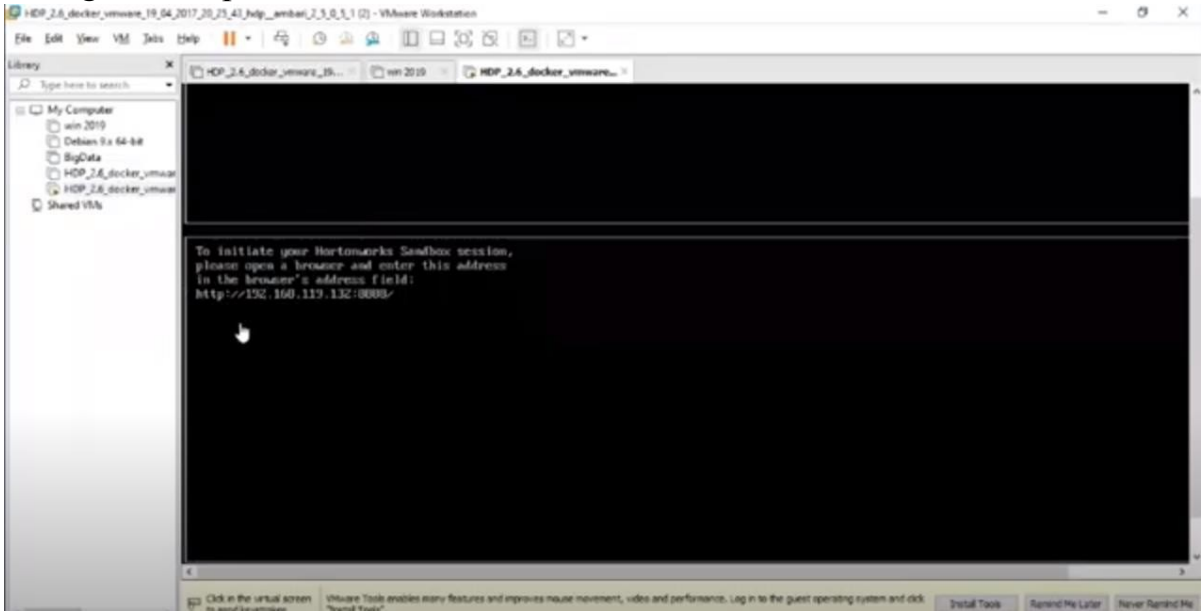
_____
External Examiner

INDEX

Practical 1

Install, configure and run Hadoop and HDFS ad explore HDFS.

Download Virtual machine setup ie VMware setup (in which Hadoop is configured.Step 1: Load the server on VM ware workstation



Step 2: To enable admin login open shell and reset root

login. Open Terminal 192.168.119.132:4200

In Sandbox login enter root

And Password is Hadoop

And reset the password

```
root@sandbox.hortonworks.com's password:
You are required to change your password immedia
tely (root enforced)
Last login: Wed Jun 30 14:50:19 2021 from 172.17
.0.2
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@sandbox ~]#
```

Windows linux system and Hadoop system are different

When we type **ls** command it is executed in local system

```
(current) UNIX password:
New password:
Retype new password:
[root@sandbox ~]# ls
anaconda-ks.cfg  install.log.syslog
blueprint.json   sandbox.info
build.out        start_ambari.sh
hdp              start_hbase.sh
install.log
[root@sandbox ~]#
```

When we type **hdfs dfs -ls** it will execute in Hadoop system directory

```
New password:
Retype new password:
[root@sandbox ~]# ls
anaconda-ks.cfg  install.log.syslog
blueprint.json   sandbox.info
build.out        start_ambari.sh
hdp              start_hbase.sh
install.log
[root@sandbox ~]# hdfs dfs -ls /
```

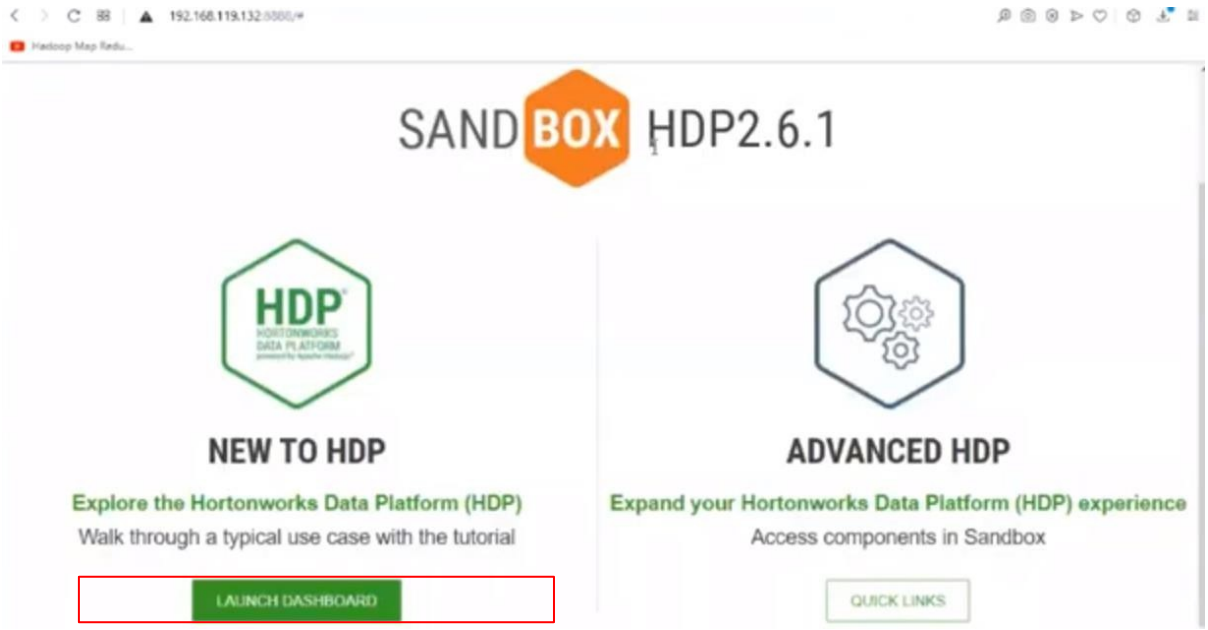
Step 3: Reset Admin account Password

```
[root@sandbox ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....
```

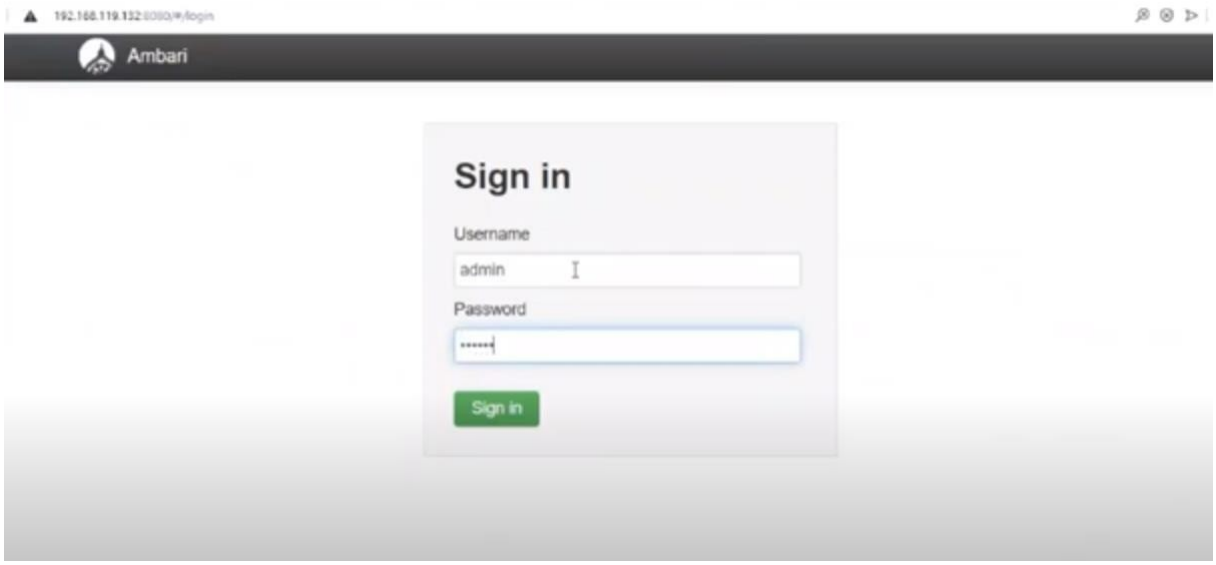
Server listening on 8080 and shell login is complete.

```
The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....................................................
Server started listening on 8080

DB configs consistency check: no errors and warnings were found.
[root@sandbox ~]#
```

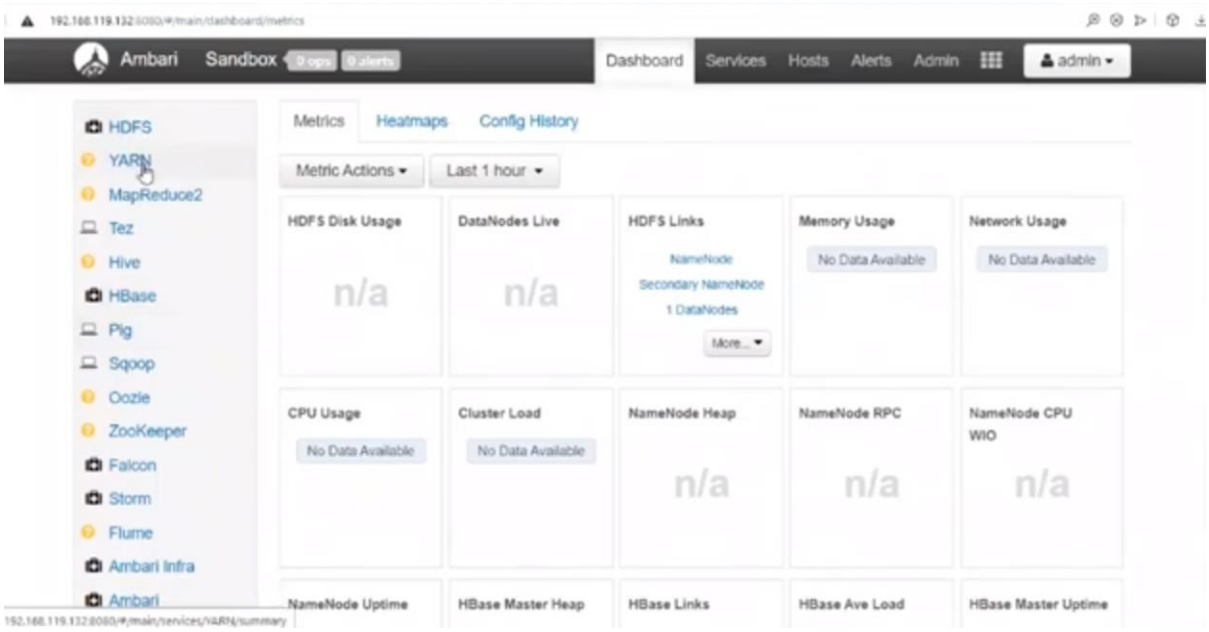
To use graphical user interface login to 192.168.119.132:4200



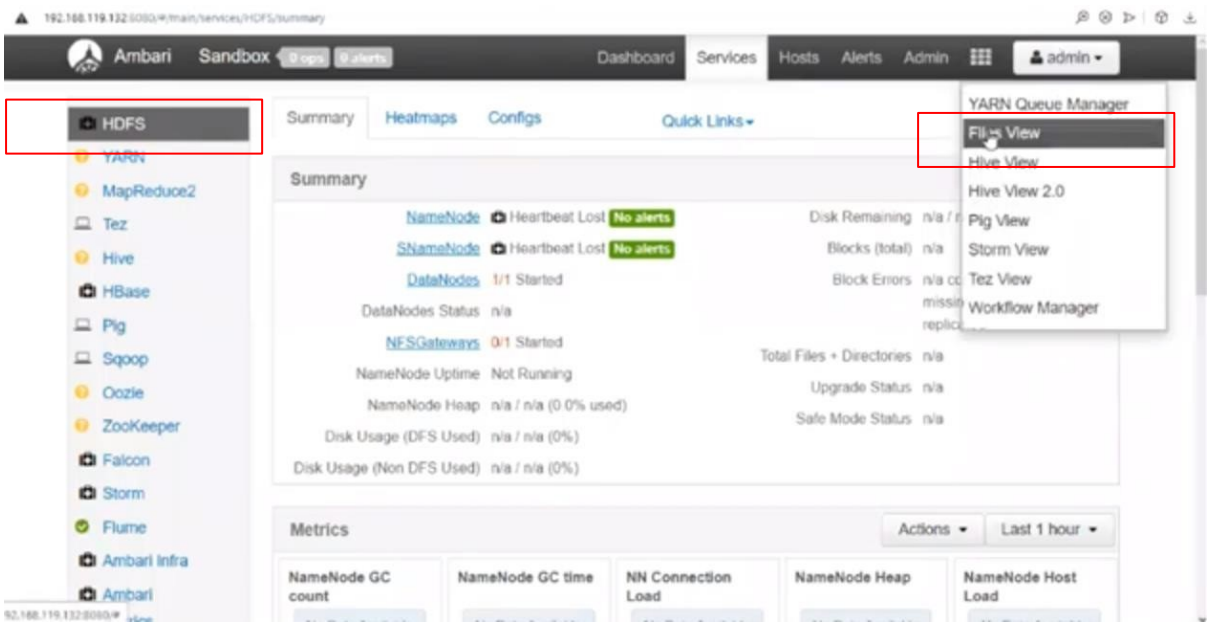
Click on Launch DashboardEnter the username and password for admin login.



Below is the Hadoop server.



To view file in HDFS click on HDFS and click on File view.

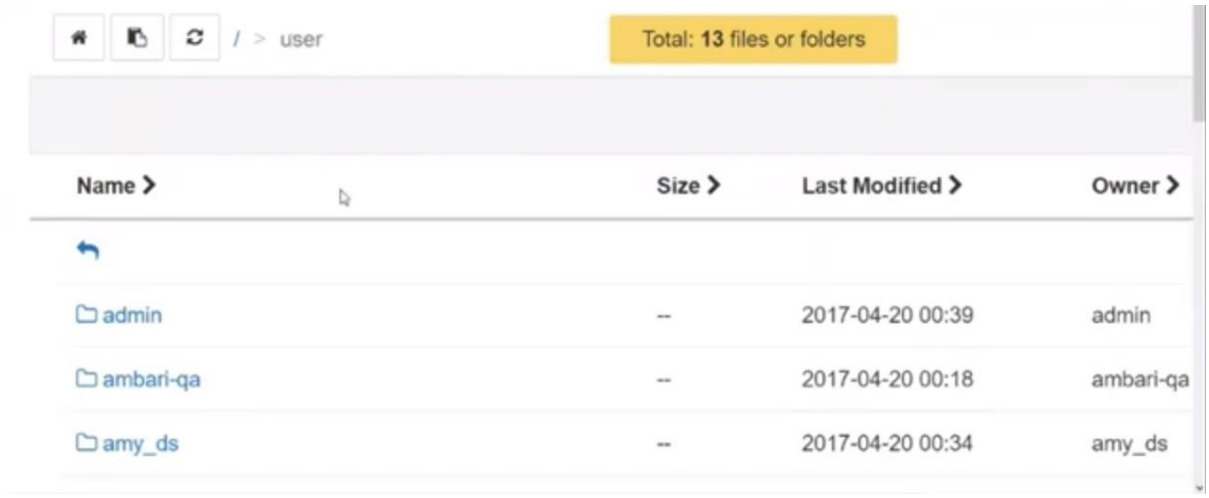


Commands:

1) To view root folder file from terminal use command hdfs dfs -ls /user and press enter.
   It will display all the files in the root user that we see in UI(Screenshot 2).
   ls: This command is used to list all the files

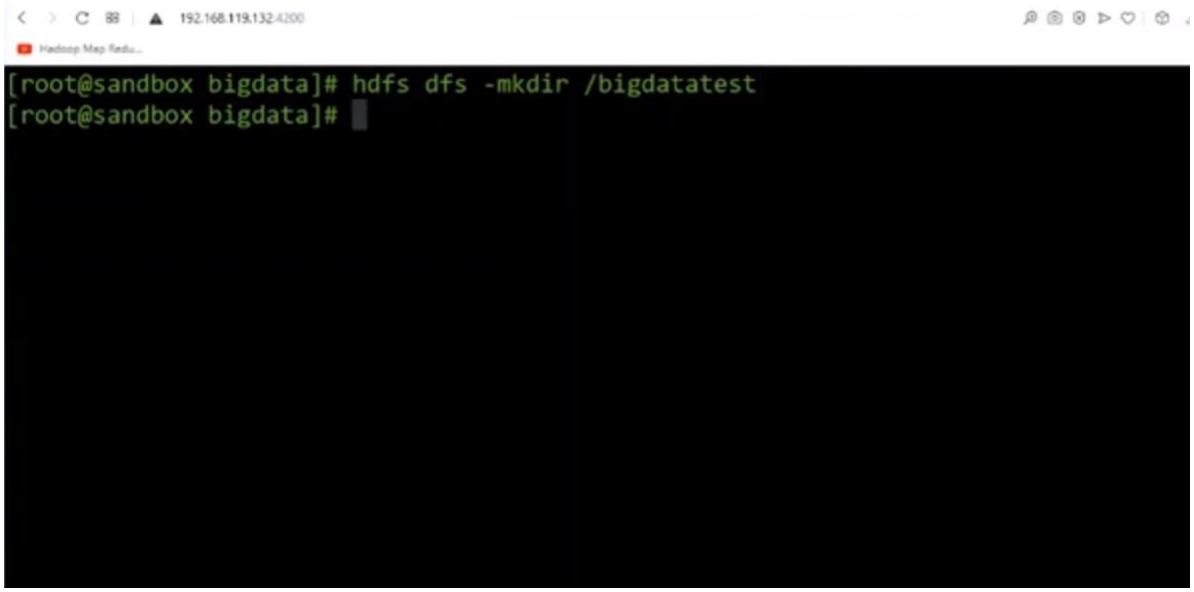
```
DB configs consistency check: no errors and warnings were found.
[root@sandbox ~]# hdfs dfs -ls /user
Found 13 items
drwxr-xr-x   - admin      hdfs          0 2017-04-19 19:09 /user/admin
drwxrwx---   - ambari-qa  hdfs          0 2017-04-19 18:48 /user/ambari-qa
drwxr-xr-x   - amy_ds     hdfs          0 2017-04-19 19:04 /user/amy_ds
drwxr-xr-x   - hbase      hdfs          0 2017-04-19 18:48 /user/hbase
drwxr-xr-x   - hcat       hdfs          0 2017-04-19 18:51 /user/hcat
drwxr-xr-x   - hive       hdfs          0 2017-04-19 19:08 /user/hive
drwxr-xr-x   - holger_gov hdfs          0 2017-04-19 19:05 /user/holger_gov
drwxrwxr-x   - livy       hdfs          0 2017-04-19 18:49 /user/livy
drwxr-xr-x   - maria_dev  hdfs          0 2017-04-19 18:58 /user/maria_dev
drwxrwxr-x   - oozie      hdfs          0 2017-04-19 18:52 /user/oozie
drwxr-xr-x   - raj_ops    hdfs          0 2017-04-19 19:06 /user/raj_ops
drwxrwxr-x   - spark      hdfs          0 2017-04-19 18:49 /user/spark
drwxr-xr-x   - zeppelin   hdfs          0 2017-04-19 18:49 /user/zeppelin
[root@sandbox ~]#
```

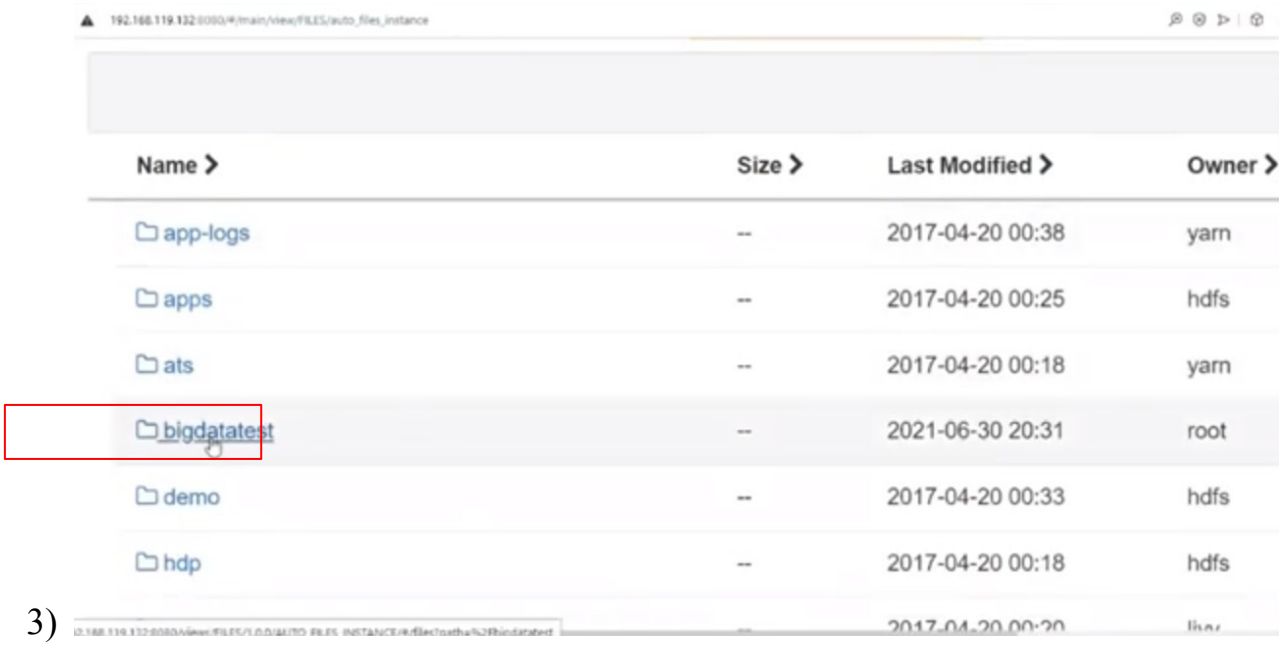
2)



mkdir: To create a directory.

Create a folder in Hadoop directory. Type command hdfs dfs -mkdir /bigdatatest and enter. After it execute the command, we will see whether it is created folder in UI.



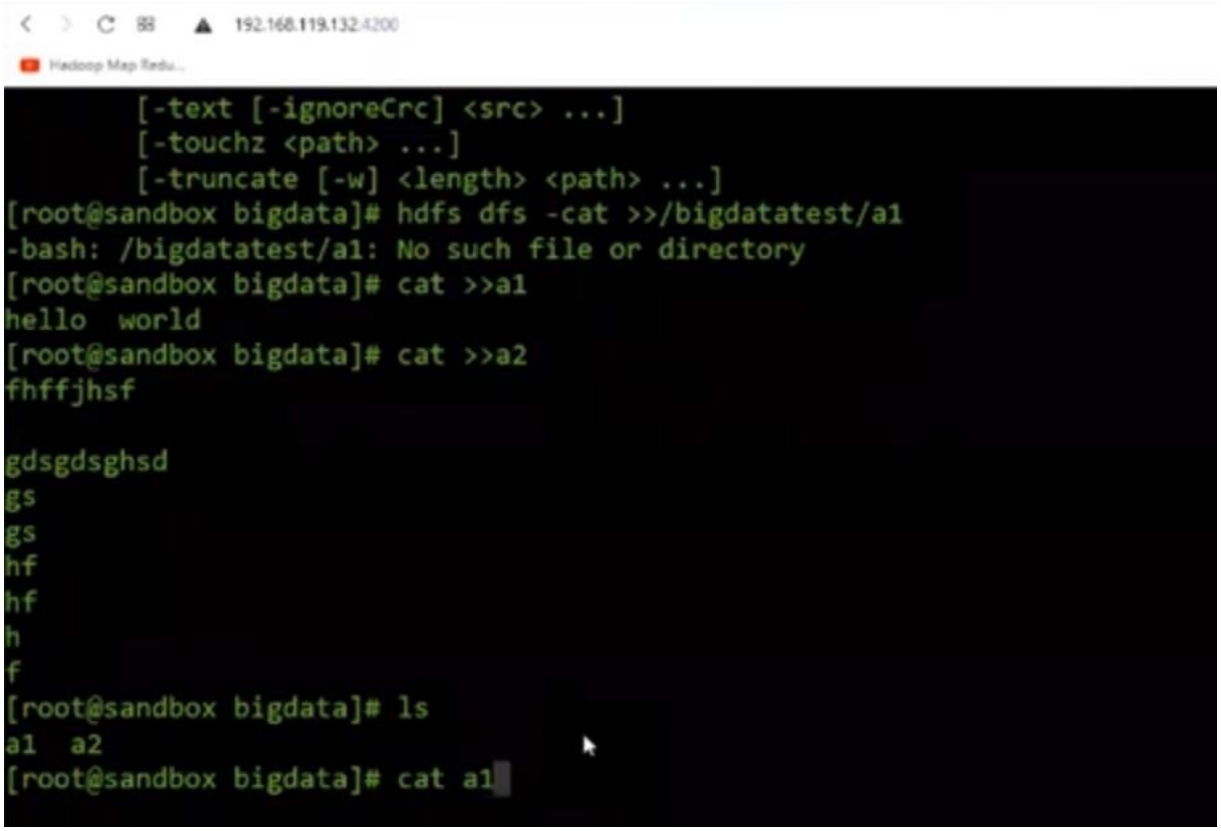


3) Create a file in local directory

Cat: Create a file.

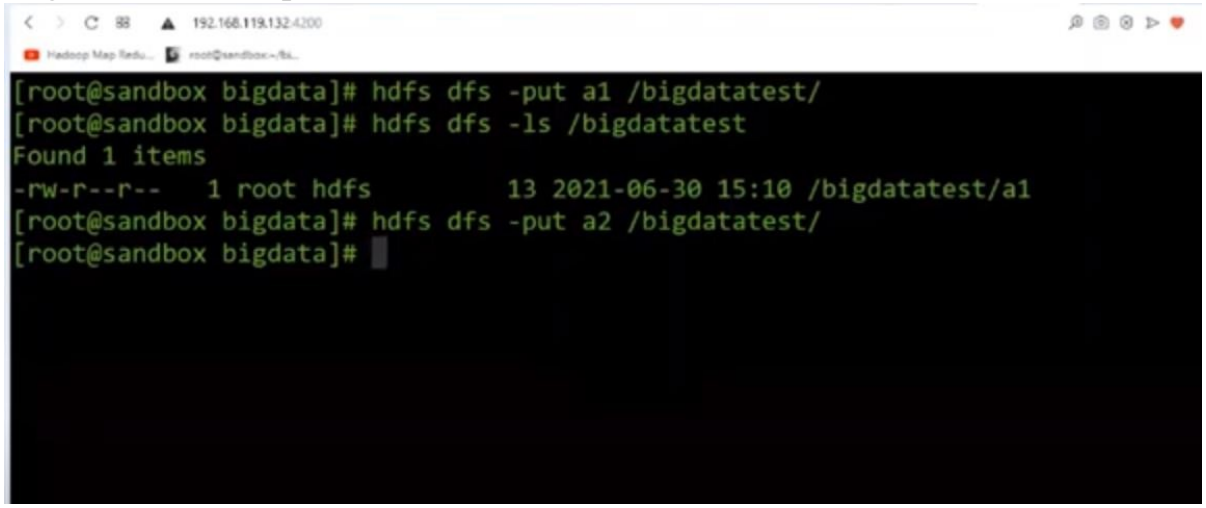
Cat>>

To terminate press ctrl+d

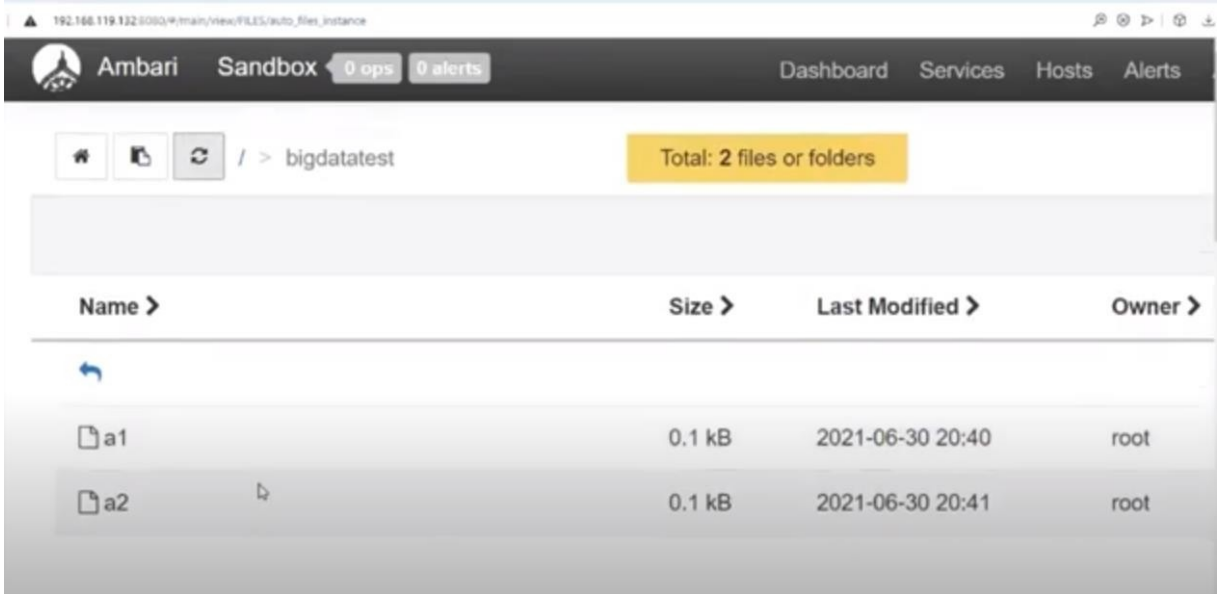


4) To upload files/directory to from local to HDFS

Put: to move a local file or directories into the distributed file systemCommand: hdfs dfs -put a1 /bigdatatest/ and hdfs dfs -put a2 /bigdatatest/ will upload both the files.

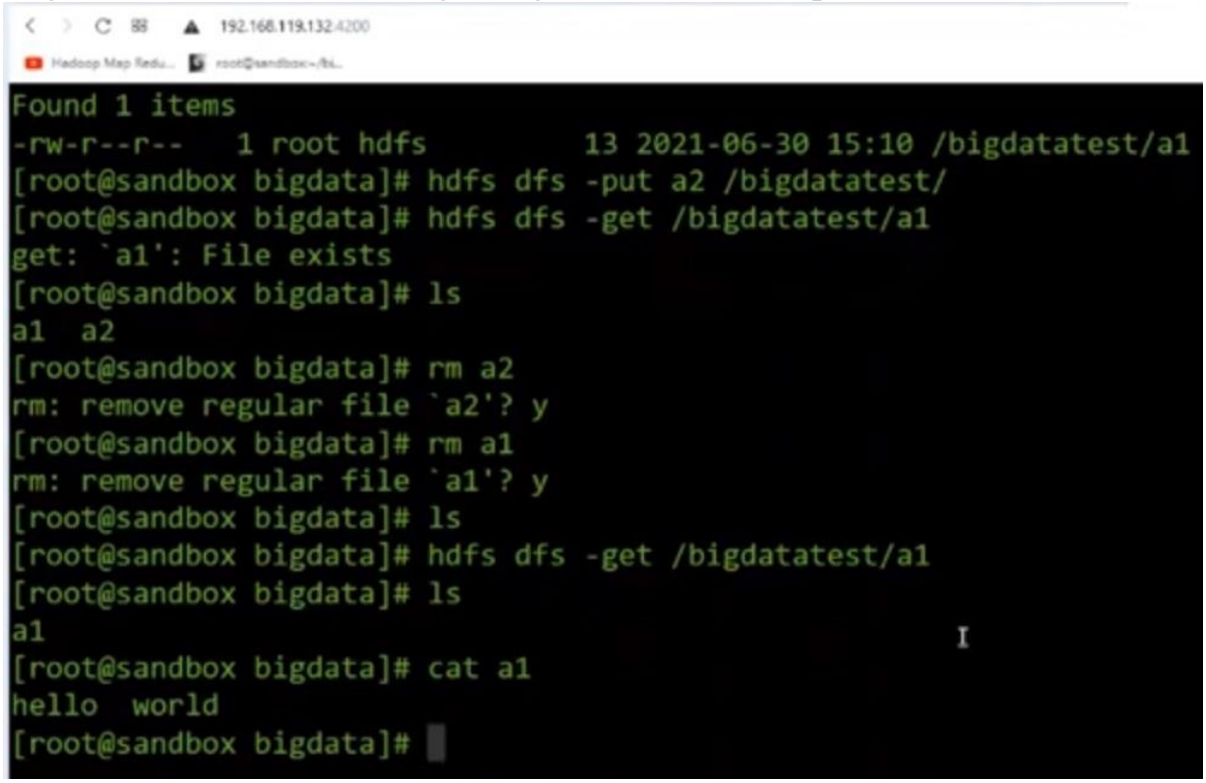


Refresh the user interface we can see both the files.

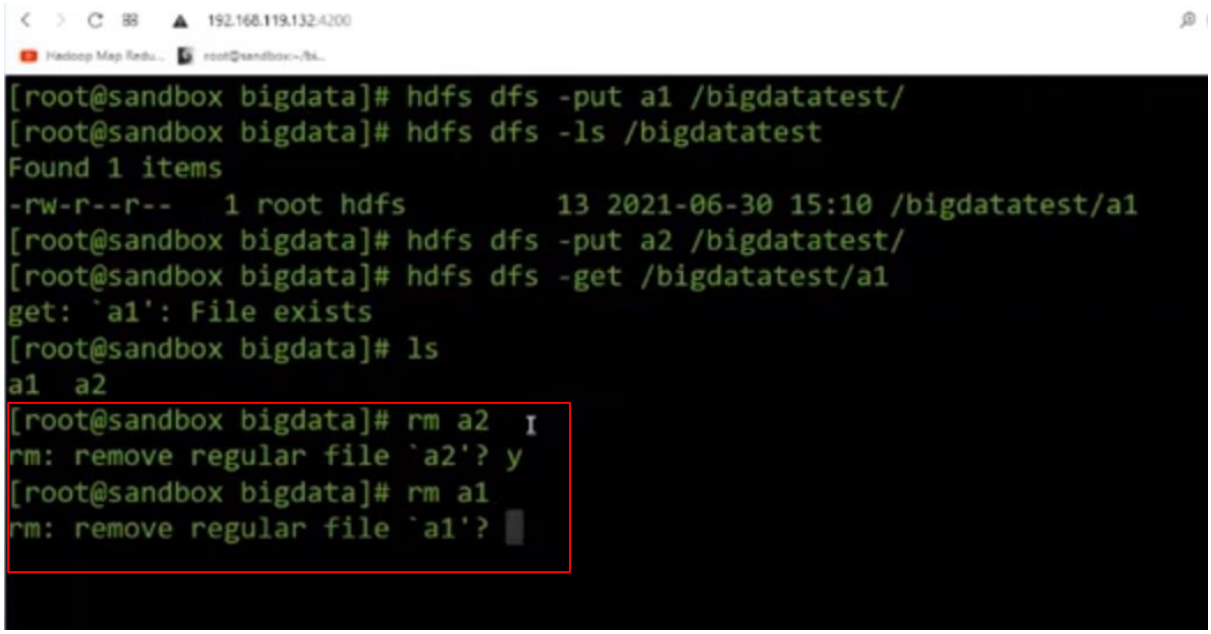


To download files/directories to from hdfs to local**Get:** To copy files/folders from hdfs store to local file system.Command: hdfs dfs -get

/bigdatatest/a1 and hdfs dfs -get /bigdatatest/ a2 will upload both the files.

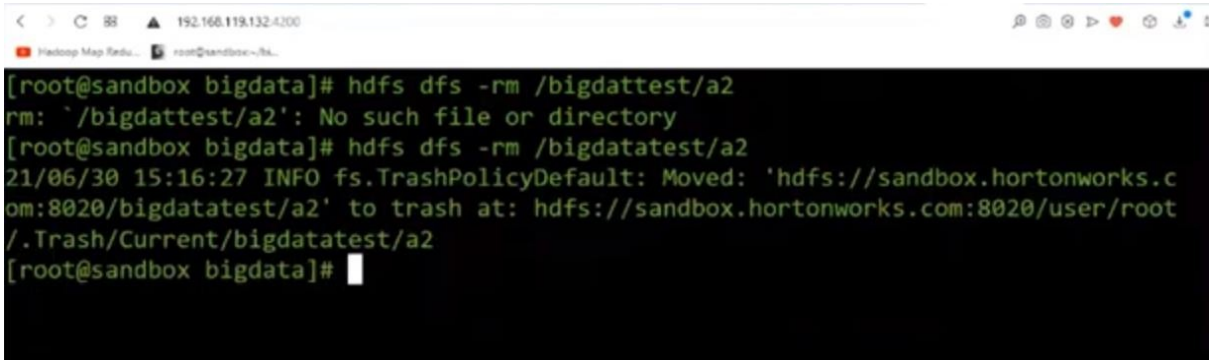


5) To remove file from local use rm command

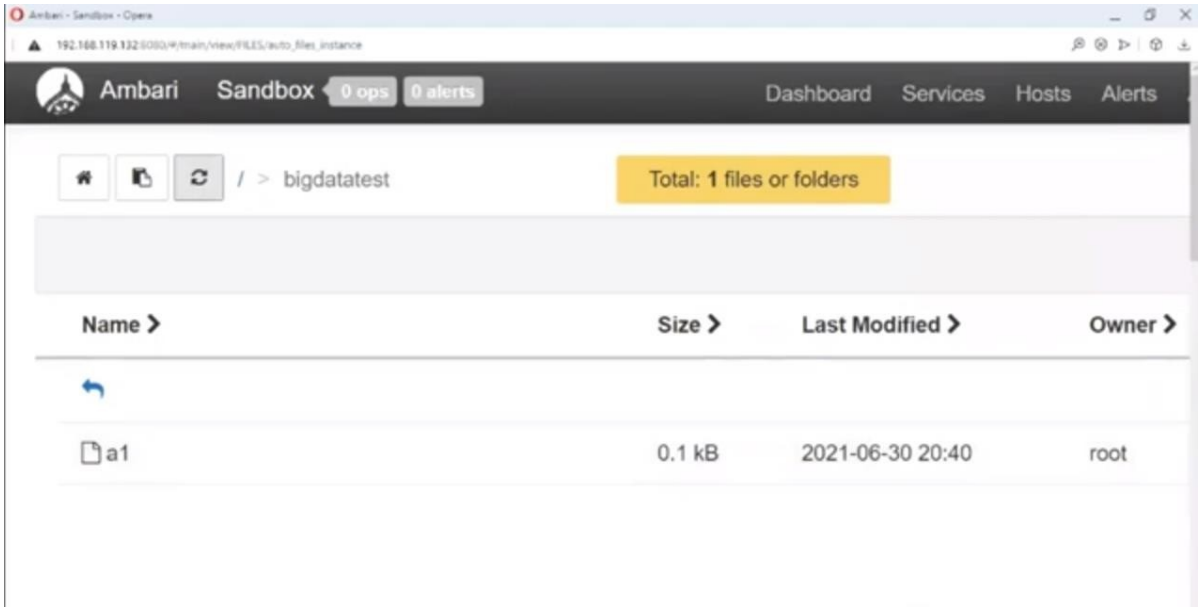


6) To remove file from Hadoop directory

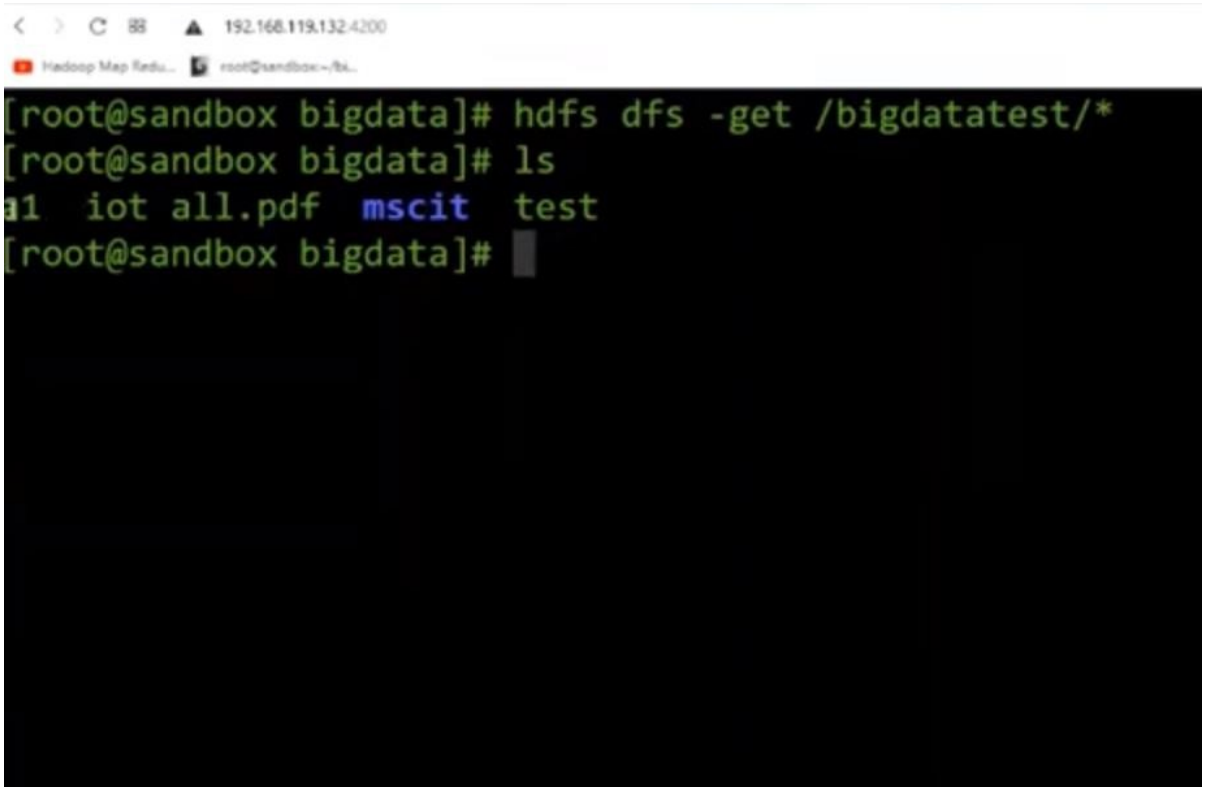
Command: hdfs dfs -rm a2 /gibdatatest/a2



Now refresh the UI and the file will be deleted.



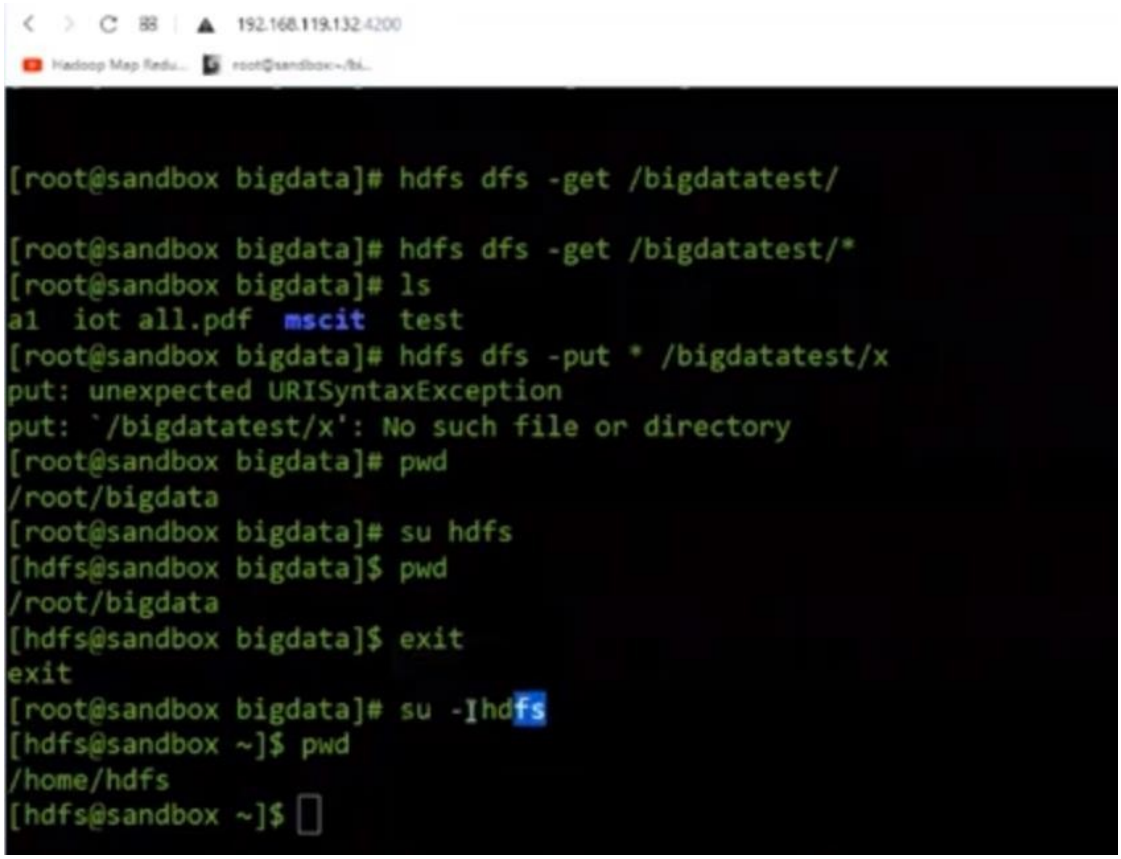
7) To download all the files from hdfs to local Command: hdfs dfs -get /bigdatatest/*



8) Change user and directory and change user

only Command: su – hdfs (Change user and directory)

Su hdfs (change user only)


192.168.119.132:4200
[root@sandbox bigdata]# hdfs dfs -get /bigdatatest/
[root@sandbox bigdata]# hdfs dfs -get /bigdatatest/*
[root@sandbox bigdata]# ls
a1 iot all.pdf mscit test
[root@sandbox bigdata]# hdfs dfs -put * /bigdatatest/x
put: unexpected URISyntaxException
put: `/bigdatatest/x': No such file or directory
[root@sandbox bigdata]# pwd
/root/bigdata
[root@sandbox bigdata]# su hdfs
[hdfs@sandbox bigdata]$ pwd
/root/bigdata
[hdfs@sandbox bigdata]$ exit
exit
[root@sandbox bigdata]# su -hdfs
[hdfs@sandbox ~]$ pwd
/home/hdfs
[hdfs@sandbox ~]$

Practical 2

Implement word count / frequency programs using MapReduce

Map Reduce as two component Map and Reduce.

Java program:

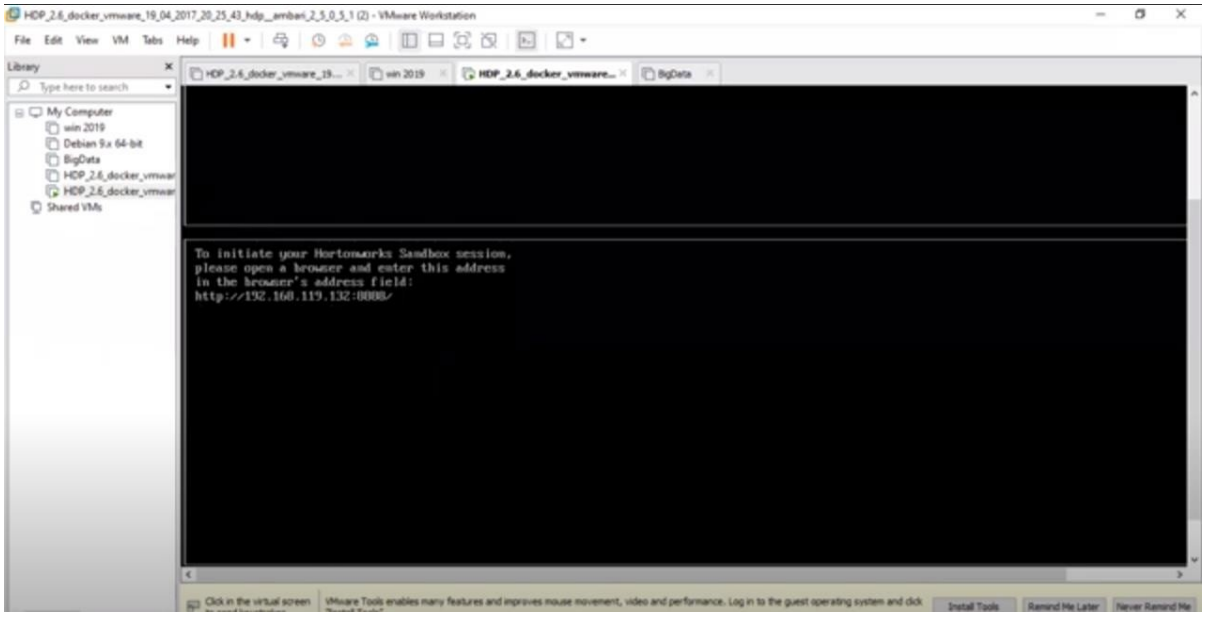
write program save as WordCount.java//////////////////////
```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount { public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{
 private final static IntWritable one = new IntWritable(1);
 private Text word = new Text();
 public void map(Object key, Text value, Context context
 ) throws IOException, InterruptedException
 { StringTokenizer itr = new
 StringTokenizer(value.toString()); while
 (itr.hasMoreTokens()) {//"This is the output is the"
 word.set(itr.nextToken());
 context.write(word, one);
 }
 }
 }
 public static class IntSumReducer extends
Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
 public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException,
InterruptedException
 {//is,3
 int sum = 0;
 for (IntWritable val : values)
 { sum += val.get();
```

```
 }
 result.set(sum);
 context.write(key, result);
 }
 }public static void main(String[] args) throws Exception
 { Configuration conf = new Configuration();
 Job job = Job.getInstance(conf, "word count");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job, new Path(args[0]));
 FileOutputFormat.setOutputPath(job, new Path(args[1]));
 System.exit(job.waitForCompletion(true)?0:1);
 }
}
```

//////////////////////**Text File:**

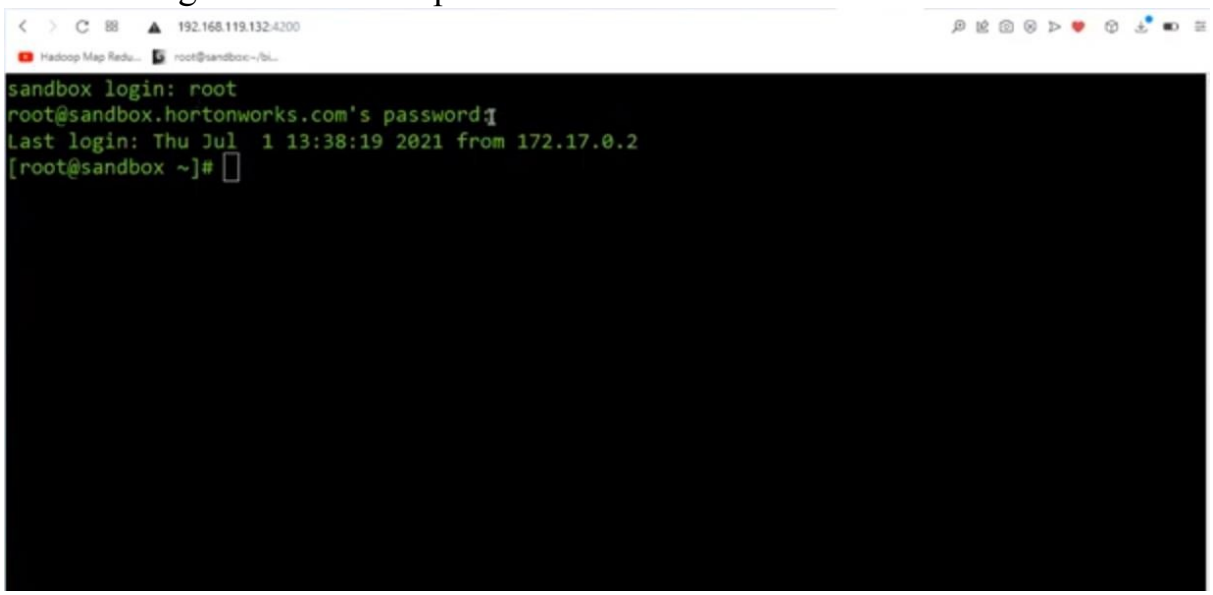
Hello World

This is the output is theStart the server (Horton Sandbox)



Open the terminal with 192.168.119.132/4200

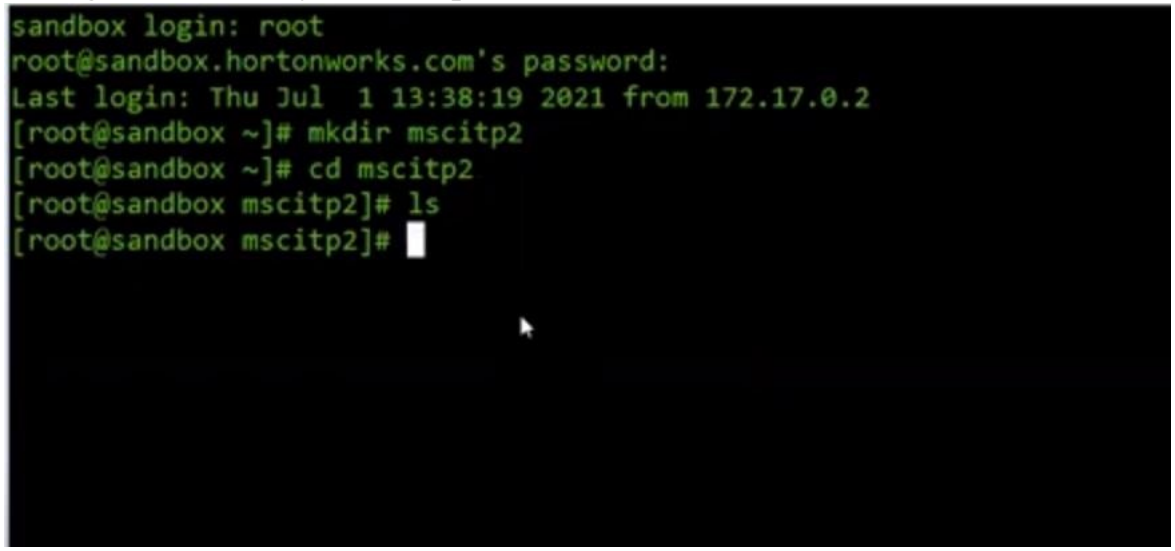
Enter the login: root and the password and enter



Create a folder in local directory.

Command: mkdir mscitp2

Change the directory cd mscitp2



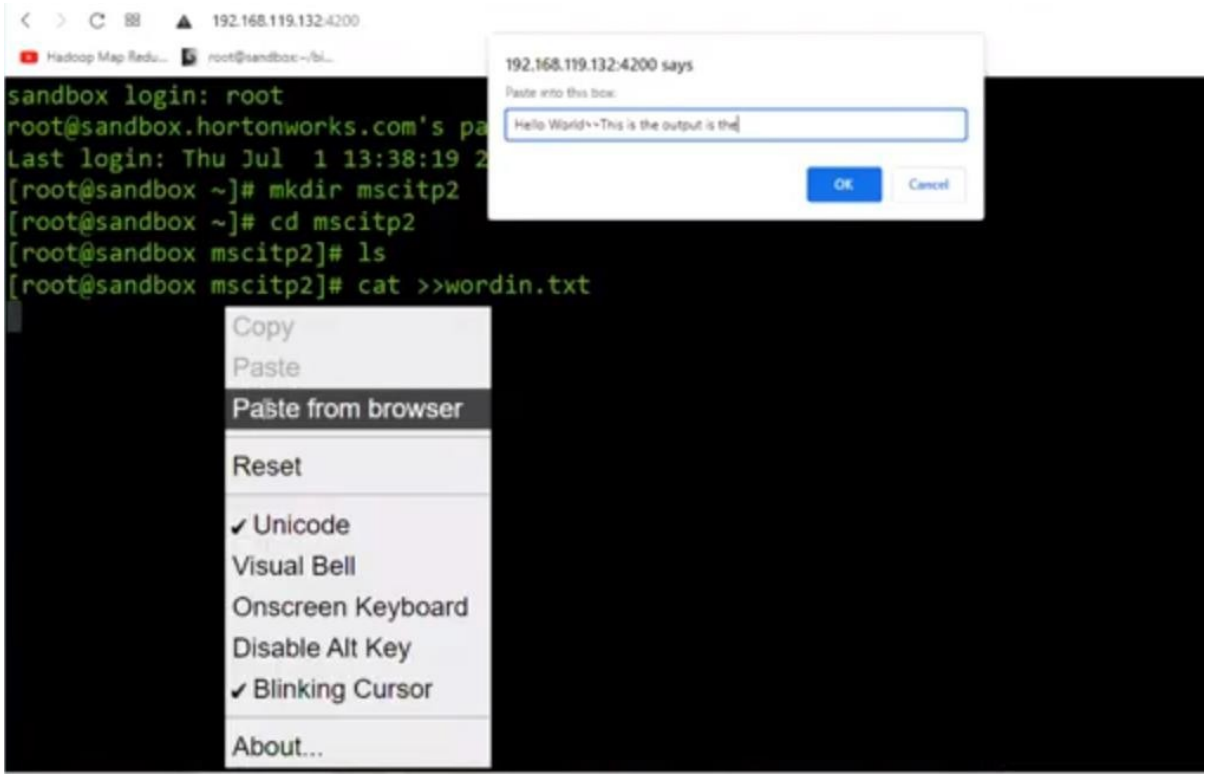
Now create input file

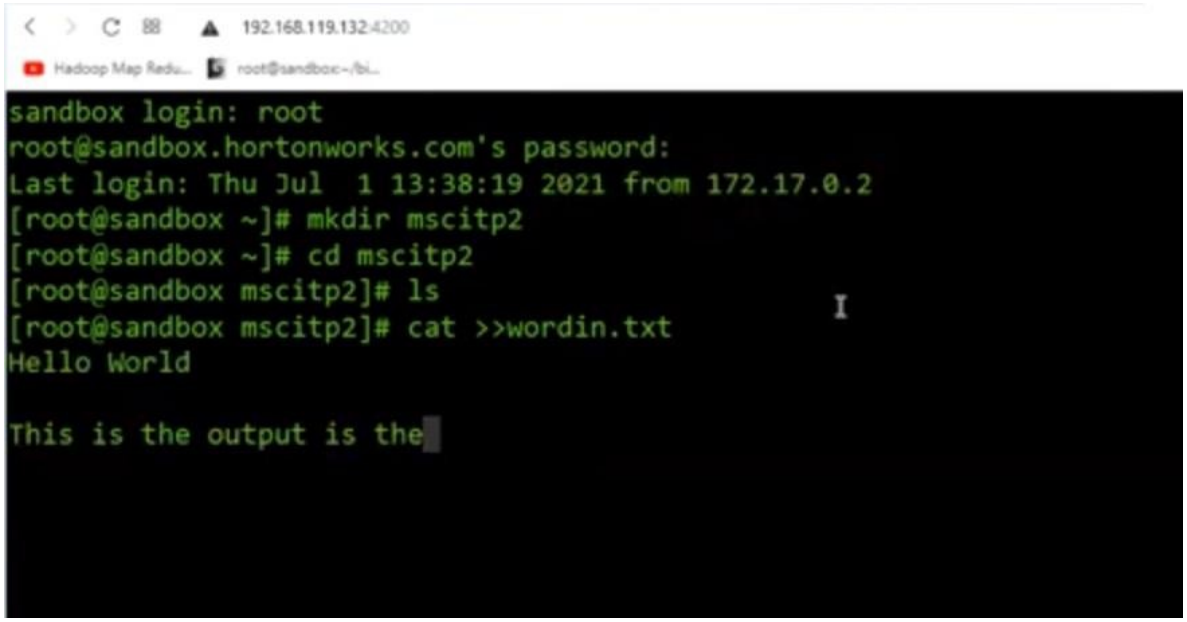
Command: cat >> wordin.txt

Paste the text by right clicking on terminal

Hello World

This is the output is the



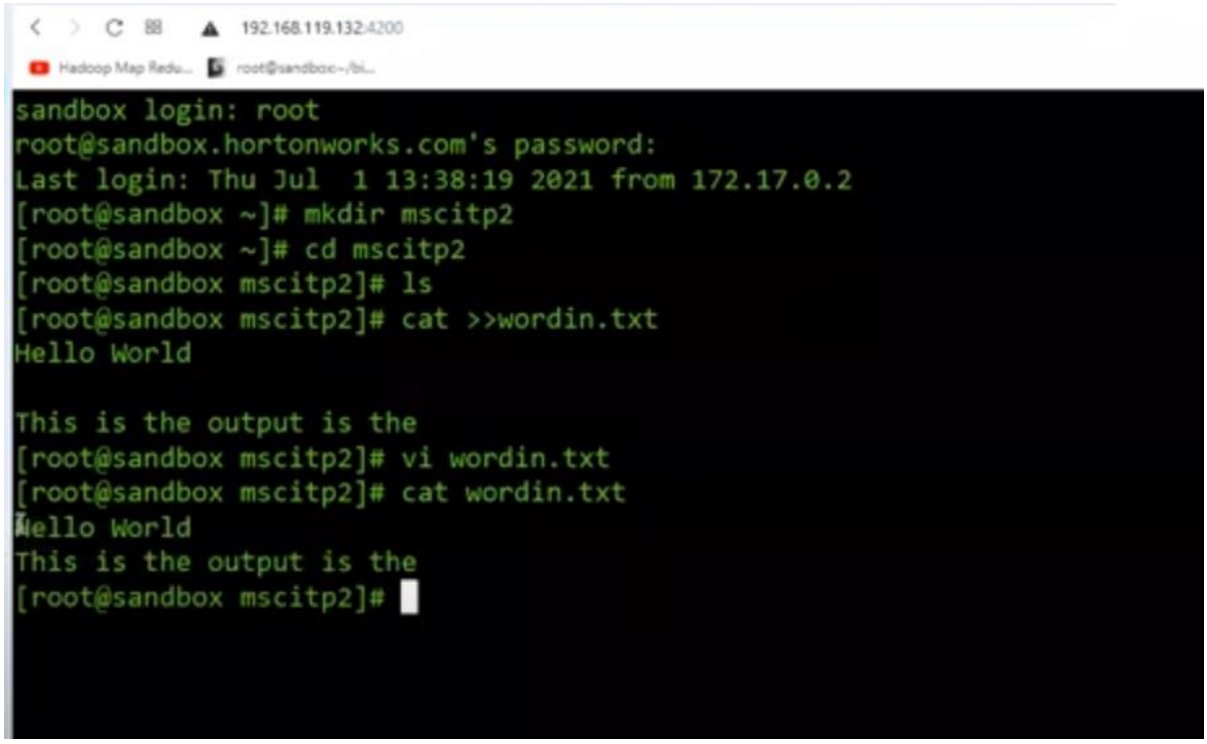


To remove the extra space type command

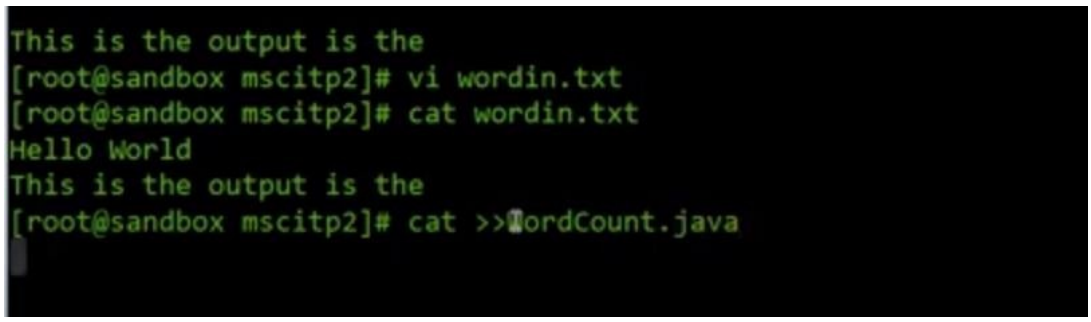
vi wordin.txt

After removing the extra space check the content of the file

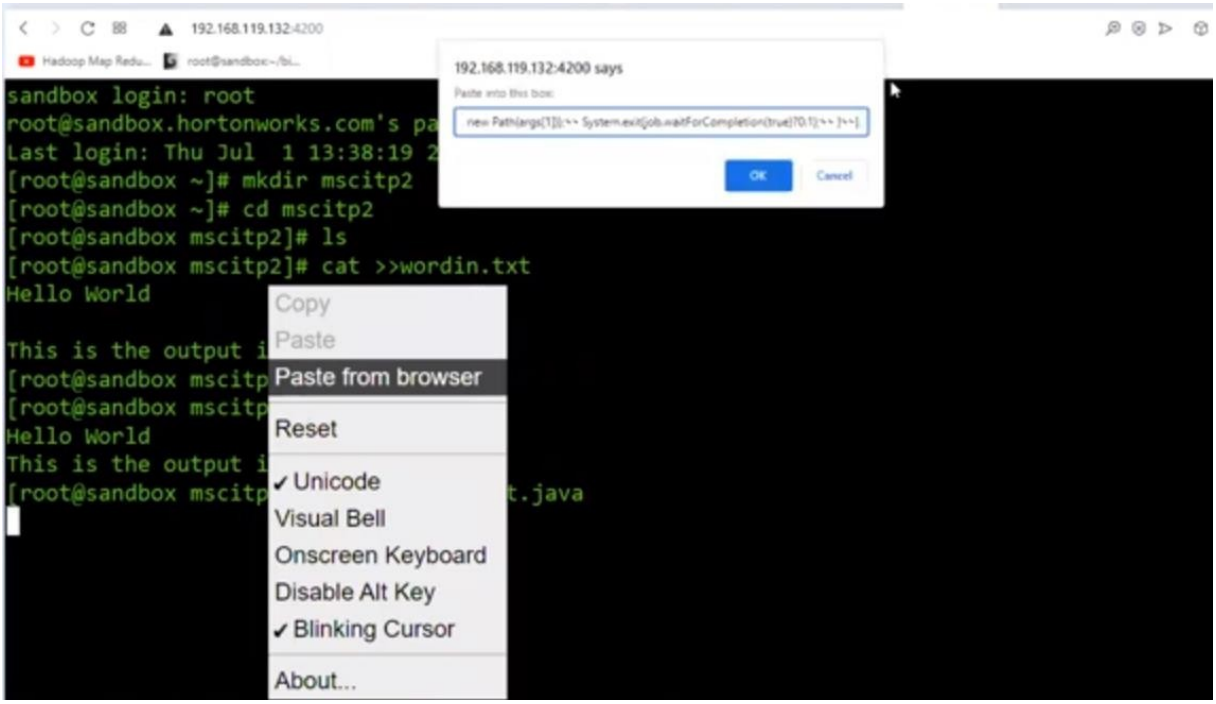
cat wordin.txt



Create another file wordcount.java

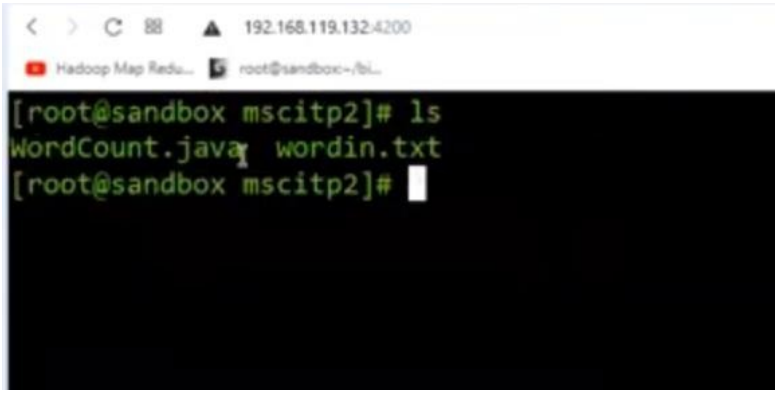


Paste the java code.



Press control d to save the file

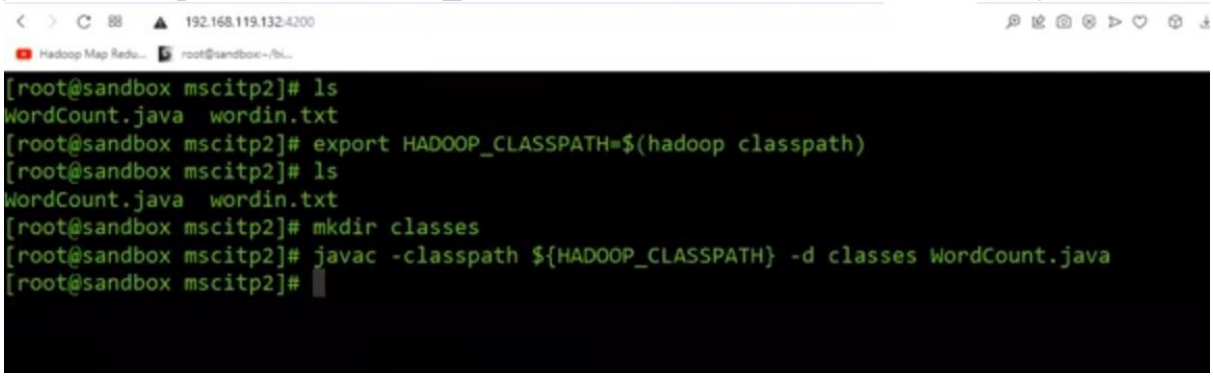
Check both the files create with command ls



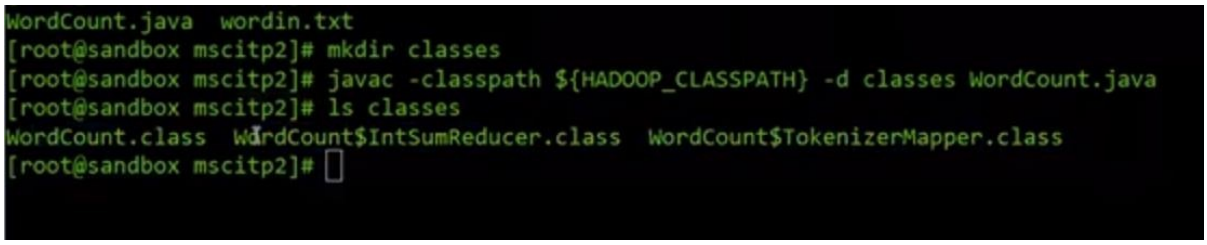
Now, to compile the java file

export HADOOP_CLASSPATH=$(hadoop classpath)
mkdir classes (To keep the compile files)

javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java

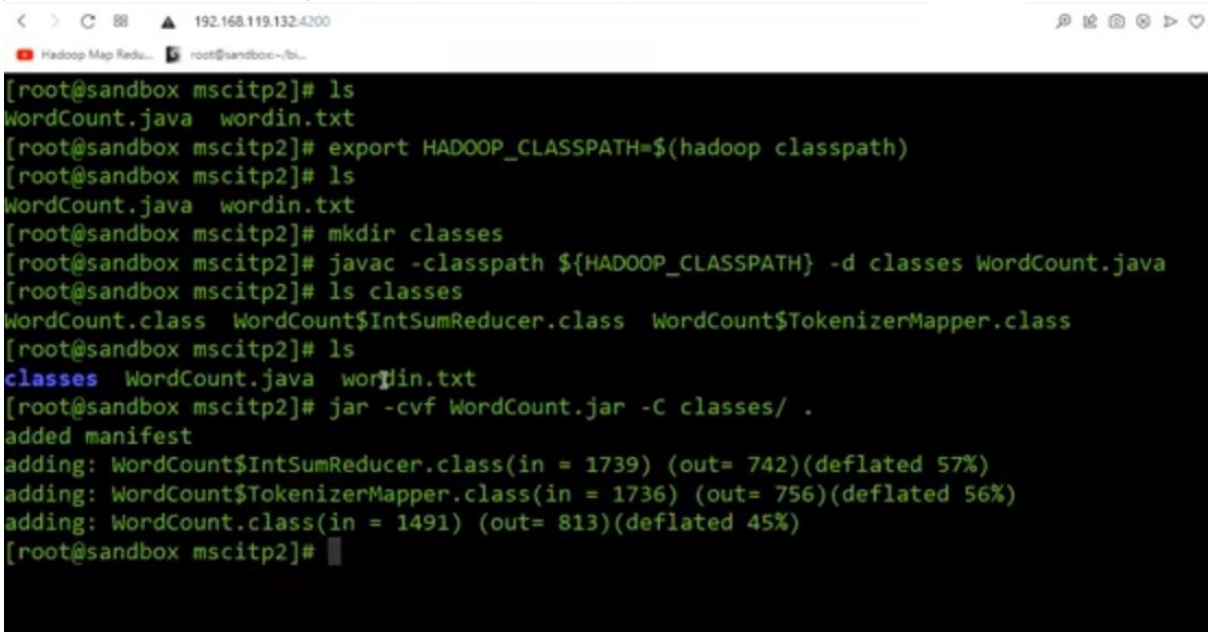


Check class files are created with command ls classes



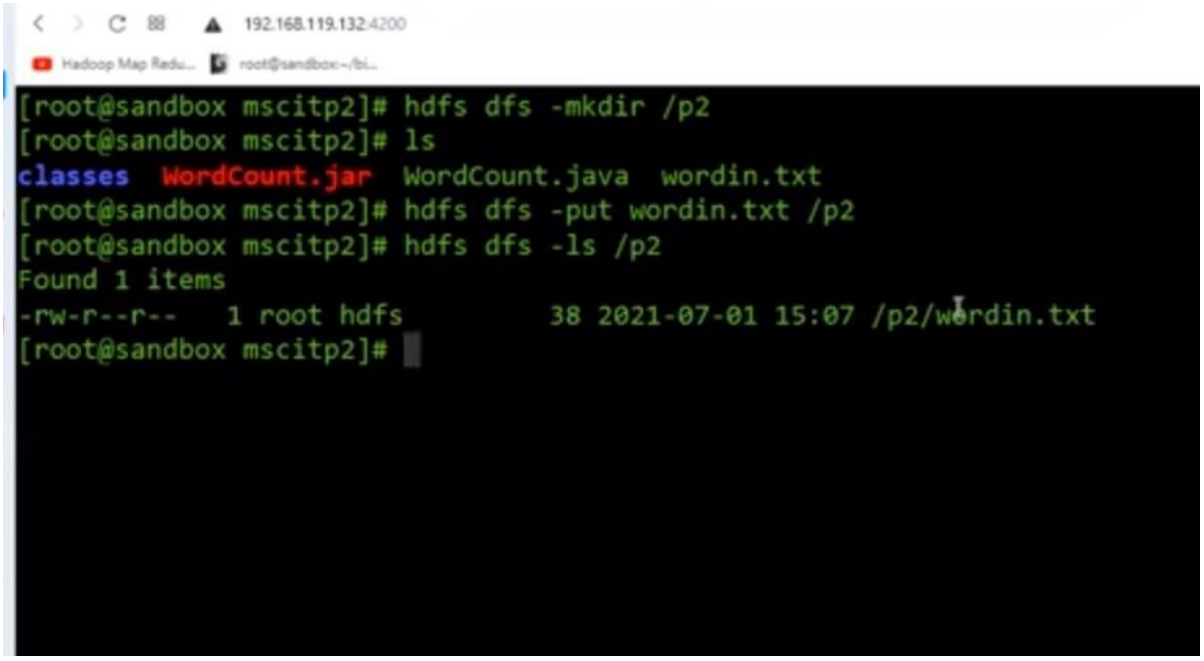
Now we have to bind all the class into single jar file with below command

jar -cvf WordCount.jar -C classes/ .



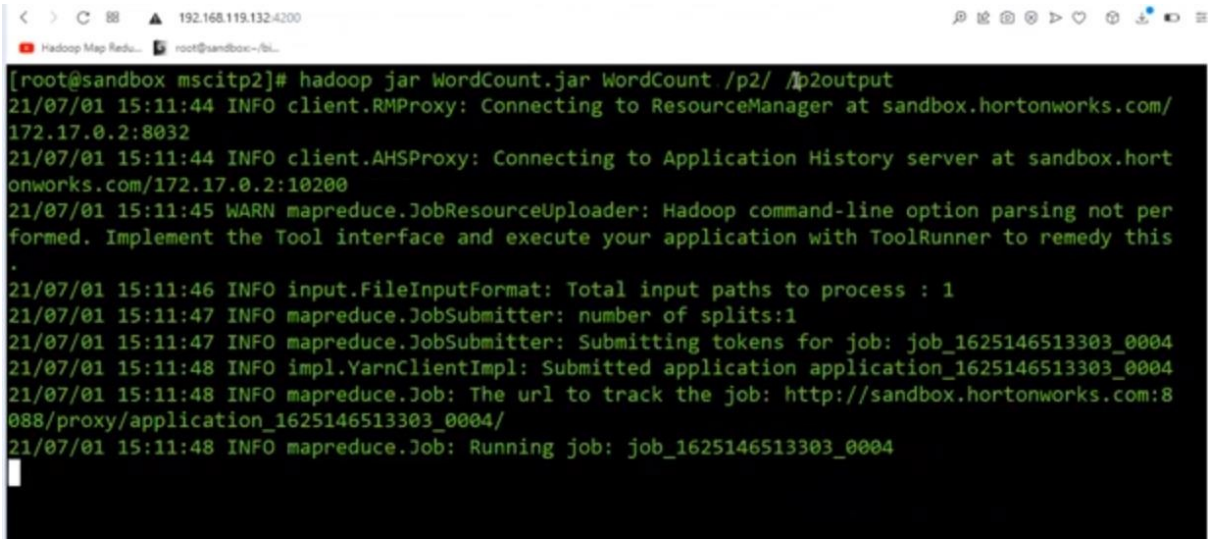
Run ls command we can see jar file is created.

wordin.txt should be present in word directory of hdfs. So we need to upload wordin.txt file.



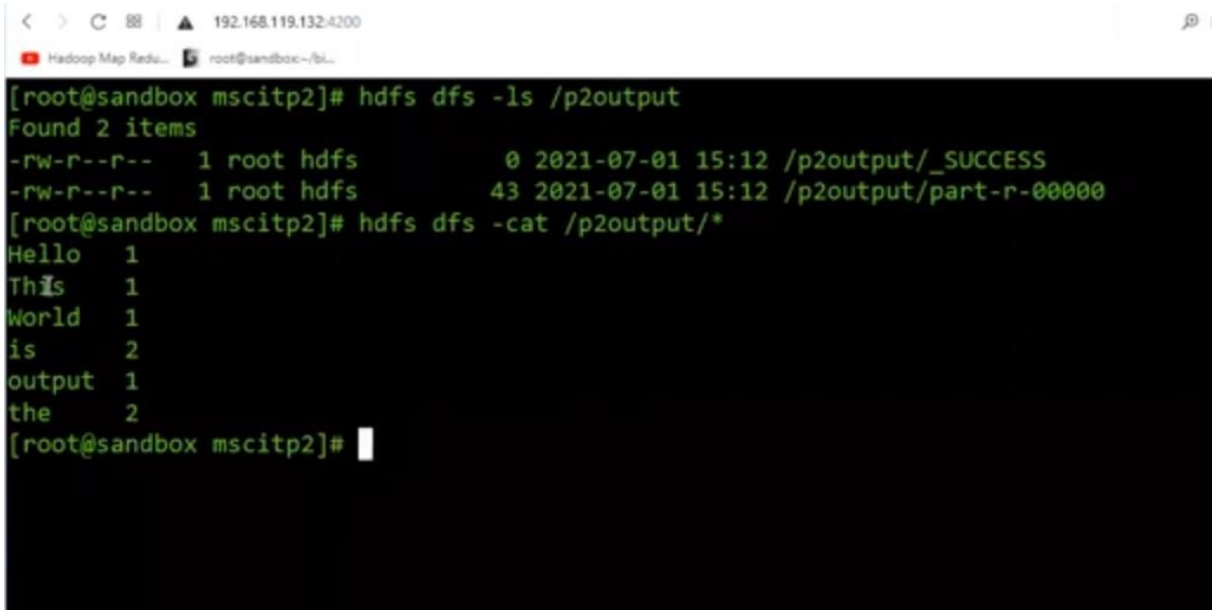
We need to put the final output p2output.

hadoop jar WordCount.jar WordCount /p2/ /p2output



Print the content of the output file

Command: hdfs dfs -cat /p2ouput/*



Ctrl + l to clear the screen.

vi filename.txt= this command will create/ open filename.txt

two modes of vi editor

1) Insert mode – i (press i key)
2) Command mode – esc key

:wq is to save and exit

Practical 3

Implement an MapReduce program that processes a weather dataset.

Java program:

MyMaxMin.java

```
//////////////////////////
// importing Libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import
org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import
org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;public class MyMaxMin {
        // Mapper

        /*MaxTemperatureMapper class is static
        * and extends Mapper abstract class
        * having four Hadoop generics type
        * LongWritable, Text, Text, Text.
        */
```

```
public static class MaxTemperatureMapper extends
                Mapper<LongWritable, Text, Text, Text> {
public static final int MISSING = 9999;

@Override
        public void map(LongWritable arg0, Text Value, Context context)
                throws IOException, InterruptedException {

        String line = Value.toString();

                // Check for the empty line
                if (!(line.length() == 0)) {

                        // from character 6 to 14 we have
                        // the date in our dataset
                        String date = line.substring(6, 14);
    // similarly we have taken the maximum
                        // temperature from 39 to 45 characters
                        float temp_Max = Float.parseFloat(line.substring(39,
45).trim());

                        // similarly we have taken the minimum
                        // temperature from 47 to 53 characters

                        float temp_Min = Float.parseFloat(line.substring(47,
53).trim());                    // if maximum temperature is
                        // greater than 30, it is a hot day
```

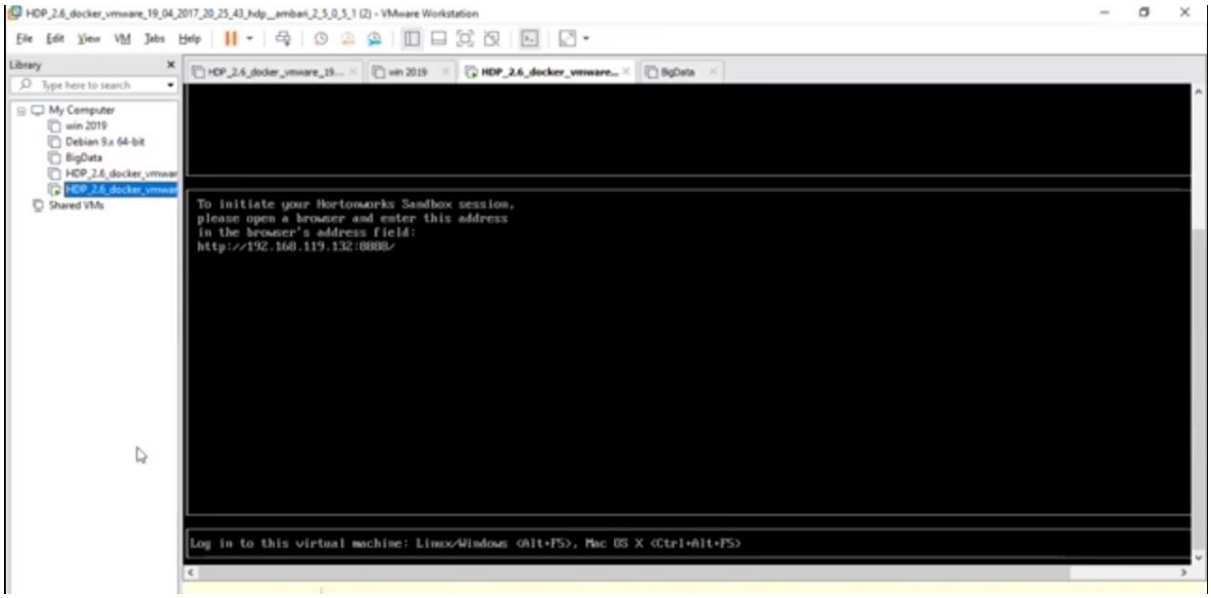
```
                        if (temp_Max > 30.0) {

                            // Hot day
                            context.write(new Text("The Day is Hot Day :"
+ date),
                                                    new
Text(String.valueOf(temp_Max)));
                        }                       // if the minimum
temperature is
                        // less than 15, it is a cold day
                        if (temp_Min < 15) {

                            // Cold day
                            context.write(new Text("The Day is Cold Day
:" + date),
                                                new
Text(String.valueOf(temp_Min)));
                        }
                    }
                }       }// Reducer

        /*MaxTemperatureReducer class is
        static and extends Reducer abstract class
        having four Hadoop generics type
        Text, Text, Text, Text.
        */
        //The Day is Cold Day :20150101 ,-21.8

        public static class MaxTemperatureReducer extends
```

```
        Reducer<Text, Text, Text, Text> {            /**
     * @method reduce
     * This method takes the input as key and
     * list of values pair from the mapper,
     * it does aggregation based on keys and
     * produces the final context.
     */

     public void reduce(Text Key, Iterator<Text> Values, Context
context)
                throws IOException, InterruptedException {
          // putting all the values in
          // temperature variable of type String
          String temperature = Values.next().toString();
          context.write(Key, new Text(temperature));
     }    }    /**
  * @method main
  * This method is used for setting
  * all the configuration properties.
  * It acts as a driver for map-reduce
  * code.
  */

  public static void main(String[] args) throws Exception {        // reads
the default configuration of the
     // cluster from the configuration XML files
     Configuration conf = new Configuration();
```

```
        // Initializing the job with the
        // default configuration of the cluster
        Job job = new Job(conf, "weather example");

        // Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);          // Key type coming
out of mapper
        job.setMapOutputKeyClass(Text.class);

        // value type coming out of mapper
        job.setMapOutputValueClass(Text.class);          // Defining the
mapper class name
        job.setMapperClass(MaxTemperatureMapper.class);

        // Defining the reducer class name
        job.setReducerClass(MaxTemperatureReducer.class);          //
Defining input Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setInputFormatClass(TextInputFormat.class);

        // Defining output Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setOutputFormatClass(TextOutputFormat.class);          //
setting the second argument
        // as a path in a path variable
        Path OutputPath = new Path(args[1]);          // Configuring the
input path
```

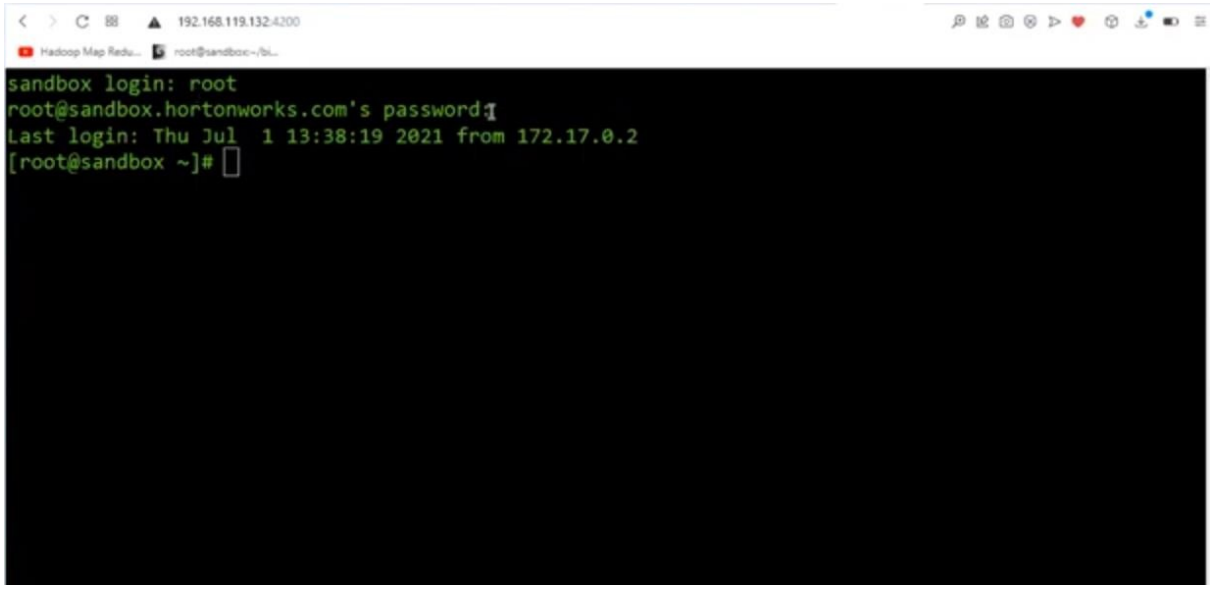
```
        // from the filesystem into the job
        FileInputFormat.addInputPath(job, new Path(args[0]));          //
Configuring the output path from
        // the filesystem into the job
        FileOutputFormat.setOutputPath(job, new Path(args[1]));        //
deleting the context path automatically
        // from hdfs so that we don't have
        // to delete it explicitly
        OutputPath.getFileSystem(conf).delete(OutputPath);
        // flag value becomes false
        System.exit(job.waitForCompletion(true) ? 0 : 1);  }
}
```

//////////////////////Start the server



Open the terminal with 192.168.119.132/4200

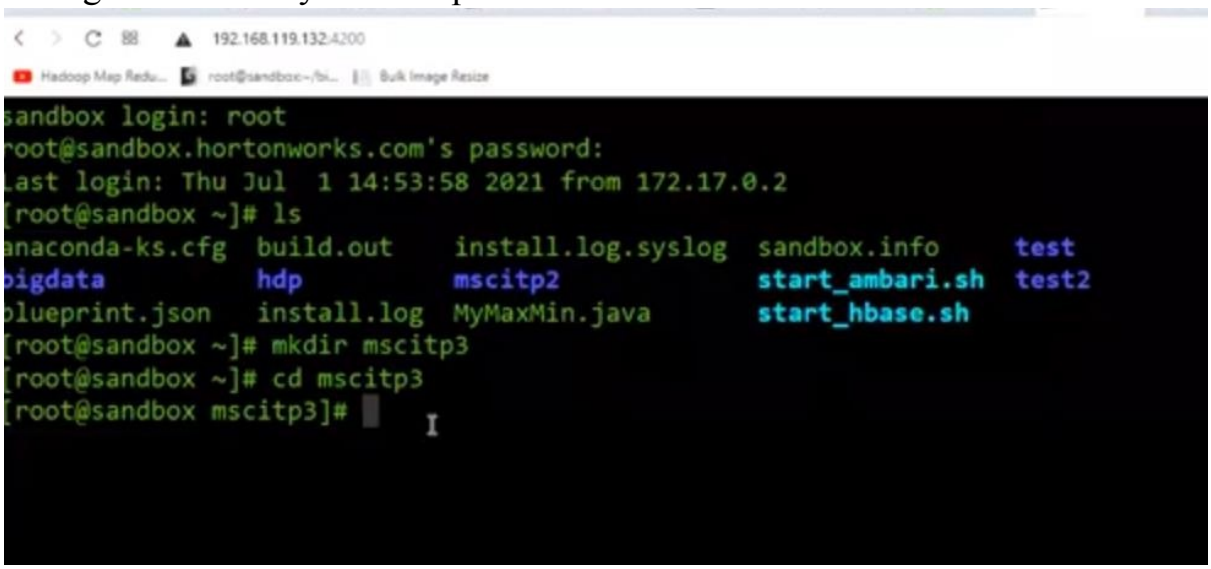
Enter the login: root and the password and enter



Create a folder in local directory.

Command: mkdir mscitp3

Change the directory cd mscitp3



Now create input file

Command: cat >> weatherin2.txt

Paste the weather dataset by right clicking on terminal

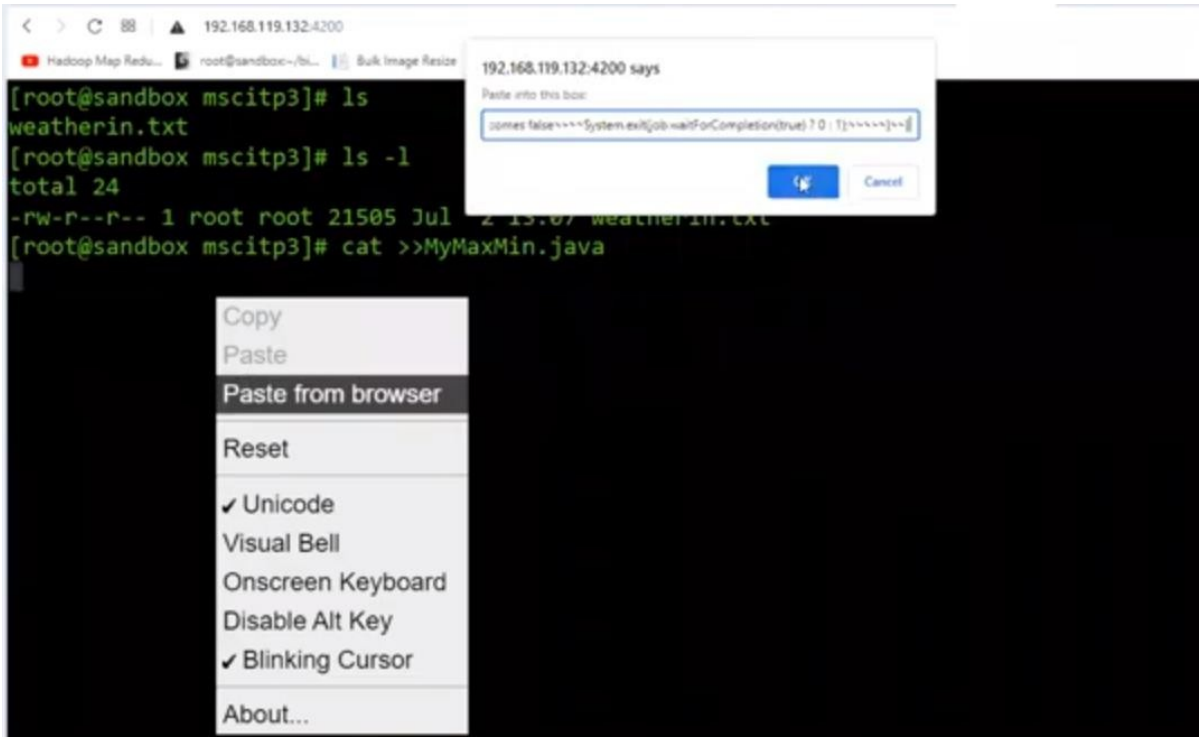
Ctrl d will save the file

Run command ls to see the file.

Create java file

Command: cat >>MyMaxMin.java

Paste the java code and ctrl d to save the file



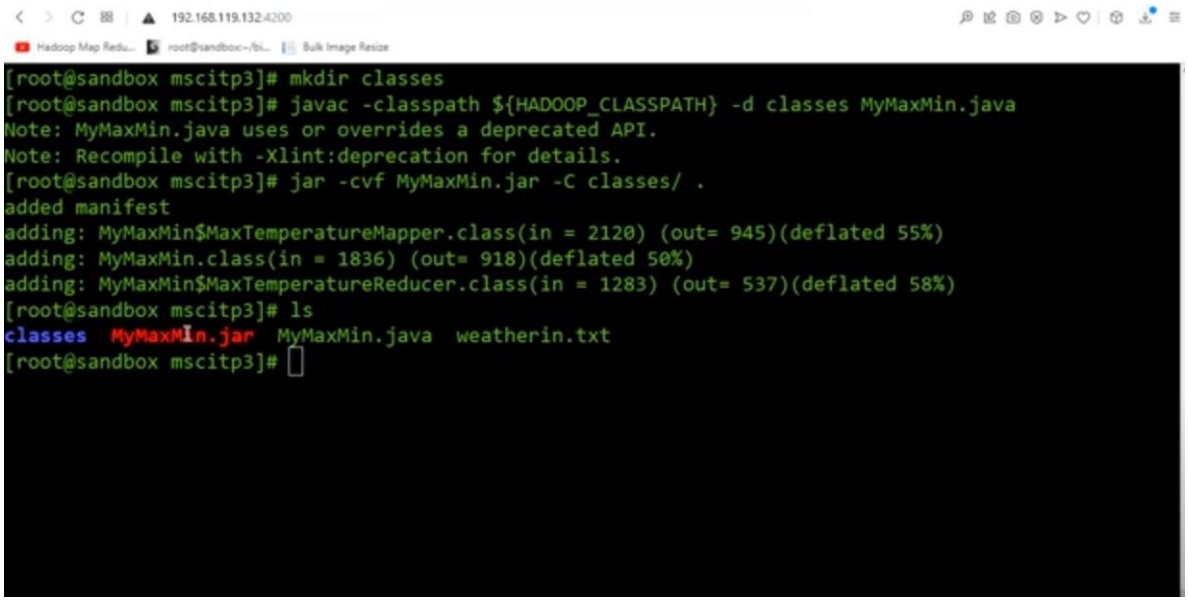
export HADOOP_CLASSPATH=$(hadoop classpath) ////compile and to create jar file

mkdir classes

javac -classpath ${HADOOP_CLASSPATH} -d classes MyMaxMin.java

After compile need to create a jar file

jar -cvf MyMaxMin.jar -C classes/ .



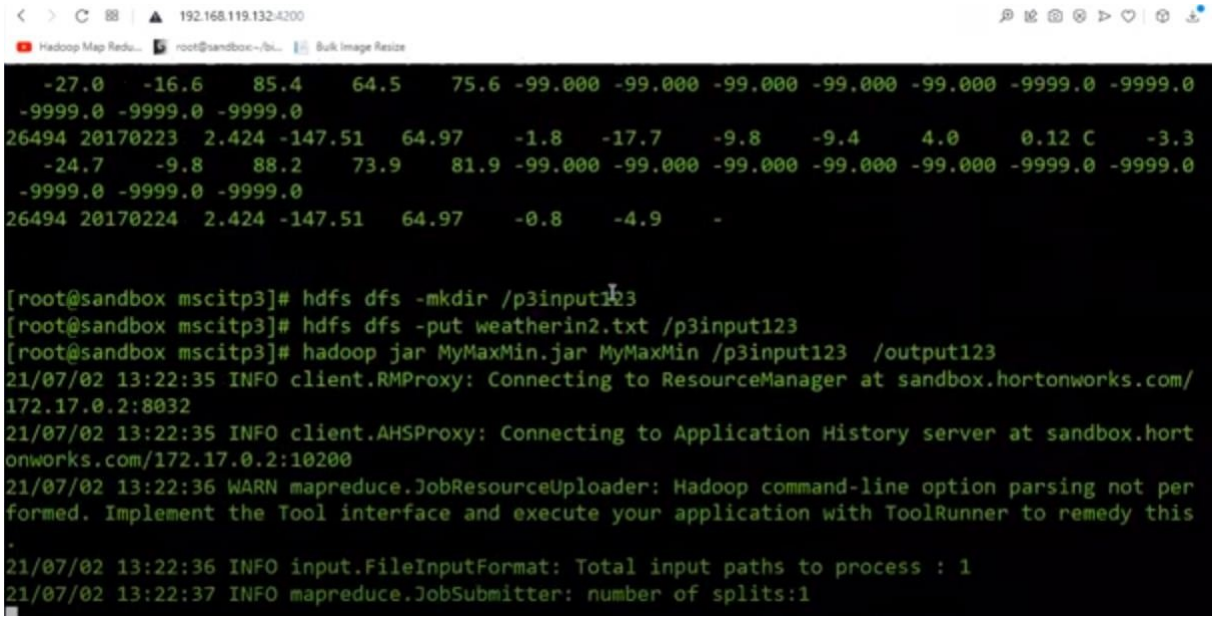
Now, put weatherin.txt in hdfs

Before that create a folder

Command: hdfs dfs -mkdir /p3input123

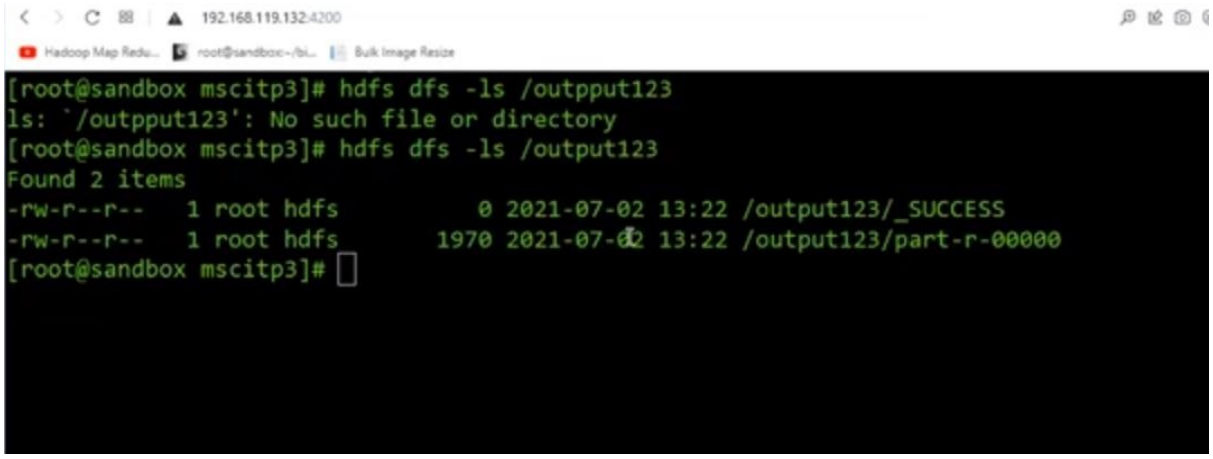
Then run command: hdfs dfs -put weatherin2.txt /p3input123

hadoop jar MyMaxMin.jar MyMaxMin /p3inputw /output123

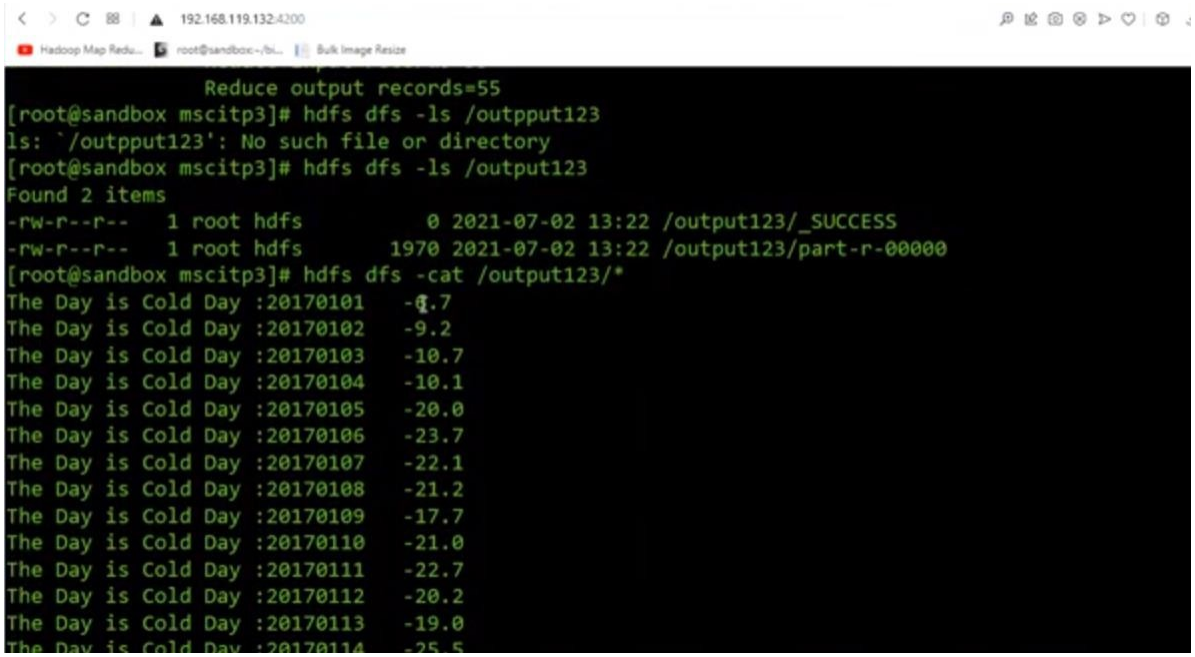


Check outfile is created

Command: hdfs dfs -ls /output123



hdfs dfs -cat /output123/*


192.168.119.132:4200
Hadoop Map Redu...
root@sandbox:~/bi...
Bulk Image Resize
Reduce output records=55
[root@sandbox mscitp3]# hdfs dfs -ls /outpput123
ls: `/outpput123': No such file or directory
[root@sandbox mscitp3]# hdfs dfs -ls /output123
Found 2 items
-rw-r--r-- 1 root hdfs 0 2021-07-02 13:22 /output123/_SUCCESS
-rw-r--r-- 1 root hdfs 1970 2021-07-02 13:22 /output123/part-r-00000
[root@sandbox mscitp3]# hdfs dfs -cat /output123/*
The Day is Cold Day :20170101 -6.7
The Day is Cold Day :20170102 -9.2
The Day is Cold Day :20170103 -10.7
The Day is Cold Day :20170104 -10.1
The Day is Cold Day :20170105 -20.0
The Day is Cold Day :20170106 -23.7
The Day is Cold Day :20170107 -22.1
The Day is Cold Day :20170108 -21.2
The Day is Cold Day :20170109 -17.7
The Day is Cold Day :20170110 -21.0
The Day is Cold Day :20170111 -22.7
The Day is Cold Day :20170112 -20.2
The Day is Cold Day :20170113 -19.0

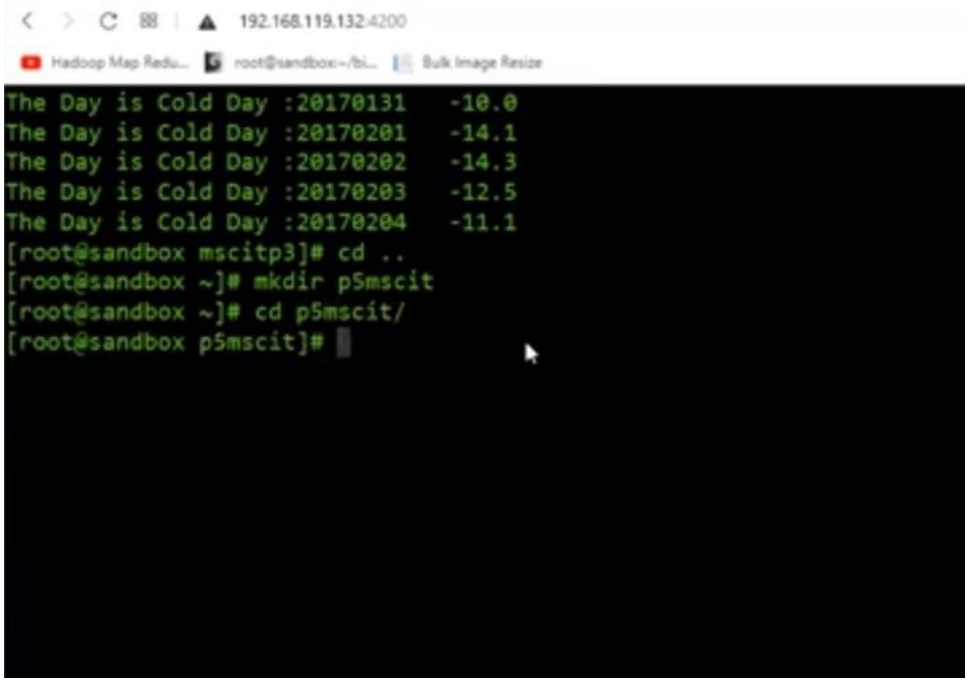
Practical 4

Implement the program using Pig.

Dataset:

001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
#student.txtcreate a directory and get into that directory

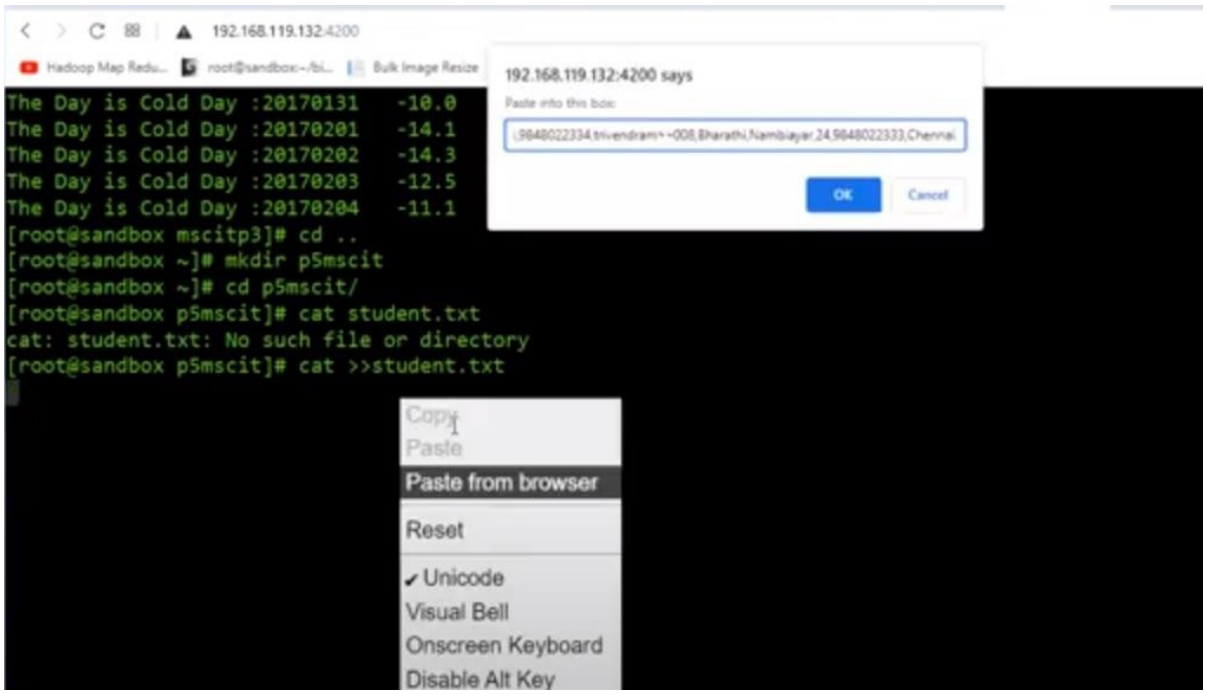
Command: mkdir p5mscit

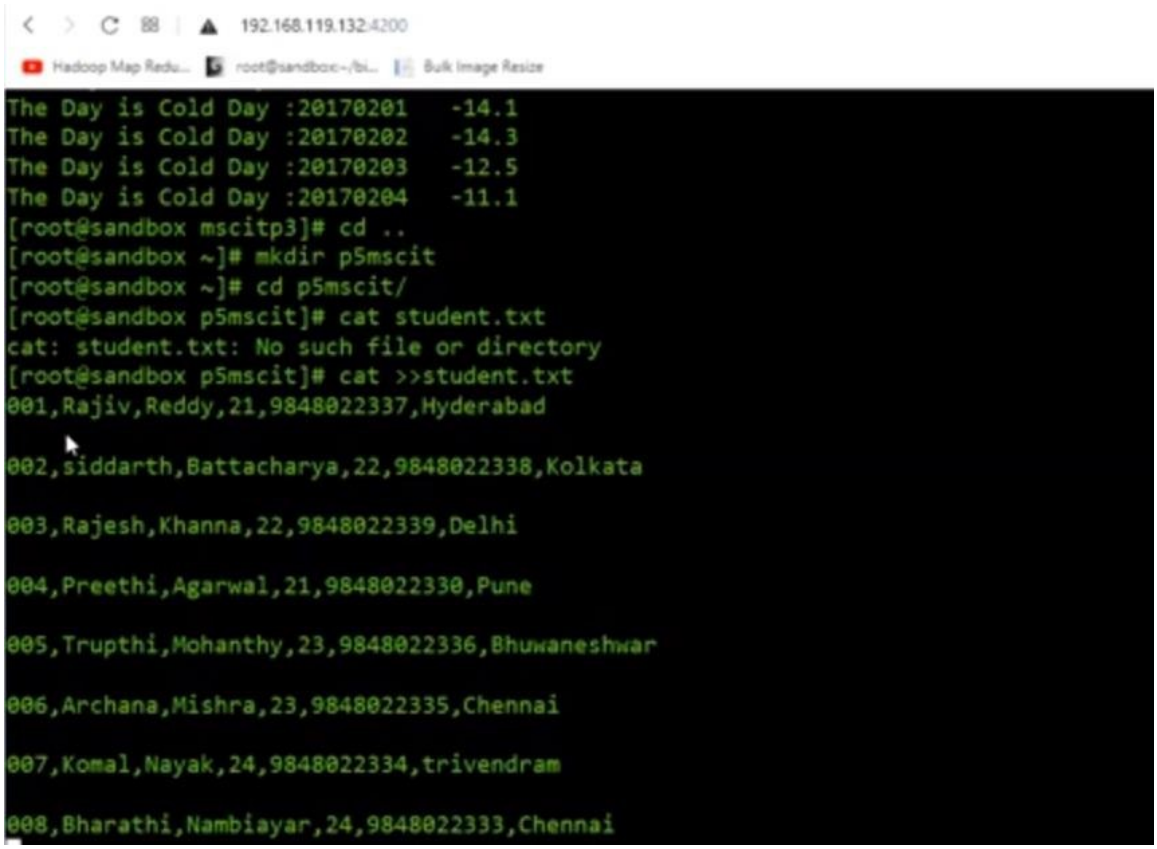


Create a file

Command: cat >>student.txt

Right click and paste the text



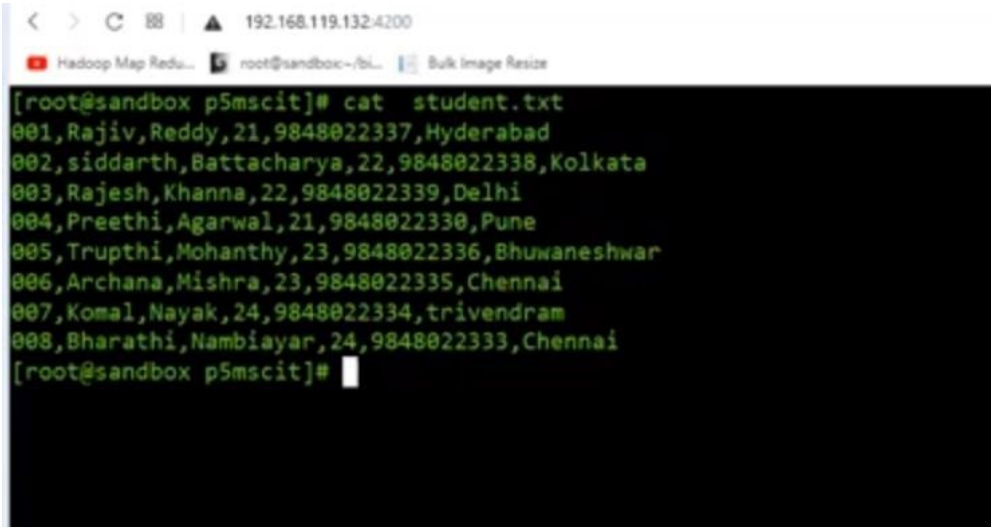


Remove the space with vi editor

Command: vi student.txt and press i for insert mode

After editing: wq and enter

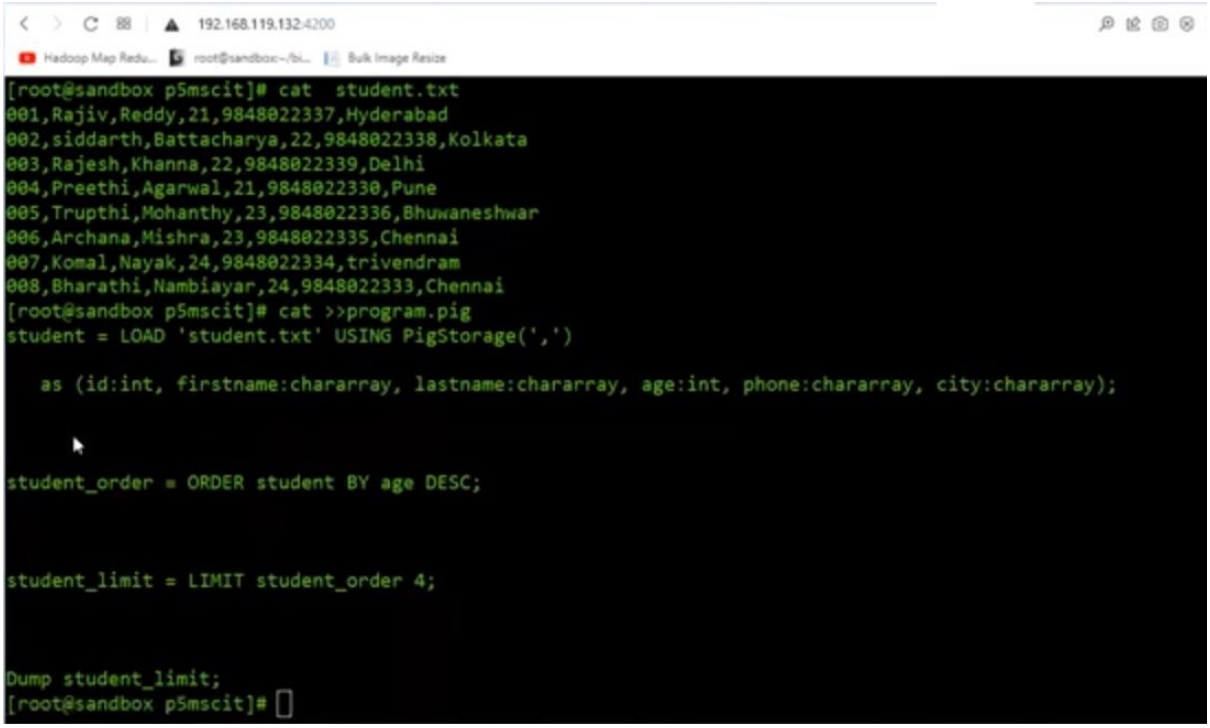
Print the content and see the text



Create a program file

//////////////////////////script start
student = LOAD 'student.txt' USING PigStorage(',')
  as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

student_order = ORDER student BY age DESC;student_limit = LIMIT student_order 4;Dump student_limit;
////////script end



Upload student on hdfs

Command: hdfs dfs -put student.txt /user/root/

Run the pig program

```
[root@sandbox p5mscit]# hdfs dfs -put student.txt /user/root/
[root@sandbox p5mscit]# pig program.pig
```

Output:

```
dent.txt"

Output(s):
Successfully stored 4 records (211 bytes) in: "hdfs://sandbox.hortonworks.com:8020/tmp/temp81689
1614/tmp500376273"

2021-07-02 13:51:34,079 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Tot
al input paths to process : 1
2021-07-02 13:51:34,079 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUt
il - Total input paths to process : 1
(7,Komal,Nayak,24,9848022334,trivendram)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,23,9848022335,Chennai)
2021-07-02 13:51:34,563 [main] INFO  org.apache.pig.Main - Pig script completed in 39 seconds an
d 726 milliseconds (39726 ms)
2021-07-02 13:51:34,575 [main] INFO  org.apache.pig.backend.hadoop.executionengine.tez.TezLaunch
er - Shutting down thread pool
2021-07-02 13:51:34,608 [pool-1-thread-1] INFO  org.apache.pig.backend.hadoop.executionengine.te
z.TezSessionManager - Shutting down Tez session org.apache.tez.client.TezClient@132aba44
2021-07-02 13:51:34 Shutting down Tez session , sessionName=PigLatin:program.pig, applicationId=
application_1625229346715_0004
```

Practical 5

Implement the application in Hive.

Dataset:

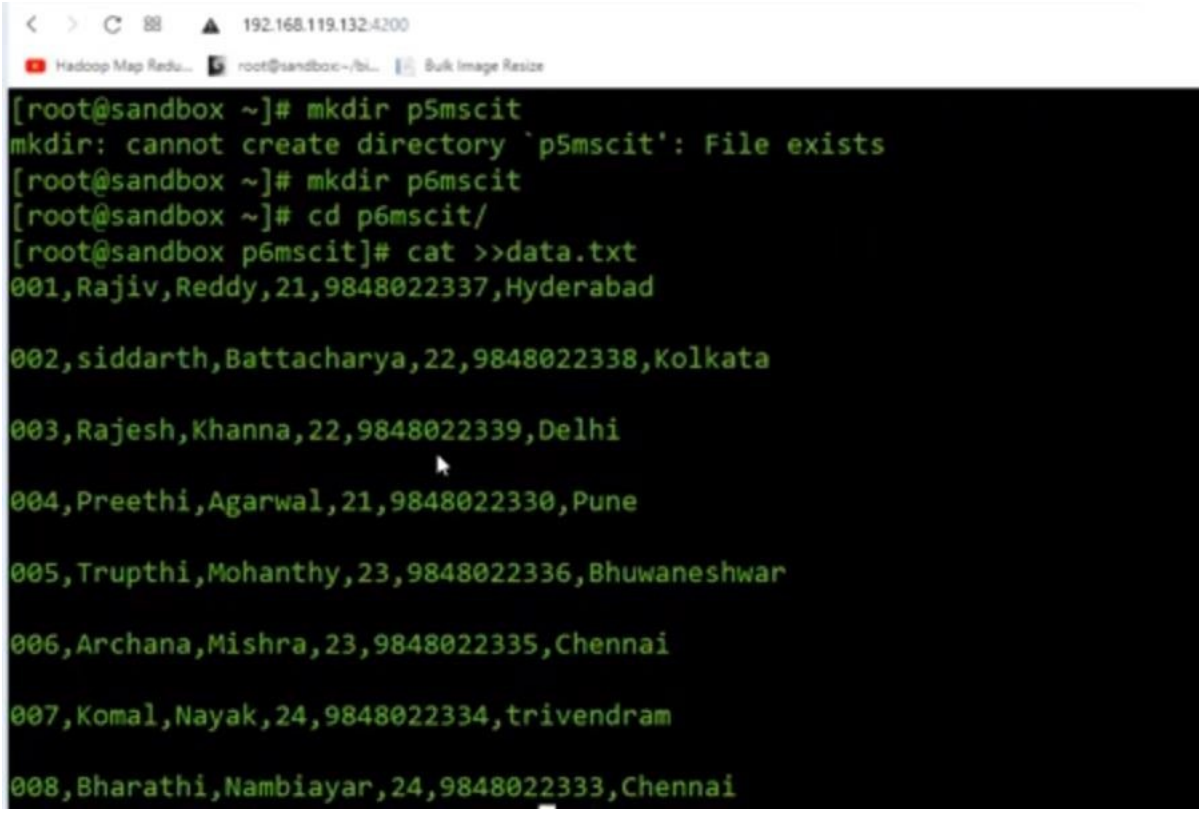
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai
#student.txtcreate a directory and get into that directory

Command: mkdir p6mscit

Create a file

Command: cat >>data.txt

Right click and paste the text



Remove the space with vi editor

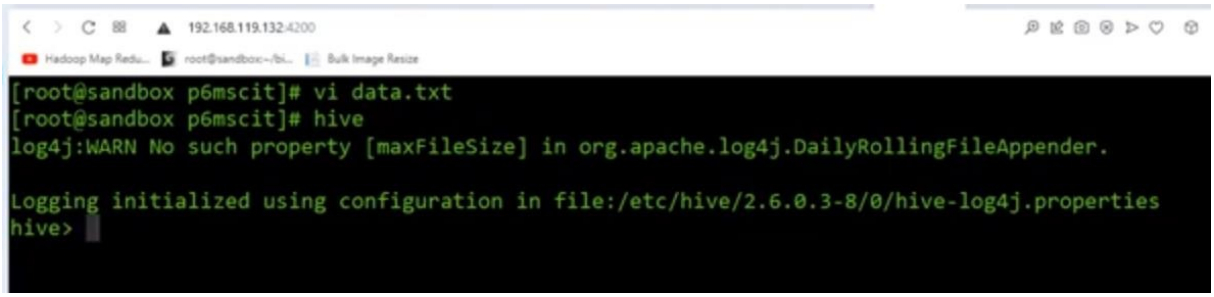
Command: vi student.txt and press i for insert mode

After editing: wq and enter

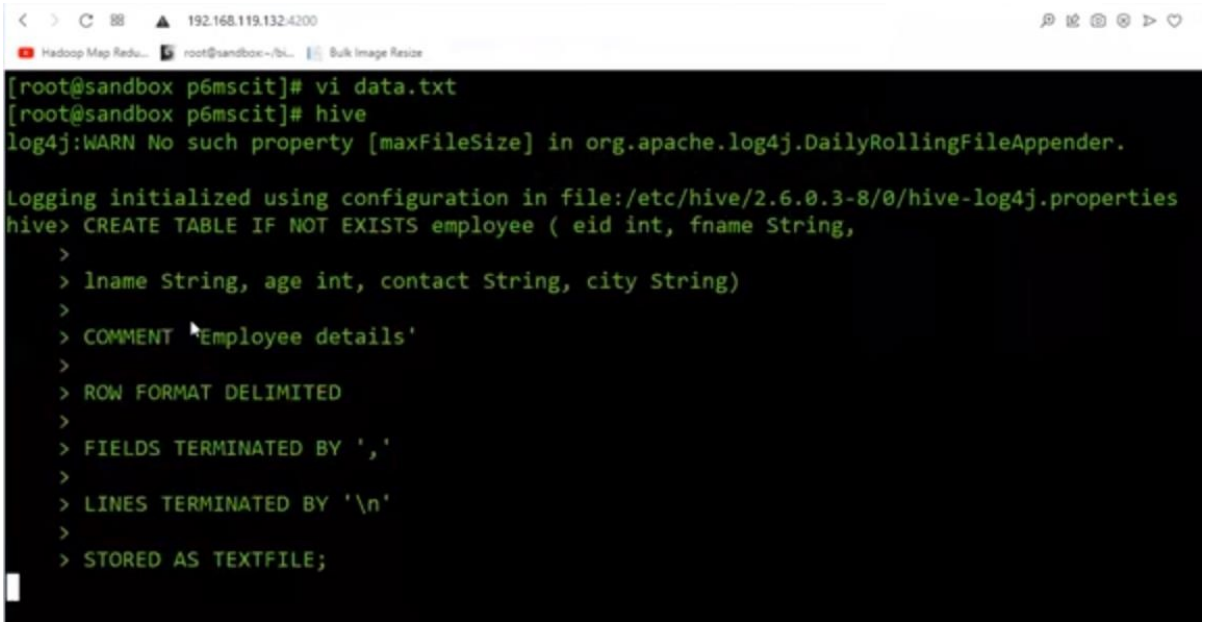
Print the content and see the text

Now start the hive terminal

Command: hive



Copy paste below command on hive and entercreate table
CREATE TABLE IF NOT EXISTS employee (eid int, fname String,
lname String, age int, contact String, city String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

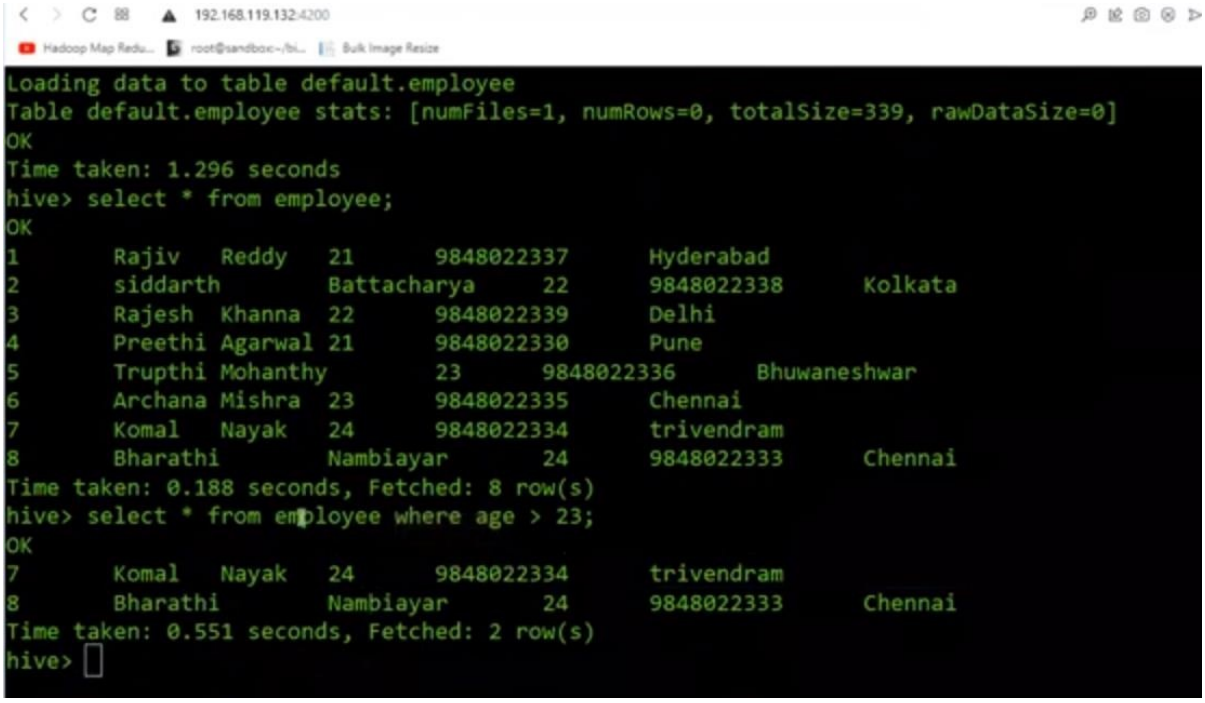


Run command: LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO

TABLE employee;

```
hive> LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=0, totalSize=339, rawDataSize=0]
OK
Time taken: 1.296 seconds
hive> ;
```

Run the command like select * from employee;

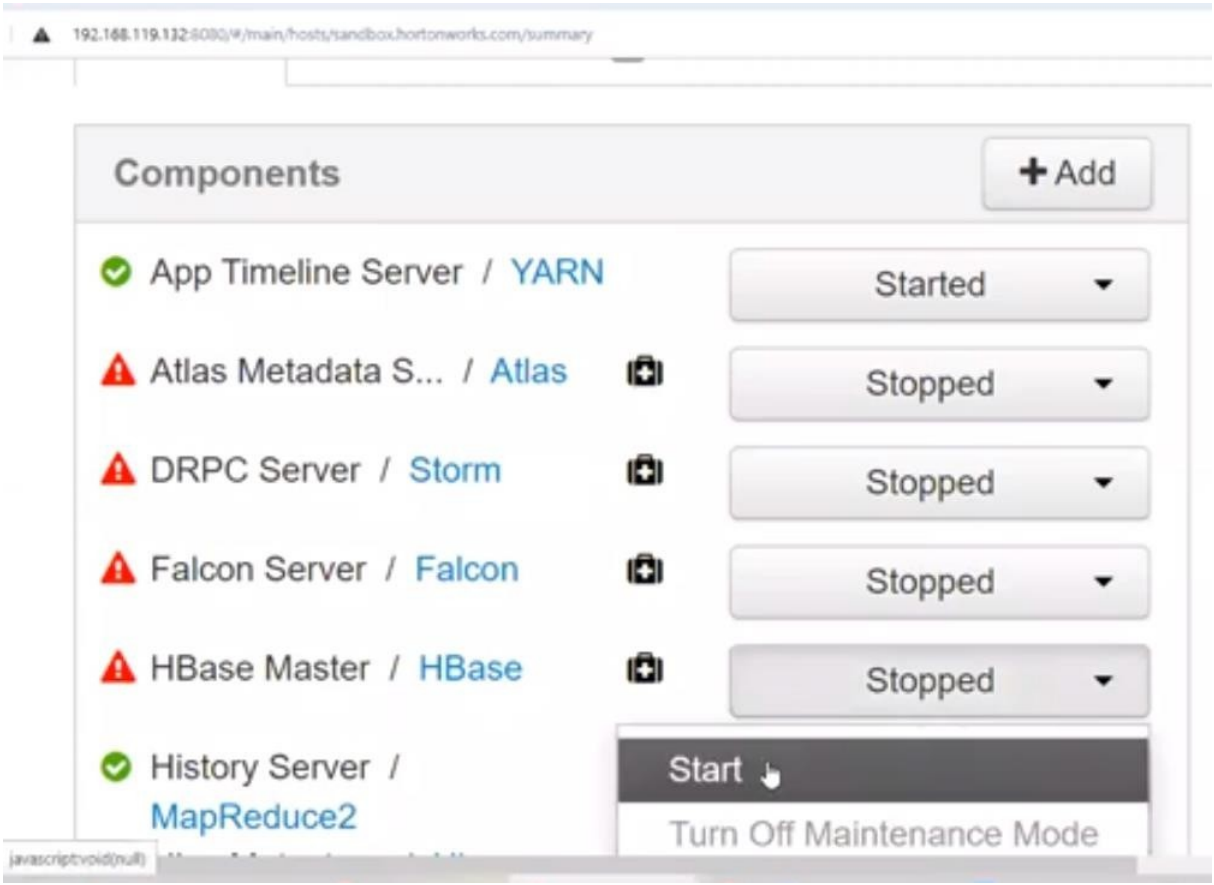


Practical 6

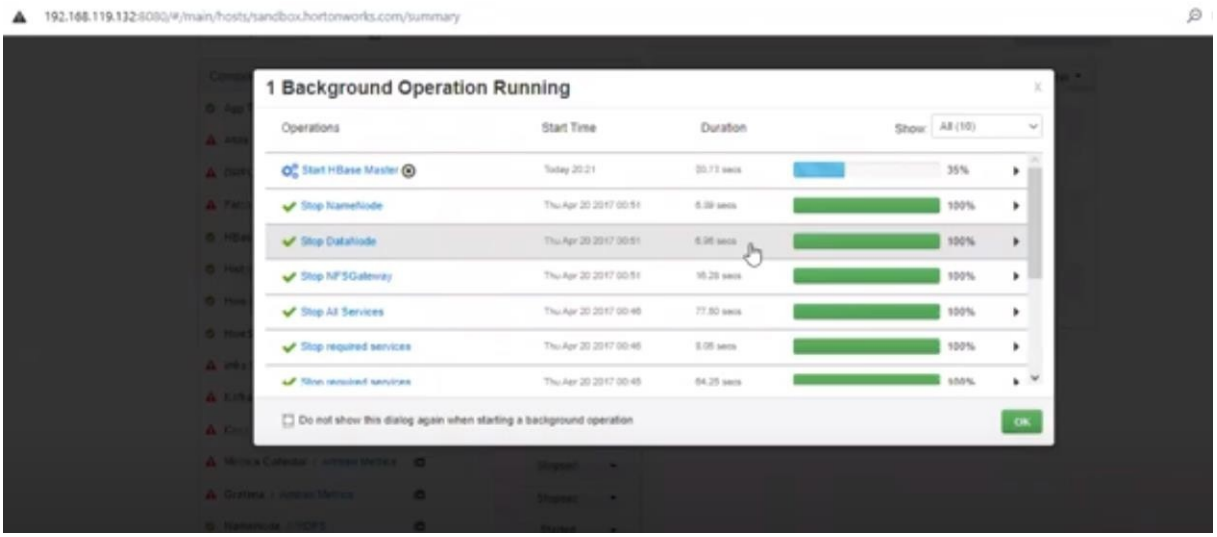
Implement an application that stores big data in Hbase/

Python What is HBase?

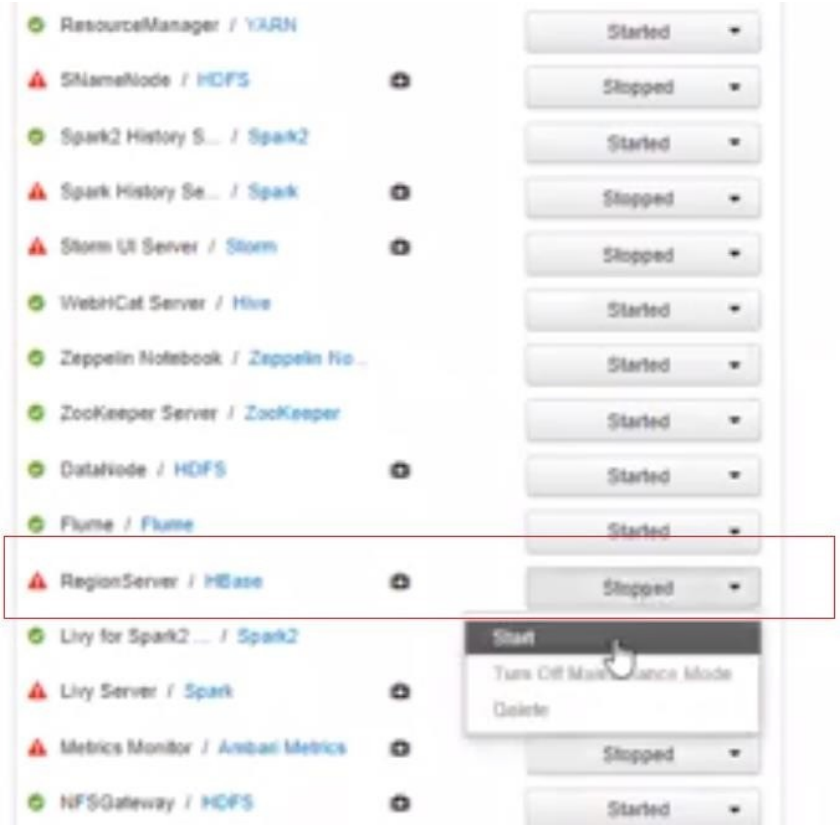
HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.Go to GUI page and start the hbase service.

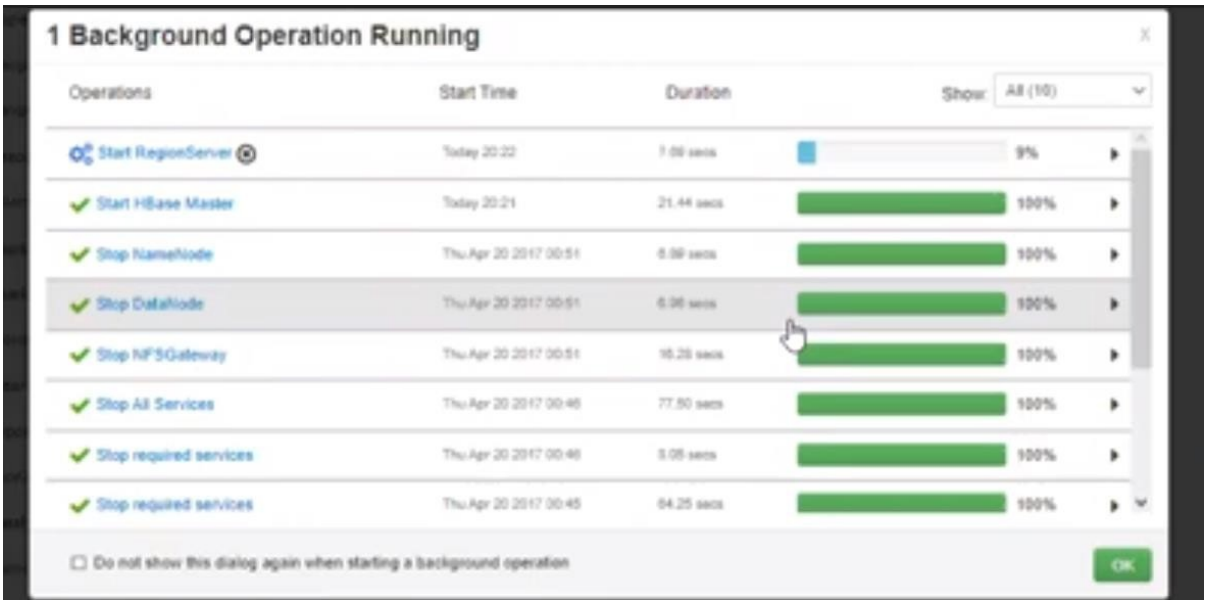


Click on OK to start the service.

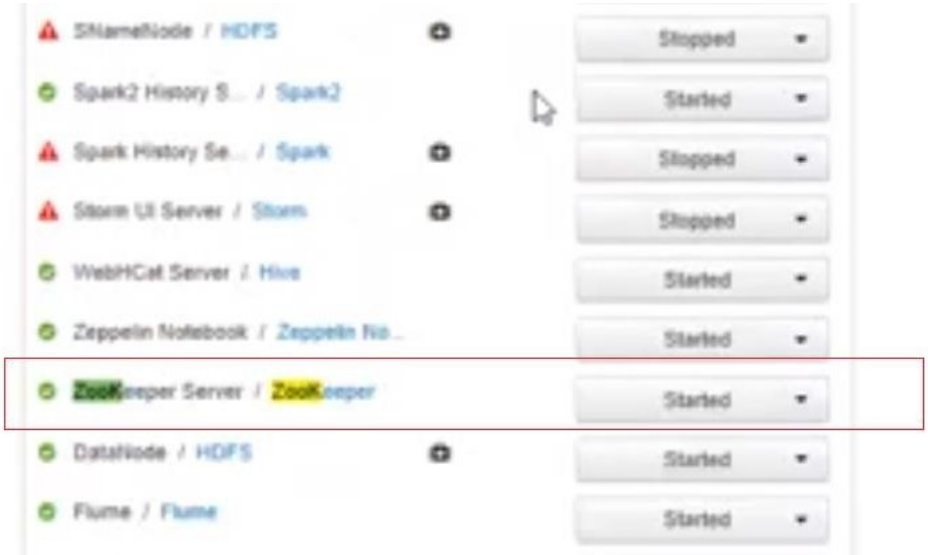


Now we must start region server.

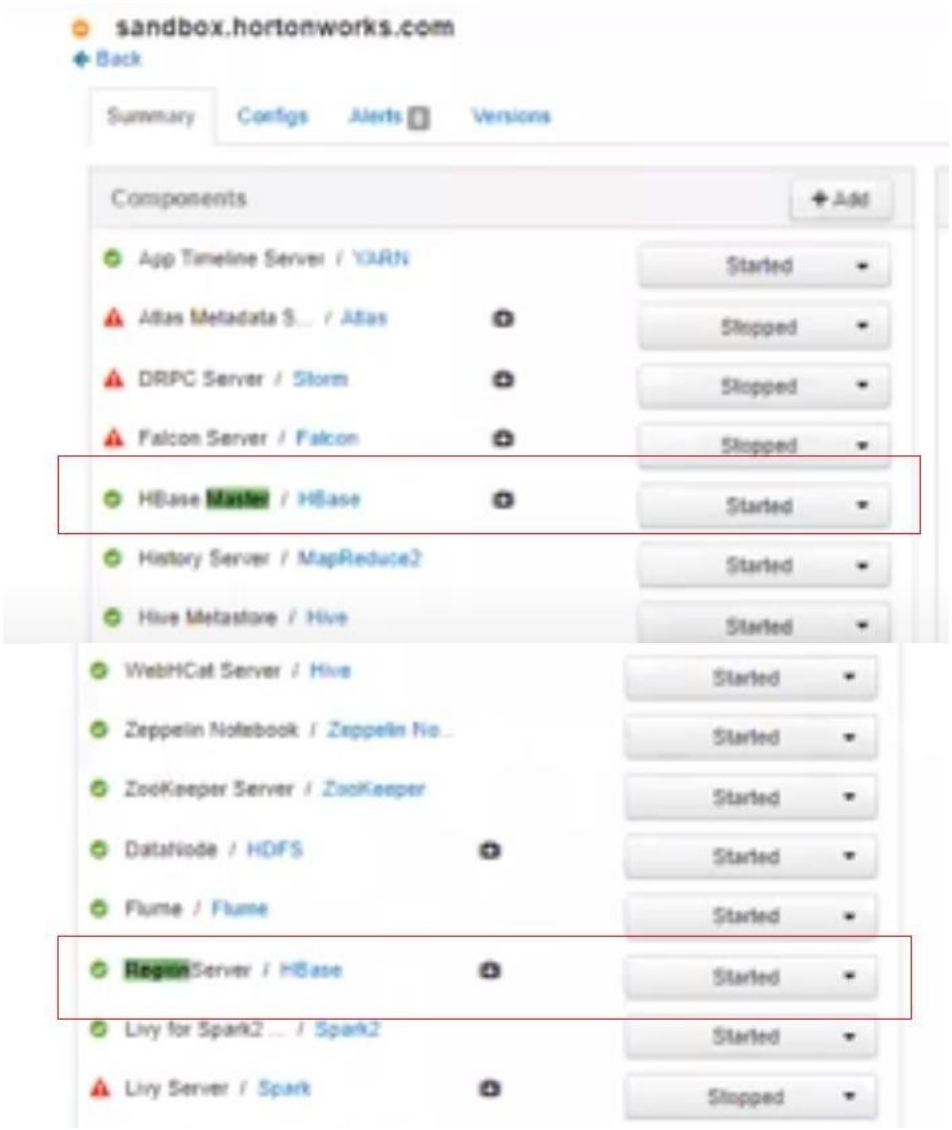




Check zooperkeeper server is started.



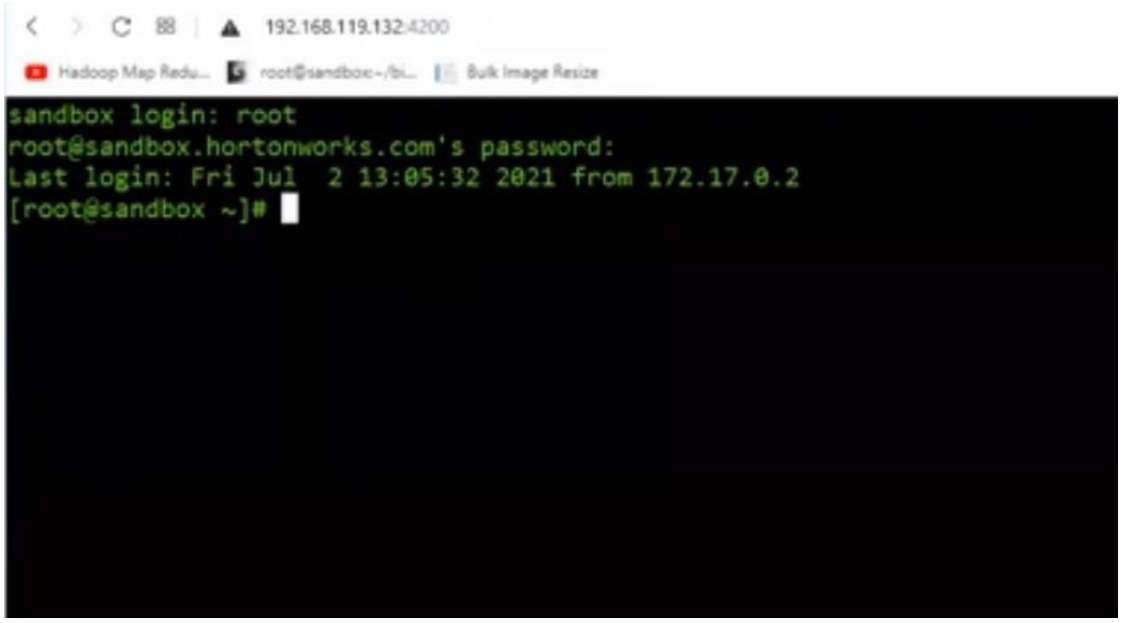
Check hbase and region server are started.



Command: **whichapplication-name** gives directory in which application-name is installed.

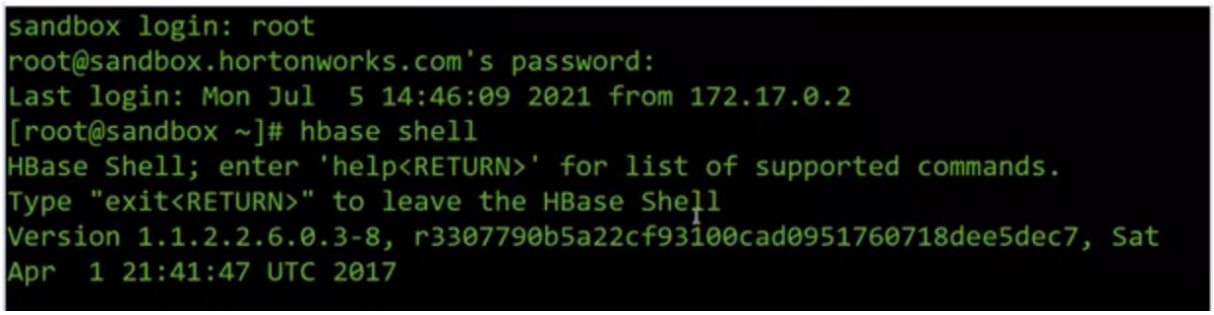
Open the shell

192.168.119.132:4200

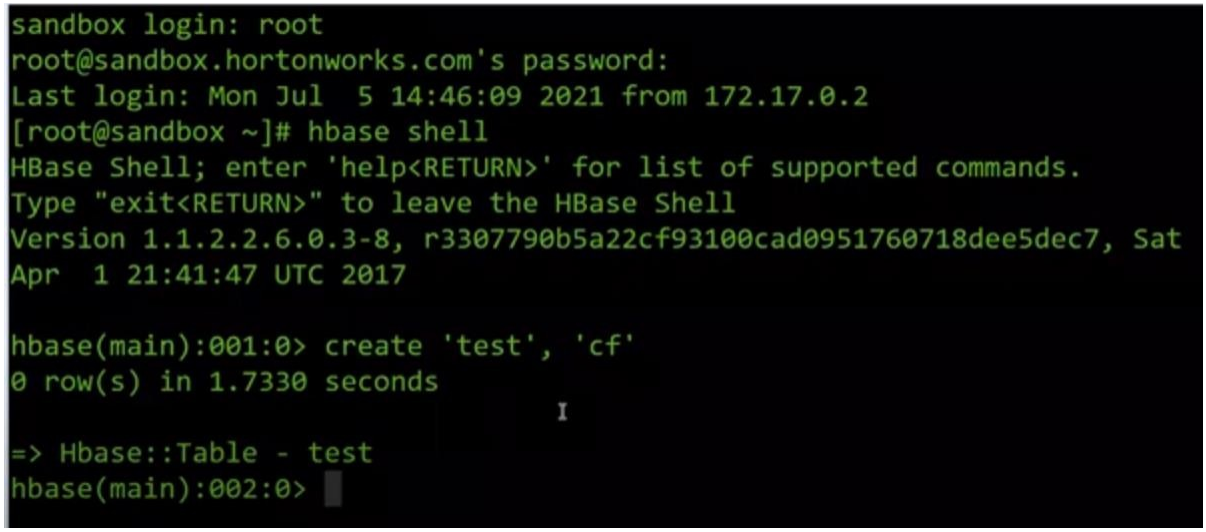


Command: hbase shell

It will start the server



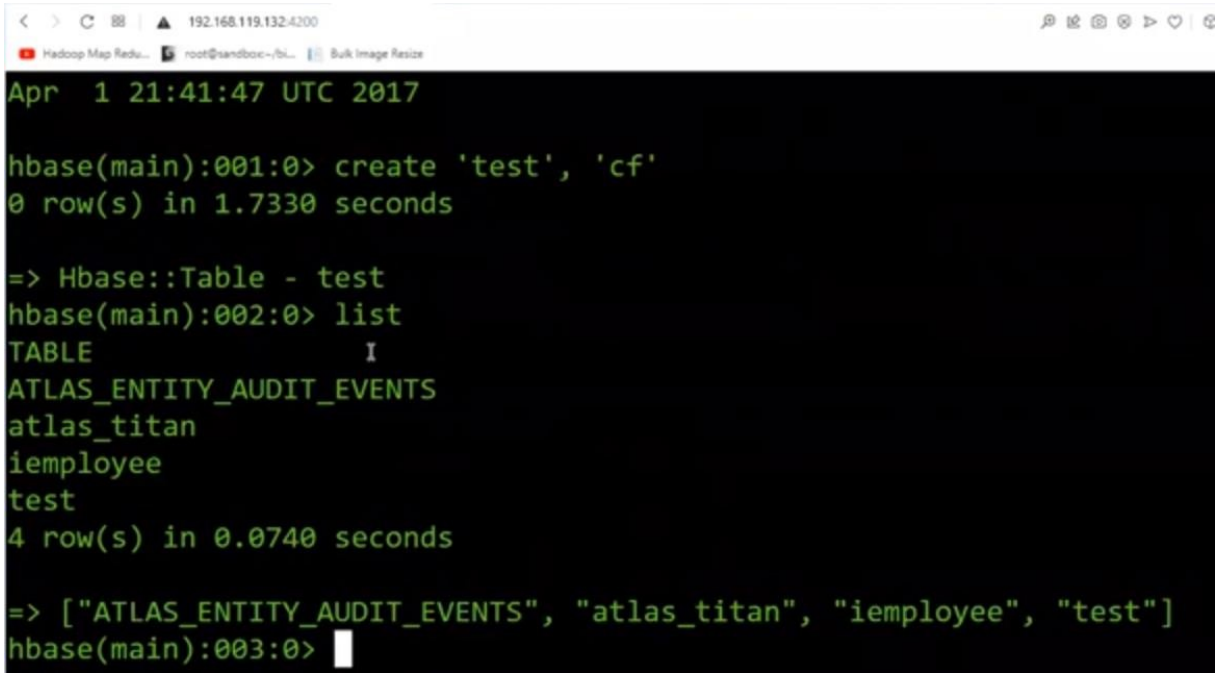
Enter the command create ‘test’, ‘cf’ and it will create the table



Check the table is created with

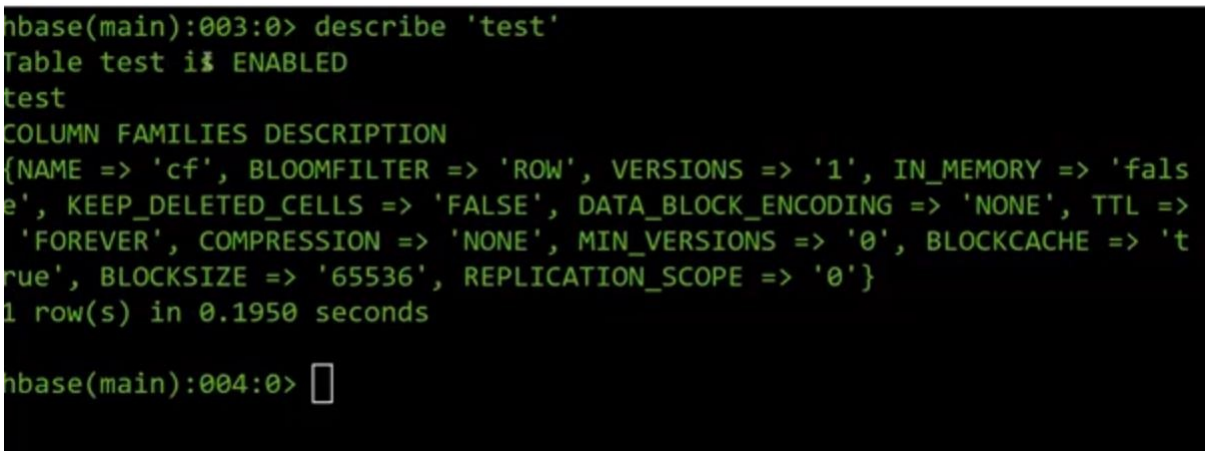
command List- It will list all the tables

created.



If we want to see column description of a table.

Command- describe tablename



Now, we have to put the values in table

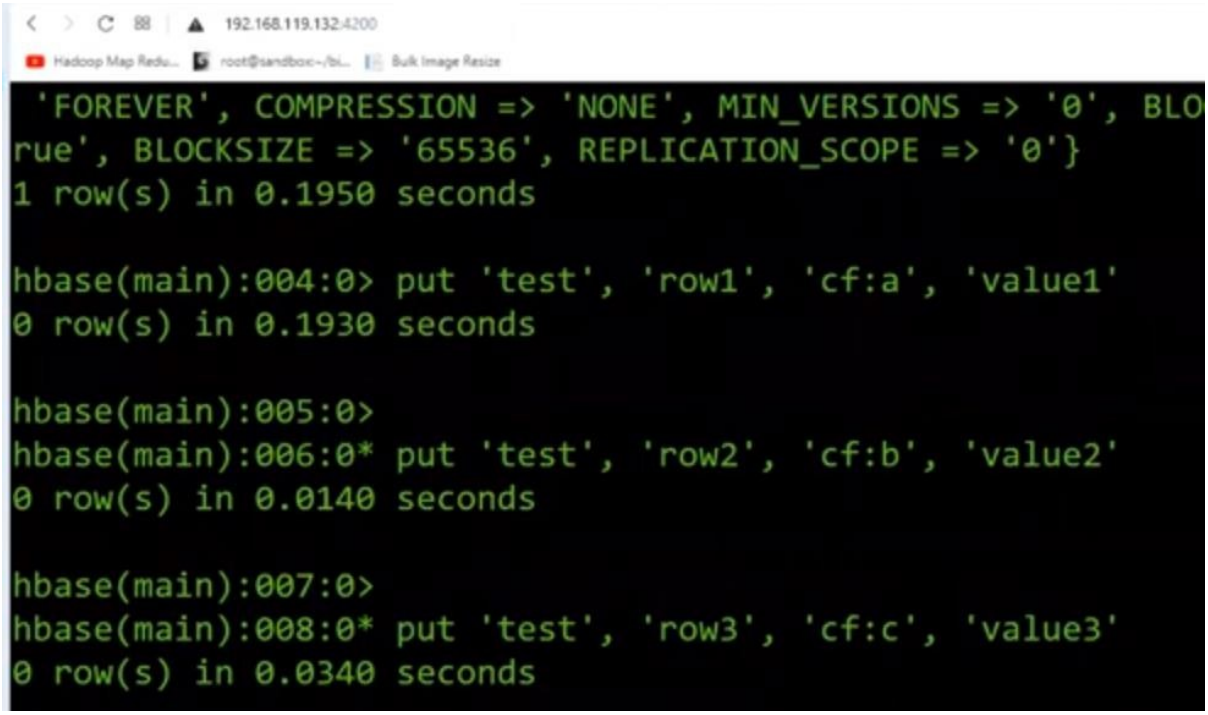
Values:

put ‘test’, ‘row1’, ‘cf:a’, ‘value1’

put ‘test’, ‘row2’, ‘cf:b’, ‘value2’

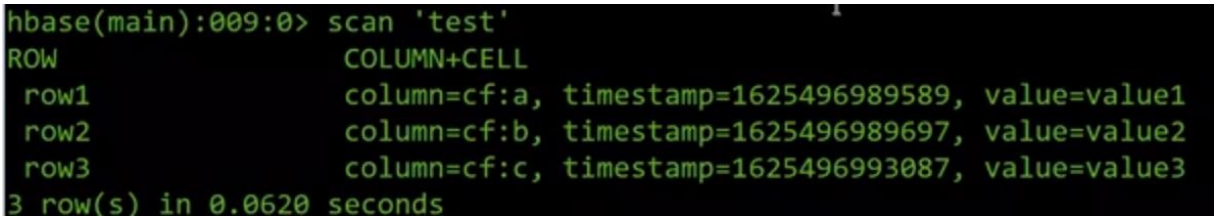
put ‘test’, ‘row3’, ‘cf:c’, ‘value3’

copy paste the data in shell.



We to display the records of table

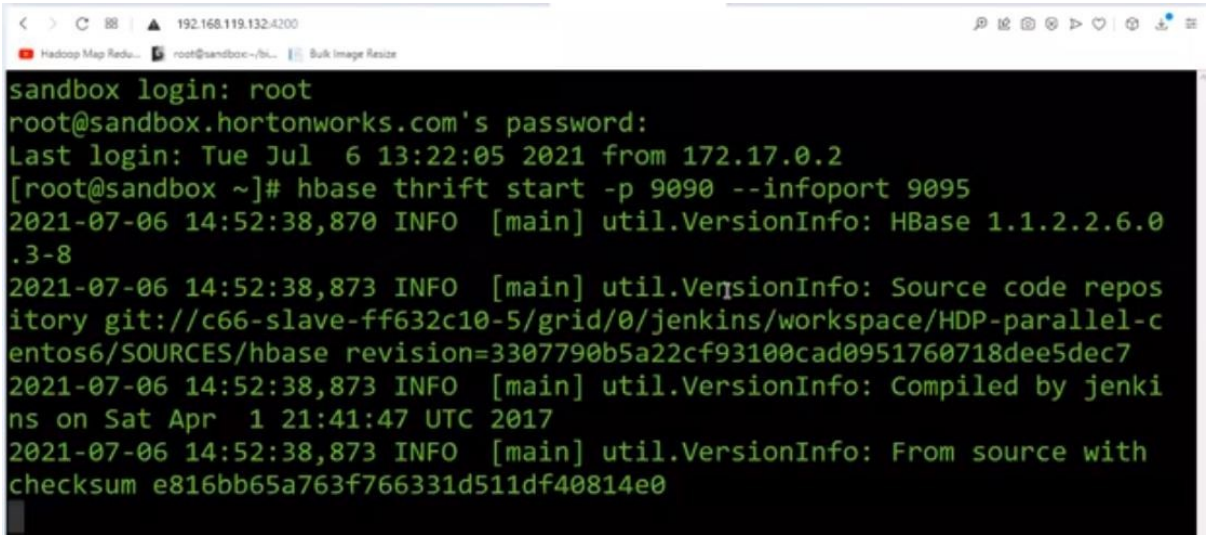
Command: scan ‘test’



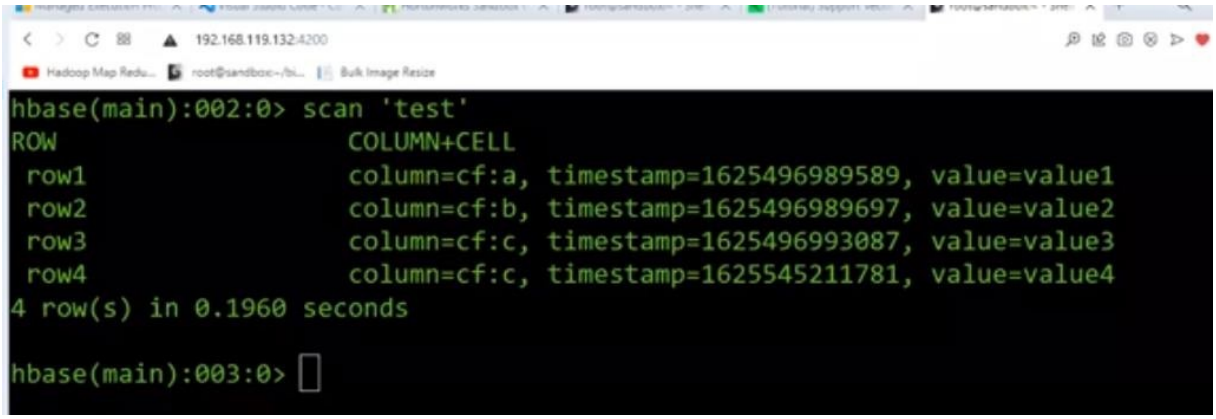
Python: storage/retrieval

Start the service with command

Hbase thrift start -p 9090 –inforport 9095



Create the table the way we did it in hbase and see the records using scan command



Create a program file

Import happybase as hb

conn=hb.connection('192.168.119.132', 9090)

print(conn.table('test').row('row1')

print(conn.table('test').row('row2')

print(conn.table('test').row('row3')

print(conn.table('test').row('row4')

table = conn.table('test')

table.put(b'row5', {b'cf:r': b'value5'})

print(conn.table('test').row('row5')

```
import happybase as hb
conn=hb.Connection('192.168.119.132', 9

print(conn.table('test').row('row1'))
print(conn.table('test').row('row2'))
print(conn.table('test').row('row3'))
print(conn.table('test').row('row4'))

table = conn.table('test')
table.put(b'row5', {b'cf:r': b'value5'})

print(conn.table('test').row('row5'))
```

```
all class\Big data\All code and steps\
hbase2.py
>>>

====== RESTART: C:\Users\Ganesh\Desktop\Pract
icle\Big Data\hbaseprogram.py =====
{b'cf:a': b'value1'}
{b'cf:b': b'value2'}
{b'cf:c': b'value3'}
{b'cf:r': b'value5'}
>>>

====== RESTART: C:\Users\Ganesh\Desktop\Pract
icle\Big Data\hbaseprogram.py =====

{b'cf:a': b'value1'}
{b'cf:b': b'value2'}
{b'cf:c': b'value3'}
{b'cf:c': b'value4'}
{b'cf:r': b'value5'}
>>>
```

Run a scan command on shell to display the values

```
hbase(main):004:0> scan 'test'
ROW                   COLUMN+CELL
 row1                 column=cf:a, timestamp=1625496989589, value=value1
 row2                 column=cf:b, timestamp=1625496989697, value=value2
 row3                 column=cf:c, timestamp=1625496993087, value=value3
 row4                 column=cf:c, timestamp=1625545211781, value=value4
 row5                 column=cf:r, timestamp=1625583481042, value=value5
5 row(s) in 0.0320 seconds
```

Now, try with duplicate value at row 5 say value t

```
import happybase as hb
conn=hb.Connection('192.168.119.132', 9

print(conn.table('test').row('row1'))
print(conn.table('test').row('row2'))
print(conn.table('test').row('row3'))
print(conn.table('test').row('row4'))

table = conn.table('test')
table.put(b'row5', {b'cf:t': b'value5'})

print(conn.table('test').row('row5'))
```

```
>>>
====== RESTART: C:\Users\Ganesh\Desktop\Pract
icle\Big Data\hbaseprogram.py =====

{b'cf:a': b'value1'}
{b'cf:b': b'value2'}
{b'cf:c': b'value3'}
{b'cf:c': b'value4'}
{b'cf:r': b'value5'}
>>>

====== RESTART: C:\Users\Ganesh\Desktop\Pract
icle\Big Data\hbaseprogram.py =====
{b'cf:a': b'value1'}
{b'cf:b': b'value2'}
{b'cf:c': b'value3'}
{b'cf:c': b'value4'}
{b'cf:r': b'value5', b'cf:t': b'value5
'}
```

Run a scan command on shell to display the values

When there is unique value, it will create a record. If duplicate value it will not create a record

```
hbase(main):005:0> scan 'test'
ROW                   COLUMN+CELL
 row1                 column=cf:a, timestamp=1625496989589, value=value1
 row2                 column=cf:b, timestamp=1625496989697, value=value2
 row3                 column=cf:c, timestamp=1625496993087, value=value3
 row4                 column=cf:c, timestamp=1625545211781, value=value4
 row5                 column=cf:r, timestamp=1625583481042, value=value5
 row5                 column=cf:t, timestamp=1625583505297, value=value5
5 row(s) in 0.1320 seconds
```

Practical 7

Implement Decision tree classification techniques**Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.Using the Iris dataset, we can construct a tree as follows:

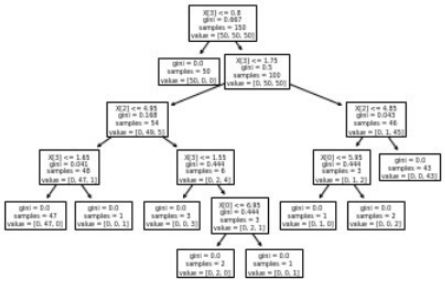
+ Code + Text

```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
X, y = iris.data, iris.target
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)
```

Once trained, we can plot the tree with the **plot_tree** function:

```
[4]  tree.plot_tree(clf)
```

```
[Text(167.4, 199.32, 'X[3] <= 0.8\ngini = 0.667\nsamples = 150\nvalue = [50, 50, 50]'),
 Text(141.64615384615385, 163.07999999999998, 'gini = 0.0\nsamples = 50\nvalue = [50, 0, 0]'),
 Text(193.15384615384616, 163.07999999999998, 'X[3] <= 1.75\ngini = 0.5\nsamples = 100\nvalue = [0, 50, 50]'),
 Text(103.01538461538462, 126.83999999999999, 'X[2] <= 4.95\ngini = 0.168\nsamples = 54\nvalue = [0, 49, 5]'),
 Text(51.50769230769231, 90.6, 'X[3] <= 1.65\ngini = 0.041\nsamples = 48\nvalue = [0, 47, 1]'),
 Text(25.753846153846155, 54.359999999999985, 'gini = 0.0\nsamples = 47\nvalue = [0, 47, 0]'),
 Text(77.26153846153846, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(154.52307692307693, 90.6, 'X[3] <= 1.55\ngini = 0.444\nsamples = 6\nvalue = [0, 2, 4]'),
 Text(128.76923076923077, 54.359999999999985, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3]'),
 Text(180.27692307692308, 54.359999999999985, 'X[0] <= 6.95\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(154.52307692307693, 18.119999999999976, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(206.03076923076924, 18.119999999999976, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(283.2923076923077, 126.83999999999999, 'X[2] <= 4.85\ngini = 0.043\nsamples = 46\nvalue = [0, 1, 45]'),
 Text(257.53846153846155, 90.6, 'X[0] <= 5.95\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(231.7846153846154, 54.359999999999985, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(283.2923076923077, 54.359999999999985, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(309.04615384615386, 90.6, 'gini = 0.0\nsamples = 43\nvalue = [0, 0, 43]')]
```



Practical 8

Implement SVM classification techniques

Support Vector Machines

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.**Loading data:**

```
[1] #Import scikit-learn dataset library
    from sklearn import datasets

    #Load dataset
    cancer = datasets.load_breast_cancer()
```

Exploring data:

```
# print the names of the 13 features
print("Features: ", cancer.feature_names)

# print the label type of cancer('malignant' 'benign')
print("Labels: ", cancer.target_names)
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
Labels:  ['malignant' 'benign']
```

Check the shape of the dataset using shape.

```
# print data(feature)shape
cancer.data.shape
```

```
(569, 30)
```

Check top 5 records of the feature set.

```
[4]  # print the cancer data features (top 5 records)
     print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

Target set:

```
[5]  # print the cancer labels (0:malignant, 1:benign)
     print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

Splitting Data:

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function train_test_split(). you need to pass 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
[7]  # Import train_test_split function
     from sklearn.model_selection import train_test_split

     # Split dataset into training set and test set
     X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.3,random_state=109) # 70% training and 30% test
```

Generate Model:

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.

Then, fit your model on train set using fit() and perform prediction on the test set using predict().

```
[8]  #Import svm model
     from sklearn import svm

     #Create a svm Classifier
     clf = svm.SVC(kernel='linear') # Linear Kernel

     #Train the model using the training sets
     clf.fit(X_train, y_train)

     #Predict the response for test dataset
     y_pred = clf.predict(X_test)
```

Evaluating the Model:

Let's estimate how accurately the classifier or model can predict the breast cancer of patients. Accuracy can be computed by comparing actual test set values and predicted values.

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
```

Vidya Prasarak Mandal's

B. N. BANDODKAR COLLEGE OF SCIENCE (AUTONOMOUS), THANE.

(Affiliated to University of Mumbai)

NAAC REACCREDITED 'A' GRADE

Best College Award, University of Mumbai

माहिती व तंत्रज्ञान विभाग

दूरध्वनी क. २५३३ ६५०७

Department of Information Technology

Tel. No. 2533 6507

Email : itbnb@vpmthane.org

CERTIFICATE

This is to certify that

Shri / Kum. _____

of M. Sc. (Information Technology) Part I Semester - II has completed the required number of experiments (Total =) signed herein, in this laboratory during the year 2023 – 2024.

Seal

Incharge
Department of Information Technology

Principal
B. N. Bandodkar College of Science, Thane

_____
External Examiner