ARUNAI ENGINEERING COLLEGE

(An Autonomous Institution)
Velu Nagar, Thiruvannamalai-606603
www.arunai.org

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

BACHELOR OF TECHNOLOGY

2024-2025

FIFTH SEMESTER

CCS334 –BIG DATA ANALYTICS LABORATORY

ARUNAI ENGINEERING COLLEGE

TIRUVANNAMALAI – 606 603

DEPARTMENT OF ARTIFICIAL INTELLIGENCE &DATA SCIENCE

CERTIFICATE

Certified that this is a bonafide record of work done by

Name :

University Reg.No :

Semester :

Branch :

Year :

Staff-in-Charge **Head of the Department**

Submitted for the _____

Practical Examination held on _____

Internal Examiner **External Examiner**

S.NO	DATE	List Of Experiments	Pg.No	Signature
1		**INSTALLATION OF HADOOP**		
2		**HADOOP FILE MANAGEMENT TASKS**		
3		**MATRIX MULTIPLICATION**		
4		**WORD COUNT MAP REDUCE**		
5		**HIVE**		
6.1		**HBASE**		
6.2		**THRIFT**		
7.1		**CASSANDRA**		
7.2		**MongoDB**		

EX.NO: 1 **INSTALLATION OF HADOOP**

DATE:

AIM:

To Download and install Hadoop; Understand different Hadoop modes. Startup scripts, Configuration files.

THEORY:

Hadoop is a Java-based programming framework that supports the processing and storage of extremely large datasets on a cluster of inexpensive machines. It was the first major open-source project in the big data playing field and is sponsored by the Apache Software Foundation.

Hadoop-2.8.0 is comprised of four main layers:

- **Hadoop Common** is the collection of utilities and libraries that support other Hadoop modules.
- **HDFS**, which stands for Hadoop Distributed File System, is responsible for persisting data to disk.
- **YARN**, short for Yet Another Resource Negotiator, is the "operating system" for HDFS.
- **MapReduce** is the original processing model for Hadoop clusters. It distributes work within the cluster or map, then organizes and reduces the results from the nodes into a response to a query. Many other processing models are available for the 2.x version of Hadoop.

Hadoop clusters are relatively complex to set up, so the project includes a stand-alone mode which is suitable for learning about Hadoop, performing simple operations, and debugging.

PREPARE:

These softwares should be prepared to install Hadoop 2.8.0 on window 10 64bit

1. Download Hadoop 2.8.0
2. Java JDK 1.8.0.zip

PROCEDURE:

Procedure to Run Hadoop

1. Install Apache Hadoop 2.8.0 in Microsoft Windows OS

   If Apache Hadoop 2.8.0 is not already installed then follow the post Build, Install, Configure and Run Apache Hadoop 2.8.0 in Microsoft Windows OS.

2. Start HDFS (Namenode and Datanode) and YARN (Resource Manager and Node Manager)

Run following commands.

Command Prompt

C:\Users\> hdfs namenode –format

C:\hadoop\sbin>start-dfs

C:\hadoop\sbin>start-yarn

C:\hadoop\sbin>start-all.cmd

C:\hadoop\sbin>jps (used to check how many nodes are running in background of Hadoop)

Namenode, **Datanode**, **Resource Manager** and **Node Manager** will be started in few minutes and ready to execute Hadoop **MapReduce** job in the Single Node (pseudo-distributed mode) cluster.

PREREQUISITES:

Step1: Installing Java 8 version.

Openjdk version "1.8.0_91"

OpenJDK Runtime Environment (build 1.8.0_91-8u91-b14-3ubuntu1~16.04.1-b14)

OpenJDK 64-Bit Server VM (build 25.91-b14, mixed mode)

This output verifies that OpenJDK has been successfully installed.

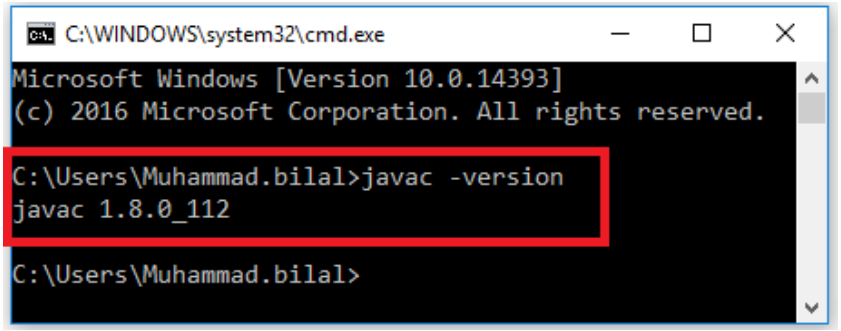
Note: To set the path for environment variables. i.e. JAVA_HOME

Step2: Installing Hadoop

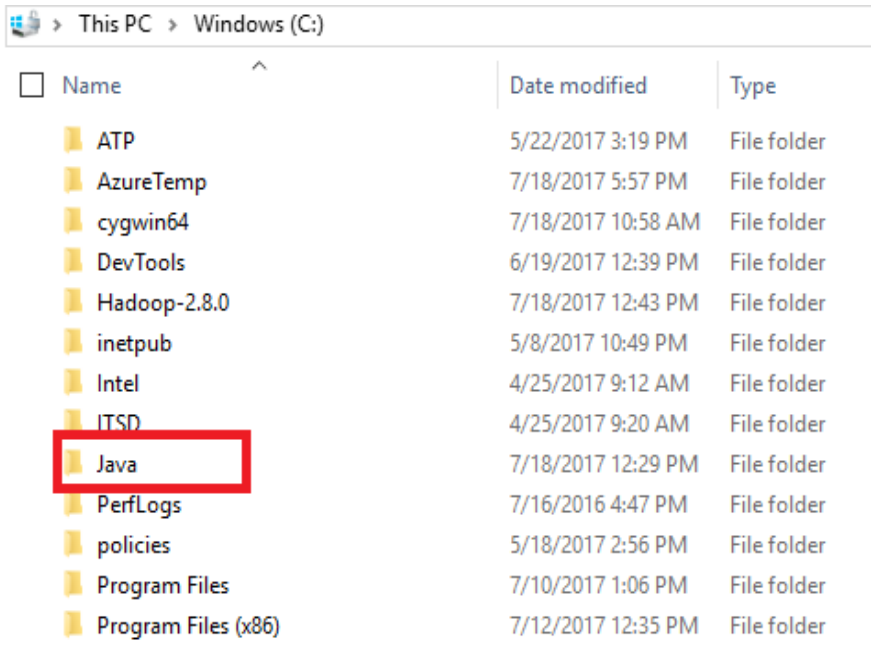
With Java in place, we'll visit the Apache Hadoop Releases page to find the mostrecent stable release. Follow the binary for the current release:

Set up

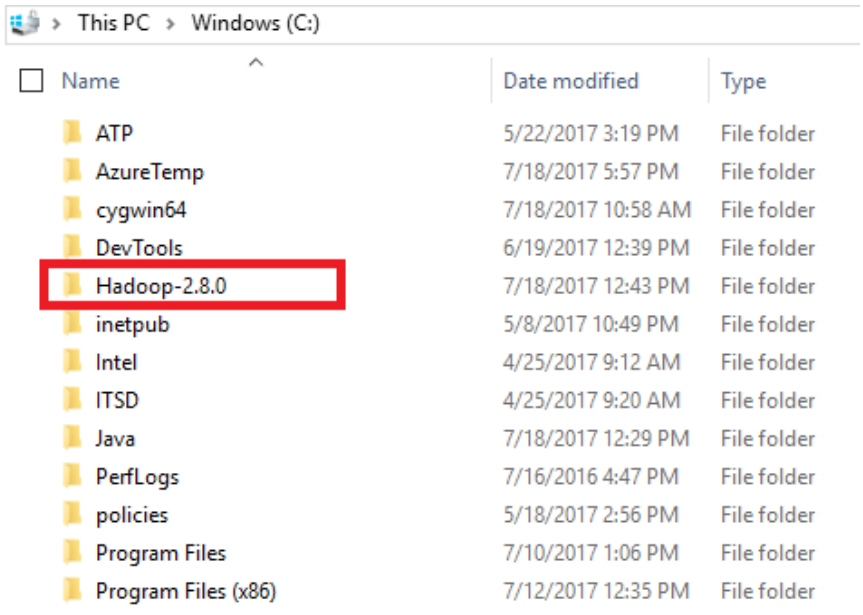
1. Check either Java 1.8.0 is already installed on your system or not, use **"Javac -version"** to check



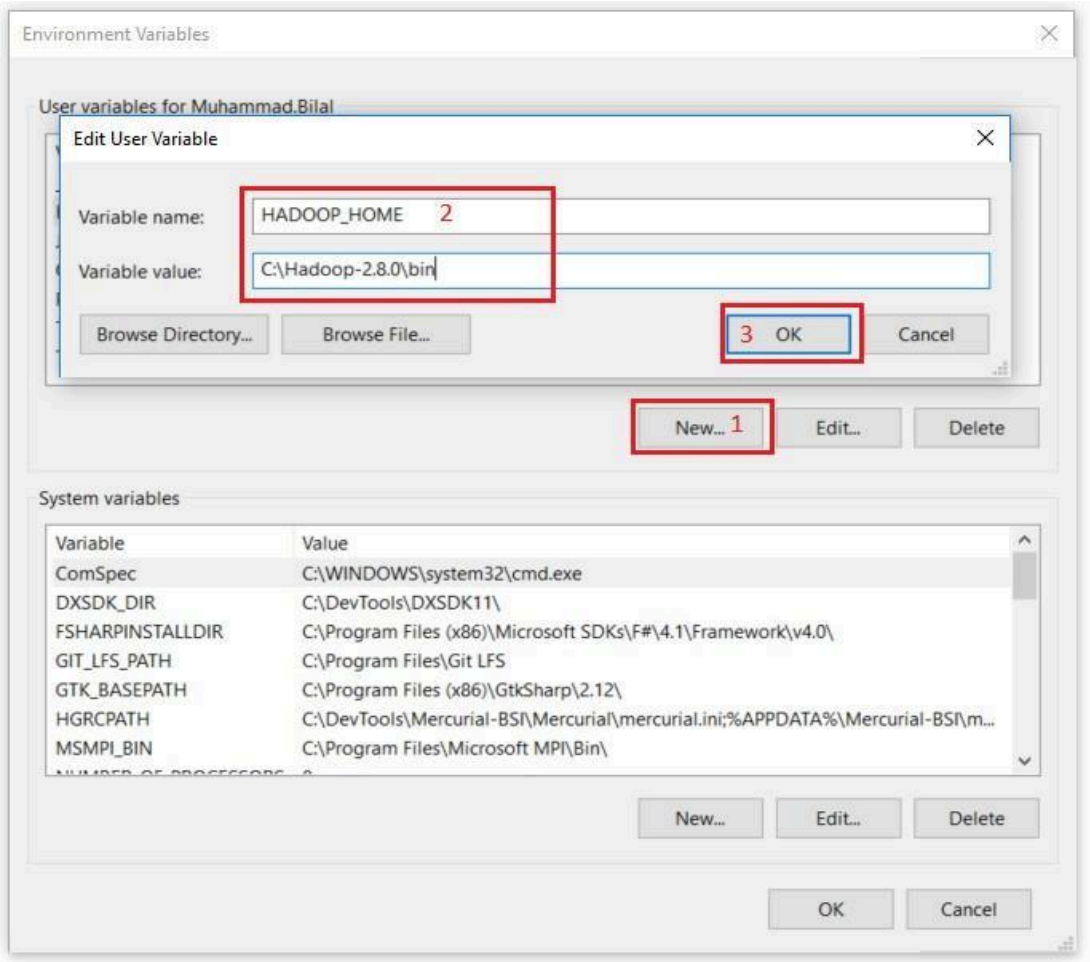
2. If Java is not installed on your system then first install java under **C:\JAVA**



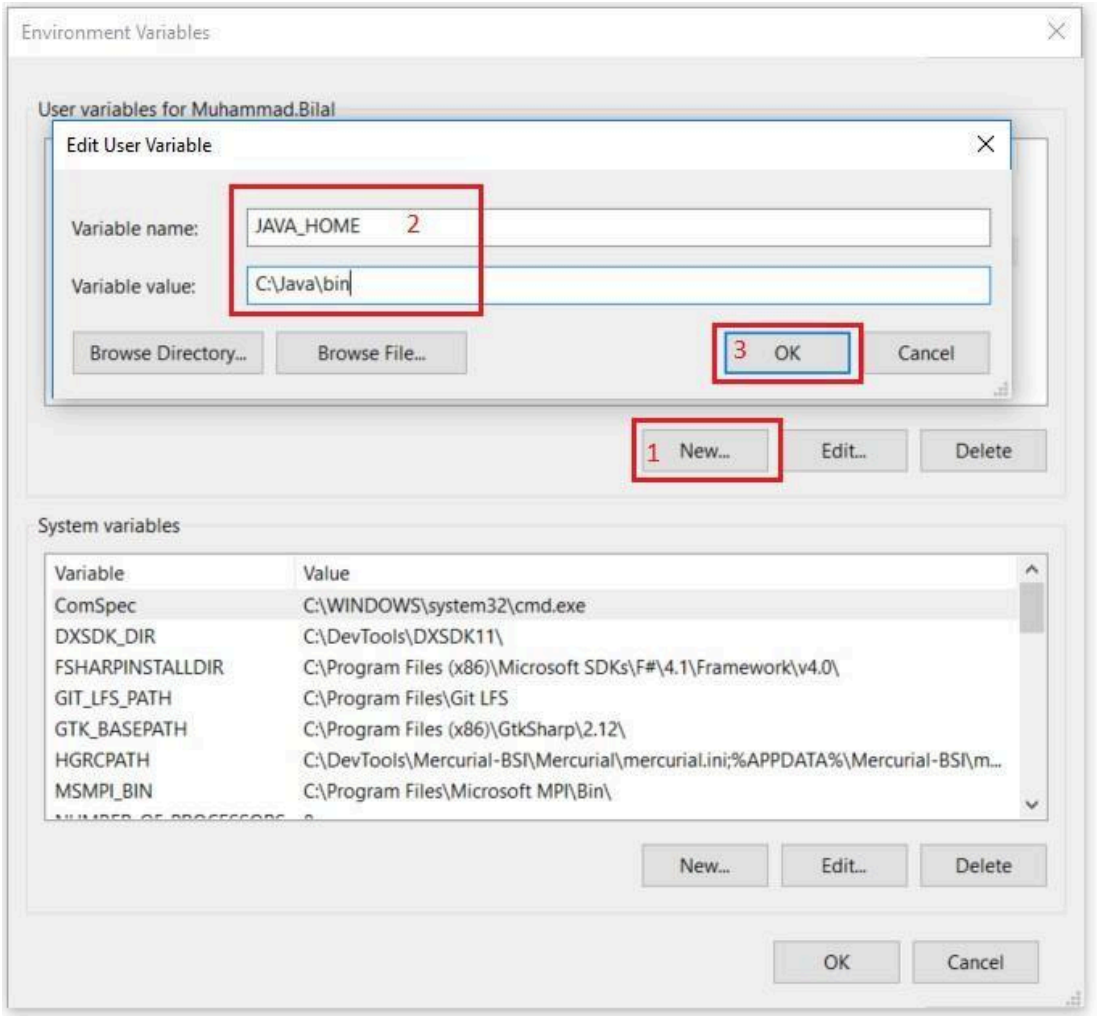
3. Extract file Hadoop 2.8.0.tar.gz or Hadoop-2.8.0.zip and place under **"C:\Hadoop-2.8.0"**



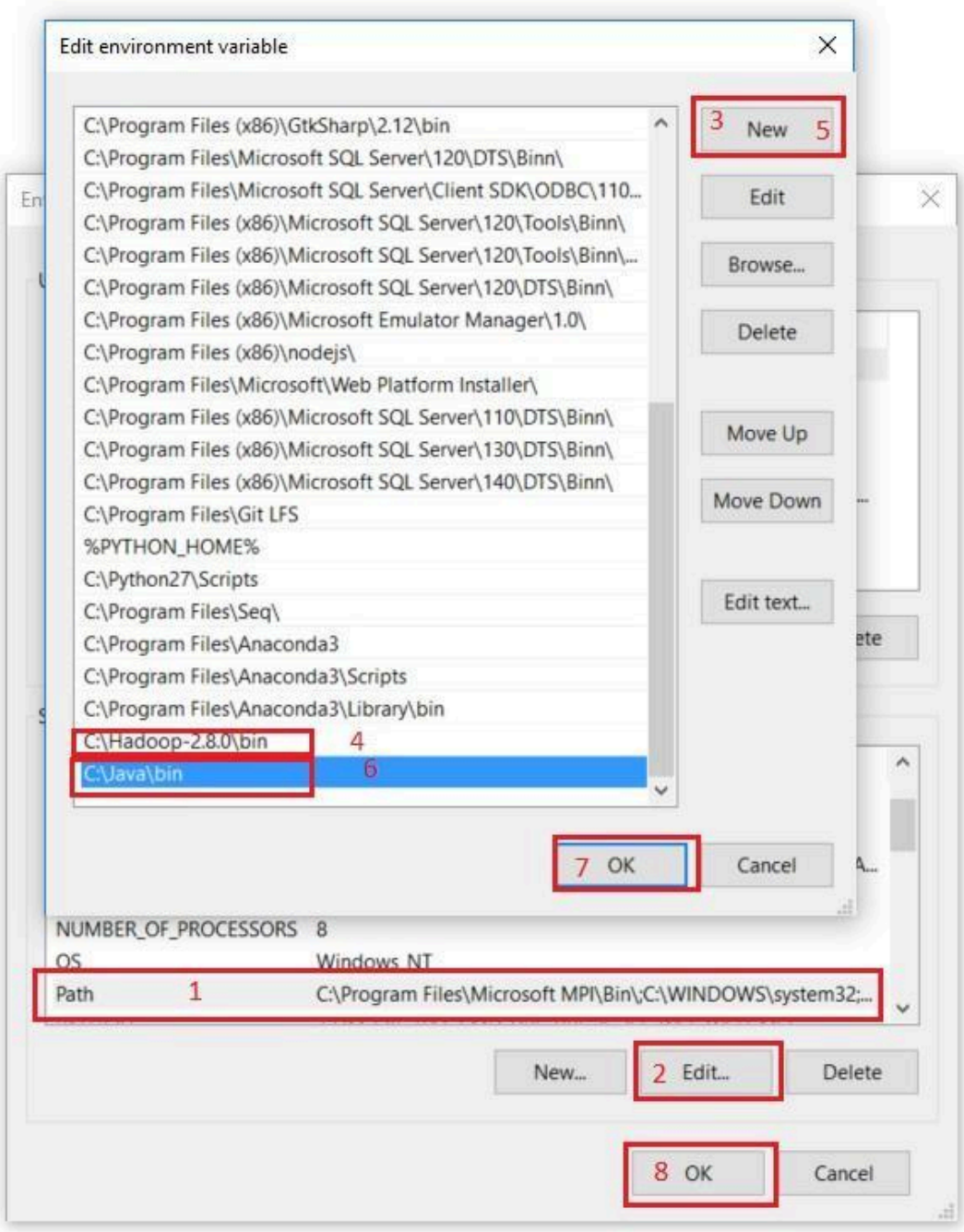
4. Set the path HADOOP_HOME Environment variable on windows 10(see Step 1,2,3 and 4 below)



5. Set the path JAVA_HOME Environment variable on windows 10(see Step 1,2,3 and 4 below)



6. Next, we set the Hadoop bin directory path and JAVA bin path



CONFIGURATION

1. Edit file **C:/Hadoop-2.8.0/etc/hadoop/core-site.xml**, paste below xml paragraph and save this file.

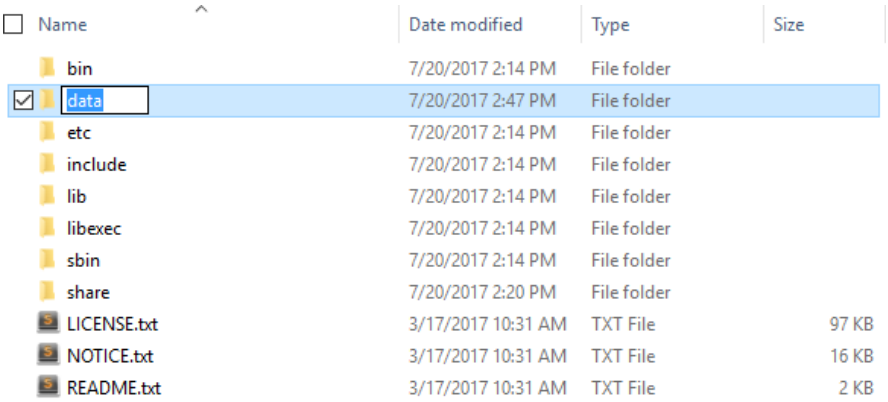
```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

2. Rename "mapred-site.xml.template" to "mapred-site.xml" and edit this file **C:/Hadoop-2.8.0/etc/hadoop/mapred-site.xml**, paste below xml paragraph and save this file.

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

3. Create folder **"data"** under **"C:\Hadoop-2.8.0"**

- Create folder **"datanode"** under **"C:\Hadoop-2.8.0\data"**
- Create folder **"namenode"** under **"C:\Hadoop-2.8.0\data"**



4. Edit file **C:\Hadoop-2.8.0/etc/hadoop/hdfs-site.xml**, paste below xml paragraph and save this file.

```
<configuration>
  <property>
    <name>dfs.replication</name>
```

```
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/hadoop-2.8.0/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/hadoop-2.8.0/data/datanode</value>
  </property>
</configuration>
```

5. Edit file **C:/Hadoop-2.8.0/etc/hadoop/yarn-site.xml**, paste below xml paragraph and save this file.

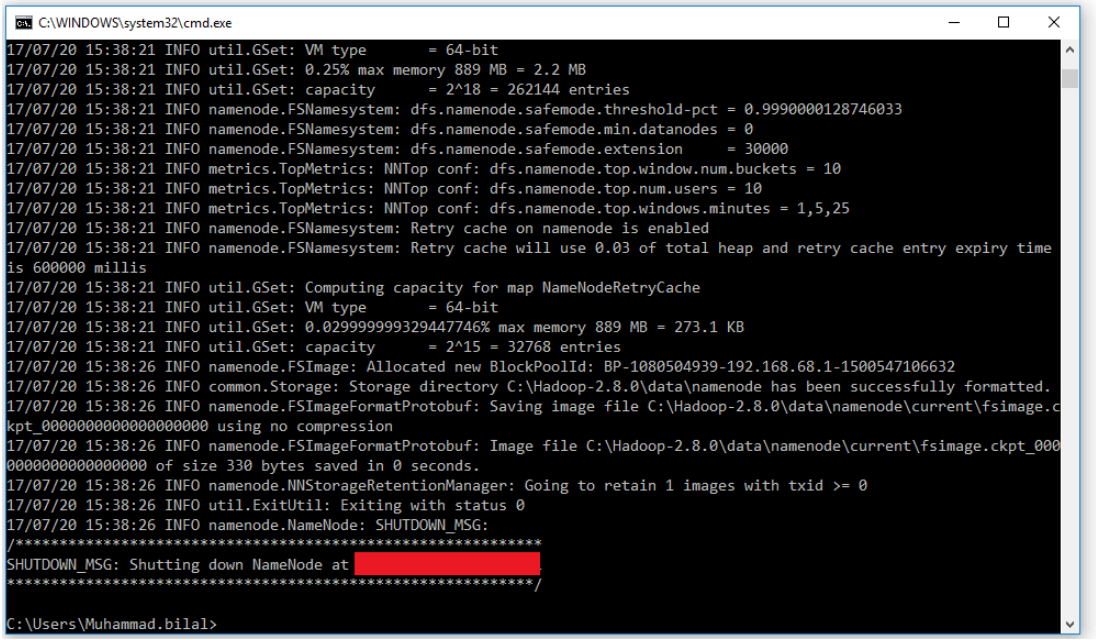
```
<configuration>
  <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
  </property>
  <property>
        <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>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

6. Edit file **C:/Hadoop-2.8.0/etc/hadoop/hadoop-env.cmd** by closing the command line **"JAVA_HOME=%JAVA_HOME%"** instead of set **"JAVA_HOME=C:\Java"** (On C:\java this is path to file jdk.18.0)

```
@rem The java implementation to use.  Required.
@rem set JAVA_HOME=%JAVA_HOME%
set JAVA_HOME=C:\java
```

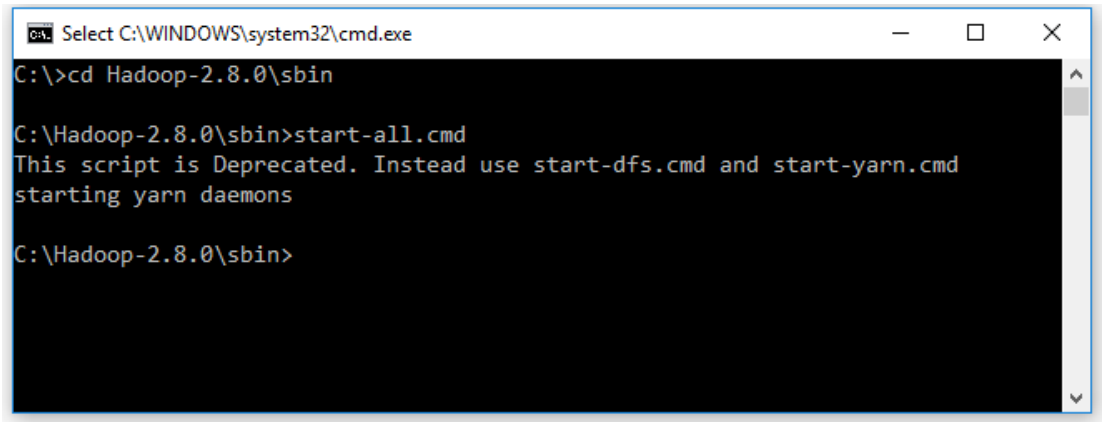
Hadoop Configuration

1. Dowload file Hadoop Configuration.zip
2. Delete file bin on C:\Hadoop-2.8.0\bin, replaced by file bin on file just download (from Hadoop Configuration.zip).
3. Open cmd and typing command **"hdfs namenode –format"** . You will see

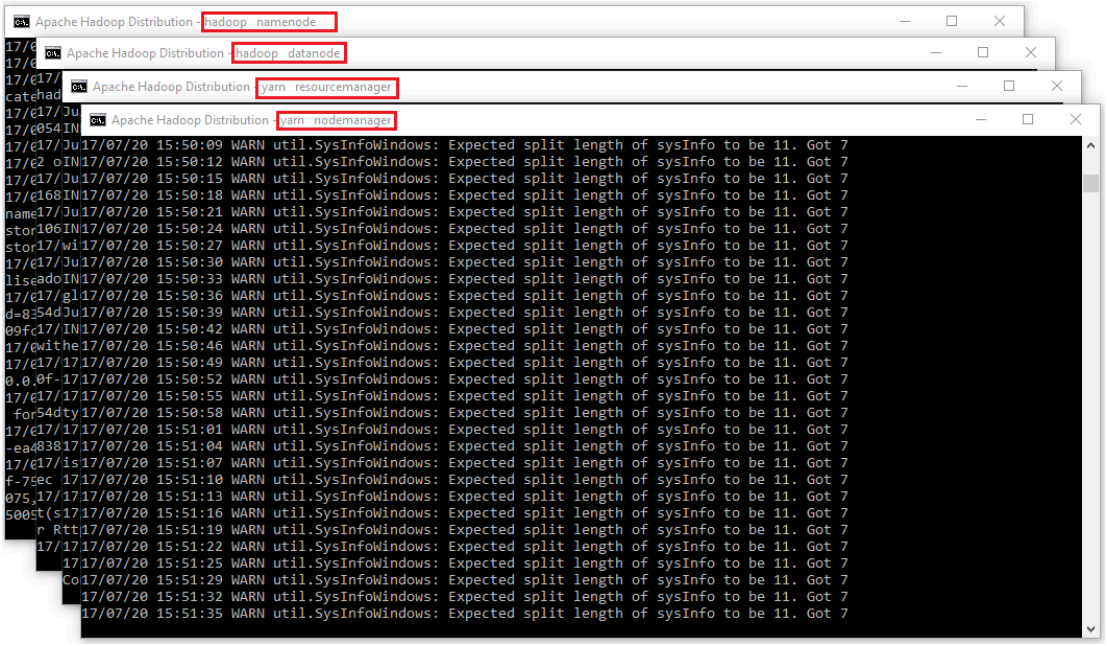


Testing

1. Open cmd and change directory to "C:\Hadoop-2.8.0\sbin" and type **"start-all.cmd"** to start Hadoop



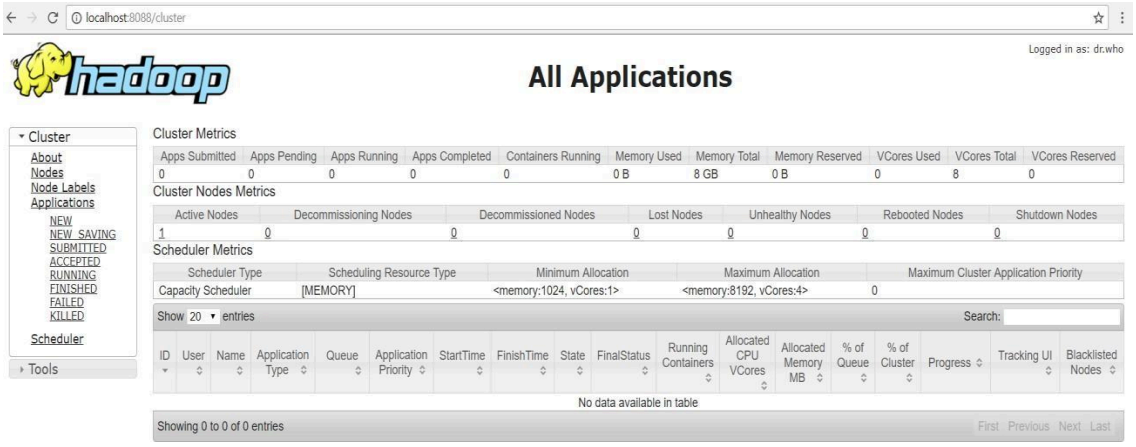
2. Make sure these apps are running
    - Hadoop Namenode
    - Hadoop datanode
    - YARN Resourc Manager
    - YARN Node Manager



3. OUTPUT:

Open:

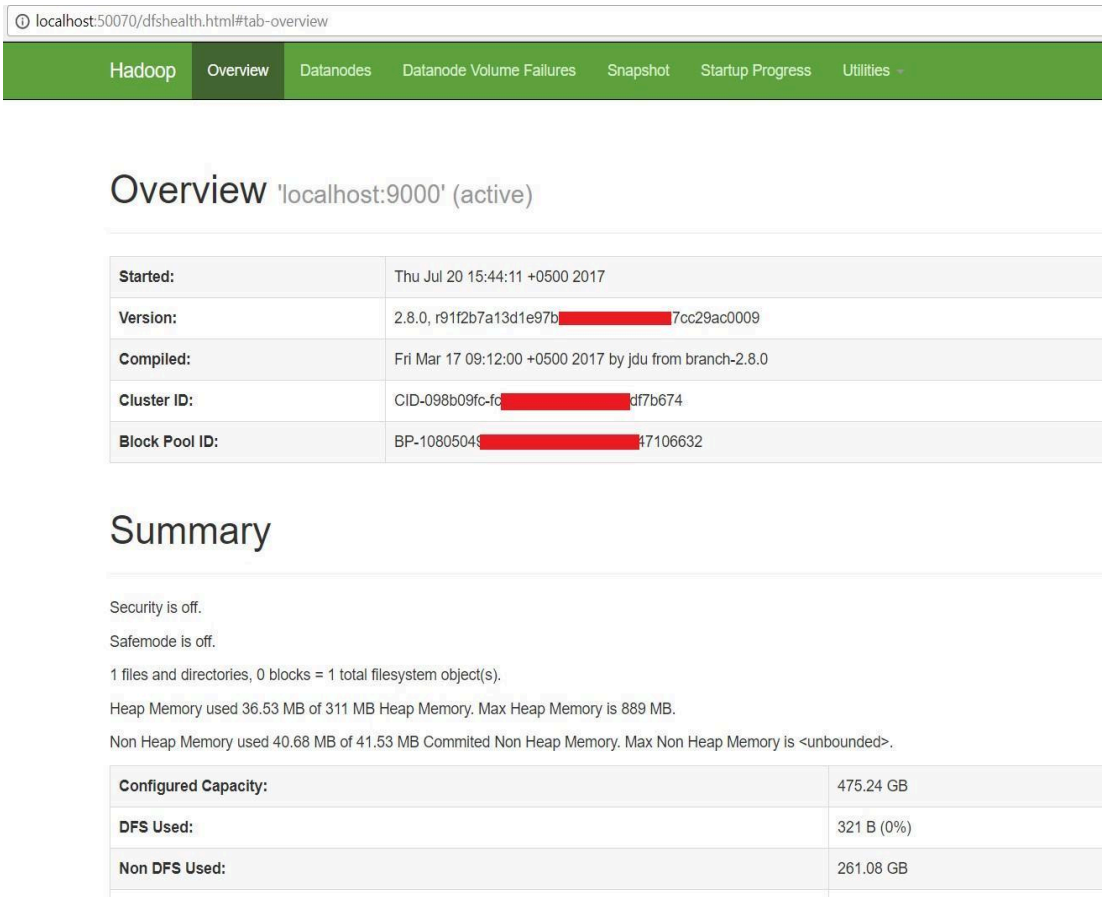
http://localhost:8088



4. **OUTPUT:**

Open:

http://localhost:50070



RESULT:

Thus, a procedure to installation of Hadoop cluster was successfully executed.

EX.NO: 2 **HADOOP FILE MANAGEMENT TASKS**
DATE:

AIM:

Implement the following file management tasks in Hadoop:

- Adding files and directories
- Retrieving files
- Deleting files

DESCRIPTION:

HDFS is a scalable distributed filesystem designed to scale to petabytes of data while running on top of the underlying filesystem of the operating system. HDFS keeps track of where the data resides in a network by associating the name of its rack (or network switch) with the dataset. This allows Hadoop to efficiently schedule tasks to those nodes that contain data, or which are nearest to it, optimizing bandwidth utilization. Hadoop provides a set of command line utilities that work similarly to the Linux file commands, and serve as your primary interface withHDFS. We‘re going to have a look into HDFS by interacting with it from the command line. We will take a look at the most common file management tasks in Hadoop, which include:

- Adding files and directories to HDFS
- Retrieving files from HDFS to local filesystem
- Deleting files from HDFS

ALGORITHM:

SYNTAX AND COMMANDS TO ADD, RETRIEVE AND DELETE DATA FROM HDFS

Step-1: Adding Files and Directories to HDFS

Before you can run Hadoop programs on data stored in HDFS, you‘ll need to put the data into HDFS first. Let‘s create a directory in and put a file in it. HDFS has a default working directory of /user/$USER, where $USER is your login user name. This directory isn‘t automatically created for you, though, so let‘s create it with the mkdir command.

Note: input_file.txt is created in sbin with some contents

```
C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /input_dir
C:\hadoop-2.8.0\sbin>hadoop fs -put input_file.txt /input_dir/input_file.txt
```

Step 2: List the contents of a directory.:

C:\hadoop-2.8.0\sbin>hadoop fs -ls /input_dir/

Step 3: Retrieving Files from HDFS

The Hadoop command get copies files from HDFS back to the local filesystem. To retrieveexample.txt, we can run the following command:

C:\hadoop-2.8.0\sbin>Hadoop fs -cat /input_dir/input_file.txt

Output: Hello world hello hi (which is stored in input_file .txt)

Step 4: Download the file:

Command: hadoop fs -get: Copies/Downloads files to the local file system Example:

hadoop fs -get /user/saurzcode/dir3/Samplefile.txt /home/

Step 5: Copy a file from source to destination

This command allows multiple sources as well in which case the destination must be a directory.

Command: hadoop fs -cp

Example: hadoop fs -cp /user/saurzcode/dir1/abc.txt /user/saurzcode/ dir2 **Step 6: Copy a file from/To Local file system to HDFS copyFromLocal**

Command: hadoop fs -copyFromLocal URI

Example: hadoop fs -copyFromLocal /home/saurzcode/abc.txt /user/ saurzcode/abc.txt

copyToLocal

Command: hadoop fs -copyToLocal [-ignorecrc] [-crc] URI

Step 7: Move file from source to destination

Note:- Moving files across filesystem is not permitted.

Command: hadoop fs -mv

Example: hadoop fs -mv /user/saurzcode/dir1/abc.txt /user/saurzcode/ dir2

Step 8: Deleting Files from HDFS

C:\hadoop-2.8.0\sbin>hadoop fs -rm input_file.txt /input_dir/input_file.txt

Recursive version of delete:

Command: hadoop fs -rmr

Example: hadoop fs -rmr /user/saurzcode/

Step 9: Display last few lines of a file

Similar to tail command in Unix.

Usage : hadoop fs -tail

Example: hadoop fs -tail /user/saurzcode/dir1/abc.txt

Step 10: Display the aggregate length of a file

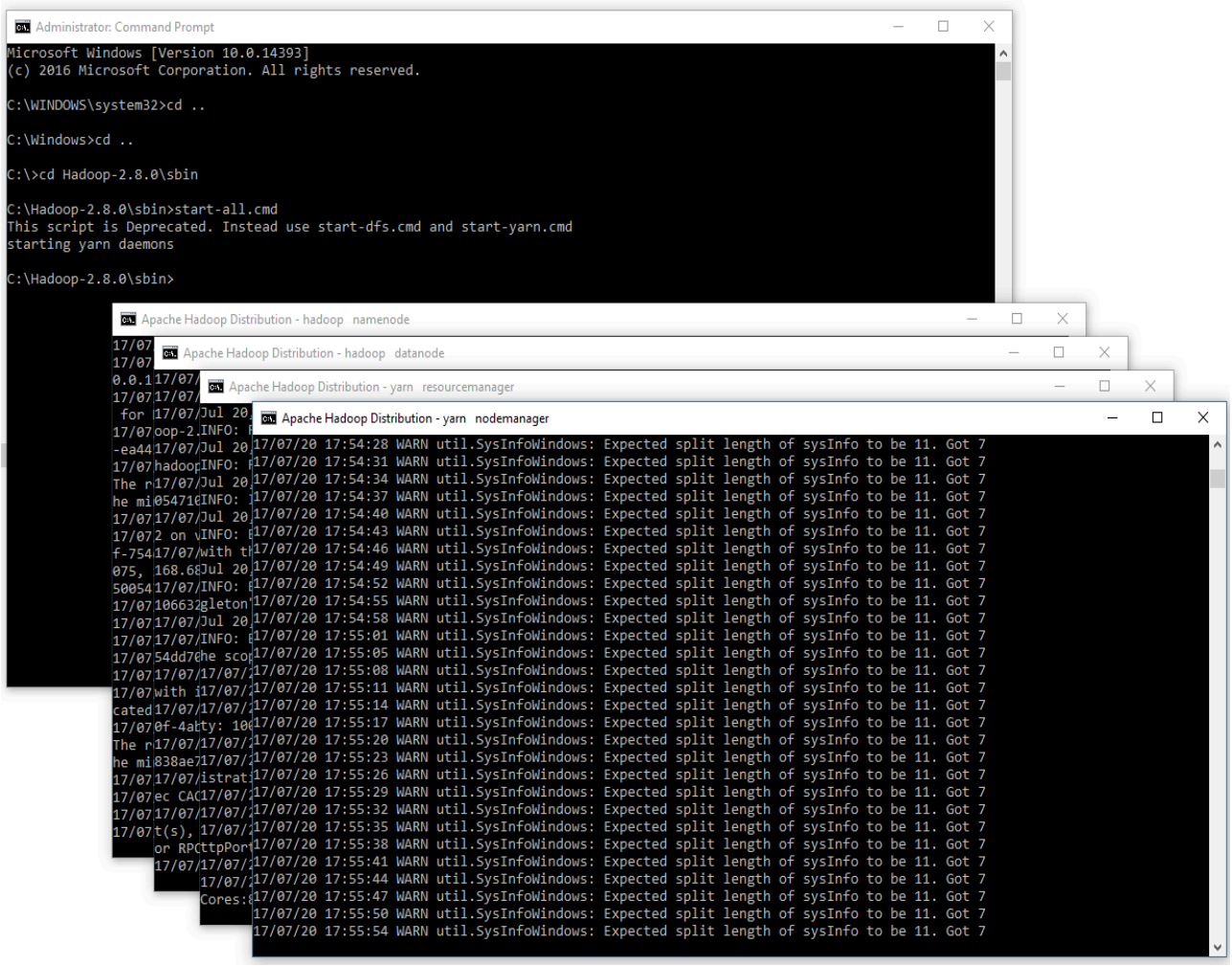
Command: hadoop fs -du

Example: hadoop fs -du /user/saurzcode/dir1/abc.txt

HADOOP OPERATION:

1. Open cmd in administrative mode and move to “C:/Hadoop-2.8.0/sbin” and start cluster

   Start-all.cmd



1. Create an input directory in

   HDFS. hadoop fs -mkdir

   /input_dir

2. Copy the input text file named input_file.txt in the input directory (input_dir)of HDFS.

   hadoop fs -put C:/input_file.txt /input_dir

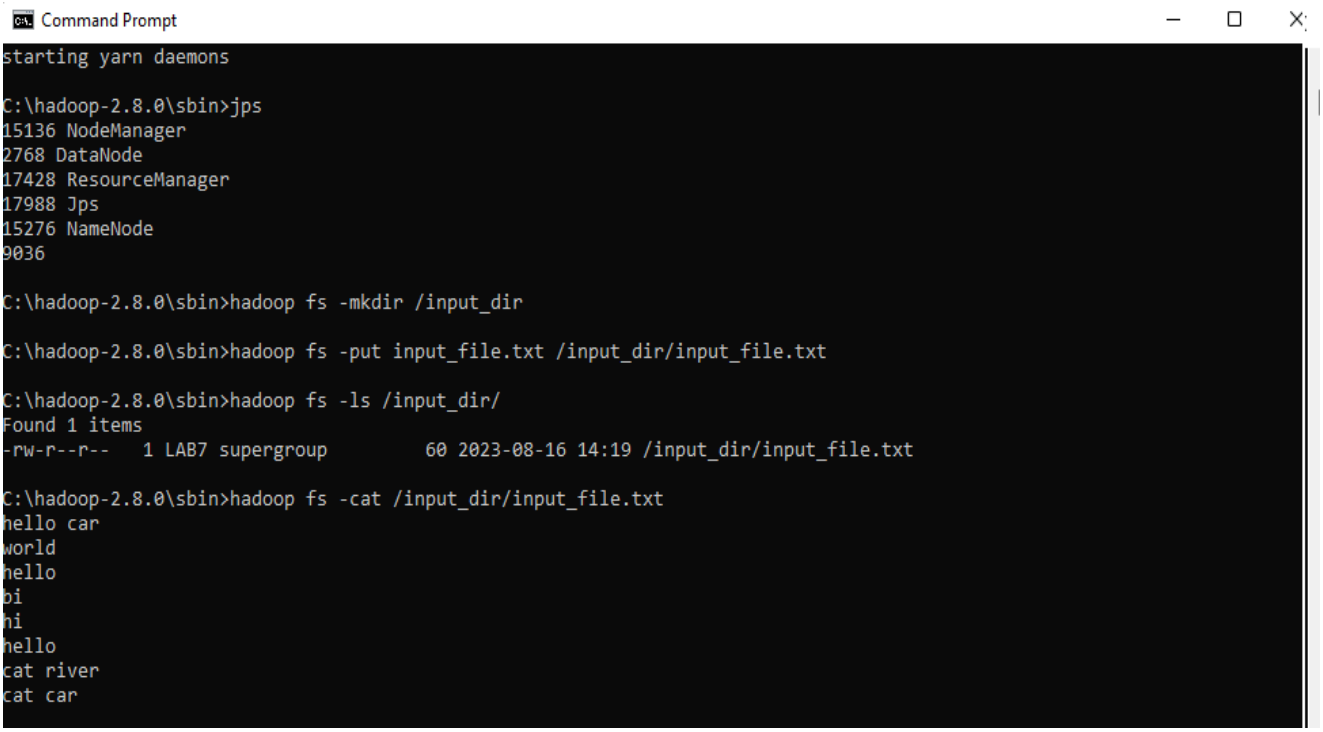
3. Verify input_file.txt available in HDFS input directory (input_dir).

   hadoop fs -ls /input_dir/

Verify content of the copied file.

hadoop dfs -cat /input_dir/input_file.txt

OUTPUT:



OTHER COMMANDS:

1. To leave Safe mode
   hadoop dfsadmin –safemode leave

2. To delete file from HDFS directory
   hadoop fs -rm -r /iutput_dir/input_file.txt

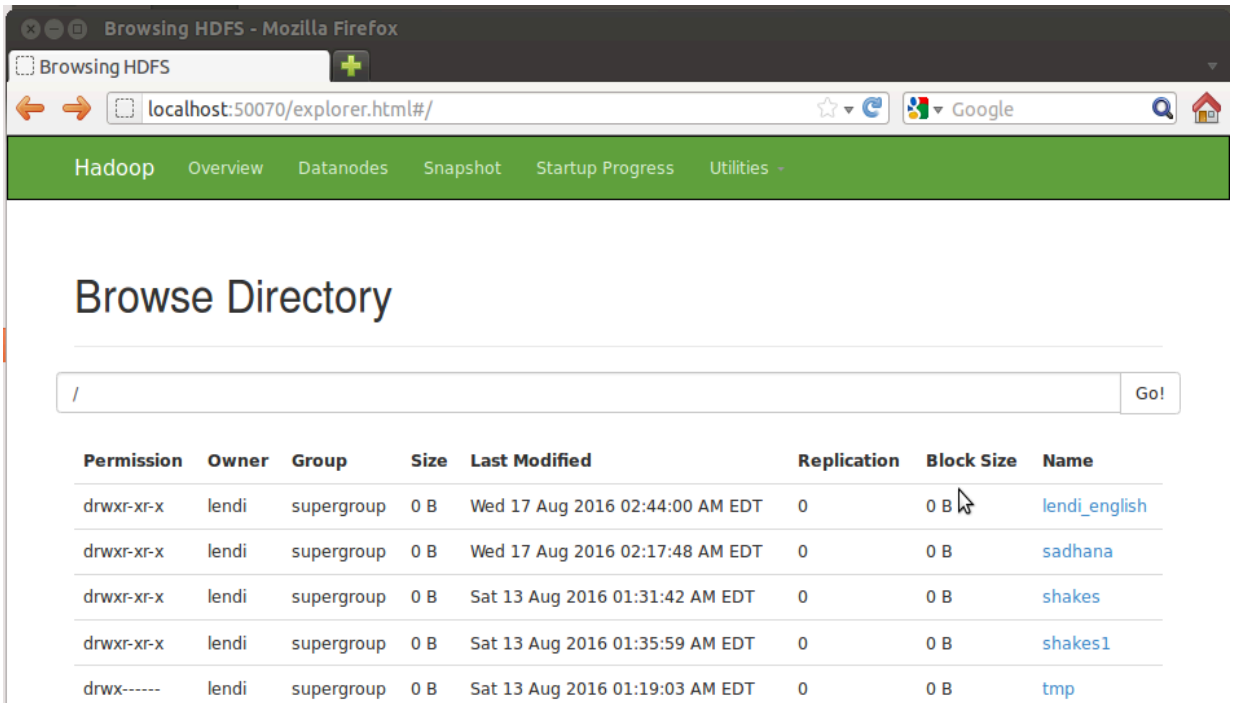
3. To delete directory from HDFS directory

   hadoop fs -rm -r /iutput_dir

OUTPUT

```
C:\>hadoop dfsadmin -safemode leave
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
Safe mode is OFF

C:\>hadoop fs -rm -r /input_dir/input_file.txt
Deleted /input_dir/input_file.txt




C:\>hadoop fs -rm -r /input_dir
Deleted /input_dir

C:\>
```

OUTPUT:



RESULT:

Thus, the implementation for file management in Hadoop was successfully executed.

EX.NO: 3 **MATRIX MULTIPLICATION**

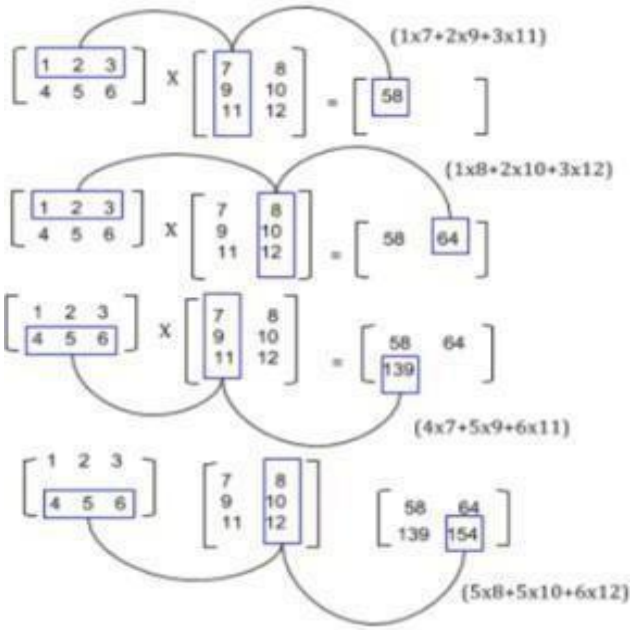
DATE:

AIM:

To Implement of Matrix Multiplication with Hadoop Map Reduce.

THEORY:

In mathematics, matrix multiplication or the matrix product is a binary operation that produces amatrix from two matrices. In more detail, if A is an n × m matrix and B is an m × p matrix, theirmatrix product AB is an n × p matrix, in which the m entries across a row of A are multiplied with the m entries down a column of B and summed to produce an entry of AB. When two linear transformations are represented by matrices, then the matrix product represents the composition of the two transformations.



ALGORITHM FOR MAP FUNCTION:

for each element mij of M do

produce (key,value) pairs as ((i,k), (M,j,mij), for k=1,2,3,.. upto the number of columns of N for each

element njk of N do

produce (key,value) pairs as ((i,k),(N,j,Njk), for i = 1,2,3,.. Upto the number of rows of M.

return Set of (key,value) pairs that each key (i,k), has list with values (M,j,mij) and (N, j,njk)for all possible values of j.

ALGORITHM FOR REDUCE FUNCTION:

for each key (i,k) do

sort values begin with M by j in listMsort values begin with N by j in listN

multiply mij and njk for jth value of each listsum

up mij x njk return (i,k), Σj=1 mij x njk

HADOOP OPERATION:

Make sure that Hadoop is installed on your system with java idk Steps to follow

Step 1: Open Eclipse> File > New > Java Project > (Name it – MRProgramsDemo) > Finish

Step 2: Right Click > New > Package (Name as **com.mapreduce.wc**) > Finish

Step 3: Right Click on Package > New > Class (Name it - Matrixmultiply) **Step 4:**

Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

1. C:/Hadoop/share/Hadoop->common/lib -> add all jar
2. C:/Hadoop/share/Hadoop-> client -> add all jar
3. C:/Hadoop/share/Hadoop-> mapreduce -> add all jar
4. C:/Hadoop/share/Hadoop-> yarn -> add all jar
4. C:/lib/hadoop-2.8.0/lib/Commons-cli-1.2.jar
5. C:/lib/hadoop-2.8.0/hadoop-core.jar

By Downloading the hadoop jar files with these links.

- Download Hadoop Common Jar files: https://goo.gl/G4MyHp
  hadoop-common-2.2.0.jar
- Download Hadoop Mapreduce Jar File: https://goo.gl/KT8yfB
  hadoop-mapreduce-client-core-2.7.1.jar

PROGRAM:

Creating Map file for Matrix Multiplication.

```
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
public class Map
 extends org.apache.hadoop.mapreduce.Mapper<LongWritable, Text, Text, Text> {
        @Override
        public void map(LongWritable key, Text value, Context context)
                        throws IOException, InterruptedException {
                Configuration conf = context.getConfiguration();
                int m = Integer.parseInt(conf.get("m"));
                int p = Integer.parseInt(conf.get("p"));
                String line = value.toString();              // (M, i, j, Mij);

                String[] indicesAndValue = line.split(",");
                Text outputKey = new Text();
```

```
                Text outputValue = new Text();
                if (indicesAndValue[0].equals("M")) {
                        for (int k = 0; k < p; k++) {
                                outputKey.set(indicesAndValue[1] + "," + k);
                                // outputKey.set(i,k);
                                outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2]
                                                + "," + indicesAndValue[3]);
                                // outputValue.set(M,j,Mij);
                                context.write(outputKey, outputValue);
                                        }
                                } else {
                                        // (N, j, k, Njk);
                                        for (int i = 0; i < m; i++) {
                                                outputKey.set(i + "," +
                                                indicesAndValue[2]);
                                                outputValue.set("N," +
                                                indicesAndValue[1] + ","
                                                                +
                                                indicesAndValue[3]);
                                                context.write(outputKey,
                                                outputValue);
                        }
                }
        }
}
```

Creating Reduce file for Matrix Multiplication.

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.HashMap;

public class Reduce
  extends org.apache.hadoop.mapreduce.Reducer<Text, Text, Text, Text> {
        @Override
        public void reduce(Text key, Iterable<Text> values, Context context)
                        throws IOException, InterruptedException {
                String[] value;
                //key=(i,k),
                //Values = [(M/N,j,V/W),..]
                HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
                HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
                for (Text val : values) {
                        value = val.toString().split(",");
                        if (value[0].equals("M")) {
                                hashA.put(Integer.parseInt(value[1]),   Float.parseFloat(value[2]));
                        } else {
```

```
}
        hashB.put(Integer.parseInt(value[1]),  Float.parseFloat(value[2])
}
int n = Integer.parseInt(context.getConfiguration().get("n"));
 float result = 0.0f;
float m_ij;
float n_jk;
for (int j = 0; j < n; j++) {
        m_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;
        n_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
        result += m_ij * n_jk;
}
if (result != 0.0f) {
        context.write(null,new Text(key.toString() + "," + Float.toString(result)));
                }
        }
}
```

Creating MatrixMultiply.java file for

```
import org.apache.hadoop.conf.*;
 import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class MatrixMultiply {
public static void main(String[] args) throws Exception {
        if (args.length != 2) {
        System.err.println("Usage: MatrixMultiply <in_dir> <out_dir>");
         System.exit(2);
}
Configuration conf = new Configuration();
// M is an m-by-n matrix;
N is an n-by-p matrix. conf.set("m", "1000");
conf.set("n", "100");
```

```
conf.set("p", "1000");
@SuppressWarnings("deprecation")
Job job = new Job(conf, "MatrixMultiply");
job.setJarByClass(MatrixMultiply.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.waitForCompletion(true);
        }
}
```

Step 5: Uploading the M, N file which contains the matrix multiplication data to HDFS.

Create M.txt in sbin

M,0,0,1
M,0,1,2
M,1,0,3
M,1,1,4

Create N.txt in sbin

N,0,0,5 1,6 N,1,0,7
N,1,1,8

Run following commands in cmd prompt:

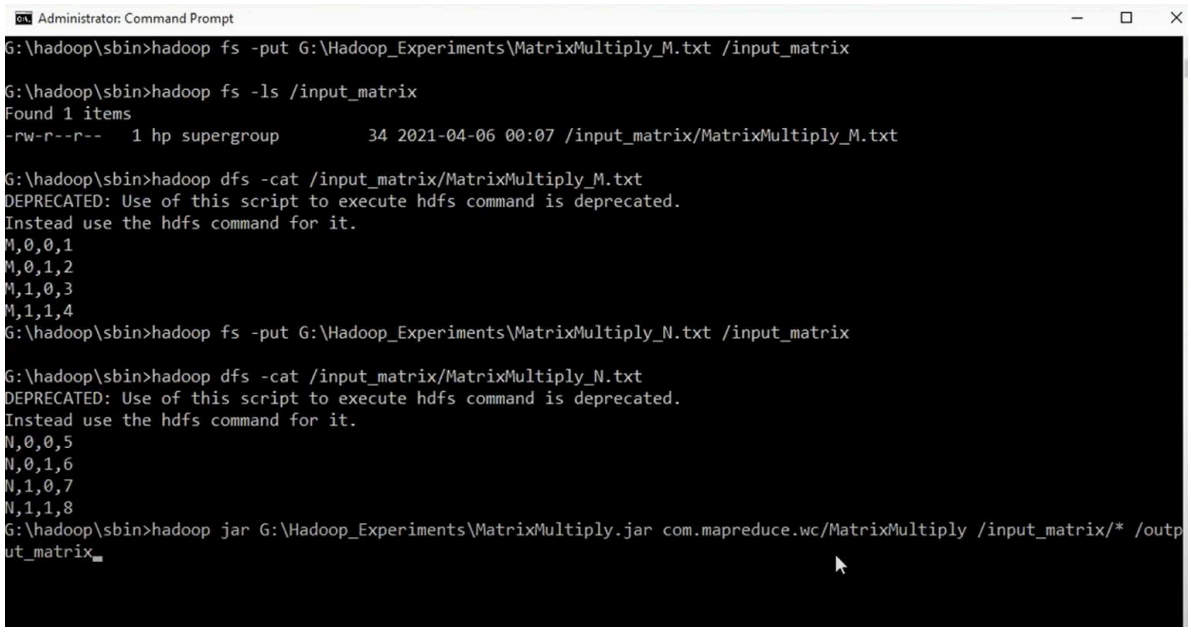
```
$ hadoop fs -mkdir /input_matrix/
$ hadoop fs -put M.txt / input_matrix
$ hadoop fs -put N.txt / input_matrix
$ hadoop fs -cat N.txt / input_matrix / M.txt
$ hadoop fs -cat N.txt / input_matrix / N.txt
```

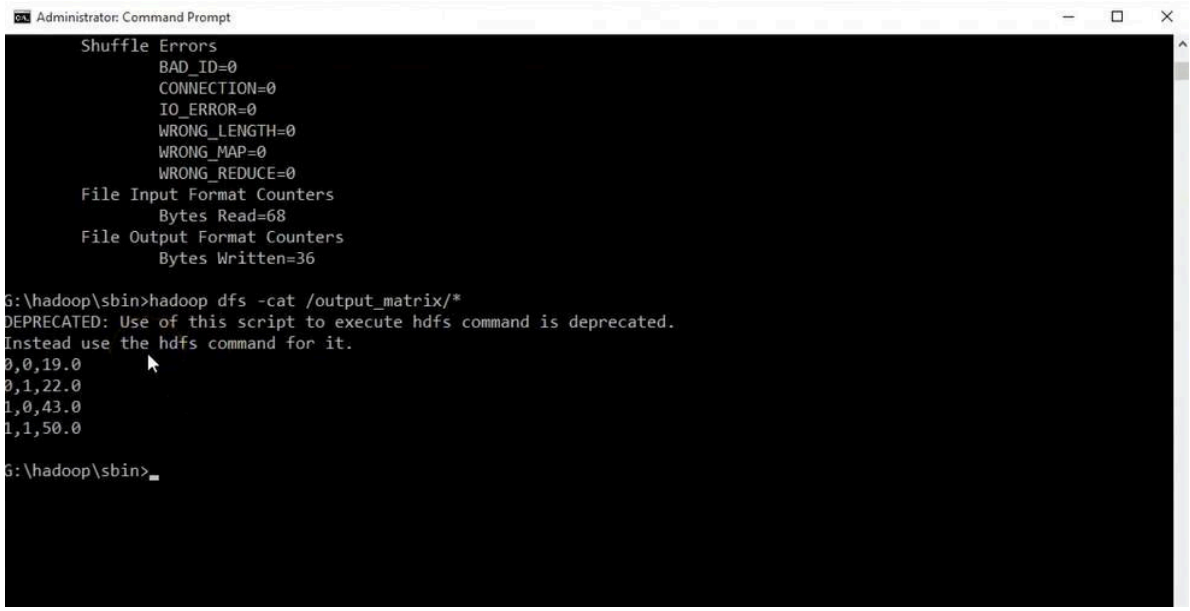
Step 6: Export jar and run following hadoop command.

Step 1: Open Eclipse> open -> (MRProgramsDemo)project -> Right Click -> Export->java->JAR file
-> Next -> name it as matrix.jar Step 2: Open Command prompt

Sbin >hadoop jar C:/MRProgramsDemo\matrix com.mapreduce.wc/matrixmultiply /input_matrix/* /output_matrix

OUTPUT:


Administrator: Command Prompt
G:\hadoop\sbin>hadoop fs -put G:\Hadoop_Experiments\MatrixMultiply_M.txt /input_matrix
G:\hadoop\sbin>hadoop fs -ls /input_matrix
Found 1 items
-rw-r--r--   1 hp supergroup   34 2021-04-06 00:07 /input_matrix/MatrixMultiply_M.txt
G:\hadoop\sbin>hadoop dfs -cat /input_matrix/MatrixMultiply_M.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
M,0,0,1
M,0,1,2
M,1,0,3
M,1,1,4
G:\hadoop\sbin>hadoop fs -put G:\Hadoop_Experiments\MatrixMultiply_N.txt /input_matrix
G:\hadoop\sbin>hadoop dfs -cat /input_matrix/MatrixMultiply_N.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
N,0,0,5
N,0,1,6
N,1,0,7
N,1,1,8
G:\hadoop\sbin>hadoop jar G:\Hadoop_Experiments\MatrixMultiply.jar com.mapreduce.wc/MatrixMultiply /input_matrix/* /outp
ut_matrix


Administrator: Command Prompt
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=68
File Output Format Counters
Bytes Written=36
G:\hadoop\sbin>hadoop dfs -cat /output_matrix/*
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
0,0,19.0
0,1,22.0
1,0,43.0
1,1,50.0
G:\hadoop\sbin>

RESULT:

Thus, a Implementation of Matrix Multiplication with Hadoop Map Reduce was executed successfully.

EX.NO: 4 **WORD COUNT MAP REDUCE**

DATE:

AIM:

To Run a basic Word Count MapReduce program to understand Map Reduce Paradigm.

THEORY:

MapReduce is a programming model used for efficient processing in parallel over large data-sets in a distributed manner. The data is first split and then combined to produce the final result. The libraries for MapReduce is written in so many programming languages with various different-different optimizations.

Workflow of MapReduce consists of 5 steps:

1. Splitting – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. Mapping – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).
3. Intermediate splitting – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar KEY data should be on same cluster.
4. Reduce – it is nothing but mostly group by phase
5. Combining – The last phase where all the data (individual result set from each cluster) is combined together to form a Result.

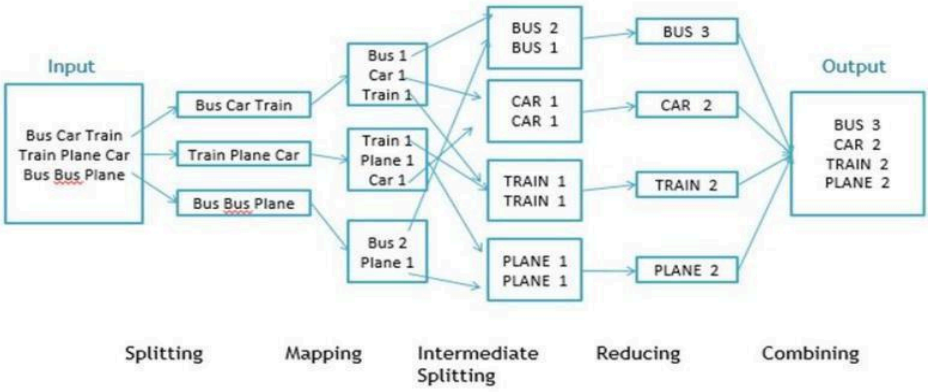


Fig. WorkFlow of MapReducing

PREPARE:

1. Download MapReduceClient.jar

   (Link: https://github.com/MuhammadBilalYar/HADOOP- INSTALLATION-ON-WINDOW-10/blob/master/MapReduceClient.jar)

2. Download Input_file.txt

   (Link: https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob/master/input_file.txt)

Place both files in "C:/"

Make sure that Hadoop is installed on your system with java jdk Steps to follow

Step 1: Open Eclipse> File > New > Java Project > (Name it – MRProgramsDemo) >Finish

Step 2: Right Click > New > Package (Name it - PackageDemo) > Finish **Step 3:** Right Click on Package > New > Class (Name it - WordCount) **Step 4:** Add Following Reference Libraries –

Right Click on Project > Build Path> Add External Archivals

- /usr/lib/hadoop-0.20/hadoop-core.jar
- Usr/lib/hadoop-0.20/lib/Commons-cli-1.2.jar

PROGRAM:

```
package PackageDemo;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new Path(files[1]);
```

```
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text, Text,IntWritable>
{
public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException
{
String line = value.toString();
String[] words=line.split(",");for(String word: words )
{
Text outputKey = new Text(word.toUpperCase().trim());
IntWritable outputValue = new IntWritable(1);
con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throwsIOException, InterruptedException
{
int sum = 0;
for(IntWritable value : values)
{
sum += value.get();
}
con.write(word, new IntWritable(sum));
}
}
}
```

Make Jar File

Right Click on Project> Export> Select export destination as Jar File > next> Finish

To Move this into Hadoop directly,

open the terminal and enter the followingcommands:

```
$ hadoop fs -put wordcountFile /input_dir
```

Run Jar file

Syntax: Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFile

PathToOutputDirectry

sbin> Hadoop jar MRProgramsDemo.jar PackageDemo.WordCount /input_dir /out

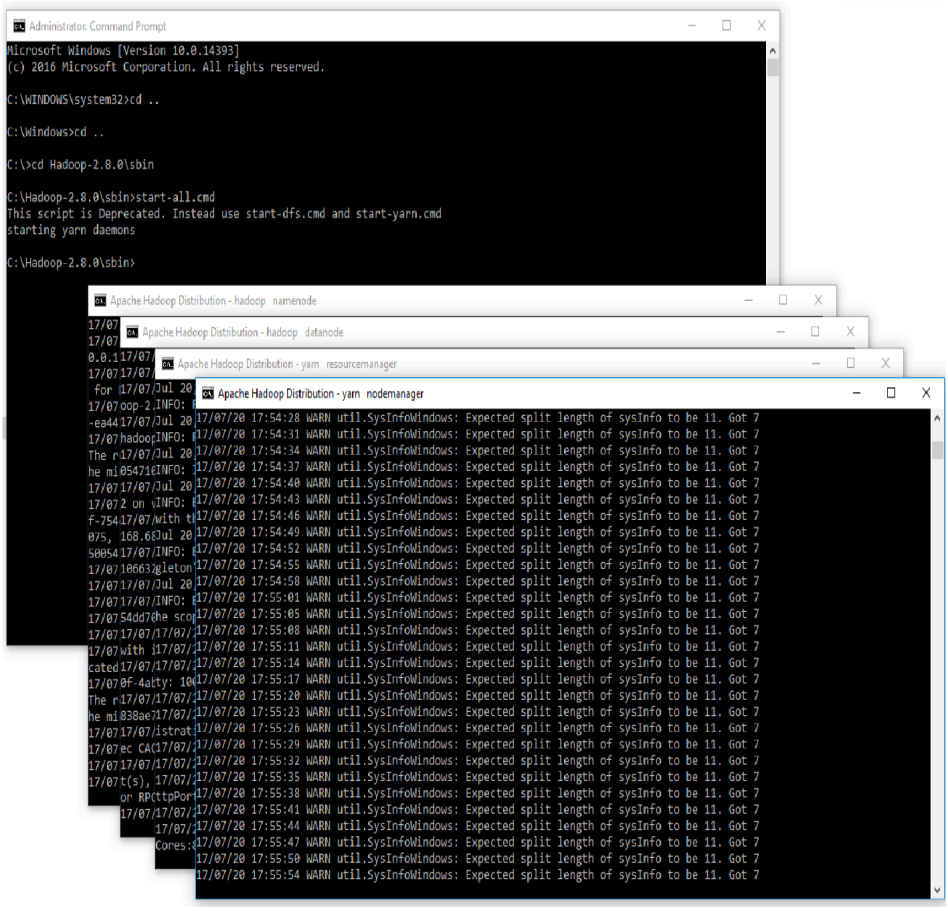
OUTPUT:

$ hadoop fs -ls out
1Found 1 item
-rw-r--r-- 1 training supergroup
20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000
$ hadoop fs -cat out/*
CAR 4
TRAIN 6

HADOOP OPERATION:

1. Open cmd in Administrative mode and move to "C:/Hadoop-2.8.0/sbin" andstart cluster

   Start-all.cmd



2. Create an input directory in HDFS.

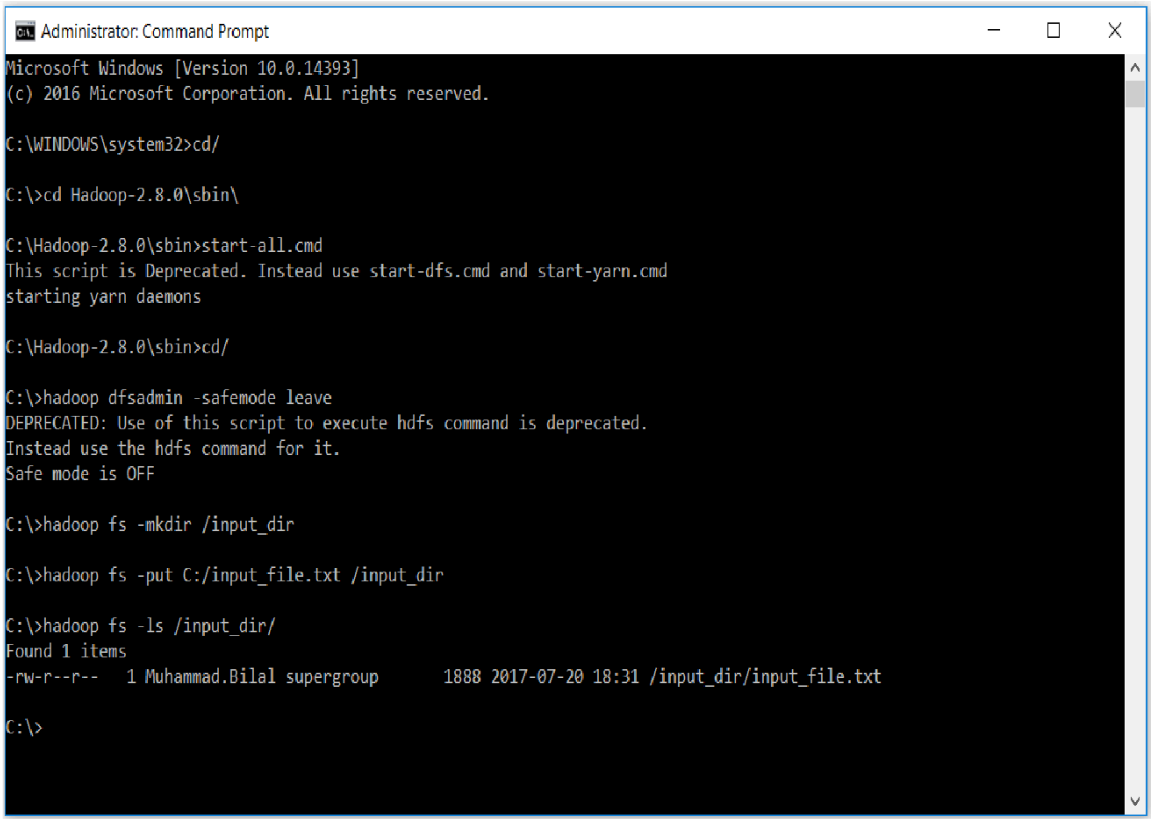
hadoop fs -mkdir /input_dir

3. Copy the input text file named input_file.txt in the input directory (input_dir)of HDFS.

hadoop fs -put C:/input_file.txt /input_dir/input_file.txt

4. Verify input_file.txt available in HDFS input directory .

hadoop fs -ls /input_dir/



5. Verify content of the copied file.

hadoop dfs -cat /input_dir/input_file.txt

6. Run MapReduceClient.jar and also provide input and out directories.

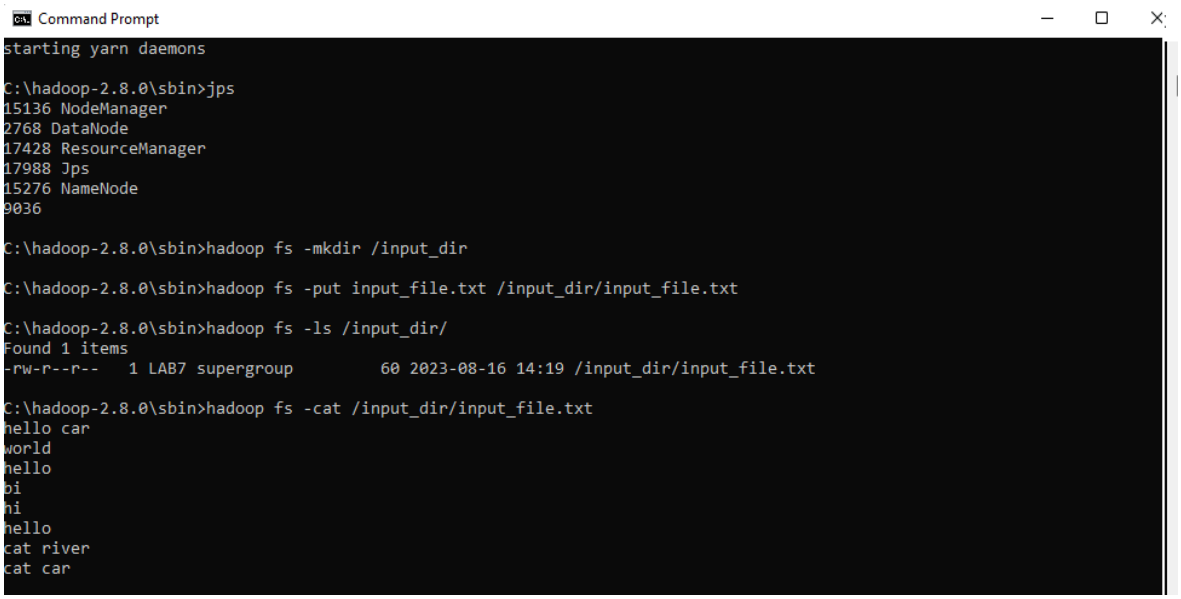
>hadoop jar C:/hadoop-2.8.0/share/Hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar

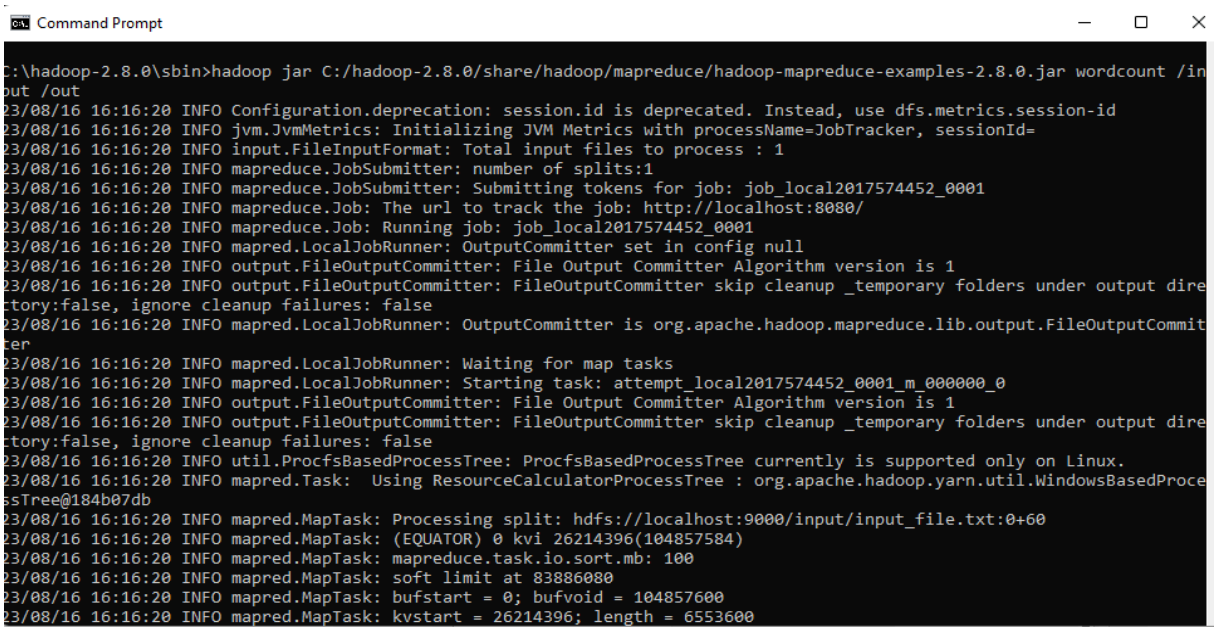
>hadoop jar C:/hadoop-2.8.0/share/Hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar wordcount /input_dir /out

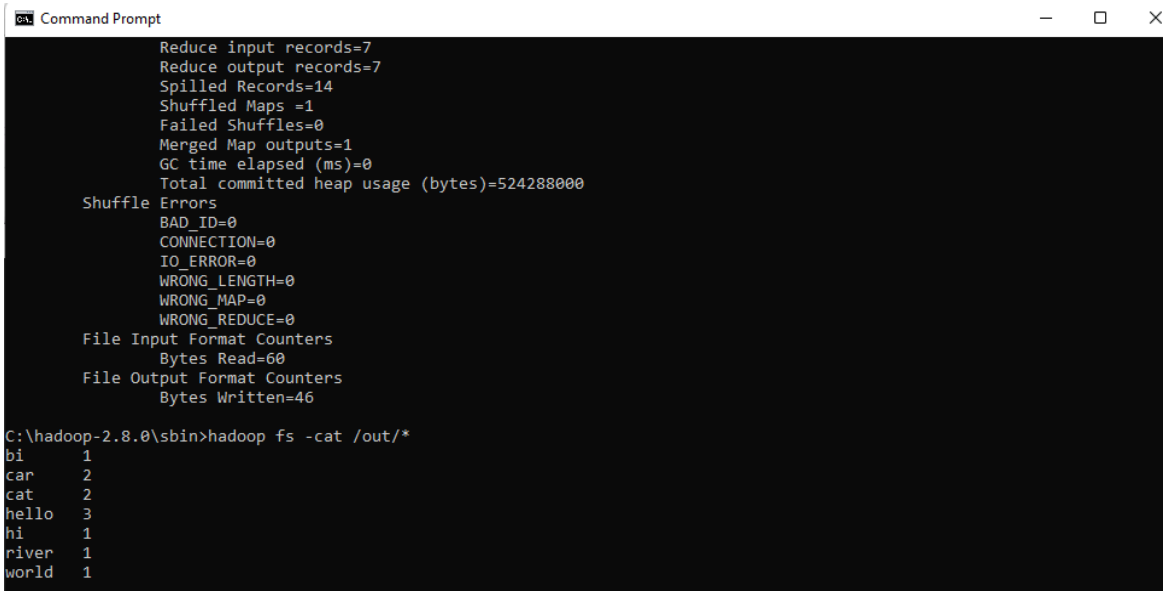
7. Verify content for generated output file.

hadoop dfs -cat /out/*

OUTPUT


Command Prompt
starting yarn daemons
C:\hadoop-2.8.0\sbin>jps
15136 NodeManager
2768 DataNode
17428 ResourceManager
17988 Jps
15276 NameNode
9036
C:\hadoop-2.8.0\sbin>hadoop fs -mkdir /input_dir
C:\hadoop-2.8.0\sbin>hadoop fs -put input_file.txt /input_dir/input_file.txt
C:\hadoop-2.8.0\sbin>hadoop fs -ls /input_dir/
Found 1 items
-rw-r--r--   1 LAB7 supergroup   60 2023-08-16 14:19 /input_dir/input_file.txt
C:\hadoop-2.8.0\sbin>hadoop fs -cat /input_dir/input_file.txt
hello car
world
hello
bi
hi
hello
cat river
cat car


Command Prompt
C:\hadoop-2.8.0\sbin>hadoop jar C:/hadoop-2.8.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.0.jar wordcount /in
out /out
23/08/16 16:16:20 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
23/08/16 16:16:20 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
23/08/16 16:16:20 INFO input.FileInputFormat: Total input files to process : 1
23/08/16 16:16:20 INFO mapreduce.JobSubmitter: number of splits:1
23/08/16 16:16:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2017574452_0001
23/08/16 16:16:20 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
23/08/16 16:16:20 INFO mapreduce.Job: Running job: job_local2017574452_0001
23/08/16 16:16:20 INFO mapred.LocalJobRunner: OutputCommitter set in config null
23/08/16 16:16:20 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/08/16 16:16:20 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output dire
ctory:false, ignore cleanup failures: false
23/08/16 16:16:20 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommit
ter
23/08/16 16:16:20 INFO mapred.LocalJobRunner: Waiting for map tasks
23/08/16 16:16:20 INFO mapred.LocalJobRunner: Starting task: attempt_local2017574452_0001_m_000000_0
23/08/16 16:16:20 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
23/08/16 16:16:20 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output dire
ctory:false, ignore cleanup failures: false
23/08/16 16:16:20 INFO util.ProcfsBasedProcessTree: ProcfsBasedProcessTree currently is supported only on Linux.
23/08/16 16:16:20 INFO mapred.Task: Using ResourceCalculatorProcessTree : org.apache.hadoop.yarn.util.WindowsBasedProce
ssTree@184b07db
23/08/16 16:16:20 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/input/input_file.txt:0+60
23/08/16 16:16:20 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
23/08/16 16:16:20 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
23/08/16 16:16:20 INFO mapred.MapTask: soft limit at 83886080
23/08/16 16:16:20 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
23/08/16 16:16:20 INFO mapred.MapTask: kvstart = 26214396; length = 6553600


Command Prompt
Reduce input records=7
Reduce output records=7
Spilled Records=14
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=0
Total committed heap usage (bytes)=524288000
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=60
File Output Format Counters
Bytes Written=46
C:\hadoop-2.8.0\sbin>hadoop fs -cat /out/*
bi 1
car 2
cat 2
hello 3
hi 1
river 1
world 1

RESULT:

Thus, the implementation of Wordcount with Hadoop Map Reduce was executed successfully.

EX.NO: 5 **HIVE**

DATE:

AIM:

To Installation of Hive along with practice examples.

THEORY:

Apache Hive is a data warehouse and an ETL tool which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop. It is built on top of Hadoop. It is a software project that provides data query and analysis. It facilitates reading, writing and handling wide datasets that stored in distributed storage and queried by Structure Query Language (SQL) syntax.

PREPARE:

These softwares should be prepared to install Hadoop 2.8.0 on window 10 64bit

1. Download Hadoop 2.8.0
2. Java JDK 1.8.0.zip
3. **Download Hive 2.1.0 :** https://archive.apache.org/dist/hive/hive-2.1.0/
4. **Download Derby Metastore 10.12.1.1:** https://archive.apache.org/dist/db/derby/db-derby-10.12.1.1/
5. **Download hive-site.xml :**

   https://drive.google.com/file/d/1qqAo7RQfr5Q6O-GTom6Rji3TdufP81zd/view?usp=sharing

PROCEDURE:

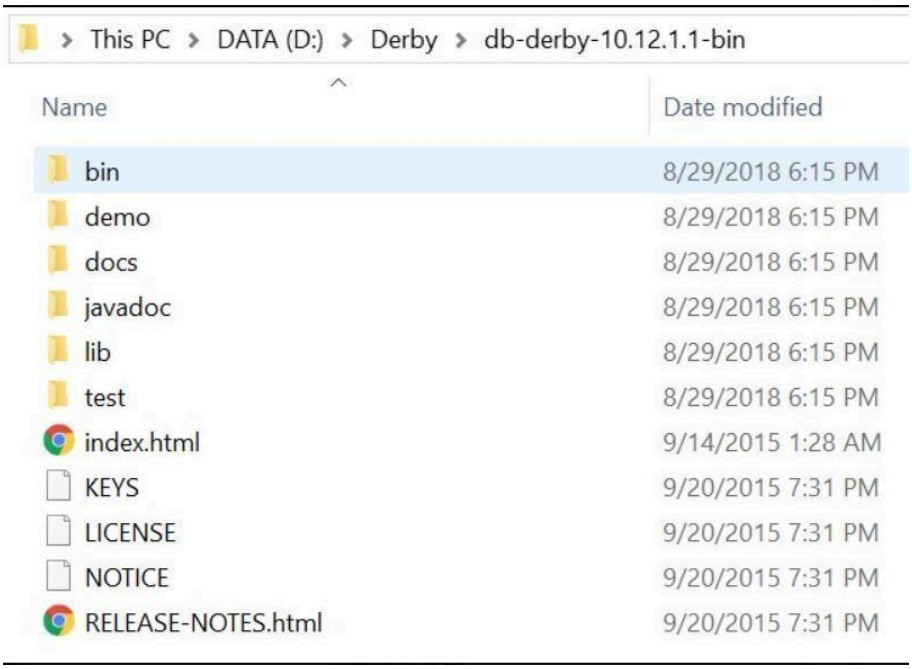
STEP - 1: Download and Extract the Hive file:

[1] Extract file apache-hive-2.1.0-bin.tar.gz and place under "D:\Hive", you can use any preferred location

[2] Copy the leaf folder “apache-hive-2.1.0-bin” and move to the root folder "D:\Hive.

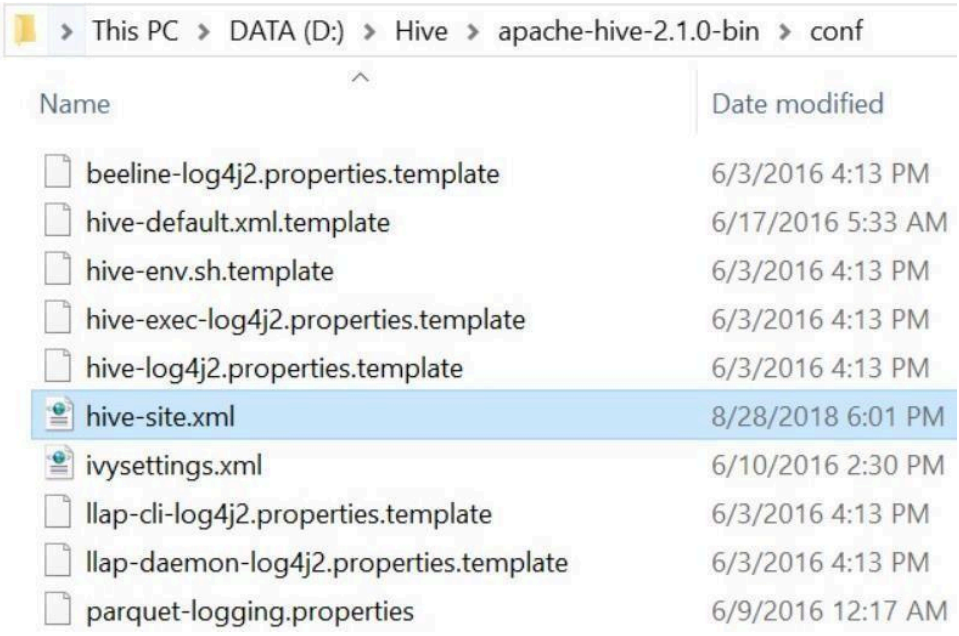
STEP - 2: Extract the Derby file

Similar to Hive, extract file db-derby-10.12.1.1-bin.tar.gz and place under "D:\Derby", you can use any preferred location:



STEP - 3: Moving hive-site.xml file

Drop the downloaded file “hive-site.xml” to hive configuration location “D:\Hive\apache-hive-2.1.0-bin\conf”.

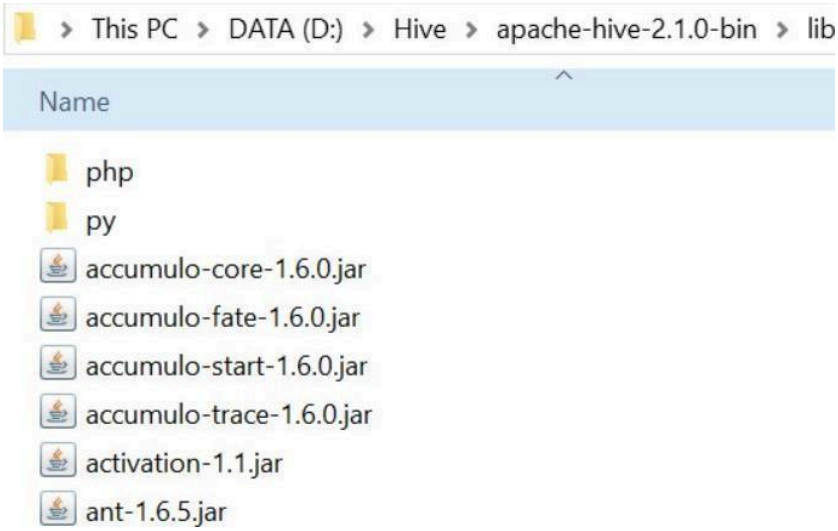


STEP - 4: Moving Derby libraries

Next, need to drop all derby library to hive library location :

[1] Move to library folder under derby location D:\Derby\db-derby-10.12.1.1-bin\lib.

2] Select all , copy and paste all libraries from derby to hive location D:\Hive\apache-hive-2.1.0-bin\lib.

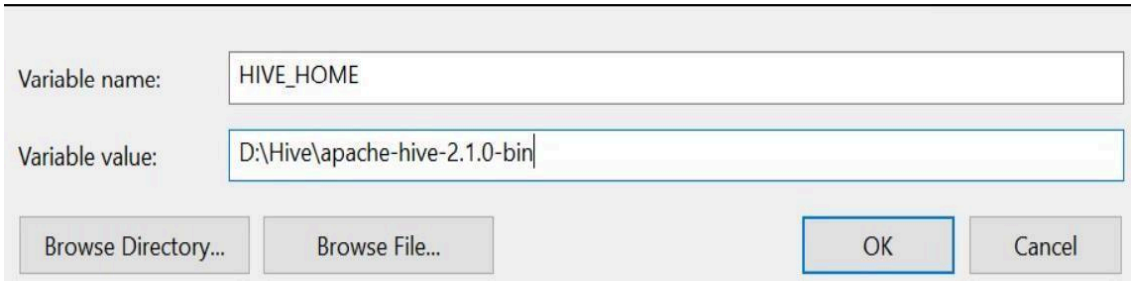


STEP - 5: Configure Environment variables

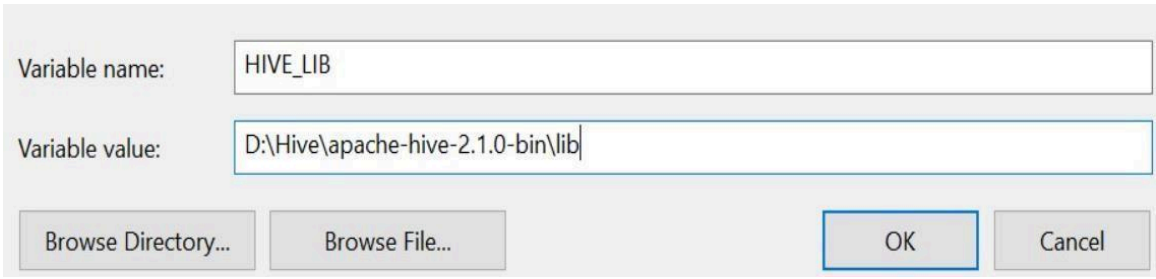
Set the path for the following Environment variables (User Variables) on windows 10 –

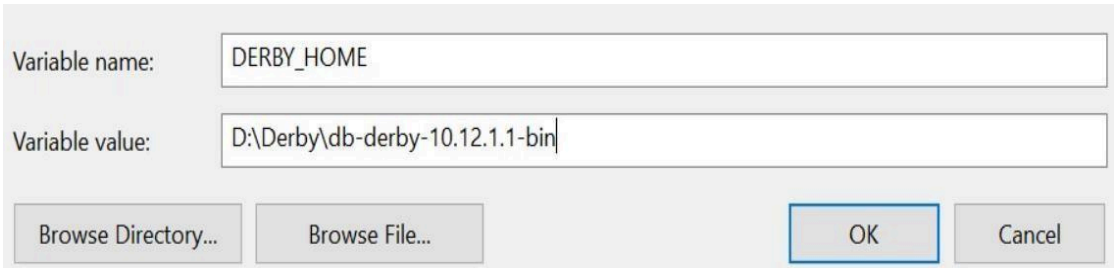
- HIVE_HOME - D:\Hive\apache-hive-2.1.0-bin
- HIVE_BIN - D:\Hive\apache-hive-2.1.0-bin\bin
- HIVE_LIB - D:\Hive\apache-hive-2.1.0-bin\lib
- DERBY_HOME - D:\Derby\db-derby-10.12.1.1-bin
- HADOOP_USER_CLASSPATH_FIRST - true

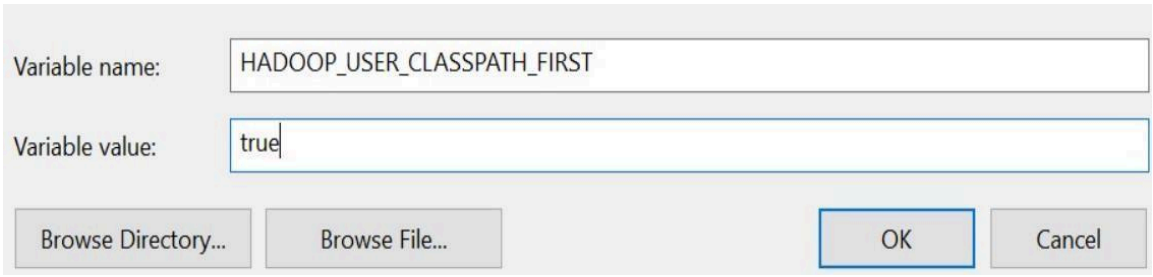
This PC - > Right Click - > Properties - > Advanced System Settings - > Advanced - > Environment Variables


Variable name:
HIVE_HOME
Variable value:
D:\Hive\apache-hive-2.1.0-bin
Browse Directory...
Browse File...
OK
Cancel


Variable name:
HIVE_BIN
Variable value:
D:\Hive\apache-hive-2.1.0-bin\bin
Browse Directory...
Browse File...
OK
Cancel


Variable name:
HIVE_LIB
Variable value:
D:\Hive\apache-hive-2.1.0-bin\lib
Browse Directory...
Browse File...
OK
Cancel


Variable name:
DERBY_HOME
Variable value:
D:\Derby\db-derby-10.12.1.1-bin
Browse Directory...
Browse File...
OK
Cancel


Variable name:
HADOOP_USER_CLASSPATH_FIRST
Variable value:
true
Browse Directory...
Browse File...
OK
Cancel

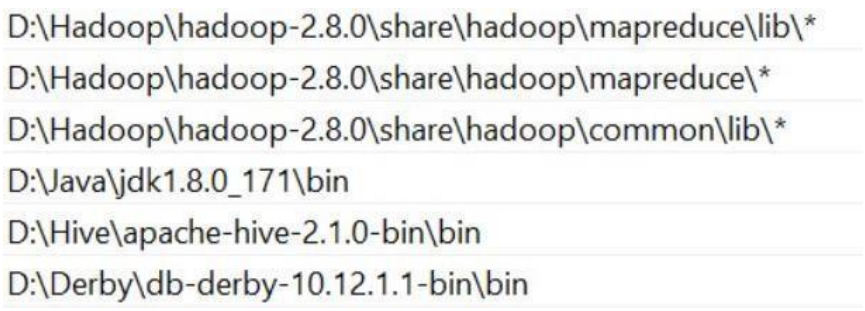
STEP - 6: Configure System variables

Next onward need to set System variables, including Hive bin directory path: HADOOP_USER_CLASSPATH_FIRST - true

Variable: Path

Value:

1. D:\Hive\apache-hive-2.1.0-bin\bin
2. D:\Derby\db-derby-10.12.1.1-bin\bin



STEP - 7: Working with hive-site.xml

Download hive-site.xml and paste it in D:/Hive/apache-hive-2.1.0-bin/conf/hive-site.xml

- hive-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration><property>  <name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:derby://localhost:1527/metastore_db;create=true</value>
<description>JDBC connect string for a JDBC metastore</description>
</property><property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>org.apache.derby.jdbc.ClientDriver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>hive.server2.enable.impersonation</name>
<description>Enable user impersonation for HiveServer2</description>
<value>true</value>
</property>
<property>
```

<name>hive.server2.authentication</name>

<value>NONE</value>

<description> Client authentication types. NONE: no authentication check LDAP: LDAP/AD based authentication KERBEROS: Kerberos/GSSAPI authentication CUSTOM: Custom authentication provider (Use with property hive.server2.custom.authentication.class) </description>

</property>

<property>

<name>datanucleus.autoCreateTables</name>

<value>True</value>

</property>

</configuration>

STEP - 8: Start the Hadoop

Here need to start Hadoop first -

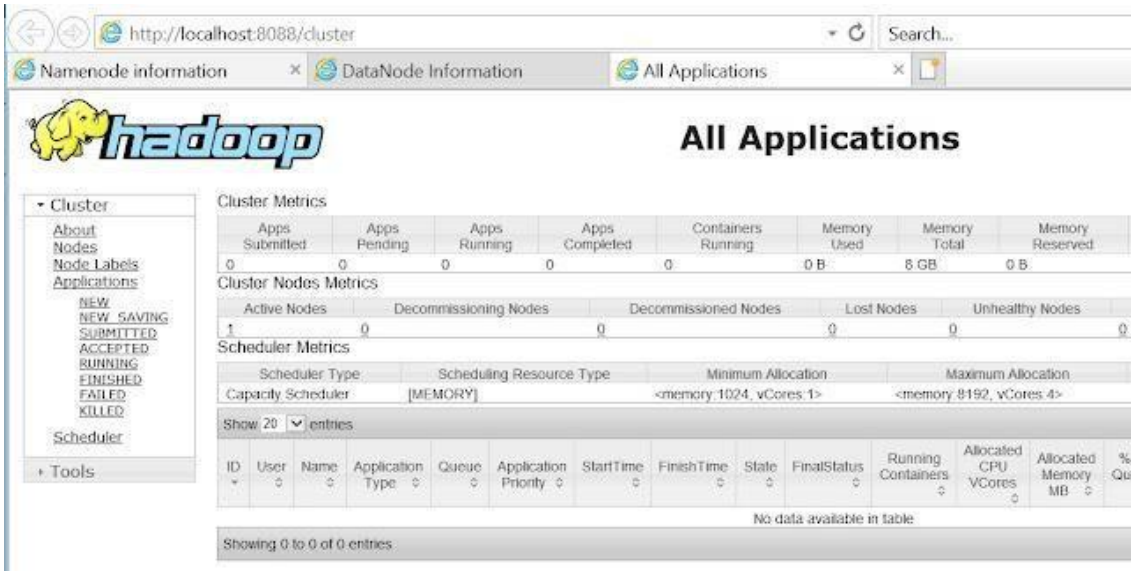
Open command prompt and change directory to “D:\Hadoop\hadoop-2.8.0\sbin" and type "**start-all.cmd**" to start apache.

It will open four instances of cmd for following tasks –

- Hadoop Datanaode
- Hadoop Namenode
- Yarn Nodemanager
- Yarn Resourcemanager

It can be verified via browser also as –

- Namenode (hdfs) - http://localhost:50070
- Datanode - http://localhost:50075
- All Applications (cluster) - http://localhost:8088 etc.



Since the 'start-all.cmd' command has been deprecated so you can use below command in order wise -

- "start-dfs.cmd" and
- "start-yarn.cmd"

STEP - 9: Start Derby server

Post successful execution of Hadoop, change directory to

>cd "D:\Derby\db-derby-10.12.1.1-bin\bin"

and type "**startNetworkServer -h 0.0.0.0**" to start derby server.

```
D:\Derby\db-derby-10.12.1.1-bin\bin>startNetworkServer -h 0.0.0.0
```

OUTPUT:

```
D:\Derby\db-derby-10.12.1.1-bin\bin>startNetworkServer -h 0.0.0.0
Thu Aug 30 10:40:27 IST 2018 : Security manager installed using the Basic server security policy.

Thu Aug 30 10:40:28 IST 2018 : Apache Derby Network Server - 10.12.1.1 - (1704137) started and re
ady to accept connections on port 1527
```

STEP - 10: Start the Hive

Derby server has been started and ready to accept connection so open a new command prompt under administrator privileges and move to hive directory as "

> cd D:\Hive\apache-hive-2.1.0-bin\bin" –

[1] Type "jps -m" to check NetworkServerControl

```
D:\Hive\apache-hive-2.1.0-bin\bin>jps -m
384 NameNode
22100 NetworkServerControl start -h 0.0.0.0
17960 Jps -m
19944 NodeManager
4776 ResourceManager
23468 DataNode
```

[2] Type "hive" to execute hive server.

```
D:\Hive\apache-hive-2.1.0-bin\bin>hive
```

OUTPUT:

```
D:\Hive\apache-hive-2.1.0-bin\bin>hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/Hive/apache-hive-2.1.0-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl
/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/D:/Hadoop/hadoop-2.8.0/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the
console.
Connecting to jdbc:hive2://
Connected to: Apache Hive (version 2.1.0)
Driver: Hive JDBC (version 2.1.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.1.0 by Apache Hive
hive>
```

Hive installed Successfully!

PROGRAM:

HIVE QUARIES AND OUTPUT:

[1] Create Database in Hive -

hive>CREATE DATABASE IF NOT EXISTS TRAINING;

```
hive> CREATE DATABASE IF NOT EXISTS TRAINING;
OK
No rows affected (1.603 seconds)
hive>
```

[2] Show Database -

hive>SHOW DATABASES;

```
hive> SHOW DATABASES;
OK
default
training
2 rows selected (0.382 seconds)
hive>
```

[3] Creating Hive Tables -

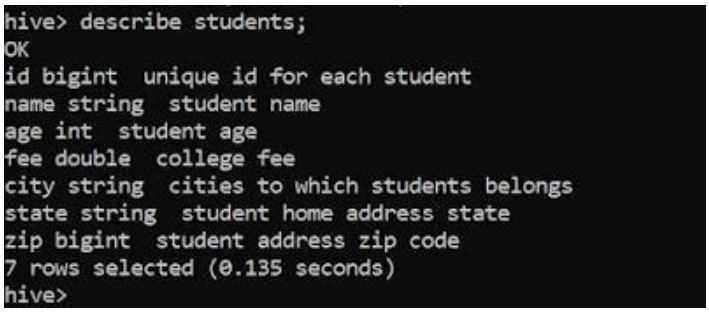
hive>CREATE TABLE IF NOT EXISTS testhive(col1 char(10), col2 char(20));

```
hive> USE TRAINING;
OK
No rows affected (0.047 seconds)
hive> CREATE TABLE IF NOT EXISTS testhive(col1 char(10),col2 char(20));
OK
No rows affected (0.669 seconds)
hive>
```

```
hive> CREATE TABLE IF NOT EXISTS students(
. . > id BIGINT COMMENT 'unique id for each student',
. . > name STRING COMMENT 'student name',
. . > age INT COMMENT 'student age',
. . > fee DOUBLE COMMENT 'college fee',
. . > city STRING COMMENT 'cities to which students belongs',
. . > state STRING COMMENT 'student home address state',
. . > zip BIGINT COMMENT 'student address zip code')
. . > COMMENT 'this table holds the demography info for each student'
. . > ROW FORMAT DELIMITED
. . > FIELDS TERMINATED BY '|'
. . > LINES TERMINATED BY '\n'
. . > STORED AS TEXTFILE
. . > LOCATION '/user/hive/warehouse/training.db/students';
OK
No rows affected (0.327 seconds)
hive>
```

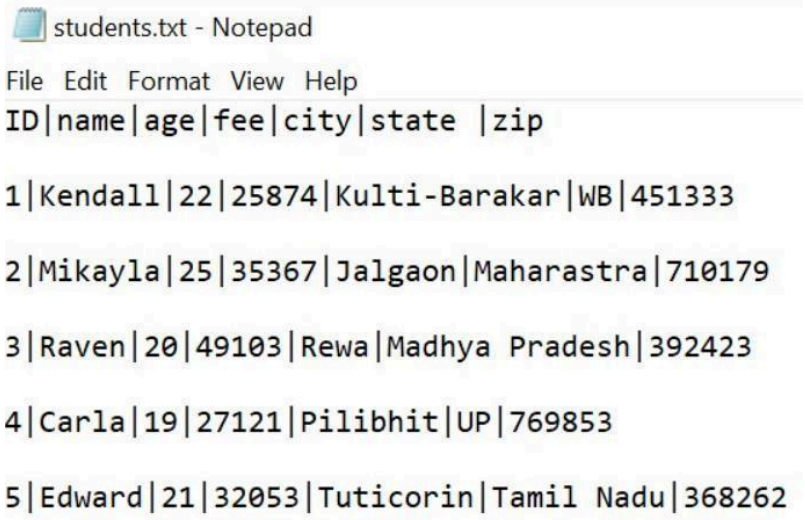
[4] DESCRIBE Table Command in Hive -

hive>describe students

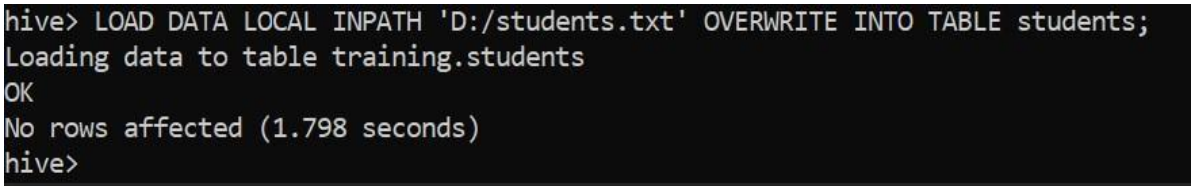


[5] LOAD Command for Inserting Data Into Hive Tables

Create a sample text file using '|' delimiter :



hive>LOAD DATA LOCAL INPATH "D:/students.txt" OVERWRITE INTO TABLE STUDENTS;



[6] Retrieve Data from Table -

hive>SELECT * FROM STUDENTS;

[7] Create another Table -

The following query creates a table named **employee** :

hive> CREATE TABLE IF NOT EXISTS employee (eid int, name String,salary String, destination String)

COMMENT ‘Employee details’

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ‘\t’

LINES TERMINATED BY ‘\n’

STORED AS TEXTFILE; **OUTPUT:**

OK
Time taken: 5.905 seconds
hive>

[8] Alter

Table - Syntax:

The statement takes any of the following syntaxes based on what attributes we wish to modify in a table.

hive> ALTER TABLE name RENAME TO new_name

hive> ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])

hive> ALTER TABLE name DROP [COLUMN] column_name

hive> ALTER TABLE name CHANGE column_name new_name new_type hive>

ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

Example:

The following query renames the table from employee to emp:

hive> ALTER TABLE employee RENAME TO emp;

The following queries rename the column name and column data type using the above data:

hive> ALTER TABLE employee CHANGE name ename String;

hive> ALTER TABLE employee CHANGE salary salary Double;

The following query adds a column named dept to the employee table:

hive> ALTER TABLE employee ADD COLUMNS (dept STRING COMMENT 'Department name');

[9] Drop Table-

The syntax is as follows:

hive>DROP TABLE [IF EXISTS] table_name;

The following query drops a table named employee:

hive> DROP TABLE IF EXISTS employee;

On successful execution of the query, you get to see the following response:

OUTPUT:

OK

Time taken: 5.3 seconds

hive>

The following query is used to verify the list of tables after deleting employee table:

hive> SHOW TABLES;

OUTPUT:

emp

ok

Time taken: 2.1 seconds

hive>

RESULT:

Thus, a procedure to installation of HIVE and commands are executed successfully.

EX NO: 6.1 **HBASE**
DATE:

AIM:

To Installation of HBase along with Practice examples.

THEORY:

Apache **HBase** is an open source non-relational (NoSQL) distributed column-oriented database that runs on top of HDFS and real-time read/write access to those large data-sets. Initially, it was Google Big Table, afterwards it was re-named as HBase and is primarily written in Java, designed to provide quick random access to huge amounts of the data-set.

In brief, the HBase can store massive amounts of data from terabytes to petabytes and allows fast random reads and writes that cannot be handled by the Hadoop. Even relational databases (RDBMS) cannot handle a variety of data that is growing exponentially.

HBase can be installed in three modes. The features of these modes are mentioned below.

[1] Standalone mode installation (No dependency on Hadoop system)

- This is default mode of HBase
- It runs against local file system
- It doesn't use Hadoop HDFS
- Only HMaster daemon can run
- Not recommended for production environment
- Runs in single JVM

[2] Pseudo-Distributed mode installation (Single node Hadoop system + HBase installation)

- It runs on Hadoop HDFS
- All Daemons run in single node
- Recommend for production environment

[3] Fully Distributed mode installation (Multi node Hadoop environment + HBase installation)

- It runs on Hadoop HDFS
- All daemons going to run across all nodes present in the cluster
- Highly recommended for production environment

PREPARE:

These softwares should be prepared to install Hadoop 2.8.0 on window 10 64bit

1. Download Hadoop 2.8.0
2. Java JDK 1.8.0.zip
3. **Download HBase 1.4.7**

   http://www.apache.org/dyn/closer.lua/hbase/

PROCEDURE:

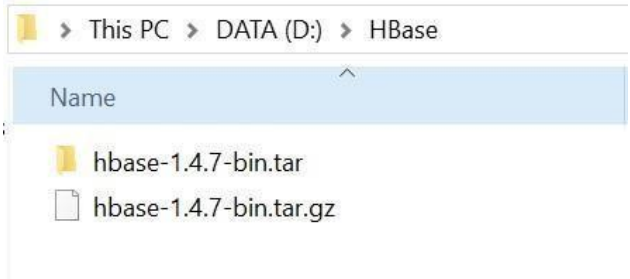
Hbase - Standalone mode installation

Here, we will go through the Standalone mode installation with Hbase on Windows 10.

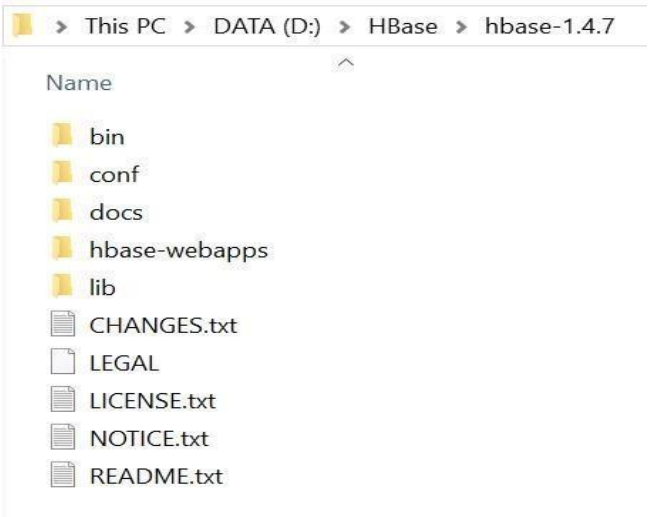
STEP - 1: Extract the HBase file

Extract file hbase-1.4.7-bin.tar.gz and place under "D:\HBase", you can use any preferred location:

[1] You will get again a tar file post extraction –



[2] Go inside of hbase-1.4.7-bin.tar folder and extract again.Then Copy the leaf folder “hbase-1.4.7” and move to the root folder "D:\HBase" folders:



STEP - 2: Configure Environment variable

Set the path for the following Environment variable (User Variables) on windows 10 –

- **HBASE_HOME - D:\HBase\hbase-1.4.7**

This PC - > Right Click - > Properties - > Advanced System Settings - > Advanced - > Environment Variables

STEP - 3: Configure System variable

Next onward need to set System variable, including Hive bin directory path: Variable:

Path

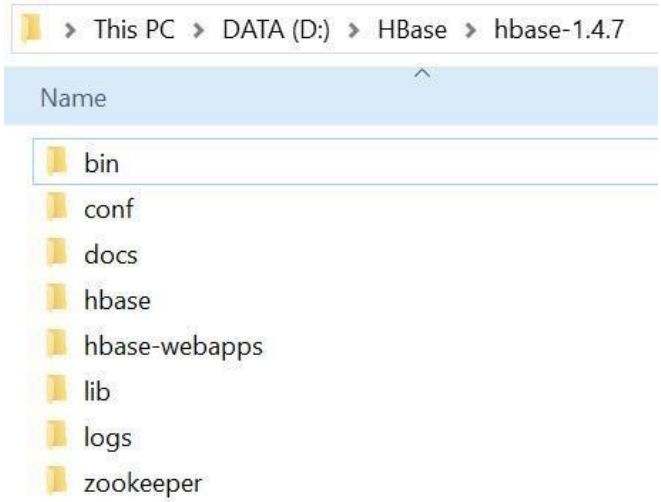
Value: >D:\HBase\hbase-1.4.7\bin

STEP - 4: Create required folders

Create some dedicated folders -

1. Create folder "hbase" under “D:\HBase\hbase-1.4.7”.
2. Create folder "zookeeper" under “D:\HBase\hbase-1.4.7”.

For example -



STEP - 5: Configured required files

Next, essential to configure two key files with minimal required details –

- hbase-env.cmd
- hbase-site.xml

[1] Edit file D:/HBase/hbase-1.4.7/conf/hbase-env.cmd, mention JAVA_HOME path in the location and save this file.

@rem set JAVA_HOME=c:\apps\java
set JAVA_HOME=%JAVA_HOME

```
@rem set JAVA_HOME=c:\apps\java
set JAVA_HOME=%JAVA_HOME%
```

[2] Edit file D:/HBase/hbase-1.4.7/conf/hbase-site.xml, paste below xml paragraph and save this file.

```
<configuration>
 <property>
  <name>hbase.rootdir</name>
  <value>file:///D:/HBase/hbase-1.4.7/hbase</value>
 </property>
 <property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/D:/HBase/hbase-1.4.7/zookeeper</value>
 </property>
```

```
 <property>
  <name> hbase.zookeeper.quorum</name>
  <value>127.0.0.1</value>
 </property>
</configuration>
```

All HMaster and ZooKeeper activities point out to this hbase-site.xml.

[3] Edit file hosts (C: /Windows/System32/drivers/etc/hosts), mention localhost IP and save this file.

127.0.0.1 localhost

```
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost

127.0.0.1       localhost
```

STEP - 6: Start HBase

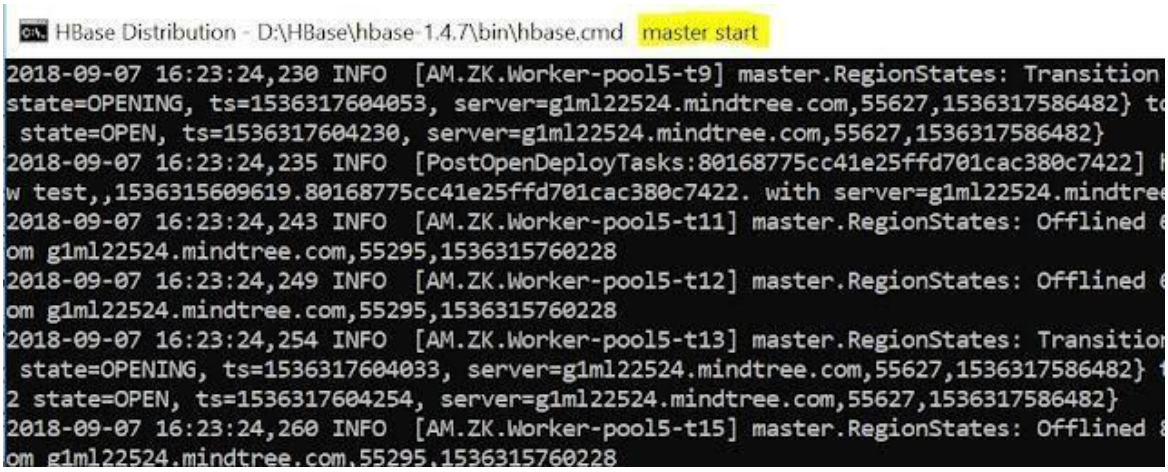
Here need to start HBase first :

Open command prompt and change directory to “D:\HBase\hbase-1.4.7\bin" and type "**start-hbase.cmd**" to start HBase.

```
D:\HBase\hbase-1.4.7\bin>start-hbase.cmd
```

It will open a separate instances of cmd for following tasks –

- HBase Master



STEP - 7: Validate HBase

Post successful execution of HBase, verify the installation using following commands –

- hbase –version
- jps

```
D:\HBase\hbase-1.4.7\bin>hbase -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
```

```
D:\HBase\hbase-1.4.7\bin>jps
20768 HMaster
14040 Jps
```

If we can see HMaster is in running mode, then our installation is okay.

STEP - 8: Execute HBase Shell

The standalone mode does not require Hadoop daemons to start. HBase can run independently. HBase shell can start by using "hbase shell" and it will enter into interactive shell mode –

```
D:\HBase\hbase-1.4.7\bin>hbase shell
```

```
D:\HBase\hbase-1.4.7\bin>hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/HBase/hbase-1.4.7/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLogger
Binder.class]
SLF4J: Found binding in [jar:file:/D:/Hadoop/hadoop-2.8.0/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/s
lf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.7, r763f27f583cf8fd7ecf79fb6f3ef57f1615dbf9b, Tue Aug 28 14:40:11 PDT 2018

hbase(main):001:0>
```

HBase installed !!

PROGRAM:

QUERIES AND OUTPUT:

[1] Create a simple table-

hBase>create 'student', 'bigdata'

```
hbase(main):001:0> create 'student', 'bigdata'
0 row(s) in 1.7430 seconds

=> Hbase::Table - student
hbase(main):002:0>
```

[2] List the table has been created-

hBase>list

```
hbase(main):002:0> list
TABLE

student

test

test_table

3 row(s) in 0.0180 seconds

=> ["student", "test", "test_table"]
hbase(main):003:0>
```

[3] Insert some data to above created table-

hBase>put ‘tablename’, ‘rowname’, ‘columnvalue’, ‘value’

hBase>put 'student', 'row1', 'bigdata:hadoop', 'hadoop couse'

```
hbase(main):006:0> put 'student', 'row1', 'bigdata:hadoop', 'hadoop couse'
0 row(s) in 0.0130 seconds
```

[4] List all rows in the table-

hBase>scan 'student'

```
hbase(main):007:0> scan 'student'
```

[5] Disabling a Table using HBase Shell

To delete a table or change its settings, you need to first disable the table using the disable command. You can re-enable it using the enable command.

Given below is the syntax to disable a table:

hBase>disable 'emp'

Example

Given below is an example that shows how to disable a table.

hbase> disable 'emp'

0 row(s) in 1.2760 seconds

Verification

After disabling the table, you can still sense its existence through **list** and **exists** commands. You cannot scan it. It will give you the following error.

hbase> scan 'emp'

ROW COLUMN + CELL

ERROR: emp is disabled.

is_disabled

This command is used to find whether a table is disabled. Its syntax is as follows.

hbase> is_disabled 'table name'

The following example verifies whether the table named emp is disabled. If it is disabled, it will return true and if not, it will return false.

hbase(main):031:0> is_disabled 'emp'

true

0 row(s) in 0.0440 seconds

disable_all

This command is used to disable all the tables matching the given regex. The syntax for **disable_all** command is given below.

hbase> disable_all 'r.*'

Suppose there are 5 tables in HBase, namely raja, rajani, rajendra, rajesh, and raju. The following code will disable all the tables starting with **raj.**

```
hbase(main):002:07> disable_all 'raj.*'
raja
rajani
rajendra
rajesh
raju
Disable the above 5 tables (y/n)?
y
5 tables successfully disabled
```

RESULT:

Thus, a procedure to installation of HBase and queries are executed successfully.

EX.NO: 6.2 **THRIFT**
DATE:

AIM:

To Installing thrift along with Practice examples.

THEORY:

Apache Thrift is a RPC framework founded by facebook and now it is an Apache project. Thrift lets you define data types and service interfaces in a language neutral definition file. That definition file is used as the input for the compiler to generate code for building RPC clients and servers that communicate over different programming languages.

software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages.

PROCEDURE:

Step 1: Installing Apache Thrift in Windows

Installation Thrift can be a tiresome process. But for windows the compiler is available as a prebuilt exe. Download thrift.exe and add it into your environment variables.

- **Download Apache Thrift**

https://archive.apache.org/dist/thrift/0.8.0/thrift-0.8.0.exe

Step 2: Build and Install the Apache Thrift compiler:

Refer

1. **https://thrift.apache.org/**
2. **https://dzone.com/articles/apache-thrift-java-quickstart**

You will then need to build the Apache Thrift compiler and install it. See the installing Thrift guide for any help with this step.

Step 3: Writing a .thrift file

After the Thrift compiler is installed, you will need to create a thrift file. This file is an interface definition made up of thrift types and Services. The services you define in this file are implemented by the server and are called by any clients. The Thrift compiler is used to generate your Thrift File into source code which is used by the different client libraries and the server you write. To generate the source from a thrift file run

thrift --gen <language> <Thrift filename>

PROGRAM:

1.Example definition file (add.thrift)

```
namespace java com.eviac.blog.samples.thrift.server // defines the namespace
    typedef i32 int //typedefs to get convenient names for your types
```

service AdditionService { // defines the service to add two numbers

int add(1:int n1, 2:int n2), //defines a method

}

2.Compiling Thrift definition file

To compile the .following command.

thrift --gen <language> <Thrift filename>

For my example the command is,

thrift --gen java add.thrift

After performing the command, inside *gen-java* directory you'll find the source codes which is useful for building RPC clients and server. it will create a java code called *AdditionService.java*

Writing a service handler

Service handler class is required to implement the *AdditionService.Iface* interface.
Example service handler (AdditionServiceHandler.java)

```
package com.eviac.blog.samples.thrift.server;
import org.apache.thrift.TException;
public class AdditionServiceHandler implements AdditionService.Iface {

@Override
public int add(int n1, int n2) throws TException {

return n1 + n2; }

}
```

Writing a simple server

Following is an example code to initiate a simple thrift server. To enable the multithreaded server uncomment the commented parts of the example code.

Example server (MyServer.java)

```
package com.eviac.blog.samples.thrift.server; import
org.apache.thrift.transport.TServerSocket;
```

```
import org.apache.thrift.transport.TServerTransport;
import org.apache.thrift.server.TServer;
import org.apache.thrift.server.TServer.Args;
import org.apache.thrift.server.TSimpleServer;

public class MyServer {

 public static void StartsimpleServer(AdditionService.Processor<AdditionServiceHandler> processor) { try
  {
  TServerTransport serverTransport = new TServerSocket(9090);
  TServer server = new TSimpleServer(
    new Args(serverTransport).processor(processor));

  // Use this for a multithreaded server
  // TServer server = new TThreadPoolServer(new
  //   TThreadPoolServer.Args(serverTransport).processor(processor));

   System.out.println("Starting the simple server...");
   server.serve();
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
 public static void main(String[] args) {
 StartsimpleServer(new AdditionService.Processor<AdditionServiceHandler>(new
 AdditionServiceHandler()));
 }
}
```

Writing the client

Following is an example java client code which consumes the service provided by AdditionService.

Example client code (AdditionClient.java)

```
package    com.eviac.blog.samples.thrift.client;
import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.protocol.TProtocol; import
org.apache.thrift.transport.TSocket;
```

```
import org.apache.thrift.transport.TTransport;
import org.apache.thrift.transport.TTransportException;
public class AdditionClient {
public static void main(String[] args) {
  try {
   TTransport transport;

   transport = new TSocket("localhost", 9090);
   transport.open();

   TProtocol protocol = new TBinaryProtocol(transport);
   AdditionService.Client client = new AdditionService.Client(protocol);
   System.out.println(client.add(100, 200));

   transport.close();

  } catch (TTransportException e) {
   e.printStackTrace();
  } catch (TException x) {
   x.printStackTrace();
  }
 }

}
```

OUTPUT:

Run the server code(MyServer.java). It should output following and will listen to the requests.

```
Starting the simple server...
```

Then run the client code(AdditionClient.java). It should output following.

```
300
```

RESULT:

Thus, a procedure to installation of Thrift and programs are executed successfully.

EX.NO: 7.1 **CASSANDRA**
DATE:

AIM:

To Export and Import data in Cassandra

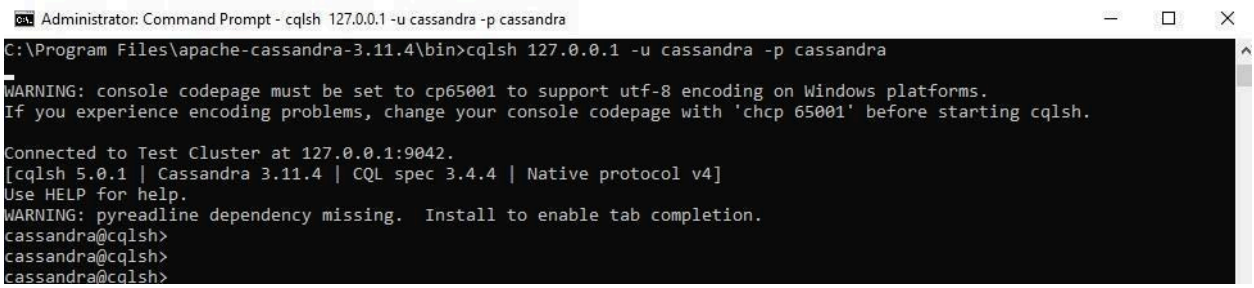
THEORY:

Cassandra is a distributed database management system which is open source with wide column store, NoSQL database to handle large amount of data across many commodity servers which provides high availability with no single point of failure. It is written in Java and developed by Apache Software Foundation.

CQL shell (cqlsh) :

This is a tool for Cassandra Query Language which supports Cassandra. cqlsh is a command-line shell for interacting with Cassandra through CQL (the Cassandra Query Language). with the help of the cql command, we can read and write data with the help of the cql query.

By default, CQL installed in bin/ directory alongside the Cassandra executable. In Cassandra, cqlsh utilizes the Python native protocol driver and connects to the single node specified on the command line.



PROGRAM:

Step 1: CREATE TABLE

Create table namely as Data in which id, firstname, lastname are the fields for sample exercise.

Table name: Data

```
CREATE TABLE Data (
  id UUID PRIMARY KEY,
  firstname text,
  lastname text
);
```

Step 2: INSERT DATA

Insert some data to export and import data.

Example:

```
INSERT INTO Data (id, firstname, lastname )
VALUES (3b6441dd-3f90-4c93-8f61-abcfa3a510e1, 'Ashish', 'Rana');

INSERT INTO Data (id, firstname, lastname)
VALUES (3b6442dd-bc0d-4157-a80f-abcfa3a510e2, 'Amit', 'Gupta');

INSERT INTO Data (id, firstname, lastname)
VALUES (3b6443dd-d358-4d99-b900-abcfa3a510e3, 'Ashish', 'Gupta');

INSERT INTO Data (id, firstname, lastname)
VALUES (3b6444dd-4860-49d6-9a4b-abcfa3a510e4, 'Dhruv', 'Gupta');

INSERT INTO Data (id, firstname, lastname)
VALUES (3b6445dd-e68e-48d9-a5f8-abcfa3a510e5, 'Harsh', 'Vardhan');

INSERT INTO Data (id, firstname, lastname)
VALUES (3b6446dd-eb95-4bb4-8685-abcfa3a510e6, 'Shivang', 'Rana');
```

Step 3: EXPORT DATA
Export Data used the following cqlsh query given below.

```
cqlsh>COPY Data(id, firstname, lastname)
TO 'AshishRana\Desktop\Data.csv' WITH HEADER = TRUE;
```

OUTPUT:
The CSV file is created:
Using 7 child processes
Starting copy of Data with columns [id, firstname, lastname].
Processed: 6 rows; Rate: 20 rows/s; Avg. rate: 30 rows/s
6 rows exported to 1 files in 0.213 seconds.

Step 4: DELETE DATA
Delete data from table ‘Data’ to import again from CSV file which is already has been created.
cqlsh>truncate Data;

Step 5: IMPORT DATA
To import Data used the following cqlsh query given below.
cqlsh>COPY Data (id, firstname, lastname) FROM 'AshishRana\Desktop\Data.csv' WITH HEADER = TRUE;
OUTPUT:
Using 7 child processes
Starting copy of Data with columns [id, firstname, lastname].
Processed: 6 rows; Rate: 10 rows/s; Avg. rate: 14 rows/s
6 rows imported from 1 files in 0.423 seconds (0 skipped).

Step 6: RETRIEVE IMPORTED DATA

To View the results whether it is successfully imported or not.

cqlsh>SELECT * FROM Data;

OUTPUT:

id	firstname	lastname
3b6446dd-eb95-4bb4-8685-abcfa3a510e6	Shivang	Rana
3b6444dd-4860-49d6-9a4b-abcfa3a510e4	Dhruv	Gupta
3b6445dd-e68e-48d9-a5f8-abcfa3a510e5	Harsh	Vardhan
3b6441dd-3f90-4c93-8f61-abcfa3a510e1	Ashish	Rana
3b6442dd-bc0d-4157-a80f-abcfa3a510e2	Amit	Gupta
3b6443dd-d358-4d99-b900-abcfa3a510e3	Ashish	Gupta

RESULT:

Thus, a procedure to installation of Cassandra was successfully completed and execution of files are exported and imported successfully.

EX.NO: 7.2 MongoDB

DATE:

AIM:

To Export and Import data in MongoDB

THEORY:

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

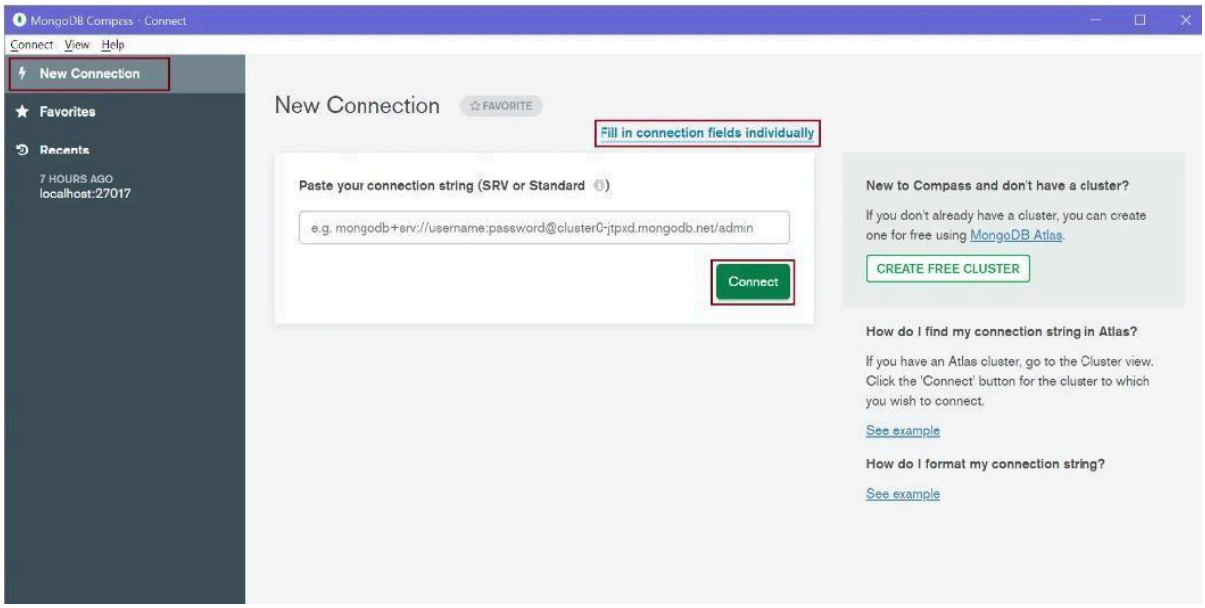
A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

PROCEDURE:

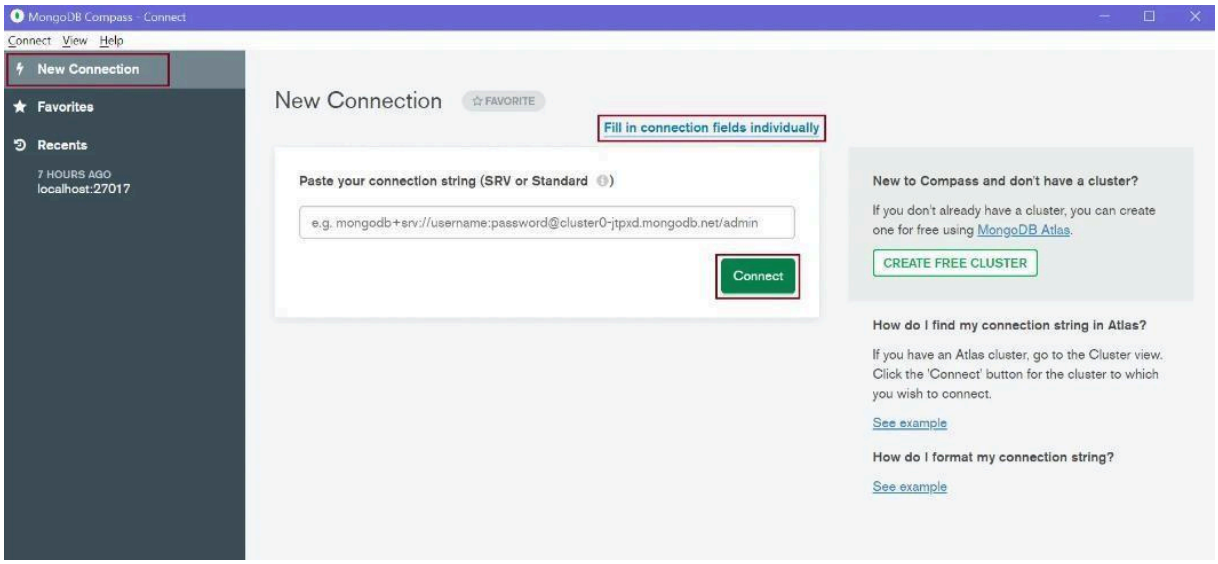
Step 1:

Install MongoDB Compass On Windows

MongoDB Compass is a GUI based tools (unline MongoDB Shell) to interact with local or remote MongoDB server and databases. Use Compass to visually explore your data, run ad hoc queries, perform CRUD operations, and view and optimize your query performance. It can be installed on Linux, Mac, or Windows.

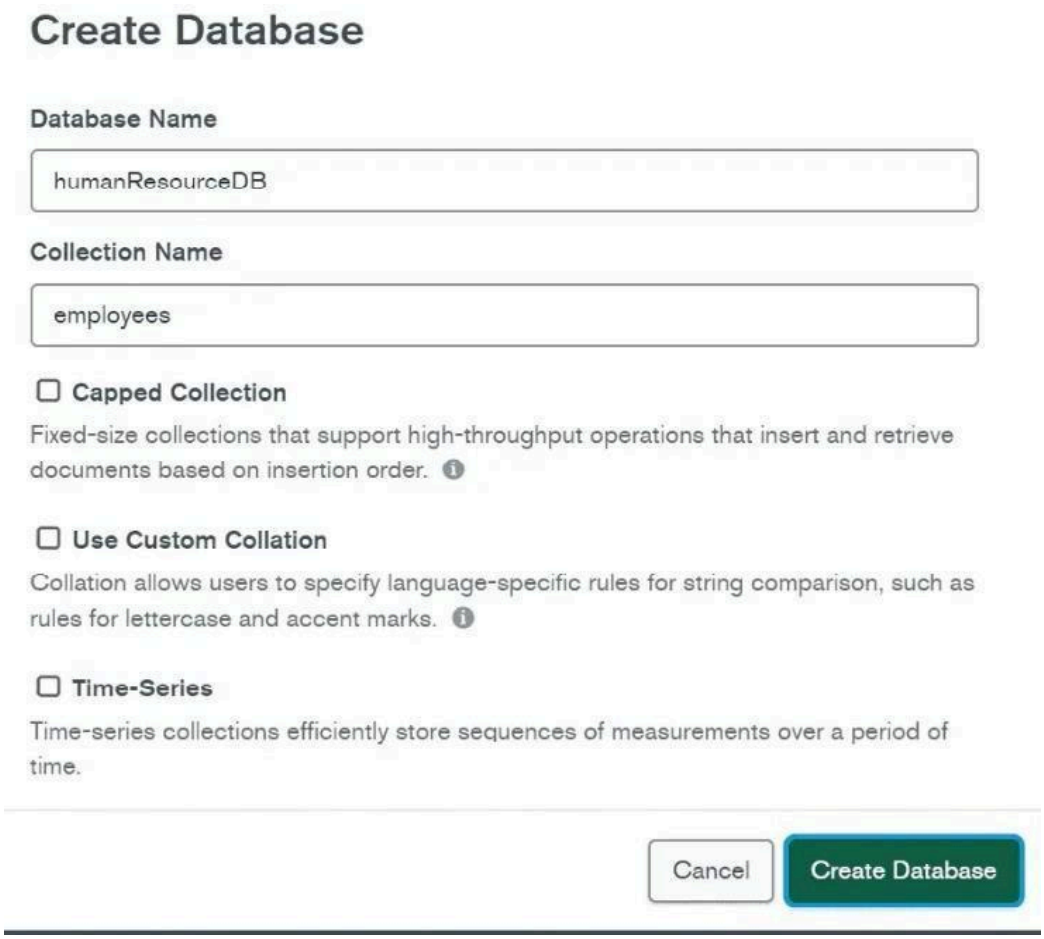


just click OK to connect with your local server, as shown below.

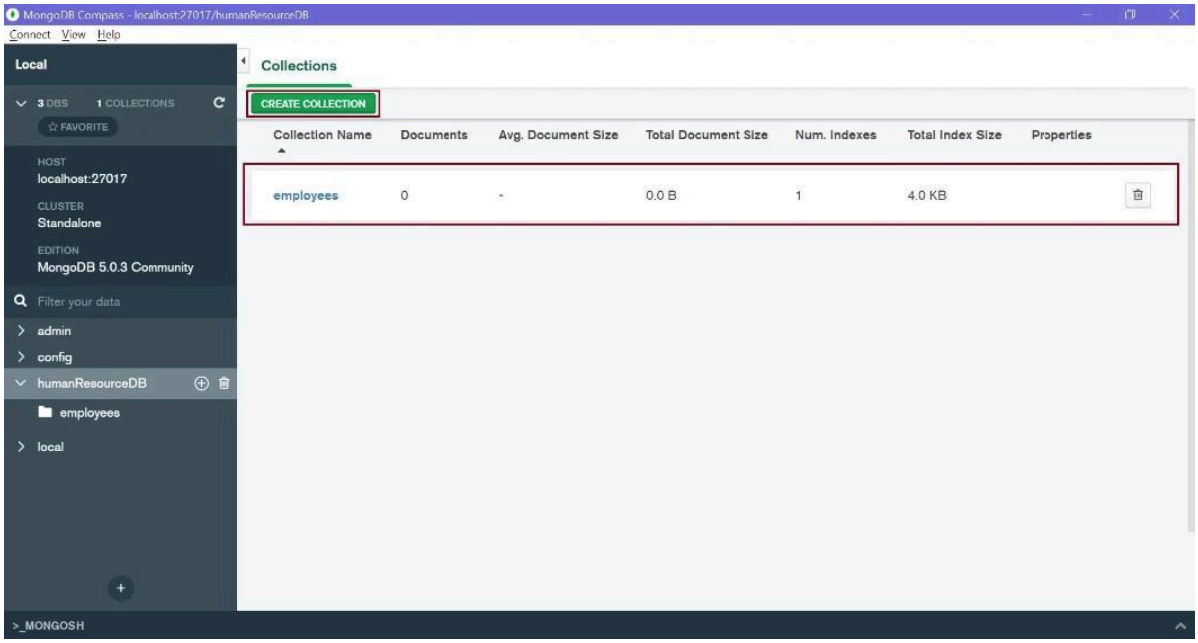


As you can see above, it will display all the databases on the connected MongoDB server. On the left pane, it displays information about the connected server.

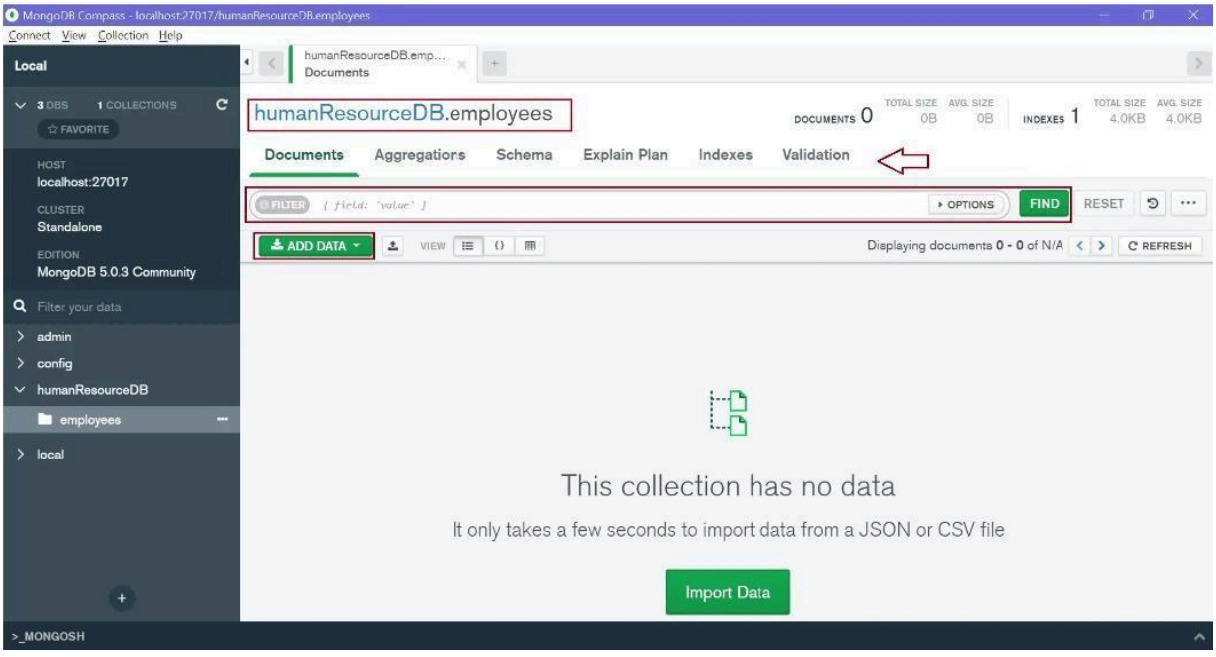
Now, you can create, modify, delete databases, collections, documents using MongoDB Compass.
Click on the CREATE DATABASE button to create a new database. This will open Create Database popup, as shown below.



Enter your database name and collection name and click Create Database. This will create a new database humanResourceDB with the new employees collection shown below.



Click on employees collection to insert, update, find documents in it. This will open the following window to manage documents.



PROGRAMS:

1. Create a new database:

Command: use <database-name>

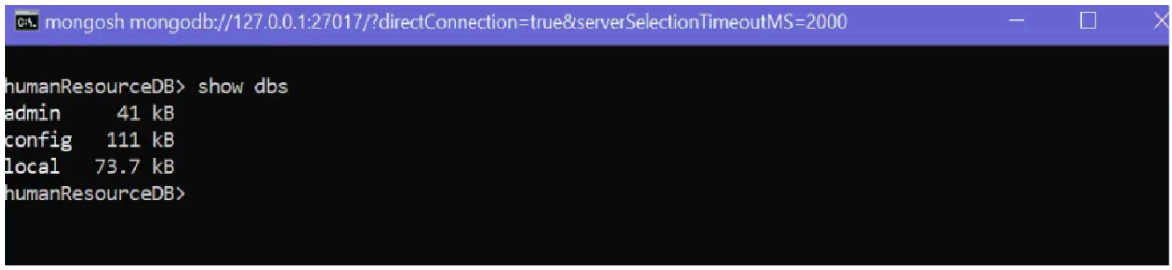
Example: use humanResourceDB

2. Check databases

list: Command: show dbs

Example: show dbs

OUTPUT:

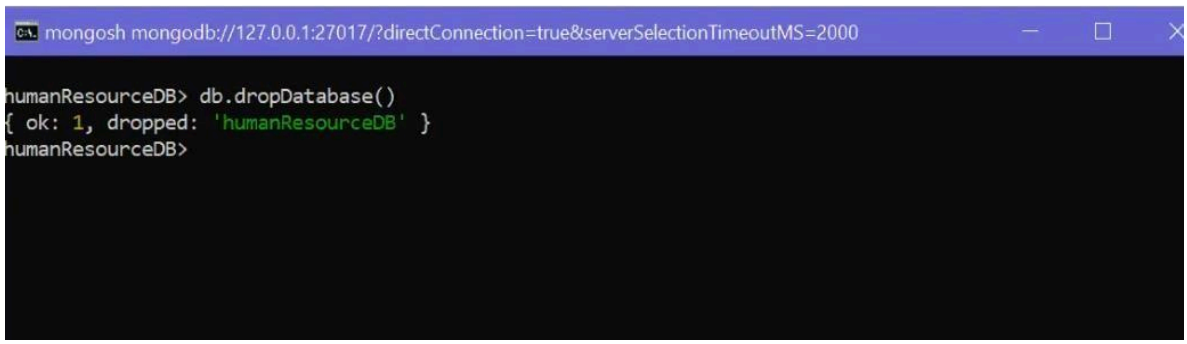


3. Delete a database:

Command: db.dropDatabase()

Example: db.dropDatabase()

OUTPUT:

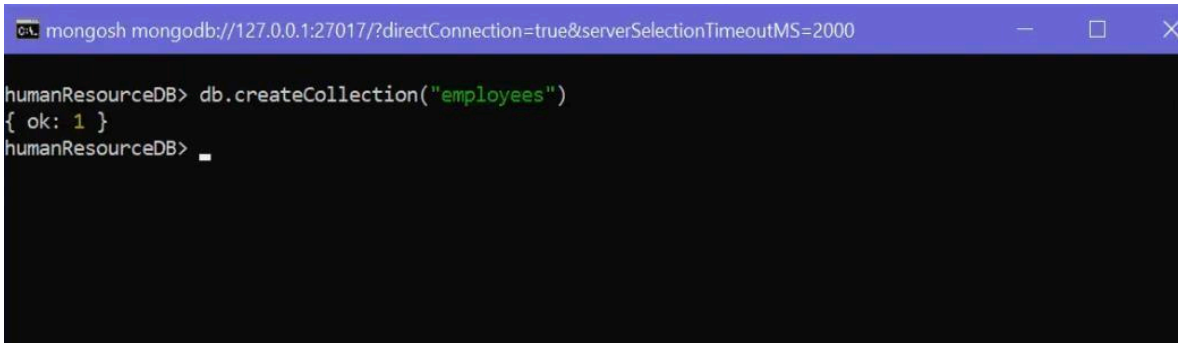


4. Create a collection:

Command: db.createCollection()

Example: db.createCollection("employee")

OUTPUT:



Creates multiple collections.

To show Db:

To delete a collection, use the db.<collection-name>.drop() method

5. Insert documents into a collection:

5.1 insertOne() - Inserts a single document into a collection. Command: db.<collection>.insertOne()

Example: db.employees.insertOne({

```
    firstName: "John",
    lastName: "King",
    email: "john.king@abc.com"
})
```

OUTPUT:

```
{
  acknowledged: true,
  insertedId: ObjectId("616d44bea861820797edd9b0")
}
```

5.2 insert() - Inserts one or more documents into a collection. Command: db.<collection>.insert()

Example: db.employees.insert(

```
   [
      {
         firstName: "John",
         lastName: "King",
         email: "john.king@abc.com"
      },
      {
            firstName: "Sachin",
            lastName: "T",
            email: "sachin.t@abc.com"
         ])
```

OUTPUT:

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("616d63eda861820797edd9b3"),
    '1': 1,
    '2': ObjectId("616d63eda861820797edd9b5")
  }
}
```

5.3 insertMany() - Insert multiple documents into a collection. Commands: db.<collection>.insertMany()

Example 1:

```
            db.employees.insertMany(
   [
      {
         firstName: "John",
         lastName: "King",
         email: "john.king@abc.com"
      },
      {
            firstName: "Sachin",
            lastName: "T",
            email: "sachin.t@abc.com"
 },
      {
            firstName: "James",
            lastName: "Bond",
            email: "jamesb@abc.com"
      },
   ])
```

Example 2: insertMany() with Custom _id

```
db.employees.insertMany([
```

```
    _id:1,
    firstName: "John",
    lastName: "King",
    email: "john.king@abc.com",
    salary: 5000
  },
  {
    _id:2,
    firstName: "Sachin",
    lastName: "T",
    email: "sachin.t@abc.com",
    salary: 8000
  },
  {
    _id:3,
    firstName: "James",
    lastName: "Bond",
    email: "jamesb@abc.com",
    salary: 7500
  },
{
    _id:4,
    firstName: "Steve",
    lastName: "J",
    email: "steve.j@abc.com",
    salary: 9000
  },
  {
    _id:5,
    firstName: "Kapil",
    lastName: "D",
    email: "kapil.d@abc.com",
    salary: 4500
  },
```

```
    _id:6,
    firstName: "Amitabh",
    lastName: "B",
    email: "amitabh.b@abc.com",
    salary: 11000
  }
])
```

6. Find the data:

1. findOne() - returns a the first document that matched with the specified criteria.
2. find() - returns a cursor to the selected documents that matched with the specified criteria.

Command: find()

Example : db.employees.find().pretty()

OUTPUT:

```
{   _id: ObjectId("616d44bea861820797edd9b0"),
   firstName: "John",
   lastName: "King",
email: "john.king@abc.com"
}
```

Example: db.employees.findOne({firstName: "Kapil"})

OUTPUT:

```
{
 _id: 5,
 firstName: 'Kapil',
 lastName: 'D',
 email: 'kapil.d@abc.com',
 salary: 4500
}
```

6.1 Find Multiple Documents:

Example: db.employees.find({salary: 7000})

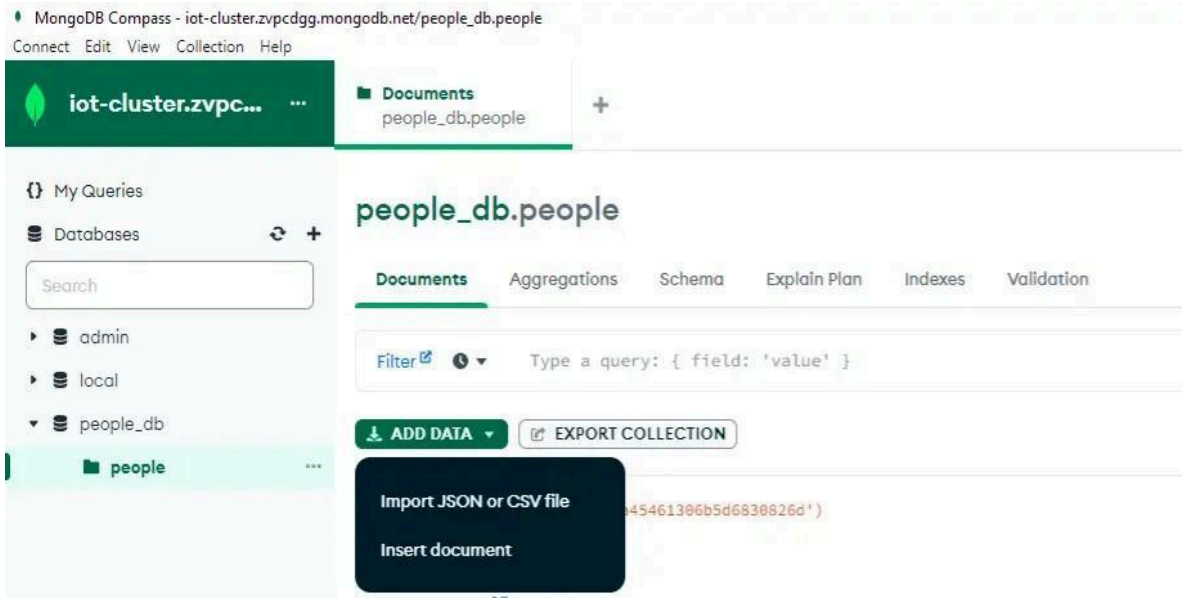
OUTPUT:

```
[{
      _id:4,
      firstName: "Steve",
      lastName: "J",
      email: "steve.j@abc.com",
      salary: 7000
   },
   {
      _id:6,
      firstName: "Amitabh",
      lastName: "B",
      email: "amitabh.b@abc.com",
      salary: 7000
   }
]
```

7. Import Data: Import data into a collection from either a **JSON** or **CSV** file

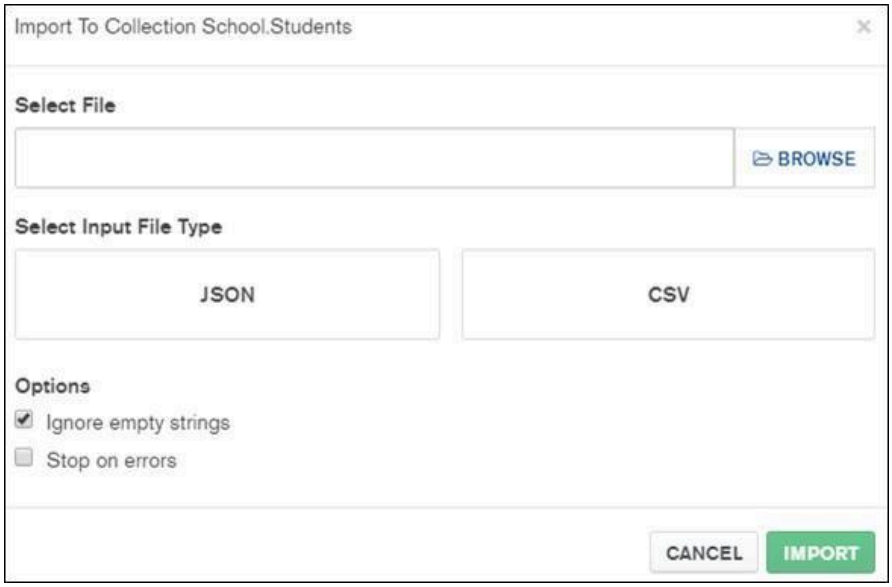
Step 1: Navigate to your target collection: Either select the collection from the Collections tab or click the collection in the left-hand pane.

Step 2: Click the Add Data **dropdown and select** Import JSON or CSV file**.**

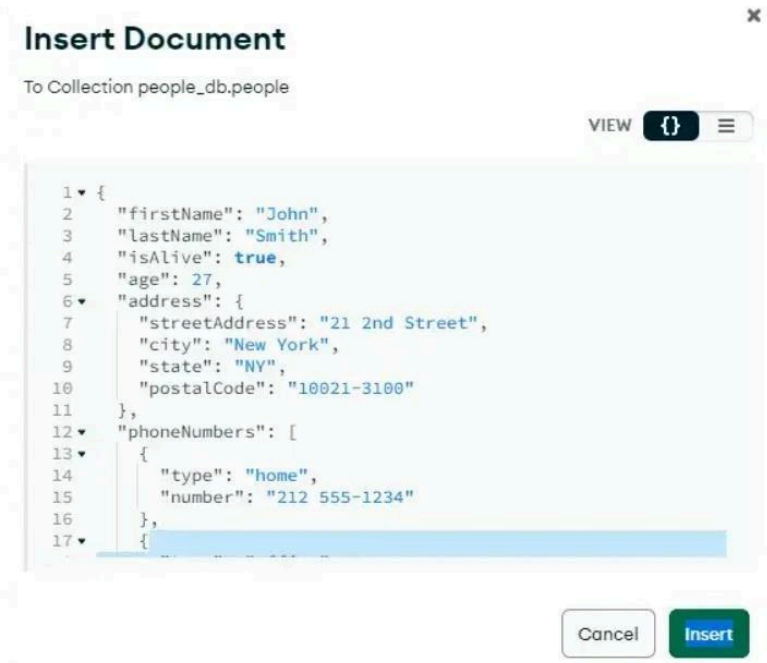


Step 3: Select the appropriate file type.

Select either a JSON or CSV file to import and click **Select**.

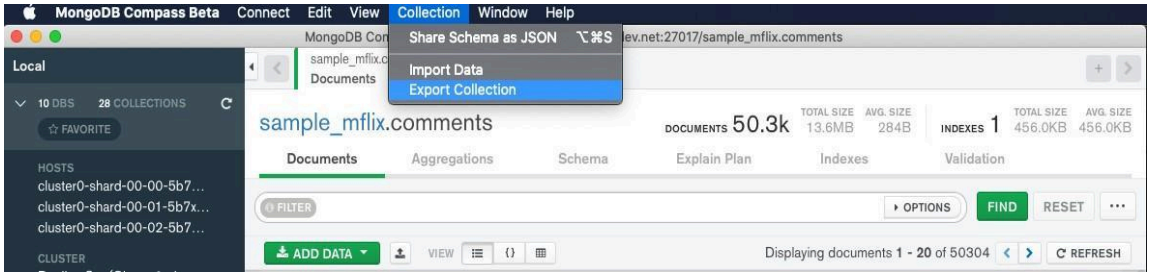


Step 4: Click insert or Import**.**

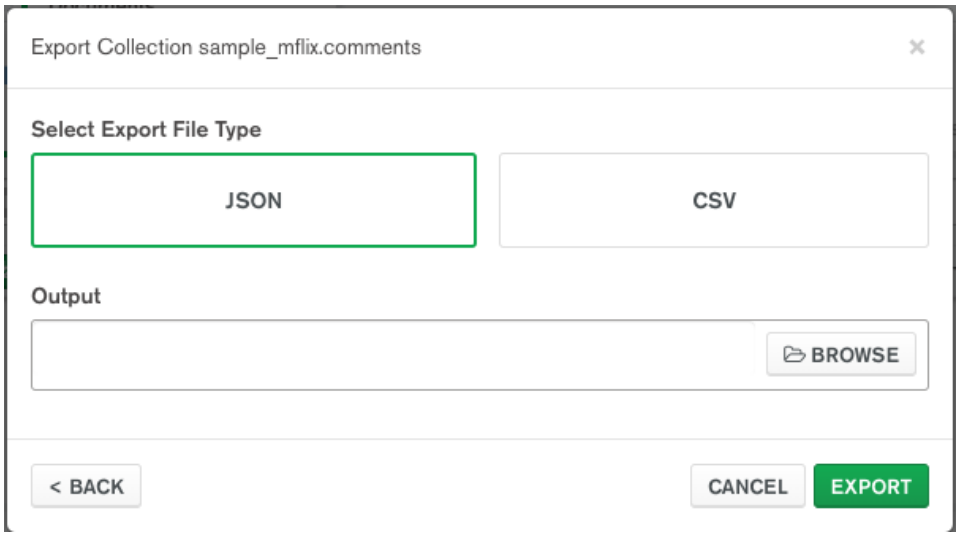


8. Export Data from a Collection: Export data from a collection as either a **JSON** or **CSV** file.

Step 1: Click the Export Data **dropdown and select** Export the full collection**.**



Step 2: Select your file type: You can select either **JSON** or **CSV**.



Step 3: Click Export and Choose where to export the file and click **Select**.

RESULT:

Thus, a procedure to installation of MongoDB was completed and execution of files are exported and imported successfully.