LAB 1 Olumuyiwa Ojo - 0232763507

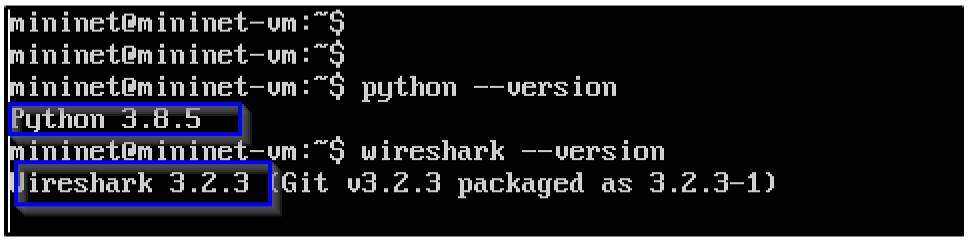
This report explicitly describes every step taken to set up an SDN environment in order to create a basic topology. It took a while to complete because of the steep learning curve and also trying to solve every error encountered.

As Included in the section on the tools required, the following tools were used to complete the exercise;

- Mininet VM
- Windows 11 (Host environment)
- Virtualbox Manager
- Docker
- OpenFlow (for communication between switches and controllers) – included in MininetVM
- Open Network Operating System [ONOS] (as the SDN controller)

Step 1

Download and Install Mininet VM. The VM is provisioned with mininet as the username and password. The VM also has wireshark and python pre-installed as shown in Figure 1.



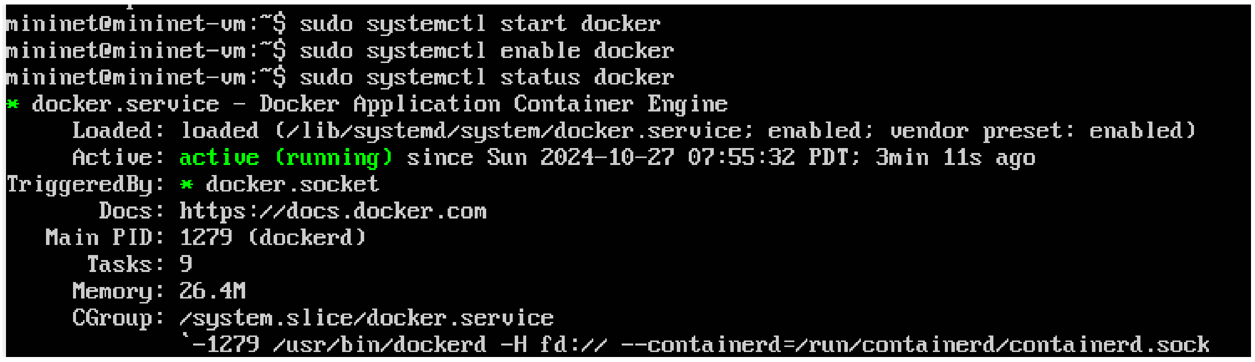
```
mininet@mininet-vm:~$
mininet@mininet-vm:~$
mininet@mininet-vm:~$ python --version
Python 3.8.5
mininet@mininet-vm:~$ wireshark --version
Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1)
```

Figure 1.

Step 2

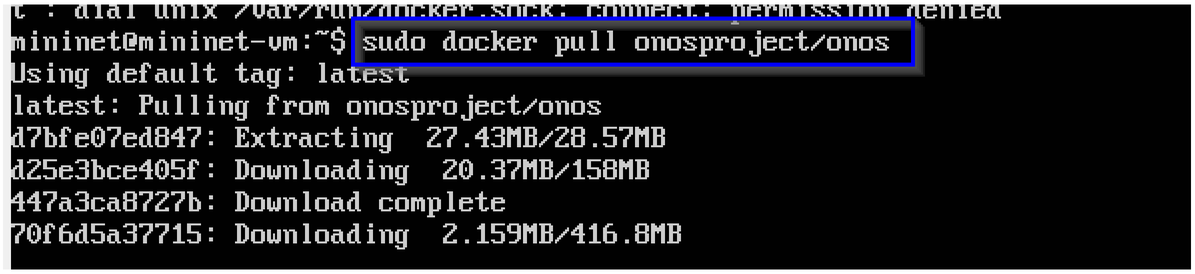
Install Docker and also the ONOS Docker Image.

- sudo apt install docker.io



```
mininet@mininet-vm:~$ sudo systemctl start docker
mininet@mininet-vm:~$ sudo systemctl enable docker
mininet@mininet-vm:~$ sudo systemctl status docker
* docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-10-27 07:55:32 PDT; 3min 11s ago
TriggeredBy: * docker.socket
       Docs: https://docs.docker.com
   Main PID: 1279 (dockerd)
      Tasks: 9
     Memory: 26.4M
     CGroup: /system.slice/docker.service
             `-1279 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Figure 2. Activate & Check the Docker Status

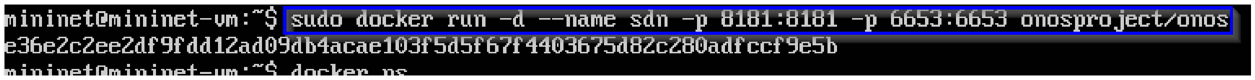


```
t : dial unix /var/run/docker.sock: connect: permission denied
mininet@mininet-vm:~$ sudo docker pull onosproject/onos
Using default tag: latest
latest: Pulling from onosproject/onos
d7bfe07ed847: Extracting  27.43MB/28.57MB
d25e3bce405f: Downloading  20.37MB/158MB
447a3ca8727b: Download complete
70f6d5a37715: Downloading  2.159MB/416.8MB
```

Figure 3. Installing ONOS Docker Image

Step 3

Run the ONOS Docker image as shown in Figure 4.



```
mininet@mininet-vm:~$ sudo docker run -d --name sdn -p 8181:8181 -p 6653:6653 onosproject/onos
e36e2c2ee2df9fdd12ad09db4acae103f5d5f67f4403675d82c280adfccf9e5b
mininet@mininet-vm:~$ docker ps
```

Figure 4.

`-d`: runs the container in the background (detached mode).

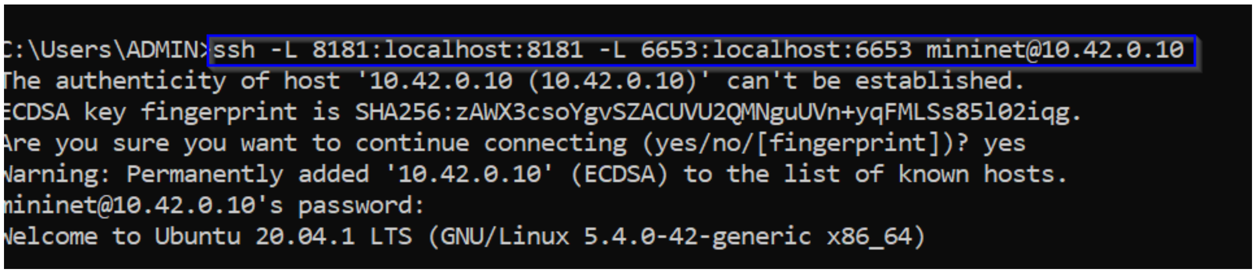
`--name sdn`: names the container `sdn`.

`-p 8181:8181`: maps port `8181` of the container (ONOS GUI) to `8181` on the host.

`-p 6653:6653`: maps port `6653` of the container (ONOS OpenFlow) to `6653` on the host

Step 4

Due to the terminal interface provided on the Mininet VM and the ability to interact with ONOS web UI, access to the VM was done through SSH with the command described in Figure 5.



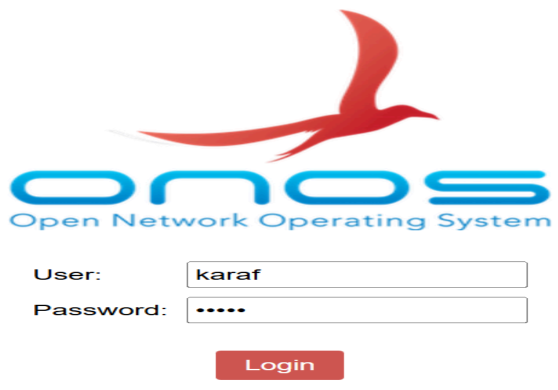
```
C:\Users\ADMIN>ssh -L 8181:localhost:8181 -L 6653:localhost:6653 mininet@10.42.0.10
The authenticity of host '10.42.0.10 (10.42.0.10)' can't be established.
ECDSA key fingerprint is SHA256:zAWX3csoYgvSZACUVU2QMNguUVn+yqFMLSs85l02iqg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.42.0.10' (ECDSA) to the list of known hosts.
mininet@10.42.0.10's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)
```

Figure 5. SSH Connection

-L 8181:localhost:8181: forwards local port 8181 to remote port 8181 on localhost (the remote machine-mininet) for accessing the ONOS GUI.
-L 6653:localhost:6653: forwards local port 6653 to remote port 6653 on localhost (the remote machine-mininet) for OpenFlow communication.
mininet@10.42.1.10: Specifies the remote server's username (mininet) and IP address (10.42.1.10) for the SSH connection.

Step 5

Verify access to the ONOS web UI on the host machine. Login credential is 'karaf' for both username and password.



.

Figure 6. Login Interface

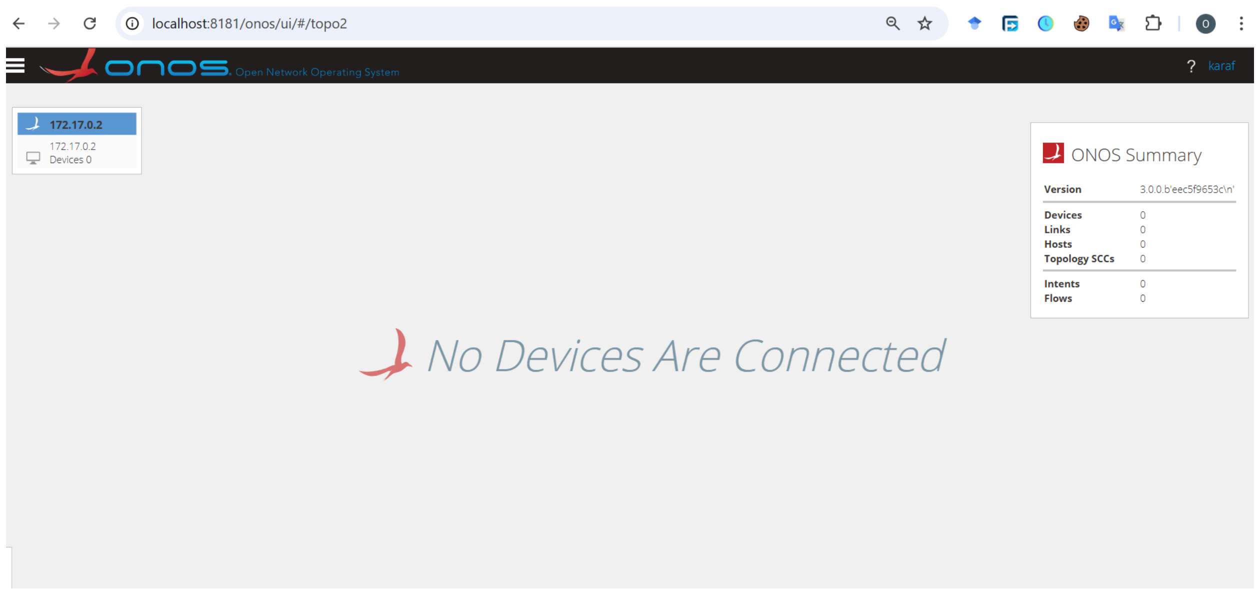


Figure 7. Access Granted

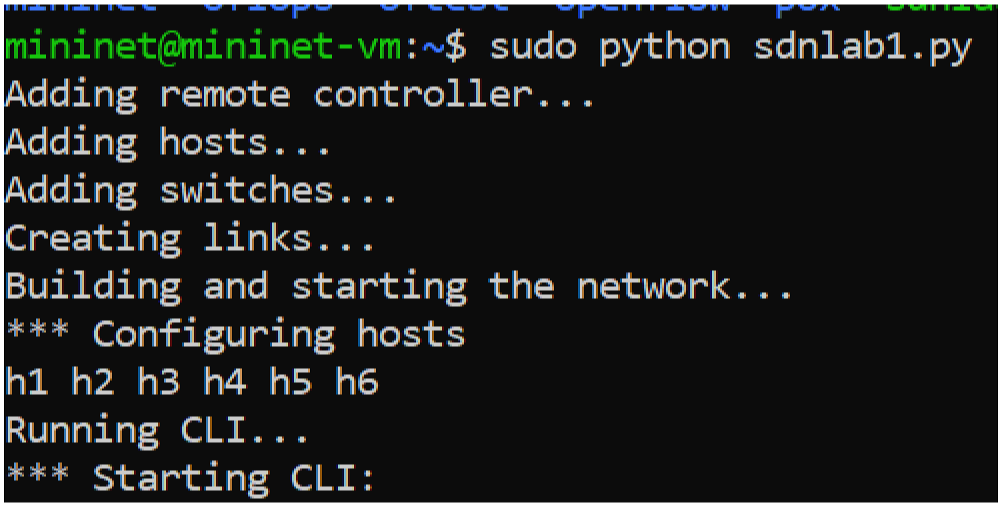
There are no devices because no network has been set up yet.

Step 6

A Python script was created to set up the network with the constraint defined in the task concerning the topology and IP assignments. Figure 8 describes the execution of the script, the device reachability is shown in Figure 9, and the GUI outcome is shown in Figure 10.

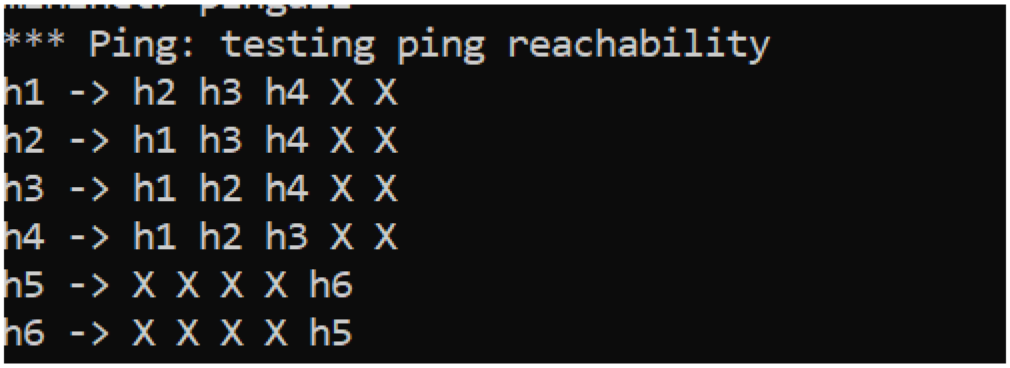
Six hosts (h1, h2, h3, h4, h5, h6)
Three SDN-enabled switches (s1, s2, s3)

Topology
-s1: This switch connects hosts h1 and h3 and has a link to s2.
-s2: This switch connects hosts h2 and h4, links to s1, and directly connects to the Controller.
-s3: This switch connects hosts h5 and h6 and directly connects to the Controller.
-h6 directly connects to the Controller.



```
mininet@mininet-vm:~$ sudo python sdnlab1.py
Adding remote controller...
Adding hosts...
Adding switches...
Creating links...
Building and starting the network...
*** Configuring hosts
h1 h2 h3 h4 h5 h6
Running CLI...
*** Starting CLI:
```

Figure 8. Python Script



```
*** Ping: testing ping reachability
h1 -> h2 h3 h4 X X
h2 -> h1 h3 h4 X X
h3 -> h1 h2 h4 X X
h4 -> h1 h2 h3 X X
h5 -> X X X X h6
h6 -> X X X X h5
```

Figure 9. Device Reachability

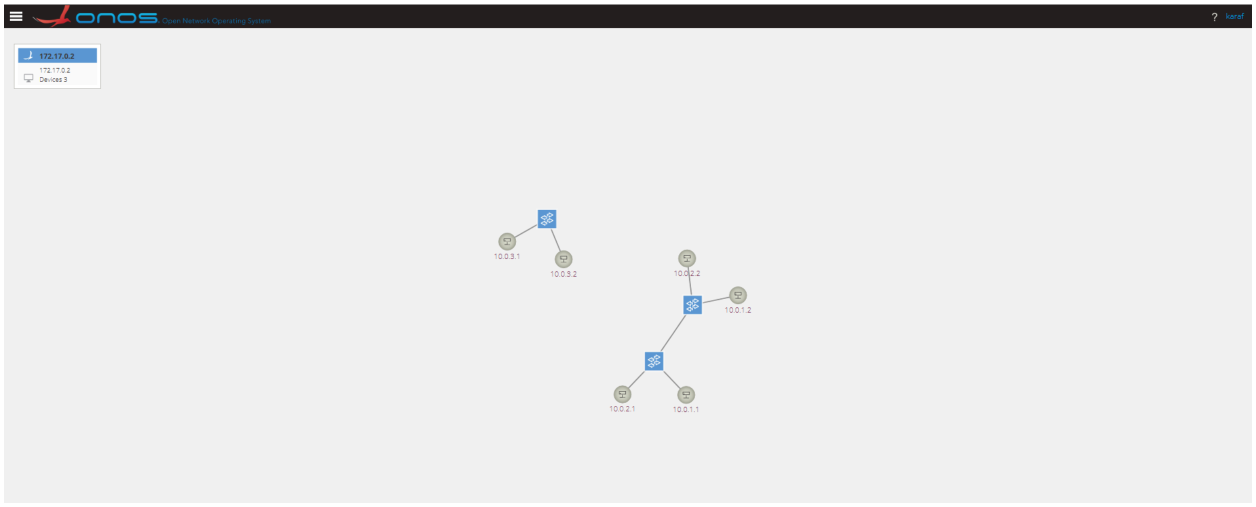
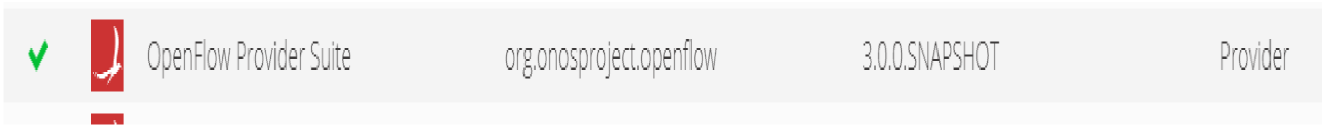


Figure 10. Outcome

Challenges Faced

After the execution of the Python file, the network did not show up in the GUI. After going through the documentation, it was discovered that the outcome is typical because ONOS doesn't yet know how to communicate with the switches using OpenFlow.

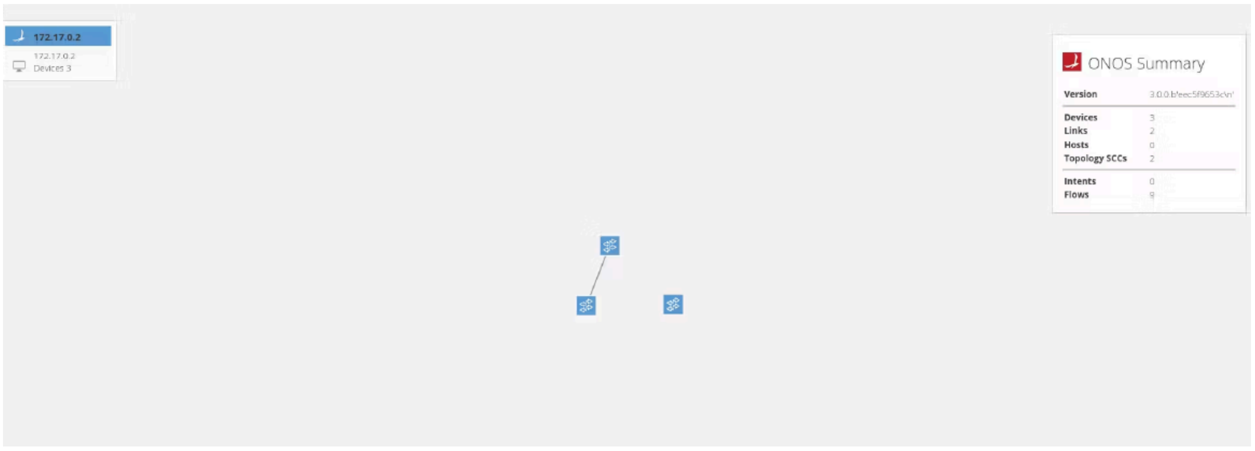
It was resolved by activating OpenFlow through the application tab.



✓	OpenFlow Provider Suite	org.onosproject.openflow	3.0.0.SNAPSHOT	Provider

Figure 11. OpenFlow Activation

In addition, after the activation of OpenFlow, the hosts were not discovered by ONOS (figure 12) because the different hosts had yet to send data between them. Running the pingall command shown in Figure 13 also failed. The reason is that even though the controller can talk to the switches through OpenFlow, the controller has no idea what to do with the packets the switch receives from the different hosts.

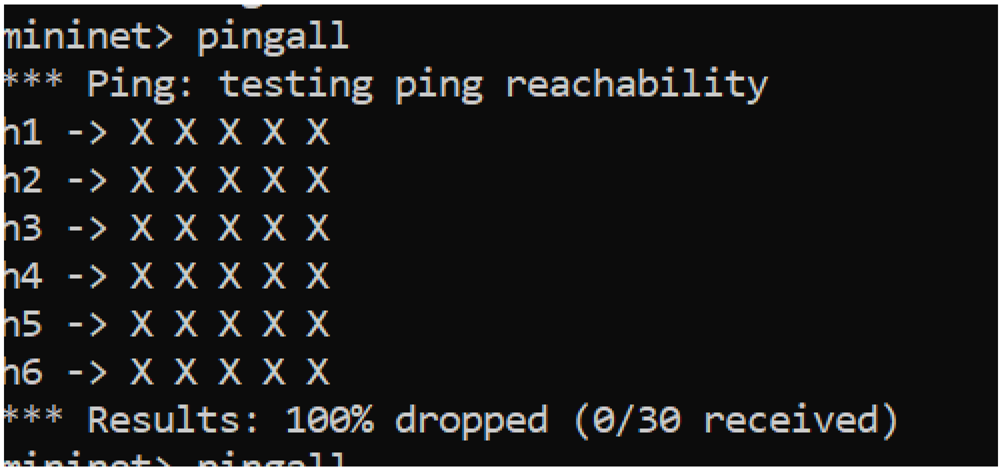


172.17.0.2
172.17.0.2
Devices 3

ONOS Summary

Version	3.0.0.b'eec5f9653c\n'
Devices	3
Links	2
Hosts	0
Topology SCCs	2
Intents	0
Flows	9

Figure 12. Hosts not discovered.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X
h2 -> X X X X X
h3 -> X X X X X
h4 -> X X X X X
h5 -> X X X X X
h6 -> X X X X X
*** Results: 100% dropped (0/30 received)
```

Figure 12.

It was solved by activating **org.onosproject.fwd** through the application tab.

Conclusion

This lab is a hands-on experience setting up an SDN environment using different tools, creating a basic topology and ensuring the connection works correctly.

The experience presented a steep learning curve, understanding the relationship between using different tools, being able to troubleshoot, and sourcing ways to solve issues encountered to complete the tasks.