LIST OF EXPERIMENTS

Year: 2024-25 **Sem:** Even **Class:** Final T.Y (ECO)

Course Code: 21BTCO611R **Course Name:** Cloud Computing Laboratory

Expt. No.	Title of Experiment
1	Use gcc to compile c-programs. Split the programs to different modules and create anapplication using make commandr.
2	Use version control systems command to clone, commit, push, fetch, pull, checkout, reset, and delete repositories.
3	Install VirtualBox/VMware Workstation with different flavors of Linux or windows OS on top of windows7 or 8.
4	Install a C compiler in the virtual machine created using virtual box and execute Simple Programs
5	Install Google App Engine. Create hello world app and other simple web applications using python/java.
6	Use GAE launcher to launch the web applications.
7	Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim
8	Find a procedure to transfer the files from one virtual machine to another virtual machine.

Prof. Dnyaneshwar Kokare	Prof. Dr. Gopal Gawande	Prof. Dr. Sachin Takale
Course Teacher	**Academic Coordinator**	**I/C Head of Dept.**

Practical No-1

Problem Statement:

The task is to split a C program into different modules and create an application using the make command. The goal is to compile the modules separately and link them together into a final executable using the make utility. The solution should be able to handle any number of modules and should be easy to maintain.

Theory:

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules. In the context of C programming, this involves splitting a program into separate source files, with each file containing a single module. Each module provides a specific set of functions or variables, and these modules can be compiled separately and linked together to form the final executable.

The make command is a popular utility for building programs in Unix-like operating systems. It automates the process of building executables by managing dependencies between source files, and compiling only the files that have changed since the last build. By defining rules in a makefile, developers can create a set of instructions for building a program, making it easier to manage complex programs with many source files.

Using modular programming and the make command together can make it easier to manage and maintain large C programs, especially those with multiple contributors. By breaking the program into smaller, independent modules, it becomes easier to isolate bugs and make changes to specific parts of the program without affecting other modules. The make command then automates the process of building the program, ensuring that the final executable is always up-to-date and free of errors.

Here is an example of how to use gcc to compile C programs for splitting the programs into different modules and creating an application using the make command:

Suppose we have a program consisting of three files:

main.c: This file contains the main function that starts the program.

module1.c: This file contains functions related to module 1.

module2.c: This file contains functions related to module 2.

To compile these files and create an application using the make command, we can follow these steps:

1. Create a makefile: Create a makefile named "Makefile" (without the quotes) in the same directory as the C files. The makefile contains rules for building the application.

2. Define variables: Define variables for the C compiler, compiler flags, and the name of the output file.
   CC=gcc
   CFLAGS=-c -Wall
   LDFLAGS=
   EXECUTABLE=myapp

3. Define targets: Define targets for each object file and the executable.
   all: $(EXECUTABLE)

   $(EXECUTABLE): main.o module1.o module2.o
   $(CC) $(LDFLAGS) $^ -o $@

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

   module1.o: module1.c
   $(CC) $(CFLAGS) $< -o $@

   module2.o: module2.c
   $(CC) $(CFLAGS) $< -o $@

4. Build the application: Run the make command in the terminal to build the application.

   Make

This will compile each C file into an object file and link them together into an executable named "myapp". The make command will automatically rebuild the application if any of the source files change.

Note: The above makefile assumes that all C files are in the same directory. If the files are in different directories, you will need to modify the makefile accordingly.

Execution:

1. Create a new directory and navigate into it:

```
mkdir my_program

cd my_program
```

2. Create a source file for each module:

```
touch module1.c

touch module2.c
```

3. Write some code for each module. For example, module1.c might contain the following:

```
#include <stdio.h>

#include "module2.h"


void hello_world() {

  printf("Hello, world!\n");

  goodbye();

}
```

And module2.c might contain:

```
#include <stdio.h>


void goodbye() {

  printf("Goodbye, world!\n");

}
```

Note that module1 includes the header file module2.h, which defines the function goodbye().

4. Create header files for each module:

```
touch module1.h
touch module2.h
```

5. Define the functions in the header files. For example, module1.h might contain:

```
void hello_world();
```

6. And module2.h might contain:

```
void goodbye();
```

7. Create a makefile to automate the building process:

```
touch Makefile
```

8. And add the following contents:

```
CC = gcc
CFLAGS = -Wall -Wextra -pedantic
all: my_program
my_program: module1.o module2.o
    $(CC) $(CFLAGS) $^ -o $@
module1.o: module1.c module1.h module2.h
    $(CC) $(CFLAGS) -c module1.c -o $@
module2.o: module2.c module2.h
    $(CC) $(CFLAGS) -c module2.c -o $@
clean:
    rm -f *.o my_program
```

9. This makefile defines several targets:
   - all: the default target that builds the entire program
   - my_program: the final executable that links the two modules together
   - module1.o and module2.o: the object files for each module
   - clean: a target for cleaning up the directory by removing all object files and the executable
10. Use the make command to build the program:

make

This will compile each module separately and link them together into a final executable named "my_program".

11. Run the program:
    ./my_program

The output should be:

Copy code
Hello, world!
Goodbye, world!

Conclusion:

In conclusion, this assignment required us to use the gcc compiler to compile C programs using modular programming and the make command. By dividing the program into separate modules, we were able to break down the code into more manageable and reusable parts, making it easier to maintain and modify in the future. Using the make command, we automated the building process, making it faster and more efficient.

Overall, this assignment provided a good introduction to modular programming and the make command, which are essential tools for any software developer. By learning how to split a program into separate modules and use make to automate the building process, we can write more efficient and maintainable code, saving time and reducing the risk of errors.

Practical No -02

Problem Statement:

The objective of this assignment is to find a procedure to transfer files from one virtual machine to another virtual machine.

Theory: Git is a **distributed version control system (DVCS)** that allows multiple people to collaborate on projects by tracking changes, managing versions, and enabling easy rollbacks.

Git is an open-source version control system that helps developers:

✅ Track changes in code over time.
✅ Work collaboratively on a project.
✅ Maintain different versions of files efficiently.
✅ Roll back changes if needed.

Git Command

git clone

Purpose:

- Used to copy a remote repository to a local machine.
- Creates a new directory with all repository files, history, and branches.

git clone <repository_url>

Example:

git clone https://github.com/user/repository.git

- This command will download the repository. It from GitHub and create a local copy.

2. **git commit**

Purpose:

- Saves changes to the local repository.
- Every commit represents a snapshot of the project at a specific point in time.

Syntax:

git add .
git commit -m "Descriptive commit message"

Example:

git commit -m "Added a new login feature"

- git add . stages all modified files.
- git commit -m records the changes with a message.

---

3. git push

Purpose:

- Sends local commits to a remote repository.
- Ensures that the latest changes are available for other team members.

Syntax:

git push origin <branch_name>

Example:

git push origin main

- Pushes changes from the local main branch to the remote repository.

4. **git fetch**

Purpose:

- Retrieves updates from a remote repository but does not merge them.
- Allows users to check for changes before merging.

git fetch

Example:

git fetch origin

- Fetches the latest updates from the remote repository named origin.
- Local branches remain unchanged until merged.

5**. git pull**

Purpose:

- Fetches and merges changes from a remote repository into the current branch.
- Ensures the local branch is up to date with the remote repository.

Syntax:

git pull origin <branch_name>

Example:

git pull origin main

- Retrieves the latest changes from main and merges them into the local branch.

6. git checkout

Purpose:

- Switches between branches or restores previous versions of files.
- Can be used to create and switch to a new branch.

Syntax:

git checkout <branch_name>

Example:

git checkout develop

- Switches to the develop branch.

Create and switch to a new branch:

git checkout -b <new_branch_name>

Example:

git checkout -b feature-login

- Creates a new branch called feature-login and switches to it.

7. git reset

Purpose:

- Used to undo changes in the local repository.
- Can reset commits, unstage files, or discard changes permanently.

Types of Reset:

- Soft Reset (Keeps changes staged)

  git reset --soft HEAD~1

  - Moves the last commit to staging without deleting changes.
- Mixed Reset (Unstages changes but keeps them in working directory)

  git reset --mixed HEAD~1

    - Moves the last commit to working directory but unstages the changes.
- Hard Reset (Deletes changes permanently)

  git reset --hard HEAD~1

    - Completely removes the last commit and its changes.

8. Deleting a Repository

Purpose:

- Removes a Git repository locally or remotely.

Delete a Local Repository:

rm -rf <repository_name>

- This will permanently delete the repository folder and its contents.

Delete a Remote Repository (GitHub CLI Example):

gh repo delete <repository_name>

- This command deletes a repository from GitHub.
- Can also delete manually via GitHub/GitLab settings.

Conclusion

These Git commands are essential for working with repositories effectively. Whether you're cloning, committing, pushing, pulling, or resetting changes, understanding these commands will help you manage your code efficiently

Practical No 03

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

Theory:

1.1 What is Virtualisation?

Virtualization is the process of running a virtual instance of a computer system in a layer abstracted from the actual hardware. Most commonly, it refers to running multiple operating systems on a computer system simultaneously. To the applications running on top of the virtualized machine, it can appear as if they are on their own dedicated machine, where the operating system, libraries, and other programs are unique to the guest virtualized system and unconnected to the host operating system which sits below it.

1.2 Why is it important in computing?

There are many reasons why people utilize virtualization in computing. To desktop users, the most common use is to be able to run applications meant for a different operating system without having to switch computers or reboot into a different system. For administrators of servers, virtualization also offers the ability to run different operating systems, but perhaps, more importantly, it offers a way to segment a large system into many smaller parts, allowing the server to be used more efficiently by a number of different users or applications with different needs. It also allows for isolation, keeping programs running inside of a virtual machine safe from the processes taking place in another virtual machine on the same host.

1.3 Difference between Virtual machines and containers

Virtual machines are like containers. While both containers and virtual machines allow for running applications in an isolated environment, allowing you to stack many onto the same machine as if they are separate computers, containers are not full, independent machines. A container is actually just an isolated process that shared the same Linux kernel as the host operating system, as well as the libraries and other files needed for the execution of the program running inside of the container, often with a network interface such that the container can be exposed to the world in the same way as a virtual machine. Typically, containers are designed to run a single program, as opposed to emulating a full multi-purpose server.

1.4 What is Hypervisor?

A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

1.5 Types of Hypervisor:

There are two main hypervisor types, referred to as “Type 1” (or “bare metal”) and

“Type 2” (or “hosted”).

1.6 Type1 & Type 2 Hypervisor

A type 1 hypervisor acts like a lightweight operating system and runs directly on the host’s hardware, while a type 2 hypervisor runs as a software layer on an operating system, like other computer programs. The most commonly deployed type of hypervisor is the type 1 or bare-metal hypervisor, where virtualization software is installed directly on the hardware where the operating system is normally installed. Because bare-metal hypervisors are isolated from the attack-prone operating system, they are extremely secure. In addition, they generally perform better and more efficiently than hosted hypervisors. For these reasons, most enterprise companies choose bare-metal hypervisors for data center computing needs. While bare-metal hypervisors run directly on the computing hardware, hosted hypervisors run on top of the operating system (OS) of the host machine. Although hosted hypervisors run within the OS, additional (and different) operating systems can be installed on top of the hypervisor. The downside of hosted hypervisors is that latency is higher than bare-metal hypervisors. This is because communication between the hardware and the hypervisor must pass through the extra layer of the OS. Hosted hypervisors are sometimes known as client hypervisors because they are most often used with end users and software testing, where higher latency is less of a concern.

1.7 Cloud hypervisor

As cloud computing becomes pervasive, the hypervisor has emerged as an invaluable tool for running virtual machines and driving innovation in a cloud environment. Since a hypervisor is a software layer that enables one host computer to simultaneously support multiple VMs, hypervisors are a key element of the technology that makes cloud computing possible. Hypervisors make cloud-based applications available to users across a virtual environment while still enabling IT to maintain control over a cloud environment’s infrastructure, applications and sensitive data.

1.8 Working of Hypervisor

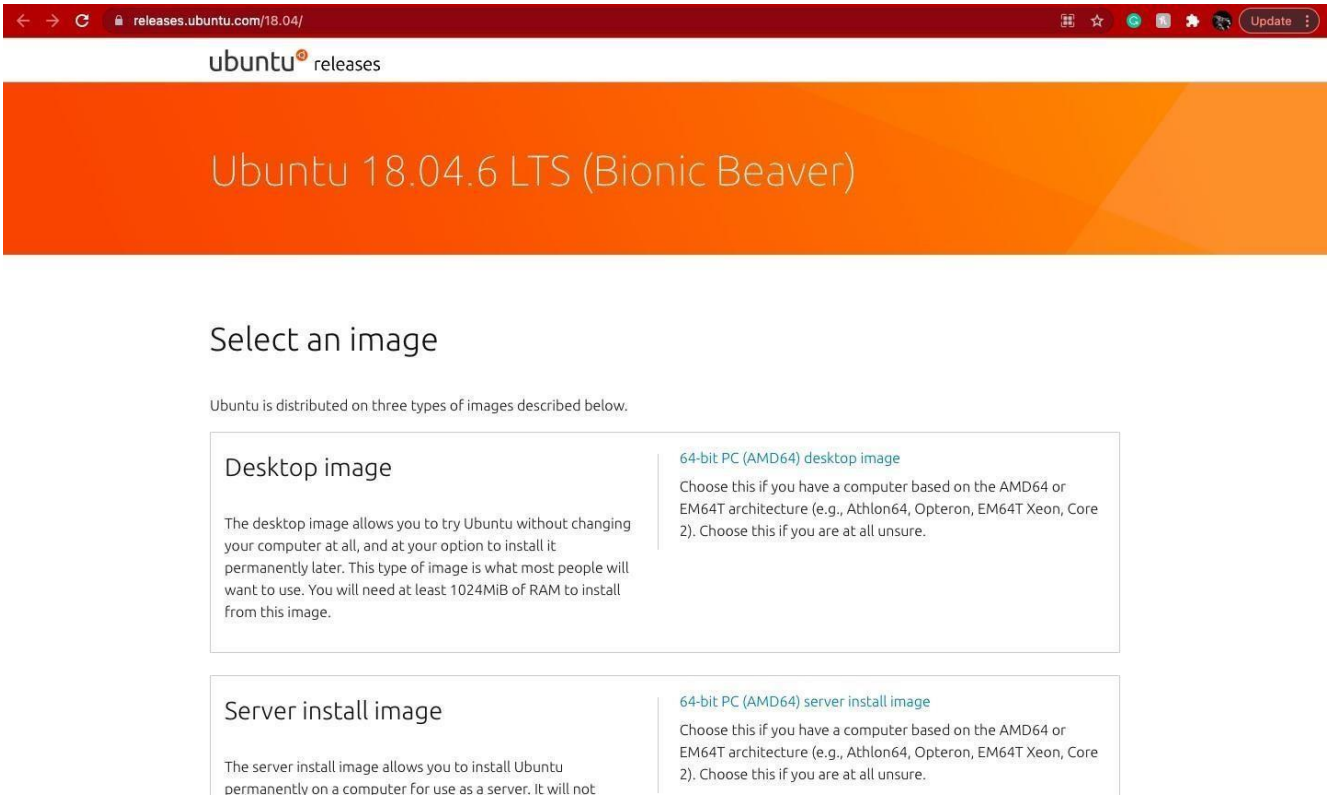
Hypervisors support the creation and management of virtual machines (VMs) by abstracting a computer’s software from its hardware. Hypervisors make virtualization possible by translating requests between the physical and virtual resources. Bare-metal hypervisors are sometimes embedded into the firmware at the same level as the motherboard basic input/output system (BIOS) to enable the operating system on a computer to access and use virtualization software.

Installation Steps:

Step 1: Install ubuntu ISO

Install Ubuntu ISO of your choice of version. Click on link provided in right block of

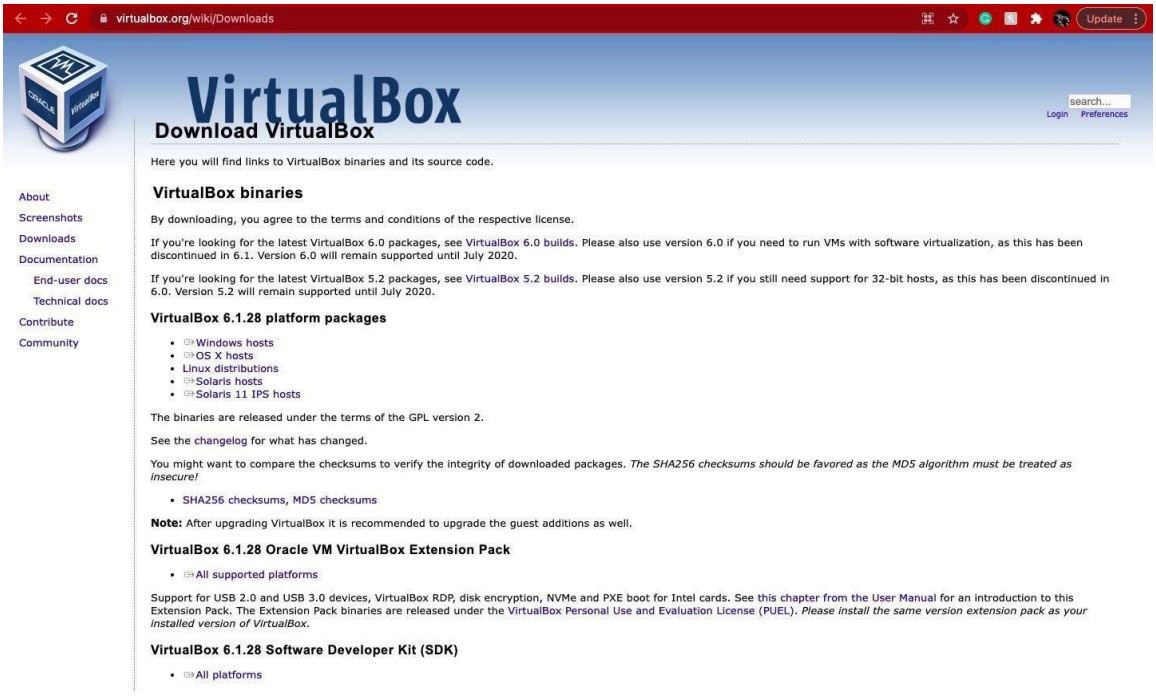
desktop image. Link: https://releases.ubuntu.com/18.04/



Step 2: Install Virtual machine.

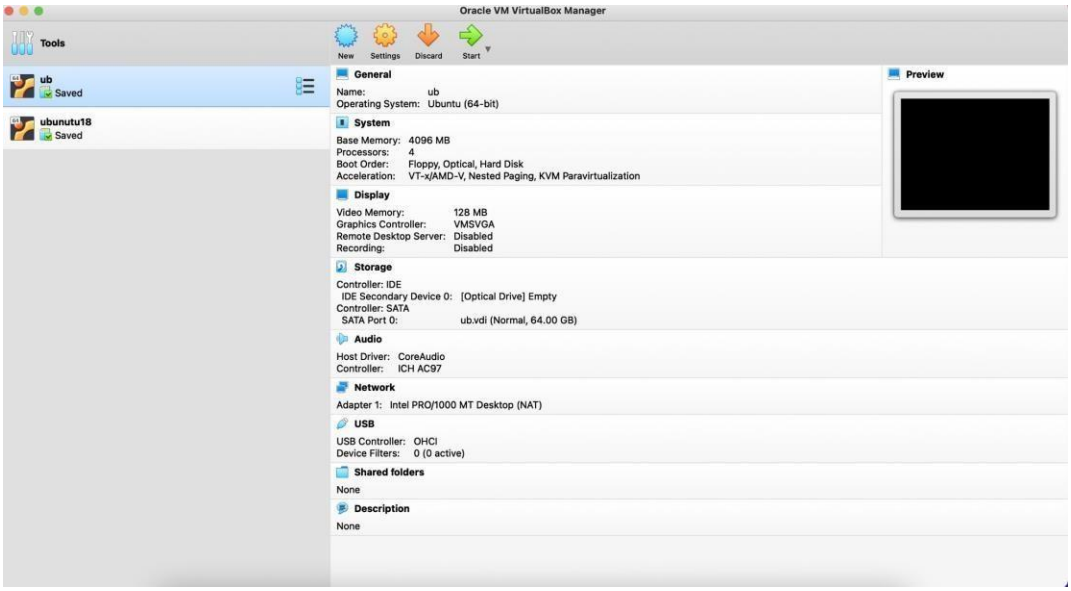
Go to VirtualBox website to download the binary for your current operating system. When download is finished, run the executable file. Continue with the installation of VirtualBox with the defaults. This will open VirtualBox at the end of the installation.

Link : https://www.virtualbox.org/wiki/Downloads



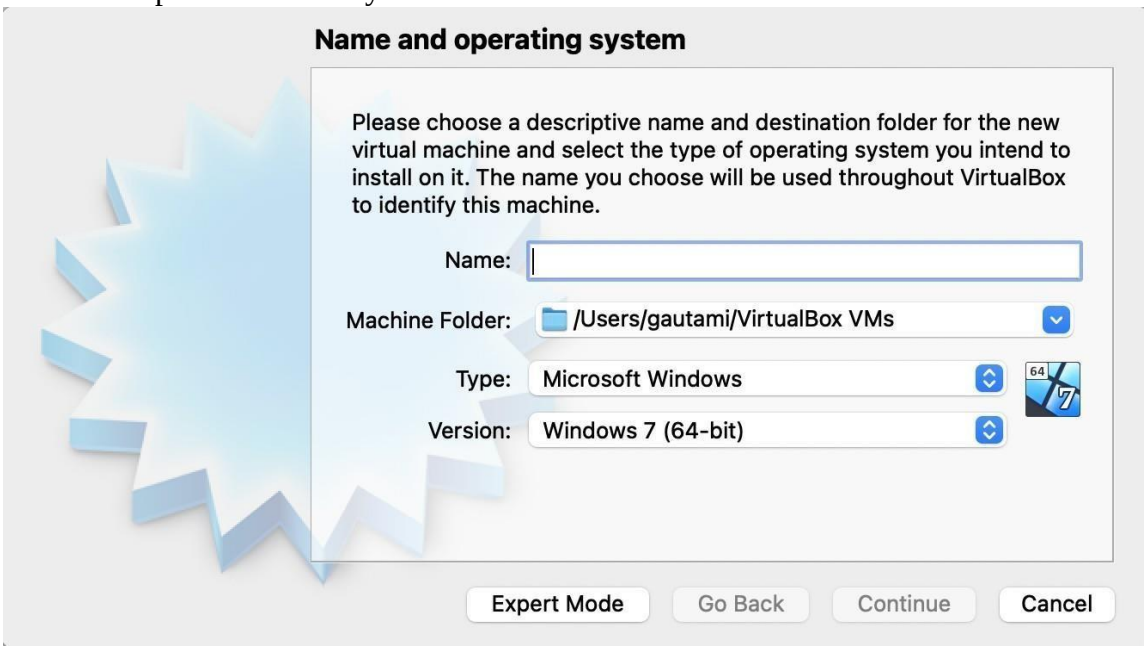
Step 3: Create Virtual Machine

Click 'New' button to open a dialog



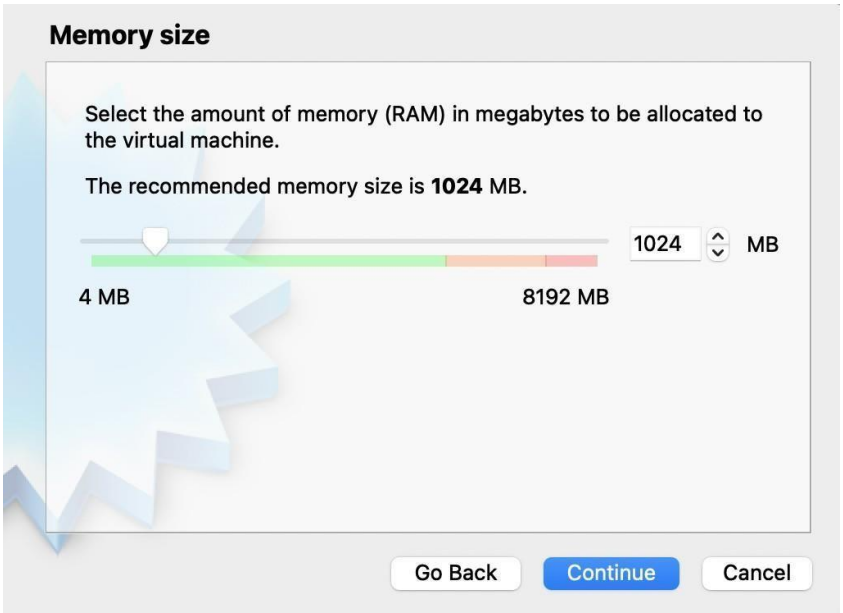
Type a name for the new virtual machine.

Note that VirtualBox automatically changes 'Type' to Linux and 'Version' to 'Ubuntu (64 bit)'. These two options are exactly what we need

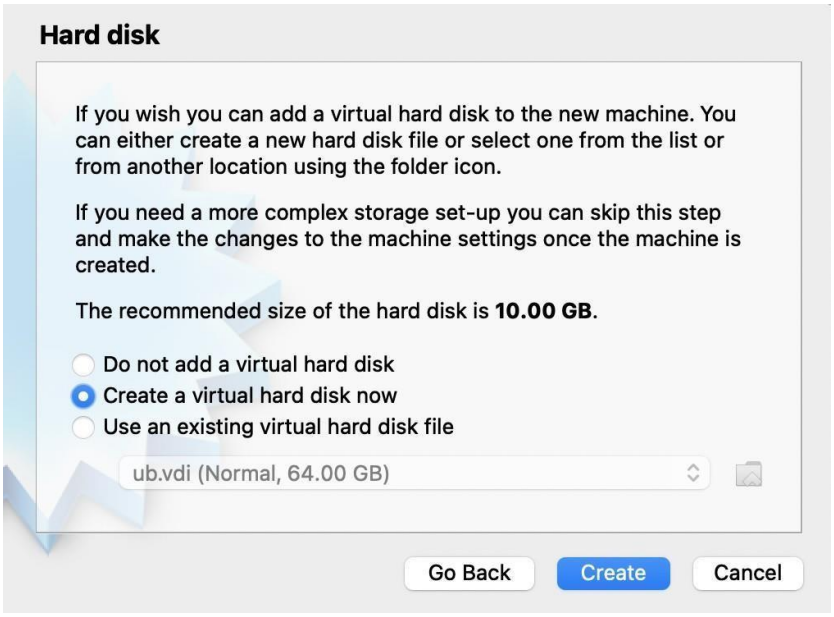


The memory size depends on your host machine memory size.

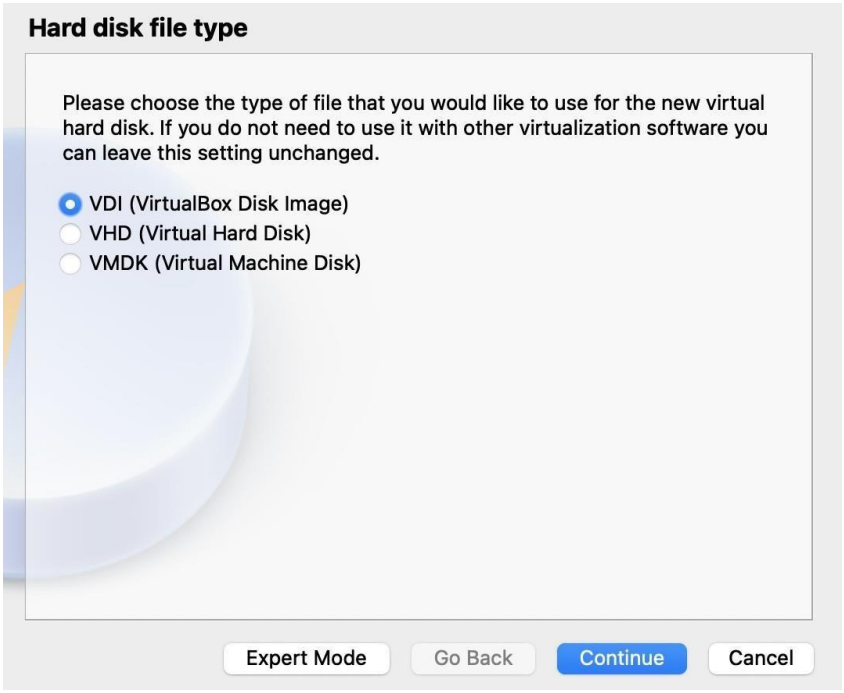
Note that VirtualBox will create a swap partition with the same amount of space as base memory you have entered here. So later when you are selecting the size of the virtual hard drive, make sure it is large enough since the hard drive will be splitted into root (/)and swap partitions. The root partition contains by default all your system files, program settings and documents.



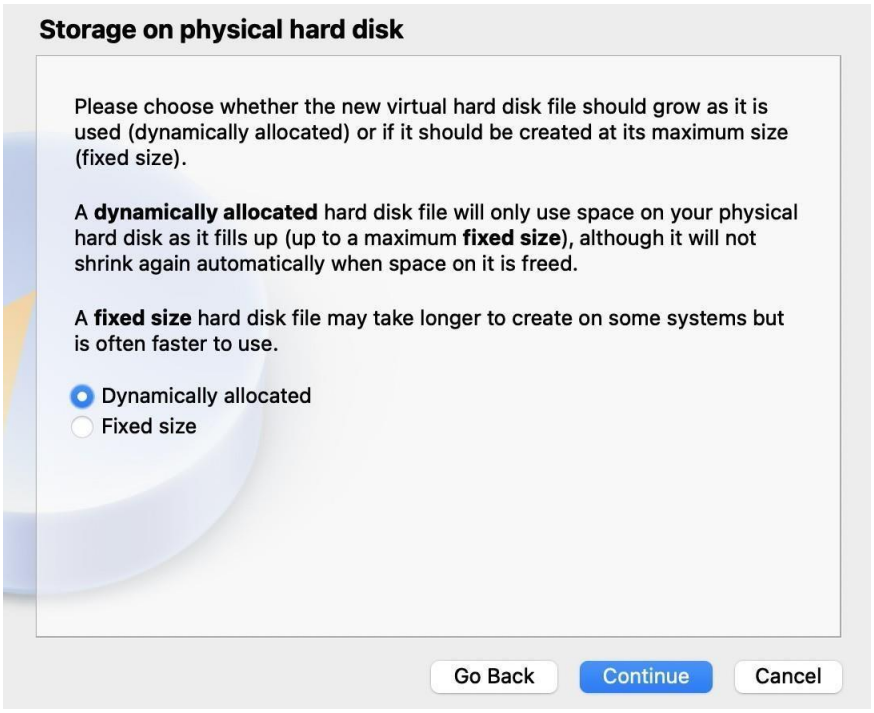
Accept the default 'Create a virtual hard drive now' and click 'Create' button.



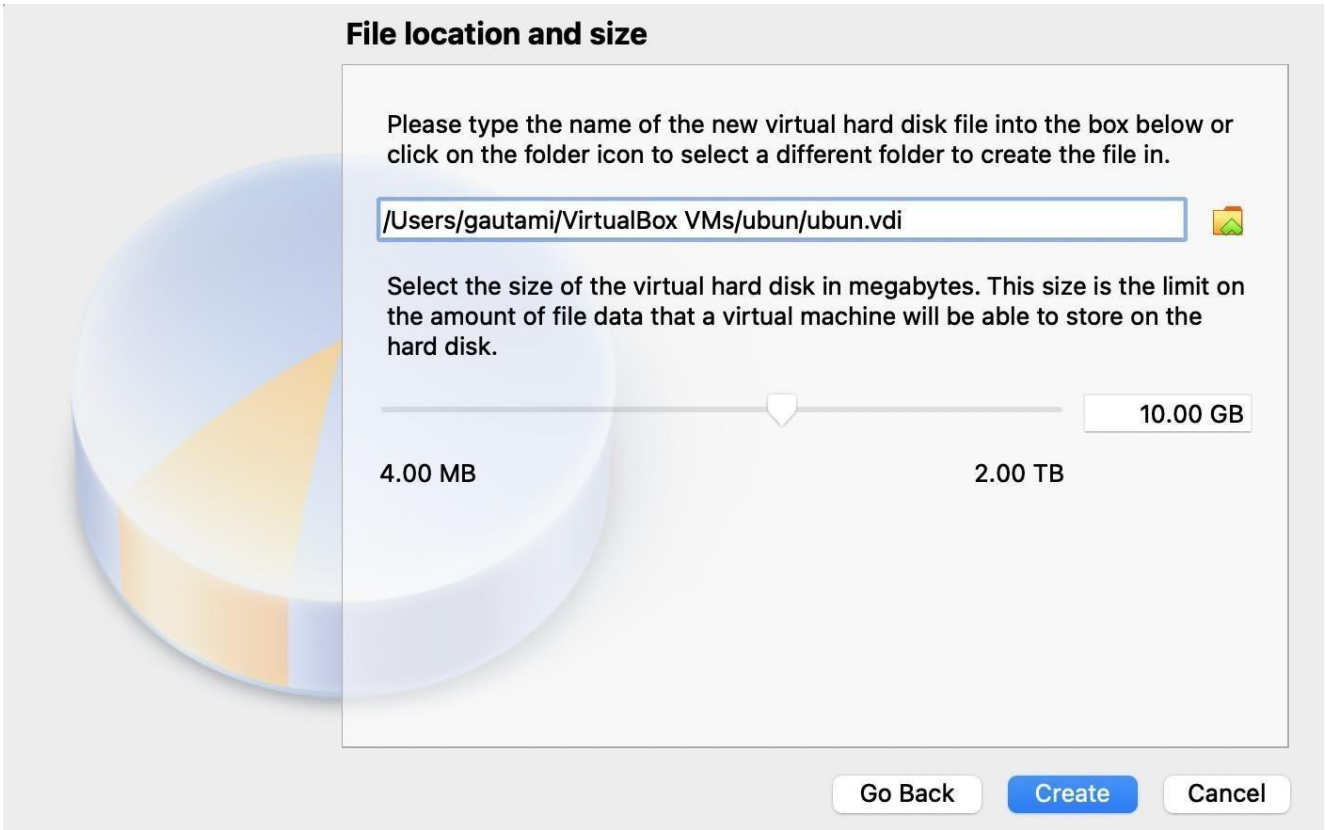
Continue to accept the default 'VDI' drive file type and click 'Next' button.



Continue to accept the default or you can change the storage type from the default 'Dynamically allocated' to 'Fixed size' to increase performance.

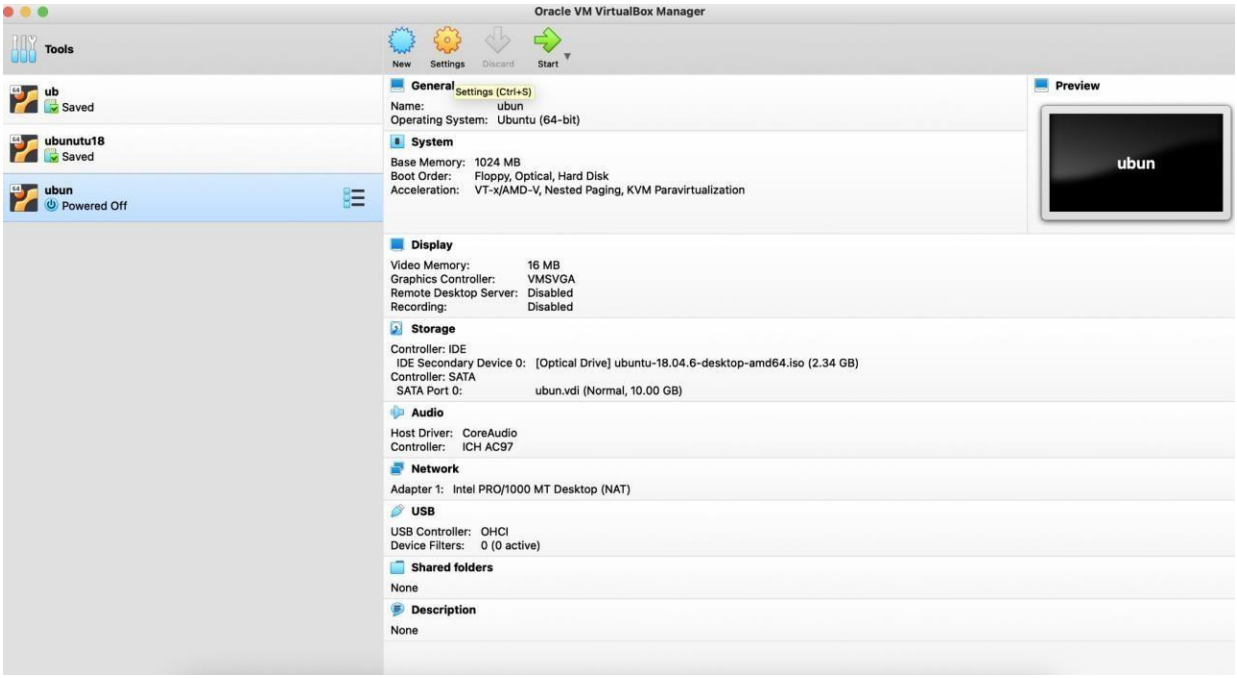


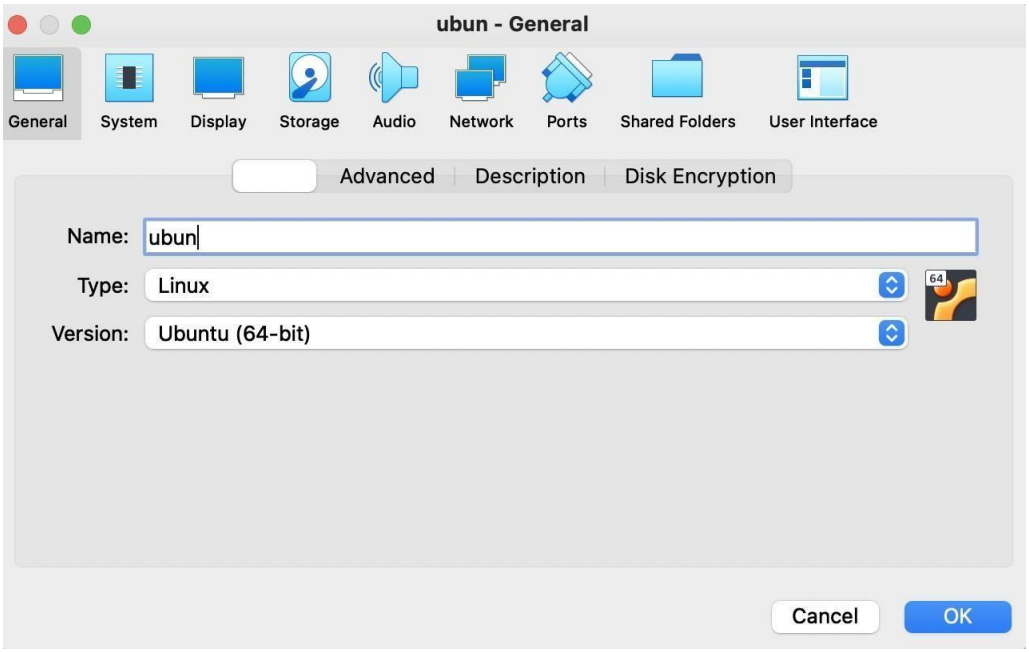
Select the size of the virtual hard disk and then click on create button.



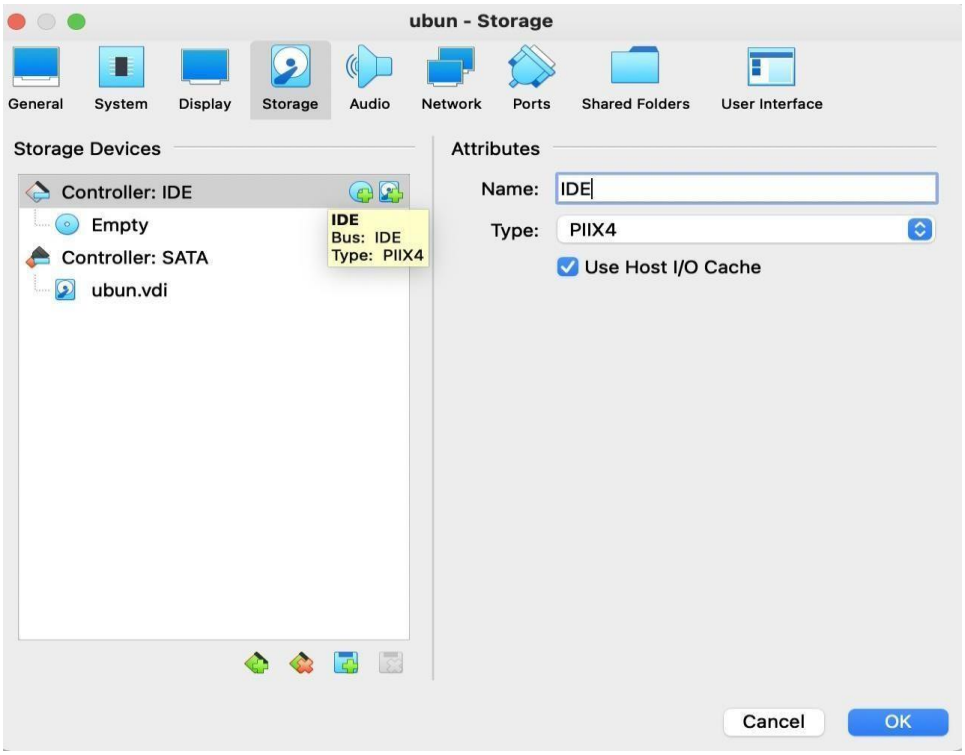
Now a virtual machine is created but we need to add an iso file to it.

Click on settings.

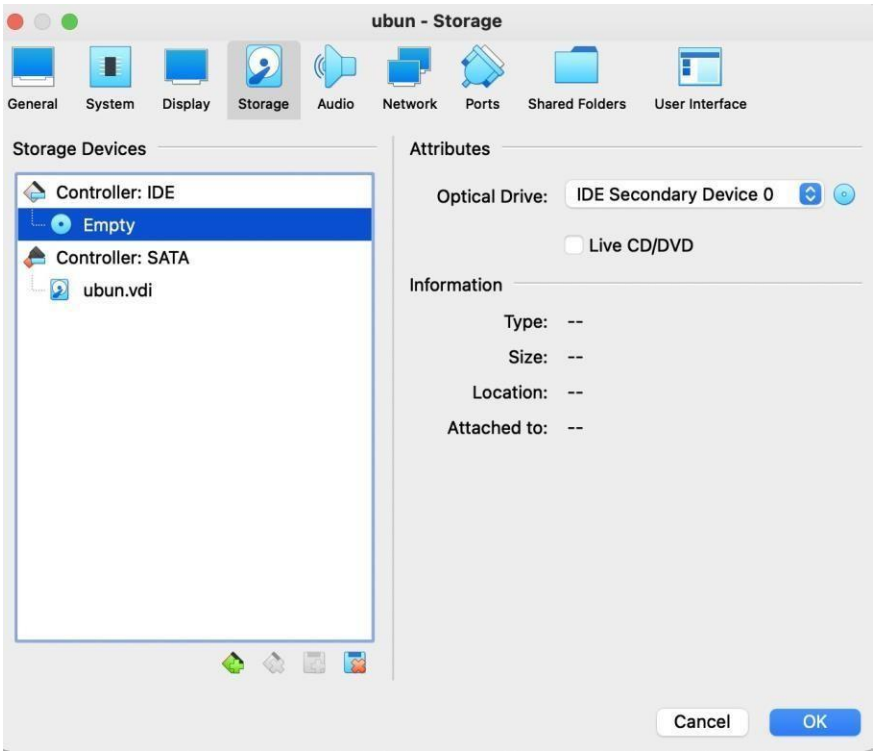




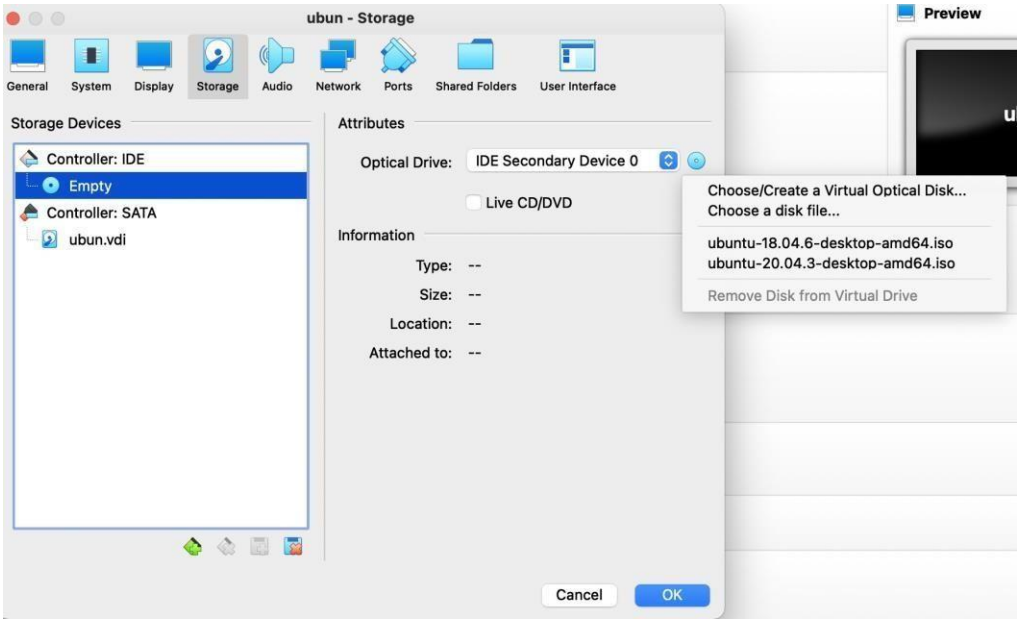
Click on storage



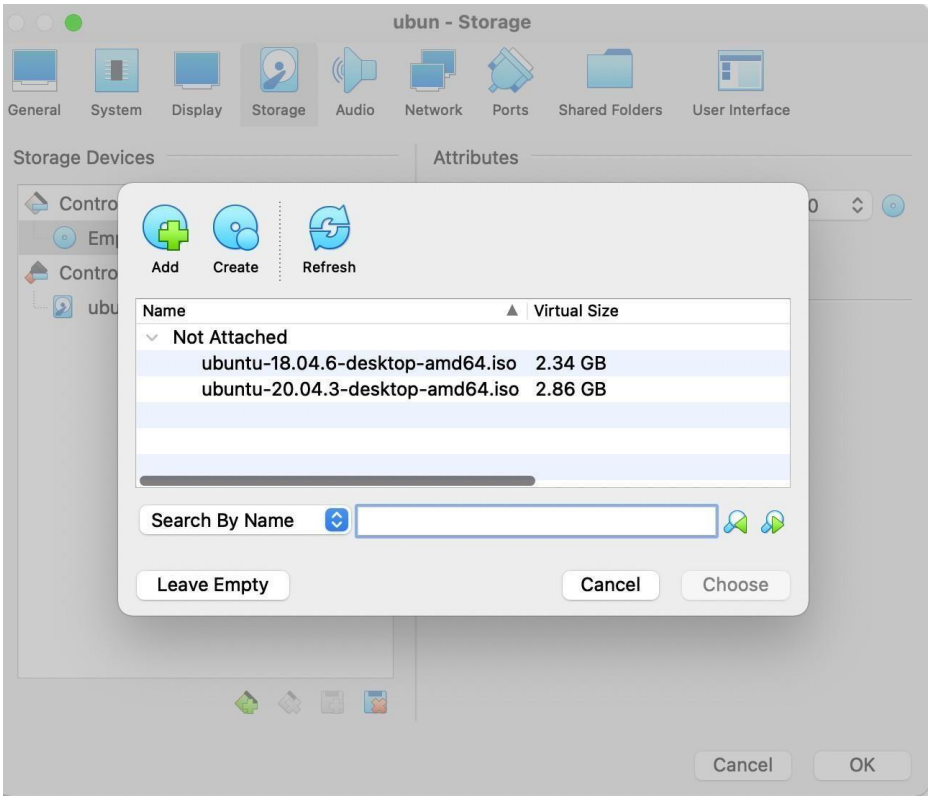
Now, click on empty.



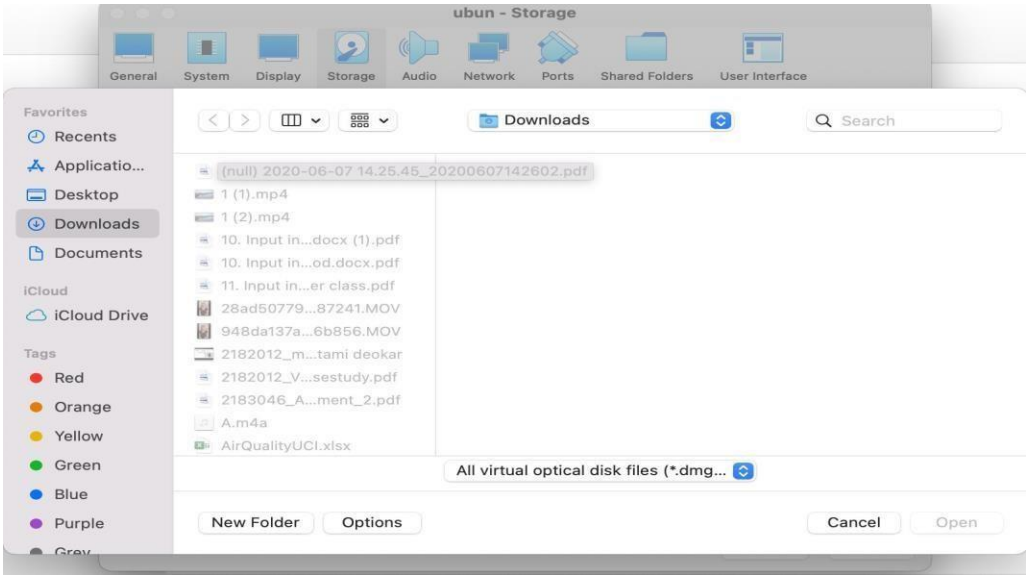
CLick on right small blue circle near optical drive



Click on choose/create a virtual optical disk



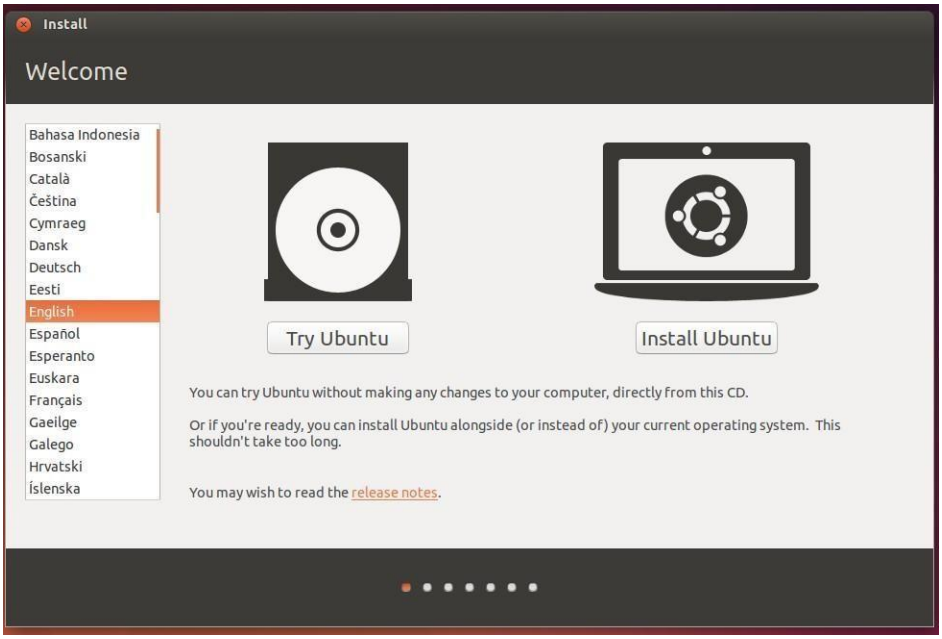
Now as you can see some of ubuntu iso are already attached, but if you are creating it for newly time. You will require to attach iso file
Click on add



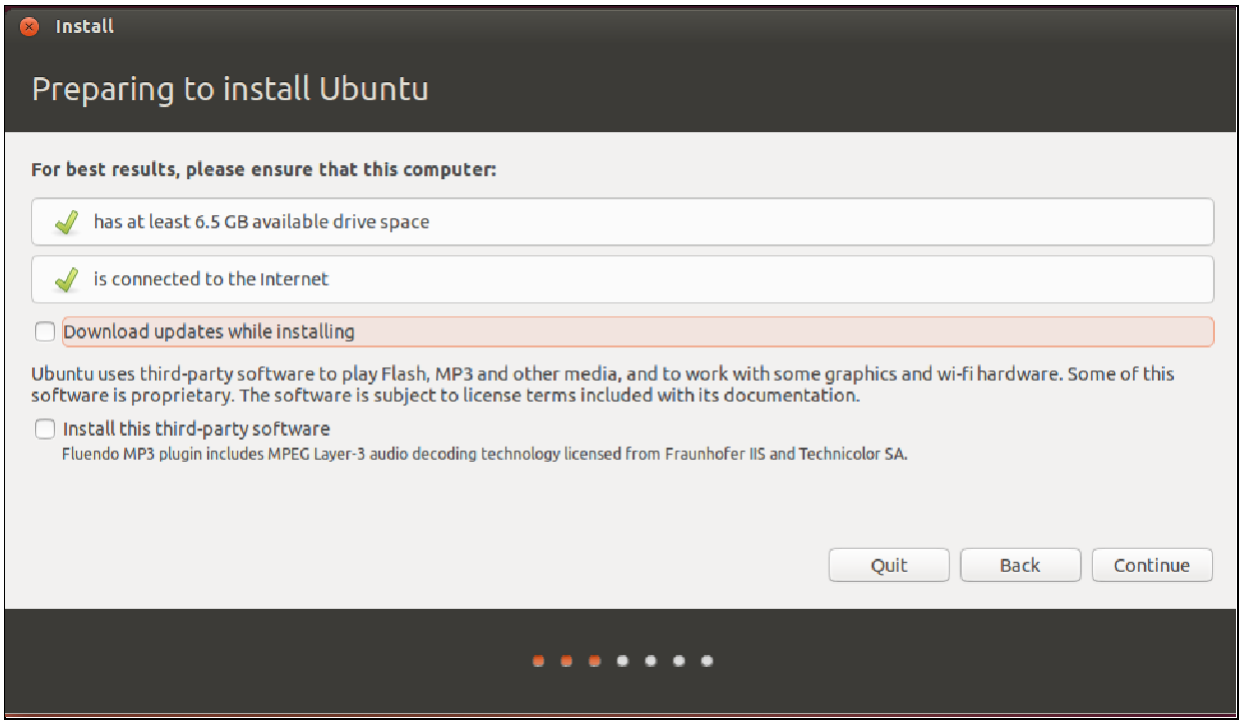
Select your iso file and click on open. After getting attach , select iso file and it will be mounted.

Step 4: Install Ubuntu

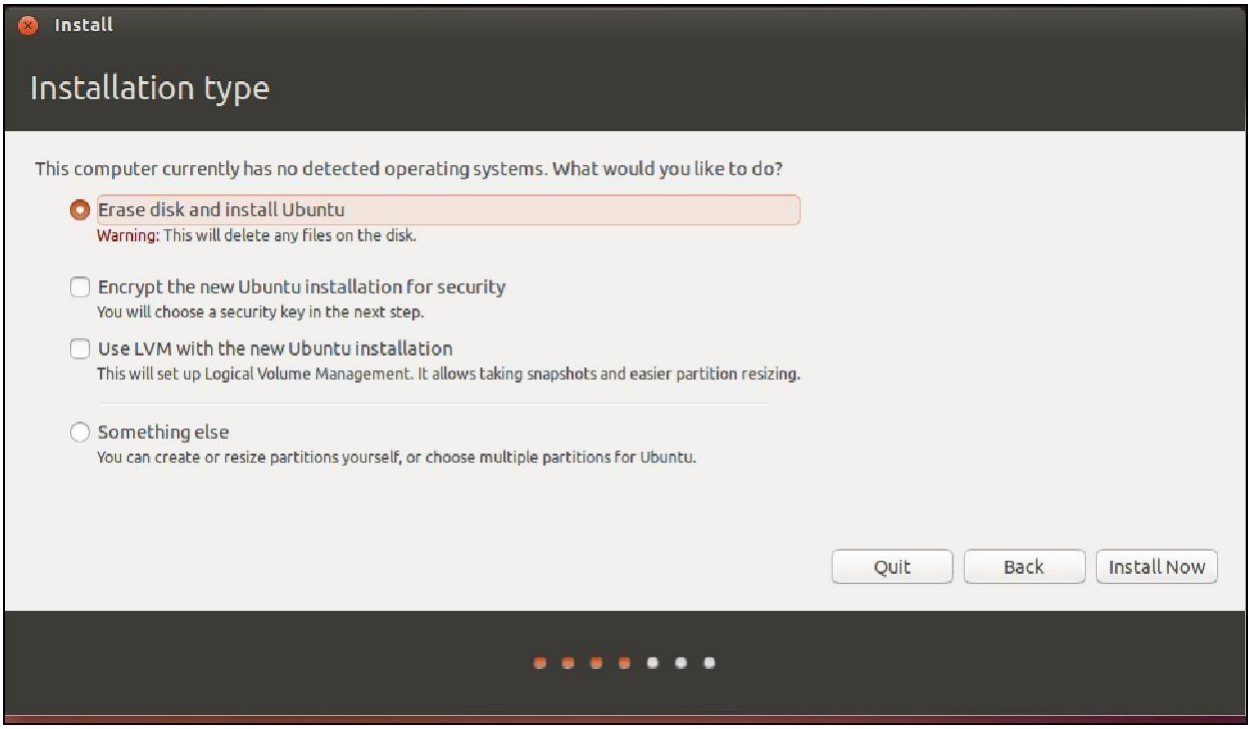
Back to Oracle VM VirtualBox Manager, click on the new Ubuntu virtual machine created and hit 'Start' button. Now you shall see a 'Welcome' screen. Click 'Install Ubuntu' button.



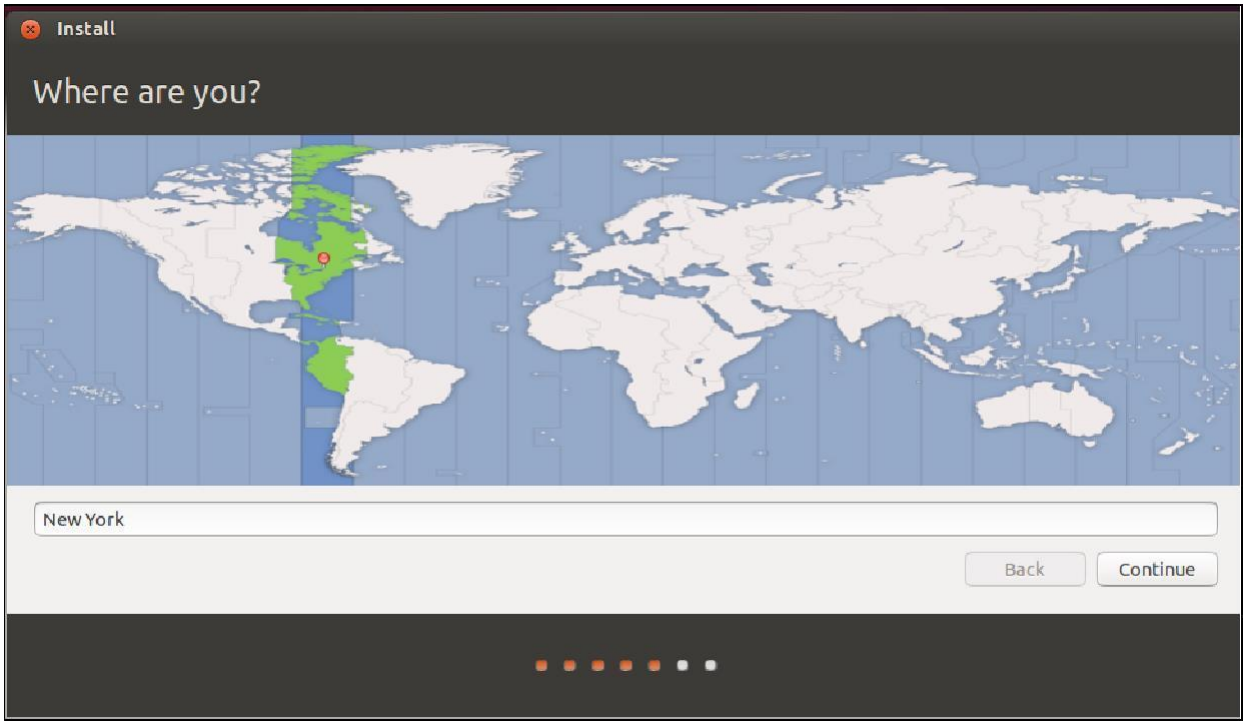
Click 'Continue' button.

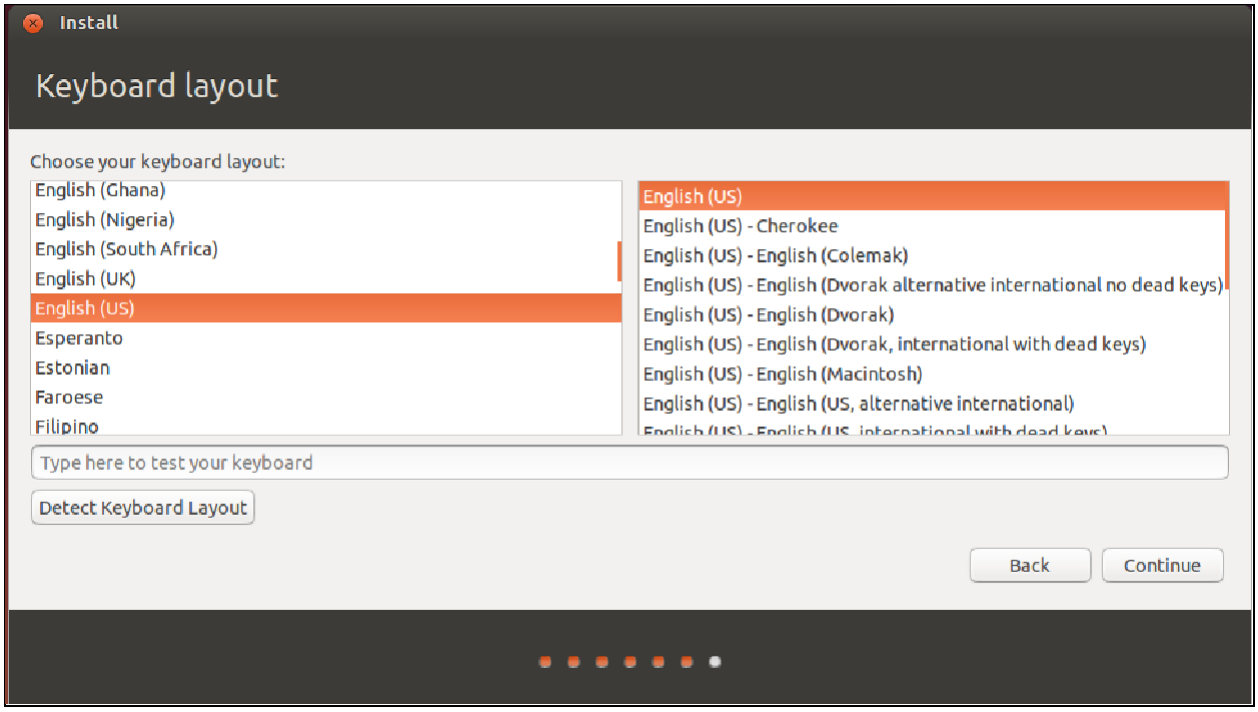


Make sure 'Erase disk and install Ubuntu' option is selected and click 'Install Now' button.



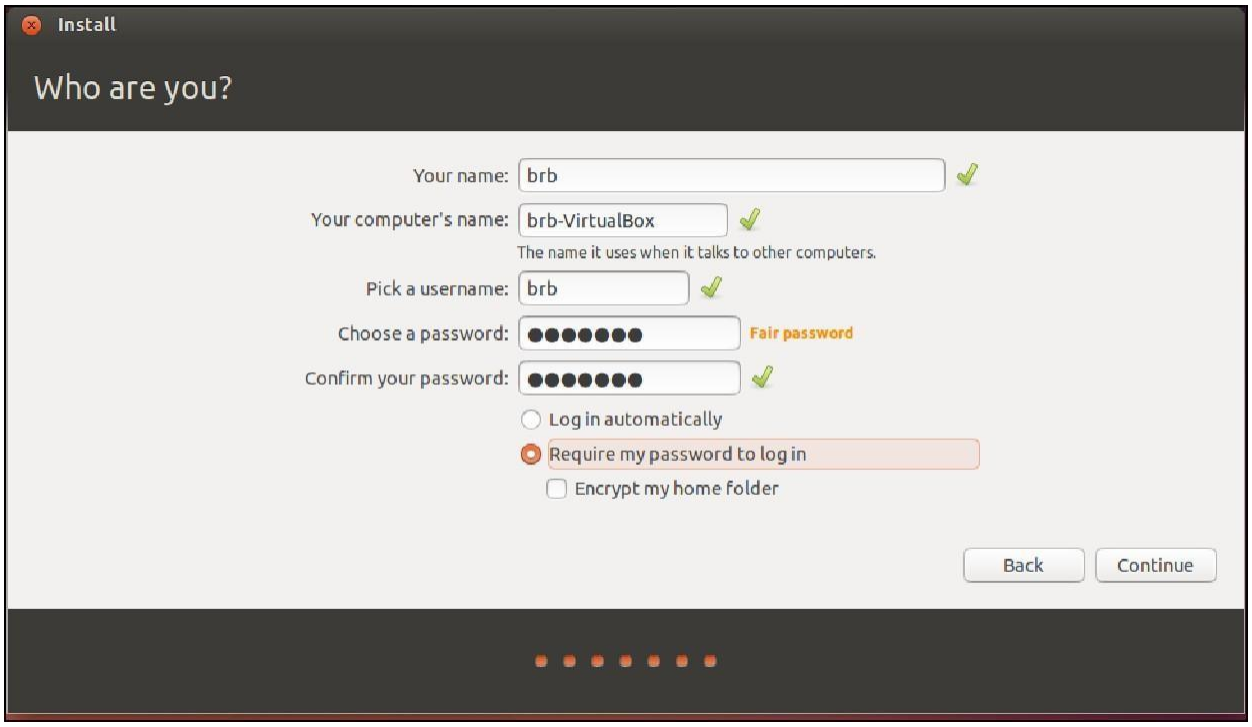
Ubuntu will ask you a few questions. If the default is good, click 'Continue' button.





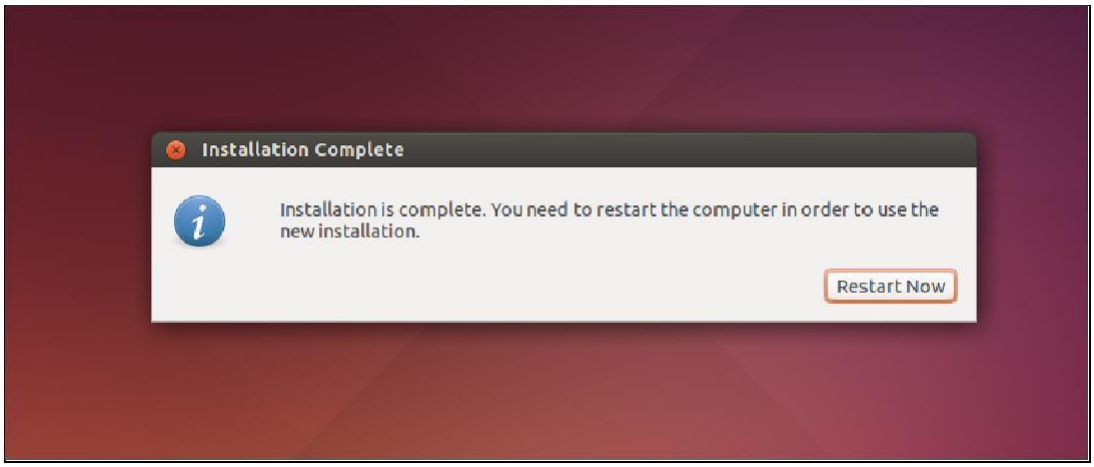
In 'Who are you?' dialog, enter your preferred name, username and password.

Note that this user will have root/sudo privilege. Click 'Continue' button.



The installation will continue until it is finished.

After installation is complete, click 'Restart Now' button. When you see a screen with a black background saying 'Please remove installation media and close the tray (if any) then press ENTER:', just follow it.

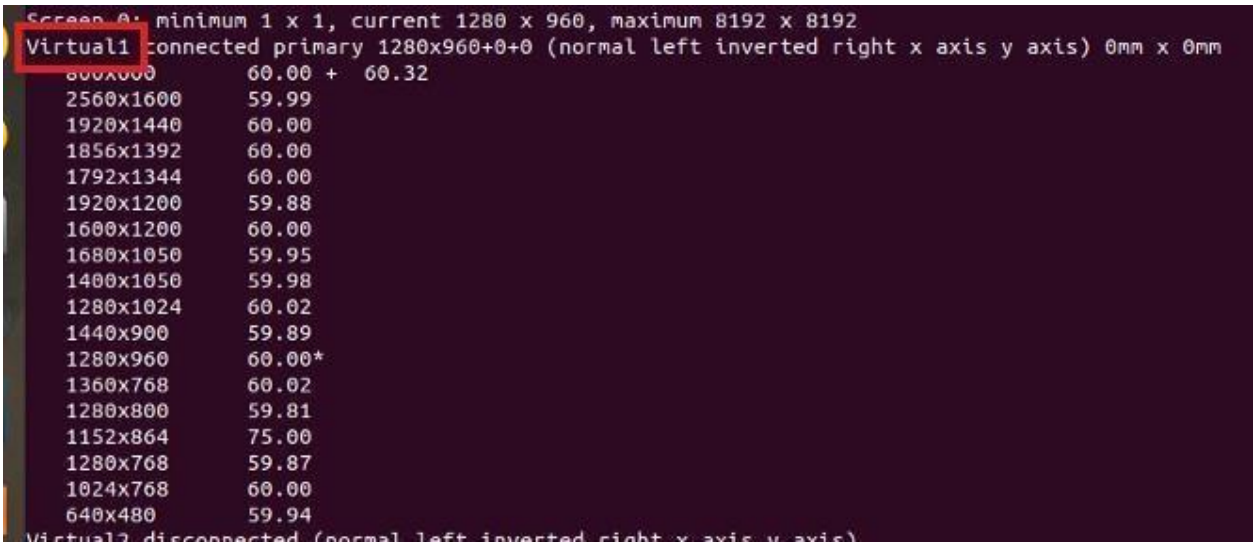


You will be redirected to the login page. Enter the password you have chosen and press 'Enter'.

The Ubuntu Desktop OS is ready. You may find the desktop screen is too small. Don't worry. You can solve this easily with changing resolutions. You can use following command in terminal

Command: xrandr

After putting command you can see couple of resolution



Star mark will be your current resolution. To change resolution put command

Command: sudo xrandr --addmode Virtual1 1360x768 //change resolution as needed

Congratulations! You are all done.

Conlcusion: The problem statement was to install VirtualBox or VMware Workstation on Windows 7 or 8 and set up different flavors of Linux or Windows operating systems as virtual machines.
After completing this task, we can conclude that virtualization software like VirtualBox and VMware Workstation provide an excellent way to run multiple operating systems on a single computer. By using virtual machines, we can create isolated environments for testing, development, or other purposes without affecting the host operating system.

In addition, setting up virtual machines with different operating systems is a great way to learn and explore new technologies without having to purchase additional hardware. With virtualization software, we can easily create and configure virtual machines with different specifications, such as RAM, CPU, and storage, to meet our specific needs.

Overall, the installation and setup of VirtualBox or VMware Workstation with different flavors of Linux or Windows operating systems on top of Windows 7 or 8 was successful and allowed us to create a versatile and flexible computing environment.

Practical No -04

Problem Statement: Install a C compiler in the virtual machine created using virtual box and execute Simple Programs.

Theory:

VirtualBox is a popular virtualization software that allows us to run multiple operating systems on a single computer. To install a C compiler in a virtual machine, we need to first start the virtual machine and log in to the operating system. Then, we can download and install the C compiler of our choice, such as GCC or Clang. Once the compiler is installed, we can write simple C programs and compile them using the C compiler. If there are no errors, we can execute the compiled program and view the output.

Virtualization technology has revolutionized the way computing resources are utilized in modern computing environments. Virtualization allows the creation of multiple virtual machines (VMs) on a single physical machine, enabling each virtual machine to have its own operating system and resources. This technology is particularly useful for developers, system administrators, and software engineers who need to test software applications, experiment with different configurations, and isolate environments.

VirtualBox is a popular virtualization software that allows us to run multiple operating systems on a single computer. It is an open-source software package that can be used on Windows, macOS, Linux, and Solaris hosts. It allows us to create multiple virtual machines, each with its own virtual hardware, including CPUs, memory, storage, and network interfaces.

Installing a C compiler in a virtual machine is a common task for many developers and software engineers. A C compiler is a program that translates C source code into executable machine code that can be run on a computer. GCC (GNU Compiler Collection) is a popular open-source C compiler that is available on most Linux distributions. Clang is another popular C compiler that is known for its speed and modern features.

To install a C compiler in a virtual machine created using VirtualBox, we first need to start the virtual machine and log in to the operating system. We can then download and install the C compiler of our choice. For example, to install GCC on a Ubuntu virtual machine, we can use the following command:

```
sudo apt-get install build-essential
```

This command installs the necessary packages for building software, including GCC.

Once the C compiler is installed, we can write C programs using a text editor or integrated development environment (IDE) and compile them using the C compiler. For example, to compile a simple "Hello, World!" program using GCC, we can use the following command:

```
gcc hello.c -o hello
```

This command compiles the program and generates an executable file called "hello". We can then execute the program using the following command:

./hello

This command runs the "hello" executable and displays the output to the console.

Using a virtual machine to install and use a C compiler provides several advantages. First, it allows us to experiment with different versions of the C compiler without affecting the host operating system. We can also create multiple virtual machines with different operating systems and configurations to test software applications in different environments. Additionally, virtual machines provide a level of isolation and security, as any changes made in the virtual machine do not affect the host system.

In conclusion, virtualization technology and VirtualBox provide an efficient and effective way to install and use a C compiler in a virtual machine. By following the necessary steps to install the C compiler and compile a simple C program, we can gain valuable experience with C programming concepts and virtualization technology. The ability to create and use virtual machines provides developers and software engineers with a flexible and powerful tool for testing, experimentation, and software development.

Execution:
Here is an example of a simple C program that prints "Hello, World!" to the console:

```
#include <stdio.h>

int main() {
   printf("Hello, World!\n");
   return 0;
}
```

To compile and execute this program using the GCC compiler, we can follow these steps:

- Open a terminal window in the virtual machine.
- Navigate to the directory where the program is saved.
- Type "gcc hello.c" to compile the program.
- Type "./a.out" to execute the compiled program.
- The output should be "Hello, World!" printed to the console.

Conclusion:

In conclusion, installing a C compiler in a virtual machine created using VirtualBox allows us to practice and learn C programming without affecting the host operating system. By following the steps to install the compiler and compile a simple C program, we can verify that the compiler is working correctly and gain experience with basic C programming concepts. Overall, this assignment provides a good introduction to using virtual machines for programming and software development.

Practical No -05

Problem Statement:
The aim of this assignment is to install Google App Engine and develop simple web applications using Python/Java.

Theory:
Google App Engine (GAE) is a Platform-as-a-Service (PaaS) that enables developers to build and host web applications on Google's infrastructure. It supports several programming languages including Python, Java, Go, and Node.js. In this assignment, we will focus on developing web applications using Python/Java.

To get started with GAE, we need to follow the below steps:

Create a Google account
Install the Google Cloud SDK
Create a new GAE project
Choose the programming language (Python/Java)
Write the code for the web application
Deploy the web application on GAE
Here are the detailed steps for installing GAE and creating a web application:

Create a Google account:
To use GAE, we need to create a Google account. If we already have a Gmail account, we can use that to sign in to GAE.

Install the Google Cloud SDK:
The Google Cloud SDK is a set of tools for managing and deploying applications on GAE. It includes the command-line interface (CLI) tools, such as gcloud and gsutil, which we can use to manage GAE projects. We can download the SDK from the official Google Cloud website and install it on our local machine.

Create a new GAE project:
Once we have installed the Google Cloud SDK, we can create a new GAE project using the gcloud command-line tool. We need to provide a unique name for our project.

Choose the programming language (Python/Java):
GAE supports several programming languages, including Python and Java. We need to choose the programming language based on our preferences and expertise.

Write the code for the web application:
To create a web application on GAE, we need to write code in our chosen programming language. We can use any text editor or Integrated Development Environment (IDE) to write the code. We also need to include the GAE libraries in our project to access the GAE APIs.

Deploy the web application on GAE:

Once we have written the code for the web application, we can deploy it on GAE using the gcloud CLI tool. We need to provide the project name and the version of the application that we want to deploy.

Installation on Windows/Mac:

Step 1: Installing supported language

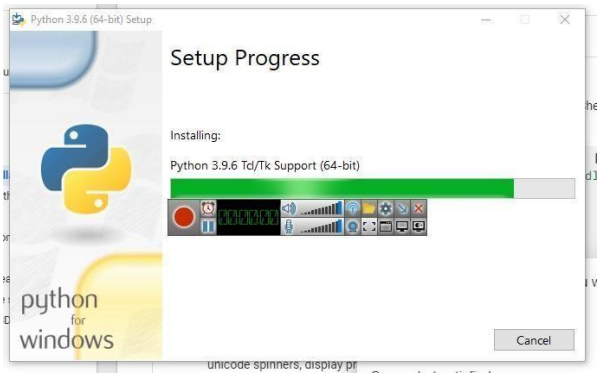
Install supported language. I have installed python here

according to my easeLink to download :

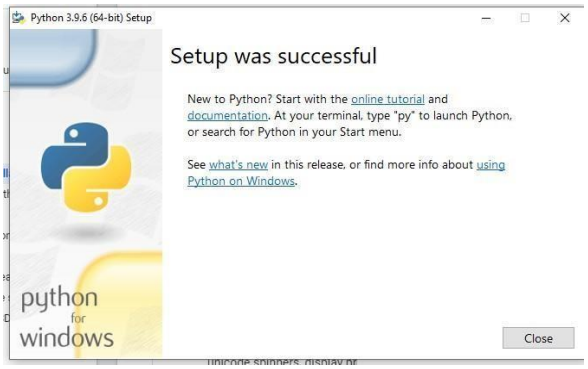
https://www.python.org/downloads/windows/

As I have download and trying to install, you can see setup has been in progress

Note: Just keep on clicking next , there is no require of any alteration to be made.

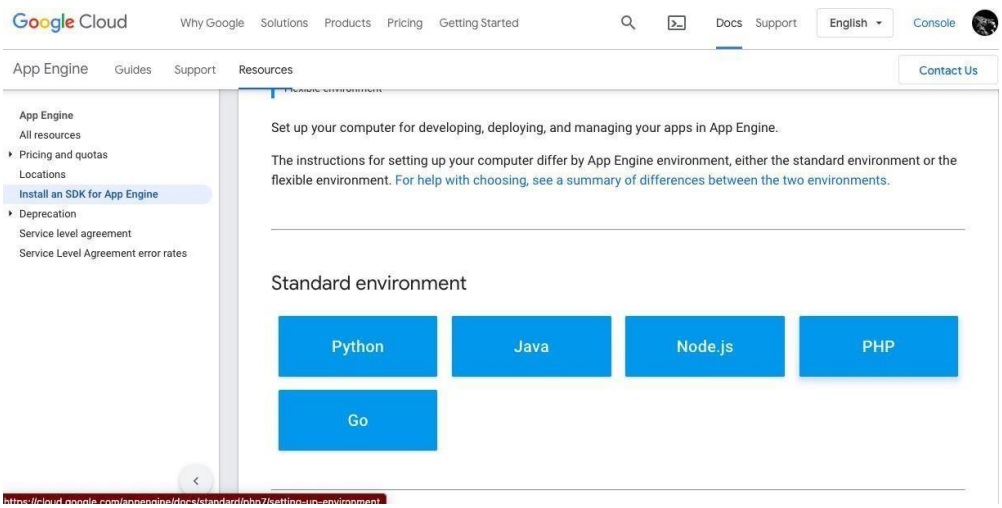


After successful setup, you will get a message as below:



Step 2: Installing google app engine

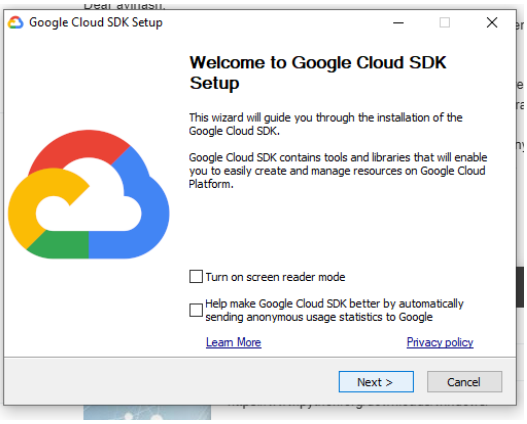
Link to download : https://cloud.google.com/appengine/downloads?csw=1



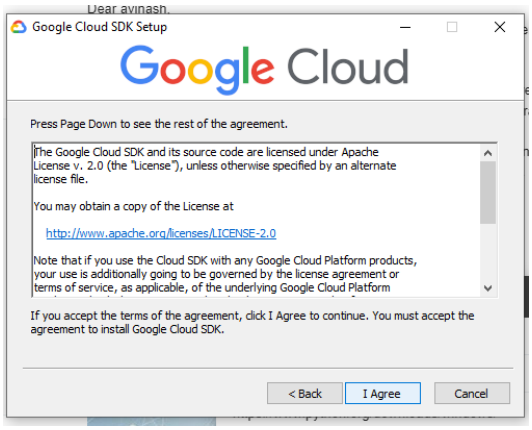
Note: For mac , please click in the mac section and download and follow the same steps.

Here you can choose your language environment to download gcl in that

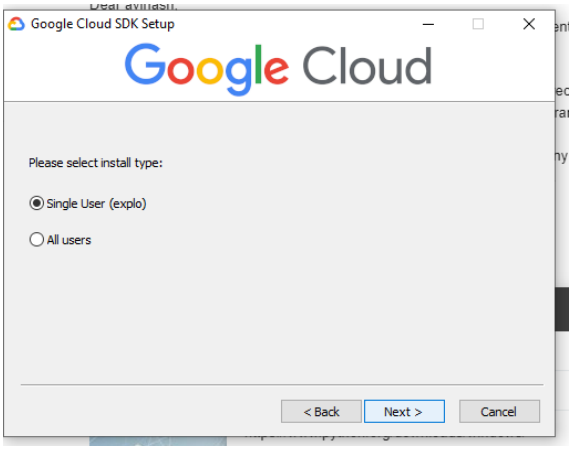
supported languageAs, I have chose python I will install my gcl in python,
On clicking python or (https://cloud.google.com/sdk/docs/install) it will my google cloud sdk

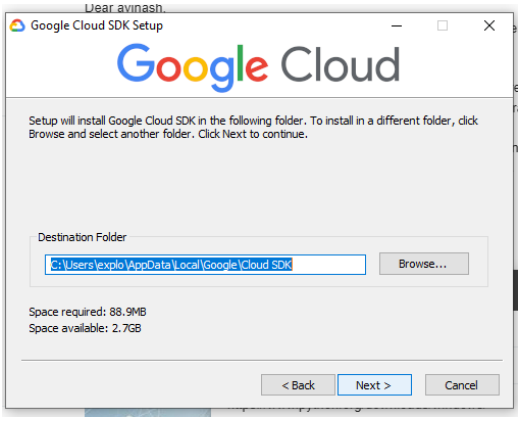


Note: Just keep on clicking next , there is no require of any alteration to be made.



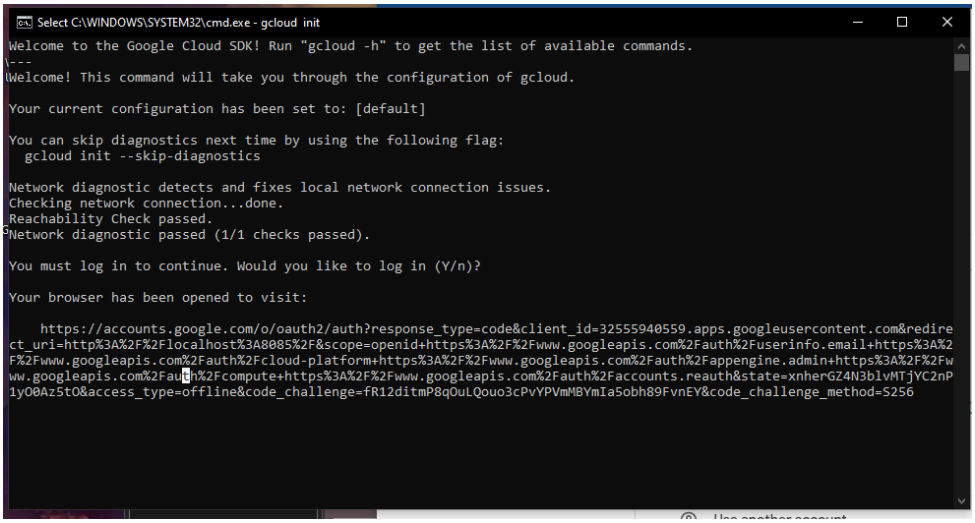
You need to accept the terms and conditions to move to next step





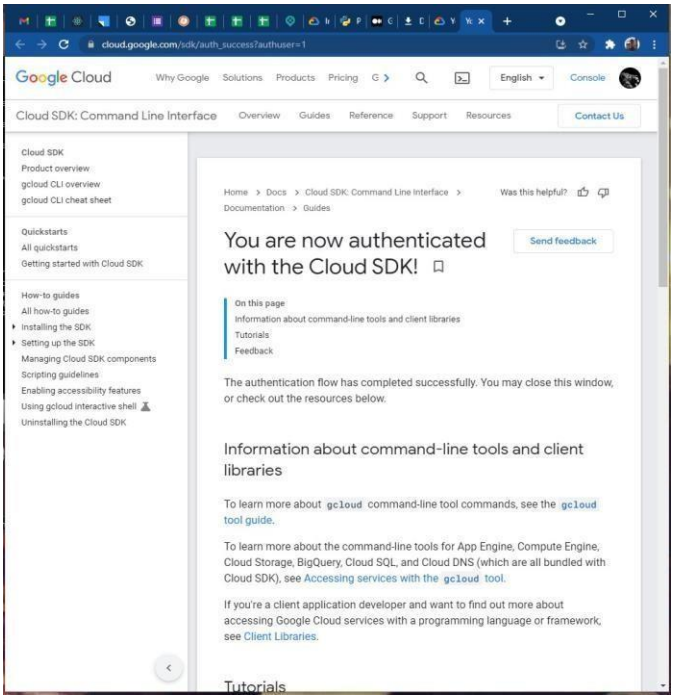
Now as my google cloud sdk is installed , I will try to access gcl for further projects.

Step 3: Accessing gcl (Google Command Line)



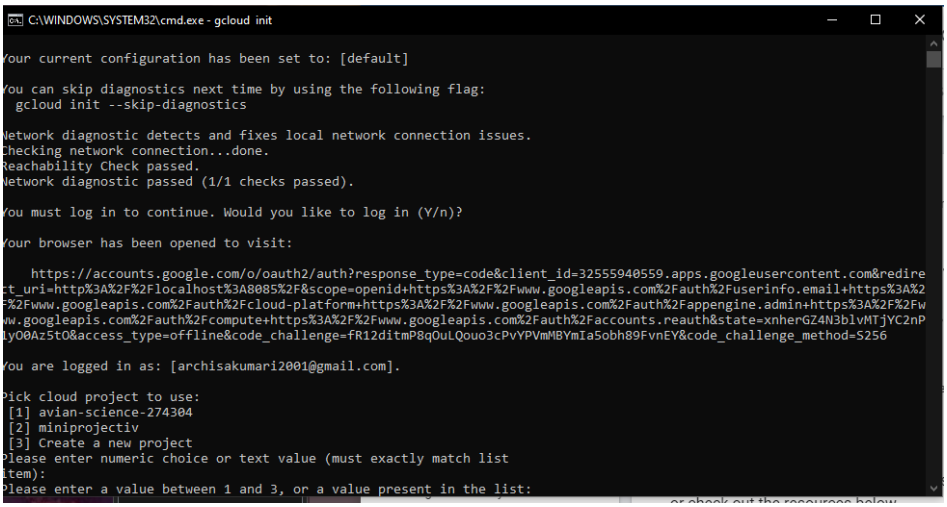
I need to login to continue. So enter Y and your browser will open to login page.

After logging in, your browser will look like:



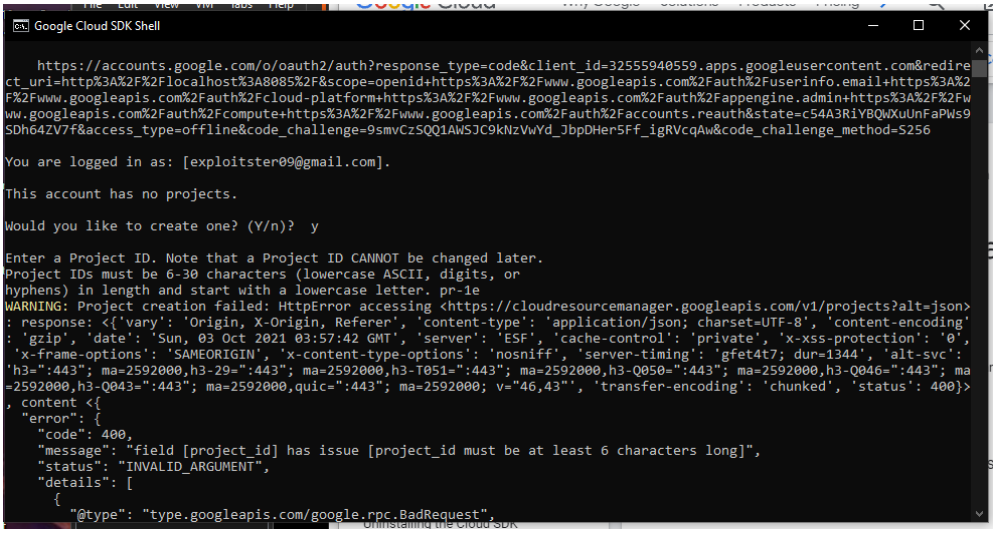
Note: Don’t close the browser, you will require ahead.

After logging in, as you can see i have already created projects, so i can select one project which i wish towork on:

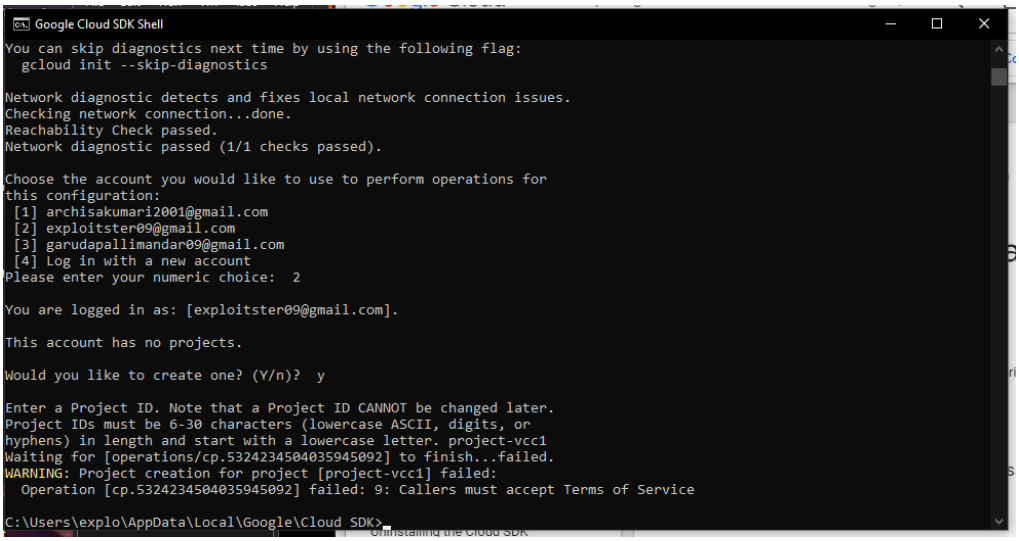


If you are a new user:

After logging in you can see it shows no project created, you need to create a project and give an id as perinformed in gcl

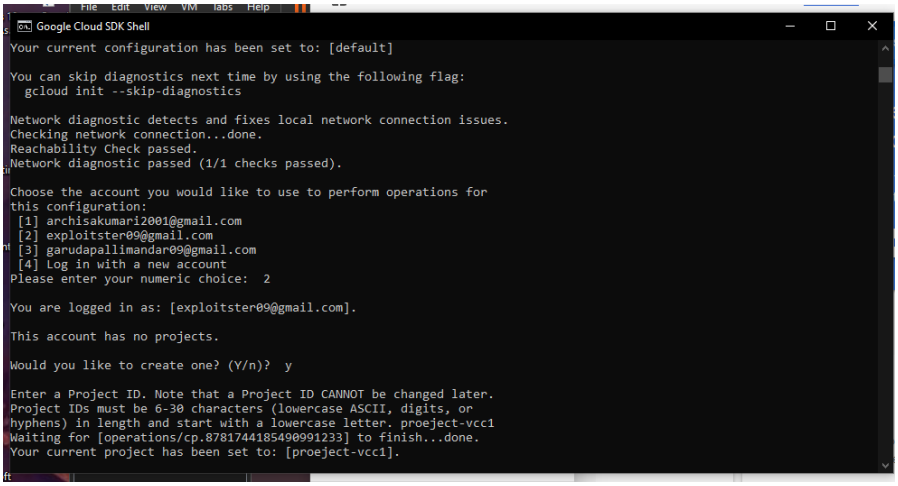


Now as you can see , my characters were less than 6, so it gave me an error of invalid argument. Not to retry , you need to put the command "*gcloud init"* and relogin . Now as you can see below, my id is nolonger invalid but it failed to create as I, being a new user, did not yet accept the terms and conditions of gcl.



To accept terms and conditions, Please follow Step 4.

After accepting terms and condition I again retried by using command and entering as required



You can see my project is successfully created.

Now whenever I login I can see my project and can start working on it.

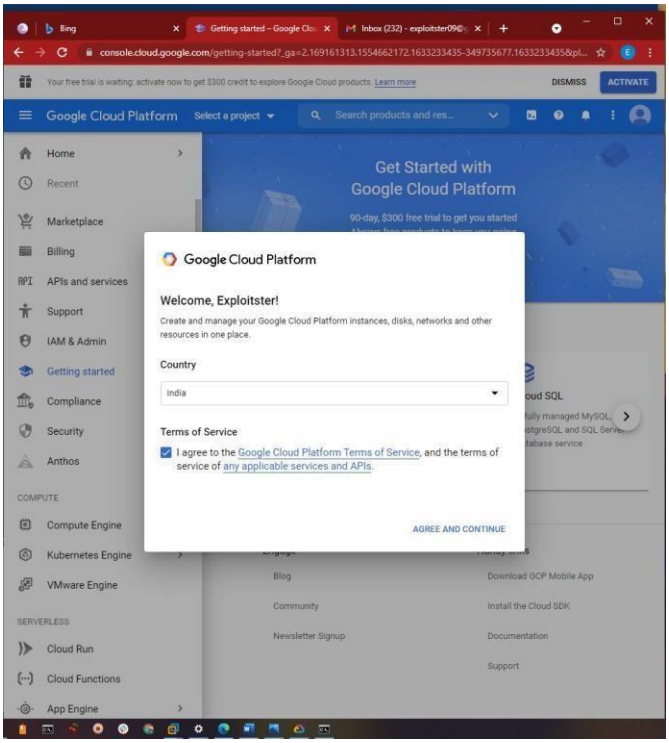
Step 4: Accepting

Terms and condition

To accept terms and

condition,
You need to reopen browser, (which we receive after loggin in) and click on console or you can go to link https://console.cloud.google.com/home/dashboard?project=avian-science-274304&_ga=2.162905982.315020232.1633236299-1476878796.1630504928&_gac=1.185056731.1630504967.Cj0KCQjwpreJBhDvARIsAF1_BU1Mnsq5RP0iIv3WSueOzUlzmPO9HgntOPWEpNYXxbkrQ33lbCoExKAaAuHrEALw_wcB&pli=1

you will receive a notification to agree and continue



And now you are all set. Congratulation !

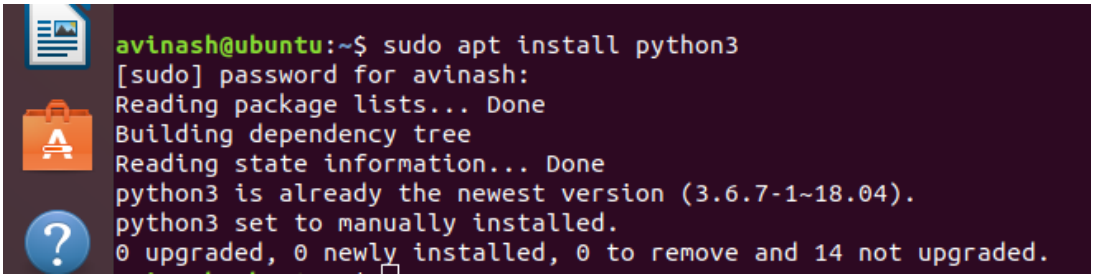
NOTE: YOU CAN ONLY CREATE 11 FREE PROJECTS, SO PLEASE USE IT SAFELY. DO NOTKEEP ON JUST CREATING IF NOT REQUIRED.

Installation on Ubuntu:

Step 1: Installing supported language

Install supported language. I have installed python here

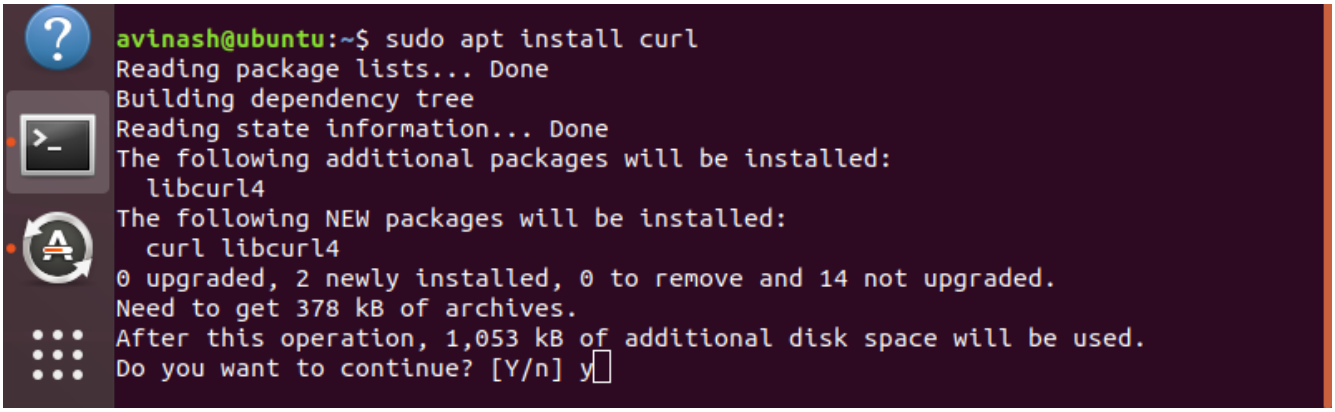
according to my easeCommand: *sudo apt install python3*



Step 2: Installing google app engine

First you would require to install curl. curl is a command line tool to transfer data to or from a server, usingany of the supported protocols. We will use curl here to install Google App engine.

Command : *sudo apt install curl*



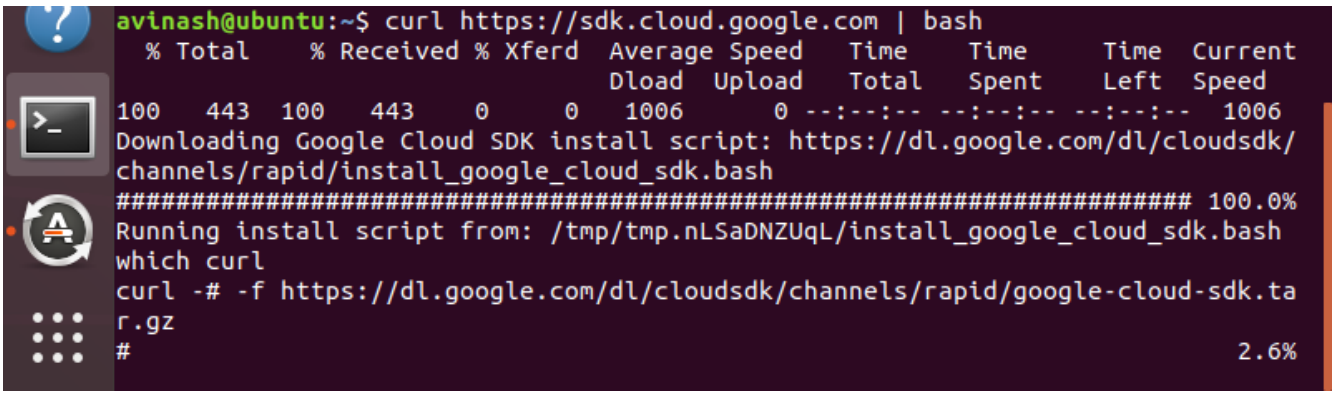
Press Y and enter,

wherever required.

Now to install

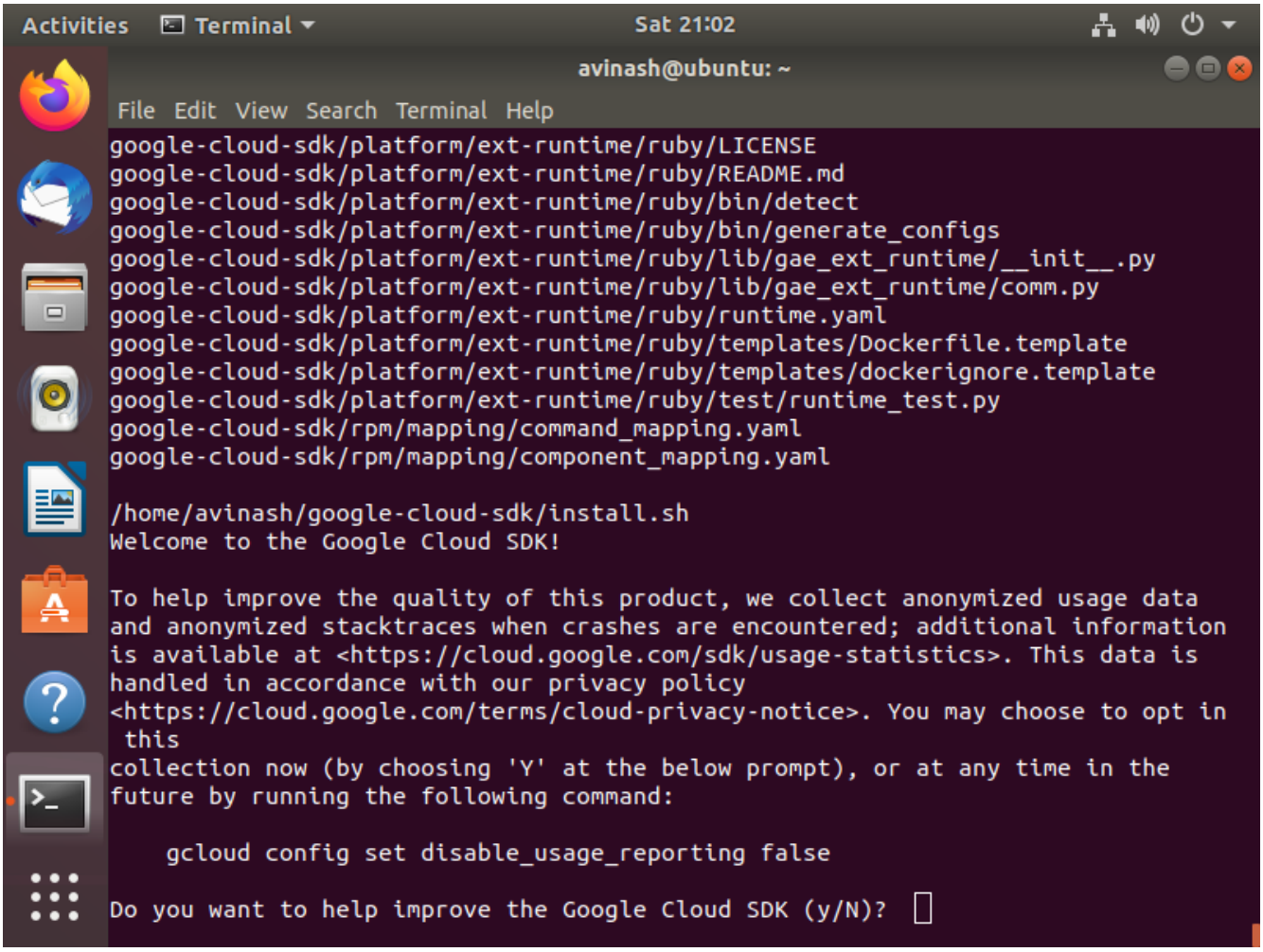
google app engine.

Command: *curl https://sdk.cloud.google.com | bash*

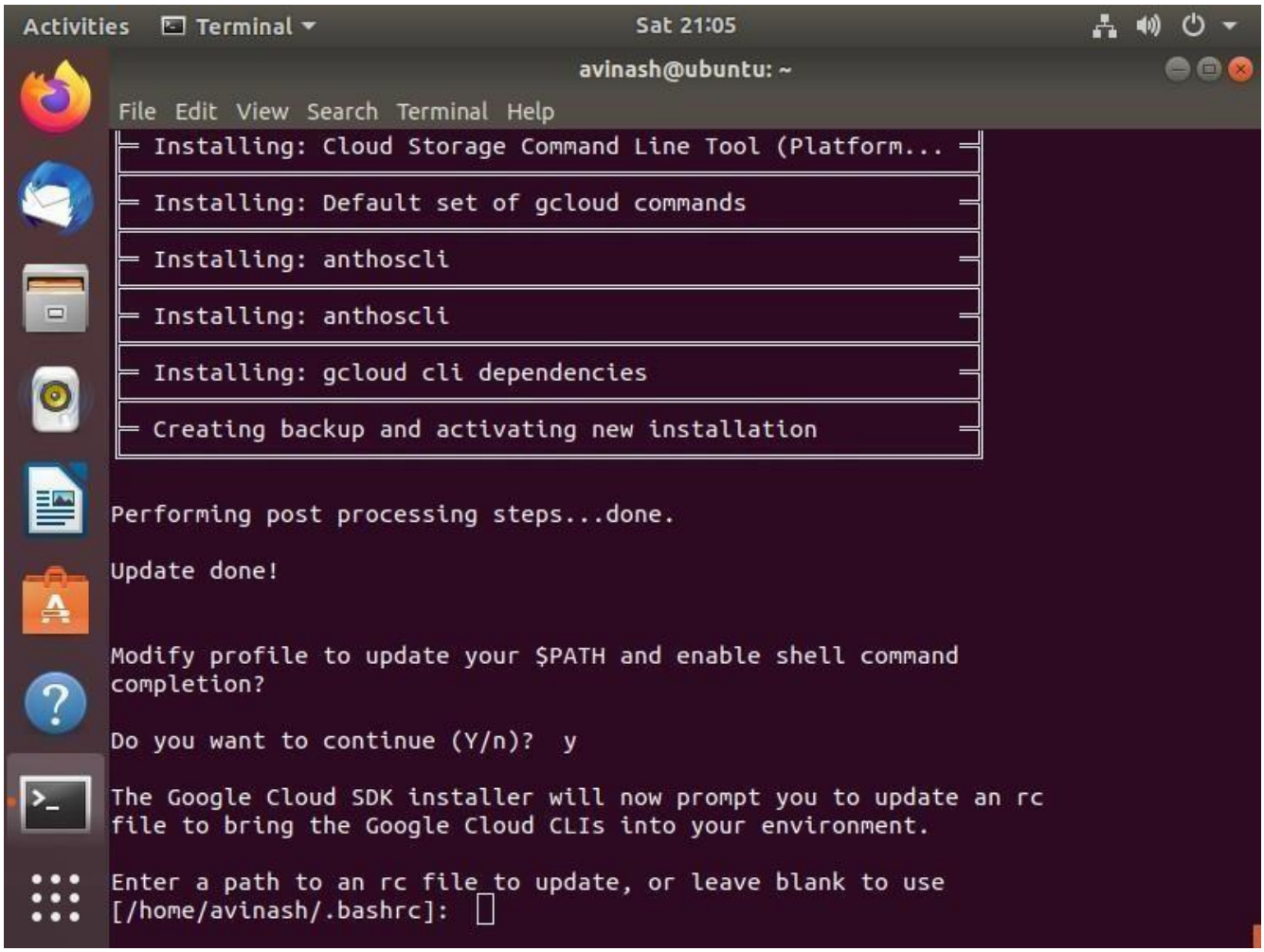


Press enter, wherever required.

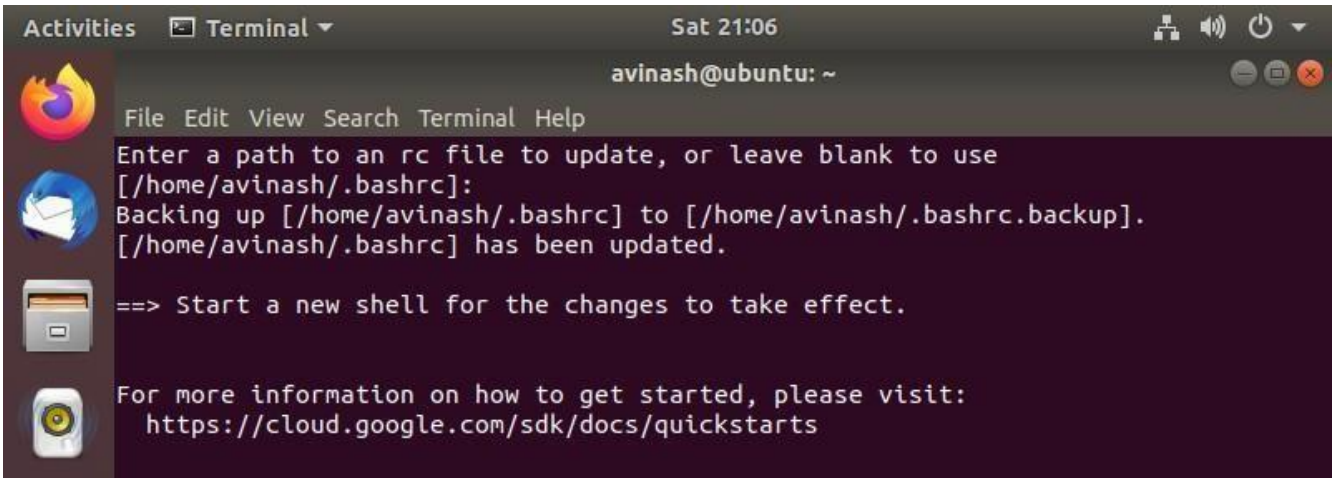
Now , google cloud sdk is installed and you can see a welcome message.



Press Y and enter



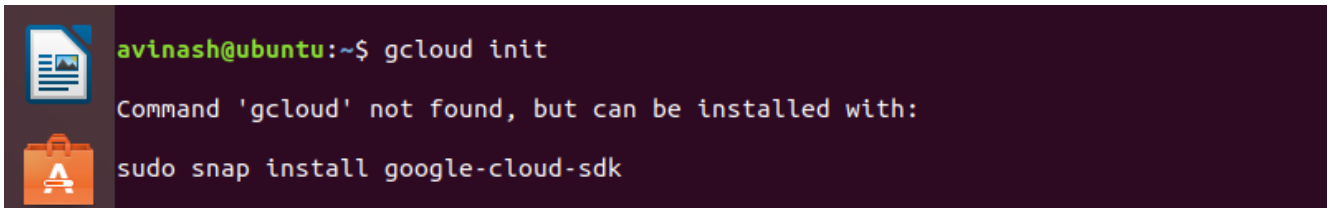
Now you can see , it asks a path to an rc file, leave blank and enter.



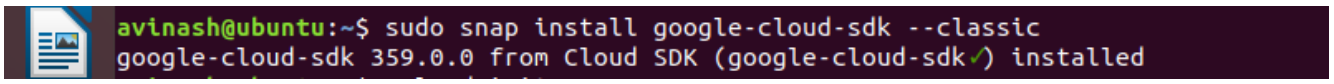
Now as you can see gcl shell is successfully created.

Step 3: Accessing instance

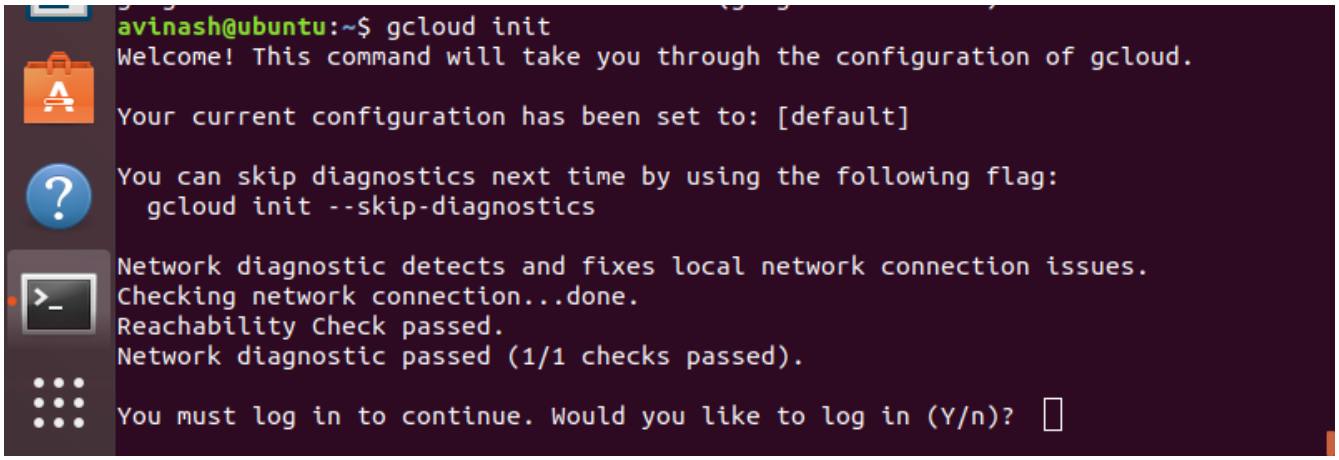
To access
instance, you
will require
Command:
gcloud init



As you can see, I received an error as i did not
installed all packagesTo install all supporting
packages, use
Command: *sudo snap install google-cloud-sdk --classic*



Now after accession instance , you can see it is successfully configured and asks for login



For further steps/ understanding you can follow from step 3 of installation on windows/mac

NOTE: YOU CAN ONLY CREATE 11 FREE PROJECTS, SO PLEASE USE IT SAFELY.
DO NOTKEEP ON JUST CREATING IF NOT REQUIRED.

Practical No -06

Problem Statement: Use GAE launcher to launch the web applications.

Theory: Google App Engine (GAE) is a platform-as-a-service (PaaS) cloud computing platform that allows developers to build and host web applications on Google's infrastructure. GAE supports several programming languages, including Java, Python, Go, and more. GAE also provides a set of tools and libraries to help developers build and deploy their applications.

One of the tools that GAE provides is the GAE Launcher, which is a graphical user interface (GUI) tool that allows developers to easily manage their GAE applications. The GAE Launcher provides a simple interface to start, stop, and deploy GAE applications, as well as view their logs and statistics.

To use the GAE Launcher to launch web applications, developers must first create a new GAE project and configure the necessary settings. This includes specifying the project name, selecting the programming language, and setting the project ID. Once the project is created, developers can then create a new web application and write their code using their preferred programming language.

To launch the web application using the GAE Launcher, developers can simply click on the "Run" button in the GAE Launcher interface. This will start a local development server on the developer's machine and launch the web application in their web browser. Developers can then test their application and make any necessary changes before deploying it to the live GAE server.

Using the GAE Launcher to launch web applications offers several benefits, including:

Simplified management: The GAE Launcher provides a simple and intuitive interface for managing GAE applications, making it easy for developers to start, stop, and deploy their applications.

Local development: The GAE Launcher allows developers to run their applications locally on their own machine, which can help to speed up the development process and reduce errors.

Quick feedback: By launching the web application locally, developers can quickly test their code and get feedback on any issues or bugs before deploying it to the live server.

Overall, the GAE Launcher is a useful tool for developers who are building and deploying web applications on the GAE platform. By providing a simple and intuitive interface for managing GAE applications, the GAE Launcher can help to streamline the development process and improve the quality of the final product.

Conclusion:

In conclusion, installing and using Google App Engine (GAE) can be a useful tool for developers looking to build and host web applications on the cloud. The GAE Launcher is a user-friendly graphical interface that makes it easy to manage GAE applications, launch local development servers, and test code changes. By providing a simple and intuitive interface, the

GAE Launcher can help to streamline the development process and improve the quality of the final product.

Overall, this assignment is a great opportunity for developers to gain hands-on experience with GAE and explore the benefits of using cloud computing for web application development. By installing a C compiler in the virtual machine, creating simple programs, and using the GAE Launcher to launch web applications, developers can gain a deeper understanding of how cloud platforms like GAE can help to accelerate the development process and improve the scalability and reliability of their applications.

Practical No -07

Problem Statement:
Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim

Theory:
CloudSim is a simulation framework that allows researchers and developers to model and simulate cloud computing environments and services. It provides a set of tools and libraries to create and evaluate cloud computing scenarios, including the behavior of data centers, virtual machines (VMs), and cloud applications (cloudlets). Here are some key features and components of CloudSim

Steps to Simulate a Cloud Scenario in CloudSim and Implement a Custom Scheduling Algorithm

1. Set Up CloudSim

- Download **CloudSim** from the official GitHub repository or install it using a Java IDE.
- Extract the CloudSim package and set up the environment using an IDE like **Eclipse** or **IntelliJ**.
- Ensure **JDK (Java Development Kit)** is installed.

2. Create a CloudSim Simulation

- Initialize **CloudSim**.
- Create **Datacenters** (to represent cloud infrastructure).
- Create **Hosts** inside the datacenter.
- Create **VMs (Virtual Machines)** to run cloudlets.
- Create **Cloudlets** (representing tasks) and assign them to VMs.

3. Implement a Custom Scheduling Algorithm

By default, CloudSim provides basic scheduling policies (like Space-Shared and Time-Shared). To implement a **custom scheduling algorithm**, follow these steps:

1. **Extend the VmAllocationPolicy or VmScheduler Class**
   - If you want to change how VMs are assigned to hosts, extend `VmAllocationPolicy`.
   - If you want to change how Cloudlets are scheduled inside a VM, extend `CloudletScheduler`.
2. **Override the Scheduling Methods**
   - Implement your custom logic inside methods like:

```
public boolean allocateHostForVm(Vm vm) {
    // Your custom scheduling logic here
    return true; // Return true if allocation is successful
}
```

3. **Register Your Custom Policy in the Simulation**
   - Use the custom scheduling class while creating data center objects:

```
DatacenterBroker broker = new DatacenterBroker("CustomBroker");
broker.submitVmList(vmList);
broker.submitCloudletList(cloudletList);
```

4. Run the Simulation

- Execute the Java program.

- CloudSim will simulate the cloud environment based on the implemented scheduling algorithm.
- Results can be analyzed to compare performance with default schedulers.

5. Analyze the Results

- Capture metrics such as execution time, CPU utilization, and resource allocation.
- Compare your scheduler's efficiency with the default ones.

Example: Custom Scheduling Algorithm in CloudSim

Here's a simple example of a **custom VM scheduling algorithm**:

```
public class CustomVmAllocationPolicy extends VmAllocationPolicy {
    public CustomVmAllocationPolicy(List<? extends Host> list) {
        super(list);
    }

    @Override
    public boolean allocateHostForVm(Vm vm) {
        for (Host host : getHostList()) {
            if (host.isSuitableForVm(vm)) {
                return host.vmCreate(vm);
            }
        }
        return false;
    }
}
```

Conclusion:
Hence we have study a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim

Practical No -08

Problem Statement:
The objective of this assignment is to find a procedure to transfer files from one virtual machine to another virtual machine.

Theory:
There are several ways to transfer files between virtual machines, including:

Shared Folders: If both virtual machines are running on the same host, you can set up a shared folder that allows files to be transferred between them. This method is simple and efficient, but requires the virtual machines to be running on the same host machine.

Network File Transfer: If the virtual machines are running on different hosts, you can use a network file transfer protocol such as FTP or SFTP to transfer files between them. This method requires a network connection between the virtual machines and can be slower than using shared folders.

Cloud Storage: If both virtual machines are connected to the internet, you can use cloud storage services such as Google Drive, Dropbox, or OneDrive to transfer files between them. This method can be convenient, but requires an internet connection and can be slower than using shared folders.

Procedure:
Here is a step-by-step procedure for transferring files between virtual machines using shared folders:

Ensure that both virtual machines are running on the same host machine.

Create a shared folder on the host machine and add files to it that you want to transfer.

In each virtual machine, configure the shared folder by selecting the "Devices" menu and choosing "Shared Folders".

In the "Shared Folders" settings, add the shared folder from the host machine.

In each virtual machine, mount the shared folder by entering the following command in the terminal:

sudo mount -t vboxsf [shared_folder_name] [mount_point]

Replace [shared_folder_name] with the name of the shared folder and [mount_point] with the location where you want to mount the shared folder.

Once the shared folder is mounted, you can access its contents from within the virtual machine and copy files between the virtual machines as needed.

To unmount the shared folder, enter the following command in the terminal:

sudo umount [mount_point]

Replace [mount_point] with the location of the mounted shared folder.

Conclusion:

In conclusion, there are several ways to transfer files between virtual machines, including shared folders, network file transfer, and cloud storage. By using shared folders, files can be easily transferred between virtual machines running on the same host machine. This method is simple and efficient, and can be a convenient option for developers and users who need to transfer files between virtual machines. By following the procedure outlined above, users can transfer files between virtual machines quickly and easily, without the need for additional software or tools.