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!

Cisco Webex App

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

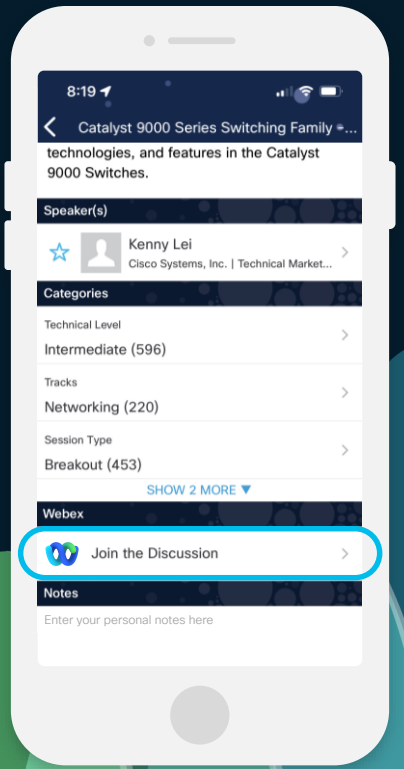
Questions?

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.

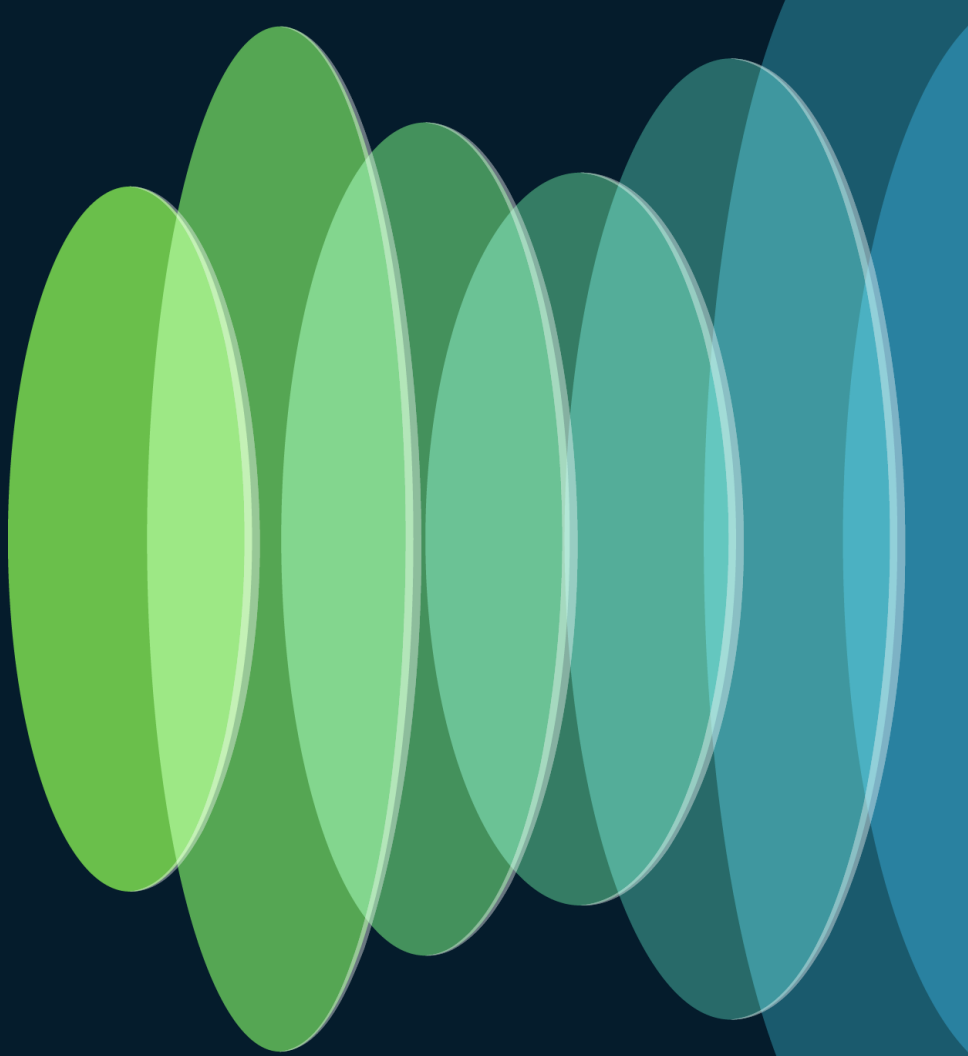


cisco Live!

Agenda

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

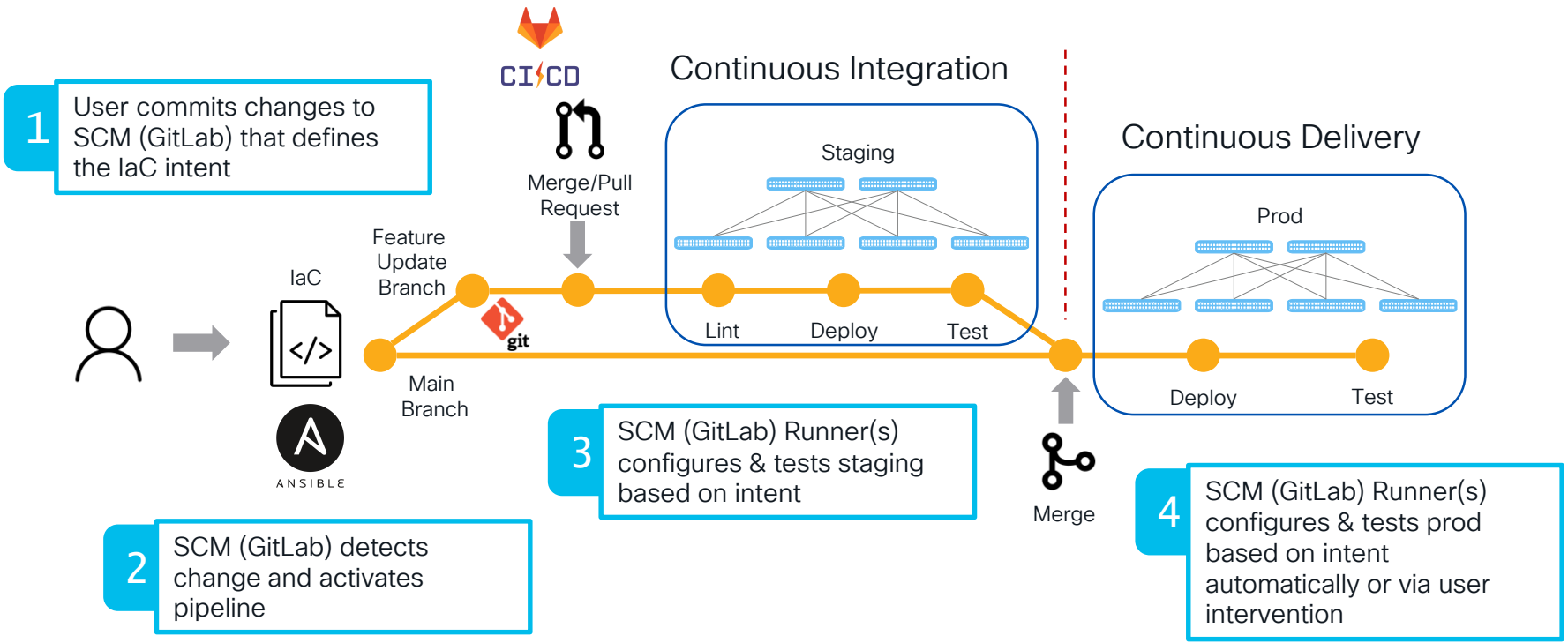
What is Infrastructure as Code?



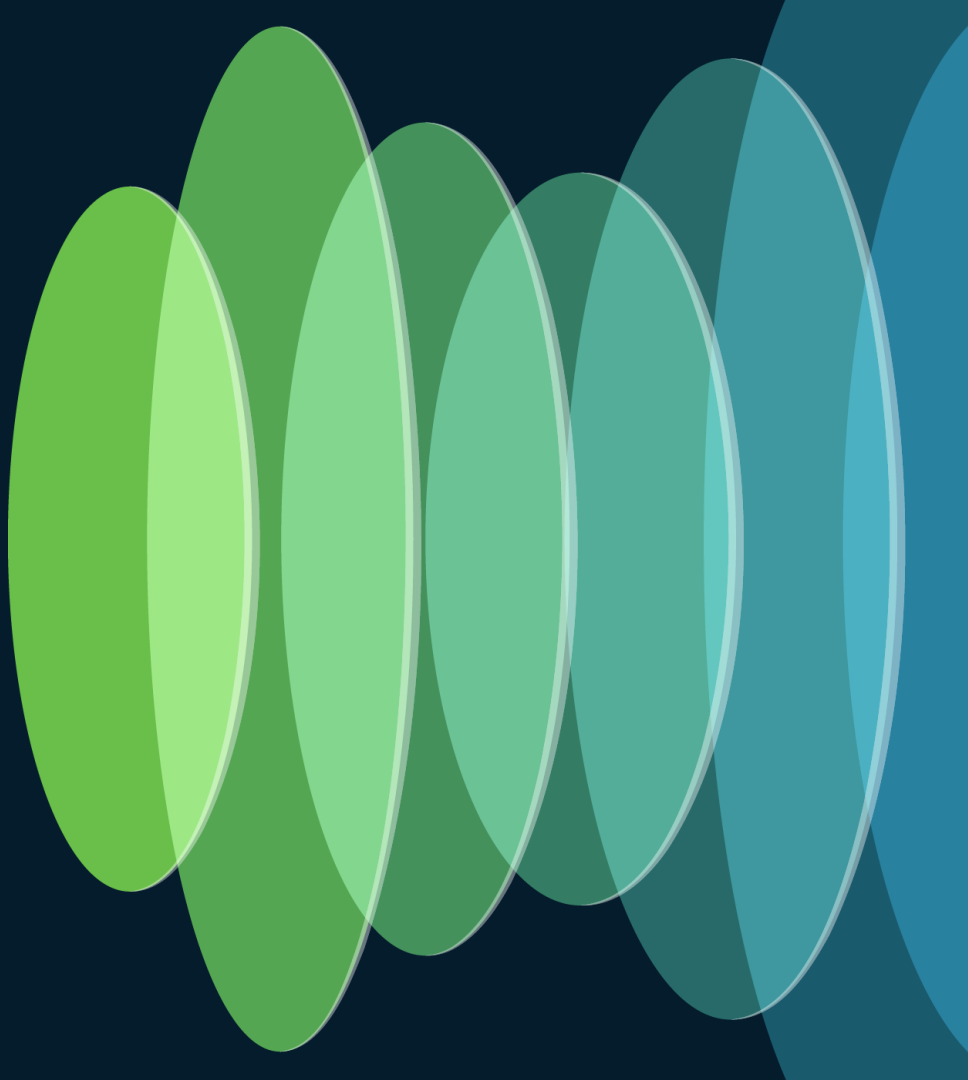
Infrastructure as Code for Network Ops

- Using “code” to provision and/or manage infrastructure
- Infrastructure as Code is not specific to a particular automation engine or specific programing 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



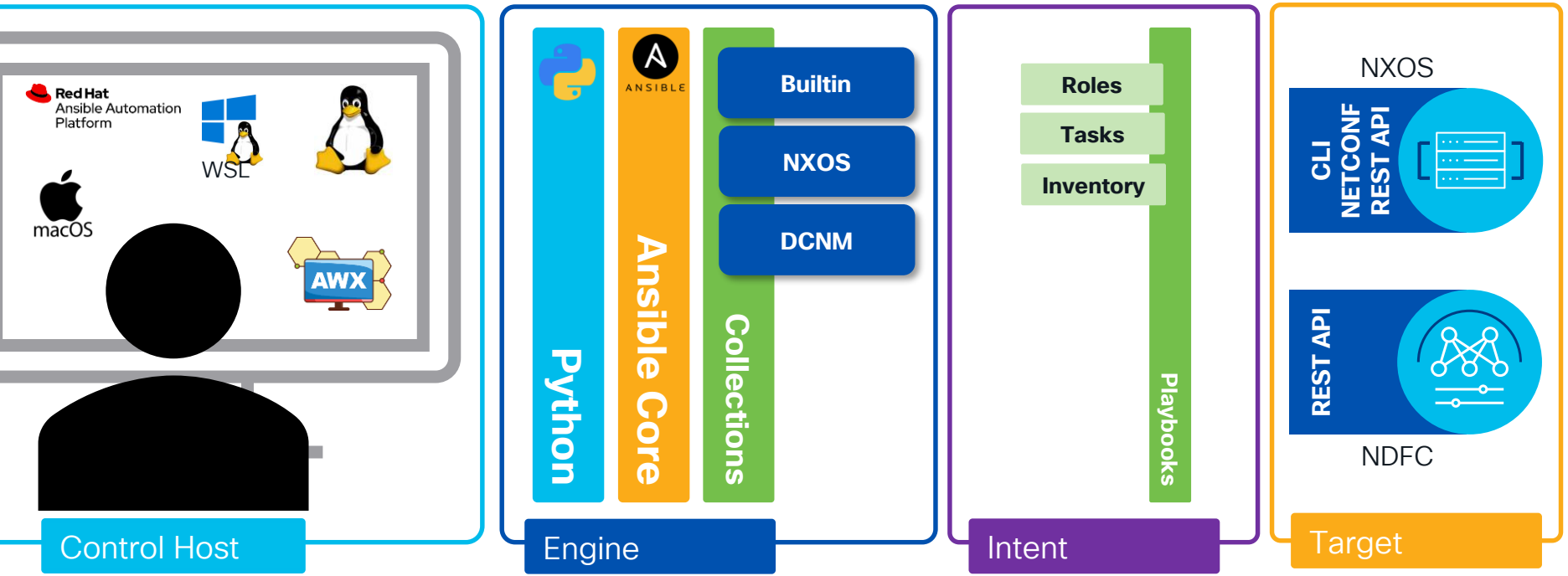
cisco Live!

What is 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?



Reference Slide

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

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
```

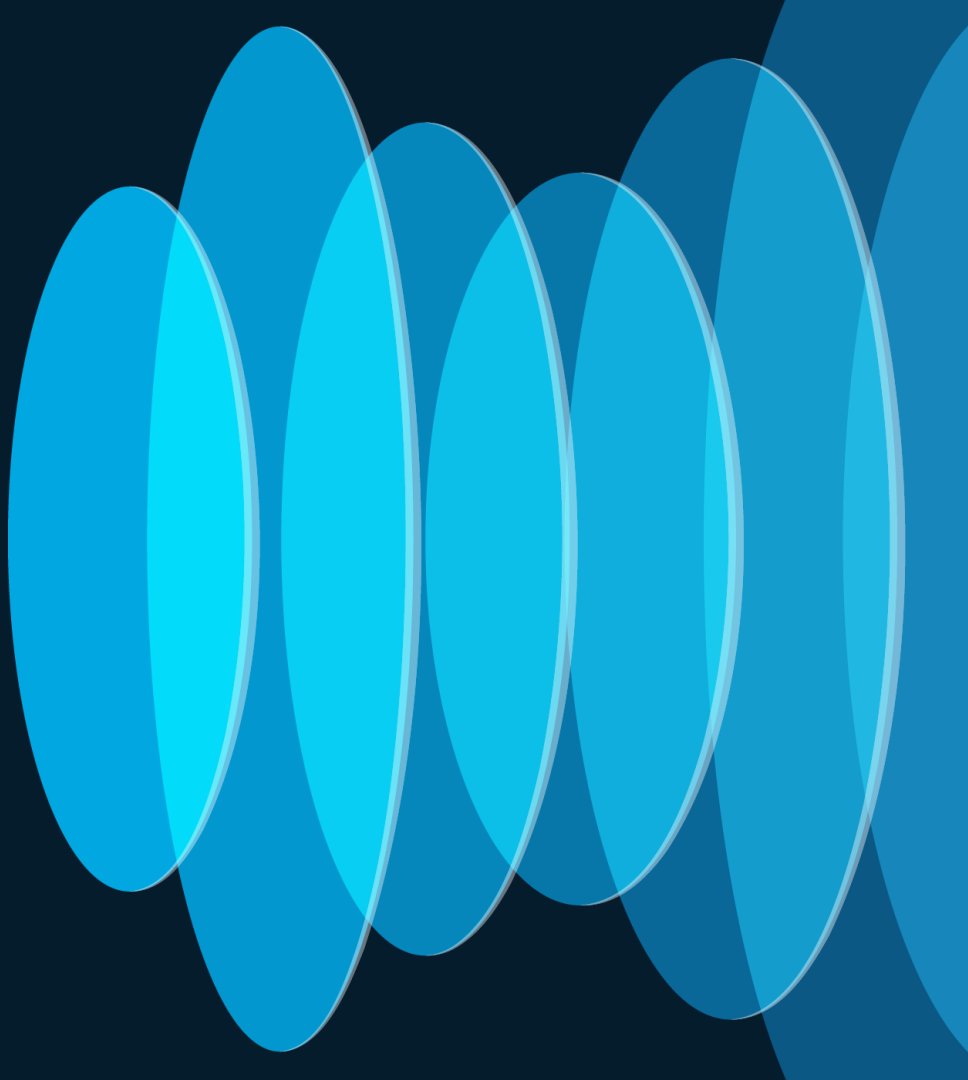
- Installs only the **core components**
- Collections must be installed by you
- Smaller footprint and more control
- Assures install of latest collection version released!

```
% 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

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

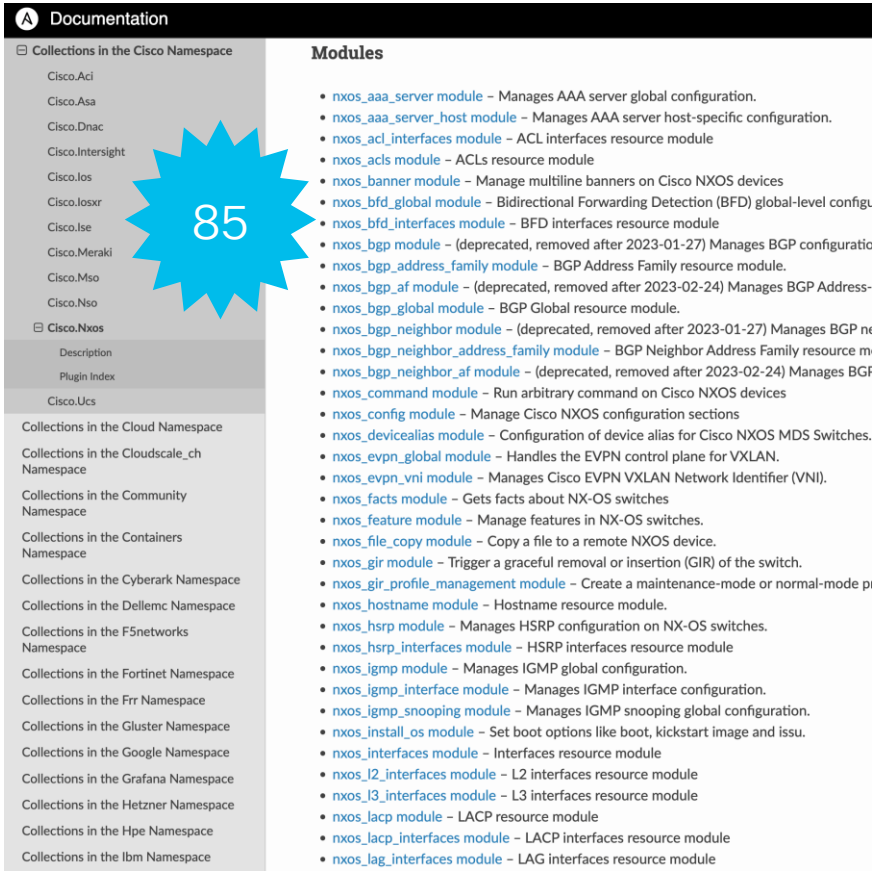
- Modules perform specific task like set facts (variables), import roles, tasks, vars, and more
- Includes many filters for working with data sets
- Actively maintained by RedHat

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

Ansible Cisco.Nxos

Collection Modules

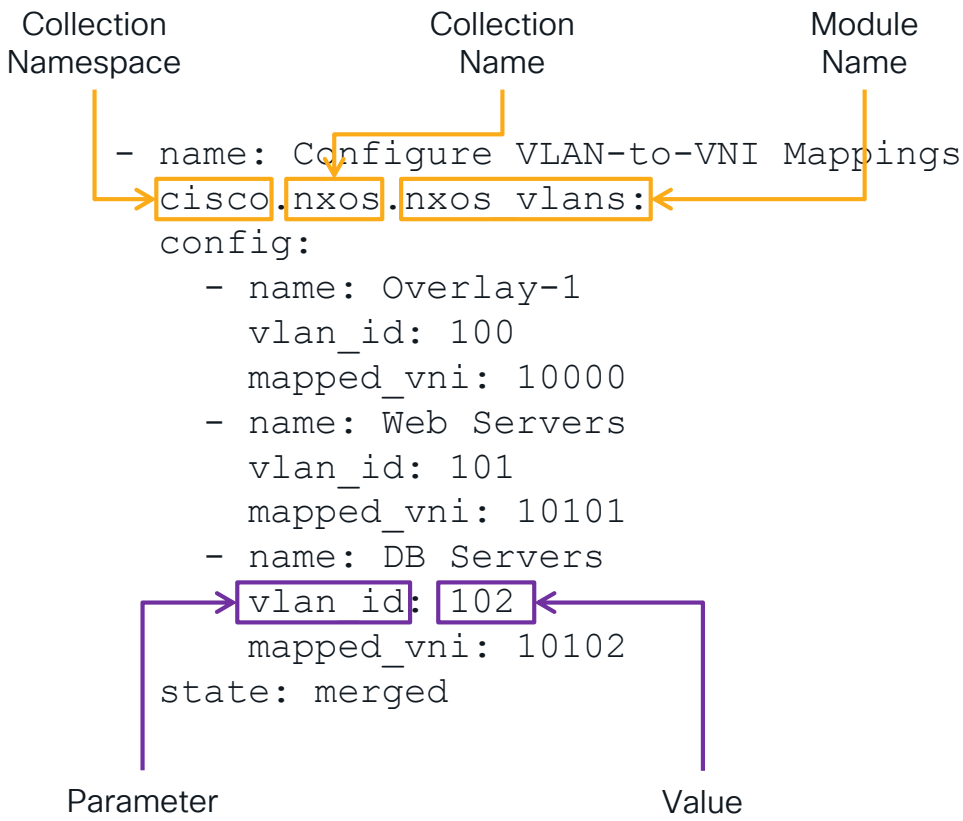
- Modules perform specific task like configure vlans, interfaces, OSPF, BGP, and more
- Documentation provides usage details, required variables, default variables, etc
- Actively maintained by RedHat with Cisco support



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

ATOM

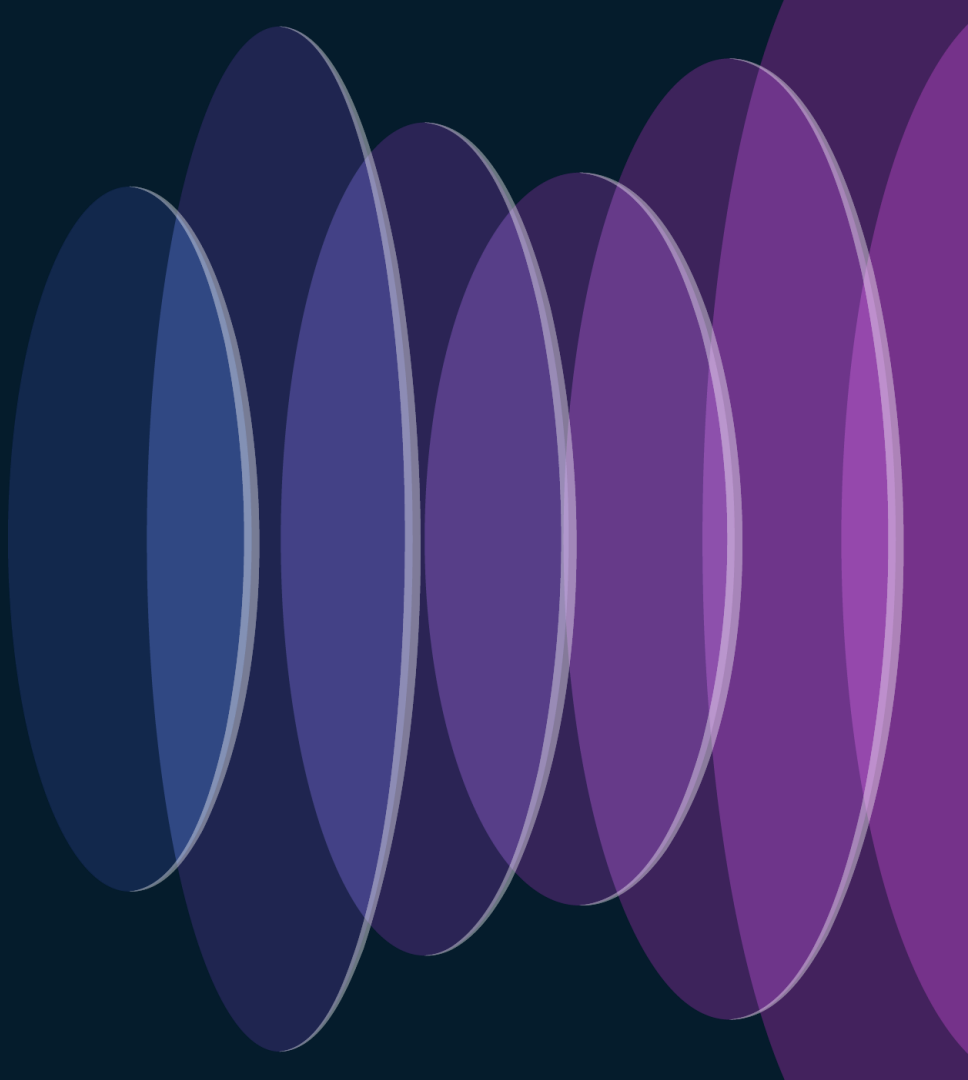


PyCharm

Eclipse


Notepad++

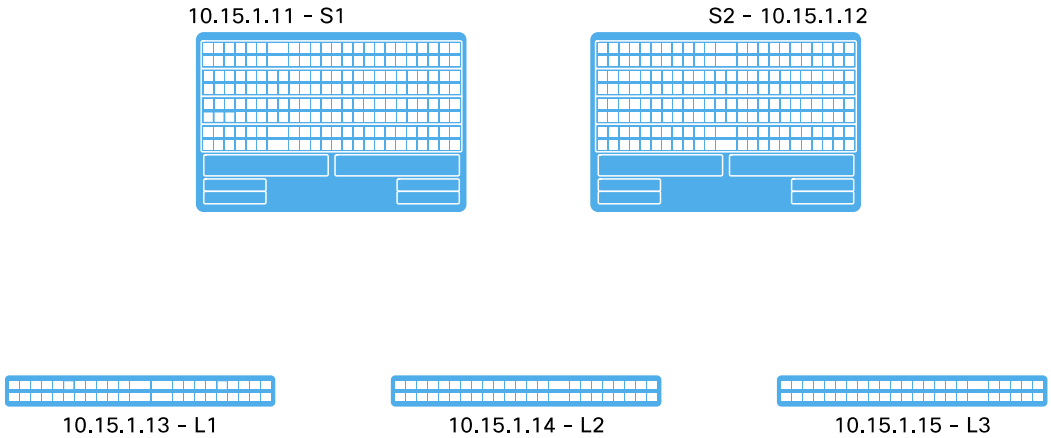
Ansible Concepts



VXLAN EVPN Fabric

IaC – Nexus as Code

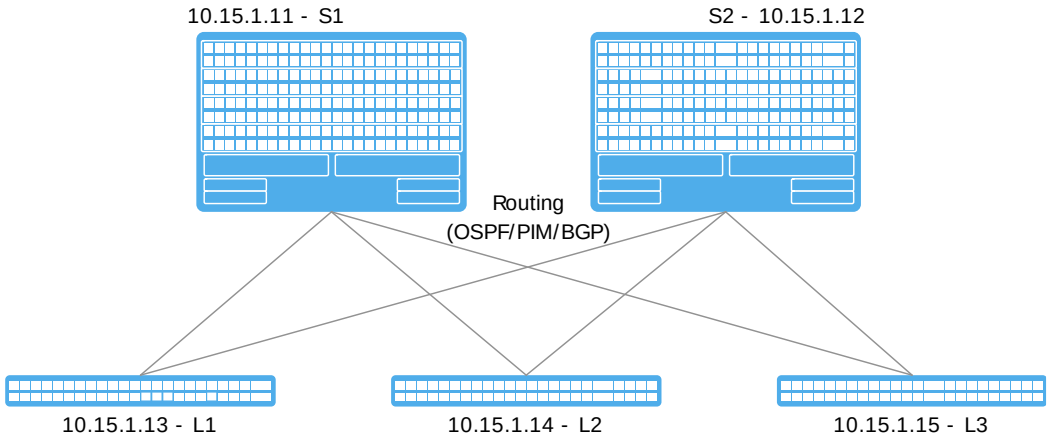
- Configure Hostnames, Features, etc (Common configuration)



VXLAN EVPN Fabric

IaC – Nexus as Code

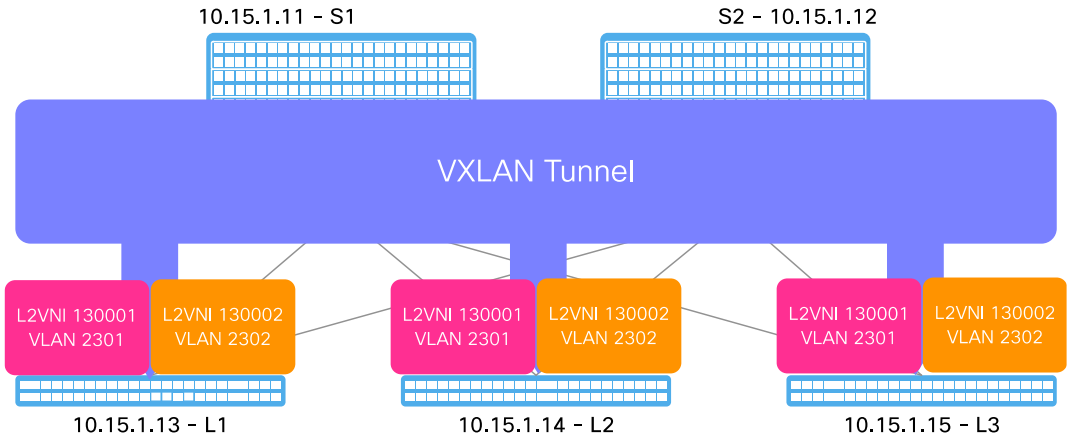
- Configure **Underlay** (Interfaces, Routing protocols, etc)



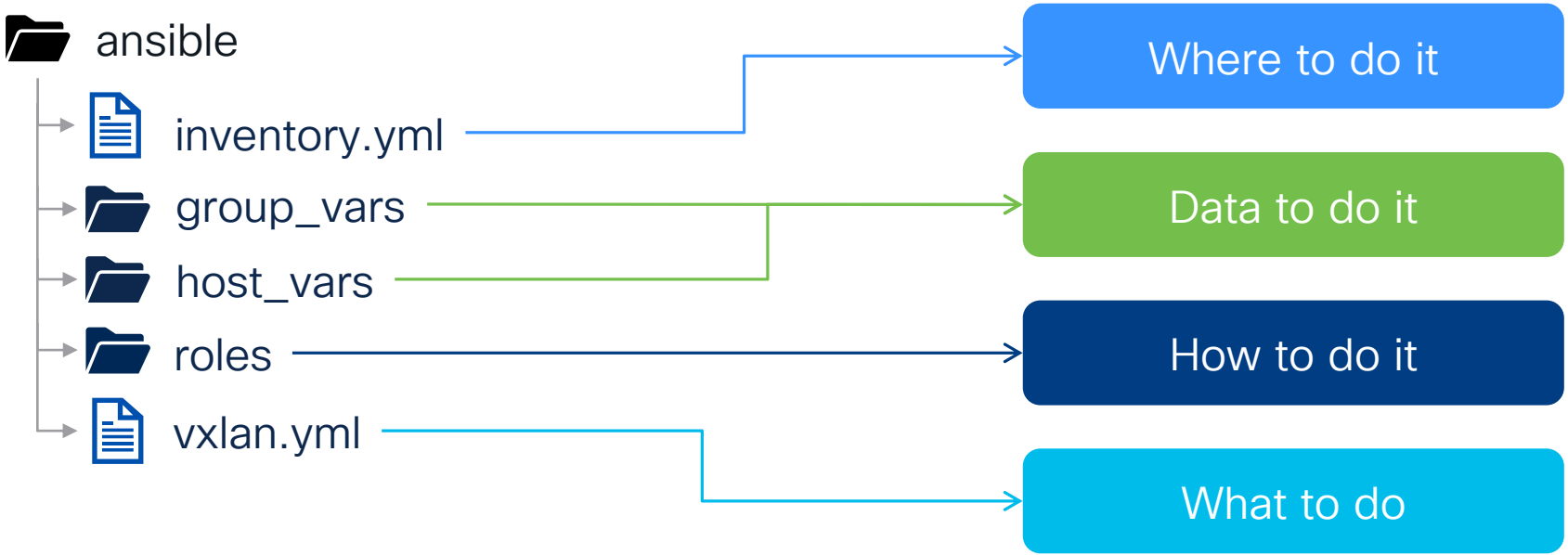
VXLAN EVPN Fabric

IaC – Nexus as Code

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

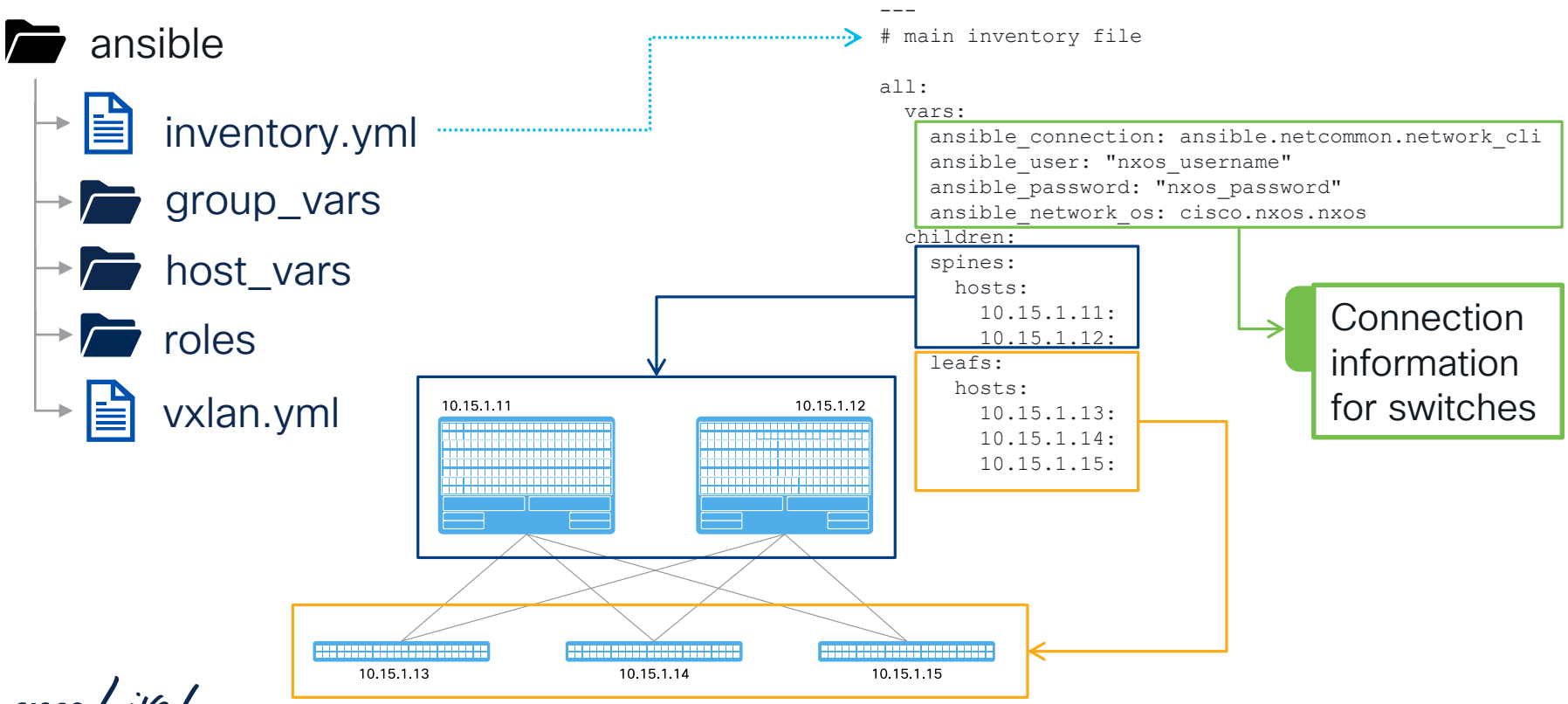


Ansible Directory Structure



Ansible Inventory

Where to do it

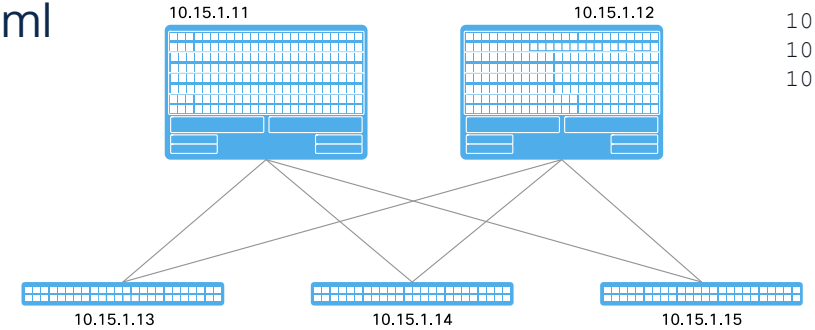


Ansible Inventory

Where to do it


```
---
# 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:
    leafs:
      hosts:
        10.15.1.13:
        10.15.1.14:
        10.15.1.15:
```

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



Ansible Playbook

What to do

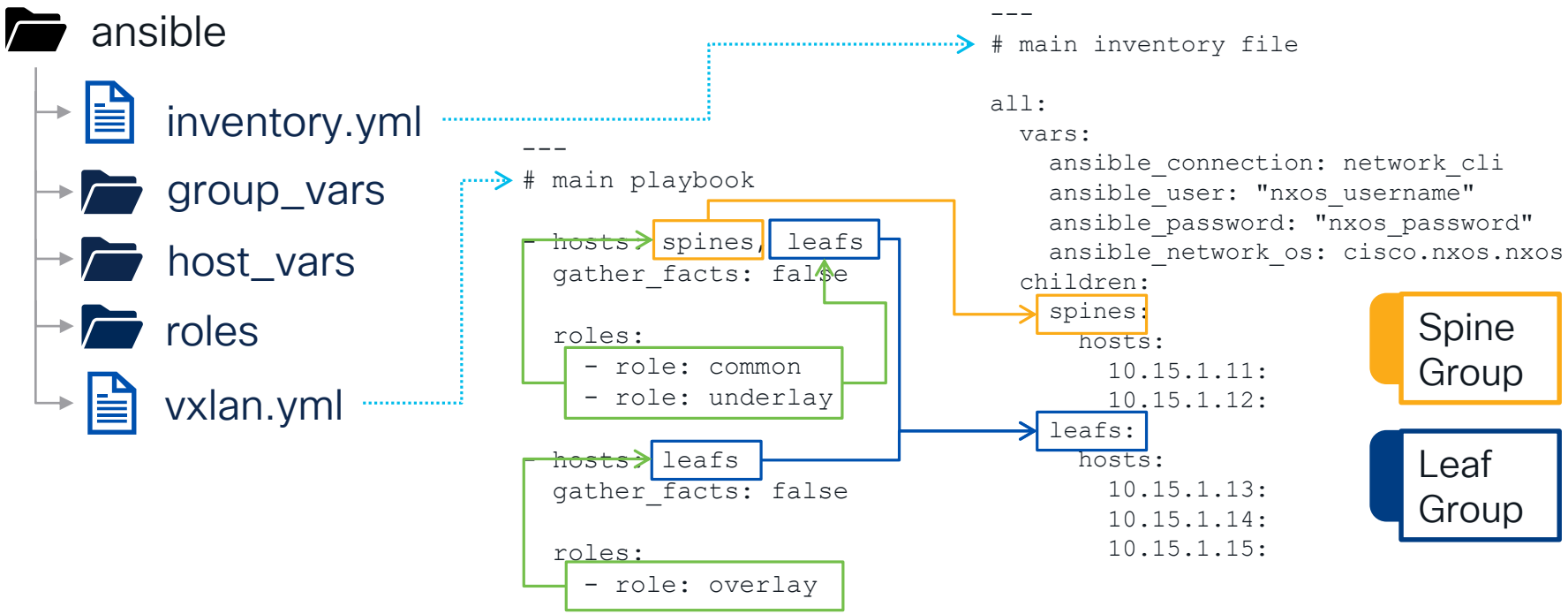

```
---
# main playbook

- 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

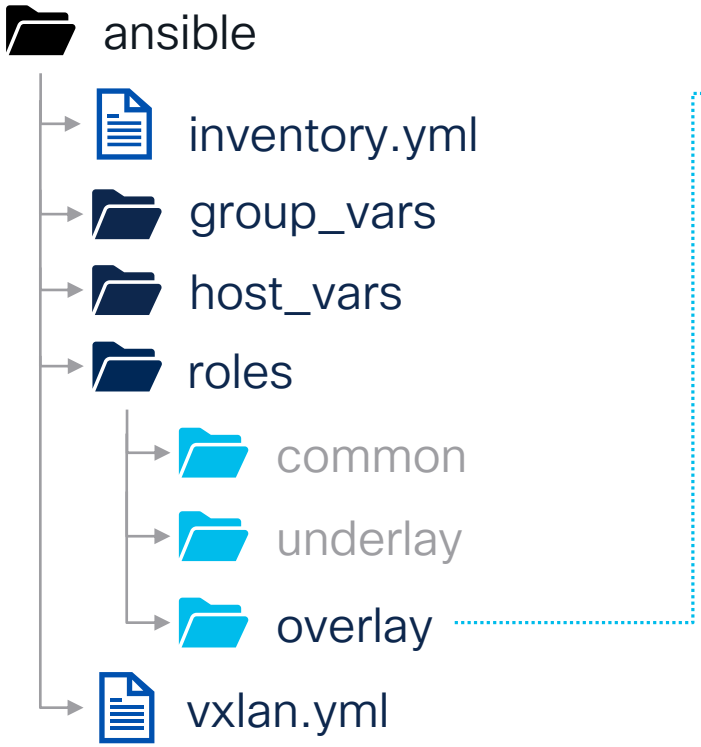
Value of ansible_connection	Protocol	Requires	Persistent
ansible.netcommon.network_cli	CLI over SSH	ansible_network_os	Yes
ansible.netcommon.httpapi	API over HTTP/HTTPS	ansible_network_os	Yes

Ansible Playbook Relationships



Ansible Roles

How to do it

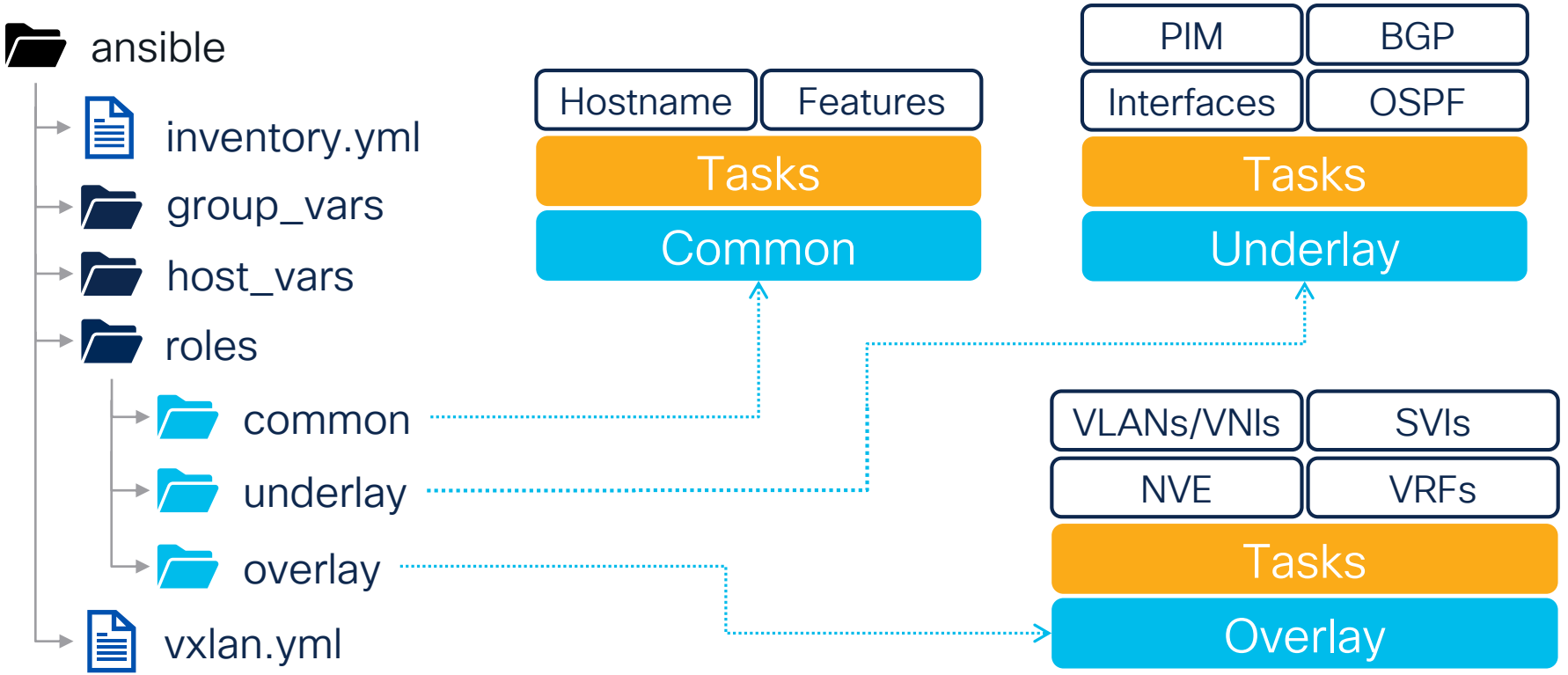



```
❯ ansible-galaxy init overlay
- Role overlay was created successfully
❯ tree overlay
overlay
├── README.md
├── tasks
│   └── main.yml
├── templates
└── vars
    └── main.yml
```

This is how we do it

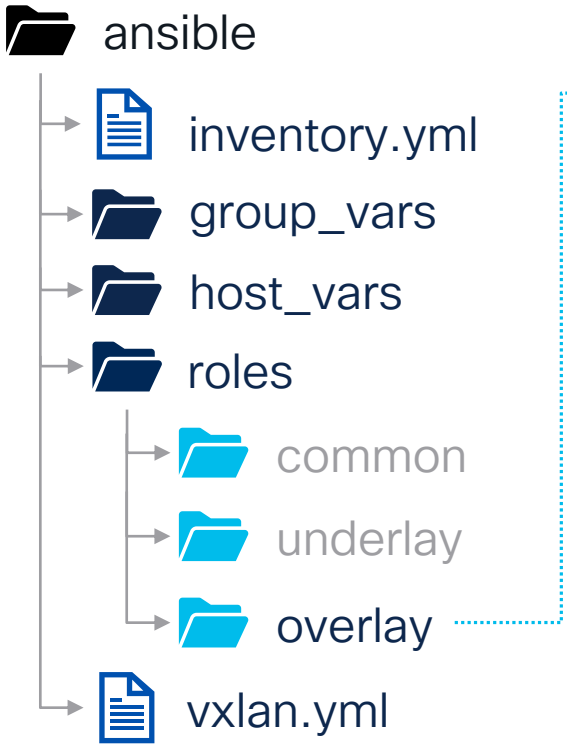
Ansible Roles – Tasks

How to do it



Ansible Roles – Tasks

How to do it



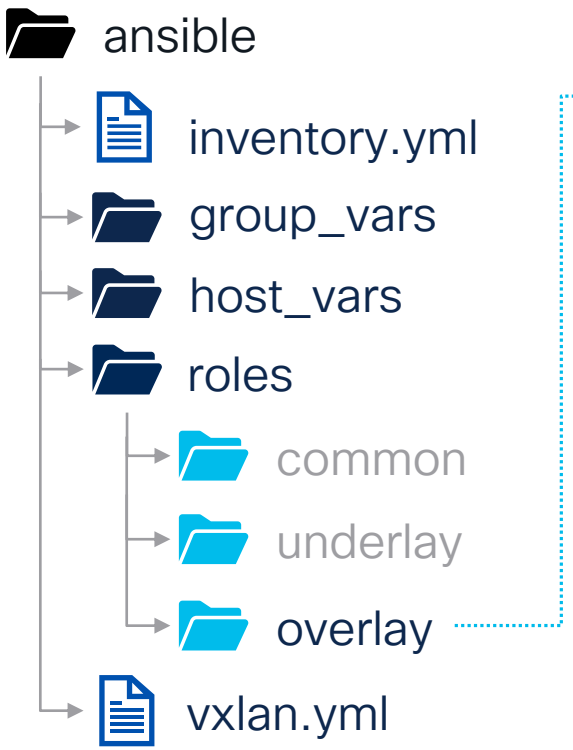
roles/overlay/tasks/main.yml

```
---
# 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

How to do it



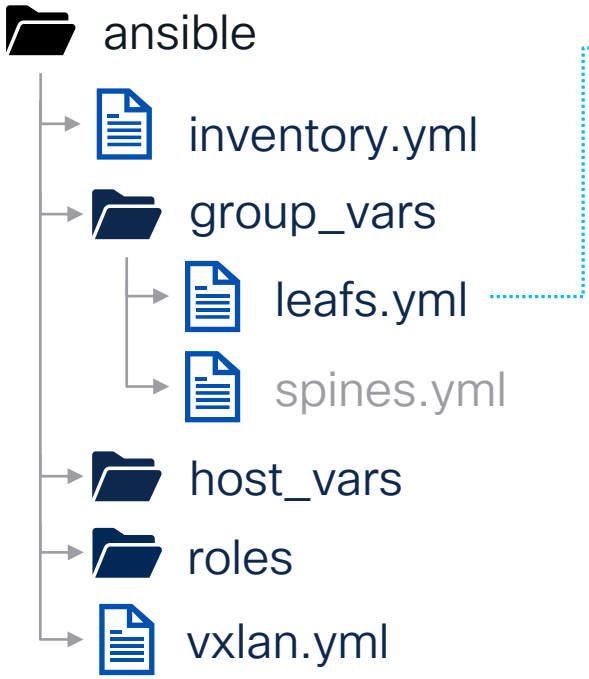
roles/overlay/tasks/main.yml

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

- name: Configure VLAN-to-VNI Mappings
  cisco.nxos.nxos_vlans:
    config:
      - name: Web Servers
        vlan_id: 1
        mapped_vni: 10101
      - name: DB Servers
        vlan_id: 102
        mapped_vni: 10102
      - name: vMotion
        vlan_id: 103
        mapped_vni: 10103
    state: merged
<snip>
```

Do not do Ansible task parameters this way!

Ansible Group Vars

Data to do it

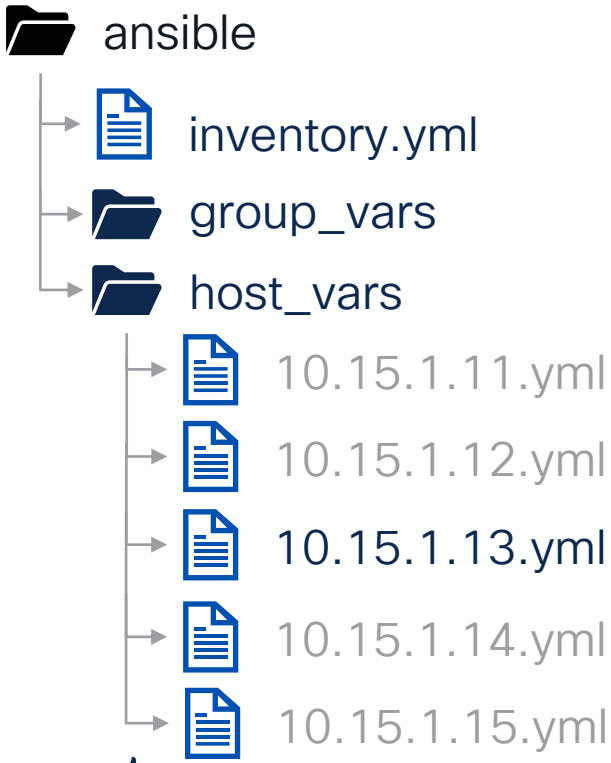



```
---
# var file for leafs group

features:
  - ospf
  - pim
  - bgp
  - nv overlay
  - vn-segment-vlan-based
  - interface-vlan

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
```

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

Ansible Host Vars

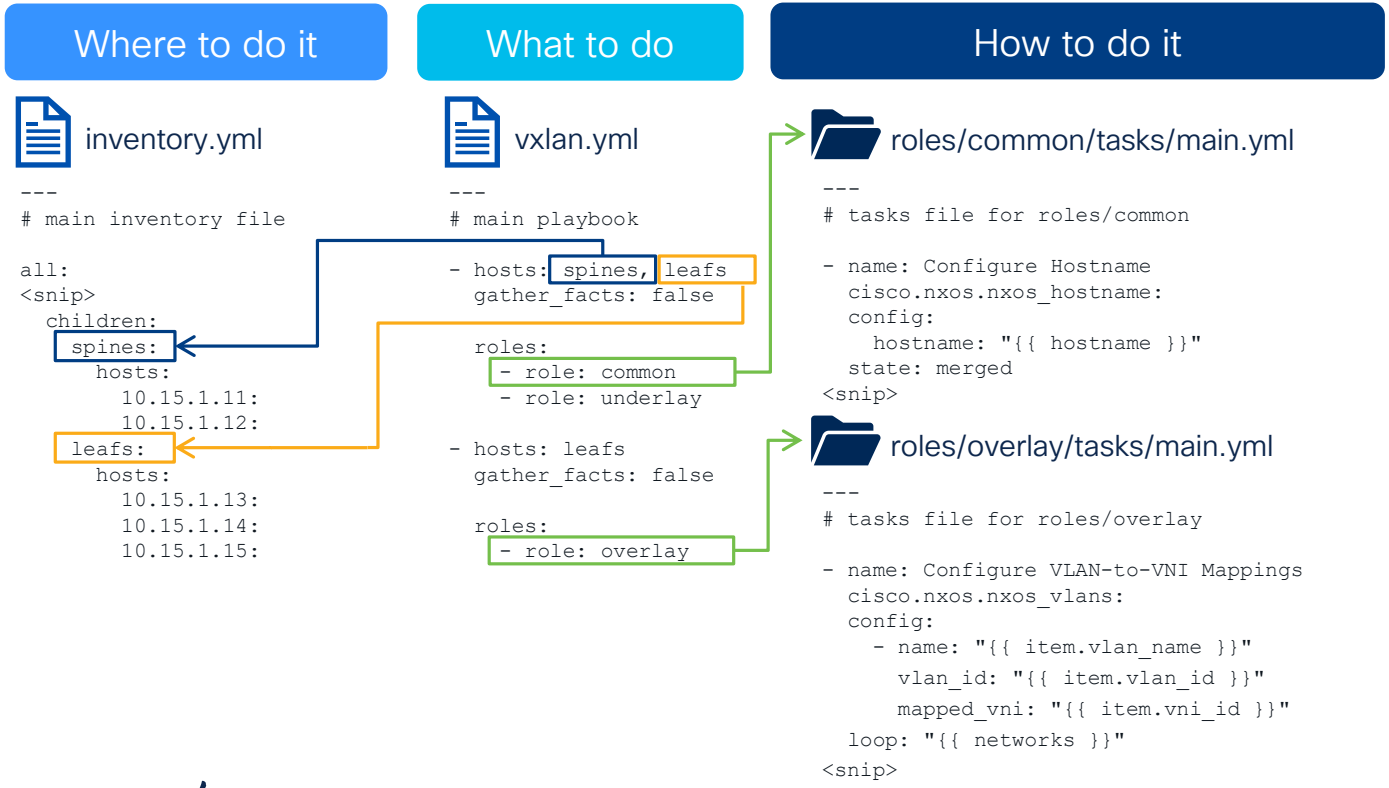
Data to do it




```
---
# 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



Putting It All Together

Where to do it | **What to do** | **How 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:
```

vxlan.yml

```
---
# main playbook

- hosts: spines, leafs
  gather_facts: false

  roles:
    - role: common
    - role: underlay

- hosts: leafs
  gather_facts: false

  roles:
    - role: overlay
```

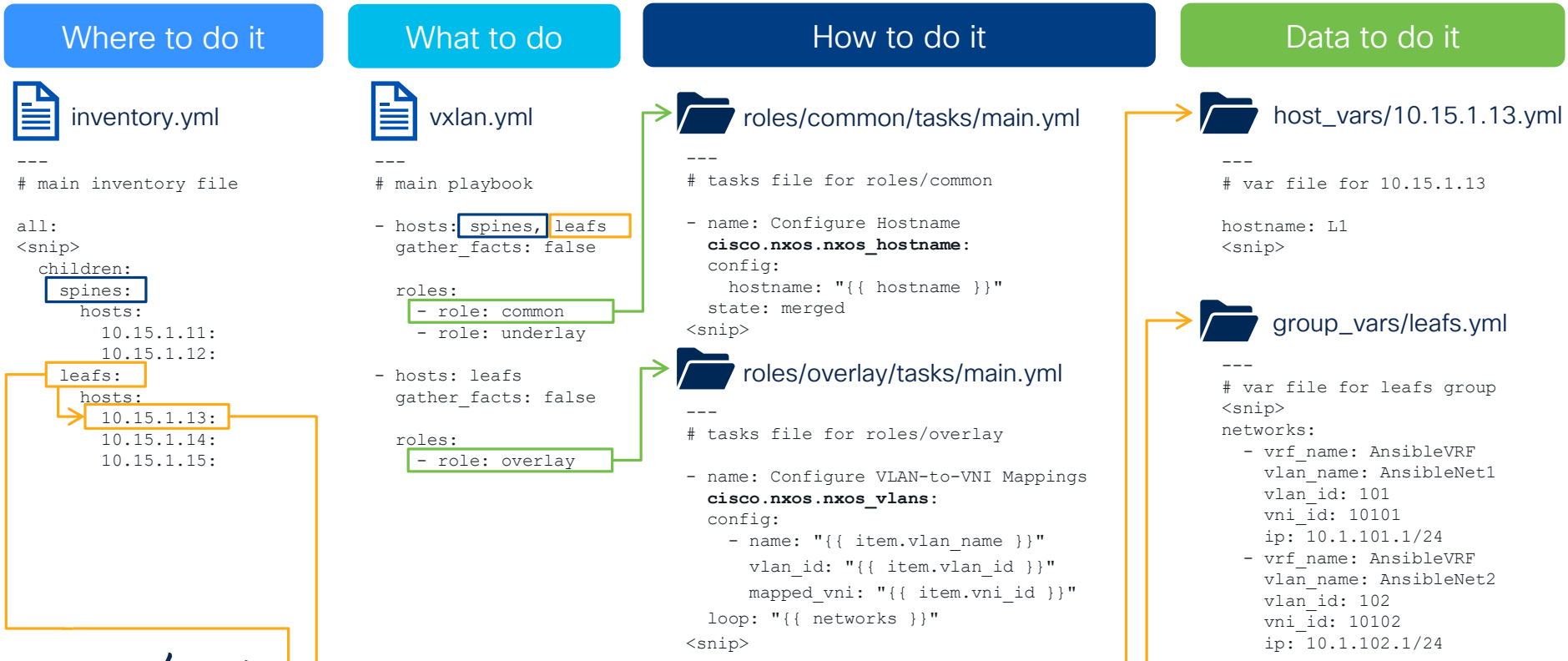
roles/common/tasks/main.yml

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

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

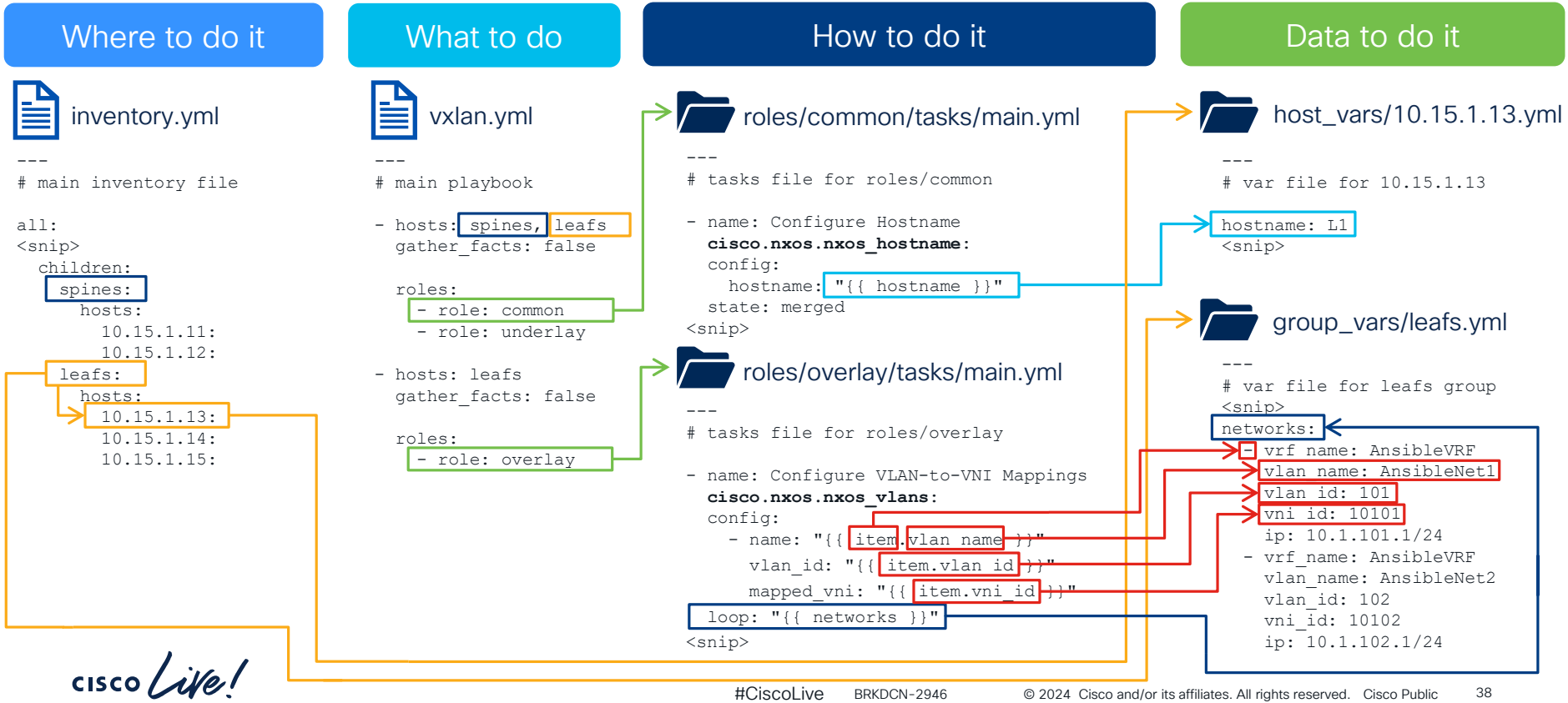
roles/overlay/tasks/main.yml

```
---
# 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>
```

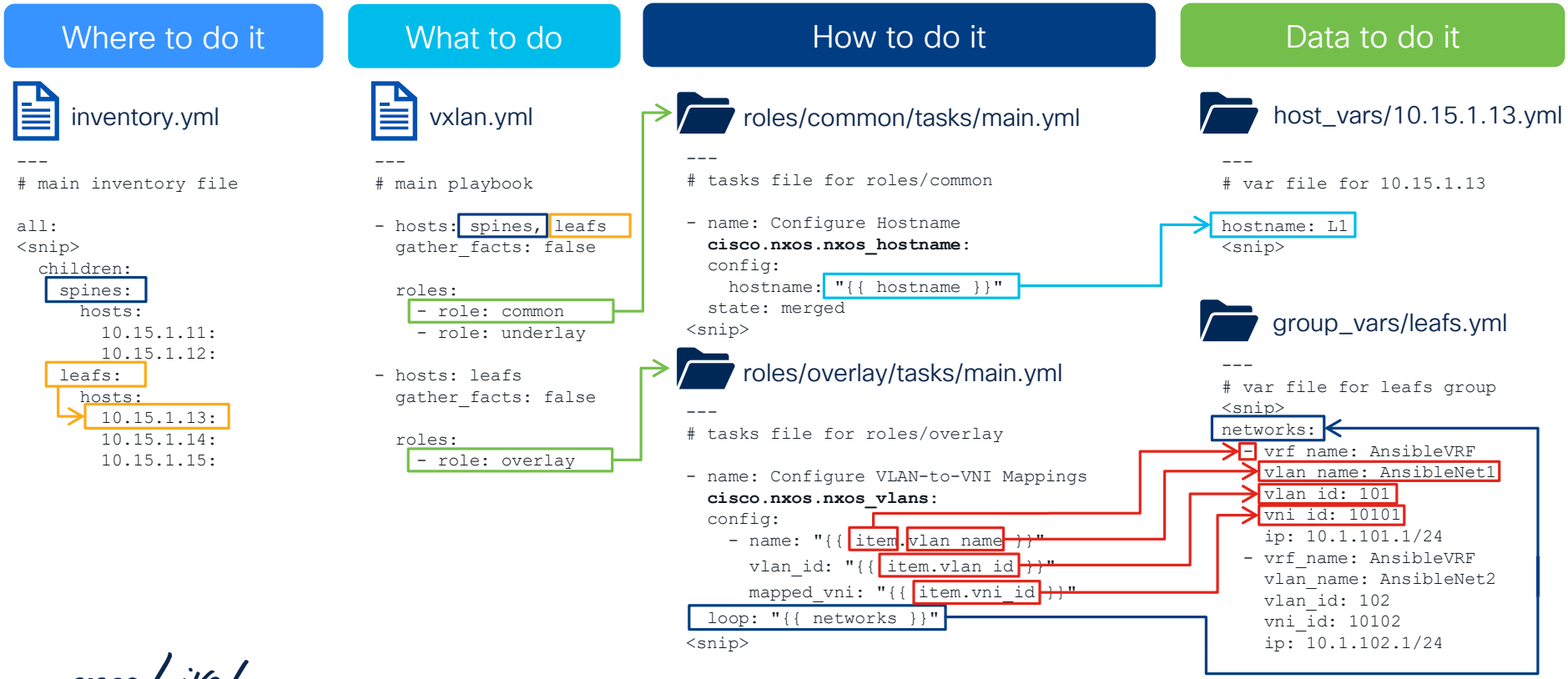
Putting It All Together



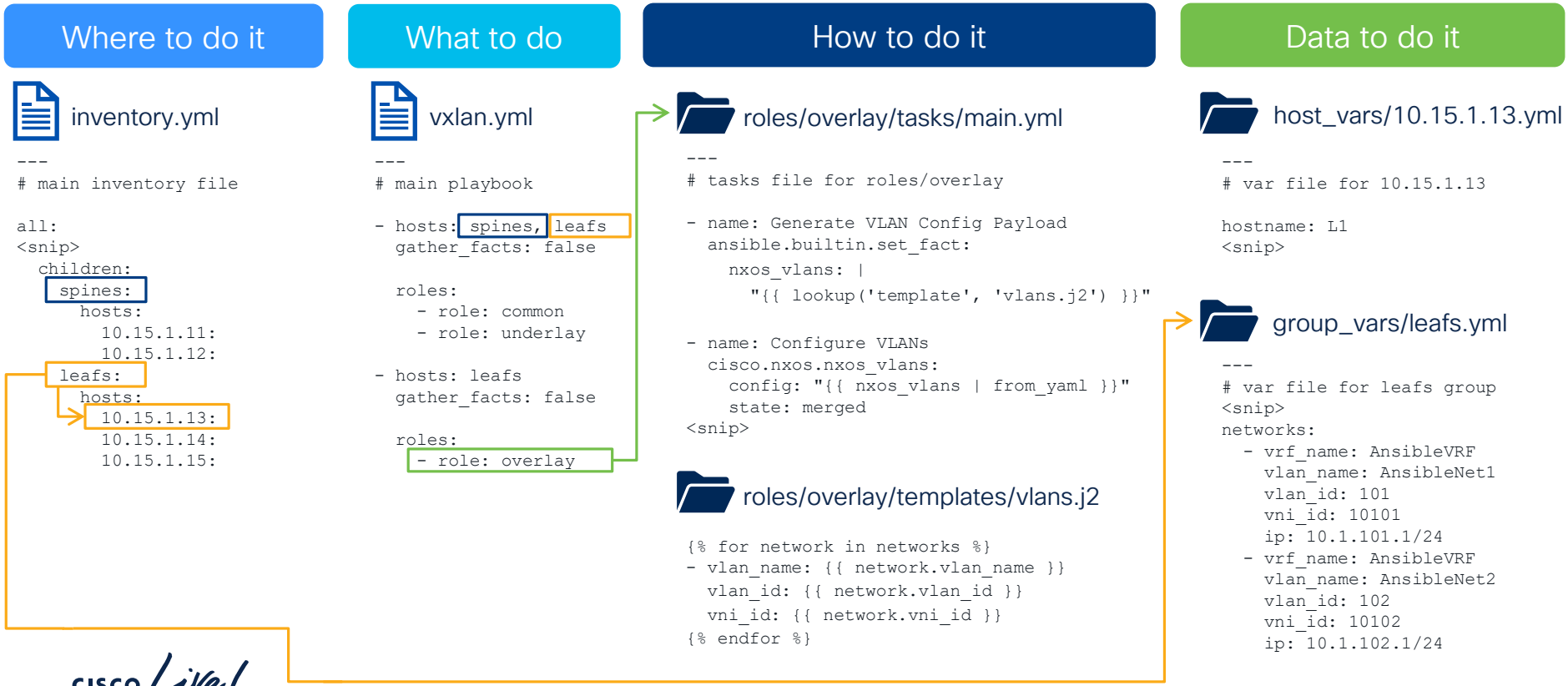
Putting It All Together



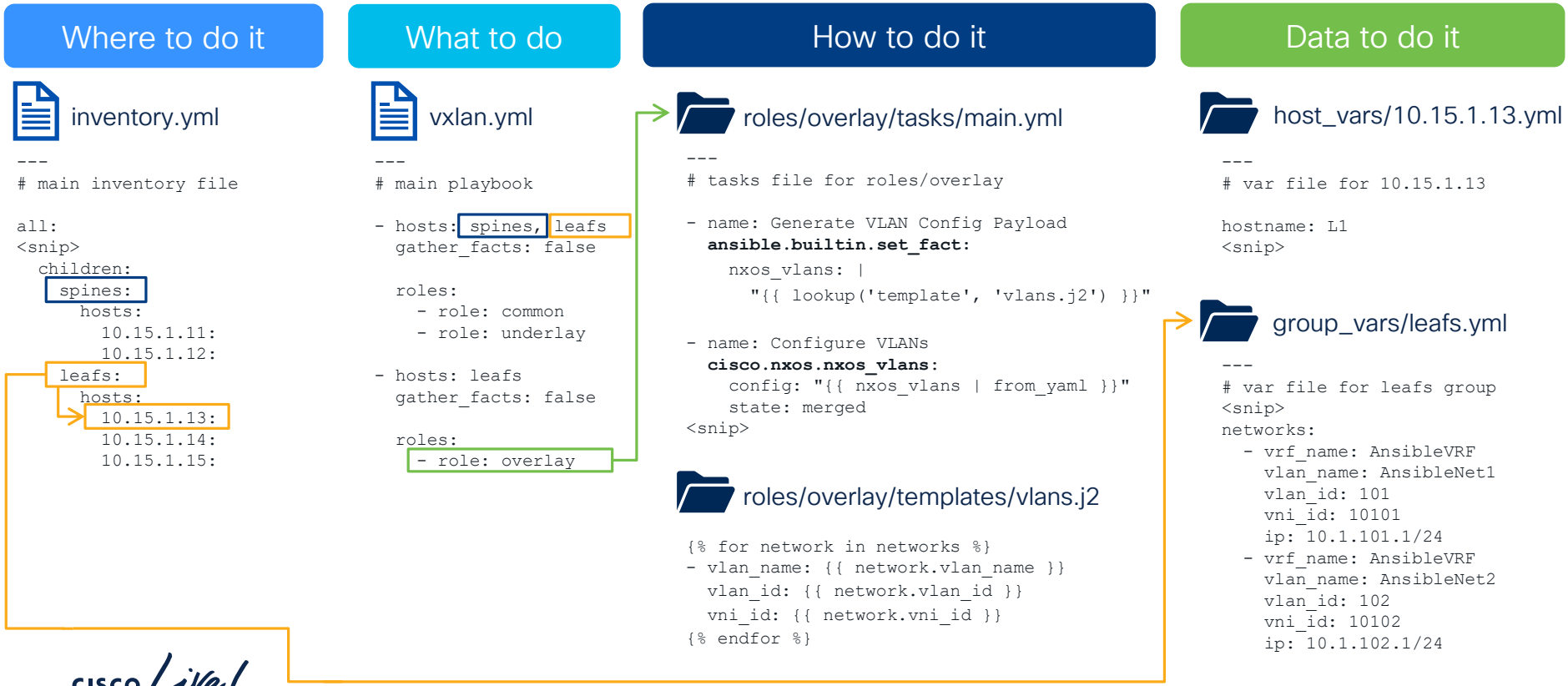
Putting It All Together



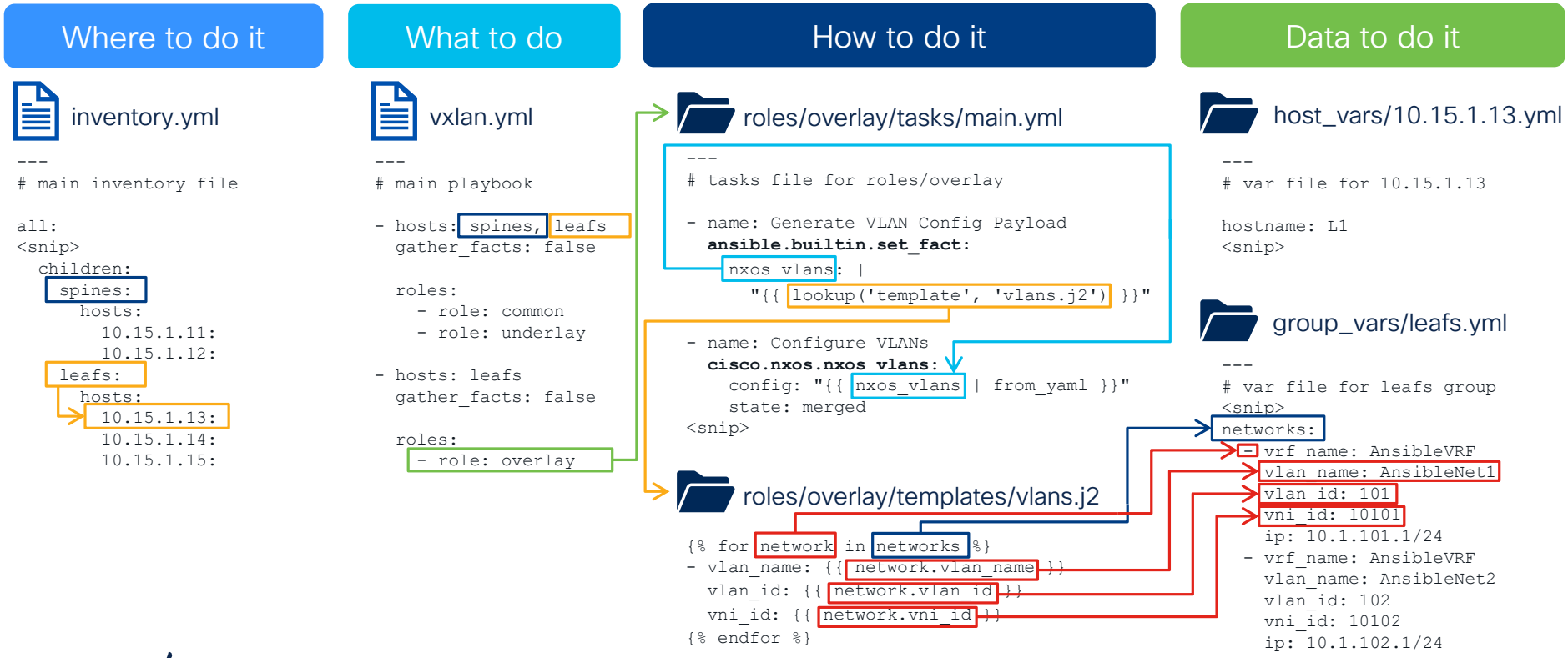
Putting It All Together



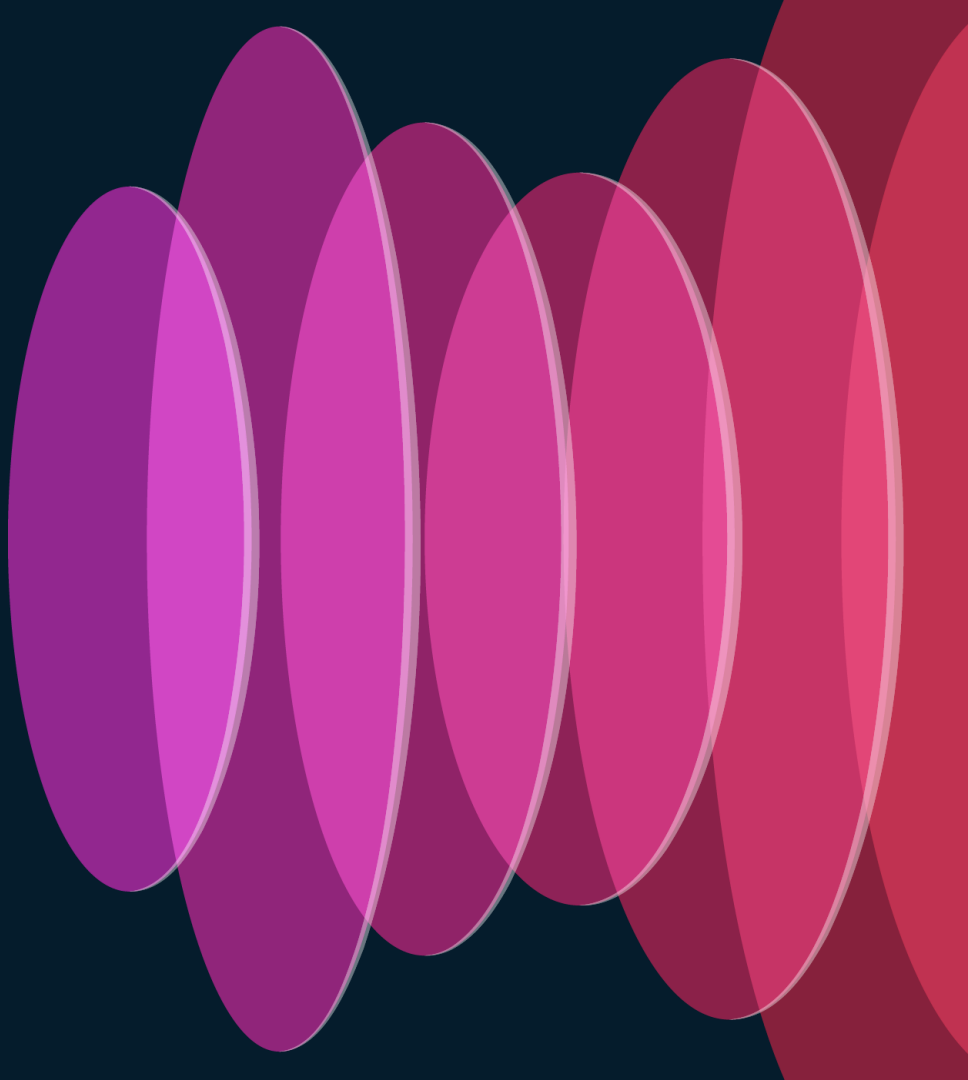
Putting It All Together



Putting It All Together



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

Variables with Jinja2 Syntax

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

```
# tasks file for roles/overlay

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

```
# vars defined somewhere in Ansible

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

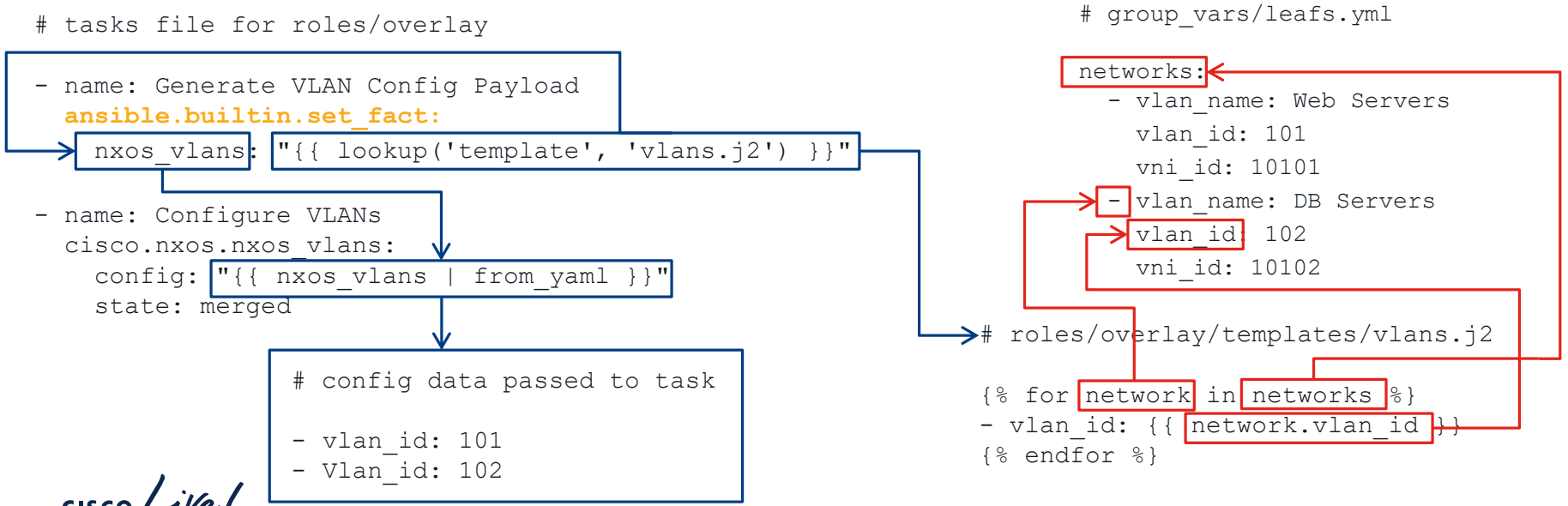
- Use Ansible loop to iterate data for a tasks

```
# 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 }}"
```

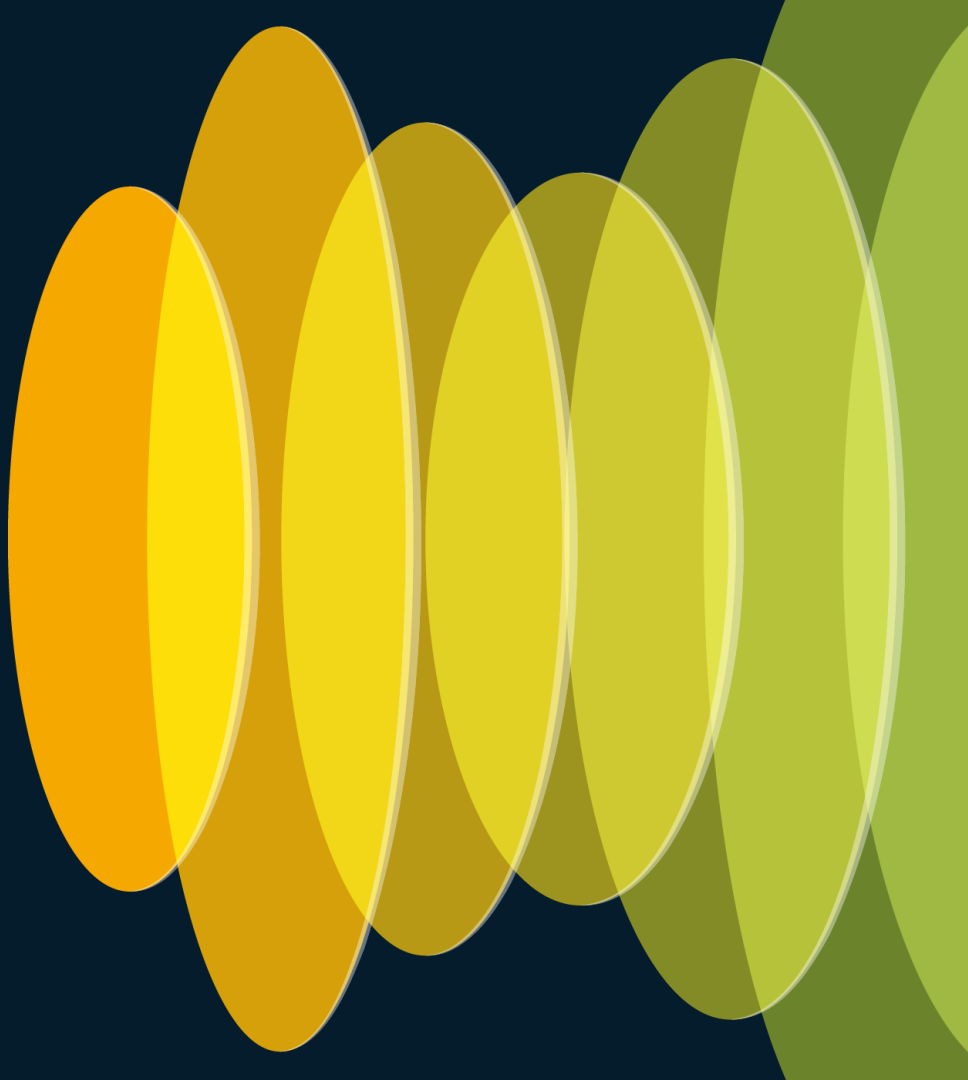
```
# vars defined somewhere in Ansible

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



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

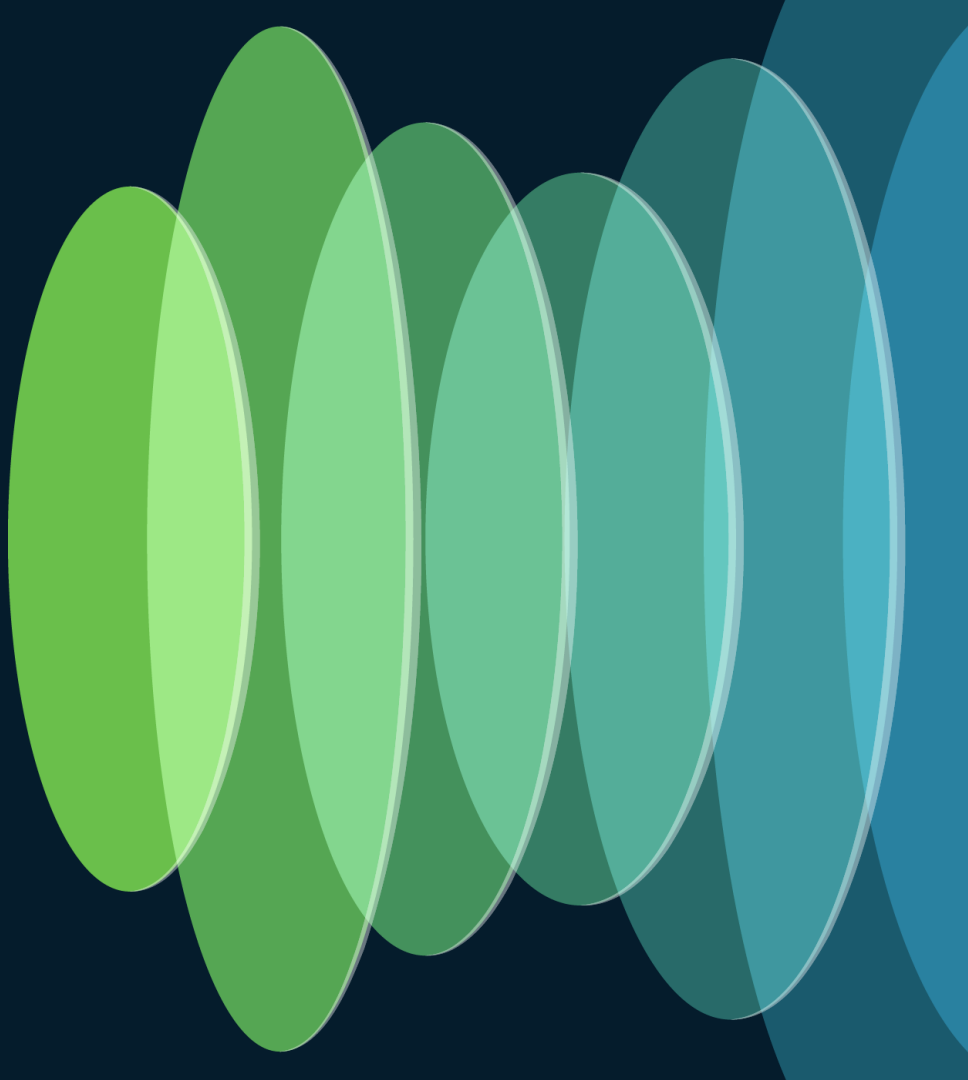
How to send commands to NXOS when a module is missing

- 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
```

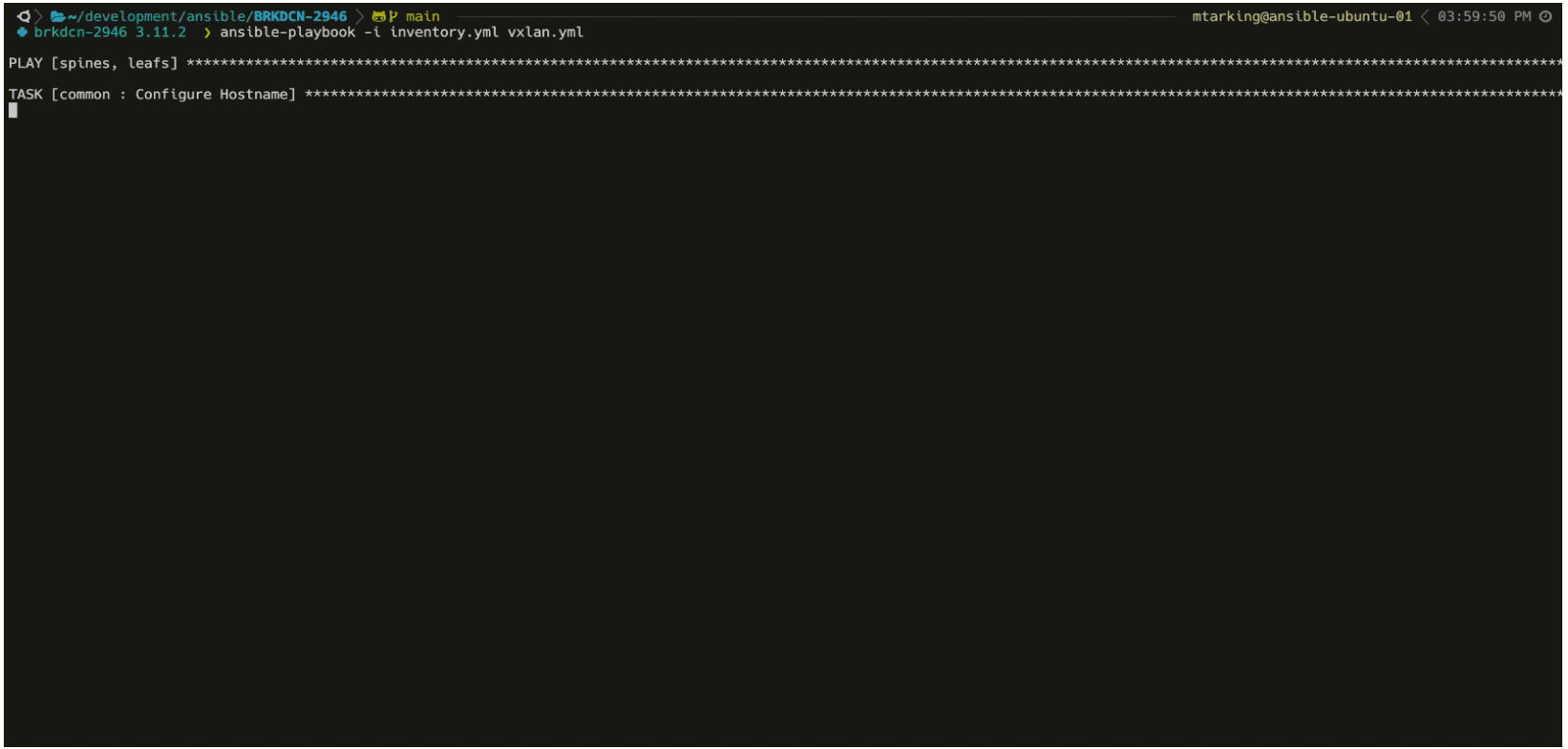
```
- name: Misc Commands
  cisco.nxos.nxos_command:
    commands: copy ftp://nxos.bin bootflash:
    prompt:
      - "Username:"
      - "Password:"
    answer:
      - <username>
      - <password>
```

An Example

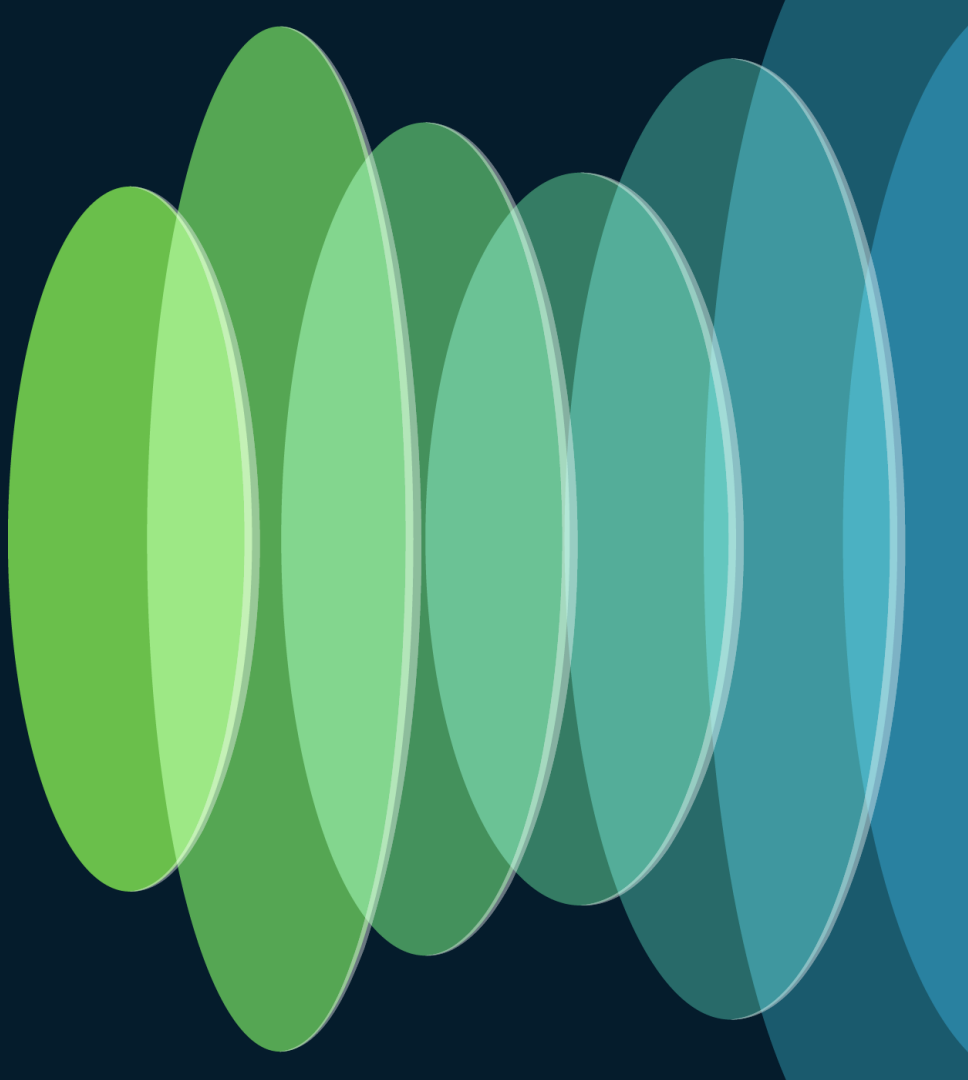


Executing Ansible Playbooks

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



Infrastructure as Code with NDFC and Ansible



cisco Live!

Cisco Nexus Dashboard Fabric Controller (NDFC)

Formerly Called (DCNM)

Automation

Accelerate provisioning from days to minutes

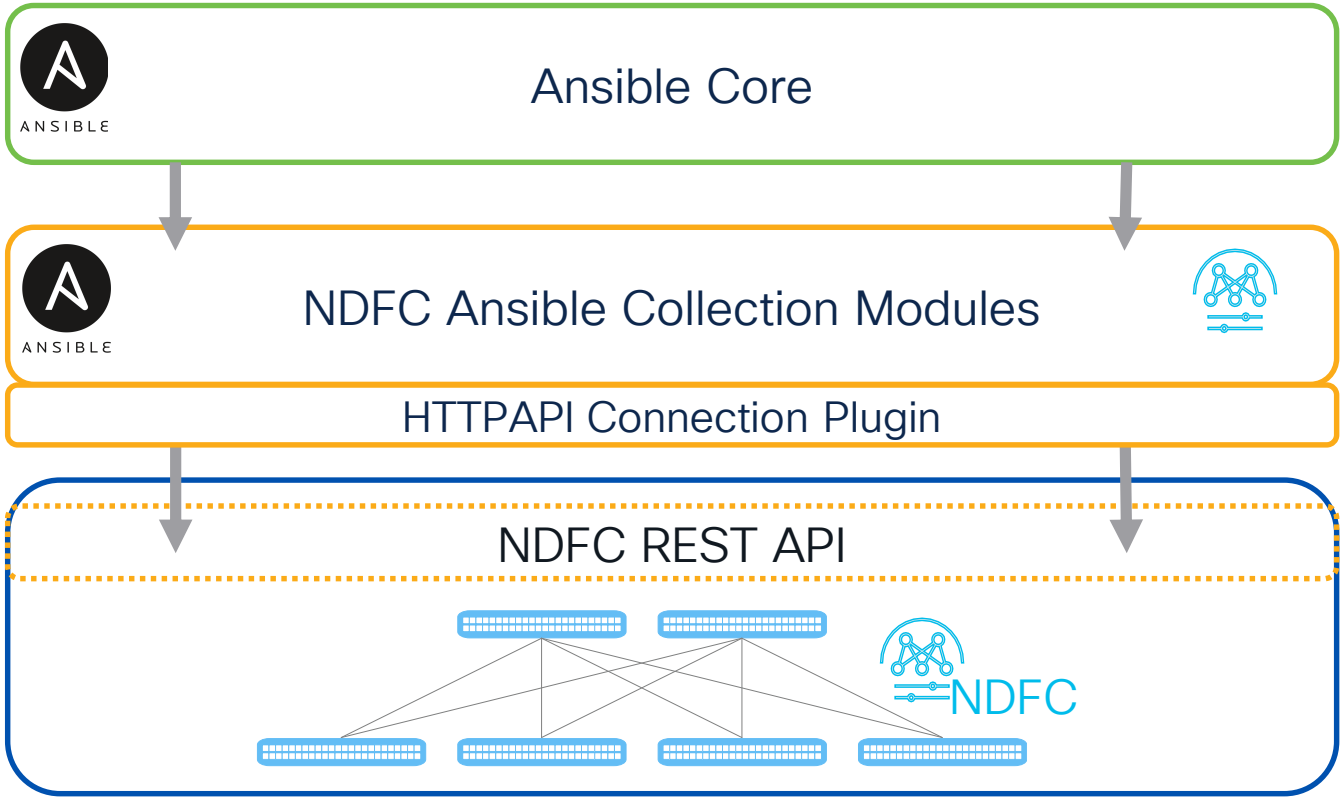
Easy to understand approach to auto-bootstrapping of entire fabric

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

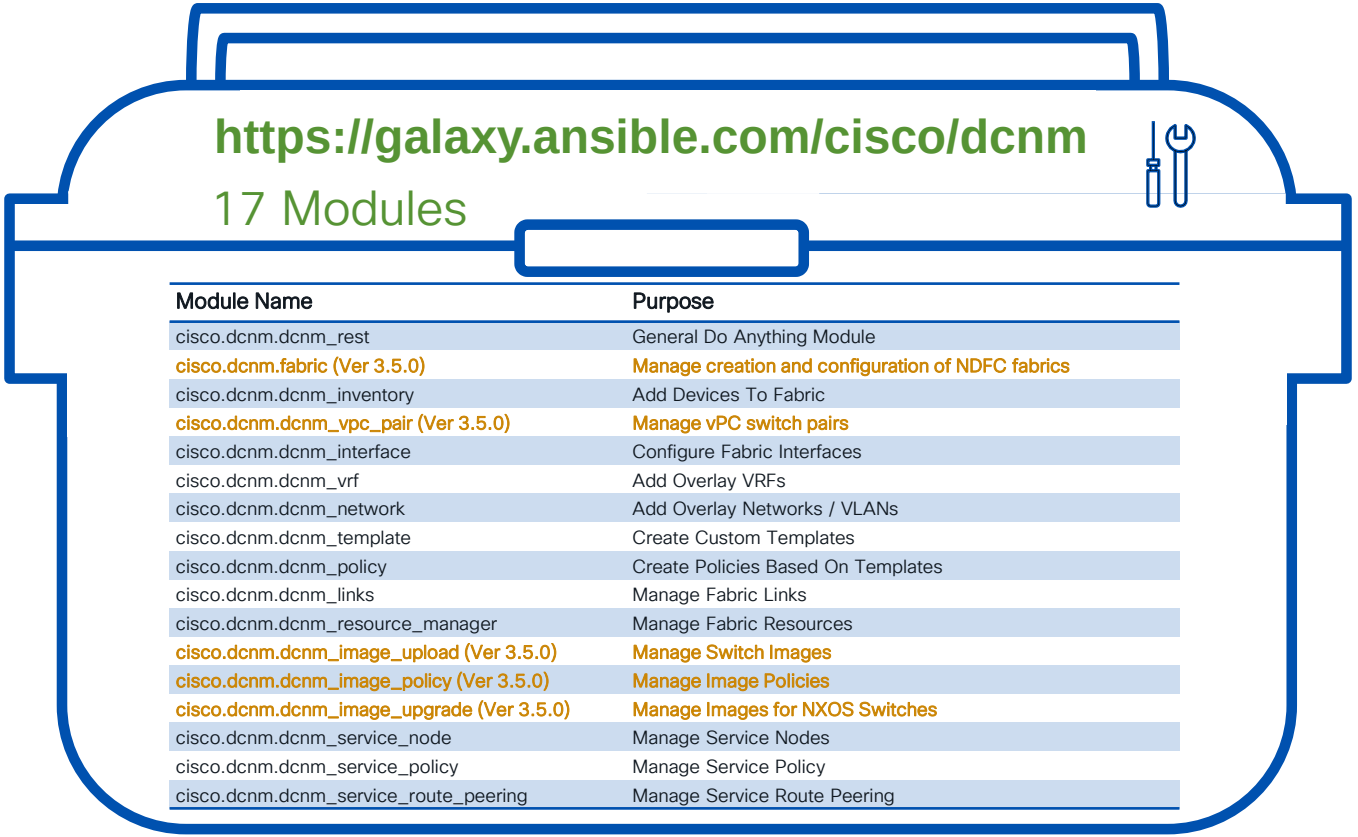
DevOps friendly

Enhanced Programmability

NDFC DevOps “Friendliness”



NDFC Collection Modules – Toolbox

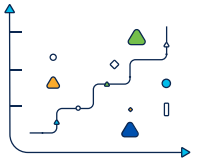


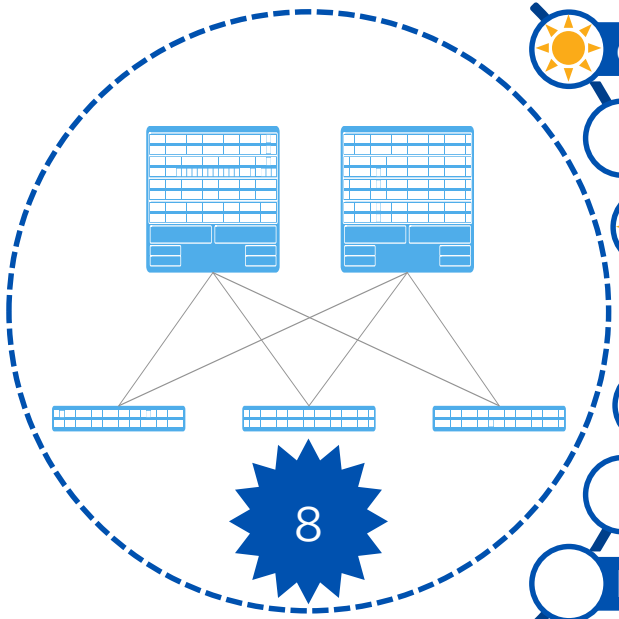
https://galaxy.ansible.com/cisco/dcnm

17 Modules

Module Name	Purpose
cisco.dcnm.dcnm_rest	General Do Anything Module
cisco.dcnm.fabric (Ver 3.5.0)	Manage creation and configuration of NDFC fabrics
cisco.dcnm.dcnm_inventory	Add Devices To Fabric
cisco.dcnm.dcnm_vpc_pair (Ver 3.5.0)	Manage vPC switch pairs
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
cisco.dcnm.dcnm_image_upload (Ver 3.5.0)	Manage Switch Images
cisco.dcnm.dcnm_image_policy (Ver 3.5.0)	Manage Image Policies
cisco.dcnm.dcnm_image_upgrade (Ver 3.5.0)	Manage Images for NXOS Switches
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





- cisco.dcnm.dcnm_fabric
- cisco.dcnm_inventory
- cisco.dcnm.dcnm_vpc_pair
- cisco.dcnm_interface
- cisco.dcnm_vrf & cisco.dcnm_network
- cisco.dcnm_rest
- httpapi plugin

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”

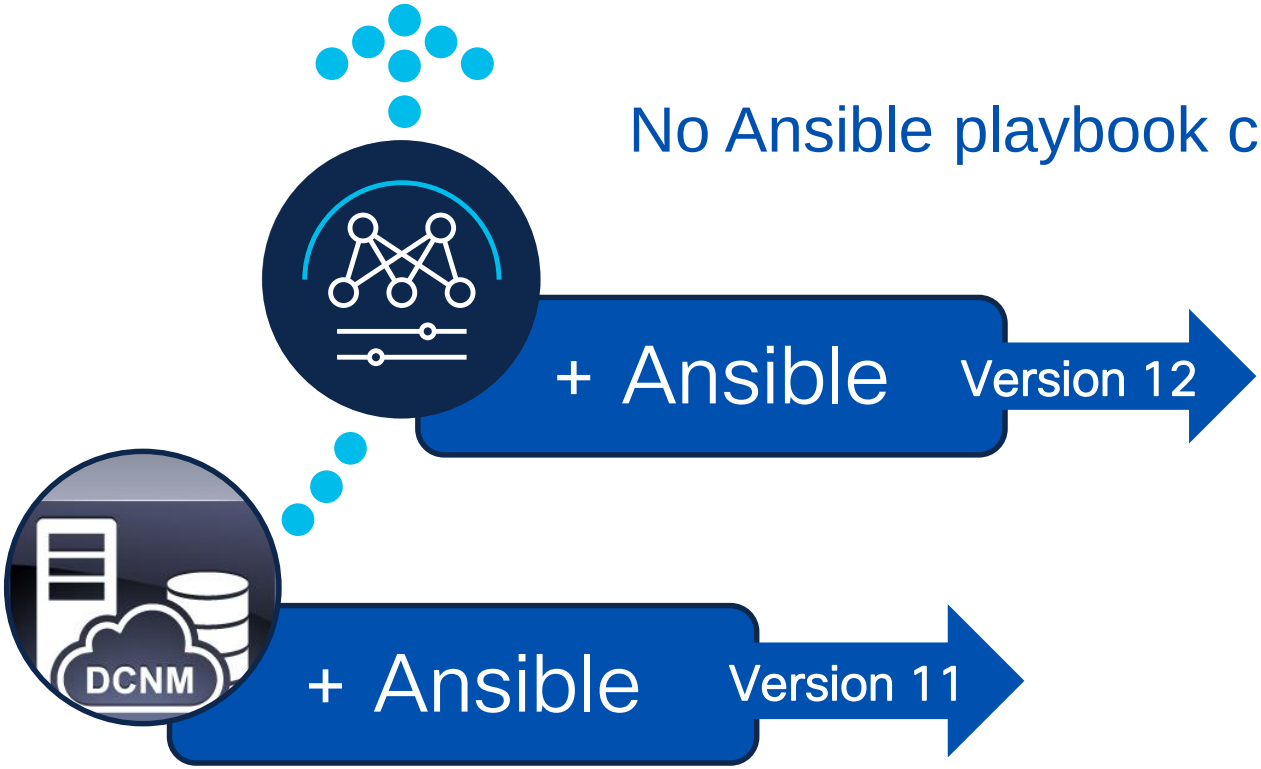
NDFC - https://galaxy.ansible.com/cisco/dcnm

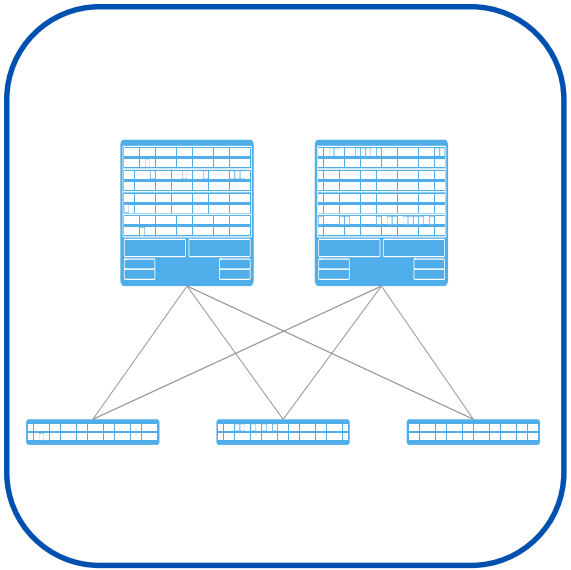
Hindsight “Specs” Model

I See Ansible Collection Names

Transparent NDFC/DCNM Controller Support

No Ansible playbook changes required!

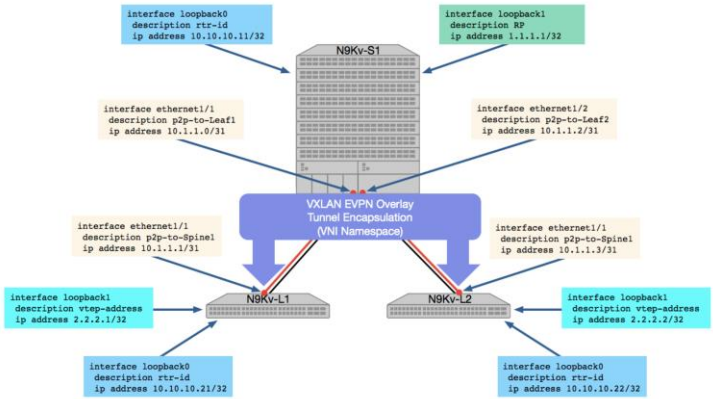




Let’s Build Something With NDFC and Ansible Together!

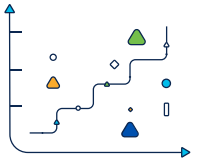
Levels of Complexity – CLI

```
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 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 – 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

ANSIBLE

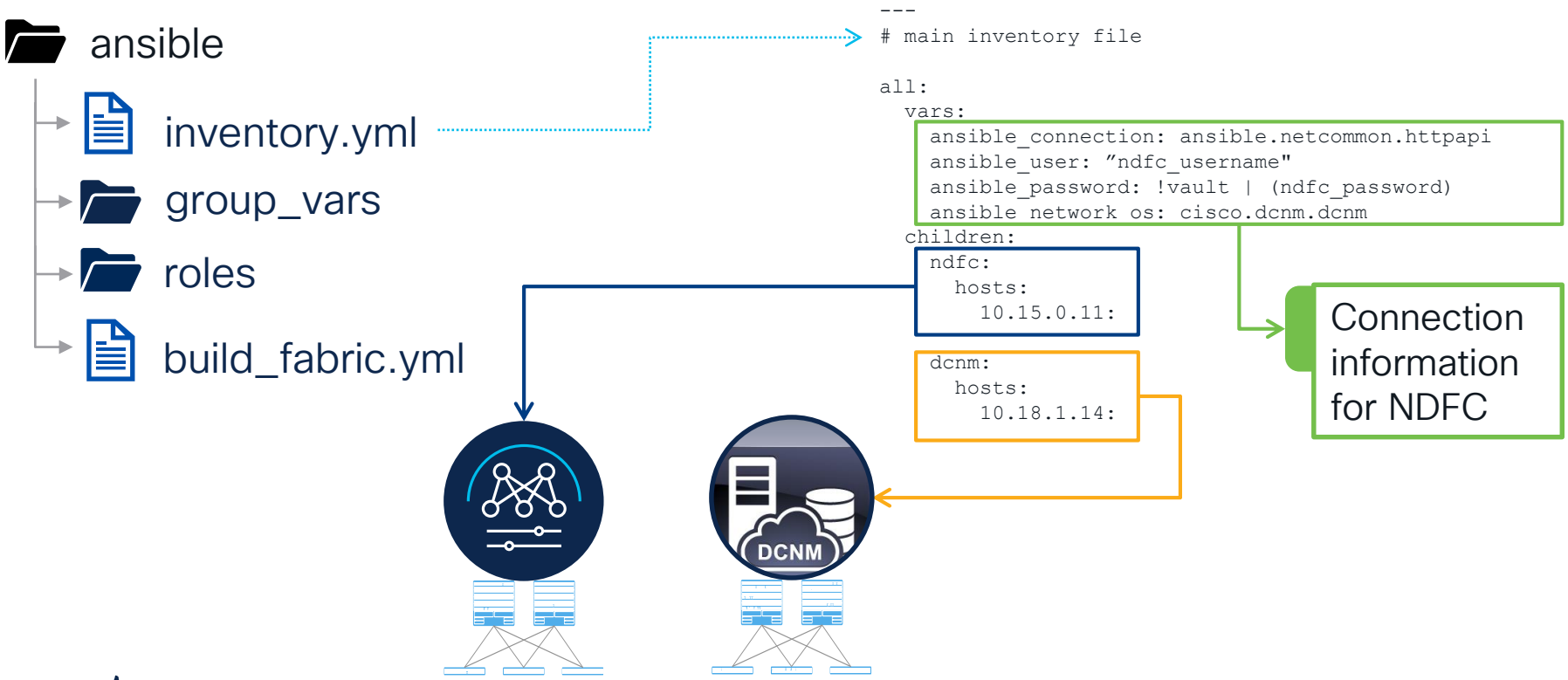
19

```
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

Where to do it



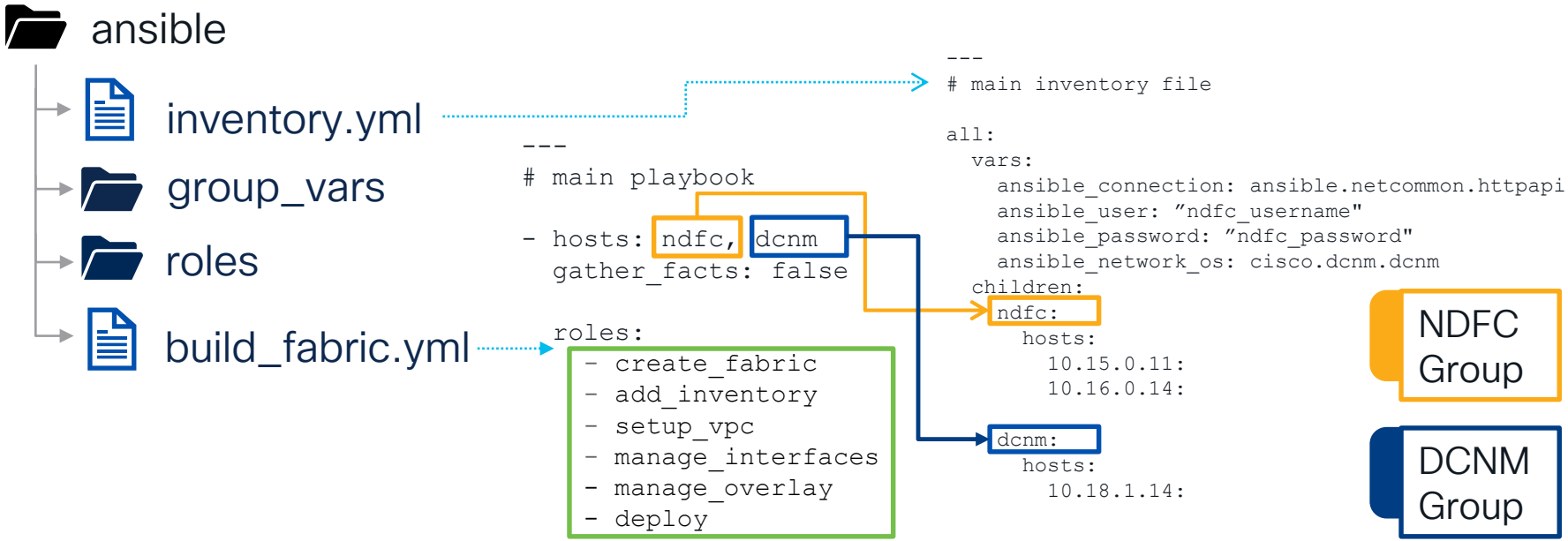
Ansible Playbook

What to do


```
---
# 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
```

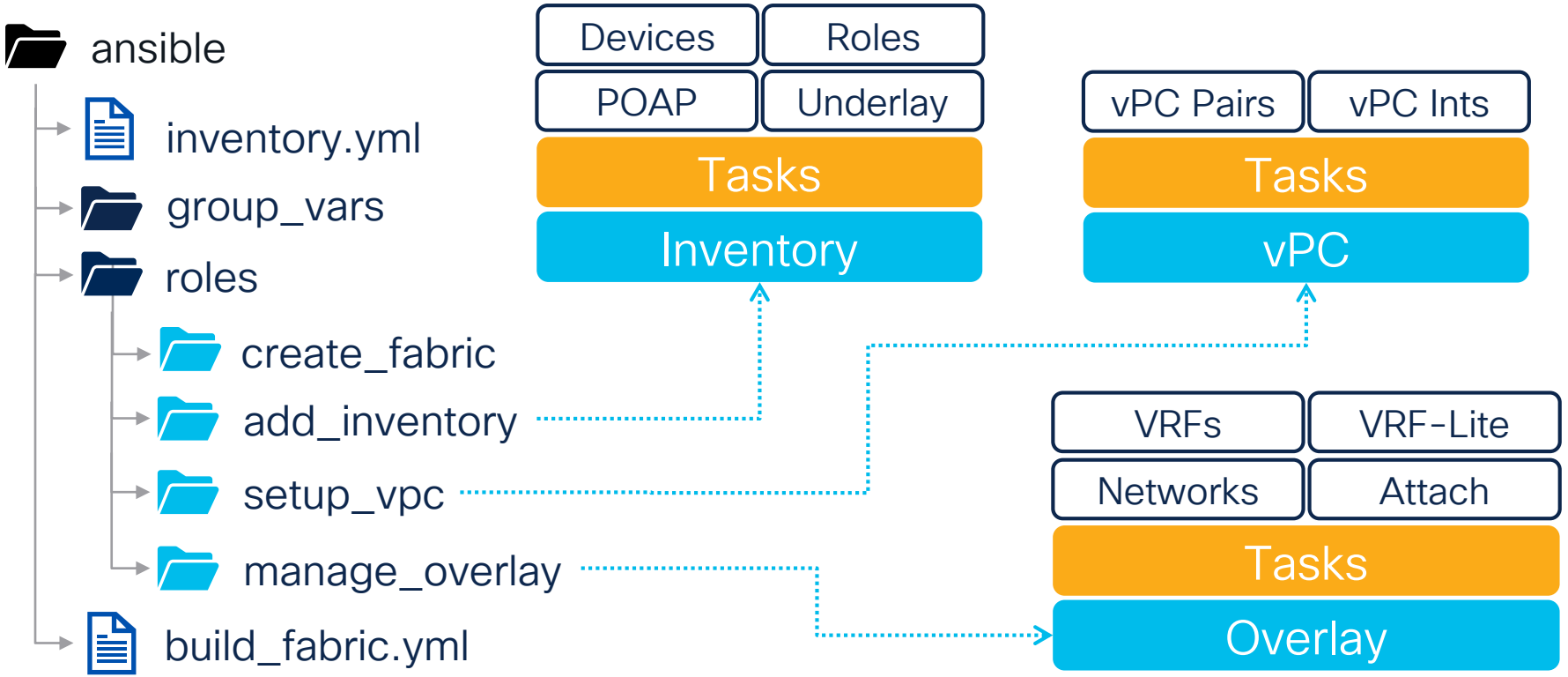
Ansible Playbook Relationships

Where to do it

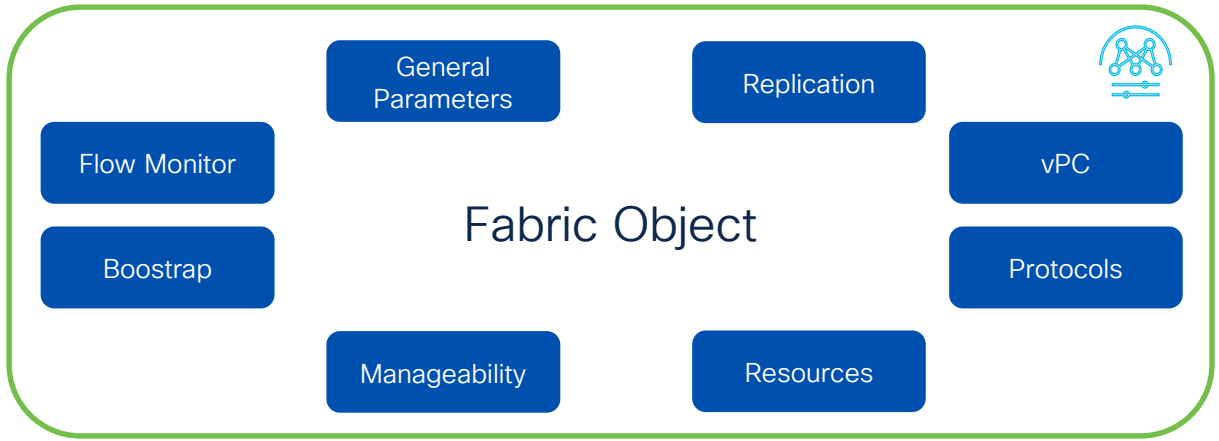


Ansible Roles – Tasks

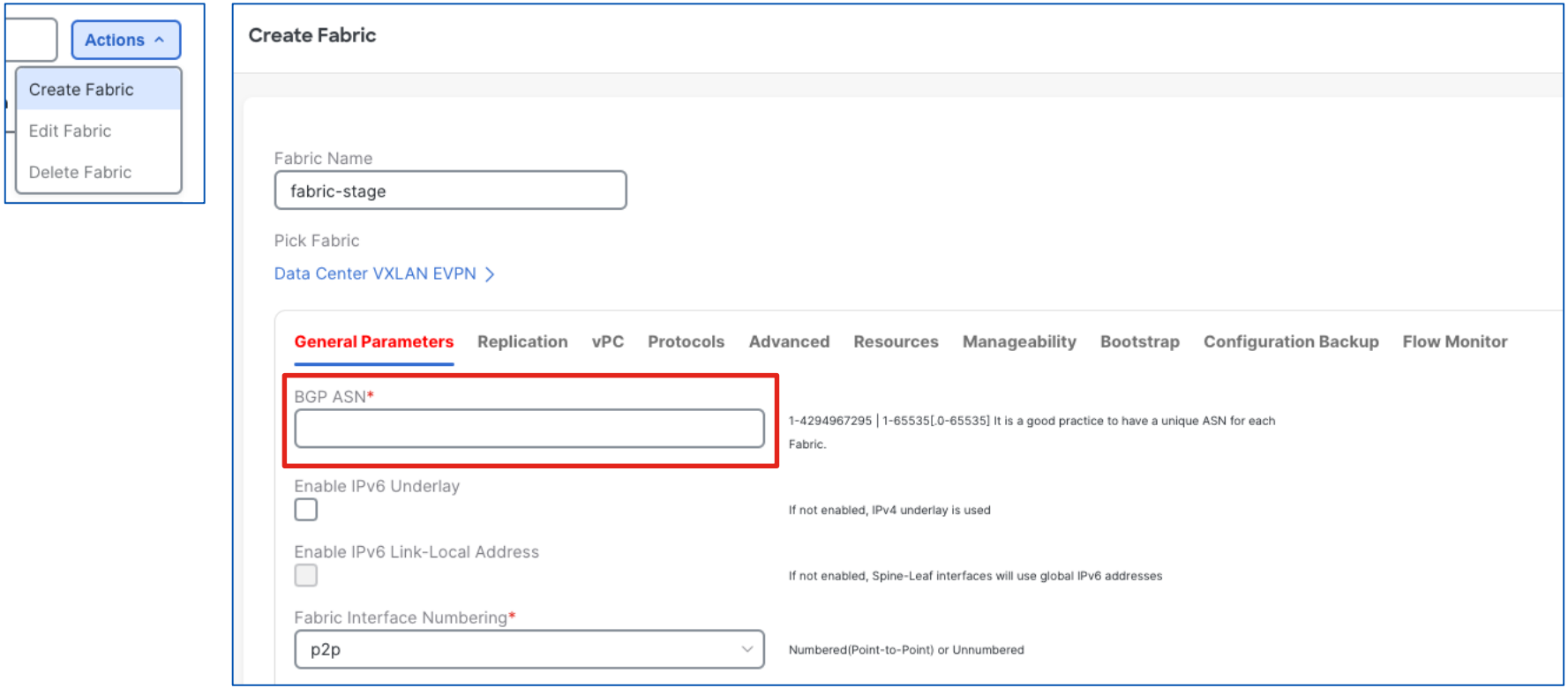
How to do it



Workflow Step1: Create VXLAN Fabric

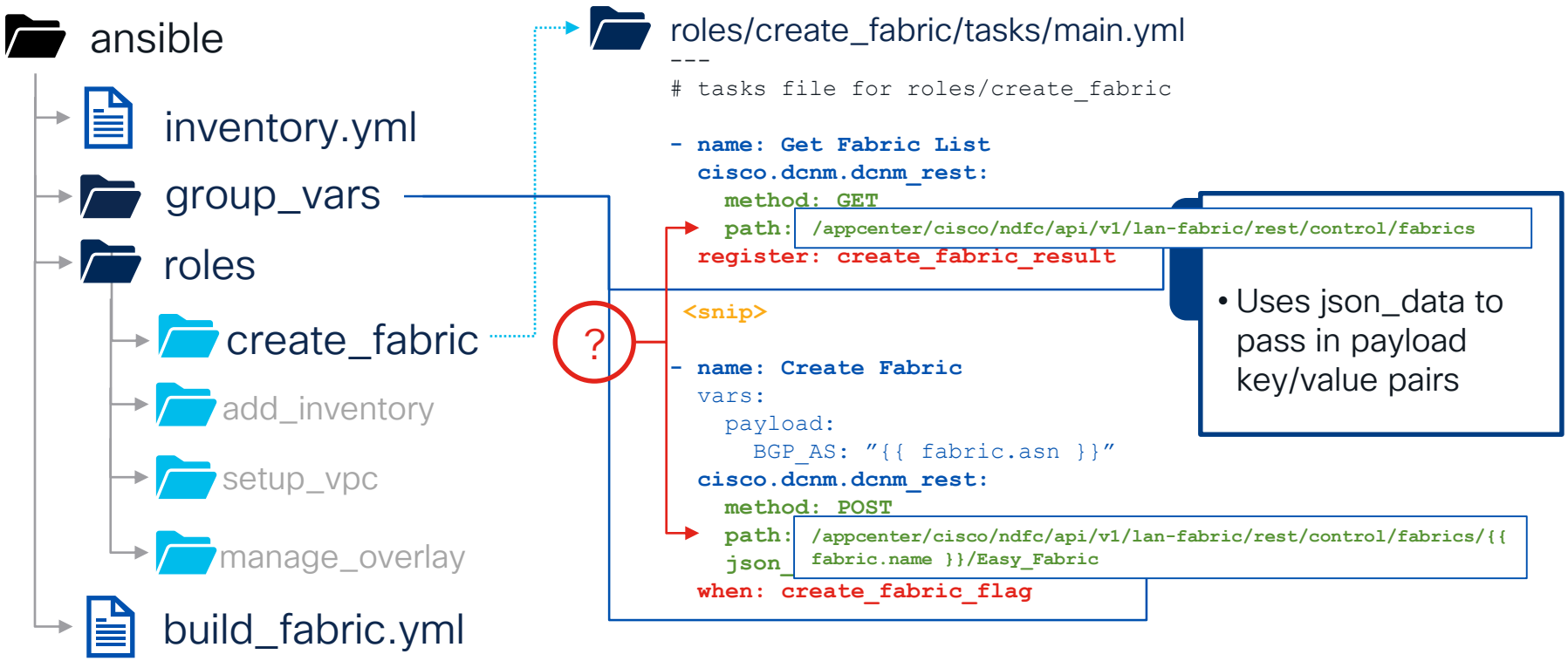


Workflow Step1: Create VXLAN Fabric



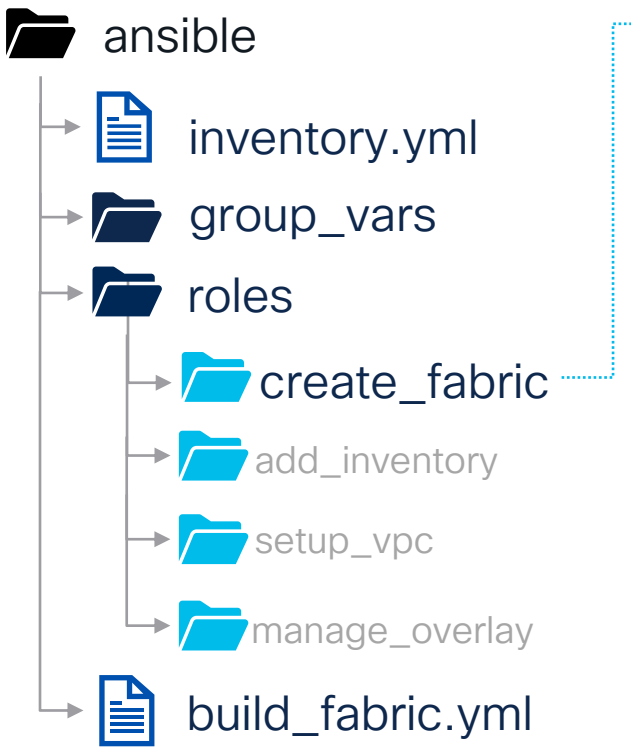
Ansible Roles – Fabric Tasks

How to do it



Ansible Roles – Fabric Tasks

How to do it



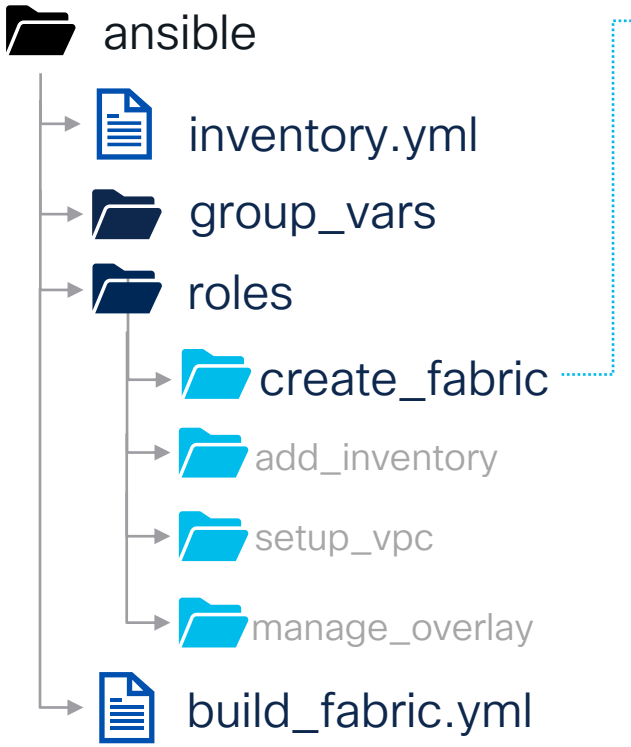
roles/create_fabric/tasks/main.yml

<snip>

```yaml
- name: Intialize create_fabric_flag
  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

How to do it



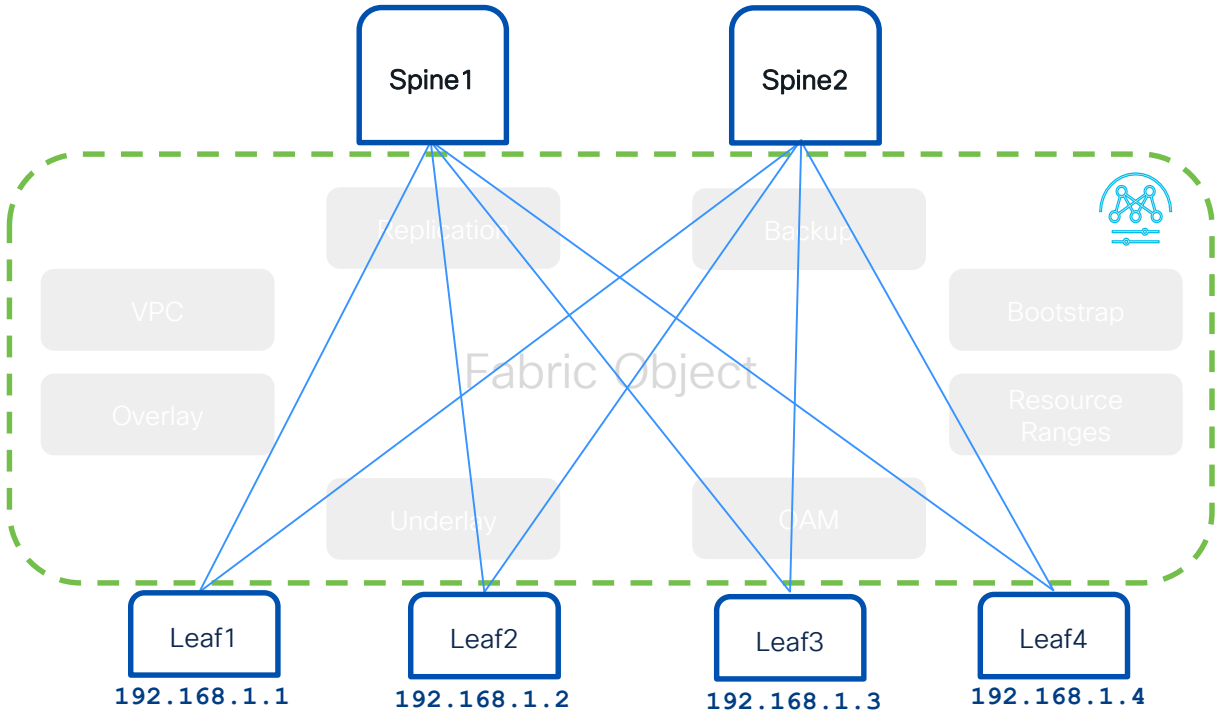
roles/create_fabric/tasks/main.yml

```
---
# 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
```

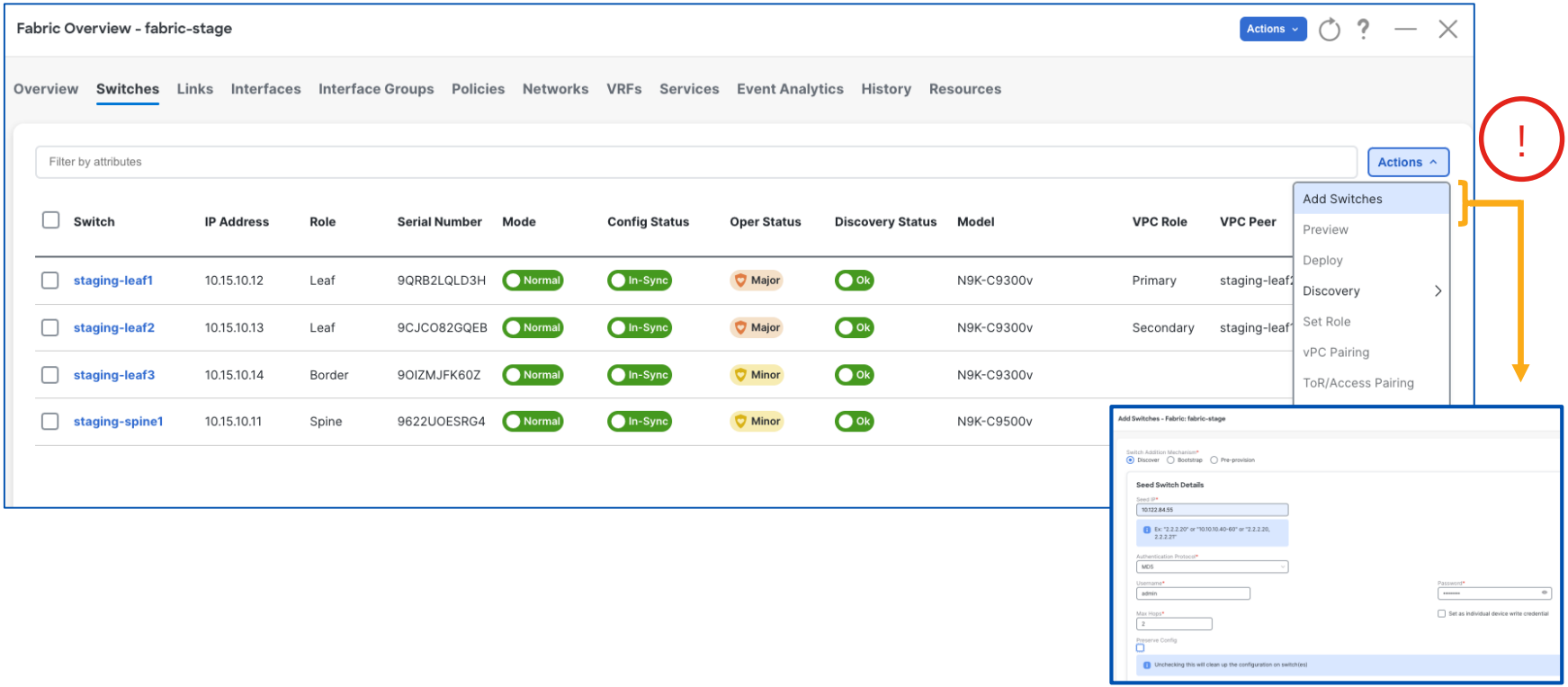
New Module! (3.5.0)

- Creates Fabric
- Handles Mutually Exclusive Properties
- Supports multiple types (VXLAN_EVPN, MSD, LAN_CLASSIC)

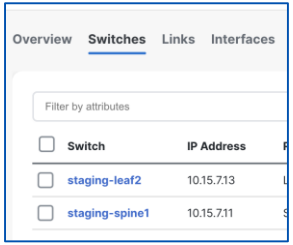
Workflow Step2: Add Inventory

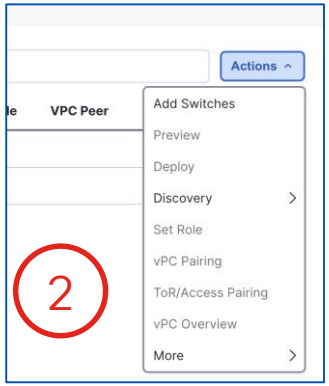


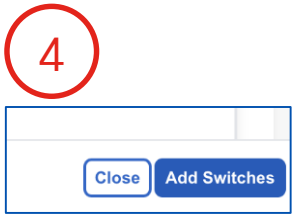
Workflow Step2: Add Inventory

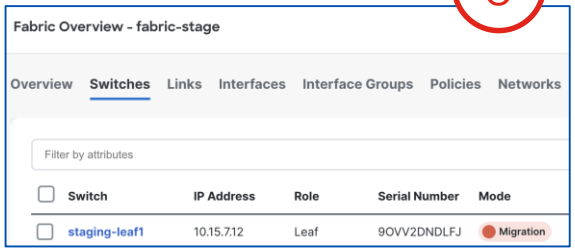


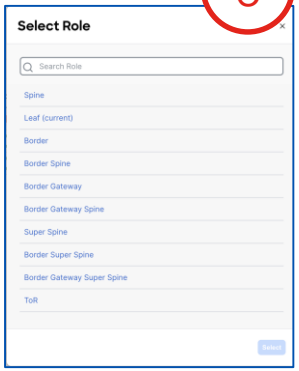
Workflow Step2: Add Inventory

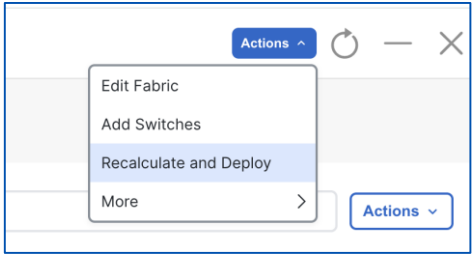






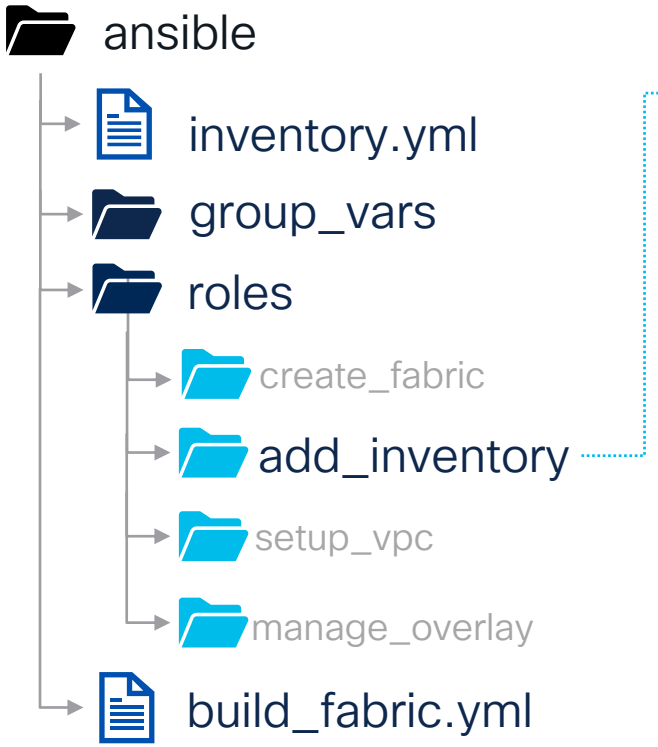






Ansible Roles – Inventory Tasks

How to do it

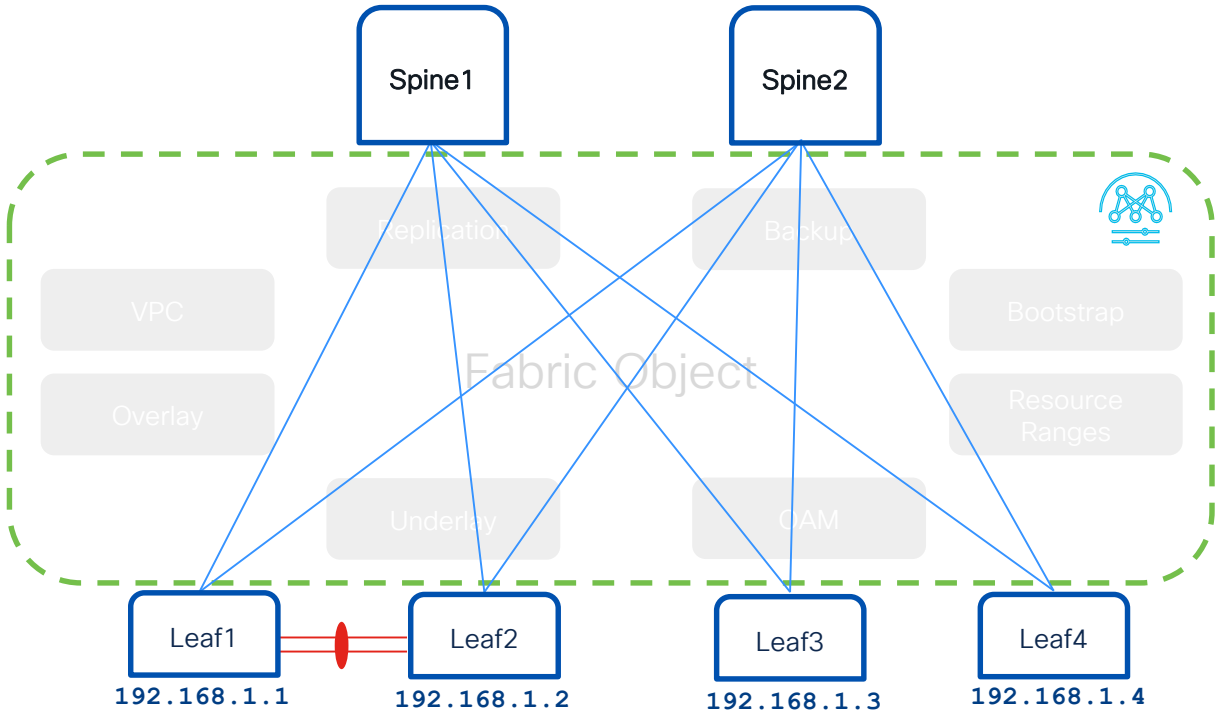


roles/add_inventory/tasks/main.yml

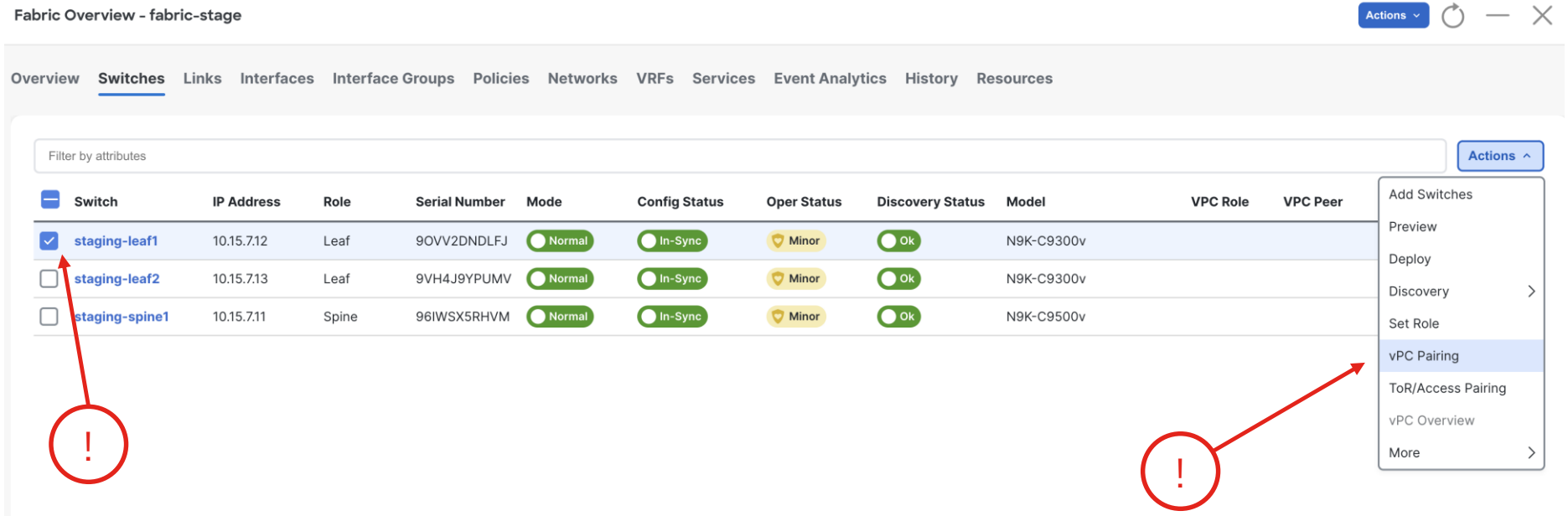
```
---
# tasks file for roles/add_inventory

- 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
<snip>
```

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

Workflow Step3: Setup VPC



Workflow Step3: Setup VPC



Ansible Roles – vPC Tasks

How to do it

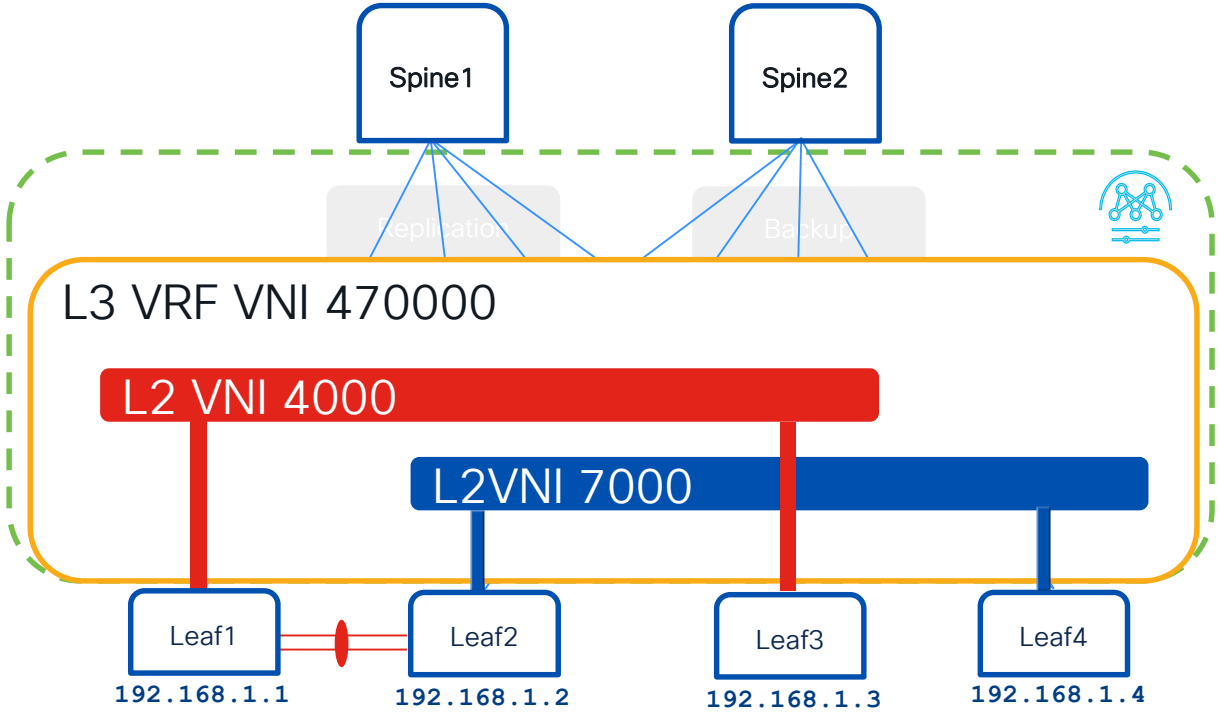
roles/setup_vpc/tasks/main.yml

```
---
# 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
```

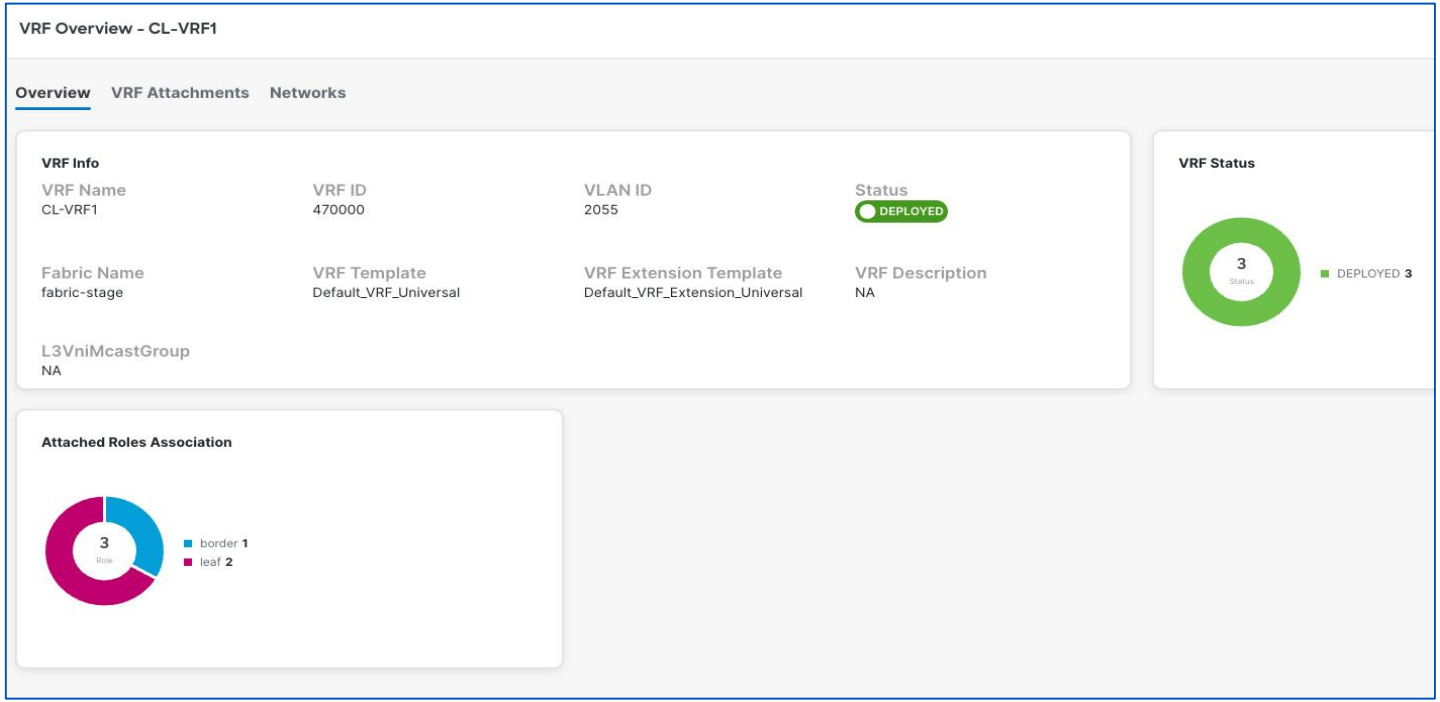
New Module! (3.5.0)

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

Workflow Step4: Manage Overlay



Workflow Step4: Add Overlay – VRFs / Networks



Ansible Roles – VRF / Net Tasks

How to do it

roles/manage_overlay/tasks/main.yml

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

- name: Add Overlay VRFs
  cisco.dcnm.dcnm_vrf:
    fabric: "{{ fabric.name }}"
    state: replaced
    config:
      - vr  me:  VRF1
        v  id: 4700
        an  d: 2055
        ttach
          - 192.  .1.
          - 192.16
          - 92.16  .3
          - 1  .1.4
```

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

- name: Add Overlay Networks
  cisco.dcnm.dcnm_network:
    fabric: "{{ fabric.name }}"
    state: overridden
    config:
      - net  NET7000
        vr  ame: CL-  F1
        n  7000
        v  88
        a  tach:
          192.168
          92.168
```

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

Jinja2 for VRF Data

ansible

roles

manage_overlay

tasks/main.yml

1

roles/manage_overlay/tasks/main.yml

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

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
```

2

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 %}
```

3

roles/manage_overlay/files/attach_vrfs.yml

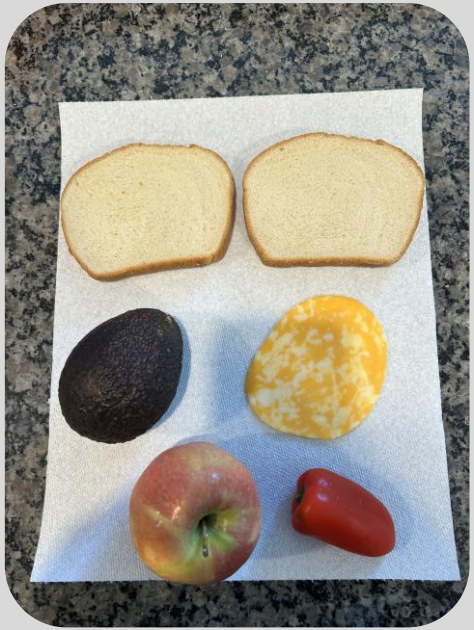
```
1  ---
2  # This file is auto-generated
3  # DO NOT EDIT MANUALLY
4  #
5  - vrf_name: vrf_devnet
6    vrf_id: 150001
7    vlan_id: 2000
8    attach:
9      - ip_address: 10.15.10.12
10     - ip_address: 10.15.10.13
11   deploy: true
```

```
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 }}"
```

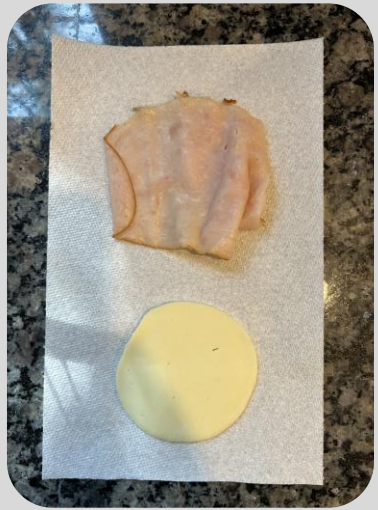
4

How Does One Merge a Sandwich?

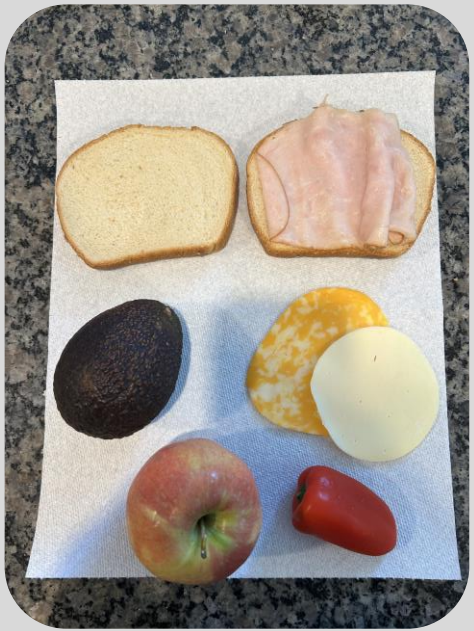
Initial State



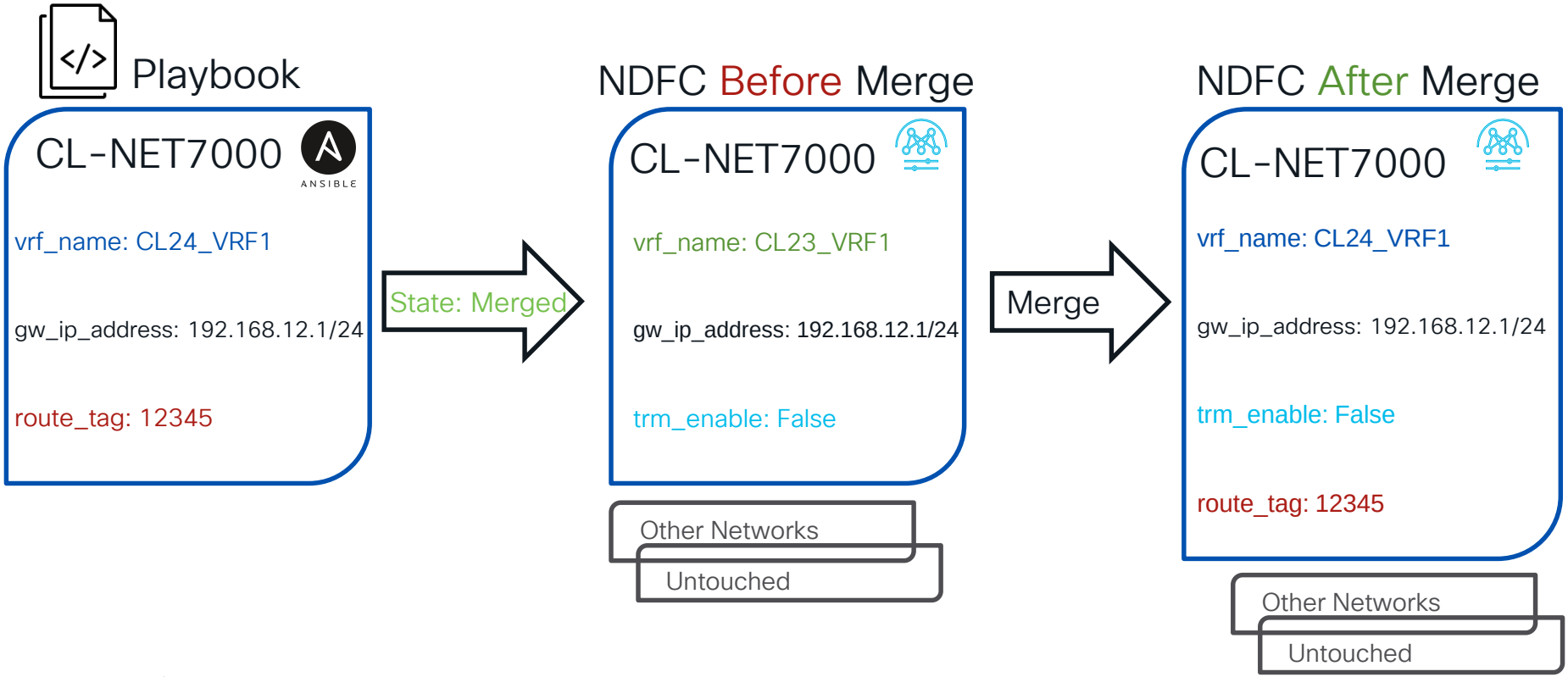
Merge Operation



Merged State



A Word About Module States - (Merged)

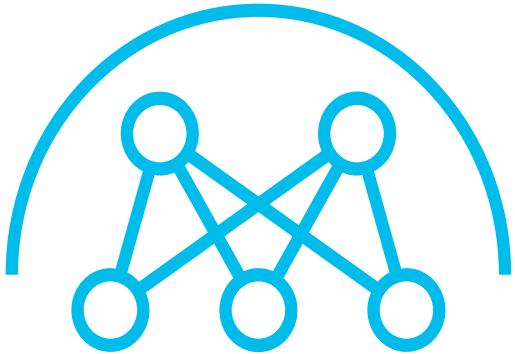


Who Is The Source Of Truth?

It's Me!

ANSIBLE

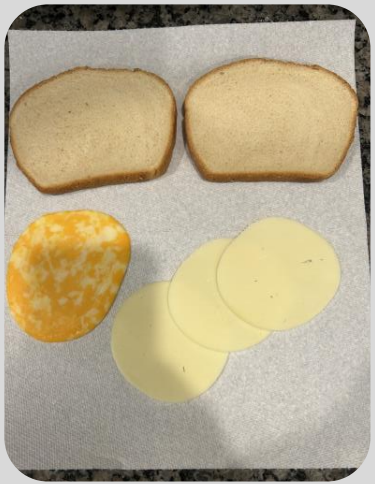
It's Me!



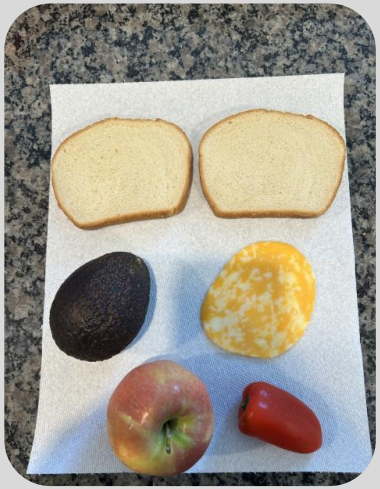
NDFC

How Does One Replace a Sandwich?

Desired State

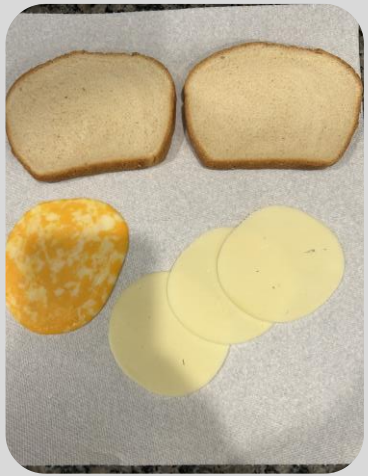


Actual State



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

State: Replaced

NDFC Before Replace

CL-NET7000

vrf_name: CL23_VRF1

gw_ip_address: 192.168.12.1/24

trm_enable: False

Other Networks

Untouched

Replace

NDFC After Replace

CL-NET7000

vrf_description: CL24_VRF1

gw_ip_address: 192.168.12.1/24

trm_enable: False

route_tag: 12345

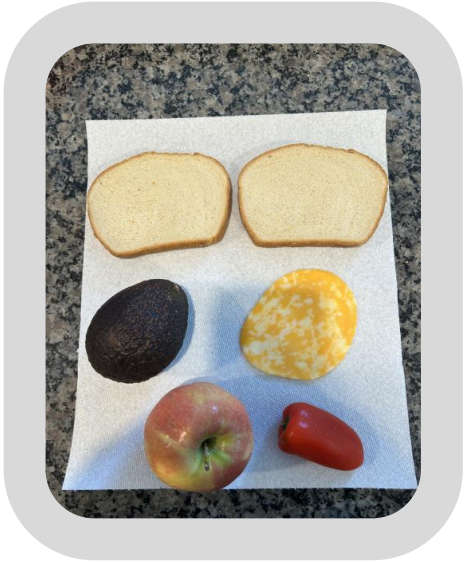
Other Networks

Untouched

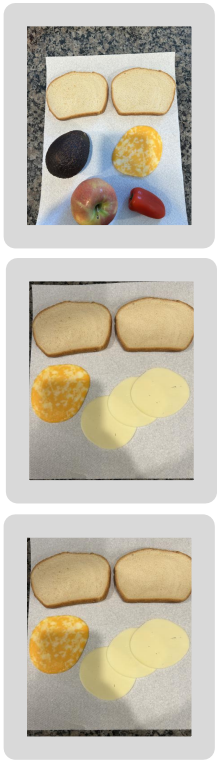
Source of Truth

How Does One Override a Sandwich?

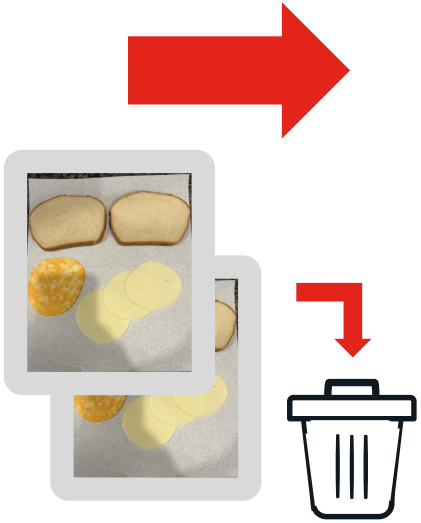
Desired State



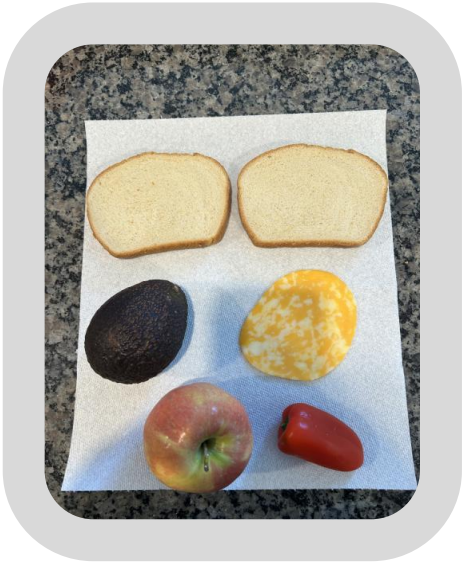
Actual State



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

NETWORK 2

NETWORK 3

Override

NDFC After Overridden

CL-NET7000

vrf_description: CL24_VRF1

gw_ip_address: 192.168.12.1/24

trm_enable: False

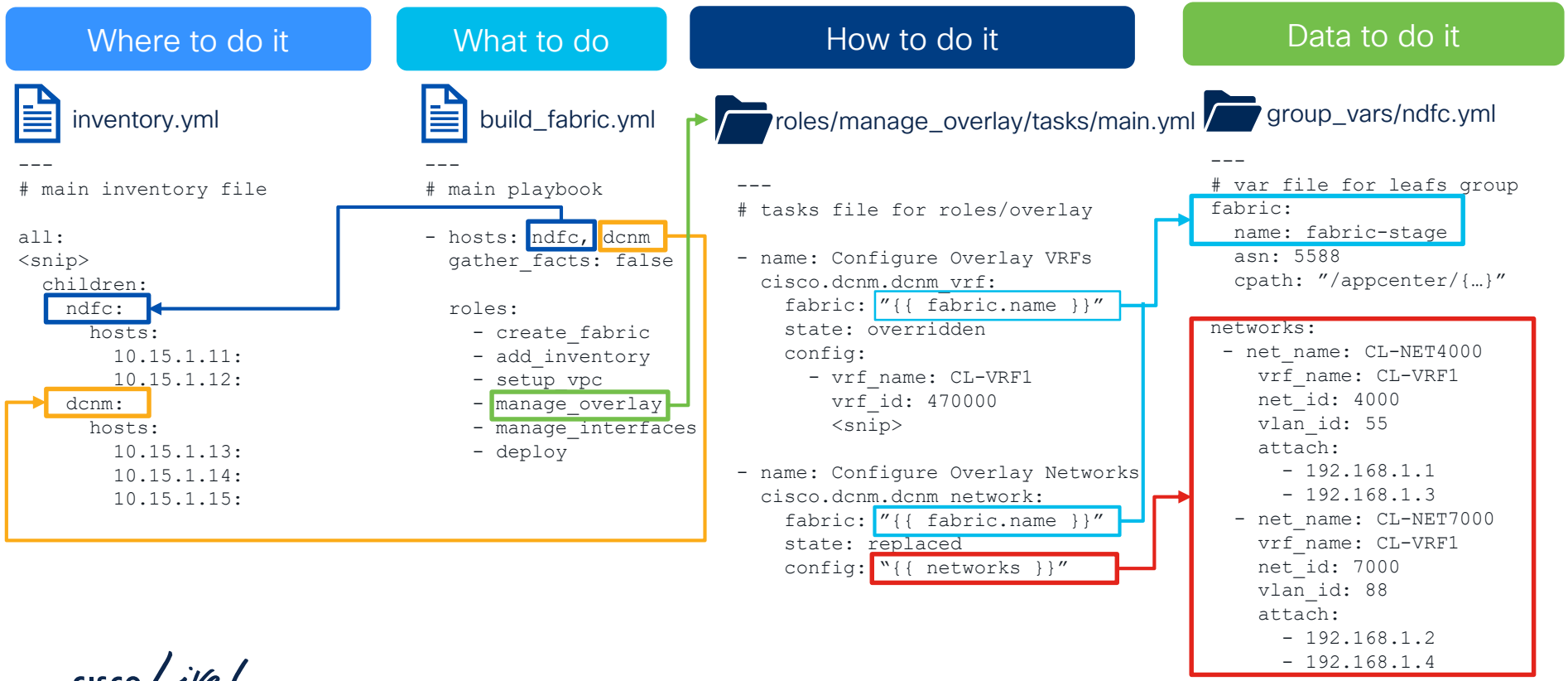
route_tag: 12345

NETWORK 2

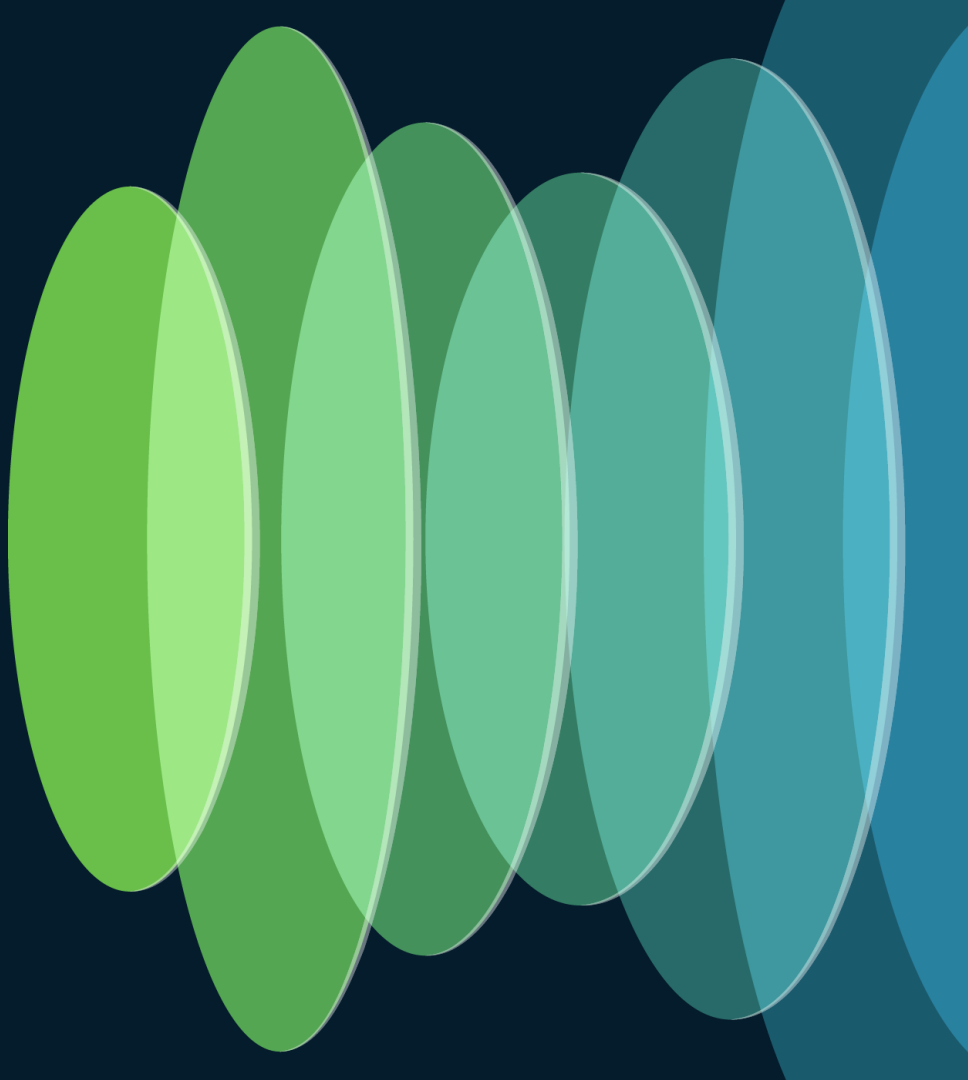
NETWORK 3

Source of Truth

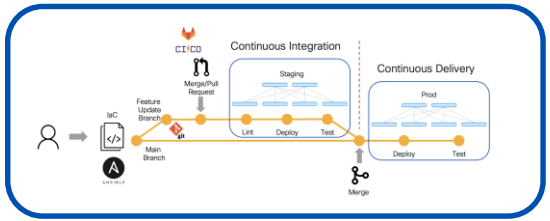
Putting It All Together

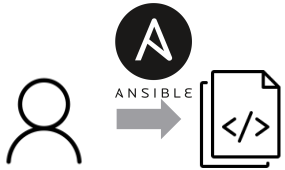


CI/CD Pipeline Demo



IaC NDFC Pipeline Demo





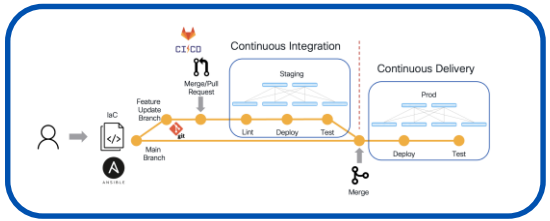
1

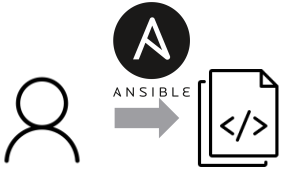
- Add Networks group_vars
- State: Overridden
- Commit / Push changes to GitLab **Staging** branch

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

CI/CD

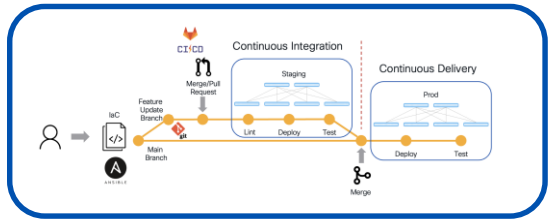
2

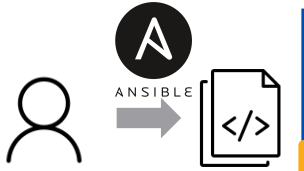
- Open Merge Request
- Triggers **Staging** Pipeline for Deploy and Verify

.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.dcnm
    - ansible-playbook -i hosts.stage.yml build_fabric.yml
```

IaC NDFC Pipeline Demo





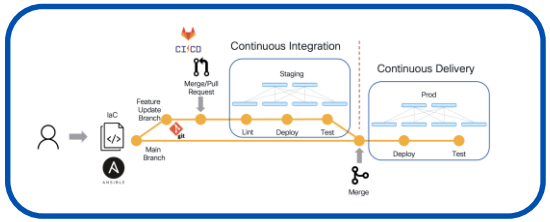
.gitlab-ci.yml

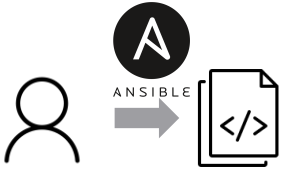
1

- Add Networks group_vars
- State: Overridden
- Commit / Push changes to GitLab Staging branch

```yaml
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





1

- Add Networks group_vars
- State: Overridden
- Commit / Push changes to GitLab **Staging** branch

CI/CD

2

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

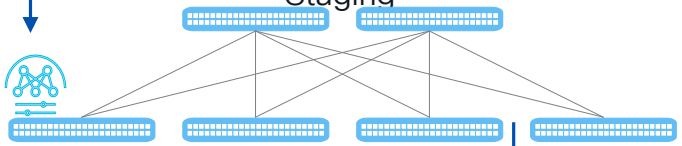
.gitlab-ci.yml

```
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.dcnm
    - ansible-playbook -i hosts.stage.yml build_fabric.yml
```

CL-NET4000

CL-NET7000

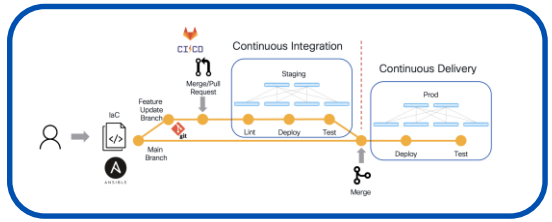
Staging

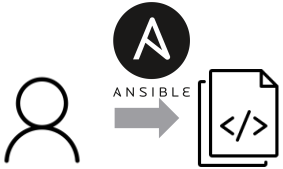


Other Networks

Other Networks

IaC NDFC Pipeline Demo





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

CI/CD

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

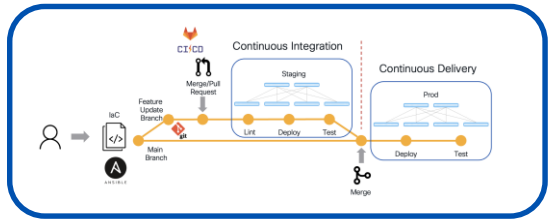
CI/CD

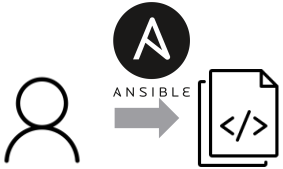
3
- Click Merge
- Triggers **Production** Pipeline for Deploy and Verify

.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





1

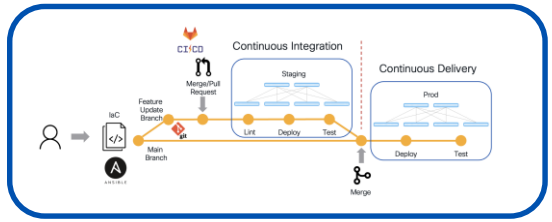
- Add Networks group_vars
- State: Overridden
- Commit / Push changes to GitLab Staging branch

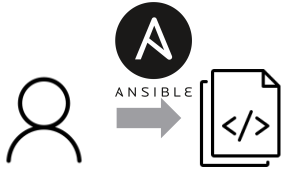
2

.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





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

CI/CD

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

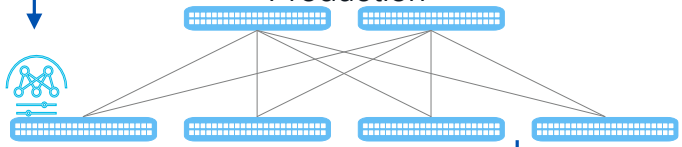
CI/CD

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

CL-NET4000

CL-NET7000

Production



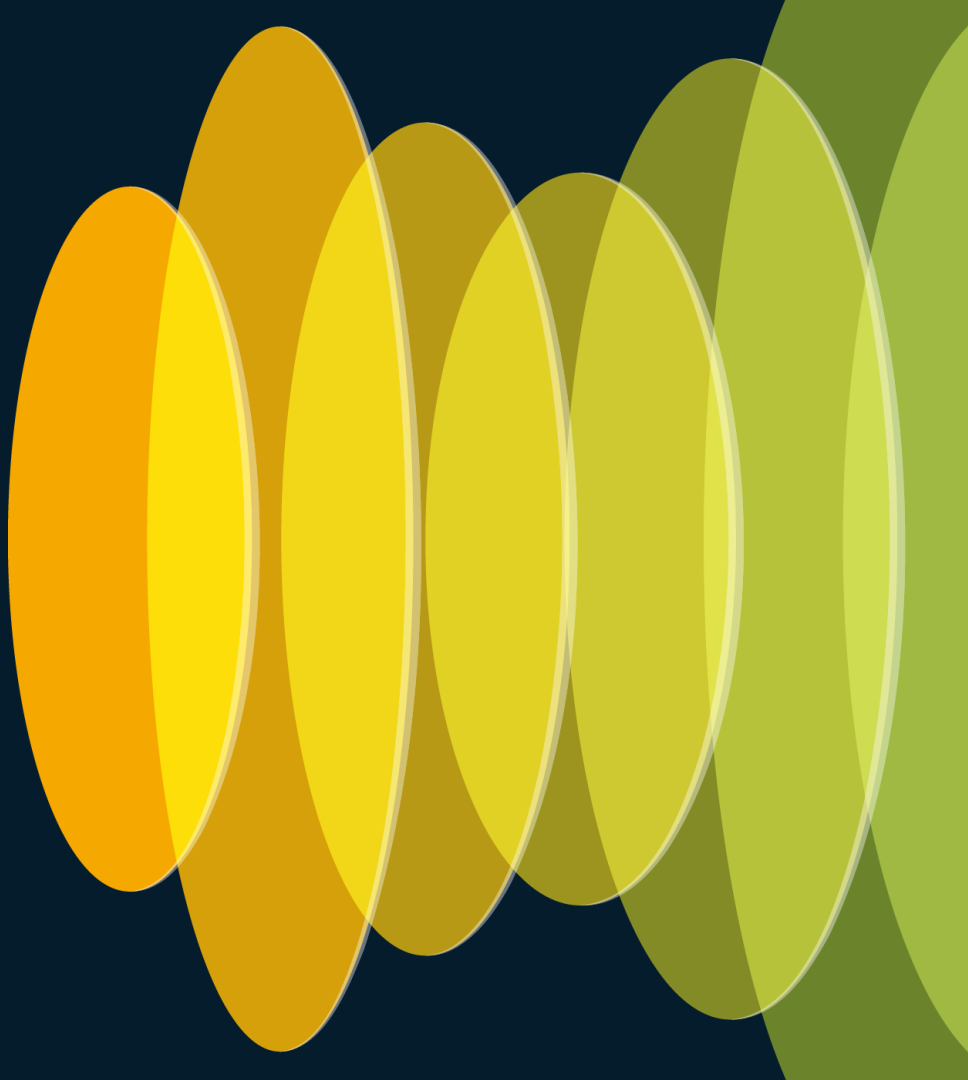
Other Networks

Other Networks

.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!

Many sessions
@ ***ciscolive.com***
with great material
also ***DevNet***
& ***developer.cisco.com***

I need help?!

Many Cisco *services*
To assist you in your
automation journey

Checkout
Services as Code
@ World of Solutions

More Information

- https://www.ansible.com/resources/get-started
- 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

- DEVWKS-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.

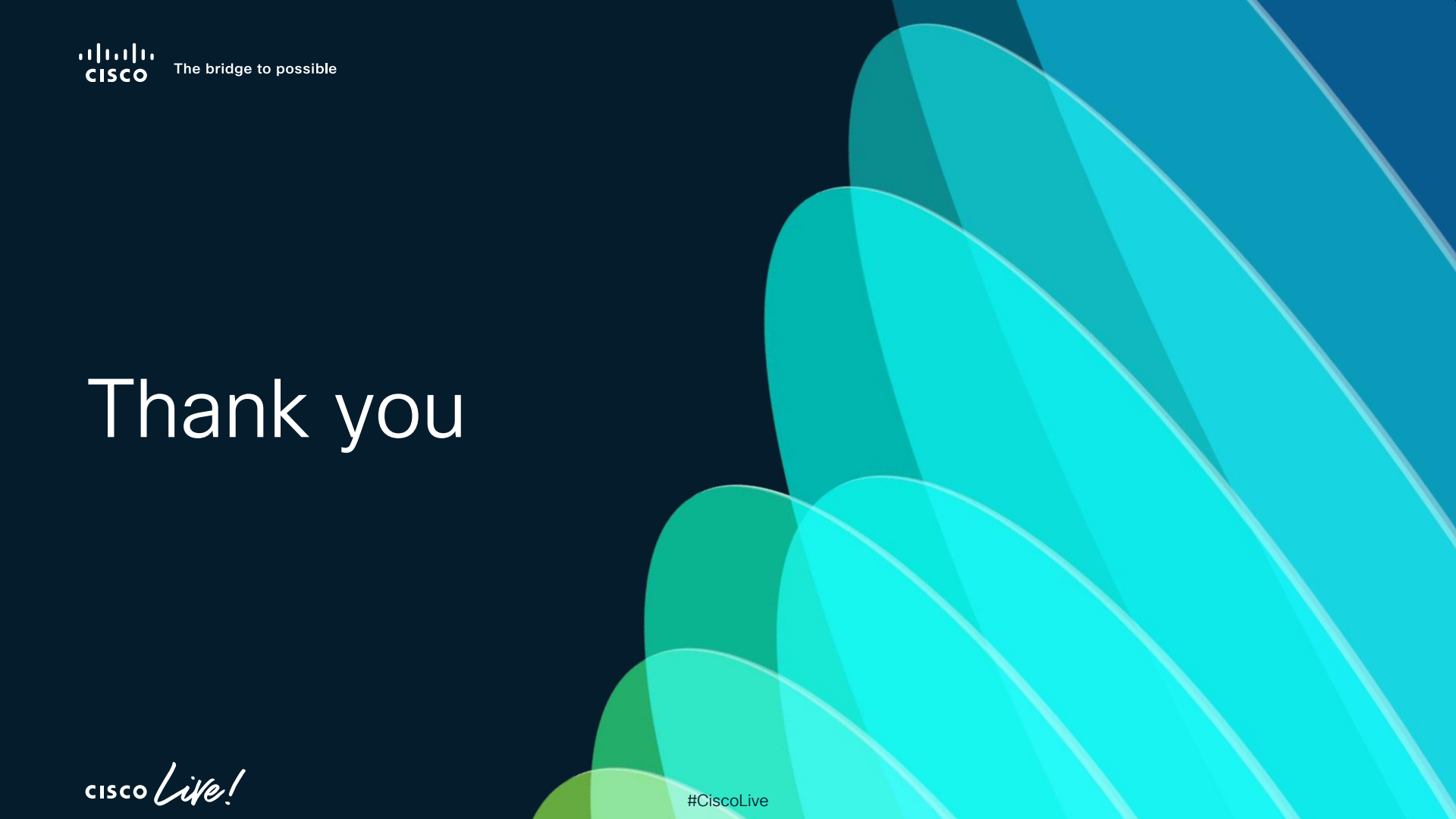
Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

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


The bridge to possible
Thank you
cisco Live!
#CiscoLive