sudo apt update sudo apt-get install curl sudo apt install default-jre sudo apt install default-jdk

sudo apt-get install -y openvswitch-switch sudo apt-get install -y mininet sudo apt-get install -y libxml2-utils

wget https://nexus.opendaylight.org/content/repositories/opendaylight.release/org

/opendaylight/integration/opendaylight/15.3.0/opendaylight-15.3.0.tar.gz

tar xvzf opendaylight-15.3.0.tar.gz

cd opendaylight-15.3.0/

./bin/karaf

feature:install odl-mdsal-apidocs

feature:install odl-restconf

feature:install odl-openflowplugin-flow-services-rest

feature:install odl-openflowplugin-app-table-miss-enforcer

feature:install odl-openflowplugin-app-topology

feature:install odl-openflowplugin-app-topology-manager

feature:install odl-openflowplugin-app-lldp-speaker

feature:install odl-openflowplugin-app-topology-lldp-discovery

sudo apt-get install ansible git aptitude

git clone <a href="https://github.com/containernet/containernet.git">https://github.com/containernet/containernet.git</a>

cd containernet/ansible/

sudo ansible-playbook -i "localhost," -c local install.yml sudo apt-get install openvswitch-switch

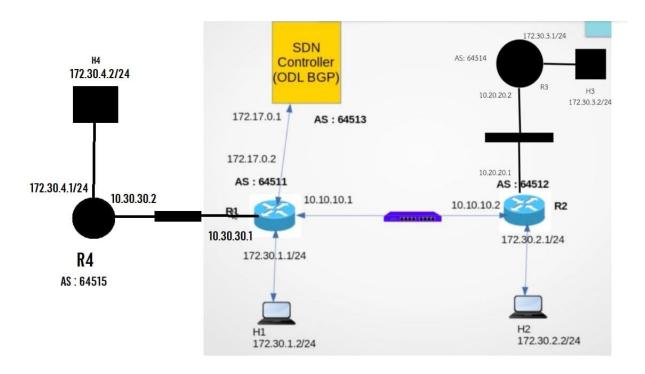
® หากติดปัญหา
 sudo docker ps -a
 sudo docker stop mn.r1
 sudo docker rm mn.r1
 sudo service bird restart

#### Create the ODL BGL Instance

curl -v --user "admin": "admin" -H "Accept: application/xml" -H "Content-Type: application/xml" -X POST http://localhost:8181/restconf/config/openconfig@network-instance:network-instances/network-instance/global-bgp/openconfig@network-instance:protocols/-d @bgp\_router.xml

### Create bgp neighbor

curl -v --user "admin": "admin" -H "Accept: application/xml" -H "Content-Type: application/xml" -X POST http://localhost:8181/restconf/config/openconfig@network-instance:network-instances/network-instance/global-bgp/openconfig@network-instance:protocols/protocol/openconfig-policy-types:BGP/bgp-odl@router/bgp/neighbors/-d @bgp\_neighbor.xml



# topo1.py

```
SDN\_ODL > odlbgp > topo\_TEST >  topo1.py >  topology
          from mininet.net import Containernet
          from mininet.node import RemoteController, Docker, OVSSwitch
   3
          from mininet.cli import CLI
          from mininet.log import setLogLevel, info
         from mininet.link import TCLink, Link
   6
   7
         # topo diagram
   8
         # h1----h2
   9
  10
  11
         def topology():
  12
  13
                 "Create a network with some docker containers acting as hosts."
  14
  15
                 net = Containernet(controller=RemoteController)
  16
  17
                 info('*** Adding switch\n')
  18
  19
                 #add r
                r1 = net.addDocker('r1', ip='172.30.1.1/24', dimage="knet/urouter:1.4")
r2 = net.addDocker('r2', ip='172.30.2.1/24', dimage="knet/urouter:1.4")
r3 = net.addDocker('r3', ip='172.30.3.1/24', dimage="knet/urouter:1.4")
r4 = net.addDocker('r4', ip='172.30.4.1/24', dimage="knet/urouter:1.4")
  20
  21
  22
  23
  24
  25
                 #add h
                h1 = net.addDocker('h1', ip='172.30.1.2/24', defaultRoute='via 172.30.1.1', dimage="knet/host-ubuntu:1-2") h2 = net.addDocker('h2', ip='172.30.2.2/24', defaultRoute='via 172.30.2.1', dimage="knet/host-ubuntu:1-2") h3 = net.addDocker('h3', ip='172.30.3.2/24', defaultRoute='via 172.30.3.1', dimage="knet/host-ubuntu:1-2") h4 = net.addDocker('h4', ip='172.30.4.2/24', defaultRoute='via 172.30.4.1', dimage="knet/host-ubuntu:1-2")
  26
  27
  28
  29
  30
  31
                s3 = net.addSwitch('s3', failMode='standalone')
s4 = net.addSwitch('s4', failMode='standalone')
s5 = net.addSwitch('s5', failMode='standalone')
  32
  33
  34
  35
  36
                 info('*** Creating links\n')
  37
  38
                 #add h - r
                 net.addLink(h1, r1)
  39
  40
                 net.addLink(h2, r2)
                 net.addLink(h3, r3)
  41
  42
                 net.addLink(h4, r4)
  43
```

```
44
        #add r - sw
         net.addLink(r1, s3, params1={"ip": "10.10.10.1/24"})
45
        net.addLink(r2, s3, params1={"ip": "10.10.10.2/24"})
46
47
        net.addLink(r2, s4, params1={"ip": "10.20.20.1/24"})
48
        net.addLink(r3, s4, params1={"ip": "10.20.20.2/24"})
         net.addLink(r1, s5, params1={"ip": "10.30.30.1/24"})
49
50
         net.addLink(r4, s5, params1={"ip": "10.30.30.2/24"})
51
52
53
         info('*** Starting network\n')
54
        net.start()
55
56
         #copy the bird config files
         s3.cmd("sudo docker cp r1.conf mn.r1:/etc/bird.conf")
57
         s3.cmd("sudo docker cp r2.conf mn.r2:/etc/bird.conf")
58
         s3.cmd("sudo docker cp r3.conf mn.r3:/etc/bird.conf")
59
60
61
         s4.cmd("sudo docker cp r1.conf mn.r1:/etc/bird.conf")
62
         s4.cmd("sudo docker cp r2.conf mn.r2:/etc/bird.conf")
63
         s4.cmd("sudo docker cp r3.conf mn.r3:/etc/bird.conf")
64
         s5.cmd("sudo docker cp r1.conf mn.r1:/etc/bird.conf")
65
         s5.cmd("sudo docker cp r2.conf mn.r2:/etc/bird.conf")
66
67
         s5.cmd("sudo docker cp r3.conf mn.r3:/etc/bird.conf")
68
         s5.cmd("sudo docker cp r4.conf mn.r4:/etc/bird.conf")
69
70
        #add r
71
        r1.cmd("bird -c /etc/bird.conf")
72
        r2.cmd("bird -c /etc/bird.conf")
73
         r3.cmd("bird -c /etc/bird.conf")
         r4.cmd("bird -c /etc/bird.conf")
74
75
76
         info('*** Running CLI\n')
77
78
        CLI(net)
79
         info('*** Stopping network')
80
        net.stop()
81
    if __name__ == '__main__':
82
        setLogLevel('info')
83
84
        topology()
85
```

#### r1.conf

```
SDN_ODL > odlbgp > topo_TEST > * r1.conf
      log "/var/log/bird.log" all;
      debug protocols all
   4
      router id 10.10.10.1;
   5
       protocol direct {
           interface "*";
   6
   7
   8
   9
      protocol kernel {
  10
           learn;
  11
           scan time 20;
  12
           export all;
  13
           import all;
  14
  15
  16
  17
       protocol device {
  18
           scan time 10;
  19
  20
  21
  22
  23
      #BGP Configuration
  24
  25
      protocol bgp R2{
  26
               export all;
  27
               import all;
  28
               local as 64511;
  29
               neighbor 10.10.10.2 as 64512;
  30
  31
       protocol bgp R3{
  32
  33
               export all;
  34
               import all;
  35
               local as 64511;
  36
               neighbor 10.20.20.2 as 64514;
  37
  38
  39
       protocol bgp R4{
  40
               export all;
  41
               import all;
  42
               local as 64511;
  43
               neighbor 10.30.30.2 as 64515;
  44
  45
46 ∨ protocol bgp OD1{
47
               export all;
48
               import all;
49
               local as 64511;
               neighbor 172.17.0.1 as 64513;
50
51
52
```

```
SDN_ODL > odlbgp > topo_TEST > 2 r2.conf
      log "/var/log/bird.log"
                                 all;
  2
      debug protocols all
  3
      router id 10.10.10.2;
  5
      protocol direct {
          interface "*";
  6
  7
  8
  9
 10
      protocol kernel {
 11
          learn;
          scan time 20;
 12
 13
          export all;
 14
          import all;
 15
 16
 17
 18
      protocol device {
 19
          scan time 10;
 20
 21
 22
 23
      #BGP Configuration
 24
 25
 26
      protocol bgp R1{
 27
               export all;
 28
               import all;
 29
               local as 64512;
 30
               neighbor 10.10.10.1 as 64511;
 31
 32
      protocol bgp R4{
 33
 34
               export all;
 35
               import all;
 36
               local as 64512;
 37
               neighbor 10.30.30.2 as 64515;
 38
 39
 40
      protocol bgp R3{
 41
               export all;
 42
               import all;
 43
               local as 64512;
 44
               neighbor 10.20.20.2 as 64514;
 45
```

#### r3.conf

```
SDN_ODL > odlbgp > topo_TEST > * r3.conf
      log "/var/log/bird.log"
                                 all;
  2
      debug protocols all
  3
      router id 10.20.20.2;
  4
  5
      protocol direct {
          interface "*";
  6
  7
  8
  9
      protocol kernel {
 10
 11
          learn;
          scan time 20;
 12
 13
          export all;
 14
          import all;
 15
 16
 17
 18
      protocol device {
 19
          scan time 10;
 20
 21
 22
 23
 24
      #BGP Configuration
 25
 26
      protocol bgp R2{
 27
               export all;
 28
               import all;
 29
               local as 64514;
 30
               neighbor 10.20.20.1 as 64512;
 31
 32
      protocol bgp R1{
 33
 34
               export all;
 35
               import all;
 36
               local as 64514;
 37
               neighbor 10.10.10.1 as 64511;
 38
 39
 40
      protocol bgp R4{
               export all;
 41
 42
               import all;
 43
               local as 64514;
               neighbor 10.30.30.2 as 64511;
 44
 45
```

#### r4.conf

```
SDN_ODL > odlbgp > topo_TEST > * r4.conf
      log "/var/log/bird.log"
    debug protocols all
  2
  3
  4
     router id 10.30.30.2;
  5 v protocol direct {
          interface "*";
  6
  7
  8
  9
 10 v protocol kernel {
 11
          learn;
 12
          scan time 20;
 13
          export all;
 14
          import all;
 15
 16
 17
 18 ∨ protocol device {
 19
          scan time 10;
 20
 21
 22
 23
 24
     #BGP Configuration
 25
 26 v protocol bgp R1{
 27
              export all;
 28
              import all;
 29
              local as 64515;
 30
              neighbor 10.30.30.1 as 64511;
 31
 32
 33 ∨ protocol bgp R2{
 34
              export all;
 35
              import all;
 36
              local as 64515;
 37
              neighbor 10.10.10.2 as 64512;
 38
 39
 40 ∨ protocol bgp R3{
 41
              export all;
 42
              import all;
 43
              local as 64515;
 44
              neighbor 10.20.20.2 as 64514;
 45
```

## bgp\_router.xml

```
1 v protocol xmlns="http://openconfig.net/yang/network-instance">
          <name>bgp-odl-router</name>
          <identifier xmlns:x="http://openconfig.net/yang/policy-types">x:BGP</identifier>
<bgp xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
  3
  4 ~
  5 ~
               <global>
  6 ~
                       <router-id>172.17.0.1</router-id>
  8
                       <as>64513</as>
  0
                   </config>
 10 ~
                   <afi-safis>
                       <afi-safi>
 11 ~
                            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-name>
 12
                       </afi-safi>
 13
 14 ~
                       <afi-safi>
 15
                            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST</afi-safi-name>
 16
                        </afi-safi>
 17
                   </afi-safis>
 18
               </global>
          </bgp>
 19
 20
      </protocol>
```

# bgp neighbor.xml

```
<meighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
         <neighbor-address>172.17.0.2</neighbor-address>
3
        <timers>
             <config>
                <hold-time>90</hold-time>
6
                 <connect-retry>10</connect-retry>
             </config>
8
         </timers>
        <transport>
10
             <config>
                 <remote-port>179</remote-port>
11
                 <passive-mode>false</passive-mode>
12
13
             </config>
14
        </transport>
15
        <config>
            <peer-type>EXTERNAL</peer-type>
16
             <peer-as>64511</peer-as>
18
         </config>
19
         <afi-safis>
             <afi-safi>
21
                <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-name>
             </afi-safi>
22
23
             <afi-safi>
                 <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST</afi-safi-name>
25
             </afi-safi>
         </afi-safis>
26
     </neighbor>
```

#### **RED BLUE GREEN**

#สร้าง host ชื่อ red

sudo ip netns add red

sudo ip link add red-veth0 type veth peer name red-veth1

sudo ip link set red-veth1 netns red

sudo ip netns exec red ip addr add 10.10.20.1/24 dev red-veth1

sudo ip netns exec red ip link set red-veth1 up

#สร้าง host ชื่อ blue

sudo ip netns add blue

sudo ip link add blue-veth0 type veth peer name blue-veth1

sudo ip link set blue-veth1 netns blue

sudo ip netns exec blue ip addr add 10.10.20.2/24 dev blue-veth1

sudo ip netns exec blue ip link set blue-veth1 up

#สร้าง host ชื่อ green

sudo ip netns add green

sudo ip link add green-veth0 type veth peer name green-veth1

sudo ip link set green-veth1 netns green

sudo ip netns exec green ip addr add 10.10.20.3/24 dev green-veth1

sudo ip netns exec green ip link set green-veth1 up

#open port

sudo ifconfig red-veth0 up sudo ifconfig blue-veth0 up sudo ifconfig green-veth0 up

sudo ovs-vsctl add-br s1

# เพิ่ม port red-vethe ลงใน bridge s1 sudo ovs-vsctl add-port s1 red-veth0

# เพิ่ม port blue-vethe ลงใน bridge s1 sudo ovs-vsctl add-port s1 blue-veth0

# เพิ่ม port green-vethe ลงใน bridge s1 sudo ovs-vsctl add-port s1 green-veth0

sudo ovs-ofctl -O OpenFlow13 add-flow s1 actions=normal

# chack ping with ip port-name sudo ip netns exec blue ping 10.10.20.3

DETORIC setup

sudo ip netns delete red sudo ip netns delete blue sudo ip netns delete green sudo ovs-vsctl del-br s1