Virtualization and Cloud computing

Amresh Kumar   M23CSA004

Assignment- 01

Github link:https://github.com/m23csa004/vcc_assignment1
Video link:🎬 m23csa004_vcc_assign1.mkv

1.Introduction:

This report provides a step-by-step guide to deploying a microservice-based application using VirtualBox with multiple virtual machines (VMs). The microservice architecture consists of three services: a **User Service**, an **Order Service**, and a **Gateway Service**, deployed across different VMs. The goal is to ensure communication between these services through a configured network..

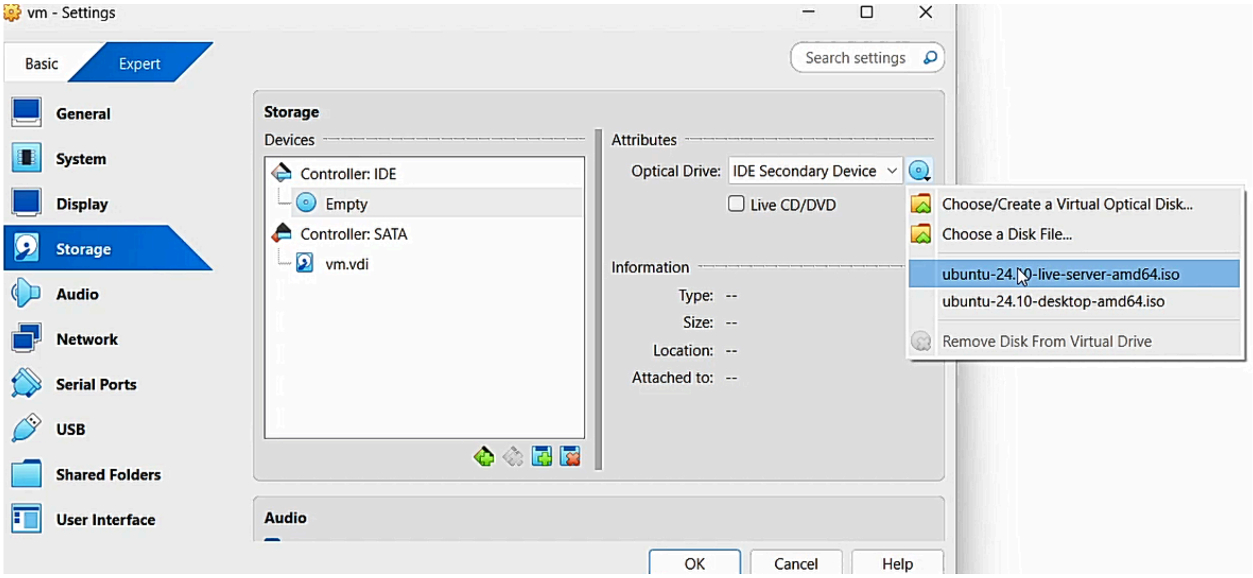
2. Installation of VirtualBox and Creation of Multiple VMs

Step 1: Install VirtualBox

To set up the virtual machines, download and install Oracle VirtualBox from the official website (https://www.virtualbox.org/).

Step 2: Create Virtual Machines

1. Open VirtualBox and click on New.
2. Provide a name for the VM (vm1,vm2,vm).
3. Choose the operating system (Ubuntu 24.10 derver from the storage menu from the setting of vm).

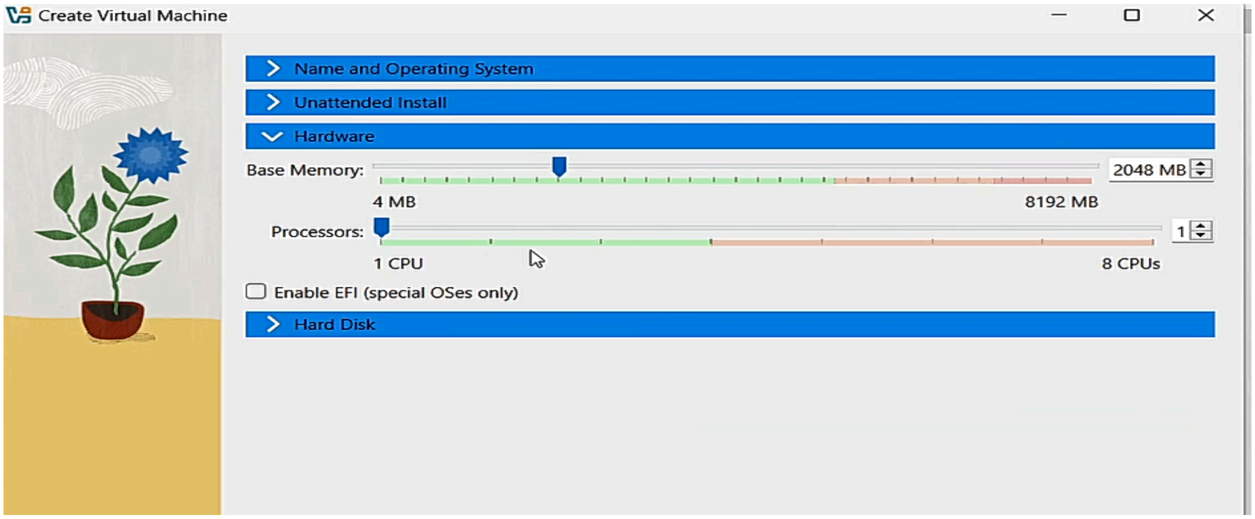


Optical Drive: IDE Secondary Device

□ Live CD/DVD

Type: --
Size: --
Location: --
Attached to: --

4. Allocate memory (at least 2GB recommended per VM).

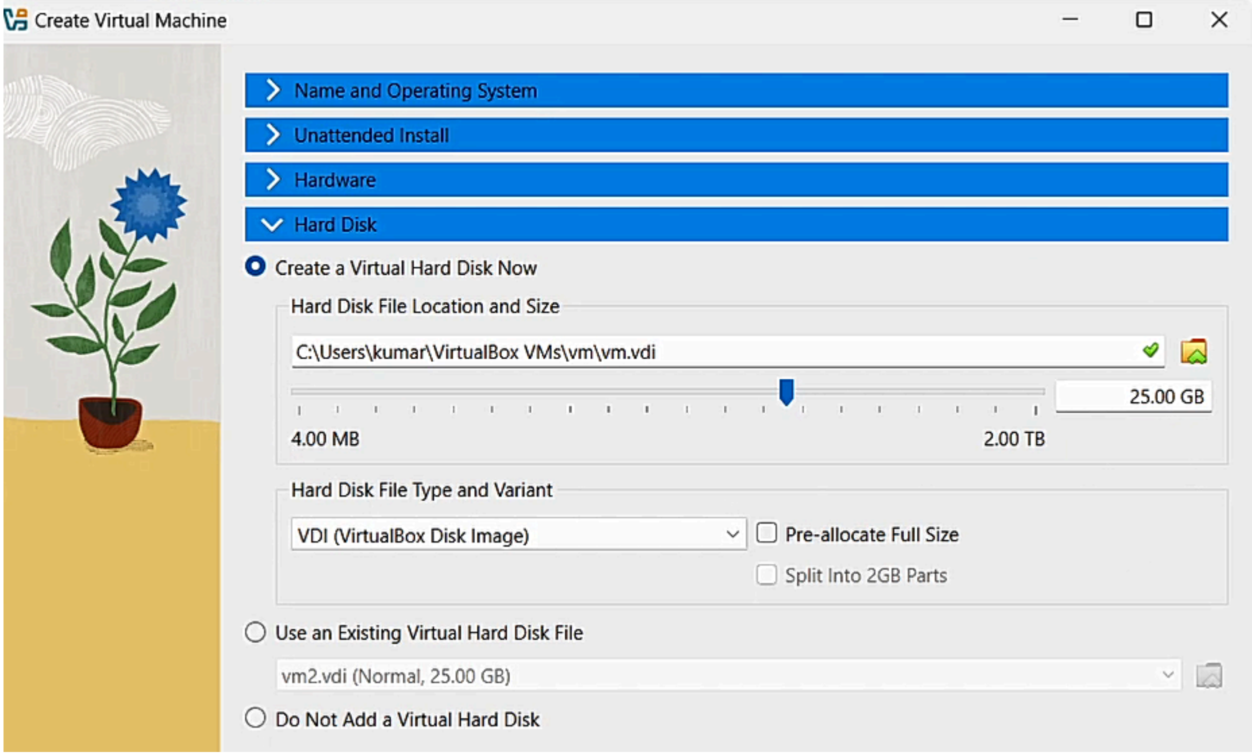


Base Memory: 2048 MB

Processors: 1

□ Enable EFI (special OSes only)

5. Create a virtual hard disk(25 GB) and set storage to dynamically allocated.



Create Virtual Machine

- Name and Operating System
- Unattended Install
- Hardware
- Hard Disk

● Create a Virtual Hard Disk Now

Hard Disk File Location and Size

C:\Users\kumar\VirtualBox VMs\vm\vm.vdi

25.00 GB

4.00 MB — 2.00 TB

Hard Disk File Type and Variant

VDI (VirtualBox Disk Image)

□ Pre-allocate Full Size

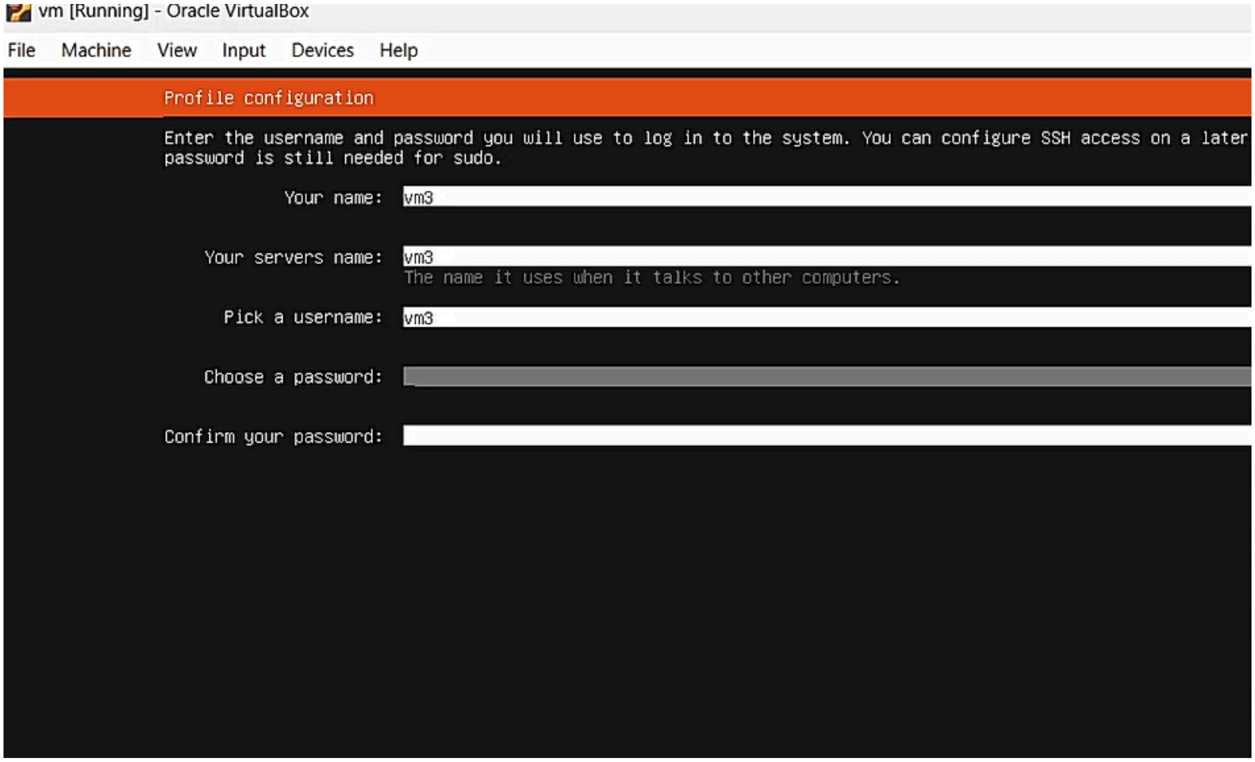
□ Split Into 2GB Parts

○ Use an Existing Virtual Hard Disk File

vm2.vdi (Normal, 25.00 GB)

○ Do Not Add a Virtual Hard Disk

6. And then proceed according to the instruction on screen(i havenot changed anything just pressed done and enter.)
7. Create the username, password for your vm


vm [Running] - Oracle VirtualBox
File Machine View Input Devices Help

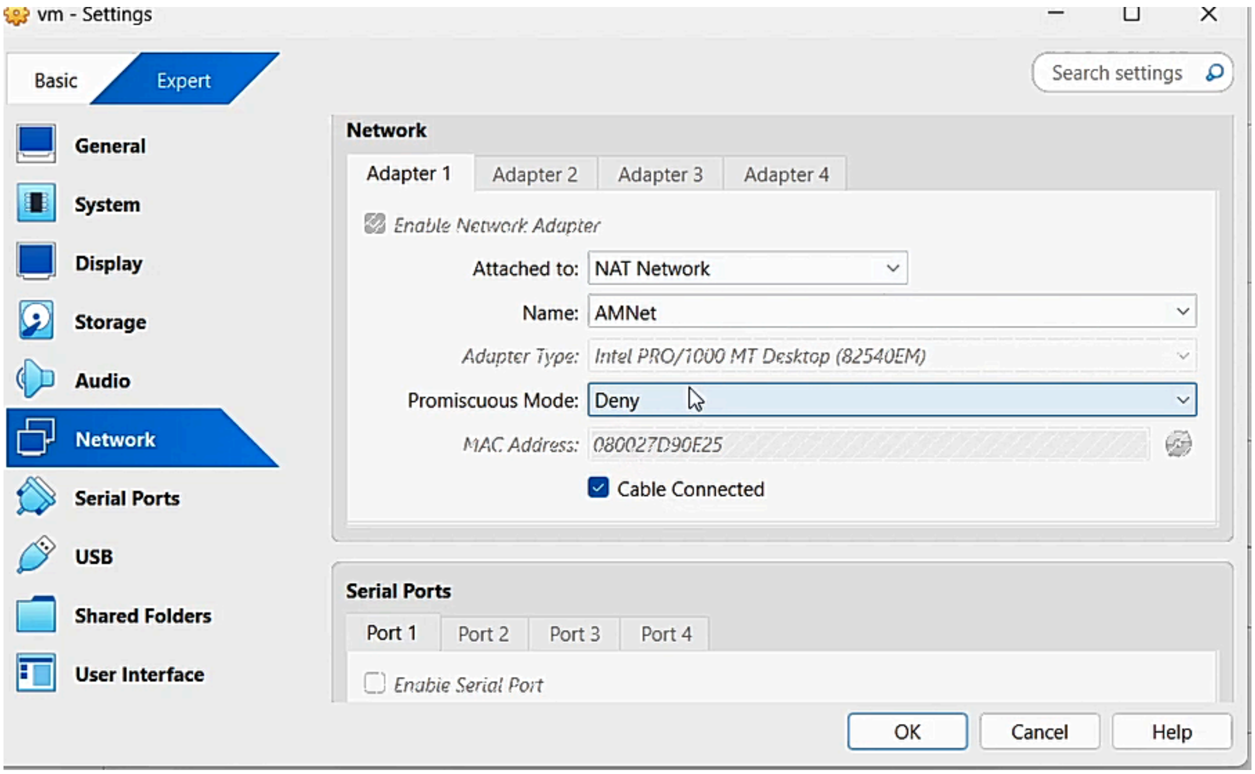
```
Profile configuration

Enter the username and password you will use to log in to the system. You can configure SSH access on a later
password is still needed for sudo.

                  Your name: vm3

          Your servers name: vm3
                             The name it uses when it talks to other computers.

            Pick a username: vm3

          Choose a password:

      Confirm your password:
```

8. And on final step it will be installed and you can check with the commands.
9. Repeat this process to create multiple VMs or clone any vm to get others.

3. Configuration of Network Settings to Connect the VMs

Step 1: Set Up Network Adapter

1. In VirtualBox, go to Settings → Network.
2. Set Adapter 1 to NAT (I have given the name as AMNET and provided the ip address dynamically as 192.168.100.4).



☑ Enable Network Adapter

Attached to: NAT Network

Name: AMNet

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 080027D90E25

☑ Cable Connected

□ Enable Serial Port

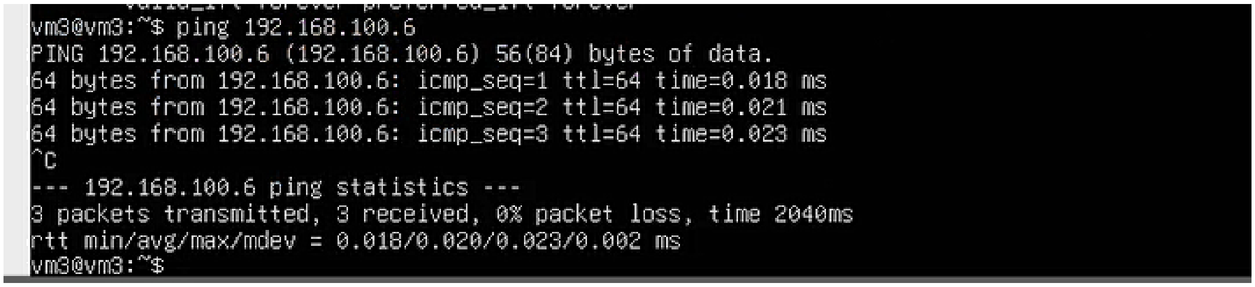
3. Enable Adapter 2, set it to internal network .
4. Apply changes and restart the VMs.
5. Once you do this then for all you just need to select AMNET and internal net and ip address will be allocated automatically.

Step 2: Verify Network Configuration

Check the assigned IP addresses on each VM: ip a

And then check each other is communicating with each other using

Ping followed by ip address of other vm



```
       valid_lft forever preferred_lft forever
vm3@vm3:~$ ping 192.168.100.6
PING 192.168.100.6 (192.168.100.6) 56(84) bytes of data.
64 bytes from 192.168.100.6: icmp_seq=1 ttl=64 time=0.018 ms
64 bytes from 192.168.100.6: icmp_seq=2 ttl=64 time=0.021 ms
64 bytes from 192.168.100.6: icmp_seq=3 ttl=64 time=0.023 ms
^C
--- 192.168.100.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.018/0.020/0.023/0.002 ms
vm3@vm3:~$
```

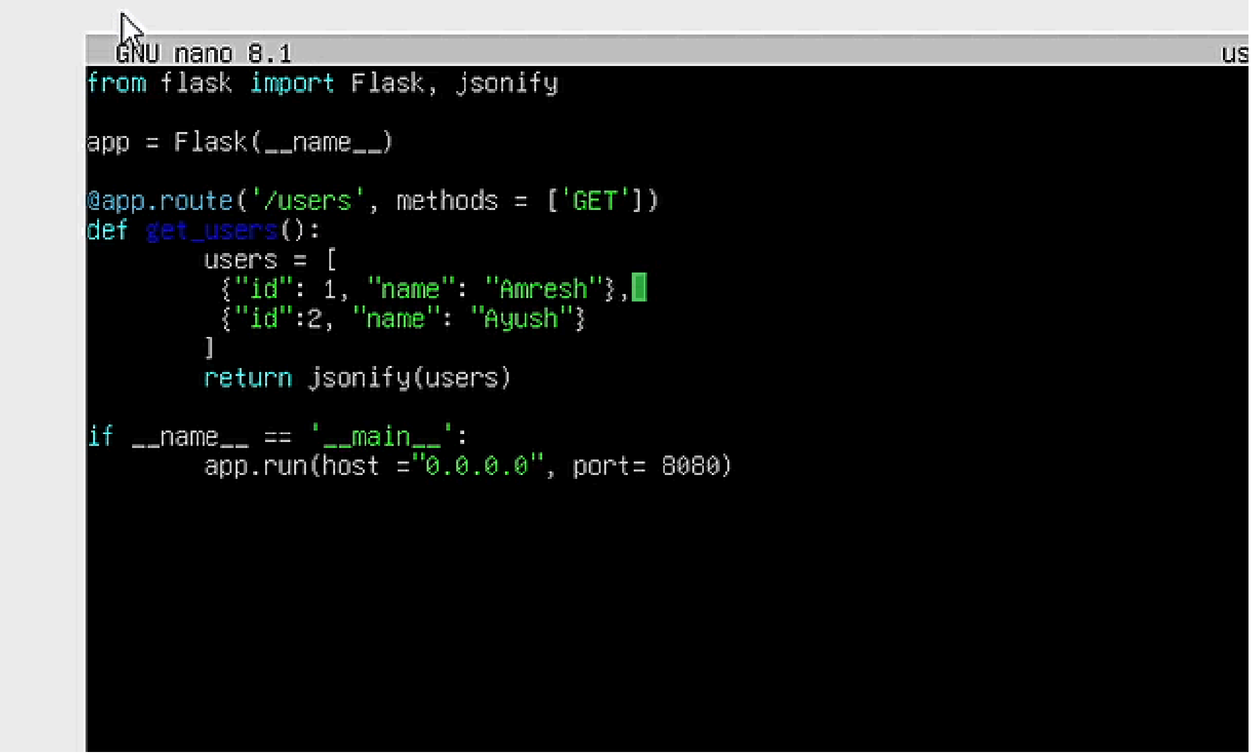
<u>4. Deployment of the Microservice Application</u>

First install all dependencies like **pip, flask** using
Sudo apt install **python3-pip**
Sudo apt install **python3-flask**
And then make the directory according to yourself like mkdir **micrservice2**
And then go to editor using **nano userinfo.py, nano orderhandler.py, nano gateway.py**

<u>The application consists of three microservices:</u>

1. **<u>User Service (VM1) – Provides user data.</u>**
2. **<u>Order Service (VM2) – Manages orders.</u>**
3. **<u>Gateway Service (VM3) – Acts as an API gateway to route requests.</u>**

<u>User Service (userinfo.py on VM1)</u>



```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/users', methods = ['GET'])
def get_users():
        users = [
         {"id": 1, "name": "Amresh"},
         {"id":2, "name": "Ayush"}
        ]
        return jsonify(users)

if __name__ == '__main__':
        app.run(host ="0.0.0.0", port= 8080)
```

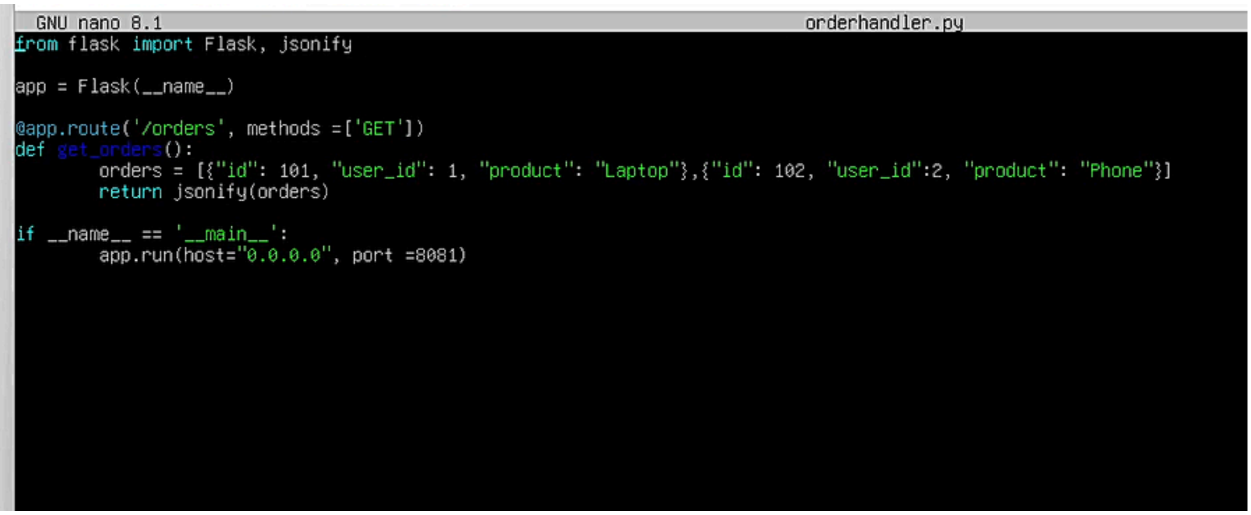
This service is a simple Flask API that returns a list of users.

Code has been pushed to github repo.

Ip address of this vm is 192.168.100.4 and port no. of this service is 8080

You need to run the code **python3 userinfo.py**

Order Service (orderhandler.py on VM2): This service provides order details.

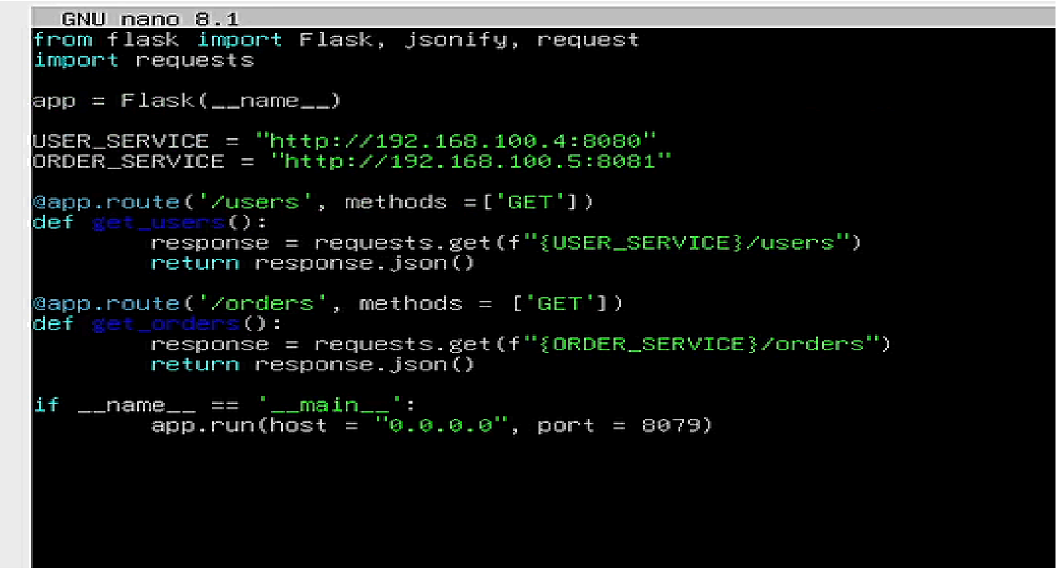


```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/orders', methods =['GET'])
def get_orders():
        orders = [{"id": 101, "user_id": 1, "product": "Laptop"},{"id": 102, "user_id":2, "product": "Phone"}]
        return jsonify(orders)

if __name__ == '__main__':
        app.run(host="0.0.0.0", port =8081)
```

Run the service using: **python3 orderhandler.py**

Gateway Service (gateway.py on VM3)

This service acts as an API gateway, forwarding requests to the appropriate services.



```
from flask import Flask, jsonify, request
import requests

app = Flask(__name__)

USER_SERVICE = "http://192.168.100.4:8080"
ORDER_SERVICE = "http://192.168.100.5:8081"

@app.route('/users', methods =['GET'])
def get_users():
        response = requests.get(f"{USER_SERVICE}/users")
        return response.json()

@app.route('/orders', methods = ['GET'])
def get_orders():
        response = requests.get(f"{ORDER_SERVICE}/orders")
        return response.json()

if __name__ == '__main__':
        app.run(host = "0.0.0.0", port = 8079)
```

Run using: **python3 gateway.py**

5. Testing the Microservices

Once all three services are running, test the connections from any VM.

Check if User Service is Running

Unset

```
curl -X GET http://192.168.100.4:8080/users
```

Check if Order Service is Running

Unset

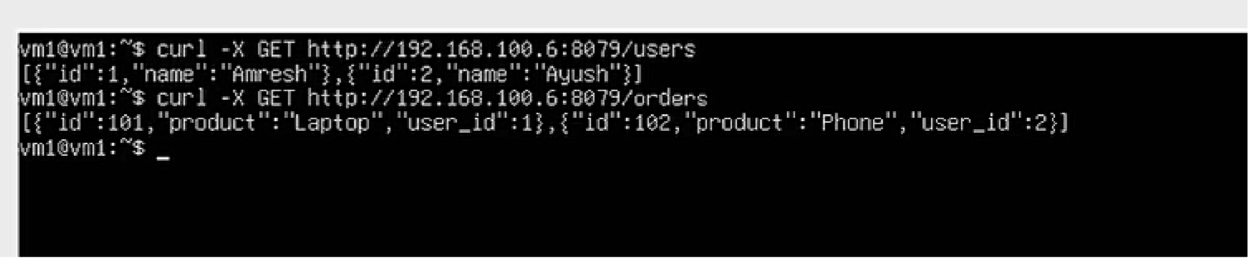
```
curl -X GET http://192.168.100.5:8081/orders
```

Check Gateway Service Routing

You can run these from any vm just open any new terminal on any vm by ctrl+alt+Fn2

Unset

```
curl -X GET http://192.168.100.6:8079/users
curl -X GET http://192.168.100.6:8079/orders
```



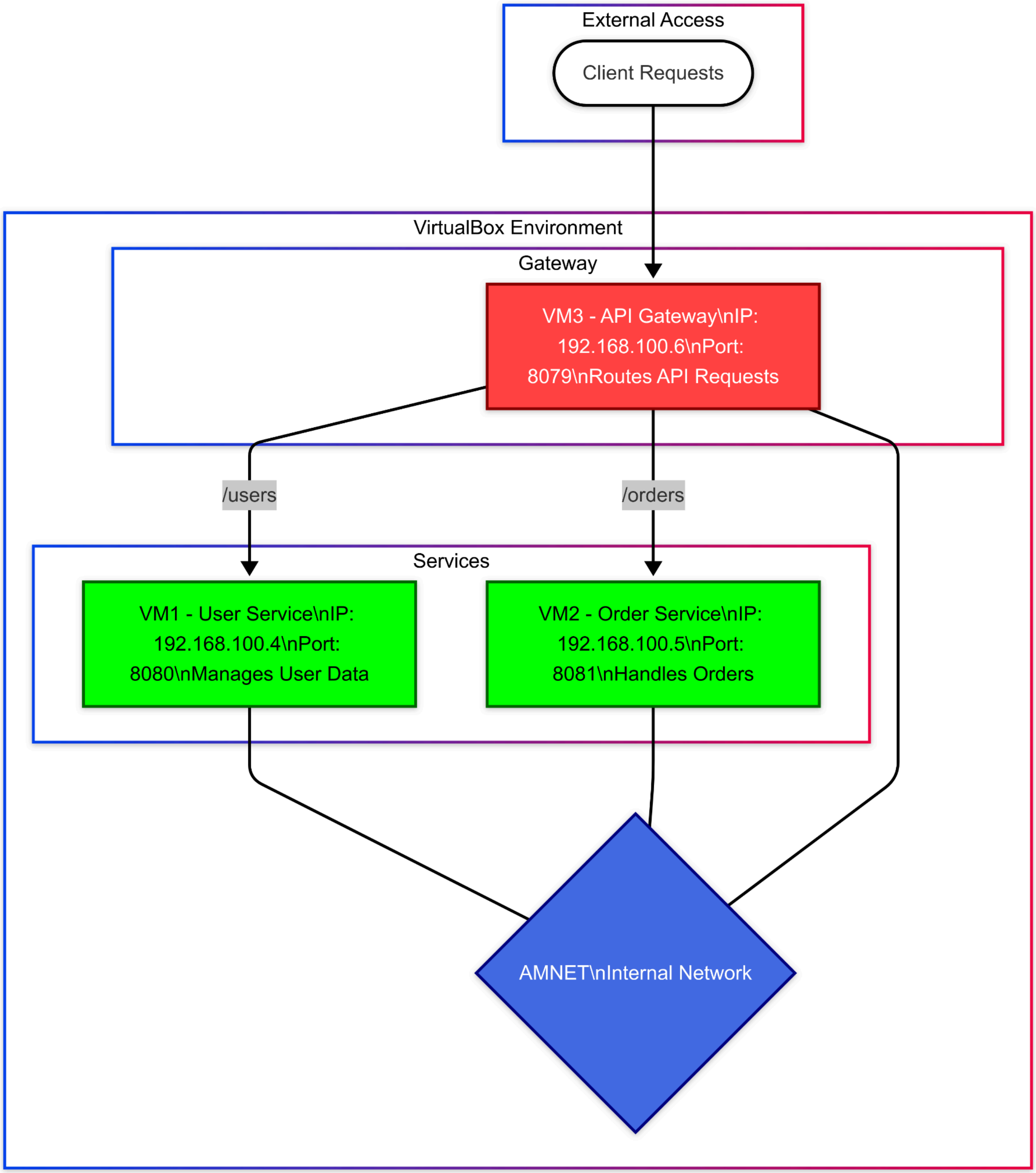
```
vm1@vm1:~$ curl -X GET http://192.168.100.6:8079/users
[{"id":1,"name":"Amresh"},{"id":2,"name":"Ayush"}]
vm1@vm1:~$ curl -X GET http://192.168.100.6:8079/orders
[{"id":101,"product":"Laptop","user_id":1},{"id":102,"product":"Phone","user_id":2}]
vm1@vm1:~$ _
```

6. Conclusion

In this report, we demonstrated how to:

- Set up VirtualBox and multiple Ubuntu VMs.
- Configure network settings for inter-VM communication.
- Deploy a microservice-based architecture with separate **User**, **Order**, and **Gateway** services.
- Validate API endpoints using curl
- Also added the architecture diagram

6. Architecture Diagram


External Access
Client Requests
VirtualBox Environment
Gateway
VM3 - API Gateway\nIP: 192.168.100.6\nPort: 8079\nRoutes API Requests
/users
/orders
Services
VM1 - User Service\nIP: 192.168.100.4\nPort: 8080\nManages User Data
VM2 - Order Service\nIP: 192.168.100.5\nPort: 8081\nHandles Orders
AMNET\nInternal Network