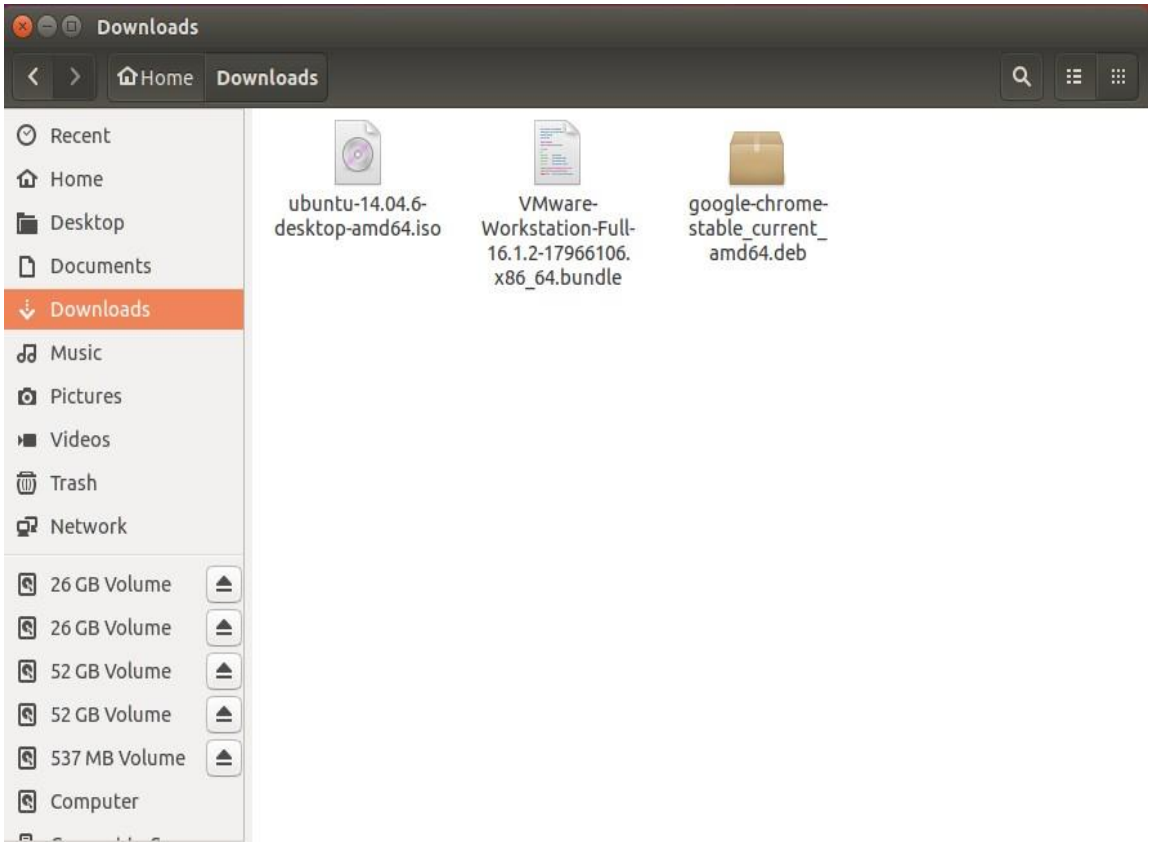
EX.NO.1 INSTALL VIRTUALBOX WITH LINUX OS ON TOP OF WINDOWS
DATE:

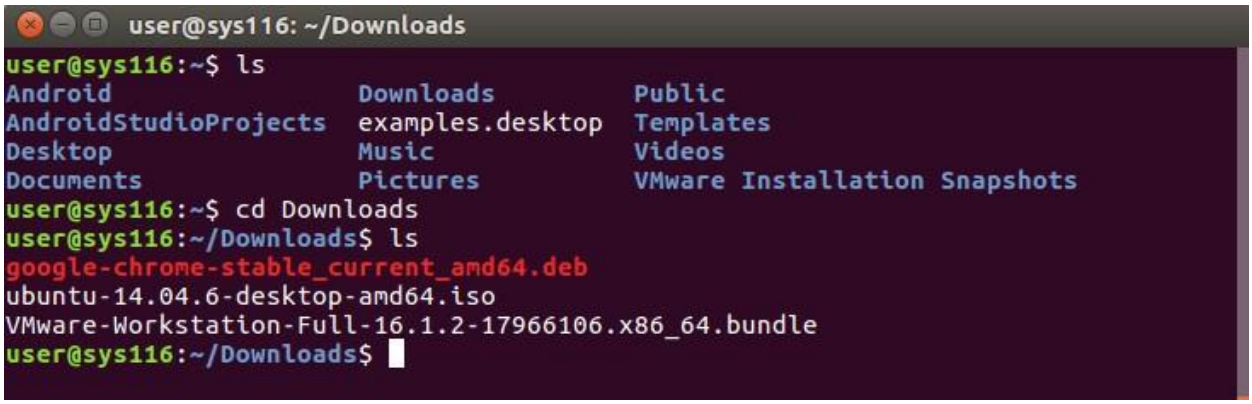
AIM:

To install Virtualbox with different flavours of linux or windows OS on top of windows host operating system.

Step 1:
Download the following
VMware package
Guest OS (Ubuntu) ISO file



Step 2: Navigate to the directory where the above files are downloaded.



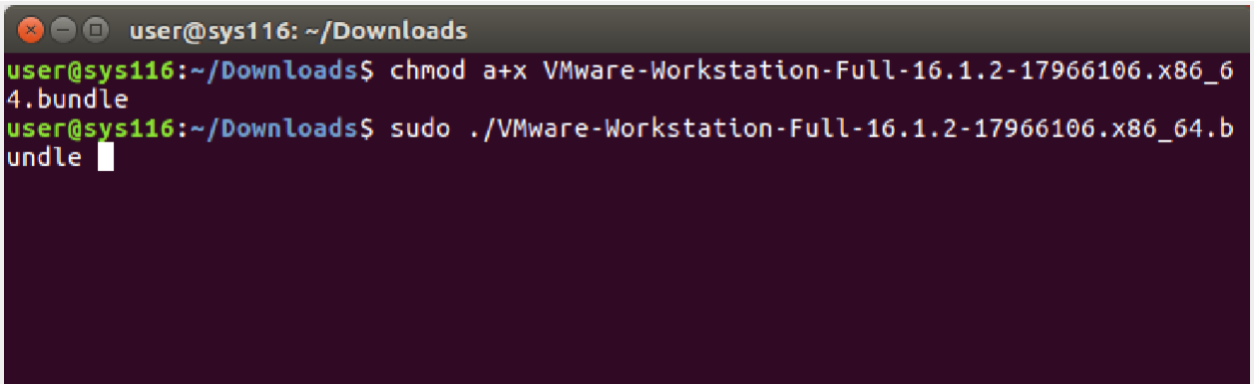
Step 3:

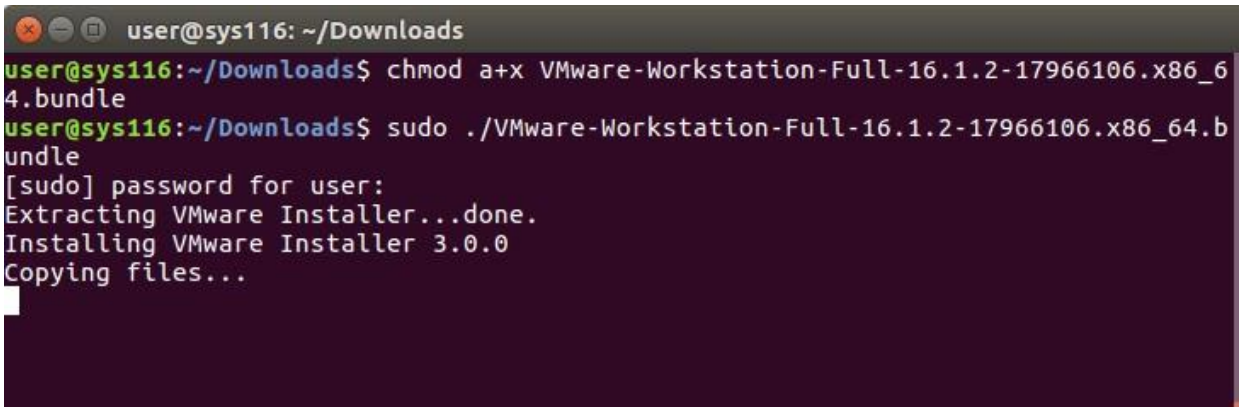
Change the permissions of the VMware package. Use the following command, chmod a+x <VMWARE_PACKAGE_NAME> That is,

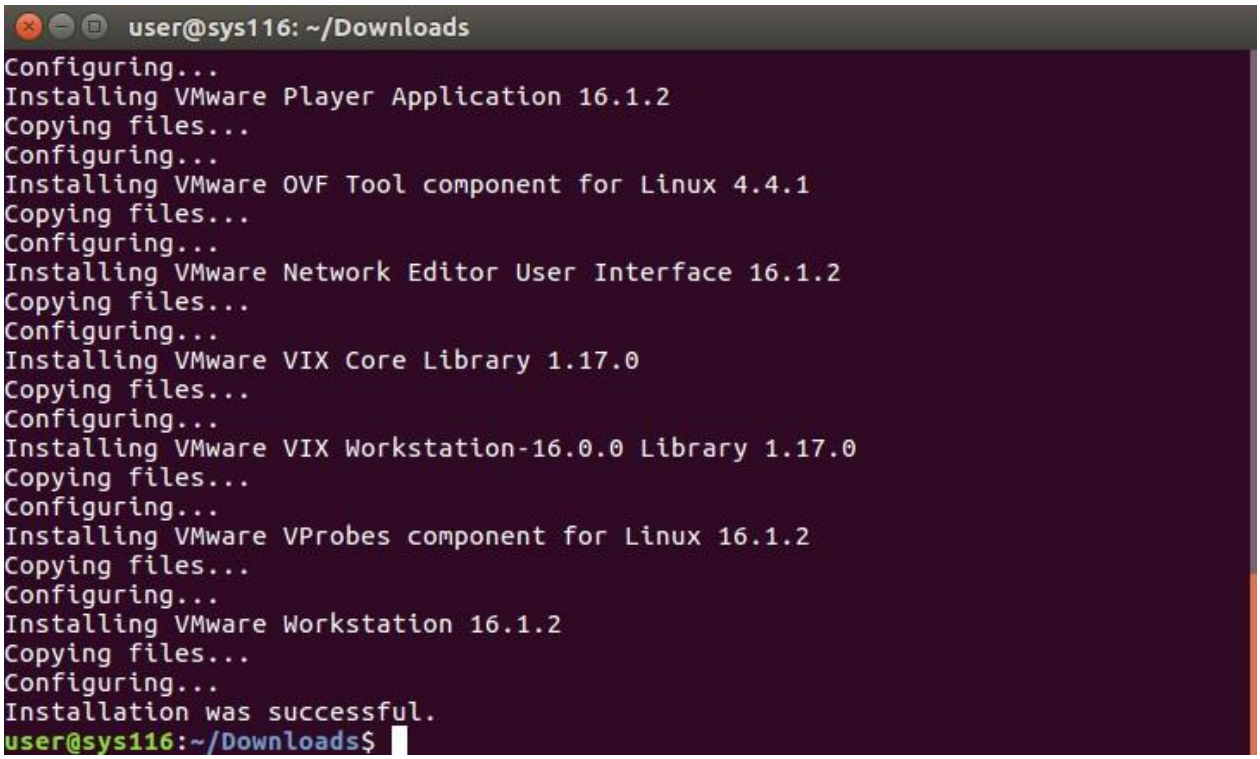
chmod a+x VMware-Workstation-Full-16.1.2-17966106.x86_64.bundle

Run the next command to install the VMware.

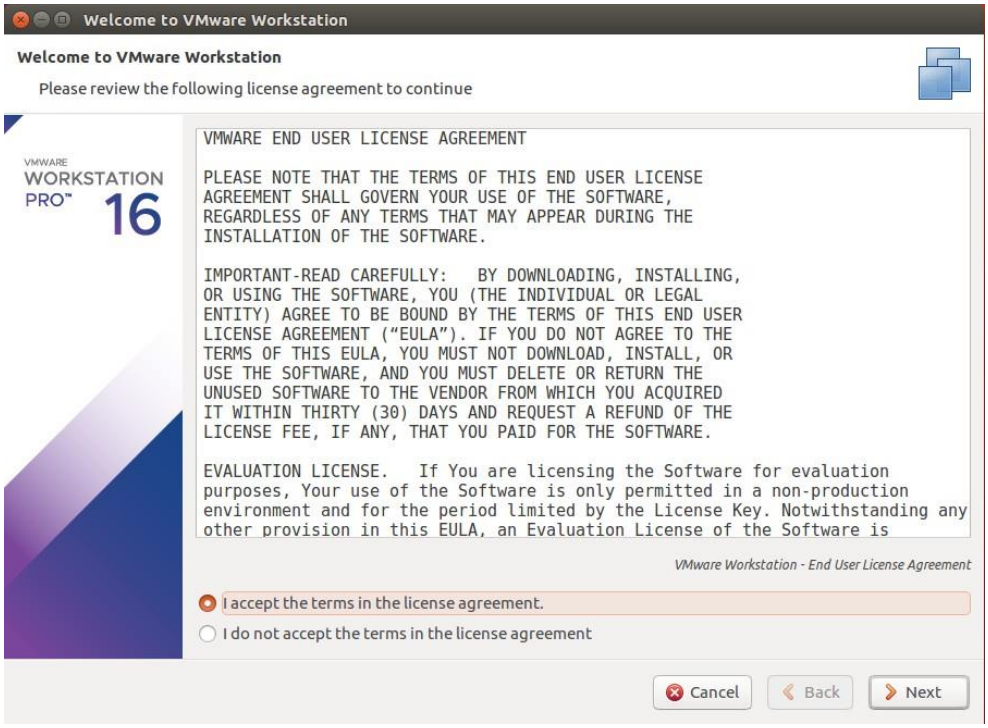
sudo ./VMware-Workstation-Full-16.1.2-17966106.x86_ 64.bundle

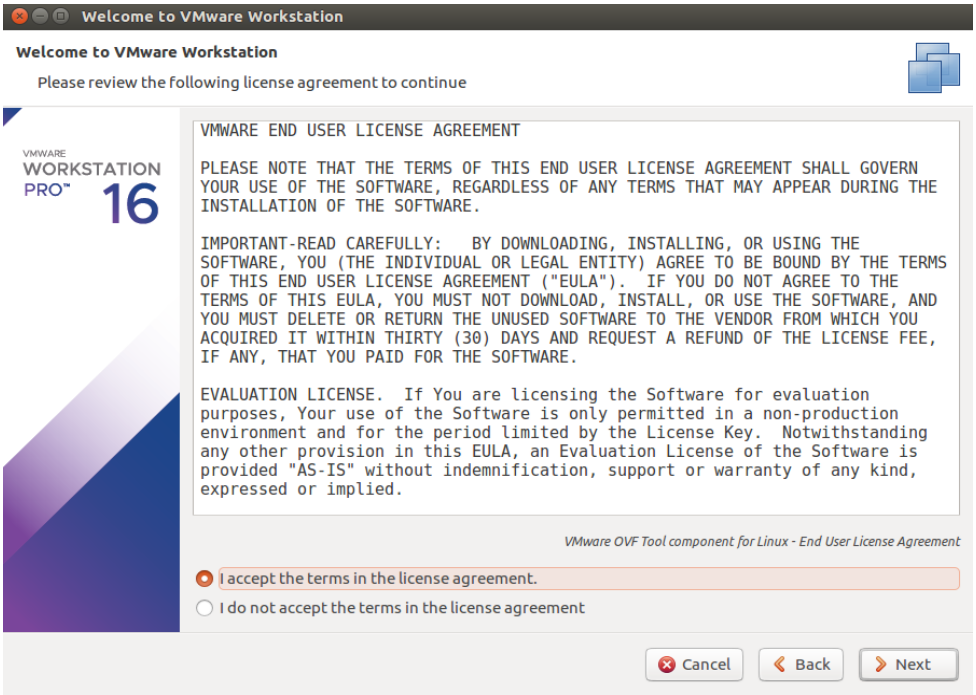


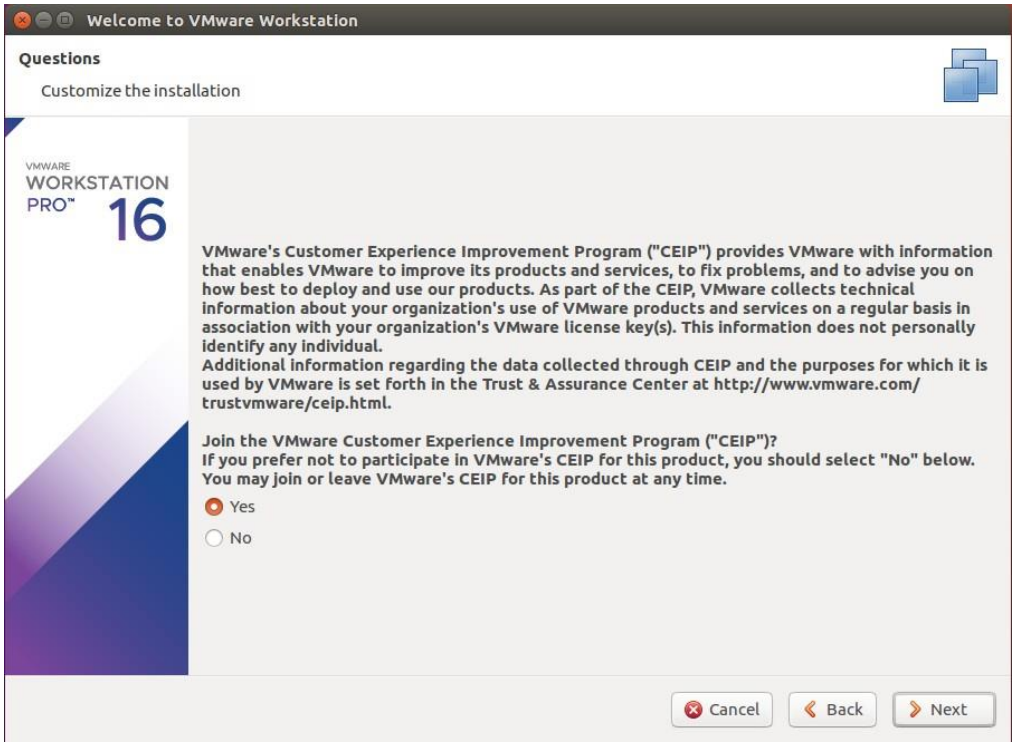


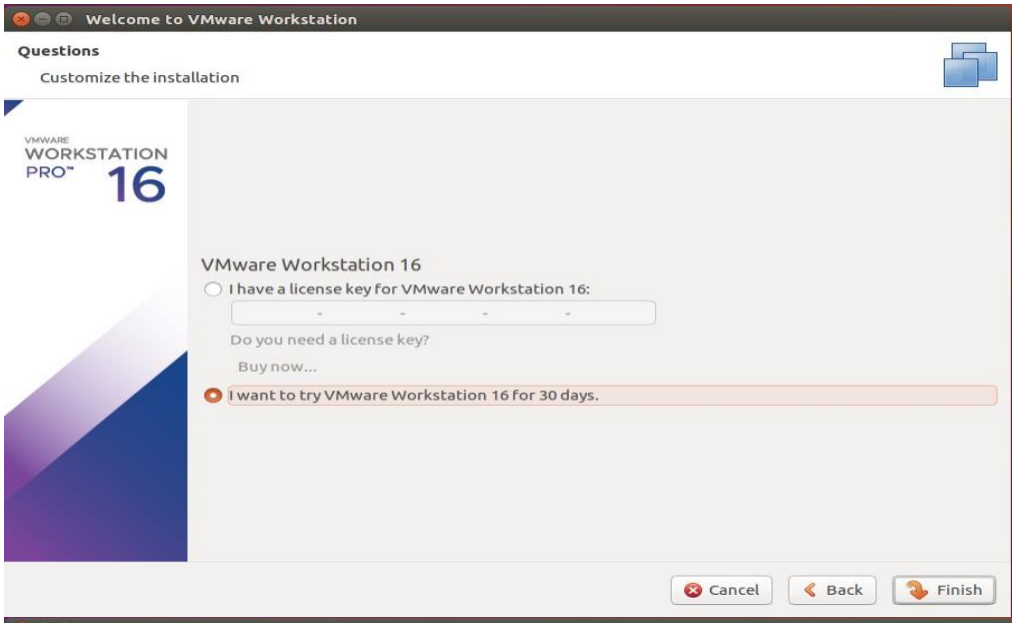


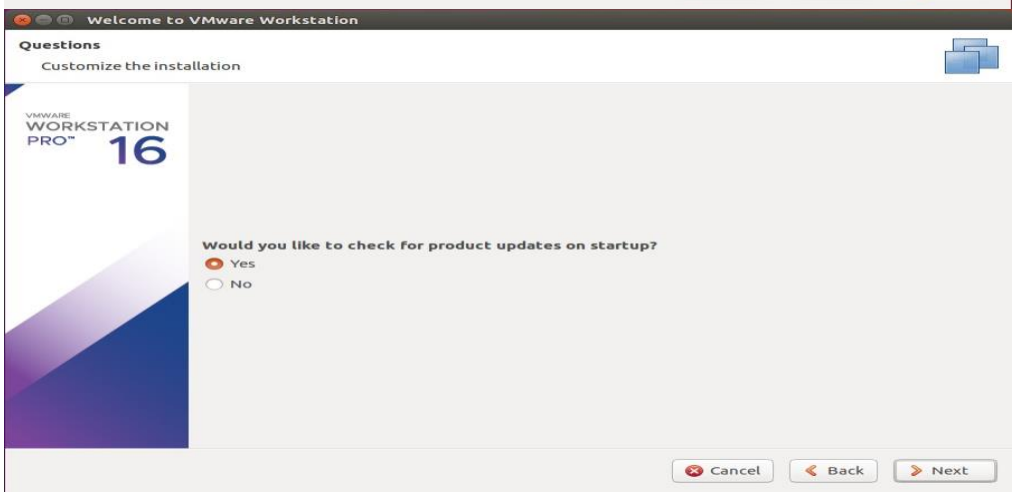
Step 4:
Open VMware Player
Accept all the agreements, terms and conditions.
Select all the default settings and continue.
VMware is successfully installed on your Host Machine.

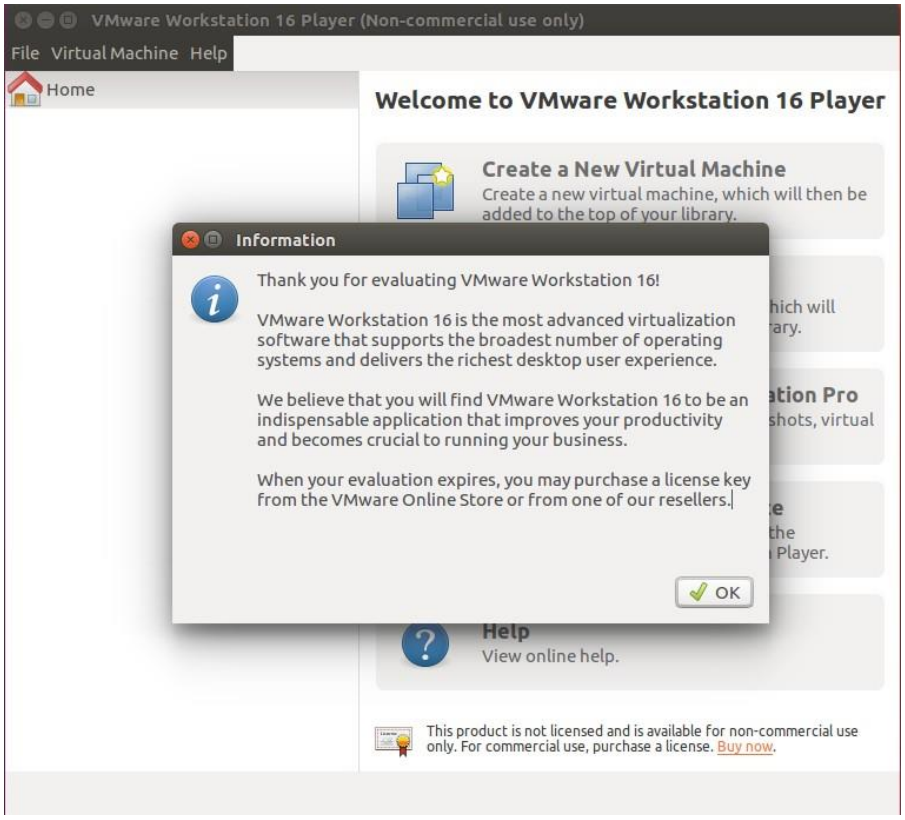


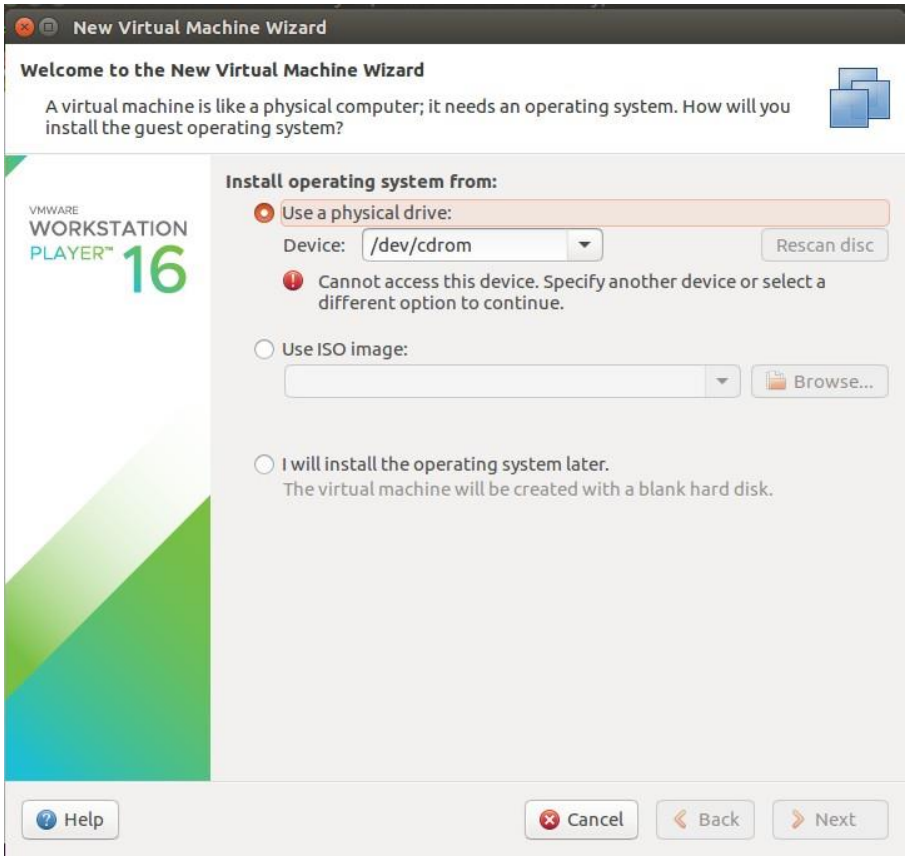



Welcome to VMware Workstation
Questions
Customize the installation
VMWARE
WORKSTATION
PRO™
16
VMware's Customer Experience Improvement Program ("CEIP") provides VMware with information that enables VMware to improve its products and services, to fix problems, and to advise you on how best to deploy and use our products. As part of the CEIP, VMware collects technical information about your organization's use of VMware products and services on a regular basis in association with your organization's VMware license key(s). This information does not personally identify any individual.
Additional information regarding the data collected through CEIP and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center at http://www.vmware.com/trustvmware/ceip.html.
Join the VMware Customer Experience Improvement Program ("CEIP")?
If you prefer not to participate in VMware's CEIP for this product, you should select "No" below. You may join or leave VMware's CEIP for this product at any time.
Yes
No
Cancel
Back
Next

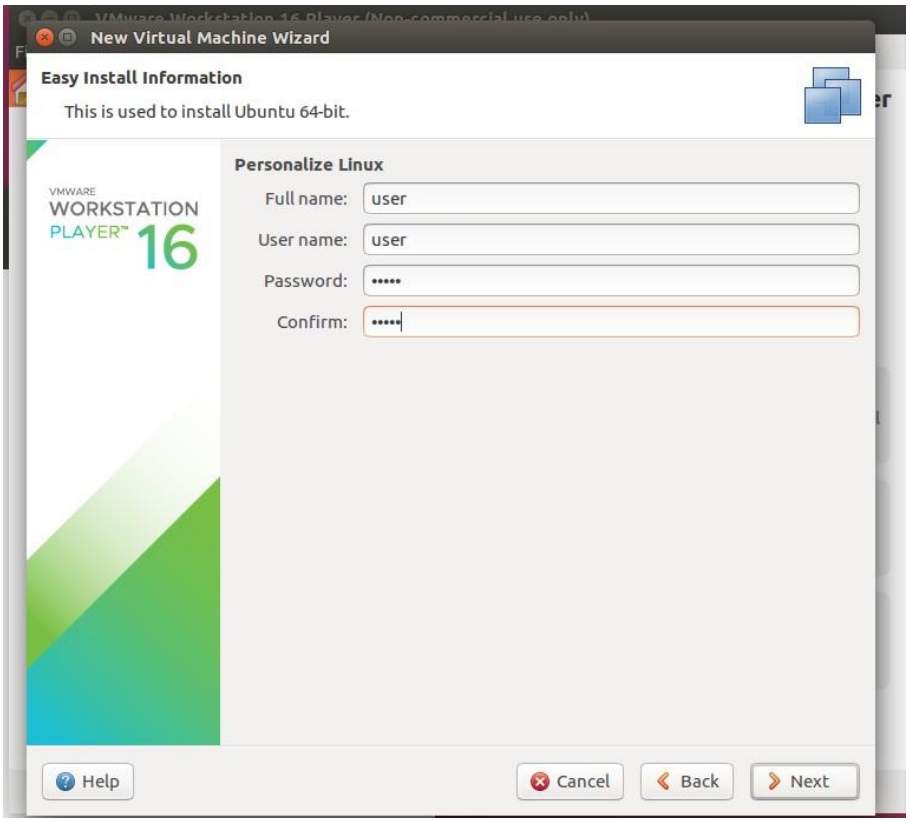

Welcome to VMware Workstation
Questions
Customize the installation
VMWARE
WORKSTATION
PRO™
16
VMware Workstation 16
I have a license key for VMware Workstation 16:
Do you need a license key?
Buy now...
I want to try VMware Workstation 16 for 30 days.
Cancel
Back
Finish

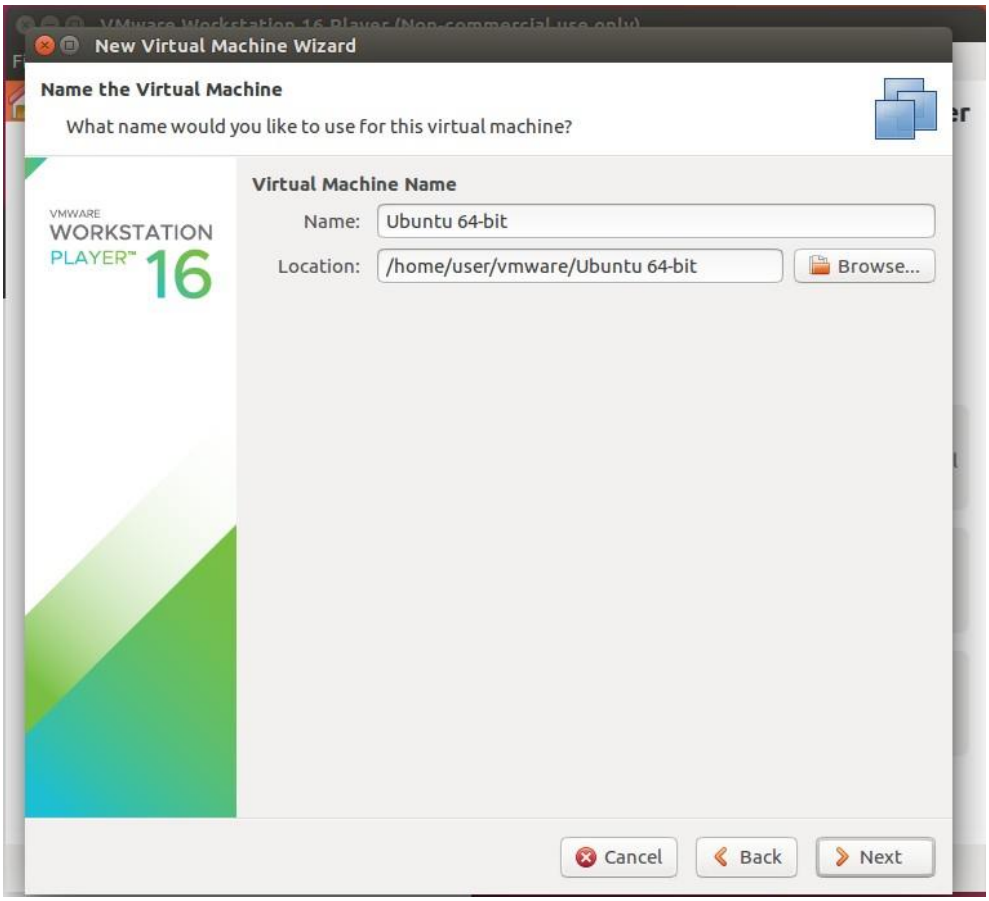

Welcome to VMware Workstation
Questions
Customize the installation
VMWARE
WORKSTATION
PRO™
16
Would you like to check for product updates on startup?
Yes
No
Cancel
Back
Next

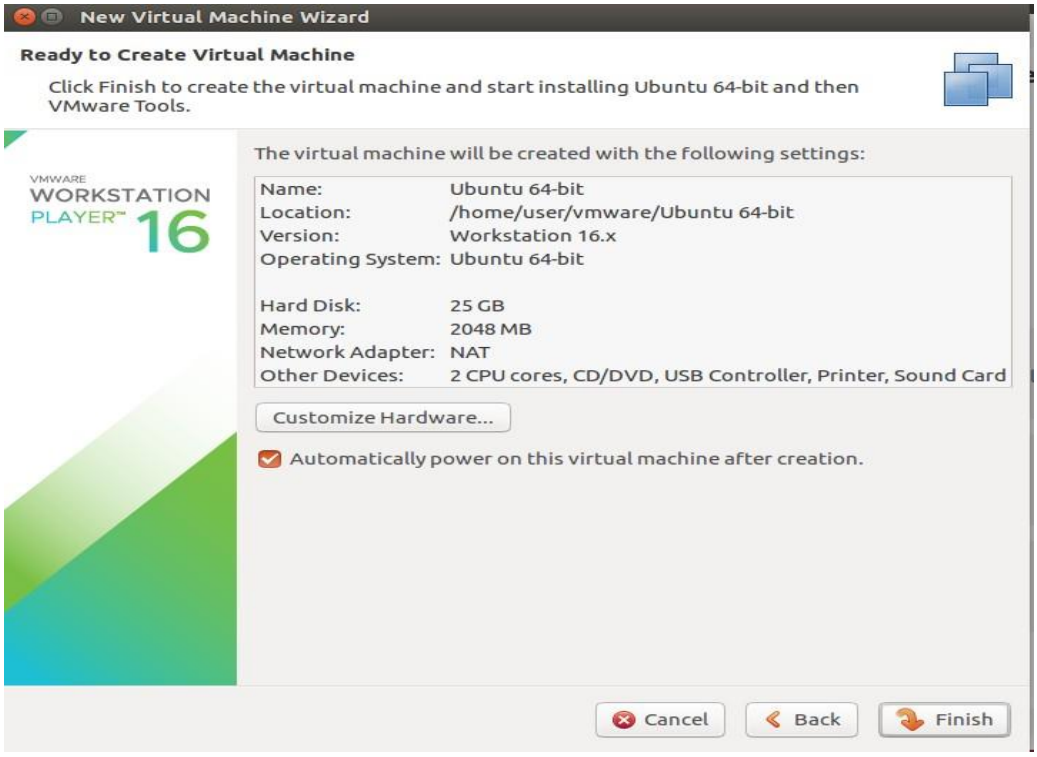

VMware Workstation 16 Player (Non-commercial use only)
File Virtual Machine Help
Home
Welcome to VMware Workstation 16 Player
Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.
Information
Thank you for evaluating VMware Workstation 16!
VMware Workstation 16 is the most advanced virtualization software that supports the broadest number of operating systems and delivers the richest desktop user experience.
We believe that you will find VMware Workstation 16 to be an indispensable application that improves your productivity and becomes crucial to running your business.
When your evaluation expires, you may purchase a license key from the VMware Online Store or from one of our resellers.
OK
Help
View online help.
This product is not licensed and is available for non-commercial use only. For commercial use, purchase a license. Buy now.


New Virtual Machine Wizard
Welcome to the New Virtual Machine Wizard
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?
VMWARE WORKSTATION PLAYER 16
Install operating system from:
Use a physical drive:
Device: /dev/cdrom
Rescan disc
Cannot access this device. Specify another device or select a different option to continue.
Use ISO image:
Browse...
I will install the operating system later.
The virtual machine will be created with a blank hard disk.
Help
Cancel
Back
Next

Step 5:
Select “Create a new Virtual Machine”
Select “Use ISO image” and Browse your Guest OS ISO image path and click “Next”.
Use your password as “admin” and click “Next”.
Click “Next”
Select your disk size upto a maximum of 25GB and Click “Next”.

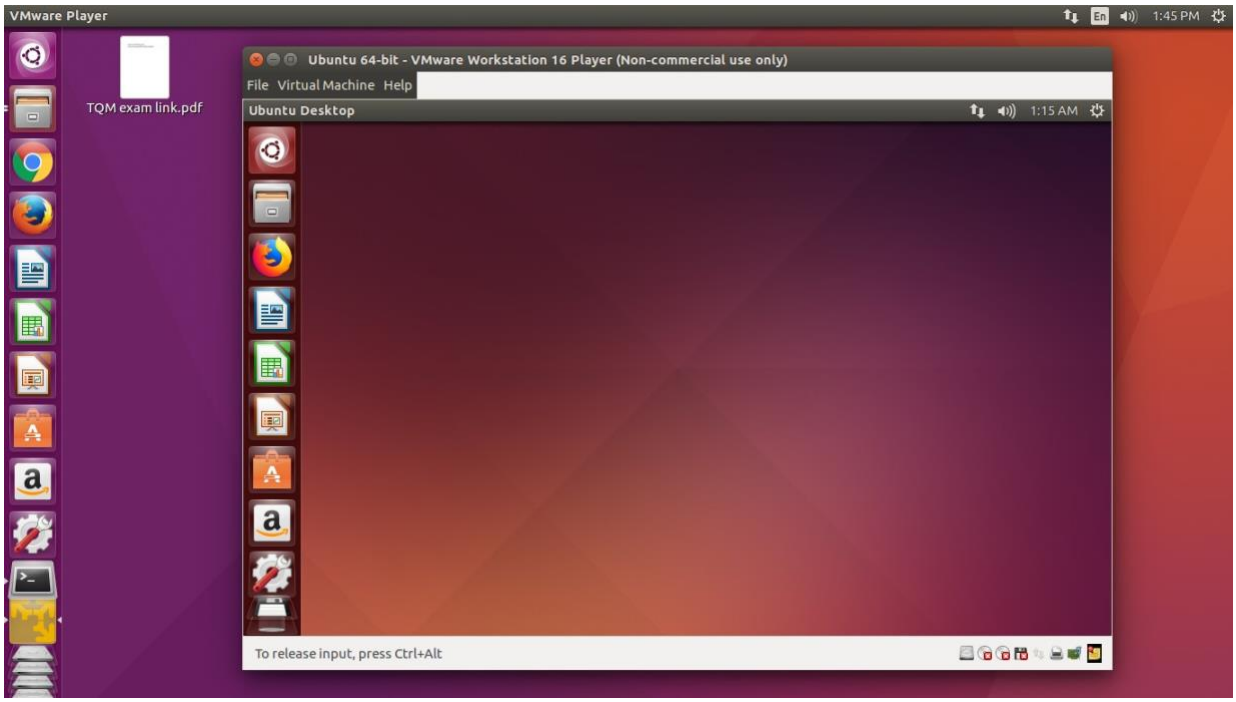



New Virtual Machine Wizard
Name the Virtual Machine
What name would you like to use for this virtual machine?
VMWARE
WORKSTATION
PLAYER™ 16
Virtual Machine Name
Name: Ubuntu 64-bit
Location: /home/user/vmware/Ubuntu 64-bit
Browse...
Cancel
Back
Next


New Virtual Machine Wizard
Ready to Create Virtual Machine
Click Finish to create the virtual machine and start installing Ubuntu 64-bit and then VMware Tools.
VMWARE
WORKSTATION
PLAYER™ 16
The virtual machine will be created with the following settings:
Name: Ubuntu 64-bit
Location: /home/user/vmware/Ubuntu 64-bit
Version: Workstation 16.x
Operating System: Ubuntu 64-bit
Hard Disk: 25 GB
Memory: 2048 MB
Network Adapter: NAT
Other Devices: 2 CPU cores, CD/DVD, USB Controller, Printer, Sound Card
Customize Hardware...
Automatically power on this virtual machine after creation.
Cancel
Back
Finish

Select Finish

Guest OS is installed Successfully.



RESULT:

Thus installation of Virtualbox with different flavours of linux or windows OS on top of windows host operating system is completed successfully.

Ex.No: 2 Install Gcc Compilers in Virtual Machine
Date :

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

Algorithm:

Step 1: To check Gcc compiler is installed or not.
$ dpkg - l | grep gcc
Step 2: If GCC compiler is not installed in VM, to execute the following command
$sudo apt-get install gcc (or) $sudo apt-get install build-essential
Step 3: Open a Vi Editor
Step 4: Get the no. of rows and columns for first and second
matrix. Step 5: Get the values of x and y matrix using for loop.
Step 6: Find the product of first and second and store the result in multiply
matrix. multiply[i][j]=multiply[i][j]+(first[i][k]*second[k][j]);
Step 7: Display the resultant
matrix. Step 8: Stop the program.

Program:

```
#include <stdio.h>
void main()
{
 int m, n, p, q, c, d, k, sum = 0;
 int first[10][10], second[10][10], multiply[10][10];
 printf("Enter the number of rows and columns of first matrix\n");
 scanf("%d%d", &m, &n);
 printf("Enter the elements of first matrix\n");
  for (  c = 0 ; c < m ; c++ )
   for ( d = 0 ; d < n ; d++ )
   scanf("%d", &first[c][d]);
  printf("Enter the number of rows and columns of second matrix\n");
 scanf("%d%d", &p, &q);
        if ( n != p )
  printf("Matrices with entered orders can't be multiplied with each other.\n");
 else
 {
  printf("Enter the elements of second matrix\n");

  for ( c = 0 ; c < p ; c++ ) for ( d
   = 0 ; d < q ; d++ ) scanf("%d",
   &second[c][d]); for ( c = 0 ; c
   < m ; c++ )
  {
    for ( d = 0 ; d < q ; d++ )
    {
      for ( k = 0 ; k < p ; k++ )
      {
        sum = sum + first[c][k]*second[k][d];
```

```
      }


       multiply[c][d] = sum;
      sum = 0;
     }
    }
     printf("Product of entered matrices:-\n");
     for ( c = 0 ; c < m ; c++ )
    {



}
}
}
```

Output:

Compile: cse105@it105-HP-ProDesk-400-G1-SFF:~$ gcc matrix.c
Run: cse105@cse105-HP-ProDesk-400-G1-SFF:~$ ./a.out

```
Enter the number of rows and columns of first matrix
2 2
Enter the elements of first matrix
2 2 2 2
Enter the number of rows and columns of second matrix
2 2
Enter the elements of second matrix
2 2 2 2
Product of entered matrices:-
8   8
8   8
```

Result :

Thus a C compiler in the virtual machine to execute a sample program has been executed successfully.

EXNO: 3 INSTALLING GOOGLE APP ENGINE

AIM:

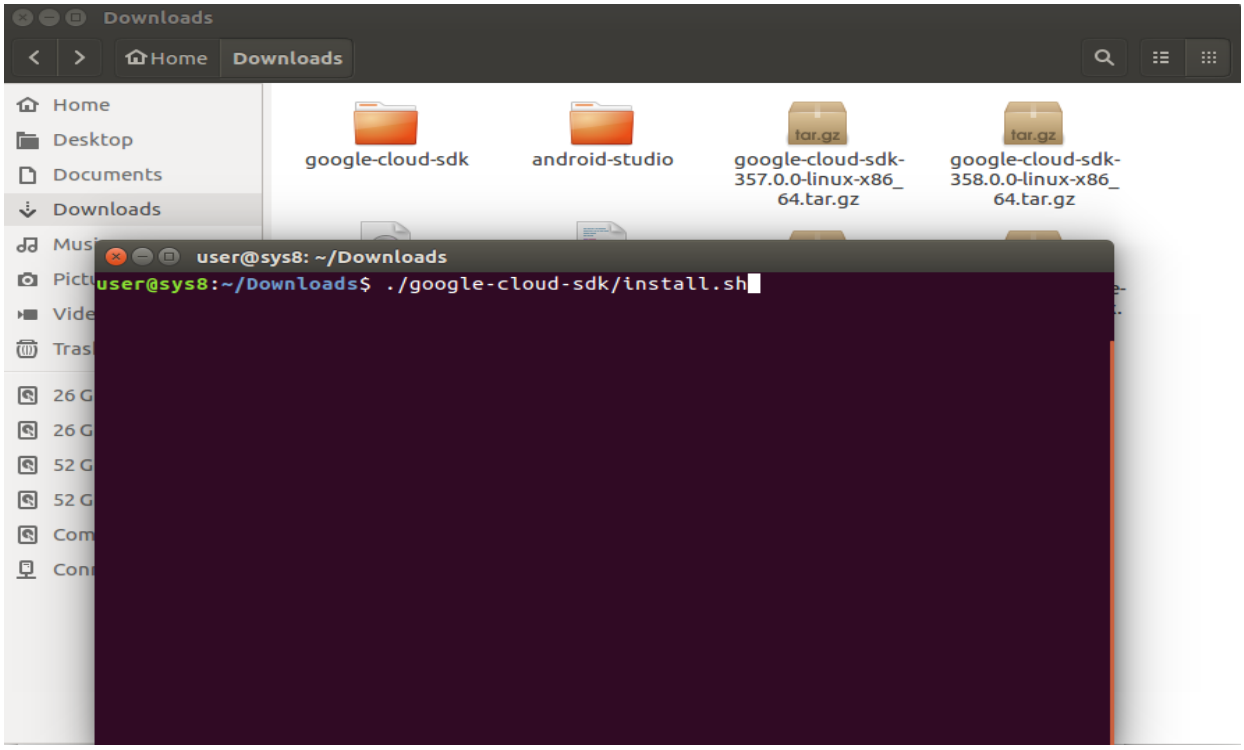
To Install Google App Engine. Create hello world app and other simple web applications using python/java.

STEP 1

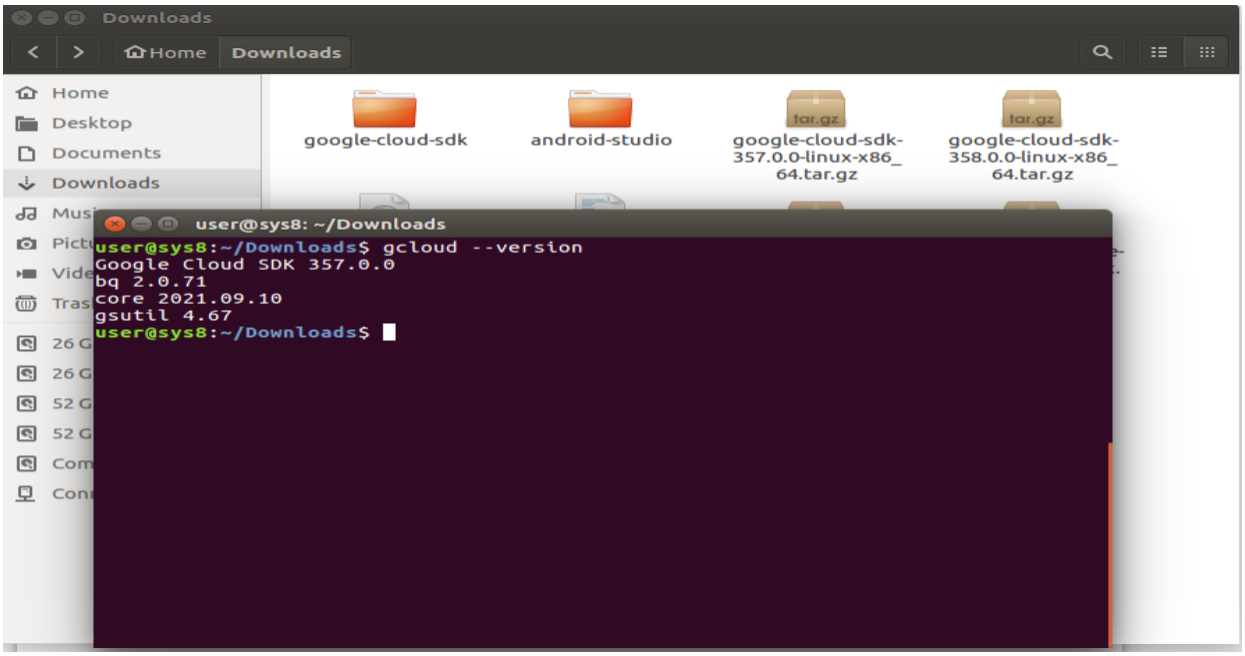
- Download google cloud SDK from the google cloud website
  [**https://cloud.google.com/sdk/docs/install**]
  (**OR**)
- Click the following link to download the required Google Cloud SDK
  [https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-357.0.0-linux-x86_64.tar.gz]
  Extract the zip folder in Downloads/ folder

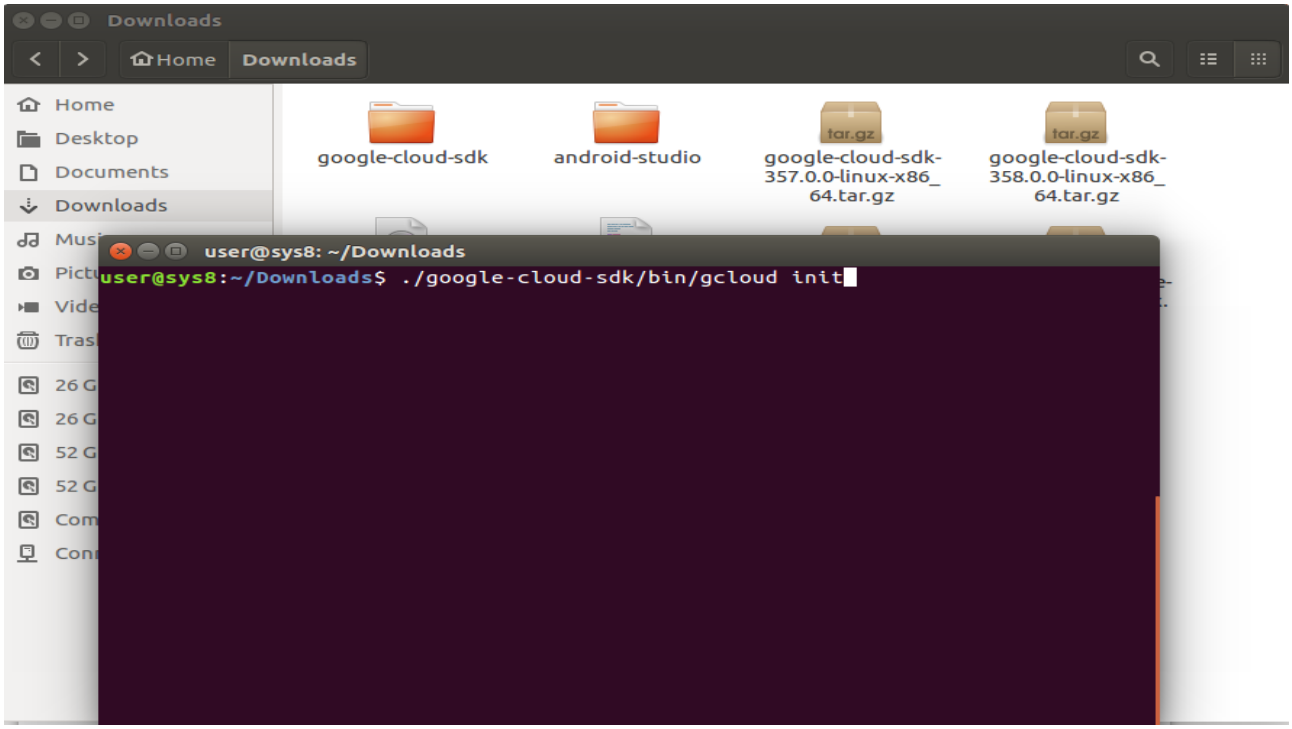
STEP 2

- Open new terminal in Downloads/ folder and install the google cloud
  ./google-cloud-sdk/install.sh

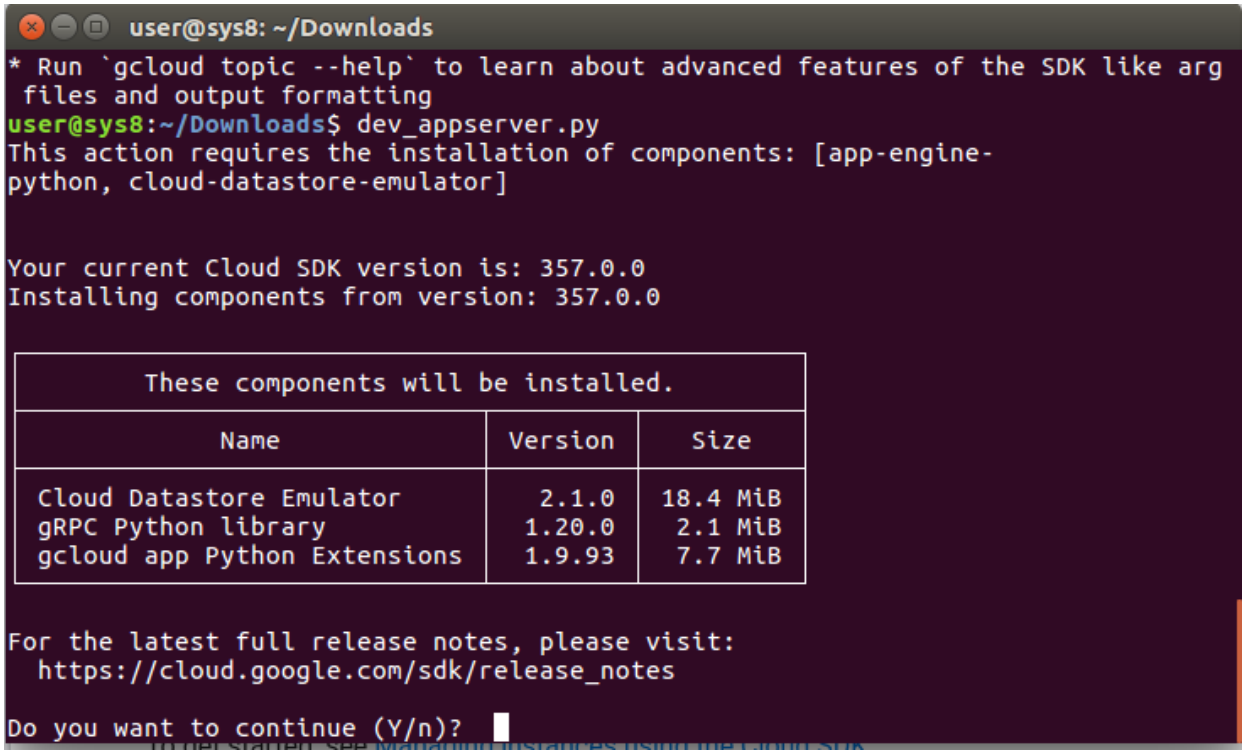


- Open a new terminal and initialized the gcloud
  ./google-cloud-sdk/bin/gcloud init

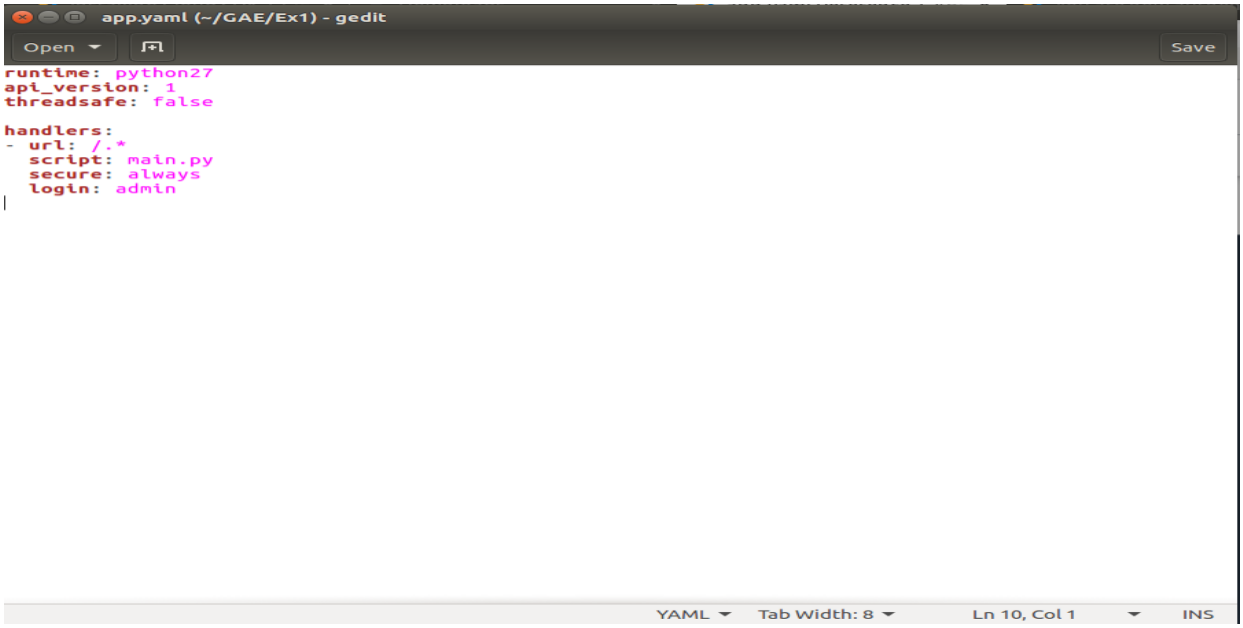

Downloads
Home
Downloads
Home
Desktop
Documents
Downloads
google-cloud-sdk
android-studio
google-cloud-sdk-357.0.0-linux-x86_64.tar.gz
google-cloud-sdk-358.0.0-linux-x86_64.tar.gz
user@sys8: ~/Downloads
user@sys8:~/Downloads$ gcloud --version
Google Cloud SDK 357.0.0
bq 2.0.71
core 2021.09.10
gsutil 4.67
user@sys8:~/Downloads$


Downloads
Home
Downloads
Home
Desktop
Documents
Downloads
google-cloud-sdk
android-studio
google-cloud-sdk-357.0.0-linux-x86_64.tar.gz
google-cloud-sdk-358.0.0-linux-x86_64.tar.gz
user@sys8: ~/Downloads
user@sys8:~/Downloads$ ./google-cloud-sdk/bin/gcloud init

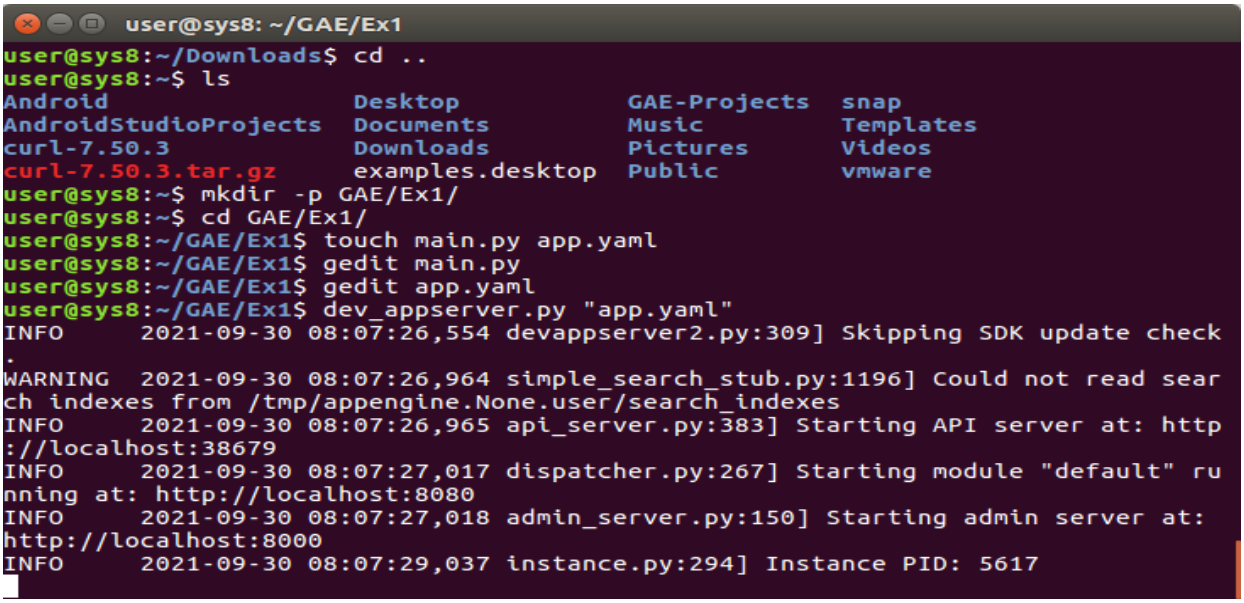
- Login to your google account and continue the process by accepting the terms and conditions
- Now run dev_appserver.py command in your terminal.



- Create a main.py file and app.yaml file in your desired folder
- Inside main.py write a simple hello world program



- Finally run the app using



$ dev_appserver.py “app.yaml”

- Go to **http://localhost:8000/**

Result :

Thus installation of Google App Engine is completed and hello world app using python is completed successfully

EXNO: 4

GAE LAUNCHER TO LAUNCH THE WEB APPLICATIONS.

AIM:

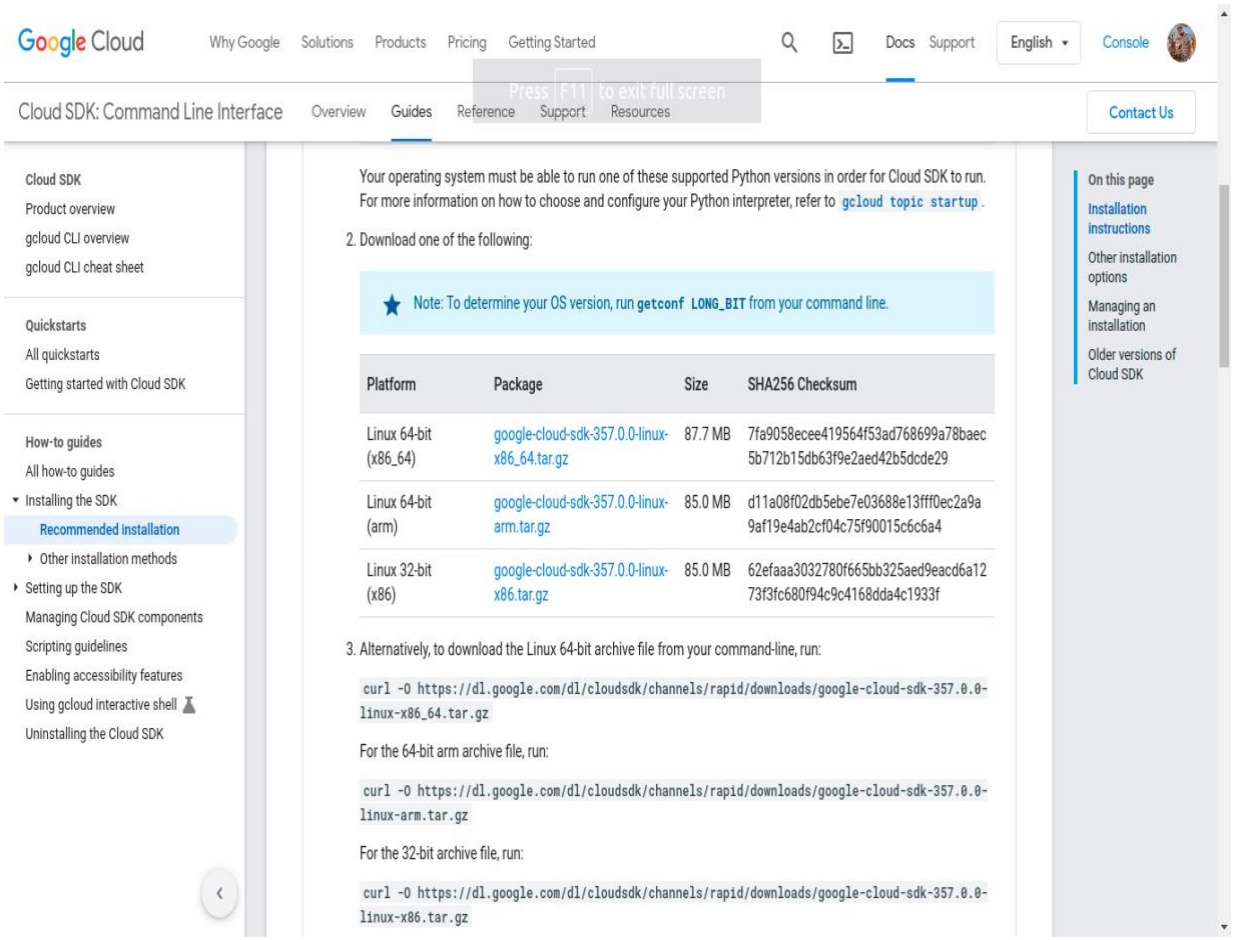
To use GAE launcher to launch the web applications.

PROCEDURE :

Step 1 – Go to Cloud SDK

https://cloud.google.com/sdk/docs/install

Step 2 – Download Linux 64-bit (x86_64)



Step 3 – Extract the Zip File from downloads

Step 4 – Open Terminal in Downloads

Step 5 – Run the following command

./google-cloud-sdk/install.sh

Step 6 – Init the cloud sdk using the following command

./google-cloud-sdk/bin/gcloud init

Step 7 – Run **"dev_appserver.py**" in the terminal

Step 8 – Create Project Folder, Download the zip file from the below link and unzip it.

https://drive.google.com/file/d/18gHSvne6r11yT70X6x7HtmJlpLTBieYq/view?usp=sharing

Step 9 - Create a App.yaml file

App.yaml

```
runtime: python27
api_version: 1
threadsafe: true
handlers:
- url: /
  static_files: project/index.html
  upload: project/index.html
- url: /
  static_dir: project
```

Step 10 - Go to the Project Folder and open the terminal

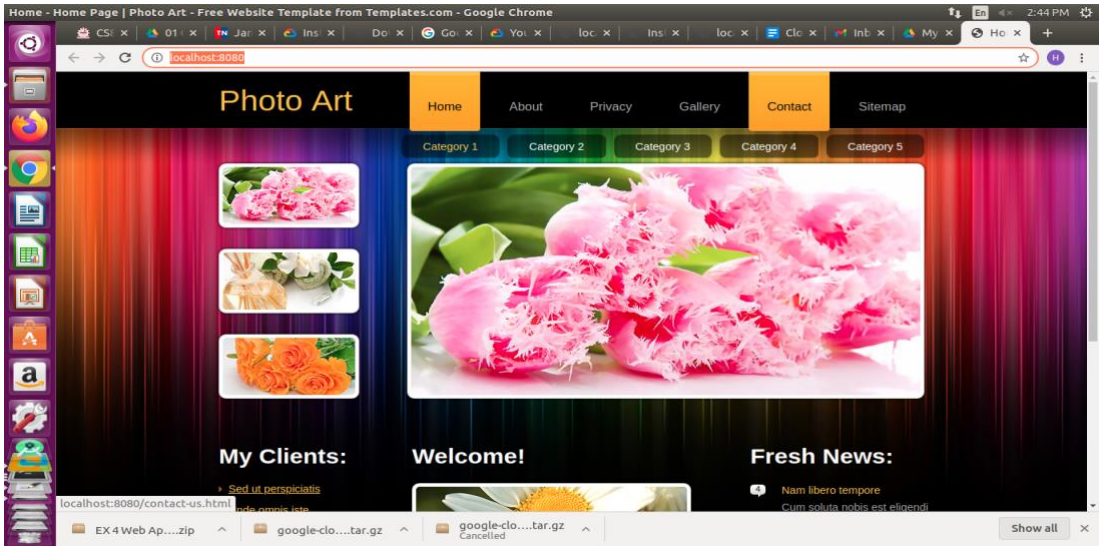
dev_appserver.py "app.yaml"

Step 11 - Go to http://localhost:8000

"If Localhost not opening then run this command"

fuser -n tcp -k 8080

OUTPUT :



RESULT :

Thus the GAE launcher has been successfully used to launch the web applications.

EXNO: 5

Study on CloudSim is a simulation toolkit

Aim:

To study about the tools and features of Cloud sim

Description:

CloudSim is a simulation toolkit that supports the modeling and simulation of the core functionality of cloud, like job/task queue, processing of events, creation of cloud entities(datacenter, datacenter 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 modeling and simulation of large scale computing environment as federated cloud data centers, 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 modeling cloud's service brokers, provisioning, and allocation policies.
- It supports the simulation of network connections among simulated system element

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 center node.
- Flexibility to switch between space shared and time shared allocation of processing cores to virtualized services.

The cloudsim allows to model and simulate the cloud system components, therefore to support its function different set of classes has been developed by its developers like:

- To simulating the regions and datacenters the class named "Datacenter.java" is available in org.cloudbus.cloudsim package.
- To simulate the workloads for cloud, the class named as "Cloudlet.java" is available in org.cloudbus.cloudsim package.
- To simulate the load balancing and policy-related implementation the classes named "DatacenterBroker.java", "CloudletScheduler.java", "VmAllocationPolicy.java", etc are available under org.cloudbus.cloudsim package.
- Now because all the different simulated hardware models are required to communicate with each other to share the simulation work updates for this cloudsim has implemented a discrete event simulation engine that keeps track of all the task assignments among the different simulated cloud components.

Cloudsim is used for

- Load Balancing of resources and tasks
- Task scheduling and its migrations
- Optimizing the Virtual machine allocation and placement policies
- Energy-aware Consolidations or Migrations of virtual machines
- Optimizing schemes for Network latencies for various cloud scenarios

Cloudsim simulation toolkit setup is easy. Before you start to setup CloudSim, following resources must be Installed/downloaded on the local system

- **Java Development Kit(JDK)**
- **Eclipse IDE for Java developers**
- **Download CloudSim source code**
- **Download Common Math libraries**

Result:

Thus the study on cloudsim simulation tool is done successfully.

EXNO: 5 a **SIMULATE A CLOUD SCENARIO USING CLOUDSIM**

Aim:

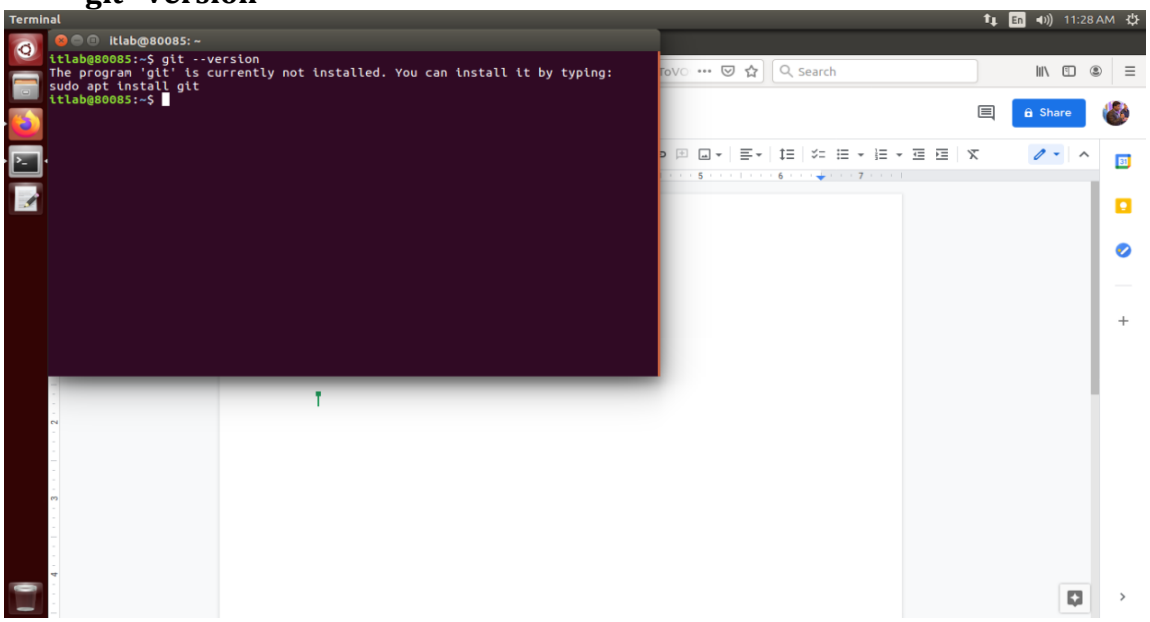
To simulate a cloud scenario using CloudSim and run a scheduling algorithm.

Description:

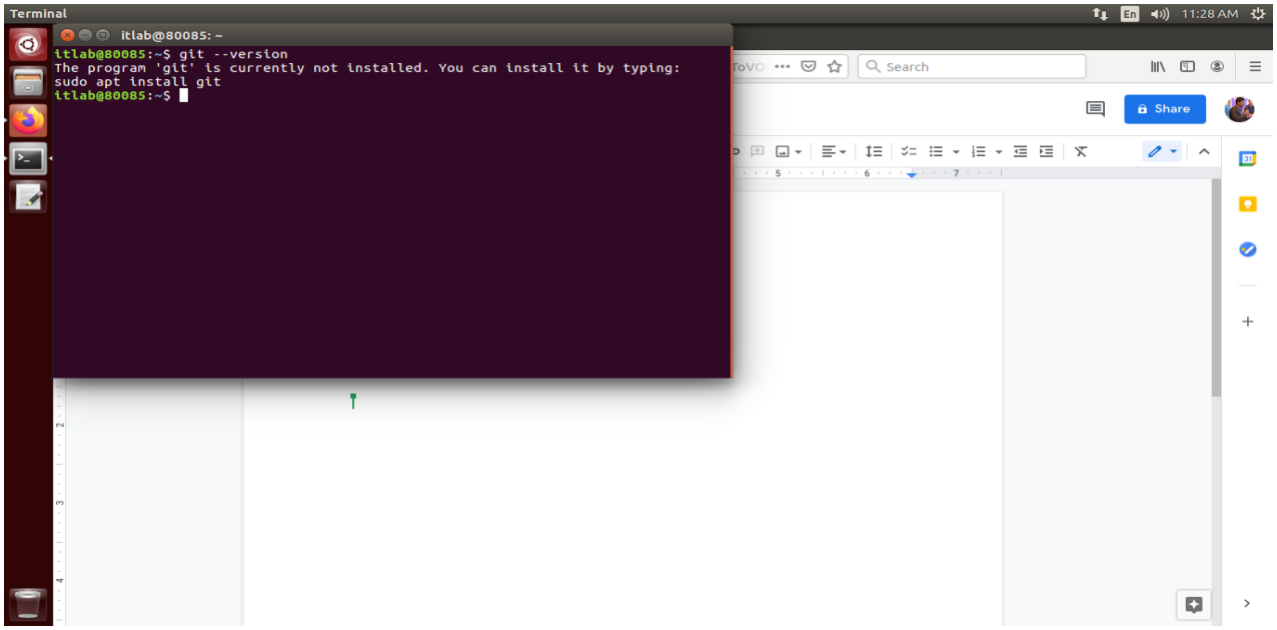
Cloudsim is a simulation toolkit that supports the modeling and simulation of the core functionality of the cloud. This article discusses the overview of the cloudsim toolkit and helps you get started along with reference to the relevant resources.

Procedure:

1. Check for git installation
   git –version

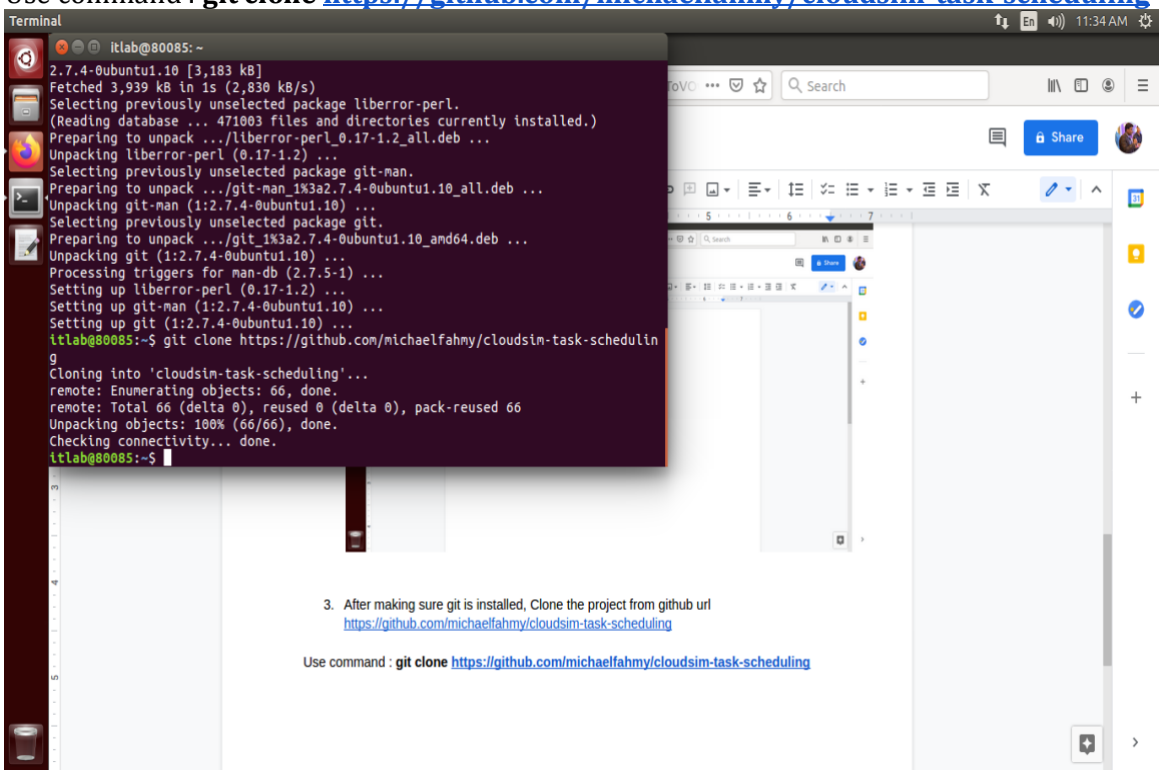


2. If git is not installed,
   Use **sudo apt-get install git**

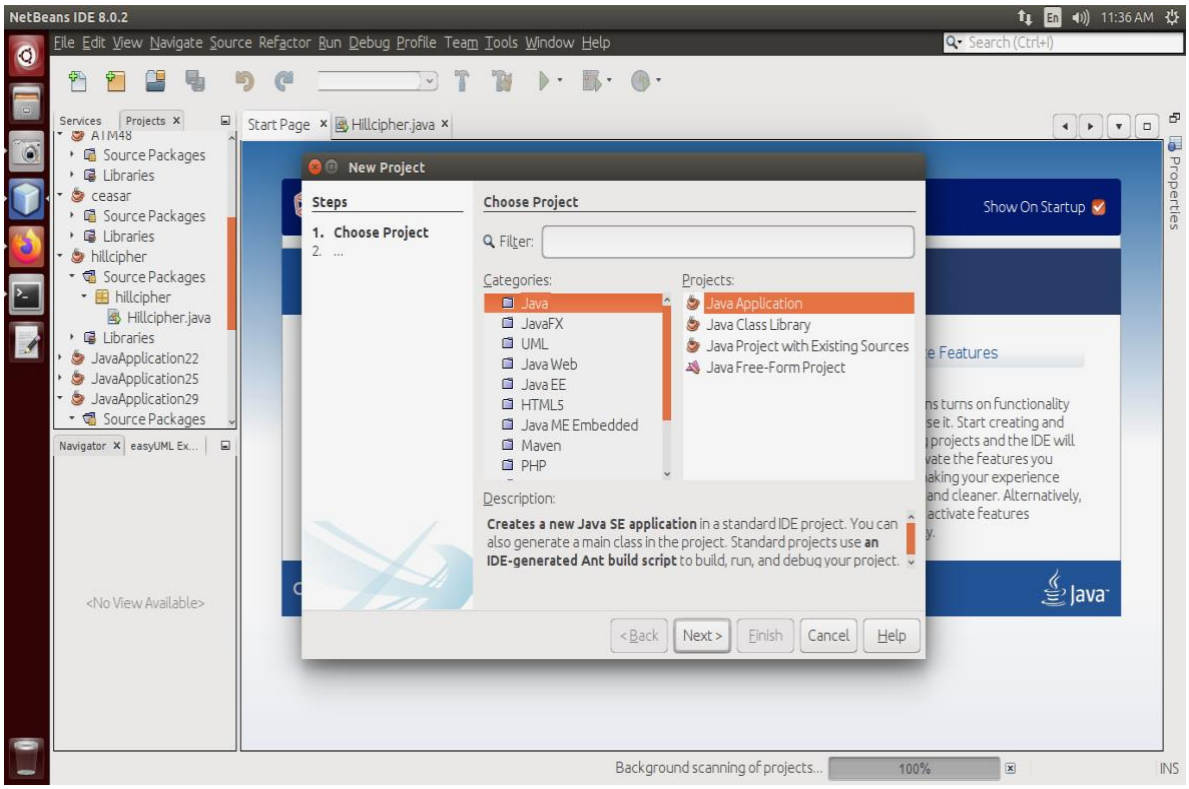


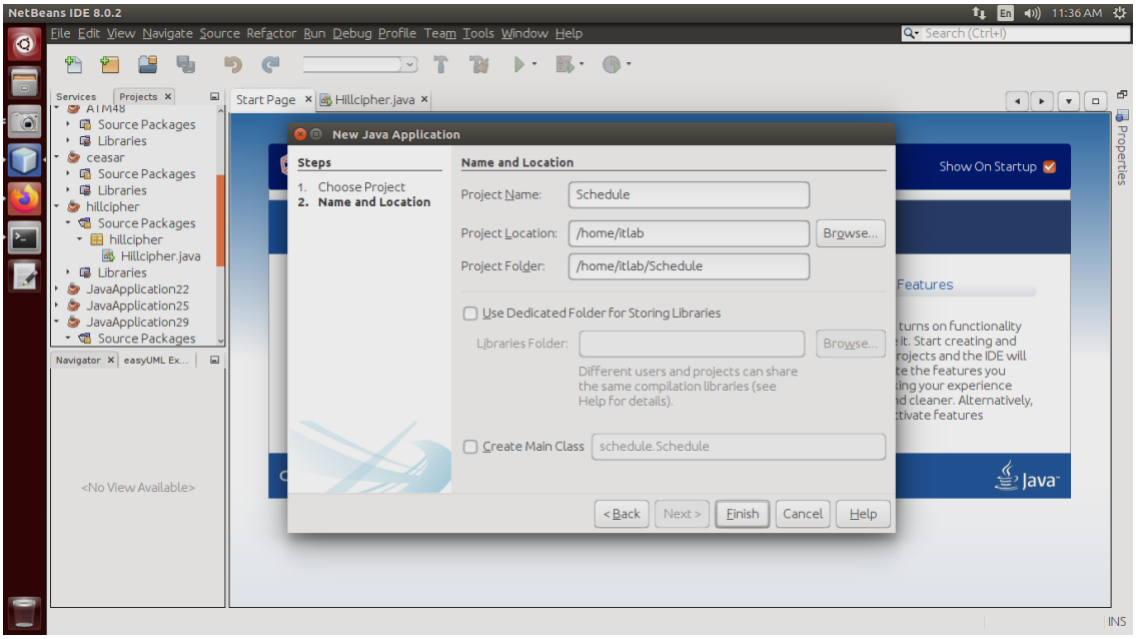
3. After making sure git is installed, Clone the project from github url https://github.com/michaelfahmy/cloudsim-task-scheduling

Use command : **git clone https://github.com/michaelfahmy/cloudsim-task-scheduling**

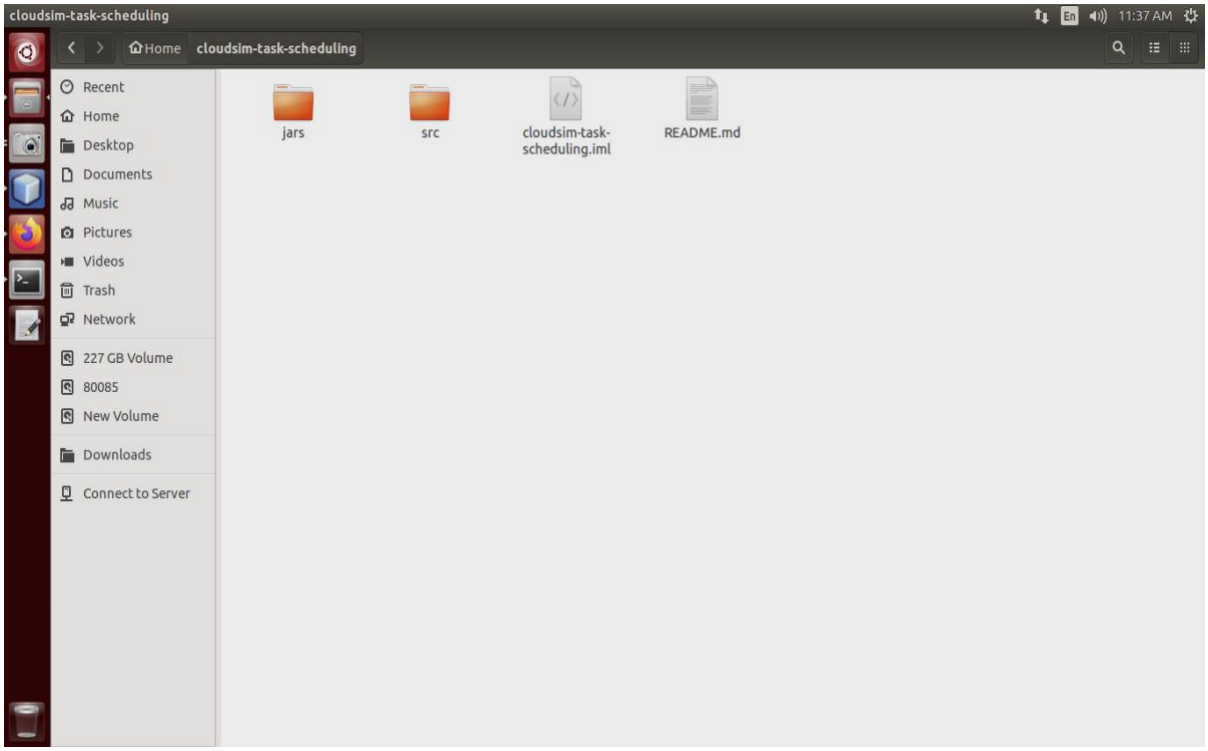


4. Open netbeans 8.x and create a new java project from files

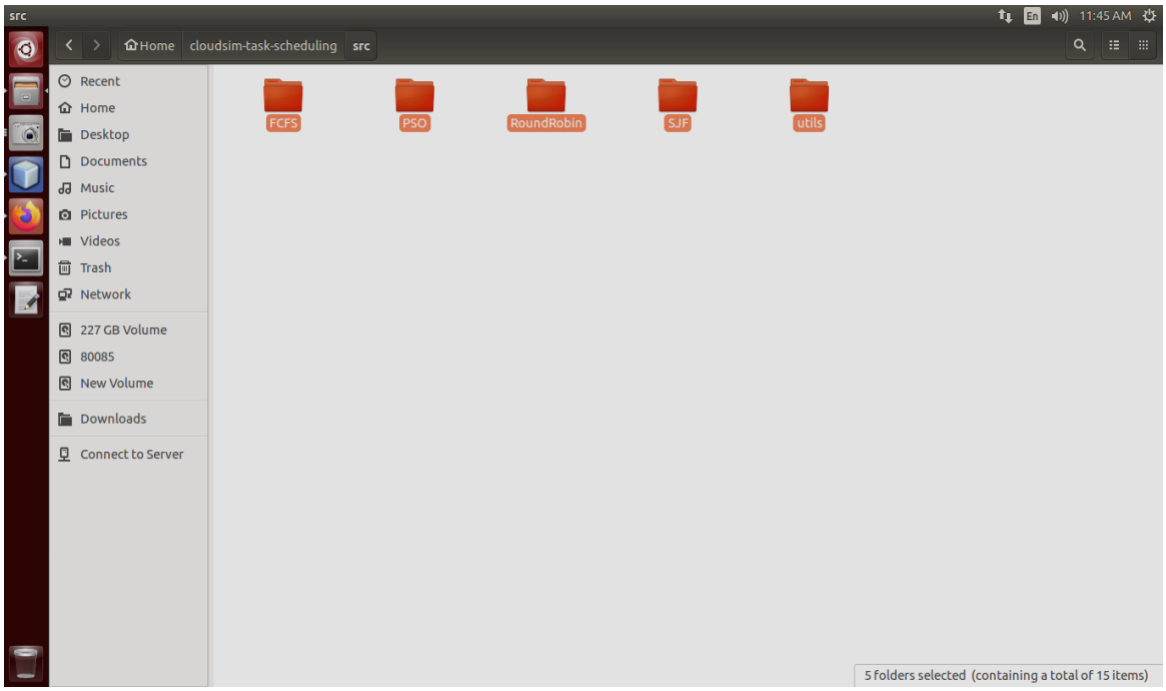




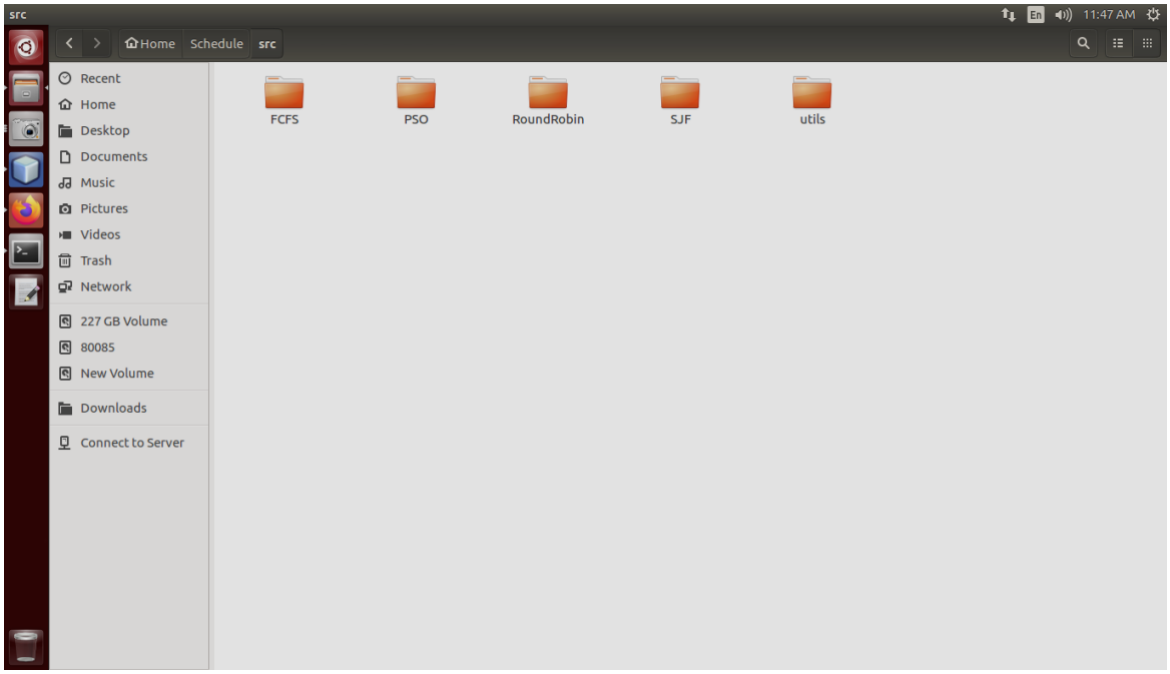
5. Now find the cloned project folder which generally in linux will be in /home/itlab



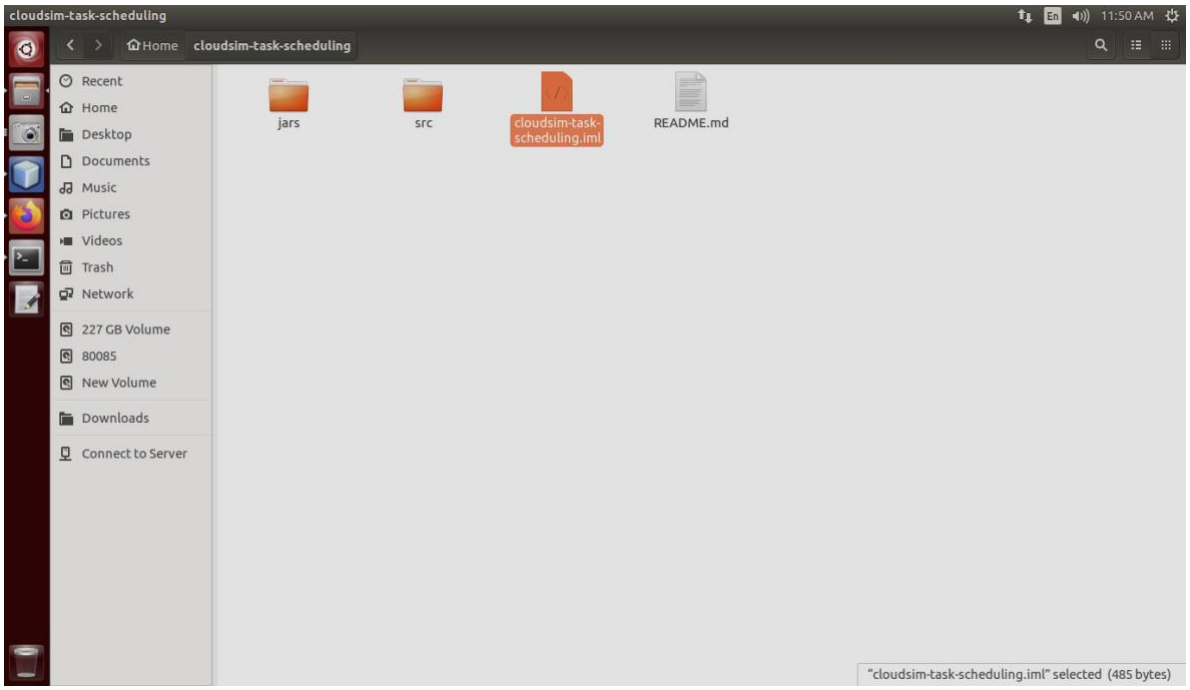
6. Open the src folder and copy the folders

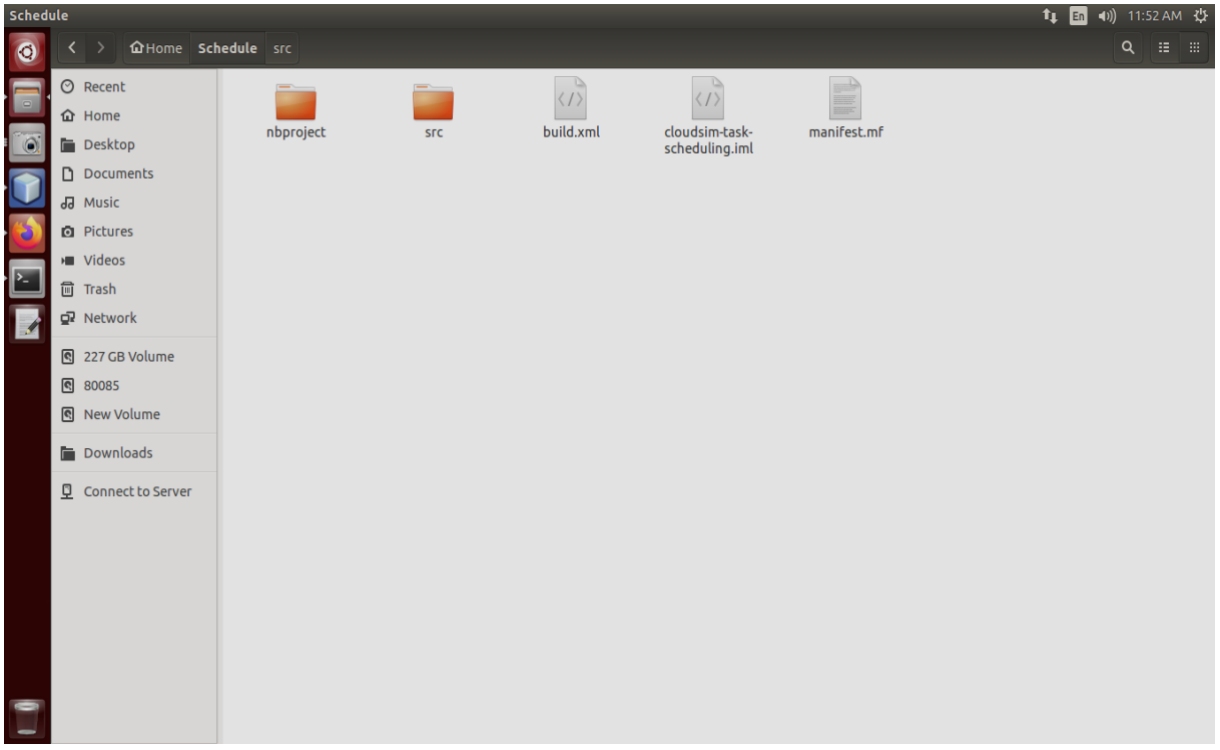


Paste the folders in the src folder of the newly created netbeans project

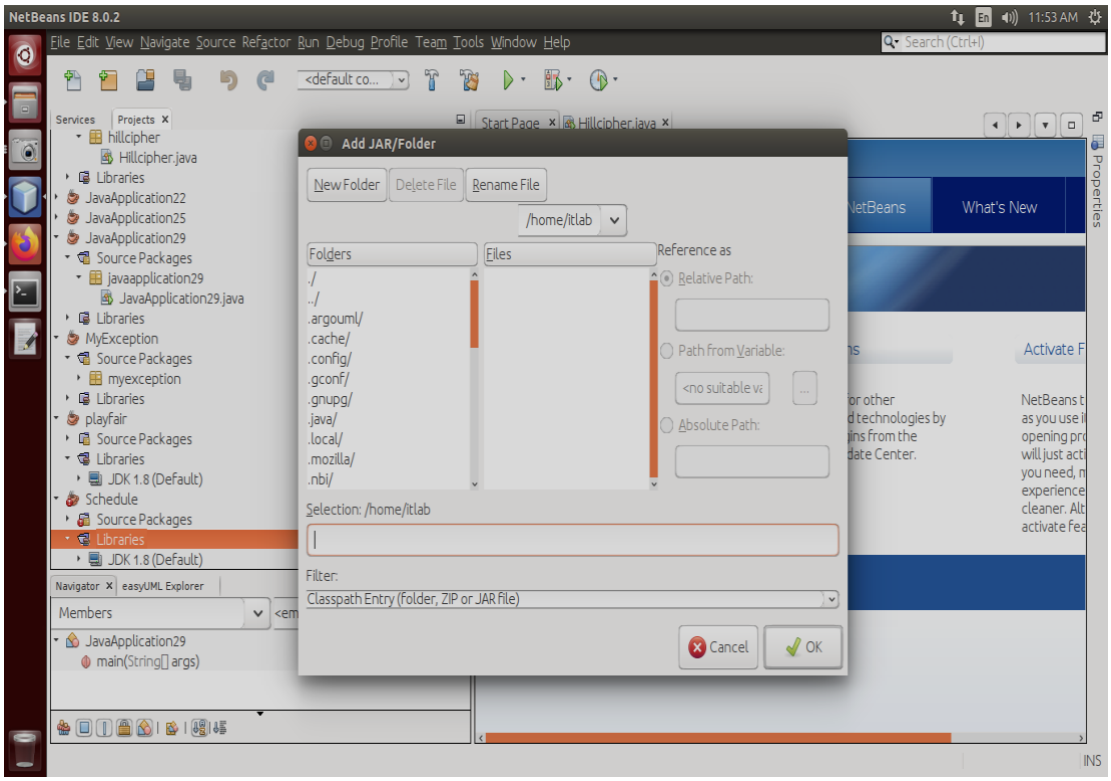


7. Also copy the cloudsim-task-scheduling.iml to the netbeans project

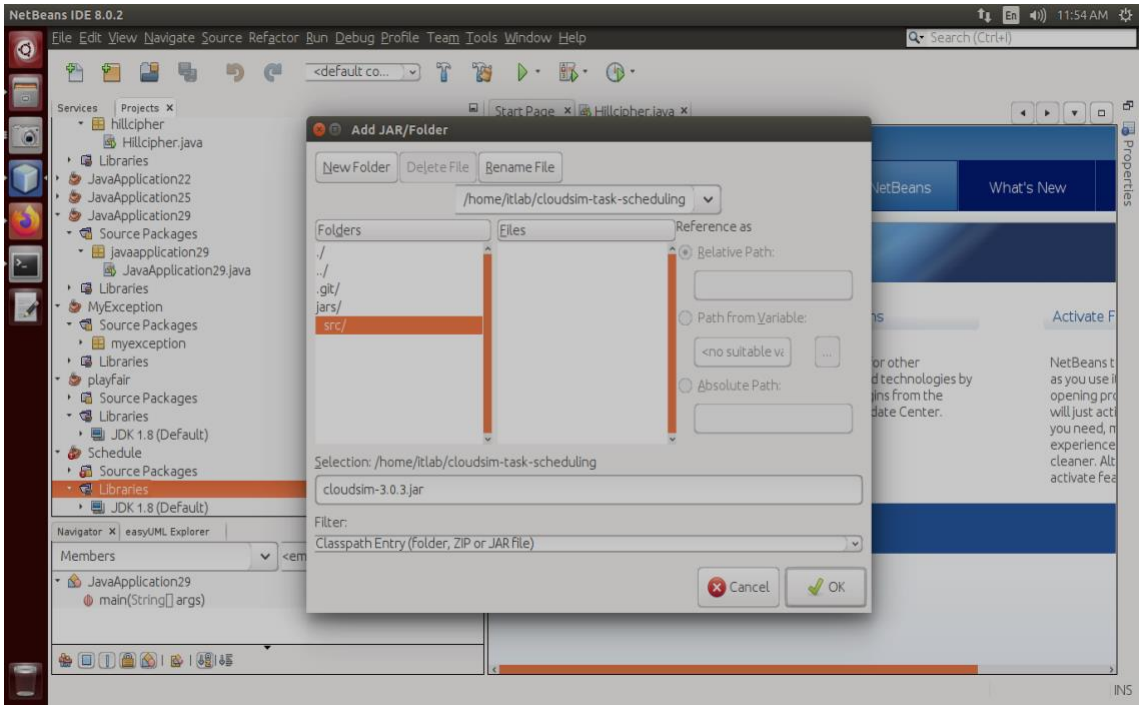




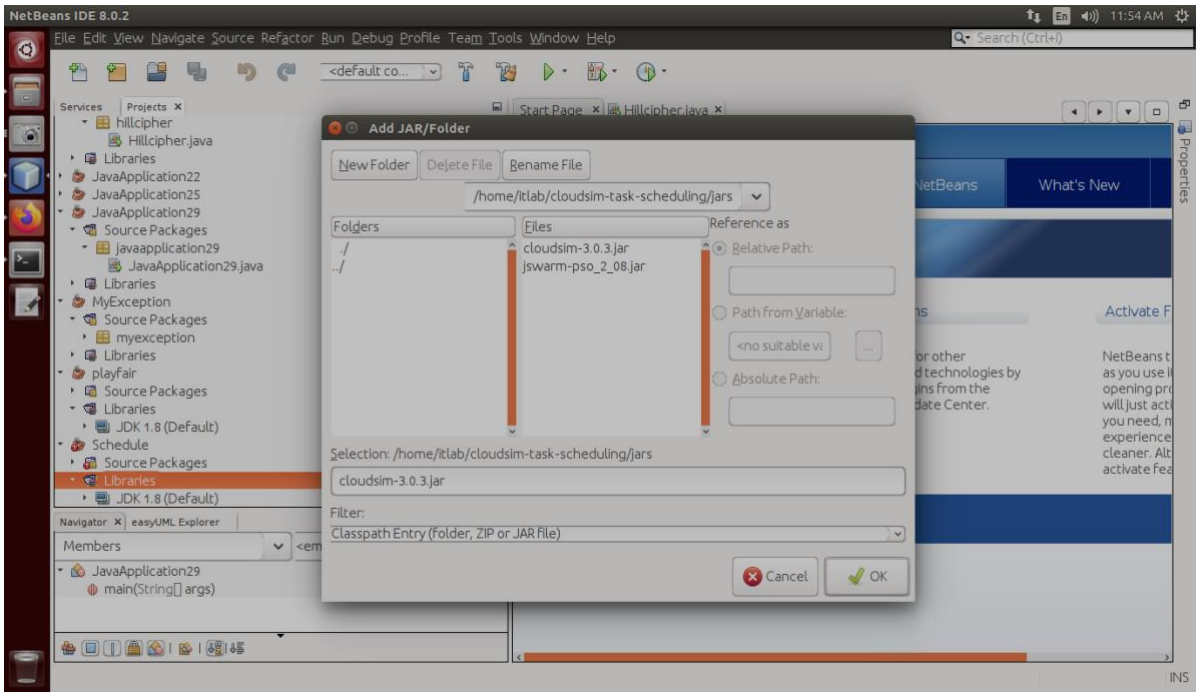
8. Go to netbeans and right click on project's libraries to add jars



Select the folder where you cloned the git project and go to jars



And double click



Select the jar file - cloudsim-3.0.3.jar and click Ok
Similarly add jswarm-pso_2_08.jar

9. To run any scheduling , expand the folder and right click on the file ending on _scheduler and run file

For FCFS, run FCFS_Scheduler,java

Program:

```
package org.cloudbus.cloudsim.examples;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerTimeShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
```

```
/**
 * An example showing how to create
 * scalable simulations.
 */
public class CloudSimExample6 {
        /** The cloudlet list. */
        private static List<Cloudlet> cloudletList;
        /** The vmlist. */

private static List<Vm> vmlist;

        private static List<Vm> createVM(int userId, int vms) {

                //Creates a container to store VMs. This list is passed to the broker later
                LinkedList<Vm> list = new LinkedList<Vm>();

                //VM Parameters
                long size = 10000; //image size (MB)
                int ram = 512; //vm memory (MB)
                int mips = 1000;
                long bw = 1000;
                int pesNumber = 1; //number of cpus
                String vmm = "Xen"; //VMM name

                //create VMs
                Vm[] vm = new Vm[vms];

                for(int i=0;i<vms;i++){
                        vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
                        //for creating a VM with a space shared scheduling policy for cloudlets:
                        //vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, priority, vmm, new
CloudletSchedulerSpaceShared());
                        list.add(vm[i]);
                }
                return list;
        }
        private static List<Cloudlet> createCloudlet(int userId, int cloudlets){
                // Creates a container to store Cloudlets
                LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

                //cloudlet parameters
                long length = 1000;
                long fileSize = 300;
                long outputSize = 300;
                int pesNumber = 1;
                UtilizationModel utilizationModel = new UtilizationModelFull();
                Cloudlet[] cloudlet = new Cloudlet[cloudlets];
                for(int i=0;i<cloudlets;i++){
                        cloudlet[i] = new Cloudlet(i, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
                        // setting the owner of these Cloudlets
                        cloudlet[i].setUserId(userId);
                        list.add(cloudlet[i]);
```

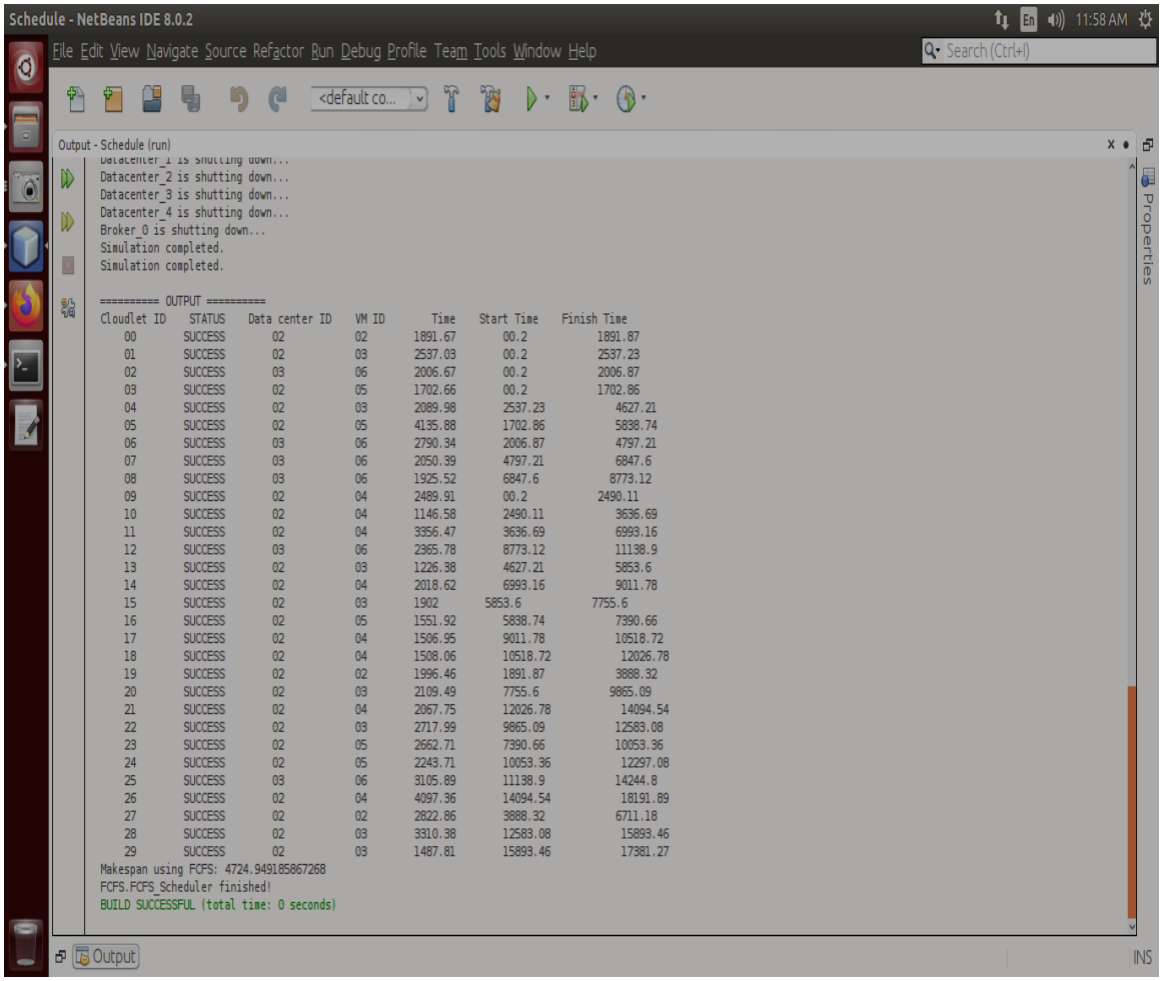
```
                }
                return list;
        }
        ////////////////////////// STATIC METHODS ///////////////////////
        /**
         * Creates main() to run this example
         */
        public static void main(String[] args) {
                Log.printLine("Starting CloudSimExample6...");

                try {
                        // First step: Initialize the CloudSim package. It should be called
                        // before creating any entities.
                        int num_user = 1; // number of grid 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
                        //Datacenters are the resource providers in CloudSim. We need at list one of
them to run a CloudSim simulation
                        @SuppressWarnings("unused")
                        Datacenter datacenter0 = createDatacenter("Datacenter_0");
                        @SuppressWarnings("unused")
                        Datacenter datacenter1 = createDatacenter("Datacenter_1");

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

                        //Fourth step: Create VMs and Cloudlets and send them to broker
                        vmlist = createVM(brokerId,20); //creating 20 vms
                        cloudletList = createCloudlet(brokerId,40); // creating 40 cloudlets
                        broker.submitVmList(vmlist);
                        broker.submitCloudletList(cloudletList);
                        // Fifth 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("CloudSimExample6 finished!");
                }
                catch (Exception e)
                {
                     Log.printLine("The simulation has been terminated due to an unexpected
                       }

        private static Datacenter createDatacenter(String name){

                // Here are the steps needed to create a PowerDatacenter:
                // 1. We need to create a list to store one or more
                //   Machines
                List<Host> hostList = new ArrayList<Host>();
```

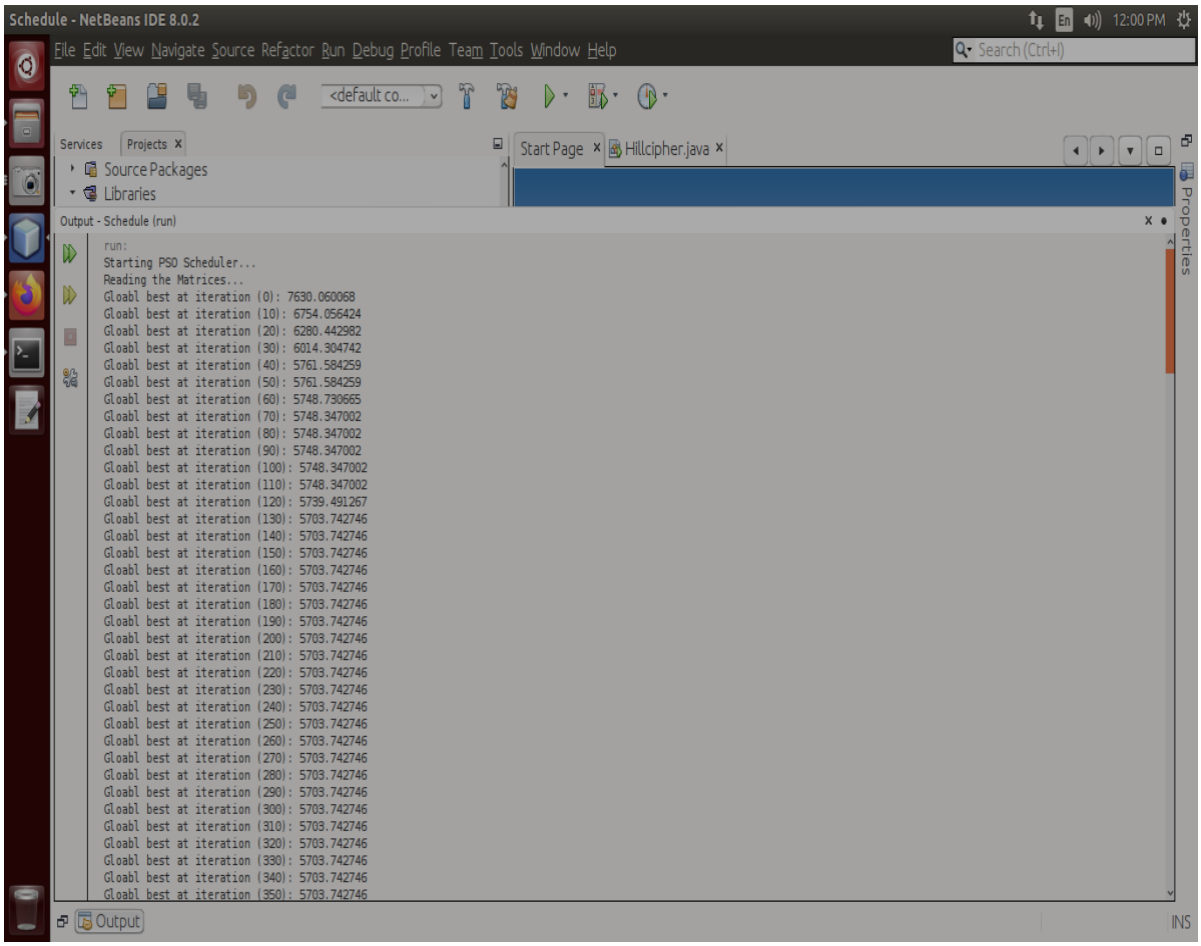
```
                // 2. A Machine contains one or more PEs or CPUs/Cores. Therefore, should
                //   create a list to store these PEs before creating
                //   a Machine.
                List<Pe> peList1 = new ArrayList<Pe>();
                int mips = 1000;
                // 3. Create PEs and add these into the list.
                //for a quad-core machine, a list of 4 PEs is required:
                peList1.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and
MIPS Rating
  peList1.add(new       Pe(1,       new       PeProvisionerSimple(mips)));
  peList1.add(new       Pe(2,       new       PeProvisionerSimple(mips)));
  peList1.add(new Pe(3, new PeProvisionerSimple(mips)));
  //Another list, for a dual-core machine List<Pe> peList2
  = new ArrayList<Pe>();
  peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
  peList2.add(new Pe(1, new PeProvisionerSimple(mips)));
  //4. Create Hosts with its id and list of PEs and add them to the list of machinesint hostId=0;
  int ram = 2048; //host memory (MB) long storage =
  1000000; //host storageint bw = 10000;
  hostList.add(
                new Host(  // This is our
                first machinehostId++;
                hostList.add(
                        new Host(
                                hostId,
                                new RamProvisionerSimple(ram),
                                new BwProvisionerSimple(bw),
                                storage,
                                peList2,
                                new VmSchedulerTimeShared(peList2)
                        )
                ); // Second machine
                //To create a host with a space-shared allocation policy for PEs to VMs:
                //hostList.add(
        //              new Host(
        //                      hostId,
        //                      new CpuProvisionerSimple(peList1),
        //                      new RamProvisionerSimple(ram),
        //                      new BwProvisionerSimple(bw),
        //                      storage,
        //                      new VmSchedulerSpaceShared(peList1)

                                new wProvisionerSimple(bw),
                                storage,
                                peList1,
                                new VmSchedulerTimeShared(peList1)

                Create a DatacenterCharacteristics object that stores the
                //   properties of a data center: architecture, OS, list of
  Resource //    Machines, allocation policy: time- or space-shared, time zone
  //   and its price (G$/Pe time unit).
  String arch = "x86";      // system architectureString os =
  "Linux";                    // operating system String vmm =
  "Xen";
  double time_zone = 10.0;        // time zone this resource located
```

```
                double cost = 3.0;              // the cost of using processing in this resource
                double costPerMem = 0.05;       // the cost of using memory in this resource
                double costPerStorage = 0.1; // the cost of using storage in this resource
                double costPerBw = 0.1;         // the cost of using bw in this
                                                I
                LinkedList<Storage> storageList = new LinkedList<Storage>();          //we are not
                                                ,
adding SAN devices by now

                DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
        arch, os, vmm, hostList, time_zone, cost, costPerMem, costPerStorage, costPerBw);

                // 6. Finally, we need to create a PowerDatacenter object.
                Datacenter datacenter = null;
                try {
                        datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return datacenter;
        }
        //We strongly encourage users to develop their own broker policies, to submit vms and
cloudlets according
        //to the specific rules of the simulated scenario
        private static DatacenterBroker createBroker(){

                DatacenterBroker broker = null;
                try {
                        broker = new DatacenterBroker("Broker");
                } catch (Exception e) {
                        e.printStackTrace();
                        return null;
                }return broker;
        }
        /**
         * Prints the Cloudlet objects
         * @param list  list of Cloudlets
         */
        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 + 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 + indent +
dft.format(cloudlet.getActualCPUTime()) +
                                    indent + indent +
dft.format(cloudlet.getExecStartTime())+ indent + indent + indent +
dft.format(cloudlet.getFinishTime()));
                        }
                    }

            }


    }
```

OUTPUT :





Result:

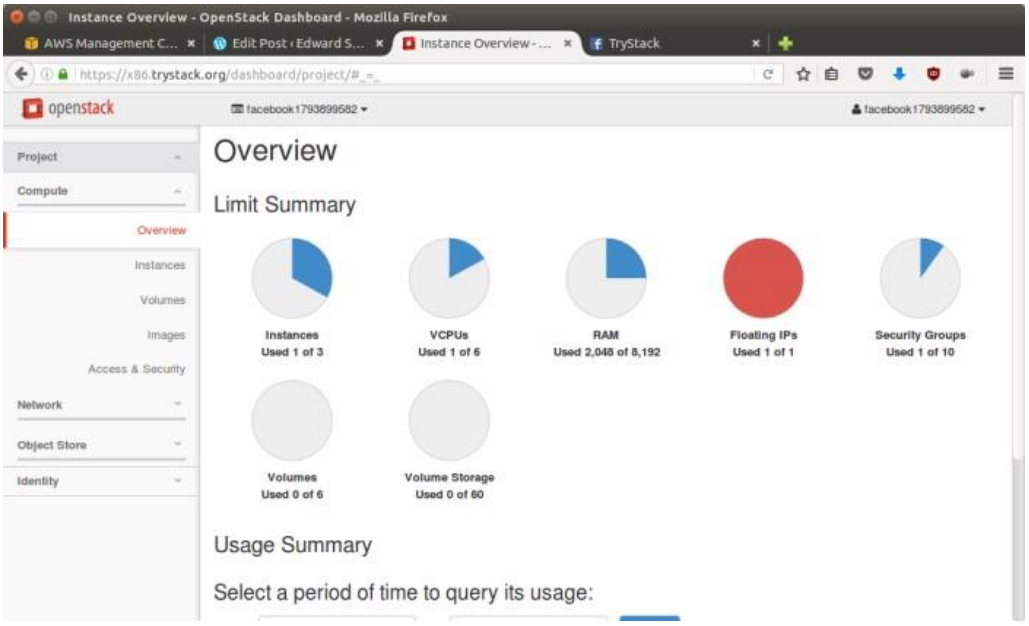
Thus the study on cloudsim simulation tool is done successfully.

Ex:No: 6 Procedure to launch virtual machine using trystack

Aim:
To understand the procedure to launch virtual machine using trystack.
Description:
OpenStack is an open-source software cloud computing platform. OpenStack is primarily used for deploying an infrastructure as a service (IaaS) solution like Amazon Web Service (AWS). In other words, you can *make your own AWS* by using OpenStack.
Procedure:

1. **Open trystack page using trystack.org**

2. **Login to trystack by creating account.**



3. **Create Network**
   a) Go to Network > Networks and then click Create Network.

b) In Network tab, fill Network Name for example internal and then click Next.
c) In Subnet tab,
d) Fill Network Address with appropriate CIDR, for example 192.168.1.0/24. Use private network CIDR block as the best practice.
e) Select IP Version with appropriate IP version, in this case IPv4.
f) Click Next.
g) In Subnet Details tab, fill DNS Name Servers with 8.8.8.8 (Google DNS) and then click Create.3.

4. Create Instance

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

5. Create Router

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

6. Configure Floating IP Address

Floating IP address is public IP address. It makes your instance is accessible from the internet. When
you launch your instance, the instance will have a private network IP, but no public IP. In OpenStack,the public IPs is collected in a pool and managed by admin.

a) Go to Compute > Instance.
b) In one of your instances, click More > Associate Floating IP.
c) In IP Address, click Plus [+].
d) Select Pool to external and then click Allocate IP.
e) Click Associate.
f) Now you will get a public IP, e.g. 8.21.28.120, for your instance.

7. Configure Access & Security

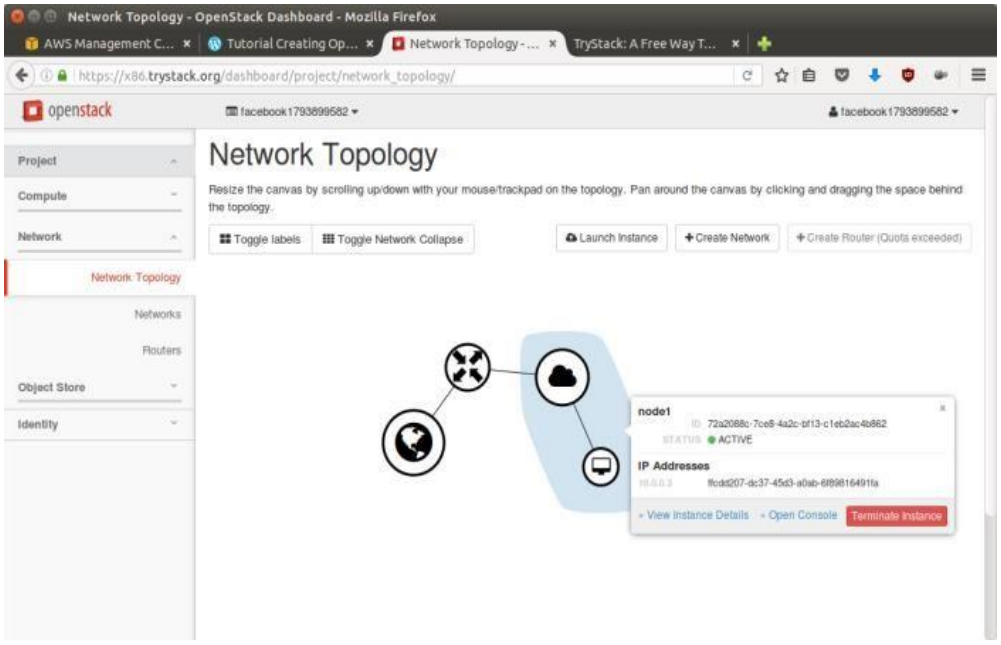
OpenStack has a feature like a firewall. It can whitelist/blacklist your in/out connection. It iscalled *Security Group*.

a) Go to Compute > Access & Security and then open Security Groups tab.
b) In default row, click Manage Rules.
c) Click Add Rule, choose ALL ICMP rule to enable ping into your instance, and then click Add.
d) Click Add Rule, choose HTTP rule to open HTTP port (port 80), and then click Add.
e) Click Add Rule, choose SSH rule to open SSH port (port 22), and then click Add.
f) You can open other ports by creating new rules.

8. SSH to Your Instance

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

OUTPUT :



Result: Thus the procedure to launch virtual machine in trystack is studied