Document Report: Virtual Machine-Based Microservice Deployment

Student Information

Name: SUVODIP SOM

Roll No: M23CSA533

Subject: Virtualization and Cloud Computing

Project Resources

GitHub Repository: https://github.com/suvosom1997/VCC_project_M23CSA533 (https://github.com/suvosom1997/VCC_project_M23CSA533)

Project Demo Video: https://www.youtube.com/watch?v=flOZtjLqVSc (https://www.youtube.com/watch?v=flOZtjLqVSc)

Implementation Guide for Virtual Machine Microservices

1. VirtualBox Installation and VM Setup

Initial Setup Process

Begin by installing Oracle VirtualBox:

- Download the software from the official VirtualBox website: https://www.virtualbox.org/ (https://www.virtualbox.org/)
- Complete the installation by following the provided wizard

Virtual Machine Creation

Create two separate virtual machines with these specifications:

For VM1 and VM2:

- Name: VM1/VM2 respectively
- Operating System: Linux
- Memory Allocation: Minimum 1GB (2GB recommended)
- Storage: Minimum 10GB virtual hard disk
- Boot Media: Puppy Linux ISO

---

2. Network Configuration

To establish VM communication, configure the network settings as follows:

NAT Network Setup

1. Navigate to File > Preferences > Network in VirtualBox
2. Access the NAT Networks tab
3. Create new network (Name suggestion: `NatNetwork`)

Individual VM Network Configuration

For each virtual machine:

1. Access Settings > Network
2. Configure Adapter 1:
    - Set to NAT Network
    - Select `NatNetwork` from options
3. Configure Adapter 2:
    - Enable the adapter
    - Set to Host-Only Adapter

Connection Verification

Test network connectivity:

```
# On VM1 - Find IP address
ip a

# On VM2 - Test connection
ping <VM1-IP-Address>
```

---

3. Microservice Application Deployment

Server Setup (VM1)

1. Create server directory:

```
mkdir -p ~/Downloads/server
cd ~/Downloads/server
```

2. Install required packages:

```
sudo apt-get update
sudo apt-get install nodejs npm nano -y
```

3. Create server application:

```
nano server.js
```

4. Server code implementation:

```
const express = require('express');
const service = express();
const PORT = 4001;

service.get('/status', (req, res) => {
    res.send('Hello from VM1');
});

service.listen(PORT, () => {
    console.log(`VM1 service listening on port ${PORT}`);
});
```

5. Install Express and start server:

```
npm install express
node server.js
```

Client Setup (VM2)

1. Create client directory:

```
mkdir -p ~/Downloads/client
cd ~/Downloads/client
```

2. Install required packages:

```
sudo apt-get update
sudo apt-get install nodejs npm nano -y
```

3. Create client application:

```
nano client.js
```

4. Client code implementation (with Axios):

```
const axios = require('axios');
const URL = 'http://192.168.100.8:4001/status';

axios.get(URL)
    .then(res => {
        console.log('Response from VM1:', res.data);
    })
    .catch(error => {
        console.error('Error occurred:', error);
    });
```

5. Install dependencies and run:

```
npm install axios
node client.js
```

---

4. System Testing & Troubleshooting

Network Connectivity Tests

- Run ping test between VMs:

```
ping <VM1-IP>
```

Server Verification

- Check server process:

```
ps aux | grep node
```

- Restart if needed:

```
node server.js
```

Port Accessibility

- Verify port 4001 status:

```
sudo netstat -tulnp | grep 4001
```

- Configure firewall if needed:

```
sudo iptables -A INPUT -p tcp --dport 4001 -j ACCEPT
```

Client Debugging

- Verify Node.js installation:

```
node -v
```

- Check URL and port configuration in client.js

Project Conclusion

This implementation successfully demonstrates:

- Basic microservice architecture deployment
- Node.js application distribution across VMs
- Inter-VM communication via NAT Network
- Server-client interaction on Puppy Linux VMs

Future Enhancement Possibilities

- Multiple microservice integration
- Database system implementation (MongoDB, MySQL)
- Migration to Docker containers

This project provides fundamental knowledge in:

- VM networking
- Microservice deployment
- Virtualized environment management

End of Report