SRI SAI RAM ENGINEERING COLLEGE

SAI LEO NAGAR WEST TAMBARAM, CHENNAI-44

NAME :

REGISTER NUMBER:

IT8711 – FOSS AND CLOUD COMPUTING LABORATORY (IV YEAR/ VII SEM)

(BATCH: 2019 – 2023)

B.TECH INFORMATION TECHNOLOGY *ACADEMIC YEAR: 2022 – 2023*

Sri SAI RAM ENGINEERING COLLEGE

An Autonomous Institution | Affiliated to Anna University & Approved by AICTE, New Delhi

Accredited by NBA and NAAC "A+" | An ISO 9001:2015 Certified and MHRD NIRF ranked institution

Sai Leo Nagar, West Tambaram, Chennai - 600 044. www.sairam.edu.in

Founder Chairman : MJF. Ln. Leo Muthu

Certificate

Register No: _____________

Certified that this is the Bonafide Record of work done by Mr./Ms. ______________________________________ in the B.Tech **Degree Course** Information Technology **in the** IT8711 – FOSS and Cloud Computing laboratory **during the academic year** 2022-2023 **.**

Station : Chennai – 600 044
Date :

STAFF IN-CHARGE **HEAD OF THE DEPARTMENT**

Submitted for University Practical Examination held on ________

at **Sri Sai Ram Engineering College**, Chennai – 600 044.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

Ex.No:1	CREATE C PROGRAM USING GCC AND MAKE COMMAND
Date: 17.10.22	

AIM:

To create a C Program using gcc and make commands.

ALGORITHM:

1. Start.
2. Create a main file and the submodule files with the extension .c
3. Now type the appropriate C program inside the file we have created.
4. Create the Makefile to compile the submodules to object files and finally link to produce an executable file.
5. To Compile the program $ make all
6. To Execute the program $ ./calculator
7. The output of the program will be displayed on the screen.
8. Stop

PROGRAM:

main.c

```
#include<stdio.h>
#include "main.h"

int main(){
   int first,second,choice;
   printf("Enter First number : ");
   scanf("%d",&first);
   printf("Enter second number : ");
   scanf("%d",&second);
   printf("Operation       | Choice\n");
   printf("Addition        |  1    \n");
   printf("Subtraction     |  2    \n");
   printf("Multiplication  |  3    \n");
   printf("Division        |  4    \n");
   printf("Enter your choice : ");
   scanf("%d",&choice);
   switch(choice){
      case 1:
                  add(first,second);
                  break;
```

```
    case 2:
            subtract(first,second);
            break;
    case 3:
            multiply(first,second);
            break;
    case 4:
            divide(first,second);
            break;
    default:
            printf("Invalid Operands");
  }
  printf("\n");
  return 0;
}
```

main.h

```
void add(int,int);
void subtract(int,int);
void multiply(int,int);
void divide(int,int);
```

add.c

```
#include<stdio.h>
void add(int a,int b){
  printf("The value of %d + %d is %d",a,b,a+b);
}
```

subtract.c

```
#include<stdio.h>
void subtract(int a,int b){
  printf("The value of %d - %d is %d",a,b,a-b);
}
```

multiply.c

```
#include<stdio.h>
void multiply(int a,int b){
  printf("The value of %d * %d is %d",a,b,a*b);
}
```

divide.c

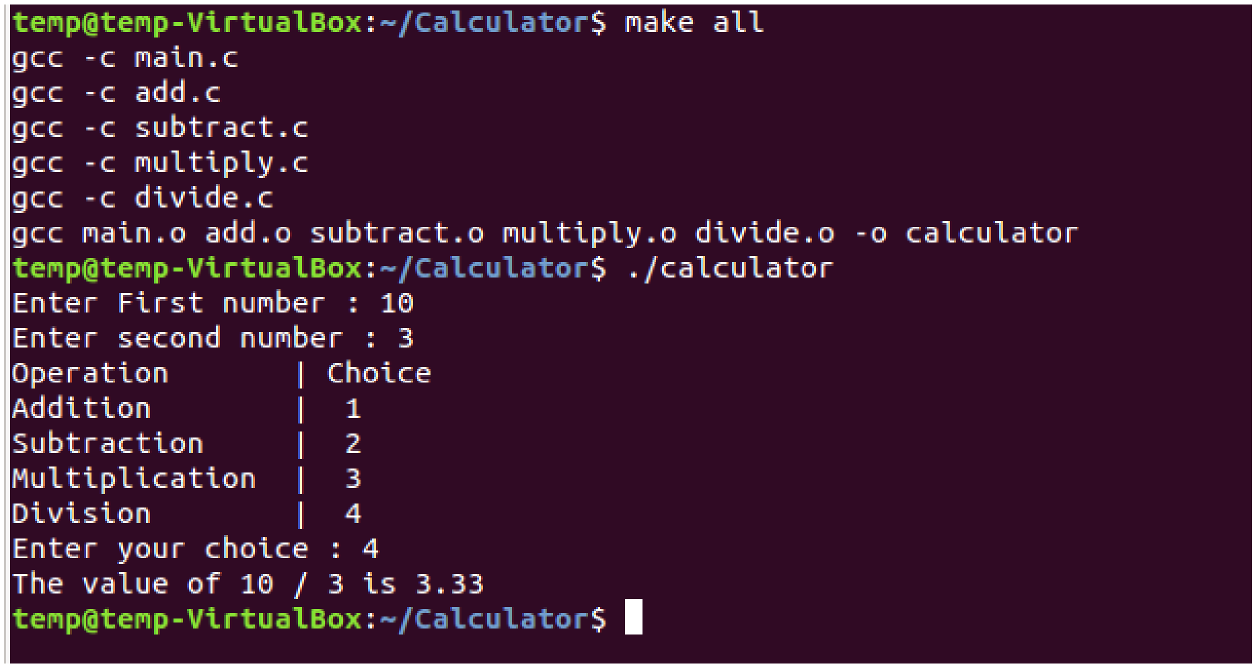
```
#include<stdio.h>
void divide(int a,int b){
  printf("The value of %d / %d is %.2f",a,b,(float)a/b);
}
```

Makefile

```
all: main.o main.h add.o subtract.o multiply.o divide.o
	gcc main.o add.o subtract.o multiply.o divide.o -o calculator
main.o: main.c
```

```
	gcc -c main.c
add.o: add.c
	gcc -c add.c
subtract.o: subtract.c
	gcc -c subtract.c
multiply.o: multiply.c
	gcc -c multiply.c
divide.o: divide.c
	gcc -c divide.c
clean:
	rm *.o
```

Output:



```
temp@temp-VirtualBox:~/Calculator$ make all
gcc -c main.c
gcc -c add.c
gcc -c subtract.c
gcc -c multiply.c
gcc -c divide.c
gcc main.o add.o subtract.o multiply.o divide.o -o calculator
temp@temp-VirtualBox:~/Calculator$ ./calculator
Enter First number : 10
Enter second number : 3
Operation        | Choice
Addition         |  1
Subtraction      |  2
Multiplication   |  3
Division         |  4
Enter your choice : 4
The value of 10 / 3 is 3.33
temp@temp-VirtualBox:~/Calculator$
```

RESULT:

Thus the program to create a C program using gcc and make command is completed and executed successfully.

Ex.No:2	USE VERSION CONTROL SYSTEMS COMMAND TO CLONE, COMMIT, PUSH, FETCH, PULL CHECKOUT, RESET AND DELETE REPOSITORIES
Date: 29.08.22	

AIM:

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

ALGORITHM:

1. Start
2. Install the necessary packages for git to work.
3. Then, create a Github account and log into the same for git to work.
4. Now, type the commands provided below to push, pull, commit, clone, fetch, reset and delete a repository in Github.
5. The output will appear in the git cmd as shown below.
6. Stop.

COMMANDS:

The commands are,

➢ Clone
➢ Commit
➢ Push
➢ Fetch
➢ Pull
➢ Checkout
➢ Reset
➢ Delete

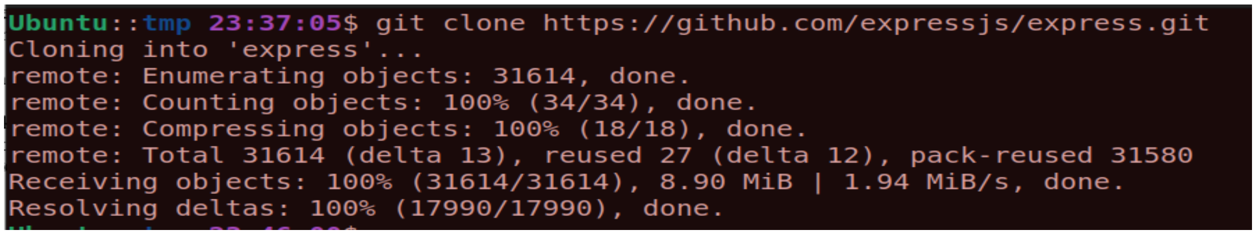
Clone:

The git clone command is used to create a copy of a specific repository or branch within a repository.

Command:

```
git clone https://github.com/github/training-kit.git
```

Output:



```
Ubuntu::tmp 23:37:05$ git clone https://github.com/expressjs/express.git
Cloning into 'express'...
remote: Enumerating objects: 31614, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 31614 (delta 13), reused 27 (delta 12), pack-reused 31580
Receiving objects: 100% (31614/31614), 8.90 MiB | 1.94 MiB/s, done.
Resolving deltas: 100% (17990/17990), done.
```

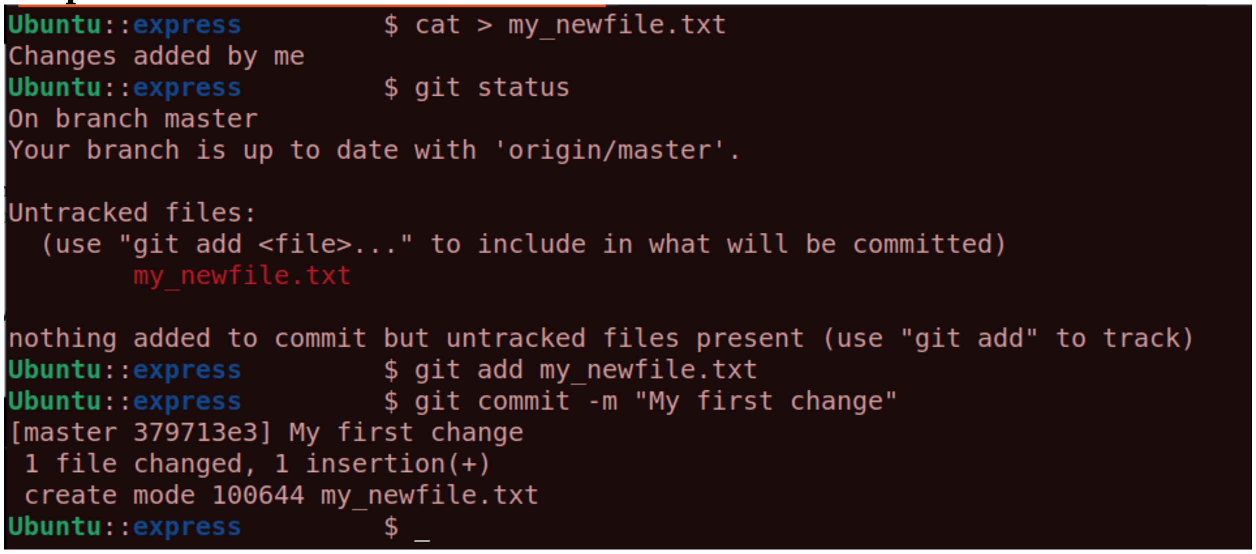
Commit:

git commit creates a commit, which is like a snapshot of your repository. Commits include lots of metadata in addition to the contents and message, like the author, timestamp, and more.

Command:

git commit -m "update the README.md with link to contributing guide"

Output:



```
Ubuntu::express         $ cat > my_newfile.txt
Changes added by me
Ubuntu::express         $ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        my_newfile.txt

nothing added to commit but untracked files present (use "git add" to track)
Ubuntu::express         $ git add my_newfile.txt
Ubuntu::express         $ git commit -m "My first change"
[master 379713e3] My first change
 1 file changed, 1 insertion(+)
 create mode 100644 my_newfile.txt
Ubuntu::express         $ _
```

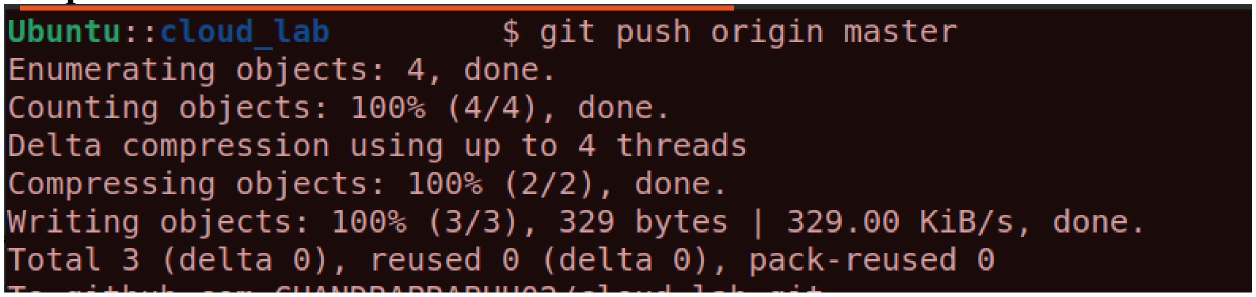
Push:

git push updates the remote branch with local commits.

Command:

git push origin master

Output:



```
Ubuntu::cloud_lab         $ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 329 bytes | 329.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

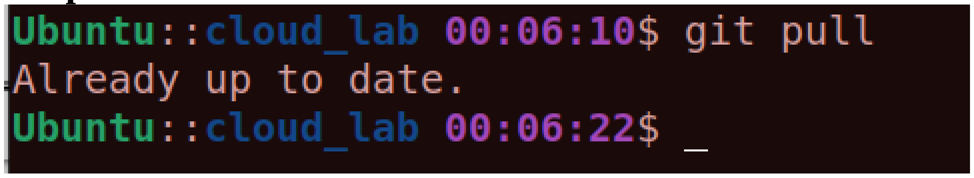
Pull

git pull is used to update changes made in your local repository to the remote repository on Github. If no branch name is specified, by default it will make pull request to the current repository.

Command:

git pull

Output:



```
Ubuntu::cloud_lab 00:06:10$ git pull
Already up to date.
Ubuntu::cloud_lab 00:06:22$ _
```

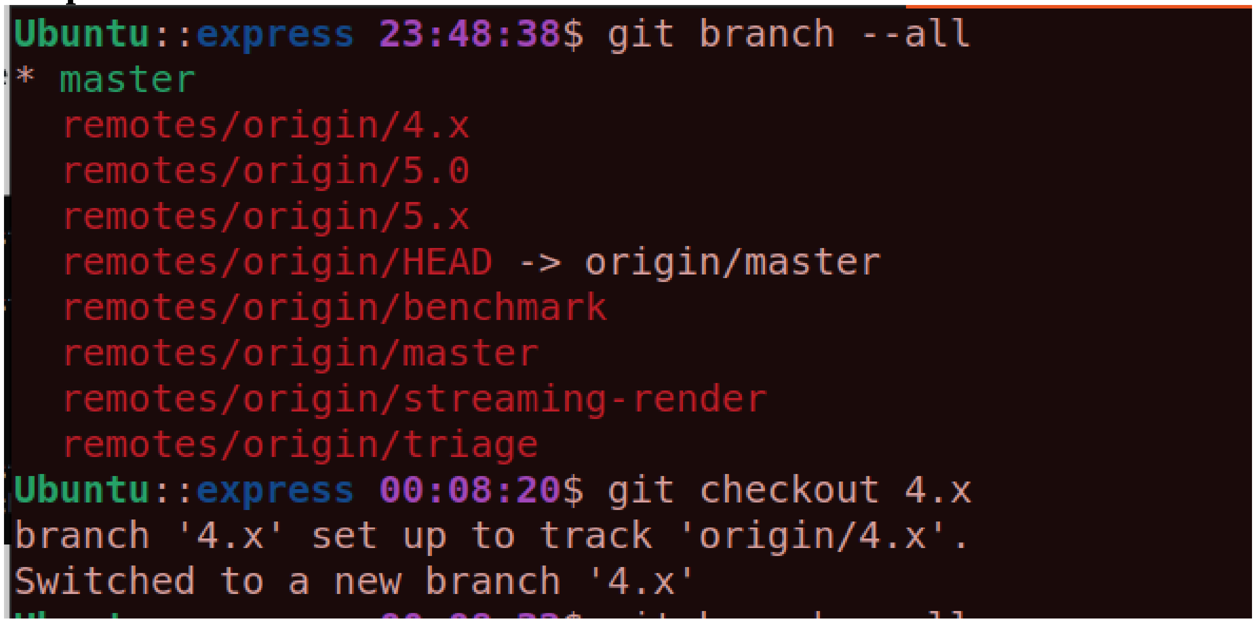
Checkout:

Use git checkout on the command line to create a new branch, change your current working branch to a different branch, or even to switch to a different version of a file from a different branch.

Command:

git checkout branchtest

Output:



```
Ubuntu::express 23:48:38$ git branch --all
* master
  remotes/origin/4.x
  remotes/origin/5.0
  remotes/origin/5.x
  remotes/origin/HEAD -> origin/master
  remotes/origin/benchmark
  remotes/origin/master
  remotes/origin/streaming-render
  remotes/origin/triage
Ubuntu::express 00:08:20$ git checkout 4.x
branch '4.x' set up to track 'origin/4.x'.
Switched to a new branch '4.x'
```

Reset:

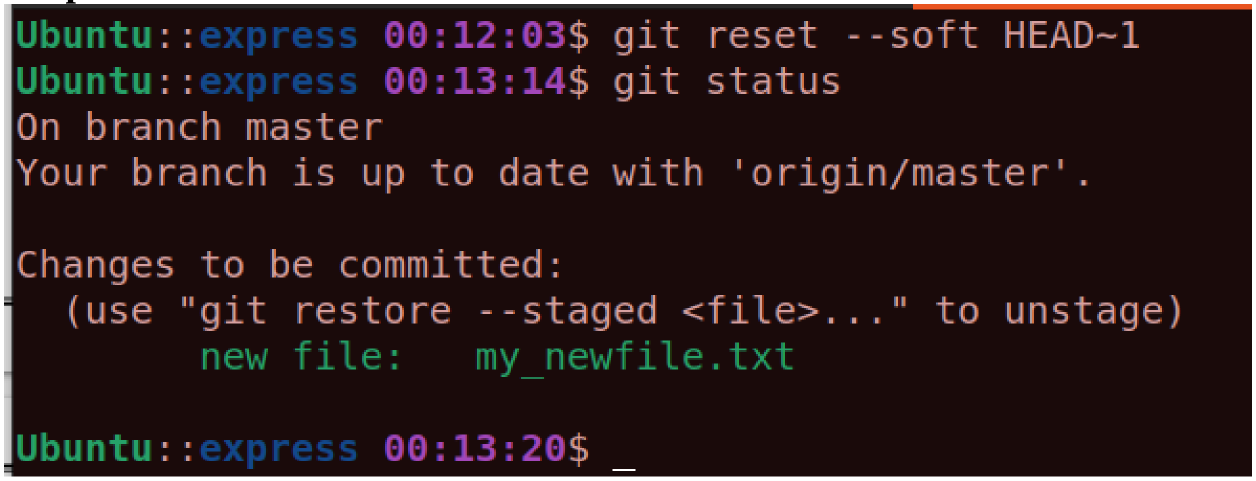
The term reset stands for undoing changes. The git reset command is used to reset the changes. The git reset command has three core forms of invocation. These forms are as follows:

- **Soft**
- **Mixed**
- **Hard**

Command:

git reset <option> <commit-sha>

Output:

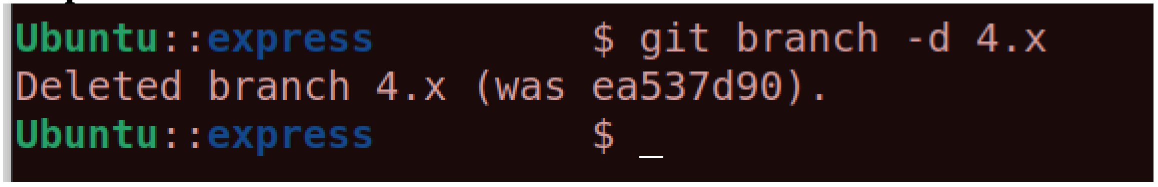


```
Ubuntu::express 00:12:03$ git reset --soft HEAD~1
Ubuntu::express 00:13:14$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   my_newfile.txt

Ubuntu::express 00:13:20$ _
```

Delete:
This command is used to delete a local repository or a remote repository from Github.

Command:
git branch -d testbranch

Output:



```
Ubuntu::express         $ git branch -d 4.x
Deleted branch 4.x (was ea537d90).
Ubuntu::express         $ _
```

RESULT:

Thus the program to use version control systems commands to clone, commit, push, fetch, pull, checkout, reset, and delete repositories have been completed successfully.

Ex. No: 3	INSTALLING VIRTUAL BOX WITH DIFFERENT FLAVORS OF OPERATING SYSTEMS
Date: 22.08.22	

AIM:

To Install Oracle Virtualbox and then install and run Ubuntu Linux in the Virtual Machine on top of Windows Operating System .

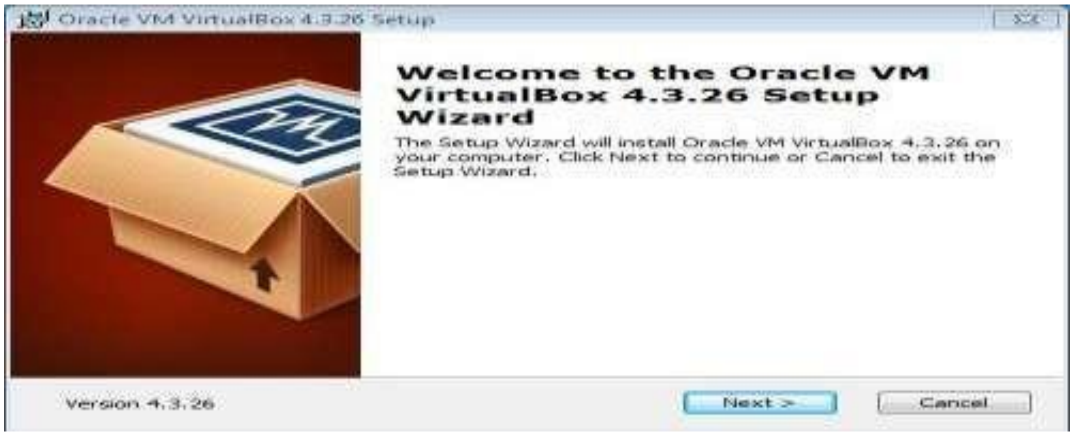
ALGORITHM:

1. Start
2. Download the latest version of Virtualbox and the Ubuntu ISO file from official sources.
3. Now, first install the Virtualbox application as specified below. After installing, create a Virtual Machine in Virtualbox.
4. Then, start the Virtual Machine and then select the downloaded Ubuntu ISO file. The Virtual Machine will restart and the Ubuntu installation screen will appear.
5. Follow all the necessary steps carefully leaving out all the default settings as is.
6. The OS installation process will start now. Wait patiently for about 10 – 15 minutes for Ubuntu to get installed.
7. The VM will reboot again and you'll be greeted by the Ubuntu desktop which indicates successful installation of Ubuntu in the VM.
8. Stop.

PROCEDURE:

Installing VirtualBox is just like installing any other application on your Windows OS, so you shouldn't have any problem with the installation.

To start the installation, double-click on the installer file you've downloaded

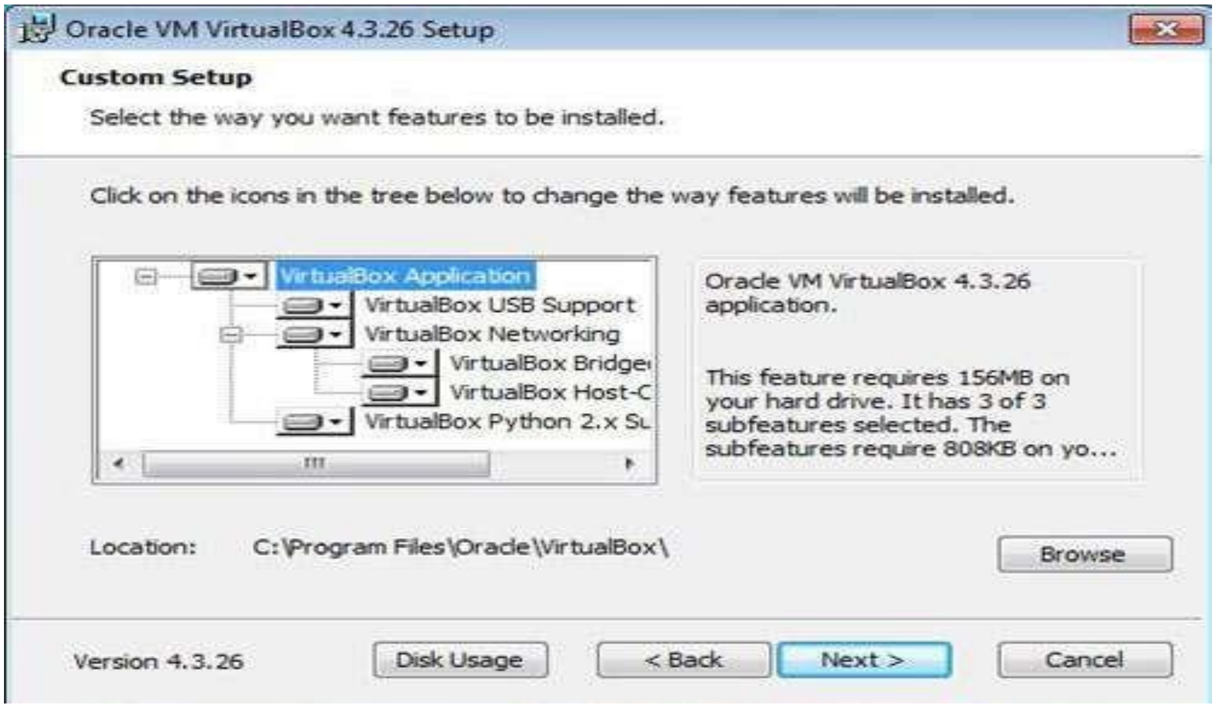

Oracle VM VirtualBox 4.3.26 Setup

Welcome to the Oracle VM VirtualBox 4.3.26 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 4.3.26 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 4.3.26

In the Welcome window, click **Next**. Then, select the installation folder and click **Next**:


Oracle VM VirtualBox 4.3.26 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

Oracle VM VirtualBox 4.3.26 application.

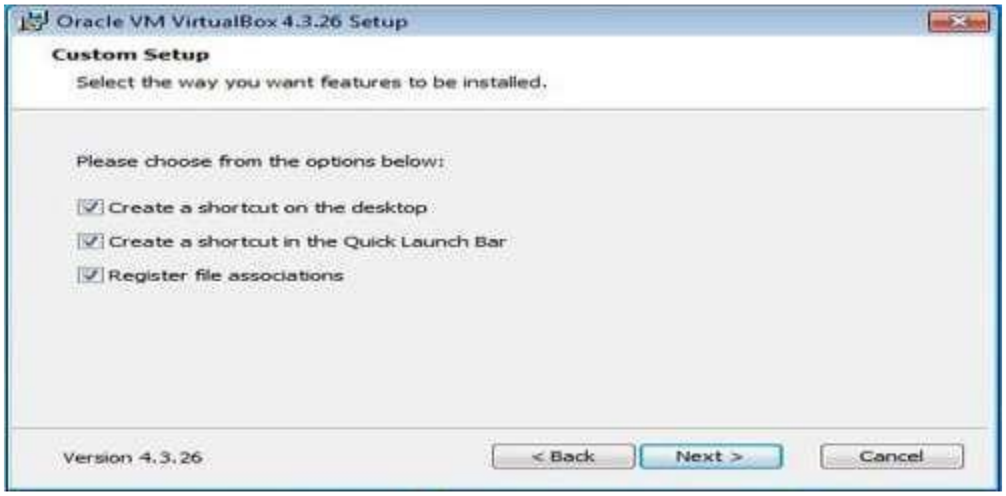
This feature requires 156MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 808KB on yo...

Location: C:\Program Files\Oracle\VirtualBox\

Version 4.3.26

Select the way you want features to be installed and then click **Next**:

Click **Yes** to install Oracle VirtualBox interfaces:


Oracle VM VirtualBox 4.3.26 Setup

Custom Setup

Select the way you want features to be installed.

Please choose from the options below:

- [✓] Create a shortcut on the desktop
- [✓] Create a shortcut in the Quick Launch Bar
- [✓] Register file associations

Version 4.3.26

After a minute the installation process should finish.

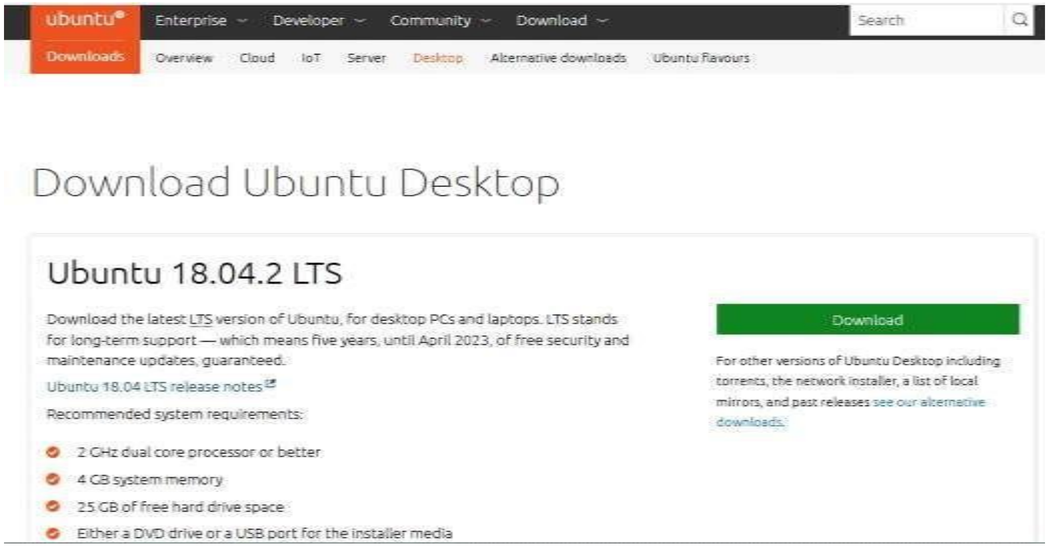
Install Ubuntu on Virtual Box as a VM:

System Configuration of the Host Machine

In order to install Ubuntu on VirtualBox, you should have a physical computer with at least 4 GB of RAM (Random Access Memory), a hard disk drive with at least 30 GB of free space (SSD is preferred due to its higher performance). Your CPU (Central Processor Unit) must support Intel VT-x or AMD-v hardware virtualization features which must also be enabled in UEFI/BIOS. This point is especially important if you are looking for how to install Ubuntu 64-bit on Virtual Box.

Downloading the Installation Image

You need to download the Ubuntu distribution for installing Ubuntu on Virtual Box. Go to the official Ubuntu website and download the necessary version of the Ubuntu installer. Let's download Ubuntu 18.04.2 LTS – this is the latest long term support (LTS) Ubuntu version available at this moment. You can find version numbers that are higher than 18.04.2, but they may not offer long term support yet. Five-year support is provided for Ubuntu LTS distributions (both Ubuntu Desktop and Ubuntu Server). Ubuntu LTS is more widely tested, enterprise-focused and compatible with new hardware. Click the green Download button and save the ISO file to the custom location. In our case, the file name is ubuntu-18.04.2-desktop-amd64.iso. Ubuntu 18 is provided only as 64-bit editions.



Download Ubuntu Desktop

Ubuntu 18.04.2 LTS

Download the latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years, until April 2023, of free security and maintenance updates, guaranteed.

Ubuntu 18.04 LTS release notes

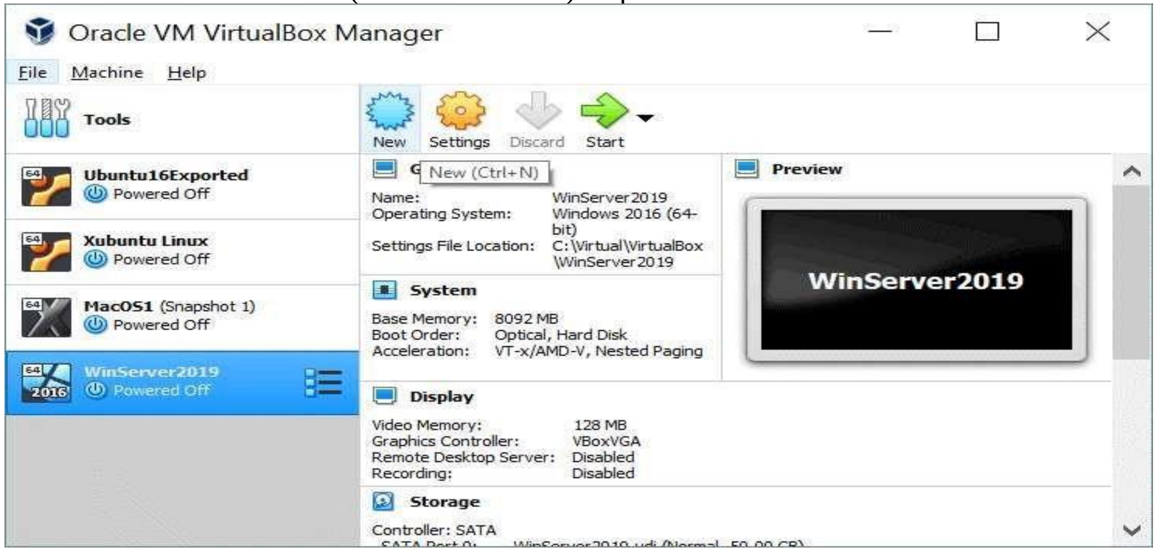
Recommended system requirements:

- 2 GHz dual core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors, and past releases see our alternative downloads.

Creating a New VM

In order to create a new virtual machine for installing Ubuntu on VirtualBox, open VirtualBox and click New (Machine > New) or press Ctrl+N.



Ubuntu16Exported
Powered Off

Xubuntu Linux
Powered Off

MacOS1 (Snapshot 1)
Powered Off

WinServer2019
Powered Off

Name: WinServer2019
Operating System: Windows 2016 (64-bit)
Settings File Location: C:\Virtual\VirtualBox\WinServer2019

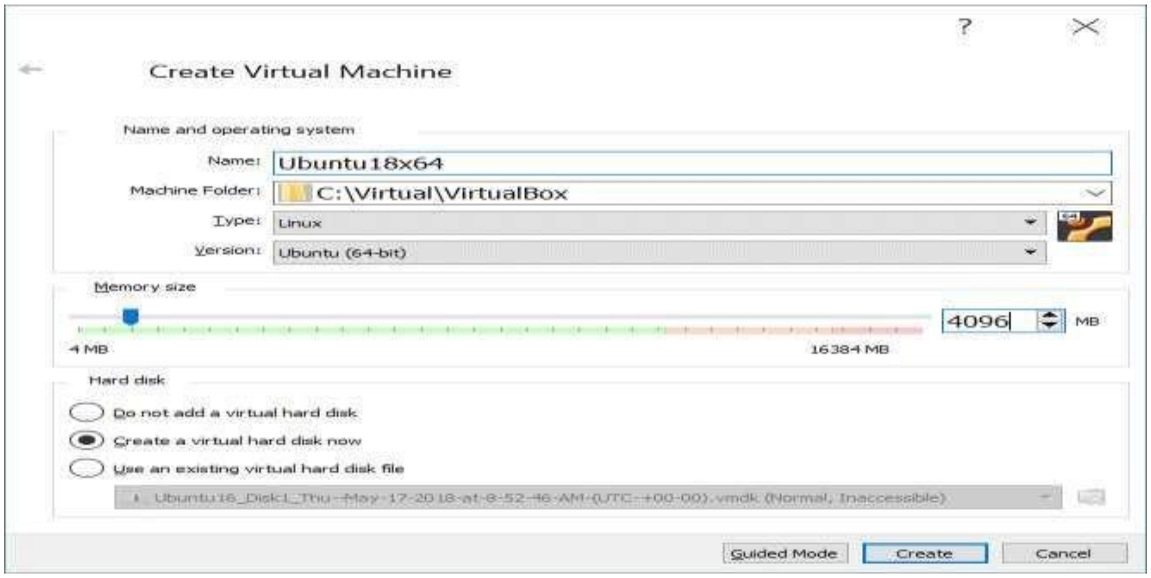
System
Base Memory: 8092 MB
Boot Order: Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

Display
Video Memory: 128 MB
Graphics Controller: VBoxVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: SATA

In the Create Virtual Machine screen, set the options for a new VM. In our example of installing Ubuntu on VirtualBox, the new VM options are the following:
Name: Ubuntu 18x64
Machine Folder: C:\Virtual\VirtualBox (try to use disk D, E or other non-system partitions if you have them).
Type: Linux

Version: Ubuntu (64-bit)
Memory size: Set at least 1 GB of RAM. As our physical machine used in this example has 16 GB of RAM, we can set 4 GB of RAM for a virtual machine to install Ubuntu on VirtualBox. You should leave enough memory for your host operating system to operate normally.
Select the Create a virtual hard disk now option. Hit Create to continue.



Name and operating system

Name: Ubuntu18x64

Machine Folder: C:\Virtual\VirtualBox

Type: Linux

Version: Ubuntu (64-bit)

Memory size

4 MB — 16384 MB — 4096 MB

Hard disk

( ) Do not add a virtual hard disk

(●) Create a virtual hard disk now

( ) Use an existing virtual hard disk file

Ubuntu16_Disk1_Thu-May-17-2018-at-8-52-46-AM-(UTC-+00-00).vmdk (Normal, Inaccessible)

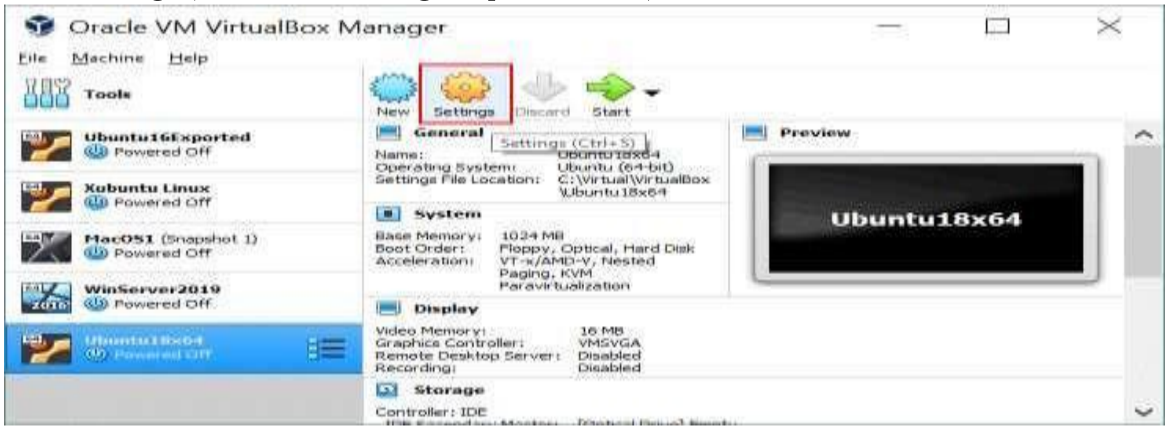
On the next Create Virtual Hard Disk screen, set the virtual disk file location, for example, C:\Virtual\VirtualBox\Ubuntu 18x64\Ubuntu 18x64.vdi The file size of the virtual disk: 20 GB or more.

Hard disk file type: VDI (VirtualBox Disk Image). Let's select the native VirtualBox virtual disk format. Storage on physical hard disk: Dynamically allocated. This option allows you to save space on your physical disk until the virtual disk grows to its maximum allocated size.

Hit Create to finish creating a new VM to install Ubuntu on VirtualBox.

VM Configuration

A new virtual machine to install Ubuntu on VirtualBox has now been created and its name is displayed in the list of VMs in the main VirtualBox window. You need to edit VM settings after VM creation. Select your new VM (Ubuntu18x64 in this case) and click Settings (Machine > Settings or press Ctrl+S).



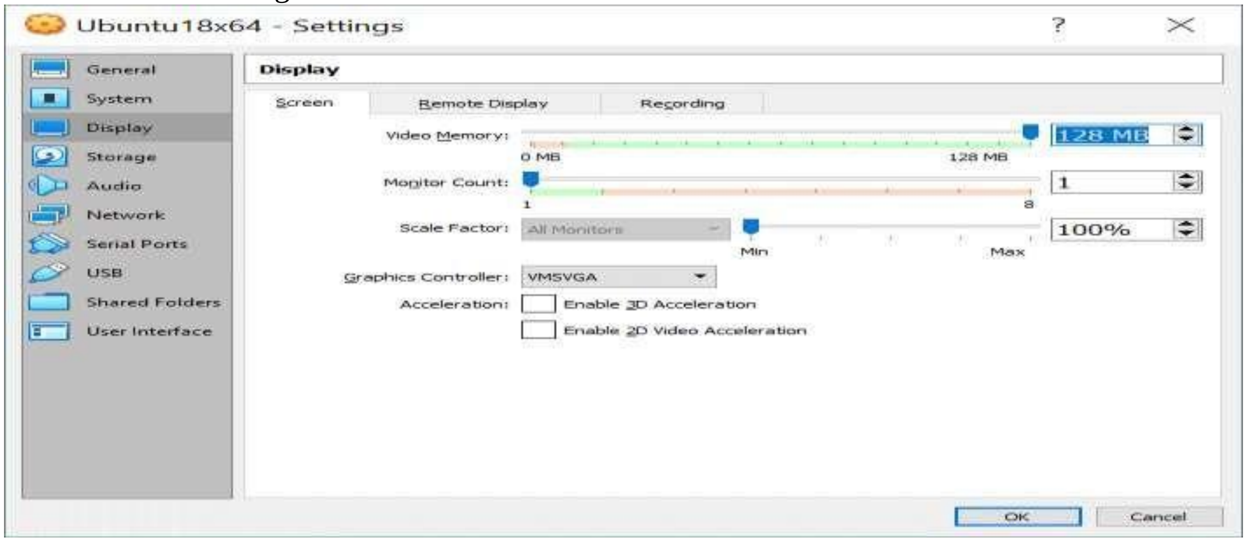
General
Name: Ubuntu18x64
Operating System: Ubuntu (64-bit)
Settings File Location: C:\Virtual\VirtualBox\Ubuntu18x64

System
Base Memory: 1024 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: IDE

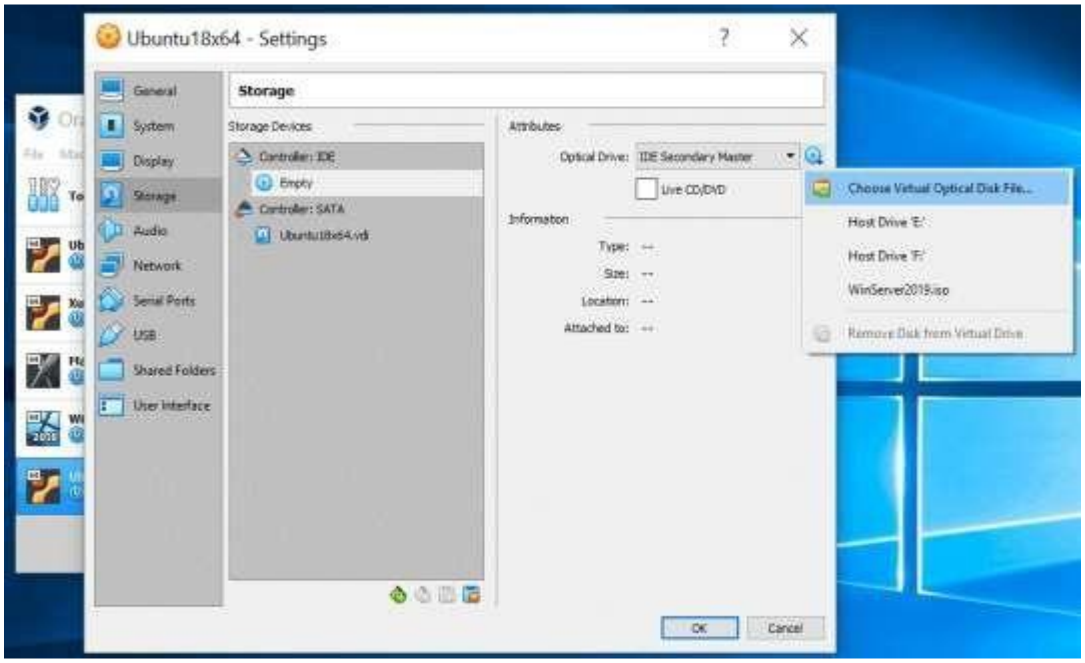
In the Settings window, go to the Display section and select the Screen tab. Set video memory to 128 MB. Otherwise the Ubuntu installer may hang on some installation steps, keyboard may not respond etc. You can enable 3D acceleration.
Hit OK to save settings.



Video Memory: 128 MB
Monitor Count: 1
Scale Factor: All Monitors 100%
Graphics Controller: VMSVGA
Acceleration: ☐ Enable 3D Acceleration
☐ Enable 2D Video Acceleration

Select the Boot Disk Image

You don't need to burn the ISO image onto a DVD disk as you would for installing an operating system on a physical machine. You can mount the ISO image to the virtual DVD drive of the virtual machine and boot a VM from this media. Let's insert the ubuntu 18.04.2- desktop-amd64.iso image that was downloaded from the official Ubuntu web site before, into a virtual DVD drive of the Ubuntu 18x64 VM. Open your VM settings and go to the Storage section. Select your virtual controller used for connecting a virtual DVD drive (by default a virtual DVD drive is empty). Click the Empty status and in the right pane near the IDE Secondary Master, click the disc icon. In the menu that appears, click Choose Virtual Optical Disk File and browse your Ubuntu installation ISO image file (ubuntu-18.04.2- desktop-amd64.iso).
Hit OK to save settings. Now your VM is ready to install Ubuntu on VirtualBox.



Storage

Storage Devices

Controller: IDE

Empty

Controller: SATA

Ubuntu18x64.vdi

Attributes

Optical Drive: IDE Secondary Master

Live CD/DVD

Information

Type: --

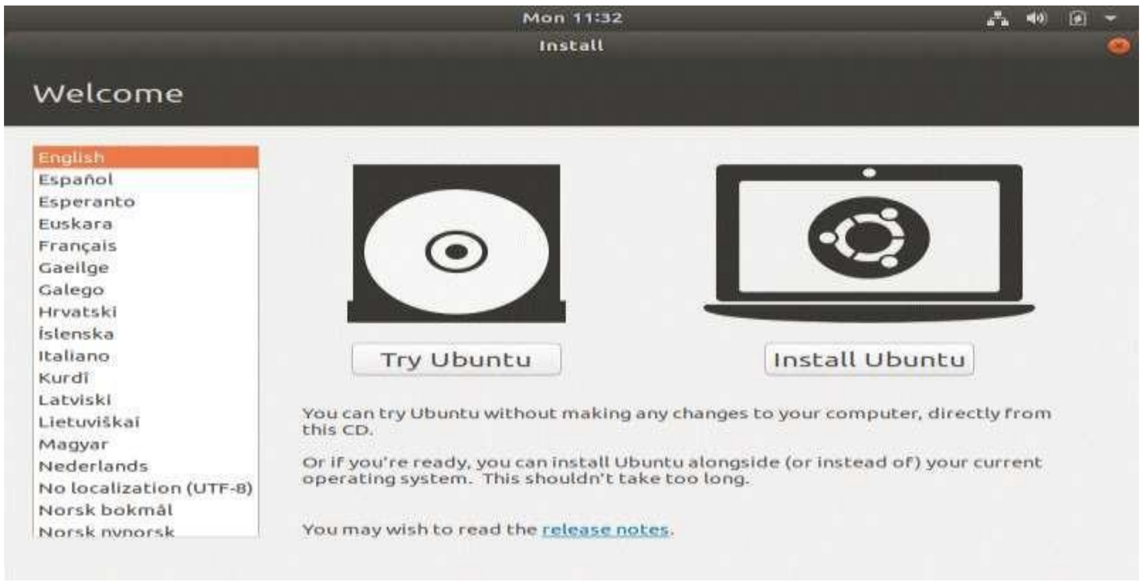
Size: --

Location: --

Attached to: --

Install Ubuntu on VirtualBox VMs

Once the new VM is prepared for installing Ubuntu on VirtualBox,start the VM (Machine > Start). The VM boots from the ISO Ubuntu installation image. The first screen that you can see after booting is the Welcome screen.
In the left pane select Language for displaying information in the installer interface. English is selected in the current example. Then click Install Ubuntu.



You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

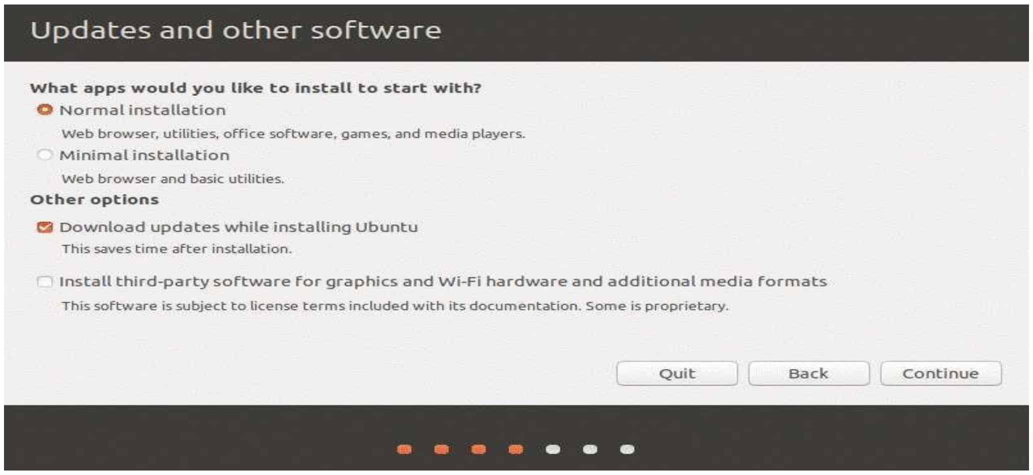
You may wish to read the release notes.

Keyboard layout. Choose your keyboard layout. Let's select English (US).

Updates and other software. There are a few options to choose from on this screen. Normal installation. A web browser, utilities, office applications and media players are

installed.
Minimal installation. Only the main components including a web browser and basic utilities are installed.
Let's select the normal installation

Download updates while installing Ubuntu. The Ubuntu team is always working towards making Linux better. That's why after downloading the installer, some updates may be already available. You can automatically download and install updates right during Ubuntu installation, letting you save time after OS installation. Let's select this option.
Install third-party software for graphics and Wi-Fi hardware and additional media formats. Tick this checkbox if you would like to install additional software, such as proprietary Wi-Fi drivers, video drivers, some TTF fonts etc.


Updates and other software

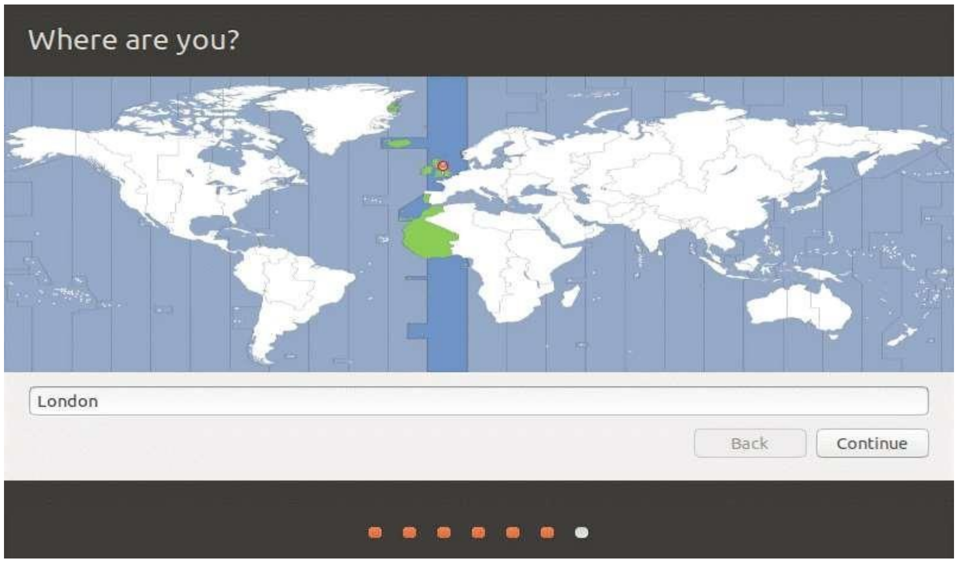
What apps would you like to install to start with?
● Normal installation
Web browser, utilities, office software, games, and media players.
○ Minimal installation
Web browser and basic utilities.
Other options
✓ Download updates while installing Ubuntu
This saves time after installation.
☐ Install third-party software for graphics and Wi-Fi hardware and additional media formats
This software is subject to license terms included with its documentation. Some is proprietary.

Installation type.
This screen contains options for preparing a disk for Ubuntu installation. Erase disk and install Ubuntu. This is the default option. All disk space will be automatically allocated to Ubuntu. If you select Erase disk and install Ubuntu on VirtualBox VMs, one big /dev/sda1 partition is created on /dev/sda. This /dev/sda1 partition with ext4 file system is mounted to the / directory (root directory), though a separate swap partition is not created. Attention: All data on the virtual disk will be erased—there is no reason to worry about it, however, because an empty virtual disk created previously is being used for installing Ubuntu on VirtualBox.

There are some additional options:

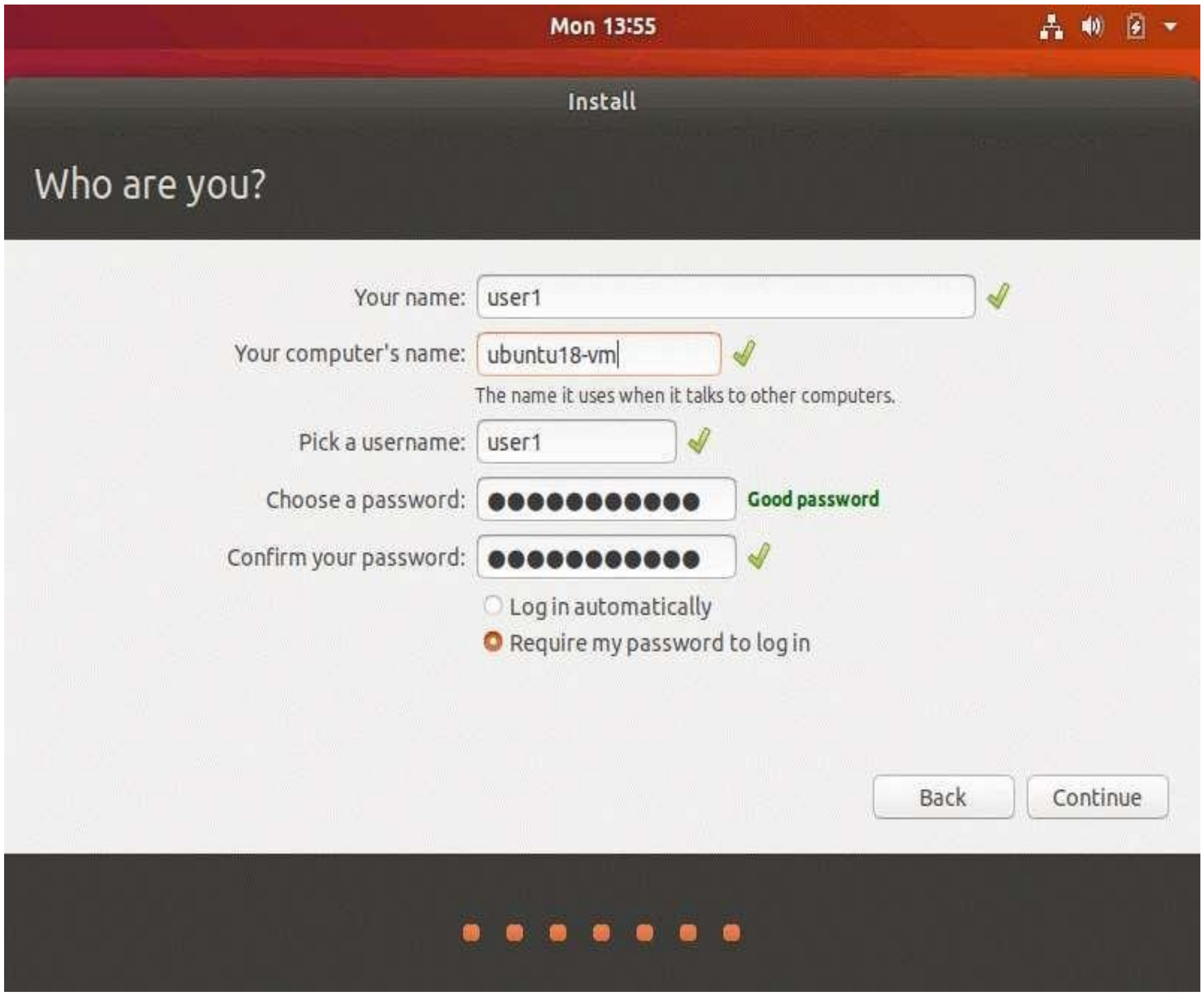
- Encrypt the new Ubuntu installation for security.
- Use LVM (Logical Volume Management) with the new Ubuntu installation. Something else. Use this option for manual creation of the partition table on your virtual disk which is used to install Ubuntu on VirtualBox.

Click Install Now when you are ready to continue. Then on the confirmation screen, hit Continue.
Where are you? Select your location to set the time zone and regional settings. The time for your selected region will be set automatically. Let's select London.



Enter your user name, computer's name, and set the password. Select Require my password to log in for a higher level of security. In our example, the username is user1 and the computer's name is ubuntu18-vm.

As you can see, useful tips are displayed on the screen during the installation process. When installation is complete, you will see a notification window. You have to restart your VM with Ubuntu on VirtualBox.


Mon 13:55

Install

Who are you?

Your name: user1

Your computer's name: ubuntu18-vm

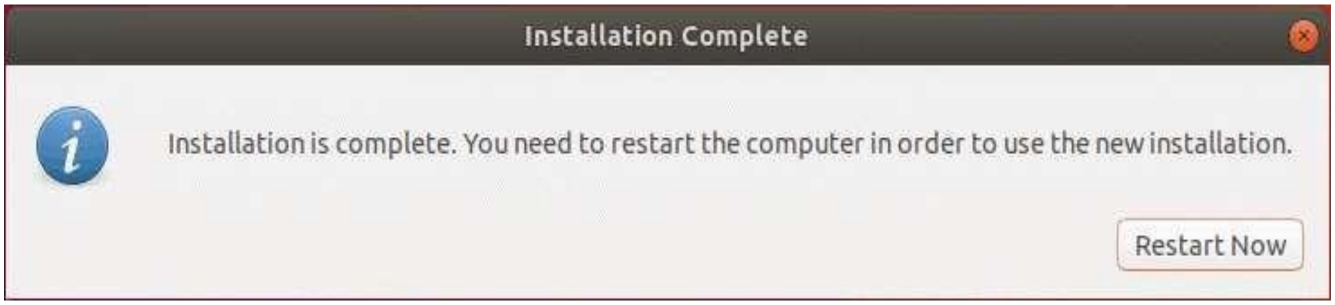
The name it uses when it talks to other computers.

Pick a username: user1

Choose a password: ●●●●●●●●●●●● Good password

Confirm your password: ●●●●●●●●●●●●

Log in automatically [✓ Require my password to log in]


Installation Complete

Installation is complete. You need to restart the computer in order to use the new installation.

Now you can eject the Ubuntu installation ISO disk from the virtual CD/DVD drive of the VM. This dialog box indicates that the Ubuntu OS has been installed successfully. Reboot your VM to experience the goodness of Ubuntu OS.

RESULT:

Thus the installation of Virtual machine and Ubuntu OS on top of Windows is completed successfully

Ex.No:4a	**INSTALL A ‘C’ COMPILER IN A VIRTUAL MACHINE AND EXECUTE PROGRAM**
Date: 22.08.22	

AIM:

To install a C compiler in an Ubuntu Virtual Machine and execute C programs in that Ubuntu based Virtual Machine.

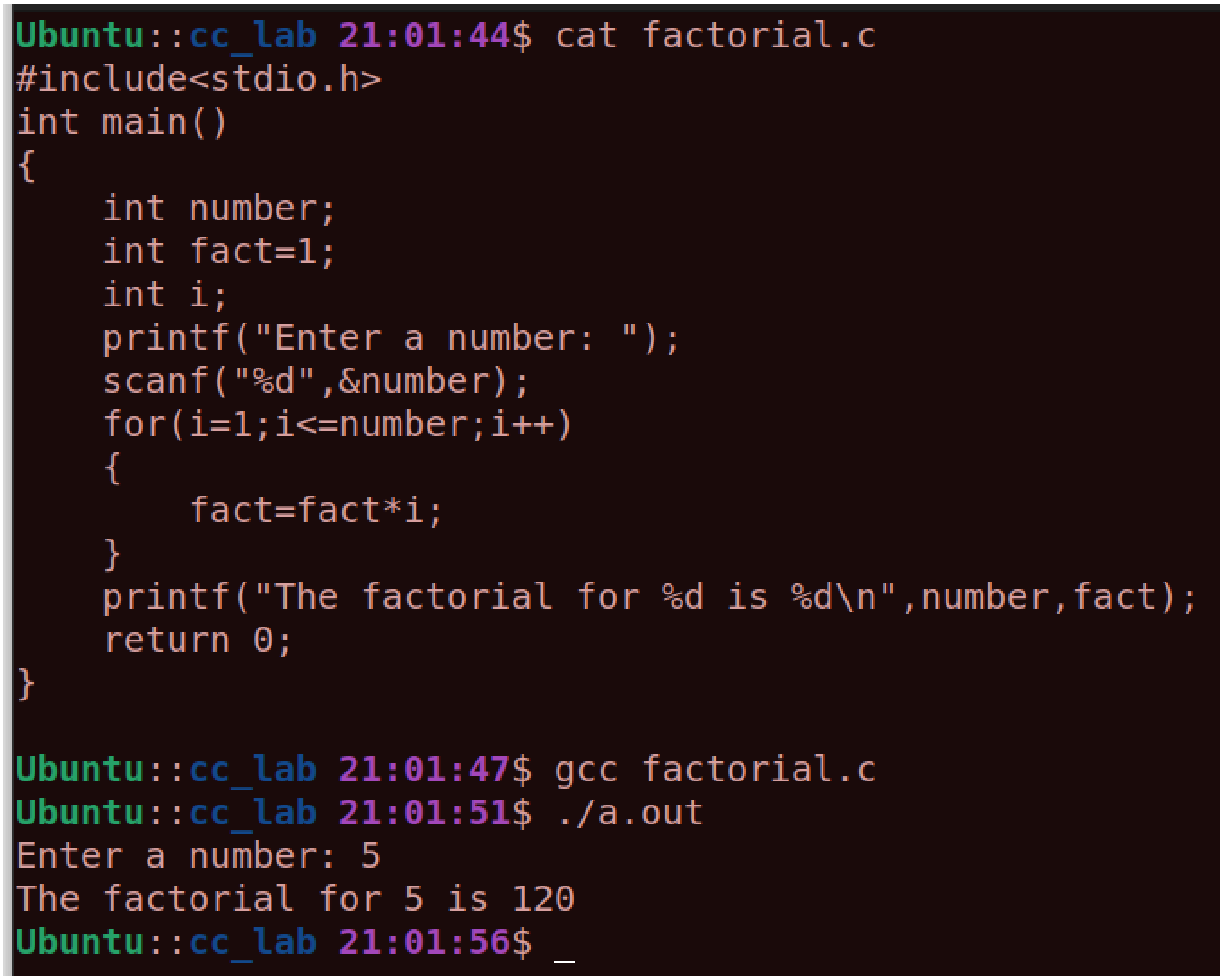
ALGORITHM:

1. Start
2. Open Terminal in the Ubuntu VM by selecting the Terminal Icon on the sidebar.
3. Now, terminal opens up. Type the necessary commands needed to install gcc on the Ubuntu VM.
4. Then, create a new file with ‘.c’ extension and type the program mentioned below
5. After that, execute the program by typing the necessary commands. Then run the required command for generating the output. The C program’s output will be displayed on the screen.
6. Stop

PROGRAM:

```
#include<stdio.h>
int main()
{
   int number;
   int fact=1;
   int i;
   printf("Enter a number: ");
   scanf("%d",&number);
   for(i=1;i<=number;i++)
   {
      fact=fact*i;
   }
   printf("The factorial for %d is %d\n",number,fact);
   return 0;
}
```

OUTPUT:



```
Ubuntu::cc_lab 21:01:44$ cat factorial.c
#include<stdio.h>
int main()
{
    int number;
    int fact=1;
    int i;
    printf("Enter a number: ");
    scanf("%d",&number);
    for(i=1;i<=number;i++)
    {
        fact=fact*i;
    }
    printf("The factorial for %d is %d\n",number,fact);
    return 0;
}

Ubuntu::cc_lab 21:01:47$ gcc factorial.c
Ubuntu::cc_lab 21:01:51$ ./a.out
Enter a number: 5
The factorial for 5 is 120
Ubuntu::cc_lab 21:01:56$ _
```

RESULT:

Thus the installation of a C compiler in an Ubuntu VM and executing C program in it was completed successfully.

Ex.No:4b	INSTALL A JAVA COMPILER IN A VIRTUAL MACHINE AND EXECUTE PROGRAM
Date: 22.08.22	

AIM:

To install a java compiler in an Ubuntu Virtual Machine and execute java programs in that Ubuntu based Virtual Machine.

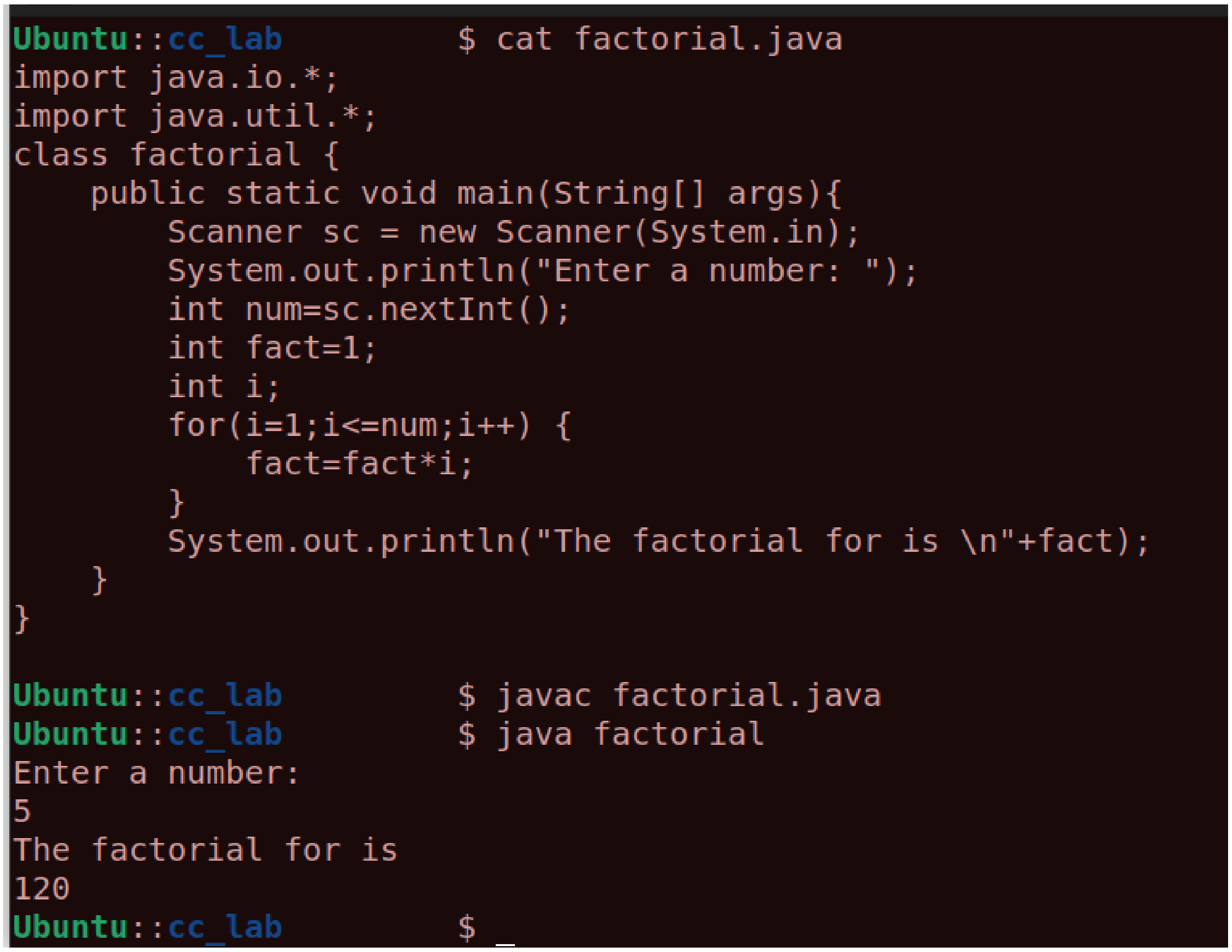
ALGORITHM:

1. Start
2. Open Terminal in the Ubuntu VM by selecting the Terminal Icon on the sidebar.
3. Now, terminal opens up. Type the necessary commands needed to install openjdk on the Ubuntu VM.
4. Then, create a new file with '.java' extension and type the program mentioned below
5. After that, execute the program by typing the necessary commands. Then run the required command for generating the output. The Java program's output will be displayed on the screen.
6. Stop

PROGRAM:

```
import java.io.*;
import java.util.*;
class factorial {
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter a number: ");
      int num=sc.nextInt();
      int fact=1;
      int i;
      for(i=1;i<=num;i++) {
         fact=fact*i;
      }
      System.out.println("The factorial for is \n"+fact);
   }
}
```

OUTPUT:



```
Ubuntu::cc_lab         $ cat factorial.java
import java.io.*;
import java.util.*;
class factorial {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number: ");
        int num=sc.nextInt();
        int fact=1;
        int i;
        for(i=1;i<=num;i++) {
            fact=fact*i;
        }
        System.out.println("The factorial for is \n"+fact);
    }
}

Ubuntu::cc_lab         $ javac factorial.java
Ubuntu::cc_lab         $ java factorial
Enter a number:
5
The factorial for is
120
Ubuntu::cc_lab         $ _
```

RESULT:

Thus the program to run java program in Ubuntu has been completed successfully and the output has been verified.

Ex.No:05	**INSTALLING GOOGLE APP ENGINE AND CREATING HELLO WORLD APPLICATION**
Date: 10.10.22	

AIM:

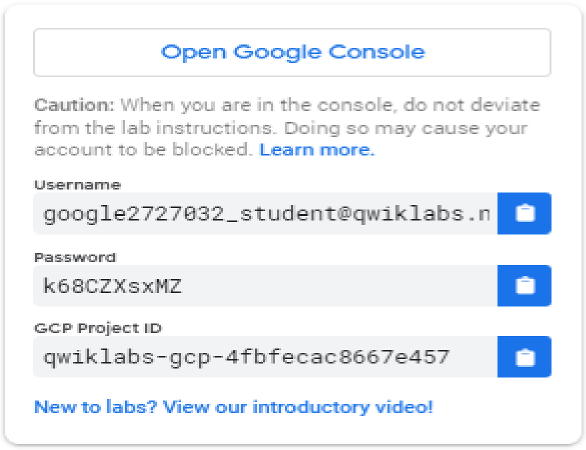
To install Google App Engine and create a Hello World application.

ALGORITHM:

1. Start
2. Create a Google Cloud account in the Google Cloud Platform console. After that, create a new project, and enable API access for that project.
3. Now, install and initialize the Google Cloud SDK(Software Development Kit).
4. Initialize your App Engine app with your project and choose its region by using this command: “gcloud app create --project=[YOUR_PROJECT_ID]”.
5. Run the following command to install the gcloud component that includes the App Engine extension for Python 3: “gcloud components install app-engine-python”.
6. Install the latest version of Python and type the below specified code in a file named ‘main.py’.
7. Create an isolated Python environment by using these commands: “python -m venvenv” and “.\env\Scripts\activate”.
8. Navigate to your project directory and install dependencies by: “pip install -r requirements.txt”
9. After that, Run the application by using this command: “python main.py” and enter the following address in your web browser - http://localhost:8080
10. The Hello World message from the sample app displays on the page. In your terminal window, press Ctrl+C to exit the web server.
11. Stop

PROCEDURE:

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

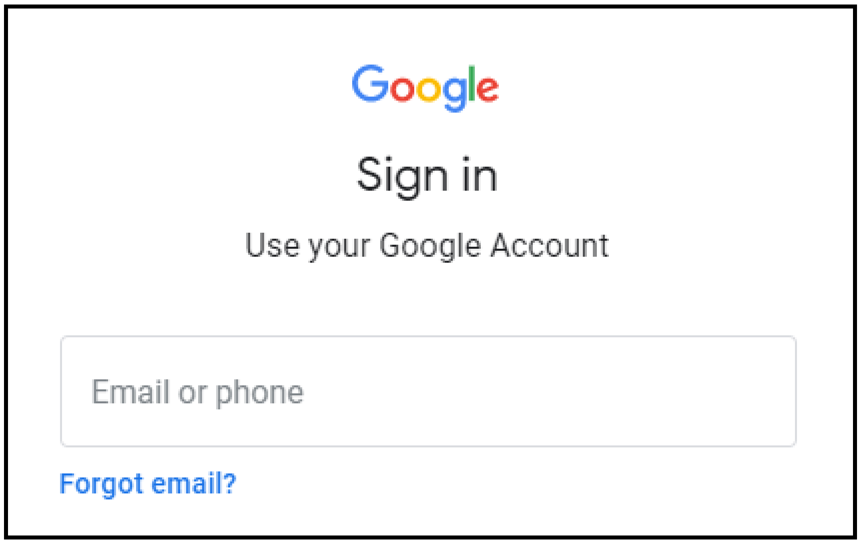
Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

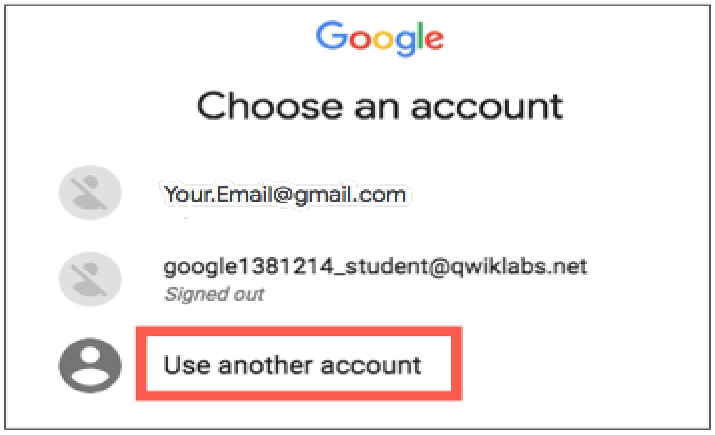
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

New to labs? View our introductory video!

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.
If you see the **Choose an account** page, click **Use Another Account**



.

3. In the **Sign in** page, paste the username that you copied from the left panel.

Then copy and paste the password.

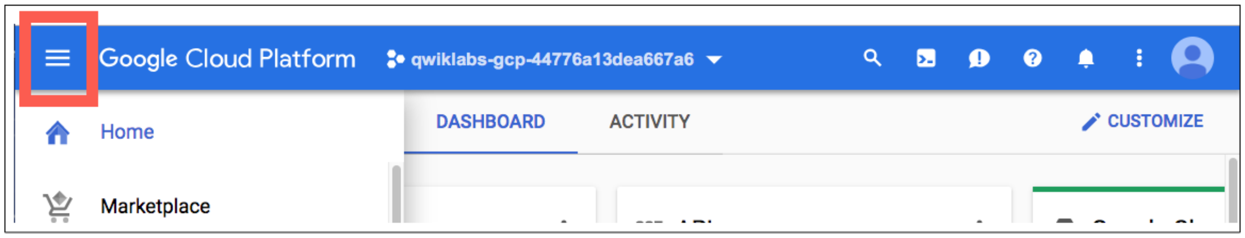
Important: You must use the credentials from the left panel. Do not use your Google Cloud Training credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

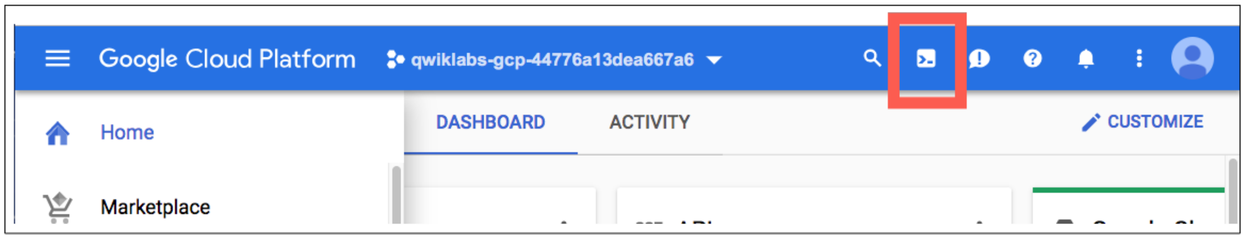
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top left.



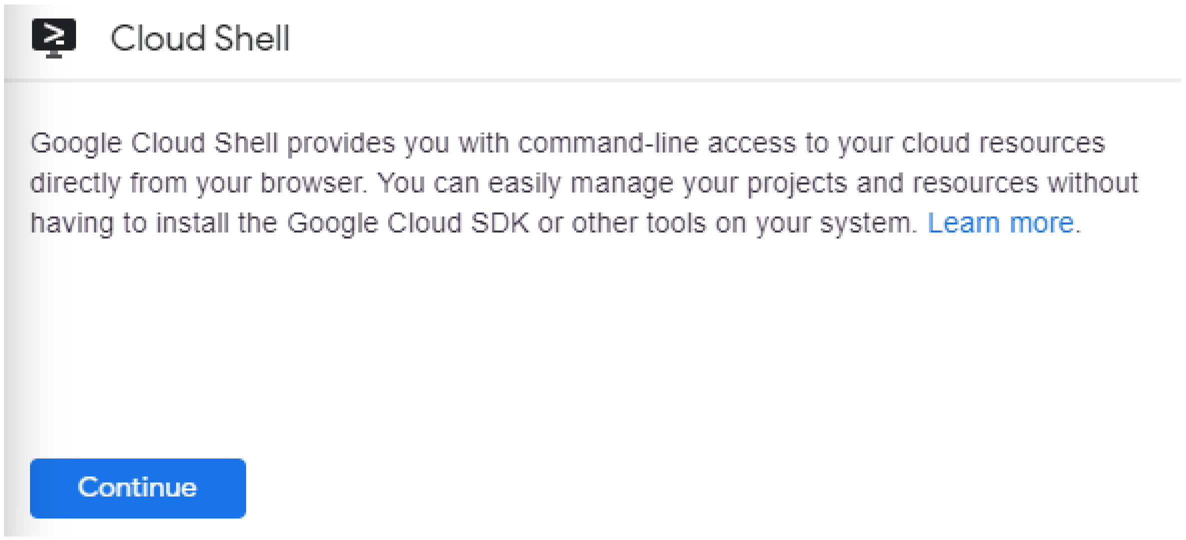
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.

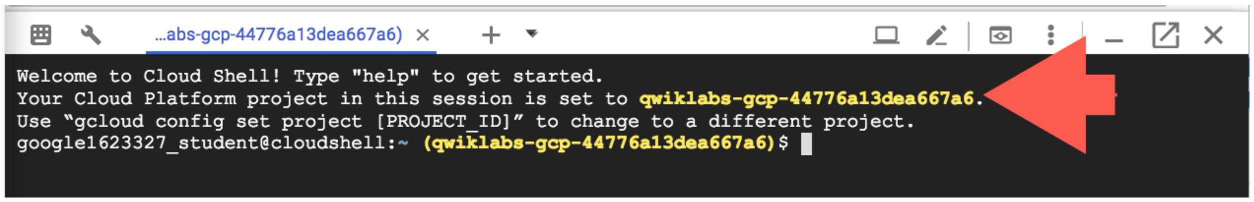


Click **Continue**.



Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

```
 [core]
project = <project_ID>
```

(Example output)

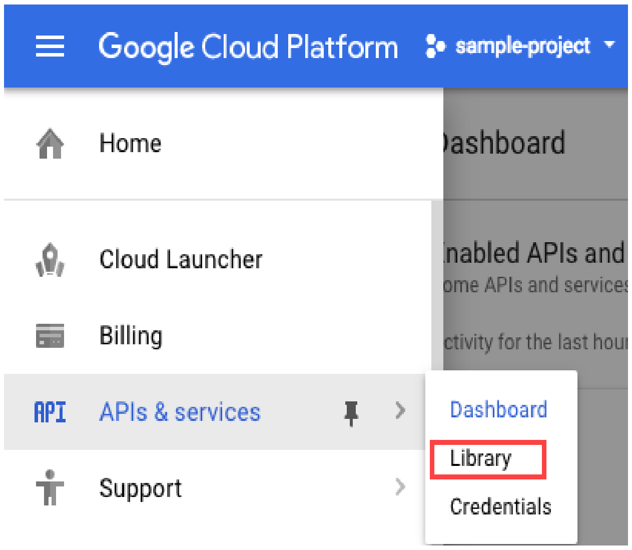
```
[core]
```

project = qwiklabs-gcp-44776a13dea667a6

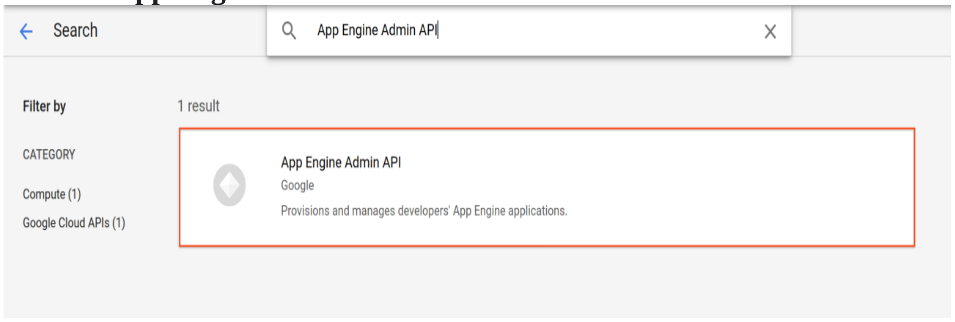
Enable Google App Engine Admin API

The App Engine Admin API enables developers to provision and manage their App Engine Applications.

- In the left menu click **APIs & Services** > **Library**.

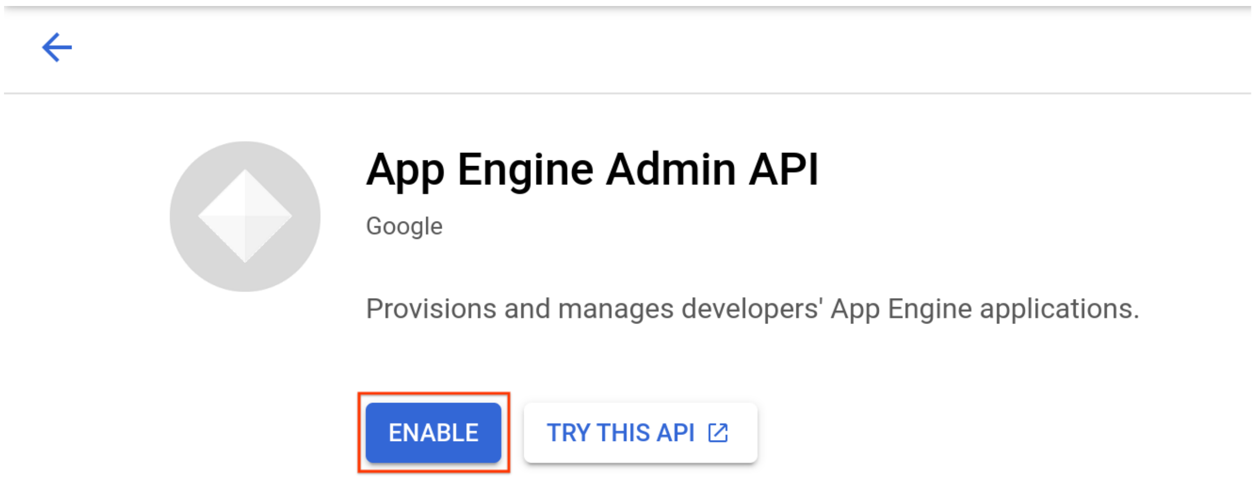


- Type "App Engine Admin API" in search box.
- Click **App Engine Admin API**.



Provisions and manages developers' App Engine applications.

- Click **Enable**. If there is no prompt to enable the API, then it is already enabled and no action is needed.



App Engine Admin API

Google

Provisions and manages developers' App Engine applications.

Download the Hello World app

There is a simple Hello World app for Python you can use to quickly get a feel for deploying an app to Google Cloud. Follow these steps to download Hello World to your Google Cloud instance.

1. Enter the following command to copy the Hello World sample app repository to your Google Cloud instance

2. gsutil -m cp -r gs://spls/gsp067/python-docs-samples .

3. Copied!

   content_copy

4. Go to the directory that contains the sample code:

```
cd python-docs-samples/appengine/standard_python3/hello_world
```
Copied!

content_copy

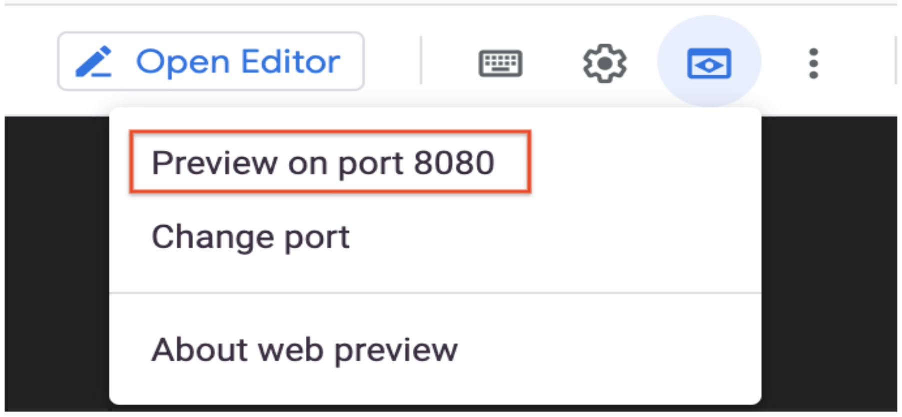
Test the application

Test the application using the Google Cloud development server (dev_appserver.py), which is included with the preinstalled App Engine SDK.

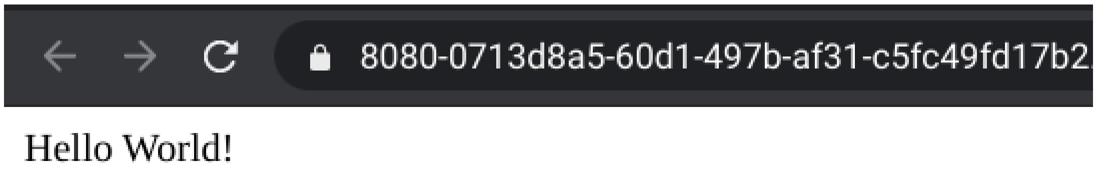
1. From within your helloworld directory where the app's app.yaml configuration file is located, start the Google Cloud development server with the following command: dev_appserver.py app.yaml
Copied!

content_copy

The development server is now running and listening for requests on port 8080. 2. View the results by clicking the **Web preview** > **Preview on port 8080**.



You'll see this in a new browser window:



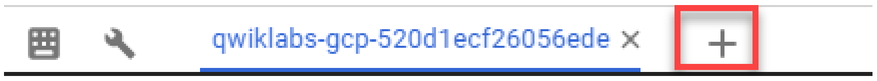
Hello World!

Make a change

You can leave the development server running while you develop your application. The development server watches for changes in your source files and reloads them if necessary.

Let's try it. Leave the development server running. We'll open another command line window, then edit main.py to change "Hello World!" to "Hello, Cruel World!".

1. Click the (+) next to your Cloud Shell tab to open a new command line session.



2. Enter this command to go to the directory that contains the sample code.
3. cd python-docs-samples/appengine/standard_python3/hello_world
4. Enter the following to open main.py in nano to edit the content.

```
nano main.py
```

5. Change "Hello World!" to "Hello, Cruel World!". Exit and save the file.

6. Reload the Hello World! Browser or click the **Web Preview** > **Preview on port 8080** to see the results.

Deploy your app

To deploy your app to App Engine, run the following command from within the root directory of your application where the app.yaml file is located:

```
gcloud app deploy
```

Copied!

content_copy

You will be prompted to enter where your App engine will be located.

```
Please choose the region where you want your App Engine application
located:
[1] asia-east2
[2] asia-northeast1
[3] asia-northeast2
[4] asia-northeast3
[5] asia-south1
[6] asia-southeast2
[7] australia-southeast1
[8] europe-west
[9] europe-west2
[10] europe-west3
[11] europe-west6
[12] northamerica-northeast1
[13] southamerica-east1
[14] us-central
[15] us-east1
```

```
[16] us-east4
[17] us-west2
[18] us-west3
[19] us-west4
[20] cancel
Please enter your numeric choice:
```

Copied!

content_copy

Enter the number that represents your region. The App Engine application will

then be created. Example output:

```
Creating App Engine application in project [qwiklabs-gcp-233dca09c0ab577b] and
region [asia-south1]....done. Services to deploy:
descriptor: [/home/gcpstaging8134_student/python-docs-
samples/appengine/standard/hello_world/app.yaml] source:
[/home/gcpstaging8134_student/python-docs-
samples/appengine/standard/hello_world] target project: [qwiklabs-gcp-
233dca09c0ab577b]
target service: [default]
target version: [20171117t072143]
target url: [https://qwiklabs-gcp-233dca09c0ab577b.appspot.com]
Do you want to continue (Y/n)?
```

Copied!

content_copy

Enter **Y** when prompted to confirm the details and begin the deployment of service.

Example output:

```
Beginning deployment of service [default]...
Some files were skipped. Pass `--verbosity=info` to see which ones.
You may also view the gcloud log file, found at
[/tmp/tmp.dYC7xGu3oZ/logs/2017.11.17/07.18.27.372768.log].
 File  upload done.
Updating service [default]...done.
      Waiting for operation [apps/qwiklabs-gcp-
233dca09c0ab577b/operations/2e88ab76-33dc-4aed-93c4- fdd944a95ccf] to
complete...done.
Updating service [default]...done.
Deployed service [default] to [https://qwiklabs-gcp-233dca09c0ab577b.appspot.com]
You can stream logs from the command line by running:
 $ gcloud app logs tail -s default
To view your application in the web browser run:
 $ gcloud app browse
```

Copied!

content_copy

Note: If you receive an error as "Unable to retrieve P4SA" while deploying the app, then re-run the above command.

View your application

To launch your browser enter the following command, then click on the link it provides.

```
gcloud app browse
```

Copied!

content_copy

Example output, note that your link will be different:

Did not detect your browser. Go to this link to view your app:
https://qwiklabs-gcp-233dca09c0ab577b.appspot.com
Copied!

Your application is deployed and you can read the short message in

your browser. Click **Check my progress** to verify the objective.

PROGRAM:

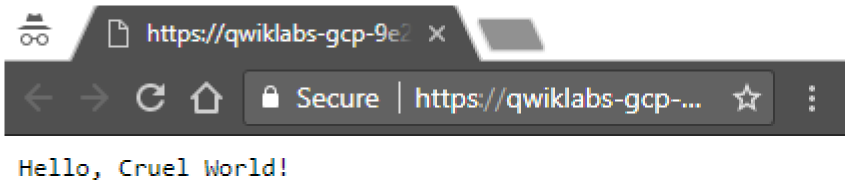
main.py:

```
    from flask import Flask
    app = Flask(__name__)
    @app.route('/')


    def hello():
 return 'Hello World!'

    if __name__ == '__main__':
 app.run(host='127.0.0.1', port=8080, debug=True)
```

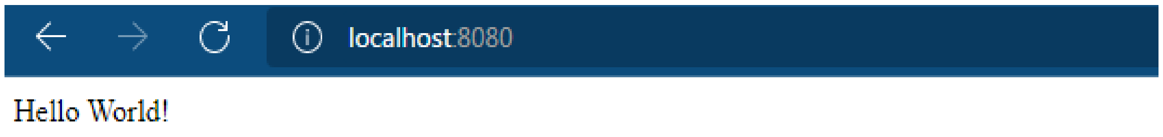
requirements.txt:

```
    Flask==2.0.1
```

OUTPUT:



```
Hello, Cruel World!
```



```
Hello World!
```

RESULT:

Thus the program to install Google App Engine and create Hello world Application has been completed successfully.

Ex.No: 6a	**LAUNCHING WEB APPLICATIONS USING GAE LAUNCHER**
Date: 10.10.22	

AIM:

To host the website using Google Application Engine Launcher using Cloud SDK.

ALGORITHM:

1. Start
2. Install the necessary Google Cloud SDK packages in your system.
3. Now, create a new project in the Google Cloud Platform after creating an account in GCP.
4. Create a new python program. In this case Flask framework is used to create the Python Web Application program.
5. After that, type the necessary commands to run the project in GAE Environment.
6. The output will appear on the screen if all the steps are performed correctly.
7. Stop.

PROCEDURE:

STEP 1: Creating the Google Cloud Platform Project

1. Go to the App Engine dashboard on the Google Cloud Platform Console and press the *Create* button.

2. If you've not created a project before, you'll need to select whether you want to receive email updates or not, agree to the Terms of Service, and then you should be able to continue.

3. Enter a name for the project, edit your project ID and note it down.

4. Click the *Create* button to create your project.

STEP 2: **Creating an Application**

Each Cloud Platform project can contain one App Engine application. Let's prepare an app for our project.

1. We'll need a sample application to publish. If you've not got one to use, create one by following these steps.

2. First create a new directory. Inside of it create two files: one called "main.py" and another one called requirements.txt

3. Now, type the code specified below in each of these files.

STEP 3: **Publishing the Application**

Now that we've got our project made and sample app files collected together,

let's publish our app.

1. Open Google Cloud Shell.
2. Drag and drop the sample-app folder into the left pane of the code editor.
3. Run the following in the command line to select yourproject (In this case the project's name is "YOUR_PROJECT_NAME").

   gcloud config set project *YOUR_PROJECT_NAME*

4. Then run the following command to go to your app's directory

   cd *sample-app*

5. You are now ready to deploy your application, i.e. upload your app to App Engine

   gcloud app deploy

6. Enter a number to choose the region where you want your applicationlocated.
7. Enter Y to confirm.
8. Now navigate your browser to *your-project-id*.appspot.com to see your website online. For example, for the project ID *YOUR_PROJECT_NAME*, go to *YOUR_PROJECT_NAME*.appspot.com.

PROGRAM:

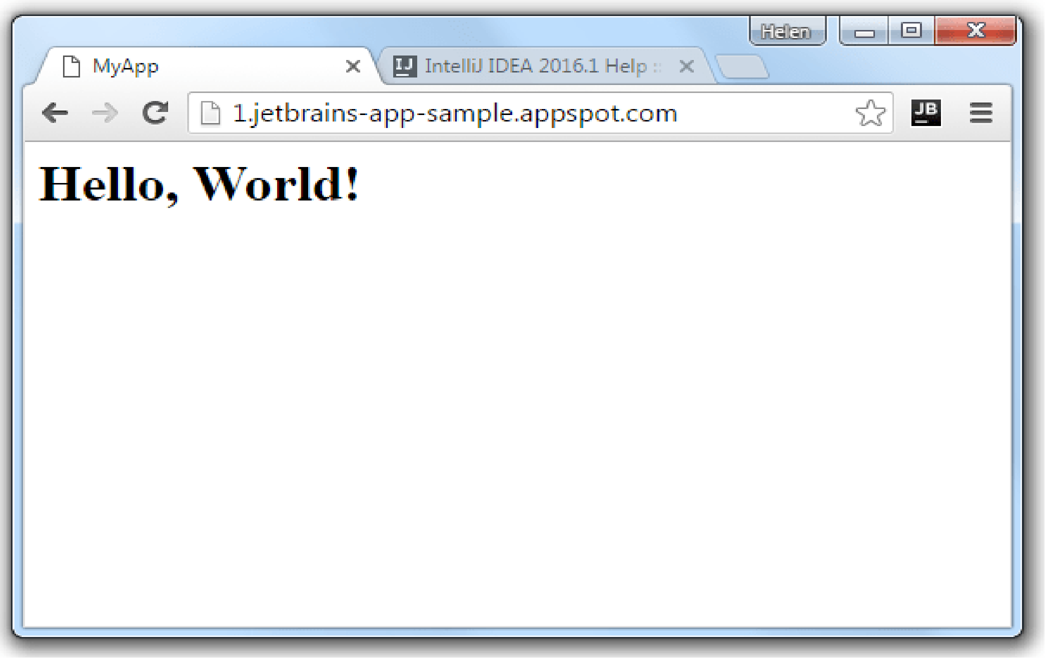
main.py:

```
    from flask import Flask
app = Flask(__name__)
@app.route('/')


def hello():
 return 'Hello, World!'
if __name__ == '__main__':
 app.run(host='127.0.0.1', port=8080, debug=True)
```

requirements.txt:

```
Flask==2.0.1
```

OUTPUT:



Hello, World!

RESULT:

Thus the program to launch web applications using GAE Launcher was completed successfully.

Ex.No:6b	DEPLOYING THE WEBSITE USING GAE IN LOCAL MACHINE
Date: 15.10.22	

AIM:

To simulate a cloud scenario with one datacenter using Cloudsim in Netbeans and to learn about the working of Cloudsim.

ALGORITHM:

1. Start
2. Install the necessary Google Cloud SDK packages in your system.
3. Create the project
4. Deploy the application using GAE in localhost
5. Test the application by opening the url local localhost:8080/
6. Stop

PROCEDURE:

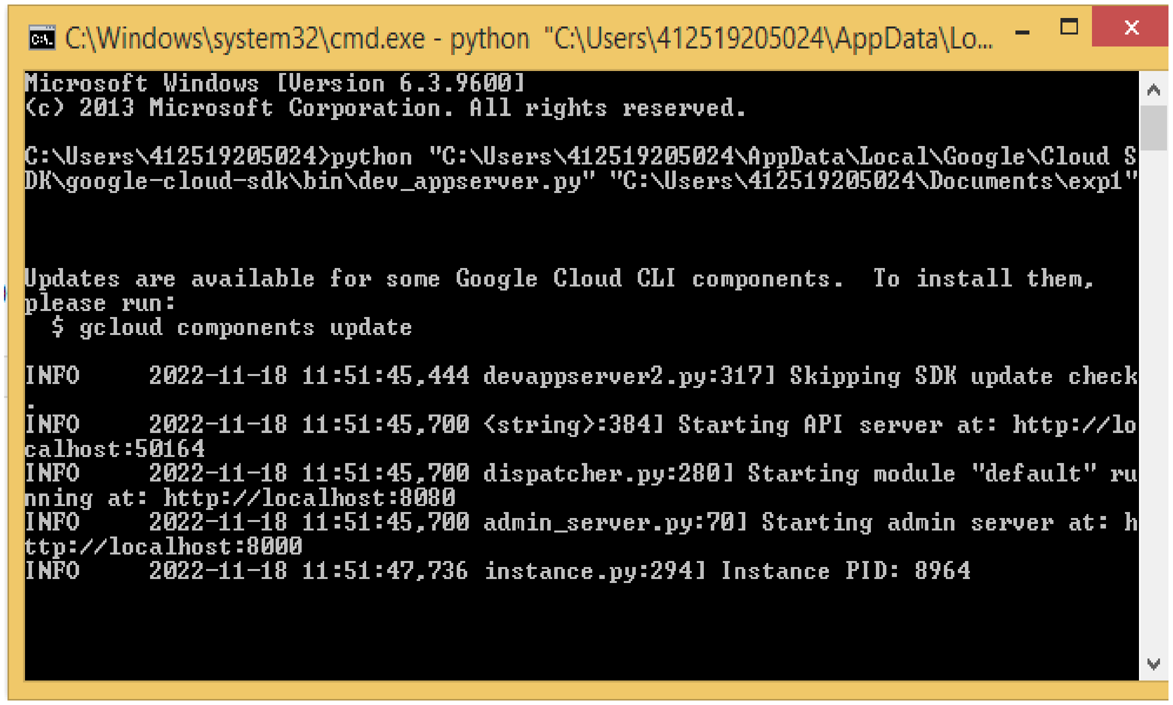
1. Download the google cloud installer and run the installer
   https://cloud.google.com/sdk/docs/install
2. Download the python version 3.7 from
   https://www.python.org/downloads/
3. Run the google cloud CLI in command prompt by using the command
   python " C:\Users\it\AppData\Local\Google\Cloud SDK\google-cloud-sdk\bin\dev_server" "our_project_location"

PROGRAM:

```
print("""
<html>
        <head>
                <title>Simple interest calculator</title>
                <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
        </head>
        <body>
                <div class="container">
                        <div class="row align-items-start m-2">
                                <div class="col">
                                        <h1>Simple interest calculator</h1>
                                </div>
                </div>
                <br/>
                <div class="row align-items-start m-2">
                        <div class="col-2">
                                <label> Principal amount </label>
                        </div>
                        <div class="col">
                                <input id = "principal" type = "number" />
                        </div>
                </div>

                <div class="row align-items-start m-2">
                        <div class="col-2">
                                <label> Rate of interest (p.a) </label>
                        </div>
                        <div class="col">
                                <input id = "interest" type = "number" />
                        </div>
                </div>

                <div class="row align-items-start m-2">
                        <div class="col-2">
                                <label> Number of years </label>
                        </div>
                        <div class="col">
                                <input id = "years" type = "number" />
                        </div>
                </div>
                <br/>
                <div class="row align-items-start m-2">
                        <div class="col-2"></div>
                        <div class="col">
                                <input type = "submit" value="Calculate S.I"
onclick="calculateSI(event)" />
                        </div>
                </div>
                <div class="row align-items-start m-2">
                        <div class="col">
                                <div class = "fs-2" id = "result"> </div>
                        </div>
                </div>

        </div>
        <script>
                function calculateSI(event){
                        console.log(event);
```

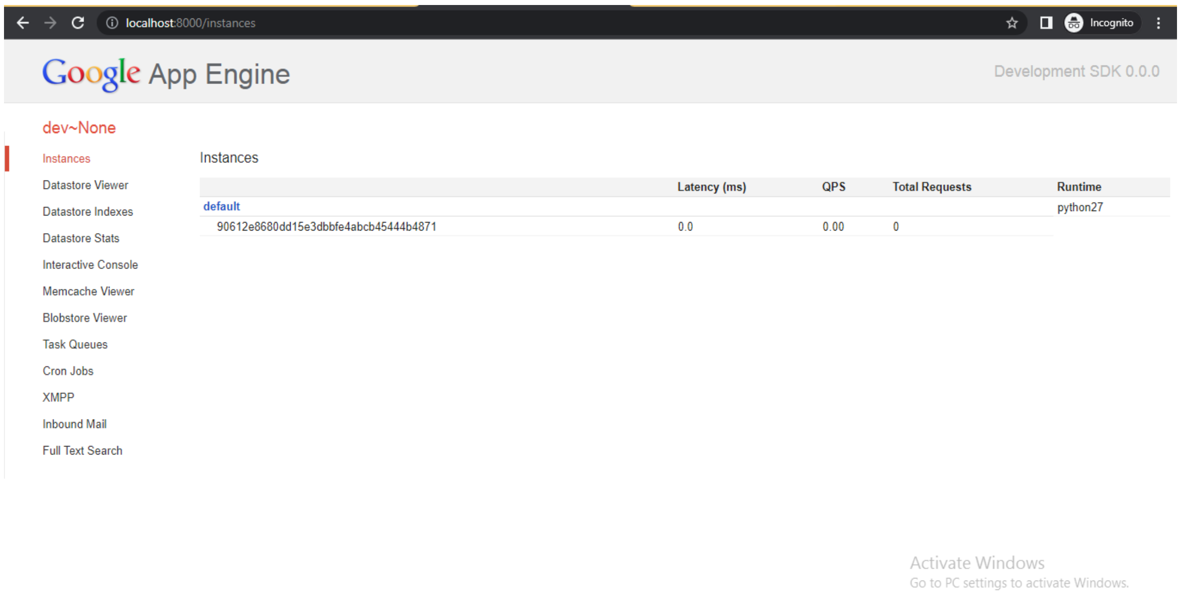
```
                                        event.preventDefault();
                                        var p = document.getElementById("principal").value;
                                        var n = document.getElementById("years").value;
                                        var r = document.getElementById("interest").value;
                                        var interest = p*n*r/100.0;
                                        console.log(interest);
                                        document.getElementById("result").innerHTML = `<strong>The
simple interest for &#8377; ${p} for ${n} years at ${r} % interest is &#8377; ${interest}
.</strong>`;
                              }
                    </script>
                    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
          </body>
</html>

""")
```

OUTPUT:



C:\Windows\system32\cmd.exe - python "C:\Users\412519205024\AppData\Lo...

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\412519205024>python "C:\Users\412519205024\AppData\Local\Google\Cloud S
DK\google-cloud-sdk\bin\dev_appserver.py" "C:\Users\412519205024\Documents\exp1"



Updates are available for some Google Cloud CLI components.  To install them,
please run:
  $ gcloud components update

INFO     2022-11-18 11:51:45,444 devappserver2.py:317] Skipping SDK update check
.
INFO     2022-11-18 11:51:45,700 <string>:384] Starting API server at: http://lo
calhost:50164
INFO     2022-11-18 11:51:45,700 dispatcher.py:280] Starting module "default" ru
nning at: http://localhost:8080
INFO     2022-11-18 11:51:45,700 admin_server.py:70] Starting admin server at: h
ttp://localhost:8000
INFO     2022-11-18 11:51:47,736 instance.py:294] Instance PID: 8964
```

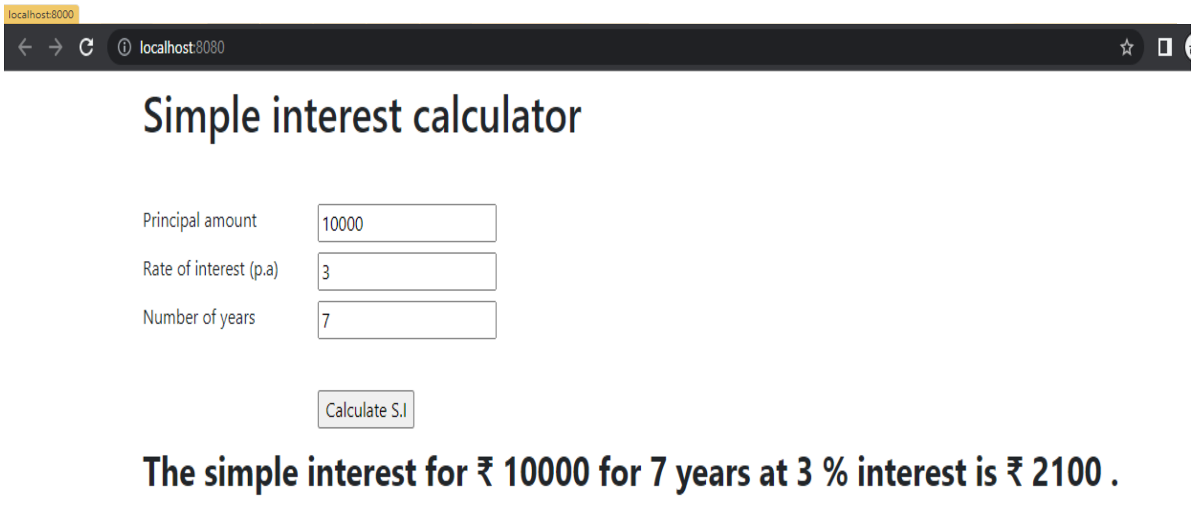

localhost:8000/instances

Google App Engine — Development SDK 0.0.0

dev~None

Instances

	Latency (ms)	QPS	Total Requests	Runtime
default				python27
90612e8680dd15e3dbbfe4abcb45444b4871	0.0	0.00	0	


localhost:8080

Simple interest calculator

Principal amount	10000
Rate of interest (p.a)	3
Number of years	7

The simple interest for ₹ 10000 for 7 years at 3 % interest is ₹ 2100 .

RESULT:

Thus the program to launch web applications using GAE Launcher was completed successfully.

Ex.No:7a	SIMULATION OF A CLOUD SCENARIO WITH ONE DATACENTER USING CLOUDSIM
Date: 12.09.22	

AIM:

To simulate a cloud scenario with one datacenter using Cloudsim in Netbeans and to learn about the working of Cloudsim.

ALGORITHM:

1. Start
2. Initialize a broker for a cloudlet along with its entities.
3. Assign the necessary cloud resources needed and start a Virtual Machine using that cloud resource.
4. Now, assign the already created cloudlet to the newly created Virtual Machine.
5. After assigning, the broker receives the necessary cloudlet and starts executing.
6. All the Virtual Machines are destroyed after the necessary cloudlets get executed.
7. Now, notify all cloudsim entities for shutting down.
8. Finally the datacenter and broker are shut down as well.
9. Print the obtained output from clousim simulation to the user.
10. Stop

PROGRAM:

```
package org.cloudbus.cloudsim.examples;

/*
 * Title:        CloudSim Toolkit
 * Description:  CloudSim (Cloud Simulation) Toolkit for Modeling and Simulation
 *               of Clouds
 * Licence:      GPL - http://www.gnu.org/copyleft/gpl.html
 *
 * Copyright (c) 2009, The University of Melbourne, Australia
 */

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.provisioners.*;
```

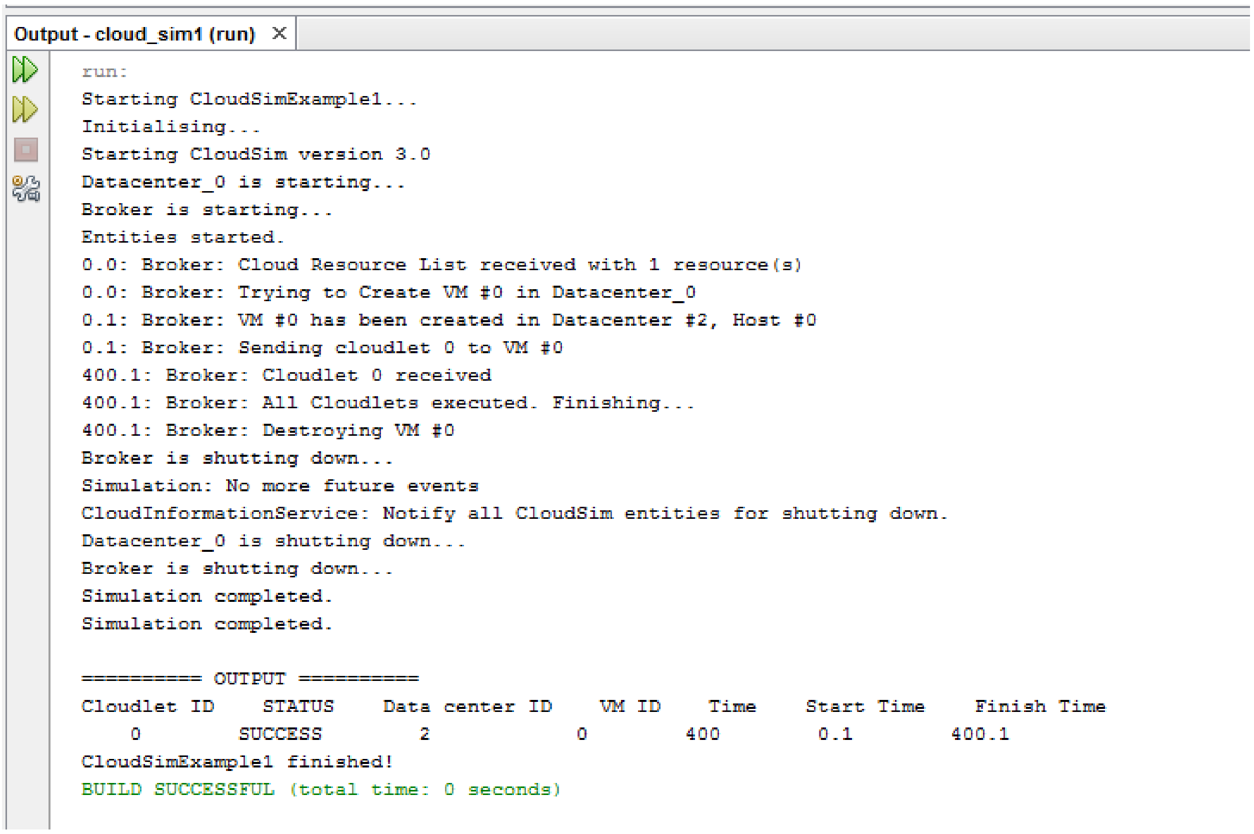
```
/**
 * A simple example showing how to create a datacenter with one host and run one
 * cloudlet on it.
 */
public class CloudSimExample1 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example.
	 *
	 * @param args the args
	 */
	@SuppressWarnings("unused")
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample1...");

		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
			// Datacenters are the resource providers in CloudSim. We need at
			// list one of them to run a CloudSim simulation
			Datacenter datacenter0 = createDatacenter("Datacenter_0");

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

			// Fourth step: Create one virtual machine
			vmlist = new ArrayList<Vm>();

			// VM description
			int vmid = 0;
			int mips = 1000;
			long size = 10000; // image size (MB)
			int ram = 512; // vm memory (MB)
			long bw = 1000;
			int pesNumber = 1; // number of cpus
			String vmm = "Xen"; // VMM name

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

			// add the VM to the vmList
			vmlist.add(vm);

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

			// Fifth step: Create one Cloudlet
			cloudletList = new ArrayList<Cloudlet>();

			// Cloudlet properties
			int id = 0;
			long length = 400000;
			long fileSize = 300;
			long outputSize = 300;
			UtilizationModel utilizationModel = new UtilizationModelFull();

			Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
			cloudlet.setUserId(brokerId);
			cloudlet.setVmId(vmid);

			// add the cloudlet to the list
			cloudletList.add(cloudlet);

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

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

			CloudSim.stopSimulation();

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

```
                        printCloudletList(newList);

                        Log.printLine("CloudSimExample1 finished!");
                } catch (Exception e) {
                        e.printStackTrace();
                        Log.printLine("Unwanted errors happen");
                }
        }

        /**
         * Creates the datacenter.
         *
         * @param name the name
         *
         * @return the datacenter
         */
        private static Datacenter createDatacenter(String name) {

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

                // 2. A Machine contains one or more PEs or CPUs/Cores.
                // In this example, it will have only one core.
                List<Pe> peList = new ArrayList<Pe>();

                int mips = 1000;

                // 3. Create PEs and add these into a list.
                peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and
MIPS Rating

                // 4. Create Host with its id and list of PEs and add them to the list
                // of machines
                int hostId = 0;
                int ram = 2048; // host memory (MB)
                long storage = 1000000; // host storage
                int bw = 10000;

                hostList.add(
                        new Host(
                                hostId,
                                new RamProvisionerSimple(ram),
                                new BwProvisionerSimple(bw),
                                storage,
                                peList,
                                new VmSchedulerTimeShared(peList)
                        )
                ); // This is our machine

                // 5. Create a DatacenterCharacteristics object that stores the
                // properties of a data center: architecture, OS, list of
                // Machines, allocation policy: time- or space-shared, time zone
                // and its price (G$/Pe time unit).
                String arch = "x86"; // system architecture
                String 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 = 0.001; // the cost of using storage in this
                                                                                // resource
                double costPerBw = 0.0; // the cost of using bw in this resource
                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
        /**
         * Creates the broker.
         *
         * @return the datacenter broker
```

```
	 */
	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 + "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()));
			}
		}
```

```
        }
}
```

OUTPUT:



Output - cloud_sim1 (run)

```
run:
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
400.1: Broker: Cloudlet 0 received
400.1: Broker: All Cloudlets executed. Finishing...
400.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time     Start Time    Finish Time
    0         SUCCESS         2              0        400        0.1          400.1
CloudSimExample1 finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

RESULT:

Thus the simulation of a Cloud Scenario using Cloudsim was executed successfully and the output is verified.

Ex.No:7b	SIMULATION OF A CLOUD SCENARIO WITH TWO DATACENTER USING CLOUDSIM
Date: 19.09.22	

AIM:

To simulate a cloud scenario with two datacenter using Cloudsim in Netbeans and to learn about the working of Cloudsim.

ALGORITHM:

1. Start
2. Initialize two brokers for the two cloudlets along with its entities.
3. Assign the necessary cloud resources needed and start two Virtual Machines using that cloud resource.
4. Now, assign the already created cloudlet to the newly created Virtual Machine.
5. After assigning, the broker receives the necessary cloudlet and starts executing.
6. All the Virtual Machines are destroyed after the necessary cloudlets get executed.
7. Now, notify all cloudsim entities for shutting down.
8. Finally the datacenter and broker are shut down as well.
9. Print the obtained output from clousim simulation to the user.
10. Stop

PROGRAM:

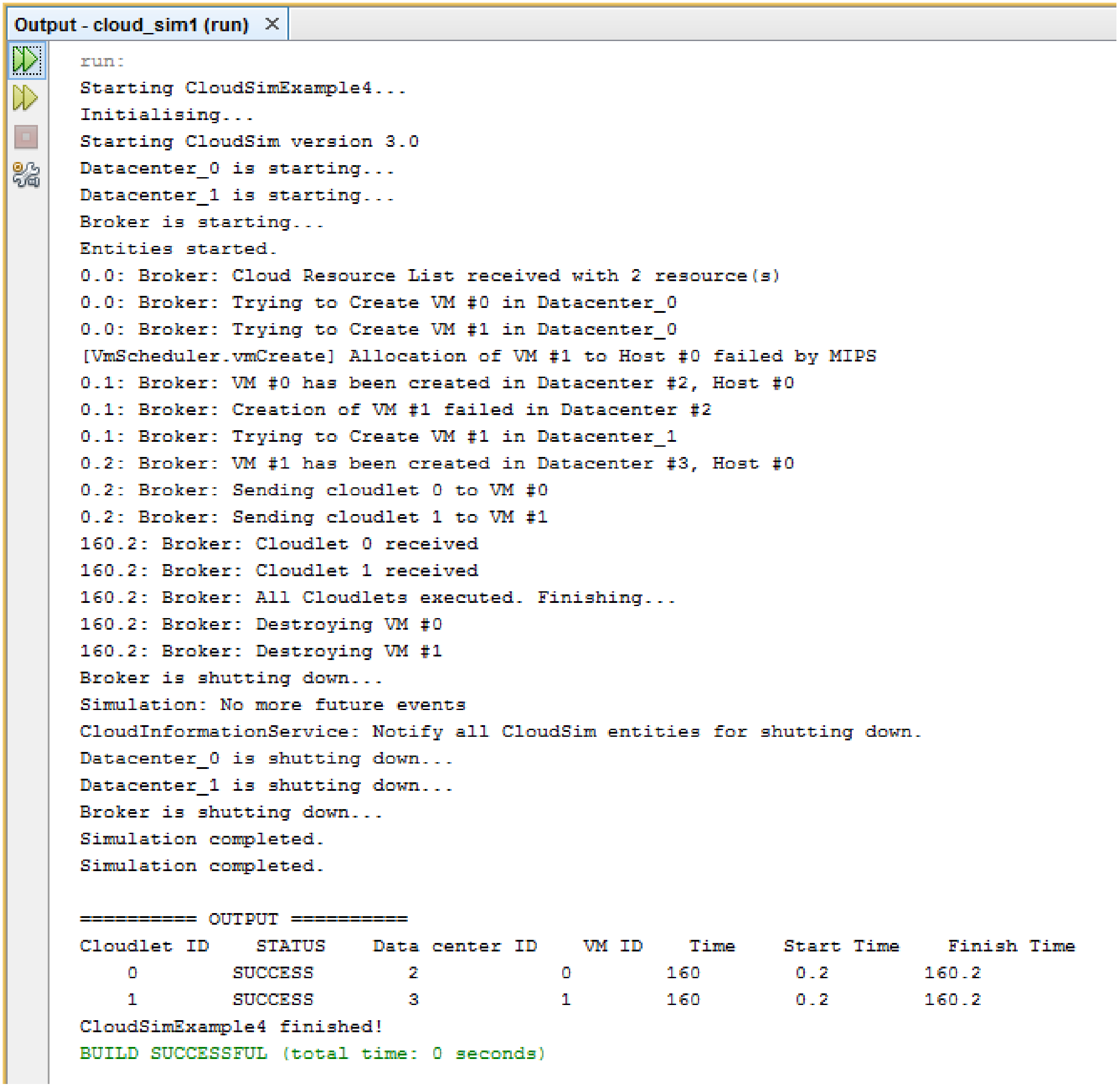
```
package org.cloudbus.cloudsim.examples;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.LinkedList;
import java.util.List;

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.*;


/**
 * A simple example showing how to create
 * two datacenters with one host each and
 * run two cloudlets on them.
 */
public class CloudSimExample4 {
	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example
	 */
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample4...");

		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 GridSim 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 one virtual machine
			vmlist = new ArrayList<Vm>();

			//VM description
			int vmid = 0;
			int mips = 250;
			long size = 10000; //image size (MB)
			int ram = 512; //vm memory (MB)
			long bw = 1000;
			int pesNumber = 1; //number of cpus
```

```
                String vmm = "Xen"; //VMM name

                //create two VMs
                Vm vm1 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());

                vmid++;
                Vm vm2 = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());

                //add the VMs to the vmList
                vmlist.add(vm1);
                vmlist.add(vm2);

                //submit vm list to the broker
                broker.submitVmList(vmlist);
                //Fifth step: Create two Cloudlets
                cloudletList = new ArrayList<Cloudlet>();

                //Cloudlet properties
                int id = 0;
                long length = 40000;
                long fileSize = 300;
                long outputSize = 300;
                UtilizationModel utilizationModel = new UtilizationModelFull();

                Cloudlet cloudlet1 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                cloudlet1.setUserId(brokerId);
                id++;
                Cloudlet cloudlet2 = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
                cloudlet2.setUserId(brokerId);
                //add the cloudlets to the list
                cloudletList.add(cloudlet1);
                cloudletList.add(cloudlet2);
                //submit cloudlet list to the broker
                broker.submitCloudletList(cloudletList);


                //bind the cloudlets to the vms. This way, the broker
                // will submit the bound cloudlets only to the specific VM
                broker.bindCloudletToVm(cloudlet1.getCloudletId(),vm1.getId());
                broker.bindCloudletToVm(cloudlet2.getCloudletId(),vm2.getId());
                // 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("CloudSimExample4 finished!");
		}
		catch (Exception e) {
			e.printStackTrace();
			Log.printLine("The simulation has been terminated due to an unexpected
error");
		}
	}

	private static Datacenter createDatacenter(String name){

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

		// 2. A Machine contains one or more PEs or CPUs/Cores.
		// In this example, it will have only one core.
		List<Pe> peList = new ArrayList<Pe>();

		int mips = 1000;

		// 3. Create PEs and add these into a list.
		peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe id and
MIPS Rating

		//4. Create Host with its id and list of PEs and add them to the list of machines
		int hostId=0;
		int ram = 2048; //host memory (MB)
		long storage = 1000000; //host storage
		int bw = 10000;


		//in this example, the VMAllocatonPolicy in use is SpaceShared. It means that
only one VM
		//is allowed to run on each Pe. As each Host has only one Pe, only one VM can
run on each Host.
		hostList.add(
			new Host(
				hostId,
                                        new RamProvisionerSimple(ram),
                                        new BwProvisionerSimple(bw),
                                        storage,
                                        peList,
                                        new VmSchedulerSpaceShared(peList)
                                )
                        ); // This is our first machine

                        // 5. Create a DatacenterCharacteristics object that stores the
                        //    properties of a data center: architecture, OS, list of
                        //    Machines, allocation policy: time- or space-shared, time zone
                        //    and its price (G$/Pe time unit).
                        String arch = "x86";      // system architecture
                        String 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 = 0.001;          // the cost of using storage in this resource
                        double costPerBw = 0.0;                                 // the cost of using bw in this
resource
                        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 + "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()));
			}
		}

	}
}
```

OUTPUT:



```
run:
Starting CloudSimExample4...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
[VmScheduler.vmCreate] Allocation of VM #1 to Host #0 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Creation of VM #1 failed in Datacenter #2
0.1: Broker: Trying to Create VM #1 in Datacenter_1
0.2: Broker: VM #1 has been created in Datacenter #3, Host #0
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
160.2: Broker: Cloudlet 0 received
160.2: Broker: Cloudlet 1 received
160.2: Broker: All Cloudlets executed. Finishing...
160.2: Broker: Destroying VM #0
160.2: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0         SUCCESS         2              0        160       0.2          160.2
    1         SUCCESS         3              1        160       0.2          160.2
CloudSimExample4 finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

RESULT:

Thus the simulation of two data centers using Cloudsim has been completed successfully and the output is verified.

Ex.No:7c	SCHEDULING ALGORITHM IN CLOUDSIM ROUND ROBIN
Date: 19.10.22	

AIM:

To run the Round Robin scheduling algorithm with the help of Cloudsim.

ALGORITHM:

1. Start
2. Initiate two datacenters Datacenter 0, 1 and a broker Broker_0.
3. Now, create four Virtual Machines namely VM 0, 1, 2, and 3 respectively. The VM's will be created and will be allocated to the respective hosts in the Datacenter.
4. Then, send the cloudlets to the respective Virtual Machines with the help of Broker.
5. The broker allocates the cloudlets and a message acknowledging its transfer will be displayed.
6. After that, all the Virtual Machines will be destroyed and broker will shut down.
7. Finally the simulation is complete and the output will be displayed on the screen.
8. Stop

PROGRAM:

```
package cloudsim_round_robin;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

import org.cloudbus.cloudsim.*;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.*;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CloudSimRoundRobin
{
        /** The cloudlet list. */
        private static List<Cloudlet> cloudletList;

        /** The vmList. */
```

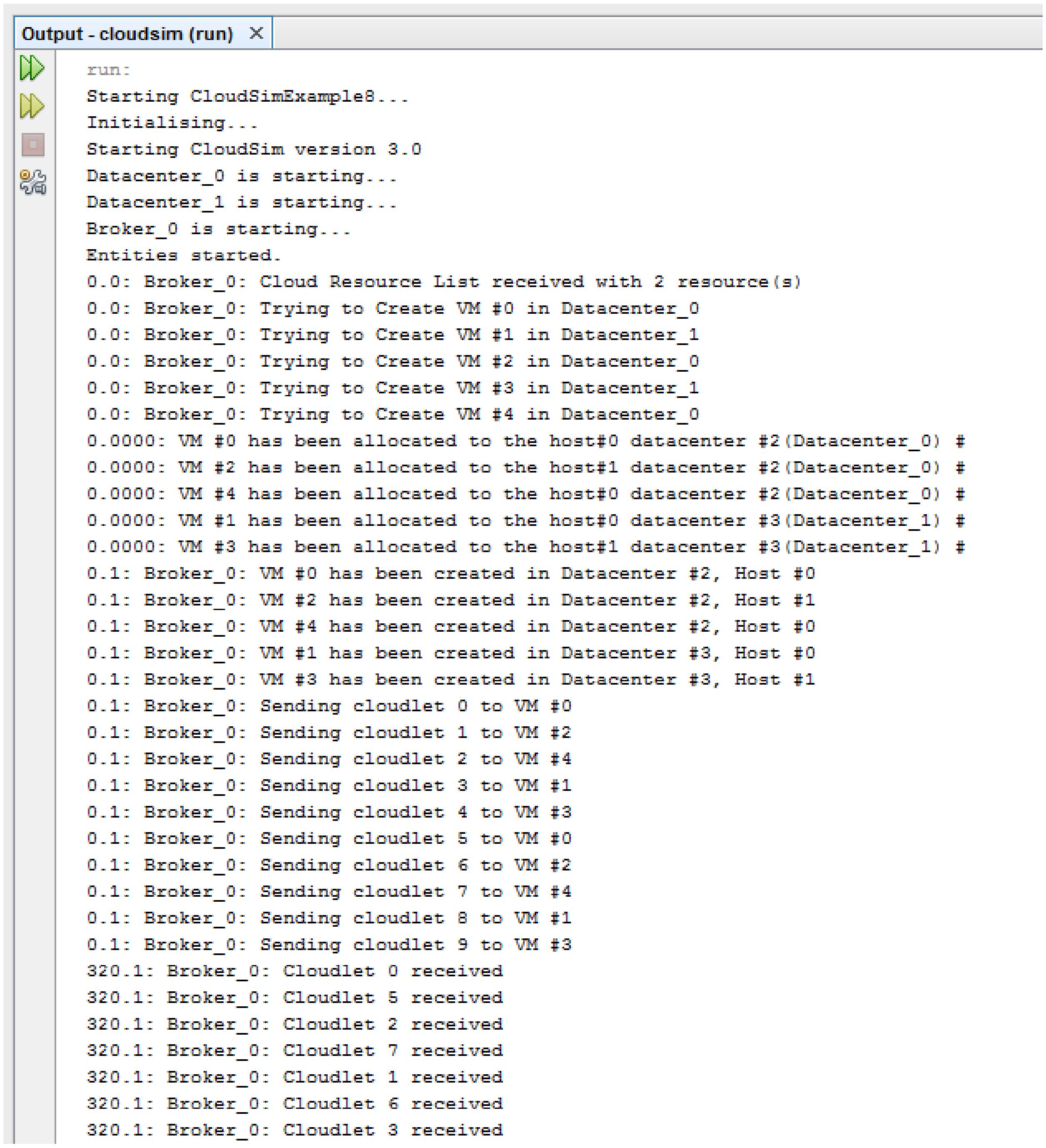
```
	private static List<Vm> vmList;

	private static List<Vm> createVM(int userId, int vms, int idShift)
	{
		//Creates a container to store VMs. This list is passed to the broker later
		LinkedList<Vm> vmList = new LinkedList<Vm>();

		//VM Parameters
		long size = 10000; //image size (MB)
		int ram = 512; //vm memory (MB)
		int mips = 250;
		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(idShift + i, userId, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());
			vmList.add(vm[i]);
		}

		return vmList;
	}

	private static List<Cloudlet> createCloudlet(int userId, int cloudlets, int idShift)
	{
		// Creates a container to store Cloudlets
		LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

		//cloudlet parameters
		long length = 40000;
		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(idShift + i, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel);
```

```
				// setting the owner of these Cloudlets
				cloudlet[i].setUserId(userId);
				list.add(cloudlet[i]);
			}

			return list;
		}


		/**
		 * Creates main() to run this example
		 */
		public static void main(String[] args)
		{
			Log.printLine("Starting CloudSimExample8...");

			try
			{
				// First step: Initialize the CloudSim package. It should be called
				// before creating any entities.
				int num_user = 2;   // 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);

				//GlobalBroker globalBroker = new GlobalBroker("GlobalBroker");

				// Second step: Create Datacenters
				//Datacenters are the resource providers in CloudSim. We need at list one
of them to run a CloudSim simulation
				Datacenter datacenter0 = createDatacenter("Datacenter_0");
				Datacenter datacenter1 = createDatacenter("Datacenter_1");

				//Third step: Create Broker
				DatacenterBroker broker = createBroker("Broker_0");
				int brokerId = broker.getId();

				//Fourth step: Create VMs and Cloudlets and send them to broker
				vmList = createVM(brokerId, 5, 0); //creating 5 vms
				cloudletList = createCloudlet(brokerId, 10, 0); // creating 10 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();
			//newList.addAll(globalBroker.getBroker().getCloudletReceivedList());

			CloudSim.stopSimulation();

			printCloudletList(newList);

			//Print the debt of each user to each datacenter
			//datacenter0.printDebts();
			//datacenter1.printDebts();

			Log.printLine(CloudSimRoundRobin.class.getName() + " finished!");
		}
		catch (Exception e)
		{
			e.printStackTrace();
			Log.printLine("The simulation has been terminated due to an unexpected
error");
		}
	}

	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)));
```

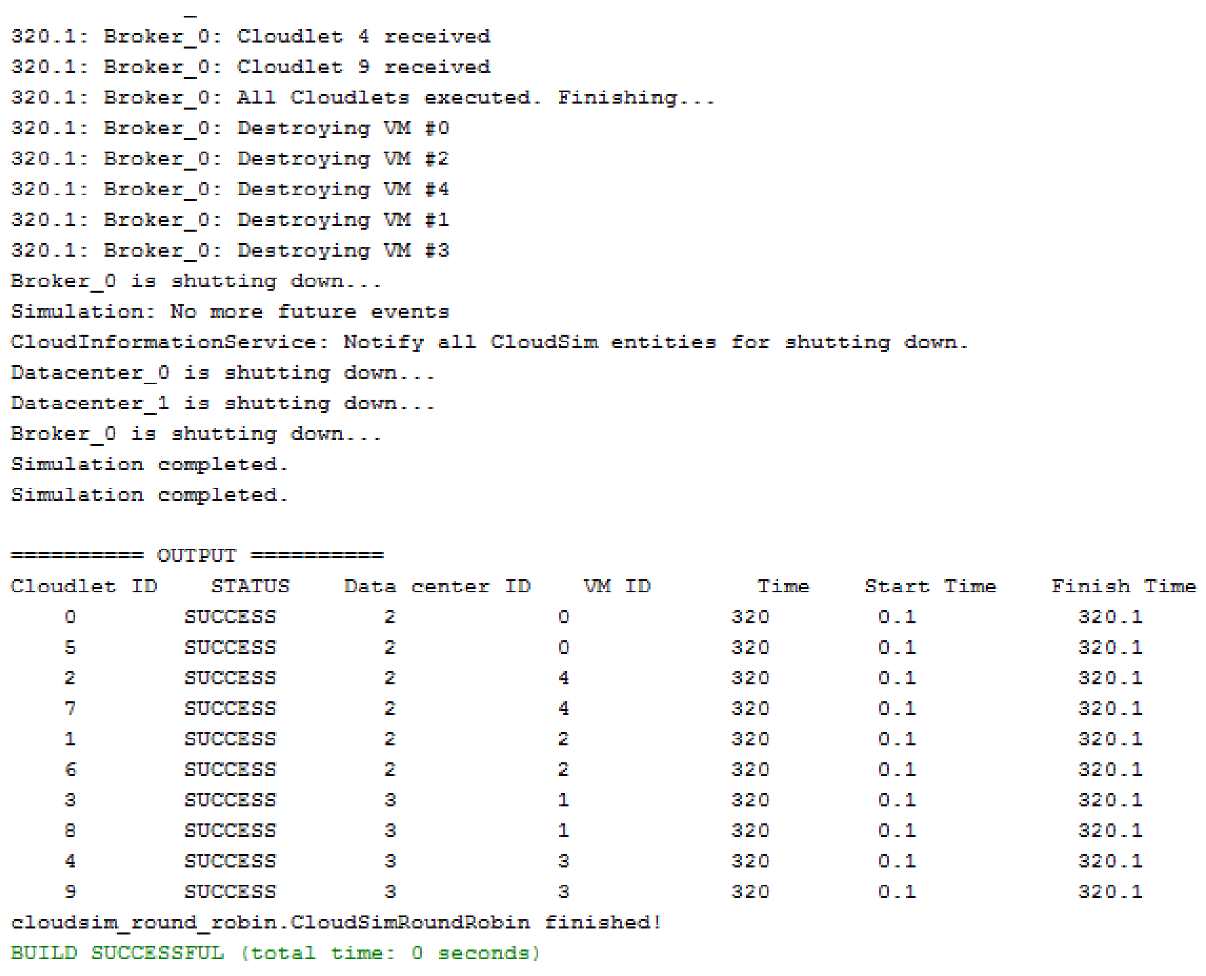
```
//Another list, for a dual-core machine
List<Pe> peList2 = new ArrayList<Pe>();

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 machines
int hostId=0;
int ram = 16384; //host memory (MB)
long storage = 1000000; //host storage
int bw = 10000;

hostList.add(
        new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList1,
                new VmSchedulerTimeShared(peList1)
        )
); // This is our first machine

hostId++;

hostList.add(
        new Host(
                hostId,
                new RamProvisionerSimple(ram),
                new BwProvisionerSimple(bw),
                storage,
                peList2,
                new VmSchedulerTimeShared(peList2)
        )
); // Second machine

// 5. Create a DatacenterCharacteristics object that stores the
//    properties of a data center: architecture, OS, list of
//    Machines, allocation policy: time- or space-shared, time zone
//    and its price (G$/Pe time unit).
String arch = "x86";      // system architecture
String 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
resource
		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
		{
			RoundRobinVmAllocationPolicy vm_policy = new
RoundRobinVmAllocationPolicy(hostList);
			datacenter = new Datacenter(name, characteristics, vm_policy,
storageList, 0);
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}

		return datacenter;
	}


	public static class VmAllocationPolicyMinimum extends
org.cloudbus.cloudsim.VmAllocationPolicy
	{

		private Map<String, Host> vm_table = new HashMap<String, Host>();

		private final Hosts hosts;
		private Datacenter datacenter;

		public VmAllocationPolicyMinimum(List<? extends Host> list)
		{
			super(list);
			hosts = new Hosts(list);
		}

		public void setDatacenter(Datacenter datacenter)
		{
			this.datacenter = datacenter;
		}

		public Datacenter getDatacenter()
		{
			return datacenter;
		}

		@Override
		public boolean allocateHostForVm(Vm vm)
		{

			if (this.vm_table.containsKey(vm.getUid()))
				return true;

			boolean vm_allocated = false;
			int tries = 0;

			do
			{
				Host host =
this.hosts.getWithMinimumNumberOfPesEquals(vm.getNumberOfPes());
				vm_allocated = this.allocateHostForVm(vm, host);

			} while (!vm_allocated && tries++ < hosts.size());

			return vm_allocated;
		}

		@Override
		public boolean allocateHostForVm(Vm vm, Host host)
		{
			if (host != null && host.vmCreate(vm))
			{
				vm_table.put(vm.getUid(), host);
				Log.formatLine("%.4f: VM #" + vm.getId() + " has been allocated
to the host#" + host.getId() +
						" datacenter #" + host.getDatacenter().getId() + "("
+ host.getDatacenter().getName() + ") #",
						CloudSim.clock());
				return true;
			}
			return false;
		}

		@Override
		public List<Map<String, Object>> optimizeAllocation(List<? extends Vm>
vmList)
		{
			return null;
		}

		@Override
		public void deallocateHostForVm(Vm vm)
		{
			Host host = this.vm_table.remove(vm.getUid());

			if (host != null)
			{
				host.vmDestroy(vm);
			}
		}

		@Override
		public Host getHost(Vm vm)
		{
			return this.vm_table.get(vm.getUid());
		}

		@Override
		public Host getHost(int vmId, int userId)
		{
			return this.vm_table.get(Vm.getUid(userId, vmId));
		}
	}


	private static DatacenterBroker createBroker(String name) throws Exception
	{
		return new RoundRobinDatacenterBroker(name);
	}

	/**
	 * 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:


Output - cloudsim (run)

```
run:
Starting CloudSimExample8...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Datacenter_1 is starting...
Broker_0 is starting...
Entities started.
0.0: Broker_0: Cloud Resource List received with 2 resource(s)
0.0: Broker_0: Trying to Create VM #0 in Datacenter_0
0.0: Broker_0: Trying to Create VM #1 in Datacenter_1
0.0: Broker_0: Trying to Create VM #2 in Datacenter_0
0.0: Broker_0: Trying to Create VM #3 in Datacenter_1
0.0: Broker_0: Trying to Create VM #4 in Datacenter_0
0.0000: VM #0 has been allocated to the host#0 datacenter #2(Datacenter_0) #
0.0000: VM #2 has been allocated to the host#1 datacenter #2(Datacenter_0) #
0.0000: VM #4 has been allocated to the host#0 datacenter #2(Datacenter_0) #
0.0000: VM #1 has been allocated to the host#0 datacenter #3(Datacenter_1) #
0.0000: VM #3 has been allocated to the host#1 datacenter #3(Datacenter_1) #
0.1: Broker_0: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #2 has been created in Datacenter #2, Host #1
0.1: Broker_0: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker_0: VM #1 has been created in Datacenter #3, Host #0
0.1: Broker_0: VM #3 has been created in Datacenter #3, Host #1
0.1: Broker_0: Sending cloudlet 0 to VM #0
0.1: Broker_0: Sending cloudlet 1 to VM #2
0.1: Broker_0: Sending cloudlet 2 to VM #4
0.1: Broker_0: Sending cloudlet 3 to VM #1
0.1: Broker_0: Sending cloudlet 4 to VM #3
0.1: Broker_0: Sending cloudlet 5 to VM #0
0.1: Broker_0: Sending cloudlet 6 to VM #2
0.1: Broker_0: Sending cloudlet 7 to VM #4
0.1: Broker_0: Sending cloudlet 8 to VM #1
0.1: Broker_0: Sending cloudlet 9 to VM #3
320.1: Broker_0: Cloudlet 0 received
320.1: Broker_0: Cloudlet 5 received
320.1: Broker_0: Cloudlet 2 received
320.1: Broker_0: Cloudlet 7 received
320.1: Broker_0: Cloudlet 1 received
320.1: Broker_0: Cloudlet 6 received
320.1: Broker_0: Cloudlet 3 received
```



```
320.1: Broker_0: Cloudlet 4 received
320.1: Broker_0: Cloudlet 9 received
320.1: Broker_0: All Cloudlets executed. Finishing...
320.1: Broker_0: Destroying VM #0
320.1: Broker_0: Destroying VM #2
320.1: Broker_0: Destroying VM #4
320.1: Broker_0: Destroying VM #1
320.1: Broker_0: Destroying VM #3
Broker_0 is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker_0 is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID        Time    Start Time    Finish Time
    0        SUCCESS        2             0           320       0.1           320.1
    5        SUCCESS        2             0           320       0.1           320.1
    2        SUCCESS        2             4           320       0.1           320.1
    7        SUCCESS        2             4           320       0.1           320.1
    1        SUCCESS        2             2           320       0.1           320.1
    6        SUCCESS        2             2           320       0.1           320.1
    3        SUCCESS        3             1           320       0.1           320.1
    8        SUCCESS        3             1           320       0.1           320.1
    4        SUCCESS        3             3           320       0.1           320.1
    9        SUCCESS        3             3           320       0.1           320.1
cloudsim_round_robin.CloudSimRoundRobin finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

RESULT:

Thus the Round Robin scheduling algorithm using Cloudsim was executed successfully and the output has been verified.

.

Ex.No:8	LAUNCHING A VIRTUAL MACHINE USING TRYSTACK
Date: 07.11.22	

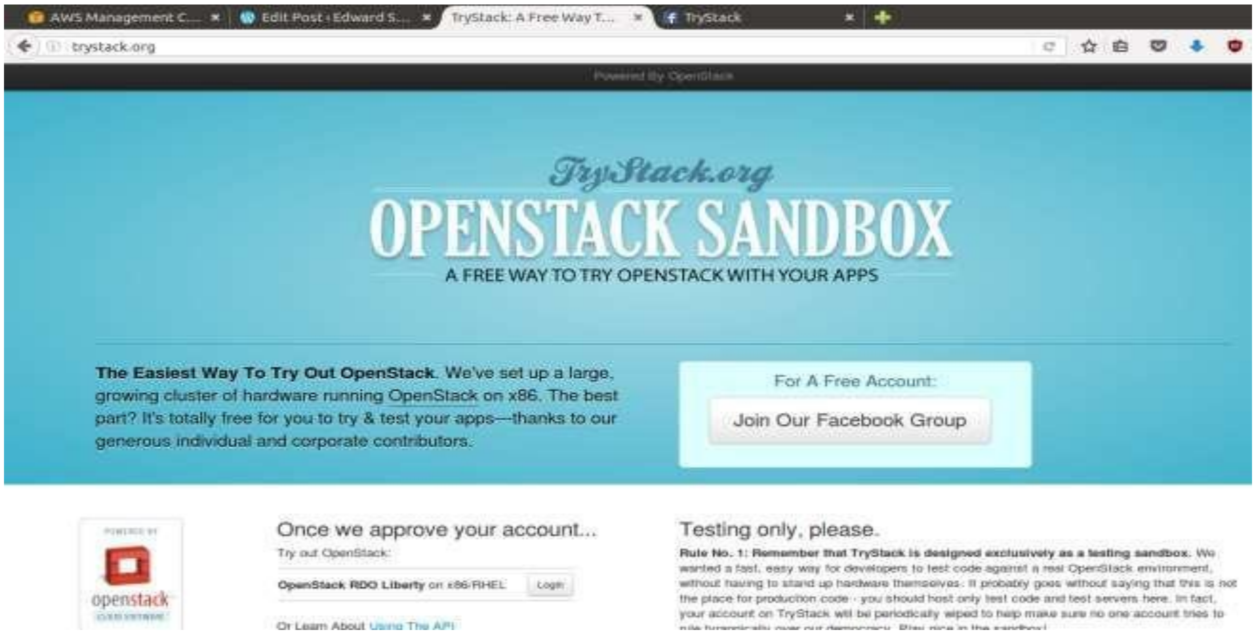
AIM:

To launch virtual machine online using trystack which is nothing but an online Openstack demo tool

ALGORITHM:

1. Start
2. Register in Trystack service by registering in Trystack Facebook group online.
3. After registering, log into the trystack portal with the created account..
4. Then create a network in Trystack by filling out the IP Address, Network address and Subnet.
5. Create an instance in Trystack by clicking Compute > Instances and then by clicking Launch Instance.
6. Now, import the needed Key Pair from the "Access & Security" tab..
7. Finally, select your network and launch your cloud instance by clicking the launch button..
8. Stop

PROCEDURE:



The Easiest Way To Try Out OpenStack. We've set up a large, growing cluster of hardware running OpenStack on x86. The best part? It's totally free for you to try & test your apps—thanks to our generous individual and corporate contributors.

Once we approve your account...

Try out OpenStack:

OpenStack RDO Liberty on x86/RHEL Login

Or Learn About Using The API

Testing only, please.

Rule No. 1: Remember that TryStack is designed exclusively as a testing sandbox. We wanted a fast, easy way for developers to test code against a real OpenStack environment, without having to stand up hardware themselves. It probably goes without saying that this is not the place for production code - you should host only test code and test servers here. In fact, your account on TryStack will be periodically wiped to help make sure no one account tries to

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 Services.
86In order to try OpenStack in TryStack, you must register yourself by joining TryStack Facebook Group. The acceptance of the group needs a couple days because it's approved manually. After you have been accepted in the TryStack Group, you can log in to TryStack.

Step 1: Create Network

1. Go to **Network > Networks** and then click **Create Network**.
2. In the Network tab, fill **Network Name** for example internal and then

   click **Next**.

3. In **Subnet** tab,

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

Step 2: Create Instance

1. Go to **Compute > Instances** and then click **Launch Instance**.
2. In **Details** tab,

   a

1. Fill **Instance Name**, for example Ubuntu 1.
2. Select **Flavor**, for example m1.medium.
3. Fill **Instance Count** with **1**.
4. Select **Instance Boot Source** with **Boot from Image**.
5. Select **Image Name** with **Ubuntu 14.04 amd64 (243.7 MB)** if you want to install Ubuntu 14.04 in your virtual machine.

3. In **Access & Security** tab,

1. Click [**+**] button of **Key Pair** to import key pair. This key pair is a public and private key that we will use to connect to the instance from our machine.
2. In **Import Key Pair** dialog,
1. Fill **Key Pair Name** with your machine name (for example Edward-Key).
2. Fill **Public Key** with your **SSH public key** (usually is in ~/.ssh/id_rsa.pub). See description in Import Key Pair dialog box for more information. If you are using Windows, you can use **Puttygen** to generate key pair.
3. Click **Import key pair**.

3. In **Security Groups**, mark/check **default**.

4. In **Networking** tab,

1. In **Selected Networks**, select network that have been created in Step 1, for

example internal. 5. Click **Launch**.

6. If you want to create multiple instances, you can repeat step 1-5. I created one more instance with instance name Ubuntu 2.

Step 3: Create Router

To make our network has an internet connection, we need a router that running as the gateway to the internet.

1. Go to **Network > Routers** and then click **Create Router**.

2. Fill **Router Name** for example router1 and then click **Create router**.

3. Click on your **router name link**, for example router1, **Router Details**

page. 4. Click **Set Gateway** button in upper right:

1. Select **External networks** with **external**.

2. Then **OK**.

5. Click **Add Interface** button.

1. Select **Subnet** with the network that you have been created in

Step 1. 2. Click **Add interface**.

6. Go to **Network > Network Topology**. You will see the network topology. In the example, there are two network, i.e. external and internal, those are bridged by a router. There are instances those are joined to internal network.

Step 4: Configure Floating IP Address

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

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

Step 5: Configure Access & Security

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

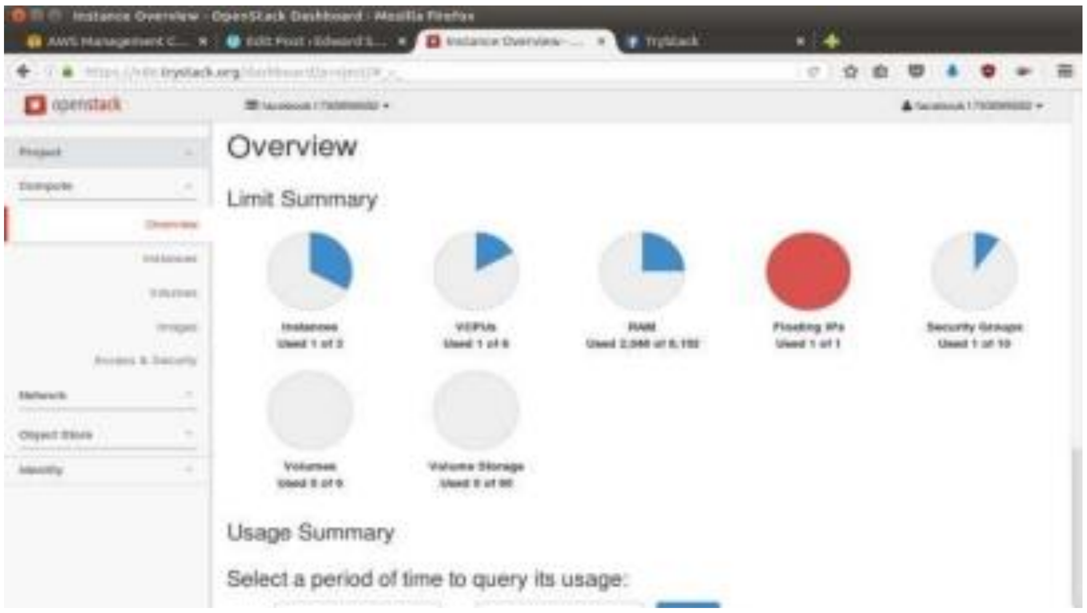
1. Go to **Compute > Access & Security** and then open **Security**

Groups tab.

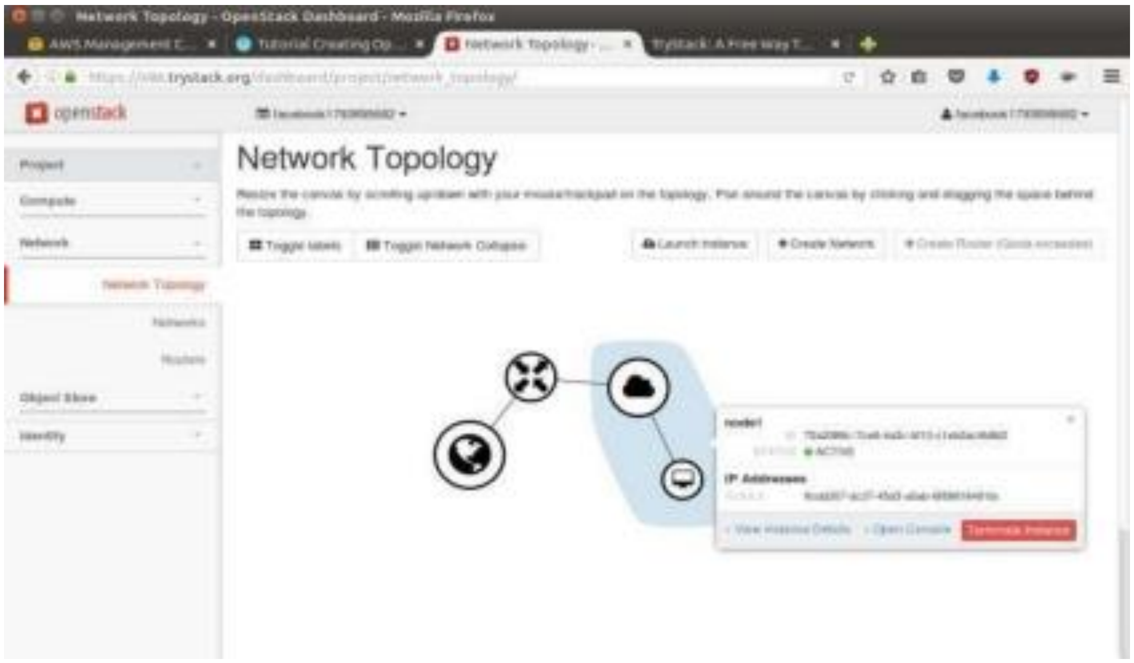
2. In **default** row, click **Manage Rules**.
3. Click **Add Rule**, choose **ALL ICMP** rule to enable ping into your instance, and then click **Add**.
4. Click **Add Rule**, choose **HTTP** rule to open HTTP port (port 80), and then click **Add**.
5. Click **Add Rule**, choose **SSH** rule to open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

Step 6: SSH to Your Instance

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



The instance will be accessible through the internet (have a public IP address). The final topology will like:



The instance will be connected to a local network and the local network will be connected to internet.

RESULT:

Thus the creation of virtual machine using Trystack is completed successfully and the output has been verified.

Ex.No:9	INSTALLATION OF HADOOP SINGLE NODE CLUSTER AND RUN WORDCOUNT
Date:31.10.22	

AIM:

To install Hadoop single node cluster and to run applications such as Wordcount in Hadoop single node cluster.

ALGORITHM:

1. Start
2. First download and install Java in your system as Hadoop works with the help of Java.
3. Then, download and install Hadoop from the official Apache Hadoop website. Choose your Operating System before downloading.
4. After installation select the directory that you'll be working with in Hadoop.
5. Then, start dfs and yarn services and check the no of nodes currently running.
6. Now, check if Hadoop is installed correctly with the help of your web browser.
7. Add a file in Hadoop and check if the file can be viewed in Hadoop.
8. Implement the grep and word count command on the file to be processed in Hadoop.
9. Now, the no of words in the file will be displayed. This is the output needed.
10. Stop

PROCEDURE:

1. Update the source list

```
$ sudo apt-get update
```

2. Install Java

```
$ sudo  apt-get  install  openjdk-7-jdk
$ java  -version
```

3. Adding a dedicated Hadoop user

```
$ sudo  addgroup  hadoop
$ sudo adduser  --ingroup  hadoop  hduser
```

4. Installing ssh

```
$ sudo  apt-get  install  ssh
```

5. Create and Setup SSH certificates

```
$ su   hduser
$ ssh-keygen -t rsa -P "  "
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
To  check the ssh  works ,
$  ssh  localhost
```

6. Install Hadoop

```
$wget  https://dlcdn.apache.org/hadoop/common/hadoop-2.10.2/hadoop-2.10.2.tar.gz
$ tar xvzf   hadoop-2.10.2.tar.gz
$ sudo  mv  *  /usr/local/hadoop
hduser  make root privilege
$ sudo  su  hduser
$ sudo  mv  *  /usr/local/hadoop
$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

7. Setup configuration files

```
Check the java installed location
Append to the end of  ~/.bashrc
$  vi   ~/.bashrc
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

```
$ source  ~/.bashrc
```

To check JAVA_HOME

```
$ javac  -version
$ which javac
$ readlink -f /usr/bin/javac
```

2. Edit the hadoop environment variable

```
$  vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

3. Edit the core-site.xml

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
$ vi /usr/local/hadoop/etc/hadoop/core-site.xml
<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>
 <property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

4. Edit mapred-site.xml

```
cp  /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
            /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Edit the mapred-site.xml between <configuration> </configuration>

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

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

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

```
$ vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Format the new Hadoop filesystem

/usr/local/hadoop_store/hdfs/namenode

```
$ hadoop  namenode  -format
```

Start Hadoop

```
/usr/local/hadoop/sbin$ cd /usr/local/hadoop/sbin
/usr/local/hadoop/sbin S ls
/usr/local/hadoop/sbin $ sudo  su  hduser
/usr/local/hadoop/sbin $ start-all.sh
```

To check

```
/usr/local/hadoop/sbin S jps
```

```
$ netstat -plten | grep java
```

To stop the hadoop

```
/usr/local/hadoop/sbin $ stop-all.sh
```

Web interface

http://localhost:50070/

Secondary Namenode

http://localhost:50090/status.jsp

Hadoop Distributed File System (HDFS) shell commands

1. To list the contents of a directory

```
$ hadoop  fs  -ls   /user
Found 8 items
drwxr-xr-x   - cloudera cloudera           0 2015-03-28 23:43 /user/cloudera
drwxr-xr-x   - hdfs     supergroup         0 2015-03-14 20:11 /user/hdfs
drwxr-xr-x   - mapred   hadoop             0 2015-03-15 14:08 /user/history
drwxrwxrwx   - hive     hive               0 2014-12-18 04:33 /user/hive
drwxrwxr-x   - hue      hue                0 2015-03-21 15:34 /user/hue
drwxrwxrwx   - oozie    oozie              0 2014-12-18 04:34 /user/oozie
drwxr-xr-x   - sample   sample             0 2015-03-14 22:05 /user/sample
drwxr-xr-x   - spark    spark              0 2014-12-18 04:34 /user/spark
```

2. To view the contents of a file

```
$ hadoop  fs  -cat  /user/hdfs/dir1/myfile
This is myfile
```

3. To create a directory in HDFS

```
$ sudo  su  hdfs
bash-4.1$ hadoop  fs  -mkdir  /user/hdfs/dir1
```

4. To copy src files from local file system to the Hadoop data file system

```
$ hadoop  fs  -put  <local_src>  <HDFS_dest_Path>
$ whoami
hdfs
$ hadoop fs -put myfile /user/hdfs/dir1
```

5. To copy files from the Hadoop data file system to the local file system

```
$ hadoop  fs  -get   /user/hdfs/dir1/myfile .
```

6. To copy a file from source to destination

```
$ hadoop fs -cp /user/hdfs/dir1/myfile /user/hdfs/dir2
$  hadoop fs -ls /user/hdfs/dir2/myfile
-rw-r--r--   1  hdfs  supergroup          7 2015-03-29 11:43 /user/hdfs/dir2/myfile
```

7. To copy a file from Local file system to HDFS

```
hadoop  fs  -copyFromLocal  <localsrc>  URI
$ pwd
/home/cloudera/workspace/temp
$ ls
myfile   myfile2
$ hadoop  fs  -copyFromLocal  myfile2   /user/hdfs/dir2/
$ hadoop  fs  -cat   /user/hdfs/dir2/myfile2
This is myfile2
```

8. To copy a file to Local file system from HDFS

```
$ hadoop fs -copyToLocal /user/hdfs/dir1/myfile myfile3
```

9. To remove a file from HDFS

```
$ hadoop fs  -ls  /user/hdfs/dir2
Found 2 items
-rw-r--r--   1 hdfs supergroup          7 2015-03-29 11:43 /user/hdfs/dir2/myfile
-rw-r--r--   1 hdfs supergroup         16 2015-03-29 11:52 /user/hdfs/dir2/myfile2
$ hadoop  fs  -rm   /user/hdfs/dir2/myfile2
$ hadoop  fs  -ls   /user/hdfs/dir2
-rw-r--r--   1 hdfs supergroup          7 2015-03-29 11:43 /user/hdfs/dir2/myfile
```

10. To remove a directory from HDFS

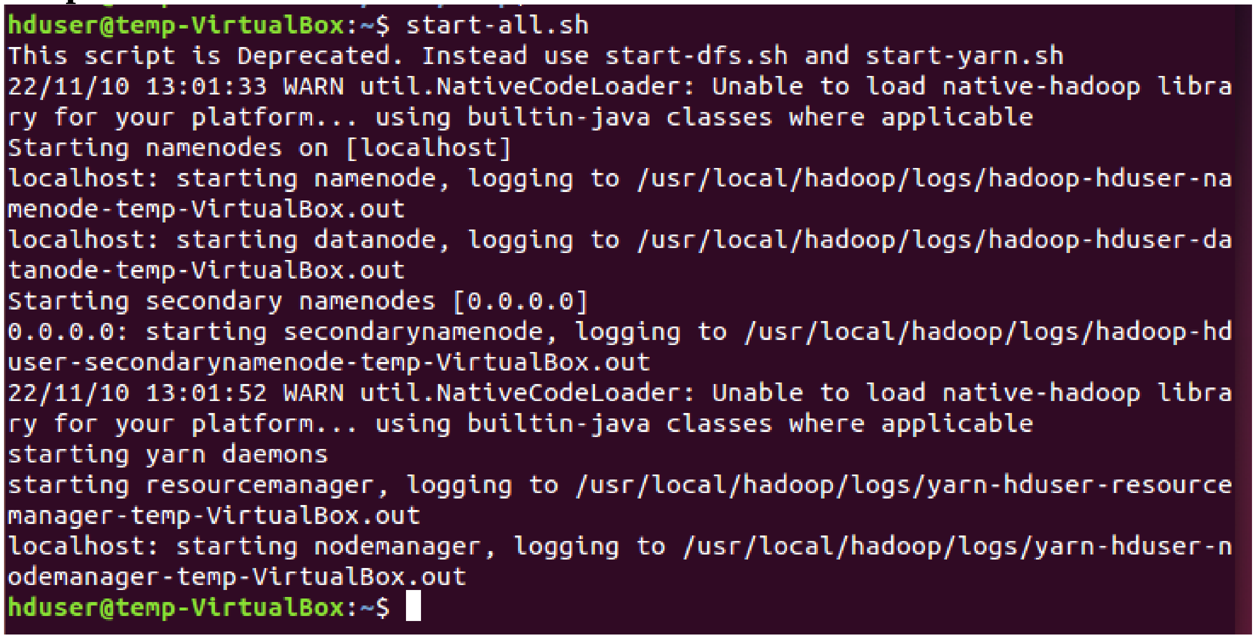
```
bash-4.1$ hadoop fs -ls /user/hdfs/
Found 3 items
drwxr-xr-x   - hdfs supergroup          0 2015-03-28 14:08 /user/hdfs/.Trash
drwxr-xr-x   - hdfs supergroup          0 2015-03-29 10:55 /user/hdfs/dir1
drwxr-xr-x   - hdfs supergroup          0 2015-03-29 12:12 /user/hdfs/dir2
bash-4.1$ hadoop fs -rm -r /user/hdfs/dir2
bash-4.1$ hadoop fs -ls /user/hdfs/
drwxr-xr-x   - hdfs supergroup          0 2015-03-28 14:08 /user/hdfs/.Trash
drwxr-xr-x   - hdfs supergroup          0 2015-03-29 10:55 /user/hdfs/dir1
```

Program:

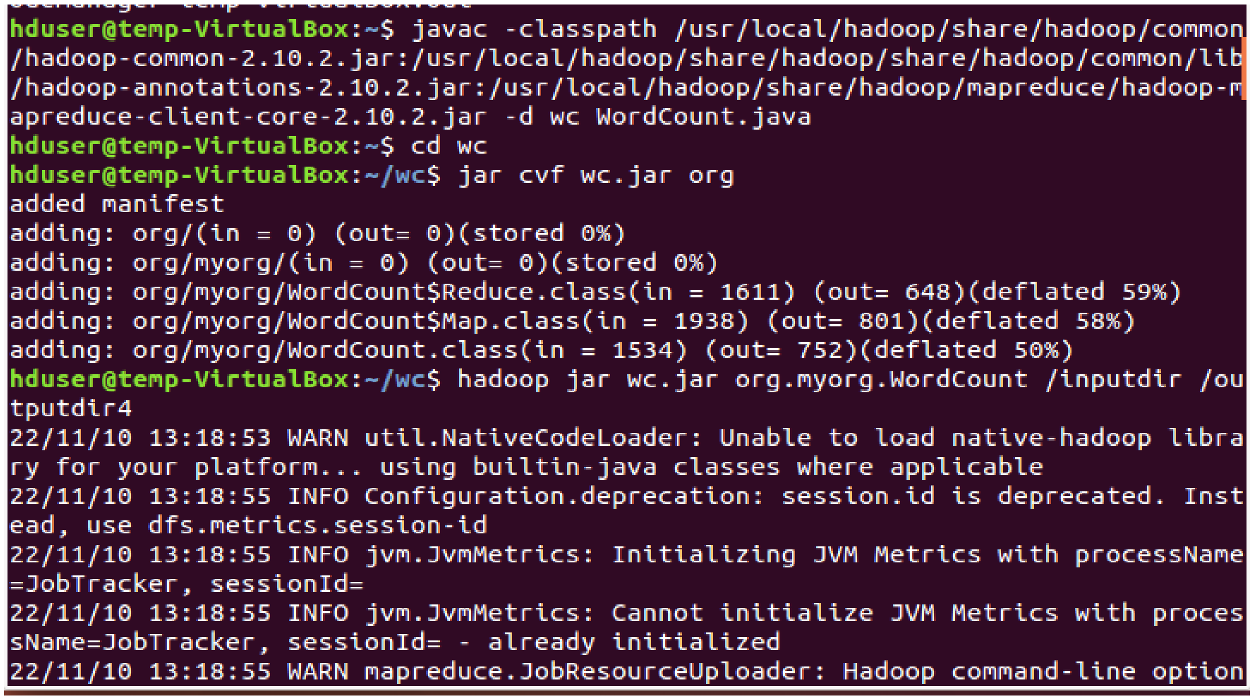
```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class WordCount {
  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text,
Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter     reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
      }
    }
  }
  public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }
  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
```

```
   conf.setInputFormat(TextInputFormat.class);
   conf.setOutputFormat(TextOutputFormat.class);
   FileInputFormat.setInputPaths(conf, new Path(args[0]));
   FileOutputFormat.setOutputPath(conf, new Path(args[1]));
   JobClient.runJob(conf);
  }
}
```

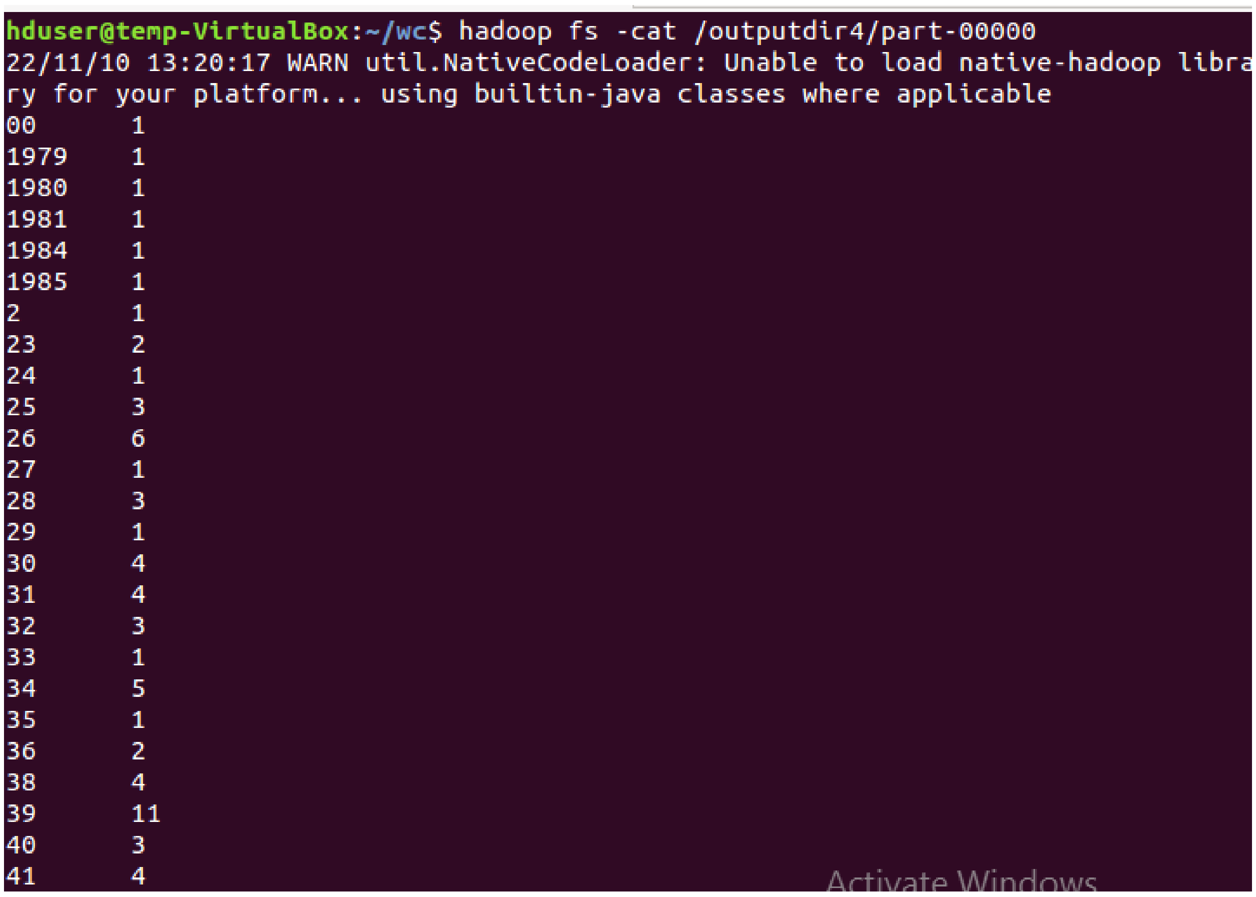
Output:



```
hduser@temp-VirtualBox:~$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
22/11/10 13:01:33 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-na
menode-temp-VirtualBox.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-da
tanode-temp-VirtualBox.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hd
user-secondarynamenode-temp-VirtualBox.out
22/11/10 13:01:52 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resource
manager-temp-VirtualBox.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-n
odemanager-temp-VirtualBox.out
hduser@temp-VirtualBox:~$
```



```
hduser@temp-VirtualBox:~$ javac -classpath /usr/local/hadoop/share/hadoop/common
/hadoop-common-2.10.2.jar:/usr/local/hadoop/share/hadoop/share/hadoop/common/lib
/hadoop-annotations-2.10.2.jar:/usr/local/hadoop/share/hadoop/mapreduce/hadoop-m
apreduce-client-core-2.10.2.jar -d wc WordCount.java
hduser@temp-VirtualBox:~$ cd wc
hduser@temp-VirtualBox:~/wc$ jar cvf wc.jar org
added manifest
adding: org/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/(in = 0) (out= 0)(stored 0%)
adding: org/myorg/WordCount$Reduce.class(in = 1611) (out= 648)(deflated 59%)
adding: org/myorg/WordCount$Map.class(in = 1938) (out= 801)(deflated 58%)
adding: org/myorg/WordCount.class(in = 1534) (out= 752)(deflated 50%)
hduser@temp-VirtualBox:~/wc$ hadoop jar wc.jar org.myorg.WordCount /inputdir /ou
tputdir4
22/11/10 13:18:53 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
22/11/10 13:18:55 INFO Configuration.deprecation: session.id is deprecated. Inst
ead, use dfs.metrics.session-id
22/11/10 13:18:55 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName
=JobTracker, sessionId=
22/11/10 13:18:55 INFO jvm.JvmMetrics: Cannot initialize JVM Metrics with proces
sName=JobTracker, sessionId= - already initialized
22/11/10 13:18:55 WARN mapreduce.JobResourceUploader: Hadoop command-line option
```



```
hduser@temp-VirtualBox:~/wc$ hadoop fs -cat /outputdir4/part-00000
22/11/10 13:20:17 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
00      1
1979    1
1980    1
1981    1
1984    1
1985    1
2       1
23      2
24      1
25      3
26      6
27      1
28      3
29      1
30      4
31      4
32      3
33      1
34      5
35      1
36      2
38      4
39      11
40      3
41      4
```

RESULT:

Thus the installation of Hadoop single node cluster and running of Wordcount program has been completed successfully.