PRACTICAL 1

AIM:

To install Hadoop framework, configure it and setup a single node cluster. Use web based tools to monitor your Hadoop setup.

THEORY:

Hadoop:

- The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.
- The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
- It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

STEPS OF INSTALLATION:

The Hadoop framework is written in Java, and its services require a compatible Java Runtime Environment (JRE) and Java Development Kit (JDK). Use the following command to update your system before initiating a new installation:

sudo apt update

At the moment, Apache Hadoop 3.x fully supports Java 8. The OpenJDK 8 package in Ubuntu contains both the runtime environment and development kit.

Type the following command in your terminal to install OpenJDK 8:

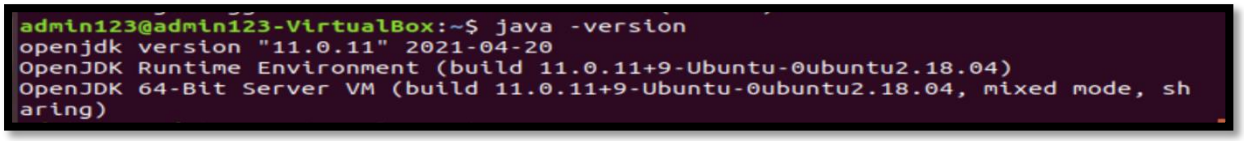
sudo apt-get install default-jdk

The OpenJDK or Oracle Java version can affect how elements of a Hadoop ecosystem interact. To install a specific Java version, check out our detailed guide on how to install Java on Ubuntu.

Once the installation process is complete, verify the current Java version:

java -version

The output informs you which Java edition is in use.



Install the OpenSSH server and client using the following command:

sudo apt-get install openssh-server

Generate an SSH key pair and define the location is is to be stored in:

ssh-keygen -t rsa -P ""

Use the cat command to store the public key as authorized_keys in the ssh directory:

cat $HOME/ .ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

The new user is now able to SSH without needing to enter a password every time. Verify everything is set up correctly by using the hdoop user to SSH to localhost:

ssh localhost

```
admin123@admin123-VirtualBox:~$ ssh-keygen -t RSA -P ""
Generating public/private RSA key pair.
Enter file in which to save the key (/home/admin123/.ssh/id_rsa):
Your identification has been saved in /home/admin123/.ssh/id_rsa.
Your public key has been saved in /home/admin123/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:839YmC3X2PIvB9MZpNO7hlUw9m/njSHKIUEgJvLTSa8 admin123@admin123-VirtualBox
The key's randomart image is:
+---[RSA 2048]----+
|. . + ..         |
| o = +  .     +. |
|  o o ..     .++ |
|   . .  .    o oo|
|    E   S.   +.=*|
|        .o. = X+B|
|         o.o *oX+|
|          o...+o=|
|            ...oo|
+----[SHA256]-----+
admin123@admin123-VirtualBox:~$ cat $HOME/ .ssh/id_rsa.pub >> $HOME/.ssh/author
ized_keys
cat: /home/admin123/: Is a directory
```

Visit the official Apache Hadoop project page, and select the version of Hadoop you want to implement.

Select your preferred option, and you are presented with a mirror link that allows you to download the Hadoop tar package.

Once the download is complete, extract the files to initiate the Hadoop installation:

tar xzf hadoop-3.3.1.tar.gz

Hadoop excels when deployed in a fully distributed mode on a large cluster of networked servers. However, if you are new to Hadoop and want to explore basic commands or test applications, you can configure Hadoop on a single node.

This setup, also called pseudo-distributed mode, allows each Hadoop daemon to run as a single Java process. A Hadoop environment is configured by editing a set of configuration files:

- bashrc
- hadoop-env.sh
- core-site.xml
- hdfs-site.xml
- mapred-site-xml
- yarn-site.xml

Edit the .bashrc shell configuration file using a text editor of your choice (we will be using nano):

sudo nano .bashrc

Define the Hadoop environment variables by adding the following content to the end of the file:

#Hadoop Related Options

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export HADOOP_HOME=/usr/local/hadoop

export HADOOP_INSTALL=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_HDFS_HOME=$HADOOP_HOME

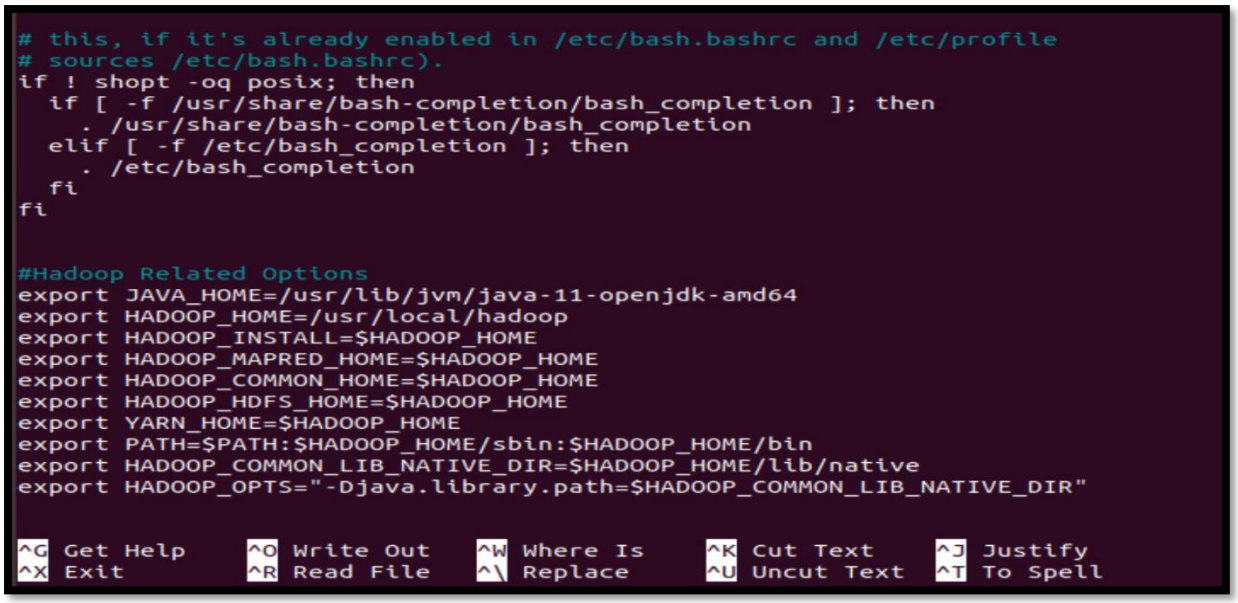
export YARN_HOME=$HADOOP_HOME

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export HADOOP_OPTS="-Djava.library.path=$HADOOP_COMMON_LIB_NATIVE_DIR"

Once you add the variables, save and exit the .bashrc file.



The hadoop-env.sh file serves as a master file to configure YARN, HDFS, MapReduce, and Hadoop-related project settings.

Uncomment the $JAVA_HOME variable (i.e., remove the # sign) and add the full path to the OpenJDK installation on your system. If you have installed the same version as presented in the first part of this tutorial, add the following line:

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

The path needs to match the location of the Java installation on your system.

The core-site.xml file defines HDFS and Hadoop core properties.

To set up Hadoop in a pseudo-distributed mode, you need to specify the URL for your NameNode, and the temporary directory Hadoop uses for the map and reduce process.

Open the core-site.xml file in a text editor:

sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml

Add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting:

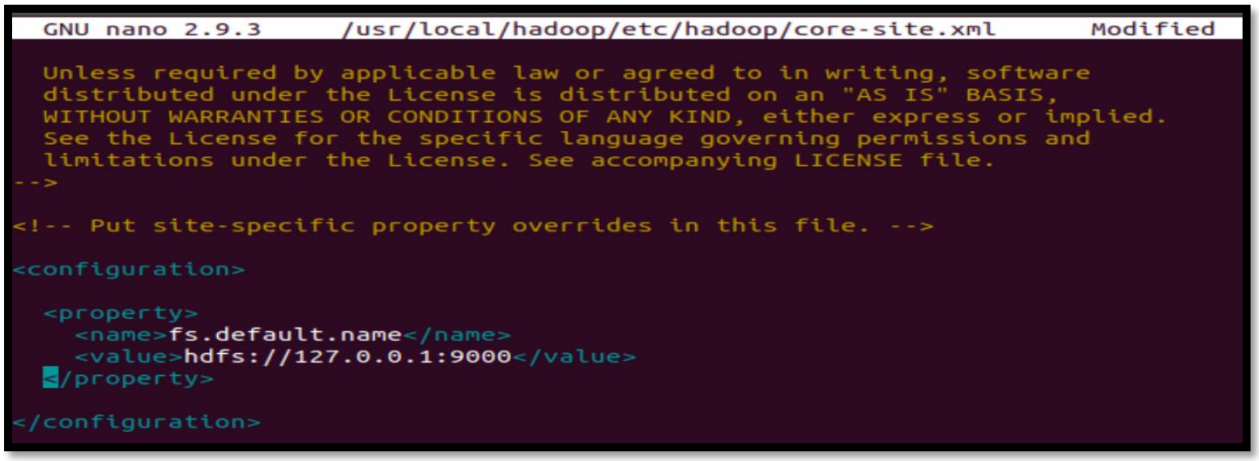
<property>

<name>fs.default.name</name>

<value>hdfs://127.0.0.1:9000</value>

</property>

This example uses values specific to the local system. You should use values that match your systems requirements. The data needs to be consistent throughout the configuration process.



The properties in the hdfs-site.xml file govern the location for storing node metadata, fsimage file, and edit log file. Configure the file by defining the NameNode and DataNode storage directories.

Additionally, the default dfs.replication value of 3 needs to be changed to 1 to match the single node setup.

Use the following command to open the hdfs-site.xml file for editing:

sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml

Add the following configuration to the file and, if needed, adjust the NameNode and DataNode directories to your custom locations:

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>

</property>

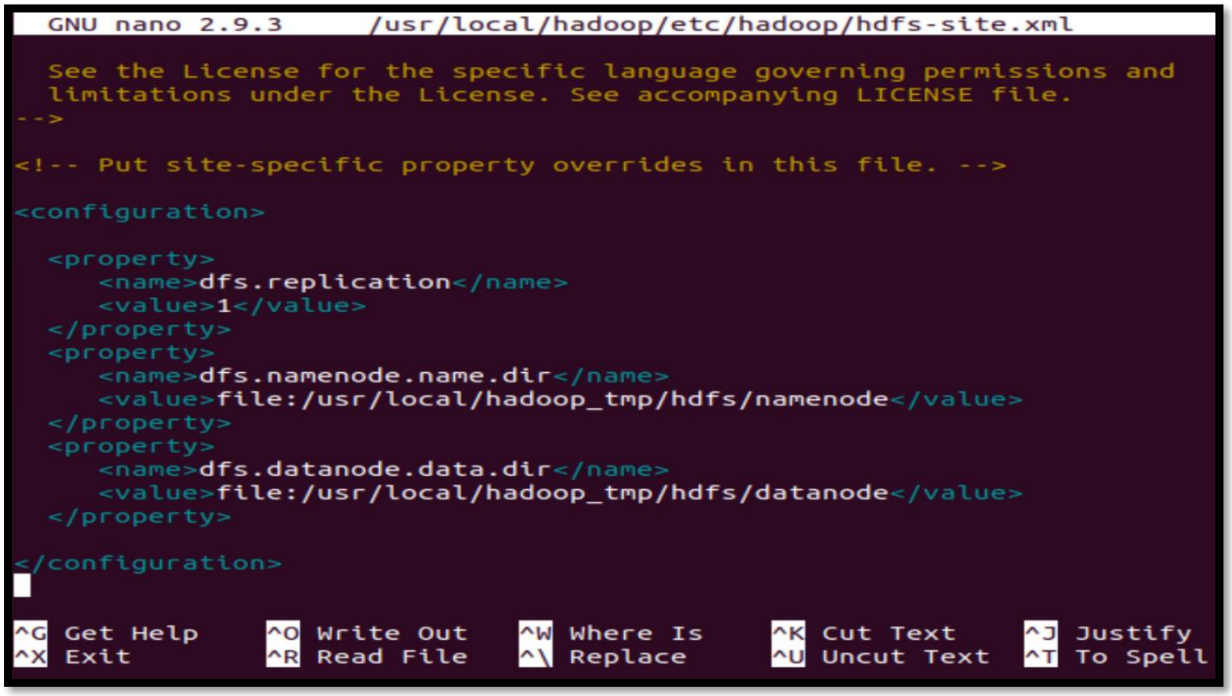
<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>

</property>

If necessary, create the specific directories you defined for the dfs.data.dir value.



Use the following command to access the mapred-site.xml file and define MapReduce values:

sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml

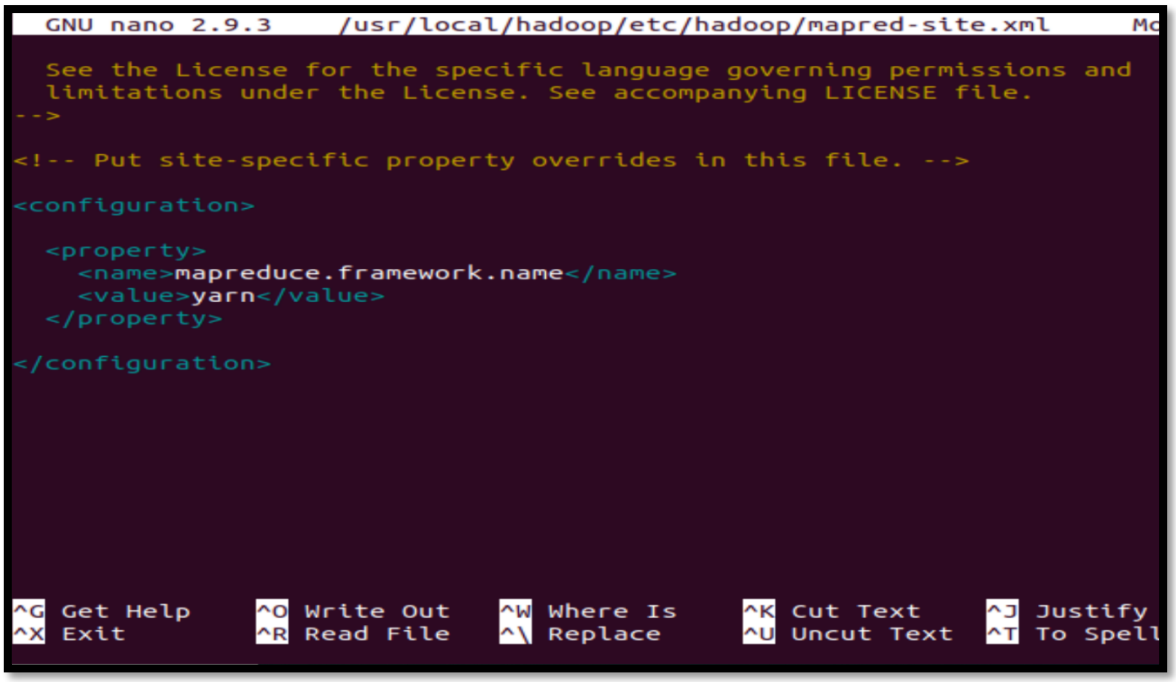
Add the following configuration to change the default MapReduce framework name value to yarn:

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>



The yarn-site.xml file is used to define settings relevant to YARN. It contains configurations for the Node Manager, Resource Manager, Containers, and Application Master.

Open the yarn-site.xml file in a text editor:

sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

Append the following configuration to the file:

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

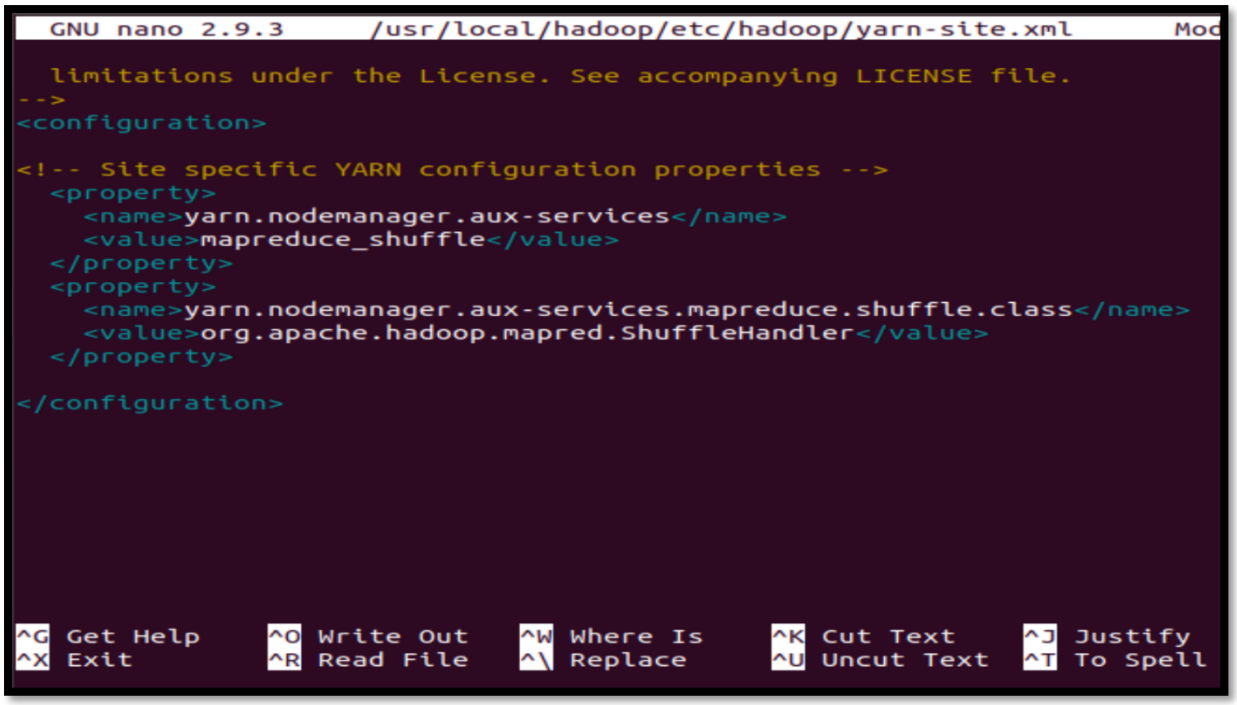
</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>



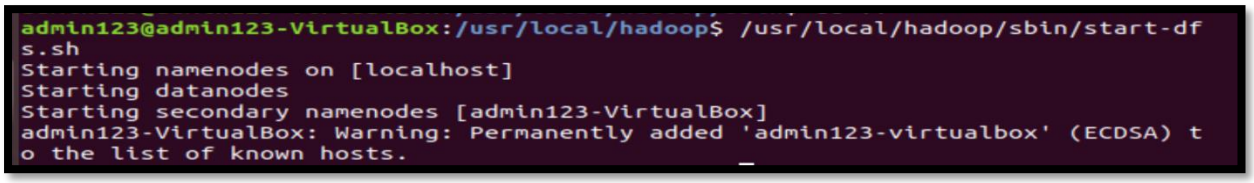
It is important to format the NameNode before starting Hadoop services for the first time:

hdfs namenode -format

The shutdown notification signifies the end of the NameNode format process.

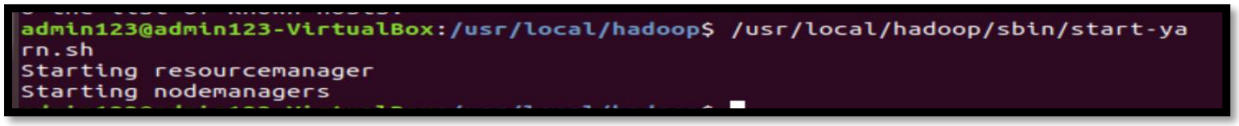
Navigate to the hadoop-3.2.1/sbin directory and execute the following commands to start the NameNode and DataNode:

./start-dfs.sh



Once the namenode, datanodes, and secondary namenode are up and running, start the YARN resource and nodemanagers by typing:

./start-yarn.sh



Type this simple command to check if all the daemons are active and running as Java processes:

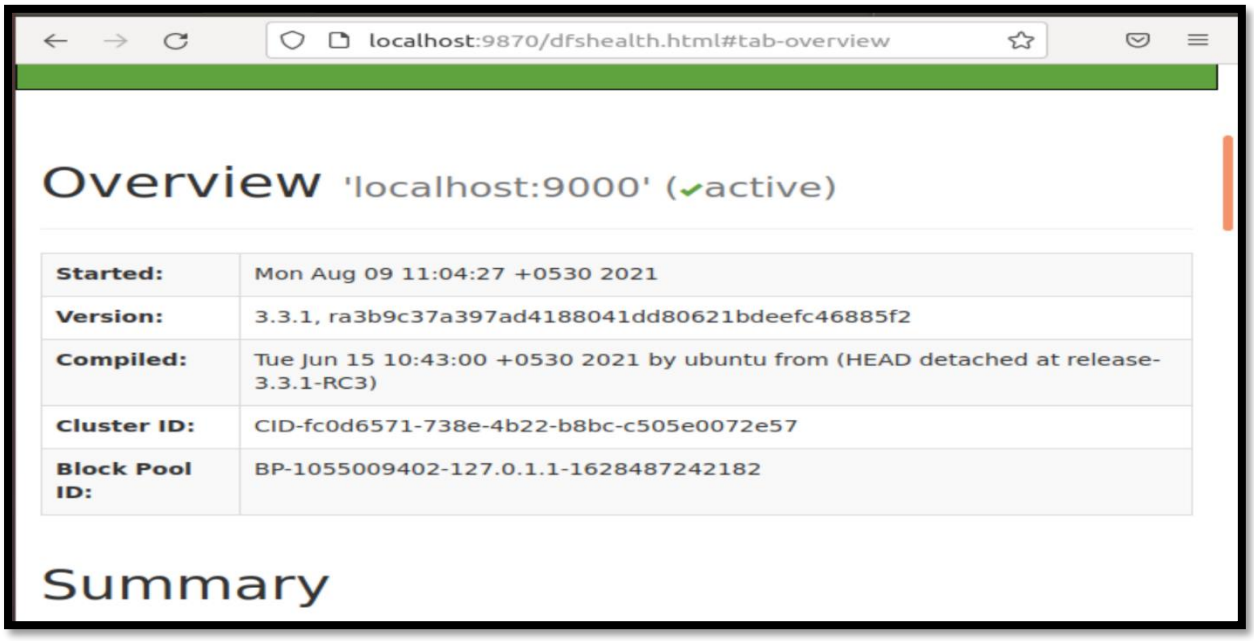
Jps

```
admin123@admin123-VirtualBox:/usr/local/hadoop$ jps
9027 ResourceManager
8772 SecondaryNameNode
9509 Jps
8566 DataNode
9166 NodeManager
```

Use your preferred browser and navigate to your localhost URL or IP. The default port number 9870 gives you access to the Hadoop NameNode UI:

http://localhost:9870

The NameNode user interface provides a comprehensive overview of the entire cluster.

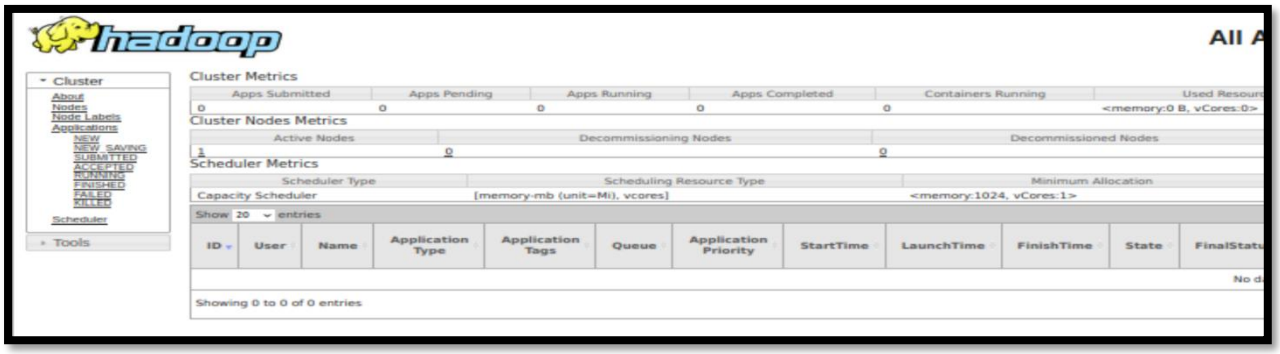


The default port 9864 is used to access individual DataNodes directly from your browser:

http://localhost:9864

The YARN Resource Manager is accessible on port 8088:

http://localhost:8088



CONCLUSION:

In this practical, we learnt about Hadoop and we installed 3.3.1 version and configured it.

PRACTICAL 2

AIM:

To implement file management tasks in Hadoop HDFS and perform Hadoop commands.

PRACTICAL:

There are many more commands in "$HADOOP_HOME/bin/hadoop fs" than are demonstrated here, although these basic operations will get you started. Running ./bin/hadoop dfs with no additional arguments will list all the commands that can be run with the FsShell system. Furthermore, $HADOOP_HOME/bin/hadoop fs -help commandName will display a short usage summary for the operation in question, if you are stuck.

-mkdir <path>

Creates a directory named path in HDFS.

Creates any parent directories in path that are missing (e.g., mkdir -p in Linux).

-ls <path>

Lists the contents of the directory specified by path, showing the names, permissions, owner, size and modification date for each entry.

```
admin123@admin123-VirtualBox:/usr/local/hadoop$ ./bin/hdfs dfs -mkdir /DSA
admin123@admin123-VirtualBox:/usr/local/hadoop$ ./bin/hadoop dfs -ls /
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

Found 1 items
drwxr-xr-x   - admin123 supergroup          0 2021-08-09 11:15 /DSA
admin123@admin123-VirtualBox:/usr/local/hadoop$
```

-touchz <path>

Creates a file at path containing the current time as a timestamp. Fails if a file already exists at path, unless the file is already size 0.

```
admin123@admin123-VirtualBox:/usr/local/hadoop$ ./bin/hadoop dfs -touchz /DSA/e
xample
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.
```

-appendToFile <path>

Append single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and appends to destination file system.

Press Ctrl + D to write on the file.

```
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -appendToFile - /D
SA/example.txt
This is example file.
```

-cat <filen-ame>

Displays the contents of filename on stdout.

```
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -cat /DSA/example.
txt
This is example file.
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$
```

-mv <src><dest>

Moves the file or directory indicated by src to dest, within HDFS.

```
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -mkdir /DSA_move
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -mv /DSA/example.t
xt /DSA_move/
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -ls /DSA2
ls: `/DSA2': No such file or directory
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -ls /DSA_move
Found 1 items
-rw-r--r--   1 admin123 supergroup         22 2021-08-09 11:35 /DSA_move/exampl
e.txt
```

-cp <src> <dest>

Copies the file or directory identified by src to dest, within HDFS.

```
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -cp /DSA_move/exam
ple.txt /DSA/
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hdfs dfs -ls /DSA
Found 1 items
-rw-r--r--   1 admin123 supergroup         22 2021-08-09 11:38 /DSA/example.txt
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$
```

CONCLUSION:

In this practical, we learnt about basic commands in Hadoop file system.

PRACTICAL 3

AIM:

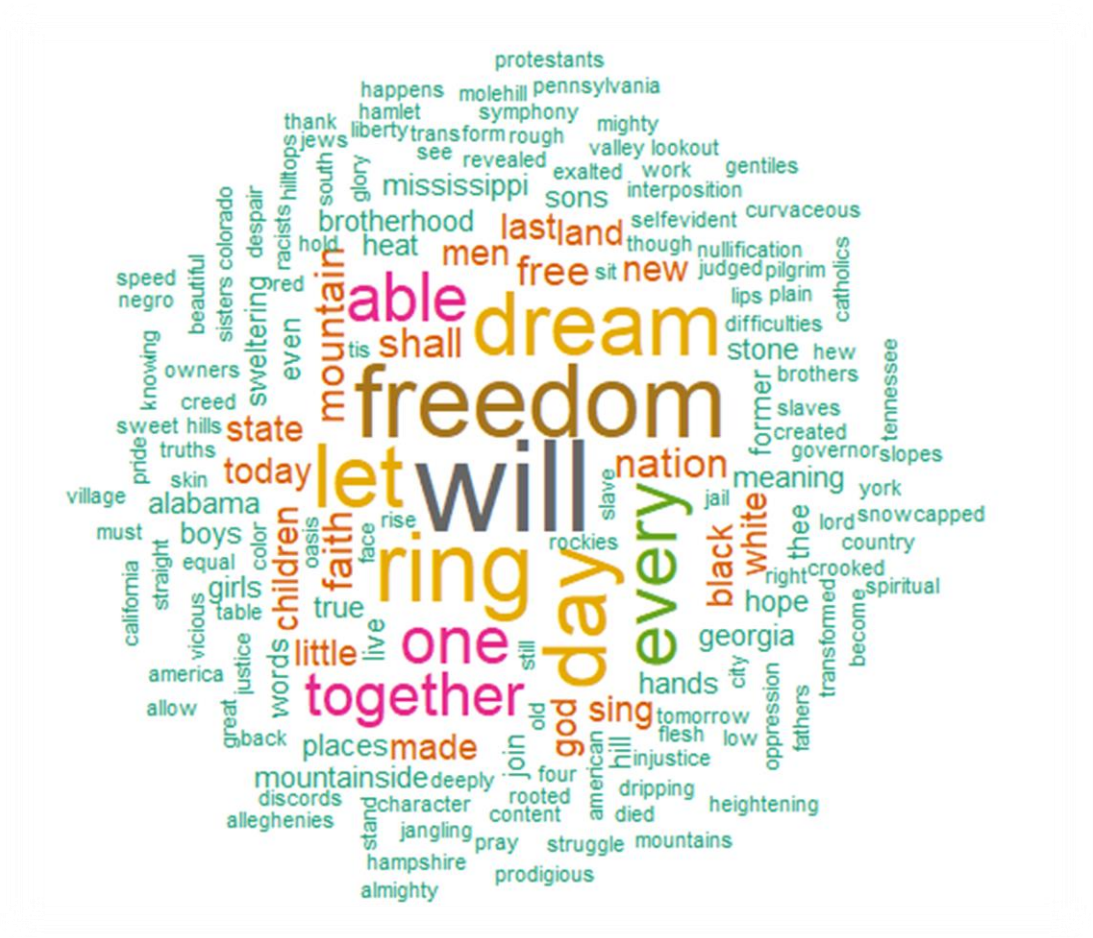
To implement basic functions and commands in R Programming. To build WordCloud, a text mining method using R for easy to understand and better visualization than a data table.

PRACTICAL:

```
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
library("Rcpp")

filePath <-
  "http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt"
text <- readLines(filePath)
docs <- Corpus(VectorSource(text))
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, stripWhitespace)
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
```

```
d <- data.frame(word = names(v),freq=v)
head(d, 10)
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
        max.words=200, random.order=FALSE, rot.per=0.35,
        colors=brewer.pal(8, "Dark2"))
```

OUTPUT:



CONCLUSION:

In this practical, we learnt about R and implemented wordCloud using R.

PRACTICAL 4

AIM:

- To implement a word count application using the MapReduce programming model.
- To implement program that count the occurrences of word based on the length.

THEORY:

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. As the processing component, MapReduce is the heart of Apache Hadoop. The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

MapReduce programming offers several benefits to help you gain valuable insights from your big data:

Scalability. Businesses can process petabytes of data stored in the Hadoop Distributed File System (HDFS).

Flexibility. Hadoop enables easier access to multiple sources of data and multiple types of data.

Speed. With parallel processing and minimal data movement, Hadoop offers fast processing of massive amounts of data.

Simple. Developers can write code in a choice of languages, including Java, C++ and Python.

PRACTICAL:

```
import java.io.IOException;

import java.util.StringTokenizer;


import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

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

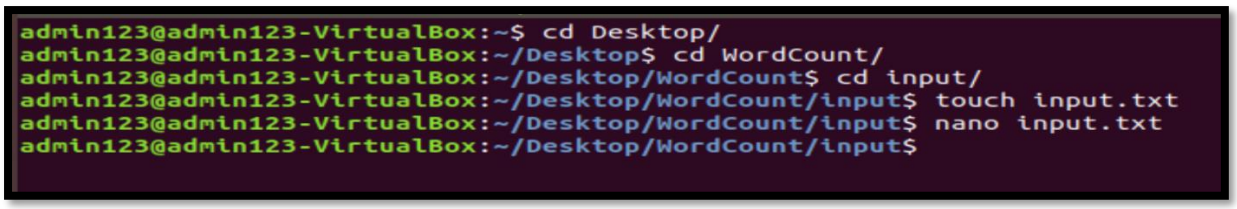
```
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

 public static class TokenizerMapper
      extends Mapper<Object, Text, Text, IntWritable>{

   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();

   public void map(Object key, Text value, Context context
                   ) throws IOException, InterruptedException {
     StringTokenizer itr = new StringTokenizer(value.toString());
     while (itr.hasMoreTokens()) {
       word.set(itr.nextToken());
       context.write(word, one);
     }
   }
 }

 public static class IntSumReducer
      extends Reducer<Text,IntWritable,Text,IntWritable> {
   private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<IntWritable> values,
```

```
                Context context
                ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
 }

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

STEPS TO RUN:

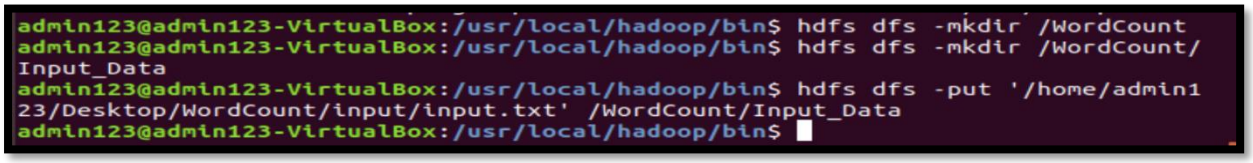
Create a directory in local system and create 2 directories inside it called Classes and Input. We write this code in WordCount.java file and put it in the same directory.

We will also create input.txt file in Input directory and write random words which we want to count.



We will now create an HDFS directory called WordCount and inside that another directory called Input_Data.

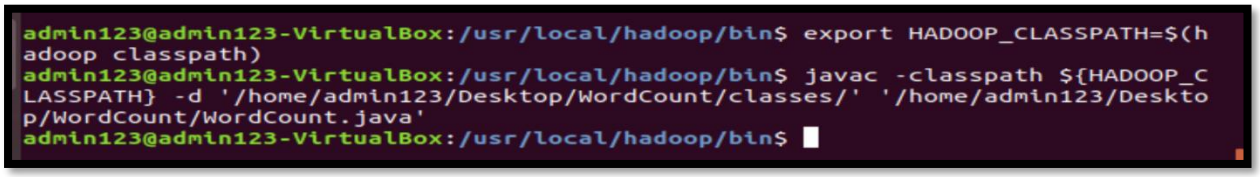
We will put the input.txt from local file system to HDFS.



We will now use following commands.

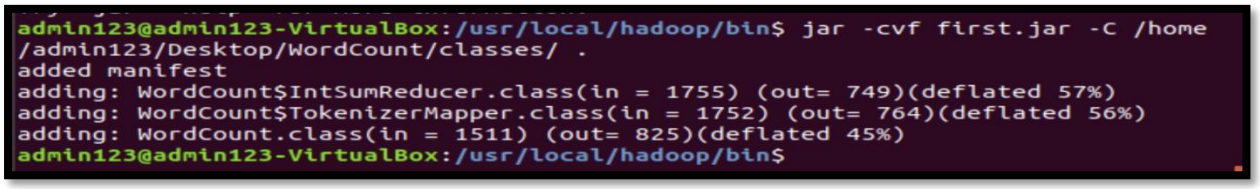
Export HADOOP_CLASSPATH=$(Hadoop classpath)

Javac -classpath ${HADOOP_CLASSPATH} -d <path-to-local-classes-folder> <path-to-wordcount-java-file>



Now we will use jar command.

Jar -cvf <any-file-name.jar> -C <path-to-class-folder-in-local> .



Now we will run it on Hadoop by following command.

Hadoop jar <path-to-jar-file> <classname> <path-to-hdfs-input_data> <path-to-hdfs-output>

```
admin123@admin123-VirtualBox:/usr/local/hadoop/bin$ hadoop jar '/home/admin123/
Desktop/WordCount/first.jar' WordCount /WordCount/Input_Data /WordCount/Output
2021-08-09 13:04:11,492 INFO client.DefaultNoHARMFailoverProxyProvider: Connect
ing to ResourceManager at /0.0.0.0:8032
2021-08-09 13:04:12,602 WARN mapreduce.JobResourceUploader: Hadoop command-line
 option parsing not performed. Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
2021-08-09 13:04:12,627 INFO mapreduce.JobResourceUploader: Disabling Erasure C
oding for path: /tmp/hadoop-yarn/staging/admin123/.staging/job_1628481778259_00
01
2021-08-09 13:04:13,145 INFO input.FileInputFormat: Total input files to proces
s : 1
2021-08-09 13:04:13,303 INFO mapreduce.JobSubmitter: number of splits:1
2021-08-09 13:04:13,885 INFO mapreduce.JobSubmitter: Submitting tokens for job:
 job_1628481778259_0001
2021-08-09 13:04:13,886 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-08-09 13:04:14,302 INFO conf.Configuration: resource-types.xml not found
2021-08-09 13:04:14,302 INFO resource.ResourceUtils: Unable to find 'resource-t
ypes.xml'.
2021-08-09 13:04:14,997 INFO impl.YarnClientImpl: Submitted application applica
tion_1628481778259_0001
2021-08-09 13:04:15,143 INFO mapreduce.Job: The url to track the job: http://ad
min123-VirtualBox:8088/proxy/application_1628481778259_0001/
2021-08-09 13:04:15,146 INFO mapreduce.Job: Running job: job_1628481778259_0001
```

We can check the output by cat command at output path specified in previous command.

OUTPUT:

```
admin123@admin123-VirtualBox:/usr/local/hadoop$ ./bin/hdfs dfs -cat /WordCount/
Output_Data/part-r-00000
black   4
blue    4
pink    4
red     2
white   4
admin123@admin123-VirtualBox:/usr/local/hadoop$
```

CONCLUSION:

In this practical, we learnt about mapreduce and implemented a word count program using Java.

PRACTICAL 5

AIM:

A. To design and implement MapReduce algorithms to take a very large file of integers and produce as output:
   a. The largest integer
   b. The average of all the integers.
   c. The same set of integers, but with each integer appearing only once.
   d. The count of the number of distinct integers in the input.
B. To design an application to find mutual friend using map reduce.

STEPS OF IMPLEMENTATION:

We will use the same steps for implementing our logic in all the above cases. The names of files will change accordingly.

Create a directory in local system and create 2 directories inside it called Classes and Input. We write this code in IntMax.java file and put it in the same directory.

We will also create input.txt file in Input directory and write random integers to process.

We will now create an HDFS directory called prac5_a and inside that another directory called Input_Data.

We will put the input.txt from local file system to HDFS.

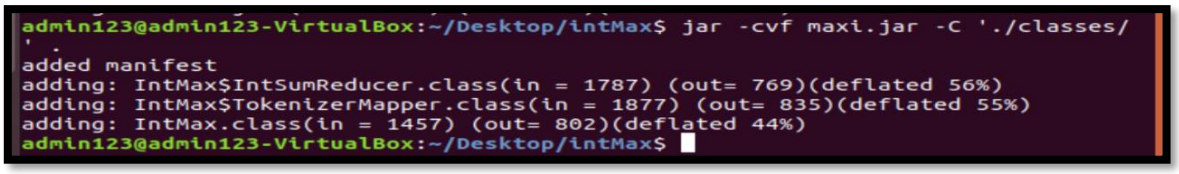
We will now use following commands.

Export HADOOP_CLASSPATH=$(Hadoop classpath)

Javac -classpath ${HADOOP_CLASSPATH} -d <path-to-local-classes-folder> <path-to-wordcount-java-file>

Then we will use jar command.

Jar -cvf <any-file-name.jar> -C <path-to-class-folder-in-local> .



Now we will run it on Hadoop by following command.

Hadoop jar <path-to-jar-file> <classname> <path-to-hdfs-input_data> <path-to-hdfs-output>

```
File  Edit  View  Search  Terminal  Help
admin123@admin123-VirtualBox:/usr/local/hadoop/sbin$ hadoop jar '/home/admin123
/Desktop/intMax/maxi.jar' IntMax '/prac5_a/Input_Data' '/prac5_a/Output3'
2021-08-23 14:42:10,563 INFO client.DefaultNoHARMFailoverProxyProvider: Connect
ing to ResourceManager at /0.0.0.0:8032
2021-08-23 14:42:11,373 WARN mapreduce.JobResourceUploader: Hadoop command-line
 option parsing not performed. Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
2021-08-23 14:42:11,419 INFO mapreduce.JobResourceUploader: Disabling Erasure C
oding for path: /tmp/hadoop-yarn/staging/admin123/.staging/job_1629709300511_00
04
2021-08-23 14:42:11,924 INFO input.FileInputFormat: Total input files to proces
s : 1
2021-08-23 14:42:12,091 INFO mapreduce.JobSubmitter: number of splits:1
2021-08-23 14:42:12,718 INFO mapreduce.JobSubmitter: Submitting tokens for job:
 job_1629709300511_0004
2021-08-23 14:42:12,718 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-08-23 14:42:13,104 INFO conf.Configuration: resource-types.xml not found
2021-08-23 14:42:13,109 INFO resource.ResourceUtils: Unable to find 'resource-t
ypes.xml'.
2021-08-23 14:42:13,247 INFO impl.YarnClientImpl: Submitted application applica
tion_1629709300511_0004
2021-08-23 14:42:13,328 INFO mapreduce.Job: The url to track the job: http://ad
min123-VirtualBox:8088/proxy/application_1629709300511_0004/
2021-08-23 14:42:13,330 INFO mapreduce.Job: Running job: job_1629709300511_0004
2021-08-23 14:42:25,747 INFO mapreduce.Job: Job job_1629709300511_0004 running
in uber mode : false
2021-08-23 14:42:25,751 INFO mapreduce.Job:  map 0% reduce 0%
2021-08-23 14:42:33,896 INFO mapreduce.Job:  map 100% reduce 0%
2021-08-23 14:42:46.139 INFO mapreduce.Job:  map 100% reduce 100%
```

We can check the output by cat command at output path specified in previous command.

```
admin123@admin123-VirtualBox:/usr/local/hadoop/sbin$ hdfs dfs -cat /prac5_a/Out
put3/part-r-00000
max     9
admin123@admin123-VirtualBox:/usr/local/hadoop/sbin$
```

CODE A:

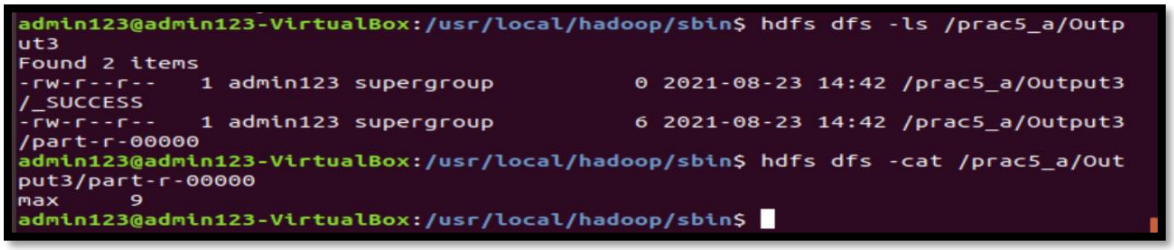
IntMax.java

```
import java.io.IOException;

import java.util.StringTokenizer;


import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class IntMax {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text("max");
    private int max = Integer.MIN_VALUE;

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        //word.set(itr.nextToken());
        //context.write(word, one);
           int temp = Integer.parseInt(itr.nextToken());
           if(temp > max)
             max = temp;
      }

      context.write(word, new IntWritable(max));
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
```

```
  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int max = Integer.MIN_VALUE;
    for (IntWritable val : values) {
      //sum += val.get();
         if(val.get() > max)
           max = val.get();
    }
    result.set(max);
    context.write(key, result);
  }
 }

 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "Max int");
  job.setJarByClass(IntMax.class);
  job.setMapperClass(TokenizerMapper.class);
  //job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

OUTPUT A:



CODE B:

IntAvg.java

```
import java.io.IOException;
import java.util.StringTokenizer;
import java.io.DataInput;
import java.io.DataOutput;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Writable;
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;

class Custom implements Writable {
    private int sum;
    private int count;
```

```
public Custom()
{
        sum = 0;
        count = 0;
}

public void write(DataOutput dataOutput) throws IOException
{
        dataOutput.writeInt(sum);
        dataOutput.writeInt(count);
}

public void readFields(DataInput dataInput) throws IOException
{
        sum = dataInput.readInt();
        count = dataInput.readInt();
}

public void setSum(int value)
{
        sum = value;
}

public int getSum()
{
        return sum;
}
```

```
    public void setCount(int value)
    {
            count = value;
    }

    public int getCount()
    {
            return count;
    }

    public String toString()
    {
            return "("+sum+","+count+")";
    }
}

public class IntAvg {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, Custom>{

    //private final static IntWritable one = new IntWritable(1);
    private Text word = new Text("avg");
    private Custom obj = new Custom();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      int sum = 0;
      int count = 0;
```

```
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      //word.set(itr.nextToken());
      //context.write(word, one);
          int temp = Integer.parseInt(itr.nextToken());
          sum += temp;
          count++;
          //context.write(new Text(""+temp), new Text("("+sum+","+count+")"));
    }
    obj.setSum(sum);
    obj.setCount(count);
    context.write(word, obj);
    //context.write(new Text("Avg"), new Text("("+sum+","+count+")"));
  }
 }

 public static class IntSumReducer
      extends Reducer<Text,Custom,Text,IntWritable> {
   private IntWritable result = new IntWritable();

   public void reduce(Text key, Iterable<Custom> values,
                      Context context
                      ) throws IOException, InterruptedException {
    int sum = 0;
    int count = 0;
    for (Custom val : values) {
      sum += val.getSum();
          count += val.getCount();
```

```
    }
    result.set((int)(sum/count));
    context.write(key, result);
  }
 }


 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "Average");
  job.setJarByClass(IntAvg.class);
  job.setMapperClass(TokenizerMapper.class);
  //job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);

  job.setMapOutputValueClass(Custom.class);

  job.setOutputKeyClass(Text.class);
  //job.setOutputValueClass(IntWritable.class);
  job.setOutputValueClass(Text.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));
  System.exit(job.waitForCompletion(true) ? 0 : 1);
 }
}
```

INPUT B:

OUTPUT B:

```
admin123@admin123-VirtualBox:/usr/local/hadoop/sbin$ hdfs dfs -cat /prac5_b/Out
put19/part-r-00000
avg     10
admin123@admin123-VirtualBox:/usr/local/hadoop/sbin$
```

CODE C:

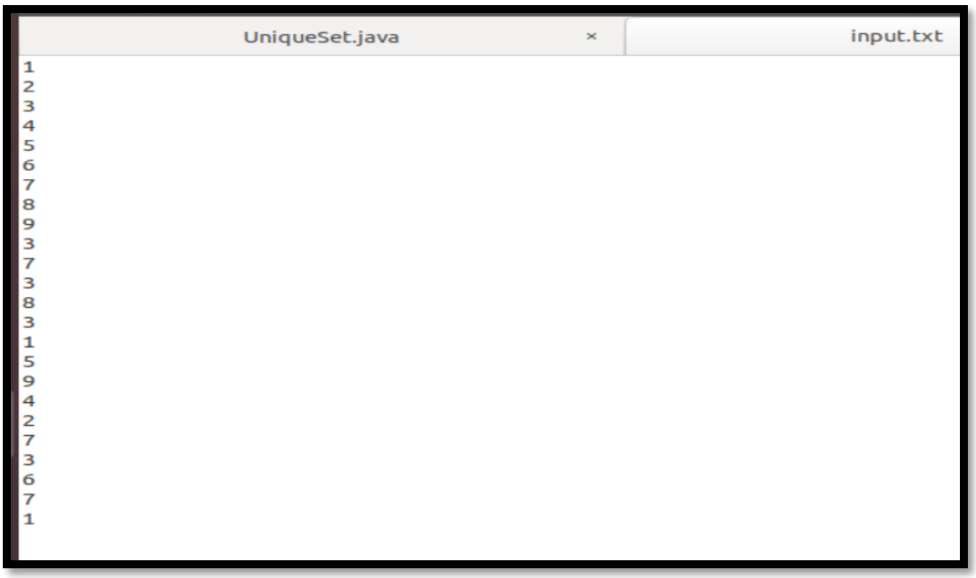
UniqueSet.java

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

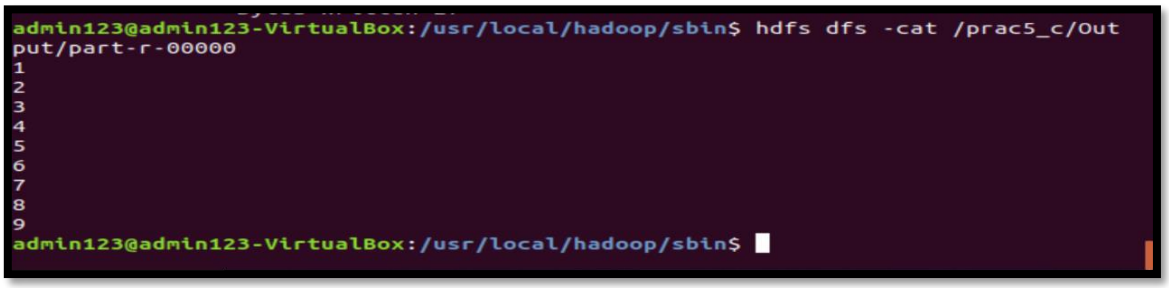
```
public class UniqueSet {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class UniqueKeys
       extends Reducer<Text,IntWritable,Text,Text> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      //int sum = 0;
      //for (IntWritable val : values) {
      //  sum += val.get();
      //}
```

```
      //result.set(sum);
      context.write(key, new Text(""));
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Unique Set");
    job.setJarByClass(UniqueSet.class);
    job.setMapperClass(TokenizerMapper.class);
    //job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(UniqueKeys.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

INPUT C:



OUTPUT C:



CODE D:

UniqueCount.java

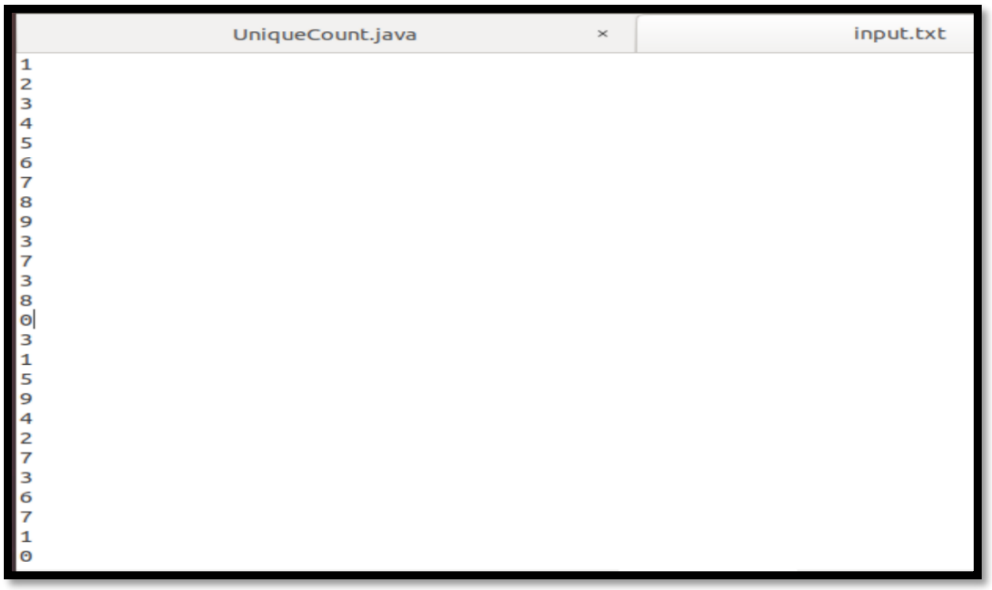
```
import java.io.IOException;
import java.util.StringTokenizer;
import java.util.LinkedList;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
```

```
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class UniqueCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    //private final static IntWritable one = new IntWritable(1);
    //private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        context.write(new Text("Count"), new IntWritable(Integer.parseInt(itr.nextToken())));
      }
    }
  }

  public static class UniqueKeys
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    //private LinkedList intSet = new LinkedList<IntWritable>();
    //private int count = 0;

    public void reduce(Text key, Iterable<IntWritable> values,
```

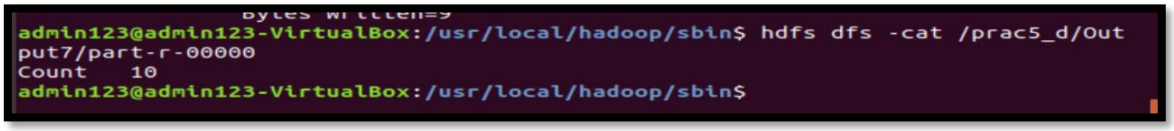
```
                Context context
                ) throws IOException, InterruptedException {
    //int sum = 0;
    int count = 0;
    LinkedList intSet = new LinkedList<Integer>();
    for (IntWritable val : values) {
    //  sum += val.get();
       //int temp = val.get();
          if(!intSet.contains(val.get()))
          {
            count++;
            intSet.add(val.get());
            //context.write(new Text("Added"), val);
          }
          //context.write(key, val);
    }
    //result.set(sum);
    context.write(key, new IntWritable(count));
  }
}

 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  Job job = Job.getInstance(conf, "Unique Count");
  job.setJarByClass(UniqueCount.class);
  job.setMapperClass(TokenizerMapper.class);
  //job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(UniqueKeys.class);
  job.setOutputKeyClass(Text.class);
```

```
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

INPUT D:



OUTPUT D:



CONCLUSION:

In this practical, we performed different tasks on the set of integers using MapReduce on Hadoop.

PRACTICAL 6

AIM:

To implement basic CRUD operations (create, read, update, delete) in MongoDB and Cassandra.

THEORY:

MongoDB:

- MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).
- Main Features:
    - Ad-hoc queries
        - MongoDB supports field, range query, and regular-expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.
    - Indexing
        - Fields in a MongoDB document can be indexed with primary and secondary indices or index.
    - Replication
        - MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica-set member may act in the role of primary or secondary replica at any time. All writes and reads are done on the primary replica by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally serve read operations, but that data is only eventually consistent by default.
        - If the replicated MongoDB deployment only has a single secondary member, a separate daemon called an arbiter must be added to the set. It has a single responsibility, which is to resolve the election of the new primary.[30] As a consequence, an idealized distributed MongoDB deployment requires at least three separate servers, even in the case of just one primary and one secondary.
    - Load balancing
        - MongoDB scales horizontally using sharding. The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed

across multiple shards. (A shard is a master with one or more replicas.). Alternatively, the shard key can be hashed to map to a shard – enabling an even data distribution.
        - MongoDB can run over multiple servers, balancing the load or duplicating data to keep the system up and running in case of hardware failure.
    - File storage
        - MongoDB can be used as a file system, called GridFS, with load balancing and data replication features over multiple machines for storing files.
        - This function, called grid file system, is included with MongoDB drivers. MongoDB exposes functions for file manipulation and content to developers. GridFS can be accessed using mongofiles utility or plugins for Nginx and lighttpd. GridFS divides a file into parts, or chunks, and stores each of those chunks as a separate document.
    - Aggregation
        - MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single-purpose aggregation methods.
        - Map-reduce can be used for batch processing of data and aggregation operations. But according to MongoDB's documentation, the Aggregation Pipeline provides better performance for most aggregation operations.
        - The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used. Aggregation operators can be strung together to form a pipeline – analogous to Unix pipes. The aggregation framework includes the $lookup operator which can join documents from multiple collections, as well as statistical operators such as standard deviation.
    - Server-side JavaScript execution
        - JavaScript can be used in queries, aggregation functions (such as MapReduce), and sent directly to the database to be executed.
    - Capped collections
        - MongoDB supports fixed-size collections called capped collections. This type of collection maintains insertion order and, once the specified size has been reached, behaves like a circular queue.
    - Transactions
        - MongoDB claims to support multi-document ACID transactions since the 4.0 release in June 2018. This claim was found to not be true as MongoDB violates snapshot isolation.

Cassandra:

- Apache Cassandra is a free and open-source, distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients. Cassandra was designed to implement a combination of Amazon's Dynamo distributed storage and replication techniques combined with Google's Bigtable data and storage engine model.
- Main features
    - Distributed
        - Every node in the cluster has the same role. There is no single point of failure. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.
    - Supports replication and multi data center replication
        - Replication strategies are configurable. Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of Cassandra’s distributed architecture are specifically tailored for multiple-data center deployment, for redundancy, for failover and disaster recovery.
    - Scalability
        - Designed to have read and write throughput both increase linearly as new machines are added, with the aim of no downtime or interruption to applications.
    - Fault-tolerant
        - Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.
    - Tunable consistency
        - Cassandra is typically classified as an AP system, meaning that availability and partition tolerance are generally considered to be more important than consistency in Cassandra, Writes and reads offer a tunable level of consistency, all the way from "writes never fail" to "block for all replicas to be readable", with the quorum level in the middle.
    - MapReduce support
        - Cassandra has Hadoop integration, with MapReduce support. There is support also for Apache Pig and Apache Hive.
    - Query language
        - Cassandra introduced the Cassandra Query Language (CQL). CQL is a simple interface for accessing Cassandra, as an alternative to the traditional Structured Query Language (SQL).
    - Eventual consistency

- Cassandra manages eventual consistency of reads, upserts and deletes through Tombstones.

PRACTICAL:

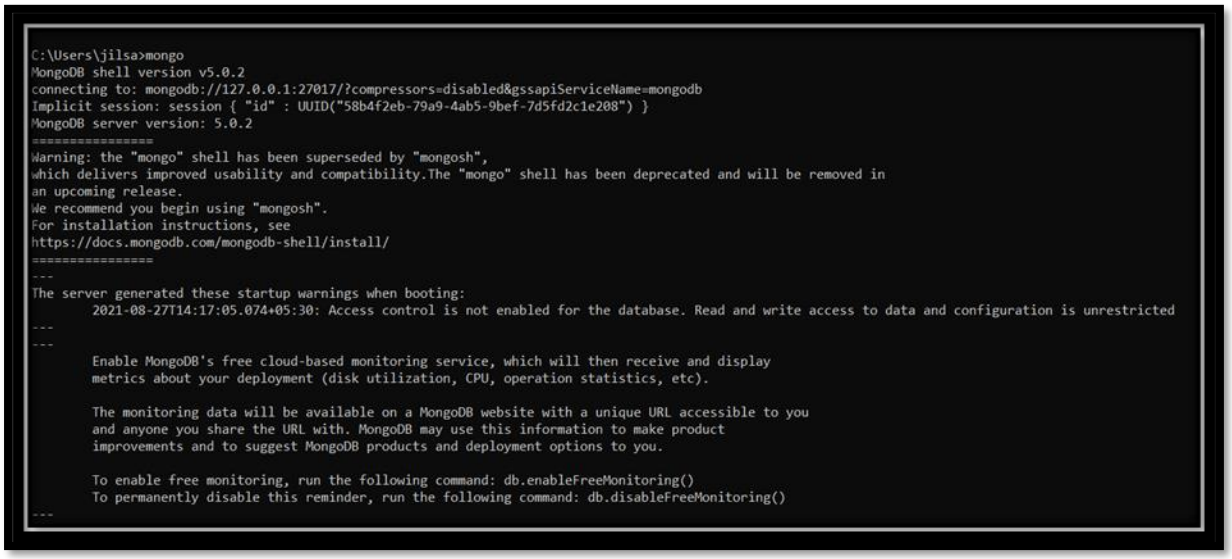
We can download MongoDB and Cassandra with the help of following tutorials.

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

https://phoenixnap.com/kb/install-cassandra-on-windows

MongoDB:

We can start the MongoDB service using “mongo” command.



We can check available databases using “show dbs” command.

We can create database using “use” keyword easily.

We can insert the data in database using “insertOne” command.

```
> db.products.insertOne({name:'product1', price:40, color:'pink'});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6131e482ae9c55a33bbd67be")
}
```

```
> db.products.insertOne({name:'product2', price:60, color:'blue'});
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6131e496ae9c55a33bbd67bf")
}
```

We can check all the records of database using “find” and “pretty()” to format the output.

```
}
> db.products.find();
{ "_id" : ObjectId("6131e482ae9c55a33bbd67be"), "name" : "product1", "price" : 40, "color" : "pink" }
{ "_id" : ObjectId("6131e496ae9c55a33bbd67bf"), "name" : "product2", "price" : 60, "color" : "blue" }
> db.products.insertMany([{name:'product3', price:20, color:'green'}, {name:'product4', price: 15, color:'yellow'}]);
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("6131e4e1ae9c55a33bbd67c0"),
                ObjectId("6131e4e1ae9c55a33bbd67c1")
        ]
}
> db.products.find().pretty();
{
        "_id" : ObjectId("6131e482ae9c55a33bbd67be"),
        "name" : "product1",
        "price" : 40,
        "color" : "pink"
}
{
        "_id" : ObjectId("6131e496ae9c55a33bbd67bf"),
        "name" : "product2",
        "price" : 60,
        "color" : "blue"
}
{
        "_id" : ObjectId("6131e4e1ae9c55a33bbd67c0"),
        "name" : "product3",
        "price" : 20,
        "color" : "green"
}
{
        "_id" : ObjectId("6131e4e1ae9c55a33bbd67c1"),
        "name" : "product4",
        "price" : 15,
        "color" : "yellow"
}
```

We can also update our record using “updateOne”.

```
> db.products.updateOne({name:'product2'}, {$set: {price:50, qty:10}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

Cassandra:

We can create and use keyspace in Cassandra using following command.

```
CREATE KEYSPACE keysp WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'}  AND durable_writes = true;
```

We can create table and insert data using sql-type query. We can also check the data with same query.

```
cqlsh:keysp> CREATE TABLE emp(emp_id int PRIMARY KEY, emp_name text, emp_city text, emp_sal varint, emp_phone varint);
```

```
cqlsh:keysp> INSERT INTO emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) VALUES (101, 'Robin', 'Hyderabad', 50000, 123456789);
cqlsh:keysp> INSERT INTO emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) VALUES (102, 'Timmy', 'Delhi', 53000, 256478269);
cqlsh:keysp> INSERT INTO emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) VALUES (103, 'Heli', 'Mumbai', 49000, 793067431);
cqlsh:keysp> SELECT * FROM emp;

 emp_id | emp_city  | emp_name | emp_phone | emp_sal
--------+-----------+----------+-----------+---------
    102 |     Delhi |    Timmy | 256478269 |   53000
    101 | Hyderabad |    Robin | 123456789 |   50000
    103 |    Mumbai |     Heli | 793067431 |   49000

(3 rows)
```

We can also update and delete the record using corresponding queries.

```
cqlsh:keysp> UPDATE emp SET emp_sal=48000 WHERE emp_id=103;
cqlsh:keysp> SELECT * FROM emp;

 emp_id | emp_city  | emp_name | emp_phone | emp_sal
--------+-----------+----------+-----------+---------
    102 |     Delhi |    Timmy | 256478269 |   53000
    101 | Hyderabad |    Robin | 123456789 |   50000
    103 |    Mumbai |     Heli | 793067431 |   48000

(3 rows)
cqlsh:keysp> DELETE FROM emp WHERE emp_id=102;
cqlsh:keysp> SELECT * FROM emp;

 emp_id | emp_city  | emp_name | emp_phone | emp_sal
--------+-----------+----------+-----------+---------
    101 | Hyderabad |    Robin | 123456789 |   50000
    103 |    Mumbai |     Heli | 793067431 |   48000

(2 rows)
```

CONCLUSION:

In this practical, we learnt about mongoDB and Cassandra and performed basic CRUD operations in both the databases.

PRACTICAL 7

AIM:

To develop a MapReduce application and implement a program that analyses weather data.

CODE:

MyMaxMin.java

```
// importing Libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
public class MyMaxMin {
        // Mapper
        /*MaxTemperatureMapper class is static
        * and extends Mapper abstract class
        * having four Hadoop generics type
        * LongWritable, Text, Text, Text.
        */
        public static class MaxTemperatureMapper extends
                        Mapper<LongWritable, Text, Text, Text> {
```

```
        /**
        * @method map
        * This method takes the input as a text data type.
        * Now leaving the first five tokens, it takes
        * 6th token is taken as temp_max and
        * 7th token is taken as temp_min. Now
        * temp_max > 30 and temp_min < 15 are
        * passed to the reducer.
        */
// the data in our data set with
// this value is inconsistent data
public static final int MISSING = 9999;
@Override
        public void map(LongWritable arg0, Text Value, Context context)
                        throws IOException, InterruptedException {

        // Convert the single row(Record) to
        // String and store it in String
        // variable name line

        String line = Value.toString();

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

                        // from character 6 to 14 we have
                        // the date in our dataset
                        String date = line.substring(6, 14);
```

```
                    // similarly we have taken the maximum
                    // temperature from 39 to 45 characters
                    float temp_Max = Float.parseFloat(line.substring(39,
45).trim());

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

                    float temp_Min = Float.parseFloat(line.substring(47,
53).trim());

                    // if maximum temperature is
                    // greater than 30, it is a hot day
                    if (temp_Max > 30.0) {

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

                    // if the minimum temperature is
                    // less than 15, it is a cold day
                    if (temp_Min < 15) {

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

```
        }

    }

// Reducer

    /*MaxTemperatureReducer class is static
    and extends Reducer abstract class
    having four Hadoop generics type
    Text, Text, Text, Text.
    */

    public static class MaxTemperatureReducer extends
            Reducer<Text, Text, Text, Text> {

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

        public void reduce(Text Key, Iterator<Text> Values, Context context)
                throws IOException, InterruptedException {


            // putting all the values in
            // temperature variable of type String
```

```
                String temperature = Values.next().toString();
                context.write(Key, new Text(temperature));
        }

}
/**
* @method main
* This method is used for setting
* all the configuration properties.
* It acts as a driver for map-reduce
* code.
*/

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

        // reads the default configuration of the
        // cluster from the configuration XML files
        Configuration conf = new Configuration();

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

        // Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);

        // Key type coming out of mapper
        job.setMapOutputKeyClass(Text.class);
```

```
// value type coming out of mapper
job.setMapOutputValueClass(Text.class);

// Defining the mapper class name
job.setMapperClass(MaxTemperatureMapper.class);

// Defining the reducer class name
job.setReducerClass(MaxTemperatureReducer.class);

// Defining input Format class which is
// responsible to parse the dataset
// into a key value pair
job.setInputFormatClass(TextInputFormat.class);
// Defining output Format class which is
// responsible to parse the dataset
// into a key value pair
job.setOutputFormatClass(TextOutputFormat.class);
// setting the second argument
// as a path in a path variable
Path OutputPath = new Path(args[1]);
// Configuring the input path
// from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
// Configuring the output path from
// the filesystem into the job
FileOutputFormat.setOutputPath(job, new Path(args[1]));
// deleting the context path automatically
// from hdfs so that we don't have
// to delete it explicitly
```

```
        OutputPath.getFileSystem(conf).delete(OutputPath);
        // exiting the job only if the
        // flag value becomes false
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}}
```

OUTPUT:

```
 1 The Day is Cold Day :20200101   -21.8
 2 The Day is Cold Day :20200102   -23.4
 3 The Day is Cold Day :20200103   -25.4
 4 The Day is Cold Day :20200104   -26.8
 5 The Day is Cold Day :20200105   -28.8
 6 The Day is Cold Day :20200106   -30.0
 7 The Day is Cold Day :20200107   -31.4
 8 The Day is Cold Day :20200108   -33.6
 9 The Day is Cold Day :20200109   -26.6
10 The Day is Cold Day :20200110   -24.3
```

CONCLUSION:

In this practical, we learnt about analysis of data using mapreduce in Hadoop.

PRACTICAL 8

AIM:

To Install and Run Hive. Use Hive to create, alter, and drop databases, tables, views, functions, and indexes. To create HDFS tables and load them in Hive and implement joining of tables in Hive.

THEORY:

Hive:

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- This is a brief tutorial that provides an introduction on how to use Apache Hive HiveQL with Hadoop Distributed File System. This tutorial can be your first step towards becoming a successful Hadoop Developer with Hive.
- Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis.
- Hive gives an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Traditional SQL queries must be implemented in the MapReduce Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like queries (HiveQL) into the underlying Java without the need to implement queries in the low-level Java API. Since most data warehousing applications work with SQL-based querying languages, Hive aids portability of SQL-based applications to Hadoop.
- While initially developed by Facebook, Apache Hive is used and developed by other companies such as Netflix and the Financial Industry Regulatory Authority (FINRA).
- Amazon maintains a software fork of Apache Hive included in Amazon Elastic MapReduce on Amazon Web Services.

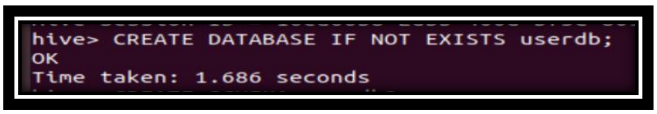
PRACTICAL:

We can download Hive with the help of following tutorials.

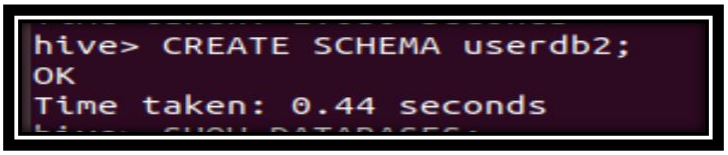
https://phoenixnap.com/kb/install-hive-on-ubuntu

Commands:

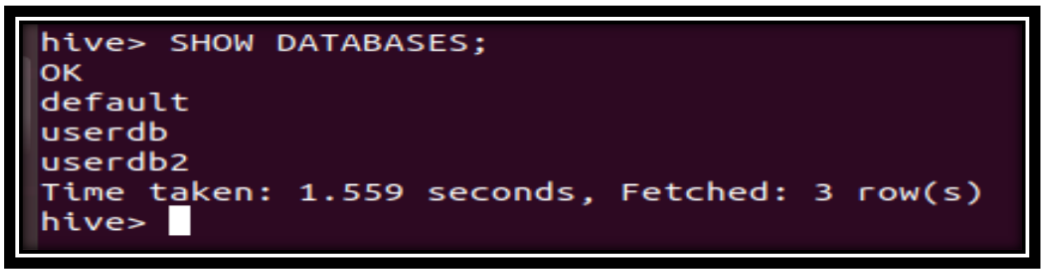
CREATE DATABASE [IF NOT EXISTS] userdb;



CREATE SCHEMA userdb;



SHOW DATABASES;



CREATE TABLE IF NOT EXISTS employee (eid int, name String,

salary String, destination String)

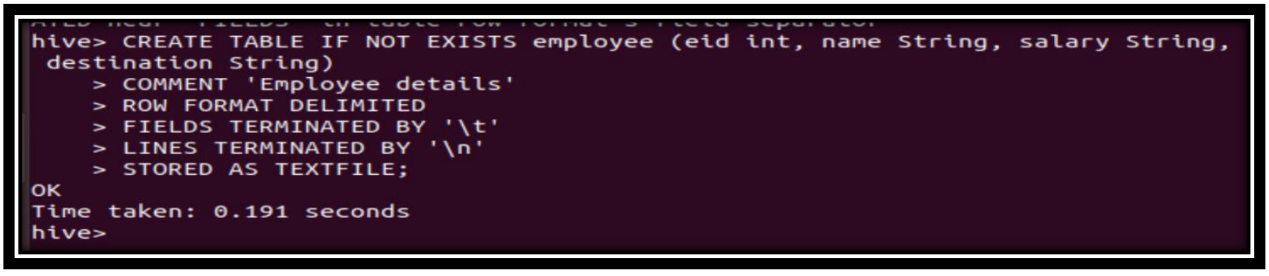
COMMENT ‘Employee details’

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ‘\t’

LINES TERMINATED BY ‘\n’

STORED AS TEXTFILE;



INSERT INTO employee VALUES (1, ‘Robin’, ‘20000’, ‘London’);

SELECT * FROM employee;

8	1	Robin	20000	London
9	4	Leo	30000	Silicon Valley
10	2	Cooper	25000	New York
11	3	Lucy	10000	California

ALTER TABLE name RENAME TO new_name;

```
hive> ALTER TABLE employee RENAME TO new_employee;
OK
Time taken: 0.536 seconds
```

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

```
hive> ALTER TABLE new_employee ADD COLUMNS (designation String);
OK
Time taken: 0.403 seconds
hive>
```

ALTER TABLE name CHANGE column_name new_name new_type;

```
hive> ALTER TABLE new_employee CHANGE designation promotion String;
OK
Time taken: 0.29 seconds
hive>
```

CONCLUSION:

In this practical, we learnt about hive and performed different operations using it.

PRACTICAL 9

AIM:

To install and run Pig and then write Pig Latin scripts to sort, group, join, project, and filter your data.

THEORY:

Pig:

- Apache Pig is a high-level platform for creating programs that run on Apache Hadoop.
- The language for this platform is called Pig Latin.
- Pig can execute its Hadoop jobs in MapReduce, Apache Tez, or Apache Spark.
- Pig Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for relational database management systems.
- Pig Latin can be extended using user-defined functions (UDFs) which the user can write in Java, Python, JavaScript, Ruby or Groovy and then call directly from the language.
- Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows. Pig is generally used with Hadoop; we can perform all the data manipulation operations in Hadoop using Pig.
- Hadoop Pig is nothing but an abstraction over MapReduce. While it comes to analyze large sets of data, as well as to represent them as data flows, we use Apache Pig. Generally, we use it with Hadoop. By using Pig, we can perform all the data manipulation operations in Hadoop.
- In addition, Pig offers a high-level language to write data analysis programs which we call as Pig Latin. One of the major advantages of this language is, it offers several operators.
- Through them, programmers can develop their own functions for reading, writing, and processing data.
- It has following key properties such as:
- Ease of programming
- Basically, when all the complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, that makes them easy to write, understand, and maintain.
- Optimization opportunities
- It allows users to focus on semantics rather than efficiency, to optimize their execution automatically, in which tasks are encoded permits the system.
- Extensibility
- In order to do special-purpose processing, users can create their own functions.
- Hence, programmers need to write scripts using Pig Latin language to analyze data using Apache Pig.

- However, all these scripts are internally converted to Map and Reduce tasks. It is possible with a component; we call as Pig Engine. That accepts the Pig Latin scripts as input and further convert those scripts into MapReduce jobs.

PRACTICAL:

You can start the pig using “pig -x local”

```
admin123@admin123:~$ pig -x local
2021-10-16 23:18:20,023 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2021-10-16 23:18:20,025 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2021-10-16 23:18:20,351 [main] INFO  org.apache.pig.Main - Apache Pig version 0
.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2021-10-16 23:18:20,356 [main] INFO  org.apache.pig.Main - Logging error messag
es to: /home/admin123/pig_1634406500342.log
2021-10-16 23:18:20,509 [main] INFO  org.apache.pig.impl.util.Utils - Default b
ootup file /home/admin123/.pigbootup not found
2021-10-16 23:18:20,772 [main] INFO  org.apache.hadoop.conf.Configuration.depre
cation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.ad
dress
2021-10-16 23:18:20,778 [main] INFO  org.apache.pig.backend.hadoop.executioneng
ine.HExecutionEngine - Connecting to hadoop file system at: file:///
2021-10-16 23:18:21,175 [main] INFO  org.apache.hadoop.conf.Configuration.depre
cation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checks
um
2021-10-16 23:18:21,256 [main] INFO  org.apache.pig.PigServer - Pig Script ID f
or the session: PIG-default-d0a6cf65-8a2c-4384-bf5f-f6801cc9244b
2021-10-16 23:18:21,257 [main] WARN  org.apache.pig.PigServer - ATS is disabled
 since yarn.timeline-service.enabled set to false
grunt> student_details = LOAD './student_details.txt' USING PigStorage(',')
>> as (id:int, firstname:chararray, age:int, city:chararray);
2021-10-16 23:21:21,081 [main] INFO  org.apache.hadoop.conf.Configuration.depre
cation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checks
um
grunt>
```

We will create a file called student_details.txt with input data and load it into pig.

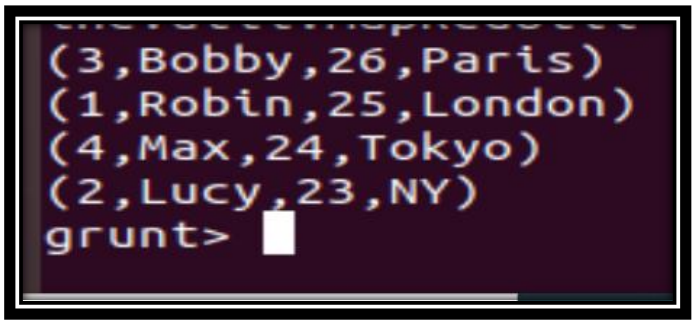
```
Open                                   student_details.txt
                                                ~/
001,Robin,25,London
002,Lucy,23,NY
003,Bobby,26,Paris
004,Max,24,Tokyo
```

```
grunt> student_details = LOAD './student_details.txt' USING PigStorage(',')
>> as (id:int, firstname:chararray, age:int, city:chararray);
2021-10-16 23:27:27,398 [main] INFO  org.apache.hadoop.conf.Configuration.depre
cation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checks
um
```

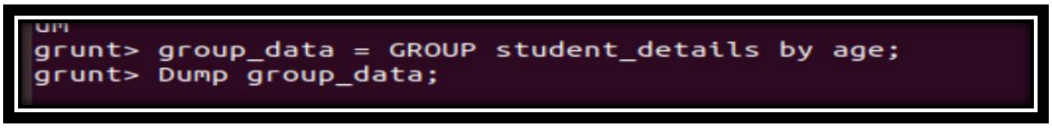
We can perform different operations using following commands.

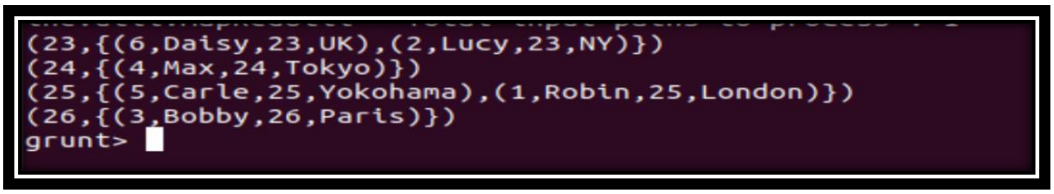
```
Order_by_data = ORDER student_details BY age DESC;

Dumb order_by_data;
```

```
grunt> order_by_data = ORDER student_details BY age DESC;
grunt> Dump order_by_data;
```

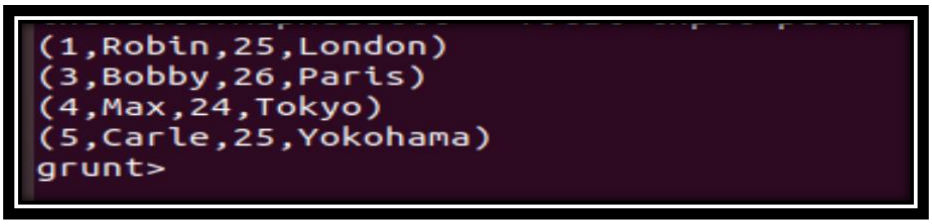



```
Group_data = GROUP student_details by age;

Dump group_data;
```






```
Filter_data = FILTER student_details BY age>23;

Dump filter_data;
```



CONCLUSION:

In this practical, we learnt about pig and used various commands to manipulate data.

PRACTICAL 10

AIM:

To install, deploy & configure Apache Spark Cluster. To Select the fields from the dataset using Spark SQL. To explore Spark shell and read from HDFS.

THEORY:

Spark:

- Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since.
- Apache Spark has its architectural foundation in the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.
- The Dataframe API was released as an abstraction on top of the RDD, followed by the Dataset API. In Spark 1.x, the RDD was the primary application programming interface (API), but as of Spark 2.x use of the Dataset API is encouraged even though the RDD API is not deprecated. The RDD technology still underlies the Dataset API.
- Spark and its RDDs were developed in 2012 in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk.
- Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.
- Inside Apache Spark the workflow is managed as a directed acyclic graph (DAG). Nodes represent RDDs while edges represent the operations on the RDDs.
- Spark facilitates the implementation of both iterative algorithms, which visit their data set multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data.
- The latency of such applications may be reduced by several orders of magnitude compared to Apache Hadoop MapReduce implementation. Among the class of iterative algorithms are the training algorithms for machine learning systems, which formed the initial impetus for developing Apache Spark.
- Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster, where you can launch a cluster either manually or use the launch scripts provided by the install package. It is also possible to run these daemons on a single machine for testing), Hadoop YARN, Apache Mesos or Kubernetes.

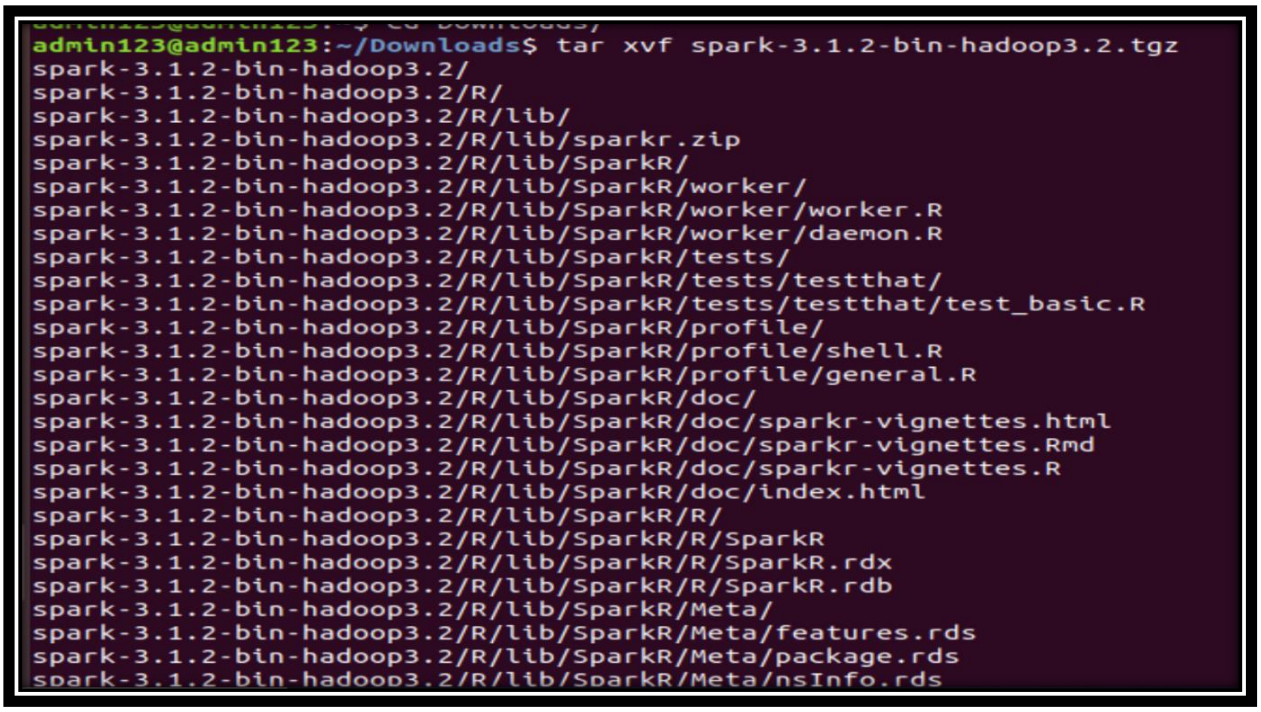
- For distributed storage, Spark can interface with a wide variety, including Alluxio, Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon S3, Kudu, Lustre file system, or a custom solution can be implemented.
- Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.

PRACTICAL:

You can download spark from official download.

Then extract the file using tar command.

```
tar xvf spark-*
```



Finally, move the unpacked directory spark-3.0.1-bin-hadoop2.7 to the opt/spark directory.

```
sudo mv spark-3.0.1-bin-hadoop2.7 /opt/spark
```

Before starting a master server, you need to configure environment variables. There are a few Spark home paths you need to add to the user profile.

Use the echo command to add these three lines to .profile:

```
echo "export SPARK_HOME=/opt/spark" >> ~/.profile

echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin" >> ~/.profile
```

```
echo "export PYSPARK_PYTHON=/usr/bin/python3" >> ~/.profile
```

```
admin123@admin123:~/Downloads$ sudo mv spark-3.1.2-bin-hadoop3.2 /opt/spark
[sudo] password for admin123:
admin123@admin123:~/Downloads$ echo "export SPARK_HOME=/opt/spark" >> ~/.profil
e
admin123@admin123:~/Downloads$ echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_H
OME/sbin" >> ~/.profile
admin123@admin123:~/Downloads$ echo "export PYSPARK_PYTHON=/usr/bin/python3" >>
 ~/.profile
admin123@admin123:~/Downloads$ source ~/.profile
admin123@admin123:~/Downloads$ start-master.sh
start-master.sh: command not found
admin123@admin123:~/Downloads$
```

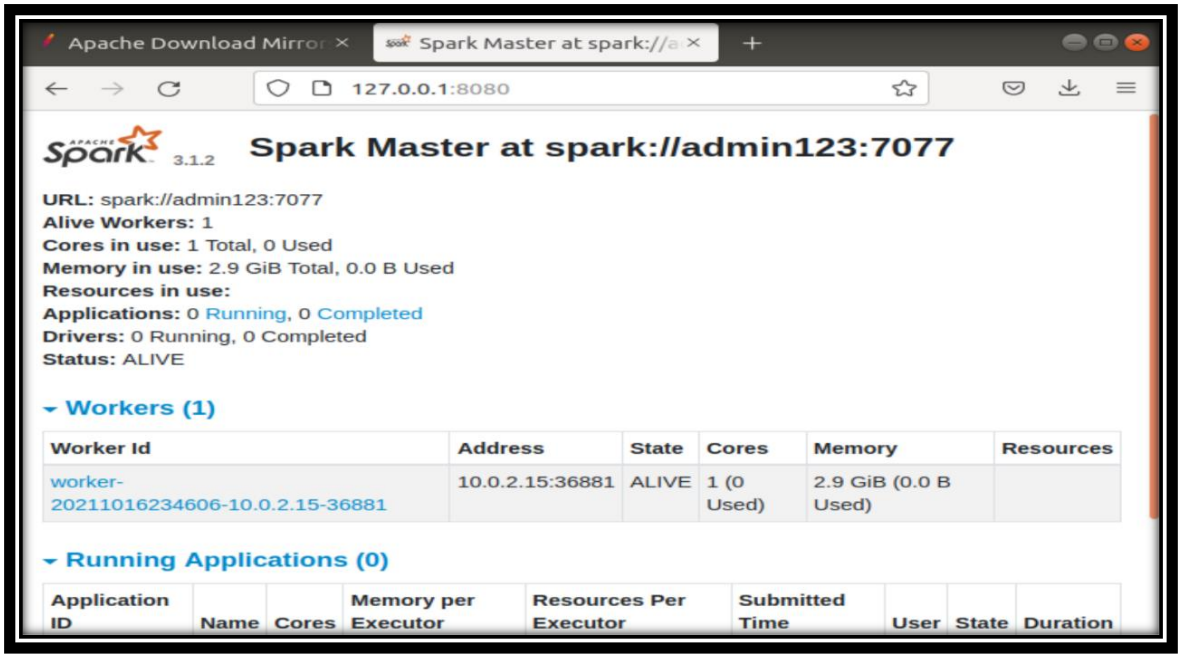
Now that you have completed configuring your environment for Spark, you can start a master server.

In the terminal, type:

```
start-master.sh
```

```
admin123@admin123:/opt/spark/sbin$ ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spar
k-admin123-org.apache.spark.deploy.master.Master-1-admin123.out
```

Now that a worker is up and running, if you reload Spark Master's Web UI, you should see it on the list:



In this single-server, standalone setup, we will start one slave server along with the master server.

To do so, run the following command in this format:

```
start-slave.sh spark://master:port
```

The master in the command can be an IP or hostname.

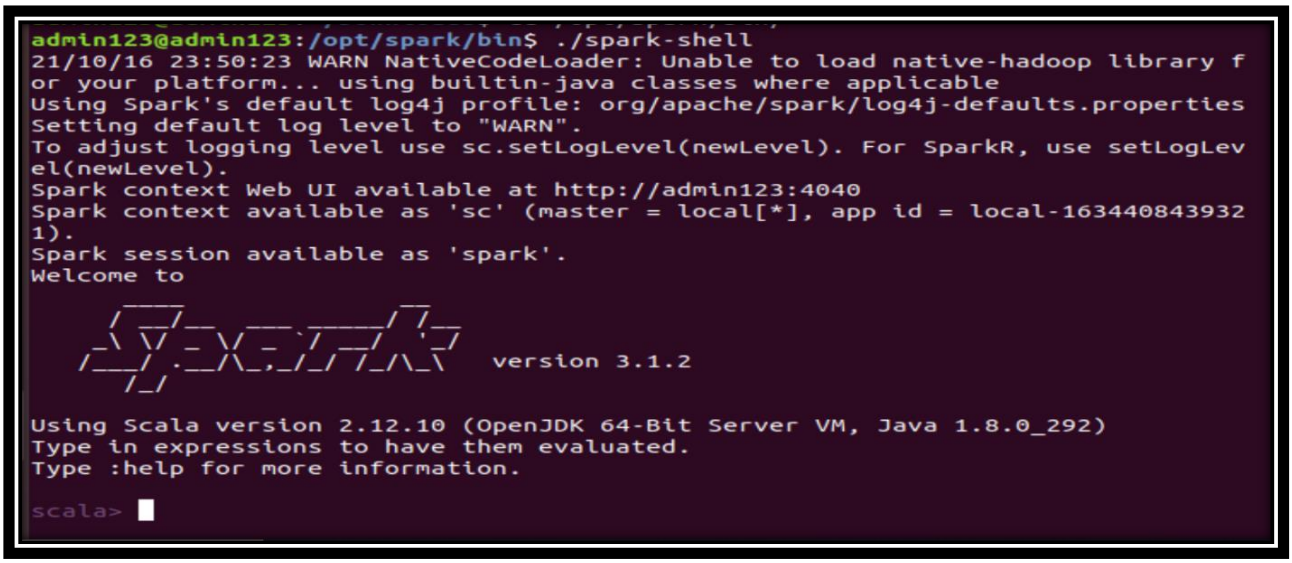
In our case it is ubuntu1:

```
start-slave.sh spark://ubuntu1:7077
```

After you finish the configuration and start the master and slave server, test if the Spark shell works.

Load the shell by entering:

```
spark-shell
```



CONCLUSION:

In this practical, we learnt about spark and installed it and configured it. We explored the spark shell as well.

PRACTICAL 11

AIM: To perform Sentiment Analysis using Twitter data, Scala and Spark.

Implementation:

Reading data

Code:

```
from pyspark import SparkConf, SparkContext
import sys

conf = SparkConf().setMaster("local").setAppName("RatingsHistogram")
sc = SparkContext(conf = conf)

# Create a hardcoded RDD
numbers = sc.parallelize([1, 2, 3, 4])

# Load plain text files (e.g. CSVs) from different sources
hdfs_lines = sc.textFile("hdfs:///user/cloudera/ml-100k/u.data", minPartitions=1)
local_lines = sc.textFile("file:///home/alex/ml-100k/u.data")
s3_lines = sc.textFile("s3n://bucket/ml-100k/u.data")
CLI_arg_lines = sc.textFile(sys.argv[1])

# Create RDDs from an existing Hive repository
hive_ctx = HiveContext(sc)
hive_lines = hive_ctx.sql("SELECT name, age FROM users WHERE age > 18")
```

- Notice that this can not be run with the standard Python interpreter. Instead, you use the spark-submit to submit it as a batch task, or name the Shell Pyspark.

Writing of data:

- The RDD class has the method saveAsTextFile. However, this saves the representation of each variable in a string. In Python, the resulting text file will contain lines such as (1949, 111).
- If you want to save the data in CSV or TSV format, you can either use Python's StringIO and csv modules or simply map each variable (vector) into a single string, e.g. as follows:

Code:

```
res.saveAsTextFile("hdfs:///user/cloudera/res_raw.txt")  # bad format

res.map(lambda row: str(row[0]) + "\t" + str(row[1])) \
  .saveAsTextFile("hdfs:///user/cloudera/res_tsv.txt")  # good format
```

Sentiment Analysis using Scala:

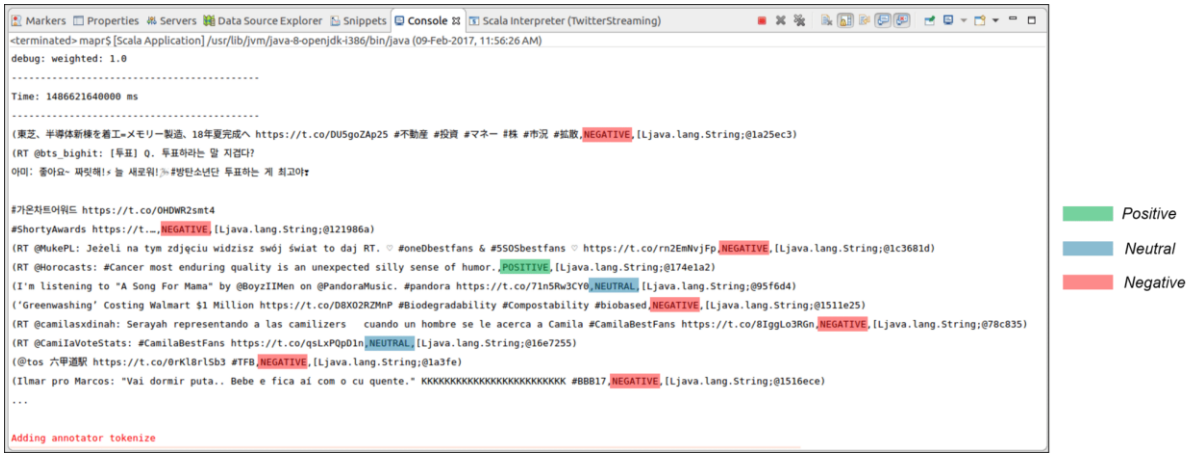
Spark Streaming Implementation:

Code:

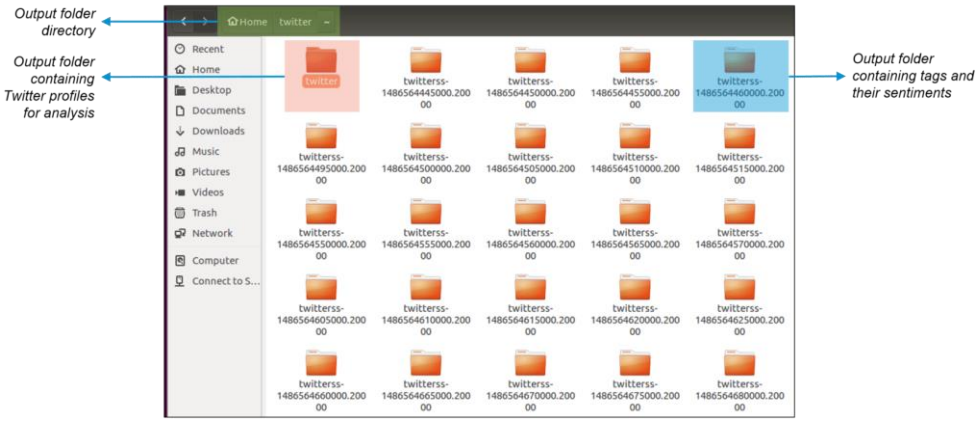
```
//Import the necessary packages into the Spark Program
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.SparkContext._
...
import java.io.File

object twitterSentiment {

def main(args: Array[String]) {
if (args.length < 4) {
System.err.println("Usage: TwitterPopularTags <consumer key> <consumer secret>
System.exit(1)
}

StreamingExamples.setStreamingLogLevels()
//Passing our Twitter keys and tokens as arguments for authorization
val Array(consumerKey, consumerSecret, accessToken, accessTokenSecret) = args.
val filters = args.takeRight(args.length - 4)

// Set the system properties so that Twitter4j library used by twitter stream
// Use them to generate OAuth credentials
System.setProperty("twitter4j.oauth.consumerKey", consumerKey)
...
System.setProperty("twitter4j.oauth.accessTokenSecret", accessTokenSecret)

val sparkConf = new SparkConf().setAppName("twitterSentiment").setMaster("loca
val ssc = new Streaming Context
val stream = TwitterUtils.createStream(ssc, None, filters)

//Input DStream transformation using flatMap
val tags = stream.flatMap { status => Get Text From The Hashtags }

//RDD transformation using sortBy and then map function
tags.countByValue()
.foreachRDD { rdd =>
val now = Get current time of each Tweet
rdd
.sortBy(_._2)
.map(x => (x, now))
//Saving our output at ~/twitter/ directory
.saveAsTextFile(s"~/twitter/$now")
}

val data = tweets.map { status =>
val sentiment = SentimentAnalysisUtils.detectSentiment(status.getText)
val tagss = status.getHashtagEntities.map(_.getText.toLowerCase)
(status.getText, sentiment.toString, tagss.toString())
}

data.print()
//Saving our output at ~/ with filenames starting like twitters
data.saveAsTextFiles("~/twitters","20000")

ssc.start()
ssc.awaitTermination()
 }
}
```

Results:

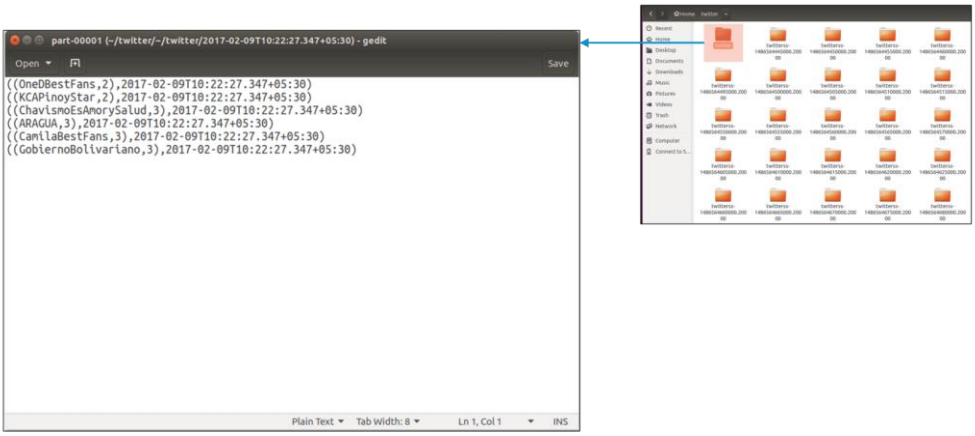
The following are the results shown in the Eclipse IDE when running the Twitter Sentiment Streaming software.



- As we can see from the screenshot, all tweets are categorised as Positive, Neutral and Negative according to the feelings of the contents of the tweets.
- The output of the Sentiments of the Tweets is stored in folders and files, depending on the time they were created. This output can be stored on the local file system or HDFS as appropriate. The output directory looks like this one:

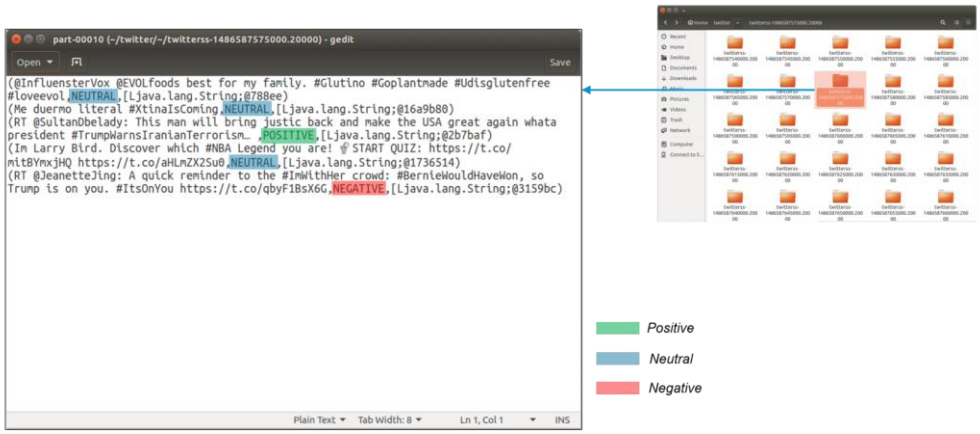


- Here, within the twitter folders, we will find the usernames of the Twitter users and the timestamp for. tweet as seen below:

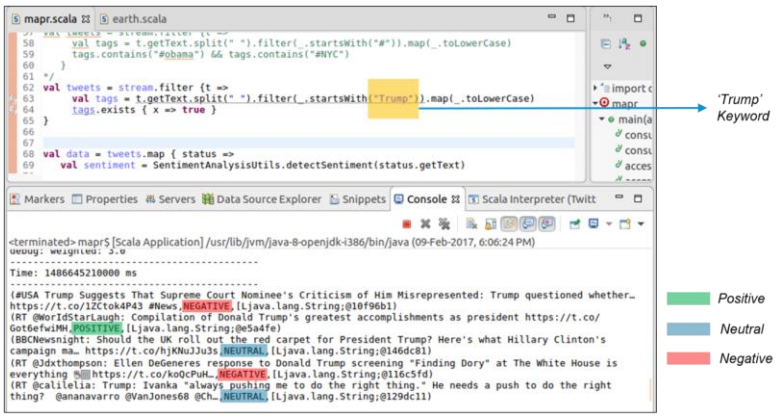


- Now that we have Twitter usernames and timestamps, let 's look at the Sentiments and Tweets saved in the main directory. Here, every tweet is accompanied by a feeling of emotion. This Sentiment, which is processed, is further used to examine a broad variety of business observations.

s



- Tweaking Code: Now, let's modify our code a little to get feelings about specific hashtags (topics). Donald Trump, the President of the United States, is now travelling through television outlets and online social media. Let us look at the emotions associated with the keyword 'Trump.'



CONCLUSION:

In this practical, we learned Sentiment Analysis using Twitter data, Scala and Spark.

PRACTICAL 12

AIM:

To perform Graph Path and Connectivity analytics and implement basic queries after loading data using Neo4j.

THEORY:

Neo4j:

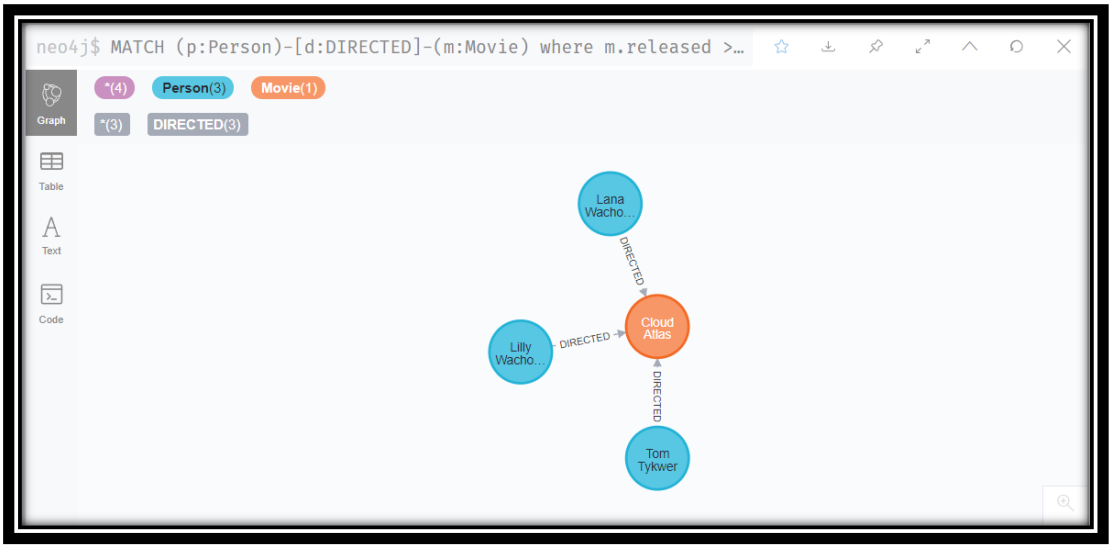
- Neo4j is a graph database management system developed by Neo4j, Inc. Described by its developers as an ACID-compliant transactional database with native graph storage and processing, Neo4j is available in a GPL3-licensed open-source "community edition", with online backup and high availability extensions licensed under a closed-source commercial license.
- Neo also licenses Neo4j with these extensions under closed-source commercial terms.
- Neo4j is implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint, or through the binary "Bolt" protocol.
- In Neo4j, everything is stored in the form of an edge, node, or attribute. Each node and edge can have any number of attributes. Both nodes and edges can be labelled. Labels can be used to narrow searches. As of version 2.0, indexing was added to Cypher with the introduction of schemas. Previously, indexes were supported separately from Cypher.
- Neo4j is developed by Neo4j, Inc., based in the San Francisco Bay Area, United States, and also in Malmö, Sweden. The Neo4j, Inc. board of directors consists of Michael Treskow (Eight Roads), Emmanuel Lang (Greenbridge), Christian Jepsen, Denise Persson (CMO of Snowflake), David Klein (One Peak), and Emil Eifrem (CEO of Neo4j).
- Neo4j comes in 2 editions: Community and Enterprise. It is dual-licensed: GPL v3 and a commercial license. The Community Edition is free but is limited to running on one node only due to the lack of clustering and is without hot backups.
- The Enterprise Edition unlocks these limitations, allowing for clustering, hot backups, and monitoring. The Enterprise Edition is available under a closed-source Commercial license.

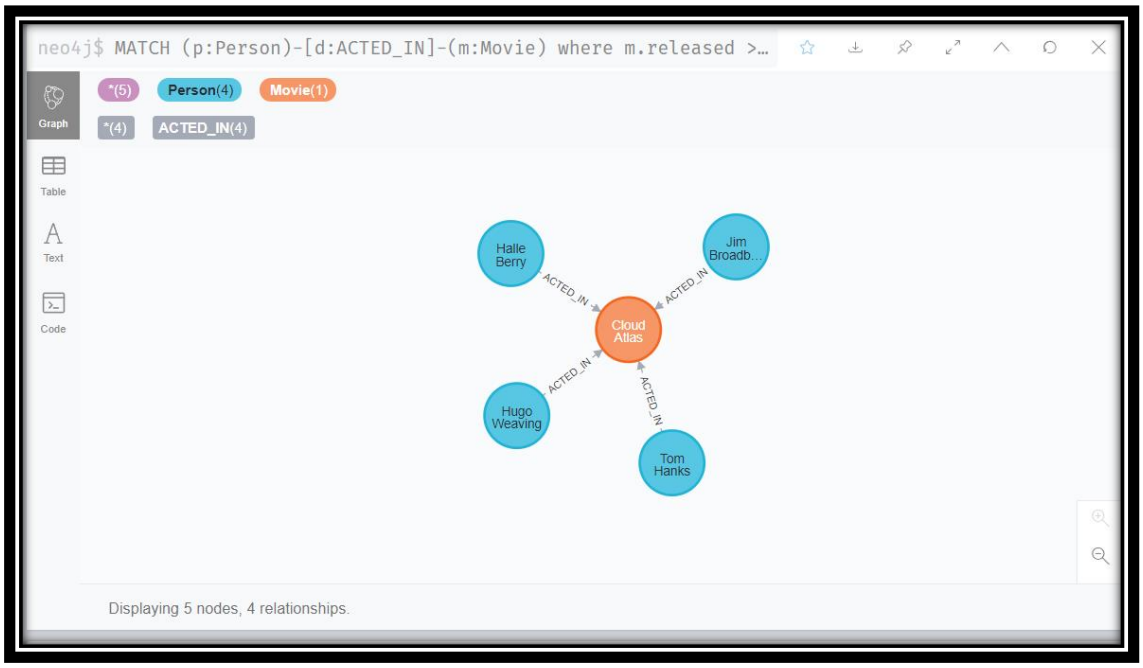
PRACTICAL:

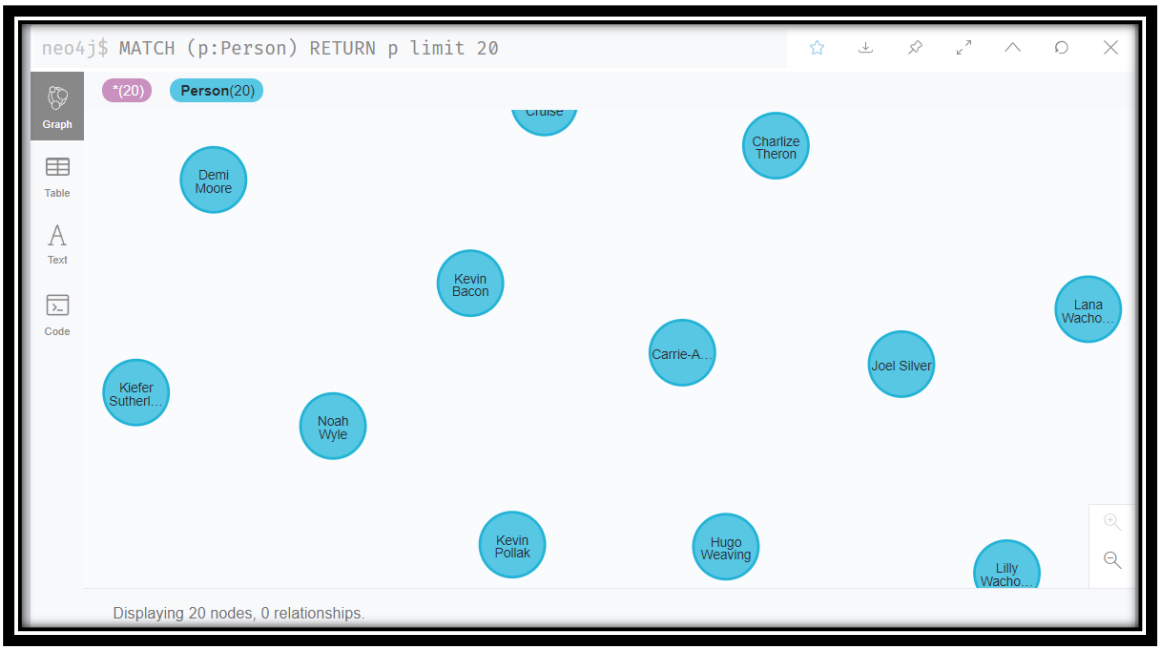
We will use neo4j sandbox to fire different queries as shown in the screen shots below.

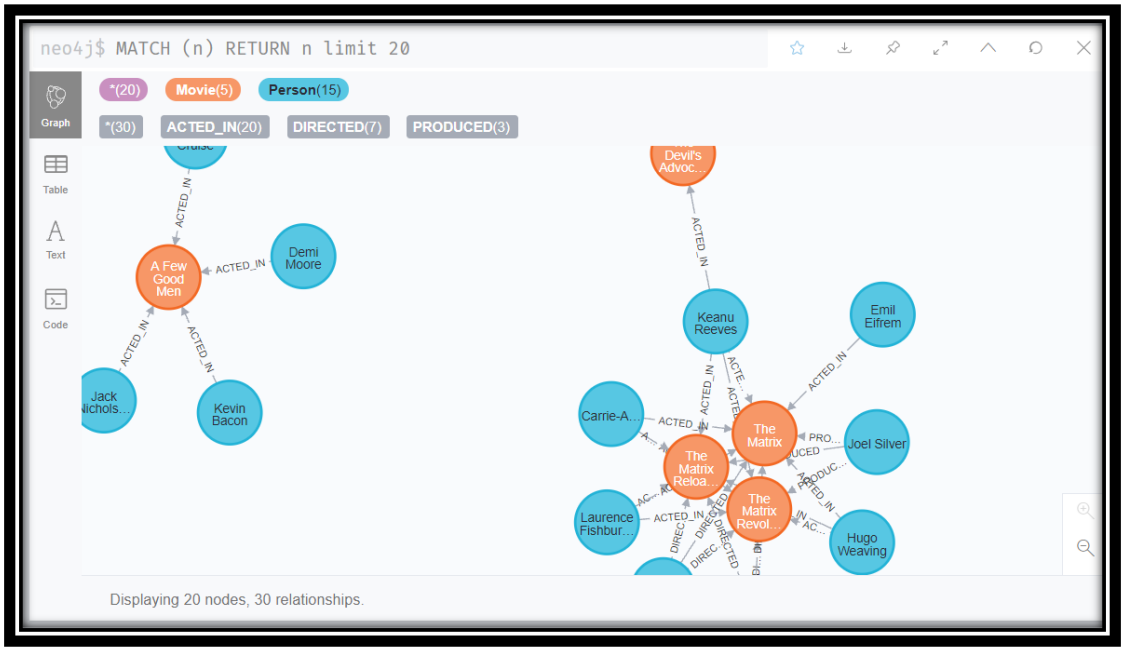

neo4j$ Match (m:Movie) where m.released > 2005 RETURN m
*(8)
Movie(8)
Graph
Table
Text
Code
Speed Racer
Charlie Wilson's War
Frost/Ni...
V for Vendetta
Ninja Assassin
The Da Vinci Code
Cloud Atlas
Displaying 8 nodes, 0 relationships.


neo4j$ Match (m:Movie) where m.released > 2005 RETURN count(m)
Table
Text
Code
count(m)
1
8


neo4j$ MATCH (p:Person)-[d:DIRECTED]-(m:Movie) where m.released >…
Graph
Table
Text
Code
*(4)
Person(3)
Movie(1)
*(3)
DIRECTED(3)
Lana Wacho...
DIRECTED
Cloud Atlas
Lilly Wacho...
DIRECTED
DIRECTED
Tom Tykwer


neo4j$ MATCH (p:Person)-[d:ACTED_IN]-(m:Movie) where m.released >…
Graph
Table
Text
Code
*(5)
Person(4)
Movie(1)
*(4)
ACTED_IN(4)
Halle Berry
Jim Broadb…
Cloud Atlas
Hugo Weaving
Tom Hanks
ACTED_IN
Displaying 5 nodes, 4 relationships.


neo4j$ MATCH (p:Person) RETURN p limit 20
Graph
Table
Text
Code
*(20)
Person(20)
Demi Moore
Charlize Theron
Kevin Bacon
Lana Wacho…
Carrie-A…
Joel Silver
Kiefer Sutherl…
Noah Wyle
Kevin Pollak
Hugo Weaving
Lilly Wacho…
Displaying 20 nodes, 0 relationships.



CONCLUSION:

In this practical, we learnt about neo4j and used different type of queries on sandbox on movie dataset.

PRACTICAL 13

AIM:

To perform a case study of the following platforms for solving any big data analytic problem of your choice. (1) Amazon web services, (2) Microsoft Azure, (3)Google App engine.

IMPLEMENTATION:

1. AWS: OLX Autos Delivers a Seamless Online Marketplace by Running Containers on AWS

- OLX Autos' goal is to enable people to better their life through its online marketplace of services and used goods that offer superior consumer value.
- As part of the OLX Group, which runs more than 20 brands in 30 countries, OLX Autos blends the versatility of a start-up with the experience of a multinational entity.
- With a vision to shape the future of commerce, OLX Autos is excited about leveraging state-of-the-art technologies to improve consumer service.
- India is one of the world's biggest growth markets for OLX Cars.
- The organization has a highly innovative technology team running its Panamera Classified Marketplace, which is active in 12 countries, on the Amazon Web Services (AWS) cloud.
- OLX Autos is committed to investing in its people and the new technologies to unlock optimum value for its customers.

"Using AWS has kept our environment running in a stable manner in the most cost-optimized way." Abhishek Tomar Infrastructure Head, OLX Autos

➔ **Modernizing Dynamic Architectural:**

- OLX Autos suffered an outage on its platform in 2018 after its internal OpenShift containerization program licenses expired.
- The organization used OpenShift version 3.5 to handle its resources and install software on its Docker container site. As part of the company's pledge to provide consumers with a seamless experience, engineers quickly found the problem and rapidly rolled out new certificates.
- However, they had to spend a substantial amount of time rebuilding the website. The team concluded that their staging environment was dysfunctional due to problems with the OpenShift Control Panel and started searching for solutions to modernize their infrastructure stack.
- Abhishek Tomar, infrastructure manager at OLX Autos, says, "Buyers and sellers were unable to check, view new advertisements, or build new listings during the shutdown. We found that even though we had just one accident a year, it would always be easier to unload the control and the guy.

➔ **Seamlessly Migrating 124 Microservices:**

- OLX Autos launched a conversation with the AWS team a year before the migration began. "First, we reached out to AWS Business Support to provide guidance on the right migration approach," says Sharma. At the same time, OLX Autos revamped its operating system, Kubernetes, and Docker versions, and the difficulty of its OpenShift architecture, in addition to software version limitations, made migration especially difficult.
- The OLX Autos SRE team prepared a relocation approach and began shifting their staging environment to AWS in January 2020. "We received immediate assistance from AWS to address any problems in the staging area that helped us move seamlessly to the production environment," says Sharma.
- The AWS team proposed a plan for future new capabilities of Panamera using Amazon EKS in addition to other AWS offerings. OLX Autos then began constructing an Amazon EKS cluster and migrating selected workloads from OpenShift, which will be finished in only a few weeks.
- OLX Autos has now migrated all 124 of its microservices to Amazon EKS, which has improved efficiency and scalability while unlocking cost savings.

➔ **Achieving Efficiency:**

- After relocation, OLX Autos infrastructure teams have now become well prepared to meet crucial internal product delivery deadlines.
- The organization originally used the Puppet Setup Tool to handle its OpenShift program.
- Engineers wanted to focus on their Puppet expertise and spent three to four days a month applying and tracking big improvements to the OLX Autos infrastructure.
- Through the migration to AWS, the organization removed Puppet from its design and switched container control to AWS.
- By operating on Amazon EKS, the OLX Autos website benefits from increased performance and scalability, and engineers may redirect their time to higher value-added activities.

➔ **Scaling in Minutes and Improving Uptime:**

- When OLX Autos was running an old version of OpenShift, the company faced autoscaling difficulties, which triggered interruptions to its applications. "Now with Amazon EKS, our Amazon Elastic Compute Cloud (Amazon EC2) fleet can be auto-scaled in a few minutes if some marketing plan triggers a surge or a rapid surge in traffic.
- If online traffic declines, the fleet will be scaled down — something we haven't been able to do with OpenShift before, "says Tomar.
- OLX Autos has also unloaded the vital role of handling Stable Sockets Layer (SSL)/Transport Layer Security (TLS) certificates to AWS Credential Manager. Previously, teams had to manually buy and instal new certificates each year, but with AWS, new certificates are deployed with a few quick API calls.
- "It's a relief that AWS is now going to take care of that," says Tomar. "We don't have to spend a single minute in testing and upgrading certificates that may have an effect on crucial deadlines for our company."

➔ **Unlocking savings with a reserved instance:**

- OLX Autos had reserved a number of Amazon EC2 reserved instances for form C5 instances but was unable to use them for the older version of OpenShift. "As a part of this conversion and version update, we have transferred all of our C4 instances to C5 and will leverage the price of reserved instances.
- That eventually saved us 33% of the cost of computation, "says Tomar.
- OLX Autos have benefited from the AWS Container Network Interface (CNI) plug-in, which was blocked by its previous OpenShift setup.
- Not only has the CNI plug-in increased device latency, but OLX Autos now hopes to save at least 10% of its average monthly AWS bill for more efficient applications. Tomar says, "Using AWS has kept our climate secure in the most cost-optimized way."

2. **Microsoft Azure:** Health data analytics company accelerates critical data insights with SQL Server 2019 Big Data Clusters.
   - Vital responses to overall health care patterns, such as COVID-19 test outcomes, demand consistency, and quick delivery times. That's why businesses like the Telstra Health affiliate Dr. Foster are so important to transforming care.
   - Headquartered in London, the healthcare analytics organization responded to the need to make large databases accessible to various teams, to offer analytics to its healthcare clients quicker with the Microsoft SQL Server 2019 Big Data Clusters, and to secure confidential data.
   - These benefits translate into improved customer service, increased competitive edge, and improved health insurance for all.

We're building next-generation services that cut through complexity to surface actionable insights to customers and pinpoint potential areas of concern. We needed a data analytics platform to accelerate that goal and when I learned about SQL Server 2019 Big Data Clusters, it aligned with everything we were trying to achieve.

George Bayliffe: Head of Data
Dr Foster

- Digitized patient records are replacing dog-eared paper files, driving the rapid growth of the health care analytics field. Leading UK health analytics company Dr Foster helps hospitals understand the factors influencing the quality of care. To deliver value, Dr Foster must aggregate and analyze vast amounts of data.

➔ **Stepping up to diverse demands:**

- Dr Foster is creating a 'one-stop shop' for data through a single access point. To surface vital insights quickly, up to five different teams will touch a given data store.
- Dr Foster has relied on the same technology stack for a long time, making minimal changes. "We're building next-generation services that cut through complexity to surface actionable insights to customers," says George Bayliffe, Head of Data at Dr Foster.

- Dr Foster operates in a highly regulated space with stringent customer requirements and regulations like the General Data Protection Regulation (GDPR)

- To help ensure high-velocity service delivery, his teams require tools that foster agility, says Bayliffe, who works to create a DevOps culture across all aspects of delivery.
- "We're custodians of other people's data, and we never lose sight of that," he says, adding: "We treat it with the acute care and attention it deserves"

➔ **Staying ahead of the curve with big data technology:**

- Reimagining the architecture at Dr Foster was never going to be an overnight job. The engineering team needed to expand its expertise to containerized storage.
- The company needed persistent and non-volatile storage, and containers aren't designed to persist after being rebuilt and moved.
- "Despite any concerns we had about making such a big change, the perceived benefits outweighed all the risks," says Dr Foster's engineer.
- "With those initial challenges behind us, we're now in a strong position," he says. Dr Foster runs on Dell EMC PowerEdge R630 servers with embedded Integrated Dell Remote Access Controller (iDRAC) to easily update, monitor, and maintain each server.

➔ **Fixing on future needs:**

- Many customers require Dr Foster to maintain their data on-premises, precluding the public cloud. The company prefers the cloud for its own data and for answering the needs of some of its international customers.
- "We're cloud-ready with our hybrid model, and we can transition to Microsoft Azure when the need arises," says Bayliffe. Dr Foster aspires to a scalable platform that responds to peak and fast-growing demand.

➔ **Lowering storage costs even as performance rises:**

- The combination of SQL Server 2019 Big Data Clusters and Pure Storage FlashArray creates a performance advantage. Core data processing cycle times are 45 percent lower, according to Dr Foster.
- The data reduction in the array gives the company 50–60 TB of storage from a purchase of 30–40 TB. "If we'd embarked on this project five years ago, we'd probably have had to buy double or triple that amount of storage," says Bayliff.
- Dr Foster says it can answer any need that comes up in the next three to four years with the new technology.

➔ **Responding to a pandemic:**

- When COVID-19 forced lockdowns all over the world, the re-engineered Dr Foster landscape eased the sudden transition for its employees.
- The company continues to meet via Microsoft Teams, using the whiteboarding functionality to collaborate. Dr Foster is beginning to process 60,000 test results from parent company Telstra Health in Australia each morning. The solution allows the company to react as the market requires, says Bayliffe. "Because of the SQL Server 2019 Big Data Clusters solution and the architecture we've put into place, we can react to the market," he says.

We're cloud-ready with our hybrid model, and we can transition to Microsoft Azure when the need arises.

George Bayliffe: Head of Data
Dr Foster

3. **Google App Engine: Travlytix: Turns to Google Cloud to run a customer data platform and personalization engine**
    - With big data products running on Google Cloud Platform, Travlytix created an intelligent customer data platform for airlines in six months with a small engineering team.
    - Travelers generate data across a range of touchpoints, presenting opportunities for businesses to understand more about how they behave. Malaysia-headquartered Travlytix decided to create a platform to consolidate real-time data from these touchpoints into a single location. The business also had to ensure the data was secured from leakage, disruption, or theft.

"Google Cloud Platform services like Cloud Pub/Sub, Cloud Dataflow, and Big Query allow us to minimize the effort and resources needed to build data pipelines for airline customers and focus instead on the quality, volume, and velocity of data."

—Srinivas Sri Perumbuduru, Head of Product, Travlytix

- Airlines, online travel agents, and travel eCommerce companies face challenges in capturing, processing and utilizing data. The business turned to the cloud to deliver its platform. "The cloud gave us an opportunity to work with large volumes of data while becoming more agile," says Travlytix.
- When COVID-19 forced lockdowns all over the world, the re-engineered Dr. Foster landscape eased the sudden transition for its employees.
- The company continues to meet via Microsoft Teams, using the whiteboarding functionality to collaborate. Dr. Foster is beginning to process 60,000 test results from parent company Telstra Health in Australia each morning.
- The solution allows the company to react as the market requires, says Bayliff. "Because of the SQL Server 2019 Big Data Clusters solution and the architecture we've put into place, we can react to the market," he says.

➔ **Making data transparent:**

- "Over the history of the industry, people have used data to make more educated choices," adds Srinivas Sri Perumbuduru, Product Manager at Travlytix.
- "Our goal in Travelytix is to make the data clearer."
- Travlytix needed a highly available, resilient, and secure cloud platform to deliver what Sri Perumbuduru calls "one of the most workable customer data platforms in the travel industry." The corporation then switched to the Google Cloud Network.

➔ **Google Cloud Platform supports architecture principles:**

- Travlytix's review showed that Google Cloud Platform will support its founding principles of keeping its architecture basic, microservice-based, and highly secure.
- "The documentation and expertise of the Google Cloud Platform and simple contact with Google's customer engineers have made this possible," states Sri Perumbuduru.
- With Google Cloud Technology, the company was able to develop consumer data technology in just six months.

- Travlytix uses Google Cloud Platform services to collect, process, and store traveler data.
- The platform captures data in real-time, processes it through data pipelines, and stores it in a data warehouse.
- TravlyTix also relies on Compute Engine to provide compute resources; Google Kubernetes Engine to manage and orchestrate Docker containers; App Engine to develop and host its applications, and Cloud Memorystore to provide caches to enable sub-millisecond data access.
- "Google Cloud managed services give us more time to strategize our data pipeline architecture and allow us to be less concerned about maintaining the infrastructure," says Kevin Chin, Data Engineer, Travleytix.

➔ **Low-latency services:**

- Travlytix also utilizes several Google Cloud Network zones to offer ongoing, low-latency support to clients around the globe.
- In addition, applications such as the Cloud Key Management Service, which helps companies to control cryptographic keys and secure confidential data on the Google Cloud Network, facilitate compliance with global security regulations.
- These support Travlytix's own data encryption during collection and processing to protect it from intrusion.

CONCLUSION:

In this practical, we perform a case study on big data analytic problems on Amazon web services, Microsoft Azure, Google App Engine.