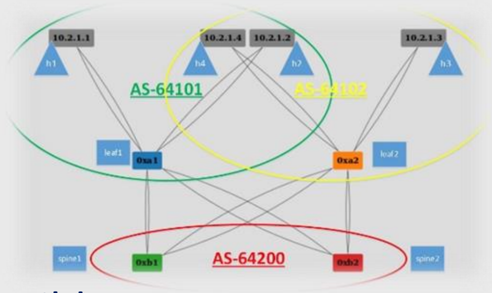


Custom topology guide

This guide was created to explain how to build a new topology for the INT over p4 system.

The default topology of the system is declared in the script int_ref_topology.py which runs the framework (p4factory/mininet/int_ref_topology.py).

We'll explain how to customize a topology step by step:

-Configure VXLAN tunnels.

-Configure the hosts.

-Configure the switches.

-Configure the links.

> All the configurations above must be under 'run_cgf' class (declared in int_ref_topology.py).

-Configure the routing according to BGP protocol

> The BGP routing declaration is defined in 2 different files (Specification below).

Before we start, there are a few important notes that can save a lot of time and efforts later:

a) Pay attention to white spaces – stick to the original structure in 'int_ref_topology.py'.

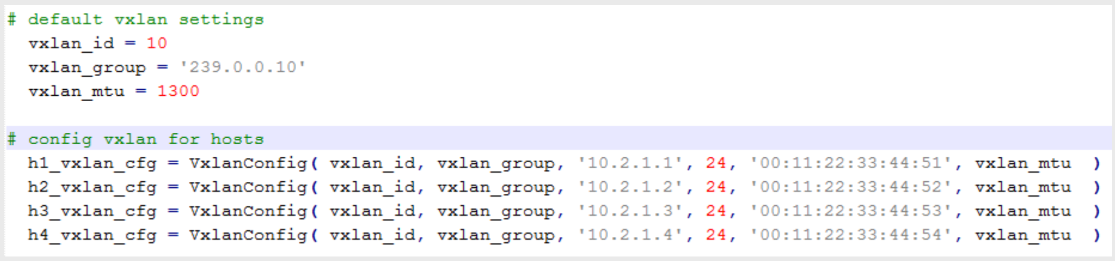
b) For each topology customizing, you should configure the BGP files accordingly.

c) The parameter 'model_dir' references the installations directory.

Customizing a new topology steps:

VXLAN Configuration:

For each host you should set VXLAN configurations. The tunneling is defined with vxlan_id and vxlan_group:

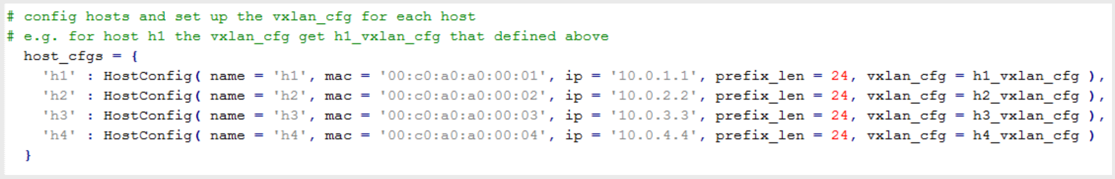


```
# default vxlan settings
  vxlan_id = 10
  vxlan_group = '239.0.0.10'
  vxlan_mtu = 1300

# config vxlan for hosts
  h1_vxlan_cfg = VxlanConfig( vxlan_id, vxlan_group, '10.2.1.1', 24, '00:11:22:33:44:51', vxlan_mtu  )
  h2_vxlan_cfg = VxlanConfig( vxlan_id, vxlan_group, '10.2.1.2', 24, '00:11:22:33:44:52', vxlan_mtu  )
  h3_vxlan_cfg = VxlanConfig( vxlan_id, vxlan_group, '10.2.1.3', 24, '00:11:22:33:44:53', vxlan_mtu  )
  h4_vxlan_cfg = VxlanConfig( vxlan_id, vxlan_group, '10.2.1.4', 24, '00:11:22:33:44:54', vxlan_mtu  )
```

Host Configuration:

For each host you should define its name, mac address, IP address and vxlan configuration:



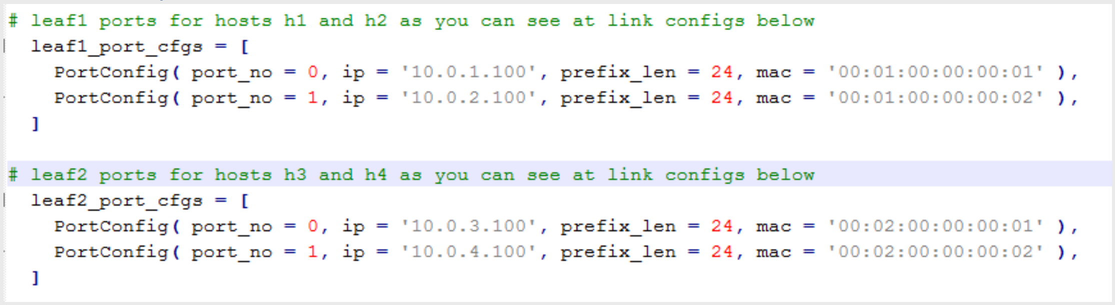
```
# config hosts and set up the vxlan_cfg for each host
# e.g. for host h1 the vxlan_cfg get h1_vxlan_cfg that defined above
  host_cfgs = {
    'h1' : HostConfig( name = 'h1', mac = '00:c0:a0:a0:00:01', ip = '10.0.1.1', prefix_len = 24, vxlan_cfg = h1_vxlan_cfg ),
    'h2' : HostConfig( name = 'h2', mac = '00:c0:a0:a0:00:02', ip = '10.0.2.2', prefix_len = 24, vxlan_cfg = h2_vxlan_cfg ),
    'h3' : HostConfig( name = 'h3', mac = '00:c0:a0:a0:00:03', ip = '10.0.3.3', prefix_len = 24, vxlan_cfg = h3_vxlan_cfg ),
    'h4' : HostConfig( name = 'h4', mac = '00:c0:a0:a0:00:04', ip = '10.0.4.4', prefix_len = 24, vxlan_cfg = h4_vxlan_cfg )
  }
```

Switch Configuration

-Ports for hosts:

In this section, the switching and the linking are declared.

For each switch, you should first define ports that designated for hosts. Each port declaration includes port number, IP address and mac address:



```
# leaf1 ports for hosts h1 and h2 as you can see at link configs below
  leaf1_port_cfgs = [
    PortConfig( port_no = 0, ip = '10.0.1.100', prefix_len = 24, mac = '00:01:00:00:00:01' ),
    PortConfig( port_no = 1, ip = '10.0.2.100', prefix_len = 24, mac = '00:01:00:00:00:02' ),
  ]

# leaf2 ports for hosts h3 and h4 as you can see at link configs below
  leaf2_port_cfgs = [
    PortConfig( port_no = 0, ip = '10.0.3.100', prefix_len = 24, mac = '00:02:00:00:00:01' ),
    PortConfig( port_no = 1, ip = '10.0.4.100', prefix_len = 24, mac = '00:02:00:00:00:02' ),
  ]
```

-<u>Switch decleration:</u>

NOTE: Each switch can be used as a leaf or a spine.

For each switch you should define:

a) name – The name of the switch.

b) port_cfgs - Ports for hosts array.

c) swapi_port - API port - note that the ports are different for each switch.

d) bmcli_port – CLI port - note that the ports are different for each switch.

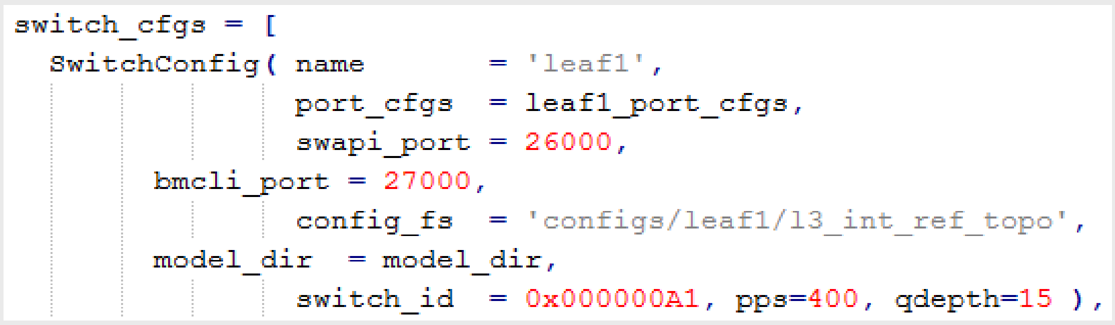
e) config_fs – The path for the BGP configurations directory.

f) model_dir – As described above.

g) switch_id - The value of the ID should be incremented for each switch. The first letter of the the leaves is 'A' and of the spines is 'B' as you can see bellow.

NOTE: switch_id value is an hex number(0x...).

Leaf1 and spine2 declaration:



```
switch_cfgs = [
  SwitchConfig( name        = 'leaf1',
                port_cfgs   = leaf1_port_cfgs,
                swapi_port  = 26000,
        bmcli_port = 27000,
                config_fs   = 'configs/leaf1/l3_int_ref_topo',
        model_dir   = model_dir,
                switch_id   = 0x000000A1, pps=400, qdepth=15 ),
SwitchConfig( name        = 'spine2',
              port_cfgs   = [],
              swapi_port = 26003,
              bmcli_port = 27003,
              config_fs   = 'configs/spine2/l3_int_ref_topo',
              model_dir   = model_dir,
              switch_id   = 0x000000B2, pps=400, qdepth=15 ),
```

Link Configuration:

For linking switch and host you should indicate only the switch port number.

For linking 2 switches you should indicate the designated port(index) for both.

```
link_cfgs = [
  LinkConfig( 'leaf1', 'h1', 0 ),
  LinkConfig( 'leaf1', 'h2', 1 ),
  LinkConfig( 'leaf1', 'spine1', 2, 0 ),
  LinkConfig( 'leaf1', 'spine2', 3, 0 ),

  LinkConfig( 'leaf2', 'h3', 0 ),
  LinkConfig( 'leaf2', 'h4', 1 ),
  LinkConfig( 'leaf2', 'spine1', 2, 1 ),
  LinkConfig( 'leaf2', 'spine2', 3, 1 ),
```

The linking of switch and host is executed by 'configHostRoutesAndArp' function, which in 'int_cfg.py' line 227.

```
def configHostRoutesAndArp(self):
  for l in self.link_cfgs:
    if l.port2 == None:
      sw = self.switch_cfgs[l.node1]
      h = self.net.get(l.node2)
      port = sw.port_cfgs[l.port1]
      h.cmd("route add default gw %s" % port.ip)
```

This function scans 'link_cfgs'.

For each link configuration with 3 parameters, 'linkConfig(switch, host, portIndex)', it links the switch to the host.

It uses the port which is defined 'port_cfgs' (ports for hosts) of the switch in index 'portIndex – switch.port[linkConfig.portIndex]'.

BGP Configuration:

The framework lays on BPG protocol.

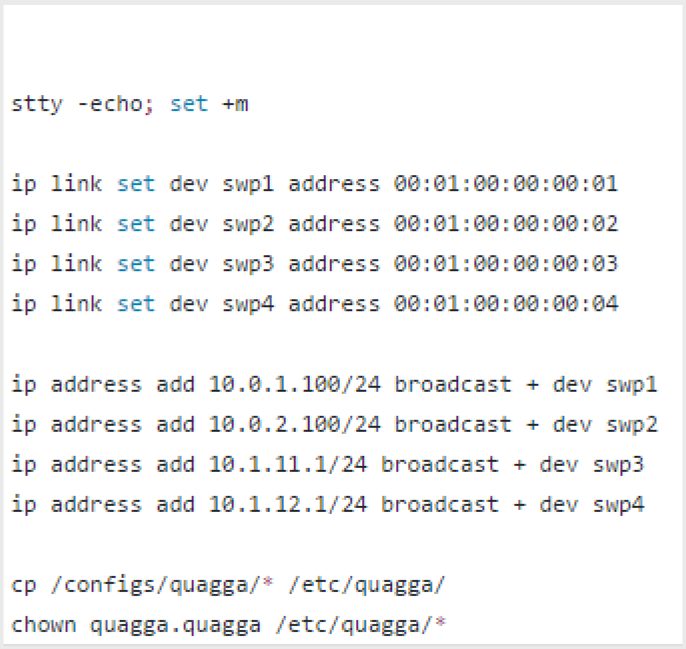
For each switch the switching is defined in its BGP folder.

The BGP folder contains 2 main files:

1. 'Startup_config' – Defines switch ports and IP addresses.

e.g. leaf1:

p4factory/mininet/configs/leaf1/l3_int_ref_topo/startup_config.sh



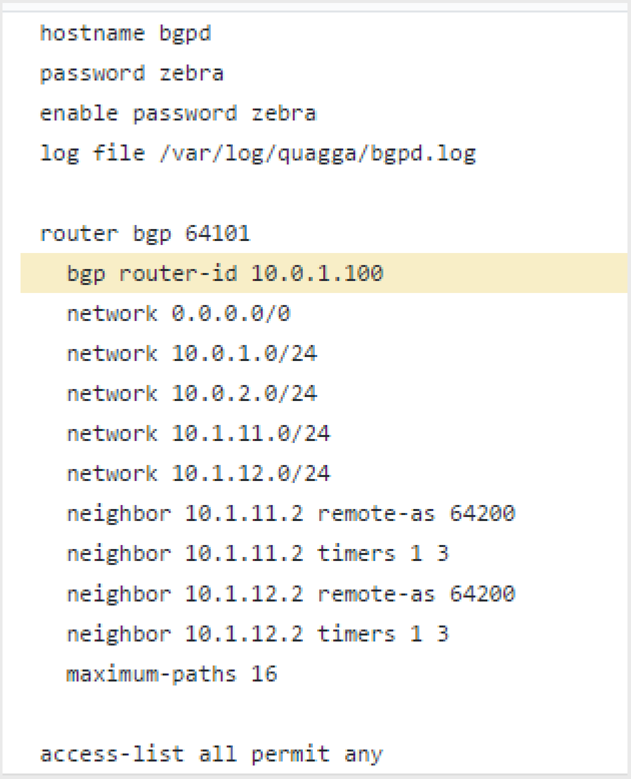
```
stty -echo; set +m

ip link set dev swp1 address 00:01:00:00:00:01
ip link set dev swp2 address 00:01:00:00:00:02
ip link set dev swp3 address 00:01:00:00:00:03
ip link set dev swp4 address 00:01:00:00:00:04

ip address add 10.0.1.100/24 broadcast + dev swp1
ip address add 10.0.2.100/24 broadcast + dev swp2
ip address add 10.1.11.1/24 broadcast + dev swp3
ip address add 10.1.12.1/24 broadcast + dev swp4

cp /configs/quagga/* /etc/quagga/
chown quagga.quagga /etc/quagga/*
```

2. 'bgp.conf' – AS name, ID in AS, networks to publish and as neighbors.
   p4factory/mininet/configs/leaf1/l3_int_ref_topo/quagga/bgpd.conf



```
hostname bgpd
password zebra
enable password zebra
log file /var/log/quagga/bgpd.log

router bgp 64101
  bgp router-id 10.0.1.100
  network 0.0.0.0/0
  network 10.0.1.0/24
  network 10.0.2.0/24
  network 10.1.11.0/24
  network 10.1.12.0/24
  neighbor 10.1.11.2 remote-as 64200
  neighbor 10.1.11.2 timers 1 3
  neighbor 10.1.12.2 remote-as 64200
  neighbor 10.1.12.2 timers 1 3
  maximum-paths 16

access-list all permit any
```

After finishing the guide you can run the framework with you new topology. If you want to see your topology on the web UI, you should run 'setTopo4UI.py' before you raise the framework.