Dell EMC Networking Ansible Integration Documentation

Release 2.0

Dell EMC Networking Team

Jan 07, 2020

Table of Contents

CHAPTER 1

Introduction

This information explains Ansible and the Dell EMC Ansible integration.

1.1 Ansible

Ansible is a simple agentless automation framework. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates. Ansible supports network automation as part of its core framework.

See Integration: Network Automation with Ansible for more information.

1.2 Dell EMC Ansible integration

Starting with Ansible 2.3, the Ansible core supports Dell EMC modules. You can use these to manage and automate your Dell EMC switches running OS6, OS9, and OS10. All modules are currently executed in local connection mode, using CLI and SSH transport.

See Integration: Ansible and Dell EMC Networking for more information.

CHAPTER 2

Installation

You can install Ansible roles on the control machine using Dell EMC devices.

2.1 Ansible modules

Dell EMC Ansible modules for dellos6, dellos9, and dellos10 are part of the Ansible core. Install Ansible 2.3 or later to use these modules. To use OpenSwitch Ansible "opx_cps" module, install Ansible 2.7 or later. See Ansible documentation for more information.

2.2 Ansible roles

Install all Dell EMC Ansible roles.

```
ansible-galaxy install -r dellemc_roles.txt
```

where `dellemc_roles.txt` is defined as:

```
Dell-Networking.dellos-aaa
Dell-Networking.dellos-acl
Dell-Networking.dellos-bgp
Dell-Networking.dellos-copy-config
Dell-Networking.dellos-dcb
Dell-Networking.dellos-dns
Dell-Networking.dellos-ecmp
Dell-Networking.dellos-flow-monitor
Dell-Networking.dellos-image-upgrade
Dell-Networking.dellos-interface
Dell-Networking.dellos-lag
Dell-Networking.dellos-lldp
Dell-Networking.dellos-logging
Dell-Networking.dellos-ntp
```

(continues on next page)

(continued from previous page)

```
Dell-Networking.dellos-prefix-list
Dell-Networking.dellos-qos
Dell-Networking.dellos-route-map
Dell-Networking.dellos-sflow
Dell-Networking.dellos-snmp
Dell-Networking.dellos-system
Dell-Networking.dellos-users
Dell-Networking.dellos-vlan
Dell-Networking.dellos-vlt
Dell-Networking.dellos-vrf
Dell-Networking.dellos-vrrp
Dell-Networking.dellos-xstp
```

You can also install an individual Dell EMC Networking Ansible role using a single command. For example, to install the AAA role use `ansible-galaxy install Dell-Networking.dellos.aaa`.

See Ansible Galaxy for more information on Dell EMC Ansible roles.

2.3 Dell EMC devices

Dell EMC devices require minimal configuration to run Ansible playbooks.

2.3.1 OS6

1. Create a username and password for Ansible.
2. Configure the Management interface (static/dynamic IP address).
3. Enable the SSH server.

```
console(config)# username admin password ansible@123
console(config)# enable password ansible@123
console(config)# interface  out-of-band
console(conf-if)# ip address 10.16.148.79 255.255.255.0 10.16.148.254
console(conf-if)# exit
console(config)# ip ssh  server
```

2.3.2 OS9

1. Create a username and password for Ansible.
2. Configure the Management interface (static/dynamic IP address).
3. Enable the SSH server.
4. Set the maximum connection rate limit.

```
Dell(config)# username ansible password ansible
Dell(config)# enable password ansible
Dell(config)# interface managementethernet 0/0
Dell(conf-if-ma-0/0)# ip add 10.16.148.72/24
Dell(conf-if-ma-0/0)# no shutdown
Dell(conf-if-ma-0/0)# exit
```

(continues on next page)

(continued from previous page)

```
Dell(config)# ip ssh server enable
Dell(config)# ip ssh connection-rate-limit 60
```

2.3.3 OS10

1. Create an Ansible username and password.
2. Configure the Management interface (static/dynamic IP address).

```
OS10# config t
OS10(config)# username ansible password ansible
OS10(config)# interface mgmt 1/1/1
OS10(conf-if-ma-1/1/1)# ip address 10.16.149.62/16
OS10(conf-if-ma-1/1/1)# no shutdown
OS10(conf-if-ma-1/1/1)# do commit
OS10(conf-if-ma-1/1/1)# exit
```

> **NOTE**: SSH is enabled in OS10 by default.

2.3.4 OPX

1. Create an Ansible username and password.
2. Configure the Management interface (static/dynamic IP address).

```
root@os10:/config/home/linuxadmin# useradd testuser
root@os10:/config/home/linuxadmin# passwd testuser
New password:
Retype new password:
passwd: password updated successfully
root@os10:/config/home/linuxadmin# ifconfig eth0 10.16.148.123 netmask 255.255.255.0␣
↪up
root@os10:/config/home/linuxadmin# route default gw 10.16.148.254
```

CHAPTER 3

Dell EMC Ansible modules

Note: Ansible has deprecated support for the template module (see Deprecations).

3.1 OS6 modules

- dellos6_command: Run commands on remote devices running Dell EMC OS6
- dellos6_config: Manage configuration sections on remote devices running Dell EMC OS6
- dellos6_facts: Collect facts from remote devices running Dell EMC OS6

3.2 OS9 modules

- dellos9_command: Run commands on remote devices running Dell EMC OS9
- dellos9_config: Manage configuration sections on remote devices running Dell EMC OS9
- dellos9_facts: Collect facts from remote devices running Dell EMC OS9

3.3 OS10 modules

- dellos10_command: Run commands on remote devices running Dell EMC SmartFabric OS10
- dellos10_config: Manage configuration sections on remote devices running Dell EMC SmartFabric OS10
- dellos10_facts: Collect facts from remote devices running Dell EMC SmartFabric OS10

CHAPTER 4

Dell EMC Ansible roles

The Dell EMC Ansible roles facilitate device provisioning running Dell EMC OS6, OS9, or OS10. This information describes the Dell EMC Ansible roles.

4.1 AAA role

The dellos-aaa role facilitates the configuration of authentication authorization acccounting (AAA), and supports the configuration of TACACS and RADIUS server and AAA.

Abstracted for `OS6` `OS10`

4.2 ACL role

The dellos-acl role facilitates the configuration of an access control list (ACL). It supports the configuration of different types of ACLs (standard and extended) for both IPv4 and IPv6, and assigns the access-class to line terminals.

Abstracted for `OS6` `OS9`

4.3 BGP role

The dellos-bgp role facilitates the configuration of border gateway protocol (BGP) attributes, and supports router ID, networks, neighbors, and maximum path configurations.

Abstracted for `OS6` `OS9` `OS10`

4.4 Copy-config role

The dellos-copy-config role pushes the backup running configuration into a device. This role merges the configuration in the template file with the running configuration of the Dell EMC Networking device.

Abstracted for `OS6` `OS9` `OS10`

4.5 DCB role

The dellos-dcb role facilitates the configuration of data center bridging (DCB), supports the configuration of DCB map and DCB buffer and assigns them to interfaces.

Abstracted for `OS9`

4.6 DNS role

The dellos-dns role facilitates the configuration of domain name service (DNS).

Abstracted for `OS9`

4.7 ECMP role

The dellos-ecmp role facilitates the configuration of equal cost multi-path (ECMP). It supports the configuration of ECMP for IPv4.

Abstracted for `OS9`

4.8 Flow-monitor role

The dellos-flow-monitor role facilitates the configuration of ACL flow-based monitoring attributes. Flow-based mirroring is a mirroring session in which traffic matches specified policies that are mirrored to a destination port. Port-based mirroring maintains a database that contains all monitoring sessions, including port monitor sessions.

Abstracted for `OS10`

4.9 Image-upgrade role

The dellos-image-upgrade role facilitates upgrades or installation of an OS10 software image.

Abstracted for `OS6` `OS9` `OS10`

4.10 Interface role

The dellos-interface role facilitates the configuration of interface attributes. It supports the configuration of administrative state, description, MTU, IP address, IP helper, and port mode.

Abstracted for `OS10`

4.11 LAG role

The dellos-lag role facilitates the configuration of link aggregation group (LAG) attributes. This role supports the creation and deletion of a LAG and its member ports, and supports the configuration of type (static/dynamic), hash scheme, and minimum required link.

Abstracted for `OS6` `OS9` `OS10`

4.12 LLDP role

The dellos-lldp role facilitates the configuration of link layer discovery protocol (LLDP) attributes at global and interface level. This role supports the configuration of hello, mode, multiplier, advertise tlvs, management interface, fcoe, iscsi at global and interface levels.

Abstracted for `OS6` `OS9` `OS10`

4.13 Logging role

The dellos-logging role facilitates the configuration of global logging attributes, and supports the configuration of logging servers.

Abstracted for `OS6` `OS9` `OS10`

4.14 NTP role

The dellos-ntp role facilitates the configuration of network time protocol attributes.

Abstracted for `OS6` `OS9` `OS10`

4.15 Prefix-list role

The dellos-prefix-list role facilitates the configuration of a prefix-list, supports the configuration of IP prefix-list, and assigns the prefix-list to line terminals.

Abstracted for `OS9`

4.16 QoS role

The dellos-qos role facilitates the configuration of quality of service attributes including policy-map and class-map.

Abstracted for `OS6` `OS10`

4.17 Route-map role

The dellos-route-map role facilitates the configuration of route-map attributes.

Abstracted for `OS10`

4.18 sFlow role

The dellos-sflow role facilitates the configuration of global and interface-level sflow attributes. This role supports the configuration of sflow collectors at the global level, enabling and disabling of sFlow and specification of sFlow polling-interval, sample-rate, max-datagram sizes, and so on are supported at interface and global levels.

Abstracted for `OS9`

4.19 SNMP role

The dellos-snmp role facilitates the configuration of global snmp attributes. It supports the configuration of SNMP server attributes like users, group, community, location, traps, and so on.

Abstracted for `OS9` `OS10`

4.20 System role

The dellos-system role facilitates the configuration of global system attributes. This role specifically enables configuration of hostname, NTP server, and enables the password for dellos6, dellos9, and dellos10. dellos9 supports the configuration of the management route, hash alogrithm, clock, line terminal, banner and reload type.

Abstracted for `OS6` `OS9` `OS10`

4.21 Users role

The dellos-users role facilitates the configuration of global system user attributes. This role supports the configuration of CLI users.

Abstracted for `OS6` `OS9` `OS10`

4.22 VLAN role

The dellos-vlan role facilitates configuring virtual LAN (VLAN) attributes. This role supports the creation and deletion of a VLAN and its member ports.

Abstracted for `OS6` `OS9` `OS10`

4.23 VLT role

The dellos-vlt role facilitates the configuration of the basics of virtual link trunking (VLT) to provide a loop-free topology.

Abstracted for `OS9` `OS10`

4.24 VRF role

The dellos-vrf role facilitates the configuration of basic virtual routing and forwarding (VRF) that helps in the partition of physical routers to multiple virtual routers.

Abstracted for `OS9`

4.25 VRRP role

The dellos-vrrp role facilitates configuration of virtual router redundancy protocol (VRRP) attributes. This role supports the creation of VRRP groups for interfaces, and setting the VRRP group attributes.

Abstracted for `OS6` `OS9` `OS10`

4.26 xSTP role

The dellos-xstp role facilitates the configuration of xSTP attributes. This role supports multiple version of spanning-tree protocol (STP), rapid spanning-tree (RSTP) protocol, multiple spanning-tree (MST), and per-VLAN spanning-tree (PVST). This role supports the configuration of bridge priority, enabling and disabling spanning-tree, creating and deleting instances, and mapping virtual LAN (VLAN) to instances.

Abstracted for `OS6` `OS9` `OS10`

4.27 VXLAN role

The dellos-vxlan role facilitates the configuration of virtual extensible LAN (VXLAN) attributes. It supports the configuration of virtual networks, Ethernet virtual private network (EVPN), and network virtualization edge (NVE).

Abstracted for `OS10`

4.28 BFD role

The dellos-bfd This role facilitates the configuration of BFD global attributes, and is abstracted for dellos10. It specifically enables configuration of BFD interval , min_rx, multiplier, and role.

Abstracted for `OS10`

4.29 TEMPLATE role

The dellos-template This role facilitates the TEXTFSM parsing engine. TextFSM is a template based state machine . It takes the raw string input from the CLI of network devices dellos10 , run them through a TEXTFSM template and return structured text in the form of a Python dictionary.

Abstracted for `OS10`

4.30 UPLINK role

The dellos-uplink This role facilitates the configuration of uplink attributes, and is abstracted for dellos10. It specifically enables configuration of association between upstream and downstream interfaces known as uplink-state group.

Abstracted for `OS10`

4.31 Fabric-Summary role

The dellos_fabric_summary This role facilitates to get show system information of all the switches in the fabric.

Abstracted for `OS10`

4.32 Network-Validation role

The dellos_network_validation This role facilitates to verify the Networks. It validates networking features of wiring connection, BGP neighbors, MTU between neighbors and VLT pair.

Abstracted for `OS10`

CHAPTER 5

Support matrix of Dell EMC Ansible roles

This table shows the support matrix between Ansible roles and Dell EMC OS6, OS9, and OS10.

Role	OS6	OS9	OS10
dellos-aaa	No	Yes	Yes
dellos-acl	No	Yes	Yes
dellos-bgp	Yes	Yes	Yes
dellos-copy-config	No	Yes	Yes
dellos-dcb	No	Yes	Yes
dellos-dns	No	Yes	Yes
dellos-ecmp	No	Yes	Yes
dellos-flow-monitor	No	No	Yes
dellos-image-upgrade	No	No	Yes
dellos-interface	Yes	Yes	Yes
dellos-lag	Yes	Yes	Yes
dellos-lldp	No	Yes	Yes
dellos-logging	Yes	Yes	Yes
dellos-ntp	Yes	Yes	Yes
dellos-prefix-list	No	Yes	Yes
dellos-qos	No	No	Yes
dellos-route-map	No	No	Yes
dellos-sflow	No	Yes	Yes
dellos-snmp	Yes	Yes	Yes
dellos-system	Yes	Yes	Yes
dellos-users	Yes	Yes	Yes
dellos-vlan	Yes	Yes	Yes
dellos-vlt	No	Yes	Yes
dellos-vrf	No	Yes	Yes
dellos-vrrp	Yes	Yes	Yes
dellos-xstp	Yes	Yes	Yes

CHAPTER 6

Dell EMC Ansible module examples

These module examples explain how to create a simple Ansible playbook, run the Dell EMC Ansible modules, then configure a switch using Ansible roles.

6.1 Create simple Ansible playbook

Step 1

Create an inventory file called `inventory.yaml`, then specify the IP address.

```
spine1 ansible_host=10.11.182.16
```

Step 2

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and transport.

```
hostname: spine1
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_become_method: enable
ansible_become: yes
ansible_become_pass: xxxxx
ansible_network_os: xxxxx
```

Step 3

Create a playbook called `showver.yaml`.

```
hosts: spine1
connection: network_cli
gather_facts: no

tasks:
- name: "Get Dell EMC OS9 Show version"
```

(continues on next page)

(continued from previous page)

```
  dellos9_command:
    commands: ['show version']
  register: show_ver

- debug: var=show_ver
```

Step 4

Run the playbook.

```
ansible-playbook -i inventory.yaml showver.yaml
```

6.2 Create simple Ansible playbook using connection="netconf"

Step 1

Create an inventory file called `inventory.yaml`, then specify the IP address.

```
spine1 ansible_host=10.11.182.16
```

Step 2

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and transport.

```
hostname: spine1
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
```

Step 3

Create a playbook called `create_vlan.yaml`.

```
hosts: spine1
connection: netconf
gather_facts: no

tasks:
- name: "Create a vlan entry"
  netconf_config:
  host: 10.16.138.15
  username: admin
  password: admin
  hostkey_verify: false
  xml: |
      <config>
         <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"␣
↪xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type" xmlns:dell-if="http://www.
↪dellemc.com/networking/os10/dell-interface" xmlns:dell-eth="http://www.dellemc.com/
↪networking/os10/dell-ethernet" xmlns:dell-lag="http://www.dellemc.com/networking/
↪os10/dell-lag" xmlns:dell-lacp="http://www.dellemc.com/networking/os10/dell-lacp">
           <interface>
             <type>ianaift:l2vlan</type>
             <name>vlan106</name>
           </interface>
```

(continues on next page)

(continued from previous page)

```
        </interfaces>
      </config>
```

Step 4

Run the playbook.

```
ansible-playbook -i inventory.yaml create_vlan.yaml
```

6.3 Run Dell EMC Ansible examples

Use these sample Ansible playbooks to understand how to use Dell EMC Ansible modules.

6.3.1 Installation and setup

1. Install Ansible.
2. Clone the Ansible-dellos-examples repository in the control machine.
3. Update the `inventory.yaml` file to configure the device IP.
4. Update the corresponding host variables; use `hosts_var/dellos10_sw1.yaml` for device credentials.

OS6

dellos6_facts module that collects the facts from the OS6 device example.

```
ansible-playbook -i inventory.yaml getfacts_os6.yaml
```

dellos6_command module that executes the show version command example.

```
ansible-playbook -i inventory.yaml showver_os6.yaml
```

dellos6_config module that configures the hostname on the OS6 device example.

```
ansible-playbook -vvv -i inventory.yaml hostname_os6.yaml
```

OS9

dellos9_facts module that collects the facts from the OS9 device example.

```
ansible-playbook -i inventory.yaml getfacts_os9.yaml
```

dellos9_command module that executes the show version command example.

```
ansible-playbook -i inventory.yaml showver_os9.yaml
```

dellos9_config module that configures the hostname on the OS9 device example.

```
ansible-playbook -vvv -i inventory.yaml hostname_os9.yaml
```

OS10

dellos10_facts module that collects the facts from the OS10 device example.

```
ansible-playbook -i inventory.yaml getfacts_os10.yaml
```

dellos10_command module that executes the show version command example.

```
ansible-playbook -i inventory.yaml showver_os10.yaml
```

dellos10_config module that configures the hostname on the OS10 device example.

```
ansible-playbook -vvv -i inventory.yaml hostname_os10.yaml
```

6.4 Run Ansible example

Use this example to configure VLAN using CPS operations.

Step 1

Create an inventory file called `inventory.yaml`, then specify the IP address.

```
spine1 ansible_host=10.11.182.16
```

Step 2

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and transport.

```
hostname: spine1
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
```

Step 3

Create a file called ‘‘create_vlan.yaml‘, then define the CPS operations.

```
- hosts: opx_cps
  tasks:
    - name: Create vlan
      opx_cps:
        module_name: "dell-base-if-cmn/if/interfaces/interface"
        attr_data: "{{ attr_vlan }}"
        operation: "create"
      environment:
        PYTHONPATH: "/usr/lib/opx:/usr/lib/x86_64-linux-gnu/opx"
        LD_LIBRARY_PATH: "/usr/lib/opx:/lib/x86_64-linux-gnu:/usr/lib/x86_64-linux-
↪gnu:/usr/lib:/lib"
```

Step 4

Run the playbook.

```
ansible-playbook -i inventory.yaml create_vlan.yaml
```

6.5 Playbook using Ansible roles example

Use these examples to configure the switch using Ansible roles.

Step 1

Create an inventory file called `inventory.yaml`, then specify the device IP address.

```
spine1 ansible_host= <ip_address>
```

Step 2

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and transport.

```
---
hostname: dellos9

ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_become: yes
ansible_become_method: enable
ansible_become_pass: xxxxx
ansible_network_os: dellos9


dellos_interface:
        fortyGigE 0/32:
          desc: "Connected to Spine1"
          portmode:
          switchport: False
          mtu: 2500
          admin: up
          ipv6_and_mask: 2001:4898:5808:ffa2::5/126
          suppress_ra : present
          ip_type_dynamic: true
          ip_and_mask: 192.168.23.22/24
          class_vendor_identifier: present
          option82: true
          remote_id: hostname
        fortyGigE 0/20:
          portmode:
          switchport: False
        fortyGigE 0/64:
          portmode:
          switchport: True
        fortyGigE 0/60:
          portmode:
          switchport: True
        fortyGigE 0/12:
          portmode:
          switchport: True
        loopback 0:
          ip_and_mask: 1.1.1.1/32
          admin: up
        Port-channel 12:
          switchport: True
dellos_vlan:
        vlan 100:
```

(continues on next page)

(continued from previous page)

```
          name: "Mgmt Network"
          description: "Int-vlan"
          tagged_members:
                - port: fortyGigE 0/60
                  state: present
          untagged_members:
                - port: fortyGigE 0/12
                  state: present
          state: present
```

Step 3

Create a playbook called `switch_config.yaml`.

```
---
- hosts: dellos9
  gather_facts: no
  connection: network_cli
  roles:
        - Dell-Networking.dellos-interface
        - Dell-Networking.dellos-vlan
```

Step 4

Run the playbook.

```
ansible-playbook -i inventory.yaml switch_config.yaml
```

CHAPTER 7

Provision CLOS fabric using Dell EMC Ansible modules example

This example describes how to use Ansible to build a CLOS fabric with Dell EMC SmartFabric OS10 switches. The sample topology is a two-tier CLOS fabric with two spines and four leafs connected as mesh. eBGP is running between the two tiers. All switches in spine have the same AS number, and each leaf switch has a unique AS number. All AS numbers used are private.

For application load-balancing purposes, the same prefix is advertised from multiple leaf switches and uses *BGP multipath relax* feature.

7.1 Create a simple Ansible playbook

Step 1

Create an inventory file called `inventory.yaml`, then specify the device IP address.

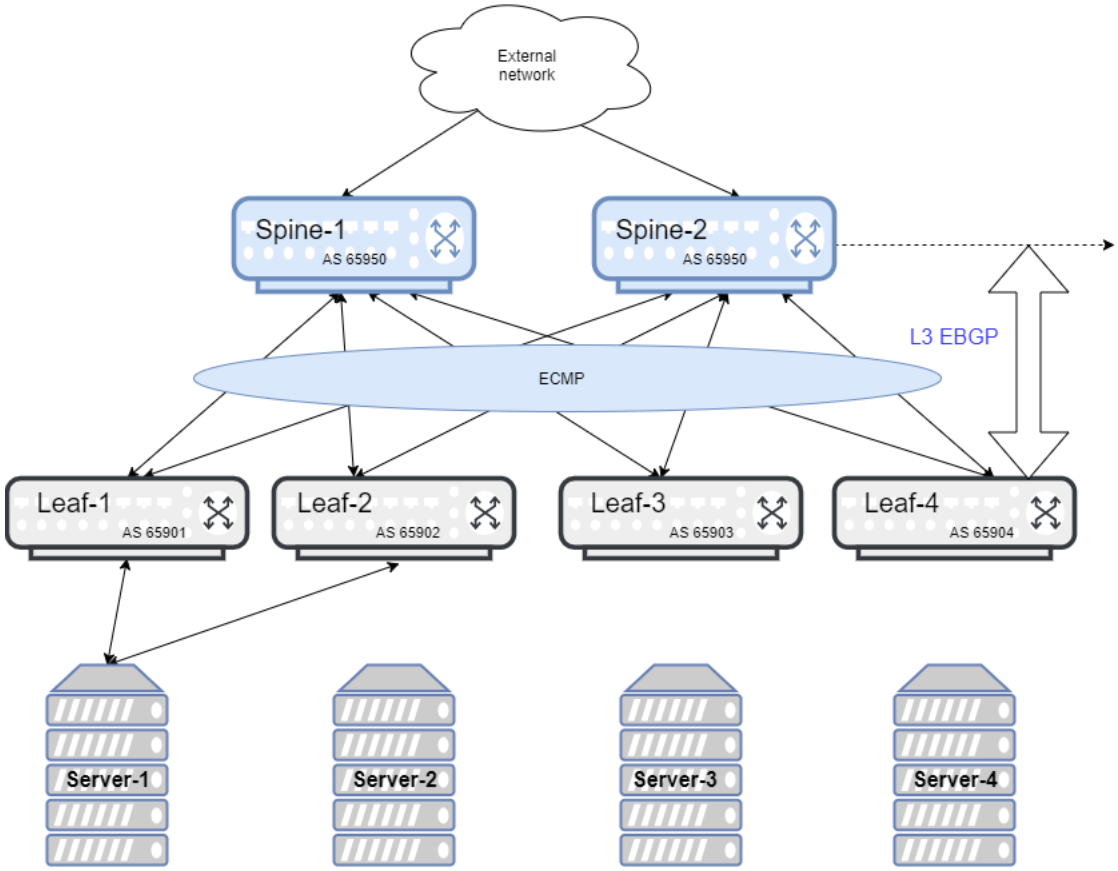
```
spine1 ansible_host=10.11.182.25
spine2 ansible_host=10.11.182.26
leaf1 ansible_host=10.11.182.27
leaf2 ansible_host=10.11.182.28
leaf3 ansible_host=10.11.182.29
leaf4 ansible_host=10.11.182.30

[spine]
spine1
spine2

[leaf]
leaf1
leaf2
leaf3
leaf4

[datacenter:children]
```

(continues on next page)


External network
Spine-1
AS 65950
Spine-2
AS 65950
L3 EBGP
ECMP
Leaf-1
AS 65901
Leaf-2
AS 65902
Leaf-3
AS 65903
Leaf-4
AS 65904
Server-1
Server-2
Server-3
Server-4

(continued from previous page)

```
spine
leaf
```

Step 2

Create a group variable file called `group_vars/all`, then define credentials and SNMP variables.

```
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

dellos_snmp:
  snmp_community:
    - name: public
      access_mode: ro
      state: present
```

Step 3

Create a group variable file called `group_vars/spine.yaml`, then define credentials, hostname, and BGP neighbors of spine group.

```
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

dellos_system:
  hostname: "{{ spine_hostname }}"

dellos_bgp:
   asn: 64901
   router_id: "{{ bgp_router_id }}"
   best_path:
      as_path: multipath-relax
      as_path_state: present
      med:
       - attribute: missing-as-worst
         state: present
   neighbor:
     - type: ipv4
       remote_asn: "{{ bgp_neigh1_remote_asn }}"
       ip: "{{ bgp_neigh1_ip }}"
       admin: up
       state: present
     - type: ipv4
       remote_asn: "{{ bgp_neigh2_remote_asn }}"
       ip: "{{ bgp_neigh2_ip }}"
       admin: up
       state: present
     - type: ipv4
       remote_asn: "{{ bgp_neigh3_remote_asn }}"
       ip: "{{ bgp_neigh3_ip }}"
       admin: up
       state: present
     - type: ipv4
       remote_asn: "{{ bgp_neigh4_remote_asn }}"
       ip: "{{ bgp_neigh4_ip }}"
```

(continues on next page)

(continued from previous page)

```
      admin: up
      state: present
    - type: ipv6
      remote_asn: "{{ bgp_neigh5_remote_asn }}"
      ip: "{{ bgp_neigh5_ip }}"
      admin: up
      address_family:
        - type: ipv4
          activate: false
          state: present
        - type: ipv6
          activate: true
          state: present
      state: present
    - type: ipv6
      remote_asn: "{{ bgp_neigh6_remote_asn }}"
      ip: "{{ bgp_neigh6_ip }}"
      admin: up
      address_family:
        - type: ipv4
          activate: false
          state: present
        - type: ipv6
          activate: true
          state: present
      state: present
    - type: ipv6
      remote_asn: "{{ bgp_neigh7_remote_asn }}"
      ip: "{{ bgp_neigh7_ip }}"
      admin: up
      address_family:
        - type: ipv4
          activate: false
          state: present
        - type: ipv6
          activate: true
          state: present
      state: present
    - type: ipv6
      remote_asn: "{{ bgp_neigh8_remote_asn }}"
      ip: "{{ bgp_neigh8_ip }}"
      admin: up
      address_family:
        - type: ipv4
          activate: false
          state: present
        - type: ipv6
          activate: true
          state: present
  state: present
```

Step 4

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and transport.

```
hostname: spine1
ansible_ssh_user: xxxxx
```

(continues on next page)

(continued from previous page)

```
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
spine_hostname: "spine-1"

dellos_interface:
    ethernet 1/1/1:
            desc: "Connected to leaf 1"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.1.1.1/24
            ipv6_and_mask: 2001:100:1:1::1/64
            state_ipv6: present
    ethernet 1/1/17:
            desc: "Connected to leaf 2"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.1.33.1/24
            ipv6_and_mask: 2001:100:1:21::1/64
            state_ipv6: present
    ethernet 1/1/25:
            desc: "Connected to leaf 3"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.1.17.1/24
            ipv6_and_mask: 2001:100:1:11::1/64
            state_ipv6: present
    ethernet 1/1/9:
            desc: "Connected to leaf 4"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.1.49.1/24
            ipv6_and_mask: 2001:100:1:31::1/64
            state_ipv6: present

bgp_router_id: "100.0.1.1"
bgp_neigh1_remote_asn: 64801
bgp_neigh1_ip: "100.1.1.2"
bgp_neigh2_remote_asn: 64803
bgp_neigh2_ip: "100.1.33.2"
bgp_neigh3_remote_asn: 64802
bgp_neigh3_ip: "100.1.17.2"
bgp_neigh4_remote_asn: 64804
bgp_neigh4_ip: "100.1.49.2"
bgp_neigh5_remote_asn: 64801
bgp_neigh5_ip: "2001:100:1:1::2"
bgp_neigh6_remote_asn: 64802
bgp_neigh6_ip: "2001:100:1:11::2"
bgp_neigh7_remote_asn: 64803
bgp_neigh7_ip: "2001:100:1:21::2"
```

(continues on next page)

(continued from previous page)

```
bgp_neigh8_remote_asn: 64804
bgp_neigh8_ip: "2001:100:1:31::2"
```

Create a host variable file called `host_vars/spine2.yaml`, then define the host, credentials, and transport.

```
hostname: spine2
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
spine_hostname: "spine-2"
dellos_interface:
    ethernet 1/1/1:
            desc: "Connected to leaf 1"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.2.1.1/24
            ipv6_and_mask: 2001:100:2:1::1/64
            state_ipv6: present
    ethernet 1/1/25:
            desc: "Connected to leaf 2"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.2.17.1/24
            ipv6_and_mask: 2001:100:2:11::1/64
            state_ipv6: present
    ethernet 1/1/17:
            desc: "Connected to leaf 3"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.2.33.1/24
            ipv6_and_mask: 2001:100:2:21::1/64
            state_ipv6: present
    ethernet 1/1/9:
            desc: "Connected to leaf 4"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.2.49.1/24
            ipv6_and_mask: 2001:100:2:31::1/64
            state_ipv6: present

bgp_router_id: "100.0.1.2"
bgp_neigh1_remote_asn: 64801
bgp_neigh1_ip: "100.2.1.2"
bgp_neigh2_remote_asn: 64802
bgp_neigh2_ip: "100.2.33.2"
bgp_neigh3_remote_asn: 64803
bgp_neigh3_ip: "100.2.17.2"
bgp_neigh4_remote_asn: 64804
bgp_neigh4_ip: "100.2.49.2"
```

(continues on next page)

(continued from previous page)

```
bgp_neigh5_remote_asn: 64801
bgp_neigh5_ip: "2001:100:2:1::2"
bgp_neigh6_remote_asn: 64802
bgp_neigh6_ip: "2001:100:2:11::2"
bgp_neigh7_remote_asn: 64803
bgp_neigh7_ip: "2001:100:2:21::2"
bgp_neigh8_remote_asn: 64804
bgp_neigh8_ip: "2001:100:2:31::2"
```

Create a host variable file called `host_vars/leaf1.yaml`, then define the host, credentials, and transport.

```
hostname: leaf1
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
dellos_system:
  hash_algo:
    algo:
      - name: ecmp
        mode: crc
        state: present
dellos_interface:
    ethernet 1/1/1:
             desc: "Connected to Spine 1"
             mtu: 9216
             portmode:
             admin: up
             switchport: False
             ip_and_mask: 100.1.1.2/24
             ipv6_and_mask: 2001:100:1:1::2/64
             state_ipv6: present
    ethernet 1/1/9:
             desc: "Connected to Spine 2"
             mtu: 9216
             portmode:
             admin: up
             switchport: False
             ip_and_mask: 100.2.1.2/24
             ipv6_and_mask: 2001:100:2:1::2/64
             state_ipv6: present
dellos_bgp:
    asn: 64801
    router_id: 100.0.2.1
    address_family_ipv4: true
    address_family_ipv6: true
    best_path:
       as_path: multipath-relax
       as_path_state: present
       med:
        - attribute: missing-as-worst
          state: present
    neighbor:
      - type: ipv4
        remote_asn: 64901
        ip: 100.1.1.1
        admin: up
        state: present
```

(continues on next page)

(continued from previous page)

```
      - type: ipv4
        remote_asn: 64901
        ip: 100.2.1.1
        admin: up
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:1:1::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:2:1::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
        state: present
    state: present
```

Create a host variable file called `host_vars/leaf2.yaml`, then define the host, credentials, and transport.

```
hostname: leaf2
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
dellos_system:
  hash_algo:
    algo:
      - name: ecmp
        mode: crc
        state: present
dellos_interface:
    ethernet 1/1/1:
             desc: "Connected to Spine 1"
             mtu: 9216
             portmode:
             admin: up
             switchport: False
             ip_and_mask: 100.1.17.2/24
             ipv6_and_mask: 2001:100:1:11::2/64
             state_ipv6: present
    ethernet 1/1/9:
             desc: "Connected to Spine 2"
             mtu: 9216
             portmode:
```

(continues on next page)

(continued from previous page)

```
            admin: up
            switchport: False
            ip_and_mask: 100.2.17.2/24
            ipv6_and_mask: 2001:100:2:11::2/64
dellos_bgp:
    asn: 64802
    router_id: 100.0.2.2
    address_family_ipv4: true
    address_family_ipv6: true
    best_path:
       as_path: multipath-relax
       as_path_state: present
       med:
        - attribute: missing-as-worst
          state: present
    neighbor:
      - type: ipv4
        remote_asn: 64901
        ip: 100.1.18.1
        admin: up
        state: present
      - type: ipv4
        remote_asn: 64901
        ip: 100.1.17.1
        admin: up
        state: present
      - type: ipv4
        remote_asn: 64901
        ip: 100.2.17.1
        admin: up
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:1:11::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:2:11::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
    state: present
```

```
hostname: leaf3
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
dellos_system:
  hash_algo:
    algo:
      - name: ecmp
        mode: crc
        state: present
dellos_interface:
    ethernet 1/1/1:
            desc: "Connected to Spine 1"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.1.33.2/24
            ipv6_and_mask: 2001:100:1:21::2/64
            state_ipv6: present
    ethernet 1/1/9:
            desc: "Connected to Spine 2"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.2.33.2/24
            ipv6_and_mask: 2001:100:2:21::2/64
dellos_bgp:
    asn: 64803
    router_id: 100.0.2.3
    address_family_ipv4: true
    address_family_ipv6: true
    best_path:
       as_path: multipath-relax
       as_path_state: present
       med:
        - attribute: missing-as-worst
          state: present
    neighbor:
      - type: ipv4
        remote_asn: 64901
        ip: 100.1.33.1
        admin: up
        state: present
      - type: ipv4
        remote_asn: 64901
        ip: 100.2.33.1
        admin: up
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:1:21::1
        admin: up
        state: present
      - type: ipv6
        remote_asn: 64901
```

(continues on next page)

(continued from previous page)

```
        ip: 2001:100:1:22::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:2:21::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
    state: present
```

Create a host variable file called `host_vars/leaf4.yaml`, then define the host, credentials, and transport.

```
hostname: leaf4
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
dellos_system:
  hash_algo:
    algo:
      - name: ecmp
        mode: crc
        state: present
dellos_interface:
    ethernet 1/1/5:
            desc: "Connected to Spine 1"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.1.49.2/24
            ipv6_and_mask: 2001:100:1:31::2/64
            state_ipv6: present
    ethernet 1/1/17:
            desc: "Connected to Spine 2"
            mtu: 9216
            portmode:
            admin: up
            switchport: False
            ip_and_mask: 100.2.49.2/24
            ipv6_and_mask: 2001:100:2:31::2/64
            state_ipv6: present
dellos_bgp:
    asn: 64804
    router_id: 100.0.2.4
```

(continues on next page)

(continued from previous page)

```
    address_family_ipv4: true
    address_family_ipv6: true
    best_path:
       as_path: multipath-relax
       as_path_state: present
       med:
        - attribute: missing-as-worst
          state: present
    neighbor:
      - type: ipv4
        remote_asn: 64901
        ip: 100.1.49.1
        admin: up
        state: present
      - type: ipv4
        remote_asn: 64901
        ip: 100.2.49.1
        admin: up
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:1:31::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
        state: present
      - type: ipv6
        remote_asn: 64901
        ip: 2001:100:2:31::1
        admin: up
        address_family:
          - type: ipv4
            activate: false
            state: present
          - type: ipv6
            activate: true
            state: present
    state: present
```

Step 5

Create a playbook called `datacenter.yaml`.

```
---
- hosts: datacenter
  gather_facts: no
  connection: network_cli
  roles:
        - Dell-Networking.dellos-interface
        - Dell-Networking.dellos-bgp
        - Dell-Networking.dellos-snmp
        - Dell-Networking.dellos-system
```

Step 6

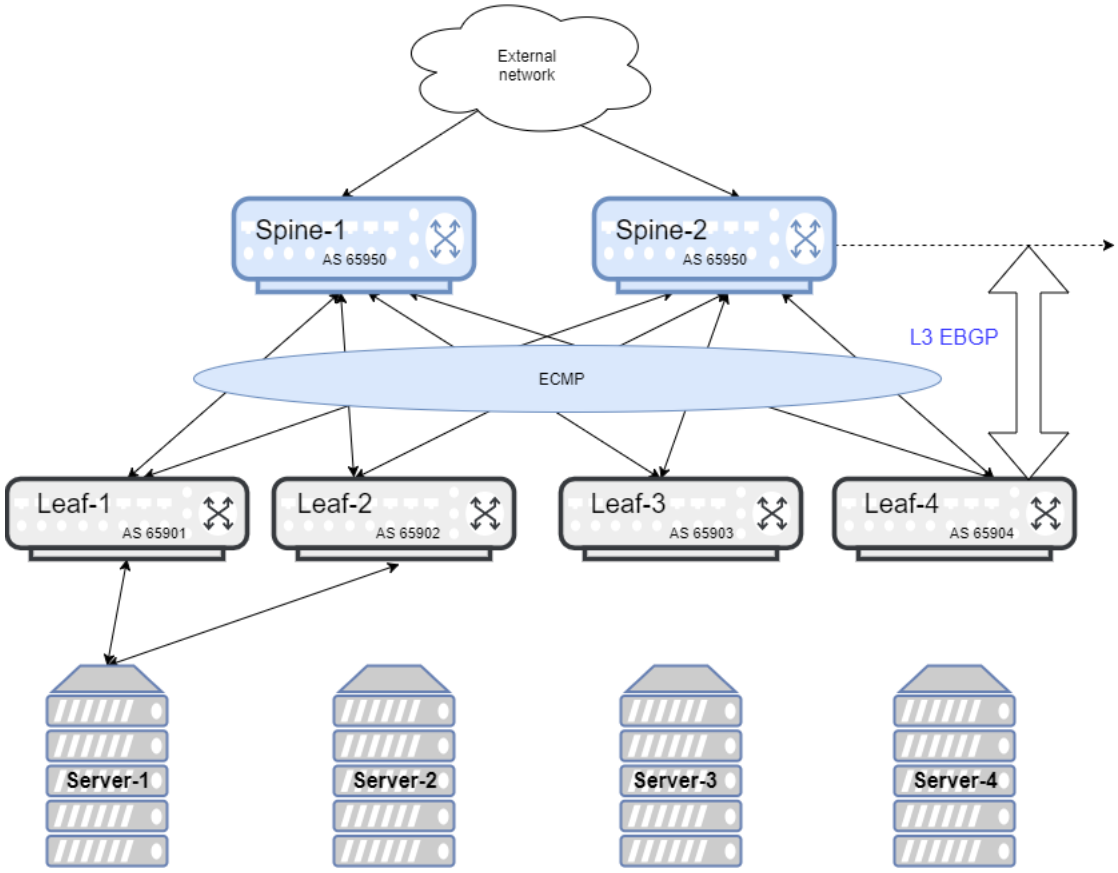
Run the playbook.

```
ansible-playbook -i inventory.yaml datacenter.yaml
```

CHAPTER 8

Provisioning hot swap use case using Dell EMC Ansible modules

This example use case topology includes a simple two-tier CLOS fabric with two spines and four leafs. These steps show how you Spine 2 can be hot swapped without traffic loss.



8.1 Create simple Ansible playbook

- **Part 1** ? Covers creating an inventory file and host variable file for spine2, creating a pre-step hot swap playbook, then running the playbook
- **Part 2** ? Covers creating an inventory file and host variable file for each leaf (four), creating a playbook to delete the ECMP path for spine2 from each leaf, then running the playbook
- **Part 3** ? Covers replacing spine2 with a new switch, booting an OS10 image, creating inventory and host variable files for the new spine2, creating a post hot swap playbook, then running the playbook

8.2 Part 1

See the CLOS fabric example to configure a six-node CLOS fabric with eBGP. Use the example and run the playbook.

Step 1

Create an inventory file called `inventory.yaml`, then specify the device IP address for spine2.

```
spine2 ansible_host=10.16.204.57

[spine]
spine2

[leaf]

[datacenter:children]
spine
```

Step 2

Create a host variable file called `host_vars/spine2.yaml`, then define the host and credentials.

- Take a backup of the running configuration to a remote location
- Shut down the BGP neighbors in the hot swap switch to avoid traffic drop

```
hostname: spine2
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

copy_running_remote:
    - copy_type: scp
      username: linuxadmin
      password: linuxadmin
      host_ip: 10.16.204.62
      file_path: /home/linuxadmin/running-config

dellos_bgp:
    asn: 64901
    neighbor:
      - type: ipv4
        remote_asn: 64801
        ip: 100.2.1.2
        admin: down
        state: present
```

(continues on next page)

(continued from previous page)

```
      - type: ipv4
        remote_asn: 64802
        ip: 100.2.33.2
        admin: down
        state: present
      - type: ipv4
        remote_asn: 64803
        ip: 100.2.17.2
        admin: down
        state: present
      - type: ipv4
        remote_asn: 64804
        ip: 100.2.49.2
        admin: down
        state: present
      - type: ipv6
        remote_asn: 64801
        ip: 2001:100:2:1::2
        admin: down
        state: present
      - type: ipv6
        remote_asn: 64802
        ip: 2001:100:2:11::2
        admin: down
        state: present
      - type: ipv6
        remote_asn: 64803
        ip: 2001:100:2:21::2
        admin: down
        state: present
      - type: ipv6
        remote_asn: 64804
        ip: 2001:100:2:31::2
        admin: down
        state: present
    state: present
```

Step 3

Create a playbook called `hot_swap_pre_step.yaml`.

```
---
- hosts: datacenter
  gather_facts: no
  connection: network_cli
    tasks:
      - name: Assembling configfurations
        assemble: src={{ build_dir }} dest={{ build_dir }}/{{hostname}}.conf regexp=
↪'\\S_{{hostname}}\\S'
      - name: "copy running config to remote location"
        dellos10_command:
          commands:
             - command: 'copy running-configuration {{item.copy_type}}://{{item.
↪username}}:{{item.password}}@{{item.host_ip}}:{{item.file_path}}'
               #If the switch asks for credentials for copy command, use the below␣
↪commented statements to give the prompt and password
               #prompt: 'admin:'
```

(continues on next page)

(continued from previous page)

```
              #answer: 'admin'
        with_items: '{{copy_running_remote}}'
- hosts: datacenter
  connection: network_cli
  vars:
    build_dir: "/root/debug"
  roles:
    - Dell-Networking.dellos-bgp
```

Step 4

Run the playbook.

```
ansible-playbook -i inventory.yaml hot_swap_pre_step.yaml
```

8.3 Part 2

Step 1

After shutting the neighborship in the spine2 switch, check if the ECMP path to spine2 is deleted in each of the leaf switches.

Step 2

Create an inventory file called `inventory.yaml`, then specify the device IP address of all leaf switches.

```
leaf1 ansible_host=10.16.204.27
leaf2 ansible_host=10.16.204.28
leaf3 ansible_host=10.16.204.29
leaf4 ansible_host=10.16.204.30

[spine]

[leaf]
leaf1
leaf2
leaf3
leaf4

[datacenter:children]
leaf
```

Step 3

Create a host variable file called `host_vars/leaf1.yaml`, then define the host and credentials. The remote_neighbor_ip is the EBGP neighbor IP of spine2 with each of each leaf switch (see the CLOS fabric example for EBGP configuration):

```
hostname: leaf1
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

remote_neighbor_ip: "100.2.1.1"
```

Create a host variable file called `host_vars/leaf2.yaml`, then define the host and credentials.

```
hostname: leaf2
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

remote_neighbor_ip: "100.2.17.1"
```

Create a host variable file called `host_vars/leaf3.yaml`, then define the host and credentials.

```
hostname: leaf3
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

remote_neighbor_ip: "100.2.33.1"
```

Create a host variable file called `host_vars/leaf4.yaml`, then define the host and credentials.

```
hostname: leaf4
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

remote_neighbor_ip: "100.2.49.1"
```

Step 4

Create a playbook called `waitfor_ecmp_path_delete.yaml`.

A debug message will print when the ECMP path for spine2 is deleted in each of the leaf switches.

```
---
- hosts: datacenter
  gather_facts: no
  connection: network_cli
  vars:
    build_dir: "/root/debug"
  tasks:
    - name: Assembling configfurations
        assemble: src={{ build_dir }} dest={{ build_dir }}/{{hostname}}.conf regexp=
↪'\\S_{{hostname}}\\S'
    - name: "Wait for spine2 routes delete in {{ hostname }}"
        dellos10_command:
          commands:
             - command: "show ip route bgp | grep {{ remote_neighbor_ip }}"
      retries: 10
      delay: 5
      register: result
      until: result.stdout[0] == ""
    - debug:
        msg: "{{ hostname }} has deleted the ECMP to spine2 switch"
      when: result.stdout[0] == ""
```

Step 5

Run the playbook.

```
ansible-playbook -i inventory.yaml waitfor_ecmp_path_delete.yaml
```

8.4 Part 3

Step 1

After checking the spine2 ECMP path deletion in all leaf switches, replace spine2 with a new switch. The new spine2 switch should be connected as the old spine switch after it boots up with an OS10 image.

- Manually assign the same spine2 management IP address (for example, 10.16.204.57)
- Use the Management IP provided by the DHCP server

Step 2

Create an inventory file called `inventory.yaml`, then specify the device IP address for spine2. The device IP can be same spine2 IP or an IP obtained from the DHCP server (x.x.x.x).

```
spine2 ansible_host=x.x.x.x

[spine]
spine2

[leaf]

[datacenter:children]
spine
```

Step 3

Create a host variable file called `host_vars/spine2.yaml`, then define the host, credentials, and apply the same backup configuration that was saved earlier.

```
hostname: spine2
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

copy_remote_running:
    - copy_type: scp
      username: linuxadmin
      password: linuxadmin
      host_ip: 10.16.204.62
      file_path: /home/linuxadmin/running-config
```

Step 4

Create a playbook called `hot_swap_post_step.yaml`.

```
---
- hosts: datacenter
  gather_facts: no
  connection: network_cli
    tasks:
      - name: Assembling configfurations
        assemble: src={{ build_dir }} dest={{ build_dir }}/{{hostname}}.conf regexp=
↪'\\S_{{hostname}}\\S'
      - name: "copy running config to remote location"
        dellos10_command:
          commands:
             - command: 'copy {{item.copy_type}}://{{item.username}}:{{item.password}}
↪@{{item.host_ip}}:{{item.file_path}} running-configuration'
```

(continues on next page)

(continued from previous page)

```
                #If the switch asks for credentials for copy command, use the below␣
↪commented statements to give the prompt and password
                #prompt: 'admin:'
                #answer: 'admin'
        with_items: '{{copy_remote_running}}'
```

Step 5

Run the playbook.

```
ansible-playbook -i inventory.yaml hot_swap_post_step.yaml
```

CHAPTER 9

Install or upgrade devices running Dell EMC SmartFabric OS10 using Ansible

This example explains how to use Ansible to install or upgrade the software image on a device running Dell EMC SmartFabric OS10. The example playbook uses the `dellos-image-upgrade` role to upgrade or install a SmartFabric OS10 image on a specified switch.

Before using Ansible to install the software image, you must download the software image via FTP/TFTP/SCP/HTTPS, then specify the path to the image in the playbook. The `dellos-image-upgrade` role uses `dellos10_command` to install or upgrade the switch, and `wait_for` is used to identify the progress of the upgrade operation. Validation of the upgrade operation is handled using the `dellos10_facts` module.

9.1 Creating simple Ansible playbook

9.1.1 Step 1

Create an inventory file called `inventory.yaml`, then specify the device IP address.

```
spine1 ansible_host=2.2.2.1

[spine]
spine1

[datacenter:children]
spine
```

9.1.2 Step 2

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and transport:

```
hostname: spine1
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10

dellos_image_upgrade:
    operation_type: install
    software_image_url: tftp://1.1.1.1/PKGS_OS10-Enterprise-10.2.9999E.5790-installer-
↪x86_64.bin
    software_version: 10.2.9999E
```

9.1.3 Step 3

Create a playbook called `datacenter.yaml`.

```
---
- hosts: datacenter
  gather_facts: no
  connection:network_cli
  roles:
        - Dell-Networking.dellos-image-upgrade
```

9.1.4 Step 4

Run the playbook.

```
ansible-playbook -i inventory.yaml datacenter.yaml
```

CHAPTER 10

Use Ansible to perform ZTD on devices running Dell EMC SmartFabric OS10

This example describes how to use Ansible to perform zero-touch deployment (ZTP). It installs or upgrades a software image on a device running Dell EMC SmartFabric OS10.

The example playbook uses the `dellos-image-upgrade` role to upgrade or install a SmartFabric OS10 image on a specified switch, followed by a `dellos-copy-config` role to push configurations post installation on the device.

Before using Ansible to install the software image, you must download the software image via FTP/TFTP/SCP/HTTPDS, then specify the path to the image in the playbook.

10.1 Installation

Step 1 ? Set up AWX

- Download AWX 4.0.0 release, make sure you have latest ansible version and Install AWX

```
ansible-galaxy install -r dellemc_roles.txt
apt-add-repository --yes --update ppa:ansible/ansible
apt install ansible -y
apt install docker.io
apt install python-pip -y
pip install docker
pip install docker-compose
wget https://github.com/ansible/awx/archive/4.0.0.zip # Download the zip file
unzip 4.0.0.zip # unzip the downloaded file
```

Open installer/inventory file and change Docker parameters.

```
postgres_data_dir=/var/lib/pgdocker # change from /tmp to /var/lib
docker_compose_dir=/var/lib/awxcompose # change from /tmp to /var/lib
```

Under installer folder, run the `install.yml` command.

```
cd installer
ansible-playbook -i inventory install.yml -vvv
```

- Launch AWX in browser
- Go to Projects and create AWX project (name, description, and scm type)
- Create playbook in project directory
- Go to Inventories and create inventory and Hostkey
- Go to template and create job template

Step 2 ? Add curl script to contact an Ansible server

- Go to ztd-provision-url(http://X.X.X.X/ztd.sh) defined in the DHCP server configuration, and include the curl command to the ztd script.

```
e.g /usr/bin/curl -H "Content-Type:application/json" -k -X POST --data '{"host_
↪config_key":"'7d07e79ebdc8f7c292e495daac0fe16b'"}' -u username:password https://
↪X.X.X.X/api/v2/job_templates/xxx/callback/
```

Step 3 ? Run ZTD from the SmartFabric OS10 device

- Run the ZTD by rebooting the switch. Enter the reload ztd command.

```
OS10# reload ztd
```

10.2 Example playbook

The `dellos-image-upgrade` role uses the `dellos10_command` to install or upgrade the switch, and `wait_for` is used to identify the progress of the upgrade operation. The `dellos-copy-config` role uses the `dellos10_config` module to push configurations to the device.

Sample hosts file

```
ztdswitch ansible_host= <ip_address>
```

Sample host_vars for Dell-Networking.dellos-image-upgrade

```
hostname: ztdswitch
ansible_become: yes
ansible_become_method: xxxxx
ansible_become_pass: xxxxx
ansible_ssh_user: xxxxx
ansible_ssh_pass: xxxxx
ansible_network_os: dellos10
dellos_image_upgrade:
  operation_type: install
  software_image_url: tftp://X.X.X.X/PKGS_OS10-Enterprise-10.2.9999E.5790-
↪installer-x86_64.bin
  software_version: 10.2.9999E
```

Simple playbook to setup ZTD

```
- hosts: ztdswitch
  connection: network_cli
  roles:
     - Dell-Networking.dellos-image-upgrade
     - Dell-Networking.dellos-copy-config
```

CHAPTER 11

Provision SmartFabric Services using Dell EMC Ansible modules example

This example describes how to use Ansible to build a SmartFabric cluster and provision SFS with Dell EMC SmartFabric OS10 switches. The sample topology is built with one spine and two leafs connected as mesh, with BGP running between the leafs. VLTi is configured between the leafs.

The module example also describes the configuration of SFS and provisioning of attributes through REST APIs.

11.1 Create an Ansible playbook for SmartFabric setup

Step 1

Create an inventory file called `hosts.yaml` and specify the device IP address and python_interpreter.

```
leaf1 ansible_host=10.11.180.9 ansible_python_interpreter=/usr/bin/python3
leaf2 ansible_host=10.11.180.8 ansible_python_interpreter=/usr/bin/python3
spine1 ansible_host=10.11.180.10 ansible_python_interpreter=/usr/bin/python3

[Spine]
spine1

[Leaf]
leaf1
leaf2

[LeafAndSpineSwitch:children]
Spine
Leaf
```

Step 2

Create a host variable file called `host_vars/leaf1.yaml`, then define the host, credentials, and SFS fabric cluster setup input.

```
---
ansible_host: 10.11.180.8
ansible_network_os: dellos10
ansible_user: XXXXX
ansible_password: XXXXX

sfs_setup:
  - service_enable: True
    role: LEAF
    icl_ports: ["ethernet1/1/5","ethernet1/1/6"]
```

Create a host variable file called `host_vars/leaf2.yaml`, then define the host, credentials, and SFS fabric cluster setup input.

```
---
ansible_host: 10.11.180.9
ansible_network_os: dellos10
ansible_user: XXXXX
ansible_password: XXXXX

sfs_setup:
  - service_enable: True
    role: LEAF
    icl_ports: ["ethernet1/1/5","ethernet1/1/6"]
```

Create a host variable file called `host_vars/spine1.yaml`, then define the host, credentials, and SFS fabric cluster setup input.

```
---
ansible_host: 10.11.180.10
ansible_network_os: dellos10
ansible_user: XXXXX
ansible_password: XXXXX

sfs_setup:
  - service_enable: True
    role: SPINE
    icl_ports: ["ethernet1/1/5","ethernet1/1/6"]
```

Step 3

Create a playbook called `sfs_setup.yml`.

```
---
- name: SFS setup
  hosts: LeafAndSpineSwitch
  gather_facts: False
  connection: local
  roles:
    - sfs-setup
```

Step 4

Run the playbook.

```
ansible-playbook -i hosts.yaml sfs_setup.yml
```

11.2 Create an Ansible playbook for SmartFabric API services

Step 1

Use the same inventory `hosts.yaml` for provisioning once SFS setup is ready. Create a group variable file called `group_vars/sfs.all.yaml`, then define the SFS input model.

```
---
sfs_port_breakout:
  - target_port: GGVQG02:ethernet1/1/22
    breakout_type: 4X10GE
  - target_port: GGVQG02:ethernet1/1/23
    breakout_type: 1X100GE
  - target_port: GGVQG02:ethernet1/1/24
    breakout_type: 1X40GE

sfs_port_property:
  - target_port: GGVQG02:ethernet1/1/25
    port_description: "Description for ethernet1/1/25"
    port_name: ethernet1/1/25
    admin_status: Enabled
    mtu: 1564
    auto_neg: Enabled
    configured_speed: 1024
  - target_port: GGVQG02:ethernet1/1/26
    port_description: "Description for ethernet1/1/26"
    port_name: ethernet1/1/26
    admin_status: Enabled
    mtu: 2564
    auto_neg: Enabled
    configured_speed: 1024

sfs_uplinks:
  - uplink_name: Leaf-1-port-21
    uplink_description: "Leaf-1-port-21"
    uplink_id: "Leaf-1-port-21"
    media_type: Ethernet
    node: GGVQG02
    configuration_interfaces:
      - "ethernet1/1/21"
      - "ethernet1/1/22"
    tagged_networks:
      - "Client_Control_Network"
    untagged_network: "Client_Control_Network"
    lag_type: "Static"
    uplink_type: "Normal"
    state: present
  - uplink_name: Leaf-1-port-25
    uplink_description: "Leaf-1-port-25"
    uplink_id: "Leaf-1-port-25"
    media_type: Ethernet
    node: GGVQG02
    configuration_interfaces:
      - "ethernet1/1/25"
      - "ethernet1/1/26"
    tagged_networks:
      - "Client_Management_Network"
```

(continues on next page)

(continued from previous page)

```
    untagged_network: "Client_Control_Network"
    lag_type: "Dynamic"
    uplink_type: "JumpBox"
    state: present

sfs_route_policies:
  - policy_id: policyBGP100
    policy_name: policyBGP100name
    policy_description: policyBGP100desc
    address_family_type: ipv4
    remote_address: "192.168.2.6"
    remote_loopback_address: "192.168.2.8"
    remote_as: 65001
    policy_type: 1
    sender_side_loop_detection: 1
    route_filter_enable: 1
    state: present
  - policy_id: policyBGP101
    policy_name: policyBGP101name
    policy_description: policyBGP101desc
    address_family_type: ipv4
    remote_address: "192.168.2.2"
    remote_loopback_address: "192.168.2.4"
    remote_as: 65001
    policy_type: 1
    sender_side_loop_detection: 1
    route_filter_enable: 1
    state: present
  - policy_id: policyStaticCRoute1
    policy_name: policyStaticRoute1name
    policy_description: policyStaticRoute1desc
    policy_type: 2
    ipv4_address_prefix: "99.99.99.0"
    ipv4_prefix_len: 24
    ipv4_nexthop_ip: "99.99.99.2"
    state: present

sfs_node_policy_mapping:
  - node: "GGVQG02"
    policy_list:
      - policyBGP100
      - policyBGP101
      - policyStaticCRoute1
    state: present

sfs_networks:
  - name: Leaf-test-sfs-VXLAN
    id: Leaf-test-sfs-VXLAN
    vlan_min: 650
    vlan_max: 650
    qos_priority: Silver
    type: VXLAN
    description: "SFS Network Create Test From Ansible"
    address_family: inet
    gateway_ip_address: ["192.168.1.3"]
    helper_address: ["10.10.10.10","11.11.11.11"]
    ip_address_list: ["192.168.1.2","192.168.1.4"]
```

(continues on next page)

(continued from previous page)

```
    prefix_length: 31
    route_map: "routemap1"
    virtual_network: esxi_build650
    state: present
  - name: Leaf-test-sfs-VXLAN
    id: Leaf-test-GeneralPurpose
    vlan_min: 750
    vlan_max: 750
    qos_priority: Gold
    type: GeneralPurpose
    description: "SFS Network Create Test From Ansible"
    address_family: inet
    virtual_network: vn750
    state: present
  - name: Leaf-test-sfs1-network-l3
    id: Leaf-test-sfs1-network-l3
    vlan_min: 550
    vlan_max: 550
    qos_priority: Bronze
    type: L3
    description: "SFS L3 Network Create Test From Ansible"
    address_family: inet
    gateway_ip_address: ["192.168.1.3"]
    helper_address: ["10.10.10.10","11.11.11.11"]
    ip_address_list: ["192.168.1.2","192.168.1.4"]
    prefix_length: 31
    route_map: "routemap1"
    state: present
  - name: Leaf-test-sfs1-network-l3-routed
    id: Leaf-test-sfs1-network-l3-routed
    qos_priority: Bronze
    type: L3_ROUTED
    description: "SFS L3-ROUTED Network Create Test From Ansible"
    address_family: inet
    gateway_ip_address: ["192.168.1.3"]
    helper_address: ["10.10.10.10","15.15.15.15"]
    ip_address_list: ["192.168.1.2","192.168.1.4","192.168.1.6"]
    prefix_length: 31
    route_map: "routemap2"
    state: present

sfs_virtual_networks:
  - virtual_network_name: "vnet604"
    virtual_network_description: "vnet604 Create"
    virtual_network_type: "General Purpose (Bronze)"
    vxlanvni: 1604
    vltvlanid: 604
    gateway_ip_address: "172.17.105.1"
    gateway_mac_address: "00:11:12:01:23:36"
    prefix_length: 24
    address_family: "inet"
    ip_address_list:
      - "172.17.105.2"
      - "172.17.105.3"
    helper_address: ["2.2.2.2","3.3.3.3"]
    state: present
  - virtual_network_name: "vnet605"
```

(continues on next page)

(continued from previous page)

```
    virtual_network_description: "vnet605 Create"
    virtual_network_type: "Cluster Interconnect"
    vxlanvni: 1605
    vltvlanid: 605
    gateway_ip_address: "172.17.105.1"
    gateway_mac_address: "00:11:12:01:23:36"
    prefix_length: 24
    address_family: "inet"
    ip_address_list:
      - "172.17.105.10"
      - "172.17.105.11"
    helper_address: ["10.10.10.10","11.11.11.11"]
    state: present

sfs_server_profiles:
  - server_id: server-1
    bonding_technology: Static
    interface_profiles:
      - id: ethernet1/1/43
        tagged_networks:
          - Client_Control_Network
        nic_bonded: True
        state: present
      - id: ethernet1/1/44
        tagged_networks:
          - Client_Control_Network
        nic_bonded: True
        state: present
    state: present
  - server_id: server-2
    bonding_technology: LACP
    interface_profiles:
      - id: ethernet1/1/33
        tagged_networks:
          - Client_Management_Network
        nic_bonded: True
        state: present
      - id: ethernet1/1/34
        tagged_networks:
          - Client_Management_Network
        nic_bonded: True
        state: present
    state: present

sfs_fabric_property:
  - leaf_asn: 65011
    spine_asn: 65012
    private_subnet_prefix: "172.16.0.0"
    private_prefix_len: 16
    global_subnet_prefix: "172.30.0.0"
    global_prefix_len: 16
    client_control_vlan: 3939
    client_management_vlan: 4091

sfs_fabric_reboot:
  - node: GGVQG02
    state: absent
```

Step 2

Create a playbook called `sfs_provision.yml`.

```
---
- name: SFS Provisioning
  hosts: localhost
  gather_facts: False
  connection: local
  pre_tasks:
    - name: Include Variables for sfs provisioning
      include_vars:
        file: group_vars/sfs.all.yaml
  roles:
    - sfs-network
    - sfs-virtual-network
    - sfs-uplink
    - sfs-route-policy
    - sfs-node-policy-mapping
    - sfs-port-breakout
    - sfs-port-properties
    - sfs-validation-errors
    - sfs-server-profile
```

Step 3

Run the playbook.

```
ansible-playbook -i hosts.yaml sfs_provision.yml
```

CHAPTER 12

Frequently asked questions

You can easily find answers to commonly asked questions about Dell EMC Ansible modules and roles.

Which version of Ansible supports Dell EMC Ansible modules?

Ansible 2.2 and later.

What are the minimum OS versions for Ansible support?

OS version 6.3.1 and above; OS version 9.10.0.1P13 and above; OS version 10.2 and later.

What do the Dell EMC Ansible roles provide?

The roles are a package of multiple Dell EMC OS features which are provided for easy installation, configuration, and packaging. They currently contain configuration for system, interface, VLAN, LAG, BGP, and xSTP.

Do Dell EMC Ansible roles work with Ansible Tower?

Yes, these roles work with Ansible Tower for management.

Is there dnosX_template module support for OS6/OS9/OS10?

No. Ansible has deprecated support for the template module, replacing it with the config module (see Deprecations).

CHAPTER 13

Release notes

This information contains the release notes for Dell EMC Ansible support.

13.1 Release 3.0.0

This release introduces new roles.

- `dellos-copy-config`
- `dellos-flow-monitor`
- `dellos-image-upgrade`
- `dellos-ntp`
- `dellos-qos`
- `dellos-route-map`

13.2 Release 2.0.0

This release introduces new roles.

- `dellos-aaa`
- `dellos-acl`
- `dellos-dcb`
- `dellos-dns`
- `dellos-ecmp`
- `dellos-lldp`
- `dellos-prefix-list`

- dellos-sflow
- dellos-vlt
- dellos-vrf
- dellos-vrrp
- dellos-snmp *
- dellos-users *
- dellos-logging *

Note: Roles with an asterisk (*) are part of dellos-system role in version 1.0.0.

13.3 Release 1.0.0

This release introduces initial Ansible support for Dell EMC OS6, OS9, and OS10.

- New modules:
  - dellos6_command
  - dellos6_config
  - dellos6_facts
  - dellos9_command
  - dellos9_config
  - dellos9_facts
  - dellos10_command
  - dellos10_config
  - dellos10_facts
- New roles:
  - dellos-bgp
  - dellos-interface
  - dellos-lag
  - dellos-system
  - dellos-vlan
  - dellos-xstp
- Known issues:
  - dellos9_command Ansible hangs after reload command issued to remote device (see Issue 5462)
  - dellos9_command confirm prompt timeout (see Issue 5534)

CHAPTER 14

Support

You can submit issues for Dell EMC modules at Ansible Github Issues.

Submit issues for Dell EMC roles at:

- dellos-aaa role
- dellos-acl role
- dellos-bgp role
- dellos-copy-config role
- dellos-dcb role
- dellos-dns role
- dellos-ecmp role
- dellos-flow-monitor role
- dellos-image-upgrade role
- dellos-interface role
- dellos-lag role
- dellos-lldp role
- dellos-logging role
- dellos-ntp role
- dellos-prefix-list role
- dellos-qos role
- dellos-route-map role
- dellos-sflow role
- dellos-snmp role
- dellos-system role

- dellos-users role
- dellos-vlan role
- dellos-vlt role
- dellos-vrf role
- dellos-vrrp role
- dellos-xstp role

14.1 Contact

You can send general comments and feedback to networking_devops_tools@dell.com.

CHAPTER 15

License