Assignment 1 — Virtual Machines and Microservice Deployment

Plagiarism Clause

I declare that this assignment report, implementation steps, configurations, and deployed code are my own work performed during the laboratory practice. Any form of plagiarism in implementation, configuration, code or documentation may lead to disqualification as per academic integrity policies.

PART 1 — Document Report

---

1. Objective

The objective of this assignment was to create multiple virtual machines using Oracle VirtualBox, establish network communication between them and deploy a simple microservice application that could be accessed from another VM within an isolated internal network.

---

2. Virtual Machine Setup

Step 1 — Installed VirtualBox

Oracle VirtualBox was installed on the host system to create and manage virtual machines.

Step 2 — Created Two Virtual Machines

VM	Role	Operating System	RAM	CPU	Disk
VM1	Server VM	Ubuntu Linux	2 GB	2 Cores	30 GB
VM2	Client VM	Ubuntu Linux	2 GB	2 Cores	30 GB

VM2 was created by **cloning VM1** to reduce setup time.

Step 3 — Clone Identity Fix (Important Step Performed)

After cloning, both machines had identical host identity, which caused network conflicts. This was resolved on **VM2**:

sudo hostnamectl set-hostname vm2-client

sudo nano /etc/hosts

Changed:

127.0.1.1 vm1-microservice

to

127.0.1.1 vm2-client

VM2 was then rebooted.

---

3. Network Configuration

Adapter 1 (NAT) was used for internet access to install packages such as Apache, while Adapter 2 (Internal Network) was used for VM-to-VM communication.

VirtualBox Adapter Setup

Adapter	**Setting**
Adapter 1	NAT (for internet access)
Adapter 2	Internal Network → **intnet**

This created a **private network** between the two VMs.

Initial IP Assignment (Temporary Method)

Temporary IPs were assigned using:

VM1

sudo ip addr add 192.168.100.10/24 dev enp0s8

sudo ip link set enp0s8 up

VM2

sudo ip addr add 192.168.100.20/24 dev enp0s8

sudo ip link set enp0s8 up

Issue Faced

After reboot, IP addresses disappeared. This caused Apache and services to stop working. The reason was that the above method assigns **temporary IP addresses**.

---

4. Permanent IP Configuration (Final Working Method)

To solve the issue, Netplan configuration was used.

VM1

network:

 version: 2

 renderer: NetworkManager

 ethernets:

  enp0s8:

dhcp4: no

addresses:

- 192.168.100.10/24

Permissions were corrected using:
sudo chmod 600 /etc/netplan/01-network-manager-all.yaml

sudo netplan apply

VM2

network:

version: 2

renderer: NetworkManager

ethernets:

enp0s8:

dhcp4: no

addresses:

- 192.168.100.20/24

Verification

ping 192.168.100.10

Successful replies confirmed permanent connectivity.

---

5. Web Server Setup (Apache Test)

Apache was installed on VM1:

sudo apt update

sudo apt install apache2 -y

sudo systemctl restart apache2

sudo systemctl status apache2

Verification from VM2:

http://192.168.100.10

The Apache default page appeared, confirming server operation.

---

6. Microservice Deployment

A lightweight Python microservice was deployed using Python's built-in HTTP server.

HTML File Created

nano index.html

<h1>Python Microservice Running</h1>

Server: VM1-Microservice

Status: Active

Port: 5000

Start Microservice

python3 -m http.server 5000

This started an HTTP service on port **5000**.

The server console displayed Serving HTTP on 0.0.0.0 port 5000, confirming the service started successfully.

---

7. Microservice Verification

From VM2 browser:

http://192.168.100.10:5000

The custom HTML page loaded successfully, confirming:

- Microservice running
- VM-to-VM communication working
- Successful service hosting

---

8. Issues Faced and Solutions

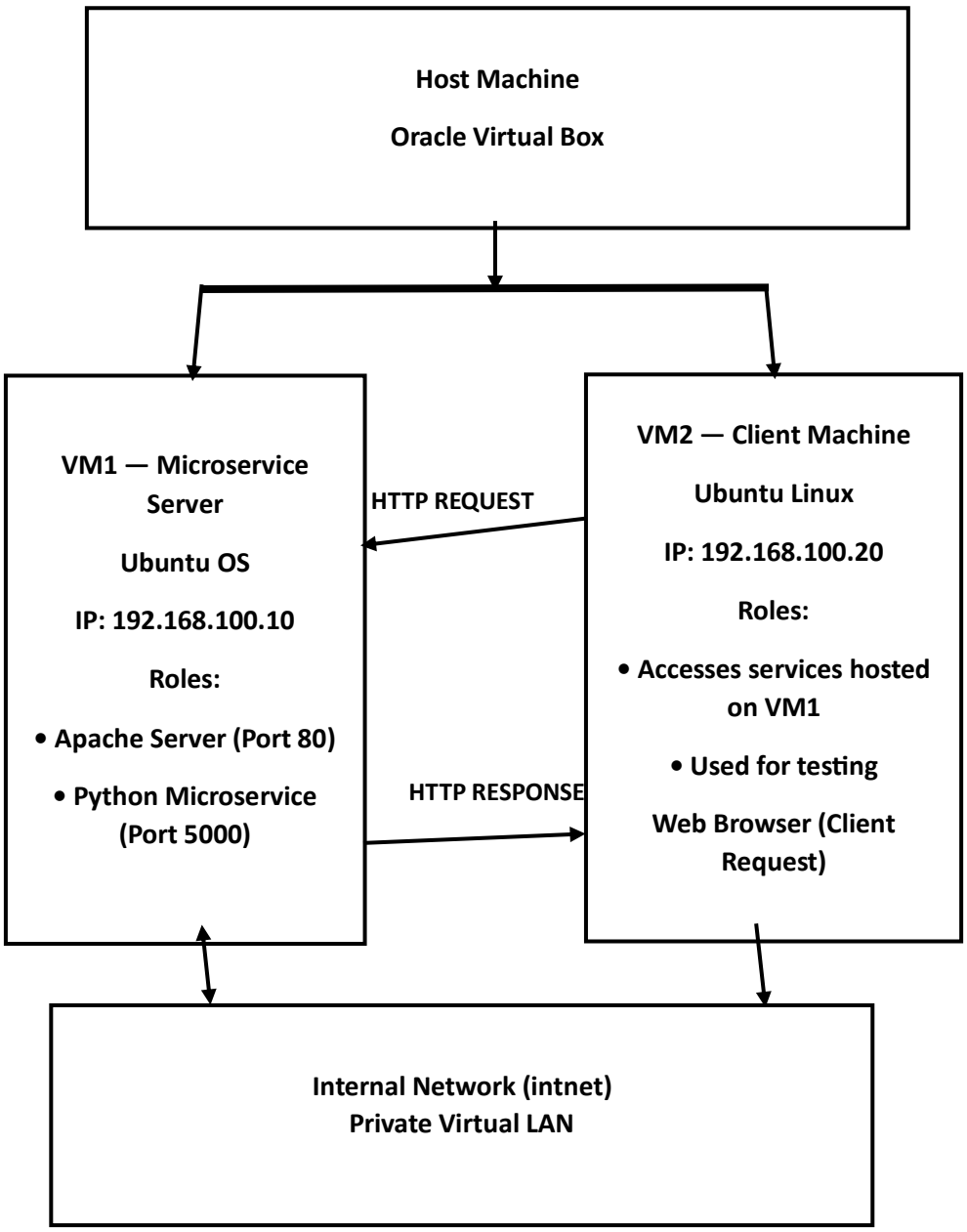
Issue	Cause	Solution
Ping not working	Clone hostname conflict	Changed hostname & hosts file
Apache not loading	IP was temporary	Configured Netplan
Netplan warnings	Wrong file permission	Used chmod 600
Microservice stopped	System reboot	Restarted Python server
IP lost after reboot	Temporary IP configuration	Configured Netplan for permanent IP

PART 2 — Architecture Description

The system architecture consists of:

- Host Machine running Oracle VirtualBox
- VM1 (Server) hosting Apache Web Server (Port 80) and a Python microservice (Port 5000)
- VM2 (Client) used to access the services
- Both VMs connected through an internal network (intnet)

VM2 sends an HTTP request through the internal virtual network to VM1, where **Apache or the Python microservice** processes the request and sends back an HTTP response.



This simulates a **private cloud environment** where services run on one node and are accessed remotely by another.

PART 3 — Source Code Repository

The complete implementation files, microservice code, and configuration details are available in the following GitHub repository:

GitHub Repository Link:
https://github.com/arpit-systems/mini-cloud-microservice

This repository contains:

• Assignment report
• Microservice HTML file
• Deployment demonstration files

The repository serves as proof of implementation and version control for the deployed microservice system.

PART 4 — Recorded Video Demonstration

A complete working demonstration of the project implementation, virtual machine setup, network configuration, and microservice deployment is provided in the following video:

YouTube Video Demo Link:
https://www.youtube.com/watch?v=kfJDeKl-_SY

The video shows:
• Creation and configuration of Virtual Machines
• Internal network communication between VMs
• Deployment of Apache Web Server on VM1
• Deployment of Python-based microservice on port 5000
• Successful access of services from the client VM
• Explanation of system architecture

This video serves as visual proof of successful system implementation and functioning.

---

9. Conclusion

This assignment provided practical exposure to:

- Virtual machine creation and management
- Internal networking between multiple systems
- Static IP configuration for stable communication
- Web server deployment using Apache
- Microservice hosting and remote access

The implementation simulated a basic cloud-like environment where services were hosted on one virtual node and accessed remotely from another. This demonstrated hands-on understanding of distributed systems, service deployment models and network-based resource sharing core principles of modern cloud computing and microservice architectures.

Both the Apache web server and the Python-based microservice were successfully deployed on the server VM and accessed from the client VM, confirming correct network configuration and service availability. All assignment objectives and technical requirements were achieved successfully