Program: 1

AIM: To install Hadoop

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

STEPS TO INSTALL HADOOP:

- Install JAVA JDK 1.8
- Download Hadoop and extract and place under C drive
- Set Path in Environment Variables
- Config files under Hadoop directory
- Create folder datanode and namenode under data directory
- Edit HDFS and YARN files
- Set Java Home environment in Hadoop environment
- Setup Complete. Test by executing start-all.cmd

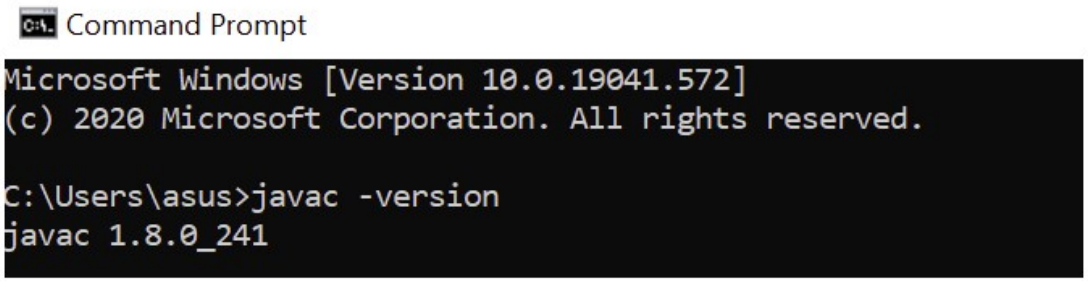
1.Install JAVA

Java JDK Link to download

https://www.oracle.com/java/technologies/javase-jdk8-downloads.html

– extract and install Java in C:\Java

– open cmd and type -> javac -version



2.Download Hadoop

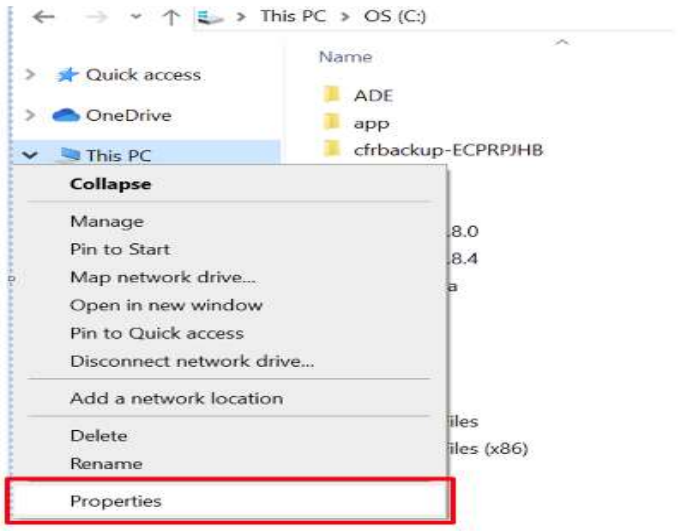
– https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz

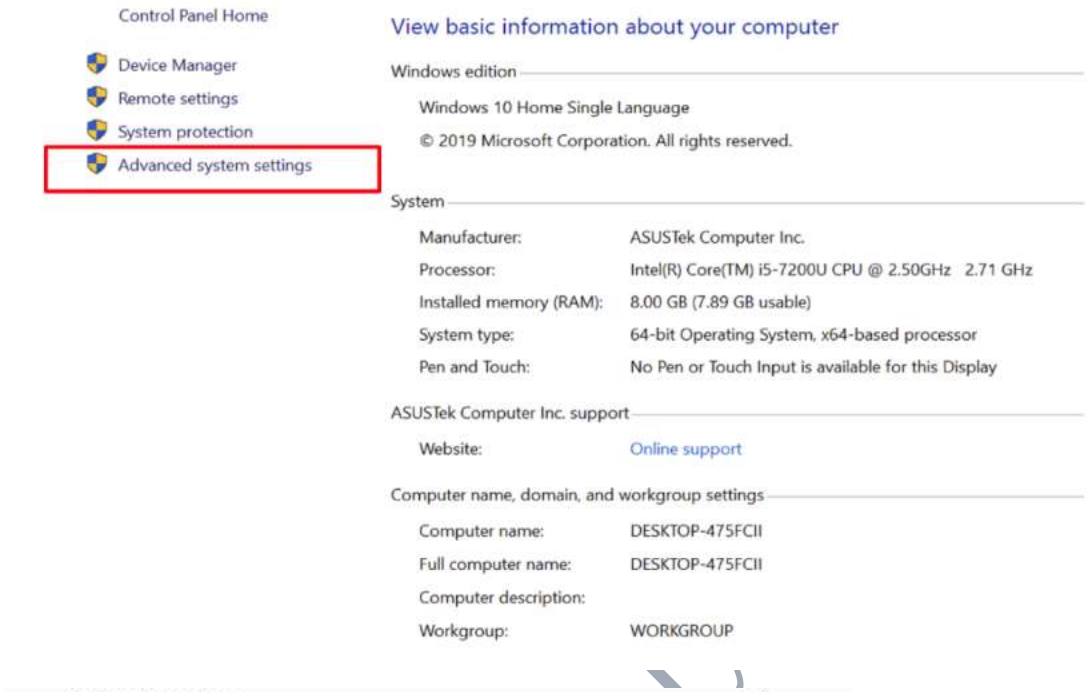
– extract to C:\Hadoop

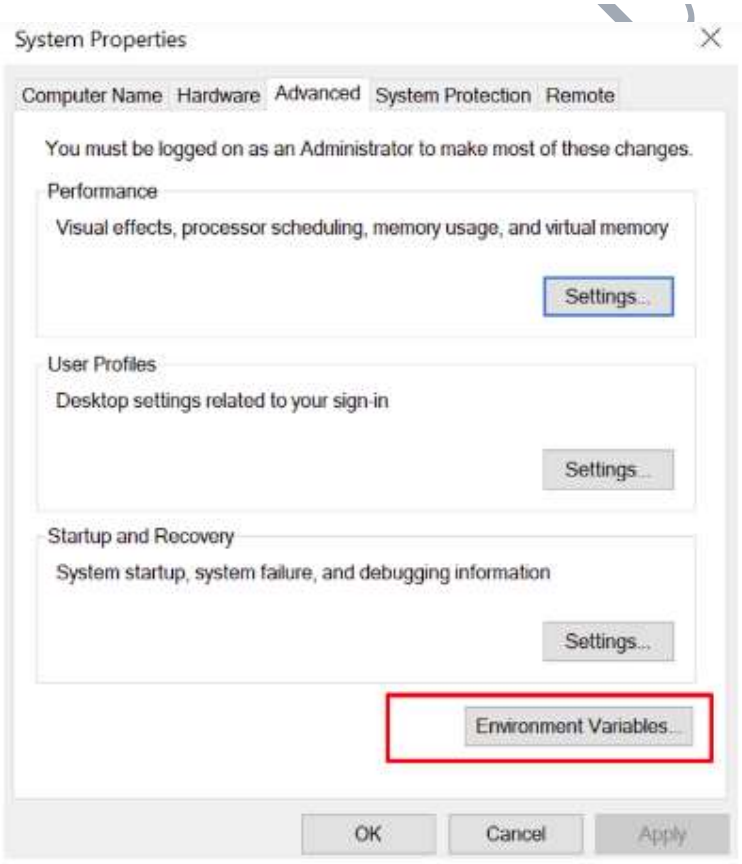
ADE	1/26/2020 11:13 AM	File folder
app	1/26/2020 10:53 AM	File folder
cfrbackup-ECPRPJHB	4/18/2019 10:25 PM	File folder
eSupport	7/13/2017 5:22 AM	File folder
Games	8/20/2019 9:40 PM	File folder
hadoop	11/8/2020 3:15 PM	File folder
hadoop-2.8.0	12/10/2019 3:02 PM	File folder
hadoop-2.8.4	6/14/2019 9:36 PM	File folder
hadoop-3.3.0	11/8/2020 4:30 PM	File folder
Hortanwork	11/8/2020 2:40 PM	File folder
Informatica	1/28/2020 12:52 AM	File folder
Java	11/8/2020 3:25 PM	File folder
logs	3/27/2020 9:36 PM	File folder
oraclexe	1/29/2020 11:52 PM	File folder

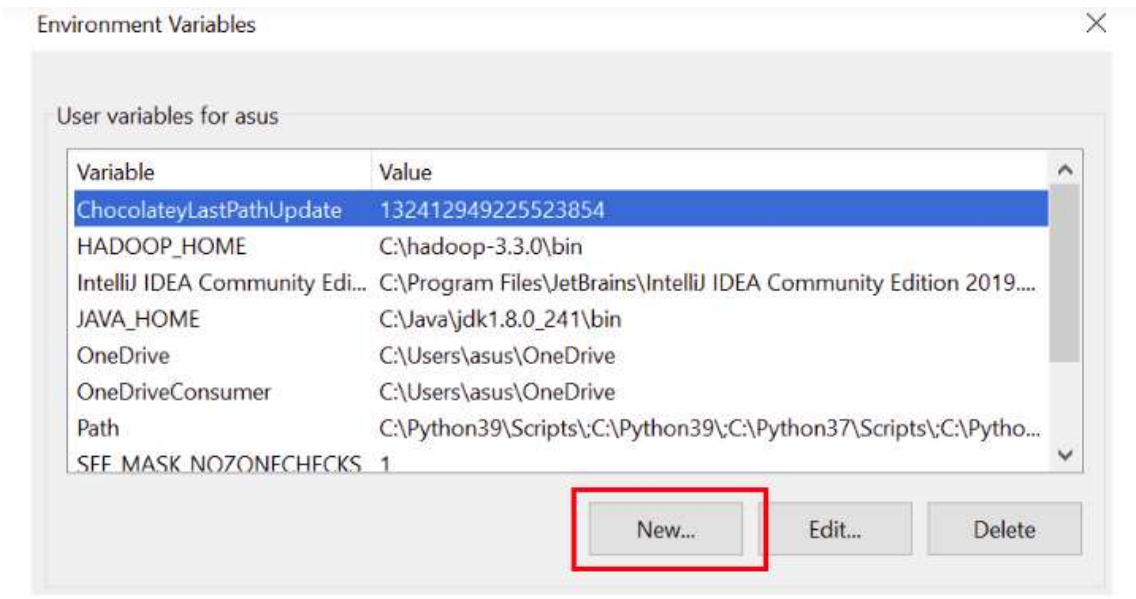
3.Set the path JAVA_HOME Environment variable

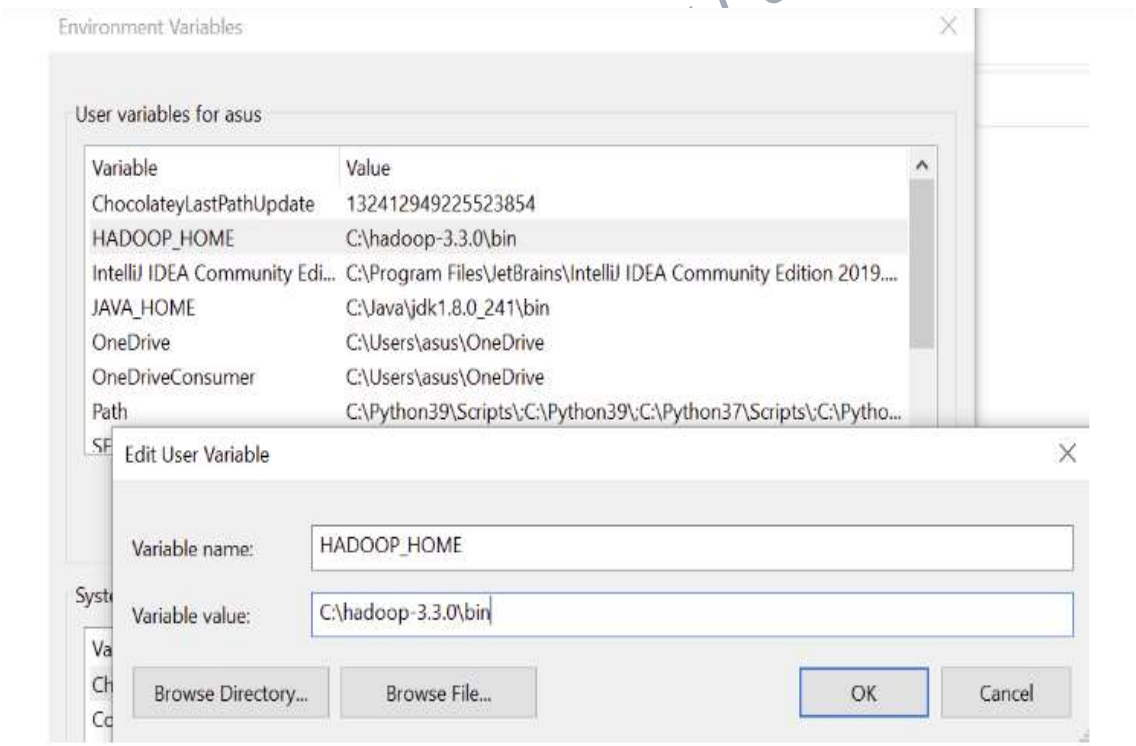
4.Set the path HADOOP_HOME Environment variable

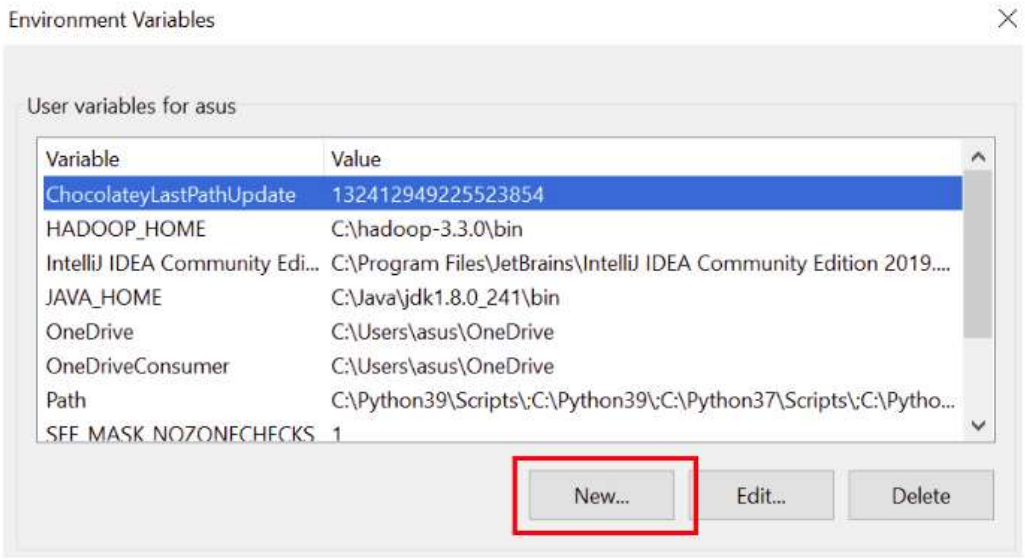


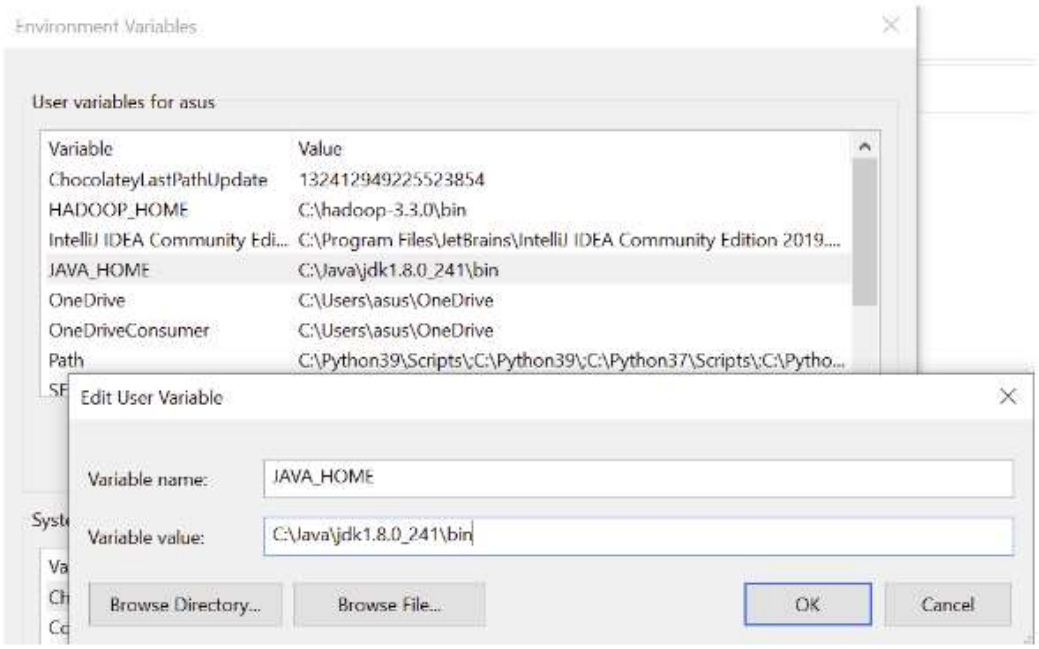

Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings
View basic information about your computer
Windows edition
Windows 10 Home Single Language
© 2019 Microsoft Corporation. All rights reserved.
System
Manufacturer: ASUSTek Computer Inc.
Processor: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
Installed memory (RAM): 8.00 GB (7.89 GB usable)
System type: 64-bit Operating System, x64-based processor
Pen and Touch: No Pen or Touch Input is available for this Display
ASUSTek Computer Inc. support
Website: Online support
Computer name, domain, and workgroup settings
Computer name: DESKTOP-475FCII
Full computer name: DESKTOP-475FCII
Computer description:
Workgroup: WORKGROUP

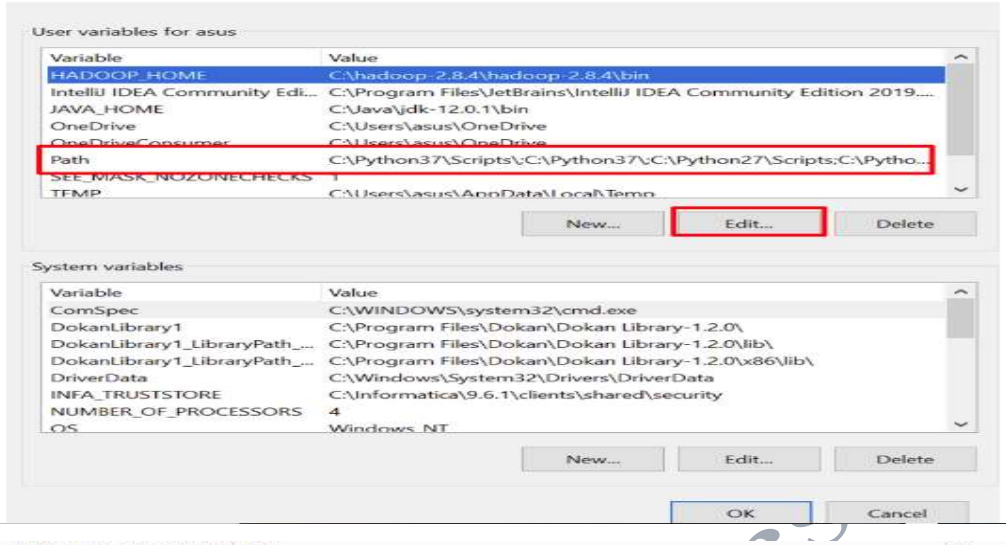

System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

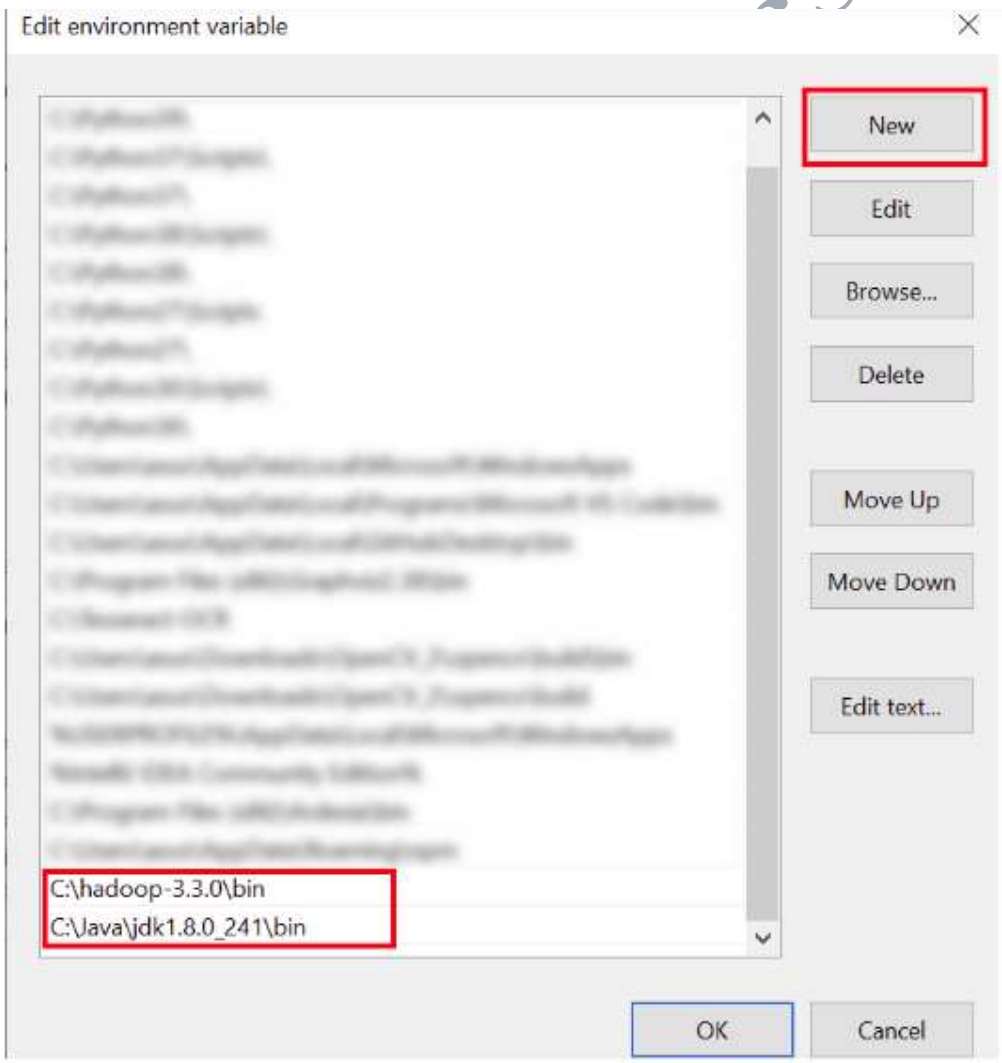

Environment Variables
User variables for asus
Variable
Value
ChocolateyLastPathUpdate
132412949225523854
HADOOP_HOME
C:\hadoop-3.3.0\bin
IntelliJ IDEA Community Edi...
C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019....
JAVA_HOME
C:\Java\jdk1.8.0_241\bin
OneDrive
C:\Users\asus\OneDrive
OneDriveConsumer
C:\Users\asus\OneDrive
Path
C:\Python39\Scripts\;C:\Python39\;C:\Python37\Scripts\;C:\Pytho...
New...
Edit...
Delete


Environment Variables
User variables for asus
Variable
Value
ChocolateyLastPathUpdate
132412949225523854
HADOOP_HOME
C:\hadoop-3.3.0\bin
IntelliJ IDEA Community Edi...
C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019....
JAVA_HOME
C:\Java\jdk1.8.0_241\bin
OneDrive
C:\Users\asus\OneDrive
OneDriveConsumer
C:\Users\asus\OneDrive
Path
C:\Python39\Scripts\;C:\Python39\;C:\Python37\Scripts\;C:\Pytho...
Edit User Variable
Variable name:
HADOOP_HOME
Variable value:
C:\hadoop-3.3.0\bin
Browse Directory...
Browse File...
OK
Cancel


Environment Variables
User variables for asus
Variable
Value
ChocolateyLastPathUpdate
132412949225523854
HADOOP_HOME
C:\hadoop-3.3.0\bin
IntelliJ IDEA Community Edi...
C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019....
JAVA_HOME
C:\Java\jdk1.8.0_241\bin
OneDrive
C:\Users\asus\OneDrive
OneDriveConsumer
C:\Users\asus\OneDrive
Path
C:\Python39\Scripts\;C:\Python39\;C:\Python37\Scripts\;C:\Pytho...
New...
Edit...
Delete


Environment Variables
User variables for asus
Variable
Value
ChocolateyLastPathUpdate
132412949225523854
HADOOP_HOME
C:\hadoop-3.3.0\bin
IntelliJ IDEA Community Edi...
C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019....
JAVA_HOME
C:\Java\jdk1.8.0_241\bin
OneDrive
C:\Users\asus\OneDrive
OneDriveConsumer
C:\Users\asus\OneDrive
Path
C:\Python39\Scripts\;C:\Python39\;C:\Python37\Scripts\;C:\Pytho...
Edit User Variable
Variable name:
JAVA_HOME
Variable value:
C:\Java\jdk1.8.0_241\bin
Browse Directory...
Browse File...
OK
Cancel


User variables for asus
Variable
Value
HADOOP_HOME
C:\hadoop-2.8.4\hadoop-2.8.4\bin
IntelliJ IDEA Community Edi...
C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019....
JAVA_HOME
C:\Java\jdk-12.0.1\bin
OneDrive
C:\Users\asus\OneDrive
Path
C:\Python37\Scripts\;C:\Python37\;C:\Python27\Scripts;C:\Pytho...
TEMP
C:\Users\asus\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\WINDOWS\system32\cmd.exe
DokanLibrary1
C:\Program Files\Dokan\Dokan Library-1.2.0\
DokanLibrary1_LibraryPath_...
C:\Program Files\Dokan\Dokan Library-1.2.0\lib\
DokanLibrary1_LibraryPath_...
C:\Program Files\Dokan\Dokan Library-1.2.0\x86\lib\
DriverData
C:\Windows\System32\Drivers\DriverData
INFA_TRUSTSTORE
C:\Informatica\9.6.1\clients\shared\security
NUMBER_OF_PROCESSORS
4
OS
Windows_NT
New...
Edit...
Delete
OK
Cancel


Edit environment variable
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
C:\hadoop-3.3.0\bin
C:\Java\jdk1.8.0_241\bin
OK
Cancel

5.Configurations :

- Edit file C:/Hadoop-3.3.0/etc/hadoop/core-site.xml,
- Paste the xml code in folder and save

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

- Rename "mapred-site.xml.template" to "mapred-site.xml" and edit this file C:/Hadoop-3.3.0/etc/hadoop/mapred-site.xml, paste xml code and save this file.

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

- Create folder "data" under "C:\Hadoop-3.3.0"
- Create folder "datanode" under "C:\Hadoop-3.3.0\data"
- Create folder "namenode" under "C:\Hadoop-3.3.0\data"
- Edit file C:\Hadoop-3.3.0/etc/hadoop/hdfs-site.xml,
- Paste xml code and save this file.

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/hadoop-3.3.0/data/namenode</value>
```

```
        </property>
        <property>
            <name>dfs.datanode.data.dir</name>
            <value>/hadoop-3.3.0/data/datanode</value>
        </property>
</configuration>
```

- Edit file C:/Hadoop-3.3.0/etc/hadoop/yarn-site.xml,
- Paste xml code and save this file.

```
<configuration>
        <property>
            <name>yarn.nodemanager.aux-services</name>
            <value>mapreduce_shuffle</value>
        </property>
        <property>
            <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
            <value>org.apache.hadoop.mapred.ShuffleHandler</value>
        </property>
 </configuration>
```

- Edit file C:/Hadoop-3.3.0/etc/hadoop/hadoop-env.cmd by closing the command line

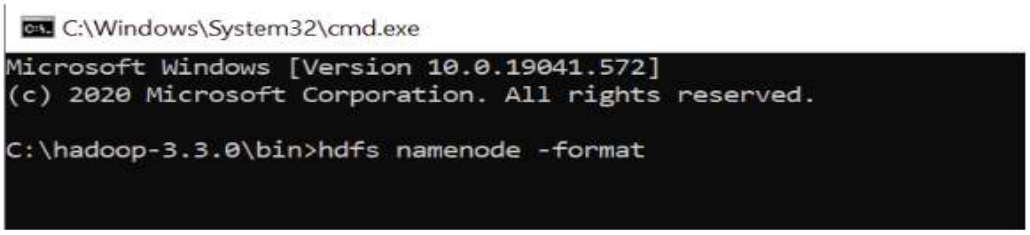
  "JAVA_HOME=%JAVA_HOME%" instead of set "JAVA_HOME=C:\Java"

6. Hadoop Configurations :

- Download

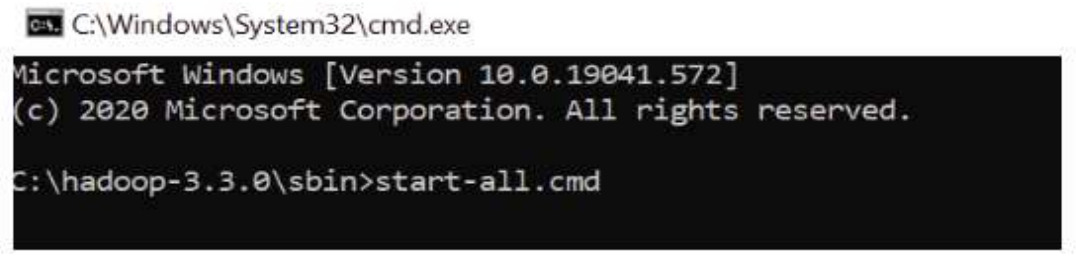
  https://github.com/brainmentorspvtltd/BigData_RDE/blob/master/Hadoop%20Configuration.zip or (for hadoop 3)

  https://github.com/s911415/apache-hadoop-3.1.0-winutils
- Copy folder bin and replace existing bin folder in **C:\Hadoop-3.3.0\bin**
- Format the NameNode
- Open cmd and type command **"hdfs namenode –format"**



7.Testing

- Open cmd and change directory to C:\Hadoop-3.3.0\sbin
- type **start-all.cmd**

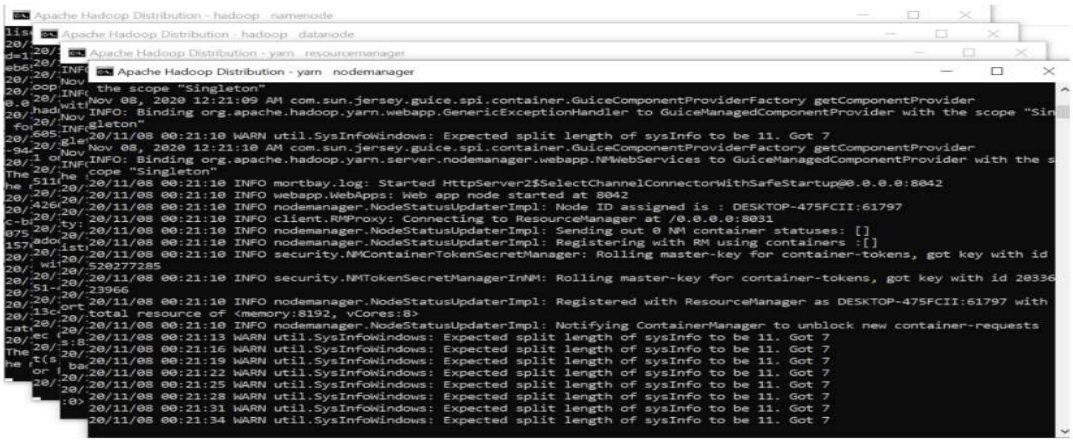


(Or you can start like this)

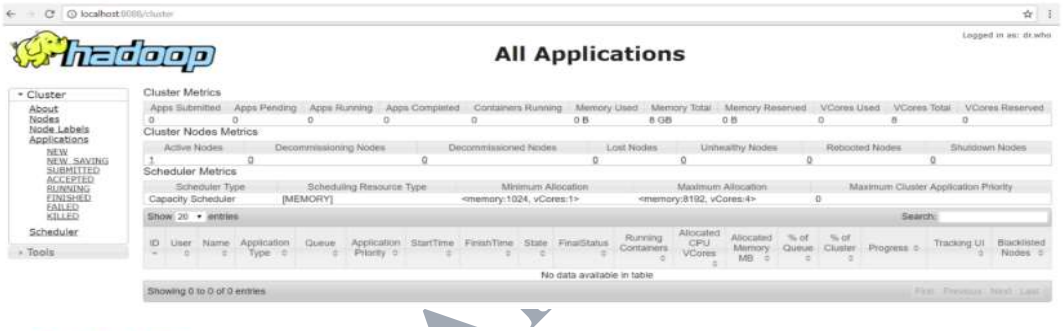
- Start namenode and datanode with this command
- type start-dfs.cmd
- Start yarn through this command
- type start-yarn.cmd

Make sure these apps are running

- Hadoop Namenode
- Hadoop datanode
- YARN Resource Manager
- YARN Node Manager



- Open : **http://localhost:8088**



Open: http://localhost:9870

RESULT: Installation of Hadoop is succesfully done.

Program: 2

AIM: To install Vmware.

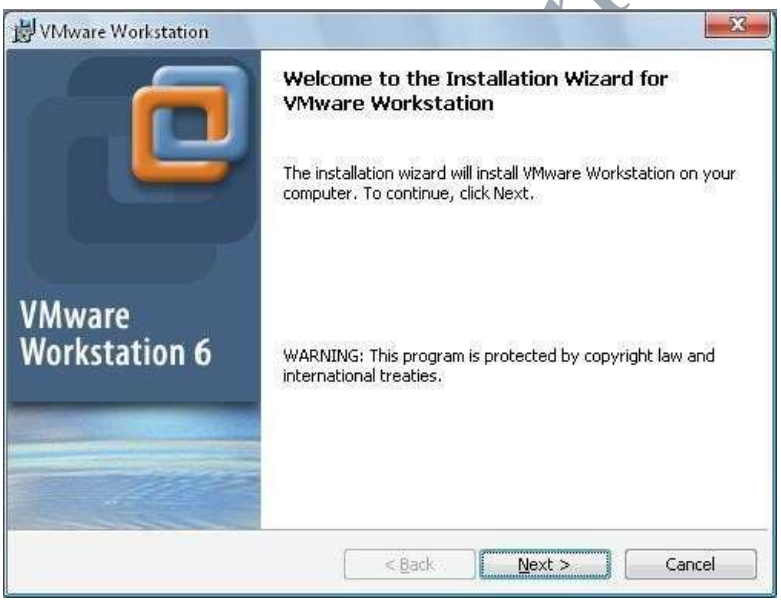
Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

STEP 1. First of all, enter to the official site of VMware and download VMware Workstation https://www.vmware.com/tryvmware/?p=workstation-w

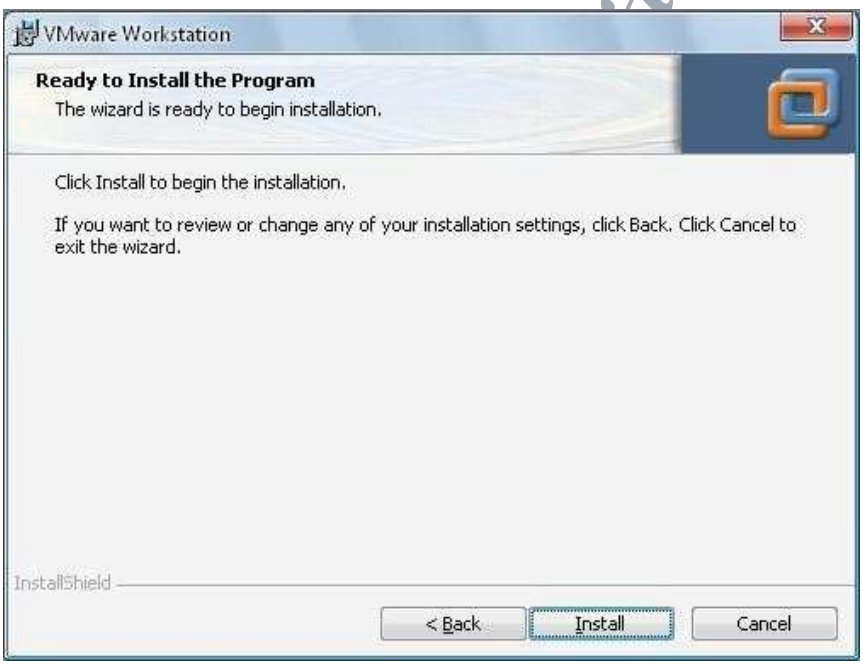
STEP 2. After downloading VMware workstation, install it on your PC

STEP 3. Setup will open Welcome Screen

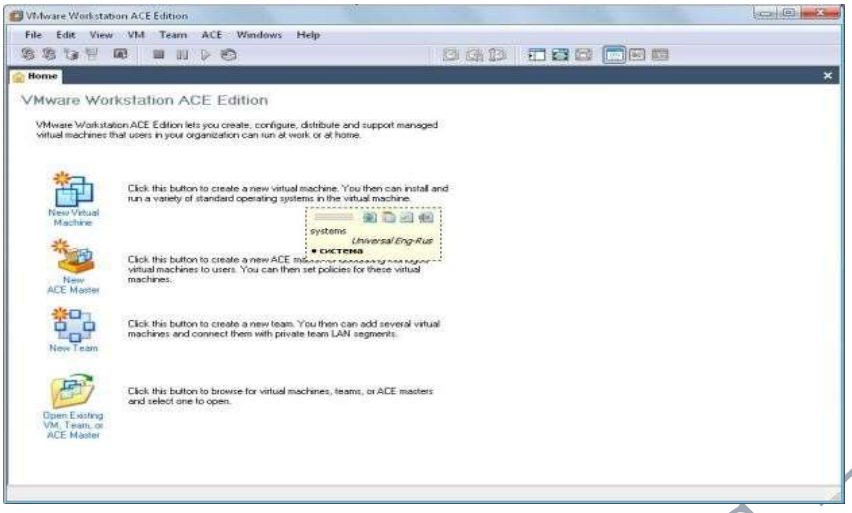


Click on **Next** button and choose **Typical** option

STEP 4. By clicking "**Next**" buttons, to begin the installation, click on **Install** button at the end

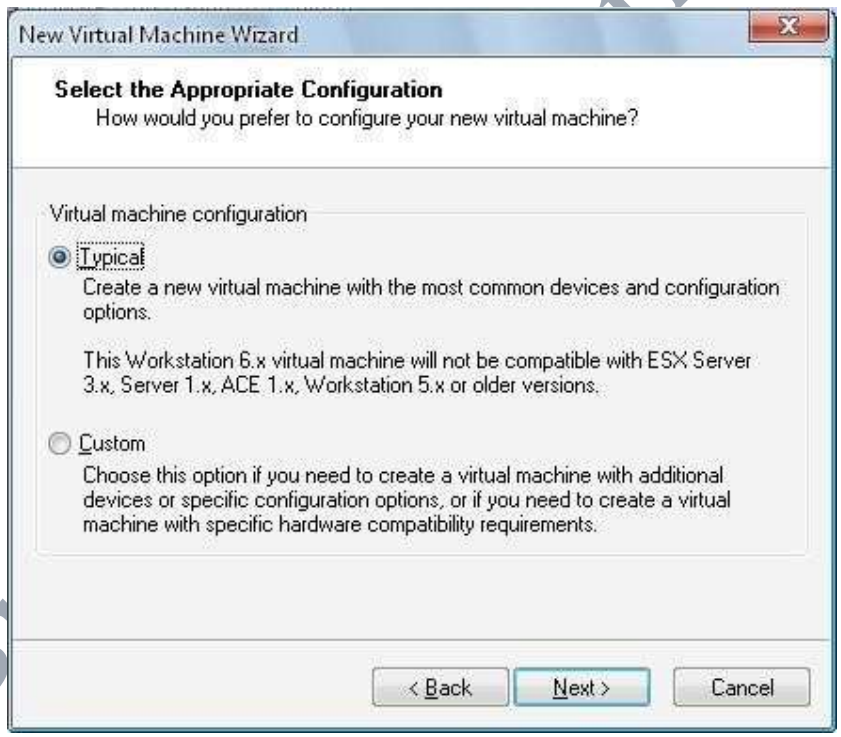


STEP 5. This will install VMware Workstation software on your PC, After installation complete, click on **Finish** button. Then restart your PC. Then open this software



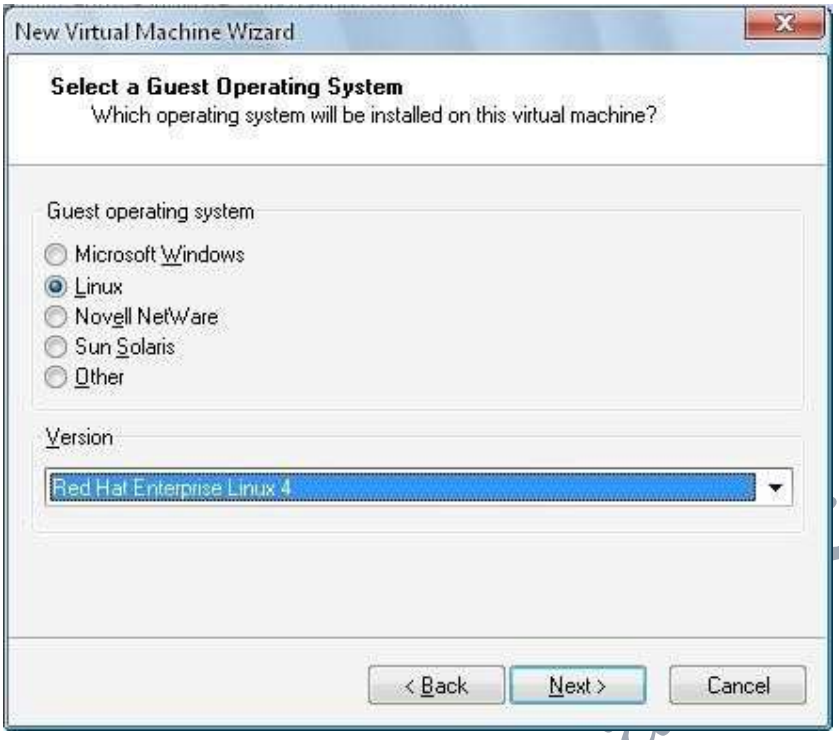
STEP 6: In this step we try to create new "virtual machine". Enter to File menu, then New-> Virtual Machine

Click on **Next** button, then check **Typical** option as below

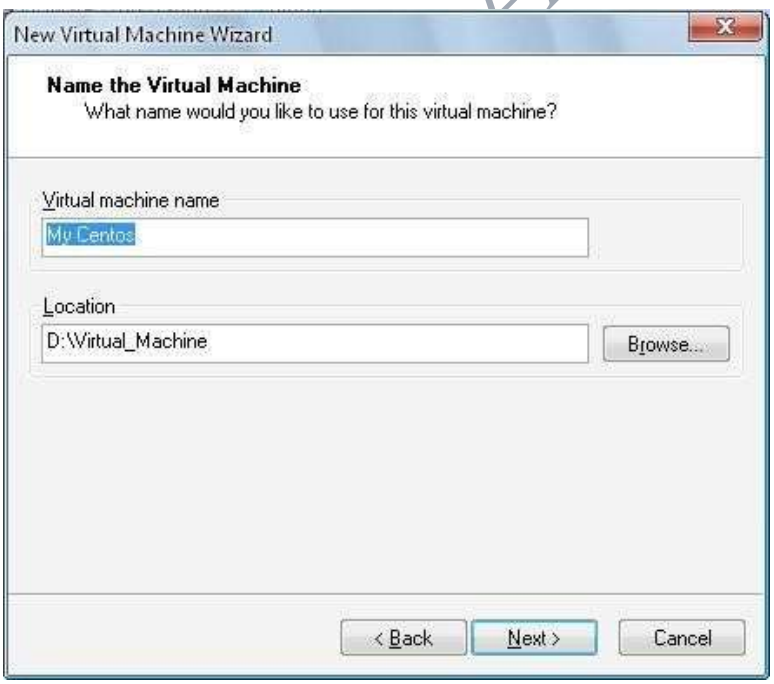


Then click **Next** button, and check your OS version. In this example, as we"re going to setup

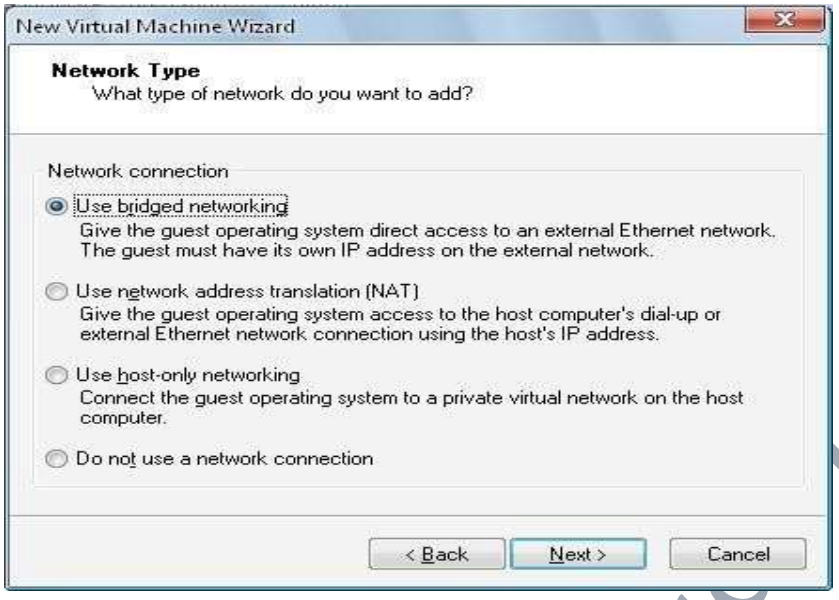
Oracle server on CentOS, we"ll check **Linux** option and from "*version*" option we"ll check **Red Hat Enterprise Linux 4**



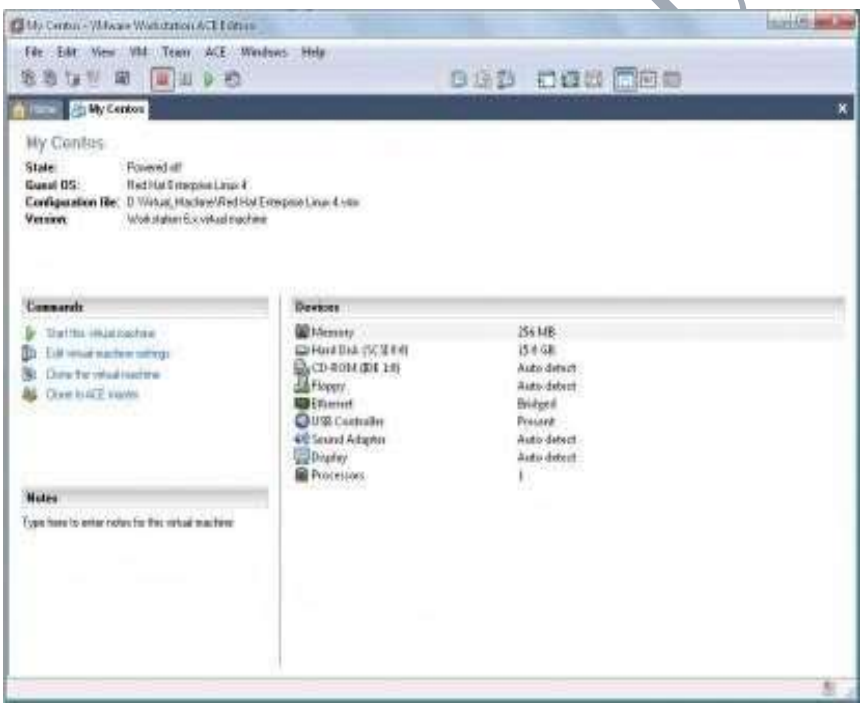
By clicking **Next** button, we"ll give a name to our virtual machine, and give directory to create this new virtual machine



Then select **Use bridged networking** option and click **Next**.

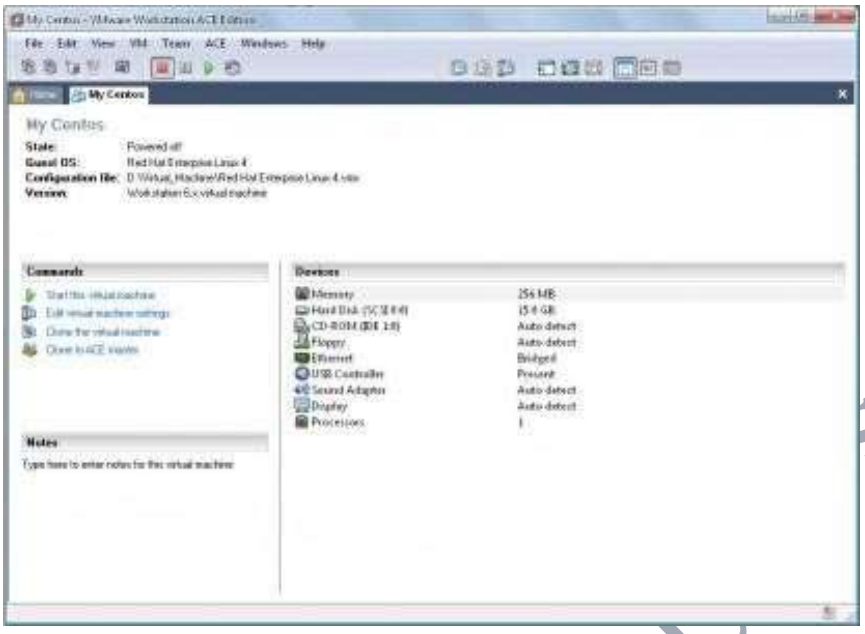


Then you"ve to define size of hard disk by entering its size. I"ll give 15 GB hard disk space and please check **Allocate all disk space now** option



Here, you can delete **Sound Adapter, Floppy and USB Controller** by entering "Edit virtual machine settings"**.** If you"re going to setup Oracle Server, please make sure you"ve increased your Memory (RAM) to 1GB.

OUTPUT:



RESULT: Installation of VMware is succesfully done.

Program: 3

AIM: To perform the file management tasks in Hadoop.

Name: ***Roll no:***

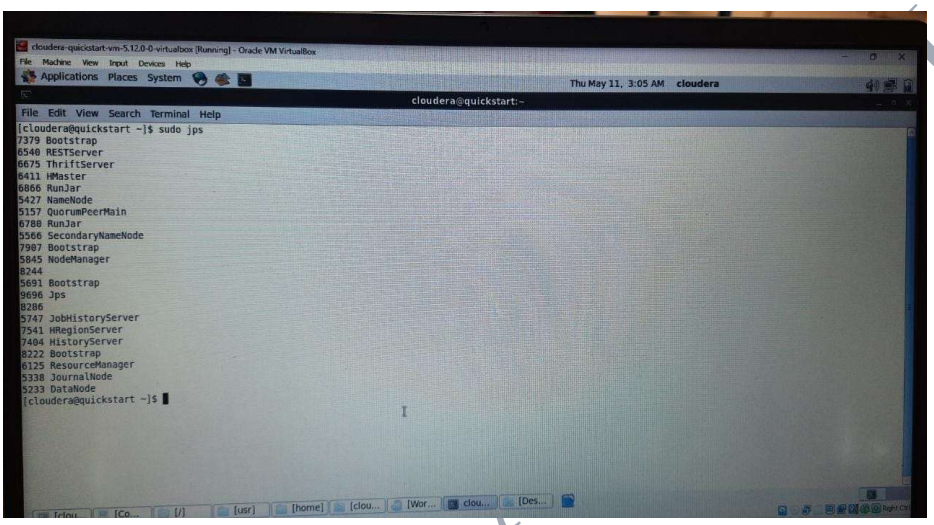
Branch & Sec: CSE - ***Date:***

Source Code:

1.To check if a specific daemon is up or not:

sudo jps

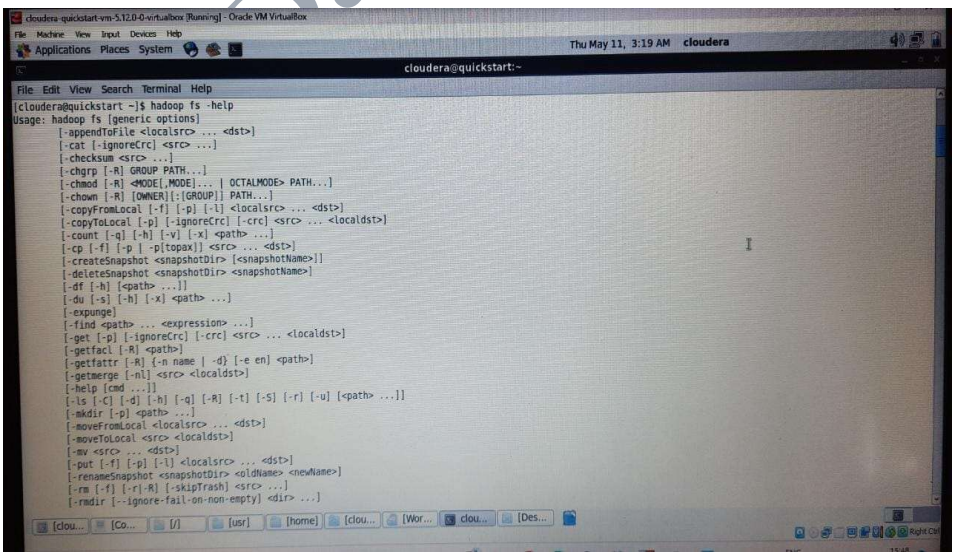
Output:



2.To see a list of all supported options:

hdfs dfs -help

Output:



3.To create a directory:

hdfs dfs -mkdir /user/cloudera/bda

4.To create a file:

hdfs dfs -mkdir /user/cloudera/sample1.txt

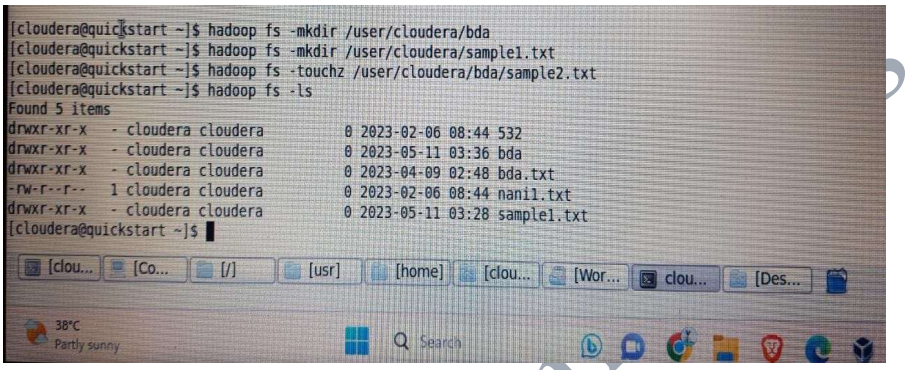
5.To create a file into directory:

hdfs dfs -touchz /user/cloudera/bda/sample2.txt

6.To see list of files/directories present:

hdfs dfs -ls

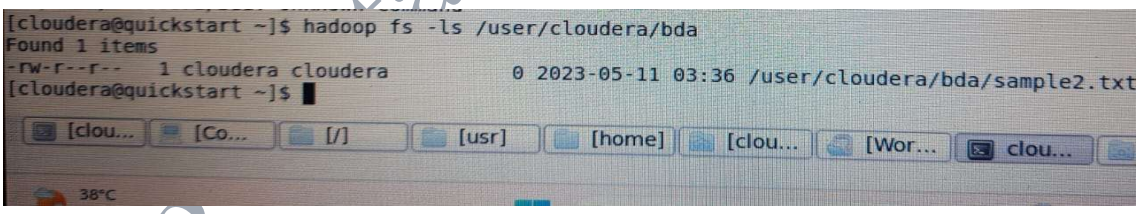
Output:



7.To see the files inside the directory:

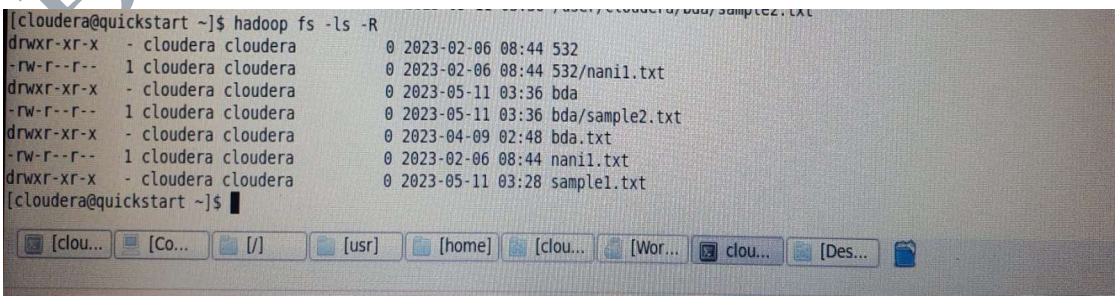
hdfs dfs -ls /user/cloudera/bda

Output:



8.To know the total files and directories:

hdfs dfs -ls -R

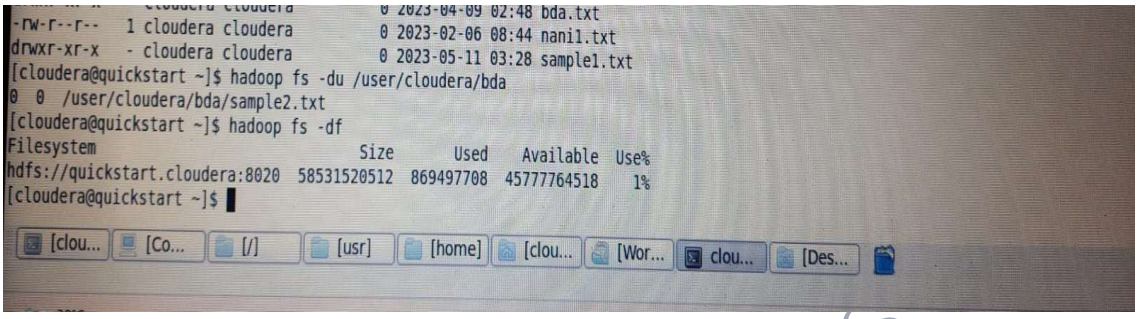


9.To know the disk usage of files under any directory:

hdfs dfs -du /user/cloudera/bda

10.To display disk usage of current hdfs:

hdfs dfs -df



11.To copy file from local fs to hadoop:

hdfs dfs -copyFromLocal /home/cloudera/Desktop/sample3.txt /user/cloudera/bda

12.To display the last few lines of file:

hdfs dfs -tail /user/cloudera/bda/sample1.txt

13.To remove files under a directory:

hdfs dfs -rm /user/cloudera/bda/sample2.txt

Output:

Deleted /user/cloudera/bda/sample2.txt

14.To clear all files we use:

hdfs dfs -expunge

15.To move one directory to another directory:

hdfs dfs -mv <source path> <destination path>

16.To move a file from local file system to hdfs file system

hdfs dfs -moveFromLocal <source path> <destination path>

17. To move a file from hdfs file system to local file system

hdfs dfs -moveToLocal <source path> <destination path>

RESULT: Hence the file management tasks in Hadoop are performed succesfully.

Program: 4

AIM: Implementation of Matrix Multiplication with Hadoop MapReduce.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

1.Create two files M1.txt ,M2.txt and put matrix values

2.To create M1.txt file we use following command:

cat >/home/cloudera/M1.txt

3.In M1.txt file insert values

1 2 3

4 5 6

4. To create M2.txt file we use following command:

cat >/home/cloudera/M2.txt

5. In M2.txt file insert values

7 8

9 10

11 12

6.Put the above files to HDFS location

7.To move the M1.txt and M2.txt files to HDFS create a matrix directory

hdfs dfs -mkdir /matrix

8.To move M1.txt to hadoop system we use put command

hdfs dfs -put /home/cloudera/M1.txt/matrix

9. To move M2.txt to hadoop system we use put command

hdfs dfs -put /home/cloudera/M2.txt/matrix

10.Now use command **"nano mapper.py"** to create mapper program

11.Press **"Ctrl+O"** to write the program

Mapper.py:

```
import sys
m_r=2
m_c=3
n_r=3
n_c=2
i=0
```

#Read each line i.e a row from stdin and split then to separate elements.

for line in sys.stdin:el=map(int,line.split())

The mapper will first read the first matrix and then the second. To differentiate them we can keep a count i of the line number we are reading and the first m_r lines will belong to the first matrix.

```
if(i<m_r):
        for j in range(len(el)):
                for k in range(n_c):
                        print "{0}\t{1}\t{2}\t{3}".format(i,k,j,el[j])
else:
        for j in range(len(el)):
                for k in range(m_r):
                        print "{0}\t{1}\t{2}\t{3}".format(k,j,i-m_r,el[j])
i=i+1
```

Now comes the crucial part, printing the key value. We need to think of a key which will group elements that need to be multiplied, elements that need to be summed and elements that belong to the same row.

[cloudera@quickstart ~]$ cat *.txt | python mapper.py

```
0	0	0	7
0	1	0	7
0	0	1	8
0	1	1	8
0	0	0	9
0	1	0	9
0	0	1	10
0	1	1	10
0	0	0	11
0	1	0	11
0	0	1	12
0	1	1	12
0	0	0	7
0	1	0	7
0	0	1	8
0	1	1	8
0	0	0	9
0	1	0	9
0	0	1	10
0	1	1	10
0	0	0	11
```

0	1	0	11
0	0	1	12
0	1	1	12
0	0	0	1
0	1	0	1
0	0	1	2
0	1	1	2
0	0	2	3
0	1	2	3
0	0	0	4
0	1	0	4
0	0	1	5
0	1	1	5
0	0	2	6
0	1	2	6

12.Save the program and then press Ctrl+X to exit the program

13. Now use command **"nano reducer.py"** to create reduce program

Reducer.py:

```
import sys
m_r=2
m_c=3
n_r=3
n_c=2
matrix=[]
for row in range(m_r):
        r=[]
        for col in range(n_c):
                s=0
                for el in range(m_c):
                        mul=1
                        for num in range(2):
                                line=sys.stdin.readline()
                                n=map(int,line.split('\t'))[-1]
                                mul*=n
                        s+=mul
                r.append(s)
        matrix.append(r)
print('\n'.join([str(x) for x in matrix]))
```

Now use the command **chmod 777 mapper.py reducer.py**

Upto noe we are in read and write mode only .To change it into execution mode we have to use that command.

14.Running the Map-Reduce Job on Hadoop

You can run the map reduce job and view the result by the following code (considering you have already put input files in HDFS)

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
> -input /user/cloudera/matrices/ \
> -output /user/cloudera/mat_output \
> -mapper ~/Desktop/mr/matrix-mul/Mapper.py \
> -reducer ~/Desktop/mr/matrix-mul/Reducer.py
```

15.Use the following command to see the output

hdfs dfs -cat /user/cloudera/mat_output/*

Above command should output the resultant matrix:

- **[14, 245]**
  [313, 77]

RESULT: Matrix Multiplication with Hadoop MapReduce is implemented successfully.

Program: 5

AIM: Write the Word Count program and describe step by step procedure.

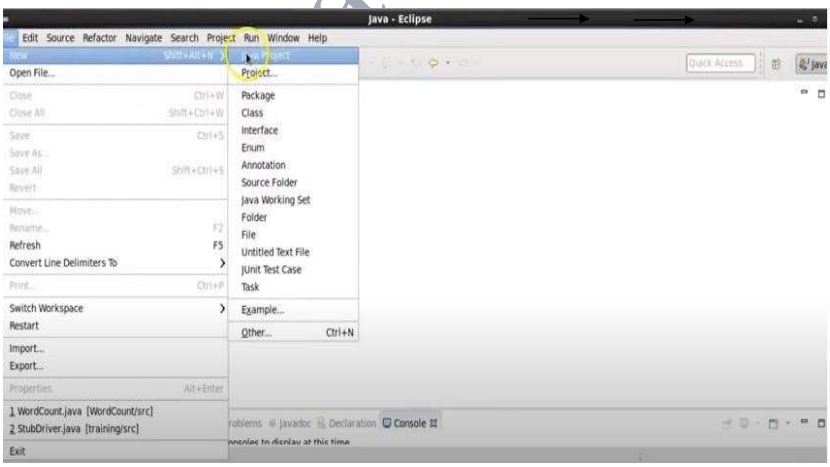
Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

PROCEDURE:

1.Cloudera desktop Eclipse Icon

2. In Eclipse window, File → New → Java Project

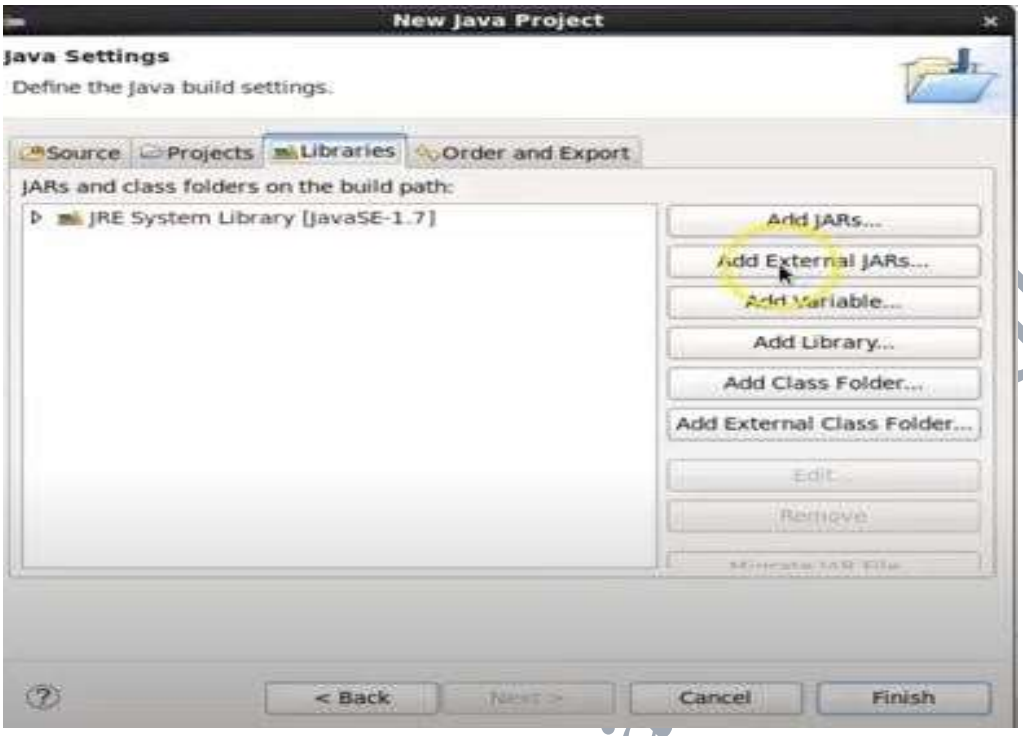


3.Give the Project name **as** WordCount

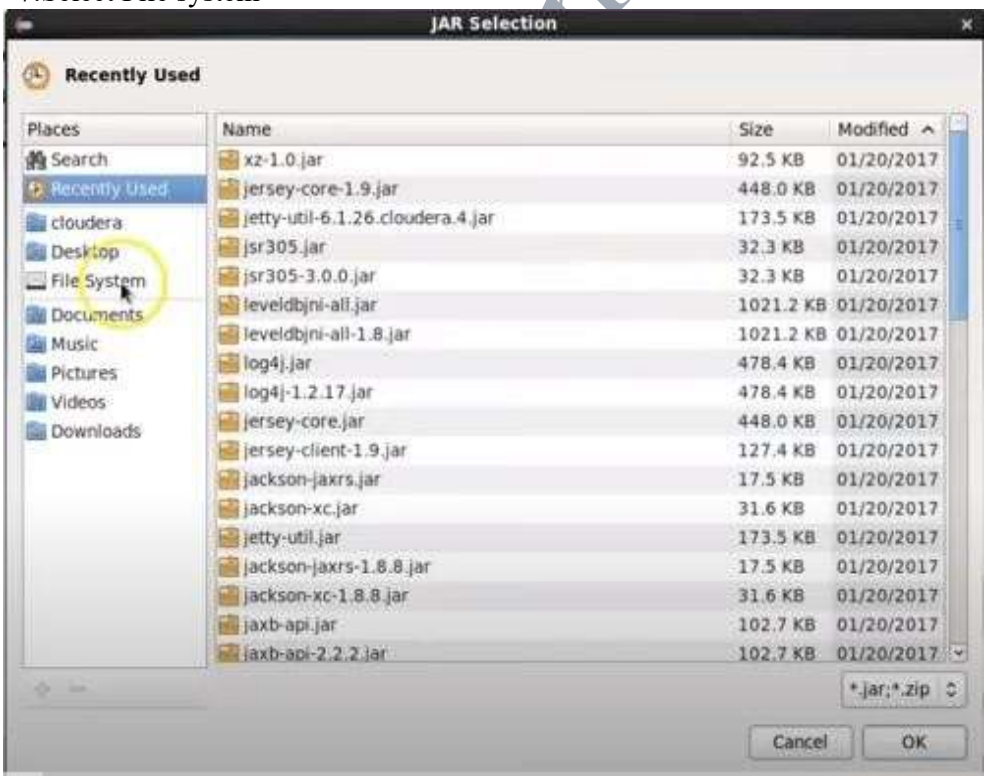
4.Click next button. Don't click finish button

5.Click Libraries

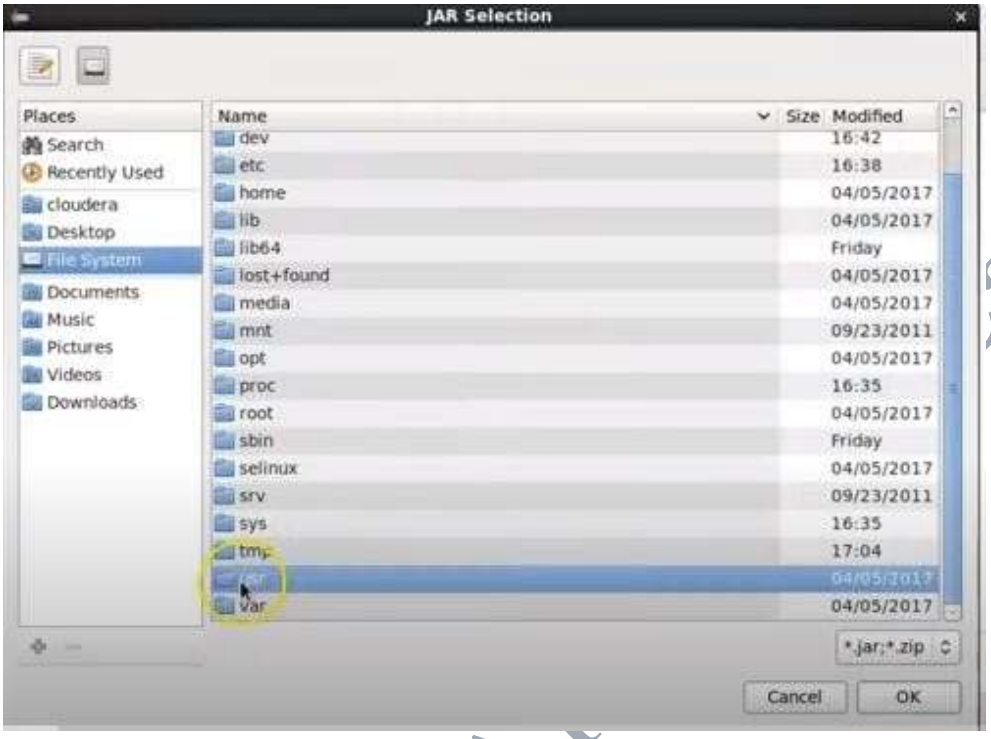
6.To import the libraries click on Add External JARs



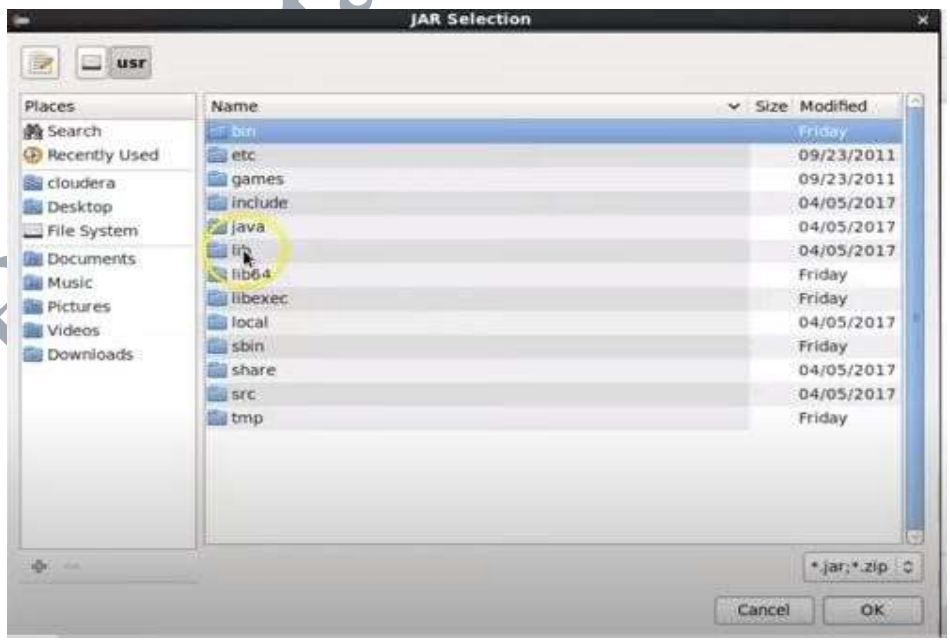
7.Select File system



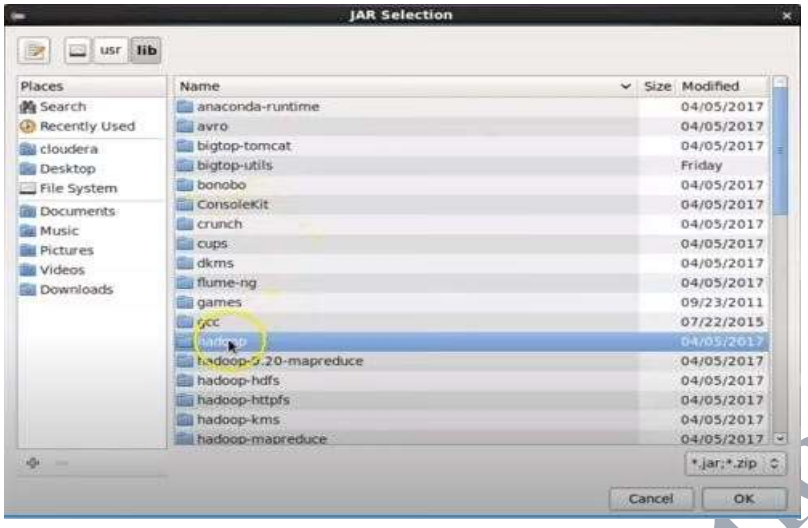
8.Select usr



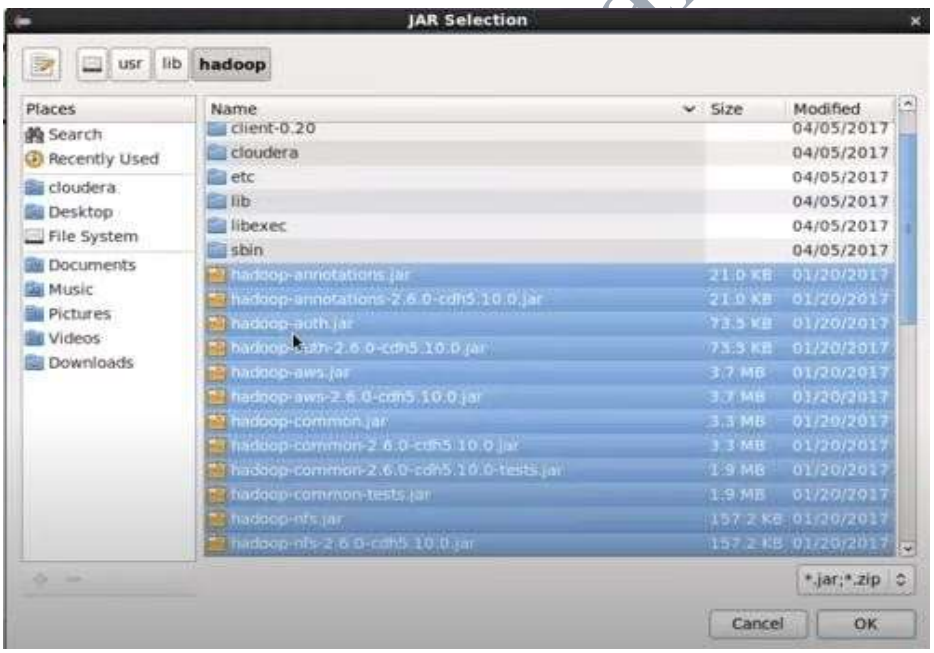
9. usr ⟶ Lib



10.Lib → Hadoop



11.Select all the jar files

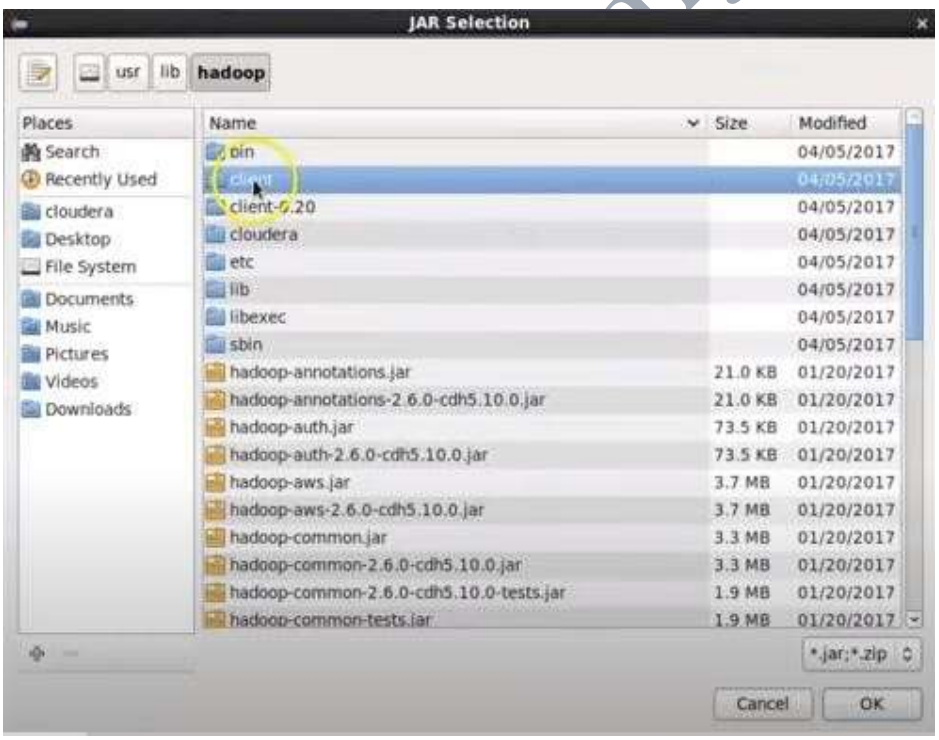


12.Click on OK BUTTON

13.All the jar files would be added to the libraries

14. Click again Add External Jars

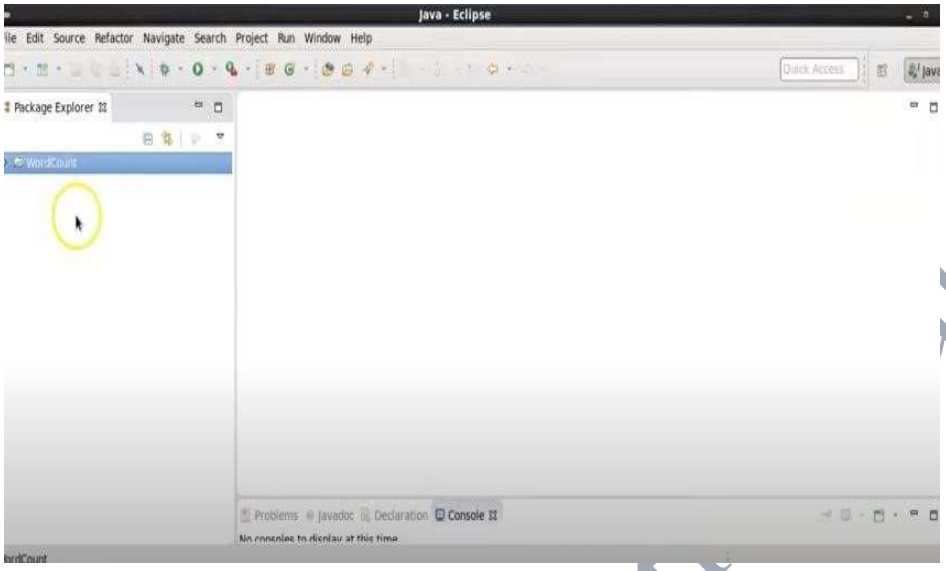
15. Click client



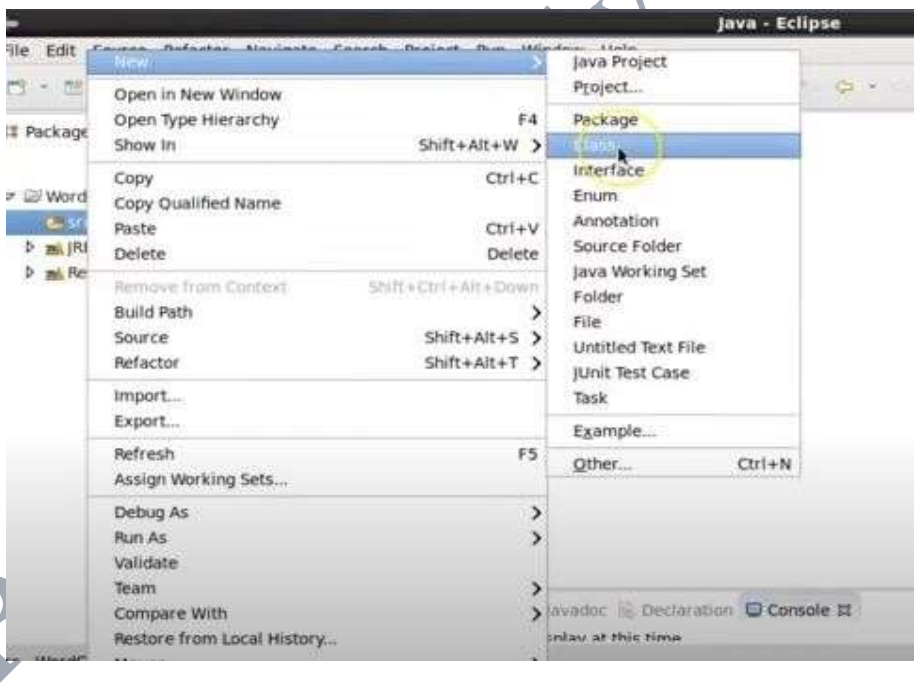
16. Select all the jar files and click on OK

17.Once all the JAR files are added to the libraries click on Finish

18.You can see file WordCount here

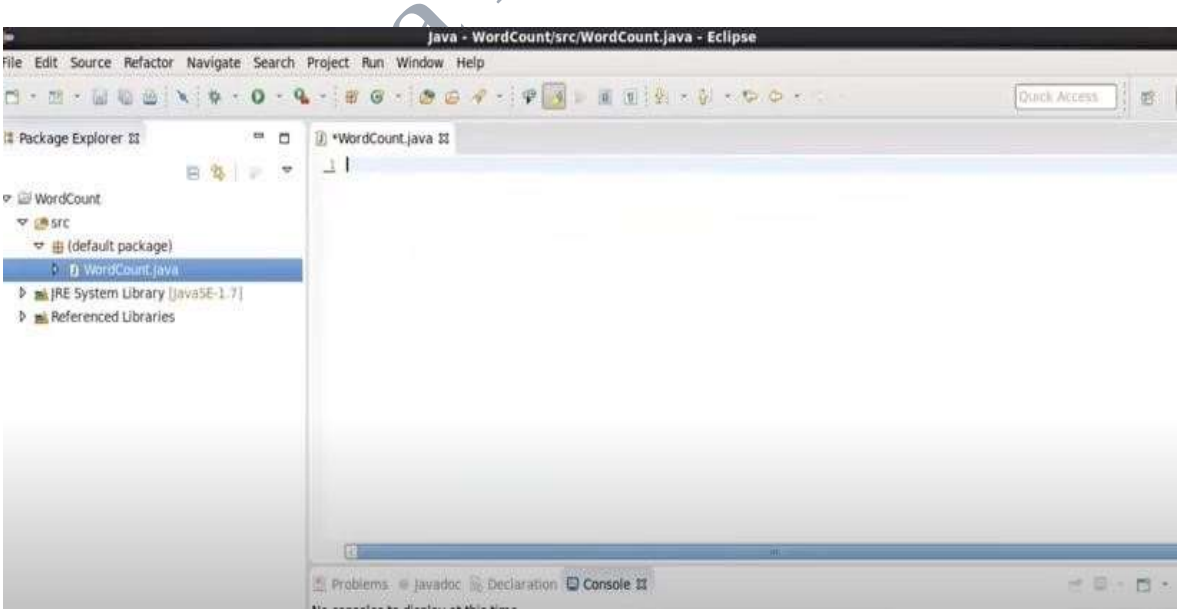


1 Click WordCount → src → new → class



20. In the name field type WordCount and click Finish

21.Enter the java program in the File WordCount.java



22.Write the code for Mapreduce client.

23. **Program:**

```
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()) {
      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 {
    int sum = 0;
```

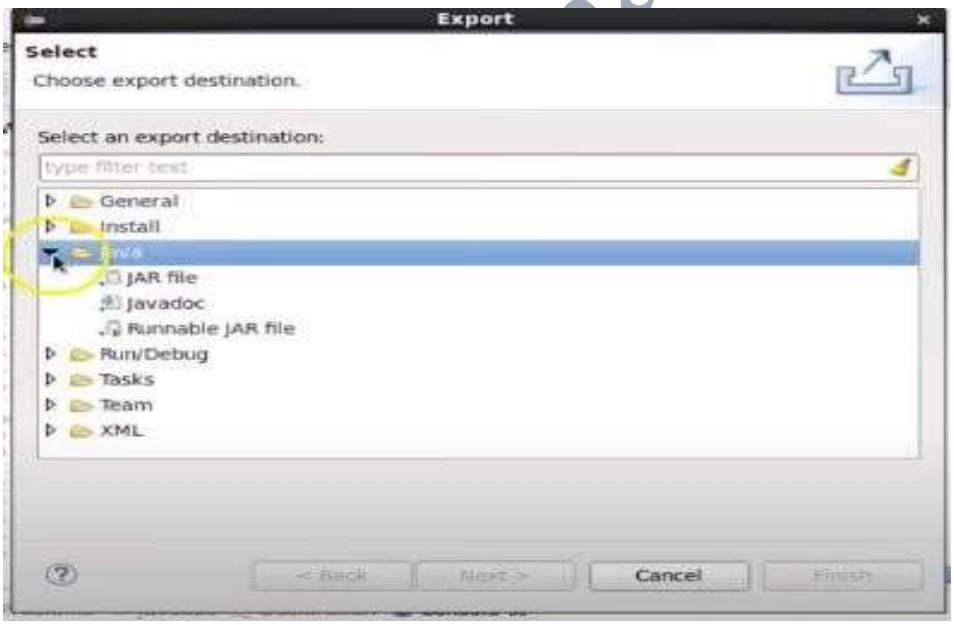
```
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
 }
 public static void main(String[] args) throws Exception {
   Configuration conf = new Configuration();
   Job job = Job.getInstance(conf, "word count");
   job.setJarByClass(WordCount.class);
   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);
  }
}
```


24.In the console window check any errors (Errors will be indicated with red mark)

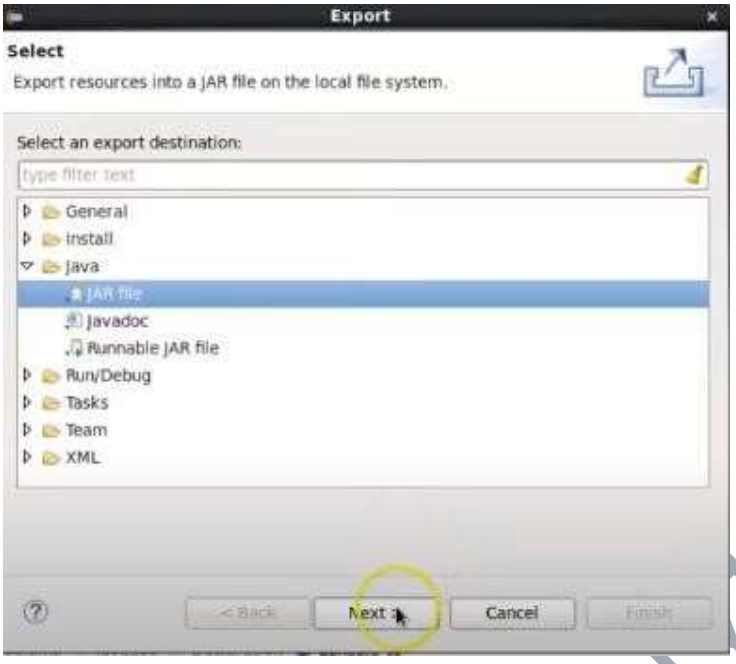
25. Next step export the JAR files

Right click on wordcount ⟶ export

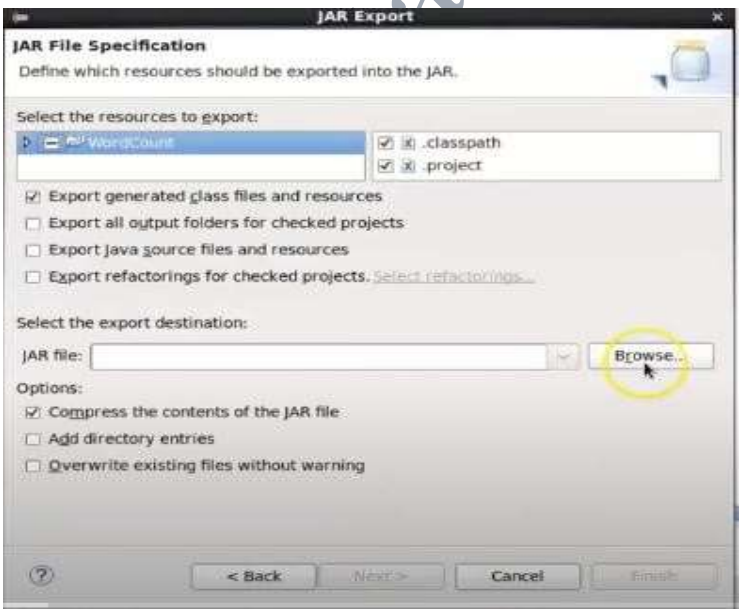
26. Select Java



27. Select JAR file and click next

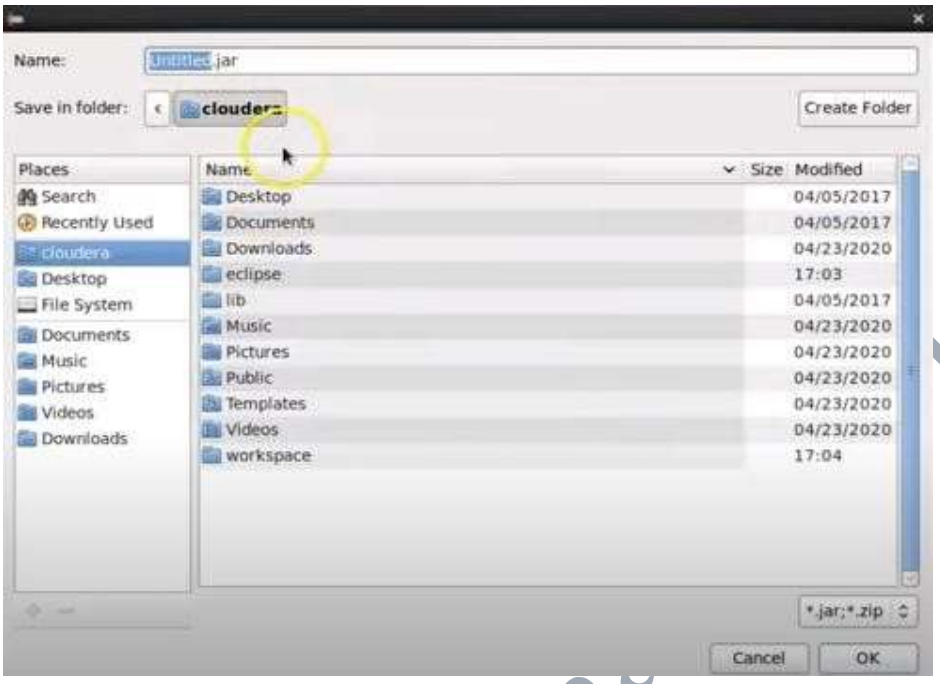


28.Browse the JAR File

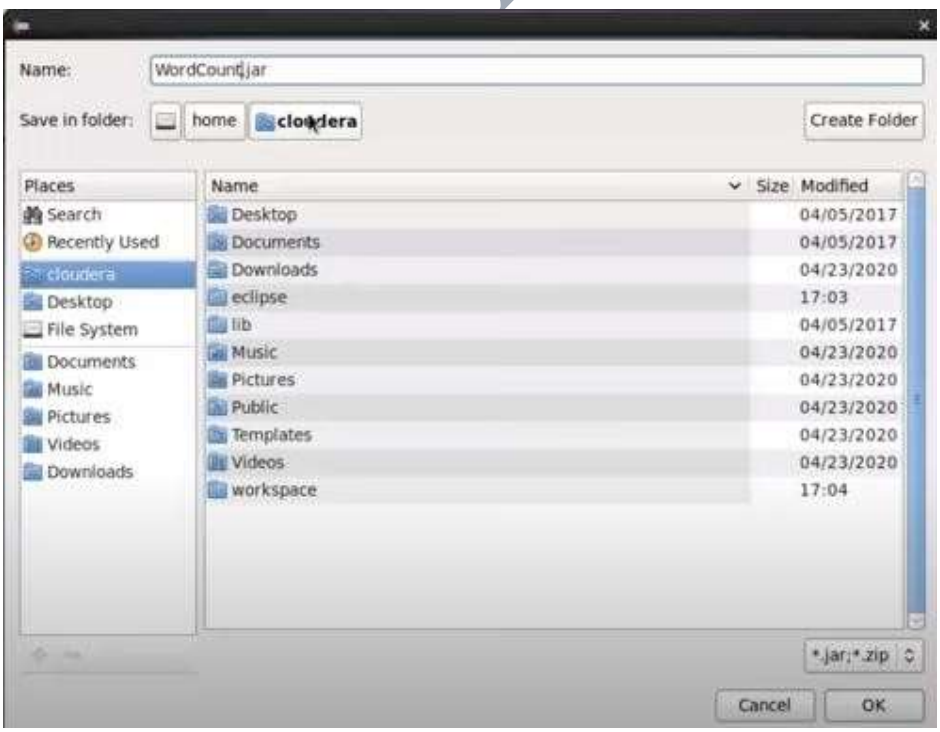


29. Select Cloudera

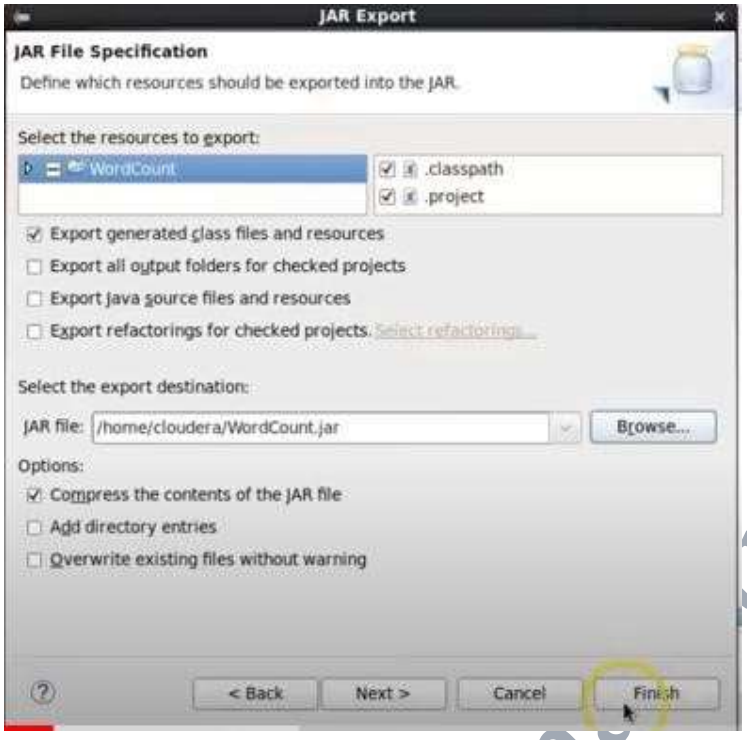
You can find untitled.jar at the top



30.Rename it as WordCount.



31.Click ok and then click finish



32. We will check in the filesystem whether WordCount.jar is exported successfully or not

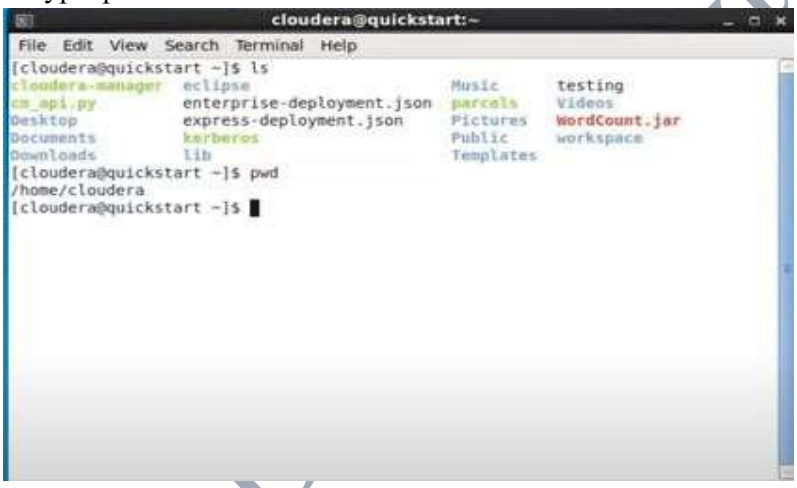
In desktop click Computer → File system → home → cloudera → wordcount.jar

33.Once the jar file is verified open terminal

Type ls

34.Check the working directory

Type pwd



35.Create a file cat > / home/cloudera/Processfile1.txt

This
is
BDA
Lab
BDA
is
Hadoop
File
This
Lab

Press Ctrl+Z to stop entering

36. To verify the contents present in Processfile1.txt

cat /home/cloudera/Processfile1.txt

37.Move the file Processfile1.txt to hdfs

38.To check whether hdfs is working

Hdfs dfs –ls

39.To see the list of directories in hdfs

Hdfs dfs –ls /

40.Create a directory:

hdfs dfs –mkdir /inputfolder1

41.move the file (Processfile1.txt) into hadoop system:

hdfs dfs –put /home/cloudera/Processfile1.txt /inputfolder1/

42.Check whether the file is copied into the hdfs and to display the contents of t the file: **hdfs dfs –cat /inputfolder1/Processfile1.txt**

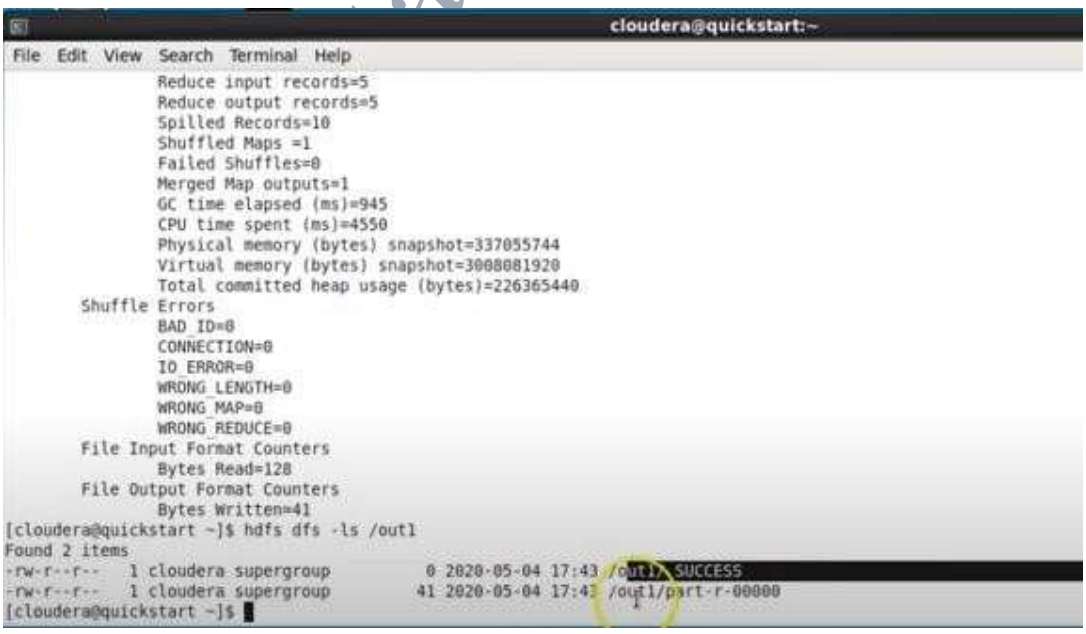
43.Open new terminal

Hadoop jar /home/cloudera/WordCount.jar WordCount /inputfolder1/Processfile1.txt /out1

In the output u can see the no of splits, map tasks, reduce tasks, map output records

44 .To see the output file of the mapreduce in out1 directory:

hdfs dfs –ls /out1



45.To see the output:

hdfs dfs –cat /out1/part-r-00000

46. Alternative way to see the output in the Admin Console

(i) localhost:50070

(ii) Select Utilities and select Browse file system and click output folder directory out1 and click on file part-r-00000 and download the file and save.

OUTPUT:

BDA	2
File	1
Hadoop	1
Lab	2
This	2
is	2

RESULT: Step by step procedure of the Word Count program is described and implemented successfully.

Program: 6

AIM: To write MapReduce program for Weather dataset and describe step by step procedure.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

Steps for MapReduce Program for Weather dataset

1. Open Eclipse
2. Erase the old program and close the program windows present.
3. File → New → Java Project
   Give the project name as **MaxTemp**
4. Click on Next Button
5. Click on Libraries and select Add External Jars
6. File system → usr → Lib → hadoop . (select all the jar files and click on OK Button)
7. Once again in Libraries Tab select **Add external Jar**
8. Click client and select all JAR files. Click on OK and click on Finish Button
9. Select Max Temp from left panel u can see three dropdowns
10. Creating **Mapper File**
    Right click on src.
    New → class
    Name field : **MaxTempMapper**
    Click on Finish
11. Creating **Reducer File**
    Right click on src.
    New → class
    Name field : **MaxTempReducer**
    Click on Finish
12. Creating **driver File**
    Right click on src.
    New → class
    Name field : **MaxTemp**

Click on Finish

13. Copy the mapper, reducer and driver code into the java files
14. Change the class name of mapper, reducer and reducer and change the conf name to **MaxTemp**
15. Change the **arg value** of input and output path
16. Right click on **MaxTemp** in the left panel select **export**
17. Select java → JAR file.
18. Click on **Browse button**
19. Select Desktop. Give the name as **MaxTemp.jar**
20. Click on **OK** Button. *U can see the jar file created in the desktop*
21. Keep the dataset in the desktop tempinput.txt
22. Open terminal

    hadoop dfs –mkdir /maxoutput
23. We have to go to desktop as we have dataset

    cd Desktop
24. Copy the dataset to hadoop

    Hadoop dfs –copyFromLocal tempinput.txt /maxinput
25. Hadoop jar MaxTemp.jar MaxTemp /maxinput /max output
26. To see the output
    hadoop dfs -cat /maxoutput/part-r-00000
27. We can also see the output from the browser
    Localhost:50070

    In utilities tab select browse file system and we can see the output name Maxoutput click on it and download.

Programs:

MaxTempMapper:

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Mapper;
public class MaxTempMapper extends Mapper<LongWritable, Text, Text, IntWritable > {
```

```
        public void map(LongWritable key, Text value, Context context)
                        throws IOException, InterruptedException {
String line=value.toString();
String year=line.substring(15,19);
int airtemp;
if(line.charAt(87)== '+') {
         airtemp=Integer.parseInt(line.substring(88,92));
} else
                airtemp=Integer.parseInt(line.substring(87,92));
                String q=line.substring(92,93);
                if(airtemp!=9999&&q.matches("[01459]"))
                {
                context.write(new Text(year),new IntWritable(airtemp));
                }
        }
}
```

MaxTempReducer:

```
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;
public class MaxTempReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context)
                        throws IOException, InterruptedException {
                int maxvalue=Integer.MIN_VALUE;
                for (IntWritable value : values) {
maxvalue=Math.max(maxvalue, value.get());
                }
                context.write(key, new IntWritable(maxvalue));
        }
}
```

MaxTempDriver:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class MaxTemp {
        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                Job job = Job.getInstance(conf, "MaxTemp");
                job.setJarByClass(MaxTemp.class);
                // TODO: specify a mapper
                job.setMapperClass(MaxTempMapper.class);
                // TODO: specify a reducer
                job.setReducerClass(MaxTempReducer.class);
               // TODO: specify output types
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(IntWritable.class);
               // TODO: specify input and output DIRECTORIES (not files)
                FileInputFormat.setInputPaths(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));
                if (!job.waitForCompletion(true))
                        return;
        }
}
```

OUTPUT:

```
cloudera@quickstart ~]$ hadoop dfs -mkdir /maxinput
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
 [cloudera@quickstart ~]$ cd Desktop
[cloudera@quickstart Desktop]$ cd Desktop/
```

```
bash: cd: Desktop/: No such file or directory
[cloudera@quickstart Desktop]$ hadoop dfs -copyFromLocal tempinput.txt /maxinput
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

[cloudera@quickstart Desktop]$ hadoop jar MaxTemp.jar MaxTemp /maxinput /maxoutput
23/04/24 22:07:42 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
23/04/24 22:07:46 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
23/04/24 22:07:48 INFO input.FileInputFormat: Total input paths to process : 1
23/04/24 22:07:49 INFO mapreduce.JobSubmitter: number of splits:1
23/04/24 22:07:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1682395925952_0001
23/04/24 22:07:52 INFO impl.YarnClientImpl: Submitted application application_1682395925952_0001
23/04/24 22:07:52 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1682395925952_0001/
23/04/24 22:07:52 INFO mapreduce.Job: Running job: job_1682395925952_0001
23/04/24 22:08:50 INFO mapreduce.Job: Job job_1682395925952_0001 running in uber mode : false
23/04/24 22:08:50 INFO mapreduce.Job:  map 0% reduce 0%
23/04/24 22:09:15 INFO mapreduce.Job:  map 100% reduce 0%
23/04/24 22:09:36 INFO mapreduce.Job:  map 100% reduce 100%
23/04/24 22:09:37 INFO mapreduce.Job: Job job_1682395925952_0001 completed successfully
23/04/24 22:09:38 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=61
                FILE: Number of bytes written=286393
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=635
                HDFS: Number of bytes written=17
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=21659
                Total time spent by all reduces in occupied slots (ms)=19101
                Total time spent by all map tasks (ms)=21659
                Total time spent by all reduce tasks (ms)=19101
```

```
                Total vcore-milliseconds taken by all map tasks=21659
                Total vcore-milliseconds taken by all reduce tasks=19101
                Total megabyte-milliseconds taken by all map tasks=22178816
                Total megabyte-milliseconds taken by all reduce tasks=19559424
        Map-Reduce Framework
                Map input records=5
                Map output records=5
                Map output bytes=45
                Map output materialized bytes=61
                Input split bytes=105
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=61
                Reduce input records=5
                Reduce output records=2
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=497
                CPU time spent (ms)=3040
                Physical memory (bytes) snapshot=347107328
                Virtual memory (bytes) snapshot=3015774208
                Total committed heap usage (bytes)=226365440
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=530
        File Output Format Counters
                Bytes Written=17
[cloudera@quickstart Desktop]$ hadoop dfs -cat /maxoutput/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

1949    111

1950    22
```

RESULT: MapReduce program for Weather dataset is done successfully.

Program: 7

AIM: Warm Up Linux Exercise on Linux.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

Linux Commands and their purposes

Commands	Purposes
Pwd	Print working directory, In other words 'where I am ?'
Ls	List files It lists everything in the current working directory
Cd	Change Directory, In other words change my current location
mkdir	Create a directory
Cp	To copy directories and files
Rm	Remove a file or directory
Cat	To create a file To display contents of file
Gedit	To create a file

1.To check the present working directory:

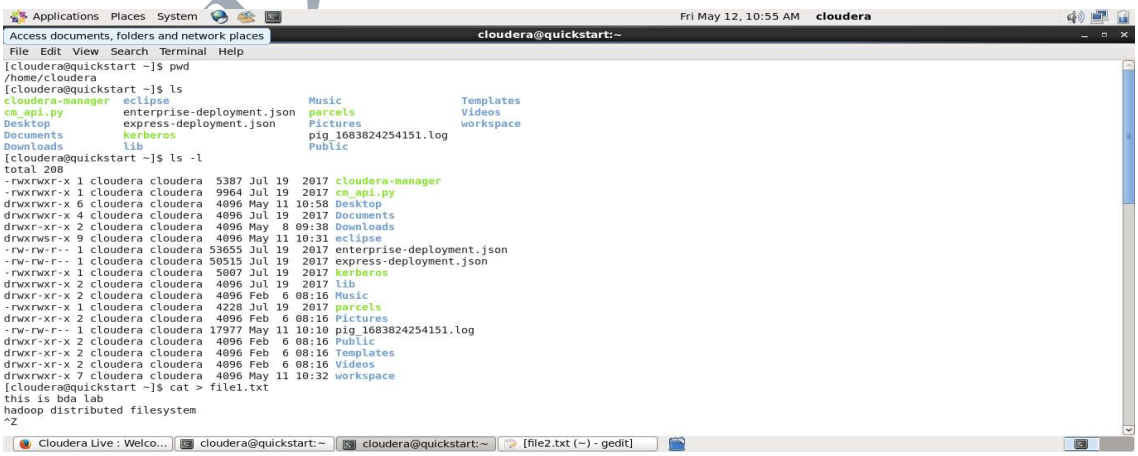
Syntax: **pwd**

2.To list the files and directories on the desktop:

Syntax: **ls**

3.To list all the details of files and directories:

Syntax: **ls -l**



4.To create a file on desktop using cat command:

Syntax: **cat** > *file_name*

cat > file1.txt

Enter the data. To stop data entering press Ctrl+Z

5.To display contents of the file1.txt using cat command:

Syntax: **cat** *file_name*

cat file1.txt

6.To create a file using gedit command:

Syntax**: gedit** *file_name*

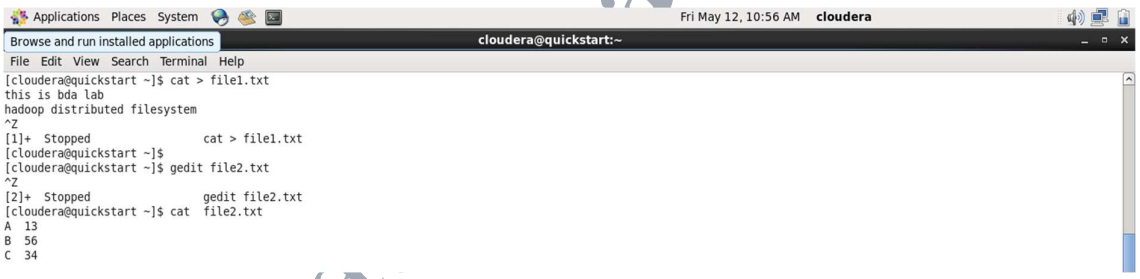
gedit file2.txt

Now a file named file2 will be opened. Enter the data you need ,save the file, close the file.

To stop data entering press *Ctrl+Z.*

7. To display contents of the file2.txt using cat command:

cat file2.txt



8.To make a directory:

Syntax: **mkdir** *directory_name*

mkdir CSE

9.To copy a file into directory:

Syntax: **cp** *file_name directory_name*

cp file1.txt CSE

cp file2.txt CSE

10.To list the files in the directory:

Syntax: **ls** *directory_name*

ls CSE

```
[cloudera@quickstart ~]$ mkdir CSE
[cloudera@quickstart ~]$ cp file1.txt CSE
[cloudera@quickstart ~]$ ls CSE
file1.txt
[cloudera@quickstart ~]$ cp file2.txt CSE
[cloudera@quickstart ~]$ ls CSE
file1.txt  file2.txt
```

11.To remove file in a directory:

Syntax: **rm** *file_name /directory_name*

rm file1.txt /CSE

12.To move existing file into directory:

Syntax: **mv** *file_name /directory_name*

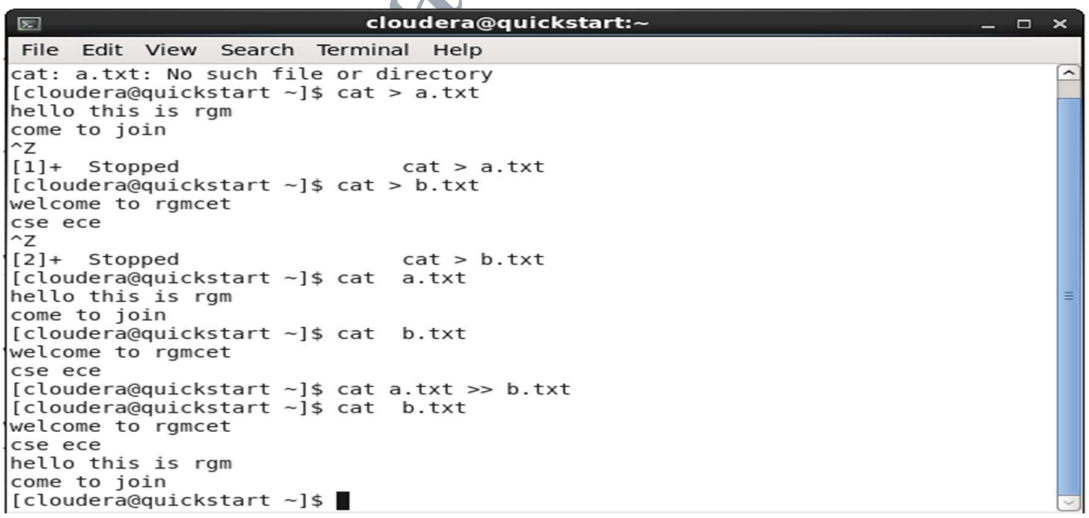
mv file3.txt CSE

```
[cloudera@quickstart ~]$ cat >file3.txt
hadoop ecosystem
hive pig yarn
^Z
[3]+  Stopped                 cat > file3.txt
[cloudera@quickstart ~]$ mv file3.txt CSE
[cloudera@quickstart ~]$ ls CSE
file1.txt  file2.txt  file3.txt
```

13.To combine two files:

Syntax: **cat** *file_name1* **>>** *file_name2*

cat a.txt >> b.txt



RESULT: Warm Up Linux Exercise on Linux is done successfully.

Program: 8

AIM: Write the HDFS (Hadoop Distributed File System) Commands.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

HDFS COMMANDS

Q1) What is HDFS?

HDFS stands for "Hadoop Distributed File System" . HDFS commands are used to access the Hadoop File System. The HDFS is a sub-project of the Apache Hadoop project. This Apache Software Foundation project is designed to provide a fault-tolerant file system designed to run on commodity hardware. HDFS is accessed through a set of shell commands. For executing the HDFS commands , open the terminal.

Q2) How to check the version of Hadoop framework?

Command: **hadoop version**

[cloudera@quickstart Desktop]$ hadoop version

Hadoop 2.6.0-cdh5.10.0

Q3) List the files and subdirectories present in HDFS

Command**: hdfs dfs –ls**

This command will list all the available files and subdirectories under default directory. For instance, in our example the default directory for Cloudera VM is /user/cloudera

[cloudera@quickstart Desktop]$ hdfs dfs -ls (or)

[cloudera@quickstart Desktop]$ hdfs dfs -ls /user/cloudera

Q4) Return all the directories under root directory

[cloudera@quickstart Desktop]$ hdfs dfs -ls /

Q5) Copy a file from local to HDFS?

Command**:copyFromLocal**

This command is used to copy the file from a Local file system to HDFS

Usage: **hdfs dfs -copyFromLocal <localsrc><hdfs destination>**

First I have created a file in local named student.txt

[cloudera@quickstart Desktop]$ gedit students.txt

[cloudera@quickstart Desktop]$ hdfs dfs -copyFromLocal students.txt /user/cloudera

Q6)Check if the file is copied to HDFS?

[cloudera@quickstart Desktop]$ hdfs dfs -ls

Q7) Check the contents of students.txt (or file name you have given) in Hdfs ?

Command**: cat**

This command is uesd to read a file on Hdfs and print content of file to the standardoutput.

Usage: **hdfs dfs –cat /path_to_file_in_hdfs**

[cloudera@quickstart Desktop]$ hdfs dfs -cat students.txt

1,kriti,cse,45

2,neha,ece,56

Q8) Achieve the same operation as above with copyFromLocal with put command?

This command is used to copy single source or multiple sources from local file system to the destination file system.

Usage: **hdfs dfs -put <localsrc><destination>**

[cloudera@quickstart Desktop]$ hdfs dfs -put students.txt /user/cloudera/studentscopied.txt [cloudera@quickstart Desktop]$ hdfs dfs -ls

Q9) Copy any file from HDFS to Local File System

Command: **copyToLocal**

This command is used to copy the file from HDFS to Local File System.

Usage: **hdfs dfs -copyToLocal <hdfs source><localdst>**

[cloudera@quickstart Desktop]$ hdfs dfs -copyToLocal students.txt studentscopiedtolocal.txt

[cloudera@quickstart Desktop]$ ls

Q10) HDFS Command to check the health of the Hadoop file system.

Command: **fsck**

This command is used to check the health of the Hadoop file system.

Usage: **hdfs fsck /**

The filesystem under path '/' is HEALTHY

Q11) Create a directory in HDFS

Command**: mkdir** -- >This command is used to create the directory in HDFS.

Usage: **hdfs dfs –mkdir /directory_name**

[cloudera@quickstart Desktop]$ hdfs dfs -mkdir /kriti2018

[cloudera@quickstart Desktop]$ hdfs dfs -ls

Q12) create a file inside a directory in HDFS

[cloudera@quickstart Desktop]$ hdfs dfs -mkdir /kriti2018/bda.txt

[cloudera@quickstart Desktop]$ hdfs dfs -ls

Q13) Create an emp1ty file in HDFS?

Command**:Touchz -**This command is used to create a file in HDFS with file size 0 bytes.

Usage: **hdfs dfs –touchz /directory/filename**

[cloudera@quickstart Desktop]$ hdfs dfs –touchz emp1ty.txt

[cloudera@quickstart Desktop]$ hdfs dfs -ls

Q14) Check the file size of any file in HDFS?

Command: **du -** This command is used to check the file size.

Usage**: hdfs dfs –du –s /directory/filename**

[cloudera@quickstart Desktop]$ hdfs dfs –du students.txt

67 67 /user/cloudera/students.txt

Q15) Print the contents of a file stored in HDFS?

Command: **cat -**This command is used to read a file on HDFS and print the content of that file to the standard output.

Usage**: hdfs dfs –cat /path/to/file_in_hdfs**

[cloudera@quickstart Desktop]$ hdfs dfs -cat students.txt

1,kriti,cse,45

2,neha,ece,56

Q16) Count the number of directories and files inside a directory in HDFS?

Command:**count**

This command is used to count the number of directories, files, and bytes under the paths that match the specified file pattern.

Usage**: hdfs dfs -count<path>**

[cloudera@quickstart Desktop]$ hdfs dfs –count kriti2018

1 1 13 /user/cloudera/kriti2018

Q17) Remove a file from HDFS?

Command:**rm**

This command is used to remove the file from HDFS.

Usage**: hdfs dfs dfs –rm <path>**

[cloudera@quickstart Desktop]$ hdfs dfs -rm emp1ty.txt

Deleted emp1ty.txt

Q18) Delete a directory completely in HDFS?

Command:**rm -r**

This command is used to remove the entire directory and all of its content from HDFS.

Usage: **hdfs dfs -rm -r<path>**

[cloudera@quickstart Desktop]$ hdfs dfs -rm –r kriti2018

Q19) Copy a file or multiple files in a directory in HDFS

Command:**cp**

This command is used to copy files from source to destination. This command allows multiple sources as well, in which case the destination must be a directory.

Usage**: hdfs dfs -cp<src><dest>**

hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2

[cloudera@quickstart Desktop]$ hdfs dfs -mkdir /user/cloudera/kriti2018

[cloudera@quickstart Desktop]$ hdfs dfs –cp dummy3.txt kriti2018

Q20) Move a file into a directory in HDFS?

Command: **mv**

This command is used to move files from source to destination and allows multiple sources as well, in which case the destination needs to be a directory.

Usage**: hdfs dfs -mv <src><dest>**

[cloudera@quickstart Desktop]$ hdfs dfs –mv emp1tyfile.txt kriti2018

Q21) Find a help for an individual command?

This command gives all the options that can be used with a particular hdfs command.and is used to return the help for an individual command.

Usage: hdfs dfs -usage<command>

Command: hdfs dfs -usage mkdir

[cloudera@quickstart Desktop]$ hdfs dfs -usage mkdir

Q22) Find the help for a given or all commands?

Command:**help**

This command displays help for given command or all commands if none is specified.

Usage: **hdfs dfs -help**

Q23) Check the memory status?

Command: **df**

This command is used to check memory status

Usage**: hdfs dfs -df**

[cloudera@quickstart Desktop]$ hdfs dfs -df

Filesystem Size Used Available Use%

hdfs://quickstart.cloudera:8020 58531520512 1245229056 46116413440 2%

Q24) Check for cluster balancing in HDFS?

Command:**balancer**

This command is used for Cluster Balancing

Usage: **hadoop balancer**

Q25) Change permission for a file to 777?

Command: **chmod**

This command changes the permissions of files.

[cloudera@quickstart Desktop]$ hdfs dfs -ls -r

[cloudera@quickstart Desktop]$ hdfs dfs -chmod 777 /user/cloudera/students.txt

[cloudera@quickstart Desktop]$ hdfs dfs -ls -r

Q26) Empty the trash in HDFS?

Command:**expunge**

This command emp1ties the trash. When you delete a file, it isn't removed immediately from HDFS, but is renamed to a file in the /trash directory.

Usage**: hdfs dfs -expunge**

27) Display the last kilobyte of the particular file?

Command: **tail**

This hadoop command will show the last kilobyte of the file to stdout.

Usage: **hdfs dfs -tail <path>**

[cloudera@quickstart Desktop]$ hdfs dfs -tail /user/cloudera/n.txt

28) Append the contents of a file present in local to a file present in HDFS

[cloudera@quickstart Desktop]$ gedit first.txt

[cloudera@quickstart Desktop]$ gedit second.txt

[cloudera@quickstart Desktop]**$ hdfs dfs -copyFromLocal second.txt /user/cloudera/**

[cloudera@quickstartDesktop]**$hdfs dfs -appendToFile**

/home/cloudera/Desktop/first.txt /user/cloudera/second.txt

[cloudera@quickstart Desktop]$ **-cat /user/cloudera/second.txt**

RESULT: The HDFS (Hadoop Distributed File System) Commands executed succesfully.

Program: 9

AIM: To implement basic PIG commands.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

BASIC PIG COMMANDS:

1.To enter into pig:

Command:**pig -x local**

2.To list all the files in the HDFS:

- Command:**fs -ls**
- Usage:fs -ls

3.To clear the interactive Grunt shell:

- Command: **clear**
- Usage: clear

4.To see the commands that executed so far:

- Command:**history**
- Usage:history

5.To read the data that reside in HDFS to Pig:

- Command: **variable= LOAD <hdfs_path> USING PigStorage(‘,’) as(value1:property2,value2:property2,.....);**
- Usage: college_students = LOAD ‘hdfs://localhost:9000/pig_data/college_data.txt’ USING PigStorage(‘,’)
  as (id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray);
- PigStorage() is the function that loads and stores data as structured text files

6.To store the processed/loaded data:

- Command:**STORE variable INTO <hdfs_path> USING PigStorage(‘,’);**
- Usage: STORE college_students INTO ‘ hdfs://localhost:9000/pig_Output/ ‘ USING PigStorage (‘,’);
- Here, “/pig_Output/” is the directory where relation needs to be stored.

7.To display the results on screen:

- Command:**DUMP variable;**
- Usage: Dump college_students;

- It usually helps in debugging.

8.To view the schema of the relation:

- Command: **DESCRIBE variable;**
- Usage: describe college_students;

RESULTS: Implementation of basic PIG commands is done successfully.

Program: 10

AIM: To install PIG and implement wordcount program using pig commands.

Name: ***Roll no:***

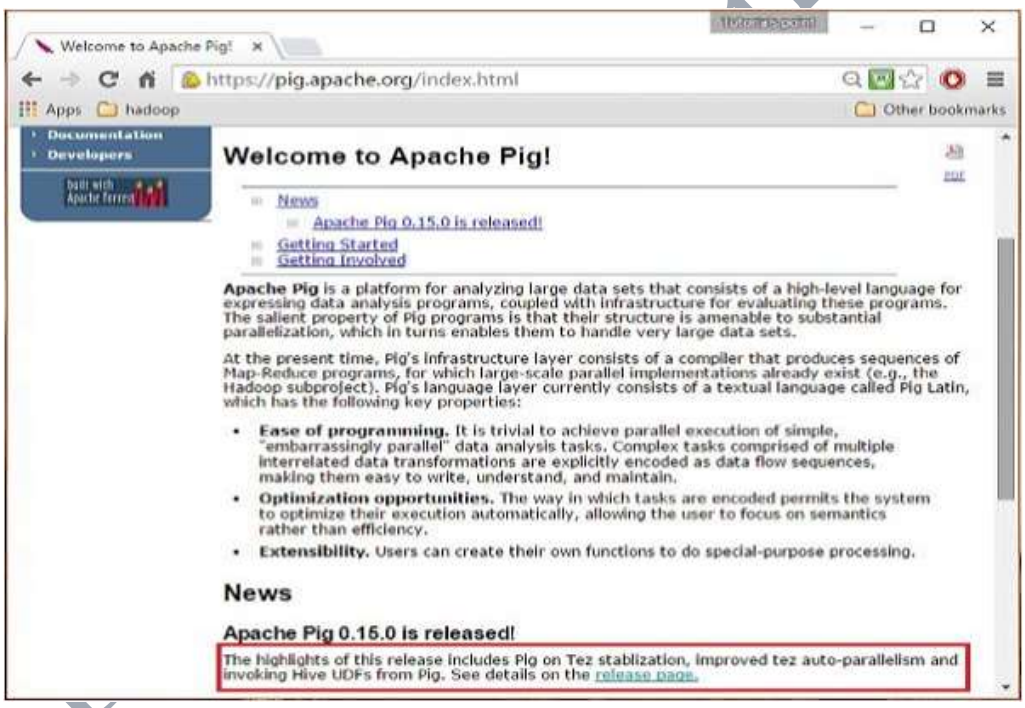
Branch & Sec: CSE - ***Date:***

Download Apache Pig:

First of all, download the latest version of Apache Pig from the following website − https://pig.apache.org/

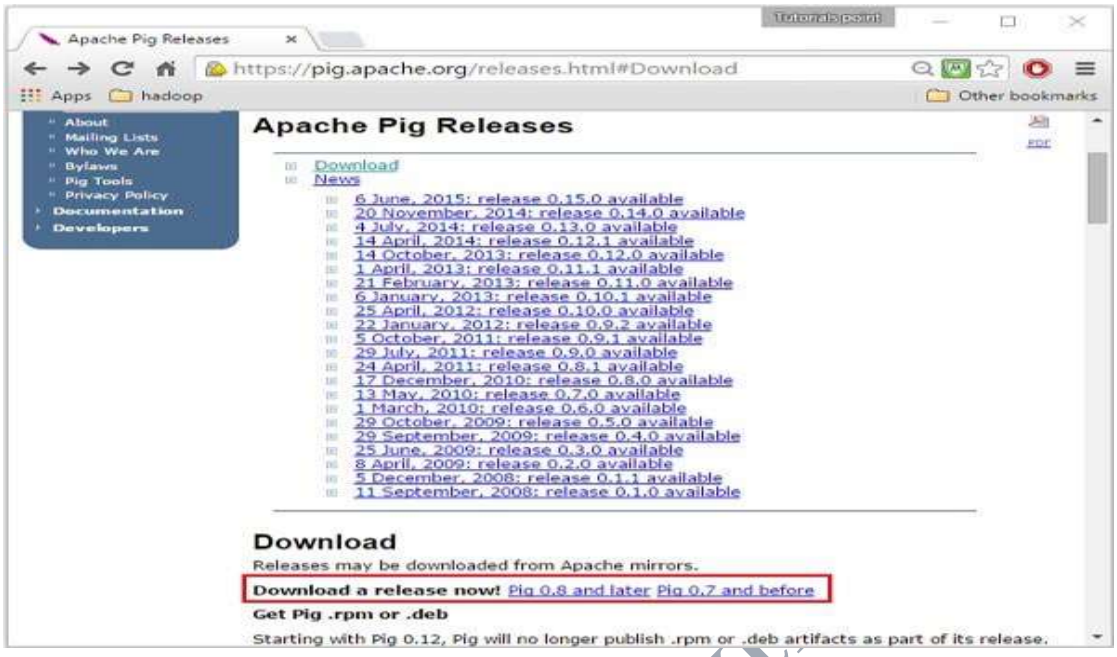
Step 1:

Open the homepage of Apache Pig website. Under the section **News,** click on the link **release page** as shown in the following snapshot.



Step 2:

On clicking the specified link, you will be redirected to the **Apache Pig Releases** page. On this page, under the **Download** section, you will have two links, namely, **Pig 0.8 and later** and **Pig 0.7 and before**. Click on the link **Pig 0.8 and later**, then you will be redirected to the page having a set of mirrors.

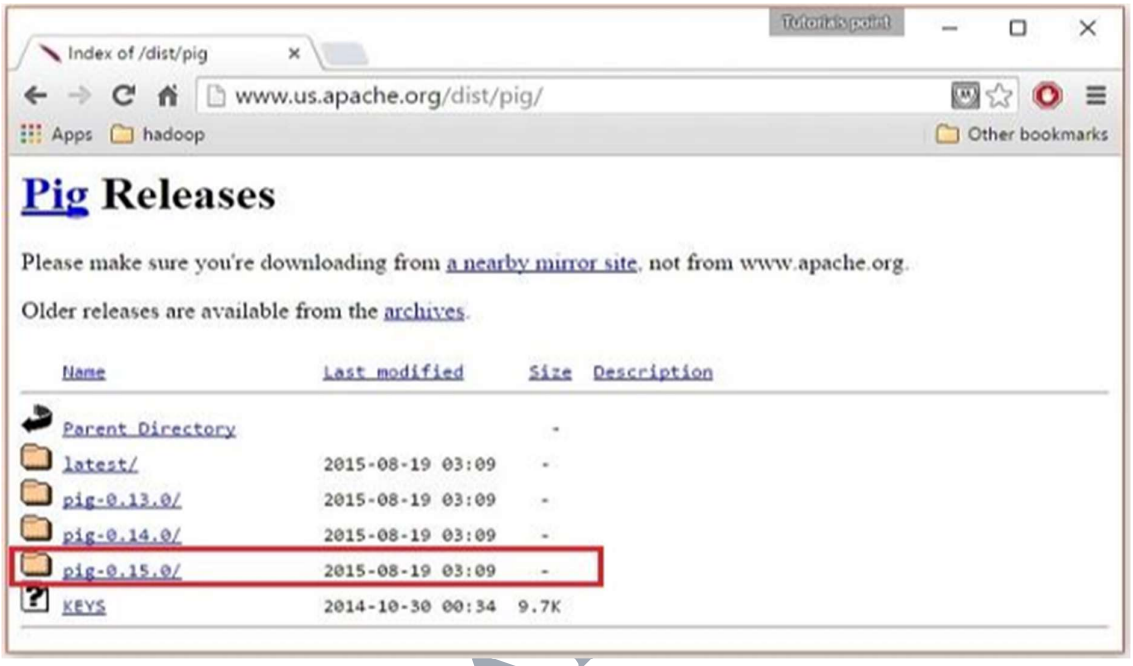


Step 3

Choose and click any one of these mirrors as shown below.

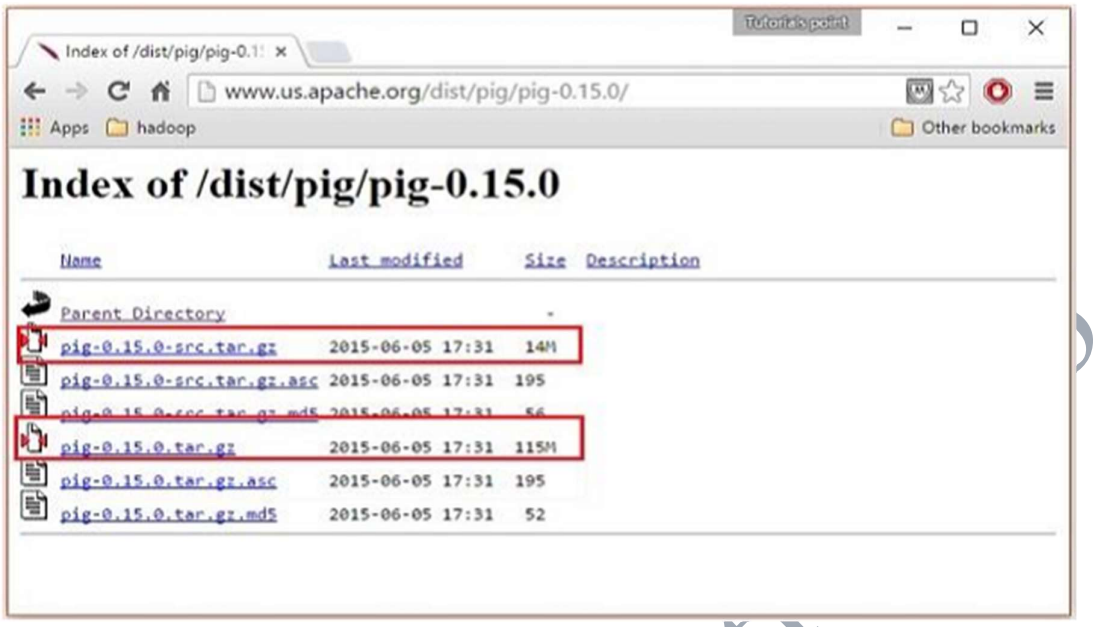
Step 4

These mirrors will take you to the **Pig Releases** page. This page contains various versions of Apache Pig. Click the latest version among them.



Step 5

Within these folders, you will have the source and binary files of Apache Pig in various distributions. Download the tar files of the source and binary files of Apache Pig 0.15, **pig0.15.0-src.tar.gz** and **pig-0.15.0.tar.gz.**



Install Apache Pig

After downloading the Apache Pig software, install it in your Linux environment by following the steps given below.

Step 1

Create a directory with the name Pig in the same directory where the installation directories of **Hadoop, Java,** and other software were installed. (In our tutorial, we have created the Pig directory in the user named Hadoop).

```
$ mkdir Pig
```

Step 2

Extract the downloaded tar files as shown below.

```
$ cd Downloads/
$ tar zxvf pig-0.15.0-src.tar.gz
$ tar zxvf pig-0.15.0.tar.gz
```

Step 3

Move the content of **pig-0.15.0-src.tar.gz** file to the **Pig** directory created earlier as shown below.

```
$ mv pig-0.15.0-src.tar.gz/* /home/Hadoop/Pig/
```

Configure Apache Pig

After installing Apache Pig, we have to configure it. To configure, we need to edit two files − **bashrc and pig.properties**.

.bashrc file

In the **.bashrc** file, set the following variables −

- **PIG_HOME** folder to the Apache Pig's installation folder,
- **PATH** environment variable to the bin folder, and
- **PIG_CLASSPATH** environment variable to the etc (configuration) folder of your Hadoop installations (the directory that contains the core-site.xml, hdfs-site.xml and mapred-site.xml files).

export PIG_HOME = /home/Hadoop/Pig
export PATH = $PATH:/home/Hadoop/pig/bin
export PIG_CLASSPATH = $HADOOP_HOME/conf

pig.properties file

In the **conf** folder of Pig, we have a file named **pig.properties**. In the pig.properties file, you can set various parameters as given below.

pig -h properties

Verifying the Installation

Verify the installation of Apache Pig by typing the version command. If the installation is successful, you will get the version of Apache Pig as shown below.

$ pig –version

Apache Pig version 0.15.0 (r1682971)
compiled Jun 01 2015, 11:44:35

Pig Program for WordCount:

Step1:

Use Load statement to load the data into a relation .As keyword used to declare column names, as we dont have any columns, we declared only one column named line.

input1 = LOAD '/home/cloudera/Desktop/pigword.txt' as (line:Chararray);

Step2 & Step3:

Convert the Sentence into words.

The data we have is in sentences. So we have to convert that data into words using TOKENIZE Function.

Convert Column into Rows

we have to convert every line of data into multiple rows ,for this we have function called FLATTEN in pig.

Words = FOREACH input1 GENERATE FLATTEN(TOKENIZE(line,' '))AS word;

Step4:

Apply GROUP BY

We have to count each word occurance, for that we have to group all the words.

Grouped =GROUP Words BY word;

Step5:

Generate word count

wordcount = FOREACH Grouped GENERATE group, COUNT(Words);

Step6:

Print the word count on console using Dump.

DUMP wordcount;

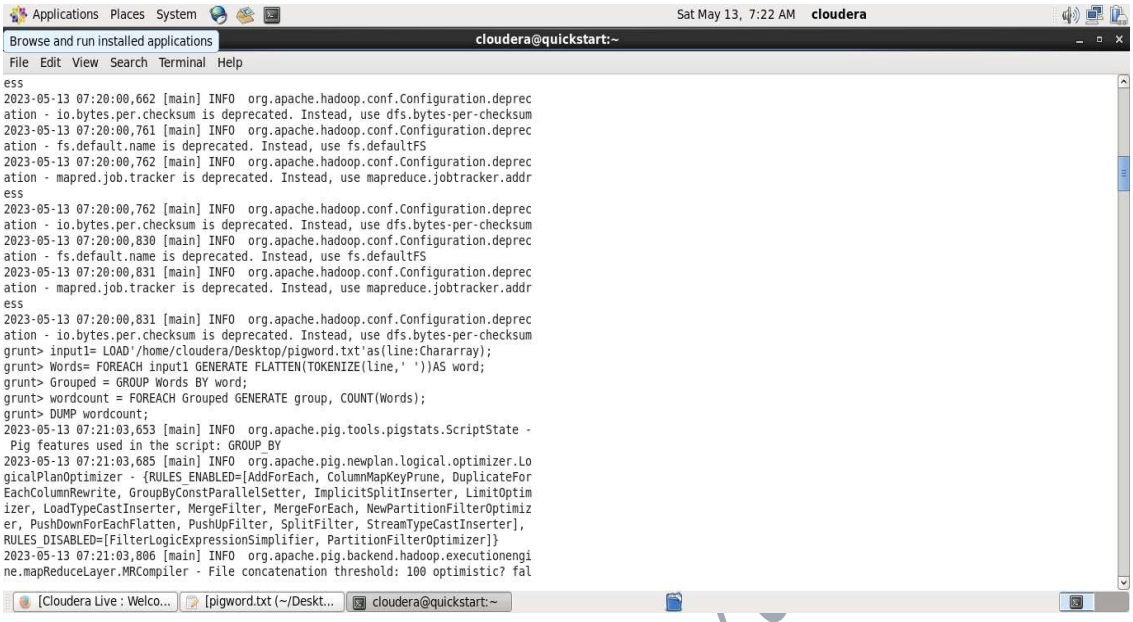
Input:

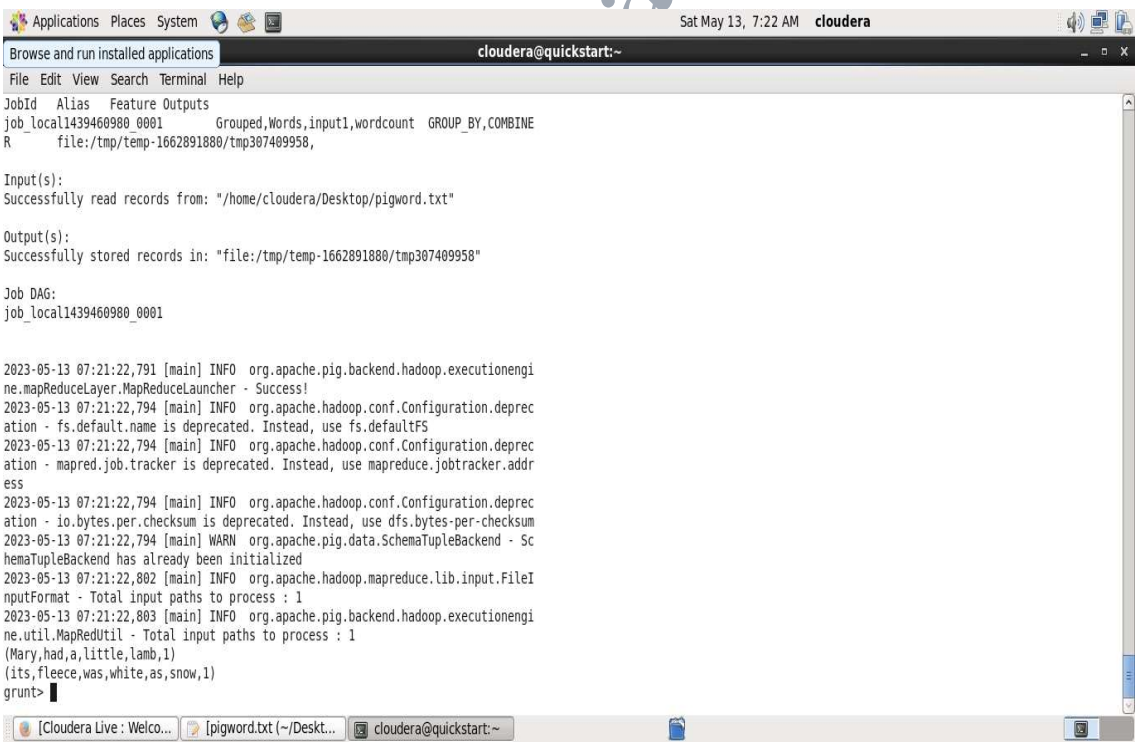
pigword.txt

Marry had a little lamb

Its fleece was white as snow

OUTPUT:





RESULTS: Implementation wordcount program using pig commands is done successfully.

Program: 11

AIM: To find the max Temperature in the given weather dataset using pig Latin.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

PIG WEATHER DATA SET:

weatherdataset.txt

1947 87

1978 78

1941 31

1945 78

1988 90

1999 10

2000 21

Step1:

A = LOAD '/home/cloudera/Desktop/weatherdataset.txt' USING PigStorage(' ') AS (Year:int,Temp:int);

Step2:

B= ORDER A BY Temp DESC;

Step3:

C = LIMIT B 1;

Step4:

D = FOREACH C GENERATE Temp;

Step5:

DUMP D;

OUTPUT:

Success!

Job Stats (time in seconds):

JobId	Alias	FeatureOutputs
job_local14883896_0014	B,D	file:/tmp/temp-1026199787/tmp2135421797,
job_local1587477527_0011	A	MAP_ONLY
job_local1878033746_0013	B	ORDER_BY,COMBINER
job_local690663710_0012	B	SAMPLER

Input(s):
Successfully read records from: "/home/cloudera/Desktop/weatherdataset.txt"

Output(s):
Successfully stored records in: "file:/tmp/temp-1026199787/tmp2135421797"

Job DAG:
job_local1587477527_0011 -> job_local690663710_0012,
job_local690663710_0012 -> job_local1878033746_0013,
job_local1878033746_0013 -> job_local14883896_0014,
job_local14883896_0014

2023-05-03 08:40:55,524 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2023-05-03 08:40:55,526 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2023-05-03 08:40:55,526 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2023-05-03 08:40:55,528 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2023-05-03 08:40:55,529 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2023-05-03 08:40:55,553 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2023-05-03 08:40:55,553 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(1988,90)

RESULT: Weather dataset program using pig is executed successfully.

Program: 12

AIM: Write the Hive script to create, alter and drop datbases, tables, views, functions, and indexes of the sample data.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:***

HIVE:

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Hive Commands:

SYNTAX for HIVE Database Operations:

1.To create database:

- CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

2.To drop database:

- DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];

3.To create and drop table:

- CREATE [TEMP1ORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name [(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment] [ROW FORMAT row_format] [STORED AS file_format]

4.To load data into table:

- LOAD DATA LOCAL INPATH '<path>/u.data' OVERWRITE INTO TABLE u_data;

5.To alter table:

- ALTER TABLE name RENAME TO new_name
- ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
- ALTER TABLE name DROP [COLUMN] column_name
- ALTER TABLE name CHANGE column_name new_name new_type
- ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

6.To create a view:

- CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...)] [COMMENT table_comment] AS SELECT

7.To drop view:

- DROP VIEW view_name

8.Functions in Hive:

- **String Functions**:- round(), ceil(), substr(), upper(), reg_exp() etc
- **Date and Time Functions**:- year(), month(), day(), to_date() etc
- **Aggregate Functions** :- sum(), min(), max(), count(), avg() etc

9.INDEXES:

- CREATE INDEX index_name ON TABLE base_table_name (col_name, ...) AS 'index.handler.class.name' [WITH DEFERRED REBUILD] [IDXPROPERTIES (property_name=property_value, ...)] [IN TABLE index_table_name] [PARTITIONED BY (col_name, ...)] [[ROW FORMAT ...] STORED AS ... | STORED BY ...] [LOCATION hdfs_path] [TBLPROPERTIES (...)]

10.To create index:

- CREATE INDEX index_ip ON TABLE log_data(ip_address) AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
  WITH DEFERRED REBUILD;

11.To alter and insert index:

- ALTER INDEX index_ip_address ON log_data REBUILD;

12.To drop index:

- DROP INDEX INDEX_NAME on TABLE_NAME

13.To sort index data in Metastore:

- SET hive.index.compact.file=/home/administrator/Desktop/big/metastore_db/tmp/index_ipadd ress_result;
- SET hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexInputFormat.

Basic Commands in Hive:

1.To enter the Hive shell:

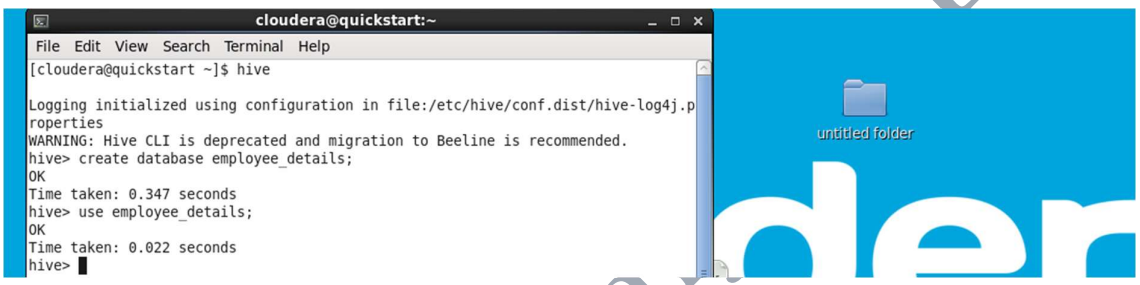
- Command: **hive**

2.To create a database:

- Command: **create database** <*database_name*>
- Usage:: create database employee_details;

3.To use particular database:

- Command: **use database** <*database_name*>
- Usage:: use database employee_details;

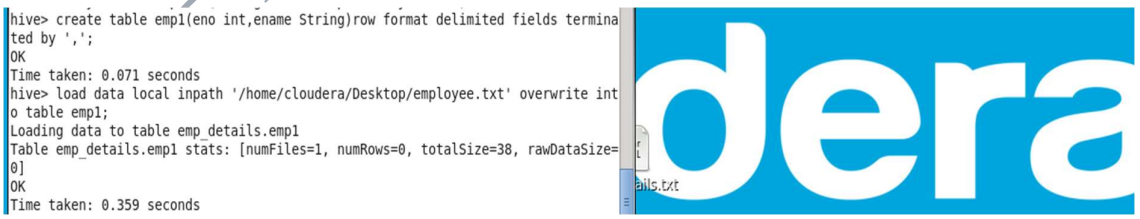


4.To create a table:

- Command: **create table** <*table_name*>(value1 datatype1,value2 datatype2,.....)row format delimited fileds terminated by ‘,’;
- Usage create table emp1(eno int,ename String) row format delimited fileds terminated by‘,’;

5.To describe the property of the table:

- Command: **desc** <*table_name*>;
- Usage:: desc emp1;

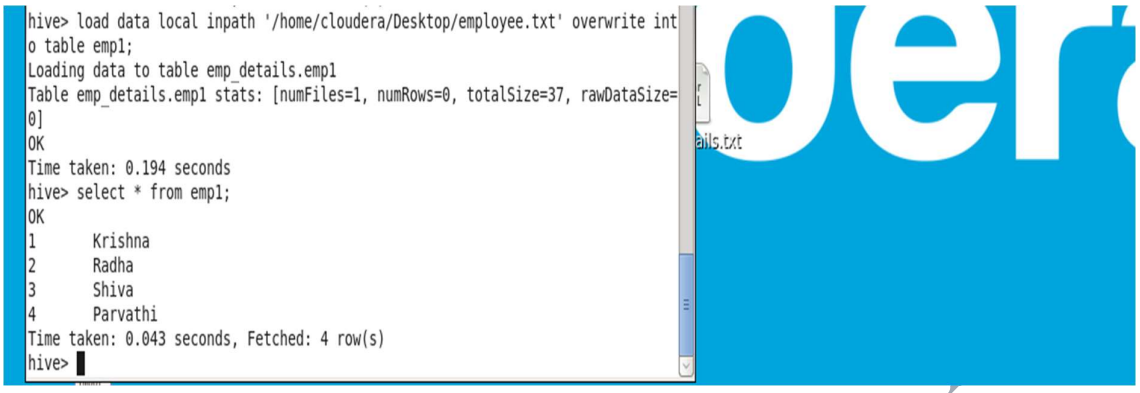


6.To load data into table:

- Command: **load local inpath** <*local_path*> **into table** <*table_name*>;
- Usage:: load data local inpath ‘/home/cloudera/Desktop/employee.txt’ overwrite into table emp1;

7.To display selected table:

- Command: **select * from** <*table_name*>;
- Usage:: select * from emp1;

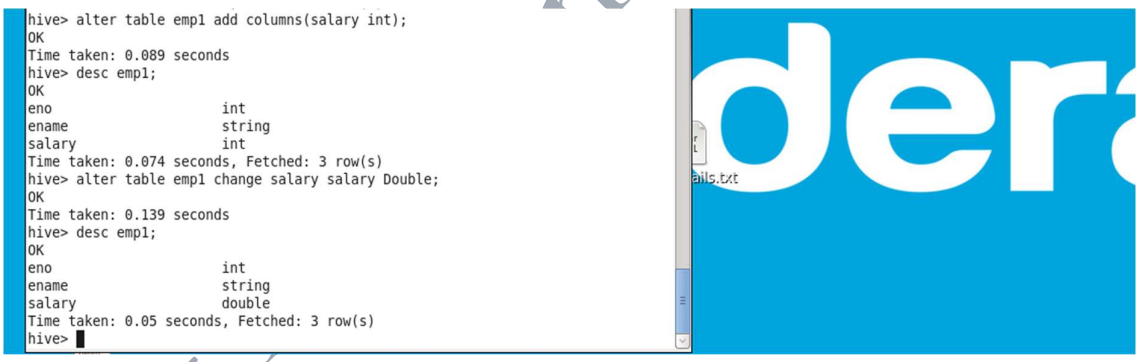


8.To add a new column into current table:

- Command: **alter table** <*table_name*> **add columns(value datatype);**
- Usage:: alter table emp1 add columns(salary int);

9.To change the property of the value:

- Command:**alter table** <*table_name*> **change value value new_datatype;**
- Usage:: alter table emp1 change salary salary Double;



10.To select a record from table:

- Command**: select** <*column_name*> **from** <*table_name*>;
- Usage:: select ename from emp1;

11. To select a name which start with specified letter:

- command: **like *'letter%';***
- Usage::select ename from emp1 where ename like 'K%';

```
hive> select ename from emp1;
OK
Krishna
Radha
Shiva
Parvathi
Time taken: 0.061 seconds, Fetched: 4 row(s)
hive> select ename from emp1 where ename like 'K%';
OK
Krishna
Time taken: 0.101 seconds, Fetched: 1 row(s)
hive>
```

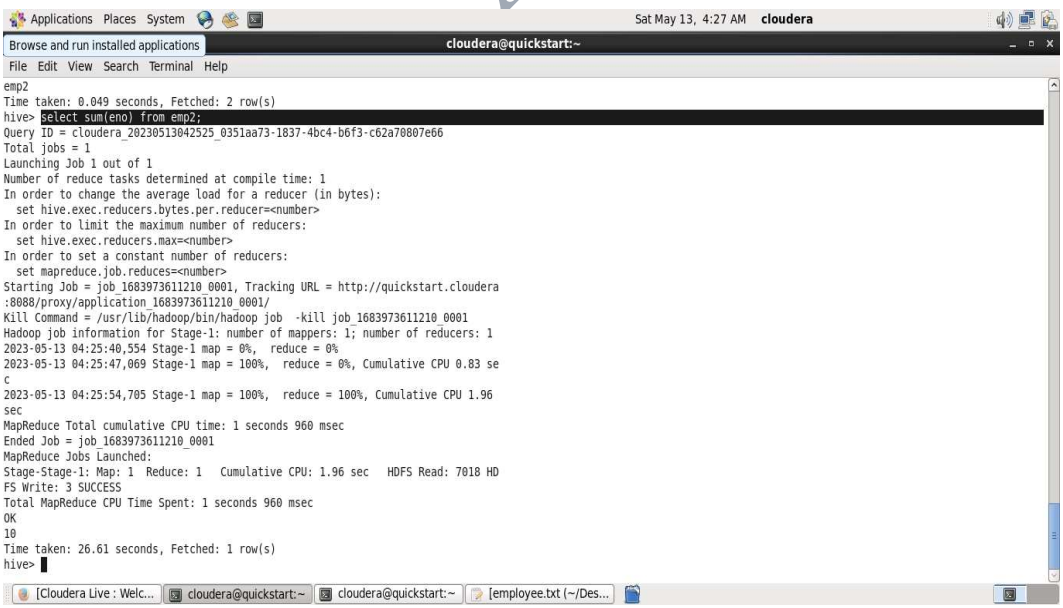
12.To rename tablename:

- command: **alter table** <*table_name*> **rename** <*new_tablename*>;
- Usage:: alter table emp1 rename emp2;

```
hive> alter table emp1 rename to emp2;
OK
Time taken: 0.08 seconds
hive> show tables;
OK
emp
emp2
Time taken: 0.049 seconds, Fetched: 2 row(s)
```

13.To know the sum of specified column:

- Command:**select sum**(<*column_name*>) **from** <*table_name*>;
- Usage: select sum(eno) from emp2;



RESULT: Hive script to create, alter and drop datbases, tables, views, functions, and indexes of the sample data is implemented succesfully.

Program: 13

AIM: Implement Hive WordCount Program.

Name: ***Roll no:***

Branch & Sec: CSE - ***Date:*** 05/10/2021

Hive WordCount Program:

sample.txt:
This is BDA Lab
We are working in Hive
BDA Lab

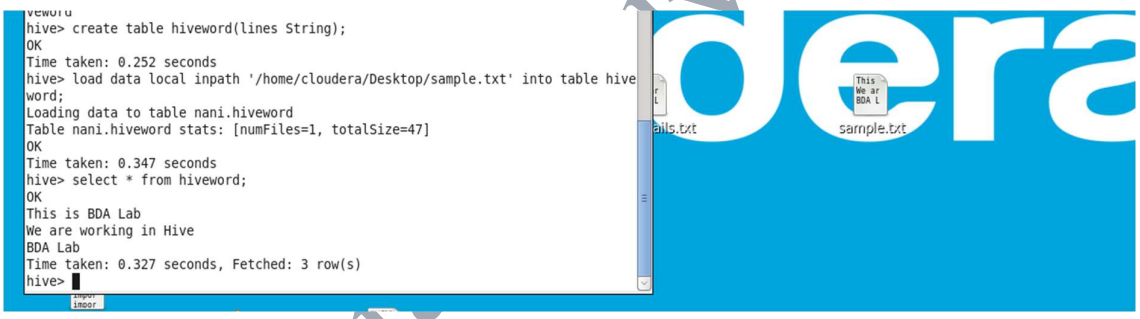
Steps foe WordCount program:

1.Create a table:

- create table hiveword(lines String);

2.Load data from sample file:

- load data local inpath '/home/cloudera/Desktop/sample.txt' into table hiveword;



3.Convert lines into an array of strings:

- select split(lines,' ') from hiveword;

4.Explode the data:

- select explode(split(lines,' ') from hiveword;

```
hive> select explode(split(lines,' '))from hiveword;
OK
This
is
BDA
Lab
We
are
working
in
Hive
BDA
Lab
Time taken: 0.087 seconds, Fetched: 11 row(s)
hive>
```

5.Count the words:

- select word,count(1) as count from (select explode(split(lines,' ') as word from hiveword) w group by word;

Output:

```
OK
BDA     2
Hive    1
Lab     2
This    1
We      1
are     1
in      1
is      1
working 1
Time taken: 24.529 seconds, Fetched: 9 row(s)
hive>
```

[Cloudera Live : Welco...] [sample.txt (~/Deskto...] [cloudera@quickstart:~]

RESULT: Implementation of Hive WordCount Program is done successfully.