

**EX.No:1.a**

## **Installation of VirtualBox /VMware Workstation and Mininet**

**Date:**

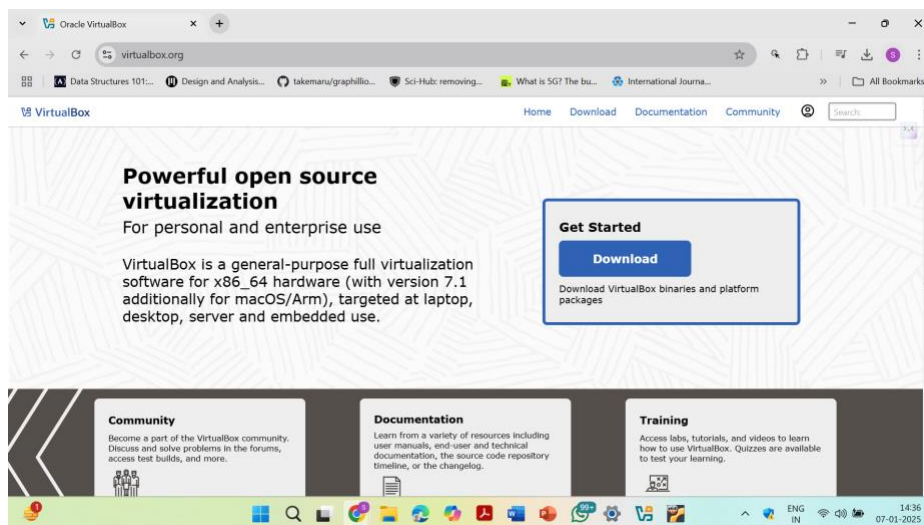
### **Aim:**

To install Virtualbox/VMware Workstation and Mininet environment for Software Defined Network and run basic Virtual box and Mininet commands.

### **Procedure:**

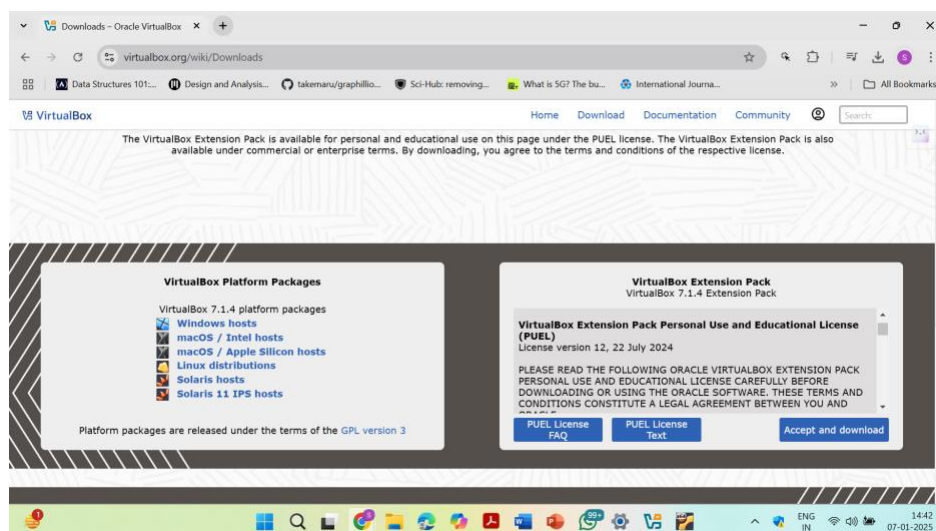
#### **Installation of VirtualBox in Windows**

Step 1: Go to the **VirtualBox Website** through the <https://www.virtualbox.org/> and click download.

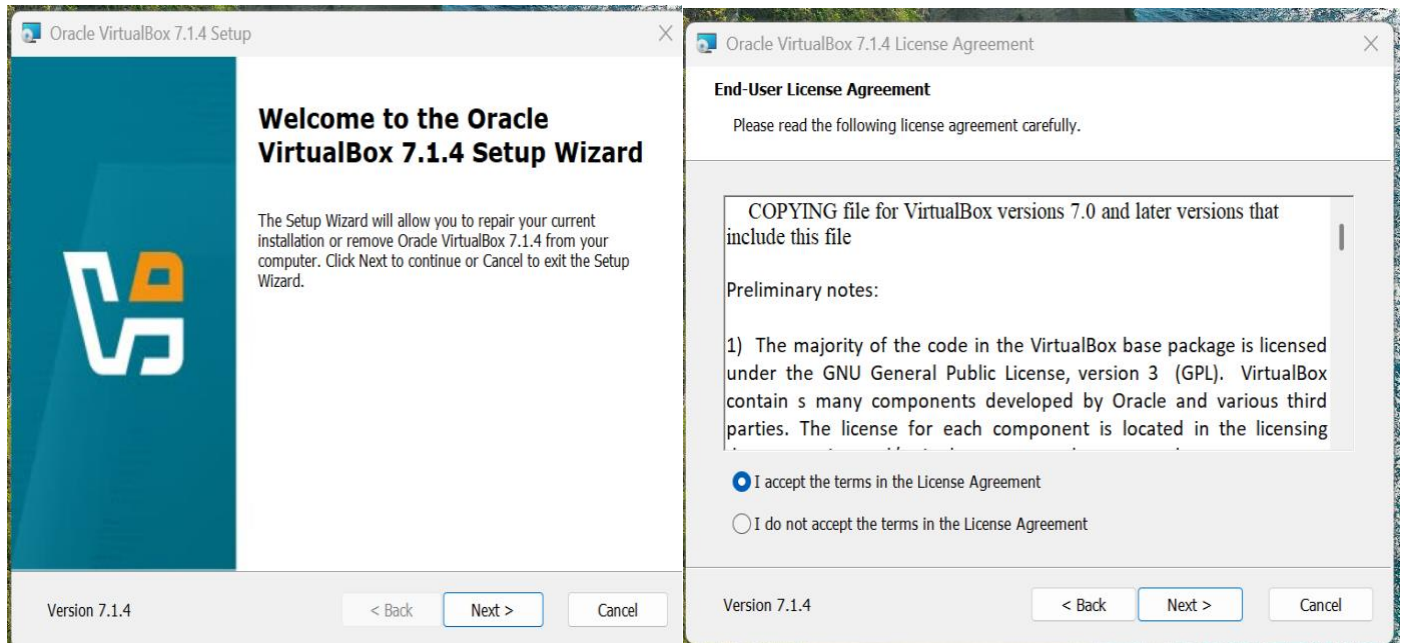


Step 2: Click Windows Host to download Virtual Box for Windows OS.

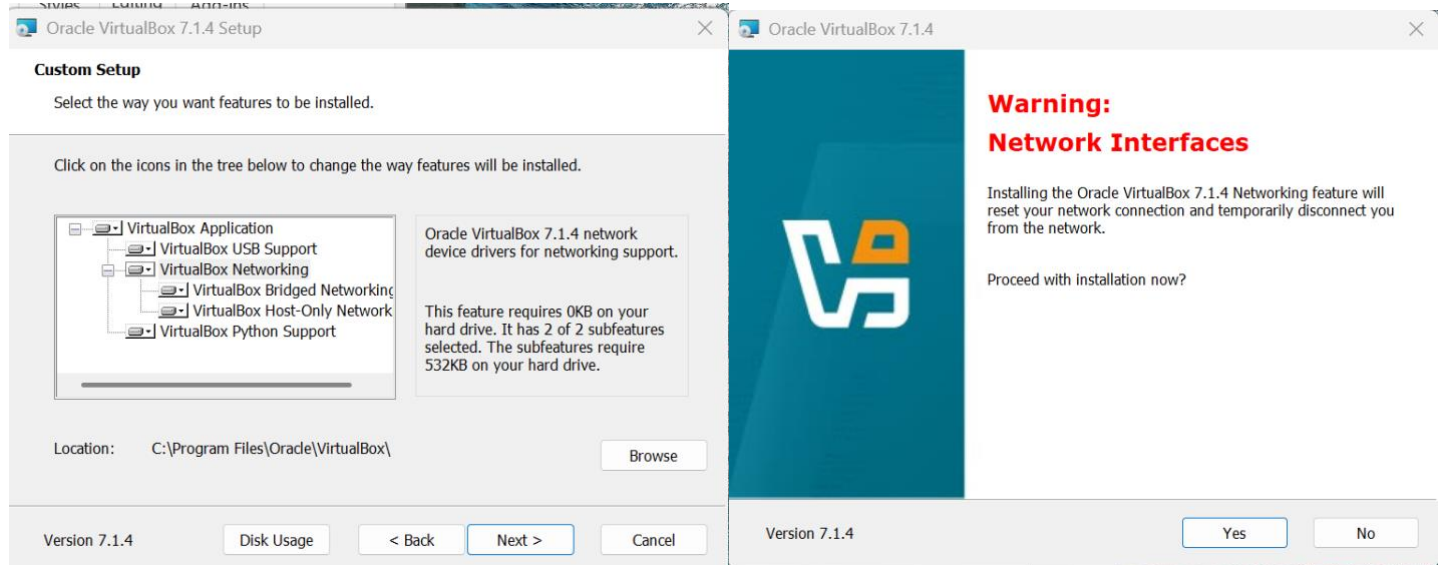
<https://download.virtualbox.org/virtualbox/7.1.4/VirtualBox-7.1.4-165100-Win.exe>

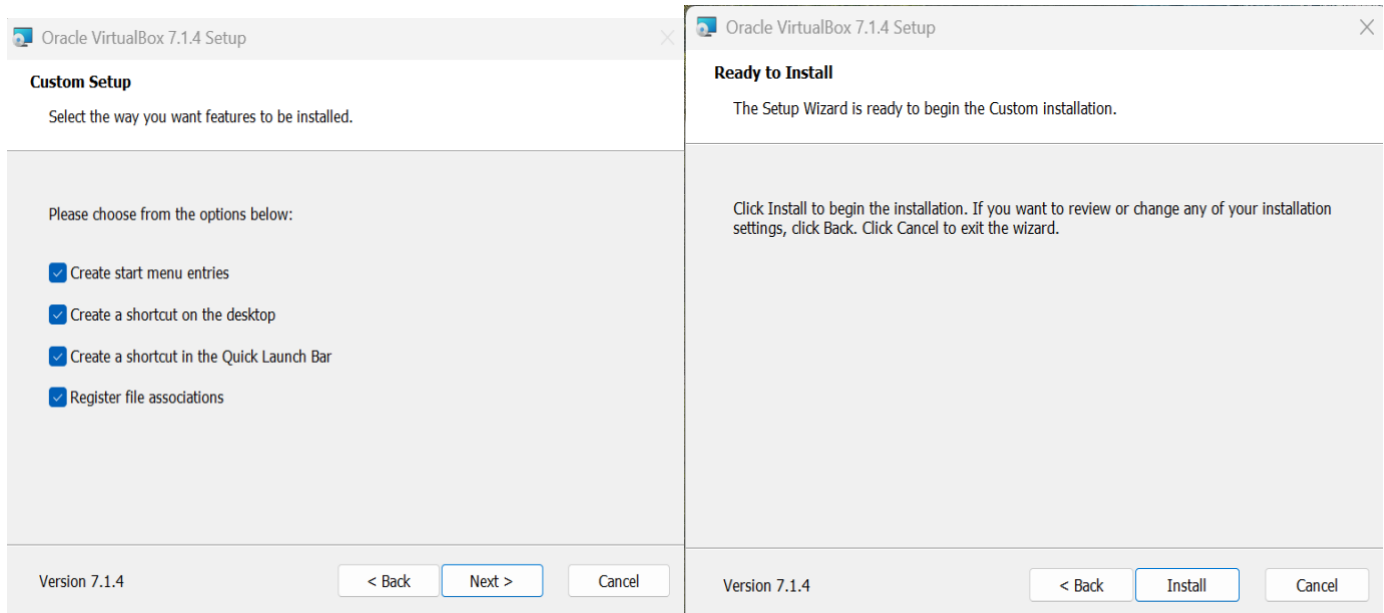


Step 3: Follow the steps to Install virtual box

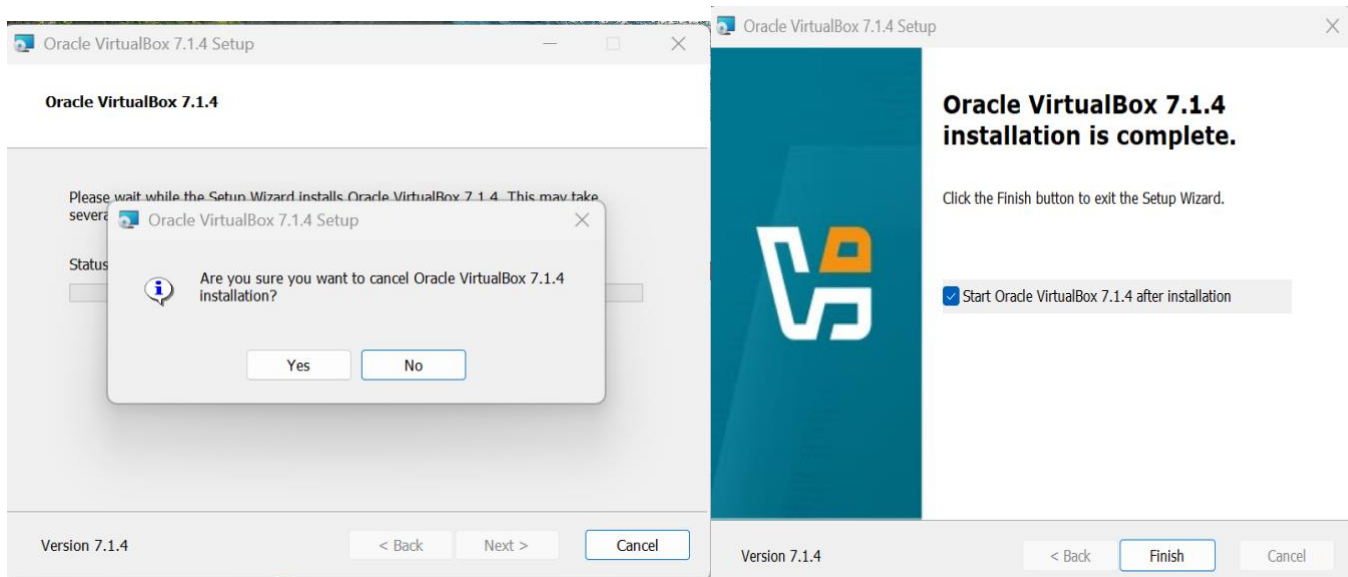


Choose the **installation folder** and click on the **Next** button.



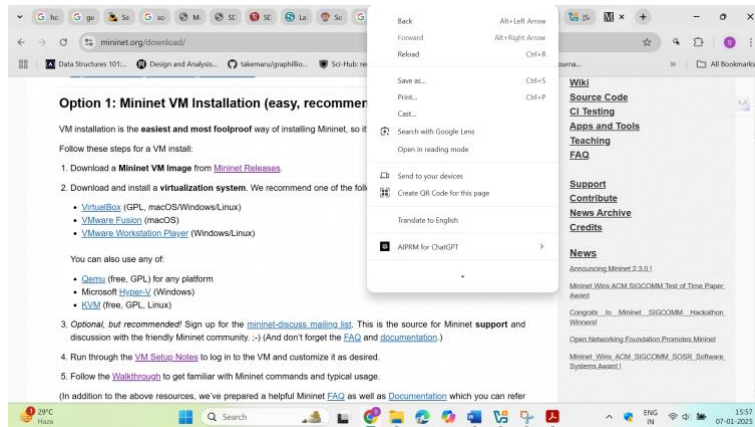


Click on “Yes” to continue the installation

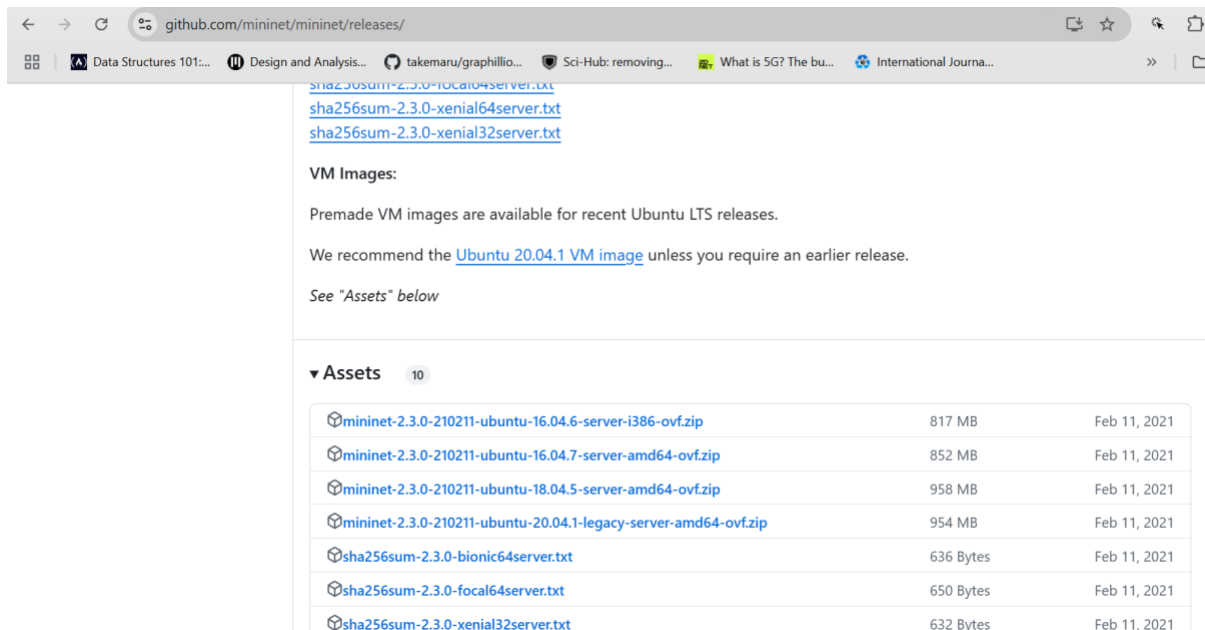


## Steps to interface Mininet in Virtual Box

Step 1: Download and Install Mininet from <https://mininet.org/download/>



Step 2: Click on Download a **Mininet VM Image** from [Mininet Releases](#).

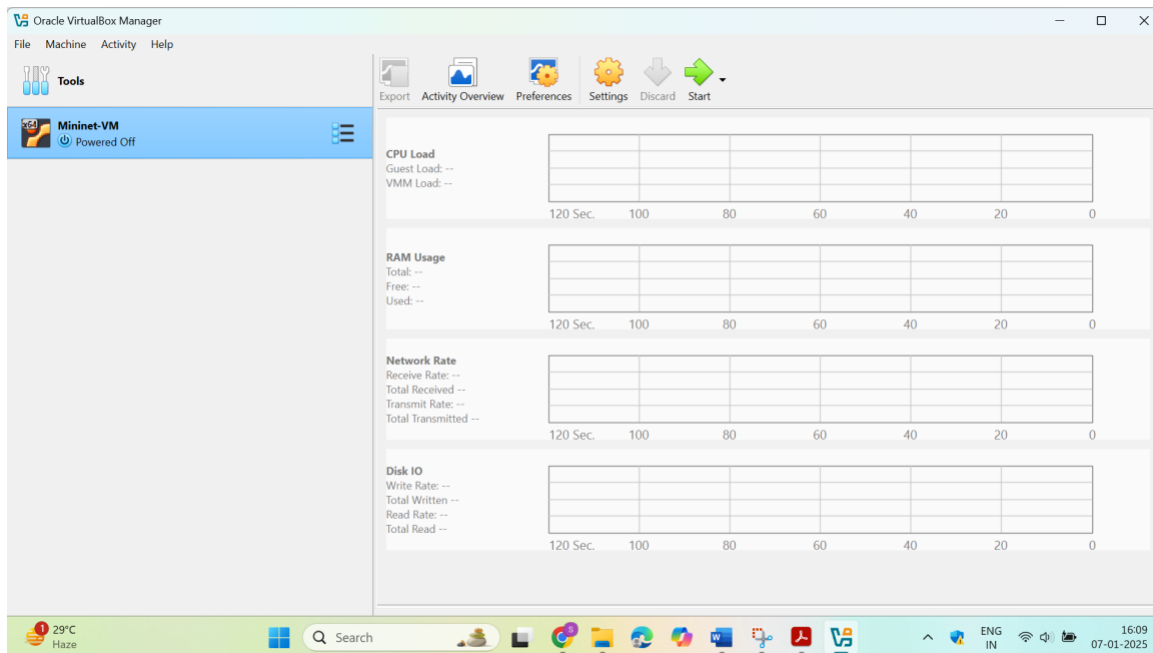
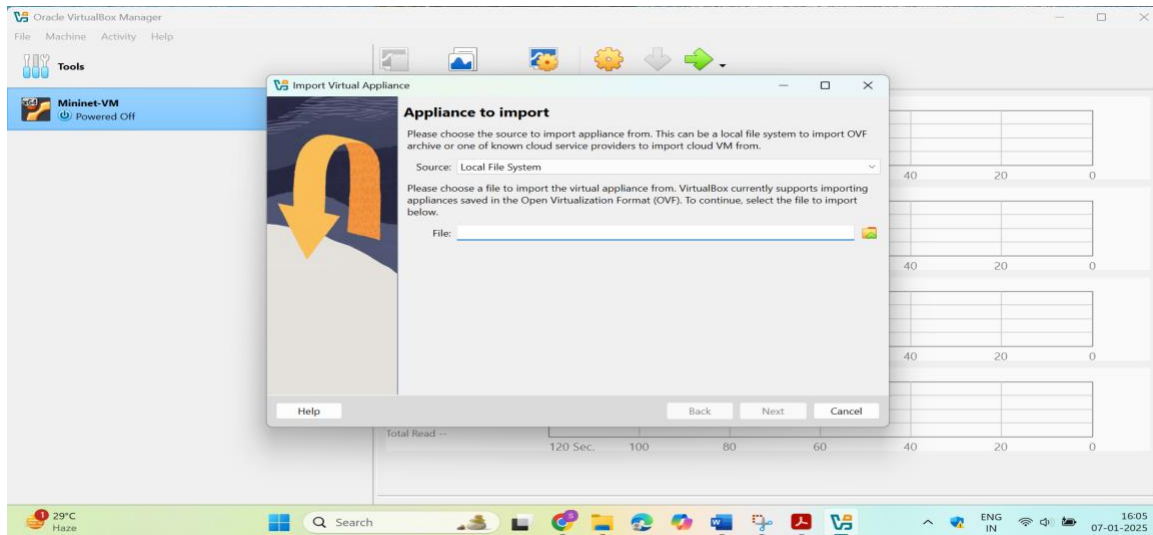


Click on [mininet-2.3.0-210211-ubuntu-16.04.6-server-i386-ovf.zip](#) from Asset  
<https://github.com/mininet/mininet/releases/download/2.3.0/mininet-2.3.0-210211-ubuntu-16.04.6-server-i386-ovf.zip>

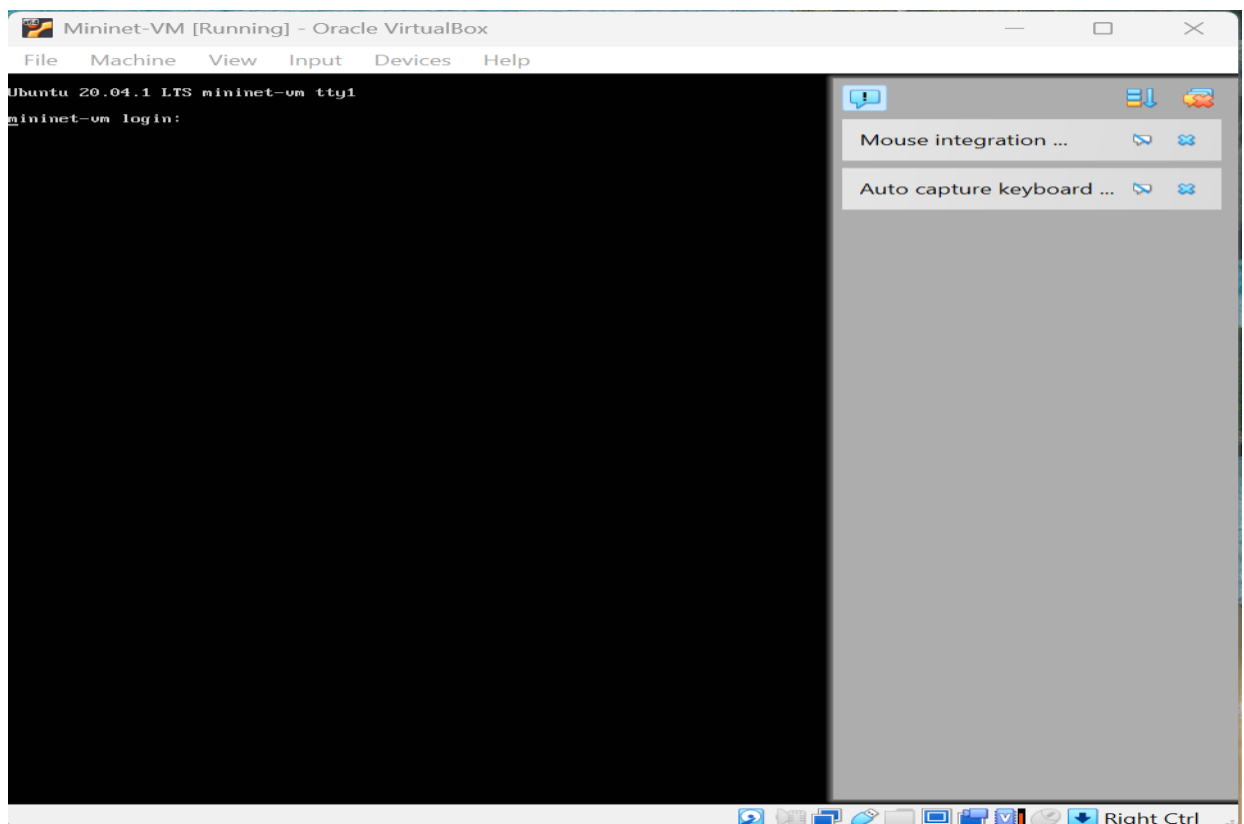
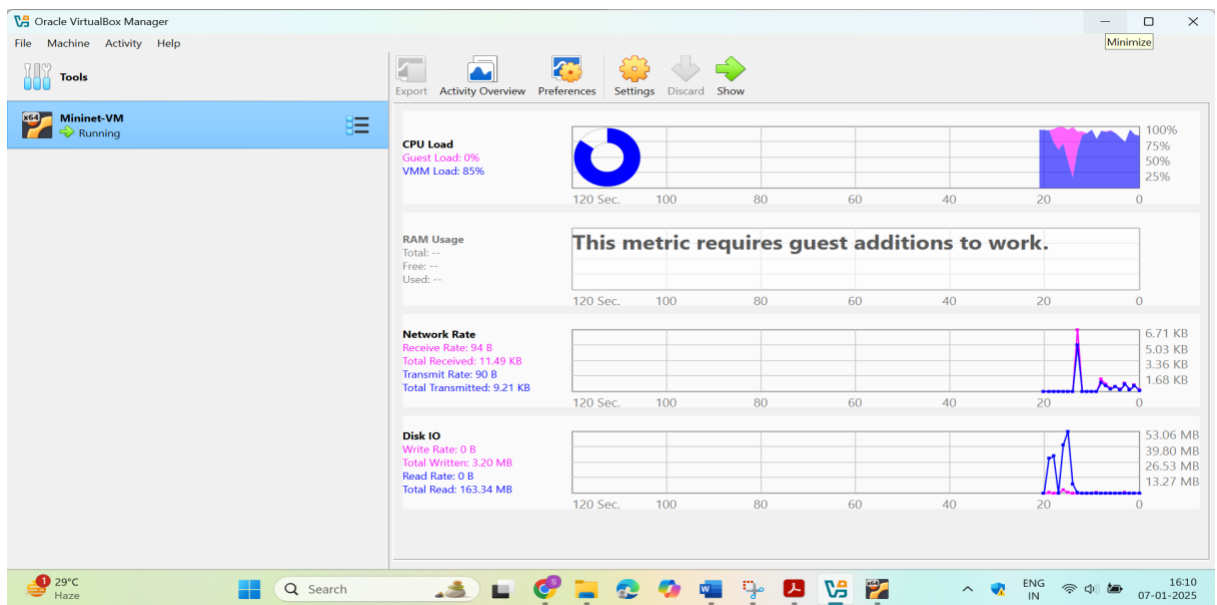
The Zip file Contains Open Virtualization Format file mininet-2.3.0-210211-ubuntu-20.04.1-legacy-server-amd64.ovf

### Step 3: Open Virtual Box 7.1.4

And click on Import virtual appliance and upload the above OVF file in Mininet



Step 4: Click on start to start the Mininet Virtual Machine.



Login: mininet

Password: mininet

```

Ubuntu 14.04.4 LTS mininet-vm tty1
mininet-vm login: mininet
Password:
Last login: Tue Mar 21 21:13:43 PDT 2017 on ttyS0
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic i686)

 * Documentation:  https://help.ubuntu.com/
mininet@mininet-vm:~$ _

```

## Mininet Comments and its usage:

1. Command to display a help message describing Mininet's startup options

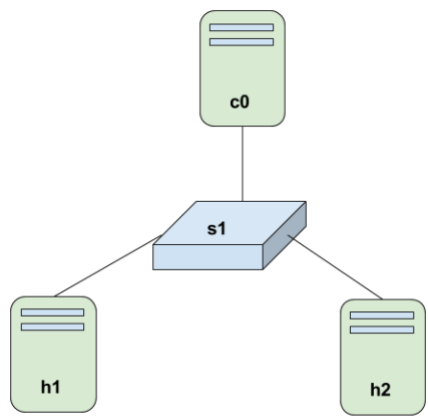
\$ sudo mn -h

```

...l linear=LinearTopo
    reversed=SingleSwitchReversedTopo tree=TreeTopo
    single=SingleSwitchTopo torus=TorusTopo
    minimal=MinimalTopo
-c, --clean          clean and exit
--custom=CUSTOM      read custom classes or params from .py file(s)
--test=TEST          none|build|all|liperflpingpairliperfudplpingall
-x, --xterms         spawn xterms for each node
-i IPBASE, --ibase=IPBASE
                    base IP address for hosts
--mac               automatically set host MACs
--arp              set all-pairs ARP entries
-u VERBOSITY, --verbosity=VERBOSITY
                    info|warn|warning|critical|error|debug|output
--innamespace       sw and ctrl in namespace?
--listenport=LISTENPORT
                    base port for passive switch listening
--nolistenport      don't use passive listening port
--pre=PRE           CLI script to run before tests
--post=POST         CLI script to run after tests
--pin              pin hosts to CPU cores (requires --host cfs or --host
                    rt)
--nat               [option=val...] adds a NAT to the topology that
                    connects Mininet hosts to the physical network.
                    Warning: This may route any traffic on the machine
                    that uses Mininet's IP subnet into the Mininet
                    network. If you need to change Mininet's IP subnet,
                    see the --ibase option.
--version           prints the version and exits
-w, --wait          wait for switches to connect
-t WAIT, --twait=WAIT
                    timed wait (s) for switches to connect
--cluster=server1,server2...
                    run on multiple servers (experimental!)
--placement=block|random
                    node placement for --cluster (experimental!)
mininet@mininet-vm:~$

```

2. Create a minimal topology network which includes one OpenFlow kernel switch connected to two hosts, plus the OpenFlow reference controller.



\$ sudo mn

```

mininet@mininet-vm:~$ 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>

```

### 3. Display Mininet CLI commands

mininet> help

```

mininet> help
Documented commands (type help <topic>):
=====
EOF  gterm  iperfudp  nodes      pingpair    py      switch  xterm
dpctl help  link      noecho     pingpairfull  quit    time
dump  intfs  links     pingall     ports      sh      wait
exit  iperf  net       pingallfull  px        source  x

You may also send a command to a node using:
<node> command {args}
For example:
mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
mininet> xterm h2

mininet> _

```



#### 4. Display nodes

mininet> nodes

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> _
```

#### 5. Display links

mininet> net

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet> _
```

#### 6. Dump information about all nodes:

mininet> dump

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=2489>
<Host h2: h2-eth0:10.0.0.2 pid=2492>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=2498>
<Controller c0: 127.0.0.1:6653 pid=2482>
mininet> _
```

#### 7. View the network interfaces on *h1*

mininet> h1 ifconfig -a

```
mininet> h1 ifconfig -a
h1-eth0  Link encap:Ethernet  HWaddr 62:df:3b:1d:92:66
         inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

8. Show the switch interfaces, plus the VM's connection out (eth0).

```
mininet> s1 ifconfig -a
```

```
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:253 errors:0 dropped:0 overruns:0 frame:0
            TX packets:253 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:14708 (14.7 KB)  TX bytes:14708 (14.7 KB)

vhost-net0  Link encap:Ethernet  HWaddr b2:c5:38:d5:51:6b
            BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1          Link encap:Ethernet  HWaddr 16:ad:fa:e2:9a:49
            BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1-eth1     Link encap:Ethernet  HWaddr c2:a3:e8:5d:de:28
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

s1-eth2     Link encap:Ethernet  HWaddr c6:d4:01:99:5b:8f
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

9. Print the process list from a host process:

```
mininet> h1 ps -a
```

```
mininet> h1 ps -a
  PID TTY          TIME CMD
 1634 tty1        00:00:00 bash
 2476 tty1        00:00:00 sudo
 2477 tty1        00:00:00 mn
 2521 pts/0        00:00:00 controller
 2680 pts/1        00:00:00 ps
mininet> _
```

10. Print the process list from root network namespace

```
mininet> s1 ps -a
```

```
mininet> s1 ps -a
  PID TTY          TIME CMD
 1634 tty1        00:00:00 bash
 2476 tty1        00:00:00 sudo
 2477 tty1        00:00:00 mn
 2521 pts/0        00:00:00 controller
 2692 pts/3        00:00:00 ps
mininet>
```

11. Test connectivity between hosts

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

```
mininet> pingall
```

```
mininet> h1 ping -c 1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.27 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.271/1.271/1.271/0.000 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

12. To Run a simple web server and client

```
mininet> h1 python -m http.server 80 &
```

```
mininet> h2 wget -O - h1
```

```
mininet> h1 kill %python
```

```
mininet> h1 python -m http.server 80 &
mininet> h2 wget -O - h1
--2025-01-07 20:49:18-- http://10.0.0.1/
Connecting to 10.0.0.1:80... failed: Connection refused.
mininet> h1 kill % python
/usr/bin/python: No module named http
bash: kill: python: arguments must be process or job IDs
mininet> _
```

13. To Exit the Mininet CLI:

```
mininet> exit
```

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 1045.073 seconds
mininet@mininet-vm:~$ _
```

14. To Cleanup Mininet

```
$ sudo mn -c
```

```
mininet@mininet-vm:~$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([_.,[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
mininet@mininet-vm:~$ _
```

15. Create a minimal Mininet, run an iperf server on one host, run an iperf client on the second host, and parse the bandwidth achieved

```
$ sudo mn --test iperf
```

```

mininet@mininet-vm:~$ sudo mn --test iperf
*** 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 ...
*** Waiting for switches to connect
s1
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['71.5 Gbits/sec', '71.6 Gbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 10.487 seconds
mininet@mininet-vm:~$

```

16. To verify all-pairs ping connectivity with one switch and three hosts:

\$ sudo mn --test pingall --topo single,3

```

mininet@mininet-vm:~$ sudo mn --test pingall --topo single,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 1 switches
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 5.204 seconds
mininet@mininet-vm:~$

```

17. Verify all-pairs ping connectivity with a linear topology (where each switch has one host, and all switches connect in a line)

```
$ sudo mn --test pingall --topo linear,4
```

```
mininet@mininet-vm:~$ sudo mn --test pingall --topo linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Waiting for switches to connect
s1 s2 s3 s4
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.....
*** Stopping 4 switches
s1 s2 s3 s4
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 5.445 seconds
mininet@mininet-vm:~$ _
```

**Result:** Thus virtualBox and Mininet environment for working with software defined networks has been successfully installed and Mininet CLI commands are executed.