Cloud Computing Lab Record

**1934008 GOKUL RAM K 1934023 NANDA KISHORE V  
1934051 SURYA R E**

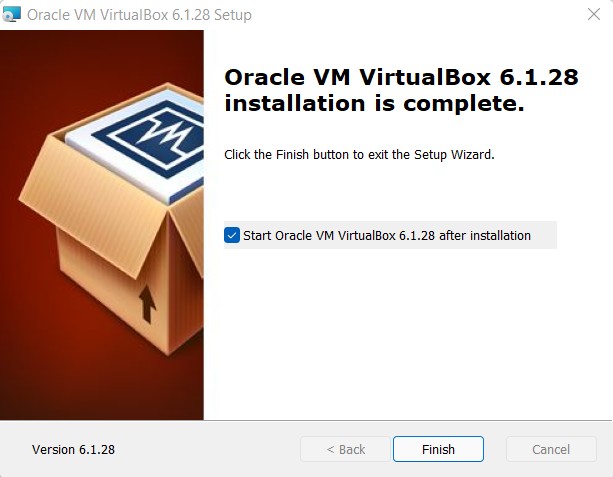
|  |  |
| --- | --- |
| **S.no** | **Index** |
| 1 | Installing Oracle VirtualBox |
| 2 | Running a C program on Virtual Machine |
| 3 | Communication Between Host and Virtual Machine |
| 4 | VM to VM Connection |
| 5 | Exporting VirtualBox VM |
| 6 | Importing VirtualBox VM |
| 7 | Create VM on Azure |
| 8 | Create WebApp on Azure |
| 9 | Creating a shared folder |
| 10 | Creating a Docker Container |
| 11 | Entering into Container making Logs of the activities |
| 12 | Installing Kubernetes |
| 13 | Creating a GRPC server-client program and Dockerizing it |
| 14 | Containerizing the GRPC server-client program using Kubernetes |

# Exercise 1 Installing Oracle VirtualBox

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

Select your OS

1. Run the file
2. Click Next
3. Leave the default settings and click next to install



# Exercise 2 Running a C program on Virtual Machine Linux

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

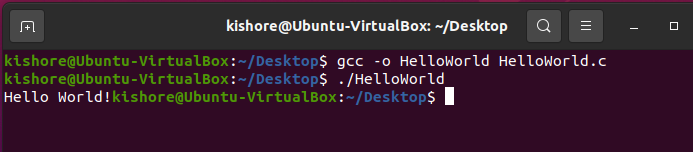
#include <stdio.h> void main()

{

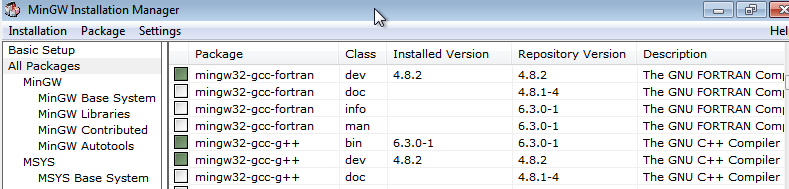
printf("Hello World!");

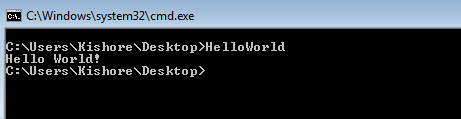
}

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



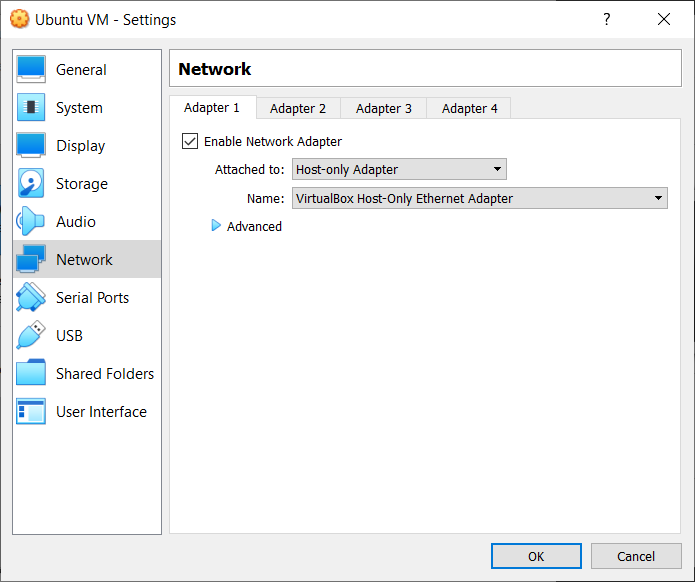
# Windows

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

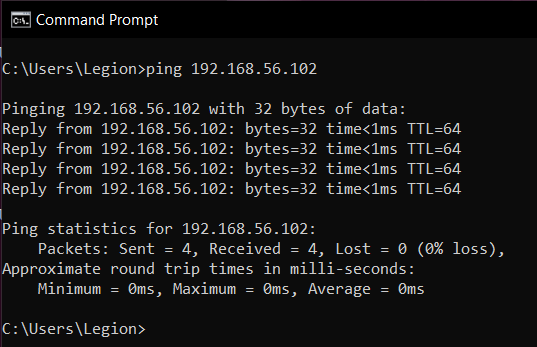


# Exercise 3 Communication Between Host and Virtual Machine

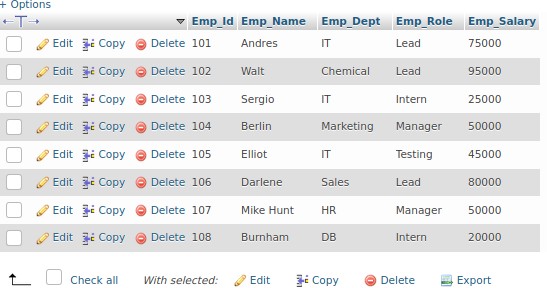
1. Change the adapter Settings to Host-only Adapter



1. Ping the Guest VM from the host

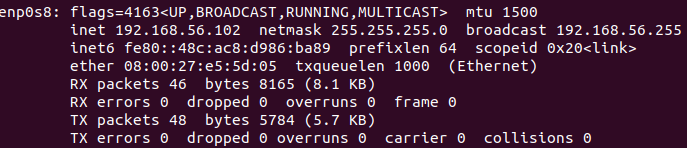


1. Install XAMPP and change the required files
2. Start XAMPP and Create a Database and a Table



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

IP



User

1. Download the MySQL-connector jar file and add it to the project path in Host.
2. 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 {

PASSWORD);

Class.forName(JDBC\_DRIVER); System.out.println("Connecting to a selected database..."); conn = DriverManager.getConnection(DB\_URL, USER,

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 {

}

}try {

if(stmt!=null)

conn.close();

}catch(SQLException se) {

if(conn!=null)

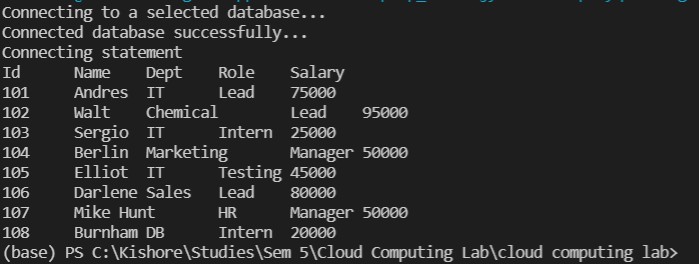
conn.close();

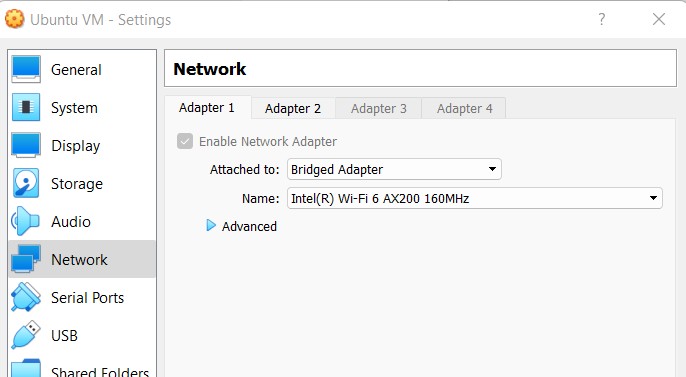
}catch(SQLException se) { se.printStackTrace();

}

}

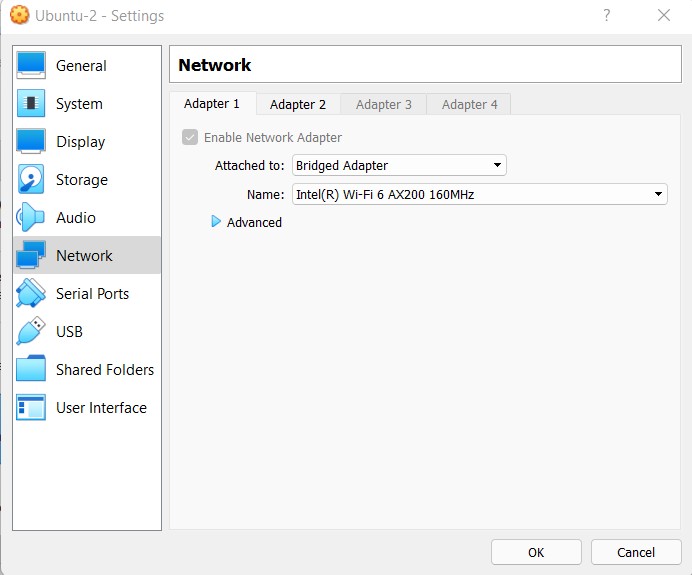
}

1. Run the program and Verify the result

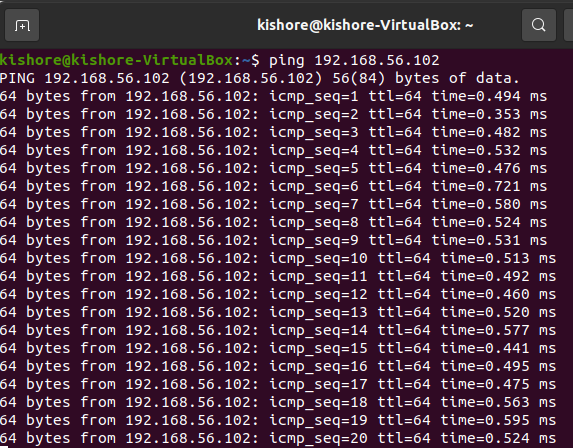
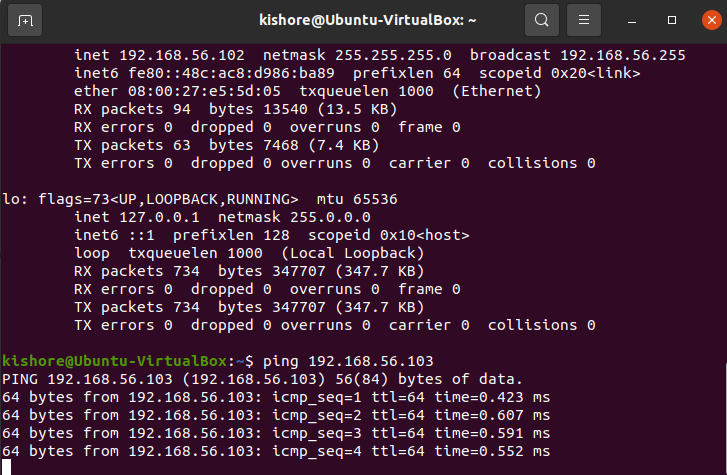


# Exercise 4 VM to VM Connection

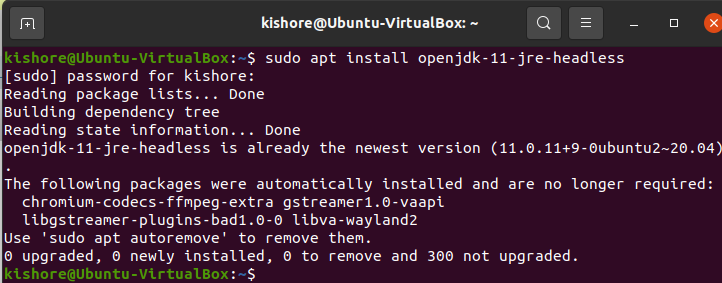
1. After installing two VMs, change their network settings to BridgedAdapter



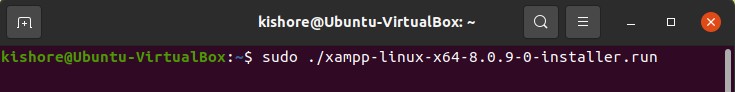
1. Ping and verify both are able to communicate with each other.



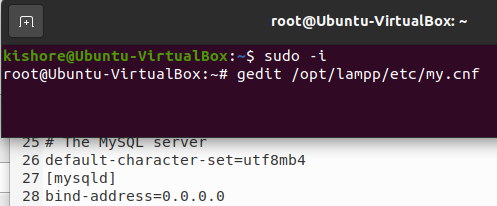
1. Install Java in the VM-B (From which we are going to access the database).

VM-B like VisualStudio Code or Eclipse

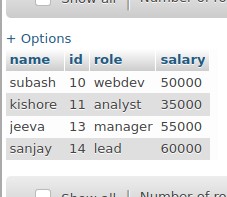
1. Install a code editor/IDE in the
2. Download the XAMPP package in the VM-A (Where we are going to create the Database and Table).



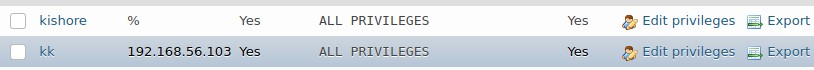
1. Edit my.cnf file and start XAMPP



1. Create a Database and Table using phpmyadmin
2. Insert records into the table



1. In both the VMs change the second network adapter to Host-only-adapter
2. Create a user in the PHPMyAdmin with IP either as “%” or the IP of the VM-B



1. Download the MySQL-connector-java and add it to the project path where the code is present in VM-B
2. 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 {

PASSWORD);

Class.forName(JDBC\_DRIVER); System.out.println("Connecting to a selected database..."); conn = DriverManager.getConnection(DB\_URL, USER,

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 {

}

}try {

if(stmt!=null)

conn.close();

}catch(SQLException se) {

if(conn!=null)

conn.close();

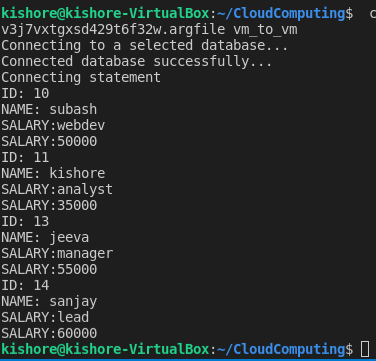
}catch(SQLException se) { se.printStackTrace();

}

}

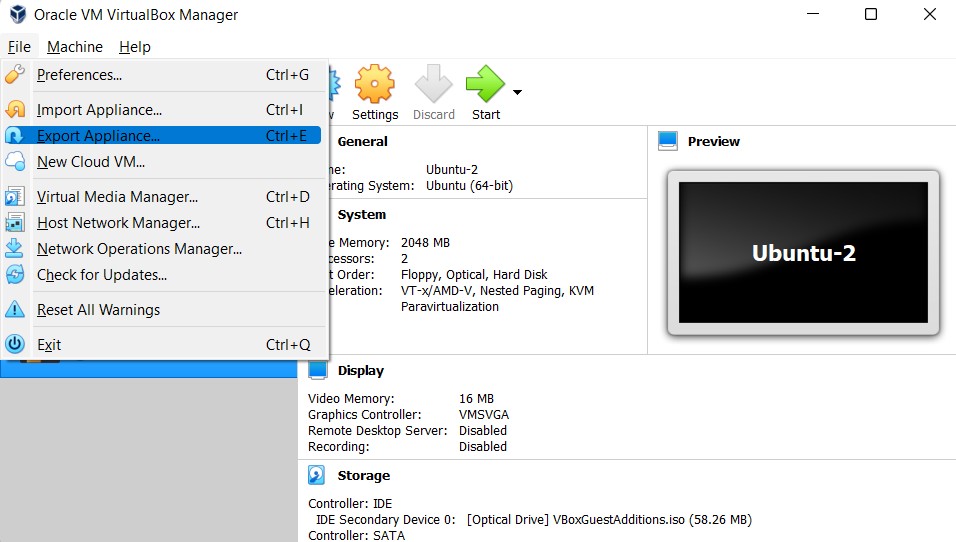
}

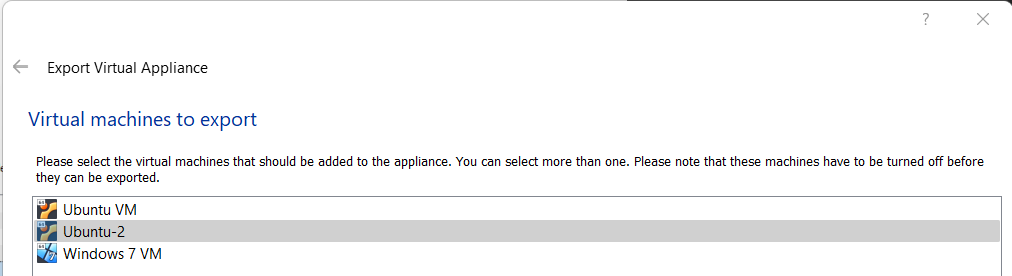
1. Run the program to see the results

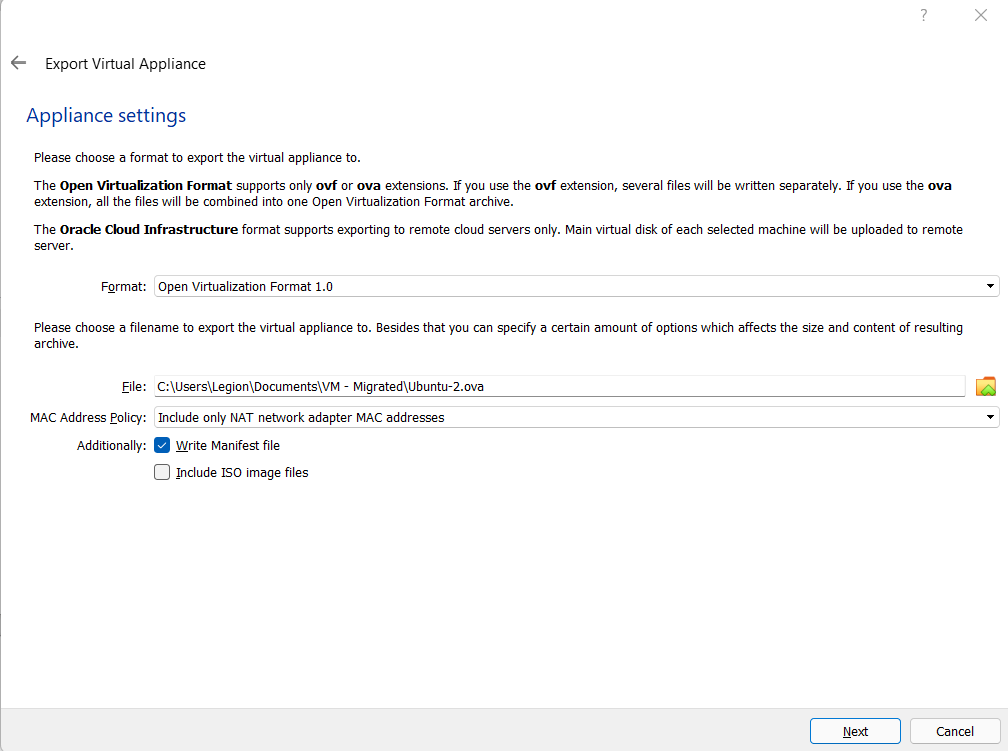


# Exercise 5 Exporting VirtualBox VM

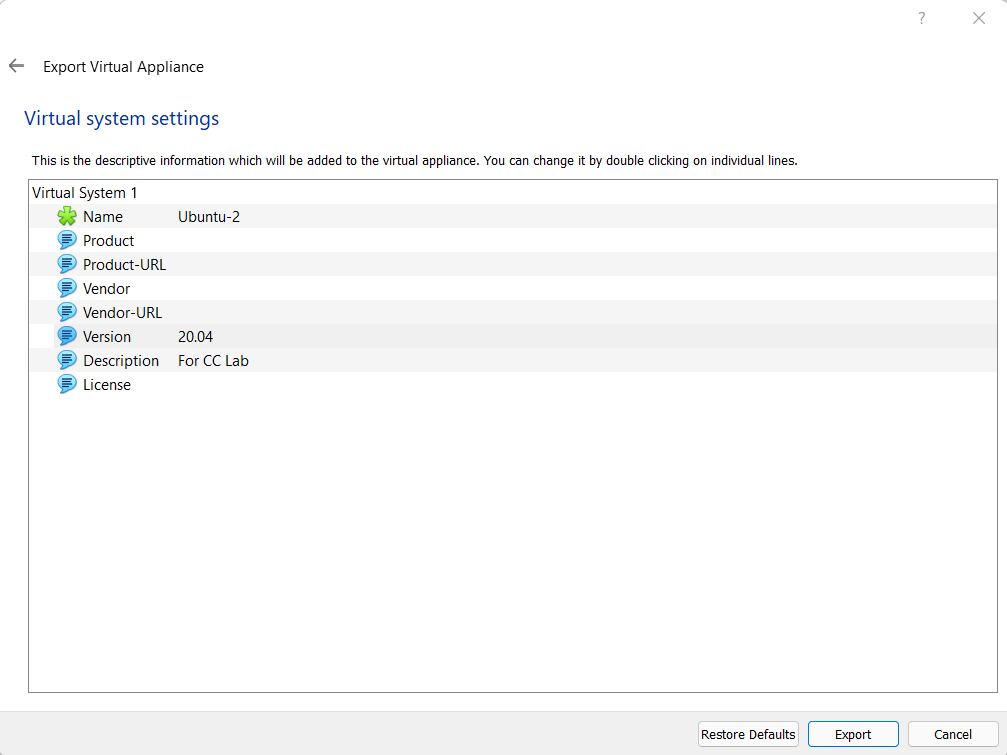
* 1. Open VirtualBox
  2. Click on File --> Export Appliance



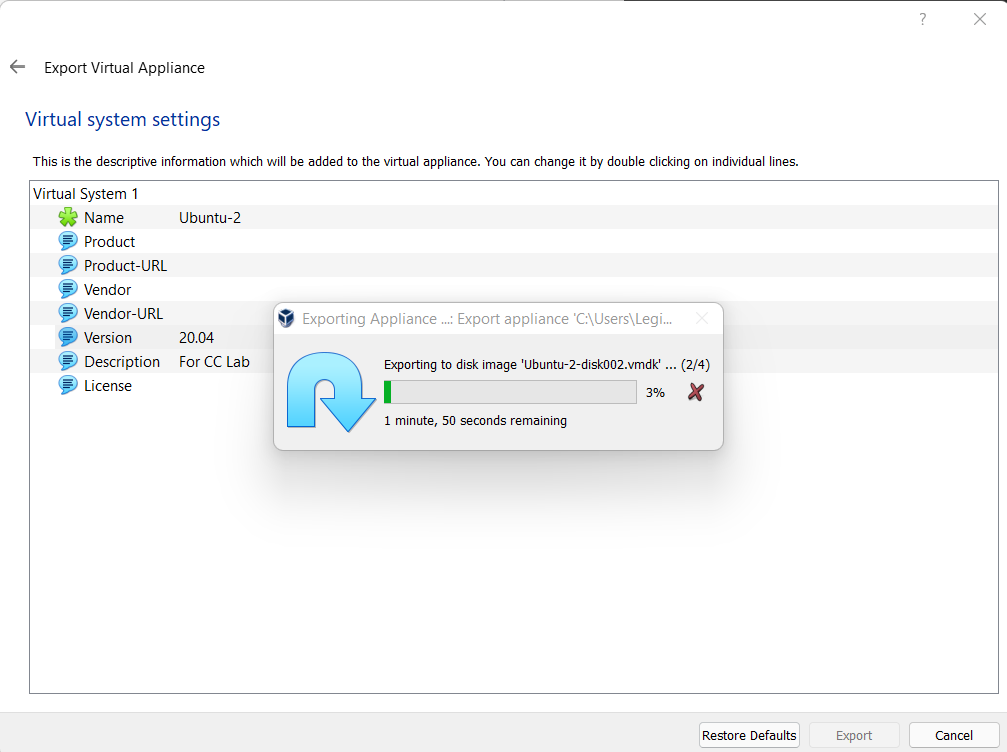
* 1. Select the VM you want to export
  2. Select Location to store the exported VM and click Next



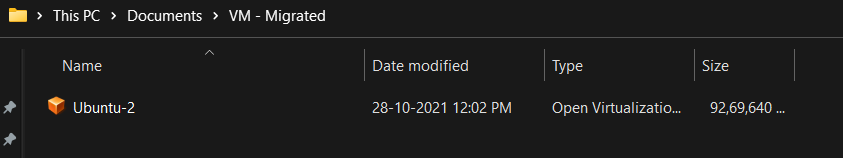
* 1. Add Description if wanted and click export to store the file in the location specified



* 1. Wait for the process to complete

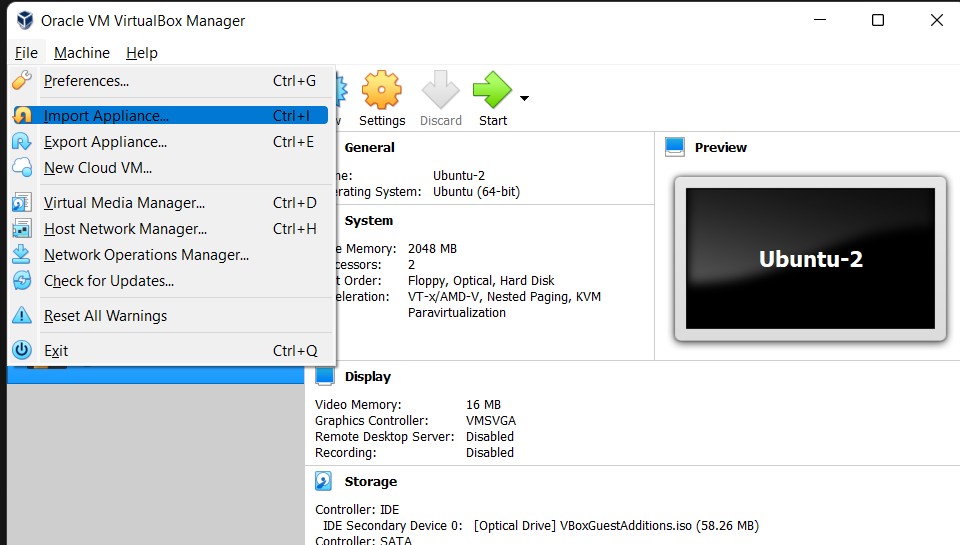


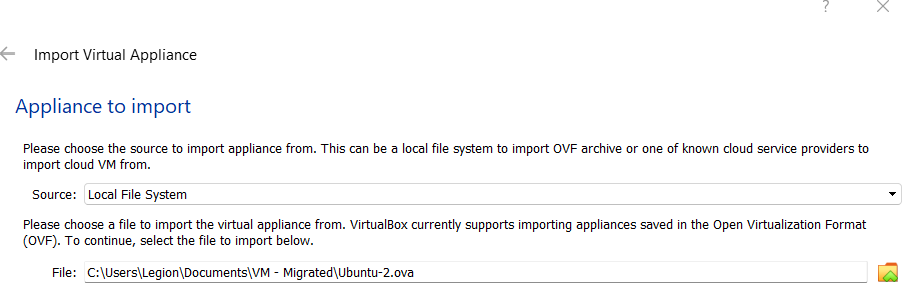
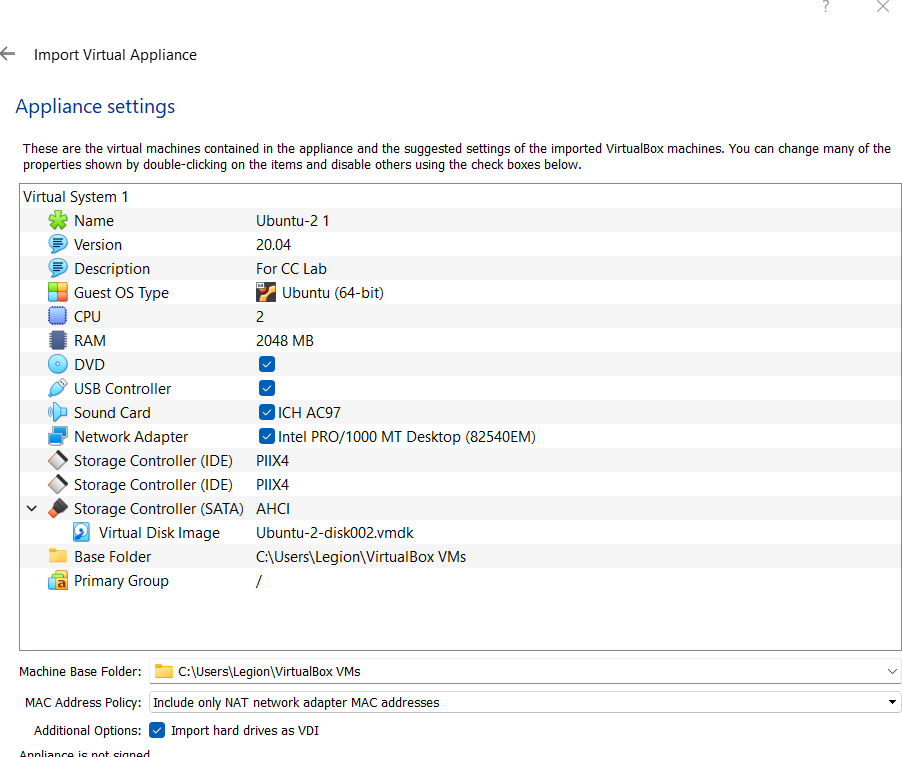
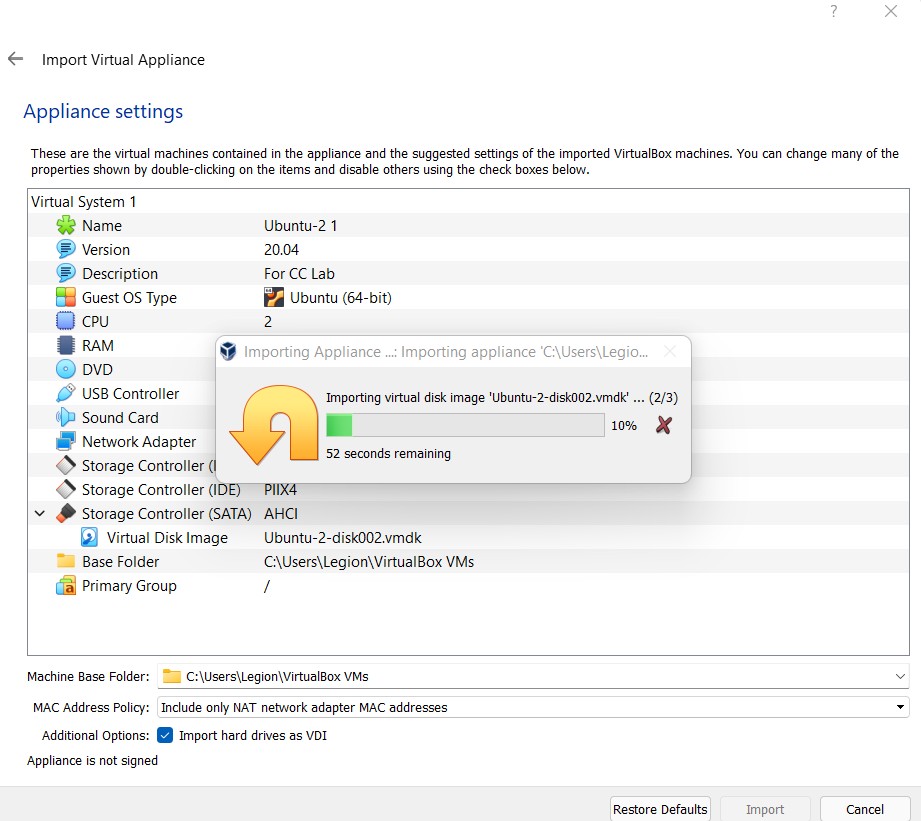
* 1. Check the Location to see the exported .ova file.



# Exercise 6 Importing VirtualBox VM

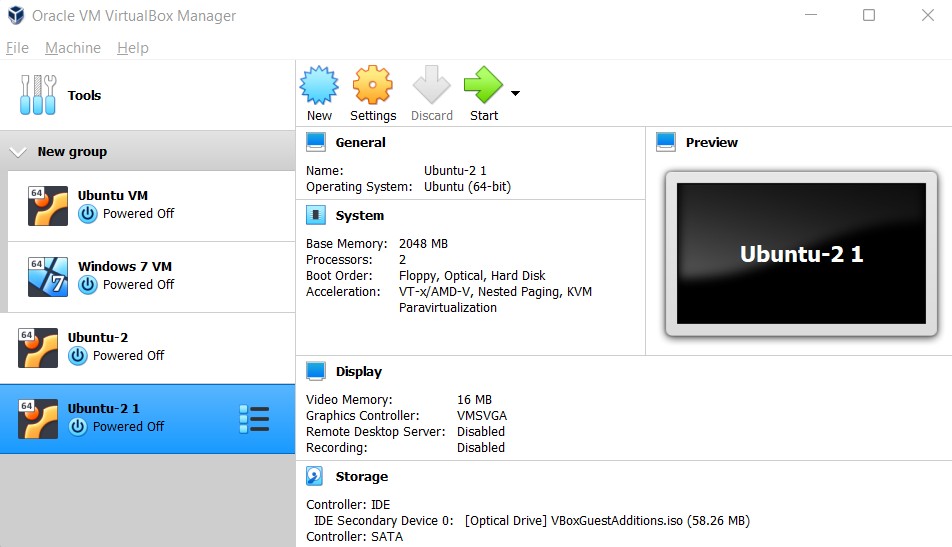
1. Start VirtualBox
2. Click File --> Import Appliance



1. Select the OVA file and click next
2. Leave the default settings and Click import
3. Wait for the import to finish
4. The machine

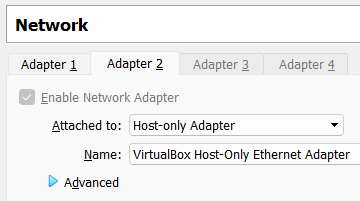
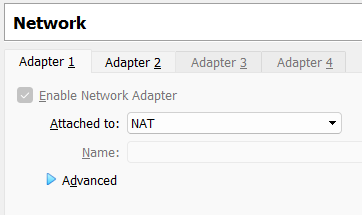
will be seen on the

VirtualBox homepage

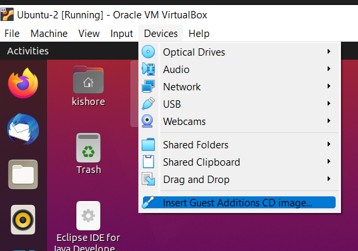


# Exercise 7 Creating a shared folder

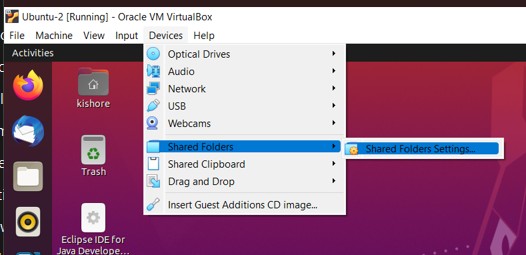
1. Configuring the network settings



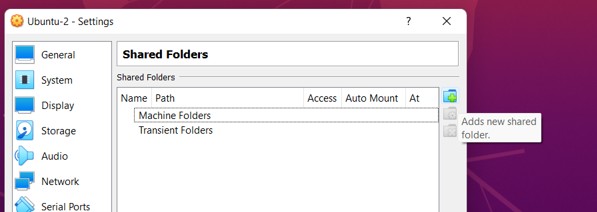
1. Install Guest addition images



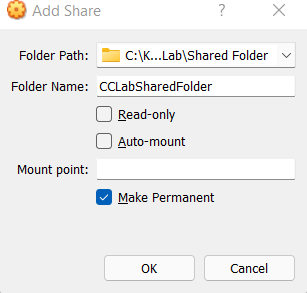
1. Go to shared folder settings



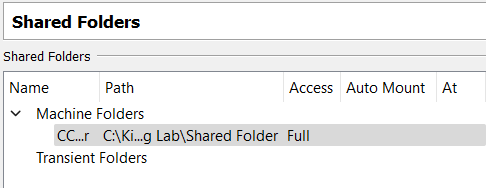
1. Click on Add new Shared folder



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

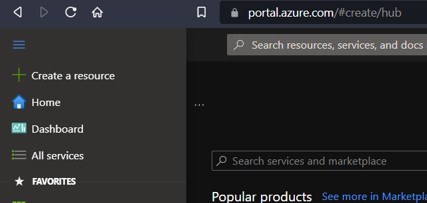


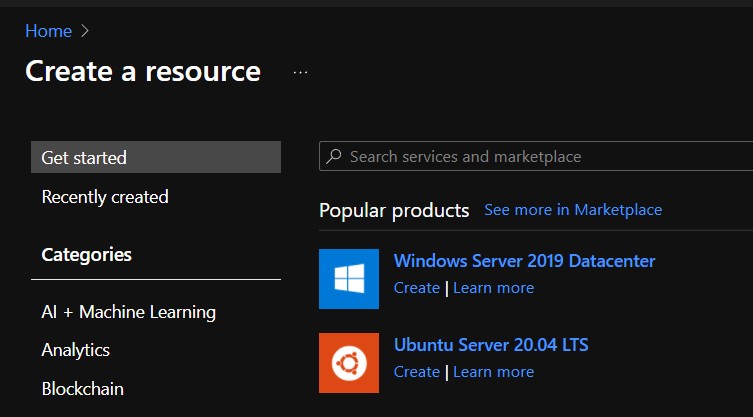
1. Permanent folder will be under machine folders, if not under Transient folder

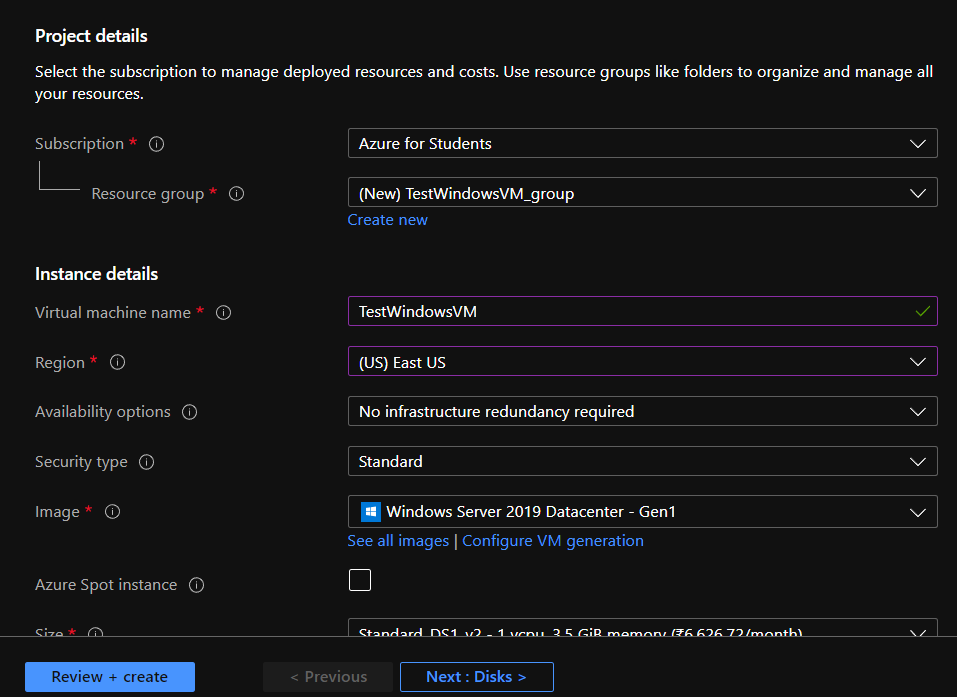


# Exercise 8 Creating a VM on Azure

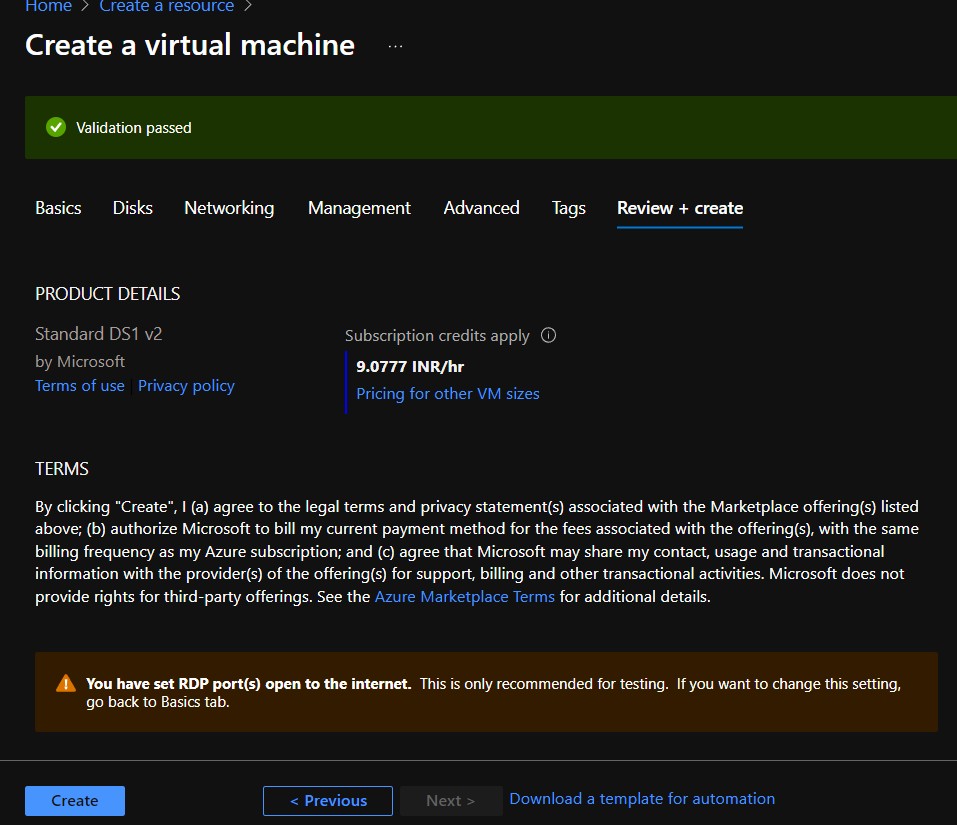
1. Click on the three horizontal bars to see the create resource icon.
2. Click on create resource

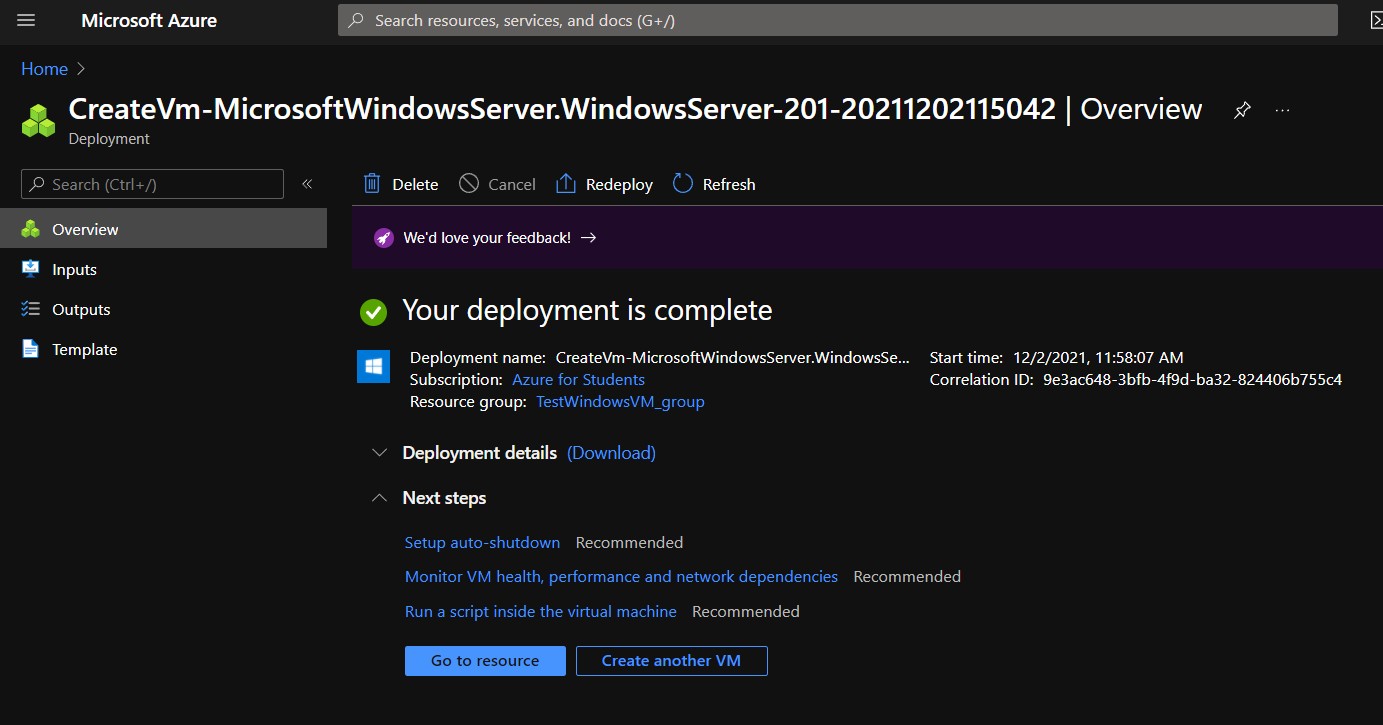


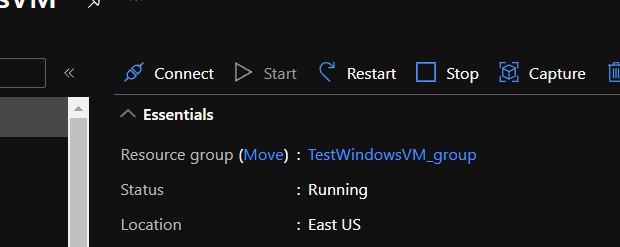
1. Select windows server
2. Fill out the details



1. Click review and create
2. Click on create

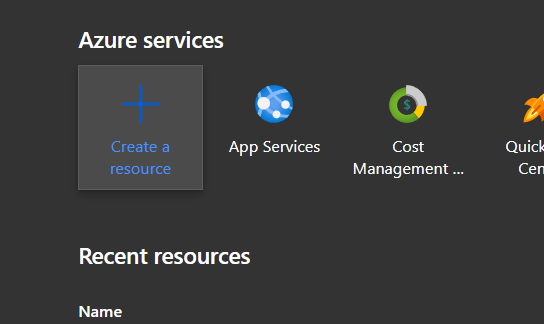


1. Click on Go to resource once deployed
2. Click connect to remotely connect using SSH or RDP (If needed)

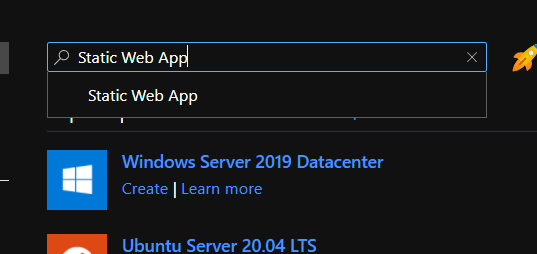


# Exercise 9 Creating a static web app in Azure Portal

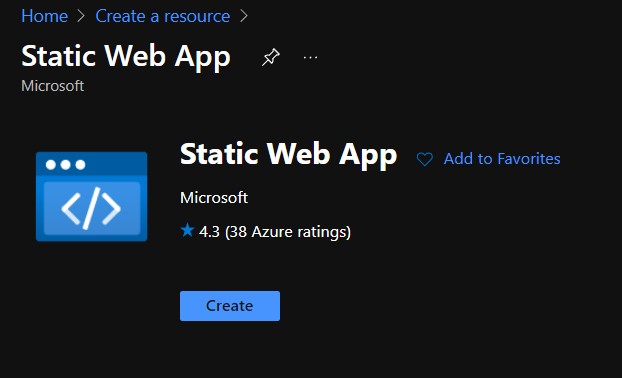
1. Click on Create Resource



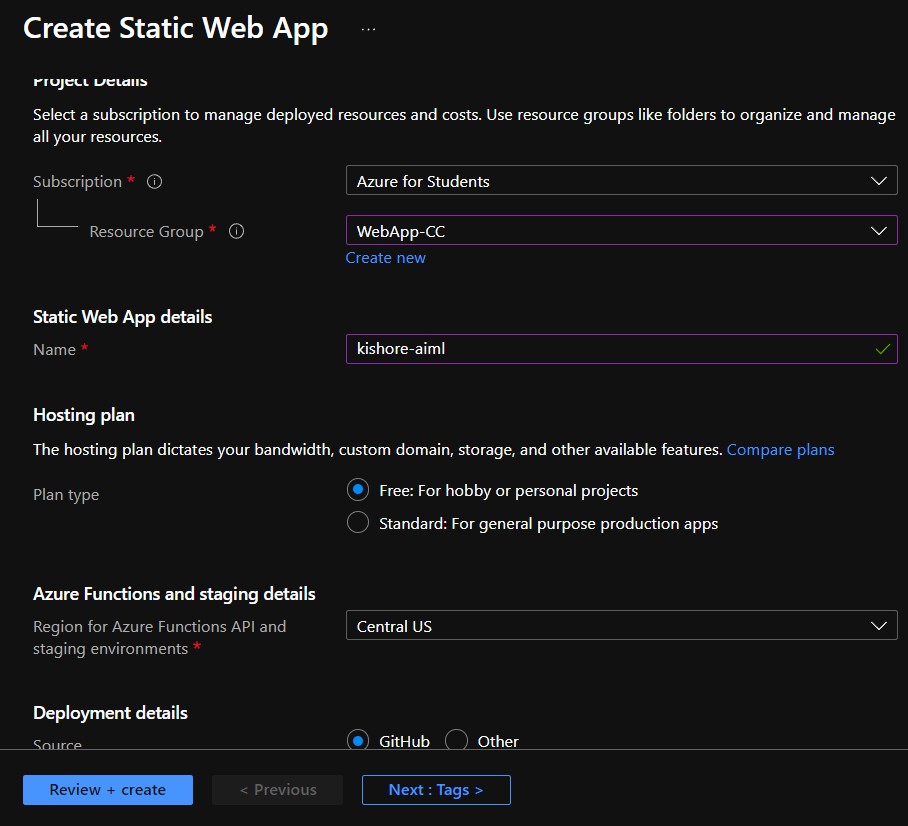
1. Search for Static Web App



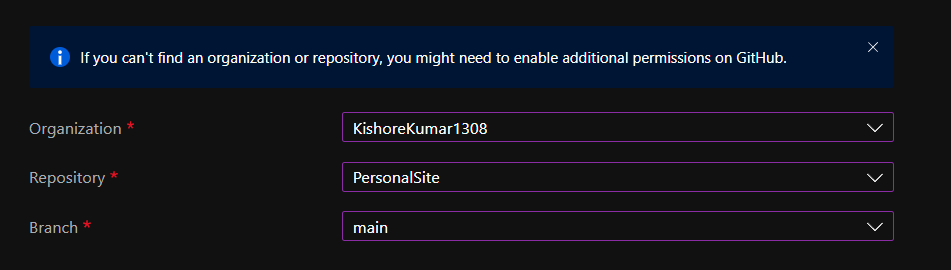
1. Select Static web app
2. Select Create



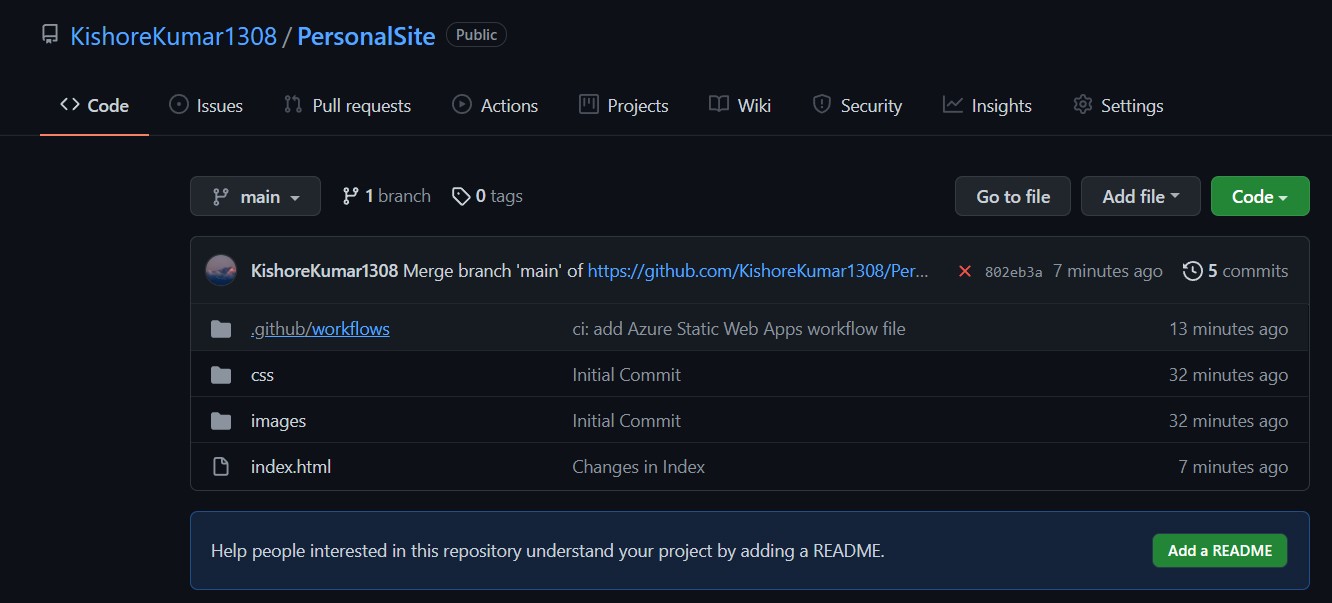
1. Fill out the details



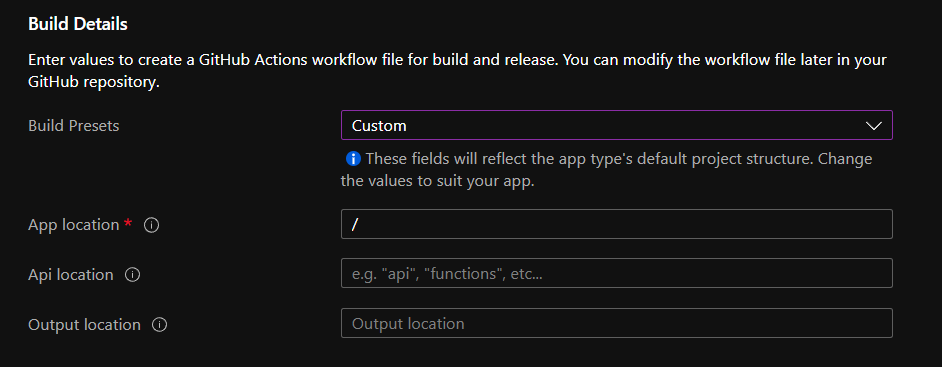
1. Sign in via GitHub
2. Select organization, Repository and Branch where the HTML file is located



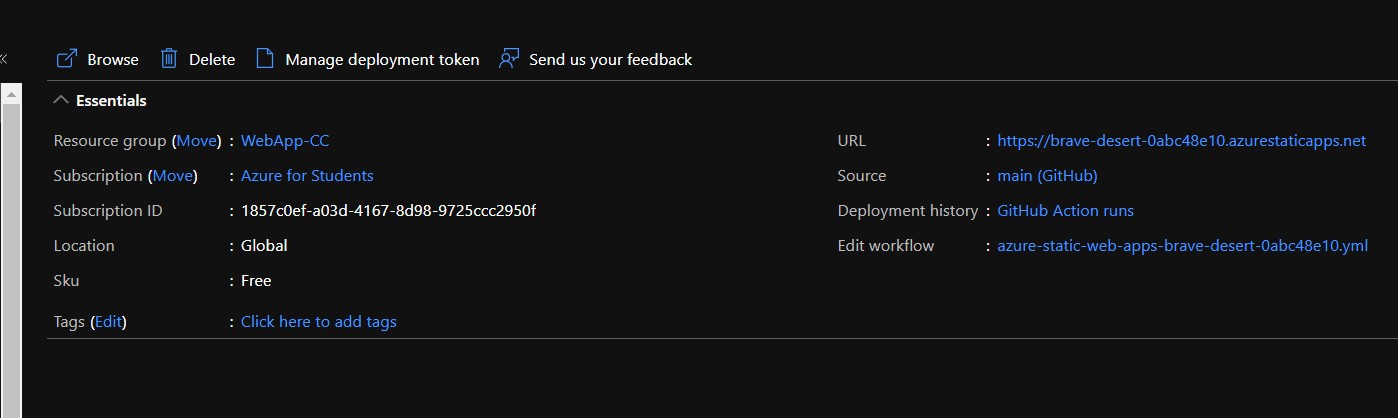
1. The GitHub repo should look like this



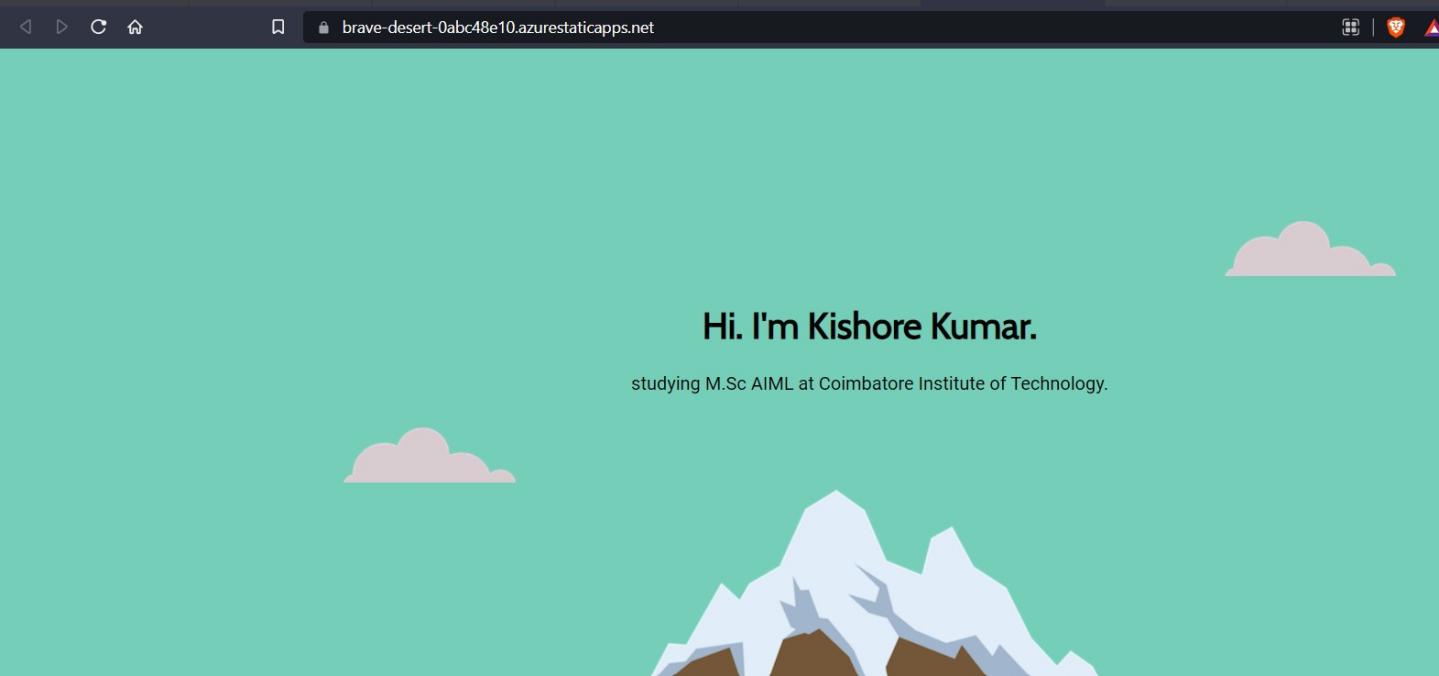
1. Fill in the following Build Details



1. Click on Review + Create
2. Click on Create
3. Click go to resource
4. The URL for the site will be available



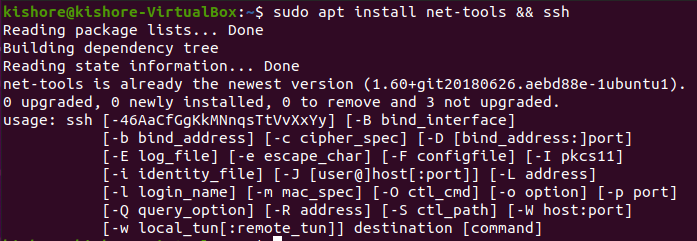
1. Click on the URL to see if it is working



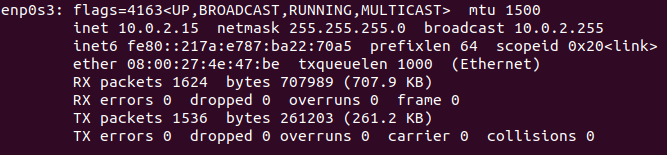
1. This is the URL for the Static Web Page created by me: <https://brave-desert-0abc48e10.azurestaticapps.net/>

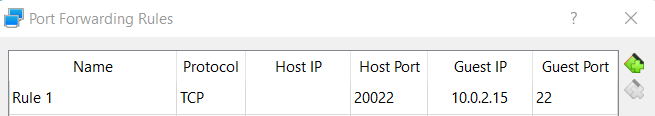
# Exercise 10 Creating a Docker Container

* 1. Install a Linux OS in VirtualBox.
  2. Install net-tools and ssh.

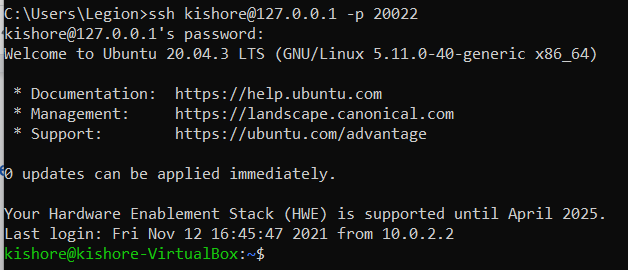


* 1. Find IP of the Linux guest machine.

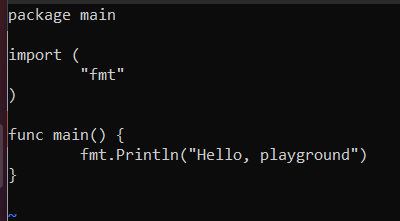


* 1. In the VirtualBox Setting for this OS, Go to Network -> Advanced -> Port Forwarding, Give Host port as 20022, Guest IP found from ifconfig, and Guest port as 22. Save the changes.
  2. In the host machine, Open Command prompt. Type ssh

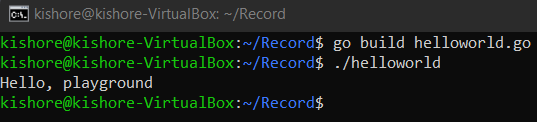
[Linux-username@127.0.0.1](mailto:Linux-username@127.0.0.1) -p 20022. Grant access if asked and type the guest Linux OS password.

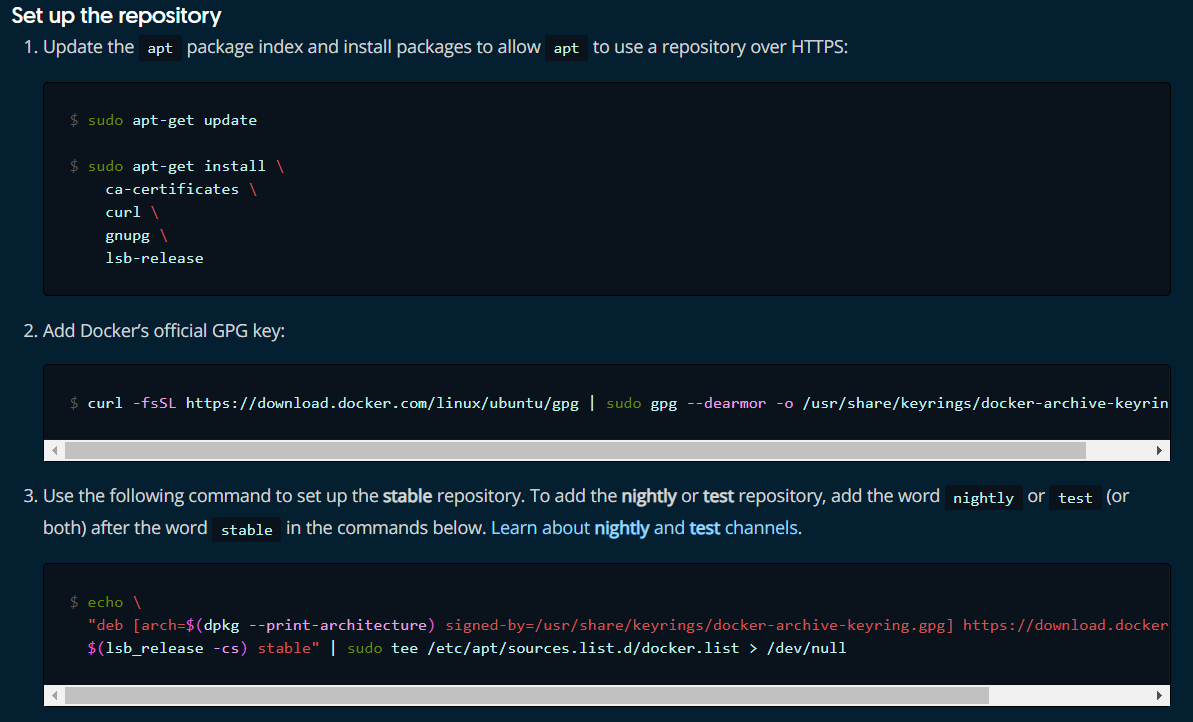


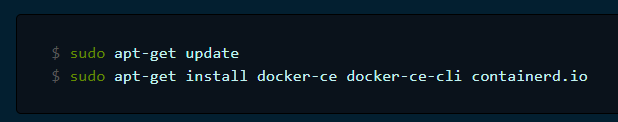
* 1. Create a simple HelloWorld program in the language of your choice. (GoLang here)



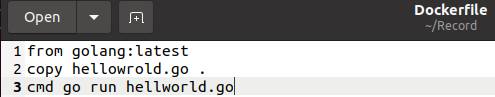
* 1. Build the program and verify the output



* 1. Go to https://docs.docker.com/engine/install/ubuntu/ and add the docker repository to the Ubuntu OS.
  2. Install Docker



* 1. In Linux create a “Dockerfile” with the following contents

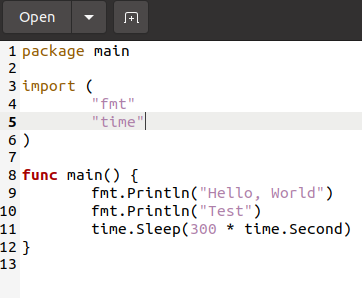


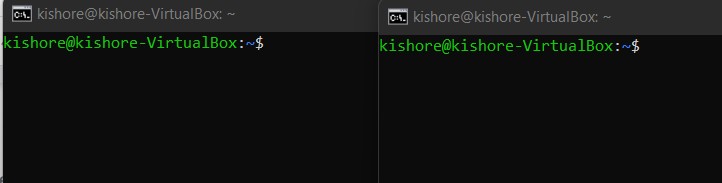
* 1. To build the container type “sudo docker build -t test:1 .”
  2. To view the image do “sudo docker images”
  3. To run the image do “sudo docker run test:1”

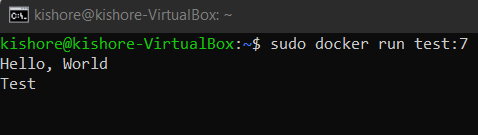


# Exercise 11 Entering into Container making Logs of the activities

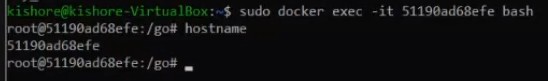
1. Modify the helloworld program to run for 5 minutes

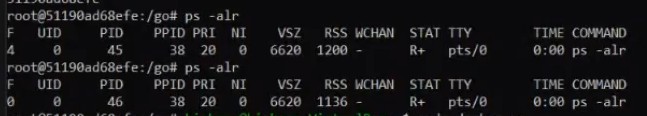


1. Open two SSH terminals from host
2. Build the modified hellworld program as a docker container and run it in SSH one terminal.

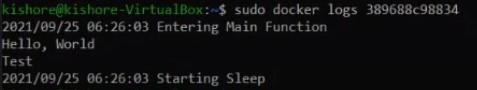


1. Enter into the running container from the other terminal using the following commands



1. To view the process running inside the container do
2. Add log comments to the file to check for any runtime errors



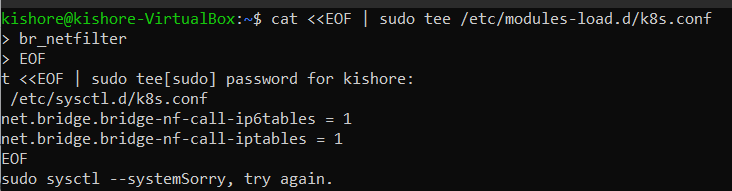
1. To view, the log messages do the following
2. Instead of using the entire golang, we can use only the required packages using alpine

# Exercise 12 Installing Kubernetes

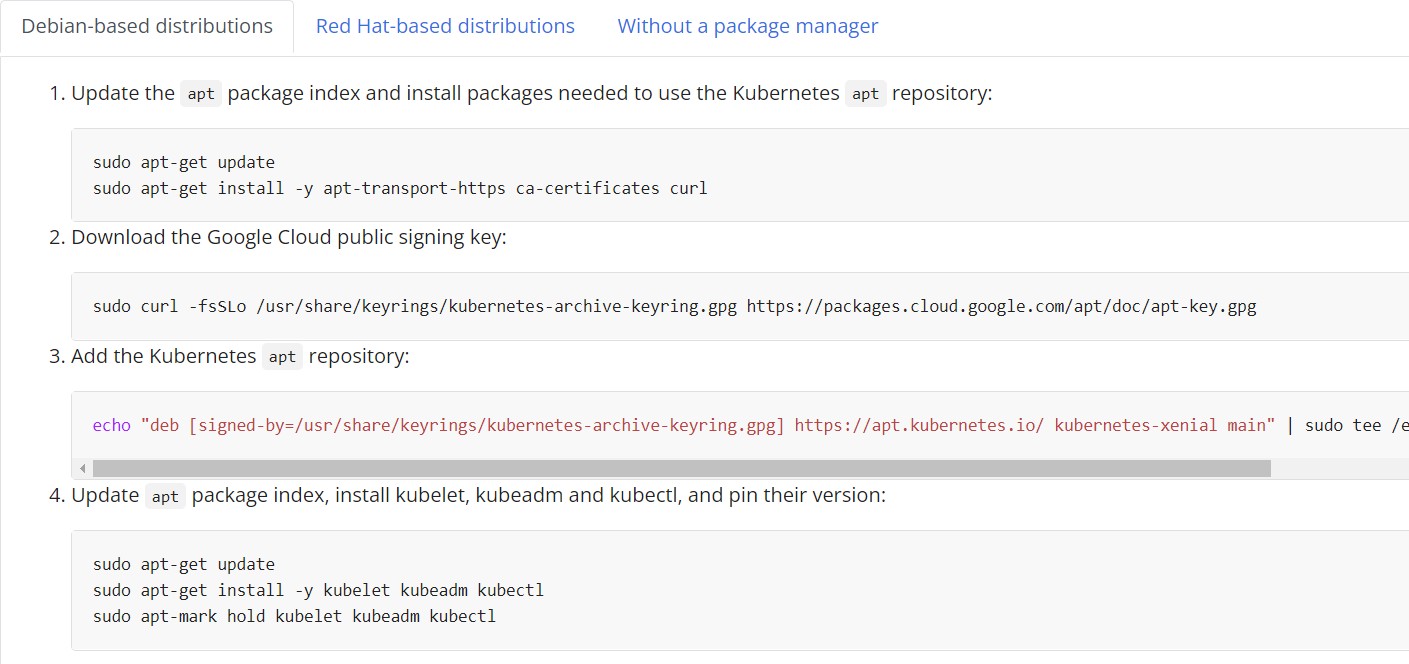
1. Go to

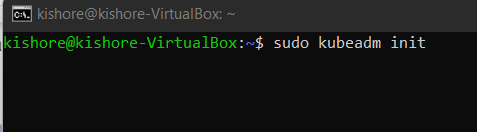
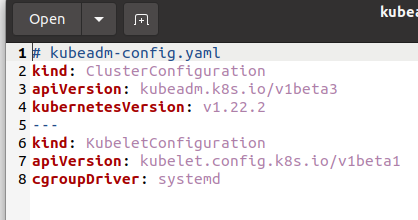
[https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-ku](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/) [beadm/](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/)

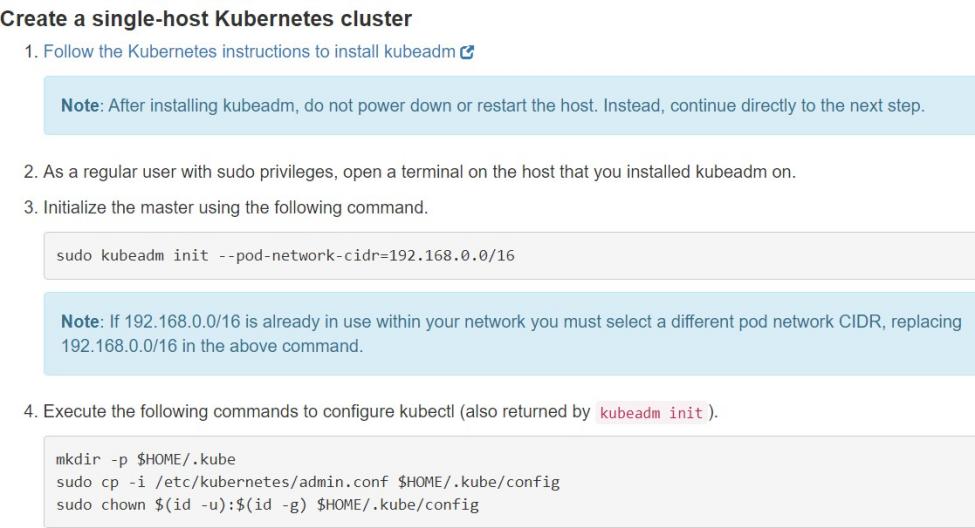
1. Install the prerequisites as mentioned in the website



1. To install kubeadm, kubectl, kubelet type the commands as mentioned in the website



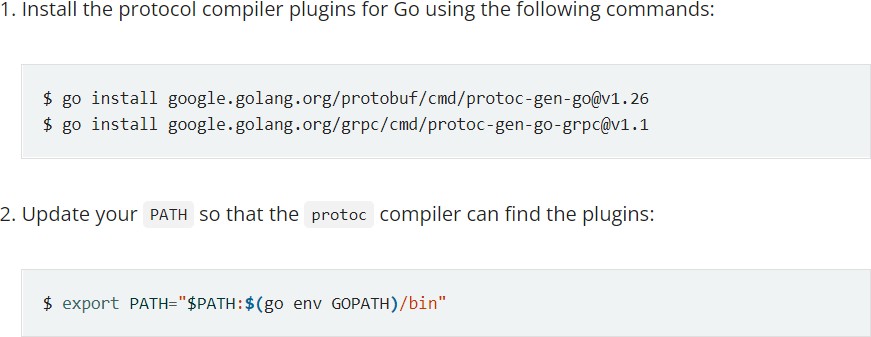
1. To initialize kubeadm, use sudo kubeadm init.
2. Turn off swap using swap -a
3. Create a kube configuration yaml file
4. Use “kubectl get pods -n kube-system” to see the pods.
5. Add Calico for Kuberentes from <https://docs.projectcalico.org/getting-started/kubernetes/quickstart>



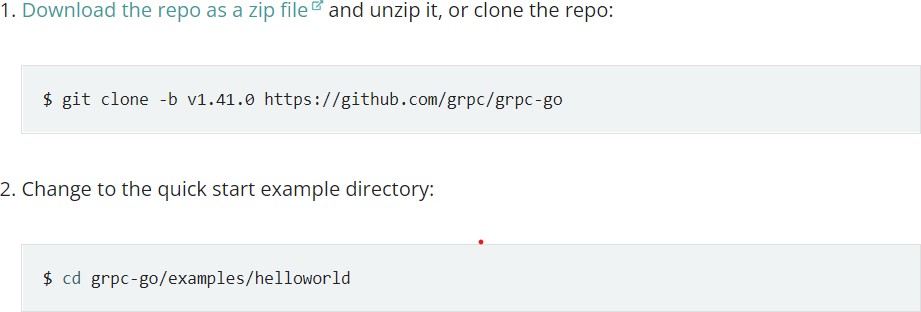
1. Create a helloworld kubernetes file by following the steps in the GitHub page <https://github.com/paulbouwer/hello-kubernetes>

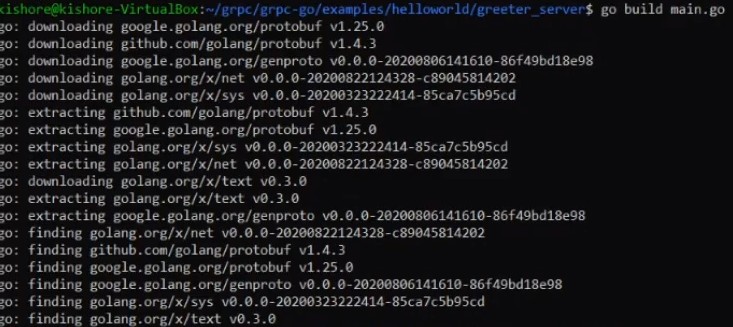
# Exercise 13 Creating a GRPC server-client program and Dockerizing it

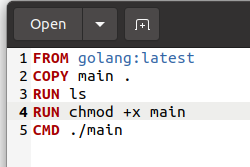
1. Install grpc plugins for golang

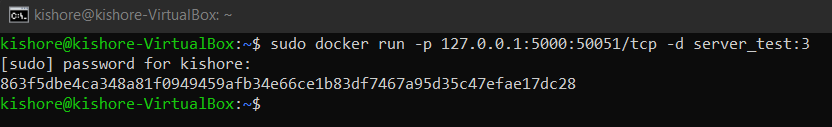


1. Get the example code

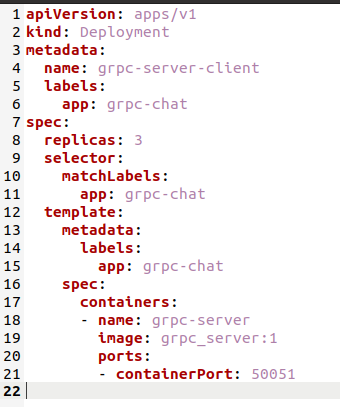


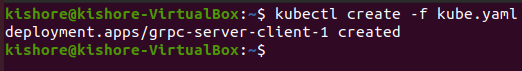
1. Build the main.go file
2. Create a docker file for the grpc server and client program

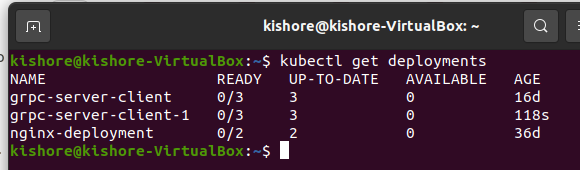


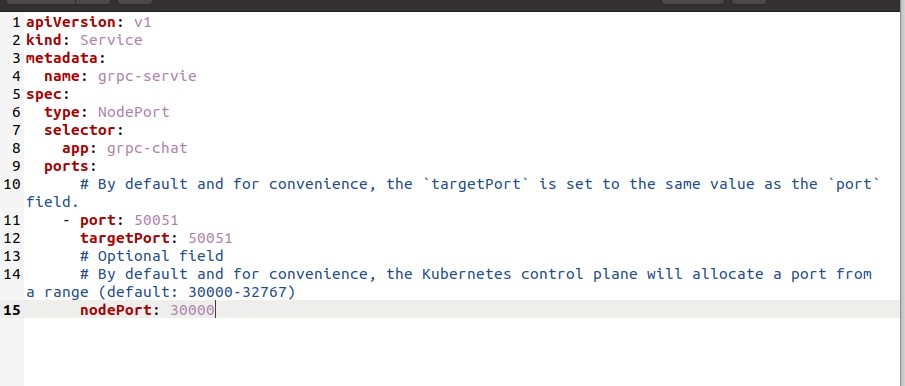
1. Build both the docker files
2. Change port as 50051 in the greeter go file and enable port forwarding in server using
3. Change the listening port in client as 5000 and run the main.go file

# Exercise 14 Containerizing the GRPC server-client program using Kubernetes

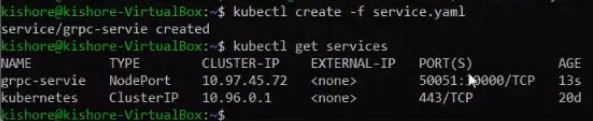
1. Create a Kubernetes deployment YAML file from <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
2. Make the necessary changes in the file as below
3. Create the deployment using



1. See the deployments using kubectl get deployments
2. Create service.yaml file from <https://kubernetes.io/docs/concepts/services-networking/service/>



1. Deploy the services



1. Change the IP in the greeter\_client file and run the program to see the results

