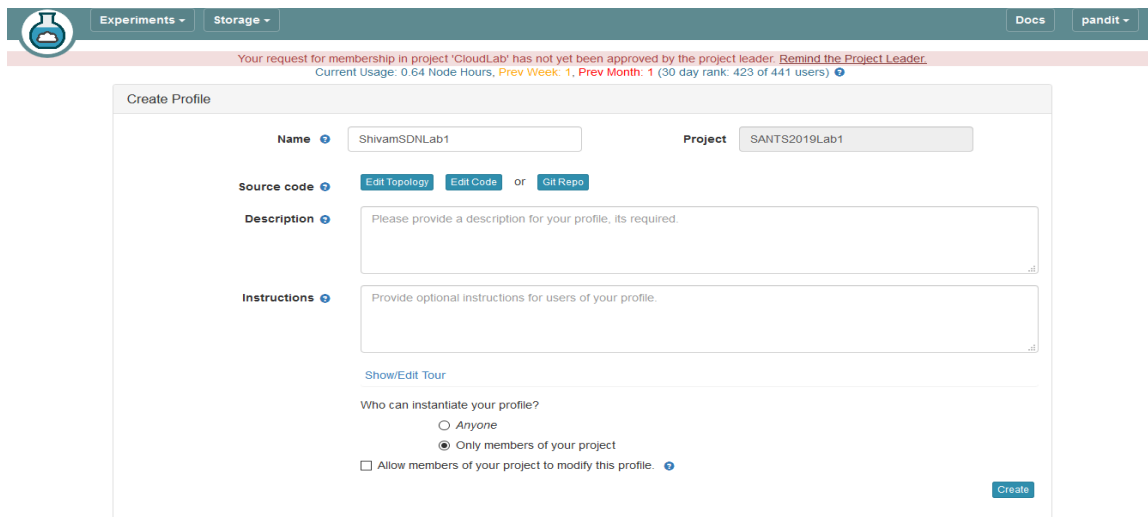


CloudLab and SDN Basic Lab-1

By: Shivam Pandit

1. Creating Profile

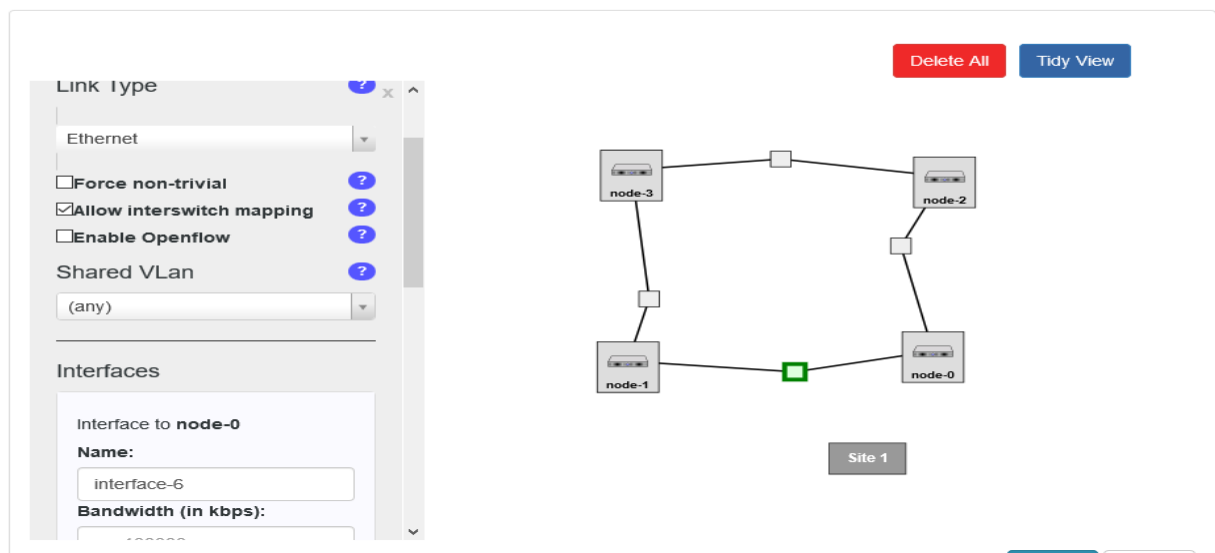
We first create a profile with 4 Xen VMs, Ubuntu 16OS Hardware type is set as any, node type as emulab-xen and link type as ethernet. After creating a topology, we accept and instantiate.



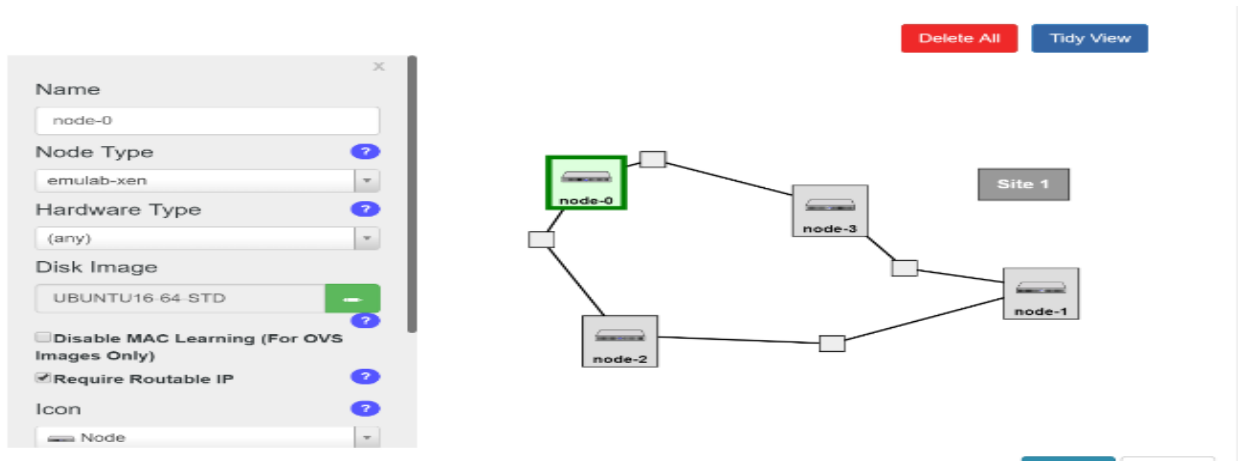
The screenshot shows the 'Create Profile' form in the CloudLab interface. At the top, there's a navigation bar with 'Experiments' and 'Storage' tabs, and a 'Docs' button. A message bar indicates that the user's request for membership in the 'CloudLab' project has not yet been approved. The form itself has a 'Name' field set to 'ShivamSDNLab1' and a 'Project' dropdown set to 'SANTS2019Lab1'. Below these are sections for 'Source code' (with buttons for 'Edit Topology', 'Edit Code', and 'Git Repo'), 'Description' (a text area), and 'Instructions' (another text area). There's a 'Show/Edit Tour' link. At the bottom, there's a section 'Who can instantiate your profile?' with radio buttons for 'Anyone' and 'Only members of your project' (which is selected), and a checkbox for 'Allow members of your project to modify this profile.' A 'Create' button is at the bottom right.

- Figure 1: Start to create experiment profile. Click “Create Topology” to start

Topology Editor



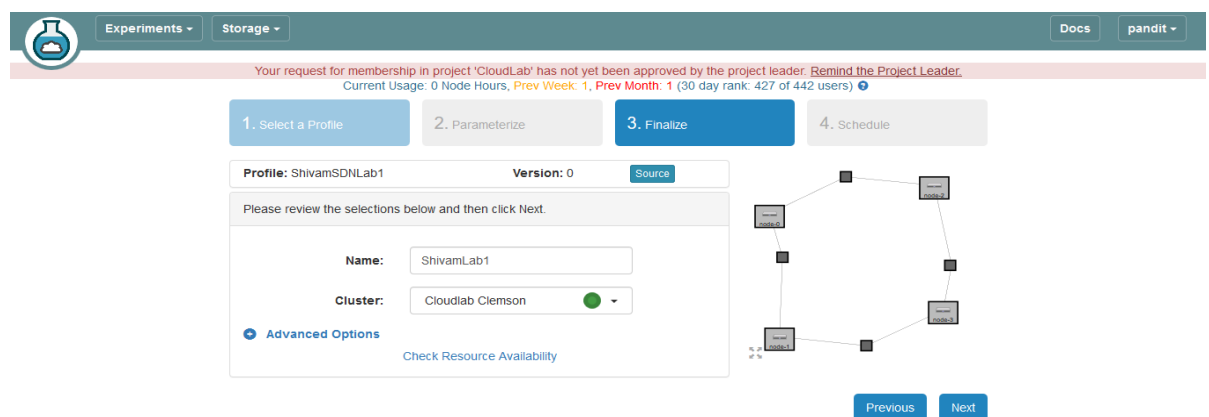
- Figure 2: We make 4 Xen VM nodes, select link type as ethernet and connect them as shown above in Topology editor above



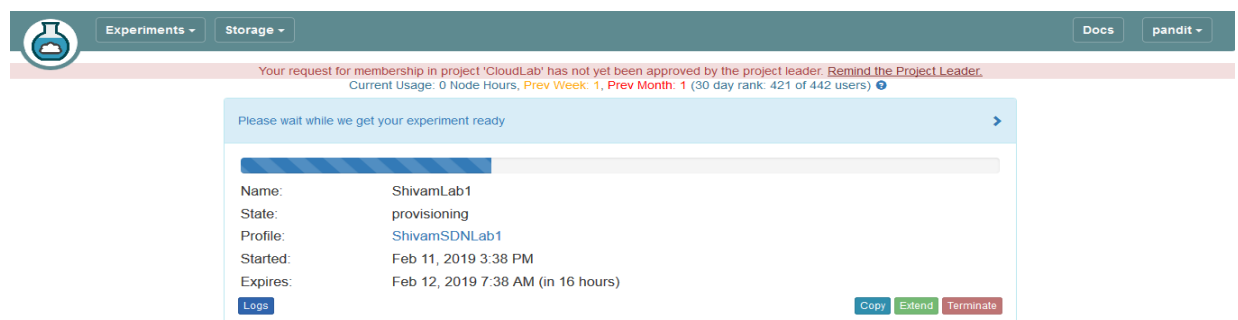
- Figure 3: We provide details for each hardware node with hardware type as any, disk image as Ubuntu 16OS and Require routable IP checked

2. Save and start experiment

After creating topology, we start experiment using cluster available and wait for profile to bootup.



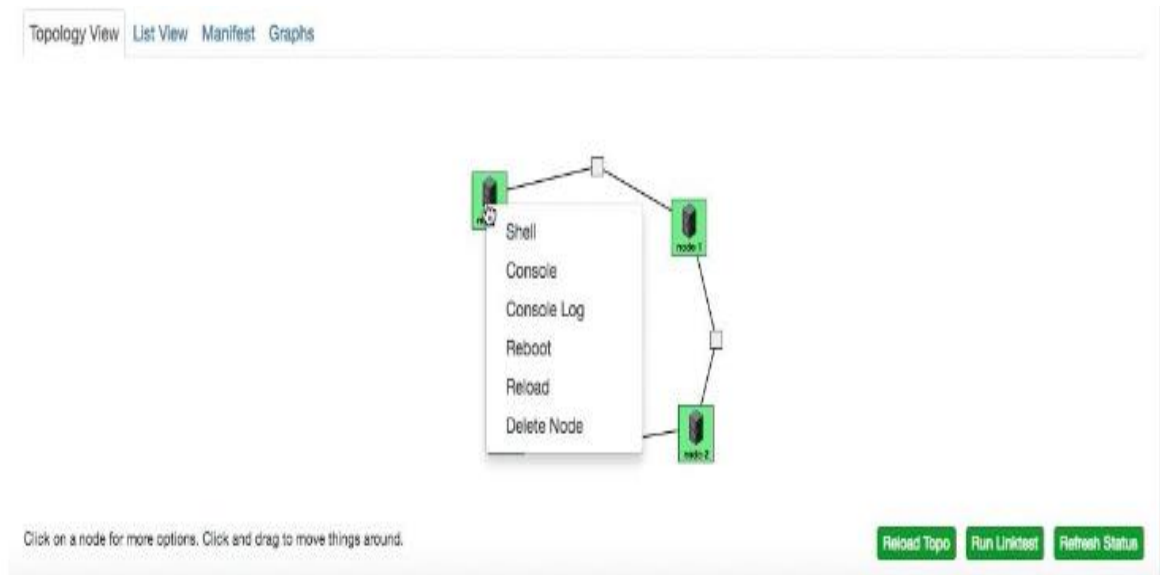
- Figure 4: Start Experiment



- Figure 5: Instantiate Experiment

3. Test Ping connectivity

After instantiating experiment, we check ping connectivity using ping command. We use ping command at shell of all 4 nodes. We use ifconfig to get ip address of each node and then ping the node using respective ip addresses and observe the traffic in the shell. We can see the individual packet headers with important features like source, ttl, time, packet_size & icmp_sequence.



- Figure 6: Opening shell for each node to do ping test

```
Topology View List View Manifest Graphs node-0 X node-1 X node-2 X node-3 X
collisions:0 txqueuelen:1
RX bytes:1560 (1.5 KB) TX bytes:1560 (1.5 KB)

pandit@node-0:~$ ping 10.10.1.1
PING 10.10.1.1 (10.10.1.1) 56(84) bytes of data.
64 bytes from 10.10.1.1: icmp_seq=1 ttl=64 time=0.463 ms
64 bytes from 10.10.1.1: icmp_seq=2 ttl=64 time=0.191 ms
64 bytes from 10.10.1.1: icmp_seq=3 ttl=64 time=0.195 ms
64 bytes from 10.10.1.1: icmp_seq=4 ttl=64 time=0.242 ms
64 bytes from 10.10.1.1: icmp_seq=5 ttl=64 time=0.198 ms
64 bytes from 10.10.1.1: icmp_seq=6 ttl=64 time=0.193 ms
64 bytes from 10.10.1.1: icmp_seq=7 ttl=64 time=0.265 ms
64 bytes from 10.10.1.1: icmp_seq=8 ttl=64 time=0.206 ms
64 bytes from 10.10.1.1: icmp_seq=9 ttl=64 time=0.204 ms
64 bytes from 10.10.1.1: icmp_seq=10 ttl=64 time=0.207 ms
64 bytes from 10.10.1.1: icmp_seq=11 ttl=64 time=0.315 ms
64 bytes from 10.10.1.1: icmp_seq=12 ttl=64 time=0.252 ms
64 bytes from 10.10.1.1: icmp_seq=13 ttl=64 time=0.239 ms
64 bytes from 10.10.1.1: icmp_seq=14 ttl=64 time=0.191 ms
64 bytes from 10.10.1.1: icmp_seq=15 ttl=64 time=0.331 ms
```

- Figure 7: Ping test at node0

```
Topology View List View Manifest Graphs node-0 x node-1 x node-2 x node-3 x
collisions:0 txqueuelen:1000
RX bytes:2252 (2.2 KB) TX bytes:1610 (1.6 KB)

lo
  Link encap:Local Loopback
  inet addr:127.0.0.1 Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING MTU:65536 Metric:1
  RX packets:24 errors:0 dropped:0 overruns:0 frame:0
  TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1
  RX bytes:1560 (1.5 KB) TX bytes:1560 (1.5 KB)

pandit@node-1:~$ ping 10.10.3.1
PING 10.10.3.1 (10.10.3.1) 56(84) bytes of data.
64 bytes from 10.10.3.1: icmp_seq=1 ttl=63 time=0.720 ms
64 bytes from 10.10.3.1: icmp_seq=2 ttl=63 time=0.492 ms
64 bytes from 10.10.3.1: icmp_seq=3 ttl=63 time=0.553 ms
64 bytes from 10.10.3.1: icmp_seq=4 ttl=63 time=0.409 ms
64 bytes from 10.10.3.1: icmp_seq=5 ttl=63 time=0.409 ms
```

○ Figure 8: Ping test at node1

```
Topology View List View Manifest Graphs node-0 x node-1 x node-2 x node-3 x
eth2
  Link encap:Ethernet HWaddr 02:50:56:37:87:27
  inet addr:10.10.3.1 Bcast:10.10.3.255 Mask:255.255.255.0
  inet6 addr: fe80::96:90ff:fe57:a727/64 Scope:Link
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
  RX packets:38 errors:0 dropped:0 overruns:0 frame:0
  TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:2657 (2.6 KB) TX bytes:1568 (1.5 KB)

lo
  Link encap:Local Loopback
  inet addr:127.0.0.1 Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING MTU:65536 Metric:1
  RX packets:24 errors:0 dropped:0 overruns:0 frame:0
  TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1
  RX bytes:1560 (1.5 KB) TX bytes:1560 (1.5 KB)

pandit@node-2:~$ ping 10.10.4.1
PING 10.10.4.1 (10.10.4.1) 56(84) bytes of data.
64 bytes from 10.10.4.1: icmp_seq=1 ttl=63 time=0.520 ms
```

○ Figure 9: Ping test at node2

```
Topology View List View Manifest Graphs node-0 x node-1 x node-2 x node-3 x
eth2
  Link encap:Ethernet HWaddr 02:50:56:37:87:27
  inet addr:10.10.4.2 Bcast:10.10.4.255 Mask:255.255.255.0
  inet6 addr: fe80::5c:b6ff:fe88:b35c/64 Scope:Link
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
  RX packets:72 errors:0 dropped:0 overruns:0 frame:0
  TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:3968 (3.9 KB) TX bytes:3339 (3.3 KB)

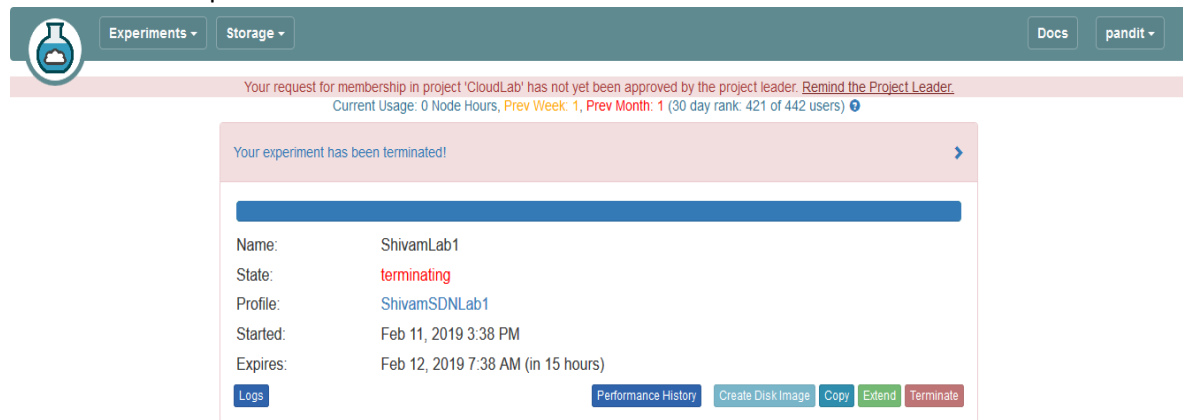
lo
  Link encap:Local Loopback
  inet addr:127.0.0.1 Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING MTU:65536 Metric:1
  RX packets:24 errors:0 dropped:0 overruns:0 frame:0
  TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1
  RX bytes:1560 (1.5 KB) TX bytes:1560 (1.5 KB)

pandit@node-3:~$ ping 10.10.2.2
PING 10.10.2.2 (10.10.2.2) 56(84) bytes of data.
64 bytes from 10.10.2.2: icmp_seq=1 ttl=64 time=0.224 ms
64 bytes from 10.10.2.2: icmp_seq=2 ttl=64 time=0.325 ms
```

○ Figure 10: Ping test at node3

4. Terminating Experiment

After creating experiment and testing ping connectivity, once we are done with experiment, we terminate the experiment as shown below.

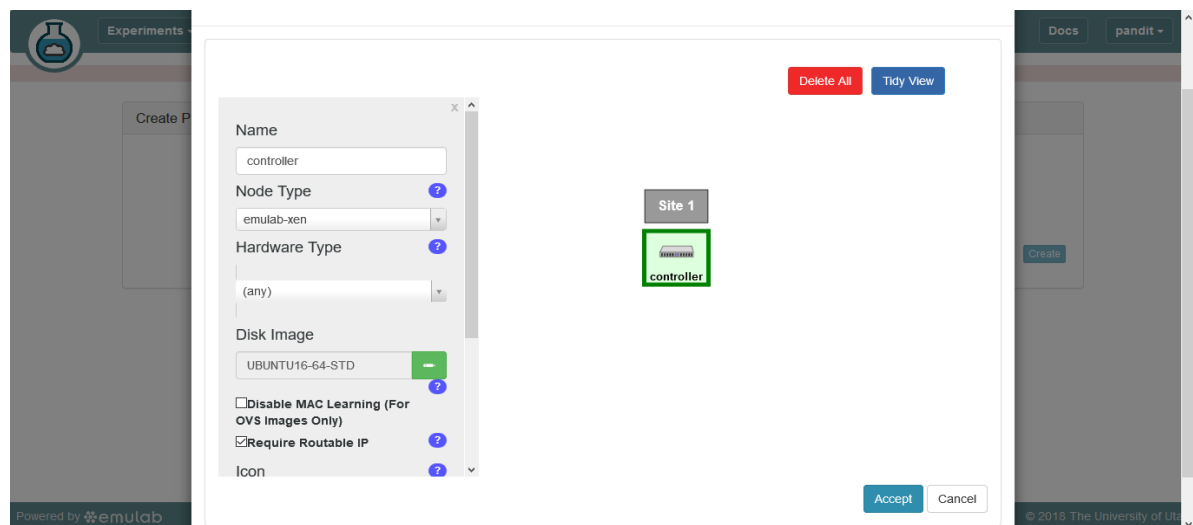


○ Figure 11: Terminating experiment

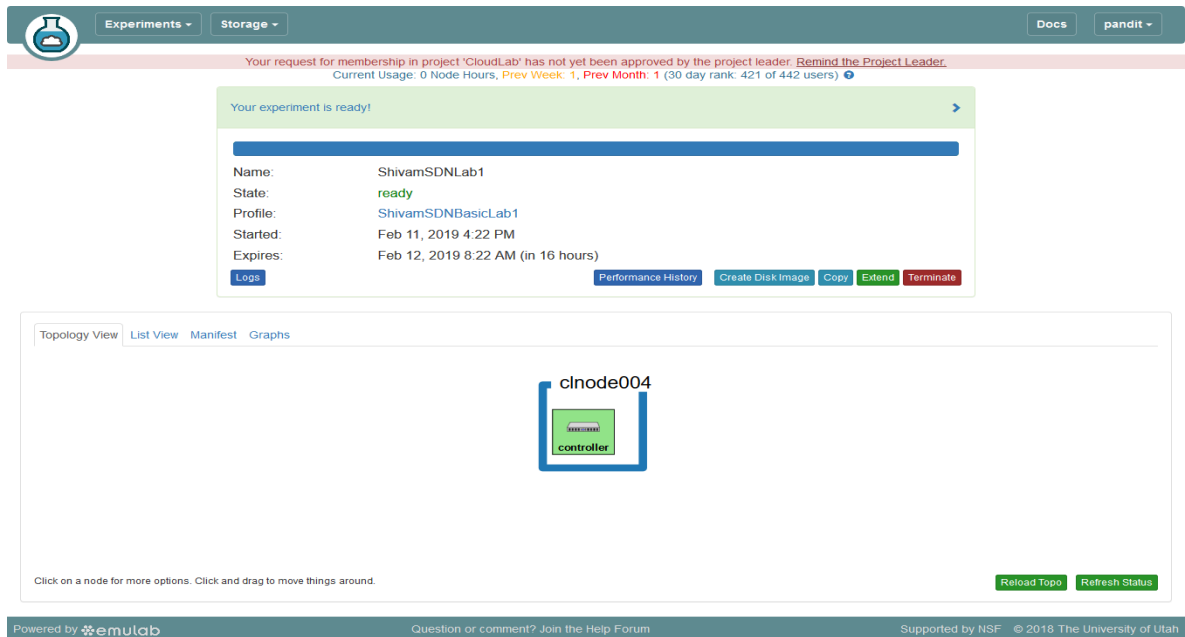
■ Now, we will create Open Flow Topology with SDN Controller

5. Creating Profile for SDN Controller

We first create a profile with single node named controller, operating system as Ubuntu 16 OS, Hardware type is set as any and node type as emulab-xen. We select a available cluster and click finish.



○ Figure 12: Creating single node for SDN controller with checked require routable ip so that all nodes can access controller



○ Figure 13: creating experiment and checking the topology view for controller

```

Topology View List View Manifest Graphs controller X
pandit@controller:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:44:39:53:3a:c5
          inet addr:130.127.132.234  Bcast:130.127.135.255  Mask:255.255.252.0
          inet6 addr: fe80::44:39ff:fe53:3ac5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18686 errors:0 dropped:65 overruns:0 frame:0
          TX packets:457 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:917158 (917.1 KB)  TX bytes:50737 (50.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1560 (1.5 KB)  TX bytes:1560 (1.5 KB)

pandit@controller:~$ 

```

○ Figure 14: Used ifconfig to get ip address of the controller

6. Installing Floodlight

We first open shell at controller node and note controller ip address using ifconfig command. Then using sequence of commands, we install floodlight as shown below.

```
Get sudo user privileges: "sudo su"
-Update APT repo: "apt-get update"
-Install java 8: "apt-get install default-jdk" and "apt-get install default-jre". This will
install java 8 on Ubuntu16.
-Install dependencies: "apt-get install build-essential ant maven python-dev"-
```

➤ Install Floodlight:

```
•git clone git://github.com/floodlight/floodlight.git-b v1.2
•cd floodlight
•git submodule init
•git submodule update
•ant
•sudo mkdir /var/lib/floodlight
•sudo chmod 777 /var/lib/floodlight
```

-Start the controller: "java -jar target/floodlight.jar"

The screenshot displays the CloudLab web interface. At the top, there's a navigation bar with 'Experiments' and 'Storage' tabs, and buttons for 'Docs' and 'pandit'. A message states: 'Your request for membership in project 'CloudLab' has not yet been approved by the project leader. [Remind the Project Leader.](#) Current Usage: 0 Node Hours, [Prev Week: 1](#), [Prev Month: 1](#) (30 day rank: 421 of 442 users)'. Below this, a green box indicates 'Your experiment is ready!'. The experiment details are as follows:

Name:	ShivamSDNLab1
State:	ready
Profile:	ShivamSDNBasicLab1
Started:	Feb 11, 2019 4:22 PM
Expires:	Feb 12, 2019 8:22 AM (in 16 hours)

Below the details are buttons for 'Logs', 'Performance History', 'Create Disk Image', 'Copy', 'Extend', and 'Terminate'. The main part of the interface shows a terminal window titled 'controller x controller x' with the following output:

```
Topology View List View Manifest Graphs controller x controller x
Setting up openjdk-8-jdk:amd64 (8u191-b12-2ubuntu0.16.04.1) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/appletviewer to provide /usr/bin/appletviewer (appletviewer) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Setting up default-jdk (2:1.8-56ubuntu2) ...
Processing triggers for libc-bin (2.23-0ubuntu10) ...
Processing triggers for ca-certificates (20170717~16.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
root@controller:/users/pandit# apt-get install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jre is already the newest version (2:1.8-56ubuntu2).
default-jre set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 146 not upgraded.
root@controller:/users/pandit# apt-get install build-essential ant mavenpython-dev
```

At the bottom of the interface, there's a footer with 'Powered by emulab', 'Question or comment? Join the Help Forum', and 'Supported by NSF © 2018 The University of Utah'.

○ Figure 15: Installing Java, JRE and dependencies for Floodlight controller

7. Starting Floodlight

After installing floodlight, we start controller using command:

```
"java -jar target/floodlight.jar"
```

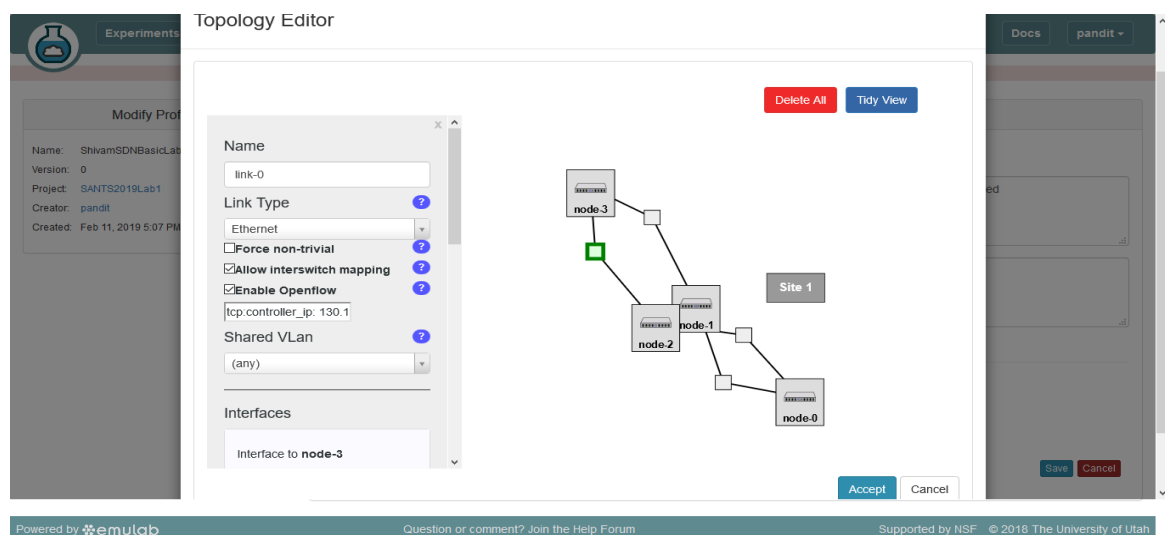


```
Topology View List View Manifest Graphs controller X controller X
Compile test.
[javac] Compiling 91 source files to /users/pandit/floodlight/target/bin-test
[javac] warning: [options] bootstrap class path not set in conjunction with -source 1.7
[javac] 1 warning
dist:
[echo] Setting Floodlight version: 1.2
[echo] Setting Floodlight name: floodlight
[jar] Building jar: /users/pandit/floodlight/target/floodlight.jar
[jar] Building jar: /users/pandit/floodlight/target/floodlight-test.jar
BUILD SUCCESSFUL
Total time: 48 seconds
root@controller:/users/pandit/floodlight# sudo mkdir /var/lib/floodlight
root@controller:/users/pandit/floodlight# sudo chmod 777 /var/lib/floodlight
root@controller:/users/pandit/floodlight# java -jar target/floodlight.jar
14:48:57.229 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from src/main/resources/floodlightdefault.properties
14:48:57.833 WARN [n.f.r.RestApiServer:main] HTTPS disabled; HTTPS will not be used to connect to the REST API.
14:48:57.834 WARN [n.f.r.RestApiServer:main] HTTP enabled; Allowing unsecure access to REST API on port 8080.
14:49:04.234 WARN [n.f.c.i.OFSwitchManager:main] SSL disabled. Using unsecure connections between Floodlight and switches.
14:49:04.234 INFO [n.f.c.i.OFSwitchManager:main] Clear switch flow tables on initial handshake as master: TRUE
```

○ Figure 16: Starting Floodlight Controller

8. Setting Profile for Experiment

After installing and starting floodlight, we create new profile for experiment with 4 Xen VM nodes, hardware type as any, node type as emulab-xen, link-type as Ethernet. Also, selecting open flow for the links with controller ip address.



○ Figure 17: Connecting nodes with open flow configuration and required routable ip checked

9. Starting Experiments

After making topology for experiment, select available cluster and instantiate experiment.

The screenshot shows the CloudLab Experiments interface. At the top, there's a navigation bar with 'Experiments' and 'Storage' tabs, and a 'pandit' dropdown. Below the navigation bar, a message states: 'Your request for membership in project 'CloudLab' has not yet been approved by the project leader. [Remind the Project Leader.](#)' followed by usage statistics: 'Current Usage: 0 Node Hours, Prev Week: 1, Prev Month: 1 (30 day rank: 421 of 442 users)'. The main workflow consists of four steps: 1. Select a Profile, 2. Parameterize, 3. Finalize (highlighted in blue), and 4. Schedule. Below the steps, the 'Profile' is 'ShivamSDNBasicLab1' and the 'Version' is '0'. A 'Source' button is next to the version. A message says 'Please review the selections below and then click Next.' The 'Name' field is 'ShivamSDNLab1'. The 'Cluster' dropdown is set to 'Cloudlab Clemson' with a green status indicator. There are links for 'Advanced Options' and 'Check Resource Availability'. At the bottom right, there are 'Previous' and 'Next' buttons.

○ Figure 18: Starting experiment

10. Install OpenVSwitch and Setup bridges on all nodes

All links in our topology are connected to SDN controller and to check that we will setup a bridge on all nodes and connect to floodlight controller so that it learns and sends appropriate flow rules.

Commands to install OpenVSwitch on all nodes to set bridge.

Install OpenVSwitch:

- "sudo apt-get update"
- "sudo apt-get install openvswitch-switch"

```
Topology View List View Manifest Graphs node-3 x
Get:14 http://us.archive.ubuntu.com/ubuntu xenial-updates/main i386 Packages [800 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu xenial-updates/main Translation-en [368 kB]
Get:16 http://us.archive.ubuntu.com/ubuntu xenial-updates/restricted amd64 Packages [7,556 B]
Get:17 http://us.archive.ubuntu.com/ubuntu xenial-updates/restricted i386 Packages [7,524 B]
Get:18 http://us.archive.ubuntu.com/ubuntu xenial-updates/restricted Translation-en [2,272 B]
Get:19 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [727 kB]
Get:20 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [613 kB]
Get:21 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe i386 Packages [666 kB]
Get:22 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe Translation-en [300 kB]
Get:23 http://security.ubuntu.com/ubuntu xenial-security/main i386 Packages [516 kB]
Get:24 http://security.ubuntu.com/ubuntu xenial-security/main Translation-en [253 kB]
Get:25 http://security.ubuntu.com/ubuntu xenial-security/restricted amd64 Packages [7,204 B]
Get:26 http://security.ubuntu.com/ubuntu xenial-security/restricted i386 Packages [7,224 B]
Get:27 http://security.ubuntu.com/ubuntu xenial-security/restricted Translation-en [2,152 B]
Get:28 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 Packages [424 kB]
Get:29 http://security.ubuntu.com/ubuntu xenial-security/universe i386 Packages [369 kB]
Get:30 http://security.ubuntu.com/ubuntu xenial-security/universe Translation-en [169 kB]
Fetched 29.0 MB in 5s (4,902 kB/s)
Reading package lists... Done
pandit@node-3:~$ sudo apt-get install openvswitch-switch
```

○ Figure 19: Installing OpenVSwitch on all nodes

- o Using these commands, we setup bridge on each node and connect to controller

```

sudo su
ovs-vsctl add-br <bridge_name>
ovs-vsctl add-port <bridge_name> eth1
ovs-vsctl add-port <bridge_name> eth2
ifconfig eth1 0ifconfig eth2 0
ovs-vsctl set-controller <bridge_name>tcp:<controller_IP_Address>:6653
ifconfig <bridge_name> 10.10.10.1 netmask 255.255.255.0 up

```

```

Topology View List View Manifest Graphs node-0 node-1 node-2 node-3
Processing triggers for systemd (229-4ubuntu21.15) ...
Setting up openvswitch-common (2.5.5-0ubuntu0.16.04.2) ...
Setting up openvswitch-switch (2.5.5-0ubuntu0.16.04.2) ...
update-alternatives: using /usr/lib/openvswitch-switch/ovs-vswitchd to provide /usr/sbin/ovs-vswitchd (ovs-vswitchd) in auto mode
inserv: can not symlink(/init.d/pubsubd, ../rc1.d/K01pubsubd): File exists
inserv: can not symlink(/init.d/pubsubd, ../rc2.d/S01pubsubd): File exists
inserv: can not symlink(/init.d/pubsubd, ../rc3.d/S01pubsubd): File exists
inserv: can not symlink(/init.d/pubsubd, ../rc6.d/K01pubsubd): File exists
openvswitch-nonetwork.service is a disabled or a static unit, not starting it.
Processing triggers for systemd (229-4ubuntu21.15) ...
Processing triggers for ureadahead (0.100.0-19) ...
pandit@node-2:~$ sudo su
root@node-2:/users/pandit# ovs-vsctl add-br ovs-lan2
root@node-2:/users/pandit# ovs-vsctl add-port ovs-lan2 eth1
root@node-2:/users/pandit# ovs-vsctl add-port ovs-lan2 eth2
root@node-2:/users/pandit# ifconfig eth1 0
root@node-2:/users/pandit# ifconfig eth2 0
root@node-2:/users/pandit# ovs-vsctl set-controller ovs-lan2 tcp:128.110.99.138:6653
root@node-2:/users/pandit# ifconfig ovs-lan2 10.10.10.2 netmask 255.255.255.0 up
root@node-2:/users/pandit#

```

- o Figure 20: Setup Bridge on each node and connect to controller

11. Ping and dump Flows

After completing configurations at nodes, ping from one node to other will work. We can use `tcpdump -I eth1` on node-1 and node-2 to check which path the ping takes and also administer network packet as shown below.

```

Topology View List View Manifest Graphs node-2 node-1
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Feb 12 00:19:05 2019 from 155.98.33.74
pandit@node-1:~$ sudo su
root@node-1:/users/pandit# tcpdump -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
00:21:36.769749 LLDP, length 61
00:21:36.799904 02:53:26:be:c8:de (oui Unknown) > Broadcast, ethertype Unknown (0x8942), length 83:
    0x0000:  2000 0604 0002 0000 0207 0456 4dc9 90f7  ....VM...
    0x0010:  4c04 0302 0001 0602 0078 fe0c 0026 e100  L.....X...&..
    0x0020:  0000 564d cb90 f74c 1808 0287 e327 9b36  ..VM...L.....'.6
    0x0030:  ef3d e601 01fe 0c00 26e1 0100 0001 68e0  .=.....&.....h.
    0x0040:  2745 d100 00                                'E...

```

- o Figure 21: Using tcpdump to check flows between nodes

➤ **Checking Flow:**

Check Flow Rules of all 4 nodes using `ovs-ofctl dump-flows <bridge_name> -O OpenFlow13` as shown below.

```
Topology View List View Manifest Graphs node-1 x node-2 x node-3 x node-0 x
Reading package lists... Done
Building dependency tree
Reading state information... Done
openvswitch-switch is already the newest version (2.5.5-0ubuntu0.16.04.2).
0 upgraded, 0 newly installed, 0 to remove and 146 not upgraded.
root@node-0:/users/pandit# ovs-vsctl add-br ovs-lan0
root@node-0:/users/pandit# ovs-vsctl add-port ovs-lan0 eth1
root@node-0:/users/pandit# ovs-vsctl add-port ovs-lan0 eth2
ovs-vsctl add-port: command not found
root@node-0:/users/pandit# ifconfig eth1 0ifconfig eth1 0
0ifconfig: Unknown host
ifconfig: '--help' gives usage information.
root@node-0:/users/pandit# ifconfig eth1 0
root@node-0:/users/pandit# ifconfig eth2 0
root@node-0:/users/pandit# ovs-vsctl set-controller ovs-lan0 tcp:130.127.132.234:6653
root@node-0:/users/pandit# ifconfig ovs-lan0 10.10.10.0 netmask 255.255.255.0 up
root@node-0:/users/pandit#
OFPT_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=258.144s, table=0, n_packets=10, n_bytes=778, priority=0 actions=CONTROLLER:65535
root@node-0:/users/pandit#
```

- Figure 22: using `ovs-ofctl dump-flows ovs-lan0 -O OpenFlow13` for node0 to check flow entry by floodlight controller for node 0

```
Topology View List View Manifest Graphs node-1 x node-3 x node-0 x node-2 x
inserv: can not symlink(/init.d/pubsubd, ../rc2.d/S01pubsubd): File exists
inserv: can not symlink(/init.d/pubsubd, ../rc3.d/S01pubsubd): File exists
inserv: can not symlink(/init.d/pubsubd, ../rc6.d/K01pubsubd): File exists
openvswitch-nonetwork.service is a disabled or a static unit, not starting it.
Processing triggers for systemd (229-4ubuntu21.15) ...
Processing triggers for ureadahead (0.100.0-19) ...
root@node-1:/users/pandit# ovs-vsctl add-br ovs-lan1
root@node-1:/users/pandit# ovs-vsctl add-port ovs-lan1 eth1
root@node-1:/users/pandit# ovs-vsctl add-port ovs-lan1 eth2
ovs-vsctl add-port: command not found
root@node-1:/users/pandit# ovs-vsctl add-port ovs-lan1 eth2
root@node-1:/users/pandit# ifconfig eth1 0
root@node-1:/users/pandit# ifconfig eth2 0
root@node-1:/users/pandit# ovs-vsctl set-controller ovs-lan1 tcp:130.127.132.234:6653
root@node-1:/users/pandit# ifconfig ovs-lan1 10.10.10.1 netmask 255.255.255.0 up
root@node-1:/users/pandit# ovs-ofctl dump-flows ovs-lan1 -O OpenFlow13
OFPT_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=1304.243s, table=0, n_packets=53, n_bytes=4217, priority=0 actions=CONTROLLER:65535
root@node-1:/users/pandit#
```

- Figure 23: using `ovs-ofctl dump-flows ovs-lan1 -O OpenFlow13` for node1 for node1 to check flow entry by floodlight controller for node 1

```
Topology View List View Manifest Graphs node-1 x node-2 x node-3 x node-0 x node-2 x
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-142-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Feb 11 23:49:33 2019 from 155.98.33.74
pandit@node-2:~$ sudo su
root@node-2:/users/pandit# ovs-ofctl dump-flows ovs-lan2 -O OpenFlow13
OFPT_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=1003.483s, table=0, n_packets=34, n_bytes=2692, priority=0 actions=CONTROLLER:65535
root@node-2:/users/pandit#
```

- Figure 24: using `ovs-ofctl dump-flows ovs-lan2 -O OpenFlow13` for node2 to check flow entry by floodlight controller for node 2

```

Topology View List View Manifest Graphs node-1 x node-2 x node-3 x node-0 x
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:24 errors:0 dropped:0 overruns:0 frame:0
TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:1560 (1.5 KB) TX bytes:1560 (1.5 KB)

ovs-lan3 Link encap:Ethernet HWaddr 6a:af:c3:5c:25:45
inet addr:10.10.10.3 Bcast:10.10.10.255 Mask:255.255.255.0
inet6 addr: fe80::68af:c3ff:fe5c:2545/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3 errors:0 dropped:6 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:168 (168.0 B) TX bytes:648 (648.0 B)

root@node-3:/users/pandit# ovs-ofctl dump-flows ovs-lan3 -O OpenFlow13
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=491.748s, table=0, n_packets=37, n_bytes=3006, priority=0 actions=CONTROLLER:65535
root@node-3:/users/pandit#
root@node-3:/users/pandit#

```

- Figure 25: using `ovs-ofctl dump-flows ovs-lan3 -O OpenFlow13` for node3 to check flow entry by floodlight controller for node 3

- ✓ The `tcpdump -I eth1` command is used to output the packet contents flowing in the network. It is a packet analyzer that is running under command line. It captures packets on network and display headers of the packet. It is used for solving problems in network and for monitoring network activities.

- ✓ The flow rules in step 6 using `ovs-ofctl` command is used for monitoring and administering OpenFlow switches. It shows the flow table that is updated by the controller and keeps track of many features of open flow switches, entries and features. This command works with any open flow switch. We have in our experiment given bridges names as `ovs-lan0`, `ovs-lan1`, `ovs-lan2` and `ovs-lan3` for 4 Xen VM nodes running Ubuntu 16OS. So, using this command with the bridge name tells the controller to keep track of the flow in the form of flow rules and flow table.