

Ex. No 1

LAUNCHING VIRTUALBOX USING TRYSTACK

5/10/21

AIM:

To Launch Virtual Box using Try stack

PROCEDURE:

STEP 0: Prepare the environment and Update

\$ sudo apt-get update -y

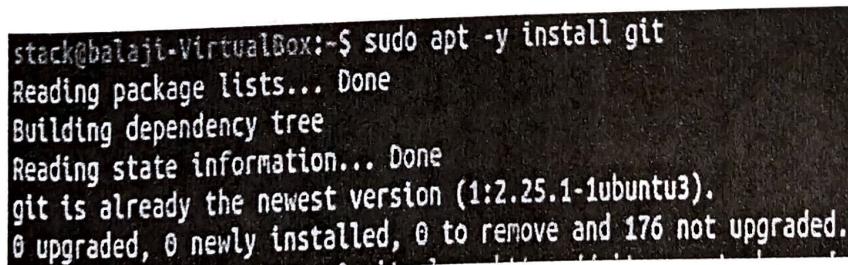


```
stack@balaji-VirtualBox:~$ sudo apt-get update -y
Get:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [168 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [319 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [55 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [295 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [144 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [78 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Meta-data [208 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Meta-data [24.3 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [93.8 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 C-n-f Meta-data [9,988 B]
Get:15 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [250 kB]
Get:16 http://security.ubuntu.com/ubuntu focal-security/universe i386 Packages [50.3 kB]
```

STEP1: Find your IP address in Linux Server ~\$ ipaddr show

STEP2: Install Git Package to access Git-Repository

~\$ sudo apt -y install git



```
stack@balaji-VirtualBox:~$ sudo apt -y install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3).
0 upgraded, 0 newly installed, 0 to remove and 176 not upgraded.
```

```
stack@balaji-VirtualBox:~/devstack
```

```
admin_password=$ADMIN_PASSWORD  
database_password=$ADMIN_PASSWORD  
rabbit_password=$ADMIN_PASSWORD  
service_password=$ADMIN_PASSWORD  
host_ip=10.0.4.15
```

git clone https://github.com/openstack/devstack

STEP 3: GitCLone Command Make copy of existing Repo

\$ git clone https://git.openstack.org/openstack-dev/devstack

```
stack@balaji-VirtualBox:~$ git clone https://git.openstack.org/openstack-dev/devstack  
Cloning into 'devstack'...  
warning: redirecting to https://opendev.org/openstack/devstack/  
remote: Enumerating objects: 46226, done.  
remote: Counting objects: 100% (46226/46226), done.  
remote: Compressing objects: 100% (21129/21129), done.  
remote: Total 46226 (delta 32674), reused 37512 (delta 24392)  
Receiving objects: 100% (46226/46226), 9.50 MiB | 417.00 KiB/s, done.  
Resolving deltas: 100% (32674/32674), done.
```

STEP4: Command to Change Directory To devstack

```
stack@balaji-VirtualBox:~$ cd devstack
```

STEP 5: Command to list files in Devstack

```
stack@balaji-VirtualBox:~/devstack$ ls
clean.sh          functions-common  local.conf      run_tests.sh  tools
CONTRIBUTING.rst  FUTURE.rst       MAINTAINERS.rst  samples     tox.ini
data              gate             Makefile       setup.cfg    unstack.sh
docs              HACKING.rst     openrc        setup.py
extras.d          lib              playbooks     stackrc
fixes            LICENSE         README.rst   stack.sh
functions        nano local.conf  roles        tests
```

STEP 6: Insert Set Of code in Local.conf

-nano local.conf

Ax
y

-nano is an easy to use command line text editor for Unix and Linux

STEP 7: Final Command to install OpenStack

```
stack@balaji-VirtualBox:~/devstack$ ./stack.sh
+ unset GREP_OPTIONS
+ unset LANG
+ unset LANGUAGE
+ LC_ALL=en_US.UTF-8
+ export LC_ALL
++ env
++ grep -E '^OS_'
++ cut -d= -f 1
+ unset ...
+ umask 022
+ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
/usr/local/games:/snap/bin:/usr/local/sbin:/usr/local/bin:/sbin
+++ dirname ./stack.sh
++ cd .
++ pwd
+ TOP_DIR=/home/stack/devstack
+ NOUNSET=
+ [[ -n '' ]]
++ date +xs
+ DEVSTACK_START_TIME=1602054267
+ [[ -r /home/stack/devstack/.stackenv ]]
+ FILES=/home/stack/devstack/files
+ '[' '!' -d /home/stack/devstack/files ']'
+ '[' '!' -d /home/stack/devstack/inc ']'
+ '[' '!' -d /home/stack/devstack/lib ']'
+ [[ '=' == \y ]]
+ [[ 2001 -eq 0 ]]
+ [[ -n '' ]]
```

RESULT

To Launch Virtual Box using Try tack is done and output is obtained

Ex. No 2

5\10\21

TO GAE LAUNCHER TO LAUNCH THE WEB APPLICATIONS.

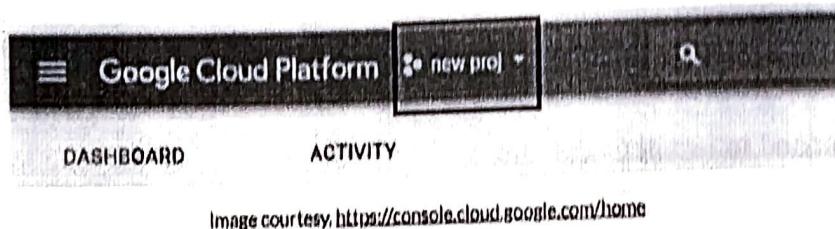
AIM:

To use GAE Launcher to launch web applications.

PROCEDURE:

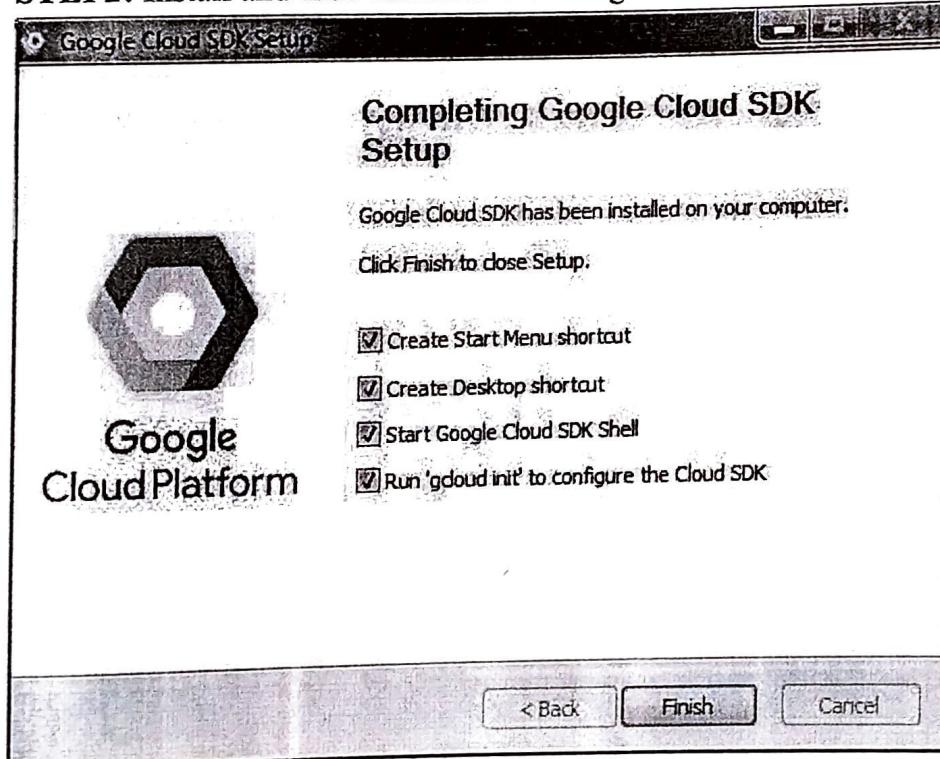
STEP0: visit <https://console.cloud.google.com/>

STEP1: Create a new Cloud Console project or retrieve the project ID of an existing project to use: Go to the Project page





STEP2: Install and then initialize the Google Cloud SDK Download SDK



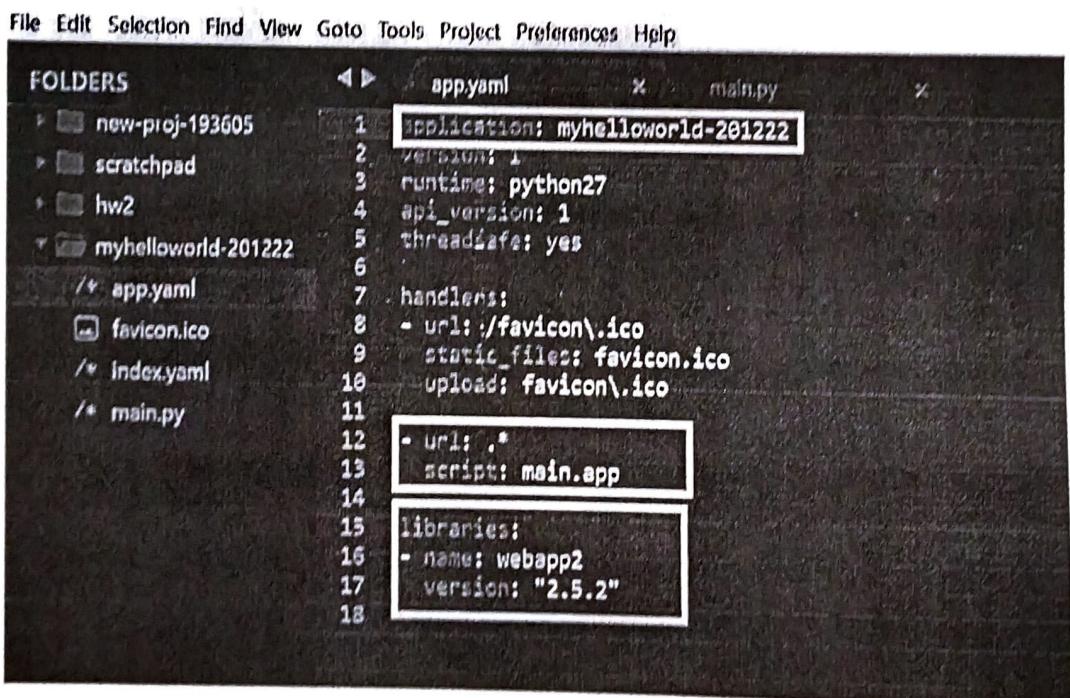
STEP3: Creating a website to host on Google App Engine

Basic structure for the project

This guide uses the following structure for the project:

- app.yaml: Configure the settings of your App Engine application.
- www/: Directory to store all of your static files, such as HTML, CSS, images, and JavaScript.
- css/: Directory to store stylesheets.
- style.css: Basic stylesheet that formats the look and feel of your site.
- Images/: Optional directory to store images.
- index.html: An HTML file that displays content for your website.
- js/: Optional directory to store JavaScript files.
- Other asset directories.

STEP 4: Creating the app.yaml file



The screenshot shows a code editor with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a toolbar with FOLDERS, app.yaml, and main.py buttons. The left pane shows a file tree with folders new-proj-193605, scratchpad, hw2, and myhelloworld-201222, which contains app.yaml, favicon.ico, index.yaml, and main.py. The right pane shows the contents of the app.yaml file:

```
application: myhelloworld-201222
version: 1
runtime: python27
api_version: 1
threadsafe: yes
handlers:
- url: /*
  script: main.app
libraries:
- name: webapp2
  version: "2.5.2"
```

- 1.Create a directory that has the same name as your project ID. You can find your project ID in the Console.
- 2.In directory that you just created, create a file named app.yaml.
- 3.Edit the app.yaml file and add the following code to the file:

```

4.runtime: python27
api_version: 1
threadsafe: true
handlers:
- url: /
  static_files: www/index.html upload: www/index.html

- url: /(.*) static_files: www\1 upload: www/(.*)

```

STEP5: Creating the index.html file

The screenshot shows a file explorer on the left and a code editor on the right.

File Explorer (Left):

- FOLDERS
 - node_modules
 - src
 - app
 - assets
 - environments
 - browserslist
 - favicon.ico
 - index.html
 - tsconfig.json
 - main.ts
 - polyfills.ts
 - styles.css
 - test.ts
- tsconfig.app.json
- tsconfig.spec.json
- tslib.json
- es2015only
- codegen.json
- angular.json
- package-lock.json
- package.json
- README.md
- tsconfig.json
- tslint.json

Code Editor (Right):

```

1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>HelloWorld</title>
6   <a href="/">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
8     <link rel="icon" type="image/x-icon" href="favicon.ico">
9   </head>
10  <body>
11    <ng-root></ng-root>
12  </body>
13 </html>
14 </!DOCTYPE>
15

```

```

<html>
<head>
<title>Hello, world!</title>
<link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
<h1>Hello, world!</h1>
<p>
This is a simple static HTML file that will be served from Google App Engine.

```

```
</p>
</body>
</html>
```

STEP 6: Deploying your application to App Engine

```
(env) pavanraonavule@cloudshell:~/fastapi (fastapi-tutlinks-demo)$ gcloud app deploy app.yaml
Services to deploy:
descriptor:    [/home/pavanraonavule/fastapi/app.yaml]
source:        [/home/pavanraonavule/fastapi]
target project: [fastapi-tutlinks-demo]
target service: [default]
target version: [20290326t204745]
target url:    [https://fastapi-tutlinks-demo.appspot.com]

Do you want to continue (Y/n)? Y
Beginning deployment of service [default]...
Created .gcldignore file. See 'gcloud gcldignore' for details.
[= Uploading 1029 files to Google Cloud Storage =]

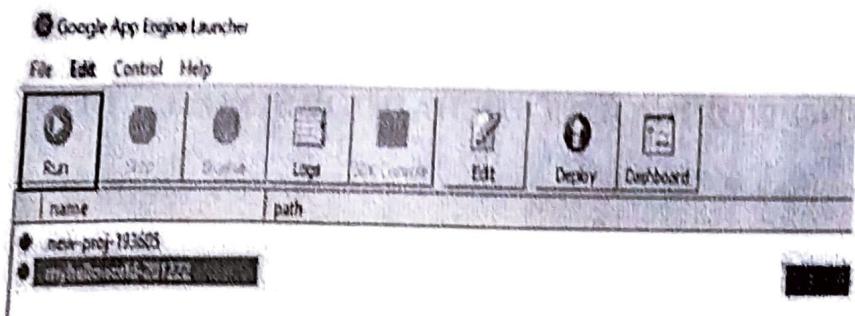
File upload done.
Updating service [default]...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://fastapi-tutlinks-demo.appspot.com]

You can stream logs from the command line by running:
$ gcloud app logs tail -n default

To view your application in the web browser run:
$ gcloud app browse
(env) pavanraonavule@cloudshell:~/fastapi (fastapi-tutlinks-demo)$
```

STEP 7:Once the shell is open, enter the following commands:

1. gcloud app create
2. git clone https://github.com/GoogleCloudPlatform/python-docs-samples
3. cd python-docs-samples/appengine/standard_python3/hello_world
4. virtualenv --python python3 ~/envs/hello_world
5. source ~/envs/hello_world/bin/activate
6. pip install -r requirements.txt
7. python main.py



STEP 8: Then click “web preview” and select “preview on port 8080.

OUTPUT



RESULT

Thus the procedure to GAE Launcher to launch web applications is done and output is obtained

5/10/21

Ex.No.3 RUNNING WORDCOUNT ON HADOOP SINGLE NODE CLUSTER

AIM:

To Run wordcount on hadoop single node cluster

PROCEDURE:

STEPS:

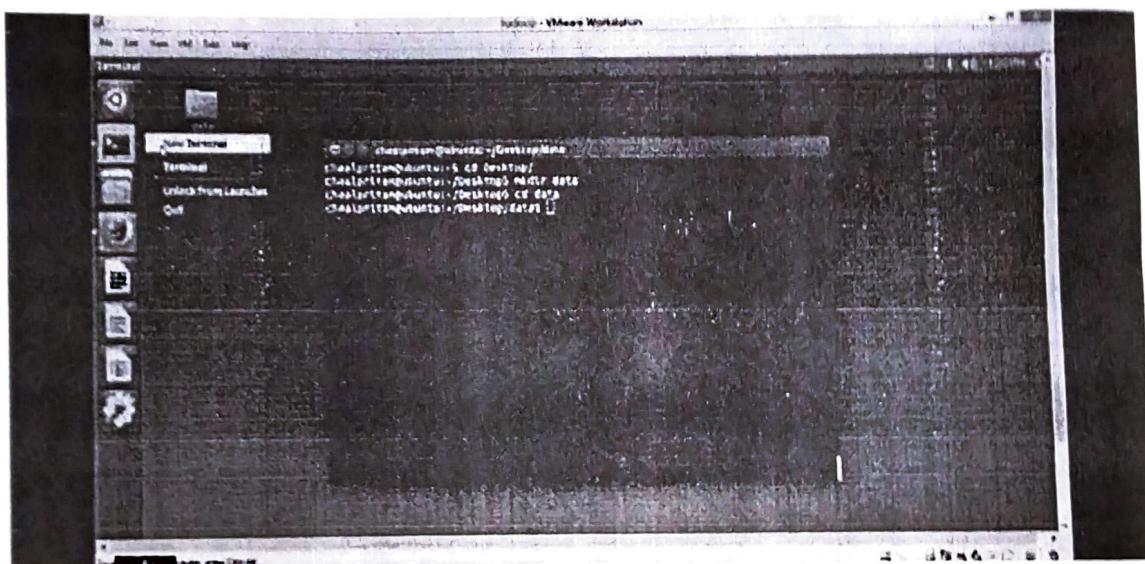
Step 1: Creating a working directory for your data

```
user@ubuntu:~$ cd Desktop/
```

```
user@ubuntu:~/Desktop$ mkdir data
```

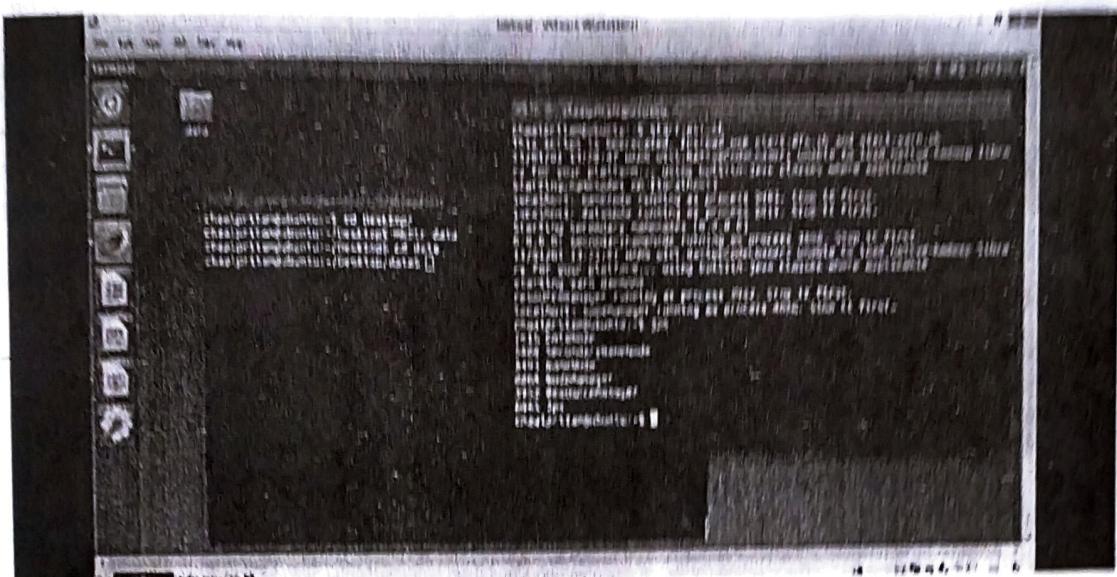
```
user@ubuntu:~/Desktop$ cd data
```

```
user@ubuntu:~/Desktop/data$
```



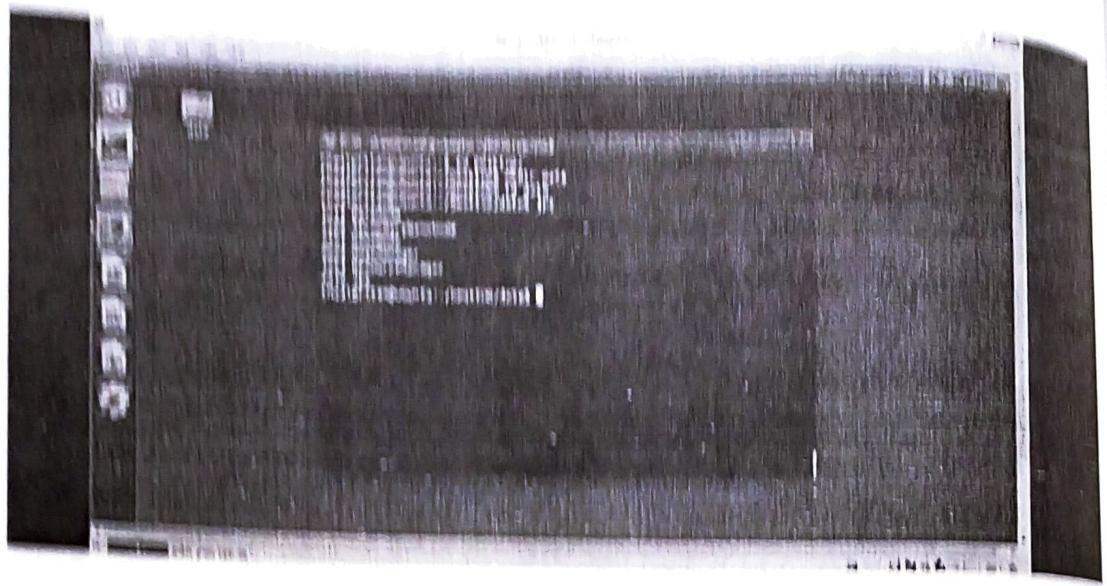
Step 2:

```
user@ubuntu:~ start-all.sh
```



Step 3:

```
user@ubuntu:~$ cd Desktop/  
user@ubuntu:~/Desktop$ mkdir data  
user@ubuntu:~/Desktop$ cd data  
user@ubuntu:~/Desktop/ data$ jps
```

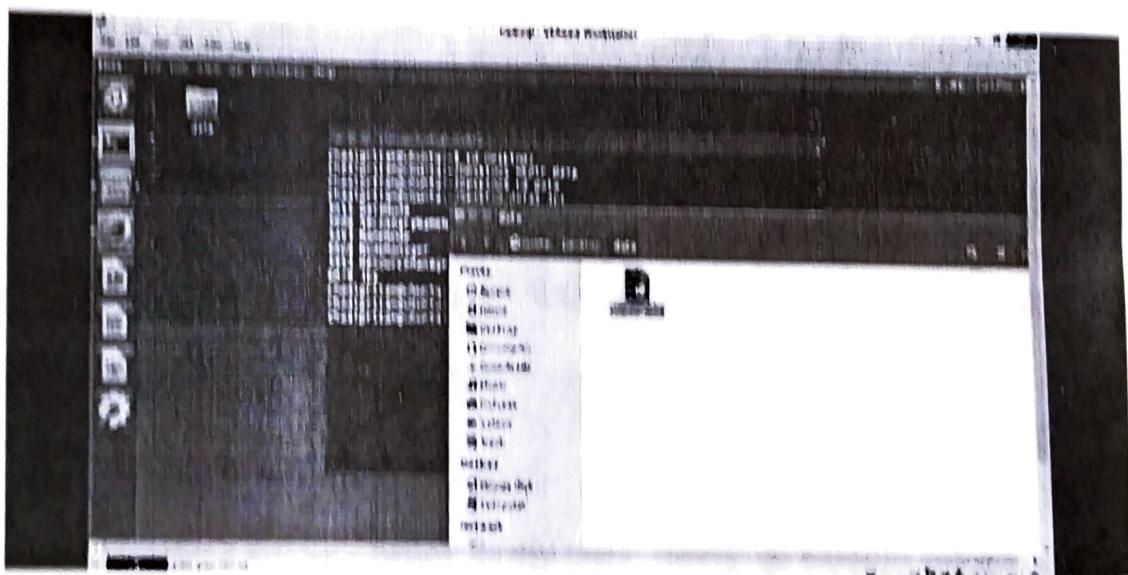


Step 4:

```
user@ubuntu:~/Desktop/data$ jps>> testing.txt
```

```
user@ubuntu:~/Desktop/data$ cd
```

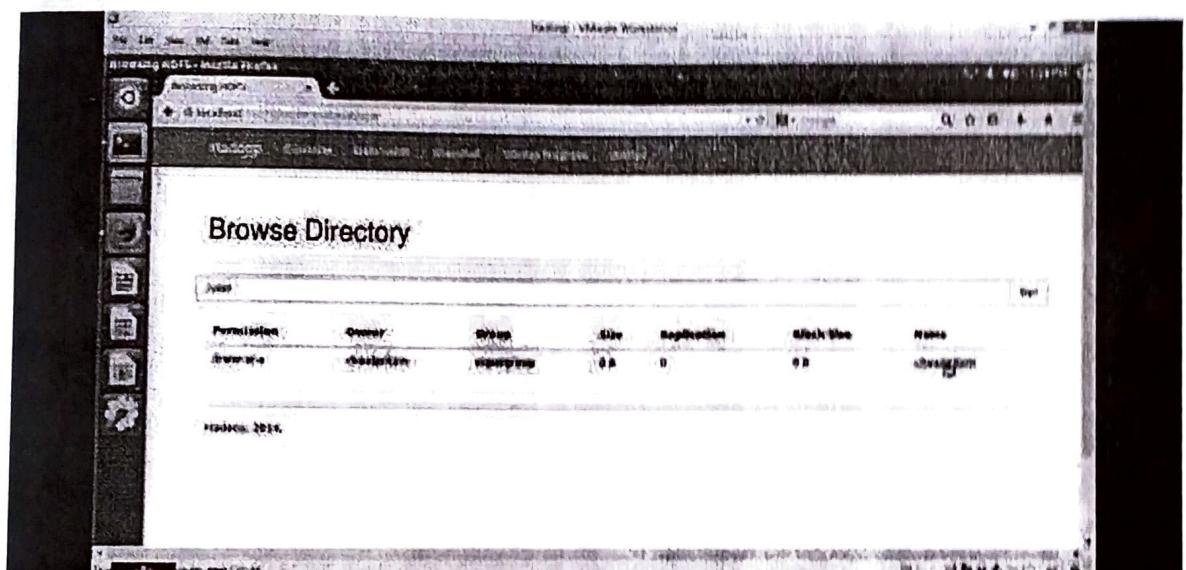
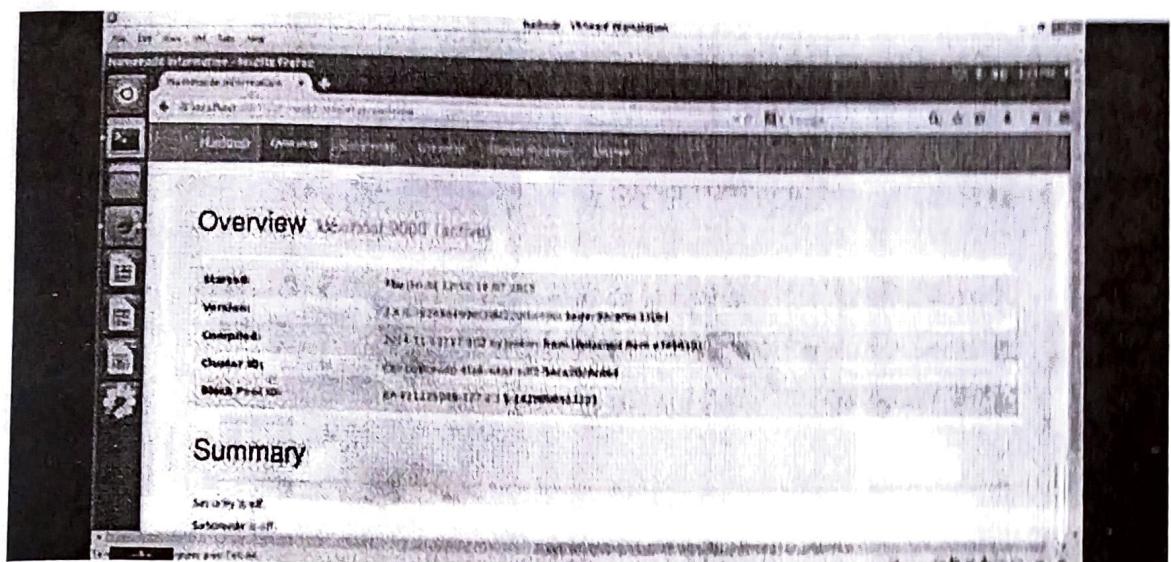
```
user@ubuntu:~$
```



Step 5:

```
user@ubuntu:~$ cd /user/local/hadoop/
```

```
user@ubuntu: /user /local /hadoop$ bin/hdfsdfs -mkdir /user
```

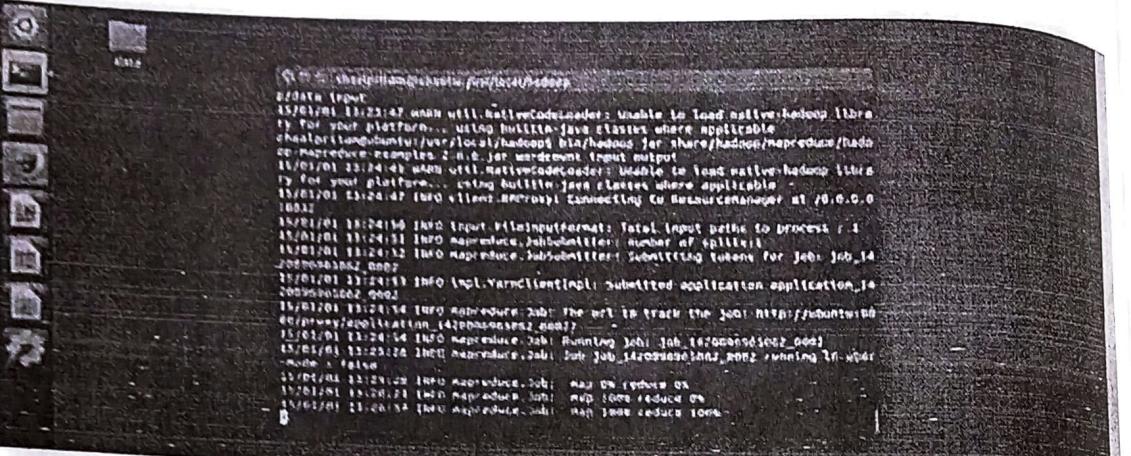


Step 6:

```
user@ubuntu: /user /local /hadoop$ bin/hdfsdfs -put /home /user  
/Desktop /data input
```

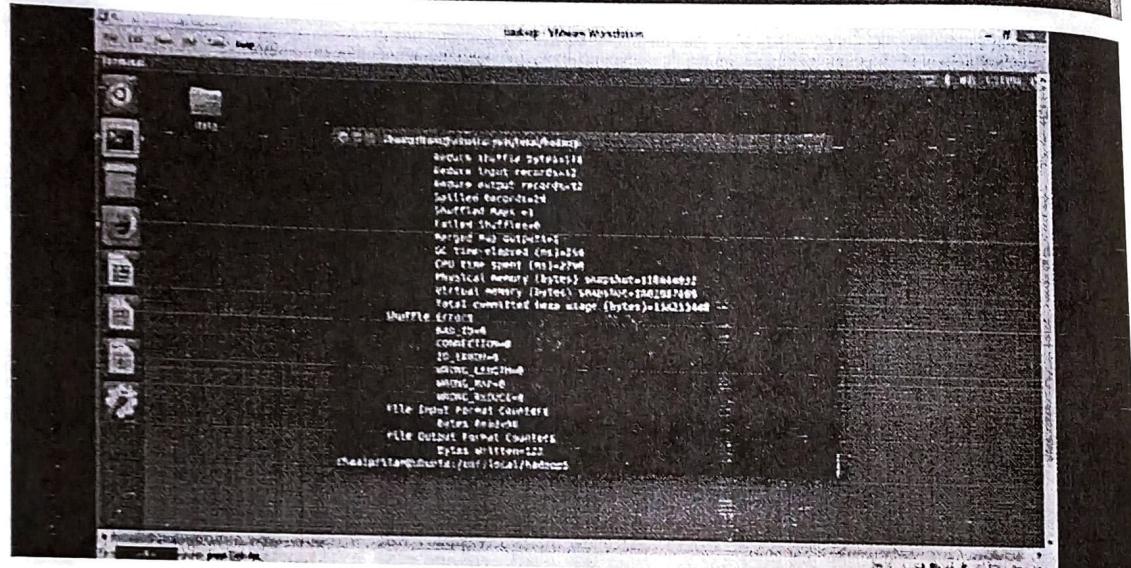
```
user@ubuntu: /user /local /hadoop$ bin/hadoop jar share/hadoop  
/mapreduce /hadoop-mapreduce-examples-2.6.0. jar wordcount input
```

OUTPUT



```
user@ubuntu:~/local/hadoop$ bin/hdfsdfs -cat output /*
```

```
15/01/01 13:23:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-Java classes where applicable
15/01/01 13:23:47 INFO mapreduce.JobSubmitter: Number of splits:2
15/01/01 13:23:47 INFO mapreduce.JobSubmitter: Hadoop version:2.0.0-alpha
15/01/01 13:23:47 INFO mapreduce.JobSubmitter: Using default input format: org.apache.hadoop.mapred.TextInputFormat
15/01/01 13:23:47 INFO mapreduce.JobSubmitter: Connecting to ResourceManager at /0.0.0.0:16030
15/01/01 13:23:47 INFO InputFormat: Total input paths to process : 1
15/01/01 13:23:47 INFO MapReduce: JobSubmission: number of splits:1
15/01/01 13:23:47 INFO MapReduce: Submitter: Submitting tokens for job: job_1420969450862_0002
15/01/01 13:23:51 INFO impl.YarnClientImpl: submitted application application_1420969450862_0002
15/01/01 13:23:54 INFO MapReduce: Job: The url to track the job: http://ubuntu:19888/jobs/1420969450862_0002? - 
15/01/01 13:23:54 INFO MapReduce: Job: Running job job_1420969450862_0002
15/01/01 13:23:54 INFO MapReduce: Job: Due job_1420969450862_0002 running in user mode: false
15/01/01 13:23:54 INFO MapReduce: Job: map 0% reduce 0%
15/01/01 13:23:54 INFO MapReduce: Job: map 100% reduce 100%
```

```
user@ubuntu:~/local/hadoop$ bin/hdfsdfs -cat output /*
```

```
15/01/01 13:23:54 INFO mapreduce.JobSubmitter: Number of splits:2
15/01/01 13:23:54 INFO mapreduce.JobSubmitter: Hadoop version:2.0.0-alpha
15/01/01 13:23:54 INFO mapreduce.JobSubmitter: Using default input format: org.apache.hadoop.mapred.TextInputFormat
15/01/01 13:23:54 INFO mapreduce.JobSubmitter: Connecting to ResourceManager at /0.0.0.0:16030
15/01/01 13:23:54 INFO InputFormat: Total input paths to process : 1
15/01/01 13:23:54 INFO MapReduce: JobSubmission: number of splits:1
15/01/01 13:23:54 INFO MapReduce: Submitter: Submitting tokens for job: job_1420969450862_0002
15/01/01 13:23:54 INFO impl.YarnClientImpl: submitted application application_1420969450862_0002
15/01/01 13:23:54 INFO MapReduce: Job: The url to track the job: http://ubuntu:19888/jobs/1420969450862_0002? - 
15/01/01 13:23:54 INFO MapReduce: Job: Running job job_1420969450862_0002
15/01/01 13:23:54 INFO MapReduce: Job: Due job_1420969450862_0002 running in user mode: false
15/01/01 13:23:54 INFO MapReduce: Job: map 0% reduce 0%
15/01/01 13:23:54 INFO MapReduce: Job: map 100% reduce 100%
```

Step 7:

```
user@ubuntu:~/local/hadoop$ bin/hdfsdfs -cat output /*
```

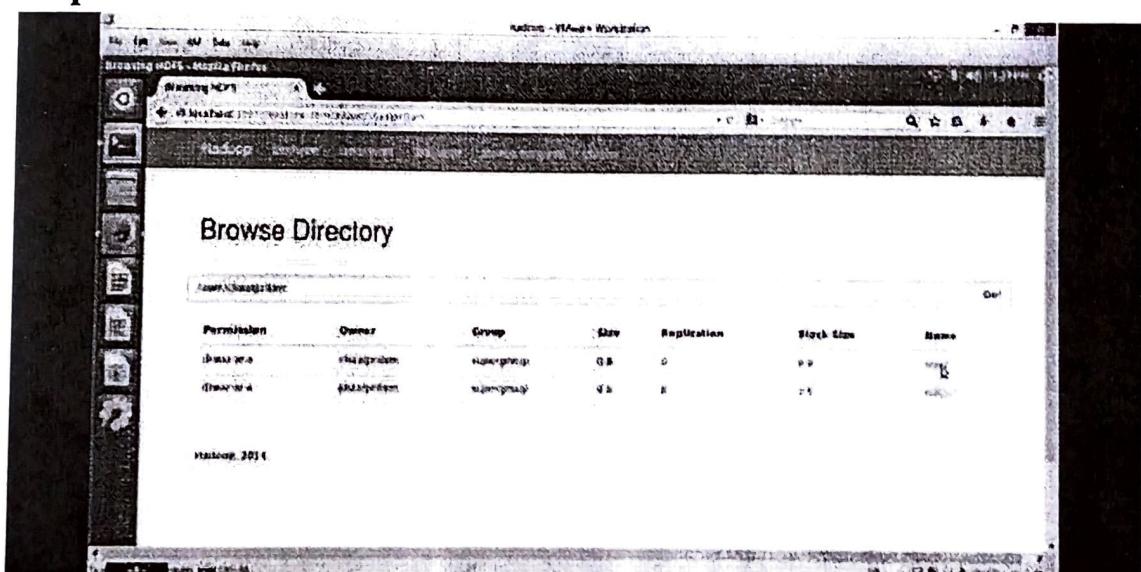
```
java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)

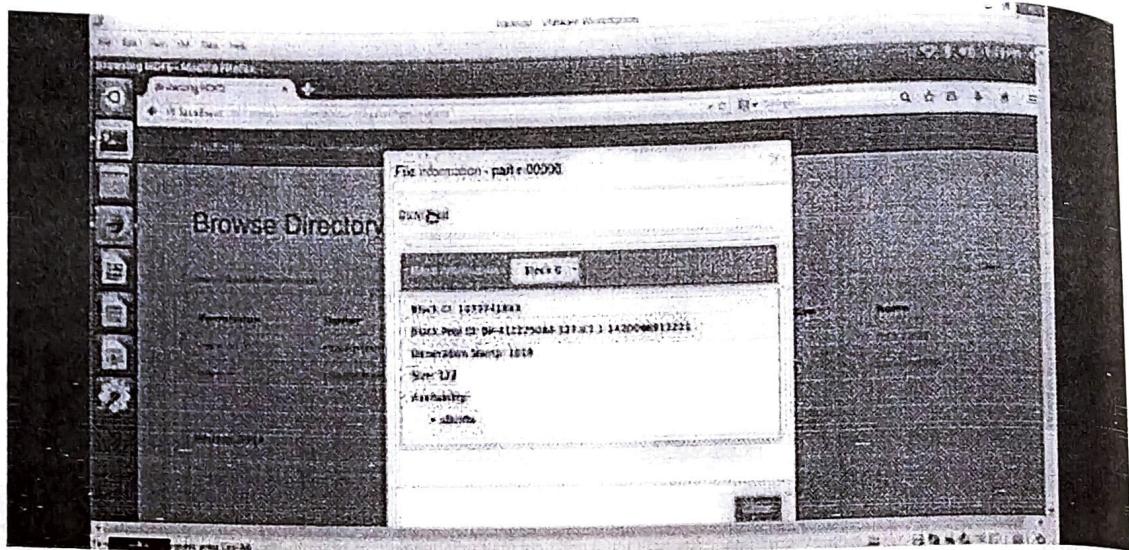
java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)

java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)

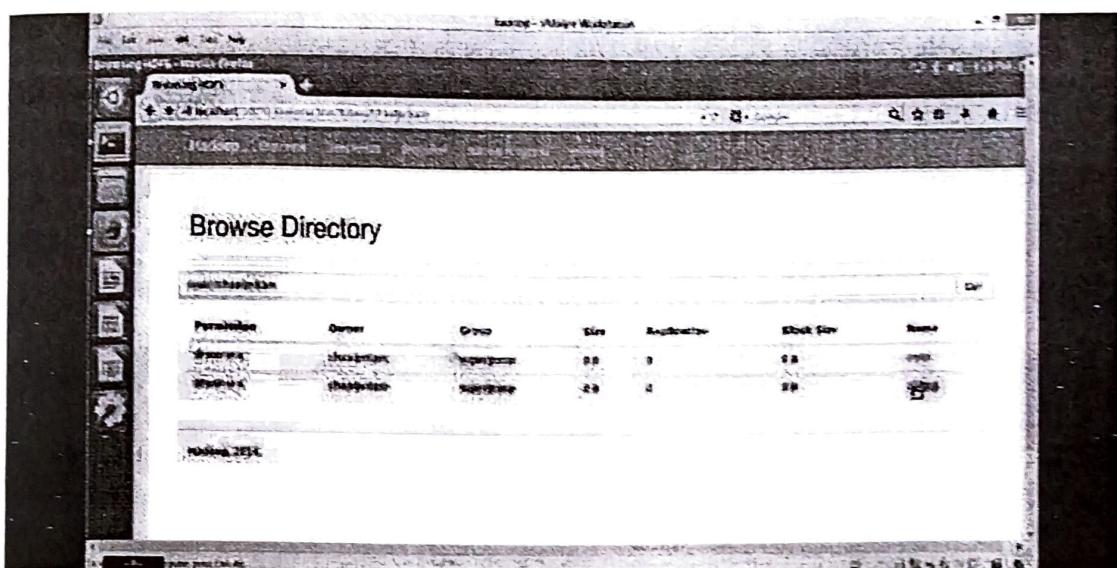
java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)
```

Step 8:

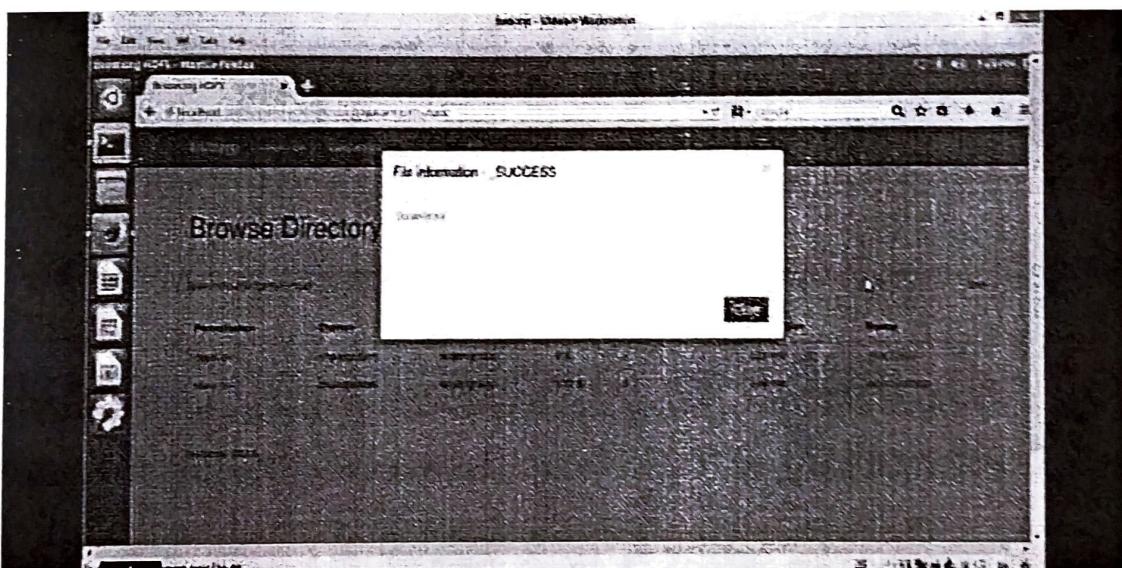




Step 9:



Step 10:



RESULT

Thus the running word count on hadoop single node cluster is done and output is obtained

Ex.no.4

12/10/21

LAUNCHING VIRTUALBOX USING TRYSTACK

AIM:

To Launch virtualbox using trystack

PROCEDURE:

STEP 0: Prepare the environment and Update

\$sudo apt-get update -y

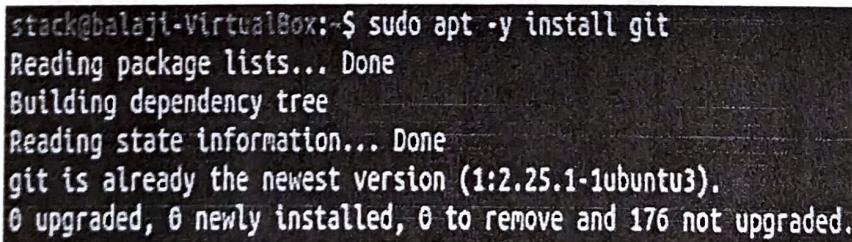


```
stack@balaji-VirtualBox:~$ sudo apt-get update -y
Get:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [100 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [319 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [55 9 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [295 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [14 4 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [70 .8 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Meta data [268 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metad ata [24.3 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [93.8 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metad ata [9,988 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Package s [250 kB]
Get:16 http://security.ubuntu.com/ubuntu focal-security/universe i386 Packages [50.3 kB]
```

STEP1: Find your IP address in Linux Server ~\$ ipaddr show

STEP2: Install Git Package to access Git-Repository

~\$ sudo apt -y install git



```
stack@balaji-VirtualBox:~$ sudo apt -y install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version (1:2.25.1-1ubuntu3).
0 upgraded, 0 newly installed, 0 to remove and 176 not upgraded.
```

STEP 3: GitCLone Command Make copy of existing Repo

```
~$git clone https://git.openstack.org/openstack-dev/devstack
```

```
stack@balaji-VirtualBox:~$ git clone https://git.openstack.org/openstack-dev/devstack
Cloning into 'devstack'...
warning: redirecting to https://opendev.org/openstack/devstack/
remote: Enumerating objects: 46226, done.
remote: Counting objects: 100% (46226/46226), done.
remote: Compressing objects: 100% (21129/21129), done.
remote: Total 46226 (delta 32674), reused 37512 (delta 24392)
Receiving objects: 100% (46226/46226), 9.50 MiB | 417.00 KiB/s, done.
Resolving deltas: 100% (32674/32674), done.
```

STEP4: Command to Change Directory To devstack

```
stack@balaji-VirtualBox:~$ cd devstack
```

STEP 5: Command to list files in Devstack

```
stack@balaji-VirtualBox:~/devstack$ ls
clean.sh          functions-common local.conf      run_tests.sh  tools
CONTRIBUTING.rst  FUTURE.rst       MAINTAINERS.rst samples      tox.ini
data              gate             Makefile        setup.cfg    unstack.sh
doc               HACKING.rst     openrc         setup.py
extras.d          inc              playbooks     stackrc
files             lib              README.rst   stack.sh
functions        LICENSE          roles         tests
stack@balaji-VirtualBox:~/devstack$ nano local.conf
```

The screenshot shows a terminal window titled "stack@balaji-VirtualBox:~/devstack". The window contains the following text:

```
GNU nano 4.8
[[local|localrc]]
ADMIN_PASSWORD=admin123
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
HOST_IP=10.8.2.15
```

STEP 6: Insert Set Of code in Local.conf

~\$nano local.conf

-nano is an easy to use command line text editor for Unix and Linux

STEP7: Final Command to install OpenStack

```
stack@batelajt-VirtualBox:~/devstack$ ./stack.sh
+ unset GREP_OPTIONS
+ unset LANG
+ unset LANGUAGE
+ LC_ALL=en_US.utf8
+ export LC_ALL
++ env
++ grep -E '^OS_'
++ cut -d _ -f 1
+ unset -
+ umask 022
+ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
/usr/local/games:/snap/bin:/usr/local/sbin:/usr/sbin:/sbin
++ dirname ./stack.sh
++ cd .
++ pwd
+ TOP_DIR=/home/stack/devstack
+ NOUNSET=
+ [[ -n '' ]]
++ date +%
+ DEVSTACK_START_TIME=1602054267
+ [[ -r /home/stack/devstack/.stackenv ]]
+ FILES=/home/stack/devstack/files
+ '[' '' -d /home/stack/devstack/files ']'
+ '[' '' -d /home/stack/devstack/inc ']'
+ '[' '' -d /home/stack/devstack/lib ']'
+ [[ '' == \y ]]
+ [[ 1001 -eq 0 ]]
+ [[ -n '' ]]
```

RESULT

Thus the process for Launching Virtualbox Using Trystack is done and output is obtained.

Ex. No. 5 INSTALL HADOOP SINGLE NODE CLUSTER

12/10/21

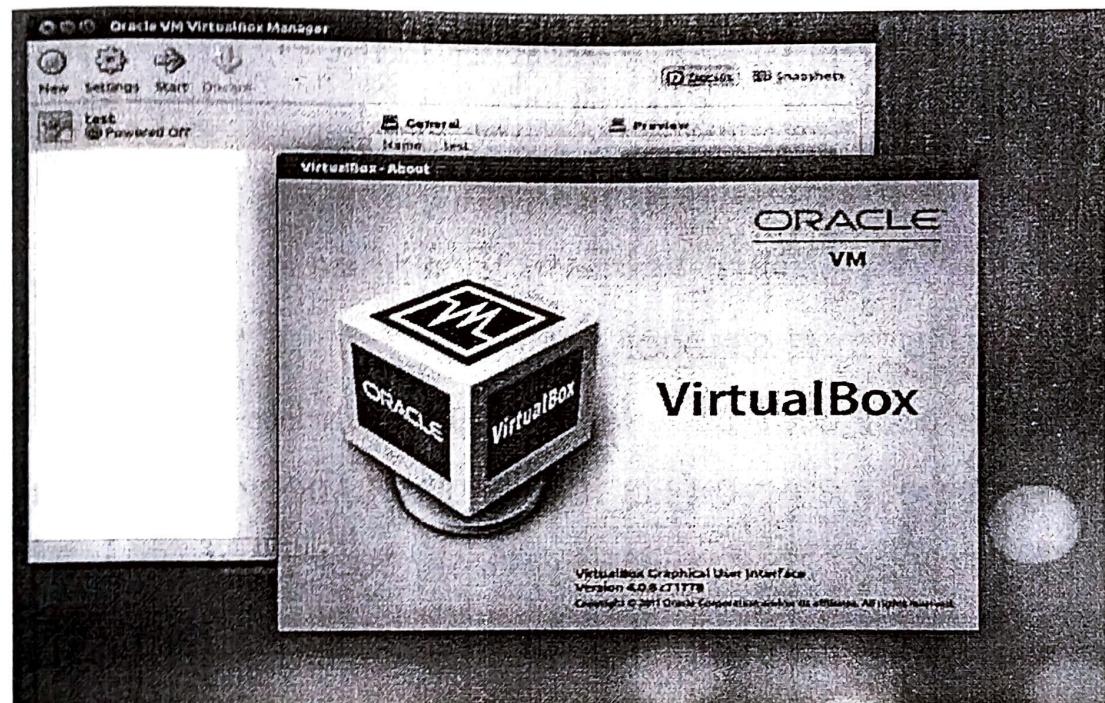
AIM:

- VIRTUAL BOX: it is used for installing the operating system on it.
- OPERATING SYSTEM: You can install Hadoop on Linux based operating systems. Ubuntu is used.
- JAVA: You need to install the Java 8 package on your system.
- HADOOP: You require Hadoop 3.2.1 package.

PROCEDURE:

Install virtual box

<https://download.virtualbox.org/virtualbox/6.1.14/VirtualBox-6.1.14-140239-Win.exe>



After completing the virtual box installation ,Install Ubuntu

<https://ubuntu.com/download/desktop>

Click to download Hadoop:

<https://hadoop.apache.org/releases.html>

STEPS:

Step 1: Install Java

Command: sudo apt install openjdk-8-jdk

cd /usr/lib/jvm

Step 2: Open bashrc file

Command: sudo nano ~/.bashrc

Location Path of the Environment variable:

Past these:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64  
export PATH=$PATH:$HOME/.local/bin  
export PATH=$PATH:/usr/lib/jvm/java-8-openjdk-amd64/bin  
export HADOOP_HOME=~/Downloads/hadoop-3.2.1  
export PATH=$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=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar  
export HADOOP_LOG_DIR=$HADOOP_HOME/logs  
export PDSH_RCMD_TYPE=ssh  
export HIVE_HOME=~/Downloads/apache-hive-3.1.2-bin  
export PATH=$PATH:~/Downloads/apache-hive-3.1.2-bin/bin
```

Then come out of the file.

Step 3:Install Secure shell

Command: sudo apt-get install ssh

Step 4: Install parallel shell

Command: sudo apt-get install pdsh

Step 5: Extract Hadoop tar file:

Command: tar -zxvf ~/Downloads/hadoop-3.2.1.tar.gz

Go to Hadoop directory

Command: cd hadoop-3.2.1/etc/Hadoop

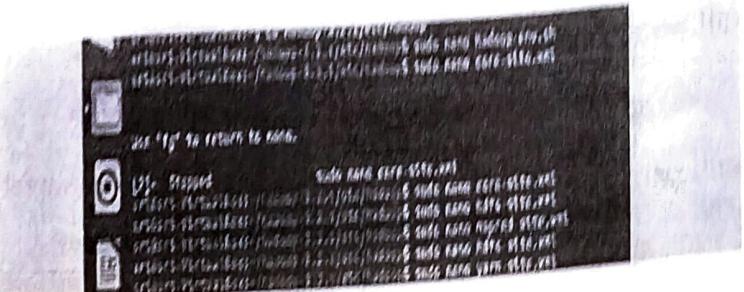
Step 6: Open Hadoop-env.h

Command: sudo nano hadoop-env.h

.JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

(Set the path for JAVA_HOME)

```
1 Set Hadoop-specific environment variables here.
2
3 # The port number to use for the master task for the HDFS namenode.
4 # This port must be used by all HDFS daemons. Therefore,
5 # one has to edit this file to set JAVA_HOME, and update
6 # other variable settings (including classpath).
7
8 # Miscellaneous rules:
9
10 # If you want to run Hadoop in a distributed mode
11 # (multiple hosts), edit the HDFS configuration
12 # files to reflect the topology of your cluster.
13 # You can specify customized values on the command line
14 # for example:
15
16 .JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
17
18 # therefore, the most sensible thing is to set JAVA_HOME
19 # environment variable for compilation and use export if needed
20 # (or setable, update this file accordingly).
```



Step 7: Go
coresite.xml

Command: sudo nano coresite.xml

Paste these:

```
<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>
```

```
-- but some specific permission inheritance in this file --  
  
#  
# HDFS  
#  
# fs.defaultFS = hdfs://localhost:9000  
#  
#  
# HDFS  
#  
# hadoop.proxyuser.datataskt.groups =  
#  
#  
# HDFS  
#  
# hadoop.proxyuser.datataskt.hosts =  
#  
#  
# HDFS  
#  
# hadoop.proxyuser.tserver.hosts =  
#  
#  
# HDFS  
#  
# hadoop.proxyuser.tserver.groups =
```

Step 8: Go to hdfs-site.xml

Command: sudo nano hdfs-site.xml

Paste these:

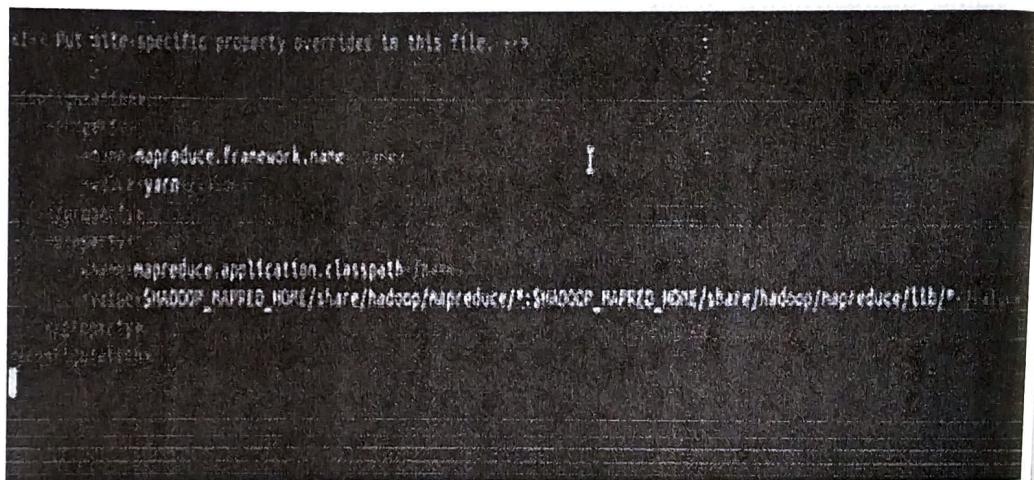
```
<configuration>  
  <property>  
    <name>dfs.replication</name>  
    <value>1</value>  
  </property>  
</configuration>
```

Step 9: Go to mapred-site.xml

Command: sudo nano mapred-site.xml

Paste these:

```
<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>
```



The screenshot shows a terminal window with the following text:

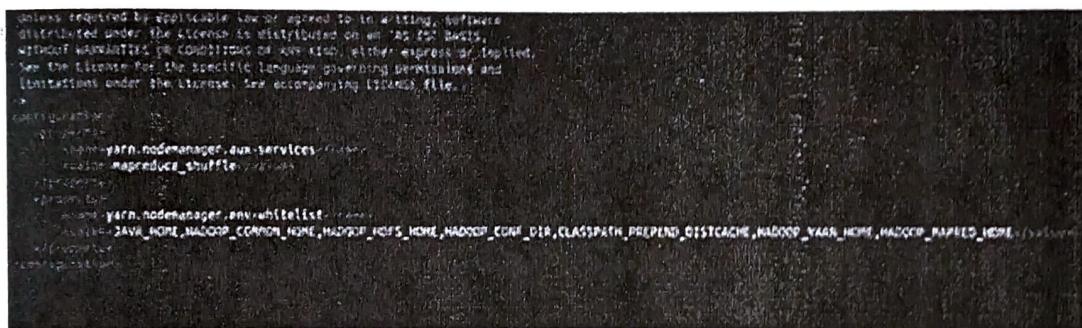
```
the put site-specific property overrides to this file. . .>
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.application.classpath</name>
<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
</property>
</configuration>
```

Step 10: Go to yarn-site.xml

Command: sudo nano yarn-site.xml

Paste these:

```
<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_PREPEND,DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
</property>
</configuration>
```



Step11:Establish Localhost:

Command: ssh localhost

Step 12:Key generation:

Command: ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa

```
centos-virtualbox:~$ ssh-keygen -t rsa -P '' -f /home/centos/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/centos/.ssh/id_rsa
Your public key has been saved in /home/centos/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:KwHmMvVqkQeLqZGm7HCn18C7Oz779hB91gPz-1P7u28o8
The key's randomart image is:
+---[SHA256]---
```

Step 13:Granting Permission to the file

Command: chmod 0600 ~/.ssh/authorized_keys

Command:~/Downloads/hadoop-3.2.1/bin/hdfsnamenode-format

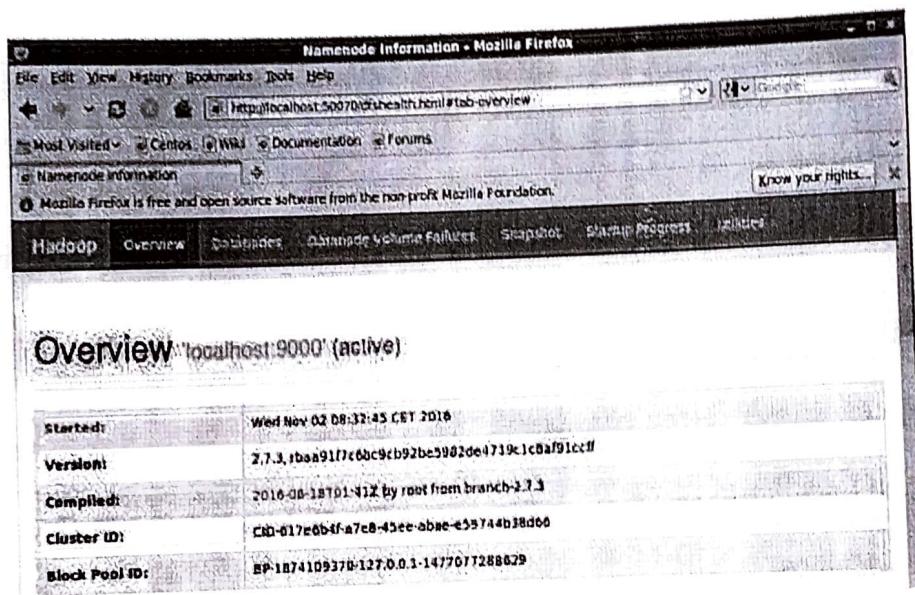
Step 14: Export pdsh

Command: export PDSH_RCMD_TYPE=ssh

Step 15:dfs node:

Command:~/Downloads/hadoop-3.2.1/sbin/start-dfs.sh

Step 16:Open the browser: Type localhost:9870



RESULT:

Thus the process to install hadoop single node cluster is done and output is obtained.

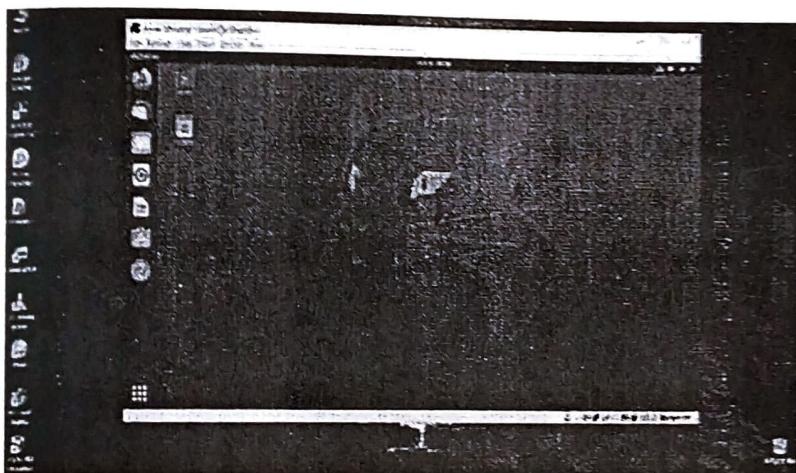
Ex.No.06 Find a procedure to transfer the files from one virtual machine to another virtual machine.

AIM:

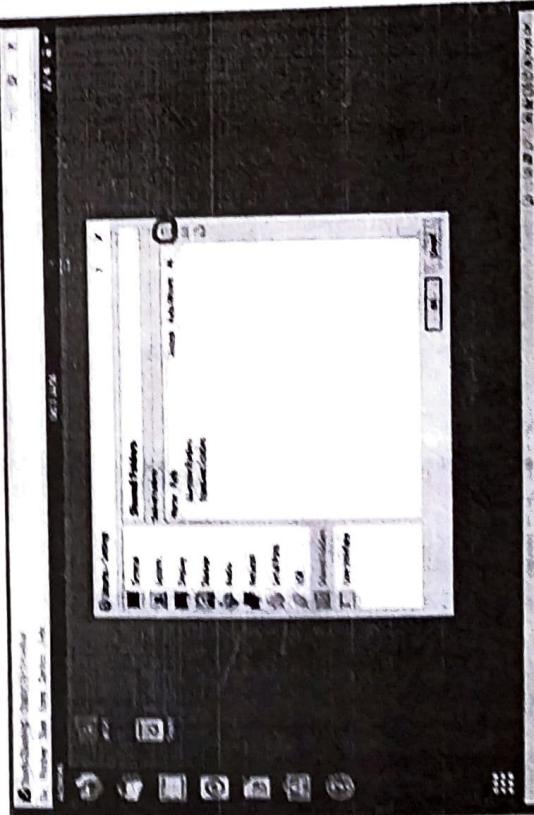
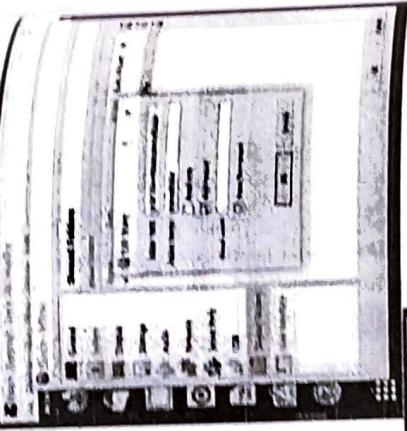
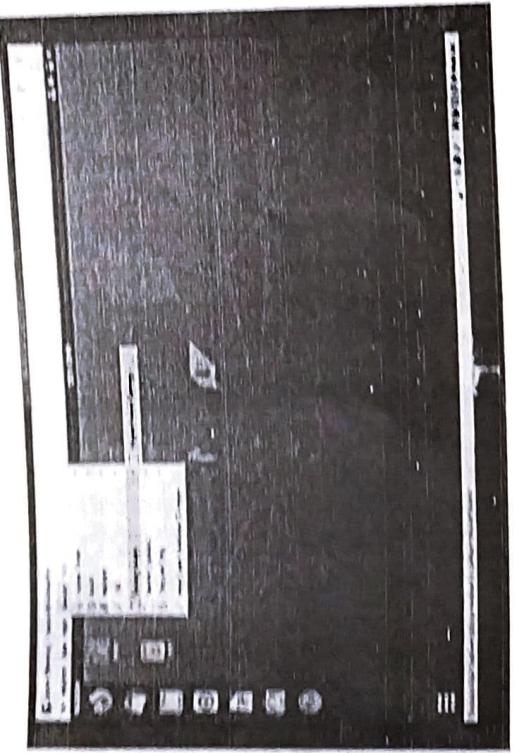
To find a procedure to transfer the files from one virtual machine to another virtual machine.

PROCEDURE:

1. Download the virtual Box in your Host OS and Install the Guest OS in the virtual box.
2. First we will share a folder from host machine to one vm. Create a folder in the Host Machine.



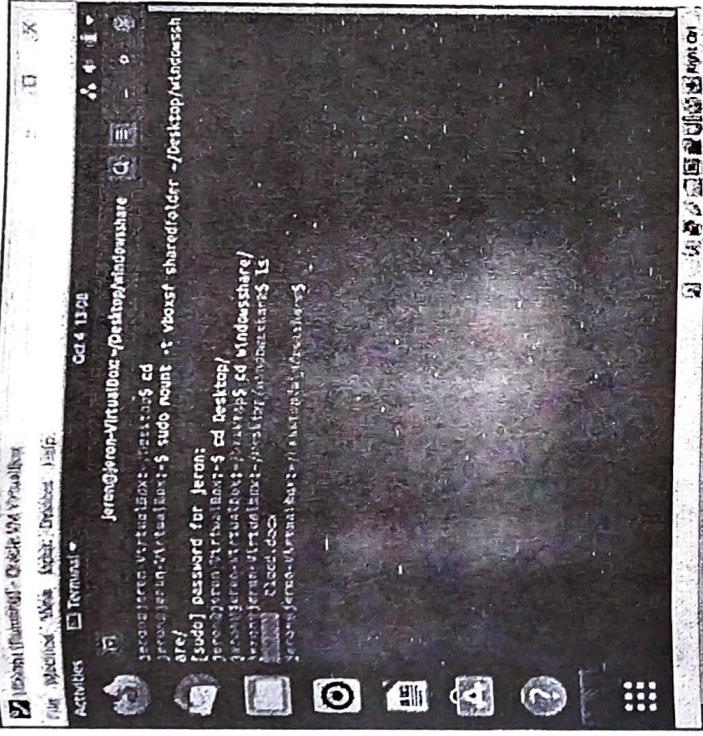
3. Install the vm and download the necessary files and then install another vm say vm2 and download the necessary files.
4. Now run the vm and Go to Device->Insert Guest additional CD image option and install the CD image.
And eject the CD image this helps in the access of file from one vm to another.
5. Go to Device->Shared Folder->Shared Folder Settings and a dialog box appears.



6.Click on the AddFileas marked in the above picture and give the directory of a sharefolder in the Host machine and click ok

7.Now open terminal and create a folder of your choice with the commands
`sudo mkdir ~/Desktop/windowsshare` in the vm, the folder with windows\$

will be created in the vm desktop. And type sudo mount -t vboxsf “folder_name” ~/Desktop/windowsshare/ and enter.



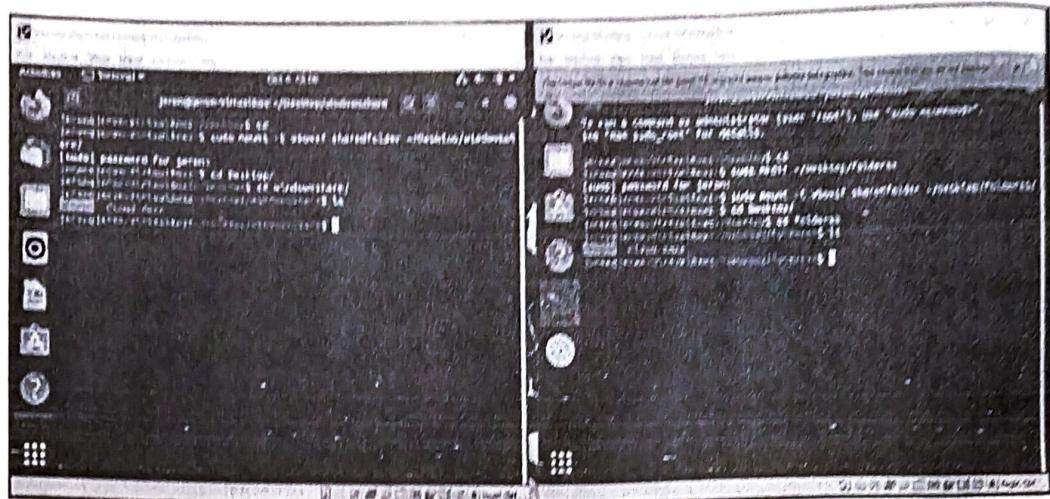
8. Now, we can access the shared file in the vm. Now to share folder from one vm to another.

9. Create a file inside the folder windowsshare of vm and Run the vm2 and do step 4 to 6.

10. Now open the terminal in vm2 and create a folder of your choice as step 7.

11. Now type sudo mount -t vboxsf “filename” ~/Desktop/”filename”

12. Now enter and view the folder and the folder in the vm 1 will be seen in vm2



RESULT

Thus program to find a procedure to transfer the files from one virtual machine to another virtual machine.

Ex. No 7
26/10/21

AIM

To execute cloud sim into eclipse

CLOUD SIM INTO ECLIPSE

What is a Cloudsim?



- A framework made on java
- Simulates cloud services
- Collection of java classes
- It Models DataCentre, Host, VM, Cloudlet

DataCentre:

- represents complete hardware
- it has set of hosts(Physical machines)

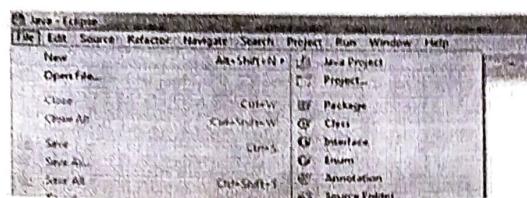
Host:

- physical machine
- it has variables to represent memory, processors, Id, scheduling scheme, etc.

PROCEDURE:

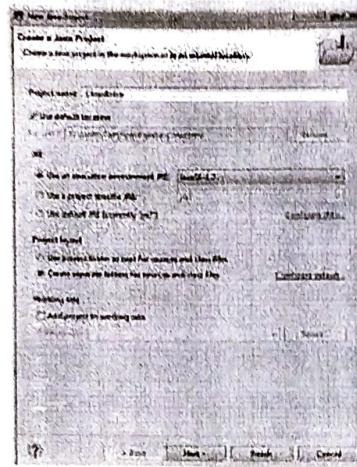
STEP BY STEP INSTALLATION OF CLOUD SIM INTO ECLIPSE

1. Open up Eclipse and go to Menu Section, then click File, keep on clicking New and finally select java project.



2. A new window will get open, Put a foot on to the following steps:-

- 2.1.Enter project name. (I have named it as CloudIntro)
- 2.2 In the next line you will see the path where your project will be created.
- 2.3 Next You need to select the JRE environment.
- 2.4 Finally Click Finish.



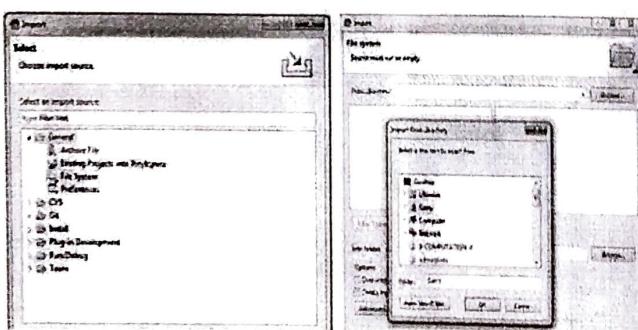
4. Next step is to go to the project CloudIntro, right click on it.

Click Import as shown.

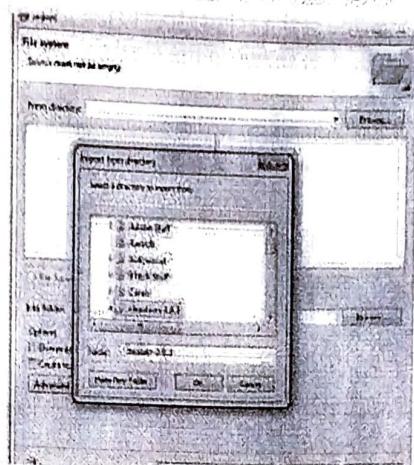


5. A new window will get open, now click File System as demonstrated in the Figure.

6. Next Step is to go to the directory where you have extracted your cloudsim tool. Figure6 is shown to guide you to get into the directory where your cloudsim folder is located.



7. Select the cloudsim and click Finish as shown in the Figure

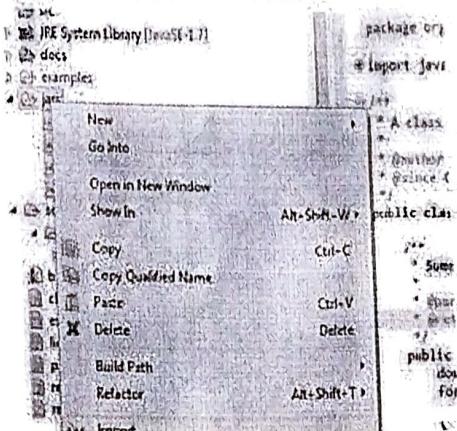


8. Now go to the link

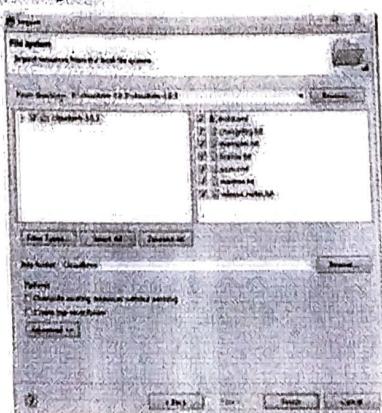
http://commons.apache.org/proper/commonsmath/download_math.cgi.

Download the file named as "commonsmath3-3.4.1-bin.zip". Unzip this file. We need jar files for math functions.

9. Now go to the left side of the eclipse tool in the project bar. Go to jar and right click on it. Click import as shown in the Figure.



10. Now go to the folder where you have placed the downloaded and extracted file as described by point 8. Then all you have to do is select that jar file and hit finish as shown by the Figure.



11. Finally the cloud sim is installed into your Eclipse environment.

RESULT

Thus the process of CloudSim into eclipse is done and output is obtained

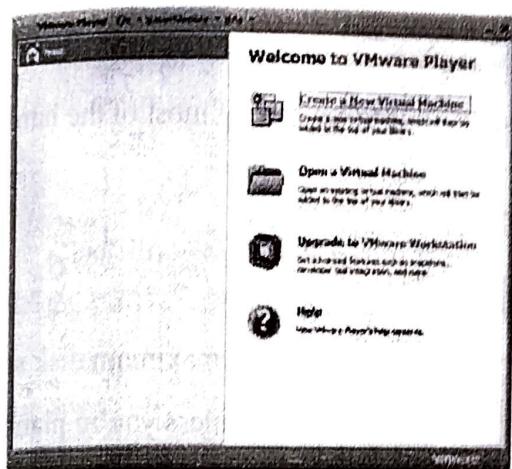
Ex. no. 8 Installing Ubuntu in a VMWare
2b | 10 | 21

AIM:

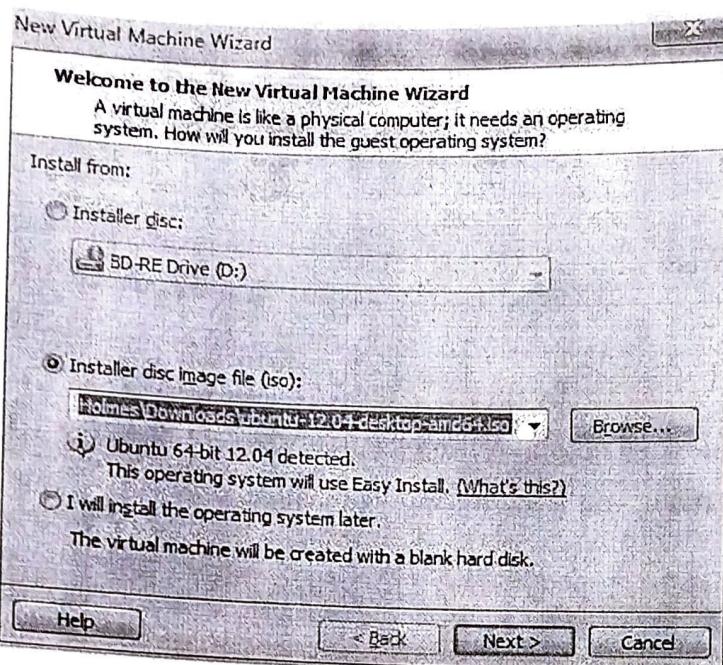
To install Ubuntu in a VMWare

PROCEDURE:

1. Download the Ubuntu iso (desktop not server) and the free VMware Player.
2. Install VMware Player and run it, you'll see something like this:

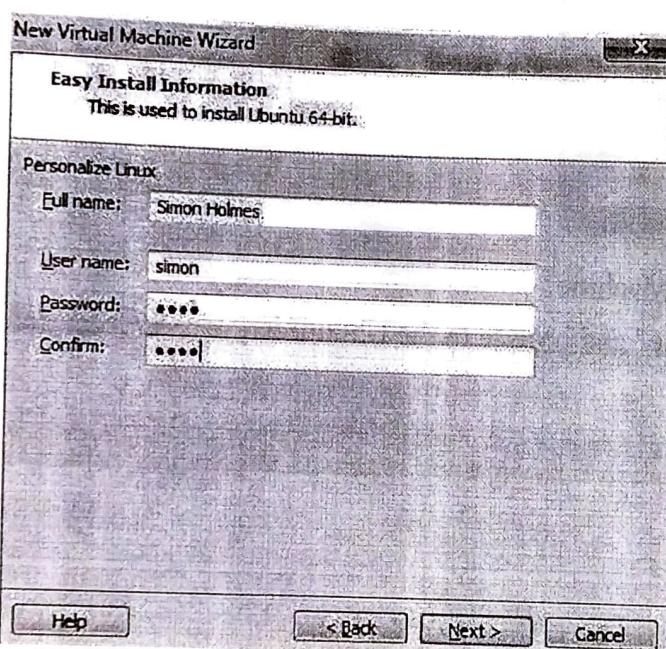


3. Select "Create a New Virtual Machine"
4. Select "Installer disc image file" and browse to the Ubuntu iso you downloaded.

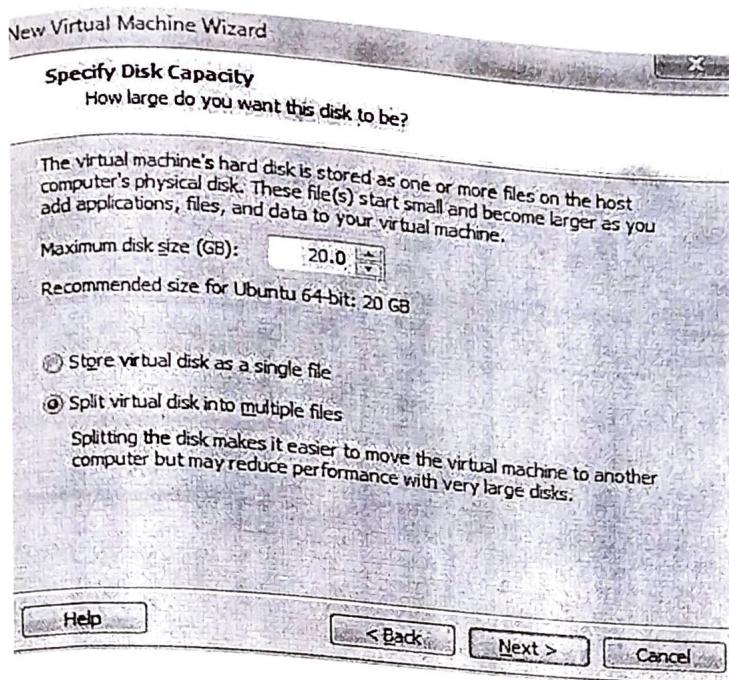


You should see that it will use Easy Install – this takes care of most of the hard work for you. Click next

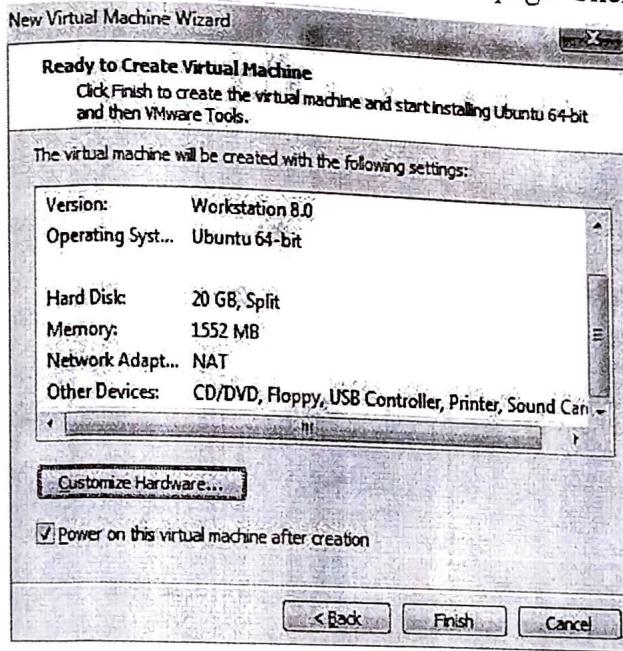
5. Enter your full name, username and password and hit next



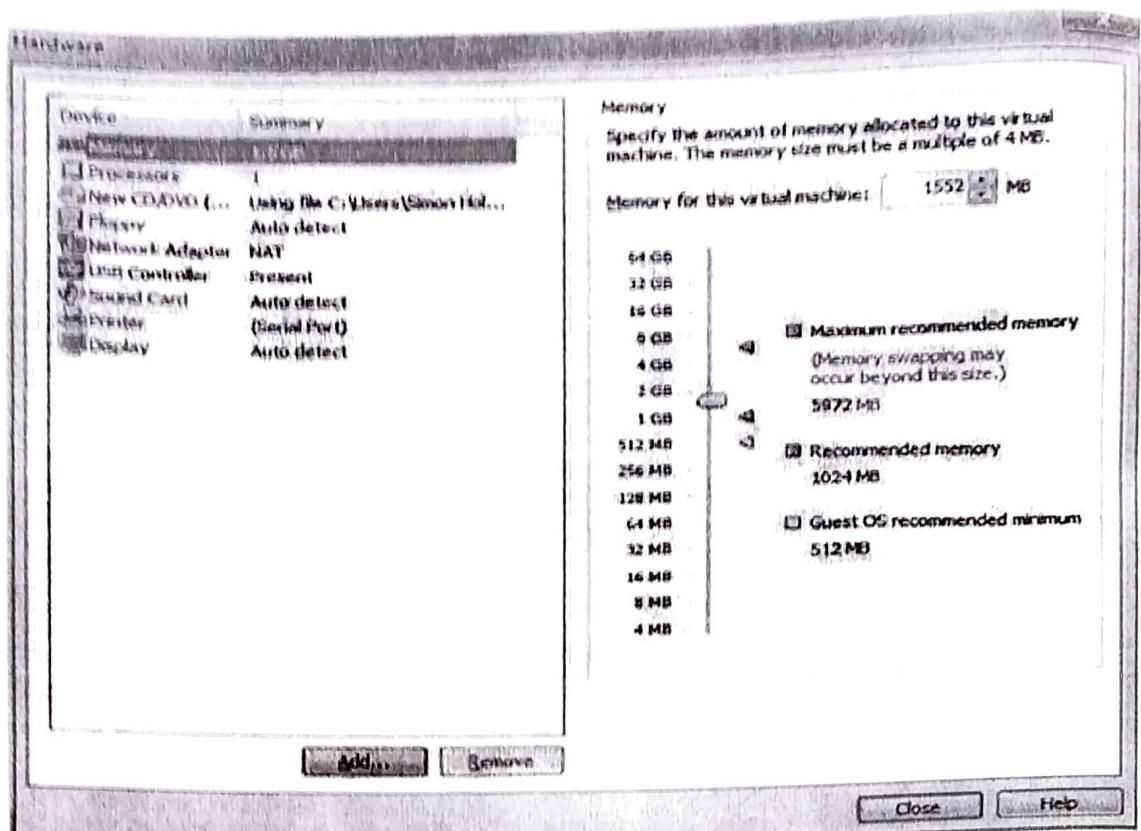
6. Select the maximum disk size and type. Unless you're planning on some really CPU intensive work inside the VM, select the "Split virtual disk into multiple files" option. Hit next when you're happy with the settings.



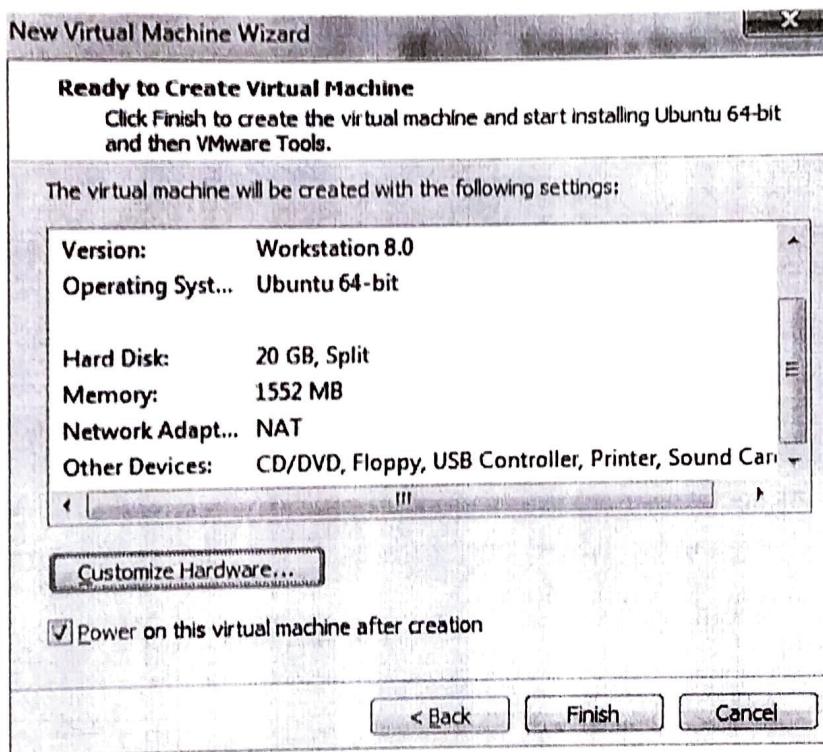
7. This brings you to the confirmation page. Click "Customize Hardware"



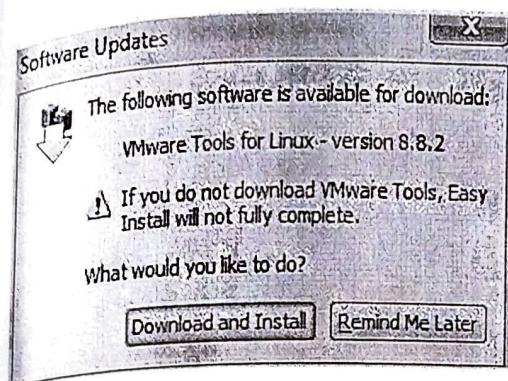
8. In the hardware options section select the amount of memory you want the VM to use. In this instance I've gone for 1.5GB out of the 8GB installed in my laptop. Leave everything else as it is and click Close.



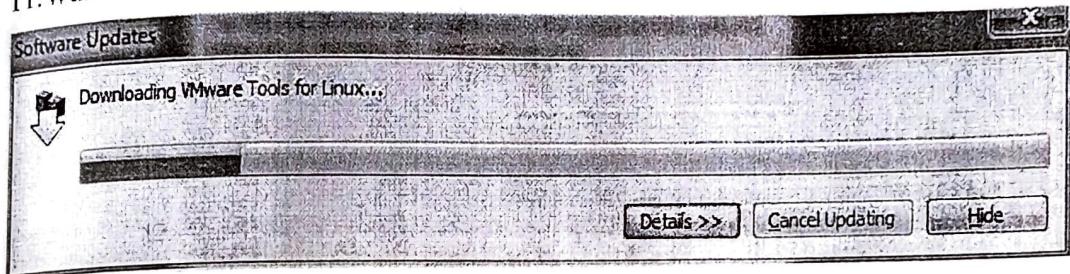
9. This brings you back to the confirmation page. Click Finish this time



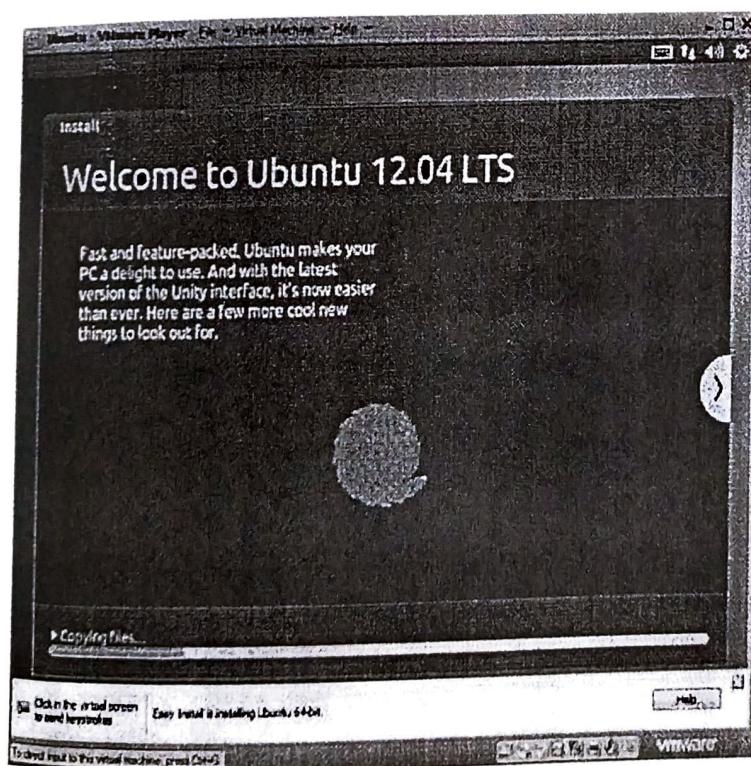
10. You will probably be prompted to download VMware Tools for Linux. Click "Download and Install" to continue



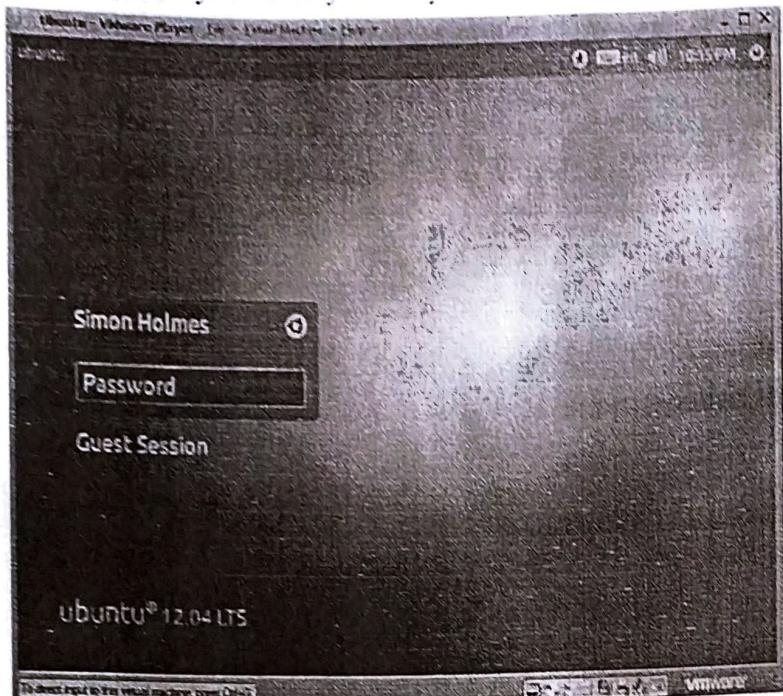
11. Wait for it to install



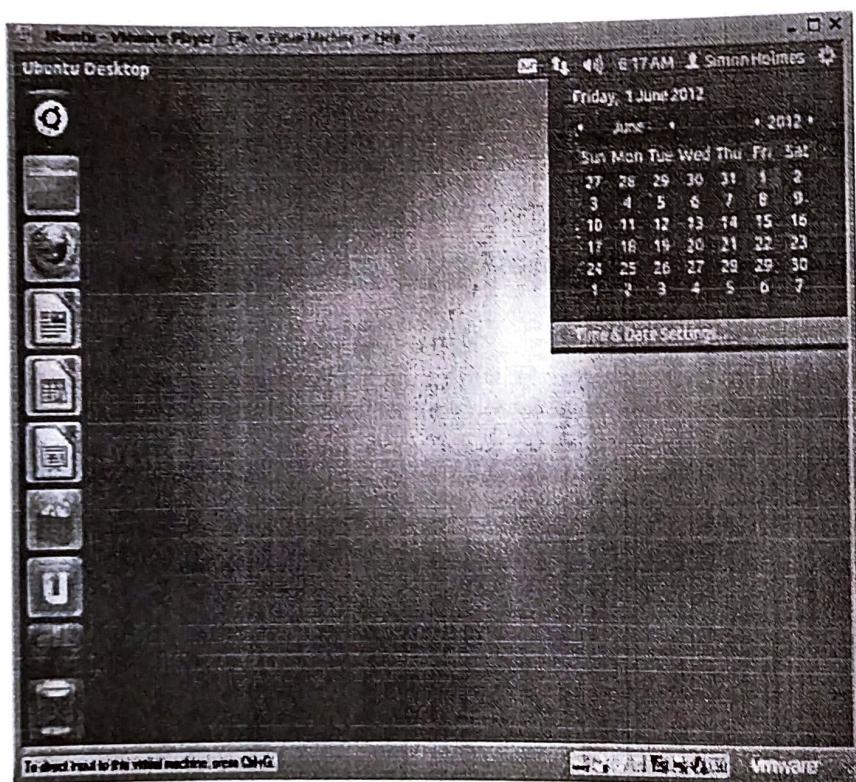
12. Ubuntu will then start to install, so keep waiting (or do what I did and go to bed!)



13. When all is done you'll be presented with the Ubuntu login screen. So enter your password and you're on your way.



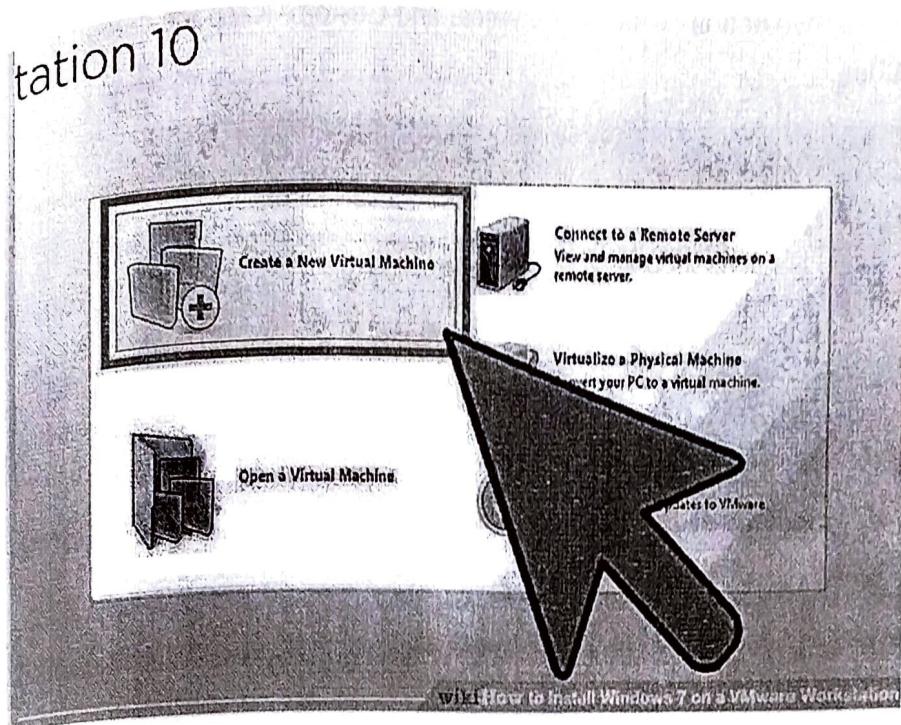
14. Click the clock in the top right to set your time and date settings



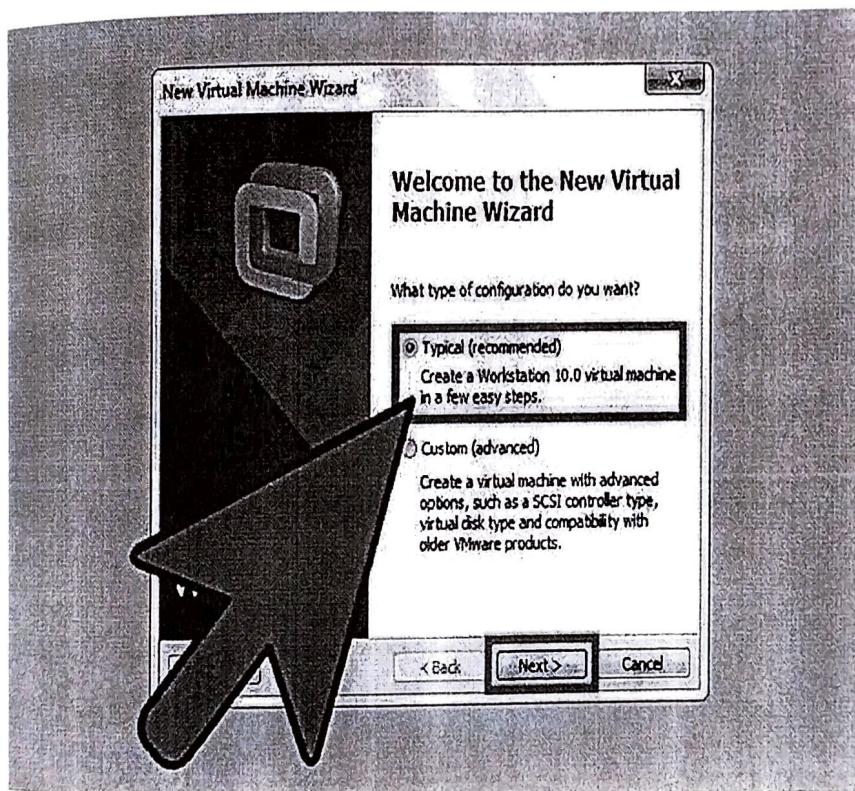
15. Once you've set that up, you're up and running with Ubuntu in VMware Play on your Windows machine. Congratulations and enjoy!

Installing Windows 7 in a VMware

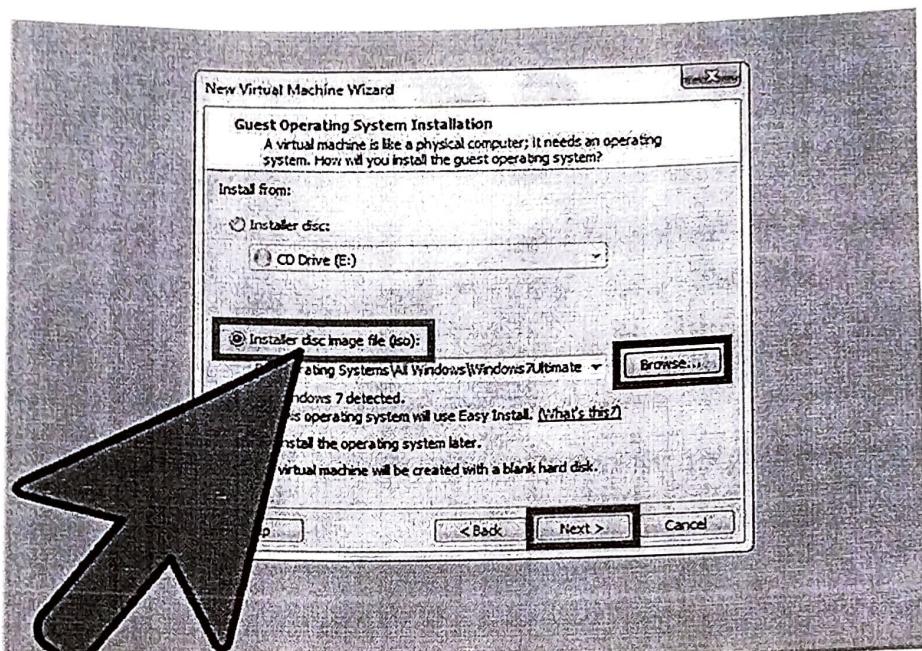
station 10



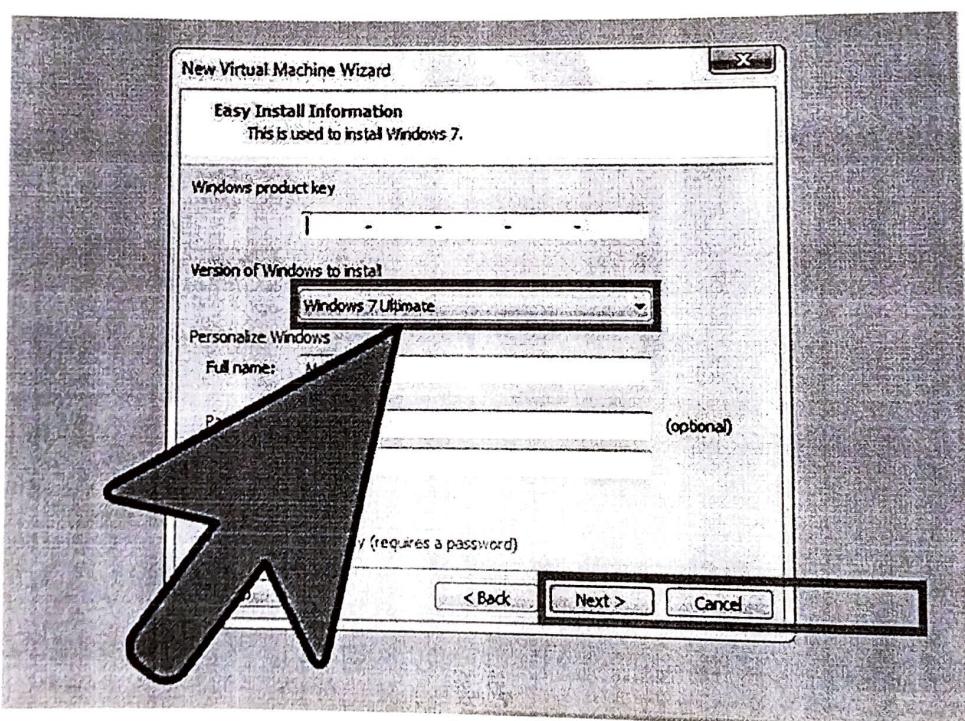
1. Create a new virtual machine. Once you open VMware Workstation, click "Create a New Virtual Machine".



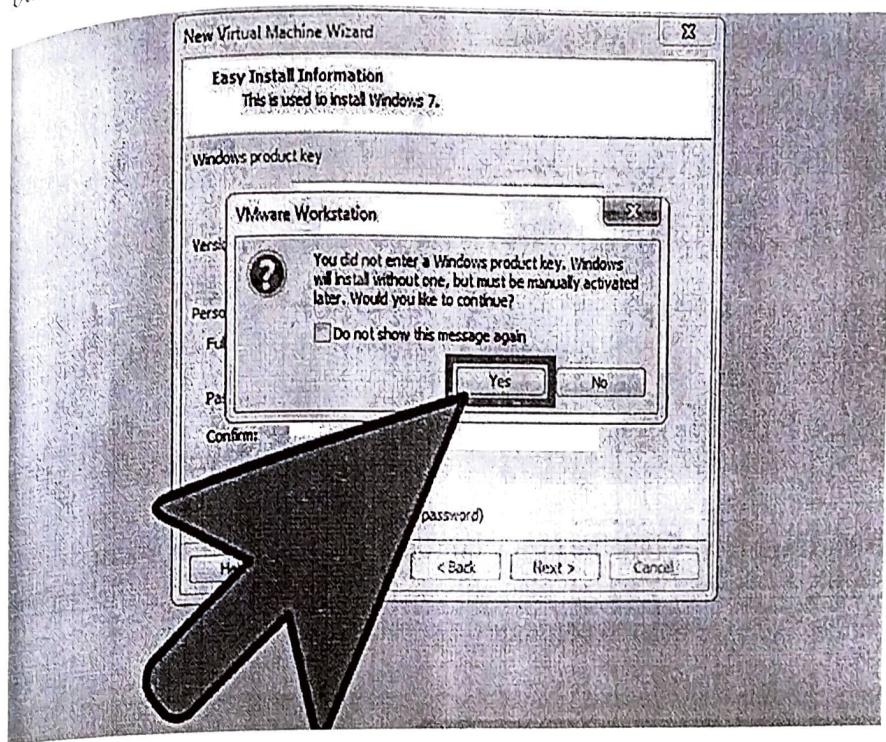
2. Select type of configuration. You will see a New Virtual Machine Wizard dialog box. There are two options which are Typical and Custom. Keep the default and click "Next" button.



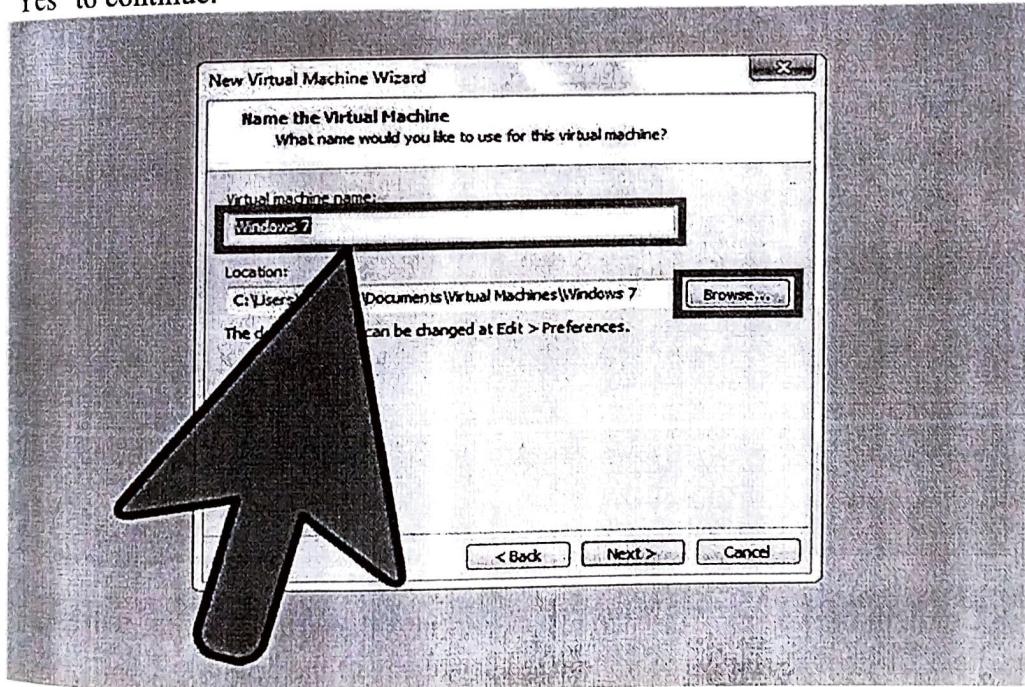
3. Choose "Installer disc image file". This type matches the iso file you download. Click "Browse" to locate your Windows 7 iso file. Then, click "Next".



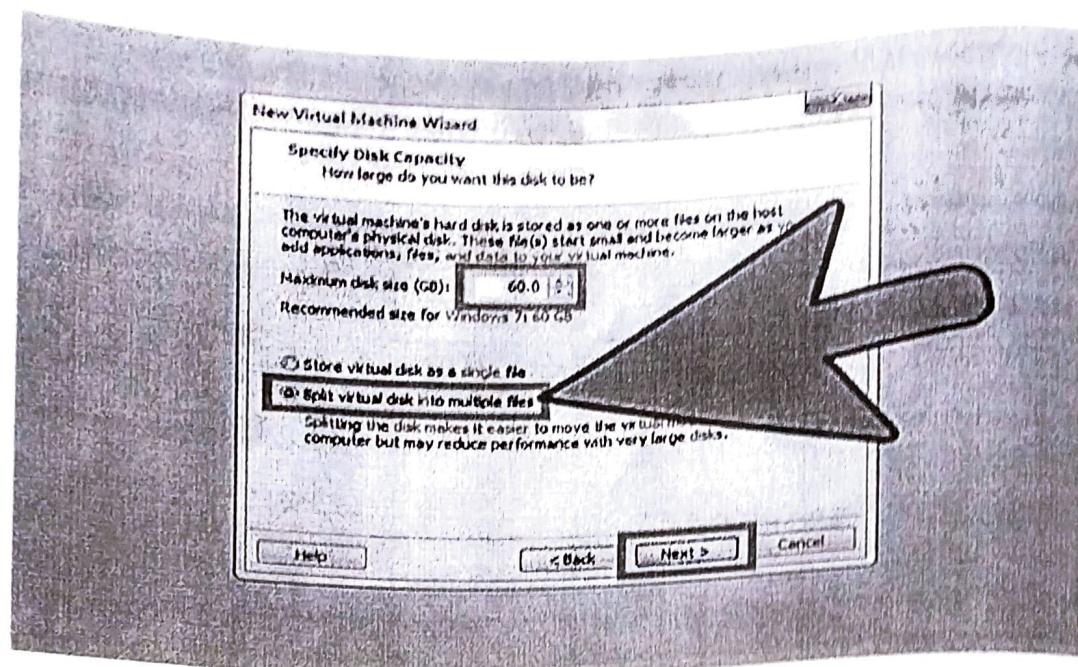
4. Select the version of Windows to install. The version depends on the iso file you download. You can set your product key and personalize Windows later. Click "Next" button to continue.



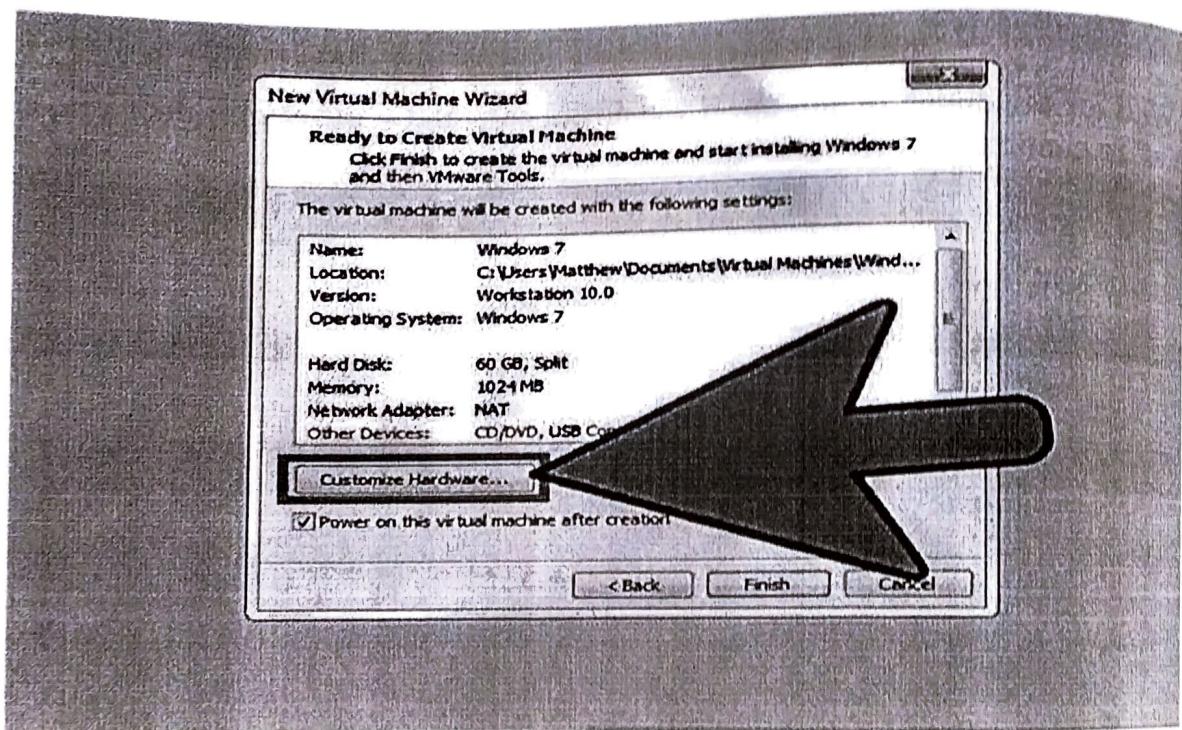
5. Wait for the dialog to pop up. If you do not enter the Windows product key, click "Yes" to continue.



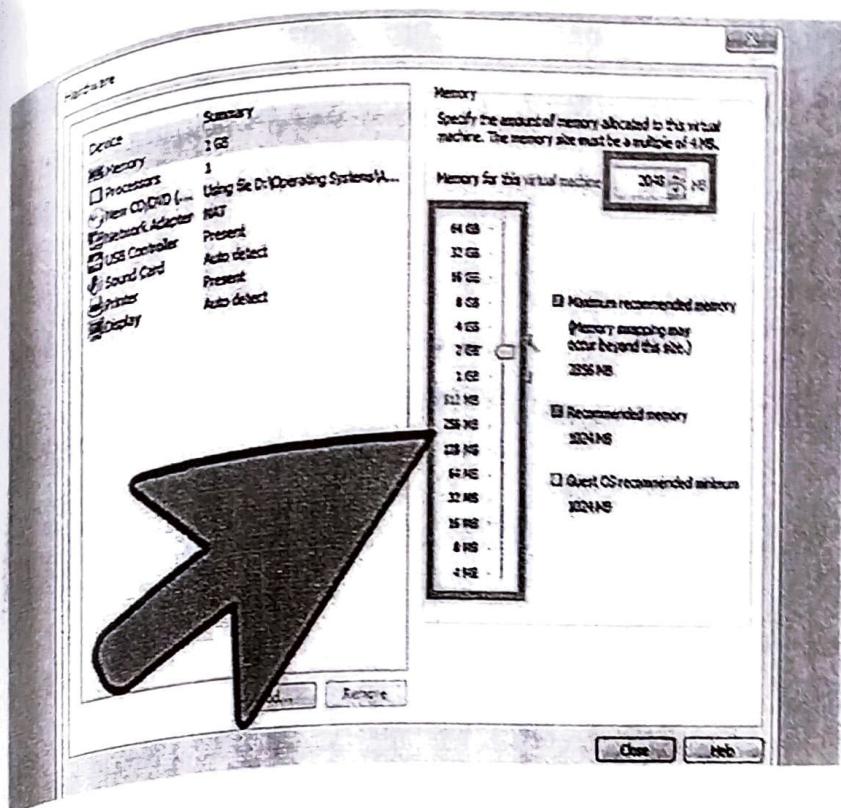
6. Name the virtual machine. Change the name and location if you want. Click "Browse" to modify the path. Then, click "Next".



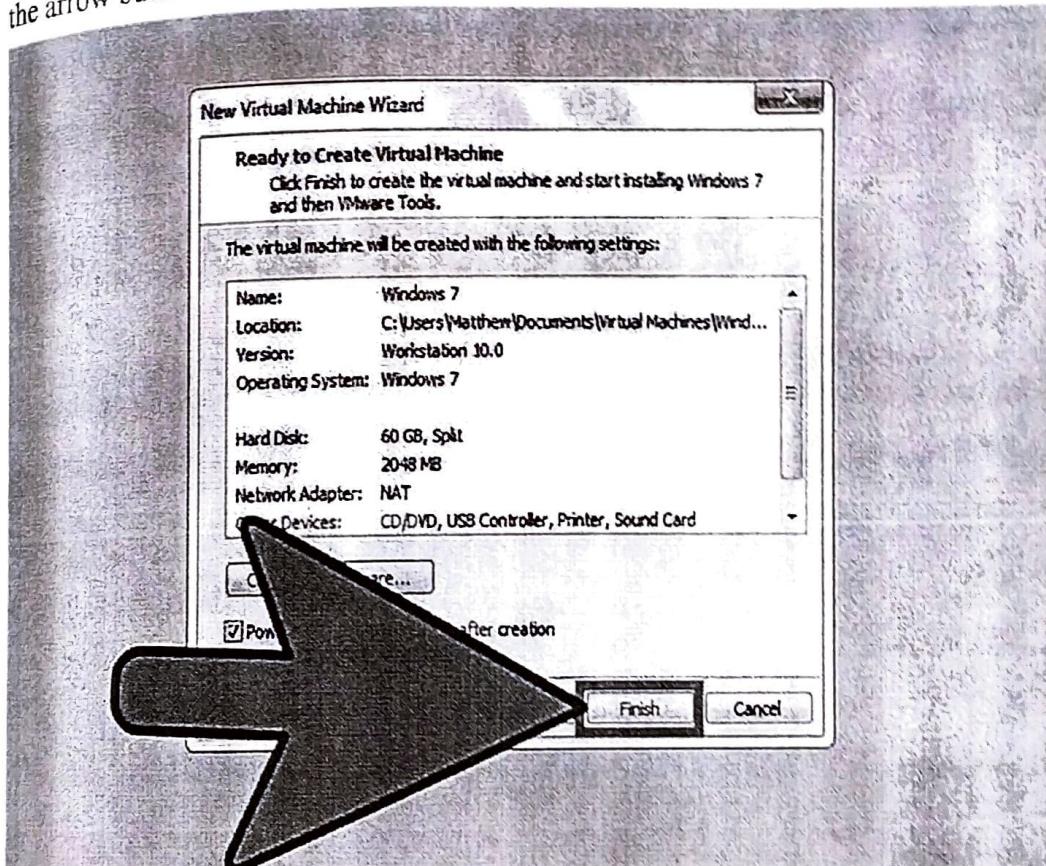
7.Specify Disk Capacity. Click the arrow button to change the maximum size of virtual machine's hard disk. You can also choose store as one or more files on host computer. Click "Next" button to continue.



8.Confirm the setting. This step lists the settings the virtual machine will create. Click "Customize Hardware" button to change any of the details.



9. Change Memory. If you want to change the memory of the virtual machine, select the arrow button or drag the slider tab. Click "Close" to go back to the last dialog box.



10.Create the new virtual machine. After confirming that the settings are all accurate, click the "Finish" button to start the process.

RESULT

Thus the process of installing ubuntu in a VMWare is done and output is obtained.

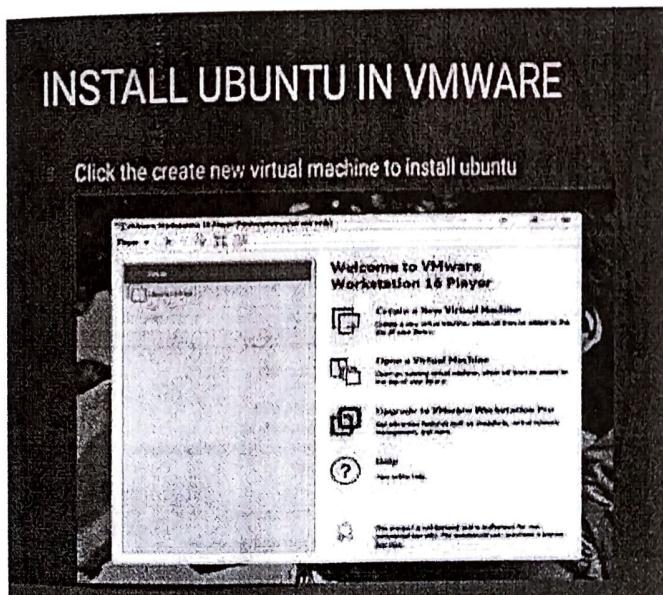
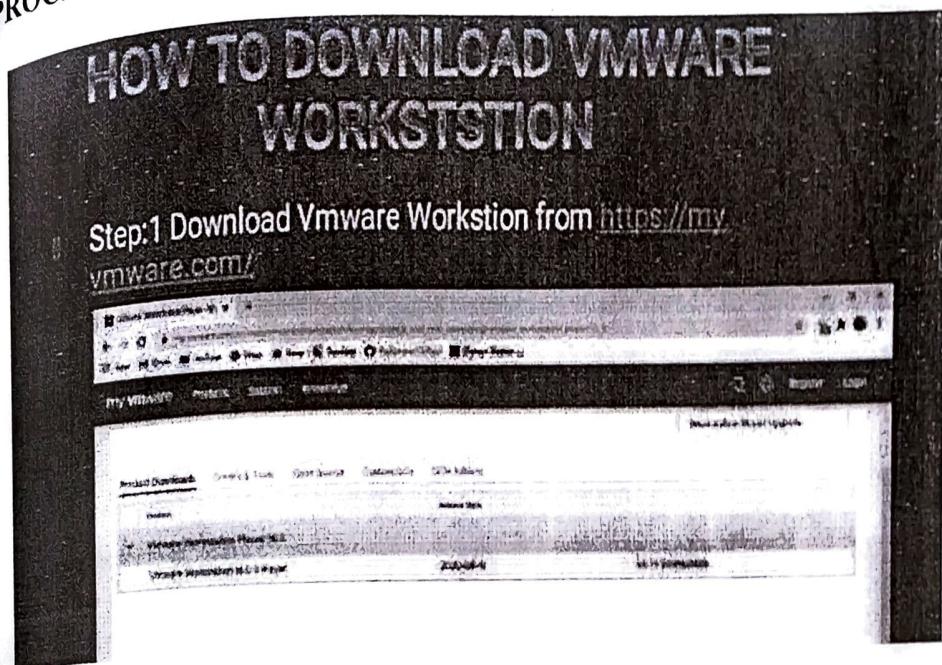
INSTALL AND RUN C COMPILER IN UBUNTU

Ex.No. 9
26/10/21

AIM:

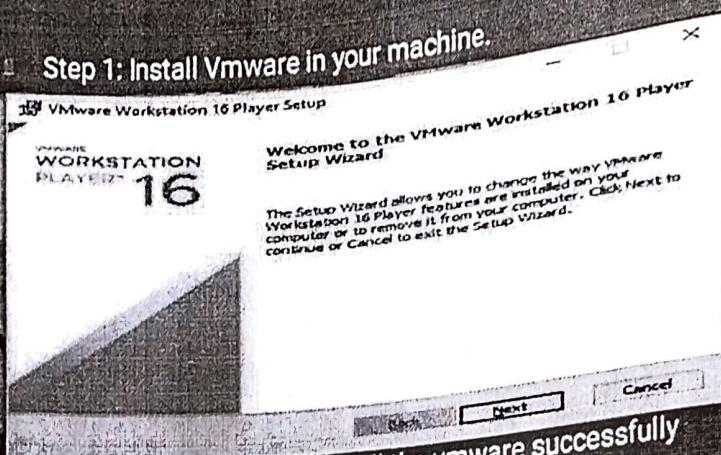
To install and run C compiler in ubuntu

PROCEDURE:



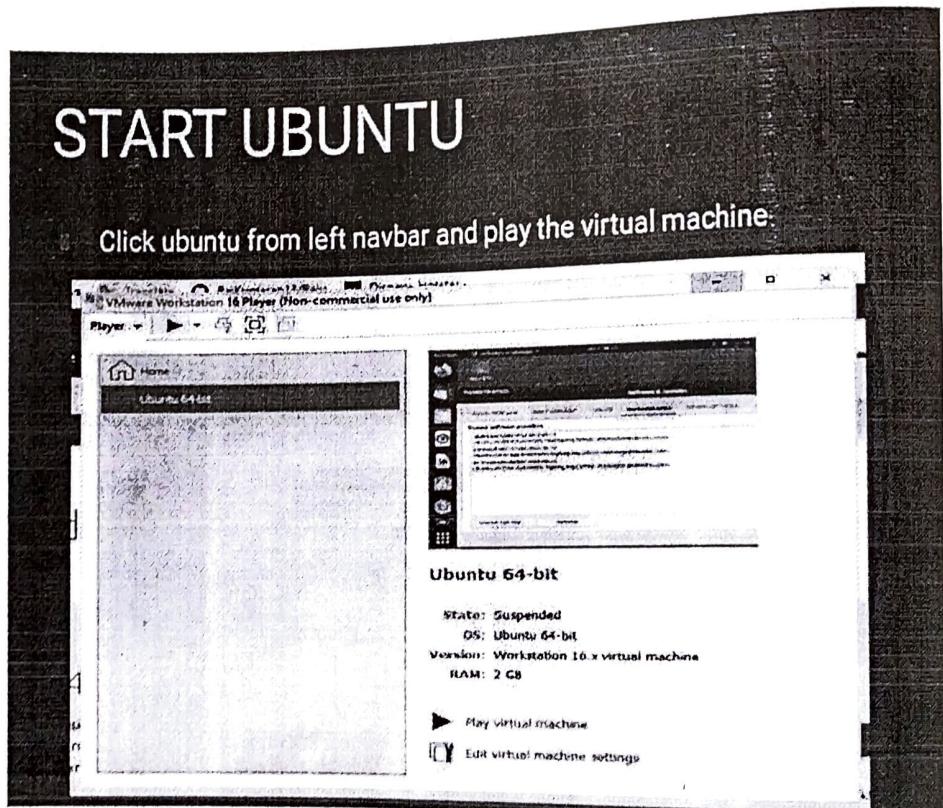
INSTALL Vmware IN YOUR MACHINE

Step 1: Install Vmware in your machine.



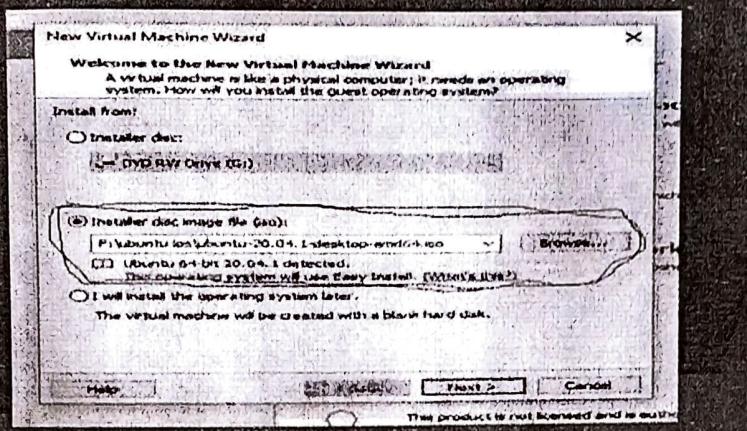
START UBUNTU

Click ubuntu from left navbar and play the virtual machine.



Step 1: click installer disc image file and locate the file directory.

Download UBUNTU iso file from <https://ubuntu.com/download/desktop>



INSTALL C COMILER IN UBUNTU

Open terminal in ubuntu by Search box-Terminal-Open

Install GCC by typing this command in the terminal \$ sudo apt install build-essential

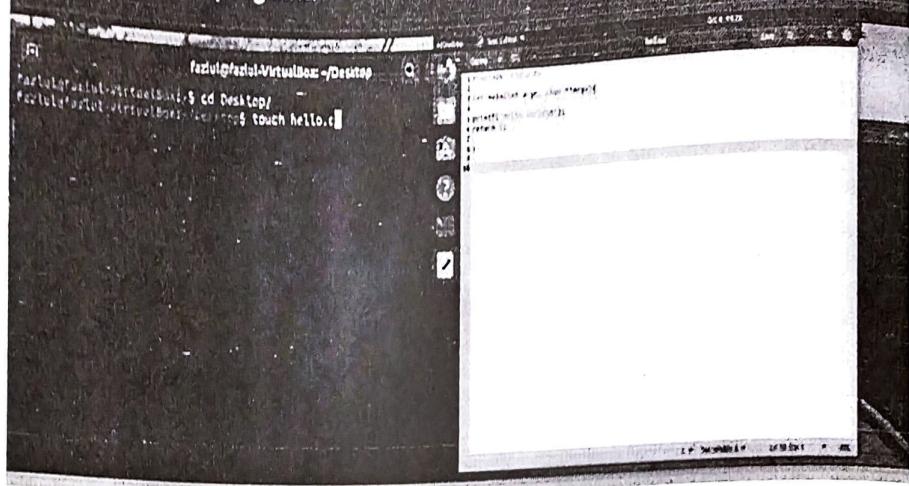
Success fully installed c compiler.

```
fastul@fastul-VirtualBox:~/Desktop$ sudo apt install build-essential
```

Create a C file using terminal

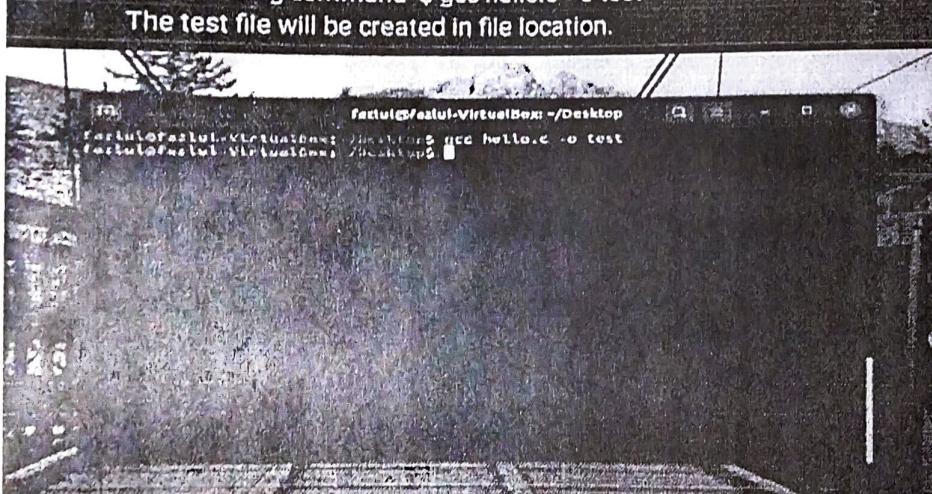
Type `$ touch hello.c` command in the terminal to create a C file.

Write the program.



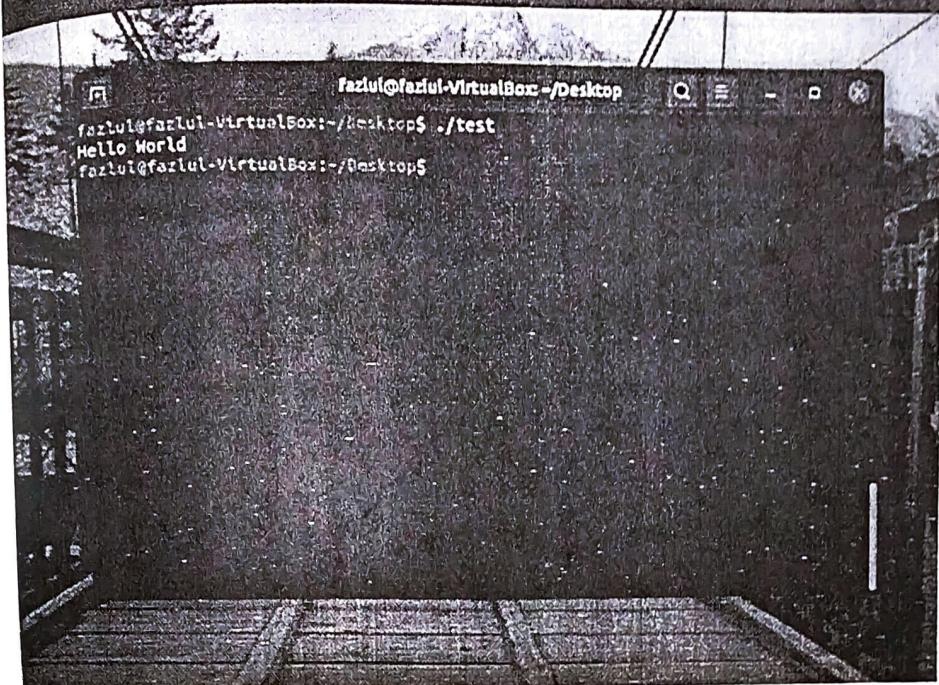
Create a test file to view the output from the terminal using the following command `$ gcc hello.c -o test`

The test file will be created in file location.



Compile the test file using this command `./test` to get the output.

Compile the test file using this command \$./test to get the output.



A screenshot of a terminal window titled "Fazlul@fazlul-VirtualBox ~ Desktop". The window shows the command "fazlul@fazlul-VirtualBox:~/Desktop\$./test" followed by the output "Hello World".

RESULT

Thus the program to install and run c compiler in ubuntu is done and output is obtained

```
Sudo apt update  
Sudo apt install build-essential  
Sudo apt-get install man pages-dev  
gcc--version  
nano hello.c  
gcc hello.c -o hello  
. /hello
```

Ex.No.10 SIMULATE CLOUD SCENARIO USING CLOUDSIM
211121

AIM:-

Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is ^{not} present in CloudSim.

PROCEDURE:-

After successful installation of CloudSim in your Eclipse, next part is to run a scheduling algorithm that is not present in CloudSim.

1. Choose any scheduling algorithm to run on Eclipse and Simulate on CloudSim, here ShortestJobFirst(SJF) algorithm is taken.
2. Create a Java File named ShortestJobFirst/ SJF
under the in-built package
`org.cloudbus.cloudsim.eaxmple` in the examples folder already present in the project and paste the SJF code for Simulation over the CloudSim.
3. Select SJF.java and run the application, building the files and simulation output will take few minutes.

This refers to the successful

simulation of our own scheduling algorithm simulated over the CloudSim.

RESULT

Thus a process to simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim

Ex.No. 11

21/1/21

Executing HELLO WORLD program in Google App Engine.

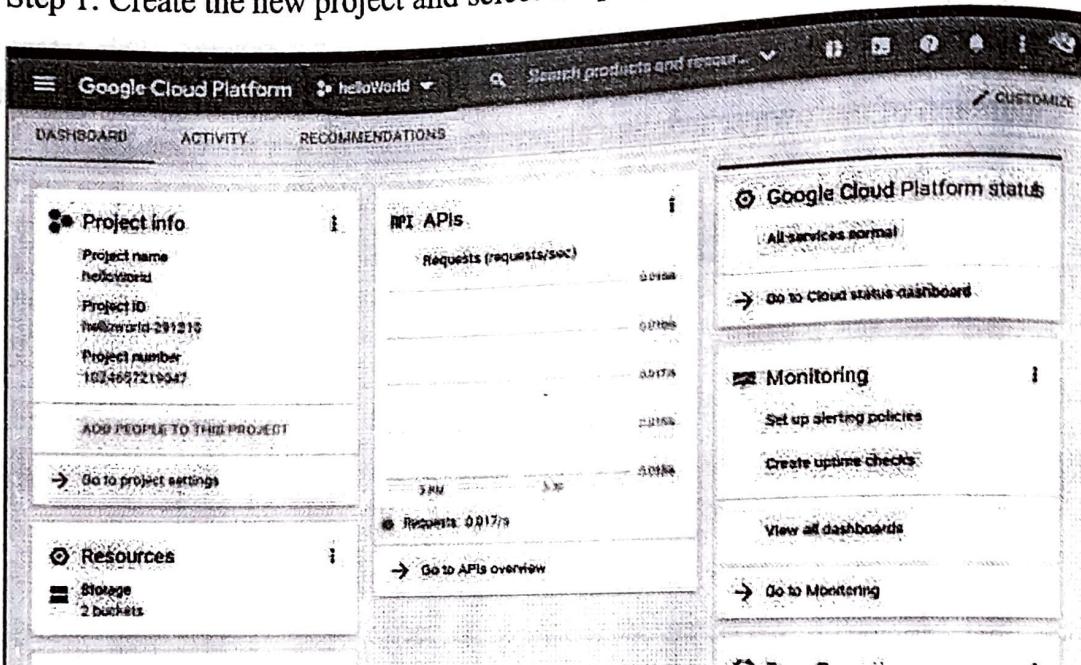
AIM:

To execute HELLO WORLD program in Google App Engine

PROCEDURE:

Step 0: visit <https://console.cloud.google.com/>

Step 1: Create the new project and select the project



Step 3: Click the cloud shell icon on TOP RIGHT CORNER

The screenshot shows the Google Cloud Platform dashboard for the project "HelloWorld". The "Project Info" section displays the project name "HelloWorld", project ID "helloworld-291910", and project number "102467319047". The "APIs" section shows requests for "https://www.googleapis.com/auth/cloud-platform" and "https://www.googleapis.com/auth/cloud-platform" with 0.00/s and 0.00/s rates respectively. The "Google Cloud Platform status" section indicates "All services normal" with a link to the "Go to Cloud status dashboard". The "Monitoring" section includes links for "Set Up Alerting policies" and "Create system checks". The "Resources" section has a link to "View all dashboards".

Step 4: Once the shell is open, enter the following commands:

1. gcloud app create
2. git clone <https://github.com/GoogleCloudPlatform/python-docs-samples>
3. cd python-docs-samples/appengine/standard_python3/hello_world
4. virtualenv --python python3 ~/envs/hello_world
5. source ~/envs/hello_world/bin/activate
6. pip install -r requirements.txt
7. python main.py

Step 5: Then click “web preview” and select “preview on port 8080”

The screenshot shows a browser window with the URL "ssh.cloud.google.com/cloudshell/editor?team_USA#cloudshell=true&shellonly=true". The main area is a "Cloud Shell" terminal window titled "Cloud Shell" with the command "gcloud app deploy" running. A context menu is open over the terminal, with the "Preview on port 8080" option highlighted. Other options in the menu include "Change port" and "About web preview".

Step 6: To Deploy the program, enter the following command:

```
gcloud app deploy app.yaml
```

RESULT

Thus the process of executing HELLO WORLD program in Google App Engine.