# Vel Tech Multi Tech

Dr.Rangarajan Dr.Sakunthala Engineering College

An Autonomous Institution

**Approved by AICTE, Affiliated to Anna University, Chennai.**
**ISO 9001:2015 Certified Institution, Accredited by NBA (BME, CSE, ECE, EEE, IT & MECH),**
**Accredited by NAAC**
**#42, Avadi-Vel Tech Road, Avadi, Chennai- 600062, Tamil Nadu, India.**

**DEPARTMENT OF COMPUTER SCEINEC AND ENGINEERING**

## 191CS72A/ CLOUD COMPUTING LABORATORY

**NAME** :

**REGISTER NO** :

**ROLL NO** :

**BRANCH** : **B.E- Computer Science and Engineering**

**YEAR** : **IV**

**SEMESTER** : **VII**

---

**Department of Computer Science and Engineering**

**Vision**

To emerge as centre for academic excellence in the field of Computer Science and Engineering by exposure to research and industry practices.

**Mission**

**M1** - To provide good teaching and learning environment with conducive research atmosphere in the field of Computer Science and Engineering.

**M2** - To propagate lifelong learning.

**M3** - To impart the right proportion of knowledge, attitudes and ethics in students to enable them take up positions of responsibility in the society and make significant contributions

# Vel Tech Multi Tech

## Dr.Rangarajan Dr.Sakunthala Engineering College

## An Autonomous Institution

**Approved by AICTE, Affiliated to Anna University, Chennai.**
**ISO 9001:2015 Certified Institution, Accredited by NBA (BME, CSE, ECE, EEE, IT & MECH),**
**Accredited by NAAC**
**#42, Avadi-Vel Tech Road, Avadi, Chennai- 600062, Tamil Nadu, India.**


**CERTIFICATE**

Name …….. Year: …….

Semester:……. Branch: **B.E– COMPUTER SCIENCE AND ENGINEERING**

Register No…….. ……VM No:……..

Certified that this is the bonafide record of work done by the above student in the **191CS72A – CLOUD COMPUTING LABORATORY** during the academic year **2022-2023.**

**Signature of Course Incharge** **Signature of Head of the Department**

---

Submitted for the University Practical Examination held on ..... at **VEL TECH MULTI TECH Dr.RANGARAJAN Dr.SAKUNTHALA ENGINEERING COLLEGE, No.42, AVADI – VEL TECHROAD, AVADI, CHENNAI- 600062.**

**Signature of Examiners**

**Internal Examiner:……** **External Examiner:……**

**Date:……**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

| PEOs | PROGRAMME EDUCATIONAL OBJECTIVES |
|---|---|
| **PEO1** | Ability to identify, formulate and analyze complex Computer Science and Engineering problems in the areas of hardware, software, theoretical Computer Science and applications to reach significant conclusions by applying Mathematics, Natural sciences, Computer Science and Engineering principles. |
| **PEO2** | Apply knowledge of mathematics, natural science, engineering fundamentals and system fundamentals, software development, networking & communication, and information security to the solution of complex engineering problems in computer science and engineering to get benefits in their professional career or higher education and research or technological entrepreneur. |
| **PEO3** | Design solutions for complex computer science and engineering problems using state of the art tools and techniques, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations. |

### PROGRAMME SPECIFIC OUTCOMES (PSOs)

| PSO's | PROGRAMME SPECIFIC OUTCOMES |
|---|---|
| **PSO1** | An ability to apply, design and development of application oriented software systems and to test and document in accordance with Computer Science and Engineering. |
| **PSO2** | The design techniques, analysis and the building, testing, operation and maintenance of networks, databases, security and computer systems (both hardware and software). |
| **PSO3** | An ability to identify, formulate and solve hardware and software problems using sound computer engineering principles. |

\

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### PROGRAMME OUTCOMES (POs)

| POs | Programme Outcomes (POs) |
|---|---|
| **PO1** | Apply knowledge of **mathematics, natural science**, **engineering** fundamentals and system fundamentals, software development, networking & communication, and information assurance & security to the solution of complex engineering problems in computer science and engineering. |
| **PO2** | Ability to **identify**, **formulate** and **analyze complex** Computer Science and Engineering problems in the areas of hardware, software, theoretical Computer Science and applications to reach significant conclusions by applying Mathematics, Natural sciences, Computer Science and Engineering principles. |
| **PO3** | **Design solutions** for complex computer science and engineering problems and design systems, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations. |
| **PO4** | Ability to use **research based knowledge** and **research methods** to perform literature survey, design experiments for complex problems in designing, developing and maintaining a computing system, collect data from the experimental outcome, analyze and interpret valid/interesting patterns and conclusions from the data points. |
| **PO5** | Ability to create, select and apply **state of the art tools** and techniques in designing, developing and testing a computing system or its component. |
| **PO6** | Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to **professional engineering practice** in system development and solutions to **complex engineering problems** related to system fundamentals, software development, networking & communication, and information assurance & security. |
| **PO7** | Understand and evaluate the sustainability and **impact of professional engineering** work in the solution of complex engineering problems related to system fundamentals, software development, networking & communication, and information assurance & security in societal and environmental contexts. |
| **PO8** | Apply **ethical principles** and commit to **professional ethics** and **responsibilities** and norms of computer science and engineering practice. |
| **PO9** | Ability to function as an individual and as a team player or leader in multidisciplinary teams and strive towards **achieving a common goal**. |
| **PO10** | **Communicate effectively** on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| **PO11** | Demonstrate 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, to **manage projects** and in multidisciplinary environments. |
| **PO12** | Recognize the need for, and have the preparation and ability to **engage in independent** and **lifelong learning** in the broadest context of technological change. |

**COURSE OBJECTIVES**

## COURSE OBJECTIVES

- ✓ To develop web applications in cloud
- ✓ To design and development process involved in creating a cloud based Application
- ✓ To implement and use parallel programming using Hadoop

**COURSE OUTCOMES**

| **COURSE OUTCOMES**<br>On completion of the course, students will be able to | |
|---|---|
| **CO1** | Install various virtualization tools such as Virtual Box, VMware workstation. |
| **CO2** | Use Cloud SIM to run a various scheduler |
| **CO3** | Design a web application in a IaaS environment. |
| **CO4** | Develop a generic cloud environment which can be used as a private cloud |
| **CO5** | Implement version control systems with various command repositories |

**CO-PO & PSO Mapping**

| CO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO 8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO 1 | 2 | 1 | 1 | 1 | 1 | - | - | - | - | - | - | - | 2 | 2 | 2 |
| CO 2 | 2 | 1 | 1 | 1 | 1 | - | - | - | - | - | - | 1 | - | 2 | - |
| CO 3 | 2 | 1 | 1 | 1 | 1 | 1 | - | - | 1 | - | - | 1 | 2 | - | 2 |
| CO 4 | 3 | 2 | 2 | 1 | 1 | 1 | -- | - | - | - | - | - | 2 | 2 | 2 |
| CO 5 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | - | - | - | - | - | 2 | 2 | 1 |
| **CO** | **3** | **1** | **1** | **1** | **1** | **1** | **1** | **-** | **1** | **-** | **-** | **1** | **2** | **2** | **2** |

**1 – Low** **2 – Medium** **3 – High**

# Table of Contents

**Ex.No.1** **CREATION OF VIRTUAL MACHINES**

**Date:**

**AIM:**

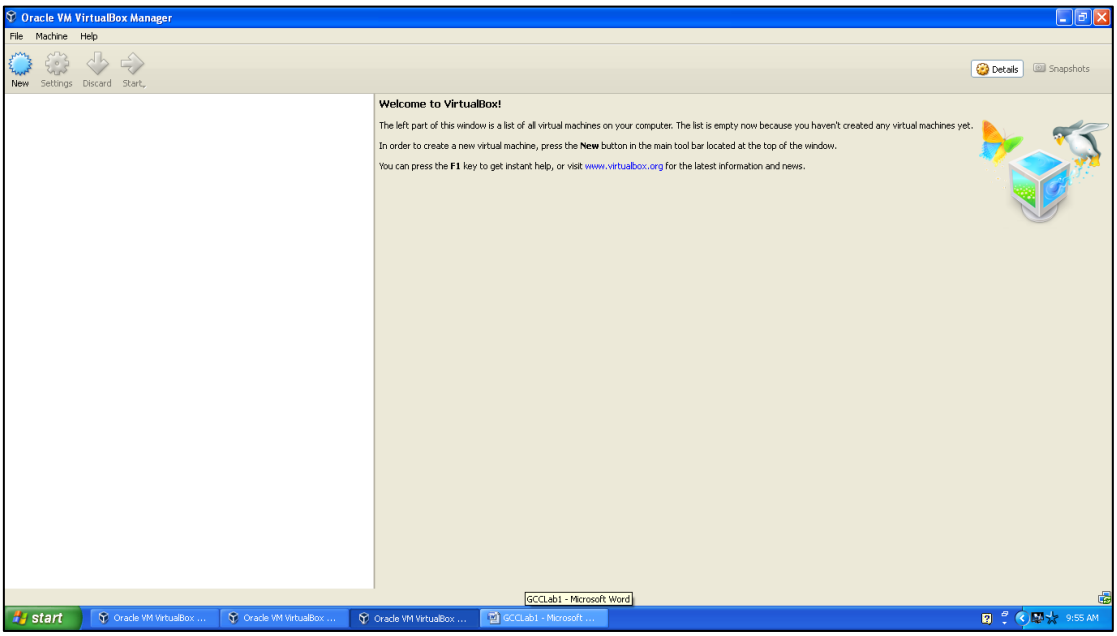
To find procedure to run the virtual machine of different configuration and check how many virtual machines can be utilized at particular time.

**PROCEDURE:**

**Installing Ubuntu using Oracle Virtual Box**

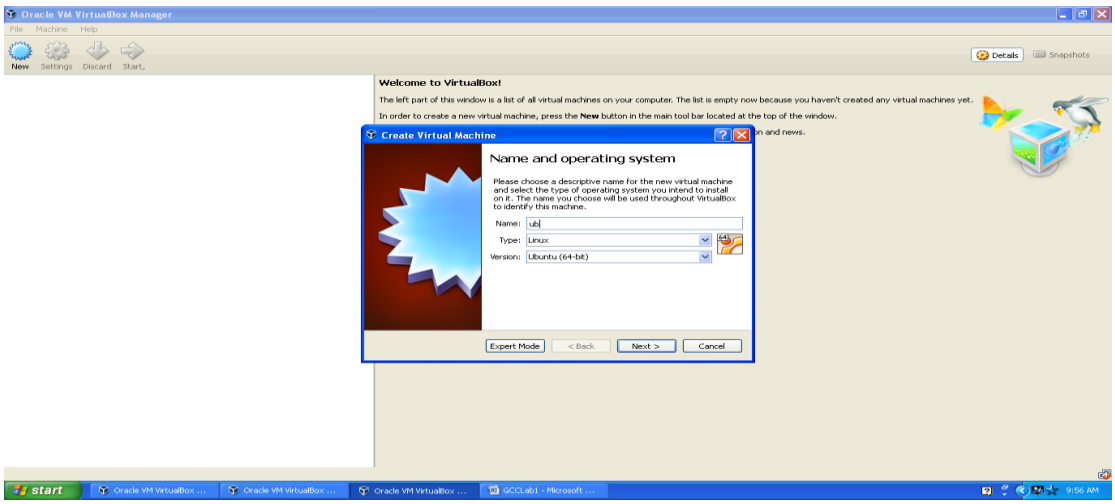
**Step1:**

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

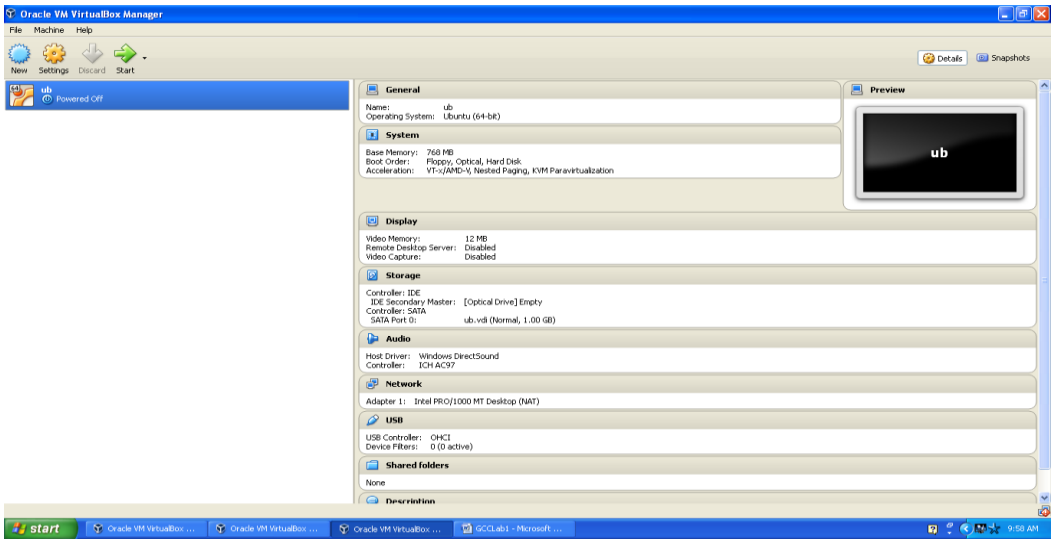



**Step 2:**

Provide a name for the virtual machine and select the hard disk size for the virtual machine.



Select the storage size as dynamically allocated memory size and click OK.




The virtual machine will be created.

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

**Step 4:**
The virtual OS Ubuntu is opened successfully.

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

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

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

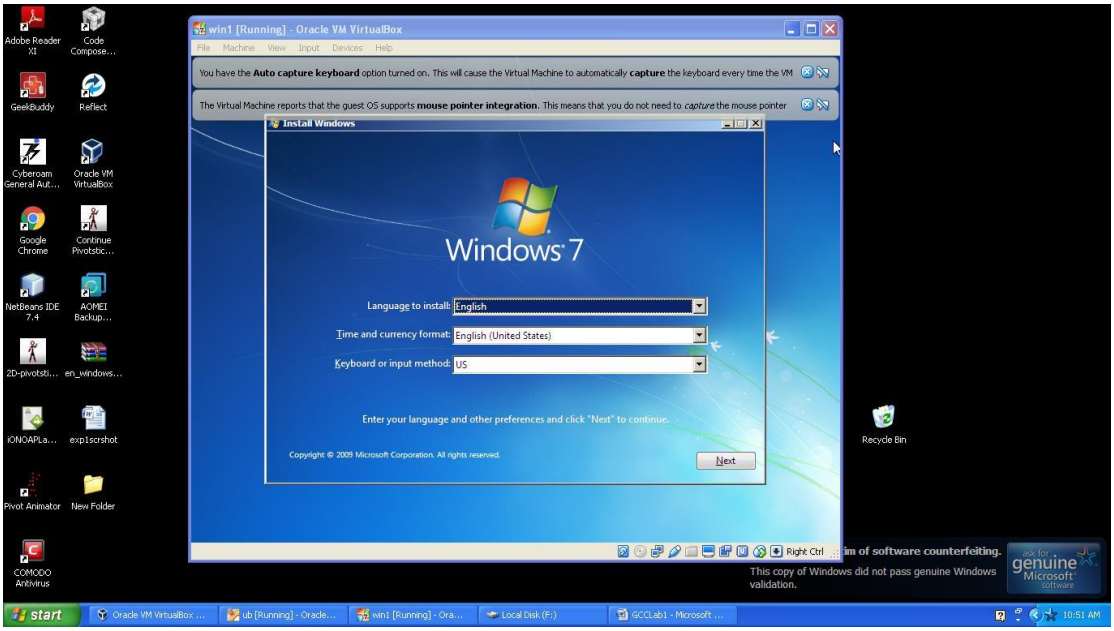
**Step 8:**
Thus the new Operating System Ubuntu will be opened as the virtual machine.

**Installing Windows 7 using Oracle Virtual Box**
**PROCEDURE:**

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

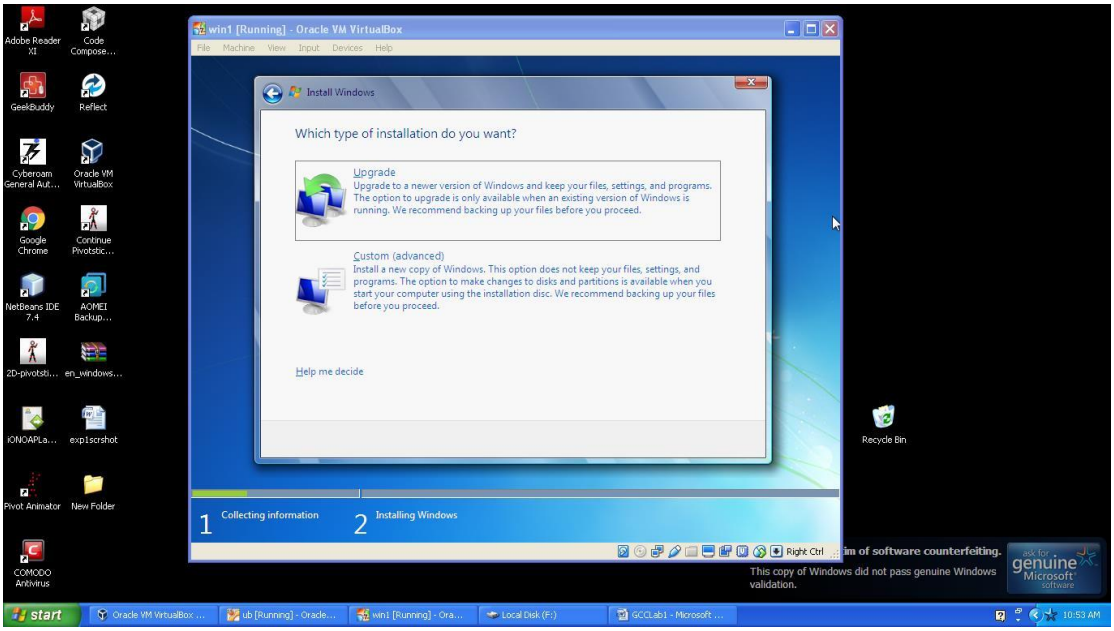
**Step 2:**
Select the iso file of the virtual OS Windows7 and click Start.

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




**Step 4:**

Select the type of installation as Custom for new installation and allocate Disk



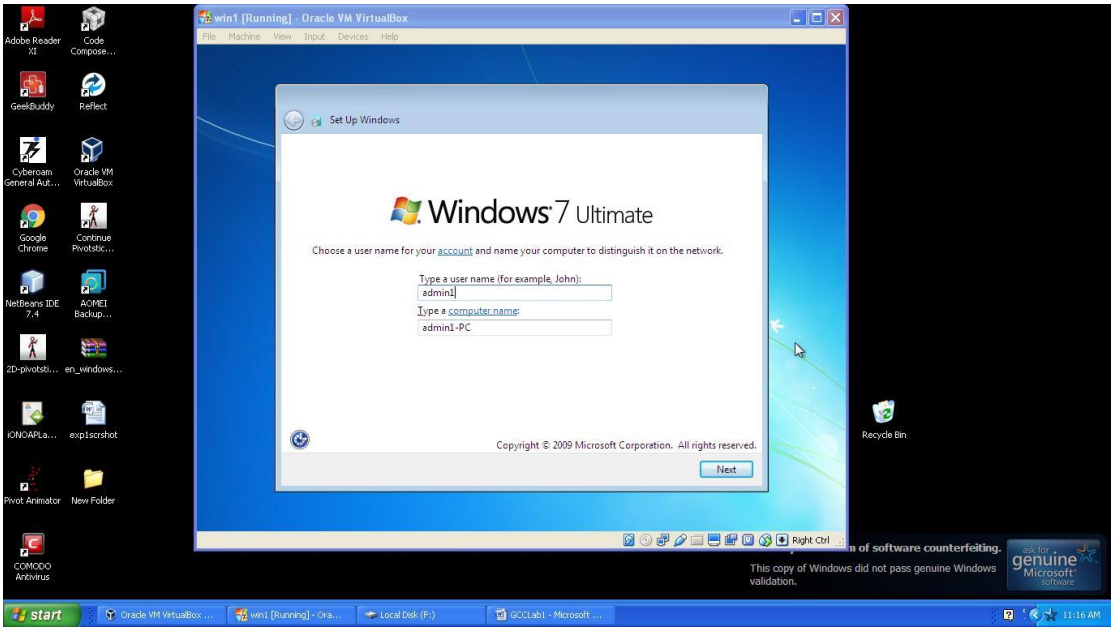

space according to your convenience. Click Next to start the installation.

**Step 5:**

After installation the system will be restarted to complete the installation.

**Step 6:**

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

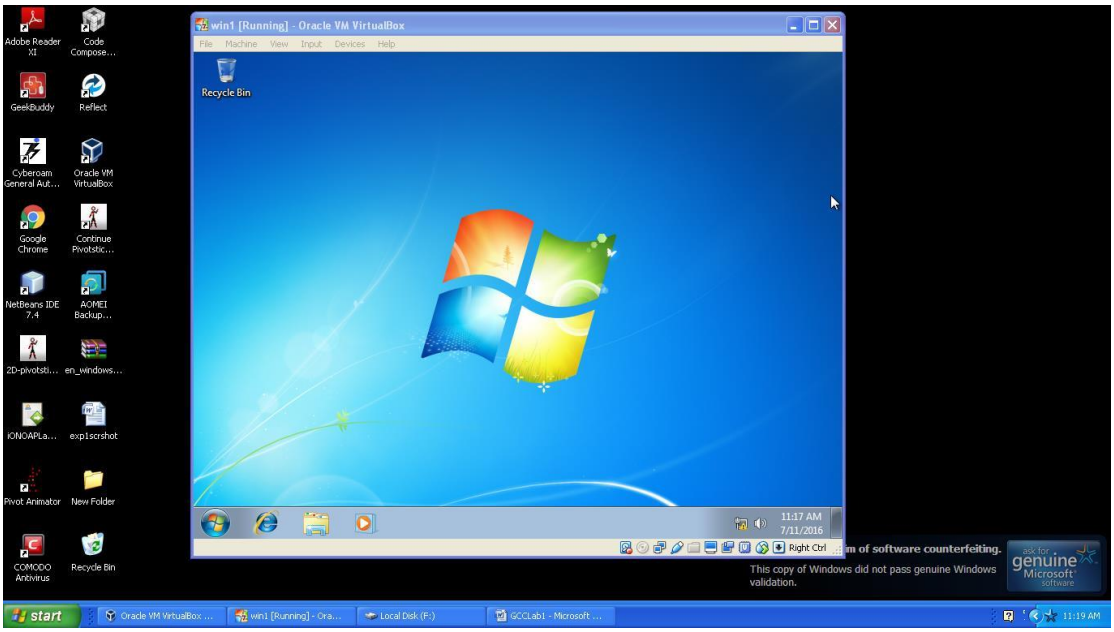



**Step 7:**

Set the time and date for the new Operating System.

**Step 8:**

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




**RESULT:**

Thus the procedure to run different virtual machines on a single system using Oracle Virtual Box is studied and implemented successfully.

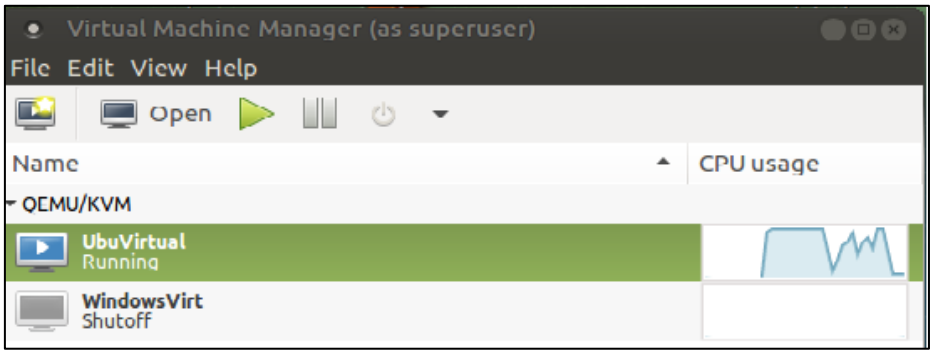
**Ex.No.2** **INSTALL A C COMPILER IN THE VIRTUAL MACHINE AND EXECUTE SIMPLE PROGRAMS**
**Date:**

**AIM:**

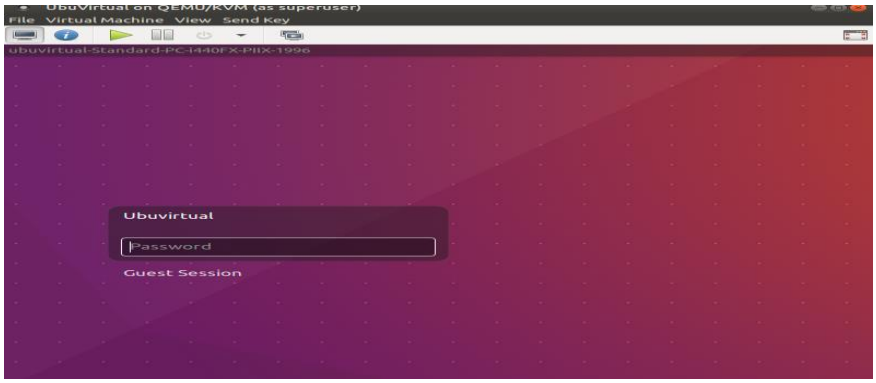
To install a C compiler in the virtual machine and execute a sample program.

**PROCEDURE:**

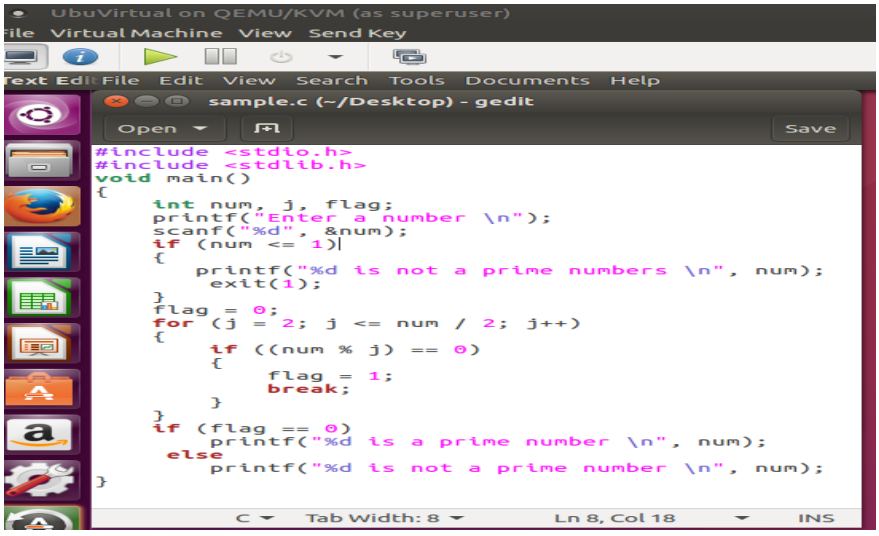
1. Before install a C compiler in a virtual machine, we have Create a virtual machine by opening the Kernal Virtual Machine (KVM). Open the installed virtual manager by using the command “sudo virt-manager”. As well as install different Ubuntu and Windows OS in that virtual machine with different names.




2. Open the Ubuntu OS in our Virtual Machine.



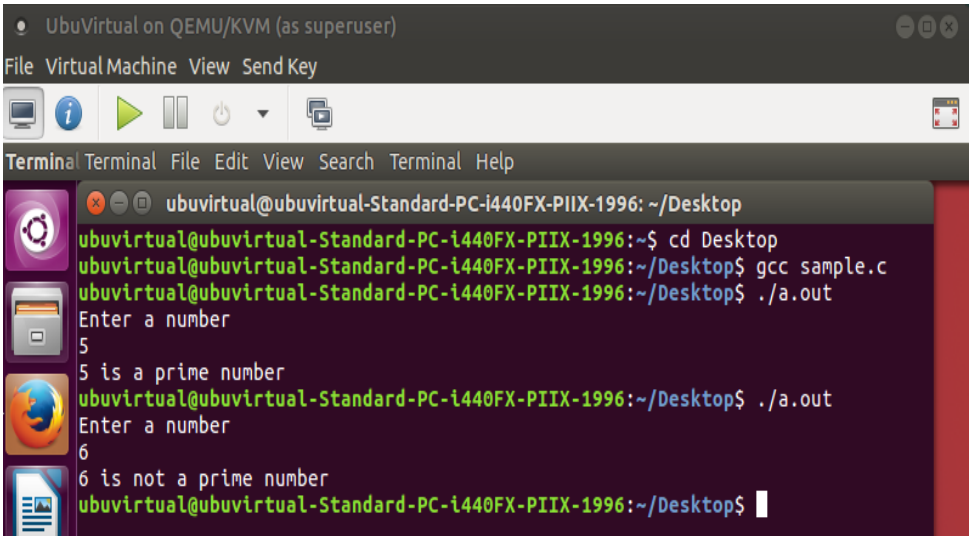

3. Open TextEditor in Ubuntu OS and type a C program and save it in desktop to execute.



4. To install the C compiler in Ubuntu OS, open the terminal and type the command.

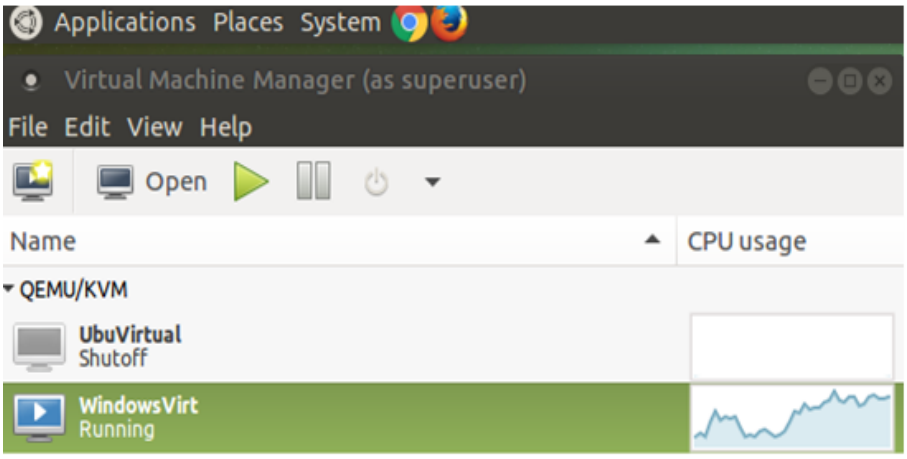
**$sudo apt-get install gcc**

5. And then complier the C program and execute it.

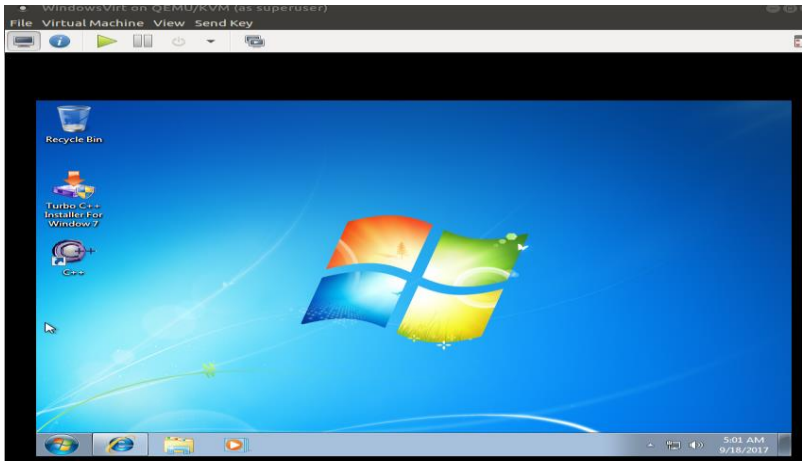



After shutting down the Ubuntu we can work in Windows Os.

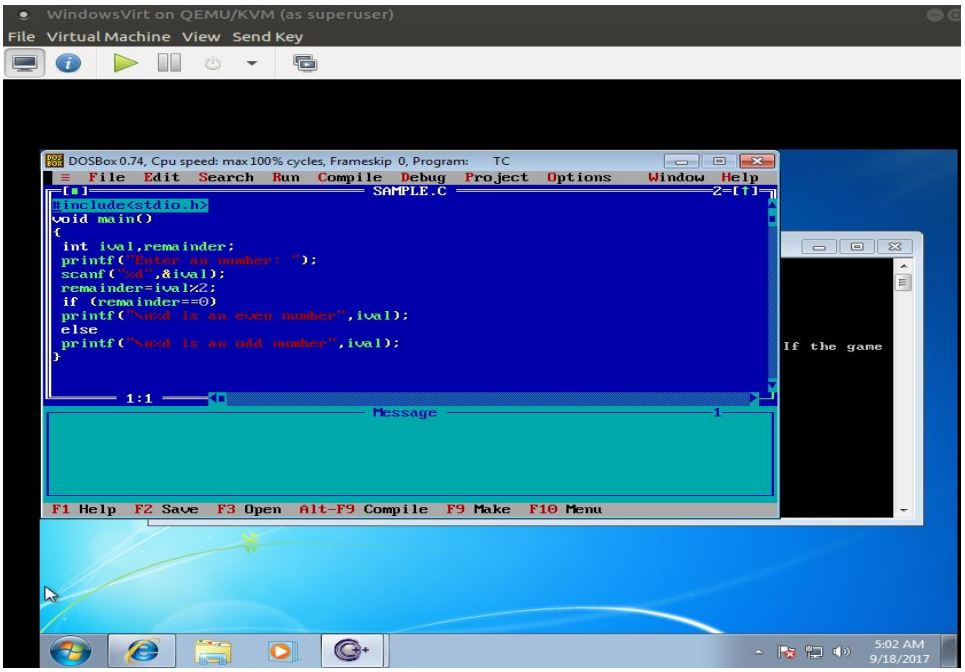
6. Open the Windows7 OS in our Virtual Machine.



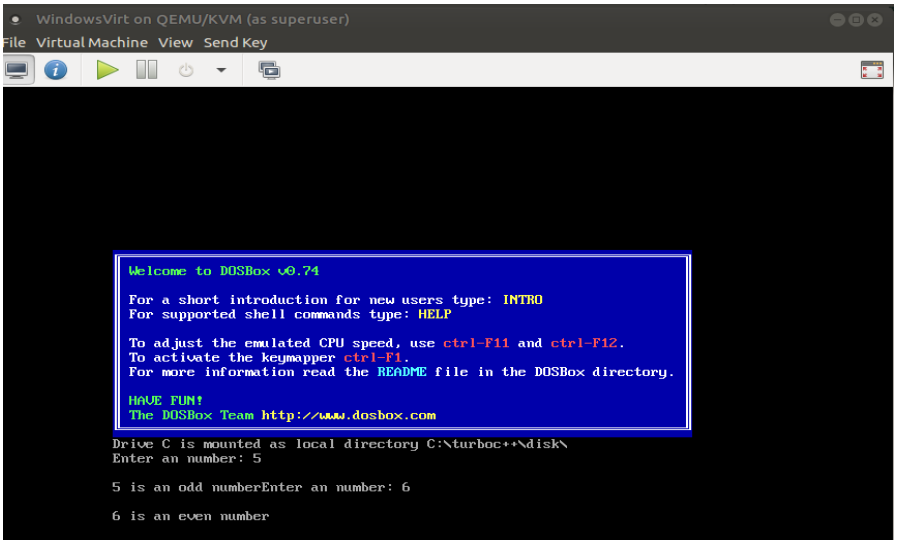

7. Download the Turbo C++ software from a Web site  As well as install the application in our system.



8. Open installed Turbo C++ and write a simple C program to run it.




9. Open installed Turbo C++ and write a simple C program to run it.




**RESULT:**

Thus, C compiler is installed in the virtual machine and executed the program

**Ex.No.3 INSTALL GOOGLE APP ENGINE. CREATE HELLO WORLD APP.**
**Date:**

**AIM:**

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

**PROCEDURE:**

https://console.cloud.google.com/appengine/start?walkthrough_tutorial_id=java_gae_quickstart
This tutorial shows you how to deploy a sample Java application to App Engine using the App Engine Maven plugin.
Here are the steps you will be taking.

- **Create a project**

Projects bundle code, VMs, and other resources together for easier development and monitoring.

- **Build and run your "Hello, world!" app**

You will learn how to run your app using Cloud Shell, right in your browser. At the end, you'll deploy your app to the web using the App Engine Maven plugin.
GCP (Google Cloud Platform) organizes resources into projects, which collect all of the related resources for a single application in one place.

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

**STEP 1:**

Select a project, or create a new one

**STEP 2 : Using Cloud Shell**

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

**Open Cloud Shell**

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

**Clone the sample code**

Use Cloud Shell to clone and navigate to the "Hello World" code. The sample code is cloned from your project repository to the Cloud Shell.
Note: If the directory already exists, remove the previous files before cloning:

```
rm -rf appengine-try-java
git clone \ https://github.com/GoogleCloudPlatform/appengine-try-java
cd appengine-try-java
```

**STEP 3: Configuring your deployment**

You are now in the main directory for the sample code. You'll look at the files that configure your application.

**Exploring the application**

Enter the following command to view your application code:

```
cat \ src/main/java/myapp/DemoServlet.java
```

This servlet responds to any request by sending a response containing the message Hello, world!.

**Exploring your configuration**
For Java, App Engine uses XML files to specify a deployment's configuration.
Enter the following command to view your configuration file:
**cat pom.xml**
The helloworld app uses Maven, which means you must specify a Project Object Model, or POM, which contains information about the project and configuration details used by Maven to build the project.

**STEP 4 : Test your app on Cloud Shell**
Cloud Shell lets you test your app before deploying to make sure it's running as intended, just like debugging on your local machine.
To test your app enter the following:
**mvn appengine:run**

**Preview your app with "Web preview"**
Your app is now running on Cloud Shell. You can access the app by clicking the **Web preview** button at the top of the Cloud Shell pane and choosing **Preview on port 8080**.
**Terminating the preview instance**
Terminate the instance of the application by pressing Ctrl+C in the Cloud Shell

**STEP 5 :Deploying to App Engine**

To deploy your app, you need to create an app in a region:
**gcloud app create**
Note: If you already created an app, you can skip this step.

**Deploying with Cloud Shell**
Now you can use Cloud Shell to deploy your app.
First, set which project to use:
**gcloud config set project **
***<YOUR-PROJECT>***

Then deploy your app:
**mvn appengine:deploy**

**Visit your app**
Congratulations! Your app has been deployed.
The default URL of your app is a subdomain on appspot.com that starts with your project's ID: <your-project>.appspot.com.
Try visiting your deployed application.

**View your app's status**

You can check in on your app by monitoring its status on the App Engine dashboard.
Open the **Navigation menu** in the upper-left corner of the console.
Then, select the **App Engine** section

**RESULT:**

Thus, the Installation of Google App Engine. Create hello world app and other simple web applications using python/java is installed and created successfully

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

**AIM:**

To Use GAE launcher to launch the web applications

**Procedure:**

**https://cloud.google.com/appengine/docs/standard/python/getting-started/hosting-a-static-website**

You can use Google App Engine to host a static website. Static web pages can contain client-side technologies such as HTML, CSS, and JavaScript. Hosting your static site on App Engine can cost less than using a traditional hosting provider, as App Engine provides a free tier. Sites hosted on App Engine are hosted on the **REGION_ID** (#appengine-urls).r.appspot.com subdomain, such as [my-project-id].uc.r.appspot.com. After you deploy your site, you can map your own domain name to your App Engine-hosted website.

Before you can host your website on Google App Engine:

1. Create a new Cloud Console project or retrieve the project ID of an existing project to use:

Go to the Projects page (https://console.cloud.google.com/project)

Tip: You can retrieve a list of your existing project IDs with the gcloud command line tool (#before_you_begin).

2. Install and then initialize the Google Cloud SDK:

Download the SDK (/sdk/docs)

Creating a website to host on Google App Engine

Basic structure for the project

This guide uses the following structure for the project:

app.yaml: Configure the settings of your App Engine application.

www/: Directory to store all of your static files, such as HTML, CSS, images, and JavaScript.

css/: Directory to store stylesheets.

style.css: Basic stylesheet that formats the look and feel of your site.

images/: Optional directory to store images.

index.html: An HTML file that displays content for your website.

js/: Optional directory to store JavaScript files.

**Creating the app.yaml _le**

The app.yaml file is a configuration file that tells App Engine how to map URLs to your static files. In the following steps, you will add handlers that will load www/index.html when someone visits your website, and all static files will be stored in and called from the www directory.

**Create the app.yaml file in your application's root directory:**

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

```
runtime: python27
api_version: 1
threadsafe: true
handlers:
- url: /
static_files: www/index.html
upload: www/index.html
- url: /(.*)
static_files: www/\1
upload: www/(.*)
```

**Creating the index.html _le**

Create an HTML file that will be served when someone navigates to the root page of your website. Store this file in your www directory.

Deploying your application to App Engine

When you deploy your application files, your website will be uploaded to App Engine. To deploy your app, run the following command from within the root directory of your application where the app.yaml file is located:

**Optional flags:**

Include the --project flag to specify an alternate Cloud Console project ID to what you initialized as the default in the gcloud tool. Example: --project

[YOUR_PROJECT_ID]
Include the -v flag to specify a version ID, otherwise one is generated for you.
Example: -v [YOUR_VERSION_ID]
l>
<head> <title>Hello, world!</title>
<link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body> <h1>Hello, world!</h1>
 This is a simple static HTML file that will be served from Google App Engine. 
</body> </html>

**Deploying your application to App Engine**

When you deploy your application files, your website will be uploaded to App Engine. To deploy your app, run the following command from within the root directory of your application where the app.yaml file is located:
id app deploy

**Optional flags:**

Include the --project flag to specify an alternate Cloud Console project ID to what you initialized as the default in the gcloud tool. Example: --project [YOUR_PROJECT_ID]
Include the -v flag to specify a version ID, otherwise one is generated for you.
Example: -v [YOUR_VERSION_ID

**Viewing your application**

To launch your browser and view the app at https://**PROJECT_ID**.**REGION_ID** (#appengine-urls).r.appspot.com, run the following command
ud app browse

**RESULT:**

Thus, the GAE launcher to launch the web applications is launched successfully.

**Ex.No.5 SIMULATE A CLOUD SCENARIO USING CLOUDSIM AND RUN A SCHEDULING ALGORITHM THAT IS NOT PRESENT IN CLOUDSIM**

**Date:**

**AIM:**

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

**PROCEDURE:**

**What is Cloudsim?**

CloudSim is a simulation toolkit that supports the modelling and simulation of the core functionality of cloud, like job/task queue, processing of events, creation of cloud entities (datacentre, datacentre brokers, etc), communication between different entities, implementation of broker policies, etc. This toolkit allows to:

- Test application services in a repeatable and controllable environment.
- Tune the system bottlenecks before deploying apps in an actual cloud.
- Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques

**Core features of CloudSim are:**

- The Support of modelling and simulation of large scale computing environment as federated cloud data centres, virtualized server hosts, with customizable policies for provisioning host resources to virtual machines and energy-aware computational resources
- It is a self-contained platform for modelling cloud's service brokers, provisioning, and allocation policies.
- It supports the simulation of network connections among simulated system elements.
- Support for simulation of federated cloud environment, that inter-networks resources from both private and public domains.
- Availability of a virtualization engine that aids in the creation and management of multiple independent and co-hosted virtual services on a data centre node.
- Flexibility to switch between space shared and time-shared allocation of processing cores to virtualized services.

```
import java.text.DecimalFormat;
import java.util.Calendar;
import java.util.List;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.core.CloudSim;
```

```
public class FCFS {
/** The cloudlet list. */
private static List<Cloudlet> cloudletList;
/** The vmlist. */
private static List<Vm> vmlist;
private static int reqTasks = 5;
private static int reqVms = 2;
/**
* Creates main() to run this example
*/
public static void main(String[] args) {
Log.printLine("Starting FCFS...");
try {
// First step: Initialize the CloudSim package. It should be called
// before creating any entities.
int num_user = 1; // number of cloud users

Calendar calendar = Calendar.getInstance();
boolean trace_flag = false; // mean trace events

// Initialize the CloudSim library
CloudSim.init(num_user, calendar, trace_flag);

// Second step: Create Datacenters
@SuppressWarnings("unused")
Datacenter datacenter0 = createDatacenter("Datacenter_0");

//Third step: Create Broker
FcfsBroker broker = createBroker();
int brokerId = broker.getId();

//Fourth step: Create one virtual machine
vmlist = new VmsCreator().createRequiredVms(reqVms, brokerId);

//submit vm list to the broker
broker.submitVmList(vmlist);

//Fifth step: Create two Cloudlets
cloudletList = new CloudletCreator().createUserCloudlet(reqTasks, brokerId);

//submit cloudlet list to the broker
broker.submitCloudletList(cloudletList);

//call the scheduling function via the broker
broker.scheduleTaskstoVms();
```

```
// Sixth step: Starts the simulation
CloudSim.startSimulation();

// Final step: Print results when simulation is over
List<Cloudlet> newList = broker.getCloudletReceivedList();

CloudSim.stopSimulation();
printCloudletList(newList);
Log.printLine("FCFS finished!");
}

catch (Exception e) {
e.printStackTrace();
Log.printLine("The simulation has been terminated due to an unexpected error");
}
}

private static Datacenter createDatacenter(String name){
Datacenter datacenter=new DataCenterCreator().createUserDatacenter(name, reqVms);
return datacenter;
}

private static FcfsBroker createBroker(){
FcfsBroker broker = null;
try {
broker = new FcfsBroker("Broker");
}
catch (Exception e) {
e.printStackTrace();
return null;
}

return broker;

private static void printCloudletList(List<Cloudlet> list) {
int size = list.size();
Cloudlet cloudlet;
String indent = " ";
Log.printLine();
Log.printLine("========== OUTPUT ==========");
Log.printLine("Cloudlet ID" + indent + "STATUS" + indent +
"Data center ID" + indent + "VM ID" + indent + "Time" + indent + "Start Time" + indent + "Finish
Time");
DecimalFormat dft = new DecimalFormat("###.##");
for (int i = 0; i < size; i++) {
cloudlet = list.get(i);
Log.print(indent + cloudlet.getCloudletId() + indent + indent);
```

```
if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS){
Log.print("SUCCESS");
Log.printLine( indent + indent + cloudlet.getResourceId() + indent + indent + indent +
cloudlet.getVmId() +
indent + indent + dft.format(cloudlet.getActualCPUTime()) + indent + indent +
dft.format(cloudlet.getExecStartTime())+
indent + indent + dft.format(cloudlet.getFinishTime()));
}
}
}
}
```

**RESULT:**

Thus, the Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim run successfully.

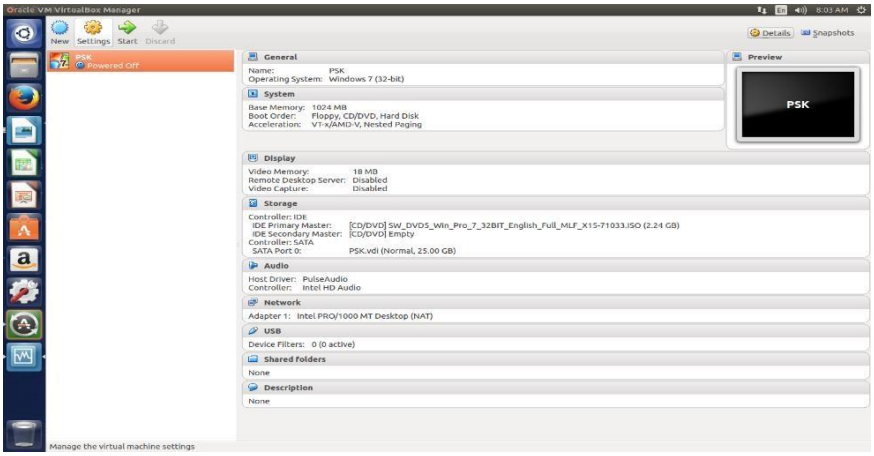
**Ex.No.6** **CREATION OF VIRTUAL MACHINES**
**Date:**

**AIM:**

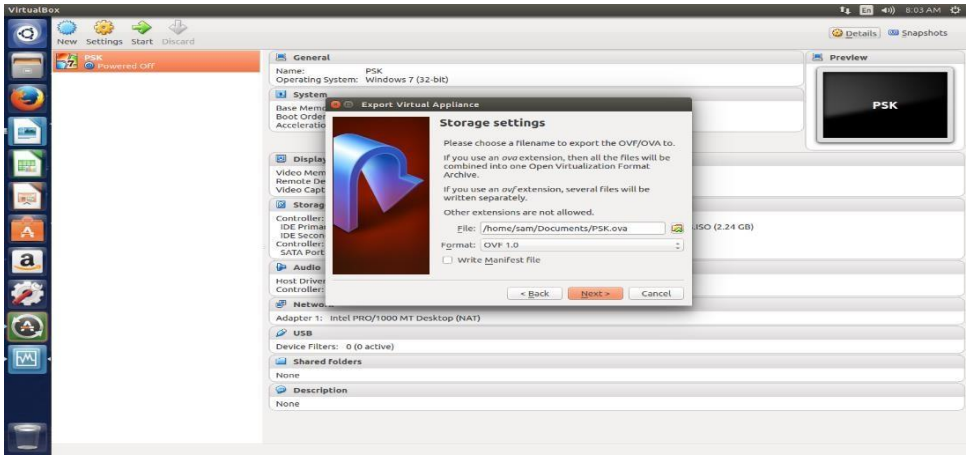
To transfer the files from one virtual machine to another virtual machine.

**PROCEDURE:**

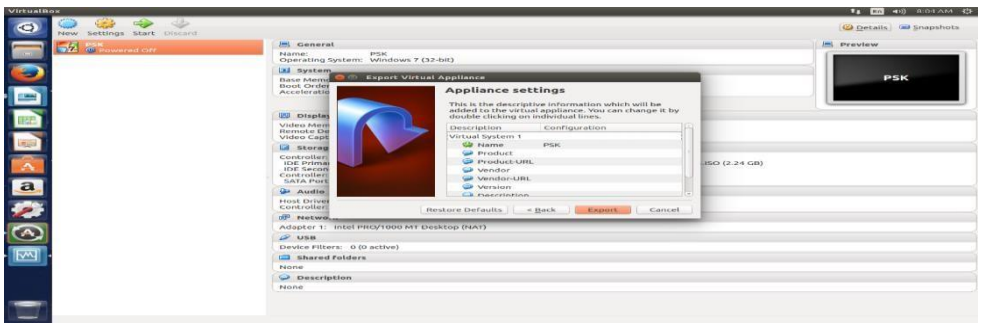
1. Select the VM and click File->Export Appliance



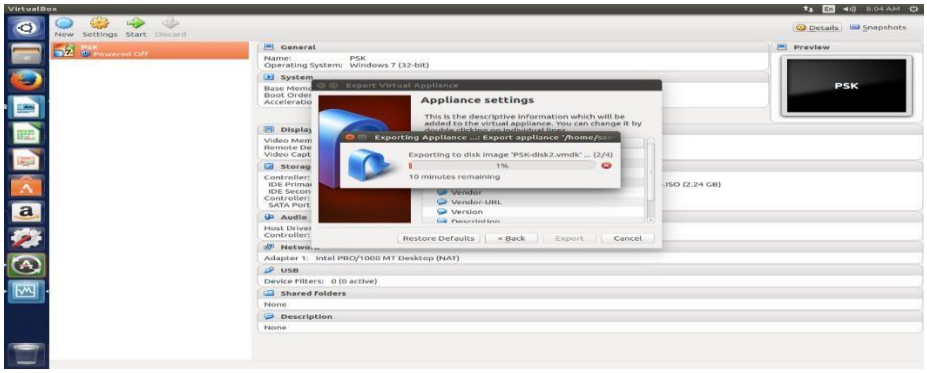

2. Select the VM to be exported and click NEXT.
3. Note the file path and click “Next”



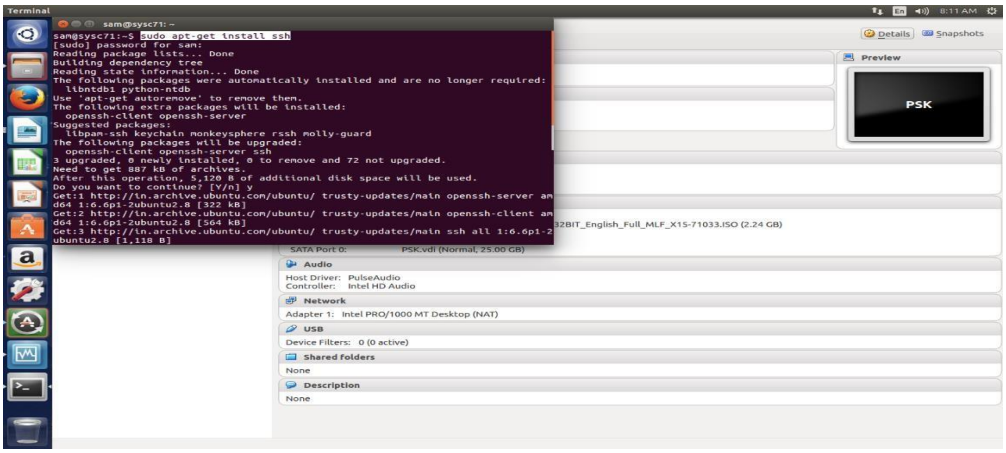

4. Click “Export”



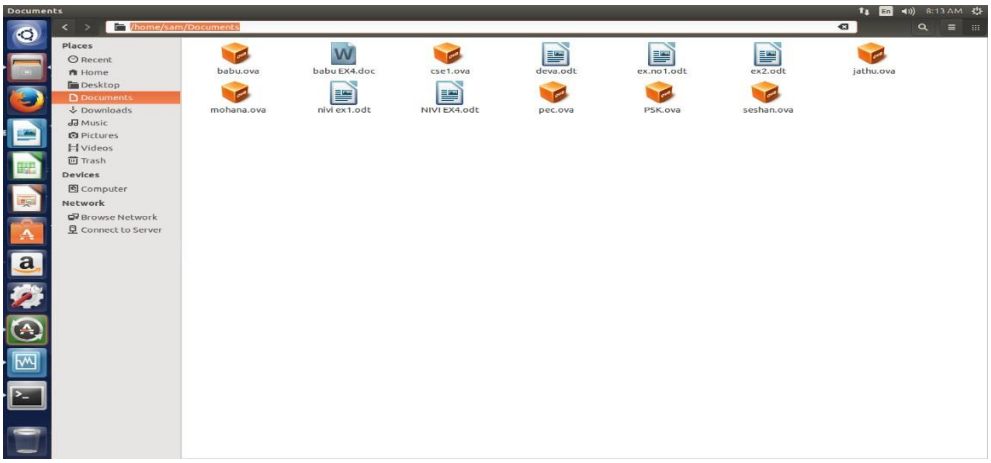
5. The Virtual machine is being exported.




6. Install “ssh” to access the neighbour's VM.

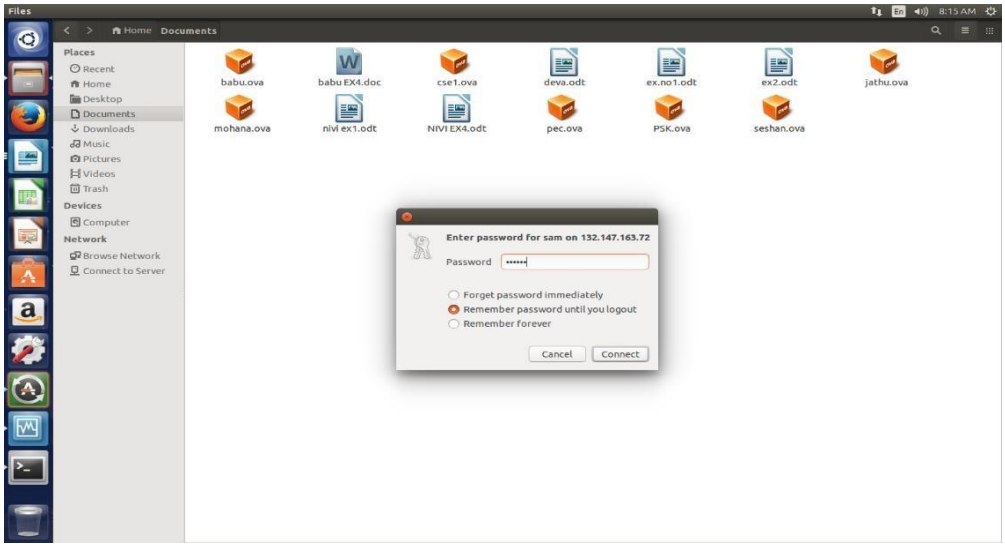



7. Go to File->Computer:/home/sam/Documents/

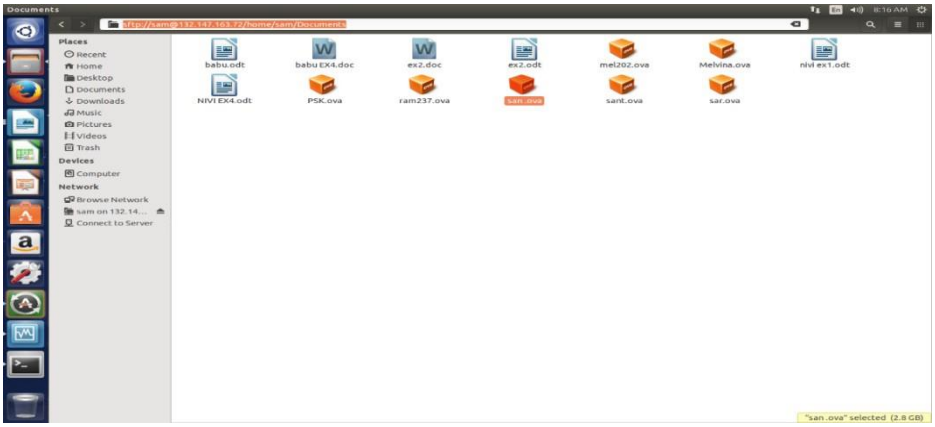


8. Type the neighbour's URL: sftp://sam@172.16.42._/

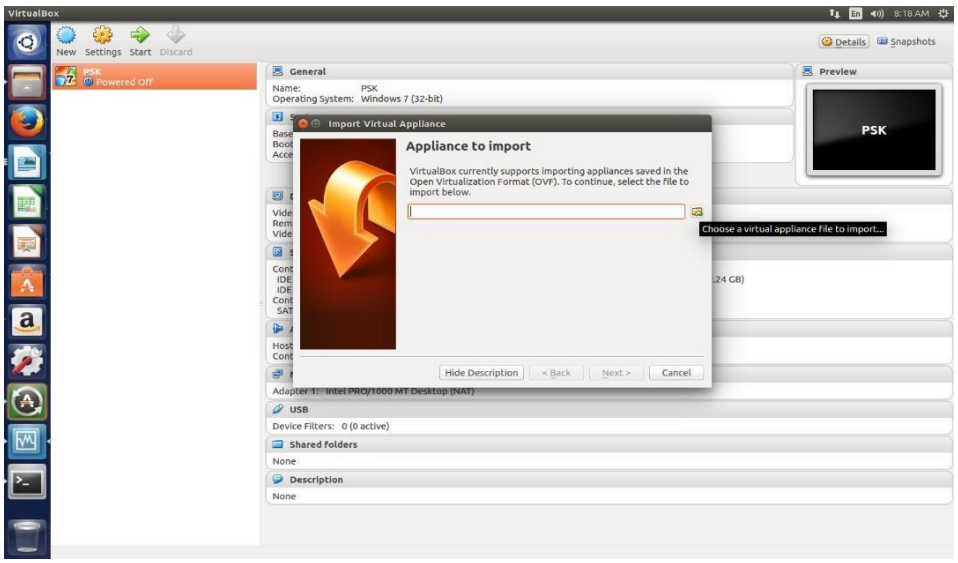
9. Give the password (sam123) and get connected.



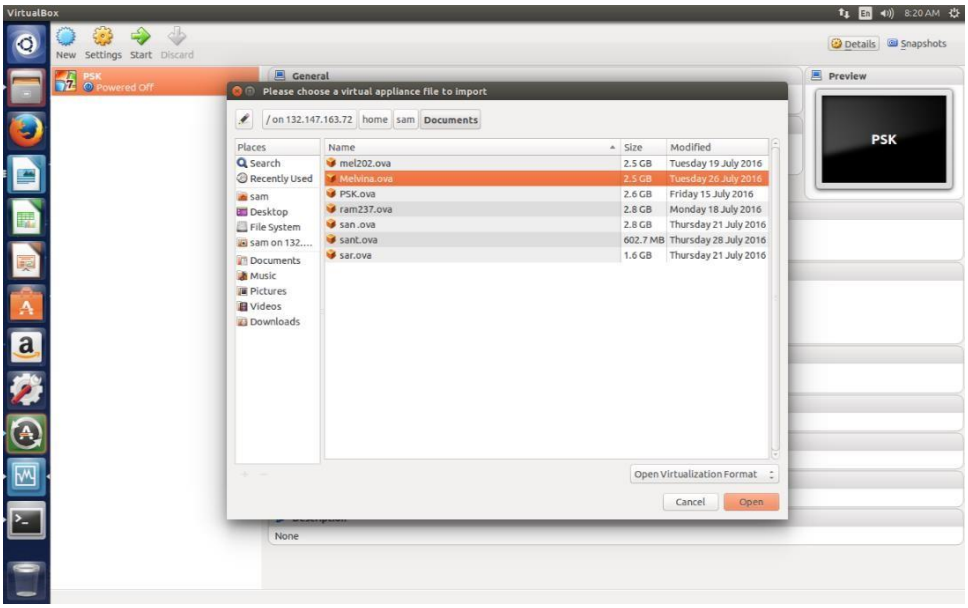

10. Select the VM and copy it in desktop.



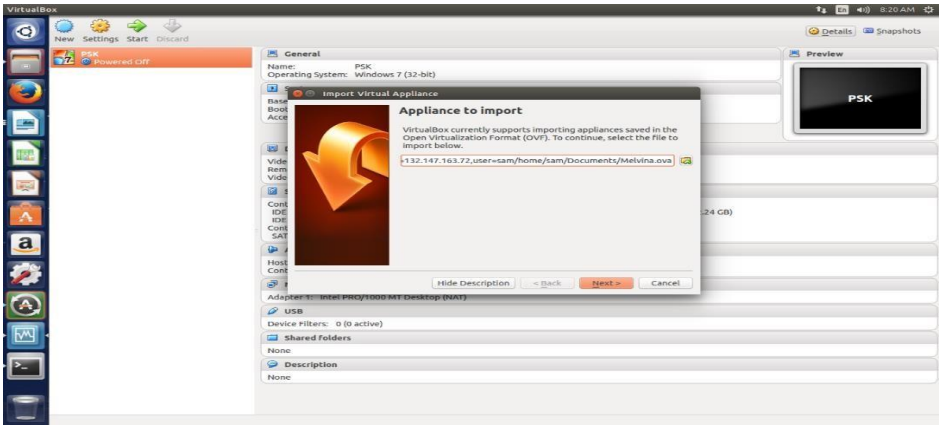

11 .Open VirtualBox and select File->Import Appliance->Browse



12. Select the VM to be imported and click “Open”.

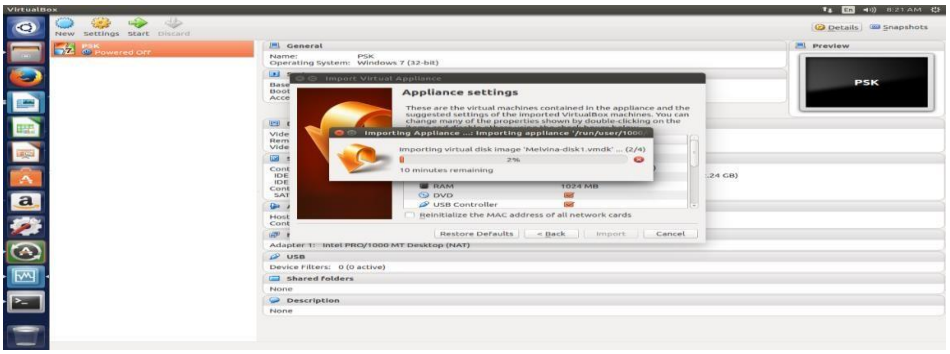



13. Click “Next”

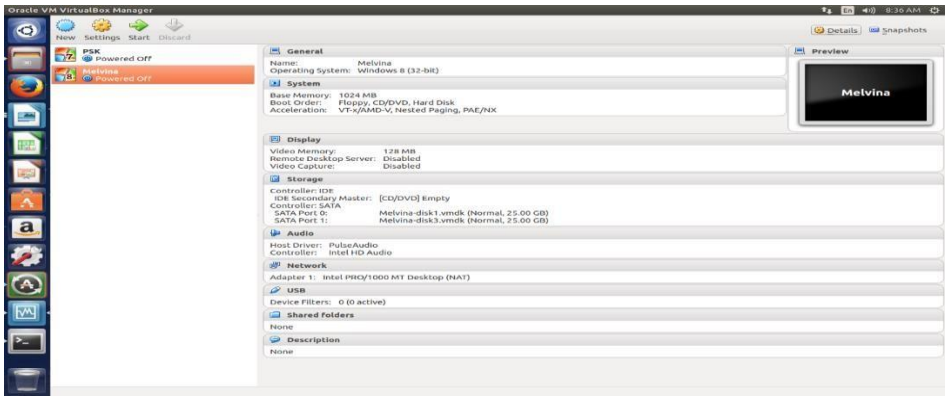



14. Click “Import”.

15. VM is being imported.




16. VM is imported.




**RESULT:**

Thus, Virtual machine migration has been implemented.

**Ex.No.7 FIND A PROCEDURE TO LAUNCH VIRTUAL MACHINE USING TRYSTACK**
**Date:**

**AIM:**

To launch virtual machine using trystack.

**PROCEDURE:**

https://cyberpersons.com/2017/06/22/how-to-install-openstack-and-create-your-first-virtual-machineinstance/

OpenStack is basically a cloud operating system. That let you deploy public and private clouds and take care of all the things for you. In this article, we will see that how to install OpenStack and create your first virtual machine or in other words launch your first instance.

We will perform our installation on Ubuntu 16.04 Server flavor because as mentioned on OpenStack site, they have done extensive testing with this flavor and this operating system is best suited for OpenStack

Minimum requirements for OpenStack is listed below:

- 4 GB Of Ram.
- 4 CPU Units.
- 30 GB Disk Space.

Once you have a virtual machine installed with mentioned version of Ubuntu, you are ready to install OpenStack and take it for a spin.

**Step 1: Prepare the environment for installing OpenStack!**

There are few things that need to be done before we start installing OpenStack. Just pleaserun the following commands and you will be done with it:

```
sudo apt-get update sudo
apt-get upgrade
sudo apt-get dist-upgradesudo
apt-get install git -y sudo
reboot
```

These commands will just bring all your packages to the latest version and install git so that we can clone OpenStack to our Linux machine.

**Step 2: Download and Install OpenStack!**

Note: If you already have a “stack” user on your virtual machine or laptop (with sudoprivileges), then you do not need to create an additional user

After your virtual machine is done with a reboot, you are now ready to install OpenStack. Normally OpenStack runs under non-root user with sudo privileges. We can easily create one to start with using:

```
sudo useradd -s /bin/bash -d /opt/stack -m stack
```

We’ve just created a user “stack” on our Linux machine, now let us give this user sudoprivileges using:

**echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack**

This will give the stack user sudo privileges. We now have to log in as user “stack” to proceed with our installation, that can be done using the command:

sudo su – stack

Now you are logged in as user “stack”, let’s start with the installation of OpenStack by downloading the required material.

git clone https://git.openstack.org/openstack-dev/devstack# cd to the cloned directory

**cd devstack**

Normally during installing it will ask you to set various passwords, you can automate thisprocess by creating a file in your current directory named “local.conf”.

# create the filenano

local.conf

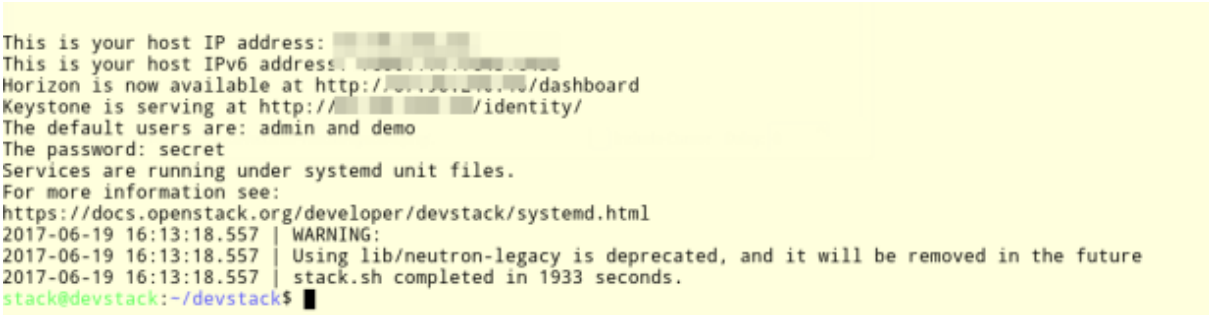
**# Now paste following contents in the file**

[[local|localrc]] ADMIN_PASSWORD=secret
**DATABASE_PASSWORD=$ADMIN_PASSWORD**
**RABBIT_PASSWORD=$ADMIN_PASSWORD**
**SERVICE_PASSWORD=$ADMIN_PASSWORD**

Save and exit the file, this will automate the installation process. We are now ready to runthe installation script. Installation script can be launched using the command:

**./stack.sh**

Depending on your hardware and internet connection installation can take from 10-20 minutes, once installation is complete you will have something like this on your console (atthe end of the installation):



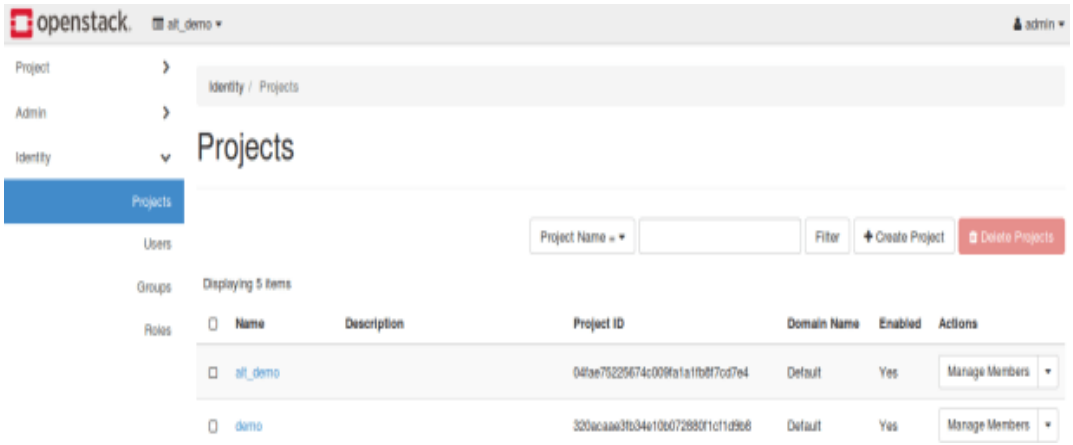

OpenStack dashboard can now be accessed at : http://<IPAddress>/dashboard/.....We will now see that how we can launch our very first and basic instance inside the cloudwe just created.

**Step 3: Create Network!**

Note: Please remember that without having a network, you can not launch an

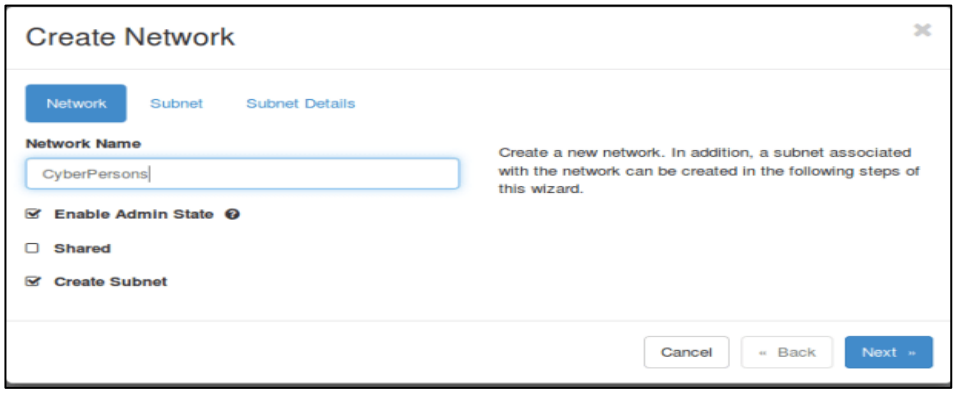
instance/virtual machine.

After you are logged into the OpenStack Dashboard it will look something like this:



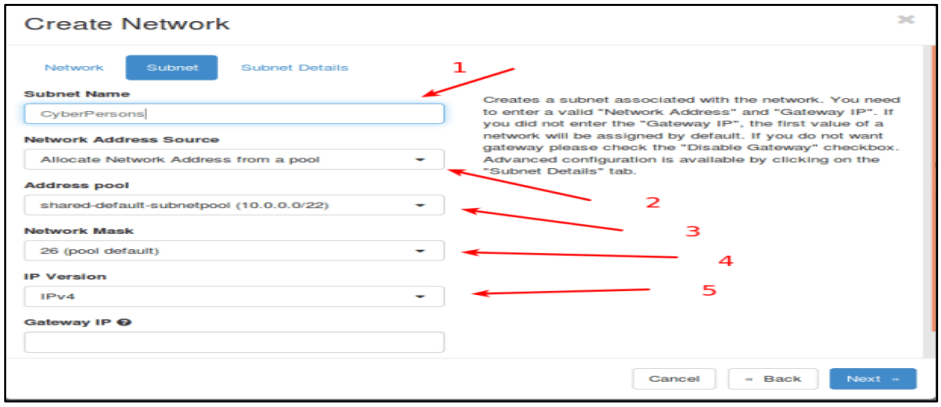

Before we launch our first virtual machine, we need to create a network that virtual machinecan use. For now, it will just be a dummy network, because our main purpose in this article is to launch our first virtual machine or instance. Let see how we can create a network inside OpenStack.


1. Click the Project Drop Down.
2. Click the Network Drop Down.
3. From network Drop Down select Networks, and this window will open that you see on the right side.
4. Finally, click Create Network. Something like this will pop-up:



Now on this window, you have to create a subnet, your settings should look like as theyare in the image above. Except for the “Subnet Name” box all of these are drop downs and you can choose from the given options

1. First, give your subnet a suitable name.
2. “**Network Address Source**” is a dropdown, it has two available options. You can selectthe second as chosen in the image above.
3. “**Address pool**” also have two possible option, select the first option.
4. “**Network Mask**” have many options in the drop down, you can keep the default which is 26.



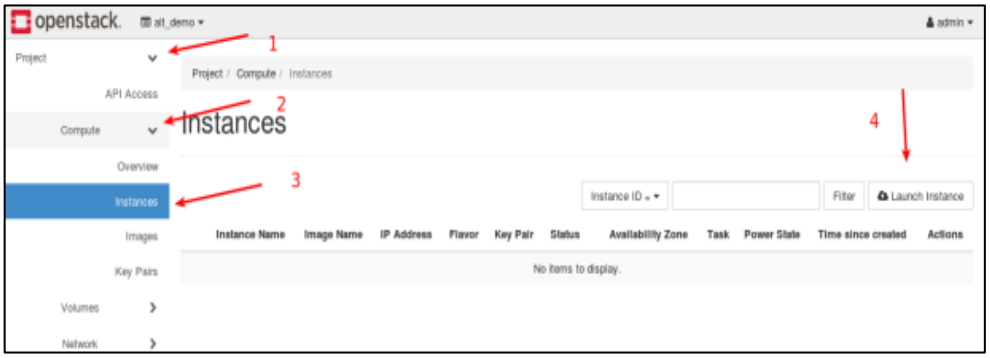

5. IP version should be IPv4.

Once all these things are done, click “Next”.


Now everything is optional in this window if you are interested in filling something up, you can. Otherwise, leave everything as it is and click “Create”. You now have a network that you can use to launch a virtual machine.

**Step 4: Create Virtual Machine/Instance**

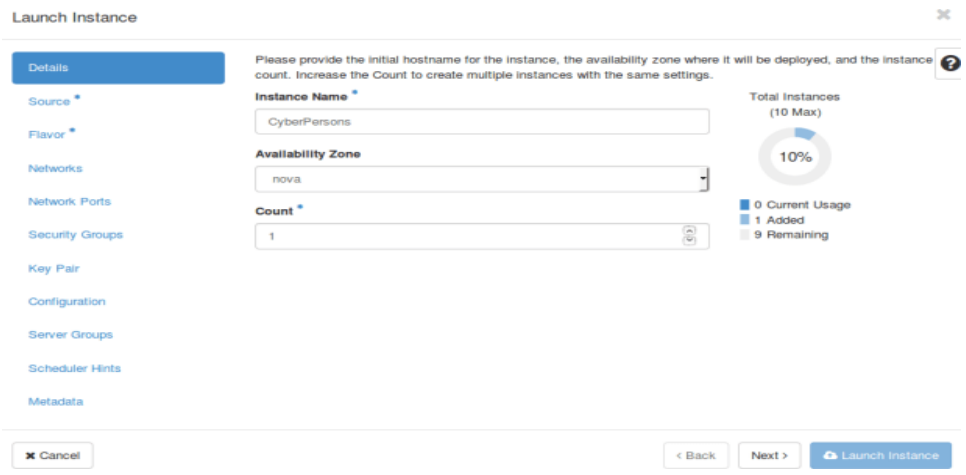
After the network is created, we are now ready to create our very first virtual machine.




1. Click on “Project” drop down.
2. Inside project click “Compute” drop down.
3. Under compute you have four options, since we are interested in creating an instance,you have to click on “Instance”.

Finally, click “Launch Instance”

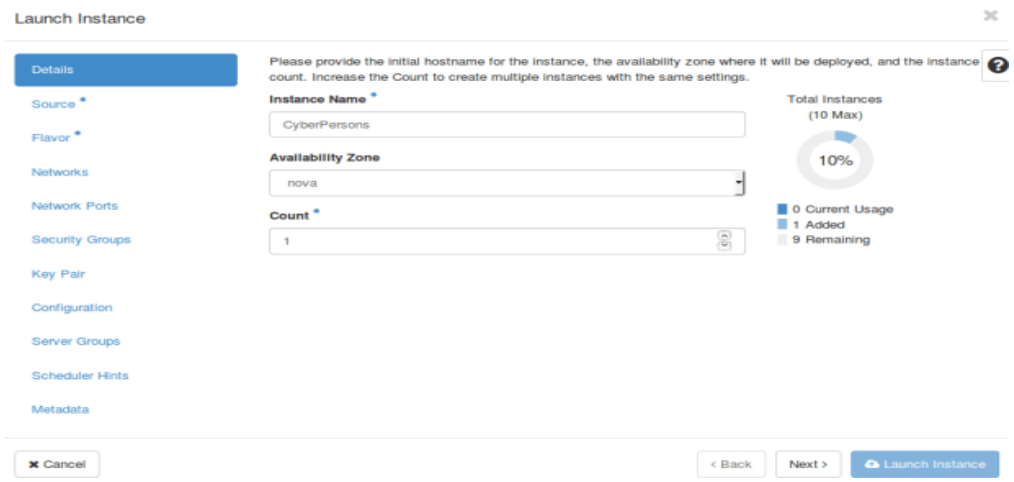
Something like this will pop-up.




Now, there are 11 tabs to create an instance, we will go through each tab one by one.

Details Tab

This is a general information tab for creating an instance, you will have to assign a name to your virtual machine on this tab. Select zone to launch a virtual machine, and tell how many copies of virtual machine you want. Just make sure your settings look like this:



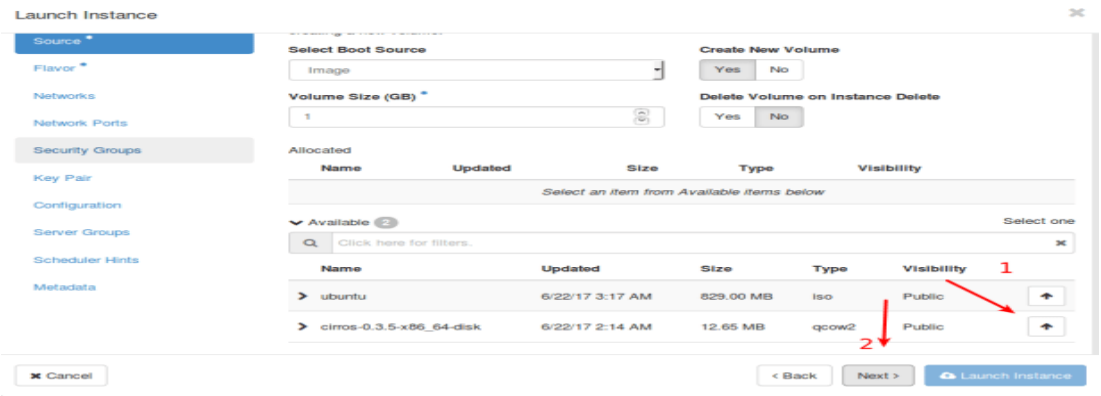
Source Tab
Normally when we create a virtual machine on Proxmox or VMWare we need to insert CD-ROM (virtual CD room), that VPS uses to install the operating system.

In OpenStack this is done by Source Tab, you can use various ways to launch a new virtual machine, OpenStack allows you to choose following as a source to create your instance.

- Image
- Snapshot of already created instance

Volume or a volume Snapshot

We are going to use “Cirros” image to create our instance.

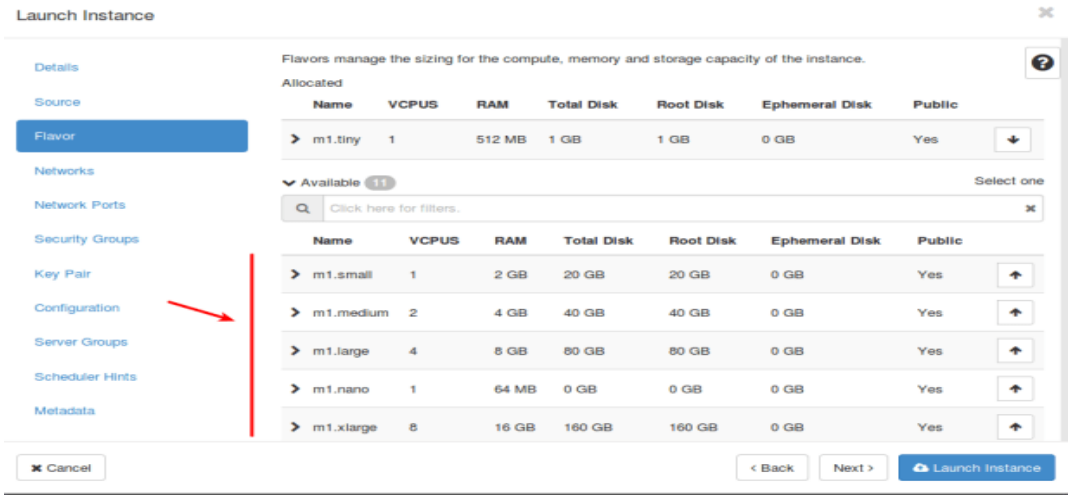



1. Click on the icon where the first arrow is pointing, so that we can use “Cirros” to launch our virtual machine.
2. After the image is selected, just click “Next” so that we can move to “Flavor” tab.

Flavor Tab
Flavor tab will allow you to allocate resource to your instance. Like:

- Ram.
- CPU.
- Disk Space.

It is similar to giving virtual resources to the virtual machine, but OpenStack gives fancynames to everything. ☺




You can see that there are 11 available pre-configured templates to choose from. The one I choose gave following resources to the instance:

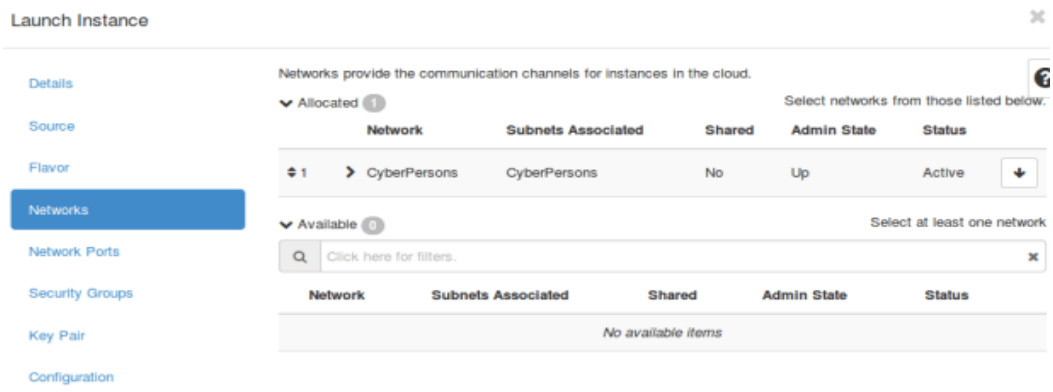
- 1 virtual CPU
- 512 MB Ram
- 1 GB Disk.

You can choose any template from the available 11, depending upon the resources available on your host machine (On which OpenStack is installed).

After flavor is selected, just press “Next”.

Network Tab

Network tab allows us to define a network for our virtual machine, you might have remembered that we’ve created a network above for this purpose. Now by default, the network you have created above will be selected for this machine, as seen in the imagebelow:




Don’t change anything just click “Next”.

**Network Ports Tab**

For now, just leave the default settings on “Network Ports” tab and click next.

**Security Groups Tab**

Security groups define how a specific virtual machine is allowed to talk with the outer world.As for now, we are just trying to create our first virtual machine, you can leave all the defaults. In a later article, we will see that how we can define “Security Groups” depending upon our requirements.

**Key-Pair Tab**

Leave defaults and click Next.

**Configuration Tab**

Leave defaults and click Next.

**Server Groups Tab**

Leave defaults and click Next.

**Scheduler Hints Tab**

Leave defaults and click Next.

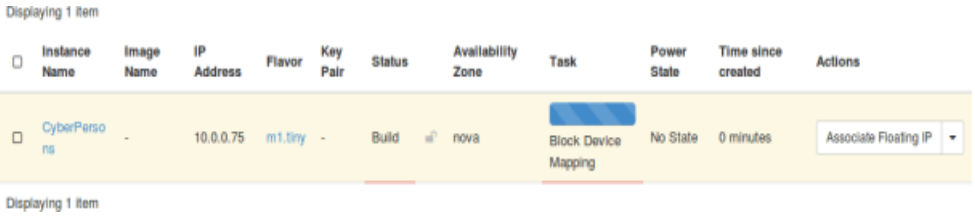
**Metadata Tab**

Leave defaults and click Next.

**Launch Instance**

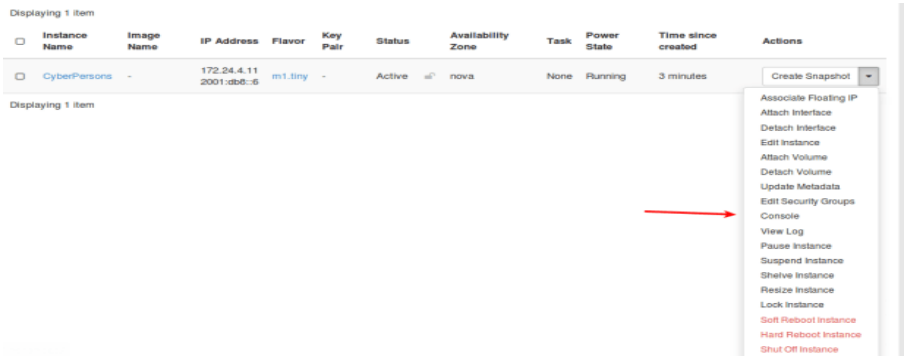
After going through all the tabs, you are now ready to press that magic “Launch Instance” button, you must be wondering that for some tabs we’ve left them with default settings. Its fine for now, because we are just launching our test machine at this time. Later we will go into depth of each tab and see why it is important and why it is not.

Once you click “Launch Instance” button, OpenStack will start creating our virtual machine, and it is going to look something like this:

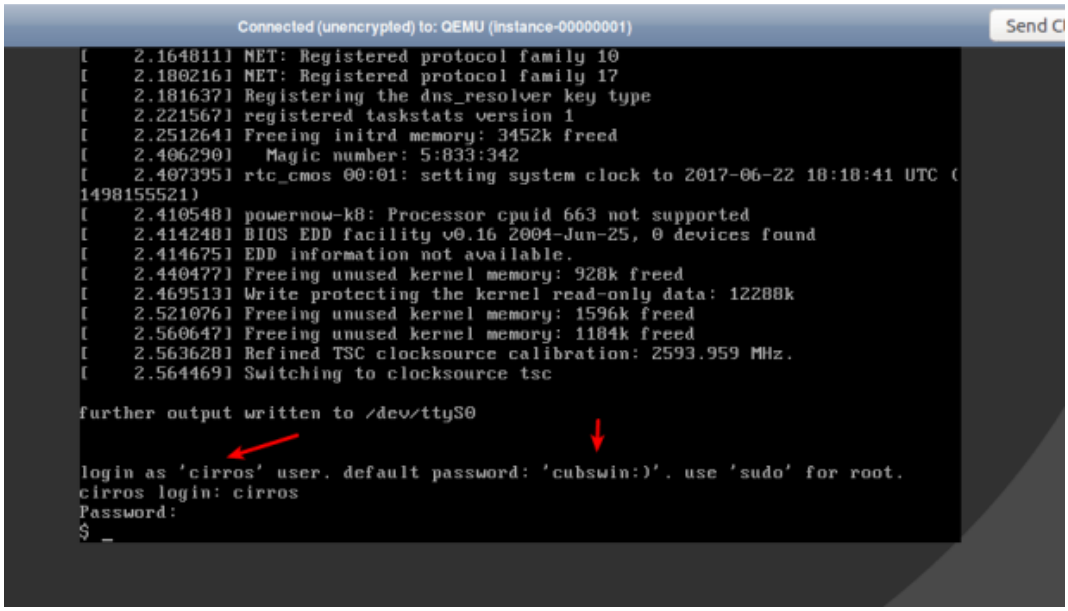



**Step 5: Access Virtual Machine Console!**

Once you click “Launch Instance” it will take OpenStack few seconds to create your virtual machine. Once ready you can access the console to see how the command line of your first virtual machine looks like




Click on “Console” and OpenStack will take you to the console of the virtual machine. The console will look something like this



This is simple command line, you can use following details to log in:Username:

cirros

Password: cubswin:)

Congratulations, you have successfully created your first virtual machine/instance on OpenStack.

**RESULT:**

Thus, successfully created your first virtual machine/instance on OpenStack.

**Ex.No.8** **INSTALL HADOOP SINGLE NODE CLUSTER AND RUN SIMPLE APPLICATIONS LIKE WORDCOUNT.**

**Date:**

**AIM:**

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

**PROCEDURE:**

sam@sysc40:~$ **sudo apt-get update**
sam@sysc40:~$ **sudo apt-get install default-jdk**
sam@sysc40:~$ **java -version**
openjdk version "1.8.0_131"
OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-0ubuntu1.16.04.2-b11)
OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)

sam@sysc40:~$ **sudo addgroup hadoop**
Adding group `hadoop' (GID 1002) ...
Done.

sam@sysc40:~$ sudo adduser --ingroup hadoop hduser
Adding user `hduser' ...
Adding new user `hduser' (1002) with group `hadoop' ...
Creating home directory `/home/hduser' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: \\Note: Enter any password and remember that, this is only for unix(applicable for hduser)
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser
Enter the new value, or press ENTER for the default \\Note: Just enter your name and then click enter button for remaining
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] y

sam@sysc40:~$ **groups hduser**
hduser : hadoop

sam@sysc40:~$ **sudo apt-get install ssh**
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed: ssh
0 upgraded, 1 newly installed, 0 to remove and 139 not upgraded.
Need to get 7,076 B of archives.
After this operation, 99.3 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu xenial-updates/main amd64 ssh all 1:7.2p2-4ubuntu2.2 [7,076 B]
Fetched 7,076 B in 0s (16.2 kB/s)
Selecting previously unselected package ssh.

(Reading database ... 233704 files and directories currently installed.)
Preparing to unpack .../ssh_1%3a7.2p2-4ubuntu2.2_all.deb ...
Unpacking ssh (1:7.2p2-4ubuntu2.2) ...
Setting up ssh (1:7.2p2-4ubuntu2.2) ...

sam@sysc40:~$ **which ssh**
/usr/bin/ssh
sam@sysc40:~$ **which sshd**
/usr/sbin/sshd
sam@sysc40:~$ **su hduser**
Password
hduser@sysc40:/home/sam$
hduser@sysc40:/home/sam$ **cd**

hduser@sysc40:~$ **ssh-keygen -t rsa -P ""**
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa
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:
SHA256:QWYjqMI0g/ElhpXhVvgVITSn4O4HWS98MDqCX7Gsf/g hduser@sysc40
The key's randomart image is:
```
+---[RSA 2048]----+
|o+*=*.=o=        |
|oOo=.=.= .       |
|o Bo*.  .        |
|o+.*.*   .       |
|o.* * o S        |
| + = o           |
|  + ..           |
|   o. .          |
|    .oE          |
+----[SHA256]-----+
```

hduser@sysc40:~$ **cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys**
hduser@sysc40:~$ **ssh localhost**
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:+kILEX2sGtgsoPfCQ+Vw2cWHbbWGJt0qTEMu9tEvaX8.
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 16.04.2 LTS (GNU/Linux 4.8.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

143 packages can be updated.
15 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hduser@sysc40:~$ **wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.5/hadoop-2.6.5.tar.gz**
--2017-07-21 11:17:53-- http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.5/hadoop-2.6.5.tar.gz
Resolving mirrors.sonic.net (mirrors.sonic.net)... 69.12.162.27
Connecting to mirrors.sonic.net (mirrors.sonic.net)|69.12.162.27|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 199635269 (190M) [application/x-gzip]
Saving to: ‘hadoop-2.6.5.tar.gz.2’
hadoop-2.6.5.tar.gz 100%[==================>] 190.39M 180KB/s in 17m 4s
2017-07-21 11:34:58 (190 KB/s) - ‘hadoop-2.6.5.tar.gz.2’ saved [199635269/199635269]

hduser@sysc40:~$ **tar xvzf hadoop-2.6.5.tar.gz hadoop-2.6.5/**
hadoop-2.6.5/include/
hadoop-2.6.5/include/hdfs.h
hadoop-2.6.5/include/Pipes.hh
hadoop-2.6.5/include/TemplateFactory.hh
hadoop-2.6.5/include/SerialUtils.hh
hadoop-2.6.5/include/StringUtils.hh
hadoop-2.6.5/README.txt
hadoop-2.6.5/LICENSE.txt
---
---
hadoop-2.6.5/share/hadoop/tools/lib/jasper-compiler-5.5.23.jar
hadoop-2.6.5/share/hadoop/tools/lib/apacheds-kerberos-codec-2.0.0-M15.jar
hadoop-2.6.5/share/hadoop/tools/lib/aws-java-sdk-1.7.4.jar

hduser@sysc40:~$ **sudo mkdir -p /usr/local/hadoop**
[sudo] password for hduser:
hduser is not in the sudoers file. This incident will be reported.

hduser@sysc40:~$ **cd hadoop-2.6.5**
hduser@sysc40:~/hadoop-2.6.5$ **su sam**
Password: sam123

sam@sysc40:/home/hduser/hadoop-2.6.5$ **sudo adduser hduser sudo**
[sudo] password for sam:
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.

sam@sysc40:/home/hduser/hadoop-2.6.5$ **su hduser**
Password: \\Note: Enter the password that we have given above for hduser

hduser@sysc40:~/hadoop-2.6.5$ **sudo mkdir /usr/local/hadoop**
hduser@sysc40:~/hadoop-2.6.5$ **sudo mv * /usr/local/hadoop**
hduser@sysc40:~/hadoop-2.6.5$ **sudo chown -R hduser:hadoop /usr/local/hadoop**
hduser@sysc40:~/hadoop-2.6.5$ **cd**
hduser@sysc40:~$ **update-alternatives --config java**
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
Nothing to configure.

hduser@sysc40:~$ **nano ~/.bashrc**
Add the below content at the end of the file and save it
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-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@sysc40:~$ **source ~/.bashrc**
hduser@sysc40:~$ **javac -version**
javac 1.8.0_131
hduser@sysc40:~$ **which javac**
/usr/bin/javac
hduser@sysc40:~$ **readlink -f /usr/bin/javac**
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac

hduser@sysc40:~$ **nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh**
Add the below line at the end of the file
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

hduser@sysc40:~$ **sudo mkdir -p /app/hadoop/tmp**

hduser@sysc40:~$ **sudo chown hduser:hadoop /app/hadoop/tmp**
hduser@sysc40:~$ nano /usr/local/hadoop/etc/hadoop/core-site.xml
Add the below line inside the <configuration></configuration> tag.
<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>
```

hduser@sysc40:~$ **cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml**

hduser@sysc40:~$ **nano  /usr/local/hadoop/etc/hadoop/mapred-site.xml**

Add the below line inside 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>
```

hduser@sysc40:~$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode**

hduser@sysc40:~$ **sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode**

hduser@sysc40:~$ **sudo chown -R hduser:hadoop /usr/local/hadoop_store**

hduser@sysc40:~$ **nano /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>
```

hduser@sysc40:~$ **hadoop namenode -format**
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
16/11/10 13:07:15 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = laptop/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.5
...
...
...
16/11/10 13:07:23 INFO util.ExitUtil: Exiting with status 0
16/11/10 13:07:23 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at laptop/127.0.1.1
************************************************************/

**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@sysc40:~$ su sam
Password: sam123
sam@sysc40:/home/hduser$ cd
sam@sysc40:~$ cd /usr/local/hadoop/sbin
sam@sysc40:/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

sam@sysc40:/usr/local/hadoop/sbin$ **sudo su hduser**
[sudo] password for sam: sam123

**Start NameNode daemon and DataNode daemon:**
hduser@sysc40:/usr/local/hadoop/sbin$ **start-dfs.sh**
16/11/10 14:51:44 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-laptop.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-laptop.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.

ECDSA key fingerprint is SHA256:e9SM2INFNu8NhXKzdX9bOyKIKbMoUSK4dXKonloN7JY.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-laptop.out
16/11/10 14:52:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library f

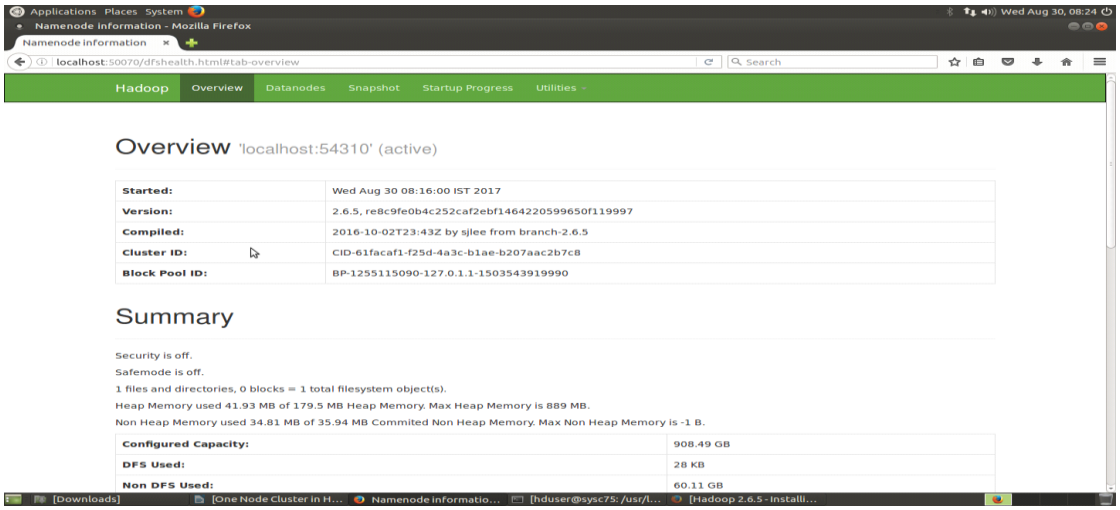
hduser@sysc40:/usr/local/hadoop/sbin$ **start-yarn.sh**
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-laptop.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-laptop.out

hduser@sysc70:/usr/local/hadoop/sbin$ **jps**
14306 DataNode14660 ResourceManager
14505 SecondaryNameNode
14205 NameNode
14765 NodeManager
15166 Jps

hduser@laptop:/usr/local/hadoop/sbin$ **stop-dfs.sh**
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
**stop-yarn.sh**

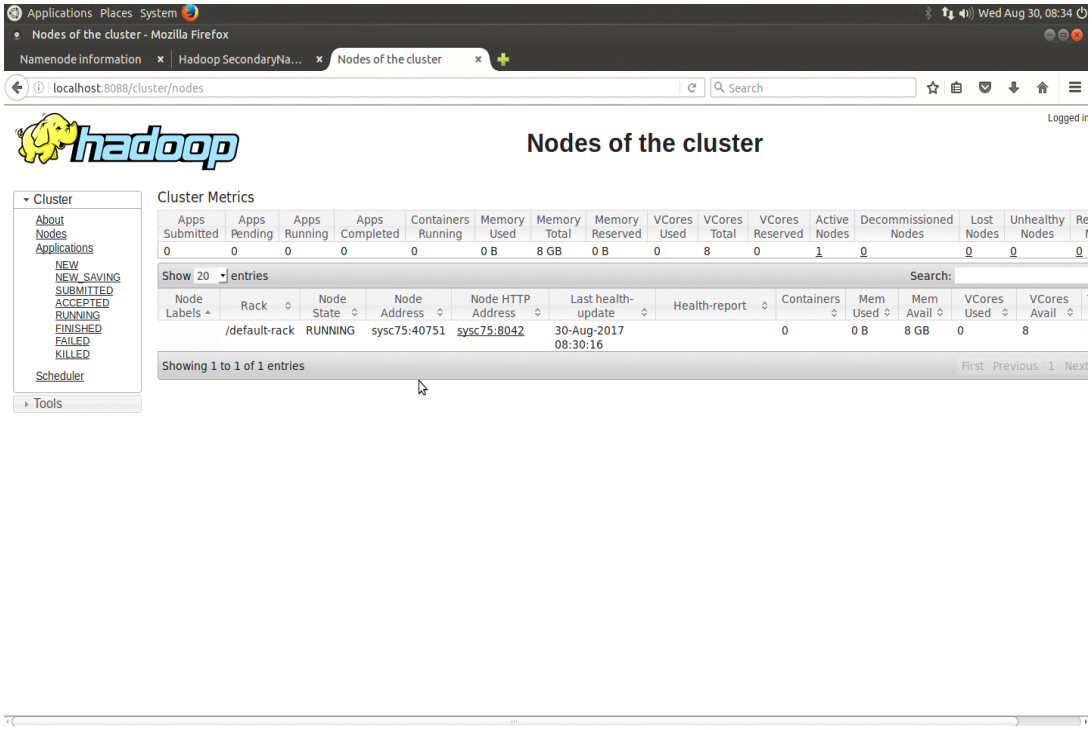
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
no proxyserver to stop

**Hadoop Web Interfaces**
hduser@laptop:/usr/local/hadoop/sbin$ **start-dfs.sh**
hduser@laptop:/usr/local/hadoop/sbin$ **start-yarn.sh**
Type **http://localhost:50070/** into our browser, then we'll see the web UI of the NameNode daemon:  In the Overview tab, you can see the Overview, Summary, NameNode Journal Status and the NameNode Storage informations.



Type in **http://localhost:50090/status.jsp** as url, we get **SecondaryNameNode**:


The default port number to access all the applications of cluster is 8088. Use the following url to visit **Resource Manager**: **http://localhost:8088/**
We need to click the Nodes option in the left Cluster panel, then it will sho the node that we have created.




**PROCEDURE FOR WORD COUNT:**

**Step 0:**
We need to check whether the hadoop dashboard has been created with all the six nodes.

sam@sysc65:~$ **su hduser**
Password:
hduser@sysc65:/home/sam$ **cd /usr/local/hadoop/sbin**
hduser@sysc65:/usr/local/hadoop/sbin$ **start-all.sh**
hduser@sysc65:/usr/local/hadoop/sbin$ **jps**
3797 NameNode
4279 ResourceManager
4120 SecondaryNameNode

3916 DataNode
4396 NodeManager
5486 Jps
hduser@sysc65:/usr/local/hadoop/sbin$
Type this command in Browser **“localhost:50070”**
We should get the dashboard for hadoop environment.

**Step 1**: sam@sysc65:~$ **su hduser**
Password:
**Step 2**:hduser@sysc65:/home/sam$ **cd**
**Step 3**: hduser@sysc65:~$ **cd /home/hduser**
**Step 4:** hduser@sysc65:~$ **nano WordCount.java**
Paste the program into that file and save it by Ctrl+o, Enter & Ctrl+x

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}

public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException
{
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
     Text outputKey = new Text(word.toUpperCase().trim());
  IntWritable outputValue = new IntWritable(1);
```

```
  con.write(outputKey, outputValue);
}
}
}

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable>values, Context con) throws IOException,
InterruptedException
{
int sum = 0;
   for(IntWritable value : values)
   {
   sum += value.get();
   }
   con.write(word, new IntWritable(sum));
}
}
}
```

**Step5:** hduser@sysc65:~$ **/usr/local/hadoop/bin/hadoop classpath**
/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/*:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/share/hadoop/hdfs:/usr/local/hadoop/share/hadoop/hdf

**Step 6:** Copy and paste the above classpath to compile the java program and make it as jar file.
hduser@sysc65:~$ **javac -cp "/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/*:/usr/local/hadoop/share/hadoop/common/*:/usr/local/hadoop/share/hadoop/hdfs:/usr/local/hadoop/share/hadoop/hdfs/lib/*:/usr/local/hadoop/share/hadoop/hdfs/*:/usr/local/hadoop/share/hadoop/yarn/lib/*:/usr/local/hadoop/share/hadoop/yarn/*:/usr/local/hadoop/share/hadoop/mapreduce/lib/*:/usr/local/hadoop/share/hadoop/mapreduce/*:/contrib/capacity-scheduler/*.jar" WordCount.java**
Note: WordCount.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

**Step 7:** hduser@sysc65:~$ **jar cvf MyJar.jar *.class**
added manifest
adding: WordCount.class(in = 1678) (out= 885)(deflated 47%)
adding: WordCount$MapForWordCount.class(in = 1788) (out= 752)(deflated 57%)
adding: WordCount$ReduceForWordCount.class(in = 1651) (out= 690)(deflated 58%)

**Step 8:** hduser@sysc65:~$ **nano wc.txt**
Paste the below lines into that file andd save it by Ctrl+o, Enter  & Ctrl+x
bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,bus,train,car,bUs,TrAiN,cAr,train,bus,bus

**Step 9:** hduser@sysc65:~$ **unset HADOOP_COMMON_HOME**
**Step 10:**  hduser@sysc65:~$ **/usr/local/hadoop/bin/hadoop fs -mkdir -p /home/hduser**

**Step 11:**
hduser@sysc65:~$ **/usr/local/hadoop/bin/hadoop fs -put /home/hduser/wc.txt /home/hduser/wc.txt**

17/08/23 13:11:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

**Step 12:**
hduser@sysc65:~$ **/usr/local/hadoop/bin/hadoop jar /home/hduser/MyJar.jar WordCount /home/hduser/wc.txt /home/hduser/MRDir1**
17/08/23 13:15:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/08/23 13:15:17 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
Map-Reduce Framework
Map input records=1
Map output records=69
Map output bytes=598
Map output materialized bytes=742
Merged Map outputs=1
GC time elapsed (ms)=77
CPU time spent (ms)=0
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=446169088
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=322
File Output Format Counters
Bytes Written=23

**Step 13:** hduser@sysc65:~$ **/usr/local/hadoop/bin/hadoop fs -ls /home/hduser/MRDir1**
Found 2 items
-rw-r--r-- 1 hduser supergroup 0 2017-08-23 13:15 /home/hduser/MRDir1/_SUCCESS
-rw-r--r-- 1 hduser supergroup 23 2017-08-23 13:15 /home/hduser/MRDir1/part-r-00000

Step 14:
hduser@sysc65:~$ **/usr/local/hadoop/bin/hadoop fs -cat /home/hduser/MRDir1/part-r-00000**
17/08/23 13:15:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
BUS 24
CAR 22
TRAIN 23

**RESULT:**

Thus, Hadoop single node cluster has been installed and word count application is executed and verified successfully.

.

**Ex.No.9** **INSTALLING C/GCC COMPILER FOR WINDOWS**
**Date:**

**AIM:**

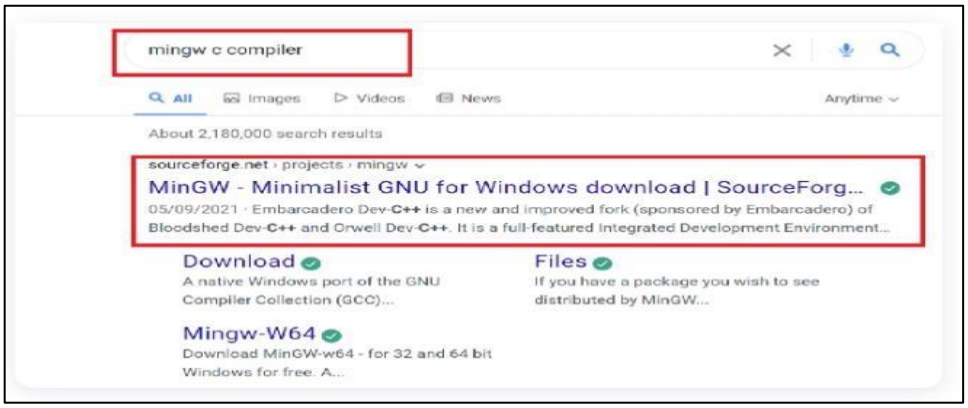
To install C/GCC compiler in windows.

**PROCEDURE:**

Following are the steps to download and install the MinGW GCC Compiler for windows.

**Step 1: Search MinGW C Compiler on the Web**

To download the MinGW compiler, go to your favorite browser and search MinGW C Compiler or click on the sourceforge.net link.

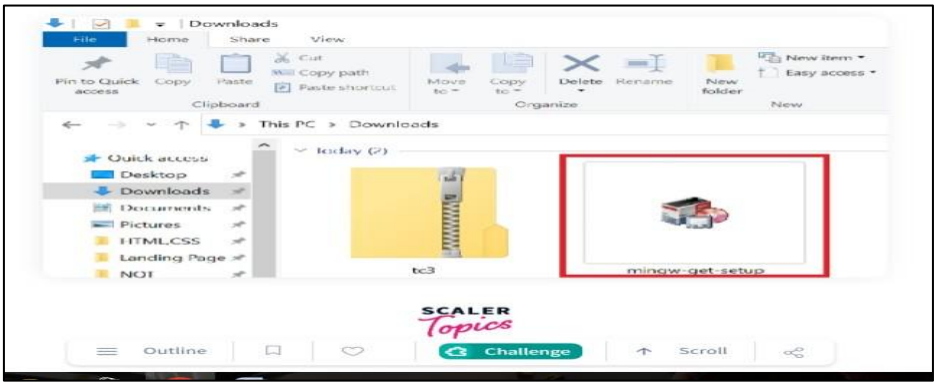



**Step 2: Download MinGW.**

After clicking on the green-colored download button on the website, the MinGW setup file will start downloading.

**Step 3: Locate the MinGW-get-setup.exe File and Start Installation.**

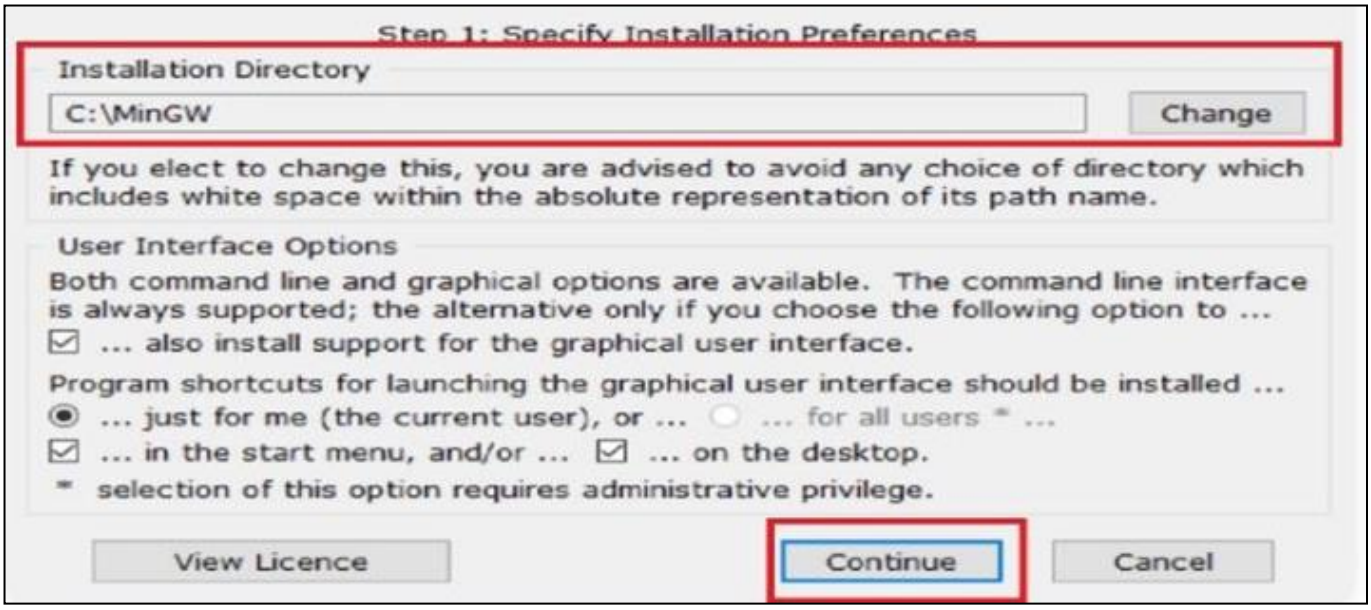
Locate the setup.exe file on your Downloads folder and double-click on it.




After double-clicking on the setup file, MinGW Installation Manager Setup Tool will now open. It will show the information like version, name, etc. Click on the Install button and proceed to start the installation.

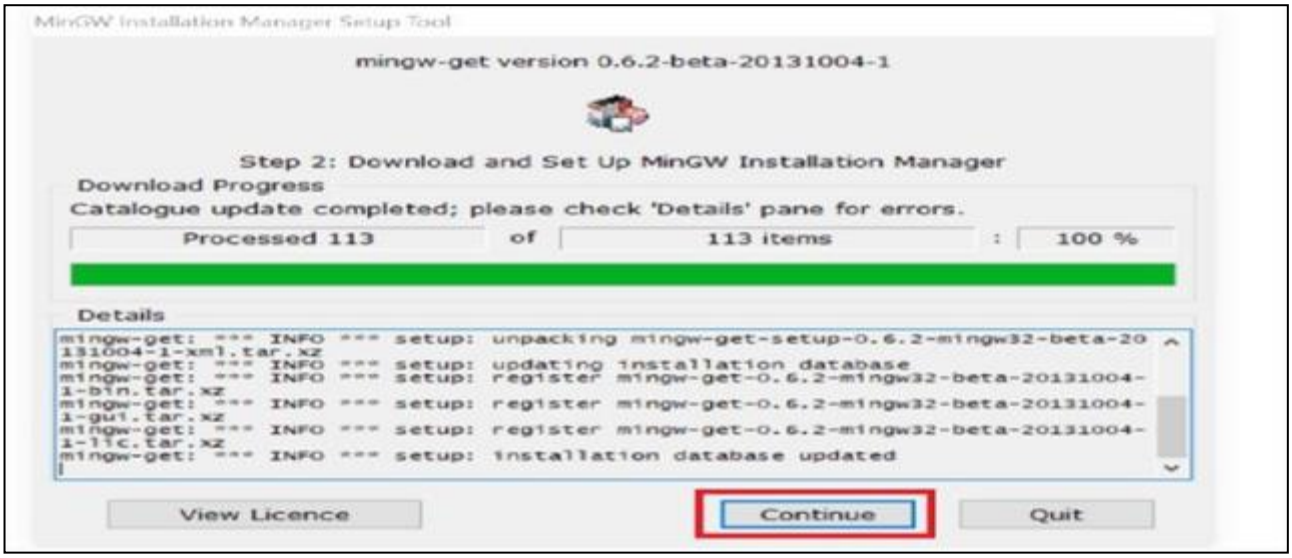
**Step 4: Specify Installation Preferences.**

Now the installation manager will ask you to specify the installation preferences. For that, you will be asked to choose the installation directory. If you wish to change it, you can browse the explorer and specify the location as per your requirement. After that, click on continue to proceed further.




**Step 5: Download and Set up MinGW Installation Manager.**

The installer will now automatically download the required files for MinGW to install on your Windows system. Grab a cup of coffee and wait patiently till the installation manager finishes downloading all the files. When it is done, click on continue to proceed ahead.



**Step 6: Select Packages Required for the Compiler.**

There are three packages required for the basic MinGW setup that you have to choose from the MinGW Installation Manager.

**1. MinGW32-base Package.**

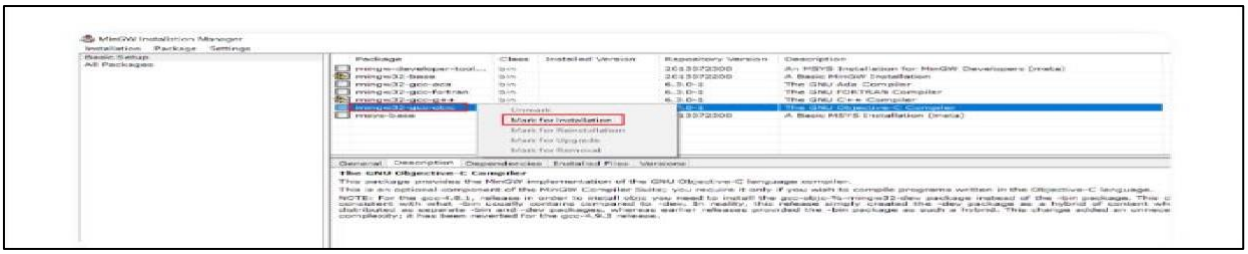
First, you have to install the *MinGW32-base package*. This package is used to compile the C program, including linker and other binary tools. Right-click on the MinGW32-base option and select Mark for Installation.


2. **MinGW32-gcc-g++ Package.**

Now you have to install the MinGW32-gcc-g++ package. This package is used to compile C++ source code. This is an optional component of the MinGW Compiler. It is only required if you are going to program in C++ language only. To select the MinGW32-gcc-g++ right-click on it and select Mark for Installation.

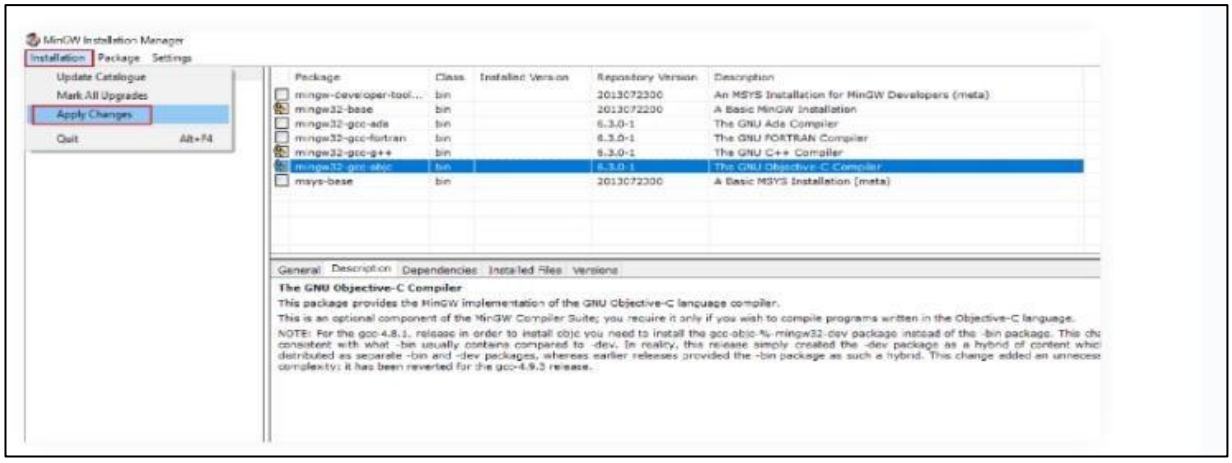
3. **MinGW32-gcc-objc package.**

At last, you have to install the MinGW32-gcc-objc package. This package is used to compile objective C language. It is an optional component. It is only required if you are going to program in objective C. To select the MinGW32-gcc-objc package, right-click on it and select *Mark for Installation.*



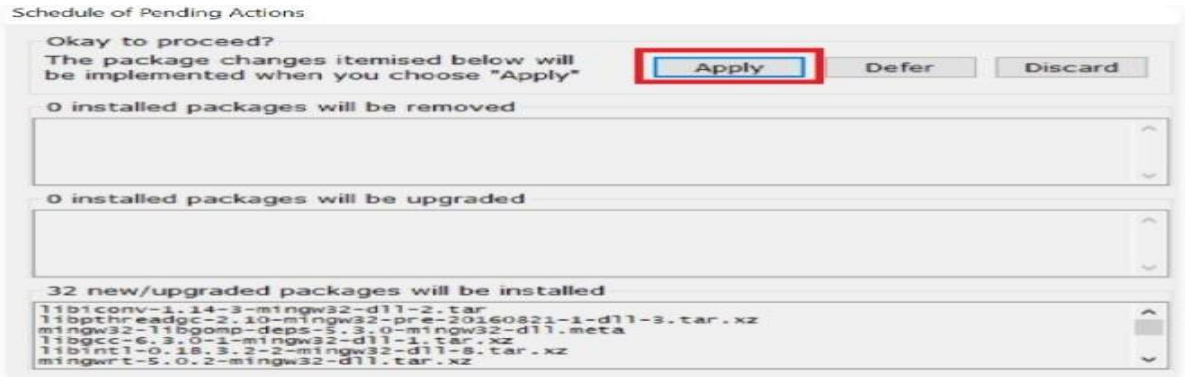
**Step 7: Apply the Changes**

After selecting all the required packages, go to Installation>>Apply Changes and click on Apply Changes




**Step 8: Download the Changes.**

Now it is time to download all the packages you selected in the previous step. Click on Apply and proceed further to download and install them.




**Step 9: Installation Completed.**

Now the installation has been completed, click on Close to close the Installation manager.

**Setting up Path Variable**

To set up the path for the C compiler for windows, follows the below steps :

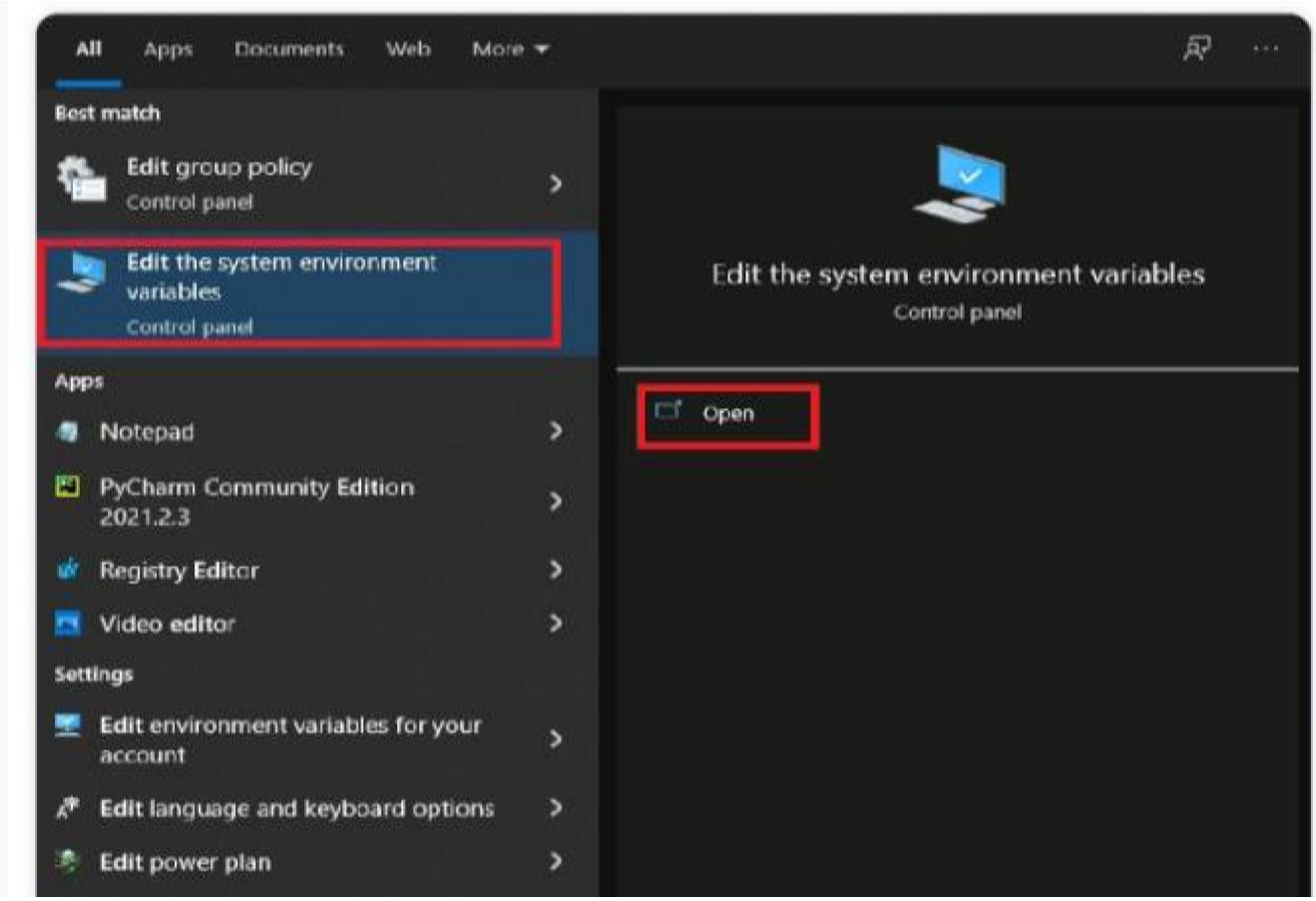
**Step 1: Copy the path of the MinGW bin.**

When you install the MinGW, it creates a folder named MinGW in C: Drive. To set the compiler's path, we need the path to the bin directory of MinGW. So, first,

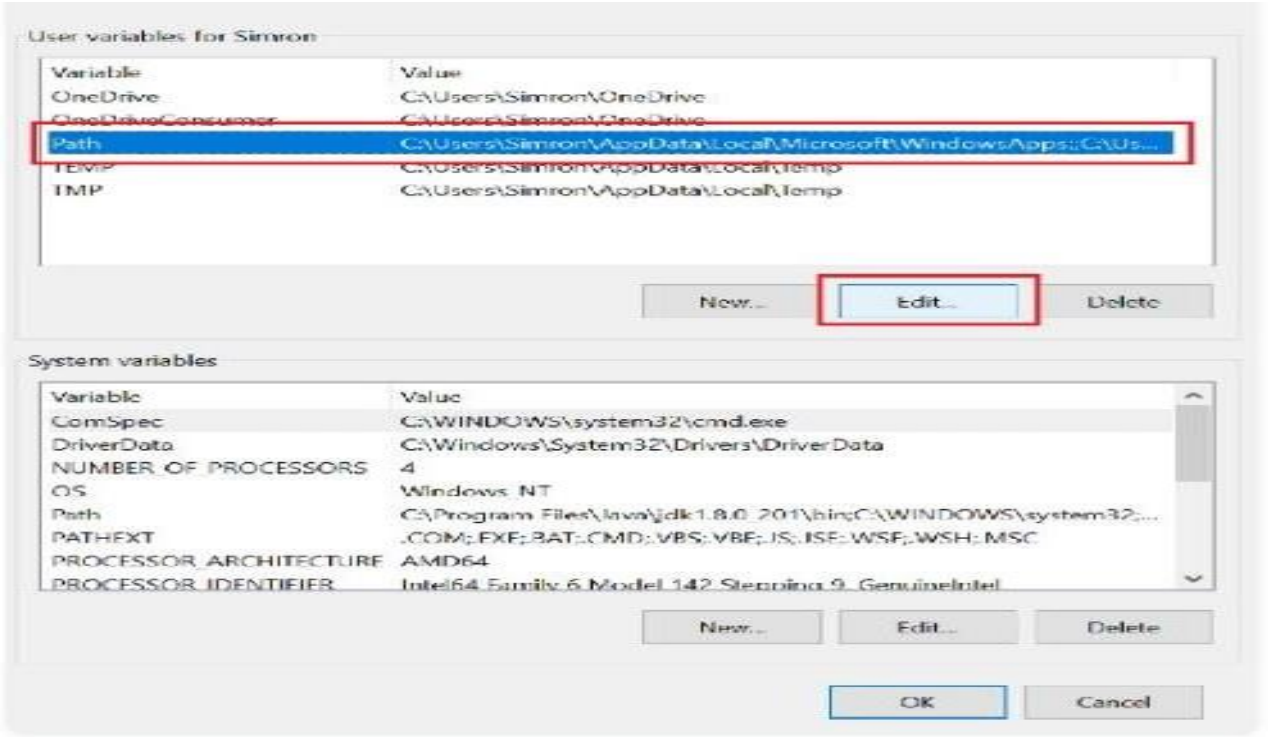
- Go to C:>MinGW>bin.
- Now, inside the bin folder, click on the address bar and copy the address.
- We require this address to be set as the path in the environment variable.
- If your install location was somewhere else, you may go to that location where you have installed MinGW.

**Step 2: Open Edit System Variables.**

Navigate to the search bar and type Edit the system environment variables and click on open to continue to edit system environment variables.

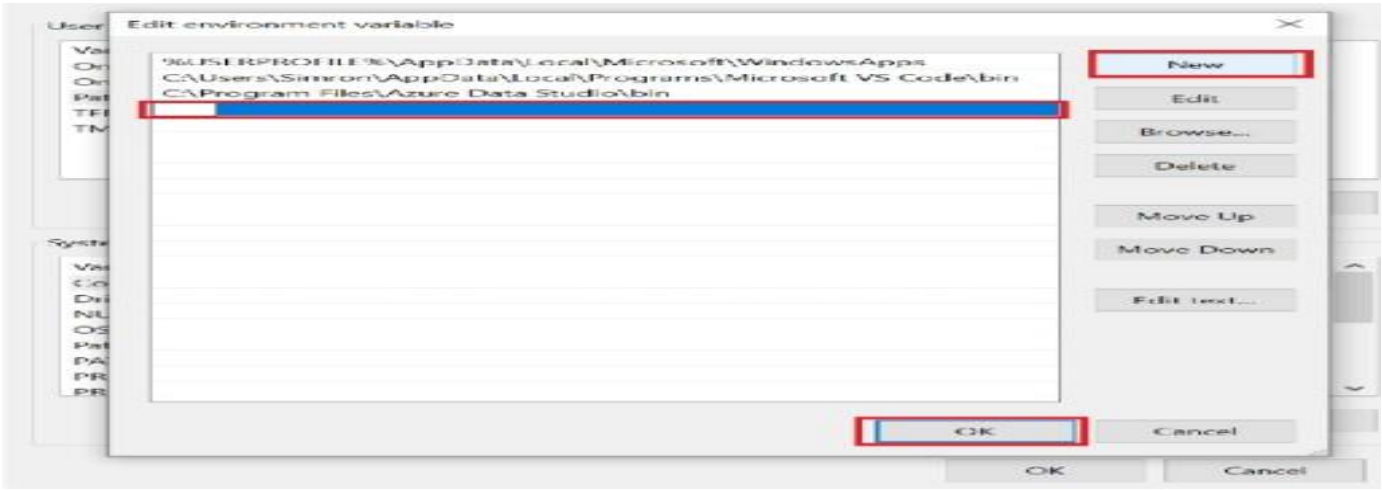


**Step 3: Edit the Path.** In the User variables for the *User* section, select the **path** and click on the Edit button.

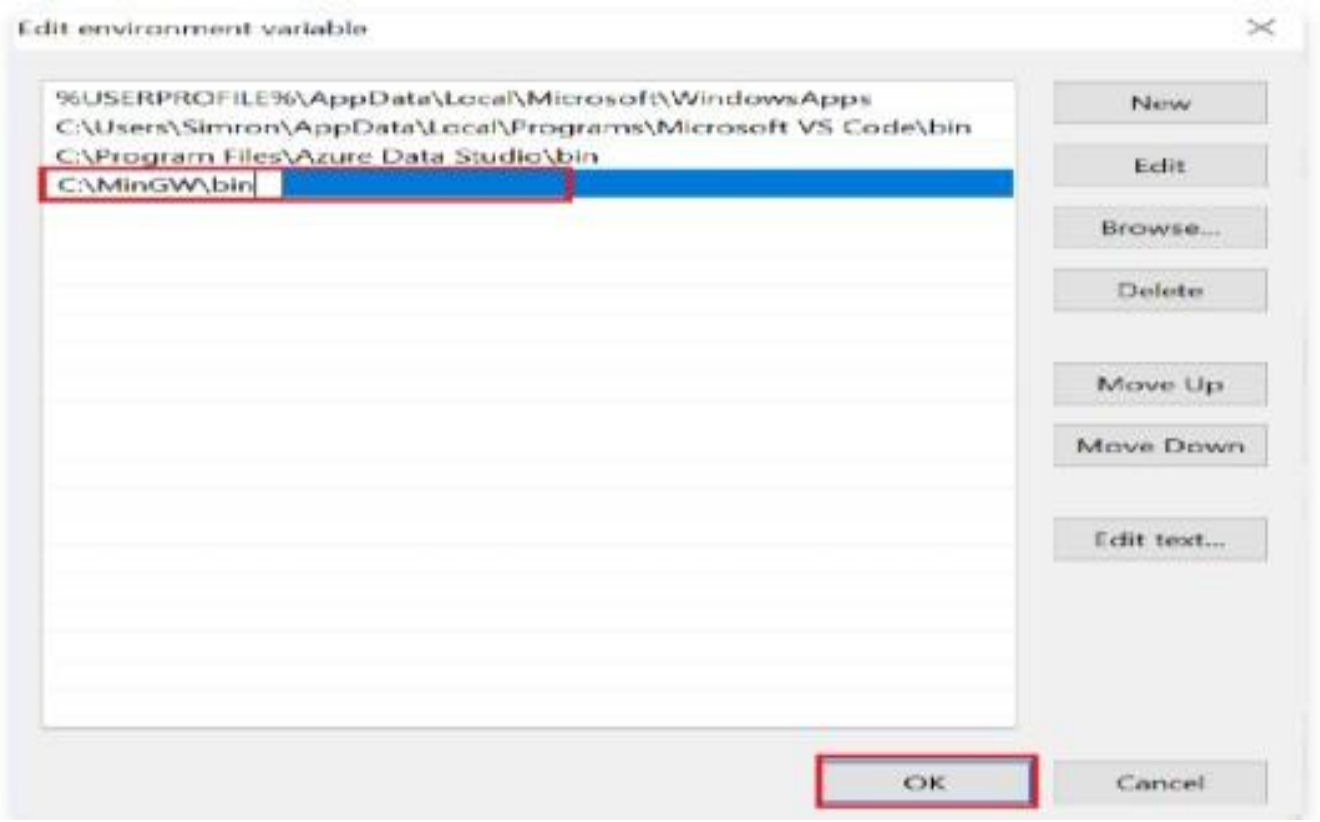



**Step 4: Setup a New Path.**

- After clicking on the Edit button, a new window, Edit environment variable will open. This window allows us to add the path as per our requirements.
- Since we want to add a new path, click on the New button. A new window, Edit environment variable, will open. This window allows us to add the path as per our requirements. Click on the ' New ' button since we want to add a new path.



**Step 5: Paste the Path.** Paste the path of the MinGW bin that was copied earlier and click on Ok.




## CREATING AND RUNNING C FILES

### A Simple Example

Start off with the following three files, hellomake.c, hellofunc.c, and hellomake.h, which would represent a typical main program, some functional code in a separate file, and an include file, respectively.

**hellomake.c**

```
#include <hellomake.h>

int main() {
  // call a function in another file
  myPrintHelloMake();

  return(0);
}
```

**hellofunc.c**

```
#include <stdio.h>
#include <hellomake.h>

void myPrintHelloMake(void) {

  printf("Hello makefiles!\n");

  return;
}
```

**hellomake.h**

```


/*
example include file
*/

void myPrintHelloMake(void);
```

Normally, you would compile this collection of code by executing the following command: gcc -o hellomake hellomake.c hellofunc.c -I.

This compiles the two .c files and names the executable hellomake. The -I. is included so that gcc will look in the current directory (.) for the include file hellomake.h. Without a makefile, the typical approach to the test/modify/debug cycle is to use the up arrow in a terminal to go back to

your last compile command so you don't have to type it each time, especially once you've added a few more .c files to the mix

The simplest makefile you could create would look something like:

Makefile 1
```
hellomake: hellomake.c hellofunc.c
gcc -o hellomake hellomake.c hellofunc.c -I.
```

If you put this rule into a file called Makefile or makefile and then type make on the command line it will execute the compile command as you have written it in the makefile. Note that make with no arguments executes the first rule in the file. Furthermore, by putting the list of files on which the command depends on the first line after the :, make knows that the rule hellomake needs to be executed if any of those files change. Immediately, you have solved One very important thing to note is that there is a tab before the gcc command in the makefile. There must be a tab at the beginning of any command, and make will not be happy if it's not there.

In order to be a bit more efficient, let's try the following:
Makefile 2 CC=gcc CFLAGS=-I.

```
hellomake: hellomake.o hellofunc.o
$(CC) -o hellomake hellomake.o hellofunc.o
```

So now we've defined some constants CC and CFLAGS. It turns out these are special constants that communicate to make how we want to compile the files hellomake.c and hellofunc.c. In particular, the macro CC is the C compiler to use, and CFLAGS is the list of flags to pass to the compilation command. By putting the object files--
hellomake.o and hellofunc.o--in the dependency list and in the rule, make knows it must first compile the .c versions individually, and then build the executable hellomake.

Using this form of makefile is sufficient for most small scale projects. However, there is one thing missing: dependency on the include files. If you were to make a change to hellomake.h, for example, make would not recompile the .c files, even though they needed to be. In order to fix this, we need to tell make that all .c files depend on certain .h files. We can do this by writing a simple rule and adding it to the makefile.

Makefile 3 CC=gcc CFLAGS=-I.
```
DEPS = hellomake.h

%.o: %.c $(DEPS)
$(CC) -c -o $@ $< $(CFLAGS)

hellomake: hellomake.o hellofunc.o
$(CC) -o hellomake hellomake.o hellofunc.o
```

This addition first creates the macro DEPS, which is the set of .h files on which the .c files depend. Then we define a rule that applies to all files ending in the .o suffix. The rule says that the .o file depends upon the .c version of the file and the .h files included in the DEPS macro. The rule then says that to generate the .o file, make needs to compile the .c file using the compiler defined in the CC macro. The -c flag says to generate the object file, the -o $@ says to put the output of the compilation in the file named on the left side of the :, the $< is the first item in the dependencies list, and the CFLAGS macro is defined as above.

As a final simplification, let's use the special macros $@ and $^, which are the left and right sides of the :, respectively, to make the overall compilation rule more general. In the example below, all of the include files should be listed as part of the macro DEPS, and all of the object files should be listed as part of the macro OBJ.

Makefile 4 CC=gcc CFLAGS=-I.
DEPS = hellomake.h
OBJ = hellomake.o hellofunc.o
%.o: %.c $(DEPS)
$(CC) -c -o $@ $< $(CFLAGS)
hellomake: $(OBJ)
$(CC) -o $@ $^ $(CFLAGS)
Makefile 5 IDIR =../include CC=gcc
CFLAGS=-I$(IDIR)
ODIR=obj LDIR =../lib
LIBS=-lm
_DEPS = hellomake.h
DEPS = $(patsubst %,$(IDIR)/%,$(_DEPS))
_OBJ = hellomake.o hellofunc.o
OBJ = $(patsubst %,$(ODIR)/%,$(_OBJ))
$(ODIR)/%.o: %.c $(DEPS)
$(CC) -c -o $@ $< $(CFLAGS)
hellomake: $(OBJ)
$(CC) -o $@ $^ $(CFLAGS) $(LIBS)
.PHONY: clean
clean:
rm -f $(ODIR)/*.o *~ core $(INCDIR)/*~
So now you have a perfectly good makefile that you can modify to manage small and medium-sized software projects. You can add multiple rules to a makefile; you can even create rules that call other rules. For more information on makefiles and the make function, check out the GNU Make Manual, which will tell you more than you ever wanted to know (really)

**RESULT:**

Thus, The installation of C/GCC compiler in Windows has been done and verified successfully.

**Ex.No.10** **VERSION CONTROL SYSTEMS COMMANDS**
**Date:**

**AIM:**

To use version control systems command to clone, commit, push, fetch,pull, checkout, reset, and delete repositories.

**PROCEDURE:**

**Clone Command**

The **git clone** is a command-line utility which is used to make a local copy of a remote repository. It accesses the repository through a remote URL.

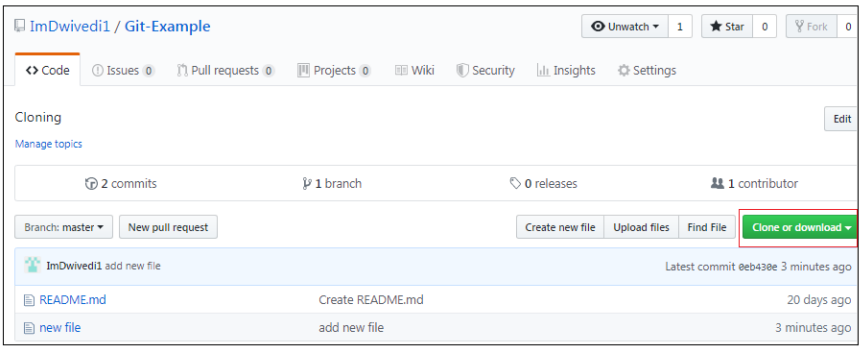
Usually, the original repository is located on a remote server, often from a Git service like GitHub, Bitbucket, or GitLab. The remote repository URL is referred to the **origin**.

**Syntax:** $ git clone **<repository** URL**>**

Suppose, you want to clone a repository from GitHub, or have an existing repository owned by any other user you would like to contribute. Steps to clone a repository are as follows:

**Step 1:**

Open GitHub and navigate to the main page of the repository.




**Step 2:**
Under the repository name, click on **Clone or download**.

**Step 3**:
Select the **Clone with HTTPs section** and **copy the clone URL** for the repository. For the empty repository, you can copy the repository page URL from your browser and skip to next step.

**Step 4:**
Open Git Bash and change the current working directory to your desired location where you want to create the local copy of the repository.

**Step 5:**
Use the git clone command with repository URL to make a copy of the remote repository. See the below command:
$ git clone https://github.com/ImDwivedi1/Git-Example.git

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop (master)
$ cd "new folder"

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/new folder (master)
$ git clone https://github.com/ImDwivedi1/Git-Example.git
Cloning into 'Git-Example'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.

HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/new folder (master)
$
```

## Commit

The above command will prompt a default editor and ask for a commit message. We have made a change to **newfile1.txt** and want it to commit it. It can be done as follows:

Consider the below output:

```
HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/NewDirectory (master)
$ git commit
[master e3107d8] Update Newfile1
 2 files changed, 1 insertion(+)
 delete mode 100644 index.jsp
```

As we run the command, it will prompt a default text editor and ask for a commit message. The text editor will look like as follows:

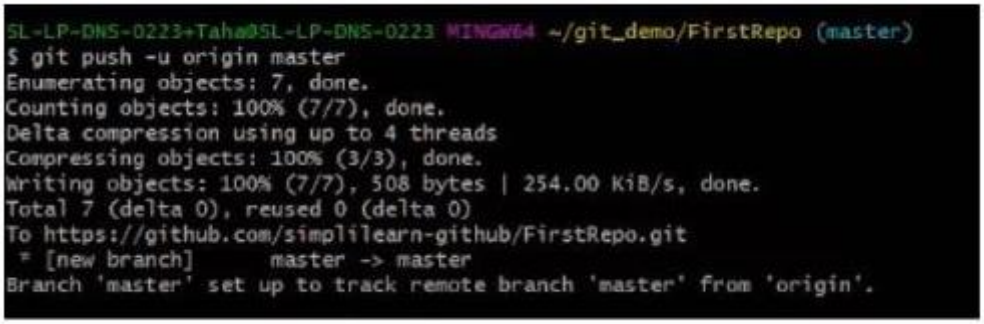
```
Update Newfile1
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#       deleted:    index.jsp
#       modified:   newfile1.txt
#
~
~
~
~
~
~
~
~
~
~
~
~
~
<U/Desktop/NewDirectory/.git/COMMIT_EDITMSG[+] [unix] (17:59 26/11/2019)1,15 All
:wq
```

Press the **Esc** key and after that '**I**' for insert mode. Type a commit message whatever you want. Press **Esc** after that '**:wq**' to save and exit from the editor. Hence, we have successfully made a commit.

**git push**

The command git push is used to transfer the commits or pushing the content from the local repository to the remote repository.

The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

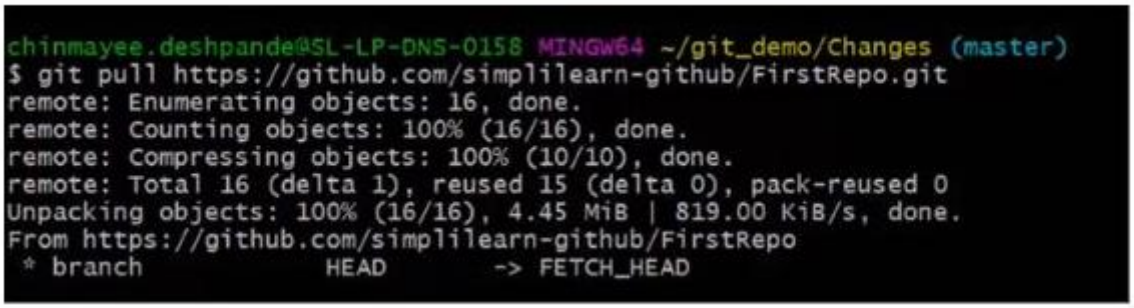



**Pull**

The git pull command is used to fetch and merge changes from the remote repository to the local repository.

The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

git pull <branch_name> <remote URL>




**Checkout**

The git checkout command is used to switch branches, whenever the work is to be started on a different branch.

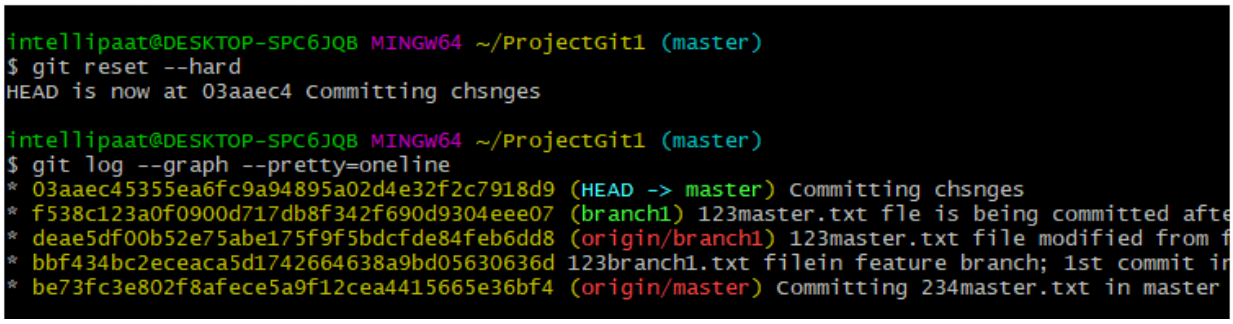
The command works on three separate entities: files, commits, and branches.

# Checkout an existing branch

# Checkout and create a new branch with that name
git checkout -b <new_branch>

**Reset –hard**

We use this command to **return** the *entire* working tree to the last committed state.




**Delete command**

git branch -d  local_branch_name

git branch is the command to delete a branch locally.

-d is a flag, an option to the command, and it's an alias for --delete. It denotes that you want to delete something, as the name suggests. - local_branch_name is the name of the branch you want to delete.

**RESULT:**

Thus, The version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete  repositories have been executed and verified successfully.