

MOOKAMBIGAI COLLEGE OF ENGINEERING
Srinivasa Nagar, Kalamavur-622 502.

CS8711 – CLOUD COMPUTING LAB



2022-2023

VII Semester

**DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING**

MOOKAMBIGAI COLLEGE OF ENGINEERING
(Approved by AICTE & Affiliated to Anna University of Technology, Chennai)
(NBA Accredited, ISO 9001:2008 Certified Institution)
Srinivasa Nagar, Kalamavur-622502, Pudukkottai District, Tamil Nadu.

Name.....

Year..... Semester..... Branch.....

University Register No:

--	--	--	--	--	--	--	--	--	--	--	--

CERTIFICATE

**Certified that this is the Bonafide record of work done by the above student in the
CS8711-CLOUD COMPUTING Laboratory during the year 2022-2023.**

Signature of Staff In-charge

Submitted for the University practical Examination held on.....

EXAMINERS

Date:

Internal:

External:

INDEX

EXPERIMENT		NAME OF THE EXPERIMENT	PAGE NO.	SIGNATURE
NO	DATE			
1		Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.		
2		Install a C compiler in the virtual machine created using virtual box and execute Simple Programs		
3		Install Google App Engine. Create <i>hello world</i> app and other simple web applications using python/java.		
4		Use GAE launcher to launch the web applications.		
5		Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.		
6		Find a procedure to transfer the files from one virtual machine to another virtual machine.		
7		Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)		
8		Install Hadoop single node cluster and run simple applications like wordcount.		

Ex. No:1
Date:

Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.

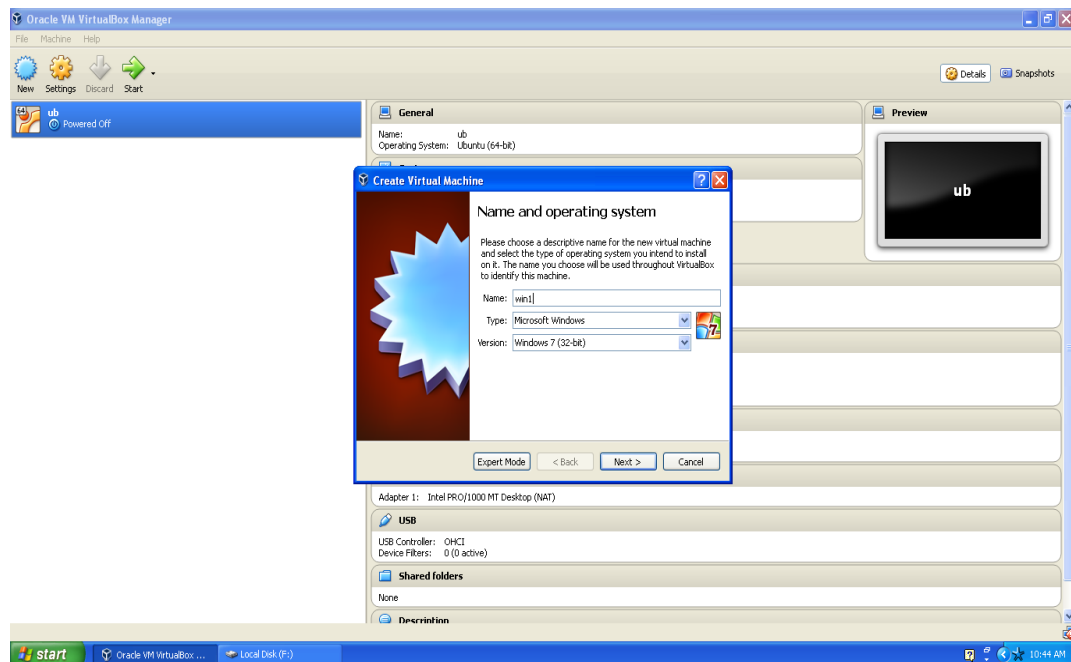
AIM:

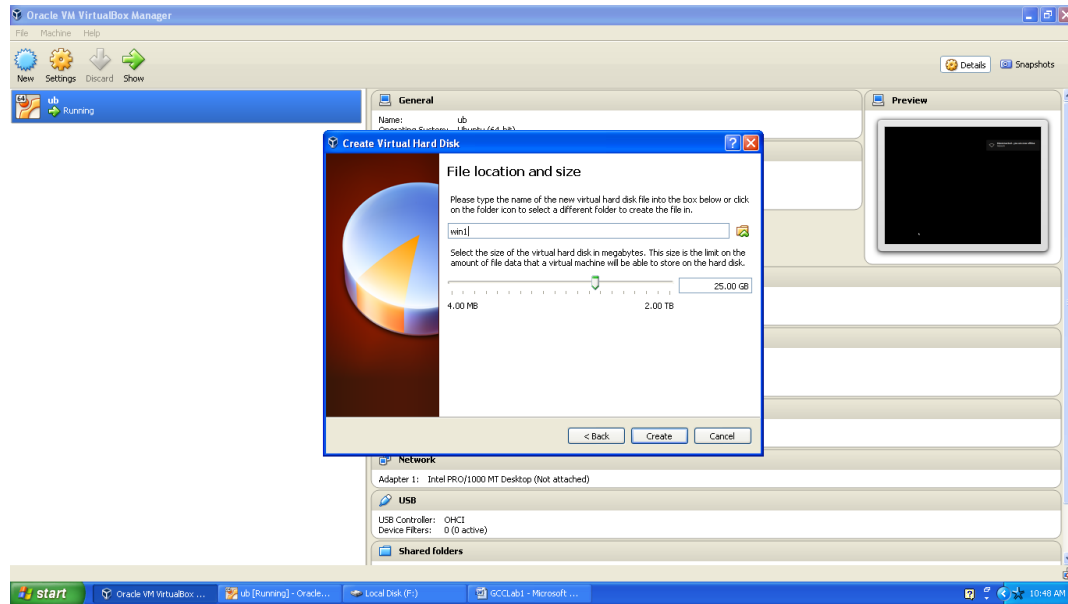
Find procedure to Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.

This experiment is to be performed through portal.

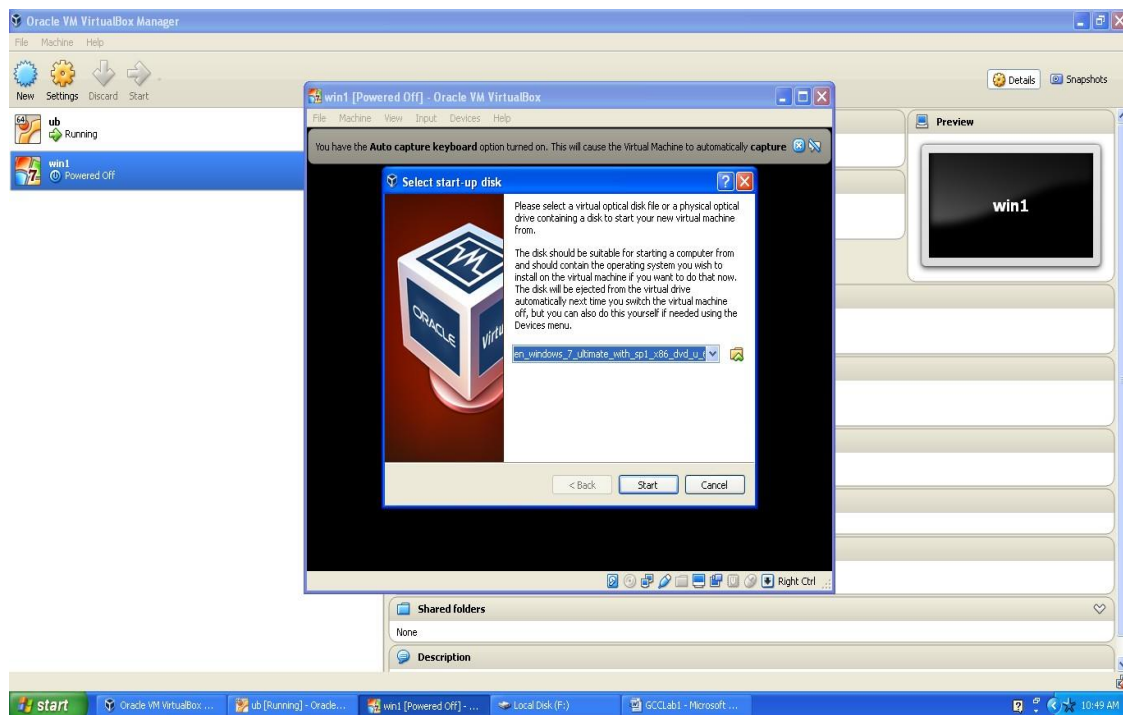
PROCEDURE TO INSTALL

Step1: Open Oracle virtual box manager and click create new -> virtual machine. Provide and name for the operating system and select the memory size to be occupied in memory.

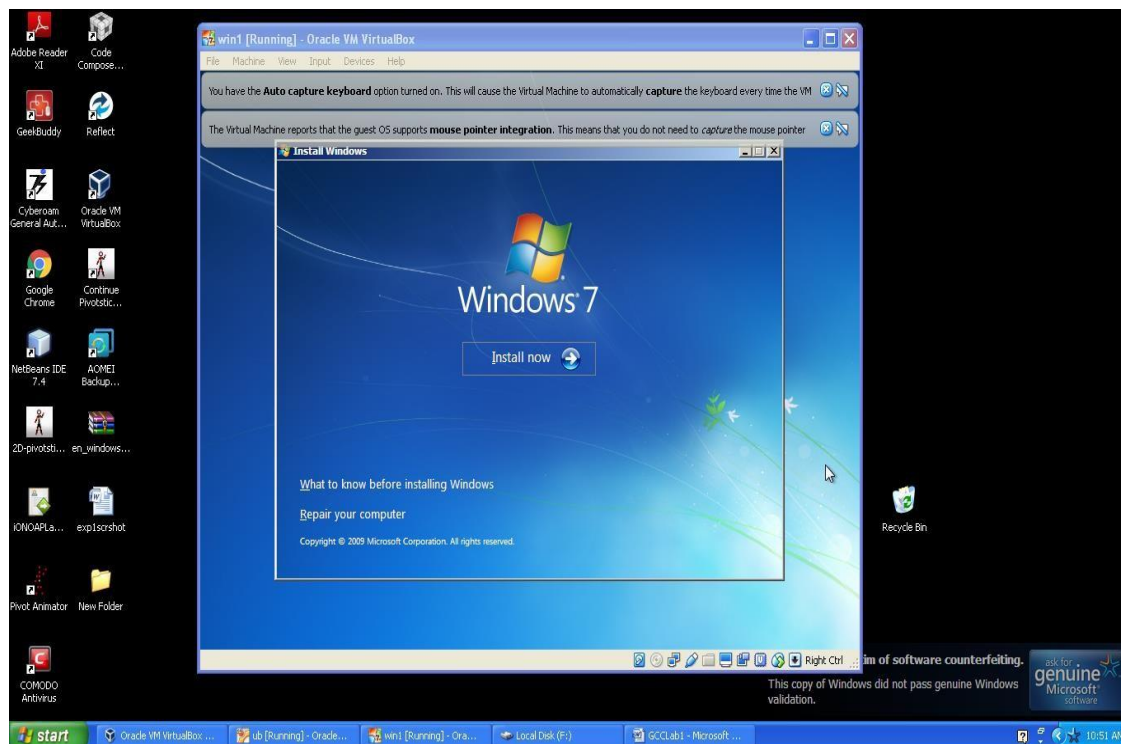
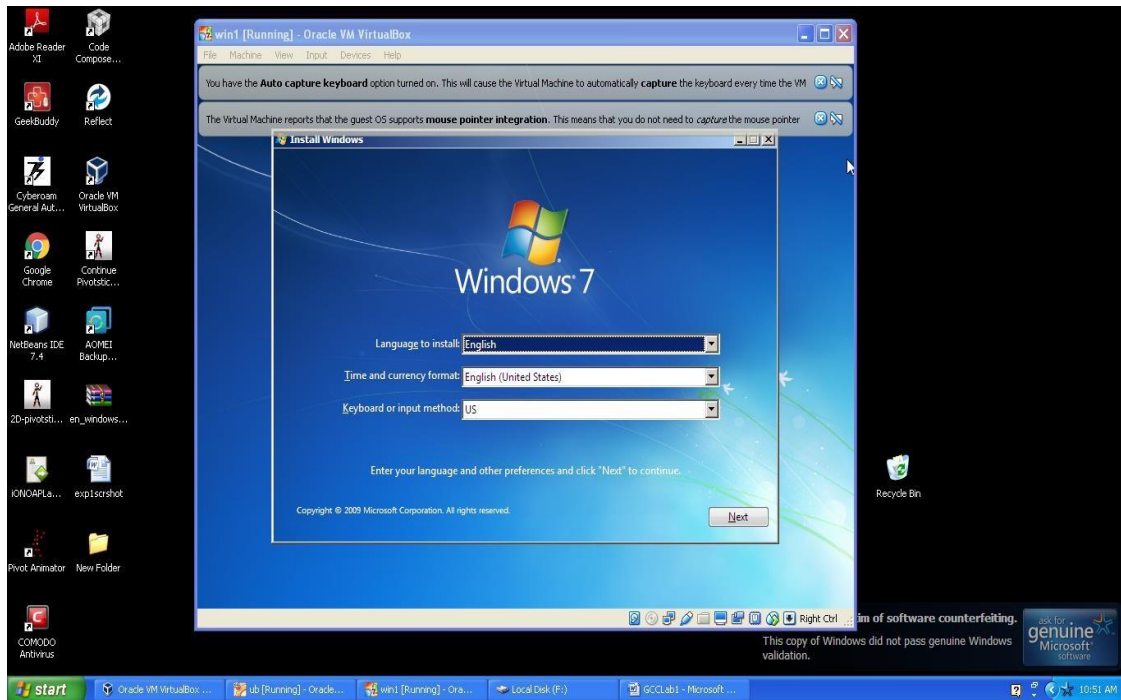




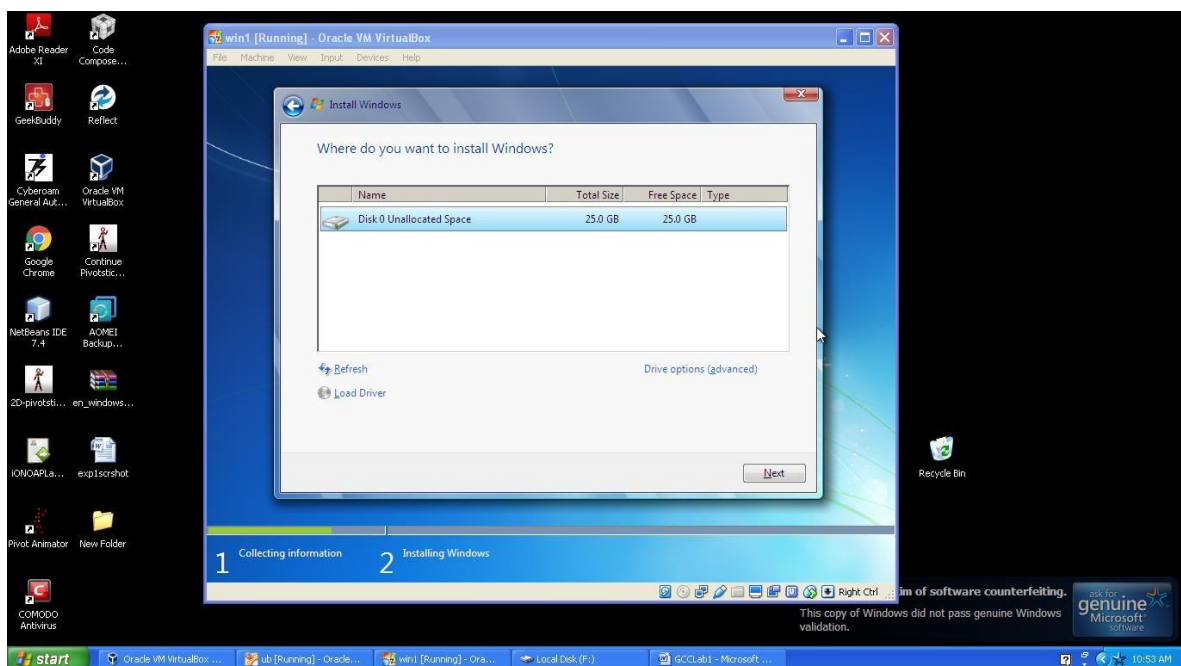
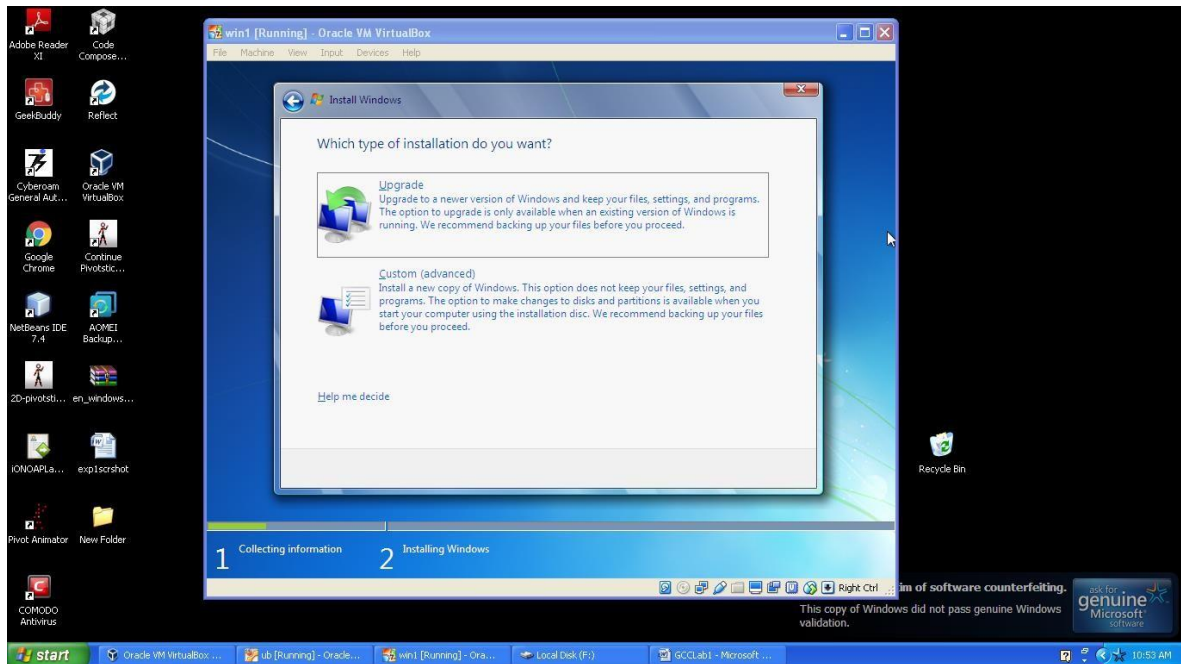
Step 2: Select the iso file of the virtual OS Windows7 and click Start.



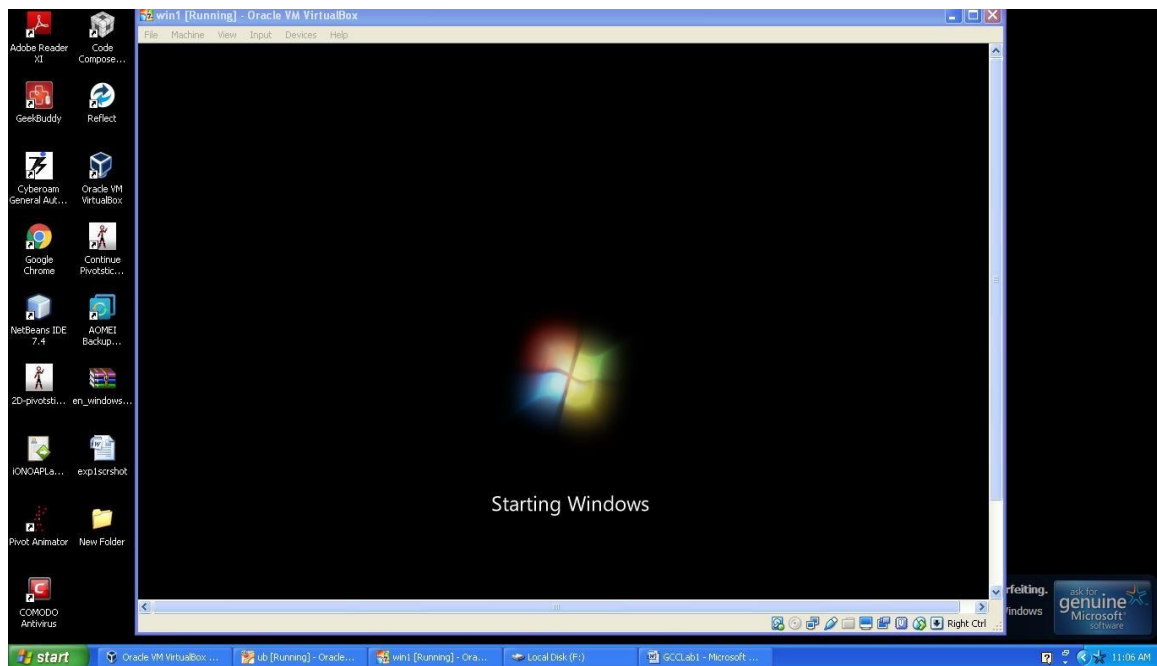
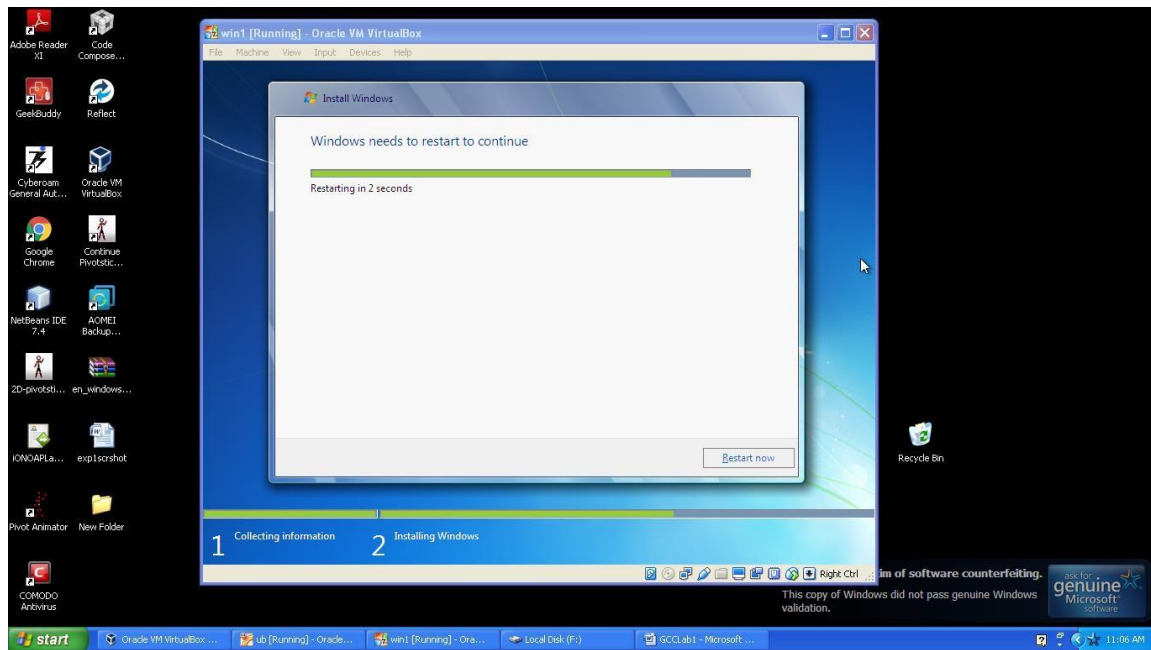
Step 3: Select the language to use in the Operating System and click Install Now.

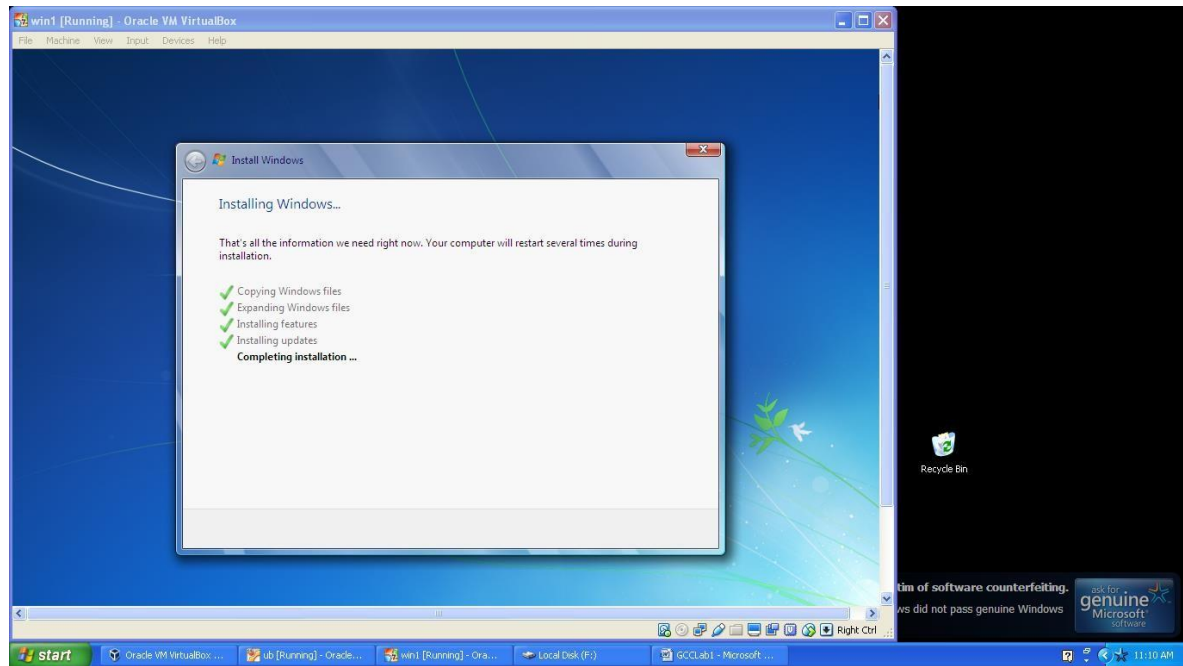


Step 4: Select the type of installation as Custom for new installation and allocate Disk space according to your convenience. Click Next to start the installation.

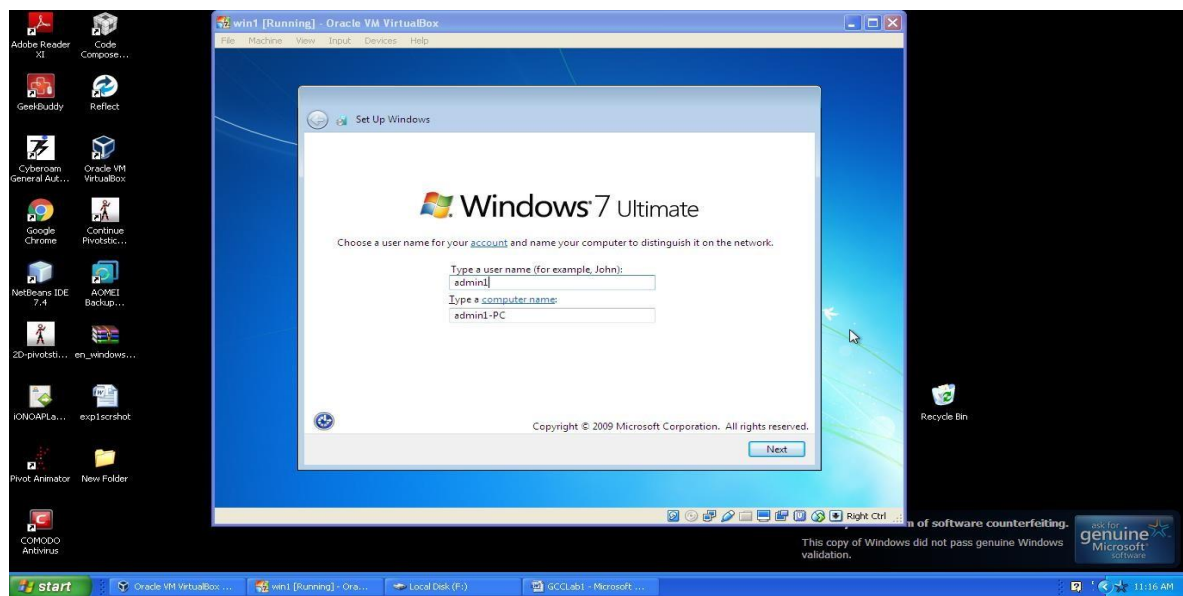


Step 5: After installation the system will be restarted to complete the installation.

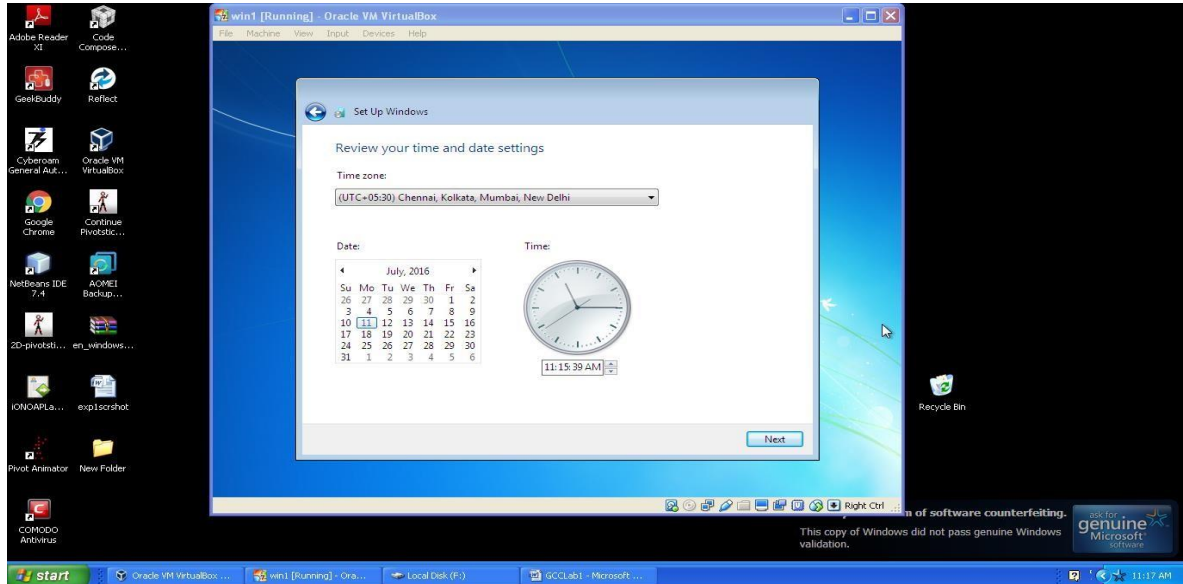




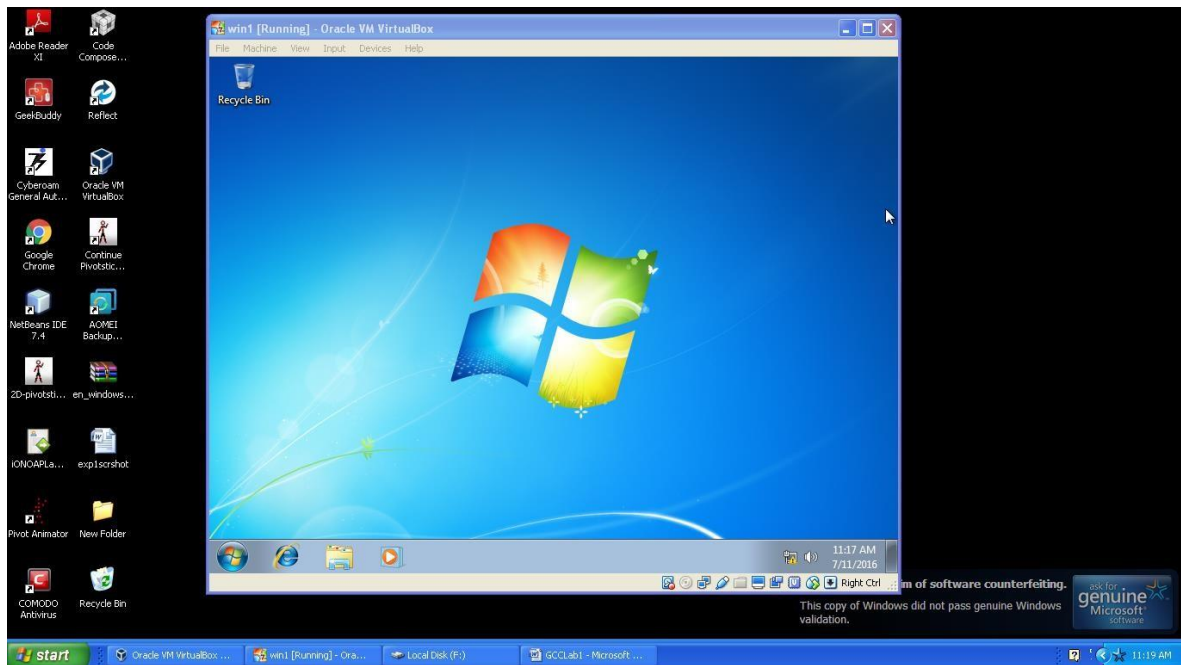
Step 6: Provide a user name and password(optional) to gain access over the OS.



Step 7: Set the time and date for the new Operating System.



Step 8: Thus the new Operating System Windows7 will be opened as the virtual machine.



RESULT:

Thus the above procedure to install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8 is checked.

Ex. No:2
Date:

Install a C compiler in the virtual machine created using virtual box and execute Simple Programs

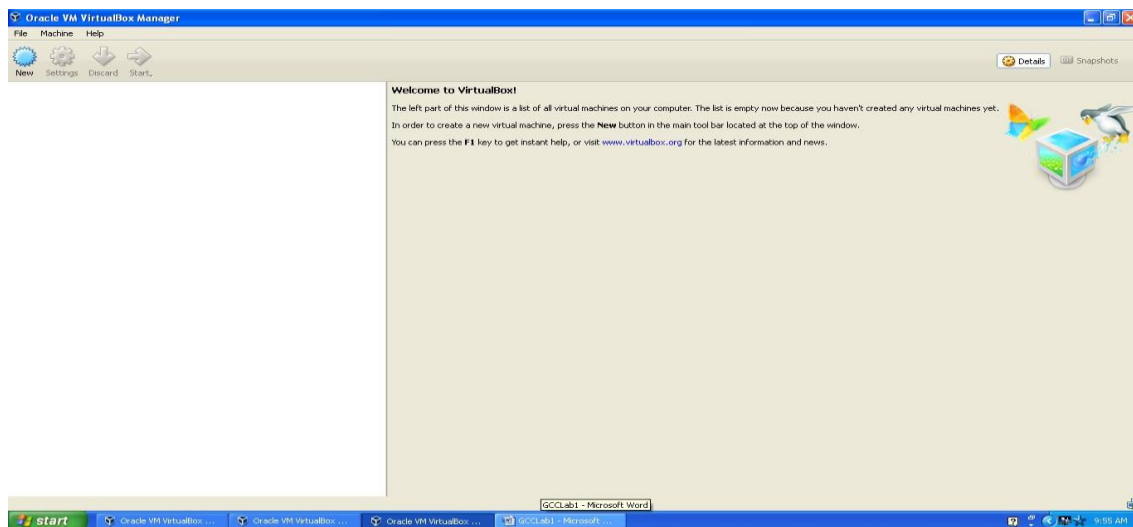
AIM:

To find a procedure to install a C compiler in the virtual machine and execute a sample program.

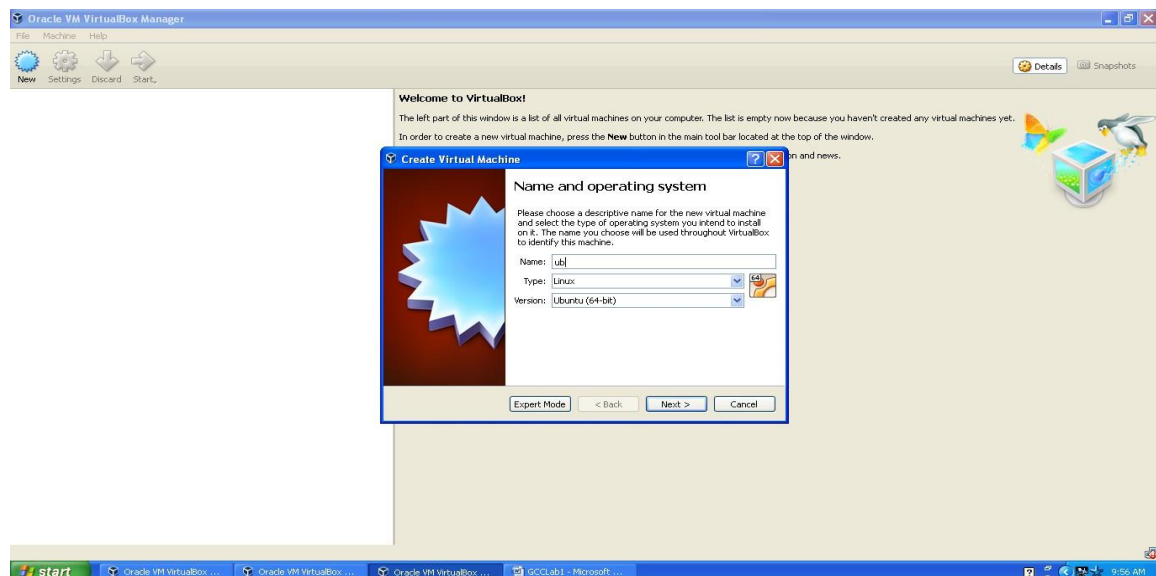
PROCEDURE:

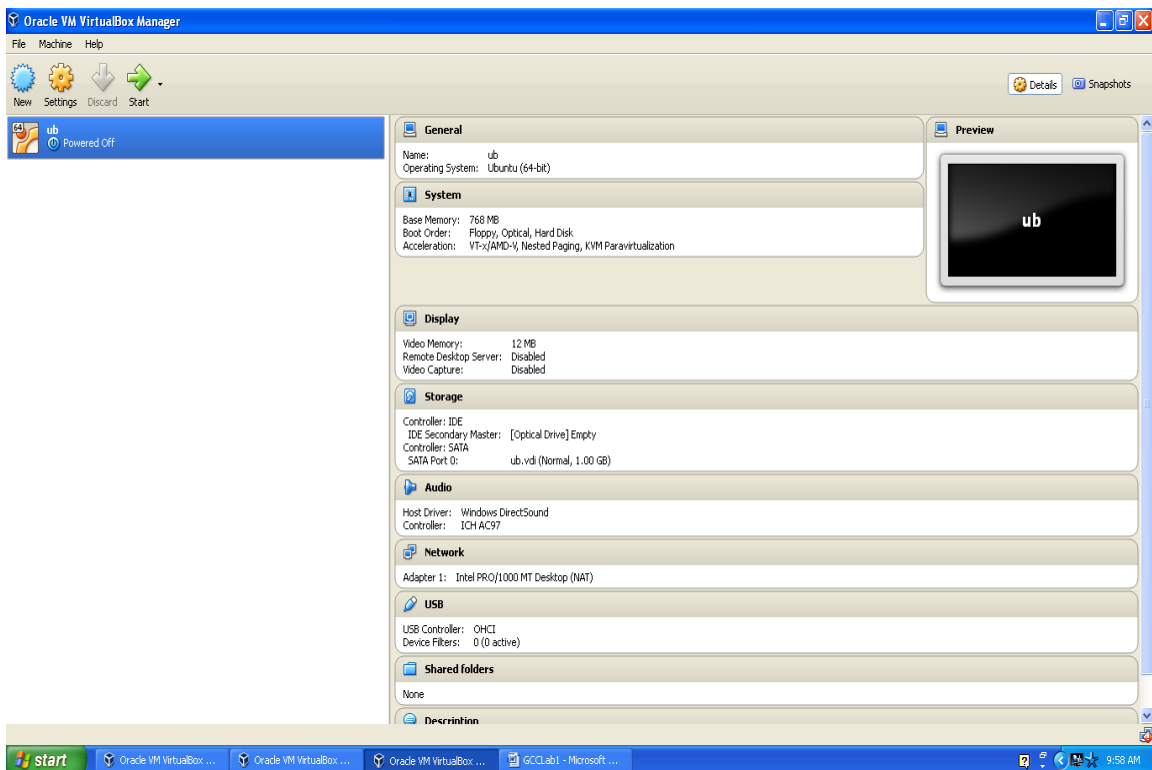
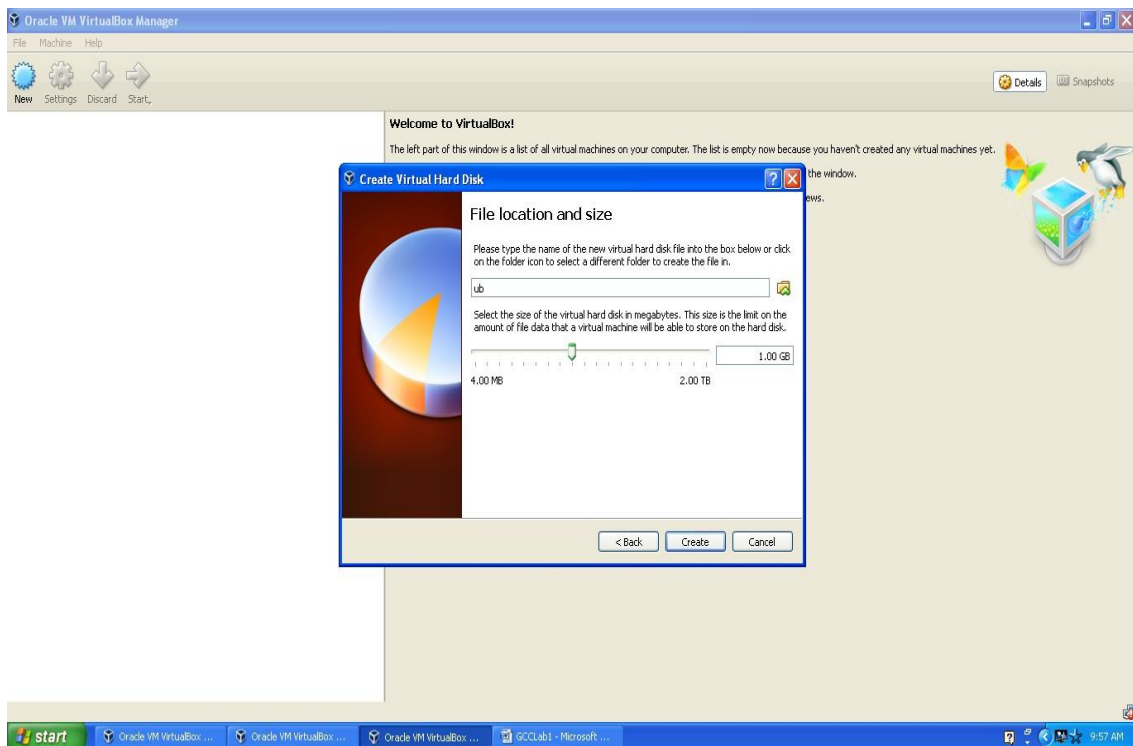
Installing Ubuntu using Oracle Virtual Box

Step1: Open Oracle virtual box manager and click create new -> virtual machine.

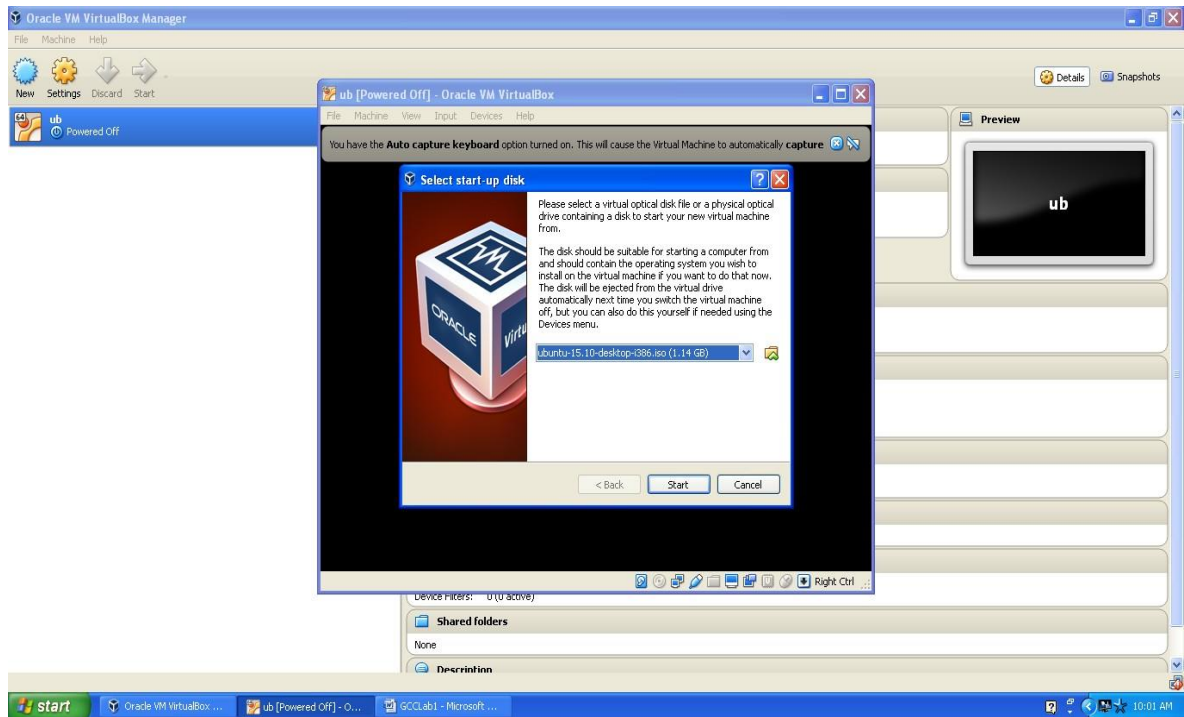


Step 2: Provide a name for the virtual machine and select the hard disk size for the virtual machine. Select the storage size as Dynamically allocated memory size and click OK. The virtual machine will be created.

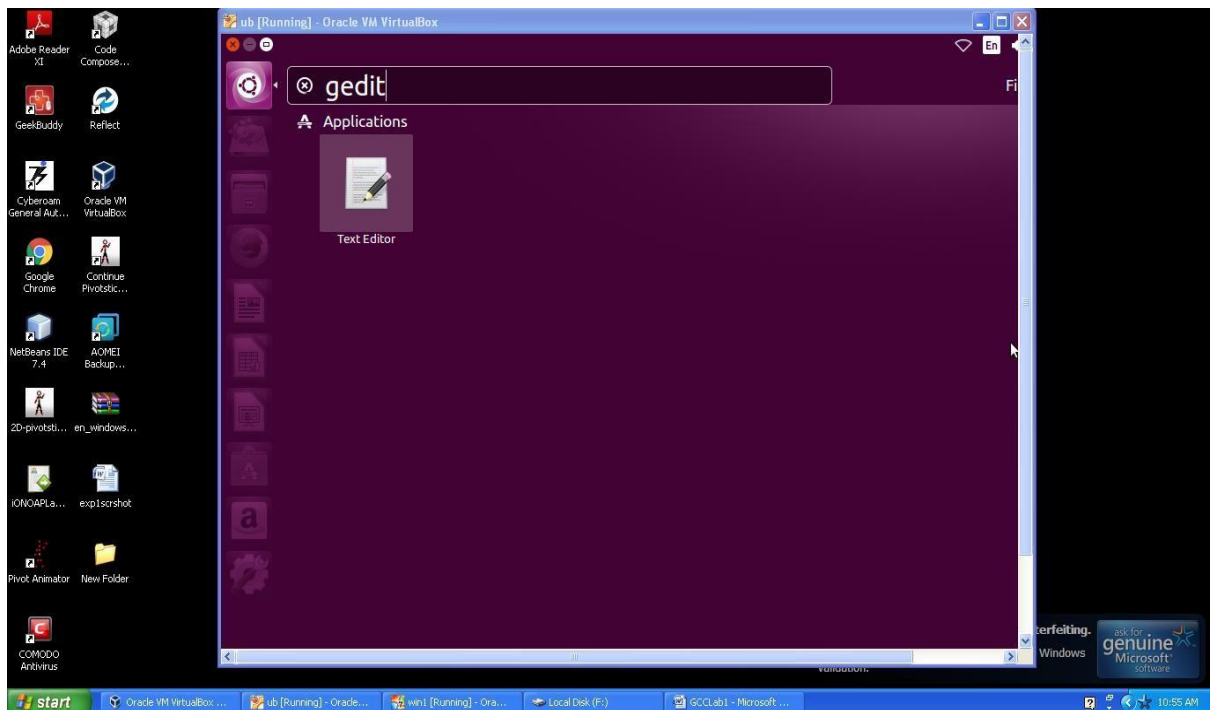




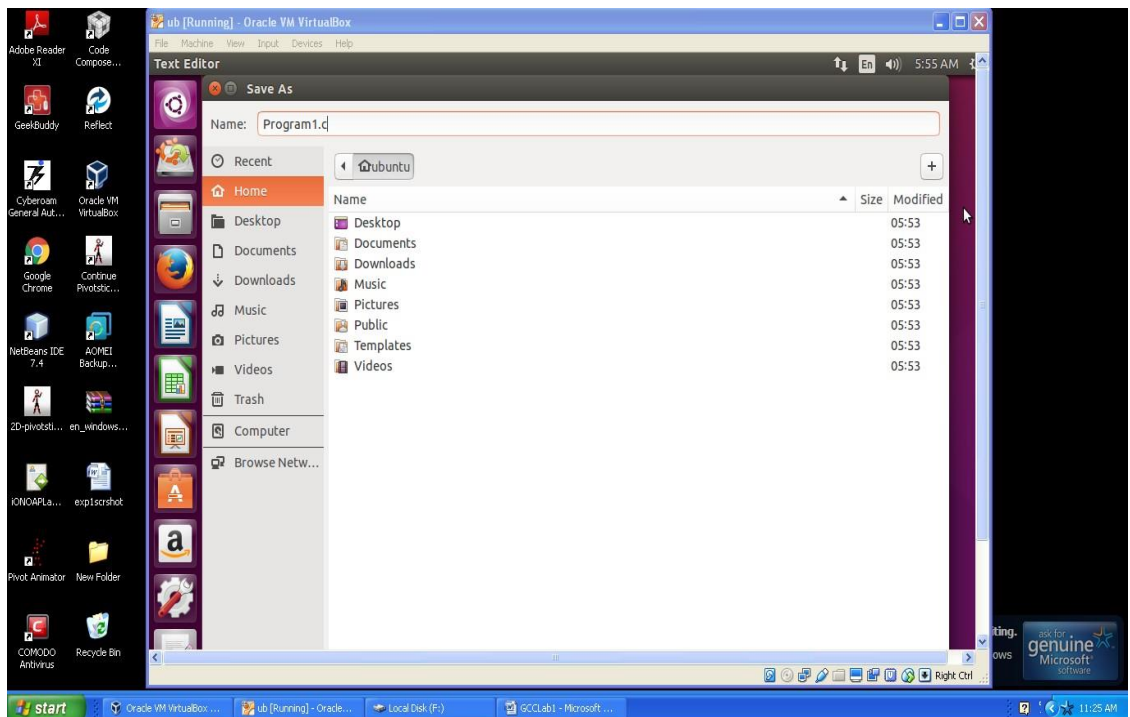
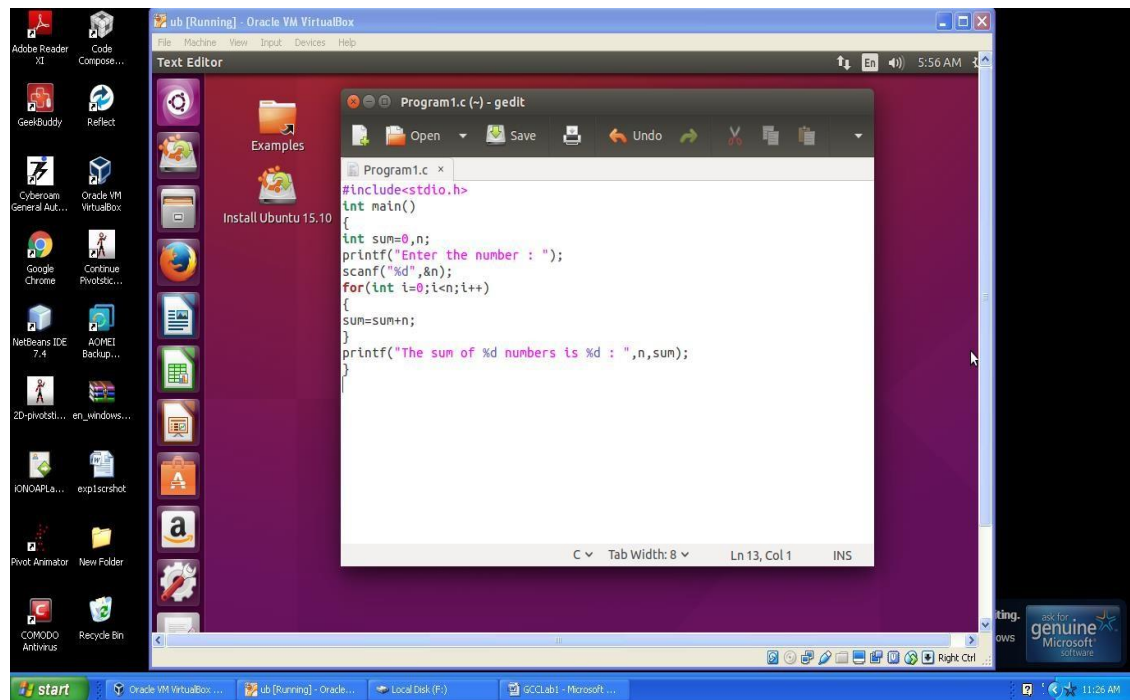
Step 3: Select the iso file of the virtual OS Ubuntu and click Start.



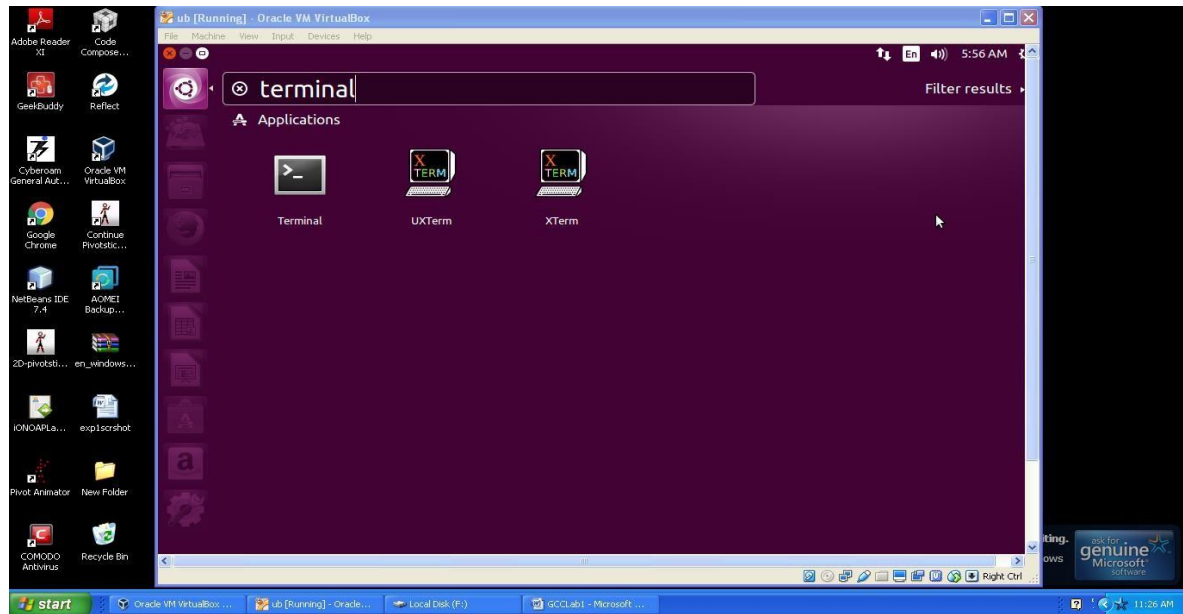
Step 4: The virtual OS Ubuntu is opened successfully. Now type “gedit” in the search box to open text editor in Ubuntu.



Step 5: Type your desired C program in text editor and save it with the extension (.c)

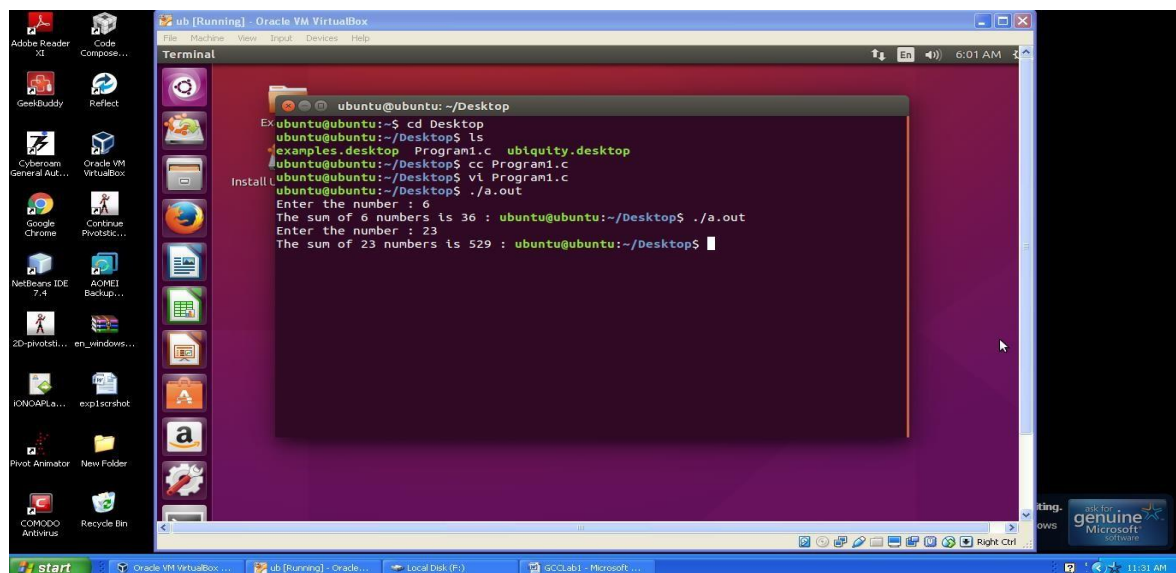


Step 6: Type “terminal” in the search box to open the command window.



Step 7: Type the necessary commands to compile and run the C program.

- cc_filename to compile the C program.
- ./a.out to display the output of the last compiled program.



RESULT:

Thus a C compiler is successfully installed in the virtual machine created using virtual box and a simple c program is compiled and output is verified successfully.

Ex. No:3 Date:	Install Google App Engine. Create <i>hello world</i> app and other simple web applications using python/java.
---------------------------------	--

AIM:

To install Google App Engine and Create *hello world* app and other simple web applications using python/java.

PROCEDURE:

This document describes the installation of the Google App Engine Software Development Kit (SDK) on a Microsoft Windows and running a simple “hello world” application.

The App Engine SDK allows you to run Google App Engine Applications on your local computer. It simulates the run---time environment of the Google App Engine infrastructure.

Pre-Requisites: Python 2.5.4

If you don't already have Python 2.5.4 installed in your computer, download and Install Python 2.5.4 from:

<http://www.python.org/download/releases/2.5.4/>

Download and Install

You can download the Google App Engine SDK by going to:

<http://code.google.com/appengine/downloads.html>

and download the appropriate install package.

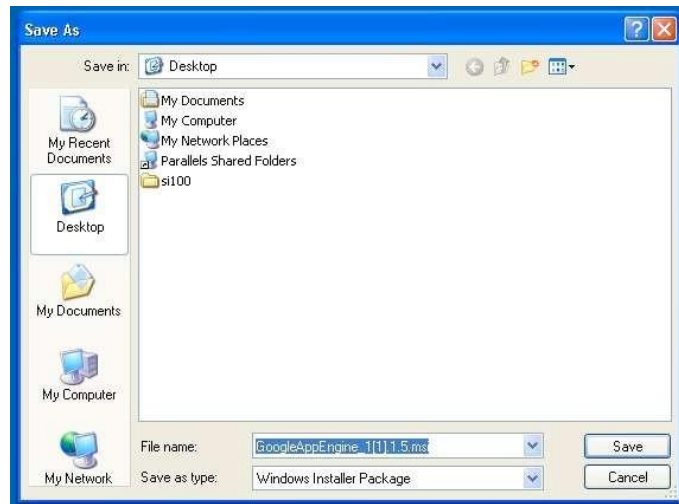
Download the Google App Engine SDK

Before downloading, please read the [Terms](#) that govern your use of the App Engine SDK.

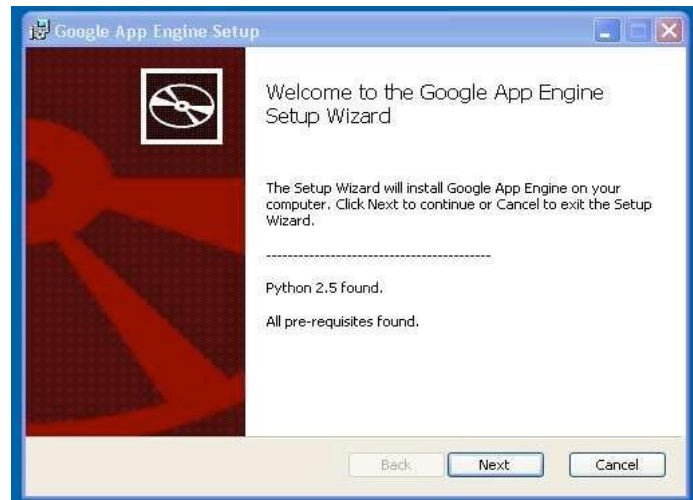
Please note: The App Engine SDK is under **active development**, please keep this in mind as you explore its capabilities. See the [SDK Release Notes](#) for the information on the most recent changes to the App Engine SDK. If you discover any issues, please feel free to notify us via our [Issue Tracker](#).

Platform	Version	Package	Size	SHA1 Checksum
Windows	1.1.5 - 10/03/08	GoogleAppEngine_1.1.5.msi	2.5 MB	e974312b4aefc0b3873ff0d93eb4c525d5e88c30
Mac OS X	1.1.5 - 10/03/08	GoogleAppEngineLauncher-1.1.5.dmg	3.6 MB	f62208ac01c1b3e39796e58100d5f1b2f052d3e7
Linux/Other Platforms	1.1.5 - 10/03/08	google_appengine_1.1.5.zip	2.6 MB	cbb9ce817bdabf1c4f181d9544864e55ee253de1

Download the Windows installer—the simplest thing is to download it to your Desktop or another folder that you remember.



Double Click on the **GoogleApplicationEngine** installer.



Click through the installation wizard, and it should install the App Engine. If you do not have Python 2.5, it will install Python 2.5 as well.

Once the install is complete you can discard the downloaded installer



Making your First Application

Now you need to create a simple application. We could use the “+” option to have the launcher make us an application – but instead we will do it by hand to get a better sense of what is going on.

Make a folder for your Google App Engine applications. I am going to make the Folder on my Desktop called “**apps**” – the path to this folder is:

C:\Documents and Settings\csev\Desktop\apps

And then make a sub-folder in within **apps** called “**ae-01-trivial**” – the path to this folder would be:

C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial

Using a text editor such as JEdit (www.jedit.org), create a file called **app.yaml** in the **ae-01-trivial** folder with the following contents:

```
application: ae-01-trivial
```

```
version: 1
```

```
runtime: python
```

```
api_version: 1
```

```
handlers:
```

```
- url: /.*
```

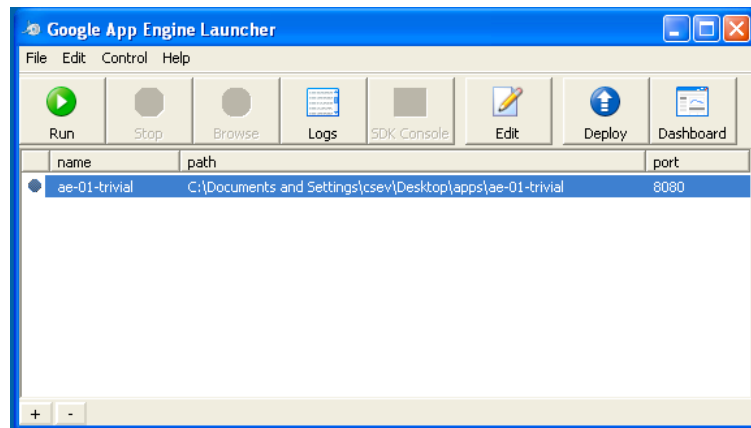
```
  script: index.py
```

Note: Please do not copy and paste these lines into your text editor – you might end up with strange characters – simply type them into your editor.

Then create a file in the **ae-01-trivial** folder called **index.py** with three lines in it:

```
print 'Content-Type:  
text/plain' print ' '  
print 'Hello World'
```

Then start the **GoogleAppEngineLauncher** program that can be found under **Applications**. Use the **File -> Add Existing Application** command and navigate into the **apps** directory and select the **ae-01-trivial** folder. Once you have added the application, select it so that you can control the application using the launcher.



Once you have selected your application and press **Run**. After a few moments your application will start and the launcher will show a little green icon next to your application. Then press **Browse** to open a browser pointing at your application which is running at **http://localhost:8080/**

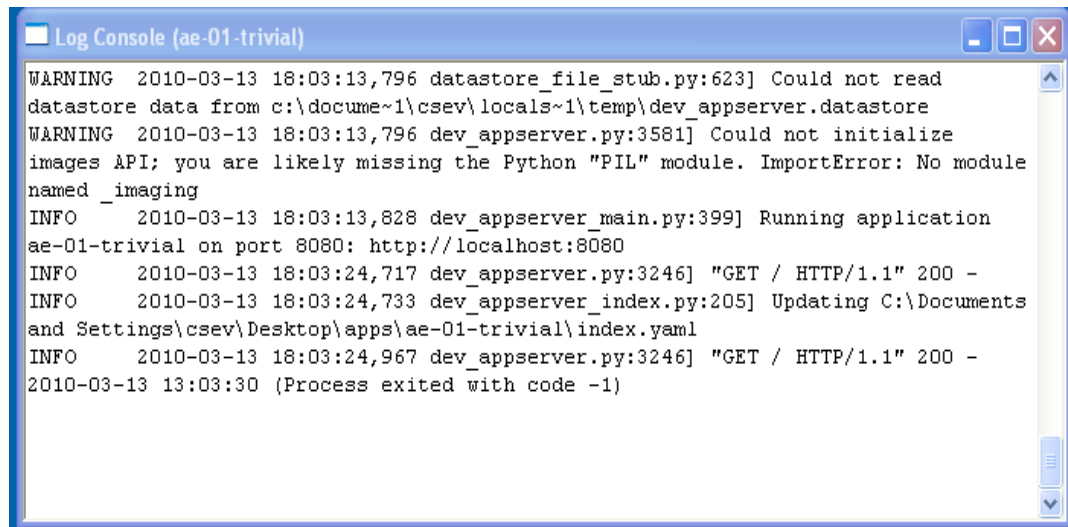
Paste **http://localhost:8080** into your browser and you should see your application as follows:



Just for fun, edit the **index.py** to change the name “Chuck” to your own name and press Refresh in the browser to verify your updates.

Watching the Log

You can watch the internal log of the actions that the web server is performing when you are interacting with your application in the browser. Select your application in the Launcher and press the **Logs** button to bring up a log window:

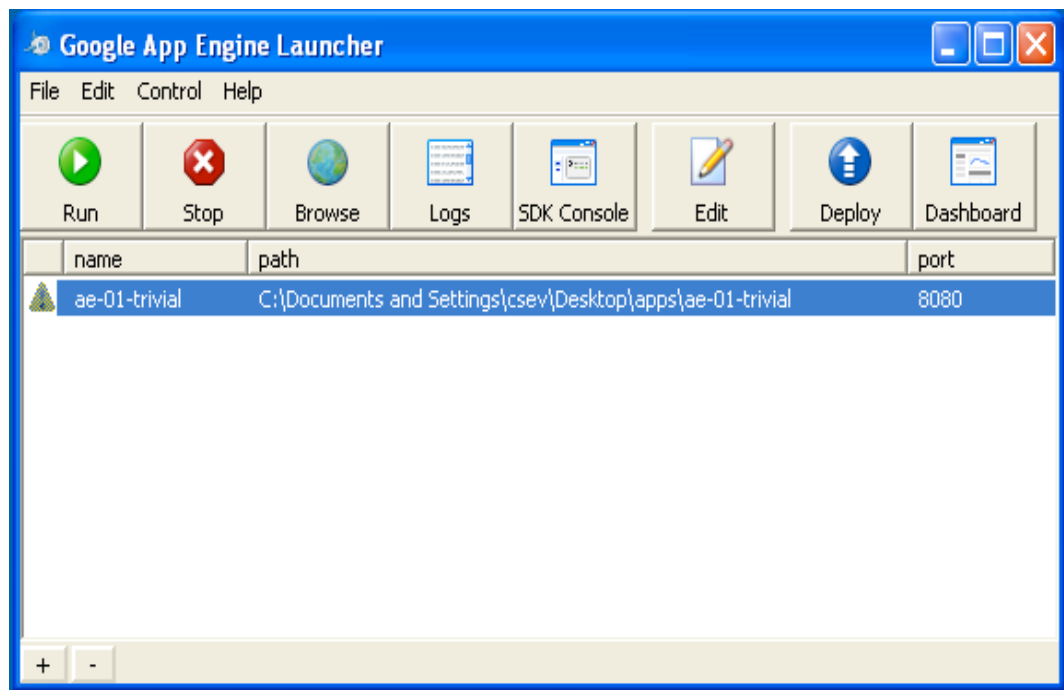


```
Log Console (ae-01-trivial)
WARNING 2010-03-13 18:03:13,796 datastore_file_stub.py:623] Could not read
datastore data from c:\docume~1\csev\locals~1\temp\dev_appserver.datastore
WARNING 2010-03-13 18:03:13,796 dev_appserver.py:3581] Could not initialize
images API; you are likely missing the Python "PIL" module. ImportError: No module
named _imaging
INFO 2010-03-13 18:03:13,828 dev_appserver_main.py:399] Running application
ae-01-trivial on port 8080: http://localhost:8080
INFO 2010-03-13 18:03:24,717 dev_appserver.py:3246] "GET / HTTP/1.1" 200 -
INFO 2010-03-13 18:03:24,733 dev_appserver_index.py:205] Updating C:\Documents
and Settings\csev\Desktop\apps\ae-01-trivial\index.yaml
INFO 2010-03-13 18:03:24,967 dev_appserver.py:3246] "GET / HTTP/1.1" 200 -
2010-03-13 13:03:30 (Process exited with code -1)
```

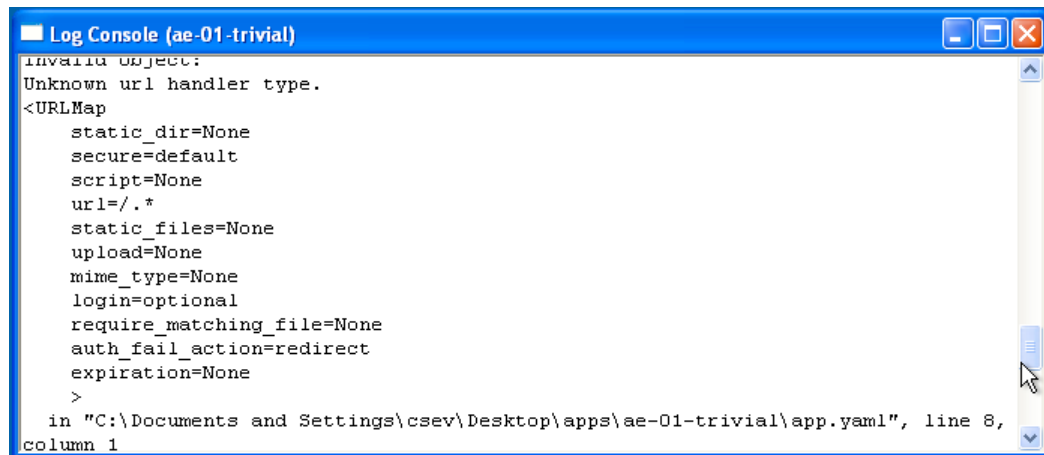
Each time you press **Refresh** in your browser – you can see it retrieving the output with a **GET** request.

Dealing With Errors

With two files to edit, there are two general categories of errors that you may encounter. If you make a mistake on the **app.yaml** file, the App Engine will not start and your launcher will show a yellow icon near your application:



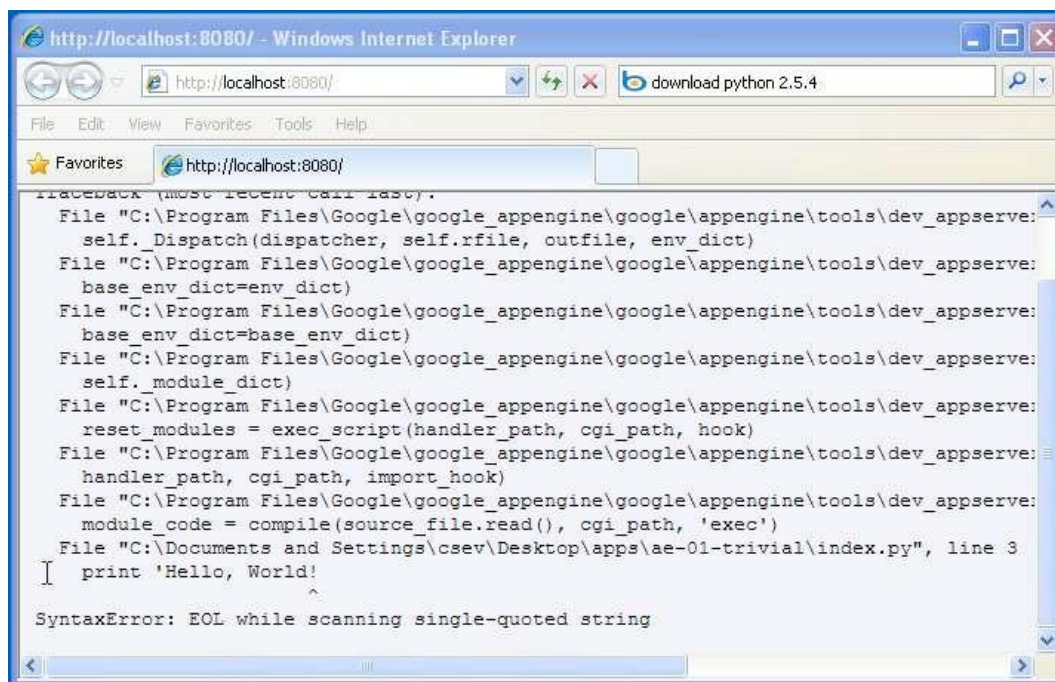
To get more detail on what is going wrong, take a look at the log for the application:



```
invalid object:
Unknown url handler type.
<URLMap
  static_dir=None
  secure=default
  script=None
  url=/. *
  static_files=None
  upload=None
  mime_type=None
  login=optional
  require_matching_file=None
  auth_fail_action=redirect
  expiration=None
>
in "C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial\app.yaml", line 8,
column 1
```

In this instance — the mistake is mis-indenting the last line in the **app.yaml** (line 8).

If you make a syntax error in the **index.py** file, a Python trace back error will appear in your browser.



```
http://localhost:8080/ - Windows Internet Explorer
http://localhost:8080/
File Edit View Favorites Tools Help
http://localhost:8080/
Traceback (most recent call last):
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    self.Dispatch(dispatcher, self.rfile, outfile, env_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    base_env_dict=env_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    base_env_dict=base_env_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    self.module_dict)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    reset_modules = exec_script(handler_path, cgi_path, hook)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    handler_path, cgi_path, import_hook)
  File "C:\Program Files\Google\google_appengine\google\appengine\tools\dev_appserve:
    module_code = compile(source_file.read(), cgi_path, 'exec')
  File "C:\Documents and Settings\csev\Desktop\apps\ae-01-trivial\index.py", line 3
    print 'Hello, World!'
      ^
SyntaxError: EOL while scanning single-quoted string
```

The error you need to see is likely to be the last few lines of the output — in this case I made a Python syntax error on line one of our one-line application.

Reference: http://en.wikipedia.org/wiki/Stack_trace

When you make a mistake in the **app.yaml** file—you must fix the mistake and attempt to start the application again.

If you make a mistake in a file like **index.py**, you can simply fix the file and press refresh in your browser—there is no need to restart the server.

Shutting Down the Server

To shut down the server, use the Launcher, select your application and press the **Stop** button.

RESULT:

Thus the Google App Engine was installed and *hello world* app and other simple web applications were created using python/java.

Ex. No:4 Date :	Use GAE launcher to launch the web applications.
----------------------------------	---

AIM:

To use GAE launcher to launch the web applications.

PROCEDURE:

Step 1. Download the basic housekeeping stuff

No matter what platform you build products on, there is always some housekeeping stuff you need to put in place before you can hit the ground running. And deploying apps within the Google App Engine is no exception.

1. Download Python 2.7

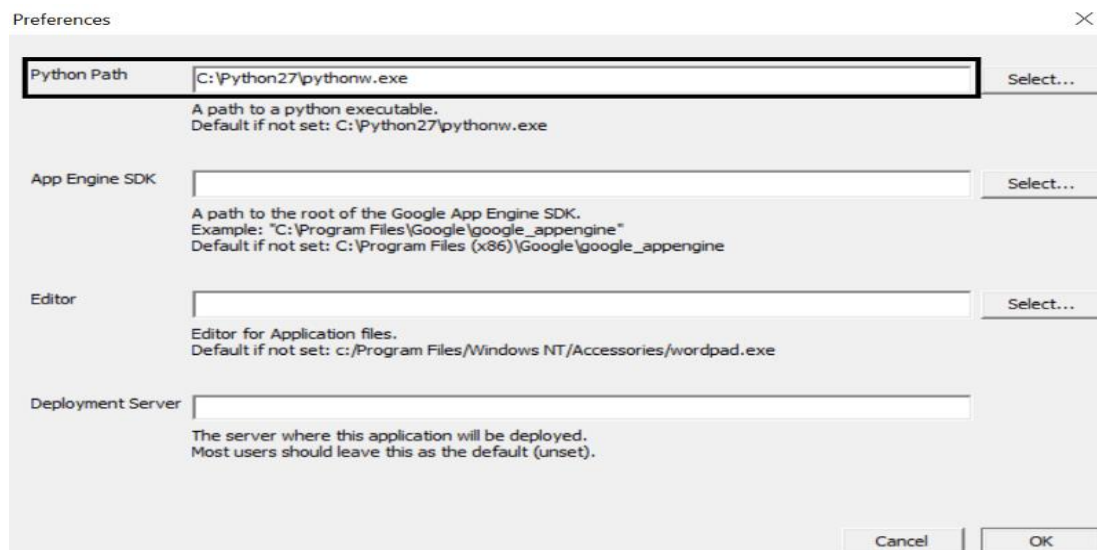
As of when this article was written, the Google App Engine standard environment supports Python only upto version 2.7. However, it is only a matter of time before support for Python 3.x is added. You can check the App Engine docs for the latest info.

2. Download Google Cloud SDK

This will allow you to fork apps onto your local machine, make changes (edit and develop the app), and deploy your app back to the cloud.

3. Set the Python path in the Google App Engine launcher

After downloading the SDK, launch the App Engine launcher, go to Edit -> Preferences and make sure you set the path for where you installed Python in step 1 above.



Set the Python path in Google App Engine launcher

That's all you need. Your local machine should now be ready to build webapps.

Step 2. App Engine sign-up

This is often the most confusing part of the entire setup. Things you should know when you sign-up:

1. Currently, App Engine offers a free trial for one year.
2. The trial includes \$300 of credit that can be used during the one year trial period.
3. You will need to add a credit card to sign-up (for verification purposes).
4. You will not be charged during the sign-up process.
5. You will not be charged during the trial period as long as you do not cross the credit limit offered.

Here are the steps you need to follow to sign-up:

1. Go to the [Google Cloud](#) landing page
2. Follow the sign-up process and go to your App Engine dashboard

Most of the hard work is complete after a successful sign-up.

Step 3. Create a new project

The next step is to create a new Python project that you can work on. Follow the screenshots below to create a new project.

Launch the new project wizard.

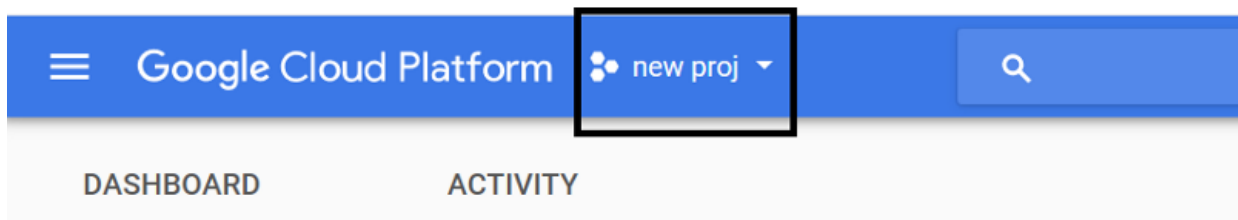
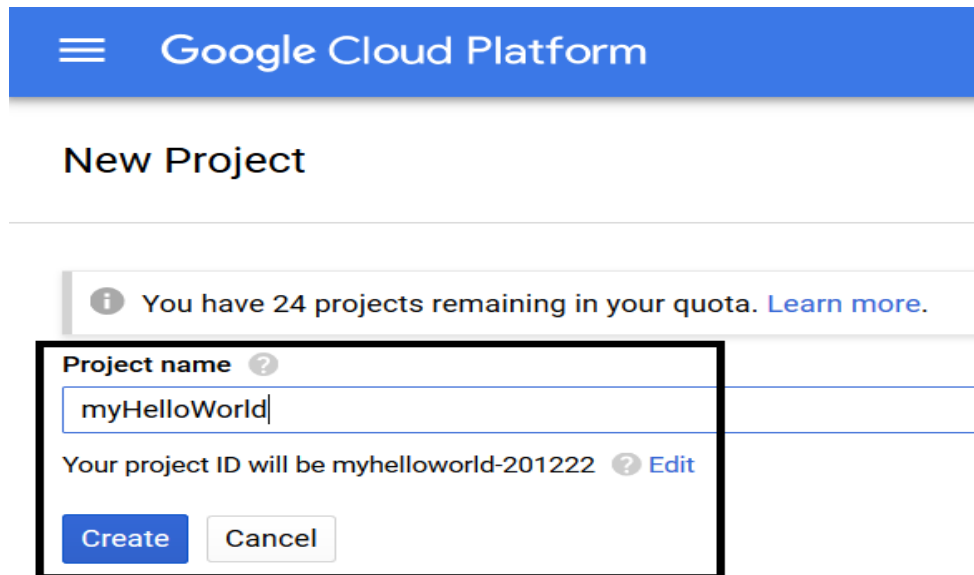


Image courtesy. <https://console.cloud.google.com/home>



Image courtesy <https://console.cloud.google.com/home>

Give your app a name and make a note of your project ID.



Google Cloud Platform

New Project

i You have 24 projects remaining in your quota. [Learn more.](#)

Project name *?*
myHelloWorld

Your project ID will be myhelloworld-201222 *?* [Edit](#)

Create Cancel

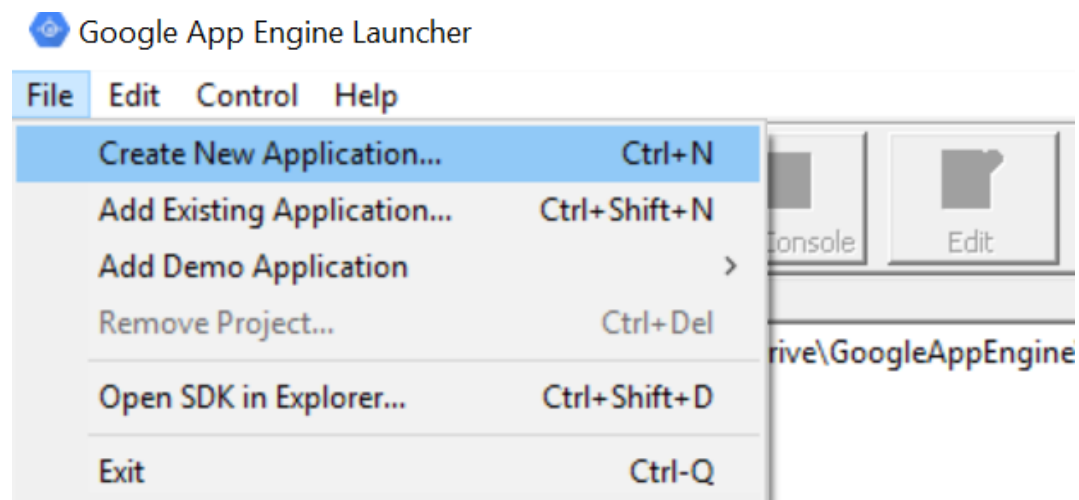
Image courtesy. <https://console.cloud.google.com/home>

Hit the create button and Google should take a few minutes to set up all that is necessary for your newly created app.

Step 4. Fork the app to develop it locally

The next step in the process is to fork the app on your local machine. This will allow you to make changes to the app locally and deploy it whenever you wish to.

Go to Google App Engine launcher and create a new application.



Enter the project ID of your newly created app. Also, provide the folder (local destination) where you wish to store the app locally. Make sure you select the Python 2.7 as your runtime engine.

Add New Application ×

Application Settings

Application Name: myhelloworld-201222

Parent Directory: Browse...

Runtime: Python 2.7 ▼

Port: 9080

Admin Port: 8001

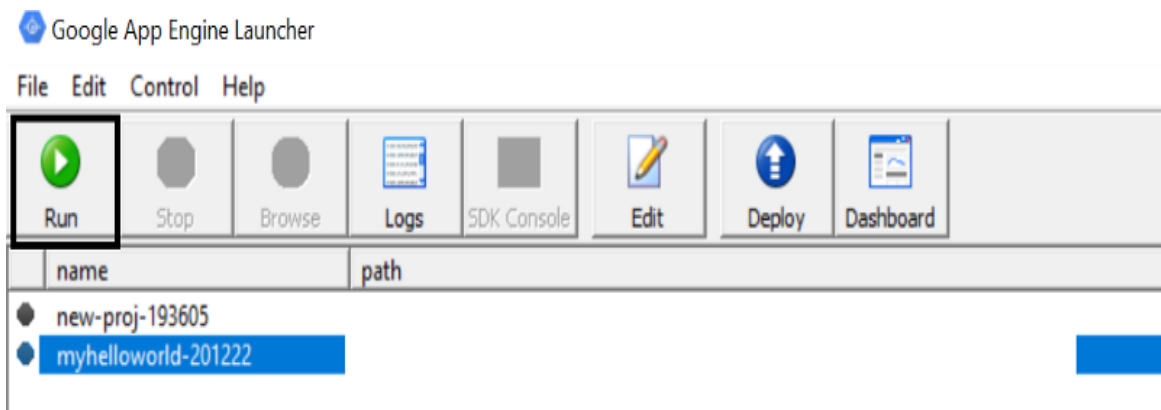
Create Cancel

Hit the create button, and you should see your app listed on the window that follows. You should also check that you now see some files in your local storage (the directory you chose in the screenshot above) after this step.

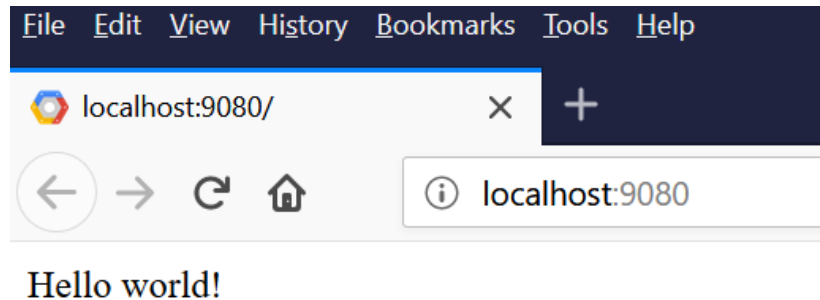
Step 5. Run the app locally

Before you go ahead and make some changes to the app, it is important to check whether or not you have executed all the above steps correctly. This can be done by simply running the app locally.

Select the app and hit the run button on the window.



Wait for a few seconds until you can hit the **Browse** button. Once the **Browse** button becomes clickable, click it. This should take you to the browser, and you should see the hello world text appear in your browser window. Alternatively, you can manually go to the browser and use the port specified to access the app.



As long as you see the above screen, you are all set.

Step 6. Understand the app structure

It is finally time to look at the lines of code which are running this webapp. Open your app folder in the text editor of your choice. I recommend [Sublime text](#) or [VS Code](#). However, feel free to choose the one you prefer.

Here is a description of the various files.

app.yaml

This file is a basic markup file that stores information (some metadata) about the app. It is important to note the following crucial parts of the file.

1. **application**

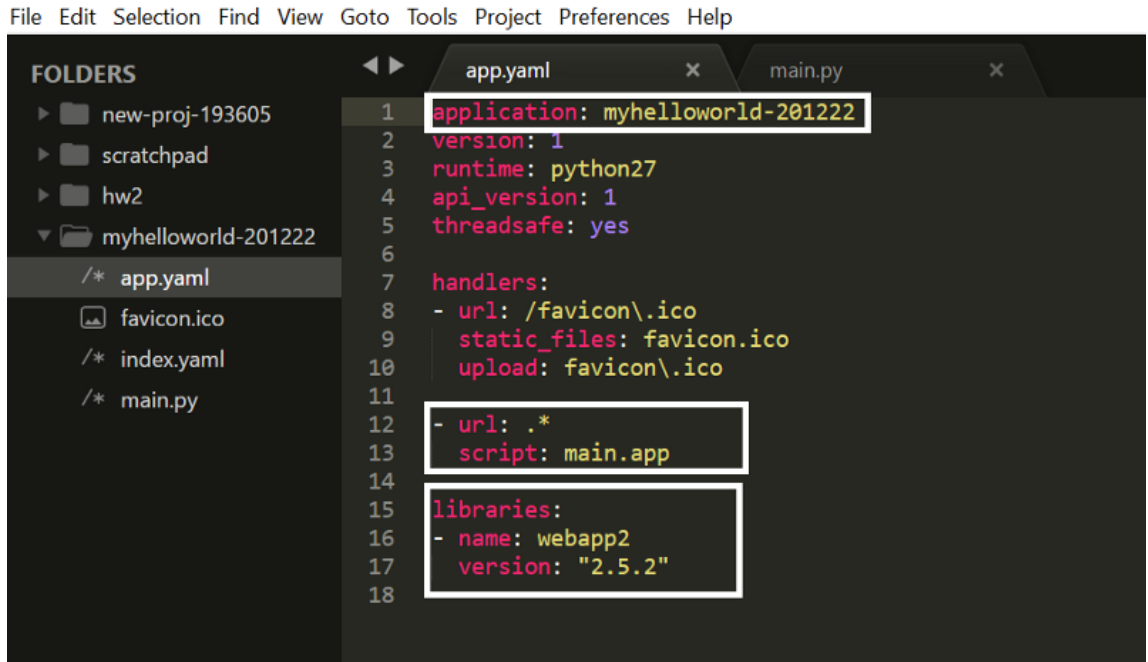
This is the project ID which you should never change. This is the unique identifier for the app

2. **url -> script**

This is the homepage for the app. In other words, this file will be rendered in your browser when you launch the app

3. **libraries**

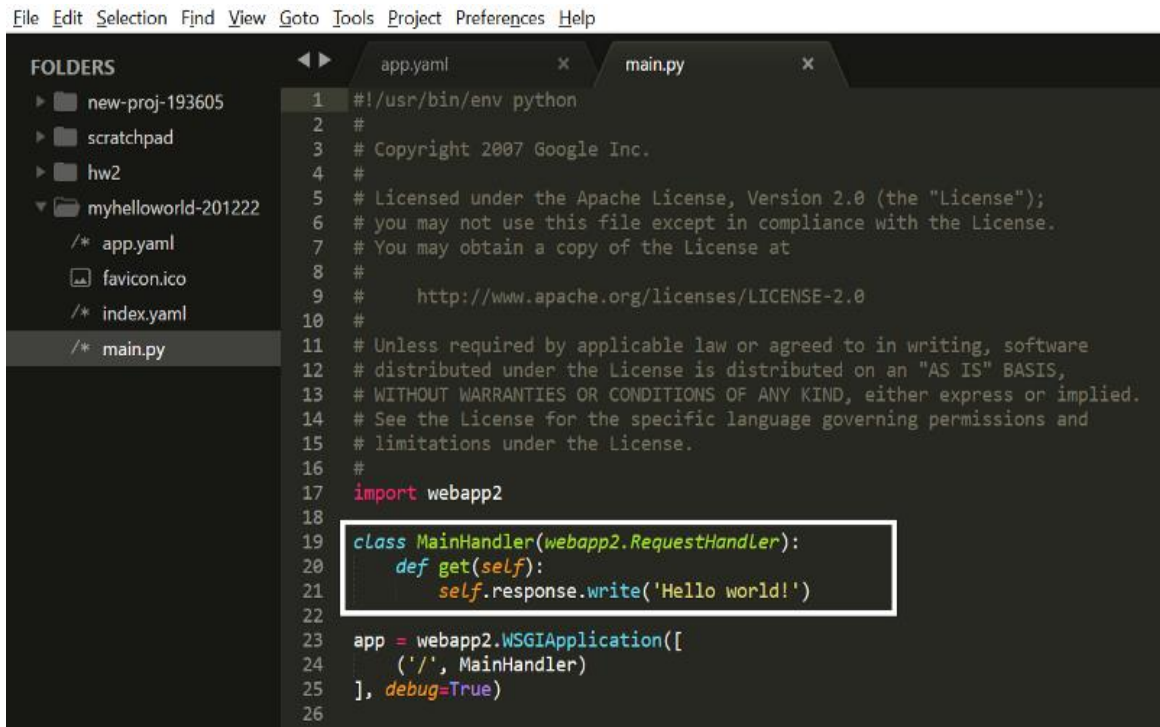
This is where you can include external libraries to use within the webapp



app.yaml file in the webapp folder

main.py

This is the homepage of the app (as discussed above). Note that the hello world text in the browser window (step 5) is due to the code you see highlighted below.



main.py file in the webapp folder

Step 7. Make your changes and deploy the new app

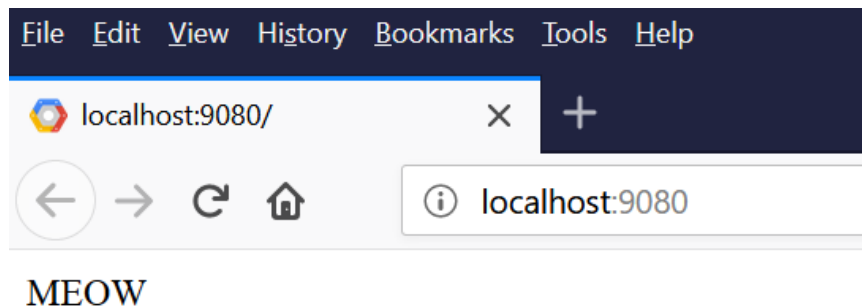
No hello world app is ever complete without the developer changing the hello world text to something else just to make sure that everything happening behind the scenes is working as it should.

Go ahead and change the text in the above screenshot to something else.

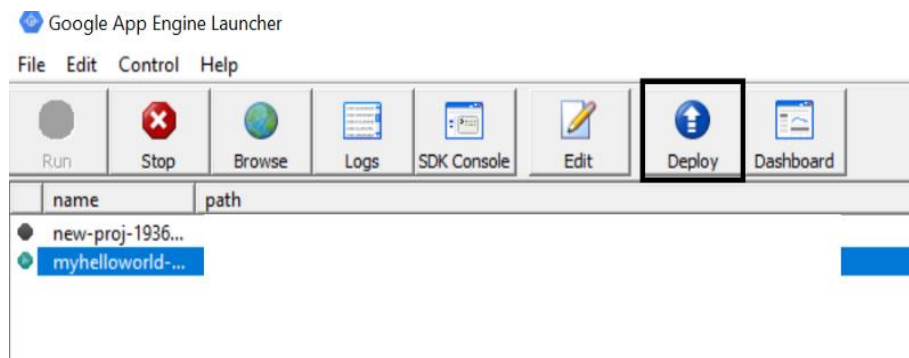
```
17 import webapp2
18
19 class MainHandler(webapp2.RequestHandler):
20     def get(self):
21         self.response.write('MEOW')
22
23 app = webapp2.WSGIApplication([
24     ('/', MainHandler)
25 ], debug=True)
26
```

main.py file in the webapp folder

Save the changes, go to the browser and refresh the page. You should see the page with the text “MEOW” displayed.



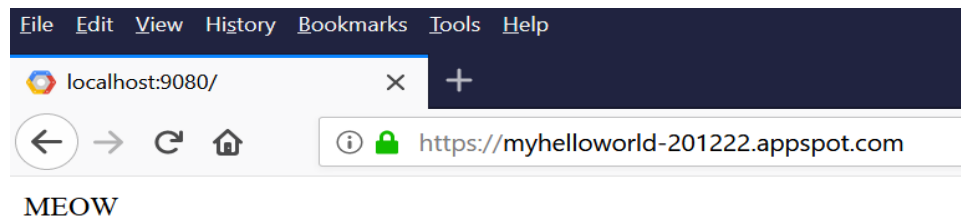
Finally, it is time to deploy your changes to the cloud to make them globally accessible via a URL. Go to the App Engine launcher, select the app, and hit the **Deploy** button.



This will ensure your app gets deployed onto Google Cloud. To check whether or not everything worked just fine, go to the URL below:

`https://<yourProjectID>.appspot.com/`

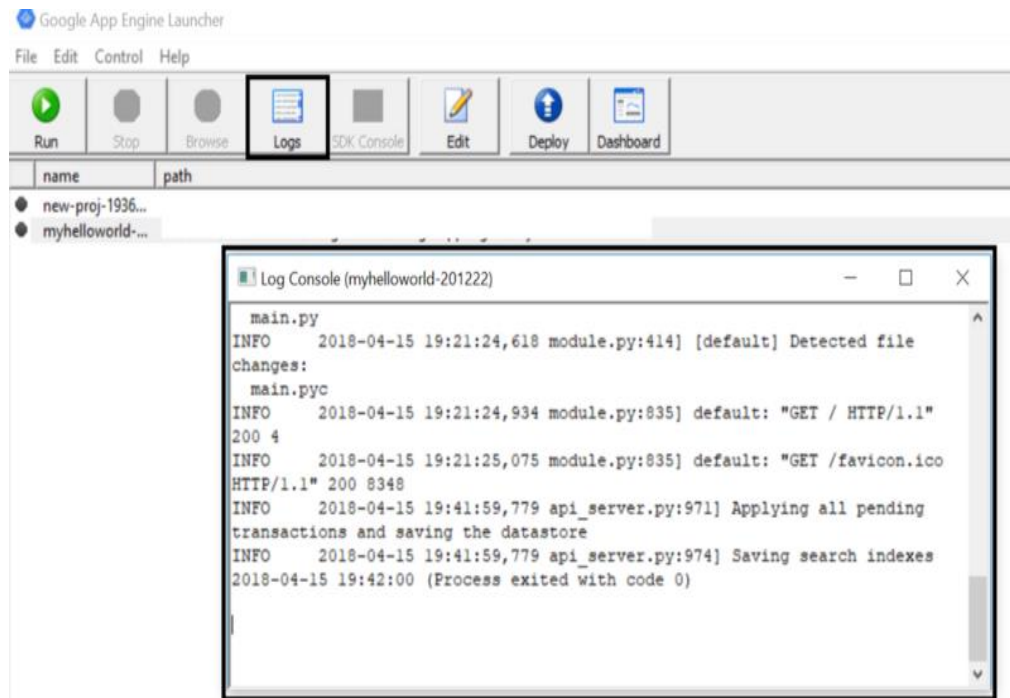
You should see the exact same window as above, expect now, it is a URL that is globally accessible.



Step 8. Misc

Congratulations, you've finally gotten your first Python webapp deployed on the Google App Engine. Here are some other points which you may find useful.

1. Jinja 2 is an amazing front end templating library for Python that can do some cool stuff, such as passing objects from Python to HTML, using for loops, if conditions, and so on directly out of the box
2. Here's a very useful Udacity course on web development that I have personally found quite resourceful
3. Viewing the logs while running your webapp can be handy to debug and also discover some bugs on the fly



Log console of the webapp

RESULT:

Thus the GAE launcher is used successfully to launch the web applications.

Ex. No:5 Date:	Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.
---------------------------------	---

AIM:

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

PROCEDURE:

How to use CloudSim in Eclipse

CloudSim is written in Java. The knowledge you need to use CloudSim is basic Java programming and some basics about cloud computing. Knowledge of programming IDEs such as Eclipse or NetBeans is also helpful. It is a library and, hence, CloudSim does not have to be installed. Normally, you can unpack the downloaded package in any directory, add it to the Java classpath and it is ready to be used. Please verify whether Java is available on your system.

To use CloudSim in Eclipse:

1. Download CloudSim installable files from
<https://code.google.com/p/cloudsim/downloads/list> and unzip
2. Open Eclipse
3. Create a new Java Project: File -> New
4. Import an unpacked CloudSim project into the new Java Project
5. The first step is to initialise the CloudSim package by initialising the CloudSim library, as follows:

```
CloudSim.init(num_user, calendar, trace_flag)
```

6. Data centres are the resource providers in CloudSim; hence, creation of data centres is a second step. To create Datacenter, you need the DatacenterCharacteristics object that stores the properties of a data centre such as architecture, OS, list of machines, allocation policy that covers the time or spaceshared, the time zone and its price:

```
Datacenter datacenter9883 = new Datacenter(name, characteristics, new  
VmAllocationPolicySimple(hostList), s)
```

7. The third step is to create a broker:

```
DatacenterBroker broker = createBroker();
```

8. The fourth step is to create one virtual machine unique ID of the VM, userId ID of the VM's owner, mips, number Of Pes amount of CPUs, amount of RAM, amount of bandwidth, amount of storage, virtual machine monitor, and cloudletScheduler policy for cloudlets:

```
Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size,  
vmm, new CloudletSchedulerTimeShared())
```


9. Submit the VM list to the broker:

```
broker.submitVmList(vmlist)
```

10. Create a cloudlet with length, file size, output size, and utilisation model:

```
Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize,  
utilizationModel, utilizationMode)
```

11. Submit the cloudlet list to the broker:

```
broker.submitCloudletList(cloudletList)
```

12. Start the simulation:

```
CloudSim.startSimulation()
```

Starting CloudSimExample1...

Initialising...

Starting CloudSim version 3.0

Datacenter_0 is starting...

[illegible]

Broker is starting...

Entities started.

```

: Broker: Cloud Resource List received with 1 resource(s)

```

0.0: Broker: Trying to Create VM #0 in Datacenter_0

```

: Broker: VM #0 has been created in Datacenter #2, Host #0

```

0.1: Broker: Sending cloudlet 0 to VM #0

400.1: Broker: Cloudlet 0 received

```
: Broker: All Cloudlets executed. Finishing...
```

400.1: Broker: Destroying VM #0

Broker is shutting down...

Simulation: No more future events

CloudInformationService: Notify all CloudSim entities for shutting down.

Datacenter_0 is shutting down...

Broker is shutting down...

Simulation completed.

Simulation completed.

===== OUTPUT =====

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time
-------------	--------	----------------	-------	------	------------	-------------

0	SUCCESS	2	0	400	0.1	400.1
---	---------	---	---	-----	-----	-------

****Datacenter: Datacenter_0**** User id Debt

3 35.6

```
CloudSimExample1 finished!
```

RESULT:

Thus a cloud scenario using CloudSim was simulated and a scheduling algorithm that is not present in CloudSim was executed successfully.

Ex. No:6 Date:	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. You can copy few (or more) lines with **copy & paste** mechanism.
For this you need to share clipboard between host OS and guest OS, installing **Guest Addition** on both the virtual machines (probably setting *bidirectional* and restarting them). You *copy* from *guest OS* in the clipboard that is shared with the *host OS*.
Then you *paste* from the *host OS* to the second *guest OS*.
2. You can enable **drag and drop** too with the same method (Click on the machine, settings, general, advanced, drag and drop: set to *bidirectional*)
3. You can have **common Shared Folders** on both virtual machines and use one of the directory shared as buffer to copy.
Installing **Guest Additions** you have the possibility to set Shared Folders too. As you put a file in a shared folder from *host OS* or from *guest OS*, is immediately visible to the other. (Keep in mind that can arise some problems for date/time of the files when there are different clock settings on the different virtual machines).
If you use the same folder shared on more machines you can exchange files directly copying them in this folder.
4. You can use **usual method to copy files between 2 different computer** with client-server application. (e.g. scp with sshd active for linux, winscp... you can get some info about SSH servers e.g. here)
You need an active server (sshd) on the receiving machine and a client on the sending machine. Of course you need to have the authorization setted (via password or, better, via an automatic authentication method).
Note: many Linux/Ubuntu distribution install sshd by default: you can see if it is running with `pgrep sshd` from a shell. You can install with `sudo apt-get install openssh-server`.
5. You can **mount part of the file system** of a virtual machine via NFS or SSHFS on the other, or you can **share file and directory** with Samba.
You may find interesting the article *Sharing files between guest and host without VirtualBox shared folders* with detailed step by step instructions.

You should remember that you are deling with a little network of machines with different operative systems, and in particular:

- Each virtual machine has its own operative system running on and acts as a physical machine.

- Each virtual machine is an instance of a program *owned* by an *user* in the hosting operative system and should undergo the restrictions of the *user* in the *hosting OS*.
E.g Let we say that Hastur and Meow are users of the hosting machine, but they did not allow each other to see their directories (no read/write/execute authorization). When each of them run a virtual machine, for the hosting OS those virtual machine are two normal programs owned by Hastur and Meow and cannot see the private directory of the other user. This is a restriction due to the *hosting OS*. It's easy to overcome it: it's enough to give authorization to read/write/execute to a directory or to choose a different directory in which both users can read/write/execute.
- Windows likes mouse and Linux fingers.
I mean I suggest you to enable *Drag & drop* to be cosy with the Windows machines and the *Shared folders* or to be cosy with Linux.
When you will need to be fast with Linux **you will feel the need** of ssh-keygen and to Generate once SSH Keys to copy files on/from a remote machine without writing password anymore. In this way it functions bash auto-completion remotely too!

RESULT:

Thus a procedure to transfer the files from one virtual machine to another virtual machine was found and the files are transferred successfully.

Ex. No:7 Date:	Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)
---------------------------------	--

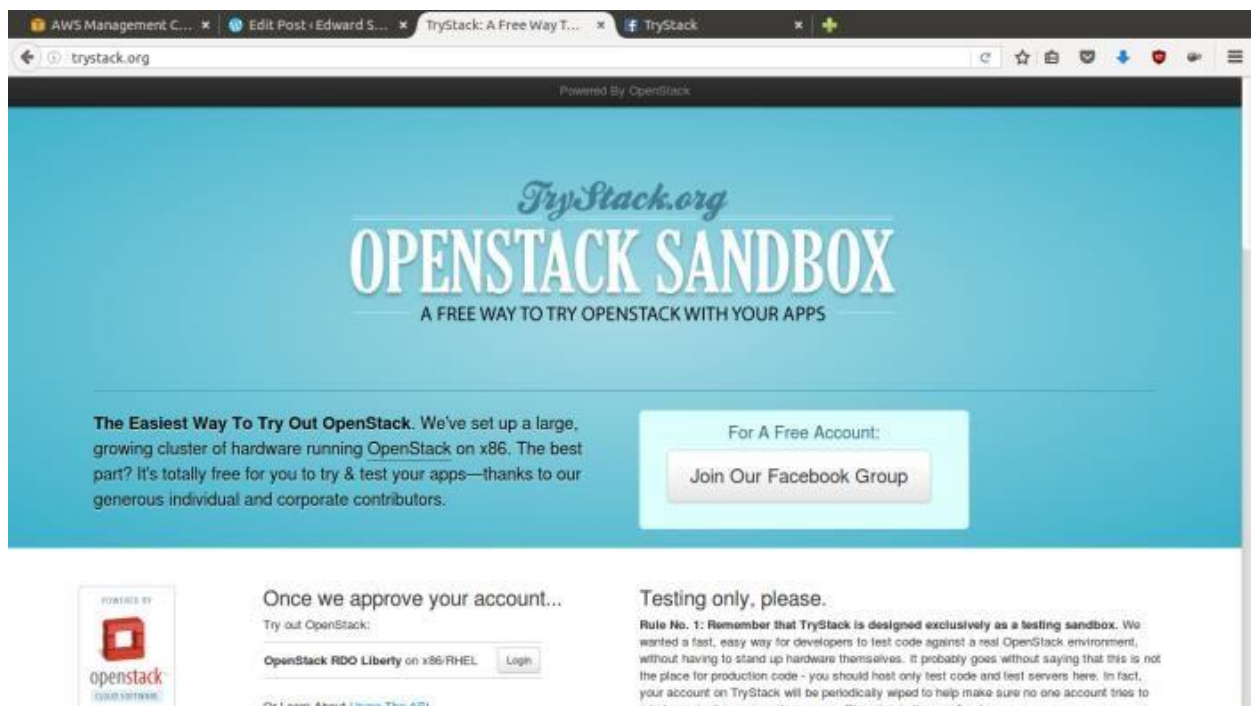
AIM:

To find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)

PROCEDURE:

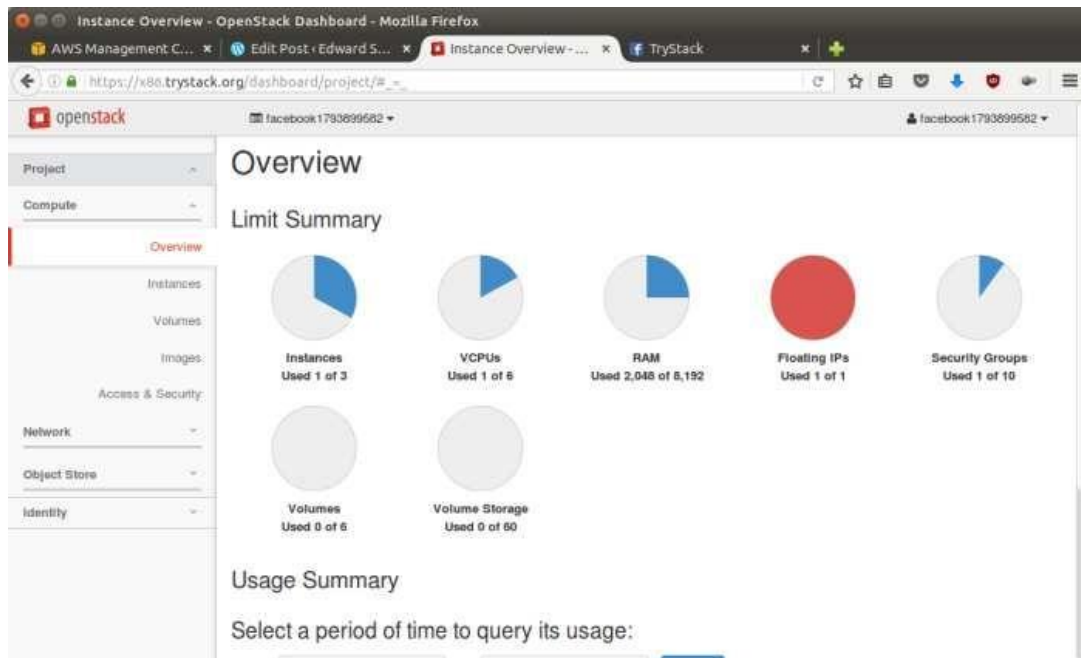
OpenStack is an open-source software cloud computing platform. OpenStack is primarily used for deploying an infrastructure as a service (IaaS) solution like Amazon Web Service (AWS). In other words, you can *make your own AWS* by using OpenStack. If you want to try out OpenStack, **TryStack** is the easiest and free way to do it.

In order to try OpenStack in TryStack, you must register yourself by joining TryStack Facebook Group. The acceptance of group needs a couple days because it's approved manually. After you have been accepted in the TryStack Group, you can log in TryStack.



TryStack.org Homepage

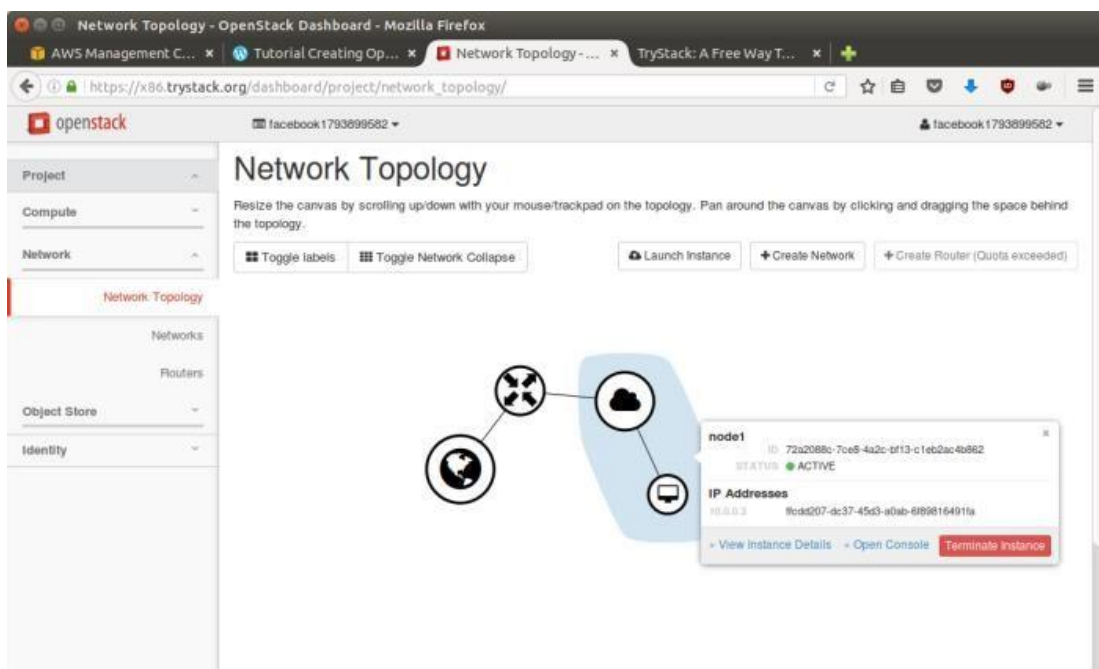
I assume that you already join to the Facebook Group and login to the dashboard. After you log in to the TryStack, you will see the Compute Dashboard like:



OpenStack Compute Dashboard

Overview: What we will do?

In this post, I will show you how to run an OpenStack instance. The instance will be accessible through the internet (have a public IP address). The final topology will like:



Network topology

As you see from the image above, the instance will be connected to a local network and the local network will be connected to internet.

Step 1: Create Network

Network? Yes, the network in here is our own local network. So, your instances will be not mixed up with the others. You can imagine this as your own LAN (Local Area Network) in the cloud.

1. Go to **Network > Networks** and then click **Create Network**.
2. In **Network** tab, fill **Network Name** for example `internal` and then click **Next**.
3. In **Subnet** tab,
 1. Fill **Network Address** with appropriate CIDR, for example `192.168.1.0/24`. Use private network CIDR block as the best practice.
 2. Select **IP Version** with appropriate IP version, in this case `IPv4`.
 3. Click **Next**.
4. In **Subnet Details** tab, fill **DNS Name Servers** with `8.8.8.8` (Google DNS) and then click **Create**.

Step 2: Create Instance

Now, we will create an instance. The instance is a virtual machine in the cloud, like AWS EC2. You need the instance to connect to the network that we just created in the previous step.

1. Go to **Compute > Instances** and then click **Launch Instance**.
2. In **Details** tab,
 1. Fill **Instance Name**, for example `Ubuntu 1`.
 2. Select **Flavor**, for example `m1.medium`.
 3. Fill **Instance Count** with `1`.
 4. Select **Instance Boot Source** with **Boot from Image**.
 5. Select **Image Name** with **Ubuntu 14.04 amd64 (243.7 MB)** if you want install Ubuntu 14.04 in your virtual machine.
3. In **Access & Security** tab,
 1. Click **[+]** button of **Key Pair** to import key pair. This key pair is a public and private key that we will use to connect to the instance from our machine.
 2. In **Import Key Pair** dialog,
 1. Fill **Key Pair Name** with your machine name (for example `Edward-Key`).
 2. Fill **Public Key** with your **SSH public key** (usually is in `~/.ssh/id_rsa.pub`). See description in Import Key Pair dialog box for more information. If you are using Windows, you can use **Puttygen** to generate key pair.
 3. Click **Import key pair**.
 3. In **Security Groups**, mark/check **default**.
4. In **Networking** tab,
 1. In **Selected Networks**, select network that have been created in Step 1, for example `internal`.
5. Click **Launch**.
6. If you want to create multiple instances, you can repeat step 1-5. I created one more instance with instance name `Ubuntu 2`.

Step 3: Create Router

I guess you already know what router is. In the step 1, we created our network, but it is isolated. It doesn't connect to the internet. To make our network has an internet connection, we need a router that running as the gateway to the internet.

1. Go to **Network > Routers** and then click **Create Router**.
2. Fill **Router Name** for example `router1` and then click **Create router**.
3. Click on your **router name link**, for example `router1`, **Router Details** page.
4. Click **Set Gateway** button in upper right:
 1. Select **External networks** with **external**.
 2. Then **OK**.
5. Click **Add Interface** button.
 1. Select **Subnet** with the network that you have been created in Step 1.
 2. Click **Add interface**.
6. Go to **Network > Network Topology**. You will see the network topology. In the example, there are two network, i.e. external and internal, those are bridged by a router. There are instances those are joined to internal network.

Step 4: Configure Floating IP Address

Floating IP address is public IP address. It makes your instance is accessible from the internet. When you launch your instance, the instance will have a private network IP, but no public IP. In OpenStack, the public IPs is collected in a pool and managed by admin (in our case is TryStack). You need to request a public (floating) IP address to be assigned to your instance.

1. Go to **Compute > Instance**.
2. In one of your instances, click **More > Associate Floating IP**.
3. In **IP Address**, click Plus [+].
4. Select **Pool** to **external** and then click **Allocate IP**.
5. Click **Associate**.
6. Now you will get a public IP, e.g. 8.21.28.120, for your instance.

Step 5: Configure Access & Security

OpenStack has a feature like a firewall. It can whitelist/blacklist your in/out connection. It is called *Security Group*.

1. Go to **Compute > Access & Security** and then open **Security Groups** tab.
2. In **default** row, click **Manage Rules**.
3. Click **Add Rule**, choose **ALL ICMP** rule to enable ping into your instance, and then click **Add**.
4. Click **Add Rule**, choose **HTTP** rule to open HTTP port (port 80), and then click **Add**.
5. Click **Add Rule**, choose **SSH** rule to open SSH port (port 22), and then click **Add**.
6. You can open other ports by creating new rules.

Step 6: SSH to Your Instance

Now, you can SSH your instances to the floating IP address that you got in the step 4. If you are using Ubuntu image, the SSH user will be ubuntu.

RESULT:

Thus the procedure to launch virtual machine using trystack (Online Openstack Demo Version) was executed successfully.

Ex. No:8 Date:	Install Hadoop single node cluster and run simple applications like wordcount.
---------------------------------	---

AIM:

To install Hadoop single node cluster and run simple applications like wordcount.

PROCEDURE:

Install Hadoop

Step 1: [Click here](#) to download the Java 8 Package. Save this file in your home directory.

Step 2: Extract the Java Tar File.

Command: `tar -xvf jdk-8u101-linux-i586.tar.gz`

```
edureka@localhost:~$ tar -xvf jdk-8u101-linux-i586.tar.gz
```

Fig: Hadoop Installation – Extracting Java Files

Step 3: Download the Hadoop 2.7.3 Package.

Command: `wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz`

```
edureka@localhost:~$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

Fig: Hadoop Installation – Downloading Hadoop

Step 4: Extract the Hadoop tar File.

Command: `tar -xvf hadoop-2.7.3.tar.gz`

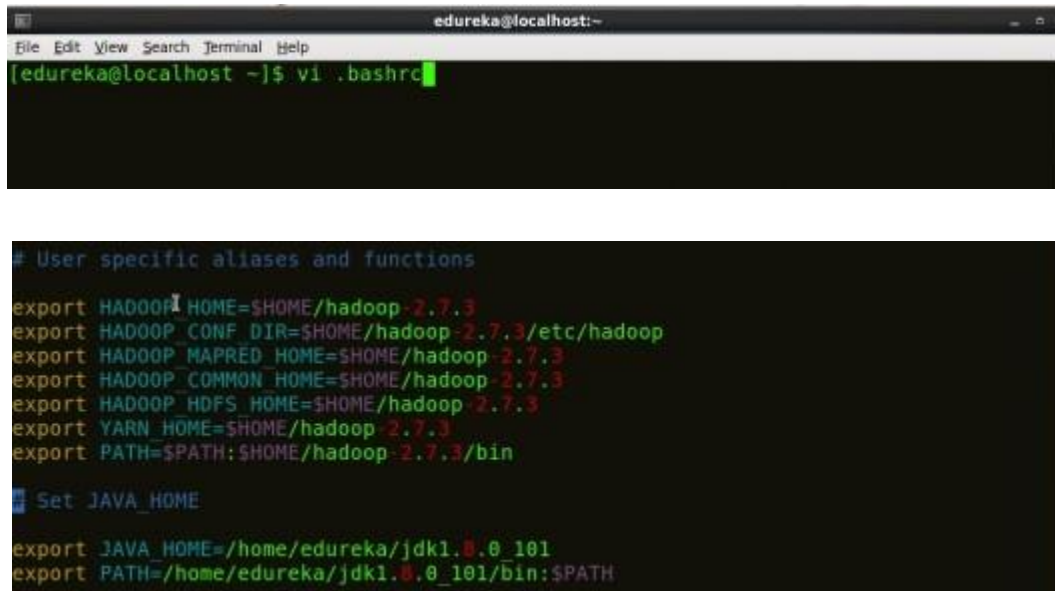
```
edureka@localhost:~ (on localhost.localdomain)$ tar -xvf hadoop-2.7.3.tar.gz
```

Fig: Hadoop Installation – Extracting Hadoop Files Step 5:

Add the Hadoop and Java paths in the bash file (.bashrc). Open. **bashrc** file.

Now, add Hadoop and Java Path as shown below.

Command: vi .bashrc



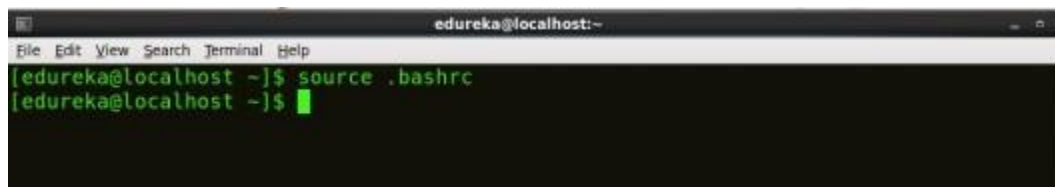
```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ vi .bashrc  
  
# User specific aliases and functions  
  
export HADOOP_HOME=$HOME/hadoop-2.7.3  
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop  
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3  
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3  
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3  
export YARN_HOME=$HOME/hadoop-2.7.3  
export PATH=$PATH:$HOME/hadoop-2.7.3/bin  
  
# Set JAVA_HOME  
  
export JAVA_HOME=/home/edureka/jdk1.8.0_101  
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

Fig: Hadoop Installation – Setting Environment Variable

Then, save the bash file and close it.

For applying all these changes to the current Terminal, execute the source command.

Command: source .bashrc

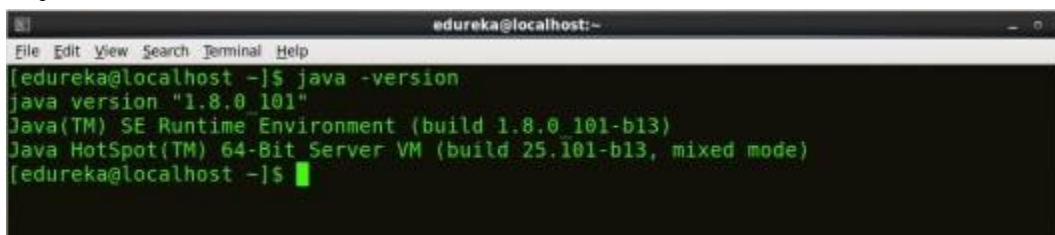


```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ source .bashrc  
[edureka@localhost ~]$
```

Fig: Hadoop Installation – Refreshing environment variables

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the java -version and hadoop version commands.

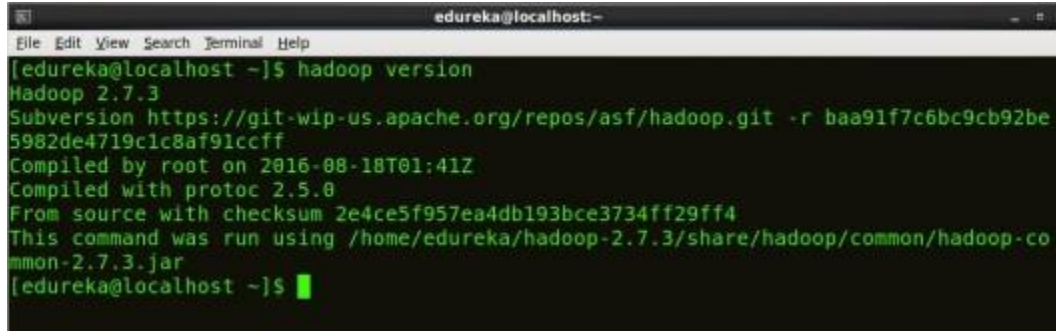
Command: java -version



```
edureka@localhost:~  
File Edit View Search Terminal Help  
[edureka@localhost ~]$ java -version  
java version "1.8.0_101"  
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)  
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)  
[edureka@localhost ~]$
```

Fig: Hadoop Installation – Checking Java Version

Command: `hadoop version`



```
edureka@localhost:~$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /home/edureka/hadoop-2.7.3/share/hadoop/common/hadoop-common-2.7.3.jar
[edureka@localhost ~]$
```

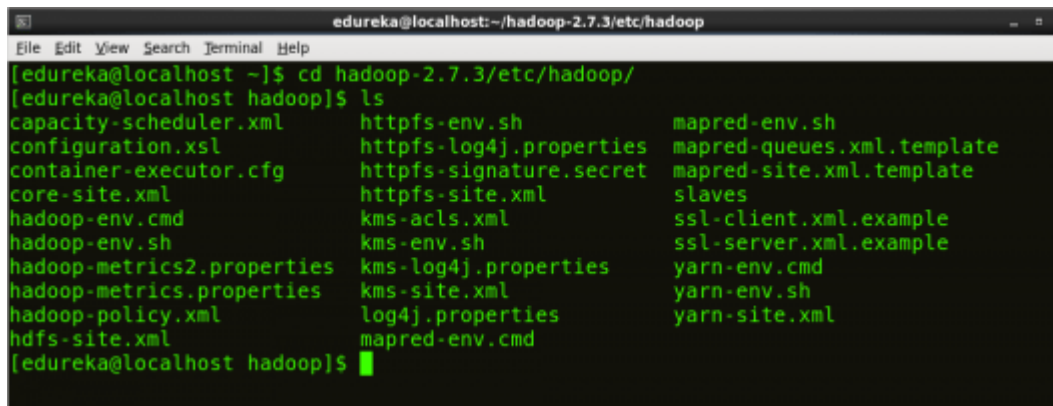
Fig: Hadoop Installation – Checking Hadoop Version

Step 6: Edit the **Hadoop Configuration files**.

Command: `cd hadoop-2.7.3/etc/hadoop/`

Command: `ls`

All the Hadoop configuration files are located in **hadoop-2.7.3/etc/hadoop** directory as you can see in the snapshot below:



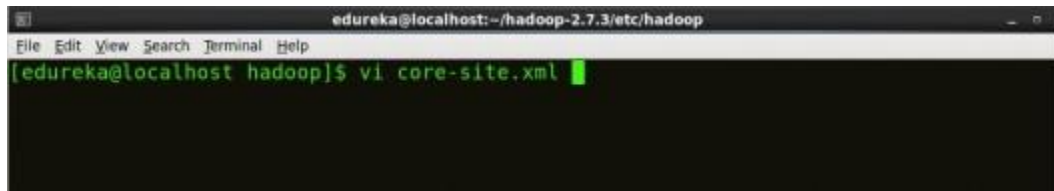
```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop$ ls
capacity-scheduler.xml  httpfs-env.sh          mapred-env.sh
configuration.xml       httpfs-log4j.properties mapred-queues.xml.template
container-executor.cfg  httpfs-signature.secret mapred-site.xml.template
core-site.xml           httpfs-site.xml        slaves
hadoop-env.cmd          kms-acls.xml           ssl-client.xml.example
hadoop-env.sh           kms-env.sh             ssl-server.xml.example
hadoop-metrics2.properties kms-log4j.properties  yarn-env.cmd
hadoop-metrics.properties kms-site.xml            yarn-env.sh
hadoop-policy.xml       log4j.properties       yarn-site.xml
hdfs-site.xml           mapred-env.cmd
```

Fig: Hadoop Installation – Hadoop Configuration Files

Step 7: Open *core-site.xml* and edit the property mentioned below inside configuration tag:

core-site.xml informs Hadoop daemon where NameNode runs in the cluster. It contains configuration settings of Hadoop core such as I/O settings that are common to HDFS & MapReduce.

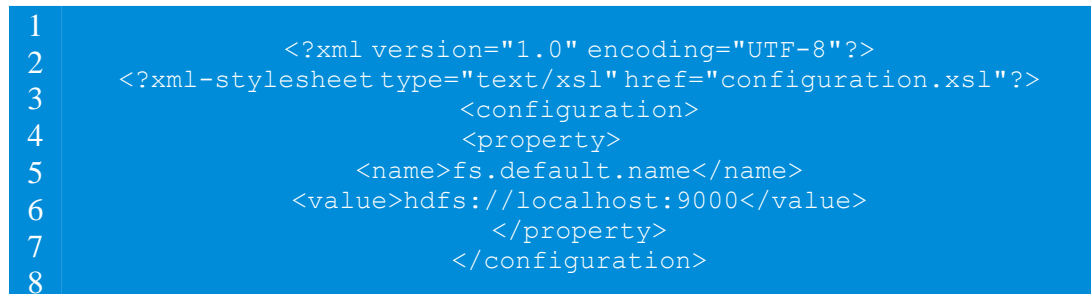
Command: vi core-site.xml



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
[edureka@localhost hadoop]$ vi core-site.xml
```



```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```



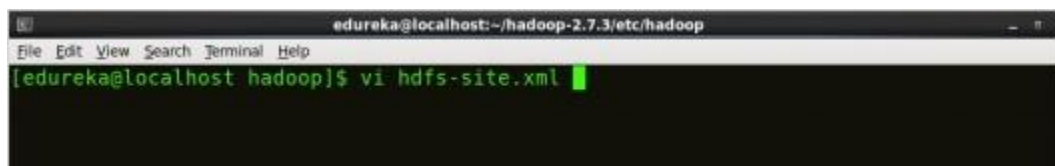
```
1      <?xml version="1.0" encoding="UTF-8"?>
2      <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3          <configuration>
4              <property>
5                  <name>fs.default.name</name>
6                  <value>hdfs://localhost:9000</value>
7              </property>
8          </configuration>
```

Fig: Hadoop Installation – Configuring core-site.xml

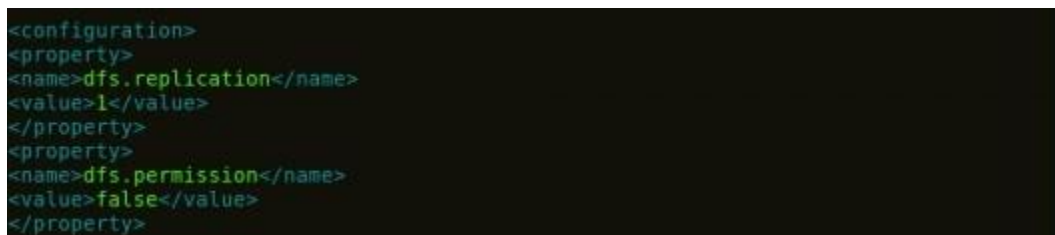
Step 8: Edit *hdfs-site.xml* and edit the property mentioned below inside configuration tag:

hdfs-site.xml contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

Command: vi hdfs-site.xml



```
edureka@localhost:~/hadoop-2.7.3/etc/hadoop
[edureka@localhost hadoop]$ vi hdfs-site.xml
```



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

Fig: Hadoop Installation – Configuring hdfs-site.xml

```

1
2      <?xml version="1.0" encoding="UTF-8"?>
3 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
4      <configuration>
5          <property>
6              <name>dfs.replication</name>
7              <value>1</value>
8          </property>
9          <property>
10             <name>dfs.permission</name>
11             <value>>false</value>
12         </property>
13     </configuration>

```

Step 9: Edit the *mapred-site.xml* file and edit the property mentioned below

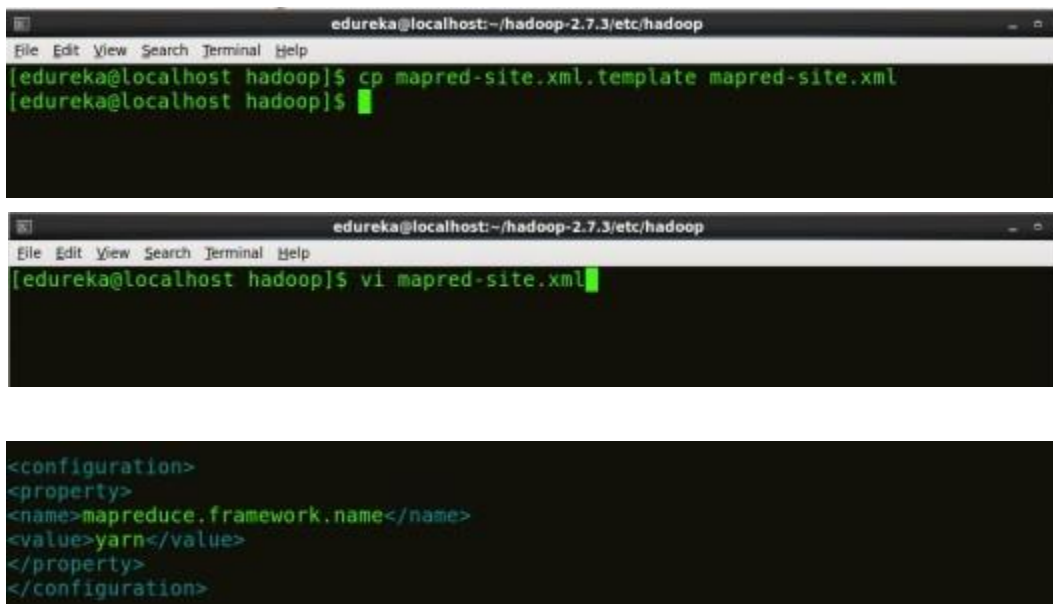
inside configuration tag:

mapred-site.xml contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc.

In some cases, *mapred-site.xml* file is not available. So, we have to create the *mapred-site.xml* file using *mapred-site.xml* template.

Command: `cp mapred-site.xml.template mapred-site.xml`

Command: `vi mapred-site.xml`.



```

edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ cp mapred-site.xml.template mapred-site.xml
[edureka@localhost hadoop]$

edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi mapred-site.xml

<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

```

Fig: Hadoop Installation – Configuring mapred-site.xml

```

1      <?xml version="1.0" encoding="UTF-8"?>
2      <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3          <configuration>
4              <property>
5                  <name>mapreduce.framework.name</name>
6                  <value>yarn</value>
7              </property>
8          </configuration>

```

Step 10: Edit *yarn-site.xml* and edit the property mentioned below inside configuration tag:

yarn-site.xml contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

Command: vi yarn-site.xml

```

edureka@localhost:~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi yarn-site.xml

```

```

<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>

```

Fig: Hadoop Installation – Configuring yarn-site.xml

```

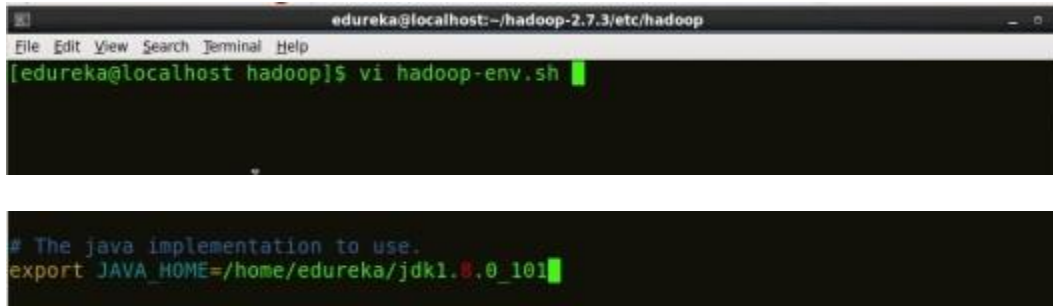
1
2      <?xml version="1.0">
3      <configuration>
4          <property>
5              <name>yarn.nodemanager.aux-services</name>
6              <value>mapreduce_shuffle</value>
7          </property>
8          <property>
9              <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</
10             name>
11             <value>org.apache.hadoop.mapred.ShuffleHandler</value>
12         </property>
13     </configuration>

```

Step 11: Edit *hadoop-env.sh* and add the Java Path as mentioned below:

hadoop-env.sh contains the environment variables that are used in the script to run Hadoop like Java home path, etc.

Command: vi *hadoop-env.sh*



```
edureka@localhost: ~/hadoop-2.7.3/etc/hadoop
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ vi hadoop-env.sh

# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

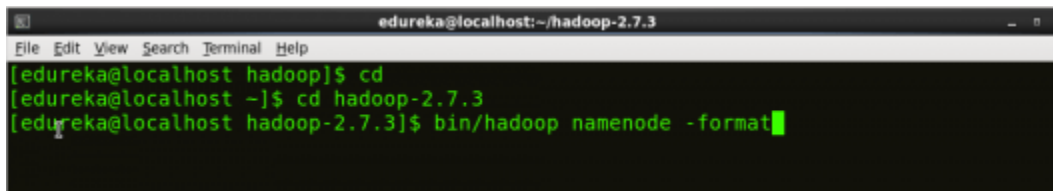
*Fig: Hadoop Installation – Configuring *hadoop-env.sh**

Step 12: Go to Hadoop home directory and format the NameNode.

Command: cd

Command: cd *hadoop-2.7.3*

Command: bin/*hadoop namenode -format*



```
edureka@localhost: ~/hadoop-2.7.3
File Edit View Search Terminal Help
[edureka@localhost hadoop]$ cd
[edureka@localhost ~]$ cd hadoop-2.7.3
[edureka@localhost hadoop-2.7.3]$ bin/hadoop namenode -format
```

Fig: Hadoop Installation – Formatting NameNode

This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the *dfs.name.dir* variable.

Never format, up and running Hadoop filesystem. You will lose all your data stored in the HDFS.

Step 13: Once the NameNode is formatted, go to *hadoop-2.7.3/sbin* directory and start all the daemons.

Command: `cd hadoop-2.7.3/sbin`

Either you can start all daemons with a single command or do it individually.

Command: `./start-all.sh`

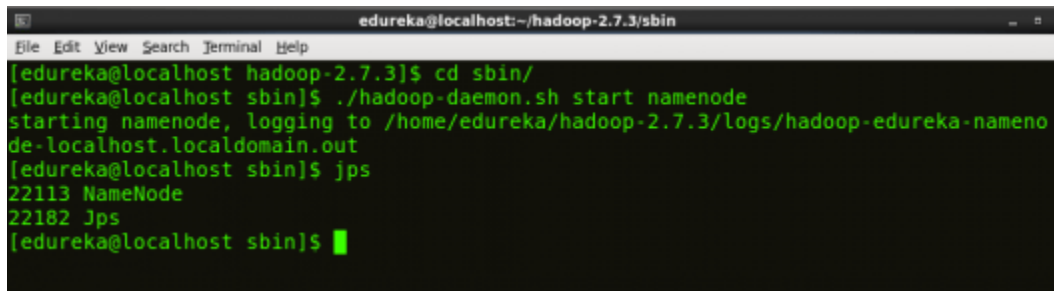
The above command is a combination of *start-dfs.sh*, *start-yarn.sh* & *mr-jobhistory-daemon.sh*

Or you can run all the services individually as below:

Start NameNode:

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the file stored across the cluster.

Command: `./hadoop-daemon.sh start namenode`

A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

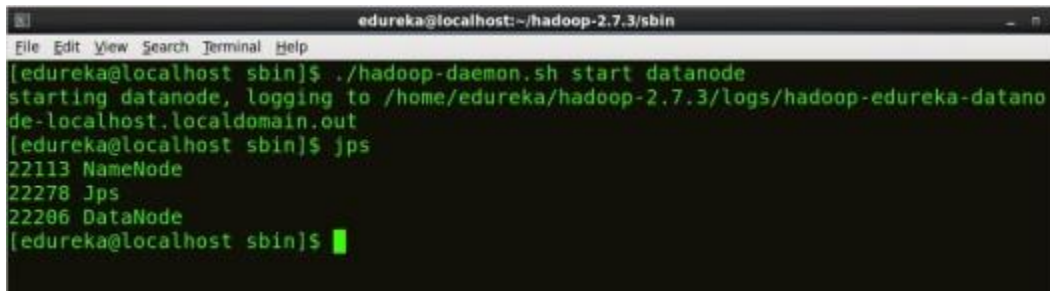
```
[edureka@localhost hadoop-2.7.3]$ cd sbin/
[edureka@localhost sbin]$ ./hadoop-daemon.sh start namenode
starting namenode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-nameno
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22182 Jps
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting NameNode

Start DataNode:

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for different operations.

Command: `./hadoop-daemon.sh start datanode`

A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' shows the execution of the command `./hadoop-daemon.sh start datanode`. The output indicates that the datanode is starting and logging to `/home/edureka/hadoop-2.7.3/logs/hadoop-edureka-datano`. Following this, the `jps` command is run, showing the following processes: `22113 NameNode`, `22278 Jps`, and `22206 DataNode`. The prompt returns to `[edureka@localhost sbin]$` with a green cursor.

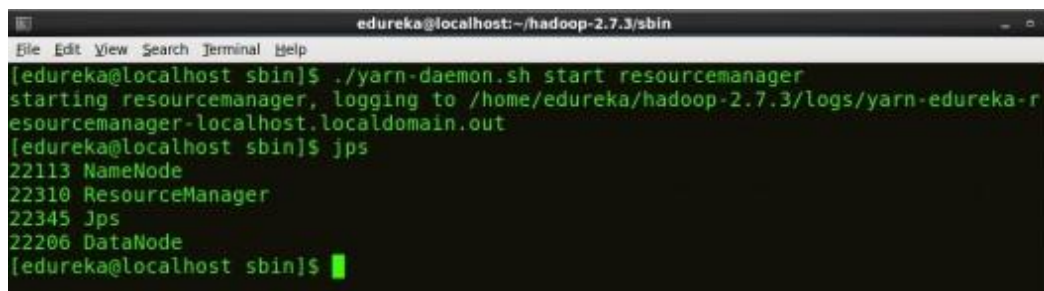
```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./hadoop-daemon.sh start datanode
starting datanode, logging to /home/edureka/hadoop-2.7.3/logs/hadoop-edureka-datano
de-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22278 Jps
22206 DataNode
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting DataNode

Start ResourceManager:

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and the each application's ApplicationMaster.

Command: `./yarn-daemon.sh start resourcemanager`

A terminal window titled 'edureka@localhost:~/hadoop-2.7.3/sbin' shows the execution of the command `./yarn-daemon.sh start resourcemanager`. The output indicates that the resourcemanager is starting and logging to `/home/edureka/hadoop-2.7.3/logs/yarn-edureka-r`. Following this, the `jps` command is run, showing the following processes: `22113 NameNode`, `22310 ResourceManager`, `22345 Jps`, and `22206 DataNode`. The prompt returns to `[edureka@localhost sbin]$` with a green cursor.

```
edureka@localhost:~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./yarn-daemon.sh start resourcemanager
starting resourcemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-r
esourcemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22345 Jps
22206 DataNode
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting ResourceManager

Start NodeManager:

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

Command: `./yarn-daemon.sh start nodemanager`



```
edureka@localhost: ~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./yarn-daemon.sh start nodemanager
starting nodemanager, logging to /home/edureka/hadoop-2.7.3/logs/yarn-edureka-nodemanager-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22592 Jps
22113 NameNode
22310 ResourceManager
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Starting NodeManager

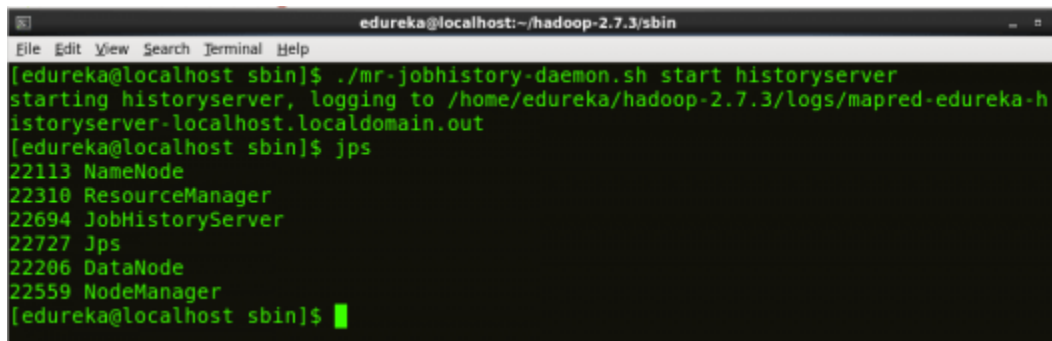
Start JobHistoryServer:

JobHistoryServer is responsible for servicing all job history related requests from client.

Command: `./mr-jobhistory-daemon.sh start historyserver`

Step 14: To check that all the Hadoop services are up and running, run the below command.

Command: `jpsn`



```
edureka@localhost: ~/hadoop-2.7.3/sbin
File Edit View Search Terminal Help
[edureka@localhost sbin]$ ./mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/edureka/hadoop-2.7.3/logs/mapred-edureka-historyserver-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22694 JobHistoryServer
22727 Jps
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

Fig: Hadoop Installation – Checking Daemons

Step 15: Now open the Mozilla browser and goto **localhost:50070/dfshealth.html** to check the NameNode interface.

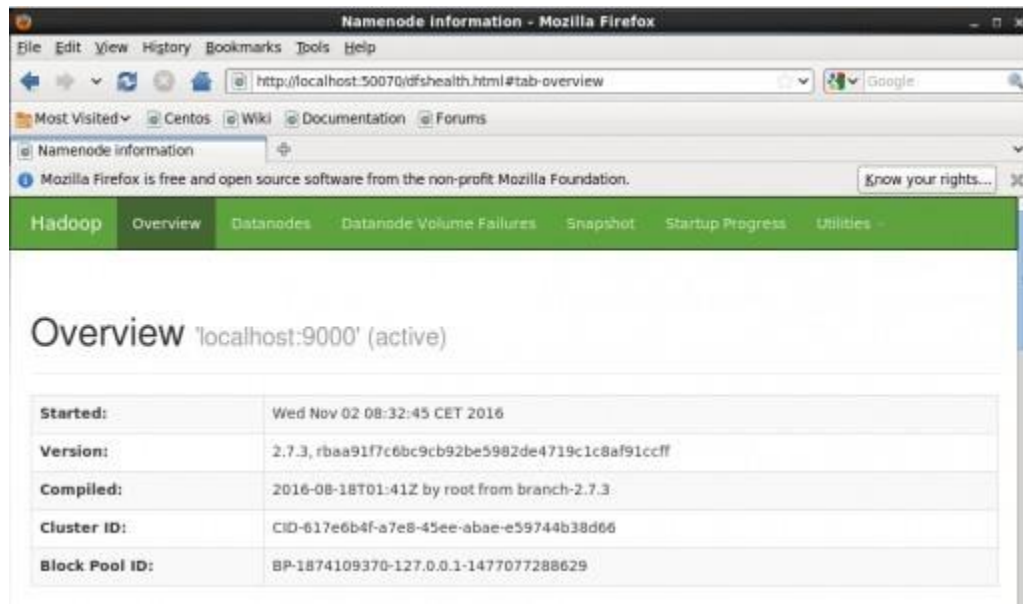


Fig: Hadoop Installation – Starting WebUI

Congratulations, you have successfully installed a single node Hadoop cluster

PROCEDURE FOR WORD COUNT:

su – hadoop

\$ vim WordCount.java

/home/hduser/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); }  
  
}
```

Start Hadoop Cluster

```
$ cd /home/hadoop/hadoop/sbin/
```

```
$ start-dfs.sh
```

```
$ start-yarn.sh
```

```
$ hadoop fs -mkdir /cloud
```

```
$ hadoop fs -ls
```

COPYING A FILE INTO HDFS INPUT DIRECTORY

ROOT USER (open in new terminal)

```
# su - hadoop
```

```
$ vim 5.txt
```

any message type here

```
:wq
```

Hadoop user(old terminal)

```
$ hadoop fs -put /home/hadoop/5.txt /cloud
```

```
$ hadoop fs -ls /cloud
```

```
$ hadoop fs -cat /cloud/5.txt
```

Download the jar file

```
$ cd /home/hadoop
```

```
$ pwd
```

/home/hadoop

\$ wget ftp://115.248.6.21/hadoop-core-1.2.1.jar

\$ ls (Check the file)

hadoop-core-1.2.1.jar

Create a Directory to collect class files

\$ mkdir /home/hadoop/wc

Compiling the java file - WordCount.java

\$ javac -classpath /home/hadoop/hadoop-core-1.2.1.jar -d /home/hadoop/wc
/home/hadoop/WordCount.java

\$ ls /home/hadoop/wc

WordCount.class WordCount\$IntSumReducer.class WordCount\$TokenizerMapper.class

Creating jar file for WordCount.java

\$ jar -cvf /home/hadoop/wc.jar -C /home/hadoop/wc/ .

added manifest

adding: WordCount.class(in = 1497) (out= 808)(deflated 46%)

adding: WordCount\$TokenizerMapper.class(in = 1732) (out= 750)(deflated 56%)

adding: WordCount\$IntSumReducer.class(in = 1731) (out= 734)(deflated 57%)

Executing jar file for WordCount.java

\$ **hadoop jar /home/hadoop/wc.jar WordCount /cloud/5.txt /cse/rad**

Map-Reduce Framework

Map input records=9

Map output records=146

Map output bytes=1621

Map output materialized bytes=1465

Input split bytes=98

Combine input records=146

Combine output records=106

Reduce input groups=106

Reduce shuffle bytes=1465

Reduce input records=106

Reduce output records=106

Spilled Records=212

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=29

CPU time spent (ms)=1130

Physical memory (bytes) snapshot=397852672

Virtual memory (bytes) snapshot=1907441664

Total committed heap usage (bytes)=245235712

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=1042

File Output Format Counters

Bytes Written=1035

\$ hadoop fs -ls /cse/rad

16/08/25 15:34:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Found 2 items

-rw-r--r-- 1 hadoop supergroup 0 2016-08-25 15:33 /cse/rad/_SUCCESS

-rw-r--r-- 1 hadoop supergroup 1035 2016-08-25 15:33 /cse/rad/part-r-00000

\$ hadoop fs -cat /cse/rad/*

16/08/25 15:34:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

(CentOS 1

100% 1

American 1

CentOS 5

Chat, 1

Community 1

RESULT:

Thus the Hadoop single node cluster was successfully installed and a simple application like wordcount was executed successfully.