

Expt No: 01

Date:

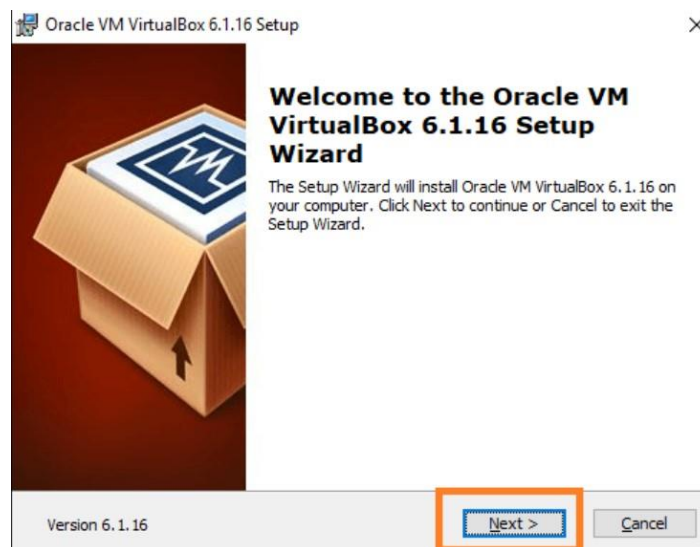
Install Virtual Box with Linux on the top of Windows 10 or 11

Aim:

To install Virtual Box with linux os in the top of Windows 10 or 11.

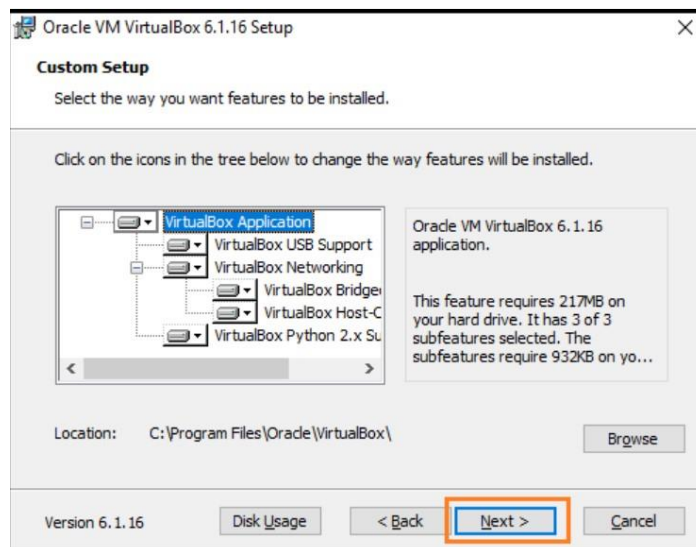
Procedure:

Step 1 : Download the Virtual Box from <https://www.virtualbox.org/>. Now, open the VirtualBox.exe file from where we have downloaded this file in the system. After that, the VirtualBox installation window will open.

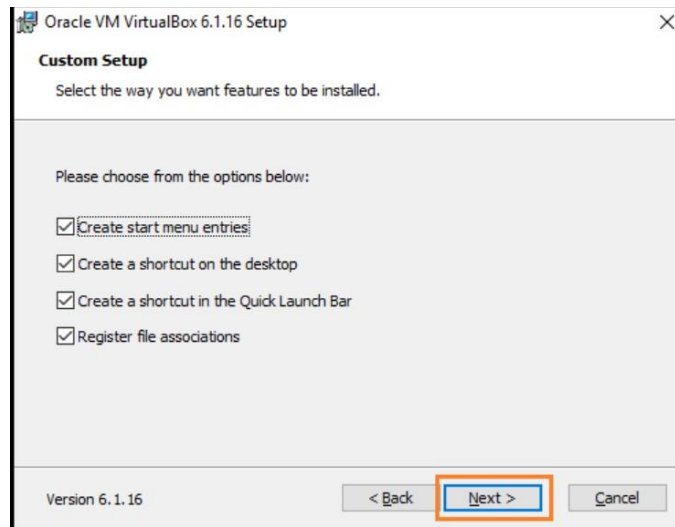


Step 2 : Click the next button in the above window shown.

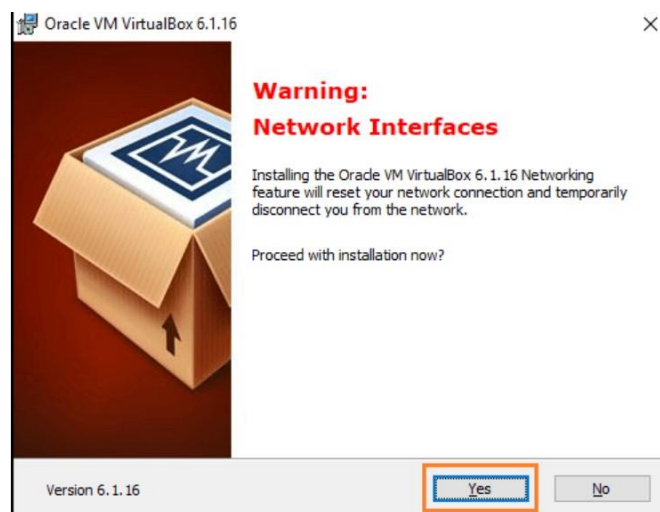
Step 3: Choose the installation folder and click on the Next button.



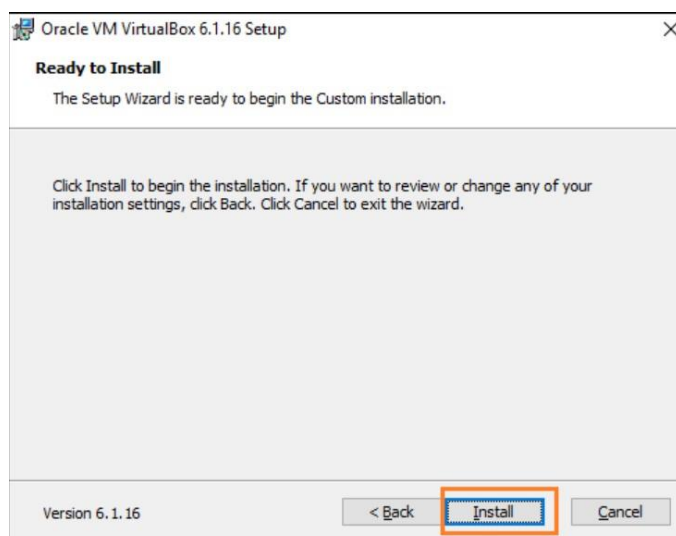
Step 4 : Now, we will choose the features that we want to install and then click on the Next button.



Step 5 : Click on the Yes button to install the Oracle VirtualBox interfaces.



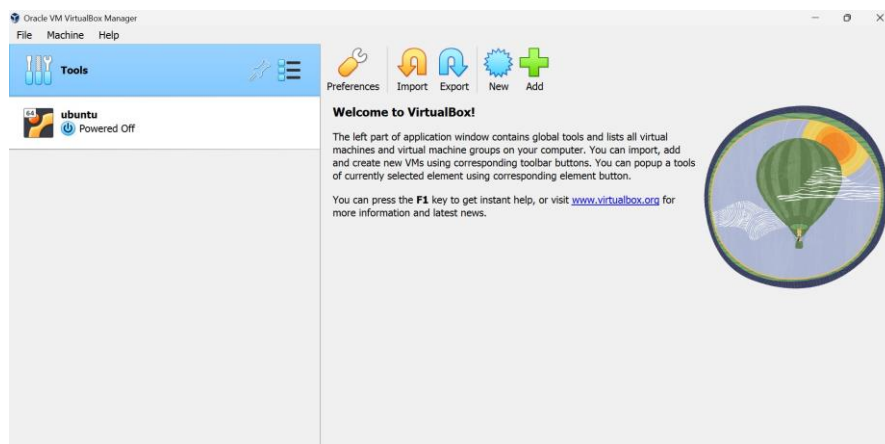
Step 6 : Click on the Install button when it prompted. After that, it will start installing the VirtualBox in our system.



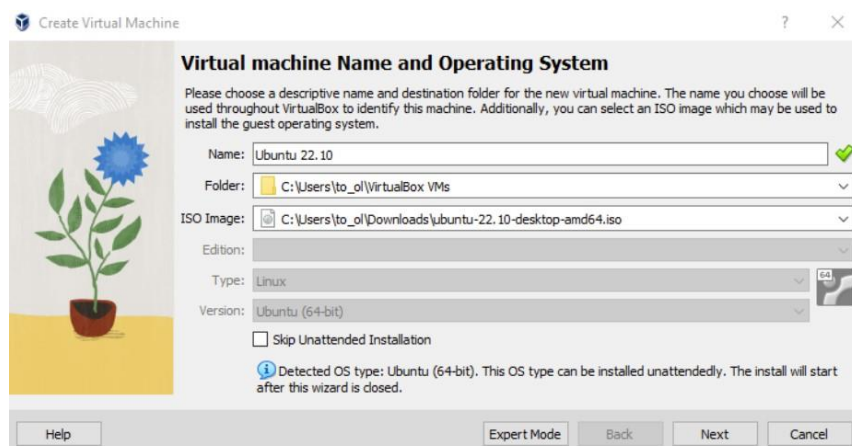
Step 7 : After completing all these processes, click on the Finish button. When we do this, the installation tab will be closed, and VirtualBox will be opened.



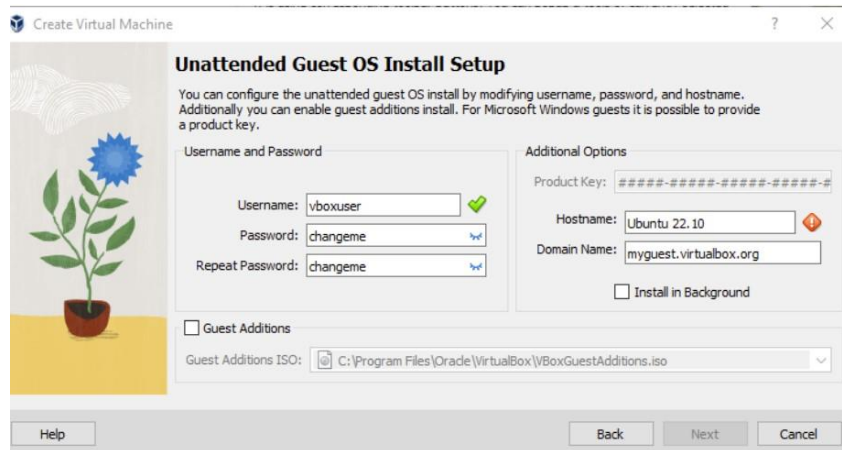
Step 8 : Open the VirtualBox.



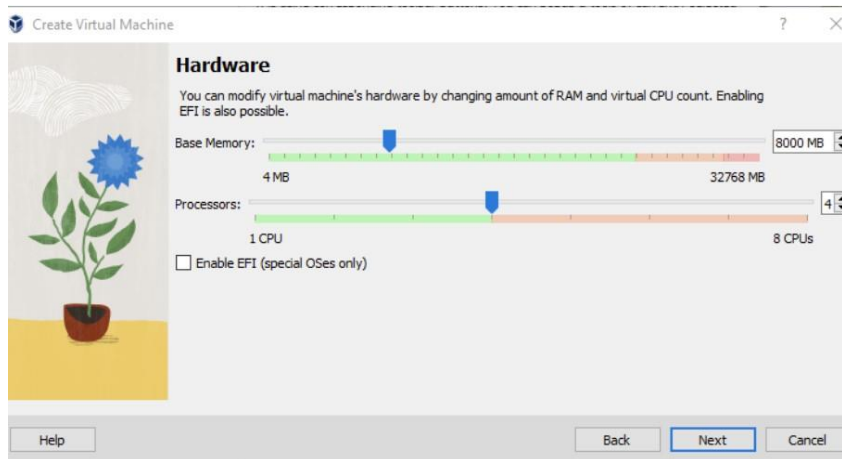
Step 9 : Download the Ubuntu (Linux flavour) from <https://ubuntu.com/download/desktop>. Create a new Virtual Machine by clicking the New icon on the top of the window. A pop window is opened. Fill the name of the machine, and the directory path where the work is supposed to be saved. Link the downloaded Ubuntu iso file in the newly created Virtual Machine. Click Next button.



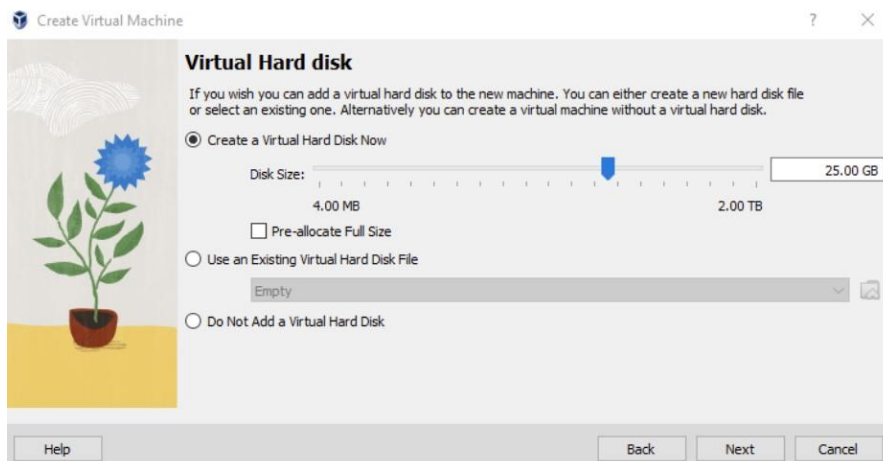
Step 10 : A default credentials will be provided by Virtual Box. It is advised to change the username and password for the created virtual machine. After editing it click Next button.



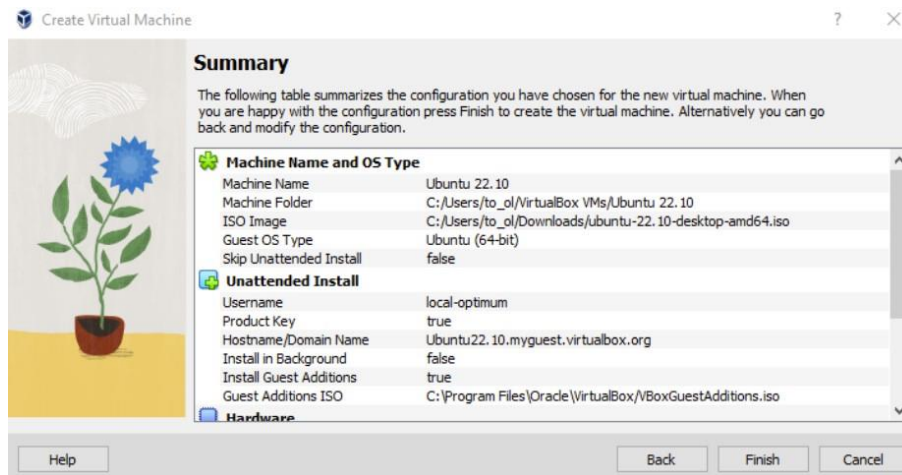
Step 11 : According to your usage configure the memory and processor number and click Next button.



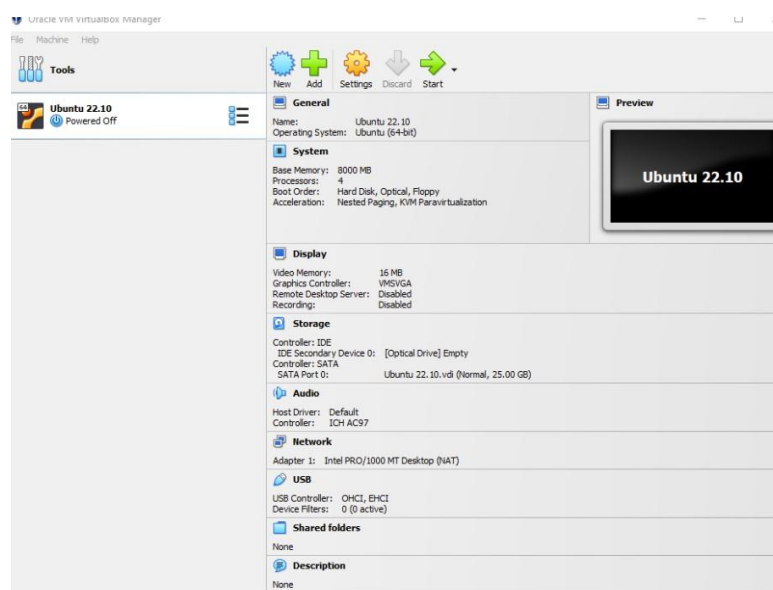
Step 12 : Configure the Disk space for virtual machine and click Next.



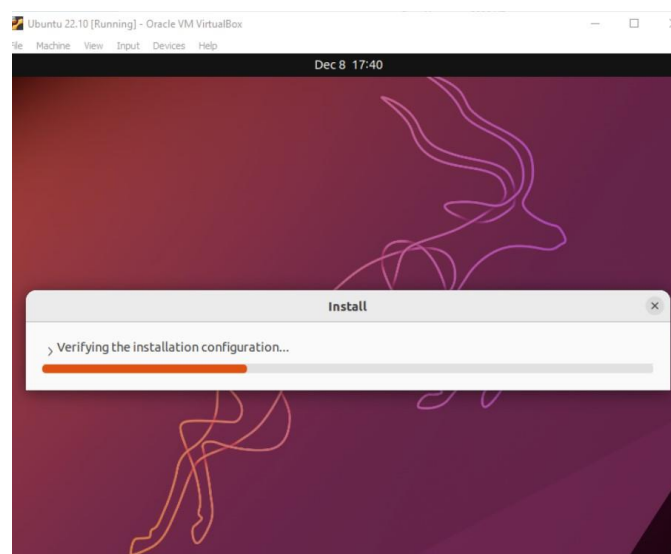
Step 13 : Summary of the machine details will be displayed. Click Finish.



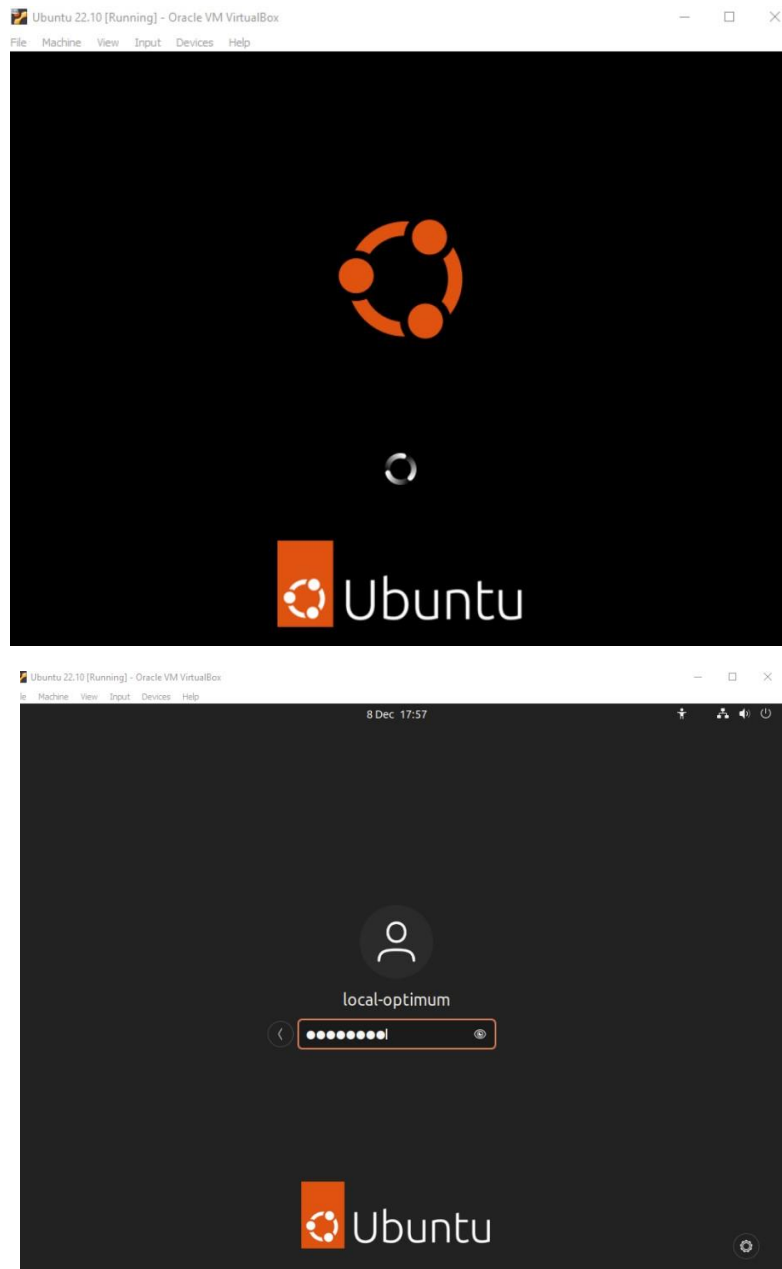
Step 14 : Click Start to launch the created Virtual Machine.



Step 15 : On first boot the unattended installation will kick in so do not interact with the prompt to 'Try and Install Ubuntu' and let it progress automatically to the splash screen and into the installer.



Step 16 : Once the installation completes, the machine will automatically reboot to complete the installation.



Result:

Thus, the Virtual Box was installed with Ubuntu (Linux flavour) on the top of Windows 11 successfully.

Expt No: 02

Date:

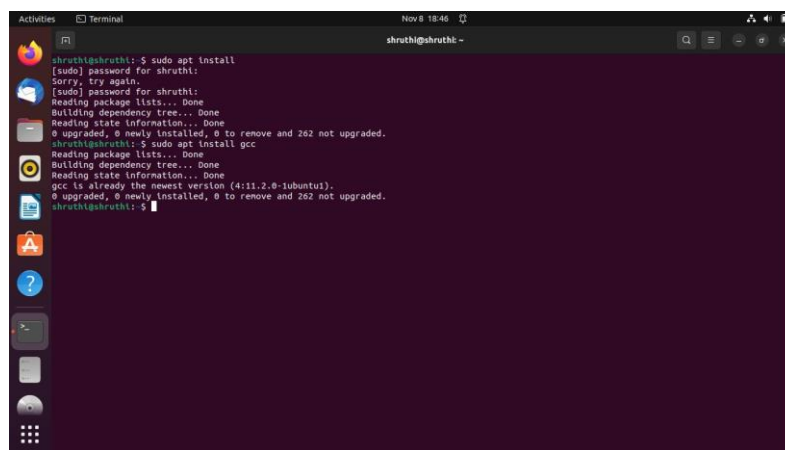
Install a C compiler in the virtual machine created using virtual box and execute Simple Programs

Aim:

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

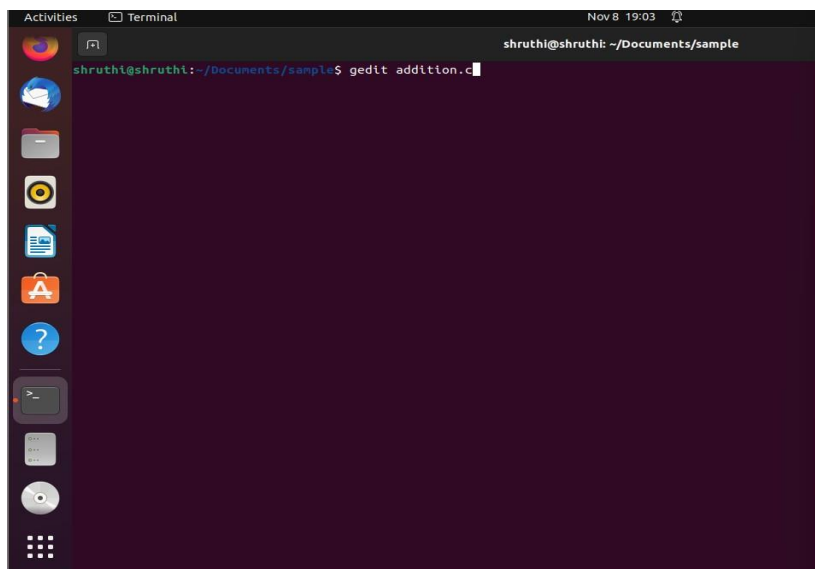
Procedure:

Step 1 : Open the virtual machine in the virtual box. In the terminal enter the command “sudo apt install gcc” for installing c compiler.



```
shruthi@shruthi:~$ sudo apt install gcc
[sudo] password for shruthi:
[sudo] password for shruthi:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 262 not upgraded.
shruthi@shruthi:~$ sudo apt install gcc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc is already the newest version (4:11.2.0-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 262 not upgraded.
shruthi@shruthi:~$
```

Step 2 : Type gedit <filename>.



```
shruthi@shruthi:~/Documents/sample$ gedit addition.c
```

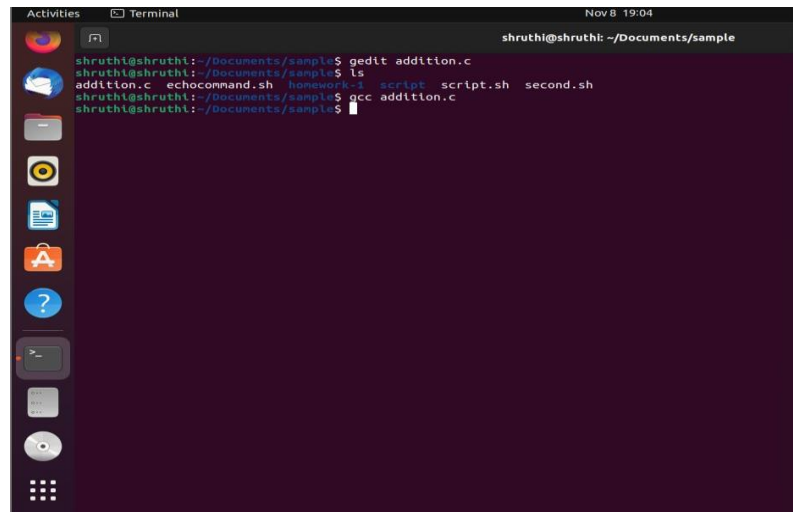
Step 3 : Type the c program



A screenshot of a Linux desktop environment. The 'Activities' panel is on the left. A text editor window titled 'addition.c' is open, showing the following code:

```
#include<stdio.h>
void main(){
    printf(" Cloud Computing Laboratory");
}
```

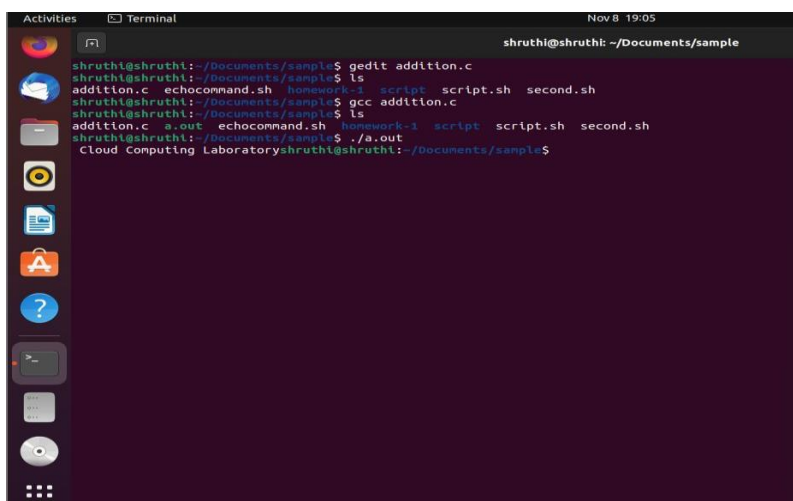
Step 4 : Compile the C program



A screenshot of a terminal window. The user has executed the following commands:

```
shruthi@shruthi: ~/Documents/sample
shruthi@shruthi:~/Documents/sample$ gedit addition.c
shruthi@shruthi:~/Documents/sample$ ls
addition.c  echocommand.sh  homework-1  script  script.sh  second.sh
shruthi@shruthi:~/Documents/sample$ gcc addition.c
shruthi@shruthi:~/Documents/sample$
```

Step 5 : Run the C program



A screenshot of a terminal window. The user has executed the following commands:

```
shruthi@shruthi:~/Documents/sample$ gedit addition.c
shruthi@shruthi:~/Documents/sample$ ls
addition.c  echocommand.sh  homework-1  script  script.sh  second.sh
shruthi@shruthi:~/Documents/sample$ gcc addition.c
shruthi@shruthi:~/Documents/sample$ ls
addition.c  a.out  echocommand.sh  homework-1  script  script.sh  second.sh
shruthi@shruthi:~/Documents/sample$ ./a.out
Cloud Computing Laboratoryshruthi@shruthi:~/Documents/sample$
```

Result:

Thus a C compiler has been installed and a simple program has compiled and executed successfully.

Expt No: 03

Date:

Create hello world app and host the app in GitHub.

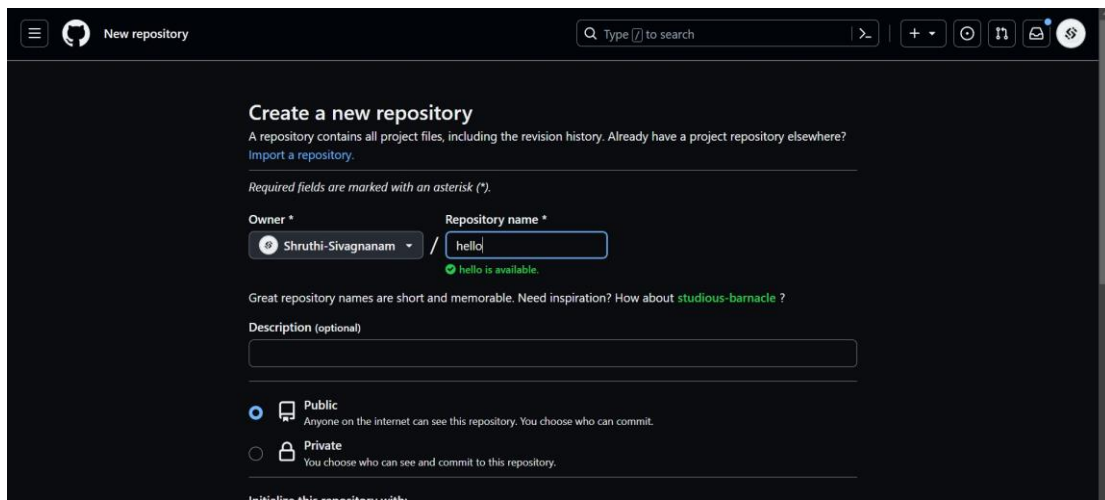
Aim:

To create a hello world app and host the app in the GitHub.

Procedure:

Step 1 : Create a GitHub account.

Step 2 : Create a repo "hello".

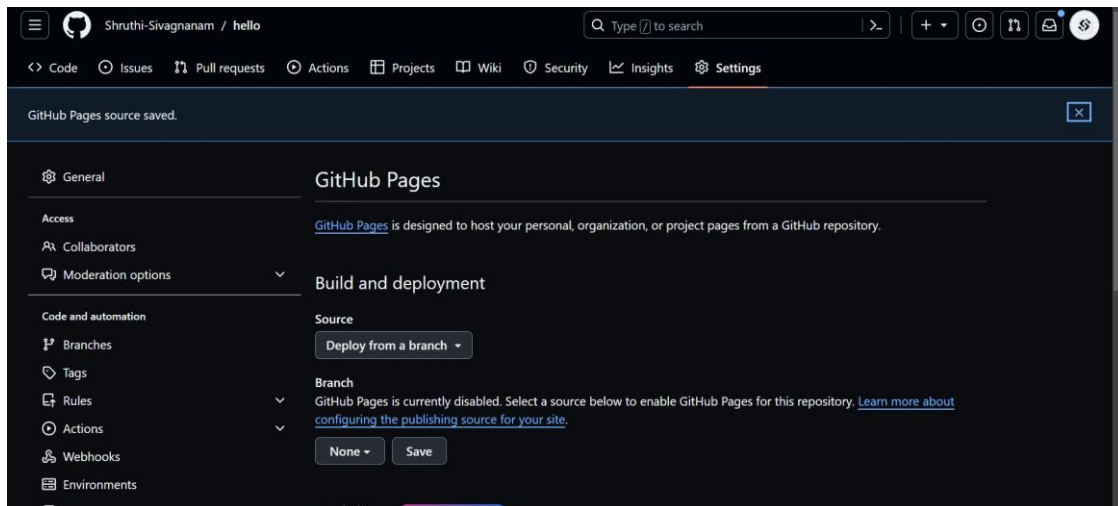


Step 3 : Upload the below index.html file in the created hello repo.

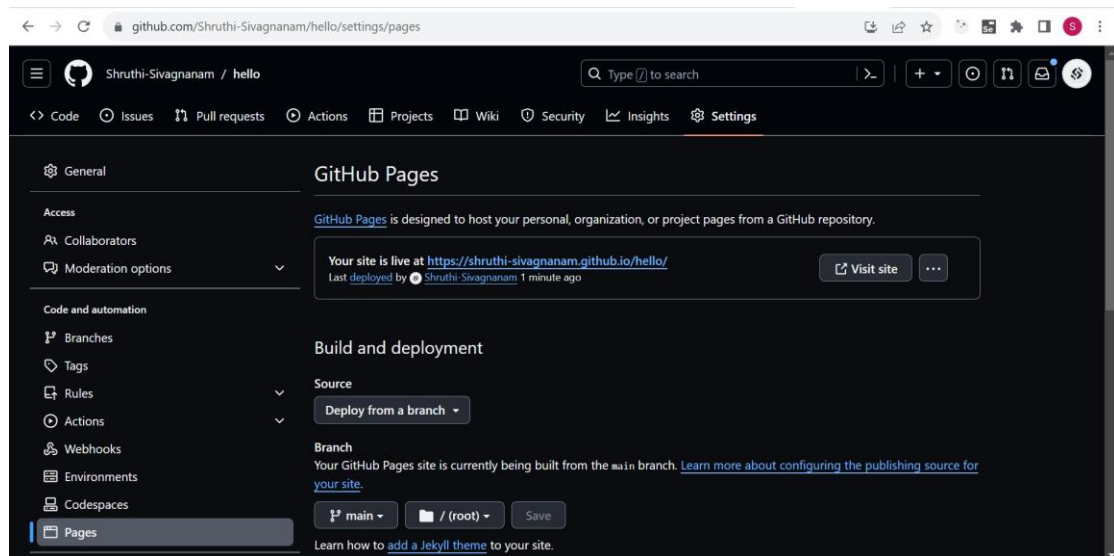
Program :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Hello World</title>
  </head>
  <body>
    Hello World
  </body>
</html>
```

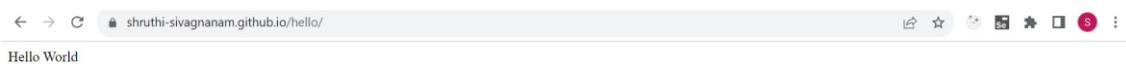
Step 4 : Go to Setting > Pages (in left side menu). Click Save.



Step 5 : Wait for few minutes for live link to appear.



Step 6 : Now the link is available for use.



Result:

Thus, a hello world app has been created and hosted in the GitHub pages successfully.

Expt No: 04

Date:

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

Aim:

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

Procedure:

Step 1 : Download the CloudSim zip file from <https://github.com/Cloudslab/cloudsim/releases> .

Step 2 : Unzip the file in a desired folder.

Step 3 : Open the project directory (cloudsim-3.0.3\examples) in any text editor (VS code).

Step 4 : In reference section of the project add all the jar files from cloudsim-3.0.3\jars.

Step 5 : Execute the sample existing CloudSim examples in the project.

Step 6 : Create a file RoundRobin.java and code the below program in that file.

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 java.util.Random;
import org.cloudbus.cloudsim.Cloudlet;
import org.cloudbus.cloudsim.CloudletSchedulerSpaceShared;
import org.cloudbus.cloudsim.Datacenter;
import org.cloudbus.cloudsim.DatacenterBroker;
import org.cloudbus.cloudsim.DatacenterCharacteristics;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Pe;
import org.cloudbus.cloudsim.Storage;
import org.cloudbus.cloudsim.UtilizationModel;
import org.cloudbus.cloudsim.UtilizationModelFull;
```

```

import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicySimple;
import org.cloudbus.cloudsim.VmSchedulerTimeShared;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
public class RoundRobin {
    private static float timeSlice = (float) 8;
    private static List<Cloudlet> cloudletList;
    private static List<Vm> vmList;
    private static List<Vm> createVM(int userId, int vms) {
        LinkedList<Vm> list = new LinkedList<Vm>();
        long size = 10000; // image size (MB)
        int ram = 512; // vm memory (MB)
        int mips = 1000;
        long bw = 1000;
        int pesNumber = 1; // number of cpus
        String vmm = "Xen"; // VMM name
        Vm[] vm = new Vm[vms];
        for (int i = 0; i < vms; i++) {
            vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerSpaceShared());
            list.add(vm[i]);
        }
        return list;
    }
    private static List<Cloudlet> createCloudlet(int userId, int cloudlets) {
        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();
        long length = 1000;
        long fileSize = 300;

```

```

    long outputSize = 300;

    int pesNumber = 1;

    UtilizationModel utilizationModel = new UtilizationModelFull();

    Cloudlet[] cloudlet = new Cloudlet[cloudlets];

    for (int i = 0; i < cloudlets; i++) {
        Random r = new Random();

        cloudlet[i] = new Cloudlet(i, length + r.nextInt(2000), pesNumber, fileSize, outputSize,
utilizationModel,
        utilizationModel, utilizationModel);

        cloudlet[i].setUserId(userId);

        list.add(cloudlet[i]);
    }
    return list;
}

public static void main(String[] args) {
    Log.println("===== Round Robin Task Scheduling Algorithm Implementation
=====");

    try {
        Log.println("===== Starting Execution =====");

        int num_user = 3; // number of grid users

        Calendar calendar = Calendar.getInstance();

        boolean trace_flag = false; // mean trace events

        CloudSim.init(num_user, calendar, trace_flag);

        @SuppressWarnings("not used")

        Datacenter datacenter0 = createDatacenter("Datacenter_0");

        Datacenter datacenter1 = createDatacenter("Datacenter_1");

        DatacenterBroker broker = createBroker();

        int brokerId = broker.getId();

        vmList = createVM(brokerId, 10); // creating 10 vms

        cloudletList = createCloudlet(brokerId, 40); // creating 40 cloudlets

```

```

        broker.submitVmList(vmlist);
        broker.submitCloudletList(cloudletList);
        CloudSim.startSimulation();
        List<Cloudlet> newList = broker.getCloudletReceivedList();
        CloudSim.stopSimulation();
        printCloudletList(newList);
        Log.println("Round Robin has finished executing!");
    } catch (Exception e) {
        e.printStackTrace();
        Log.println("The simulation has been terminated due to an unexpected error");
    }
}

private static Datacenter createDatacenter(String name) {
    List<Host> hostList = new ArrayList<Host>();
    List<Pe> peList1 = new ArrayList<Pe>();
    int mips = 1000;

    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)));
    List<Pe> peList2 = new ArrayList<Pe>();
    peList2.add(new Pe(0, new PeProvisionerSimple(mips)));
    peList2.add(new Pe(1, new PeProvisionerSimple(mips)));
    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,
        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

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);
Datacenter datacenter = null;
try {
    datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
} catch (Exception e) {

```

```

        e.printStackTrace();
    }
    return datacenter;
}

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

private static void printCloudletList(List<Cloudlet> list) {
    int size = list.size();
    Cloudlet cloudlet;
    int pes = 0;
    float sum = 0;
    float burstTime[] = new float[size];
    float waitingTime[] = new float[size];
    float turnAroundTime[] = new float[size];
    float a[] = new float[size];
    String indent = "  ";
    DecimalFormat dft = new DecimalFormat("###.##");
    for (int i = 0; i < size; i++) {
        cloudlet = list.get(i);
        String cpuTime = dft.format(cloudlet.getActualCPUTime());
        float convertedCPUTime = (float) Double.parseDouble(cpuTime);
        burstTime[i] = convertedCPUTime; // burst time is equal to execution time.
    }
}

```



```

    }
    for (int i = 0; i < size; i++) {
        a[i] = burstTime[i];
    }
    for (int i = 0; i < size; i++) {
        waitingTime[i] = 0;
    }
    do {
        for (int i = 0; i < size; i++) {
            if (burstTime[i] > timeSlice) {
                burstTime[i] -= timeSlice;
                for (int j = 0; j < size; j++) {
                    if ((j != i) && (burstTime[j] != 0)) {
                        waitingTime[j] += timeSlice;
                    }
                }
            } else {
                for (int j = 0; j < size; j++) {
                    if ((j != i) && (burstTime[j] != 0)) {
                        waitingTime[j] += burstTime[i];
                    }
                }
                burstTime[i] = 0;
            }
        }
        sum = 0;
        for (int k = 0; k < size; k++) {
            sum += burstTime[k];
        }
    } while (sum != 0);

```

```

for (int i = 0; i < size; i++) {
    turnAroundTime[i] = waitingTime[i] + a[i];
}
Log.println("===== OUTPUT =====");
Log.print("Cloudlet \t Burst Time \t Waiting Time \t Turn Around Time");
Log.println();
Log.print("-----");
for (int i = 0; i < size; i++) {
    cloudlet = list.get(i);
    pes = list.get(i).getNumberOfPes();
    System.out.println("Cloudlet: " + cloudlet.getCloudletId() + "\t\t" + a[i] + "\t\t" +
waitingTime[i]
        + "\t\t" + turnAroundTime[i]);
}
float averageWaitingTime = 0;
float averageTurnAroundTime = 0;
for (int j = 0; j < size; j++) {
    averageWaitingTime += waitingTime[j];
}
for (int j = 0; j < size; j++) {
    averageTurnAroundTime += turnAroundTime[j];
}
System.out.println("Average Waiting Time on Total: " + (averageWaitingTime / size)
    + "\nAverage Turn Around Time on Total: " + (averageTurnAroundTime / size));
Log.println();
Log.println("Cloudlet ID" + indent + "STATUS" + indent +
    "Data center ID" + indent + "VM ID" + indent + indent + "Time" + indent + "Start
Time" + indent
    + "Finish Time" + indent + "User ID" + indent + "Waiting Time" + indent + indent +
"Turn Around Time");
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.println(indent + indent + indent + cloudlet.getResourceId() + indent + indent +
indent

    + cloudlet.getVmId() +

    indent + indent + indent + dft.format(cloudlet.getActualCPUTime()) +

    indent + indent + dft.format(cloudlet.getExecStartTime()) + indent + indent

    + dft.format(cloudlet.getFinishTime()) + indent + indent + indent +

cloudlet.getUserId()

    + indent + indent + indent + waitingTime[i] + indent + indent + indent +

turnAroundTime[i]);

}

}

}

}

}

```

Output:

```

===== Round Robin Task Scheduling Algorithm Implementation =====
===== Starting Execution =====
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
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.0: Broker: Trying to Create VM #5 in Datacenter_0
0.0: Broker: Trying to Create VM #6 in Datacenter_0
0.0: Broker: Trying to Create VM #7 in Datacenter_0
0.0: Broker: Trying to Create VM #8 in Datacenter_0
0.0: Broker: Trying to Create VM #9 in Datacenter_0
[VMScheduler.vmCreate] Allocation of VM #6 to Host #0 failed by RAM
[VMScheduler.vmCreate] Allocation of VM #7 to Host #0 failed by RAM
[VMScheduler.vmCreate] Allocation of VM #7 to Host #1 failed by MIPS
[VMScheduler.vmCreate] Allocation of VM #8 to Host #0 failed by RAM
[VMScheduler.vmCreate] Allocation of VM #8 to Host #1 failed by MIPS
[VMScheduler.vmCreate] Allocation of VM #9 to Host #0 failed by RAM
[VMScheduler.vmCreate] Allocation of VM #9 to Host #1 failed by MIPS
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: VM #3 has been created in Datacenter #2, Host #1
0.1: Broker: VM #4 has been created in Datacenter #2, Host #0
0.1: Broker: VM #5 has been created in Datacenter #2, Host #1
0.1: Broker: Creation of VM #6 failed in Datacenter #2
0.1: Broker: Creation of VM #7 failed in Datacenter #2
0.1: Broker: Creation of VM #8 failed in Datacenter #2

```

```

0.1: Broker: Creation of VM #7 failed in Datacenter #2
0.1: Broker: Creation of VM #8 failed in Datacenter #2
0.1: Broker: Creation of VM #9 failed in Datacenter #2
0.1: Broker: Trying to Create VM #6 in Datacenter_1
0.1: Broker: Trying to Create VM #7 in Datacenter_1
0.1: Broker: Trying to Create VM #8 in Datacenter_1
0.1: Broker: Trying to Create VM #9 in Datacenter_1
0.2: Broker: VM #6 has been created in Datacenter #3, Host #0
0.2: Broker: VM #7 has been created in Datacenter #3, Host #0
0.2: Broker: VM #8 has been created in Datacenter #3, Host #0
0.2: Broker: VM #9 has been created in Datacenter #3, Host #1
0.2: Broker: Sending cloudlet 0 to VM #0
0.2: Broker: Sending cloudlet 1 to VM #1
0.2: Broker: Sending cloudlet 2 to VM #2
0.2: Broker: Sending cloudlet 3 to VM #3
0.2: Broker: Sending cloudlet 4 to VM #4
0.2: Broker: Sending cloudlet 5 to VM #5
0.2: Broker: Sending cloudlet 6 to VM #6
0.2: Broker: Sending cloudlet 7 to VM #7
0.2: Broker: Sending cloudlet 8 to VM #8
0.2: Broker: Sending cloudlet 9 to VM #9
0.2: Broker: Sending cloudlet 10 to VM #0
0.2: Broker: Sending cloudlet 11 to VM #1
0.2: Broker: Sending cloudlet 12 to VM #2
0.2: Broker: Sending cloudlet 13 to VM #3
0.2: Broker: Sending cloudlet 14 to VM #4
0.2: Broker: Sending cloudlet 15 to VM #5
0.2: Broker: Sending cloudlet 16 to VM #6
0.2: Broker: Sending cloudlet 17 to VM #7
0.2: Broker: Sending cloudlet 18 to VM #8
0.2: Broker: Sending cloudlet 19 to VM #9
0.2: Broker: Sending cloudlet 20 to VM #0
0.2: Broker: Sending cloudlet 21 to VM #1
0.2: Broker: Sending cloudlet 22 to VM #2
0.2: Broker: Sending cloudlet 23 to VM #3
0.2: Broker: Sending cloudlet 24 to VM #4

```

```

PROBLEMS ① OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
3.673: Broken: Cloudlet 19 received
3.786: Broken: Cloudlet 14 received
4.281: Broken: Cloudlet 18 received
4.282: Broken: Cloudlet 13 received
4.3909999999999999: Broken: Cloudlet 17 received
4.3919999999999999: Broken: Cloudlet 15 received
4.5649999999999999: Broken: Cloudlet 16 received
4.7429999999999999: Broken: Cloudlet 28 received
4.9469999999999999: Broken: Cloudlet 12 received
5.0879999999999999: Broken: Cloudlet 24 received
5.1769999999999999: Broken: Cloudlet 23 received
5.5969999999999999: Broken: Cloudlet 29 received
5.9829999999999999: Broken: Cloudlet 21 received
6.0929999999999998: Broken: Cloudlet 25 received
6.457: Broken: Cloudlet 27 received
6.786: Broken: Cloudlet 28 received
6.9219999999999998: Broken: Cloudlet 34 received
6.984: Broken: Cloudlet 26 received
7.0319999999999997: Broken: Cloudlet 31 received
7.1419999999999997: Broken: Cloudlet 22 received
7.336: Broken: Cloudlet 39 received
7.6739999999999997: Broken: Cloudlet 38 received
7.782: Broken: Cloudlet 37 received
8.309: Broken: Cloudlet 38 received
8.3259999999999997: Broken: Cloudlet 33 received
8.4359999999999996: Broken: Cloudlet 32 received
8.52: Broken: Cloudlet 36 received
9.0189999999999997: Broken: Cloudlet 35 received
9.0189999999999997: Broken: All cloudlets executed. Finishing...
9.0189999999999997: Broken: Destroying VM #0
9.0189999999999997: Broken: Destroying VM #1
9.0189999999999997: Broken: Destroying VM #2
9.0189999999999997: Broken: Destroying VM #3
9.0189999999999997: Broken: Destroying VM #4
9.0189999999999997: Broken: Destroying VM #5
9.0189999999999997: Broken: Destroying VM #6

```

| PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | SQL CONSOLE | | | | |
|--|--------|---------------|----------------|-----------|-------------|------------|-------------|---------|---|
| Cloudlet: 35 | | | | | | | | | |
| Average Waiting Time on Total: 37.1175 | | 74.4 | | 77.33 | | | | | |
| Average Turn Around Time on Total: 39.850747 | | | | | | | | | |
| Cloudlet ID | ID | STATUS | Data center ID | VM ID | Time | Start Time | Finish Time | User ID | |
| 0 | 0 | SUCCESS | 1.15 | 2 | 0 | 1.15 | 0.2 | 1.35 | 4 |
| 8 | 0 | SUCCESS | 3 | 8 | 1 | 1.54 | 0.2 | 1.74 | 4 |
| 4 | 1 | SUCCESS | 2.69 | 2 | 4 | 1.67 | 0.2 | 1.87 | 4 |
| 6 | 2 | SUCCESS | 4.36 | 3 | 6 | 1.89 | 0.2 | 2.09 | 4 |
| 1 | 3 | SUCCESS | 6.25 | 2 | 1 | 2.03 | 0.2 | 2.23 | 4 |
| 9 | 4 | SUCCESS | 8.28 | 3 | 9 | 2.08 | 0.2 | 2.28 | 4 |
| 3 | 5 | SUCCESS | 10.36 | 2 | 3 | 2.21 | 0.2 | 2.41 | 4 |
| 10 | 6 | SUCCESS | 12.57 | 2 | 0 | 1.23 | 1.35 | 2.58 | |
| 4 | 7 | SUCCESS | 13.799999 | 2 | 2 | 2.61 | 0.2 | 2.81 | 4 |
| 2 | 8 | SUCCESS | 15.41 | 3 | 7 | 2.91 | 0.2 | 3.11 | 4 |
| 7 | 9 | SUCCESS | 16.41 | 2 | 5 | 2.96 | 0.2 | 3.16 | 4 |
| 5 | 10 | SUCCESS | 19.32 | 22.279999 | 1 | 1.24 | 2.23 | 3.47 | |
| 11 | 11 | SUCCESS | 23.519999 | 3 | 9 | 1.4 | 2.28 | 3.67 | 4 |
| 4 | 12 | SUCCESS | 24.919998 | 2 | 4 | 1.91 | 1.87 | 3.79 | |
| 14 | 13 | SUCCESS | 26.829998 | 3 | 8 | 2.54 | 1.74 | 4.28 | |
| 4 | 18 | SUCCESS | 26.829998 | 3 | | | | | |
| 4 | 19 | SUCCESS | 26.829998 | 3 | | | | | |

Result:

Thus a scheduling algorithm in CloudSim has been implemented and executed successfully.

Expt No: 05

Date:

Use Vercel to launch the web applications.

Aim:

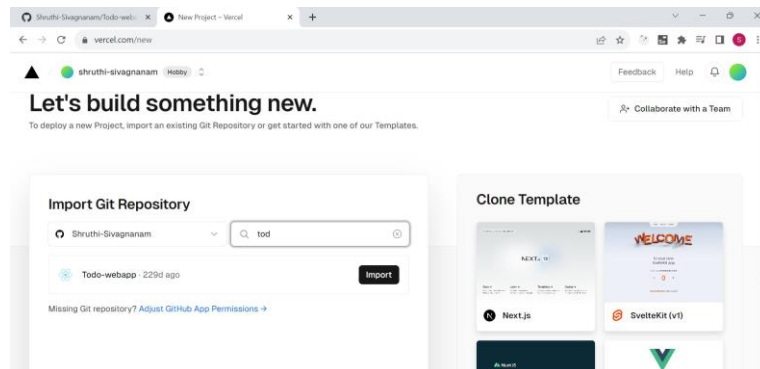
To use Vercel web service to launch the web application.

Procedure:

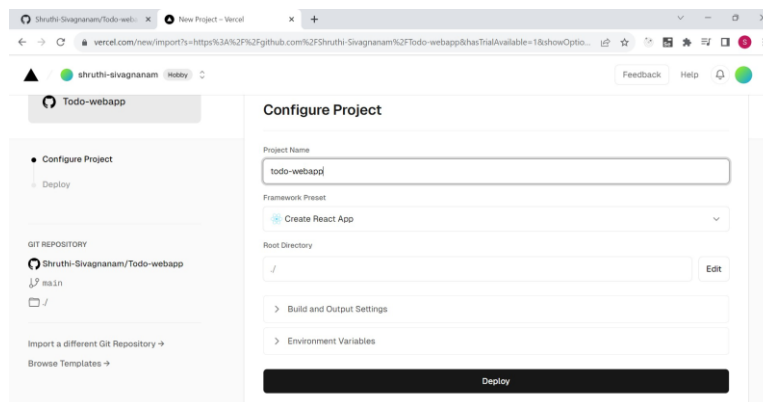
Step 1 : Create a web application (here to-do web application is developed in react). Push the project into a GitHub repo.

Step 2 : Sign up in Vercel.

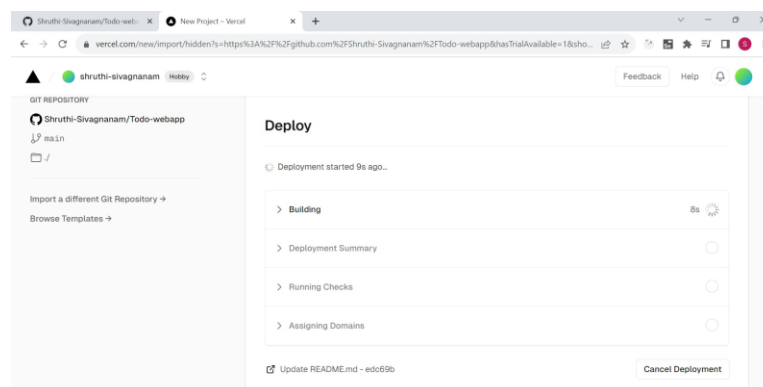
Step 3 : Create a new Project, and link the repo with this Vercel project.



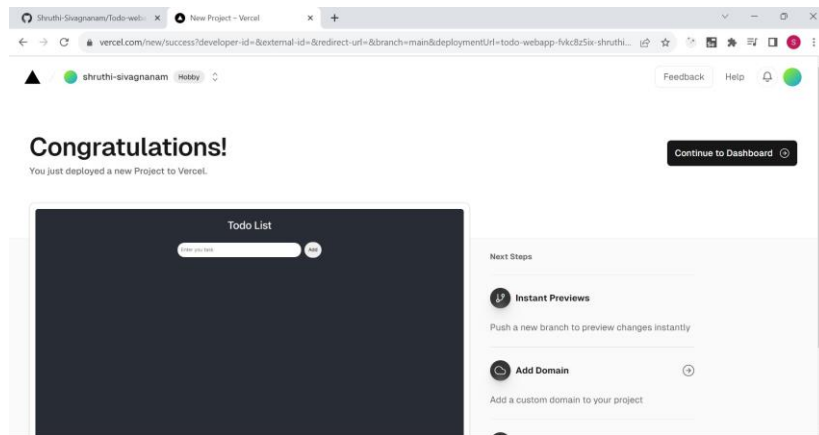
Step 4 : Configure the project details and click Deploy.



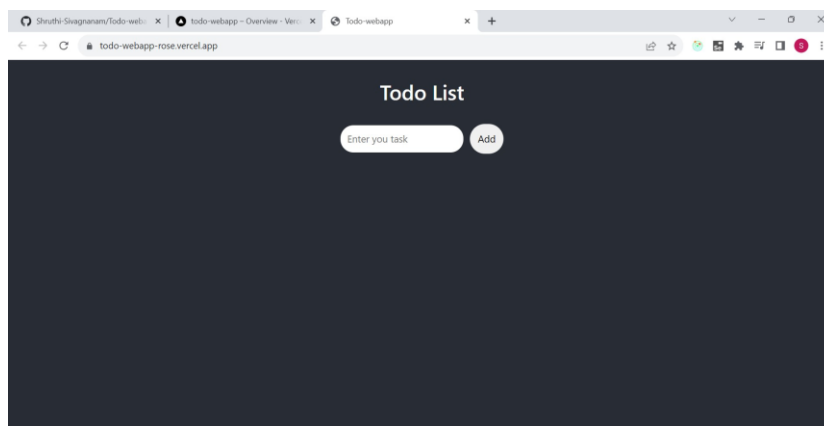
Step 5 : It takes time to deploy the project.



Step 6 : The project is deployed successfully.



Step 7 : The link is available to use.



Result :

Thus, a to-do web application was developed and deployed to the Vercel.

Expt No: 06

Date:

Find a procedure to transfer the files from one virtual machine to another virtual machine

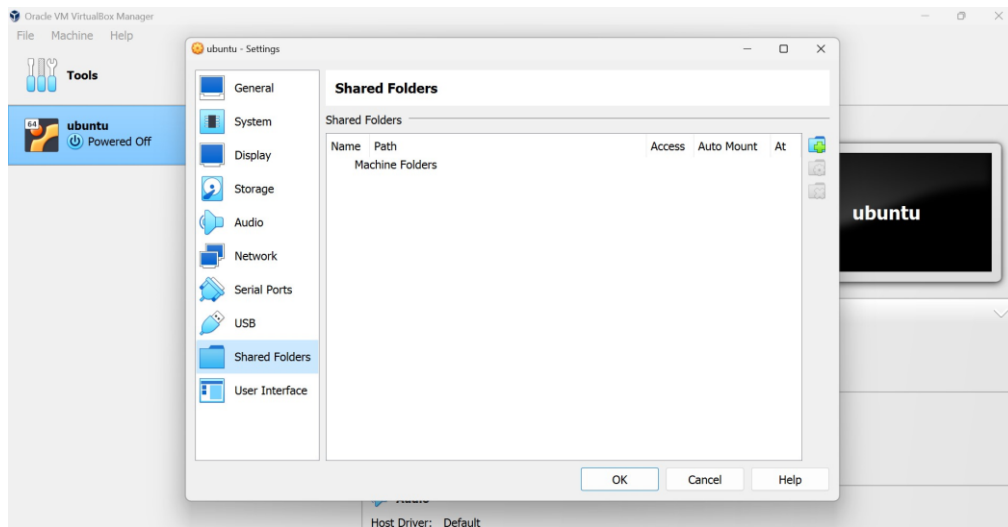
Aim:

To transfer a file or folder from one virtual machine to another virtual machine.

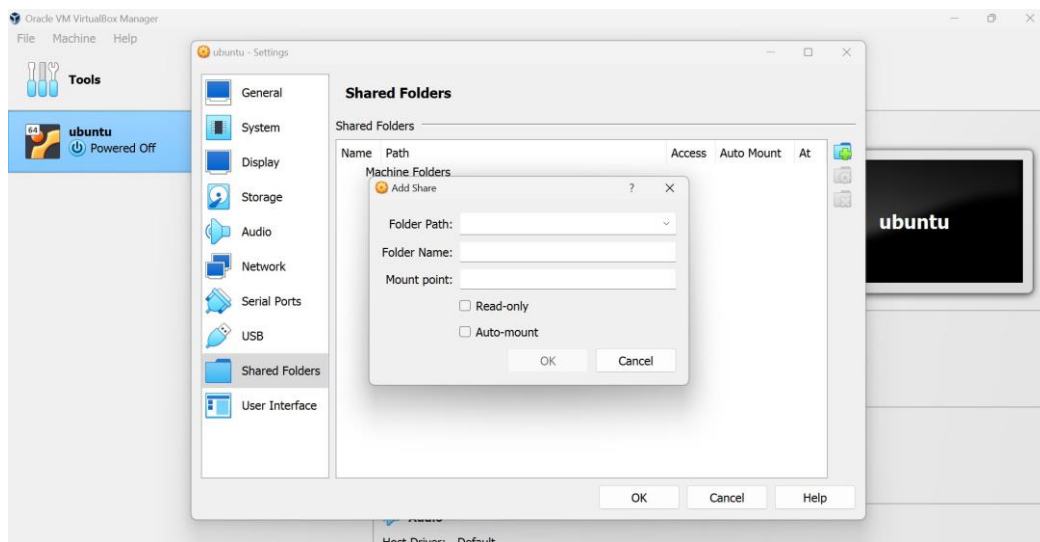
Procedure:

Step 1 : Create a folder or file in the host machine.

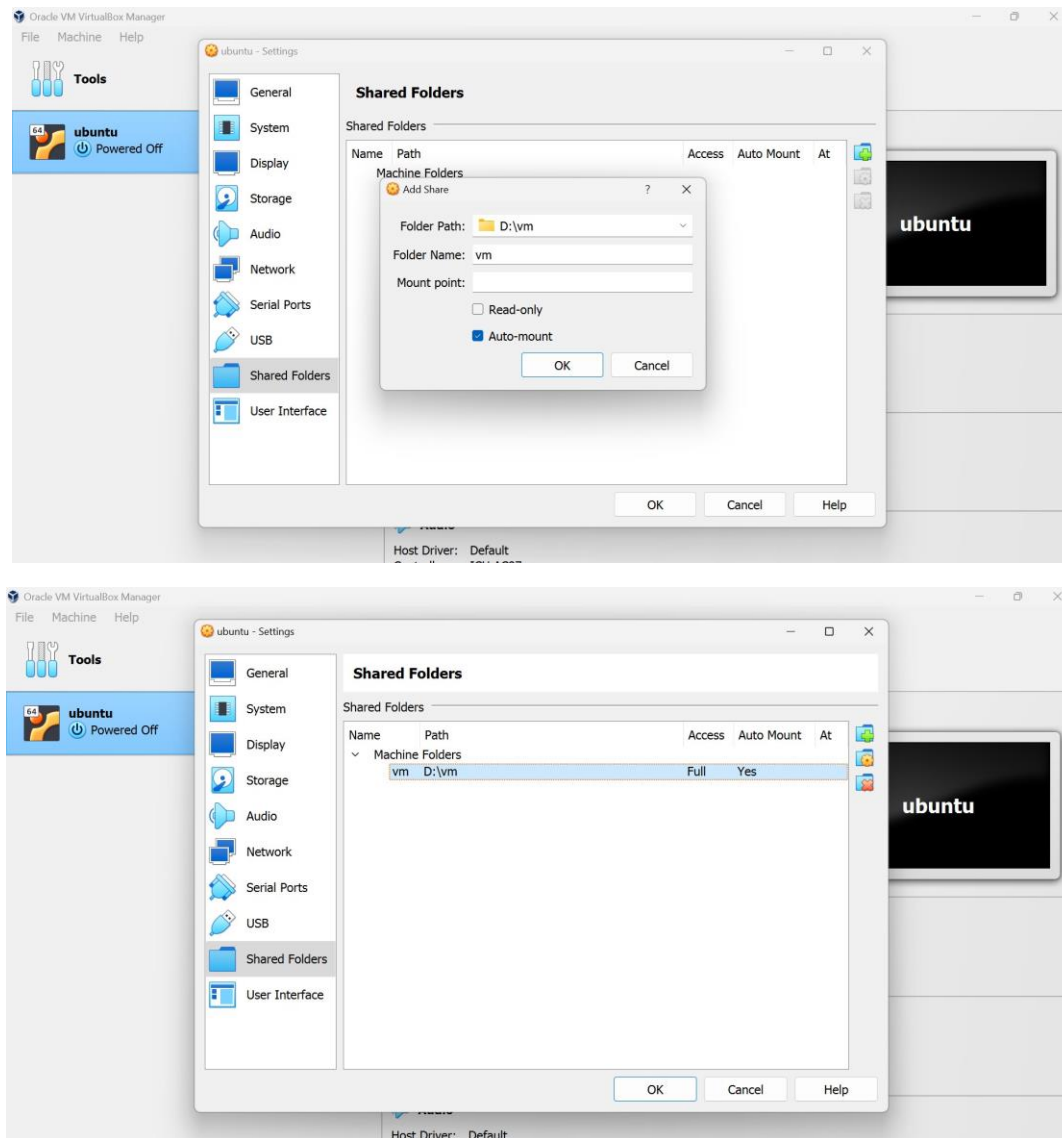
Step 2 : Open VirtualBox. Go to Machine > Settings > Shared Folder



Step 3 : Click the enable file icon in the right corner. A dialog box is opened as shown below.



Step 4 : Link the desired folder in the virtual machine.



Step 5 : Open the virtual machine.

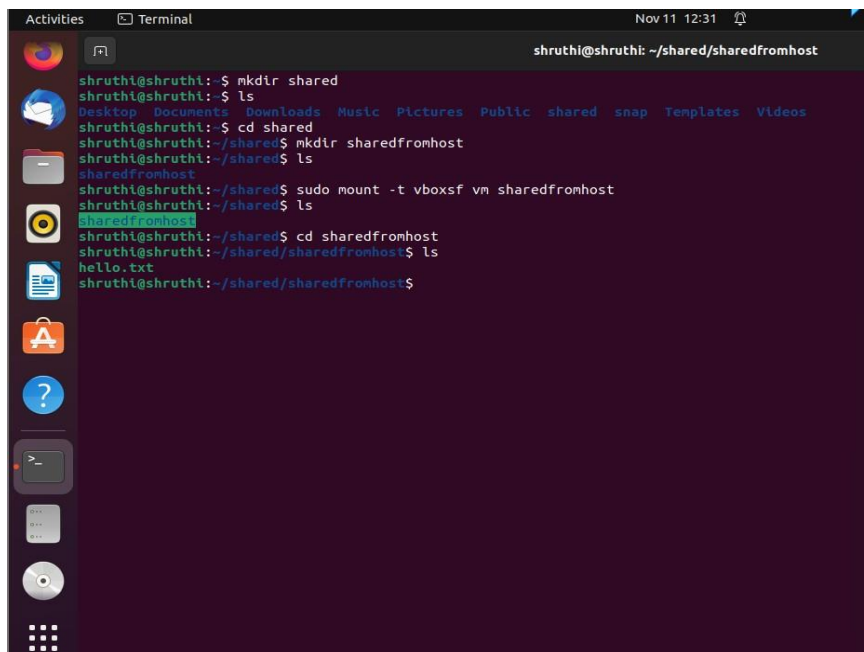
Step 6 : Open the terminal and execute the following commands.

```

shruthi@shruthi:~$ mkdir shared
shruthi@shruthi:~$ ls
Desktop Documents Downloads Music Pictures Public shared snap Templates Videos
shruthi@shruthi:~$ cd shared
shruthi@shruthi:~/shared$ mkdir sharedfromhost
shruthi@shruthi:~/shared$ ls
sharedfromhost
shruthi@shruthi:~/shared$ sudo mount -t vboxsf vm sharedfromhost
shruthi@shruthi:~/shared$ ls
sharedfromhost
shruthi@shruthi:~/shared$

```


Step 7 : The folder is shared successfully.

A screenshot of a Linux terminal window with a dark purple background. The window title is "Terminal" and the top bar shows "Nov 11 12:31". The user is "shruthi@shruthi" and the current directory is "~/shared/sharedfromhost". The terminal shows the following commands and output:

```
shruthi@shruthi:~$ mkdir shared
shruthi@shruthi:~$ ls
Desktop Documents Downloads Music Pictures Public shared snap Templates Videos
shruthi@shruthi:~$ cd shared
shruthi@shruthi:~/shared$ mkdir sharedfromhost
shruthi@shruthi:~/shared$ ls
sharedfromhost
shruthi@shruthi:~/shared$ sudo mount -t vboxsf vm sharedfromhost
shruthi@shruthi:~/shared$ ls
sharedfromhost
shruthi@shruthi:~/shared$ cd sharedfromhost
shruthi@shruthi:~/shared/sharedfromhost$ ls
hello.txt
shruthi@shruthi:~/shared/sharedfromhost$
```

Result:

Thus, a folder is transferred from host os to virtual machine successfully.

Expt No: 07

Date:

Find a procedure to launch virtual machine using trystack (Online Openstack Demo Version)

Aim:

To find a procedure to launch the virtual machine using trystack (Online Openstack Demo Version).

Procedure:

Step 1 : Create a network

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

Step 2 : Create an Instance

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

- Click Launch.
- 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 a Router

- Go to Network > Routers and then click Create Router.
- Fill Router Name for example router1 and then click Create router.
- Click on your router name link, for example router1, Router Details page.
- Click Set Gateway button in upper right:
 - Select External networks with external.
 - Then OK.
- Click Add Interface button.
 - Select Subnet with the network that you have been created in Step 1.
 - Click Add interface.
- 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

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

Step 5 : Configure Access and Security

- Go to Compute > Access & Security and then open Security Groups tab.
- In default row, click Manage Rules.
- Click Add Rule, choose ALL ICMP rule to enable ping into your instance, and then click Add.
- Click Add Rule, choose HTTP rule to open HTTP port (port 80), and then click Add.
- Click Add Rule, choose SSH rule to open SSH port (port 22), and then click Add.
- 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.

Result:

Thus, the procedure for creating a virtual machine in trystack was found.

Expt No: 08

Date:

Install Hadoop single node cluster and run simple applications like wordcount.

Aim:

To install Hadoop single node cluster and run simple application like word count in Hadoop.

Hadoop:

Hadoop is an open-source framework from Apache and is used to store process and analyse data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing. It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

Procedure:

Step 1 : Download the Hadoop from

<https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz> .

Step 2 : Unzip the downloaded tar file in the C directory.

Step 3 : Set the environmental variable as “HADOOP_HOME”.

Step 4 : Edit the following file from etc/hadoop/

- core-site.xml

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

- hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/hadoop-3.3.1/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/hadoop-3.3.1/data/datanode</value>
  </property>
</configuration>
```

- mapred-site.xml

```

<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:54311</value>
  </property>
</configuration>

```

Step 5 : Format the NameNode before you can start Hadoop by the following command
“hadoop namenode -format”

Step 6 : Start the Hadoop by the command “start-all.cmd” in /sbin. This command will start all the required Hadoop services, including the NameNode, DataNode, and JobTracker. Wait for a few minutes until all the services are started.

Step 7 : To ensure that Hadoop is properly installed, open a web browser and go to <http://localhost:50070/>. This will launch the web interface for the Hadoop NameNode.

Step 8 : Create a Java project with three classes and link all the external jar file presented in /usr/lib.

WCMapper.java

```

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,
                                                                    Text, Text,
                                                                    IntWritable> {

    public void map(LongWritable key, Text value, OutputCollector<Text,
                                                                    IntWritable> output, Reporter rep) throws IOException
    {
        String line = value.toString();
        for (String word : line.split(" "))
        {
            if (word.length() > 0)
            {

```

```

        output.collect(new Text(word), new IntWritable(1));
    }
}
}
}

```

WCReducer.java

```

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
public class WCReducer extends MapReduceBase implements Reducer<Text,
                                                                    IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> value,
                        OutputCollector<Text, IntWritable> output,
                        Reporter rep) throws IOException
    {
        int count = 0;
        while (value.hasNext())
        {
            IntWritable i = value.next();
            count += i.get();
        }
        output.collect(key, new IntWritable(count));
    }
}

```

WCDriver.java

```

import java.io.IOException;
import org.apache.hadoop.conf.Configured;

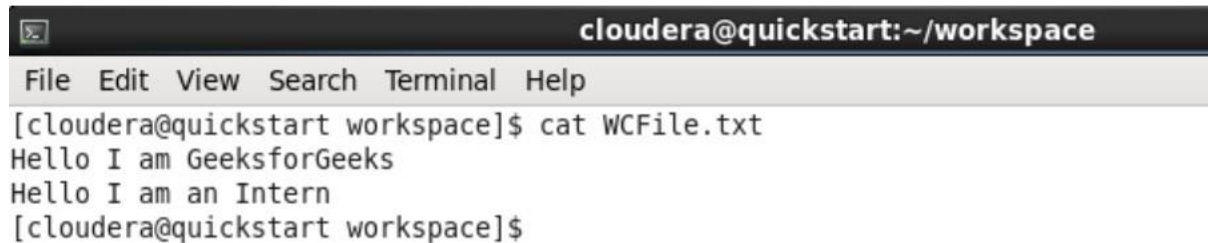
```

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class WCDriver extends Configured implements Tool {
    public int run(String args[]) throws IOException
    {
        if (args.length < 2)
        {
            System.out.println("Please give valid inputs");
            return -1;
        }
        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }
    public static void main(String args[]) throws Exception
    {
        int exitCode = ToolRunner.run(new WCDriver(), args);
```

```
        System.out.println(exitCode);
    }
}
```

Step 9 : Convert this program into jar file.

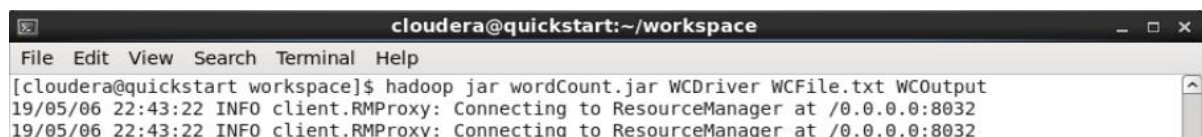
Step 10 : Create a text file(**WCFile.txt**) and move it to HDFS.

A terminal window titled 'cloudera@quickstart:~/workspace' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command '[cloudera@quickstart workspace]\$ cat WCFile.txt' and its output: 'Hello I am GeeksforGeeks' and 'Hello I am an Intern'. The prompt '[cloudera@quickstart workspace]\$' is shown at the bottom.

```
cloudera@quickstart:~/workspace
File Edit View Search Terminal Help
[cloudera@quickstart workspace]$ cat WCFile.txt
Hello I am GeeksforGeeks
Hello I am an Intern
[cloudera@quickstart workspace]$
```

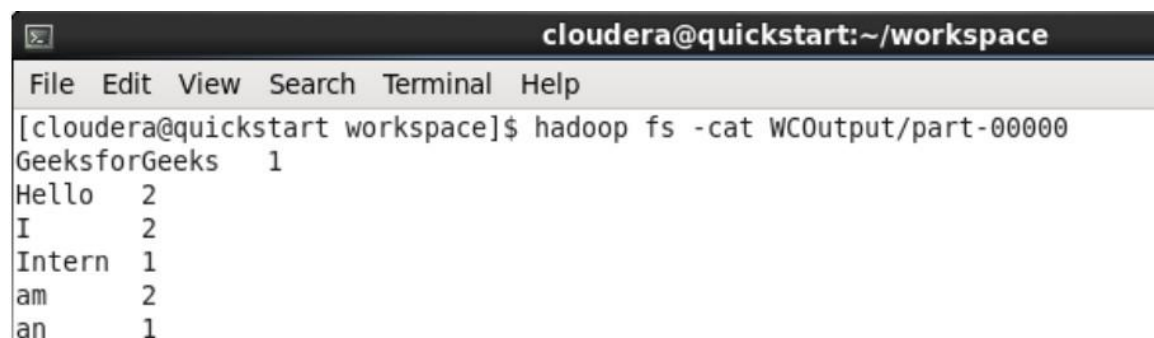
Step 11 : Now, run the command “`hadoop fs -put WCFile.txt WCFile.txt`” to copy the file input file into the HDFS.

Step 12 : Now to run the jar file by writing the code as shown in the screenshot.

A terminal window titled 'cloudera@quickstart:~/workspace' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command '[cloudera@quickstart workspace]\$ hadoop jar wordCount.jar WCDriver WCFile.txt WCOuput' and its output: '19/05/06 22:43:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032' and '19/05/06 22:43:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032'. The prompt '[cloudera@quickstart workspace]\$' is shown at the bottom.

```
cloudera@quickstart:~/workspace
File Edit View Search Terminal Help
[cloudera@quickstart workspace]$ hadoop jar wordCount.jar WCDriver WCFile.txt WCOuput
19/05/06 22:43:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/05/06 22:43:22 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
[cloudera@quickstart workspace]$
```

Step 13 : After Executing the code, you can see the result in *WCOuput* file or by writing following command on terminal.

A terminal window titled 'cloudera@quickstart:~/workspace' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command '[cloudera@quickstart workspace]\$ hadoop fs -cat WCOuput/part-000000' and its output: 'GeeksforGeeks 1', 'Hello 2', 'I 2', 'Intern 1', 'am 2', and 'an 1'. The prompt '[cloudera@quickstart workspace]\$' is shown at the bottom.

```
cloudera@quickstart:~/workspace
File Edit View Search Terminal Help
[cloudera@quickstart workspace]$ hadoop fs -cat WCOuput/part-000000
GeeksforGeeks 1
Hello 2
I 2
Intern 1
am 2
an 1
[cloudera@quickstart workspace]$
```

Result:

Thus, Hadoop has been installed and configured. Simple word count application has been executed in Hadoop successfully.