## BGP Routing Design and Implementation

This is the documentation for our BGP routing design and how it was implemented using Ansible and the dynamic inventory from GraphQL.

## Diagram and Subnets

A computer network diagram with text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Device** | **ASN** | **Interface** | **IP Address** |
| SWITCH1 | 100 | Loopback1 | 192.168.1.1/24 |
| Loopback2 | 192.168.2.1/24 |
| Loopback3 | 192.168.3.1/24 |
| Ethernet1 | 192.168.10.1/30 |
| Ethernet2 | 192.168.11.1/30 |
| SWITCH2 | 200 | Loopback4 | 192.168.4.1/24 |
| Loopback5 | 192.168.5.1/24 |
| Loopback6 | 192.168.6.1/24 |
| Ethernet1 | 192.168.10.2/30 |
| Ethernet2 | 192.168.12.1/30 |
| SWITCH3 | 300 | Loopback7 | 192.168.7.1/24 |
| Loopback8 | 192.168.8.1/24 |
| Loopback9 | 192.168.9.1/24 |
| Ethernet1 | 192.168.11.2/30 |
| Ethernet2 | 192.168.12.2/30 |

## What is BGP

We have chosen Border Gateway Protocol (BGP) as our preferred routing protocol due to its advantageous features and compatibility with our telemetry solution. BGP is an Exterior Gateway Protocol (EGP) commonly used by Internet Service Providers (ISPs) and other WAN providers to exchange data between different autonomous systems (AS). Being a path-vector protocol, BGP allows routing decisions to be made based on a list of path attributes such as local preference, MED, route origin, and AS path length. This provides network administrators with more precise control over how traffic is forwarded over a WAN, since the shortest path may not always be the best route to a given destination for an ISP sending data across the Internet.

## Rationale for Choosing BGP

Our team has concluded that OSPF is the ideal routing protocol for our small network due to its ease of management and optimal complexity level for a LAN such as this one. However, we encountered difficulties with our telemetry solution as it was unable to detect OSPF neighbors. To ensure compatibility, we opted to switch to BGP, although it requires more precise control over network traffic and peer establishment. This decision successfully resolved the issues we were experiencing with our telemetry solution.

## Explanation of Design

In the network topology that we have designed in ContainerLab, each switch will be allocated its own autonomous system number (ASN). This will simulate multiple networks within each AS. To achieve this, we will deploy several loopback interfaces on each switch, with each interface assigned to a unique subnet. This will allow for efficient communication and management of the various network segments via BGP. Additionally, we must create new interfaces for our intra-switch links in ContainerLab and assign IP addresses to them so that the devices can ping each other. This is necessary since the switches are unaware of our management IP addresses. For routing to work, we so needed to remove EtherChannel port groups from the internal ethernet interfaces.

## Implementation and Security (step by step) ansible

In this section, we will walk through how to implement BGP routing on our Arista devices using Ansible, which will pull our devices from our GraphQL dynamic inventory file creates from our Nautobot Source of Truth (SoT)

1. At the desktop, open Visual Studio Code by going to Applications > Development > Visual Studio Code from the top menu.

A screenshot of a computer

Description automatically generated

1. In the Terminal, change into the ansible-gql folder on the Desktop, which contains our dynamic inventory and Ansible playbooks.



1. Create a new file for our BGP playbook.



1. Copy and paste the reference Ansible playbook code into the BGP playbook file in Visual Studio Code. This playbook consists of three plays and each play performs the following tasks on each switch in our topology:
   1. Create Ethernet1 and Ethernet2 internal interfaces for inter-switch communications.
   2. Configure IP addresses for the Loopback, Ethernet1, and Ethernet2 interfaces.
   3. Establish BGP peer sessions between directly connected routers with MD5 peer authentication on each neighbor.
   4. Advertise the networks residing on each switch into the switch’s BGP routing table.
2. Save the playbook before running the playbook using CTRL+S.
3. Run the playbook in the Terminal window using the -i option to specify the dynamic inventory configurationfile in the current directory. The reference playbook should execute successfully as shown below.



A black screen with white text

Description automatically generated

## Verification

In this section, we will review helpful EOS CLI commands to verify BGP routing and security.

1. On the desktop-linux-1-vm instance, open a web browser by going to Applications > Development > Visual Studio Code from the top menu.

A screenshot of a computer menu

Description automatically generated

1. From Google Cloud on the host computer, enter the internal IP address of the ntca-capstone-2024-container-lab instance into the web browser on the desktop-linux-1-vm instance. This IP address leads to our ContainerLab Graphite GUI.

A screenshot of a computer

Description automatically generated

1. Access the CLI of one of the switches by clicking on the device and clicking on SSH next to the IPv4 option.

A screenshot of a phone

Description automatically generated

1. Enter **admin** for the username and password. Click Sign in to access the CLI of the switch.

A screenshot of a computer

Description automatically generated

1. To view a summary of BGP peers along with their adjacency states, type the command **show ip bgp summary** on the switch. This command provides crucial information such as the remote ASN of the neighbor, the status of the BGP adjacency, and the prefixes that are being advertised to each neighbor.

A black background with white text

Description automatically generated

1. To obtain more comprehensive BGP routing information, you can use the **show ip bgp** command to display the BGP routing table. This table contains detailed path metrics for each destination, including the local preference, AS path length, and path origin. This information is particularly useful for administrators who need to perform BGP traffic engineering.

A screenshot of a computer screen

Description automatically generated

1. To display the current BGP routes in the switch’s routing table, use the command **show ip route bgp**. Each switch should receive six eBGP routes among its two peers.

A computer screen with white text and red line

Description automatically generated

1. Finally, to verify that BGP MD5 peer authentication is enabled, use the **show bgp neighbors | grep auth** command. In the full output of the command, this information is shown near the end of each peer’s section.

**show bgp neighbors:**

A screenshot of a computer

Description automatically generated

A red line with white text

Description automatically generated

## Reference BGP Ansible Playbook

- name: Configure BGP on SWITCH1

hosts: SWITCH1

gather\_facts: false

connection: network\_cli

vars:

ansible\_connection: network\_cli

ansible\_network\_os: eos

ansible\_user: admin

ansible\_password: admin

ansible\_become: true

ansible\_become\_method: enable

ansible\_become\_password: admin

ansible\_python\_interpreter: /usr/bin/python3

tasks:

- name: Create Ethernet1 and Ethernet2 Internal Interfaces

arista.eos.eos\_interfaces:

config:

- name: Ethernet1

enabled: true

mode: layer3

- name: Ethernet2

enabled: true

mode: layer3

- name: Configure Loopback Interfaces 1-3, Ethernet 1-2 IP Addresses on SWITCH1

arista.eos.eos\_l3\_interfaces:

config:

- name: Loopback1

ipv4:

- address: 192.168.1.1/24

- name: Loopback2

ipv4:

- address: 192.168.2.1/24

- name: Loopback3

ipv4:

- address: 192.168.3.1/24

- name: Ethernet1

ipv4:

- address: 192.168.10.1/30

- name: Ethernet2

ipv4:

- address: 192.168.11.1/30

- name: Establish BGP Peers on SWITCH1

arista.eos.eos\_bgp\_global:

config:

as\_number: '100'

router\_id: 1.1.1.1

neighbors:

- neighbor\_address: 192.168.10.2

remote\_as: 200

description: Peer with SWITCH2

encryption\_password:

password: '@Stout2024'

type: 0

- neighbor\_address: 192.168.11.2

remote\_as: 300

description: Peer with SWITCH3

encryption\_password:

password: '@Stout2024'

type: 0

- name: Advertise Networks on SWITCH1

arista.eos.eos\_bgp\_global:

config:

as\_number: '100'

networks:

- address: 192.168.1.0/24

- address: 192.168.2.0/24

- address: 192.168.3.0/24

- address: 192.168.10.0/30

- address: 192.168.11.0/30

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

- name: Configure BGP on SWITCH2

hosts: SWITCH2

gather\_facts: false

connection: network\_cli

vars:

ansible\_connection: network\_cli

ansible\_network\_os: eos

ansible\_user: admin

ansible\_password: admin

ansible\_become: true

ansible\_become\_method: enable

ansible\_become\_password: admin

ansible\_python\_interpreter: /usr/bin/python3

tasks:

- name: Create Ethernet1 and Ethernet2 Internal Interfaces

arista.eos.eos\_interfaces:

config:

- name: Ethernet1

enabled: true

mode: layer3

- name: Ethernet2

enabled: true

mode: layer3

- name: Configure Loopback Interfaces 4-6, Ethernet 1-2 IP Addresses on SWITCH2

arista.eos.eos\_l3\_interfaces:

config:

- name: Loopback4

ipv4:

- address: 192.168.4.1/24

- name: Loopback5

ipv4:

- address: 192.168.5.1/24

- name: Loopback6

ipv4:

- address: 192.168.6.1/24

- name: Ethernet1

ipv4:

- address: 192.168.10.2/30

- name: Ethernet2

ipv4:

- address: 192.168.12.1/30

- name: Establish BGP Peers on SWITCH2

arista.eos.eos\_bgp\_global:

config:

as\_number: '200'

router\_id: 2.2.2.2

neighbors:

- neighbor\_address: 192.168.10.1

remote\_as: 100

description: Peer with SWITCH1

encryption\_password:

password: '@Stout2024'

type: 0

- neighbor\_address: 192.168.12.2

remote\_as: 300

description: Peer with SWITCH3

encryption\_password:

password: '@Stout2024'

type: 0

- name: Advertise Networks on SWITCH2

arista.eos.eos\_bgp\_global:

config:

as\_number: '200'

networks:

- address: 192.168.4.0/24

- address: 192.168.5.0/24

- address: 192.168.6.0/24

- address: 192.168.10.0/30

- address: 192.168.12.0/30

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

- name: Configure BGP on SWITCH3

hosts: SWITCH3

gather\_facts: false

connection: network\_cli

vars:

ansible\_connection: network\_cli

ansible\_network\_os: eos

ansible\_user: admin

ansible\_password: admin

ansible\_become: true

ansible\_become\_method: enable

ansible\_become\_password: admin

ansible\_python\_interpreter: /usr/bin/python3

tasks:

- name: Create Ethernet1 and Ethernet2 Internal Interfaces

arista.eos.eos\_interfaces:

config:

- name: Ethernet1

enabled: true

mode: layer3

- name: Ethernet2

enabled: true

mode: layer3

- name: Configure Loopback Interfaces 7-9, Ethernet 1-2 IP Addresses on SWITCH3

arista.eos.eos\_l3\_interfaces:

config:

- name: Loopback7

ipv4:

- address: 192.168.7.1/24

- name: Loopback8

ipv4:

- address: 192.168.8.1/24

- name: Loopback9

ipv4:

- address: 192.168.9.1/24

- name: Ethernet1

ipv4:

- address: 192.168.11.2/30

- name: Ethernet2

ipv4:

- address: 192.168.12.2/30

- name: Establish BGP Peers on SWITCH3

arista.eos.eos\_bgp\_global:

config:

as\_number: '300'

router\_id: 3.3.3.3

neighbors:

- neighbor\_address: 192.168.11.1

remote\_as: 100

description: Peer with SWITCH1

encryption\_password:

password: '@Stout2024'

type: 0

- neighbor\_address: 192.168.12.1

remote\_as: 200

description: Peer with SWITCH2

encryption\_password:

password: '@Stout2024'

type: 0

- name: Advertise Networks on SWITCH3

arista.eos.eos\_bgp\_global:

config:

as\_number: '300'

networks:

- address: 192.168.7.0/24

- address: 192.168.8.0/24

- address: 192.168.9.0/24

- address: 192.168.11.0/30

- address: 192.168.12.0/30