



The bridge to possible

# Infrastructure as Code for NXOS and NDFC with Ansible

Matt Tarkington, Technical Leader  
Mike Wiebe, Technical Leader

BRKDCN-2946

CISCO *Live!*

#CiscoLive

# Cisco Webex App

## Questions?

Use Cisco Webex App to chat with the speaker after the session

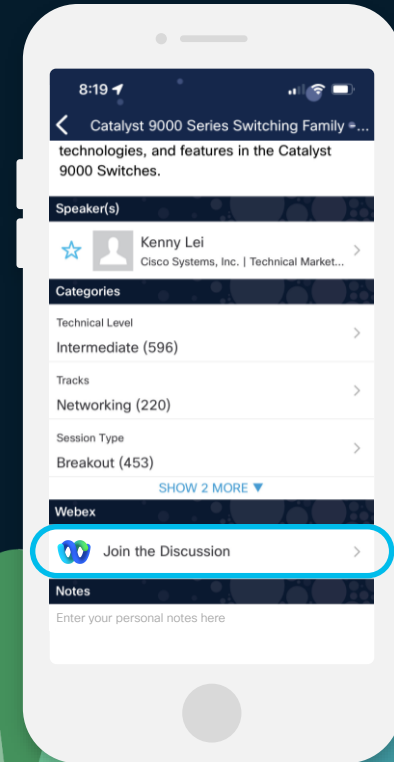
## How

- 1 Find this session in the Cisco Live Mobile App
- 2 Click “Join the Discussion”
- 3 Install the Webex App or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 7, 2024.

**CISCO** *Live!*

<https://ciscolive.ciscoevents.com/ciscolivebot/#BRKDCN-2946>

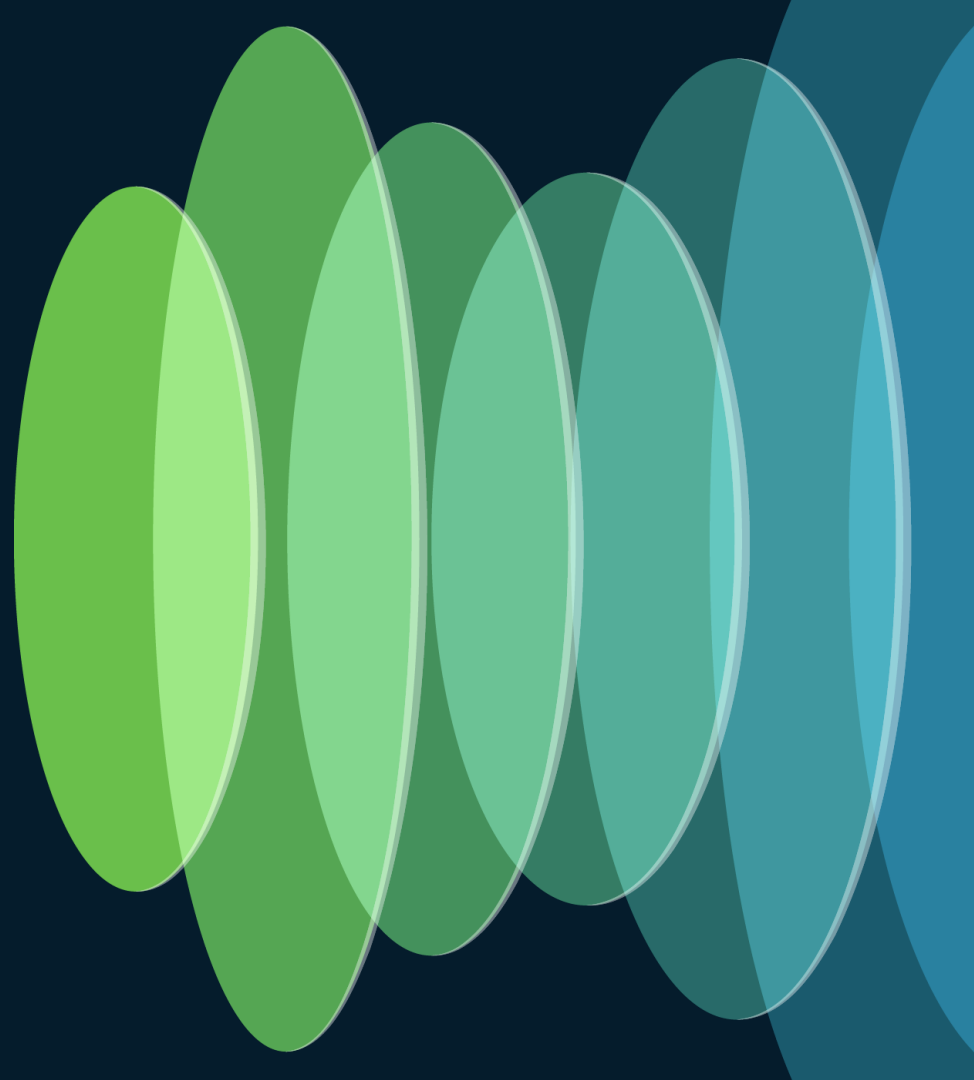




# Agenda

- What is Infrastructure as Code?
- Infrastructure as Code with NXOS and Ansible
- Infrastructure as Code with NDFC and Ansible
- Start Your IaC Journey!

# What is Infrastructure as Code?

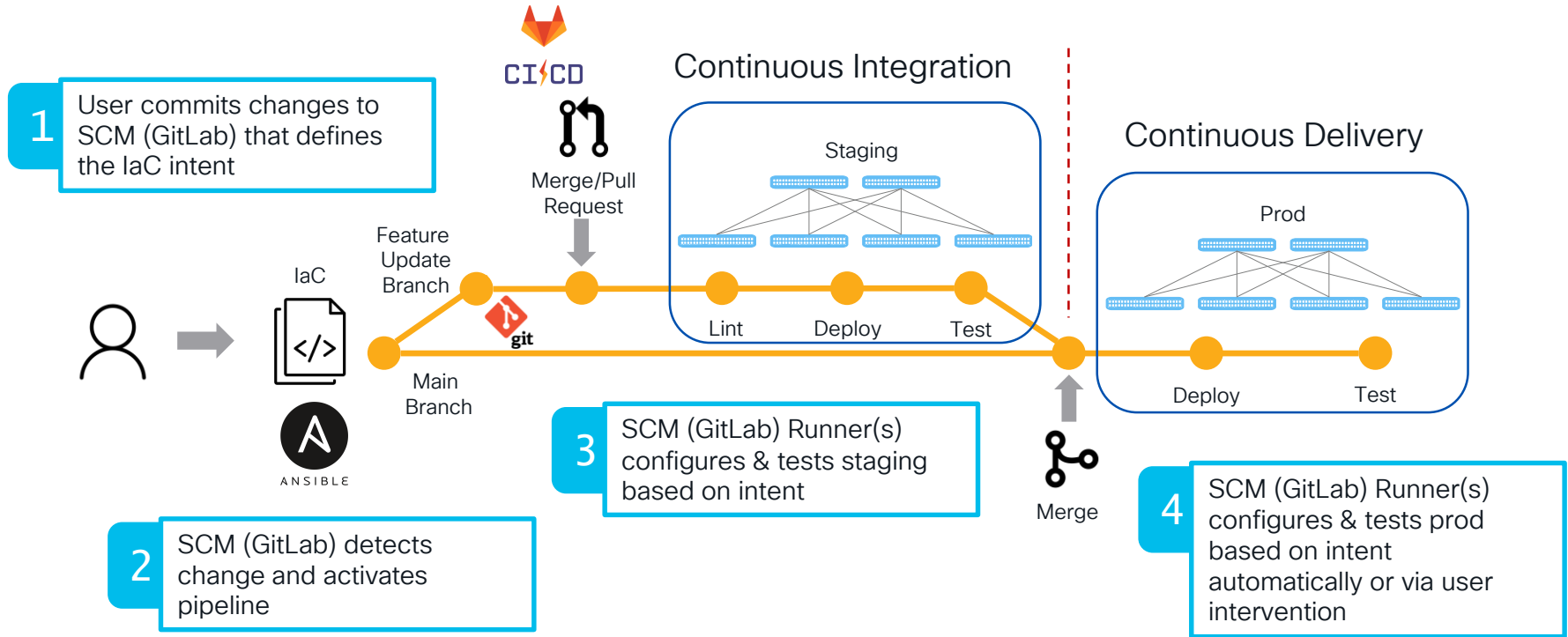


# Infrastructure as Code for Network Ops

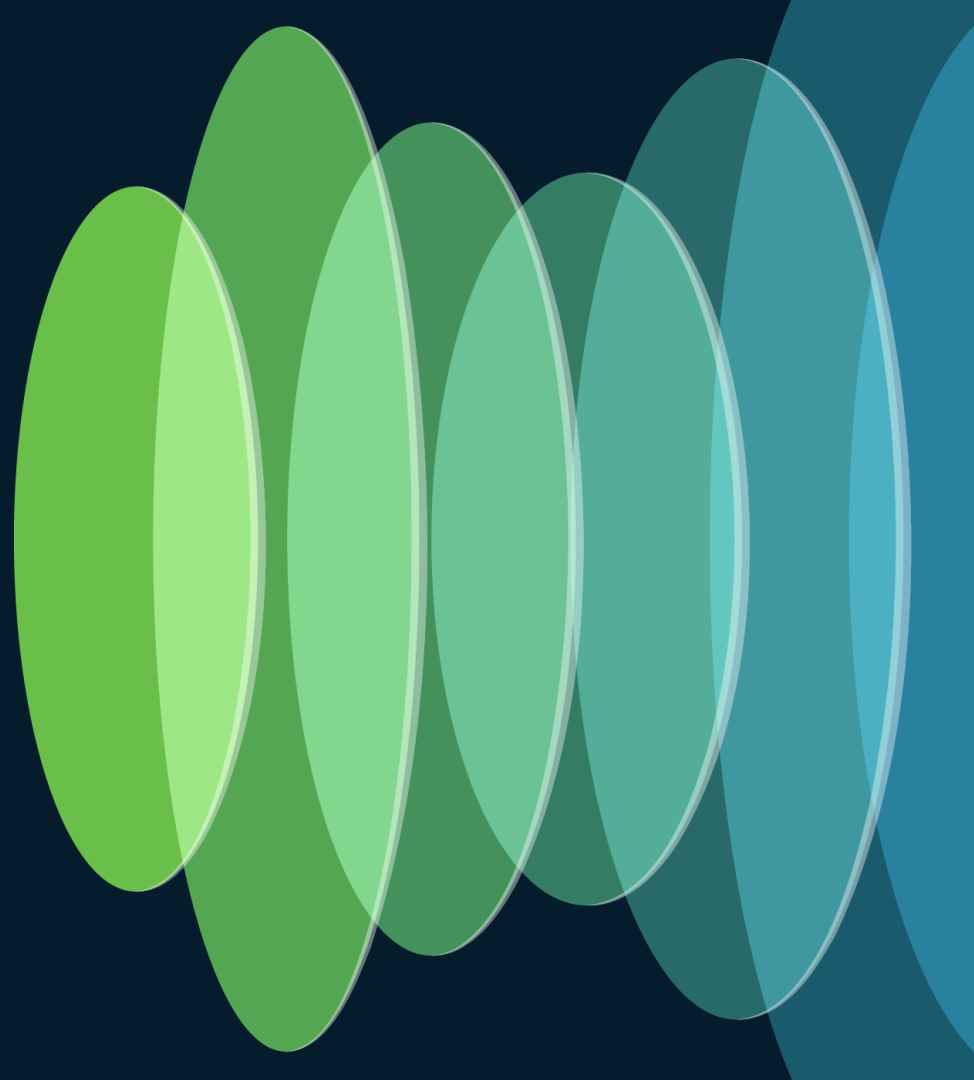
- Using “code” to provision and/or manage infrastructure
- Infrastructure as Code is not specific to a particular automation engine or specific programming language
- The intended configuration state of network devices are sourced from source code management (git) instead of the devices themselves



# Infrastructure as Code for Network Ops



# Infrastructure as Code with NXOS and Ansible



# What is Ansible?

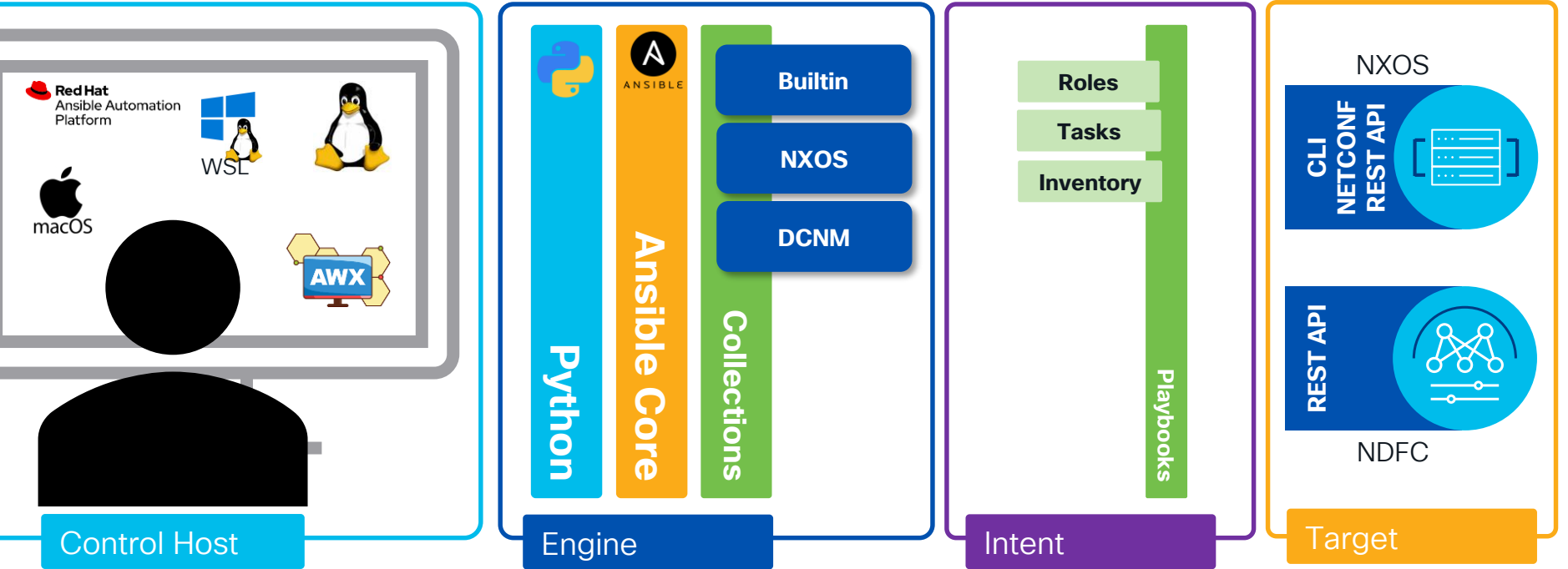


ANSIBLE

- Automation / Configuration / Orchestration tool
- Open Source
- Agentless Push Model
- Produces the same results no matter how many times it is executed\*
- No programming knowledge required
- Requires only data-structure manipulation knowledge
- Network CLI and REST API interaction

\*idempotent

# What makes up Ansible?





# Python Virtual Environments

- You should use a **virtual environment**
- Allows for installing Ansible inside a contained area with specific version of Python
- Makes it possible to run different Python scripts that require different versions of Python and libraries
- Detailed steps beyond scope of this session

# pyenv

- pyenv is the best mechanism to control python virtual environments
- Allows control of python version to execute independent of system version
- pyenv virtualenv also needed

Install instructions:

<https://github.com/pyenv/pyenv/wiki>  
<https://github.com/pyenv/pyenv-virtualenv>

**CISCO** *Live!*

- 1 install a version of python

```
% pyenv install 3.9.11
```

- 2 create virtual environment

```
% pyenv virtualenv 3.9.11 ansible
```

- 3 create directory for ansible development

```
% mkdir my_ansible_dir
```

- 4 Set pyenv virtual environment

```
% pyenv local ansible
```

# Installing Ansible

```
% pip install ansible-core
```

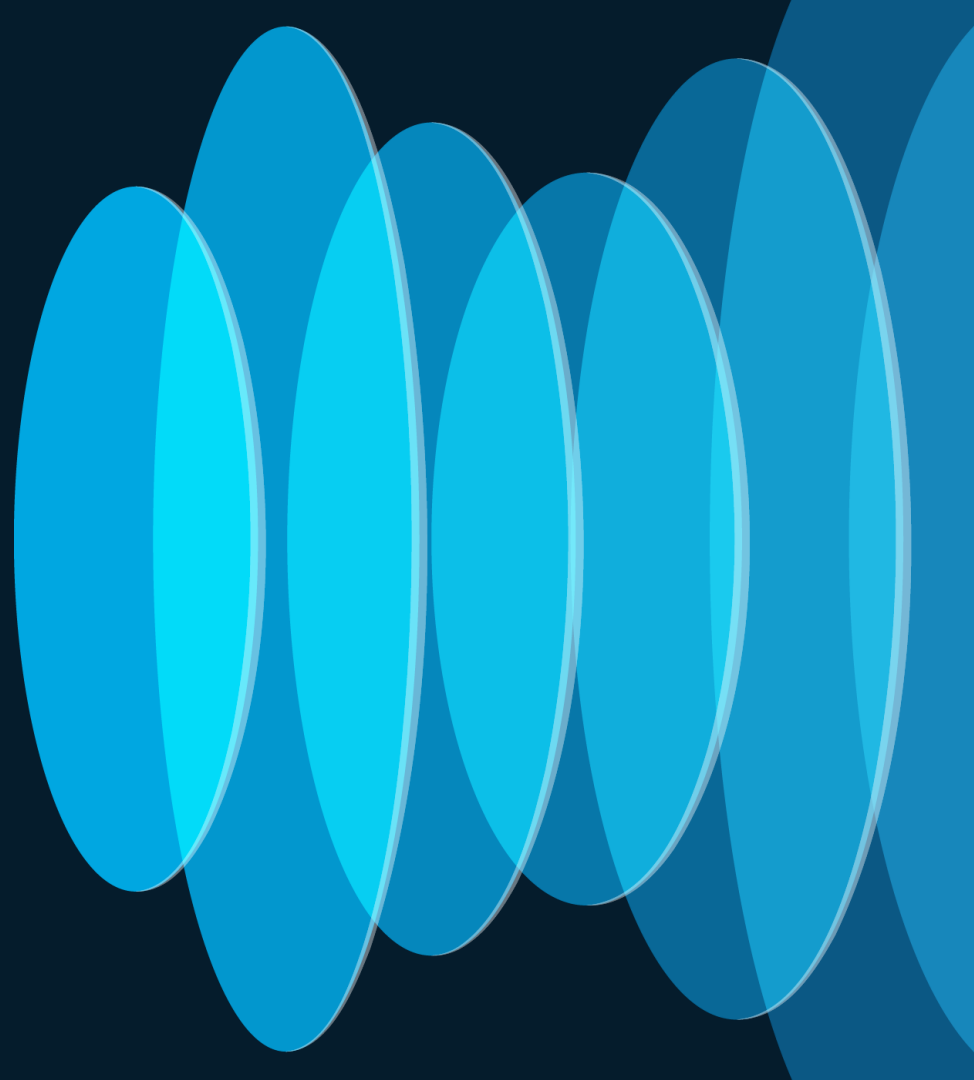
- Installs only the **core components**
- Collections must be installed by you
- Smaller footprint and more control
- Assures install of latest collection version released!

```
% pip install ansible
```

- “batteries included”
- Installs community-curated selection of Ansible Collections
- Complete package but larger footprint on filesystem
- Might not install the latest version of a desired collection!

[https://docs.ansible.com/ansible/latest/installation\\_guide/intro\\_installation.html](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html)

# Ansible Collections



# Ansible Collections

- Introduced in Ansible 2.9
- Uses ***Ansible Galaxy*** as the delivery vehicle
- Contains modules, plugins, filters
- Collections not related to Ansible release schedules
  - Allows vendor flexibility in relation to product releases


```
% ansible-galaxy collection install cisco.nxos cisco.dcnm
```

NXOS - <https://galaxy.ansible.com/cisco/nxos>

NDFC - <https://galaxy.ansible.com/cisco/dcnm>

# Ansible.Builtin

- Modules perform specific task like set facts (variables), import roles, tasks, vars, and more
- Includes many filters for working with data sets
- Actively maintained by RedHat

 Collections in the Ansible Namespace
Ansible.Builtin <ul style="list-style-type: none"> <li>Description</li> <li>Communication</li> <li>Plugin Index</li> </ul>
Ansible.Netcommon
Ansible.Posix
Ansible.Utilis
Ansible.Windows
Collections in the Arista Namespace
Collections in the Awx Namespace
Collections in the Azure Namespace
Collections in the Check_point Namespace
Collections in the Chocolatey Namespace
Collections in the Cisco Namespace
Collections in the Cloud Namespace
Collections in the Cloudscale_ch Namespace
Collections in the Community Namespace
Collections in the Containers Namespace
Collections in the Cyberark Namespace
Collections in the Dellemc Namespace
Collections in the F5networks Namespace
Collections in the Fortinet Namespace
Collections in the Frr Namespace
Collections in the Gluster Namespace
Collections in the Google Namespace
Collections in the Grafana Namespace
Collections in the Hetzner Namespace

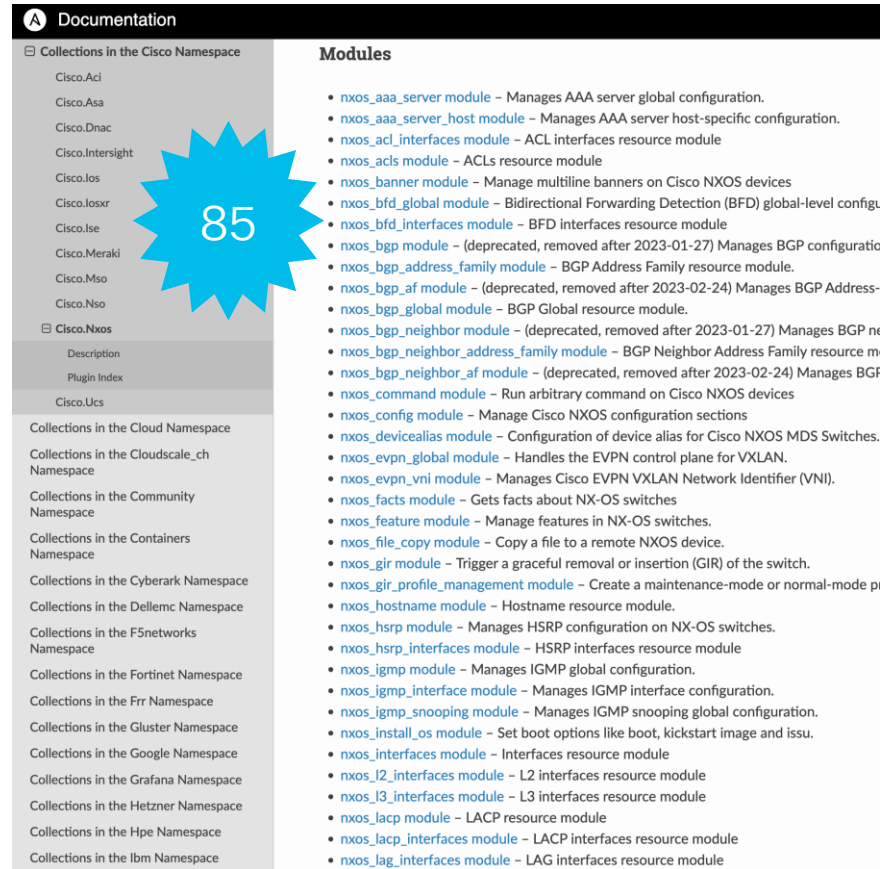
## Modules

- **add\_host module** – Add a host (and alternatively a group) to the ansible-playbook in-mem
- **apt module** – Manages apt-packages
- **apt\_key module** – Add or remove an apt key
- **apt\_repository module** – Add and remove APT repositories
- **assemble module** – Assemble configuration files from fragments
- **assert module** – Asserts given expressions are true
- **async\_status module** – Obtain status of asynchronous task
- **blockinfile module** – Insert/update/remove a text block surrounded by marker lines
- **command module** – Execute commands on targets
- `<snip>`
- **import\_role module** – Import a role into a play
- **import\_tasks module** – Import a task list
- **include module** – Include a task list
- **include\_role module** – Load and execute a role
- **include\_tasks module** – Dynamically include a task list
- **include\_vars module** – Load variables from files, dynamically within a task
- **iptables module** – Modify iptables rules
- **known\_hosts module** – Add or remove a host from the `known_hosts` file
- **lineinfile module** – Manage lines in text files
- **meta module** – Execute Ansible 'actions'
- **package module** – Generic OS package manager
- **package\_facts module** – Package information as facts
- **pause module** – Pause playbook execution
- **ping module** – Try to connect to host, verify a usable python and return `.pong` on success
- **pip module** – Manages Python library dependencies
- **raw module** – Executes a low-down and dirty command
- **reboot module** – Reboot a machine
- **replace module** – Replace all instances of a particular string in a file using a back-referen
- **rpm\_key module** – Adds or removes a gpg key from the rpm db
- **script module** – Runs a local script on a remote node after transferring it
- **service module** – Manage services
- **service\_facts module** – Return service state information as fact data
- **set\_fact module** – Set host variable(s) and fact(s).
- **set\_stats module** – Define and display stats for the current ansible run
- **setup module** – Gathers facts about remote hosts
- **shell module** – Execute shell commands on targets

<https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html>

# Ansible Cisco.Nxos Collection Modules

- Modules perform specific task like configure vlans, interfaces, OSPF, BGP, and more
- Documentation provides usage details, required variables, default variables, etc
- Actively maintained by RedHat with Cisco support



**Documentation**

☐ Collections in the Cisco Namespace

- Cisco.Aci
- Cisco.Asa
- Cisco.Dnac
- Cisco.Intersight
- Cisco.Ios
- Cisco.Iosxr
- Cisco.Ise
- Cisco.Meraki
- Cisco.Mso
- Cisco.Nso
- ☑ Cisco.Nxos
- Description
- Plugin Index
- Cisco.Ucs

Collections in the Cloud Namespace

Collections in the Cloudscale\_ch Namespace

Collections in the Community Namespace

Collections in the Containers Namespace

Collections in the Cyberark Namespace

Collections in the Dell EMC Namespace

Collections in the F5 Networks Namespace

Collections in the Fortinet Namespace

Collections in the Frr Namespace

Collections in the Gluster Namespace

Collections in the Google Namespace

Collections in the Grafana Namespace

Collections in the Hetzner Namespace

Collections in the Hpe Namespace

Collections in the Ibm Namespace

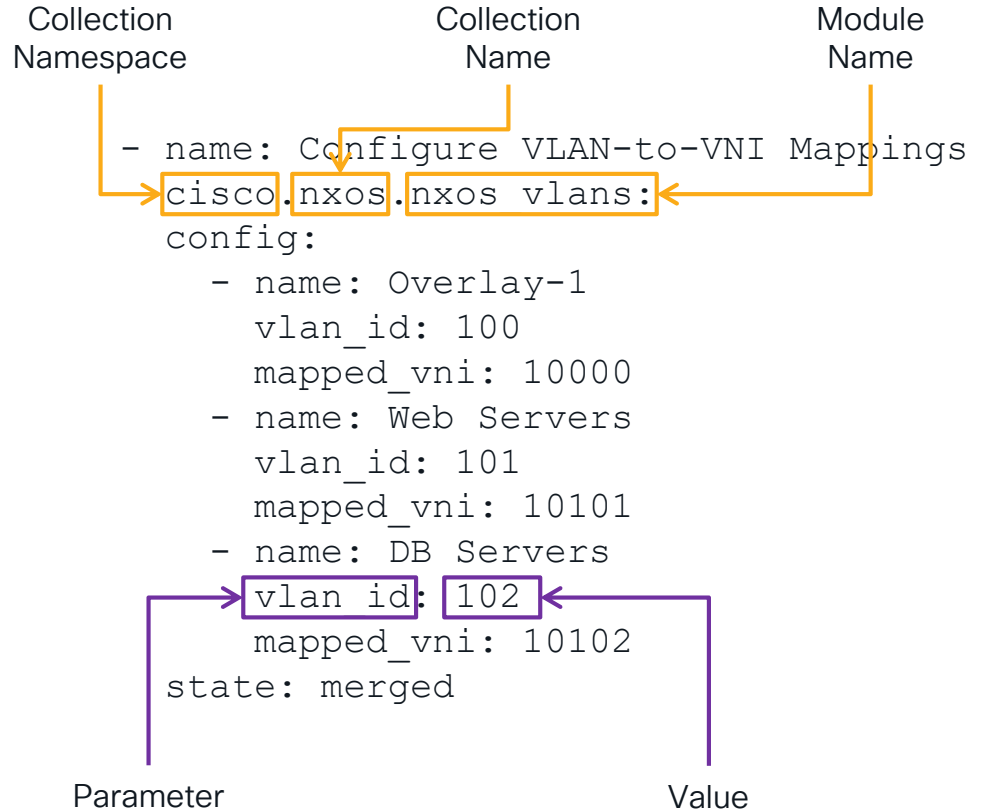
### Modules

- [nxos\\_aaa\\_server module](#) – Manages AAA server global configuration.
- [nxos\\_aaa\\_server\\_host module](#) – Manages AAA server host-specific configuration.
- [nxos\\_acl\\_interfaces module](#) – ACL interfaces resource module
- [nxos\\_acls module](#) – ACLs resource module
- [nxos\\_banner module](#) – Manage multiline banners on Cisco NXOS devices
- [nxos\\_bfd\\_global module](#) – Bidirectional Forwarding Detection (BFD) global-level configuration
- [nxos\\_bfd\\_interfaces module](#) – BFD interfaces resource module
- [nxos\\_bgp module](#) – (deprecated, removed after 2023-01-27) Manages BGP configuration
- [nxos\\_bgp\\_address\\_family module](#) – BGP Address Family resource module.
- [nxos\\_bgp\\_af module](#) – (deprecated, removed after 2023-02-24) Manages BGP Address Family resource module.
- [nxos\\_bgp\\_global module](#) – BGP Global resource module.
- [nxos\\_bgp\\_neighbor module](#) – (deprecated, removed after 2023-01-27) Manages BGP Neighbor resource module.
- [nxos\\_bgp\\_neighbor\\_address\\_family module](#) – BGP Neighbor Address Family resource module
- [nxos\\_bgp\\_neighbor\\_af module](#) – (deprecated, removed after 2023-02-24) Manages BGP Neighbor Address Family resource module.
- [nxos\\_command module](#) – Run arbitrary command on Cisco NXOS devices
- [nxos\\_config module](#) – Manage Cisco NXOS configuration sections
- [nxos\\_devicealias module](#) – Configuration of device alias for Cisco NXOS MDS Switches.
- [nxos\\_evpn\\_global module](#) – Handles the EVPN control plane for VXLAN.
- [nxos\\_evpn\\_vni module](#) – Manages Cisco EVPN VXLAN Network Identifier (VNI).
- [nxos\\_facts module](#) – Gets facts about NX-OS switches
- [nxos\\_feature module](#) – Manage features in NX-OS switches.
- [nxos\\_file\\_copy module](#) – Copy a file to a remote NXOS device.
- [nxos\\_gir module](#) – Trigger a graceful removal or insertion (GIR) of the switch.
- [nxos\\_gir\\_profile\\_management module](#) – Create a maintenance-mode or normal-mode profile
- [nxos\\_hostname module](#) – Hostname resource module.
- [nxos\\_hsrp module](#) – Manages HSRP configuration on NX-OS switches.
- [nxos\\_hsrp\\_interfaces module](#) – HSRP interfaces resource module
- [nxos\\_igmp module](#) – Manages IGMP global configuration.
- [nxos\\_igmp\\_interface module](#) – Manages IGMP interface configuration.
- [nxos\\_igmp\\_snooping module](#) – Manages IGMP snooping global configuration.
- [nxos\\_install\\_os module](#) – Set boot options like boot, kickstart image and issu.
- [nxos\\_interfaces module](#) – Interfaces resource module
- [nxos\\_l2\\_interfaces module](#) – L2 interfaces resource module
- [nxos\\_l3\\_interfaces module](#) – L3 interfaces resource module
- [nxos\\_lacp module](#) – LACP resource module
- [nxos\\_lacp\\_interfaces module](#) – LACP interfaces resource module
- [nxos\\_lag\\_interfaces module](#) – LAG interfaces resource module

<https://docs.ansible.com/ansible/latest/collections/cisco/nxos/index.html>

# Ansible Modules

- Always use the fully qualified collection name (FQCN) for the module
- The modules require parameters with values assigned that define your configuration intent
- Documentation provides details on default values and required values



# A word about YAML Syntax

## YAML Ain't Markup Language

- Human Readable Data Structures
  - Lists, Dictionaries, etc
- Used in inventory, playbooks, & variable files
- Best practice is to use:
  - Text editor (e.g. Notepad++)
  - IDE (e.g. VSCode) with language assistant support for YAML
  - Indentation is very important, and the proper editor will help you



Microsoft VSCode

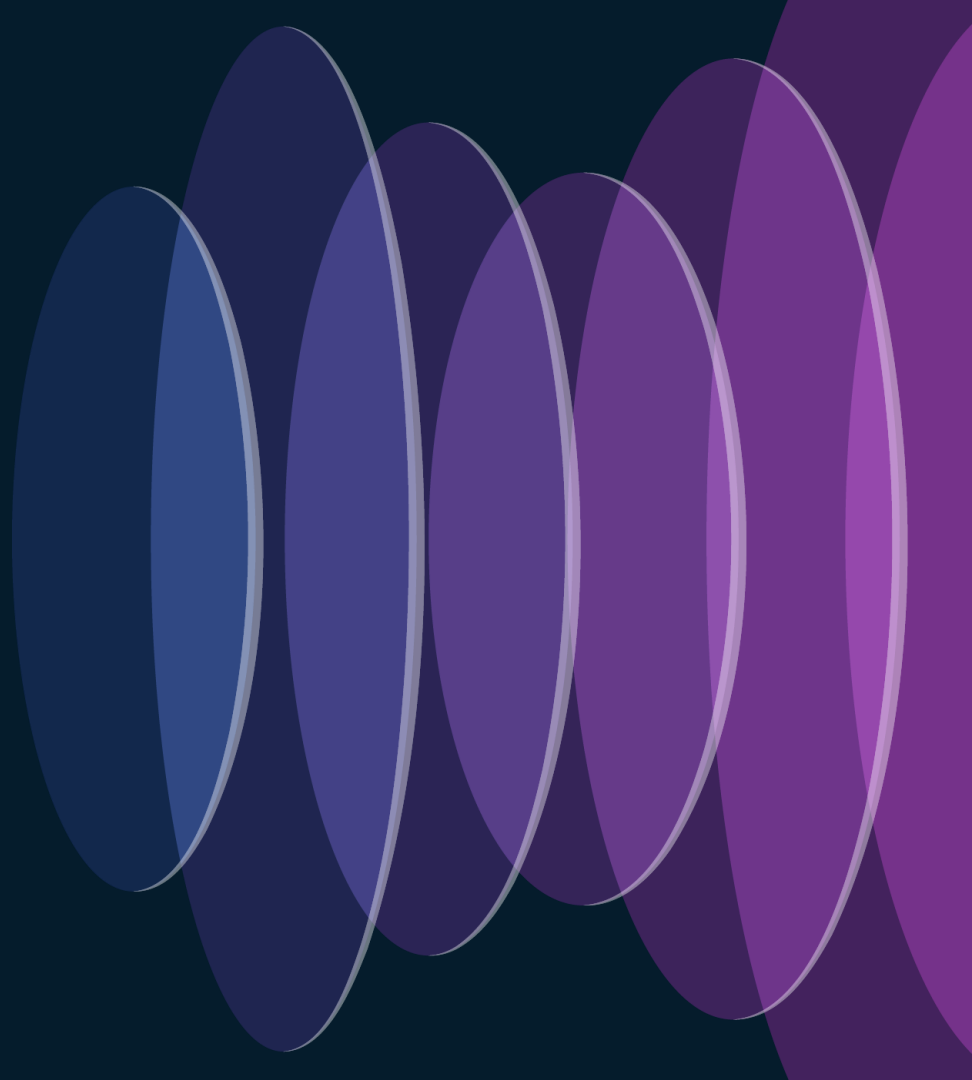


PyCharm



Notepad++

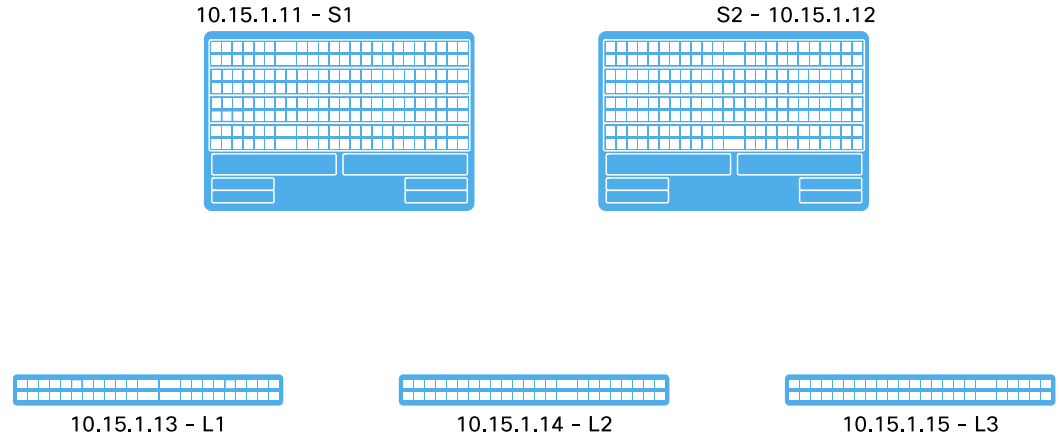
# Ansible Concepts



# VXLAN EVPN Fabric

## IaC – Nexus as Code

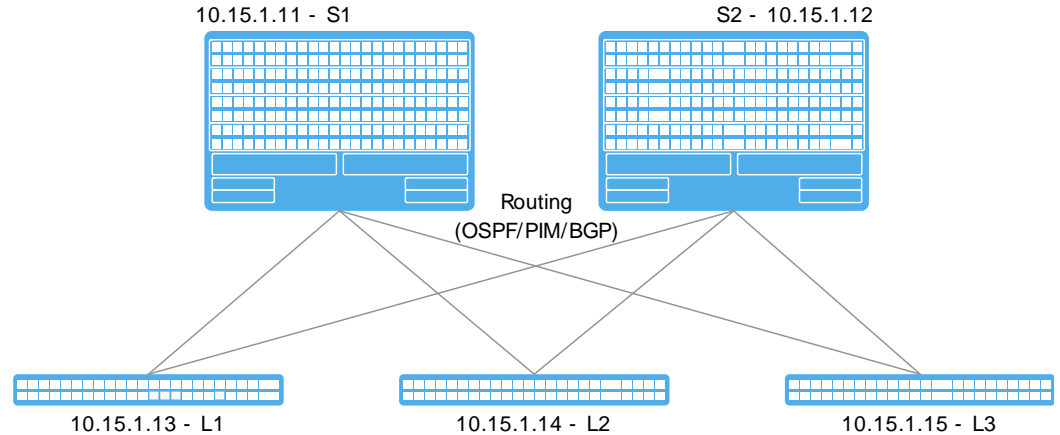
- Configure Hostnames, Features, etc (Common configuration)



# VXLAN EVPN Fabric

## laC – Nexus as Code

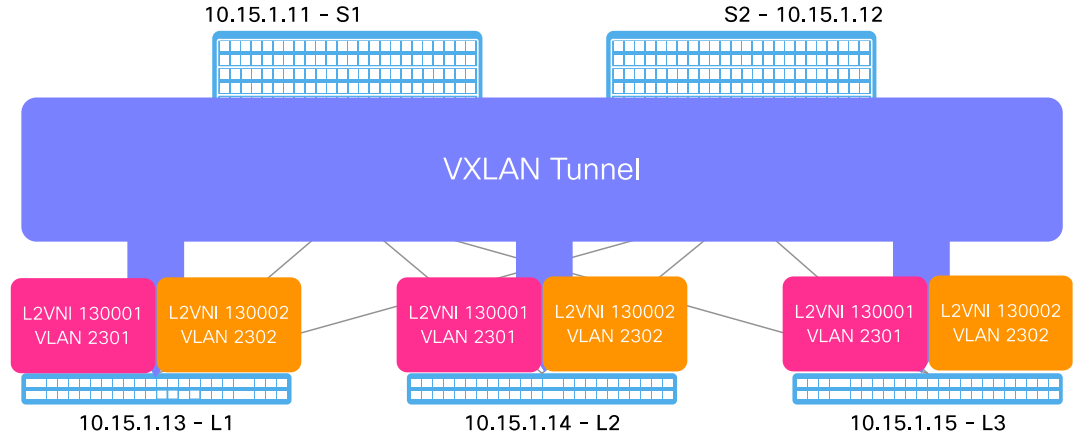
- Configure **Underlay** (Interfaces, Routing protocols, etc)



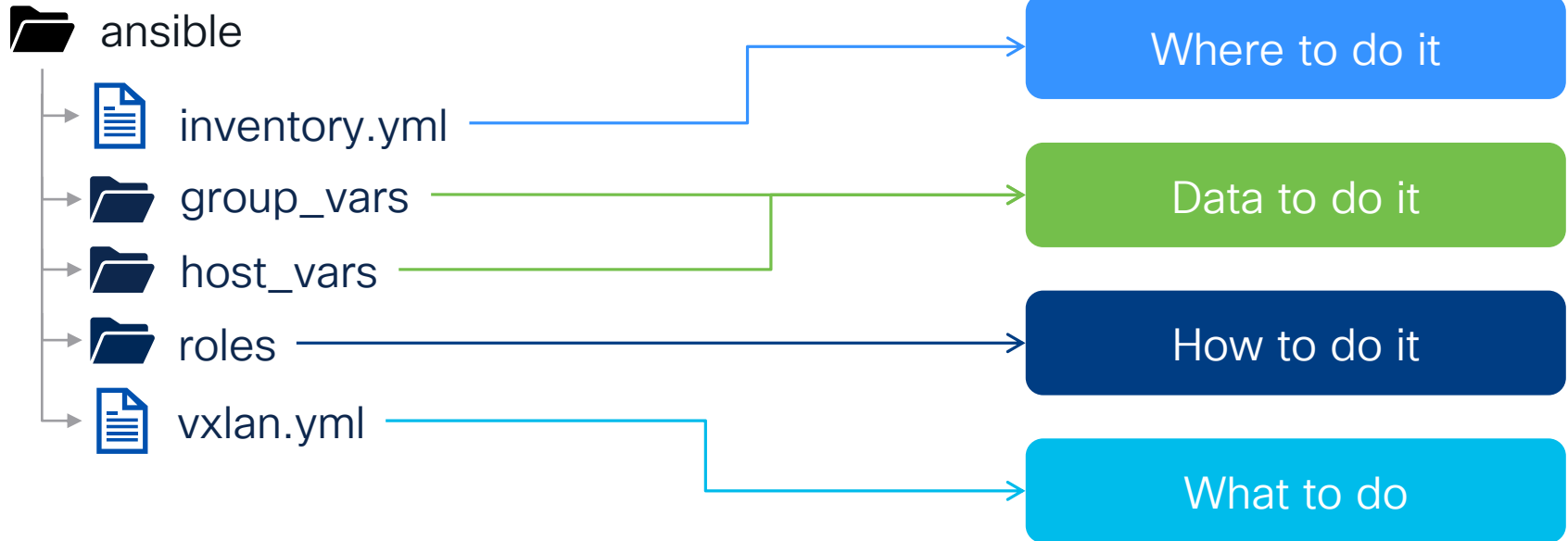
# VXLAN EVPN Fabric

## laC – Nexus as Code

- Configure **Overlay**  
(VRFs, VLANs, SVIs, etc)



# Ansible Directory Structure



# Ansible Inventory

ansible



inventory.yml



group\_vars



host\_vars



roles



vxlan.yml

```
---
# main inventory file
```

```
all:
```

```
vars:
```

```
ansible_connection: ansible.netcommon.network_cli
ansible_user: "nxos_username"
ansible_password: "nxos_password"
ansible_network_os: cisco.nxos.nxos
```

```
children:
```

```
spines:
```

```
hosts:
```

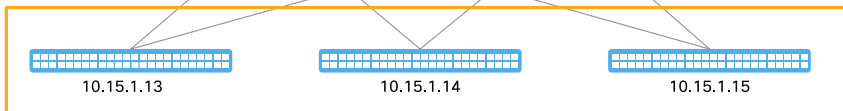
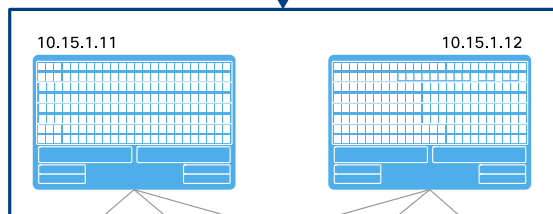
```
10.15.1.11:
10.15.1.12:
```

```
leaves:
```

```
hosts:
```

```
10.15.1.13:
10.15.1.14:
10.15.1.15:
```

Connection  
information  
for switches



# Ansible Inventory

ansible



inventory.yml



group\_vars



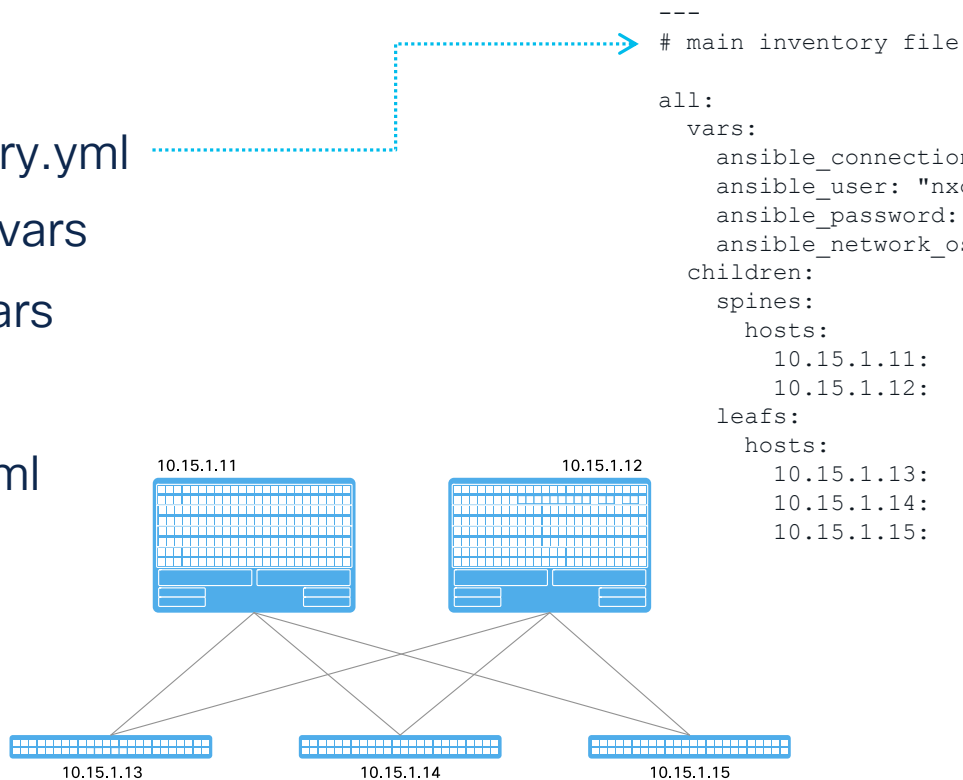
host\_vars



roles

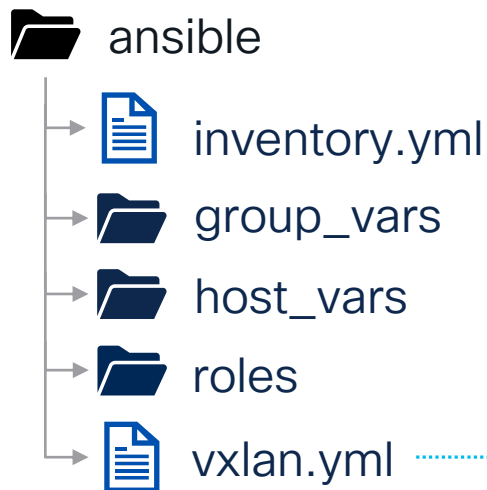


vxlan.yml



This will be annotated  
shorthand in subsequent  
slides as `network_cli`

# Ansible Playbook



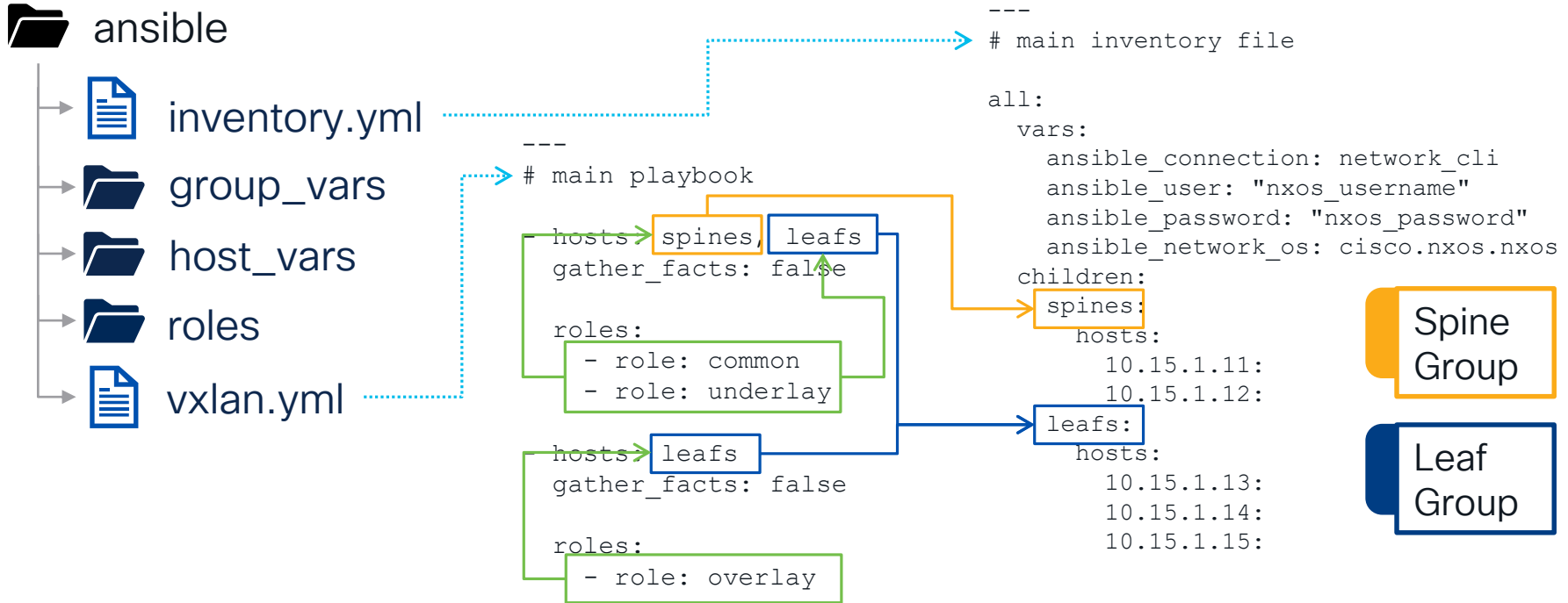
```
---  
# main playbook  
  
- hosts: spines, leafs  
  gather_facts: false  
  
  roles:  
    - role: common  
    - role: underlay  
  
- hosts: leafs  
  gather_facts: false  
  
  roles:  
    - role: overlay
```

# Ansible Networking

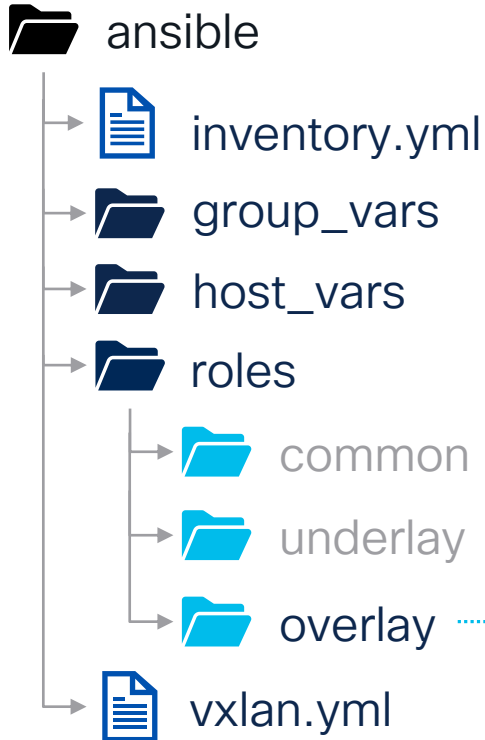
- Network modules execute from control node (Ansible host)
- Collections organized by network platform/OS
- Offers multiple connection protocols

Value of <code>ansible_connection</code>	Protocol	Requires	Persistent
<code>ansible.netcommon.network_cli</code>	CLI over SSH	<code>ansible_network_os</code>	Yes
<code>ansible.netcommon.httpapi</code>	API over HTTP/HTTPS	<code>ansible_network_os</code>	Yes

# Ansible Playbook Relationships



# Ansible Roles

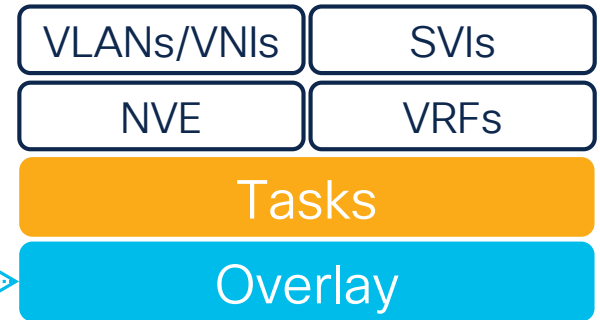
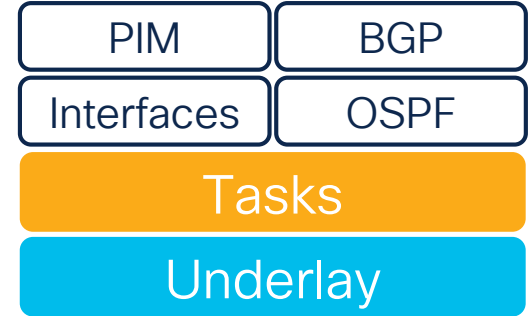
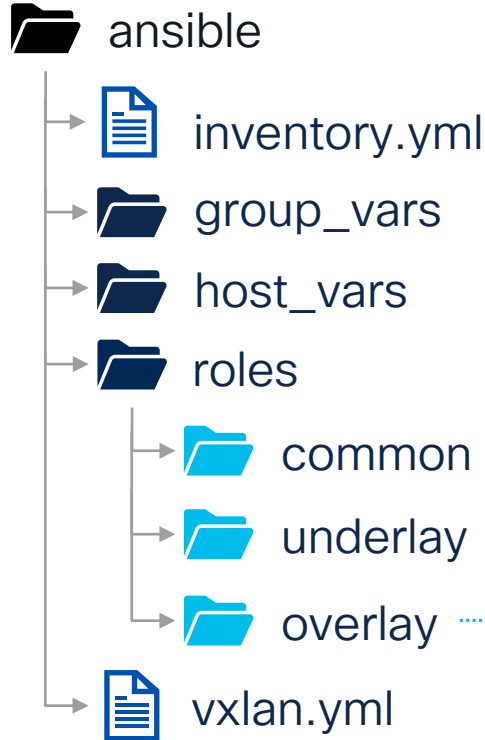


```
> ansible-galaxy init overlay
- Role overlay was created successfully
> tree overlay
overlay
├── README.md
├── tasks
│   └── main.yml
├── templates
└── vars
    └── main.yml
```

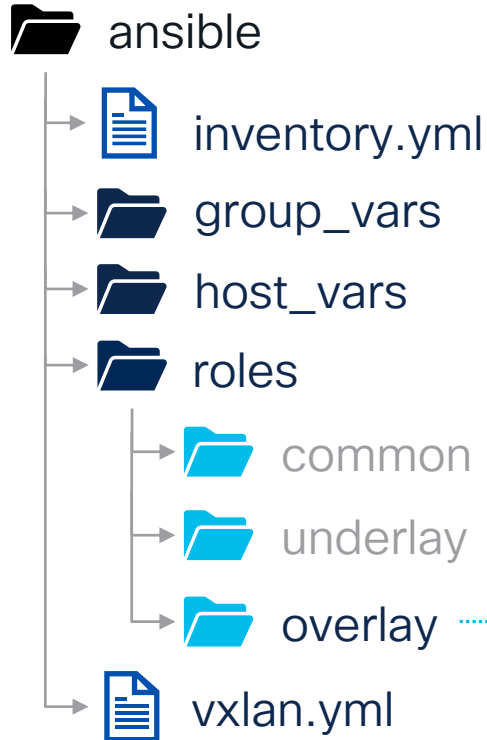


# Ansible Roles – Tasks

## How to do it



# Ansible Roles – Tasks



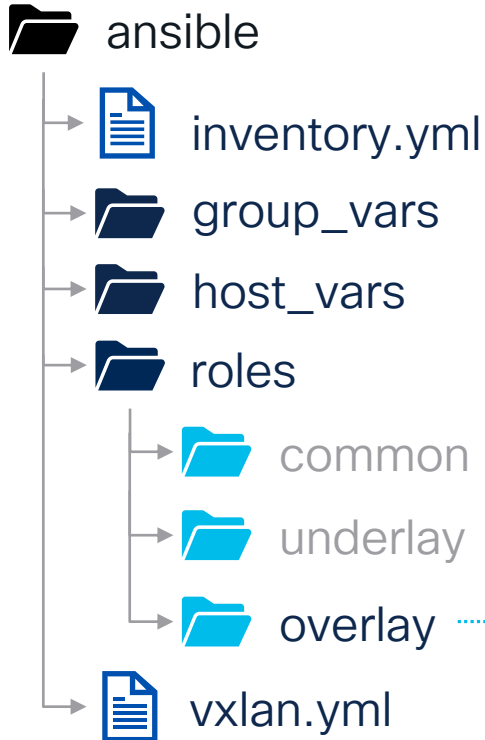
roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
    config:
      - name: Web Servers
        vlan_id: 101
        mapped_vni: 10101
      - name: DB Servers
        vlan_id: 102
        mapped_vni: 10102
      - name: vMotion
        vlan_id: 103
        mapped_vni: 10103
    state: merged
<snip>
```

- Defines VLAN-to-VNI mappings in config list block
- Config block allows for YAML list of dictionary objects

# Ansible Roles – Tasks



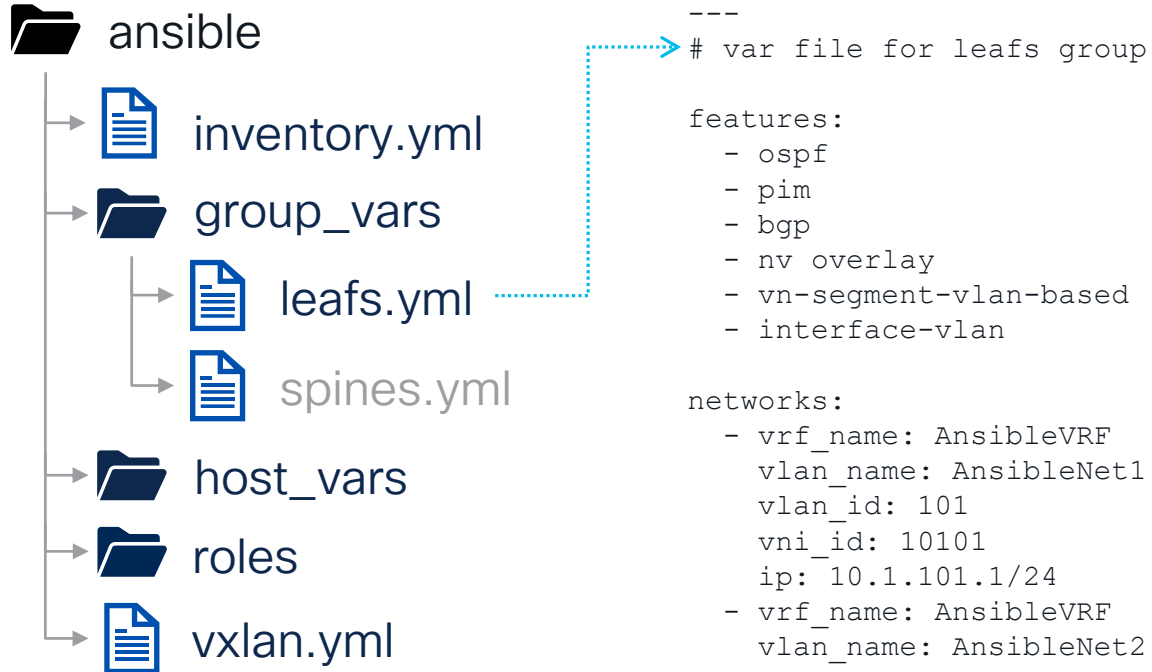
roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
    config:
      - name: vMOT Servers
        vni_id: 10101
        mapped_vni: 10101
      - name: vDB Servers
        vni_id: 10102
        mapped_vni: 10102
      - name: vMot
        vni_id: 10103
        mapped_vni: 10103
    state: merged
<snip>
```

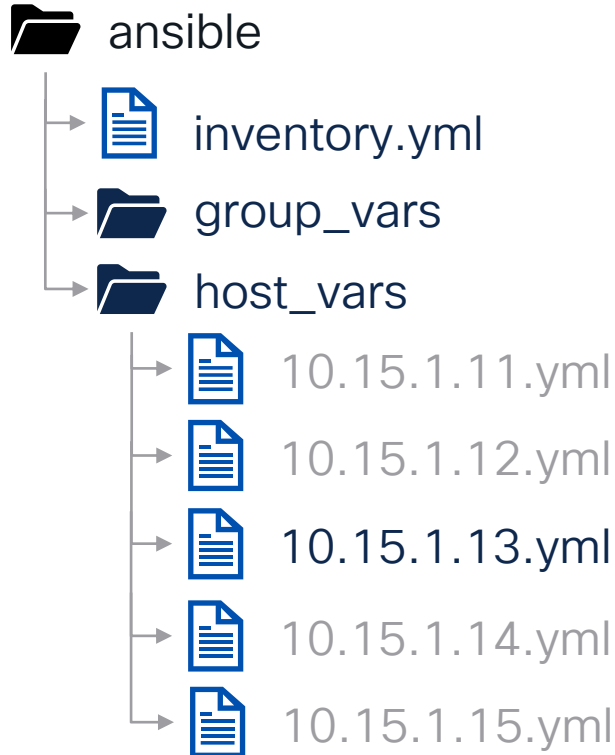
Do not do Ansible task parameters this way!

# Ansible Group Vars



- group\_vars files are named and referenced after inventory groups
- match inventory group!
- contains data common to specified group

# Ansible Host Vars



```
---
# vars file for L1

hostname: L1

layer3_physical_interfaces:
- interface: ethernet1/11
  description: To S1 Eth1/1
  mode: layer3
  ip_address: 10.1.1.1
  mask: 31
  mtu: 9216
- interface: ethernet1/12
  description: To S2 Eth1/1
  mode: layer3
  ip_address: 10.2.2.1
  mask: 31
  mtu: 9216
```

- host\_vars files are named and referenced after device IP address or FQDN
- match inventory name!
- contains data specific to that device

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
<snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/common/tasks/main.yml

```
---
# tasks file for roles/common

- name: Configure Hostname
  cisco.nxos.nxos_hostname:
  config:
    hostname: "{{ hostname }}"
  state: merged
<snip>
```



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ item.vlan_name }}"
      vlan_id: "{{ item.vlan_id }}"
      mapped_vni: "{{ item.vni_id }}"
  loop: "{{ networks }}"
<snip>
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
<snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/common/tasks/main.yml

```
---
# tasks file for roles/common

- name: Configure Hostname
  cisco.nxos.nxos_hostname:
  config:
    hostname: "{{ hostname }}"
  state: merged
<snip>
```



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ item.vlan_name }}"
      vlan_id: "{{ item.vlan_id }}"
      mapped_vni: "{{ item.vni_id }}"
  loop: "{{ networks }}"
<snip>
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
  <snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/common/tasks/main.yml

```
---
# tasks file for roles/common

- name: Configure Hostname
  cisco.nxos.nxos_hostname:
  config:
    hostname: "{{ hostname }}"
    state: merged
  <snip>
```



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ item.vlan_name }}"
      vlan_id: "{{ item.vlan_id }}"
      mapped_vni: "{{ item.vni_id }}"
    loop: "{{ networks }}"
  <snip>
```

## Data to do it



host\_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

hostname: L1
<snip>
```



group\_vars/leafs.yml

```
---
# var file for leafs group
<snip>
networks:
  - vrf_name: AnsibleVRF
    vlan_name: AnsibleNet1
    vlan_id: 101
    vni_id: 10101
    ip: 10.1.101.1/24
  - vrf_name: AnsibleVRF
    vlan_name: AnsibleNet2
    vlan_id: 102
    vni_id: 10102
    ip: 10.1.102.1/24
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
  <snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/common/tasks/main.yml

```
---
# tasks file for roles/common

- name: Configure Hostname
  cisco.nxos.nxos_hostname:
  config:
    hostname: "{{ hostname }}"
  state: merged
<snip>
```



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ item.vlan_name }}"
      vlan_id: "{{ item.vlan_id }}"
      mapped_vni: "{{ item.vni_id }}"
  loop: "{{ networks }}"
<snip>
```

## Data to do it



host\_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

hostname: L1
<snip>
```



group\_vars/leafs.yml

```
---
# var file for leafs group
<snip>
networks:
  - vrf name: AnsibleVRF
    vlan name: AnsibleNet1
    vlan id: 101
    vni id: 10101
    ip: 10.1.101.1/24
  - vrf name: AnsibleVRF
    vlan name: AnsibleNet2
    vlan id: 102
    vni id: 10102
    ip: 10.1.102.1/24
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
<snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/common/tasks/main.yml

```
---
# tasks file for roles/common

- name: Configure Hostname
  cisco.nxos.nxos_hostname:
  config:
    hostname: "{{ hostname }}"
  state: merged
<snip>
```



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ item.vlan_name }}"
      vlan_id: "{{ item.vlan_id }}"
      mapped_vni: "{{ item.vni_id }}"
  loop: "{{ networks }}"
<snip>
```

## Data to do it



host\_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

hostname: L1
<snip>
```



group\_vars/leafs.yml

```
---
# var file for leafs group
<snip>
networks:
  - vrf name: AnsibleVRF
    vlan name: AnsibleNet1
    vlan id: 101
    vni id: 10101
    ip: 10.1.101.1/24
  - vrf name: AnsibleVRF
    vlan name: AnsibleNet2
    vlan id: 102
    vni id: 10102
    ip: 10.1.102.1/24
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
<snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leaves:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leaves
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leaves
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Generate VLAN Config Payload
  ansible.builtin.set_fact:
    nxos_vlans: |
      "{{ lookup('template', 'vlans.j2') }}"

- name: Configure VLANs
  cisco.nxos.nxos_vlans:
    config: "{{ nxos_vlans | from_yaml }}"
    state: merged
<snip>
```



roles/overlay/templates/vlans.j2

```
{% for network in networks %}
- vlan_name: {{ network.vlan_name }}
  vlan_id: {{ network.vlan_id }}
  vni_id: {{ network.vni_id }}
{% endfor %}
```

## Data to do it



host\_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

hostname: L1
<snip>
```



group\_vars/leafs.yml

```
---
# var file for leaves group
<snip>
networks:
  - vrf_name: AnsibleVRF
    vlan_name: AnsibleNet1
    vlan_id: 101
    vni_id: 10101
    ip: 10.1.101.1/24
  - vrf_name: AnsibleVRF
    vlan_name: AnsibleNet2
    vlan_id: 102
    vni_id: 10102
    ip: 10.1.102.1/24
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
<snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leaves:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leaves
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leaves
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Generate VLAN Config Payload
  ansible.builtin.set_fact:
    nxos_vlans: |
      "{{ lookup('template', 'vlans.j2') }}"

- name: Configure VLANs
  cisco.nxos.nxos_vlans:
    config: "{{ nxos_vlans | from_yaml }}"
    state: merged
<snip>
```



roles/overlay/templates/vlans.j2

```
{% for network in networks %}
- vlan_name: {{ network.vlan_name }}
  vlan_id: {{ network.vlan_id }}
  vni_id: {{ network.vni_id }}
{% endfor %}
```

## Data to do it



host\_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

hostname: L1
<snip>
```



group\_vars/leaves.yml

```
---
# var file for leaves group
<snip>
networks:
  - vrf_name: AnsibleVRF
    vlan_name: AnsibleNet1
    vlan_id: 101
    vni_id: 10101
    ip: 10.1.101.1/24
  - vrf_name: AnsibleVRF
    vlan_name: AnsibleNet2
    vlan_id: 102
    vni_id: 10102
    ip: 10.1.102.1/24
```

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
<snip>
  children:
    spines:
      hosts:
        10.15.1.11:
        10.15.1.12:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

## How to do it



roles/overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Generate VLAN Config Payload
  ansible.builtin.set_fact:
    nxos_vlans: |
      "{{ lookup('template', 'vlans.j2') }}"

- name: Configure VLANs
  cisco.nxos.nxos_vlans:
    config: "{{ nxos_vlans | from_yaml }}"
    state: merged
<snip>
```



roles/overlay/templates/vlans.j2

```
{% for network in networks %}
- vlan_name: {{ network.vlan_name }}
  vlan_id: {{ network.vlan_id }}
  vni_id: {{ network.vni_id }}
{% endfor %}
```

## Data to do it



host\_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

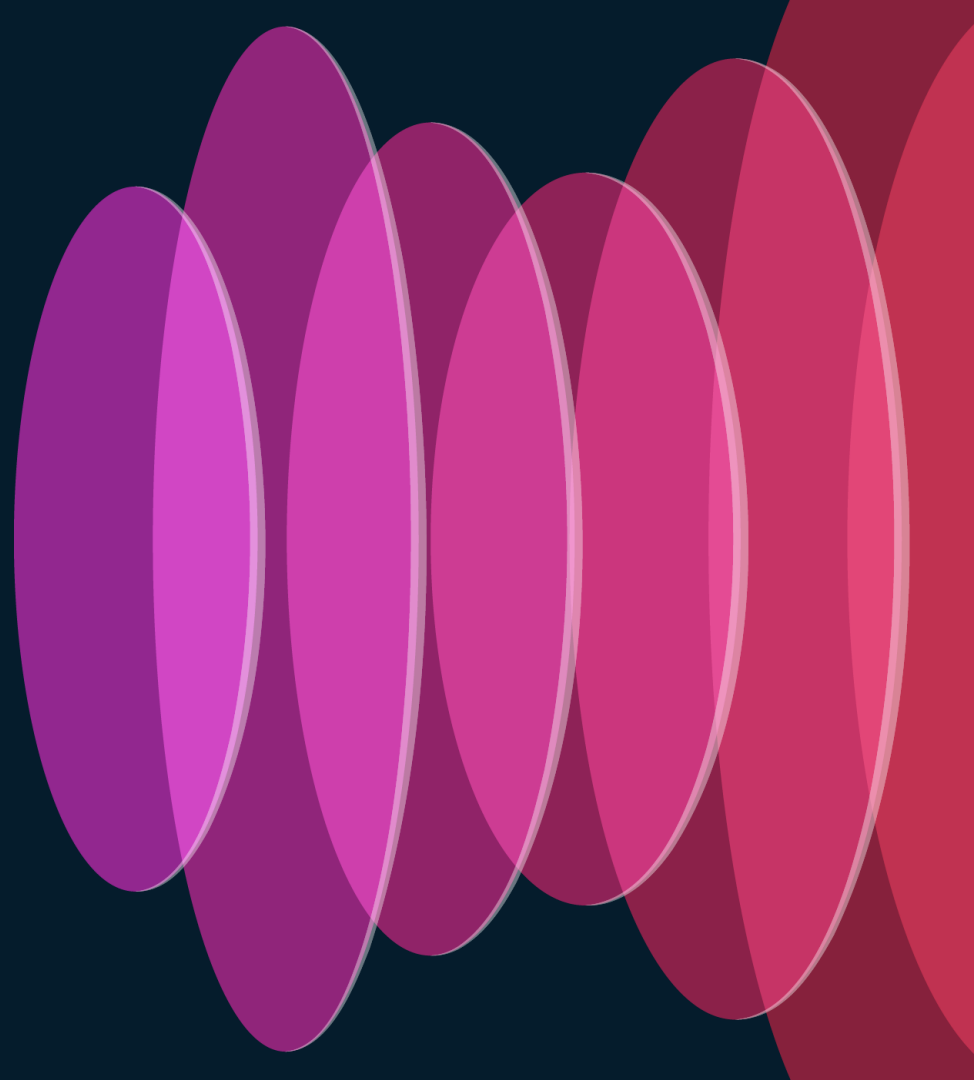
hostname: L1
<snip>
```



group\_vars/leafs.yml

```
---
# var file for leafs group
<snip>
networks:
  - vrf name: AnsibleVRF
    vlan name: AnsibleNet1
    vlan id: 101
    vni id: 10101
    ip: 10.1.101.1/24
  - vrf name: AnsibleVRF
    vlan name: AnsibleNet2
    vlan id: 102
    vni id: 10102
    ip: 10.1.102.1/24
```

Demo



# A word about Ansible Variables

- Can be defined in **many** different places
  - Most commonly in group\_vars and host\_vars directory
- Created dynamically during runtime
- Used in task modules, conditional logic, templates, etc
- Jinja2 syntax used to reference
- Variable precedence is used

[https://docs.ansible.com/ansible/latest/playbook\\_guide/playbooks\\_variables.html#understanding-variable-precedence](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#understanding-variable-precedence)

# Variables with Jinja2 Syntax

- Variable substitution in tasks
- Uses double curly braces wrapped in quotes: "{{ }}"

```
# tasks file for roles/overlay                                # vars defined somewhere in Ansible

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ vlan_name }}"
      vlan_id: "{{ vlan_id }}"
      mapped_vni: "{{ vni_id }}"

vlan_name: Web Servers
vlan_id: 101
vni_id: 10101
```

Set of  
Key/Value Pairs

# Ansible Variable Lists

Go from this...



To this...

```
# vars defined somewhere in Ansible
```

```
vlan_name: Web Servers  
vlan_id: 101  
vni_id: 10101
```

```
# vars defined somewhere in Ansible
```

```
networks:
```

```
- vlan_name: Web Servers  
  vlan_id: 101  
  vni_id: 10101  
  
- vlan_name: DB Servers  
  vlan_id: 102  
  vni_id: 10102  
  
- vlan_name: vMotion  
  vlan_id: 103  
  vni_id: 10103
```

Sequential list of three dictionary objects containing VLAN information that can be referenced iteratively

# Ansible Loop with Jinja2 Syntax

- Use Ansible loop to iterate data for a tasks

```
# tasks file for roles/overlay
- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
  config:
    - name: "{{ item.vlan_name }}"
      vlan_id: "{{ item.vlan id }}"
      mapped_vni: "{{ item.vni id }}"
  loop: "{{ networks }}"

# vars defined somewhere in Ansible
networks:
  - vlan_name: Web Servers
    vlan_id: 101
    vni_id: 10101
  - vlan_name: DB Servers
    vlan_id: 102
    vni_id: 10102
  - vlan_name: vMotion
    vlan_id: 103
    vni_id: 10103
```

# A word about Jinja Templating

- Leverage module and filters from Ansible Builtin collection
- Create template file(s) in a role's template directory: **.j2** file extension

```
# tasks file for roles/overlay
```

```
- name: Generate VLAN Config Payload
```

```
  ansible.builtin.set_fact:
```

```
    nxos_vlans: "{{ lookup('template', 'vlans.j2') }}"
```

```
- name: Configure VLANs
```

```
  cisco.nxos.nxos_vlans:
```

```
    config: "{{ nxos_vlans | from_yaml }}"
```

```
    state: merged
```

```
# config data passed to task
```

```
- vlan_id: 101
```

```
- Vlan_id: 102
```

```
# group_vars/leafs.yml
```

```
networks:
```

```
- vlan_name: Web Servers
```

```
  vlan_id: 101
```

```
  vni_id: 10101
```

```
- vlan_name: DB Servers
```

```
  vlan_id: 102
```

```
  vni_id: 10102
```

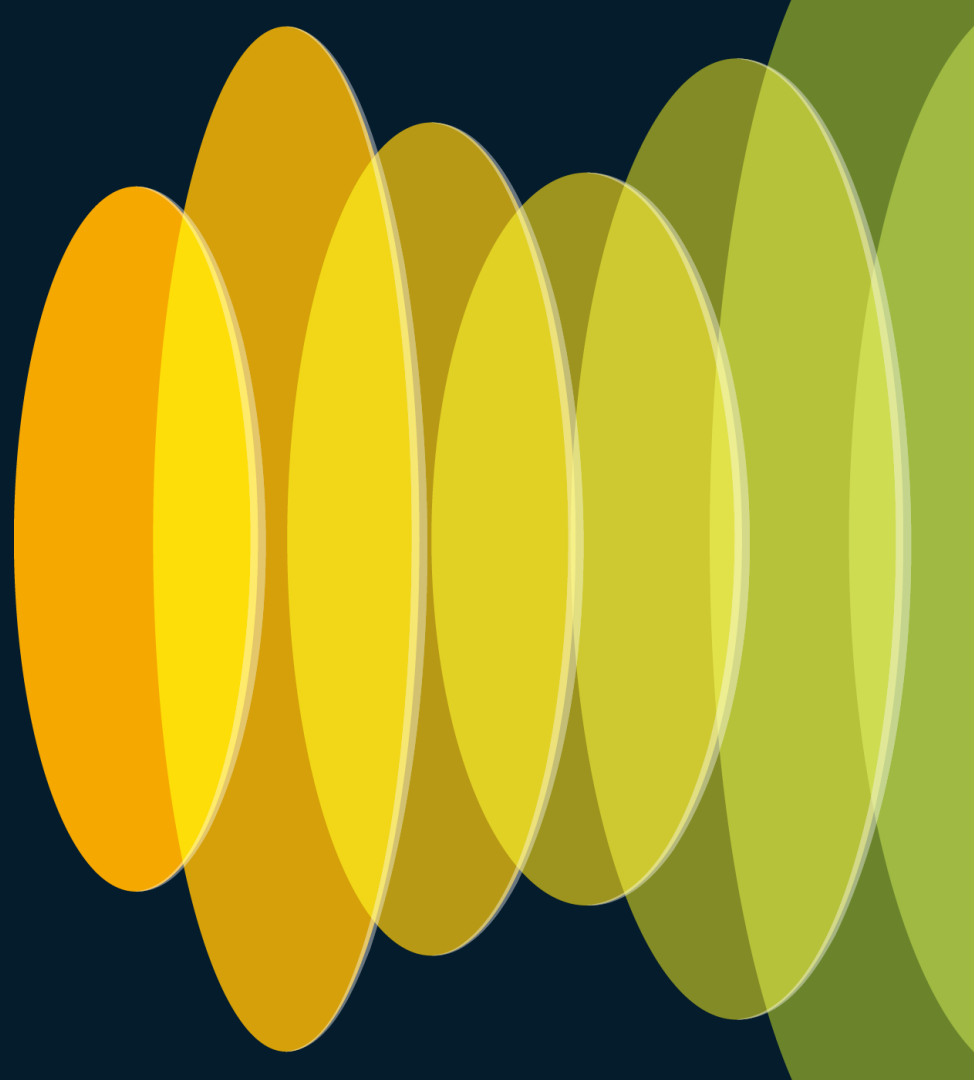
```
# roles/overlay/templates/vlans.j2
```

```
{% for network in networks %}
```

```
- vlan_id: {{ network.vlan_id }}
```

```
{% endfor %}
```

# Additional Details for NXOS



# Two Types of Modules for NXOS

## Legacy Modules

- Inconsistent across different network devices
- Requires task loops for more than one configuration item
- Simple states, present or absent

## Resource Modules

- Consistent across different network devices
- Can leverage task loops or Jinja2 templating for config blocks
- Introduces new states for Ansible to be the source of truth

# A word about Resource Module States

- Merged - Ansible merges the on-device configuration with the provided configuration in the task.
- Replaced - Ansible replaces the on-device configuration subsection with the provided configuration subsection in the task.
- Overridden - Ansible overrides the on-device configuration for the resource with the provided configuration in the task. Use caution with this state as you could remove your access to the device (for example, by overriding the management interface configuration).
- Deleted - Ansible deletes the on-device configuration subsection and restores any default settings.

# NXOS Config Fallback Module

How to configure NXOS when a module is missing

- Allows passing direct cli configuration
- Can take full running-config backup, e.g. before a change operation
- Perform a save operation, i.e. "copy run start"

```
- name: Configure PIM Anycast RP
  cisco.nxos.nxos_config:
    lines:
      - "ip pim anycast-rp {{ s1_loopback1 }} {{ s1_loopback0 }}"
      - "ip pim anycast-rp {{ s2_loopback1 }} {{ s2_loopback0 }}"
    save_when: modified
```

## Options:

- always – copy always
- modified – copy only if changed since last save
- changed – copy only if the **task** made a change
- never – never copy

# NXOS Command Fallback Module

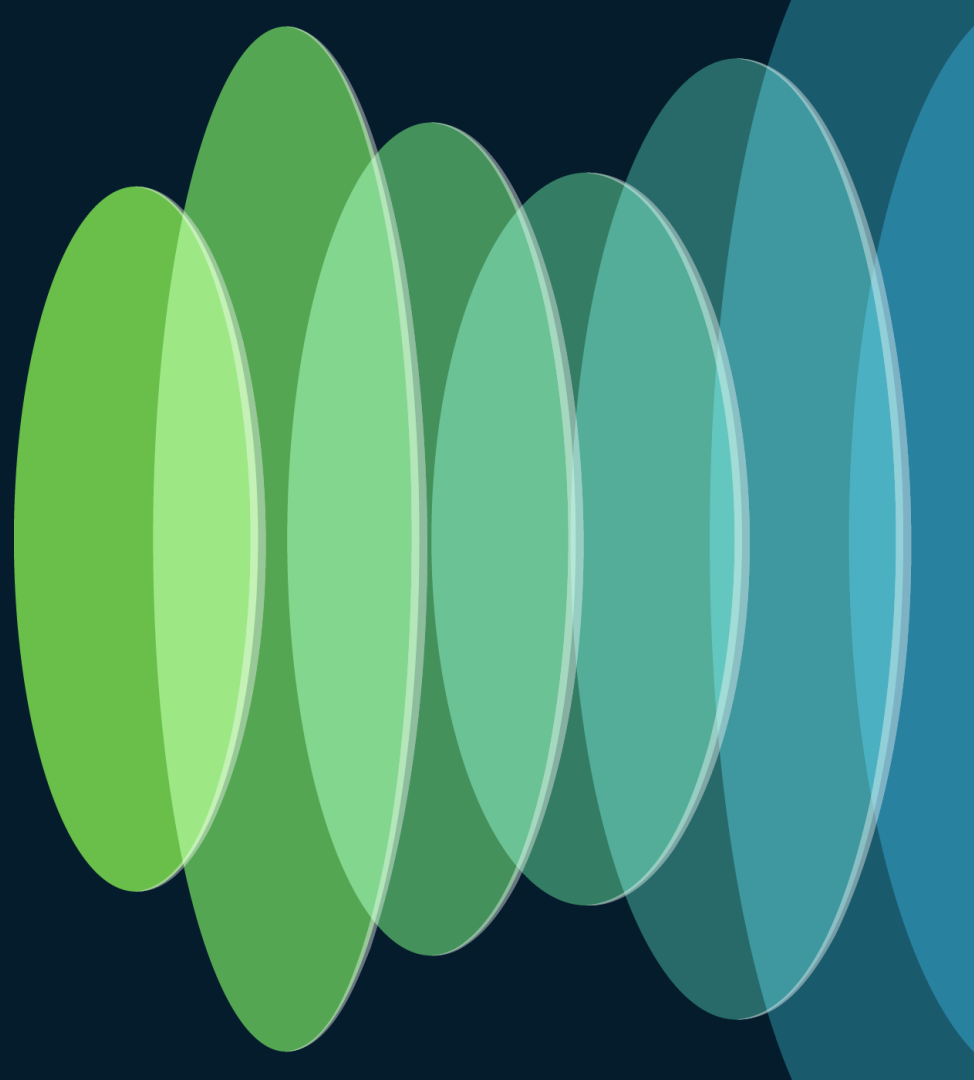
How to send commands to NXOS when a module is missing

- Allows sending arbitrary commands, e.g. show commands
- Supports prompt handling
- Can handle list of commands or prompts

```
- name: Get Show Commands
  cisco.nxos.nxos_command:
  commands:
    - show version
    - show ip ospf neighbor
    - show ip pim neighbor
```

```
- name: Misc Commands
  cisco.nxos.nxos_command:
  commands: copy ftp://nxos.bin bootflash:
  prompt:
    - "Username:"
    - "Password:"
  answer:
    - <username>
    - <password>
```

# An Example



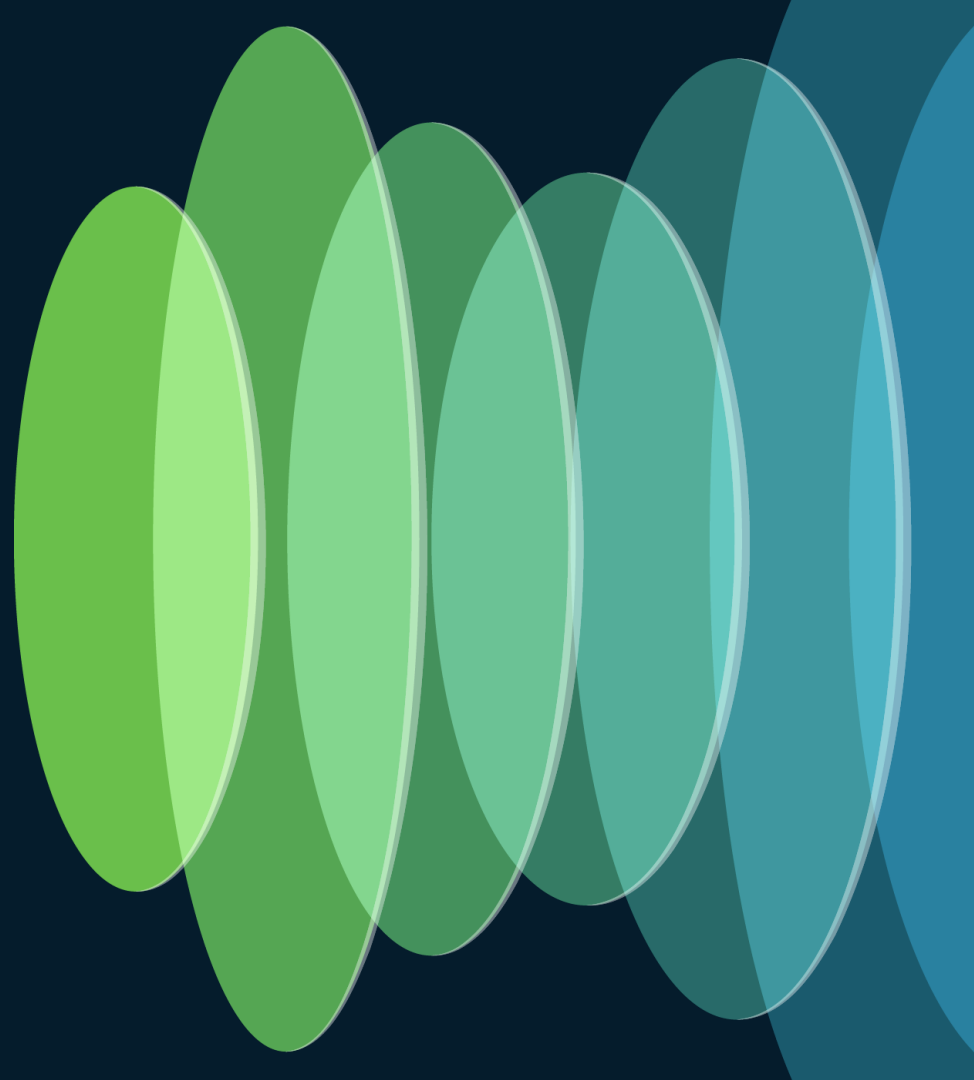
# Executing Ansible Playbooks

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

```
~/development/ansible/BRKDCN-2946 > main
brkdcn-2946 3.11.2 > ansible-playbook -i inventory.yml vxlan.yml

PLAY [spines, leafs] *****
TASK [common : Configure Hostname] *****
```

# Infrastructure as Code with NDFC and Ansible



# Cisco Nexus Dashboard Fabric Controller (NDFC)

Formerly Called (DCNM)



# Automation



Accelerate provisioning from days to minutes

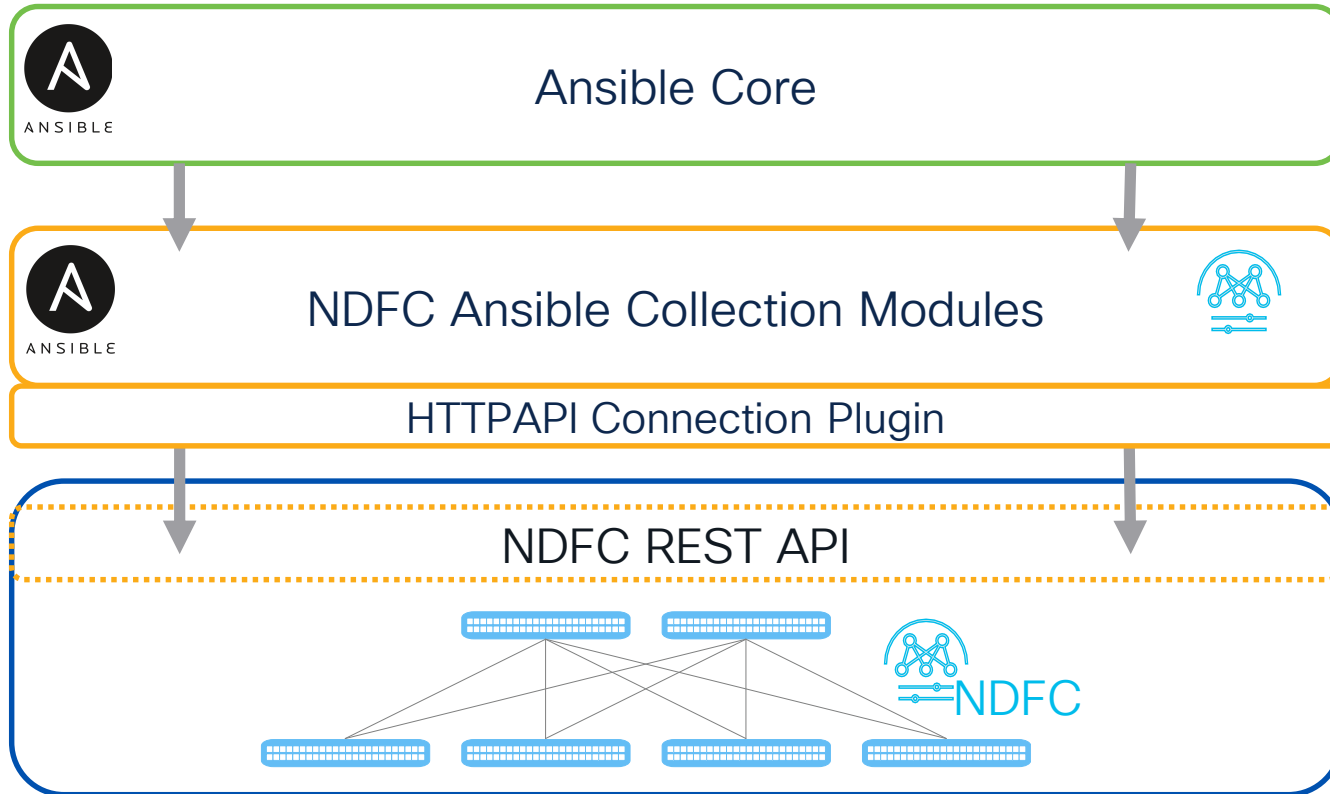
Easy to understand approach to auto-bootstrapping of entire fabric

Rapid Deployment with Fabric Builder best practice templates for VXLAN-EVPN

DevOps friendly

Enhanced Programmability

# NDFC DevOps “Friendliness”



# NDFC Collection Modules – Toolbox

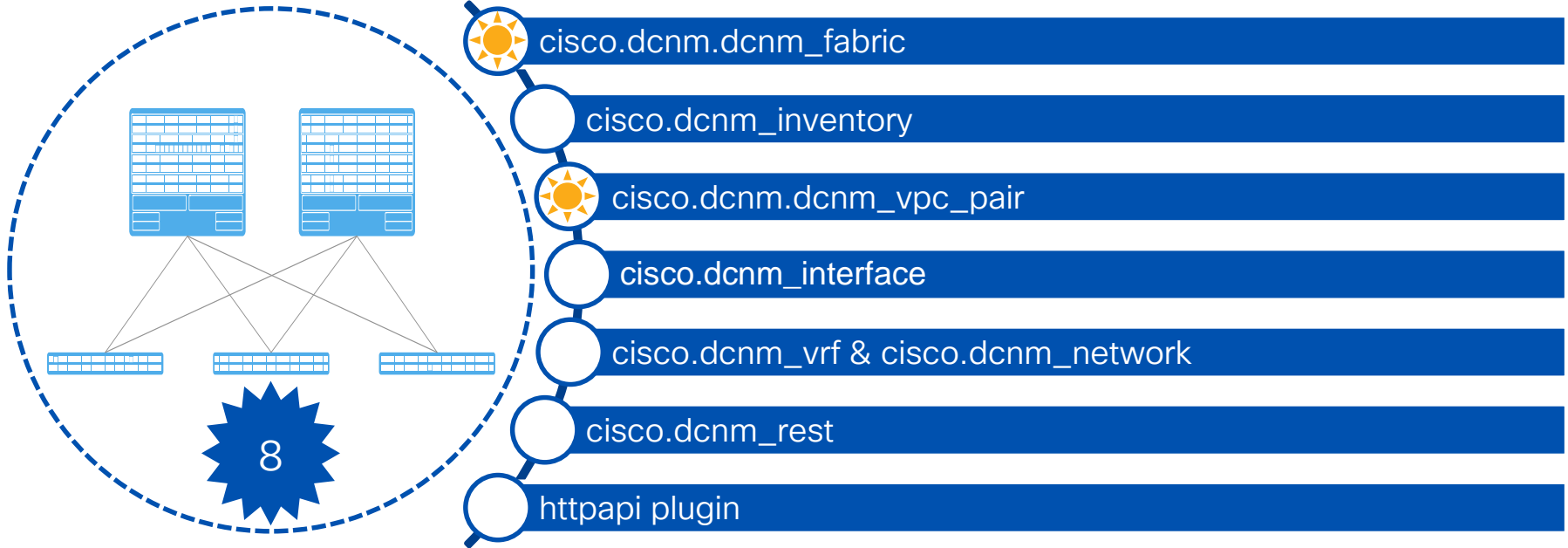
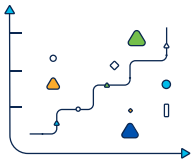
<https://galaxy.ansible.com/cisco/dcnm>



17 Modules

Module Name	Purpose
cisco.dcnm.dcnm_rest	General Do Anything Module
<b>cisco.dcnm.fabric (Ver 3.5.0)</b>	<b>Manage creation and configuration of NDFC fabrics</b>
cisco.dcnm.dcnm_inventory	Add Devices To Fabric
<b>cisco.dcnm.dcnm_vpc_pair (Ver 3.5.0)</b>	<b>Manage vPC switch pairs</b>
cisco.dcnm.dcnm_interface	Configure Fabric Interfaces
cisco.dcnm.dcnm_vrf	Add Overlay VRFs
cisco.dcnm.dcnm_network	Add Overlay Networks / VLANs
cisco.dcnm.dcnm_template	Create Custom Templates
cisco.dcnm.dcnm_policy	Create Policies Based On Templates
cisco.dcnm.dcnm_links	Manage Fabric Links
cisco.dcnm.dcnm_resource_manager	Manage Fabric Resources
<b>cisco.dcnm.dcnm_image_upload (Ver 3.5.0)</b>	<b>Manage Switch Images</b>
<b>cisco.dcnm.dcnm_image_policy (Ver 3.5.0)</b>	<b>Manage Image Policies</b>
<b>cisco.dcnm.dcnm_image_upgrade (Ver 3.5.0)</b>	<b>Manage Images for NXOS Switches</b>
cisco.dcnm.dcnm_service_node	Manage Service Nodes
cisco.dcnm.dcnm_service_policy	Manage Service Policy
cisco.dcnm.dcnm_service_route_peering	Manage Service Route Peering

# Primary VXLAN EVPN Fabric Plugins



 New Modules – NDFC Collection Version 3.5.0

# Getting started

## Ansible Collection Installation

Collection Location: <https://galaxy.ansible.com/cisco/dcnm>

Install Command:

```
* pip install ansible
* ansible-galaxy collection install cisco.dcnm
```

Ansible uses the Fully Qualified Collection Name (FQCN)

Namespace: `cisco`

Collection Name: `dcnm`

# 20/20 Hindsight Tech “Specs”

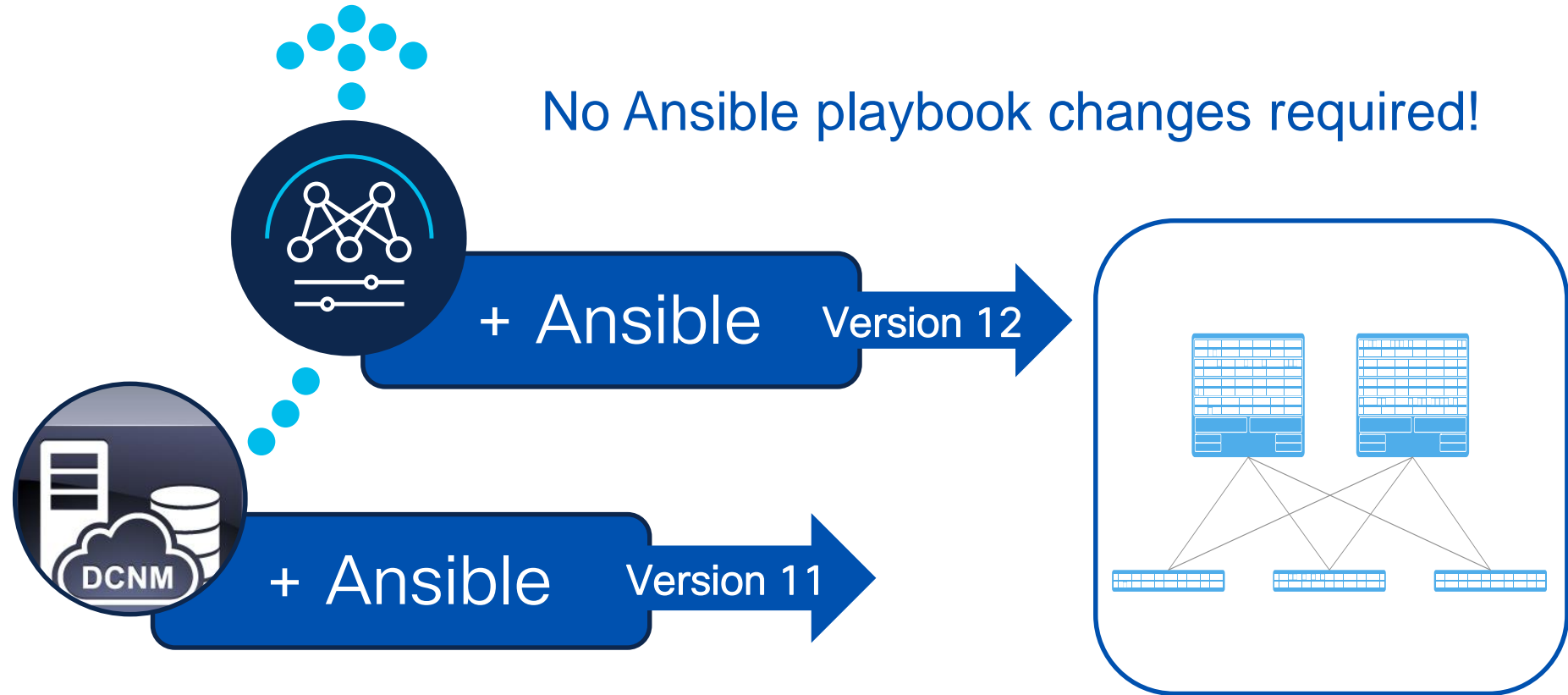


NDFC - <https://galaxy.ansible.com/cisco/dcnm>

# Hindsight “Specs” Model



# Transparent NDFC/DCNM Controller Support





+

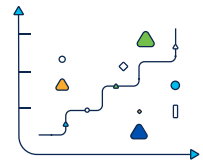


ANSIBLE

Let's Build Something With NDFC and Ansible Together!

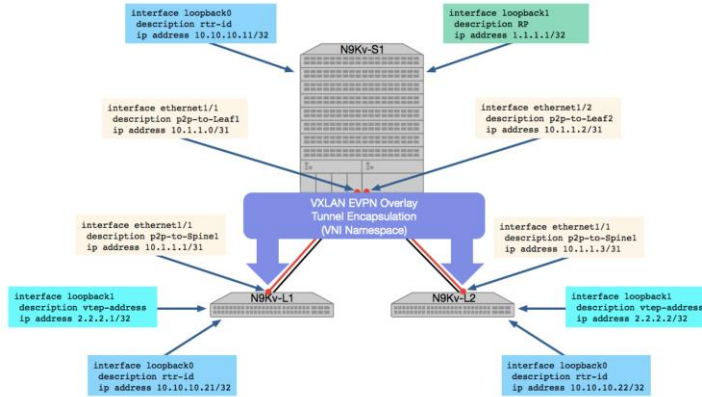
**CISCO** *Live!*

# Levels of Complexity – CLI



```

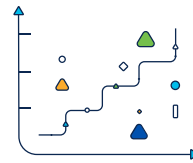
feature bgp
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay
nv overlay evpn
vlan 10
  vn-segment 10000
vlan 11
  vn-segment 10011
interface loopback0
  ip address 10.10.10.21/32
  ip pim sparse-mode
  ip router ospf UNDERLAY area 0
interface loopback1
  ip address 2.2.2.1/32
  ip pim sparse-mode
  ip router ospf UNDERLAY area 0
vrf context Tenant-1
  vni 10000
  rd auto
  address-family ipv4 unicast
    route-target both auto evpn
  
```



```

router bgp 65001
  router-id 10.10.10.21
  neighbor 10.10.10.11
    remote-as 65001
    update-source loopback0
    address-family 12vpn evpn
      send-community
      send-community extended
  vrf Tenant-1
    address-family ipv4 unicast
      advertise 12vpn evpn
evpn
  vni 10011 12
    rd auto
    route-target import auto
    route-target export auto
interface Vlan10
  no shutdown
  vrf member Tenant-1
  ip forward
interface Vlan11
  no shutdown
  vrf member Tenant-1
  ip address 10.0.11.1/24
  fabric forwarding mode anycast-gateway
interface nvel
  no shutdown
  source-interface loopback1
  host-reachability protocol bgp
  member vni 10000 associate-vrf
  member vni 10011
  mcast-group 239.0.0.11
  
```

# Levels of Complexity – NXOS Modules



```
feature bgp
feature interface-vlan
feature vn-segment-vlan-based
feature nv overlay
nv overlay evpn
vlan 10
  vn-segment 10000
vlan 11
  vn-segment 10011
interface loopback0
  ip address 10.10.10.21/32
  ip pim sparse-mode
  ip router ospf UNDERLAY area 0
interface loopback1
  ip address 2.2.2.1/32
  ip pim sparse-mode
  ip router ospf UNDERLAY area 0
vrf context Tenant-1
  vni 10000
  rd auto
  address-family ipv4 unicast
  route-target both auto evpn
```

- > *nxos\_feature*
- > *nxos\_interface*
- > *nxos\_l3\_interface*
- > *nxos\_interface\_ospf*
- > *nxos\_pim\_rp\_address*
- > *nxos\_pim\_interface*
- > *nxos\_evpn\_global*
- > *nxos\_bgp*
- > *nxos\_bgp\_af*
- > *nxos\_bgp\_neighbor*
- > *nxos\_bgp\_neighbor\_af*
- > *nxos\_vlan*
- > *nxos\_vrf*
- > *nxos\_vrf\_af*
- > *nxos\_vrf\_interface*
- > *nxos\_vxlan\_vtep*
- > *nxos\_vxlan\_vtep\_vni*
- > *nxos\_evpn\_vni*
- > *nxos\_config*



```
router bgp 65001
  router-id 10.10.10.21
  neighbor 10.10.10.11
    remote-as 65001
    update-source loopback0
    address-family l2vpn evpn
      send-community
      send-community extended
  vrf Tenant-1
    address-family ipv4 unicast
    advertise l2vpn evpn

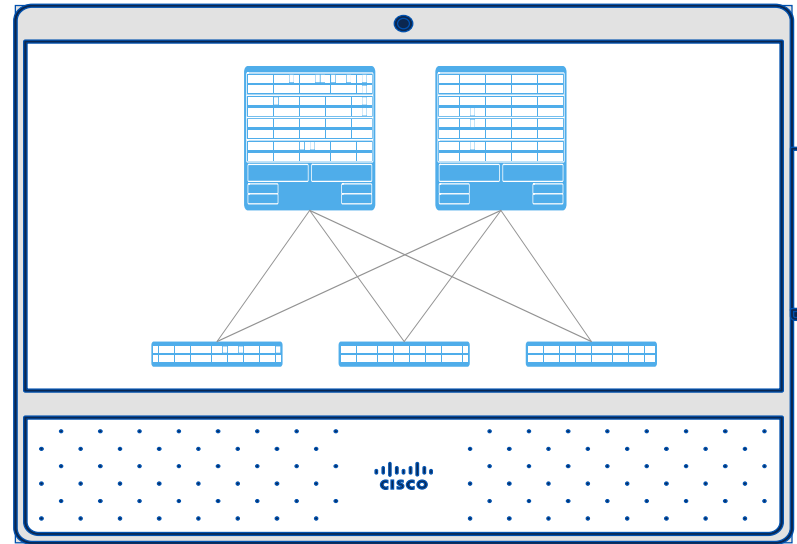
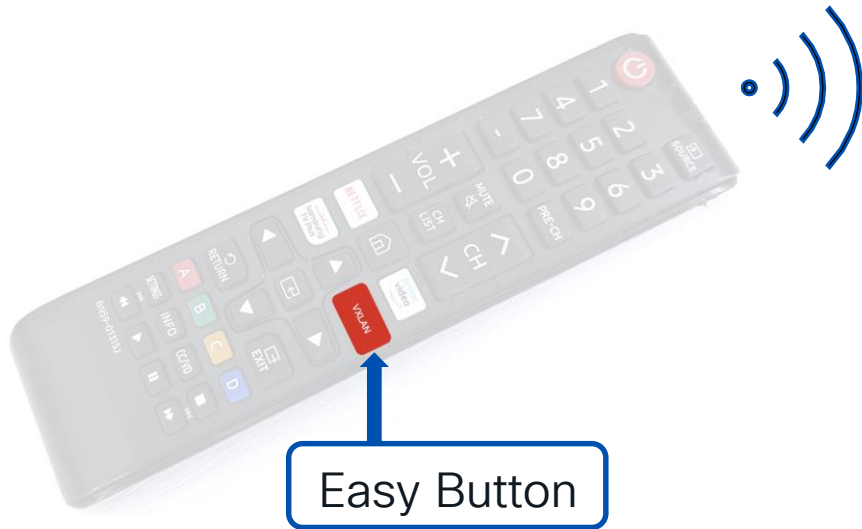
evpn
  vni 10011 l2
    rd auto
    route-target import auto
    route-target export auto

interface Vlan10
  no shutdown
  vrf member Tenant-1
  ip forward

interface Vlan11
  no shutdown
  vrf member Tenant-1
  ip address 10.0.11.1/24
  fabric forwarding mode anycast-gateway

interface nve1
  no shutdown
  source-interface loopback1
  host-reachability protocol bgp
  member vni 10000 associate-vrf
  member vni 10011
  mcast-group 239.0.0.11
```

# Levels of Complexity – NDFC Controller



# Ansible Inventory

ansible



inventory.yml



group\_vars



roles



build\_fabric.yml

```
---  
# main inventory file
```

```
all:
```

```
vars:
```

```
ansible_connection: ansible.netcommon.httpapi  
ansible_user: "ndfc_username"  
ansible_password: !vault | (ndfc_password)  
ansible_network_os: cisco.dcnm.dcnm
```

```
children:
```

```
ndfc:
```

```
hosts:
```

```
10.15.0.11:
```

```
dcnm:
```

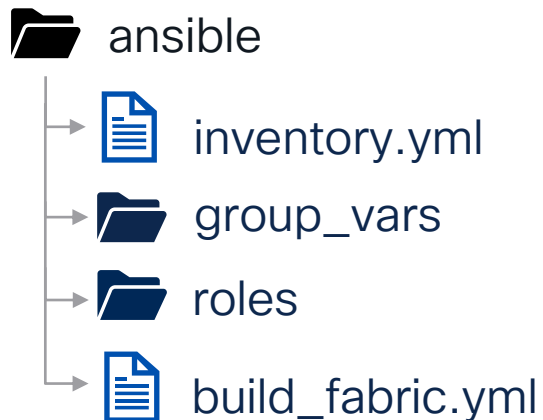
```
hosts:
```

```
10.18.1.14:
```

Connection  
information  
for NDFC



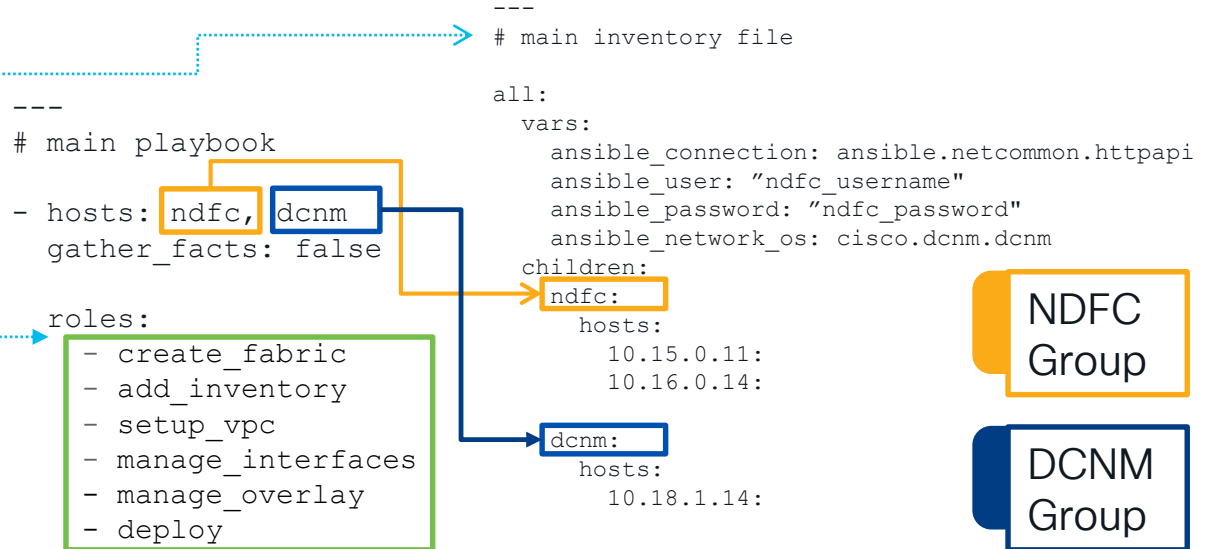
# Ansible Playbook



```
---  
# main NDFC playbook  
  
- name: Build VXLAN EVPN Fabric on NDFC  
  hosts: ndfc  
  gather_facts: false  
  
  roles:  
    - create_fabric  
    - add_inventory  
    - setup_vpc  
    - manage_interfaces  
    - manage_overlay  
    - deploy
```

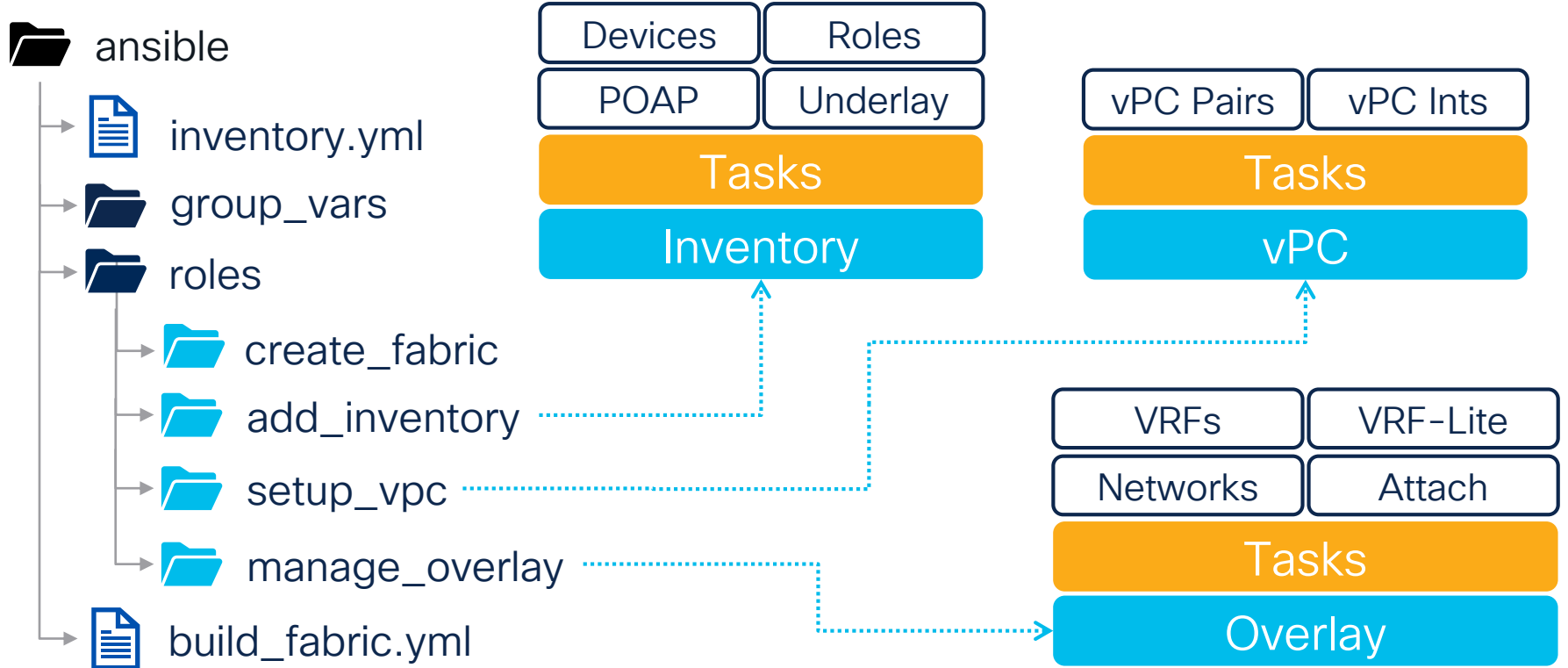
# Ansible Playbook Relationships

ansible



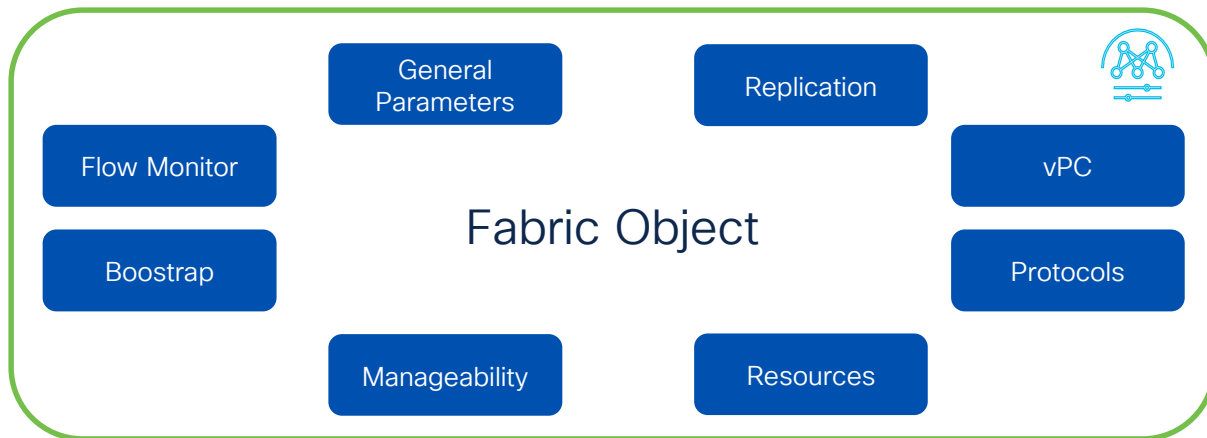
# Ansible Roles – Tasks

How to do it





# Workflow Step1: Create VXLAN Fabric





# Workflow Step1: Create VXLAN Fabric

Actions ^

- Create Fabric
- Edit Fabric
- Delete Fabric

## Create Fabric

Fabric Name

Pick Fabric  
[Data Center VXLAN EVPN >](#)

**General Parameters** | Replication | vPC | Protocols | Advanced | Resources | Manageability | Bootstrap | Configuration Backup | Flow Monitor

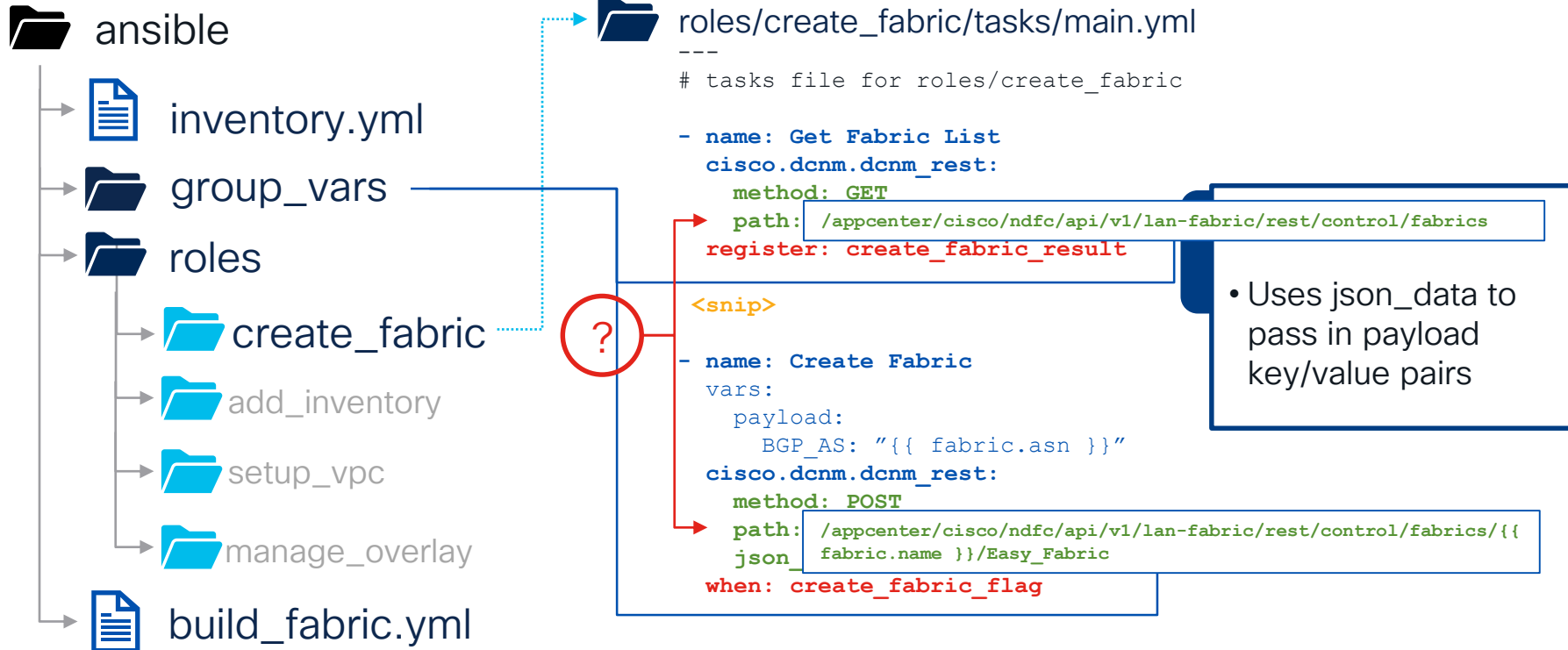
**BGP ASN\***

Enable IPv6 Underlay  
 If not enabled, IPv4 underlay is used

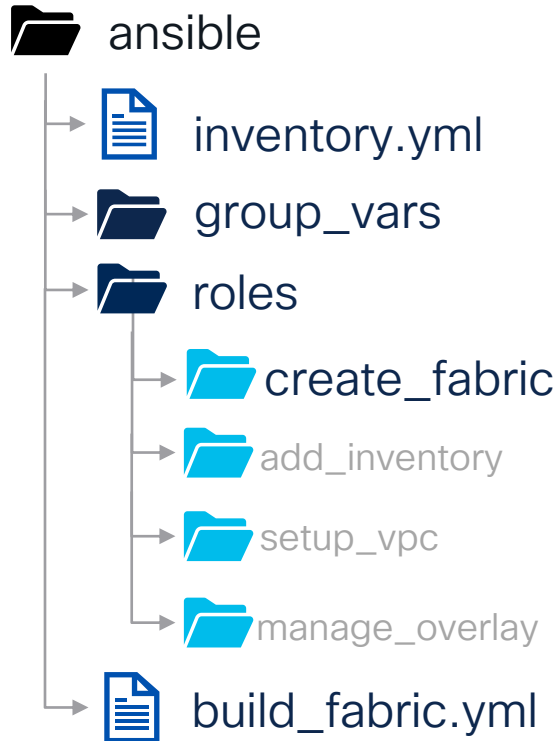
Enable IPv6 Link-Local Address  
 If not enabled, Spine-Leaf interfaces will use global IPv6 addresses

Fabric Interface Numbering\*  
 Numbered(Point-to-Point) or Unnumbered

# Ansible Roles – Fabric Tasks



# Ansible Roles – Fabric Tasks



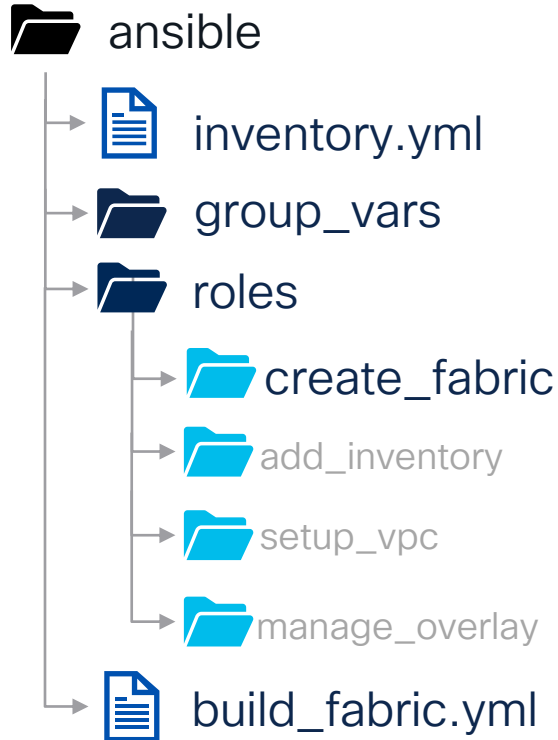
roles/create\_fabric/tasks/main.yml

```
- name: Intialize create_fabric_flag <snip>
  ansible.builtin.set_fact:
    create_fabric_flag: true

- name: Check If Fabric Exists
  ansible.builtin.set_fact:
    create_fabric_flag: false
  when: item.fabricName == fabric.name
  loop: "{{ create_fabric_result.response.DATA }}"
  loop_control:
    label: "{{ item.fabricName }}"

- name: Check If Fabric Exists Log
  ansible.builtin.debug:
    msg: "Fabric {{ fabric.name }} Already Exists"
  when: not create_fabric_flag
```

# Ansible Roles – Fabric Tasks



roles/create\_fabric/tasks/main.yml

```
---
# tasks file for roles/create_fabric

- name: Create Fabric with Module
  cisco.dcnm.dcnm_fabric:
    state: merged
    config:
      - FABRIC_NAME: CL_STAGING
        FABRIC_TYPE: VXLAN_EVPN
        BGP_AS: 65000
        ANYCAST_GW_MAC: 0001.aabb.ccdd
        UNDERLAY_IS_V6: false
```

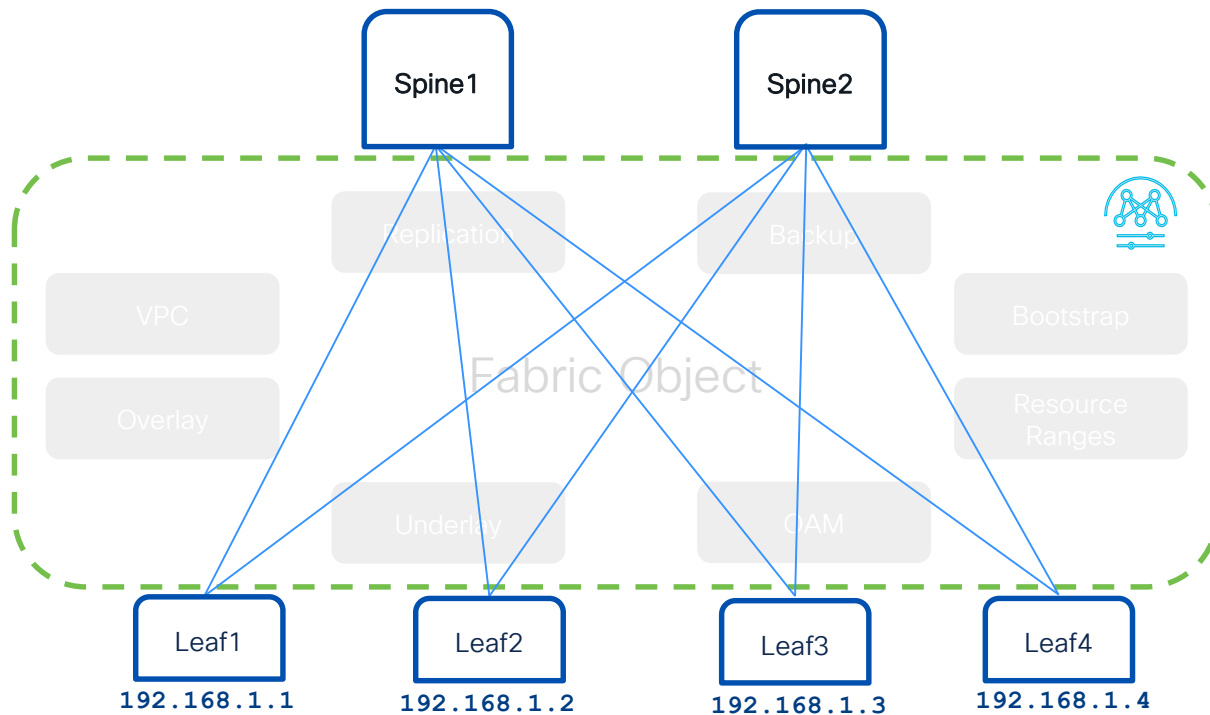


New Module!  
(3.5.0)

- Creates Fabric
- Handles Mutually Exclusive Properties
- Supports multiple types (VXLAN\_EVPN, MSD, LAN\_CLASSIC)



# Workflow Step2: Add Inventory





# Workflow Step2: Add Inventory

Fabric Overview - fabric-stage

Overview **Switches** Links Interfaces Interface Groups Policies Networks VRFs Services Event Analytics History Resources

Filter by attributes

Switch	IP Address	Role	Serial Number	Mode	Config Status	Oper Status	Discovery Status	Model	VPC Role	VPC Peer
<input type="checkbox"/> staging-leaf1	10.15.10.12	Leaf	9QRB2LQLD3H	Normal	In-Sync	Major	Ok	N9K-C9300v	Primary	staging-leaf2
<input type="checkbox"/> staging-leaf2	10.15.10.13	Leaf	9CJCO82GQEB	Normal	In-Sync	Major	Ok	N9K-C9300v	Secondary	staging-leaf1
<input type="checkbox"/> staging-leaf3	10.15.10.14	Border	9OIZMJFK60Z	Normal	In-Sync	Minor	Ok	N9K-C9300v		
<input type="checkbox"/> staging-spine1	10.15.10.11	Spine	9622UOESRG4	Normal	In-Sync	Minor	Ok	N9K-C9500v		

Actions

- Add Switches
- Preview
- Deploy
- Discovery
- Set Role
- VPC Pairing
- ToR/Access Pairing

! (Warning icon)

Add Switches - Fabric: fabric-stage

Switch Addition Mechanism\*  
 Discover  Bootstrap  Pre-provision

Seed Switch Details

Select IP\*  
10.122.84.35  
Ex: "2.2.2.20" or "10.10.10.40-60" or "2.2.2.20, 2.2.2.21"

Authentication Protocol\*  
MDS

Username\*  
admin

Password\*  
admin

Max hops\*  
3

Reverse Config

Unchecking this will clean up the configuration on switches



# Workflow Step2: Add Inventory

Overview **Switches** Links Interfaces

Filter by attributes

<input type="checkbox"/> Switch	IP Address
<input type="checkbox"/> staging-leaf2	10.15.713
<input type="checkbox"/> staging-spine1	10.15.711

1

Actions ^

- Add Switches
- Preview
- Deploy
- Discovery >
- Set Role
- vPC Pairing
- ToR/Access Pairing
- vPC Overview
- More >

2

Switch Addition Mechanism\*  
Discover

**Seed Switch Details**

Seed IP\*  
10.15.712

Ex: "2.2.2.20" or "10.10.10.40-60" or "2.2.2.20, 2.2.2.21"

Authentication Protocol\*  
MD5

Username\*  
admin

Password\*  
\*\*\*\*\*

Max Hops\*  
2

Preserve Config

Unchecking this will clean up the configuration on switch(es)

3

4

Close Add Switches

Fabric Overview - fabric-stage

Overview **Switches** Links Interfaces Interface Groups Policies Networks

Filter by attributes

<input type="checkbox"/> Switch	IP Address	Role	Serial Number	Mode
<input type="checkbox"/> staging-leaf1	10.15.712	Leaf	90VV2DNDLFJ	Migration

5

Select Role

Search Role

- Spine
- Leaf (current)
- Border
- Border Spine
- Border Gateway
- Border Gateway Spine
- Super Spine
- Border Super Spine
- Border Gateway Super Spine
- ToR

6

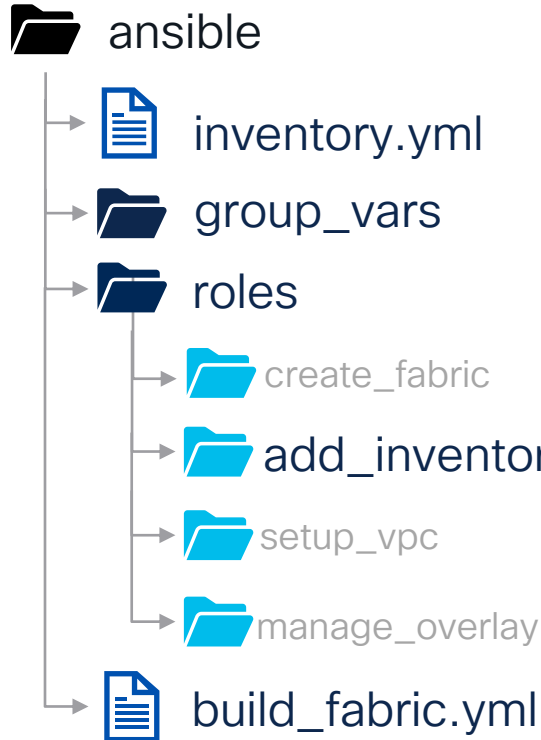
7

Actions ^

- Edit Fabric
- Add Switches
- Recalculate and Deploy
- More >

Actions v

# Ansible Roles – Inventory Tasks



roles/add\_inventory/tasks/main.yml

```
---
# tasks file for roles/add_inventory

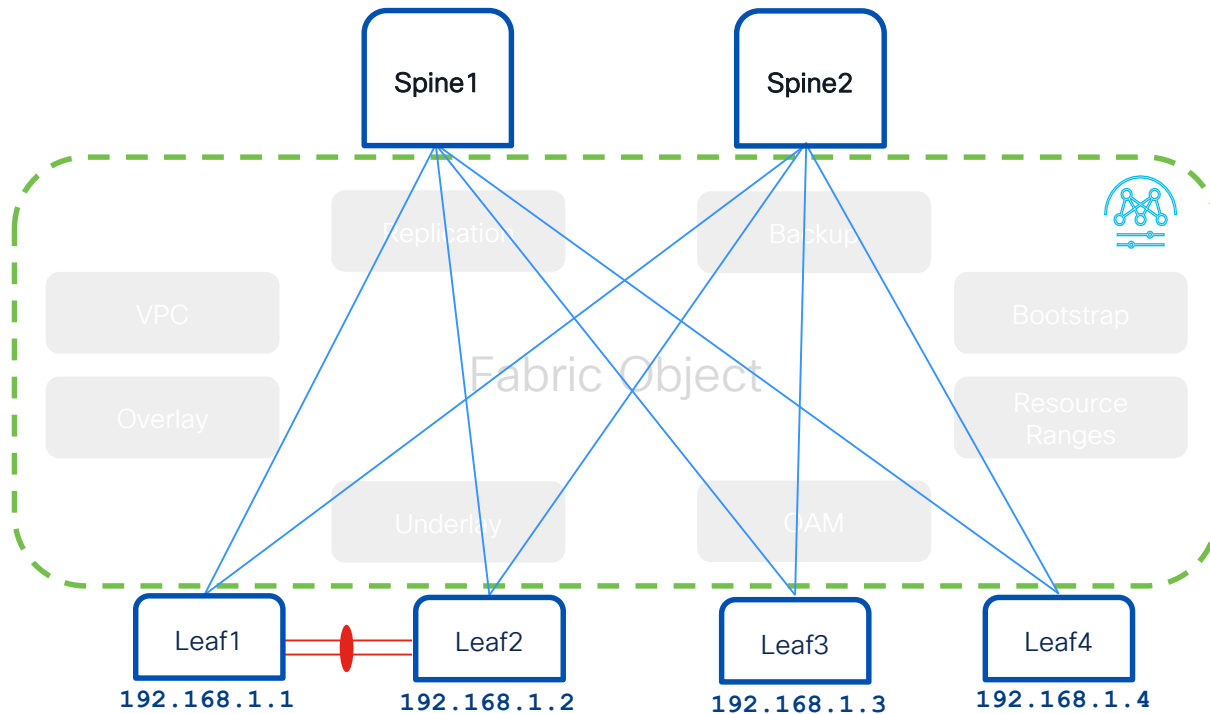
- name: Add Fabric Devices
  cisco.dcnm.dcnm_inventory:
    fabric: "{{ fabric.name }}"
    state: merged
    config:
      - seed_ip: 192.168.1.1
        role: spine
        < credentials >
      - seed_ip: 192.168.1.2
        role: leaf
        < credentials >
      - seed_ip: 192.168.1.4
        role: border
        < credentials >
    poap:
      - serial_number: 2A3BCDEFJKL
```

- Adds devices to the fabric
- Defines role in the fabric
- Deploy control

<snip>



# Workflow Step3: Setup VPC





# Workflow Step3: Setup VPC

Fabric Overview - fabric-stage

Actions

Overview Switches Links Interfaces Interface Groups Policies Networks VRFs Services Event Analytics History Resources

Filter by attributes

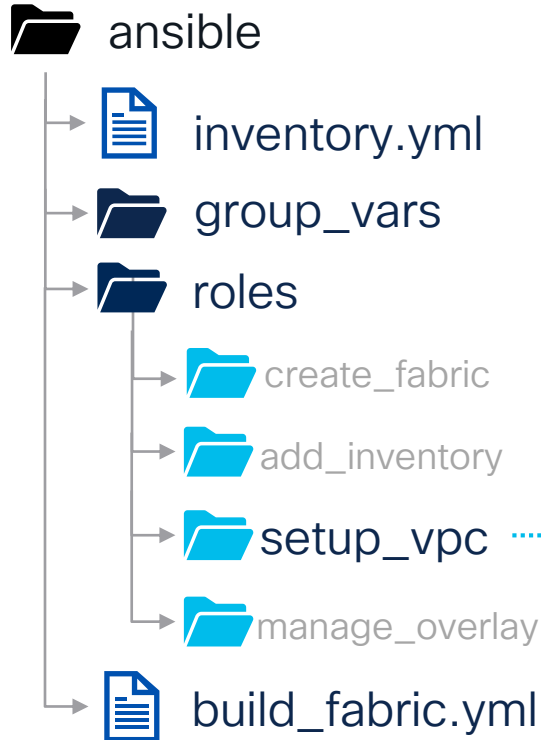
Actions ^

Switch	IP Address	Role	Serial Number	Mode	Config Status	Oper Status	Discovery Status	Model	VPC Role	VPC Peer
<input checked="" type="checkbox"/> staging-leaf1	10.15.7.12	Leaf	90VV2DNDLFJ	<span>Normal</span>	<span>In-Sync</span>	<span>Minor</span>	<span>Ok</span>	N9K-C9300v		
<input type="checkbox"/> staging-leaf2	10.15.7.13	Leaf	9VH4J9YPUMV	<span>Normal</span>	<span>In-Sync</span>	<span>Minor</span>	<span>Ok</span>	N9K-C9300v		
<input type="checkbox"/> staging-spine1	10.15.7.11	Spine	96IWSX5RHVM	<span>Normal</span>	<span>In-Sync</span>	<span>Minor</span>	<span>Ok</span>	N9K-C9500v		

- Add Switches
- Preview
- Deploy
- Discovery >
- Set Role
- vPC Pairing**
- ToR/Access Pairing
- vPC Overview
- More >



# Ansible Roles – vPC Tasks



```
roles/setup_vpc/tasks/main.yml
```

```
---
# tasks file for roles/inventory

- name: Add Fabric Devices
  cisco.dcnm.vpc_pair:
    src_fabric: "{{ fabric.name }}"
    state: merged
    config:
      - peerOneId: 192.168.1.1
        peerTwoId: 192.168.1.2
```

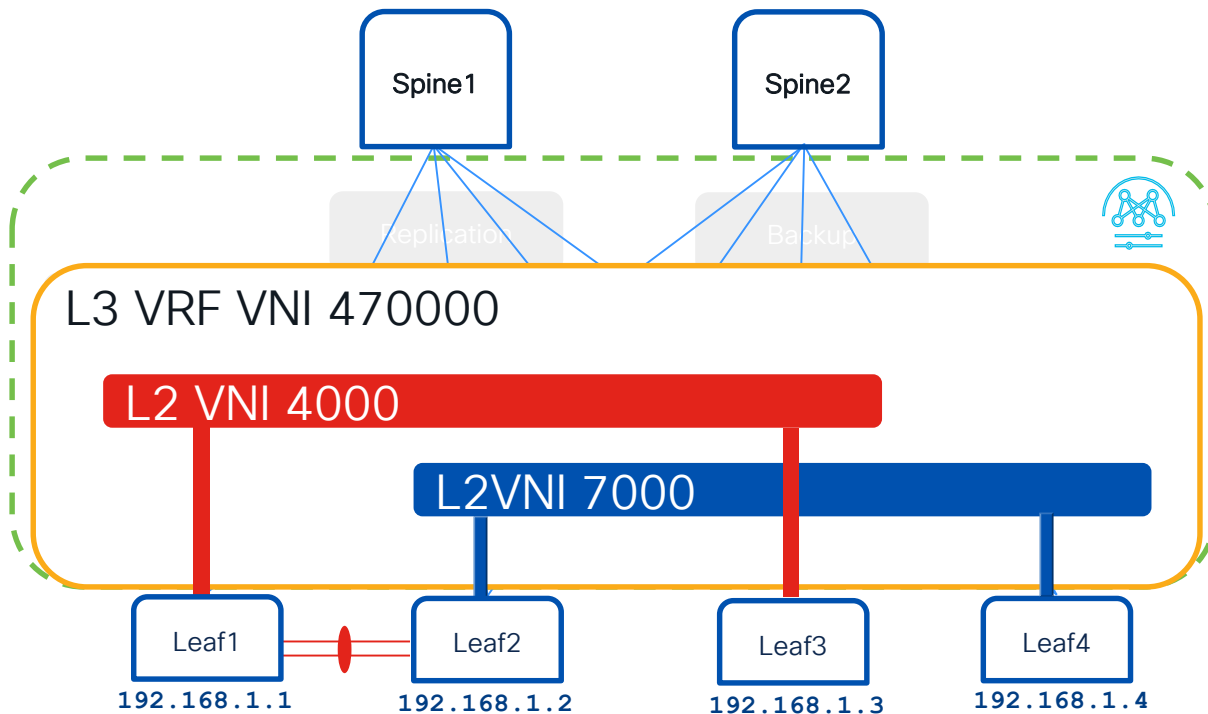


New Module!  
(3.5.0)

- Puts two leaf devices into a vPC pair
- Automatically discovers compatible devices and interfaces



# Workflow Step4: Manage Overlay



# Workflow Step4: Add Overlay – VRFs / Networks

**Create VRF**

VRF Name\*

VRF ID\*

VLAN ID  
 Propose VLAN

VRF Template\*  
[Default\\_VRF\\_Universal >](#)

VRF Extension Template\*  
[Default\\_VRF\\_Extension\\_Universal >](#)

[General Parameters](#) [Advanced](#) [Route Target](#)

**Create Network**

Network Name\*

Layer 2 Only

VRF Name\*  
 Create VRF

Network ID\*

VLAN ID  
 Propose VLAN

**VRF Overview - CL-VRF1**

[Overview](#) [VRF Attachments](#) [Networks](#)

**VRF Info**

VRF Name	VRF ID	VLAN ID	Status
CL-VRF1	470000	2055	DEPLOYED
Fabric Name	VRF Template	VRF Extension Template	VRF Description
fabric-stage	Default_VRF_Universal	Default_VRF_Extension_Universal	NA
L3VniMcastGroup			
NA			

**VRF Status**

3  
Status

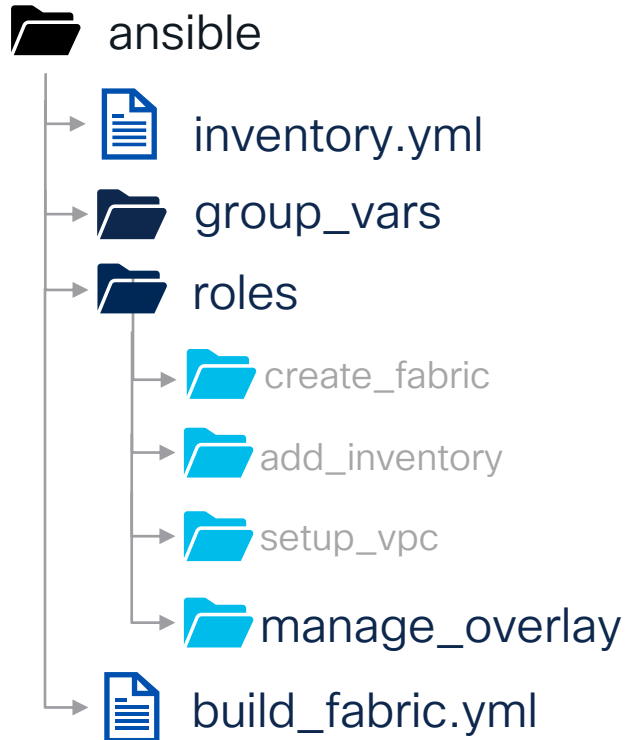
DEPLOYED 3

**Attached Roles Association**

3  
Roles

- border 1
- leaf 2

# Ansible Roles – VRF / Net Tasks



```

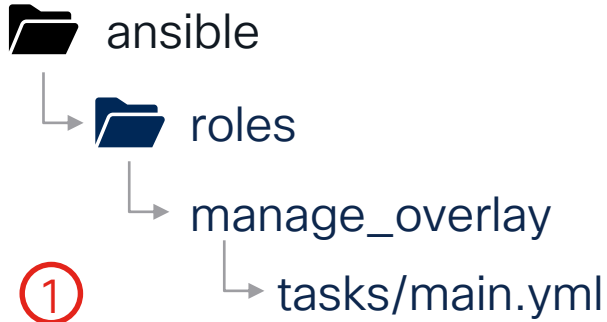
roles/manage_overlay/tasks/main.yml
---
# tasks file for roles/overlay
---
# tasks file for roles/overlay

- name: Add Overlay VRFs
  cisco.dcnm.dcnm_vrf:
    fabric: "{{ fabric.name }}"
    state: replaced
    config:
      - vrf_name: VRF1
        vrf_id: 47000
        vrf_vlan_id: 2055
        attach:
          - 192.168.1.1
          - 192.168.1.2
          - 192.168.1.3
          - 192.168.1.4

- name: Add Overlay Networks
  cisco.dcnm.dcnm_network:
    fabric: "{{ fabric.name }}"
    state: overridden
    config:
      - net_name: NET7000
        vrf_name: CL-F1
        net_id: 7000
        vrf_vlan_id: 88
        attach:
          - 192.168.1.1
          - 192.168.1.2
  
```

- Creates VRF and Network objects
- Attach and deploy VRF and Network config to fabric leaf devices
- Deploy Control

# Jinja2 for VRF Data



1

```
roles/manage_overlay/tasks/main.yml
1 ---
2 # -----
3 # CREATE VRF SECTION
4 # -----
5
6 - name: Create file to hold rendered VRF information
  ansible.builtin.template:
7   src: attach_vrfs.j2
8   dest: "{{ role_path }}/files/attach_vrfs.yml"
9
10
11 - name: Create and store generated VRF configuration
  ansible.builtin.set_fact:
12   vrf_config: "{{ lookup('file', 'attach_vrfs.yml') | from_yaml }}"
13
14
```

2

```
group_vars/all/overlay.yml
1 ---
2 vrfs:
3   - vrf_name: &refvrf_devnet vrf_devnet
4     vrf_id: 150001
5     vlan_id: 2000
6     attach_group: all_leaf
7
```

3

```
roles/manage_overlay/files/attach_vrfs.yml
1 ---
2 # This file is auto-generated
3 # DO NOT EDIT MANUALLY
4 #
5 - vrf_name: vrf_devnet
6   vrf_id: 150001
7   vlan_id: 2000
8   attach:
9     - ip_address: 10.15.10.12
10     - ip_address: 10.15.10.13
11   deploy: true
```

4

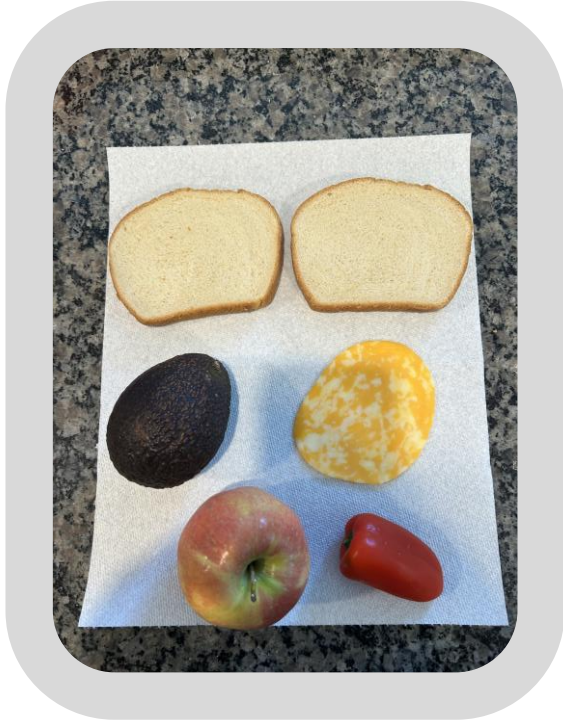
```
28 # -----
29 # Manage VRF Configuration on NDFC
30 # -----
31 - name: Manage NDFC Fabric VRFs
32   cisco.dcnm.dcnm_vrf:
33     fabric: "{{ fabric.name }}"
34     state: replaced
35     config: "{{ vrf_config }}"
```

```
roles/manage_overlay/templates/attach_vrfs.j2
1 ---
2 # This file is auto-generated
3 # DO NOT EDIT MANUALLY
4 #
5 {% for vrf in vrfs %}
6   - vrf_name: {{ vrf['vrf_name'] }}
7   {# ----- #}
8   {# Properties Section #}
9   {# ----- #}
10   vrf_id: {{ vrf['vrf_id'] }}
11   vlan_id: {{ vrf['vlan_id'] }}
12   {# ----- #}
13   {# Attach Group Section #}
14   {# ----- #}
15   attach:
16     {% for switch in vrf_attach_group.all_leaf %}
17       - ip_address: {{ switch['ip_address'] }}
18     {% endfor %}
19   deploy: true
20 {% endfor %}
```

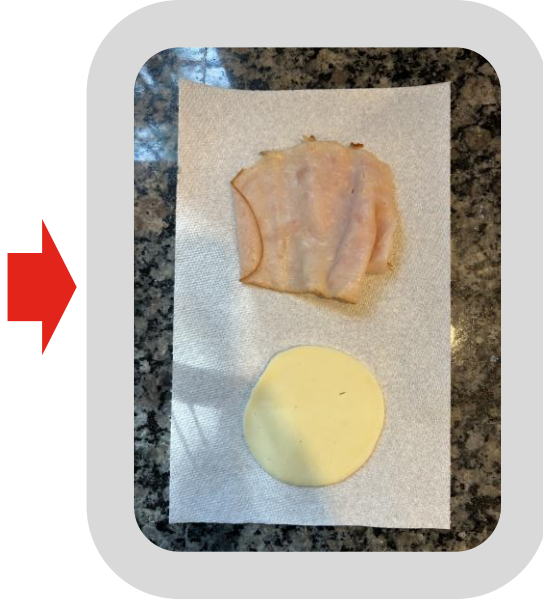


# How Does One Merge a Sandwich?

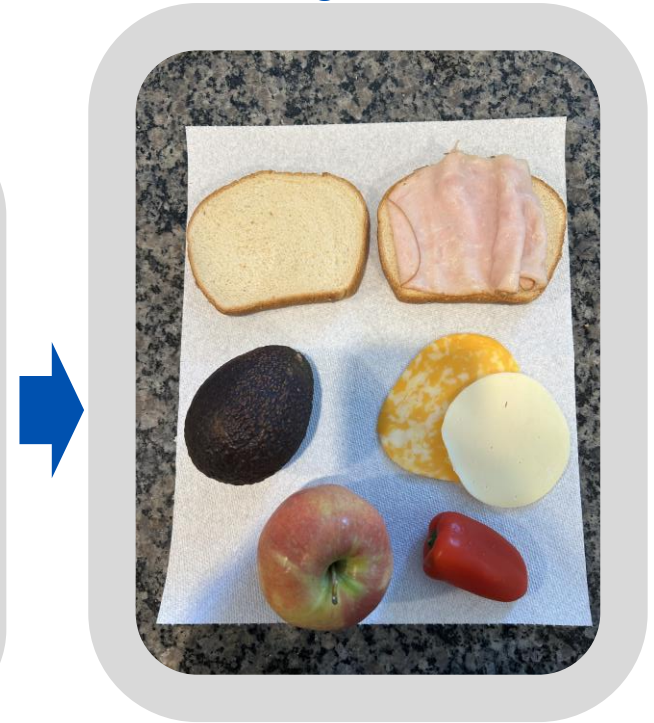
Initial State



Merge Operation




Merged State



# A Word About Module States - (Merged)



Playbook

CL-NET7000   
ANSIBLE


vrf\_name: CL24\_VRF1

gw\_ip\_address: 192.168.12.1/24

route\_tag: 12345



NDFC **Before** Merge

CL-NET7000 

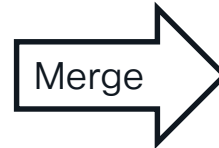
vrf\_name: CL23\_VRF1

gw\_ip\_address: 192.168.12.1/24


trm\_enable: False

Other Networks

Untouched



NDFC **After** Merge

CL-NET7000 

vrf\_name: CL24\_VRF1

gw\_ip\_address: 192.168.12.1/24

trm\_enable: False

route\_tag: 12345

Other Networks

Untouched

# Who Is The Source Of Truth?

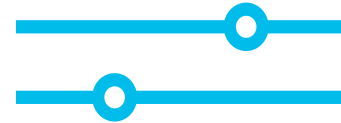
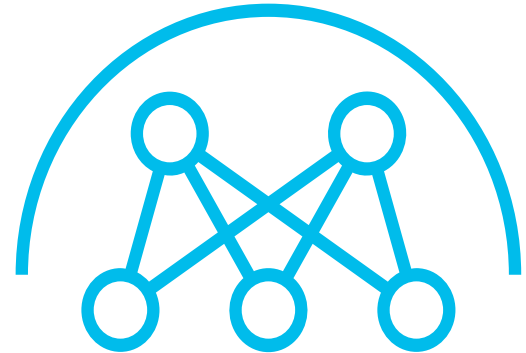
It's Me!



ANSIBLE



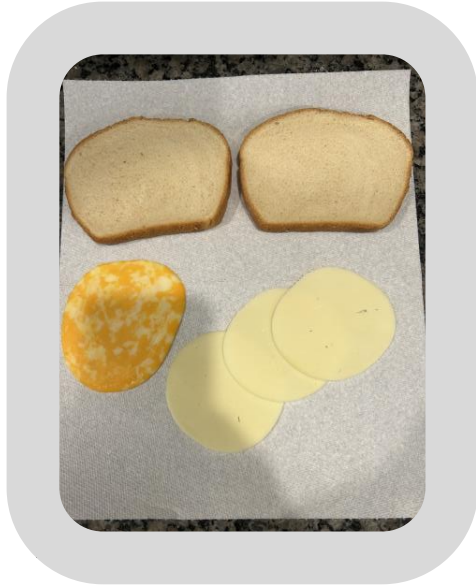
It's Me!



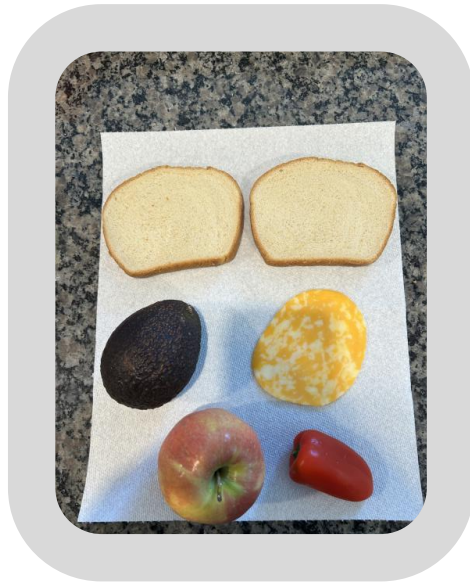
NDFC

# How Does One Replace a Sandwich?

Desired State



Actual State



Replace Operation




Replaced State



# A Word About Module States - (Replaced)



Playbook

CL-NET7000   
ANSIBLE


vrf\_name: CL24\_VRF1

gw\_ip\_address: 192.168.12.1/24

route\_tag: 12345



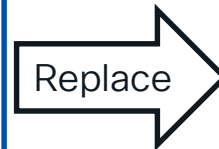
NDFC **Before** Replace

CL-NET7000 


vrf\_name: CL23\_VRF1

gw\_ip\_address: 192.168.12.1/24

trm\_enable: False



NDFC **After** Replace

CL-NET7000 

vrf\_description: CL24\_VRF1

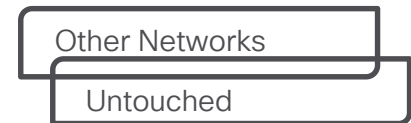
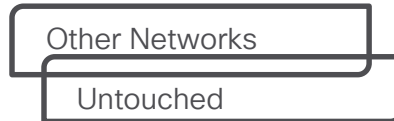
gw\_ip\_address: 192.168.12.1/24

~~trm\_enable: False~~

route\_tag: 12345

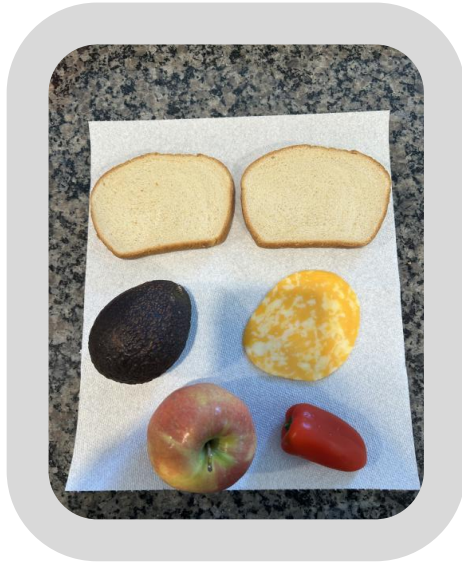


Source of Truth

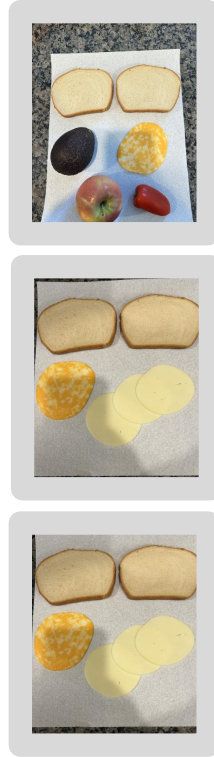


# How Does One **Override** a Sandwich?

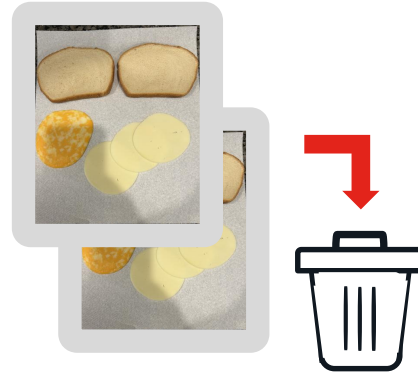
Desired State



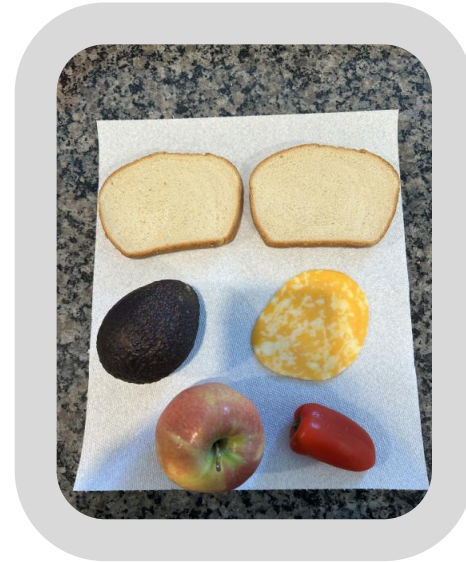
Actual State



Override Operation




Overridden State



# A Word About Module States - (Overridden)



Playbook

CL-NET7000   
ANSIBLE


vrf\_name: CL24\_VRF1

gw\_ip\_address: 192.168.12.1/24

route\_tag: 12345

State: Overridden

NDFC **Before** Overridden

CL-NET7000 


vrf\_name: CL23\_VRF1

gw\_ip\_address: 192.168.12.1/24

trm\_enable: False

Override

NDFC **After** Overridden

CL-NET7000 

vrf\_description: CL24\_VRF1

gw\_ip\_address: 192.168.12.1/24

~~trm\_enable: False~~

route\_tag: 12345



Source of Truth

NETWORK 2

NETWORK 3

~~NETWORK 2~~

~~NETWORK 3~~

# Putting It All Together

## Where to do it



inventory.yml

```
---
# main inventory file

all:
  <snip>
  children:
    ndfc:
      hosts:
        10.15.1.11:
        10.15.1.12:
    dcnm:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

## What to do



build\_fabric.yml

```
---
# main playbook

- hosts: ndfc, dcnm
  gather_facts: false

  roles:
    - create_fabric
    - add_inventory
    - setup_vpc
    - manage_overlay
    - manage_interfaces
    - deploy
```

## How to do it



roles/manage\_overlay/tasks/main.yml

```
---
# tasks file for roles/overlay

- name: Configure Overlay VRFs
  cisco.dcnm.dcnm_vrf:
    fabric: "{{ fabric.name }}"
    state: overridden
  config:
    - vrf_name: CL-VRF1
      vrf_id: 470000
      <snip>

- name: Configure Overlay Networks
  cisco.dcnm.dcnm_network:
    fabric: "{{ fabric.name }}"
    state: replaced
    config: "{{ networks }}"
```

## Data to do it

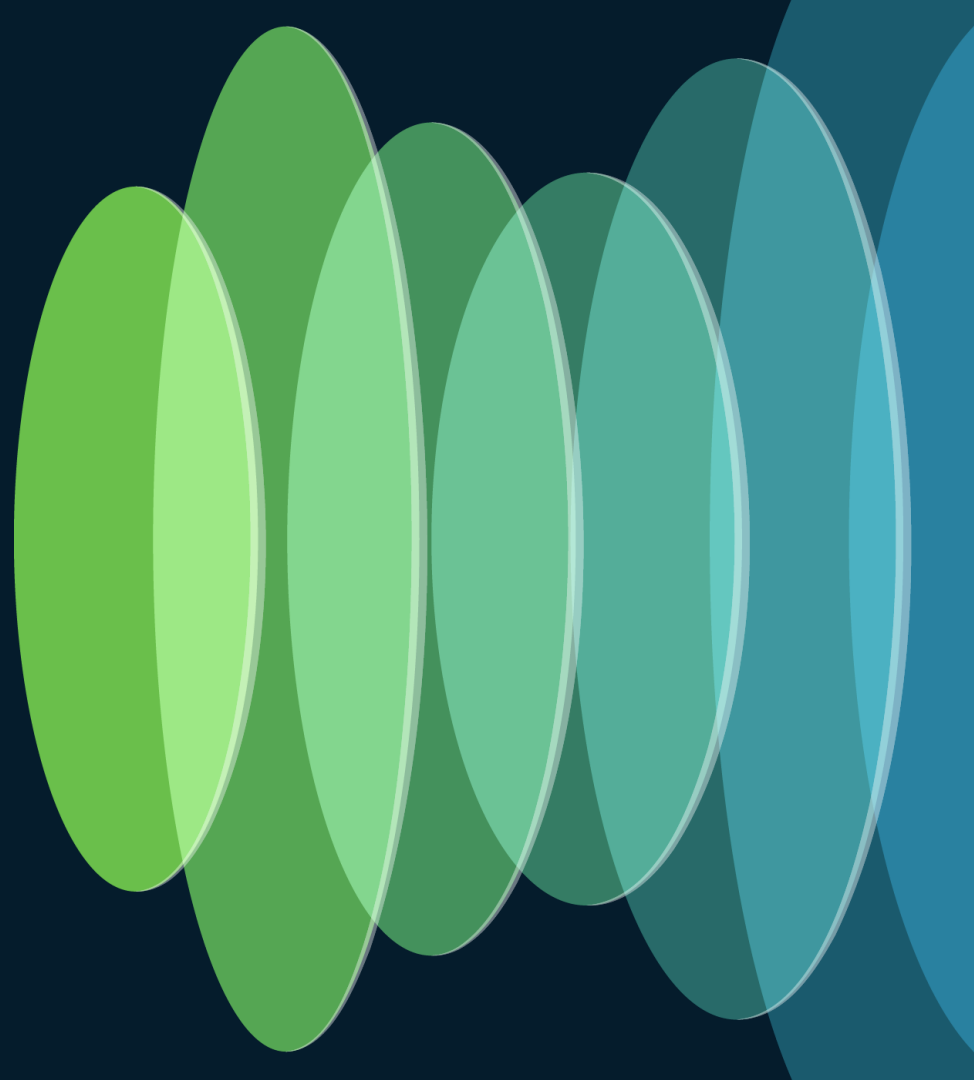


group\_vars/ndfc.yml


```
---
# var file for leafs group
fabric:
  name: fabric-stage
  asn: 5588
  cpath: "/appcenter/{...}"

networks:
  - net_name: CL-NET4000
    vrf_name: CL-VRF1
    net_id: 4000
    vlan_id: 55
    attach:
      - 192.168.1.1
      - 192.168.1.3
  - net_name: CL-NET7000
    vrf_name: CL-VRF1
    net_id: 7000
    vlan_id: 88
    attach:
      - 192.168.1.2
      - 192.168.1.4
```


# CI/CD Pipeline Demo



# IaC NDFC Pipeline Demo



ANSIBLE

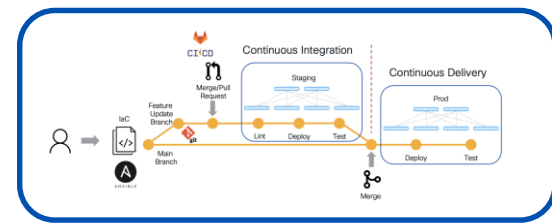


1

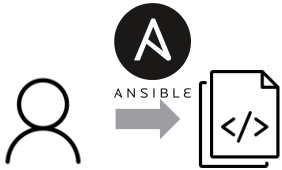
- Add Networks group\_vars
- State: Overridden
- Commit / Push changes to GitLab Staging branch




```
group_vars
networks:
- net_name: AnsibleNet1
  vrf_name: *refvrf_ansible
  net_id: 130001
  vlan_id: 2301
  vlan_name: AnsibleNet1_vlan2301
  gw_ip_subnet: "192.168.11.1/24"
  attach: []
  attach_group: esxi
- net_name: AnsibleNet2
  vrf_name: *refvrf_ansible
  net_id: 130002
  vlan_id: 2302
  vlan_name: AnsibleNet2_vlan2302
  gw_ip_subnet: "192.168.12.1/24"
  attach_group: esxi
- net_name: GitLabNet1
  vrf_name: *refvrf_ansible
  net_id: 130003
  vlan_id: 2303
  vlan_name: GitLabNet1_vlan2303
  gw_ip_subnet: "192.168.13.1/24"
  attach: []
  attach_group: esxi
- net_name: GitLabNet2
  vrf_name: *refvrf_ansible
  net_id: 130004
  vlan_id: 2304
  vlan_name: GitLabNet2_vlan2304
  gw_ip_subnet: "192.168.14.1/24"
  attach_group: esxi
#
# New Networks Added Below
#
- net_name: CL-NET4000
  vrf_name: *refvrf_ansible
  net_id: 134000
  vlan_id: 2400
  vlan_name: CL-NET4000_vlan2400
  gw_ip_subnet: "192.168.40.1/24"
  attach_group: esxi
- net_name: CL-NE74000
  vrf_name: *refvrf_ansible
  net_id: 137000
  vlan_id: 2700
  vlan_name: CL-NET7000_vlan2700
  gw_ip_subnet: "192.168.70.1/24"
  attach_group: esxi
```



# IaC NDFC Pipeline Demo



1

- Add Networks group\_vars 
- State: Overridden
- Commit / Push changes to GitLab **Staging** branch

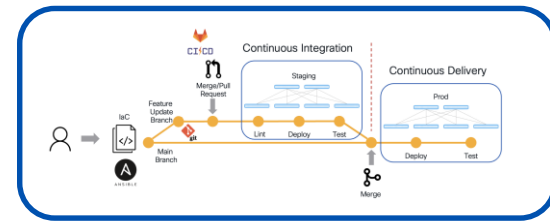
2

- Open Merge Request 
- Triggers **Staging** Pipeline for Deploy and Verify

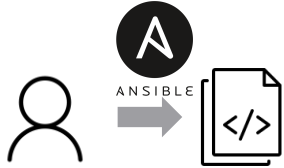
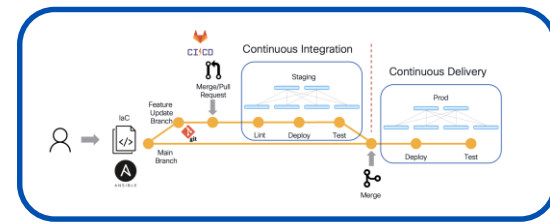
 .gitlab-ci.yml

```

deploy_on_stage:
  stage: deploy
  image:
    name: "cytopia/ansible"
    entrypoint: [""]
  rules:
    - if: $CI_PIPELINE_SOURCE == 'merge_request_event' && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME == "main"
  before_script:
    - export ANSIBLE_PERSISTENT_COMMAND_TIMEOUT=1000
    - export ANSIBLE_PERSISTENT_CONNECT_TIMEOUT=1000
    - python3 -m pip install requests
    - python3 -m pip install jmespath
  script:
    - ansible-playbook --version
    - ansible-galaxy -vvv collection install cisco.dcm
    - ansible-playbook -i hosts.stage.yml build_fabric.yml
  
```



# IaC NDFC Pipeline Demo



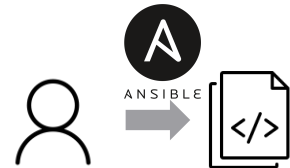
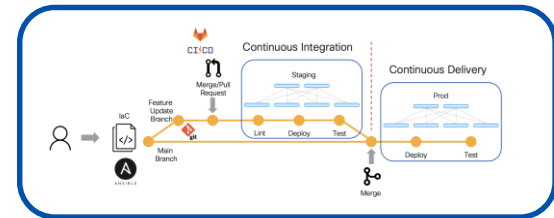
`.gitlab-ci.yml`

1

- Add Networks group\_vars
- State: Overridden
- Commit / Push changes to GitLab Staging branch

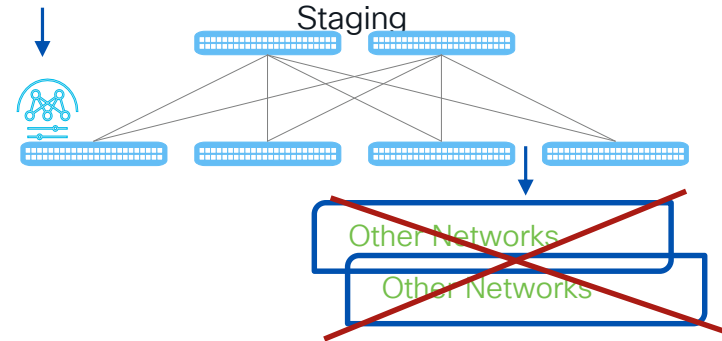
```
deploy_on_stage:
  stage: deploy
  image:
    name: "cytopia/ansible"
    entrypoint: [""]
  rules:
    - if: $CI_PIPELINE_SOURCE == 'merge_request_event' && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME == "main"
  before_script:
    - export ANSIBLE_PERSISTENT_COMMAND_TIMEOUT=1000
    - export ANSIBLE_PERSISTENT_CONNECT_TIMEOUT=1000
    - python3 -m pip install requests
    - python3 -m pip install jmespath
  script:
    - ansible-playbook --version
    - ansible-galaxy -vvv collection install cisco.dcnm
    - ansible-playbook -i hosts.stage.yml build_fabric.yml
```

# IaC NDFC Pipeline Demo



- Add Networks group\_vars
  - State: Overridden
  - Commit / Push changes to GitLab **Staging** branch

- Open Merge Request
  - Triggers **Staging** Pipeline for Deploy and Verify



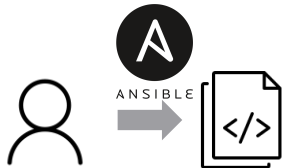
```

.gitlab-ci.yml


deploy_on_stage:
  stage: deploy
  image:
    name: "cytopia/ansible"
    entrypoint: [""]
  rules:
    - if: $CI_PIPELINE_SOURCE == 'merge_request_event' && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME == "main"
  before_script:
    - export ANSIBLE_PERSISTENT_COMMAND_TIMEOUT=1000
    - export ANSIBLE_PERSISTENT_CONNECT_TIMEOUT=1000
    - python3 -m pip install requests
    - python3 -m pip install jmespath
  script:
    - ansible-playbook --version
    - ansible-galaxy -vvv collection install cisco.dcm
    - ansible-playbook -i hosts.stage.yml build_fabric.yml
  
```





# IaC NDFC Pipeline Demo



1

- Add Networks group\_vars 
- State: Overridden
- Commit / Push changes to GitLab **Staging** branch

2

- 
- Open Merge Request 
  - Triggers **Staging** Pipeline for Deploy and Verify

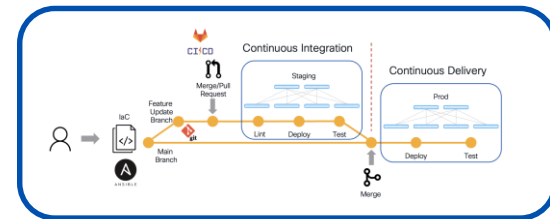
3

- 
- Click Merge 
  - Triggers **Production** Pipeline for Deploy and Verify



.gitlab-ci.yml

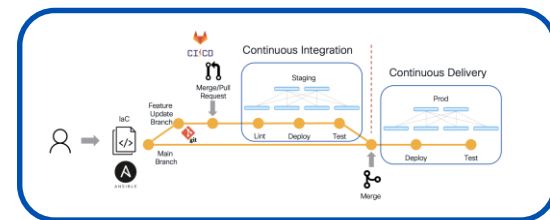
```
deploy_on_prod:
  stage: deploy
  image:
    name: "cytopia/ansible"
    entrypoint: ["" ]
  rules:
    - if: '$CI_PIPELINE_SOURCE == "push" && $CI_COMMIT_BRANCH == "main"'
  before_script:
    - export ANSIBLE_PERSISTENT_COMMAND_TIMEOUT=1000
    - export ANSIBLE_PERSISTENT_CONNECT_TIMEOUT=1000
    - python3 -m pip install requests
    - python3 -m pip install jmespath
  script:
    - ansible-playbook --version
    - ansible-galaxy -vvv collection install cisco.dcnm
    - ansible-playbook -i hosts.prod.yml build_fabric.yml
```




# IaC NDFC Pipeline Demo



.gitlab-ci.yml



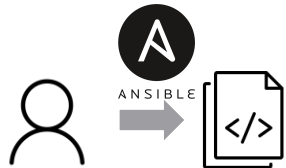
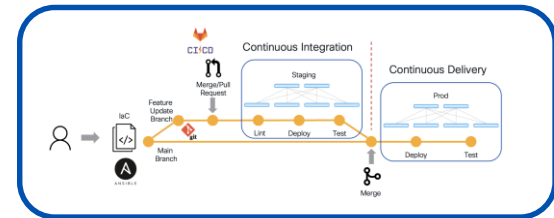
1

- Add Networks group\_vars 
- State: Overridden
- Commit / Push changes to GitLab **Staging** branch

2

```
deploy_on_prod:
  stage: deploy
  image:
    name: "cytopia/ansible"
    entrypoint: [""]
  rules:
    - if: '$CI_PIPELINE_SOURCE == "push" && $CI_COMMIT_BRANCH == "main"'
  before_script:
    - export ANSIBLE_PERSISTENT_COMMAND_TIMEOUT=1000
    - export ANSIBLE_PERSISTENT_CONNECT_TIMEOUT=1000
    - python3 -m pip install requests
    - python3 -m pip install jmespath
  script:
    - ansible-playbook --version
    - ansible-galaxy -vvv collection install cisco.dcnm
    - ansible-playbook -i hosts.prod.yml build_fabric.yml
```

# IaC NDFC Pipeline Demo



- 1**
- Add Networks group\_vars
  - State: Overridden
  - Commit / Push changes to GitLab **Staging** branch

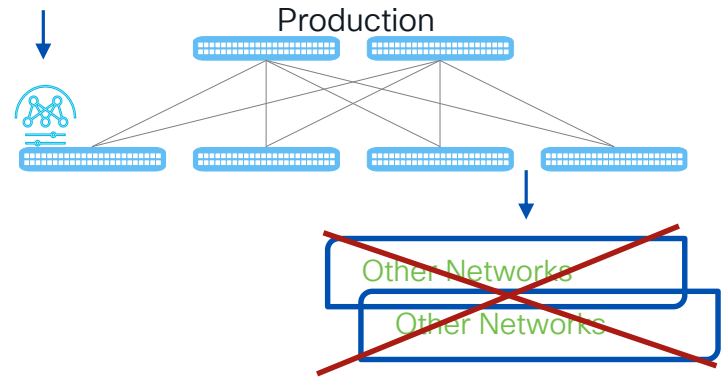
- 2**
- Open Merge Request
  - Triggers **Staging** Pipeline for Deploy and Verify

- 3**
- Click Merge
  - Triggers **Production** Pipeline for Deploy and Verify

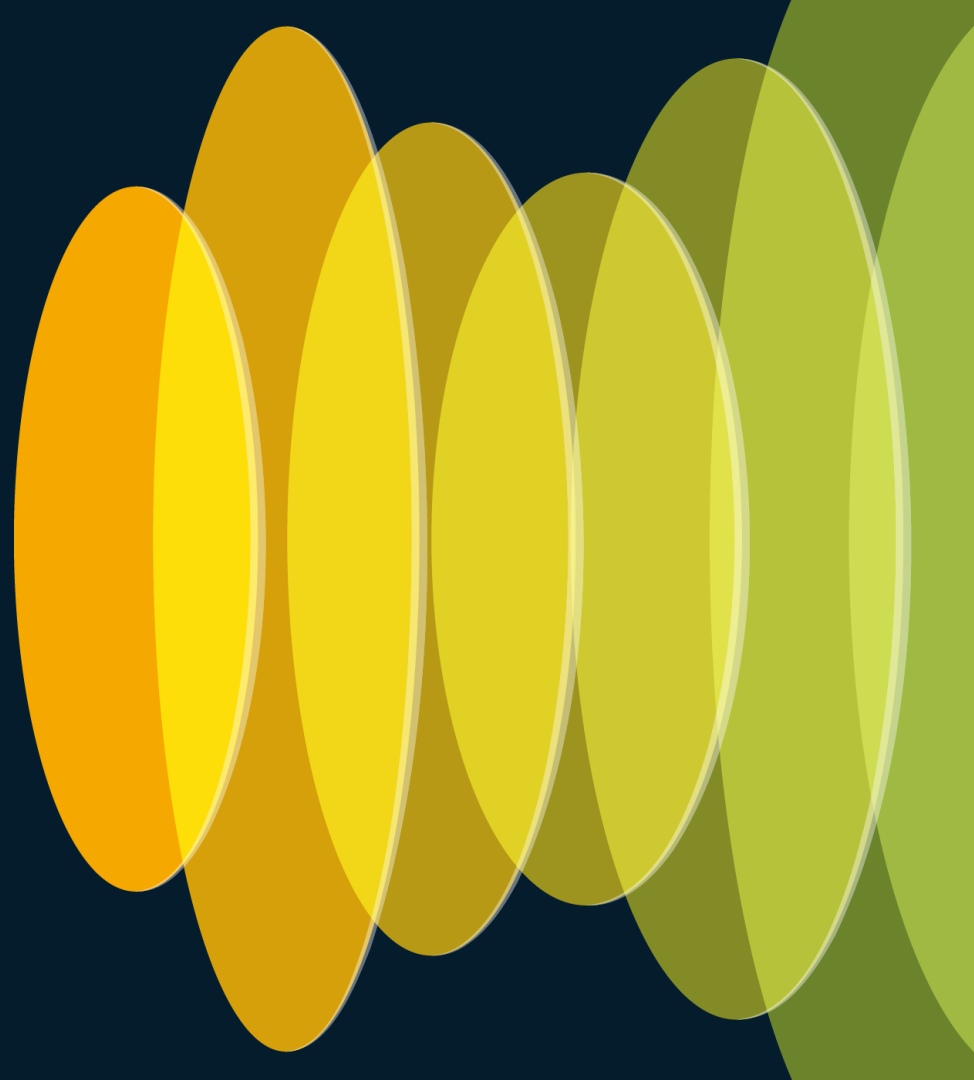


```

.gitlab-ci.yml
deploy_on_prod:
  stage: deploy
  image:
    name: "cytopia/ansible"
    entrypoint: [""]
  rules:
    - if: '$CI_PIPELINE_SOURCE == "push" && $CI_COMMIT_BRANCH == "main"'
  before_script:
    - export ANSIBLE_PERSISTENT_COMMAND_TIMEOUT=1000
    - export ANSIBLE_PERSISTENT_CONNECT_TIMEOUT=1000
    - python3 -m pip install requests
    - python3 -m pip install jmespath
  script:
    - ansible-playbook --version
    - ansible-galaxy -vvv collection install cisco.dcnm
    - ansible-playbook -i hosts.prod.yml build_fabric.yml
    
```



# References to Start Your Journey



# Ansible for NXOS and NDFC Repos



BRKDCN-2946  
Session Repo

<https://github.com/mtarking/BRKDCN-2946>



NDFC Roles  
Repo

<https://github.com/allenrobel/ndfc-roles>

# What is your path?



I got this!

I need help?!

Many sessions  
@ [ciscolive.com](https://www.ciscolive.com)

with great material

also [DevNet](https://www.devnet.com)

& [developer.cisco.com](https://www.developer.cisco.com)

Many Cisco   
services

To assist you in your  
*automation journey*

Checkout

**Services as Code**

@ World of Solutions 

*cisco Live!*



# More Information

- <https://www.ansible.com/resources/get-started>
- [https://docs.ansible.com/ansible/latest/collections\\_guide/index.html](https://docs.ansible.com/ansible/latest/collections_guide/index.html)
- <https://galaxy.ansible.com/cisco/dcnm>
- <https://galaxy.ansible.com/cisco/nxos>
- <https://developer.cisco.com/docs/nexus-as-code/#!nx-os-with-ansible>
- <https://developer.cisco.com/docs/nexus-as-code/#!ndfc-with-ansible>

# More Information – Other Sessions/Labs

- DEWKS-3928: Build VXLAN Fabric with NDFC and Ansible
- BRKDCN-2929 (Simple VXLAN/EVPN Fabric Setup with Nexus Dashboard)
- BRKDCN-1619 (Introduction to NDFC: Simplifying Management of Your Data Center)
- BRKDCN-2988 (Design, Automate, and Manage Next-Gen Data Center VXLAN BGP EVPN Fabric with NDFC)

# Complete Your Session Evaluations



Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to **win 1 of 5 full conference passes** to Cisco Live 2025.

---



**Earn 100 points** per survey completed and compete on the Cisco Live Challenge leaderboard.

---



Level up and earn **exclusive prizes!**

---



Complete your surveys in the **Cisco Live mobile app.**

# Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at [www.CiscoLive.com/on-demand](https://www.CiscoLive.com/on-demand)



The bridge to possible

# Thank you

CISCO *Live!*

#CiscoLive