INDEX

Documentation for using Opendaylight with Mininet.

1. Installation :

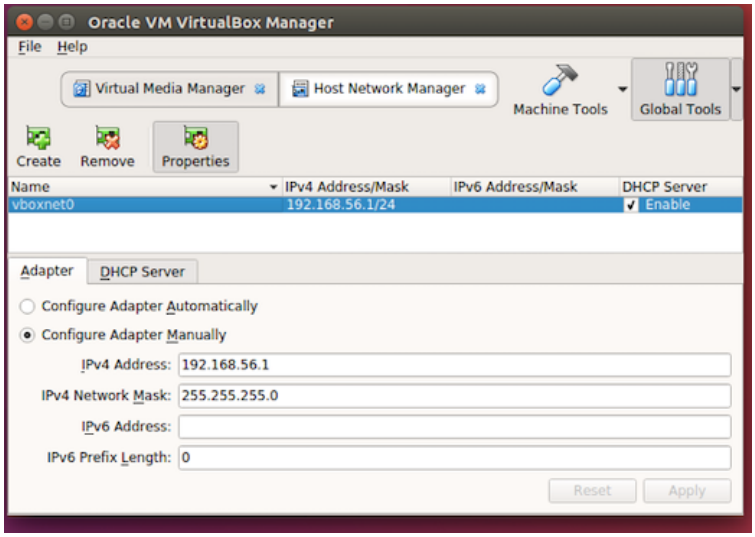
Opendaylight is an SDN controller, mininet is used to create virtual SDN for ODL to connect.

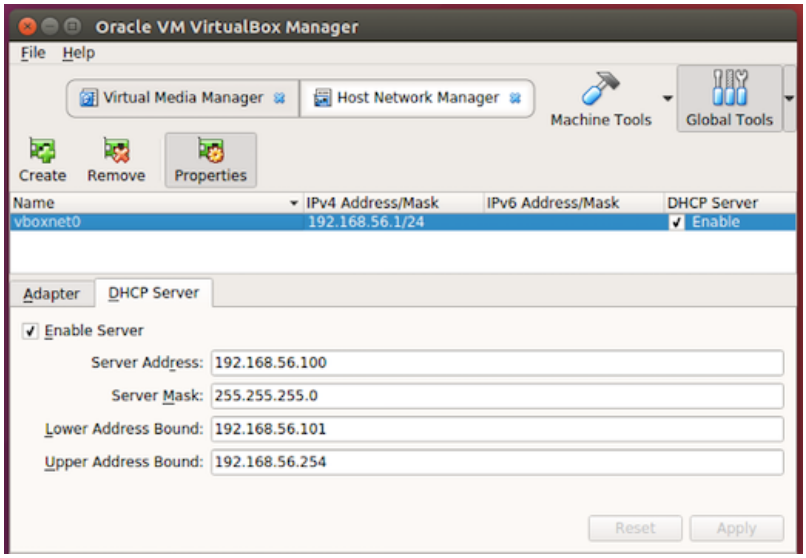
- Primary OS Ubuntu 16.04 on nuc.
- Mininet runs on VM which runs Ubuntu 14.04 and ODL runs on Ubuntu.

Setting Up Mininet

- Download and set up mininet virtual machine using any of the options in the link. http://mininet.org/download/ . Here, Option 1 is used.
- mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.ovf is used. After downloading, go to the downloads folder and decompress the file in cli using

  tar -xvf
- Create a VM by name Mininet-VM and import the mininet's .ovf file.
- Go to the Global Tools to the right corner of the VM window and select 'Host Network Manager'.
- Click on 'create' on the top right corner of the window and input name as 'vboxnet0'. Edit configurations as follows.





- Add network adapter to VM by clicking on *settings -> Network -> Adapter2 -> Click on Enable Network Adapter then attached to as Host-only Adapter and name as vboxnet0*. Click on OK.
- Start the VM named Mininet-VM
- The VM can be accessed using cli argument *ssh -Y mininet@192.168.56.101* and click *yes* for the RSA warning.
- Configure eth0 using *ifconfig -a* and *sudo dhclient eth0.* Verify 192.168.56.101 is assigned to eth0 by running *ifconfig* again.
- Permanently configure the new interface by editing the following lines.

```
sudo vi /etc/network/interfaces.

auto eth0
iface eth0 inet dhcp
```

2. Setting up OpenDaylight

Setup Java before OpenDayLight is installed. Here, OpenDaylight is run on the primary OS which runs Ubuntu 16.04.

```
sudo apt-get update
sudo apt-get install default-jre-headless

nano ~/.bashrc

export JAVA_HOME=/usr/lib/jvm/default-java

source ~/.bashrc
```

Install whatever version seems to be compatible with the tasks that needs to be performed. However here, OpenDayLight - Beryllium is installed first. Though less complicated than other versions, it takes lot of time to run and install any features. The version used here is OpenDayLight Oxygen.

- Go to the link, https://docs.opendaylight.org/en/latest/downloads.html and download the oxygen Tar file.
- Decompress the file using *tar -xvf*
- Since OpenDayLight is packages in a karaf container, the karaf container should be started.

```
cd ~/karaf-0.8.4
./bin/karaf
```

Controller starts as shown and the ODL terminal appears.

- *feature:list* lists all the features that were already installed in OpenDayLight.
- *feature:install* - Installs the required features. To establish paths among switches and hosts which is the first step to check the connection between mininet and ODL, it is suggested to install all the features as in the provided sequence, else, there are chances of conflicts.
- To view ODL GUI in oxygen, *odl-dlux-all* is deprecated. *odl-dlux* is modified to *odl-dluxapps* with various features out of which necessary features are to be selected.

  odl-dlux-core
  odl-dluxapps-nodes
  odl-dluxapps-topology
  odl-dluxapps-yangui
  odl-dluxapps-yangvisualizer
  odl-dluxapps-yangman
- The installation of the features should be done with attention since there might be conflicts between two or more features. It's better to restart ODL after installing any of the features.
- Here, *odl-dluxapps-topology and odl-dluxapps-nodes* are installed for GUI.

  feature:install odl-dluxapps-topology odl-restconf-all odl-l2switch-switch

- All the installed features can be seen using the command
  feature:list —installed
- Exit ODL by *system:shutdown* or *ctrl+d*

***2.* Testing the Connectivity**

- Find out the ip by running ifconfig in your CLI terminal and the inet addr will be your system ip.
- Run the following command from your mininet terminal.

sudo mn —topo linear, 4 —mac —controller=remote,ip =<your ip from ifconfig>, port=6633 —switch ovs,protocols=OpenFlow13

- Since l2switch is installed as one of the feature, ping functionality should work by displaying connections among all the nodes.

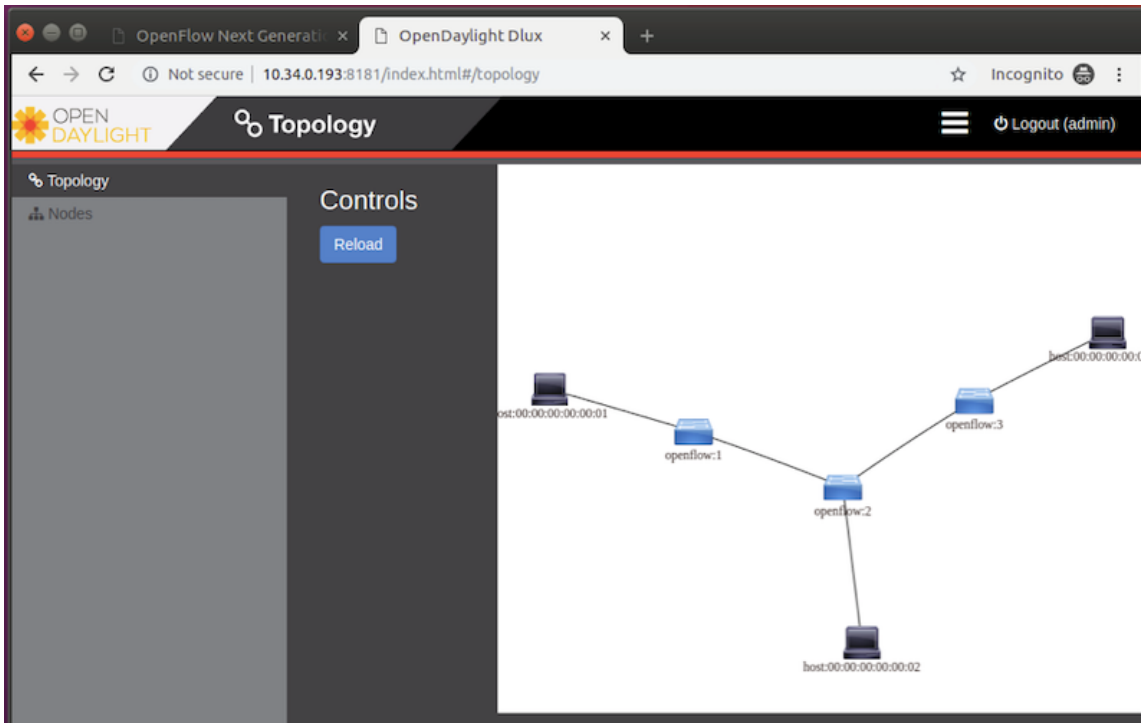
mininet > pingall

The following output has to be displayed as shown.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

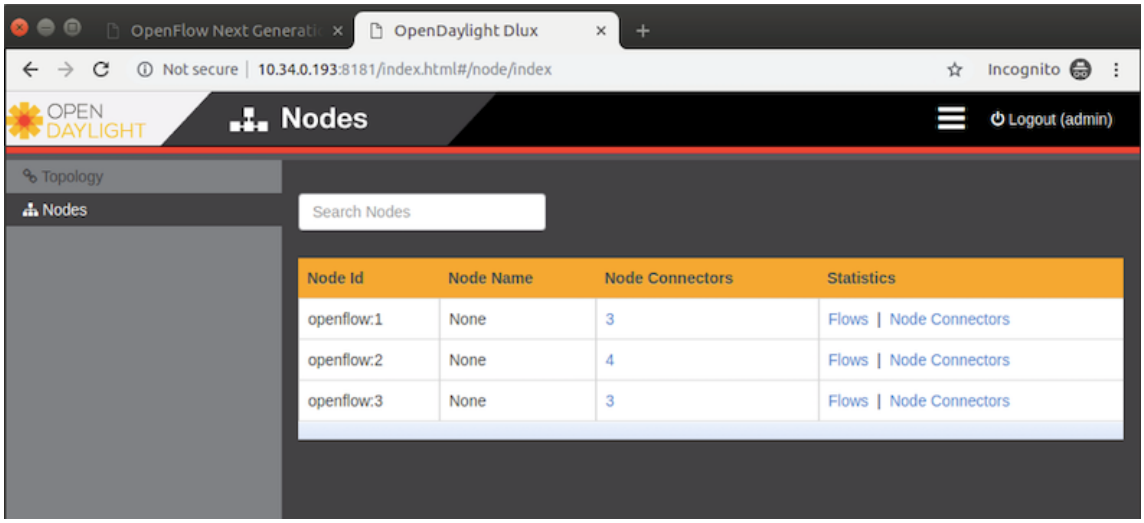
GUI

Open a browser and enter the URL http://<ip>:8181/index.html.
The credentials for the site is admin/admin.



The nodes for each switch can be visualised by running the following command in ODL terminal.

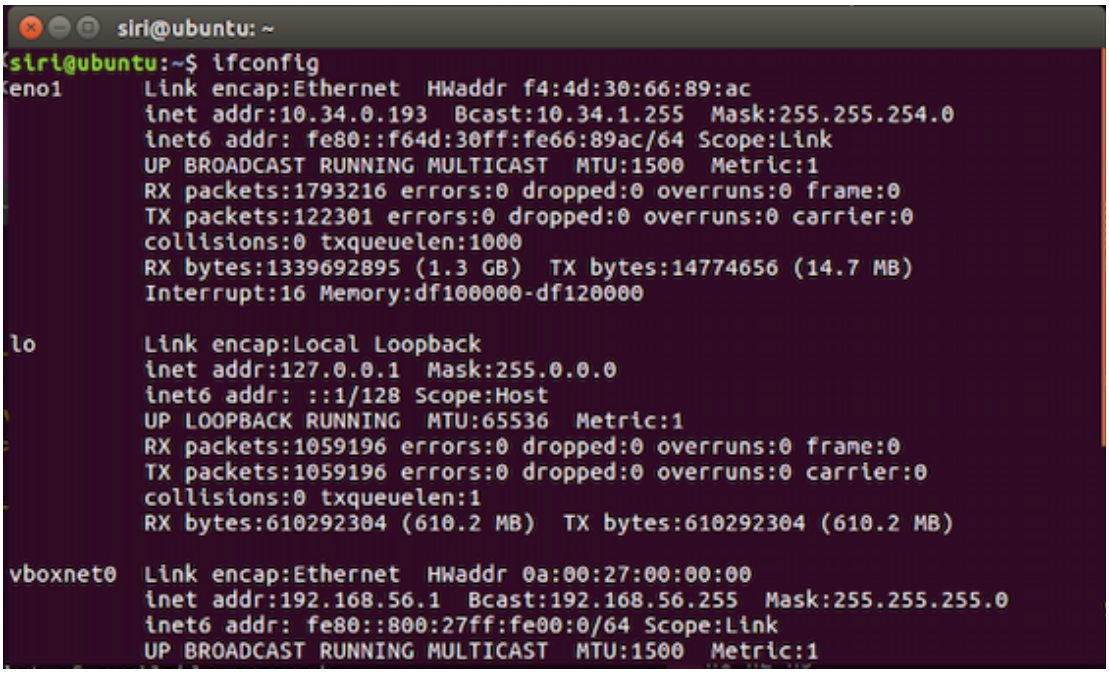
feature:install odl-dluxapps-nodes

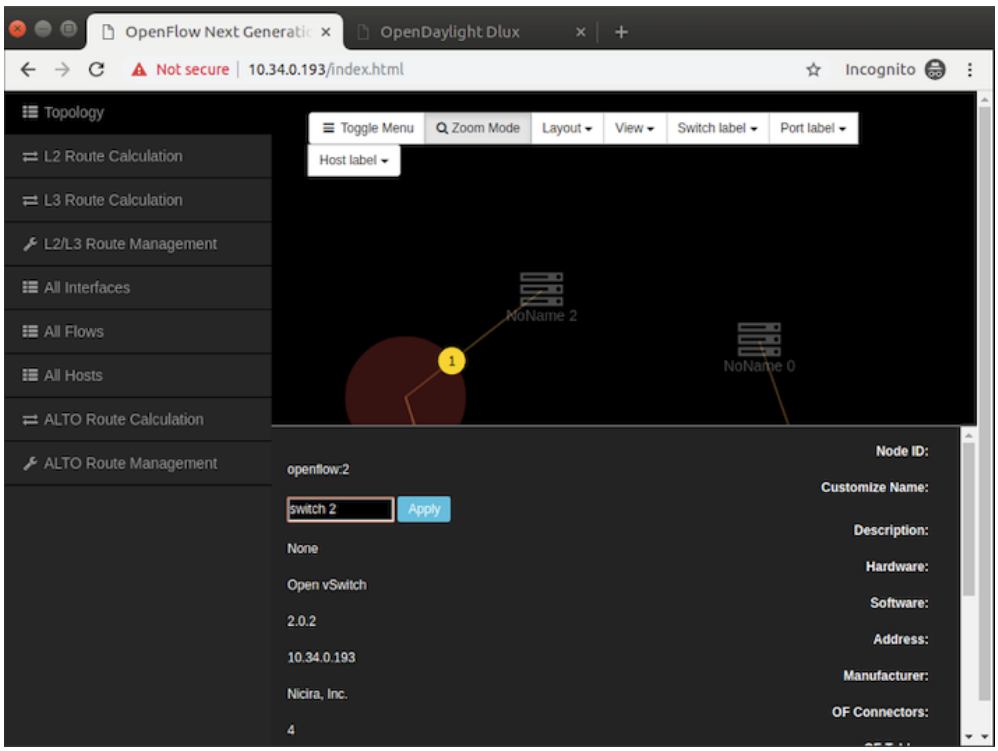


Here we are using an Open repo where the Route calculations and managements as well as hosts, interface and forwarding paths are properly configured. Hence, https://github.com/fno2010/web-ui repo is extracted. To run the GUI in the browser, INSTALL.debian.md in ubuntu. The browser points to http://<ip>.

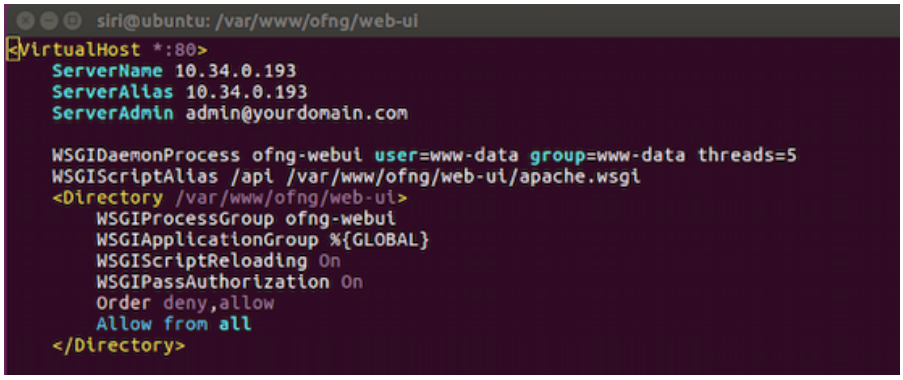
If the ip does not points to the browser, the ip has to be changed in apache configuration files.

sudo vi /var/www/ofng/web-ui/apache.conf

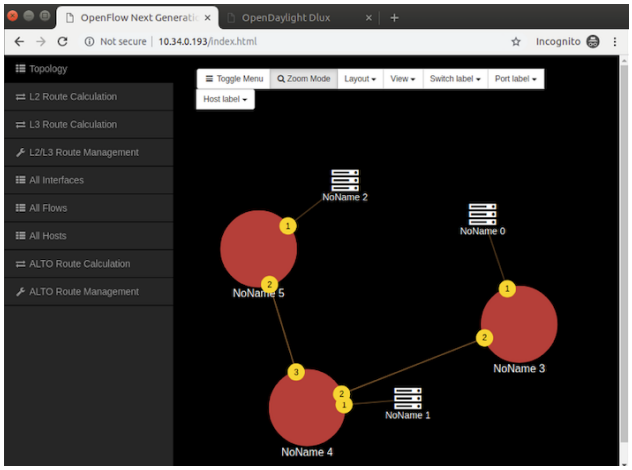




Since the ip is 10.34.0.193 and hence the conf file is changed as follows.



The credentials are guest/guest. The browser opens http://10.34.0.193/index.html as follows.



The names of the switches can be changed by clicking on the switches.

3. Pinging hosts using GUI.
Here the task is to disable ping feature and enable connections manually to enable paths between hosts and enable route calculation.

- To disable the ping feature, shutdown ODL terminal.
  cd karaf-0.8.4
  ./bin/karaf clean (Clean removes all the installed features in karaf)
  opendaylight-user@root > *feature:install odl-dluxapps-topology odl-dluxapps-nodes odl-restconf-all odl-openflowplugin-southbound odl-openflowplugin-flow-services odl-openflowplugin-flow-services-rest*

  feature:install odl-l2switch-arphandler

- Exit the opendaylight terminal. Create the following file and make changes in l2switch to restrict the flow of packets across hosts.(If the following path or the directory is not present according to the versions, create one)

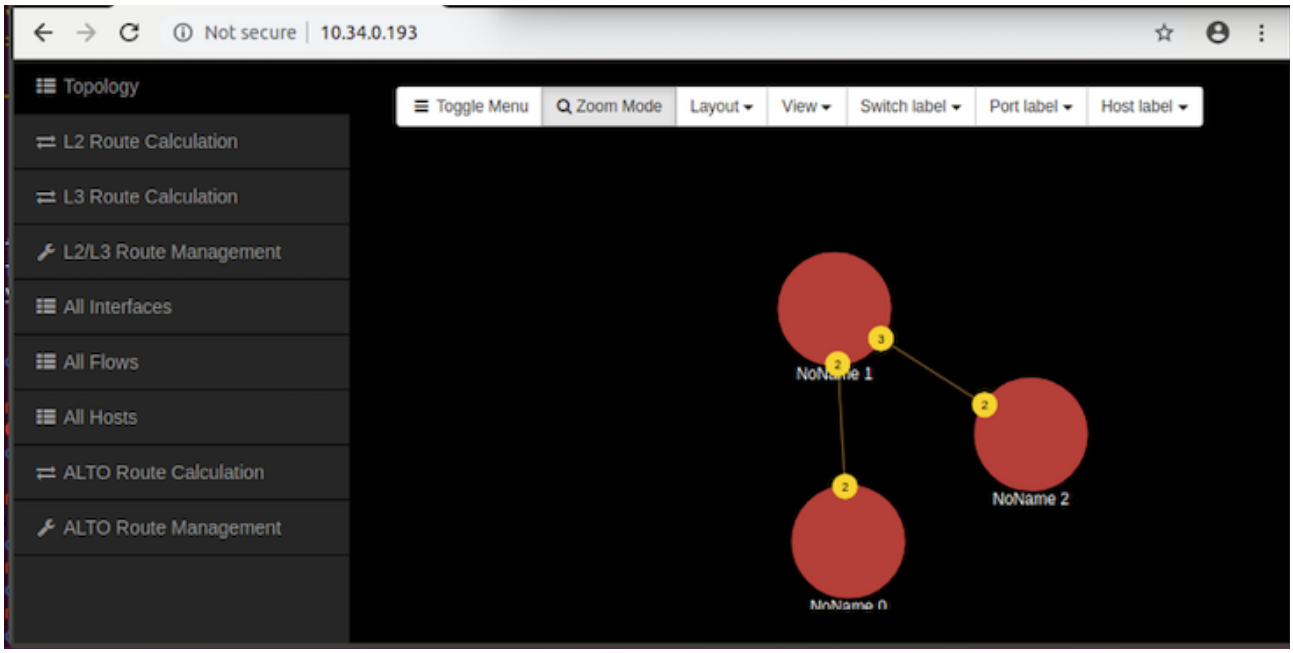
  cd /home/siri/Downloads/karaf-0.8.4/etc/opendaylight/datastore/initial/config

- Create a file and add the following lines to it.
  sudo vi arp-handler-config_arp-handler-config.xml

  <?xml version="1.0" encoding="UTF-8"?>
  <arp-handler-config xmlns="urn:opendaylight:packet:arp-handler-config">
  <is-proactive-flood-mode>false</is-proactive-flood-mode>
  <is-learning-only-mode>true</is-learning-only-mode>
  </arp-handler-config>

- After installing any feature, restart ODL to reflect the changes
- Run the mininet commands to connect to ODL and ping the hosts.

```
mininet@mininet-vm: ~
mininet@mininet-vm:~$ sudo mn --topo linear,3 --mac --controller=remo
te,ip=10.34.0.193,port=6633 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
mininet>
```



- Connect the hosts using the L2Route Calculation and ping the hosts.

4. Python Scripts

Instead of writing default command for the mining, python scripts can be created to add and delete hosts, create links and label the switches as well as hosts.

Create a python file at *~/mininet/custom/example.py* in mining terminal.

sudo mn --custom ~/mininet/custom/example.py --topo=mytopo --controller=remote,ip=192.168.56.101,port=6633

Example python file to create 3 switches and 3 hosts.

```
class MyTopo( Topo ):
   “3-switch example."

   def __init__( self ):
       Topo.__init__( self )

       # Add hosts and switches
       h1 = self.addHost( 'h1' )
       h2 = self.addHost( 'h2' )
       h3 = self.addHost( 'h3' )
       s1 = self.addSwitch( 's1' )
       s2 = self.addSwitch( 's2' )
       s3 = self.addSwitch( 's3' )

       # Add links
       self.addLink( s1, s2)
       self.addLink( s1, s3)
       self.addLink( s1, h1)
       self.addLink( s2, h2)
       self.addLink( s3, h3)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Note: Complicated multi graphs and trees with numerous and varied links can be constructed using python scripts.

5. TroubleShooting:

- Restart opendaylight as well as mininet if there are conflicts and errors in execution. Sometimes restarting the OS might help.
- The above commands may vary depends on the ODL version.
- :Shutdown is used to stop ODL.
- The credentials for the ITS systems are - primary OS : siri/1234 , VM : mininet/mininet.

Main References:

- https://docs.opendaylight.org/en/stable-fluorine/downloads.html
- https://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/
- https://docs.opendaylight.org/en/stable-oxygen/user-guide/using-the-opendaylight-user-interface-(dlux).html
- https://github.com/fno2010/web-ui
- https://docs.opendaylight.org/en/stable-fluorine/user-guide/l2switch-user-guide.html
- https://ask.opendaylight.org/question/11862/how-to-work-with-config-flows-only/
- https://lists.opendaylight.org/pipermail/l2switch-dev/2015-November/000939.html
- https://wiki.opendaylight.org/view/OpenDaylight_Virtual_Tenant_Network_(VTN):Scripts:Mininet
- https://sreeninet.wordpress.com/2013/12/25/tools-used-with-opendaylight/