# Ansible for SmartFabric OS10

Technical Note

H18530

**Abstract**

This guide introduces Ansible concepts and describes how to use Ansible for SmartFabric OS10. This document also provides Ansible playbook snippets for building data-center fabrics by using Ansible Collections.

**Dell Technologies Solutions**

December 2020

## Notes, cautions, and warnings

**NOTE:** A NOTE indicates important information that helps you make better use of your product.

**CAUTION: A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.**

**WARNING: A WARNING indicates a potential for property damage, personal injury, or death.**

# Contents

1

# Getting Started with Ansible

## Introduction to network automation

Networking has evolved drastically over the years—from VLANs and Spanning Tree CLI to software-defined API-based provisioning. Modern networking requires automation to :

- Build scalable infrastructure
- Build predictable infrastructure
- Minimize errors

Ansible is a simple IT automation engine for achieving configuration management and automation. It is an agentless solution that runs sets of instructions remotely over a Secure Shell (SSH) connection.

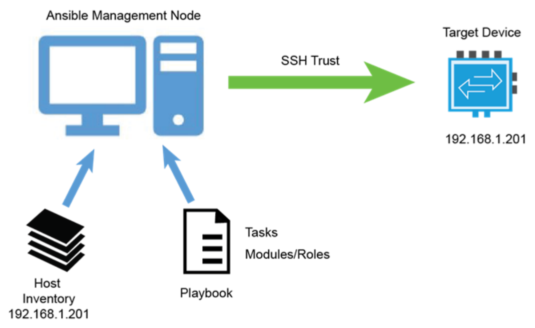


**Figure 1. Ansible basics**

This guide provides information about to how to configure Dell EMC PowerSwitch switches by using Dell EMC Networking Ansible modules and roles. It introduces Ansible Collections, which is a new model to package and deliver Ansible content, and provides configuration examples for leaf-spine architectures.

## Install Ansible

Install Ansible 2.10 or later to use Dell EMC Networking Ansible Collections, which then uses SSH by default to communicate with the end devices that you want to automate (managed nodes).

### Prerequisites

Requirements for the control node and managed nodes are:

- Control node — Python 2.7 or higher and Python 3.5 or higher
- Managed node—SSH to communicate remotely with OS10 switches

### Steps

1. Log in to your Ansible control node.
2. In the terminal window, run the following commands:

```
sudo apt update
sudo apt install software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

3. Verify that Ansible is installed:

   ```
   ansible --version
   ```

**Next steps**

For more information about installing Ansible on different platforms, see Installing Ansible in the *Ansible Installation Guide*.

## Ansible configuration file

You can adjust certain settings in Ansible by using `ansible.cfg`.

Run the following command to view the Ansible configuration file:

```
$ cat /etc/ansible/ansible.cfg
```

The stock configuration is sufficient for most users; however, in some cases, you might want to change the default values. For example, host key checking is turned on by default in Ansible 1.3 and later. To disable host key checking, set the value to False:

```
host_key_checking = False
```

## Ansible UI

Ansible Tower, which was previously called the AWX project, is a comprehensive web-based UI for Ansible. You can use it to centralize and control your IT infrastructure with a visual dashboard and REST API for Ansible. Tower includes real-time output of playbook runs, a new dashboard, and expanded cloud support.

Within Ansible Tower, playbook runs stream by in real time. As Ansible performs automation across your infrastructure, it displays plays and tasks as they are completed, per machine, and each success or failure, complete with output. With Ansible Tower, you can launch playbooks with a single click. It can prompt you for variables, let you choose from available secure credentials, and monitor the resulting deployments.

The following figure illustrates Ansible Tower:

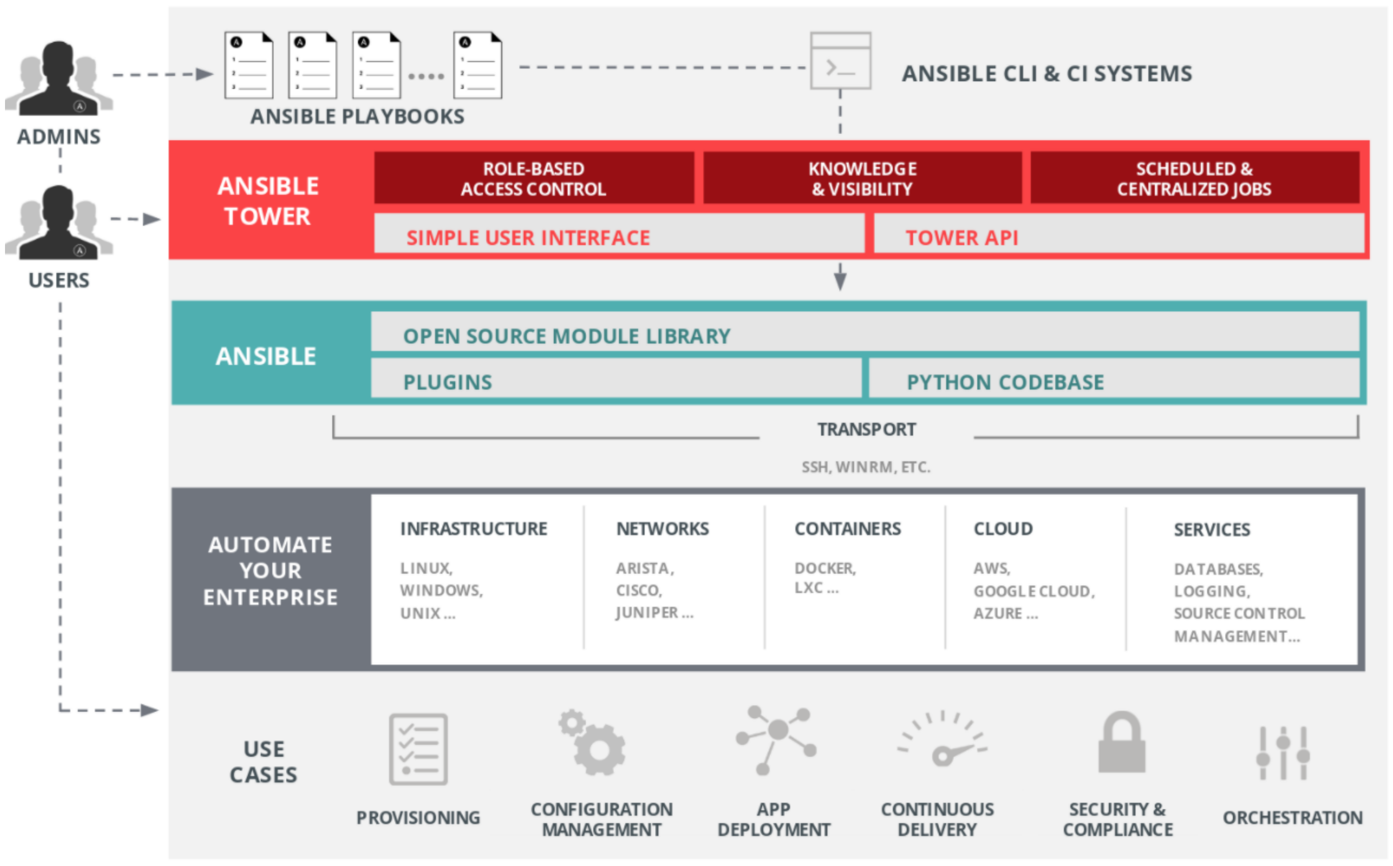


**Figure 2. Ansible Tower (Source: Red Hat)**

For more information about using Ansible Tower, see Red Hat Ansible Tower.

## Ansible inventory file

Ansible can configure multiple systems simultaneously. By default, the Ansible inventory is saved at `/etc/ansible/hosts`. To specify a different inventory file, use the `-i <path>` option.

The inventory file can be of many formats. Here is one example:

```
[inventory.yaml]

[Spine]
Spine1 ansible_host=192.168.1.201

[Leaf]
Leaf1 ansible_host=192.168.1.202
Leaf2 ansible_host=192.168.1.203

[datacenter:children]
[Spine]
[Leaf]

[datacenter:vars]
ansible_network_os=dellemc.os10.os10
ansible_ssh_user=admin
ansible_ssh_pass=admin
```

You can make one or more groups, as shown by `[Leaf]` in the preceding example. In the example, `Leaf1` specifies the host with IP address 192.168.1.202. You can specify a group of groups by using the `:children` suffix. In the preceding example, `datacenter` is a super group. You can include variables by specifying `vars:` or `:vars`. Ansible host and group variables provides more information about variables.

## Ansible host and group variables

Ansible uses variables to enable flexibility in playbooks and roles. Although you can store variables in the main inventory file, storing separate host and group variable files might help you organize the variable values more easily.

Host and group variable files must use YAML syntax. Most commonly used variables are the user-defined variables *host_vars* and *group_vars.*

Host variable files contain host-specific configuration variables and role variables, as shown in this example:

```
[leaf1.yaml]
hostname: Leaf1
ansible_ssh_user: admin
ansible_ssh_pass: admin
ansible_network_os: dellemc.os10.os10

os10_vlan:
    vlan 5:
      description: "Blue"
      tagged_members:
          - port: ethernet 1/1/32
            state: present
          - port: ethernet 1/1/31
            state: present
      untagged_members:
          - port: ethernet 1/1/30
            state: present
          - port: ethernet 1/1/29
            state: present
    vlan 10:
      description: "test"
```

## Ansible playbooks

Playbooks are the basis for simple configuration management and multi-machine deployments. They are designed to be human-readable and are expressed in YAML format.

Each playbook includes one or more plays in a list. The following example shows a playbook file that is designed to configure a VLAN on a switch:

```
[vlan_os10.yaml]
---
- hosts: Leaf
  connection: network_cli
  collections:
   - dellemc.os10
  roles:
   - os10_vlan
```

The following example shows how the playbook, by working with the inventory file, runs the play to create the VLAN:

```
$ ansible-playbook -i inventory.yaml vlan_os10.yaml
PLAY [Leaf] *******************************************************
TASK [Configure vlan on the Dell EMC OS10 Device]
*******************************************************
changed: [leaf2]
changed: [leaf1]
PLAY RECAP ********************************************************
leaf1                       : ok=2    changed=1    unreachable=0    failed=0
leaf2                       : ok=2    changed=1    unreachable=0    failed=0
```

The `PLAY RECAP` section confirms that the VLAN has been configured on the leaf1 and leaf2 switches, which belong to the `Leaf` group.

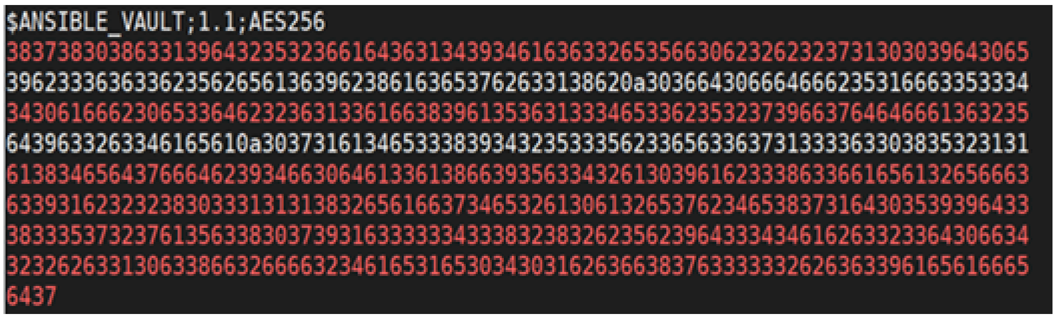
## Ansible password protection

Ansible Vault is an Ansible feature that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plain text in playbooks or roles. You can then distribute the vault files or put them in source control.

To enable Ansible Vault, edit files by using the command-line tool `ansible-vault` and a command-line flag (`--ask-vault-pass`, `--vault-password-file`, or `--vault-id`). Alternately, specify the location of a password file or edit your `ansible.cfg` file to command Ansible to always prompt for the password. These options require no command-line flag usage. Ansible Vault can encrypt any structured datafile used by Ansible. The advantage of variable-level encryption is that files are still easily legible even if they mix plain text and encrypted variables.

If you have existing files that you want to encrypt, use the `ansible-vault encrypt` command. This command can operate on multiple files at once:

```
ansible-vault encrypt main.yaml
```



```
$ANSIBLE_VAULT;1.1;AES256
38373830386331396432353236616436313439346163633265356630623262323731303039643065
396233363633623562656136396238616365376263313862 0a303664306664666235316663353334
34306166623065336462323631336166383961353631333465336235323739663764646661363235
6439633263346165610a303731613465333839343235333562336563363731333363303835323131
61383465643766646239346630646133613866393563343261303961623338633661656132656663
63393162323238303331313138326561663734653261306132653762346538373164303539396433
38333537323761356338303739316333333433383238326235623964333434616263323364306634
32326263313063386632666632346165316530343031626366383763333332626363396165616665
6437
```

Figure 3. Ansible Vault encrypted file

To permanently decrypt the files, run the `ansible-vault decrypt` command:

```
ansible-vault decrypt main.yaml
```

For more information about Ansible Vault, see Encrypting content with Ansible Vault in the *Ansible User Guide*.

# Dell EMC SmartFabric OS10

Dell EMC SmartFabric OS10 is a network operating system that supports multiple architectures and environments. OS10 allows multilayered disaggregation of network functionality.

OS10 also bundles industry-standard management, monitoring, and Layer 2 and Layer 3 networking stacks over CLI, SNMP, and REST interfaces. Users can choose their own third-party networking, monitoring, management, and orchestration applications. To develop scalable Layer 2 and Layer 3 networks, the SmartFabric OS delivers a modular and disaggregated solution in a single-binary image.

## Configure switch management

The OS10 management interface provides out-of-band management access to the network device. You can configure the management interface with DHCP, IPv4, or IPv6.

**Steps**

1. Configure the management interface in CONFIGURATION mode:

   ```
   OS10(config)#interface mgmt 1/1/1
   ```

2. If a static IP address is being used, disable the DHCP client operations in INTERFACE mode:

   ```
   OS10(conf-if-ma-1/1/1)#no ip address dhcp
   ```

By default, the DHCP client is enabled on the management interface.

3. Configure an IP address and mask on the management interface in INTERFACE mode:

```
OS10(conf-if-ma-1/1/1)#ip address 10.1.1.10/24
```

4. Enable the management interface in INTERFACE mode:

```
OS10(conf-if-ma-1/1/1)#no shutdown
```

5. Configure a management route:

```
OS10(config)#ip address 10.10.20.0/24 10.1.1.1
```

## RSA-SSH authentication login

Dell EMC PowerSwitch switches support SSH version 2.0 in combination with Rivest-Shamir-Adleman (RSA) encryption for authentication. RSA-SSH authentication permits the user to establish an SSH session without entering a password.

This functionality allows an Ansible script user to automatically launch an SSH session to a Dell EMC PowerSwitch, eliminating the potential security vulnerability of including a password in the script. Using RSA-SSH authentication requires creating a public key or private key pair on the host where the SSH session is launched. The host's public key is configured on the Dell EMC PowerSwitch switch, and the host's private key helps in authenticating the login to the switch.

### Configure an RSA-SSH login

Enable SSH password-less login and verify the SSH connection.

**Steps**

1. Generate the public/private key pair on the Ansible control node by running the following command:

```
$ ssh-keygen -t rsa
      Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa
Your public key has been saved in     /home/user/.ssh/id_rsa.pub
      $ cat /home/user/.ssh/id_rsa.pub
      ssh-rsa AAAAB3NzaC1yc2EAAAADA ...
```

2. Enable SSH password-less login, using the public key of an Ansible control node:

```
OS10(conf)#username admin sshkey “ssh-rsa AAAAB3Nz…”
```

The remote client is not prompted to enter a password.

3. Verify the SSH connection by running the following command:

```
$ ssh admin@10.1.1.10
```

## Linux subshell

The OS10 CLI is the software interface that you use to access a device running the software-from the console or through a network connection. The CLI is an OS10-specific command shell that runs on top of a Linux-based kernel.

By leveraging industry-standard tools and utilities, the CLI provides a powerful set of commands that you can use to monitor and configure devices running OS10. For example, to access the Linux shell:

```
OS10# system bash
```

2

# SmartFabric OS10 Ansible Collection, Modules, and Roles

This chapter presents the following topics:

## Ansible Content Collections

An Ansible Content Collection is a package of Ansible content.

For example:

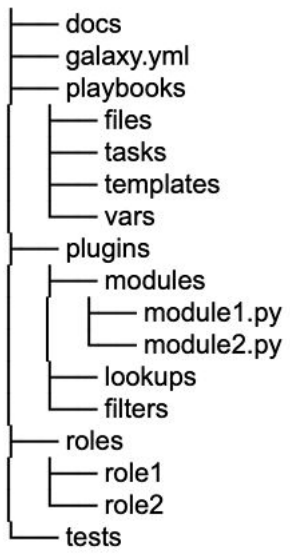


**Figure 4. Ansible Content Collection tree**

This format has a simple, predictable data structure:

- docs— Local documentation for the collection
- galaxy.yml—Source data or the MANIFEST.json that will be part of the collection package
- playbooks—Ansible playbooks
- tasks— Task list files for `include_tasks` and `import_tasks` use
- plugins— All Ansible plug-ins and modules in their own subdirectories
- modules—Ansible modules
- lookups—Look-up plug-ins
- filters—Jinja2 filter plug-ins
- connection—Connection plug-ins (required for a connection other than the default)
- roles—Ansible roles
- tests—Tests for the collection's content

## OS10 Ansible collection

The OS10 Ansible collection includes Ansible modules, plug-ins, and roles that are needed to provision and manage Dell EMC PowerSwitch platforms running Dell EMC SmartFabric OS10. The collection also includes sample playbooks and documentation that illustrate how the collection can be used.

Install the latest version of the OS10 collection from Ansible Galaxy by running the following command:

```
ansible-galaxy collection install dellemc.os10
```

## Ansible modules

Ansible modules are discrete units of code that can be used from the command line or in a playbook task. Users can also write their own modules. These modules can control system resources, such as services, packages, or files, and they can run system commands.

Collection core modules are:

- `os10_command.py`—Run commands on devices running OS10
- `os10_config.py`—Manage configuration on devices running OS10
- `os10_facts.py`—Collect facts from devices running OS10

For more information about Dell EMC Networking modules, see Ansible Network Collection for Dell EMC SmartFabric OS10.

## Ansible roles

Ansible roles are a structured way of grouping tasks, variables, files, and handlers that are stored in a standardized file structure.

Among the roles included in the collection are interface role, VLT role, BGP role, and so on. The `docs` directory in the collection includes documentation for each of the roles in the collection. The roles mentioned are summarized as follows:

- Interface role—The os10_interface role facilitates the configuration of interface attributes. It supports the configuration of administrative state, description, MTU, IP address, IP helper, and port mode.
- BGP role—The os10_bgp role facilitates the configuration of border gateway protocol (BGP) attributes and supports router ID, networks, neighbors, and maximum path configurations.
- VLT role—The os10_vlt role facilitates the configuration of the basics of virtual link trunking (VLT) to provide a loop-free topology.

Using roles:

- Helps simplify writing complex playbooks.
- Allows you to reuse common configurations without having to write tasks.
- Provides another layer of abstraction that can be useful for organizing playbooks.

3

# Ansible Playbook Examples

This chapter presents the following topics:

## Set hostnames and configure VLANs by using the Ansible OS10 module

This example shows how to configure a hostname and create VLANs for a leaf-spine fabric by using the `os10_config` module.

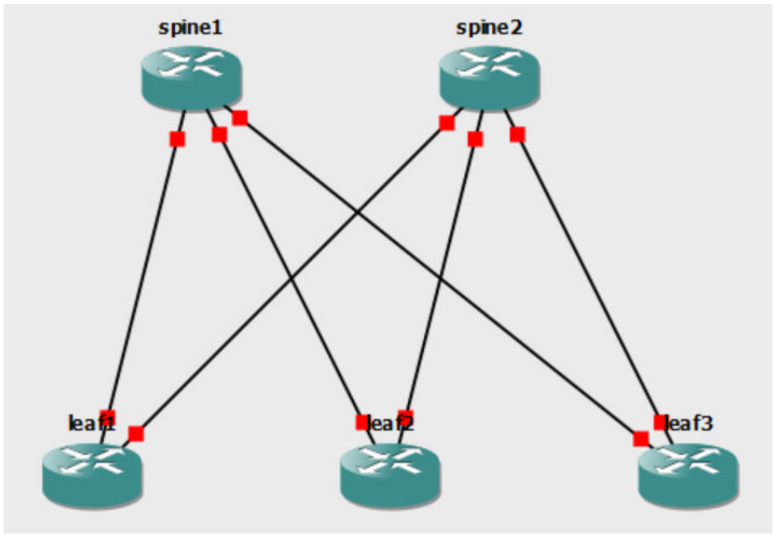


**Figure 5. Example 1 - Simple Leaf-Spine fabric**

### Create an inventory file

Create an inventory file with the IP addresses of the switches to be configured.

**Steps**

1. Create an inventory file:

   ```
   vim inventory.yaml
   ```

2. Edit the inventory file to include the IP addresses of the switches to be configured:

   ```
   [os10]
   leaf1 ansible_host=192.168.1.210 ansible_network_os=dellemc.os10.os10
   leaf2 ansible_host=192.168.1.211 ansible_network_os=dellemc.os10.os10
   spine1 ansible_host=192.168.1.212 ansible_network_os=dellemc.os10.os10
   ```

3. Save and exit the file:
   a. Press Esc.
   b. Type the following command: `:wq`
   c. Press Enter.

## Create leaf and spine variable files

Create variable files for leaf1, leaf2, leaf3, spine1, and spine2.

**Steps**

1. Create the leaf1 variable file:
   a. Enter the following command to create the leaf1 variable file:

      ```
      vim host_vars/leaf1.yaml
      ```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

      ```
      hostname: Leaf1
      ansible_ssh_user: admin
      ansible_ssh_pass: admin
      ```

   c. Save and exit the file:
      i. Press Esc.
      ii. Type the following command: `:wq`
      iii. Press Enter.
2. Create the leaf2 variable file:
   a. Enter the following command to create the leaf2 variable file:

      ```
      vim host_vars/leaf2.yaml
      ```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

      ```
      hostname: Leaf2
      ansible_ssh_user: admin
      ansible_ssh_pass: admin
      ```

   c. Save and exit the file:
      i. Press Esc.
      ii. Type the following command: `:wq`
      iii. Press Enter.
3. Create the leaf3 variable file:
   a. Enter the following command to create the leaf3 variable file:

      ```
      vim host_vars/leaf3.yaml
      ```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

      ```
      hostname: Leaf3
      ansible_ssh_user: admin
      ansible_ssh_pass: admin
      ```

   c. Save and exit the file:
      i. Press Esc.
      ii. Type the following command: `:wq`
      iii. Press Enter.
4. Create the spine1 variable file:
   a. Enter the following command to create the spine1 variable file:

      ```
      vim host_vars/spine1.yaml
      ```

b. After the file opens, type `i` to edit the file, and then enter the following commands:

```
hostname: Spine1
ansible_ssh_user: admin
ansible_ssh_pass: admin
```

c. Save and exit the file:
   i. Press Esc.
   ii. Type the following command: `:wq`
   iii. Press Enter.

5. Create the spine2 variable file:
   a. Enter the following command to create the spine2 variable file:

```
vim host_vars/spine2.yaml
```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

```
hostname: Spine2
ansible_ssh_user: admin
ansible_ssh_pass: admin
```

   c. Save and exit the file:
      i. Press Esc.
      ii. Type the following command: `:wq`
      iii. Press Enter.

## Create a playbook file

Create and run an Ansible playbook for setting the hostname and configuring VLANs.

**Steps**

1. Create the playbook:
   a. Create a playbook file by entering the following command:

```
vim hostname_vlan.yaml
```

   b. After the file opens, type `i` to edit the file, and enter the following commands:

```
---
- hosts: datacenter
  gather_facts: false
  connection: network_cli
  collections:
    - dellemc.os10

  tasks:
  - name: "Set hostname and configure VLAN 10"
    os10_config:
     commands:
       - hostname {{hostname}}
       - interface vlan 10
    register: set_hostname_vlan

  - debug: var=set_hostname_vlan
```

   c. Save and exit the file:
      i. Press Esc.
      ii. Type the following command: `:wq`
      iii. Press Enter.

2. Enter the following command to run the playbook file:

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

## Build a Layer 2 fabric

Virtual Link Trunking (VLT) is a Layer 2 multipathing technology that creates a link aggregation group (LAG) for a server, switch, or any device that supports LACP to two different upstream devices. The active-active connection improves and provides high availability to the southbound server/switch. The current data center designs use VLT technology to provide high availability of access layer switching to the southbound servers/switches.

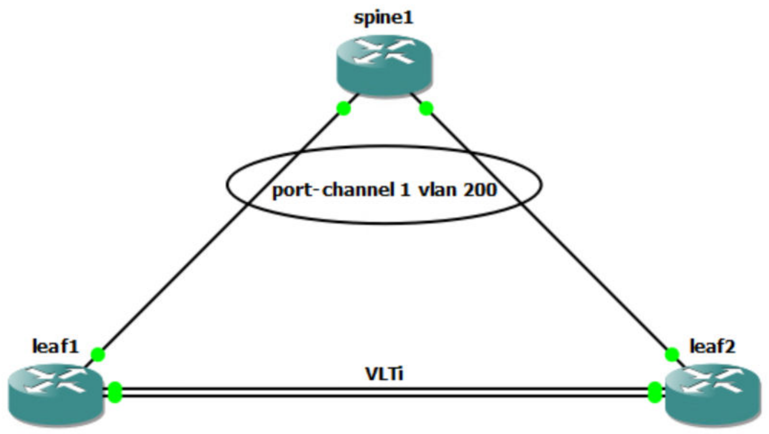


**Figure 6. Example 2 – Layer 2 VLT fabric**

In this playbook example, we use the following roles to configure VLT:

- `os10_interface`
- `os10_vlan`
- `os10_lag`
- `os10_vlt`

## Create an inventory file

Create an inventory file with the IP addresses of the switches to be configured.

**Steps**

1. Create an inventory file:

```
vim inventory.yaml
```

2. Edit the inventory file to include the IP addresses of the switches to be configured:

```
[os10]
leaf1 ansible_host=192.168.1.210 ansible_network_os=dellemc.os10.os10
leaf2 ansible_host=192.168.1.211 ansible_network_os=dellemc.os10.os10
spine1 ansible_host=192.168.1.212 ansible_network_os=dellemc.os10.os10
```

3. Save and exit the file:
   a. Press Esc.
   b. Type the following command: `:wq`
   c. Press Enter.

## Create leaf and spine variable files

Create leaf1, leaf2, and spine1 variable files for a Layer 2 network.

**Steps**

1. Create leaf and spine variable files:
    a. Enter the following command to create the leaf1 variable file:

    ```
    vim host_vars/leaf1.yaml
    ```

    b. After the file opens, type `i` to edit the file, and then enter the following commands:

    ```
    hostname: Leaf1
    os10_cfg_generate: True
    build_dir: /home/labuser/config

    ansible_ssh_user: admin
    ansible_ssh_pass: admin

    os10_interface:
      ethernet 1/1/1:
       admin: up
       switchport: False
      ethernet 1/1/2:
       admin: up
       switchport: False
      vlan 200:
       admin: up

    os10_vlan:
      vlan 200:
       tagged_members:
        - port: port-channel 1

    Os10_lag:
      Po 1:
       admin: up
       state: present
       type: dynamic
       channel_members:
        - port: ethernet 1/1/3
          mode: "active"

    os10_vlt:
      domain: 1
      discovery_intf: 1/1/1-1/1/2
      discovery_intf_state: present
      peer_routing: True
      vlt_mac: aa:aa:aa:aa:aa:aa
      vlt_peers:
        Po 1:
         peer_lag: 1
      state: present
    ```

    c. Save and exit the file:
        i. Press Esc.
        ii. Type the following command: `:wq`
        iii. Press Enter.
2. Create the Leaf2 variable file:
    a. Enter the following command to create the leaf2 variable file:

    ```
    vim host_vars/leaf2.yaml
    ```

b. After the file opens, type `i` to edit the file, and then enter the following commands:

```
hostname: Leaf2
os10_cfg_generate: True
build_dir: /home/labuser/config

ansible_ssh_user: admin
ansible_ssh_pass: admin

os10_interface:
  ethernet 1/1/1:
   admin: up
   switchport: False
  ethernet 1/1/2:
   admin: up
   switchport: False
  vlan 200:
   admin: up

os10_vlan:
  vlan 200:
   tagged_members:
    - port: port-channel 1

Os10_lag:
  Po 1:
   admin: up
   state: present
   type: dynamic
   channel_members:
    - port: ethernet 1/1/3
      mode: "active"

os10_vlt:
  domain: 1
  discovery_intf: 1/1/1-1/1/2
  discovery_intf_state: present
  peer_routing: True
  vlt_mac: aa:aa:aa:aa:aa:aa
  vlt_peers:
    Po 1:
      peer_lag: 1
  state: present
```

c. Save and exit the file:
   i. Press Esc.
   ii. Type the following command: `:wq`
   iii. Press Enter.

3. Create the spine1 variable file:
   a. Enter the following command to create the spine1 variable file:

```
vim host_vars/spine1.yaml
```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

```
hostname: Spine1
os10_cfg_generate: True
build_dir: /home/labuser/config

ansible_ssh_user: admin
ansible_ssh_pass: admin

os10_interface:
  vlan 200:
   admin: up

os10_vlan:
  vlan 200:
   tagged_members:
    - port: port-channel 1

os10_lag:
  Po 1:
   admin: up
   state: present
   type: dynamic
   channel_members:
    - port: ethernet 1/1/1
      mode: "active"
    - port: ethernet 1/1/2
      mode: "active"
```

c. Save and exit the file:
   i. Press Esc.
   ii. Type the following command: `:wq`
   iii. Press Enter.

## Create a playbook file

Create and run an Ansible playbook for the Layer 2 fabric.

**Steps**

1. Create a playbook file by entering the following command:

```
vim vlt.yaml
```

2. After the file opens, type `i` to edit the file, and then enter the following commands:

```
---
- hosts: os10
  connection: network_cli
  collections:
    - dellemc.os10
  roles:
    - os10_interface
    - os10_lag
    - os10_vlan
    - os10_vlt
```

3. Save and exit the file:
   a. Press Esc.
   b. Type the following command: `:wq`
   c. Press Enter.
4. Enter the following command to run the playbook file:

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

# Build a Layer 3 VXLAN EVPN fabric

This example demonstrates how to build a VXLAN EVPN fabric with Symmetric IRB and BGP Unnumbered by using OS10 roles for a leaf-spine fabric.

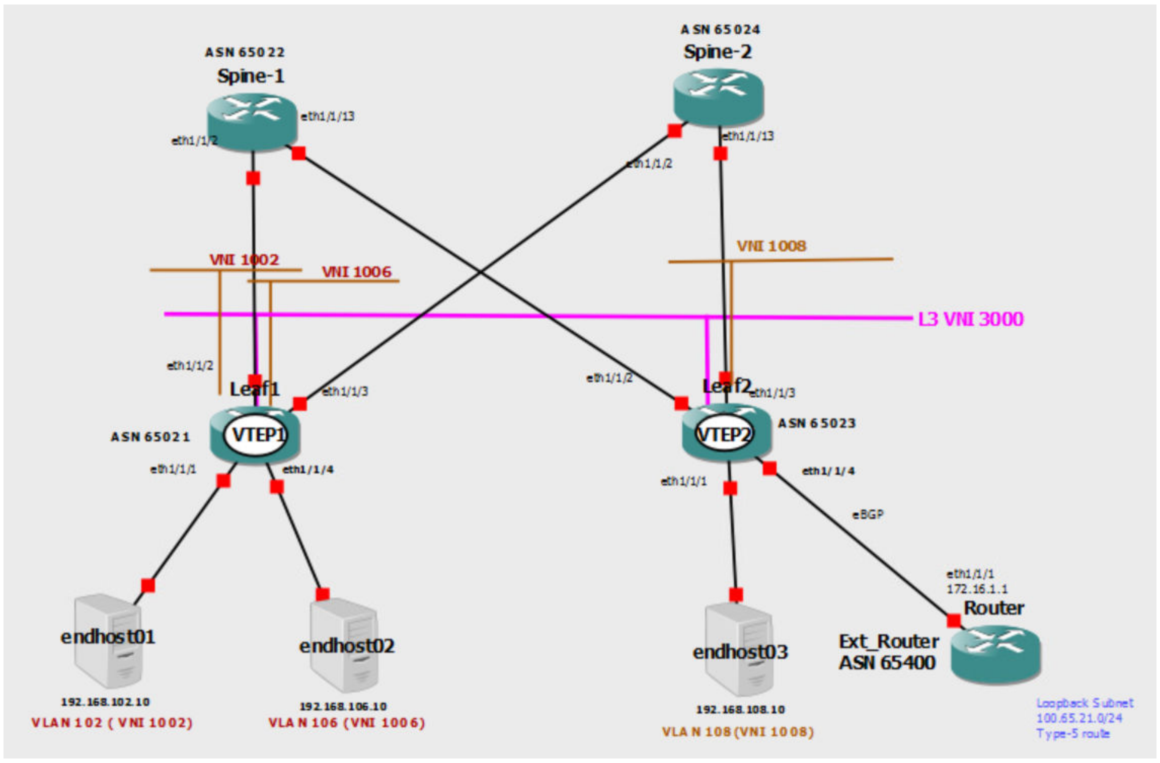


Figure 7. Example 3 – Layer 3 VXLAN EPVN fabric with Symmetric IRB

## Create an inventory file

Create an inventory file with the IP addresses of the switches to be configured.

### Steps

1. Create an inventory file:

```
vim inventory.yaml
```

2. Edit the inventory file to include the IP addresses of the switches to be configured:

```
leaf1 ansible_host=192.168.1.213
ansible_network_os=dellemc.os10.os10
leaf2 ansible_host=192.168.1.214
ansible_network_os=dellemc.os10.os10
spine1 ansible_host=192.168.1.215 ansible_network_os=dellemc.os10.os10
spine2 ansible_host=192.168.1.216 ansible_network_os=dellemc_.os10.os10

[spine]
spine-1
spine-2

[leaf]
leaf-1
leaf-2

[datacenter:children]
spine
leaf
```

3. Save and exit the file:
   a. Press Esc.
   b. Type the following command: `:wq`
   c. Press Enter.

## Create leaf and spine variable files

Create variable files for leaf1, leaf2, spine1, and spine2.

**Steps**

1. Create the leaf1 variable file:
   a. Enter the following command to create the leaf1 variable file:

   ```
   host_vars/leaf1.yaml
   ```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

   ```
   hostname: leaf1
   os10_cfg_generate: True
   build_dir: /home/labuser/config

   ansible_ssh_user: admin
   ansible_ssh_pass: admin

   os10_system:
     hostname: "Leaf1"

   os10_interface:
     loopback 0:
         admin: up
         ip_and_mask: 10.1.1.1/32
     virtual-network 1002:
       vrf: "TENANT1"
       admin: up
       ip_and_mask: "192.168.102.2/24"
       virtual_gateway_ip: "192.168.102.1"
     virtual-network 1006:
       vrf: "TENANT1"
       admin: up
       ip_and_mask: "192.168.106.2/24"
       virtual_gateway_ip: "192.168.106.1"
     ethernet 1/1/1:
       desc: "link to H1"
       admin: up
       portmode: trunk
     ethernet 1/1/2:
       desc: "link to SPINE1"
       admin: up
       switchport: False
       mtu: 9216
       suppress_ra: absent
       min_ra: 3
       max_ra: 4
     ethernet 1/1/3:
       desc: "link to SPINE2"
       admin: up
       switchport: False
       mtu: 9216
       suppress_ra: absent
       min_ra: 3
       max_ra: 4
     ethernet 1/1/4:
       desc: "link to H2"
       admin: up
       portmode: trunk

   os10_vrf:
     vrfdetails:
      - vrf_name: "TENANT1"
        state: "present"

   os10_bgp:
     asn: 65021
     router_id: 100.1.1.1
   ```

```
  ipv4_network:
    - address: 10.1.1.1/32
      state: present
  neighbor:
    - type: ipv4
      interface: ethernet1/1/2
      send_community:
       - type: extended
         state: present
      address_family:
        - type: "l2vpn"
          activate: true
          state: present
      admin: up
      state: present
    - type: ipv4
      interface: ethernet1/1/3
      send_community:
       - type: extended
         state: present
      address_family:
        - type: "l2vpn"
          activate: true
          state: present
      admin: up
      state: present
  state: present

os10_vxlan:
    anycast_gateway_mac: "00:01:01:01:01:01"
    nve:
      source_interface: 0
      state: "present"
    evpn:
      autoevi: True
      rmac: 00:00:01:02:03:04
      vrf:
       - name: "TENANT1"
         vni: 3000
         route_target:
          - type: "manual"
            asn_value: "3000:3000"
            route_target_type: "both"
            state: "present"
    virtual_network:
      virtual_net:
        - id: 1002
          member_interface:
            - ifname: "ethernet 1/1/1"
              type: "tagged"
              vlanid: 102
              state: "present"
          vxlan_vni:
            id: 1002
            state: "present"
          state: "present"
        - id: 1006
          member_interface:
            - ifname: "ethernet 1/1/4"
              type: "tagged"
              vlanid: 106
              state: "present"
          vxlan_vni:
            id: 1006
            state: "present"
          state: "present"
```

2. Create the leaf2 variable file:
    a. Enter the following command to create leaf2 variable file:

```
vim host_vars/ leaf2.yaml
```

b. After the file opens, type `i` to edit the file, and then enter the following commands:

```
hostname: leaf2
os10_cfg_generate: True
build_dir: /home/labuser/config

ansible_ssh_user: admin
ansible_ssh_pass: admin

os10_system:
  hostname: "Leaf2"

os10_interface:
  loopback 0:
    admin: up
    ip_and_mask: 10.2.1.1/32
  virtual-network 1008:
    vrf: "TENANT1"
    admin: up
    ip_and_mask: "192.168.108.2/24"
    virtual_gateway_ip: "192.168.108.1"
  ethernet 1/1/1:
    desc: "link to H3"
    admin: up
    portmode: trunk
  ethernet 1/1/2:
    desc: "link to SPINE1"
    admin: up
    switchport: False
    mtu: 9216
    suppress_ra: absent
    min_ra: 3
    max_ra: 4
  ethernet 1/1/3:
    desc: "link to SPINE2"
    admin: up
    switchport: False
    mtu: 9216
    suppress_ra: absent
    min_ra: 3
    max_ra: 4
  ethernet 1/1/4:
    desc: "link to ext-router"
    switchport: False
    vrf: "TENANT1"
    admin: up
    ip_and_mask: 172.16.1.2/24

os10_vrf:
  vrfdetails:
   - vrf_name: "TENANT1"
     state: "present"

os10_bgp:
  asn: 65023
  router_id: 100.1.1.1
  ipv4_network:
    - address: 10.2.1.1/32
      state: present
  neighbor:
    - type: ipv4
      interface: ethernet1/1/2
      send_community:
       - type: extended
         state: present
      address_family:
        - type: "l2vpn"
          activate: true
          state: present
      admin: up
      state: present
    - type: ipv4
      interface: ethernet1/1/3
```

```
       send_community:
        - type: extended
          state: present
       address_family:
         - type: "l2vpn"
           activate: true
           state: present
       admin: up
       state: present

   vrf:
    name: "TENANT1"
    address_family:
      type: "ipv4"
      redistribute:
       - route_type: "l2vpn"
         address_type: ipv4
         state: present
    neighbor:
      - type: ipv4
        ip: "172.16.1.1"
        remote_asn: 65400
        state: present
        admin: up

os10_vxlan:
    anycast_gateway_mac: "00:01:01:01:01:01"
    nve:
      source_interface: 0
      state: "present"
    evpn:
      autoevi: True
      rmac: 00:00:01:02:03:05
      vrf:
       - name: "TENANT1"
         vni: 3000
         adv_ipv4:
          - type: "bgp"
            state: "present"
         route_target:
          - type: "manual"
            asn_value: "3000:3000"
            route_target_type: "both"
            state: "present"
    virtual_network:
      virtual_net:
        - id: 1008
          member_interface:
            - ifname: "ethernet 1/1/1"
              type: "tagged"
              vlanid: 108
              state: "present"
          vxlan_vni:
            id: 1008
            state: "present"
          state: "present"
```

3. Create the spine1 variable file:
   a. Enter the following command to create the spine1 variable file:

   ```
   vim host_vars/spine1.yaml
   ```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

   ```
   hostname: spine1
   os10_cfg_generate: True
   build_dir: /home/labuser/config

   ansible_ssh_user: admin
   ansible_ssh_pass: admin
   ```

```
os10_system:
  hostname: "Spine1"

os10_interface:
  loopback 1:
    admin: up
    ip_and_mask: 10.3.1.1/32
  ethernet 1/1/2:
    desc: "link to Leaf1"
    admin: up
    switchport: False
    mtu: 9216
    suppress_ra: absent
    min_ra: 3
    max_ra: 4
  ethernet 1/1/3:
    desc: "link to Leaf2"
    admin: up
    switchport: False
    mtu: 9216
    suppress_ra: absent
    min_ra: 3
    max_ra: 4

os10_bgp:
  asn: 65022
  router_id: 100.3.1.1
  ipv4_network:
   - address: 10.3.1.1/32
     state: present
  neighbor:
   - type: "peergroup"
     name: "ebgp_session"
     send_community:
       - type: extended
         state: present
     address_family:
       - type: "l2vpn"
         activate: true
         state: present
     state: present
   - type: ipv4
     interface: ethernet1/1/2
     peergroup: ebgp_session
     peergroup_type: ebgp
     admin: up
     state: present
   - type: ipv4
     interface: ethernet1/1/3
     peergroup: ebgp_session
     peergroup_type: ebgp
     admin: up
     state: present
  state: present
```

4. Create the spine2 variable file:
   a. Enter the following command to create the spine2 variable file:

   ```
   vim host_vars/spine2.yaml
   ```

   b. After the file opens, type `i` to edit the file, and then enter the following commands:

   ```
   hostname: spine2
   os10_cfg_generate: True
   build_dir: /home/labuser/config

   ansible_ssh_user: admin
   ansible_ssh_pass: admin

   os10_system:
   ```

```
  hostname: "Spine2"

os10_interface:
  loopback 1:
    admin: up
    ip_and_mask: 10.4.1.1/32
  ethernet 1/1/2:
    desc: "link to Leaf1"
    admin: up
    switchport: False
    mtu: 9216
    suppress_ra: absent
    min_ra: 3
    max_ra: 4
  ethernet 1/1/3:
    desc: "link to Leaf2"
    admin: up
    switchport: False
    mtu: 9216
    suppress_ra: absent
    min_ra: 3
    max_ra: 4

os10_bgp:
  asn: 65024
  router_id: 100.4.1.1
  ipv4_network:
   - address: 10.4.1.1/32
     state: present
  neighbor:
   - type: "peergroup"
     name: "ebgp_session"
     send_community:
       - type: extended
         state: present
     address_family:
       - type: "l2vpn"
         activate: true
         state: present
     state: present
   - type: ipv4
     interface: ethernet1/1/2
     peergroup: ebgp_session
     peergroup_type: ebgp
     admin: up
     state: present
   - type: ipv4
     interface: ethernet1/1/3
     peergroup: ebgp_session
     peergroup_type: ebgp
     admin: up
     state: present
  state: present
```

## Create a playbook

Create and run an Ansible playbook for the VXLAN EVPN fabric.

### Steps

1. Create a playbook file by entering the following command:

```
vim vxlan.yaml
```

2. After the file opens, type `i` to edit the file, and then enter the following commands:

```
hosts: datacenter
connection: network_cli
```

```
collections:
 - dellemc.os10
roles:
 - os10_system
 - os10_vrf
 - os10_interface
 - os10_bgp
 - os10_vxlan
```

3. Enter the following command to run the playbook file:

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

4

# References

## Dell Technologies

Ansible Network Automation Hands-on Lab

## Ansible

Ansible Galaxy

GitHub Repo

Ansible Installation Guide

Network Debug and Troubleshooting Guide

Ansible Automation Hub

## Support and feedback

For technical support, visit http://www.dell.com/support or call (USA) 1-800-945-3355.

We encourage readers to provide feedback on the quality and usefulness of this publication by sending an email to Feedback-Ansible-Dell-Networking@Dell.com.