ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46,Chennai-Bengaluru National Highways, Arappakkam, Ranipet- 632517,TamilNadu,India

Telephone:04172-292925 Fax:04172-292926

Email:amcet.rtet@gmail.com/info@amcet.inWeb:www.amcet.in

DEPARTMENT OF INFORMATION TECHNOLOGY

CCS365 – SOFTWARE DEFINED NETWORKS LABORATORY

NAME	**: ………………………………………………….**
REGISTER NUMBER	**: ………………………………………………….**
YEAR & BRANCH	**: ………………………………………………….**
SEMESTER	**: ………………………………………………….**
ACADEMIC YEAR	**: ………………………………………………….**

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46,Chennai-Bengaluru National Highways, Arappakkam,Ranipet-632517,TamilNadu, India

Telephone: 04172-292925 Fax: 04172-292926

CERTIFICATE

This is to Certify that the Bonafide record of the practical work done by ……………………………….……………...Register Number………………………………. of III year B.Tech **INFORMATION TECHNOLOGY** submitted for the B.E-Degree practical examination (VI Semester) in **CCS365 – SOFTWARE DEFINED NETWORKS LABORATORY** during the academic year **2025 –2026.**

STAFF IN–CHARGE **HEAD OF THE DEPARTMENT**

Submitted for the practical examination held on __________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENT

EX: NO: 1 **Setup your own virtual SDN lab**

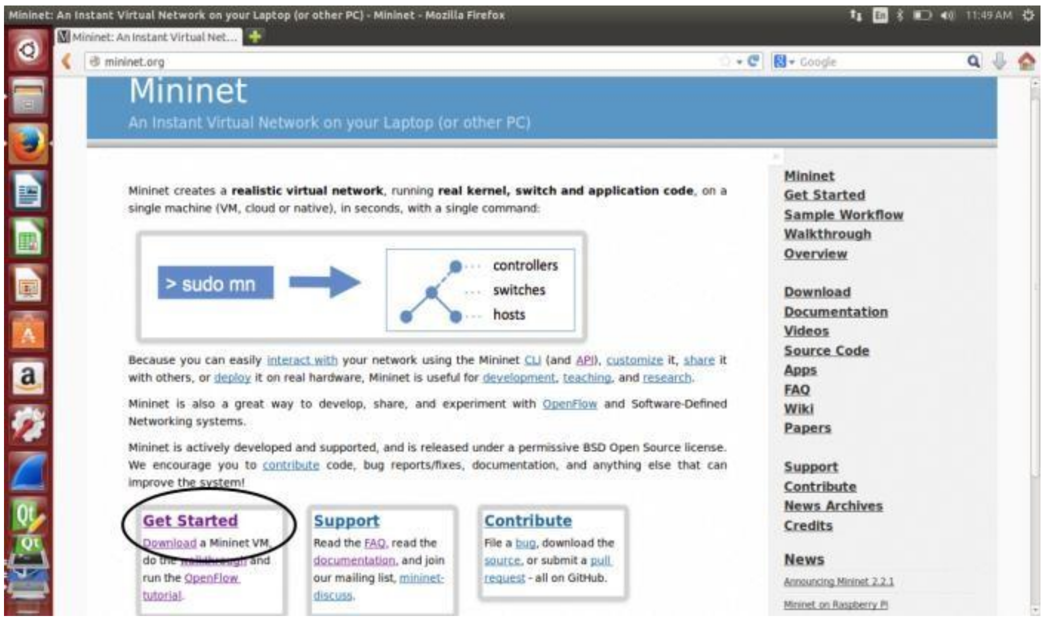
DATE: **Virtual Box /Mininet Environment for SDN**

AIM:

To install virtual Box/Mininet environment for setup your own virtual Software definition network.

PROCEDURE:

Step 1 : Download Mininet from http:mininet.org and start installation.



Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet - Mozilla Firefox

mininet.org

Mininet

An Instant Virtual Network on your Laptop (or other PC)

Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native), in seconds, with a single command:

Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems.

Mininet is actively developed and supported, and is released under a permissive BSD Open Source license. We encourage you to contribute code, bug reports/fixes, documentation, and anything else that can improve the system!

Get Started
Download a Mininet VM, do the walkthrough and run the OpenFlow tutorial.

Support
Read the FAQ, read the documentation, and join our mailing list, mininet-discuss.

Contribute
File a bug, download the source, or submit a pull request - all on GitHub.

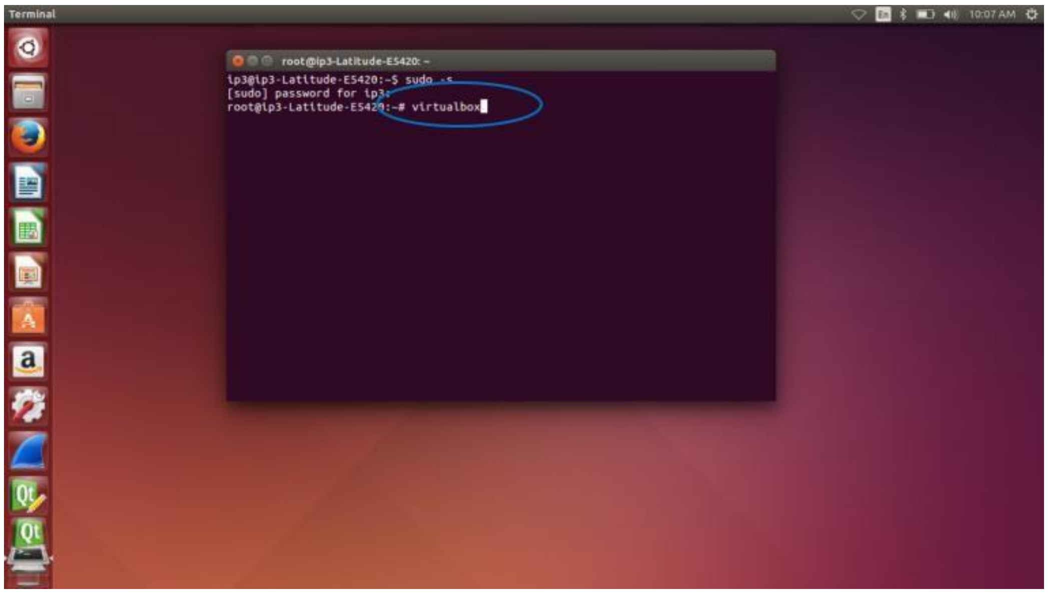
Mininet
Get Started
Sample Workflow
Walkthrough
Overview

Download
Documentation
Videos
Source Code
Apps
FAQ
Wiki
Papers

Support
Contribute
News Archives
Credits

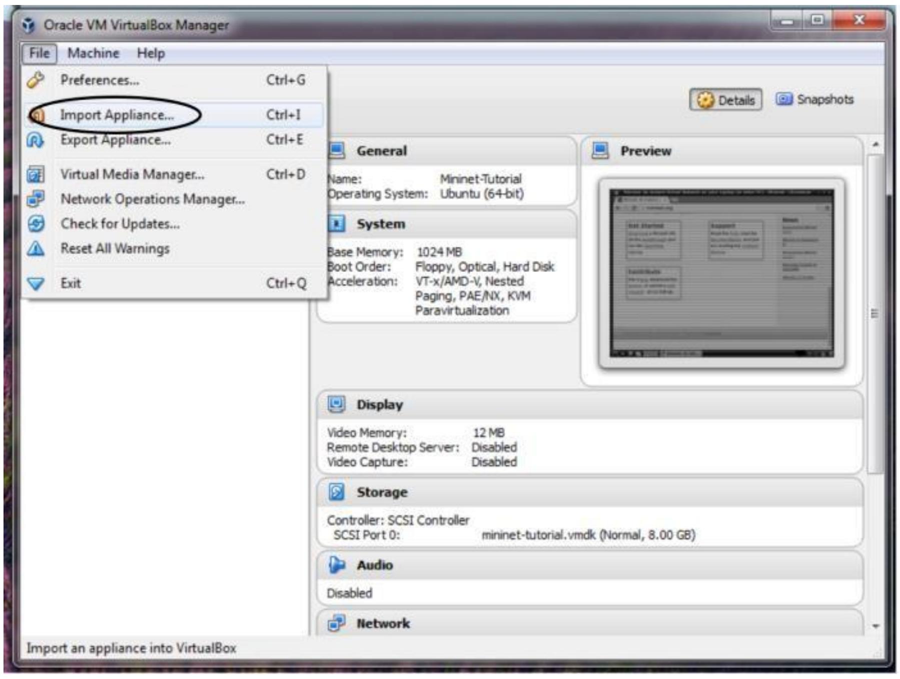
News
Announcing Mininet 2.2.1
Mininet on Raspberry Pi

After installed mininet and Download VirtualBox click here



```
root@ip3-Latitude-E5420: ~
ip3@ip3-Latitude-E5420:~$ sudo -s
[sudo] password for ip3:
root@ip3-Latitude-E5420:~# virtualbox
```

Step 2 : Then open oracle VM virtual box and choose File –> Import Appliance.



Name: Mininet-Tutorial
Operating System: Ubuntu (64-bit)

Base Memory: 1024 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

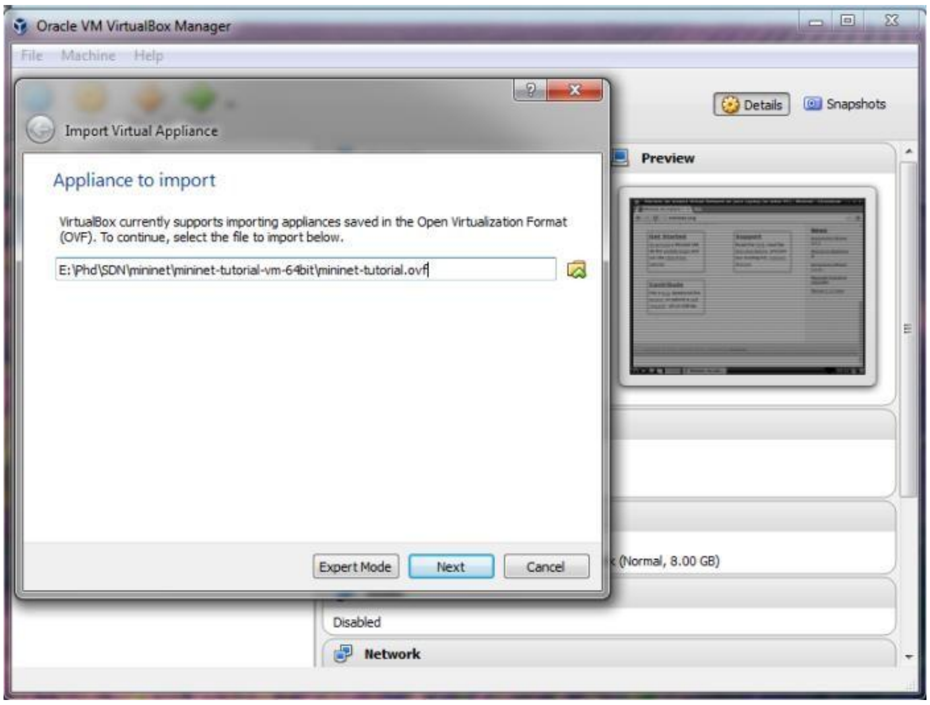
Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: SCSI Controller
SCSI Port 0: mininet-tutorial.vmdk (Normal, 8.00 GB)

Disabled

Import an appliance into VirtualBox

Step 3: Browse to open by clicking



Appliance to import

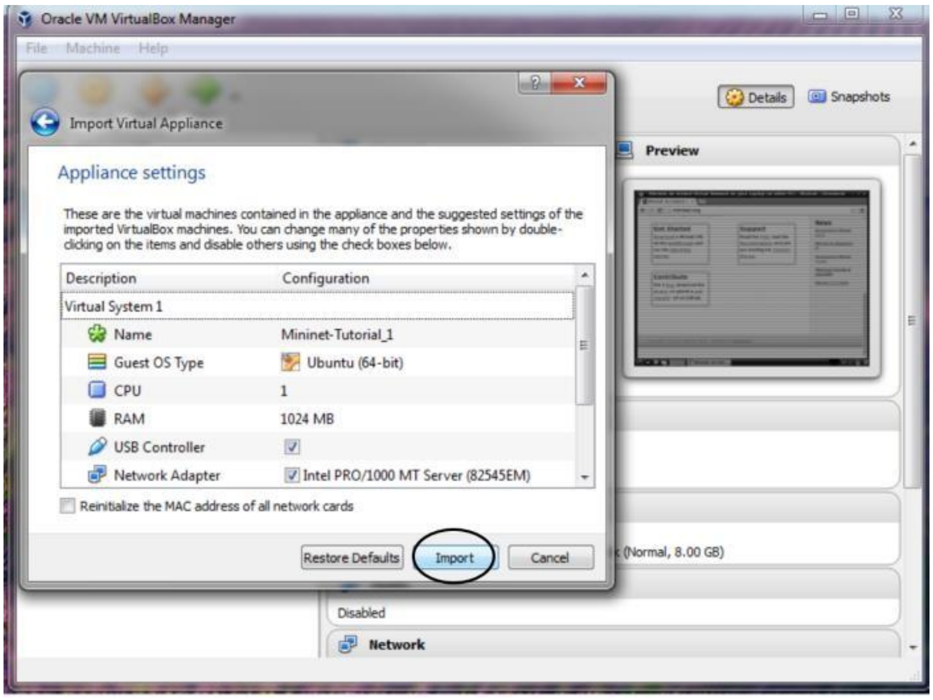
VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

E:\Phd\SDN\mininet\mininet-tutorial-vm-64bit\mininet-tutorial.ovf

(Normal, 8.00 GB)

Disabled

Step 4 : Click Next and Type Name as Mininet , select guest OS type as ubuntu 64bit or windows 2016 64bit ,CPU as 1 and RAM as 1024MB.

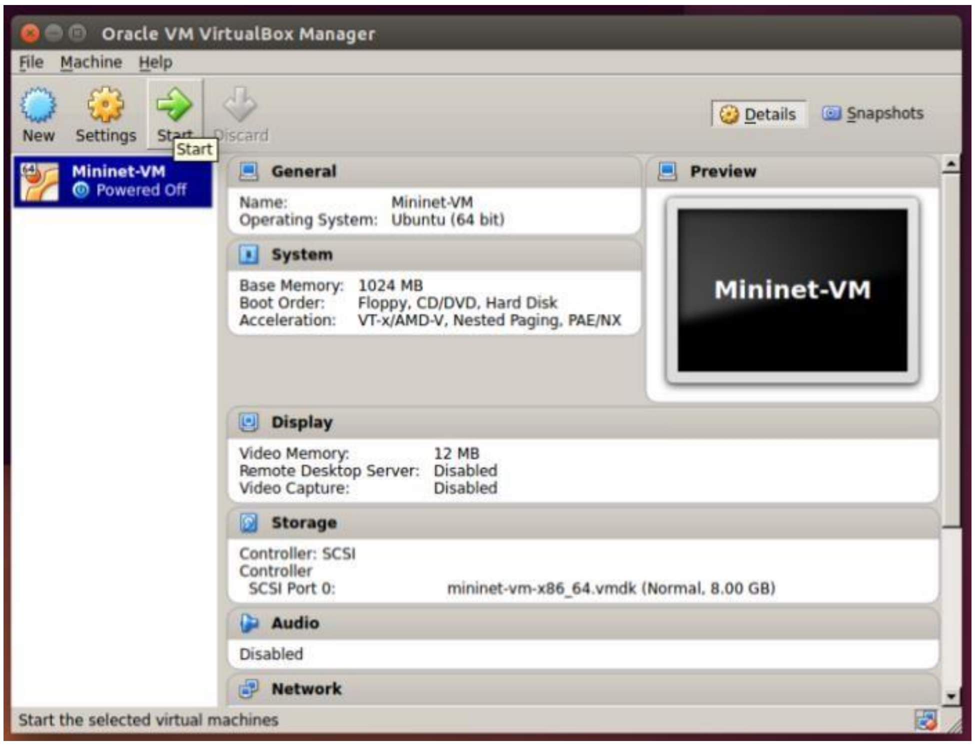

Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Description	Configuration
Virtual System 1	
Name	Mininet-Tutorial_1
Guest OS Type	Ubuntu (64-bit)
CPU	1
RAM	1024 MB
USB Controller	☑
Network Adapter	☑ Intel PRO/1000 MT Server (82545EM)

☐ Reinitialize the MAC address of all network cards

Step 5 : Click Import



General
Name: Mininet-VM
Operating System: Ubuntu (64 bit)

System
Base Memory: 1024 MB
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

Display
Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

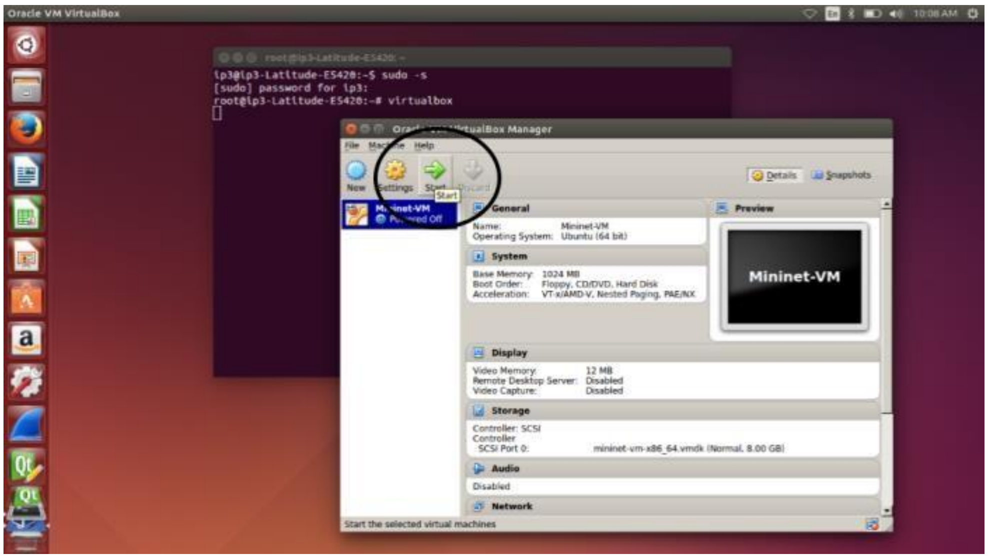
Storage
Controller: SCSI
Controller
SCSI Port 0: mininet-vm-x86_64.vmdk (Normal, 8.00 GB)

Audio
Disabled

Network

Start the selected virtual machines

Step 6 : To Start Mininet. Click Start button to start a mininet…



```
lp3@lp3-Latitude-E5420:~$ sudo -s
[sudo] password for lp3:
root@lp3-Latitude-E5420:~# virtualbox
```

Name: Mininet-VM
Operating System: Ubuntu (64 bit)

Base Memory: 1024 MB
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX

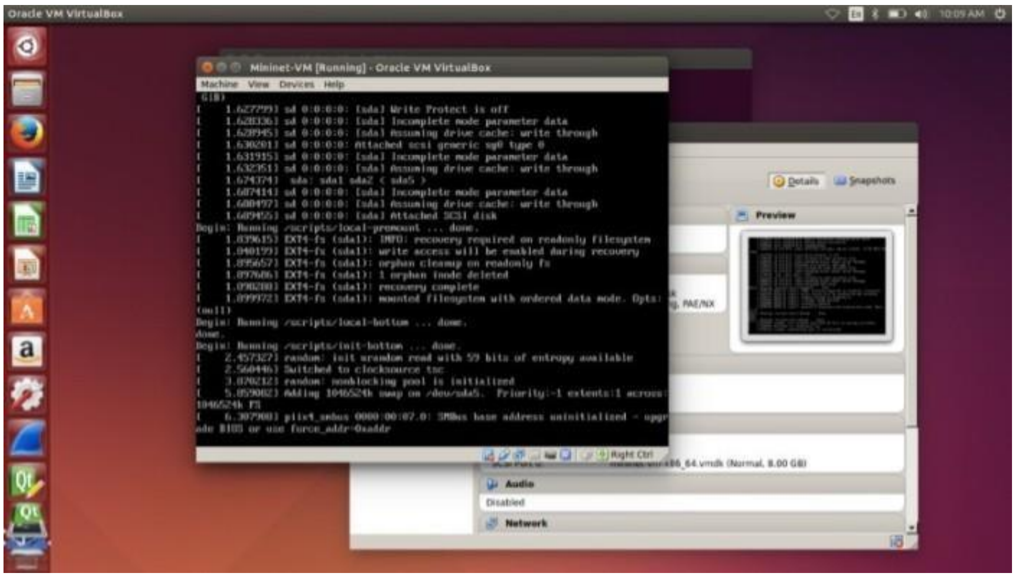
Video Memory: 12 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Controller: SCSI
Controller
SCSI Port 0: mininet-vm-x86_64.vmdk (Normal, 8.00 GB)

Disabled

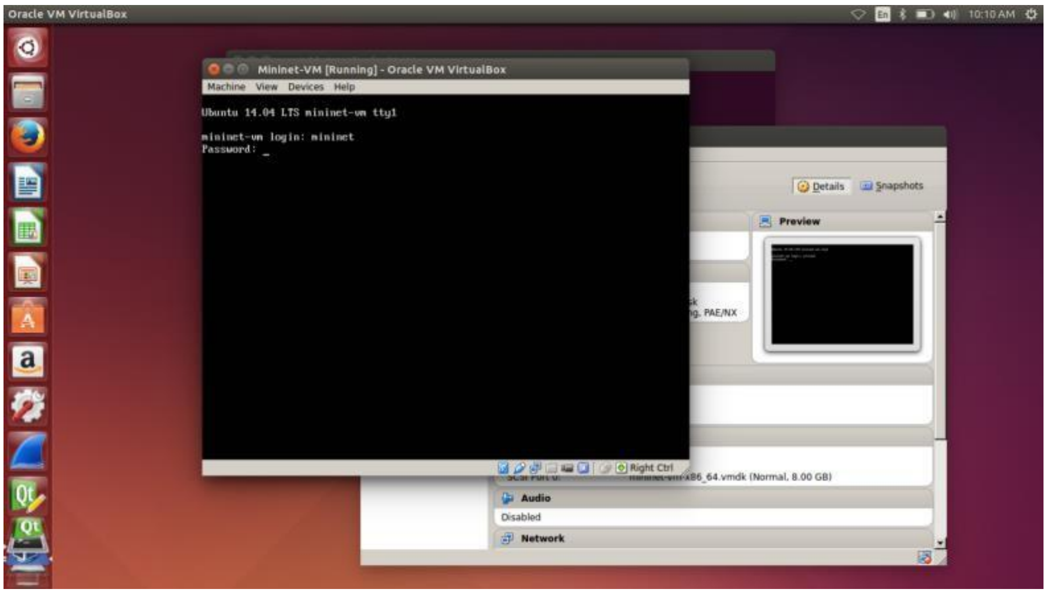
Start the selected virtual machines

Which takes some time to start to compile and updates of mininet in VM…



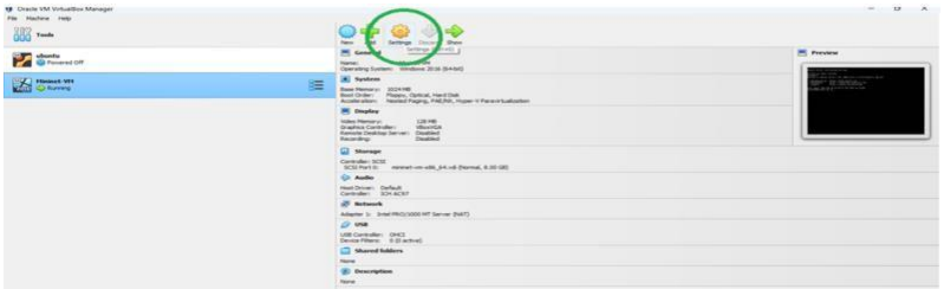
```
GiB)
[    1.627799] sd 0:0:0:0: [sda] Write Protect is off
[    1.628336] sd 0:0:0:0: [sda] Incomplete mode parameter data
[    1.628945] sd 0:0:0:0: [sda] Assuming drive cache: write through
[    1.630201] sd 0:0:0:0: Attached scsi generic sg0 type 0
[    1.631915] sd 0:0:0:0: [sda] Incomplete mode parameter data
[    1.632351] sd 0:0:0:0: [sda] Assuming drive cache: write through
[    1.674374]  sda: sda1 sda2 < sda5 >
[    1.687414] sd 0:0:0:0: [sda] Incomplete mode parameter data
[    1.688497] sd 0:0:0:0: [sda] Assuming drive cache: write through
[    1.689455] sd 0:0:0:0: [sda] Attached SCSI disk
Begin: Running /scripts/local-premount ... done.
[    1.839615] EXT4-fs (sda1): INFO: recovery required on readonly filesystem
[    1.840159] EXT4-fs (sda1): write access will be enabled during recovery
[    1.895657] EXT4-fs (sda1): orphan cleanup on readonly fs
[    1.897686] EXT4-fs (sda1): 1 orphan inode deleted
[    1.898288] EXT4-fs (sda1): recovery complete
[    1.899972] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts:
(null)
Begin: Running /scripts/local-bottom ... done.
done.
Begin: Running /scripts/init-bottom ... done.
[    2.457327] random: init urandom read with 59 bits of entropy available
[    2.560446] Switched to clocksource tsc
[    3.870212] random: nonblocking pool is initialized
[    5.859082] Adding 1046524k swap on /dev/sda5.  Priority:-1 extents:1 across:
1046524k FS
[    6.307908] piix4_smbus 0000:00:07.0: SMBus base address uninitialized - upgr
ade BIOS or use force_addr=0xaddr
```

Username : mininet
password: mininet



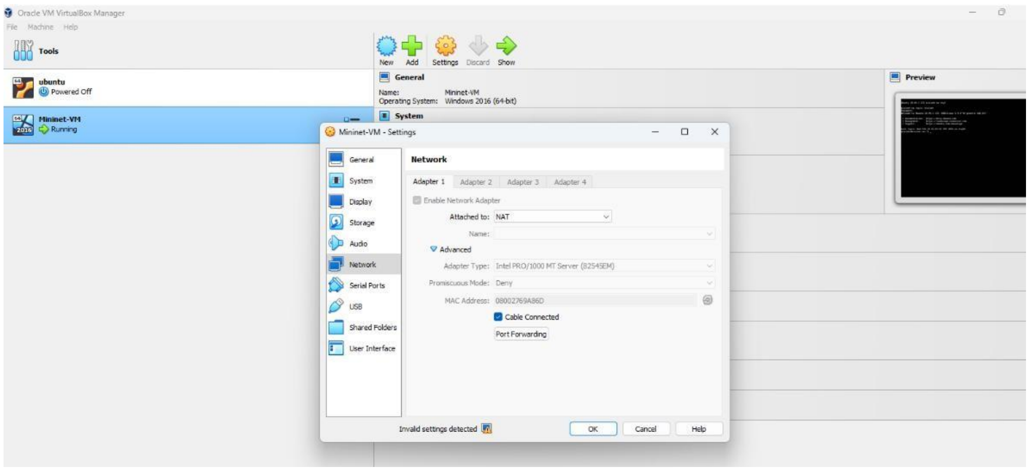
```
Ubuntu 14.04 LTS mininet-vm tty1

mininet-vm login: mininet
Password: _
```

Step 7**:** After started mininet and again go to the settings then click the NEXT, after go to “attach to” select NAT option



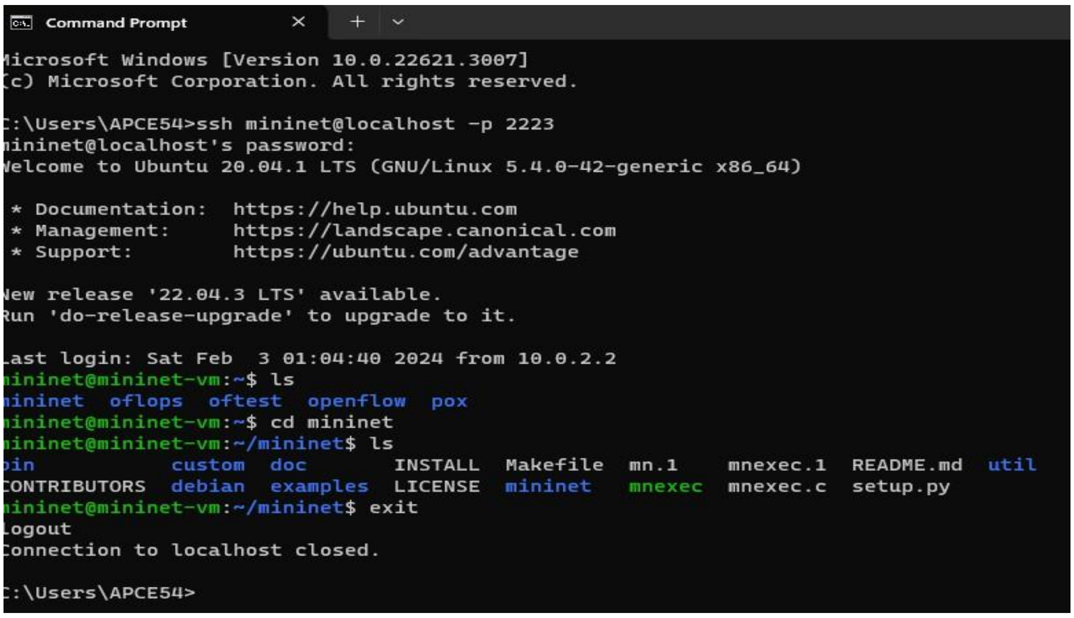
```
General
Name:              Mininet-VM
Operating System:  Windows 2016 (64-bit)
System
Base Memory:   2024 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  Nested Paging, PAE/NX, Hyper-V Paravirtualization
Display
Video Memory:           128 MB
Graphics Controller:    VBoxVGA
Remote Desktop Server:  Disabled
Recording:              Disabled
Storage
Controller: SCSI
 SCSI Port 0:    mininet-vm-x86_64.vdi (Normal, 8.00 GB)
Audio
Host Driver:  Default
Controller:   ICH AC97
Network
Adapter 1:  Intel PRO/1000 MT Server (NAT)
USB
USB Controller:  OHCI
Device Filters:  0 (0 active)
Shared folders
None
Description
None
```

Then advance to click the port forwarding. Then assign name=ssh, host port=2223,guest port=22 then click on OK.



```
Mininet-VM - Settings
Network
Adapter 1
Enable Network Adapter
Attached to:  NAT
Name:
Advanced
Adapter Type:       Intel PRO/1000 MT Server (82545EM)
Promiscuous Mode:   Deny
MAC Address:        08002769A86D
Cable Connected
Port Forwarding
Invalid settings detected
```

Step 8**:** After this step open command prompt and Type cmd “>ssh Mininet@localhost -p 2223” & password: Mininet” too see the virtual network established in virtualbox.



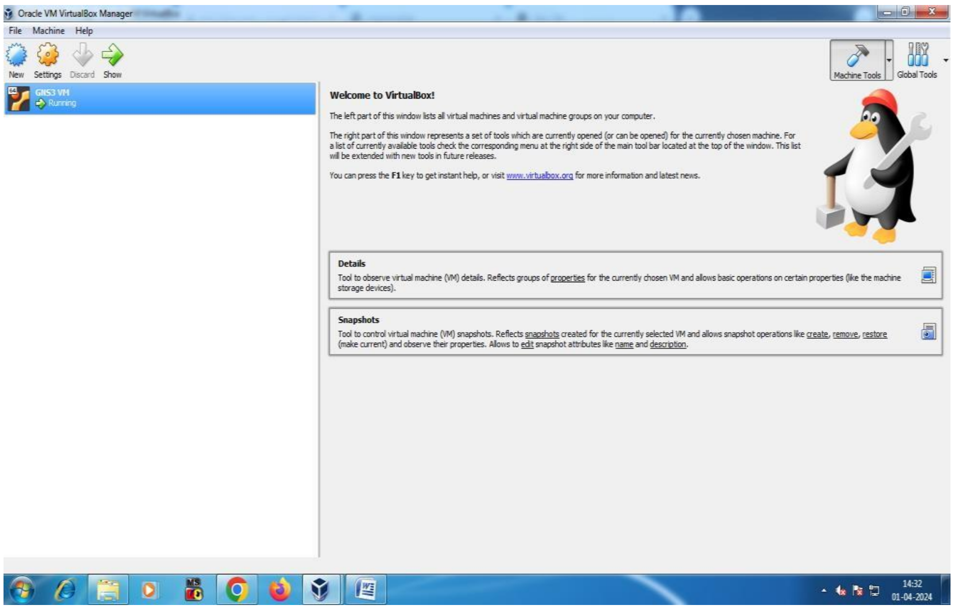
```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\APCE54>ssh mininet@localhost -p 2223
mininet@localhost's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Feb  3 01:04:40 2024 from 10.0.2.2
mininet@mininet-vm:~$ ls
mininet  oflops  oftest  openflow  pox
mininet@mininet-vm:~$ cd mininet
mininet@mininet-vm:~/mininet$ ls
bin           custom  doc       INSTALL  Makefile  mn.1    mnexec.1  README.md  util
CONTRIBUTORS  debian  examples  LICENSE  mininet   mnexec  mnexec.c  setup.py
mininet@mininet-vm:~/mininet$ exit
logout
Connection to localhost closed.

C:\Users\APCE54>
```

Type the commands $ls to list the open APIs and again use $cd Mininet to see the mininet environment and use $exit command to exit from that network.

ii) https://www.kathara.org

iii) GNS3

Step 1: Download GNS 3 software\



Welcome to VirtualBox!

The left part of this window lists all virtual machines and virtual machine groups on your computer.

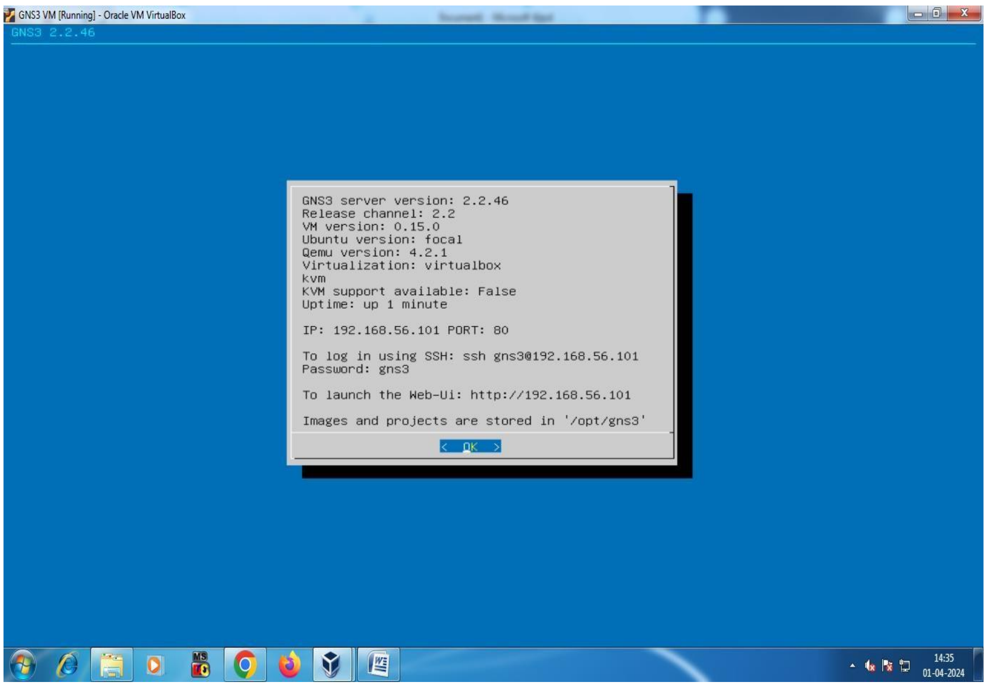
The right part of this window represents a set of tools which are currently opened (or can be opened) for the currently chosen machine. For a list of currently available tools check the corresponding menu at the right side of the main tool bar located at the top of the window. This list will be extended with new tools in future releases.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

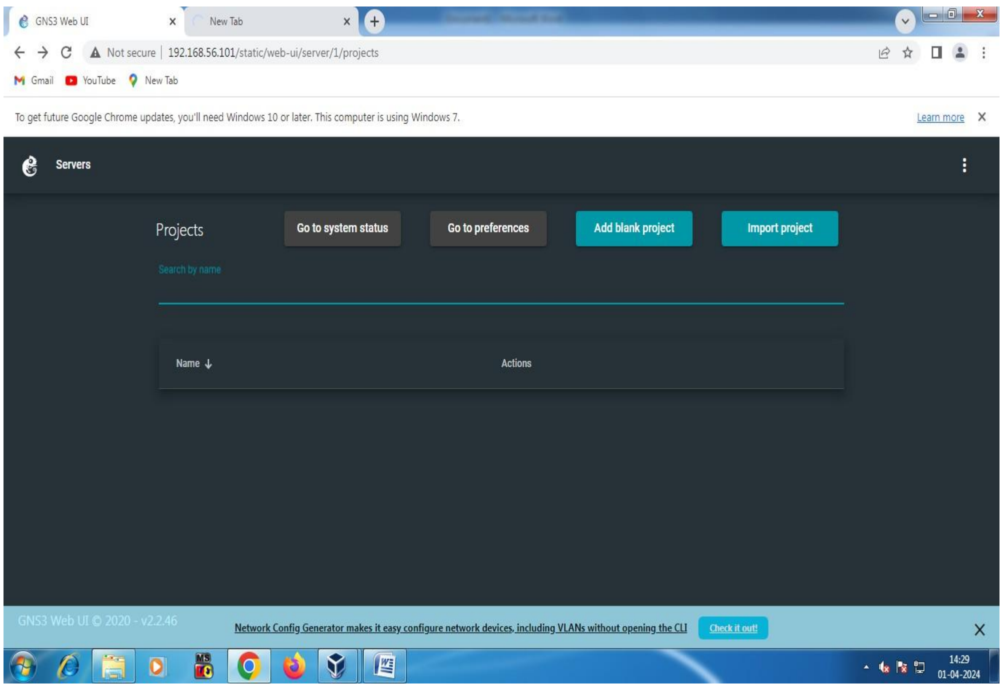
Details
Tool to observe virtual machine (VM) details. Reflects groups of properties for the currently chosen VM and allows basic operations on certain properties (like the machine storage devices).

Snapshots
Tool to control virtual machine (VM) snapshots. Reflects snapshots created for the currently selected VM and allows snapshot operations like create, remove, restore (make current) and observe their properties. Allows to edit snapshot attributes like name and description.

Step 2: Extract the GNS3 file . then go to virtual box -→ import appliance → To import GNS 3.OVA
Step 3: GNS 3 loading process



```
GNS3 server version: 2.2.46
Release channel: 2.2
VM version: 0.15.0
Ubuntu version: focal
Qemu version: 4.2.1
Virtualization: virtualbox
kvm
KVM support available: False
Uptime: up 1 minute

IP: 192.168.56.101 PORT: 80

To log in using SSH: ssh gns3@192.168.56.101
Password: gns3

To launch the Web-Ui: http://192.168.56.101

Images and projects are stored in '/opt/gns3'
                    < OK >
```



GNS3 Web UI
New Tab
Not secure | 192.168.56.101/static/web-ui/server/1/projects
Gmail
YouTube
New Tab
To get future Google Chrome updates, you'll need Windows 10 or later. This computer is using Windows 7.
Learn more
Servers
Projects
Go to system status
Go to preferences
Add blank project
Import project
Search by name
Name
Actions
GNS3 Web UI © 2020 - v2.2.46
Network Config Generator makes it easy configure network devices, including VLANs without opening the CLI
Check it out!
14:29
01-04-2024

RESULT:

Thus the above virtual box/mininet was successfully installed.

EX.NO: 2

DATE:

Create a simple mininet topology with SDN controller and use Wireshark to visualize the OpenFlow messages such as OpenFlow FLOW MOD, PACKET IN, PACKET OUT etc,.

AIM:

1. Create a simple Mininet network with an SDN controller (Ryu).
2. Capture OpenFlow messages, including FLOW_MOD, PACKET_IN, and PACKET_OUT, using Wireshark.
3. Visualize the captured OpenFlow messages for analysis.

PROCEDURE:

1. Install Mininet:
Ensure you have Mininet installed on your system. You can use a Linux distribution for this task.

2. Install Ryu SDN Controller:
Install Ryu, a popular SDN controller, using pip:

```
pip install ryu
```

3. Create the Mininet Topology:
Create a Python script (e.g., `mininet_topology.py`) to define your Mininet network topology and start Mininet with the Ryu controller. Here's an example topology with a single switch and two hosts:

```python
from mininet.net import Mininet
from mininet.topo import SingleSwitchTopo
from mininet.node import RemoteController
# Create a Mininet instance
net = Mininet(topo=SingleSwitchTopo(2), controller=RemoteController)
# Start Mininet
net.start()
```

4. Start the Ryu Controller:
In a separate terminal, start the Ryu controller:

```
ryu-manager
```

5. Capture OpenFlow Messages with Wireshark:

- Start Wireshark and select your network interface (e.g., `eth0`).
- Apply a display filter to capture only OpenFlow messages. Use the filter expression: `of`.
- Begin capturing packets by clicking the "Start" button in Wireshark.

6. Generate OpenFlow Messages:
In the Mininet terminal, you can use the Mininet CLI to generate OpenFlow messages. For example, you can add a flow rule (FLOW_MOD) or generate traffic (PACKET_OUT).

To add a flow rule (FLOW_MOD):

```
mininet> h1 ovs-ofctl add-flow s1 in_port=1,actions=output:2
```

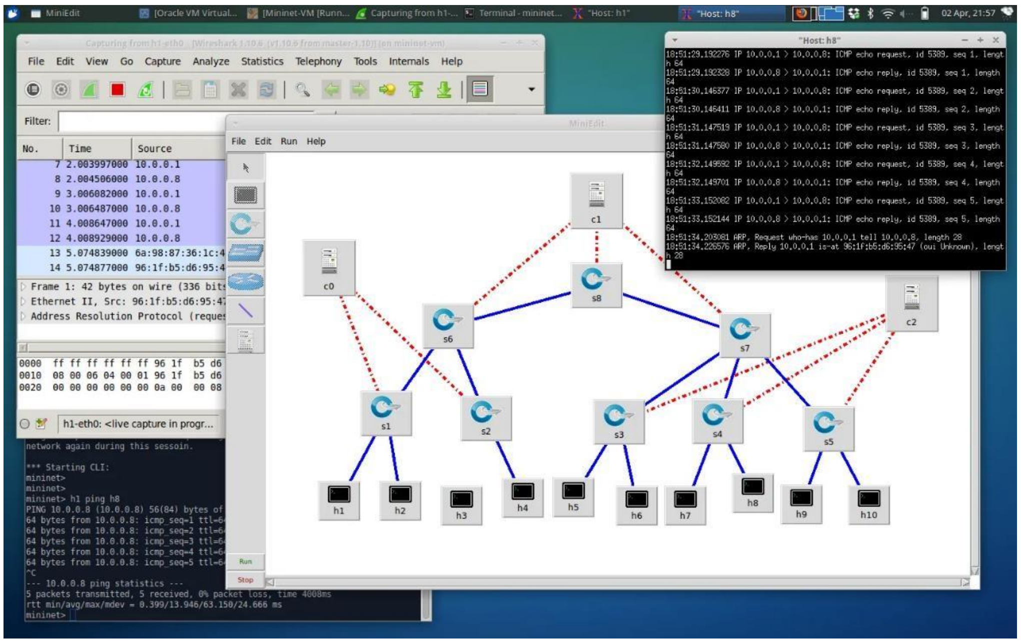
To generate traffic (PACKET_OUT):

```
mininet> h1 ping -c 1 h2
```

7. Stop Wireshark Capture:
   Stop capturing packets in Wireshark when you've generated enough OpenFlow messages.

OUTPUT:

- The Wireshark capture should display OpenFlow messages exchanged between the Ryu controller and the Mininet switch. You will see messages like FLOW_MOD, PACKET_IN, and PACKET_OUT, along with their details.

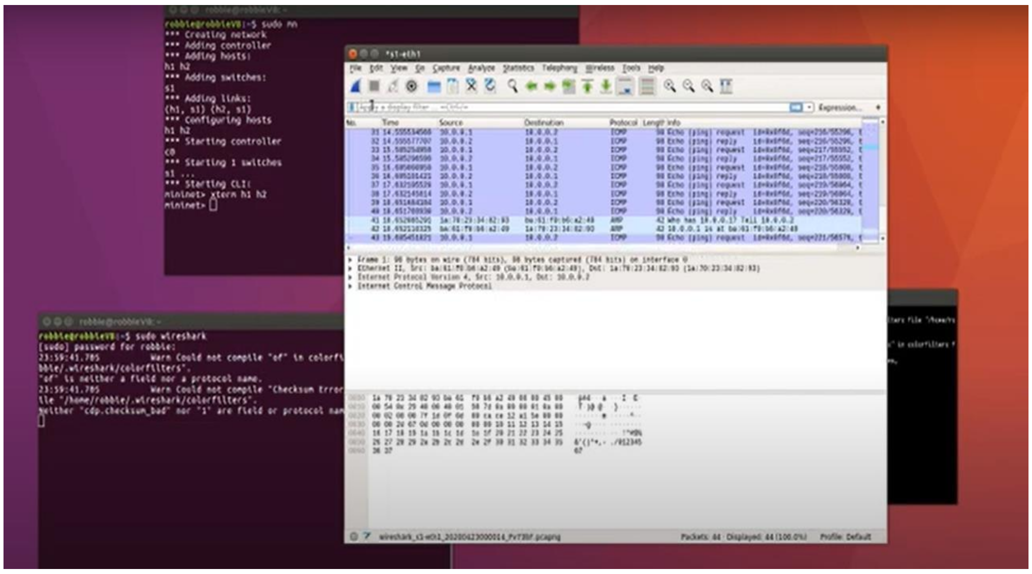


```
No.   Time          Source
   7 2.003997000 10.0.0.1
   8 2.004506000 10.0.0.8
   9 3.006082000 10.0.0.1
  10 3.006487000 10.0.0.8
  11 4.008647000 10.0.0.1
  12 4.008929000 10.0.0.8
  13 5.074839000 6a:98:87:36:1c:4
  14 5.074877000 96:1f:b5:d6:95:4
▷ Frame 1: 42 bytes on wire (336 bits
▷ Ethernet II, Src: 96:1f:b5:d6:95:47
▷ Address Resolution Protocol (reques

0000  ff ff ff ff ff ff 96 1f  b5 d6
0010  08 00 06 04 00 01 96 1f  b5 d6
0020  00 00 00 00 00 00 0a 00  00 08
```

```
network again during this sessoin.

*** Starting CLI:
mininet>
mininet>
mininet> h1 ping h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of
64 bytes from 10.0.0.8: icmp_seq=1 ttl=6
64 bytes from 10.0.0.8: icmp_seq=2 ttl=6
64 bytes from 10.0.0.8: icmp_seq=3 ttl=6
64 bytes from 10.0.0.8: icmp_seq=4 ttl=6
64 bytes from 10.0.0.8: icmp_seq=5 ttl=6
^C
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 0.399/13.946/63.150/24.666 ms
mininet>
```

```
18:51:29.192276 IP 10.0.0.1 > 10.0.0.8: ICMP echo request, id 5389, seq 1, lengt
h 64
18:51:29.192328 IP 10.0.0.8 > 10.0.0.1: ICMP echo reply, id 5389, seq 1, length
64
18:51:30.146377 IP 10.0.0.1 > 10.0.0.8: ICMP echo request, id 5389, seq 2, lengt
h 64
18:51:30.146411 IP 10.0.0.8 > 10.0.0.1: ICMP echo reply, id 5389, seq 2, length
64
18:51:31.147519 IP 10.0.0.1 > 10.0.0.8: ICMP echo request, id 5389, seq 3, lengt
h 64
18:51:31.147580 IP 10.0.0.8 > 10.0.0.1: ICMP echo reply, id 5389, seq 3, length
64
18:51:32.149692 IP 10.0.0.1 > 10.0.0.8: ICMP echo request, id 5389, seq 4, lengt
h 64
18:51:32.149701 IP 10.0.0.8 > 10.0.0.1: ICMP echo reply, id 5389, seq 4, length
64
18:51:33.152082 IP 10.0.0.1 > 10.0.0.8: ICMP echo request, id 5389, seq 5, lengt
h 64
18:51:33.152144 IP 10.0.0.8 > 10.0.0.1: ICMP echo reply, id 5389, seq 5, length
64
18:51:34.203081 ARP, Request who-has 10.0.0.1 tell 10.0.0.8, length 28
18:51:34.226576 ARP, Reply 10.0.0.1 is-at 96:1f:b5:d6:95:47 (oui Unknown), lengt
h 28
```

- Use Wireshark's visualization tools to analyze and inspect the captured OpenFlow messages. You can filter, sort, and drill down into specific messages to understand the communication between the controller and switches.



```
robbie@robbieVB:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1 h2
mininet> 
```

```
robbie@robbieVB:~$ sudo wireshark
[sudo] password for robbie:
23:59:41.705          Warn Could not compile "of" in colorf
bbie/.wireshark/colorfilters".
"of" is neither a field nor a protocol name.
23:59:41.705          Warn Could not compile "Checksum Error
ile "/home/robbie/.wireshark/colorfilters".
Neither "cdp.checksum_bad" nor "1" are field or protocol nam
```

```
No.  Time          Source             Destination        Protocol Length Info
  31 14.555534560  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x0f6d, seq=216/55296, t
  32 14.555577707  10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x0f6d, seq=216/55296, t
  33 15.585254958  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x0f6d, seq=217/55552, t
  34 15.585296598  10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x0f6d, seq=217/55552, t
  35 16.605060958  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x0f6d, seq=218/55808, t
  36 16.605101421  10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x0f6d, seq=218/55808, t
  37 17.632105526  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x0f6d, seq=219/56064, t
  38 17.632145614  10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x0f6d, seq=219/56064, t
  39 18.651484204  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x0f6d, seq=220/56320, t
  40 18.651700930  10.0.0.2           10.0.0.1           ICMP     98 Echo (ping) reply    id=0x0f6d, seq=220/56320, t
  41 18.652005291  1a:70:23:34:82:93  ba:61:f0:b6:a2:48  ARP      42 Who has 10.0.0.1? Tell 10.0.0.2
  42 18.652110325  ba:61:f0:b6:a2:48  1a:70:23:34:82:93  ARP      42 10.0.0.1 is at ba:61:f0:b6:a2:48
  43 19.605451825  10.0.0.1           10.0.0.2           ICMP     98 Echo (ping) request  id=0x0f6d, seq=221/56576, t

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸ Ethernet II, Src: ba:61:f0:b6:a2:49 (ba:61:f0:b6:a2:49), Dst: 1a:70:23:34:82:93 (1a:70:23:34:82:93)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
▸ Internet Control Message Protocol
```

*s1-eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
wireshark_s1-eth1_20200423000014_FvT3bf.pcapng
Packets: 44 · Displayed: 44 (100.0%)
Profile: Default

RESULT:

This setup allows you to observe and analyze the OpenFlow messaging in a simple SDN network. You can further customize the Mininet topology and generate more complex OpenFlow scenarios for testing and analysis.

EX.NO: 3

DATE:

Create a SDN application that uses the Northbound API to program flow table rules on the , Traffic Engineering, Firewall etc

AIM:

To Develop an SDN application that uses the Northbound API to program flow table rules on SDN switches for different use cases: L2 learning switch, Traffic Engineering, and Firewall.

PROCEDURE:

1. Set Up the Development Environment:
Install the Ryu SDN controller and any necessary Python libraries.

2. Create the Ryu SDN Application:
Create a Python script for your Ryu SDN application, which will implement the Northbound API to program flow rules.

3. L2 Learning Switch Use Case:
In your Ryu application, use the Northbound API to program flow rules for basic L2 learning. For example, when a packet arrives, add a flow entry to the switch's flow table based on the source MAC address and port.

4. Traffic Engineering Use Case:
Implement traffic engineering rules using the Northbound API. For instance, you can define flow rules to optimize traffic paths or prioritize specific traffic based on application requirements.

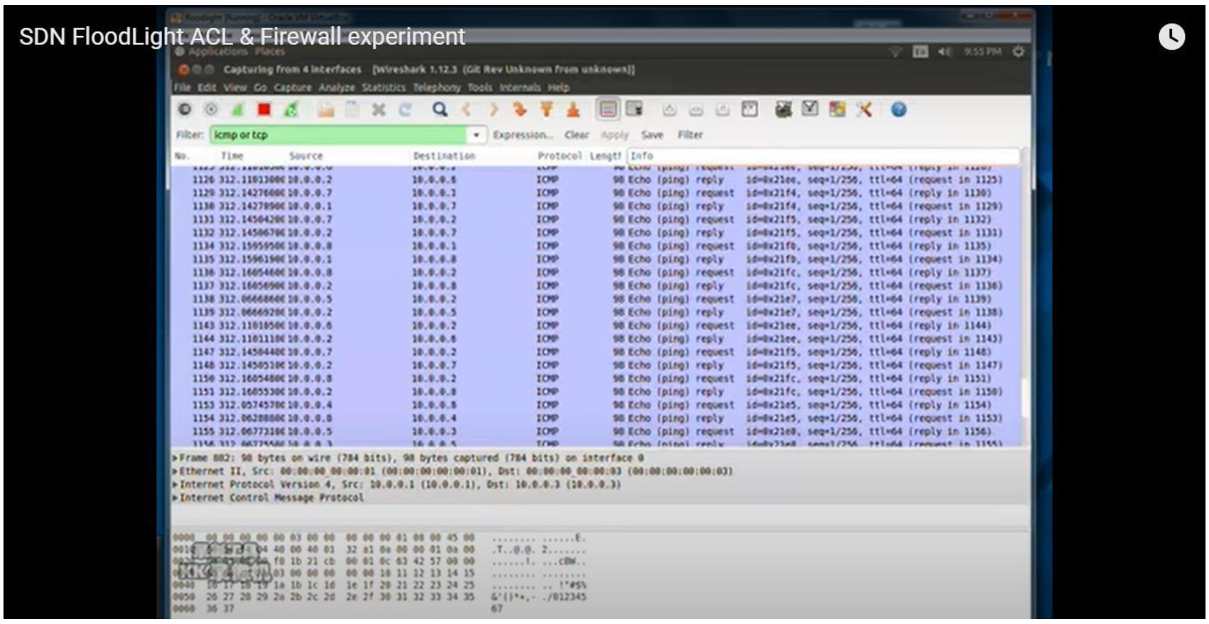
5. Firewall Use Case:
Implement firewall rules using the Northbound API to drop or allow specific traffic based on criteria such as source/destination IP addresses, ports, or protocols.

6. Run the SDN Application:
Start your Ryu SDN application with the Ryu manager using the following command:
```
ryu-manager your_application.py
```

7. Test the SDN Application:
Create a virtual network or use a Mininet topology for testing your SDN application. Generate traffic to see how the application programs the flow table rules.

OUTPUT:

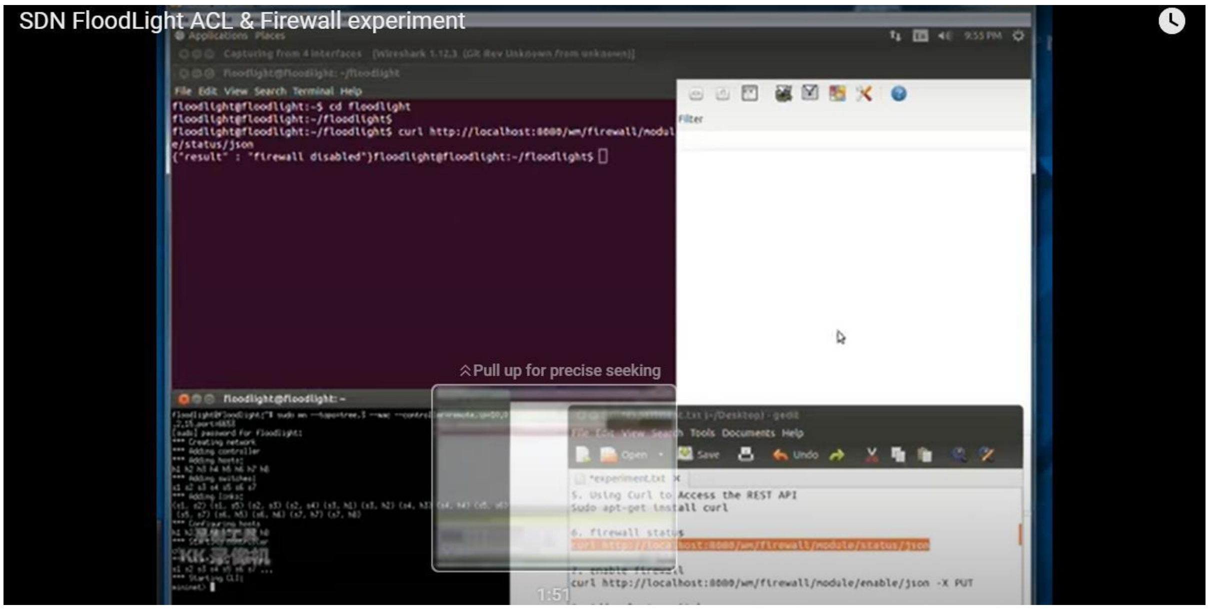
• For the L2 learning switch use case, the application should program flow rules that enable the switch to learn and forward traffic based on MAC addresses.


SDN FloodLight ACL & Firewall experiment
Capturing from 4 interfaces [Wireshark 1.12.3 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: icmp or tcp
Expression... Clear Apply Save Filter

```
No.   Time         Source      Destination  Protocol Lengt Info
1126 312.1101300( 10.0.0.2    10.0.0.6     ICMP     98 Echo (ping) reply    id=0x21ee, seq=1/256, ttl=64 (request in 1125)
1129 312.1427600( 10.0.0.7    10.0.0.1     ICMP     98 Echo (ping) request  id=0x21f4, seq=1/256, ttl=64 (reply in 1130)
1130 312.1427890( 10.0.0.1    10.0.0.7     ICMP     98 Echo (ping) reply    id=0x21f4, seq=1/256, ttl=64 (request in 1129)
1131 312.1450420( 10.0.0.7    10.0.0.2     ICMP     98 Echo (ping) request  id=0x21f5, seq=1/256, ttl=64 (reply in 1132)
1132 312.1450670( 10.0.0.2    10.0.0.7     ICMP     98 Echo (ping) reply    id=0x21f5, seq=1/256, ttl=64 (request in 1131)
1134 312.1595950( 10.0.0.8    10.0.0.1     ICMP     98 Echo (ping) request  id=0x21fb, seq=1/256, ttl=64 (reply in 1135)
1135 312.1596190( 10.0.0.1    10.0.0.8     ICMP     98 Echo (ping) reply    id=0x21fb, seq=1/256, ttl=64 (request in 1134)
1136 312.1605460( 10.0.0.8    10.0.0.2     ICMP     98 Echo (ping) request  id=0x21fc, seq=1/256, ttl=64 (reply in 1137)
1137 312.1605690( 10.0.0.2    10.0.0.8     ICMP     98 Echo (ping) reply    id=0x21fc, seq=1/256, ttl=64 (request in 1136)
1138 312.0666860( 10.0.0.5    10.0.0.2     ICMP     98 Echo (ping) request  id=0x21e7, seq=1/256, ttl=64 (reply in 1139)
1139 312.0666920( 10.0.0.2    10.0.0.5     ICMP     98 Echo (ping) reply    id=0x21e7, seq=1/256, ttl=64 (request in 1138)
1143 312.1101050( 10.0.0.6    10.0.0.2     ICMP     98 Echo (ping) request  id=0x21ee, seq=1/256, ttl=64 (reply in 1144)
1144 312.1101110( 10.0.0.2    10.0.0.6     ICMP     98 Echo (ping) reply    id=0x21ee, seq=1/256, ttl=64 (request in 1143)
1147 312.1450440( 10.0.0.7    10.0.0.2     ICMP     98 Echo (ping) request  id=0x21f5, seq=1/256, ttl=64 (reply in 1148)
1148 312.1450510( 10.0.0.2    10.0.0.7     ICMP     98 Echo (ping) reply    id=0x21f5, seq=1/256, ttl=64 (request in 1147)
1150 312.1605480( 10.0.0.8    10.0.0.2     ICMP     98 Echo (ping) request  id=0x21fc, seq=1/256, ttl=64 (reply in 1151)
1151 312.1605530( 10.0.0.2    10.0.0.8     ICMP     98 Echo (ping) reply    id=0x21fc, seq=1/256, ttl=64 (request in 1150)
1153 312.0574570( 10.0.0.4    10.0.0.8     ICMP     98 Echo (ping) request  id=0x21e5, seq=1/256, ttl=64 (reply in 1154)
1154 312.0628880( 10.0.0.8    10.0.0.4     ICMP     98 Echo (ping) reply    id=0x21e5, seq=1/256, ttl=64 (request in 1153)
1155 312.0677310( 10.0.0.5    10.0.0.3     ICMP     98 Echo (ping) request  id=0x21e8, seq=1/256, ttl=64 (reply in 1156)

▸Frame 882: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
▸Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: 00:00:00_00:00:03 (00:00:00:00:00:03)
▸Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.3 (10.0.0.3)
▸Internet Control Message Protocol

0000  00 00 00 00 00 03 00 00  00 00 00 01 08 00 45 00   ........ ......E.
0010  [illegible] 04 40 00 40 01  32 a1 0a 00 00 01 0a 00   .T..@.@. 2.......
0020  [illegible] f0 1b 21 cb  00 01 0c 63 42 57 00 00   ......!. ...cBW..
0030  [illegible] 03 00 00 00  00 00 10 11 12 13 14 15   ........ ........
0040  [illegible] 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ........ .. !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

• In the traffic engineering use case, the application should optimize traffic paths or prioritize certain types of traffic as per your defined rules.

• In the firewall use case, the application should allow or block traffic based on your defined criteria.


SDN FloodLight ACL & Firewall experiment
floodlight@floodlight: ~/floodlight
File Edit View Search Terminal Help
Pull up for precise seeking
floodlight@floodlight: ~
*experiment.txt (~/Desktop) - gedit
Tools Documents Help
Open Save Undo
1:51

```
floodlight@floodlight:~$ cd floodlight
floodlight@floodlight:~/floodlight$
floodlight@floodlight:~/floodlight$ curl http://localhost:8080/wm/firewall/modul
e/status/json
{"result" : "firewall disabled"}floodlight@floodlight:~/floodlight$
```

```
5. Using Curl to Access the REST API
Sudo apt-get install curl

6. firewall status
curl http://localhost:8080/wm/firewall/module/status/json

7. enable firewall
curl http://localhost:8080/wm/firewall/module/enable/json -X PUT
```

• The application's output should include log messages or other forms of feedback, showing that it is functioning correctly.

RESULT:

The result is a working SDN application that leverages the Northbound API to program flow table rules on SDN switches for different use cases. The application should effectively control network traffic based on the specified policies and rules.

EX.NO: 4 Create a simple end-to-end network service with two VNFs Using vim-emu https://github.com/containernet/vim-emu

DATE:

AIM:

To create a simple end-to-end network service with two VNFs using vim- emu.

PROCEDURE:

1. Set Up the Environment:

Make sure you have a Linux system. You can set up a virtual machine or a dedicated system.

- Install the required dependencies, including Docker and Docker Compose.

2. Install vim-emu:

Follow the instructions in the vim-emu GitHub repository (https://github.com/containernet/vim-emu) to install vim-emu.

3. Create Network Topology:

Define a network topology using vim-emu. You can create a Python script that specifies the network, VNFs, and their interconnections. For example, create a file named `network_topology.py`:

```python
from mininet.net import Containernet
from mininet.node import Docker
from mininet.link import TCLink
from mininet.cli import CLI
net = Containernet()

vnf1 = net.addDocker('vnf1', dimage="vnf1_image")
vnf2 = net.addDocker('vnf2', dimage="vnf2_image")
net.addLink(vnf1, vnf2, cls=TCLink)
net.start()
CLI(net)
net.stop()
```

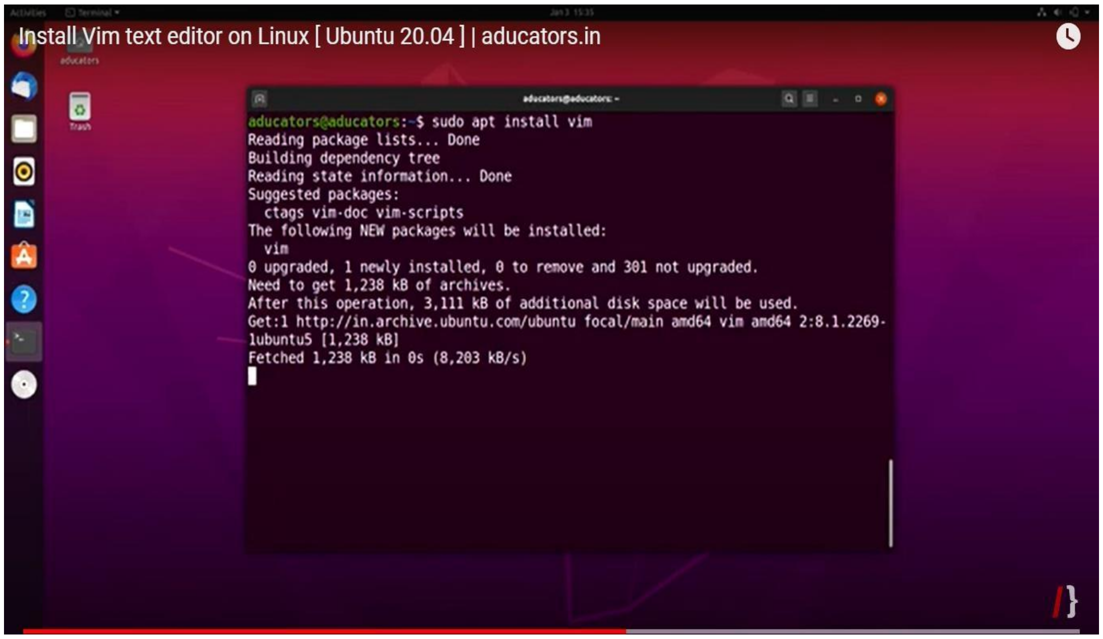
4. Create VNF Docker Images:
Build Docker images for your VNFs. Create a `Dockerfile` for each VNF, specifying its requirements and configurations. Build the images using Docker commands.
5. Launch the Network Topology:
Run your network topology with vim-emu:

```
sudo python network_topology.py
```

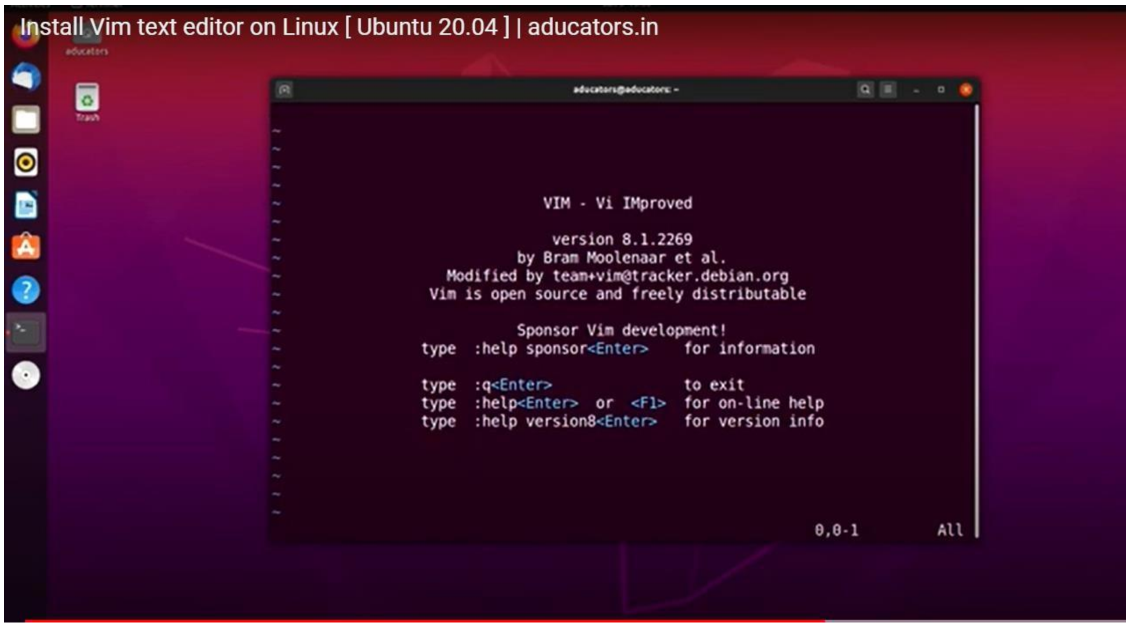
This will start the network and deploy the VNFs.

OUTPUT:

- When running the network topology script with vim-emu, you should see the network and VNFs being deployed. You can test connectivity and traffic between VNFs.


Install Vim text editor on Linux [Ubuntu 20.04] | aducators.in
aducators@aducators: ~

```
aducators@aducators:~$ sudo apt install vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim
0 upgraded, 1 newly installed, 0 to remove and 301 not upgraded.
Need to get 1,238 kB of archives.
After this operation, 3,111 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal/main amd64 vim amd64 2:8.1.2269-
1ubuntu5 [1,238 kB]
Fetched 1,238 kB in 0s (8,203 kB/s)
```


Install Vim text editor on Linux [Ubuntu 20.04] | aducators.in

```
                 VIM - Vi IMproved

                 version 8.1.2269
             by Bram Moolenaar et al.
     Modified by team+vim@tracker.debian.org
   Vim is open source and freely distributable

             Sponsor Vim development!
type  :help sponsor<Enter>    for information

type  :q<Enter>               to exit
type  :help<Enter>  or  <F1>  for on-line help
type  :help version8<Enter>   for version info

                                        0,0-1         All
```

RESULT:

The result is a fully orchestrated end-to-end network service with two VNFs, established using vim-emu and OSM. The network service is now ready to accept and process traffic as specified in your descriptor.

EX.NO: 5

Install OSM and onboard and orchestration network service

DATE:

AIM:

To Install OSM and onboard and orchestrate the network service.

PROCEDURE:

1. Install OSM:
 Follow the instructions on the OSM GitHub repository (https://osm.etsi.org) to install OSM, which includes installing the OSM client and server components.
2. Onboard the Network Service:
 Use the OSM client to onboard your network service. You'll need to provide a descriptor (e.g., a TOSCA YAML file) for your service.

```
osm ns-create <your_network_service_descriptor_file>
```

3. Instantiate the Network Service:
 Instantiate the network service using OSM:

```
osm ns-instantiate <ns-instance-name> <your_network_service_name>
```

 Replace `<ns-instance-name>` with a suitable name for your instantiated service.

OUTPUT:

- After installing OSM and onboarding the network service descriptor, you should see your network service listed within OSM.
- When instantiating the network service, OSM will orchestrate the deployment of the VNFs and connect them according to your descriptor.

RESULT:

Thus the result of install OSM and onboard and orchestrate the network service. The network service is now ready to accept and process traffic as specified in your descriptor.