

---

# **Dell EMC Networking Ansible Integration Documentation**

***Release 2.0***

**Dell EMC Networking Team**

**Nov 26, 2018**



---

## Table of Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Ansible . . . . .	1
1.2	Dell EMC Networking Ansible integration . . . . .	1
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Ansible modules . . . . .	3
2.2	Ansible roles . . . . .	3
2.3	Dell EMC Networking devices . . . . .	4
<b>3</b>	<b>Dell EMC Networking Ansible modules</b>	<b>7</b>
3.1	OS6 modules . . . . .	7
3.2	OS9 modules . . . . .	7
3.3	OS10 modules . . . . .	7
3.4	OPX module . . . . .	8
<b>4</b>	<b>Dell EMC Networking Ansible roles</b>	<b>9</b>
4.1	AAA role . . . . .	9
4.2	ACL role . . . . .	9
4.3	BGP role . . . . .	9
4.4	Copy-config role . . . . .	10
4.5	DCB role . . . . .	10
4.6	DNS role . . . . .	10
4.7	ECMP role . . . . .	10
4.8	Flow-monitor role . . . . .	10
4.9	Image-upgrade role . . . . .	10
4.10	Interface role . . . . .	10
4.11	LAG role . . . . .	11
4.12	LLDP role . . . . .	11
4.13	Logging role . . . . .	11
4.14	NTP role . . . . .	11
4.15	Prefix-list role . . . . .	11
4.16	QoS role . . . . .	11
4.17	Route-map role . . . . .	11
4.18	sFlow role . . . . .	12
4.19	SNMP role . . . . .	12
4.20	System role . . . . .	12
4.21	Users role . . . . .	12

4.22	VLAN role . . . . .	12
4.23	VLT role . . . . .	12
4.24	VRF role . . . . .	13
4.25	VRRP role . . . . .	13
4.26	xSTP role . . . . .	13
<b>5</b>	<b>Support matrix of Dell EMC Networking Ansible roles</b>	<b>15</b>
<b>6</b>	<b>Dell EMC Networking Ansible module examples</b>	<b>17</b>
6.1	Create simple Ansible playbook . . . . .	17
6.2	Create simple Ansible playbook using connection="netconf" . . . . .	18
6.3	Run Dell EMC Networking Ansible examples . . . . .	18
6.4	Run Openswitch OPX Ansible example . . . . .	20
6.5	Playbook using Ansible roles example . . . . .	20
<b>7</b>	<b>Provision CLOS fabric using Dell EMC Networking Ansible modules example</b>	<b>23</b>
7.1	Creating a Simple Ansible Playbook . . . . .	23
<b>8</b>	<b>Provisioning hot swap use case using Dell EMC Networking Ansible modules</b>	<b>37</b>
8.1	Create simple Ansible playbook . . . . .	38
<b>9</b>	<b>Install or upgrade devices running Dell EMC Networking OS10 using Ansible</b>	<b>43</b>
9.1	Creating simple Ansible playbook . . . . .	43
<b>10</b>	<b>Frequently asked questions</b>	<b>45</b>
<b>11</b>	<b>Release notes</b>	<b>47</b>
11.1	Release 3.0.0 . . . . .	47
11.2	Release 2.0.0 . . . . .	47
11.3	Release 1.0.0 . . . . .	48
<b>12</b>	<b>Support</b>	<b>49</b>
12.1	Contact . . . . .	50
<b>13</b>	<b>License</b>	<b>51</b>

This information explains Ansible and the Dell EMC Networking 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 Networking Ansible integration

Starting with Ansible 2.3, the Ansible core supports Dell EMC Networking 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 Networking devices.

### 2.1 Ansible modules

Dell EMC Networking Ansible modules for dellos9, dellos10 and dellos6 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 Networking 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 Networking Ansible roles.

## 2.3 Dell EMC Networking devices

Dell EMC Networking 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) # 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
```



---

# Dell EMC Networking 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 Networking OS6
- `dellos6_config`: Manage configuration sections on remote devices running Dell EMC Networking OS6
- `dellos6_facts`: Collect facts from remote devices running Dell EMC Networking OS6

## 3.2 OS9 modules

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

## 3.3 OS10 modules

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

## 3.4 OPX module

- `opx_cps`: Manage configuration sections using CPS on remote devices running Dell EMC Networking OS10

---

### Dell EMC Networking Ansible roles

---

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

#### 4.1 AAA role

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

Abstracted for OS9

#### 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 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 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 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 size, 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 algorithm, 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



---

## Support matrix of Dell EMC Networking Ansible roles

---

This table shows the support matrix between Ansible roles and Dell EMC Networking 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



---

## Dell EMC Networking Ansible module examples

---

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

### 6.1 Create simple Ansible playbook

1. Create an inventory file called *inventory.yaml* and specify the IP address.

```
spine1 ansible_host=10.11.182.16
```

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
```

3. Create a playbook called *showver.yaml*.

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

tasks:
- name: "Get Dell EMC OS9 Show version"
  dellos9_command:
    commands: ['show version']
  register: show_ver

- debug: var=show_ver
```

#### 4. Run the playbook

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

## 6.2 Create simple Ansible playbook using connection="netconf"

#### 1. Create an inventory file called *inventory.yaml* and specify the IP address.

```
spine1 ansible_host=10.11.182.16
```

#### 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
```

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

#### 4. Run the playbook

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

## 6.3 Run Dell EMC Networking Ansible examples

Use these sample Ansible playbooks to understand how to use Dell EMC Networking 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 Openswitch OPX Ansible example

Use the below example to configure VLAN using CPS operations.

1. Create an inventory file called *inventory.yaml* and specify the IP address.

```
spinel ansible_host=10.11.182.16
```

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
```

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"
```

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.

1. Create an inventory file called *inventory.yaml* and specify the device IP address.

```
spinel ansible_host= <ip_address>
```

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:
```

(continues on next page)



(continued from previous page)

```

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

```

### 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

```

### 4. Run the playbook.

```

ansible-playbook -i inventory.yaml switch_config.yaml

```



---

## Provision CLOS fabric using Dell EMC Networking Ansible modules example

---

This example describes how to use Ansible to build a CLOS fabric with Dell EMC Networking 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 number 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 Creating a Simple Ansible Playbook

#### Step 1

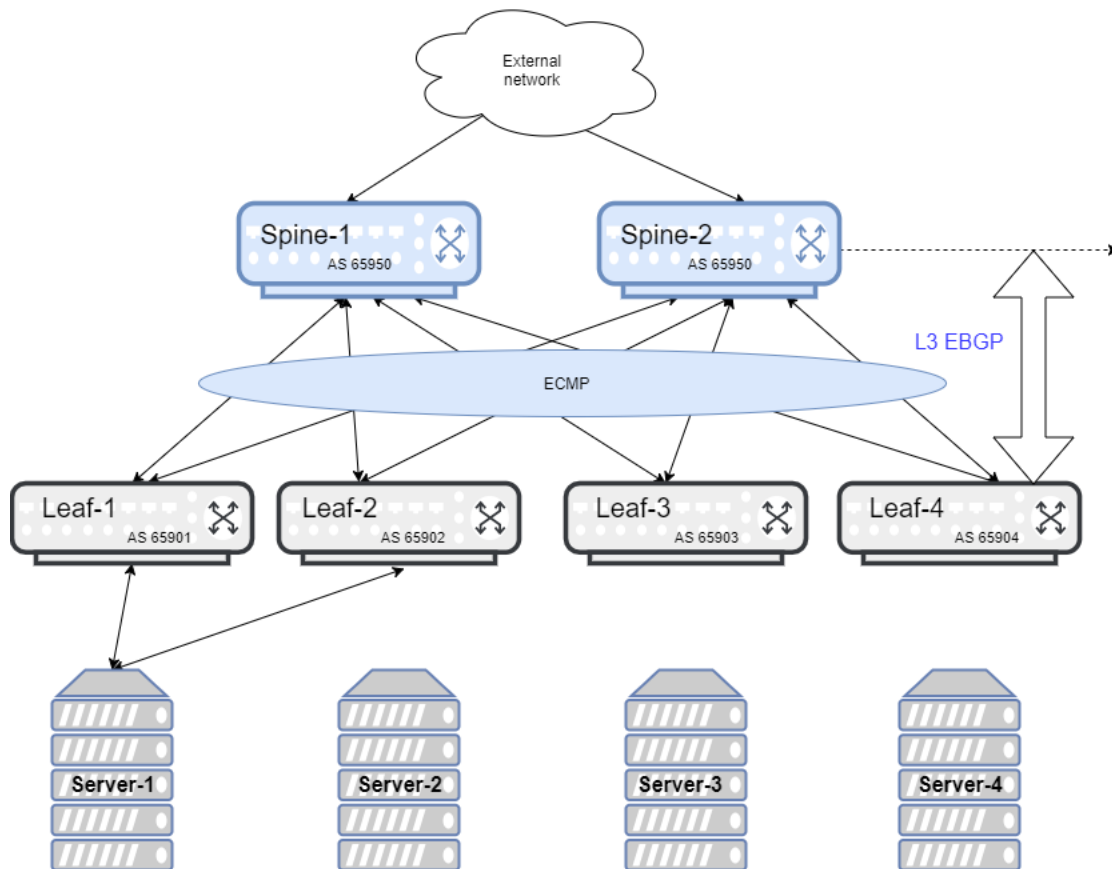
Create an inventory file called `inventory.yaml` and specify the device IP address:

```
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
```

(continues on next page)



(continued from previous page)

```
[datacenter:children]
spine
leaf
```

## 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_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
dellos_bgp:
  asn: 64901
  router_id: 100.0.1.1
  best_path:
    as_path: multipath-relax
    as_path_state: present
  med:
    - attribute: missing-as-worst
```

(continues on next page)

(continued from previous page)

```
    state: present
neighbor:
- type: ipv4
  remote_asn: 64801
  ip: 100.1.1.2
  admin: up
  state: present
- type: ipv4
  remote_asn: 64803
  ip: 100.1.33.2
  admin: up
  state: present
- type: ipv4
  remote_asn: 64802
  ip: 100.1.17.2
  admin: up
  state: present
- type: ipv4
  remote_asn: 64804
  ip: 100.1.49.2
  admin: up
  state: present
- type: ipv6
  remote_asn: 64801
  ip: 2001:100:1:1::2
  admin: up
  address_family:
    - type: ipv4
      activate: false
      state: present
    - type: ipv6
      activate: true
      state: present
  state: present
- type: ipv6
  remote_asn: 64802
  ip: 2001:100:1:11::2
  admin: up
  address_family:
    - type: ipv4
      activate: false
      state: present
    - type: ipv6
      activate: true
      state: present
  state: present
- type: ipv6
  remote_asn: 64803
  ip: 2001:100:1:21::2
  admin: up
  address_family:
    - type: ipv4
      activate: false
      state: present
    - type: ipv6
      activate: true
      state: present
```

(continues on next page)

(continued from previous page)

```

    state: present
  - type: ipv6
    remote_asn: 64804
    ip: 2001:100:1:31::2
    admin: up
    address_family:
      - type: ipv4
        activate: false
        state: present
      - type: ipv6
        activate: true
        state: present
    state: present
dellos_snmp:
  snmp_community:
    - name: public
      access_mode: ro
      state: present

```

```

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

```

(continues on next page)

(continued from previous page)

```

        switchport: False
        ip_and_mask: 100.2.49.1/24
        ipv6_and_mask: 2001:100:2:31::1/64
        state_ipv6: present
dellos_bgp:
  asn: 64901
  router_id: 100.0.1.2
  best_path:
    as_path: multipath-relax
    as_path_state: present
    med:
      - attribute: missing-as-worst
        state: present
  neighbor:
    - type: ipv4
      remote_asn: 64801
      ip: 100.2.1.2
      admin: up
      state: present
    - type: ipv4
      remote_asn: 64802
      ip: 100.2.33.2
      admin: up
      state: present
    - type: ipv4
      remote_asn: 64803
      ip: 100.2.17.2
      admin: up
      state: present
    - type: ipv4
      remote_asn: 64804
      ip: 100.2.49.2
      admin: up
      state: present
    - type: ipv6
      remote_asn: 64801
      ip: 2001:100:2:1::2
      admin: up
      address_family:
        - type: ipv4
          activate: false
          state: present
        - type: ipv6
          activate: true
          state: present
      state: present
    - type: ipv6
      remote_asn: 64802
      ip: 2001:100:2:11::2
      admin: up
      address_family:
        - type: ipv4
          activate: false
          state: present
        - type: ipv6
          activate: true
          state: present

```

(continues on next page)



(continued from previous page)

```

    state: present
  - type: ipv6
    remote_asn: 64803
    ip: 2001:100:2:21::2
    admin: up
    address_family:
      - type: ipv4
        activate: false
        state: present
      - type: ipv6
        activate: true
        state: present
    state: present
  - type: ipv6
    remote_asn: 64804
    ip: 2001:100:2:31::2
    admin: up
    address_family:
      - type: ipv4
        activate: false
        state: present
      - type: ipv6
        activate: true
        state: present
    state: present
  state: present
dellos_snmp:
  snmp_community:
    - name: public
      access_mode: ro
      state: present

```

```

hostname: leaf1
ansible_ssh_user: xxxxxx
ansible_ssh_pass: xxxxxx
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:

```

(continues on next page)

(continued from previous page)

```

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

```

```

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:
    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

```

(continues on next page)

(continued from previous page)

```

    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
dellos_snmp:
  snmp_community:
    - name: public
      access_mode: ro
      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

```

(continues on next page)

(continued from previous page)

```

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

```

```

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
  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

```

(continues on next page)

(continued from previous page)

```

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

```

**Step 3**

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

```

**Step 4**

Execute the playbook:

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



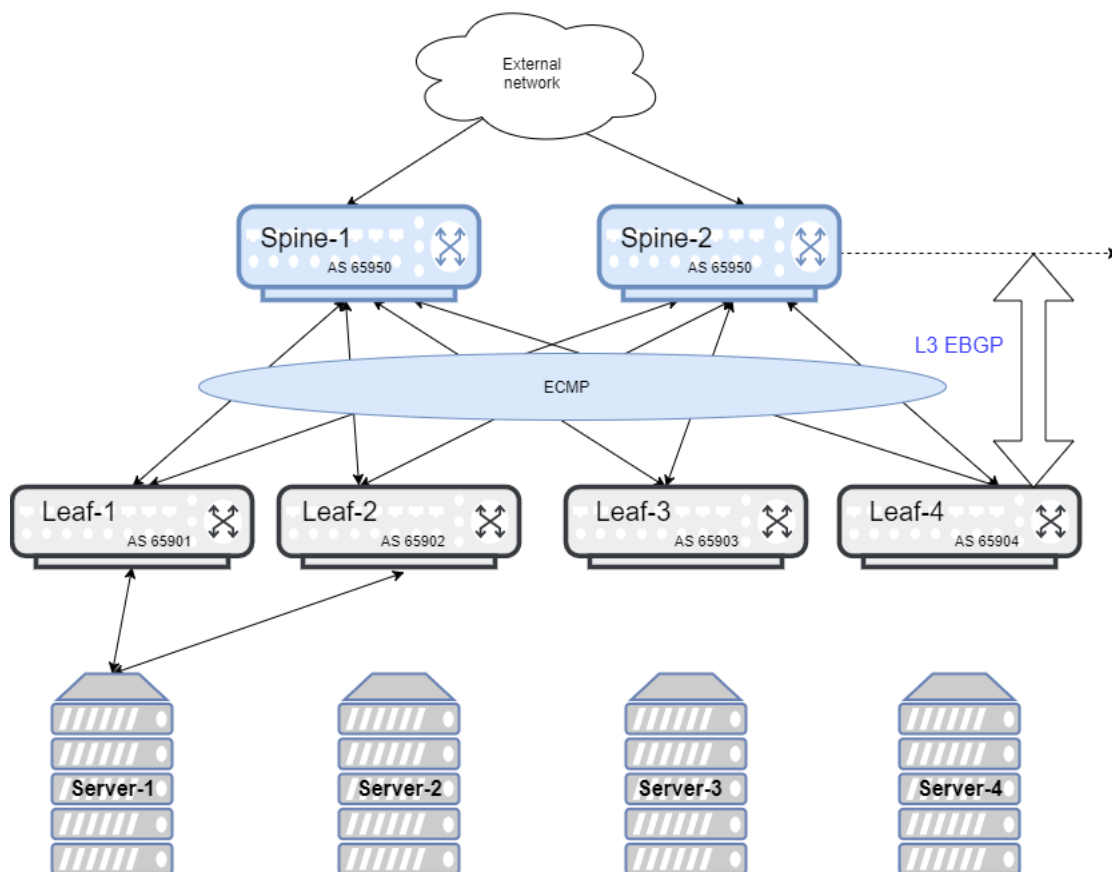


---

## Provisioning hot swap use case using Dell EMC Networking Ansible modules

---

This example use case topology includes a simple two-tier CLOS fabric with 2 spines and 4 leafs. These steps will show how you Spine 2 will 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 (4), 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

### 8.1.1 Part 1

Refer to the CLOS fabric example to configure a 6 node CLOS fabric with EBGp. Use the example and run the playbook.

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
```

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
    - type: ipv4
      remote_asn: 64802
      ip: 100.2.33.2
```

(continues on next page)

(continued from previous page)

```

    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

```

Create a playbook called `hot_swap_pre_step.yaml`:

```

---
- hosts: datacenter
  gather_facts: no
  connection: network_cli
  tasks:
    - name: Assembling configurations
      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:'
            #answer: 'admin'
          with_items: '{{copy_running_remote}}'
- hosts: datacenter
  connection: network_cli
  vars:

```

(continues on next page)

(continued from previous page)

```
build_dir: "/root/debug"
roles:
  - Dell-Networking.dellos-bgp
```

Run the playbook.

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

## 8.1.2 Part 2

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

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
```

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
```

(continues on next page)

(continued from previous page)

```
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"
```

Create a playbook called `waitfor_ecmp_path_delete.yaml`

**Note:** 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 configurations
      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] == ""
```

1. Execute the playbook.

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

### 8.1.3 Part 3

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
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
```

1. 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
```

Create a playbook called *hot\_swap\_post\_step.yaml*

```
---
- hosts: datacenter
  gather_facts: no
  connection: network_cli
  tasks:
    - name: Assembling configurations
      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'
            #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}}'
```

Execute the playbook:

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

---

# Install or upgrade devices running Dell EMC Networking OS10 using Ansible

---

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

The *dellos-image-upgrade* role uses the `dellos10_command` to install or upgrade the switch, then `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

Execute the playbook:

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



## CHAPTER 10

---

### Frequently asked questions

---

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

Q: Which version of Ansible supports Dell EMC Networking Ansible modules?

A: Ansible 2.2 and later.

Q: What are the minimum OS versions for Ansible support?

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

Q: What do the Dell EMC Networking Ansible roles provide?

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

Q: Do Dell EMC Networking Ansible roles work with Ansible Tower?

A: Yes, these roles work with Ansible Tower for management.

Q: Is there dnosX\_template module support for OS6/OS9/OS10?

A: No. Ansible has deprecated support for the template module, replacing it with the config module (see [Deprecations](#)).



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

### 11.1 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

### 11.2 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.

---

## 11.3 Release 1.0.0

This release introduces:

- Initial Ansible support for Dell EMC Networking 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](#))

You can submit issues for Dell EMC Networking modules at [Ansible Github Issues](#).

Submit issues for Dell EMC Networking 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

## 12.1 Contact

You can send general comments and feedback to [networking\\_devops\\_tools@dell.com](mailto:networking_devops_tools@dell.com).

## CHAPTER 13

---

### License

---

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the [License](#).

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.