



SSM INSTITUTE OF ENGINEERING & TECHNOLOGY

Dindigul- Palani Highways, Dindigul – 624 002

Ph: 0451 – 2448800 – 2448899 (100 lines) Fax: 0451 – 2557755

E-mail: ssmiedgl@gmail.com, website: www.ssmiet.ac.in

**Approved by AICTE, New Delhi - Affiliated to Anna University, Chennai -
Accredited by NAAC**

PRACTICAL RECORD

BONAFIDE CERTIFICATE

Register Number:

Certified that this is the bonafide record of work done by Mr/Ms..... in
CS8711 CLOUD COMPUTING LABORATORY as prescribed by the Anna University
Chennai, for the seventh semester during the academic year 2021 – 2022.

Staff in Charge

Head of the Department

Submitted for the University Practical Examination held on.....at SSM Institute of
Engineering and Technology, Dindigul – 624 002.

Internal Examiner

External Examiner

CONTENTS

EX.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO	MARKS	SIGNATURE
VIRTUALIZATION					
1		Install Virtual Machine With Different Flavours			
2		Install A C Compiler In The Virtual Machine			
GOOGLE APP ENGINE					
3		Install Google App Engine And Create Hello World App			
CLOUD COMPUTING					
4		To run the virtual machine of different configuration			
5		To attach virtual block in the virtual machine			
6		Virtual machine migration from one node to the other			
7		To install storage controller and interact with it			
HADOOP					
8		Setting one node hadoop cluster			
9		Word count program to demonstrate map and reduce task			

VIRTUALIZATION

Ex.No.1

INSTALL VIRTUAL MACHINE WITH DIFFERENT FLAVOURS

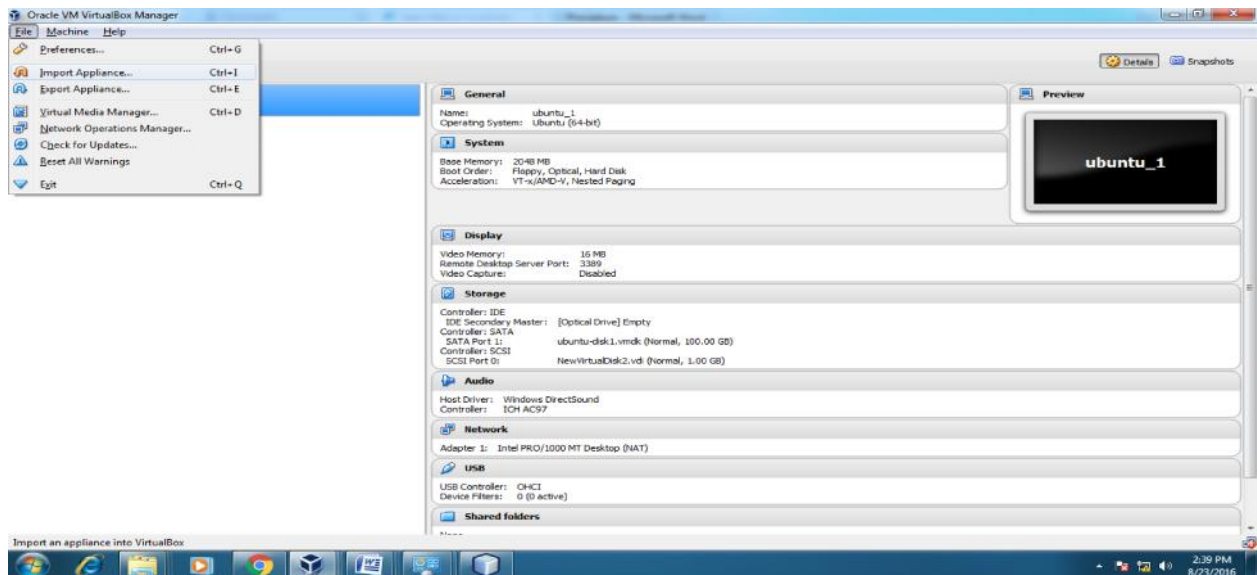
AIM:

To Install virtual machine with different flavours of linux or windows

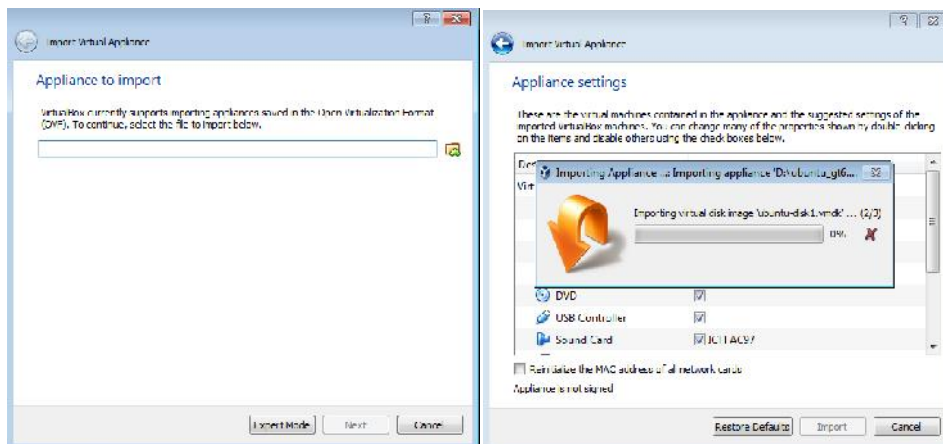
PROCEDURE:

Step 1: Open the Oracle VM VirtualBox Manager

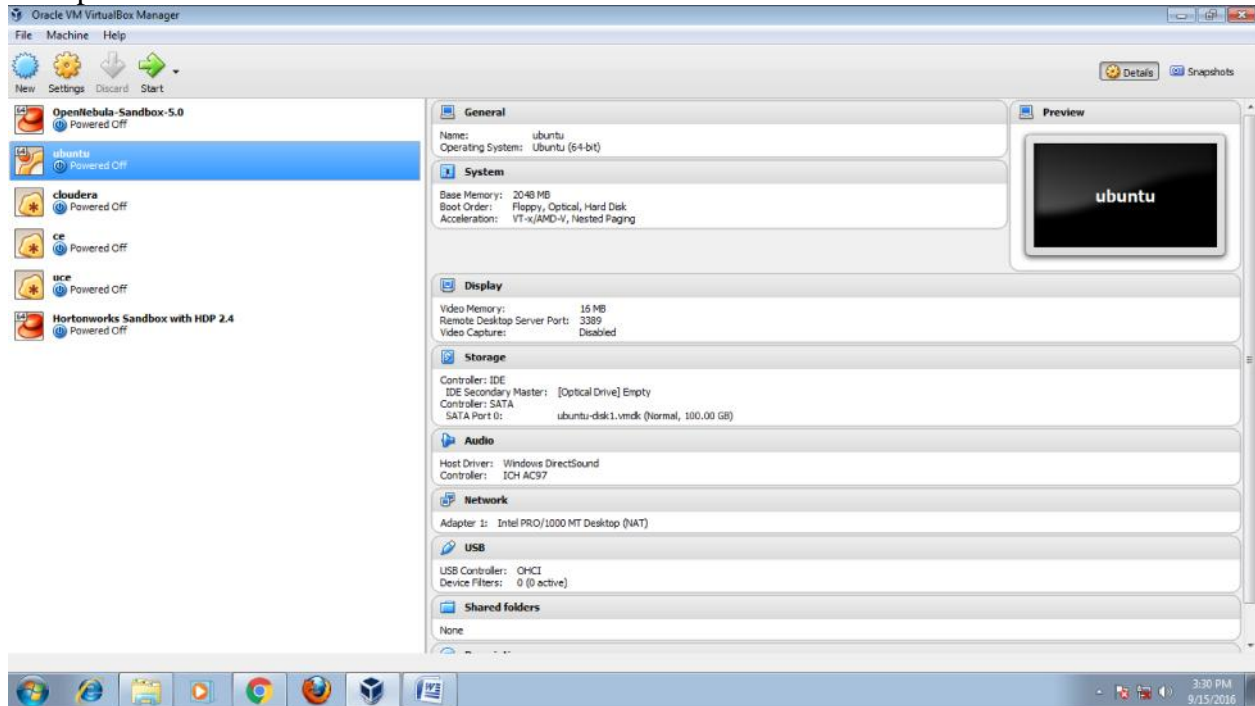
Step 2: 2.1 Open File → Appliance



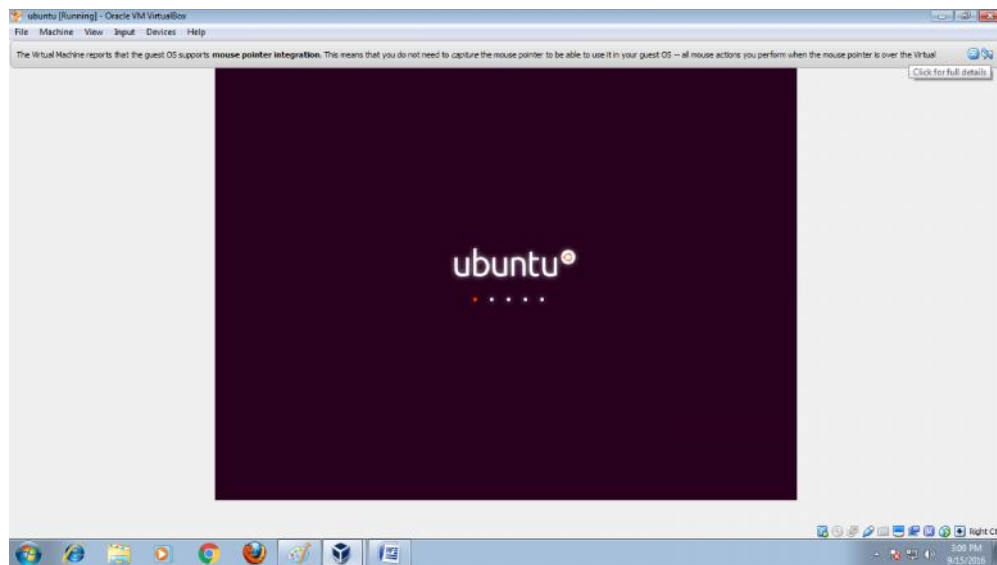
2.2 Browse ubuntu-gt6 file



Step 3: Start the Ubuntu VM



Output



RESULT:

Thus the virtual machine has been installed successfully.

Ex.No.2

INSTALL A C COMPILER IN THE VIRTUAL MACHINE

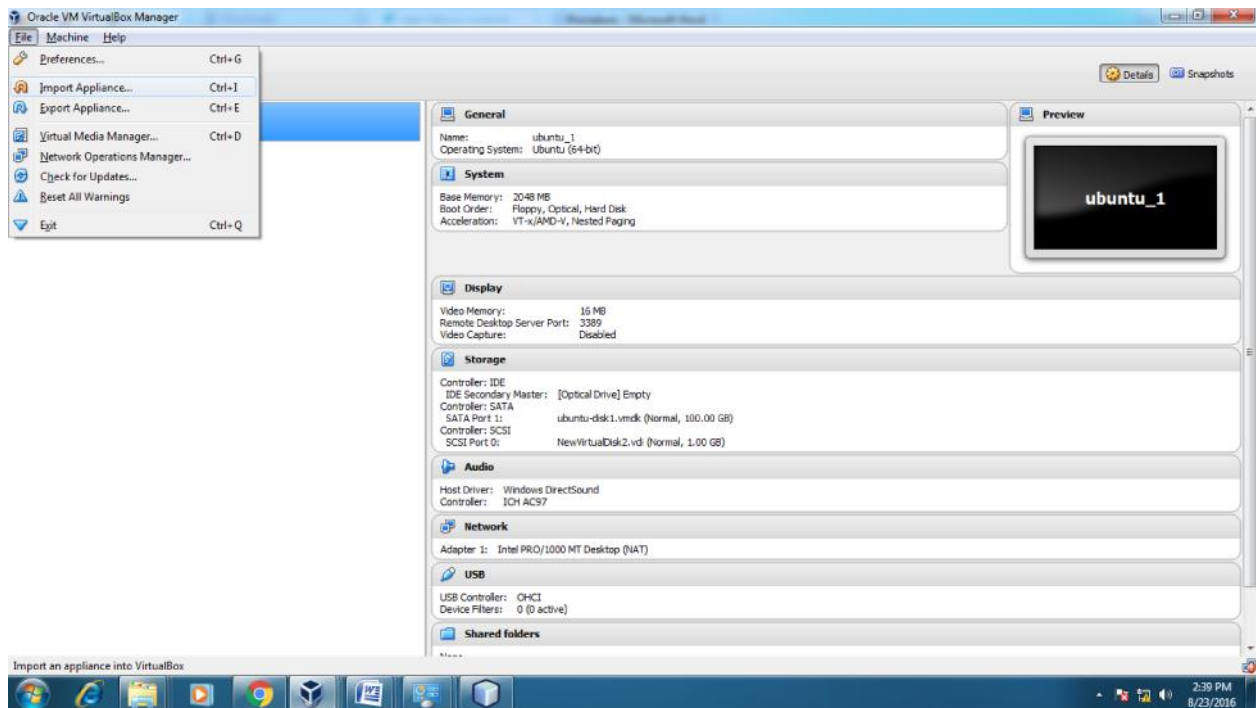
AIM:

To Install a C compiler in the virtual machine and execute a sample program. (Working on ubuntu-gt6)

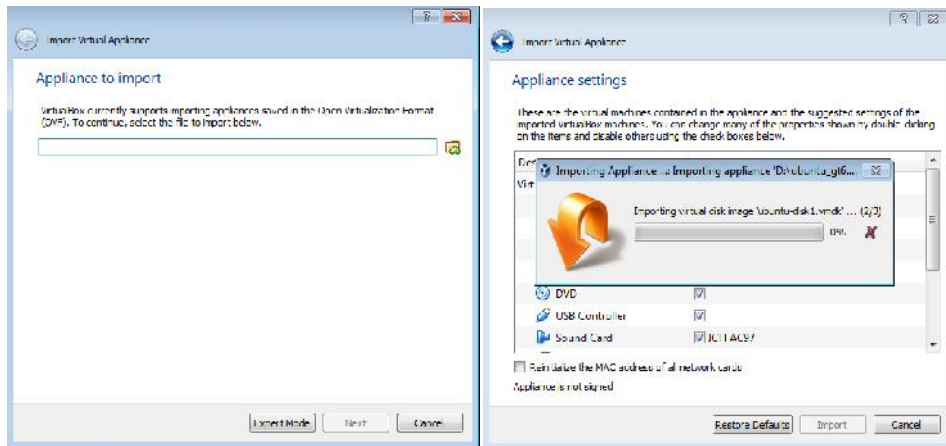
PROCEDURE:

Step 1: Open the Oracle VM VirtualBox Manager

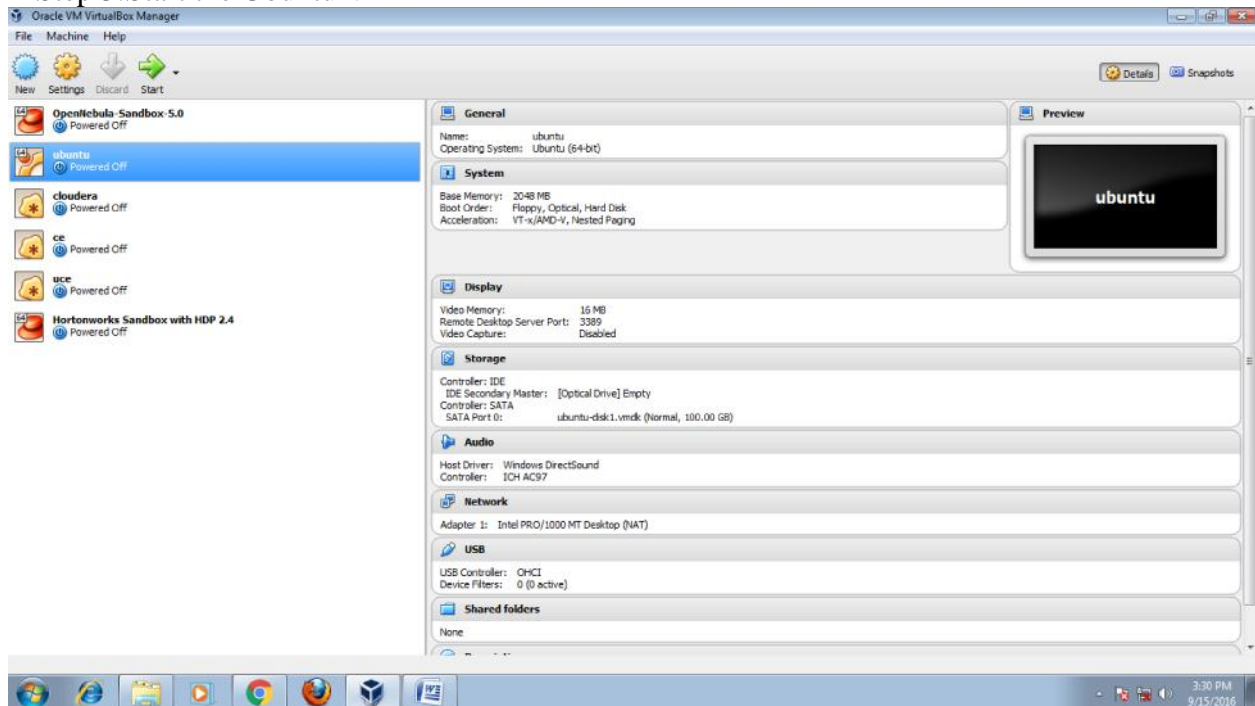
Step 2: 2.1 Open File → Appliance



2.2 Browse ubuntu-gt6 file



Step 3: Start the Ubuntu VM

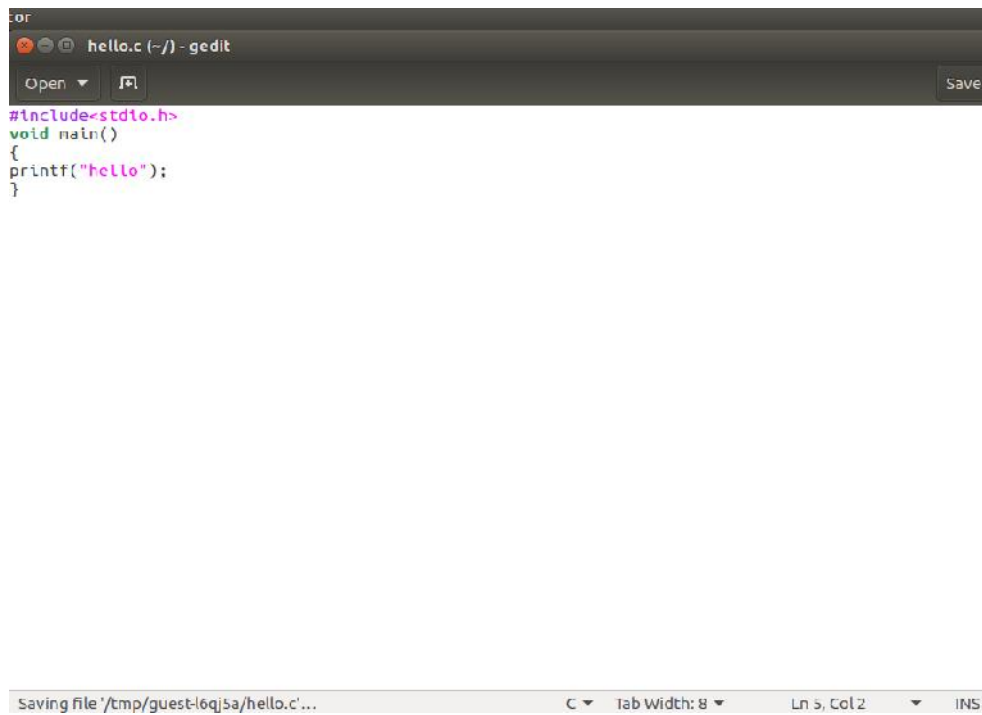


Step 4: After Running the Ubuntu VM

Step 5: Open the terminal. If gcc is available goto Step5 else install gcc by using following commands

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get update
sudo apt-get install gcc-6 gcc-6-base
```

Step 6: Type `gedit hello.c` in terminal. Type the C program in it.



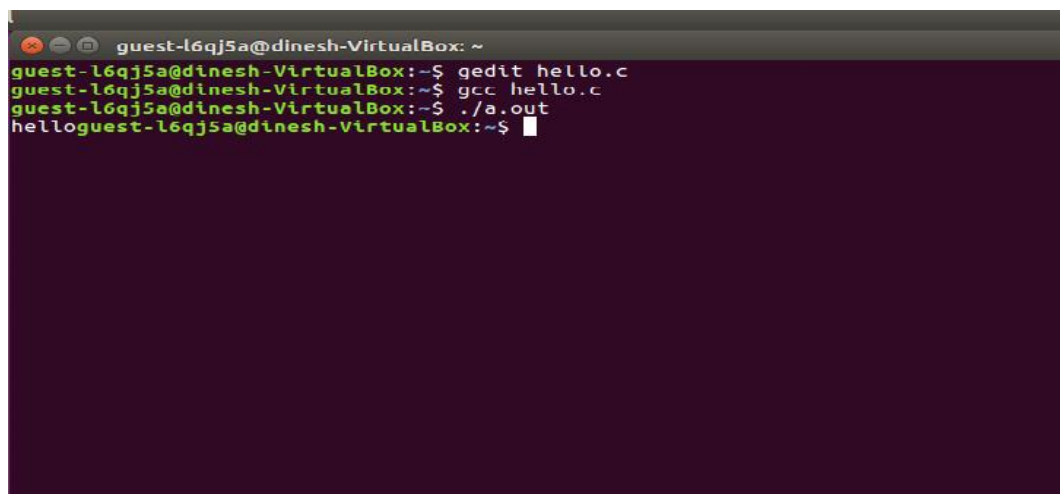
The screenshot shows a gedit editor window titled 'hello.c (~/) - gedit'. The code inside is a simple C program:

```
#include<stdio.h>
void main()
{
    printf("hello");
}
```

Below the editor window, a status bar indicates 'Saving file '/tmp/guest-l6qj5a/hello.c'...' and other settings like 'Tab Width: 8' and 'Ln 5, Col 2'.

Step 7: Type *gcc hello.c*

Step 8: Type *./a.out* to get the output



The screenshot shows a terminal window with the following commands and output:

```
guest-l6qj5a@dinesh-VirtualBox: ~
guest-l6qj5a@dinesh-VirtualBox:~$ gedit hello.c
guest-l6qj5a@dinesh-VirtualBox:~$ gcc hello.c
guest-l6qj5a@dinesh-VirtualBox:~$ ./a.out
hello
guest-l6qj5a@dinesh-VirtualBox:~$
```

RESULT:

Thus the C compiler in the virtual machine has been installed and program executed successfully.

GOOGLE APP ENGINE

Ex.No.3

INSTALL GOOGLE APP ENGINE AND CREATE HELLO WORLD APP

Aim

To Install Google App Engine and Create hello world web applications using python.

Procedure

Step:1 Go to the below URL and **create** new project

<https://console.cloud.google.com/projectcreate?previousPage=%2Fappengine%2Fstart%3Fproject%3Dfinalcsessm&folder=&organizationId=0>

Google Cloud Platform Search products and resources

New Project

You have 9 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *
ExNo3CC

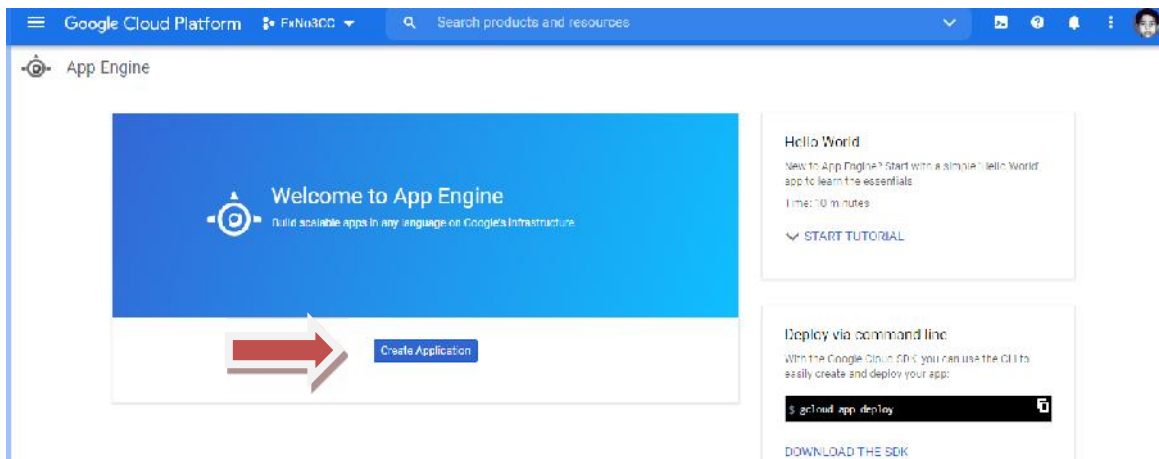
Project ID: exno3cc. It cannot be changed later. [EDIT](#)

Location *
No organisation [BROWSE](#)

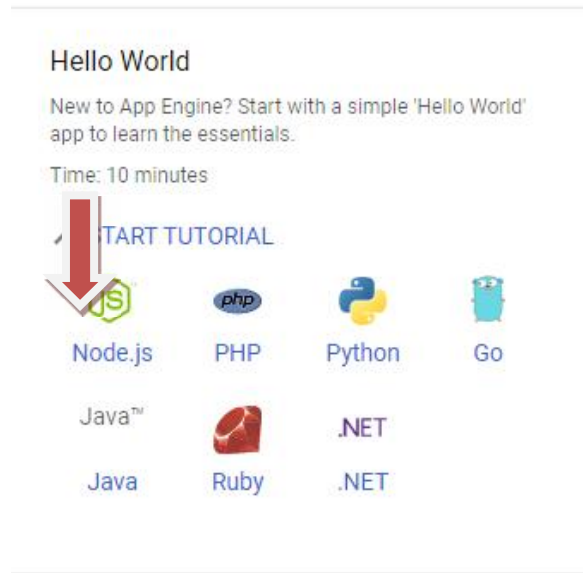
Parent organisation or folder

[CREATE](#) [CANCEL](#)

Step:2 Select create application



Step 3: Select the language **Python**



Step 4: Follow the App Engine Quickstart wizard to deploy application using gcloud command.

Step 5:

Project setup

GCP organizes resources into projects, which collect all of the related resources for a single application in one place.

Begin by creating a new project or selecting an existing project for this tutorial.

Select a project, or [create a new one](#)

 ExNo3CC

For details, see [Creating a project](#).

Previous

Step 1 of 8

Next

Project setup-GCP organizes resources into projects, which collect all of the related resources for a single application in one place.


Step:6

Open Cloud Shell

Using Cloud Shell

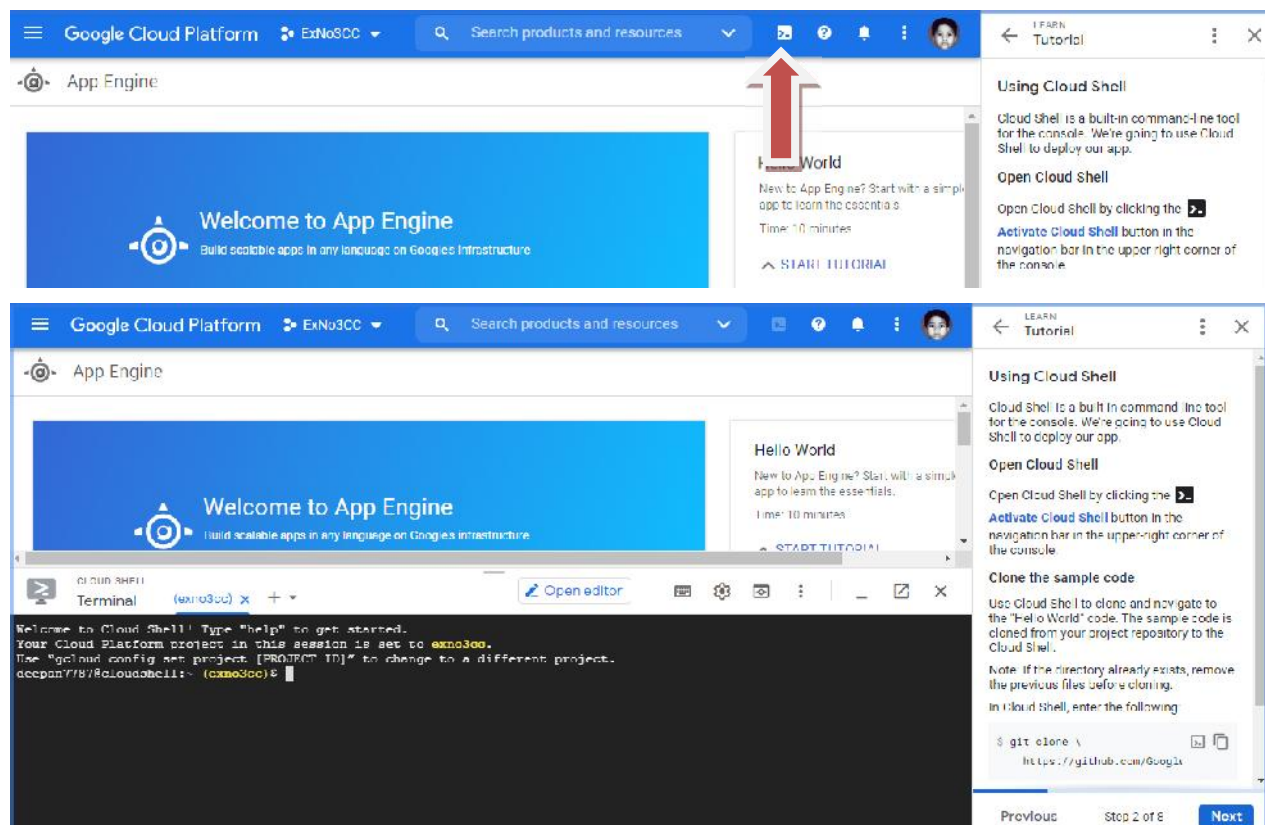
Cloud Shell is a built-in command-line tool for the console. We're going to use Cloud Shell to deploy our app.

Open Cloud Shell

Open Cloud Shell by clicking the 

Activate Cloud Shell button in the navigation bar in the upper-right corner of the console.

Open Cloud Shell by clicking the **Activate Cloud Shell** button in the navigation bar in the upper-right corner of the console.



Step:7

In Cloud Shell, enter the following:

```
git clone \ https://github.com/GoogleCloudPlatform/python-docs-samples
```

Then, switch to the tutorial directory:

```
cd \ python-docs-samples/appengi
```

A screenshot of a Cloud Shell terminal window. The window has a title bar with 'CLOUD SHELL' and 'Terminal (exno3cc)'. Below the title bar, there's a toolbar with icons for 'Open editor', settings, and other functions. The terminal output shows a welcome message, project information, and the execution of the following commands:

```
deepan7787@cloudshell:~ (exno3cc)$ git clone https://github.com/GoogleCloudPlatform/python-docs-samples
fatal: destination path 'python-docs-samples' already exists and is not an empty directory.
deepan7787@cloudshell:~ (exno3cc)$ cd python-docs-samples/appengine/standard_python3/hello_world
deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$
```

Step:8

Configuring your **main directory** for deployment

Exploring the application

Enter the command to view your application code:

```
cat main.py
```

The Python application that uses the [Flask](#) web framework. This Python app responds to a request with an HTTP header and the message `Hello World!`.

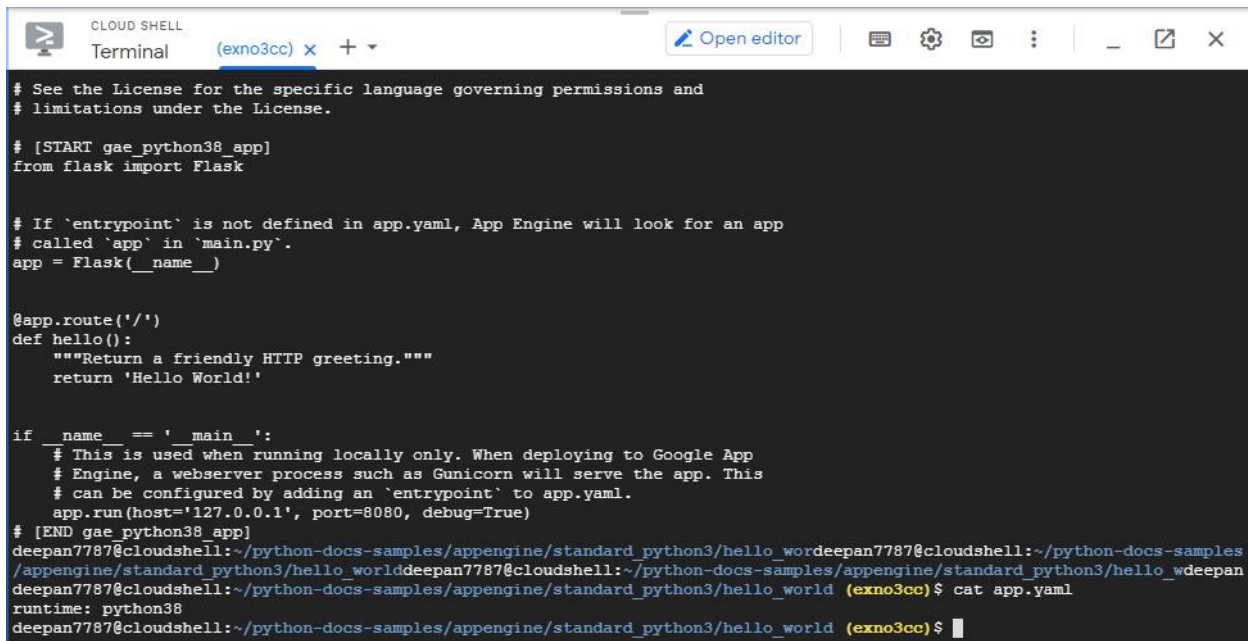
Exploring your configuration

App Engine uses YAML files to specify a deployment's configuration.

`app.yaml` files contain information about your application, like the runtime environment, environment variables, and more.

Enter the command to view your configuration file:

```
cat app.yaml
```



```
CLOUD SHELL
Terminal (exno3cc) x + - Open editor

# See the License for the specific language governing permissions and
# limitations under the License.

# [START gae_python38_app]
from flask import Flask

# If 'entrypoint' is not defined in app.yaml, App Engine will look for an app
# called 'app' in 'main.py'.
app = Flask(__name__)

@app.route('/')
def hello():
    """Return a friendly HTTP greeting."""
    return 'Hello World!'

if __name__ == '__main__':
    # This is used when running locally only. When deploying to Google App
    # Engine, a webserver process such as Gunicorn will serve the app. This
    # can be configured by adding an 'entrypoint' to app.yaml.
    app.run(host='127.0.0.1', port=8080, debug=True)
# [END gae_python38_app]

deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world$ cat app.yaml
runtime: python38
deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc) $
```

Setp:9 Testing your app

Test your app on Cloud Shell

Cloud Shell test your app before deploying to make sure it's running as intended, just like debugging on your local machine.

To test your app,

- ✓ first create an isolated virtual environment.
- ✓ This ensures that your app does not interfere with other Python applications that may be available on your system.

```
virtualenv --python python3 \ ~/envs/hello_world
```

Activate your newly created virtual environment:

```
source \ ~/envs/hello_world/bin/activate
```

Use `pip` to install project dependencies. This "Hello World" app depends on the Flask microframework:

```
pip install -r requirements.txt
```

Finally, run your app in Cloud Shell using the Flask development server:

```
python main.py
```



```

/appengine/standard_python3/hello_worlddeepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_wdeepan
deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ cat app.yaml
runtime: python38
deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ virtualenv --python python3 ~
/envs/hello_world
created virtual environment CPython3.7.3.final.0-64 in 1034ms
creator CPython3Posix(dest=/home/deepan7787/envs/hello_world, clear=False, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/deepan7787/.
local/share/virtualenv)
added seed packages: Flask==1.1.2, Jinja2==2.11.2, MarkupSafe==1.1.1, Werkzeug==1.0.1, click==7.1.2, itsdangerous==1.1.
0, pip==20.2.2, pip==20.2.3, setuptools==49.6.0, setuptools==50.3.0, wheel==0.35.1
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ source ~/envs/hello_world/bin
/activate
(hello_world) deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ pip install -r
requirements.txt
Requirement already satisfied: Flask==1.1.2 in /home/deepan7787/envs/hello_world/lib/python3.7/site-packages (from -r requi
rements.txt (line 1)) (1.1.2)
Requirement already satisfied: itsdangerous>=0.24 in /home/deepan7787/envs/hello_world/lib/python3.7/site-packages (from Fl
ask==1.1.2->-r requirements.txt (line 1)) (1.1.0)
Requirement already satisfied: Jinja2>=2.10.1 in /home/deepan7787/envs/hello_world/lib/python3.7/site-packages (from Flask=
=1.1.2->-r requirements.txt (line 1)) (2.11.2)
Requirement already satisfied: click>=5.1 in /home/deepan7787/envs/hello_world/lib/python3.7/site-packages (from Flask==1.1
.2->-r requirements.txt (line 1)) (7.1.2)
Requirement already satisfied: Werkzeug>=0.15 in /home/deepan7787/envs/hello_world/lib/python3.7/site-packages (from Flask=
=1.1.2->-r requirements.txt (line 1)) (1.0.1)
Requirement already satisfied: MarkupSafe>=0.23 in /home/deepan7787/envs/hello_world/lib/python3.7/site-packages (from Jinj
a2>=2.10.1->Flask==1.1.2->-r requirements.txt (line 1)) (1.1.1)
(hello_world) deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ █

(hello_world) deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ python main.py
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 216-565-103

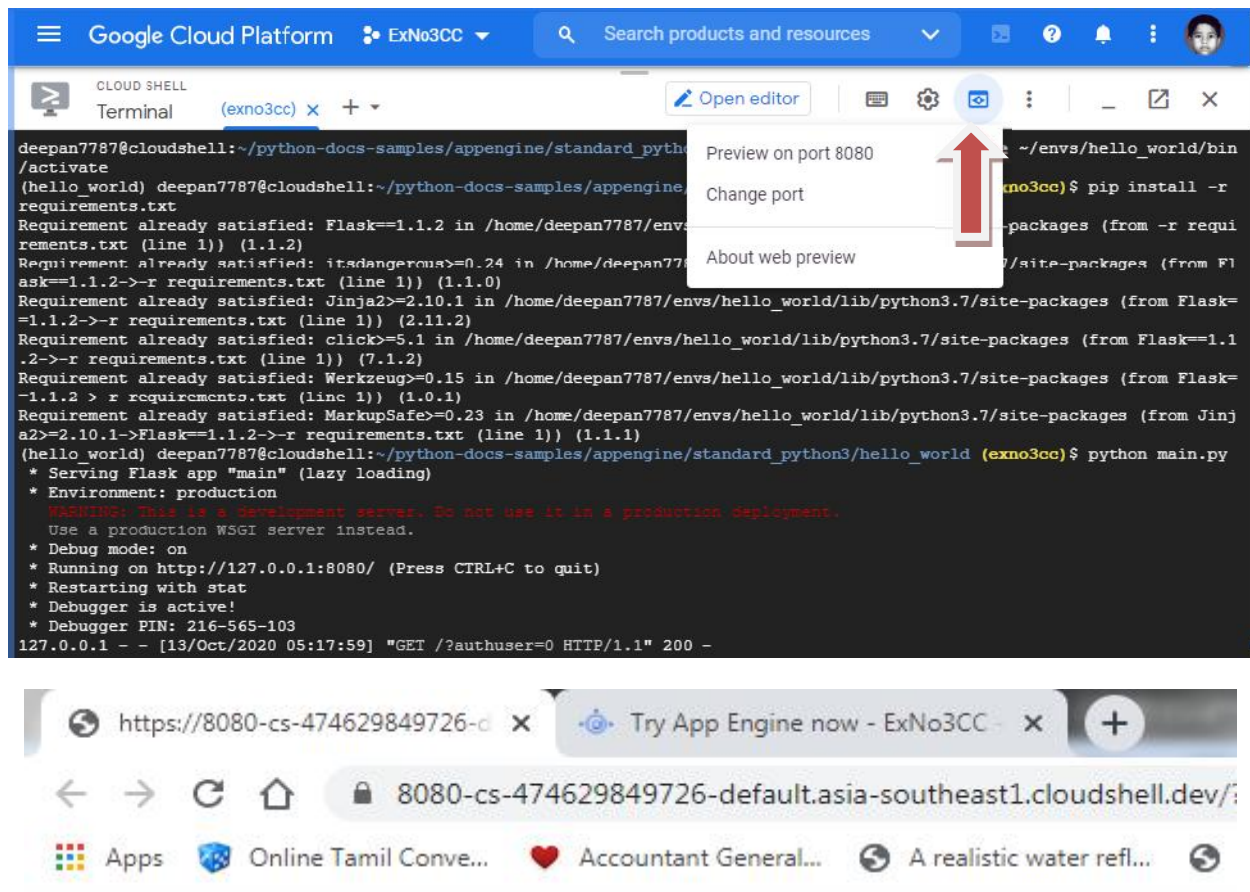
```

Preview your app with "Web preview"

- ✓ Your app is now running on Cloud Shell.
- ✓ You can access the app by clicking the **Web preview** button at the top of the Cloud Shell pane and choosing **Preview on port 8080**.

Terminating the preview instance

Terminate the instance of the application by pressing `Ctrl+C` in the Cloud Shell.



Step:10 Deploying to App Engine

In order to deploy your app, you need to create an **app in a region**:

```
gcloud app create
```

Authorise Cloud Shell

gcloud is requesting your credentials to make a GCP API call.

Click to authorize this and future calls that require your credentials.

Authorise

Reject

Step:11

Deploying with Cloud Shell

To deploy your app enter the following:

```
gcloud app deploy app.yaml \ --project exno3cc
```

```
(hello_world) deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$ gcloud app deploy app.yaml --project exno3cc
You are creating an app for project [exno3cc].
WARNING: Creating an App Engine application for a project is irreversible and the region cannot be changed. More information about regions is at <https://cloud.google.com/appengine/docs/locations>.
```

```
Please choose the region where you want your App Engine application located:
```

```
[1] asia-east2
[2] asia-northeast1
[3] asia-northeast2
[4] asia-northeast3
[5] asia-south1
[6] asia-southeast2
[7] australia-southeast1
[8] europe-west
[9] europe-west2
[10] europe-west3
[11] europe-west6
[12] northamerica-northeast1
[13] southamerica-east1
[14] us-central
[15] us-east1
[16] us-east4
[17] us-west2
[18] us-west3
[19] us-west4
[20] cancel
```

```
Please enter your numeric choice: 5
```

```
Creating App Engine application in project [exno3cc] and region [asia-south1]....
```

```
Please enter your numeric choice: 5
```

```
Creating App Engine application in project [exno3cc] and region [asia-south1]....done.
Services to deploy:
```

```
descriptor:    [/home/deepan7787/python-docs-samples/appengine/standard_python3/hello_world/app.yaml]
source:        [/home/deepan7787/python-docs-samples/appengine/standard_python3/hello_world]
target project: [exno3cc]
target service: [default]
target version: [20201013t052416]
target url:     [https://exno3cc.el.r.appspot.com]
```

```
Do you want to continue (Y/n)? y
```

```
Beginning deployment of service [default]...
```

```
Uploading 6 files to Google Cloud Storage
```

```
File upload done.
```

```
Updating service [default]...failed.
```

```
ERROR: (gcloud.app.deploy) Error Response: [7] Access Not Configured. Cloud Build has not been used in project exno3cc before or it is disabled. Enable it by visiting https://console.developers.google.com/apis/api/cloudbuild.googleapis.com/overview?project=exno3cc then retry. If you enabled this API recently, wait a few minutes for the action to propagate to our systems and retry.
```

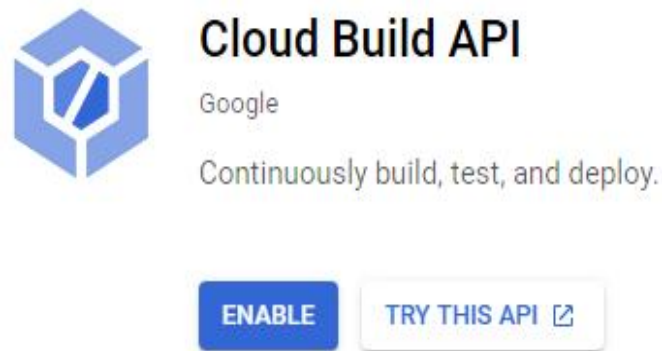
```
(hello_world) deepan7787@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world (exno3cc)$
```

Visit your app with this link

Congratulations! Your app has been deployed. The default URL of your app is a subdomain on appspot.com that starts with your project's ID: exno3cc.appspot.com.

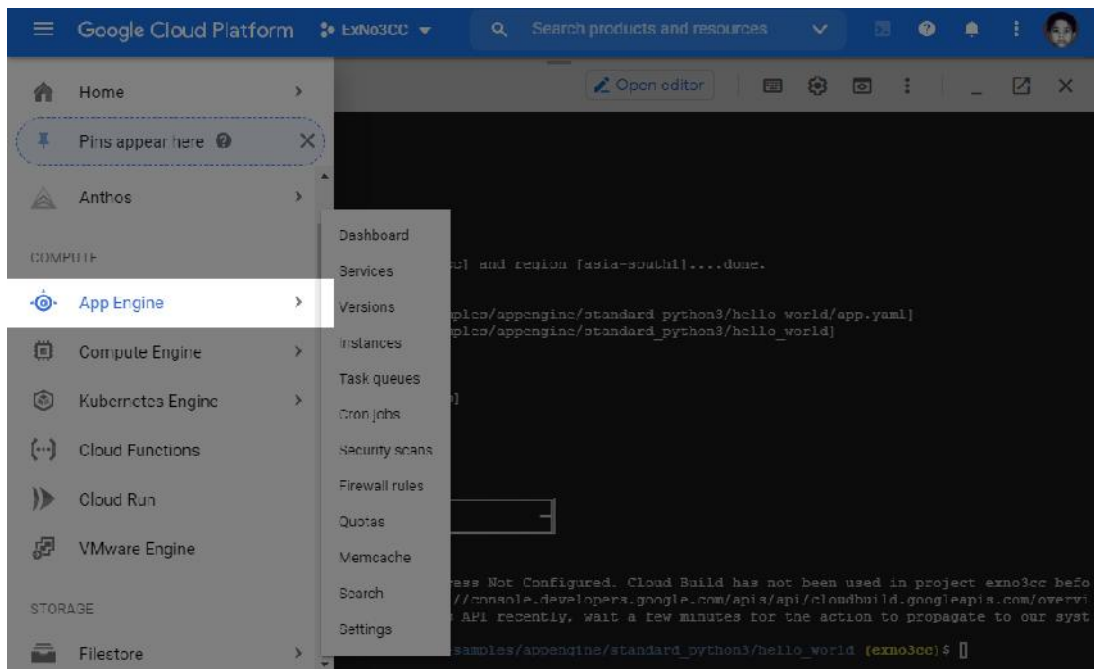
Try visiting your deployed application.

Step:12 If you find any error, need to enable Cloud Build API



Step:13

View your app's status **by monitoring its status on the App Engine dashboard.**



Step:14 Which is used for additional setting for our web application.

Disable your application

1. Go to the [Settings](#) page.
2. Click [Disable Application](#).

This is sufficient to stop billing from this app. More details on the relationship between apps and projects and how to manage each can be found [here](#).

Delete your project

If you would like to completely delete the app, you must delete the project in the **Manage resources** page. This is not reversible, and any other resources you have in your project will be destroyed:

1. Go to



2. Click [Settings](#).
3. Click [Shut down](#).

Step:15 Finally the Web application was created and deployed in the Google Cloud using Google App Engine.

Conclusion



You have successfully deployed an App Engine application!

RESULT:

Thus the installation of Google App Engine was done successfully and also a hello world web application was created the deployed using python.

CLOUD COMPUTING

Ex.No.4

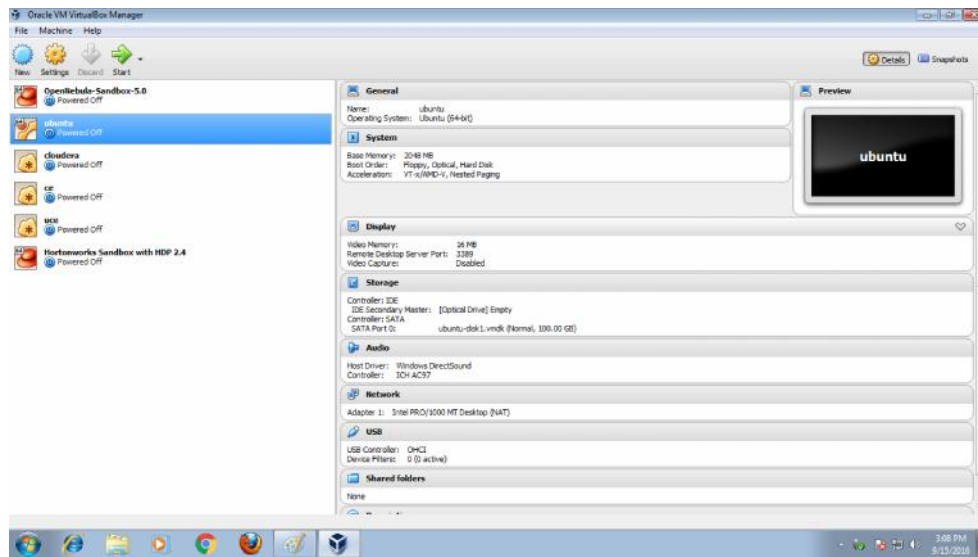
TO RUN THE VIRTUAL MACHINE OF DIFFERENT CONFIGURATION

AIM:

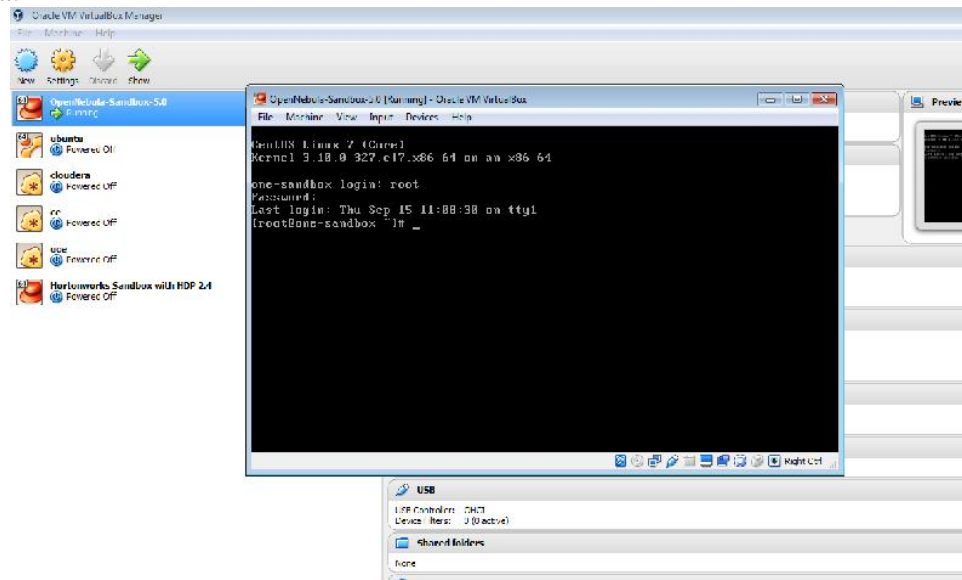
To find the procedure to run the virtual machine of different configuration and to check how many virtual machines can be utilized at particular time.

PROCEDURE:

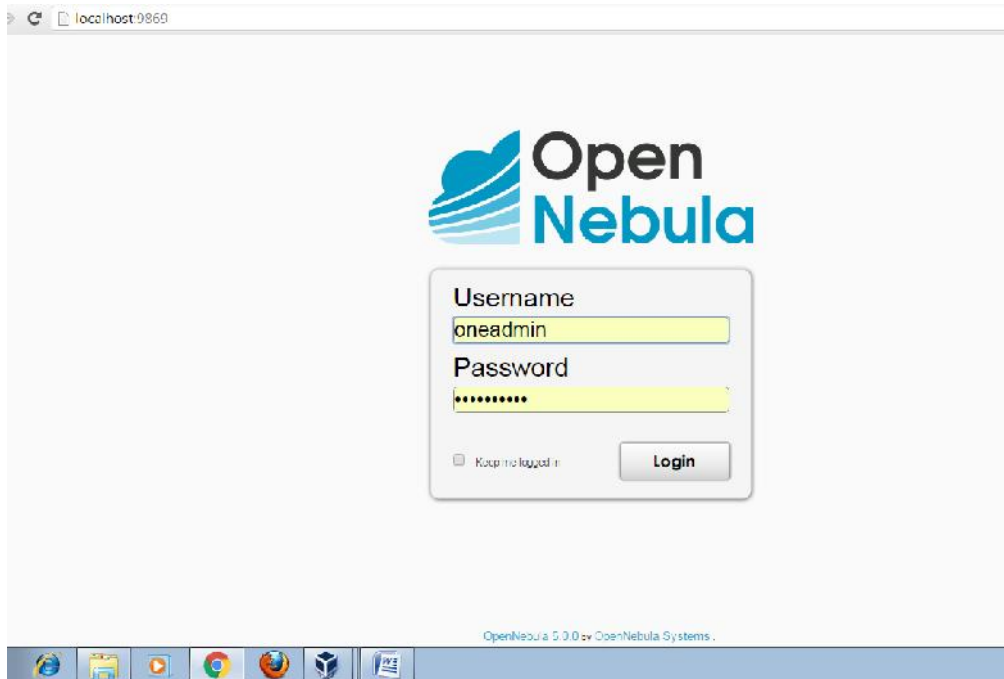
Step 1: Open the Oracle VM VirtualBox Manager.



Step 2: Start the Opennebula Sandwich 5.0 and log in with the username as root and password as opennebula.

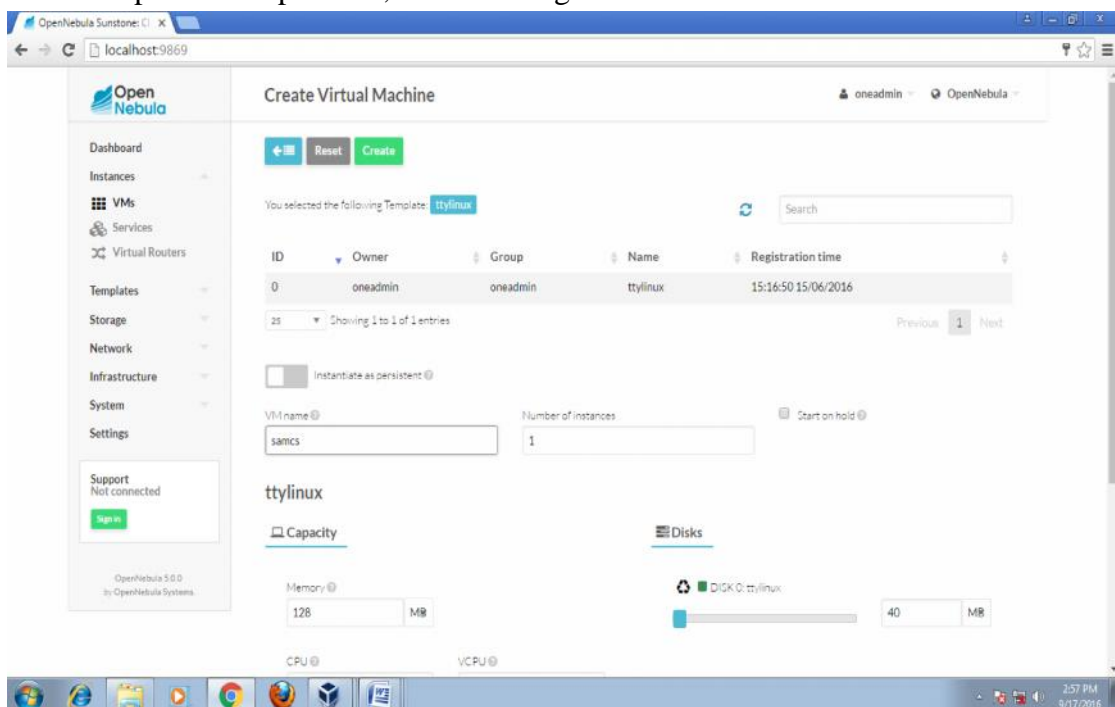


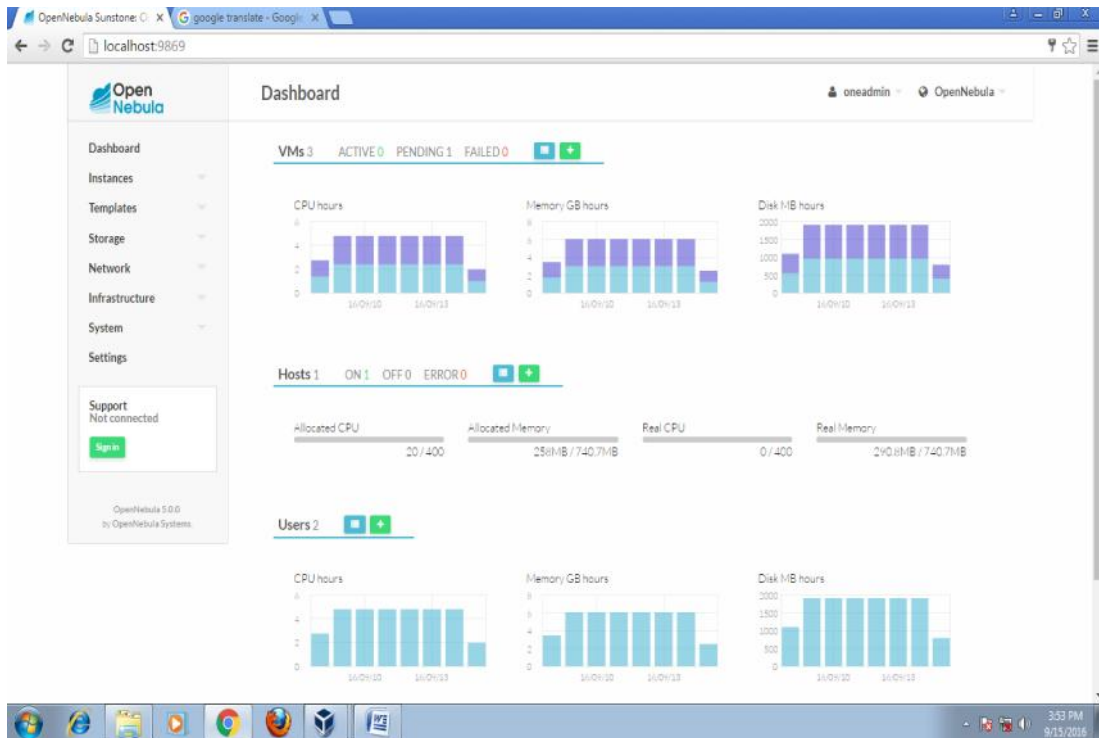
Step 3: Open Browser, type localhost:9869 and Login using username as oneadmin, password as opennebula.



Step 4: Click on instances, select VMs then follow the steps to create Virtual machine

- Expand the + symbol
- Select user oneadmin
- Then enter the VM name, no. of instance, cpu.
- Then click on create button.
- Repeat the steps the C,D for creating more than one VMs.





Step 5: When status changed from pending to running then click on the blue icon for both template.

Step 6: Give the login as root and password as opennebula for both the template.

RESULT:

Thus the virtual machine has been run on different configuration and number of virtual machine utilized at a particular time was checked.

Ex.No.5

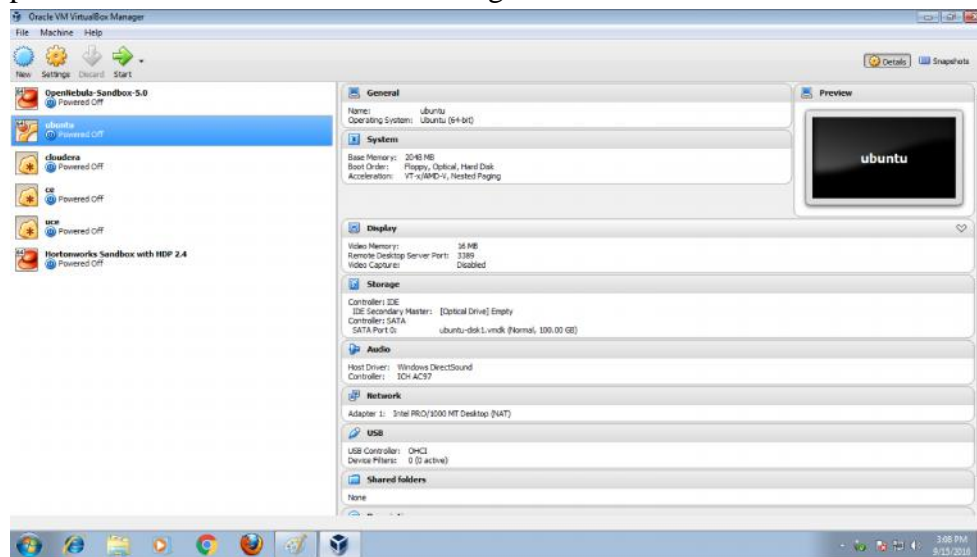
TO ATTACH VIRTUAL BLOCK TO THE VIRTUAL MACHINE

AIM:

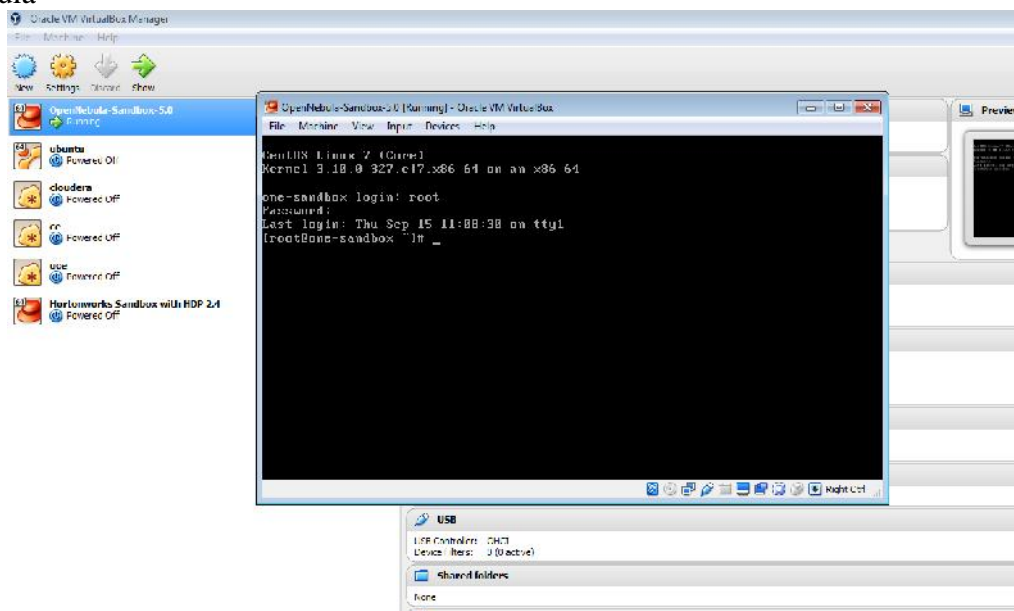
To find the procedure to attach virtual block to the virtual machine and check whether it holds the data even after the release of the virtual machine.

PROCEDURE:

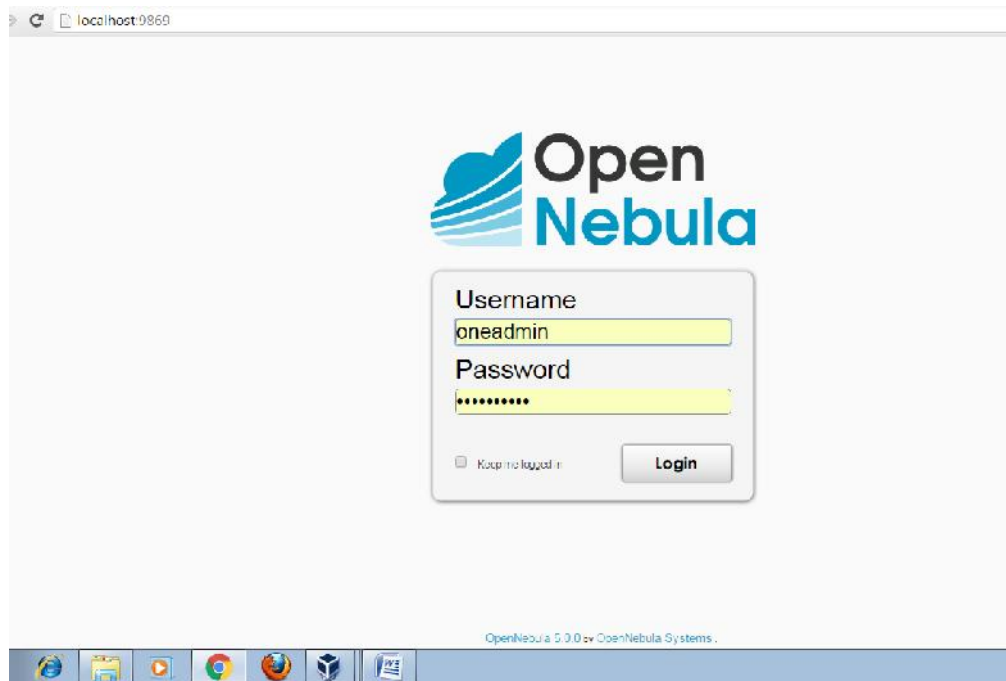
Step 1: Open the Oracle VM VirtualBox Manager.



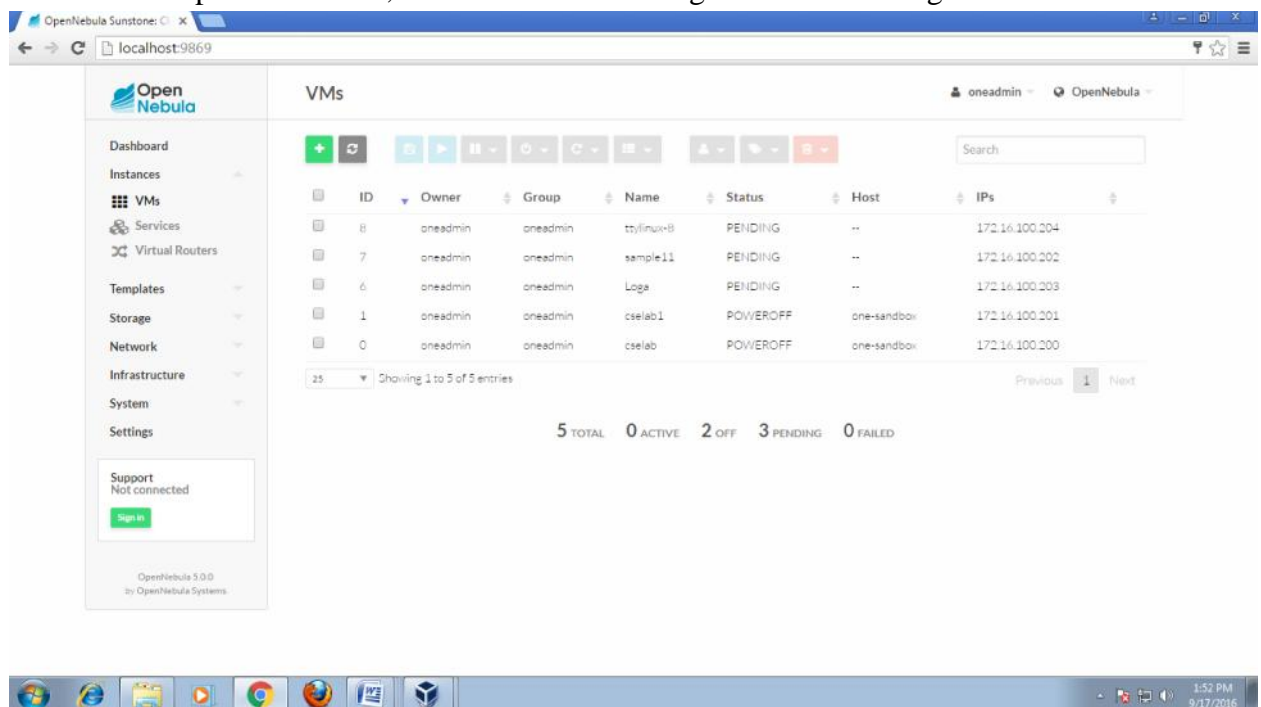
Step 2: Start the Opennebula Sandwich 5.0 and log in with the username root and password as opennebula

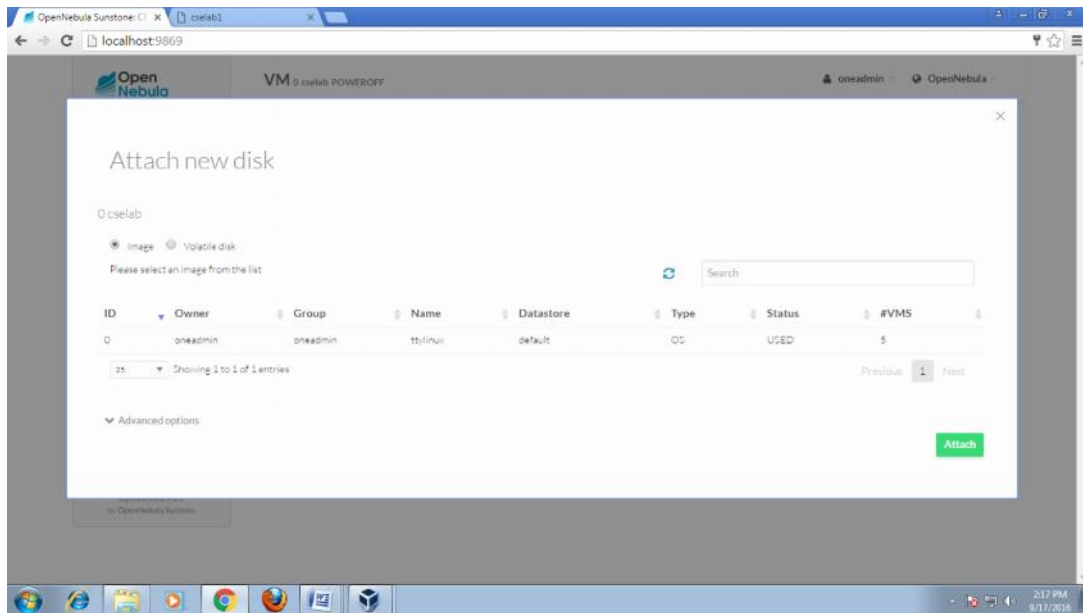


Step 3: Open Browser, type localhost:9869 and Login using username: oneadmin, password: opennebula



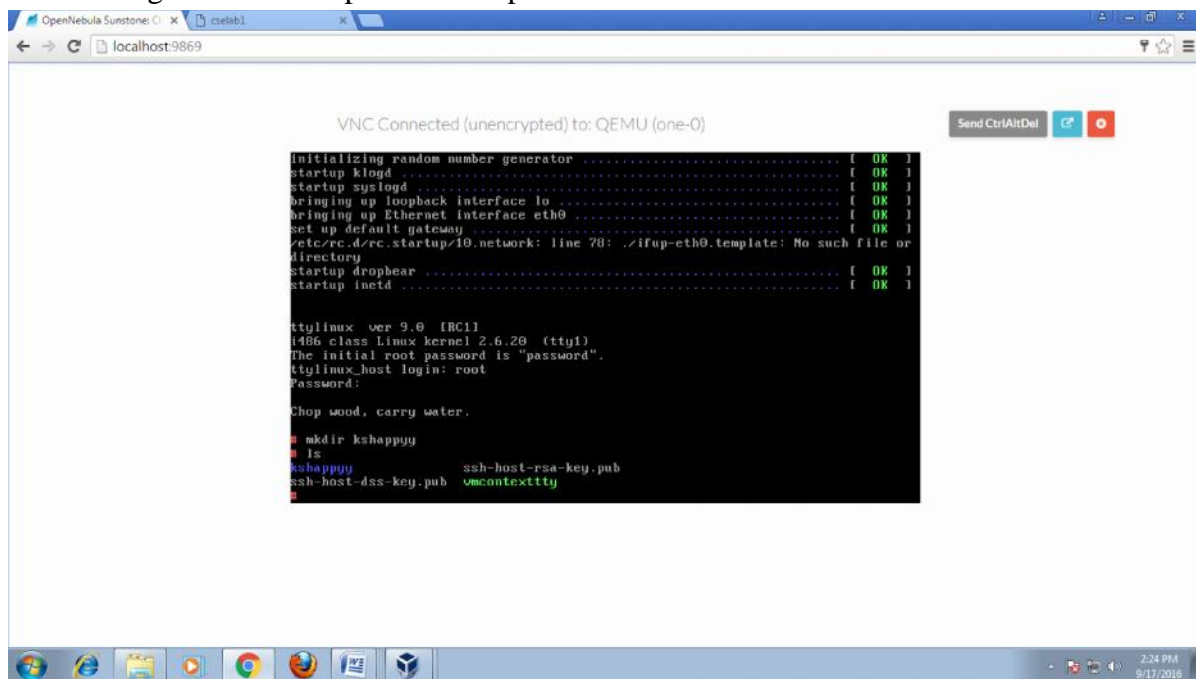
Step 4: Power off one particular VM , attach disk in the storage and choose image





Step 5: Boot the VM and wait for RUNNING status

Step 6: Click the right corner blue icon and the popup window is displayed where we need to enter the login as root and password as password.



RESULT:

Thus the procedure to attach virtual block to the virtual machine and check whether it holds the data even after the release of the virtual machine was written, executed and the output was verified successfully.

Ex.No.6

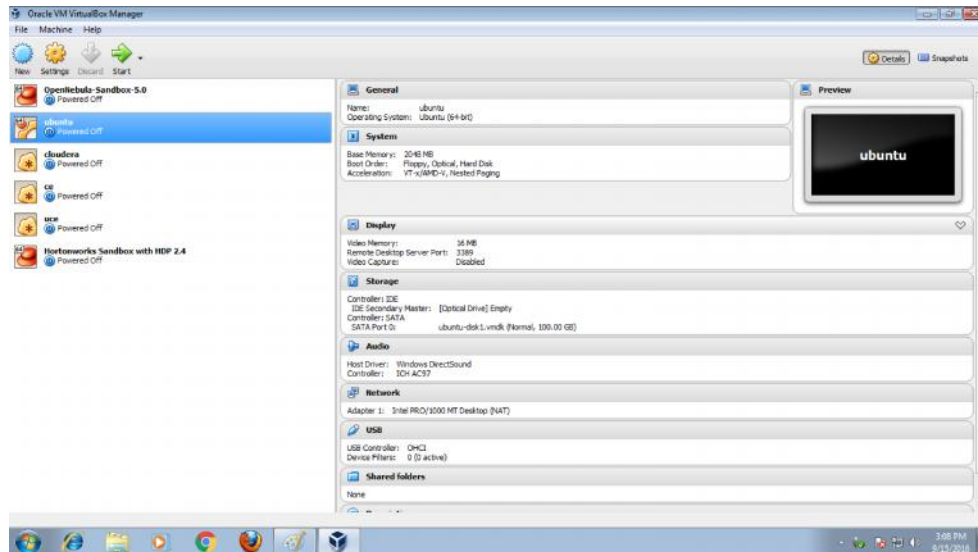
VIRTUAL MACHINE MIGRATION FROM ONE NODE TO THE OTHER

AIM:

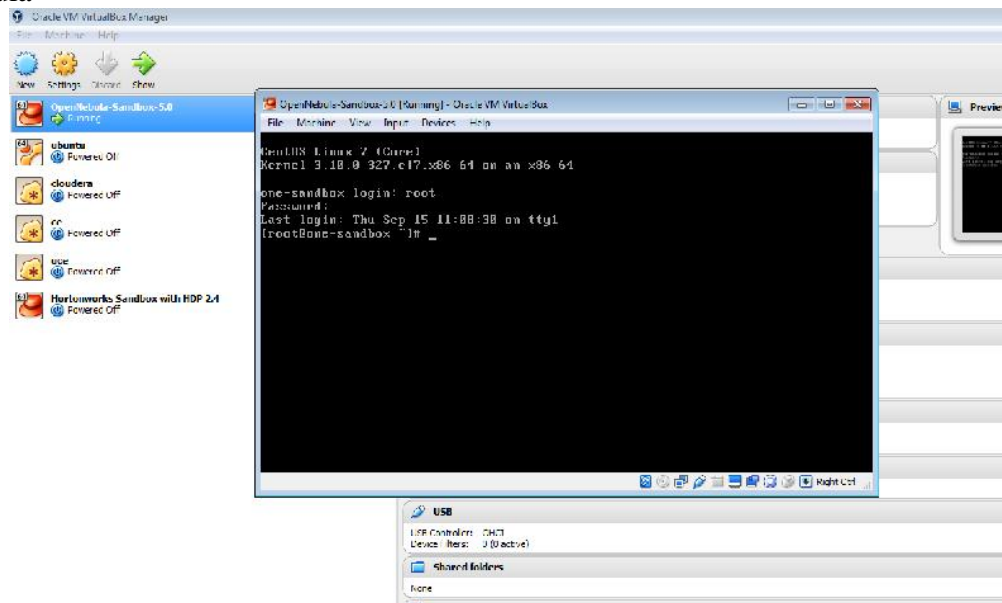
To show the virtual machine migration based on the certain condition from one node to the other.

PROCEDURE:

Step 1: Open the Oracle VM VirtualBox Manager.



Step 2: Start the Opennebula Sandwich 5.0 and log in with the username root and password as opennebula

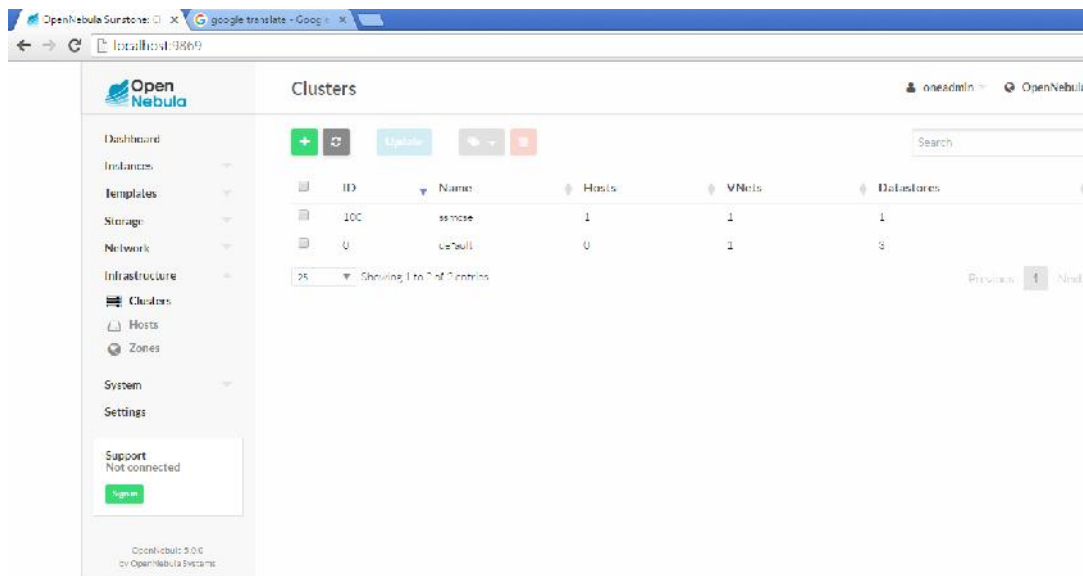


Step 3: Open Browser, type localhost:9869 and Login using username: oneadmin, password: opennebula



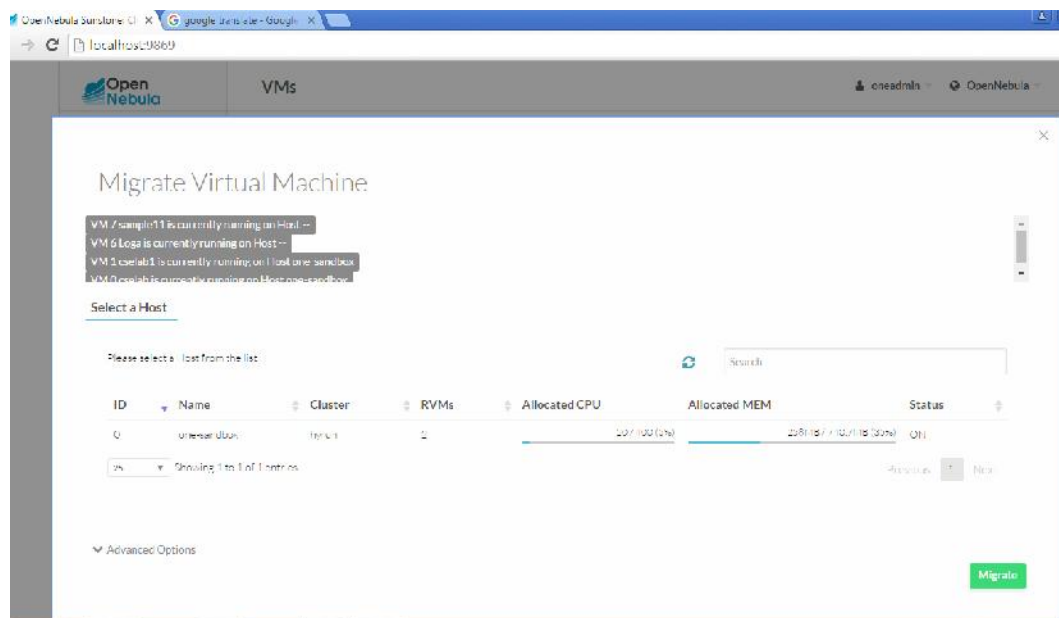
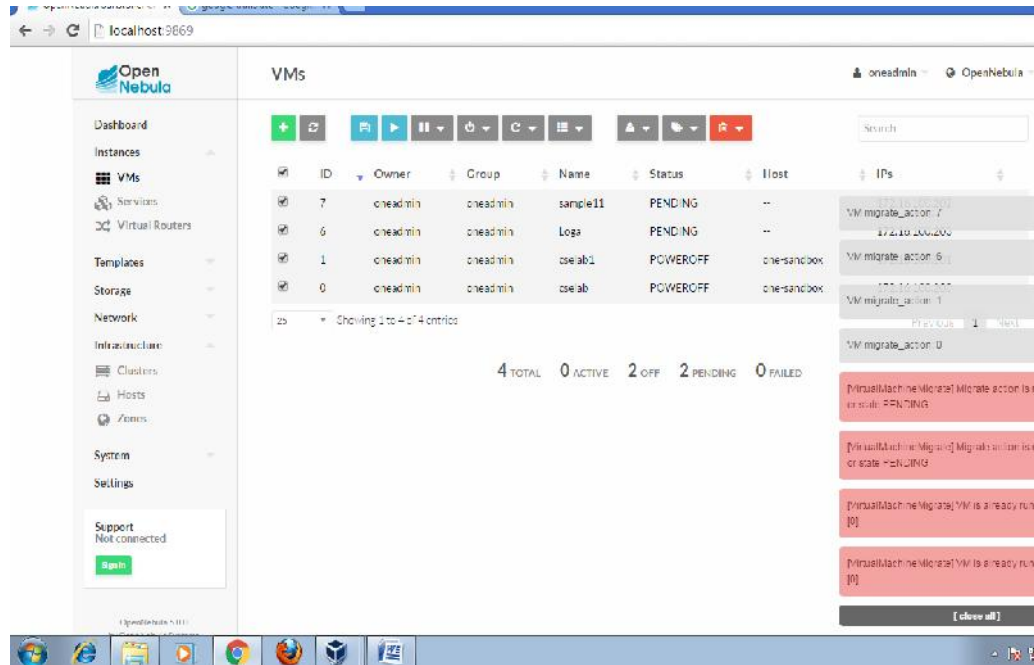
Step 4: Then follow the steps to migrate VMs

- a. Click on infrastructure
- b. Select clusters and enter the cluster name
- c. Then select host tab, and select all host
- d. Then select Vnets tab, and select all vnet
- e. Then select datastores tab, and select all datastores
- f. And then choose host under infrastructure tab
- g. Click on + symbol to add new host, name the host then click on create



Step 5: on instances, select VMs to migrate then follow the steps

- Click on 8th icon ,the drop down list display
- Select migrate on that ,the popup window display
- On that select the target host to migrate then click on migrate.



RESULT:

Thus the virtual machine migration based on the certain condition from one node to the other was written , executed and the output was verified successfully.

Ex.No.7

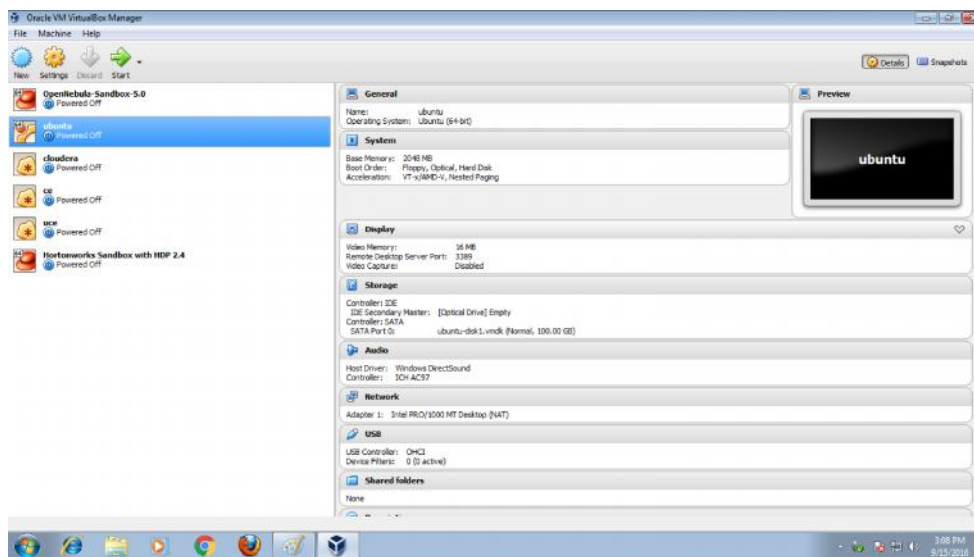
TO INSTALL STORAGE CONTROLLER AND INTERACT WITH IT

AIM:

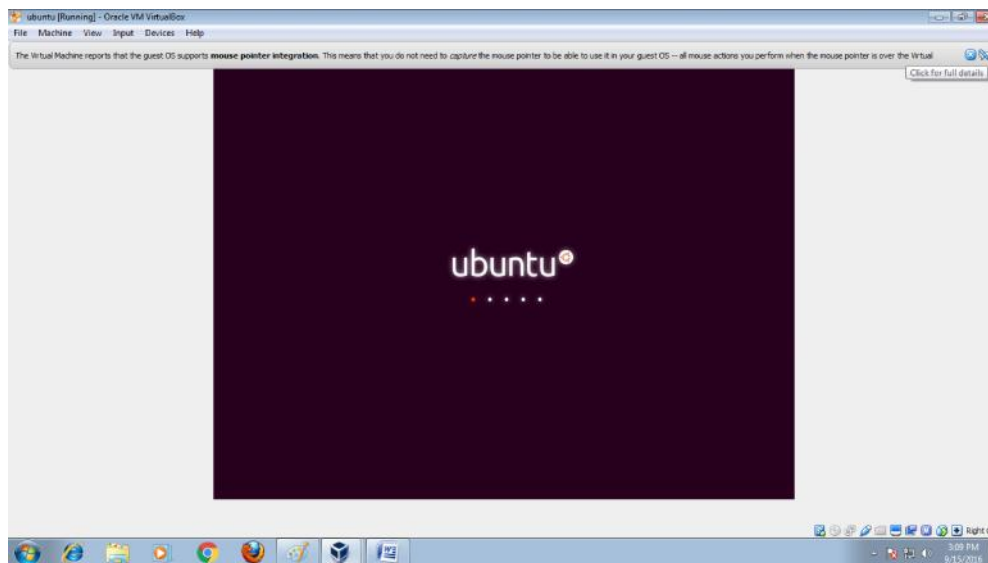
To find the procedure to install storage controller and interact with it.

PROCEDURE:

Step 1: Open the Oracle VM VirtualBox Manager.

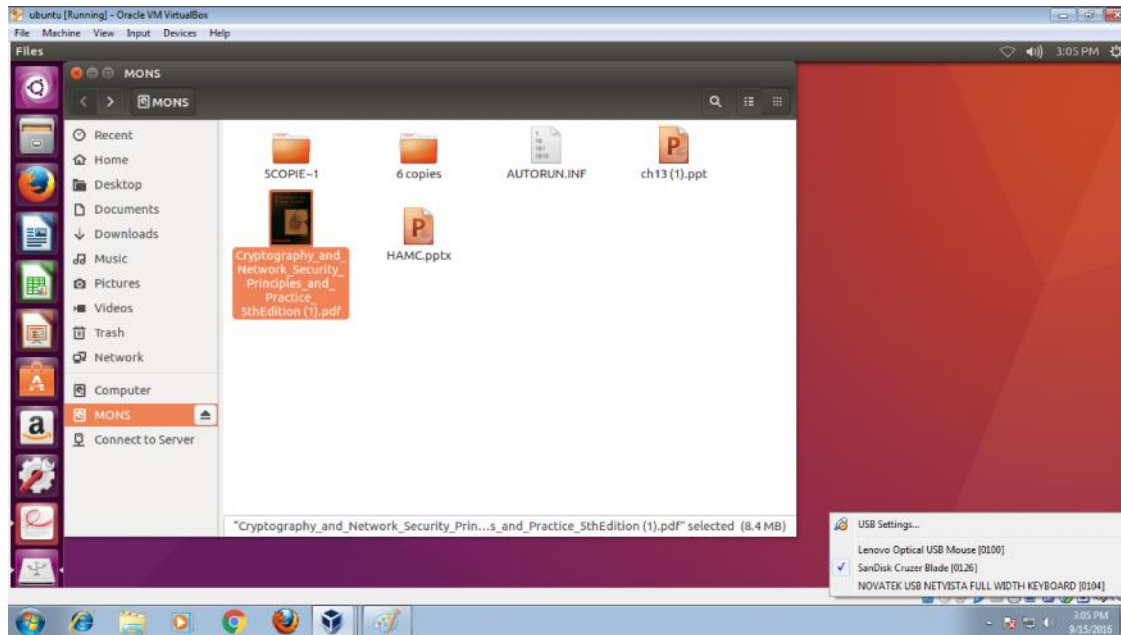


Step 2: Start the Ubuntu VM.



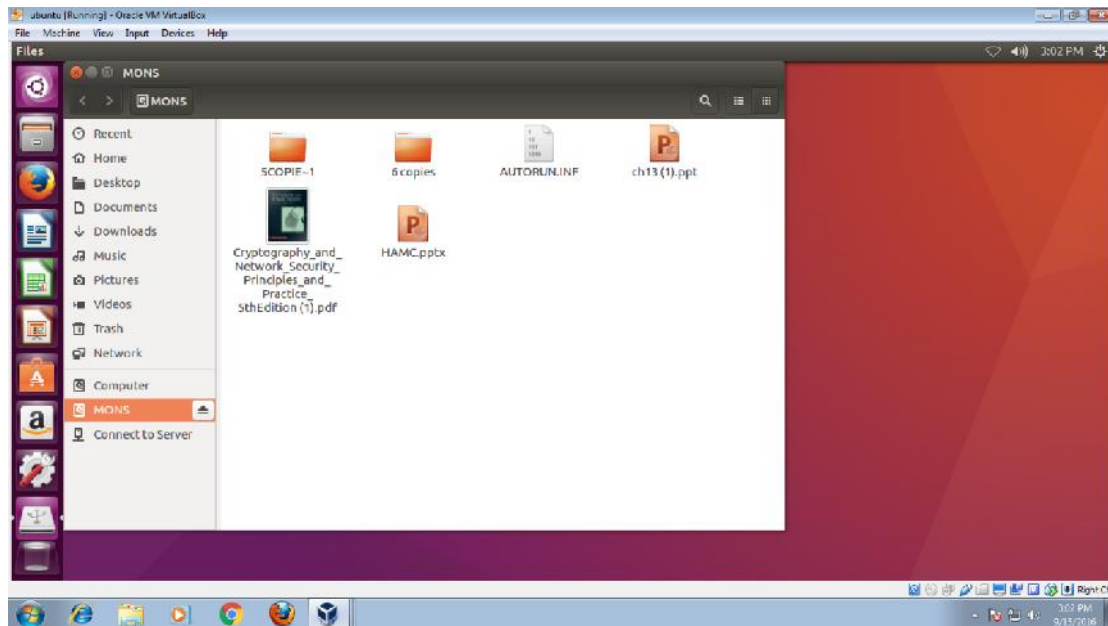
Step 3: After Running the Ubuntu VM ,plug the USB.

Step 4: Right Click on the USB icon at bottom right corner.

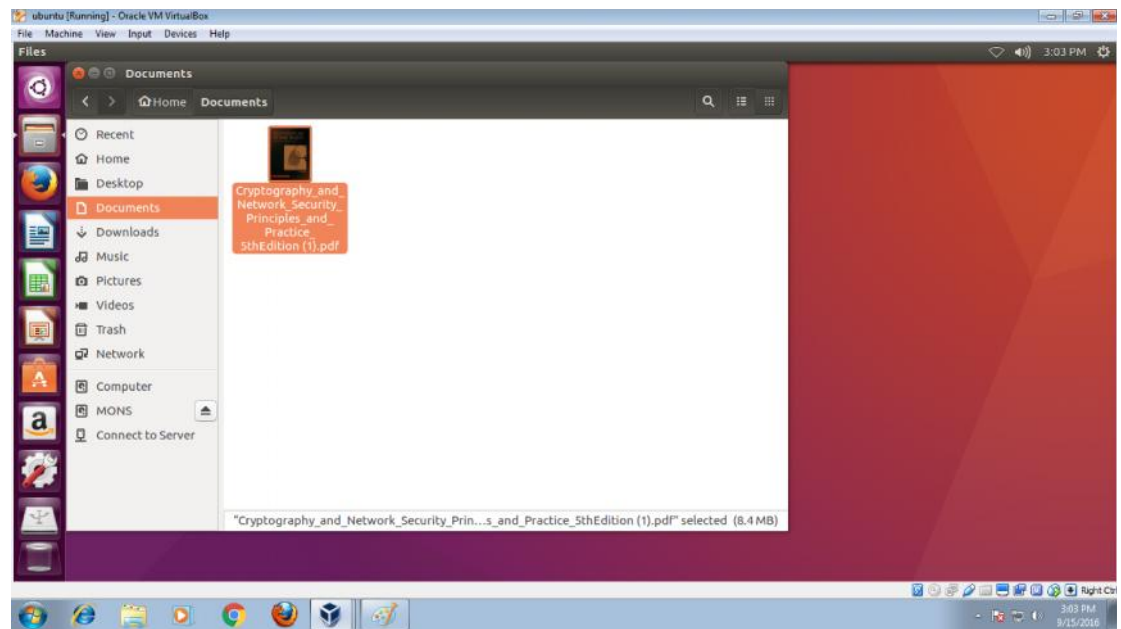


Step 5: Select your device name like jetflash,sandisk etc.

Step 6: Explorer window open.



Step 7: Then do read, write operations on the USB.



RESULT:

Thus the procedure to install storage controller and interact with it was done, executed and the output was verified successfully.

HADOOP

Ex.No.8	SETTING ONE NODE HADOOP CLUSTER

AIM:

To write a procedure for setting one node hadoop cluster.

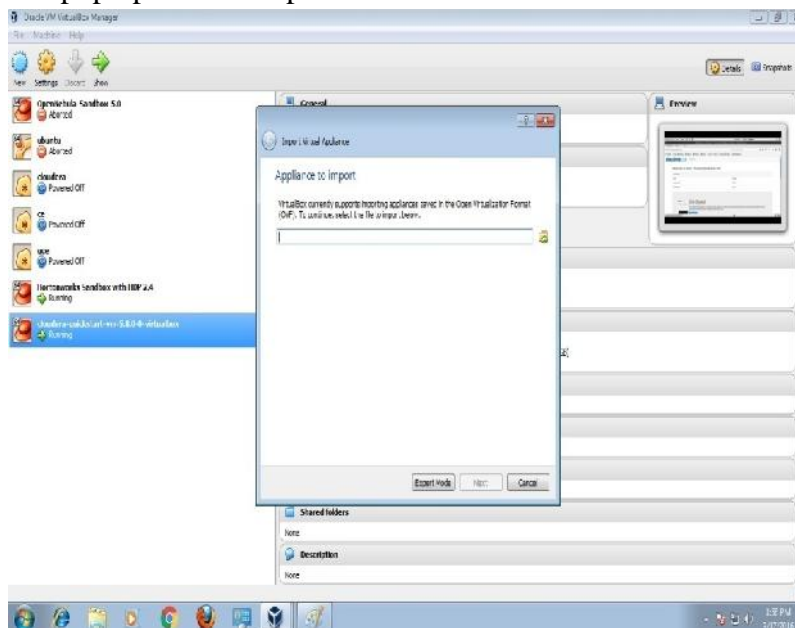
PROCEDURE:

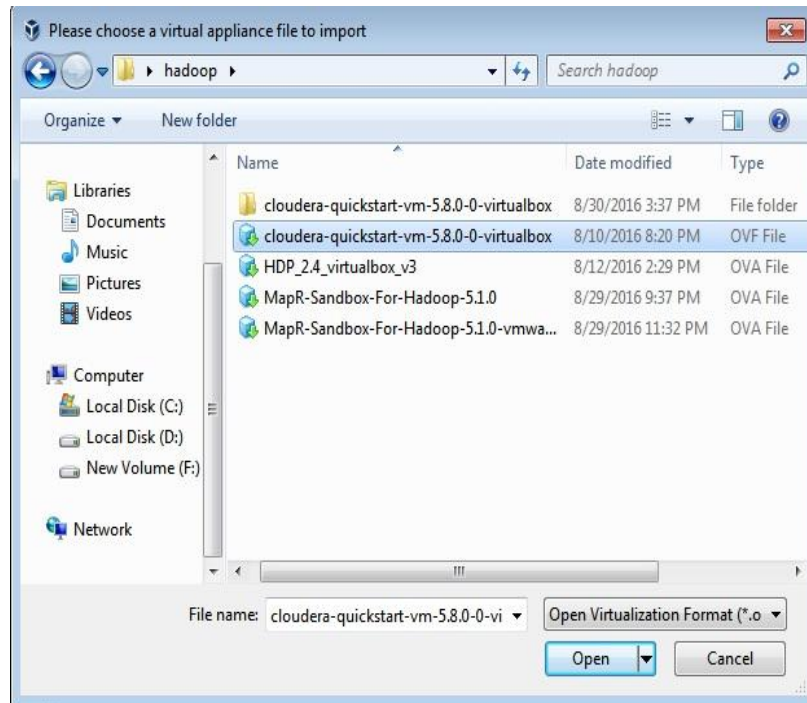
Step 1:Open the virtual box.

Step 2:click File->import appliances.

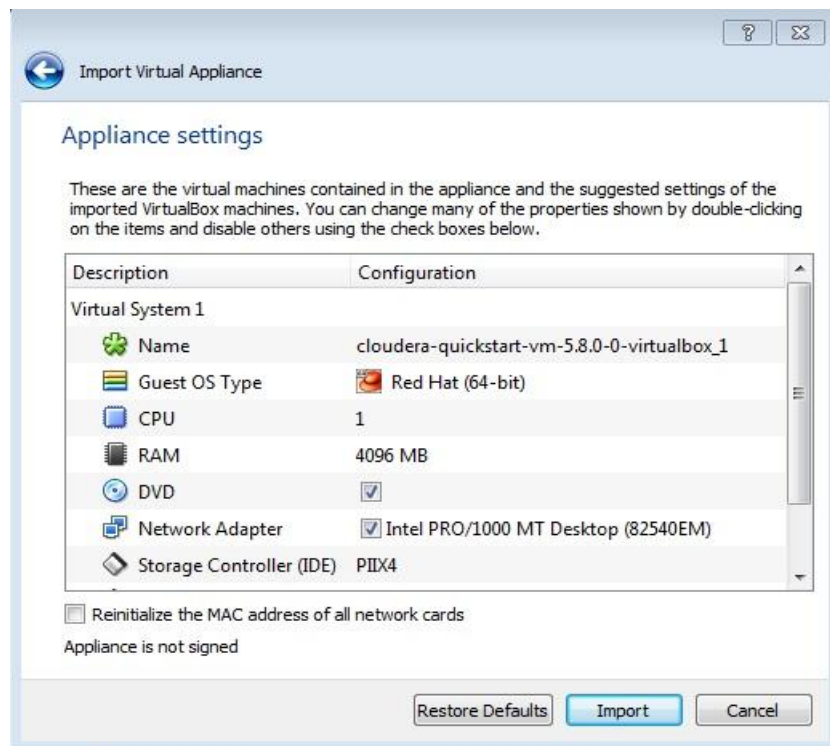


Step 3:the pop up box will open.Browse the cloudera.ovf file. and click OK.





Step 4: Import virtual appliance pop up box opened. Click import.



Step 5: Select cloudera and click on Start.

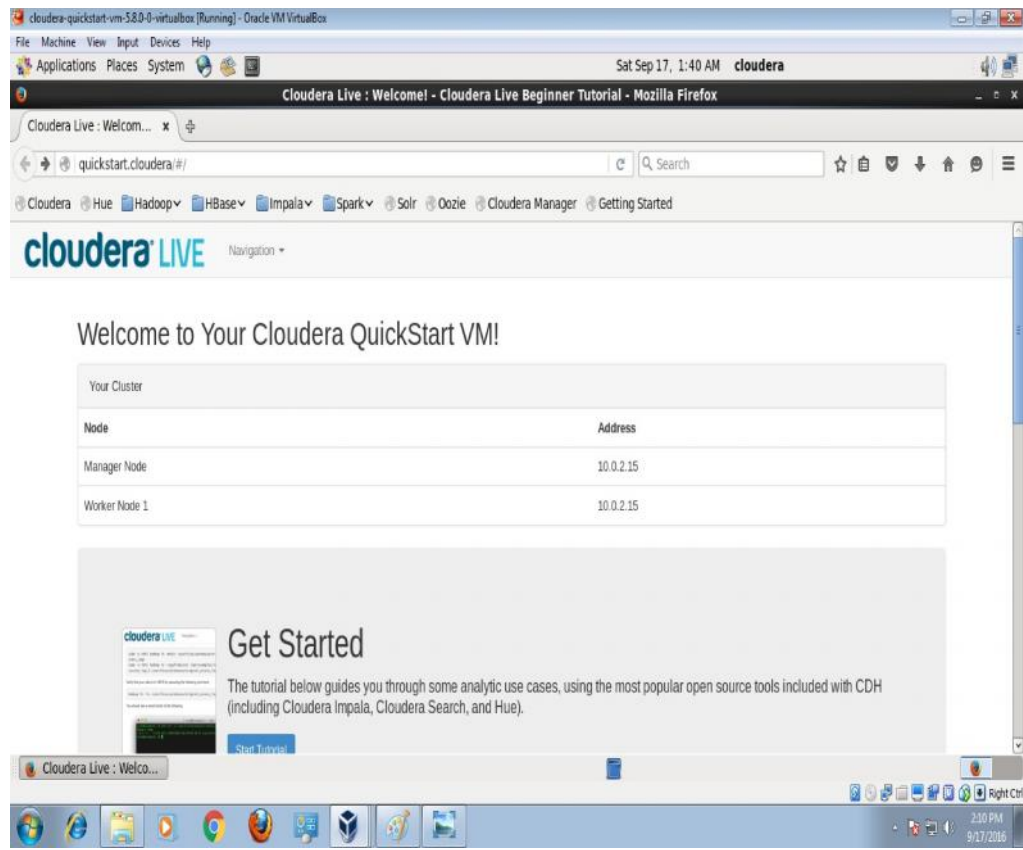


Step 6: After starting the cloudera, it will load.



Step 7: One node cluster will be create.

OUTPUT:



RESULT:

Thus the procedure for setting one node hadoop cluster has been successfully completed.

Ex.No.9	WORDCOUNT PROGRAM TO DEMONSTRATE MAP AND REDUCE TASK

AIM:

. To write a word count program to demonstrate the use of map and reduce tasks .

PROCEDURE:

step 1:Open cloudera.

step 2:Give the following command to create input output directory in hadoop.

```
$ sudo su hdfs
```

```
$ hadoop fs -mkdir /user/cloudera
```

```
$ hadoop fs -chown cloudera /user/cloudera
```

```
$ exit
```

```
$ sudo su cloudera
```

```
$ hadoop fs -mkdir /user/cloudera/wordcount /user/cloudera/wordcount/input
```

Step 3:Create the three sample text file as input using the commands

```
$ echo "Hadoop is an elephant" > file0
```

```
$ echo "Hadoop is as yellow as can be" > file1
```

```
$ echo "Oh what a yellow fellow is Hadoop" > file2
```

Step 4:Move them to the directory using the following command.

```
$ hadoop fs -put file* /user/cloudera/wordcount/input
```

Step 5:Create the following java file as WordCount.java

```
import java.io.IOException;
```

```
import java.util.StringTokenizer;
```

```
import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.fs.Path;
```

```

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        public void map(Object key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }

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

```

```

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

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

```


Step 6: Create a folder `build` to store all the class files.

```
mkdir -p build
```

Step 7: Compile the above java file using the command

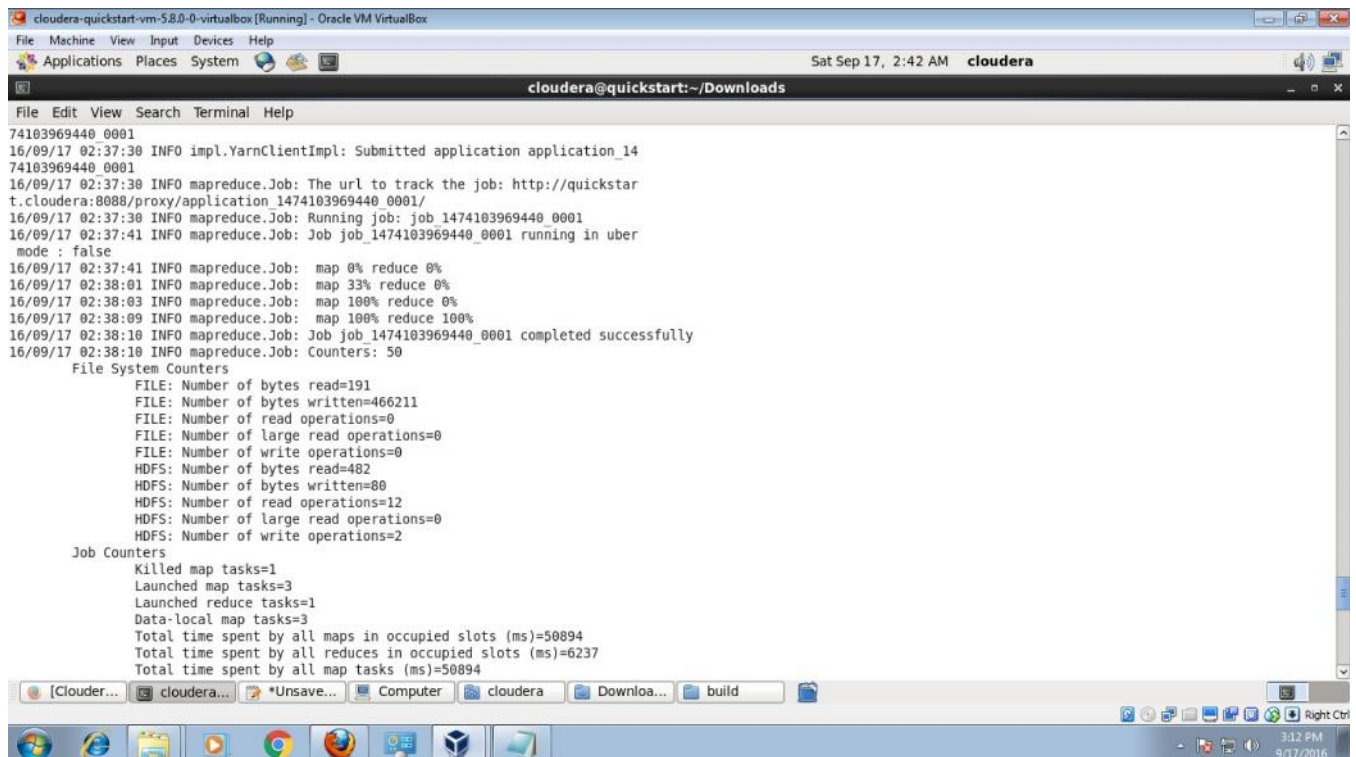
```
$ javac -cp /usr/lib/hadoop/*:/usr/lib/hadoop-mapreduce/* WordCount.java -d build -Xlint
```

Step 8: make all the class file into single jar file using the following command.

```
jar -cvf wordcount.jar -C build/ .
```

Step 9: give the following command to run the jar file in hadoop.

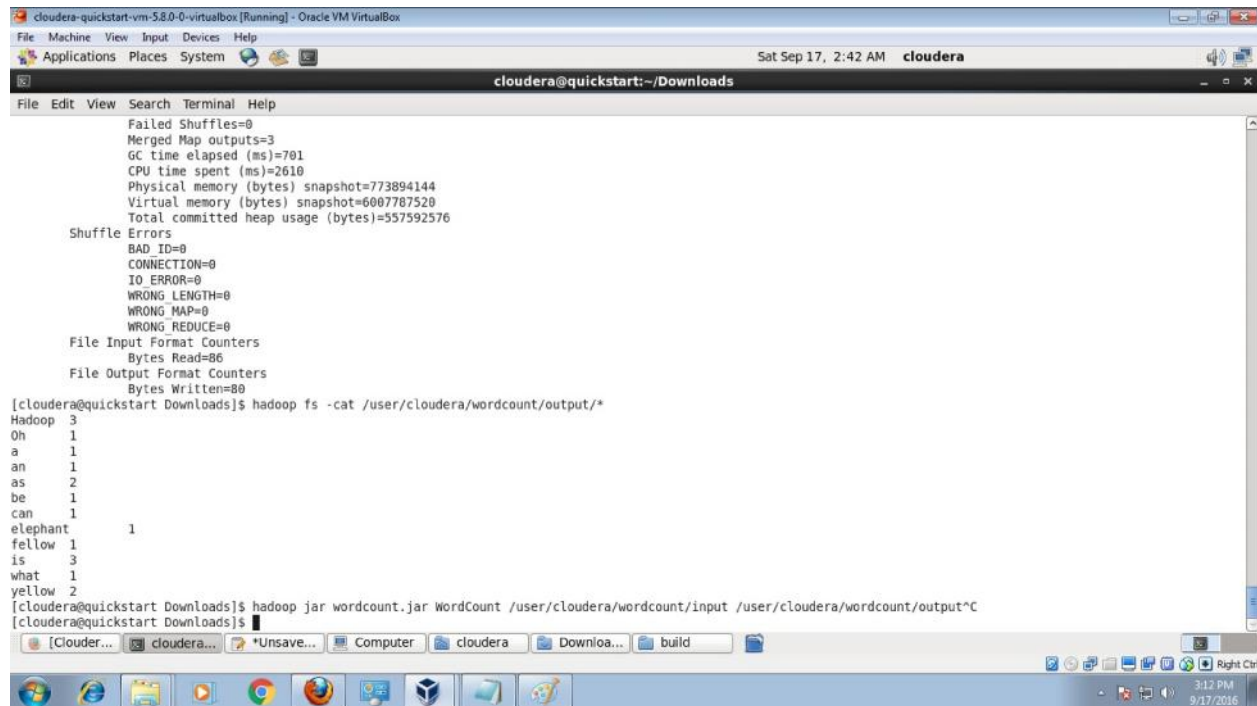
```
hadoop jar wordcount.jar WordCount /user/cloudera/wordcount/input  
/user/cloudera/wordcount/output
```

A screenshot of a terminal window titled 'cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox'. The terminal shows the output of a Hadoop job. The output includes log messages from the YarnClientImpl, mapreduce.Job, and the job's progress. It shows the job was submitted, running, and completed successfully. The output also includes File System Counters and Job Counters. The File System Counters show the number of bytes read and written, and the number of read and write operations. The Job Counters show the number of killed map tasks, launched map tasks, launched reduce tasks, and data-local map tasks. The total time spent by all maps in occupied slots is 50894 ms, and the total time spent by all reduces in occupied slots is 6237 ms. The total time spent by all map tasks is 50894 ms. The terminal window has a menu bar with File, Edit, View, Search, Terminal, and Help. The status bar at the bottom shows the date and time as Sat Sep 17, 2:42 AM, and the user as cloudera. The taskbar at the bottom shows several icons, including Cloudera, cloudera..., *Unsave..., Computer, cloudera, Download..., and build.

Step 10: to view the output in hadoop give the following command

```
hadoop fs -cat /user/cloudera/wordcount/output/*
```

OUTPUT:



The screenshot shows a terminal window titled "cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox". The terminal displays the output of a Hadoop wordcount job. The output includes statistics such as "Failed Shuffles=0", "Merged Map outputs=3", and "Total committed heap usage (bytes)=557592576". It also lists "Shuffle Errors" which are all zero. The "File Input Format Counters" show "Bytes Read=86" and the "File Output Format Counters" show "Bytes Written=80". The final output is a list of words and their counts: "Hadoop 3", "Oh 1", "a 1", "an 1", "as 2", "be 1", "can 1", "elephant 1", "fellow 1", "is 3", "what 1", and "yellow 2". The terminal prompt is "[cloudera@quickstart Downloads]\$".

```
Failed Shuffles=0
Merged Map outputs=3
GC time elapsed (ms)=701
CPU time spent (ms)=2610
Physical memory (bytes) snapshot=773894144
Virtual memory (bytes) snapshot=6007787520
Total committed heap usage (bytes)=557592576

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

File Input Format Counters
  Bytes Read=86
File Output Format Counters
  Bytes Written=80

[cloudera@quickstart Downloads]$ hadoop fs -cat /user/cloudera/wordcount/output/*
Hadoop 3
Oh 1
a 1
an 1
as 2
be 1
can 1
elephant 1
fellow 1
is 3
what 1
yellow 2

[cloudera@quickstart Downloads]$ hadoop jar wordcount.jar WordCount /user/cloudera/wordcount/input /user/cloudera/wordcount/output*C
[cloudera@quickstart Downloads]$
```

RESULT:

Thus the word count program to demonstrate the use of map and reduce tasks and the output was verified successfully.