```
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 https://github.com/containernet/containernet.git
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

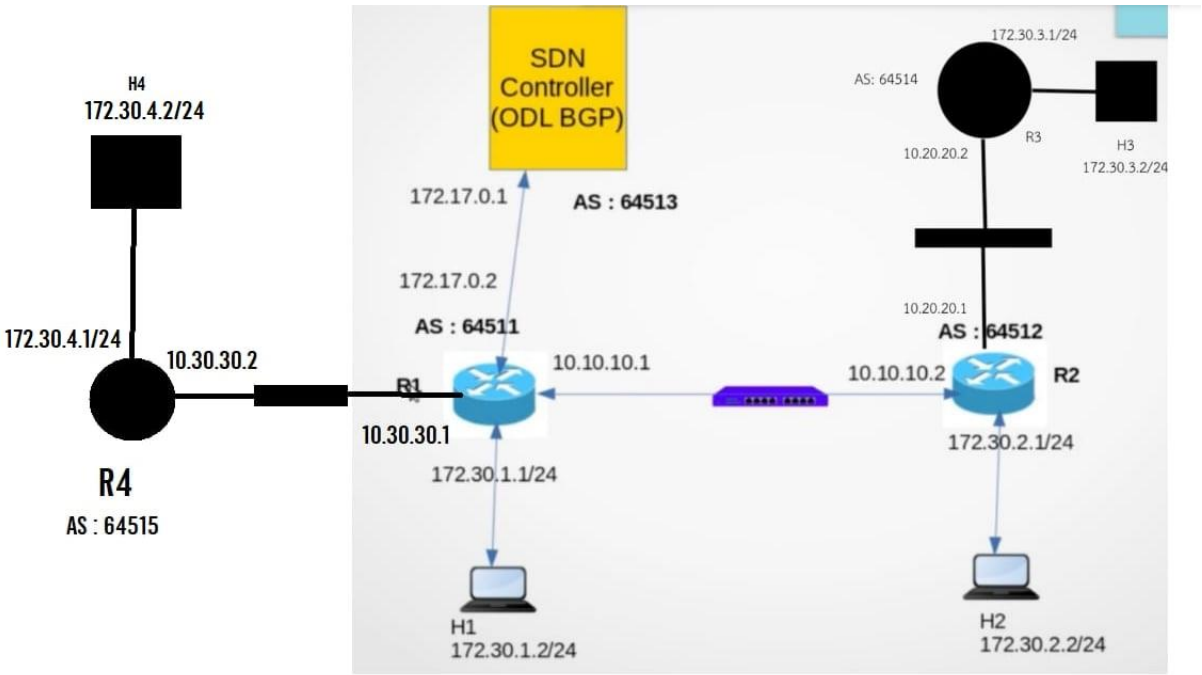
sudo service bird restart

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

```python
from mininet.net import Containernet
from mininet.node import RemoteController, Docker, OVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Link

# topo diagram
# h1----r1---s3---r2-----h2


def topology():

    "Create a network with some docker containers acting as hosts."

    net = Containernet(controller=RemoteController)

    info('*** Adding switch\n')

    #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")

    #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")

    #add sw
    s3 = net.addSwitch('s3', failMode='standalone')
    s4 = net.addSwitch('s4', failMode='standalone')
    s5 = net.addSwitch('s5', failMode='standalone')

    info('*** Creating links\n')

    #add h - r
    net.addLink(h1, r1)
    net.addLink(h2, r2)
    net.addLink(h3, r3)
    net.addLink(h4, r4)

```

```python
    #add r - sw
    net.addLink(r1, s3, params1={"ip": "10.10.10.1/24"})
    net.addLink(r2, s3, params1={"ip": "10.10.10.2/24"})
    net.addLink(r2, s4, params1={"ip": "10.20.20.1/24"})
    net.addLink(r3, s4, params1={"ip": "10.20.20.2/24"})
    net.addLink(r1, s5, params1={"ip": "10.30.30.1/24"})
    net.addLink(r4, s5, params1={"ip": "10.30.30.2/24"})


    info('*** Starting network\n')
    net.start()

    #copy the bird config files
    s3.cmd("sudo docker cp r1.conf mn.r1:/etc/bird.conf")
    s3.cmd("sudo docker cp r2.conf mn.r2:/etc/bird.conf")
    s3.cmd("sudo docker cp r3.conf mn.r3:/etc/bird.conf")

    s4.cmd("sudo docker cp r1.conf mn.r1:/etc/bird.conf")
    s4.cmd("sudo docker cp r2.conf mn.r2:/etc/bird.conf")
    s4.cmd("sudo docker cp r3.conf mn.r3:/etc/bird.conf")

    s5.cmd("sudo docker cp r1.conf mn.r1:/etc/bird.conf")
    s5.cmd("sudo docker cp r2.conf mn.r2:/etc/bird.conf")
    s5.cmd("sudo docker cp r3.conf mn.r3:/etc/bird.conf")
    s5.cmd("sudo docker cp r4.conf mn.r4:/etc/bird.conf")

    #add r
    r1.cmd("bird -c /etc/bird.conf")
    r2.cmd("bird -c /etc/bird.conf")
    r3.cmd("bird -c /etc/bird.conf")
    r4.cmd("bird -c /etc/bird.conf")


    info('*** Running CLI\n')
    CLI(net)
    info('*** Stopping network')
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology()
```

r1.conf

SDN_ODL > odlbgp > topo_TEST > r1.conf

```
log "/var/log/bird.log"  all;
debug protocols all

router id 10.10.10.1;
protocol direct {
    interface "*";
}

protocol kernel {
    learn;
    scan time 20;
    export all;
    import all;
}


protocol device {
    scan time 10;
}



#BGP Configuration

protocol bgp R2{
        export all;
        import all;
        local as 64511;
        neighbor 10.10.10.2 as 64512;
}

protocol bgp R3{
        export all;
        import all;
        local as 64511;
        neighbor 10.20.20.2 as 64514;
}

protocol bgp R4{
        export all;
        import all;
        local as 64511;
        neighbor 10.30.30.2 as 64515;
}

protocol bgp ODl{
        export all;
        import all;
        local as 64511;
        neighbor 172.17.0.1 as 64513;
}

```

r2.conf

SDN_ODL > odlbgp > topo_TEST > r2.conf

```
log "/var/log/bird.log"  all;
debug protocols all

router id 10.10.10.2;
protocol direct {
    interface "*";
}


protocol kernel {
    learn;
    scan time 20;
    export all;
    import all;
}


protocol device {
    scan time 10;
}



#BGP Configuration

protocol bgp R1{
        export all;
        import all;
        local as 64512;
        neighbor 10.10.10.1 as 64511;
}

protocol bgp R4{
        export all;
        import all;
        local as 64512;
        neighbor 10.30.30.2 as 64515;
}

protocol bgp R3{
        export all;
        import all;
        local as 64512;
        neighbor 10.20.20.2 as 64514;
}
```

r3.conf

SDN_ODL > odlbgp > topo_TEST > r3.conf

```
log "/var/log/bird.log"  all;
debug protocols all

router id 10.20.20.2;
protocol direct {
    interface "*";
}


protocol kernel {
    learn;
    scan time 20;
    export all;
    import all;
}


protocol device {
    scan time 10;
}



#BGP Configuration

protocol bgp R2{
        export all;
        import all;
        local as 64514;
        neighbor 10.20.20.1 as 64512;
}

protocol bgp R1{
        export all;
        import all;
        local as 64514;
        neighbor 10.10.10.1 as 64511;
}

protocol bgp R4{
        export all;
        import all;
        local as 64514;
        neighbor 10.30.30.2 as 64511;
}
```

r4.conf

SDN_ODL > odlbgp > topo_TEST > r4.conf

```
log "/var/log/bird.log"  all;
debug protocols all

router id 10.30.30.2;
protocol direct {
    interface "*";
}


protocol kernel {
    learn;
    scan time 20;
    export all;
    import all;
}


protocol device {
    scan time 10;
}



#BGP Configuration

protocol bgp R1{
        export all;
        import all;
        local as 64515;
        neighbor 10.30.30.1 as 64511;
}

protocol bgp R2{
        export all;
        import all;
        local as 64515;
        neighbor 10.10.10.2 as 64512;
}

protocol bgp R3{
        export all;
        import all;
        local as 64515;
        neighbor 10.20.20.2 as 64514;
}
```

bgp_router.xml

SDN_ODL > odlbgp > topo_TEST > bgp_router.xml > protocol

```xml
<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">
        <global>
            <config>
                <router-id>172.17.0.1</router-id>
                <as>64513</as>
            </config>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-name>
                </afi-safi>
                <afi-safi>
                    <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST</afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</protocol>
```

bgp_neighbor.xml

```xml
<neighbor xmlns="urn:opendaylight:params:xml:ns:yang:bgp:openconfig-extensions">
    <neighbor-address>172.17.0.2</neighbor-address>
    <timers>
        <config>
            <hold-time>90</hold-time>
            <connect-retry>10</connect-retry>
        </config>
    </timers>
    <transport>
        <config>
            <remote-port>179</remote-port>
            <passive-mode>false</passive-mode>
        </config>
    </transport>
    <config>
        <peer-type>EXTERNAL</peer-type>
        <peer-as>64511</peer-as>
    </config>
    <afi-safis>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV4-UNICAST</afi-safi-name>
        </afi-safi>
        <afi-safi>
            <afi-safi-name xmlns:x="http://openconfig.net/yang/bgp-types">x:IPV6-UNICAST</afi-safi-name>
        </afi-safi>
    </afi-safis>
</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
```