COIMBATORE INSTITUTE OF TECHNOLOGY

RECORD NOTE

TEAM DETAILS:

AATHEESWARAN M(1934001),

NARESH KUMAR R M(1934025)

SRIRAM S(1934048).

DEPARTMENT	**:** M.SC. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
SUBJECT CODE	**:** 19MAM58
SUBJECT	**:** CLOUD COMPUTING LAB

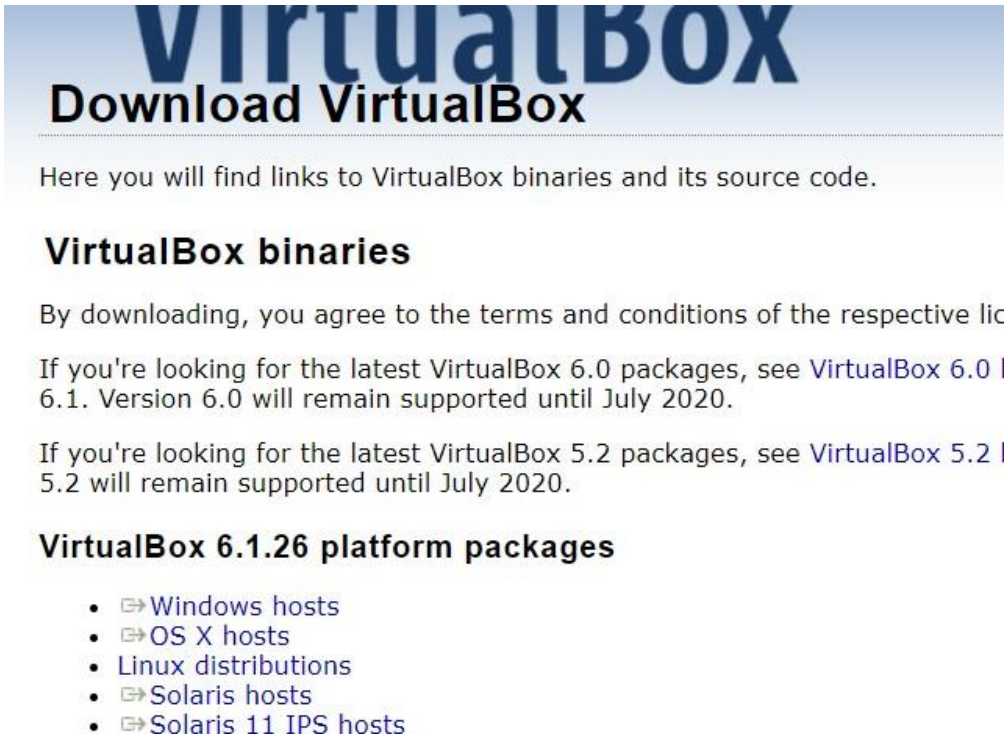
INDEX

Exercise 1

Installing Oracle VirtualBox

Steps:

1. Go to https://www.virtualbox.org/wiki/Downloads to download Oracle virtual box.

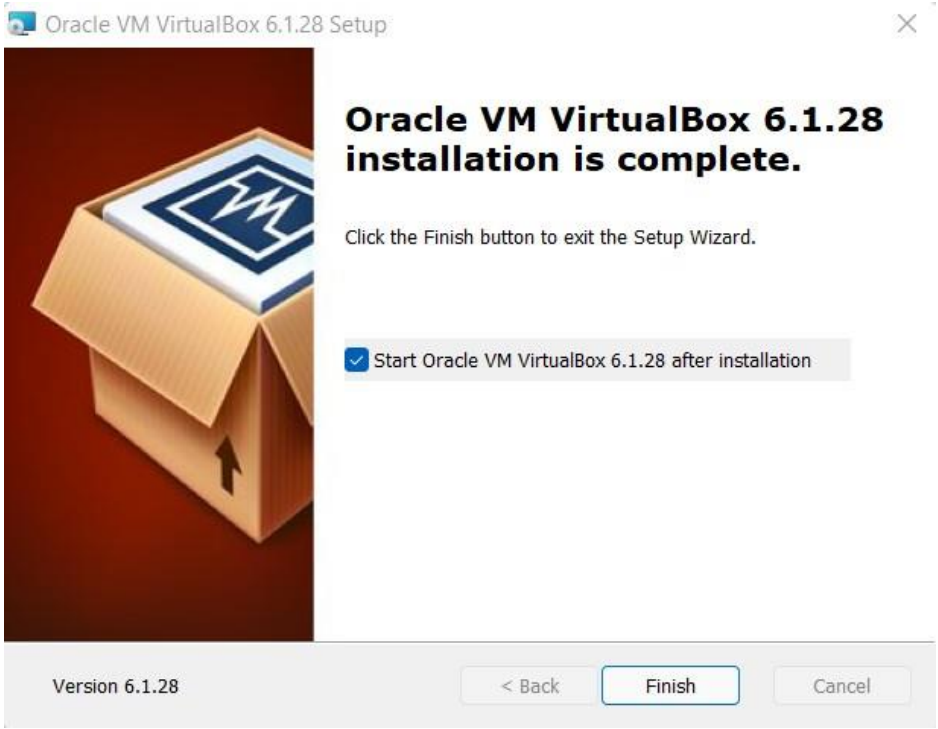


Select your OS

2. Run the file

3. Click Next

4. Leave the default settings and click next to install



Exercise-2

Running a C program on Virtual Machine

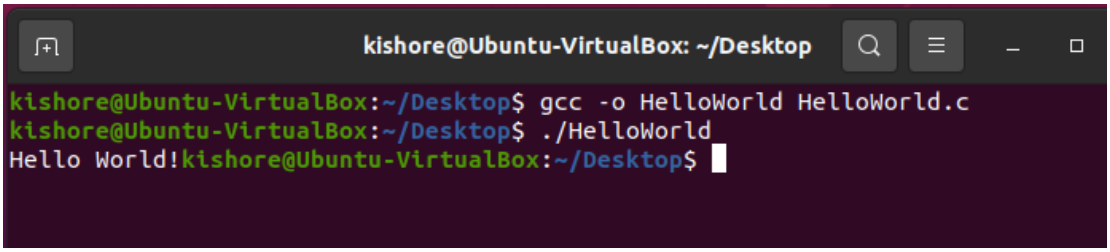
Steps:

Linux:

1. Open text editor in Ubuntu VM.
2. Write the helloworld.c Program

```
#include <stdio.h> void main()
{
printf("Hello World!");
}
```

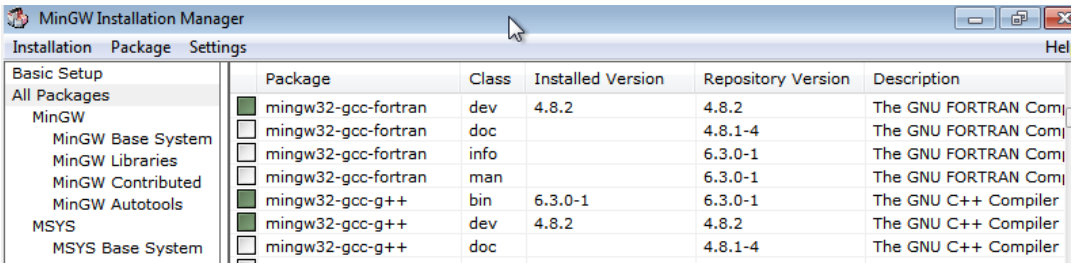
3. Save the file.
4. Compile the file using “gcc -o HelloWorld HelloWorld.c”
5. To run the file Type “./HelloWorld”



Windows

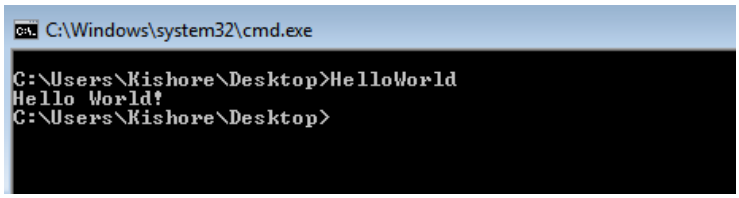
1. Download MinGW GCC Compiler from https://sourceforge.net/projects/mingw/
2. Install C compiler from MinGW

3. In a Text Editor Write the same HelloWorld.c Program.



4. In Command Prompt type “gcc -o HelloWorld HelloWorld.c” to compile the program

5. To run the program type “HelloWorld” in Command prompt

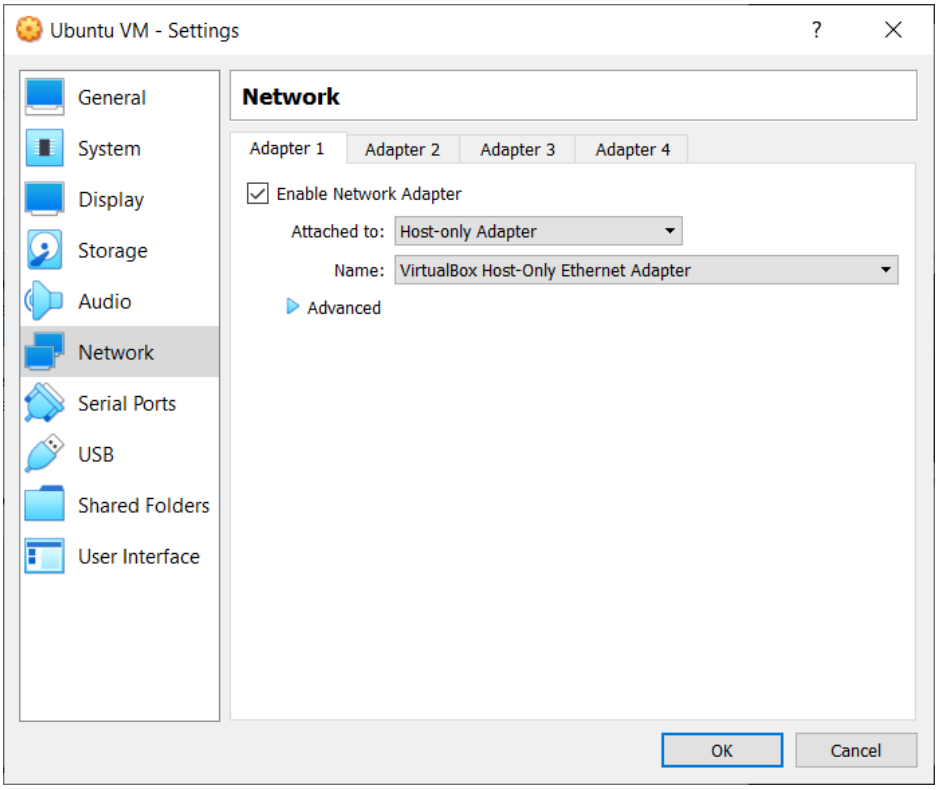


Exercise-3

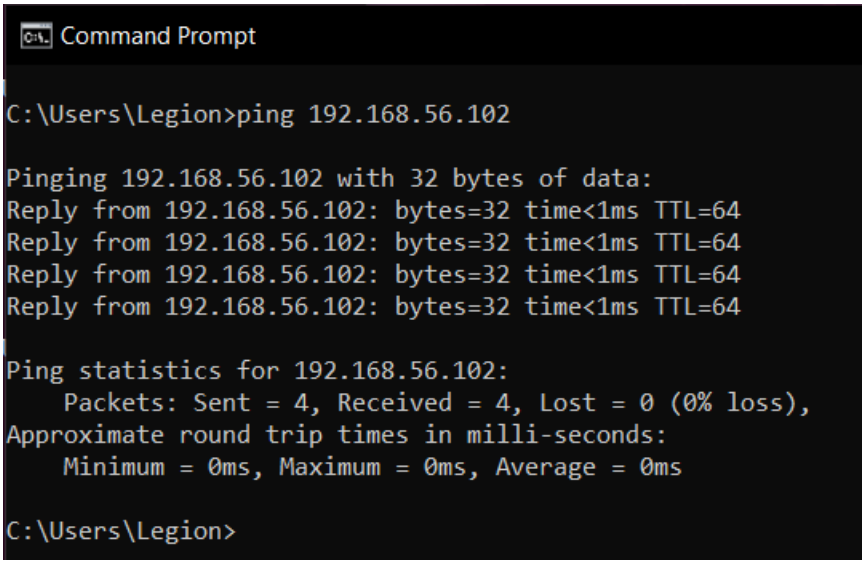
Communication Between Host and Virtual Machine

Steps:

1. Change the adapter Settings to Host-only Adapter



2. Ping the Guest VM from the host



3. Install XAMPP and change the required files
4. Start XAMPP and Create a Database and a Table

+ Options

	Emp_Id	Emp_Name	Emp_Dept	Emp_Role	Emp_Salary
Edit Copy Delete	101	Andres	IT	Lead	75000
Edit Copy Delete	102	Walt	Chemical	Lead	95000
Edit Copy Delete	103	Sergio	IT	Intern	25000
Edit Copy Delete	104	Berlin	Marketing	Manager	50000
Edit Copy Delete	105	Elliot	IT	Testing	45000
Edit Copy Delete	106	Darlene	Sales	Lead	80000
Edit Copy Delete	107	Mike Hunt	HR	Manager	50000
Edit Copy Delete	108	Burnham	DB	Intern	20000

Check all *With selected:* Edit Copy Delete Export

5. Create a new user with IP as % or the IP of guest OS found using ifconfig.

IP:-

```
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::48c:ac8:d986:ba89  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e5:5d:05  txqueuelen 1000  (Ethernet)
        RX packets 46  bytes 8165 (8.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 48  bytes 5784 (5.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

User:-

User name	Host name	Password	Global privileges	User group	Grant	Action
Any	%	No	USAGE		No	Edit privileges Export
Any	localhost	No	USAGE		No	Edit privileges Export
kishore_host	192.168.56.1	Yes	ALL PRIVILEGES		Yes	Edit privileges Export
kishore_test	%	Yes	ALL PRIVILEGES		Yes	Edit privileges Export
kk_2	192.168.1.8	Yes	ALL PRIVILEGES		Yes	Edit privileges Export
pma	localhost	No	USAGE		No	Edit privileges Export
root	127.0.0.1	No	ALL PRIVILEGES		Yes	Edit privileges Export
root	::1	No	ALL PRIVILEGES		Yes	Edit privileges Export
root	localhost	No	ALL PRIVILEGES		Yes	Edit privileges Export

6. Download the MySQL-connector jar file and add it to the project path in Host.

7. Write a Java program to access the database in Guest OS.

```
import java.sql.Connection;
import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.SQLException;
import java.sql.Statement;

public class HostVMconnection {
static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://192.168.56.102:3306/employee"; static final String
USER = "kishore_test";
static final String PASSWORD = "password";

public static void main(String[] args) { Connection conn = null; Statement stmt = null;
try {
                    Class.forName(JDBC_DRIVER); System.out.println("Connecting to a
                    selected database..."); conn = DriverManager.getConnection(DB_URL,
                    USER,

PASSWORD);          System.out.println("Connected database successfully...");

System.out.println("Connecting statement"); stmt = conn.createStatement();
System.out.println("Id\tName\tDept\tRole\tSalary"); String sql = "SELECT
Emp_Id,Emp_Name,Emp_Dept,Emp_Role,Emp_Salary FROM Employee"; ResultSet rs =
stmt.executeQuery(sql);
while(rs.next()) {
int id = rs.getInt("Emp_Id");
String name = rs.getString("Emp_Name"); String dept = rs.getString("Emp_Dept"); String
role = rs.getString("Emp_Role");
int salary = rs.getInt("Emp_Salary");
System.out.println(id + "\t" + name + "\t" + dept + "\t" + role +
"\t" + salary);
```

```
}
rs.close();
}
catch(SQLException se) {
se.printStackTrace();
}catch(Exception e) {
e.printStackTrace();
}finally {
try {
                                        if(stmt!=null)
                                        conn.close();
                                        }catch(SQLException se) {
}
}try {
                                        if(conn!=null)
                                        conn.close();
}catch(SQLException se) { se.printStackTrace();
}
}
}
```

8. Run the program and verify the result.

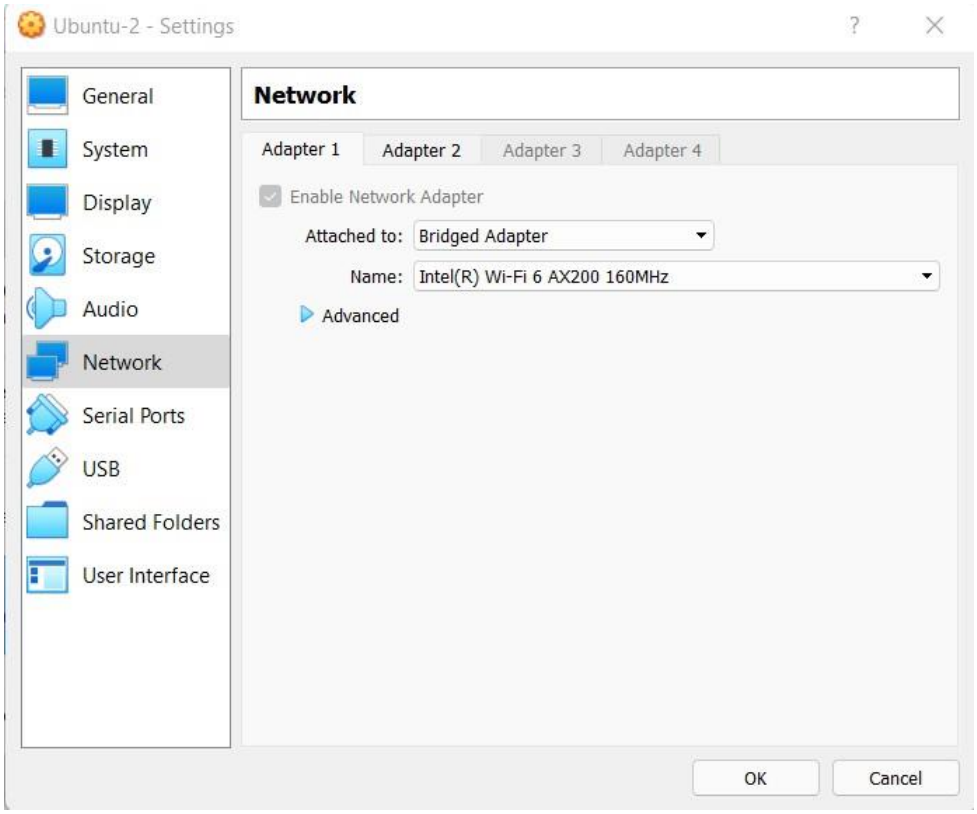
```
Connecting to a selected database...
Connected database successfully...
Connecting statement
Id      Name    Dept    Role    Salary
101     Andres  IT      Lead    75000
102     Walt    Chemical        Lead    95000
103     Sergio  IT      Intern  25000
104     Berlin  Marketing       Manager 50000
105     Elliot  IT      Testing 45000
106     Darlene Sales   Lead    80000
107     Mike Hunt       HR      Manager 50000
108     Burnham DB      Intern  20000
(base) PS C:\Kishore\Studies\Sem 5\Cloud Computing Lab\cloud computing lab>
```

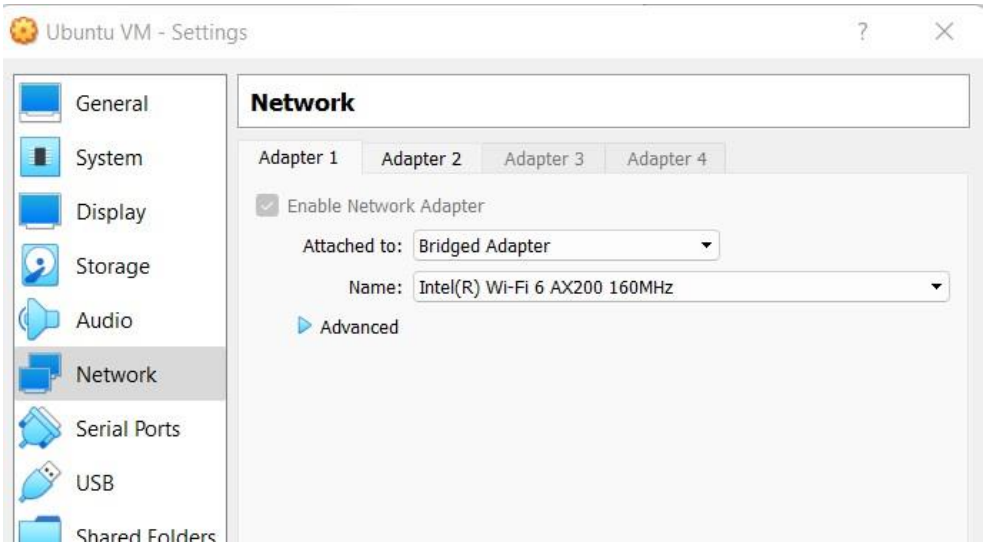
Exercise-4

VM to VM Connection

Steps:

1. After installing two VMs, change their network settings to BridgedAdapter





2. Ping and verify both are able to communicate with each other.

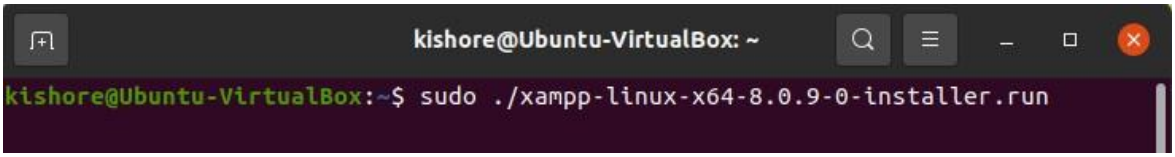
```
kishore@Ubuntu-VirtualBox: ~
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::48c:ac8:d986:ba89  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e5:5d:05  txqueuelen 1000  (Ethernet)
        RX packets 94  bytes 13540 (13.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 63  bytes 7468 (7.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 734  bytes 347707 (347.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 734  bytes 347707 (347.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

kishore@Ubuntu-VirtualBox:~$ ping 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
64 bytes from 192.168.56.103: icmp_seq=1 ttl=64 time=0.423 ms
64 bytes from 192.168.56.103: icmp_seq=2 ttl=64 time=0.607 ms
64 bytes from 192.168.56.103: icmp_seq=3 ttl=64 time=0.591 ms
64 bytes from 192.168.56.103: icmp_seq=4 ttl=64 time=0.552 ms
```

```
kishore@kishore-VirtualBox: ~
kishore@kishore-VirtualBox:~$ ping 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_seq=1 ttl=64 time=0.494 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.353 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.482 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=0.532 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=0.476 ms
64 bytes from 192.168.56.102: icmp_seq=6 ttl=64 time=0.721 ms
64 bytes from 192.168.56.102: icmp_seq=7 ttl=64 time=0.580 ms
64 bytes from 192.168.56.102: icmp_seq=8 ttl=64 time=0.524 ms
64 bytes from 192.168.56.102: icmp_seq=9 ttl=64 time=0.531 ms
64 bytes from 192.168.56.102: icmp_seq=10 ttl=64 time=0.513 ms
64 bytes from 192.168.56.102: icmp_seq=11 ttl=64 time=0.492 ms
64 bytes from 192.168.56.102: icmp_seq=12 ttl=64 time=0.460 ms
64 bytes from 192.168.56.102: icmp_seq=13 ttl=64 time=0.520 ms
64 bytes from 192.168.56.102: icmp_seq=14 ttl=64 time=0.577 ms
64 bytes from 192.168.56.102: icmp_seq=15 ttl=64 time=0.441 ms
64 bytes from 192.168.56.102: icmp_seq=16 ttl=64 time=0.495 ms
64 bytes from 192.168.56.102: icmp_seq=17 ttl=64 time=0.475 ms
64 bytes from 192.168.56.102: icmp_seq=18 ttl=64 time=0.563 ms
64 bytes from 192.168.56.102: icmp_seq=19 ttl=64 time=0.595 ms
64 bytes from 192.168.56.102: icmp_seq=20 ttl=64 time=0.524 ms
```

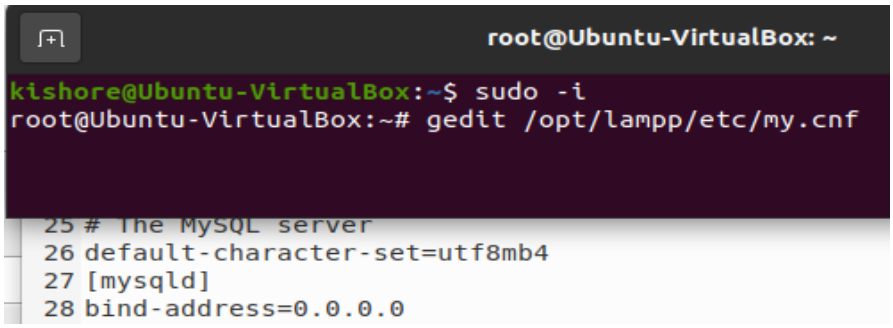
3. Install Java in the VM-B (From which we are going to access the database).

```
kishore@Ubuntu-VirtualBox: ~
kishore@Ubuntu-VirtualBox:~$ sudo apt install openjdk-11-jre-headless
[sudo] password for kishore:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openjdk-11-jre-headless is already the newest version (11.0.11+9-0ubuntu2~20.04)
.
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi
  libgstreamer-plugins-bad1.0-0 libva-wayland2
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 300 not upgraded.
kishore@Ubuntu-VirtualBox:~$
```

4. Install a Code editor or IDE in VM-B like VS code (or) Eclipse.
5. Download the XAMPP package in the VM-A (Where we are going to create the Database and Table).



6. Edit my.cnf file and start XAMPP



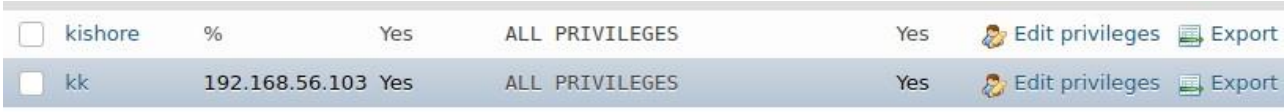
7. Create a Database and Table using phpmyadmin
8. Insert records into the table

+ Options

name	id	role	salary
subash	10	webdev	50000
kishore	11	analyst	35000
jeeva	13	manager	55000
sanjay	14	lead	60000

9. In both the VMs change the second network adapter to Host-only-adapter.

10. Create a user in the PHPMyAdmin with IP either as “%” or the IP of the VM-B



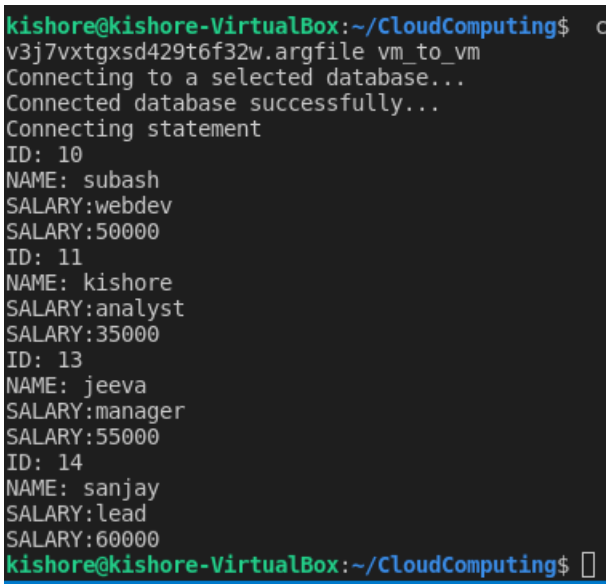
kishore	%	Yes	ALL PRIVILEGES	Yes	Edit privileges	Export
kk	192.168.56.103	Yes	ALL PRIVILEGES	Yes	Edit privileges	Export

11. Download the MySQL-connector-java and add it to the project path where the code is present in VM-B.
12. Write the Java program in VM-B to access the database from VM-A.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Statement; public class vm_to_vm {
static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://192.168.56.102:3306/Employee-1"; static final String USER = "kishore";
static final String PASSWORD = "password"; public static void main(String[] args) {
Connection conn = null; Statement stmt = null; try {
        Class.forName(JDBC_DRIVER); System.out.println("Connecting to a selected database..."); conn = DriverManager.getConnection(DB_URL, USER, PASSWORD);
        System.out.println("Connected database successfully...");
        System.out.println("Connecting statement");
        stmt = conn.createStatement();

        String sql = "SELECT id,name,role,salary FROM emp"; ResultSet rs = stmt.executeQuery(sql);

        while(rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("name"); String role = rs.getString("role");
        int salary = rs.getInt("salary"); System.out.println("ID: "+id);
        System.out.println("NAME: "+name);
        System.out.println("SALARY:"+role);
        System.out.println("SALARY:"+salary);
        }
        rs.close();
        }
        catch(SQLException se) {
        se.printStackTrace();
        }catch(Exception e) {
        e.printStackTrace();
        }finally {
```

```
try {                    }catch(SQLException se) {


                         if(conn!=null)
                         conn.close();
}
}try {                   }catch(SQLException se) { se.printStackTrace();
                         }
if(stmt!=null)           }
conn.close();            }
```

13. Run the program to see the results



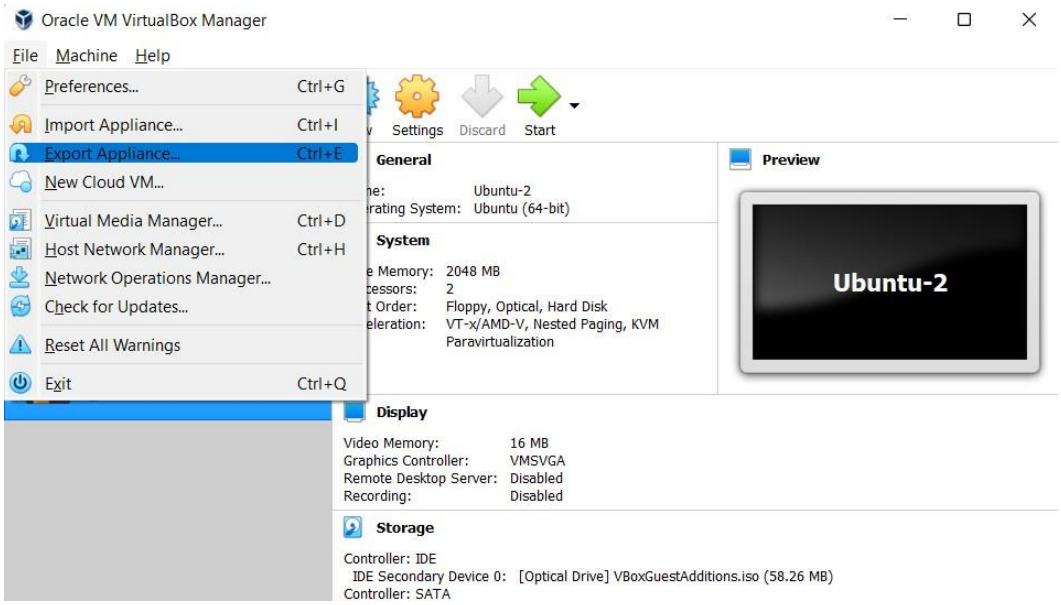
Exercise-5

Exporting VirtualBox VM

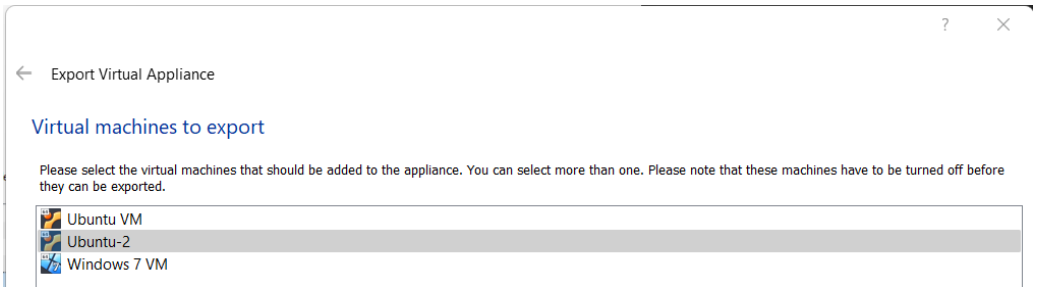
Steps:

1. Open VirtualBox

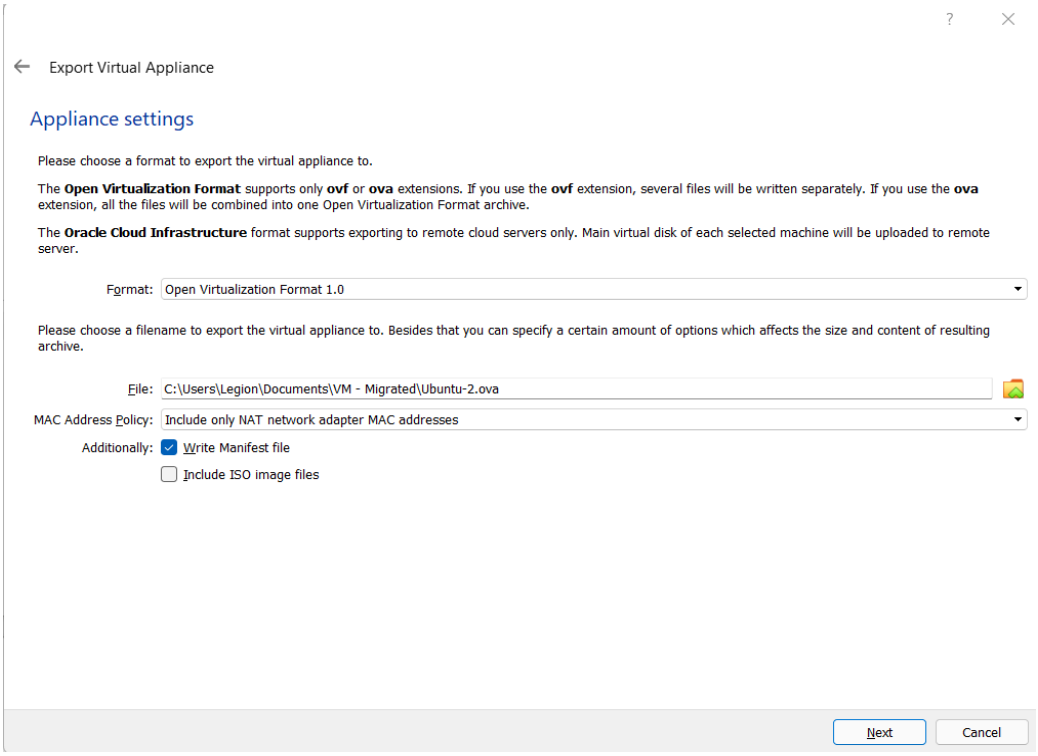
2. Click on File --> Export Appliance



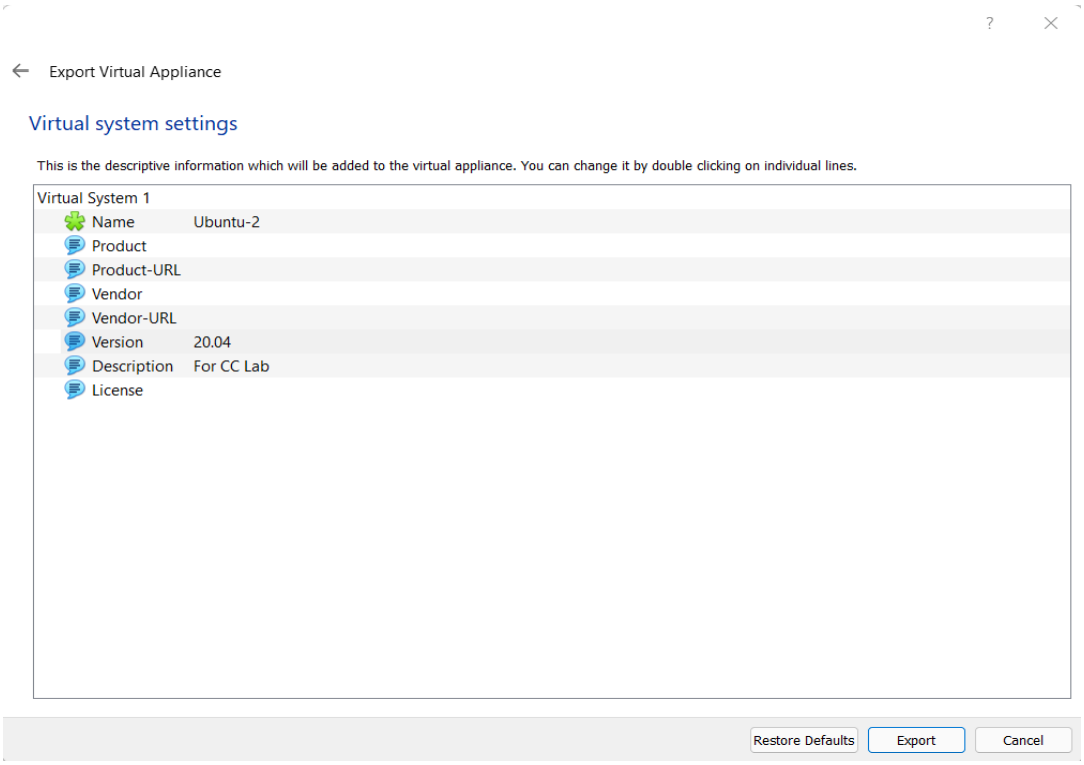
3. Select the VM you want to export



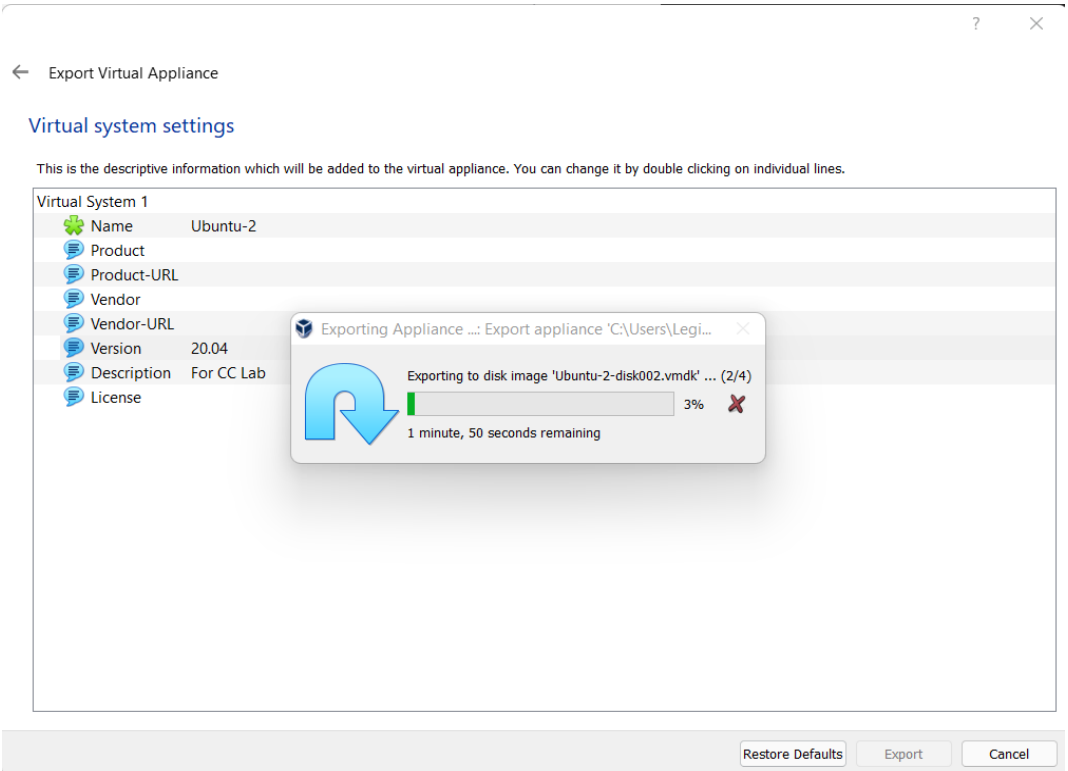
4. Select Location to store the exported VM and click Next



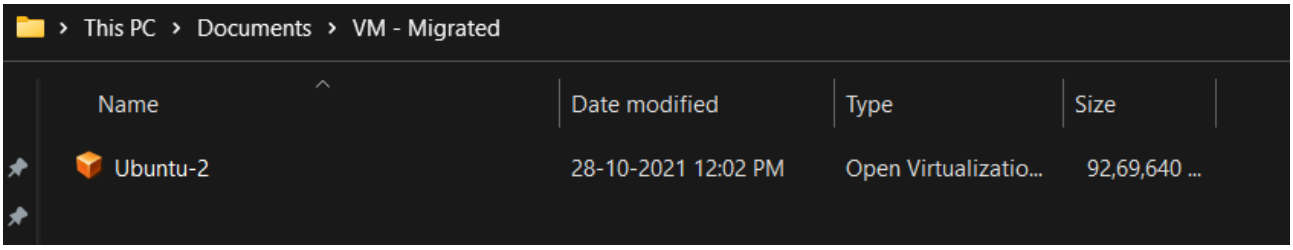
5. Add Description if wanted and click export to store the file in the location specified



6. Wait for the process to complete



7. Check the Location to see the exported .ova file.



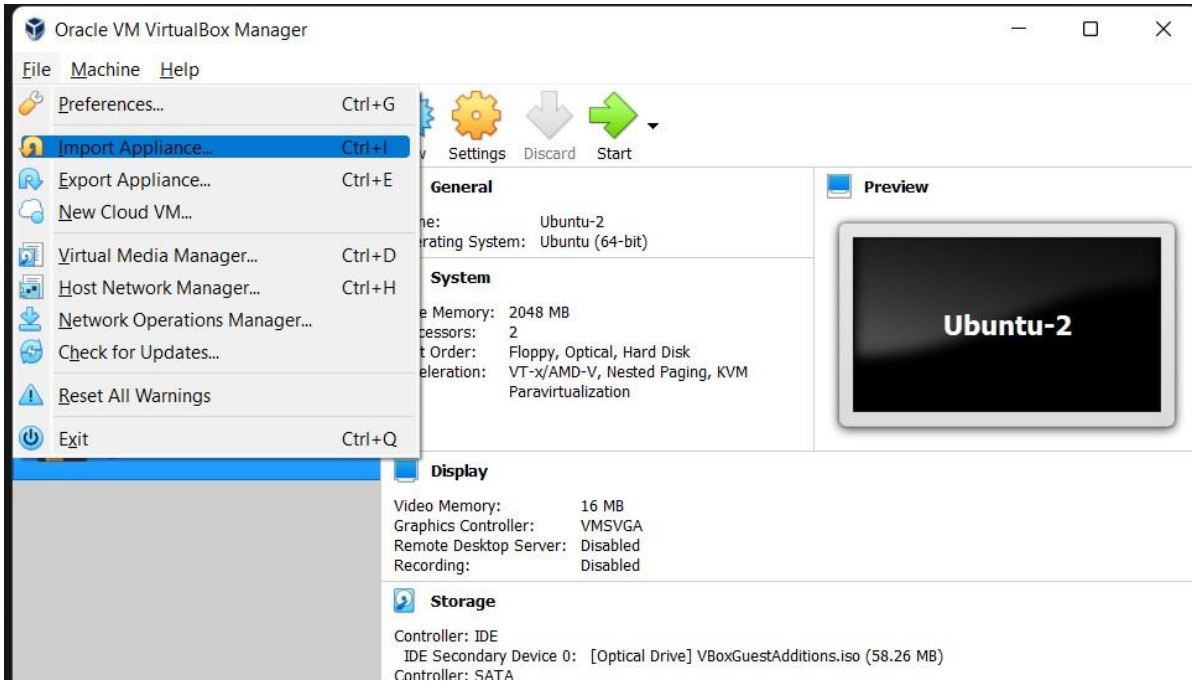
Exercise-6

Importing VirtualBox VM

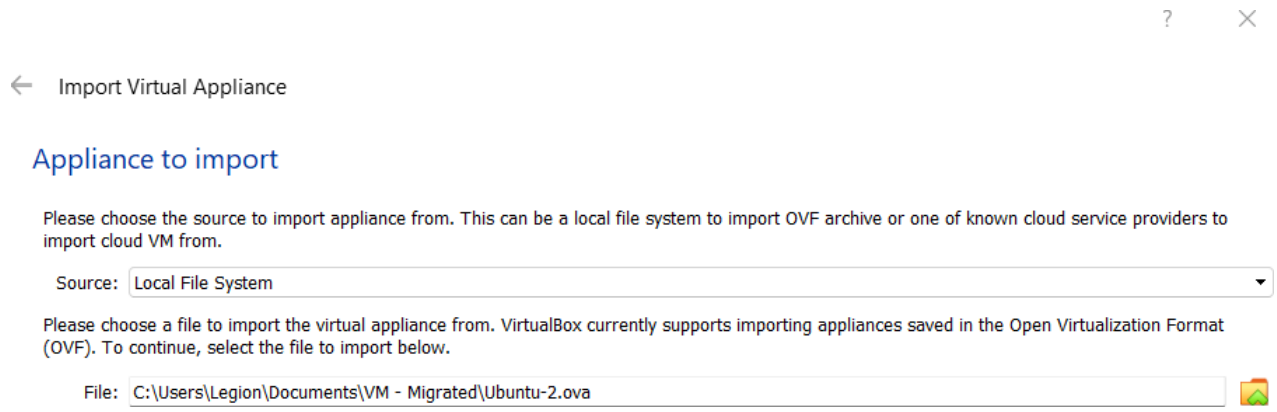
Steps:

1. Start VirtualBox

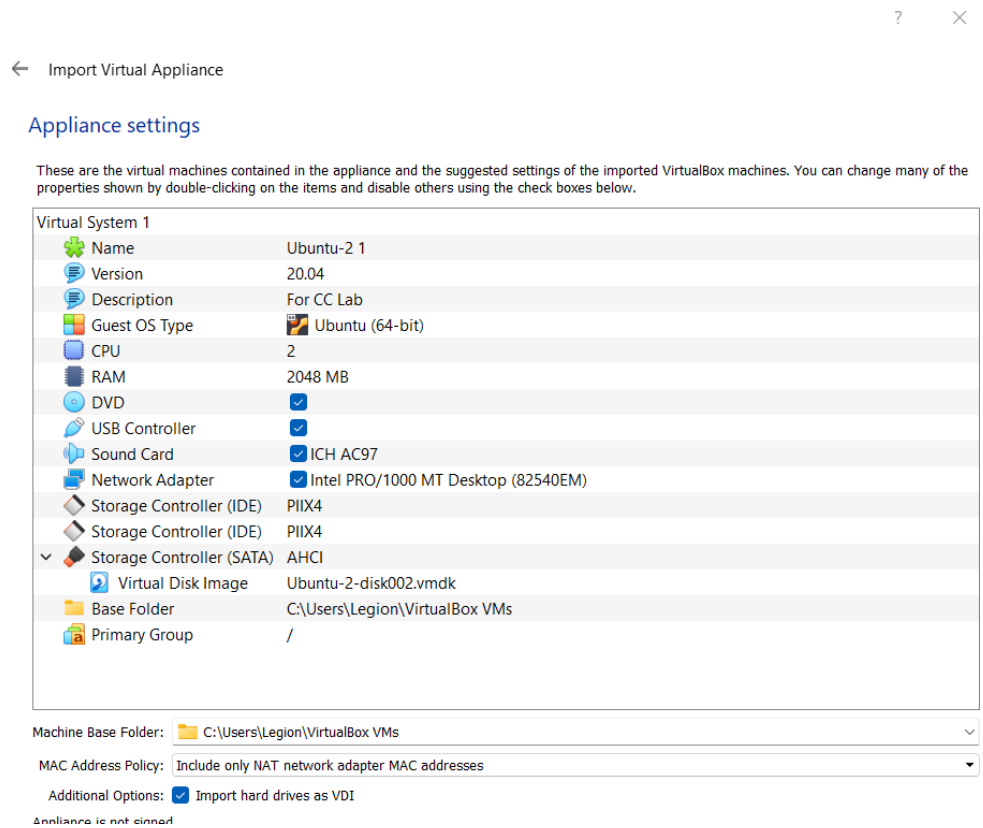
2. Click File --> Import Appliance



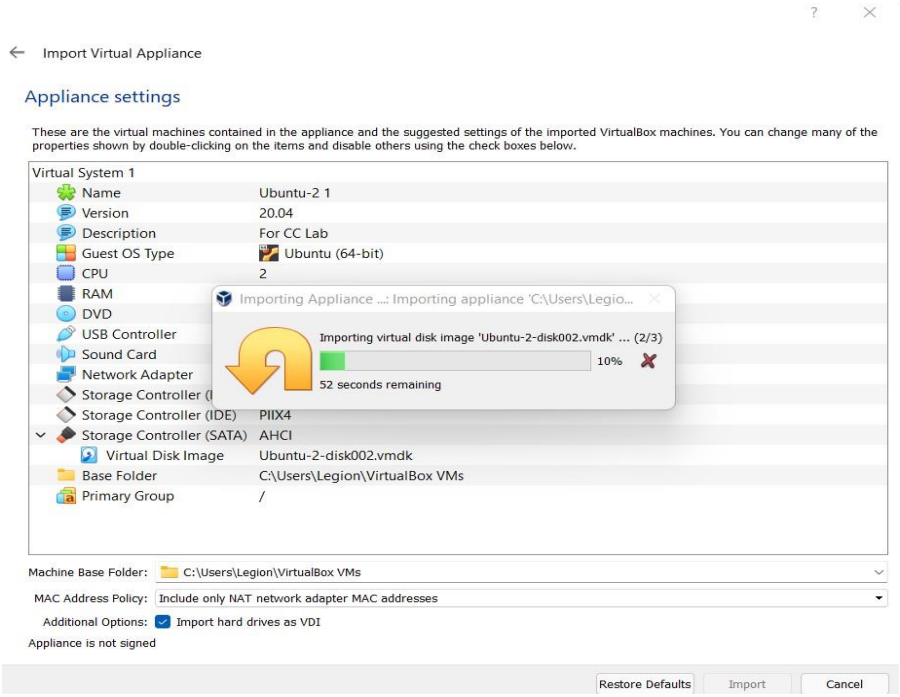
3. Select the OVA file and click next



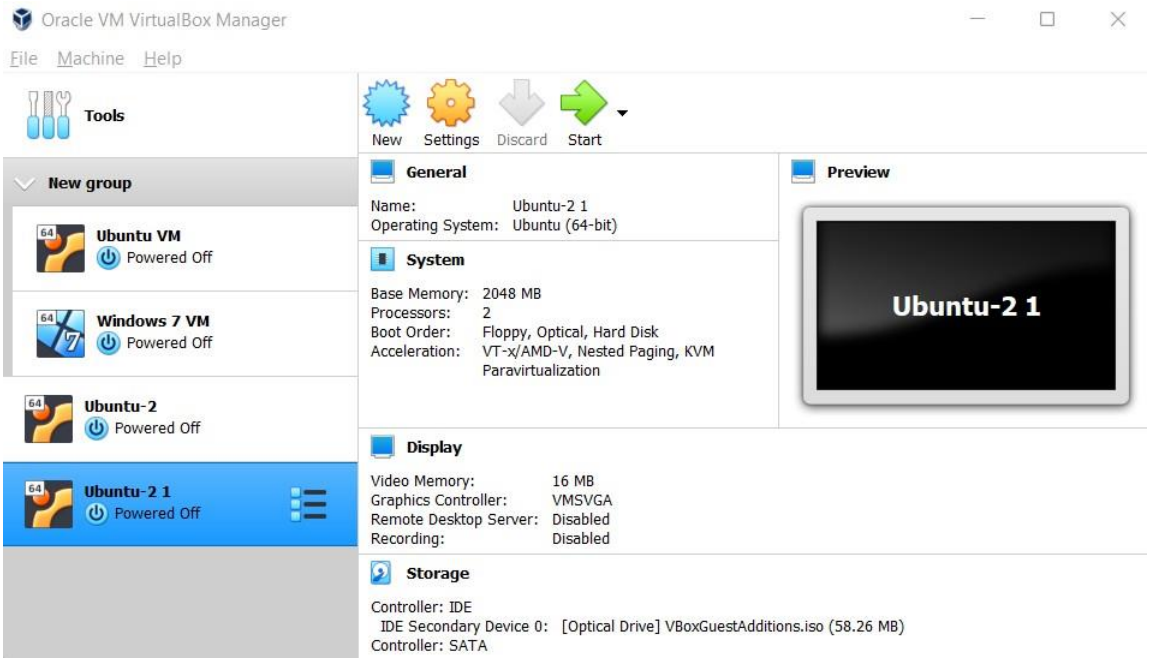
4. Leave the default settings and Click '**import**'.



5. Wait for the import to finish..



6. The machine will be seen on the VirtualBox homepage.

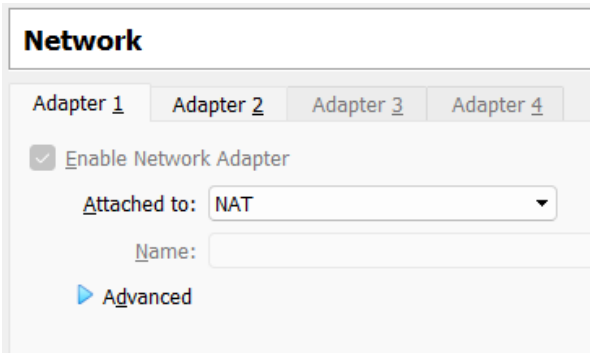


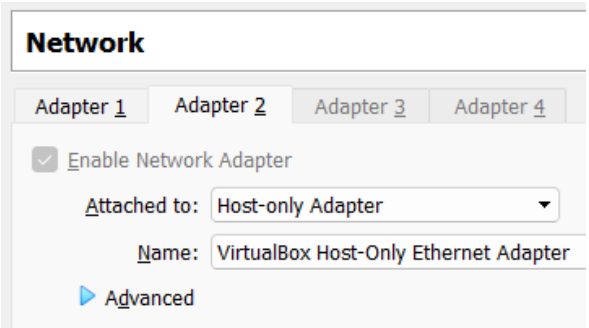
Exercise-7

Creating a shared folder

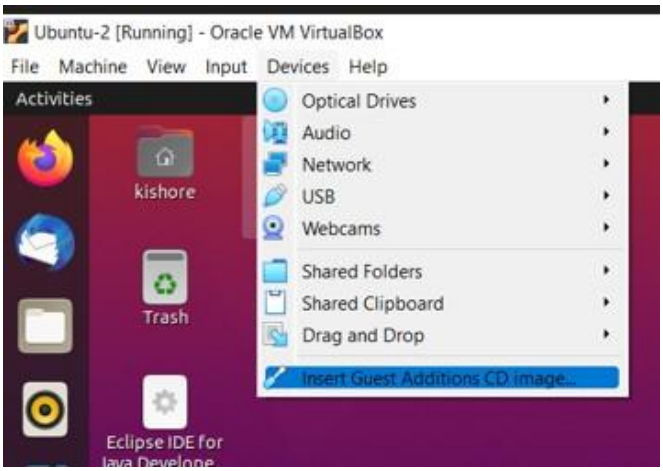
Steps:

1. Configuring the network settings

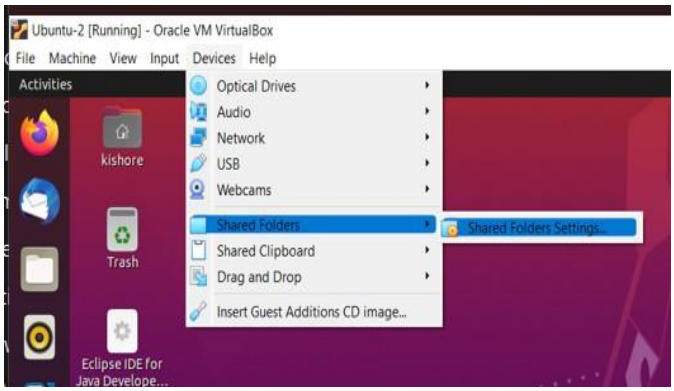




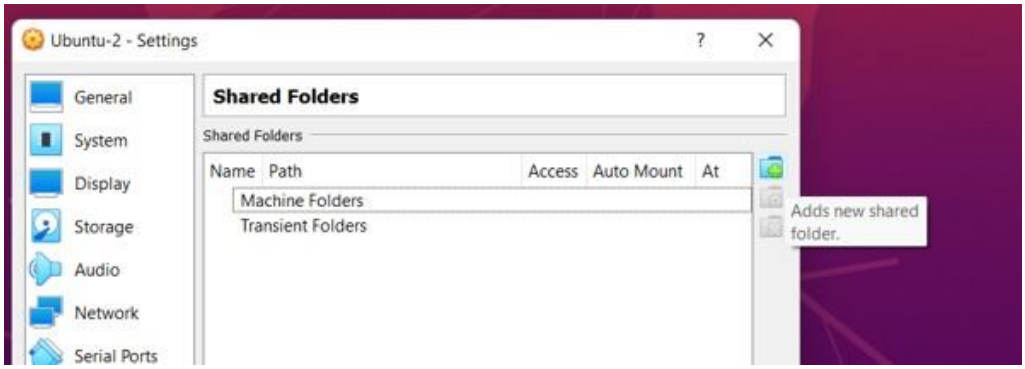
2. Install Guest addition images



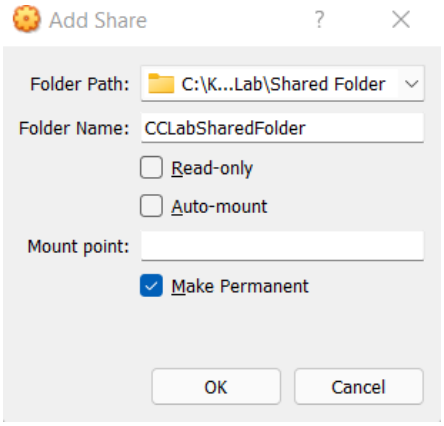
3. Go to shared folder settings



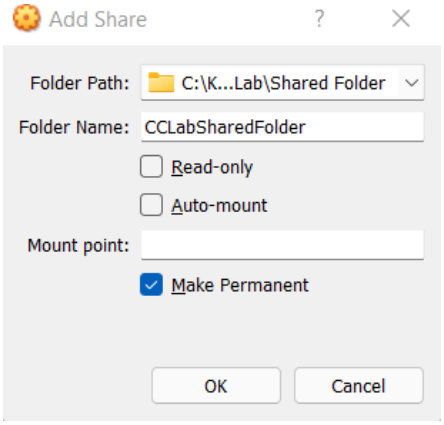
4. Click on Add new Shared folder.



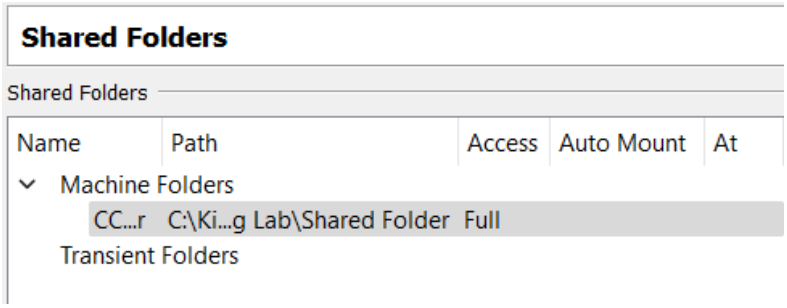
5. Select Folder Path and give it a name. You can also make it read-only, auto-mountand permanent by checking the respective boxes



6. Select Folder Path and give it a name. You can also make it read-only, auto-mount and permanent by checking the respective boxes



7. Permanent folder will be under machine folders, if not under Transient folder

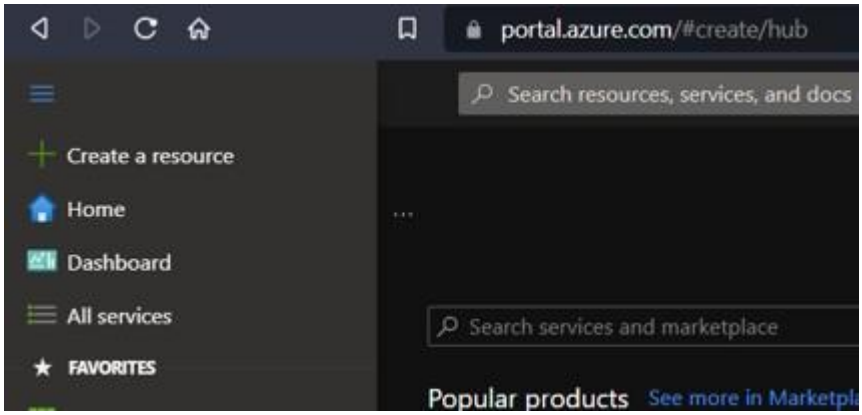


Exercise-8

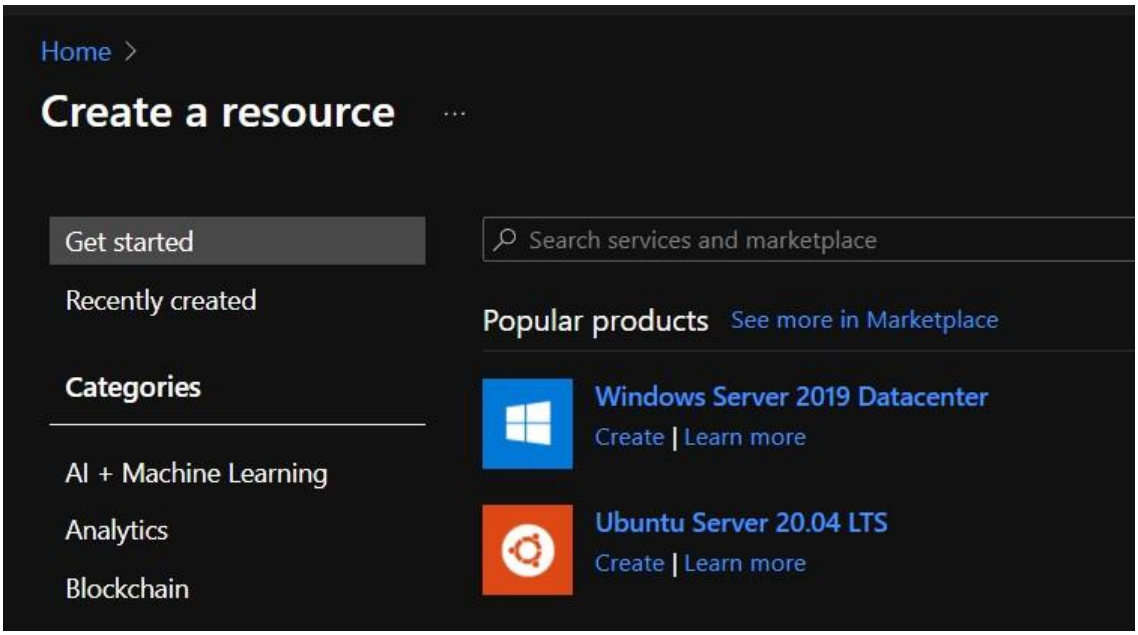
Creating a VM on Azure

Steps:

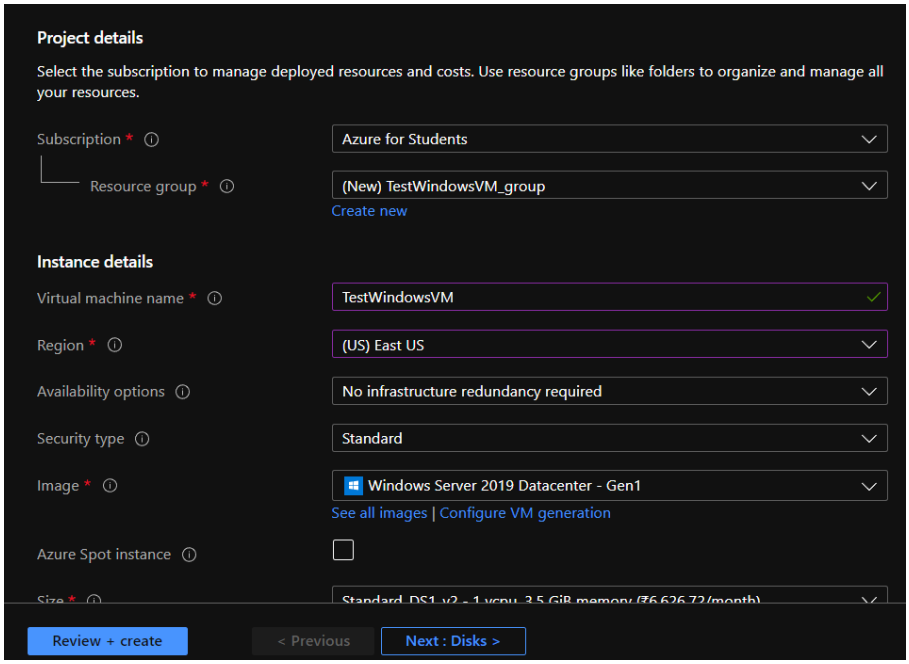
1. Click on the three horizontal bars to see the create resource icon.
2. Click on create resource..



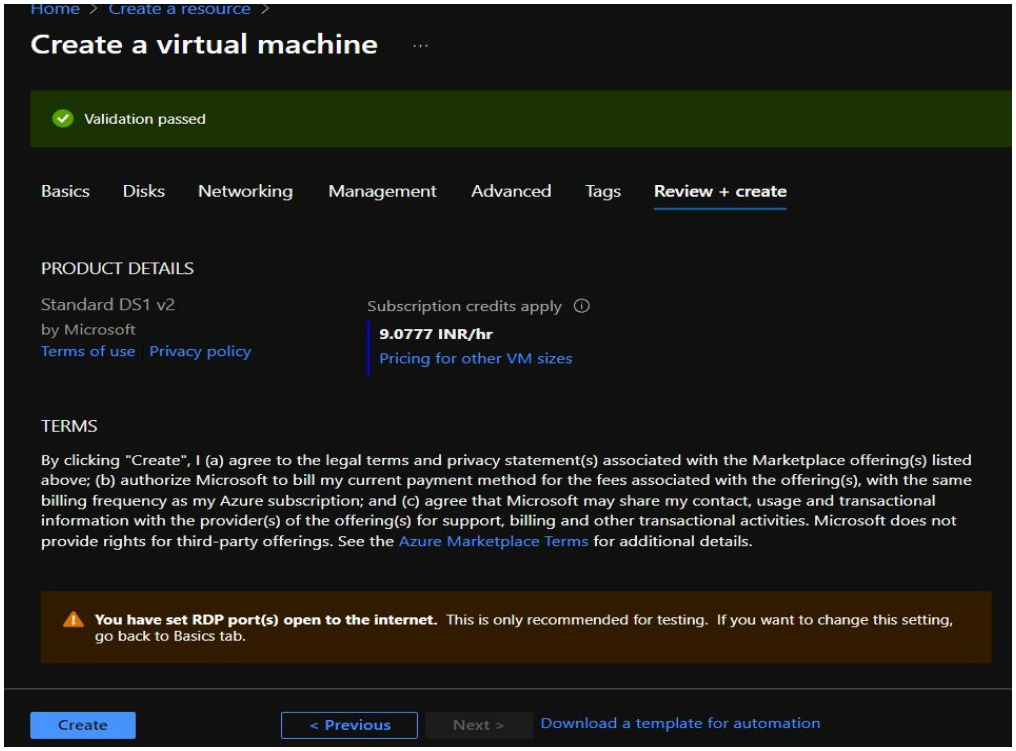
3. Select windows server



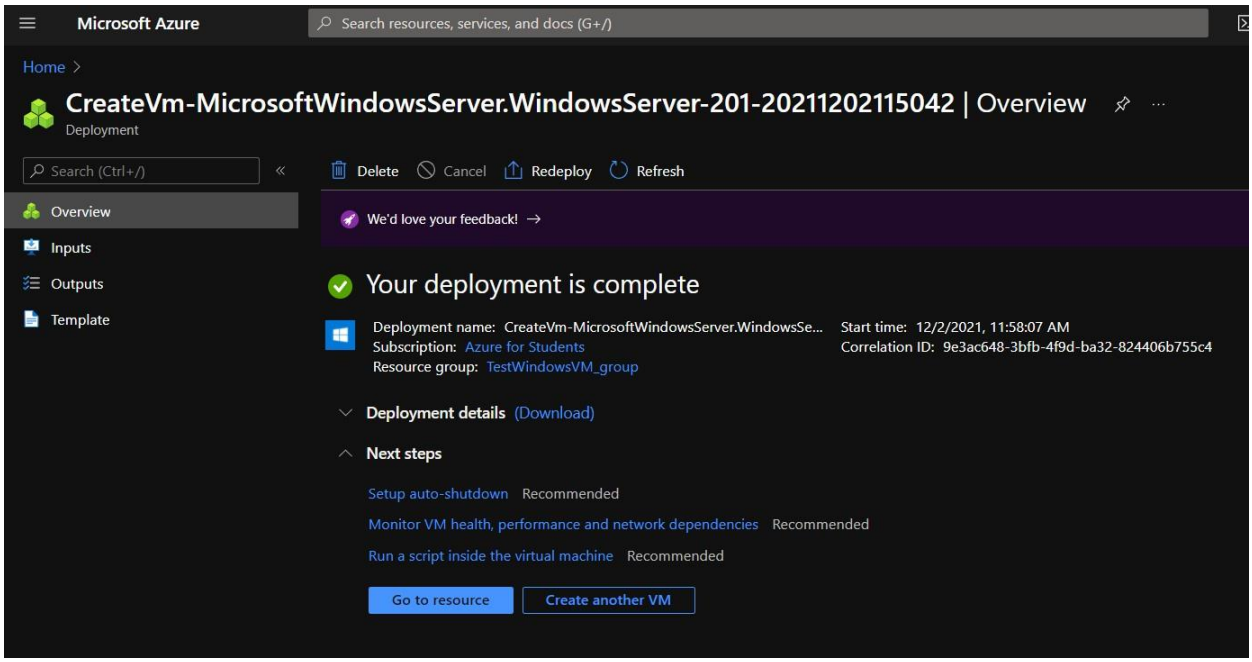
4. Click review and create



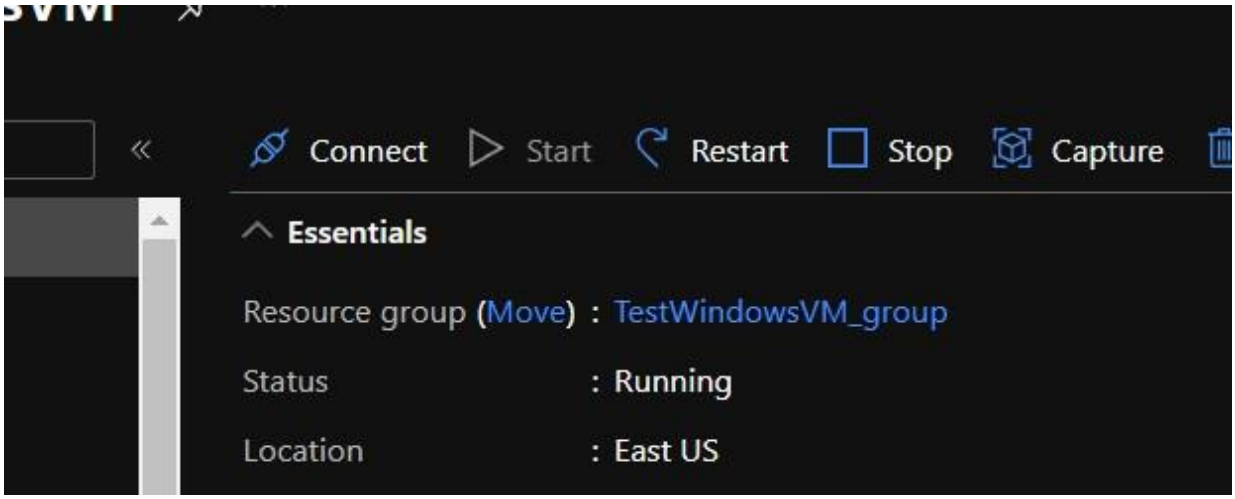
5. Click on create



6. Click on ‘Go to resource’ once deployed.



7. Click connect to remotely connect using SSH or RDP (If needed).

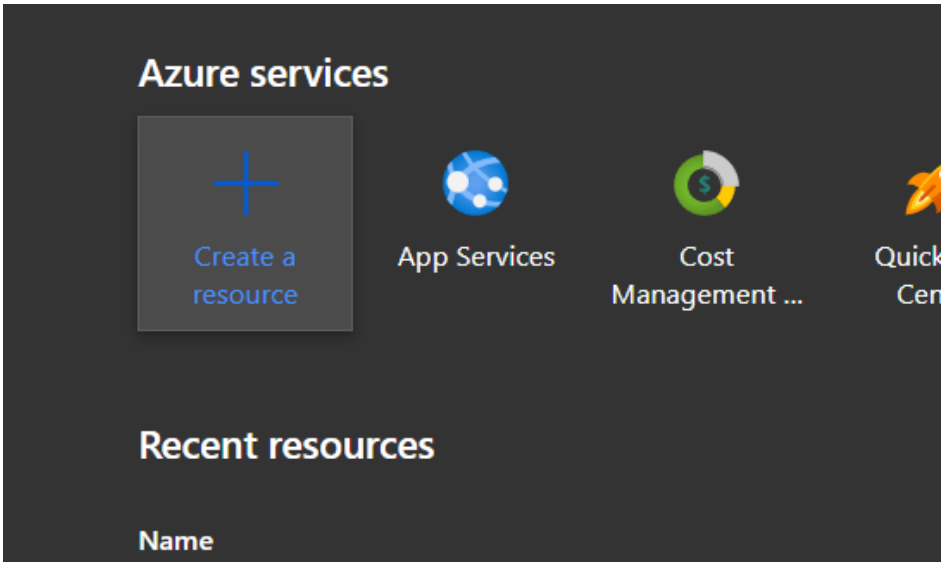


Exercise-9

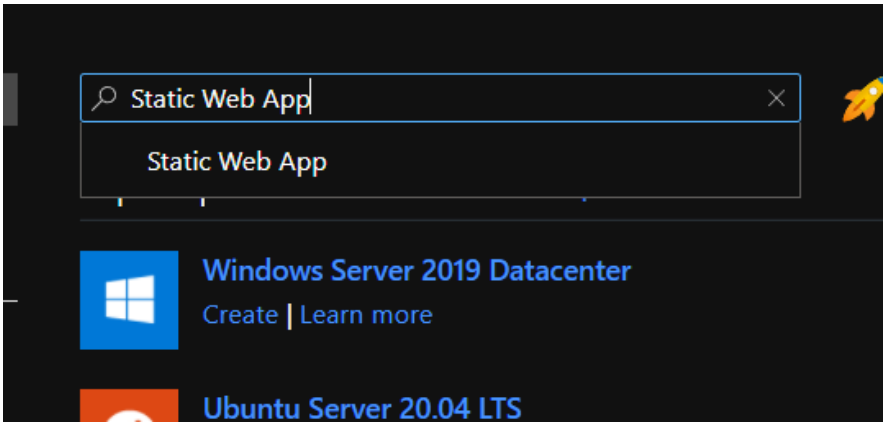
Creating a static web app in Azure Portal

Steps:

1. Click on Create Resource

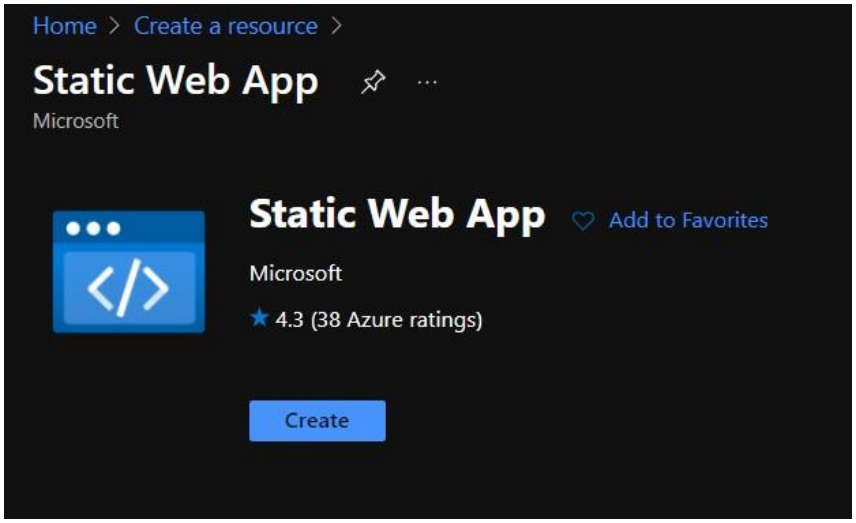


2. Search for Static Web App

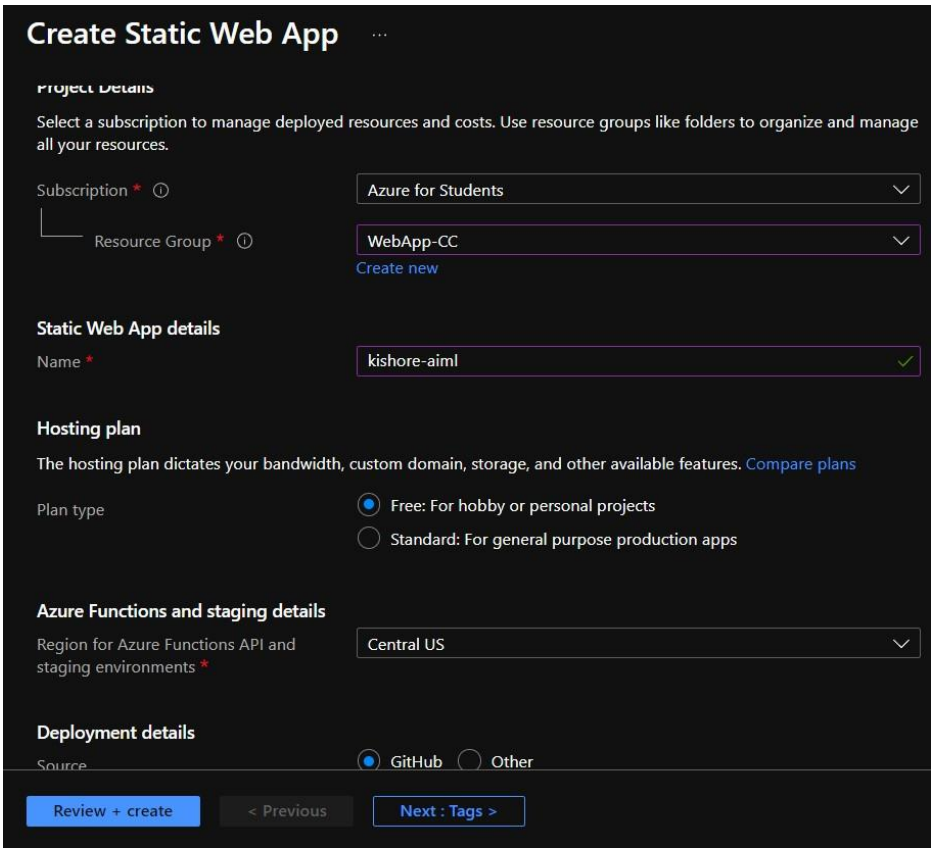


3. Select Static web app

4. Select Create



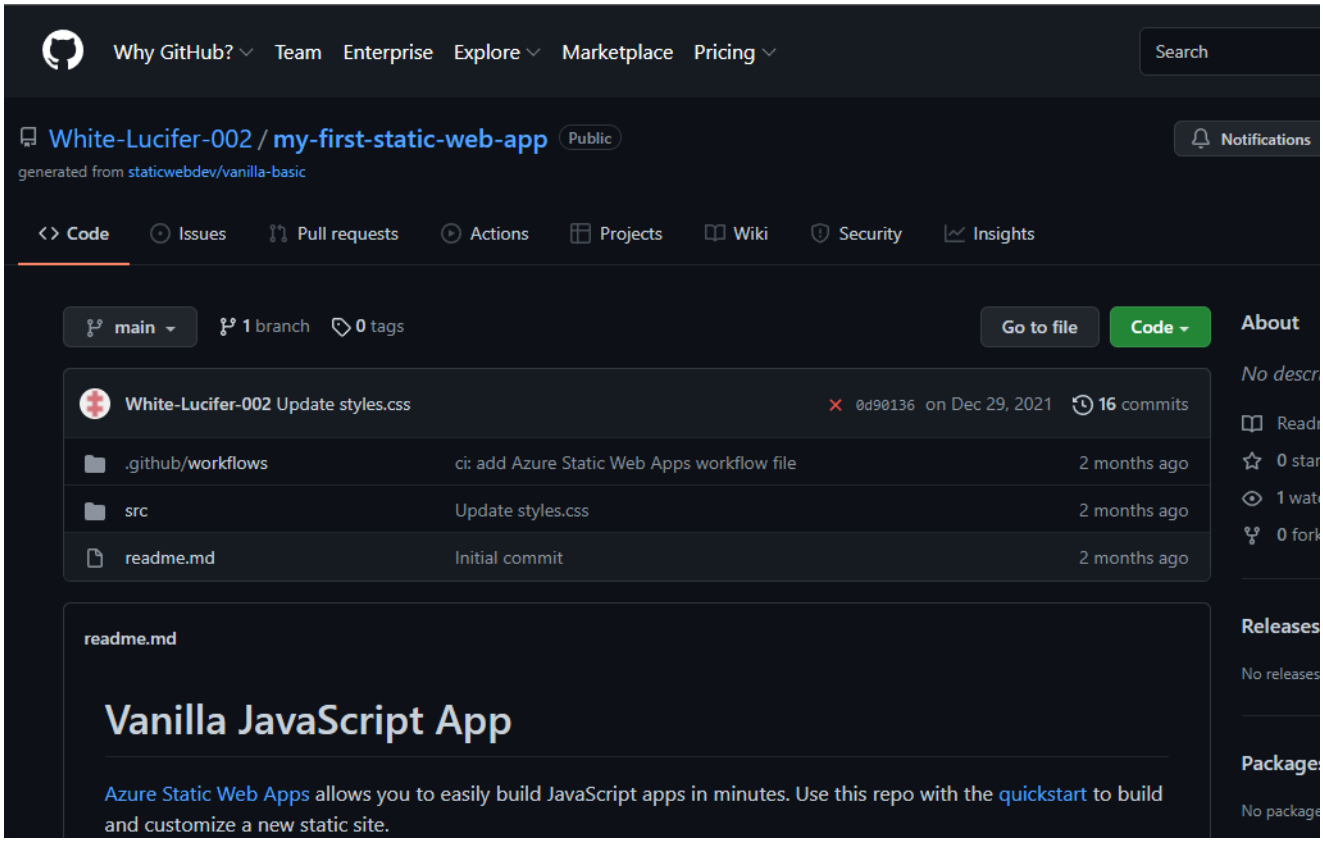
5. Fill out the details



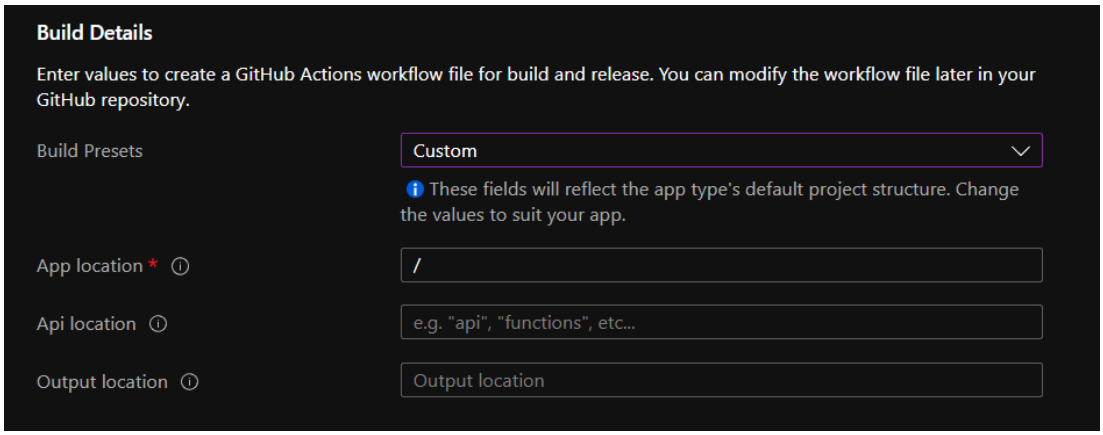
5. Sign in via GitHub

6. Select organization, Repository and Branch where the HTML file is located

7. The GitHub repo should look like this



8. Fill in the following Build Details

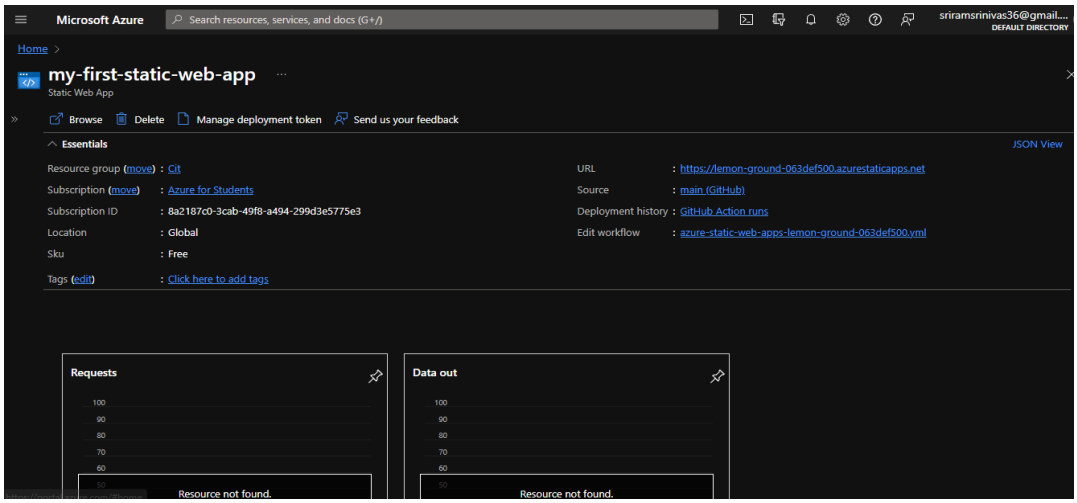


9. Click on Review + Create

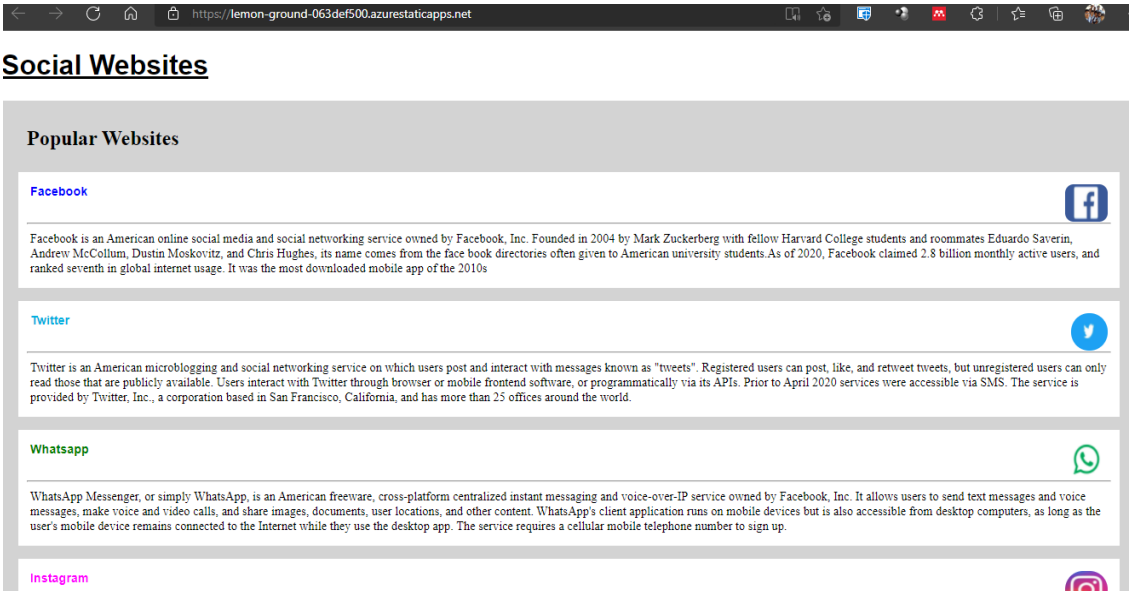
10. Click on Create

11. Click go to resource

12. The URL for the site will be available



13. Click on the URL to see if it is working



14. This is the URL for the Static Web Page created by me: https://lemon-ground-063def500.azurestaticapps.net/

Exercise-10

Creating a Docker container

Steps:

1. Configuring Virtual Machine

2. Installing SSH in VM

   Open terminal in VM and run the following command

   >> sudo apt install ssh

3. Getting the IP address of NAT adapter

   Open terminal in VM and run the following command.

>> ifconfig

Note down the ip address of the NAT adapter.

```
athi@virtualmachine:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:29ff:fe9a:495f  prefixlen 64  scopeid 0x20<link>
        ether 02:42:29:9a:49:5f  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 48  bytes 6637 (6.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.108  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe3c:ec02  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:3c:ec:02  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98  bytes 10176 (10.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.3.15  netmask 255.255.255.0  broadcast 10.0.3.255
        inet6 fe80::53d5:6e21:a47a:723f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:6e:76:a2  txqueuelen 1000  (Ethernet)
        RX packets 6404  bytes 4835404 (4.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3812  bytes 389220 (389.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 263  bytes 22355 (22.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 263  bytes 22355 (22.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ip address of the NAT adapter : 10.0.3.15

4. Go to VM settings > Network > Advanced Settings > Port forwarding

Click on the plus icon and fill the following

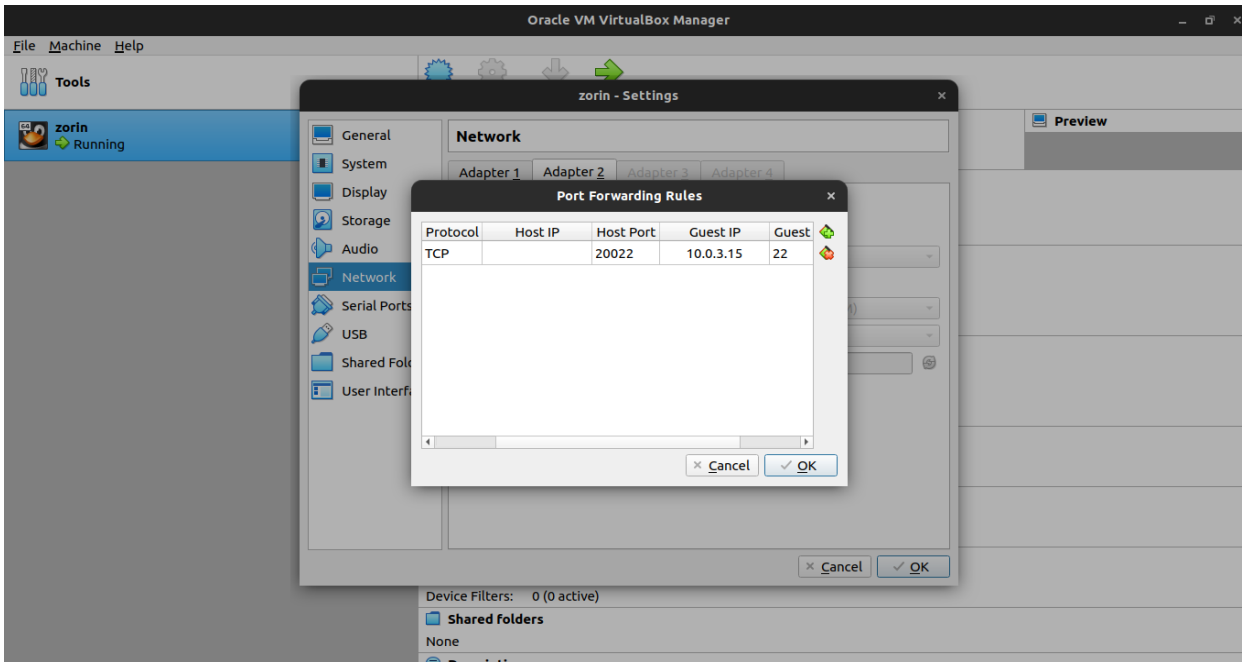
Host ip : Leave empty

Host Port : 20022

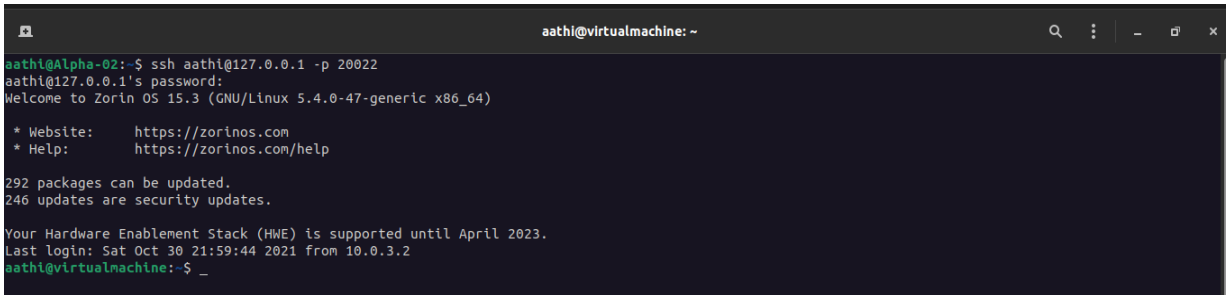
Guest ip : 10.0.3.15 (Ip of NAT adapter)

Guest port : 22

Click **OK.**



5. Open terminal(LINUX) or Command prompt(WINDOWS) in Host machine



6. Run the following command to make an ssh connection with guest machine

>> ssh username@127.0.0.1 -p 20022

Eg., Here “aathi” is the hostname of VM ., Therfore,

>> ssh aathi@127.0.0.1 -p 20022

Use the following command to check the status of SSH

>> systemctl status ssh

```
aathi@virtualmachine: ~
aathi@virtualmachine:~$ systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor prese
   Active: active (running) since Sat 2021-10-30 21:54:28 IST; 1h 9min ag
  Process: 1247 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0
  Process: 1242 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCE
  Process: 693 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCC
 Main PID: 790 (sshd)
    Tasks: 1 (limit: 2055)
   CGroup: /system.slice/ssh.service
           └─790 /usr/sbin/sshd -D

Oct 30 21:54:33 virtualmachine sshd[790]: Server listening on :: port 22.
Oct 30 21:54:54 virtualmachine systemd[1]: Reloading OpenBSD Secure Shell
Oct 30 21:54:54 virtualmachine sshd[790]: Received SIGHUP; restarting.
Oct 30 21:54:54 virtualmachine systemd[1]: Reloaded OpenBSD Secure Shell
Oct 30 21:54:54 virtualmachine sshd[790]: Server listening on 0.0.0.0 por
Oct 30 21:54:54 virtualmachine sshd[790]: Server listening on :: port 22.
Oct 30 21:59:34 virtualmachine sshd[2486]: Accepted password for aathi fr
Oct 30 21:59:34 virtualmachine sshd[2486]: pam_unix(sshd:session): sessio
Oct 30 21:59:43 virtualmachine sshd[2578]: Accepted password for aathi fr
Oct 30 21:59:43 virtualmachine sshd[2578]: pam_unix(sshd:session): sessio
lines 1-21/21 (END)
```

7. After making successful connection with VM, install golang.

>> sudo apt install golang-go

8. Create a new file “hello1.go” and write code for ‘hello world’ program.

```go
package main

import (
	"fmt"
)

func main() {
	fmt.Println("Hello, World")
}
```

9. Run the go program using following command.

>> go run hello1.go

```
aathi@virtualmachine:~$ go run hello1.go
Hello, World
aathi@virtualmachine:~$ _
```

10. Run the go program by creating executables.

>> go build hello1.go

>> ./hello1

```
aathi@virtualmachine:~$ go build hello1.go
aathi@virtualmachine:~$ ./hello1
Hello, World
aathi@virtualmachine:~$ _
```

Docker Installation :

1. Follow the steps on https://docs.docker.com/engine/install/ubuntu/ for docker installation.

2. Create a new file named, ‘Dockerfile’. Write the following code in it.

```
FROM golang:latest
COPY hello1.go .
CMD go run hello1.go
```

3. After save the, ‘Dockerfile’, run the following command in terminal to build a container.

```
>> sudo docker build -t test:1 .
```

4. Check for the builded docker images using the following command

```
>> sudo docker images
```

5. Run the docker using the following command

```
>> sudo docker run test:1
```

Exercise-11

Entering into Container making Logs of the activities

Steps:

1. To view all the container process in action
   >>sudo docker ps

```
aathi@virtualmachine:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
          PORTS     NAMES
9c4b4204d6e4   test:1    "/bin/sh -c 'go run …"   28 seconds ago   Up 24
seconds             nostalgic_colden
```

2. To remove an docker image, use the following command.

   >> sudo docker rmi -f <image id #1> <image id #2> …. <image id #n>

```
aathi@virtualmachine:~$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
test         1         6b821009acd1   19 minutes ago   941MB
test         2         6b821009acd1   19 minutes ago   941MB
<none>       <none>    27c32f200864   54 minutes ago   941MB
golang       latest    9f8b89ee4475   2 weeks ago      941MB
aathi@virtualmachine:~$ sudo docker rmi -f 6b821009acd1
Untagged: test:1
Untagged: test:2
Deleted: sha256:6b821009acd1a6bd346703f4e6dbe3b901ae09df0a8336de10b0f5c9e
f715bcf
Deleted: sha256:290fa9dd1e1dbe889f97c55a2ebb5cf981740d61e83d9657a0e34f3a5
5a9d9ff
aathi@virtualmachine:~$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
<none>       <none>    27c32f200864   55 minutes ago   941MB
golang       latest    9f8b89ee4475   2 weeks ago      941MB
aathi@virtualmachine:~$ _
```

3. **Docker File Modification:**

1. Create an executable file for hello.go program

   >> go build hello.go

2. Change the docker file code to following.

```
FROM alpine:3.14
copy hello .
cmd ./hello
```

3. Build a new container and run it. Space occupied by the container is less than the previous one.

```
aathi@virtualmachine:~$ go build hello.go
aathi@virtualmachine:~$ ls
Desktop     Documents  hello     Music     Public     Videos
Dockerfile  Downloads  hello.go  Pictures  Templates
aathi@virtualmachine:~$ sudo docker build -t log:2 .
[sudo] password for aathi:
Sending build context to Docker daemon  281.3MB
Step 1/3 : FROM alpine:3.14
3.14: Pulling from library/alpine
a0d0a0d46f8b: Pull complete
Digest: sha256:e1c082e3d3c45cccac829840a25941e679c25d438cc8412c2fa221cf1a
824e6a
Status: Downloaded newer image for alpine:3.14
 ---> 14119a10abf4
Step 2/3 : copy hello .
 ---> 4ec57ffa17de
Step 3/3 : cmd ./hello
 ---> Running in 99d352ac254c
Removing intermediate container 99d352ac254c
 ---> 492705c46881
Successfully built 492705c46881
Successfully tagged log:2
aathi@virtualmachine:~$ sudo docker run log:2
2021/10/30 17:04:37 Entering Main Function
2021/10/30 17:04:37 Code in Running
Hello, playground
^C^X^Zaathi@virtualmachine:~$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
log          2         492705c46881   38 seconds ago   7.7MB
log          1         706397e93147   22 minutes ago   943MB
<none>       <none>    7634d810db46   24 minutes ago   943MB
<none>       <none>    7bd2fd5bd7e3   2 hours ago      941MB
<none>       <none>    27c32f200864   3 hours ago      941MB
golang       latest    9f8b89ee4475   2 weeks ago      941MB
alpine       3.14      14119a10abf4   2 months ago     5.6MB
aathi@virtualmachine:~$ _
```

- Change the go file with log functions.

```go
package main

import (
	"fmt"
	"time"
	"log"
)

func main() {
	log.Println("Entering Main Function")
	fmt.Println("Hello, playground")
	log.Println("Code in Running")
	time.Sleep(300*time.Second)
	log.Println("Exiting Main Function")
}
```

- Build a new docker and run it.

```
aathi@virtualmachine:~$ sudo docker build -t log:1 .
Sending build context to Docker daemon  281.2MB
Step 1/3 : from golang:latest
 ---> 9f8b89ee4475
Step 2/3 : copy hello.go .
 ---> 23fe2ab617a1
Step 3/3 : cmd go run hello.go
 ---> Running in c4d49730e09f
Removing intermediate container c4d49730e09f
 ---> 7bd2fd5bd7e3
Successfully built 7bd2fd5bd7e3
Successfully tagged log:1
aathi@virtualmachine:~$ sudo docker run log:1
2021/10/30 15:14:44 Entering Main Function
Hello, playground
2021/10/30 15:14:44 Code in Running
2021/10/30 15:19:44 Exiting Main Function
aathi@virtualmachine:~$ sudo docker run log:1
2021/10/30 15:20:52 Entering Main Function
Hello, playground
2021/10/30 15:20:52 Code in Running
_
```

- In the adjacent shell , run docker logs to check the logs.

```
aathi@virtualmachine:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
          PORTS     NAMES
86e3389a599a   log:1     "/bin/sh -c 'go run …"   19 seconds ago   Up 14
seconds             amazing_bardeen
aathi@virtualmachine:~$ sudo docker logs 86e3389a599a
2021/10/30 15:20:52 Entering Main Function
Hello, playground
2021/10/30 15:20:52 Code in Running
```

USING LOG

- Run the docker file and execute the following commands to enter into docker.

```
aathi@virtualmachine:~$ sudo docker ps
[sudo] password for aathi:
CONTAINER ID   IMAGE     COMMAND                  CREATED              ST
ATUS              PORTS     NAMES
ba37b94852ed   test:1    "/bin/sh -c 'go run …"   About a minute ago   Up
 About a minute             brave_keldysh
aathi@virtualmachine:~$ sudo docker exec -it ba37b94852ed bash
root@ba37b94852ed:/go# aathi@virtualmachine:~$ sudo docker exec -it ba37b
aathi@virtualmachine:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
          PORTS     NAMES
9c4b4204d6e4   test:1    "/bin/sh -c 'go run …"   28 seconds ago   Up 24
seconds             nostalgic_colden
aathi@virtualmachine:~$ sudo docker exec -it 9c4b4204d6e4 bash
root@9c4b4204d6e4:/go# hostname
9c4b4204d6e4
root@9c4b4204d6e4:/go# ps -al
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 R     0    43    37  1  80   0 -  1653 -      pts/0    00:00:00 ps
root@9c4b4204d6e4:/go# aathi@virtualmachine:~$ _
```

ENTERING INTO DOCKER USING INTERACTIVE BASH SHELL

- Run the docker and use the following command to stop the docker
.

```
aathi@virtualmachine:~$ sudo docker ps
[sudo] password for aathi:
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
          PORTS     NAMES
80e14f01514f   log:1     "/bin/sh -c 'go run …"   38 seconds ago   Up 31
seconds             intelligent_dewdney
aathi@virtualmachine:~$ sudo docker exec -it 80e14f01514f bash
root@80e14f01514f:/go# uname -a
Linux 80e14f01514f 5.4.0-47-generic #51~18.04.1-Ubuntu SMP Sat Sep 5 14:3
5:50 UTC 2020 x86_64 GNU/Linux
root@80e14f01514f:/go# exit
exit
aathi@virtualmachine:~$ sudo docker stop 80e14f01514f
80e14f01514f
aathi@virtualmachine:~$ _
```

STOPPING THE RUNNING DOCKER IMAGE

- Change the docker file as follows:

```
FROM golang:latest
COPY hello.go .
RUN go version
RUN go build hello.go
RUN pwd
CMD ./hello

FROM alpine:3.14
COPY --from=0 /go/hello .
CMD ./hello
```

- Build the docker file, check for space taken by the image and run it.

```
aathi@virtualmachine:~$ sudo docker build -t log:3 .
[sudo] password for aathi:
Sending build context to Docker daemon  281.3MB
Step 1/9 : FROM golang:latest
 ---> 9f8b89ee4475
Step 2/9 : COPY hello.go .
 ---> Using cache
 ---> 23fe2ab617a1
Step 3/9 : RUN go version
 ---> Running in 3cb4ac3c85ce
go version go1.17.2 linux/amd64
Removing intermediate container 3cb4ac3c85ce
 ---> 1a34197c55a5
Step 4/9 : RUN go build hello.go
 ---> Running in 78839a288d4d
Removing intermediate container 78839a288d4d
 ---> 38c7ae4704f7
Step 5/9 : RUN pwd
 ---> Running in 40779d26f176
/go
Removing intermediate container 40779d26f176
 ---> 441d212fe96d
Step 6/9 : CMD ./hello
 ---> Running in bdf667d3f11c
Removing intermediate container bdf667d3f11c
 ---> 5c334851cead
Step 7/9 : FROM alpine:3.14
 ---> 14119a10abf4
Step 8/9 : COPY --from=0 /go/hello .
 ---> b07b55d488ee
Step 9/9 : CMD ./hello
 ---> Running in bbbebfc2776a
Removing intermediate container bbbebfc2776a
 ---> ecade66debac
Successfully built ecade66debac
Successfully tagged log:3
```

```
aathi@virtualmachine:~$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
log          3         ecade66debac   28 seconds ago   7.45MB
<none>       <none>    5c334851cead   34 seconds ago   943MB
log          2         492705c46881   24 minutes ago   7.7MB
log          1         706397e93147   46 minutes ago   943MB
golang       latest    9f8b89ee4475   2 weeks ago      941MB
alpine       3.14      14119a10abf4   2 months ago     5.6MB
aathi@virtualmachine:~$ _
```

```
aathi@virtualmachine:~$ sudo docker run log:3
2021/10/30 17:28:57 Entering Main Function
2021/10/30 17:28:57 Code in Running
Hello, playground
_
```

DOCKER BUILD AND EXECUTION

Exercise-12

INSTALLING KUBERNETES

Steps:

- Use the following commands to install kubernetes..

- `cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf br_netfilter EOF`
- `cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf net.bridge.bridge-nf-call-ip6tables = 1 net.bridge.bridge-nf-call-iptables = 1 EOF`
- `sudo sysctl --system`
- `sudo apt-get update`
- `sudo apt-get install -y apt-transport-https ca-certificates curl`
- `sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg`
- `echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list`
- `sudo apt-get update`
- `sudo apt-get install -y kubelet kubeadm kubectl`
- `sudo apt-mark hold kubelet kubeadm kubectl`

- Create a new file called 'kubeadm-config.yaml' and write the following codes in it.

```
kind: ClusterConfiguration
apiVersion: kubeadm.k8s.io/v1beta3
kubernetesVersion: v1.21.0
---
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
cgroupDriver: system
```

- use the following command to turn off swapping

  - `sudo swapoff -a`

- use the following command to initialize kubeadm

  - `sudo kubeadm init`

- Use "kubectl get pods -n kube-system" to see the pods.

- Add Calico for Kuberentes from https://docs.projectcalico.org/getting-started/kubernetes/quickstart

- Create a helloworld kubernetes file by following the steps in the GitHub page https://github.com/paulbouwer/hello-kubernetes

Exercise-13

CREATING A GRPC SERVER CLIENT PROGRAM AND DOCKERIZING IT

Steps:

- *INSTALLING GRPC PLUGINS:*
  - Install the protocol compiler plugins for Go using the following commands:
    - `go install google.golang.org/protobuf/cmd/protoc-gen-go@v1.26`
    - `go install google.golang.org/grpc/cmd/protoc-gen-go-grpc@v1.1`
  - Update your PATH so that the protoc compiler can find the plugins:
    - `export PATH="$PATH:$(go env GOPATH)/bin"`
  - Download the repo as a zip file and unzip it, or clone the repo:
    - `git clone -b v1.41.0 https://github.com/grpc/grpc-go`

```
go: downloading google.golang.org/protobuf v1.25.0
go: downloading github.com/golang/protobuf v1.4.3
go: downloading google.golang.org/genproto v0.0.0-20200806141610-86f49bd18e98
go: downloading golang.org/x/net v0.0.0-20200822124328-c89045814202
go: downloading golang.org/x/sys v0.0.0-20200323222414-85ca7c5b95cd
go: extracting github.com/golang/protobuf v1.4.3
go: extracting google.golang.org/protobuf v1.25.0
go: extracting golang.org/x/sys v0.0.0-20200323222414-85ca7c5b95cd
go: extracting golang.org/x/net v0.0.0-20200822124328-c89045814202
go: downloading golang.org/x/text v0.3.0
go: extracting golang.org/x/text v0.3.0
go: extracting google.golang.org/genproto v0.0.0-20200806141610-86f49bd18e98
go: finding golang.org/x/net v0.0.0-20200822124328-c89045814202
go: finding github.com/golang/protobuf v1.4.3
go: finding google.golang.org/protobuf v1.25.0
go: finding google.golang.org/genproto v0.0.0-20200806141610-86f49bd18e98
go: finding golang.org/x/sys v0.0.0-20200323222414-85ca7c5b95cd
go: finding golang.org/x/text v0.3.0
```

  - Change to the quick start example directory:
    - `cd grpc-go/examples/helloworld`
- BUILD THE MAIN GO FILE:
  - Run the command "go build main.go"

```
2021/11/14 15:09:51 Greeting: Hello world
```

- Create a docker file for the grpc server and client program.

```
FROM golang:latest
COPY main .
RUN ls
RUN chmod +x main
CMD ./main
```

- Build both the docker files.
- Change port as 5000 in the greeter go file and enable port forwarding in server using the following command:

```
sudo docker run -p 127.0.0.1:5000:5000/tcp -d server_test:1
```

- Change the listening port in client as 5000 and run the main.go file
- Run the following command:

```
go run main.go
```

```
2021/11/14 15:43:32 Greeting: Hello world
```

Exercise-14

CONTAINERIZING THE GRPC SERVER-CLIENT PROGRAM USING KUBERNETES

Steps:

- Create a Kubernetes deployment YAML file from…

  https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

- Write the following code in that file:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grpc-server-client
  labels:
    app: grpc-chat
spec:
  replicas: 3
  selector:
    matchLabels:
      app: grpc-chat
  template:
    metadata:
      labels:
        app: grpc-chat
    spec:
      containers:
      - name: grpc-server
        image: grpc_server:1
        ports:
        - containerPort: 50051
```

- Create the deployment using the following command.

  - `Kubectl create -f kube.yaml`

- Run the following command to run deployments.

  - `Kubectl get deployments`

```
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
grpc-server-client     0/3     3            0           16d
grpc-server-client-1   0/3     3            0           118s
nginx-deployment       0/2     2            0           36d
```

- Create a new file called **“service.yaml”.**
  - Type in the following code into the file..

```yaml
apiVersion: v1
            kind: Service
            metadata:
              name: grpc-service
            spec:
              type: NodePort
              selector:
                app: grpc-chat
              ports:
            # By default and for convenience, the `targetPort` is set to the
      same value as the `port` field.
                - port: 50051
                  targetPort: 50051
            # Optional field
            # By default and for convenience, the Kubernetes control plane will
      allocate a port from a range (default: 30000-32767)
                  nodePort: 30000
```

  - Deploy the service using following command
    - `kubectl create -f service.yaml`

  - Run the following command to get services.
    - `Kubectl get services`

```
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
grpc-servie   NodePort    10.97.45.72   <none>        50051:30000/TCP  13s
kubernetes    ClusterIP   10.96.0.1     <none>        443/TCP          20d
```

- Change the IP Address in the greeter_client file to “10.0.2.15:30000” and run the program to see the results.

```go
package main

import (
        "context"
        "log"
        "os"
        "time"

        "google.golang.org/grpc"
        pb "google.golang.org/grpc/examples/helloworld/helloworld"
)

const (
        address     = "10.0.2.15:30000"
        defaultName = "world"
```

- Run , “go run main.go”

```
2021/11/14 15:43:32 Greeting: Hello world
```