

Ex. No	Downloading and installing Hadoop; Understanding different Hadoop modes. Startup scripts, Configuration files	Date
1		

Aim:

To Install Apache Hadoop.

DESCRIPTION:-

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-2.7.3 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.

Procedur:

Prerequisites:

Step1: Installing Java 8 version.

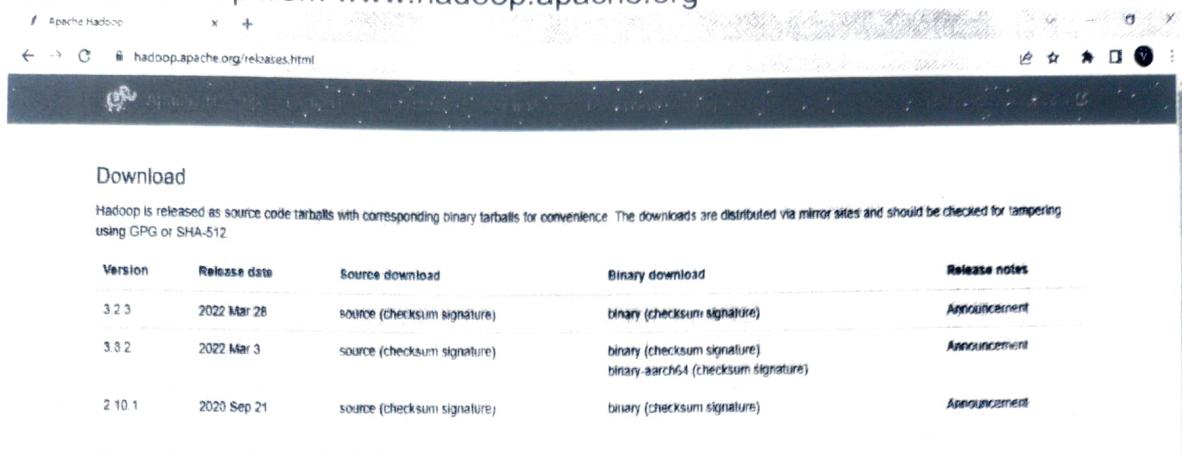
Openjdk version "1.8.0_91"

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)

Step2: Installing Hadoop

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

Download Hadoop from www.hadoop.apache.org



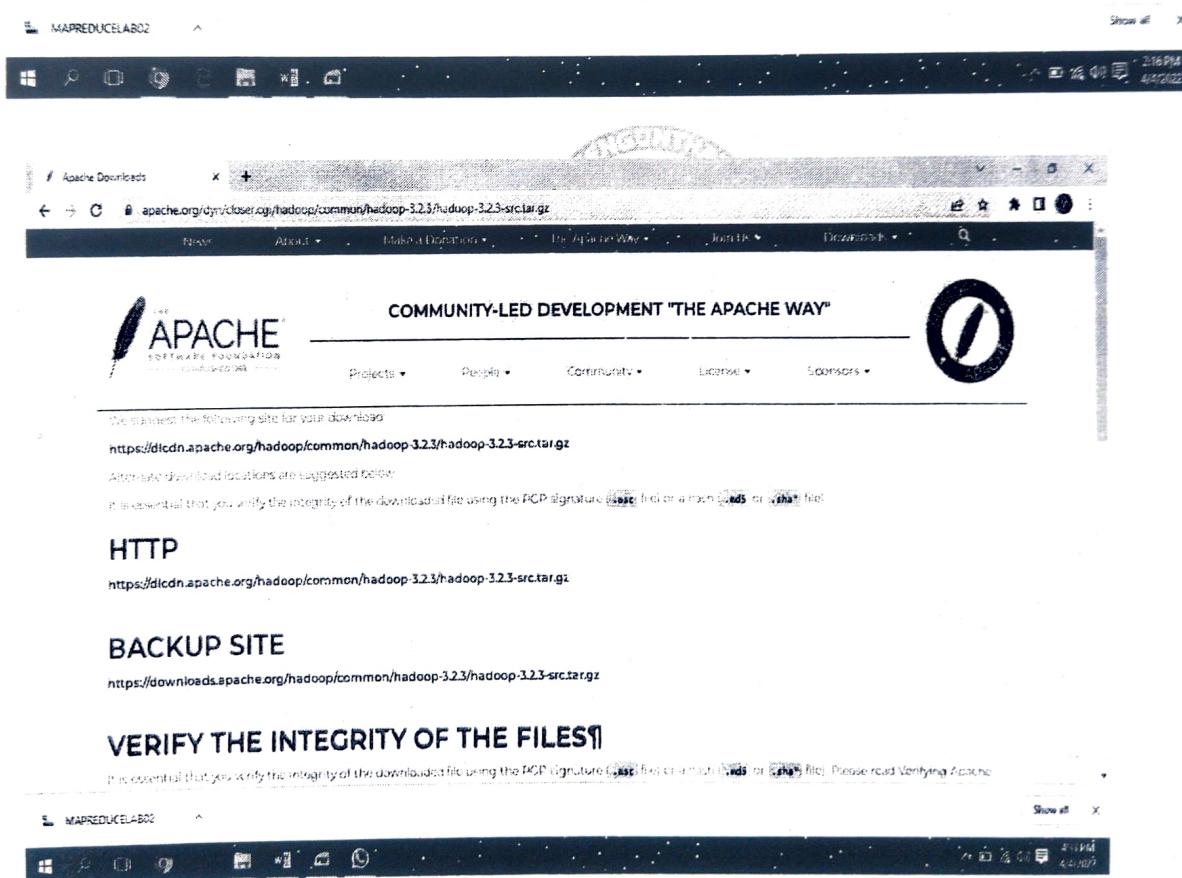
Version	Release date	Source download	Binary download	Release notes
3.2.3	2022 Mar 28	source (checksum signature)	binary (checksum signature)	Announcement
3.8.2	2022 Mar 3	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement
2.10.1	2020 Sep 21	source (checksum signature)	binary (checksum signature)	Announcement

To verify Hadoop releases using GPG:

1. Download the release `hadoop-X.Y.Z-src.tar.gz` from a mirror site.
2. Download the signature file `hadoop-X.Y.Z-src.tar.gz.asc` from Apache.
3. Download the Hadoop KEYS file
4. `gpg --import KEYS`
5. `gpg --verify hadoop-X.Y.Z-src.tar.gz.asc`

To perform a quick check using SHA-512:

1. Download the release `hadoop-X.Y.Z-src.tar.gz` from a mirror site.
2. Download the checksum `hadoop-X.Y.Z-src.tar.gz.sha512` or `hadoop-X.Y.Z-src.tar.gz.md5` from Apache.



COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"

Use this site for your download:

<https://dlcdn.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3-src.tar.gz>

Alternate download locations are suggested below:

It is essential that you verify the integrity of the downloaded file using the GPG signature (.asc) file or a md5 (.md5) or .sha1 (.sha1) file.

HTTP

<https://dlcdn.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3-src.tar.gz>

BACKUP SITE

<https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3-src.tar.gz>

VERIFY THE INTEGRITY OF THE FILES!

It is essential that you verify the integrity of the downloaded file using the GPG signature (.asc) file or a md5 (.md5) or .sha1 (.sha1) file. Please read [Verifying Apache](#).

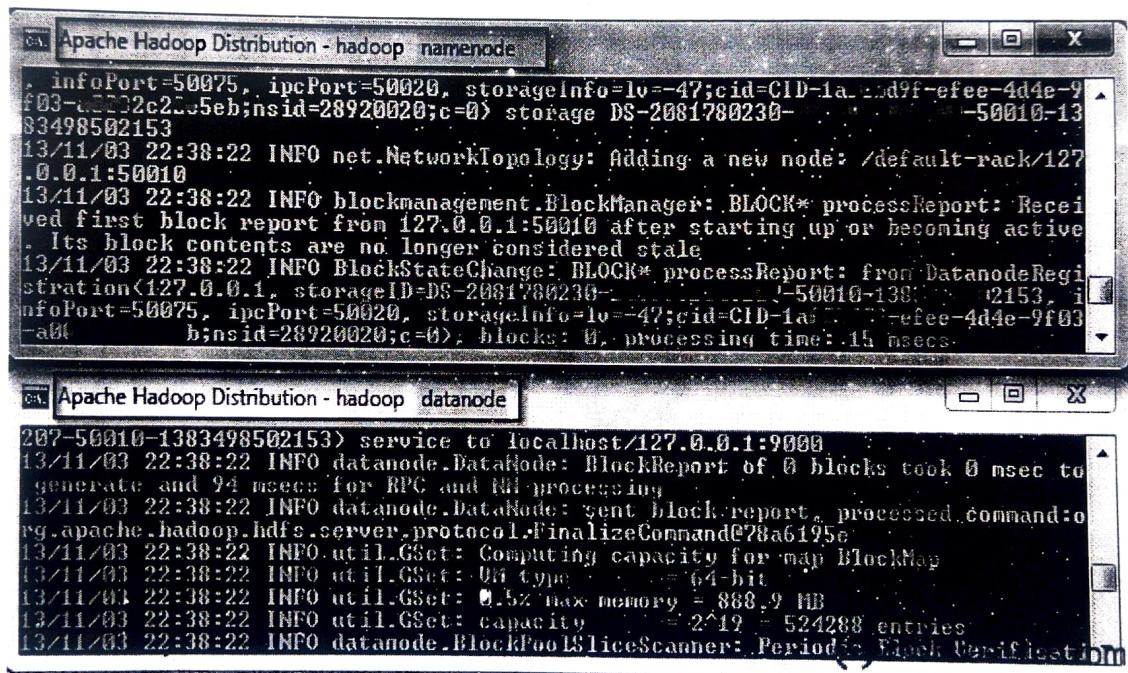
Procedure to Run Hadoop

1. Install Apache Hadoop 2.2.0 in Microsoft Windows OS
If Apache Hadoop 2.2.0 is not already installed then follow the post Build, Install, Configure and Run Apache Hadoop 2.2.0 in Microsoft Windows OS.
2. Start HDFS (Namenode and Datanode) and YARN (Resource Manager and Node Manager)

Run following commands. *Command* *Prompt* C:\Users\abhijitg>cd c:\hadoop

c:\hadoop>sbin\start-dfs c:\hadoop>sbin\start-yarn starting yarn daemons

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.



The image shows two terminal windows side-by-side. The top window is titled 'Apache Hadoop Distribution - hadoop namenode' and displays log entries for the Namenode. The bottom window is titled 'Apache Hadoop Distribution - hadoop datanode' and displays log entries for the Datanode. Both windows show standard Hadoop startup and initial processing logs.

```
Apache Hadoop Distribution - hadoop namenode
infoPort=50075, ipcPort=50020, storageInfo=lv=-47;cid=CID-1a...nd9f-efee-4d4e-9f03-1383498502153;nsid=28920020;c=0> storage DS-2081780230-50010-1383498502153
13/11/03 22:38:22 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:50010
13/11/03 22:38:22 INFO blockmanagement.BlockManager: BLOCK* processReport: Received first block report from 127.0.0.1:50010 after starting up or becoming active. Its block contents are no longer considered stale
13/11/03 22:38:22 INFO BlockStateChange: BLOCK* processReport: from DatanodeRegistration<127.0.0.1, storageID=DS-2081780230-50010-1383498502153, infoPort=50075, ipcPort=50020, storageInfo=lv=-47;cid=CID-1a...nd9f-efee-4d4e-9f03-a0t b;nsid=28920020;c=0>, blocks: 0, processing time: 14 msec

Apache Hadoop Distribution - hadoop datanode
202-50010-1383498502153> service to localhost/127.0.0.1:9000
13/11/03 22:38:22 INFO datanode.DataNode: BlockReport of 0 blocks took 0 msec to generate and 94 msec for RPC and MI processing
13/11/03 22:38:22 INFO datanode.DataNode: sent block report, processed command:org.apache.hadoop.hdfs.server.protocol.FinalizeCommand@78a6195e
13/11/03 22:38:22 INFO util.GSet: Computing capacity for map BlockMap
13/11/03 22:38:22 INFO util.GSet: Off type: 0x0000000000000000
13/11/03 22:38:22 INFO util.GSet: 0.5x max memory = 888.9 MB
13/11/03 22:38:22 INFO util.GSet: capacity = 2^12 = 524288 entries
13/11/03 22:38:22 INFO datanode.BlockPoolSliceScanner: Periodic block verification
```

Run wordcount MapReduce job

Now we'll run wordcount MapReduce job available

in %HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.2.0.jar

Create a text file with some content. We'll pass this file as input to the wordcount MapReduce job for counting words.

C:\file1.txt

Install Hadoop

Run Hadoop Wordcount Mapreduce Example

Create a directory (say 'input') in HDFS to keep all the text files (say 'file1.txt') to be used for counting words.

```
C:\Users\abhijitg>cd c:\hadoop C:\hadoop>bin\hdfs dfs -mkdir input
```

Copy the text file(say 'file1.txt') from local disk to the newly created 'input' directory in HDFS.

```
C:\hadoop>bin\hdfs dfs -copyFromLocal c:/file1.txt input
```

Check content of the copied file.

```
C:\hadoop>hdfs dfs -ls
input Found 1 items
-rw-r--r-- 1 ABHIJITG supergroup 55 2014-02-03 13:19 input/file1.txt
```

```
C:\hadoop>bin\hdfs dfs -cat input/file1.txt
```

Install Hadoop

Run Hadoop Wordcount Mapreduce Example

Run the wordcount MapReduce job provided in %HADOOP_HOME%\share\hadoop\mapreduce\hadoop-mapreduce-examples-2.2.0.jar

```
C:\hadoop>bin\yarn jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount input output
14/02/03 13:22:02 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
14/02/03 13:22:03 INFO input.FileInputFormat: Total input paths to process : 1
14/02/03 13:22:03 INFO mapreduce.JobSubmitter: number of splits:1
:
:
14/02/03 13:22:04 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1391412385921_0002
14/02/03 13:22:04 INFO impl.YarnClientImpl: Submitted application application_1391412385921_0002 to ResourceManager at /0.0.0.0:8032
14/02/03 13:22:04 INFO mapreduce.Job: The url to track the job: http://ABHIJITG:8088/proxy/application_1391412385921_0002/
14/02/03 13:22:04 INFO mapreduce.Job: Running job: job_1391412385921_0002 14/02/03 13:22:14 INFO mapreduce.Job: Job job_1391412385921_0002 running in uber mode : false
14/02/03 13:22:14 INFO mapreduce.Job: map 0% reduce 0%
14/02/03 13:22:22 INFO mapreduce.Job: map 100% reduce 0%
14/02/03 13:22:30 INFO mapreduce.Job: map 100% reduce 100%
14/02/03 13:22:30 INFO mapreduce.Job: Job job_1391412385921_0002 completed successfully
```

14/02/03 13:22:31 INFO mapreduce.Job:
Counters: 43 File System Counters
FILE: Number of bytes read=89
FILE: Number of bytes written=160142 FILE: Number of read operations=0 FILE: Number of large read operations=0 FILE: Number of write operations=0
HDFS: Number of bytes read=171 HDFS: Number of bytes written=59 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)=5657 Total time spent by all reduces in occupied slots (ms)=6128

Map-Reduce Framework

Map input records=2
Map output records=7 Map output bytes=82
Map output materialized bytes=89 Input split bytes=116
Combine input records=7 Combine output records=6
Reduce input groups=6 Reduce shuffle bytes=89
Reduce input records=6 Reduce output records=6
Spilled Records=12
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=145 CPU time spent (ms)=1418

Physical memory (bytes)
snapshot=368246784 Virtual memory (bytes)
snapshot=513716224 Total committed heap
usage (bytes)=307757056

Shuffle Errors

BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

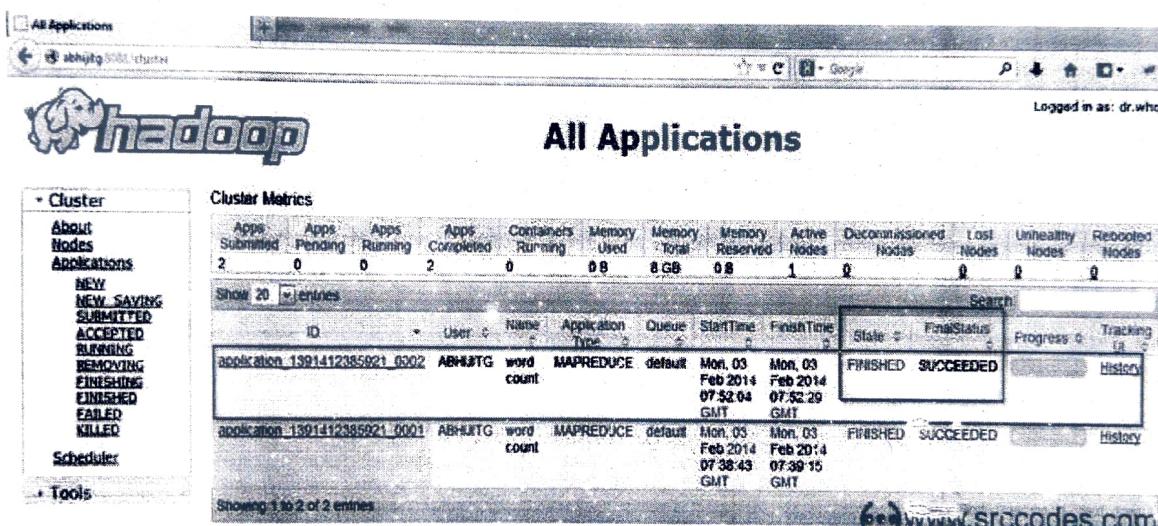
File Input Format

Counters Bytes
Read=55

File Output Format Counters

Bytes Written=59

<http://abhijitg:8088/cluster>



The screenshot shows the Hadoop 'All Applications' interface. On the left, there's a sidebar with 'Cluster Metrics' and a list of application states: NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, REMOVING, FINISHING, FINISHED, FAILED, KILLED, and Scheduler. Below that is a 'Tools' section. The main area is titled 'All Applications' and shows a table of running applications. The table has columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, and Tracking. Two entries are listed:

ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status	Progress	Tracking
application_1391412385921_0302	ABHIJITG	word count	MAPREDUCE	default	Mon, 03 Feb 2014 07:52:04 GMT	Mon, 03 Feb 2014 07:52:29 GMT	FINISHED	SUCCEEDED	0%	History
application_1391412385921_0003	ABHIJITG	word count	MAPREDUCE	default	Mon, 03 Feb 2014 07:38:43 GMT	Mon, 03 Feb 2014 07:39:15 GMT	FINISHED	SUCCEEDED	0%	History

At the bottom, it says 'Showing 1 to 2 of 2 entries'.

Result:

Thus the Hadoop is installed in stand-alone mode and verified it by running an example program it provided.

Ex. No	Hadoop Implementation of file management tasks, such as Adding files and directories, retrieving files and Deleting files	Date
2		

Aim:

To implement the following file management tasks in Hadoop

1. Adding files and directories
2. Retrieving files
3. 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 with HDFS.

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:

1. Adding files and directories to HDFS
2. Retrieving files from HDFS to local filesystem
3. Deleting files from HDFS

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

Step 1:Starting HDFS

Initially you have to format the configured HDFS file system, open namenode (HDFS server), and execute the following command.

```
$ hadoop namenode –format
```

After formatting the HDFS, start the distributed file system. The following command will start the namenode as well as the data nodes as cluster.

```
$ start-dfs.sh
```

Listing Files in HDFS

After loading the information in the server, we can find the list of files in a directory, status of a file, using `ls`. Given below is the syntax of `ls` that you can pass to a directory or a filename as an argument.

```
DESKTOP2@DESKTOP2: ~ $ hadoop fs -ls
2023-10-12 11:37:33,233 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin- java classes where applicable.
Found 4 items
drwxr-xr-x  - ambal2 supergroup          0 2023-10-11 13:04 bigdata
drwxr-xr-x  - ambal2 supergroup          0 2023-10-12 11:35 new
drwxr-xr-x  - ambal2 supergroup          0 2023-10-09 11:39 sqoop
drwxr-xr-x  - ambal2 supergroup          0 2023-10-09 12:41 sqoop1
DESKTOP2@DESKTOP2: ~ $
```

Inserting Data into HDFS

Assume we have data in the file called file.txt in the local system which is ought to be saved in the hdfs file system. Follow the steps given below to insert the required file in the Hadoop file system.

Step-2: Adding Files and Directories to HDFS

```
2023-10-10 11:37:33,243 WARN util.NativeCodeLoader: Unable to load native hadoop library for your platform... using builtin java classes where possible.
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input
```

Transfer and store a data file from local systems to the Hadoop file system using the put command.

```
$ $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /user/input
```

Step 3 :You can verify the file using ls command.

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/input
```

Step 4 Retrieving Data from HDFS

Assume we have a file in HDFS called outfile. Given below is a simple demonstration for retrieving the required file from the Hadoop file system.

Initially, view the data from HDFS using cat command.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/outfile
```

Get the file from HDFS to the local file system using get command.

```
$ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/
```

Step-5: Deleting Files from HDFS

```
$ hadoop fs -rm file.txt
```

Step 6:Shutting Down the HDFS

You can shut down the HDFS by using the following command.

```
$ stop-dfs.sh
```

Result

Thus the Installation of Hadoop in three operating modes has been successfully completed

Ex. No
3

Implementation of Matrix Multiplication with Hadoop Map Reduce

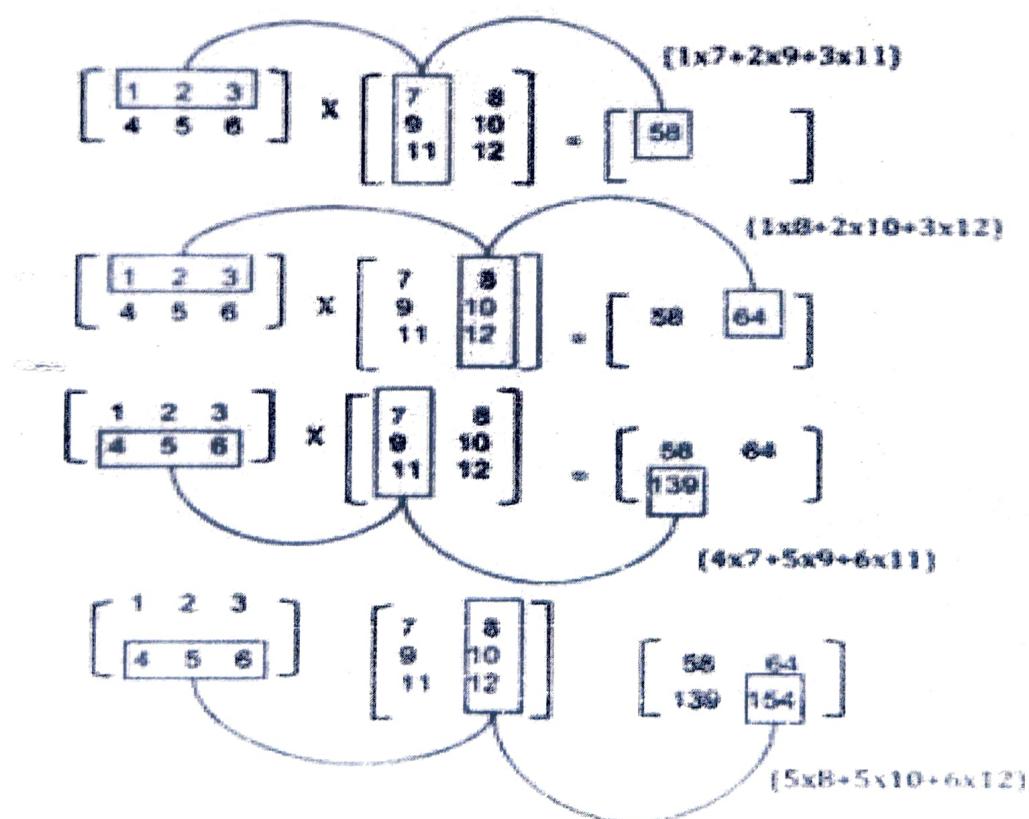
Date

Aim:

To Develop a Map Reduce program to implement Matrix Multiplication.

Theory:

In mathematics, **matrix multiplication** or the **matrix product** is a binary operation that produces a matrix from two matrices. The definition is motivated by linear equations and linear transformations on vectors, which have numerous applications in applied mathematics, physics, and engineering. In more detail, if **A** is an $n \times m$ matrix and **B** is an $m \times p$ matrix, their matrix product **AB** is an $n \times p$ 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 :

Algorithm for Map Function.

- a. for each element m_{ij} of M do
produce (key,value) pairs as $((i,k), (M,j,m_{ij}))$, for $k=1,2,3,..$ upto the number of columns of N
- b. for each element n_{jk} of N do
produce (key,value) pairs as $((i,k), (N,j,n_{jk}))$, for $i = 1,2,3,..$ Upto the number of rows of M .
- c. return Set of (key,value) pairs that each key (i,k) , has list with values
 (M,j,m_{ij}) and (N, j,n_{jk}) for all possible values of j .

Algorithm for Reduce Function.

- d. for each key (i,k) do
sort values begin with M by j in list M sort values begin with N by j in list N multiply m_{ij} and n_{jk}
for j th value of each list
sum up $m_{ij} \times n_{jk}$ return $(i,k), \sum_{j=1} m_{ij} \times n_{jk}$

procedure :

Step 1. Download the hadoop jar files with these links.

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

Step 2. Creating Mapper file for Matrix Multiplication.

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.ArrayList;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;
import org.apache.hadoop.util.ReflectionUtils;
class Element implements Writable {
    int tag;
    int index;
```

```
double value;
Element() {
    tag = 0;
    index = 0;
    value = 0.0;
}
Element(int tag, int index, double value) {
    this.tag = tag;
    this.index = index;
    this.value = value;
}
@Override
public void readFields(DataInput input) throws IOException {
    tag = input.readInt();
    index = input.readInt();
    value = input.readDouble();
}
@Override
public void write(DataOutput output) throws IOException {
    output.writeInt(tag);
    output.writeInt(index);
    output.writeDouble(value);
}
}
class Pair implements WritableComparable<Pair> {
    int i;
    int j;

    Pair() {
        i = 0;
        j = 0;
    }
    Pair(int i, int j) {
        this.i = i; this.j = j;
    }
    @Override
    public void readFields(DataInput input) throws IOException {
        i = input.readInt();
        j = input.readInt();
    }
}
```

```

}

@Override
public void write(DataOutput output) throws IOException { output.writeInt(i);
output.writeInt(j);
}
@Override
public int compareTo(Pair compare) { if (i > compare.i) {
return 1;
} else if ( i < compare.i) { return -1;
} else {
if(j > compare.j) {
return 1;
} else if (j < compare.j) { return -1;
}
}
return 0;
}
public String toString() { return i + " " + j + " ";
}
}

public class Multiply {
public static class MatriceMapperM extends Mapper<Object,Text,IntWritable,Element>
@Override
public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
String readLine = value.toString(); String[] stringTokens = readLine.split(",");
int index = Integer.parseInt(stringTokens[0]);
double elementValue = Double.parseDouble(stringTokens[2]); Element e = new Element(0,
index, elementValue); IntWritable keyValue = new
IntWritable(Integer.parseInt(stringTokens[1]));
context.write(keyValue, e);
}
}

public static class MatriceMapperN extends Mapper<Object,Text,IntWritable,Element> { @Override
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
String readLine = value.toString(); String[] stringTokens = readLine.split(",");
int index = Integer.parseInt(stringTokens[1]);
double elementValue = Double.parseDouble(stringTokens[2]); Element e = new Element(1,index,
elementValue);
IntWritable keyValue = new IntWritable(Integer.parseInt(stringTokens[0]));
}
}

```



```

        context.write(keyValue, e);
    }
}

public static class ReducerMxN extends Reducer<IntWritable, Element, Pair, DoubleWritable> {
    @Override
    public void reduce(IntWritable key, Iterable<Element> values, Context context) throws IOException, InterruptedException {
        ArrayList<Element> M = new ArrayList<Element>(); ArrayList<Element> N = new ArrayList<Element>();
        Configuration conf = context.getConfiguration(); for(Element element : values) {
            Element tempElement = ReflectionUtils.newInstance(Element.class,
            ReflectionUtils.copy(conf, element, tempElement)); if (tempElement.tag == 0) {
                M.add(tempElement);
            } else if(tempElement.tag == 1) { N.add(tempElement);
            }
        }
        for(int i=0;i<M.size();i++) {
            for(int j=0;j<N.size();j++) {
                Pair p = new Pair(M.get(i).index,N.get(j).index);
                double multiplyOutput = M.get(i).value * N.get(j).value;
                context.write(p, new DoubleWritable(multiplyOutput));
            }
        }
    }
}

public static class MapMxN extends Mapper<Object, Text, Pair, DoubleWritable> { @Override
    public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String readLine = value.toString(); String[] pairValue = readLine.split(" "); Pair p = new
        Pair(Integer.parseInt(pairValue[0]),Integer.parseInt(pairValue[1]));
        DoubleWritable val = new DoubleWritable(Double.parseDouble(pairValue[2]));
        context.write(p, val);
    }
}

public static class ReduceMxN extends Reducer<Pair, DoubleWritable, Pair, DoubleWritable> {
    @Override
    public void reduce(Pair key, Iterable<DoubleWritable> values, Context context) throws IOException, InterruptedException {
        double sum = 0.0; for(DoubleWritable value : values) {
            sum += value.get();
        }
        context.write(key, new DoubleWritable(sum));
    }
}

```

```

}

}

public static void main(String[] args) throws Exception { Job job = Job.getInstance();
job.setJobName("MapIntermediate"); job.setJarByClass(Project1.class);
MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, MatriceMapperM.class);
MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, MatriceMapperN.class);
    job.setReducerClass(ReducerMxN.class);
    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(Element.class);
    job.setOutputKeyClass(Pair.class);
    job.setOutputValueClass(DoubleWritable.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    FileOutputFormat.setOutputPath(job, new Path(args[2]));
    job.waitForCompletion(true);

    Job job2 = Job.getInstance();
    job2.setJobName("MapFinalOutput");
    job2.setJarByClass(Project1.class);

    job2.setMapperClass(MapMxN.class);
    job2.setReducerClass(ReduceMxN.class)

    job2.setMapOutputKeyClass(Pair.class);
    job2.setMapOutputValueClass(DoubleWritable.class);

    job2.setOutputKeyClass(Pair.class);
    job2.setOutputValueClass(DoubleWritable.class);

    job2.setInputFormatClass(TextInputFormat.class);
    job2.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.setInputPaths(job2, new Path(args[2]));
    FileOutputFormat.setOutputPath(job2, new Path(args[3]));

    job2.waitForCompletion(true);
}
}

```

Step 5. Compiling the program in particular folder named as operation

```

#!/bin/bash

rm -rf multiply.jar classes

module load hadoop/2.6.0 mkdir
-p classes

javac -d classes -cp classes:`$HADOOP_HOME/bin/hadoop classpath` Multiply.java jar
cf multiply.jar -C classes .

echo "end"

```

Step 6. Running the program in particular folder named as operation

```

export HADOOP_CONF_DIR=/home/$USER/cometcluster
module load hadoop/2.6.0

myhadoop-configure.sh
start-dfs.sh

```

start-yarn.sh

hdfs dfs -mkdir -p /user/\$USER

hdfs dfs -put M-matrix-large.txt /user/\$USER/M-matrix-large.txt
hdfs dfs -put N-matrix-large.txt /user/\$USER/N-matrix-large.txt

hadoop jar multiply.jar edu.uta.cse6331.Multiply /user/\$USER/M-matrix-large.txt

/user/\$USER/N-matrix-large.txt /user/\$USER/intermediate /user/\$USER/output
rm -rf output-distr

mkdir output-distr

hdfs dfs -get /user/\$USER/output/part* output-distr

stop-yarn.sh

stop-dfs.sh

myhadoop-cleanup.sh

Expected Output:

module load hadoop/2.6.0 rm -rf output intermediate

hadoop --config \$HOME jar multiply.jar edu.uta.cse6331.Multiply M-matrix-small.txt N-matrix- small.txt
intermediate output

Result:

Thus the Matrix Multiplication with Hadoop Map Reduce was successfully executed.

Ex. No	Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm	Date
4		

Aim:

Implementing distinct word count problem using Map-Reduce.

Procedure:

The function of the mapper is as follows:

- Create a IntWritable variable 'one' with value as 1
- Convert the input line in Text type to a String
- Use a tokenizer to split the line into words
- Iterate through each word and form key value pairs as

Assign each word from the tokenizer (of String type) to a Text 'word'

- Form key value pairs for each word as < word, one > and push it to the output collector

The function of Sort and Group:

After this, "aggregation" and "Shuffling and Sorting" done by framework. Then

Reducers task these final pair to produce output.

The function of the reducer is as follows

- Initialize a variable 'sum' as 0
- Iterate through all the values with respect to a key and sum up all of them
- Push to the output collector the Key and the obtained sum as value



Program:

For the given sample input1 data file (input1.txt : Hello World Bye World) mapper emits:

```
<Hello, 1>
<World, 1>
<Bye, 1>
<World, 1>
```

The second input2 data file (input2.txt : Hello Hadoop Goodbye Hadoop) mapper emits:

```
<Hello, 1>
<Hadoop, 1>
<Goodbye, 1>
<Hadoop, 1>
```

- WordCount also specifies a combiner. Hence, the output of each map is passed through the local combiner (which is same as the Reducer as per the job configuration) for local aggregation, after being sorted on the keys.

The output of the first map:

```
<Bye, 1>
```

<Hello, 1>

<World, 2>

The output of the second map:

<Goodbye, 1>

<Hadoop, 2>

<Hello, 1>

The Reducer implementation via the reduce method just sums up the values, which are the occurrence counts for each key (i.e. words in this example).

Thus the output of the job is:

<Bye, 1>

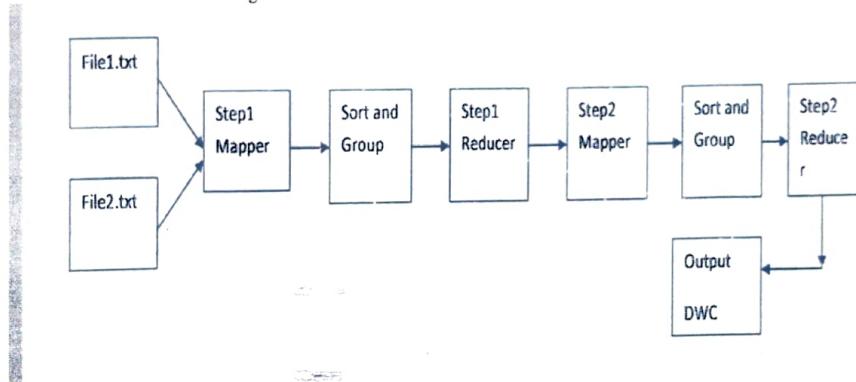
<Goodbye, 1>

<Hadoop, 2>

<Hello, 2>

<World, 2>

Data flow diagram :



Result:

Thus the Word Count Map Reduce program to understand Map Reduce Paradigm and verified it by running an example program successfully .

Ex. No	Installation of Hive along with practice examples.	Date
5		

Aim:

To install hive with proper procedure.

Steps for hive installation

- Download and Unzip Hive
- Edit .bashrc file
- Edit hive-config.sh file
- Create Hive directories in HDFS
- Initiate Derby database
- Configure hive-site.xml file

Step 1:

download and unzip Hive

=====

```
wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
tar xzf apache-hive-3.1.2-bin.tar.gz
```

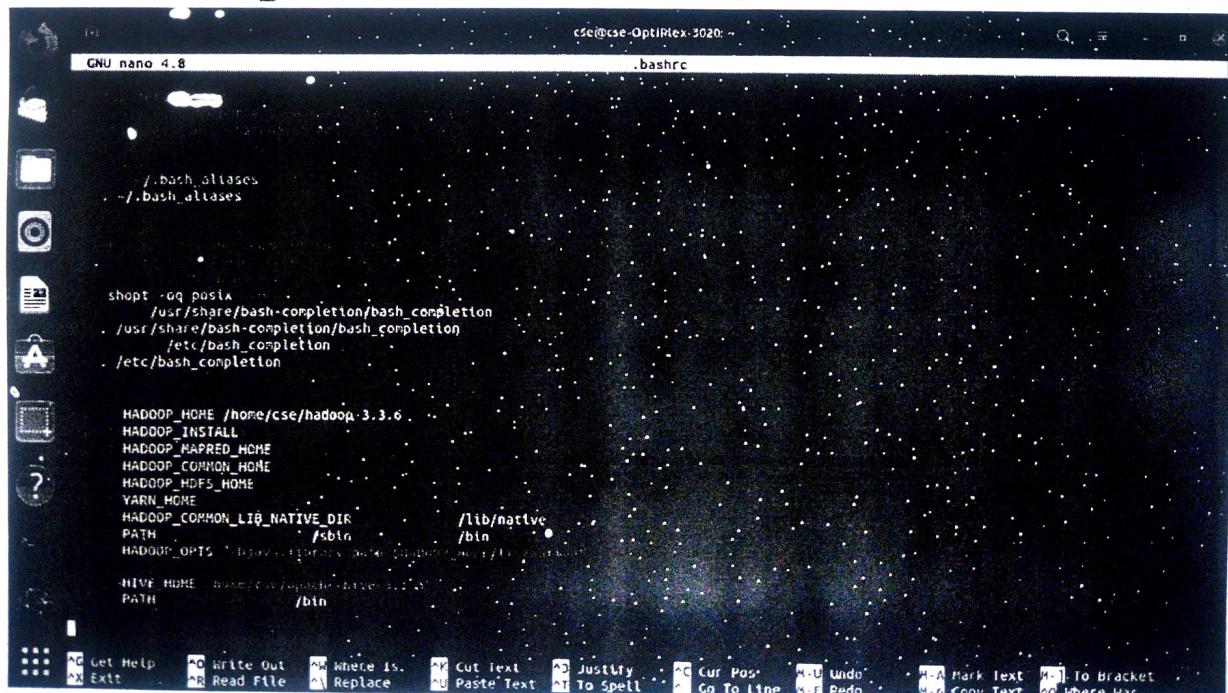
Step 2:

Edit .bashrc file

=====

```
sudo nano .bashrc
```

```
export HIVE_HOME= /home/hadoop/apache-hive-3.1.2-bin
export PATH=$PATH:$HIVE_HOME/bin
```



```
cse@cse-OptiPlex-3020: ~
GNU nano 4.8
.bashrc

./.bash_aliases
./.bash_aliases

shopt -q posix
/usr/share/bash-completion/bash_completion
/usr/share/bash-completion/bash_completion
/etc/bash_completion
/etc/bash_completion

HADOOP_HOME /home/cse/hadoop-3.3.0
HADOOP_INSTALL
HADOOP_MAPRED_HOME
HADOOP_COMMON_HOME
HADOOP_HDFS_HOME
YARN_HOME
HADOOP_COMMON_LIB_NATIVE_DIR /lib/native
PATH /sbin /bin
HADOOP_OPTS -Djava.library.path=$HADOOP_HOME/lib

HIVE_HOME /home/cse/apache-hive-3.1.2-bin
PATH /bin

Get Help F1 Get Help F1
W3 Write Out W3 Where Is W3 Cut Text W3 Justify W3 Cur Pos F2 Undo
R3 Read File R3 Replace W3 Paste Text W3 To Spell F2 Go To Line W3 Redo
H-A Mark Text H-A To Bracket H-A Copy Text H-A Where Was
```

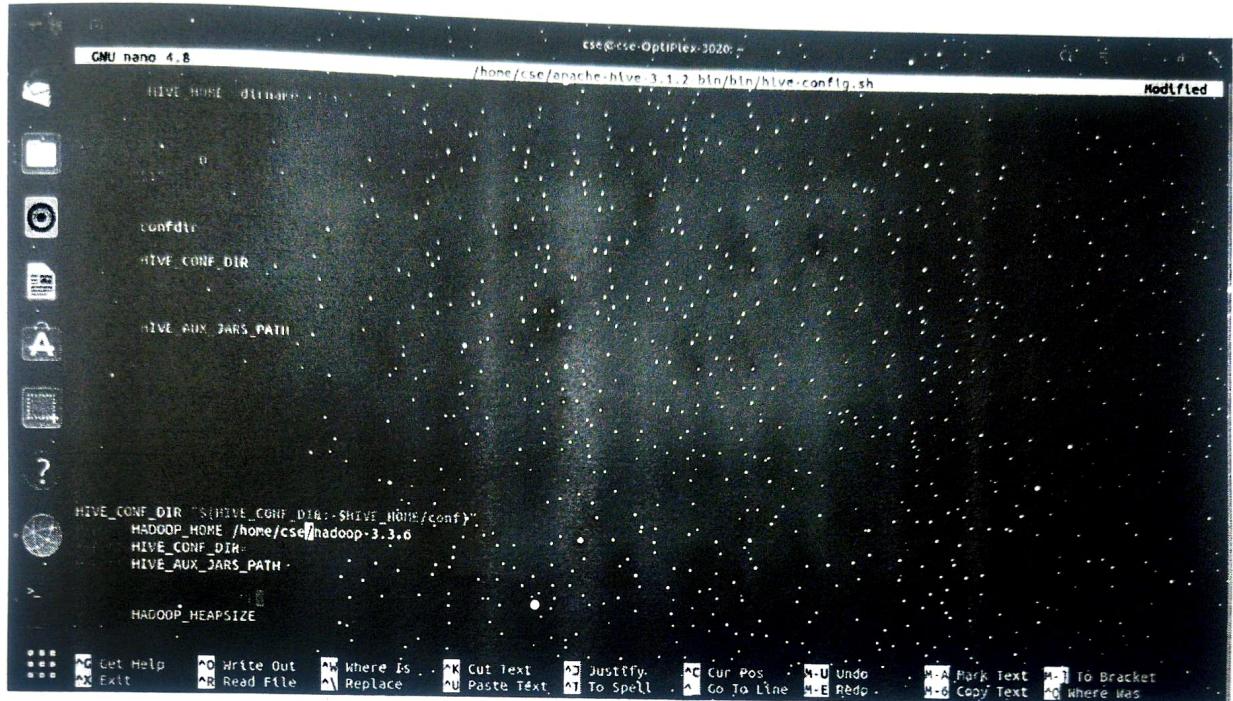
Step 3:

```
source ~/.bashrc
```

step 4:

Edit `hive-config.sh` file

```
=====  
sudo nano $HIVE_HOME/bin/hive-config.sh  
export  
HADOOP_HOME=/home/cse/hadoop-3.3.6
```



```
GNU nano 4.8  
cse@cse-OptiPlex-3020: /home/cse/apache-hive-3.1.2/bin/hive-config.sh  
HIVE_HOME /hadoop  
HIVE_CONF_DIR  
HIVE_AUX_JARS_PATH  
HADOOP_HEAPSIZE  
HIVE_CONF_DIR "$HIVE_CONF_DIR:$HIVE_HOME/conf"  
HADOOP_HOME /home/cse/hadoop-3.3.6  
HIVE_CONF_DIR  
HIVE_AUX_JARS_PATH  
HADOOP_HEAPSIZE  
Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo M-A Mark Text M-L To Bracket  
Exit Read File Replace Paste Text M-T To Spell M-G Go to Line M-E Redo M-B Copy Text M-Q Where Was
```

step 5:

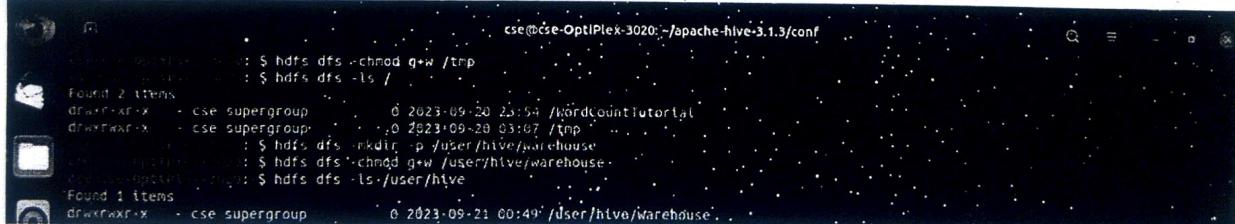
Create Hive directories in HDFS

```
=====  
hdfs dfs -mkdir /tmp
```

```
hdfs dfs -chmod g+w /tmp
```

```
hdfs dfs -mkdir -p /user/hive/warehouse hdfs
```

```
dfs -chmod g+w /user/hive/warehouse
```



```
hdfs dfs -chmod g+w /tmp  
hdfs dfs -ls /tmp  
Found 2 items  
drwxrwxr-x 2 cse supergroup 0 2023-09-20 23:54 /tmp  
drwxrwxr-x 2 cse supergroup 0 2023-09-20 03:07 /tmp/hive  
hdfs dfs -mkdir -p /user/hive/warehouse  
hdfs dfs -chmod g+w /user/hive/warehouse  
hdfs dfs -ls /user/hive  
Found 1 items  
drwxrwxr-x 2 cse supergroup 0 2023-09-21 00:49 /user/hive/warehouse
```

step 6:

Fixing guava problem – Additional step

```
=====  
rm $HIVE_HOME/lib/guava-19.0.jar
```

```
cp $HADOOP_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE_HOME/lib/
```



step 7: Configure hive-site.xml File (Optional)

Use the following command to locate the correct file:

```
cd $HIVE_HOME/conf
```

List the files contained in the folder using the ls command.

Use the hive-default.xml.template to create the hive-site.xml file:

```
cp hive-default.xml.template hive-site.xml
```

Access the hive-site.xml file using the nano text editor:

```
sudo nano hive-site.xml
```

```
DERBY
=====
  Expects one of [derby, oracle, mysql, mssql, postgres].
  Type of database used by the metastore. Information schema
  JDBCSto>

  hive.metastore.warehouse.dir
  /user/hive/warehouse
  location of default database for the warehouse

  hive.metastore.warehouse$external.dir
  Default location for 'external' tables created in the warehouse>

  hive.metastore.uris
  Thrift URI for the remote metastore. Used by metastore client>
```

Step 8: Initiate Derby Database

```
=====
$HIVE_HOME/bin/schematool -dbType derby -initSchema
```

```
celarce-OptiPlex-3020:~$apache-hive-3.1.2-bin/bin
```

```
Initialization script completed
schemaout completed
: $ cd $HIVE_HOME/bin
: $ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [/jar@file:/home/cse/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4jimpl/StaticLoggerBinder.class]
SLF4J: Found binding in [/jar@file:/home/cse/hadoop-2.3.0/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4jimpl/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.LoggerFactory]
Hive Session ID = c04c2bee-1ccc-4de6-95de-2cb5ead5b114
Logging initialized using configuration in jar@file:/home/cse/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.properties.async
Hive Session ID = 1e81fd71-6e47-4e90-a7f9-efddab623b4d
Hive on BE is deprecated in Hive 2.0 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, t
hive> 
```



Result:

Thus The Real Hive Installed Successfully .

Ex. No	Installation of Hive With Example			Date
5(B)				

Aim:

To install hive along with practical example.

Procedure:

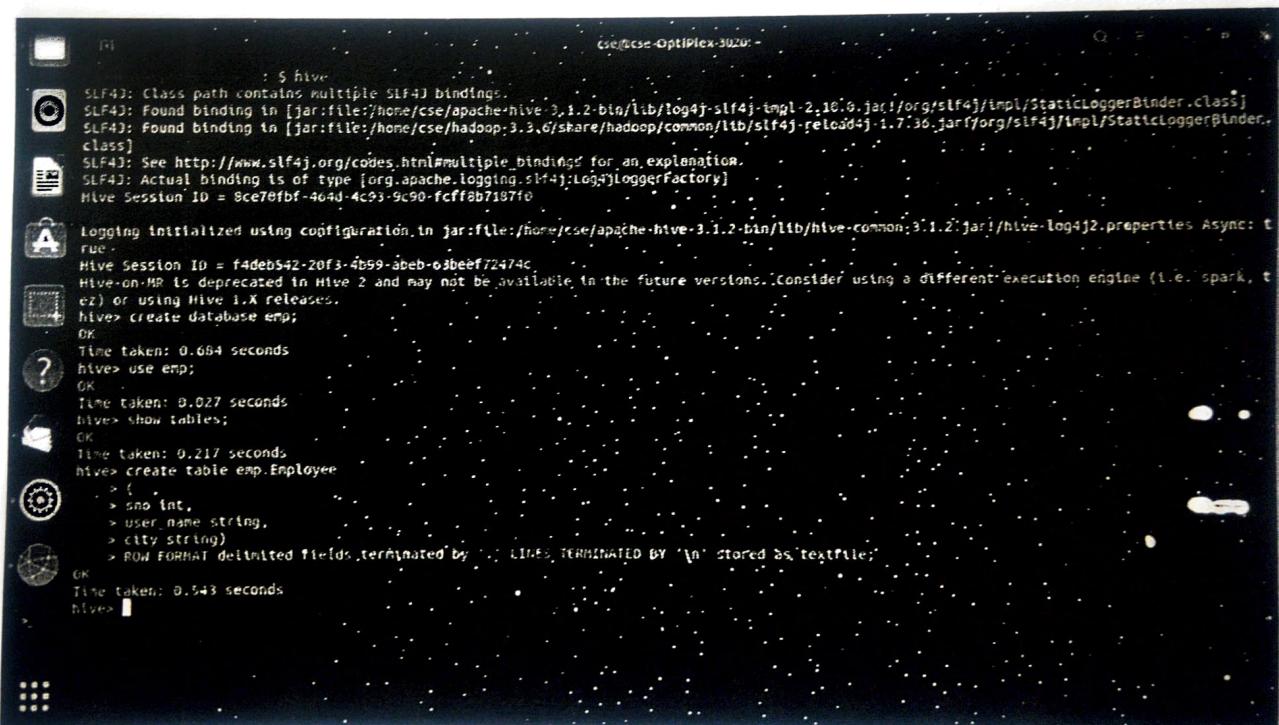
Create Database from Hive Beeline shell

1. Create database database_name; Ex:

>Create database Emp;

>use Emp;

>create table emp.employee(sno int,user String,city String)Row format delimited fields terminated by '/n' stored as textfile;



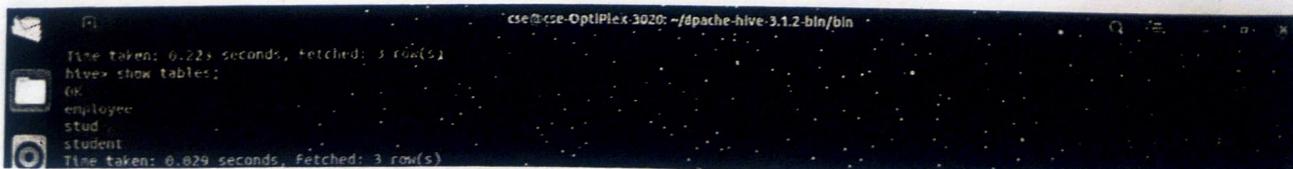
```

: S hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/cse/apache-hive-3.1.2-bin/lib/log4j-slf4j-tmlg1-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/cse/hadoop-3.3.6/share/hadoop/common/lib/slf4j-reload4j-1.7.30.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;LoggerFactory]
Hive Session ID = 8ce70fbf-464d-4c93-9c90-fcff8b7187f0

Logging initialized using configuration in jar:file:/home/cse/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.properties Async: true
Hive Session ID = f4debs42-2013-4b59-8beb-03beef72474c
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive> create database emp;
OK
Time taken: 0.084 seconds
hive> use emp;
OK
Time taken: 0.027 seconds
hive> show tables;
OK
Time taken: 0.217 seconds
hive> create table emp.Employee
> {
> sno int,
> user name string,
> city string
> ROW FORMAT delimited fields terminated by '/n' LINES TERMINATED BY '\n' stored as textfile;
OK
Time taken: 0.543 seconds
hive>

```

Show Database



```

cse@cse-OptiPlex-3020: ~/apache-hive-3.1.2-bin/bin
Time taken: 0.223 seconds, Fetched: 3 row(s)
hive> show tables;
OK
employee
stud
student
Time taken: 0.029 seconds, Fetched: 3 row(s)

```

Result:

Thus hive was successfully installed and executed with example

Ex. No	Installation of HBase, Installing thrift along with Practice examples	Date
6(A)		

Aim:

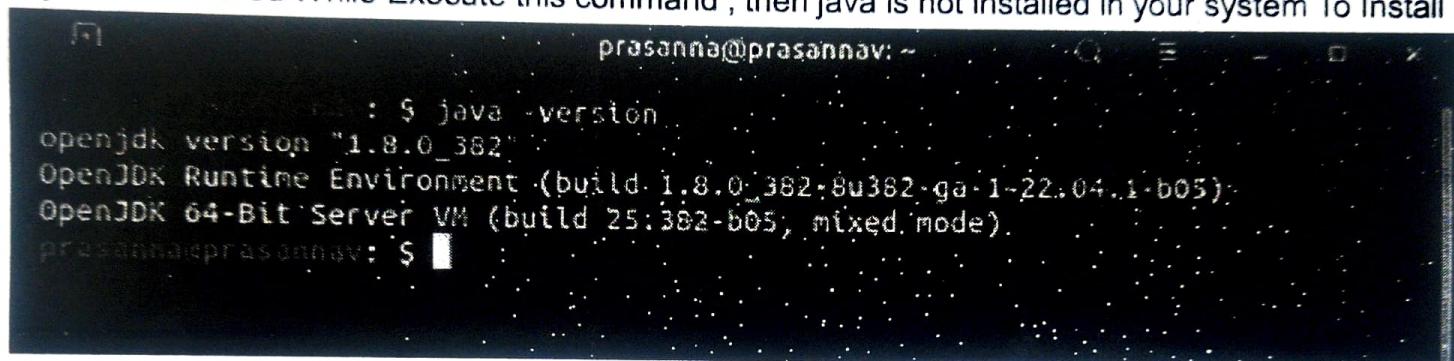
To Install H Base on Ubuntu 18.04 HBase

Procedure:**Pre-requisite:**

Ubuntu 16.04 or higher installed on a virtual machine.

Step-1: Make sure that java has installed in your machine to verify that run java -version

If any Error Occured While Execute this command , then java is not installed in your system To Install Java

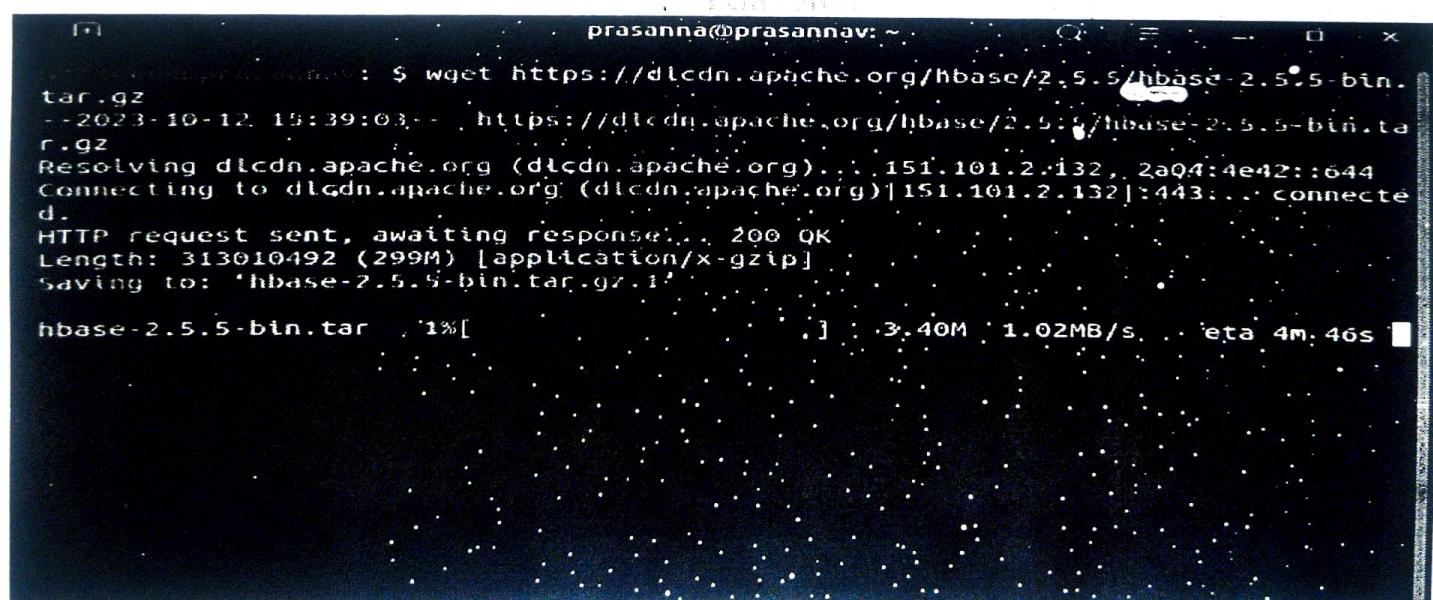


```
prasanna@prasannav: ~
: $ java -version
openjdk version "1.8.0_382"
OpenJDK Runtime Environment (build 1.8.0_382-bu382-ga-1-22.04.1-b05)
OpenJDK 64-Bit Server VM (build 25:382-b05, mixed mode)
prasanna@prasannav: $
```

sudo apt install openjdk-8-jdk -y

Step-2: Download Hbase

wget <https://dlcdn.apache.org/hbase/2.5.5/hbase-2.5.5-bin.tar.gz>



```
prasanna@prasannav: ~
: $ wget https://dlcdn.apache.org/hbase/2.5.5/hbase-2.5.5-bin.tar.gz
--2023-10-12 19:39:03-- https://dlcdn.apache.org/hbase/2.5.5/hbase-2.5.5-bin.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 313010492 (299M) [application/x-gzip]
Saving to: 'hbase-2.5.5-bin.tar.gz.1'

hbase-2.5.5-bin.tar 1%[          ] 3.40M 1.02MB/s eta 4m 46s
```

Step-3:Extract The hbase-2.5.5-bin.tar.gz file by using the command tar xvf hbase-2.5.5-bin.tar.gz

step-4: goto hbase2.5.5/conf folder and open hbase-env.sh file

```
[root@pracanner ~]# tar -xvf hbase-2.5.5-bin.tar.gz
[root@pracanner ~]# cd hbase-2.5.5/
[root@pracanner ~]# cd conf/
[root@pracanner ~]# ls
hadoop-metrics2-hbase.properties  hbase-policy.xml  log4j2.properties
hbase-env.cmd                      hbase-site.xml   regionservers
hbase-env.sh                        log4j2-hbttop.properties
[root@pracanner ~]# gedit hbase-env.sh
```

step-5 : Edit .bashrc file

and then open .bashrc file and mention HBASE_HOME path as shown in

```
below export HBASE_HOME=/home/prasanna/hbase-2.5.5
```

here you can change name according to your local machine name

eg : export

HBASE_HOME=/home/<your machine name>/hbase-2.5.5

```
export PATH= $PATH:$HBASE_HOME/bin
```

Note:*make sure that the hbase-2.5.5 folder in home directory before setting HBASE_HOME path

, if not then move the hbase-2.5.5 file to home directory*

prasanna@prasannav: ~

GNU nano 6.2 .bashrc

```
~/.bash_aliases
~/.bash_aliases

shopt -oq posix
/usr/share/bash-completion/bash_completion
/usr/share/bash-completion/bash_completion
/etc/bash_completion
/etc/bash_completion

HBASE_HOME /home/prasanna/hbase-2.5.5
PATH . /bin

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line
```

step-6 : Add properties in the hbase-site.xml

```
prasanna@prasannav: $ cd hbase-2.5.5/conf
prasanna@prasannav: $ gedit hbase-site.xml
```

put the below property between the <configuration></configuration> tag

```
<property>
<name>hbase.rootdir</name>
<value>file:///home/prasanna/HBASE/hbase</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/prasanna/HBASE/zookeeper</value>
</property>
```

step-7: Goto To /etc/ folder and run the following command and configure

```
$ gedit hosts
```

```

  Open File Save File Hosts (Read-Only)
  1 127.0.0.1 localhost
  2 127.0.0.1 prasannav.myguest.virtu
  3 localhost
  4 prasannav.myguest.virtu
  5 # The following lines are desirable for IPv6 capable hosts
  6 fe80::1%lo0 ip6-localhost-ipv6
  7 fe80::1%lo0 ip6-loopback
  8 fe80::1%lo0 ip6-localhost
  9 fe80::1%lo0 ip6-mcastprefix
  10 fe80::1%lo0 ip6-allnodes
  11 fe80::1%lo0 ip6-allrouters
  12 fe80::1%lo0 ip6-allrouters
  Plain text Tab width: 8 Line 2 Col 9 INS

```

change in line no-2 by default the ip is 127.0.1.1

change it to 127.0.0.1 in second line only
step-8:starting hbase
goto hbase-2.5.5/bin folder

```

$ cd hbase-2.5.5/bin/
$ ./start-hbase.sh
running master, logging to /home/prasanna/hbase-2.5.5/logs/hbase-prasanna-master-prasannav.out
$ 

```

After this run jps command to ensure that hbase is running

```

prasanna@prasannav: ~ $ jps
5126 Jps
4729 HMaster
prasanna@prasannav: ~ $ 

```

run <http://localhost:16010> to see hbase web UI

The screenshot shows the Apache HBase web interface. The top navigation bar includes links for Welcome to HBase, Firefox Private, hadoop install, How To Install, HBase Community, Apache HBase, and Master: localhost. The main content area has tabs for Home, Table Details, Procedures & Locks, HBase Report, Operation Details, Process Metrics, Local Logs, Log Level, Debug Dump, Metrics Dump, and Prefilter. The Home tab is selected. The 'Region Servers' section displays a table with columns: ServerName, Start time, Last contact, Version, Requests Per Second, and Num. Regions. One row is shown for 'localhost:16020, 1697107453860' with values: Thu Oct 12 16:14:13 IST 2023, 2.8, 2.5.5, 0, 4. The 'Backup Masters' section shows a table with columns: ServerName, Port, and Start Time. The 'Tables' section shows a table with columns: Namespace, Name, State, OPEN, OPENING, CLOSED, CLOSING, OFFLINE, SPLIT, Other, and Description. One row is shown for 'default' with values: default, hbase, ENABLED, 1, 0, 0, 0, 0, 0, 0, and 'p. (TABLE_ATTRIBUTES -> (METADATA -> (hbase store file tracker.impl -> DEFAULT!)) (NAME -> c))'.

Backup Masters

ServerName	Port	Start Time
localhost:16020, 1697107453860		
Total: 1		

Tables

Dimensions		System Tables		Schemas						
2 table(s) in set [GetTable]		Click count below to see list of regions currently in 'state' designated by the column title. For 'Other' Region state, browse to hbase:meta and adjust filter on 'Meta Entries' to query on states other than those listed here. Queries may take a while if the hbase:meta table is large.								
Namespace	Name	State	OPEN	OPENING	CLOSED	CLOSING	OFFLINE	SPLIT	Other	Description
default	hbase	ENABLED	1	0	0	0	0	0	0	p. (TABLE_ATTRIBUTES -> (METADATA -> (hbase store file tracker.impl -> DEFAULT!)) (NAME -> c))

step-9: accessing hbase shell by running ./hbase shell command

```
Prasanna@prasannev:~/hbase-2.5.5/bin$ ./hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.5.5, r7ebd4381261fef78fc2acf258a95184f4147ce, Thu Jun 1 17:42:49 PDT 2023
Took 0.0011 seconds
hbase:001:0>
```

Result:

HBase was successfully installed on Ubuntu 18.04

Ex. No	HBase, Installing thrift along with Practice examples	Date
6(B)		

Aim:

To Installation HBase on Ubuntu 18.04.

EXAMPLE

1) To create Table syntax:

create 'Table_Name','col_fam_1','col_fam_1', ... 'col_fam-n'

code :

create 'aamec','dept','year'

2) List All Tables

code :

list

```
hbase:010:0> list
TABLE
aamec
amazon
college
prasanna
table_name
5 row(s)
Took 0.0133 seconds
=> ["aamec", "amazon", "college", "prasanna", "table_name"]
hbase:011:0>
```

3) insert

data

syntax:

put

'table_name','row_key','column_family:attribute','value' here

row_key is a unique key to retrive data

code :

this data will enter data into the dept column family

put

'aamec','cse','dept:studentname','prasan

na' put 'aamec','cse','dept:year','third'

put 'aamec','cse','dept:section','A'

```
hbase:008:0> put 'aamec','cse','dept:studentname','prasanna'  
Took 0.0240 seconds  
hbase:009:0> put 'aamec','cse','dept:year','third'  
Took 0.0072 seconds  
hbase:010:0> put 'aamec','cse','dept:section','A'  
Took 0.0342 seconds  
hbase:011:0>
```

This data will enter data into the year column family

```
put 'aamec','cse','year:joinedyear','2021' put  
'aamec','cse','year:finishingyear','2025'
```

```
hbase:011:0> put 'aamec','cse','year:joinedyear','2021'  
Took 0.0739 seconds  
hbase:012:0> put 'aamec','cse','year:finishingyear','2025'  
Took 0.0411 seconds  
hbase:013:0>
```

4) Scan Table

same as desc in RDBMS
syntax:

```
scan 'table_name'  
code:  
scan 'aamec'
```

```
hbase:022:0> scan 'aamec'  
ROW  
  cse  
  cse  
  cse  
  cse  
  cse  
  it  
  it  
  it  
  it  
  it  
 2 row(s)  
Took 0.0810 seconds  
hbase:023:0>
```

5) To get specific data

syntax:

```
get 'table_name','row_key',[optional column family: attribute]
```

code :

```
get 'aamec','cse'
```

```
hbase:023:0> get 'aamec','cse'  
COLUMN  
dept:section  
dept:studentname  
dept:year  
year:finishingyear  
year:joinedyear  
1 row(s)  
Took 0.0908 seconds  
CELL  
timestamp=2023-10-13T12:14:26.734, value=A  
timestamp=2023-10-13T12:13:11.914, value=prasanna  
timestamp=2023-10-13T12:13:41.018, value=third  
timestamp=2023-10-13T12:16:57.291, value=2025  
timestamp=2023-10-13T12:16:41.876, value=2021
```

1. update table value

The same put command is used to update the table value ,if the row key is already present in the database then it will update data according to the value ,if not present the it will create new row with the given row key

```
hbase:025:0> put 'aamec','cse','dept:section','B'  
Took 0.0134 seconds  
hbase:026:0>
```

previously the value for the section in cse is A ,But after running this command the value will be changed into B

```
hbase:026:0> get 'aamec','cse'  
COLUMN  
dept:section  
dept:studentname  
dept:year  
year:finishingyear  
year:joinedyear  
1 row(s)  
Took 0.0506 seconds  
hbase:027:0>
```

7) To Delete Data

syntax:

```
delete 'table_name','row_key','column_family:attribute'
```

code :

```
delete 'aamec','cse','year:joinedyear'
```

```
Took 0.0056 seconds  
hbase:027:0> delete 'aamec','cse','year:joinedyear'  
Took 0.0138 seconds  
hbase:028:0> get 'aamec','cse'  
COLUMN  
dept:section  
dept:studentname  
dept:year  
year:finishingyear  
1 row(s)  
Took 0.0686 seconds  
hbase:028:0>
```



Result:

HBase was successfully installed with an example on Ubuntu 18.04.

Ex. No

7

Practice importing and exporting data from various databases.

Date

Aim:

To import and export, data from MySQL and Hive

Procedure:

Step 1: To start hdfs

```
root@ambal2: ~ $ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as ambal2 in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost].
Starting datanodes
```

Step 2: MySQL Installation

```
root@ubuntu:/home/ambal2# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.34-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

sudo apt install mysql-server (use this command to install MySQL server)

COMMANDS:**-\$ sudo su**

After this enter your linux user password, then the root mode will be open here we don't need any authentication for mysql.

~root\$ mysql

Creating user profiles and grant them permissions:

```
Mysql> CREATE USER 'bigdata'@'localhost' IDENTIFIED BY 'bigdata';
Mysql> grant all privileges on *.* to bigdata@localhost;
```

Note: This step is not required if you just use the root user to make CRUD operations in the MySQL

Mysql> CREATE USER 'bigdata'@'127.0.0.1' IDENTIFIED BY 'bigdata'; Mysql>grant all privileges on *.* to bigdata@127.0.0.1;

Note: Here, *.* means that the user we create has all the privileges on all the tables of all the databases.

Now, we have created user profiles which will be used to make CRUD operations in the mysql

Step 3: Create a database and table and insert data.

Example:

```
create database Employe;
```

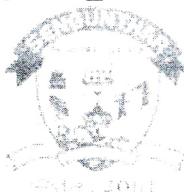
```
create table Employe.Emp(author_name varchar(65), total_no_of_articles int, phone_no int, address varchar(65));
```

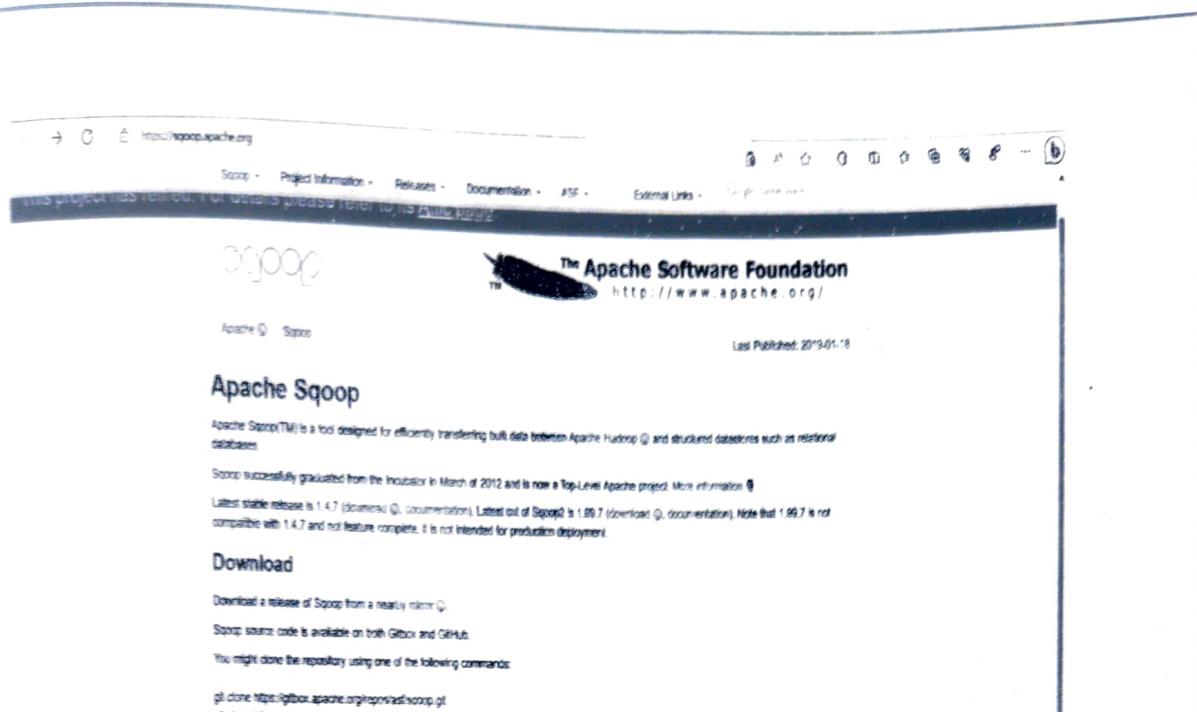
```
insert into Emp values("Rohan",10,123456789,"Lucknow");
```

Step 3: Create a database and table in the hive where data should be imported.

```
create table geeks_hive_table(name string, total_articles int, phone_no int, address string) row format delimited fields terminated by ' ';
```

Step 4: SQOOP INSTALLATION :





Apache Sqoop

Apache Sqoop™ is a tool designed for efficiently transferring bulk data between Apache Hadoop® and structured databases such as relational databases.

Sqoop successfully graduated from the Incubator in March of 2012 and is now a Top-Level Apache project. More information [here](#).

Latest stable release is 1.4.7 (downloads, documentation). Latest dev of Sqoop is 1.99.7 (downloads, documentation). Note that 1.99.7 is not compatible with 1.4.7 and not feature complete. It is not intended for production deployment.

Download

Download a release of Sqoop from a nearby mirror [here](#).

Sqoop source code is available on both GitBox and GitHub.

You might clone the repository using one of the following commands:

```
git clone https://gitbox.apache.org/repos/asf/sqoop.git  
git clone https://github.com/apache/sqoop.git
```

Use one of the following links to browse the repository online:

<https://gitbox.apache.org/repos/asf/sqoop.git> 
<https://github.com/apache/sqoop> 

Getting Involved



Read-only Resource	Link
Website	sqoop.apache.org
Existing List Archives	sqoop.apache.org/list-archives.html
Issue Tracker (JIRA)	sqoop.apache.org/jira
Search Results	sqoop.apache.org/search.html
Downloads	sqoop.apache.org/Downloads.html

The Apache Altic

- Home
- The team
- Process
- Process tracking
- Board Minutes
- Licence
- Security
- Policy Policy

Recent Apache Links

- Decaf
- Thrift
- Incubator
- ApacheCon

Projects in the Altic

- Accumulo
- ACC
- Avy21
- Avro
- Aurora
- Avalon
- Axis
- Axis2
- Axis Sandesh/BC
- Axis Service
- Axis2 Sandesh
- Beehive
- Blitz
- Chemistry
- Chukwa
- Clerezza
- Clueck
- Open Cloud Workbench
- Common
- Confluence
- Craxo
- Detached
- DeviceMap
- DirectMemory
- DRIFT
- Eagle
- EMR
- Etch
- Escolar
- Falcon
- Graph
- Hama
- Hama

After downloading the sqoop , go to the directory where we downloaded the sqoop and then extract it using the following command :

```
$ tar -xvf sqoop-1.4.4-bin_hadoop-2.0.4-alpha.tar.gz
```

Then enter into the super user : \$ su

Next to move that to the usr/lib which requires a super user privilege

```
$ mv sqoop-1.4.4-bin_hadoop-2.0.4-alpha /usr/lib/sqoop
```

Then exit : \$ exit

```
Goto .bashrc: $ sudo nano .bashrc , and then add the following export  
SQOOP_HOME=/usr/lib/sqoop  
export PATH=$PATH:$SQOOP_HOME/bin  
$ source ~/.bashrc
```

Then configure the sqoop, goto the directory of the config folder of sqoop_home and then move the contents of template file to the environment file.

```
$ cd $SQOOP_HOME/conf  
$ mv sqoop-env-template.sh sqoop-env.sh
```

Then open the sqoop-environment file and then add the following, export
HADOOP_COMMON_HOME=/usr/local/Hadoop
export HADOOP_MAPRED_HOME=/usr/local/hadoop

Note : Here we add the path of the Hadoop libraries and files and it may different from the path which we mentioned here. So, add the Hadoop path based on your installation.

Step 5: Download and Configure mysql-connector-java :

We can download mysql-connector-java-5.1.30.tar.gz file from the following [link](#).

ext. to extract the file and place it to the lib folder of sqoop

```
$ tar -zxf mysql-connector-java-5.1.30.tar.gz  
$ su  
$ cd mysql-connector-java-5.1.30  
$ mv mysql-connector-java-5.1.30-bin.jar /usr/lib/sqoop/lib
```

Note : This is library file is very important don't skip this step because it contains the libraries to connect the mysql databases to jdbc.

Verify sqoop: sqoop-version Step 3: hive database Creation

```
hive> create database sqoop_example; hive>use sqoop_example;  
hive>create table sqoop(usr_name string,no_ops int,ops_names string);
```

Hive commands much more alike mysql commands. Here, we just create the structure to store the data which we want to import in hive.

```
hive> show databases;
OK
default
sqoop
Time taken: 0.683 seconds, Fetched: 2 row(s)
hive> use sqoop;
OK
Time taken: 0.08 seconds
hive> show tables;
OK
bigdata
sqoop
Time taken: 0.148 seconds, Fetched: 2 row(s)
hive> create table dell(mdl_name string,mdl_num int);
OK
Time taken: 2.564 seconds
hive>
```

Step 6: Importing data from MySQL to hive :

```
sqoop import --connect jdbc:mysql://127.0.0.1:3306/database_name_in_mysql \
--username root --password cloudera \
--table table_name_in_mysql \
--hive-import --hive-table database_name_in_hive.table_name_in_hive \
--m 1
```

```
Activities 0 tasks
10:12:12 03 2016
ambalz@ubuntu:~
```

```
: $ sqoop import --connect jdbc:mysql://127.0.0.1:3306/sqoop-example --username dell --password dell --table dell --hive-import --hive-table sqoop.dell --m 1
```

```
-m 1
```

```

Job Counters
  Launched map tasks=1
  other local map tasks=1
  total time spent by all maps in occupied slots (ms)=8912
  total time spent by all reduces in occupied slots (ms)=0
  Total time spent by all map tasks (ms)=8912
  Total vcore-milliseconds taken by all map tasks=8912
  Total megabyte-milliseconds taken by all map tasks=9125888

Map-Reduce Framework
  Map input records=1
  Map output records=3
  Input split bytes=1024
  Spilled Records=0
  Failed Shuffles=0
  Serged Map outputs=0
  GC time elapsed (ms)=141
  CPU time spent (ms)=2800
  Physical memory (bytes) snapshot=223499740
  Virtual memory (bytes) snapshot=2542714880
  Total committed heap usage (bytes)=130839168
  Peak Map Physical memory (bytes)=223499240
  Peak Map Virtual memory (bytes)=2542714880

File Input Format Counters
  Bytes Read=0

File Output Format Counters
  Bytes Written=43

2023-10-12 12:15:07,009 INFO mapreduce.ImportJobBase: Transferred 43 bytes in 31,8907. seconds (1.3484 bytes/sec)
2023-10-12 12:15:07,047 INFO mapreduce.ImportJobBase: Retrieved 3 records
Thu Oct 12 12:15:07 IST 2023 WARN: Establishing SSL connection without server's identity verification is not recommended. According
is stated by default if explicit option isn't set. For compliance with existing applications not using SSL, the verifyServerCertificate
useSSL=false, or set useSSL=true and provide truststore for server certificate verification.
2023-10-12 12:15:07,188 INFO manager.SqlManager: Executing 'SQL' statement: SELECT * FROM dell AS t1 LIMIT 1
2023-10-12 12:15:07,282 INFO hive.HiveImport: Loading uploaded data into Hive
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Class path contains multiple SLF4J bindings.
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Found binding in [jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/jul/StaticLoggerBinder.class]
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Found binding in [jar:file:/home/ambal2/hadoop-3.2.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/jul/StaticLoggerBinder.class]
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Found binding in [jar:file:/home/ambal2/hadoop-3.2.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/jul/StaticLoggerBinder.class]
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
2023-10-12 12:15:09,557 INFO hive.HiveImport: SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2023-10-12 12:15:09,563 INFO hive.HiveImport: SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

```

OUTPUT:

```

$ ./sqoop-import -m 1 --hive-table dell --hive-import --hive-overwrite --hive-conf /home/ambal2/.hiveconf --hive-hostname localhost
SLF4J: Class path contains multiple SLF4J Bindings.
SLF4J: Found binding in [jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/jul/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/ambal2/hadoop-3.2.3/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/jul/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]
hive Session ID = ca95a42a-a65e-4d00-948a-c435099df78f

Logging initialized using configuration in jar:file:/home/ambal2/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j.properties Async: true
hive Session ID = a1776f23-cf63-4313-a2c9-e3bc02cb423e
hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
hive> show databases;
OK
default
hive>
Time taken: 0.573 seconds, Fetched: 2 row(s)
hive> use default;
OK
hive>
Time taken: 0.073 seconds
hive> select * from dell;
OK
Inspiron      3505
alienware    5005
Inspiron      3550
Time taken: 3.087 seconds, Fetched: 3 row(s)
hive>

```

Result:

Thus the import and export, the order of columns in MySQL queries are exported to hive successfully