M23CSA020

RISHABH JOHRI

VIRTUALIZATION AND CLOUD COMPUTING

ASSIGNMENT - 1

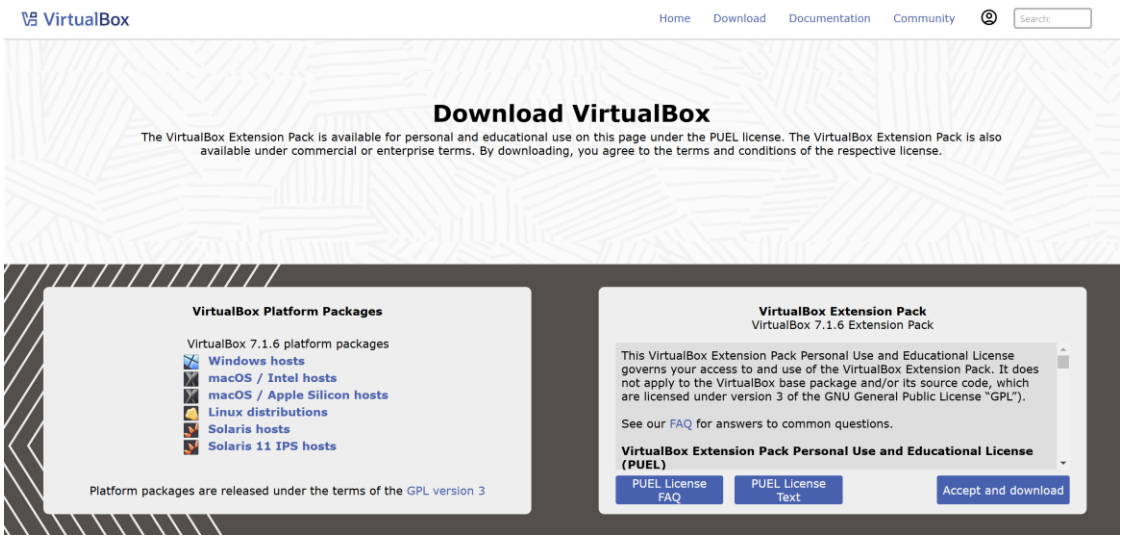
Installation of VirtualBox :

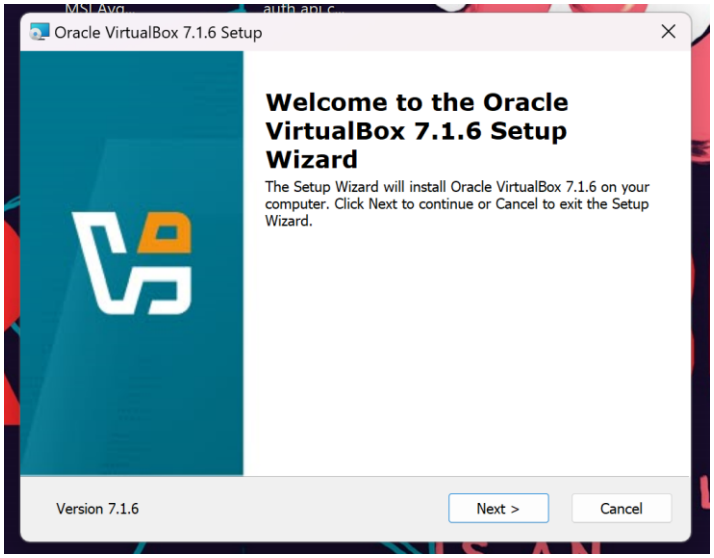
1) Go to https://www.virtualbox.org/wiki/Downloads

2) Select our OS : Windows in this case

3) Open the installer

4) Don’t disturb the default options and just keep clicking next and finally click ‘finish’

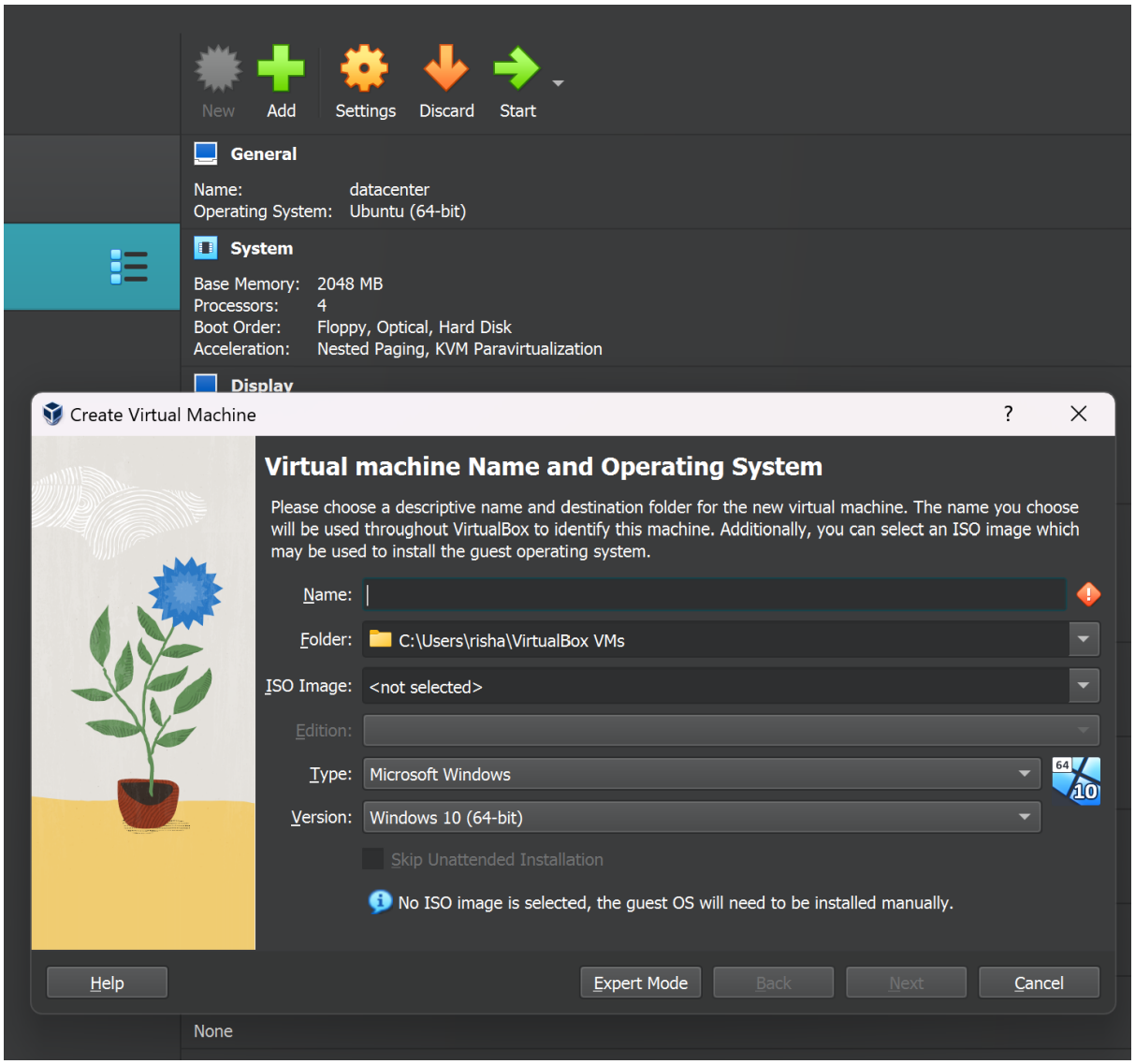




VM creation :

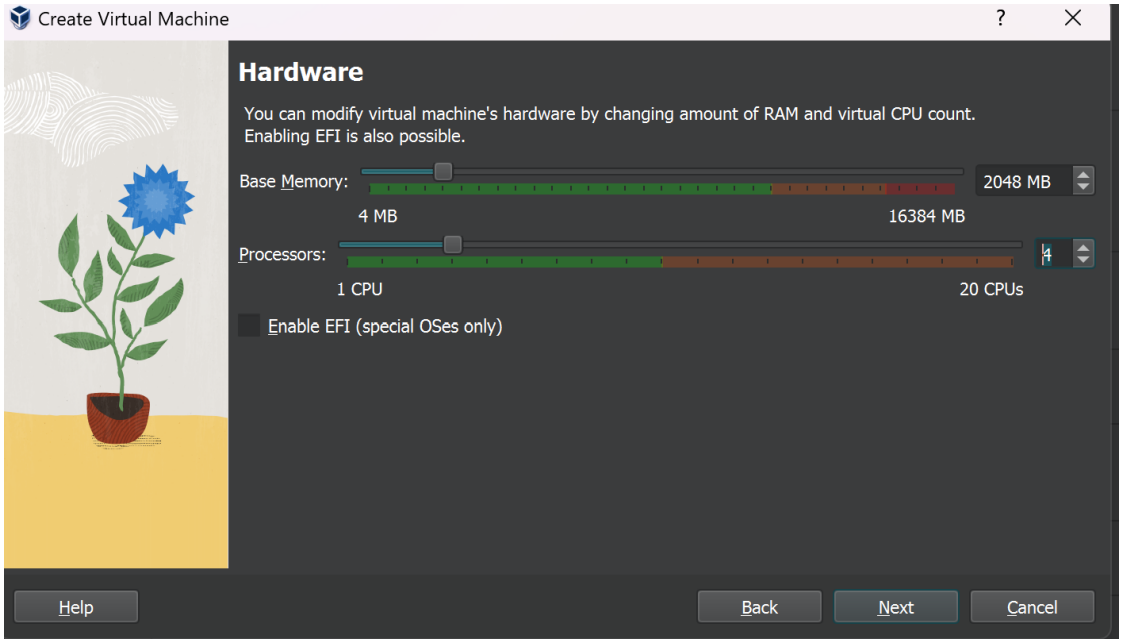
1) Open virtualbox, click on “new”

2)This will prompt a new dialogbox:

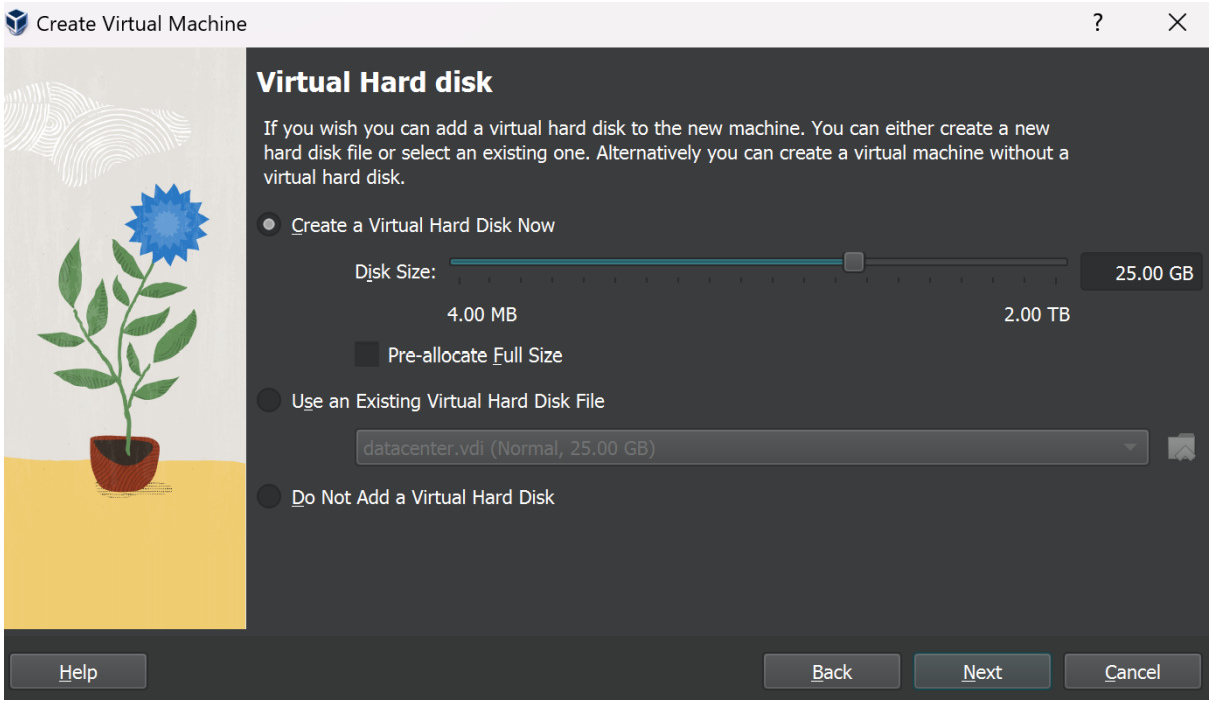


3) Fill in the name , select iso file : We have used lubuntu 24.04.1 desktop 64amd iso file. Click next.

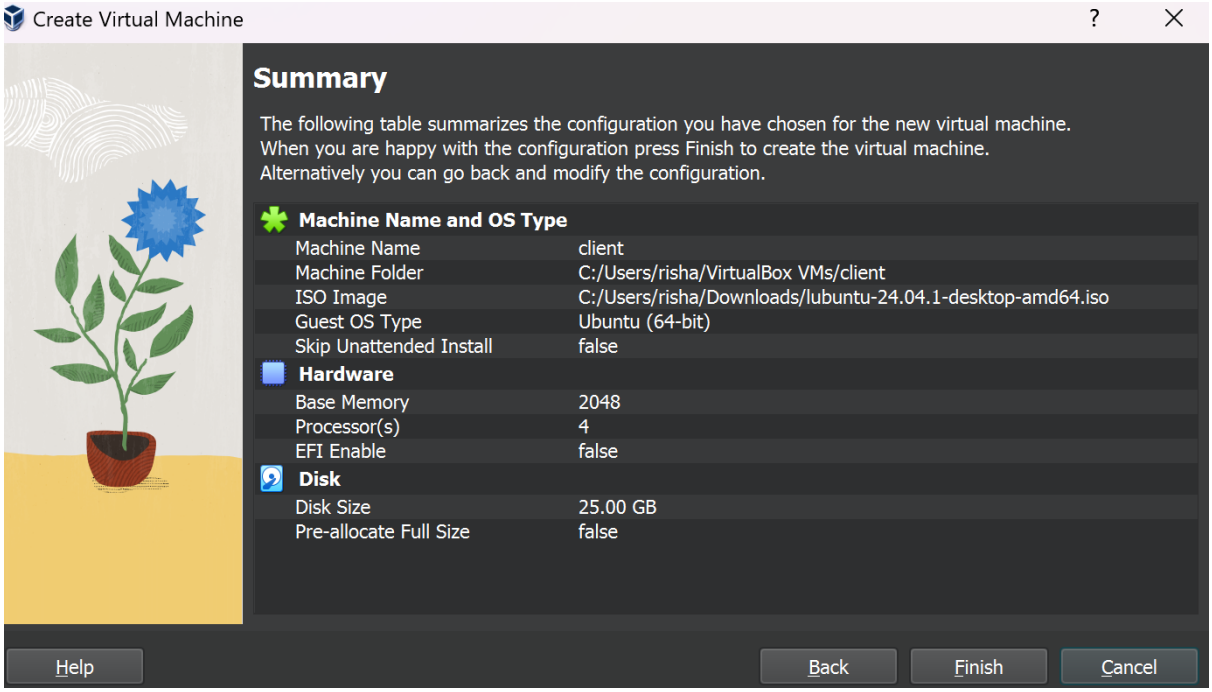
4)Allocate 2048 MB (default) base memory , 4 processors (cpu) and click next.

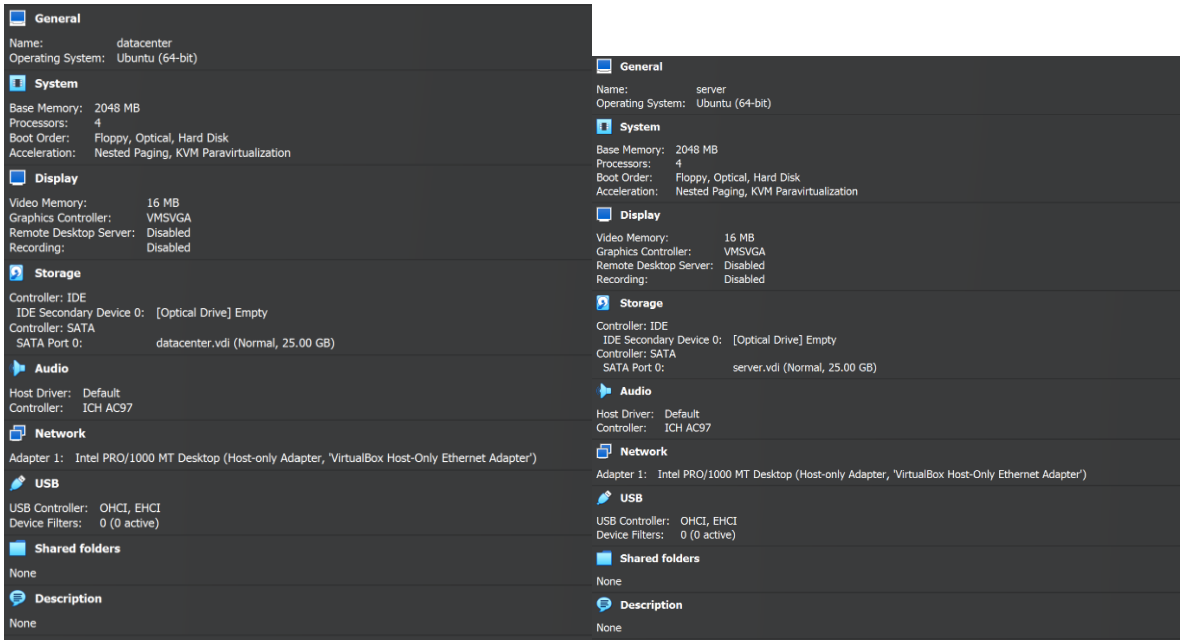


5) allocate 25 GB virtual harddisk(default),click next.



6) Verify summary , click finish

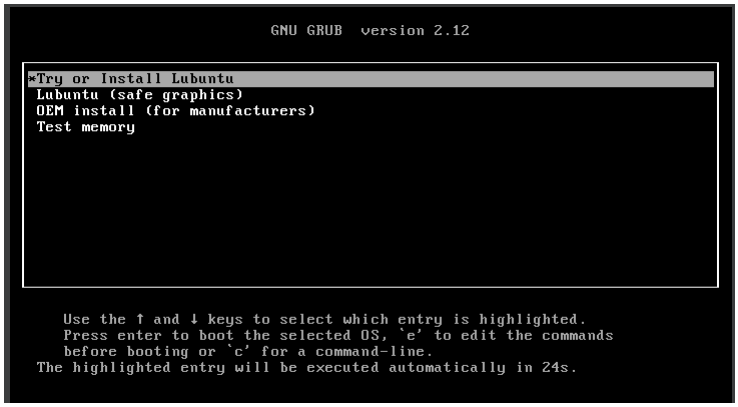




7)

Power up the VM , Install Lubuntu.

Click “Try/install lunbuntu.



Click install lubuntu

Click next foe everything while keeping default options undisturbed.

Click on “Finish and reboot”

Network Settings of VM :

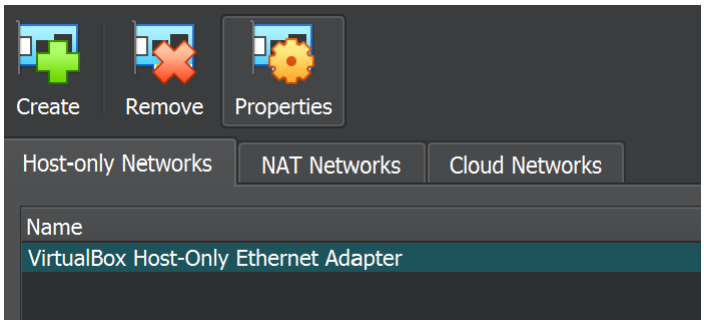
1) While downloading dependencies (when we want to connect to Internet) : NAT

2) When we want to the VMs to communicate with each other , host : Host only

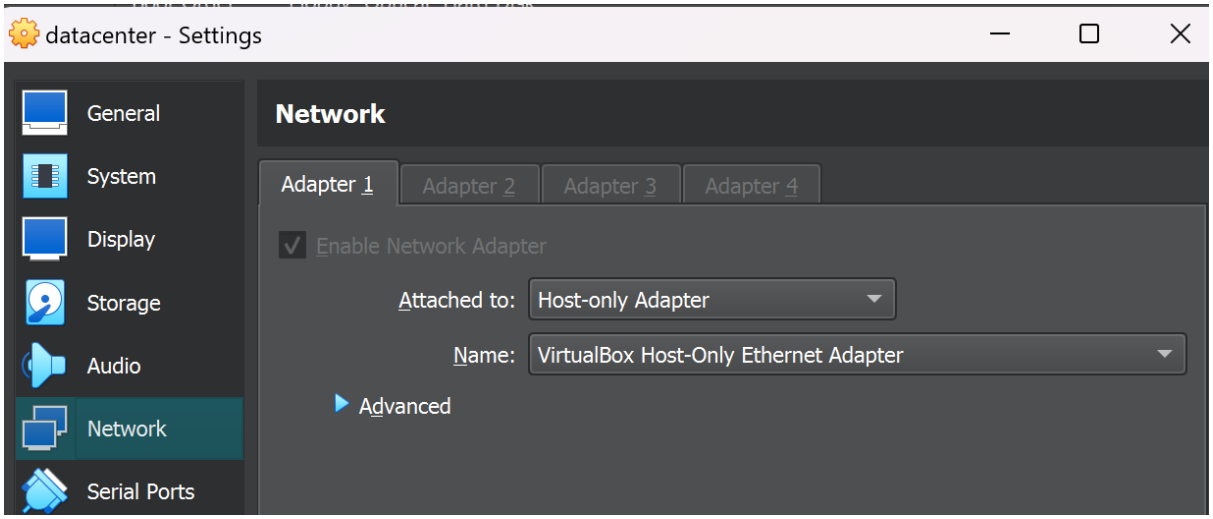
3) We can test the VM by using Ping :

_> $ ping <ip add. Of other vm> -c <no. of pings>

To create a network , use file -> tools-> network manager ->create



Go to VM settings -> network and select connection type:



Deployment of Microservice :

Do this for both VMs :

Install docker :

(NOTE : if it's showing permission denied use sudo before every command)

sudo apt update && sudo apt install -y docker.io

For datacenter VM:

Install mongodb using docker and run it as a container :

Pull MongoDB Docker image

docker pull mongo:4.2

Create a Docker network

docker network create auth-network

Run MongoDB container

docker run -d --name mongodb -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=adminpass --restart unless-stopped mongo:4.2

Verify if MongoDB is running:

docker ps

docker exec -it mongodb mongo

#NOTE : Above command will start mongo shell -> To quit the shell , type quit()

Now , you can change datacenter's connection type from NAT -> to Host only :

Also keep in mind to note down the IP address of both VMs : use "ip a" or "ifconfig" command and try PING command to test inter VM communication.

NOTE : Also change this IP address in this repo : server : Authentication-Microservice-FASTAPI -> database.py -> MONGO_URI (make sure to use the datacenter IP where mongobd is running)

For server VM:

```
git clone https://github.com/rishabhjohri/Authentication-Microservice-FASTAPI.git

cd Authentication-Microservice-FASTAPI

docker build -t auth-service .

# Create a Docker network

docker network create auth-network
```

Switch to Host only connection now

```
# Run the Authentication Microservice

docker run -d --name auth-container --network auth-network -p 8000:8000 -e MONGO_URI="mongodb://admin:adminpass@192.168.56.107:27017/auth_service?authSource=admin" auth-service

# Verify if auth-container is running:

docker ps
```

To test the Authentication Microservice :

You can either create a Client VM or even do it from your local host machine (I used my host windows cmd to run this) :

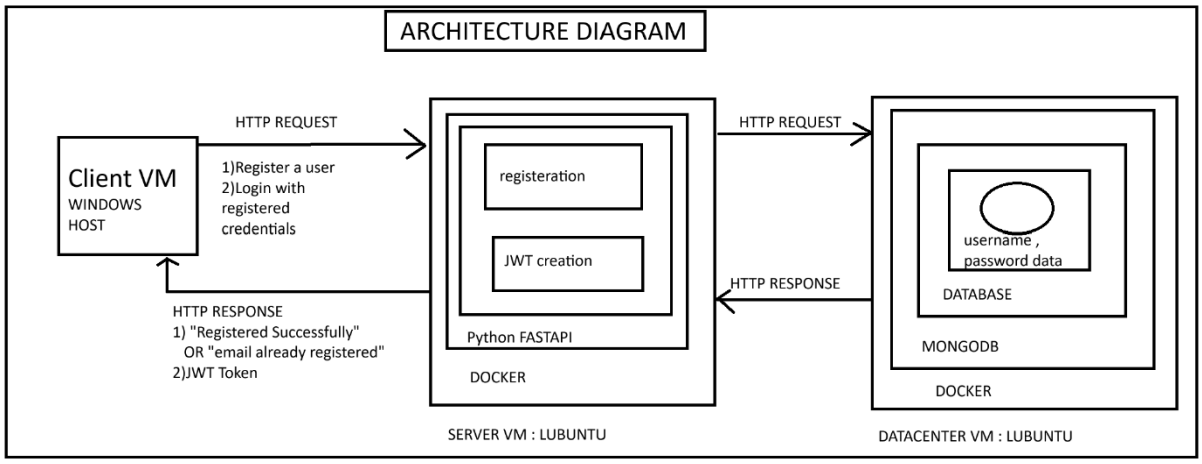
```
# Register a user

curl -X POST "http://192.168.56.108:8000/api/auth/register" -H "Content-Type: application/json" -d "{\"username\": \"testuser\", \"email\": \"test@example.com\", \"password\": \"securepassword\"}"

# Login with registered credentials

curl -X POST "http://192.168.56.108:8000/api/auth/login" -H "Content-Type: application/json" -d "{\"email\": \"test@example.com\", \"password\": \"securepassword\"}"
```

Architecture Diagram :



The architecture consists of three VMs: Client VM (Windows Host), Server VM (Lubuntu with FastAPI), and Datacenter VM (Lubuntu with MongoDB), all connected via VirtualBox. The Client VM sends HTTP requests for user registration and login to the Server VM, which runs a FastAPI-based authentication microservice inside Docker. The Server VM forwards authentication data to the Datacenter VM , where MongoDB stores and retrieves user credentials. Upon successful login, a JWT token is issued. This architecture ensures secure, scalable authentication using Docker, FastAPI, MongoDB, and JWT, enabling microservice-based user authentication across multiple VMs efficiently.

GitHub Repo link :

https://github.com/rishabhjohri/Authentication-Microservice-FASTAPI

Demo video link :

https://youtu.be/jGAxaTDG5hI
