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

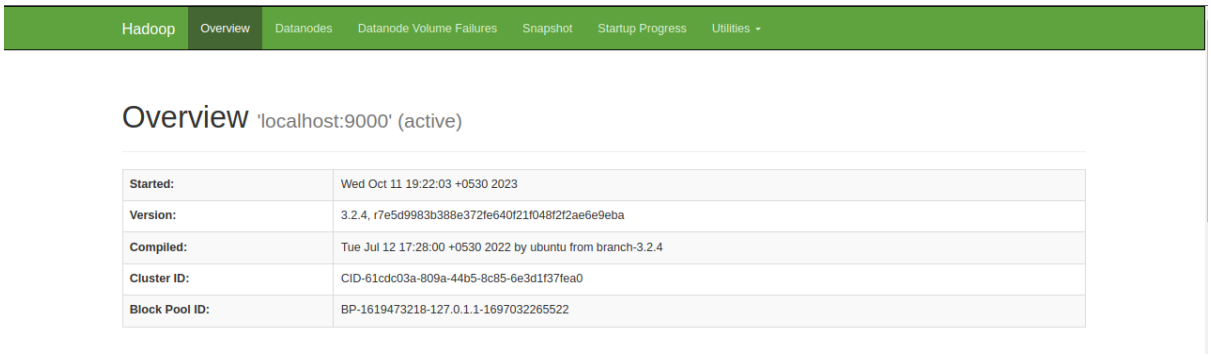
export PDSH_RCMD_TYPE=ssh

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 weather dataset**
   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>
```

grunt> A = load 'passwd' using PigStorage(':');
grunt> B = foreach A generate $0 as id;
grunt> dump B;

grunt> A =LOAD'student' AS (name:chararray, age:int, gpa:float);
grunt> DUMPA;

OUTPUT

(John,18,4.0F)
(Mary,19,3.7F)
(Bill,20,3.9F)
(Joe,22,3.8F)
(Jill,20,4.0F)

grunt> B =FILTER A BYnamematches 'J.+';
grunt> DUMPB;

OUTPUT

(John,18,4.0F)
(Joe,22,3.8F)
(Jill,20,4.0F

6. Create Virtual machines using Open-source software: VM Ware/ Oracle Virtual Box.

To create a virtual machine using Oracle VirtualBox, follow these steps:

1. ***Download and Install VirtualBox***

➔ Visit the Oracle VirtualBox website (https://www.virtualbox.org/) and download the latest version of VirtualBox for your operating system.

➔ Run the installer and follow the on-screen instructions to install VirtualBox.

2. ***Download an Operating System ISO***

➔ Obtain the ISO image of the operating system you want to install on the virtual machine. You can download Linux distributions or other OS ISOs from their respective official websites.

3. ***Open VirtualBox and create a new virtual machine***

➔ Launch Oracle VirtualBox after installation.

➔ Click on the "New" button in the VirtualBox Manager window to start creating a new virtual machine.

4. ***Name and Operating System***

➔ In the "Name and Operating System" window:

➔ Enter a name for your virtual machine.

➔ Select the type of operating system you are installing (e.g., Linux, Windows, macOS).

5. ***Memory (RAM) Allocation***

➔ Allocate memory (RAM) to your virtual machine. Choose an amount that suits your requirements but doesn't exceed the available physical RAM on your host system.

6. ***Hard Disk Creation and file type***

➔ Choose the option to "Create a virtual hard disk now" and click "Create."

➔ Select the file type for the virtual hard disk (usually VDI or VMDK) and click "Next."

7. ***File Location and Size***

➔ Specify the location where you want to store the virtual hard disk file and set the size of the disk. Ensure you allocate enough space for your OS and applications.

8. ***Create Virtual Machine***

➔ Review your settings in the summary window and click "Create" to create the virtual machine.

9. ***Attach ISO File***

➔ In the VirtualBox Manager, select your virtual machine.

➔ Click on "Settings" and go to the "Storage" section, Browse and select the OS ISO file you downloaded.

10. ***Start the Virtual Machine and Enjoy using it***

➔ click the "Start" button to power it on. Follow the on-screen instructions to install the operating system on your virtual machine.

➔ Once the OS is installed and configured, you can use your virtual machine just like a physical computer.

7. Use Amazon EC2 to create a Virtual machine.

Here are the steps to create and use an EC2 instance:

1. ***Sign in to the AWS Management Console***

➔ Go to the AWS Management Console (https://aws.amazon.com/).
➔ Sign in with your AWS account credentials.

2. ***Open the EC2 Dashboard and Launch EC2 Instance***

➔ Once signed in, select "Services" at the top left corner of the console.
➔ Under "Compute," select "EC2" to open the EC2 Dashboard.
➔ In the EC2 Dashboard, click the "Launch Instance" button to start the instance creation process.

3. ***Choose an Amazon Machine Image (AMI)***

➔ Select an AMI based on your requirements (e.g., Ubuntu, Windows Server).
➔ Choose the appropriate AMI for your use case, and click "Select."

4. ***Choose an Instance Type***

➔ Select the instance type that suits your workload. Instance types vary in terms of CPU, memory, and other resources.
➔ Click "Next: Configure Instance Details" when ready.
➔ Modify settings as needed and click "Next: Add Storage."

5. ***Add Storage***

➔ Specify the storage (EBS volumes) for your EC2 instance.
➔ Configure the size and type of the root volume and Click "Next: Add Tags" when done.

6. ***Configure Security Group***

➔ Create a new security group or select an existing one.
➔ Configure inbound and outbound rules to control traffic to and from your instance.

7. ***Review and launch***

➔ Review all the settings you've configured for your EC2 instance. Click “Launch”

8. ***create a Key Pair***

➔ If you haven't created an EC2 key pair before, you'll be prompted to create one.
➔ Download the private key (.pem) file and store it in a secure location. You'll need this key to access your instance securely.

9. ***Launch the Instance and Connect to EC2 Instance***

➔ After creating or selecting a key pair, click the "Launch Instances" button.
➔ Once the instance is running, you can connect to it using SSH (for Linux instances) or RDP (for Windows instances) with the private key.

10. ***Start Using Your EC2 Instance***

➔ You can now use your EC2 instance for various tasks, such as hosting a website, running applications, or performing data analysis.

8. Use Amazon S3 to create bucket and upload objects.

To use Amazon Simple Storage Service (Amazon S3) to create a bucket and upload objects, follow these steps:

1. ***Sign in to AWS and open S3 Dashboard***

➔ Sign in to the AWS Management Console using your AWS account credentials.
➔ From the AWS Management Console, navigate to the S3 dashboard.

2. ***Create a Bucket***

➔ Enter a globally unique name for your bucket (S3 bucket names must be unique across all AWS accounts).
➔ Choose the AWS region where you want to create the bucket.
➔ Configure optional settings like versioning, logging, and tags.
➔ Click "Create" to create the bucket.

3. ***Upload Objects***

➔ In the bucket you just created, click the "Upload" button to upload objects.
➔ Click "Add files" or "Add folder" to select the files or folders you want to upload.
➔ Configure settings such as permissions and metadata for the uploaded objects.
➔ Click "Upload" to start the upload process.

4. ***Manage Objects***

➔ After uploading objects, you can manage them within the S3 bucket.
➔ You can set permissions, configure lifecycle policies, and organize objects into folders (known as "prefixes" in S3).

5. ***Access Objects***

➔ To access objects in your S3 bucket, click on the object's name in the S3 dashboard.
➔ You'll see a URL that you can use to access the object via a web browser or programmatically through APIs.

6. ***Set Bucket Policies***

➔ If you want to control access to your bucket and objects further, you can configure bucket policies.
➔ Bucket policies allow you to define fine-grained permissions for different users or applications.

7. ***Configure Cross-Region Replication***

➔ If you need to replicate your data to another AWS region for redundancy or compliance purposes, you can set up cross-region replication.

8. ***Monitor and Manage***

➔ Regularly monitor your S3 bucket for usage, billing, and access patterns.
➔ Use AWS CloudWatch and other AWS services for monitoring and alerting.

9. Install the Simple Notification Service on Ubuntu.

1. **Set Up AWS Account:**
   If you don't have an AWS account, sign up for one at https://aws.amazon.com/.
   Log in to the AWS Management Console.

2. **Access the SNS Console:**
   In the AWS Management Console, navigate to the Simple Notification Service (SNS) section.

3. **Create a Topic:**
   Click on "Create topic" in the SNS console.
   Provide a name and display name for your topic.

4. **Subscribe to the Topic:**
   After creating a topic, you need to subscribe to it to receive notifications.
   Click on the topic you just created and then click on "Create subscription."
   Choose the protocol for your subscription (e.g., email, SMS) and provide the necessary details.

5. **Confirm Subscription:**
   For certain protocols (e.g., email), you may need to confirm the subscription. Follow the instructions sent to your chosen endpoint.

6. **Configure Access Control:**
   Optionally, you can set up access control policies to control who can publish or subscribe to your SNS topic.

7. **Publish a Message (Optional):**
   Test your setup by publishing a message to your topic. In the AWS SNS console, select your topic and click "Publish message."

8. **Integrate SNS into Your Application (Optional):**
   If you're using SNS in a real application, you'll need to integrate it into your code. AWS provides SDKs for various programming languages.

Use Amazon Cloud front to create Distribution and Use Amazon Route53 to create a domain (example: .com, .in).

Amazon CloudFront Distribution:

1. **Sign in to AWS Console:**
   Log in to the AWS Management Console.

2. **Navigate to CloudFront:**
   In the AWS Management Console, go to the CloudFront service.

3. **Create a Distribution:**
   Click on "Create Distribution."
   Choose the delivery method: Web or RTMP. For a standard website, choose "Web."
   Click "Get Started" under the Web section.

4. **Configure Origin Settings:**
   Under "Origin Domain Name," select the source of your content (e.g., an S3 bucket).
   Configure other settings like Origin Path, Origin ID, and Protocol Policy.

5. **Configure Default Cache Behavior:**
   Configure settings such as Viewer Protocol Policy, Allowed HTTP Methods, and Forward Headers.
   Set up caching behavior according to your requirements.

6. **Configure Distribution Settings:**
   Enter a unique comment (optional).
   Configure other settings like Price Class, AWS WAF WebACL (optional), and Logging (optional).

7. **Add Alternate Domain Names (CNAMEs):**
   In the "Alternate Domain Names (CNAMEs)" field, add the domain names (e.g., example.com, www.example.com) that you want to associate with your distribution.

8. **Configure SSL Certificate:**
   Under "SSL Certificate," choose the SSL/TLS certificate for your domain (e.g., use the default CloudFront certificate or upload a custom one).

9. **Review and Create:**
   Review your settings and click "Create Distribution."

10. **Wait for Deployment:**
    It may take some time for the distribution to be deployed. Once deployed, your CloudFront distribution will have a unique domain name (e.g., d12345abcde.cloudfront.net).

Amazon Route 53 Domain:

1. **Navigate to Route 53:**
   In the AWS Management Console, go to the Route 53 service.

2. **Register a Domain:**
   Click on "Domain registration" and register a new domain (e.g., example.com).
   Follow the on-screen instructions to complete the registration process.

3. **Create a Hosted Zone:**
   In the Route 53 dashboard, click on "Hosted zones" and then "Create Hosted Zone."
   Enter your domain name (e.g., example.com) and click "Create."

4. **Add Record Sets:**
   Inside your hosted zone, click on "Create Record Set."
   Add a record set for your domain (e.g., www.example.com) and associate it with the CloudFront distribution domain name.

5. **Update Name Servers:**
   After creating the hosted zone, note the provided name servers.
   Update the name servers at your domain registrar with the ones provided by Route 53.

6. **Wait for DNS Propagation:**
   It may take some time for DNS changes to propagate. Once propagated, your domain should point to your CloudFront distribution.

Now, your domain is associated with your CloudFront distribution, and users can access your content using the custom domain name.

10. Study and Implement Cloud Security management by VPC.

1. **Define VPC Requirements:**
   Outline network needs, considering IP address ranges, subnets, and availability zones.

2. **Create VPC and Subnets:**
   In AWS Console, set up a VPC with public and private subnets based on your design.

3. **Configure Route Tables:**
   Define routes for subnets, ensuring public subnets have internet access.

4. **Implement Security Groups:**
   Create security groups for instances, allowing only necessary traffic.

5. **Set Up NACLs:**
   Configure Network Access Control Lists for additional subnet-level security.

6. **Establish VPC Peering (if needed):**
   Connect multiple VPCs securely with VPC peering.

7. **Enable VPN Connections (if needed):**
   Create VPN connections for secure communication with on-premises networks.

8. **Deploy Bastion Hosts:**
   Use bastion hosts in public subnets for secure administrative access.

9. **Turn On VPC Flow Logs:**
   Capture IP traffic data with VPC Flow Logs for analysis.

10. **Implement Data Encryption:**
    Enable encryption for data in transit and at rest using HTTPS, SSL/TLS, and services like Amazon RDS.

11. **Monitor with CloudWatch:**
    Utilize CloudWatch for VPC monitoring, including metrics, logs, and alarms.

12. **Regularly Audit and Educate:**
    Conduct periodic security audits, update configurations, and educate your team on AWS security best practices.

11. Building a “Hello world” app for the cloud by using AWS Lambda.

1. **Create an AWS Account:**
   If you don't have an AWS account, sign up for one at https://aws.amazon.com/.

2. **Access AWS Lambda Console:**
   Log in to the AWS Management Console and navigate to the Lambda service.

3. **Create a New Lambda Function:**
   Click on the "Create function" button.
   Choose "Author from scratch."
   Provide a name for your function (e.g., HelloWorldFunction).
   Choose a runtime (e.g., Node.js).

4. **Configure Execution Role:**
   Create a new role with basic Lambda permissions.
   Click on "Create a new role with basic Lambda permissions" in the Execution role section.
   Provide a name for the role and click "Create function."

5. **Write Lambda Function Code:**
   In the Function code section, replace the default code with a simple "Hello, World!"

6. **Configure Basic Settings:**
   Scroll down to the Basic settings section.
   Set the timeout according to your function's expected execution time (e.g., 5 seconds).

7. **Create an API Gateway (Optional):**
   To expose your Lambda function as an HTTP endpoint, you can create an API Gateway.
   In the Lambda function designer, click on "Add trigger" and select API Gateway.
   Configure the API Gateway as needed.

8. **Test Your Lambda Function:**
   Save your Lambda function.
   In the Lambda function designer, click on "Test."
   Create a new test event or use a sample event.
   Click "Test" to execute your function and verify the output.

9. **Invoke Your Lambda Function (Optional):**
   If you created an API Gateway, you can now invoke your Lambda function through the provided endpoint.

10. **View Logs (Optional):**
    In the Monitoring tab, check CloudWatch Logs to view execution logs for your Lambda function.

11. Explore Advanced Features:
As you become familiar with AWS Lambda, explore additional features like environment variables, layers, and integrating with other AWS services.

---

12. Installing and configuring python/java/PHP platform by using Google App Engine.

1. Install Google Cloud SDK:
   Download and install the [Google Cloud SDK](https://cloud.google.com/sdk/docs/install).

2. Python Specific Steps:
   Create a Python App:
   Write a simple Python web application (e.g., using Flask or Django).

3. Configure `app.yaml`:
   Create an `app.yaml` file in your project directory to configure settings such as runtime and entrypoint.

4. Deploy Python App:
   Run `gcloud app deploy` to deploy your Python application to Google App Engine.

5. Java Specific Steps:
   Create a Java App:
   Write a simple Java web application (e.g., using Spring Boot).

6. Configure `app.yaml`:
   Create an `app.yaml` file in your project directory to configure settings such as runtime and entrypoint.

7. Deploy Java App:
   Run `gcloud app deploy` to deploy your Java application to Google App Engine.

8. PHP Specific Steps:
   Create a PHP App:
   Write a simple PHP web application.

9. Configure `app.yaml`:
   Create an `app.yaml` file in your project directory to configure settings such as runtime and entrypoint.

10. Deploy PHP App:
    Run `gcloud app deploy` to deploy your PHP application to Google App Engine.

11. Access the Apps:
    After deployment, your apps will be accessible via the assigned App Engine URLs.