Getting Started with Cumulus Linux

I – Introduction

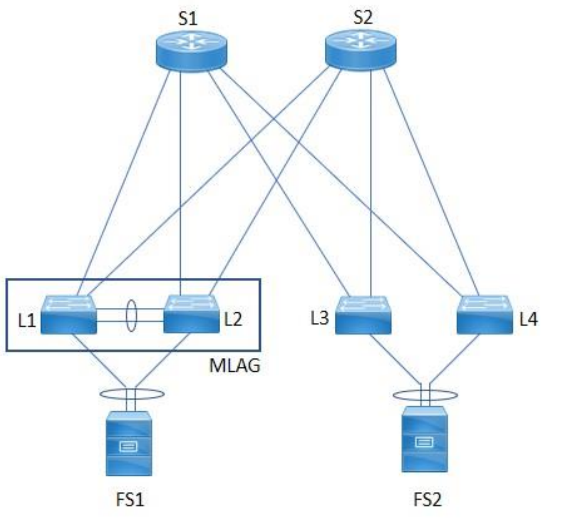
Since a while, Ethernet VPN is gaining in popularity. It suits well in datacenter environments and is now largely available on multiple platforms. One well-known fact about EVPN is that it carries L2 traffic over a « pure L3 » infrastructure. The magic of EVPN comes from its separation between Control and Data Planes. The Control Plane emulates the Ethernet flood-and-learn mechanisms with the use of MP-BGP. As for the Data Plane, VxLAN is probably the most popular type of encapsulation used in conjunction with EVPN Technologies.

Among all market players in EVPN technologies, Cumulus Networks (acquired by NVidia) provides a software solution based on a Debian distribution.

This paper describes a step-by-step deployment of a simple architecture for those who want to get started with EVPN on Cumulus.

II – Overview of the Setup

Cumulus Networks also provides a way to play scenario with its NOS through their Virtual Datacenter, Cumulus in the Cloud[1]. So you can replay, learn and explore things over there. Here is the setup:



So, two Spines four Leafs and two file servers.

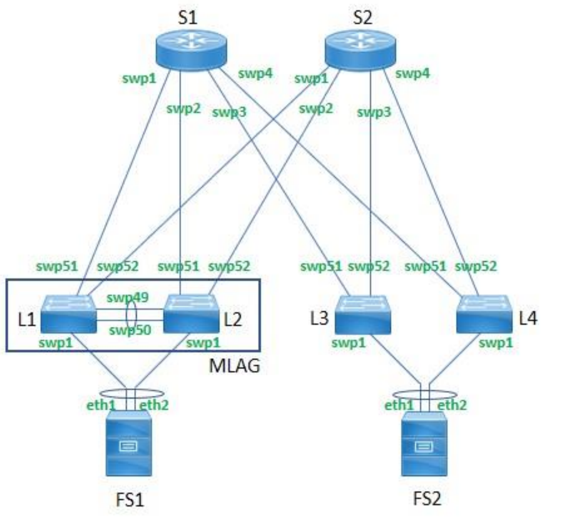
Two of the Leafs are configured to form a MLAG.

Other Leafs will be configured as standalone so that we can look closer at the EVPN Multihoming feature.

Objective of that lab is to build an EVPN architecture, configure a couple of overlays and open the heap of EVPN.

III – Physical Layout of the Setup

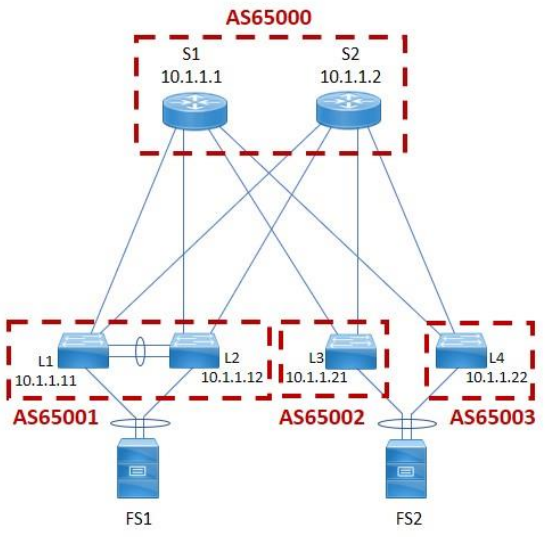
The physical layout of that virtual lab is as follow:



Please note that FS2 is server04.

Since we are dealing with p2p links between Spines & Leafs, there's no need to implement any IGP. Also, we will implement another great feature, the BGP unnumbered.

Here is our L3 map:



As you can see:

- both Spines belong to the same BGP AS65000,
- the MLAG belongs to AS65001,
- remaining Leafs get their own AS BGP 65002 & 65003.

IV – Getting Started with Cumulus Linux

That phase requires some basic Linux skills, but you'll see how easy it is to start. Clicking onto the devices opens an SSH session, the default credentials are:

```
Cumulus/CumulusLinux!
```

There's one file that needs our attention:

```
/etc/frr/daemons
```

because it contains all daemons related to FRRouting[2]. They are all disabled by default and for that lab, we'll need to enable and start `bgpd` & `pimd` daemons. The following commands will enable and load the relevant daemons:

```
sed -i s/bgpd=no/bgpd=yes/ /etc/frr/daemons
sed -i s/pimd=no/pimd=yes/ /etc/frr/daemons
systemctl enable frr.service
systemctl start frr.service
```

So just run theses commands onto the 6 devices be »fore going further.

V – Underlay

1) Configuring the MLAG

When you configure a MLAG on Cumulus Linux, the automatically created peerlink between members provides the VLAN sub interface `peerlink.4094`.

It's a kind of « service » sub interface to build and maintain the MLAG. We'll also use that sub interface to carry an iBGP session later.

The MLAG is configured with the following set of commands on L1:

```
#
# L1 - Loopback
#
net add loopback lo ip address 10.1.1.11/32
#
# L1 - Inter-chassis bond
#
net add bond peerlink bond slaves swp49,swp50
#
# L1 - MLAG
#
net add interface peerlink.4094 clag backup-ip 10.1.1.12
net add interface peerlink.4094 clag peer-ip linklocal
net add interface peerlink.4094 clag priority 1000
net add interface peerlink.4094 clag sys-mac
44:38:39:FF:00:01
net commit
```

```
#
# L2 - Loopback
#
net add loopback lo ip address 10.1.1.12/32
#
# L2 - Inter-chassis bond
#
net add bond peerlink bond slaves swp49,swp50
#
# L2 - MLAG
#
net add interface peerlink.4094 clag backup-ip 10.1.1.11
net add interface peerlink.4094 clag peer-ip linklocal
net add interface peerlink.4094 clag priority 2000
net add interface peerlink.4094 clag sys-mac
44:38:39:FF:00:01
net commit
```

Note the same MAC address on both members and the use of linklocal as the peer-ip. As that stage, the MLAG is up and running:

```
root@leaf01:mgmt:/home/cumulus# net show clag status
The peer is alive
     Our Priority, ID, and Role: 1000 44:38:39:00:00:59 primary
    Peer Priority, ID, and Role: 2000 44:38:39:00:00:5a secondary
          Peer Interface and IP: peerlink.4094 fe80::4638:39ff:fe00:5a
(linklocal)
                      Backup IP: 10.1.1.12 (inactive)
                     System MAC: 44:38:39:ff:00:01

root@leaf02:mgmt:/home/cumulus# net show clag status
The peer is alive
     Our Priority, ID, and Role: 2000 44:38:39:00:00:5a secondary
    Peer Priority, ID, and Role: 1000 44:38:39:00:00:59 primary
          Peer Interface and IP: peerlink.4094 fe80::4638:39ff:fe00:59
(linklocal)
                      Backup IP: 10.1.1.11 (inactive)
                     System MAC: 44:38:39:ff:00:01
```

2) Configuring Unnumbered BGP

Unnumbered BGP[3] is another great feature that you must use when dealing with such a topology. At that stage, what needs to be achieved first is to make sure all loopbacks are mutually IP-reachable.

Let's look closer at L1 configuration, we'll configure:

- eBGP sessions with S1 & S2,
- iBGP session with L2 through the peerlink,
- the IPv4 unicast address-family as propagating all connected networks.

```
#
# L1 - Global BGP configuration
#
net add bgp autonomous-system 65501
net add bgp router-id 10.1.1.11
net add bgp bestpath as-path multipath-relax
#
# L1 - eBGP
#
net add bgp neighbor SPINES peer-group
net add bgp neighbor SPINES remote-as external
net add bgp neighbor swp51 interface peer-group
SPINES
net add bgp neighbor swp52 interface peer-group
SPINES
#
# L1 - iBGP to L2
#
net add bgp neighbor peerlink.4094 interface
remote-as internal
#
# L1 - AF IPv4 unicast
#
net add bgp ipv4 unicast neighbor SPINES soft-
reconfiguration inbound
net add bgp ipv4 unicast  redistribute connected
net commit
```

Notice also the soft reconfiguration inbound always useful for troubleshooting and verification purposes.

The configuration of the Spines are similar, here is the configuration of S2 :

```
# S2 - Loopback
#
net add loopback lo ip address 10.1.1.2/32
#
# S2 - Global BGP configuration
#
net add bgp autonomous-system 65500
net add bgp router-id 10.1.1.2
net add bgp bestpath as-path multipath-relax
#
# S2 - eBGP
#
net add bgp neighbor LEAFS peer-group
net add bgp neighbor LEAFS remote-as external
net add bgp neighbor swp1 interface peer-group
LEAFS
net add bgp neighbor swp2 interface peer-group
LEAFS
net add bgp neighbor swp3 interface peer-group
LEAFS
net add bgp neighbor swp4 interface peer-group
LEAFS
#
# S2 - AF IPv4 unicast
#
```

```
net add bgp ipv4 unicast neighbor LEAFS soft-
reconfiguration inbound
net add bgp ipv4 unicast neighbor LEAFS allowas-in
origin
net add bgp ipv4 unicast  redistribute connected
net commit
```

Notice the `allowas-in` on the Spine configuration within the AF IPv4 unicast. Without this command, S2 will never learn S1's Loopback.

3) Verifying the Underlay

To perform some verifications, just enter the `vtysh` command (need to be sudo). It provides with the Cisco-like CLI.

A `show ip bgp summary` will show that:

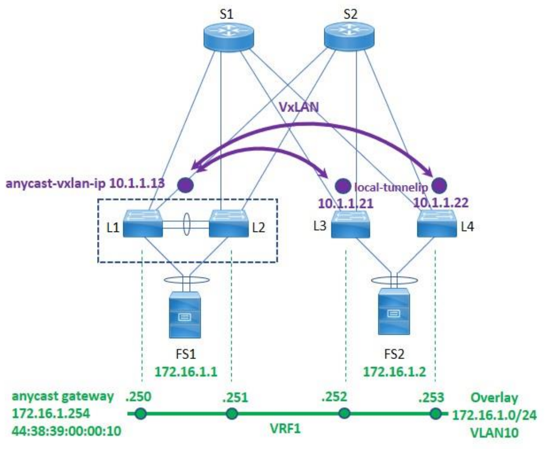
- every Spine peers with the four Leafs,
- every Leaf peers with the two Spines,
- members of the MLAG peer together.

At that point of the lab, any device should be able to ping any Loopback. If not, please review your configurations, it makes no sense to go further.

VI – Overlay

1) Introduction

We're almost ready to start playing with our very first overlay. At that point, let's clarify the big picture:



We'll create the IP overlay 172.16.1.0/24 in the vrf VRF1. FS1 & FS2 will belong to that subnet and we'll also configure the anycast gateway 172.16.1.254. In EVPN, the anycast gateway is like a VIP address in an HSRP/VRRP environment. Hence the need to configure SVI on all Leafs, that anycast gateway will be configured on top. The SVI will be the .250, .251, .252 & .253.

Second, the Data Plane will need to encapsulate native IP packets within VxLAN datagrams. Endpoints of those tunnels are usually the Loopbacks (configured as `vxlan local-tunnelip`). In case of a MLAG, we'll need to introduce an additional Loopback, the `clag vxlan-anycast-ip`. It obviously will need to be propagated within the underlay via BGP.

2) VxLAN tunnel endpoints

vrf1 & tunnel endpoints are configured as follow onto the MLAG members:

```
#
# L1 - VRF
#
net add vrf vrf1 vrf-table auto
#
# L1 - MLAG VTEP & propagation
#
net add loopback lo clag vxlan-anycast-ip 10.1.1.13
net add bgp ipv4 unicast network 10.1.1.13/32
#
# L1 - Local VTEP
#
net add loopback lo vxlan local-tunnelip 10.1.1.11
net commit


#
# L2 - VRF
#
net add vrf vrf1 vrf-table auto
#
# L2 - MLAG VTEP & propagation
#
net add loopback lo clag vxlan-anycast-ip 10.1.1.13
net add bgp ipv4 unicast network 10.1.1.13/32
#
# L2 - Local VTEP
#
net add loopback lo vxlan local-tunnelip 10.1.1.12
net commit
```

A quick check will show that 10.1.1.13/32 got correctly inserted into L3 & L4 RIB.

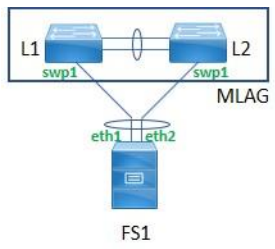
L3 & L4 just need the creation of vrf1 and the local-tunnelip :

```
#
# L3 - VRF
#
net add vrf vrf1 vrf-table auto
#
# L3 - Local VTEP
#
net add loopback lo vxlan local-tunnelip 10.1.1.21
net commit
```

3) Attaching FS1 & FS2

Configuration of Multihoming onto a MLAG or on a standalone box is slightly different.

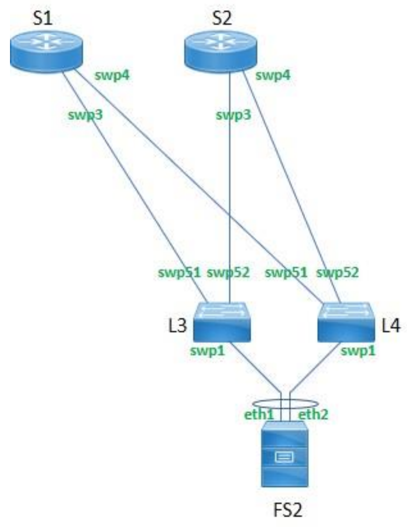
The MLAG scenario is easy to configure. This needs to assign the same clag id to the slave ports onto the members.



L1 & L2 need the following configuration:

```
#
# L1 - Bond creation FS1
#
net add bond fs1 bond slaves swp1
#
# L1 - clag Id assignment
#
net add bond fs1 clag id 1
net commit
```

The configuration onto standalone boxes needs more parameters:



```
#
# L3 - Bond creation FS2
#
net add bond fs2 bond slaves swp1
#
# L3 - EVPN MH parmeters
#
net add bond fs2 evpn mh es-df-pref 50000
net add bond fs2 evpn mh es-sys-mac
44:38:39:ff:ff:02
net add bond fs2 evpn mh es-id 2
#
# L3 - EVPN Uplinks
#
net add interface swp51 evpn mh uplink
net add interface swp52 evpn mh uplink
net commit


#
# L4 - Bond creation FS2
#
net add bond fs2 bond slaves swp1
#
# L4 - EVPN MH parmeters
#
net add bond fs2 evpn mh es-sys-mac
44:38:39:ff:ff:02
net add bond fs2 evpn mh es-id 2
#
# L3 - EVPN Uplinks
#
net add interface swp51 evpn mh uplink
net add interface swp52 evpn mh uplink
net commit
```

Notice the same `es-id` & the same `es-sys-mac`.
The `es-df-pref` parameter sets the Designated Forwarder.

4) Configuring FS1 & FS2

These are Ubuntu servers (server01 & server04). The bond interface is configured as follow on both servers:

```
#
# Bond interfaces
#
sudo ip link add bond0 type bond mode 802.3ad
sudo ip link set eth1 master bond0
sudo ip link set eth2 master bond0
```

As for the IP parameters, configure the IP/ask and the anycast gateway as the default:

```
#
# FS1- IP
#
ifconfig bond0 172.16.1.1 netmask 255.255.255.0
route add default gw 172.16.1.254


#
# FS2- IP
#
ifconfig bond0 172.16.1.2 netmask 255.255.255.0
route add default gw 172.16.1.254
```

5) VLAN & Bridge configuration

That part is a little bit tricky. It needs to define the VLAN 10, assign a VNI to it, and create the bridge and its ports into the NOS. The following need to be configured onto all the Leafs:

```
#
# VLAN 10 creation
#
net add vlan 10 vrf vrf1
net add vlan 10 vlan-id 10
net add vlan 10 vlan-raw-device bridge
#
# VxLAN assignment
#
net add vxlan vni10 vxlan id 100010
net add vxlan vni10 bridge access 10
#
# Bridge entry and bridge port
#
net add bridge bridge ports vni10
net add bridge bridge vids 10
#
# Disabling MAC Learning & ARP
#
net add vxlan vni10 bridge arp-nd-suppress on
net add vxlan vni10 bridge learning off
#
# Disabling STP
#
net add vxlan vni10 stp bpduguard
net add vxlan vni10 stp portbpdufilter
net commit
```

On Leafs L1 & L2, the bridge must include the inter-chassis peerlink and the bond fs1. Let's also configure the bond interfaces as access ports.

```
#
# L1 & L2 - Bridge entries peerlink, fs1
#
net add bridge bridge ports peerlink,fs1
net add bond fs1 bridge access 10
net commit
```

On Leafs L3 & L4, the bridge must include the bond fs2:

```
#
# L3 & L4 - Bridge entries peerlink, fs1
#
net add bridge bridge ports fs2
net add bond fs2 bridge access 10
net commit
```

6) SVI & Anycast Gateway

In that step, we configure the L3 SVI, the anycast gateway and we specify the Loopbacks as local tunnelip.

```
#
# L1
#
net add vxlan vni10 vxlan local-tunnelip 10.1.1.11
net add vlan 10 ip address 172.16.1.250/24
net add vlan 10 ip address-virtual 44:38:39:00:00:10 172.16.1.254/24
net commit

#
# L2
#
net add vxlan vni10 vxlan local-tunnelip 10.1.1.12
net add vlan 10 ip address 172.16.1.251/24
net add vlan 10 ip address-virtual 44:38:39:00:00:10 172.16.1.254/24
net commit

#
# L3
#
net add vxlan vni10 vxlan local-tunnelip 10.1.1.21
net add vlan 10 ip address 172.16.1.252/24
net add vlan 10 ip address-virtual 44:38:39:00:00:10 172.16.1.254/24
net commit

#
# L4
#
net add vxlan vni10 vxlan local-tunnelip 10.1.1.22
net add vlan 10 ip address 172.16.1.253/24
net add vlan 10 ip address-virtual 44:38:39:00:00:10 172.16.1.254/24
net commit
```

You may check FS1 has full IP connectivity with .250 & .251. As for FS2, by default, since the IP packets are not load-balanced, you should reach only one IP address. You may shut individual physical interfaces of its bond one by bond then check you have IP connectivity with .252 & .253.

7) Configuration of the AF L2VPN EVPN

We'll make configurations with the following assumptions:

- advertise the local SVI (anycast gateway included),
- advertise the VNI.

On all Leafs, here is the configuration :

```
#
# All Leafs - Enable AF L2VPN EVPN
#
net add bgp l2vpn evpn  neighbor SPINES activate
net add bgp l2vpn evpn  advertise-all-vni
net add bgp l2vpn evpn  advertise-svi-ip
net commit
```

Similar configuration is needed onto the Spines :

```
#
# All Leafs - Enable AF L2VPN EVPN
#
net add bgp l2vpn evpn  neighbor LEAFS activate
net add bgp l2vpn evpn  advertise-all-vni
net add bgp l2vpn evpn  advertise-svi-ip
net commit
```

At that point, if everything went well, FS1 & FS2 should be mutually IP reachable:

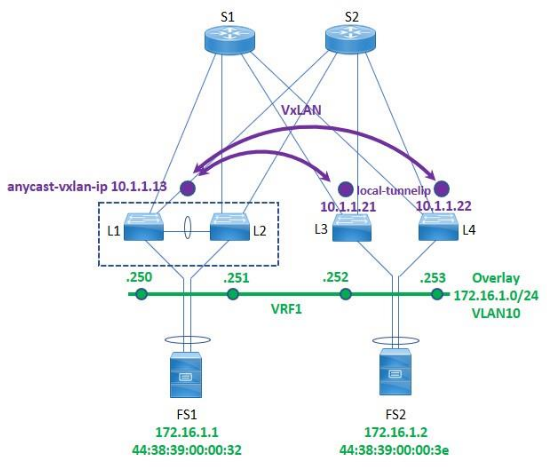
```
root@server01:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=64 time=15.6 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=64 time=4.91 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=64 time=5.15 ms
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 4.919/8.581/15.675/5.017 ms

root@server04:~# ping 172.16.1.1
PING 172.16.1.1 (172.16.1.1) 56(84) bytes of data.
64 bytes from 172.16.1.1: icmp_seq=1 ttl=64 time=5.65 ms
64 bytes from 172.16.1.1: icmp_seq=2 ttl=64 time=5.33 ms
64 bytes from 172.16.1.1: icmp_seq=3 ttl=64 time=4.05 ms
^C
--- 172.16.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 4.054/5.014/5.652/0.696 ms
```

VII – MP-BGP Route Types 1 & 2

We can now look closer at some MP-BGP Route Types[4]. Route Types 1, 2 & 3 are fundamental in EVPN.

As a reminder, here is the big picture:



1) Route Type 1

Remember that when we configured multihoming for FS2, we introduced the ESI with the command `es-sys-mac 44:38:39:ff:ff:02.` Route Type 1 are propagated by L3 & L4 indicating that 10.1.1.21 & 10.1.1.22 are the next hops to reach the ESI associated to FS244:38:39:ff:ff :02 :

```
root@leaf01:mgmt:/home/cumulus# net show bgp l2 ev route type 1
BGP table version is 28, local router ID is 10.1.1.11
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[ESI]:[EthTag]:[IPlen]:[VTEP-IP]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

   Network          Next Hop            Metric LocPrf Weight Path
                    Extended Community
Route Distinguisher: 10.1.1.21:2
*  [1]:[0]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.21                              0 65500
65502 i
                    RT:65502:100010 ET:8
*> [1]:[0]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.21                              0 65500
65502 i
                    RT:65502:100010 ET:8
Route Distinguisher: 10.1.1.21:3
*  [1]:[4294967295]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.21                              0 65500
65502 i
                    RT:65502:100010 ET:8 ESI-label-Rt:AA
*> [1]:[4294967295]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.21                              0 65500
65502 i
                    RT:65502:100010 ET:8 ESI-label-Rt:AA
Route Distinguisher: 10.1.1.22:2
*> [1]:[0]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.22                              0 65500
65503 i
                    RT:65503:100010 ET:8
*  [1]:[0]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.22                              0 65500
65503 i
                    RT:65503:100010 ET:8
Route Distinguisher: 10.1.1.22:3
*> [1]:[4294967295]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.22                              0 65500
65503 i
                    RT:65503:100010 ET:8 ESI-label-Rt:AA
*  [1]:[4294967295]:[03:44:38:39:ff:ff:02:00:00:02]:[32]:[0.0.0.0]
                    10.1.1.22                              0 65500
65503 i
                    RT:65503:100010 ET:8 ESI-label-Rt:AA
```

2) Route Type 2

From FS1, let's send ICMP to FS2:

```
root@server01:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=64 time=5.34 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=64 time=5.37 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=64 time=4.90 ms
64 bytes from 172.16.1.2: icmp_seq=4 ttl=64 time=5.02 ms
```

Obviously, looking at ICMP traffic onto L1 fs1 bond, we should see the ICMP Req/Rep:

```
root@leaf01:mgmt:/home/cumulus# tcpdump -ni fs1 icmp
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on fs1, link-type EN10MB (Ethernet), capture size
262144 bytes
10:46:00.374907  IP  172.16.1.1  >  172.16.1.2:  ICMP  echo
request, id 17991, seq 33, length 64
10:46:00.379134  IP  172.16.1.2  >  172.16.1.1:  ICMP  echo
reply, id 17991, seq 33, length 64
10:46:01.376822  IP  172.16.1.1  >  172.16.1.2:  ICMP  echo
request, id 17991, seq 34, length 64
10:46:01.395838  IP  172.16.1.2  >  172.16.1.1:  ICMP  echo
reply, id 17991, seq 34, length 64
10:46:02.378363  IP  172.16.1.1  >  172.16.1.2:  ICMP  echo
request, id 17991, seq 35, length 64
```

These ICMP packets get encapsulated, as we can see with the following tcpdump on L1:

```
root@leaf01:mgmt:/home/cumulus#  tcpdump  -ni  swp51  host
10.1.1.13
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on swp51, link-type EN10MB (Ethernet), capture
size 262144 bytes
10:55:14.254236 IP 10.1.1.13.52302 > 10.1.1.22.4789: VXLAN,
flags [I] (0x08), vni 100010
IP 172.16.1.1 > 172.16.1.2: ICMP echo request, id 17991,
seq 586, length 64
10:55:15.254531 IP 10.1.1.13.52302 > 10.1.1.22.4789: VXLAN,
flags [I] (0x08), vni 100010
IP 172.16.1.1 > 172.16.1.2: ICMP echo request, id 17991,
seq 587, length 64
10:55:16.256482 IP 10.1.1.13.52302 > 10.1.1.22.4789: VXLAN,
flags [I] (0x08), vni 100010
IP 172.16.1.1 > 172.16.1.2: ICMP echo request, id 17991,
seq 588, length 64
10:55:17.258417 IP 10.1.1.13.52302 > 10.1.1.22.4789: VXLAN,
flags [I] (0x08), vni 100010
IP 172.16.1.1 > 172.16.1.2: ICMP echo request, id 17991,
seq 589, length 64
```

We clearly see the ICMP packets encapsulated within a VxLAN datagram. We also recognize the VNI 100010 we previously configured. L1 knows where to send these packets based on the Route Type 2 messages sourced by L3 & L4:

```
root@leaf01:mgmt:/home/cumulus# net show bgp l2 ev route type 2

[...]

Route Distinguisher: 10.1.1.21:2
*> [2]:[0]:[48]:[44:38:39:00:00:3e]:[32]:[172.16.1.2]
                    10.1.1.21                              0 65500 65502 i
                    ESI:03:44:38:39:ff:ff:02:00:00:02 VNI: 0
                    RT:65502:100010 ET:8

[...]

Route Distinguisher: 10.1.1.22:2
*> [2]:[0]:[48]:[44:38:39:00:00:3e]:[32]:[172.16.1.2]
                    10.1.1.22                              0 65500 65503 i
                    ESI:03:44:38:39:ff:ff:02:00:00:02 VNI: 0
                    RT:65503:100010 ET:8

[...]
```

L3 & L4 got similar Route Type 2 messages from L1 & L2, ie propagating the MAC/IP of FS1 with the clag anycast-vxlan-ip 10.1.1.13 given as the endpoint of the tunnel:

```
*> [2]:[0]:[48]:[44:38:39:00:00:32]:[32]:[172.16.1.1]
                    10.1.1.13                              0 65500 65501 i
```

All of this is totally transparent to FS1 which just installed an entry within its ARP table:

```
root@server01:~# arp 172.16.1.2
Address           HWtype  HWaddress
172.16.1.2        ether   44:38:39:00:00:3e
```

3) Route Type 3

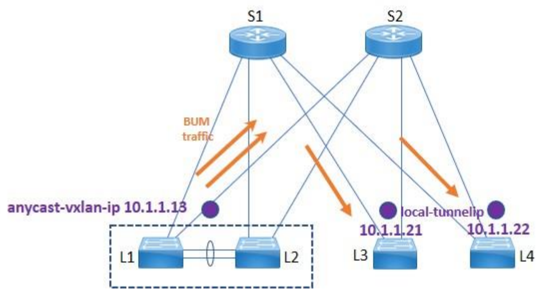
Leaf1 Routes Type 3 can be displayed with the following command:

```
root@leaf01:mgmt:/home/cumulus# net show bgp l2 ev route
type 3
BGP table version is 34, local router ID is 10.1.1.11
Status codes: s suppressed, d damped, h history, * valid,
> best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[ESI]:[EthTag]:[IPlen]:[VTEP-IP]
EVPN                    type-2                    prefix:
[2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

   Network          Next Hop            Metric LocPrf Weight
Path
                    Extended Community
Route Distinguisher: 10.1.1.11:2
*> [3]:[0]:[32]:[10.1.1.13]
                    10.1.1.13                          32768
i
                    ET:8 RT:65501:100010
Route Distinguisher: 10.1.1.11:3
*> [3]:[0]:[32]:[10.1.1.13]
                    10.1.1.13                          32768
i
                    ET:8 RT:65501:100020
Route Distinguisher: 10.1.1.21:2
*> [3]:[0]:[32]:[10.1.1.21]
                    10.1.1.21                              0
65500 65502 i
                    RT:65502:100010 ET:8
*  [3]:[0]:[32]:[10.1.1.21]
                    10.1.1.21                              0
65500 65502 i
                    RT:65502:100010 ET:8
Route Distinguisher: 10.1.1.21:3
*> [3]:[0]:[32]:[10.1.1.21]
                    10.1.1.21                              0
65500 65502 i
                    RT:65502:100020 ET:8
*  [3]:[0]:[32]:[10.1.1.21]
                    10.1.1.21                              0
65500 65502 i
                    RT:65502:100020 ET:8
Route Distinguisher: 10.1.1.22:2
*  [3]:[0]:[32]:[10.1.1.22]
                    10.1.1.22                              0
65500 65503 i
                    RT:65503:100010 ET:8
*> [3]:[0]:[32]:[10.1.1.22]
                    10.1.1.22                              0
65500 65503 i
                    RT:65503:100010 ET:8
Route Distinguisher: 10.1.1.22:3
*  [3]:[0]:[32]:[10.1.1.22]
                    10.1.1.22                              0
65500 65503 i
                    RT:65503:100020 ET:8
*> [3]:[0]:[32]:[10.1.1.22]
                    10.1.1.22                              0
65500 65503 i
                    RT:65503:100020 ET:8

Displayed 6 prefixes (10 paths) (of requested type)
```

Routes Type 3 are used to handle BUM traffic (Broadcast Unicast Multicast). Cumulus Linux uses the ingress-replication or HREP (head-en-replication) mechanism[5] .



Should L1 need to propagate a BUM packet, it will send a copy of that packet to 10.1.1.21 & 10.1.1.22. As an example, from L1, let's ping an unknown address, 172.16.1.3:

```
root@leaf01:mgmt:/home/cumulus#  ip  vrf  exec  vrf1  ping
172.16.1.3 -I 172.16.1.250
PING  172.16.1.3  (172.16.1.3)  from  172.16.1.250  :  56(84)
bytes of data.2.16.1.250
From 172.16.1.250 icmp_seq=1 Destination Host Unreachable
From 172.16.1.250 icmp_seq=2 Destination Host Unreachable
From 172.16.1.250 icmp_seq=3 Destination Host Unreachable
```

Capturing traffic onto S1 will show the two packets:

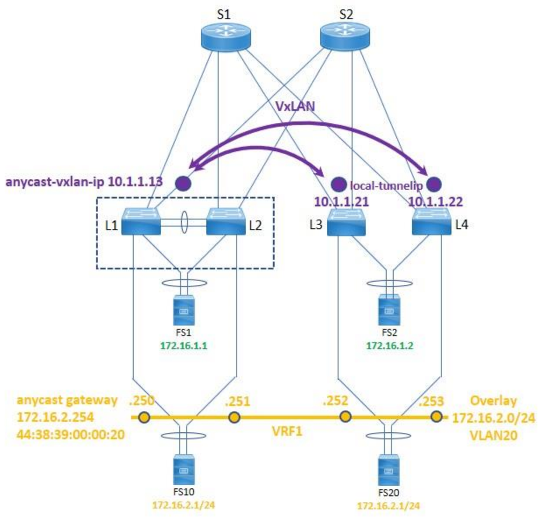
```
root@spine01:mgmt:/home/cumulus#  tcpdump  -ni  swp1  host
10.1.1.13
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening  on  swp1,  link-type  EN10MB  (Ethernet),  capture
size 262144 bytes

11:42:03.864341 IP 10.1.1.13.41105 > 10.1.1.21.4789: VXLAN,
flags [I] (0x08), vni 100010
ARP, Request who-has 172.16.1.3 tell 172.16.1.250, length
28
11:42:03.864461 IP 10.1.1.13.41105 > 10.1.1.22.4789: VXLAN,
flags [I] (0x08), vni 100010
ARP, Request who-has 172.16.1.3 tell 172.16.1.250, length
28
11:42:04.893276 IP 10.1.1.13.41105 > 10.1.1.21.4789: VXLAN,
flags [I] (0x08), vni 100010
ARP, Request who-has 172.16.1.3 tell 172.16.1.250, length
28
11:42:04.893447 IP 10.1.1.13.41105 > 10.1.1.22.4789: VXLAN,
flags [I] (0x08), vni 100010
ARP, Request who-has 172.16.1.3 tell 172.16.1.250, length
28
```

VIII – Inter-VxLAN Routing

1) Adding a new overlay

Adding an overlay in VRF1 is a very easy task. Let's configure a second overlay as depicted here:



We'll take server02 & server05 as FS10 & FS.
Also note that adding a second overlay doesn't require any additional configuration onto the Spines.

The configuration on L1 will be the following:

```
#
# L1 – FS10 bond interface
#
net add bond fs10 bond slaves swp2
net add bond fs10 bridge access 20
net add bond fs10 clag id 10
#
# L1 – Bridge entries
#
net add bridge bridge ports vni20,fs10
net add bridge bridge vids 20
#
# L1 – L3 SVI
#
net add vlan 20 ip address 172.16.2.250/24
net add vlan 20 vlan-id 20
net add vlan 20 vlan-raw-device bridge
net    add    vlan    20    ip    address-virtual
44:38:39:00:00:20 172.16.2.254/24
net add vlan 20 vrf vrf1
#
# L1 – VxLAN configuration
#
net add vxlan vni20 vxlan id 100020
net add vxlan vni20 bridge access 20
net add vxlan vni20 bridge arp-nd-suppress on
net add vxlan vni20 bridge learning off
net add vxlan vni20 stp bpduguard
net add vxlan vni20 stp portbpdufilter
net add vxlan vni20 vxlan local-tunnelip 10.1.1.11
net commit
```

As for L3, we'll introduce new EVPN parameters :

```
#
# L3 – FS20 bond interface & EVPN multihoming
#
net add bond fs10 bond slaves swp2
net add bond fs10 bridge access 20
net add bond fs10 evpn mh es-df-pref 50000
net add bond fs10 evpn mh es-id 3
net    add    bond    fs10    evpn    mh    es-sys-mac
44:38:39:ff:ff:03
#
# L3 – Bridge entries
#
net add bridge bridge ports vni10,fs10
net add bridge bridge vids 20
#
# L3 – L3 SVI
#
net add vlan 20 ip address 172.16.2.252/24
net add vlan 20 vlan-id 20
net add vlan 20 vlan-raw-device bridge
net add vlan 20 vrf vrf1
net    add    vlan    20    ip    address-virtual
44:38:39:00:00:20 172.16.2.254/24
#
# L3 – VxLAN configuration
#
net add vxlan vni20 vxlan id 100020
net add vxlan vni20 bridge access 20
net add vxlan vni20 bridge arp-nd-suppress on
net add vxlan vni20 bridge learning off
net add vxlan vni20 stp bpduguard
net add vxlan vni20 stp portbpdufilter
net add vxlan vni20 vxlan local-tunnelip 10.1.1.21
```

I let the reader to adapt the configurations onto L2 & L4.
When FS10 & FS20 have been configured, they should be mutually reachable:

```
root@server02:~# ping 172.16.2.2
PING 172.16.2.2 (172.16.2.2) 56(84) bytes of data.
64 bytes from 172.16.2.2: icmp_seq=1 ttl=64 time=4.84 ms
64 bytes from 172.16.2.2: icmp_seq=2 ttl=64 time=4.27 ms
^C
--- 172.16.2.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
1001ms
rtt min/avg/max/mdev = 4.275/4.558/4.841/0.283 ms

root@server05:~# ping 172.16.2.1
PING 172.16.2.1 (172.16.2.1) 56(84) bytes of data.
64 bytes from 172.16.2.1: icmp_seq=1 ttl=64 time=4.93 ms
64 bytes from 172.16.2.1: icmp_seq=2 ttl=64 time=4.49 ms
^C
--- 172.16.2.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
1001ms
rtt min/avg/max/mdev = 4.495/4.714/4.933/0.219 ms
```

2) Inter-VxLAN Routing – Asymmetrical Model

If everything was correctly configured, all servers should be mutually reachable. For example, FS10 can ping FS2 :

```
root@server02:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=3 ttl=63 time=5.05 ms
64 bytes from 172.16.1.2: icmp_seq=7 ttl=63 time=4.96 ms
64 bytes from 172.16.1.2: icmp_seq=9 ttl=63 time=5.13 ms
64 bytes from 172.16.1.2: icmp_seq=10 ttl=63 time=5.10 ms
^C
--- 172.16.1.2 ping statistics ---
10 packets transmitted, 4 received, 60% packet loss, time
9126ms
rtt min/avg/max/mdev = 4.961/5.062/5.132/0.108 ms
```

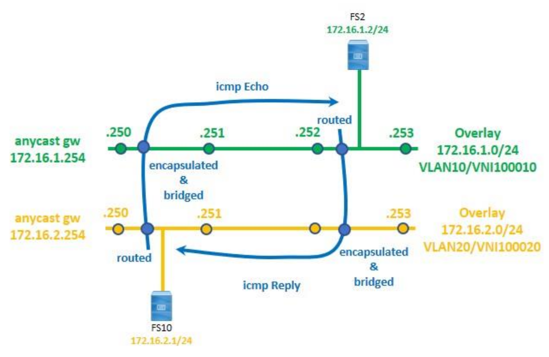
The interesting thing to monitor when FS10 pings FS2 is the packet flow of the ICMP packets.
L1 encapsulates the ICMP Echo and sends the resulting packet into VNI 100010 (VLAN10):

```
root@leaf01:mgmt:/home/cumulus# tcpdump -ni swp51 host
10.1.1.13
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on swp51, link-type EN10MB (Ethernet), capture
size 262144 bytes
12:09:34.140642 IP 10.1.1.13.51770 > 10.1.1.21.4789: VXLAN,
flags [I] (0x08), vni 100010
IP 172.16.2.1 > 172.16.1.2: ICMP echo request, id 8528, seq
360, length 64
```

Then L3 encapsulates the ICMP Reply to send it into VNI 100020 (VLAN 20):

```
root@leaf03:mgmt:/home/cumulus# tcpdump -ni swp52 host
10.1.1.21
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on swp52, link-type EN10MB (Ethernet), capture
size 262144 bytes
12:09:17.185692 IP 10.1.1.21.57109 > 10.1.1.13.4789: VXLAN,
flags [I] (0x08), vni 100020
IP 172.16.1.2 > 172.16.2.1: ICMP echo reply, id 8528, seq
343, length 64
12:09:18.188032 IP 10.1.1.21.57109 > 10.1.1.13.4789: VXLAN,
flags [I] (0x08), vni 100020
IP 172.16.1.2 > 172.16.2.1: ICMP echo reply, id 8528, seq
344, length 64
```

Packet flow is as the following :



This illustrates the asymmetrical model:

- the native packet gets routed by the ingress Leaf and the route lookup process provides the VNI of the destination,
- the resulting & encapsulated packet gets bridged by the ingress leaf into that VNI.

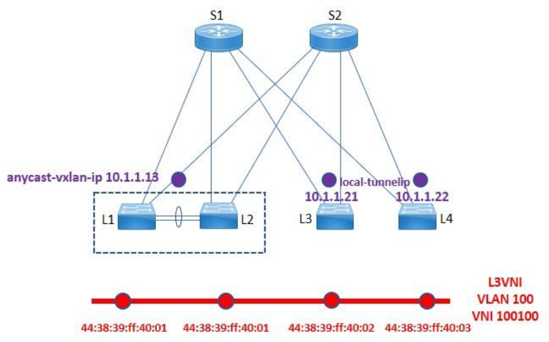
It's obvious that asymmetrical model requires that Leafs must know all sources & destinations.

It also means that Leafs need to collect as many Route Type 2 messages as there are connected hosts. Which may not scale very well in large infrastructures.

Therefore, symmetrical model came in.

3) Inter-VxLAN Routing – Symmetrical model

Symmetrical model requires the deployment of a new overlay, the L3VNI. Here are the characteristics of that L3VNI:



Basically, its configuration aligns with any other overlay. Additionally, the L3VNI will need:

- An hardware addresses to forge the encapsulated packet (note that will be the same hardware address onto members of the MLAG),
- one specific command telling that overlay is the L3VNI of vrf1.

Here is the configuration on L1:

```
#
# L1 - Make VNI100100 the L3VNI
#
net add vrf vrf1 vni 100100
#
# L1 - Bridge entries
#
net add bridge bridge ports vxlan100
net add bridge bridge vids 100
#
# L1 - VLAN, h/W address & SVI
#
net add vlan 100 hwaddress 44:38:39:ff:40:01
net add vlan 100 vlan-id 100
net add vlan 100 vlan-raw-device bridge
net add vlan 100 vrf vrf1
#
# L1 - VxLAN configuration
#
net add vxlan vxlan100 vxlan id 100100
net add vxlan vxlan100 stp bpduguard
net add vxlan vxlan100 stp portbpdufilter
net add vxlan vxlan100 vxlan local-tunnelip
10.1.1.11
net add vxlan vxlan100 bridge access 100
net commit
```

I let the reader adapt the configurations on other devices.

All Leafs should now be aware of their L3VNI:

```
root@leaf01:mgmt:/home/cumulus# net show vrf vni
VRF                                VNI          VxLAN IF
L3-SVI              State Rmac
vrf1                               100100       vxlan100
vlan100             Up    44:38:39:ff:40:01
```

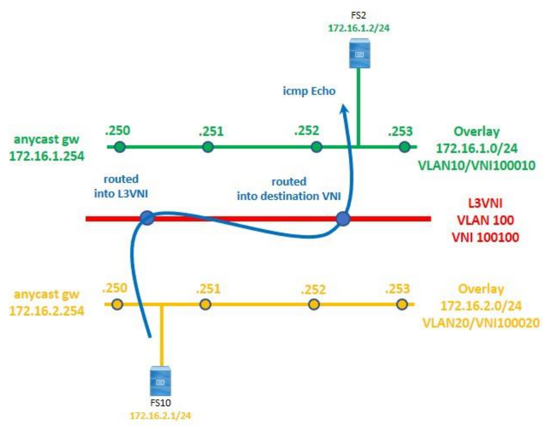
Obviously FS10 can still ping FS2:

```
root@server02:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=4 ttl=62 time=4.06 ms
64 bytes from 172.16.1.2: icmp_seq=7 ttl=62 time=12.2 ms
64 bytes from 172.16.1.2: icmp_seq=9 ttl=62 time=4.09 ms
^C
--- 172.16.1.2 ping statistics ---
9 packets transmitted, 3 received, 66% packet loss, time
8117ms
rtt min/avg/max/mdev = 4.061/6.816/12.291/3.872 ms
```

On L3, listening to traffic carried over the VLAN 100 gives the following:

```
root@leaf03:mgmt:/home/cumulus# tcpdump -ni vxlan100
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on vxlan100, link-type EN10MB (Ethernet), capture
size 262144 bytes
13:10:54.964052  IP  172.16.2.1  >  172.16.1.2:  ICMP  echo
request, id 18859, seq 2, length 64
13:10:57.012158  IP  172.16.2.1  >  172.16.1.2:  ICMP  echo
request, id 18859, seq 4, length 64
13:10:57.023725  IP  172.16.1.2  >  172.16.2.1:  ICMP  echo
reply, id 18859, seq 4, length 64
13:10:59.027862  IP  172.16.2.1  >  172.16.1.2:  ICMP  echo
request, id 18859, seq 6, length 64
13:11:01.075863  IP  172.16.2.1  >  172.16.1.2:  ICMP  echo
request, id 18859, seq 8, length 64
13:11:01.078367  IP  172.16.1.2  >  172.16.2.1:  ICMP  echo
reply, id 18859, seq 8, length 64
```

Which means that both Echo & Reply use the L3VNI as a transit. The packet flow is fundamentally different as you can see for the ICMP Echo:



The ICMP Echo gets routed to the L3VNI by the ingress Leaf, then gets routed to the VLAN destination on the receiving Leaf.

[1] https://www.nvidia.com/en-us/networking/network-simulation/ (requires registration)
[2] https://resource.nvidia.com/en-us-linux/cumulus-Linux-architecture-ebook describes the architecture of NVidia Cumulus Linux
[3] Unnumbered BGP uses Extended Next Hop Encoding capability, cf https://datatracker.ietf.org/doc/html/rfc5549
[4] A good overview of MP-BGP Route Types can be found here http://www.bgphelp.com/2017/05/22/evpn-route-types/