Microservices Deployment on VirtualBox VMs

Aryan Kumar M23CSA510

February 11, 2025

GitHub Repository: `https://github.com/m23csa510/virtulization_and_cloud_computing`

Abstract

This report outlines the step-by-step process of deploying a microservice-based application across three Ubuntu virtual machines (VMs) using Oracle VirtualBox. The architecture design and network configuration ensure seamless communication between the VMs, while virtual environments facilitate code deployment. The microservices are implemented using FastAPI, a modern Python framework, and tested for end-to-end communication.

Contents

1 Introduction

Microservices architecture enables the development and deployment of independent services that work together to form a larger application. This report demonstrates how to set up such an architecture using Ubuntu VMs and VirtualBox.

2 Installation of VirtualBox in Windows

2.1 Step 1: Download VirtualBox

Visit the official VirtualBox website and download the latest Windows installer.

2.2 Step 2: Install VirtualBox

Run the downloaded executable and follow the setup wizard. Ensure to install the Oracle VM VirtualBox Extension Pack for additional features.

3 Creating Virtual Machines with Ubuntu

3.1 Step 1: Launch VirtualBox

Open the VirtualBox application and click on "New" to create a new VM.

3.2 Step 2: Configure VM Settings

- **Name**: Assign a descriptive name (e.g., UserVM, ProductVM, OrderVM).
- **Type**: Select "Linux" and choose "Ubuntu (64-bit)" as the version.
- **Memory**: Allocate 2048 MB (2GB) RAM.
- **Hard Disk**: Create a new virtual hard disk with a size of 25 GB.
- **Network Adapters**:
  - **Adapter 1**: NAT (for internet access).
  - **Adapter 2**: Host-only Network (for inter-VM communication).

3.3 Step 3: Install Ubuntu

1. Attach the Ubuntu ISO file to the VM's optical drive.
2. Start the VM and follow the Ubuntu installation wizard.
3. During installation, choose "Minimal installation" and set the username and password.
4. Perform an **unattended installation** using preseed configurations.

4 Network Configuration

4.1 Step 1: Set Up Host-only Network

1. In VirtualBox, go to "File ¿ Host Network Manager" and create a new host-only network.
2. Enable the DHCP server with a starting IP address of 192.168.56.1.

4.2 Step 2: Get Static IPs

```
ip a
```

Output - VM1: 192.168.56.105 - VM2: 192.168.56.101 - VM3: 192.168.56.104

4.3 Step 3: Test Network Connectivity

From each VM, ping the other VMs to ensure connectivity:

```
ping 192.168.56.101
ping 192.168.56.105
ping 192.168.56.104
```

5 Microservice Deployment

5.1 Step 1: Create Project Structure

In each VM, create a project directory:

```
mkdir ~/microservices
cd ~/microservices
```

Create a virtual environment:

```
python3 -m venv venv
source venv/bin/activate
```

5.2 Step 2: Install FastAPI

```
pip install fastapi uvicorn
```

5.3 Step 3: Deploy Services

User Service (user_service.py):

```
from fastapi import FastAPI
app = FastAPI()

@app.get("/users")
def users():
    return [{"id": 1, "name": "Alice"}, {"id": 2, "name": "Bob"}]

if __name__ == "__main__":
    import uvicorn
    uvicorn.run("user_service:app", host="0.0.0.0", port=8000)
```

Product Service (product_service.py):

```
from fastapi import FastAPI
app = FastAPI()

@app.get("/products")
def products():
    return [{"id": 1, "name": "Book"}, {"id": 2, "name": "Laptop"}]
```

Order Service (order_service.py):

```
from fastapi import FastAPI, HTTPException
import requests
app = FastAPI()

@app.get("/orders")
def orders():
    try:
        users = requests.get("http://192.168.56.101:8000/users").json()
        products = requests.get("http://192.168.56.102:8000/products").json()
```

```
        return {"orders": [{"user_id": 1, "product_id": 1}], "users": users, "products":
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

5.4 Step 4: Start Services

Activate the virtual environment and run:

```
source venv/bin/activate
python user_service.py  % VM1
python product_service.py  % VM2
python order_service.py &  % VM3 (background)
```

6 Testing Microservice Communication

Test the communication between the microservices:

1. From the OrderVM, use curl to request data from UserVM and ProductVM:

   ```
   curl http://192.168.56.101:8000/users
   curl http://192.168.56.102:8000/products
   ```

2. From the host, use port forwarding to access the OrderVM:

   ```
   curl http://localhost:8000/orders
   ```

7 Architecture Diagram

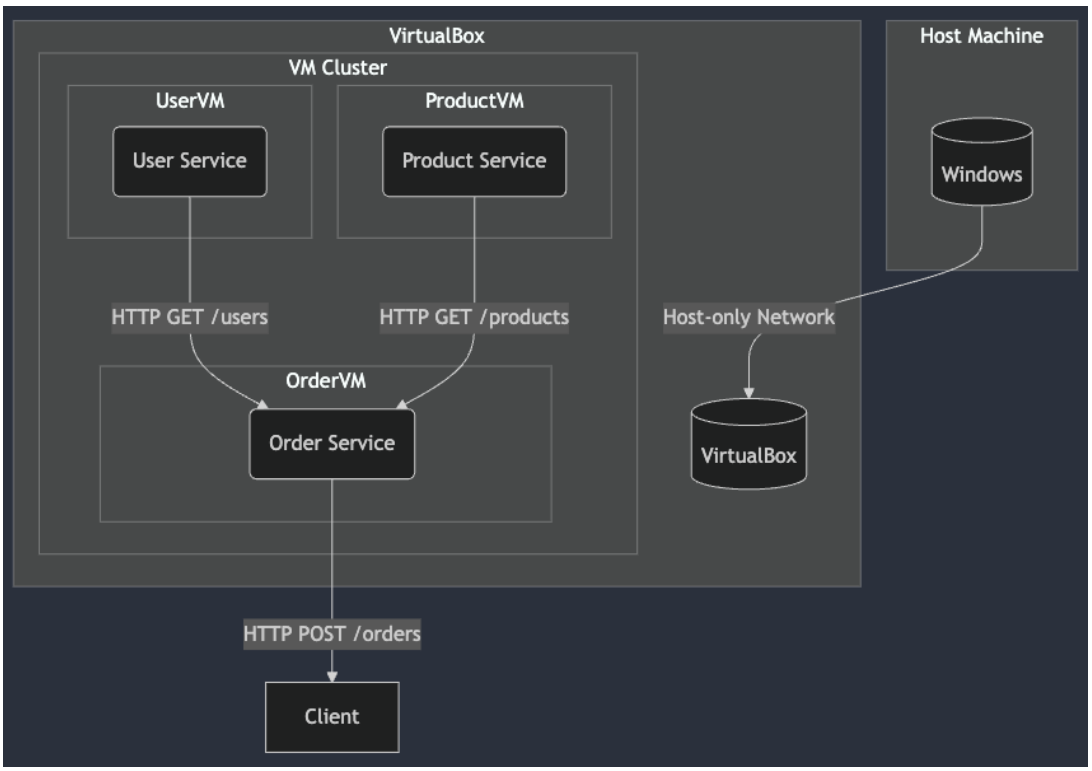

Figure 1: **Overall Architecture**

8 Conclusion

The deployment of microservices across three VMs demonstrates the principles of distributed systems and inter-VM communication. This architecture can be expanded by adding more VMs and implementing load balancing for scalability. The use of virtual environments ensures isolated deployment environments for each service.