KGiSL Institute of Technology - KiTE

Coimbatore – 641 035.

(Approved by AICTE & Affiliated to Anna University, Chennai)

NAME : …….…….

REG. No. : ……..

SUBJECT : ……..

COURSE : ……..

KGiSL Institute of Technology - KiTE

Coimbatore – 641 035.

(Approved by AICTE & Affiliated to Anna University, Chennai)

CS8711 – Cloud Computing Laboratory

NAME : CLASS :
UNIVERSITY REG No :

Certified that, this is a bonafide record of work done by …… of **Computer Science and Engineering** branch in **Cloud Computing Laboratory**, during **Seventh** semester of academic year 2022-2023.

Faculty Incharge **Head of the Department**

Submitted during Anna University Practical Examination held on …….……. at KGiSL Institute of Technology, Coimbatore – 641 035.

Internal Examiner **External Examiner**

INDEX

S. No	DATE	LIST OF THE EXPERIMENTS	PAGE NO	MARKS	SIGNATURE
1		Install Virtual box with different flavours of Linux or Windows OS on top of Windows 7 or 8			
2		Install a C compiler in the virtual machine created using virtual box and execute a simple program			
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			

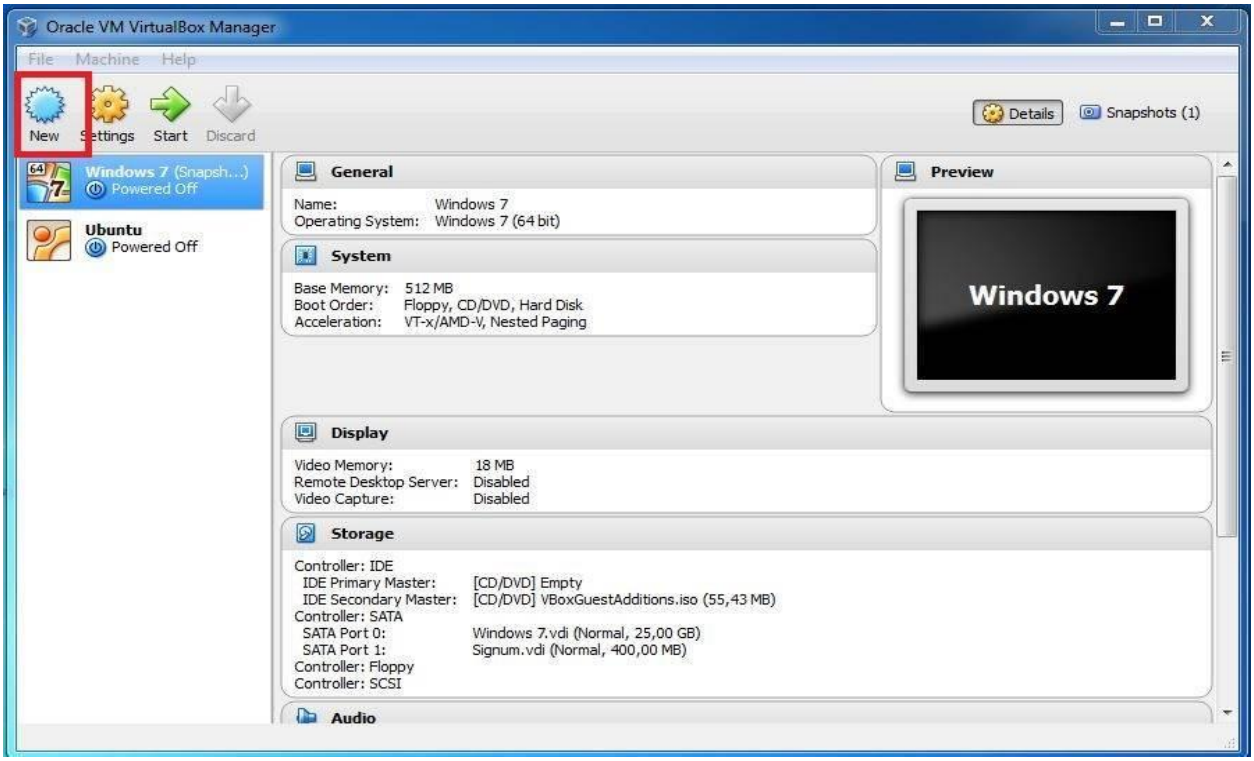
Ex. No : 1 Date :	INSTALLATION OF UBUNTU

AIM:

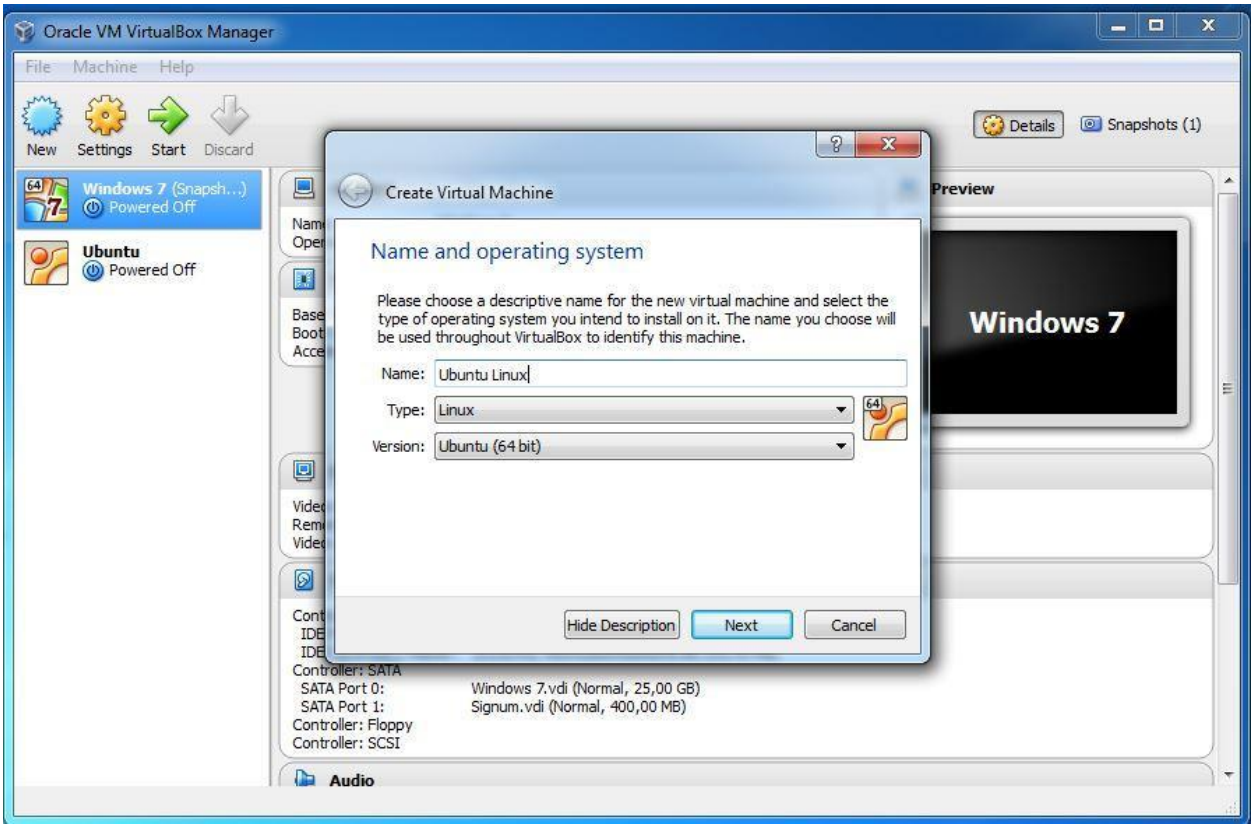
To Install Virtual Box/VMware Workstation with different flavours of Linux or Windows OS on top of windows7 or 8

PROCEDURE:

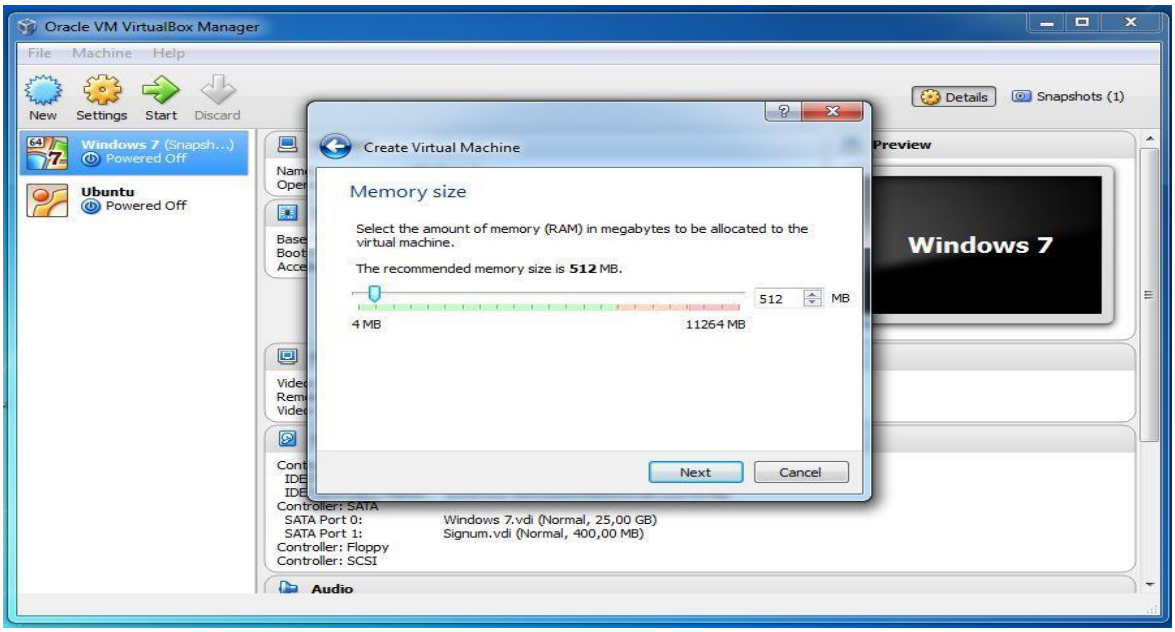
Install Ubuntu as the guest operating system from an ISO image file**.** To create a new virtual machine click on the **New** button in the **Virtual Box Manager GUI** window.



Give the VM a name and make sure that the operating system and version are **Linux** and **Ubuntu**, respectively. Click **Next**.



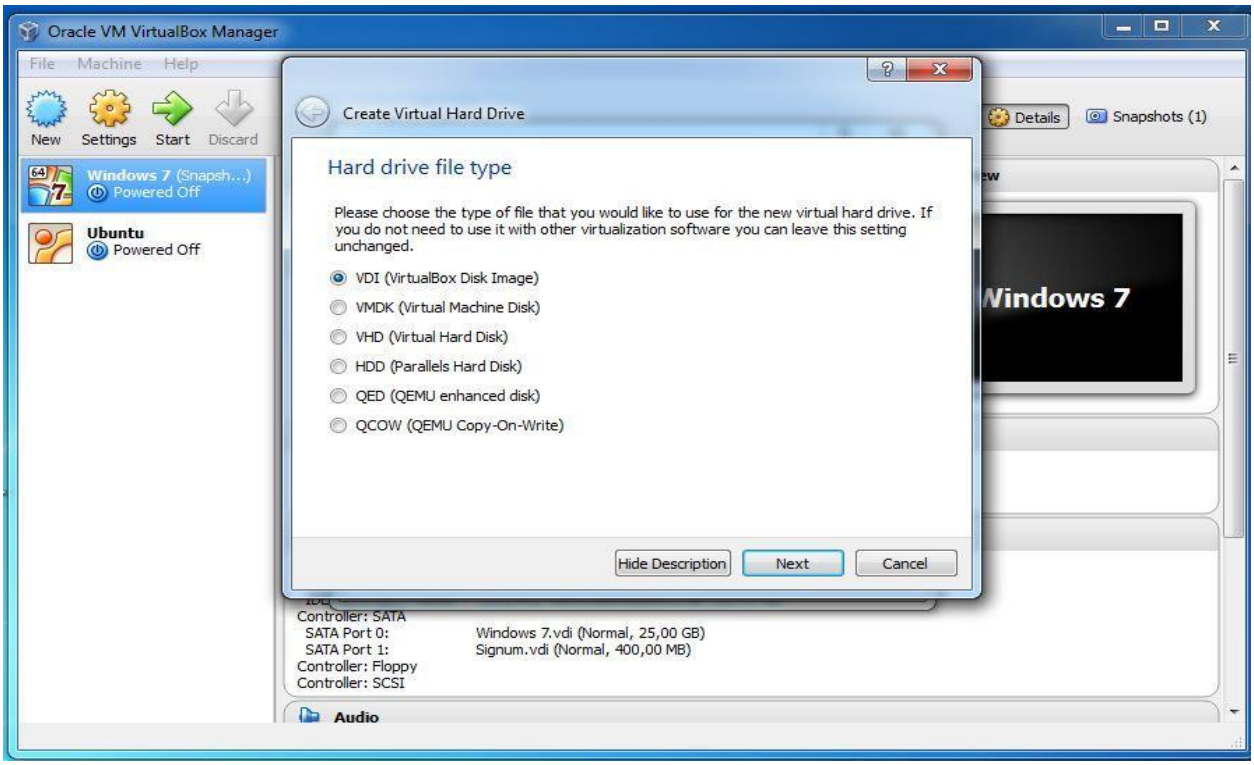
Next, select the amount of memory (RAM) to be allocated to the virtual machine:



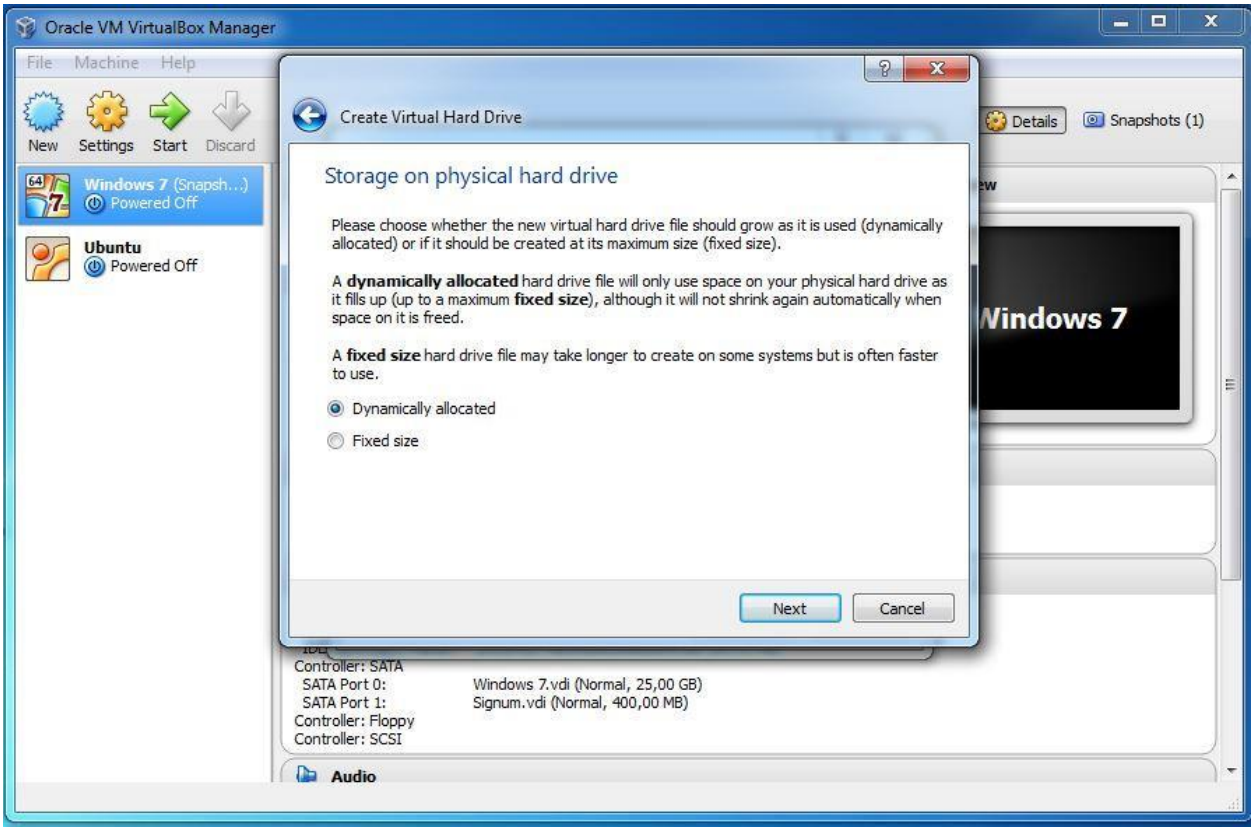
Next, you must specify a virtual hard disk for your VM. There are multiple ways in which Virtual Box can provide hard disk space to a VM, but the most common way is to use a large image file on your real hard disk, whose contents Virtual Box presents to your VM as if it were a

complete hard disk. This file represents an entire hard disk then, so you can even copy it to another host and use it with another Virtual Box installation:

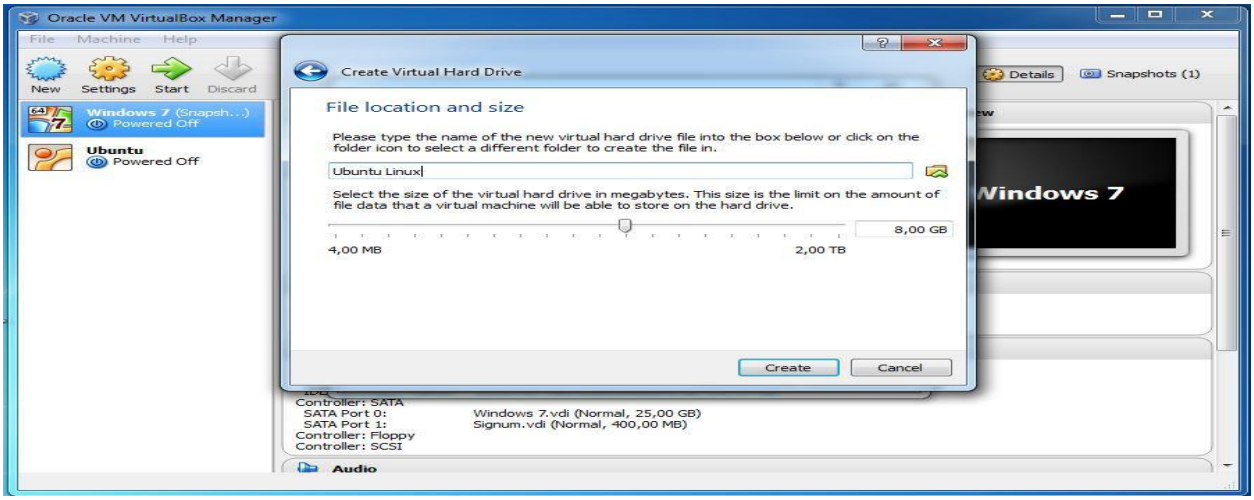
Select **VDI** hard drive type and click **Next**:



In this example we will be using dynamic disks, but fixed disks will work just as well:

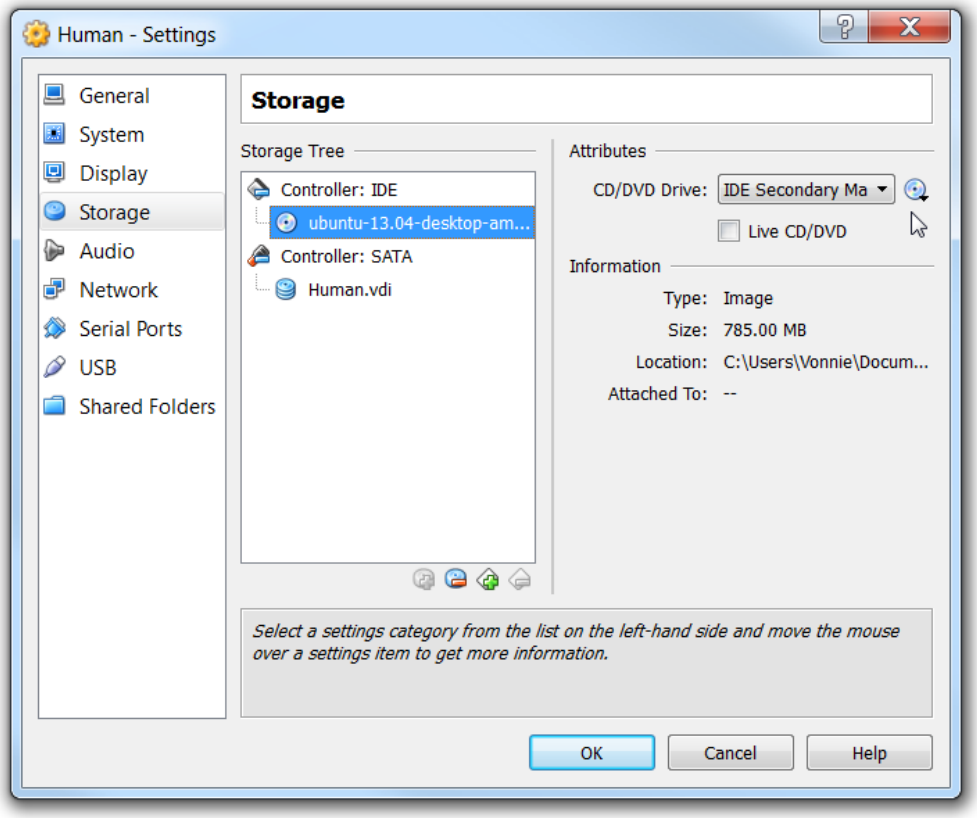


Next, type the name of the new virtual hard drive file or select a different folder to create the file in and click **Create**:



When the VM creation thingy goes away, press **Ctrl** + **s** to open your VM settings.

Next, click **Storage** in the left pane and click the CD icon in the right pane to browse for the Ubuntu ISO you downloaded in the first steps.

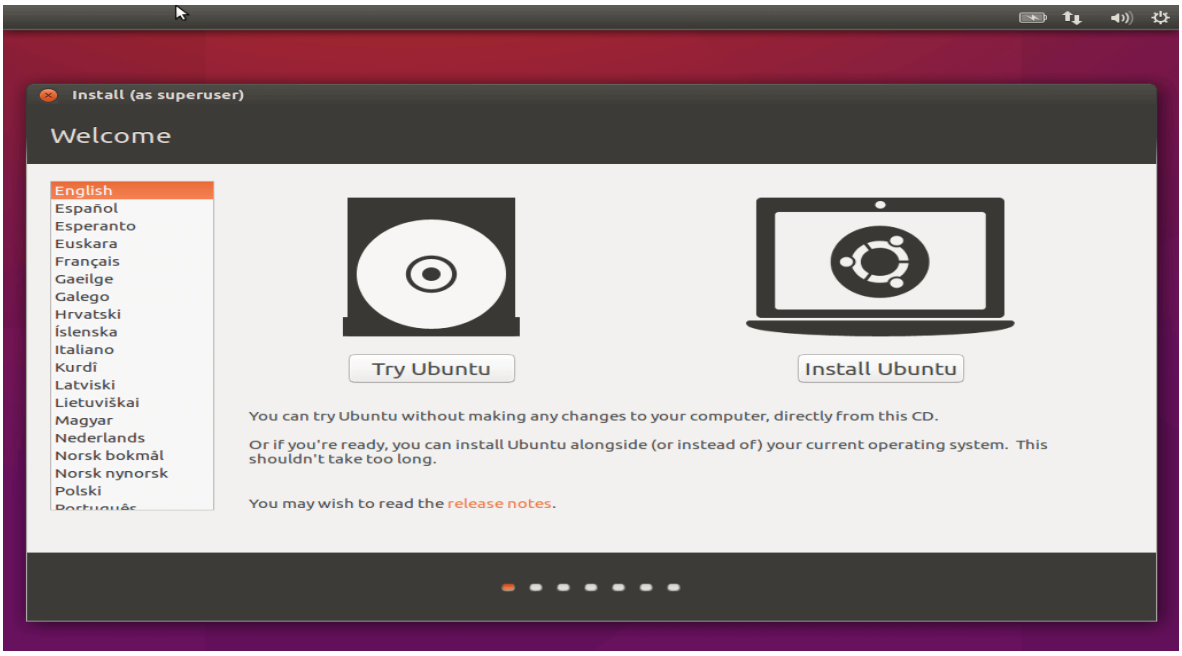


Click **OK** to close Settings and then start your new Ubuntu VM by clicking the **green Start arrow in the main Virtual Box window.**

After a few seconds the Ubuntu welcome window happily shoots on the screen ready to install Ubuntu.

Ubuntu 16.04 Installation Guide

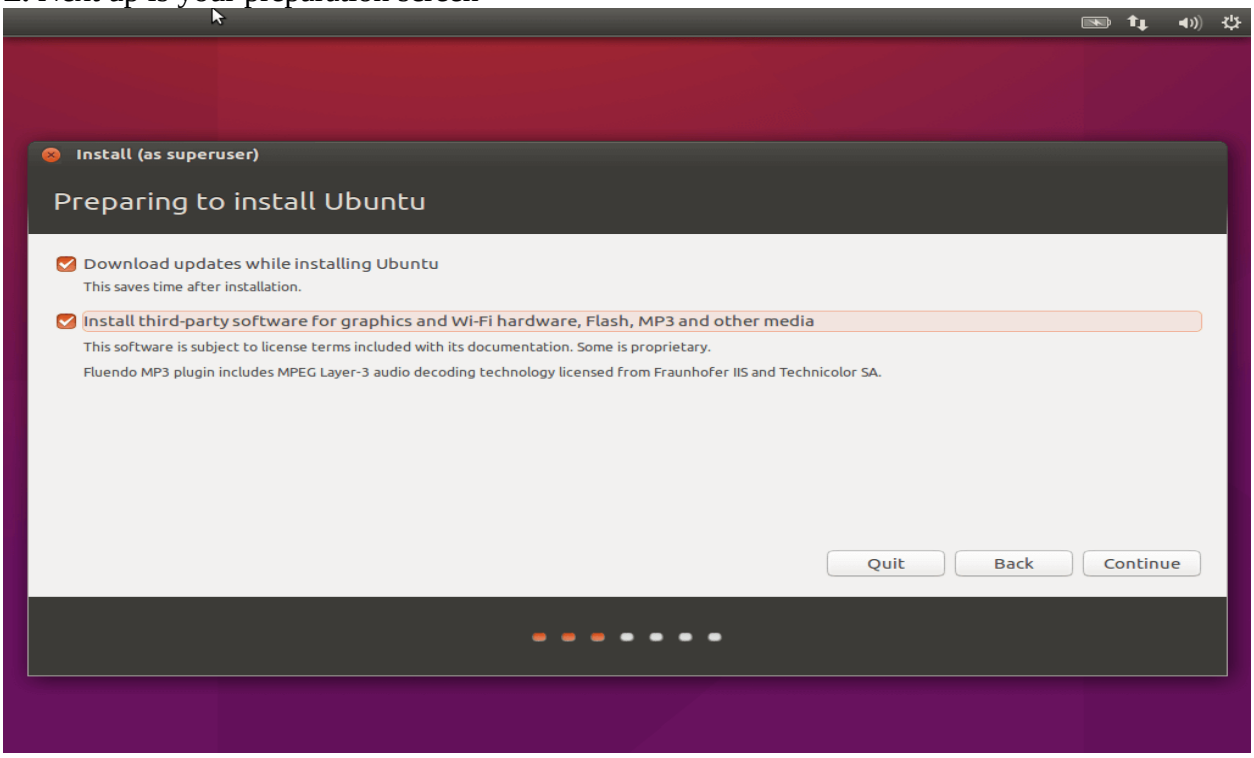
1. Proceed by clicking the **install Ubuntu** button.



Ubuntu 16.04 Installation

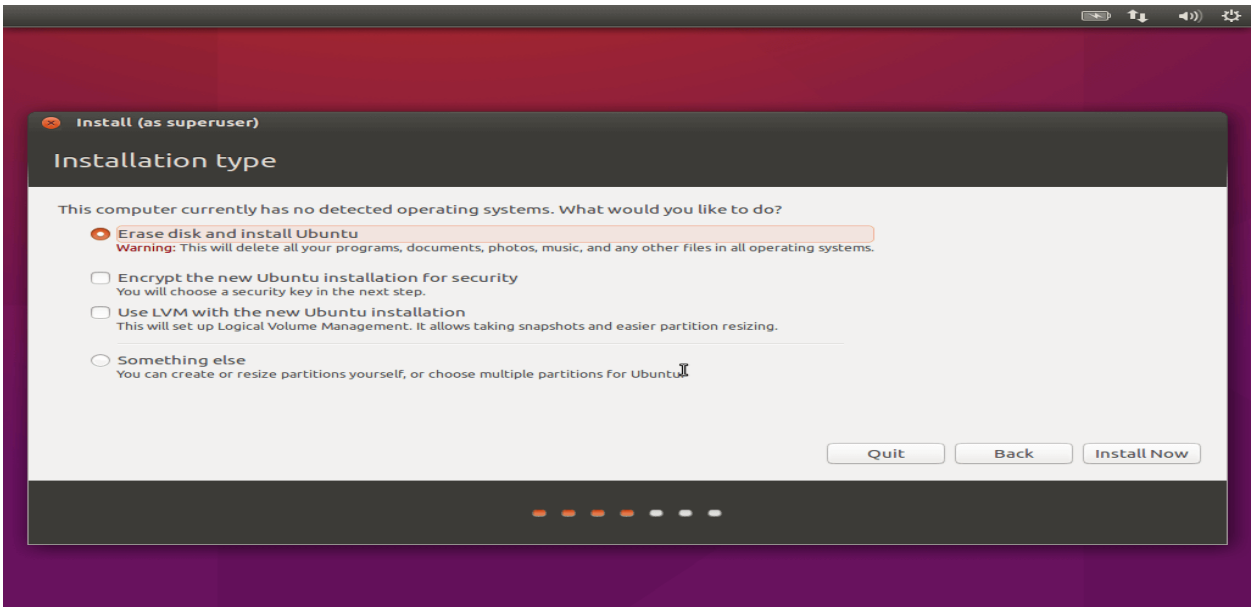
Choose the **language** as needed; and this one will be the default (once installed) throughout the system.

2. Next up is your preparation screen



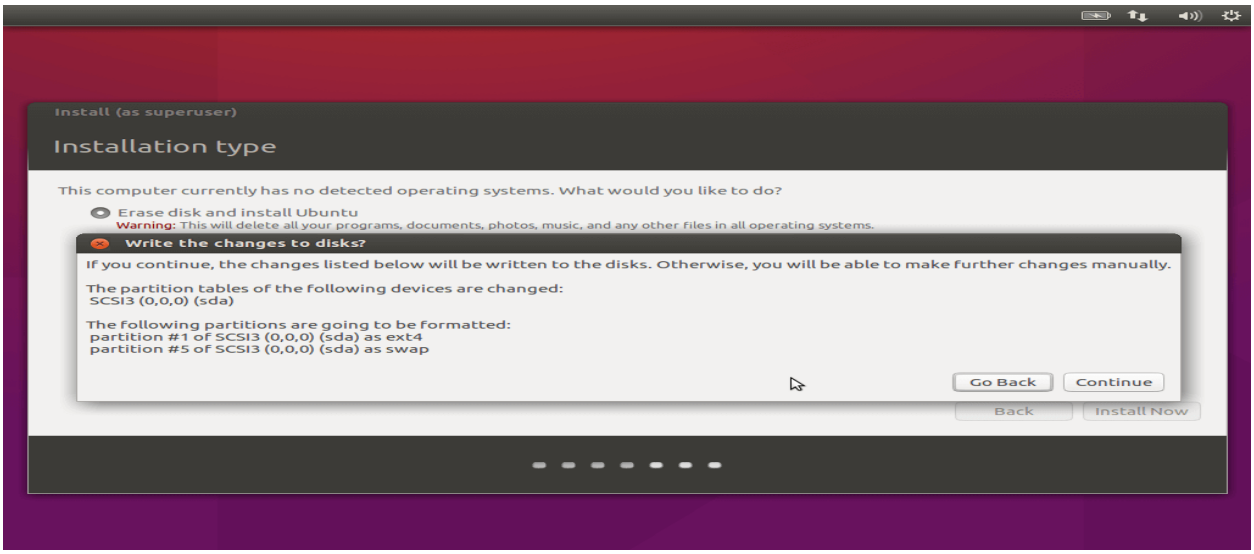
Preparing to Install Ubuntu 16.04

3. At this point, choose the installation type and the first screenshot is an automated process, even if you have an operating system already installed, the installer will auto-detect it and allow you partition the drive in the next screen with simple sliders that will auto-allocate your assigned space for the Ubuntu partition.



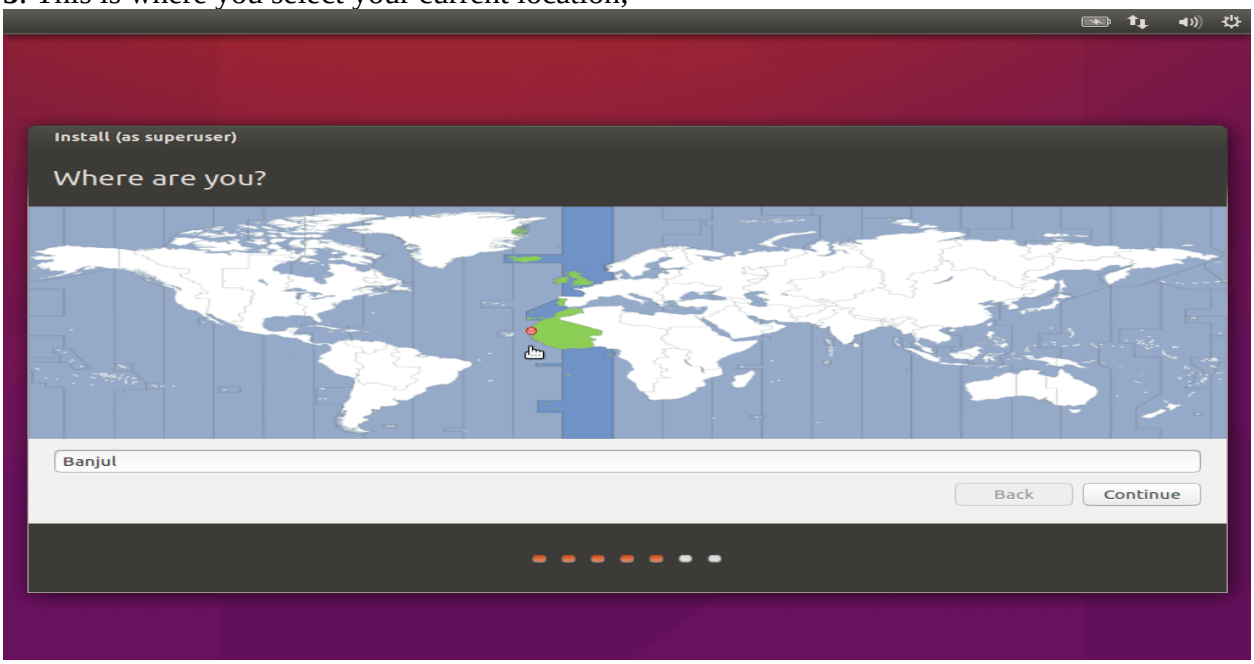
Select Ubuntu 16.04 Installation Type

4. Prompt to confirm that you want the changes to be made to your internal drive; click continue to move onto the next screen.



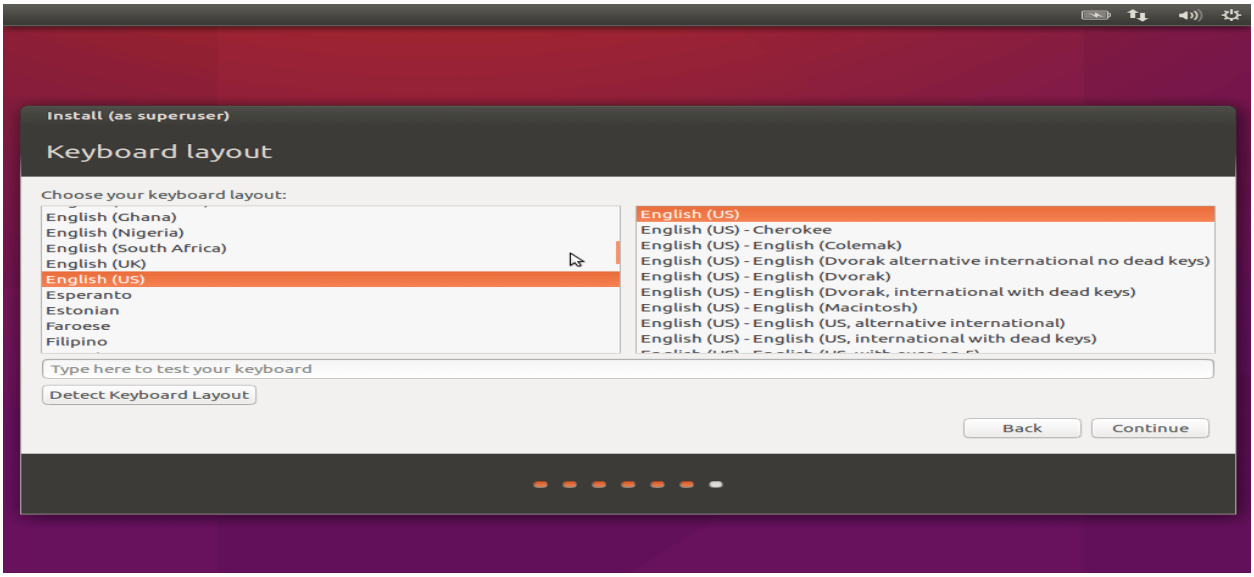
Write Changes to Disk

5. This is where you select your current location;



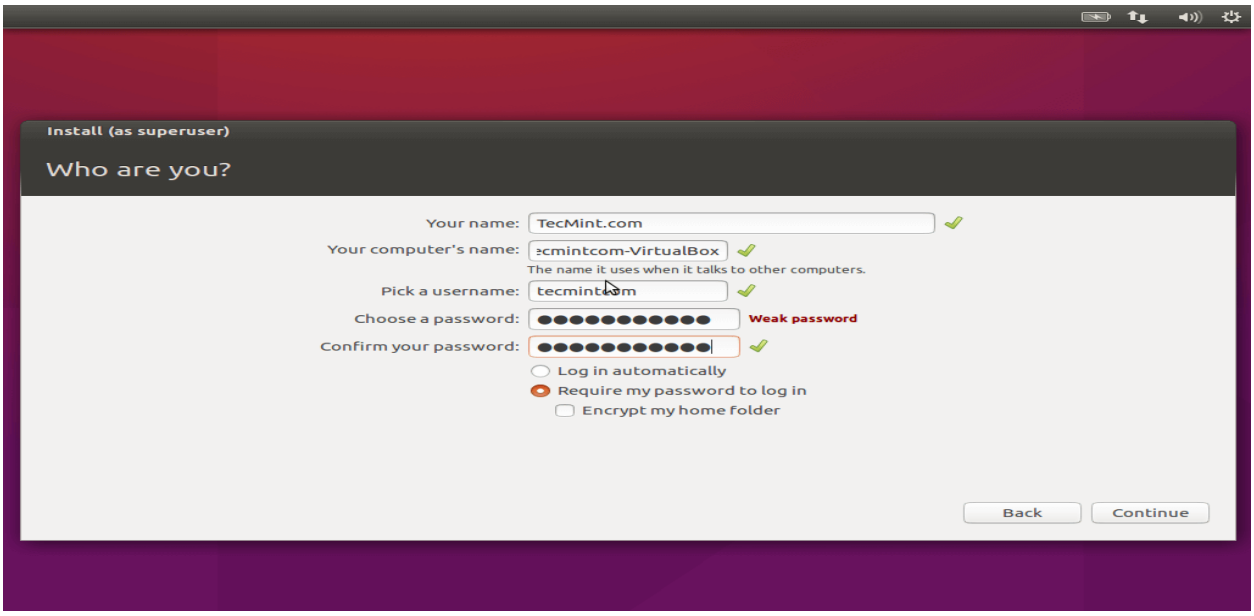
Select Current Location

6. Configure as needed — depending on your type of keyboard and default input language.



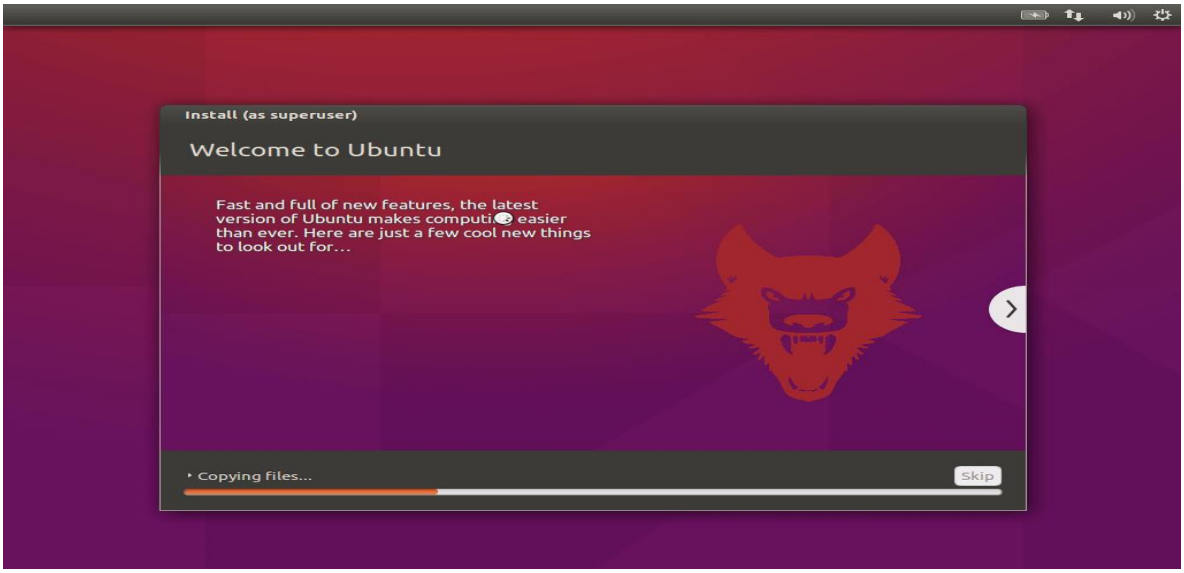
Select Keyboard Layout

7. This is where you enter your user details and clicks continue to proceed to the next screen**.**



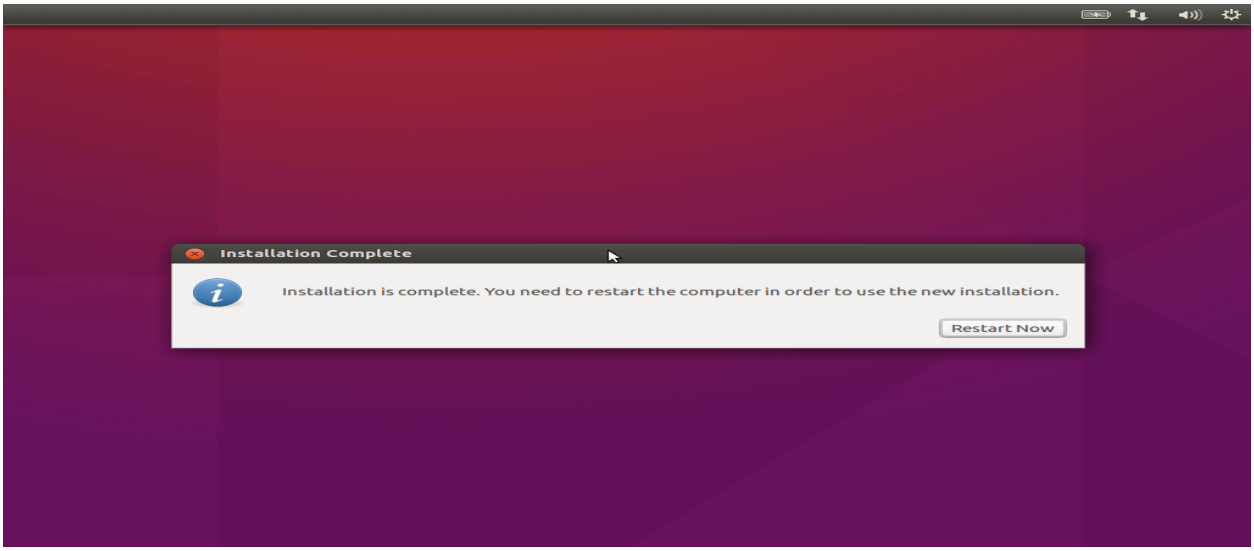
Create User Account

8. Right up next, is the beginning of installation which (depending on your PC hardware), can take a long or short time.



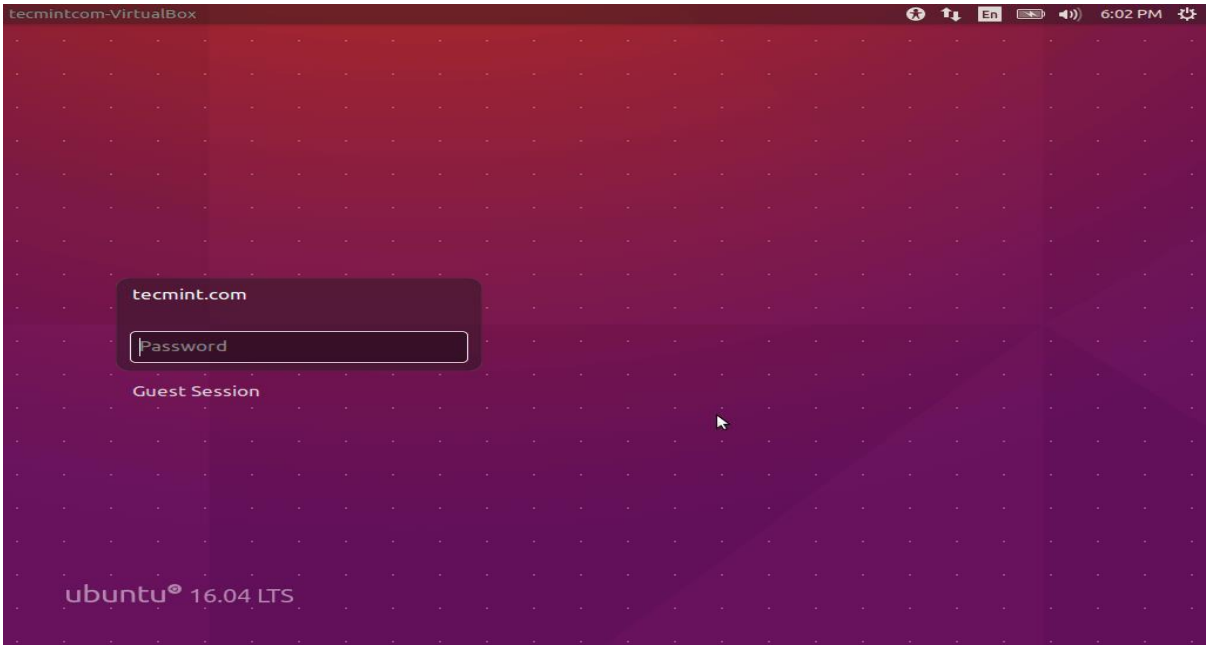
Ubuntu 16.04 Installation Process

9. At this point, the installation is complete and now, you may restart your PC.



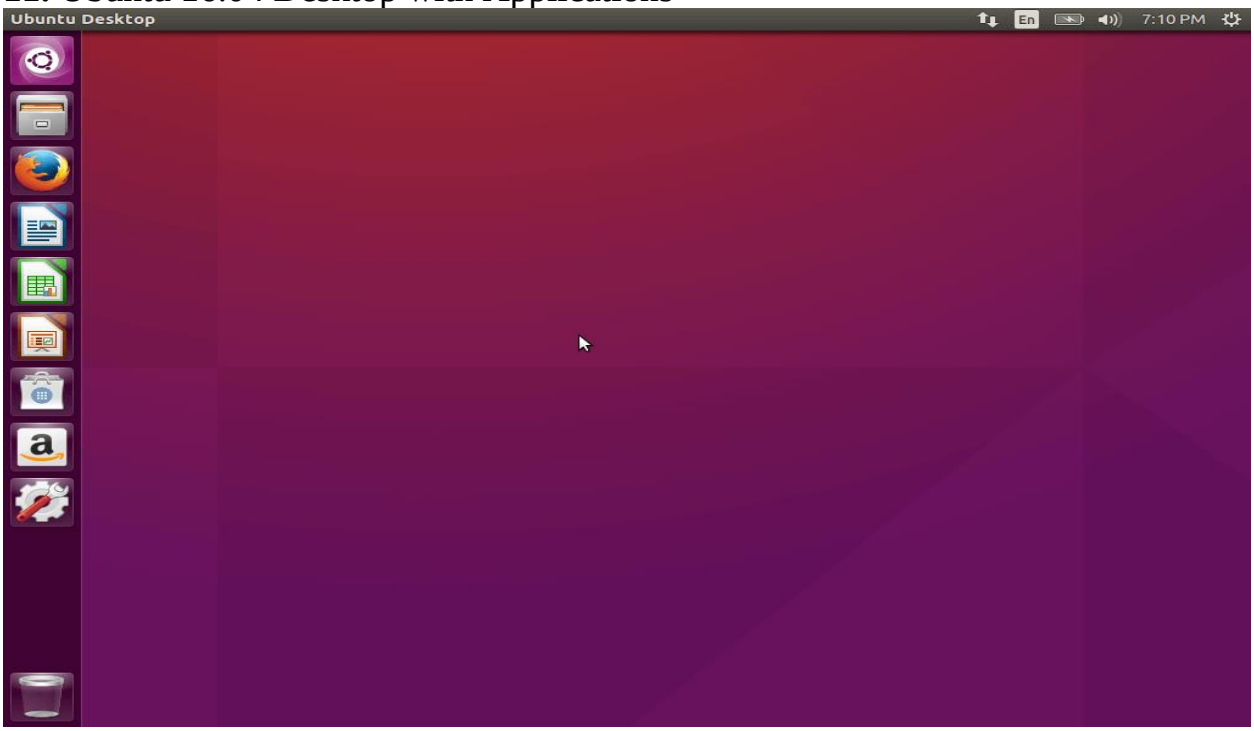
Ubuntu 16.04 Installation Complete

10. Once you've restarted, you are now greeted with the login screen where you input your password.



Ubuntu 16.04 Login Screen

11. Ubuntu 16.04 Desktop with Applications



Ubuntu 16.04 Desktop with Applications

RESULT:

Thus, the Virtual Box/VMware Workstation with different flavours of Linux or Windows OS has been successfully installed.

Ex. No : 2 **Date :**	: **INSTALL THE C COMPILER IN VIRTUAL MACHINE**

AIM:

To install C compiler in the virtual machine created using virtual box and execute a simple program.

PROCEDURE:

Install the compiler:

administrator@administrator-Virtual Box:~ sudo apt-get install gcc

Open the editor and write the c code:

administrator@administrator-Virtual Box:~ vi cse.c

Program:

```
#include <stdio.h>

int main() {

   int year;

   year = 2016;

   if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))

      printf("%d is a leap year", year);

   else

      printf("%d is not a leap year", year);

   return 0;

}
```

Compile the program:

administrator@administrator-Virtual Box:~gcc cse.c

Execute the program:

administrator@administrator-Virtual Box:~./a.out

OUTPUT:

2016 is a leap year

RESULT:

Thus, the C compiler in the virtual machine has been installed and a simple C program has been executed.

Ex. No : 3 Date :	INSTALLING AND RUNNING THE GOOGLE APP ENGINE ON WINDOWS

AIM:

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

PROCEDURE:

You can download the Google App Engine SDK by going to: http://code.google.com/appengine/downloads.html and download the appropriate install package.

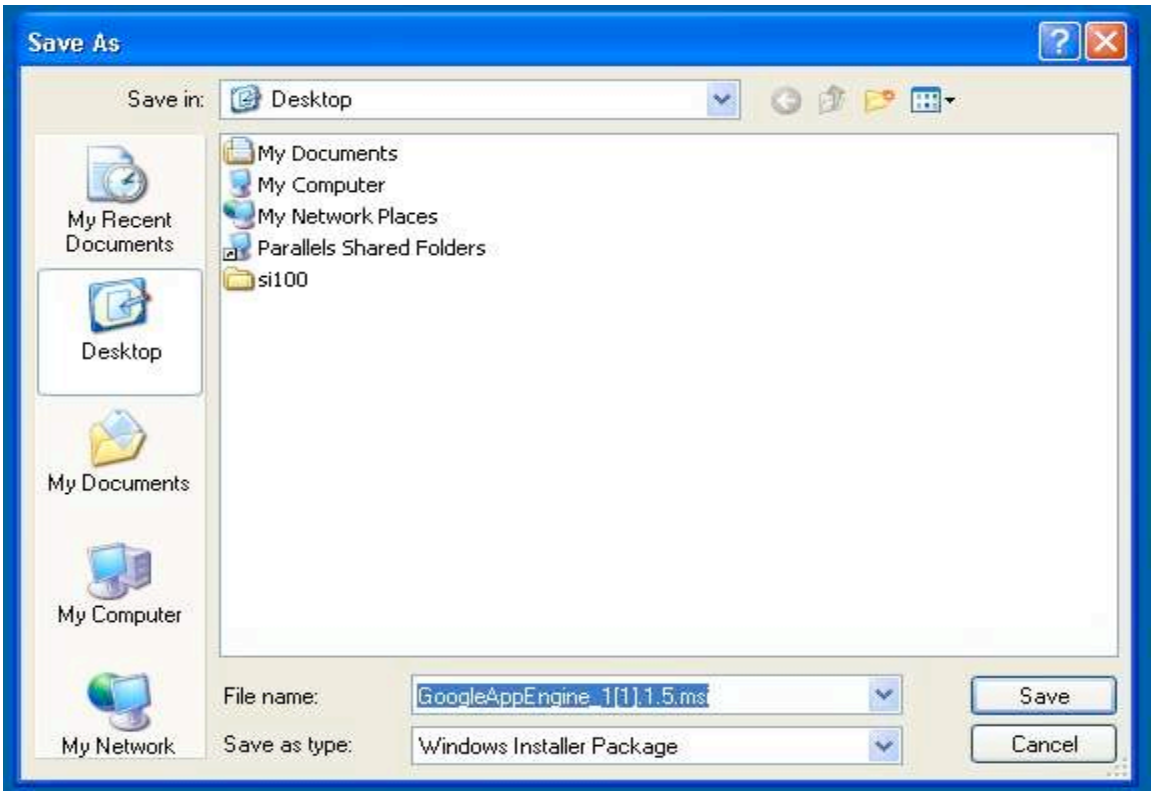
Download the Google App Engine SDK

Before downloading, please read the Terms that govern your use of the App Engine SDK.

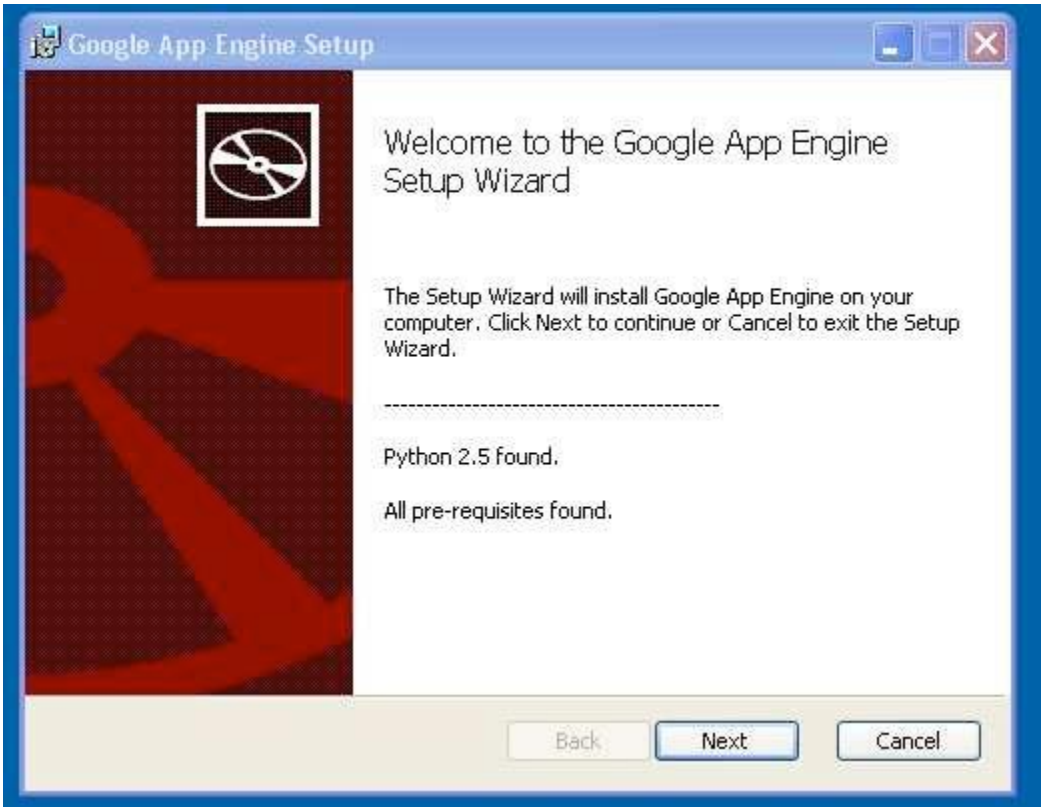
Please note: The App Engine SDK is under **active development**, please keep this in mind as you explore its capabilities. See the SDK Release Notes for the information on the most recent changes to the App Engine SDK. If you discover any issues, please feel free to notify us via our Issue Tracker.

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

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



Double Click on the GoogleApplicationEngine installer.



Click through the installation wizard, and it should install the App Engine. If you do not have Python 2.5, it will install Python 2.5 as well. Once the install is complete you can discard the downloaded installer

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

Make a folder for your Google App Engine applications.

I am going to make the Folder called "apps" – the path to this folder is:

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

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

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

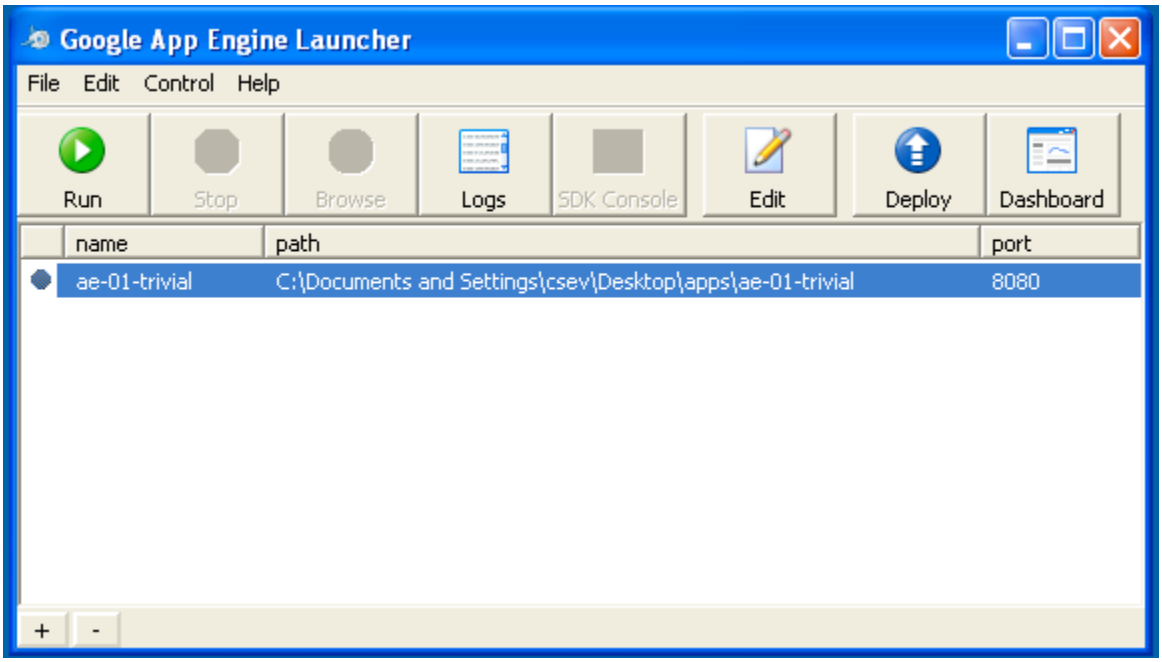
Using a text editor such as JEdit (www.jedit.org), create a file called app.yaml in the ae--01--trivial folder with the following contents:

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

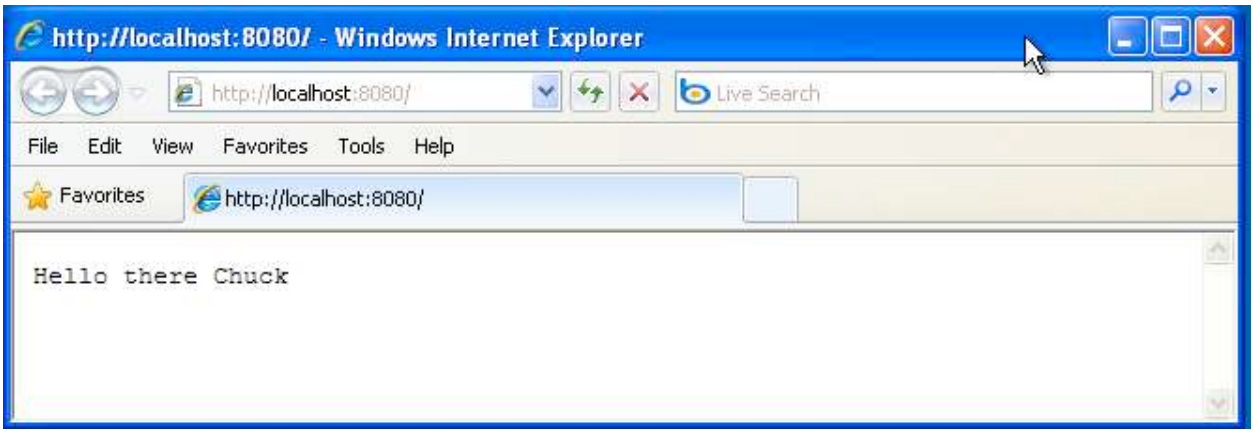
Then create a file in the ae--01--trivial folder called index.py with three lines in it:

```
print 'Content-Type: text/plain'
print ' '
print 'Hello there Chuck'
```

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



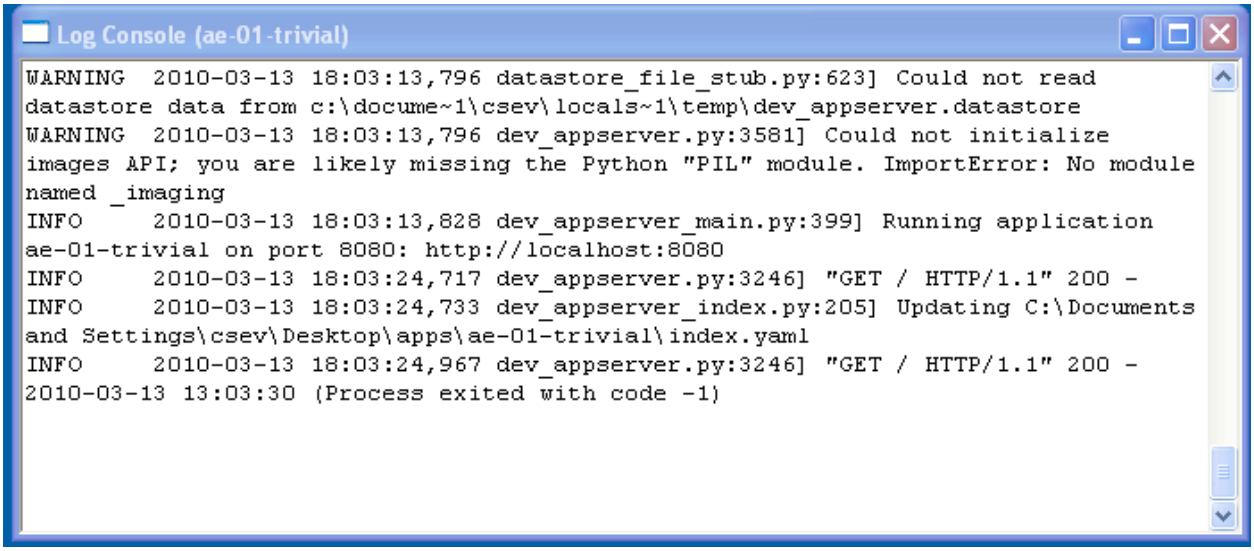
Once you have selected your application and press Run. After a few moments your application will start and the launcher will show a little green icon next to your application. Then press Browse to open a browser pointing at your application which is running at http://localhost:8080/ Paste http://localhost:8080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



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

Watching the Log

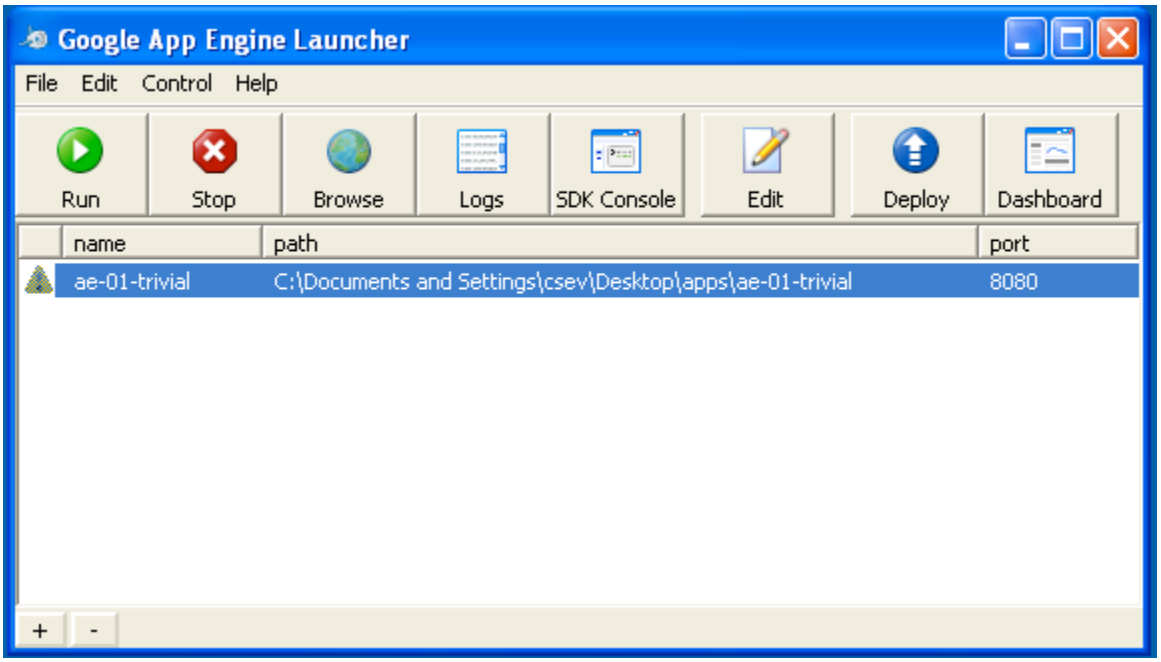
You can watch the internal log of the actions that the web server is performing when you are interacting with your application in the browser. Select your application in the Launcher and press the Logs button to bring up a log window:



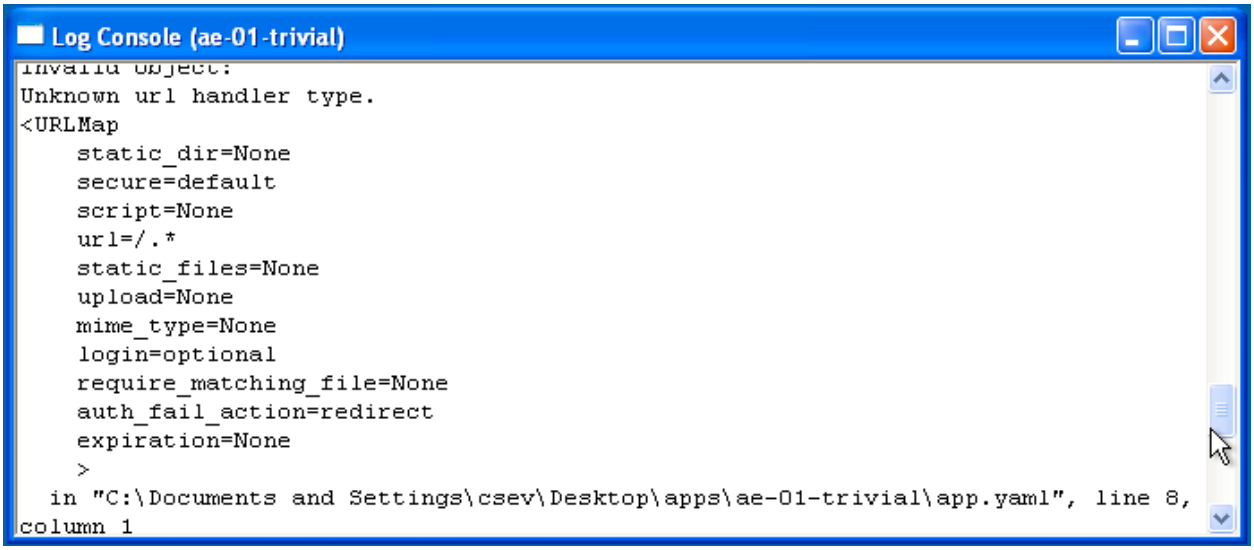
Each time you press Refresh in your browser – you can see it retrieving the output with a GET request.

Dealing With Errors

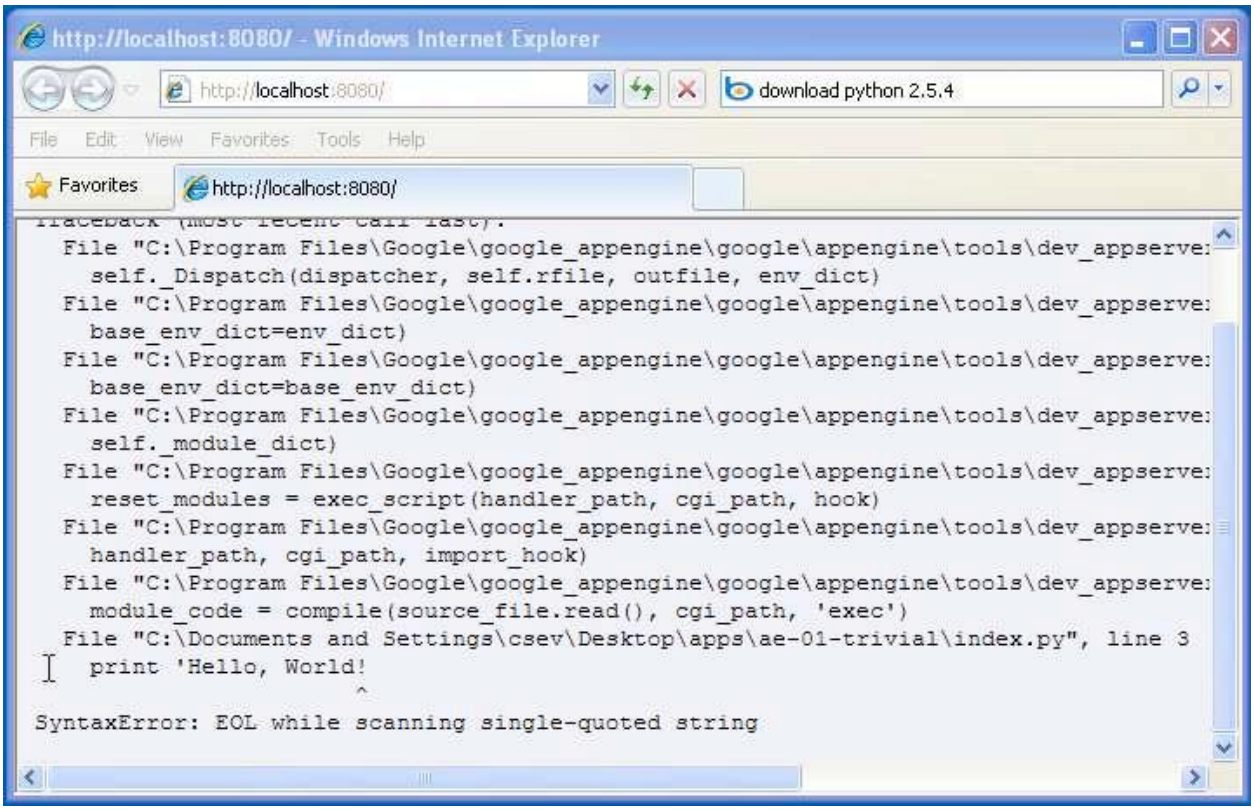
With two files to edit, there are two general categories of errors that you may encounter. If you make a mistake on the app.yaml file, the App Engine will not start and your launcher will show a yellow icon near your application:



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



In this instance – the mistake is mis---indenting the last line in the app.yaml (line 8).
If you make a syntax error in the index.py file, a Python trace back error will appear in your browser.



The error you need to see is likely to be the last few lines of the output – in this case
I made a Python syntax error on line one of our one---line application.
Reference: http://en.wikipedia.org/wiki/Stack_trace
When you make a mistake in the app.yaml file – you must the fix the mistake and
attempt to start the application again.
7
If you make a mistake in a file like index.py, you can simply fix the file and press
refresh in your browser – there is no need to restart the server.

Shutting Down the Server

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

RESULT:

Thus, the installation of Google App Engine has been successfully done.

Ex. No : 4 Date :	GAE LAUNCHER TO LAUNCH THE WEB APPLICATIONS

AIM:

To use GAE launcher to launch the web applications.

PROCEDURE:

Step 1: Download the basic

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

1. Download Python 2.7

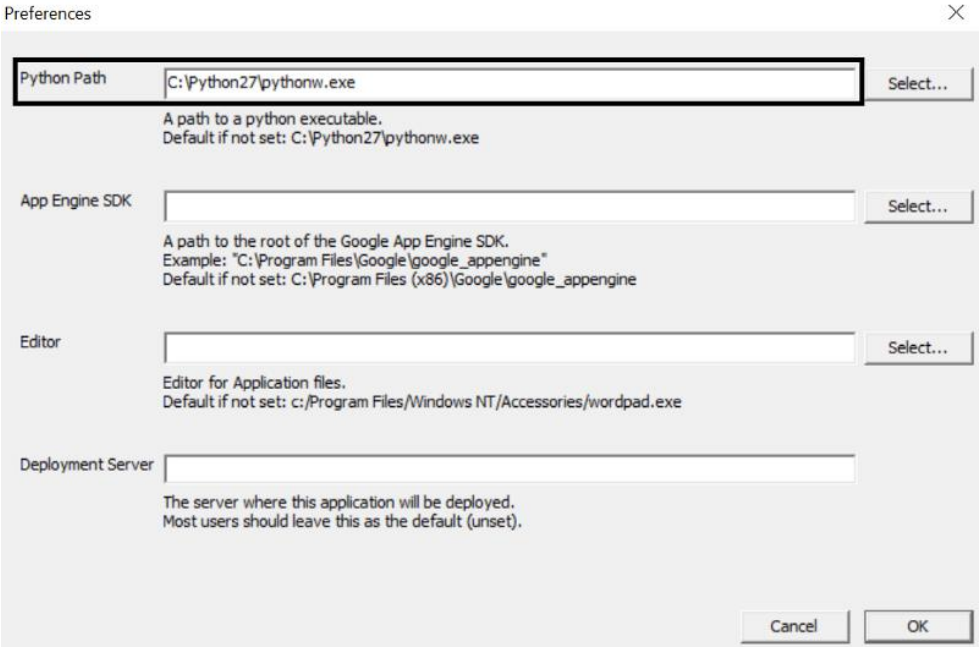
As of when this article was written, the Google App Engine standard environment supports Python only upto version 2.7. However, it is only a matter of time before support for Python 3.x is added. You can check the App Engine docs for the latest info.

2. Download Google Cloud SDK

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

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

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



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:

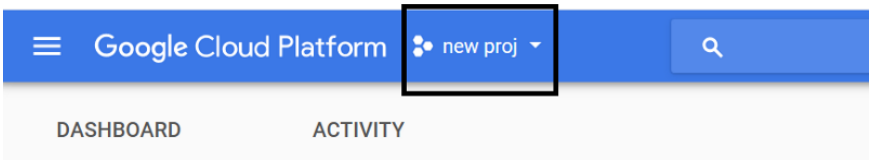
1. Go to the Google Cloud landing page
2. Follow the sign-up process and go to your App Engine dashboard

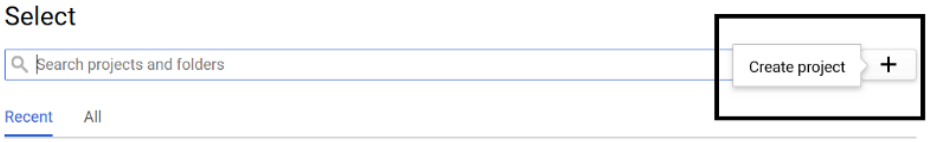
Most of the hard work is complete after a successful sign-up.

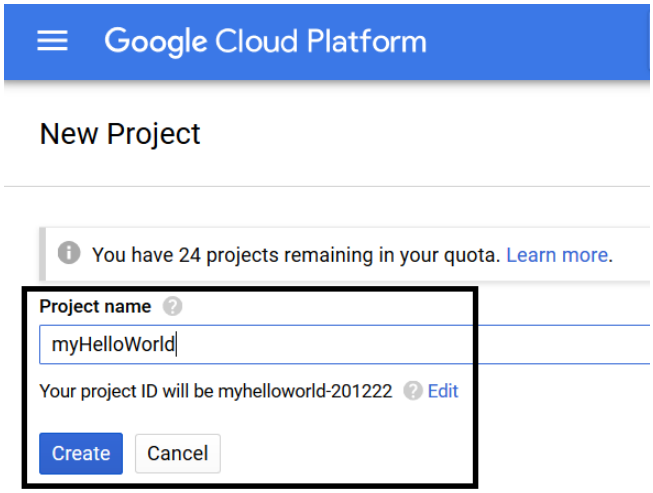
Step 3: Create a new project

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.

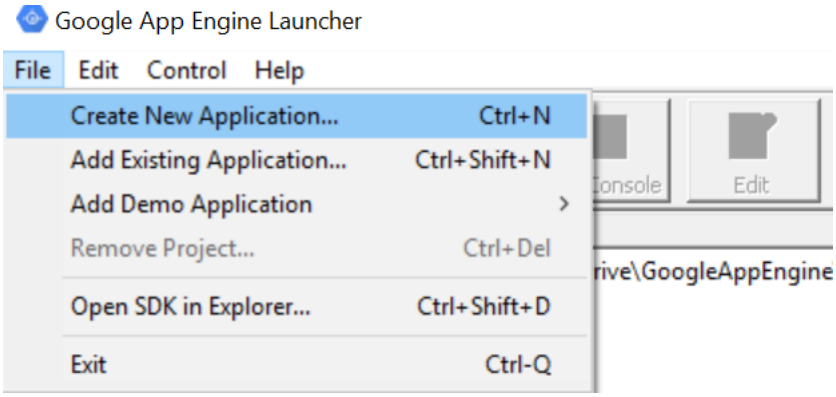


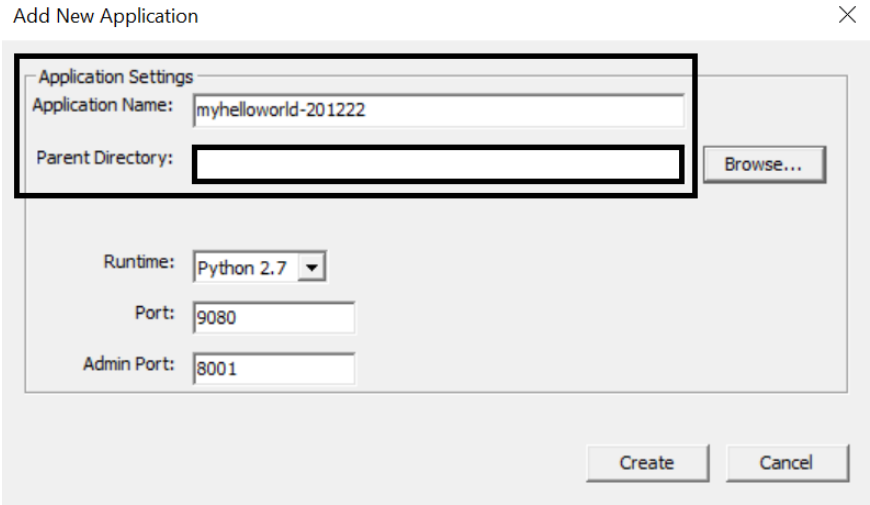




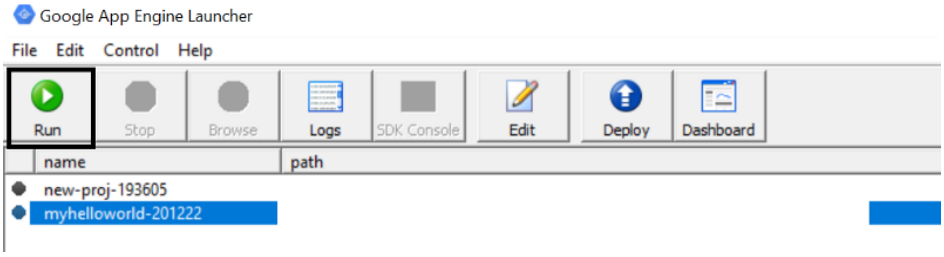
Step 4: Fork the app to develop it locally

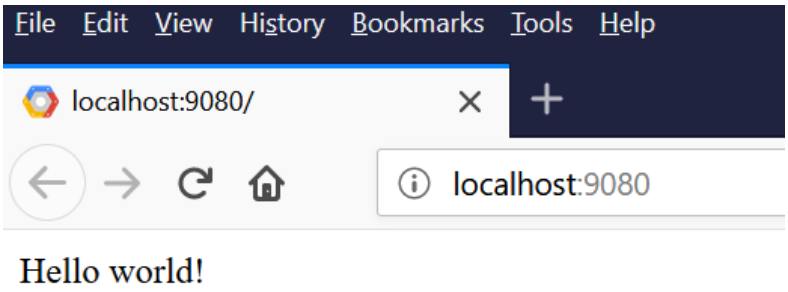
The next step in the process is to fork the app on your local machine. This will allow you to make changes to the app locally and deploy it whenever you wish to.
Go to Google App Engine launcher and create a new application.





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.





RESULT:

Thus, the GAE launcher to launch the web applications has been done and verified successfully.

Ex. No : 5 Date :	CLOUDSIM

AIM:

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

PROCEDURE:

1. Download CloudSim installable files from *https://code.google.com/p/cloudsim/downloads/list and unzip*
2. Open Eclipse
3. Create a new Java Project: File -> New
4. Import an unpacked CloudSim project into the new Java Project
5. The first step is to initialise the CloudSim package by initialising the CloudSim library, as follows:
CloudSim.init(num_user, calendar, trace_flag)
6. Data centres are the resource providers in CloudSim; hence, creation of data centres is a second step. To create Datacenter, you need the DatacenterCharacteristics object that stores the properties of a data centre such as architecture, OS, list of machines, allocation policy that covers the time or spaceshared, the time zone and its price:

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

7. The third step is to create a broker:

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

Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared())
9. Submit the VM list to the broker:

broker.submitVmList(vmlist)
10. Create a cloudlet with length, file size, output size, and utilisation model:

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

11. Submit the cloudlet list to the broker:

broker.submitCloudletList(cloudletList)

12. Start the simulation:

CloudSim.startSimulation()

Output from the Existing Example:

```
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>null
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: 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
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim
entities for shutting down. Datacenter_0 is
shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.


========== OUTPUT ==========
Cloudlet ID STATUS Data center ID VM ID Time Start
Time Finish Time 0 SUCCESS 2 0 400 0.1 400.1
*****Datacenter: Datacenter_0*****
User id Debt
 3 35.6

CloudSimExample1 finished!
```

1. You can copy few (or more) lines with ***copy & paste*** mechanism. For this you need to share clipboard between host OS and guest OS, installing **Guest Addition** on both the virtual machines (probably setting *bidirectional* and restarting them). You *copy* from *guest OS* in the clipboard that is shared with the *host OS*. Then you *paste* from the *host OS* to the second *guest OS*.
2. You can enable **drag and drop** too with the same method (Click on the machine, settings, general, advanced, drag and drop: set

to *bidirectional*)

3. You can have **common *Shared Folders*** on both virtual machines and use one of the directory shared as buffer to copy.
   Installing **Guest Additions** you have the possibility to set Shared Folders too. As you put a file in a shared folder from *host OS* or from *guest OS*, is immediately visible to the other. (Keep in mind that can arise some problems for date/time of the files when there are different clock settings on the different virtual machines).
   If you use the same folder shared on more machines you can exchange files directly copying them in this folder.
4. You can use **usual method to copy files between 2 different computer** with client-server application. (e.g. scp with sshd active for linux, winscp... you can get some info about SSH servers e.g. here)
   You need an active server (sshd) on the receiving machine and a client on the sending machine. Of course you need to have the authorization setted (via password or, better, via an automatic authentication method).

RESULT:

Thus, the cloud scenario using CloudSim has been executed successfully

Ex. No : 6 Date :	TRANSFER FILES FROM ONE VIRTUAL MACHINE TO ANOTHER VIRTUAL MACHINE

AIM:

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

PROCEDURE:

START THE HADOOP:

hduser@administrator-Virtual Box:~$ cd /usr/local/hadoop/sbin

hduser @administrator-Virtual Box:/usr/local/hadoop/sbin$ ls

hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$ start-all.sh

hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$ jps

CREATE DIRECTORY:

hduser@administrator-Virtual Box:~$ /usr/local/hadoop/bin/hdfs dfs -mkdir /sampledir

MOVE THE DATA TO INPUT DIRECTORY:

hduser@administrator-Virtual Box:~$ /usr/local/hadoop/bin/hdfs dfs -put /home/administrator/Desktop/file1.txt /sampledir

DOWNLOAD THE FUSE:

hduser@administrator-Virtual Box:~$ wget

http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-repository_1.0_all.deb

INSTALL THE FUSE:

hduser@administrator-Virtual Box:~$ sudo dpkg -i cdh5 repository_1.0_all.deb

hduser@administrator-Virtual Box:~$ sudo apt-get update

hduser@administrator-Virtual Box:~$ sudo apt-get install hadoop-hdfs-fuse

CREATE MOUNT POINT:

hduser@administrator-Virtual Box:~$ sudo mkdir -p /opt/sampletest

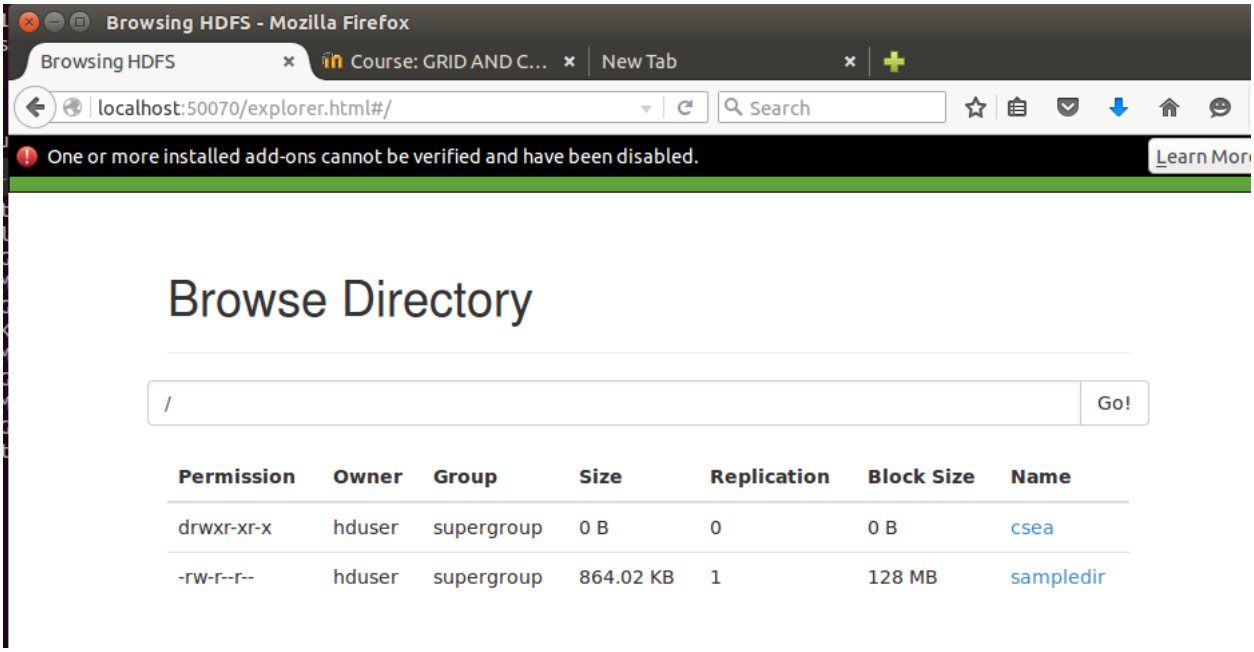
MOUNTING THE FILE SYSTEM ON MOUNT POINT:

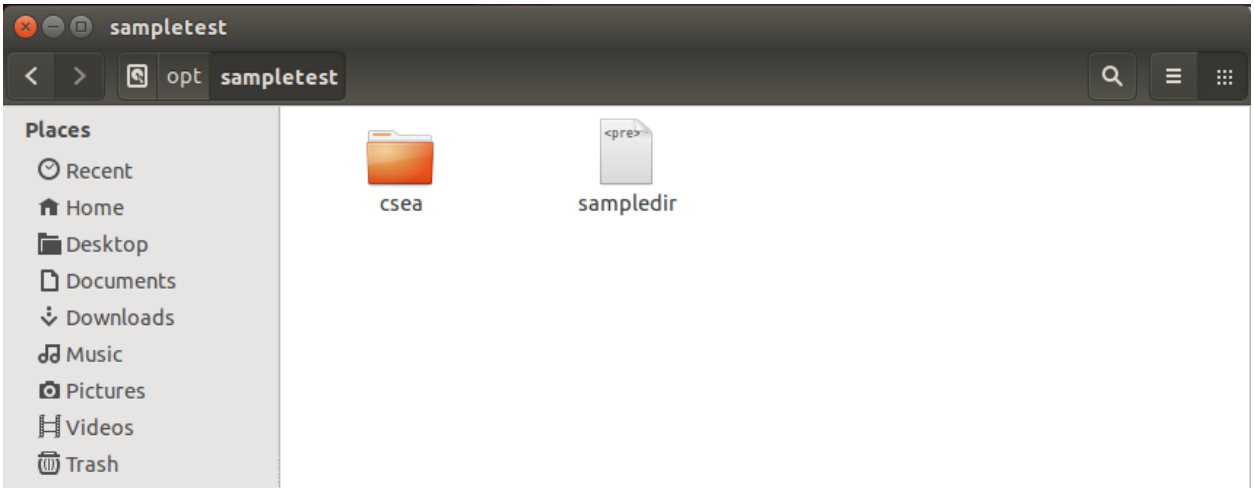
hduser@administrator-Virtual Box:~$ sudo hadoop-fuse-dfs dfs://127.0.0.1:54310 /opt/sampletest -d

TO CHECK THE MOUNT POINT:

hduser@administrator-Virtual Box:~$df -h

BROWSER OUTPUT:





RESULT:

Thus, the Procedure of transferring the files from one virtual machine to another virtual machine has been executed.

Ex. No : 7 Date :	THE VIRTUAL MACHINE MIGRATION FROM ONE NODE TO THE OTHER

AIM:

To Find a procedure to launch a virtual machine using try stack (Online Openstack Demo Version)

PROCEDURE:

Migrating instances

1. Check the ID of the instance to be migrated:

```
$ nova list
```

ID		Status	Networks
d1df1b5a-70c4-4fed-98b7-423362f2c47c		ACTIVE	private=a.b.c.d
d693db9e-a7cf-45ef-a7c9-b3ecb5f22645		ACTIVE	private=e.f.g.h

2. Check the information associated with the instance. In this example, `vm1` is running on `HostB`:

```
$ nova show d1df1b5a-70c4-4fed-98b7-423362f2c47c
```

Property	Value
...	...
OS-EXT-SRV-ATTR:host	HostB
...	...
flavor	m1.tiny
id	d1df1b5a-70c4-4fed-98b7-423362f2c47c
name	vm1
private network	a.b.c.d
status	ACTIVE
...	...

3. Select the compute node the instance will be migrated to. In this example, we will migrate the instance to `HostC`, because `nova-compute` is running on it:

4. Check that `HostC` has enough resources for migration:

```
# nova host-describe HostC
```

- `cpu`: Number of CPUs
- `memory_mb`: Total amount of memory, in MB
- `disk_gb`: Total amount of space for NOVA-INST-DIR/instances, in GB

5. Migrate the instance using the **nova live-migration** command:

```
$ nova live-migration SERVER HOST_NAME
```

In this example, SERVER can be the ID or name of the instance. Another example:

```
$ nova live-migration d1df1b5a-70c4-4fed-98b7-423362f2c47c HostC
```

```
Migration of d1df1b5a-70c4-4fed-98b7-423362f2c47c initiated.
```

6. To migrate an instance and watch the status, use this example script:

```
#!/bin/bash
# Provide usage
usage() {
echo "Usage: $0 VM_ID"
exit 1
}
[[ $# -eq 0 ]] && usage
```

```
# Migrate the VM to an alternate hypervisor

echo -n "Migrating instance to alternate host"

VM_ID=$1

nova migrate $VM_ID

VM_OUTPUT=`nova show $VM_ID`

VM_STATUS=`echo "$VM_OUTPUT" | grep status | awk '{print $4}'`

while [[ "$VM_STATUS" != "VERIFY_RESIZE" ]]; do

echo -n "."

sleep 2

VM_OUTPUT=`nova show $VM_ID`
```

```
VM_STATUS=`echo "$VM_OUTPUT" | grep status | awk '{print $4}'`
done
nova resize-confirm $VM_ID
echo " instance migrated and resized."
echo;
# Show the details for the VM
echo "Updated instance details:"
nova show $VM_ID
# Pause to allow users to examine VM details
read -p "Pausing, press <enter> to exit."
```

RESULT:

Thus, the procedure to launch a virtual machine using try stack has been successfully completed.

Ex. No : 8 Date :	: HADOOP INSTALLATION

AIM:

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

PROCEDURE:

Procedure to set up one node Hadoop Cluster

Hadoop2.6 on Ubuntu 14.04

Installation of a single-node Hadoop cluster backed by the Hadoop Distributed File System on Ubuntu.

Installing Java

Hadoop framework is written in Java.

```
administrator@administrator-Virtual Box:~$ cd ~
# Update the source list
administrator@administrator-Virtual Box:~$ sudo apt-get update

# The OpenJDK project is the default version of Java
# that is provided from a supported Ubuntu repository.
administrator@administrator-Virtual Box:~$ sudo apt-get install default-jdk

administrator@administrator-Virtual Box:~$ java -version
java version "1.7.0_65"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Adding a dedicated Hadoop user

```
administrator@administrator-Virtual Box:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1002) ...
Done.

administrator@administrator-Virtual Box:~$ sudo adduser --ingroup
hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1001) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: hadoop
Retype new UNIX password: hadoop
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default
      Full Name []:
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] Y

administrator@administrator-Virtual Box $ sudo adduser hduser sudo
[sudo] password for administrator: kgisl
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

Installing SSH

ssh has two main components:

1. ssh : The command is used for connecting to remote machines - the client.
2. sshd : The daemon that is running on the server and allows clients to connect, to the server.

The ssh is pre-enabled on Linux, but in order to start sshd daemon, install ssh first.

```
administrator@administrator-Virtual Box:~$ sudo apt-get install ssh
```

To check the installation of ssh on the machine:

```
administrator@administrator-Virtual Box:~$ which ssh
/usr/bin/ssh

administrator@administrator-Virtual Box:~$ which sshd
/usr/sbin/sshd
```

Create and Setup SSH Certificates

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH

certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

```
administrator@administrator-Virtual Box:~$ su hduser
Password:
administrator@administrator-Virtual Box:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3   hduser@administrator-
Virtual Box
The key's randomart image is:
+--[ RSA 2048]----+
|        .oo.o    |
|       . .o=. o  |
|      . + .  o . |
|       o =    E  |
|        S +      |
|         . +     |
|          O +    |
|           O o   |
|            o..  |
+-----------------+


hduser@administrator-Virtual Box:/home/k$ cat
$HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

To check if ssh works:

```
hduser@administrator-Virtual Box:/home/k$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is
e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known
hosts.
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86_64)
...
```

Install Hadoop

```
hduser@administrator-Virtual Box:~$ wget
ftp://172.16.0.3/cloudlab/hadoop-2.6.3.tar.gz
hduser@administrator-Virtual Box:~$ tar xvzf hadoop-2.6.3.tar.gz
```

To move the Hadoop installation to the /usr/local/hadoop directory using the following command:

```
hduser@administrator-Virtual Box:~/hadoop-2.6.0$ sudo mv hadoop-
2.6.3 /usr/local/hadoop
hduser@administrator-Virtual Box:~/hadoop-2.6.0$ sudo chown -R
hduser:hadoop /usr/local/hadoop
```

Setup Configuration Files

The following files will have to be modified to complete the Hadoop setup:

1. ~/.bashrc

2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh
3. /usr/local/hadoop/etc/hadoop/core-site.xml
4. /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

1. ~/.bashrc:

Before editing the .bashrc file in our home directory, find the path where Java has been installed to set the JAVA_HOME environment variable using the following command:

```
hduser@administrator-Virtual Box update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java
Nothing to configure.
```

Now append the following to the end of ~/.bashrc:

```
hduser@administrator-Virtual Box:~$ vi ~/.bashrc

#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
```

```
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END

hduser@administrator-Virtual Box:~$ source ~/.bashrc
```

2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh

To set JAVA_HOME by modifying hadoop-env.sh file.

```
hduser@administrator-Virtual Box:~$ vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Adding the above statement in the hadoop-env.sh file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

3. /usr/local/hadoop/etc/hadoop/core-site.xml:

The /usr/local/hadoop/etc/hadoop/core-site.xml file contains configuration properties that Hadoop uses when starting up.

This file can be used to override the default settings that Hadoop starts with.

```
hduser@administrator-Virtual Box:~$ sudo mkdir -p /app/hadoop/tmp
hduser@administrator-Virtual Box:~$ sudo chown hduser:hadoop /app/hadoop/tmp
```

Open the file and enter the following in between the <configuration></configuration> tag:

```
hduser@administrator-Virtual Box:~$ vi /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system. A URI whose scheme and authority determine the FileSystem implementation. The uri's scheme determines the config property (fs.SCHEME.impl) naming the FileSystem implementation class. The uri's authority is used to determine the host, port, etc. for a filesystem. </description>
 </property>
</configuration>
```

4. /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the /usr/local/hadoop/etc/hadoop/ folder contains /usr/local/hadoop/etc/hadoop/mapred-site.xml.template file which has to be renamed/copied with the name mapred-site.xml:

```
hduser@administrator-Virtual Box:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```
hduser@administrator-Virtual Box:~$ vi
/usr/local/hadoop/etc/hadoop/mapred-site.xml
```

The mapred-site.xml file is used to specify which framework is being used for MapReduce.

Enter the following content in between the <configuration></configuration> tag:

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs at. If "local", then jobs are run in-process as a single map and reduce task.
  </description>
 </property>
</configuration>
```

5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The /usr/local/hadoop/etc/hadoop/hdfs-site.xml file needs to be configured for each host in the cluster that is being used.

It is used to specify the directories which will be used as the namenode and the datanode on that host.

Before that, create two directories which will contain the namenode and the datanode for the Hadoop installation.

```
hduser@administrator-Virtual Box:~$ sudo mkdir -p
/usr/local/hadoop_store/hdfs/namenode
```

```
hduser@administrator-Virtual Box:~$ sudo mkdir -p
/usr/local/hadoop_store/hdfs/datanode
hduser@administrator-Virtual Box:~$ sudo chown -R hduser:hadoop
/usr/local/hadoop_store
```

Open the file and enter the following content in between the <configuration></configuration> tag:

```
hduser@administrator-Virtual Box:~$ vi
/usr/local/hadoop/etc/hadoop/hdfs-site.xml


<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when
the file is created.
  The default is used if replication is not specified in
create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
```

```
 <property>
   <name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Format the New Hadoop Filesystem

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates current directory under /usr/local/hadoop_store/hdfs/namenode folder:

```
hduser@administrator-Virtual Box:~$ hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

15/04/18 14:43:03 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = administrator-Virtual Box/192.168.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.0
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop
...
STARTUP_MSG:   java = 1.7.0_65
************************************************************/
15/04/18 14:43:03 INFO namenode.NameNode: registered UNIX signal
handlers for [TERM, HUP, INT]
15/04/18 14:43:03 INFO namenode.NameNode: createNameNode [-format]
```

```
15/04/18 14:43:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Formatting using clusterid: CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f

15/04/18 14:43:09 INFO namenode.FSNamesystem: No KeyProvider found.

15/04/18 14:43:09 INFO namenode.FSNamesystem: fsLock is fair:true

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.block.invalidate.limit=1000

15/04/18 14:43:10 INFO blockmanagement.DatanodeManager: dfs.namenode.datanode.registration.ip-hostname-check=true

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00:00.000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: The block deletion will start around 2015 Apr 18 14:43:10

15/04/18 14:43:10 INFO util.GSet: Computing capacity for map BlocksMap

15/04/18 14:43:10 INFO util.GSet: VM type = 64-bit

15/04/18 14:43:10 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB

15/04/18 14:43:10 INFO util.GSet: capacity = 2^21 = 2097152 entries

15/04/18 14:43:10 INFO blockmanagement.BlockManager: dfs.block.access.token.enable=false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: defaultReplication = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplication = 512

15/04/18 14:43:10 INFO blockmanagement.BlockManager: minReplication = 1

15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplicationStreams = 2

15/04/18 14:43:10 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks = false

15/04/18 14:43:10 INFO blockmanagement.BlockManager: replicationRecheckInterval = 3000

15/04/18 14:43:10 INFO blockmanagement.BlockManager: encryptDataTransfer = false
```

```
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxNumBlocksToLog
= 1000

15/04/18 14:43:10 INFO namenode.FSNamesystem: fsOwner                =
hduser (auth:SIMPLE)

15/04/18 14:43:10 INFO namenode.FSNamesystem: supergroup             =
supergroup

15/04/18 14:43:10 INFO namenode.FSNamesystem: isPermissionEnabled =
true

15/04/18 14:43:10 INFO namenode.FSNamesystem: HA Enabled: false

15/04/18 14:43:10 INFO namenode.FSNamesystem: Append Enabled: true

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map INodeMap

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB

15/04/18 14:43:11 INFO util.GSet: capacity         = 2^20 = 1048576
entries

15/04/18 14:43:11 INFO namenode.NameNode: Caching file names occuring
more than 10 times

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map
cachedBlocks

15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB

15/04/18 14:43:11 INFO util.GSet: capacity         = 2^18 = 262144
entries

15/04/18 14:43:11 INFO namenode.FSNamesystem:
dfs.namenode.safemode.threshold-pct = 0.9990000128746033

15/04/18 14:43:11 INFO namenode.FSNamesystem:
dfs.namenode.safemode.min.datanodes = 0

15/04/18 14:43:11 INFO namenode.FSNamesystem:
dfs.namenode.safemode.extension     = 30000

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache on namenode
is enabled

15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache will use
0.03 of total heap and retry cache entry expiry time is 600000 millis

15/04/18 14:43:11 INFO util.GSet: Computing capacity for map
NameNodeRetryCache
```

```
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit

15/04/18 14:43:11 INFO util.GSet: 0.029999999329447746% max memory 889
MB = 273.1 KB

15/04/18 14:43:11 INFO util.GSet: capacity      = 2^15 = 32768 entries

15/04/18 14:43:11 INFO namenode.NNConf: ACLs enabled? false

15/04/18 14:43:11 INFO namenode.NNConf: XAttrs enabled? true

15/04/18  14:43:11  INFO  namenode.NNConf:  Maximum  size  of  an  xattr:
16384

15/04/18  14:43:12  INFO  namenode.FSImage:  Allocated  new  BlockPoolId:
BP-130729900-192.168.1.1-1429393391595

15/04/18    14:43:12    INFO    common.Storage:    Storage    directory
/usr/local/hadoop_store/hdfs/namenode has been successfully formatted.

15/04/18  14:43:12  INFO  namenode.NNStorageRetentionManager:  Going  to
retain 1 images with txid >= 0

15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0

15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN_MSG:

/************************************************************

SHUTDOWN_MSG:   Shutting   down   NameNode   at   administrator-Virtual
Box/192.168.1.1

************************************************************/
```

Note that hadoop namenode -format command should be executed once before we start using Hadoop.

If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

Starting Hadoop

Now it's time to start the newly installed single node cluster.

We can use start-all.sh or (start-dfs.sh and start-yarn.sh)

```
hduser@administrator-Virtual Box:~$ cd /usr/local/hadoop/sbin


hduser @administrator-Virtual Box:/usr/local/hadoop/sbin$ ls
distribute-exclude.sh    start-all.cmd        stop-balancer.sh
hadoop-daemon.sh         start-all.sh         stop-dfs.cmd
hadoop-daemons.sh        start-balancer.sh    stop-dfs.sh
hdfs-config.cmd          start-dfs.cmd        stop-secure-dns.sh
hdfs-config.sh           start-dfs.sh         stop-yarn.cmd
httpfs.sh                start-secure-dns.sh  stop-yarn.sh
kms.sh                   start-yarn.cmd       yarn-daemon.sh
mr-jobhistory-daemon.sh  start-yarn.sh        yarn-daemons.sh
refresh-namenodes.sh     stop-all.cmd
slaves.sh                stop-all.sh


hduser@administrator-Virtual    Box:/usr/local/hadoop/sbin$    start-
all.sh
hduser@administrator-Virtual Box:~$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
Starting namenodes on [localhost]
localhost:         starting         namenode,         logging         to
/usr/local/hadoop/logs/hadoop-hduser-namenode-administrator-Virtual
Box.out
localhost:         starting         datanode,         logging         to
/usr/local/hadoop/logs/hadoop-hduser-datanode-administrator-Virtual
Box.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0:       starting       secondarynamenode,       logging       to
/usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-administrator-
Virtual Box.out
```

```
15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

starting yarn daemons

starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-administrator-Virtual Box.out

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-administrator-Virtual Box.out
```

We can check if it's really up and running:

```
hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$ jps

9026 NodeManager

7348 NameNode

9766 Jps

8887 ResourceManager

7507 DataNode
```

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

Another way to check is using netstat:

```
hduser@administrator-Virtual Box:~$ netstat -plten | grep java

(Not all processes could be identified, non-owned process info

 will not be shown, you would have to be root to see it all.)

tcp              0          0  0.0.0.0:50020                  0.0.0.0:*
LISTEN      1001       1843372     10605/java

tcp              0           0  127.0.0.1:54310               0.0.0.0:*
LISTEN      1001       1841277     10447/java

tcp              0          0  0.0.0.0:50090                  0.0.0.0:*
LISTEN      1001       1841130     10895/java
```

```
tcp              0          0  0.0.0.0:50070                   0.0.0.0:*
LISTEN      1001       1840196     10447/java

tcp              0          0  0.0.0.0:50010                   0.0.0.0:*
LISTEN      1001       1841320     10605/java

tcp              0          0  0.0.0.0:50075                   0.0.0.0:*
LISTEN      1001       1841646     10605/java

tcp6             0          0  :::8040                              :::*
LISTEN      1001       1845543     11383/java

tcp6             0          0  :::8042                              :::*
LISTEN      1001       1845551     11383/java

tcp6             0          0  :::8088                              :::*
LISTEN      1001       1842110     11252/java

tcp6             0           0  :::49630                             :::*
LISTEN      1001       1845534     11383/java

tcp6             0          0  :::8030                              :::*
LISTEN      1001       1842036     11252/java

tcp6             0          0  :::8031                              :::*
LISTEN      1001       1842005     11252/java

tcp6             0          0  :::8032                              :::*
LISTEN      1001       1842100     11252/java

tcp6             0          0  :::8033                              :::*
LISTEN      1001       1842162     11252/java
```

Stopping Hadoop

```
$ pwd

/usr/local/hadoop/sbin


$ ls

distribute-exclude.sh   httpfs.sh                          start-all.sh
start-yarn.cmd    stop-dfs.cmd        yarn-daemon.sh

hadoop-daemon.sh            mr-jobhistory-daemon.sh    start-balancer.sh
start-yarn.sh     stop-dfs.sh         yarn-daemons.sh

hadoop-daemons.sh            refresh-namenodes.sh          start-dfs.cmd
stop-all.cmd      stop-secure-dns.sh

hdfs-config.cmd            slaves.sh                        start-dfs.sh
stop-all.sh       stop-yarn.cmd
```

```
hdfs-config.sh           start-all.cmd              start-secure-dns.sh
stop-balancer.sh  stop-yarn.sh
```

We run stop-all.sh or (stop-dfs.sh and stop-yarn.sh) to stop all the daemons running on our machine:

```
hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$ pwd

/usr/local/hadoop/sbin

hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$ ls

distribute-exclude.sh   httpfs.sh                          start-all.cmd
start-secure-dns.sh  stop-balancer.sh    stop-yarn.sh

hadoop-daemon.sh          kms.sh                            start-all.sh
start-yarn.cmd       stop-dfs.cmd        yarn-daemon.sh

hadoop-daemons.sh          mr-jobhistory-daemon.sh    start-balancer.sh
start-yarn.sh        stop-dfs.sh         yarn-daemons.sh

hdfs-config.cmd              refresh-namenodes.sh         start-dfs.cmd
stop-all.cmd         stop-secure-dns.sh

hdfs-config.sh             slaves.sh                        start-dfs.sh
stop-all.sh          stop-yarn.cmd

hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$

hduser@administrator-Virtual Box:/usr/local/hadoop/sbin$ stop-all.sh

This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh

15/04/18 15:46:31 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable

Stopping namenodes on [localhost]

localhost: stopping namenode

localhost: stopping datanode

Stopping secondary namenodes [0.0.0.0]

0.0.0.0: no secondarynamenode to stop

15/04/18 15:46:59 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable

stopping yarn daemons
```

```
stopping resourcemanager
localhost: stopping nodemanager
no proxyserver to stop
```

Hadoop Web Interfaces

Let's start the Hadoop again and see its Web UI:

```
hduser@administrator-Virtual  Box:/usr/local/hadoop/sbin$  start-all.sh
```

http://localhost:50070/ - web UI of the NameNode daemon

SecondaryNameNode

(Note) I had to restart Hadoop to get this Secondary Namenode.

DataNode

WORD COUNT PROGRAM

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(Object key, Text value, Context context
) throws IOException, InterruptedException {
StringTokenizer itr = new StringTokenizer(value.toString());
```

```
while (itr.hasMoreTokens()) {
word.set(itr.nextToken());
context.write(word, one);
}
}
}
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

START THE HADOOP:

hduser@administrator-VirtualBox:~$ sudo su hduser

hduser@administrator-VirtualBox:~$ cd /usr/local/hadoop/sbin

hduser @administrator-VirtualBox:/usr/local/hadoop/sbin$ ls

hduser@administrator-VirtualBox:/usr/local/hadoop/sbin$ start-all.sh

hduser@administrator-VirtualBox:/usr/local/hadoop/sbin$ jps

INPUT DIRECTORY:

hduser@administrator-VirtualBox:~$ /usr/local/hadoop/bin/hdfs dfs -mkdir /sampledir

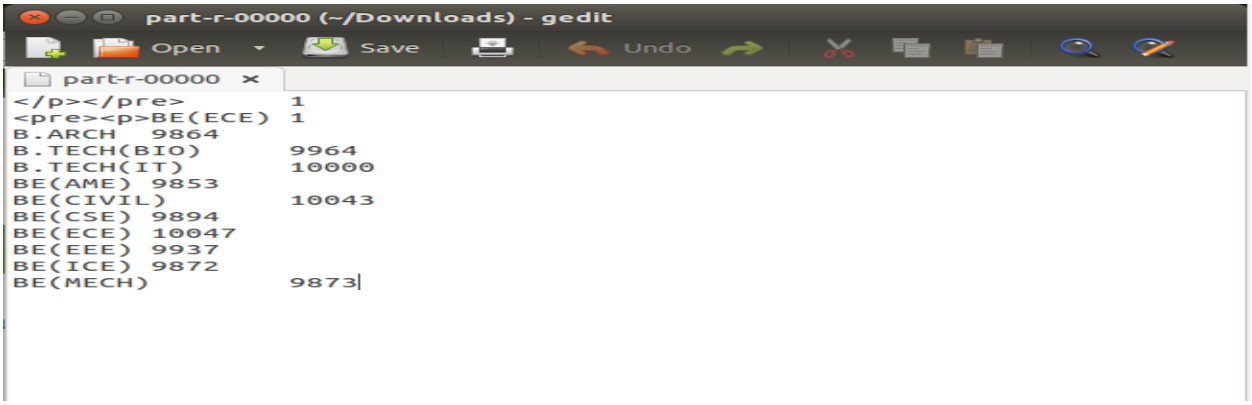
MOVE THE DATA TO INPUT DIRECTORY:

hduser@administrator-VirtualBox:~$ /usr/local/hadoop/bin/hdfs dfs -put /home/administrator/Desktop/file1.txt /sampledir

EXECUTIONOF JAR FILE:

hduser@administrator-VirtualBox:~$ /usr/local/hadoop/bin/hadoop jar /home/administrator/Desktop/WordCount.jar Wordcount /sampledir/ /output '(CSE)'

OUTPUT BROWSER:



RESULT:

Thus, the Hadoop Installation has been successfully completed.