****

**Next-Generation Cloud Networking: Optimizing Data Center in Google Cloud with VXLAN & EVPN**

**UW-Stout CNIT Spring 2024 Capstone**

White Paper Version: 5.0

Date: 30 April 2024

**Branden Ulm**, Project Manager, ulmb3679@my.uwstout.edu

**Branden Behrens**, Cloud Architect, behrensb1344@my.uwstout.edu

**Cole Melson**, Network Engineer, melsonc0083@my.uwstout.edu

**Lucas Rupp**, Security Engineer, ruppl6285@my.uwstout.edu

**Javier Perez-Sanchez**, Automation Engineer, perezsanchezj2260@my.uwstout.edu

**Michael Long**, Cloud Architect, longm2799@my.uwstout.edu

Industry Sponsor: Arista Networks

**William Goss**, Systems Engineer

# Table of Contents

[Table of Contents 2](#_Toc165314222)

[Table of Figures 4](#_Toc165314223)

[Table of Tables 6](#_Toc165314224)

[Acceptance 7](#_Toc165314225)

[Executive Summary 8](#_Toc165314226)

[Introduction 9](#_Toc165314227)

[Project Goals 9](#_Toc165314228)

[Project Significance 12](#_Toc165314229)

[Technical Challenges 12](#_Toc165314230)

[Project Deliverables 13](#_Toc165314231)

[Team Roles and Contributions 14](#_Toc165314232)

[Problem Description 16](#_Toc165314233)

[Lab Network Requirements and Setup 17](#_Toc165314234)

[Lab Requirements 17](#_Toc165314235)

[Creating a Virtual Machine on GCP 18](#_Toc165314236)

[Enabling WSL and installing a Linux OS 20](#_Toc165314237)

[Creating the Topology with ContainerLabs 21](#_Toc165314238)

[Implementation Steps 27](#_Toc165314239)

[IP Addressing Scheme 27](#_Toc165314240)

[Underlay Network 30](#_Toc165314241)

[MLAG 32](#_Toc165314242)

[Configuration 32](#_Toc165314243)

[VXLAN 34](#_Toc165314244)

[Configuration 35](#_Toc165314245)

[EVPN 36](#_Toc165314246)

[Configuration 37](#_Toc165314247)

[CloudVision 40](#_Toc165314248)

[Onboarding a Device to CloudVision 40](#_Toc165314249)

[Network Provisioning 43](#_Toc165314250)

[Creating and Pushing Configlets 55](#_Toc165314251)

[Challenges and Solutions 61](#_Toc165314252)

[Docker 61](#_Toc165314253)

[ContainerLabs 61](#_Toc165314254)

[Enrolling Devices into CloudVision 62](#_Toc165314255)

[Google Cloud Spot Instance 62](#_Toc165314256)

[Results and Analysis 64](#_Toc165314257)

[Working Underlay Network 64](#_Toc165314258)

[Working Overlay Network 65](#_Toc165314259)

[CloudVision Configlets 68](#_Toc165314260)

[Conclusion 71](#_Toc165314261)

[Acknowledgements 73](#_Toc165314262)

[Document Control with History 74](#_Toc165314263)

[Appendix 76](#_Toc165314264)

[GitHub Page 76](#_Toc165314265)

[ContainerLabs Topology YAML File 76](#_Toc165314266)

[Spine 1 Configuration 82](#_Toc165314267)

[Spine 2 Configuration 84](#_Toc165314268)

[Spine 3 Configuration 86](#_Toc165314269)

[Leaf 1 Configuration 89](#_Toc165314270)

[Leaf 2 Configuration 94](#_Toc165314271)

[Leaf 3 Configuration 100](#_Toc165314272)

[Leaf 4 Configuration 104](#_Toc165314273)

[Borderleaf 1 Configuration 108](#_Toc165314274)

[Borderleaf 2 Configuration 113](#_Toc165314275)

# Table of Figures

[Figure 1: Gannt Chart of Project Timetable 11](#_Toc165352163)

[Figure 2: Arista Capstone Team 14](#_Toc165352164)

[Figure 3: Virtual Hardware and Software 17](#_Toc165352165)

[Figure 4: Creating an Instance on GCP 18](#_Toc165352166)

[Figure 5: Instance Settings in GCP 19](#_Toc165352167)

[Figure 6: Instance E2 Machine Type Settings 19](#_Toc165352168)

[Figure 7: Boot Disc Options in GCP 20](#_Toc165352169)

[Figure 8: Windows Features Settings for WSL 21](#_Toc165352170)

[Figure 9: MGMT Network and Subnet Settings 22](#_Toc165352171)

[Figure 10: Graphite Node Settings 22](#_Toc165352172)

[Figure 11: Sample Spine and Leaf Settings 23](#_Toc165352173)

[Figure 12: Sample Host Settings 23](#_Toc165352174)

[Figure 13: Sample Ethernet Endpoint Settings 24](#_Toc165352175)

[Figure 14: MGMT IP Address 24](#_Toc165352176)

[Figure 15: Default Route & SSH Settings 25](#_Toc165352177)

[Figure 16: CLI Command to Build Lab 25](#_Toc165352178)

[Figure 17: Graphite Image of Lab 25](#_Toc165352179)

[Figure 18: Network Topology 29](#_Toc165352180)

[Figure 19: Sample Leaf eBGP Configuration 31](#_Toc165352181)

[Figure 20: Sample Spine eBGP Configuration 31](#_Toc165352182)

[Figure 21: Underlay BGP Topology 32](#_Toc165352183)

[Figure 22: VLAN 4094 Creation 32](#_Toc165352184)

[Figure 23: MLAG Port-Channel Configuration 33](#_Toc165352185)

[Figure 24: Interface VLAN 4094 Configuration 33](#_Toc165352186)

[Figure 25: MLAG Configuration 33](#_Toc165352187)

[Figure 26: Show MLAG Command Output 34](#_Toc165352188)

[Figure 27: Interface VXLAN Configuration 35](#_Toc165352189)

[Figure 28: Interface VXLAN Configuration 35](#_Toc165352190)

[Figure 29: VXLAN VRF Configuration 36](#_Toc165352191)

[Figure 30: Loopback Address used in EVPN 37](#_Toc165352192)

[Figure 31: Loopbacks used in MPLS EVPN 38](#_Toc165352193)

[Figure 32: Spine Underlay Configuration 38](#_Toc165352194)

[Figure 33: Leaf BGP EVPN Configuration 39](#_Toc165352195)

[Figure 34: Leaf BGP EVPN Address Family 39](#_Toc165352196)

[Figure 35: Leaf BGP EVPN Overlay 40](#_Toc165352197)

[Figure 36: Ping to CloudVision 41](#_Toc165352198)

[Figure 37: Installation of TerminAttr 41](#_Toc165352199)

[Figure 38: Show Clock Command 42](#_Toc165352200)

[Figure 39: Token Generation in CloudVision Portal 42](#_Toc165352201)

[Figure 40: Pasting the Token on a Device 42](#_Toc165352202)

[Figure 41: Starting TerminAttr to Complete Onboarding 43](#_Toc165352203)

[Figure 42: Network Provisioning Tab 43](#_Toc165352204)

[Figure 43: Adding Datacenter Container 44](#_Toc165352205)

[Figure 44: POD1 under UWStoutCapstone Datacenter Container 44](#_Toc165352206)

[Figure 45: Tree View of Containers 45](#_Toc165352207)

[Figure 46: Adding Device to POD Container 46](#_Toc165352208)

[Figure 47: List of devices under Undefined Devices 46](#_Toc165352209)

[Figure 48: Devices Highlighted Green Pending Changes 47](#_Toc165352210)

[Figure 49: Manage Configlet under Device Management 47](#_Toc165352211)

[Figure 50: Validating Device Configurations 48](#_Toc165352212)

[Figure 51: Comparison between Running and Designed Configuration 48](#_Toc165352213)

[Figure 52: Device Configurations after Reconciliation 49](#_Toc165352214)

[Figure 53: Tree View of all Respective Devices Reconciled and Validated 50](#_Toc165352215)

[Figure 54: Pending Tasks on the Devices 50](#_Toc165352216)

[Figure 55: List of Pending Tasks 51](#_Toc165352217)

[Figure 56: Selected List of Pending Tasks 51](#_Toc165352218)

[Figure 57: Create Change Control of Pending Tasks 52](#_Toc165352219)

[Figure 58: Change Control Window 53](#_Toc165352220)

[Figure 59: Review Change Panel 53](#_Toc165352221)

[Figure 60: Completion of Change Control 54](#_Toc165352222)

[Figure 61: Tree View of Devices in Network Provisioning Tab 54](#_Toc165352223)

[Figure 62: Configlets Management Window 55](#_Toc165352224)

[Figure 63: Add and Create Configlets 56](#_Toc165352225)

[Figure 64: Configlet Builder 57](#_Toc165352226)

[Figure 65: Python Script 57](#_Toc165352227)

[Figure 66: Built Configlet using Python 58](#_Toc165352228)

[Figure 67: Managing Configlets Through Container 59](#_Toc165352229)

[Figure 68: Applied Configlet to Container 59](#_Toc165352230)

[Figure 69: Validate Configlet to Specified Device 60](#_Toc165352231)

[Figure 70: Ping in Underlay Network 64](#_Toc165352232)

[Figure 71: Traceroute in Underlay Network 64](#_Toc165352233)

[Figure 72: Underlay Network Diagram 65](#_Toc165352234)

[Figure 73: Ping in Overlay Network 65](#_Toc165352235)

[Figure 74: Traceroute in Overlay Network 65](#_Toc165352236)

[Figure 75: Overlay Network Diagram 66](#_Toc165352237)

[Figure 76: Show BGP Summary Command 68](#_Toc165352238)

[Figure 77: Show BGP EVPN Command 68](#_Toc165352239)

[Figure 78: Loopback Configlet 69](#_Toc165352240)

[Figure 79: Sample Output of Loopback Configlet 70](#_Toc165352241)

# Table of Tables

[Table 1: Project Timetable 10](#_Toc163408329)

[Table 2: Loopback IP Addressing 26](#_Toc163408330)

[Table 3: Spine-to-Leaf IP Addressing 27](#_Toc163408331)

[Table 4: Leaf-to-Leaf IP Addressing 29](#_Toc163408332)

# Acceptance

The signatories below confirm that the design and implementation meet the requirements specified in the Statement of Work.

**Arista Networks**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Signature | Date |
| William Goss | Capstone Sponsor | *William Goss* | 4/30/24 |

**Arista Spring 2024 Capstone Group**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Signature | Date |
| Branden Ulm | Project Lead | *Branden Ulm* | 4/30/24 |
| Branden Behrens | Cloud Architect | *Branden Behrens* | 4/30/24 |
| Cole Melson | Network Engineer | *Cole Melson* | 4/30/2024 |
| Lucas Rupp | Security Engineer | *Lucas Rupp* | 4/30/2024 |
| Javier Perez-Sanchez | Automation Engineer | *Javier Perez-Sanchez* | 4/30/2024 |
| Michael Long | Cloud Architect | *Michael Long* | 4/30/2024 |

**Advisor**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Signature | Date |
| Holly Yuan | Capstone Advisor | [Signature] | [Date] |

# 

# Executive Summary

Arista Networks, a leader in data-driven networking, emphasizes scalability, high-performance, and ultra-low latency in large data center, campus, and routing environments. The capstone teams aim to leverage Arista Networks technology to build a Leaf/Spine Data Center environment within Google Cloud. Arista's emphasis on scalability and high-performance networking serves as the foundation for the project's goals, which include implementing industry-standard, nonproprietary protocols like VXLAN and EVPN. Completion of the project will offer insights into the future of networking, showcasing next-generation protocols, scalable infrastructure design, and automation for efficiency and reliability. The project's outcomes will serve as a blueprint for businesses seeking to optimize networking capabilities while also serving as a testbed for potential upcoming curriculum at the University of Wisconsin-Stout based on the contents of the project. Through hands-on experience with Arista hardware and new technologies, the team gains valuable skills and readiness for real-world networking challenges, highlighting the project's impact on both their academic and professional journey.

# Introduction

Since their formation in 2008, Arista Networks has quickly established themselves as an industry leader in creating data-driven, client to cloud networking. With a focus in large data center, campus, and routing environments, the keys to Arista’s robust and award-winning platform include scalability, high-performance, and ultra-low latency. Strategic partnerships with top industry leaders and a globally recognized leadership team have given Arista valuable insights in how to develop the most agile, available, and reliable networks and have since brought those principles to the larger industry. The advocation of a new type of network infrastructure, the use of next generation open standard protocols, automation for streamlined deployment and management, and enhanced visibility into performance metrics have all contributed to Arista’s success. In turn, the capstone team is tasked with leveraging these concepts throughout the project to better grasp these concepts while gaining valuable knowledge in the future of networking.

## Project Goals

The main goal the capstone team aims to solve is being able to use Arista technology to create a Leaf/Spine Data Center environment hosted in Google Cloud. From there, the team established specific, measurable, achievable, and realistic (SMART) goals to ensure the overall success of the project. Firstly, an emphasis was placed on creating an achievable project plan to ensure timeliness in completing all of the project deliverables. To do so, the statement of work was divided into smaller, incremented steps and divided equally among the team members. A project timetable and subsequent Gannt chart, as shown in Table 1 and Figure 1 below, were created to showcase these milestones.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Finish** |
| **Arista\_Capstone** | **64 days** | **Tue 1/30/24** | **Fri 4/26/24** |
| **Training/SOW overview** | **14 days** | **Tue 1/30/24** | **Fri 2/16/24** |
| CEOS Training | 14 days | Tue 1/30/24 | Fri 2/16/24 |
| Meeting | 1 day | Tue 1/30/24 | Tue 1/30/24 |
| **Installation** | **12 days** | **Mon 2/19/24** | **Tue 3/5/24** |
| WSL | 1 day | Mon 2/19/24 | Mon 2/19/24 |
| Container lab | 12 days | Mon 2/19/24 | Tue 3/5/24 |
| Docker | 12 days | Mon 2/19/24 | Tue 3/5/24 |
| **High Level Design Documentation & Initial setup review** | **5 days** | **Wed 3/6/24** | **Tue 3/12/24** |
| Topology design | 5 days | Wed 3/6/24 | Tue 3/12/24 |
| IP Addressing | 5 days | Wed 3/6/24 | Tue 3/12/24 |
| Service Allocation | 5 days | Wed 3/6/24 | Tue 3/12/24 |
| ASN placement | 5 days | Wed 3/6/24 | Tue 3/12/24 |
| **System Configuration** | **19 days** | **Wed 3/13/24** | **Mon 4/8/24** |
| Automation | 14 days | Wed 3/13/24 | Mon 4/1/24 |
| BGP | 5 days | Wed 3/13/24 | Tue 3/19/24 |
| MLAG | 5 days | Wed 3/13/24 | Tue 3/19/24 |
| Firewall | 14 days | Wed 3/13/24 | Mon 4/1/24 |
| VXLAN | 5 days | Wed 3/13/24 | Tue 3/19/24 |
| EVPN | 14 days | Wed 3/20/24 | Mon 4/8/24 |
| **Test and Approval** | **10 days** | **Tue 4/9/24** | **Mon 4/22/24** |
| Security | 10 days | Tue 4/9/24 | Mon 4/22/24 |
| Final Meeting | 1 day | Tue 4/9/24 | Tue 4/9/24 |
| **Presentation Week** | **3 days** | **Tue 4/23/24** | **Thu 4/25/24** |
| Draft of White Paper | 3 days | Tue 4/23/24 | Thu 4/25/24 |
| Presentation | 1 day | Tue 4/23/24 | Tue 4/23/24 |
| **Final White Paper** | **1 day** | **Fri 4/26/24** | **Fri 4/26/24** |
| Turn in White Paper | 1 day | Fri 4/26/24 | Fri 4/26/24 |
| Close Out Project | 1 day | Fri 4/26/24 | Fri 4/26/24 |

**Table 1: Project Timetable**

A screenshot of a computer

Description automatically generated

Figure 1: Gantt Chart of Project Timetable

Secondly, the team set out to gain a greater understanding of open standard protocols requested by Arista to be used in the project, such as VXLAN, EVPN, and MLAG. Through Arista-provided training videos and lab environment, along with outside research, the team was able to gain a greater understanding of these protocols which will have benefits for both the success of the project and for future careers in the IT industry. Finally, the team aims to gain valuable industry experience in implementing the next-generation infrastructure as tasked by Arista. By having this knowledge, the team will be able to stay ahead of the everchanging IT industry.

## Project Significance

As a result of completing this project, the capstone team will be able to showcase valuable insights about the future landscape of networking. The use next-generation networking protocols, a scalable infrastructure design, and incorporating automation provides a roadmap for the future of the IT industry to transition their own environments into the future. Arista’s leading-edge robust design that leverages cloud networking and automation concepts will provide a solid foundation for businesses seeking to optimize their networking capabilities for efficiency, reliability, and adaptability in an ever-evolving digital landscape. Additionally, this project will be used by Arista to test whether upper-level students can grasp the concepts of VXLAN and EVPN for a potential new course at UW-Stout, to be co-developed by the Arista team. This capstone project serves not only as a testament to the team's dedication and expertise but also to guide others towards a more resilient and agile future in networking technology along with providing future students the opportunity to explore these same concepts.

## Technical Challenges

Throughout the duration of project, the capstone team anticipates numerous challenges that will need to be addressed. Mainly, the team is faced with learning about and understanding new technologies that play a vital role in the project’s success. Specifically, protocols such as VXLAN, MLAG, and EVPN are brand new concepts to the team and expect them to be create challenges during the implementation phases of the project. Additionally, the use of automation and scripting poses an equally large technical hurdle. With limited knowledge and practical experience, the team plans to use training and hands-on practice to effectively implement and troubleshoot these technologies. Overcoming these challenges will not only require technical expertise but also effective communication, collaboration, and problem-solving skills within the team. Despite the anticipated difficulties, the capstone team remains committed to overcoming obstacles and delivering a successful project outcome while obtaining an expert understanding of any new concepts.

## Project Deliverables

The main deliverables of this capstone project are as follows:

* Configuration of a Leaf/Spine Data Center environment hosted in Google Cloud using Border Gateway Protocol (BGP) for the underlay control plane while using different AS groups to differentiate the leaf and spine devices.
* Configure VXLAN and EVPN on top of the existing network to extend layer 2 across the network.
* Configure MLAG on each leaf pair with channel groups created on the interfaces that connect to the appropriate MLAG peer switch.
* Use CloudVision to sync all devices to provide real time updates on the devices and to review the history of the configurations.
* Use configlets, which are portions of configurations that are managed and deployed in Arista’s CloudVision, to configure the border leaf devices.
* Demonstrate an expert level of understanding of technical concepts such as VXLAN and EVPN in the demonstration to the Advisory Committee.

## Team Roles and Contributions

To tackle the various deliverables of the project, the team, as pictured in Figure 2 below, divided work according to the roles assigned to better ensure the success of the project while meeting all deadlines created by the team.

A group of men standing in front of a black machine

Description automatically generated

Figure 2: Arista Capstone Team

The roles of each team member and contributions to the project are as follows:

* **Branden Ulm, Project Manager**: The Project Manager is tasked with overseeing the project through all phases, from planning to implementation. Additionally, they are responsible for coordinating tasks, managing resources, and ensuring that the project stays on schedule.
* **Branden Behrens, Cloud Architect**: The Cloud Architect is focused on the design and overseeing of the architecture of the team’s cloud-based solution. Their role involves selecting appropriate cloud services, defining the necessary networking configurations, and ensuring that the team’s architecture meets all the necessary requirements as outlined in the statement of work.
* **Cole Melson, Network Engineer**: The Network Engineer is responsible for managing and maintaining the networking environment while handling tasks such as ensuring the uptime of the Google Cloud-hosted VM, BGP and EVPN states, along with other networking concepts required in the project.
* **Lucas Rupp, Security Engineer**: The role of the Security Engineer is to ensure the security and compliance of the cloud environment, along with overseeing the successful integration of EVPN. They will additionally be collaborating with other team members to integrate security best practices into the architecture and address any vulnerabilities or compliance requirements.
* **Javier Perez-Sanchez, Automation Engineer**: The Automation Engineer focuses on the automation of repetitive tasks through the use of configlets and streamline processes within the cloud environment.
* **Michael Long, Cloud Architect**: The use of a second Cloud Architect is to provide additional expertise and support. This individual collaborates with the primary Cloud Architect to refine the architecture, address technical challenges, and ensure that the cloud solution meets performance, scalability, and reliability requirements. Together, they work to design an optimal cloud infrastructure that aligns with the project's goals and objectives.

# Problem Description

In this project, Arista aims to give the team hands-on experience and expert guidance in designing and implementing a next-generation network infrastructure. The primary focus is on addressing critical challenges prevalent in current network infrastructures, such as scalability, flexibility, and maintenance efficiency. By leveraging automation, virtualization, and telemetry, the team is tasked with constructing a next-generation network framework within the Google Cloud platform.

The problem this capstone project addresses is the deployment of a virtual L2 leaf and spine data center architecture within the Google Cloud platform while utilizing technologies such as VXLAN and EVPN, along with complying with Arista and industry best practices. Specifically, the project aims to enhance scalability, streamline operations, and improve overall network efficiency in modern digital ecosystems. To justify the solution, it is essential to understand the significant benefits it offers. Automation and virtualization promise increased operational efficiency, while telemetry provides insights for proactive maintenance and optimization. By integrating with Arista's CloudVision platform, the team gains centralized management capabilities for real-time monitoring, configuration management, and troubleshooting, ensuring seamless coordination and control across the network infrastructure. Furthermore, the significance of this project lies in its application of industry-leading practices and cutting-edge technologies, distinguishing it from traditional approaches. Through the practical implementation of Arista’s best practices while harnessing the robust features of Google Cloud, the capstone team aims to showcase the new standards in network design and management for the future.

# Lab Network Requirements and Setup

The network for this capstone project is being built using a Linux virtual machine, hosted in Google Cloud, using ContainerLabs running in Docker, as shown Figure 3 below. To test configurations on our local machines before we deploy to the main topology, the team is using Windows WSL to host an Ubuntu Linux machine with Docker and Containerlabs. ContainerLabs uses graphite to present a visual topology to the user as well as links to SSH into the devices. The ContainerLabs topology running inside docker takes quite a bit of RAM and CPU, so keep in mind that if the environment is running slow adding more RAM and/or CPU may solve this issue.

A screenshot of a computer

Description automatically generated

Figure 3: Virtual Hardware and Software

## Lab Requirements

To get ContainerLabs running, there are some prerequisites that must be met. This includes A Linux virtual machine (VM) or Windows Subsystem for Linux (WSL), Docker, a cEOS image, and ContainerLabs installed.

### Creating a Virtual Machine on GCP

The capstone team created a virtual machine hosted on GCP to remotely host the lab environment. ContainerLabs can require quite a bit of computing power to run smoothly, but there were some cost requirements that helped guide the team in the right direction of what computing spec to choose. Below shows how to create a machine that can run the lab environment.

1. First, navigate to Google Cloud Platform computing engine tab and click the create instance button.

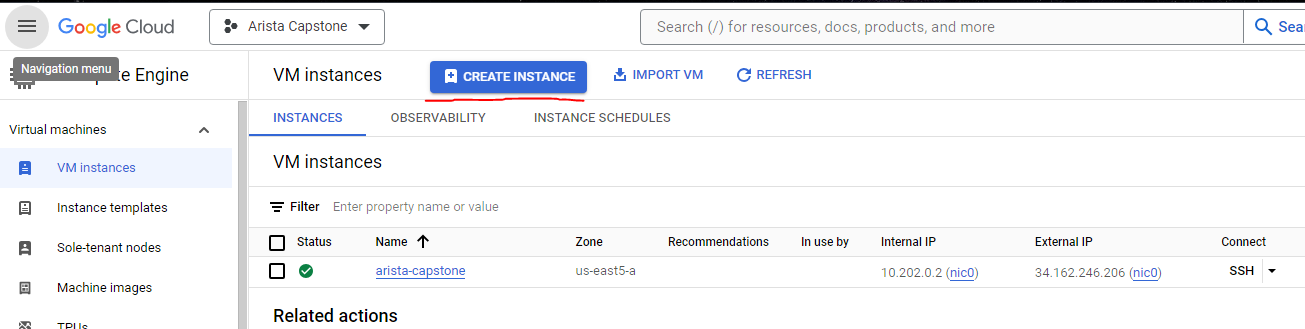


Figure 4: Creating an Instance on GCP

1. Give your instance a fitting name and ensure that E2 is the selected instance.

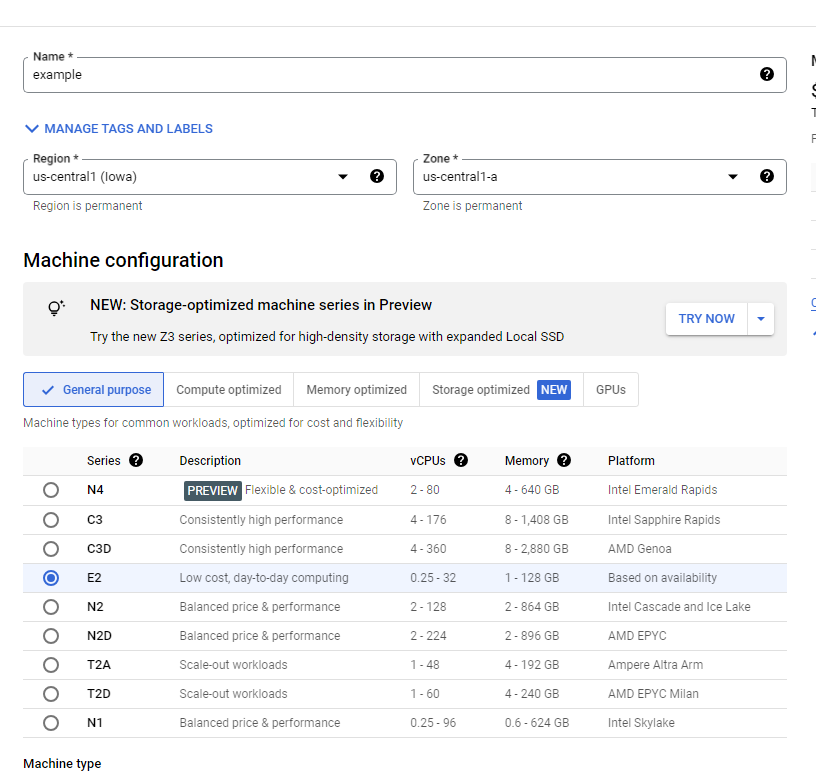


Figure 5: Instance Settings in GCP

1. Scroll down and select standard e2 medium option in machine type.

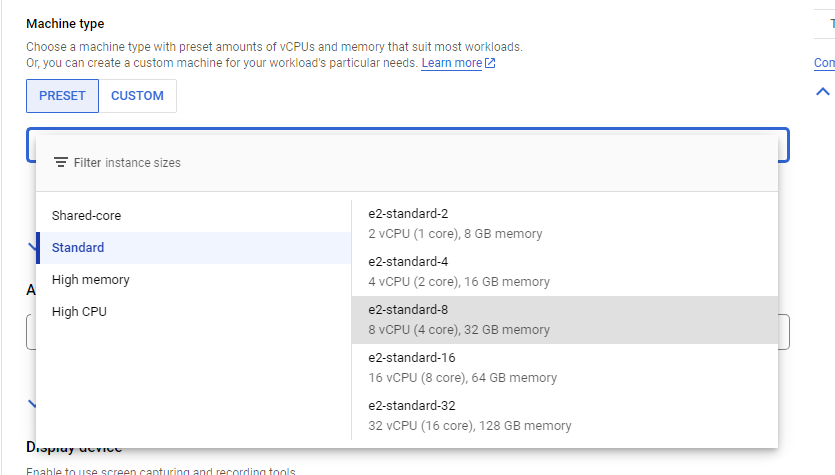


Figure 6: Instance E2 Machine Type Settings

1. Scroll down further and open the option menu for boot disk options and increase the size allotted to 50gb.

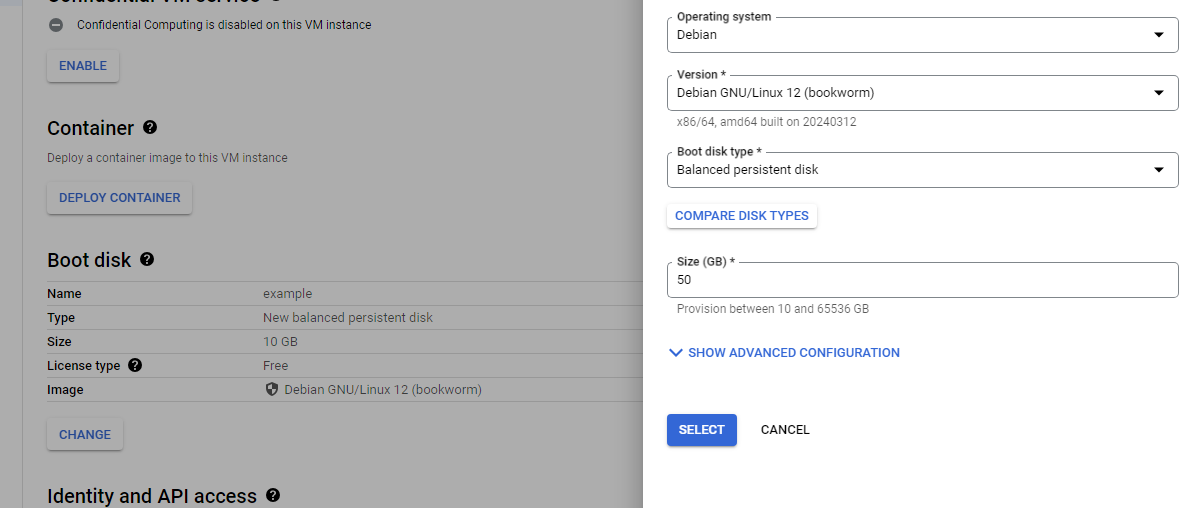


Figure 7: Boot Disc Options in GCP

1. At the Firewall section, check both the http and https options in order to allow ContainerLabs to connect to CloudVision and to allow remote configuration changes from ContainerLabs itself.
2. Lastly, click create to finalize the virtual machine configurations and it will start to boot the instance.

### Enabling WSL and installing a Linux OS

1. Open “Windows Features”
2. Scroll down to “Windows Subsystem for Windows”, check the box and click OK.

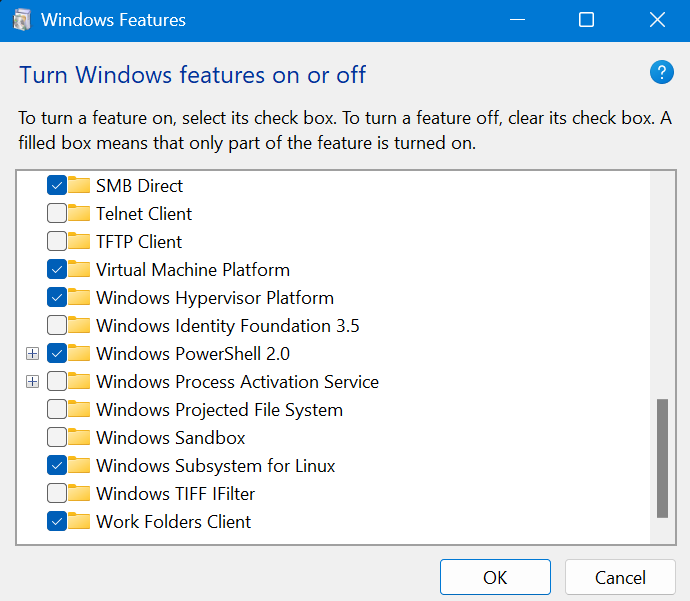


Figure 8: Windows Features Settings for WSL

1. Restart your computer.
2. Once your computer reboots, go to the Microsoft store and install the desired Linux Operating System (OS).
3. After you download the OS, in the Windows search bar, search the Linux OS previously installed, then open it.
4. Input a username and password.
5. A regular Linux terminal will now display which can be used to navigate through the Windows computer.

## Creating the Topology with ContainerLabs

ContainerLab environments are built using a YAML file which defines the variable needed to build out a virtualized lab network. This YAML file provides details including each network node, its OS version/type, unique interfaces per host, logical connectivity between nodes, and a management network for the environment.

1. The first part of creating a topology is naming the yaml file and giving it a subnet for the management addresses. Additionally, the type of interfaces that the devices will use must be set.

A black rectangular object with a black background

Description automatically generated

Figure 9: MGMT Network and Subnet Settings

1. The desired version of cEOS that will be running on the machines must also be set. To do this, the desired version of Arista cEOS must be downloaded and in a directory so ContainerLabs can use it.
2. Next is the Graphite node, which will configure how to connect to the visual of the topology. In this project, the team used the regular node that has a unique management address and defined what OS will be used, which in this case is Linux.

A screenshot of a computer program

Description automatically generated

Figure 10: Graphite Node Settings

1. To create a topology to meet the requirements, add sections that call for each machine's creation. For example, Figures 10 and 11 show a sample spine, leaf and host configuration found in the yaml file. When adding devices, make sure each has a unique management address, or the file will not work.

A screen shot of a computer

Description automatically generated

Figure 11: Sample Spine and Leaf Settings

A computer screen with white text

Description automatically generated

Figure 12: Sample Host Settings

1. Another important step is the connection so that the topology is connected with ethernet simulations. To do this, create an endpoints section that lists all the connections on each device. Something the team would recommend is to start at the top left and move right and down direction to ensure devices do not end up with repeat connections.

A screenshot of a computer program

Description automatically generated

Figure 13: Sample Ethernet Endpoint Settings

1. After the yaml file is written to meet the topology needed, it is important to check that the .cfg files have the corresponding management IP address.



Figure 14: MGMT IP Address

1. Ensure a default route is set to the default gateway of the management network. No routing command on the VRF and setting SSH transport mode will enable SSH into the devices over the management network.

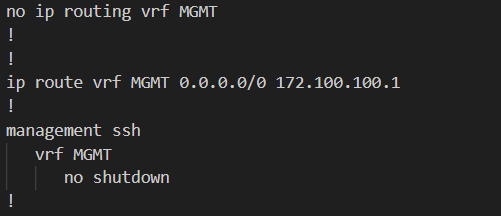


Figure 15: Default Route & SSH Settings

1. Once all the above steps are completed, the environment can be built using ContainerLabs in Docker. This is done with a command inside of the terminal of the Linux machine. The file path may vary and the yaml file name can be named as desired. If the cEOS image has not been downloaded or Docker has not been installed, ContainerLabs, and cloned the necessary repositories this will not work.



Figure 16: CLI Command to Build Lab

1. After the topology has finished being built, navigate to the URL that is provided in the terminal's output or navigate to 127.0.0.1/graphite. An image will appear, such as in Figure 16. Click on a machine and select SSH to access the CLI.

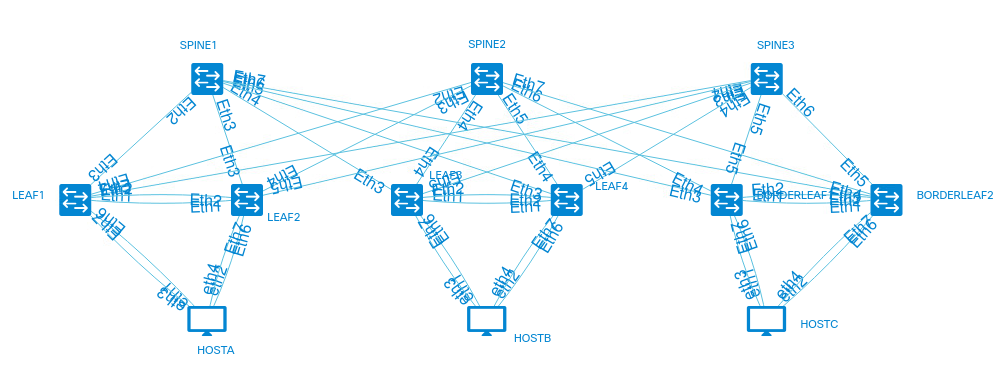


Figure 17: Graphite Image of Lab

1. For more specifics regarding the yaml and .cfg files used in this project, along with the device configurations used, please go to either the GitHub page linked in the appendix at the end of this whitepaper.

# Implementation Steps

The following steps were used to configure the capstone lab environment in accordance with the Statement of Work provided by Arista.

## IP Addressing Scheme

Following Arista’s best practices when it comes to IP addressing, the networks used are in the 172.16/12 range. Starting off with the loopback interfaces, all of the switches have Loopback0 configured on them with the Borderleaf switches having Loopback1 configured on them as well.

|  |  |  |
| --- | --- | --- |
| **Device** | **Interface** | **Address** |
| **Spine 1** | Lo0 | 172.16.0.101/32 |
| **Spine 2** | Lo0 | 172.16.0.102/32 |
| **Spine 3** | Lo0 | 172.16.0.103/32 |
| **Leaf 1** | Lo0 | 172.16.0.70/32 |
| **Leaf 2** | Lo0 | 172.16.0.71/32 |
| **Leaf 3** | Lo0 | 172.16.0.72/32 |
| **Leaf 4** | Lo0 | 172.16.0.73/32 |
| **Borderleaf 1** | Lo0 | 172.16.0.201/32 |
|  | Lo1 | 172.16.1.201/32 |
| **Borderleaf 2** | Lo0 | 172.16.0.202/32 |
|  | Lo1 | 172.16.1.201/32 |

Table 2: Loopback IP Addressing

To conserve IP addresses, the point-to-point links between the switches were all configured with /32 subnet masks, starting from 172.16.254.0/31 in eth2 on Spine 1 to 172.16.254.35/31 in eth5 on Borderleaf 2, as shown in Table 3. Furthermore, the connections between each device are graphically displayed in Figure 18.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Device** | **Interface** | **Address** | **Device 2** | **Address 2** |
| **Spine 1** | eth2 | 172.16.254.0/31 | leaf1-eth3 | 172.16.254.1/31 |
|  | eth3 | 172.16.254.2/31 | leaf2-eth2 | 172.16.254.3/31 |
|  | eth4 | 172.16.254.4/31 | leaf3-eth3 | 172.16.254.5/31 |
|  | eth5 | 172.16.254.6/31 | leaf4-eth3 | 172.16.254.7/31 |
|  | eth6 | 172.16.254.8/31 | BorderLeaf1-eth3 | 172.16.254.9/31 |
|  | eht7 | 172.16.254.10/31 | BorderLeaf2-eth3 | 172.16.254.11/31 |
| **Spine 2** | eth2 | 172.16.254.12/31 | leaf1-eth4 | 172.16.254.13/31 |
|  | eth3 | 172.16.254.14/31 | leaf2-eth4 | 172.16.254.15/31 |
|  | eth4 | 172.16.254.16/31 | leaf3-eth4 | 172.16.254.17/31 |
|  | eth5 | 172.16.254.18/31 | leaf4-eth4 | 172.16.254.19/31 |
|  | eth6 | 172.16.254.20/31 | BorderLeaf1-eth4 | 172.16.254.21/31 |
|  | eth7 | 172.16.254.22/31 | BorderLeaf2-eth4 | 172.16.254.23/31 |
| **Spine 3** | eth1 | 172.16.254.24/31 | leaf1-eth5 | 172.16.254.25/31 |
|  | eth2 | 172.16.254.26/31 | leaf2-eth5 | 172.16.254.27/31 |
|  | eth3 | 172.16.254.28/31 | leaf3-eth5 | 172.16.254.29/31 |
|  | eth4 | 172.16.254.30/31 | leaf4-eth5 | 172.16.254.31/31 |
|  | eth5 | 172.16.254.32/31 | Borderleaf1-eth5 | 172.16.254.33/31 |
|  | eth6 | 172.16.254.34/31 | Borderleaf2-eth5 | 172.16.254.35/31 |

Table 3: Spine-to-Leaf IP Addressing

A diagram of a network

Description automatically generated

Figure 18: Network Topology

To support the port channels between leaf pairs and virtual network identifiers (VNI) used for the VXLAN overlay network, the following leaf-to-leaf physical and virtual addressing is used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Device** | **Interface** | **Address** | **Interface/VNI** | **Address** |
| **Leaf1** | eth3 | 172.16.254.1/31 | 1010 | 172.16.101.1/24 |
|  | eth4 | 172.16.254.13/31 | 1030 | 172.16.103.1/24 |
|  | eth5 | 172.16.254.25/31 | Vlan 4094 | 172.16.255.1/30 |
| **Leaf2** | eth3 | 172.16.254.3/31 | 1010 | 172.16.101.2/24 |
|  | eth4 | 172.16.254.15/31 | 1030 | 172.16.103.2/24 |
|  | eth5 | 172.16.254.27/31 | Vlan 4094 | 172.16.255.2/30 |
| **Leaf3** | eth3 | 172.16.254.5/31 | 1020 | 172.16.102.1/24 |
|  | eth4 | 172.16.254.17/31 | 1030 | 172.16.103.3/24 |
|  | eth5 | 172.16.254.29/31 | Vlan 4094 | 172.16.255.5/30 |
| **Leaf4** | eth3 | 172.16.254.7/31 | 1020 | 172.16.102.2/24 |
|  | eth4 | 172.16.254.19/31 | 1030 | 172.16.103.4/24 |
|  | eth5 | 172.16.254.31/31 | Vlan 4094 | 172.16.255.6/30 |
| **BorderLeaf1** | eth3 | 172.16.254.9/31 | Vlan 4094 | 172.16.255.9/30 |
|  | eth4 | 172.16.254.21/31 |  |  |
|  | eth5 | 172.16.254.33/31 |  |  |
| **BorderLeaf2** | eth3 | 172.16.254.11/31 | Vlan 4094 | 172.16.255.10/30 |
|  | eth4 | 172.16.254.23/31 |  |  |
|  | eth5 | 172.16.254.35/31 |  |  |

Table 4: Leaf-to-Leaf IP Addressing

## Underlay Network

In accordance with Arista best practices, exterior Border Gateway Protocol (eBGP) is used for the underlay network in the Layer 2 Leaf and Spine environment. As a popular choice for large data center and cloud scale networks, BGP offers key advantages over other routing protocols including multi-vendor interoperability, native traffic engineering capabilities, minimized information flooding, reliance on TCP instead of adjacency forming, reduced complexity and simplified troubleshooting, and mature and proven stability at scale. Specifically, the use of eBGP over interior Border Gateway Protocol (iBGP) offers a more simplistic configuration while ensuring all routes/paths are utilized effectively. To configure the most scalable design possible, the team utilized peer groups to help define relationships with neighboring devices. Below, in Figure 19 and 20, are sample BGP configurations from a spine and leaf perspective. Full BGP configurations for all devices can be found in the appendices.

A screenshot of a computer

Description automatically generated

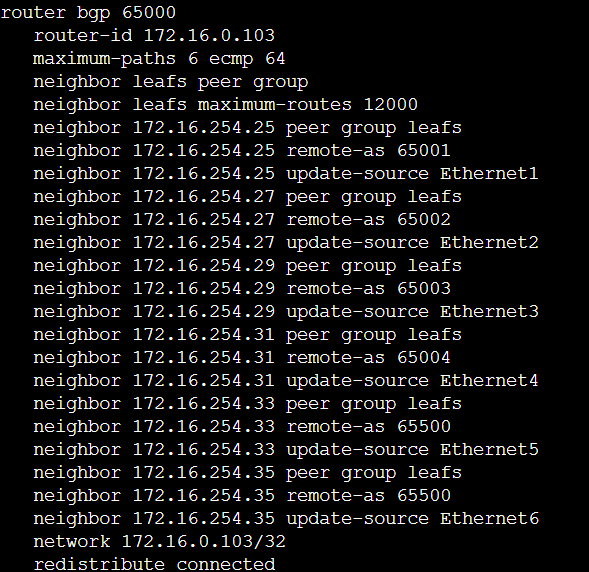
Figure 19: Sample Leaf eBGP Configuration

Figure 20: Sample Spine eBGP Configuration

The BGP ASN topology which showcases the leaf and spine devices is graphically shown in Figure 21 below.

A computer network diagram with blue dots

Description automatically generated with medium confidence

Figure 21: Underlay BGP Topology

## MLAG

MLAG (Multi-Chassis Link Aggregation) is a technology that allows two switches to appear as one switch to downstream devices. MLAG allows for redundancy, scalability, and load balancing. In the capstone topology, there are four Leaf and two Borderleaf devices that utilize MLAG. Leaf 1 and 2 are a MLAG pair, Leaf 3 and 4 are a MLAG pair, and Borderleaf 1 and 2 are a MLAG pair.

### Configuration

1. Arista's best practice for MLAG is to use VLAN 4094 for the MLAG peer connection.

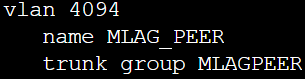


Figure 22: VLAN 4094 Creation

1. MLAG uses a port channel between the MLAG devices and the downstream device(s). As shown in the picture below, the trunk group from the VLAN 4094 configuration is applied to the port-channel between the leaf pair. The downstream links need to have the “mlag #” configuration.

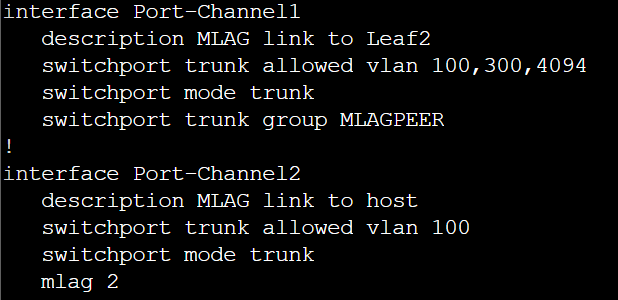


Figure 23: MLAG Port-Channel Configuration

1. Interface VLAN 4094 must be addressed for MLAG to function. The 2 devices should have unique IP addresses on the interfaces.

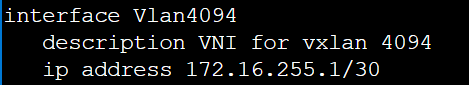


Figure 24: Interface VLAN 4094 Configuration

1. MLAG configuration between the MLAG peers should be identical apart from the peer-address. The peer address should be that of the peer switches interface VLAN 4094.

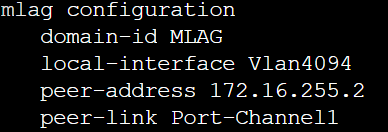


Figure 25: MLAG Configuration

1. Here is a “show mlag” command output for a working MLAG pair. This is from Leaf 1 of the capstone topology which is peering with Leaf 2.



Figure 26: Show MLAG Command Output

MLAG is not a very complex technology to set up. For troubleshooting, the best place to start is comparing the two devices’ configurations. The command “show mlag config-sanity" is a great tool when configuring MLAG as well.

## VXLAN

VXLAN (Virtual Extensible LAN) is a technology that works to encapsulate layer 2 ethernet frame inside of layer 3 UDP packets to tunnel between layer 2 networks. It is used to segment physical networks into many different virtual networks as needed. When EVPN is introduced into the network, VXLAN can also propagate layer 2 networks to be sent across layer 3 networks as well. It uses VTEP’s to determine traffic endpoints learned dynamically or statically. In this project's configuration, the VTEP’s are learned dynamically through EVPN. VXLAN also uses a 24 bit VNID that allows for more scalability and availability compared to traditional VLAN ID that is 12 bits.

### Configuration

1. To create the VXLAN, enter interface configuration mode with the interface vxlan # command.

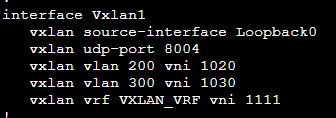


Figure 27: Interface VXLAN Configuration

1. Within the VXLAN configuration, you must specify where VXLAN is getting its source IP address for encapsulation.
   1. In this case, Lo0 is used as the source, and the IP address directly in the interface must be set, like any regular other interface configuration.



Figure 28: Interface VXLAN Configuration

1. When creating VXLAN instances you also need to specify VNI’s so that when it is encapsulated the VNI information will be included in the header as well
   1. This will also help VXLAN work smoothly with MLAG once both technologies are implemented within a network.
2. There are also different parts to VXLAN depending on your goal, depending on your usage you might need to implement VXLAN routing
   1. VXLAN routing will require you to configure virtual mac-address on each device and VRF (virtual routing and forwarding). See figure 19 for partial configuration of VRF, see the figured below for the rest of the configuration

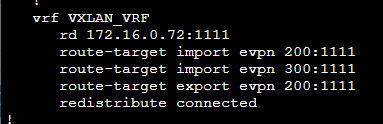


Figure 29: VXLAN VRF Configuration

1. These configurations for VXLAN help separate underlay and overlay traffic. This has some benefits, 1st this allows you to overlap IP addresses in the overlay network.

## EVPN

Ethernet VPN (EVPN) is a standards based BGP control plane to advertise MAC addresses, MAC and IP bindings, and IP Prefixes. EVPN, and its operation, works in tandem with VXLAN to create a data plane for building overlay networks in the data center. Current flood and learn models operate either with a multicast control plane, or ingress replication, where the operator manually configures the remote VTEPs in the flood list. Both are data-plane driven, that is, MACs are learned via flooding. In the IP multicast model, MACs are learned in the underlay via flooding to an IP multicast group which can be seen in the BGP configuration called EVPN groups.

The beauty of EVPN is that ingress replication of the floods to configure these VTEP endpoints and IP multicast is not needed. Which in turn, reduces bandwidth and traffic on the overlay network. A great example of this is to ping an endpoint from another endpoint, and you will get more than five responses, this is due to information being learned and encapsulating/decapsulated multiple times inside the data plane mentioned before - but if you were to do a traceroute, you would only see one hop. How is this possible? Check results and analysis section.

EVPN is a controller-less BGP MAC learning solution. It learns MAC/IP bindings in the VXLAN overlay, thus eliminating the problems of other solutions. The discovery and advertisement of the EVPN services can span across multiple vendors. This highlights the importance of this technology. It used a single control plane for multiple data-plane encapsulations and defines both L2 (Layer 2) and L3 (Layer 3) VPN services. As a network operator, your goal is to push towards simplicity and automation, BGP EVPN services will prove to be an extremely powerful tool. (don’t sleep on it)

### Configuration

Once MLAG and VXLAN are configured. BGP needs to be activated and the process needs to be started.

1. Add *service routing protocols model multi-agent* in your global configuration.
2. Configure Loopbacks for BGP Peering for all Spines and Leafs. This loopback should be a “/32”.

A black background with white text

Description automatically generated

Figure 30: Loopback Address used in EVPN

For MPLS, make sure you do this for all loopbacks (if applicable)

A screenshot of a computer screen

Description automatically generated

Figure 31: Loopbacks used in MPLS EVPN

1. Configure underlay eBGP.

Spine:

A screen shot of a computer program

Description automatically generated

Figure 32: Spine Underlay Configuration

Leaf: This is everything that has *peer group EVPN* in them

A screenshot of a computer program

Description automatically generated

Figure 33: Leaf BGP EVPN Configuration

1. Activate BGP EVPN in AFI IPV4 and in AFI EVPN on the Leafs.

A screen shot of a computer

Description automatically generated

Figure 34: Leaf BGP EVPN Address Family

1. Configure BGP EVPN Overlays on the Leafs

A screen shot of a computer

Description automatically generated

Figure 35: Leaf BGP EVPN Overlay

## CloudVision

To manage the configuration of network devices from a single location, all devices were integrated into Arista’s CloudVision SaaS environment. Some of CloudVision’s services include network provisioning and the use of configlets. In CloudVision, network provisioning is a hierarchical view of the topology. This tree view has the switches at the bottom of the tree which inherits the configurations specified in the containers above them. Containers are logical entities used to group the network devices. By default, there are two types of containers: the top-most container which is the Tenant, and the Undefined container which is a container for all of the devices that are registered through the CloudVision Portal (CVP) and are awaiting configuration. In network provisioning includes the use and management of configlets. Configlets are portion of configuration, or configuration templates, that a CloudVision user codes and maintains independently.

### Onboarding a Device to CloudVision

1. To onboard a device into CloudVision you first have to ensure that the device can hit the CloudVision site, we can do this by running the following command.



Figure 36: Ping to CloudVision

1. If the device cannot reach CloudVision, ensure that the *ip name-server vrf MGMT 8.8.8.8* command is entered to allow outside communication via the vrf MGMT interface.
2. Once we have confirmed that we can ping CloudVision we are going to move onto the next step which is installing the streaming agent extension onto our devices. Ensure that TerminAttr is installed by issuing the *show extensions* command.

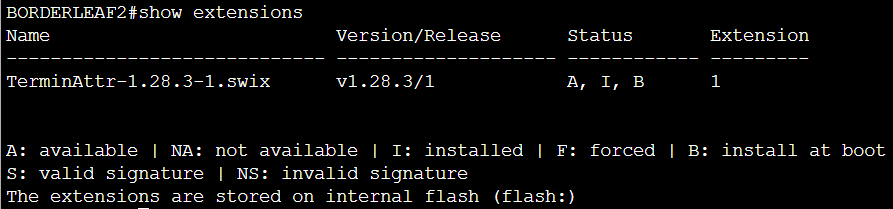


Figure 37: Installation of TerminAttr

1. After we install the streaming agent, we need to ensure that the clock is synchronized. We can check this by doing this command.

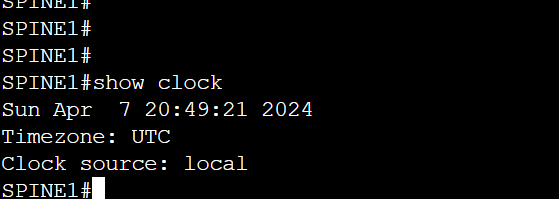


Figure 38: Show Clock Command

1. Next, we need to generate a token on the CloudVision website by clicking the Generate button.

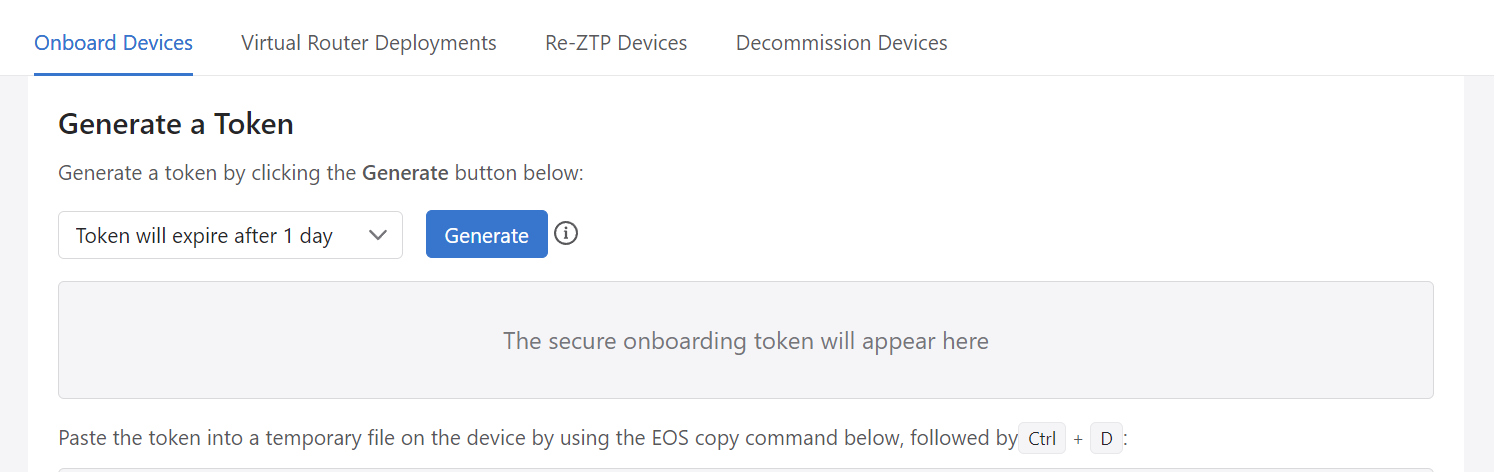


Figure 39: Token Generation in CloudVision Portal

1. After we generate the token, we need to add it to our device. We can do that by adding it to the temporary file on our device, we can open that temporary file with the following command.

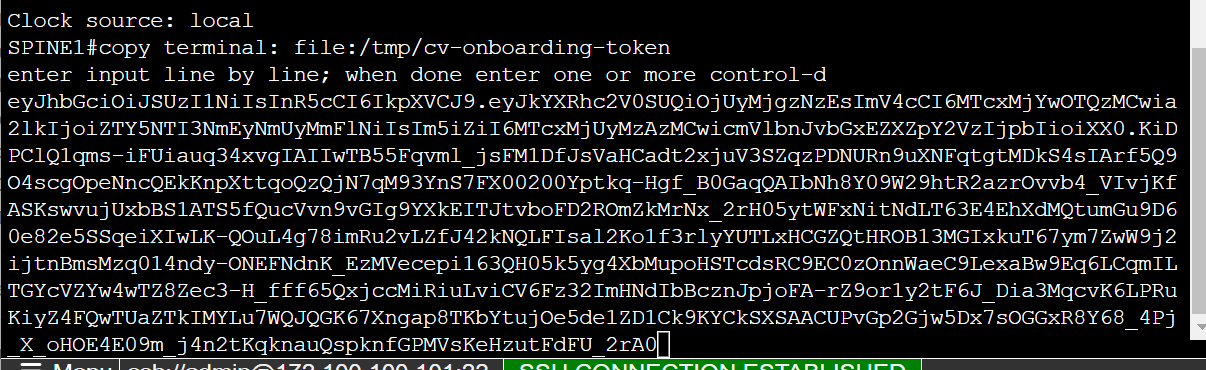


Figure 40: Pasting the Token on a Device

1. After you open the file paste the token we generated into the temporary file and then save and close the file.
2. Once the token is in the temporary file then we can complete the onboarding by using the following commands:

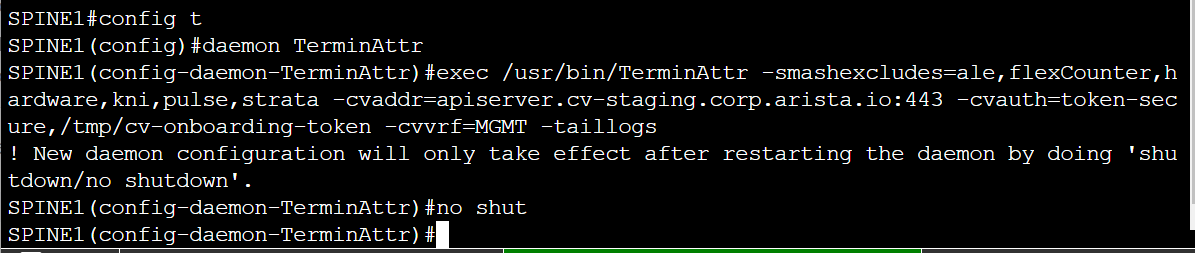


Figure 41: Starting TerminAttr to Complete Onboarding

1. The device is now successfully onboarded into CloudVision.

### Network Provisioning

1. The initial step to correctly provisioning the devices in CloudVision is to be in the provisioning tab which can be accessed in the left-hand ribbon.  
   A computer screen shot of a network

   Description automatically generated

Figure 42: Network Provisioning Tab

1. In the Network Provisioning window, you will see two containers named **Tenant** and **Undefined**. Under the **Undefined** container will hold all of the devices that were onboarded into CloudVision. Best practice to manage the devices is to create containers for every datacenter, within each datacenter to create a container for every POD (leaf-spine deployment), and to add the devices respectively to their POD. To create a container for the datacenter, right click on **Tenant** and add a container.

A computer screen shot of a network

Description automatically generated

Figure 43: Adding Datacenter Container

1. Name the container. It will now appear in the network provisioning view. Right click on the datacenter container and add a POD. The POD will appear under the datacenter container.

A diagram of a network

Description automatically generated

Figure 44: POD1 under UWStoutCapstone Datacenter Container

1. Now repeat adding PODs under the leaf-spine deployment container (POD1) to meet the respective devices.

A diagram of a network

Description automatically generated

Figure 45: Tree View of Containers

1. After adding all the containers into the datacenter container, click save. Now it’s time to add the network devices into their respective containers. To do that, right click on a bottom-most POD and click on add device.

A screenshot of a computer

Description automatically generated

Figure 46: Adding Device to POD Container

1. In the Undefined Devices window, select the respective device(s) you want to add to the POD.

A screenshot of a computer

Description automatically generated

Figure 47: List of devices under Undefined Devices

1. Go ahead and click on save at the bottom of the screen. The container and the devices you want to add will now be highlighted green. Before you save changes, you need to validate and reconcile any configurations on the devices. To do so, right click on the device, under Manage, click on Configlet.

A diagram of a computer network

Description automatically generated

Figure 48: Devices Highlighted Green Pending Changes

A screenshot of a computer

Description automatically generated

Figure 49: Manage Configlet under Device Management

1. Once in the configlet window, before adding any other designed configlets, validate the current running configuration by clicking on validate at the bottom.

A screenshot of a computer

Description automatically generated

Figure 50: Validating Device Configurations

1. After you click on validate, the Validate and Compare screen will open. A comparison between the running, designed, and proposed configurations will appear side my side. In the running configuration, all of the configurations will appear highlighted red if they are not in the designed configuration. To fix that, click on reconcile at the bottom.

A screenshot of a computer

Description automatically generated

Figure 51: Comparison between Running and Designed Configuration

1. After you’ve reconciled the configurations, there will no longer be any mismatches between the running and designed configuration. Click save.

A screenshot of a computer

Description automatically generated

Figure 52: Device Configurations after Reconciliation

1. Repeat steps 5-10 until you’ve completed adding devices to their containers and validated and reconciled any running configurations for the rest of the network devices in the leaf-spine deployment.

A computer screen shot of a network

Description automatically generated

Figure 53: Tree View of all Respective Devices Reconciled and Validated

1. Upon completion of adding all the devices to their respective PODS and their configurations have been reconciled and validated, go ahead and click save. Once the changes have been saved, the devices will appear under the **Undefined** container with a **T** above their device icon. If you hover over the name of the device, it will show that an existing task is pending on that device. This will happen to all the devices you’ve applied changes to. To complete these changes, you will need to go into the tasks window under Provisioning.

A diagram of a network

Description automatically generated

Figure 54: Pending Tasks on the Devices

A screenshot of a computer

Description automatically generated

Figure 55: List of Pending Tasks

1. In the tasks window, you will see all of the pending tasks for the devices that you’ve applied changes to. To review and approve the pending tasks, select the top most ID checkbox to select all of the pending tasks.

A screenshot of a computer

Description automatically generated

Figure 56: Selected List of Pending Tasks

1. Once that’s been done, right above the list, click on Create Change Control. In this Create Change Control window, it will ask to select an arrangement. Select either Series, Parallel, or choose a template if available. Click Create Change Control with 12 Tasks.

A screenshot of a computer

Description automatically generated

Figure 57: Create Change Control of Pending Tasks

1. This will bring you to the change control screen, click on the review and approve button on the top right-hand side.

A screenshot of a computer

Description automatically generated

Figure 58: Change Control Window

1. The review and approval screen will blow up with the designed and running configuration with any warnings and other details about the pending changes. Click on Approve and Execute at the bottom of the window. Enable the Execute immediately slider if needed.

A screenshot of a computer

Description automatically generated

Figure 59: Review Change Panel

1. Upon completion of change control, a success icon will be displayed next to the name of the change. If you go back into the Network Provisioning tab, you will see all of the devices in their respective PODS and no devices under the **Undefined** container.

A screenshot of a computer

Description automatically generated

Figure 60: Completion of Change Control

A screenshot of a computer

Description automatically generated

Figure 61: Tree View of Devices in Network Provisioning Tab

1. All of the devices have now been successfully added to the containers and Network Provisioning is complete.

### Creating and Pushing Configlets

1. Configlets, as mentioned earlier, are configuration templates. To create and push configlets to containers and devices, the initial step is to be inside the Configlets tab which is right under the Network Provisioning tab.

A screenshot of a computer

Description automatically generated

Figure 62: Configlets Management Window

1. In this window, this is where you will find any configlets that are created. To create one, click the drop arrow next to the add button or the plus sign.

A screenshot of a computer

Description automatically generated

Figure 63: Add and Create Configlets

1. Two options appear when clicked, Configlet builder and Configlets. When creating configlets, there are two methods: using python scripts or static CLI. To use python, you need to use the Configlet builder to generate the configurations. To use CLI commands you can statically create them through the Configlet option. Using a sample python script, select the Configlet builder.

A screenshot of a computer

Description automatically generated

Figure 64: Configlet Builder

1. Here, you can simply copy and paste a python script inside the Main Script field. Ensure to name the configlet.

A screenshot of a computer

Description automatically generated

Figure 65: Python Script

1. Once the script is pasted in correctly, under Form Design, in the devices drop box, select a device that the script is aimed toward. In this sample, the configlet is built for the Borderleaf devices. Click the generate button right below. This will return and show what the script is doing and be displayed under the built configuration plane.

A screenshot of a computer

Description automatically generated

Figure 66: Built Configlet using Python

1. Click save under at the bottom of the window. It will bring you back to the configlets management window where you will see the configlet you just created. To push that configlet, go into the Network Provisioning tab.
2. From there, select either the container or specific device you want to push the configlet to. Right click and under manage and choose configlet.

A screenshot of a computer

Description automatically generated

Figure 67: Managing Configlets Through Container

1. If applying to a container, select the configlet you want to push, click generate, and update. Clicking generate will create two more configlets specific to the respective devices under that container. Follow the same steps if applying to a specific device, but instead click validate instead of update.

A screenshot of a computer

Description automatically generated

Figure 68: Applied Configlet to Container

A screenshot of a computer

Description automatically generated

Figure 69: Validate Configlet to Specified Device

1. Save the configlets. The **T** will appear on the devices in the tree view again as well two pending tasks under the Tasks tab. Go into Tasks and review and approve through change control.
2. Upon success, the devices will now have those configlets applied to them.

# Challenges and Solutions

Below are some of the challenges the team overcame during this project.

## Docker

After getting through the lab requirement steps, some of the capstone team ran into a problem where docker was unable to build the topology. The capstone team could not find the exact cause of this issue, but it affected five out of the six members of the team. Below are the commands that the team used to fix the docker issue.

* sudo systemctl stop docker sudo rm -rf /var/lib/docker/network/files sudo systemctl start docker
* sudo apt-get update sudo apt-get upgrade docker.io iptables
* sudo systemctl restart docker
* sudo update-alternatives --set iptables /usr/sbin/iptables-legacy sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy

## ContainerLabs

The team ran into some issues when first importing the topology into container labs. UW-Stout school laptops were not strong enough to host every device in the final topology as it took up more memory than what the laptops had available. To sort this problem out, the topology and configurations were migrated to an onsite ubuntu machine that was hosted on an EXSI server in the UW-Stout lab. This allowed more memory to be added; as a result, the machine ran much more efficiently, allowing the team to modify configurations and properly test the network. This ended up being used as a testing environment while the team finalized the configuration for GCP. The project allowed for $300 of free credit and to avoid exhausting the credit early in the semester the team decided to start with other means of testing. Similar to the early iterations of ContainerLabs, the final deployment is in GCP (Google Cloud Platform), being hosted by a compute engine virtual machine.

## Enrolling Devices into CloudVision

When registering devices into CloudVision, the team experienced a weird issue pertaining to the devices to not be able to reach CloudVision to establish a stream connection, even though all the devices were able to reach CloudVision via a ping. After hours of troubleshooting with both the team and William, he was finally able to determine that the issue was with one singular command. Instead of using the command *ip name-server vrf default 8.8.8.8*, the command *ip name-server vrf MGMT 8.8.8.8* needed to be used to specify the management vrf. Once that command was added, each of the devices were able to be successfully enrolled.

Due to the ContainerLabs environment resetting if the virtual machine is turned off and back on, the virtual mac-addresses of the devices would change, resulting in the need to re-register each device into CloudVision again as the streaming agent uses the mac-address to establish the connection. After spending time trying to write a script to allow the devices to maintain the same address without any success, the team determined it would be easier to re-enroll the devices if needed for the remainder of the semester.

## Google Cloud Spot Instance

In order to save money throughout the semester, the team utilized a Spot Instance as the virtual machine type to host the topology. It is a cheaper option, but the main downside is that it shuts down when GCP needs to reclaim resources. Unfortunately, the day before the capstone team presented to the class and a week before presenting to the Advisory Committee, Google Cloud terminated the spot instance. This meant that the team had to re-configure the entire topology, along with re-registering each device into CloudVision to be able to complete the presentation.

To fix this issue, the team changed the type of virtual machine from a spot instance to a different type of virtual machine, not susceptible to being shut down by Google. This type of virtual machine is more expensive to run per hour, but due to budget savings throughout the semester, the team has saved more than enough of the free credit to be able to run this type of machine for the remainder of the semester.

# Results and Analysis

The following screenshots and examples showcase the successful implementation of the network and the technologies configured, as also showcased in our presentation.

## Working Underlay Network

The following screenshots show the working state of the underlay network, which is not participating in VXLAN and leverages eBGP as the control plane, with the physical leaf and spine devices acting as the data plane by physically routing the network traffic. In the example below, a ping and traceroute is sent from HostA to HostC in VLAN 400 and VLAN 402, which are not participating in VXLAN. Both are successful, with the traceroute taking three hops.

A screen shot of a computer

Description automatically generated

Figure 70: Ping in Underlay Network

A black background with white numbers

Description automatically generated

Figure 71: Traceroute in Underlay Network

Upon further inspection, the packet leaves HostA to its default gateway located on either Leaf1 or Leaf2, because both are participating in MLAG and appear as the same upstream switch from HostA’s perspective. Through BGP, the packet moves to Spine1, then to Borderleaf1, then to HostC on the respective point-to-point links that exist in the environment. The following diagram helps illustrate the underlay network not participating in VXLAN or EVPN.

A computer generated image of a network

Description automatically generated

Figure 72: Underlay Network Diagram

## Working Overlay Network

To test the functionality of the overlay network, another ping and traceroute were sent from HostA to HostC, this time on VLAN 100, which is participating in VXLAN. Both are still successful, and the traceroute now shows only one hop to get the traffic to the same destination.

A computer screen with white text

Description automatically generated

Figure 73: Ping in Overlay Network

A black background with white numbers

Description automatically generated

Figure 74: Traceroute in Overlay Network

To understand what is happening, the following diagram was created to showcase the control plane and data plane that exist in the overlay network. EVPN, acting as the control plane, is in charge of advertising the routes to endpoints based on the VTEP addresses they are directly connected too. The VNI’s in the data plane show which VTEP switch devices have what associated VLANs mapped to them, with VNI 1010 mapping to VLAN 100, VNI 1020 mapping to VLAN 200, and VNI 1030 mapping to VLAN 300.

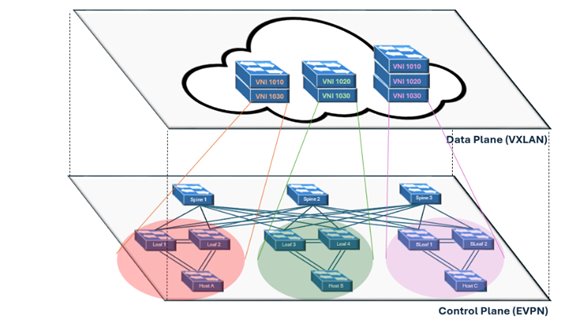


Figure 75: Overlay Network Diagram

From the time the traceroute is initiated on HostA to being received on HostC, the following process occurs:

1. HostA initiates a traceroute to HostC and broadcasts that out to on it’s connected default gateway as part of the underlay network.
2. The packet reaches either Leaf 1 or Leaf because both are participating in MLAG and appear as the same upstream switch from HostA’s perspective.
3. The receiving switch then broadcasts the BUM traffic out all of its interfaces, including VXLAN interfaces.
4. Since VLAN 100 is locally mapped to VNI 1010 on the receiving switch, it is able to use the VXLAN table advertised through the BGP EVPN control plane to find the addresses of the VTEPs connected to HostC, being either Borderleaf1 or Borderleaf2.
5. The packet is then encapsulated with the 24-bit VNI by VXLAN to be routed to VNI 1010 and in the overlay network.
6. Using HER Replication, VXLAN then forwards the encapsulated packet to only relevant destinations, being the devices with VNI 1010.
7. The receiving Borderleaf, due to having VNI 1010 locally mapped to VLAN 100, is able to decapsulate the packet which brings the packet back into the underlay network. The Borderleaf then directly forwards the packet to HostC.

Due to EVPN establishing neighbor relationships on the VTEP loopback addresses, VXLAN is able to send traffic directly from VTEP to VTEP, which showcases why the traceroute is only one hop and how the L2 network is extended over a L3 network, highlighting efficient intra-segment communication, minimizing latency, and optimizing network performance that is crucial for scalable and agile data center operations.

**Working EVPN**

The command *show bgp summary* lets us see who’s participating in the BGP routes, either by IPv4 AFI or the EVPN AFI.

A screen shot of a computer

Description automatically generated

Figure 76: Show BGP Summary Command

The command *show bgp evpn* will show a lot of information. This information is what EVPN stores to better control the routes it knows. Either IMET or MAC-IP hop types. These types of EVPN routes are type 5 and 2. EVPN uses IMET information to directly route traffic to the destination without flooding other devices.

A screenshot of a computer program

Description automatically generated

Figure 77: Show BGP EVPN Command

## CloudVision Configlets

In CloudVision, configlets are small chunks of configurations that can be easily applied to devices in seconds. The following configlet adds many loopbacks addresses to the boarder-leaf devices to show that configlets can be applied to devices easily without interfering with the current network.

A screenshot of a computer program

Description automatically generated

Figure 78: Loopback Configlet

After following the process to apply the loopback configlet to a Borderleaf, as outlined in the network configuration section above, the Borderleaf will now have the large number of created loopbacks and their respective IP addresses, as shown in the following sample output screenshot.

A screen shot of a computer

Description automatically generated

Figure 79: Sample Output of Loopback Configlet

# Conclusion

Through the completion of this capstone project, the team thought of a few key takeaways to best explain the experience. Coming into the project, none of the team members had any experience with either MLAG, VXLAN, or EVPN, or have even heard of the protocols. After spending a semester working on configuring them and understanding their role in the network, the team sees the enormous benefits of using these protocols, especially in a data center environment. Being able to extend a L2 across a L3 network allows not only for a more efficient network with reduced network congestion, but it also solves the issue of poor application design for some applications. The use of the leaf and spine architecture was also a new type of topology for the team, but we quickly realized the benefits that it can provide, such as being easily replicated and horizontally scaled to match the needs of a datacenter.

The team also got the opportunity to work on Arista hardware, which was also completely new. Up until this point, the team had mostly only worked on Cisco equipment, which makes up less than 50 percent of all hardware implemented today. The team especially enjoyed the features of Arista CLI, particularly with how much it is designed to help increase productivity. Additionally, the project allowed the team to see the benefits of using industry standard technologies in a network. Leveraging these solutions not only enhances flexibility and scalability, but also opens the door for collaboration with external contributors and multiple vendors.

Finally, the team appreciated the content of this capstone project as it presented the opportunity to learn new skills and technologies that the curriculum did not teach. By gaining these valuable skills, the team is able to take this knowledge into the professional world, enhancing our readiness to tackle real-world challenges and contribute in each of our respective fields. This capstone project has been more than just part of an academic requirement; it has been a journey that has equipped us with the tools and mindset needed to thrive in the dynamic field of networking.

# Acknowledgements

The Arista Capstone team extends heartfelt gratitude to the individuals and organizations whose support was pivotal to the successful completion of this project.

First and foremost, the team wishes to express sincere appreciation to William Goss, the project sponsor from Arista. His devoted guidance, insightful feedback, and unwavering encouragement at every stage of the project were invaluable. From weekly meetings to providing training content on Arista hardware and project technologies, to dedicating countless hours to troubleshoot issues and enhance the team's understanding of technical concepts, William's commitment played a vital role in the team’s success.

The team also extends its gratitude to Arista Networks, the project sponsor company, for generously providing the cEOS images used in the project and hosting the CloudVision SaaS environment. The hands-on experience with new hardware CLI has offered invaluable insights and knowledge that will undoubtedly benefit each member in their future careers.

To Holly Yuan, Program Director, and the esteemed professors of the Computer Networking and Information Technology program at the University of Wisconsin – Stout, the team offers profound thanks for their unwavering support over the past four years. The program's emphasis on practical knowledge and high standards has equipped the team with the essential tools to successfully fulfill all objectives outlined in this project.

In conclusion, this whitepaper stands as a collaborative effort, and the team is indebted to the individuals and companies mentioned above. Their collective contributions have left a positive permanent mark on this work.

Arista Capstone Team

# Document Control with History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Date | Description of Changes | Author |
| 1.0 | 2.29.24 | Initial draft submitted | Branden Ulm |
| 1.1 | 3.4.24 | Draft of lab setup | Cole Melson |
| 1.2 | 3.5.24 | Completed lab network requirements and setup | Cole Melson |
| 1.3 | 3.7.24 | Added Lab Requirements – WSL and Linux OS, Created problems and Solution section, Added Challenge + Solutions about docker + containerlab, Created and added to subheadings for introduction | Cole Melson, Michael Long, Lucas Rupp |
| 2.0 | 3.8.24 | Rework of introduction and problem description sections. Formatted paper in accordance to APA formatting guides | Branden Ulm |
| 2.1 | 4.1.24 | Reworded some areas in the “ContainerLabs” section as well as the “Lab Requirements” introduction | Cole Melson |
| 2.2 | 4.3.24 | Wrote “MLAG” section of “Implementation Steps” | Cole Melson |
| 2.3 | 4.4.24 | Wrote IP Addressing Section | Javier Perez Sanchez |
| 2.4 | 4.7.24 | Wrote Underlay Network Section, added table of figures and table of tables, added figure captions to all pictured, added better acceptance page, document formatting, device configurations in appendix | Branden Ulm |
| 2.5 | 4.7.24 | Wrote Onboarding to CloudVision section | Michael Long |
| 2.6 | 4.7.24 | Wrote EVPN section | Branden Behrens |
| 2.7 | 4.7.24 | Wrote GCP VM section and VXLAN section | Lucas Rupp |
| 3.0 | 4.7.24 | Final proofreading and draft editing, submit draft to Canvas | Branden Ulm |
| 3.1 | 4.25.24 | Addition of Working Underlay and Overlay sections in Results and Analysis | Michael Long |
| 3.2 | 4.25.24 | Addition of Configlet section in Results and Analysis | Javier Perez Sanchez |
| 4.0 | 4.25.24 | Final proofreading and addition of figure captions, submit draft to Canvas | Branden Ulm |
| 4.3 | 4.29.24 | Addition of Executive Summary, Conclusion, and Acknowledgements sections along with additional diagrams and general proofreading and editing | Branden Ulm |
| 4.5 | 4.29.24 | Addition of Networking Provisioning and Applying Configlet sections | Javier Perez Sanchez |
| 5.0 | 4.30.24 | Signatures by capstone members, proofreading before sent to sponsor for approval | All |

This document will be kept under revision control.

# Appendix

## GitHub Page

Link to the GitHub page for all device configurations along with the presentation and capstone poster: <https://github.com/C0083M/Arista_Capstone>

## ContainerLabs Topology YAML File

---

# -----------------------------------------------------------------------------------------------

# Arista Capstone Topology 2024

# 3 Spines & 4 Leafs - each leaf pair participating in MLAG with 2 BorderLeafs participating in MLAG

# -----------------------------------------------------------------------------------------------

name: Arista\_Capstone\_Topology

prefix: ""

mgmt:

network: mgmt

ipv4-subnet: 172.100.100.0/24

topology:

defaults:

env:

INTFTYPE: et

kinds:

ceos:

# adjust to the name of the docker image you imported

# format:

# image: <REPOSITORY>:<TAG>

# Check docker images with the following docker command:

# docker images

image: ceos:4.30.5M

linux:

image: mitchv85/ohv-host

nodes:

###########################################

# Graphite Node #

# Provides Topology Graphing and the #

# ability to connect to nodes in the #

# topology via a web/ssh session #

###########################################

graphite:

kind: linux

image: netreplica/graphite

mgmt-ipv4: 172.100.100.109

env:

HOST\_CONNECTION: ${SSH\_CONNECTION}

binds:

- \_\_clabDir\_\_/topology-data.json:/htdocs/default/default.json:ro

- \_\_clabDir\_\_/ansible-inventory.yml:/htdocs/lab/default/ansible-inventory.yml:ro

ports:

- 80:80

exec:

- sh -c 'graphite\_motd.sh 80'

labels:

graph-hide: yes

#########################

# SPINE #

#########################

#https://containerlab.dev/manual/topo-def-file/#nodes

SPINE1:

kind: ceos

mgmt-ipv4: 172.100.100.101

startup-config: configs/SPINE1.cfg

ports:

- '22001:22'

- '8001:80'

- '44301:443'

labels:

graph-level: 1

graph-icon: switch

SPINE2:

kind: ceos

mgmt-ipv4: 172.100.100.102

startup-config: configs/SPINE2.cfg

ports:

- '22002:22'

- '8002:80'

- '44302:443'

labels:

graph-level: 1

graph-icon: switch

SPINE3:

kind: ceos

mgmt-ipv4: 172.100.100.103

startup-config: configs/SPINE3.cfg

ports:

- '22003:22'

- '8003:80'

- '44303:443'

labels:

graph-level: 1

graph-icon: switch

#########################

# LEAF #

#########################

LEAF1:

kind: ceos

mgmt-ipv4: 172.100.100.104

startup-config: configs/LEAF1.cfg

ports:

- '22004:22'

- '8004:80'

- '44304:443'

labels:

graph-level: 2

graph-icon: switch

LEAF2:

kind: ceos

mgmt-ipv4: 172.100.100.105

startup-config: configs/LEAF2.cfg

ports:

- '22005:22'

- '8005:80'

- '44305:443'

labels:

graph-level: 2

graph-icon: switch

LEAF3:

kind: ceos

mgmt-ipv4: 172.100.100.106

startup-config: configs/LEAF3.cfg

ports:

- '22006:22'

- '8006:80'

- '44306:443'

labels:

graph-level: 2

graph-icon: switch

LEAF4:

kind: ceos

mgmt-ipv4: 172.100.100.107

startup-config: configs/LEAF4.cfg

ports:

- '22007:22'

- '8007:80'

- '44307:443'

labels:

graph-level: 2

graph-icon: switch

Y-BORDERLEAF1:

kind: ceos

mgmt-ipv4: 172.100.100.108

startup-config: configs/BORDERLEAF1.cfg

ports:

- '22008:22'

- '8008:80'

- '44308:443'

labels:

graph-level: 2

graph-icon: switch

Z-BORDERLEAF2:

kind: ceos

mgmt-ipv4: 172.100.100.110

startup-config: configs/BORDERLEAF2.cfg

ports:

- '22009:22'

- '8009:80'

- '44309:443'

labels:

graph-level: 2

graph-icon: switch

###########################################

# HOST #

###########################################

HOSTA:

kind: ceos

mgmt-ipv4: 172.100.100.111

ports:

- '22210:22'

- '8010:80'

- '44310:443'

labels:

graph-level: 3

graph-icon: switch

HOSTB:

kind: ceos

mgmt-ipv4: 172.100.100.112

ports:

- '22211:22'

- '8011:80'

- '44311:443'

labels:

graph-level: 3

graph-icon: switch

HOSTC:

kind: linux

mgmt-ipv4: 172.100.100.113

ports:

- '22212:22'

- '8012:80'

- '44312:443'

labels:

graph-level: 3

graph-icon: switch

#https://containerlab.dev/manual/topo-def-file/#links

links:

####################

# SPINE1 to LEAF #

####################

- endpoints: ["SPINE1:et2", "LEAF1:et3"]

- endpoints: ["SPINE1:et3", "LEAF2:et3"]

- endpoints: ["SPINE1:et4", "LEAF3:et3"]

- endpoints: ["SPINE1:et5", "LEAF4:et3"]

- endpoints: ["SPINE1:et6", "Y-BORDERLEAF1:et3"]

- endpoints: ["SPINE1:et7", "Z-BORDERLEAF2:et3"]

####################

# SPINE2 to LEAF #

####################

- endpoints: ["SPINE2:et2", "LEAF1:et4"]

- endpoints: ["SPINE2:et3", "LEAF2:et4"]

- endpoints: ["SPINE2:et4", "LEAF3:et4"]

- endpoints: ["SPINE2:et5", "LEAF4:et4"]

- endpoints: ["SPINE2:et6", "Y-BORDERLEAF1:et4"]

- endpoints: ["SPINE2:et7", "Z-BORDERLEAF2:et4"]

####################

# SPINE3 to LEAF #

####################

- endpoints: ["SPINE3:et1", "LEAF1:et5"]

- endpoints: ["SPINE3:et2", "LEAF2:et5"]

- endpoints: ["SPINE3:et3", "LEAF3:et5"]

- endpoints: ["SPINE3:et4", "LEAF4:et5"]

- endpoints: ["SPINE3:et5", "Y-BORDERLEAF1:et5"]

- endpoints: ["SPINE3:et6", "Z-BORDERLEAF2:et5"]

####################

# LEAF1 to LEAF2 #

####################

- endpoints: ["LEAF1:et1", "LEAF2:et1"]

- endpoints: ["LEAF1:et2", "LEAF2:et2"]

####################

# LEAF3 to LEAF4 #

####################

- endpoints: ["LEAF3:et1", "LEAF4:et1"]

- endpoints: ["LEAF3:et2", "LEAF4:et2"]

################################

# BORDERLEAF1 to BORDERLEAF2 #

################################

- endpoints: ["Y-BORDERLEAF1:et1", "Z-BORDERLEAF2:et1"]

- endpoints: ["Y-BORDERLEAF1:et2", "Z-BORDERLEAF2:et2"]

####################

# HOSTA #

####################

- endpoints: ["HOSTA:et1", "LEAF1:et6"]

- endpoints: ["HOSTA:et3", "LEAF1:et7"]

- endpoints: ["HOSTA:et2", "LEAF2:et6"]

- endpoints: ["HOSTA:et4", "LEAF2:et7"]

####################

# HOSTB #

####################

- endpoints: ["HOSTB:et1", "LEAF3:et6"]

- endpoints: ["HOSTB:et3", "LEAF3:et7"]

- endpoints: ["HOSTB:et2", "LEAF4:et6"]

- endpoints: ["HOSTB:et4", "LEAF4:et7"]

####################

# HOSTC #

####################

- endpoints: ["HOSTC:et1", "Y-BORDERLEAF1:et6"]

- endpoints: ["HOSTC:et3", "Y-BORDERLEAF1:et7"]

- endpoints: ["HOSTC:et2", "Z-BORDERLEAF2:et6"]

- endpoints: ["HOSTC:et4", "Z-BORDERLEAF2:et7"]

## Spine 1 Configuration

! device: SPINE1 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$wzRa9QmQU6EpR7RM$2XlyWjxmT66T3eh43JESfjzUisHhgwDGuYjXBf1z/EwDQIgshnFr4s5g3v52ca7GZw2zaOD9UFsNonP5Wjd8u/

!

daemon TerminAttr

exec /usr/bin/TerminAttr -smashexcludes=ale,flexCounter,hardware,kni,pulse,strata -cvaddr=apiserver.cv-staging.corp.arista.io:443 -cvauth=token-secure,/tmp/cv-onboarding-token -cvvrf=MGMT -taillogs

no shutdown

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname SPINE1

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vrf instance MGMT

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Ethernet1

!

interface Ethernet2

description Connected to leaf1-eth3

no switchport

ip address 172.16.254.0/31

!

interface Ethernet3

description connected to leaf1-eth3

no switchport

ip address 172.16.254.2/31

!

interface Ethernet4

description Connected to leaf3-eth3

no switchport

ip address 172.16.254.4/31

!

interface Ethernet5

description Connected to leaf4-eth3

no switchport

ip address 172.16.254.6/31

!

interface Ethernet6

description connected to BorderLeaf1-eth3

no switchport

ip address 172.16.254.8/31

!

interface Ethernet7

description Connected to BorderLeaf2-eth3

no switchport

ip address 172.16.254.10/31

!

interface Loopback0

ip address 172.16.0.101/32

!

interface Management0

vrf MGMT

ip address 172.100.100.101/24

!

ip routing

no ip routing vrf MGMT

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

router bgp 65000

router-id 172.16.0.101

maximum-paths 6 ecmp 64

neighbor leafs peer group

neighbor leafs maximum-routes 12000

neighbor 172.16.254.1 peer group leafs

neighbor 172.16.254.1 remote-as 65001

neighbor 172.16.254.3 peer group leafs

neighbor 172.16.254.3 remote-as 65002

neighbor 172.16.254.5 peer group leafs

neighbor 172.16.254.5 remote-as 65003

neighbor 172.16.254.7 peer group leafs

neighbor 172.16.254.7 remote-as 65004

neighbor 172.16.254.9 peer group leafs

neighbor 172.16.254.9 remote-as 65500

neighbor 172.16.254.11 peer group leafs

neighbor 172.16.254.11 remote-as 65500

!

management ssh

vrf MGMT

no shutdown

!

end

## Spine 2 Configuration

! device: spine2 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$Ed1zqC1XoZsjcK2W$1BaTO0h.l9lTvCfDRXr0LuMvhp21SPC0SI8QsOCRRDVyXy8/Suxw9XHCBeCIhukeKjcgbtmA1DLcJEcISME1W/

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname spine2

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vrf instance MGMT

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Ethernet1

!

interface Ethernet2

description connection to leaf1-eth4

no switchport

ip address 172.16.254.12/31

!

interface Ethernet3

description connection to leaf2-eth4

no switchport

ip address 172.16.254.14/31

!

interface Ethernet4

description connection to leaf3-eth4

no switchport

ip address 172.16.254.16/31

!

interface Ethernet5

description connection to leaf4-eth4

no switchport

ip address 172.16.254.18/31

!

interface Ethernet6

description connection to Borderleaf1-eth4

no switchport

ip address 172.16.254.20/31

!

interface Ethernet7

description connection to Borderleaf2-eth4

no switchport

ip address 172.16.254.22/31

!

interface Ethernet8

!

interface Loopback0

ip address 172.16.0.102/32

!

interface Management0

vrf MGMT

ip address 172.100.100.102/24

!

interface Management1

!

ip routing

no ip routing vrf MGMT

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

router bgp 65000

router-id 172.16.0.102

maximum-paths 6 ecmp 64

neighbor leafs peer group

neighbor leafs maximum-routes 12000

neighbor 172.16.254.13 peer group leafs

neighbor 172.16.254.13 remote-as 65001

neighbor 172.16.254.13 update-source Ethernet2

neighbor 172.16.254.15 peer group leafs

neighbor 172.16.254.15 remote-as 65002

neighbor 172.16.254.15 update-source Ethernet3

neighbor 172.16.254.17 peer group leafs

neighbor 172.16.254.17 remote-as 65003

neighbor 172.16.254.17 update-source Ethernet4

neighbor 172.16.254.19 peer group leafs

neighbor 172.16.254.19 remote-as 65004

neighbor 172.16.254.19 update-source Ethernet5

neighbor 172.16.254.21 peer group leafs

neighbor 172.16.254.21 remote-as 65500

neighbor 172.16.254.23 peer group leafs

neighbor 172.16.254.23 remote-as 65500

network 172.16.0.102/32

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Spine 3 Configuration

! device: spine3 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$HLB77PC4EhaZl8.7$S4lJV3lZWPgMf4s501NFZU6ea4eCmjkuNV9H4.NeT954uk0PBG6aykc8lAFx7EwZlwJRdID/m2w59YDfJBtb8/

!

daemon TerminAttr

exec /usr/bin/TerminAttr -smashexcludes=ale,flexCounter,hardware,kni,pulse,strata -cvaddr=apiserver.cv-staging.corp.arista.io:443 -cvauth=token-secure,/tmp/cv-onboarding-token -cvvrf=MGMT -taillogs

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname spine3

ip name-server vrf default 8.8.8.8

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vrf instance MGMT

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Ethernet1

description connection to leaf1-eth5

no switchport

ip address 172.16.254.24/31

!

interface Ethernet2

description connection to leaf2-eth5

no switchport

ip address 172.16.254.26/31

!

interface Ethernet3

description connection to leaf3-eth5

no switchport

ip address 172.16.254.28/31

!

interface Ethernet4

description connection to leaf4-eth5

no switchport

ip address 172.16.254.30/31

!

interface Ethernet5

description connection to Borderleaf1-eth5

no switchport

ip address 172.16.254.32/31

!

interface Ethernet6

description connection to Borderleaf2-eth5

no switchport

ip address 172.16.254.34/31

!

interface Ethernet7

!

interface Ethernet8

!

interface Loopback0

ip address 172.16.0.103/32

!

interface Management0

vrf MGMT

ip address 172.100.100.103/24

!

interface Management1

!

ip routing

no ip routing vrf MGMT

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

router bgp 65000

router-id 172.16.0.103

maximum-paths 6 ecmp 64

neighbor leafs peer group

neighbor leafs maximum-routes 12000

neighbor 172.16.254.25 peer group leafs

neighbor 172.16.254.25 remote-as 65001

neighbor 172.16.254.25 update-source Ethernet1

neighbor 172.16.254.27 peer group leafs

neighbor 172.16.254.27 remote-as 65002

neighbor 172.16.254.27 update-source Ethernet2

neighbor 172.16.254.29 peer group leafs

neighbor 172.16.254.29 remote-as 65003

neighbor 172.16.254.29 update-source Ethernet3

neighbor 172.16.254.31 peer group leafs

neighbor 172.16.254.31 remote-as 65004

neighbor 172.16.254.31 update-source Ethernet4

neighbor 172.16.254.33 peer group leafs

neighbor 172.16.254.33 remote-as 65500

neighbor 172.16.254.33 update-source Ethernet5

neighbor 172.16.254.35 peer group leafs

neighbor 172.16.254.35 remote-as 65500

neighbor 172.16.254.35 update-source Ethernet6

network 172.16.0.103/32

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Leaf 1 Configuration

! device: Leaf1 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$veZGzQYNzEqpT4/c$d9amDEAJ1FnUSPBayq44dtY9e463SJsHRm4C2fECrQ4xNKWUkKPY6r378KpfiwS8s9zd.bMc3PINTFI9Sk0GN0

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname Leaf1

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vlan 100

name MLAG\_L1-L2

!

vlan 300

name MLAG\_hosts

!

vlan 4094

name MLAG\_PEER

trunk group MLAGPEER

!

vrf instance MGMT

!

vrf instance VXLAN\_VRF

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Port-Channel1

description MLAG link to Leaf2

switchport trunk allowed vlan 100,300,4094

switchport mode trunk

switchport trunk group MLAGPEER

!

interface Port-Channel2

description MLAG link to host

switchport trunk allowed vlan 100

switchport mode trunk

mlag 2

!

interface Port-Channel3

description MLAG Downlink - Host11

switchport trunk allowed vlan 300

switchport mode trunk

mlag 3

!

interface Ethernet1

description MLAG link to leaf2-eth1

channel-group 1 mode active

!

interface Ethernet2

description MLAG link to leaf2-eth2

channel-group 1 mode active

!

interface Ethernet3

description connection to Spine1-eth2

no switchport

ip address 172.16.254.1/31

!

interface Ethernet4

description connection to Spine2-eth2

no switchport

ip address 172.16.254.13/31

!

interface Ethernet5

description connection to Spine3-eth1

no switchport

ip address 172.16.254.25/31

!

interface Ethernet6

switchport access vlan 300

channel-group 3 mode active

!

interface Ethernet7

description MLAG Downlink - Host10

switchport access vlan 300

channel-group 3 mode active

!

interface Loopback0

description Loopback 0 for VTEP

ip address 172.16.0.70/32

!

interface Loopback1

description BGP peering loopback

ip address 172.16.0.68/32

!

interface Management0

vrf MGMT

ip address 172.100.100.104/24

!

interface Management1

!

interface Vlan100

description VNI for vxlan 1010

vrf VXLAN\_VRF

ip address 172.16.101.11/24

ip virtual-router address 172.16.101.1

!

interface Vlan300

description VNI for vxlan 1030

ip address 172.16.103.11/24

ip virtual-router address 172.16.103.1

!

interface Vlan4094

description VNI for vxlan 4094

ip address 172.16.255.1/30

!

interface Vxlan1

vxlan source-interface Loopback0

vxlan udp-port 4789

vxlan flood vtep learned data-plane

vxlan vlan 100 vni 1010

vxlan vlan 300 vni 1030

vxlan vrf VXLAN\_VRF vni 1111

!

ip virtual-router mac-address 00:1b:74:00:00:01

!

ip routing

no ip routing vrf MGMT

ip routing vrf VXLAN\_VRF

!

ip prefix-list local-SVI

seq 10 permit 172.17.0.70/32

!

ip prefix-list mgmt-route

seq 10 deny 172.100.100.0/24 le 32

seq 20 deny 0.0.0.0/0

seq 100 permit 0.0.0.0/0 le 32

!

mlag configuration

domain-id MLAG

local-interface Vlan4094

peer-address 172.16.255.2

peer-link Port-Channel1

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

route-map mgmt-route permit 10

match ip address prefix-list mgmt-route

!

route-map redist-SVI permit 10

match ip address prefix-list local-SVI

!

router bgp 65001

router-id 172.16.0.70

rd auto

maximum-paths 6

neighbor SPINE peer group

neighbor SPINE remote-as 65000

neighbor SPINE send-community standard extended

neighbor evpn peer group

neighbor evpn update-source Loopback0

neighbor evpn ebgp-multihop 3

neighbor evpn send-community extended

neighbor evpn maximum-routes 0

neighbor 172.16.0.69 peer group evpn

neighbor 172.16.0.69 remote-as 65002

neighbor 172.16.0.69 update-source Loopback1

neighbor 172.16.0.72 peer group evpn

neighbor 172.16.0.72 remote-as 65003

neighbor 172.16.0.72 update-source Loopback0

neighbor 172.16.0.73 peer group evpn

neighbor 172.16.0.73 remote-as 65004

neighbor 172.16.0.73 update-source Loopback0

neighbor 172.16.0.201 peer group evpn

neighbor 172.16.0.201 remote-as 65500

neighbor 172.16.0.201 update-source Loopback0

neighbor 172.16.0.202 peer group evpn

neighbor 172.16.0.202 remote-as 65500

neighbor 172.16.0.202 update-source Loopback0

neighbor 172.16.101.1 peer group evpn

neighbor 172.16.101.1 update-source Vlan100

neighbor 172.16.103.1 peer group evpn

neighbor 172.16.103.1 update-source Vlan300

neighbor 172.16.254.0 peer group SPINE

neighbor 172.16.254.0 update-source Ethernet3

neighbor 172.16.254.12 peer group SPINE

neighbor 172.16.254.12 update-source Ethernet4

neighbor 172.16.254.24 peer group SPINE

neighbor 172.16.254.24 update-source Ethernet5

neighbor 172.16.255.2 remote-as 65002

neighbor 172.16.255.2 update-source Vlan4094

neighbor 172.16.255.2 maximum-routes 12000

redistribute connected route-map redist-SVI

!

vlan 100

rd 172.16.0.70:100

route-target both 100:1010

redistribute learned

!

vlan 300

rd 172.16.0.70:300

route-target both 300:1030

redistribute learned

!

address-family evpn

no bgp next-hop-unchanged

neighbor evpn activate

!

address-family ipv4

neighbor SPINE activate

no neighbor evpn activate

network 172.16.0.68/32

network 172.16.0.69/32

network 172.16.0.70/32

network 172.16.0.71/32

network 172.16.101.0/24

network 172.16.103.0/24

network 172.16.254.0/31

network 172.16.254.12/31

network 172.16.254.24/31

network 172.16.255.0/30

network 0.0.0.0/0

redistribute connected

!

vrf VXLAN\_VRF

rd 172.16.0.70:1111

route-target import evpn 100:1111

route-target import evpn 300:1111

route-target export evpn 100:1111

router-id 172.16.0.70

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Leaf 2 Configuration

! device: Leaf2 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$VX0d8SK02yBGbW/q$zjr/71V/xyycCn.Kz70l.NXBe3qORGPUJ6yeNmrcK42PxgQMGxCigGdflvuEkrXLBDrmfb.3Vv/0Yco2P8/K80

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname Leaf2

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vlan 100

name MLAG\_L1-L2

!

vlan 300

name MLAG\_hosts

!

vlan 4094

name MLAG\_PEER

trunk group MLAGPEER

!

vrf instance MGMT

!

vrf instance VXLAN\_VRF

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Port-Channel1

description MLAG link to Leaf1

switchport trunk allowed vlan 100,300,4094

switchport mode trunk

switchport trunk group MLAGPEER

!

interface Port-Channel2

description MLAG Downlink - Host1

switchport trunk allowed vlan 100

switchport mode trunk

mlag 2

!

interface Port-Channel3

description MLAG Downliank - Host11

switchport trunk allowed vlan 300

switchport mode trunk

mlag 3

!

interface Ethernet1

description MLAG link to leaf1-eth1

channel-group 1 mode active

!

interface Ethernet2

description MLAG link to leaf1-eth2

channel-group 1 mode active

!

interface Ethernet3

description connection to Spine1-eth3

no switchport

ip address 172.16.254.3/31

!

interface Ethernet4

description connection to Spine2-eth3

no switchport

ip address 172.16.254.15/31

!

interface Ethernet5

description connection to Spine3-eth2

no switchport

ip address 172.16.254.27/31

!

interface Ethernet6

switchport access vlan 300

channel-group 3 mode active

!

interface Ethernet7

description MLAG Downlink - Host11

switchport access vlan 300

channel-group 3 mode active

!

interface Loopback0

description Loopback 0 for VTEP

ip address 172.16.0.70/32

!

interface Loopback1

description BGP peering loopback

ip address 172.16.0.69/32

!

interface Management0

vrf MGMT

ip address 172.100.100.105/24

!

interface Management1

!

interface Vlan100

description VNI for vxlan 1010

vrf VXLAN\_VRF

ip address 172.16.101.12/24

ip virtual-router address 172.16.101.1

!

interface Vlan300

description VNI for vxlan 1030

ip address 172.16.103.12/24

ip virtual-router address 172.16.103.1

!

interface Vlan4094

description VNI for vxlan 4094

ip address 172.16.255.2/30

!

interface Vxlan1

vxlan source-interface Loopback0

vxlan udp-port 4789

vxlan flood vtep learned data-plane

vxlan vlan 100 vni 1010

vxlan vlan 300 vni 1030

vxlan vrf VXLAN\_VRF vni 1111

!

ip virtual-router mac-address 00:1b:74:00:00:01

!

ip routing

no ip routing vrf MGMT

ip routing vrf VXLAN\_VRF

!

ip prefix-list local-SVI

seq 10 permit 172.16.0.70/32

!

ip prefix-list mgmt-route

seq 10 deny 172.100.100.0/24 le 32

seq 20 deny 0.0.0.0/0

seq 100 permit 0.0.0.0/0 le 32

!

mlag configuration

domain-id MLAG

local-interface Vlan4094

peer-address 172.16.255.1

peer-link Port-Channel1

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

route-map mgmt-route permit 10

match ip address prefix-list mgmt-route

!

route-map redist-SVI permit 10

match ip address prefix-list local-SVI

!

router bgp 65002

router-id 172.16.0.71

rd auto

maximum-paths 6

neighbor SPINE peer group

neighbor SPINE remote-as 65000

neighbor SPINE send-community standard extended

neighbor evpn peer group

neighbor evpn update-source Loopback0

neighbor evpn ebgp-multihop 3

neighbor evpn send-community extended

neighbor evpn maximum-routes 0

neighbor 172.16.0.68 peer group evpn

neighbor 172.16.0.68 remote-as 65001

neighbor 172.16.0.68 update-source Loopback1

neighbor 172.16.0.72 peer group evpn

neighbor 172.16.0.72 remote-as 65003

neighbor 172.16.0.72 update-source Loopback0

neighbor 172.16.0.73 peer group evpn

neighbor 172.16.0.73 remote-as 65004

neighbor 172.16.0.73 update-source Loopback0

neighbor 172.16.0.201 peer group evpn

neighbor 172.16.0.201 remote-as 65500

neighbor 172.16.0.201 update-source Loopback0

neighbor 172.16.0.202 peer group evpn

neighbor 172.16.0.202 remote-as 65500

neighbor 172.16.0.202 update-source Loopback0

neighbor 172.16.101.1 peer group evpn

neighbor 172.16.101.1 update-source Vlan100

neighbor 172.16.103.1 peer group evpn

neighbor 172.16.103.1 update-source Vlan300

neighbor 172.16.254.2 peer group SPINE

neighbor 172.16.254.2 update-source Ethernet3

neighbor 172.16.254.14 peer group SPINE

neighbor 172.16.254.14 update-source Ethernet4

neighbor 172.16.254.26 peer group SPINE

neighbor 172.16.254.26 update-source Ethernet5

neighbor 172.16.255.1 remote-as 65001

neighbor 172.16.255.1 update-source Vlan4094

redistribute connected route-map redist-SVI

!

vlan 100

rd 172.16.0.70:100

route-target both 100:1010

redistribute learned

!

vlan 300

rd 172.16.0.70:300

route-target both 300:1030

redistribute learned

!

address-family evpn

no bgp next-hop-unchanged

neighbor evpn activate

!

address-family ipv4

neighbor SPINE activate

no neighbor evpn activate

network 172.16.0.68/32

network 172.16.0.69/32

network 172.16.0.70/32

network 172.16.0.71/32

network 172.16.101.0/24

network 172.16.103.0/24

network 172.16.254.0/31

network 172.16.254.12/31

network 172.16.254.24/31

network 172.16.255.0/30

network 0.0.0.0/0

redistribute connected

!

vrf VXLAN\_VRF

rd 172.16.0.70:1111

route-target import evpn 100:1111

route-target import evpn 300:1111

route-target export evpn 100:1111

router-id 172.16.0.71

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Leaf 3 Configuration

! device: Leaf3 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$VjIyjMhLbn3IWY9U$.XoyK0HvbRuIvzozEJrNcI1tHMzNtxeka7mOdlGWszwmFw5gXYURweNHPUH60.OxE3RFcGp5KK9Ds/V02VwxF0

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname Leaf3

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vlan 200

name MLAG\_L3-L4

!

vlan 300

name MLAG\_HOST

!

vlan 4094

name MLAGPEER

trunk group MLAGPEER

!

vrf instance MGMT

!

vrf instance VXLAN\_VRF

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Port-Channel1

description MLAG link to leaf 4

switchport trunk allowed vlan 200,300,4094

switchport mode trunk

switchport trunk group MLAGPEER

!

interface Port-Channel2

description Mlag link to hosts

switchport trunk allowed vlan 300

switchport mode trunk

mlag 2

!

interface Ethernet1

channel-group 1 mode active

!

interface Ethernet2

channel-group 1 mode active

!

interface Ethernet3

no switchport

ip address 172.16.254.5/31

!

interface Ethernet4

no switchport

ip address 172.16.254.17/31

!

interface Ethernet5

no switchport

ip address 172.16.254.29/31

!

interface Ethernet6

channel-group 2 mode active

!

interface Ethernet7

channel-group 2 mode active

!

interface Loopback0

ip address 172.16.0.72/32

!

interface Management0

vrf MGMT

ip address 172.100.100.106/24

!

interface Vlan200

description VNI for vxlan 1020

ip address 172.16.102.3/24

ip virtual-router address 172.16.102.1

!

interface Vlan300

description VNI for vxlan 1030

ip address 172.16.103.3/24

ip virtual-router address 172.16.103.1

!

interface Vlan4094

ip address 172.16.255.5/30

!

interface Vxlan1

vxlan source-interface Loopback0

vxlan udp-port 8004

vxlan vlan 200 vni 1020

vxlan vlan 300 vni 1030

vxlan vrf VXLAN\_VRF vni 1111

!

ip virtual-router mac-address 00:1c:75:00:00:ff

!

ip routing

no ip routing vrf MGMT

ip routing vrf VXLAN\_VRF

!

ip prefix-list mgmt-route seq 10 deny 172.100.100.0/24 le 32

ip prefix-list mgmt-route seq 20 deny 0.0.0.0/0

ip prefix-list mgmt-route seq 100 permit 0.0.0.0/0 le 32

!

mlag configuration

domain-id MLAG

local-interface Vlan4094

peer-address 172.16.255.6

peer-link Port-Channel1

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

route-map mgmt-route permit 10

match ip address prefix-list mgmt-route

!

router bgp 65003

router-id 172.16.0.72

maximum-paths 6

neighbor SPINE peer group

neighbor SPINE remote-as 65000

neighbor SPINE send-community standard extended

neighbor SPINE maximum-routes 12000

neighbor evpn peer group

neighbor evpn update-source Loopback0

neighbor evpn ebgp-multihop 3

neighbor evpn send-community extended

neighbor 172.16.0.70 peer group evpn

neighbor 172.16.0.70 remote-as 65001

neighbor 172.16.0.71 peer group evpn

neighbor 172.16.0.71 remote-as 65002

neighbor 172.16.0.71 update-source Loopback0

neighbor 172.16.0.73 peer group evpn

neighbor 172.16.0.73 remote-as 65004

neighbor 172.16.0.73 update-source Loopback0

neighbor 172.16.254.4 peer group SPINE

neighbor 172.16.254.4 update-source Ethernet3

neighbor 172.16.254.4 route-map mgmt-route out

neighbor 172.16.254.16 peer group SPINE

neighbor 172.16.254.16 update-source Ethernet4

neighbor 172.16.254.16 route-map mgmt-route out

neighbor 172.16.254.28 peer group SPINE

neighbor 172.16.254.28 update-source Ethernet5

neighbor 172.16.254.28 route-map mgmt-route out

redistribute connected

!

vlan 200

rd 172.16.0.72:200

route-target both 200:1020

redistribute learned

!

vlan 300

rd 172.16.0.72:300

route-target both 300:1030

redistribute learned

!

address-family evpn

bgp next-hop-unchanged

neighbor evpn activate

!

address-family ipv4

neighbor SPINE activate

no neighbor evpn activate

network 172.16.0.0/31

network 172.16.102.0/24

network 172.16.103.0/24

network 172.16.254.0/31

network 172.16.255.0/30

redistribute connected

!

vrf VXLAN\_VRF

rd 172.16.0.72:1111

route-target import evpn 200:1111

route-target import evpn 300:1111

route-target export evpn 200:1111

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Leaf 4 Configuration

! device: Leaf4 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$gIEz1/wul6k/OcC6$HKt1ifD7AswLPyA95N6KYz3CF758vUlrsUCJIlRM0DuJWJE1Gptwq7oihJDG/bszvVxcb3tnDO2eDtceP6caP0

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname Leaf4

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

spanning-tree mst 0 priority 4096

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vlan 200

name MLAG\_L3-L4

!

vlan 300

name MLAG\_To\_Host

!

vlan 4094

name MLAG\_PEER

trunk group MLAGPEER

!

vrf instance MGMT

!

vrf instance VXLAN\_VRF

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Port-Channel1

description MLAG link to leaf 3

switchport trunk allowed vlan 200,300,4094

switchport mode trunk

switchport trunk group MLAGPEER

!

interface Port-Channel2

description Mlag link to hosts

switchport trunk allowed vlan 300

switchport mode trunk

!

interface Ethernet1

channel-group 1 mode active

!

interface Ethernet2

channel-group 1 mode active

!

interface Ethernet3

no switchport

ip address 172.16.254.7/31

!

interface Ethernet4

no switchport

ip address 172.16.254.19/31

!

interface Ethernet5

no switchport

ip address 172.16.254.31/31

!

interface Ethernet6

channel-group 2 mode active

!

interface Ethernet7

channel-group 2 mode active

!

interface Loopback0

ip address 172.16.0.73/32

!

interface Management0

vrf MGMT

ip address 172.100.100.107/24

!

interface Vlan200

ip address 172.16.102.2/24

ip virtual-router address 172.16.102.1

!

interface Vlan300

ip address 172.16.103.4/24

ip virtual-router address 172.16.103.1

!

interface Vlan4094

ip address 172.16.255.6/30

!

interface Vxlan1

vxlan source-interface Loopback0

vxlan udp-port 8004

vxlan vlan 200 vni 1020

vxlan vlan 300 vni 1030

vxlan vrf VXLAN\_VRF vni 1111

!

ip virtual-router mac-address 00:1c:75:00:00:ff

!

ip routing

no ip routing vrf MGMT

ip routing vrf VXLAN\_VRF

!

ip prefix-list mgmt-route seq 10 deny 172.100.100.0/24 le 32

ip prefix-list mgmt-route seq 20 deny 0.0.0.0/0

ip prefix-list mgmt-route seq 100 permit 0.0.0.0/0 le 32

!

mlag configuration

domain-id MLAG

local-interface Vlan4094

peer-address 172.16.255.5

peer-link Port-Channel1

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

route-map mgmt-route permit 10

match ip address prefix-list mgmt-route

!

router bgp 65004

router-id 172.16.0.73

maximum-paths 6

neighbor SPINE peer group

neighbor SPINE remote-as 65000

neighbor SPINE send-community standard extended

neighbor evpn peer group

neighbor evpn update-source Loopback0

neighbor evpn ebgp-multihop 3

neighbor evpn send-community extended

neighbor evpn maximum-routes 0

neighbor 172.16.0.70 peer group evpn

neighbor 172.16.0.70 remote-as 65001

neighbor 172.16.0.71 peer group evpn

neighbor 172.16.0.71 remote-as 65003

neighbor 172.16.0.71 update-source Loopback0

neighbor 172.16.0.73 peer group evpn

neighbor 172.16.0.73 remote-as 65004

neighbor 172.16.0.73 update-source Loopback0

neighbor 172.16.254.4 update-source Ethernet3

neighbor 172.16.254.4 route-map mgmt-route out

neighbor 172.16.254.6 peer group SPINE

neighbor 172.16.254.18 peer group SPINE

neighbor 172.16.254.18 update-source Ethernet4

neighbor 172.16.254.18 route-map mgmt-route out

neighbor 172.16.254.30 peer group SPINE

neighbor 172.16.254.30 update-source Ethernet5

neighbor 172.16.254.30 route-map mgmt-route out

redistribute connected

!

vlan 200

rd 172.16.0.72:200

route-target both 200:1020

redistribute learned

!

vlan 300

rd 172.16.0.72:300

route-target both 300:1030

redistribute learned

!

address-family evpn

bgp next-hop-unchanged

neighbor evpn activate

!

address-family ipv4

neighbor SPINE activate

no neighbor evpn activate

network 172.16.0.0/31

network 172.16.102.0/24

network 172.16.103.0/24

network 172.16.254.0/31

network 172.16.255.0/30

redistribute connected

!

vrf VXLAN\_VRF

rd 172.16.0.72:1111

route-target import evpn 200:1111

route-target import evpn 300:1111

route-target export evpn 200:1111

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Borderleaf 1 Configuration

! device: BORDERLEAF1 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$cATpTLZ1jkLIgyMO$dpTdaatL9K3.LD9Wly.nxtOdzRs5ooQ1uGMoK7I3nfGmIQeDvoxCtRRv9f9kjzBHf91qOnatq/OZAj4pbtXGE0

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname BORDERLEAF1

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vlan 100

name MLAG\_L1-L2

!

vlan 200

name MLAG\_L3-L4

!

vlan 300

name MLAG\_hosts

!

vlan 4094

name MLAG\_Peer\_Link

trunk group MLAGPEER

!

vrf instance MGMT

!

vrf instance VXLAN\_VRF

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Port-Channel1

description MLAG Peer-link to Borderleaf2

switchport trunk allowed vlan 100,200,300,4094

switchport mode trunk

switchport trunk group MLAGPEER

!

interface Ethernet1

description MLAG Peer-link to Borderleaf2

channel-group 1 mode active

!

interface Ethernet2

description MLAG Peer-link to Borderleaf2

channel-group 1 mode active

!

interface Ethernet3

description Connection to Spine1 - eth6

no switchport

ip address 172.16.254.9/31

!

interface Ethernet4

description Connection to Spine2 - eth6

no switchport

ip address 172.16.254.21/31

!

interface Ethernet5

description Connection to Spine3 - eth5

no switchport

ip address 172.16.254.33/31

!

interface Ethernet6

description MLAG Peer-link to Borderleaf2

switchport access vlan 300

!

interface Ethernet7

switchport access vlan 300

!

interface Ethernet8

!

interface Loopback0

description Loopback 0 for VTEP

ip address 172.16.0.201/32

!

interface Loopback1

description Loopback 1 for VTEP

ip address 172.16.1.201/32

!

interface Management0

vrf MGMT

ip address 172.100.100.108/24

!

interface Management1

!

interface Vlan100

description VNI for VLAN 1010

ip address 172.16.101.1/24

!

interface Vlan200

description VNI for VLAN 1020

ip address 172.16.102.1/24

!

interface Vlan300

description VNI for VLAN 1030

ip address 172.16.103.1/24

!

interface Vlan4094

description VNI for vxlan 4094

ip address 172.16.255.9/30

!

interface Vxlan1

vxlan source-interface Loopback0

vxlan udp-port 8004

vxlan vlan 100 vni 1010

vxlan vlan 200 vni 1020

vxlan vlan 300 vni 1030

vxlan vrf VXLAN\_VRF vni 6666

!

ip virtual-router mac-address 00:01:00:02:00:06

!

ip routing

no ip routing vrf MGMT

ip routing vrf VXLAN\_VRF

!

mlag configuration

domain-id MLAG

local-interface Vlan4094

peer-address 172.16.255.10

peer-link Port-Channel1

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

router bgp 65500

router-id 172.16.255.9

maximum-paths 4 ecmp 4

neighbor EVPN peer group

neighbor EVPN update-source Loopback0

neighbor EVPN ebgp-multihop

neighbor EVPN send-community

neighbor EVPN maximum-routes 0

neighbor mlag-peers peer group

neighbor mlag-peers update-source Vlan4094

neighbor spines peer group

neighbor spines remote-as 65000

neighbor spines maximum-routes 12000

neighbor underlay\_ibgp peer group

neighbor 172.16.0.70 peer group EVPN

neighbor 172.16.0.71 peer group EVPN

neighbor 172.16.0.72 peer group EVPN

neighbor 172.16.0.73 peer group EVPN

neighbor 172.16.0.202 peer group EVPN

neighbor 172.16.0.202 remote-as 65500

neighbor 172.16.254.8 peer group spines

neighbor 172.16.254.20 peer group spines

neighbor 172.16.254.32 peer group spines

neighbor 172.16.255.10 peer group mlag-peers

neighbor 172.16.255.10 remote-as 65500

neighbor 172.16.255.10 update-source Vlan4094

neighbor 172.16.255.10 maximum-routes 12000

redistribute connected

redistribute static

!

vlan 100

rd 172.16.0.201:100

route-target both 100:1010

redistribute learned

!

vlan 200

rd 172.16.0.201:200

route-target both 200:1020

redistribute learned

!

vlan 300

rd 172.16.0.201:300

route-target both 300:1030

redistribute learned

!

address-family evpn

bgp next-hop-unchanged

neighbor EVPN activate

no neighbor mlag-peers activate

!

address-family ipv4

no neighbor EVPN activate

neighbor mlag-peers activate

neighbor underlay\_ibgp activate

network 172.16.0.202/32

network 172.16.1.201/32

network 172.16.103.0/24

redistribute connected

!

vrf VXLAN\_VRF

rd 172.16.0.201:6666

route-target import evpn 100:6666

route-target import evpn 200:6666

route-target import evpn 300:6666

route-target export evpn 100:6666

route-target export evpn 200:6666

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end

## Borderleaf 2 Configuration

! device: BORDERLEAF2 (cEOSLab, EOS-4.30.5M-35156751.4305M (engineering build))

!

no aaa root

!

username admin privilege 15 role network-admin secret sha512 $6$eucN5ngreuExDgwS$xnD7T8jO..GBDX0DUlp.hn.W7yW94xTjSanqgaQGBzPIhDAsyAl9N4oScHvOMvf07uVBFI4mKMxwdVEUVKgY/.

username aristacapstone privilege 15 secret sha512 $6$uZ6YJIaJR99Mj29g$nKH4nmzigNKX4N.iUXN.i8c7cAmWGtUyOTTVP3I9ZBjaU4TD5m9RxytimejYnTUuEWk6.ARw829ZMal.ir.Xw/

!

transceiver qsfp default-mode 4x10G

!

service routing protocols model multi-agent

!

hostname BORDERLEAF2

!

spanning-tree mode mstp

no spanning-tree vlan-id 4094

!

system l1

unsupported speed action error

unsupported error-correction action error

!

vlan 100

name MLAG\_L1-L2

!

vlan 200

name MLAG\_L3-L4

!

vlan 300

name MLAG\_hosts

!

vlan 4094

name MLAG\_Peer\_Link

trunk group MLAGPEER

!

vrf instance MGMT

!

vrf instance VXLAN\_VRF

!

management api http-commands

no shutdown

!

vrf MGMT

no shutdown

!

aaa authorization exec default local

!

interface Port-Channel1

description MLAG Peer-link to Borderleaf1

switchport trunk allowed vlan 100,200,300,4094

switchport mode trunk

switchport trunk group MLAGPEER

!

interface Ethernet1

description MLAG Peer-link to Borderleaf1

channel-group 1 mode active

!

interface Ethernet2

description MLAG Peer-link to Borderleaf1

channel-group 1 mode active

!

interface Ethernet3

description Connection to Spine1 - eth7

no switchport

ip address 172.16.254.11/31

!

interface Ethernet4

description Connection to Spine2 - eth7

no switchport

ip address 172.16.254.23/31

!

interface Ethernet5

description Connection to Spine3 - eth6

no switchport

ip address 172.16.254.35/31

!

interface Ethernet6

description MLAG Peer-link to Borderleaf1

switchport access vlan 300

!

interface Ethernet7

switchport access vlan 300

!

interface Ethernet8

!

interface Loopback0

description Loopback 0 for VTEP

ip address 172.16.0.202/32

!

interface Loopback1

description Loopback 1 for VTEP

ip address 172.16.1.201/32

!

interface Management0

vrf MGMT

ip address 172.100.100.110/24

!

interface Management1

!

interface Vlan100

description VNI for VLAN 1010

ip address 172.16.101.1/24

!

interface Vlan200

description VNI for vlan 1020

ip address 172.16.102.1/24

!

interface Vlan300

description VNI for vlan 1030

ip address 172.16.103.1/24

!

interface Vlan4094

description VNI for vxlan 4094

ip address 172.16.255.10/30

!

interface Vxlan1

vxlan source-interface Loopback0

vxlan udp-port 8004

vxlan vlan 100 vni 1010

vxlan vlan 200 vni 1020

vxlan vlan 300 vni 1030

vxlan vrf VXLAN\_VRF vni 6666

!

ip virtual-router mac-address 00:01:00:02:00:06

!

ip routing

no ip routing vrf MGMT

ip routing vrf VXLAN\_VRF

!

mlag configuration

domain-id MLAG

local-interface Vlan4094

peer-address 172.16.255.9

peer-link Port-Channel1

!

ip route vrf MGMT 0.0.0.0/0 172.100.100.1

!

router bgp 65500

router-id 172.16.255.10

maximum-paths 4 ecmp 4

neighbor EVPN peer group

neighbor EVPN update-source Loopback0

neighbor EVPN ebgp-multihop

neighbor EVPN send-community

neighbor EVPN maximum-routes 0

neighbor mlag-peers peer group

neighbor mlag-peers update-source Vlan4094

neighbor spines peer group

neighbor spines remote-as 65000

neighbor spines maximum-routes 12000

neighbor underlay\_ibgp peer group

neighbor 172.16.0.70 peer group EVPN

neighbor 172.16.0.71 peer group EVPN

neighbor 172.16.0.72 peer group EVPN

neighbor 172.16.0.73 peer group EVPN

neighbor 172.16.0.201 peer group EVPN

neighbor 172.16.0.201 remote-as 65500

neighbor 172.16.254.10 peer group spines

neighbor 172.16.254.22 peer group spines

neighbor 172.16.254.34 peer group spines

neighbor 172.16.255.9 peer group mlag-peers

neighbor 172.16.255.9 remote-as 65500

neighbor 172.16.255.9 update-source Vlan4094

neighbor 172.16.255.9 maximum-routes 12000

redistribute connected

redistribute static

!

vlan 100

rd 172.16.0.202:100

route-target both 100:1010

redistribute learned

!

vlan 200

rd 172.16.0.202:200

route-target both 200:1020

redistribute learned

!

vlan 300

rd 172.16.0.202:300

route-target both 300:1030

redistribute learned

!

address-family evpn

bgp next-hop-unchanged

neighbor EVPN activate

no neighbor mlag-peers activate

!

address-family ipv4

no neighbor EVPN activate

neighbor mlag-peers activate

neighbor underlay\_ibgp activate

network 172.16.0.202/32

network 172.16.1.201/32

network 172.16.103.0/24

redistribute connected

!

vrf VXLAN\_VRF

rd 172.16.0.202:6666

route-target import evpn 100:6666

route-target import evpn 200:6666

route-target import evpn 300:6666

route-target export evpn 100:6666

route-target export evpn 200:6666

redistribute connected

!

management ssh

vrf MGMT

no shutdown

!

end