

CUMULUS NVUE: LAB GUIDE

Scope

This workbook covers configurations of network protocols and function using Nvidia Cumulus Linux as the Network Operating System. This workbook is based on the next generation CLI (NVUE). It is prepared for being used alongside the standard 3 session Cumulus Linux Boot Camp.

Audience

This workbook is intended for Technical Training students.

Objectives

By the end of this workbook, students will be able to:

- Configure basic switch functions with Cumulus Linux
- Configure layer 2 and layer 3 protocols with Cumulus Linux
- Verify configuration and connectivity.
- Monitor and troubleshoot networking related connectivity issues

Overview

Each student will be using the Nvidia Cumulus Air © platform, exercises in this workbook on a group of devices (four servers and four switches).

Notice

Please follow the instructions below carefully to successfully complete the practice. If you encounter technical issues, please contact the Nvidia Networking Academy team:

academy-support@nvidia.com

Release Date and Disclaimer

Revision 1.2 – June 2023 (Based on CL 5.5 Software release)

The lab was created by using Cumulus VX, the behavior/functionality of a physical environment might differ.

Good Luck,

NVIDIA Academy team

TABLE OF CONTENTS

PREREQUISITES AND GUIDELINES	4
ACADEMY LAB TOPOLOGY	5
ACADEMY LAB ACCESS.....	6
PRACTICE 1: PREPARING THE LAB AND FIRST STEPS	8
PRACTICE 2: BASIC SWITCH FUNCTIONS.....	24
PRACTICE 3: VLANs AND TRUNKING	31
PRACTICE 4: CONFIGURING MLAG and VRR	38
PRACTICE 4 NEW: CONFIGURING MLAG and VRR.....	52
PRACTICE 5: CONFIGURING BGP UNNUMBERED	56
PRACTICE 6: CONFIGURING VXLAN WITH EVPN	67
PRACTICE 7: DISTRIBUTED SYMMETRIC VXLAN ROUTING	74
PRACTICE 8: NVUE API.....	80

PREREQUISITES AND GUIDELINES

Please perform and review the following steps before you start:

- Enter the Cumulus Air web page : <https://air.nvidia.com/Login>
Click “GET STARTED” button.

WELCOME TO NVIDIA CUMULUS AIR

To get started, please sign in below.

GET STARTED

- If you have already created an account, use your credentials to [Login](#).
- To sign up for the first time, click “[Register](#)” and fill in your details.
Once completed, a confirmation email will be sent, open it to activate your new account.

EMAIL ADDRESS

PASSWORD

[Forgot password?](#)

LOGIN

Don't have an account? [Register.](#)

- Once you are logged in, you will reach the “Cumulus in The Cloud” dashboard.
Wait for the lab to be [Loaded](#).
- Use the instructor provided lab name or click on the “[Academy ILT 5.3.0](#)” label/name to access your lab.




AcademyILT CL5.3.0

✓ Loaded

January 8, 2023

about 6 hours ago

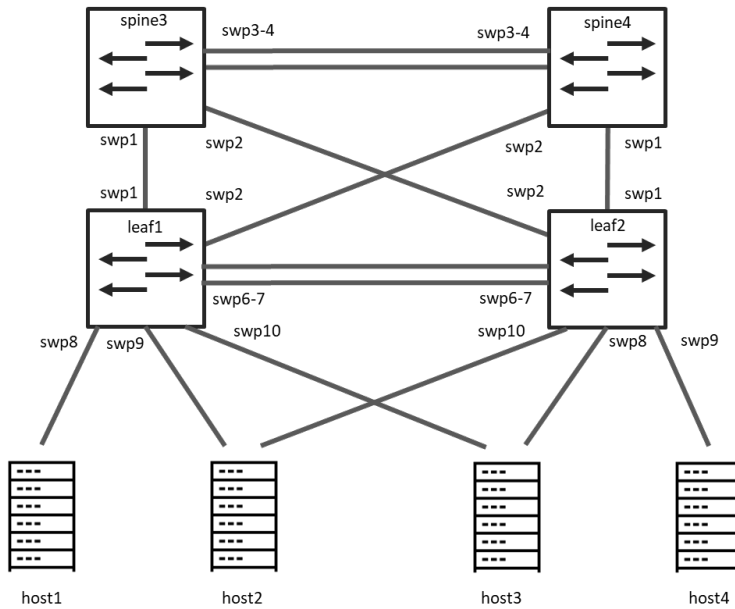
alq@nvidia.com

ACADEMY LAB TOPOLOGY

Every student is using her/his own individual lab.

The training lab is organized in the following topology:



The lab layout resembles mostly the former NCLU based Cumulus Linux boot camp lab layout.

Four switches (leaf1, leaf2, spine3, and spine4) are running Cumulus Linux, previously CL 5.0.1 as shown below, which version are your devices using today?

```
cumulus@leaf1:mgmt:~$ net show ver
NCLU_VERSION=1.0-cl5.0.0u11
DISTRIB_ID="Cumulus Linux"
DISTRIB_RELEASE=5.0.1
DISTRIB_DESCRIPTION="Cumulus Linux 5.0.1"
```

Question: If “your” version would be CL 5.0.1 like in the example above but you have to use CL 5.5.0 for example, do you know about two options how can you change the NOS from CL 5.0.1 to CL 5.5.0?

The four servers (host1, host2, host3, and host4) are running Ubuntu 20.04.

```
cumulus@host1:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.4 LTS
Release:        20.04
Codename:       focal
```

The out-of-band (OOB) infrastructure is provided via:

An oob-mgmt-switch, which is transparent during the entire labs, and is running CL 4.4.0.

The Jump-host (oob-mgmt-server) is running Ubuntu 18.04 and provides the services like DHCP and Orchestration (Ansible), next to others:

```
cumulus@oob-mgmt-server:/home/ubuntu$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
```

SSH access:

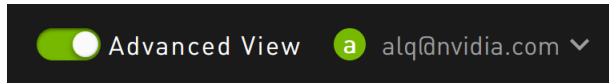
The user-account for the **oob-mgmt-server** when accessing via SSH might be “nvidia”. We will use the “cumulus” user, please change from user “nvidia” to user “cumulus” if needed. However the lab should be prepared with user “cumulus” and the password “Academy123”. Please verify that you changed to use the user “cumulus” esp. on the oob-mgmt-server.

ACADEMY LAB ACCESS

Views

The simulation environment offers two views: standard and advanced.

Via a button on the upper right corner:



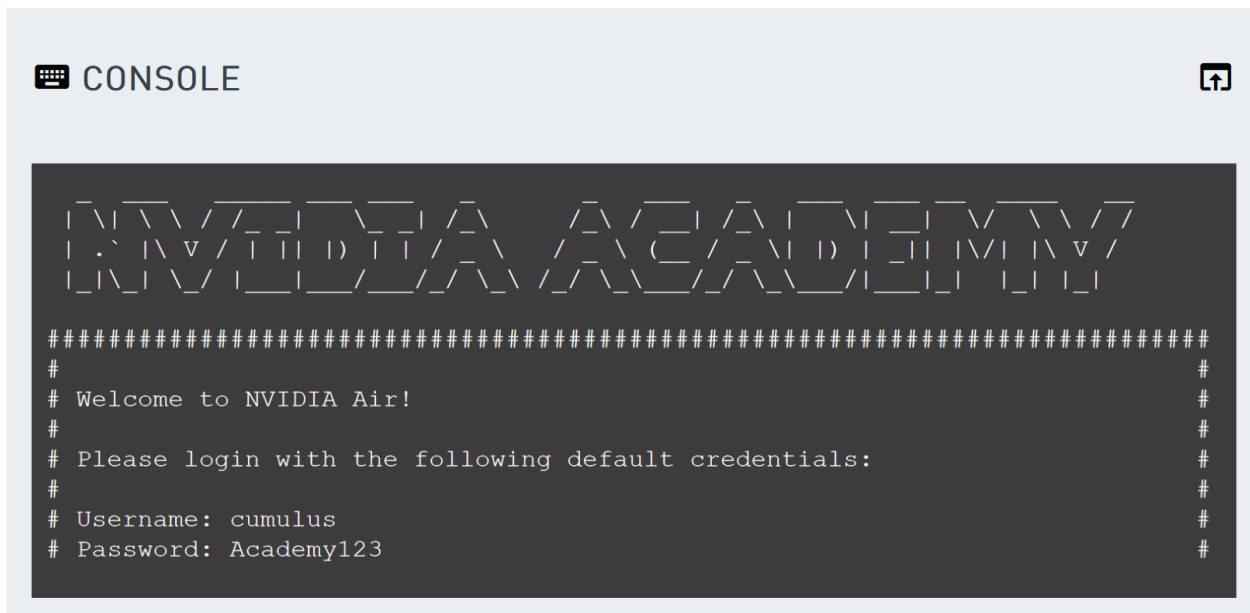
you can switch between the views. We recommend using the “Advanced View”.

Console-GUI

The console-GUI offers a connection to the oob-mgmt-server.

Using “cumulus” as the user and “Academy123” as the passwords allows access.

From here you can ssh directly to the nodes e.g., ssh leaf1

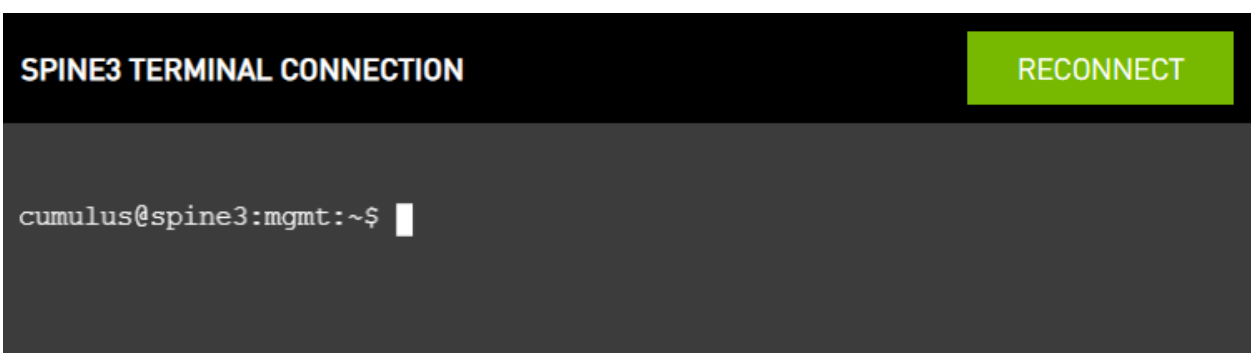


In the upper right corner you find the icon to pop-out the Console-GUI, which allows you to open multiple connections via the “plus icon” e.g. to the **oob-mgmt-server** and **leaf1**:



```
LEAF1 - RUNNING — Mozilla Firefox
https://air.nvidia.com/Terminal?node_id=1cb2f8c7-f738-4ea9-936d- 80%
+ oob-mgmt-server x leaf1 x
leaf1 login: cumulus
Password:
Last login: Mon Mar 21 08:17:22 UTC 2022 from 192.168.200.1 on pts/0
Linux leaf1 4.19.0-cl-1-amd64 #1 SMP Debian 4.19.205-1+c15.0.0u1 (2021-12-09) x86_64
Welcome to NVIDIA Cumulus VX (TM)
NVIDIA Cumulus VX (TM) is a community supported virtual appliance designed
for experiencing, testing and prototyping NVIDIA Cumulus' latest technology.
For any questions or technical support, visit our community site at:
https://www.nvidia.com/en-us/support
The registered trademark Linux (R) is used pursuant to a sublicense from LMI,
the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide
basis.
cumulus@leaf1:mgmt:~$
```

1. When the login prompt appears, enter the username – “**cumulus**”
2. When the password prompt appears, enter the password – “**Academy123**” and press Enter.
3. You should now be prompted with the node’s name. This indicates that you have successfully accessed the **node**.



```
SPINE3 TERMINAL CONNECTION RECONNECT
cumulus@spine3:mgmt:~$
```

PRACTICE 1: PREPARING THE LAB AND FIRST

netd and nvued - Daemons:

In this Lab we use CL 5.0 or newer for our “production” network (leaf1, leaf2, spine3, and spine4). Releases 4.3 and 4.4 used in former labs, could have enabled NVUE but had to have NCLU disabled. With CL 5.0 both daemons for NVUE and NCLU are active, the later one for providing a complete set of show commands, all starting with

```
$ net show ...
```

It is expected that once all show commands are available with NVUE, this will be the only daemon startend and available. In other words at this point in time it is expected that no more “**\$ net show**” commands are available and needed.

General:

All tasks are executed with the user cumulus unless explicitly stated.

Practice Objectives:

In this practice session you will become familiar with the Cumulus Linux NVUE cli:

- You will use ‘**nv set**’ and ‘**nv unset**’ commands to change the configuration.
- You will use ‘**nv config apply**’ command to apply configuration changes.
- To save the configuration we will use ‘**nv config save**’.
- You will use ‘**nv config diff**’ commands to compare revisions.
- Lastly, you will use ‘**nv config patch**’ and use ‘**nv config replace**’ to modify the pending configurations.
- To view items, we use use ‘**nv show**’.

Login-Information:

The pre-provisioned lab uses a welcome banner “ROCKET TURTLE” and also lists information about the specific node you have reached like the Management interface, Uptime and the Kernel next to the NOS.

Please observe we use for security and convenience keys instead of usernames with a password:

```
cumulus@oob-mgmt-server:~$ ssh leaf1
Linux leaf1 5.10.0-cl-1-amd64 #1 SMP Debian 5.10.162-1+c15.5.0u31 (2023-05-02) x86_64
```

welcome to NVIDIA Cumulus VX (TM)

NVIDIA Cumulus VX (TM) is a community supported virtual appliance designed for experiencing, testing and prototyping NVIDIA Cumulus' latest technology. For any questions or technical support, visit our community site at: <https://www.nvidia.com/en-us/support>

The registered trademark Linux (R) is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Last login: Tue May 30 06:11:23 2023 from 192.168.200.1

```
#####          #####          #####          #####          #####          #####
##          ##          ##          ##          ##          ##          ##
##          ##          ##          ##          ##          ##          ##
#####          ##          ##          ##          ##          ##          ##
##          ##          ##          ##          ##          ##          ##
##          ##          ##          ##          ##          ##          ##
##          ##          #####          #####          ##          ##          ##
```

```
#####          ##          ##          #####          #####          ##          #####
##          ##          ##          ##          ##          ##          ##
##          ##          ##          ##          ##          ##          ##
##          ##          ##          #####          ##          ##          #####
##          ##          ##          ##          ##          ##          ##
##          ##          ##          ##          ##          ##          ##
##          #####          ##          ##          ##          #####          #####
```

```
You are working now in: DC Site: Hamburg single-site
On device: leaf1
Operating System: Cumulus Linux 5.5.0
Kernel: 5.10.0-cl-1-amd64
Uptime: 1 day, 1 hour, 19 minutes
Managemnt via eth0: 192.168.200.2/24
Loopback:
```

The environment uses a “single site” following the leaf-spine architecture.

The “Expert Training” uses a dual-active data-center setup with two sites.

Task-01

Please **ssh** to at least one node and verify that both daemons are running if you are using a

```
cumulus@leaf1:mgmt:~$ sudo systemctl status netd
netd.service - Network Command Line Utility Daemon
   Loaded: loaded (/lib/systemd/system/netd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-03-07 08:48:57 UTC; 2 weeks 0 days ago
     Main PID: 975 (python)
        Memory: 52.9M
       CGroup: /system.slice/netd.service
               └─975 /usr/bin/python -O /usr/sbin/netd -d
```

```
cumulus@leaf1:mgmt:~$ sudo systemctl status nvued
nvued.service - NVIDIA User Experience Daemon
   Loaded: loaded (/lib/systemd/system/nvued.service; enabled; vendor preset:
enabled)
   Active: active (running) since Mon 2022-03-07 08:48:50 UTC; 2 weeks 0 days ago
     Main PID: 411 (python3)
        Memory: 222.8M
       CGroup: /system.slice/nvued.service
               └─411 /usr/bin/python3 /usr/sbin/nvued -daemon
```

CL 5.x but not the LTS release. When using a LTS release only one daemon should be started. Configuration tasks are not (anymore) feasible via NCLU.
In other words, we use NCLU only for providing show commands if needed.

Task-02

```
cumulus@spine4:mgmt:~$ nv show int
Interface  State  Speed  MTU  Type  Remote Host  Remote Port  Summary
-----
eth0       up     1G     1500  eth   oob-mgmt-switch  swp5         IP Address: 192.168.200.5/24
lo         up                65536  loopback                                IP Address: 127.0.0.1/8
                                IP Address: 172.16.100.4/32
mgmt       up                65575  vrf                                     IP Address: ::1/128
                                IP Address: 127.0.0.1/8
                                IP Address: ::1/128
swp1       up     1G     9216  swp
swp2       up     1G     9216  swp
swp3       down   1G     9216  swp
swp4       down   1G     9216  swp
```

This output suggests that a non-default configuration is in place.
Interfaces **swp** are not configured or available with factory settings, here swp1-2 for example are “up”.

We will “clean-up” any legacy configurations before we start.

Task-03

SSH to all cumulus nodes and verify if the file **/etc/nvue.d/startup.yaml** is present.

By default, this file (as of CL 5.0) is not present. If present remove the file.

Apply the empty config to all nodes:

```
$ nv config apply empty
```

This also removes the hostname and sets “cumulus” as the hostname.

```
cumulus@spine3:mgmt:~$ nv config show
- header:
  model: VX
  nvue-api-version: nvue_v1
  rev-id: 1.0
  version: Cumulus Linux 5.5.0
- set: {}
```

Thus, we like to apply the individual hostname again, via:

```
$ nv set system hostname <name>
```

```
cumulus@spine3:mgmt:~$ nv config show
- header:
  model: VX
  nvue-api-version: nvue_v1
  rev-id: 1.0
  version: Cumulus Linux 5.5.0
- set:
  system:
    hostname: spine3
```

Note: The syntax has changed between CL 4.4 and CL 5.0, so if you happen to have an older CUE or NVUE config file, please verify, and change the hostname configuration.

Apply your cleaned configuration via

```
$ nv config apply
```

Task-04

Host setup

The lab is using four hosts (host1, host2, host3, and host4) running Linux (Ubuntu 20.04). The Interface Manager by default is Netplan.

In case you are new to Linux and/or Netplan, please consider consulting external resources like:

<https://netplan.io/examples/>

or ask your instructor for a quick-live-demo/introduction.

Alternatively, to Netplan you can configure the hosts via:

- `ifconfig`
- `ip(route2)`

Both options create the same functionality for our use-cases but both are not persistent and might result in more time being consumed during the class.

However, `ip(route2)` provides the most configuration options compared to both **`ifconfig`** and **`netplan`**.

- Our order of preference:
 - 1) Netplan
 - 2) Ip(route2)
 - 3) ifconfig

Each hosts use two interfaces:

- **Eth0** for OOB Management, this interface should NOT be changed
- **Eth2** this interface connecting to the leaf switches and will be configured during class

Task-06

Enable all switch interfaces

Verify which interfaces are available on all four cumulus nodes:

Interface	State	Speed	MTU	Type	Remote Host	Remote Port	Summary
eth0	up	1G	1500	eth	oob-mgmt-switch	swp5	IP Address: 192.168.200.5/24
lo	up		65536	loopback			IP Address: 127.0.0.1/8
mgmt	up		65575	vrf			IP Address: ::1/128
swp1	down	1G	9216	swp			IP Address: 127.0.0.1/8
swp2	down	1G	9216	swp			IP Address: ::1/128
swp3	down	1G	9216	swp			IP Address: 127.0.0.1/8
swp4	down	1G	9216	swp			IP Address: ::1/128

Interface	State	Speed	MTU	Type	Remote Host	Remote Port	Summary
eth0	up	1G	1500	eth	oob-mgmt-switch	swp2	IP Address: 192.168.200.2/24
lo	up		65536	loopback			IP Address: 127.0.0.1/8
mgmt	up		65575	vrf			IP Address: ::1/128
swp1	down	1G	9216	swp			IP Address: 127.0.0.1/8
swp2	down	1G	9216	swp			IP Address: ::1/128
swp5	down	1G	9216	swp			IP Address: 127.0.0.1/8
swp6	down	1G	9216	swp			IP Address: ::1/128
swp8	down	1G	9216	swp			IP Address: 127.0.0.1/8
swp9	down	1G	9216	swp			IP Address: ::1/128
swp10	down	1G	1500	swp			IP Address: 127.0.0.1/8

We can see for example on leaf1 that the interfaces swp1, swp2, swp5, swp6, swp8, swp9 and swp10 are “available” but in DOWN state.

Enable all ADMDN interfaces (ADMDN == ADMIN DOWN):

```
cumulus@leaf1:mgmt:~$ nv set interface swp1,swp2,swp5,swp6,swp8,swp9,swp10
cumulus@leaf1:mgmt:~$ nv config diff
- set:
  interface:
    swp1-2,5-6,8-10:
      type: swp
```

Is your configuration active? ... “activate” your configuration and verify.

Example after enabling the interfaces on leaf1:

```
cumulus@leaf1:mgmt:~$ nv show int
```

Interface	State	Speed	MTU	Type	Remote Host	Remote Port	Summary
eth0	up	1G	1500	eth	oob-mgmt-switch	swp2	IP
Address: 192.168.200.2/24							
lo	up		65536	loopback			IP
Address: 127.0.0.1/8							IP
Address: ::1/128							
mgmt	up		65575	vrf			IP
Address: 127.0.0.1/8							IP
Address: ::1/128							
swp1	up	1G	9216	swp			
swp2	up	1G	9216	swp	spine4	swp2	
swp5	up	1G	9216	swp			
swp6	up	1G	9216	swp			
swp8	up	1G	9216	swp	host1	48:b0:2d:c6:f0:b6	
swp9	up	1G	9216	swp	host2	48:b0:2d:84:65:3e	
swp10	up	1G	9216	swp	host3	48:b0:2d:06:e6:98	

Hosts

Configure your hosts with the following IPv4 addresses, we don't need a default gateway right now:

Host1(eth2):	172.16.1.1/24
Host2(eth2):	172.16.1.2/24
Host3(eth2):	172.16.1.3/24
Host4(eth2):	172.16.1.4/24

Task-07

Verify your link-layer neighbors (LLDP)

Verify that your switches are connected correctly according to the lab layout by using LLDP (example leaf1 and leaf2):

swp8	up	1G	9216	swp	host1	48:b0:2d:c6:f0:b6
swp9	up	1G	9216	swp	host2	48:b0:2d:84:65:3e
swp10	up	1G	9216	swp	host3	48:b0:2d:06:e6:98
swp8	up	1G	9216	swp	host3	48:b0:2d:6e:32:77
swp9	up	1G	9216	swp	host4	48:b0:2d:36:25:05
swp10	up	1G	9216	swp	host2	48:b0:2d:7f:e7:7b

Task-08

Optional

It is beneficial but maybe old-fashioned to create on a piece of paper a network documentation with the information about Layer 1 and Layer 2 to be used as a reference during class.

Task-09

Optional PTM

PTM is the abbreviation for Prescriptive Topology Manager, a service to verify if the intended layout correlates with the actual layout (both with a simulation or physical network).

Verify that your PTM daemon is running:

```
cumulus@leaf1:gmt:/etc/ptm.d$ sudo systemctl status ptmd
ptmd.service - Prescriptive Topology Manager (PTM) Daemon.
  Loaded: loaded (/lib/systemd/system/ptmd.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/ptmd.service.d
           └─ptmd.conf
  Active: active (running) since Mon 2022-03-07 08:48:58 UTC; 2 weeks 1 days ago
    Docs: man:ptmd(8)
           man:ptmctl(8)
  Main PID: 1093 (ptmd)
    Memory: 932.0K
    CGroup: /system.slice/ptmd.service
            └─1093 /usr/sbin/ptmd -l INFO
```

PTM is using a file called **topology.dot** which needs to be stored in the **ptm.d** directory:

```
cumulus@leaf1:gmt:/etc/ptm.d$ ls
bfd-sess-down  bfd-sess-up  if-topo-fail  if-topo-pass
```

here in this example this file is missing.

Create a **topology.dot** file on one or multiple cumulus nodes.
You can use the following 3-line template as a starting point:

```
graph "<your name>" {
  "spine3":"swp1" -- "leaf1":"swp1"
}
```


Once in place restart ptm and verify. For leaf1 the expected result is:

```
cumulus@leaf1:mgmt:~$ ptmctl
```

port	cb1	BFD	BFD	BFD	BFD
	status	status	peer	local	type
swp5	pass	N/A	N/A	N/A	N/A
swp6	pass	N/A	N/A	N/A	N/A
swp8	pass	N/A	N/A	N/A	N/A
swp9	pass	N/A	N/A	N/A	N/A
swp1	pass	N/A	N/A	N/A	N/A
swp2	pass	N/A	N/A	N/A	N/A

```
cumulus@leaf1:mgmt:~$ net show lldp
```

LocalPort	Speed	Mode	RemoteHost	RemotePort
eth0	1G	Mgmt	oob-mgmt-switch	swp2
swp1	1G	Default	spine3	swp1
swp2	1G	Default	spine4	swp2
swp5	1G	Default	leaf2	swp5
swp6	1G	Default	leaf2	swp6
swp8	1G	Default	host1	44:38:39:00:00:11
swp9	1G	Default	host2	44:38:39:00:00:13

Task-10

Environment

Verify the FAN, LED, and SENSOR information, on a virtual switch dummy information is made available for the show commands:

```
cumulus@leaf2:mgmt:~$ nv show platform environment sensor
```

Sensor Name	Critical Temp	Max Temp	Min Temp	Sensor State	Current
temperature (°C)					
Board Sensor Near Virtual Switch	85.0	80.0	5	ok	25.0
Board Sensor at Front Left Corner	85.0	80.0	5	ok	25.0
Board Sensor at Front Right Corner	85.0	80.0	5	ok	25.0
Board Sensor near CPU	85.0	80.0	5	ok	25.0
Board Sensor near Fan	85.0	80.0	5	ok	25.0
PSU1 Temp Sensor	85.0	80.0	5	ok	25.0
PSU2 Temp Sensor	85.0	80.0	5	ok	25.0

```
cumulus@leaf2:mgmt:~$ nv show platform environment fan
```

Name	Limit	variance	Max Speed	Min Speed	Current Speed (RPM)	Fan State
Fan1	29000		2500	6000		ok
Fan2	29000		2500	6000		ok
Fan3	29000		2500	6000		ok
Fan4	29000		2500	6000		ok
Fan5	29000		2500	6000		ok
Fan6	29000		2500	6000		ok
PSU1Fan1	29000		2500	6000		ok
PSU2Fan1	29000		2500	6000		ok

```
cumulus@leaf2:mgmt:~$ nv show platform environment psu
```

Name	PSU State
PSU1	ok
PSU2	ok

```
cumulus@leaf2:mgmt:~$ nv show platform hardware
```

	operational	applied
base-mac	48:B0:2D:C2:C9:1F	
manufacturer	Cumulus	
memory	1.69 GB	
model	VX	
part-number	5.5.0	
product-name	VX	
serial-number	48:b0:2d:c2:c9:1f	
system-mac	48:b0:2d:c2:c9:27	

Task-11

Remove interfaces

Remove the direct interfaces between the leaf switches.

```
$ nv unset interface swp5-6
$ nv config apply
```

Task-12

Verify the configuration history

```
$ nv config history
```

Example output (CL < 5.3):

```
cumulus@leaf1:mgmt:~$ nv config history
- apply-id: n/9
  apply-meta:
    method: CLI
    reason: Config update
    rev_id: changeset/cumulus/2022-03-22_12.12.01_AWD3
    state_controls: {}
    user: cumulus
  date: '2022-03-22T12:12:16+00:00'
  message: Config update by cumulus via CLI
  ref: apply/2022-03-22_12.12.13_AWD4/done
- apply-id: n/8
  apply-meta:
    method: CLI
    reason: Config update
    rev_id: startup
    state_controls: {}
    user: root
  date: '2022-03-22T10:52:36+00:00'
  message: Config update by root via CLI
  ref: apply/2022-03-22_10.52.34_AWD2/done
```

Identify the last two apply-id's, here n/9 and n/8.

n/9 includes the unsetting of swp5-6 and n/8 should include them as the state before unsetting.

(05-JAN-2023) Starting with CL 5.3 a rev_id is used:

```
cumulus@leaf1:mgmt:~$ nv config history
- apply-meta:
  method: CLI
  reason: Config update
  rev_id: '2'
  state_controls: {}
  user: cumulus
```

```

date: '2023-01-05T19:57:26+00:00'
message: Config update by cumulus via CLI
ref: rev_2_apply_1
- apply-meta:
  method: CLI
  reason: Config update
  rev_id: '1'
  state_controls: {}
  user: cumulus
date: '2023-01-05T19:56:56+00:00'
message: Config update by cumulus via CLI
ref: rev_1_apply_1

```

Verify that the interfaces swp5-6 are not set on your leaf switches:

```

cumulus@leaf1:mgmt:~$ net show int
State Name Spd MTU Mode LLDP Summary
-----
UP lo N/A 65536 Loopback IP: 127.0.0.1/8
lo IP: ::1/128
UP eth0 1G 1500 Mgmt oob-mgmt-switch (swp2) Master: mgmt(UP)
eth0 IP: 192.168.200.2/24(DHCP)
UP swp1 1G 9216 Default spine3 (swp1)
UP swp2 1G 9216 Default spine4 (swp2)
UP swp8 1G 9216 Default host1 (44:38:39:00:00:11)
UP swp9 1G 9216 Default host2 (44:38:39:00:00:13)
UP mgmt N/A 65536 VRF IP: 127.0.0.1/8
mgmt IP: ::1/128

```

```

cumulus@leaf1:mgmt:~$ nv config apply n/8
applied

cumulus@leaf1:mgmt:~$ net show int
State Name Spd MTU Mode LLDP Summary
-----
UP lo N/A 65536 Loopback IP: 127.0.0.1/8
lo IP: ::1/128
UP eth0 1G 1500 Mgmt oob-mgmt-switch (swp2) Master: mgmt(UP)
eth0 IP:
192.168.200.2/24(DHCP)
UP swp1 1G 9216 Default spine3 (swp1)
UP swp2 1G 9216 Default spine4 (swp2)
UP swp5 1G 9216 Default leaf2 (swp5)
UP swp6 1G 9216 Default leaf2 (swp6)
UP swp8 1G 9216 Default host1 (44:38:39:00:00:11)
UP swp9 1G 9216 Default host2 (44:38:39:00:00:13)
UP mgmt N/A 65536 VRF IP: 127.0.0.1/8
mgmt IP: ::1/128

```

(05-JAN-2023) Starting with CL 5.3 a rev_id is used and can be use to re-create an old state. Here the actual state is “2” and we could activate the “startup”, an “empty” or the older state “1”

```
cumulus@leaf1:mgmt:~$ nv config apply  
1 2 empty startup
```

Note: with recent releases an auto-save option is available. If important to your use-case please raise this item in class to discuss.

Note: the interfaces swp10 are prepared for future use and eventually used. If your labguide does not yet include tasks for swp10 then they are supposed to be ignored.

Task 13: Save the configuration

- a. Make the applied configuration persistent (on all four switches):

\$ nv config save

```
cumulus@leaf1:mgmt:~$ nv config save
saved
```

Optional (more advanced) tasks:

Use a single line instruction (CLI) to make the config persistent on all four switches at once.

What would be your suggestions, please discuss in class or during lab debriefing.

Task 14: Compare

- b. Create on leaf1 a file in yaml syntax to add the interfaces swp5 and name it **/home/cumulus/replace.yaml** which includes the active configuration as a base:

\$ nv config show > /home/cumulus/replace.yaml

Select an editor (e.g. nano or vi) and add Interface swp5

\$ nano replace.yaml

```
- set:
  system:
    hostname: leaf1
  interface:
    swp1-2,5,8-9:
      type: swp
```

- c. Replace the pending configuration

\$ nv config replace ./replace.yaml

- d. Verify the pending and the applied revisions

\$ nv config diff empty

\$ nv config diff empty applied

```
cumulus@leaf1:mgmt:~$ nv config diff
- set:
  interface:
    swp5:
      type: swp

cumulus@leaf1:mgmt:~$ nv config diff empty
- set:
  system:
    hostname: leaf1
  interface:
    swp1-2,5,8-9:
      type: swp
```

Apply the configuration and verify that swp5 is available.

```
cumulus@leaf1:mgmt:~$ net show int swp5
```

	Name	MAC	Speed	MTU	Mode
UP	swp5	44:38:39:00:00:0d	1G	9216	Default

- e. Copy the replace.yaml file, rename it to patch.yaml add the missing interface swp6. Within patch.yaml remove the hostname section.

Add this set of information to the pending revision.
Use **cl config diff** commands to compare and verify.
Apply and save the new configuration.

```
$ cp replace.yaml patch.yaml
$ nano patch.yaml
$ nv config patch patch.yaml

$ nv config diff empty
$ nv config diff startup
$ nv config diff applied

$ nv config apply
$ nv config save
```

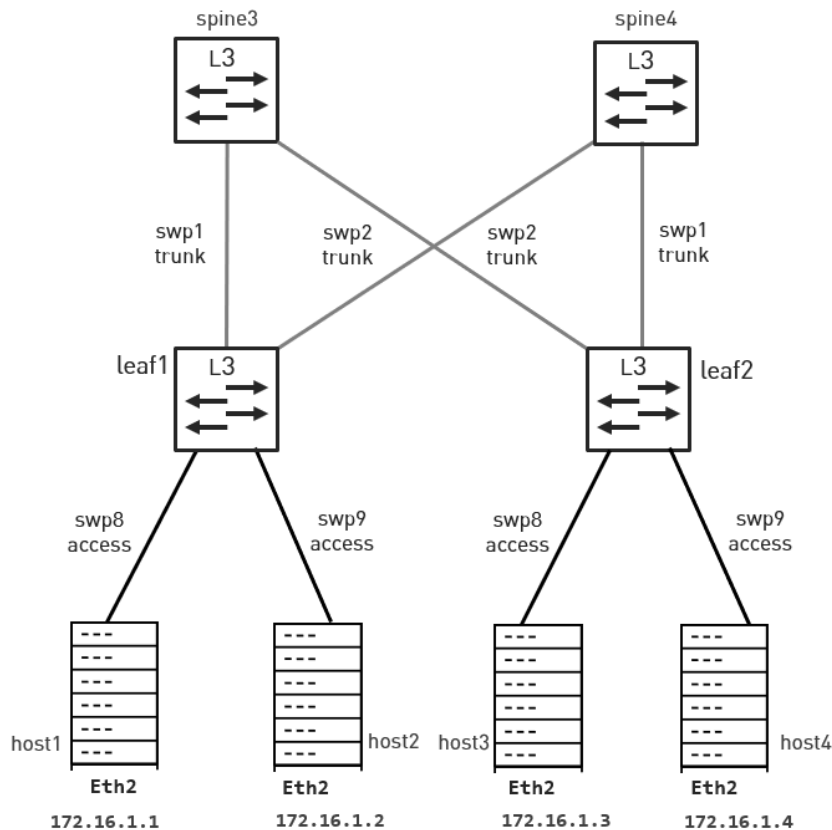
PRACTICE 2: BASIC SWITCH FUNCTIONS

Practice Objectives:

In this practice session you will create and verify IP connectivity between servers in the lab.

- You will configure the servers IP settings – an IP address, a subnet mask and a default gateway.
- You will configure a bridge on each of the Cumulus Linux switches and add switch ports to the bridge.
- You will use **‘ping’** utility to verify communication between servers in your group.
- Last, you will observe how the switch forwarding database – the MAC address table – is built and maintained.

Topology Used in this Practice:



Task 1

Configure Servers IP Settings

- a. Access the servers and check interface **'eth2'** MAC and IP settings:

```
$ ip address show dev eth2
```

or list all interfaces:

```
cumulus@host1:~$ ip -br a
lo                UNKNOWN      127.0.0.1/8 ::1/128
eth0              UP           192.168.200.6/24 fe80::4638:39ff:fe00:24/64
eth2              UP           172.16.1.1/24  fe80::4638:39ff:fe00:11/64
```

- b. If they differ from the below table, correct them (/24 subnet):

Practice Lab Servers Properties

Server	'eth2' IP Address
host1	172.16.1.1
host2	172.16.1.2
host3	172.16.1.3
host4	172.16.1.4

Task 2

Clean up and configure a Bridge

- a. Access the switches and reset configuration:

```
$ nv config apply empty -y
$ nv set system hostname [leaf1|leaf2|spine3|spine4]
$ nv config apply -y
$ nv config save
```

```
cumulus@leaf1:mgmt:~$ nv config save
saved
```

- b. On all four switches – leaves and spines – create or enable a bridge named br_default, add VLAN10.

Set the inter switch links (**swp1** and **swp2**) as trunk ports:

```
$ nv set bridge domain br_default vlan 10
$ nv set interface swp1-2 bridge domain br_default
```

On the leaf switches only – leaf1 and leaf2 - set the host-facing ports, swp8 and swp9, as access ports in VLAN 10:

```
$ nv set interface swp8-9 bridge domain br_default access 10
```

- c. Apply the pending revision:

```
$ nv config apply
```

```
cumulus@leaf1:mgmt:~$ nv config apply
applied
```

Verification:

All hosts are configured with the same subnet 172.16.1.0/24. You should be able to reach from every host every other host. No default gateway is needed.

Optional (more advanced) tasks:

Verify in the sys-file-system the resulting bridge type (traditional or vlan-aware-bridge (VAB)).

Verify in the /e/n/i file the resulting bridge type.

Verify via ifquery the resulting bridge type.

Ask your Instructor, if you are stuck and/or like to debrief this task.

d. Verify configuration:

```
$ nv config diff startup applied
$ nv config show
```

```
cumulus@leaf1:mgmt:~$ nv config show
- set:
  bridge:
    domain:
      br_default:
        vlan:
          '10': {}
  system:
    hostname: leaf1
  interface:
    swp1-2:
      bridge:
        domain:
          br_default: {}
    swp1-2,8-9:
      type: swp
    swp8-9:
      bridge:
        domain:
          br_default:
            access: 10
```

Optional (more advanced) tasks:

Why does the bridge structure contains a “domain” level?

e. Verify bridge mac-table:

```
$ nv show bridge domain br_default mac-table
```

```
$ net show bridge
```

```
cumulus@leaf1:mgmt:~$ nv show bridge domain br_default mac-table
```

	age	bridge-domain	entry-type	interface	last-update	mac	src-vni	vlan	vni	Summary
+ 0	27	br_default		swp1	90	44:38:39:00:00:05		1		
+ 1	93	br_default	permanent	swp1	93	44:38:39:00:00:06				
+ 10	93	br_default	permanent	br_default	93	44:38:39:00:00:23				
+ 2	30	br_default		swp2	88	44:38:39:00:00:17		10		
+ 3	28	br_default		swp2	88	44:38:39:00:00:15		10		
+ 4	28	br_default		swp2	89	44:38:39:00:00:0b		1		
+ 5	93	br_default	permanent	swp2	93	44:38:39:00:00:0c				
+ 6	1	br_default		swp8	88	44:38:39:00:00:11		10		
+ 7	93	br_default	permanent	swp8	93	44:38:39:00:00:12				
+ 8	1	br_default		swp9	88	44:38:39:00:00:13		10		
+ 9	93	br_default	permanent	swp9	93	44:38:39:00:00:14				

```
cumulus@leaf1:mgmt:~$ net show bridge macs
```

VLAN	Master	Interface	MAC	TunnelDest	State	Flags	LastSeen
1	br_default	swp1	44:38:39:00:00:05				00:00:29
1	br_default	swp2	44:38:39:00:00:0b				00:00:29
10	br_default	swp2	44:38:39:00:00:15				00:00:53
10	br_default	swp2	44:38:39:00:00:17				00:01:02
10	br_default	swp8	44:38:39:00:00:11				00:00:02
10	br_default	swp9	44:38:39:00:00:13				00:00:02
untagged	br_default	br_default	44:38:39:00:00:23		permanent		00:03:05
untagged	br_default	swp1	44:38:39:00:00:06		permanent		00:03:05
untagged	br_default	swp2	44:38:39:00:00:0c		permanent		00:03:05
untagged	br_default	swp8	44:38:39:00:00:12		permanent		00:03:05
untagged	br_default	swp9	44:38:39:00:00:14		permanent		00:03:05

Optional (more advanced) tasks:

Which Linux command (neither starting with net nor nv) would allow to display the learned mac addresses of the bridge?

f. Verify the bridge via the native Linux commands:

```
$ bridge fdb
$ bridge link
$ bridge vlan
```

```
cumulus@leaf1:mgmt:~$ bridge fdb
44:38:39:00:00:05 dev swp1 vlan 1 master br_default
44:38:39:00:00:06 dev swp1 master br_default permanent
44:38:39:00:00:17 dev swp2 vlan 10 master br_default
44:38:39:00:00:15 dev swp2 vlan 10 master br_default
44:38:39:00:00:0b dev swp2 vlan 1 master br_default
44:38:39:00:00:0c dev swp2 master br_default permanent
44:38:39:00:00:11 dev swp8 vlan 10 master br_default
44:38:39:00:00:12 dev swp8 master br_default permanent
44:38:39:00:00:13 dev swp9 vlan 10 master br_default
44:38:39:00:00:14 dev swp9 master br_default permanent
44:38:39:00:00:23 dev br_default master br_default permanent
```

```
cumulus@leaf1:mgmt:~$ bridge link
3: swp1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216 master br_default state forwarding
priority 8 cost 4
4: swp2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216 master br_default state forwarding
priority 8 cost 4
7: swp8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216 master br_default state forwarding
priority 8 cost 4
8: swp9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216 master br_default state forwarding
priority 8 cost 4
```

```
cumulus@leaf1:mgmt:~$ bridge vlan
port      vlan ids
swp1      1 PVID Egress Untagged
          10

swp2      1 PVID Egress Untagged
          10

swp8      10 PVID Egress Untagged

swp9      10 PVID Egress Untagged

br_default      None
```

Optional (more advanced) tasks:

If you configure an interface e.g. swp1 to work with a bridge e.g. br_default as a trunk but you have not configured any vlan for the bridge,

would swp1 be a working layer 2 interface?

Would swp1 be a trunk or an access-port and if so for which vlan?

Task 3

Observe a Switch's Forwarding Database

- Identify the MAC addresses of all four servers/hosts in your setup.

```
cumulus@host1:~$ ip -br l
lo                UNKNOWN      00:00:00:00:00:00 <LOOPBACK,UP,LOWER_UP>
eth0              UP          44:38:39:00:00:24 <BROADCAST,MULTICAST,UP,LOWER_UP>
eth2              UP          44:38:39:00:00:11 <BROADCAST,MULTICAST,UP,LOWER_UP>
```

verify that the eth2 MAC addresses is stored within the FDB/mac-address-tables of your switches

- Use '**ping**' from one server to another server in the lab.
For example, ping from server **host1** to server **host3**.

```
cumulus@host1:~$ ping 172.16.1.3
PING 172.16.1.3 (172.16.1.3) 56(84) bytes of data.
64 bytes from 172.16.1.3: icmp_seq=1 ttl=64 time=2.99 ms
64 bytes from 172.16.1.3: icmp_seq=2 ttl=64 time=2.28 ms
```

Every host should be able to reach any other host e.g. host1 -> host2,3,4

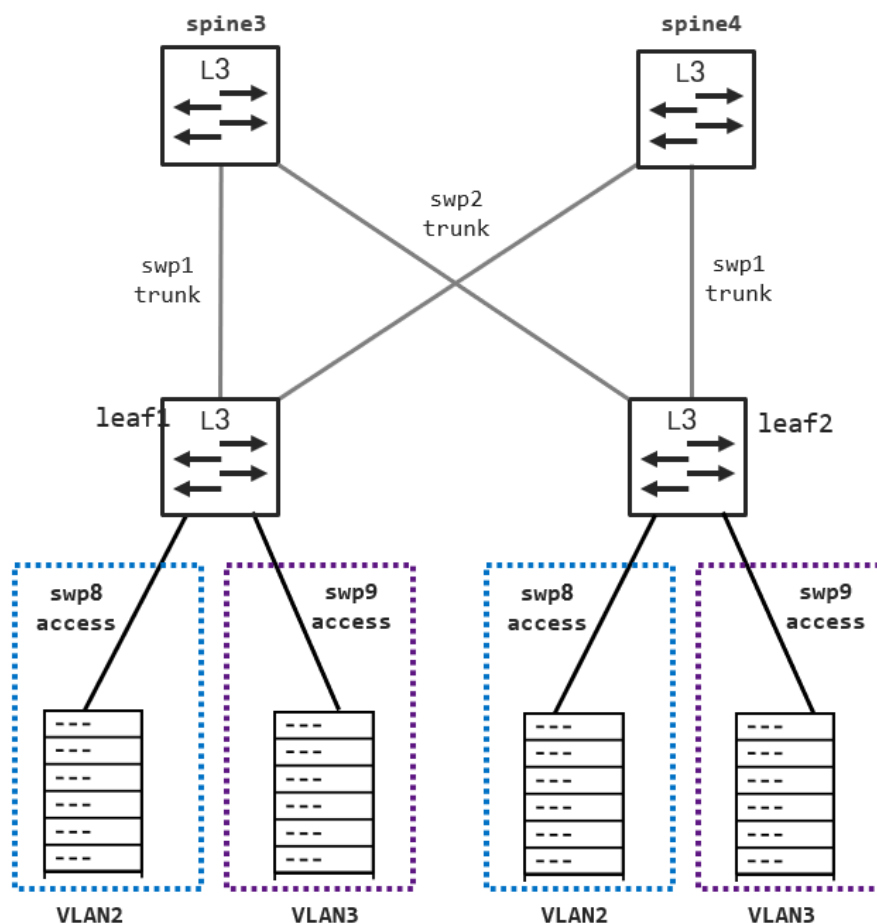
PRACTICE 3: VLANs AND TRUNKING

Practice Objectives:

In this practice session you will configure and verify VLANs and trunking:

- You will configure two new VLANs and assign switch ports connected to servers to the configured VLANs.
- You will configure SVIs to allow inter-VLAN communication.

Topology Used in this Lab:



Task 1: Configuring VLANs and Trunking

- a. Access the leaf and spine switches and reset configuration:

```
$ nv config apply empty
$ nv set system hostname [leaf1, leaf2, spine3, spine4]
$ nv config apply
```

- b. On all four switches – spines and leaves - create a vlan-aware-bridge, two vlans (2 and 3) and set the inter switch links, **swp1** and **swp2**, as trunk ports:

```
$ nv set bridge domain br_default vlan 2-3
$ nv set interface swp1-2 bridge domain br_default
```

- c. Verify if vlan 1, without explicitly configuration, is used by default and for which use case:

```
cumulus@leaf1:mgmt:~$ nv show bridge domain br_default vlan
      multicast.snooping.querier.source-ip  ptp.enable  Summary
---  -----
+ 2  0.0.0.0                                off
+ 3  0.0.0.0                                off
```

```
cumulus@leaf1:mgmt:~$ net show bridge vlan
```

Interface	VLAN	Flags
swp1	1	PVID, Egress Untagged
	2-3	[]
swp2	1	PVID, Egress Untagged
	2-3	[]

Optionally, verify that LLDP is running between the switches. Following example shows spine3 view after configuring leaf1 but before configuring leaf2.

```
$ sudo lldpccli
$ sudo lldpccli show neighbors
```

```
[lldpccli] # show neighbors summary
-----
LLDP neighbors:
-----
Interface:   eth0, via: LLDP
Chassis:
  ChassisID:  mac 44:38:39:00:00:19
  SysName:    oob-mgmt-switch
Port:
  PortID:     ifname swp4
  PortDescr:  swp4
  TTL:        120
-----
Interface:   swp1, via: LLDP
Chassis:
  ChassisID:  mac 44:38:39:00:00:1c
  SysName:    leaf1
Port:
  PortID:     ifname swp1
  PortDescr:  swp1
  TTL:        120
-----
```

- d. On leaf switches only **leaf1** and **leaf2** set the host-facing ports, swp8 and swp9, as access ports and associate them to the appropriate VLAN:

interface **swp8** in **VLAN2** and interface **swp9** in **VLAN3**

```
$ nv set interface swp9 bridge domain br_default access 3
```

- e. Commit changes and verify:

```
cumulus@leaf1:mgmt:~$ nv config apply
```

- f. Verify VLANs configuration:

```
cumulus@leaf1:mgmt:~$ nv set interface swp8 bridge domain br_default access 2
cumulus@leaf1:mgmt:~$ nv set interface swp9 bridge domain br_default access 3
```

\$ bridge vlan

```
cumulus@leaf1:mgmt:~$ bridge vlan
port      vlan ids
swp1      1 PVID Egress Untagged
          2
          3

swp2      1 PVID Egress Untagged
          2
          3

swp8      2 PVID Egress Untagged

swp9      3 PVID Egress Untagged
```

Please note:

- Access ports are shown with a single line representing the VLAN associated to the port.
- Trunk ports are shown with multiple lines representing the VLANs associated with the trunk port.

Task 2

Servers' IP settings

Access the servers and configure an IP address for interface **'eth2'** according to the table below.

Servers in the same VLAN will be configured with IP addresses in the same subnet, hence they will be able to communicate over the layer 2 network.

VLAN ID	Server	'eth2' IP Address
VLAN 2	host1	172.16.2.18/24
VLAN 3	host2	172.16.3.19/24
VLAN 2	host3	172.16.2.28/24
VLAN 3	host4	172.16.3.29/24

- a. Configure if needed the server's IP address and subnet mask according to the table above.

Optional: Configure eth3 of host2 and host3 with the following IP addresses and adjust the corresponding switchports to allow intra-vlan access.

Host2 : ethernet3 IPv4: 172.16.2.2/24

Host3: ethernet3 IPv4: 172.16.3.3/24

- b. Verify a static route entry to network **172.16.0.0/16** via the default gateway's address. Use the default gateway address in the following table:

VLAN ID	Default gateway address
VLAN 2	172.16.2.254/24
VLAN 3	172.16.3.254/24

a non-persistent setting can be achieved via ip(route2) or you might consider to configure via netplan.

```
cumulus@host1:~$ sudo ip route add 172.16.0.0/16 dev eth2 via 172.16.2.254
cumulus@host1:~$ route
Kernel IP routing table
Destination        Gateway            Genmask           Flags Metric Ref    Use Iface
default            _gateway          0.0.0.0           UG    0      0      0 eth0
172.16.0.0         172.16.2.254     255.255.0.0       UG    0      0      0 eth2
172.16.2.0         0.0.0.0           255.255.255.0     U     0      0      0 eth2
192.168.200.0      0.0.0.0           255.255.255.0     U     0      0      0 eth0
```

\$ sudo ip route add <NET_ADDRESS/MASK> dev <DEV> via <IP>

Question: how would you remove a static route you might have set incorrectly before?

- c. Use **'ping'** to check communication between servers in the same VLAN.
For example, servers **host1** and **host3**.

```
[cumulus@host1 ~]# ping 172.16.2.28
PING 172.16.2.28 (172.16.2.28) 56(84) bytes of data.
64 bytes from 172.16.2.28: icmp_seq=1 ttl=64 time=4.11 ms
64 bytes from 172.16.2.28: icmp_seq=2 ttl=64 time=2.58 ms
```

d. **Optional tasks:**

If you like to work with eth3 of host2 and host3, you need to configure the additional access links on leaf1 and leaf2:

```
cumulus@leaf1:mgmt:~$ nv set interface swp10 bridge domain br_default access 3
cumulus@leaf2:mgmt:~$ nv set interface swp10 bridge domain br_default access 2
```

```
cumulus@host2:~$ ping 172.16.2.18 -c 2
PING 172.16.2.18 (172.16.2.18) 56(84) bytes of data.
64 bytes from 172.16.2.18: icmp_seq=1 ttl=64 time=1.84 ms
64 bytes from 172.16.2.18: icmp_seq=2 ttl=64 time=1.77 ms

--- 172.16.2.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.770/1.804/1.838/0.034 ms
```

Task 3: Configuring SVIs for inter-VLAN routing

- On switch **spine3** configure two SVIs (Switch VLAN Interfaces) that will be used for routing between **VLAN2** and **VLAN3**:
 - Interface **vlan2** will serve as the default gateway for **VLAN2**
 - Interface **vlan3** will serve as the default gateway for **VLAN3**

Use the following table for IP address assignment:

Interface vlan 2	172.16.2.254/24
Interface vlan 3	172.16.3.254/24

```
$ nv set interface vlan2 ip address 172.16.2.254/24
```

```
$ nv set interface vlan3 ip address 172.16.3.254/24
```

```
cumulus@spine3:mgmt:~$ nv set interface vlan2 ip address 172.16.2.254/24
cumulus@spine3:mgmt:~$ nv set interface vlan3 ip address 172.16.3.254/24
cumulus@spine3:mgmt:~$ nv config apply
```

- Use **'ping'** and **'traceroute'** utilities to verify communication between hosts in different VLANs. For example, ping from server **'host1'** in **VLAN2** to server **'host2'** in **VLAN3**.

```
[cumulus@host1 ~]# ping 172.16.3.19
PING 172.16.3.19 (172.16.3.19) 56(84) bytes of data.
64 bytes from 172.16.3.19: icmp_seq=1 ttl=63 time=0.155 ms
```

```
[cumulus@host1 ~]# traceroute 172.16.3.19
traceroute to 172.16.3.19 (172.16.3.19), 30 hops max, 60 byte packets
 1  172.16.2.254 (172.16.2.254)  1.340 ms  1.633 ms  1.607 ms
 2  172.16.3.19 (172.16.3.19)  4.043 ms  4.023 ms  3.999 ms 2
```

Optional (more advanced) tasks:

Since **vlan2** and interface **vlan2** might cause confusion please remove the layer 3 interfaces and create on **spine3** the same functionality without using the name "vlan2" or "vlan3". Test to reach host 2 from host 1 via ICMP echo requests.

PRACTICE 4: CONFIGURING MLAG and VRR

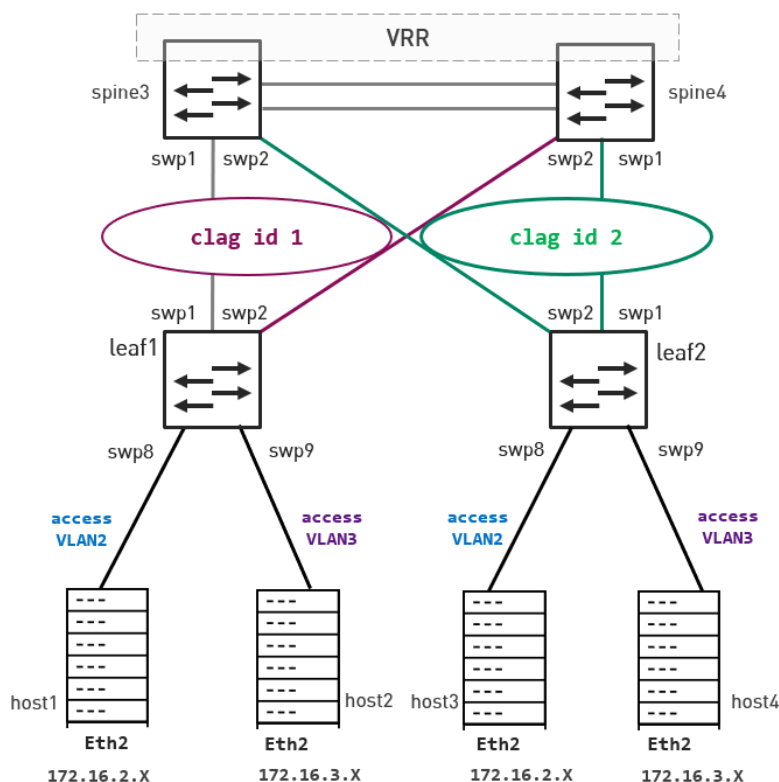
Practice 4 is now available in two flavors, the original one configuring MLAG between the spine switches and connecting the leaf's "dual-homed". A new option Practice 4 N (New) moves the MLAG connection to the leaf's and allows two hosts to multi-home.

Practice Objectives:

In this practice session you will configure the spine switches, **spine3** and **spine4**, with MLAG and VRR towards the leaf switches.

- MLAG will make the spine switches to look and behave like a single Layer 2 switch towards the Layer 2 network.
- VRR will make the spine switches to look and behave like a single router providing default gateway redundancy.
- Interface '**clag 1**' will aggregate **swp1** on **spine3** and **swp2** on **spine4** connected to switch **leaf1**.
Switch **leaf1** will be configured with a regular LAG.
- Interface '**clag 2**' will aggregate **swp2** on **spine3** and **swp1** on **spine4** connected to switch **leaf2**.
Switch **leaf2** will be configured with a regular LAG.

Topology used in this lab:



Task 1: LAG Configuration – Leaf Switches

- a. Access the leaf switches **leaf1** and **leaf2** and reset configuration:

```
$ nv config apply empty
$ nv set system hostname [leaf1|leaf2]
```

- b. Create a bridge

```
$ nv set bridge domain br_default
```

- c. Create a bond named '**BOND_TO_SPINES**', where bonds slaves are interfaces **swp1** and **swp2**.

```
$ nv set interface BOND_TO_SPINES type bond
$ nv set interface BOND_TO_SPINES bond member swp1-2
```

- d. Verify bonds configuration:

```
cumulus@leaf1:mgmt:~$ net show interface bonds
  Name          Speed  MTU  Mode      Summary
--  -
DN  BOND_TO_SPINES  N/A    9216  802.3ad  Bond Members: swp1(UP), swp2(UP)

cumulus@leaf2:mgmt:~$ net show interface bondmems
  Name  Speed  MTU  Mode      Summary
--  -
UP  swp1   1G    9216  LACP-UP   Master: BOND_TO_SPINES(DN)
UP  swp2   1G    9216  LACP-DOWN Master: BOND_TO_SPINES(DN)
```

Please note:

The bond interface is down because its peer (the MLAG peer) was not configured yet.

- e. Add the bond interface, **BOND-TO-SPINES**, and the host facing interfaces, **swp8** and **swp9**, to the bridge:

```
$ nv set interface BOND_TO_SPINES bridge domain br_default
```

- f. Configure VLANs 2-3 and associate **swp8** to **VLAN2** and **swp9** to **VLAN3**:

- g. `$ nv set interface swp8 bridge domain br_default access 2`

h. `$ nv set interface swp9 bridge domain br_default access 3`

Task 2: Configuring MLAG – spine switches

- a. Access the spine switches and reset the configuration:
 - Write down, on a side note, notepad or else, the IP addresses of the management ports, **eth0**. Those IP addresses will be configured as the MLAG

```
cumulus@spine3:mgmt:~$ ip -br a | grep eth0
eth0                UP                192.168.200.4/24 fe80::4638:39ff:fe00:20/64

cumulus@spine3:mgmt:~$ ip l | grep eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master mgmt state UP
mode DEFAULT group default qlen 1000
```

```
cumulus@spine4:mgmt:~$ ip -br a | grep eth0
eth0                UP                192.168.200.5/24 fe80::4638:39ff:fe00:22/64

cumulus@spine4:mgmt:~$ ip l | grep eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master mgmt state UP
mode DEFAULT group default qlen 1000
```

backup-IPs.

- Also verify if / which vrf is being used.
Here: 192.168.200.4 and 192.168.200.5 (both **vrf mgmt.**)

- b. Configure the spine switches, **spine3** (primary) and **spine4 (secondary)**, as MLAG peers.

```
$ nv set mlag mac-address 44:38:39:FF:00:01
$ nv set mlag backup 192.168.200.4 vrf mgmt.
$ nv set mlag init-delay 15
$ nv set mlag priority 2000
$ nv set interface peerlink bond member swp3-4
```

Please note:

There are other options to create mac-addresses for LACP.

Please note:

If you have selected your own IP address as the backup IP address, you will find the line:

Backup IP: 192.168.200.4 vrf mgmt (inactive: self)

indicating “inactive: self” as the reason.

A not yet reachable IP as the backup results in the following line:

Backup IP: 172.168.200.5 vrf mgmt (inactive)

Once you have configured both sides with a valid backup IP address, the **show** command will state “**active**”, which is the desired outcome.

```
cumulus@spine3:mgmt:~$ net show clag
The peer is not alive
  Our Priority, ID, and Role: 1000 44:38:39:00:00:01 primary
  Peer Interface and IP: peerlink.4094 fe80::4638:39ff:fe00:2 (linklocal)
    Backup IP: 192.168.200.5 vrf mgmt (active)
  System MAC: 44:38:39:ff:00:01
```

Please note:

- The init-delay is being changed for lab use, not a production best practice. For production esp. for large scale deployments consult with your account team.
- Interfaces **swp3** and **swp4** will be configured as the MLAG **peerlink**.
- Switch **spine3** is configured as the MLAG **primary** and **spine4** as the **secondary**.
- On switch **spine3** use **spine4's** IP as the backup-IP and vice-versa.

- c. Configure bridge settings – VLANs and STP priority: on both spines:

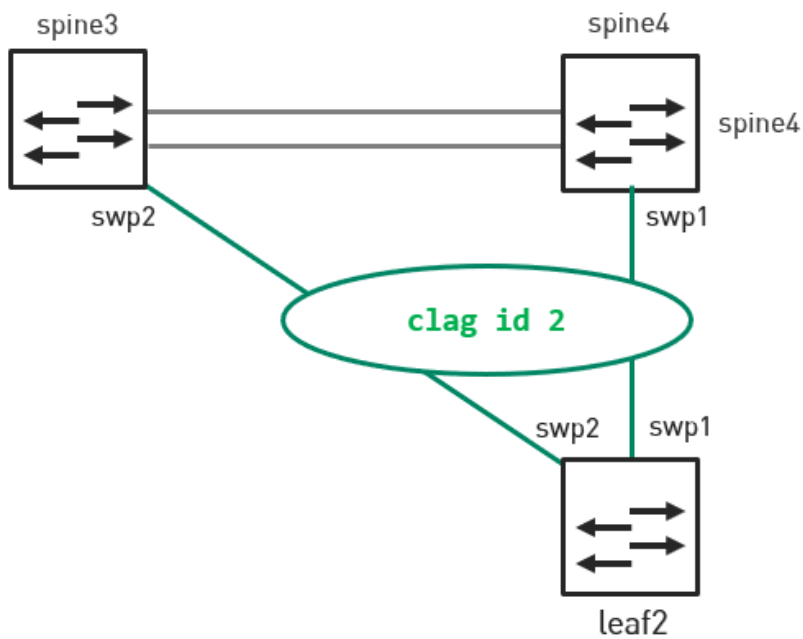
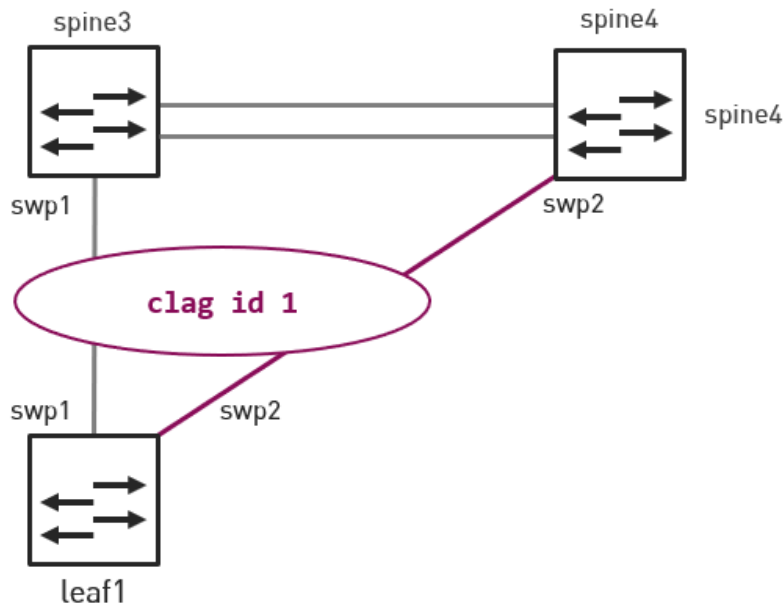
```
cumulus@spine3:mgmt:~$ nv set bridge domain br_default vlan 2-3
cumulus@spine3:mgmt:~$ nv set bridge domain br_default stp priority 4096
```

- d. Apply the changes:

```
cumulus@spine3:mgmt:~$ nv config apply
```

Task 3: Configuring CLAG interfaces - spine switches

- a. Configure two CLAG interfaces on each of the spine switches.
 - Interface '**clag 1**' will aggregate **swp1** on **spine3** and **swp2** on **spine4** which are connected to switch **leaf1**



- Interface '**clag 2**' will aggregate **swp2** on **spine3** and **swp1** on **spine4** which are connected to switch **leaf2**. For spine3:

```
$ nv set interface LEAF1 bond member swp1
$ nv set interface LEAF2 bond member swp2
$ nv set interface LEAF1-2 type bond
$ nv set interface LEAF1 bond mlag id 1
$ nv set interface LEAF2 bond mlag id 2
$ nv set interface LEAF1-2 bridge domain br_default

$ nv config apply
```

Working output:

```
cumulus@spine3:mgmt:~$ net show clag
The peer is alive
  Our Priority, ID, and Role: 1000 44:38:39:00:00:01 primary
  Peer Priority, ID, and Role: 2000 44:38:39:00:00:02 secondary
    Peer Interface and IP: peerlink.4094 fe80::4638:39ff:fe00:2 (linklocal)
      Backup IP: 192.168.200.5 vrf mgmt (active)
      System MAC: 44:38:39:ff:00:01

CLAG Interfaces
Our Interface  Peer Interface  CLAG Id  Conflicts  Proto-Down Reason
-----
      LEAF1      LEAF1        1         -           -
      LEAF2      LEAF2        2         -           -
```

b. Verify which interfaces have been added:

```
cumulus@spine4:mgmt:~$ nv config apply
```

```
<SNIP>

auto peerlink
iface peerlink
    bond-slaves swp3 swp4
    bond-mode 802.3ad
    bond-lacp-bypass-allow no

auto peerlink.4094
iface peerlink.4094
    clagd-peer-ip linklocal
    clagd-priority 1000
    clagd-backup-ip 192.168.200.5 vrf mgmt
    clagd-sys-mac 44:38:39:FF:00:01
    clagd-args --initDelay 15

auto LEAF1
iface LEAF1
    bond-slaves swp1
    bond-mode 802.3ad
    bond-lacp-bypass-allow no
    clag-id 1

auto LEAF2
iface LEAF2
    bond-slaves swp2
    bond-mode 802.3ad
    bond-lacp-bypass-allow no
    clag-id 2

auto br_default
iface br_default
    bridge-ports peerlink LEAF1 LEAF2
    hwaddress 44:38:39:00:00:25
    bridge-vlan-aware yes
    bridge-vids 2 3
    bridge-pvid 1

<SNIP>
```

```
cumulus@spine3:mgmt:~$ nv config diff empty applied
```

```
- set:
  bridge:
    domain:
      br_default:
        stp:
          priority: 4096
        vlan:
          '2': {}
          '3': {}
  mlag:
    backup:
      192.168.200.5:
        vrf: mgmt
    enable: on
    init-delay: 15
    mac-address: 44:38:39:FF:00:01
    priority: 1000
  system:
    hostname: spine3
  interface:
    LEAF1:
      bond:
        member:
          swp1: {}
        mlag:
          id: 1
    LEAF1-2:
      bond:
        mlag:
          enable: on
      bridge:
        domain:
          br_default: {}
      type: bond
    LEAF2:
      bond:
        member:
          swp2: {}
        mlag:
          id: 2
    peerlink:
      bond:
        member:
          swp3: {}
          swp4: {}
      type: peerlink
    peerlink.4094:
      base-interface: peerlink
      type: sub
      vlan: 4094
    swp1-9:
      type: swp
```

c. View MLAG resulting configuration:
Verify which timers MLAG protocol is using by:
\$ clagctl

Expected result:

“The peer is alive” and Spine3 is the primary, Spine4 the secondary with Priority values of 1000 and 2000 respectively.

An IPv6 LLA is used for the TCP connection between the peerlink.4094 sub-interfaces.

Both bonds (LEAF1 and LEAF2) are listed twice, once under “Our Interface” and once under the “Peer Interface” column.

The Backup IP is stated as (active)

```
cumulus@spine3:mgmt:~$ clagctl -v
The peer is alive
  Our Priority, ID, and Role: 1000 44:38:39:00:00:01 primary
  Peer Priority, ID, and Role: 2000 44:38:39:00:00:02 secondary
    Peer Interface and IP: peerlink.4094 fe80::4638:39ff:fe00:2 (linklocal)
      Backup IP: 192.168.200.5 vrf mgmt (active)
      System MAC: 44:38:39:ff:00:01

CLAG Interfaces
Our Interface      Peer Interface      CLAG Id      Conflicts      Proto-Down Reason
-----
          LEAF1      LEAF1          1          -          -
          LEAF2      LEAF2          2          -          -

<SNIP>
Timer Information
Timers      Value      Time remaining
-----
initDelay      15      00:00:00
peerTimeout      20      00:00:18
reloadTimer      300      00:00:00
sendTimeout      30

<SNIP>
```

Task 4: Servers' IP settings

- a. Access the servers and configure an IP address for interface **'eth2'** (see tables below).

VLAN ID	Server	'eth2' IP Address
VLAN 2	host1	172.16.2.18/24
VLAN 3	host2	172.16.3.19/24
VLAN 2	host3	172.16.2.28/24
VLAN 3	host4	172.16.3.29/24

Configure the server's IP address and subnet mask

- b. Verify a static route entry to network 172.16.0.0/16 via the default gateway's address. Use the following table for default gateway assignment.

VLAN ID	Default gateway address
vlan 2	172.16.2.254/24
vlan 3	172.16.3.254/24

\$ sudo ip route add <NET_ADDRESS/MASK> dev <DEV> via <IP>

```
cumulus@host1:~$ sudo ip route add 172.16.0.0/16 dev eth2 via 172.16.2.254
cumulus@host1:~$ route
Kernel IP routing table
Destination        Gateway            Genmask           Flags Metric Ref    Use Iface
default            _gateway          0.0.0.0           UG    0      0      0 eth0
172.16.0.0         172.16.2.254     255.255.0.0       UG    0      0      0 eth2
172.16.2.0         0.0.0.0          255.255.255.0     U     0      0      0 eth2
192.168.200.0      0.0.0.0          255.255.255.0     U     0      0      0 eth0
```

- c. Use **'ping'** to check communication between servers in the same VLAN. For example, server **host1** and **host3** in VLAN2.

```
[cumulus@host1 ~]$ ping 172.16.2.28
PING 172.16.2.28 (172.16.2.28) 56(84) bytes of data.
64 bytes from 172.16.2.28: icmp_seq=1 ttl=64 time=2.60 ms
64 bytes from 172.16.2.28: icmp_seq=2 ttl=64 time=1.83 ms
```

- d. Verify that the MLAG switches have their MAC address tables synchronized.

```
cumulus@spine3:mgmt:~$ net show bridge macs
```

VLAN	Master	Interface	MAC	TunnelDest	State	Flags	LastSeen
2	bridge	LEAF1	44:38:39:00:00:11				00:00:49
2	bridge	LEAF2	44:38:39:00:00:15				00:00:52
3	bridge	LEAF1	44:38:39:00:00:13				00:02:05
3	bridge	LEAF2	44:38:39:00:00:17				00:02:05

```
cumulus@spine4:mgmt:~$ net show bridge macs
```

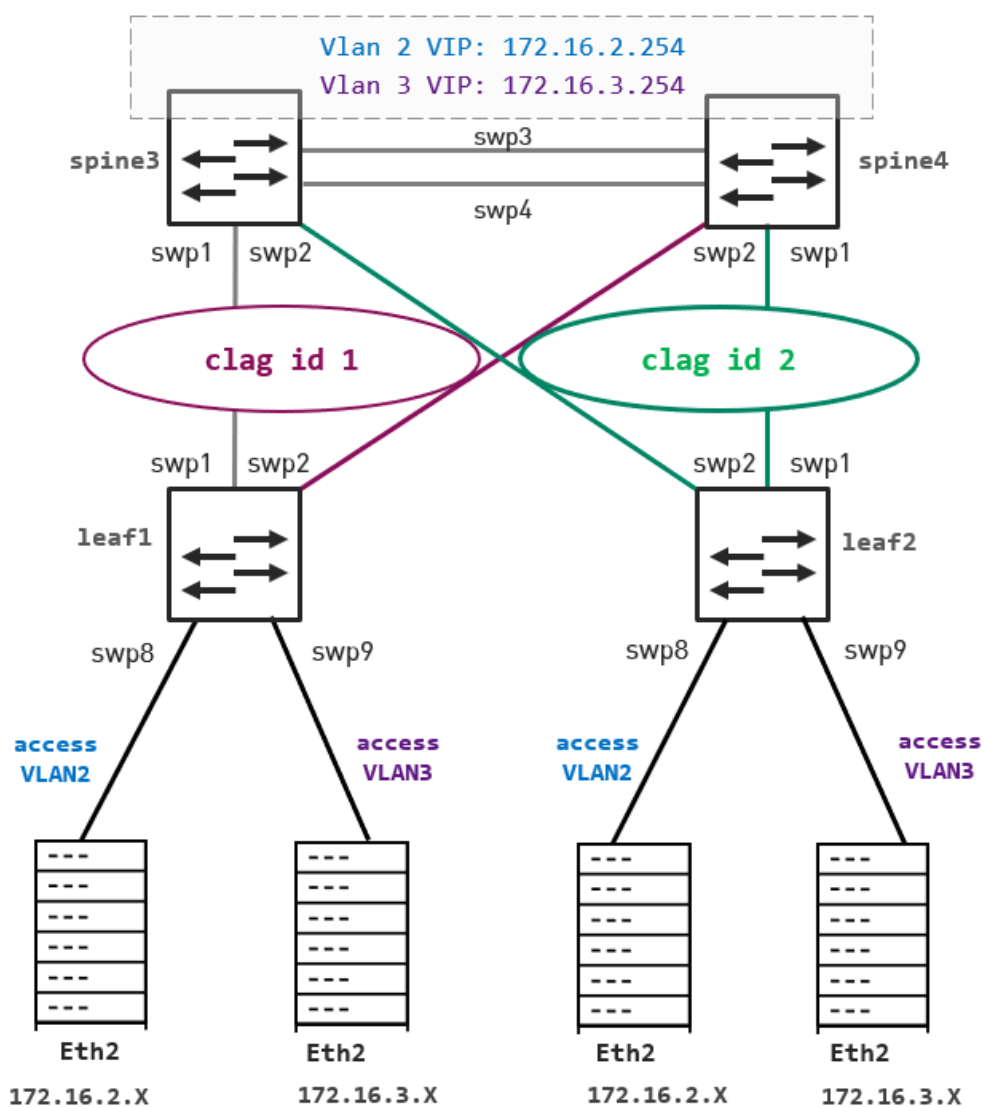
VLAN	Master	Interface	MAC	TunnelDest	State	Flags	LastSeen
2	bridge	LEAF1	44:38:39:00:00:11				00:00:49
2	bridge	LEAF2	44:38:39:00:00:15				00:00:52
3	bridge	LEAF1	44:38:39:00:00:13				00:02:05
3	bridge	LEAF2	44:38:39:00:00:17				00:02:05

Please note:

- Once the MLAG configuration is completed, the spine switches appear as a single Layer 2 switch towards to Layer 2 network. Hence, they are capable to provide an efficient load balancing and better network utilization for the layer 2 network.
- In the following task additionally, the spine switches will be configured as VRR routers. Thus, they will provide default gateway redundancy and efficient load balancing towards the layer 3 network.

Task 5: Configuring VRR

Topology used in this task:



- a. Configure two SVIs (Switch Virtual Interfaces) on each of the spine switches, `interface vlan 2` and `interface vlan 3`.
- Configure an IP for each of the vlan interfaces.
 - Configure a VIP (Virtual IP) and a VMAC (Virtual MAC).
 - Use the following table for address assignment. Use /24 as the subnet mask.

SWITCH SPINE3

VLAN	SVI	VIP	VMAC
<code>vlan 2</code>	<code>172.16.2.252/24</code>	<code>172.16.2.254/24</code>	<code>00:00:5e:00:01:02</code>
<code>vlan 3</code>	<code>172.16.3.252/24</code>	<code>172.16.3.254/24</code>	<code>00:00:5e:00:01:03</code>

SWITCH SPINE4

VLAN	SVI	VIP	VMAC
<code>vlan 2</code>	<code>172.16.2.253/24</code>	<code>172.16.2.254/24</code>	<code>00:00:5e:00:01:02</code>
<code>vlan 3</code>	<code>172.16.3.253/24</code>	<code>172.16.3.254/24</code>	<code>00:00:5e:00:01:03</code>

```
$ nv set interface vlan2 ip address 172.16.2.252/24
```

```
$ nv set interface vlan2 ip vrr address 172.16.2.254/24
```

- b. Verify VRR configuration:

```
$ ip l
<SNIP>
13: vlan2@br_default: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9216 qdisc noqueue state UP
group default qlen 1000
    link/ether 44:38:39:00:00:01 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.252/24 scope global vlan2
        valid_lft forever preferred_lft forever
    inet6 fe80::4638:39ff:fe00:1/64 scope link
        valid_lft forever preferred_lft forever
```

The VRR interface is not listed because we are missing a parameter.
This might change depending on the CL version.

\$ nv set interface vlan2 ip vrr state up

```
cumulus@spine3:mgmt:~$ nv set interface vlan2 ip vrr state up
cumulus@spine3:mgmt:~$ nv config apply
Invalid config
  'interface vlan2 ip vrr' requires one of the following to be configured:
    'system global anycast-id'
    'system global anycast-mac'
    'interface vlan2 ip vrr mac-id'
    'interface vlan2 ip vrr mac-address'
```

\$ nv set interface vlan2 ip vrr mac-address 00:00:5e:00:01:02

Repeat for vlan3 as well as for spine4.

c. Verify VRR operation.

Use **'ping'** and **'traceroute'** utilities to verify communication between hosts in different VLANs.

For example, from **'host1'** in **VLAN 2** to **'host2'** in **VLAN 3**.

```
[cumulus@host1 ~]$ ping 172.16.3.19
PING 172.16.3.19 (172.16.3.19) 56(84) bytes of data.
64 bytes from 172.16.3.19: icmp_seq=1 ttl=63 time=3.83 ms
64 bytes from 172.16.3.19: icmp_seq=2 ttl=63 time=1.90 ms

[cumulus@host1 ~]$ traceroute 172.16.3.19
traceroute to 172.16.3.19 (172.16.3.19), 30 hops max, 60 byte packets
 1  172.16.2.254 (172.16.2.254)  1.511 ms  1.423 ms  1.435 ms
 2  172.16.3.19 (172.16.3.19)  2.182 ms  1.994 ms *
```

d. Verify MLAG/VRR failover.

Use continuous **'ping'** between hosts in different VLANs.

For example, from server **'host1'** in **VLAN 2** to server **'host2'** in **VLAN 3**.

While **'ping'** is running, reboot switch **spine3**. Was the traffic disrupted?

After switch **spine3** reboots, reboot switch **spine4**. Was the traffic disrupted now?

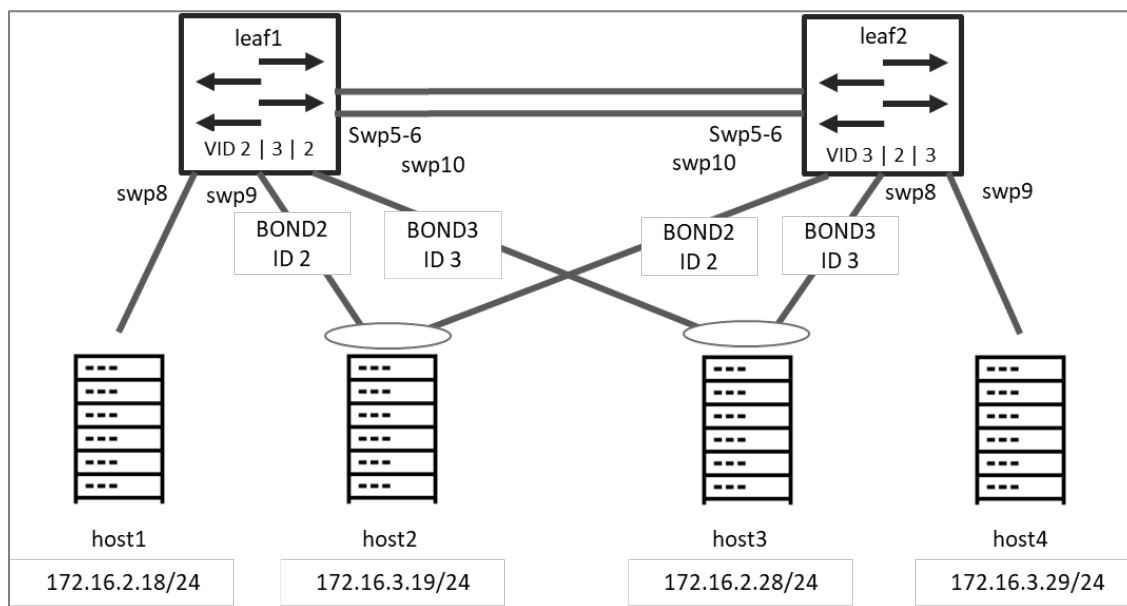
What are your conclusions regarding MLAG/VRR failover?

PRACTICE 4 NEW: CONFIGURING MLAG and VRR

Two hosts (host2 and host3) are configured with two interfaces for the fabric access.

Create a MLAG setup to dual-home host2 and host3. Verify by testing connectivity.

Topology used in this task:



The interfaces are eth2 and eth3, which need to be members of an interface of type bond.

- Create a bond interface name “bond0” via netplan using linux mode 4. Keep in mind, netplan is stateless.
- The bond0 interface of host2 should use an IPv4 address of 172.16.3.19/24 and of host3 an IPv4 address of 172.16.2.28/24.

The devices spine3 and spine4 are not used for this exercise.

- a. Configure two bonds on leaf1 and leaf2 names BOND2 and BOND3. The number identifies the connected host e.g. BOND2 connects to host2 and BOND3 connects to host3 as well as the used MLAG ID e.g. BOND2 uses ID 2 and BOND3 uses ID 3.
- b. Assign the bond interfaces as access links to the bridge br_default. BOND2 should be an access-port with VID 3 and BOND3 should be an access-port with VID 2.

Example leaf1 with BOND2:

```
cumulus@leaf1:mgmt:~$ nv config find BOND2
- set:
  interface:
    BOND2:
      bond:
        member:
          swp9: {}
        mlag:
          enable: on
          id: 2
      bridge:
        domain:
          br_default:
            access: 3
      type: bond
```

Create a MLAG setup between leaf1 and leaf2 using swp5-6 and make leaf1 the primary device. Use the management network as the path for the backup function and set the init-delay for lab use to 15 seconds.

Example leaf1:

```
- set:
  mlag:
    backup:
      192.168.200.3:
        vrf: mgmt
    enable: on
    init-delay: 15
    mac-address: 44:38:39:FF:00:01
    priority: 1000
```

complete the configuration to ensure that host3 is able to reach host1 and host2 is able to reach host4.

```
cumulus@host2:~$ ping -c 2 172.16.3.29
PING 172.16.3.29 (172.16.3.29) 56(84) bytes of data.
64 bytes from 172.16.3.29: icmp_seq=1 ttl=64 time=1.25 ms
64 bytes from 172.16.3.29: icmp_seq=2 ttl=64 time=1.03 ms

--- 172.16.3.29 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.027/1.137/1.248/0.110 ms
cumulus@host3:~$ ping -c 2 172.16.2.18
PING 172.16.2.18 (172.16.2.18) 56(84) bytes of data.
```

```
64 bytes from 172.16.2.18: icmp_seq=1 ttl=64 time=1.10 ms
64 bytes from 172.16.2.18: icmp_seq=2 ttl=64 time=1.32 ms

--- 172.16.2.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.104/1.210/1.317/0.106 ms
```

Configure SVI and VRR interfaces on leaf1 and leaf2

SWITCH LEAF1

VLAN	SVI	VIP	VMAC
vlan 2	172.16.2.252/24	172.16.2.254/24	auto
vlan 3	172.16.3.252/24	172.16.3.254/24	auto

SWITCH LEAF2

VLAN	SVI	VIP	VMAC
vlan 2	172.16.2.253/24	172.16.2.254/24	auto
vlan 3	172.16.3.253/24	172.16.3.254/24	auto

Example leaf1:

```
cumulus@leaf1:mgmt:~$ nv set interface vlan2 ip address 172.16.2.252/24
cumulus@leaf1:mgmt:~$ nv set interface vlan3 ip address 172.16.3.252/24
cumulus@leaf1:mgmt:~$ nv set interface vlan2 ip vrr address 172.16.2.254/24
cumulus@leaf1:mgmt:~$ nv set interface vlan3 ip vrr address 172.16.3.254/24
```

Verify you have intra and inter vlan connectivity:

```
cumulus@host2:~$ ping -c 2 172.16.2.18
PING 172.16.2.18 (172.16.2.18) 56(84) bytes of data.
64 bytes from 172.16.2.18: icmp_seq=1 ttl=63 time=8.30 ms
64 bytes from 172.16.2.18: icmp_seq=2 ttl=63 time=1.97 ms

--- 172.16.2.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.969/5.132/8.296/3.163 ms
cumulus@host2:~$ ping -c 2 172.16.2.28
PING 172.16.2.28 (172.16.2.28) 56(84) bytes of data.
64 bytes from 172.16.2.28: icmp_seq=1 ttl=63 time=1.21 ms
64 bytes from 172.16.2.28: icmp_seq=2 ttl=63 time=0.976 ms

--- 172.16.2.28 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.976/1.092/1.208/0.116 ms
cumulus@host2:~$ ping -c 2 172.16.3.29
PING 172.16.3.29 (172.16.3.29) 56(84) bytes of data.
64 bytes from 172.16.3.29: icmp_seq=1 ttl=64 time=0.987 ms
64 bytes from 172.16.3.29: icmp_seq=2 ttl=64 time=1.36 ms

--- 172.16.3.29 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.987/1.173/1.360/0.186 ms
```

Verify MLAG:

```
cumulus@leaf1:mgmt:~$ nv show mlag
```

	operational	applied
enable	on	on
debug	off	off
init-delay	15	15
mac-address	44:38:39:ff:00:01	44:38:39:FF:00:01
peer-ip	fe80::4ab0:2dff:fe0a:7ed0	linklocal
priority	1000	1000
[backup]	192.168.200.3	192.168.200.3
backup-active	True	
backup-reason		
backup-vrf	mgmt	
local-id	48:b0:2d:b5:dc:89	
local-role	primary	
peer-alive	True	
peer-id	48:b0:2d:0a:7e:d0	
peer-interface	peerlink.4094	
peer-priority	2000	
peer-role	secondary	

PRACTICE 5: CONFIGURING BGP UNNUMBERED

This lab starts again with a clean configuration but this time it will be the base for the next two EVPN labs and must be in a working condition before proceeding. It does however not matter if you configure BGP with legacy IP addresses or BGP unnumbered.

Practice Objectives:

In this practice session you will configure BGP Unnumbered:

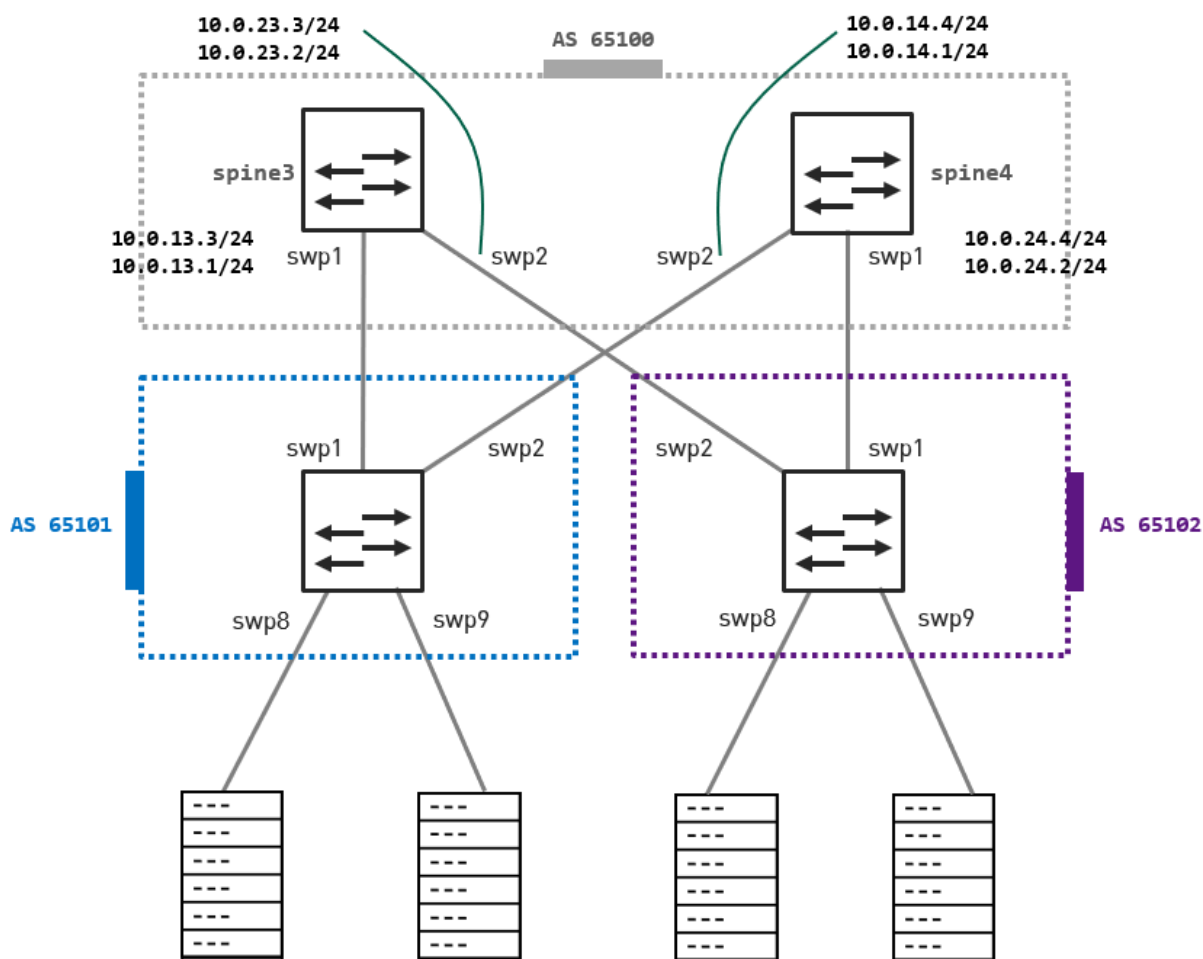
- Spine switches will be configured in the same AS and each of the leaf switches will be configured in its own AS.
- BGP unnumbered will be configured on all four switches.
- eBGP sessions will be established between the spine and leaf switches.
- Leaf switches will advertise their local IP prefixes.
- By the end of this practice session, you will achieve end-to-end connectivity between all servers in your group, over the BGP Autonomous Systems.

Please note:

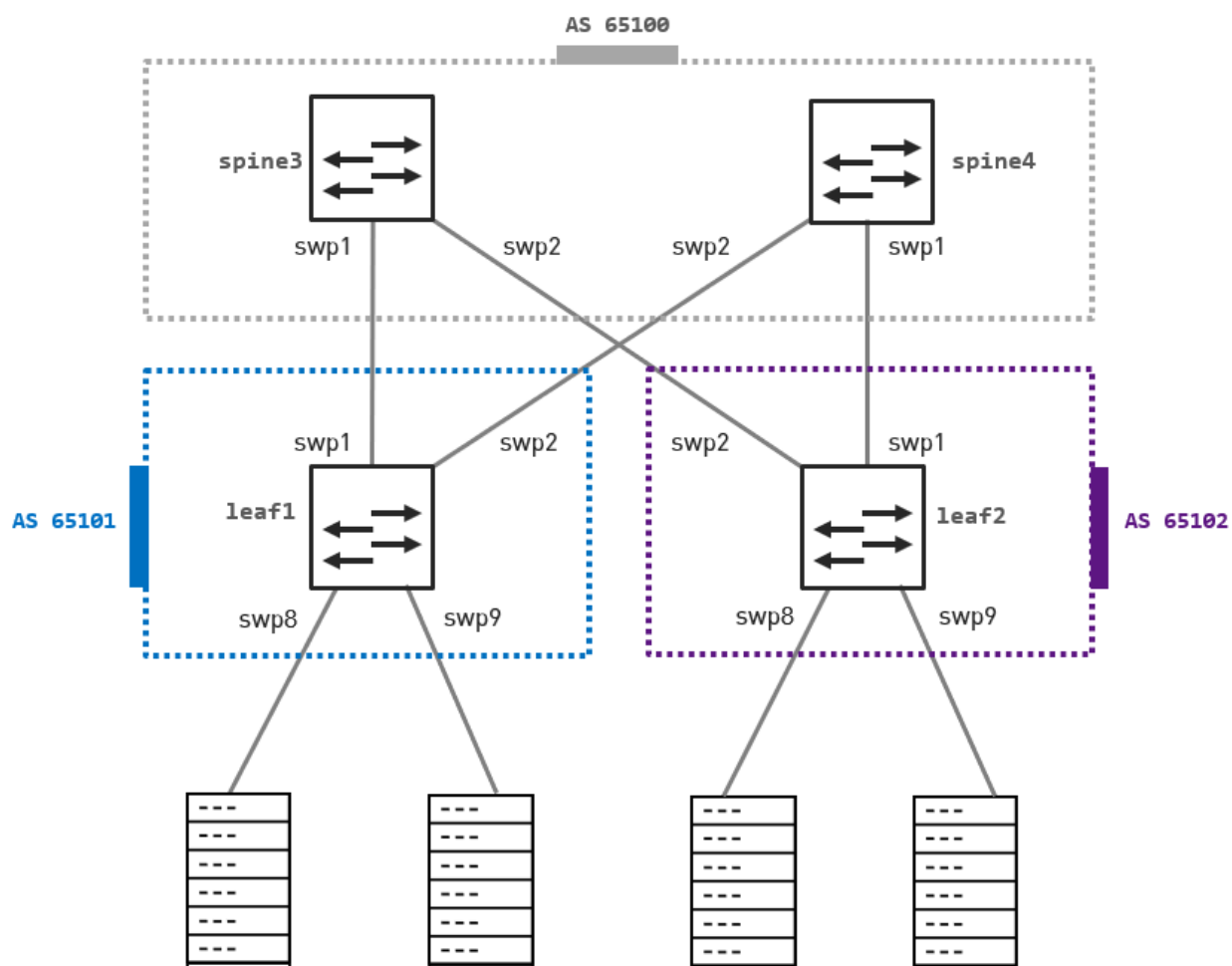
- Commands are demonstrated on **selected switches**. You should apply similar commands on the other two switches in your group.

Alternative lab:

You can configure BGP via the classical approach by using IPv4 Addresses on point-to-point links between the spines and leaf's. For teaching purposes we use a /24 network but in production a /30 or /31 is often preferred.



Topology used in this practice session covering BGP unnumbered:



Task 1: Starting FRR

- a. Access the switches, reset the configuration, and set the host names:

```
# nv config apply empty
# nv set system hostname [leaf1|leaf2|spine3|spine4]
```

Task 2: Configuring loopback interfaces and enable interfaces between the spines and leafs:

- a. On each of the switches configure a loopback interface.

Use the following table for IP address assignment:

Switch	Loopback IP address
leaf1	172.16.100.1/32
leaf2	172.16.100.2/32
spine3	172.16.100.3/32
spine4	172.16.100.4/32

```
$ nv set interface lo ip address 172.16.100.1/32
$ nv config apply
```

Enable the interfaces between the spines and leaf's as Layer 3 interfaces.

```
$ nv set interface swp1-2
$ nv config apply
```

Task 3: Configuring BGP Unnumbered

- a. Configure BGP unnumbered (the three ASN used are 65100, 65101, and 65102):

```
cumulus@leaf1:mgmt:~$ nv set router bgp autonomous-system 65101
cumulus@leaf1:mgmt:~$ nv set router bgp router-id 172.16.100.1
cumulus@leaf1:mgmt:~$ nv set vrf default router bgp neighbor swp1 remote-as external
cumulus@leaf1:mgmt:~$ nv set vrf default router bgp neighbor swp2 remote-as external
```

- b. Advertise the loopback addresses and apply the changes.

```
cumulus@leaf1:mgmt:~$ nv set vrf default router bgp address-family ipv4-unicast network
172.16.100.1/32
cumulus@leaf1:mgmt:~$ nv config apply
```

After you configured the first switch for bgp you can verify the state of the FSM and the number (zero) of prefixes exchanged:

```
cumulus@leaf1:mgmt:~$ net show bgp sum
show bgp ipv4 unicast summary
=====
BGP router identifier 172.16.100.1, local AS number 65101 vrf-id 0
BGP table version 1
RIB entries 1, using 200 bytes of memory
Peers 2, using 46 KiB of memory

Neighbor      V      AS  MsgRcvd  MsgSent  TblVer  InQ  OutQ  Up/Down  State/PfxRcd  PfxSnt
swp1          4        0        0        0        0    0    0    never      Idle           0
swp2          4        0        0        0        0    0    0    never      Idle           0

Total number of neighbors 2
```

Please note:

Commands are demonstrated on one switch, all four switches should run and all loopback addresses should be known by all switches (one local and three remote).

- c. Once you have configured all four devices you can verify configuration via net show commands or switch to the FRR provided CLI:

```
$ sudo vtysh
$ show run
$ show ip bgp sum
```

you should see two **neighbors** and a number **!=0** in the **PfxSnt** column

```
spine4# show ip bgp summary

IPv4 Unicast Summary:
BGP router identifier 172.16.100.4, local AS number 65100 vrf-id 0
BGP table version 3
RIB entries 5, using 1000 bytes of memory
Peers 2, using 46 KiB of memory

Neighbor      V      AS  MsgRcvd  MsgSent  TblVer  InQ  OutQ  Up/Down  State/PfxRcd  PfxSnt
leaf2(swp1)   4      65102      7        7        0     0     0  00:00:04          1         3
leaf1(swp2)   4      65101      9        8        0     0     0  00:00:07          1         3

Total number of neighbors 2

spine4# show ip bgp summary

IPv4 Unicast Summary:
```

You can also verify the RIB via “show ip route” within the FRR CLI

```
spine4# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure
B>* 172.16.100.1/32 [20/0] via fe80::4638:39ff:fe00:c, swp2, weight 1, 00:01:52
B>* 172.16.100.2/32 [20/0] via fe80::4638:39ff:fe00:a, swp1, weight 1, 00:01:49
C>* 172.16.100.4/32 is directly connected, lo, 00:01:56
```

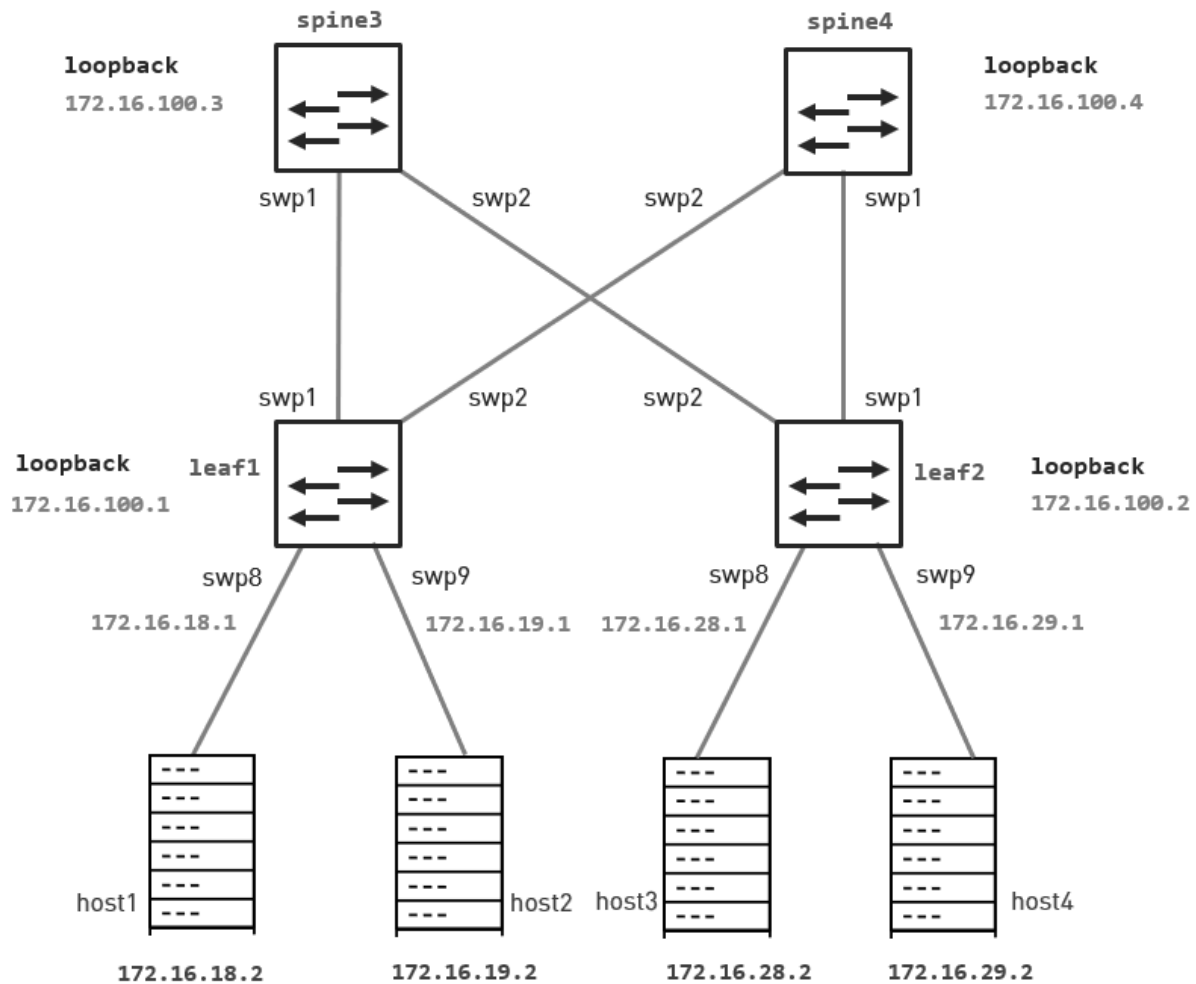
Verify loopback prefixes are reachable:

Please note:

- NEXT-HOP is the neighbor’s IPv6 Link Local Address
- BGP multipath is enabled

Task 4: Advertise local IP prefixes

- On the leaf switches, assign IP addresses to the host-facing interfaces, **swp8** and **swp9**.
Advertise the IP prefixes in the BGP process.
Use the IP addresses listed in the below topology. Use 24 bits for the subnet mask.



Example for leaf1:

```
$ nv set interface swp8 ip address 172.16.18.1/24
$ nv set interface swp9 ip address 172.16.19.1/24
$nv set vrf default router bgp address-family ipv4-unicast network 172.16.18.0/24
$nv set vrf default router bgp address-family ipv4-unicast network 172.16.19.0/24
$nv config apply
```

b. Verify that the remote leaf switch has learned the IP prefixes:

- Switch **leaf1** should have in its routing table the IP prefixes advertise by switch **leaf2**:
 - 172.16.28.0/24
 - 172.16.29.0/24

Intermediate state after configuring the first leaf, you see the local networks in the BGP Topology Table:

```
cumulus@leaf1:mgmt:~$ net show bgp
show bgp ipv4 unicast
=====
BGP table version is 9, local router ID is 172.16.100.1, vrf id 0
Default local pref 100, local AS 65101
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop              Metric LocPrf Weight Path
*> 172.16.18.0/24    0.0.0.0                  0         32768 i
*> 172.16.19.0/24    0.0.0.0                  0         32768 i
*> 172.16.100.1/32  0.0.0.0                  0         32768 i

Displayed 3 routes and 3 total paths
```

Switch **leaf2** should have in its routing table the IP prefixes advertise by switch **leaf1**:

- 172.16.18.0/24
- 172.16.19.0/24

\$ net show route bgp

```
leaf1# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure
C>* 172.16.18.0/24 is directly connected, swp8, 00:06:06
C>* 172.16.19.0/24 is directly connected, swp9, 00:06:06
B>* 172.16.28.0/24 [20/0] via fe80::4638:39ff:fe00:5, swp1, weight 1, 00:00:17
   *                      via fe80::4638:39ff:fe00:b, swp2, weight 1, 00:00:17
B>* 172.16.29.0/24 [20/0] via fe80::4638:39ff:fe00:5, swp1, weight 1, 00:00:17
   *                      via fe80::4638:39ff:fe00:b, swp2, weight 1, 00:00:17
C>* 172.16.100.1/32 is directly connected, lo, 2d18h27m
B>* 172.16.100.2/32 [20/0] via fe80::4638:39ff:fe00:5, swp1, weight 1, 00:50:33
   *                      via fe80::4638:39ff:fe00:b, swp2, weight 1, 00:50:33
B>* 172.16.100.3/32 [20/0] via fe80::4638:39ff:fe00:5, swp1, weight 1, 2d18h12m
B>* 172.16.100.4/32 [20/0] via fe80::4638:39ff:fe00:b, swp2, weight 1, 00:53:51
```

Task 5: Verify end-to-end connectivity

- Configure the servers IP settings. Configure an IP address and a subnet mask for interface '**eth2**'. Use the following tables for IP address assignment. Use /24 as the subnet mask.

Server	'eth2' IP Address	Next Hop to 172.16.0.0/16
host1	172.16.18.2	172.16.18.1
host2	172.16.19.2	172.16.19.1
host3	172.16.28.2	172.16.28.1
host4	172.16.29.2	172.16.29.1

- b. Add a route entry to 172.16.0.0/16 via interface **'eth2'**:

\$ sudo ip route add <ADDRESS/MASK> dev <DEV> via <IP_ADDRESS>

```
cumulus@host1:~$ sudo ip route add 172.16.0.0/16 dev eth2 via 172.16.18.1
```

```
cumulus@host1:~$ route
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
default          _gateway        0.0.0.0          UG    0      0      0 eth0
172.16.0.0       172.16.18.1     255.255.0.0      UG    0      0      0 eth2
172.16.2.0       0.0.0.0         255.255.255.0    U     0      0      0 eth2
192.168.200.0    0.0.0.0         255.255.255.0    U     0      0      0 eth0
```

Or via netplan, for example:

<SNIP>

```
routes:
- to: 172.16.0.0/16
  via: 172.16.18.1
```

- c. Use **'ping'** and **'traceroute'** utilities to verify communication between servers in different Autonomous Systems.

For example, ping from server **'host1'** (AS 65101) to server **'host3'** (in AS 65102).

```
[cumulus@host1 ~]# ping 172.16.28.2
PING 172.16.28.2 (172.16.28.2) 56(84) bytes of data.
64 bytes from 172.16.28.2: icmp_seq=1 ttl=61 time=4.04 ms
64 bytes from 172.16.28.2: icmp_seq=2 ttl=61 time=2.56 ms
```

```
[cumulus@host1 ~]# traceroute 172.16.28.2
traceroute to 172.16.28.2 (172.16.28.2), 30 hops max, 60 byte packets
 1 172.16.18.1 (172.16.18.1) 0.761 ms 0.927 ms 0.899 ms
 2 172.16.100.3 (172.16.100.3) 1.433 ms 1.354 ms 1.372 ms
 3 172.16.100.2 (172.16.100.2) 2.449 ms 2.058 ms 2.493 ms
 4 172.16.28.2 (172.16.28.2) 3.215 ms 3.244 ms 3.348 ms
```

Please note:

The next practice relies on BGP Unnumbered configuration. Please make sure to save the configuration before exiting.

\$ nv config save

Advanced Question:

Why can't you ping from the loopback address of spine3 to the loopback address of spine4. However, both spines are reachable from either leaf switch?

Advanced Tasks:

Make the ping work from the loopback address of spine3 to the loopback address of spine4.

PRACTICE 6: CONFIGURING VXLAN WITH EVPN

Practice Objectives:

In this practice session you will configure VXLAN with EVPN:

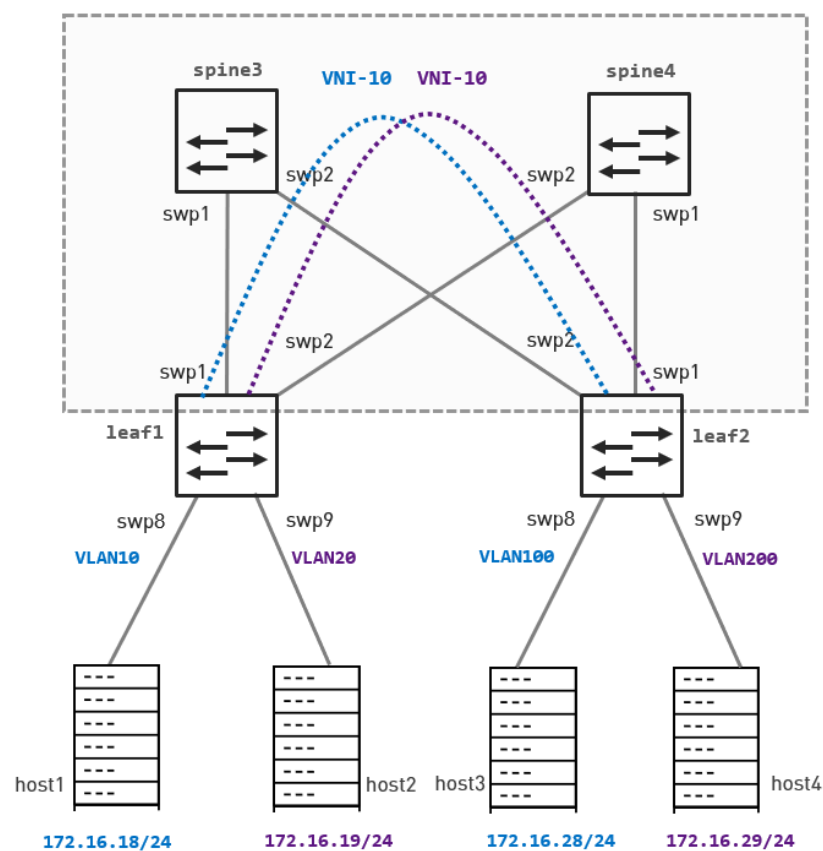
- The leaf switches will be configured as the VTEPs
- EVPN will be configured as the VXLAN control plane
- Two VXLAN Network IDs (VNIs) will be configured:
 - **VNI-10** will connect **VLAN 10** on switch **leaf1** and **VLAN 100** on switch **leaf2**.
 - **VNI-20** will connect **VLAN 20** on switch **leaf1** and **VLAN 200** on switch **leaf2**.
- Hosts in each VNI will be able to communicate in layer 2 over the underlay layer 3 network.

Please note:

The configuration in this practice session relies on the previous practice.

Make sure that BGP Unnumbered (or numbered) is properly configured and fully operational.

Topology used in this practice session:



Task 1: Configuring servers IP settings

- a. Configure an IP address and a subnet mask for interface '**eth2**'.
Use the following tables for IP address assignment. Use /24 as the subnet mask.

Server	ETH2 IP Address
host1	172.16.10.18
host2	172.16.20.19
host3	172.16.10.28
host4	172.16.20.29

- Clear existing IP configuration:
\$ sudo ifconfig <DEV> 0.0.0.0
- Configure an IP address and a subnet mask:
\$ sudo ifconfig <DEV> <IP/MASK>

Alternatively consider to use Netplan to configure your Linux hosts[1-4].

Task 2: Configuring a Bridge with two VLANs and access-ports

- a. Access the leaf switches and configure the host-facing ports **swp8** and **swp9**.
First clear any existing configuration, then assign the ports to the appropriate VLAN:
- Switch **leaf1** - assign **swp8** to **VLAN 10** and **swp9** to **VLAN 20**
 - Switch **leaf2** - assign **swp8** to **VLAN 100** and **swp9** to **VLAN 200**

```
$ nv unset interface swp8
$ nv unset interface swp9
$ nv config apply
```

Interim state, interfaces swp8-9 are unconfigured:

cumulus@leaf1:mgmt:~\$ net show int						
State	Name	Spd	MTU	Mode	LLDP	Summary
UP	lo	N/A	65536	Loopback		IP: 127.0.0.1/8
	lo					IP: 172.16.100.1/32
	lo					IP: ::1/128
UP	eth0	1G	1500	Mgmt		Master: mgmt(UP)
	eth0					IP: 192.168.200.2/24(DHCP)
UP	swp1	1G	9216	Default		
UP	swp2	1G	9216	Default		
UP	mgmt	N/A	65536	VRF		IP: 127.0.0.1/8
	mgmt					IP: ::1/128

Leaf1:

```
$ nv set interface swp8 bridge domain br_default access 10
$ nv set interface swp9 bridge domain br_default access 20
$ nv set bridge domain br_default vlan 10,20
$ nv config apply
```

cumulus@leaf1:mgmt:~\$ net show int						
State	Name	Spd	MTU	Mode	LLDP	Summary
UP	lo	N/A	65536	Loopback		IP: 127.0.0.1/8
	lo					IP: 172.16.100.1/32
	lo					IP: ::1/128
UP	eth0	1G	1500	Mgmt		Master: mgmt(UP)
	eth0					IP: 192.168.200.2/24(DHCP)
UP	swp1	1G	9216	Default		
UP	swp2	1G	9216	Default		
UP	swp8	1G	9216	Access/L2	host1 (44:38:39:00:00:11)	Master: br_default(UP)
UP	swp9	1G	9216	Access/L2		Master: br_default(UP)
UP	br_default	N/A	9216	Bridge/L2		
UP	mgmt	N/A	65536	VRF		IP: 127.0.0.1/8
	mgmt					IP: ::1/128

Leaf2:

```
$ nv set interface swp8 bridge domain br_default access 100
$ nv set interface swp9 bridge domain br_default access 200
$ nv set bridge domain br_default vlan 100,200
$ nv config apply
```

Verify the bridge configuration on both leafs:

```
$ bridge vlan
$ bridge link
```

Task 3: Configuring VNIs

- a. Access the leaf switches and configure the VNIs:
 - Create a VXLAN interface (NVE) and use the leaf's loopback IP address as its source
 - Map the VXLAN to a VLAN

Leaf1: VLAN 10 : VXLAN 10
 VLAN 20 : VXLAN 20
 Leaf2: VLAN 100 : VXLAN 10
 VLAN 200 : VXLAN 20

Leaf1:

```
$ nv set nve vxlan source address 172.16.100.1
$ nv set bridge domain br_default vlan 10 vni 10
$ nv set bridge domain br_default vlan 20 vni 20
$ nv config apply
```

Leaf2:

```
$ nv set nve vxlan source address 172.16.100.2
$ nv set bridge domain br_default vlan 100 vni 10
$ nv set bridge domain br_default vlan 200 vni 20
$ nv config apply
```

Task 4: Configuring EVPN

- a. Access the switches and configure EVPN.

```
$ nv set evpn enable on
$ nv set vrf default router bgp address-family l2vpn-evpn enable on
$ nv set vrf default router bgp neighbor swp1 address-family l2vpn-evpn enable on
$ nv set vrf default router bgp neighbor swp2 address-family l2vpn-evpn enable on
$ nv config apply
```

b. Verify MAC VXLAN Information.

```
$ show evpn mac vni all
```

```
leaf1# show evpn mac vni all
```

```
VNI 10 #MACs (local and remote) 2
```

```
Flags: B=bypass N=sync-neighs, I=local-inactive, P=peer-active, X=peer-proxy
```

MAC	Type	Flags	Intf/Remote	ES/VTEP	VLAN	Seq #'s
44:38:39:00:00:11	local		swp8		10	0/0
44:38:39:00:00:15	remote		172.16.100.2			0/0

```
VNI 20 #MACs (local and remote) 2
```

```
Flags: B=bypass N=sync-neighs, I=local-inactive, P=peer-active, X=peer-proxy
```

MAC	Type	Flags	Intf/Remote	ES/VTEP	VLAN	Seq #'s
44:38:39:00:00:13	local		swp9		20	0/0
44:38:39:00:00:17	remote		172.16.100.2			0/0

```
cumulus@leaf1:mgmt:~$ nv show bridge domain br_default mac-table
```

	age	bridge-domain	entry-type	interface	last-update	MAC address	src-vni	vlan	vni	Summary
0	16	br_default		swp8	50	48:b0:2d:c6:f0:b6		10		
1	123	br_default	permanent	swp8	123	48:b0:2d:df:08:43				
2	21	br_default		swp9	118	48:b0:2d:84:65:3e		20		
3	123	br_default	permanent	swp9	123	48:b0:2d:d1:2e:a1				
4	118	br_default		vxlan48	118	48:b0:2d:36:25:05		20	None	remote-dst: 172.16.100.2
5	118	br_default		vxlan48	50	48:b0:2d:6e:32:77		10	None	remote-dst: 172.16.100.2
6	123	br_default	permanent	vxlan48	123	b2:26:8f:35:12:c5			None	
7	122	br_default	permanent	vxlan48	56	00:00:00:00:00:00	20		None	remote-dst: 172.16.100.2
8	122	br_default	permanent	br_default	122	48:b0:2d:dc:d1:6d		1		

Task 5: Verify end-to-end communication

- Each VNI is a logical layer 2 network over the underlay layer 3 network. Logically there are no layer 3 hops between hosts in the same VNI.
- Use '**ping**' and '**traceroute**' utilities to verify end-to-end communication between hosts in a VNI. For example, from host '**host1**' to '**host3**' that are configured in **VNI-10**.

```
[cumulus@host1 ~]# ping 172.16.10.28
PING 172.16.10.28 (172.16.10.28) 56(84) bytes of data.
64 bytes from 172.16.10.28: icmp_seq=1 ttl=64 time=7.19 ms
```

```
[cumulus@host1 ~]# traceroute 172.16.10.28
traceroute to 172.16.10.28 (172.16.10.28), 30 hops max, 60 byte packets
 1  172.16.10.28 (172.16.10.28)  4.887 ms  4.817 ms  4.788 ms
```

Please note:

The next practice relies on VXLAN with EVPN configuration. Please make sure to save the configuration before exiting.

Advanced Task:

- To advertise the loopback address do NOT use the bgp network configuration and ensure that only the loopback address will be announced no other potentially later configured/added layer 3 interfaces.

PRACTICE 7: DISTRIBUTED SYMMETRIC VXLAN ROUTING

Practice Objectives:

In this practice session you will configure distributed symmetric VXLAN routing.

- VTEPs (leaf switches) will be configured as the distributed anycast default gateway, i.e., each VTEP will be configured with a VIP and VMAC for each of the subnets. Each VTEP will serve as the default gateway for its locally connected hosts.
- In symmetric VXLAN routing both the ingress and egress VTEPs perform the routing.
- A new special transit VNI is used for all routed VXLAN traffic, called the L3-VNI.

Please note:

The configuration in this practice session relies on the previous practice. Make sure that BGP Unnumbered and VXLAN with EVPN are properly configured and fully operational. Test and verify that hosts in the same VNI can communicate with each other.

As the FHRP we will use VRR under our SVI's.

```
- set:
  router:
    vrr:
      enable: on
  interface:
    vlan10:
      ip:
        address:
          172.16.10.252/24: {}
      vrr:
        address:
          172.16.10.254/24: {}
        enable: on
        mac-address: 00:00:00:00:00:0A
        state:
          up: {}
      type: svi
      vlan: 10
```

Adresses:

Leaf1:

SVI: VLAN10 172.16.10.252/24

VRR: 172.16.10.254/24

MAC 00:00:00:00:00:0A

Leaf2:

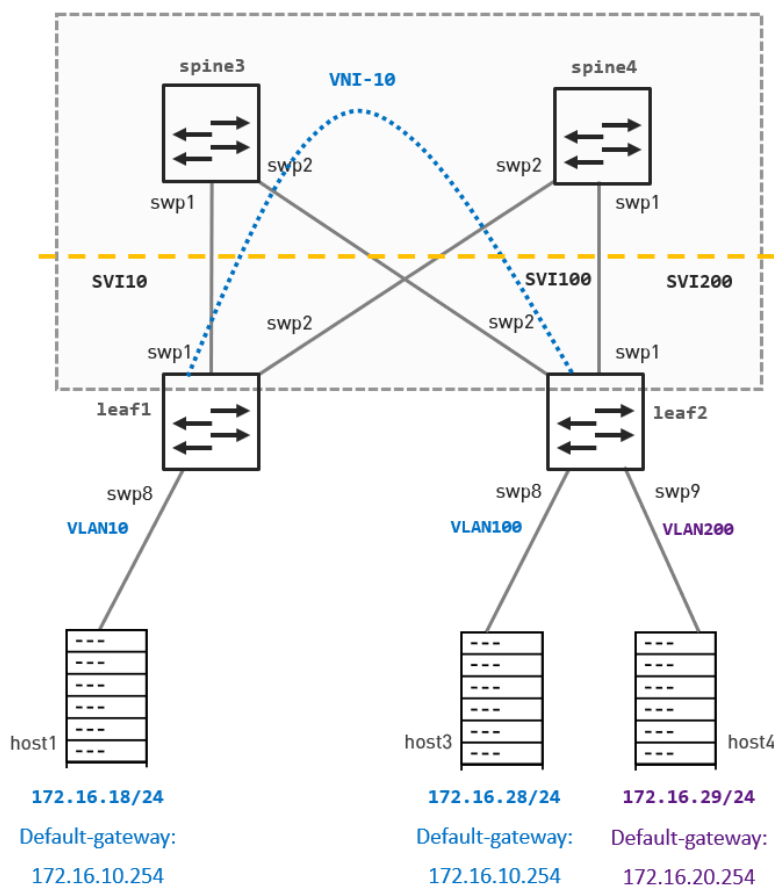
SVI: VLAN100	172.16.10.253/24
VRF:	NVIDIA
VRR:	172.16.10.254/24
MAC	00:00:00:00:00:0A
SVI: VLAN200	172.16.20.253/24
VRF:	NVIDIA
VRR:	172.16.20.254/24
MAC	00:00:00:00:00:0B

The L3-VNI construct will use the VRF named NVIDIA along with the VXLAN ID of 4001 on both leaf's.

To verify L3 Distributed Symmetric the SVI on leaf1 for vlan 20 and the vlan 20 will need to be absent. Please remove them if they are present.

Server	ETH2 IP Address
host1	172.16.10.18
host2	172.16.20.19
host3	172.16.10.28
host4	172.16.20.29

Topology used in this practice session:



Task 1: Configuring the L3-VNI

- Access both leaf switches and configure a per-tenant VXLAN interface.
Use VNI 4001.

```
$ nv set vrf NVIDIA evpn vni 4001
$ nv set interface vlan10 ip vrf NVIDIA
$ nv config apply
```

Question:

Why is the subnet 172.16.10.0/24 on leaf1 not shown in the routing table?

```
cumulus@leaf1:mgmt:~$ net show route
show ip route
=====
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
```

```
F - PBR, f - OpenFabric,  
> - selected route, * - FIB route, q - queued, r - rejected, b - backup  
t - trapped, o - offload failure  
C>* 172.16.100.1/32 is directly connected, lo, 06:48:17  
B>* 172.16.100.2/32 [20/0] via fe80::4638:39ff:fe00:5, swp1, weight 1, 00:01:40  
* via fe80::4638:39ff:fe00:b, swp2, weight 1, 00:01:40  
B>* 172.16.100.3/32 [20/0] via fe80::4638:39ff:fe00:5, swp1, weight 1, 00:01:43  
B>* 172.16.100.4/32 [20/0] via fe80::4638:39ff:fe00:b, swp2, weight 1, 00:01:40
```

Please note:

Switch leaf2 should be configured similarly but for vlan 100 and 200.

Please verify the uniqueness of the system-mac addresses between leaf1 and leaf2:

```
cumulus@leaf1:mgmt:~$ cat /run/system_mac  
44:38:39:00:00:23  
  
cumulus@leaf2:mgmt:~$ cat /run/system_mac  
44:38:39:00:00:25
```

In case the MACs are equal, please inform your instructor.

Task 3: Verify L3-VNI configuration

a. Verify that the L3-VNI is configured:

```
leaf1# show evpn vni
```

VNI	Type	VxLAN IF	# MACs	# ARPs	# Remote VTEPs	Tenant VRF
10	L2	vxlan48	2	4	1	NVIDIA
4001	L3	vxlan99	1	1	n/a	NVIDIA


```
leaf2# sh evpn vni
```

VNI	Type	VxLAN IF	# MACs	# ARPs	# Remote VTEPs	Tenant VRF
10	L2	vxlan48	2	4	1	NVIDIA
20	L2	vxlan48	1	3	0	NVIDIA
4001	L3	vxlan99	1	1	n/a	NVIDIA

```
leaf1# show evpn vni detail
```

VNI: 10
 Type: L2
 Vlan: 0
 Bridge: br_default
 Tenant VRF: NVIDIA
 VxLAN interface: vxlan48
 VxLAN ifIndex: 22
 Local VTEP IP: 172.16.100.1
 Mcast group: 0.0.0.0
 Remote VTEPs for this VNI:
 172.16.100.2 flood: HER
 Number of MACs (local and remote) known for this VNI: 2
 Number of ARPs (IPv4 and IPv6, local and remote) known for this VNI: 4
 Advertise-gw-macip: No
 Advertise-svi-macip: No

VNI: 4001
 Type: L3
 Tenant VRF: NVIDIA
 Vlan: 0
 Bridge: -
 Local Vtep Ip: 172.16.100.1
 Vxlan-Intf: vxlan99
 SVI-If: vlan444_13
 State: Up
 VNI Filter: none
 System MAC: 44:38:39:00:00:23
 Router MAC: 44:38:39:00:00:23
 L2 VNIs: 10

```
leaf1# show ip route vrf NVIDIA
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

VRF NVIDIA:
K>* 0.0.0.0/0 [255/8192] unreachable (ICMP unreachable), 01:45:23
C * 172.16.10.0/24 [0/1024] is directly connected, vlan10-v0, 01:45:23
C>* 172.16.10.0/24 is directly connected, vlan10, 01:45:23
B>* 172.16.10.28/32 [20/0] via 172.16.100.2, vlan444_l3 onlink, weight 1, 00:07:33
B>* 172.16.20.29/32 [20/0] via 172.16.100.2, vlan444_l3 onlink, weight 1, 00:07:33
```

Task 4: Test inter-VXLAN communication

- b. Use **'ping'** and **'traceroute'** utilities to verify end-to-end communication between hosts in different VNIs.

For example, ping from host **'host1'** to **'host3 and host4'**

In this case the egress switch **leaf1** will perform routing between the source VNI and the L3-VNI, while the egress switch **leaf2** will perform routing between L3-VNI and the destination VNI.

```
cumulus@host1:~$ ping 172.16.20.29
PING 172.16.20.29 (172.16.20.29) 56(84) bytes of data.
64 bytes from 172.16.20.29: icmp_seq=7 ttl=62 time=2.91 ms
64 bytes from 172.16.20.29: icmp_seq=8 ttl=62 time=3.44 ms
```

```
cumulus@host1:~$ traceroute 172.16.20.29
traceroute to 172.16.20.29 (172.16.20.29), 30 hops max, 60 byte packets
 1 172.16.10.252 (172.16.10.252) 0.588 ms 0.546 ms 0.546 ms
 2 172.16.20.253 (172.16.20.253) 2.656 ms 2.653 ms 2.652 ms
 3 172.16.20.29 (172.16.20.29) 3.431 ms 3.415 ms 3.416 ms
```

Advanced task:

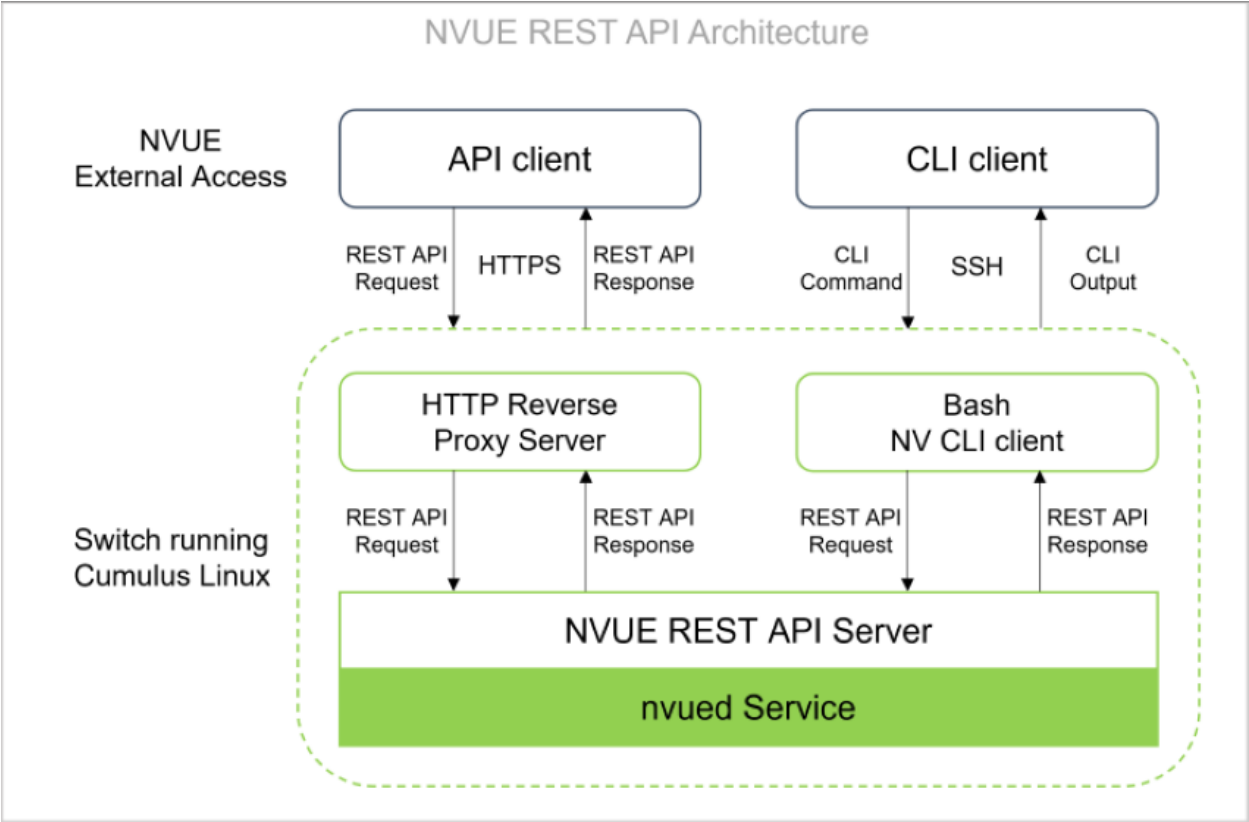
Verify the route-type transmitting the information about the remote host3 IP address.

In other words which route-type is being used.

Which information is send along this prefix

How many updates about this IP address does leaf1 receive?

PRACTICE 8: NVUE API

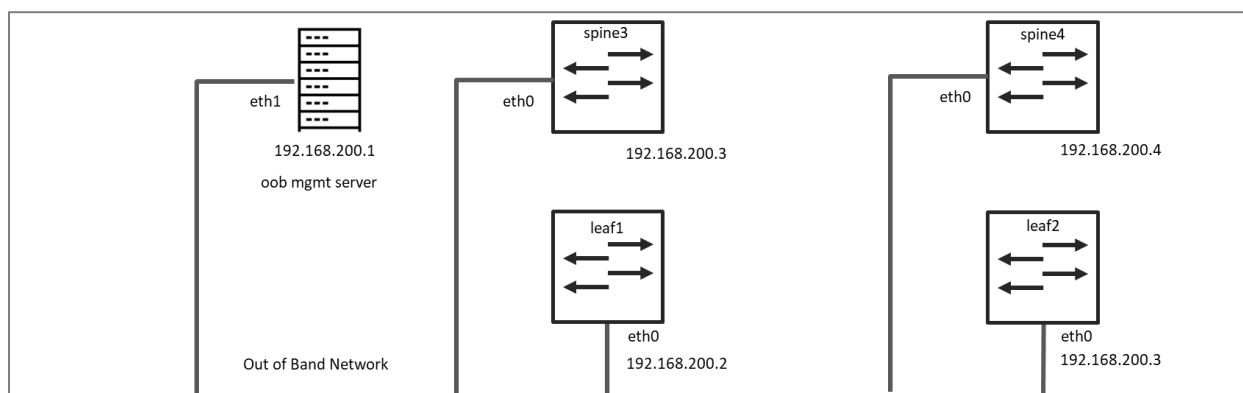


The switches should be prepared to expose the NVUE API, the documentation lists the following three configuration lines as a pre-require:

```
cumulus@switch:~$ sudo ln -s /etc/nginx/sites-{available,enabled}/nvue.conf
cumulus@switch:~$ sudo sed -i 's/listen localhost:8765 ssl;/listen \[::\]:8765
ipv6only=off ssl;/g' /etc/nginx/sites-available/nvue.conf
cumulus@switch:~$ sudo systemctl restart nginx
```

The NVUE API listens on port 8765, verify :

```
cumulus@leaf1:mgmt:~$ sudo ss -tulpn | grep LISTEN
tcp    LISTEN  0          3          127.0.0.1:2616          0.0.0.0:*
users: (("staticd",pid=30150,fd=12))
tcp    LISTEN  0          511         0.0.0.0:8765          0.0.0.0:*
users: (("nginx",pid=19577,fd=7),("nginx",pid=19576,fd=7))
tcp    LISTEN  0          64          127.0.0.1:5345          0.0.0.0:*
users: (("ltnng-runas",pid=3537,fd=39),("ltnng-consumerd",pid=3530,fd=39),("ltnng-
sessiond",pid=3444,fd=39))
tcp    LISTEN  0          3          127.0.0.1:2601          0.0.0.0:*
users: (("zebra",pid=30123,fd=60))
tcp    LISTEN  0          100         127.0.0.1:49002         0.0.0.0:*
users: (("csmgrd",pid=3476,fd=17))
tcp    LISTEN  0          3          127.0.0.1:2605          0.0.0.0:*
users: (("bgpd",pid=30143,fd=37))
tcp    LISTEN  0          128         0.0.0.0%NVIDIA:179     0.0.0.0:*
users: (("bgpd",pid=30143,fd=42))
tcp    LISTEN  0          128         0.0.0.0:179            0.0.0.0:*
users: (("bgpd",pid=30143,fd=39))
tcp    LISTEN  0          100         127.0.0.1:49012         0.0.0.0:*
users: (("csmgrd",pid=3476,fd=10))
tcp    LISTEN  0          128         0.0.0.0:22             0.0.0.0:*
users: (("sshd",pid=1488,fd=3))
tcp    LISTEN  0          128         [::]:179               [::]:*
users: (("bgpd",pid=30143,fd=40))
tcp    LISTEN  0          128         [::]%NVIDIA:179        [::]:*
users: (("bgpd",pid=30143,fd=44))
tcp    LISTEN  0          128         [::]:22                 [::]:*
users: (("sshd",pid=1488,fd=4))
```



Verify API access from your node e.g. leaf1:

```
cumulus@leaf1:mgmt:~$ curl -u 'cumulus:Academy123' --insecure
https://127.0.0.1:8765/nvue_v1/interface | grep vlan10-v0 -B 1 -A 18
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed

  0     0    0     0     0     0     0     0  --:--:-- --:--:-- --:--:--    0 },
"vlan10-v0": {
  "counters": {
    "netstat": {
      "flag": "BMRU",
      "mtu": 9216,
      "rx-drop": 0,
      "rx-error": 0,
      "rx-okay": 36,
      "rx-over": 0,
      "tx-drop": 0,
      "tx-error": 0,
      "tx-okay": 55,
      "tx-over": 0
    }
  },
  "ifindex": 34,
  "ip": {
    "address": {
      "172.16.10.254/24": {}
    }
  }
}
100 22124 100 22124    0     0 24419      0 --:--:-- --:--:-- --:--:-- 24419
```

If the API is configured to be externally accessible you can fetch the same information from the oob-mgmt-server.

```
cumulus@oob-mgmt-server:~/ON-18$ curl -u 'cumulus:Academy123' --insecure
https://192.168.200.2:8765/nvue_v1/interface | grep vlan10-v0 -B 1 -A 18
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left     Speed

100 22124 100 22124    0     0 },0      0 --:--:-- --:--:-- --:--:--    0
40711    0 --:--:-- --:--:-- --:--:-- 40744
"vlan10-v0": {
  "counters": {
    "netstat": {
      "flag": "BMRU",
      "mtu": 9216,
      "rx-drop": 0,
      "rx-error": 0,
      "rx-okay": 36,
      "rx-over": 0,
      "tx-drop": 0,
      "tx-error": 0,
      "tx-okay": 59,
      "tx-over": 0
    }
  },
  "ifindex": 34,
  "ip": {
    "address": {
      "172.16.10.254/24": {}
    }
  }
}
```