UNIVERSITY OF
SOUTH CAROLINA

SOFTWARE DEFINED NETWORKING

Lab 4: Introduction to SDN

Document Version: 05-28-2020

Award 1829698
"CyberTraining CIP: Cyberinfrastructure Expertise on High-throughput Networks for Big Science Data Transfers"

Contents

Overview

This lab is an introduction to Software Defined Networking (SDN), a new networking paradigm that overcomes several limitations of the current network infrastructure. In this lab, we will introduce the components of SDN and describe how they operate. The focus in this lab is to gain in depth knowledge about the role of the control plane and the data plane.

Objectives

By the end of this lab, the user should be able to:

1. Understand the concept of SDN.
2. Enable ONOS controller.
3. Navigate through ONOS environment.
4. Activate basic ONOS applications and understand their effects.
5. Understand flow tables in SDN devices.
6. Perform a connectivity test.
7. Visualize topology information in the GUI dashboard.

Lab settings

The information in Table 1 provides the credentials to access the Client's virtual machine.

Table 1**.** Credentials to access Client's virtual machine.

Device	Account	Password
Client	admin	password

Lab roadmap

This lab is organized as follows:

1. Section 1: Introduction.
2. Section 2: Lab topology.
3. Section 3: Starting ONOS
4. Section 4: Navigating ONOS GUI.

1 Introduction

The Software Defined Networking (SDN) posed a significant interest from academia and industry just in few years. The concept of an open, vendor neutral, control-data plane interface that enables network hardware and software to be managed and designed independently, facilitates the migration from expensive, proprietary hardware and firmware to a free, open source Network Operating System (NOS).

A key feature of SDN consists of managing the network resources and providing the applications for interacting, managing, monitoring and programming network switches. The NOS is responsible to provide an open platform that simplifies the creation of innovative and beneficial network applications and services that is compatible with a wide range of hardware[2].

1.1 Introduction to SDN

Traditional IP networks depend on distributed routing protocols running inside the routers to exchange routing information. Despite their widespread adoption, traditional IP networks are complicated by their nature, and they are not trivial to be managed. For example, network administrators need to configure each network separately using low-level, vendor-specific commands. A traditional networking device bundles the control plane (that decides how to handle network traffic) and the data plane (that forward network traffic). Thus, reducing the flexibility and hindering innovation and evolution of the networking infrastructure[3].

SDN is an emerging networking paradigm that gives hope to change the limitations of current network infrastructures[3]. It breaks the control plane from the data plane and implements each separately. The disaggregation of the control plane and the data plane allows network switches to become simple forwarding devices and the control logic to be implemented in a logically centralized controller. Thus, amplifying the flexibility in the network, breaking the network control problem into tractable pieces, introducing new abstractions, and facilitating network evolution and innovation[3].

Consider Figure 1. The *data plane* is responsible of managing the major part of the packets handled by the switch. The ports in the data plane manage the reception and transmission of packets and a forwarding table is responsible to set the logic. The data plane handles operations such as packet buffering, packet scheduling, header modification, and forwarding. For example, if the header of a data packet contains information that matches the forwarding table, the data plane may perform some header field modification in order to be forwarded offloading the intervention of the other two planes[4].

Some packets cannot be processed by the data plane directly, for example, their information is not yet inserted in the forwarding table. Such packets are forwarded to the control plane, which lies on top of the data plane. The control plane involves many activities, mainly to maintain the forwarding table of the data plane. Essentially, the control plane is responsible for processing different control protocols that may affect the forwarding table.

SDN applications run on top of the controller. They are ultimately responsible for managing the flow table on the network devices (data plane). For example, route packets through the best path between two endpoints, or balance traffic loads across multiple paths[4].

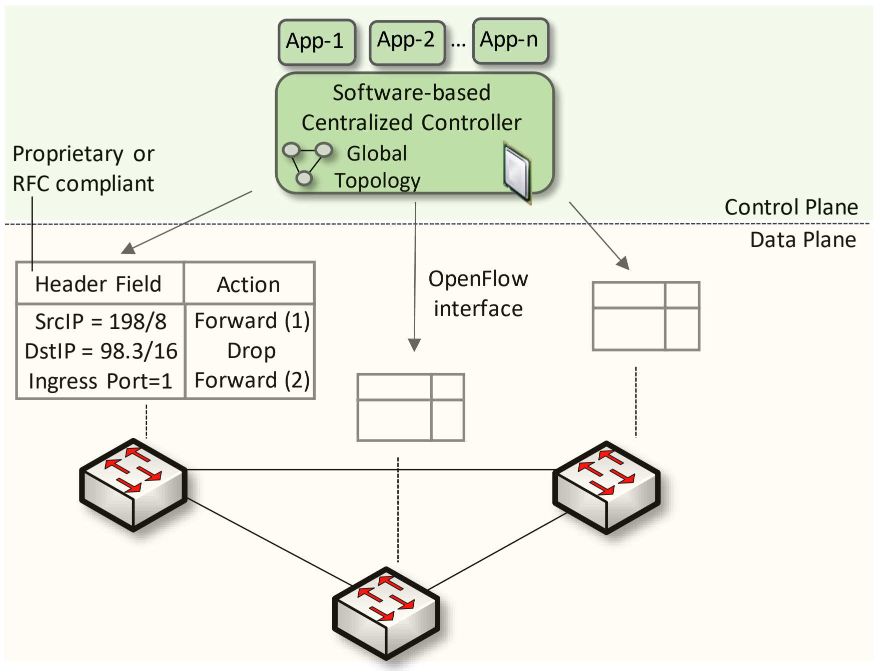


Figure 1. The control plane is embedded in a centralized server and it is decoupled from data plane devices in SDN networks.

1.2 SDN data plane architecture

Consider Figure 2. An SDN device is composed of an Application Programming Interface (API) for communication with the controller, an abstraction layer, and a packet-processing function.

The standard API used to communicate with the controller is OpenFlow[1], or it could be some proprietary alternative in certain SDN solutions[4].

The abstraction layer embodies one or more flow tables, which are the fundamental data structures in an SDN device. These flow tables allow the device to evaluate incoming packets and take the appropriate action. Flow tables consist of a number of flow entries, each of which typically consists of two components: match fields and actions. Match fields are used to compare against incoming packets, for example, matching against the Media Access Control (MAC) address, User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) port. Actions are the instructions that the forwarding device should perform if an incoming packet matches a flow entry. These actions may include forwarding the packet to a specific port, dropping the packet, or flooding the packet on all ports, among others[4].

The packet-processing logic consists of the mechanisms to take actions based on the results of evaluating incoming packets. In a hardware switch, these mechanisms are implemented by specialized hardware, such as Ternary Content Addressable Memory (TCAM) and Content Addressable Memory (CAM)[4].

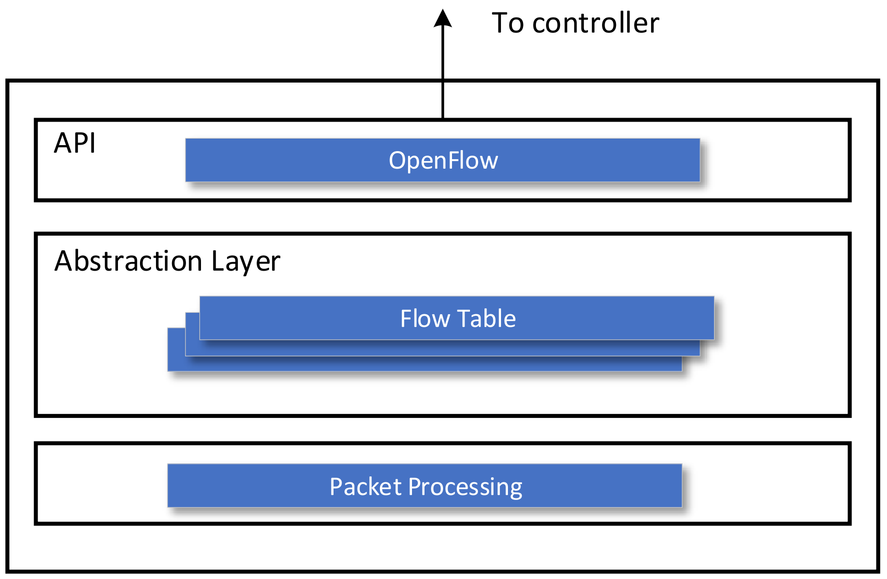


Figure 2. SDN switch architecture.

1.3 SDN controller architecture

A controller actively monitors several variables within the entire network. Additionally, it implements policy decisions, controls all the SDN devices that comprise the network infrastructure and provides a northbound API for the application. Also, controllers implement policy decisions about routing, forwarding, redirecting, and load balancing. Controllers often come with their own set of common application modules, such as a learning switch, a router, a basic firewall, and a simple load balancer. Such features are really SDN applications. However, they are often bundled with the controller[4].

Figure 3 depicts the components of the core functionalities supported by the controller, both a northbound and southbound API, and a few sample applications. The southbound API is used as an interface to communicate SDN device. Frequently, this API is OpenFlow. The controller hides several details of the SDN controller-to-device protocol so that the applications such as the GUI, learning switch, router and others can transparently communicate with the SDN devices. Every controller provides core functionality between these those interfaces. Core features in the controller are[4]:

- End-user device discovery: discovery of end-user devices, such as laptops, desktops, printers, mobile devices, etc.

- Network device discovery: discovery of network devices which comprise the infrastructure of the network, such as switches, routers, and wireless access points.

- Network device topology management: maintain information about the interconnection details of the network devise to each other, and to the end-user devices to which they are directly attached.

- Flow management: maintain a database of the flows being managed by the controller and perform all necessary coordination with the devices to ensure synchronization of the device flow entries with that database.

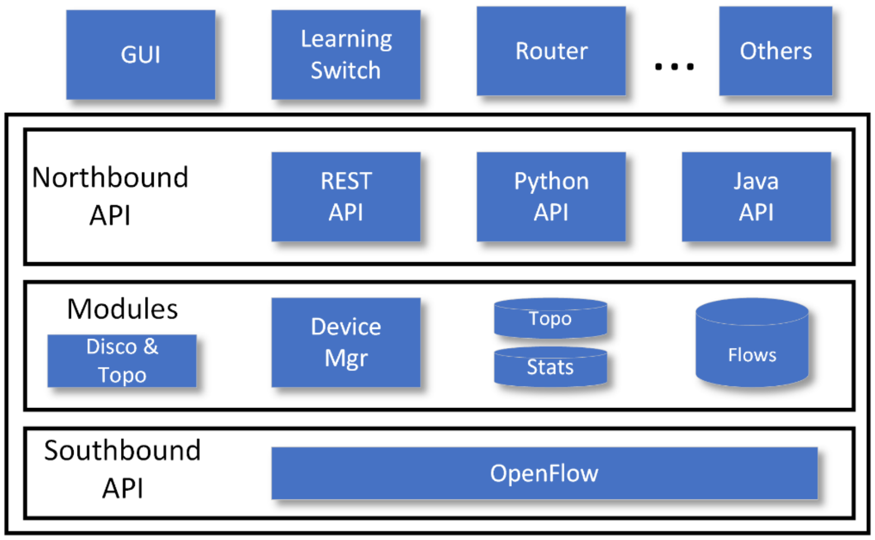


Figure 3. SDN controller architecture.

2 Lab topology

Consider Figure 4. The topology consists of four end-hosts, three switches and a controller. The blue devices represent OpenFlow switches. All switches are connected to the controller c0.

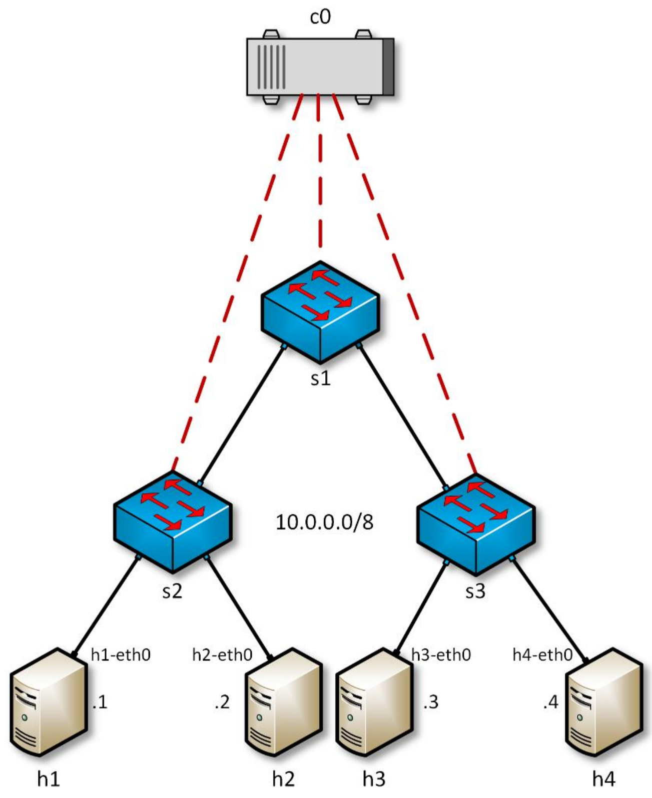


Figure 4. Lab topology.

2.1 Lab settings

The devices are already configured according to Table 2.

Table 2. Topology information.

Device	Interface	IP Address	Subnet
h1	h1-eth0	10.0.0.1	/8
h2	h2-eth0	10.0.0.2	/8
h3	h3-eth0	10.0.0.3	/8
h2	h4-eth0	10.0.0.4	/8
c0	n/a	127.0.0.1	/32

2.2 Loading a topology

In this section, the user will open MiniEdit and load the lab topology. MiniEdit provides a Graphical User Interface (GUI) that facilitates the creation and emulation of network topologies in Mininet[5]. This tool has additional capabilities such as: configuring network elements (i.e IP addresses, default gateway), save the topology and export a layer 2 model.

Step 1. A shortcut to Miniedit is located on the machine's Desktop. Start Miniedit by clicking on Miniedit's shortcut. When prompted for a password, type `password`.

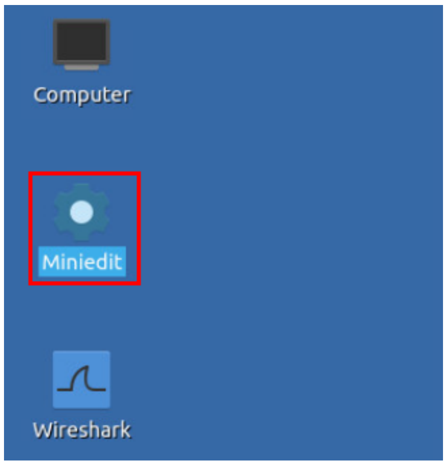
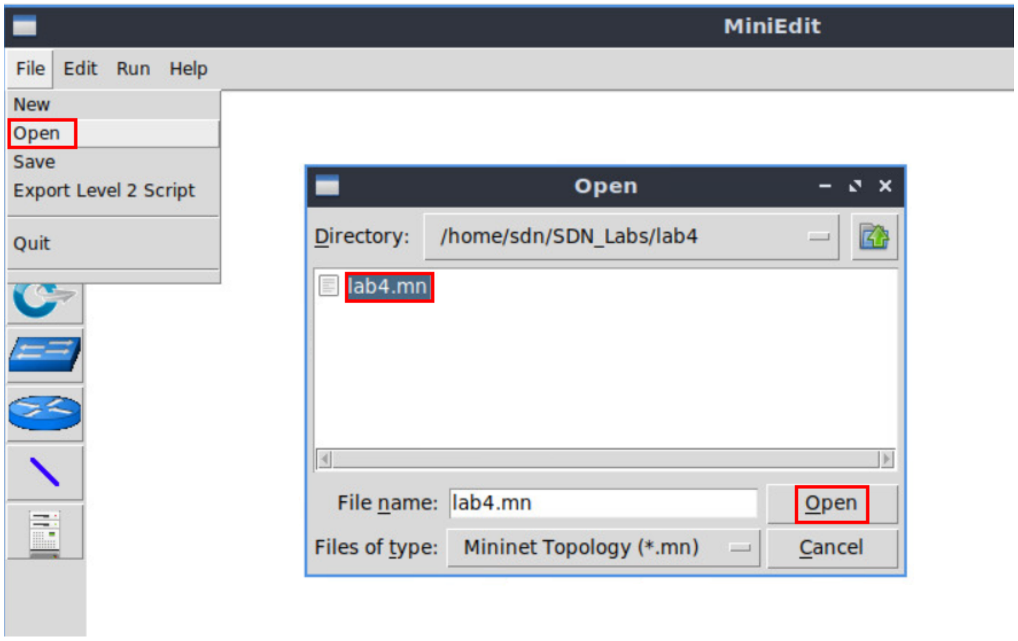


Figure 5. MiniEdit shortcut.

Step 2. On Miniedit's menu bar, click on *File* then *open* to load the lab's topology. Open the *Lab4.mn* topology file stored in the default directory, */home/sdn/SDN_Labs /lab4* and click on *Open*.



Directory: /home/sdn/SDN_Labs/lab4

lab4.mn

File name: lab4.mn

Files of type: Mininet Topology (*.mn)

Figure 6. Opening topology.

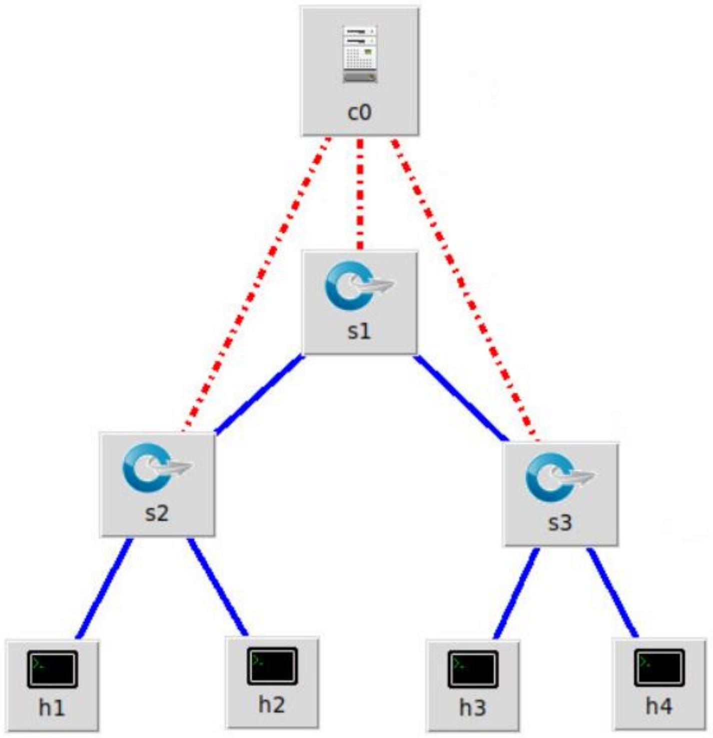


Figure 7. MiniEdit’s topology.

Step 3. Click on the *Run* button to start the emulation. The emulation will start and the buttons of the MiniEdit panel will gray out, indicating that they are currently disabled.

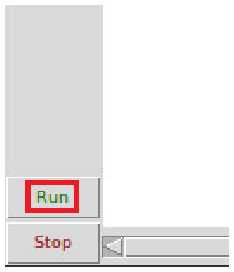


Figure 8. Starting the emulation.

3 Starting ONOS

Step 1. Open Linux terminal by clicking on the shortcut depicted below.

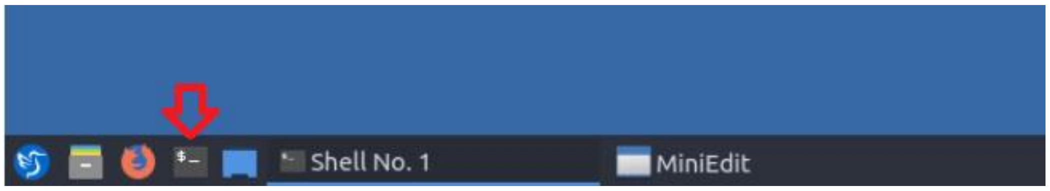
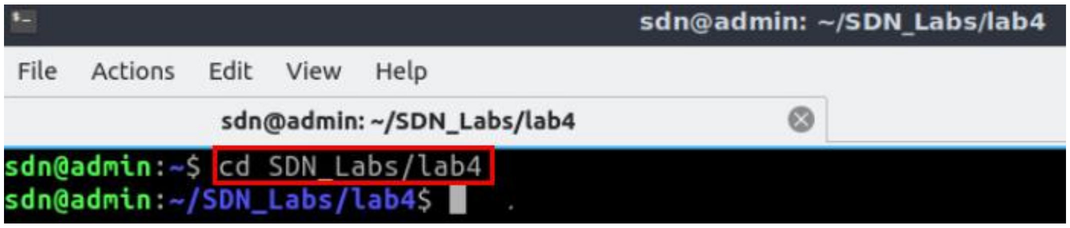


Figure 9. Opening Linux terminal.

Step 2. Navigate into *SDN_Labs/lab4* directory by issuing the following command. This folder contains the script responsible for starting ONOS. The `cd` command is short for change directory followed by an argument that specifies the destination directory.

```
cd SDN_Labs/lab4
```

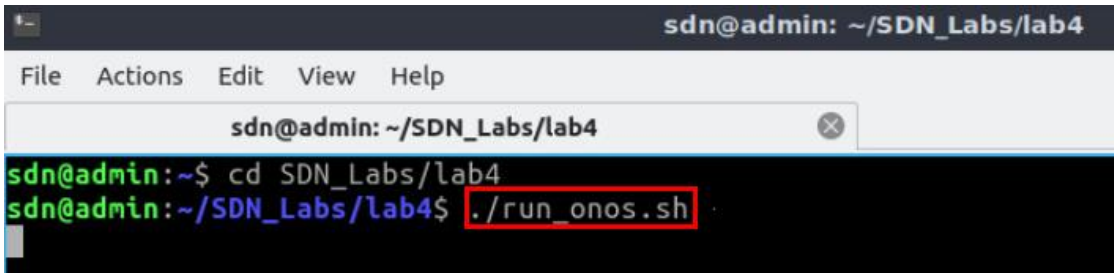

sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@admin:~$ cd SDN_Labs/lab4
sdn@admin:~/SDN_Labs/lab4$
```

Figure 10. Entering the *SDN_Labs/lab4* directory.

Step 3. A script was written to run ONOS and enter its Command Line Interface (CLI). In order to run the script, issue the following command.

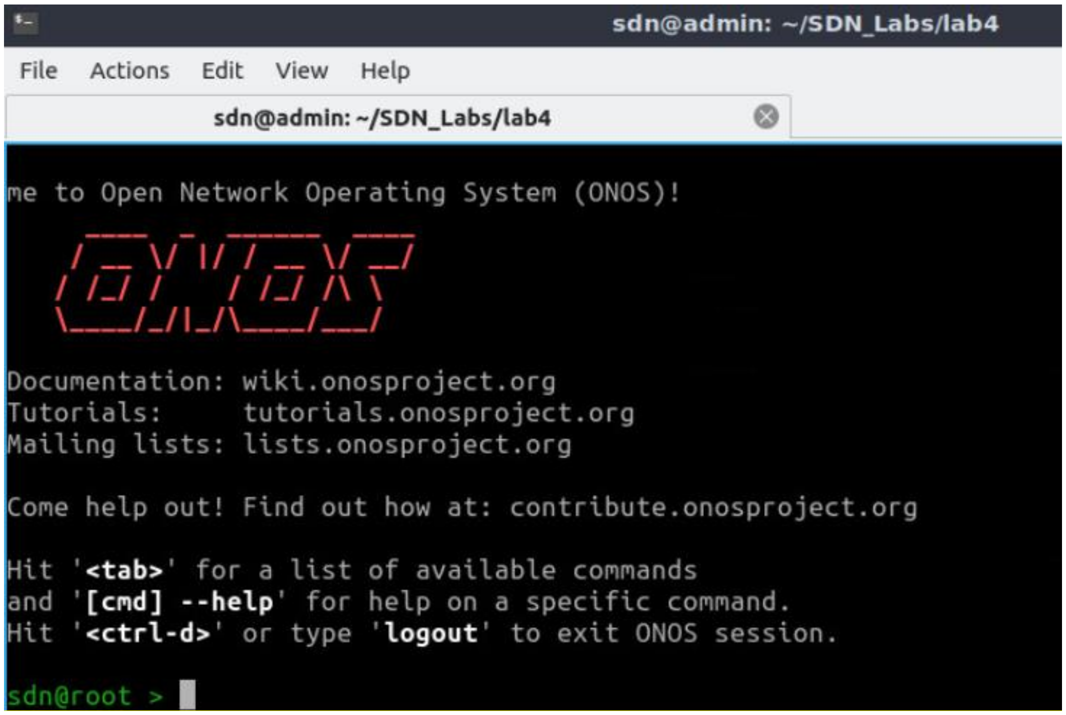
```
./run_onos.sh
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@admin:~$ cd SDN_Labs/lab4
sdn@admin:~/SDN_Labs/lab4$ ./run_onos.sh
```

Figure 11. Starting ONOS.

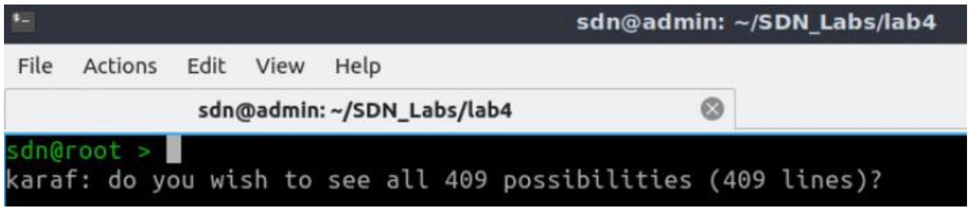
Once the script finishes executing and ONOS is ready, you will be able to execute commands on ONOS CLI as shown in the figure below. Note that this script may take a couple of minutes.


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
me to Open Network Operating System (ONOS)!

Documentation: wiki.onosproject.org
Tutorials:     tutorials.onosproject.org
Mailing lists: lists.onosproject.org

Come help out! Find out how at: contribute.onosproject.org

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'logout' to exit ONOS session.

sdn@root >
```

Figure 12. ONOS CLI.

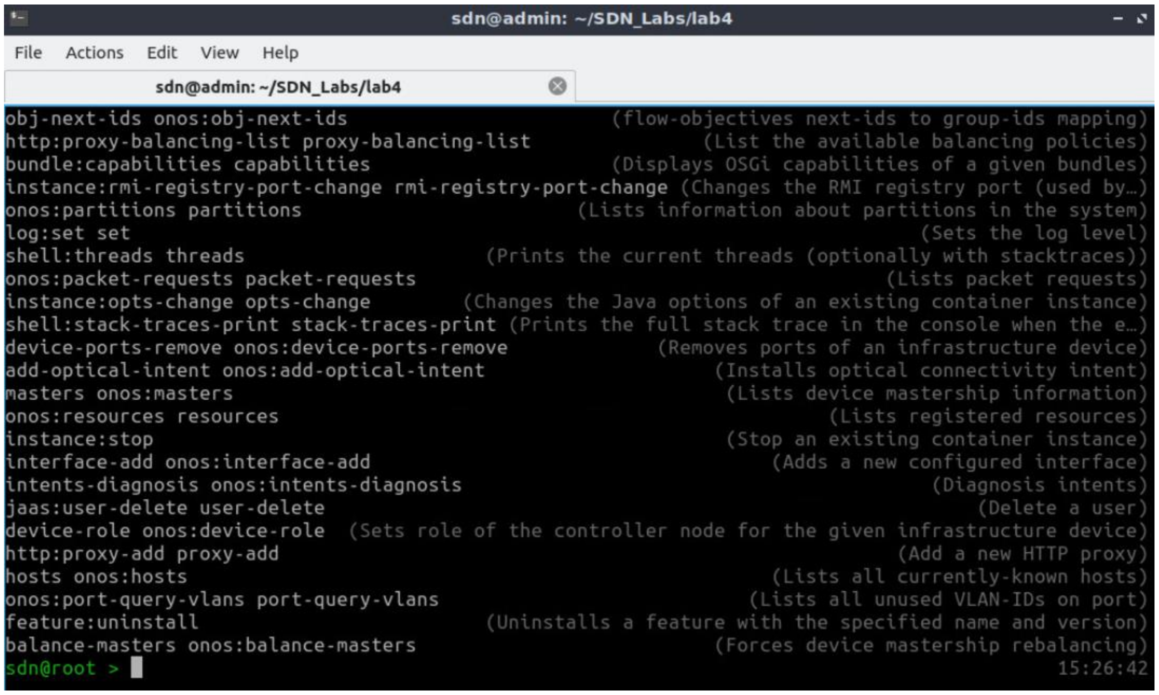
Step 4. ONOS supplies a set of its own commands, in order to list all available commands, click the TAB key.


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > 
karaf: do you wish to see all 409 possibilities (409 lines)?
```

Figure 13. Displaying a list of ONOS commands.

Step 5. To confirm displaying all 401 possibilities of ONOS commands, click the TAB key one more time. This will display all ONOS commands on the left-hand side of the CLI, as well as their explanation on the right-hand side of the CLI.

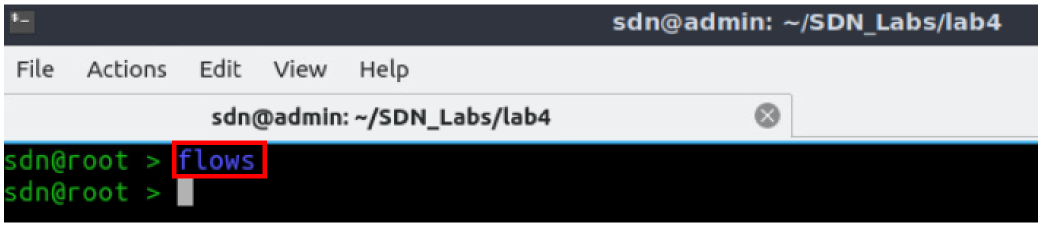

sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
obj-next-ids onos:obj-next-ids                          (flow-objectives next-ids to group-ids mapping)
http:proxy-balancing-list proxy-balancing-list                 (List the available balancing policies)
bundle:capabilities capabilities                        (Displays OSGi capabilities of a given bundles)
instance:rmi-registry-port-change rmi-registry-port-change (Changes the RMI registry port (used by…)
onos:partitions partitions                                 (Lists information about partitions in the system)
log:set set                                                                             (Sets the log level)
shell:threads threads                          (Prints the current threads (optionally with stacktraces))
onos:packet-requests packet-requests                                               (Lists packet requests)
instance:opts-change opts-change             (Changes the Java options of an existing container instance)
shell:stack-traces-print stack-traces-print (Prints the full stack trace in the console when the e…)
device-ports-remove onos:device-ports-remove                  (Removes ports of an infrastructure device)
add-optical-intent onos:add-optical-intent                           (Installs optical connectivity intent)
masters onos:masters                                                 (Lists device mastership information)
onos:resources resources                                                      (Lists registered resources)
instance:stop                                                        (Stop an existing container instance)
interface-add onos:interface-add                                          (Adds a new configured interface)
intents-diagnosis onos:intents-diagnosis                                               (Diagnosis intents)
jaas:user-delete user-delete                                                               (Delete a user)
device-role onos:device-role  (Sets role of the controller node for the given infrastructure device)
http:proxy-add proxy-add                                                               (Add a new HTTP proxy)
hosts onos:hosts                                                          (Lists all currently-known hosts)
onos:port-query-vlans port-query-vlans                                  (Lists all unused VLAN-IDs on port)
feature:uninstall                              (Uninstalls a feature with the specified name and version)
balance-masters onos:balance-masters                                   (Forces device mastership rebalancing)
sdn@root > 
                                                                                                15:26:42
```

Figure 14. Displaying a list of ONOS commands.

Step 6. To display the list of all currently known flows for the ONOS controller, type the following command.

```
flows
```

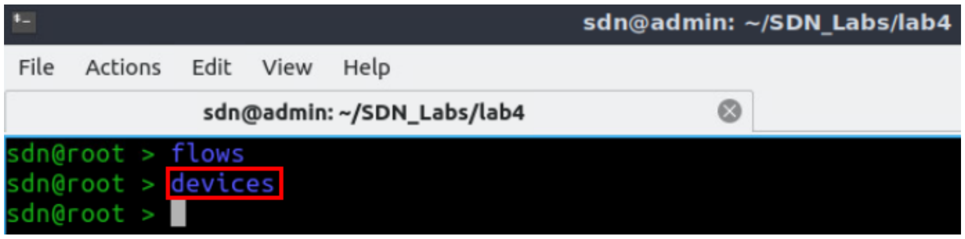

sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows
sdn@root > 
```

Figure 15. Displaying the current known flows.

Step 7. To display the list of all currently known devices (OVS switches), type the following command.

```
devices
```

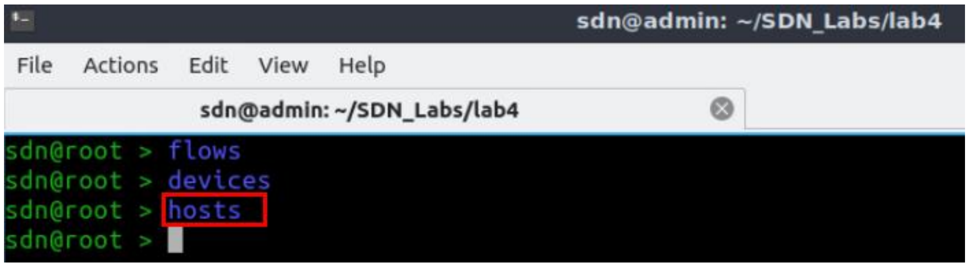

sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows
sdn@root > devices
sdn@root >
```

Figure 16. Displaying the current known devices (switches).

Step 8. To display the list of all currently known hosts, type the following command.

```
hosts
```

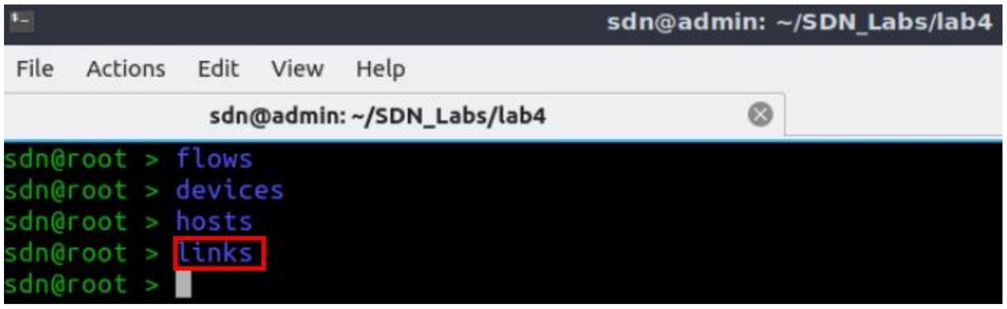

sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows
sdn@root > devices
sdn@root > hosts
sdn@root >
```

Figure 17. Displaying the current known hosts.

Step 9. To display the list of all currently known links, type the following command.

```
links
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows
sdn@root > devices
sdn@root > hosts
sdn@root > links
sdn@root >
```

Figure 18. Displaying the current known link.

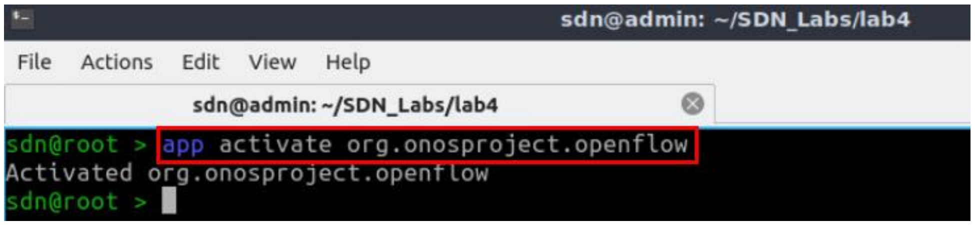
Consider Figures 15, 16, 17, and 18. No flows, devices, hosts, or link are displayed since we have not activated the necessary applications that allow the controller to discover them.

3.1 Activating ONOS OpenFlow application

In this section you will activate OpenFlow application that comes with ONOS and allows to speak OpenFlow protocol with the devices. You will notice how the Mininet topology becomes visible, i.e., the devices (switches), and hosts are now recognized by ONOS.

Step 1. In ONOS terminal, issue the following command to activate the OpenFlow application.

```
app activate org.onosproject.openflow
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > app activate org.onosproject.openflow
Activated org.onosproject.openflow
sdn@root >
```

Figure 19. Activating OpenFlow application.

Step 2. Open Mininet terminal.

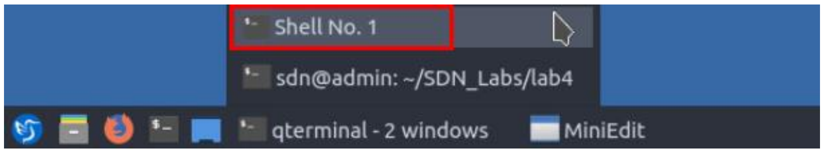
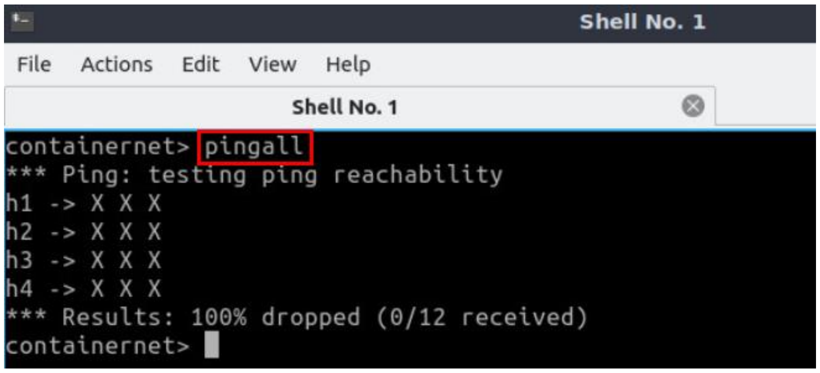


Figure 20. Opening Mininet terminal.

Step 3. Once the OpenFlow application is activated, you might not get accurate results immediately, for example, not all the hosts are discovered yet. In order to stimulate ONOS and the activated application, perform a `pingall` command. This command pings from every host in the network to all other hosts. To do so, write the following command.

```
pingall
```



```
containernet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
containernet>
```

Figure 21. Pinging all hosts to stimulate host discovery in the Controller.

Consider Figure 21. The connectivity test resulted in 100% dropped. The pings sent from a host to a destination are IP packets received by a switch. For example, pinging host h2 from h1 is received by switch s1. Since there are no flows inserted that deal with IP packets, then the packets will be dropped immediately.

Step 4. Go to ONOS terminal.

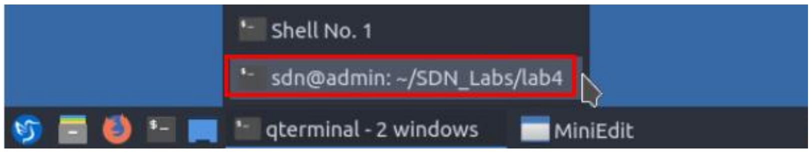
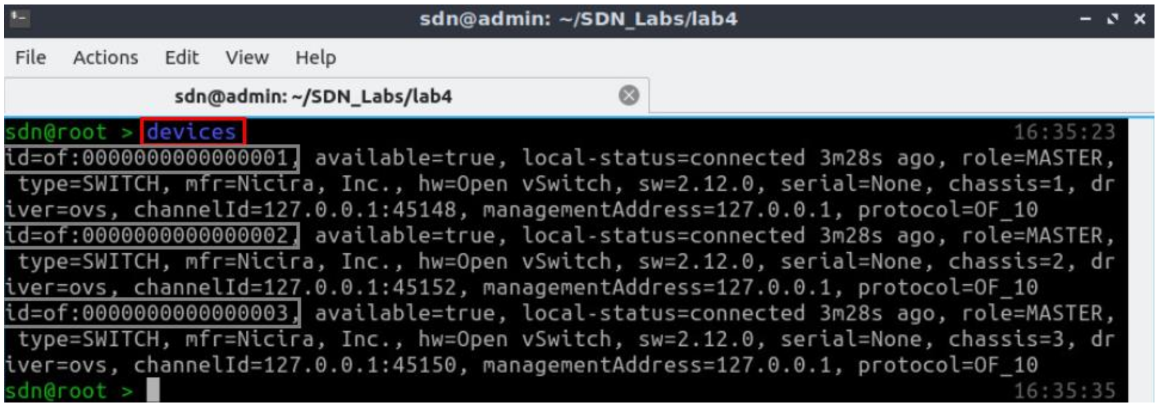


Figure 22. Opening ONOS terminal.

Step 5. To display the list of all currently known devices (OVS switches), type the following command.

```
devices
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

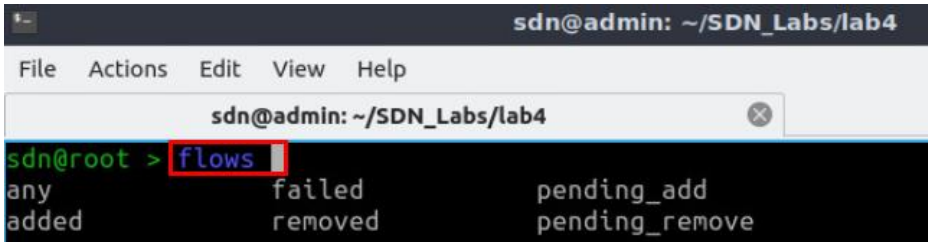
```
sdn@root > devices                                                    16:35:23
id=of:0000000000000001, available=true, local-status=connected 3m28s ago, role=MASTER,
 type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.12.0, serial=None, chassis=1, dr
iver=ovs, channelId=127.0.0.1:45148, managementAddress=127.0.0.1, protocol=OF_10
id=of:0000000000000002, available=true, local-status=connected 3m28s ago, role=MASTER,
 type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.12.0, serial=None, chassis=2, dr
iver=ovs, channelId=127.0.0.1:45152, managementAddress=127.0.0.1, protocol=OF_10
id=of:0000000000000003, available=true, local-status=connected 3m28s ago, role=MASTER,
 type=SWITCH, mfr=Nicira, Inc., hw=Open vSwitch, sw=2.12.0, serial=None, chassis=3, dr
iver=ovs, channelId=127.0.0.1:45150, managementAddress=127.0.0.1, protocol=OF_10
sdn@root >                                                            16:35:35
```

Figure 23. Displaying the current known devices (switches).

Consider Figure 23. The three switches are displayed with their corresponding ids, as well as additional attributes, such as its status (`available=true`), the type of the switch (`hw=Open vSwitch`), and which OpenFlow version the switch is running (`protocol=OF_10`).

Step 6. ONOS commands might have multiple options, to display the list of options for the flows command, write the command `flows` and click the `TAB` button.

```
flows
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows
any                 failed              pending_add
added               removed             pending_remove
```

Figure 24. Displaying the current known flows.

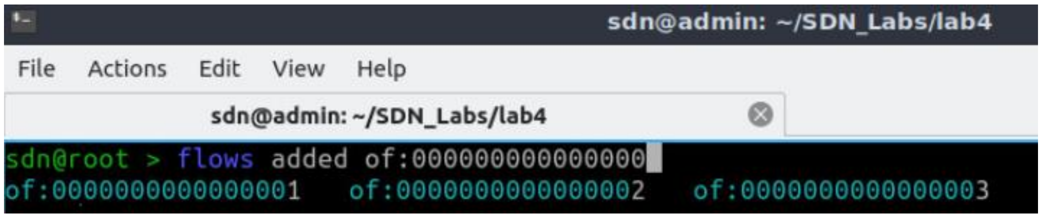
Consider Figure 24. The `flows` command has the multiple options according to the state of the flow, which could be:

- `added`: the flow has been added to the switch.
- `failed`: the flow failed to be added.
- `pending_add`: the flow has been submitted and forwarded to the switch.

- `pending_remove`: the request to remove the flow has been submitted and forwarded to the switch.
- `removed`: the rule has been removed.
- `any`: all flows.

Step 7. To display the list of all added switches, complete the above command by writing `added`, then click the `TAB` button.

```
flows added
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

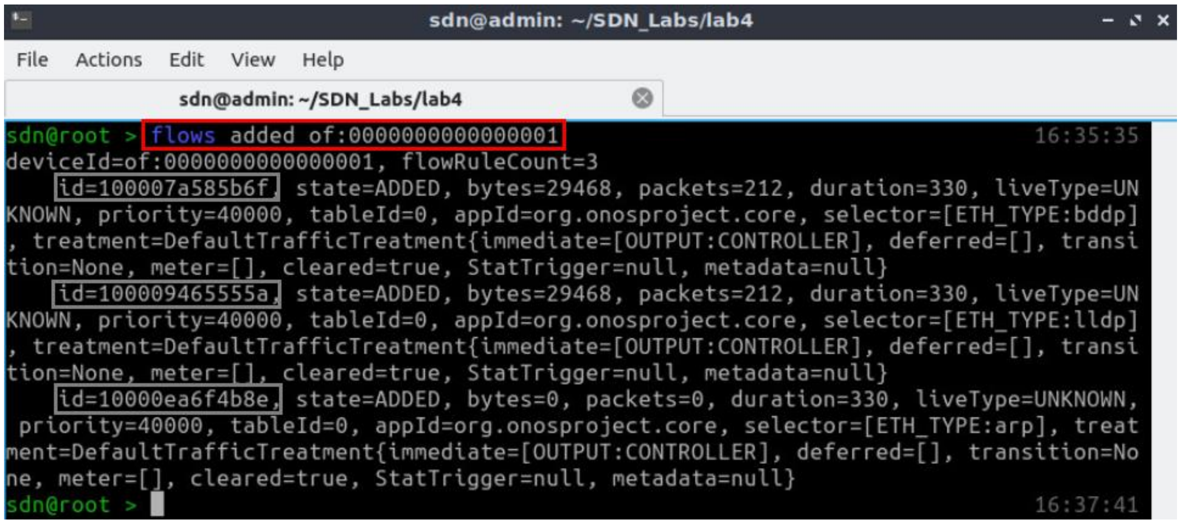
```
sdn@root > flows added of:000000000000000
of:0000000000000001    of:0000000000000002    of:0000000000000003
```

Figure 25. Displaying the current known added flows for switch 1.

Consider Figure 25. Once you click the `TAB` button, ONOS CLI automatically fills the common prefix of all the switches' ids (000000000000000). You can also alternate between the displayed switches' ids using the `TAB` button.

Step 8. To display the list of all added flows for switch 1 (id: 00000000000001), complete the id of the switch (you only have to enter the number `1` for switch s1) and hit enter.

```
flows added of:00000000000001
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows added of:0000000000000001                               16:35:35
deviceId=of:0000000000000001, flowRuleCount=3
    id=100007a585b6f, state=ADDED, bytes=29468, packets=212, duration=330, liveType=UN
KNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:bddp]
, treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transi
tion=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=100009465555a, state=ADDED, bytes=29468, packets=212, duration=330, liveType=UN
KNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:lldp]
, treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transi
tion=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=10000ea6f4b8e, state=ADDED, bytes=0, packets=0, duration=330, liveType=UNKNOWN,
 priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:arp], treat
ment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=No
ne, meter=[], cleared=true, StatTrigger=null, metadata=null}
sdn@root >                                                               16:37:41
```

Figure 26. Displaying the current known added flows for switch 1.

Consider Figure 26. Three flows for switch s1 were added. ONOS provides many details about the flows inserted in the switches. For example, each flow entry defines a `selector` and `treatment` which are the set of traffic matched by the flow entry and how this traffic should be handled, respectively.

The controller has installed three initial flows which are:

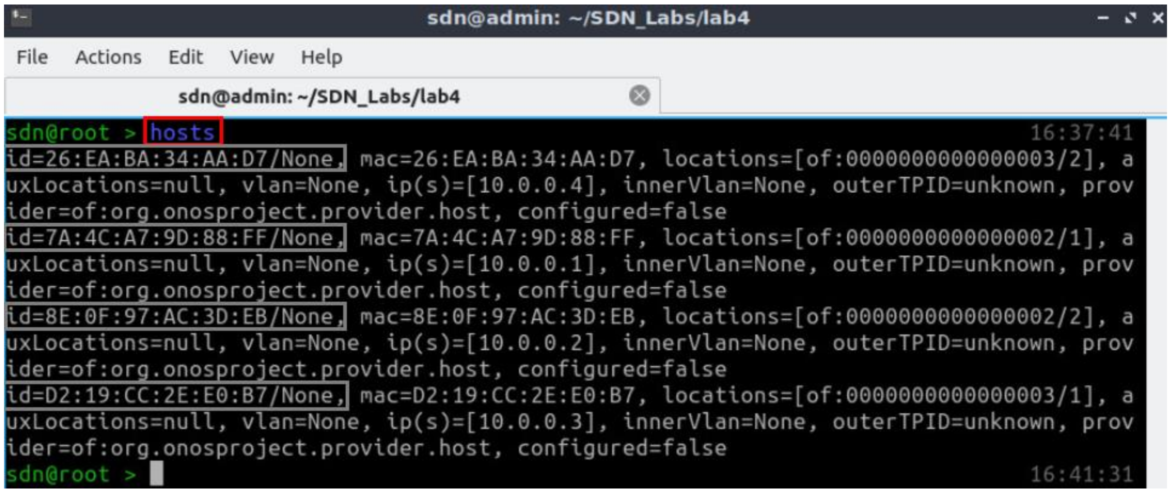
- Flow 1 (`ETH_TYPE=lldp`): forwards Link Layer Discovery Protocol (LLDP) packets to the controller (`[OUTPUT:CONTROLLER]`).
- Flow 2 (`ETH_TYPE=bddp`): forwards Broadcast Domain Discovery Protocol (BDDP) to the controller (`[OUTPUT:CONTROLLER]`).
- Flow 3 (`ETH_TYPE=arp`): forwards Address Resolution Protocol (ARP) packets to the controller (`[OUTPUT:CONTROLLER]`).

The above flows are used for link and host discovery. Notice as well that each flow entry is tagged by an `appId` (application id), this `appId` identifies which application installed the corresponding flow entry. Other important details include:

- `packets`: number of packets forwarded or dropped for that flow.
- `bytes`: byte count of the matched packets.
- `tableId`: id of the table in which these flows are installed.
- `duration`: time elapsed in seconds since the flow was installed.

Step 9. To display the list of all currently known hosts, type the following command.

```
hosts
```



```
sdn@root > hosts                                                          16:37:41
id=26:EA:BA:34:AA:D7/None, mac=26:EA:BA:34:AA:D7, locations=[of:0000000000000003/2], a
uxLocations=null, vlan=None, ip(s)=[10.0.0.4], innerVlan=None, outerTPID=unknown, prov
ider=of:org.onosproject.provider.host, configured=false
id=7A:4C:A7:9D:88:FF/None, mac=7A:4C:A7:9D:88:FF, locations=[of:0000000000000002/1], a
uxLocations=null, vlan=None, ip(s)=[10.0.0.1], innerVlan=None, outerTPID=unknown, prov
ider=of:org.onosproject.provider.host, configured=false
id=8E:0F:97:AC:3D:EB/None, mac=8E:0F:97:AC:3D:EB, locations=[of:0000000000000002/2], a
uxLocations=null, vlan=None, ip(s)=[10.0.0.2], innerVlan=None, outerTPID=unknown, prov
ider=of:org.onosproject.provider.host, configured=false
id=D2:19:CC:2E:E0:B7/None, mac=D2:19:CC:2E:E0:B7, locations=[of:0000000000000003/1], a
uxLocations=null, vlan=None, ip(s)=[10.0.0.3], innerVlan=None, outerTPID=unknown, prov
ider=of:org.onosproject.provider.host, configured=false
sdn@root >                                                                16:41:31
```

Figure 27. Displaying the current known hosts.

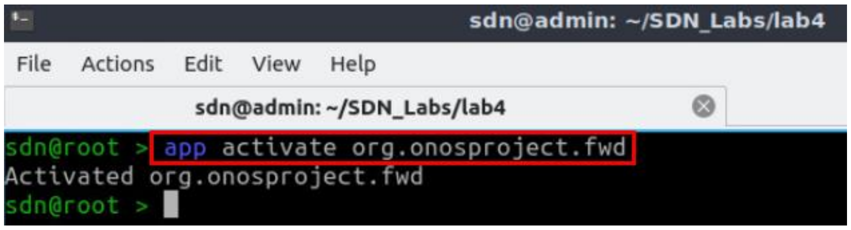
Consider Figure 27. Each discovered host is displayed along some details, such as the id of the host, the location where the host is connected to (with the id of the switch), and the IP address of it. Notice that the id of the hosts, as well as their mac address might be different than those depicted in Figure 27, since Mininet generates new values at each execution.

3.2 Activating ONOS forwarding application

In the previous section, when performing a connectivity test, 100% of the packets were dropped. The pings sent from a host to a destination are IP packets received by a switch, which does not have flows that match IP packets. In this section, you will activate a simple forwarding application that comes with ONOS. This application installs flows in response to every *miss* IP packet that arrives at the controller.

Step 1. To enable the forwarding application, type the command shown below.

```
app activate org.onosproject.fwd
```

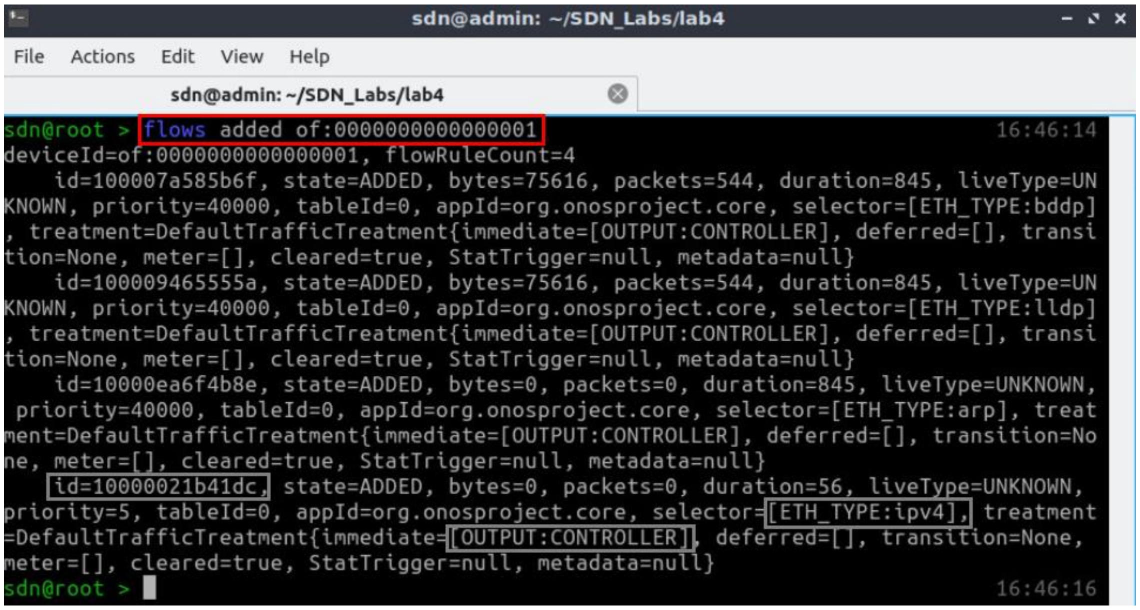


```
sdn@root > app activate org.onosproject.fwd
Activated org.onosproject.fwd
sdn@root >
```

Figure 28. Activating the forwarding application.

Step 2. To display the flows of switch s1, type the following command.

```
flows added of:0000000000000001
```



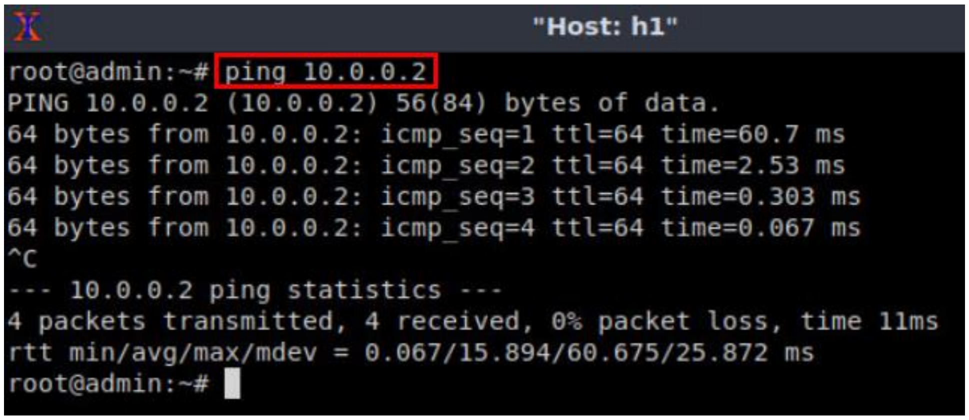
```
sdn@root > flows added of:0000000000000001                                   16:46:14
deviceId=of:0000000000000001, flowRuleCount=4
    id=100007a585b6f, state=ADDED, bytes=75616, packets=544, duration=845, liveType=UN
KNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:bddp]
, treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transi
tion=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=100009465555a, state=ADDED, bytes=75616, packets=544, duration=845, liveType=UN
KNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:lldp]
, treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transi
tion=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=10000ea6f4b8e, state=ADDED, bytes=0, packets=0, duration=845, liveType=UNKNOWN,
 priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:arp], treat
ment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=No
ne, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=10000021b41dc, state=ADDED, bytes=0, packets=0, duration=56, liveType=UNKNOWN,
priority=5, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:ipv4], treatment
=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transition=None,
meter=[], cleared=true, StatTrigger=null, metadata=null}
sdn@root >                                                                    16:46:16
```

Figure 29. Displaying the current known devices (switches).

Consider Figure 29. A new flow is added with `ETH_TYPE:ipv4` that deals with IPv4 packet by forwarding them the controller (`OUTPUT:CONTROLLER`), which in turns decides the action on the corresponding packet.

Step 3. On host h1 terminal, run a connectivity test by issuing the command shown below. To stop the test, press `Ctrl+c`.

```
ping 10.0.0.2
```

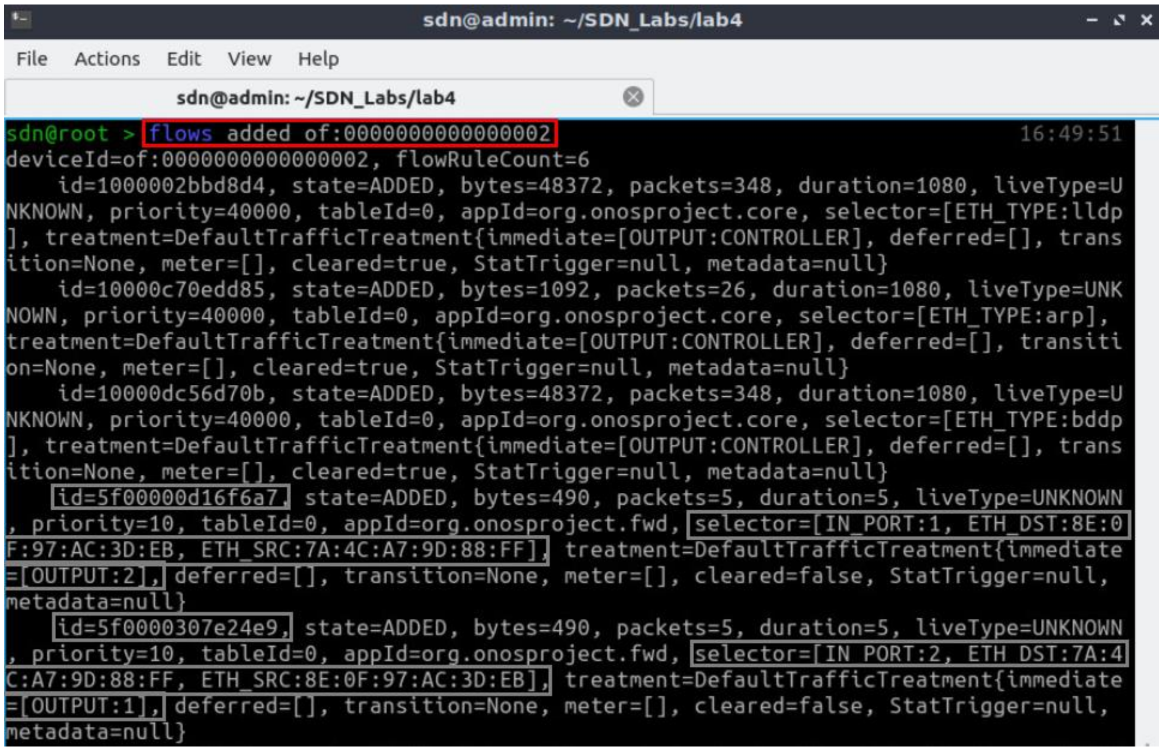


```
root@admin:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=60.7 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.53 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.303 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.067 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 0.067/15.894/60.675/25.872 ms
root@admin:~#
```

Figure 30. Pinging host h2 from host h1.

Step 4. To display the flows of switch s2, type the following command on ONOS terminal.

```
flows added of:0000000000000002
```


sdn@admin: ~/SDN_Labs/lab4
File Actions Edit View Help
sdn@admin: ~/SDN_Labs/lab4

```
sdn@root > flows added of:0000000000000002                                     16:49:51
deviceId=of:0000000000000002, flowRuleCount=6
    id=1000002bbd8d4, state=ADDED, bytes=48372, packets=348, duration=1080, liveType=U
NKNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:lldp
], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], trans
ition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=10000c70edd85, state=ADDED, bytes=1092, packets=26, duration=1080, liveType=UNK
NOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:arp],
treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], transiti
on=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=10000dc56d70b, state=ADDED, bytes=48372, packets=348, duration=1080, liveType=U
NKNOWN, priority=40000, tableId=0, appId=org.onosproject.core, selector=[ETH_TYPE:bddp
], treatment=DefaultTrafficTreatment{immediate=[OUTPUT:CONTROLLER], deferred=[], trans
ition=None, meter=[], cleared=true, StatTrigger=null, metadata=null}
    id=5f00000d16f6a7, state=ADDED, bytes=490, packets=5, duration=5, liveType=UNKNOWN
, priority=10, tableId=0, appId=org.onosproject.fwd, selector=[IN_PORT:1, ETH_DST:8E:0
F:97:AC:3D:EB, ETH_SRC:7A:4C:A7:9D:88:FF], treatment=DefaultTrafficTreatment{immediate
=[OUTPUT:2], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null,
metadata=null}
    id=5f0000307e24e9, state=ADDED, bytes=490, packets=5, duration=5, liveType=UNKNOWN
, priority=10, tableId=0, appId=org.onosproject.fwd, selector=[IN_PORT:2, ETH_DST:7A:4
C:A7:9D:88:FF, ETH_SRC:8E:0F:97:AC:3D:EB], treatment=DefaultTrafficTreatment{immediate
=[OUTPUT:1], deferred=[], transition=None, meter=[], cleared=false, StatTrigger=null,
metadata=null}
```

Figure 31. Displaying the added flows on switch s2.

Consider Figure 31. After pinging host h2 from h1, two flows are installed on switch s2. The first flow (`id=5f0000d16f6a7`) instructs the switch to forward incoming packets at port 1 (`IN_PORT:1`) out of port 2 (`OUTPUT:2`). Similarly, the second flow instructs the switch to forward the packets from port 2 to port 1. The inserted flows allow switch s1 to exchange packets between hosts h1 and h2 without relaying them to the controller.

Note that these two flows expire after a certain duration. This is because the forwarding application sets an expiry time for the inserted flows to avoid overflowing the flow table of the switches.

4 Navigating ONOS GUI

The ONOS GUI is a *single-page web-application*, providing a visual interface to the ONOS controller. In this section, you will navigate through ONOS GUI and discover a number of its features.

4.1 Accessing the web user interface

Step 1. Open the web browser by clicking on the shortcut located in the lower left-hand side.

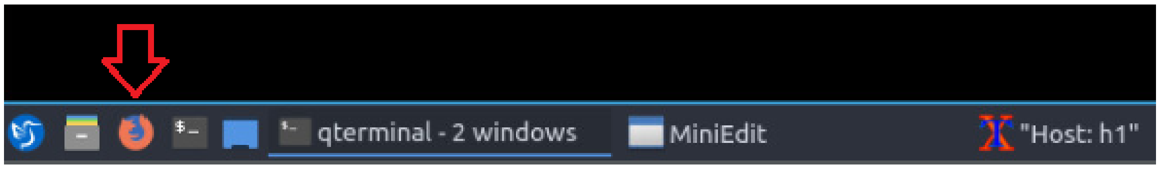


Figure 32. Opening the web browser

Step 2. Navigate to the following URL to access ONOS web user interface:

```
localhost:8181/onos/ui
```

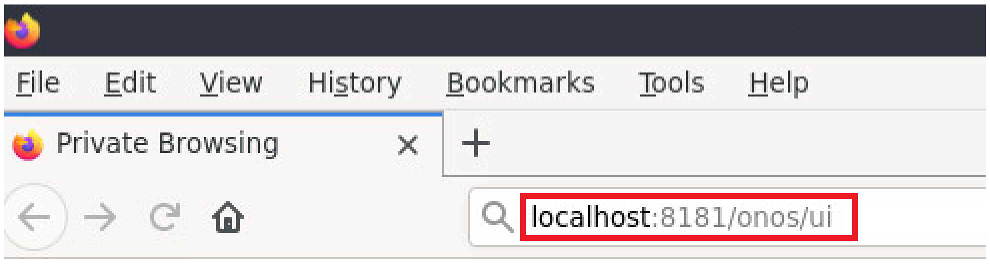


Figure 33. Opening the ONOS web user interface.

Step 3. Provide the following credentials to access the web user interface:

- User: `onos`
- Password: `rocks`

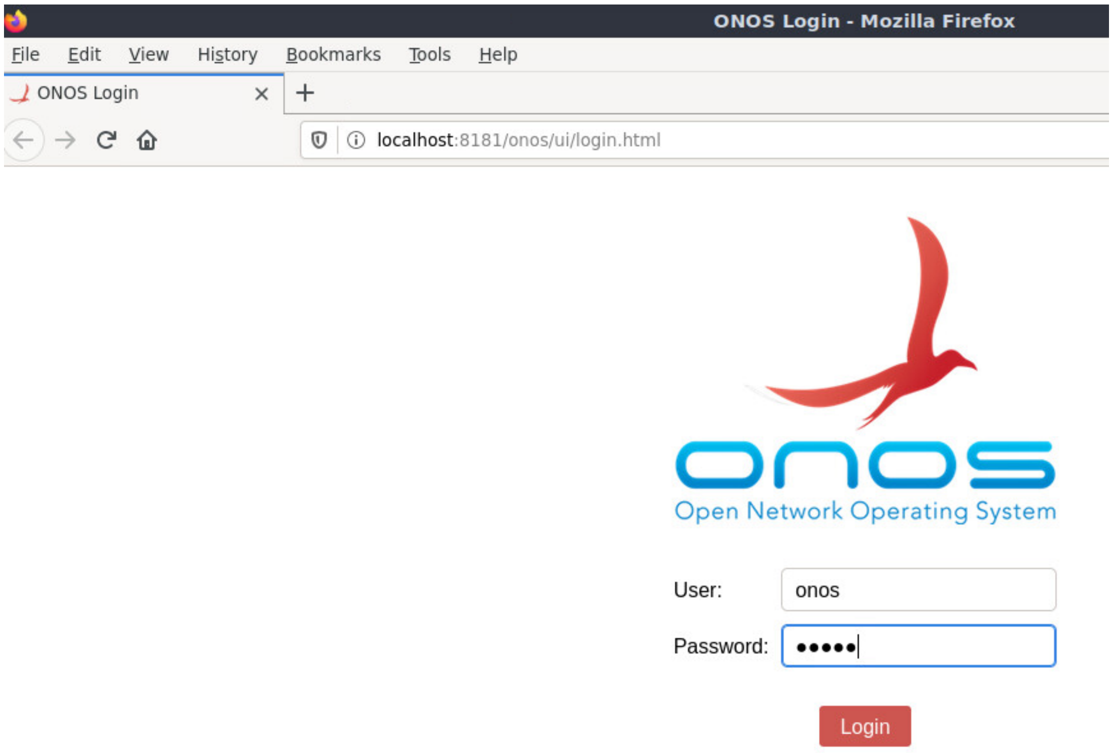
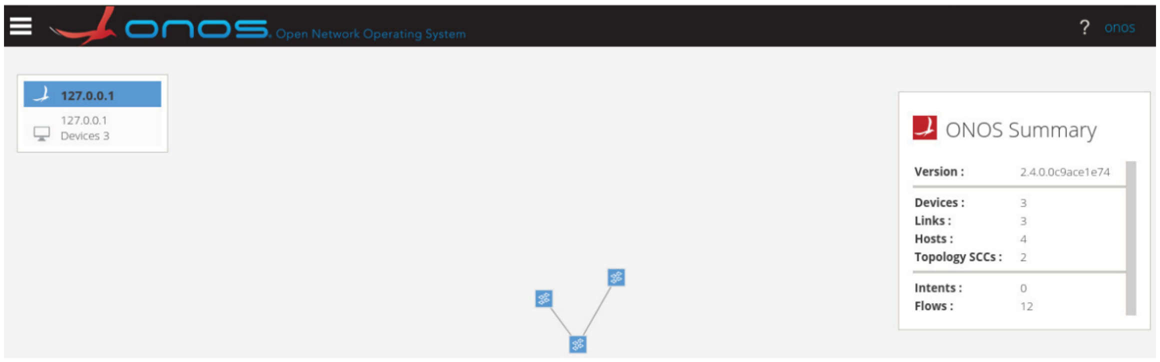

ONOS Login - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ONOS Login
localhost:8181/onos/ui/login.html
Open Network Operating System
User: onos
Password: •••••
Login

Figure 34. ONOS authentication window.

A topology consisting of three switches will be displayed. Such topology corresponds to the one created on Mininet.



ONOS Summary

Version :	2.4.0.0c9ace1e74
Devices :	3
Links :	3
Hosts :	4
Topology SCCs :	2
Intents :	0
Flows :	12

Figure 35. Displaying the topology in ONOS GUI.

Notice that the network components will show up in a random arrangement. The user can click on the components and drag them to their preferred location.

4.2 Exploring network components

ONOS web user interface provides the tools to monitor the specification of each device that comprises the network.

Step 1. To open ONOS menu, click on the upper left-hand side icon. A drop-down menu will be displayed. To check the devices, click on devices. A new window will emerge.

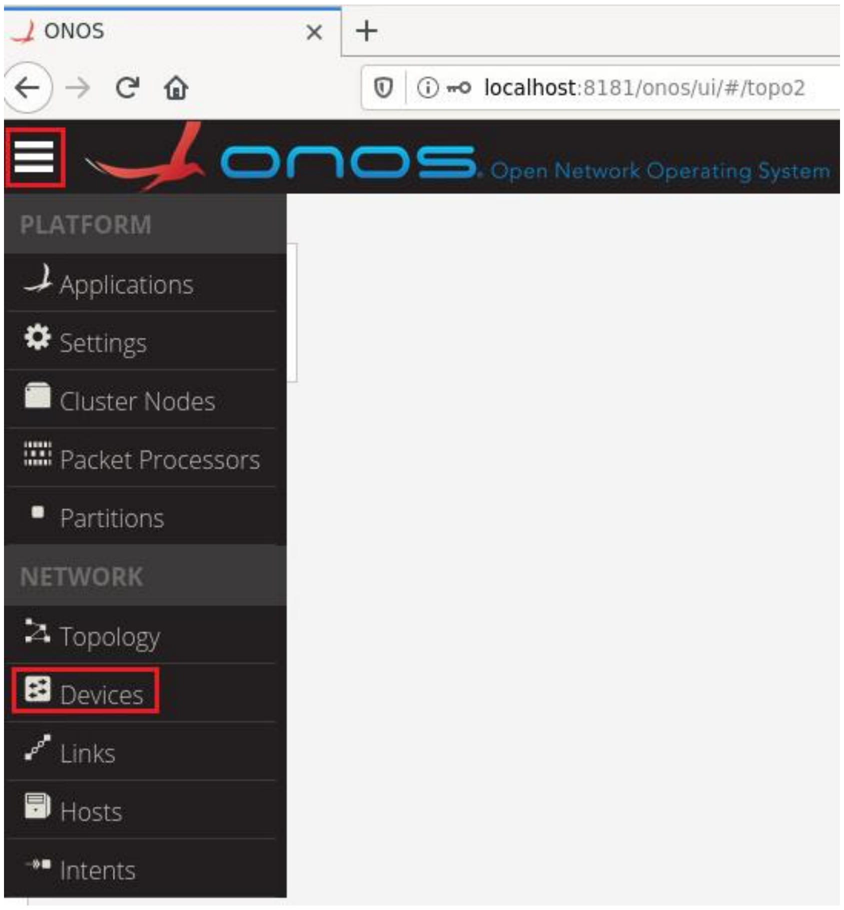
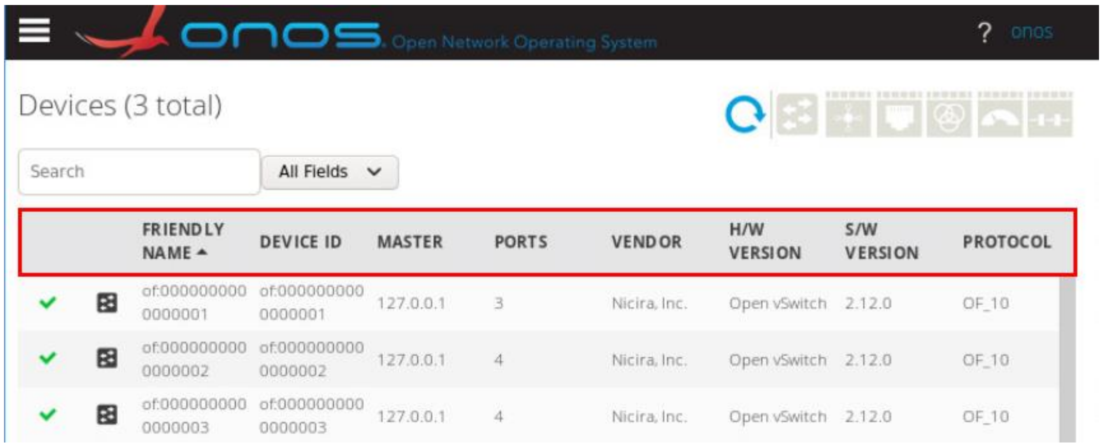


Figure 36. Opening devices.

The emerging window will display three devices. Those three devices correspond to the three switches that compounds the topology. The information provided in the GUI include:

- FRIENDLY NAME: if not specified, it is the name of the device.
- DEVICE ID: name of the switch.
- MASTER: IP address where ONOS is running. ONOS is running locally (127.0.0.1).
- PORTS: number of ports of the switch.
- PROTOCOL: OpenFlow version running on the switch.



Devices (3 total)

		FRIENDLY NAME ▲	DEVICE ID	MASTER	PORTS	VENDOR	H/W VERSION	S/W VERSION	PROTOCOL
✓		of:0000000000000001	of:0000000000000001	127.0.0.1	3	Nicira, Inc.	Open vSwitch	2.12.0	OF_10
✓		of:0000000000000002	of:0000000000000002	127.0.0.1	4	Nicira, Inc.	Open vSwitch	2.12.0	OF_10
✓		of:0000000000000003	of:0000000000000003	127.0.0.1	4	Nicira, Inc.	Open vSwitch	2.12.0	OF_10

Figure 37. Devices' information.

Step 2. From the menu list, click on hosts to verify the hosts' information.

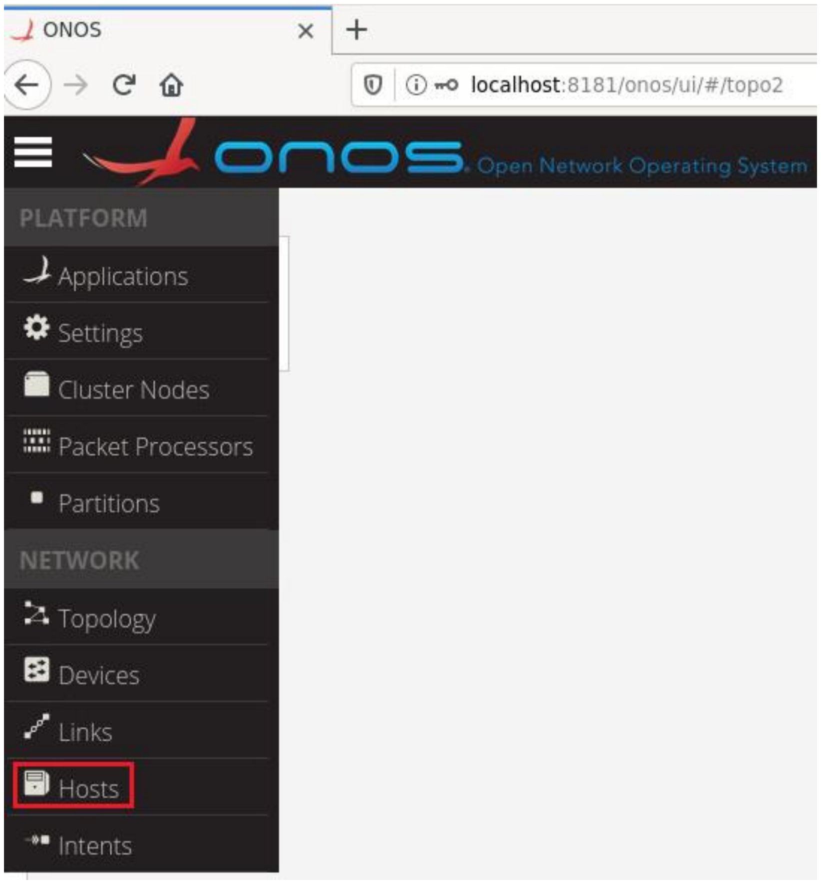
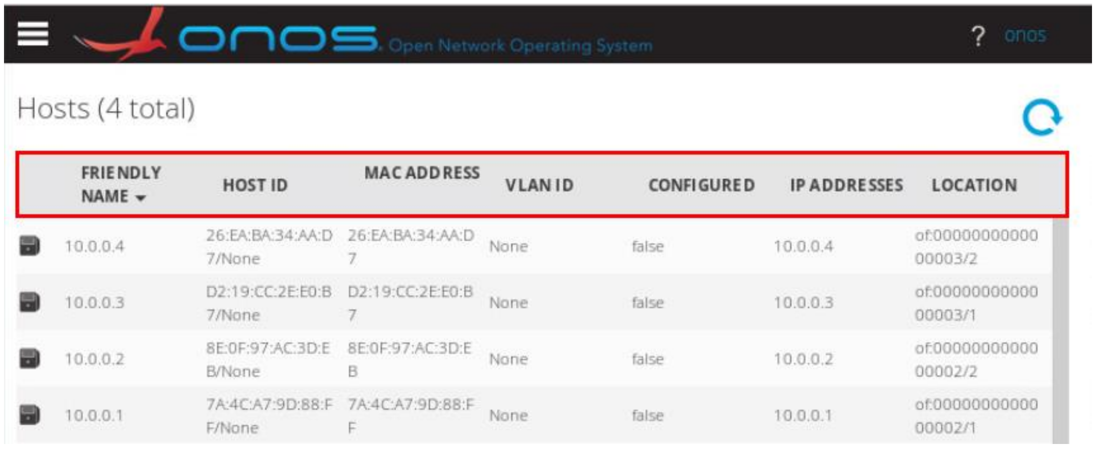


Figure 38. Opening hosts.

Similarly, the new window presents the information regarding the hosts that compound the topology. The information provided in the GUI include:

- Host ID/MAC ADDRESS: mac address of the host.
- IP ADDRESSES: IP address of the host.
- Location: the port of the switch connected to the host.



Hosts (4 total)

	FRIENDLY NAME	HOST ID	MAC ADDRESS	VLAN ID	CONFIGURED	IP ADDRESSES	LOCATION
	10.0.0.4	26:EA:BA:34:AA:D7/None	26:EA:BA:34:AA:D7	None	false	10.0.0.4	of:000000000000000003/2
	10.0.0.3	D2:19:CC:2E:E0:B7/None	D2:19:CC:2E:E0:B7	None	false	10.0.0.3	of:000000000000000003/1
	10.0.0.2	8E:0F:97:AC:3D:EB/None	8E:0F:97:AC:3D:EB	None	false	10.0.0.2	of:000000000000000002/2
	10.0.0.1	7A:4C:A7:9D:88:FF/None	7A:4C:A7:9D:88:FF	None	false	10.0.0.1	of:000000000000000002/1

Figure 39. Hosts' information.

Step 3. Go back to the main window (topology).

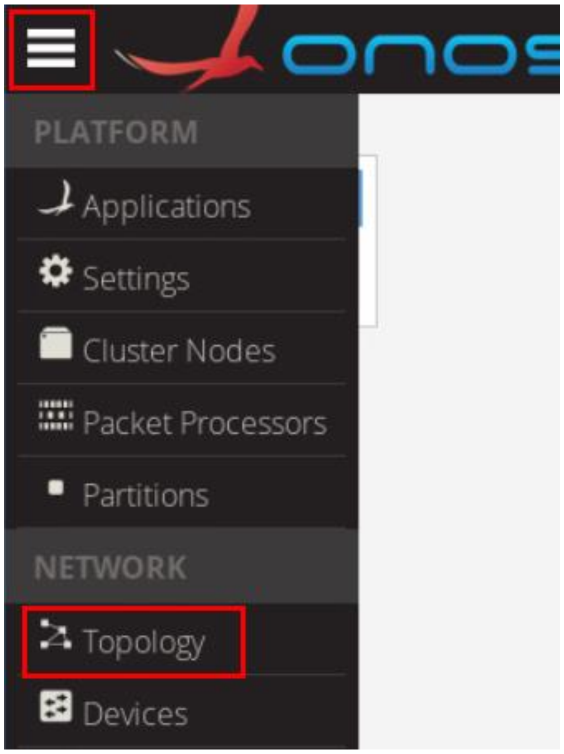


Figure 40. Opening the topology.

Step 4. Click on the bar located in the lower left-hand side. A toolbox will show up. Select the host icon to see the hosts connected to the topology.

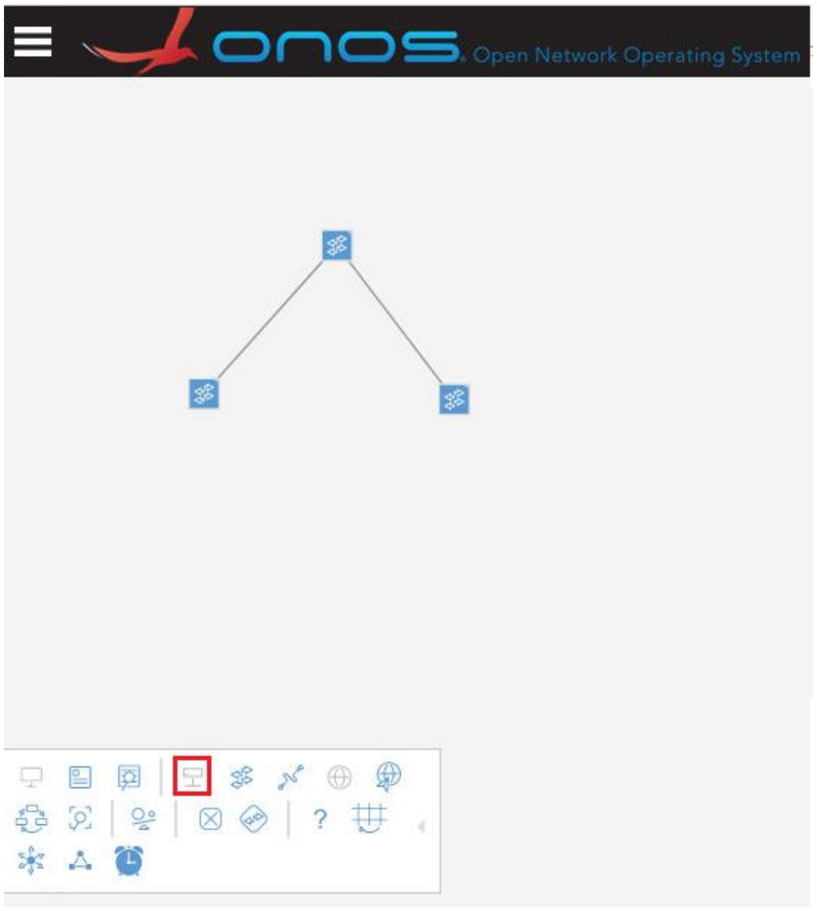


Figure 41. Enabling hosts visualization.

Step 5. Go back to the dashboard to verify the network components.

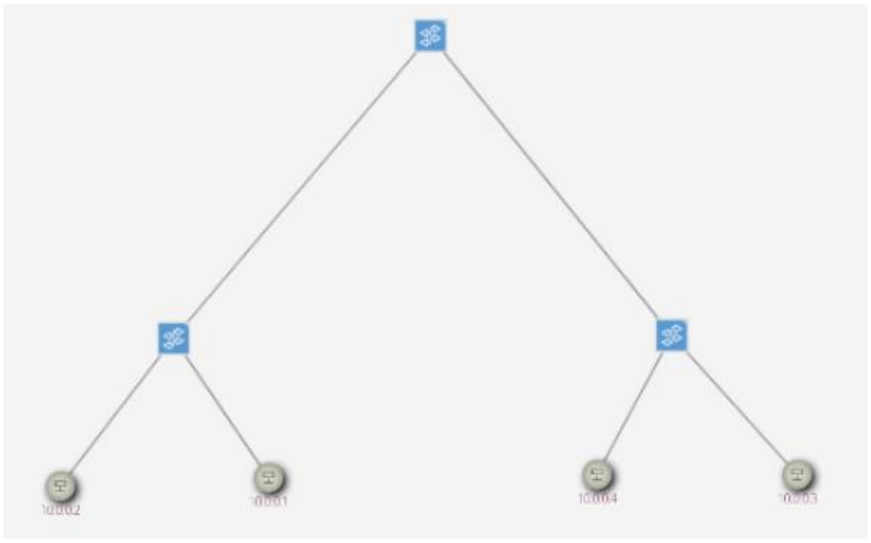
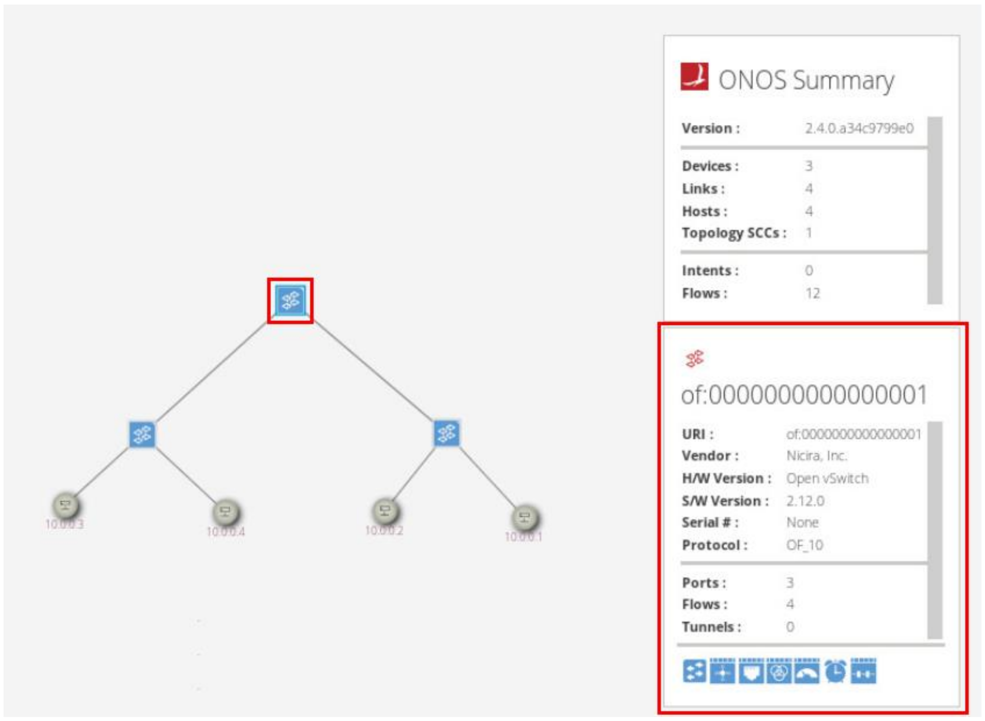


Figure 42. Network components.

Consider Figure 42. The information attached to the hosts include their IP addresses.

Step 6. To view a summary of a specific device, you can click on that device as shown in the figure below.



ONOS Summary

Version :	2.4.0.a34c9799e0
Devices :	3
Links :	4
Hosts :	4
Topology SCCs :	1
Intents :	0
Flows :	12

of:0000000000000001

URI :	of:0000000000000001
Vendor :	Nicira, Inc.
H/W Version :	Open vSwitch
S/W Version :	2.12.0
Serial # :	None
Protocol :	OF_10
Ports :	3
Flows :	4
Tunnels :	0

Figure 43. Network components.

Consider Figure 43. Upon clicking on switch s1, a panel opens on the right-hand side of the topology. The panel displays a summary of the device. For example, if it is a switch, it would display information such the name, number of ports, and flows in that switch.

Step 7. From the menu list, click on Applications to load the applications available in ONOS.

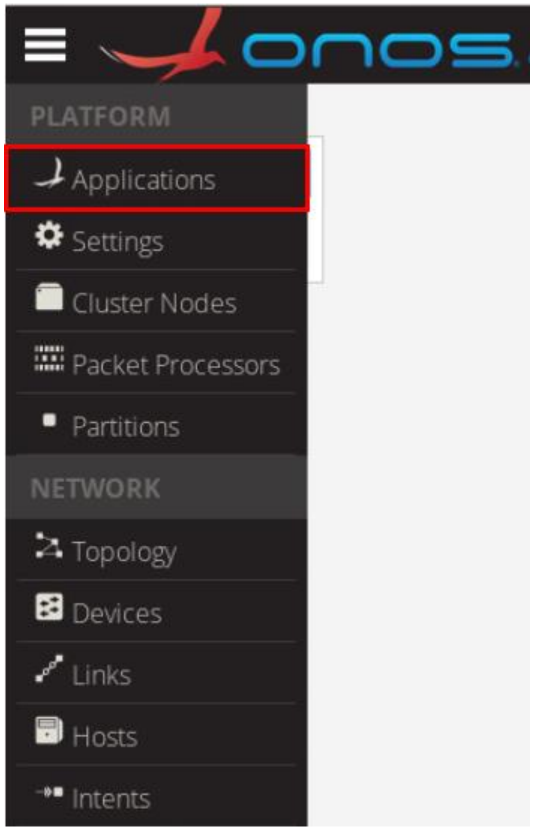
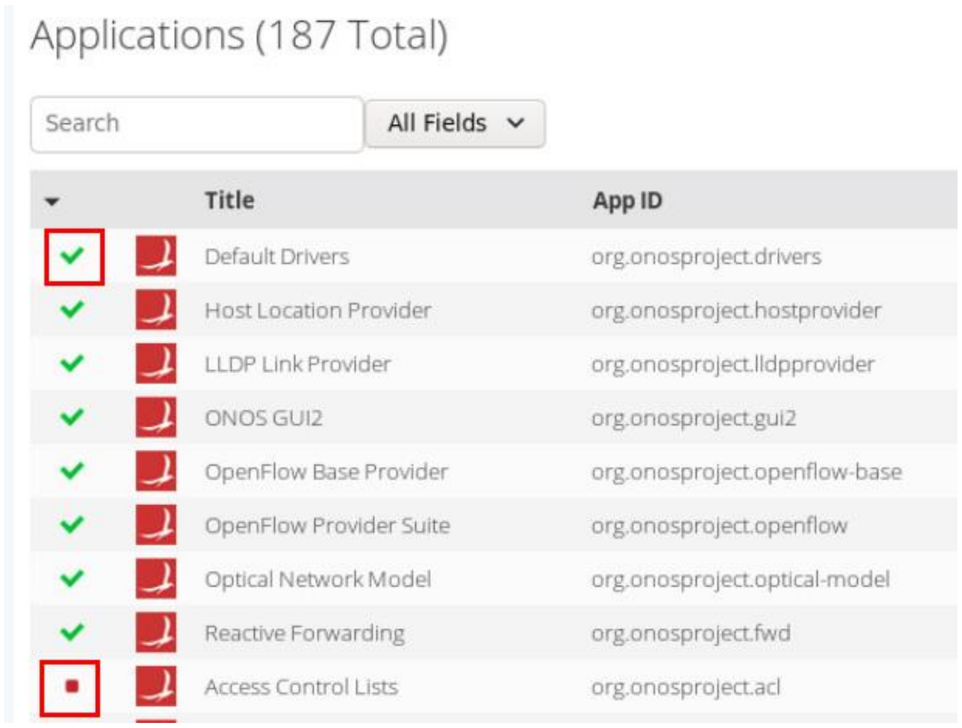


Figure 44. Opening applications.



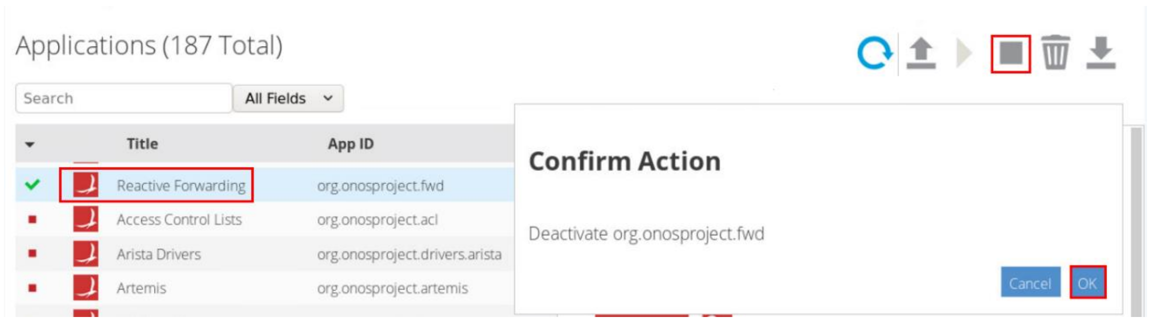
Applications (187 Total)

		Title	App ID
✓		Default Drivers	org.onosproject.drivers
✓		Host Location Provider	org.onosproject.hostprovider
✓		LLDP Link Provider	org.onosproject.lldpprovider
✓		ONOS GUI2	org.onosproject.gui2
✓		OpenFlow Base Provider	org.onosproject.openflow-base
✓		OpenFlow Provider Suite	org.onosproject.openflow
✓		Optical Network Model	org.onosproject.optical-model
✓		Reactive Forwarding	org.onosproject.fwd
■		Access Control Lists	org.onosproject.acl

Figure 45. Opening applications.

Consider Figure 45. A list of all the available applications is displayed. The activated applications are checked in green (e.g., Default Drivers application), whereas the deactivated applications are marked with a red square (e.g., Access Control Lists application). Note that the activated applications are a result of activating the OpenFlow and the forwarding application.

Step 8. To deactivate an application, you can click on the application, hit the deactivation button (gray square on the right-hand side of the window), then confirm the operation. The steps are shown in the below figure, where we deactivate the forwarding application (reactive forwarding).



Applications (187 Total)

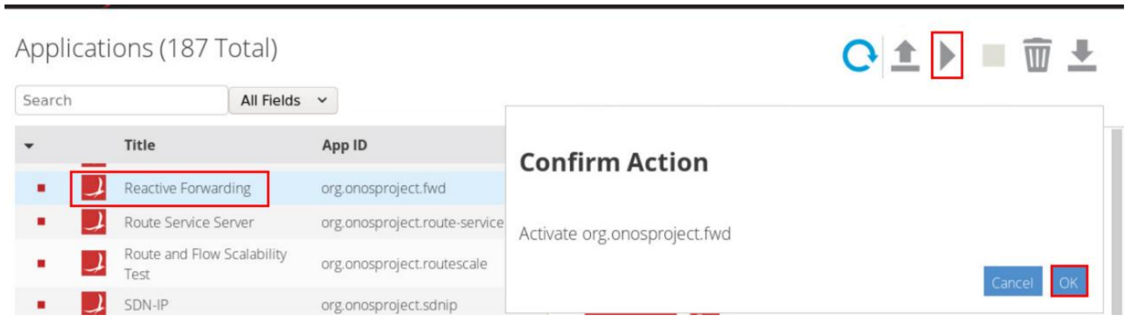
		Title	App ID
✓		Reactive Forwarding	org.onosproject.fwd
■		Access Control Lists	org.onosproject.acl
■		Arista Drivers	org.onosproject.drivers.arista
■		Artemis	org.onosproject.artemis

Confirm Action

Deactivate org.onosproject.fwd

Figure 46. Opening applications.

Step 9. To activate an application, you can click on the application, hit the activation button (gray arrow on the right-hand side of the window), then confirm the operation. The steps are shown in the below figure, where we activate the forwarding application (reactive forwarding).


Applications (187 Total)

	Title	App ID
■	Reactive Forwarding	org.onosproject.fwd
■	Route Service Server	org.onosproject.route-service
■	Route and Flow Scalability Test	org.onosproject.routescale
■	SDN-IP	org.onosproject.sdnip

Confirm Action

Activate org.onosproject.fwd

Figure 47. Opening applications.

Displaying flow tables using GUI

This concludes Lab 4. Stop the emulation and then exit out of MiniEdit and Linux terminal.

References

1. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. "*ONOS: towards an open, distributed SDN OS*," In Proceedings of the third workshop on Hot topics in software defined networking, pp. 1-6, 2014.
2. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "*OpenFlow: enabling innovation in campus networks*." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.
3. D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig, *"Software-defined networking: A comprehensive survey*," Proceedings of the IEEE, 103(1), pp.14-76, 2014.
4. P. Goransson, C. Black, T. Culver. "*Software defined networks: a comprehensive approach"*. Morgan Kaufmann, 2016.
5. Mininet walkthrough, [Online]. Available: http://mininet.org.