1. Installation, Configuration, and Running of Hadoop and HDFS.

Open Ubuntu Terminal and enter the following commands for Hadoop Installation, configuration and running HDFS files.

1. ***Install java jdk 8***
   sudo apt install openjdk-8-jdk -y

2. ***sudo nano .bashrc***
   ➔ open .bashrc file and paste these commands

   ```
   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
   export PATH=$PATH:/usr/lib/jvm/java-8-openjdk-amd64/bin
   export HADOOP_HOME=~/hadoop-3.2.4/
   export PATH=$PATH:$HADOOP_HOME/bin
   export PATH=$PATH:$HADOOP_HOME/sbin
   export HADOOP_MAPRED_HOME=$HADOOP_HOME
   export YARN_HOME=$HADOOP_HOME
   export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
   export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
   export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
   export HADOOP_STREAMING=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.4.jar
   export HADOOP_LOG_DIR=$HADOOP_HOME/logs
   export PDSH_RCMD_TYPE=ssh
   ```

3. ***sudo apt-get install ssh***

4. ***Download the Hadoop tar file***
   wget https://downloads.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz

5. ***Extract the tar file***
   tar xzf hadoop-3.2.4.tar.gz

6. ***Change directory to hadoop***
   cd hadoop-3.2.4/etc/hadoop

7. ***set path for JAVA_HOME***
   sudo nano hadoop-env.sh
   JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

8. ***sudo nano core-site.xml***

```
<configuration>
 <property>
 <name>fs.defaultFS</name>
 <value>hdfs://localhost:9000</value>  </property>
 <property>
<name>hadoop.proxyuser.dataflair.groups</name> <value>*</value>
 </property>
 <property>
<name>hadoop.proxyuser.dataflair.hosts</name> <value>*</value>
 </property>
 <property>
<name>hadoop.proxyuser.server.hosts</name> <value>*</value>
 </property>
 <property>
<name>hadoop.proxyuser.server.groups</name> <value>*</value>
 </property>
</configuration>
```

9. ***sudo nano hdfs-site.xml***

```
<configuration>
 <property>
 <name>dfs.replication</name>
 <value>1</value>
 </property>
</configuration>
```

10. ***sudo nano mapred-site.xml***

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

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_
HOME/share/hadoop/mapreduce/lib/*</value>
 </property>
</configuration>
```

11. ***sudo nano yarn-site.xml***

```
<configuration>
 <property>
 <name>yarn.nodemanager.aux-services</name>
 <value>mapreduce_shuffle</value>
 </property>
 <property>
 <name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP
_CONF_DIR,CLASSPATH_PREP
END_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
 </property>
</configuration>
```

12. localhost commands

- ➔ ssh localhost
- ➔ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
- ➔ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
- ➔ chmod 0600 ~/.ssh/authorized_keys
- ➔ hadoop-3.2.4/bin/hdfs namenode -format

13. format the file system

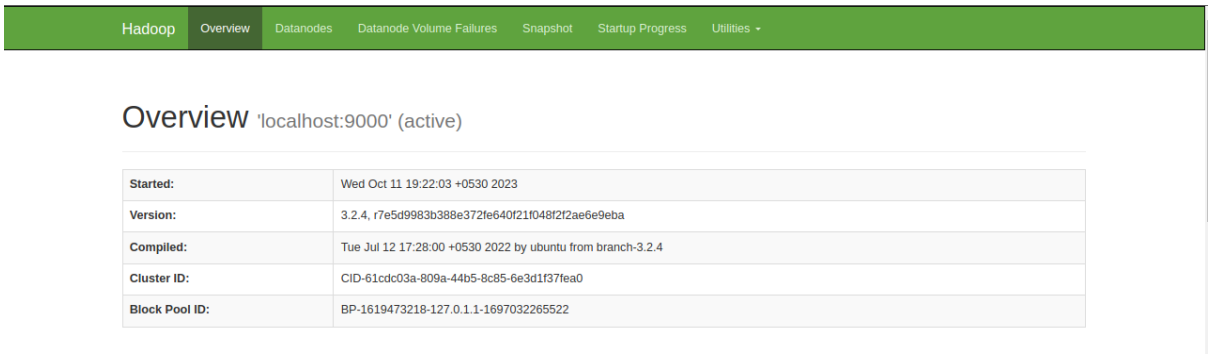
export PDSH_RCMD_TYPE=ssh

14. To start

start-all.sh

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

https://localhost:9870



15. To stop

stop-all.sh

2. Implement the following file management tasks in Hadoop: Adding files and directories, retrieving files and Deleting files.

1. ***Create a Directory***
   hdfs dfs -mkdir -p tdata

2. ***Insert a file into the directory***
   hdfs dfs -put /home/veeranna/Downloads/input.txt tdata/

3. ***Copy the file from hadoop to local directory***
   hdfs dfs -get tdata/input.txt /home/veeranna/

4. ***Create empty file in hdfs***
   hdfs dfs -touchz tdata/test.txt

5. ***Read the content from the file***
   hdfs dfs -cat tdata/test.txt

6. ***Copy From Local and copy To Local***
   hdfs dfs -copyFromLocal /home/veeranna/demo.txt tdata/
   hdfs dfs -copyToLocal tdata/test.txt test.txt.hdfs

7. ***To set replication factor***
   hdfs dfs -setrep -w 5 tdata/test.txt

   Output → Replication 5 set: tdata/test.txt
   Waiting for tdata/test.txt ... done

8. ***To get replication factor***
   hdfs dfs -stat "%r" tdata/test.txt

   Output → 5

9. ***List of files of directory***
   hdfs dfs -ls

   Output → Found 1 items
   drwxr-xr-x - veeranna supergroup 0 2023-09-03 11:34 tdata

10. ***Copy the file content from one location to other***
    hdfs dfs -cp tdata/input.txt test

11. ***Move file from one place to another***
    hdfs dfs -mv tdata/demo.txt test

12. ***To delete a directory***
    hadoop fs -rm -r /user/veeranna/test

    Output → Deleted /user/veeranna/test

3. Implementation of Word Count / Frequency Programs using MapReduce.

Steps to run Hadoop Map Reduce Program:

1. **Launch Eclipse and set the Eclipse Workspace.**
2. **create Project,** click on File→ New→Java Project.
   Note: Choose "JavaSE-1.8" while creating the project
3. ***Create a new Package,* right-click on the Project Name→New→Package.**
   ➔ **Provide the package name: org.myorg**
4. **Add the Hadoop libraries (jars).**
   ➔ Right-Click on Project Name → Build Path → configure Build Path.
   ➔ Add the External jars.
   ➔ go to hadoop-3.2.4 → share → hadoop.
      1) Add the client jar files.
      2) Add common jar files.
      3) Add yarn jar files.
      4) Add MapReduce jar files.
      5) Add HDFS jar files.

   Click Open and apply.
5. **Create a new class,** provide class name as **"WordCountMapper"**

   ➔ **WordCountMapper.java**

```
package org.myorg.Demo;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.io.LongWritable;

public class WordCountMapper extends Mapper <LongWritable, Text, Text, IntWritable>
{
        private Text wordToken = new Text();
        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException
        {
                StringTokenizer tokens = new StringTokenizer(value.toString());
                //Dividing String into tokens
                while (tokens.hasMoreTokens())
                {
                        wordToken.set(tokens.nextToken());
                        context.write(wordToken, new IntWritable(1));
                }
        }
}
```

6. Create another class that performs the reduce job

➔ WordCountReducer.java

```
package org.myorg.Demo;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer <Text, IntWritable, Text, IntWritable>
{
        private IntWritable count = new IntWritable();
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException
        {
                int valueSum = 0;
                for (IntWritable val : values)
                {
                        valueSum += val.get();
                }
                count.set(valueSum);
                context.write(key, count);
        }
}
```

7. create the driver class, which contains the main method.

➔ WordCount.java

```
package org.myorg.Demo;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount
{
        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(WordCountMapper.class);
                job.setCombinerClass(WordCountReducer.class);
                job.setReducerClass(WordCountReducer.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);
} }
```

8. **Export project into jar file**

   → Right click on **"Project"** and click **"export"**
   → Choose the desired path and save

To Run the Project using command line interface do the following steps:

1. **Start Hadoop**
   start-all.sh
2. **Create a Directory**
   hdfs dfs -mkdir -p test
3. **Insert input file into the directory**
   hdfs dfs -put /home/veeranna/input.txt test/

```
apple apple apple apple apple
bat bat bat bat
corn corn corn
dog dog
elephant
```

input.txt

4. **Mapreduce command for wordcount**
   hadoop jar /home/veeranna/eclipse-workspace/Demo/src/org/myorg/Demo/wordcount.jar org.myorg.Demo.WordCount test/input.txt test/output
5. **List the elements in directory**
   hdfs dfs -ls test/output
6. **Show the result**
   hdfs dfs -cat test/output/part-r-00000
7. **Stop Hadoop**
   stop-all.sh

OUTPUT

```
veeranna@veeranna-VirtualBox:~$ hdfs dfs -ls test/output/
Found 2 items
-rw-r--r--   1 veeranna supergroup          0 2023-10-12 19:44 test/output/_SUCCESS
-rw-r--r--   1 veeranna supergroup         38 2023-10-12 19:44 test/output/part-r-00000
veeranna@veeranna-VirtualBox:~$ hdfs dfs -cat test/output/part-r-00000
apple   5
bat     4
corn    3
dog     2
elephant        1
veeranna@veeranna-VirtualBox:~$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as veeranna in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [veeranna-VirtualBox]
Stopping nodemanagers
localhost: WARNING: nodemanager did not stop gracefully after 5 seconds: Trying to kill with kill -9
Stopping resourcemanager
```

4. Implementation of MR Program that processes a Weather Dataset.

Steps to run Hadoop MR Program:

1. **Launch Eclipse and set the Eclipse Workspace.**
2. **create Project,** click on File→ New→Java Project.
   Note: Choose "JavaSE-1.8" while creating the project
3. ***Create a new Package,* right-click on the Project Name→New→Package.**
   ➔ **Provide the package name: org.myorg**
4. **Add the Hadoop libraries (jars).**
   ➔ Right-Click on Project Name → Build Path → configure Build Path.
   ➔ Add the External jars.
   ➔ go to hadoop-3.2.4 → share → hadoop.
   6) Add the client jar files.
   7) Add common jar files.
   8) Add yarn jar files.
   9) Add MapReduce jar files.
   10) Add HDFS jar files.

   Click Open and apply.
5. **Create a new class,** provide class name as **"MaxTemperatureMapper"**

   ➔ **MaxTemperatureMapper.java**

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{
	Text k= new Text();
	@Override
	public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException
	{
		String line = value.toString();
		StringTokenizer tokenizer = new StringTokenizer(line," ");

		while (tokenizer.hasMoreTokens())
		{
			String year= tokenizer.nextToken();
			k.set(year);
			String temp= tokenizer.nextToken().trim();
			int v = Integer.parseInt(temp);
			context.write(k,new IntWritable(v));
		}
	}
}
```

6. Create another class that performs the reduce job

➔ MaxTemperatureReducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
	@Override
	public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException
	{
		int maxtemp=0;
		for(IntWritable it : values)
		{
			int temperature= it.get();
			if(maxtemp<temperature)
			{
				maxtemp =temperature;
			}
		}
		context.write(key, new IntWritable(maxtemp));
	}
}
```

7. create the driver class, which contains the main method.

➔ MaxTemperature.java

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

public class MaxTemperature
{
	public static void main(String[] args) throws Exception
	{
		Configuration conf = new Configuration();
		Job job = Job.getInstance(conf, "Max Temperature");
		job.setJarByClass(MaxTemperature.class);
		job.setMapperClass(MaxTemperatureMapper.class);
		job.setCombinerClass(MaxTemperatureReducer.class);
		job.setReducerClass(MaxTemperatureReducer.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);
        }
}
```

8. **Export project into jar file**

   → Right click on **"Project"** and click **"export"**
   → Choose the desired path and save

To Run the Project using command line interface do the following steps:

1. **Start Hadoop**
   start-all.sh
2. **Create a Directory**
   hdfs dfs -mkdir -p test
3. **Insert input file into the directory**
   hdfs dfs -put /home/veeranna/Temperature.txt test/

```
1900 39
1900 14
1900 5
1900 11
1900 20
1900 20
1900 22
1900 15
1900 41
1900 42
1900 46
1900 6
1900 13
1900 13
1900 30
1900 45
1900 13
```

Temperature.txt

4. **Mapreduce command for wordcount**
   hadoop jar /home/veeranna/eclipse-workspace/Demo/src/org/myorg/Demo/weather.jar org.myorg.Demo.MaxTemperature test/input.txt test/output

5. **List the elements in directory**
   hdfs dfs -ls test/output
6. **Show the result**
   hdfs dfs -cat test/output/part-r-00000
7. **Stop Hadoop**
   stop-all.sh

OUTPUT

```
veeranna@veeranna:~$ hdfs dfs -ls test/output
Found 2 items
-rw-r--r--   1 veeranna supergroup          0 2023-11-13 11:53 test/output/_SUCCESS
-rw-r--r--   1 veeranna supergroup        912 2023-11-13 11:53 test/output/part-r-00000
veeranna@veeranna:~$ hdfs dfs -cat test/output/part-r-00000
1900    46
1901    48
1902    49
1903    35
1904    46
1905    35
1906    32
1907    49
1908    44
1909    38
1910    47
1911    48
1912    44
1913    43
1914    49
1915    49
```

5. Pig Installation

Steps to run install Pig:

1. **Download pig tar file**
   → wget https://dlcdn.apache.org/pig/latest/pig-0.17.0.tar.gz

```
veeranna@veeranna-VirtualBox:~$ wget https://dlcdn.apache.org/pig/latest/pig-0.17.0.tar.gz
--2023-11-13 18:54:09--  https://dlcdn.apache.org/pig/latest/pig-0.17.0.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 230606579 (220M) [application/x-gzip]
Saving to: ‘pig-0.17.0.tar.gz’

pig-0.17.0.tar.gz               100%[=====================================================>] 219.92M  23.8MB/s    in 8.7s

2023-11-13 18:54:18 (25.4 MB/s) - ‘pig-0.17.0.tar.gz’ saved [230606579/230606579]
```

2. **Extract the pig tar file**
   → tar -xvf pig-0.17.0.tar.gz

3. **Add JAVA_HOME and pig paths**
   → gedit .bashrc

   #java
   export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
   export PATH=$PATH:JAVA_HOME/bin

   #pig
   export PIG_HOME=$HOME/pig-0.17.0
   export PATH=$PATH:$PIG_HOME/bin

4. **start all the daemons**
   → start-all.sh

5. **start pig**
   → pig

```
veeranna@veeranna-VirtualBox:~$ pig
2023-11-13 19:13:49,454 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2023-11-13 19:13:49,456 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2023-11-13 19:13:49,456 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2023-11-13 19:13:49,525 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2023-11-13 19:13:49,525 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/veeranna/pig_1699883029513.log
2023-11-13 19:13:49,568 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/veeranna/.pigbootup not found
2023-11-13 19:13:49,855 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead
, use mapreduce.jobtracker.address
2023-11-13 19:13:49,855 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file s
ystem at: hdfs://localhost:9000
2023-11-13 19:13:50,474 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-71981154-d95d-4754-a4b
f-73d41ddabc18
2023-11-13 19:13:50,474 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt>
```