S

Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH

Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

CLOUD COMPUTING LAB
(R22CSE4128)

LAB MANUAL

IV Year I Semester

DEPARTMENT OF INFORMATION TECHNOLOGY

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
B. TECH –INFORMATION TECHNOLOGY

INSTITUTION VISION

To be a premier Institution in Engineering & Technology and Management with competency, values and social consciousness.

INSTITUTION MISSION

IM1 Provide high quality academic programs, training activities and research facilities.

IM2 Promote Continuous Industry-Institute Interaction for Employability, Entrepreneurship, Leadership and Research aptitude among stakeholders.

IM3 Contribute to the Economical and technological development of the region, state and nation.

DEPARTMENT VISION

To be a recognized knowledge centre in the field of Information Technology with self - motivated, employable engineers to society.

DEPARTMENT MISSION

The Department has following Missions:

DM1 To offer high quality student centric education in Information Technology.

DM2 To provide a conducive environment towards innovation and skills.

DM3 To involve in activities that provide social and professional solutions.

DM4 To impart training on emerging technologies namely cloud computing and IOT with involvement of stake holders.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: **Higher Studies:** Graduates with an ability to apply knowledge of Basic sciences and programming skills in their career and higher education.

PEO2: **Lifelong Learning:** Graduates with an ability to adopt new technologies for ever changing IT industry needs through Self-Study, Critical thinking and Problem solving skills.

PEO3: **Professional skills:** Graduates will be ready to work in projects related to complex problems involving multi-disciplinary projects with effective analytical skills.

PEO4: **Engineering Citizenship:** Graduates with an ability to communicate well and exhibitsocial, technical and ethical responsibility in process or product.

PROGRAM OUTCOMES (POs) & PROGRAM SPECIFIC OUTCOMES (PSOs)

PO	Description
PO 1	**Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO 2	**Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO 3	**Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO 4	**Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO 5	**Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)
PO 6	**The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO 7	**Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
PO 8	**Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
PO 10	**Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
PO 11	**Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)
Program Specific Outcomes	
PSO 1	**Software Development:** To apply the knowledge of Software Engineering, Data Communication, Web Technology and Operating Systems for building IOT and Cloud Computing applications.
PSO 2	**Industrial Skills Ability:** Design, develop and test software systems for world-wide network of computers toprovide solutions to real world problems.
PSO 3	**Project implementation:** Analyze and recommend the appropriate IT Infrastructure required for the implementationof a project.

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time),those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with thesynopsis / program / experiment details.
3. Student should enter into the laboratory with:
    a) Laboratory observation notes with all the details (Problem statement, Aim, Algorithm,Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b) Laboratory Record updated up to the last session experiments and other utensils (if any)needed in the lab.
    c) Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation notebook, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain thediscipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the labafter completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)

B.Tech. - IV Year – I Semester

L	T	P	C
0	0	2	1

(R22CSE4128) CLOUD COMPUTING LABORATORY

OBJECTIVES:

The student should be made to:

- Be exposed to tool kits for grid and cloud environment.
- Be familiar with developing web services/Applications in grid framework
- Learn to run virtual machines of different configuration.
- Learn to use Hadoop

Course Outcome: On completion of this course, the students will be able to:

- Configure various virtualization tools such as Virtual Box, VMware workstation.
- Design and deploy a web application in a PaaS environment.
- Learn how to simulate a cloud environment to implement new schedulers.
- Install and use a generic cloud environment that can be used as a private cloud.
- Manipulate large data sets in a parallel environment.

LIST OF EXPERIMENTS:

Exercises:

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 hello world 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.

S.NO	NAME OF EXPERIMENTS
1.	Install Virtualbox/VMware Workstation with different flavours of linux or windows OS on top of windows7 or8.
2.	Install a C compiler in the virtual machine created using virtual box and execute Simple Programs
3.	Install Google App Engine. Create hello world app and other simple web applications using python/java
4.	Use GAE launcher to launch the web applications.
5.	Simulate a cloud scenario using Cloud Sim and run a scheduling algorithm that is not presentin Cloud Sim.
6.	Find a procedure to tansfer the files from one virtual machine to another virtual machine.
7.	Find a procedure to launch virtual machine using try stack (Online Open stack DemoVersion)
8.	Install Hadoop single node cluster and run simple applications likewordcount.
ADDITIONAL EXPERIMENTS:	
1.	Creating and Executing your First container using Docker
2.	Run a container from Docker Hub

EXERCISE 1:

Aim:

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

PROCEDURE TO INSTALL

Step 1- Download Link

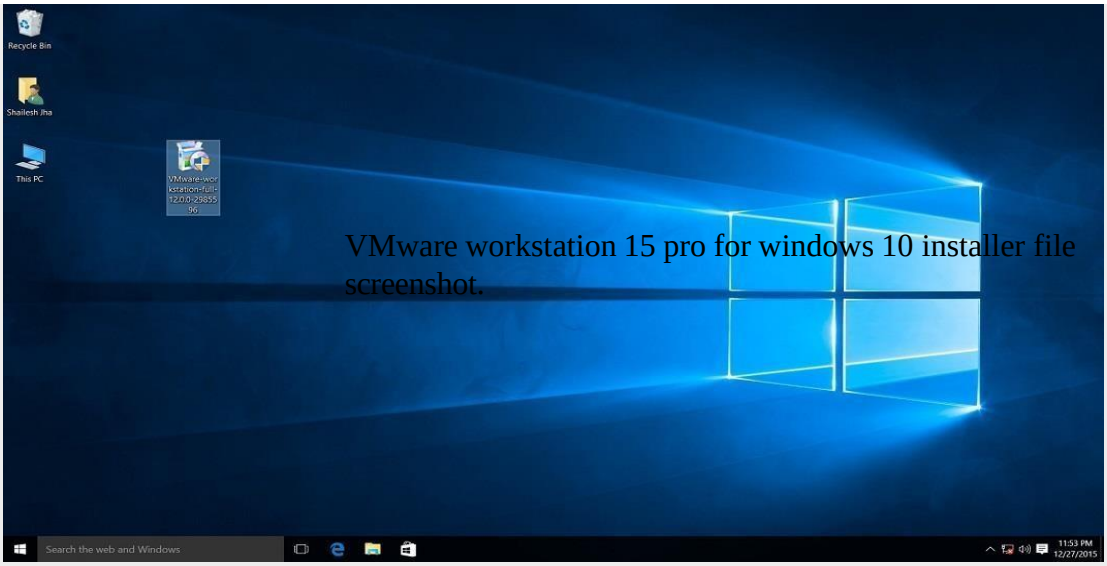
Link for downloading the software is https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html. Download the software for windows. Good thing is that there is no signup process. Click and download begins. Software is around 541MB.

Step 2- Download the installer file

It should probably be in the download folder by default, if you have not changed the settings in your browser. File name should be something like VMware-workstation-full-15.5.1-15018445.exe. This file name can change depending on the version of the software currently available for download. But for now, till the next version is available, they will all be VMware Workstation 15 Pro.

Step 3- Locate the downloaded installer file

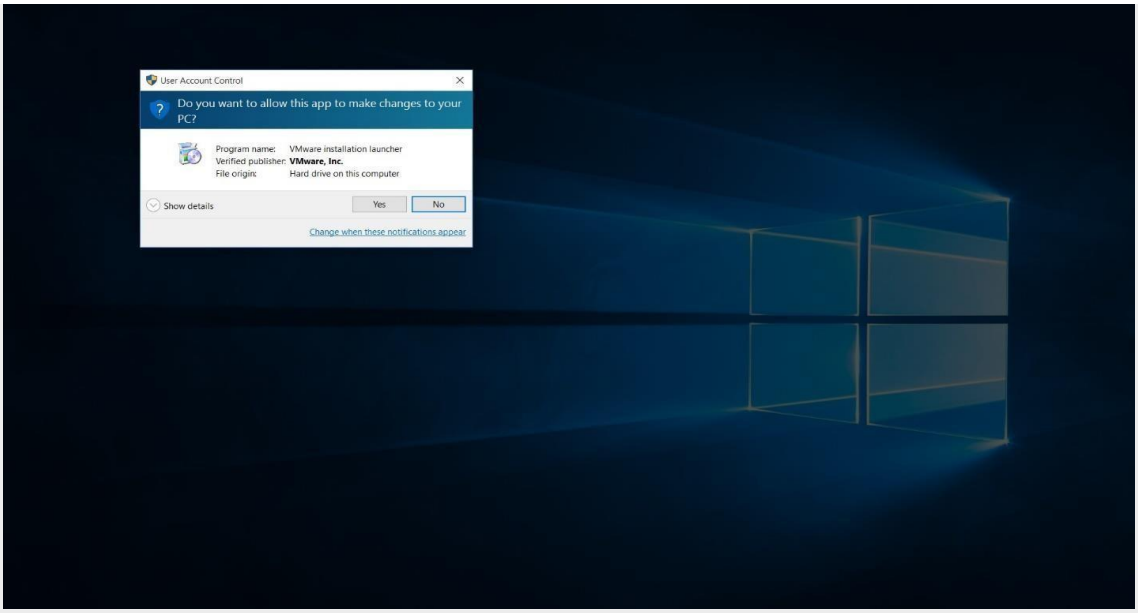
For demonstration purpose, I have placed the downloaded installer on my desktop. Find the installer on your system and double click to launch the application.



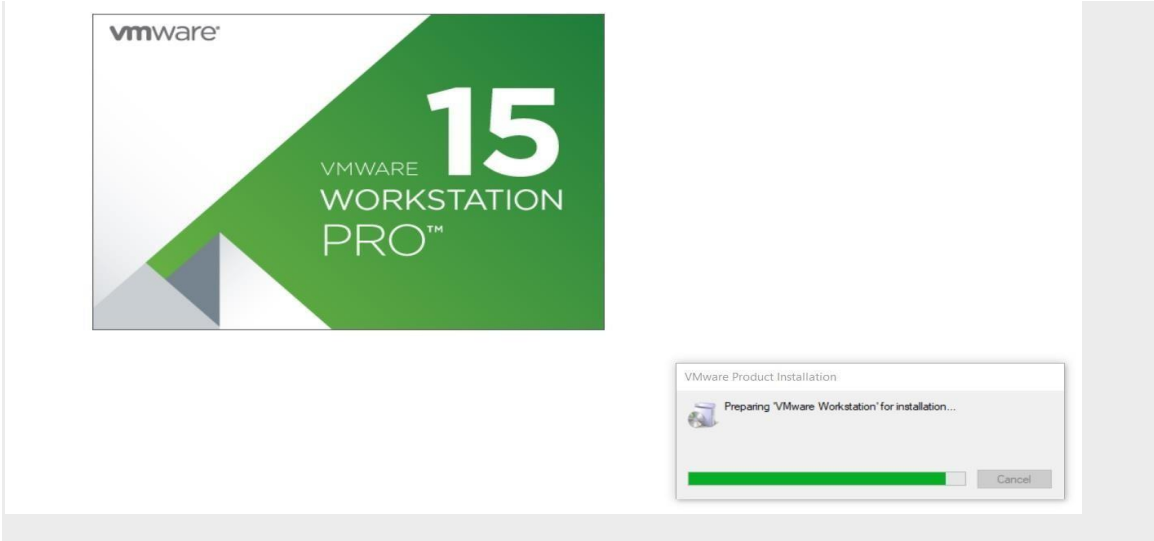
VMware workstation 15 pro for windows 10 installer file screenshot.

Step 4- User Access Control (UAC) Warning

Now you should see User Access Control (UAC) dialog box. Click yes to continue.



Initial Splash screen will appear. Wait for the process to complete.

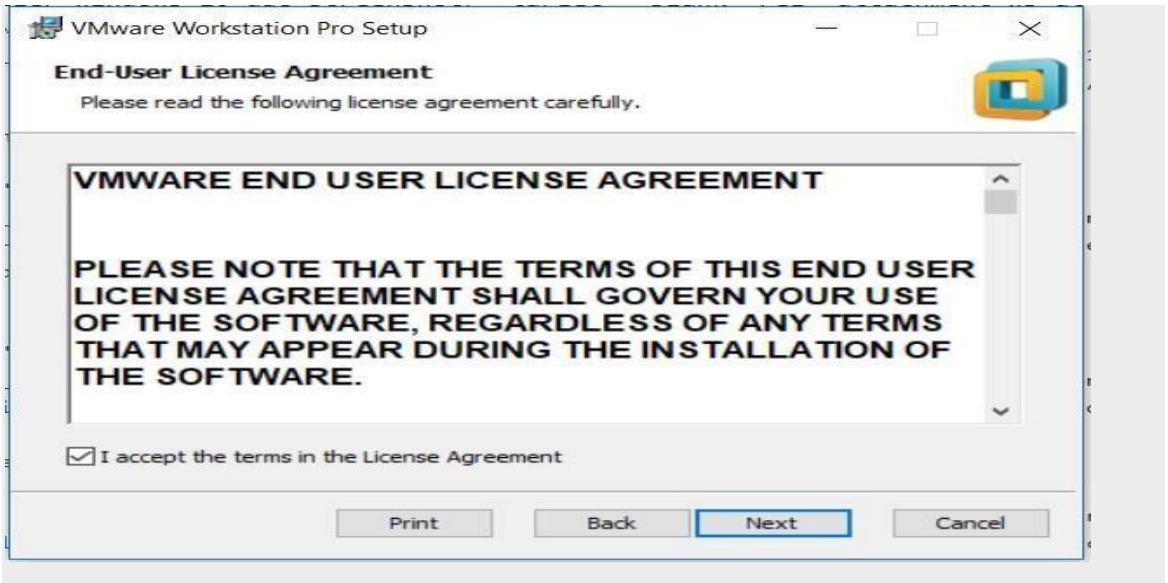


Step 5- VMware Workstation Setup wizard

Now you will see VMware Workstation setup wizard dialog box. Click next to continue.

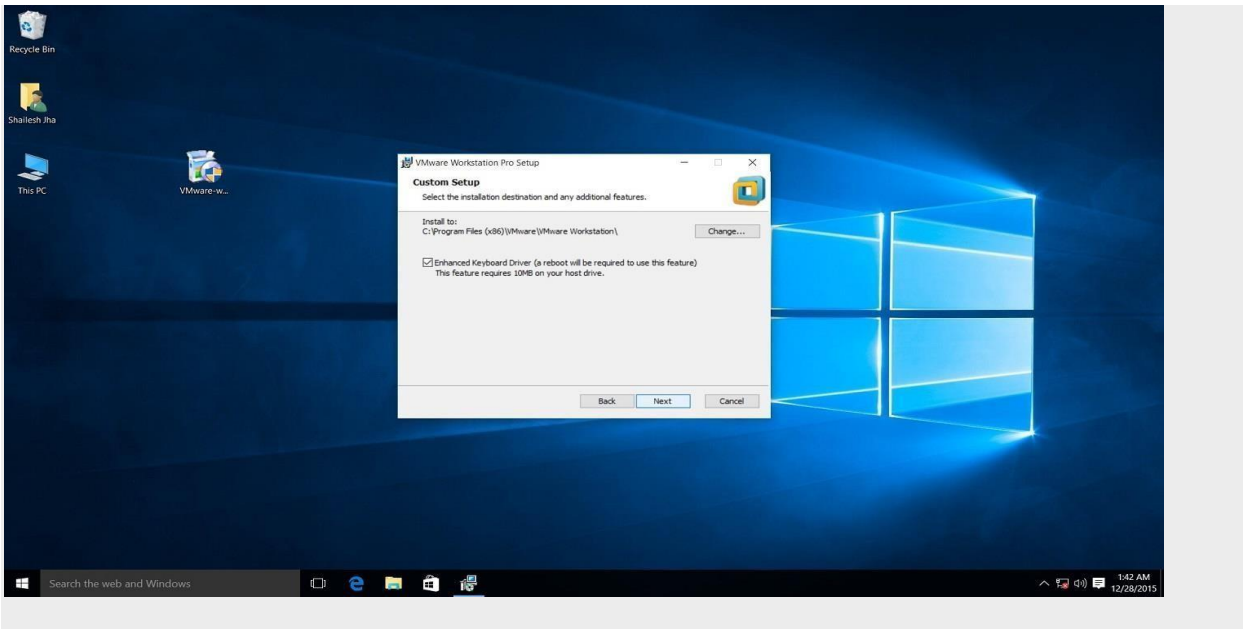
Step 6- End User Licence Agreement

This time you should see End User Licence Agreement dialog box. Check “I accept the terms in the Licence Agreement” box and press next to continue.



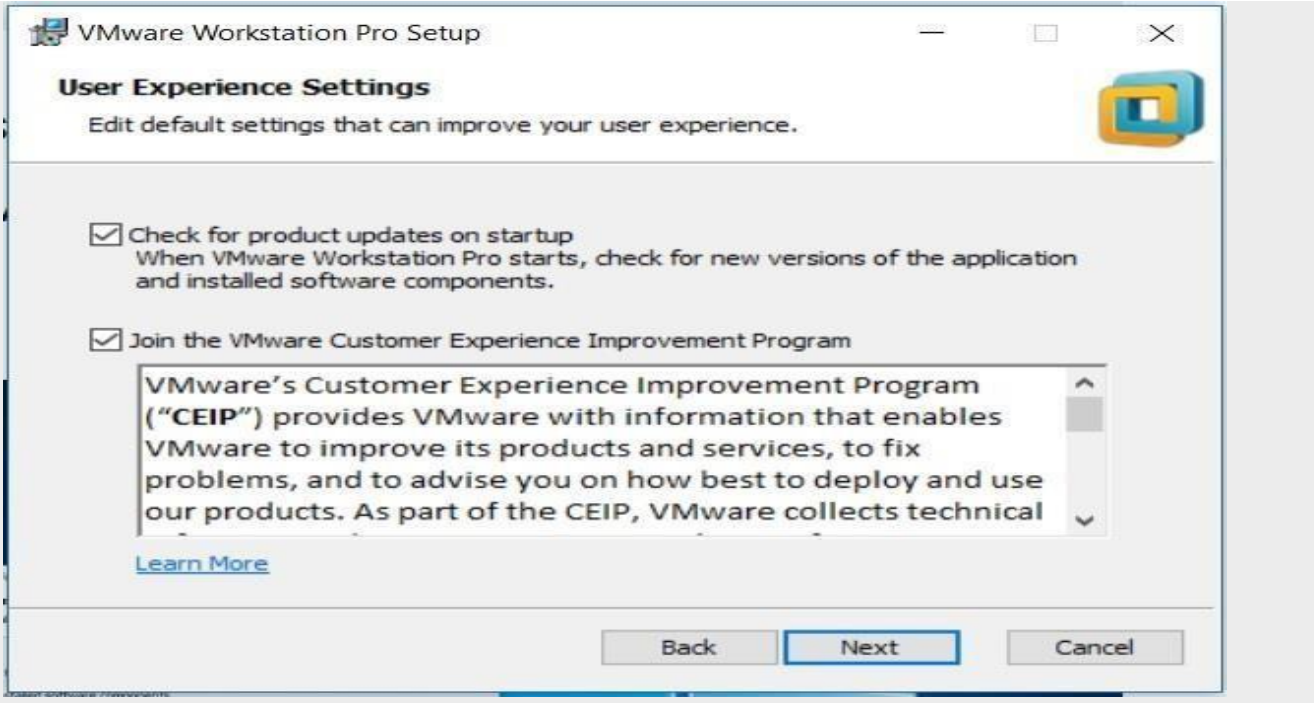
Step 7- Custom Setup options

Select the folder in which you would like to install the application. There is no harm in leaving the defaults as it is. Also select Enhanced Keyboard Driver check box.



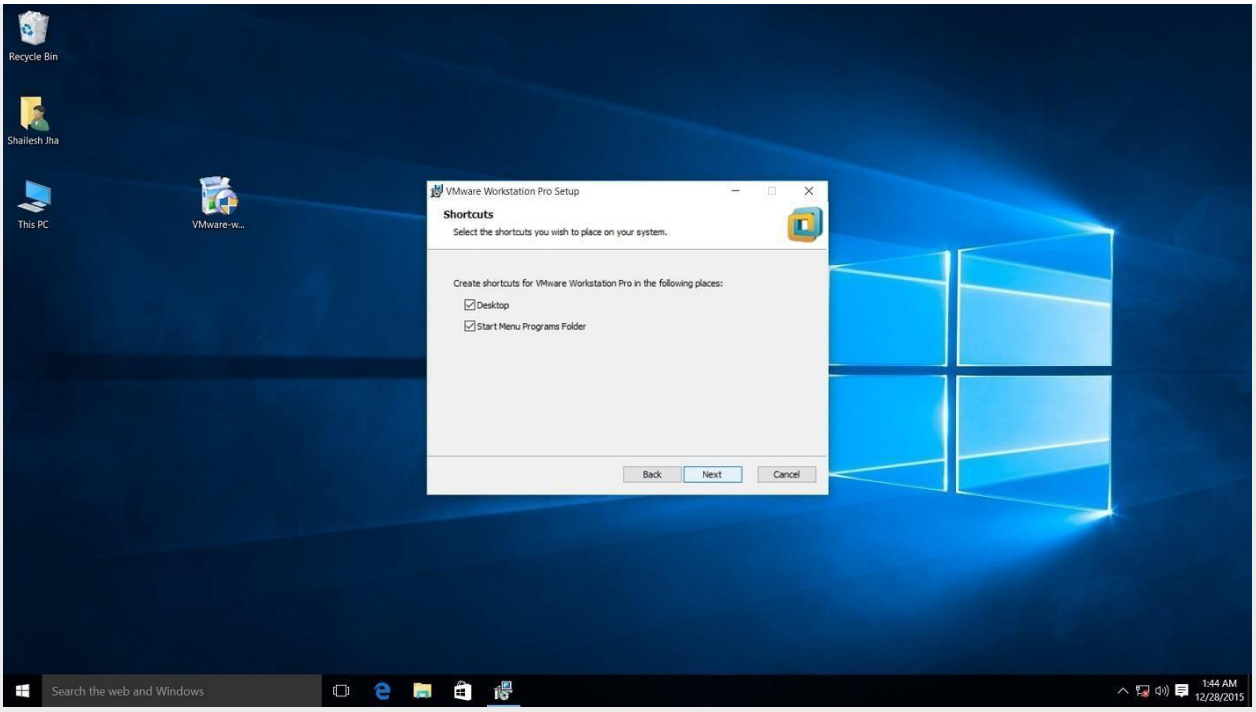
Step 8- User Experience Settings

Next you are asked to select “Check for Updates” and “Help improve VMware Workstation Pro”. Do as you wish. I normally leave it to defaults that is unchecked.



Step 9- Application Shortcuts preference

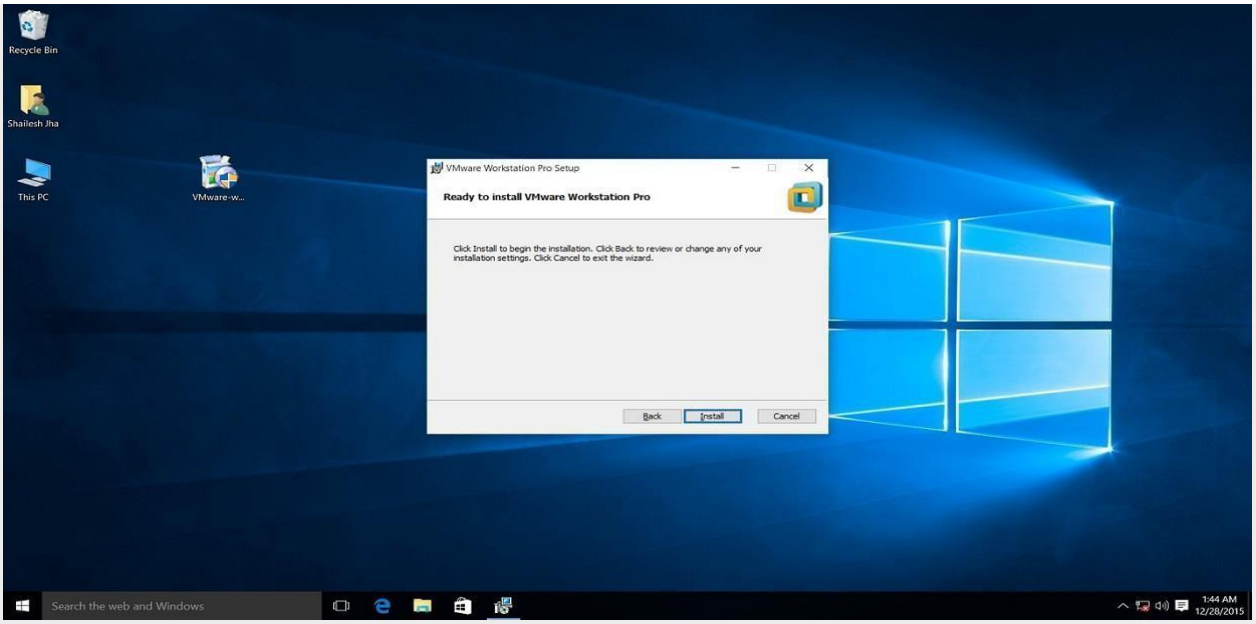
Next step is to select the place you want the shortcut icons to be placed on your system to launch the application. Please select both the options, desktop and start menu and click next.



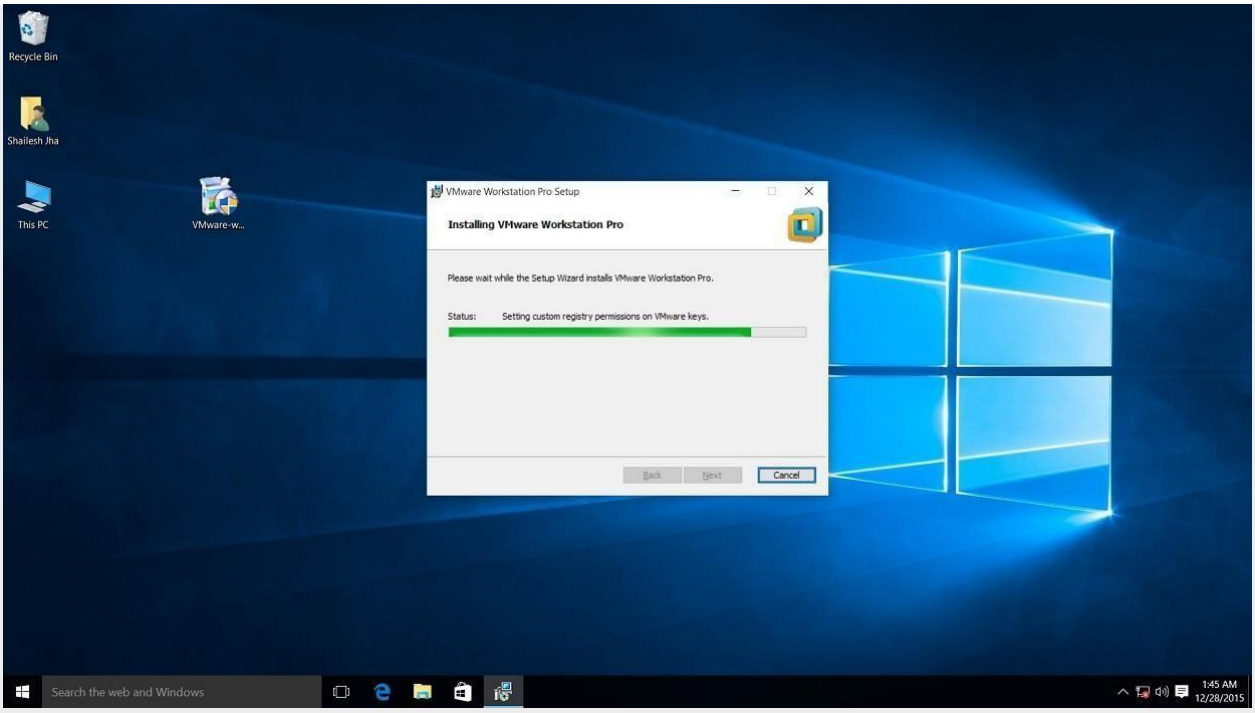
VMware workstation 15 pro installation shortcut selection checkbox screenshot.

Step 10- Installation begins

Now you see the begin installation dialog box. Click install to start the installation process.

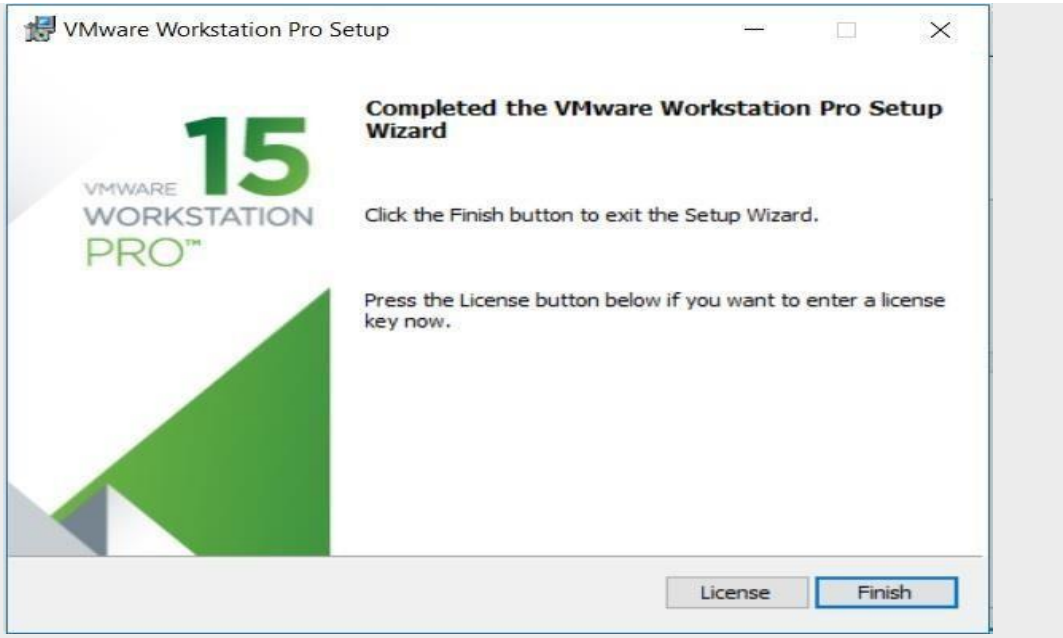


Below screenshot shows Installation in progress. Wait for this to complete.



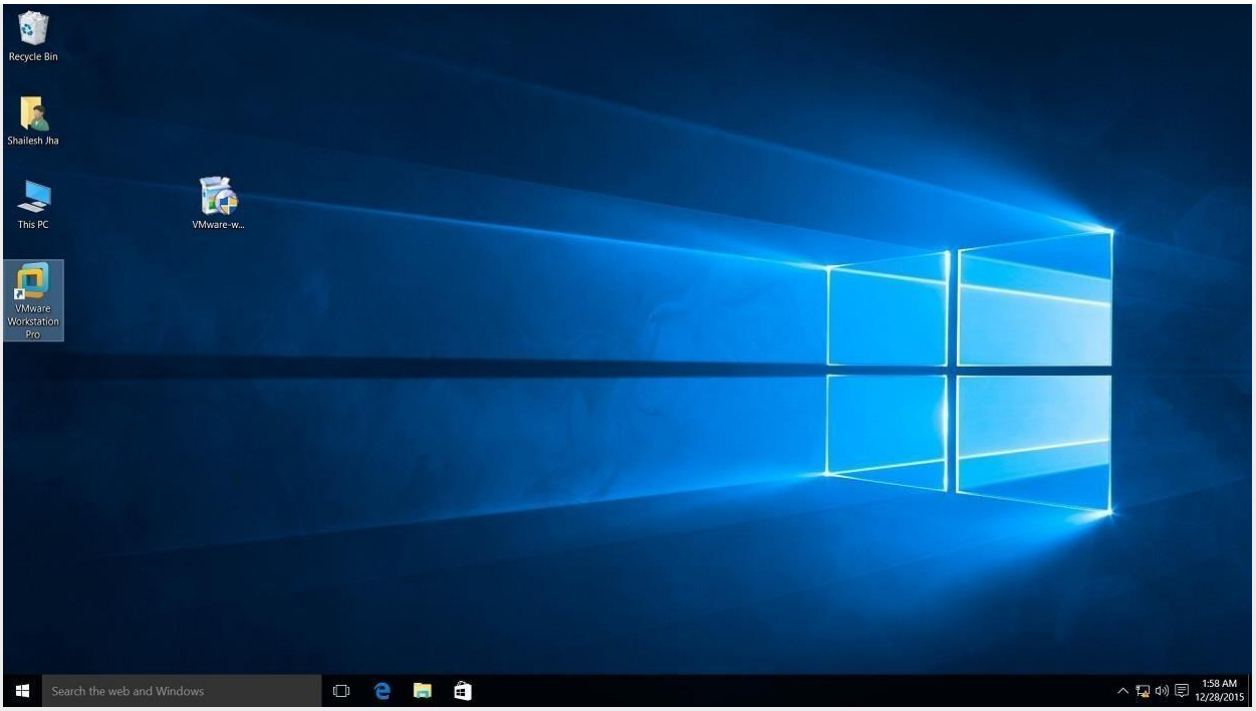
Screenshot for VMware Workstation 15 pro installation process.

At the end you will see installation complete dialog box. Click finish and you are done with the installation process. You may be asked to restart your computer. Click on Yes to restart.



Step 11- Launch VMware Workstation

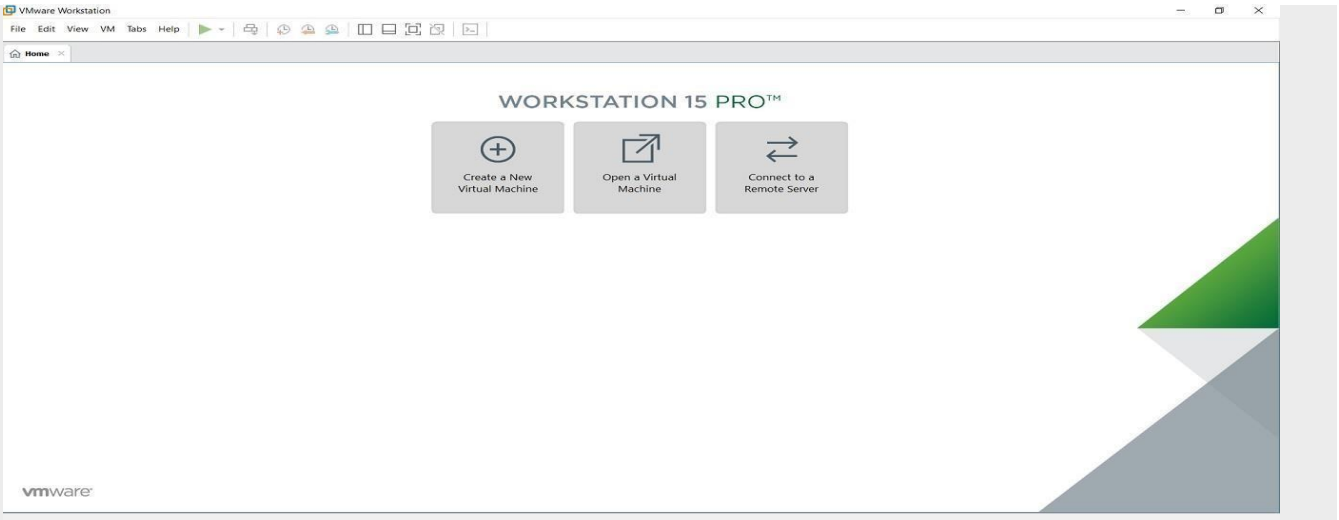
After the installation completes, you should see VMware Workstation icon on the desktop. Double click on it to launch the application.



Screenshot for VMware Workstation 15 Pro icon on windows 10 desktop.

Step 12- Licence Key

If you see the dialog box asking for licence key, click on trial or enter the licence key. Then what you have is the VMware Workstation 15 Pro running on your windows 10 desktop. If don't have the licence key, you will have 30 days trial.



VMware Workstation 15 Pro home screen

Step 13- At some point if you decide to buy

At some point of time if you decide to buy the Licence key, you can enter the Licence key by going to**Help->EnteraLicenceKey** You can enter the 25 character licence key in the dialog box shown below and click OK. Now you have the licence version of the software.

Result:

Thus the VMware workstation is created on windows10.

Viva Questions

1.What is virtualization?

2. how does VirtualBox/VMware Workstation support it?

3. What is the difference between a host OS and a guest OS in virtualization?

4. What are the steps to install a Linux distribution (e.g., Ubuntu) on VirtualBox?

5. What are the different types of network configurations available in VirtualBox or VMware?

EXERCISE 2:

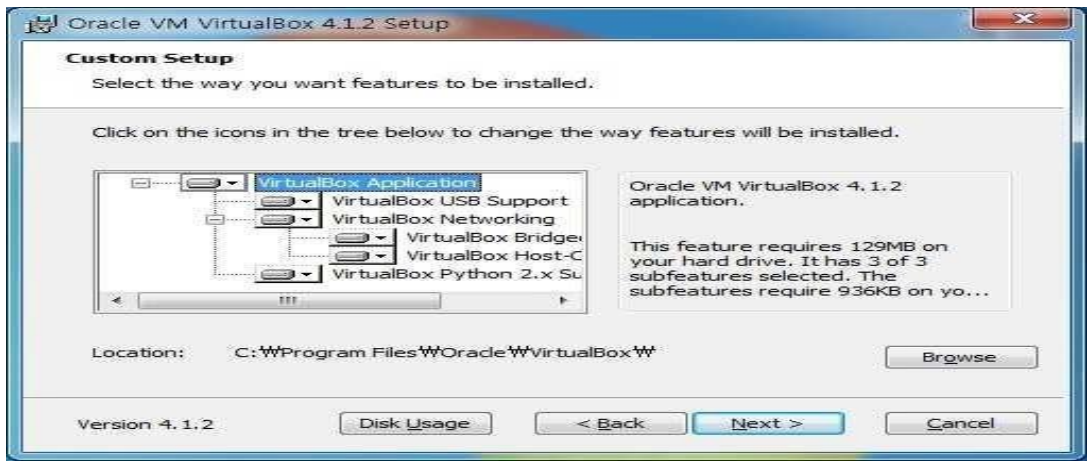
AIM:

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

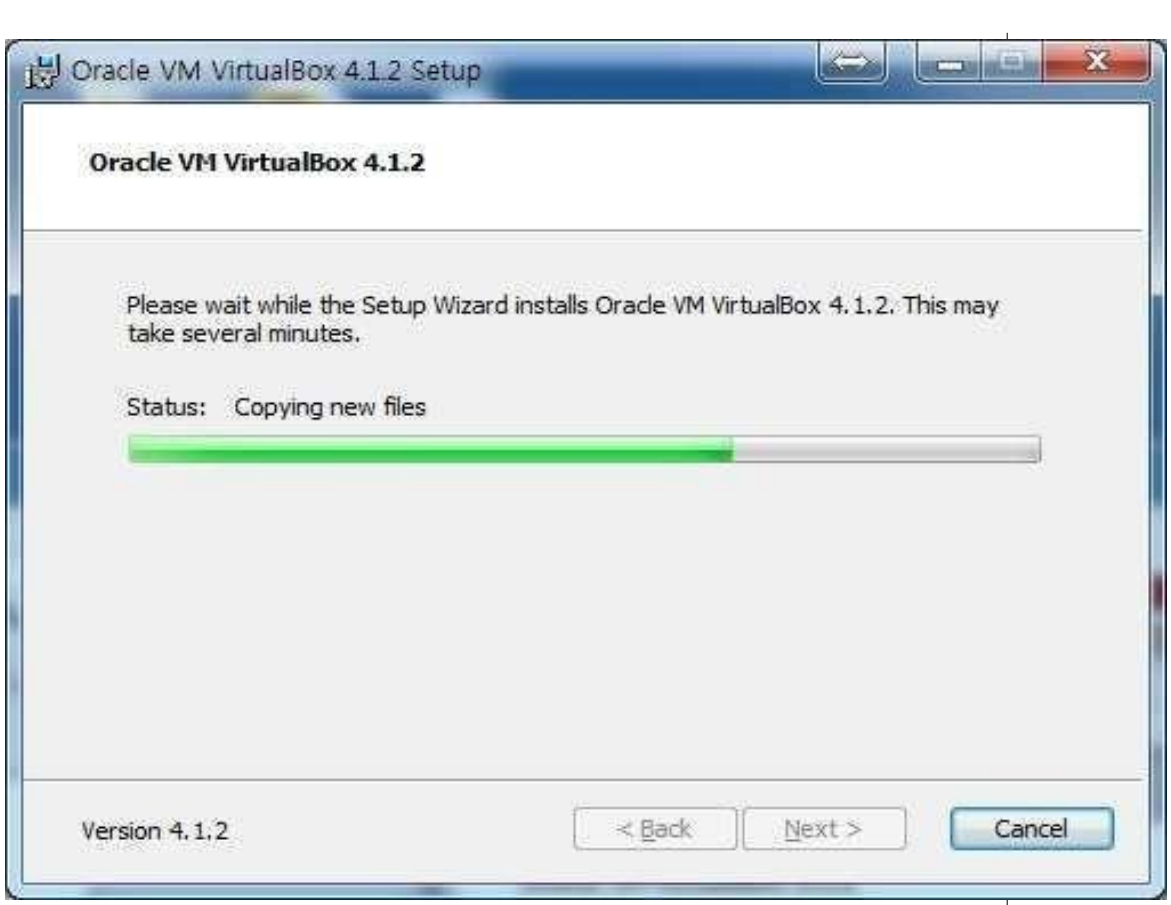
1.Visit http://www.virtualbox.org/wiki/downloads

2.Download VirtualBox platform packages for your OS

3.Open the Installation Package by double clicking



1. Click continue and finish installing Virtual Box



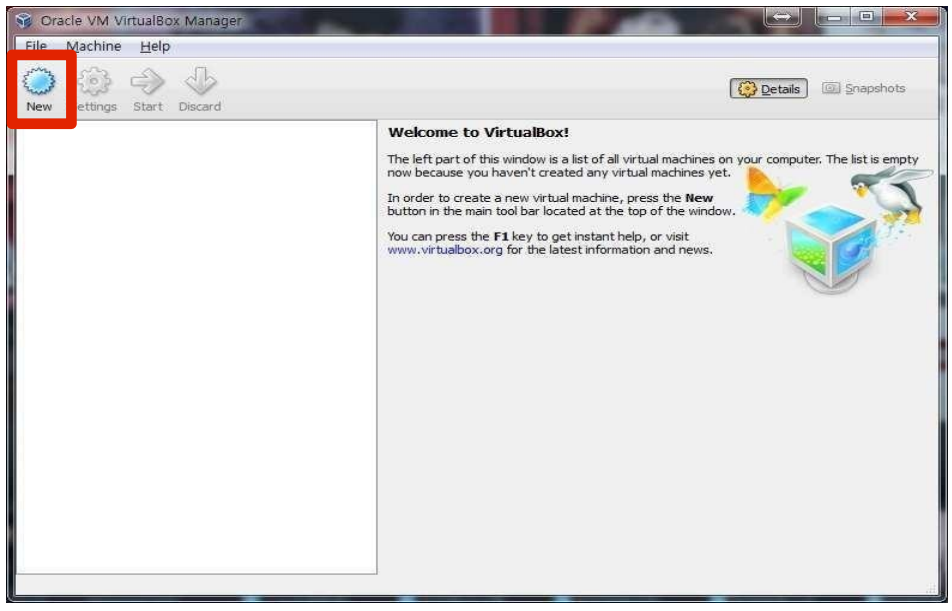
2. When finished installation, close the window.

2. Download Linux

- Visit the pagehttp://www.ubuntu.com/download/ubuntu/download
- Choose the Latest version of Ubuntu and32-bit and click “StartDownload”

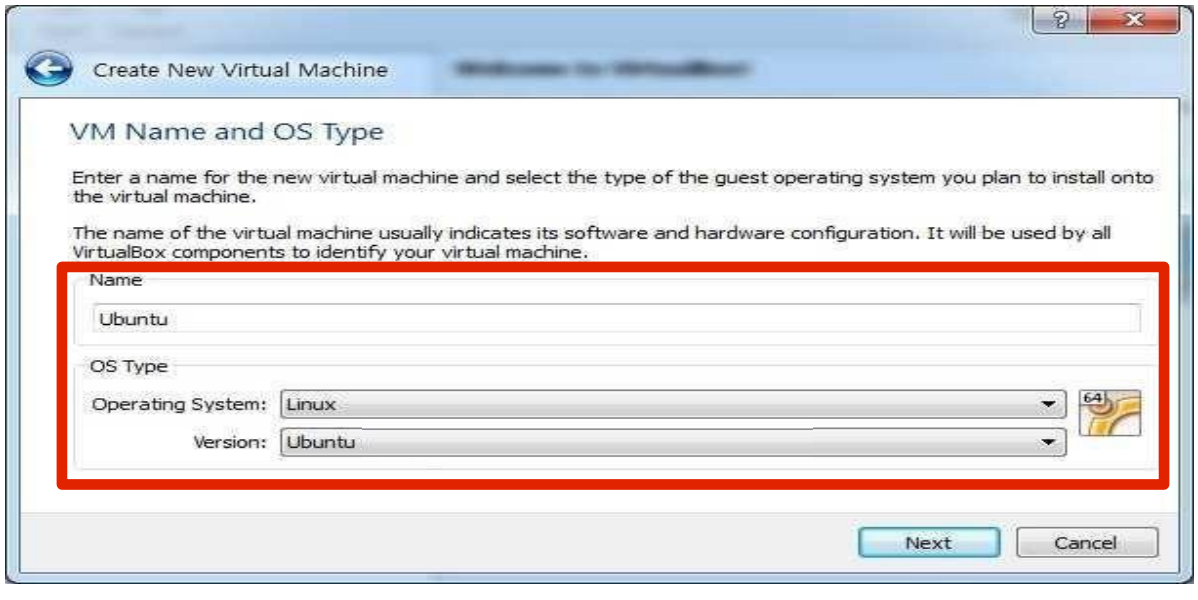
1.Run VirtualBox by double-clicking the icon

2.Click “New” button on the top left corner



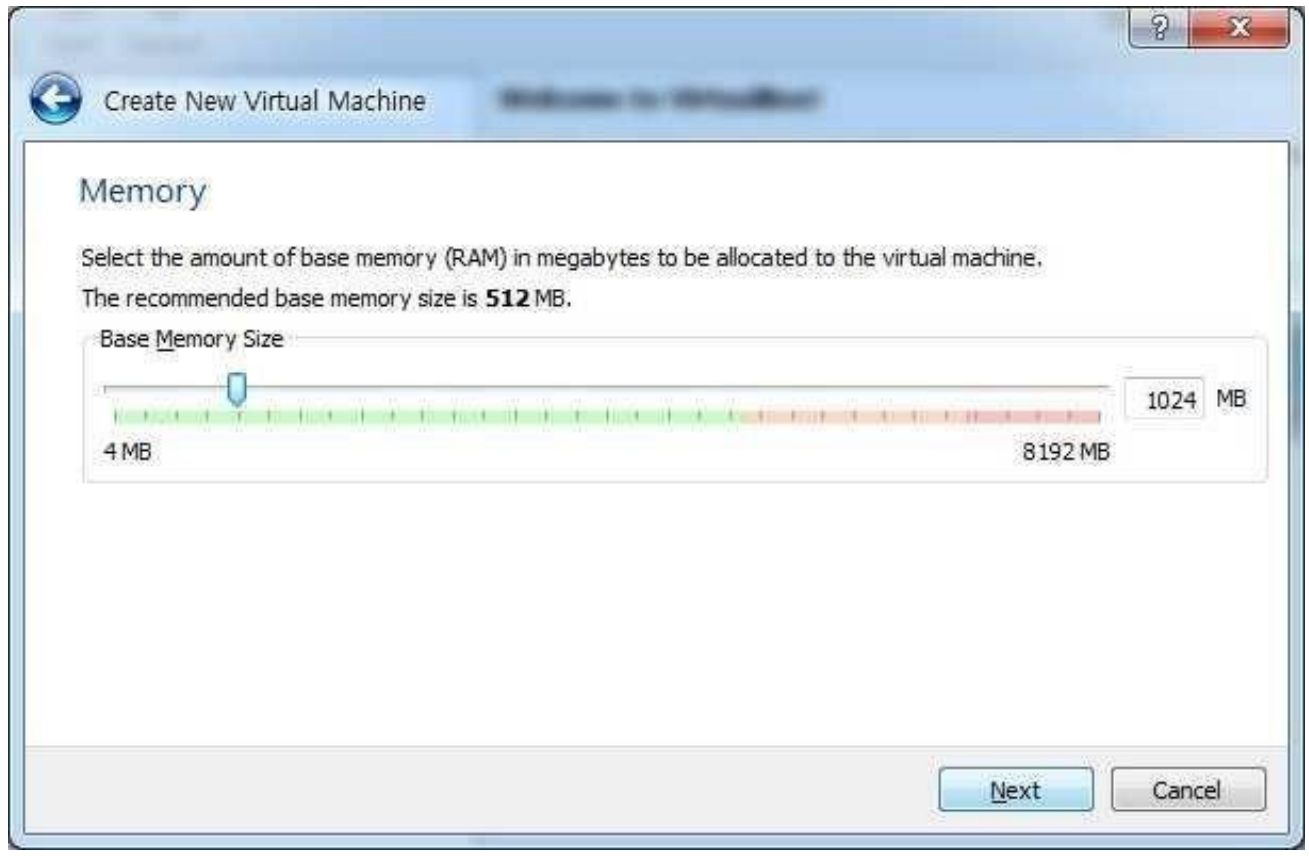
3.Click “Continue” on the pop-up window

4.Type VMname, select “Linux”for the OS and choose “Ubuntu” for the version.



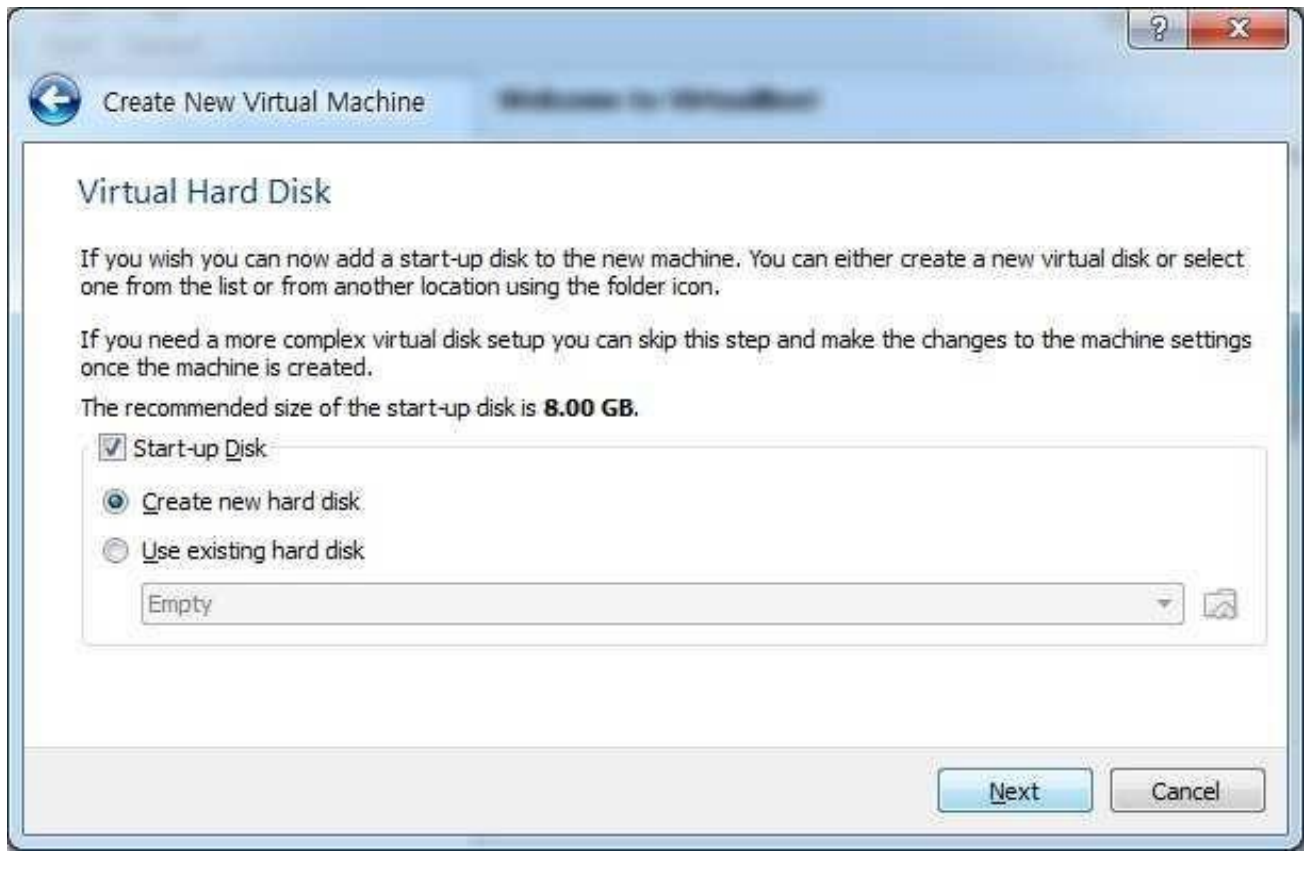
5.Choose the amount of memory to allocate (I suggest choosing between 5I2 MB to I024MB)

6.Click Continue or Next

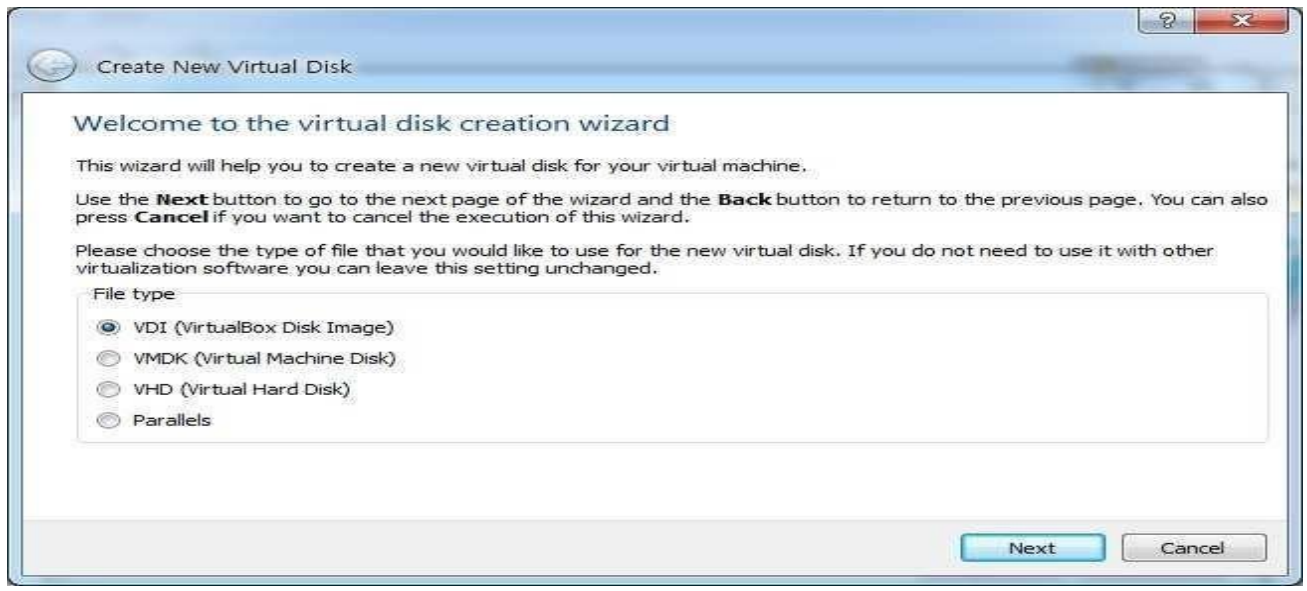


7.Choose create a new virtual hard disk

8.Click Continue or Next



9.Choose VDI (Virtual Box Disk Image) I0. Click Continue or Next

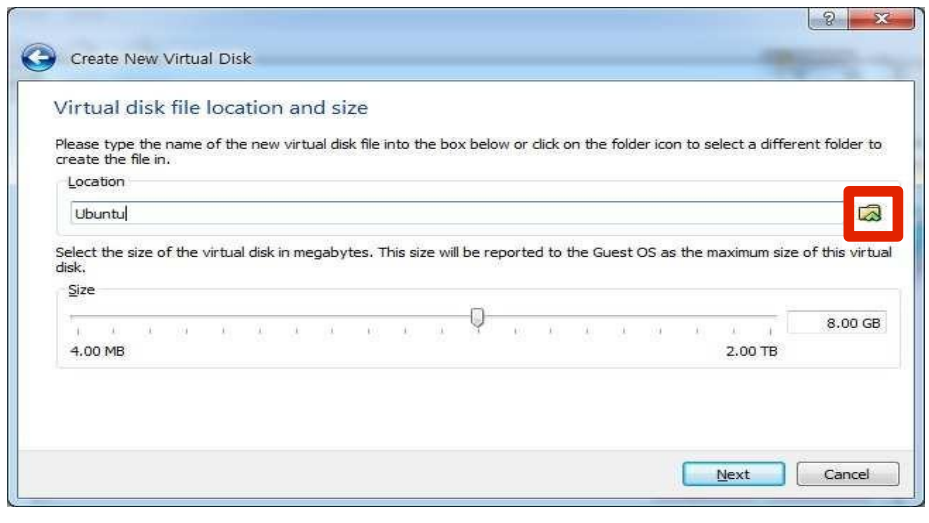


I1.Choose “DynamicallyAllocated” click continue.

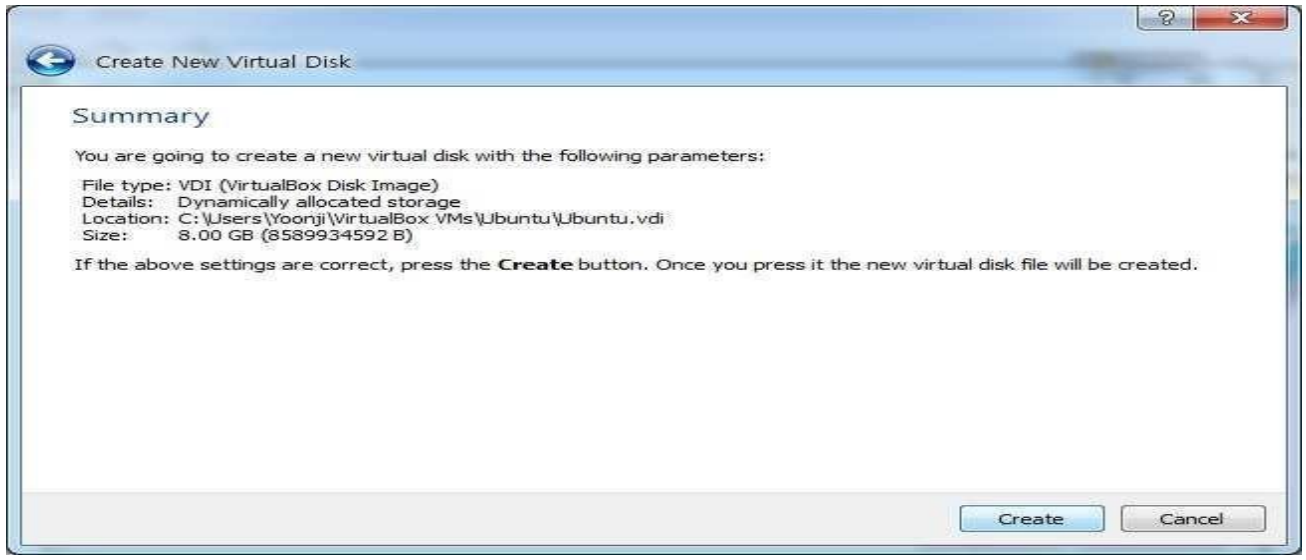
This way,the size of yourVirtual Hard Disk will grow as you use.

I2. Click the folder icon and choose the ubuntu iso file you downloaded.

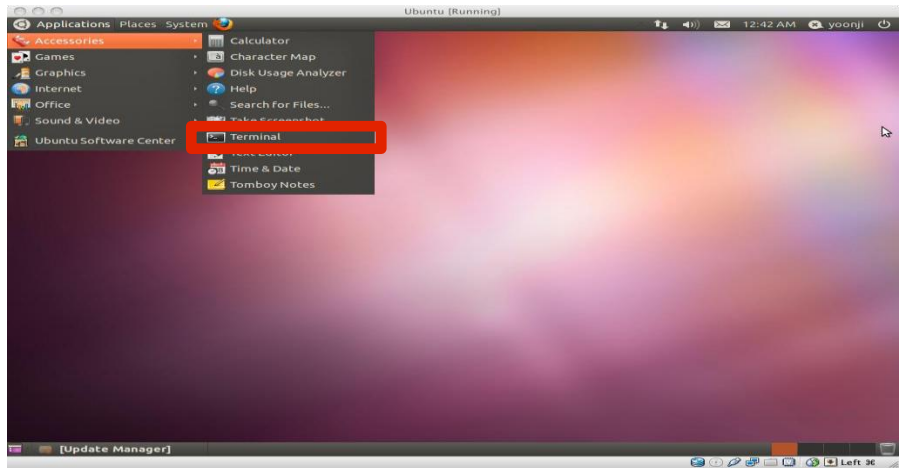
I3. Select the size of the Virtual Disk (I recommend choosing 8 GB) and click continue



I4. Click Create

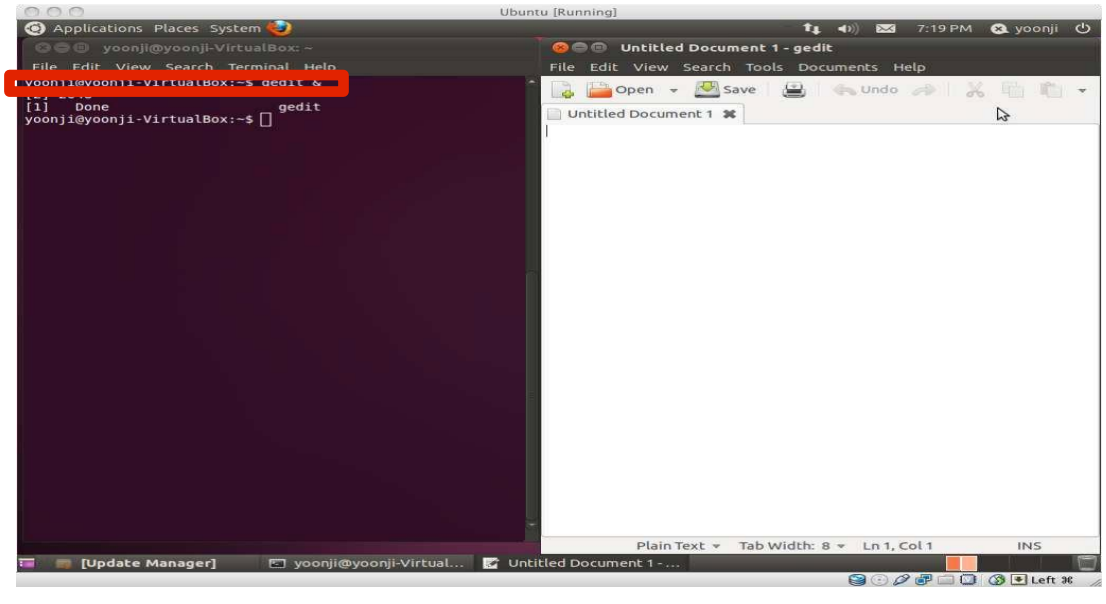


I. Open Terminal(Applications-Accessories-Terminal)



2. Open gedit by typing “gedit &” onterminal

(You can also use any other Text Editor application)

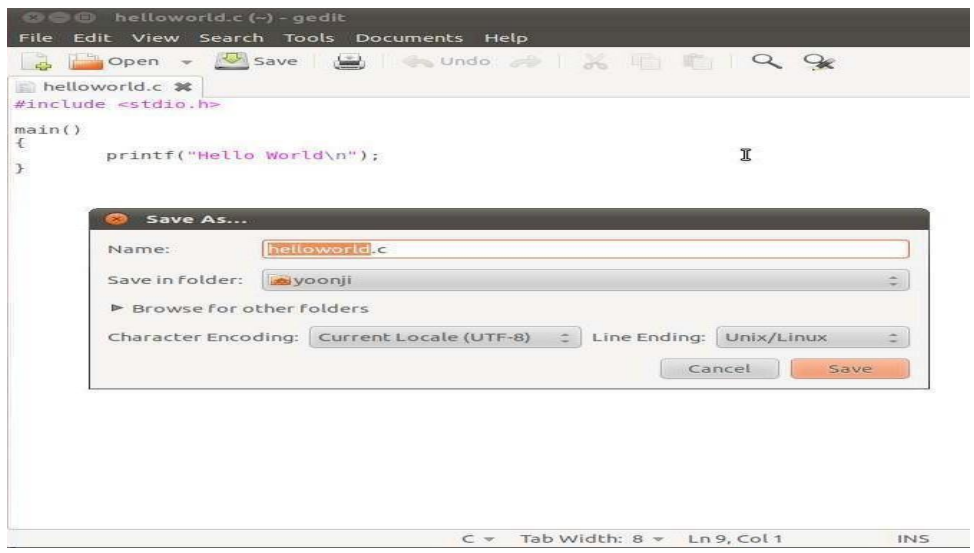


type program as

```
#include<s tdio.h>
 Void main()
 {
```

printf(“hello world”);

}

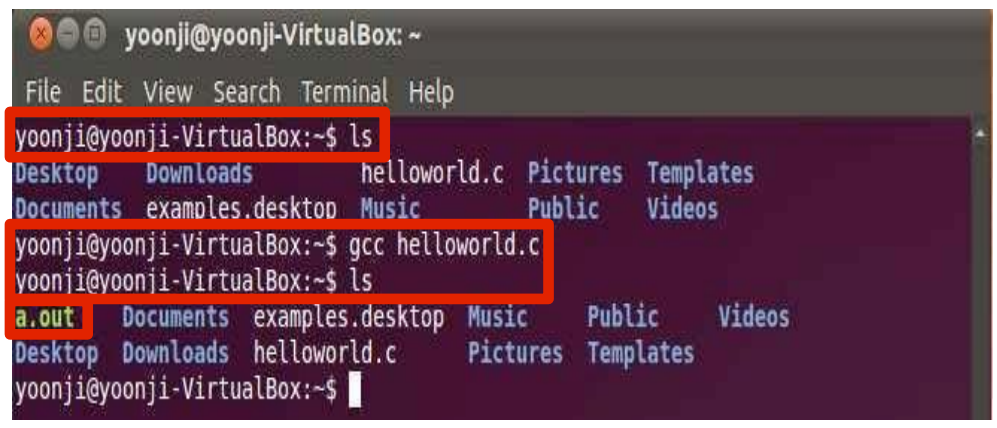


Save it as hello world.c

Type“ls” on Terminal to see all files under current folder

3. Confirm that “hello world.c” is in the current directory. If not,type cd DIRECTORY_PATH to go to the directory that has “hello world.c”

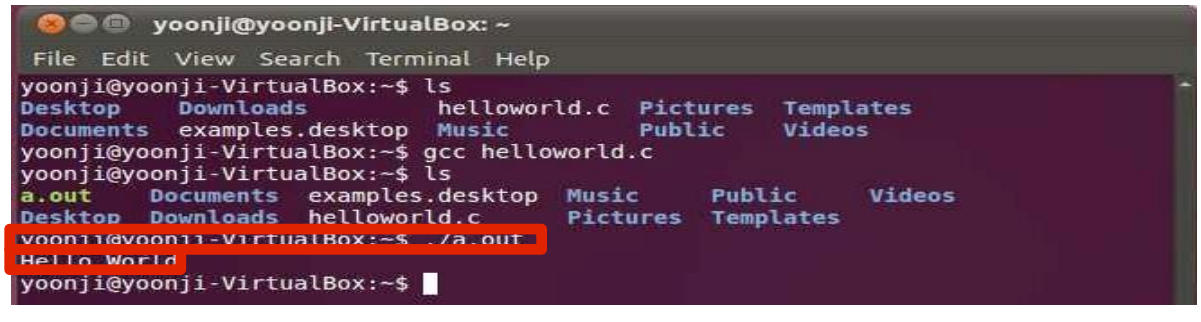
4. Type“gcc helloworld.c” to compile, and type “ls” to confirm that a new executable file “a.out” is created



6.Type“./a.out” on Terminal to run the program

7.If you see “HelloWorld” on the next line, you just successfully ran your first C program!

8.Try other codes from “A Shotgun Introduction to C” on professor Edwards’s web page.You can also find many C programing guides online. (jusgoogleit!)



Result:

Thus the virtual machine is created and the c program was executed.

Viva Questions:

1. How do you compile and run a C program inside your virtual machine?
2. What are the main service models of cloud computing?
3. What tools can be used for virtualization in a cloud lab?
4. Which cloud platform did you use in the lab?
5. What command-line tools did you use in the lab?

EXERCISE 3:

AIM:

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

Procedure:

- 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.

Step1: To install python

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/

Step 2: To install Google App Engine

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 **Google Application Engine** 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 After installation Google app engine looks that,

Step 3: Making of the FirstApplication

Now we 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 with in **apps** called" **ae--01--trivial**"–the path to this folder would be:

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

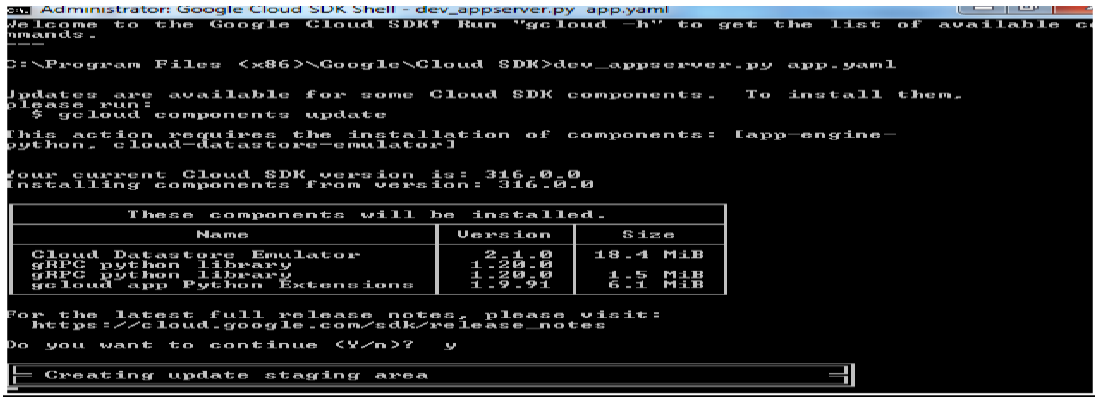
Using a text editor such as JEdit (www.jedit.org), create a file called **app.yaml** in the **ae--01--trivial** folder with the following contents:

```
application:  ae-01-
trivial version: 1
runtime: python
api_version: 1
handlers:
- url: /.*
  script: index.py
```

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 there Chuck'
```

Step 4: Run the program



Output:

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:

Result:

Thus the python application program was executed.

Viva Questions

1. What are the different service models in cloud computing?
2. What is public IP and private IP in cloud networks?
3. What is a Virtual Private Cloud (VPC)?
4. What is a subnet?
5. What is SSH? Why is it used in cloud computing?

EXERCISE 4:

AIM:

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.

Download Python 2.7

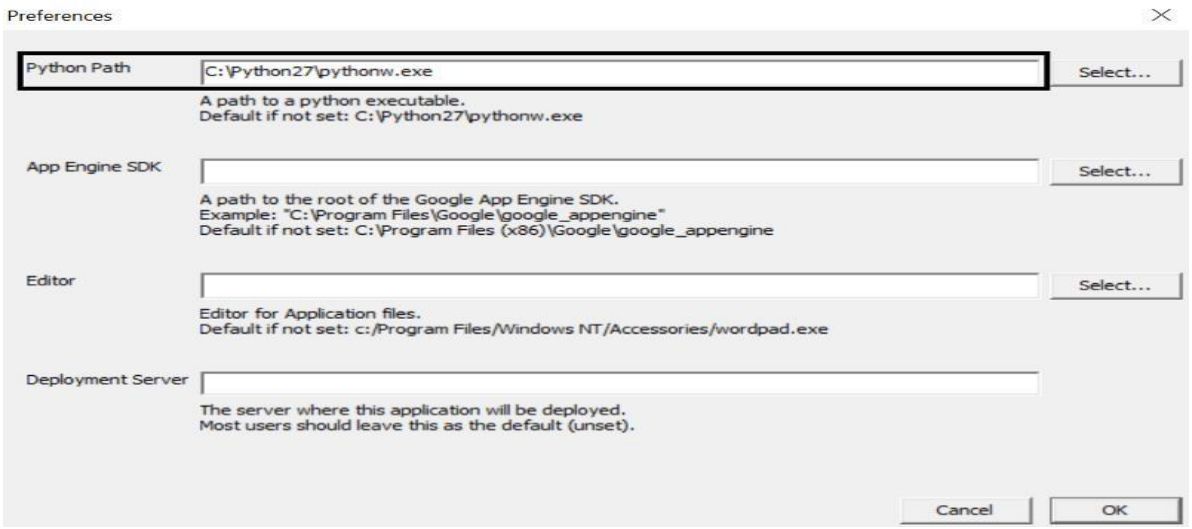
As of when this article was written, the Google App Engine standard environment supports Python only upto version 2.7. However, it is only a matter of time before support for Python 3.x is added. You can check the App Engine docs for the latest info.

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.

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 web apps.

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.

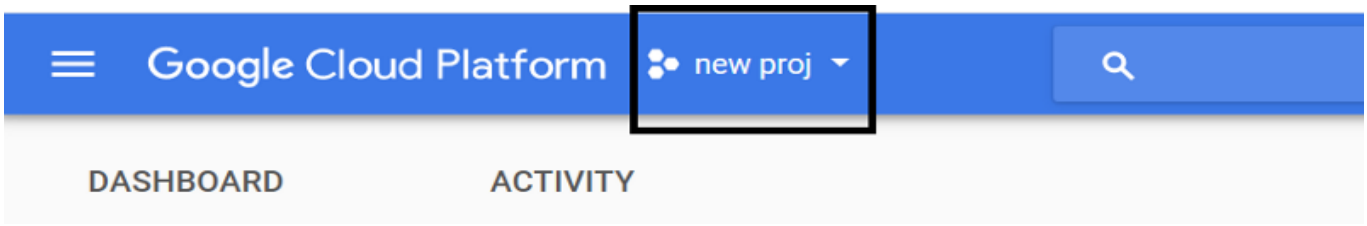
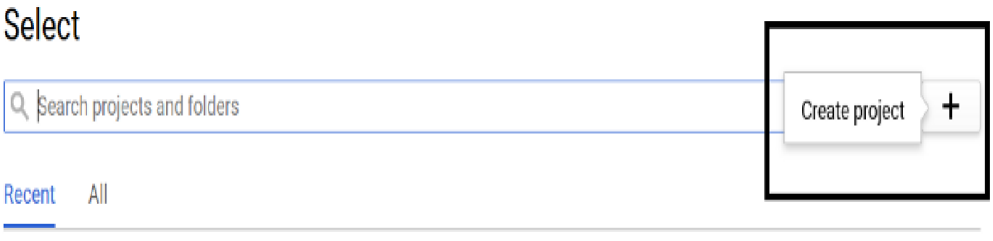
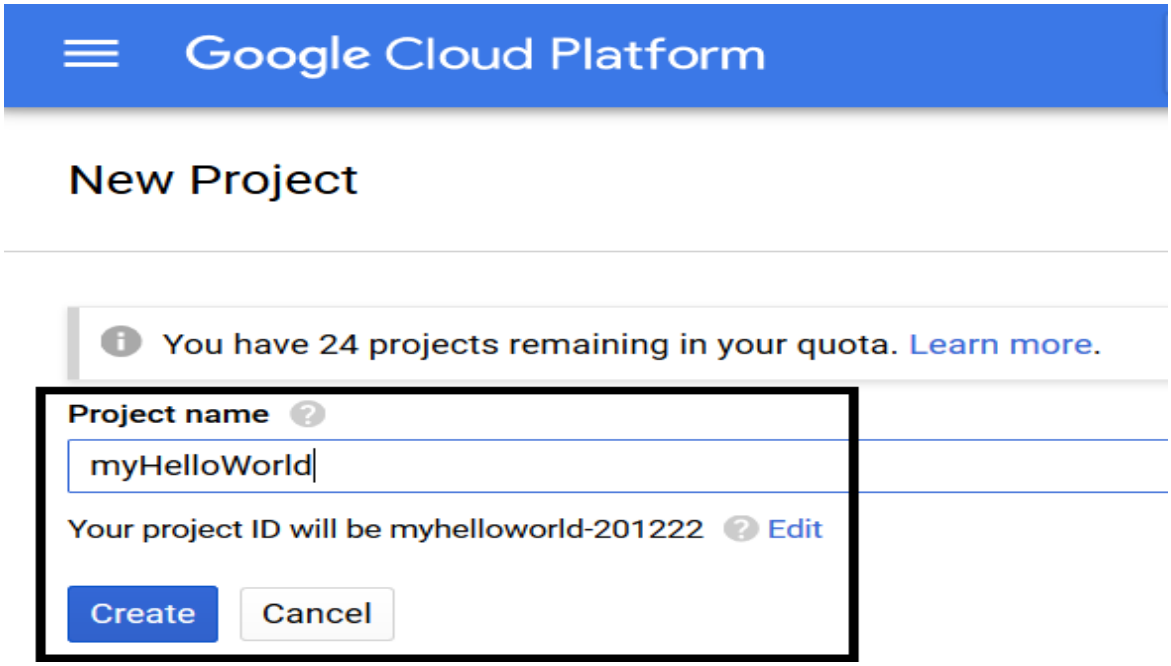


Image courtesy.

https://console.cloud.google.com/home



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



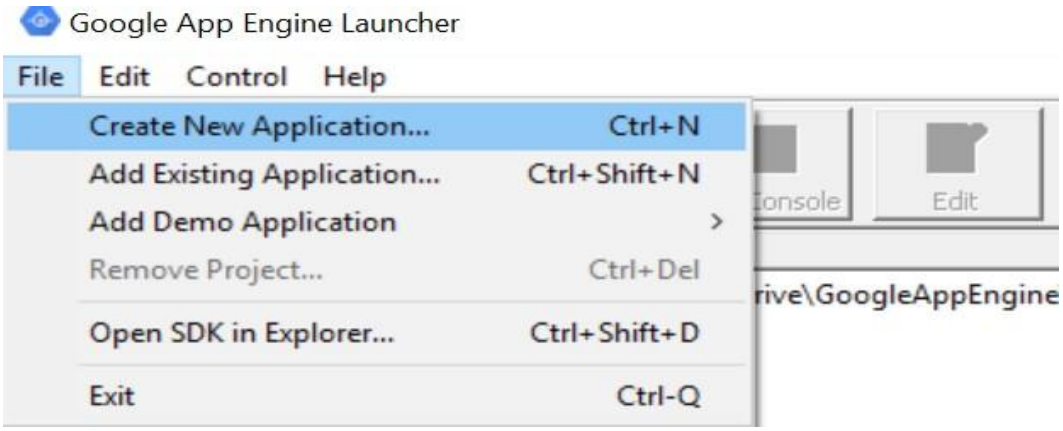
https://console.cloud.google.com/home

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

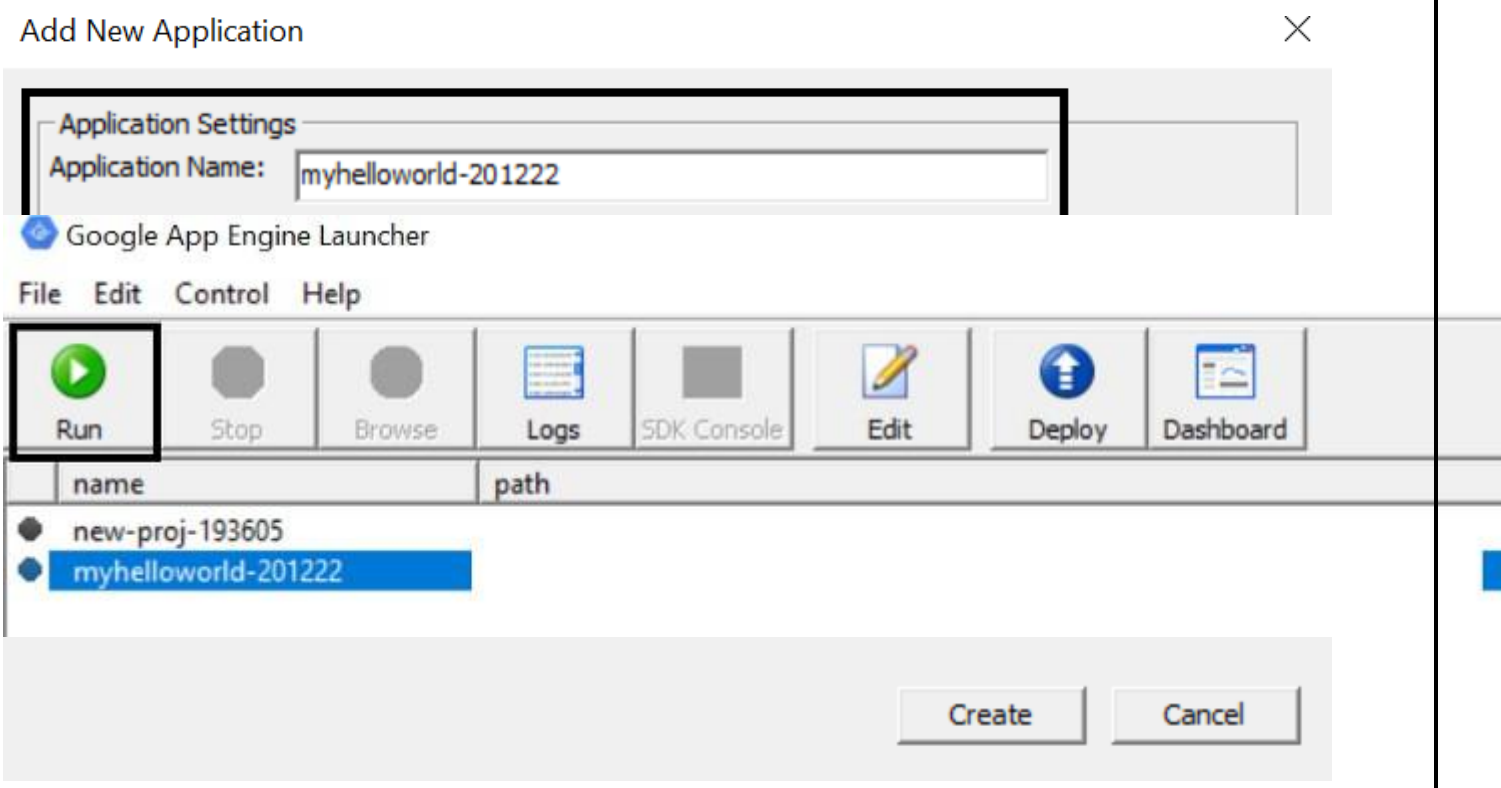
Step 4. Fork the app to develop it locally

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.



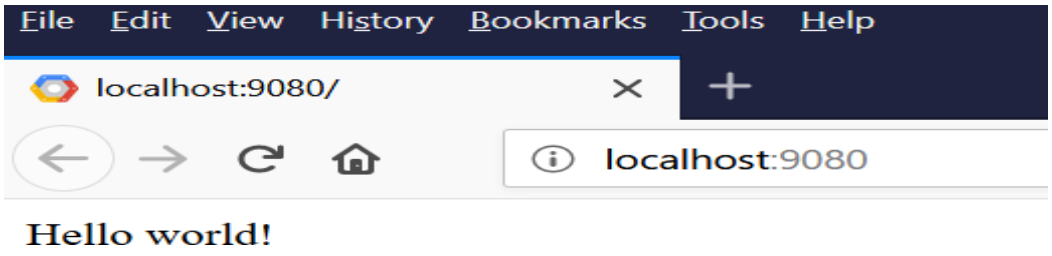
Hit the create button, and you should see your app listed on the window that follows. You should also check that you now see some files in your local storage (the directory you chose in the screenshot above) after this step.

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

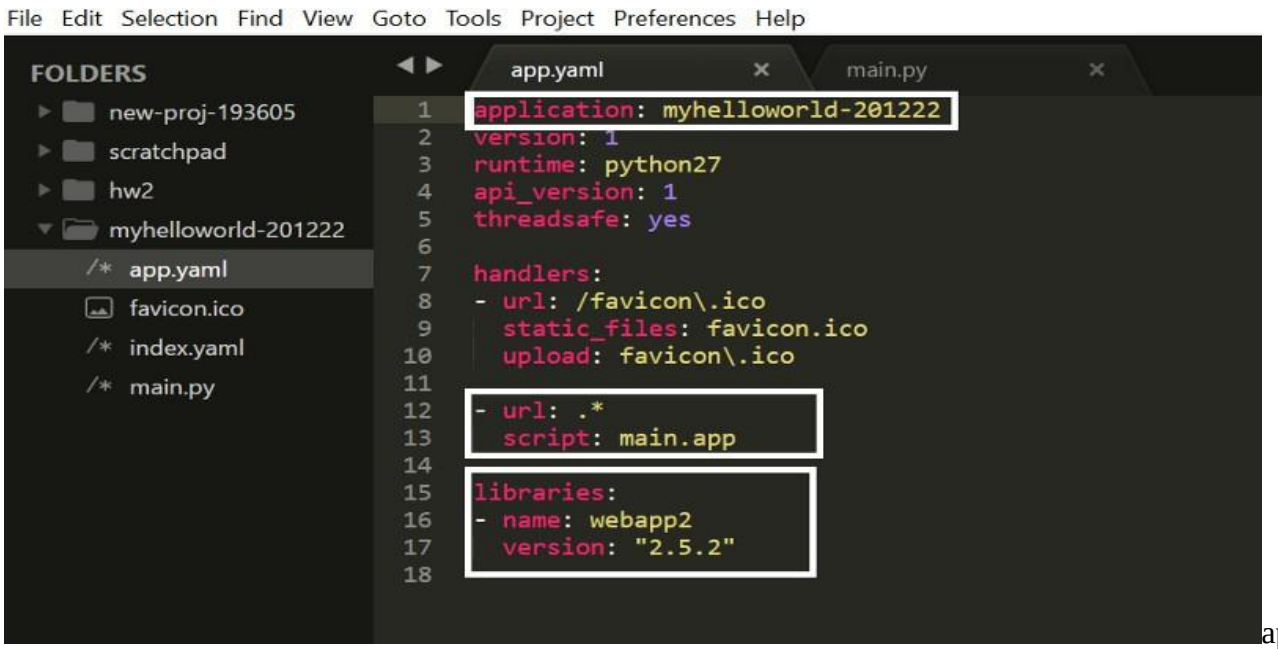
This file is a basic markup file that stores information (some metadata) about the app. It is important to note the following crucial parts of the file.

1. **application**

   This is the project ID which you should never change. This is the unique identifier for the app
2. **url -> scr**ipt

   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**

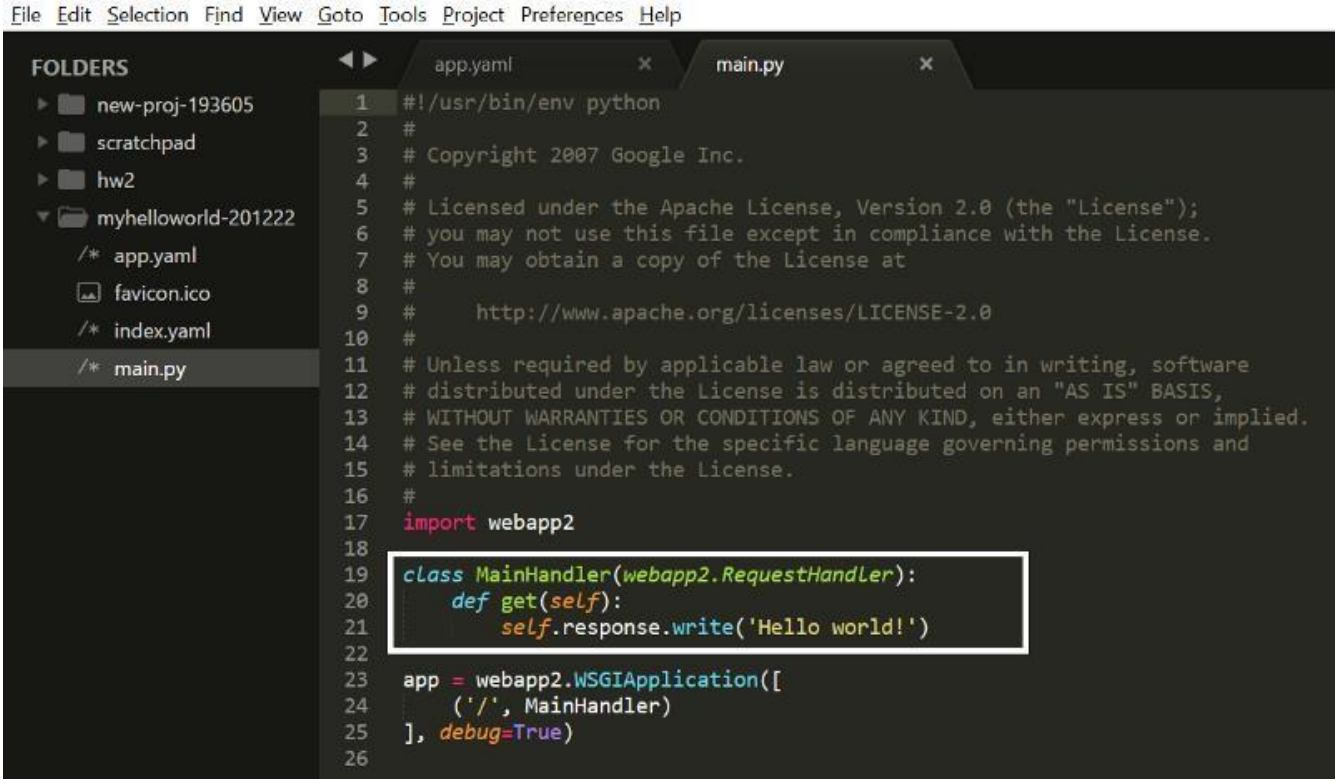
   This is where you can include external libraries to use within the webapp



app.yaml file in the web app folder

main.py program

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

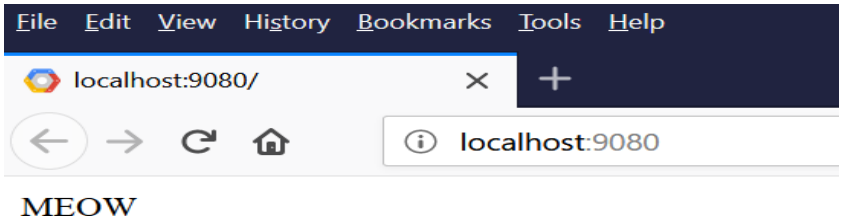
No hello world app is ever complete without the developer changing the hello world text to something else just to make sure that everything happening behind the scenes is working as it should.

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

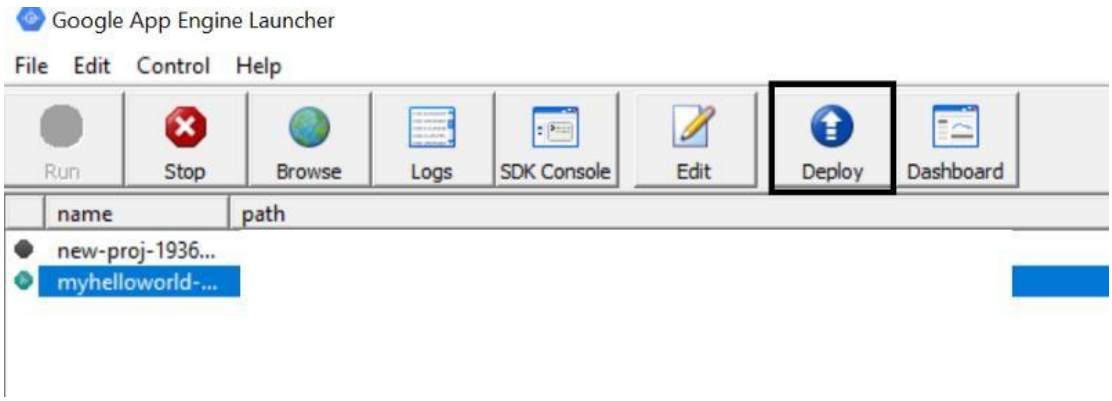
```
17  import webapp2
18
19  class MainHandler(webapp2.RequestHandler):
20      def get(self):
21          self.response.write('MEOW')
22
23  app = webapp2.WSGIApplication([
24      ('/', MainHandler)
25  ], debug=True)
26
```

Output:

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



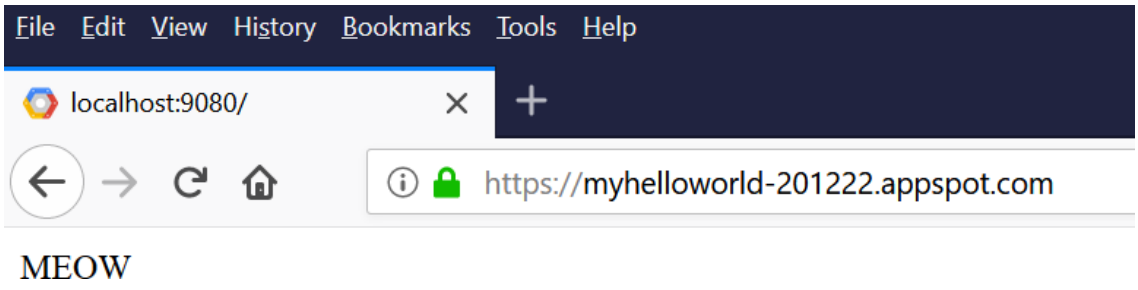
Finally, it is time to deploy your changes to the cloud to make them globally accessible via a URL. Go to the App Engine launcher, select the app, and hit the **Deploy** button.



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.



Result:

Thus the python web application is executed with the google app engine .

Viva questions

1. What is Google App Engine (GAE)?
2. What is the purpose of the GAE Launcher?
3. Which languages are supported by GAE?
4. What is app.yaml in GAE, and why is it important?
5. How do you deploy an app to Google App Engine using the Launcher?

EXERCISE 5:

AIM:

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

Procedure:

Clod Sim 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 class path and it is ready to be used. Please verify whether Java is available on your system.

To use CloudSim in Eclipse:

1. Download CloudSim installablefilesfrom *https://code.google.com/p/cloudsim/downloads/list and unzip*
2. OpenEclipse
3. Create a new Java Project: File ->New
4. Import an unpacked CloudSim project into the new JavaProject
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 Data center Characteristics 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 data center 9883 = new Datacenter(name, characteristics, new Vm Allocation Policy Simple(host List)

7.The third step is to create abroker:
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 cloud let Scheduler policy for cloudlets:

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

9.Submit the VM list to thebroker: broker.submitVmList(vmlist)

10 . Create a cloudlet with length, file size, output size, and utilisationmodel:
Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationMode

11.Submit the cloudlet list to thebroker: broker.submitCloudletList(cloudletList)

12. Start
The simulation: CloudSim.startSimulation()
Sample Output from the Existing Example: Starting

CloudSimExample1...

Initialising...

Starting CloudSim version 3.0 Datacenter_0 is starting...

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>null

Broker is

starting...

Entities

started.

: Broker: Cloud Resource List received with

1resource(s) 0.0: Broker: Trying to Create VM #0

inDatacenter_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...

Simulationcomplet

ed.

Simulation completed.

OUTPUT :

```
Cloudlet ID  STATUS  Data center ID   VM ID   Time   Start Time FinishTime
 0   SUCCESS                 20          400   0.1                   400.1
*****Datacenter:Datacenter_0***** Userid     Debt
    3          35.6
```

CloudSimExample1 finished!

RESULT:

Thus the scheduling algorithm is created with the CloudSim.

Viva Questions

1. What is CloudSim?
2. Why do we use CloudSim instead of real cloud platforms for testing?
3. What are the key components of CloudSim?
4. What is a Cloudlet in CloudSim?
5. Which classes did you modify or extend in CloudSim to implement your algorithm?

EXERCISE 6:

AIM:

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

Procedure:

Method1: Creating a Shared Folder in Virtual Box

A shared folder is a folder which makes its files available on both the guest machine *and* the host machine **at the same time**. Creating a shared folder between the guest and the host allows you to easily manage files which should be present on both machines. The course virtual machines are ready to use shared folders right away, but if you are using the virtual machine on your personal computer you will need to specify which folder to use as shared storage.

Shared Folders on SCS Lab Computers using Course VMs:

If you are using a course VM on a lab computer, it is likely that a shared folder has already been setup for you. On the desktop of your course VM you should notice a folder titled *SharedFolders*. Inside of this you will find any folders that have been shared between the course VM and lab computers.

You should see two folders that have already been configured for you: **Z_DRIVE** and **Temp**.

Z_DRIVE gives you access to your Windows Account `Z:\` drive. This is storage that is persistent to your SCS account and available as a network drive on the lab computers.

Temp gives you access to the folder found at `D:\temp` on the lab computer. Files stored in this folder are local to the machine, meaning that they can be accessed **faster**, but will **delete** from the system when you log out.

If you are working with data that you will need to use again, use the *Z_DRIVE* for your shared folder. If you need faster read/write speed, use the *Temp* folder, but remember to backup your files or they will be deleted when you log off the computer.

Shared Folders on Personal Computers:

If you are using your own personal machine, you will need to configure VirtualBox to look in the right place for your shared files.

First, click on the guest machine you intend to share files with. From there, you can select the guest*Settings* and navigate to *Shared Folders* on the left side menu. To create a new shared folder, either click the *New Folder* icon on the right menu **or** right click the empty list of shared folders and click *Add Shared Folder*. From here, there are six options:

- **Folder Path:** The folder name on the **host** machine. Click the drop down menu and navigate to the folder you would like to share.
- **Folder Name:** This is the name of the folder as it will appear on the **guest** machine.
- **Read-Only:** If you check read-only, the **guest** machine will be unable to write changes to the folder. This is valuable when you only want to send files *to* the virtual machine, but do not want to risk having the files modified by the guest.
- **Auto-Mount:** When any external storage is connected to a computer it must be *mounted* in order to be used. It is recommended that you turn on auto-mounting, unless you are familiar with the process of mounting a drive yourself.
- **Mount Point:** Unless you already know about mount points, leave this blank.
- **Make Permanent:** If you check this, the shared folder will be a permanent **machine folder**. If it is not checked, the folder will not be shared after a shutdown.

  On the course virtual machines, when you load into the desktop, you should see a folder labelled*SharedFolders*. In there you will see any folders that are currently mounted and being shared.

Steps:

1. Select the guest machine you wish to share files with
2. Click Settings > Shared Folders
3. Right-click and select Add Shared Folder and use the following settings:

   Folder Path: Click the dropdown arrow, select ***Other***, and navigate to the folder you would like to share
   Folder Name: Anything to identify it on the guest machine
   Read-Only: Unchecked (Checked, if you are exclusively pulling files ***from the host***)
   Auto-Mount: Checked
   Mount Point: Leave blank

4. Click OK

Method 2: Dragging and Dropping Files in VirtualBox

If you only need to transfer a few files quickly, you can simply drag and drop the files in. On the top bar of the running guest machine, click on *Devices* > *Drag and Drop* and make sure that *Bidirectional* is selected. This means that you will be able to drag files from the host to the guest and from the guest to the host. Once bidirectional drag and drop is checked, you should be able to begin dragging and dropping files.

NOTE: Sometimes when dragging files ***into* the course VM**, you may not be able to drag into the file browser directly. If you encounter this issue, you should drag your files onto the *Desktop* and move the files around from there. You should see the cursor change when it is ready to drop files.

You can also drag files from the guest machine into the host. To do this, simply open the file browser on the host to where you would like to drop the files and drag the files from the virtual machine into the file browser of the host. File transfers should be pretty quick; if the virtual machine seems stuck when transferring, simply cancel the transfer and try again.

Method 3: Managing Files with NextCloud

On any virtual machine, including VirtualBox, VMWare, or the virtual machines hosted on the SCS OpenStack, you can access the SCS Next Cloud services to move files between multiple machines and your SCS Windows Account storage. NextCloud offers you all of your SCS storage in one remote location, similar to how you might use other file hosting services like Dropbox or Google Drive. Before trying to use NextCloud, you should check that you can access the service by logging in here.

If you can access the Next Cloud services, you can browse the various file storage services available to you:
Linux Home: These are the files from your SCS Linux Account

- **Windows Home:** These are the files from your SCS Windows Account and your lab `Z:\` drive.
- **NextCloud:** In addition to the other storage accounts provided to you by the SCS, you can also upload up to 20GB of files directly to NextCloud.

  With NextCloud, you can upload your files from any machine with an internet connection and download them onto any other machine with an internet connection. For example, you can move project files off of your virtual machine, onto the NextCloud storage, and then download them on your personal laptop.

  Alternatively, you can upload files from your personal PC onto the NextCloud storage, place it into the*Windows Home* folder, and access those files from either the lab `Z:\` drive or download them on a virtual machine like VirtualBox or OpenStack.

Uploading Files to NextCloud from a Lab Computer:

If you would like to upload files from a lab computer, the easiest way to do this is to place the files you would like to transfer into your `Z:\` drive. These files will be automatically backup into your NextCloud storage under the *Windows Home* folder. After that, you can move them into the main NextCloud storage or choose to keep them in your `Z:\`drive.

Uploading Files to NextCloud from a VM or Other PC:

If you would like to upload files from either a VM or any other computer, you can login to the NextCloud service using any of the available interfaces, such as the web interface. Press the "+" icon in the top left of the file browser and select *Upload File*. From here, you can choose to keep it in the main NextCloud storage, move it into your Windows Account storage (the *Windows Home* folder), or into your Linux Account storage (the *Linux Home* folder).

Downloading NextCloud Files to a VM or Other PC:

Once your files are uploaded you will be able to download those files onto any machine which can connect to NextCloud. First, log in to your preferred NextCloud interface (eg. the web interface). N to the folder which contains the files you would like to download. Once you are in the target folder, click the checkbox next to each file you would like to download. Above the file listing you should notice the context barchanging to tell you how many files you have selected and a button labelled *Actions*. Click*Actions* > *Download*.

If you have selected a single file, it will prompt you to confirm the download. If you have chosen more than one file, NextCloud will place all of the selected files into a *zip archive*. Before you can use the files, you will need to extract them from the archive. Once you have downloaded your file, or extracted your archive, you are ready to use your files on your machine.

Result:

Thus the file is transferred from one virtual machine to another virtual machine.

Viva Questions

1. Why would you transfer files between two VMs?
2. What are some common protocols used for file transfer between VMs?
3. What is SSH, and how is it related to file transfer?
4. What is SCP, and how is it used?
5. How do you enable file sharing between two VMs on VirtualBox/VMware?

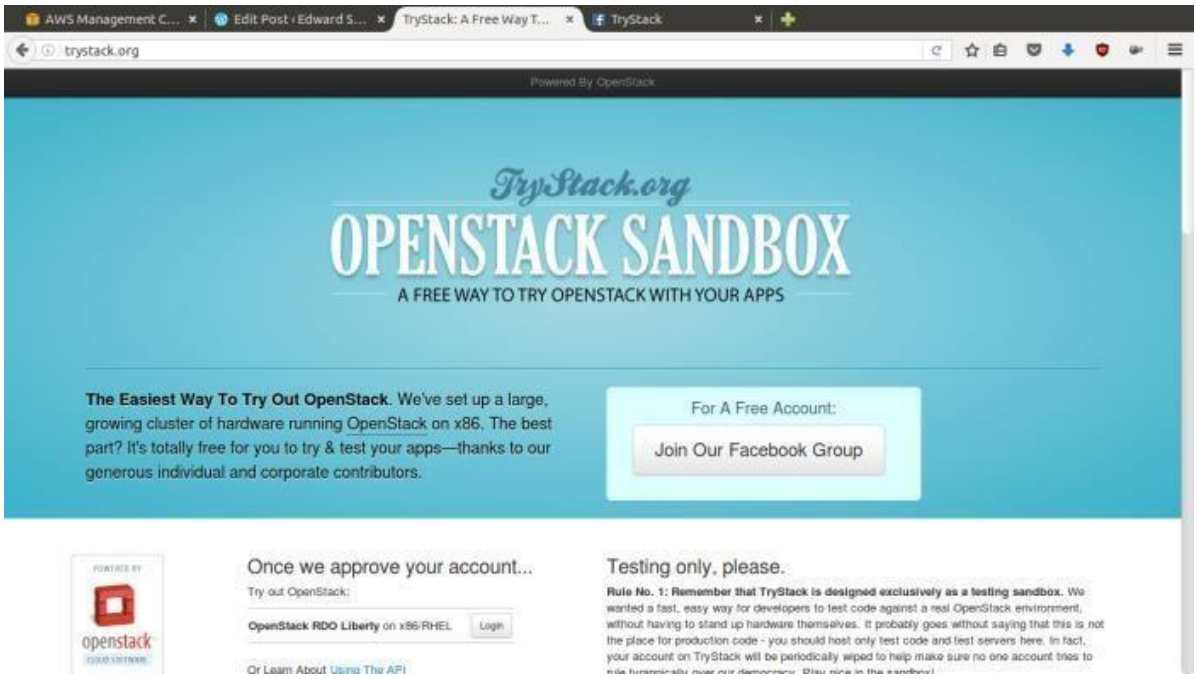
EXERCISE 7:

AIM:

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

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.

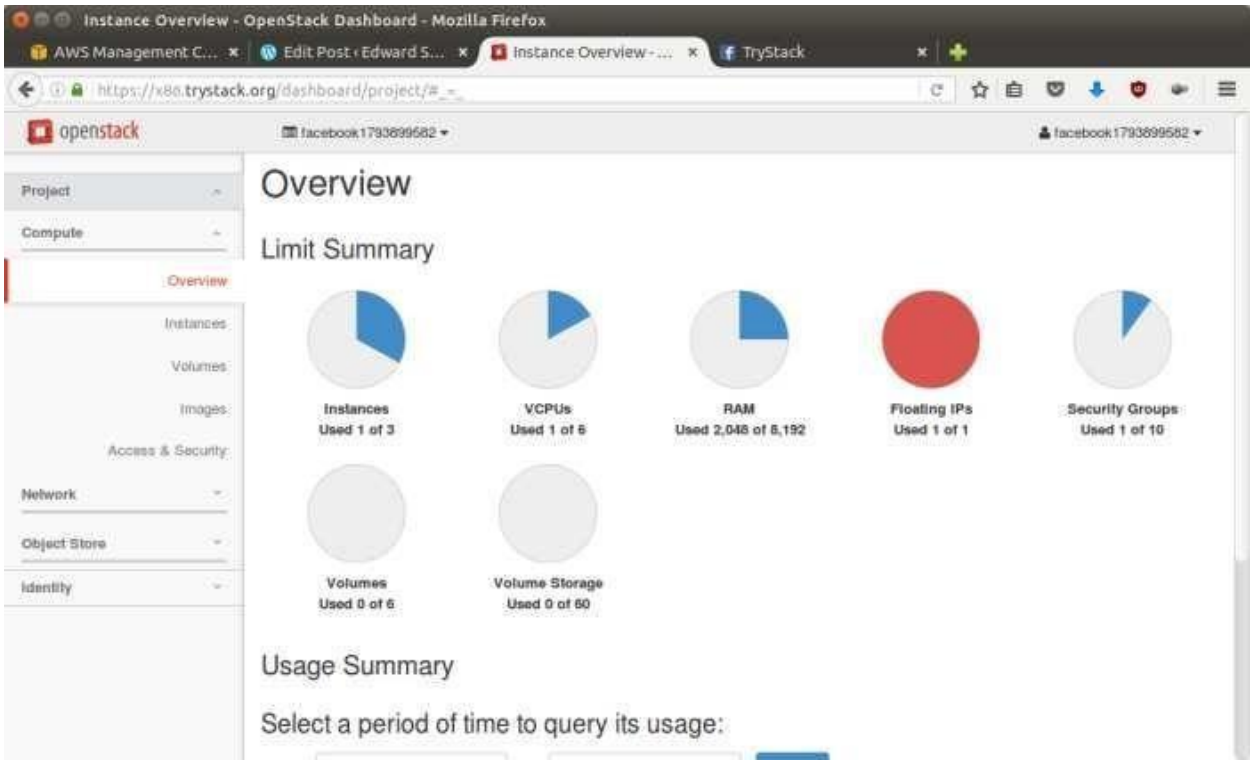
In order to try OpenStack in TryStack, you must register yourself by joining TryStack FacebookGroup. The acceptance of group needs a couple days because it's approved manually. After you have been accepted in the TryStack Group, you can log in TryStack.



TryStack.org Homepage

I assume that you already join to the Facebook Group and login to the dashboard.

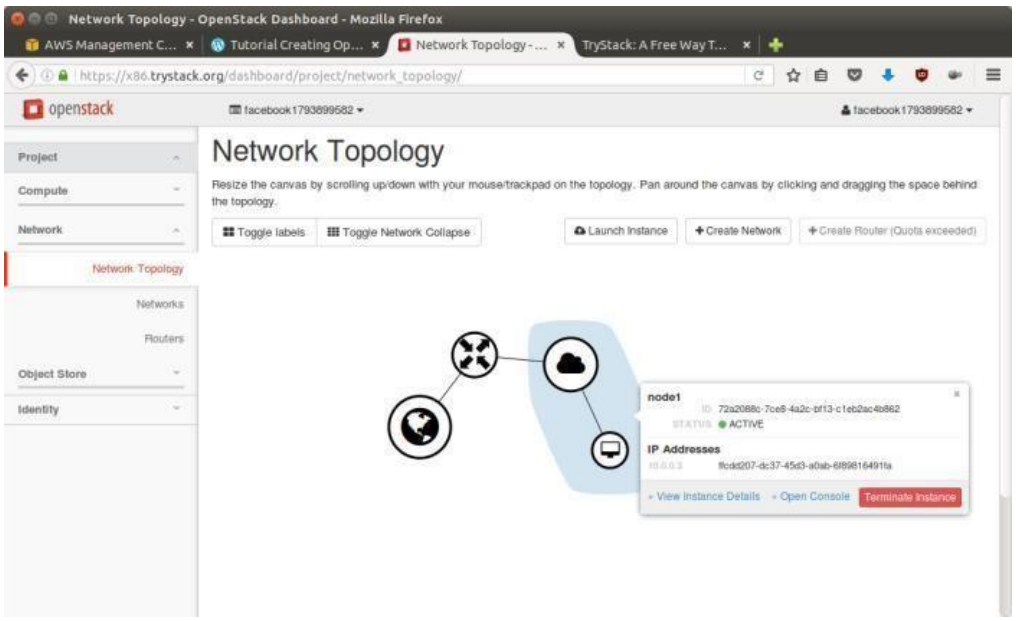
After youlog in to the TryStack, you will see the Compute Dashboard like:



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



:

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 **CreateNetwork**.
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 bestpractice.
    2. Select **IP Version** with appropriate IP version, in this caseIPv4.
    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 **LaunchInstance**.
2. In **Details**tab,
    1. Fill **Instance Name**, for example Ubuntu1.
    2. Select **Flavor**, for examplem1.medium.
    3. Fill **Instance Count** with**1**.
    4. Select **Instance Boot Source** with **Boot fromImage**.
    5. Select **Image Name** with **Ubuntu 14.04 amd64 (243.7 MB)** if you want installUbuntu 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 keythat we will use to connect to the instance from ourmachine.
    2. In **Import Key Pair**dialog,
        1. Fill **Key Pair Name** with your machine name (for exampleEdward-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 keypair.
3. Click **Import keypair**.

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 exampleinternal.
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 Ubuntu2.

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 **CreateRouter**.
2. Fill **Router Name** for example router1 and then click **Createrouter**.
3. Click on your **router name link**, for example router1, **Router Details**page.
4. Click **Set Gateway** button in upperright:
   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 **Addinterface**.
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. Thereare 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 FloatingIP**.
3. In **IP Address**, click Plus[+].
4. Select **Pool** to **external** and then click **AllocateIP**.
5. Click**Associate**.
6. Now you will get a public IP, e.g. 8.21.28.120, for yourinstance.

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 **ManageRules**.
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 virtual machine is launched using try stack.

Viva Questions

1. What is OpenStack?
2. What is TryStack?
3. Why do we use TryStack in labs?
4. What services are provided by OpenStack?
5. What is an "instance" in OpenStack?

EXERCISE 8 A:

AIM:

Install Hadoop single node cluster and run simple applications like word count.

Step 1: 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

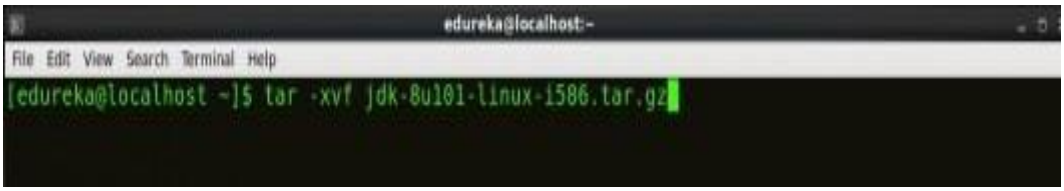


Fig: Hadoop Installation – Extracting Java Files

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

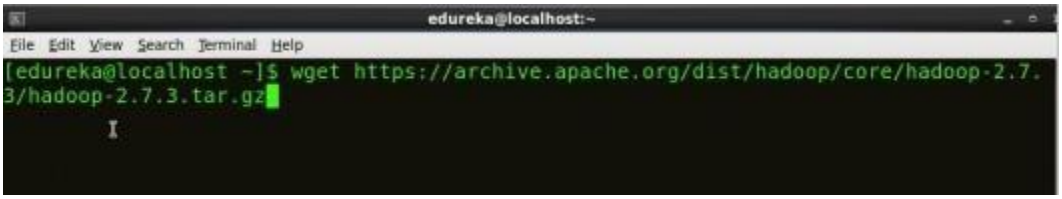


Fig: Hadoop Installation – Downloading Hadoop

Step 4: Extract the Hadoop tar File.

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

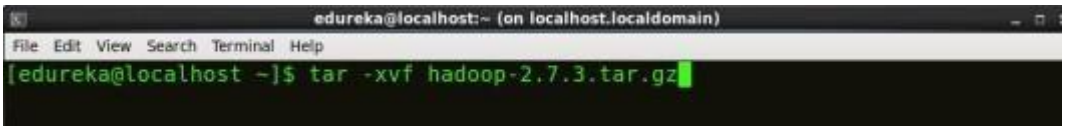


Fig: Hadoop Installation – Extracting Hadoop Files

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

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

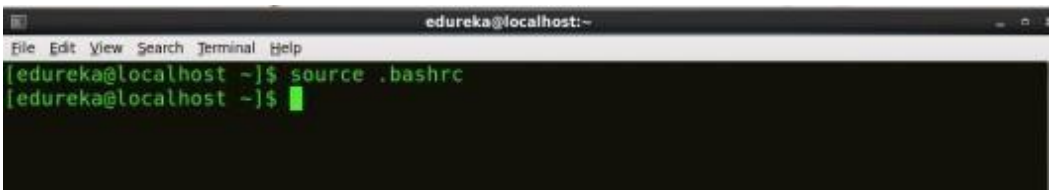


Fig: Hadoop Installation – Refreshing environment variables

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

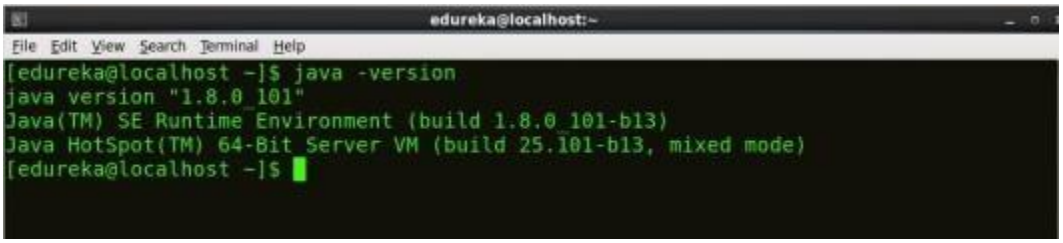


Fig: Hadoop Installation – Checking Java Version

***Command*:** hadoop version

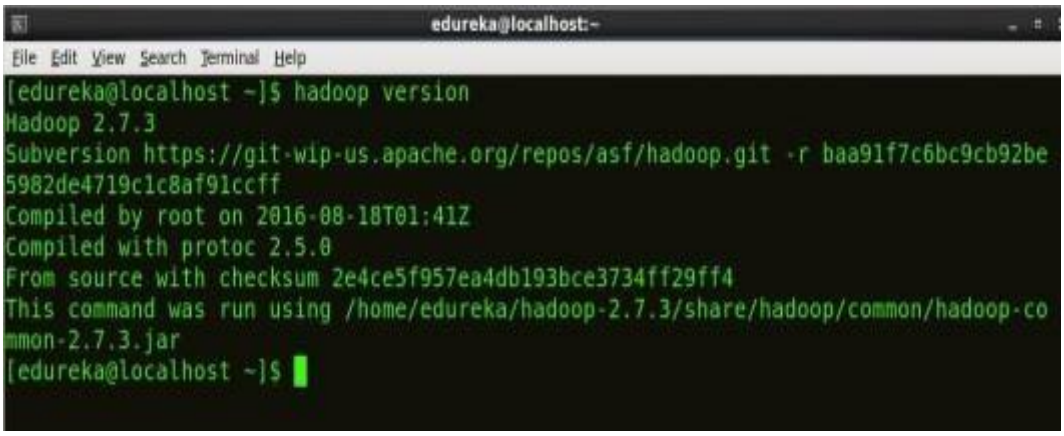


Fig: Hadoop Installation – Checking Hadoop Version

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:

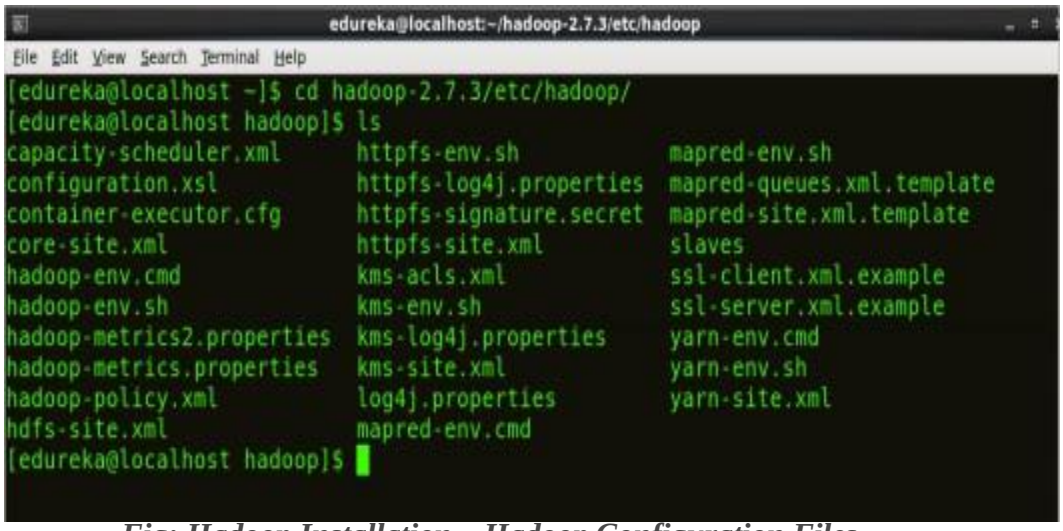


Fig: Hadoop Installation – Hadoop Configuration Files

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

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

***Command*:** vi core-site.xml

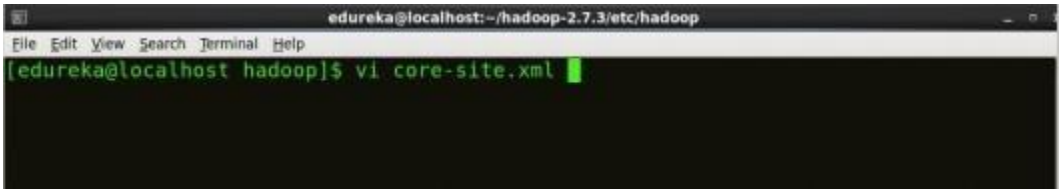


Fig: Hadoop Installation – Configuring core-site.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<?xml-stylesheettype="text/xsl"href="configuration.xsl"?>
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
</configuration>
```

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

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

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

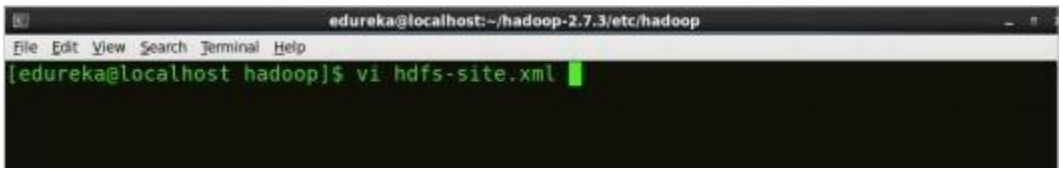


Fig: Hadoop Installation – Configuring hdfs-site.xml

```
1
2
3
4
5
6
7
8
9
```

```
<?xmlversion="1.0"encoding="UTF-8"?>
<?xml-stylesheettype="text/xsl"href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permission</name>
```

Step 9: Edit the *mapred-site.xml* file and edit the property mentionedbelow 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 map red site.xml file using mapred-site.xml template.

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

***Command*:** vi mapred-site.xml.

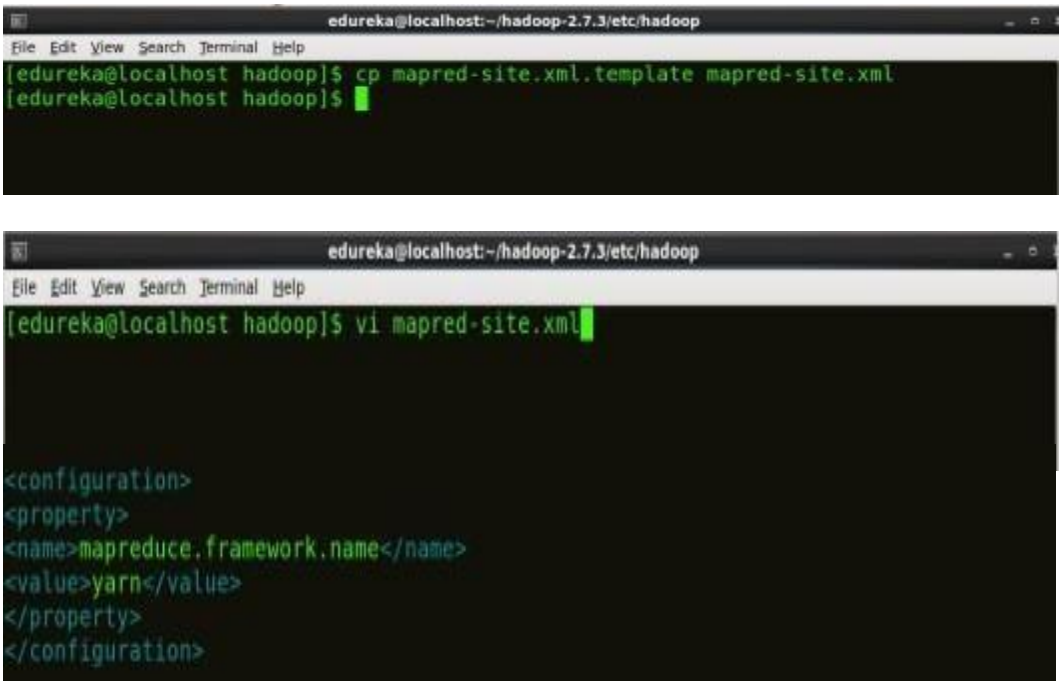


Fig: Hadoop Installation – Configuring mapred-site.xml

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

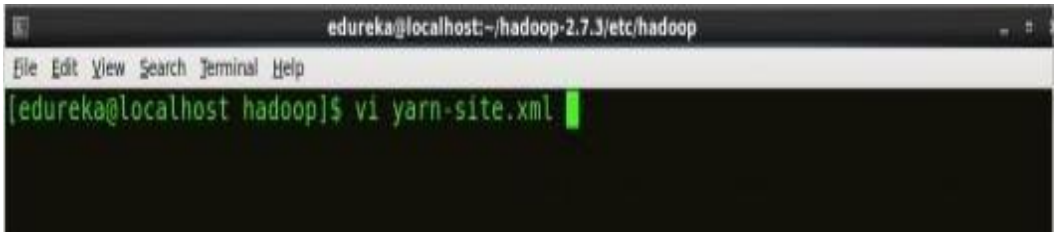


Fig: Hadoop Installation – Configuring yarn-site.xml

```
<?xmlversion="1.0">
<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>
```

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

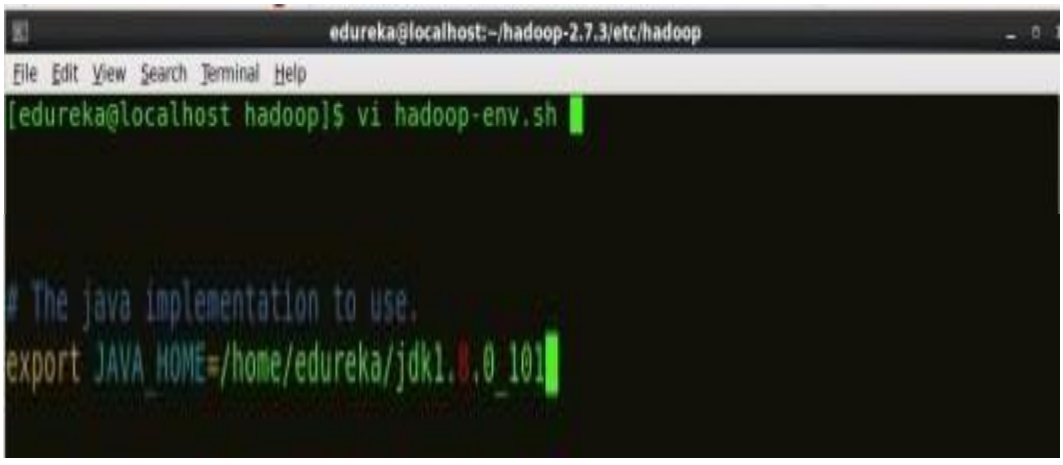


Fig: Hadoop Installation – Configuring hadoop-env.sh

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

***Command*:** cd

***Command*:** cd hadoop-2.7.3

***Command*:** bin/hadoop namenode -format

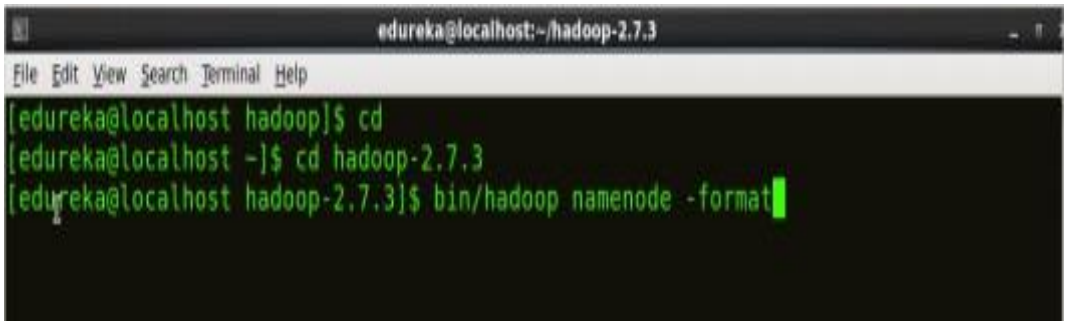


Fig: Hadoop Installation – Formatting NameNode

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

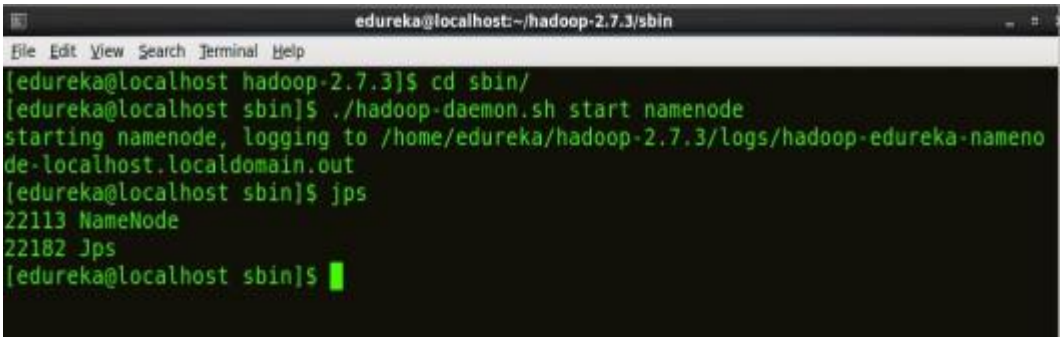
The above command is a combination of ***start-dfs.sh, start-yarn.sh*** &***mr-jobhistory-daemon.sh***

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



StartDataNode:

– *Starting NameNode*

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

Command: ./hadoop-daemon.sh start datanode

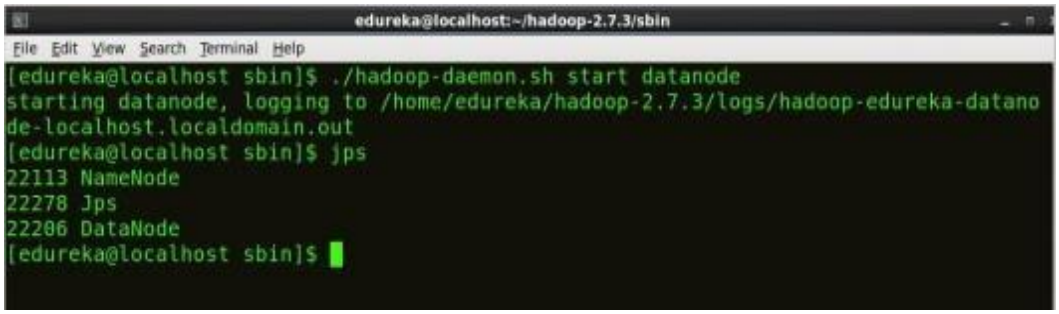


Fig: Hadoop Installation – Starting DataNode

Start Resource Manager:

Resource Manager 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 Application Master.

Command: ./yarn-daemon.sh start resourcemanager

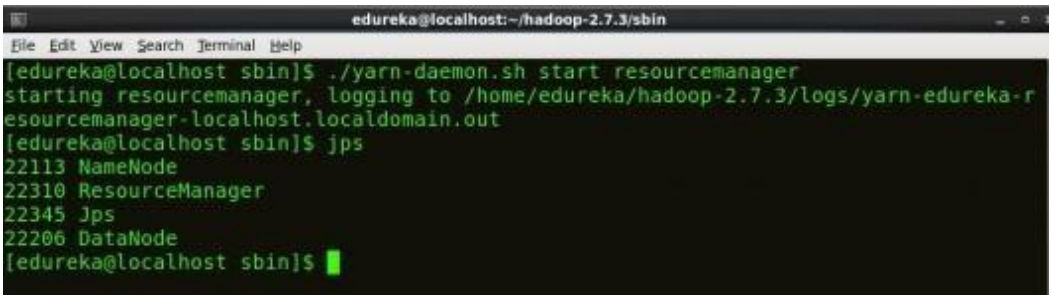


Fig: Hadoop Installation – Starting ResourceManager

Start Node Manager:

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

Command: ./yarn-daemon.sh start node manager

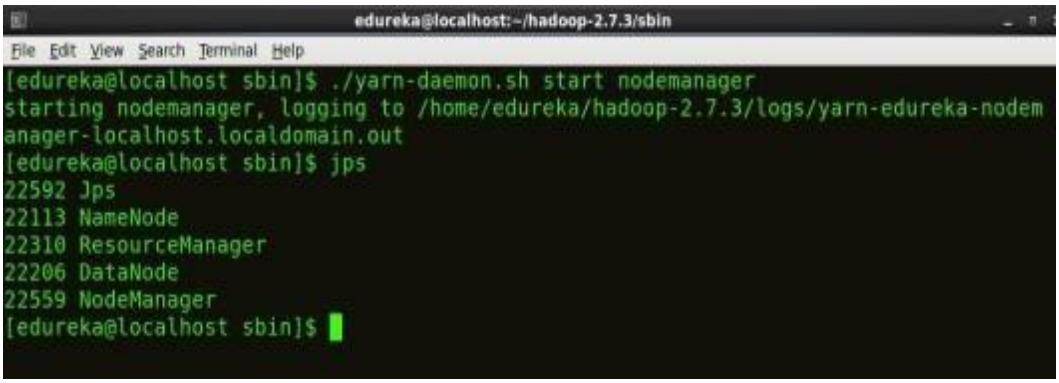


Fig: Hadoop Installation – Starting NodeManager

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: jps

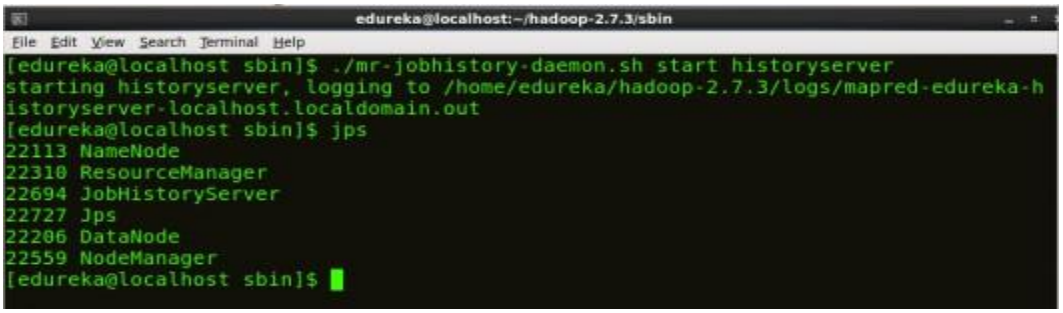


Fig: Hadoop Installation – Checking Daemons

Step 15: Now open the Mozilla browser and go

to **localhost**:**50070/dfshealth.html** to check the NameNode interface.

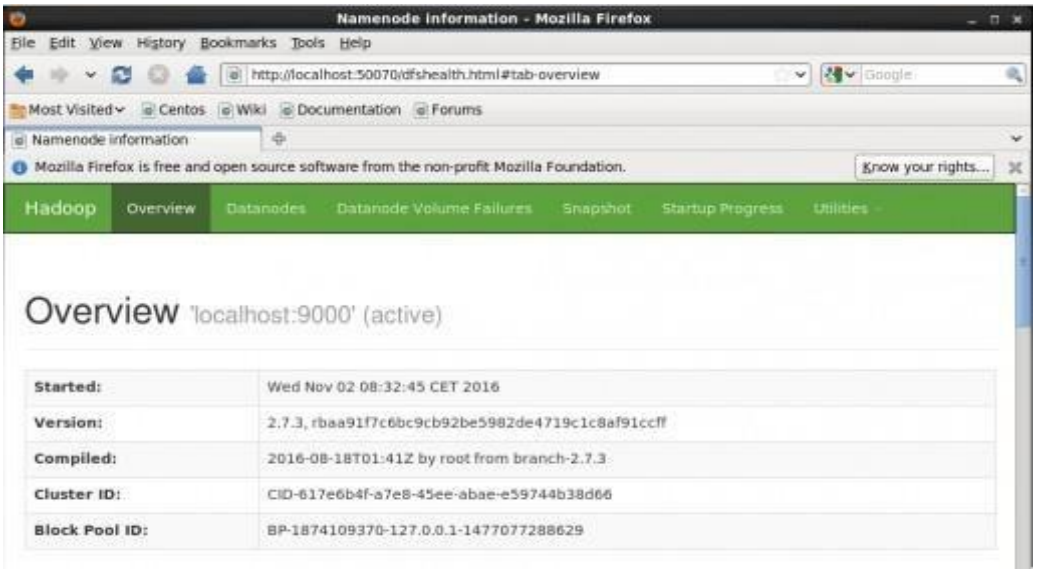


Fig: Hadoop Installation – Starting WebUI

- Congratulations, you have successfully installed a single node Hadoop cluster

Result:

Thus the Hadoop single node cluster was created successfully.

Viva Questions

1. What is Hadoop?
2. What are the main components of Hadoop?
3. What is a single-node Hadoop cluster?
4. What are the different modes in which Hadoop can run?
5. What is the difference between HDFS and MapReduce?

Exno8b.Wordcount program to demonstrate the use of Map and Reduce tasks

Aim:

To write a word count program to demonstrate the use of Map and Reduce tasks.

Procedure:

Step1

hduser@nspublin:/usr/local/hadoop/sbin$ mkdir /home/hduser/wc

Step 2:Compiling the java file - WordCount.java

hduser@nspublin:/usr/local/hadoop/sbin$ sudo /usr/lib/jvm/java-8-oracle/bin/javac -classpath /home/hduser/hadoop-core-1.2.1.jar -d /home/hduser/wc /home/hduser/WordCount.java

Step 3 :Creating jar file for wordCount.java:

hduser@nspublin:/usr/local/hadoop/sbin$ jar -cvf /home/hduser/wc.jar -C /home/hduser/wc/ .

added manifest

adding: WordCount$IntSumReducer.class(in = 1739) (out= 739)(deflated 57%)

adding: WordCount$TokenizerMapper.class(in = 1736) (out= 753)(deflated 56%)

adding: WordCount.class(in = 1491) (out= 814)(deflated 45%)

Step4.Executing jar file for WordCount.javahduser@ksrietcsevb:/usr/local/hadoop/sbin$ hadoop jar /home/hduser/wc.jar WordCount /user/input

/user/output

Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar

16/09/12 10:52:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...

using builtin-java classes where applicable

Step5: to check the file in Output file

hduser@ksrietcsevb:/usr/local/hadoop/sbin$ hadoop fs -ls /user/output

Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar

16/09/12 10:56:22 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 hdusersupergroup 0 2016-09-12 10:56 /user/output/_SUCCESS

-rw-r--r-- 1 hdusersupergroup 182 2016-09-12 10:56 /user/output/part-r-00000

Program:

/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 {
StringTokenizeritr = 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 (IntWritableval : 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); }
}
```

Input:

hai i am in gcc lab

Output

hduser@ksrietcsevb:/usr/local/hadoop/sbin$ hadoop fs -cat /user/output/*

Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar

16/09/12 10:56:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

```
am      1
gcc     1
hai     1
i       1
in      1
lab     1
```

hduser@ksrietcsevb:/usr/local/hadoop/sbin$

Result:

Thus the word count program was executed using hadoop map reduce function.

Viva Questions

1. What is MapReduce?
2. What is the purpose of the WordCount program in Hadoop?
3. What are the main phases of MapReduce?
4. What input does the WordCount Mapper take?
5. What output does the Mapper produce in WordCount?

ADDITIONAL EXPERIMENTS:

EXERCISE 1:

AIM:

Creating and Executing your First container using Docker

Procedure:

1. Docker is fast. Unlike a virtual machine, your application can start in a few seconds and stop just as quickly.
2. Docker is multi-platform. You can launch your container on any system.
3. Containers can be built and destroyed faster than a virtual machine.
4. No more difficulties setting up your working environment. Once your Docker is configured, you will never have to reinstall your dependencies manually again. If you change computers or if an employee joins your company, you only have to give them your configuration.
5. You keep your work-space clean, as each of your environments will be isolated and you can delete them at any time without impacting the rest.
6. It will be easier to deploy your project on your server in order to put it online.

Now let's create your first application

- Now that you know what Docker is, it's time to create your first application!
- The purpose of this short tutorial is to create a Python program that displays a sentence.
  This program will have to be launched through a Dockerfile.
- You will see, it's not very complicated once you understand the process.

Note: You will not need to install Python on your computer. It will be up to the Docker environment to contain Python in order to execute your code. **1. Install Docker on your machine** For Ubuntu:
First, update your packages:
$ sudo apt update
Next, install docker with apt-get:
$ sudo apt install docker.io
Finally, verify that Docker is installed correctly:
$ sudo docker run hello-world

- For MacOSX: you can follow this link.
- For Windows: you can follow this link

change computers or if an employee joins your company, you only have to give them your configuration.
7.You keep your work-space clean, as each of your environments will be isolated and you can delete them at any time without impacting the rest.

8.It will be easier to deploy your project on your server in order to put it online.

Now let's create your first application

- Now that you know what Docker is, it's time to create your first application!
- The purpose of this short tutorial is to create a Python program that displays a sentence.
  This program will have to be launched through a Dockerfile.
- You will see, it's not very complicated once you understand the process.

Note: You will not need to install Python on your computer. It will be up to the Docker environment to

contain Python in order to execute your code. **1. Install Docker on your machine** For Ubuntu:
First, update your packages:
$ sudo apt update
Next, install docker with apt-get:
$ sudo apt install docker.io
Finally, verify that Docker is installed correctly:
$ sudo docker run hello-world

- For MacOSX: you can follow this link.
- For Windows: you can follow this link.

1. Create your project

In order to create your first Docker application, I invite you to create a folder on your computer. It must contain the following two files:

- A 'main.py' file (python file that will contain the code to be executed).

- A 'Dockerfile' file (Docker file that will contain the necessary instructions to create the environment).

Normally you should have this folder architecture:

```
.
├── Dockerfile
└── main.py
0 directories, 2 files
```

2. Edit the Python file
You can add the following code to the 'main.py' file:

```
#!/usr/bin/env python3
print("Docker is magic!")
```

Nothing exceptional, but once you see "Docker is magic!" displayed in your terminal you will know that your Docker is working.
3.Edit the Docker file
Some theory: the first thing to do when you want to create your Dockerfile is to ask yourself what you want to do. Our goal here is to launch Python code.

- To do this, our Docker must contain all the dependencies necessary to launch Python. A linux (Ubuntu) with Python installed on it should be enough.
- The first step to take when you create a Docker file is to access the DockerHub website. This site contains many pre-designed images to save your time (for example: all images for linux or code languages).
- In our case, we will type 'Python' in the search bar. The first result is the official image created to execute Python. Perfect, we'll use it!

```
# A dockerfile must always start by importing the base image.
# We use the keyword 'FROM' to do that.
# In our example, we want import the python image.
# So we write 'python' for the image name and 'latest' for the version.
FROM python:latest
# In order to launch our python code, we must import it into our image.
# We use the keyword 'COPY' to do that.
# The first parameter 'main.py' is the name of the file on the host.
# The second parameter '/' is the path where to put the file on the image.
```

```
# Here we put the file at the image root folder.
COPY main.py /

# We need to define the command to launch when we are going to run the image.
# We use the keyword 'CMD' to do that.
# The following command will execute "python ./main.py".
CMD [ "python", "./main.py" ]
```

3.Create the Docker image

Once your code is ready and the Dockerfile is written, all you have to do is create your image to contain your application.

```
$ docker build -t python-test .
```

The '-t' option allows you to define the name of your image. In our case we have chosen 'python-test' but you can put what you want.

4.Run the Docker image

- Once the image is created, your code is ready to be launched. ➢ $ docker run python-test
- You need to put the name of your image after 'docker run'.
- There you go, that's it. You should normally see "Docker is magic!" displayed in your terminal.
- Code is available
- If you want to retrieve the complete code to discover it easily or to execute it, I have put it at your disposal on my GitHub.
  - ➔ GitHub: Docker First Application example
- Useful commands for Docker
- Before I leave you, I have prepared a list of commands that may be useful to you on Docker.

• List your images.

```
$ docker image ls
```

• Delete a specific image.

```
$ docker image rm [image name] •
```

Delete all existing images.

```
$ docker image rm $(docker images -a -q)
```

• List all existing containers (running and not running). $ docker ps -a

• Stop a specific container.

```
$ docker stop [container name] •
```

Stop all running containers.

```
$ docker stop $(docker ps -a -q)
```

• Delete a specific container (only if stopped).

```
$ docker rm [container name]
```

• Delete all containers (only if stopped).

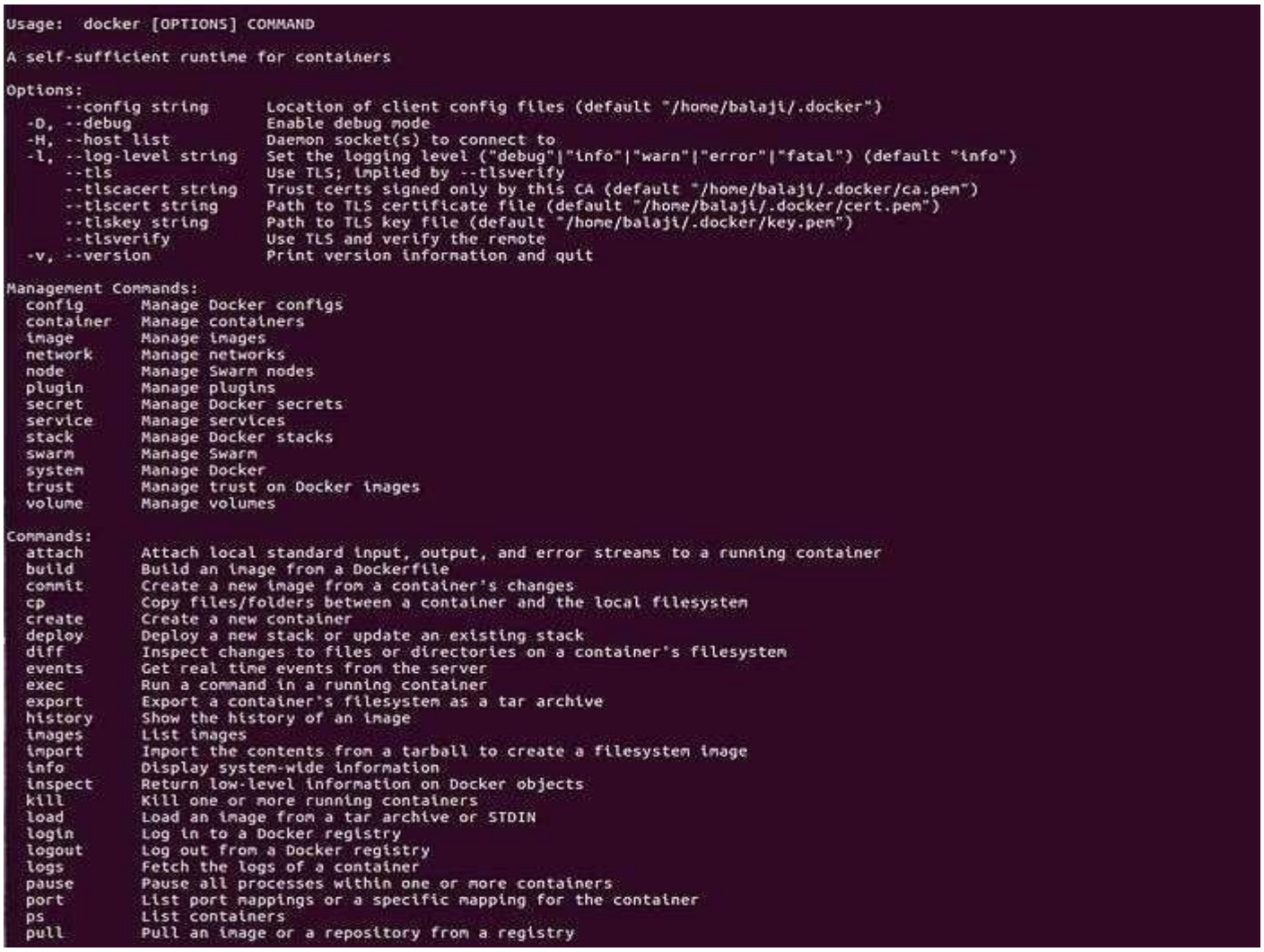
```
$ docker rm $(docker ps -a -q)
```

• Display logs of a container.

```
$ docker logs [container name]
```

This is a very simple tutorial for getting started with Docker. I'll try to keep this as simple as possible.

In this, we are going to build a basic Flask application and dockerize the application. By the end of this tutorial, you'll get familiar with Docker and a few Docker commands.



Docker man page

Overview of Docker

Docker is a platform for developing and running applications. It automates the deployment of applications. It's a tool for running applications in an **isolated** environment. Docker makes it easy to share an application with all of its dependencies across different environments.

Docker Image

Image is a template for creating an environment. The Docker image contains the Operating System, Software, and application code. These are all packaged in a single file. Images are defined with the Dockerfile.

Dockerfile

Dockerfile is built into a docker image

The Dockerfile contains the steps that are needed to package your application(to create the image). These

steps include configuring the Operating system, install the required packages or software, copy the files from one place to another.

Create the Flask application

Let's dive into code

Flask is the micro web framework for building small web applications.
The directory structure is as follows:

```
flaskapp/
├──── Dockerfile
├──── app.py
├──── requirements.txt
```

Create a directory with the name of your choice. The command for creating a directory in Linux is,

```
$mkdir flaskapp
```

Navigate inside the directory you created. Now create a file named app.py and copy the following code to it.

app.py

```
from flask import Flask app = Flask(_name_)@app.route('/') def
hello_world():          return 'Hello, This is my first Docker app!'if
__name___== "__main__":     app.run(debug=True, host='0.0.0.0')
```

This is a simple Hello world Flask app. If you want to learn about flask in detail, then visit this page and explore about flask. Let's concentrate more on Docker now. Then create the requirements.txt file.

Add the following line to it.

requirements.txt

Now we are going to create a Dockerfile that dockerizes this application.

```
Flask==1.1.2
```

Write the Docker File

The Dockerfile should not have any extension.

If you haven't downloaded the docker yet. Get the Docker from here. Detailed installation instructions can be found here.

Create a file called Dockerfile. This must start with capital letter D.

Dockerfile

FROM alpine:3.11RUN apk add--update pythonRUN apk add--update py-pipCOPY ./requirements.txt /app/requirements.txtCOPY . /appWORKDIR /appRUN pip instarllrequirements.txtEXPOSE 5000CMD python app.py

Let's look at the instructions line by line to have a better understanding

```
FROM alpine:3.11
```

This is our base image. There are a lot of images available in the Dockerhub. We choose alpine since it's one of a lightweight image.

The **FROM** allows us to initialize the build over the base image. The number after the colon is the version number. A valid Dockerfile always starts with FROM keyword.

```
RUN apk --update add python
RUN apk add --update py-pip
```

These two lines are used to install the python and the pip package respectively. The **RUN** instruction will execute a new layer on top of the current image. This instruction is used to install packages and creating new directories.

```
COPY ./requirements.txt /app/requirements.txtCOPY . /app
```

The **COPY** instruction is used to copy the files from source(.) to the destination(/app). The '.' represents the current directory. This basically copies the flask app into the image.

WORKDIR /app

The **WORKDIR** sets the working directory.

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

This reads the requirements.txt file and installs the specified packages one by one on the host.

```
EXPOSE 5000
```

The **EXPOSE** exposes a port that is used by Flask. When you run the image you'll get a container that container will run on this port.

```
CMD python app.py
```

CMD is the command that is executed when you start a container. Here, you are using the command to run your Python application. There can be only one CMD per Dockerfile. If you specify more than one, then the last `CMD` will take effect.

When you start a container this command is executed. There should be only one **CMD** instruction in the Dockerfile. If there is more than one, it'll execute the last instruction.

Build the docker image

Enough of information. Let's run the application

To build the Docker image, execute this command in the terminal in the directory we created.

```
$docker build -t flask-app .
```

The `-t` is used to name your image flask-app. The `.` , in the end, represents the current directory. When you execute this for the first time, it'll have to download all of the layers that make up to build the image. After that, it'll use the cache. If you're not an administrator, try to run the command with the `sudo`.

```
Sending build context to Docker daemon  4.096kB
Step 1/9 : FROM alpine:3.11
 ---> f70734b6a266
Step 2/9 : RUN apk add --update python
 ---> Using cache
 ---> e9800bc501b3
Step 3/9 : RUN apk add --update py-pip
 ---> Using cache
 ---> b48cca251e6b
Step 4/9 : COPY ./requirements.txt /app/requirements.txt
 ---> Using cache
 ---> b2c052f9bdc3
Step 5/9 : COPY . /app
 ---> e50bd7a55434
Step 6/9 : WORKDIR /app
 ---> Running in 8488a87861d7
Removing intermediate container 8488a87861d7
 ---> d0d28ec2d9ff
Step 7/9 : RUN pip install -r requirements.txt
 ---> Running in 05ad40e31889
Collecting Flask==1.1.2 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/
Collecting Jinja2>=2.10.1 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/30/9e/f663a2aa66a09d838042ae1a2c5659828bb9b41ea3a6efa20a20fd92b121/
Collecting click>=5.1 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/d2/3d/fa76db83bf75c4f8d338c2fd15c8d33fdd7ad23a9b5e57eb6c5de26b430e/
Collecting Werkzeug>=0.15 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/cc/94/5f7079a0e00bd6863ef8f1da638721e9da21e5bacee597595b318f71d62e/
Collecting itsdangerous>=0.24 (from Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->Flask==1.1.2->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/b9/2e/64db92e53b86efccfaea71321f597fa2e1b2bd3853d8ce658568f7a13094/
Installing collected packages: MarkupSafe, Jinja2, click, Werkzeug, itsdangerous, Flask
  Running setup.py install for MarkupSafe: started
    Running setup.py install for MarkupSafe: finished with status 'done'
Successfully installed Flask-1.1.2 Jinja2-2.11.2 MarkupSafe-1.1.1 Werkzeug-1.0.1 click-7.1.2 itsdangerous-1.1.0
Removing intermediate container 05ad40e31889
 ---> b328b914c2c5
Step 8/9 : EXPOSE 5000
 ---> Running in ffbcaffcc010
Removing intermediate container ffbcaffcc010
 ---> 5916f995523c
Step 9/9 : CMD python app.py
 ---> Running in ebe11a334937
Removing intermediate container ebe11a334937
 ---> 785b60cf0cdb
Successfully built 785b60cf0cdb
Successfully tagged flask-app:latest
```

docker build log

To run the application, execute this command

```
$docker run -p 5000:5000 flask-app
```

The -p is used to map the *port running inside the container to your host. Here we're mapping the port 5000.*

The value before the colon represents the port running on your host and the value after colon represents the port running inside the container.

```
* Serving Flask app "app" (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://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 853-721-745
```

Now, navigate to http://localhost:5000/ in your browser, you'll see the "Hello, This is my first Docker app!"

in the window.

Container

To know the container id enter the following command in your terminal.

```
$docker ps
```

This lists all the running containers in the Docker engine. This has information about containers like its ID,

created time, status and on which port it's running.

```
$docker ps-a
```

This will list the containers that have been stopped as well.

```
$docker stop [container name]
```

This will stop all the running containers.

Result

Thus the program Creating and Executing First container using Docker is run successfully

EXERCISE 2:

AIM:

Run a container from Docker Hub

PROCEDURE:

Step 1: Get the sample application. If you have git, you can clone the repository for the sample application. ...
Step 2: Explore the Dockerfile. ...
Step 3: Build your first image. ...
Step 4: Run your container. ...
Step 5: Verify that your container is running.

Step 1: Sign up for a Docker account Start by

creating a Docker ID.

A Docker ID grants you access to Docker Hub repositories and allows you to explore images that are available from the community and verified publishers. You'll also need a Docker ID to share images on Docker Hub.

Step 2: Create your first repository To create a

repository:

1. Sign in to Docker Hub.
2. Select **Create a Repository** on the Docker Hub welcome page.
3. Name it **<your-username>/my-private-repo**.
4. Set the visibility to **Private**.
5. Select **Create**.

You've created your first repository

Step 3: Download and install Docker Desktop

You need to download Docker Desktop to build, push, and pull container images.

1. Download and install Docker Desktop.

2. Sign in to Docker Desktop using the Docker ID you created in step one.

Step 4: Pull and run a container image from Docker Hub

1. In your terminal, run `docker pull hello-world` to pull the image from Docker Hub. You

should see output similar to:

```
$ docker pull hello-world

Using default tag: latest
latest: Pulling from library/hello-world

2db29710123e: Pull complete

Digest:
sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac6
6dc a625

Status: Downloaded newer image for hello-world:latest

docker.io/library/hello-world:latest
```

Run `docker run hello-world` to run the image locally. You should see output similar to:

```
$ docker run hello-world

Hello from Docker!

This message shows that your installation appears to be working
correctly.

 To generate this message, Docker took the following steps:

 1. The Docker client contacted the Docker daemon.

 2. The Docker daemon pulled the "hello-world" image from the
 Docker Hub.

 (amd64)

 3. The Docker daemon created a new container from that image
 which runs the

 executable that produces the output you are currently reading.

 4. The Docker daemon streamed that output to the Docker client,
 which sent

 it to your terminal.


 To try something more ambitious, you can run an Ubuntu container
 with:

 $ docker run -it ubuntu bash

 Share images, automate workflows, and more with a free Docker ID:

 https://hub.docker.com/
```

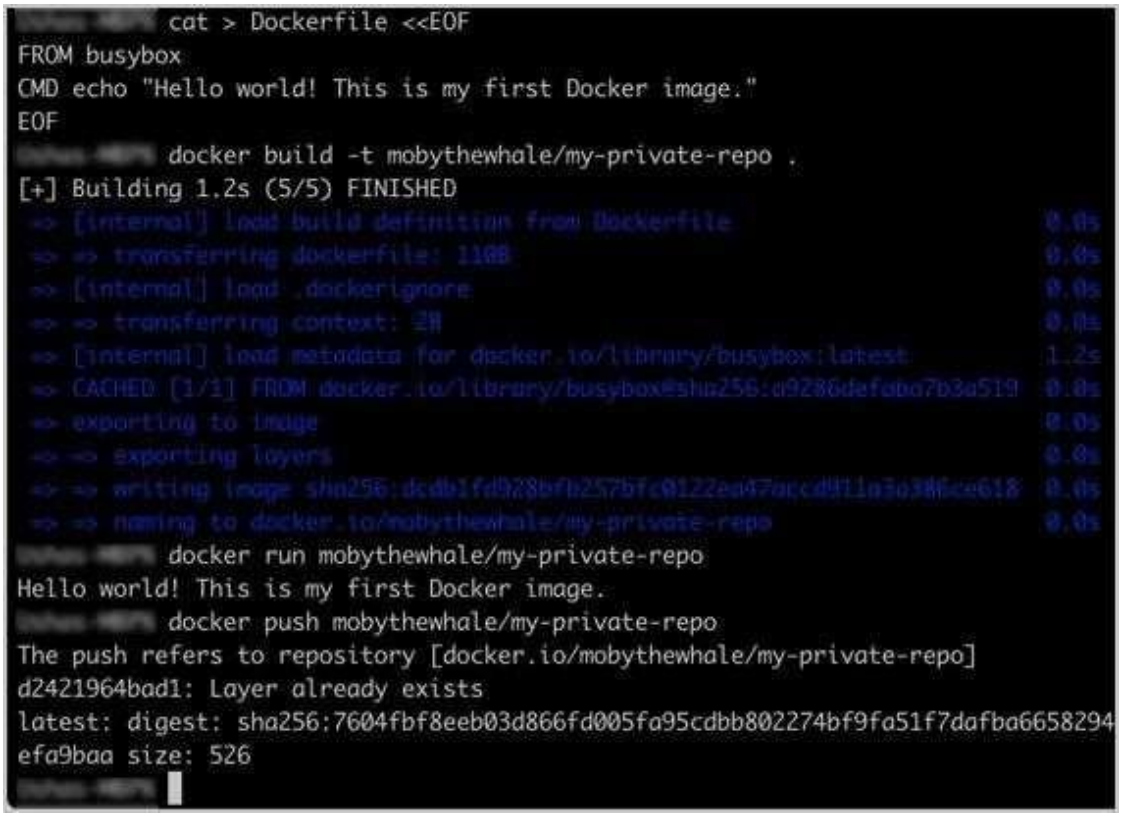
```
For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Step 5: Build and push a container image to Docker Hub from your computer

1. Start by creating a Dockerfile to specify your application as shown below:

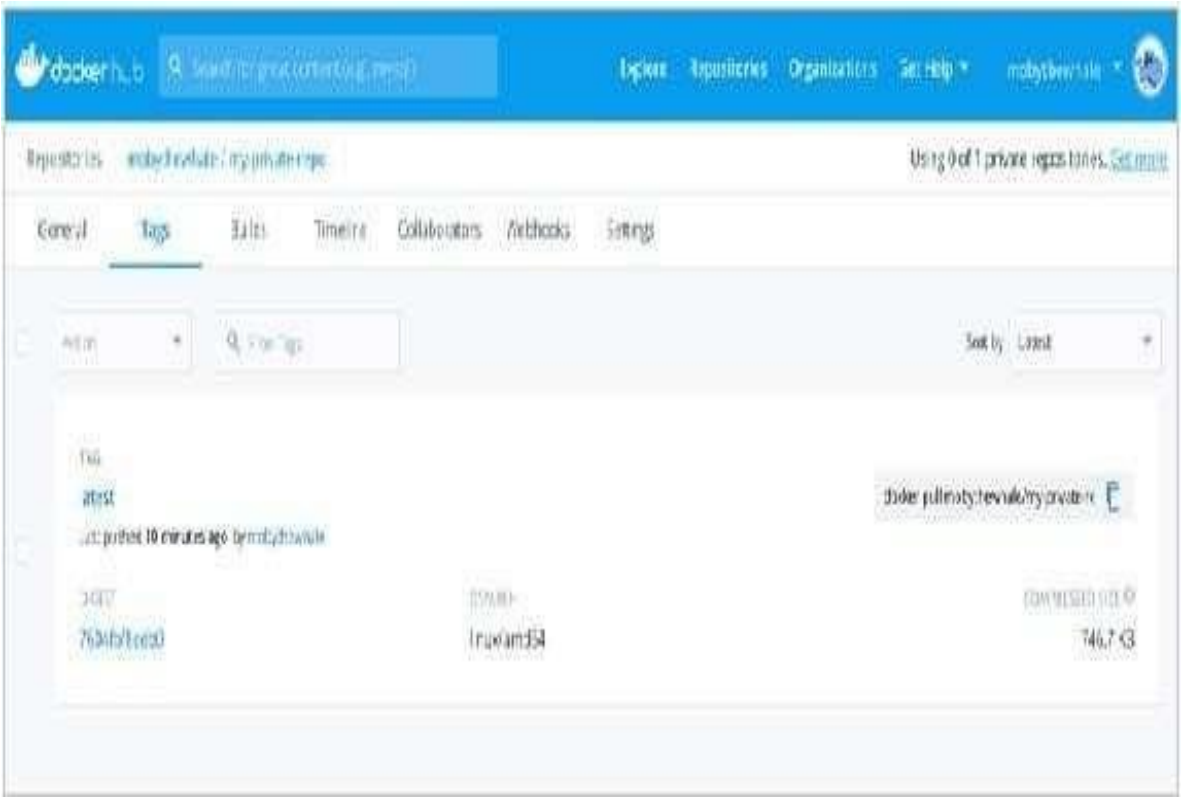
```
# syntax=docker/dockerfile:1
FROM busybox
CMD echo "Hello world! This is my first Docker image."
```

5. Run `docker build -t <your_username>/my-private-repo .` to build your Docker image.
6. Run `docker run <your_username>/my-private-repo` to test your Docker image locally.
7. Run `docker push <your_username>/my-private-repo` to push your Docker image to Docker Hub. You should see output similar to:



> **Note**
>
> You must be signed in to Docker Hub through Docker Desktop or the command line, and you must also name your images correctly, as per the above steps.

8. Your repository in Docker Hub should now display a new `latest` tag under **Tags**:



You've successfully:

- Signed up for a Docker account
- Created your first repository
- Pulled an existing container image from Docker Hub
- Built your own container image on your computer
- Pushed it successfully to Docker Hub

Next steps

- Create an organization to use Docker Hub with your team.
- Automatically build container images from code through builds.
- Explore official & publisher images.
- Upgrade your subscription to push additional private Docker images to Docker Hub.

- Docker Desktop
- Docker Extensions

- Docker Engine
- Docker Build
- Docker Compose
- Docker Hub

Result

Thus the Run a container from Docker Hub is run successfully and output is verified