An assignment report on

VXLAN Lab 3

Submitted for fufillment of award of
Bachelor of Computing and Network Communications
degree

By
Chance Page
26/11/2024

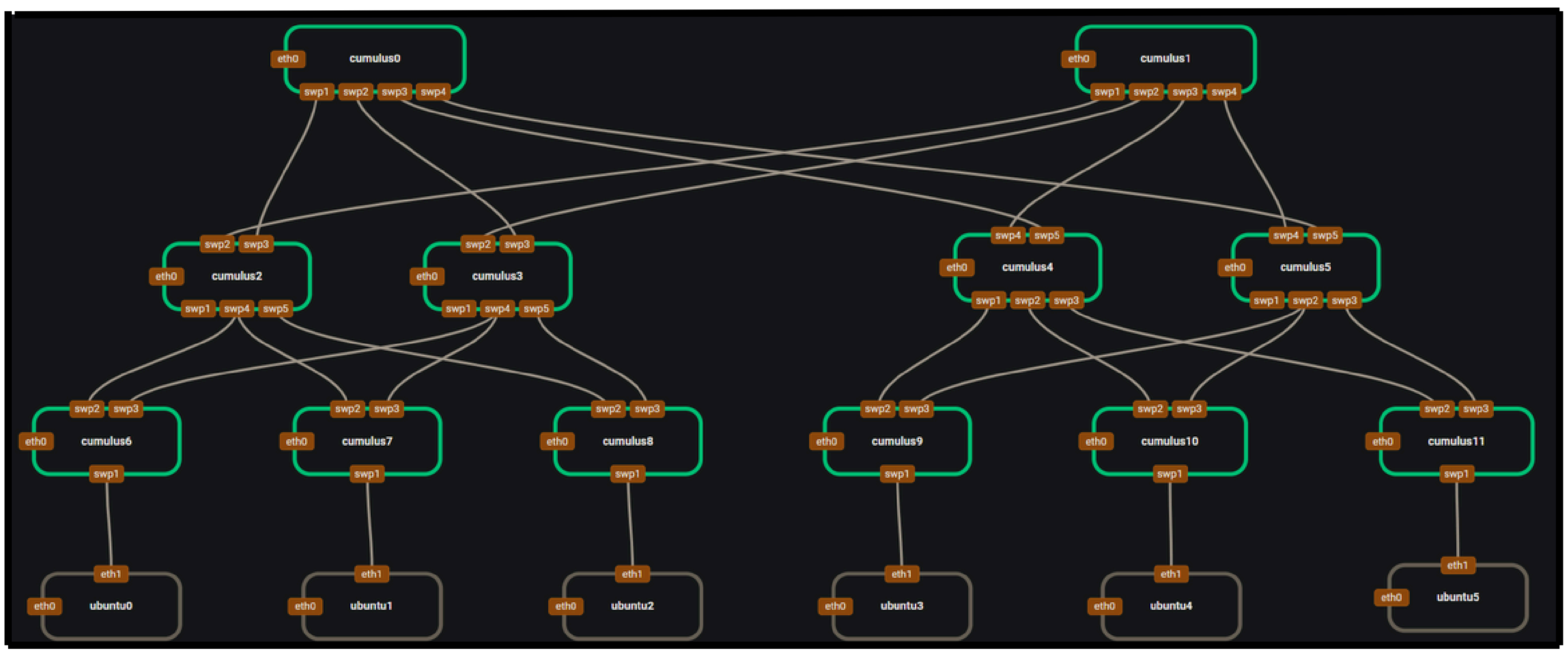
Sheridan College
Faculty of Applied Science & Technology
Professor. Felix Carapaica
Internet Core Technologies
TELE34660

Table of Contents

Introduction

In this assignment, we delve into the fundamentals of designing and implementing a scalable CLOS datacenter topology using Cumulus Linux. Understanding the basics of datacenter networking, such as VXLAN, EVPN, VLANs, and the CLOS (spine-leaf) architecture, will provide the necessary foundation to undertake this project. We will also leverage BGP (Border Gateway Protocol), which is essential for the operation of EVPN VXLAN, ensuring efficient routing and network management.

Network Topology



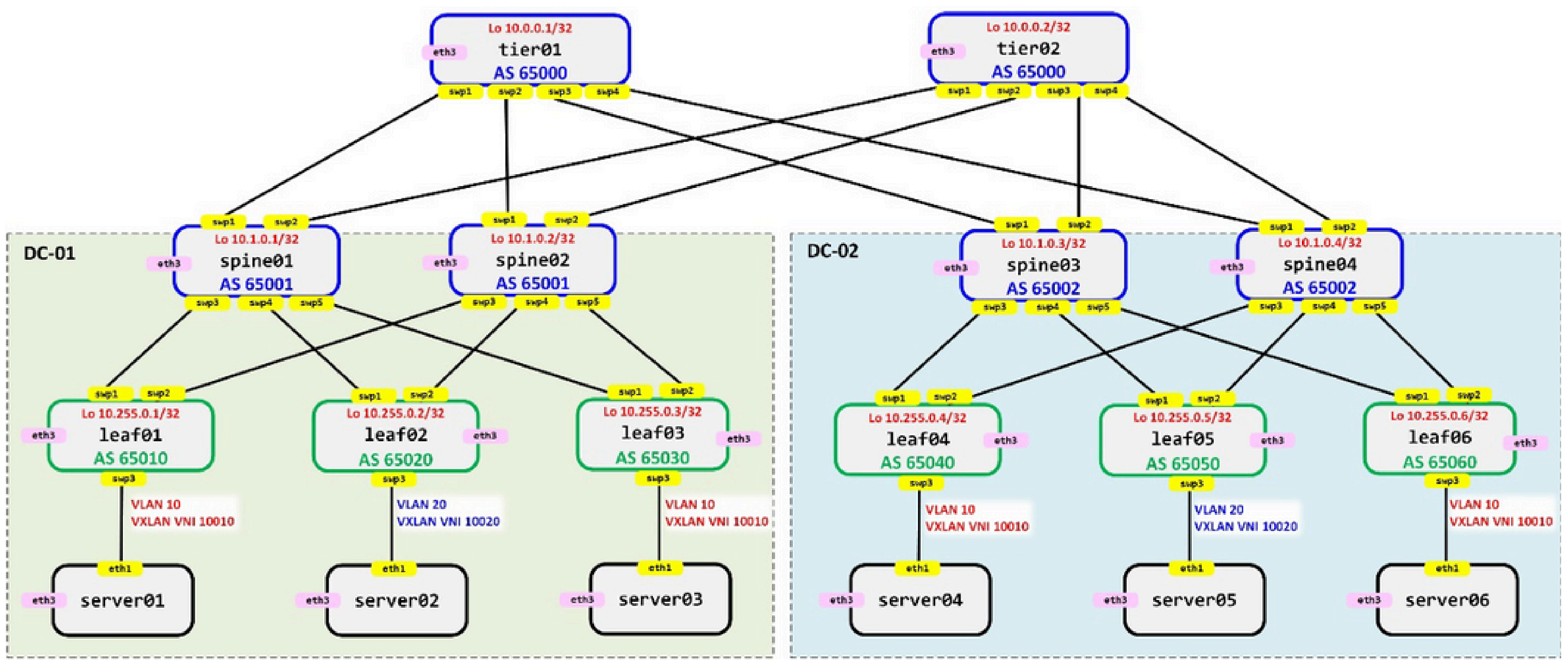
Our topology comprises a robust structure featuring 2 Superspines, which act as two separate datacenters and form the backbone of our network. We have 4 Spines, ensuring robust interconnection between the superspines and the leaf switches, and 6 Leaf Switches, providing the access layer for our servers. Additionally, we have 6 servers, segmented into two virtual networks: 4 servers on VLAN 10 VNI 10010 and 2 servers on VLAN 20 VNI 10020. We will use VLANs to segment the network traffic. By enabling EVPN VXLAN and leveraging BGP on all routers, we aim to facilitate seamless communication across the VNIs. This configuration ensures that servers can ping each other, showcasing the efficiency and scalability of our datacenter network.

Preamble

For this project, I leveraged Ansible to automate the configuration of all the spines and leafs in our network. Although the Ansible configuration was somewhat messy—enabling all interfaces by default—I focused on enabling the necessary configurations on interfaces swp1-6. I chose to use Nvidia Air over GNS3 for this setup, as Nvidia Air felt less clunky and came with pre-configured management ports, making it more suitable for Ansible automation.

Challenges

One of the most challenging parts of the assignment was manually configuring each Cumulus switch with a new password to ensure Ansible compatibility; this was necessary for the servers as well. Additionally, I encountered issues with Ansible due to YAML's strict format and syntax requirements—specifically, YAML's aversion to tabs, preferring spaces instead. This was a valuable lesson learned for future configurations. Another significant challenge was that all servers could ping each other, despite being assigned to different VNIs and VLANs. I had placed all servers in the same subnet, assuming that Layer 2 VLANs would prevent cross-communication due to the differing VNIs and VLANs. However, this did not work as expected, and I followed the same format we used in class. I was unsure how to correct this issue. In class we used different subnets for each differing VNI so this issue was never discovered.



Ansible Hosts File

```
[super_spines:vars]
ansible_user=cumulus
ansible_password=telecomS144!
ansible_become_pass=telecomS144!

[super_spines]
superspine01 ansible_host=192.168.200.2 ansible_loopback_ip=10.0.0.1
ansible_autonomous_system=65000
superspine02 ansible_host=192.168.200.3 ansible_loopback_ip=10.0.0.2
ansible_autonomous_system=65000

[spines:vars]
ansible_user=cumulus
ansible_password=telecomS144!
ansible_become_pass=telecomS144!

[spines]
spine01 ansible_host=192.168.200.4 ansible_loopback_ip=10.1.0.1 ansible_autonomous_system=65001
spine02 ansible_host=192.168.200.5 ansible_loopback_ip=10.1.0.2 ansible_autonomous_system=65001
spine03 ansible_host=192.168.200.6 ansible_loopback_ip=10.1.0.3 ansible_autonomous_system=65002
spine04 ansible_host=192.168.200.7 ansible_loopback_ip=10.1.0.4 ansible_autonomous_system=65002

[leafs:vars]
ansible_user=cumulus
ansible_password=telecomS144!
ansible_become_pass=telecomS144!

[leafs]
leaf01 ansible_host=192.168.200.8 ansible_loopback_ip=10.255.0.1 ansible_autonomous_system=65010
leaf_vlan=10 leaf_vni=10010
leaf02 ansible_host=192.168.200.9 ansible_loopback_ip=10.255.0.2 ansible_autonomous_system=65020
leaf_vlan=20 leaf_vni=10020
leaf03 ansible_host=192.168.200.10 ansible_loopback_ip=10.255.0.3 ansible_autonomous_system=65030
leaf_vlan=10 leaf_vni=10010
leaf04 ansible_host=192.168.200.11 ansible_loopback_ip=10.255.0.4 ansible_autonomous_system=65040
leaf_vlan=10 leaf_vni=10010
leaf05 ansible_host=192.168.200.12 ansible_loopback_ip=10.255.0.5 ansible_autonomous_system=65050
leaf_vlan=20 leaf_vni=10020
leaf06 ansible_host=192.168.200.18 ansible_loopback_ip=10.255.0.6 ansible_autonomous_system=65060
leaf_vlan=10 leaf_vni=10010

[ubuntu_machines:vars]
ansible_user=cumulus
ansible_password=telecomS144!
ansible_become_pass=telecomS144!

[ubuntu_machines]
ubuntu0 ansible_host=192.168.200.13
ubuntu1 ansible_host=192.168.200.14
ubuntu2 ansible_host=192.168.200.15
ubuntu3 ansible_host=192.168.200.16
ubuntu4 ansible_host=192.168.200.17
ubuntu5 ansible_host=192.168.200.19
```

Ansible Hosts File

Section: super_spines:vars
This section sets the default variables for the super spine devices. The ansible_user, ansible_password, and ansible_become_pass are defined here to ensure that Ansible can authenticate and gain elevated privileges on these devices.

Section: super_spines
This section lists the super spine devices and their specific details. Each device has an IP address (ansible_host), a loopback IP (ansible_loopback_ip), and an autonomous system number (ansible_autonomous_system). These details are essential for configuring BGP and other network settings.

Section: spines:vars
Similar to the super_spines:vars section, this defines default variables for the spine devices to ensure Ansible can properly authenticate and elevate privileges.

Section: spines
This section lists the spine devices, specifying their IP addresses, loopback IPs, and autonomous system numbers. These configurations are critical for setting up BGP routing between the spines and other network devices.

Section: leafs:vars
This section sets default variables for the leaf switches, facilitating Ansible authentication and privilege escalation.

Section: leafs
This section lists the leaf switches, with each entry detailing the device’s IP address, loopback IP, autonomous system number, VLAN, and VNI. These parameters are crucial for configuring the network segmentation and VXLAN settings.

Ansible Spines YAML

```
---
- name: Configure spine devices
  hosts: super_spines, spines
  become: yes
  gather_facts: no
  tasks:
    - name: Set system hostname
      command: nv set system hostname {{ inventory_hostname }}

    - name: Set loopback IP address
      command: nv set interface lo ip address {{ ansible_loopback_ip }}/32

    - name: Set interface state
      command: nv set interface swp1-6 link state up

    - name: Set BGP autonomous system
      command: nv set vrf default router bgp autonomous-system {{ ansible_autonomous_system }}

    - name: Set BGP path selection
      command: nv set vrf default router bgp path-selection multipath aspath-ignore on

    - name: Set BGP router ID
      command: nv set vrf default router bgp router-id {{ ansible_loopback_ip }}

    - name: Set BGP neighbor remote-as
      command: nv set vrf default router bgp neighbor swp1-6 remote-as external

    - name: Set BGP address-family
      command: nv set vrf default router bgp address-family ipv4-unicast network {{
ansible_loopback_ip }}/32

    - name: Set BGP L2VPN EVPN
      command: nv set vrf default router bgp address-family l2vpn-evpn enable on

    - name: Set BGP L2VPN EVPN Neighbor
      command: nv set vrf default router bgp neighbor swp1-6 address-family l2vpn-evpn enable on

    - name: Enable EVPN
      command: nv set evpn enable on

    - name: Apply configuration
      command: nv config apply -y
```

Ansible Leafs YAML

```
---
- name: Configure leaf devices
  hosts: leafs
  become: yes
  gather_facts: no
  tasks:
    - name: Set system hostname
      command: nv set system hostname {{ inventory_hostname }}

    - name: Set loopback IP address
      command: nv set interface lo ip address {{ ansible_loopback_ip }}/32

    - name: Set interface state
      command: nv set interface swp1-6 link state up

    - name: Set BGP autonomous system
      command: nv set vrf default router bgp autonomous-system {{ ansible_autonomous_system }}

    - name: Set BGP path selection
      command: nv set vrf default router bgp path-selection multipath aspath-ignore on

    - name: Set BGP router ID
      command: nv set vrf default router bgp router-id {{ ansible_loopback_ip }}

    - name: Set BGP neighbor remote-as
      command: nv set vrf default router bgp neighbor swp1-6 remote-as external

    - name: Set BGP address-family
      command: nv set vrf default router bgp address-family ipv4-unicast network {{
ansible_loopback_ip }}/32

    - name: Set VLAN and VNI
      command: nv set bridge domain br_default vlan {{ leaf_vlan }} vni {{ leaf_vni }}

    - name: Set VXLAN Interface
      command: nv set interface swp1 bridge domain br_default access {{ leaf_vlan }}

    - name: Set BGP L2VPN EVPN
      command: nv set vrf default router bgp address-family l2vpn-evpn enable on

    - name: Set BGP L2VPN EVPN Neighbor
      command: nv set vrf default router bgp neighbor swp1-6 address-family l2vpn-evpn enable on

    - name: Set VXLAN Source Address
      command: nv set nve vxlan source address {{ ansible_loopback_ip }}

    - name: Enable EVPN
      command: nv set evpn enable on

    - name: Apply configuration
      command: nv config apply -y
```

Topology Configuration

One of the ways I obtained all the management IPs for the devices was by accessing the management server on Nvidia Air, where I found a hosts file in /etc/ansible containing the IP addresses of each device. I then created a new hosts file, adjusted the groups, and added the necessary variables.

With the Ansible configurations ready, all I had to do was run:

```
ansible-playbook -i hosts.ini spines.yaml
```

```
ansible-playbook -i hosts.ini leafs.yaml
```

Next, I manually configured the Ubuntu servers' Netplan with the following configuration:

```
/etc/netplan/50...yaml
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: no
      dhcp6: no
      addresses:
        - 172.16.0.x/24
```

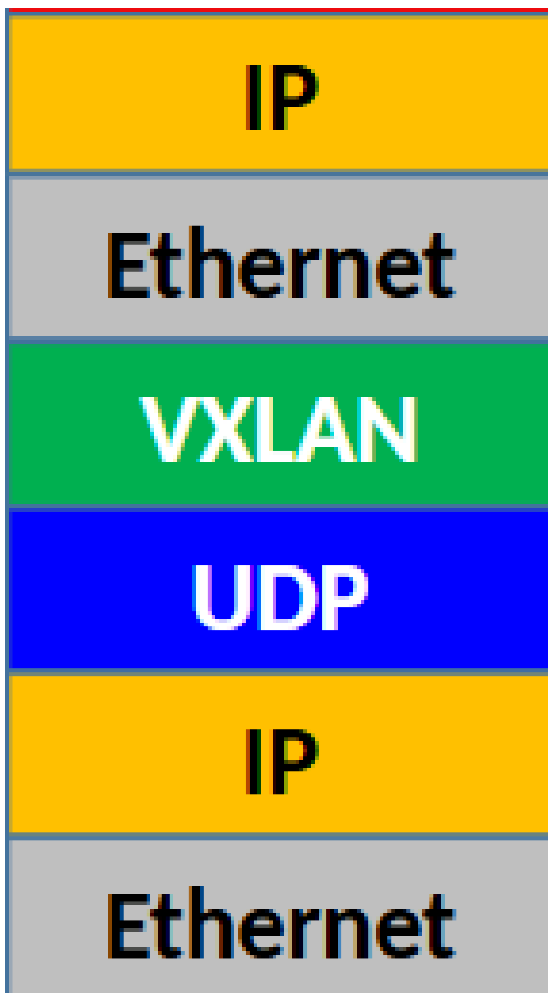
I applied the configuration using:

```
sudo netplan apply
```
Now, our topology should be working.

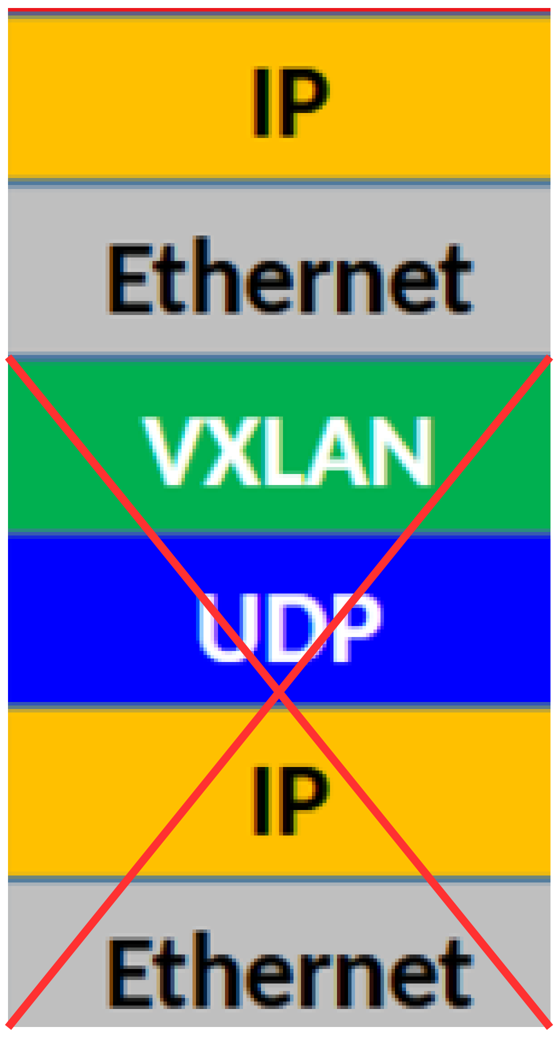
Network Overlay

Now that the topology is fully configured, here's an overview of how the servers perceive the network. The servers operate under the impression that they are all part of the same Layer 2 network, allowing them to view each other's MAC addresses. This visibility is facilitated by the Layer 2 VPN.

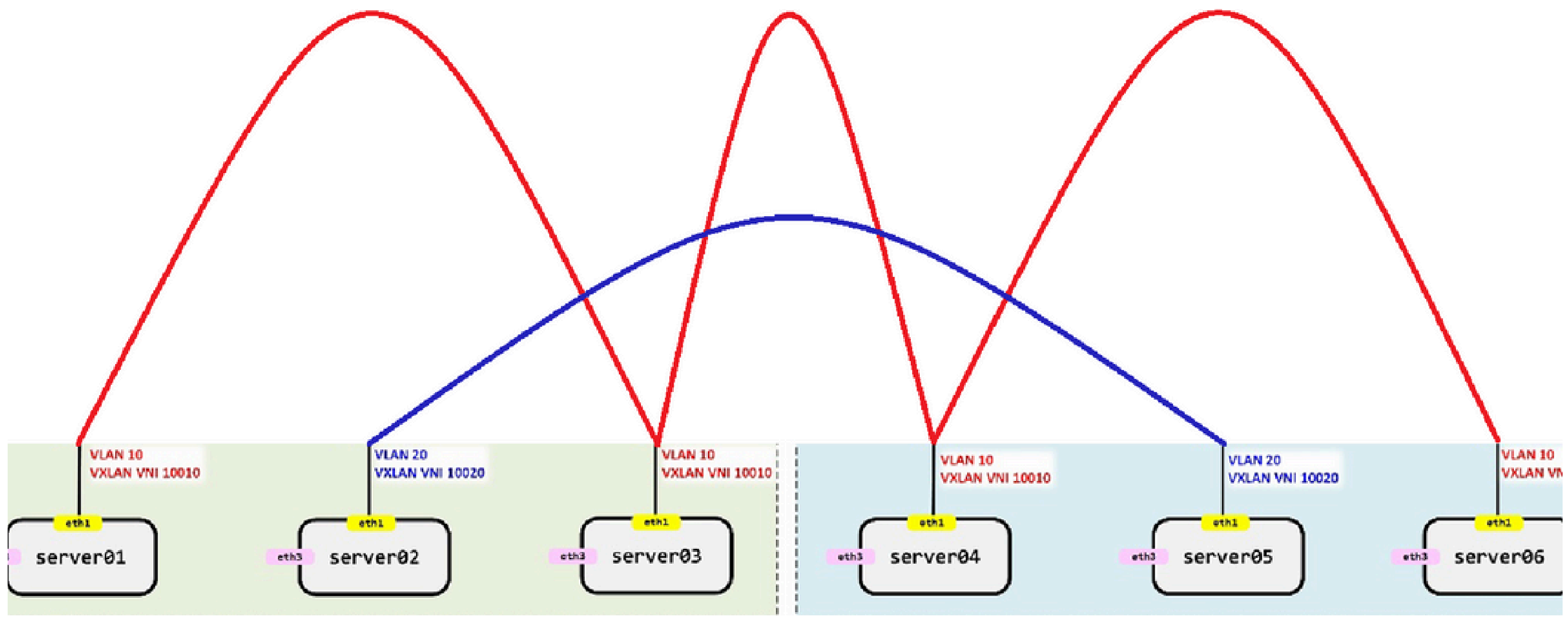
This is the stack inside the core spines and leafs.



This is the stack that the servers see.

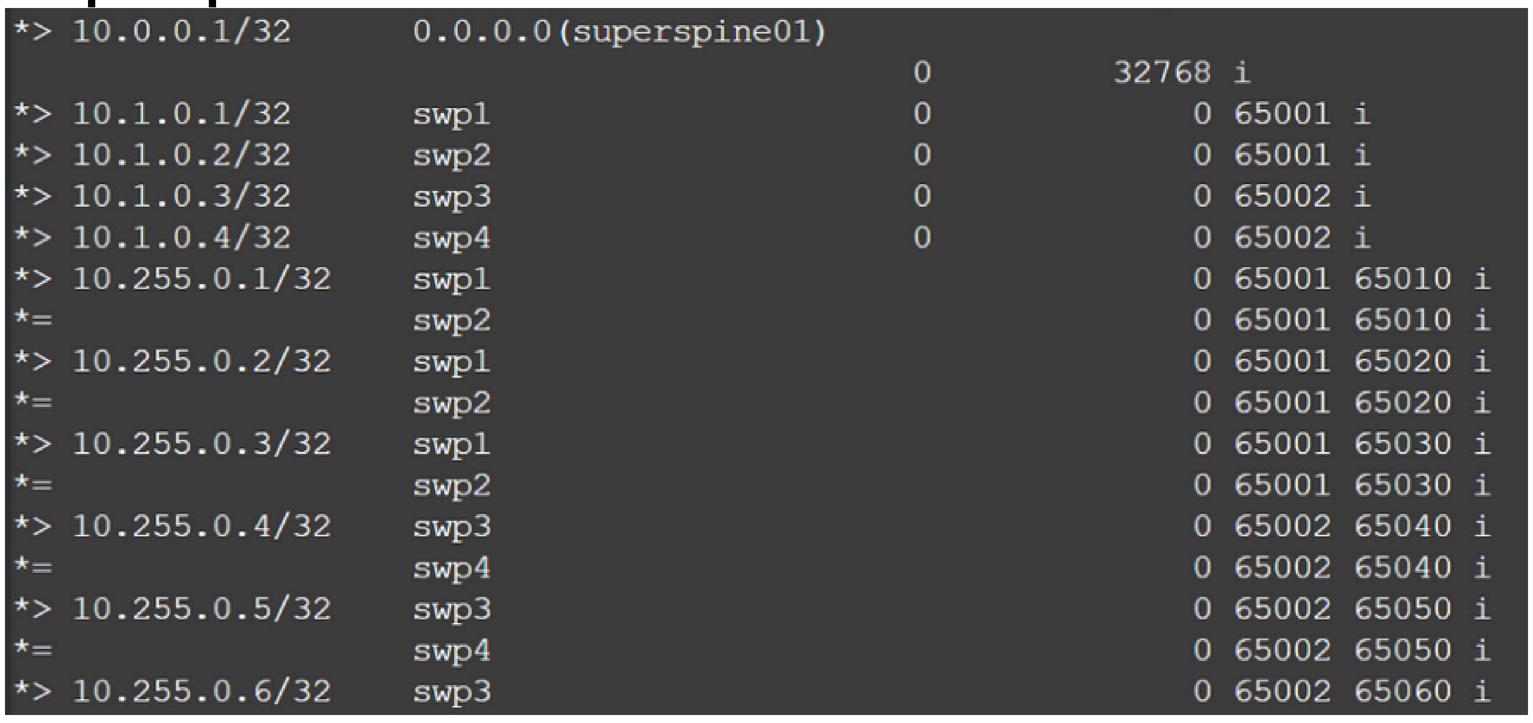


We can visualize this by removing the underlay of the network and showing just the overlay. This is the connections of the servers as they see it. They dont care about the EVPN Core.



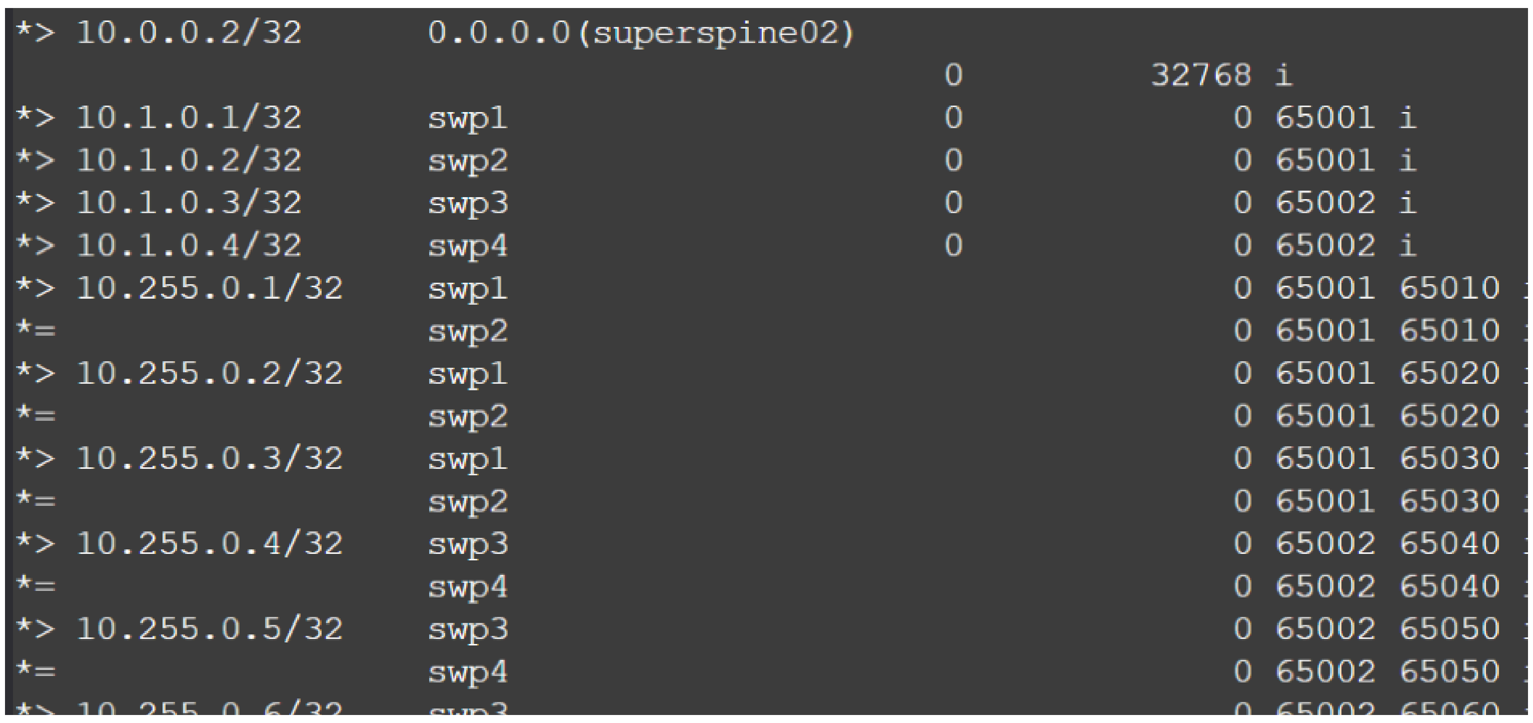
BGP Table Spines

SuperSpine01



```
*> 10.0.0.1/32      0.0.0.0(superspine01)
                                             0         32768 i
*> 10.1.0.1/32      swp1                     0             0 65001 i
*> 10.1.0.2/32      swp2                     0             0 65001 i
*> 10.1.0.3/32      swp3                     0             0 65002 i
*> 10.1.0.4/32      swp4                     0             0 65002 i
*> 10.255.0.1/32    swp1                                   0 65001 65010 i
*=                  swp2                                   0 65001 65010 i
*> 10.255.0.2/32    swp1                                   0 65001 65020 i
*=                  swp2                                   0 65001 65020 i
*> 10.255.0.3/32    swp1                                   0 65001 65030 i
*=                  swp2                                   0 65001 65030 i
*> 10.255.0.4/32    swp3                                   0 65002 65040 i
*=                  swp4                                   0 65002 65040 i
*> 10.255.0.5/32    swp3                                   0 65002 65050 i
*=                  swp4                                   0 65002 65050 i
*> 10.255.0.6/32    swp3                                   0 65002 65060 i
```

SuperSpine02

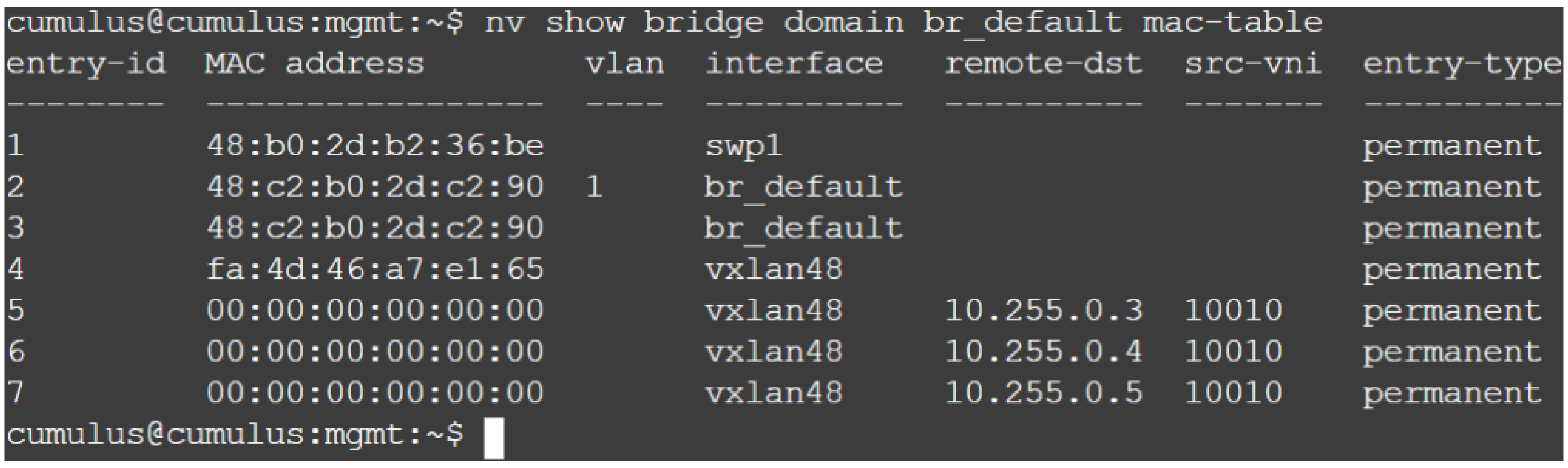


```
*> 10.0.0.2/32      0.0.0.0(superspine02)
                                             0         32768 i
*> 10.1.0.1/32      swp1                     0             0 65001 i
*> 10.1.0.2/32      swp2                     0             0 65001 i
*> 10.1.0.3/32      swp3                     0             0 65002 i
*> 10.1.0.4/32      swp4                     0             0 65002 i
*> 10.255.0.1/32    swp1                                   0 65001 65010
*=                  swp2                                   0 65001 65010
*> 10.255.0.2/32    swp1                                   0 65001 65020
*=                  swp2                                   0 65001 65020
*> 10.255.0.3/32    swp1                                   0 65001 65030
*=                  swp2                                   0 65001 65030
*> 10.255.0.4/32    swp3                                   0 65002 65040
*=                  swp4                                   0 65002 65040
*> 10.255.0.5/32    swp3                                   0 65002 65050
*=                  swp4                                   0 65002 65050
*> 10.255.0.6/32    swp3                                   0 65002 65060
```

We can see all the BGP routes using the show ip bgp command. This command displays all the routes and different paths. There are multiple paths to each network due to the CLOS topology. Initially, it might seem unusual that these different paths have the same AS number, but this is expected since the spines share the same AS number.

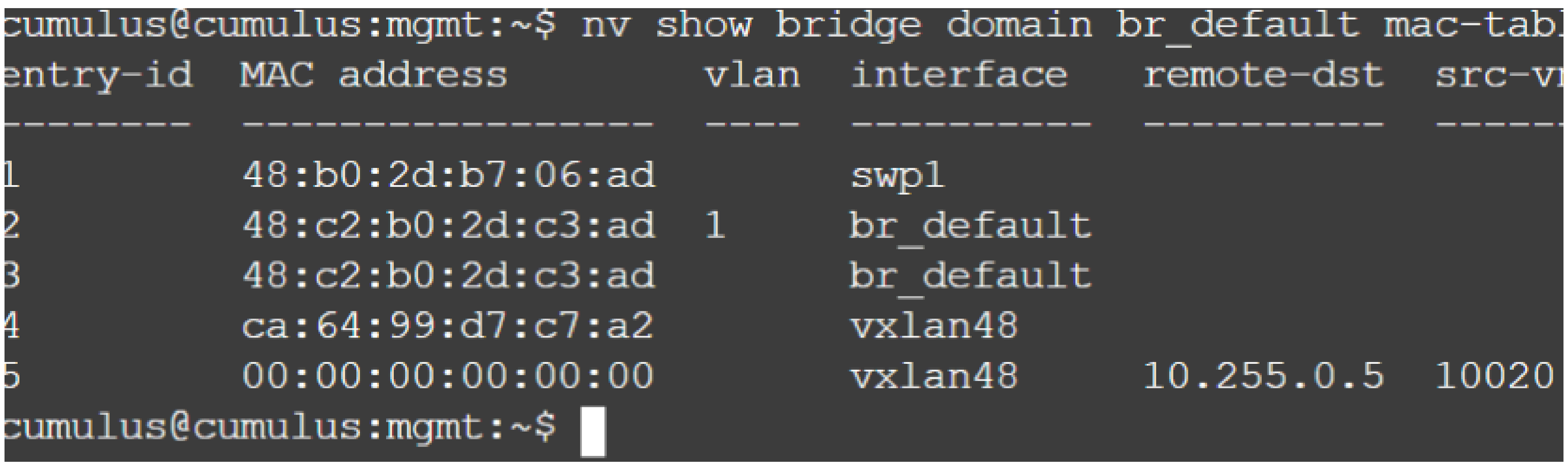
VLAN MAC Tables

Leaf06



```
cumulus@cumulus:mgmt:~$ nv show bridge domain br_default mac-table
entry-id  MAC address        vlan  interface   remote-dst  src-vni  entry-type
--------  -----------------  ----  ----------  ----------  -------  ----------
1         48:b0:2d:b2:36:be        swp1                              permanent
2         48:c2:b0:2d:c2:90  1     br_default                        permanent
3         48:c2:b0:2d:c2:90        br_default                        permanent
4         fa:4d:46:a7:e1:65        vxlan48                           permanent
5         00:00:00:00:00:00        vxlan48     10.255.0.3  10010     permanent
6         00:00:00:00:00:00        vxlan48     10.255.0.4  10010     permanent
7         00:00:00:00:00:00        vxlan48     10.255.0.5  10010     permanent
cumulus@cumulus:mgmt:~$
```

Leaf05

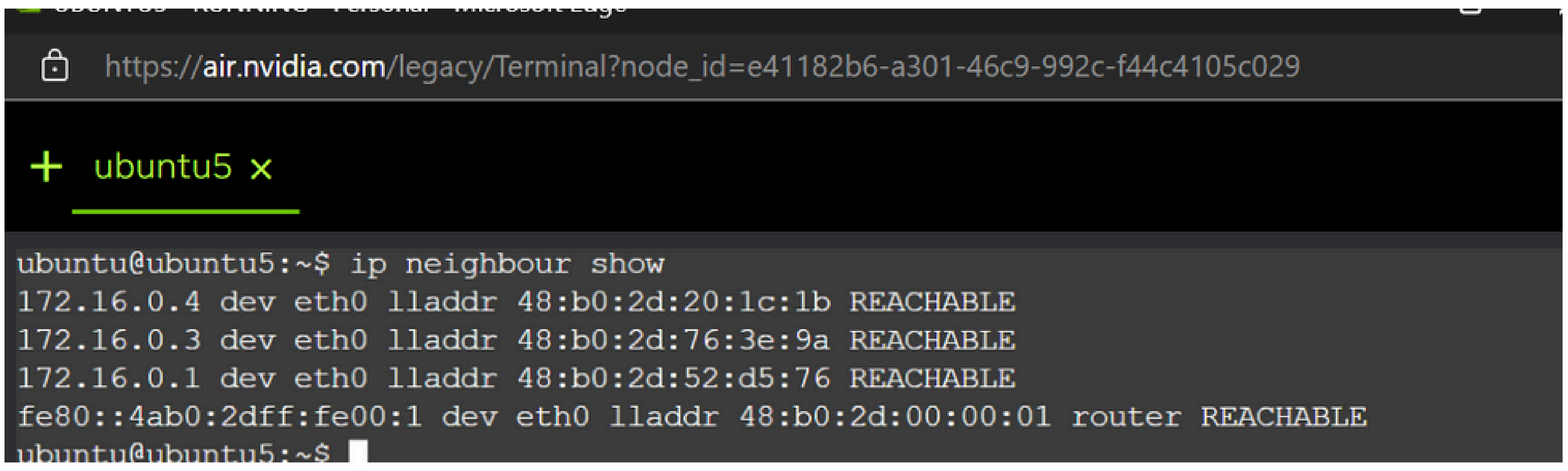


```
cumulus@cumulus:mgmt:~$ nv show bridge domain br_default mac-tabl
entry-id  MAC address        vlan  interface   remote-dst  src-vn
--------  -----------------  ----  ----------  ----------  ------
1         48:b0:2d:b7:06:ad        swp1
2         48:c2:b0:2d:c3:ad  1     br_default
3         48:c2:b0:2d:c3:ad        br_default
4         ca:64:99:d7:c7:a2        vxlan48
5         00:00:00:00:00:00        vxlan48     10.255.0.5  10020
cumulus@cumulus:mgmt:~$
```

Leaf06 is advertising VNI 10010, allowing it to see all other leaf switches advertising the same VNI 10010. Similarly, Leaf05 is advertising VNI 10020, enabling it to see the other leaf switch within the same VNI. However, I am unsure why the MAC addresses are showing up as 00:00:00:00:00.

Server Pinging

Leaf06

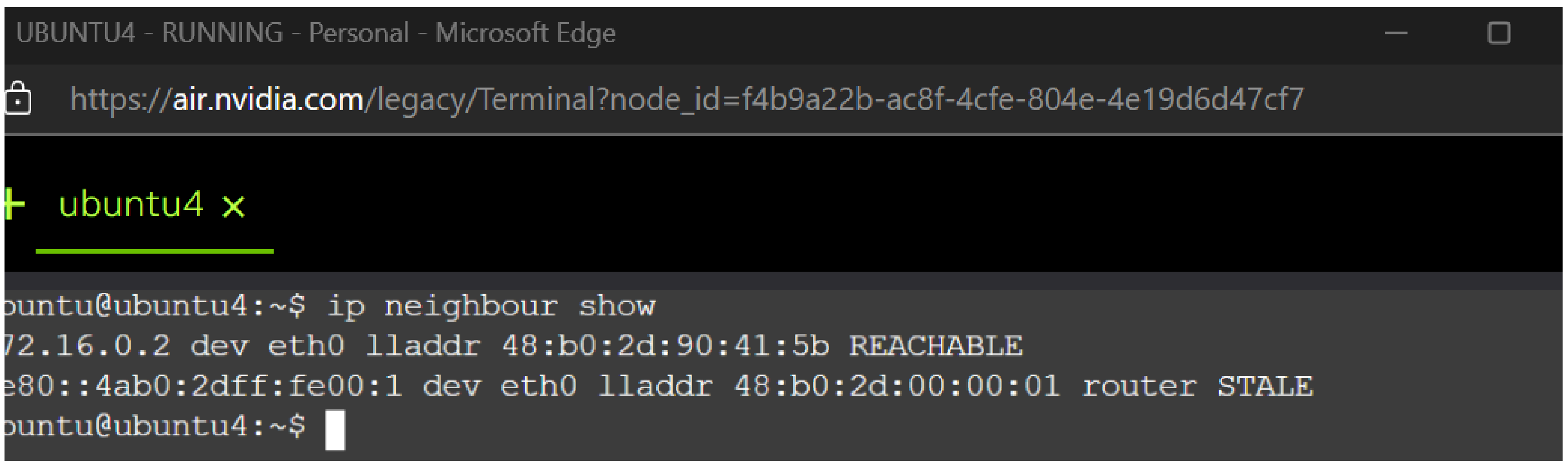


https://air.nvidia.com/legacy/Terminal?node_id=e41182b6-a301-46c9-992c-f44c4105c029

ubuntu5

```
ubuntu@ubuntu5:~$ ip neighbour show
172.16.0.4 dev eth0 lladdr 48:b0:2d:20:1c:1b REACHABLE
172.16.0.3 dev eth0 lladdr 48:b0:2d:76:3e:9a REACHABLE
172.16.0.1 dev eth0 lladdr 48:b0:2d:52:d5:76 REACHABLE
fe80::4ab0:2dff:fe00:1 dev eth0 lladdr 48:b0:2d:00:00:01 router REACHABLE
ubuntu@ubuntu5:~$
```

Leaf05



UBUNTU4 - RUNNING - Personal - Microsoft Edge

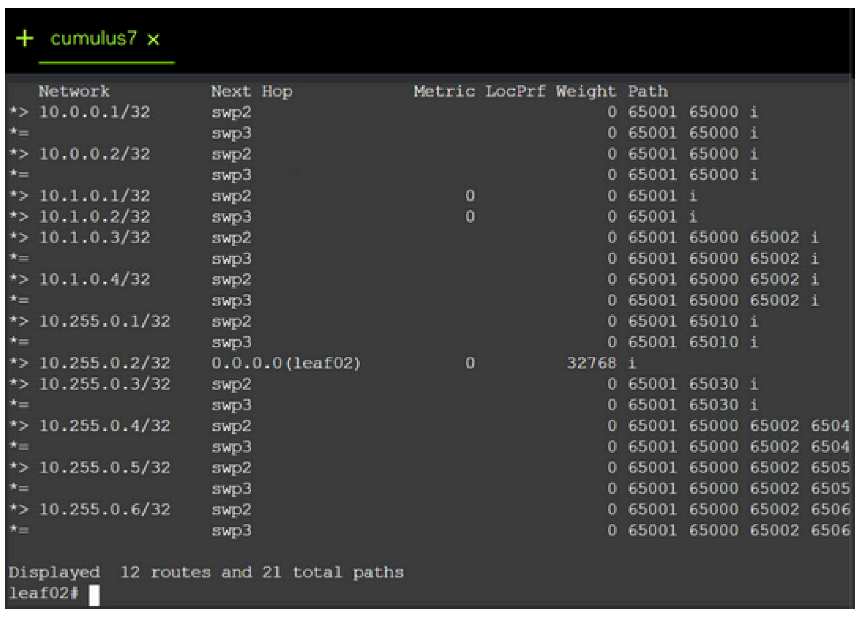
https://air.nvidia.com/legacy/Terminal?node_id=f4b9a22b-ac8f-4cfe-804e-4e19d6d47cf7

ubuntu4

```
buntu@ubuntu4:~$ ip neighbour show
72.16.0.2 dev eth0 lladdr 48:b0:2d:90:41:5b REACHABLE
e80::4ab0:2dff:fe00:1 dev eth0 lladdr 48:b0:2d:00:00:01 router STALE
buntu@ubuntu4:~$
```

I was successful in pinging the other servers for both L2VPN connections. We can see the MAC addresses of each server using the neighbor command I employed.

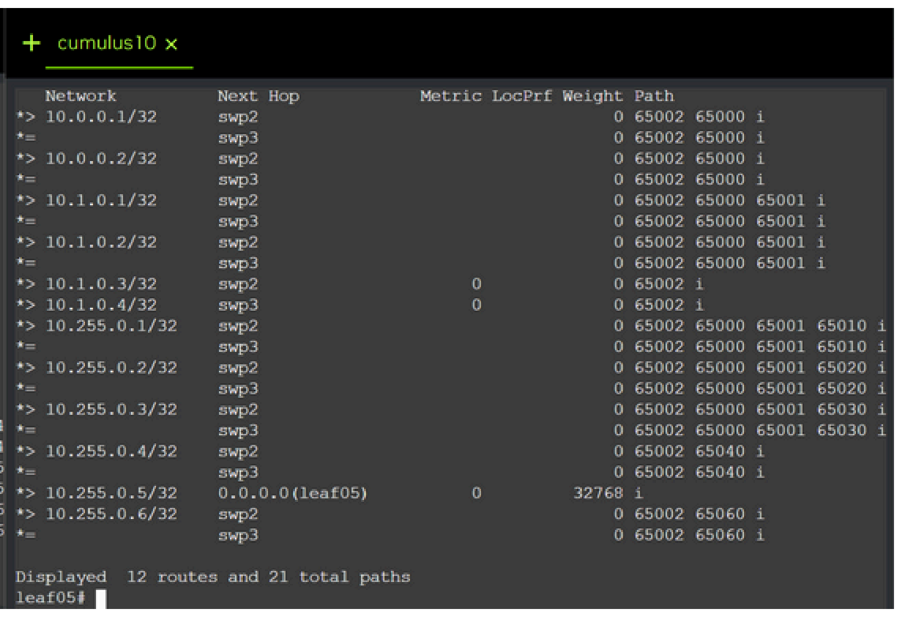
Leaf02 and Leaf05 BGP Routes



cumulus7

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.0.1/32      swp2                                   0 65001 65000 i
*=                  swp3                                   0 65001 65000 i
*> 10.0.0.2/32      swp2                                   0 65001 65000 i
*=                  swp3                                   0 65001 65000 i
*> 10.1.0.1/32      swp2                     0             0 65001 i
*> 10.1.0.2/32      swp3                     0             0 65001 i
*> 10.1.0.3/32      swp2                                   0 65001 65000 65002 i
*=                  swp3                                   0 65001 65000 65002 i
*> 10.1.0.4/32      swp2                                   0 65001 65000 65002 i
*=                  swp3                                   0 65001 65000 65002 i
*> 10.255.0.1/32    swp2                                   0 65001 65010 i
*=                  swp3                                   0 65001 65010 i
*> 10.255.0.2/32    0.0.0.0(leaf02)          0         32768 i
*> 10.255.0.3/32    swp2                                   0 65001 65030 i
*=                  swp3                                   0 65001 65030 i
*> 10.255.0.4/32    swp2                                   0 65001 65000 65002 6504
*=                  swp3                                   0 65001 65000 65002 6504
*> 10.255.0.5/32    swp2                                   0 65001 65000 65002 6505
*=                  swp3                                   0 65001 65000 65002 6505
*> 10.255.0.6/32    swp2                                   0 65001 65000 65002 6506
*=                  swp3                                   0 65001 65000 65002 6506

Displayed  12 routes and 21 total paths
leaf02#
```



cumulus10

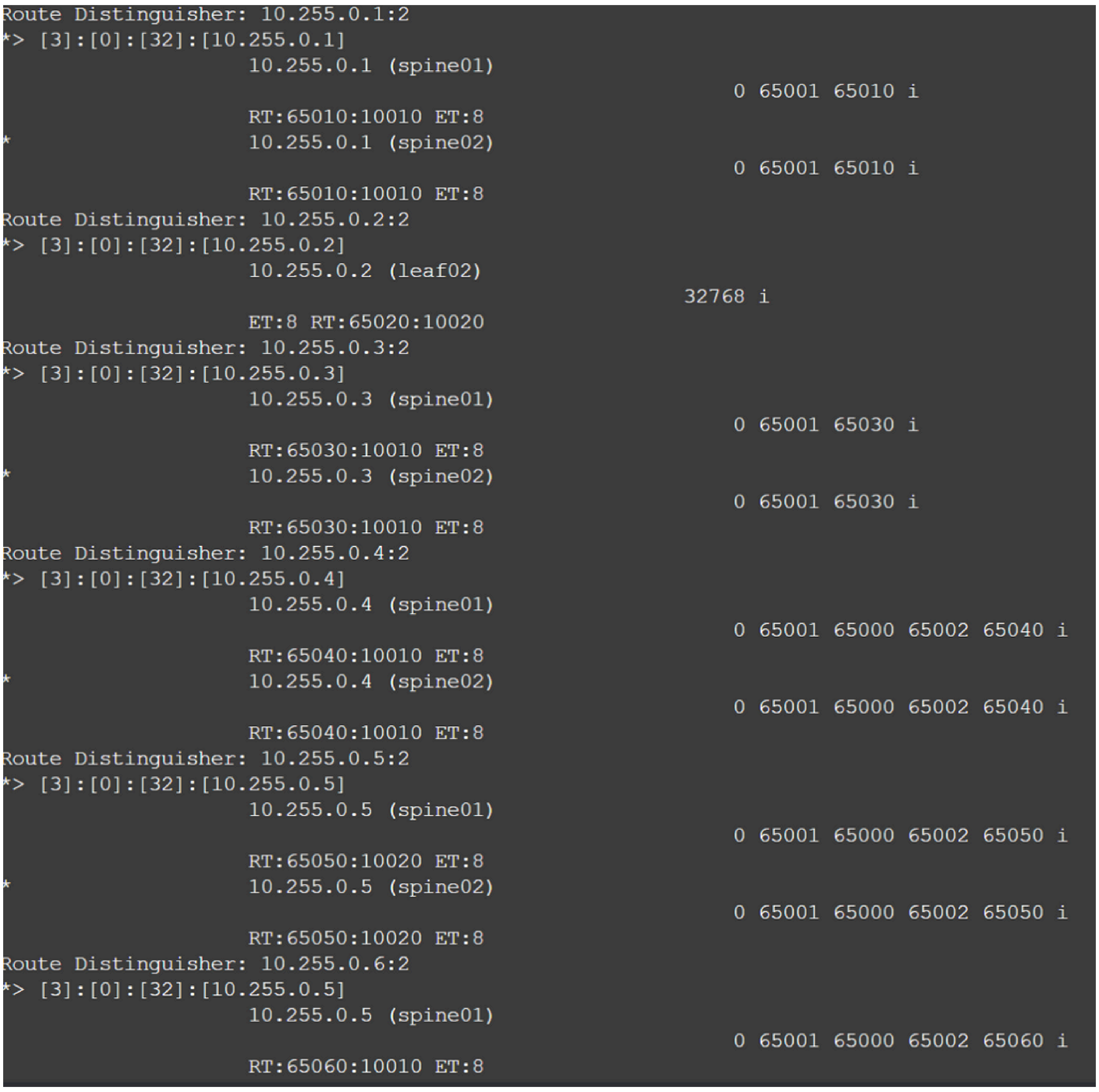
```
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.0.1/32      swp2                                   0 65002 65000 i
*=                  swp3                                   0 65002 65000 i
*> 10.0.0.2/32      swp2                                   0 65002 65000 i
*=                  swp3                                   0 65002 65000 i
*> 10.1.0.1/32      swp2                                   0 65002 65000 65001 i
*=                  swp3                                   0 65002 65000 65001 i
*> 10.1.0.2/32      swp2                                   0 65002 65000 65001 i
*=                  swp3                                   0 65002 65000 65001 i
*> 10.1.0.3/32      swp2                     0             0 65002 i
*> 10.1.0.4/32      swp3                     0             0 65002 i
*> 10.255.0.1/32    swp2                                   0 65002 65000 65001 65010 i
*=                  swp3                                   0 65002 65000 65001 65010 i
*> 10.255.0.2/32    swp2                                   0 65002 65000 65001 65020 i
*=                  swp3                                   0 65002 65000 65001 65020 i
*> 10.255.0.3/32    swp2                                   0 65002 65000 65001 65030 i
*=                  swp3                                   0 65002 65000 65001 65030 i
*> 10.255.0.4/32    swp2                                   0 65002 65040 i
*=                  swp3                                   0 65002 65040 i
*> 10.255.0.5/32    0.0.0.0(leaf05)          0         32768 i
*> 10.255.0.6/32    swp2                                   0 65002 65060 i
*=                  swp3                                   0 65002 65060 i

Displayed  12 routes and 21 total paths
leaf05#
```

This is the show ip bgp route for leaf02 and leaf05. They can see the entire network topology, just like the superspines can. While they have differing paths, the overall topology still looks good.

Leaf02 EVPN Route Targets

show bgp l2vpn evpn



```
Route Distinguisher: 10.255.0.1:2
*> [3]:[0]:[32]:[10.255.0.1]
                    10.255.0.1 (spine01)
                                                                       0 65001 65010 i
                    RT:65010:10010 ET:8
*                   10.255.0.1 (spine02)
                                                                       0 65001 65010 i
                    RT:65010:10010 ET:8
Route Distinguisher: 10.255.0.2:2
*> [3]:[0]:[32]:[10.255.0.2]
                    10.255.0.2 (leaf02)
                                                                   32768 i
                    ET:8 RT:65020:10020
Route Distinguisher: 10.255.0.3:2
*> [3]:[0]:[32]:[10.255.0.3]
                    10.255.0.3 (spine01)
                                                                       0 65001 65030 i
                    RT:65030:10010 ET:8
*                   10.255.0.3 (spine02)
                                                                       0 65001 65030 i
                    RT:65030:10010 ET:8
Route Distinguisher: 10.255.0.4:2
*> [3]:[0]:[32]:[10.255.0.4]
                    10.255.0.4 (spine01)
                                                                       0 65001 65000 65002 65040 i
                    RT:65040:10010 ET:8
*                   10.255.0.4 (spine02)
                                                                       0 65001 65000 65002 65040 i
                    RT:65040:10010 ET:8
Route Distinguisher: 10.255.0.5:2
*> [3]:[0]:[32]:[10.255.0.5]
                    10.255.0.5 (spine01)
                                                                       0 65001 65000 65002 65050 i
                    RT:65050:10020 ET:8
*                   10.255.0.5 (spine02)
                                                                       0 65001 65000 65002 65050 i
                    RT:65050:10020 ET:8
Route Distinguisher: 10.255.0.6:2
*> [3]:[0]:[32]:[10.255.0.5]
                    10.255.0.5 (spine01)
                                                                       0 65001 65000 65002 65060 i
                    RT:65060:10010 ET:8
```

Now, with the show bgp l2vpn evpn command, we can view all the paths and route distinguishers for each VNI. This comprehensive overview of the network topology, including all routes and their attributes, was automatically generated when we configured the network. This command helps us verify the configuration and ensure that everything is functioning correctly within our EVPN VXLAN setup.

Conclusion

This assignment effectively demonstrated the principles and practical implementation of a scalable CLOS datacenter topology using Cumulus Linux. By exploring fundamental concepts such as VXLAN, EVPN, VLANs, and the CLOS architecture, we established a solid foundation for creating a robust and efficient network. Leveraging BGP for routing within the EVPN VXLAN framework was crucial for managing the network effectively.

We successfully automated configurations with Ansible, which, despite some challenges, provided a streamlined approach to managing the network infrastructure. The manual adjustments and troubleshooting steps taken to address specific issues contributed to a deeper understanding of the configuration process.

The practical outcomes, including successful server communication across VNIs and the visibility of MAC addresses in the network overlay, validated the efficiency and scalability of our design. Using the show bgp l2vpn evpn command provided comprehensive insights into the network paths and configurations, confirming that our setup was functioning as intended.

Overall, this project highlighted the importance of careful planning, automation, and troubleshooting in network design. By applying these concepts, we successfully created a resilient and scalable datacenter network that meets the demands of modern network infrastructure.