Infrastructure as Code for NXOS and NDFC with Ansible

Mike Wiebe - Technical Leader

BRKDCN-2946

Webex App

Questions?

Use the Webex app to chat with the speaker after the session

How

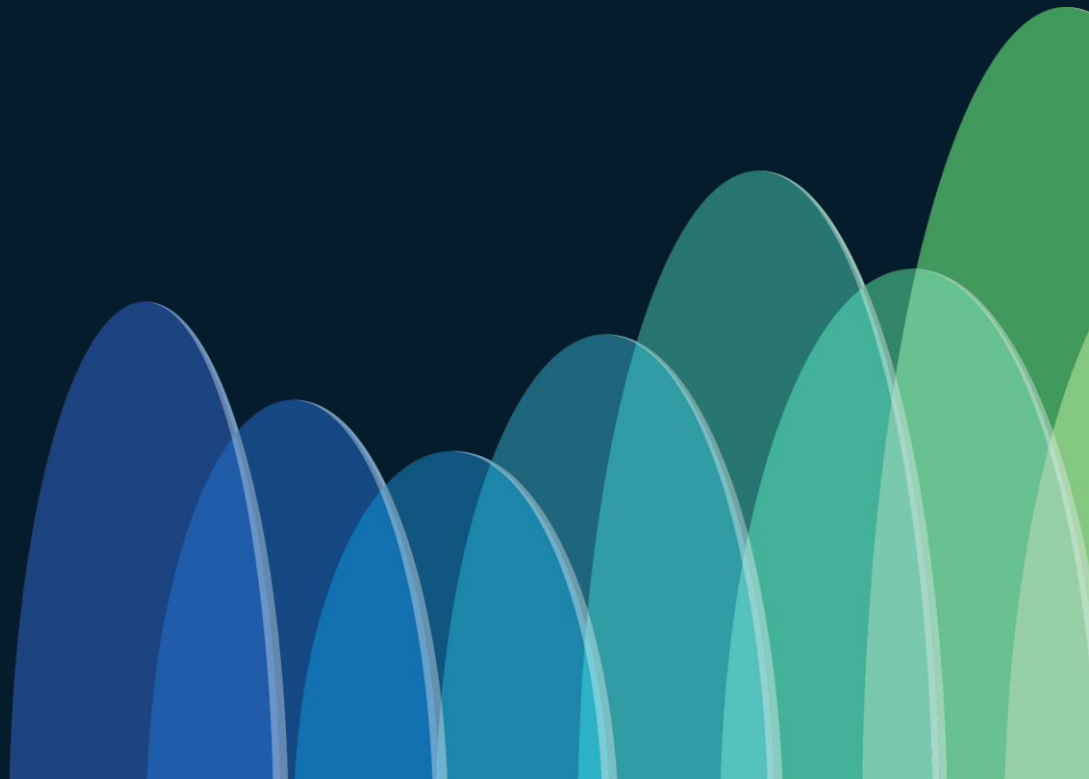
1. Find this session in the Cisco Events mobile app
2. Click “Join the Discussion”
3. Install the Webex app or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until February 28, 2025.

Agenda

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

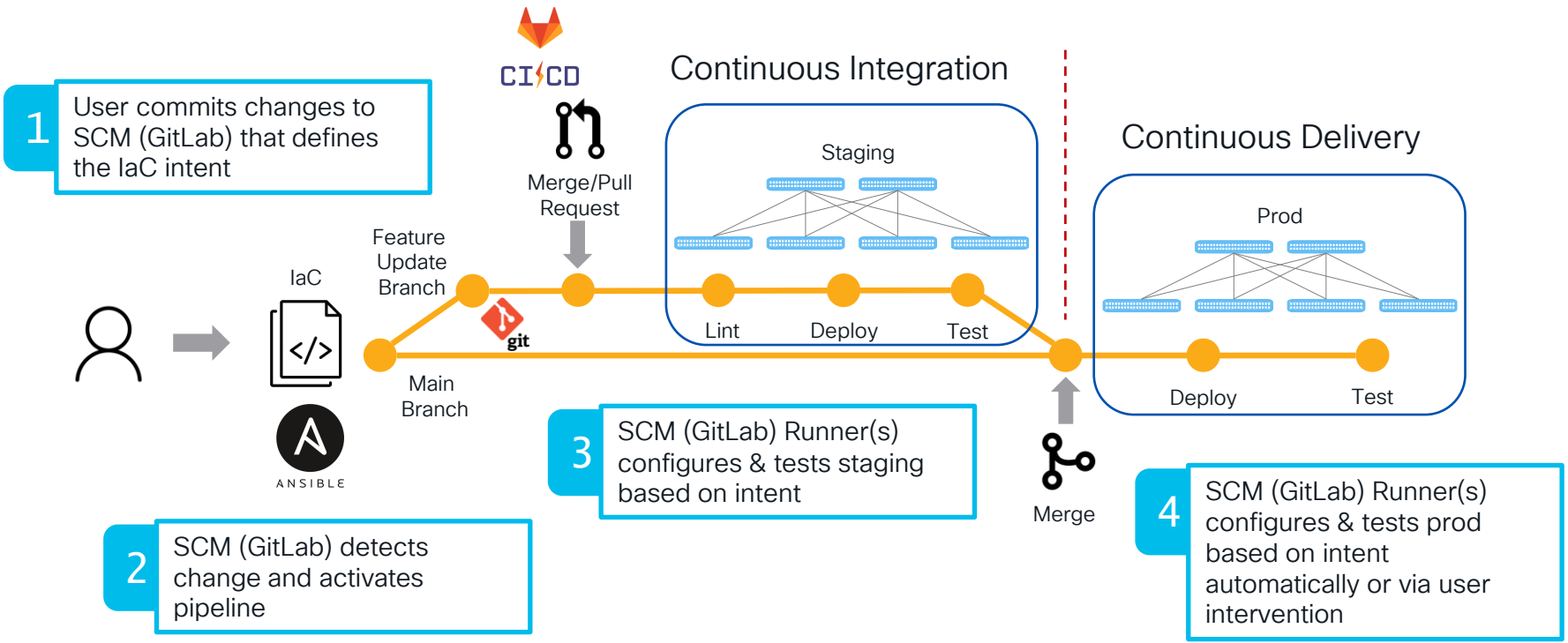
What is Infrastructure as Code?



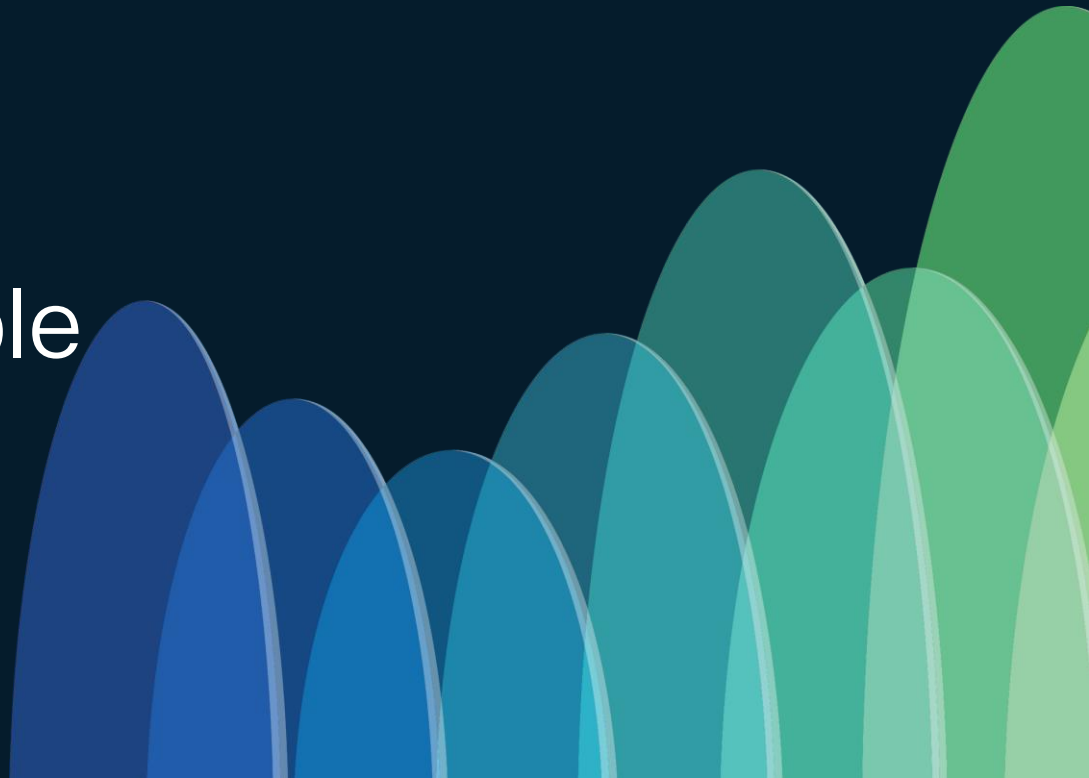
Infrastructure as Code for Network Ops

- 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



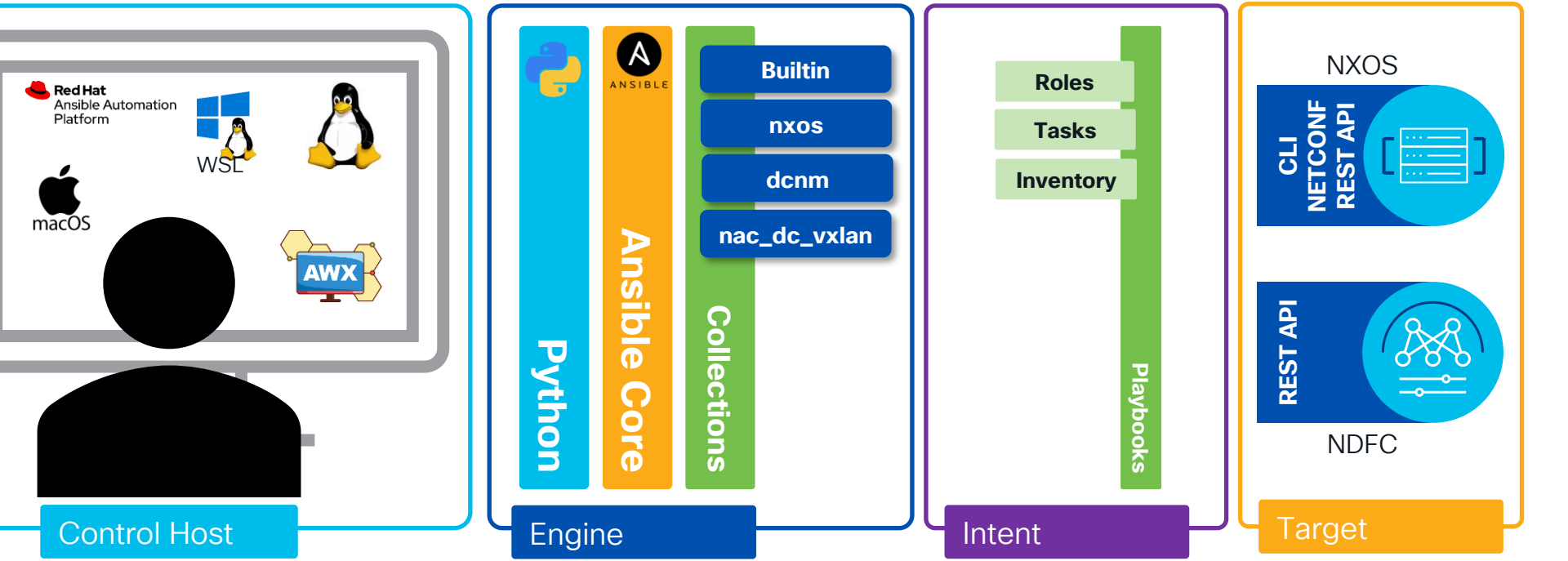
cisco Live!

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?



20/20 Hindsight Tech “Specs”

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

Installing Ansible

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

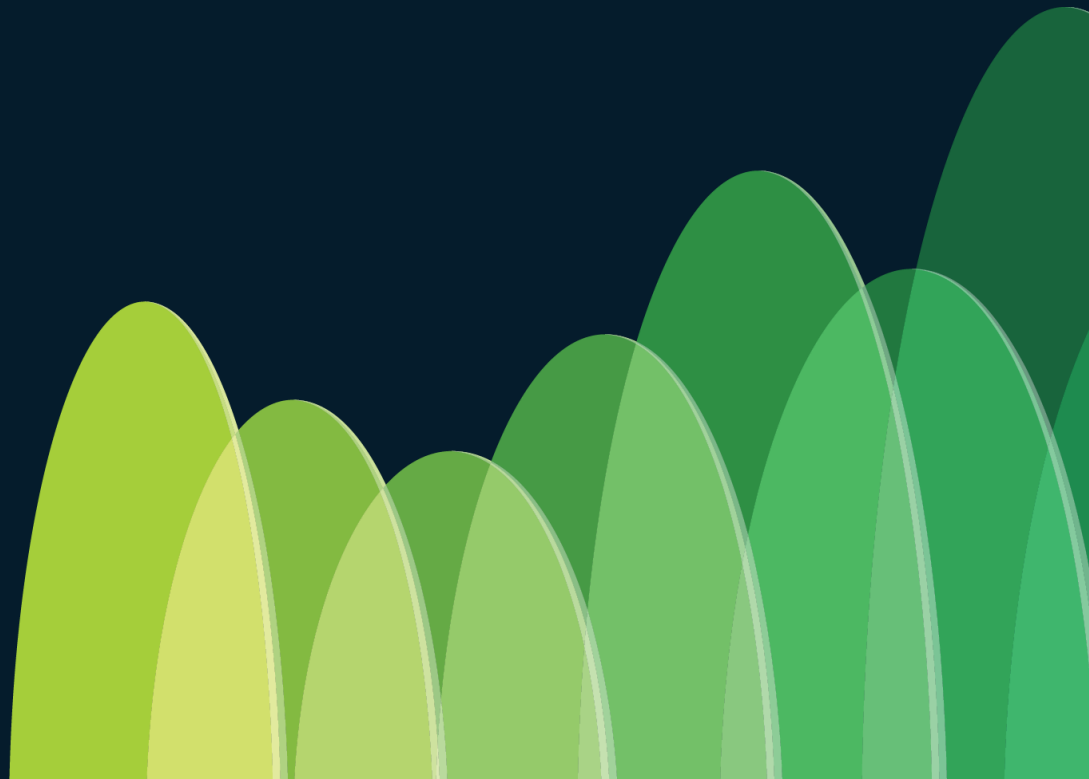
- Installs only the **core components**
- Collections must be installed by you
- Smaller footprint and more control
- Assures install of latest collection version released!

```
% 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

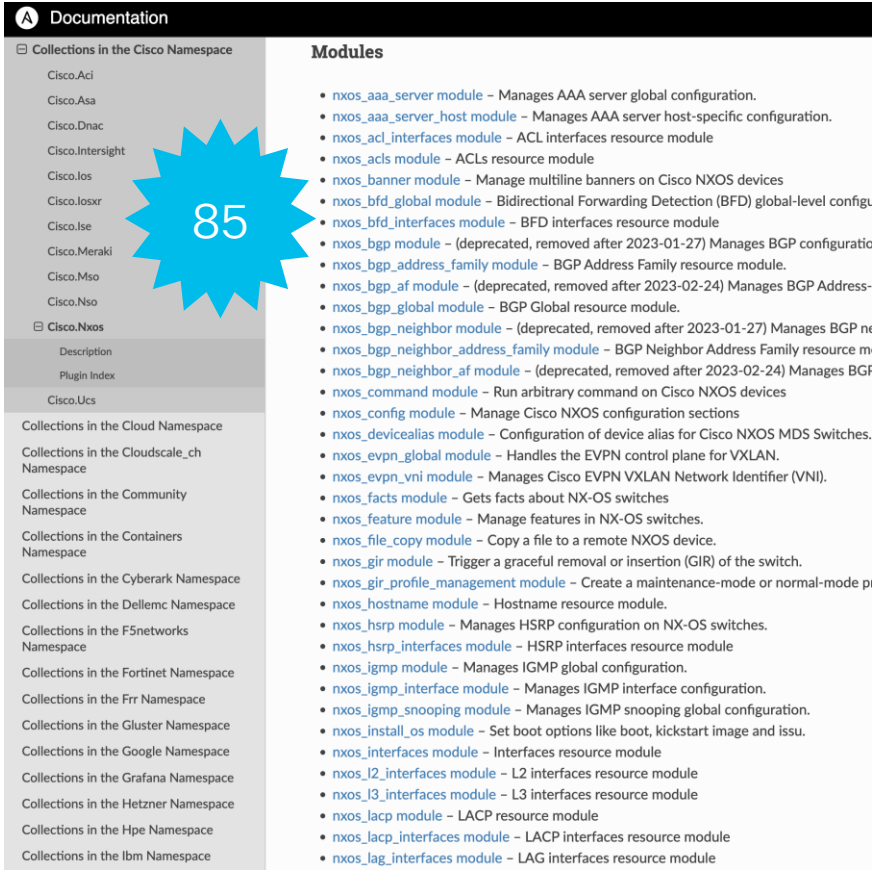
```
% ansible-galaxy collection install cisco.(nxos|dcnm|nac_dc_vxlan)
```

NXOS | DCNM | NaC - https://galaxy.ansible.com/cisco/(nxos | dcnm | nac_dc_vxlan)

Ansible Cisco.Nxos

Collection Modules

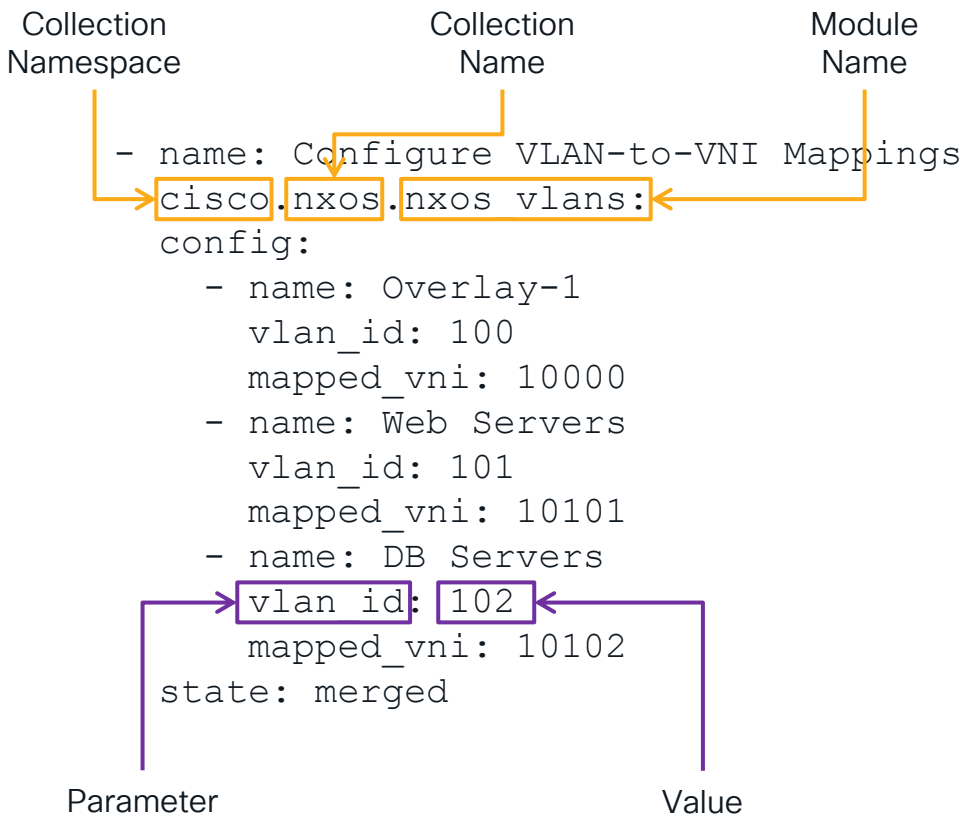
- Modules perform specific task like configure vlans, interfaces, OSPF, BGP, and more
- Documentation provides usage details, required variables, default variables, etc
- Actively maintained by RedHat with Cisco support



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

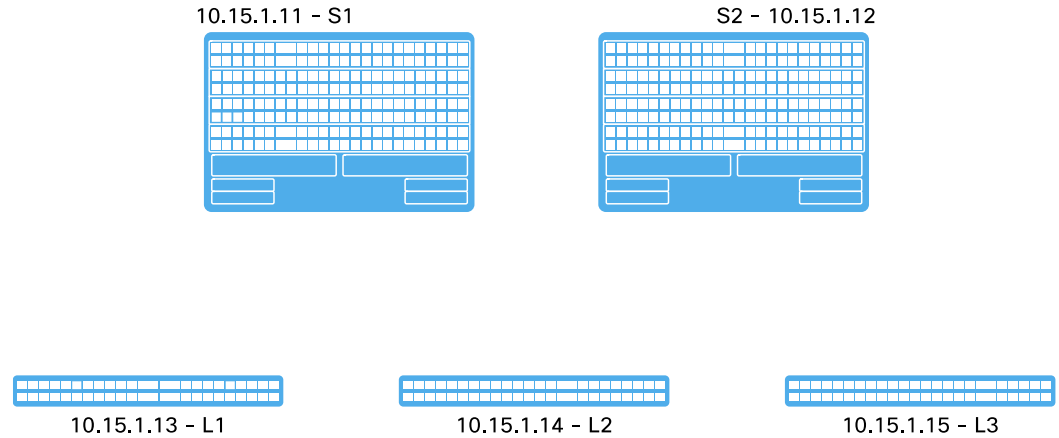


Ansible Concepts

VXLAN EVPN Fabric

IaC – Nexus as Code

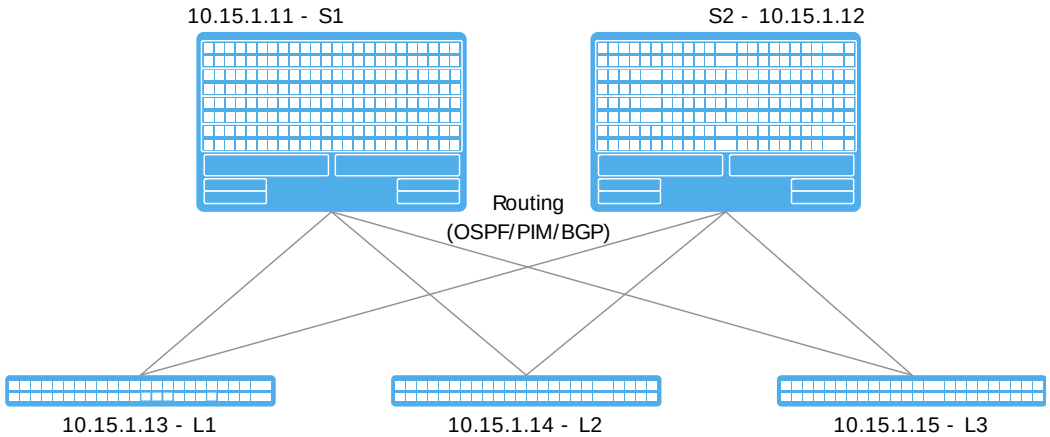
- Configure Hostnames, Features, etc (Common configuration)



VXLAN EVPN Fabric

IaC – Nexus as Code

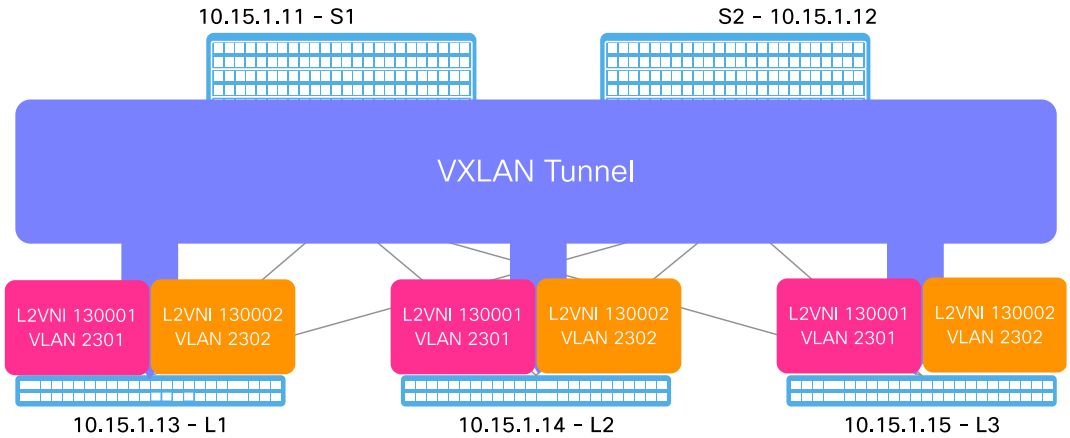
- Configure **Underlay** (Interfaces, Routing protocols, etc)



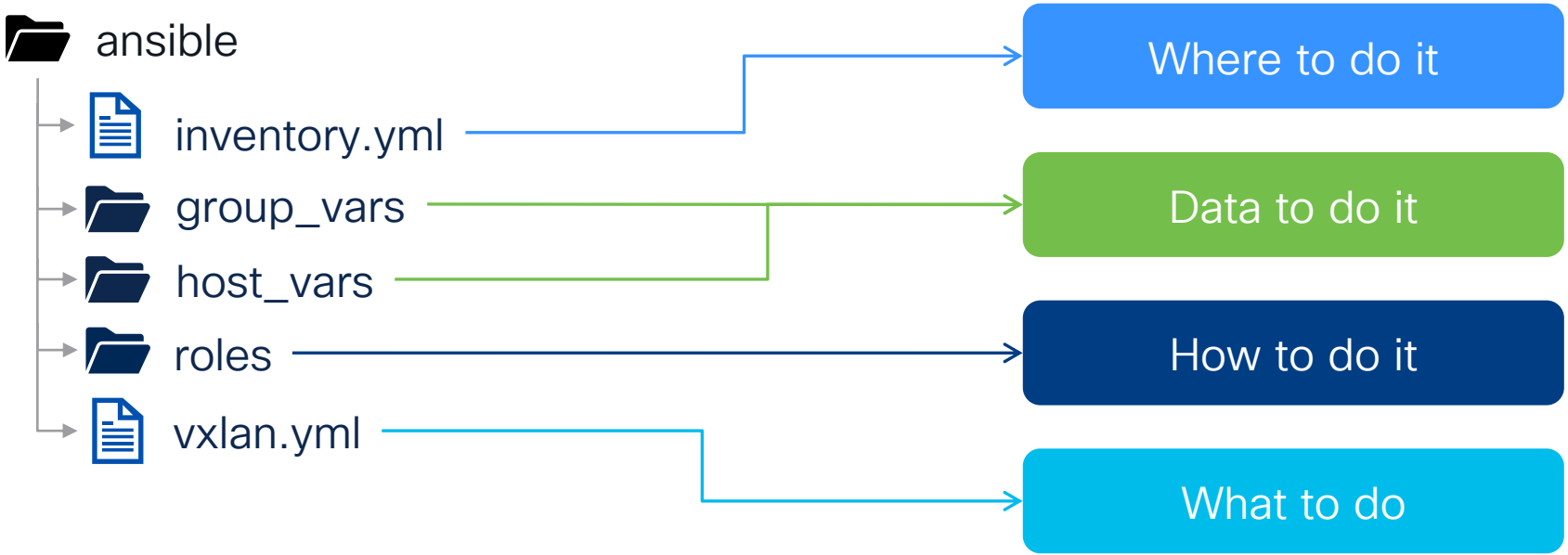
VXLAN EVPN Fabric

IaC – Nexus as Code

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



Ansible Directory Structure



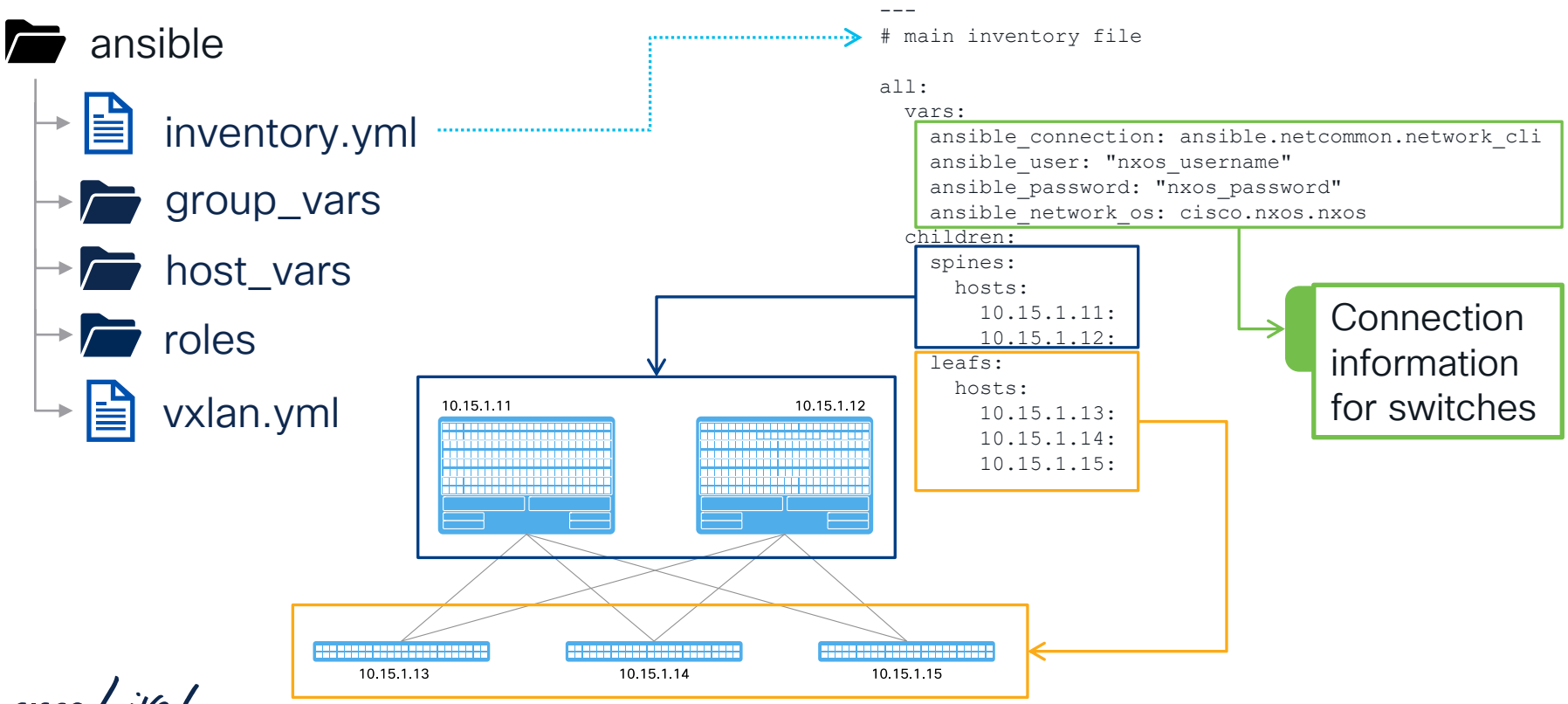
Ansible Playbook

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 Inventory

Where to do it

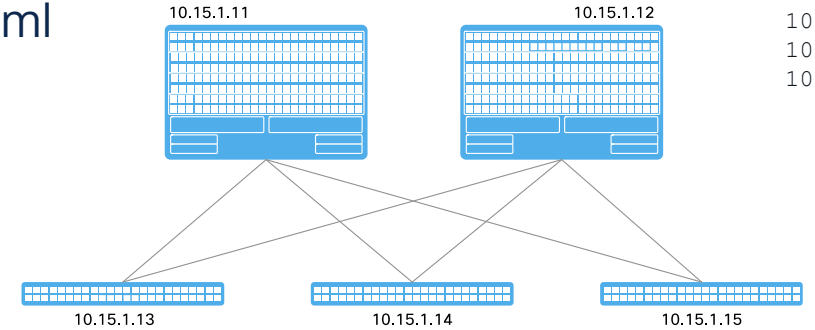


Ansible Inventory

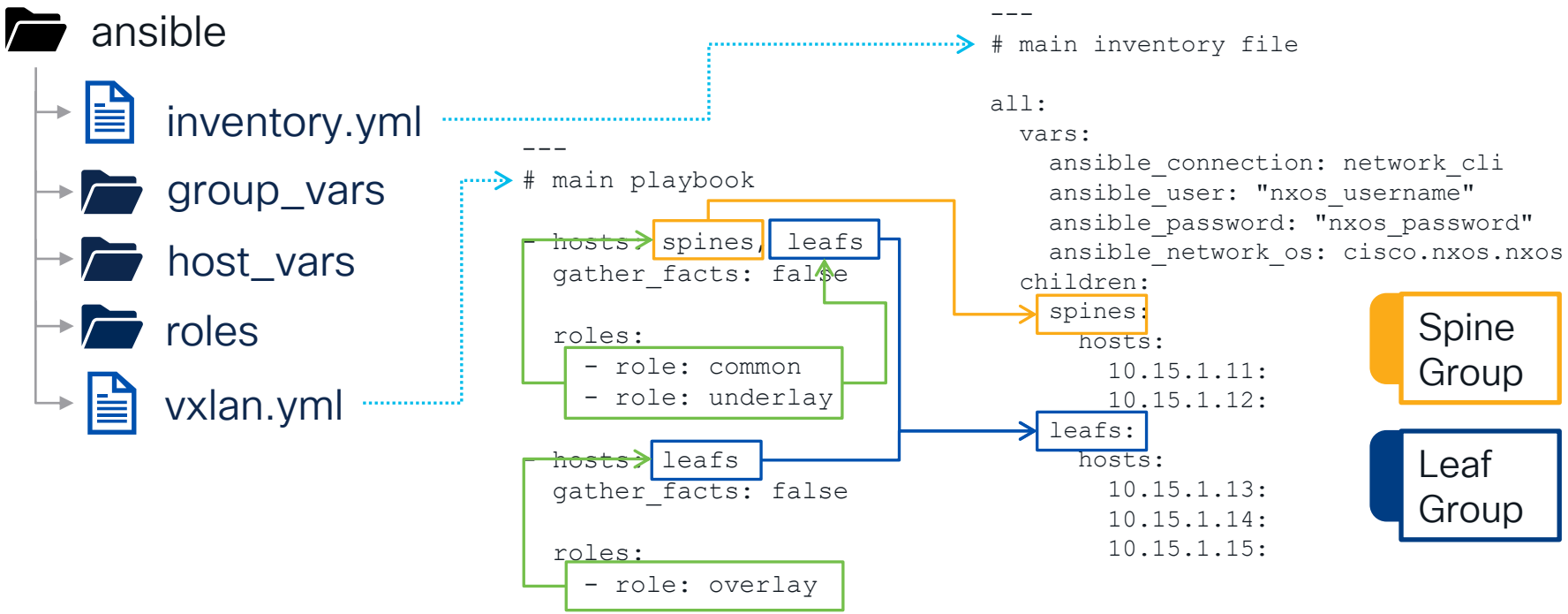
Where to do it


```
---
# 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`

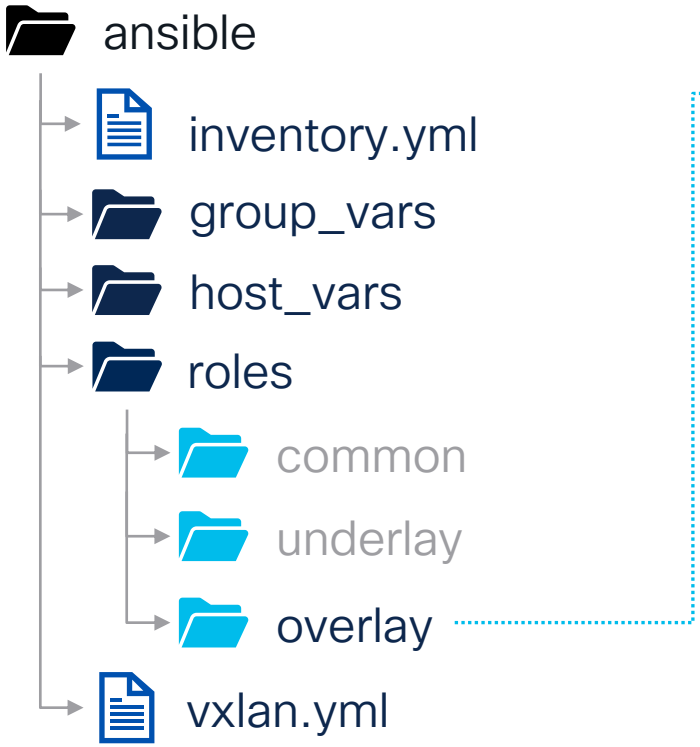


Ansible Playbook Relationships



Ansible Roles

How to do it

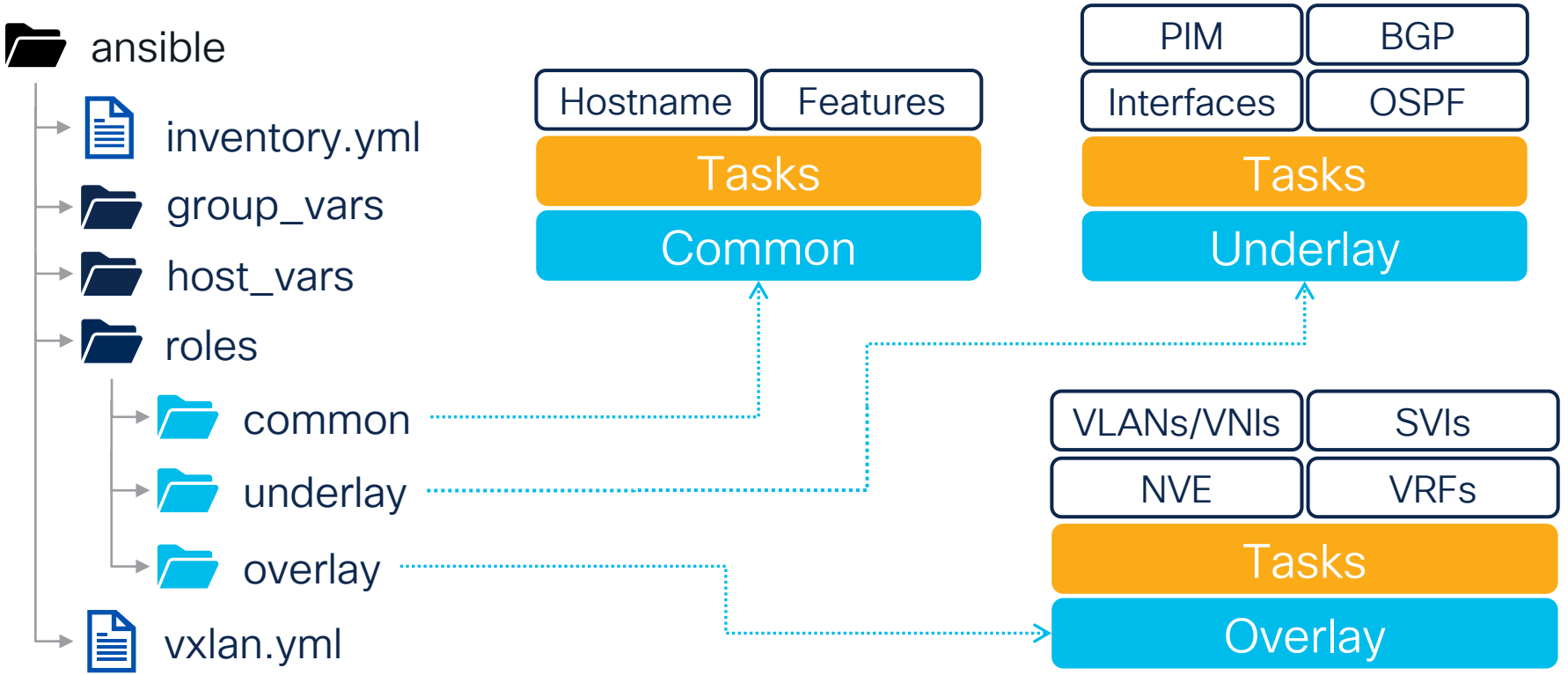



```
❯ ansible-galaxy init overlay
- Role overlay was created successfully
❯ tree overlay
overlay
├── README.md
├── tasks
│   └── main.yml
├── templates
└── vars
    └── main.yml
```

This is how we do it

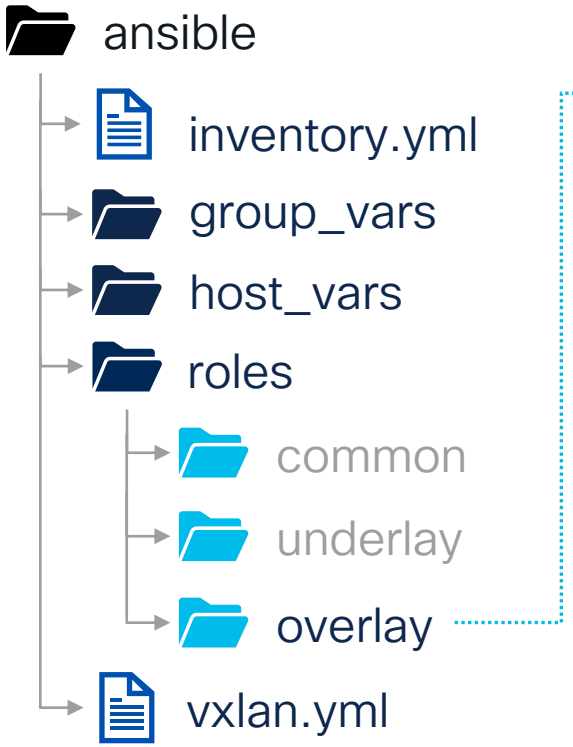
Ansible Roles – Tasks

How to do it



Ansible Roles – Tasks

How to do it



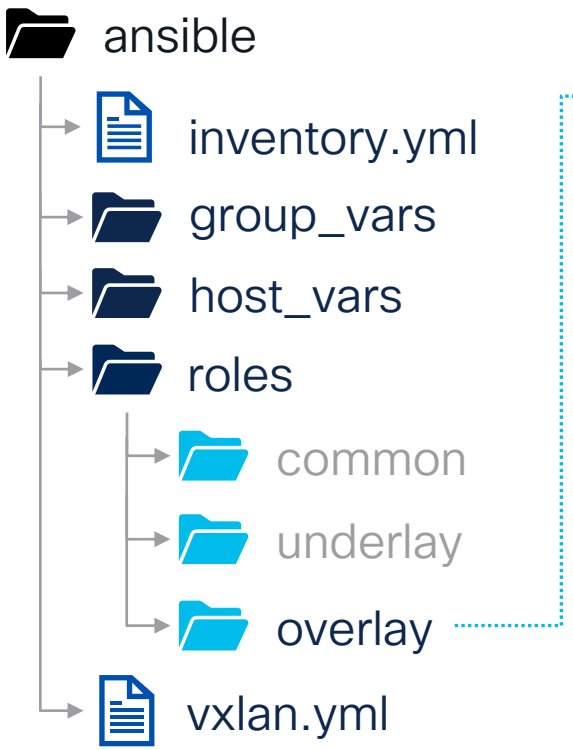
roles/overlay/tasks/main.yml

```
---
# 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



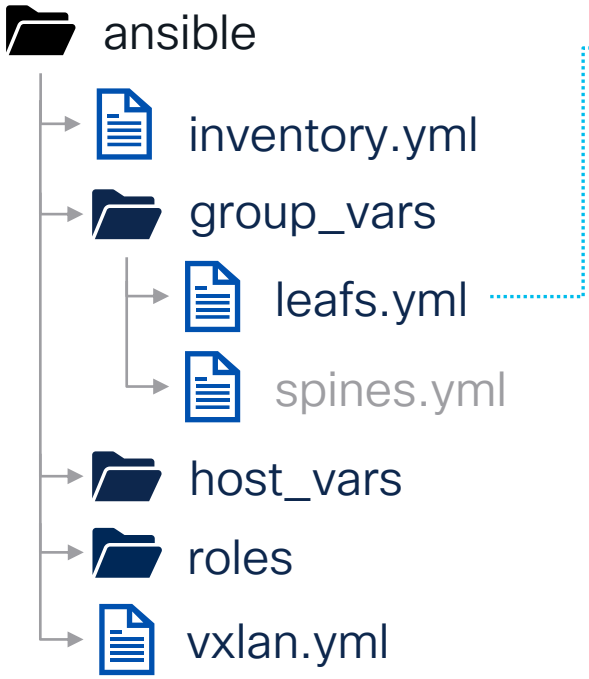
roles/overlay/tasks/main.yml

```
---
# 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 build Ansible task parameters this way!

Ansible Group Vars

Data to do it

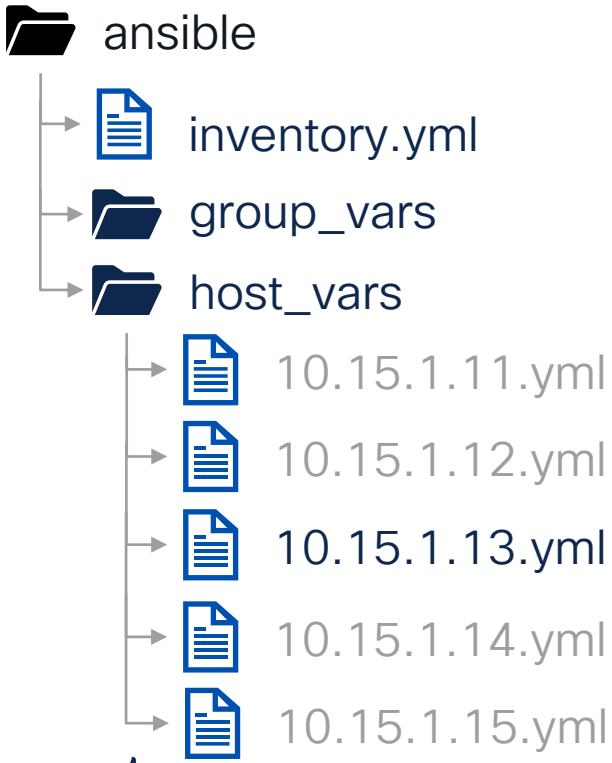



```
---
# var file for leafs group

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

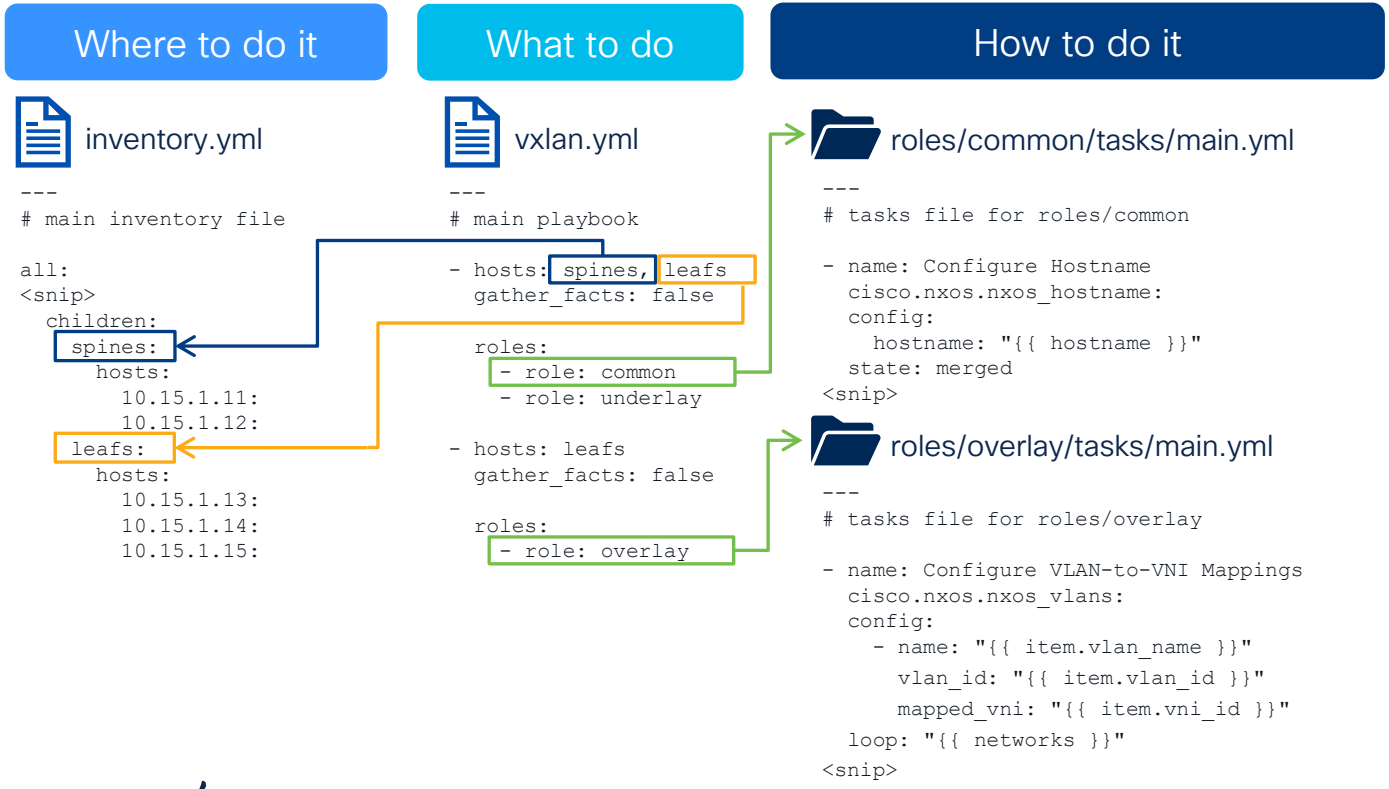
Data to do it




```
---
# 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
```

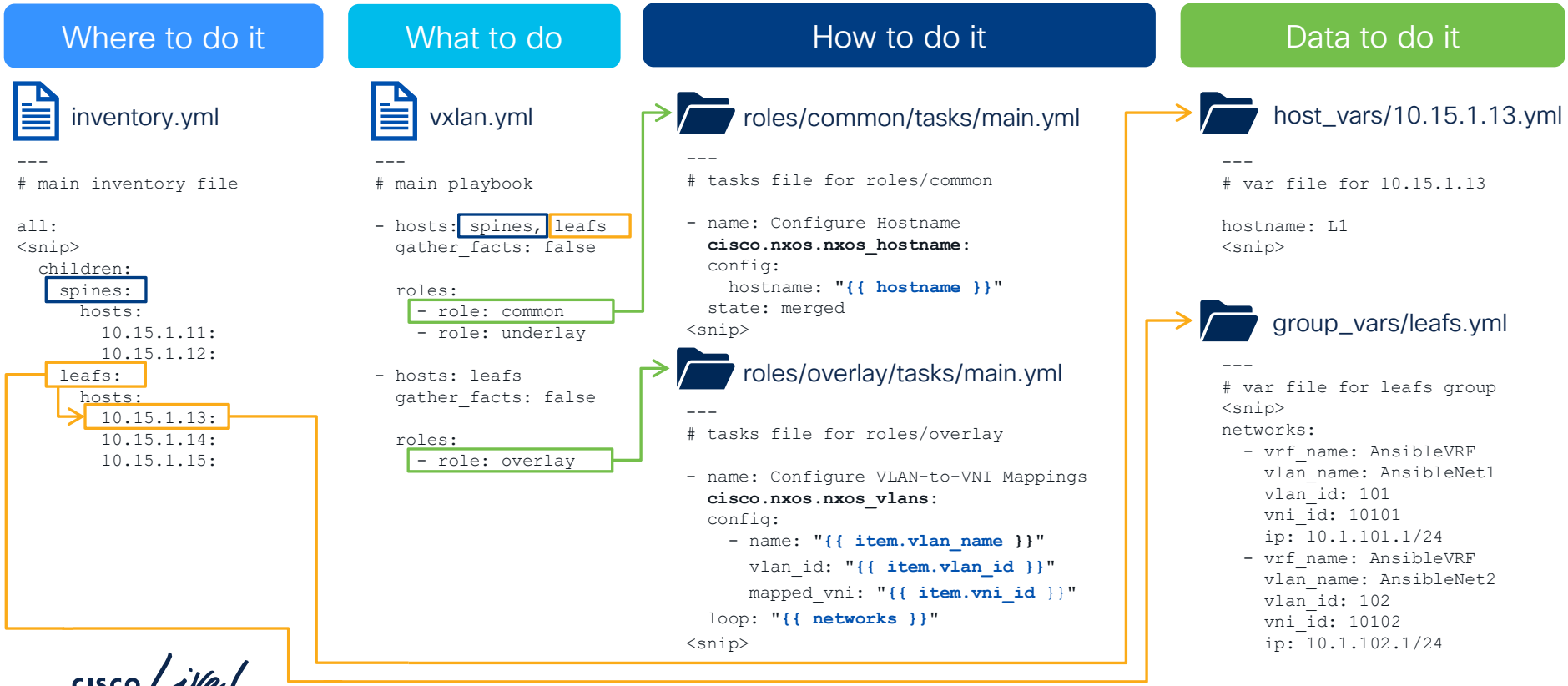
roles/common/tasks/main.yml

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

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

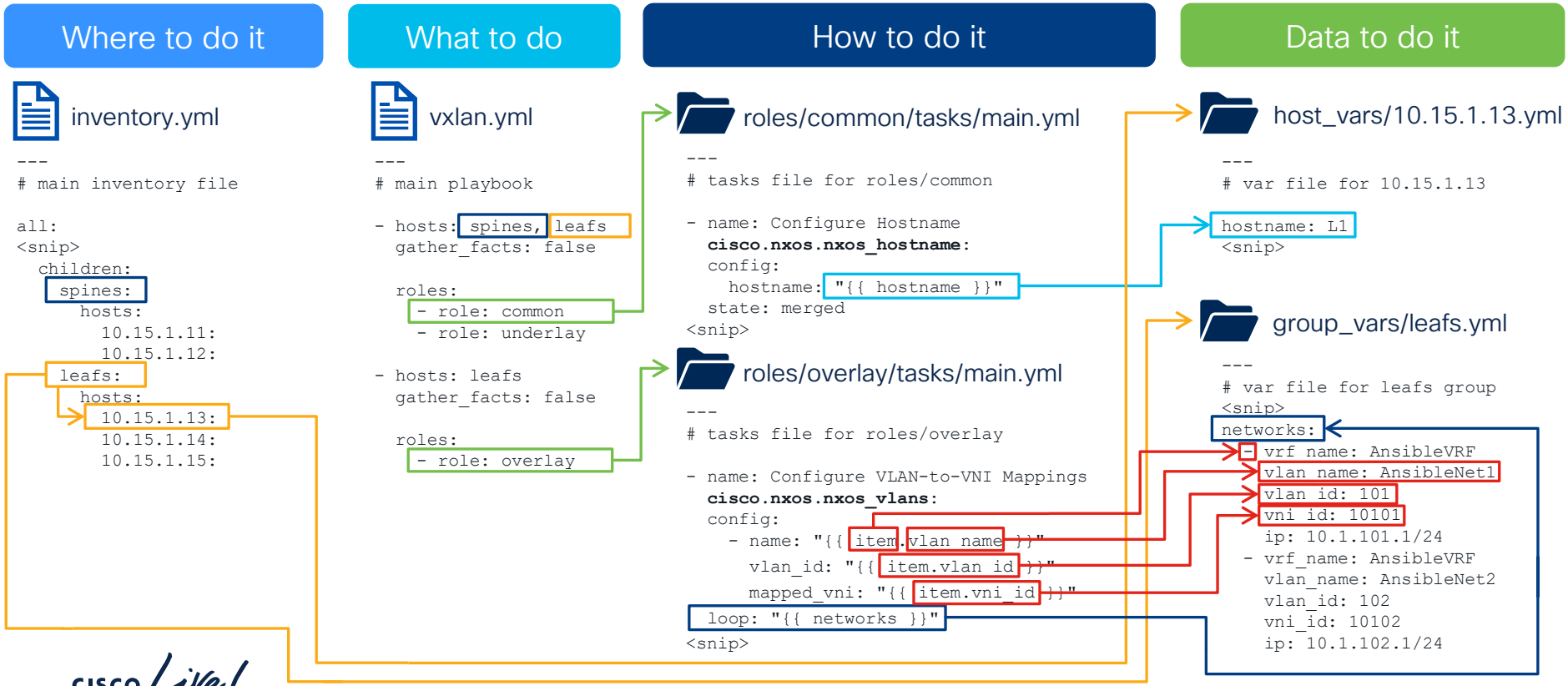
roles/overlay/tasks/main.yml

```
---
# 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>
```

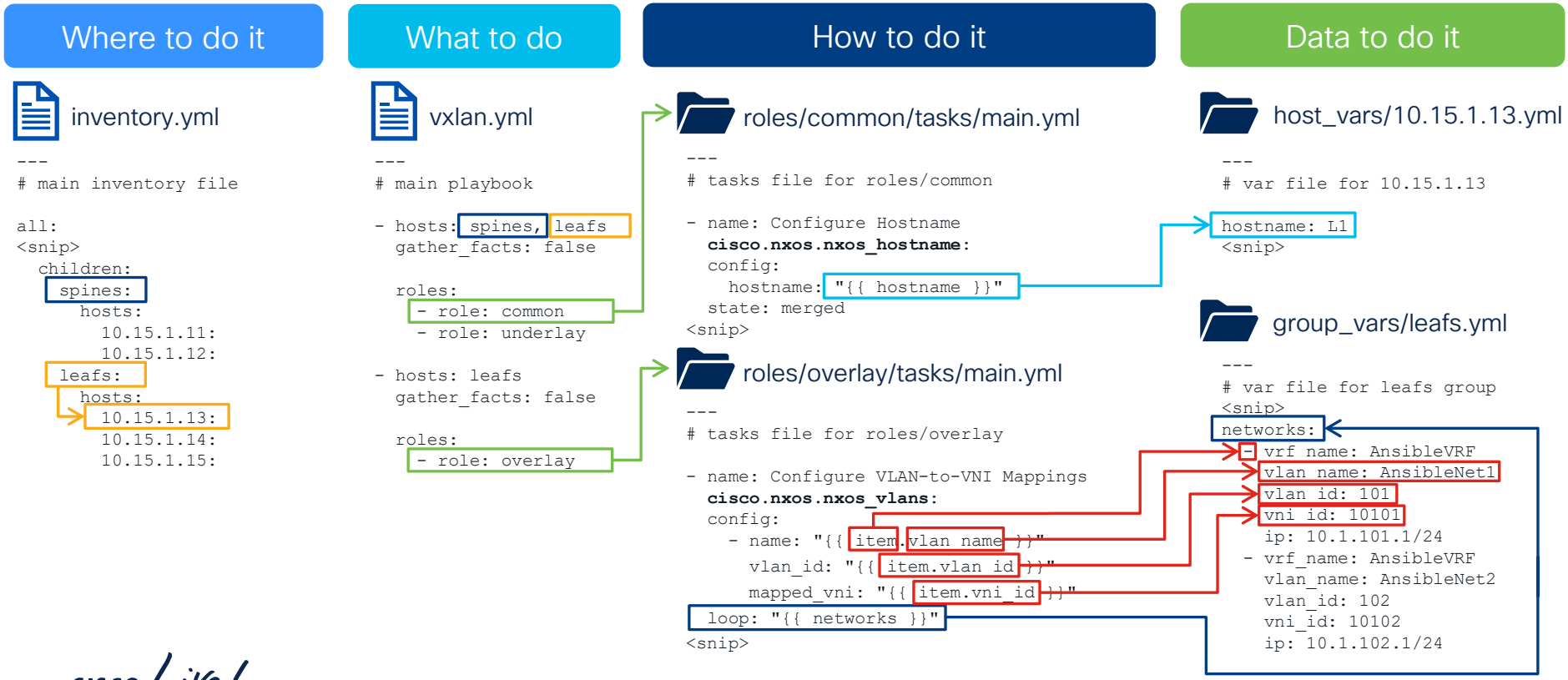
Putting It All Together



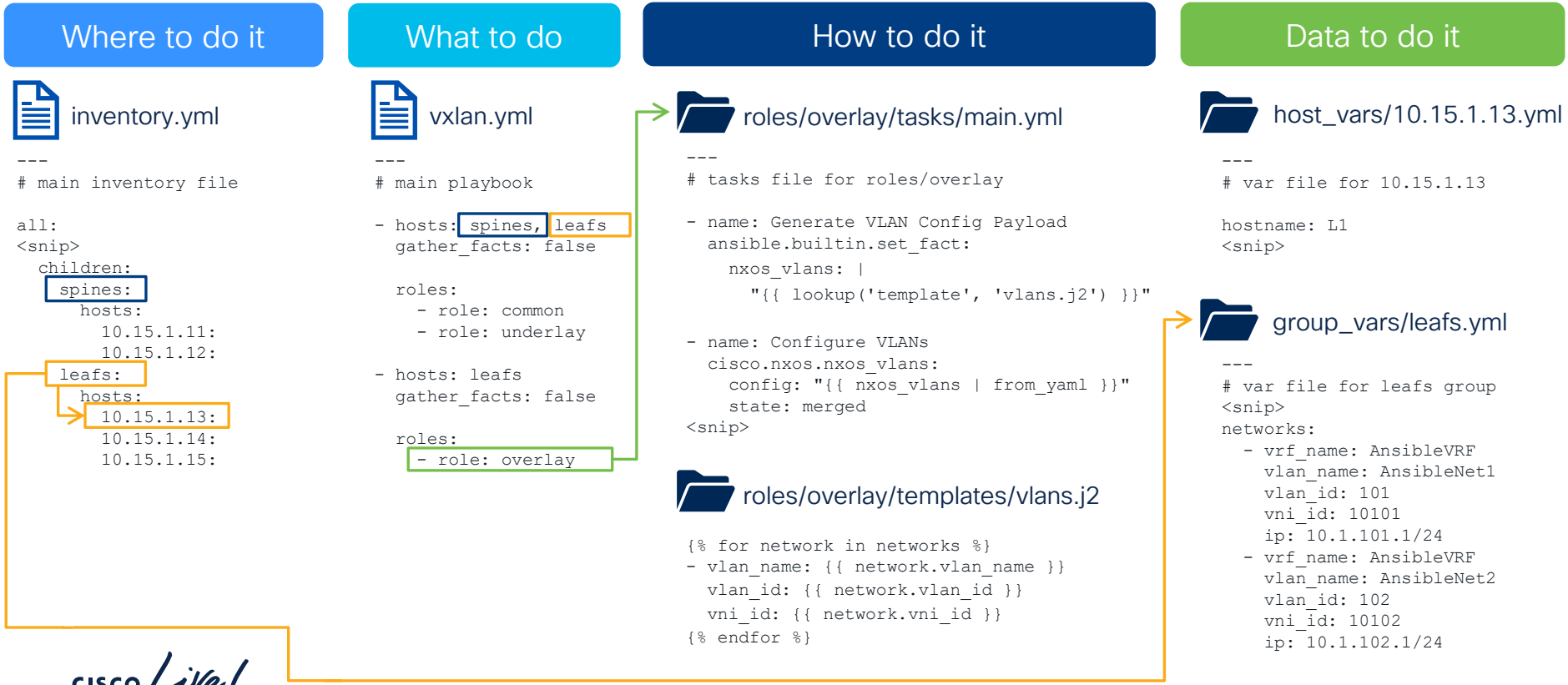
Putting It All Together



Putting It All Together



Putting It All Together



Putting It All Together

Where to do it

What to do

How to do it

Data 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/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 %}
```

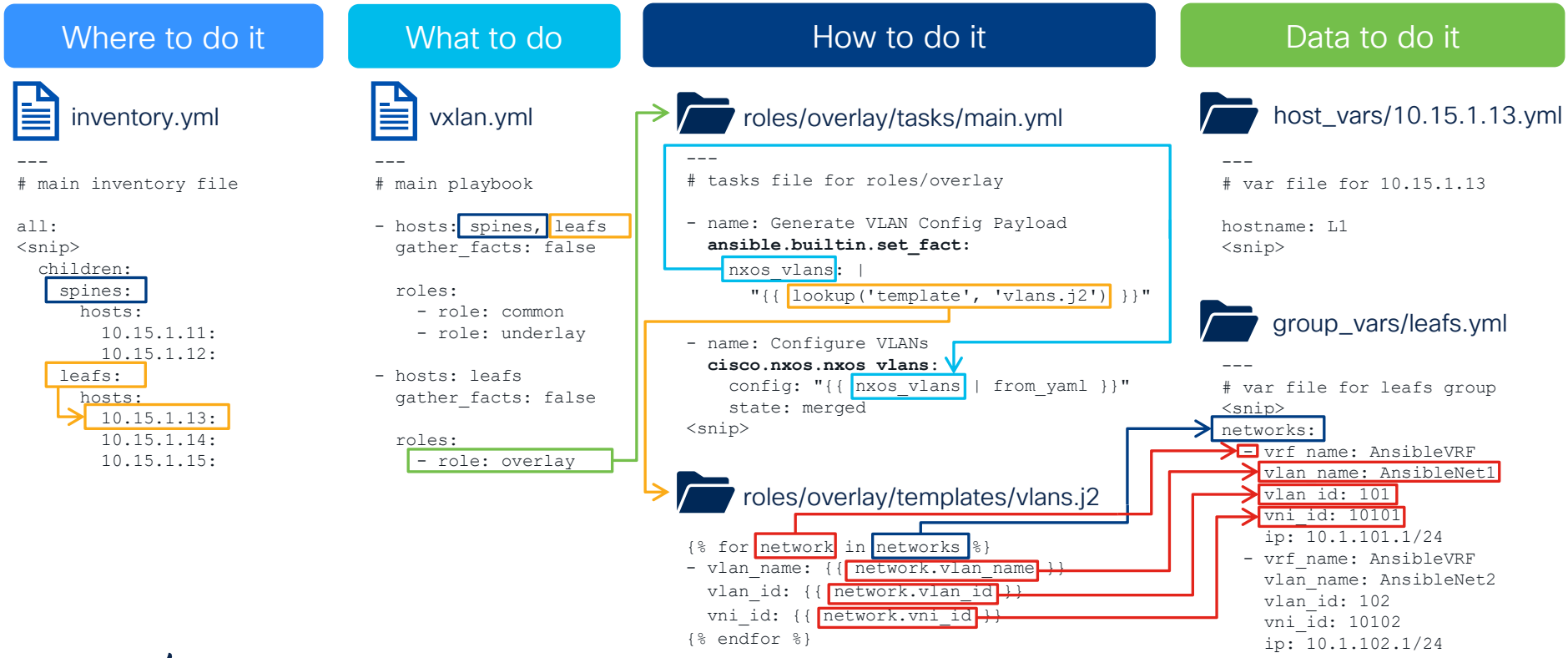
host_vars/10.15.1.13.yml

```
---
# var file for 10.15.1.13

hostname: L1
<snip>
```

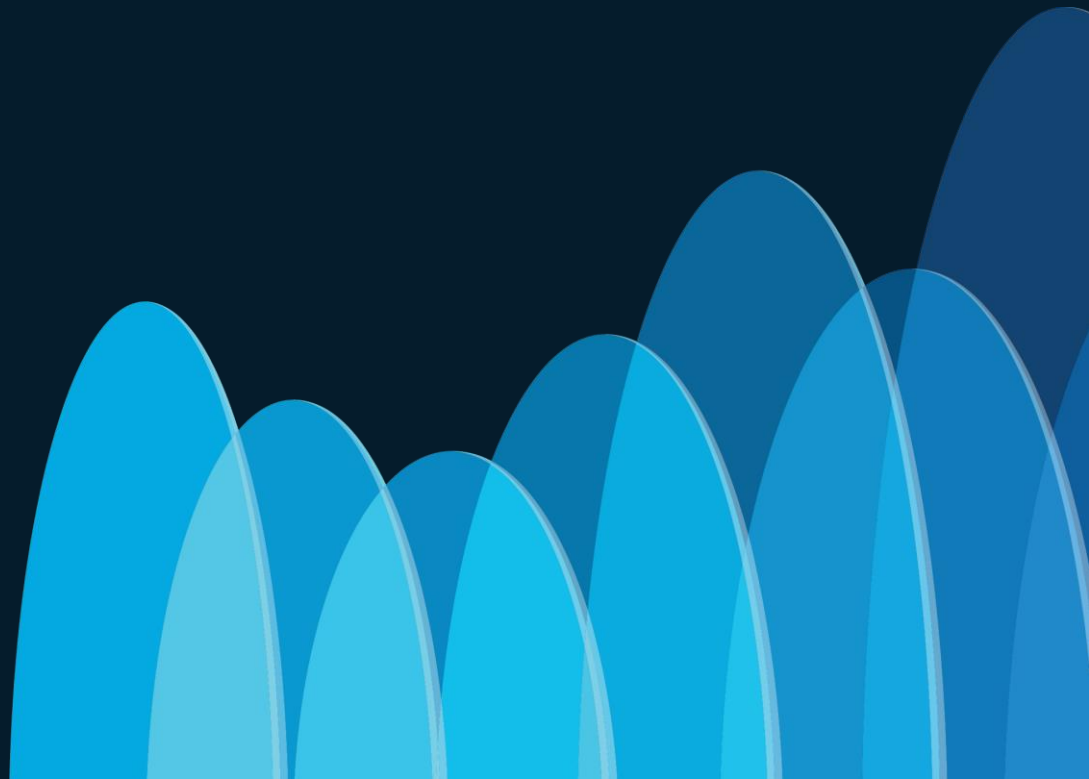
group_vars/leafs.yml

```
---
# 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



Additional Details for NXOS



cisco Live!

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

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

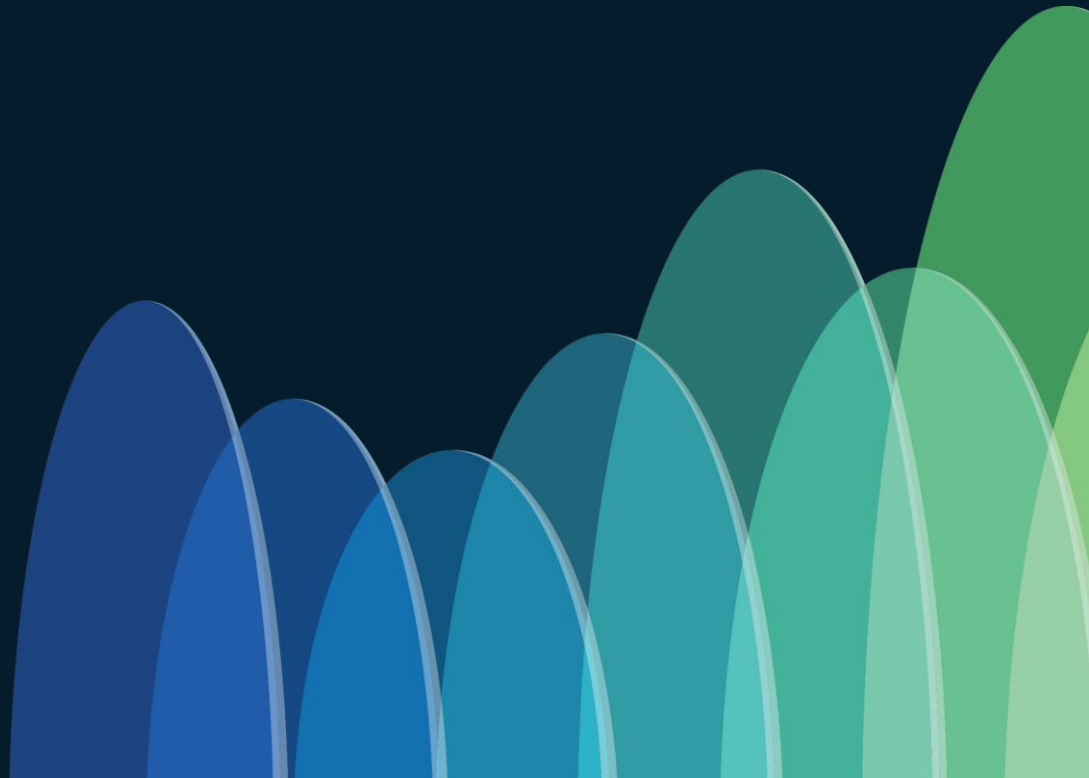
How to send commands to NXOS when a module is missing

- 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
```

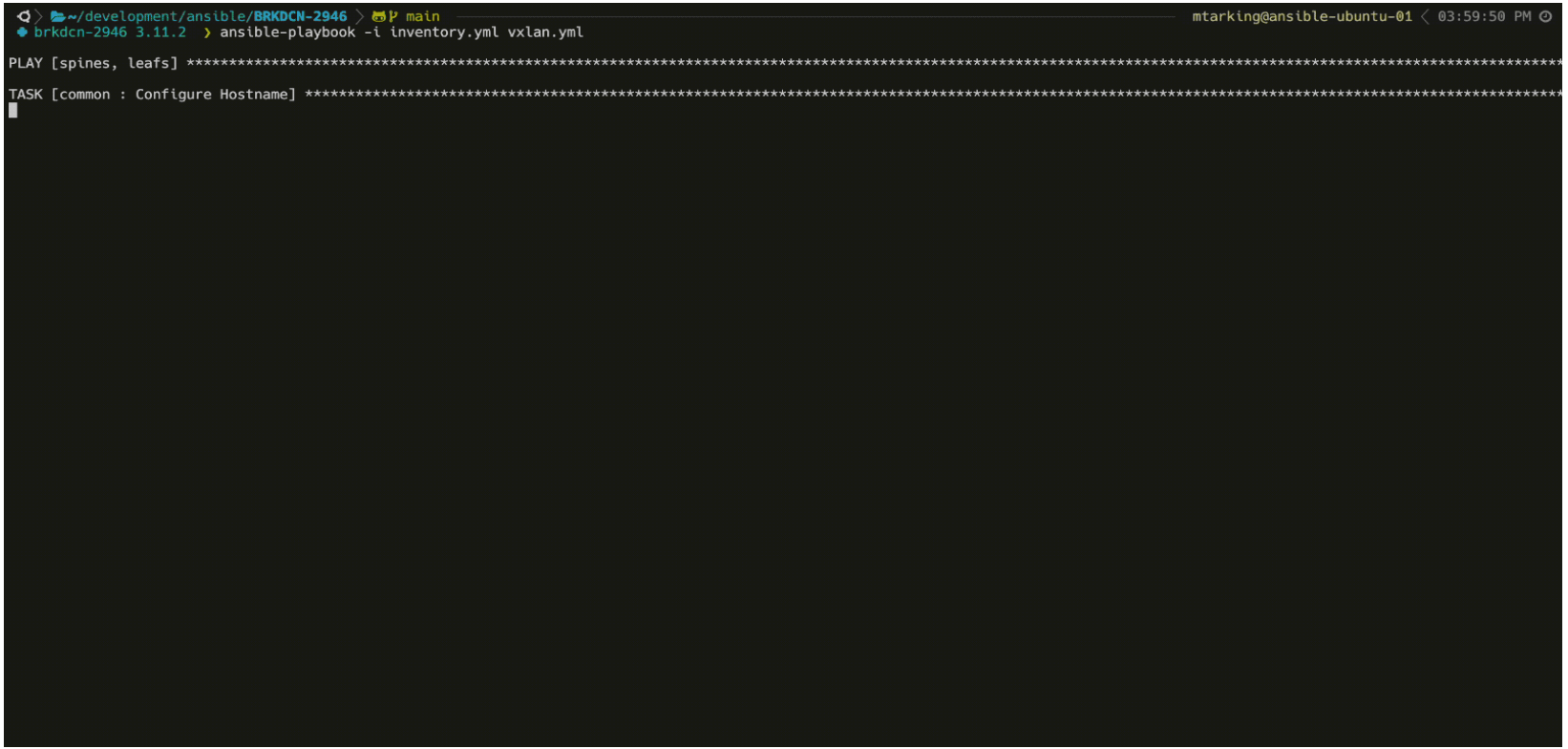
```
- name: Misc Commands
  cisco.nxos.nxos_command:
    commands: copy ftp://nxos.bin bootflash:
    prompt:
      - "Username:"
      - "Password:"
    answer:
      - <username>
      - <password>
```

An Example

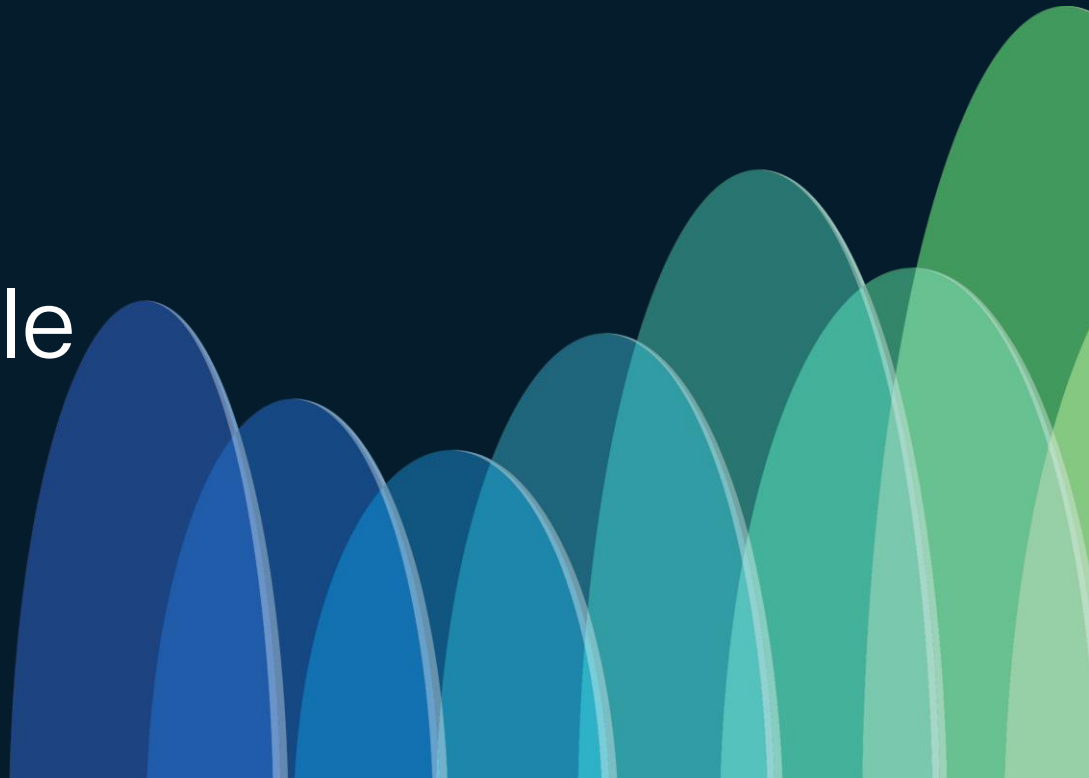


Executing Ansible Playbooks

```
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

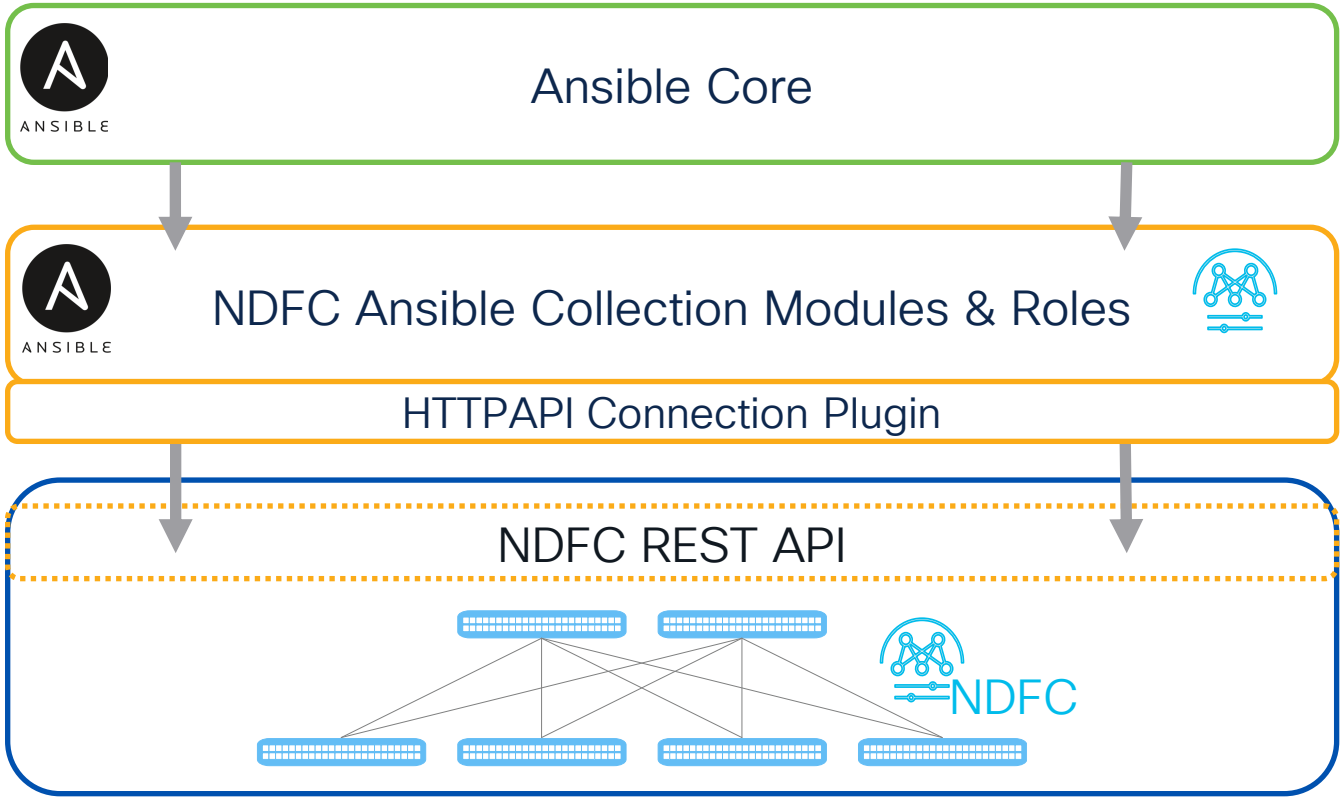
Easy to understand approach to auto-bootstrapping of entire fabric

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

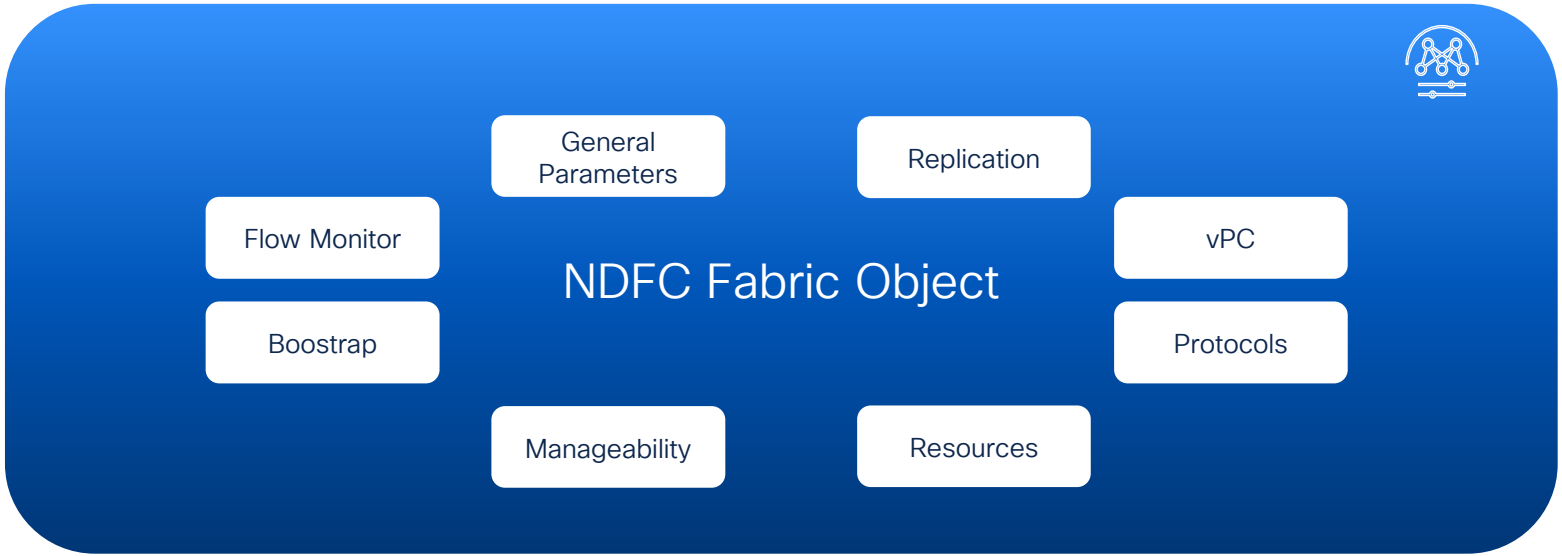
DevOps friendly

Enhanced Programmability

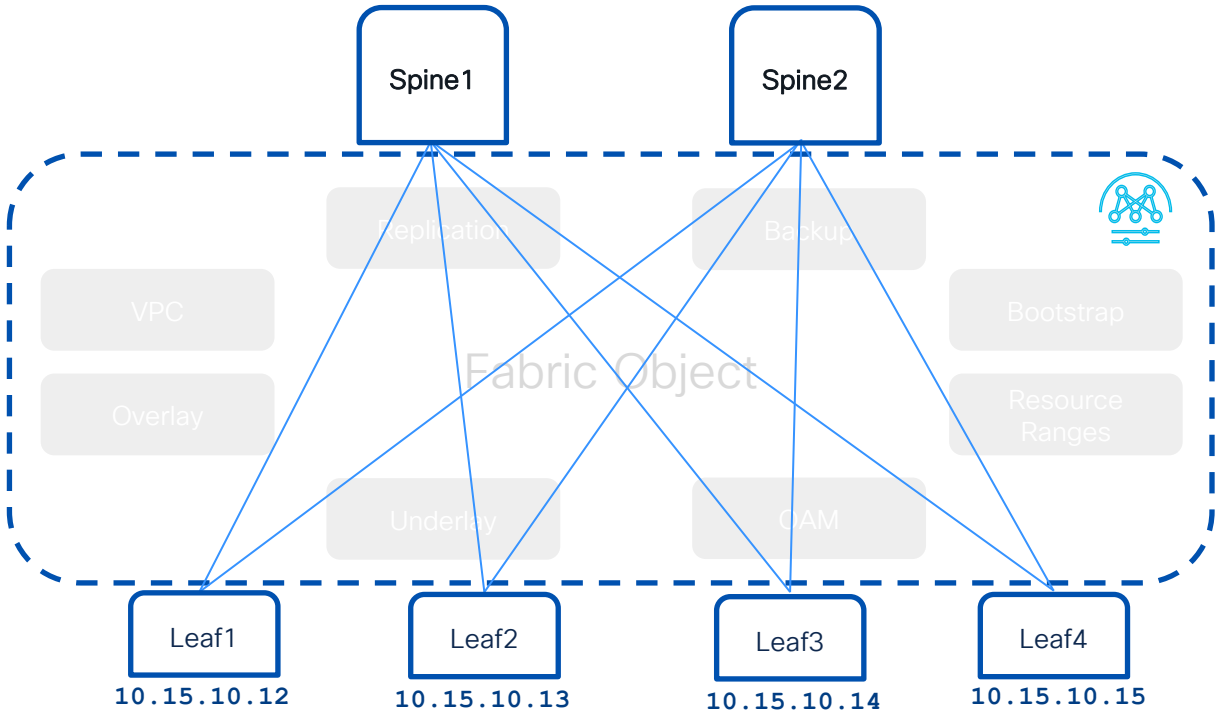
NDFC DevOps “Friendliness”



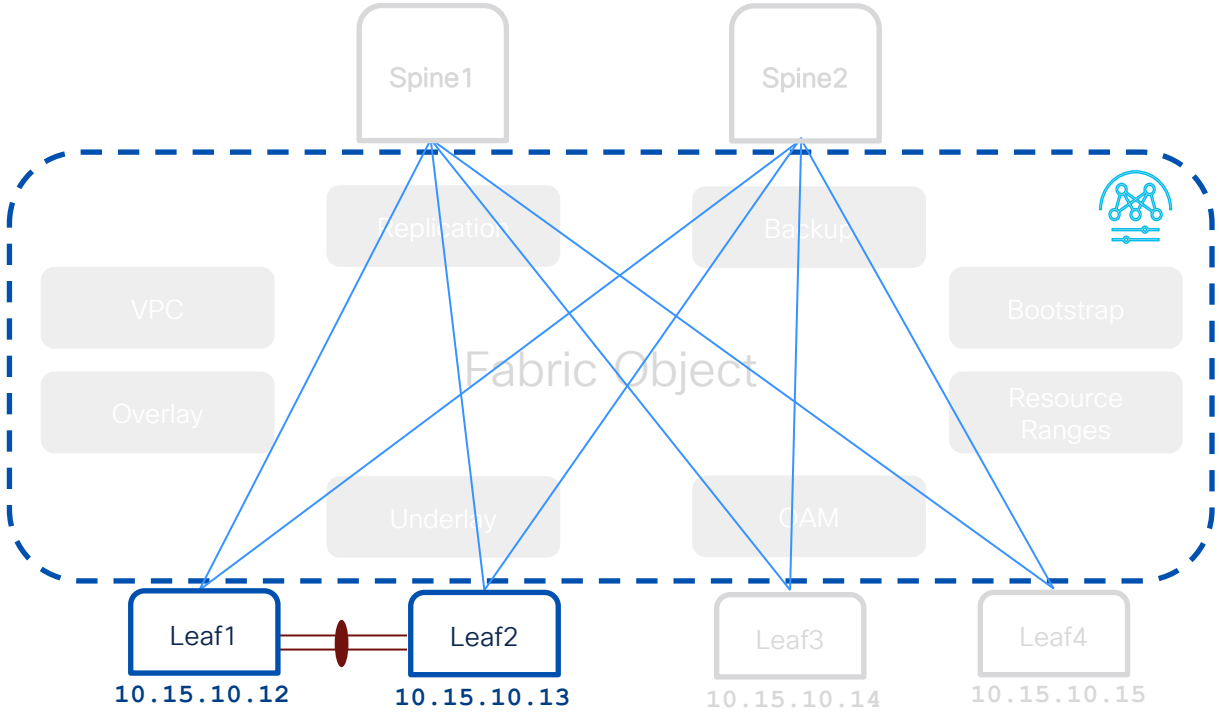
Workflow Step1: Fabric Management



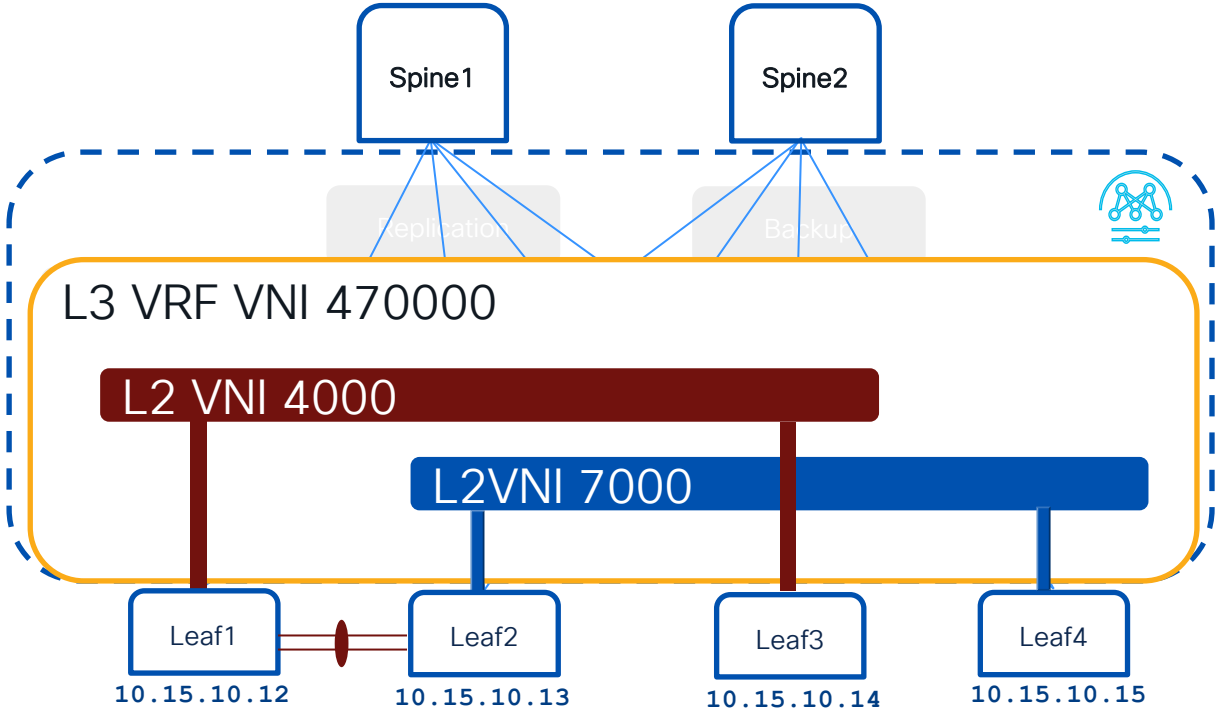
Workflow Step2: Inventory Management



Workflow Step3: vPC Pair Management



Workflow Step4: Overlay Management



NDFC Collection Modules – Toolbox

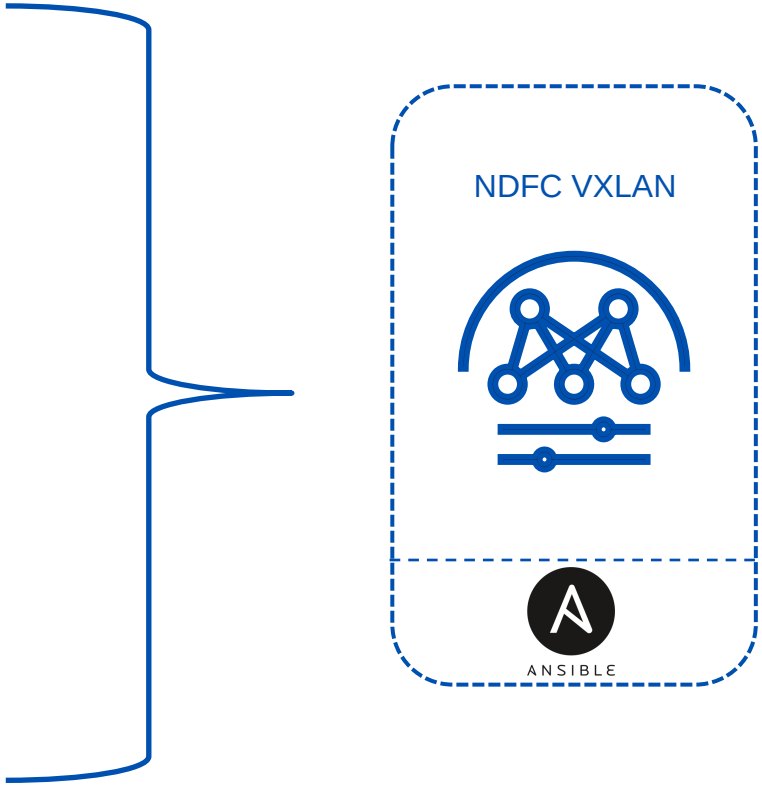
https://galaxy.ansible.com/cisco/dcnm

19 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 Nexus Switches
cisco.dcnm.dcnm_bootflash (Ver 3.6.0)	Bootflash management for Nexus Switches
cisco.dcnm.dcnm_maintenance_model (Ver 3.6.0)	Manage maintenance mode configuration of Nexus 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

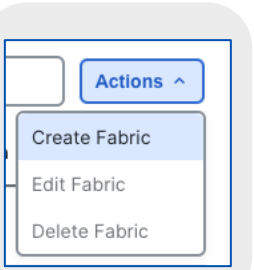
Primary VXLAN EVPN Fabric Plugins

1. cisco.dcnm.dcnm_fabric
2. cisco.dcnm.dcnm_inventory
3. cisco.dcnm.dcnm_vpc_pair
4. cisco.dcnm.dcnm_interface
5. cisco.dcnm.dcnm_vrf
6. cisco.dcnm.dcnm_network
7. cisco.dcnm.dcnm_policy

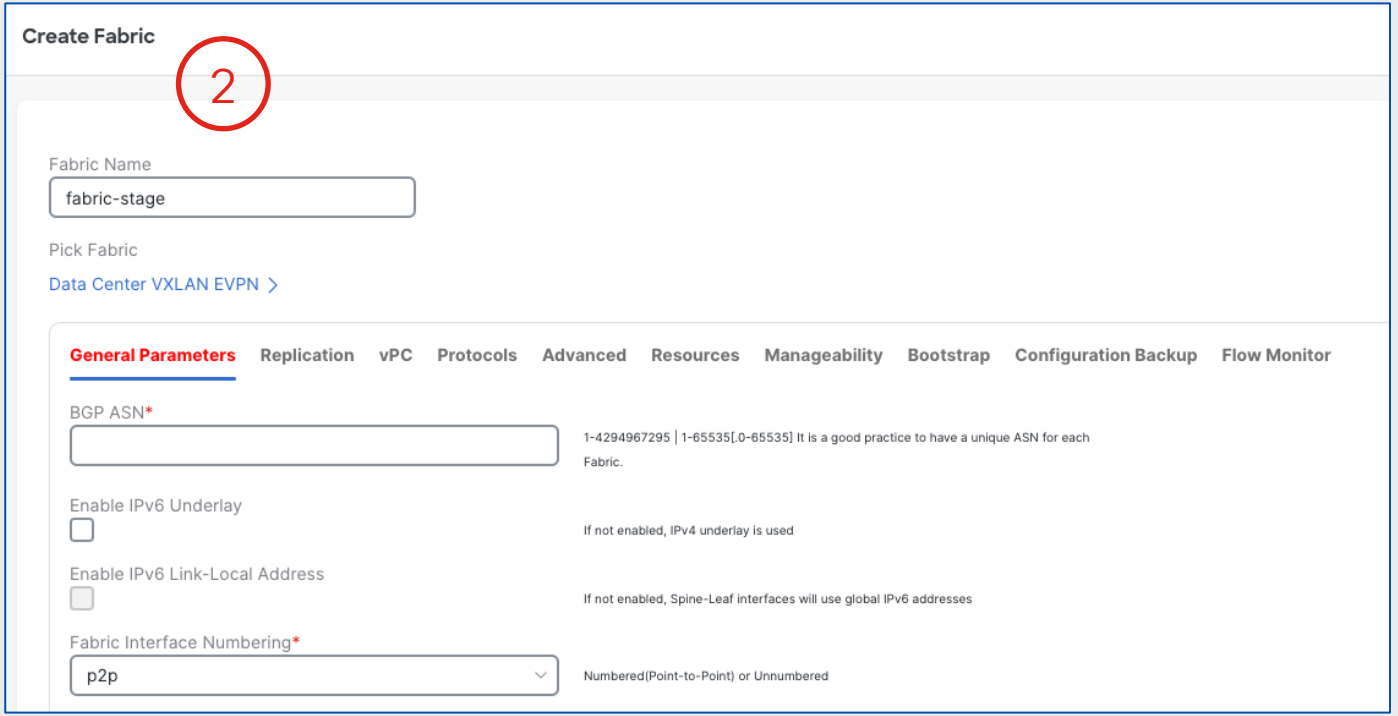


Let’s Build Something With NDFC and Ansible Together!

Workflow Step1: Create VXLAN Fabric

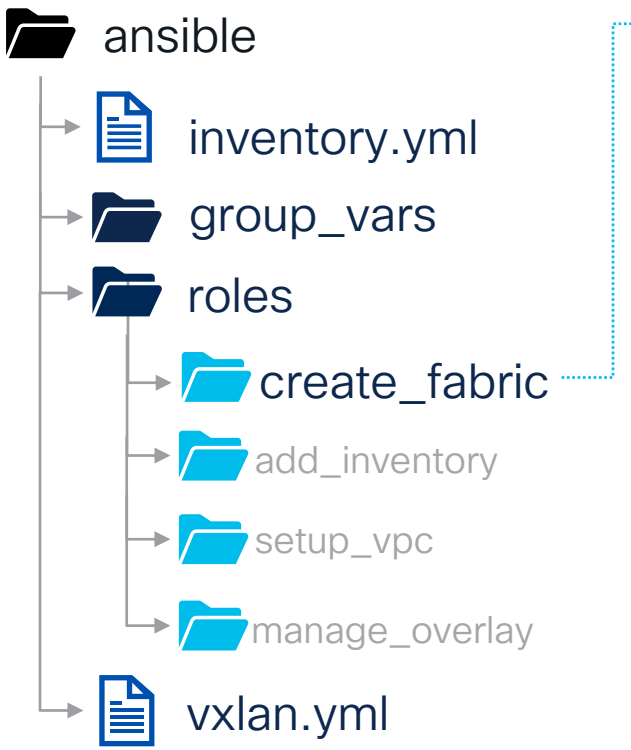


1



Ansible Roles – Fabric Tasks

How to do it



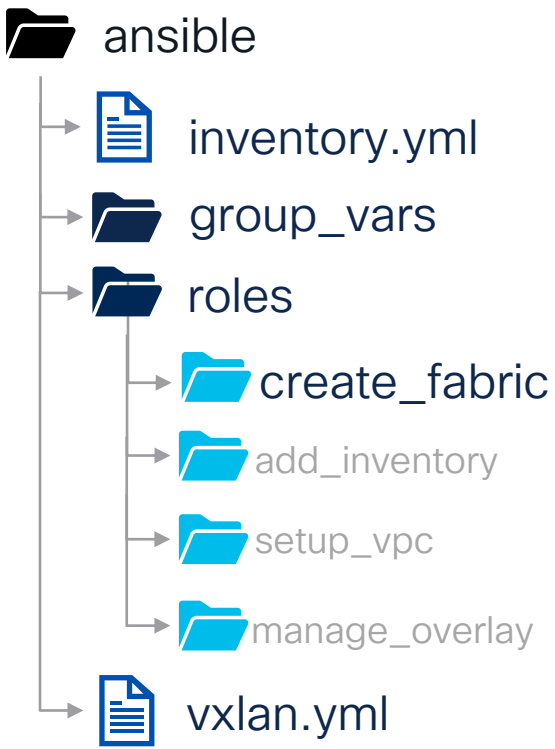
roles/create_fabric/tasks/main.yml

```
---
# 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
```

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

Ansible Roles – Fabric Tasks

How to do it



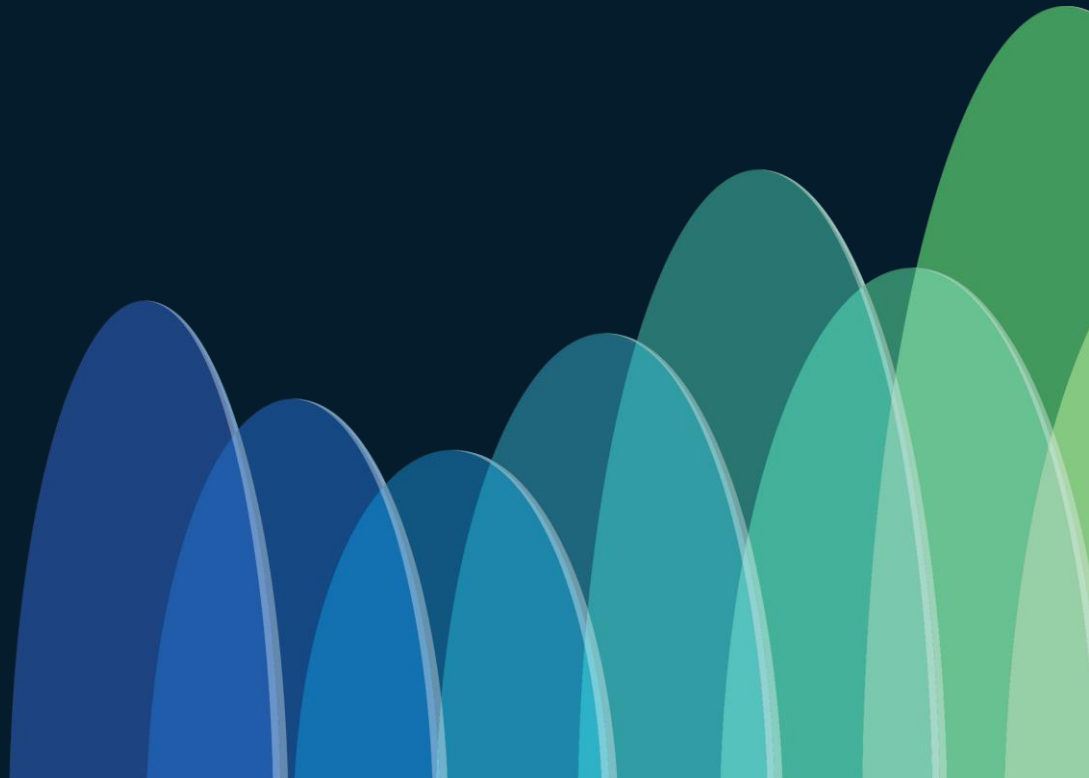
roles/create_fabric/tasks/main.yml

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


- name: Create Fabric with Module
  cisco.dcn
    state
    conf
      -
```

Let's discuss a new and better way!

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

VXLAN as Code



VXLAN as Code

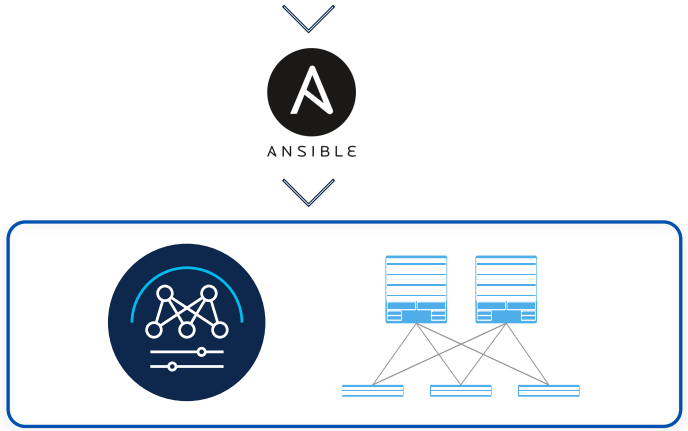
https://netascode.cisco.com

VXLAN as Code aims to reduce time to value by lowering the barrier of entry to network orchestration through simplification, abstraction, and curated examples.

It allows users to instantiate network fabrics in minutes using an easy to use, opinionated data model. It takes away the complexity of having to deal with references, dependencies or loops.

Users can focus on describing the intended configuration while using a set of maintained and tested Ansible Modules without the need to understand the low-level network device configuration.

```
vxlan:
  overlay:
    networks:
      - name: NetworkDMZ
        is_l2_only: False
        vrf_name: VrfRed
        net_id: 33100
        vlan_id: 3100
         network_attach_group: dmz
```



Data Model

Built by **network engineers** for each Cisco feature implemented

Incorporates best practices with BE/CX default values

Built to allow us to represent network complexity in an easier structure where we can incorporate validation and testing

We use Yamale* to construct schema and insert rule enhancements

```
vxlan:
  underlay:
    general:
      routing_protocol: ospf
      enable_ipv6_underlay: false
      replication_mode: multicast
      fabric_interface_numbering: p2p
      subnet_mask: 30
      underlay_routing_loopback_id: 0
      underlay_vtep_loopback_id: 1
      underlay_routing_protocol_tag: UNDERLAY
      underlay_rp_loopback_id: 254
      intra_fabric_interface_mtu: 9216
      layer2_host_interfacde_mtu: 9216
      unshut_host_interfaces: True

    bgp:
      authentication_enable: true
      authentication_key_type: 3
      authentication_key: 014903c077689077e5b4548432c3d91d
```

23 and Me
Open-source project https://github.com/23andMe/Yamale

Default Values

VXLAN as Code comes with pre-defined default values based on common best practices.

In some cases, those default values might not be the best choice for a particular deployment and can be overwritten if needed.

```
defaults:
  vxlan:
    overlay:
      vrfs:
        loopback_route_tag: 12345
        vrf_int_mtu: 9216
```

```
vxlan:
  overlay:
    vrfs:
      - name: CiscoLive
      - name: BlueNet
```

Avoid configuration errors with...
Semantic Data Validation

Verify every change with...
Context-Driven, Automated Testing

CI/CD infrastructure Automation Infrastructure
(Common Automation Framework)

Experts for the network technology and how to automate operations
End to End Support from Cisco

Effectively communicate changes with...
Email & ChatOps Integrations

Integrate into YOUR CI/CD environment using the...
Platform Agnostic CI/CD Modules

Open-source
do it yourself

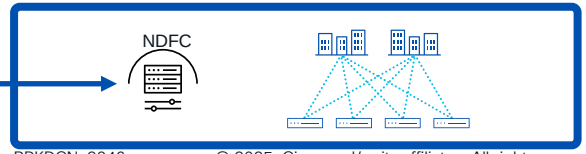
\+ Incremental **Value** *of* **CX**
over do it yourself

No TAC Support

YAML **Data Model** YAML

NDFC/VXLAN as Code

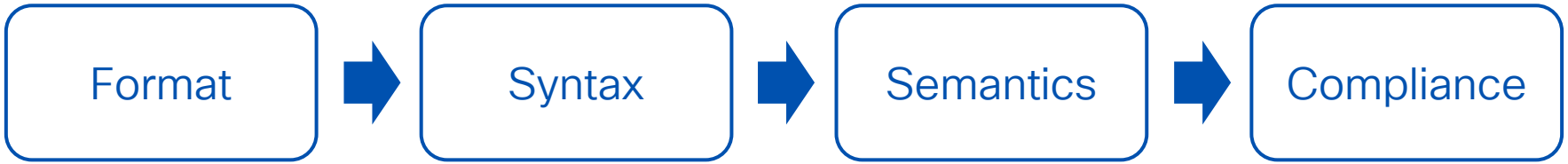
Ansible DCNM Modules



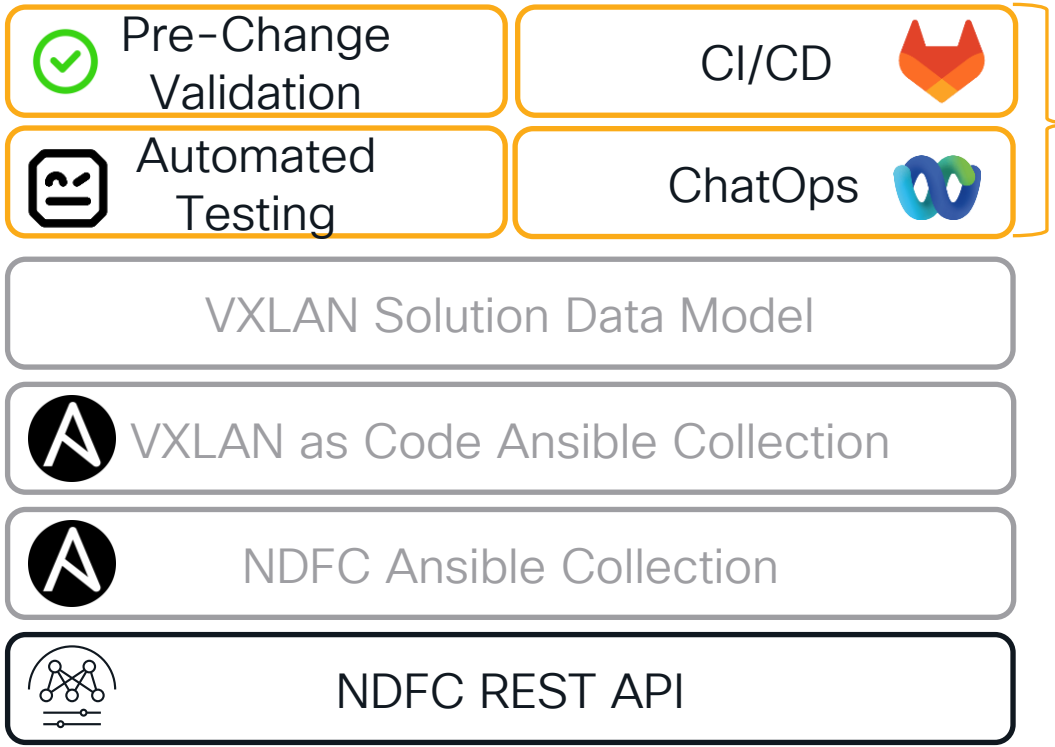
Pre-Change Validation

As the complexity of the configuration and the underlying data model increases automated validation before deploying anything in a production environment becomes a critical aspect.

Several tools can be used to ensure that the provided input data is valid, but also that common best practices and formatting guidelines are being followed.



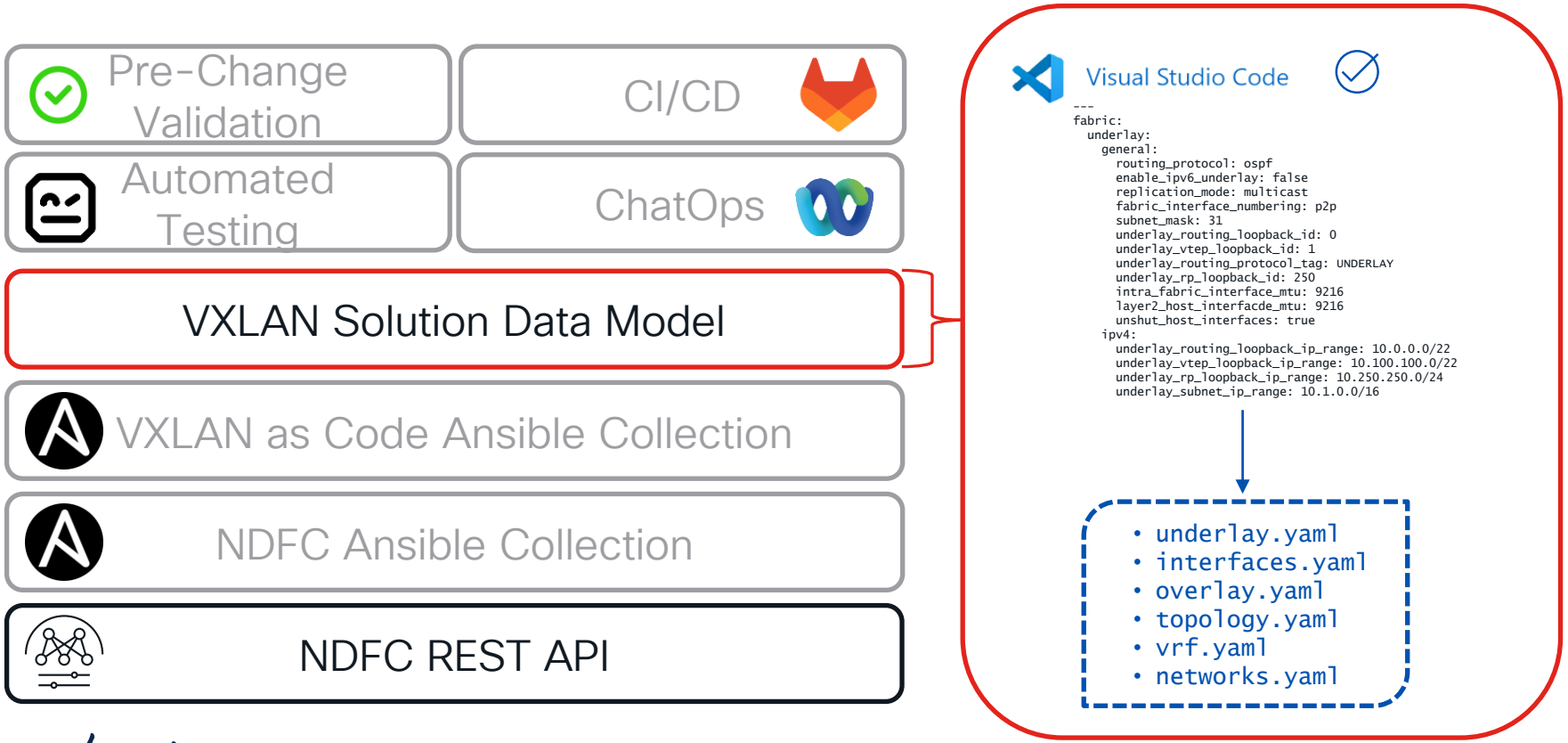
VXLAN as Code Building Blocks



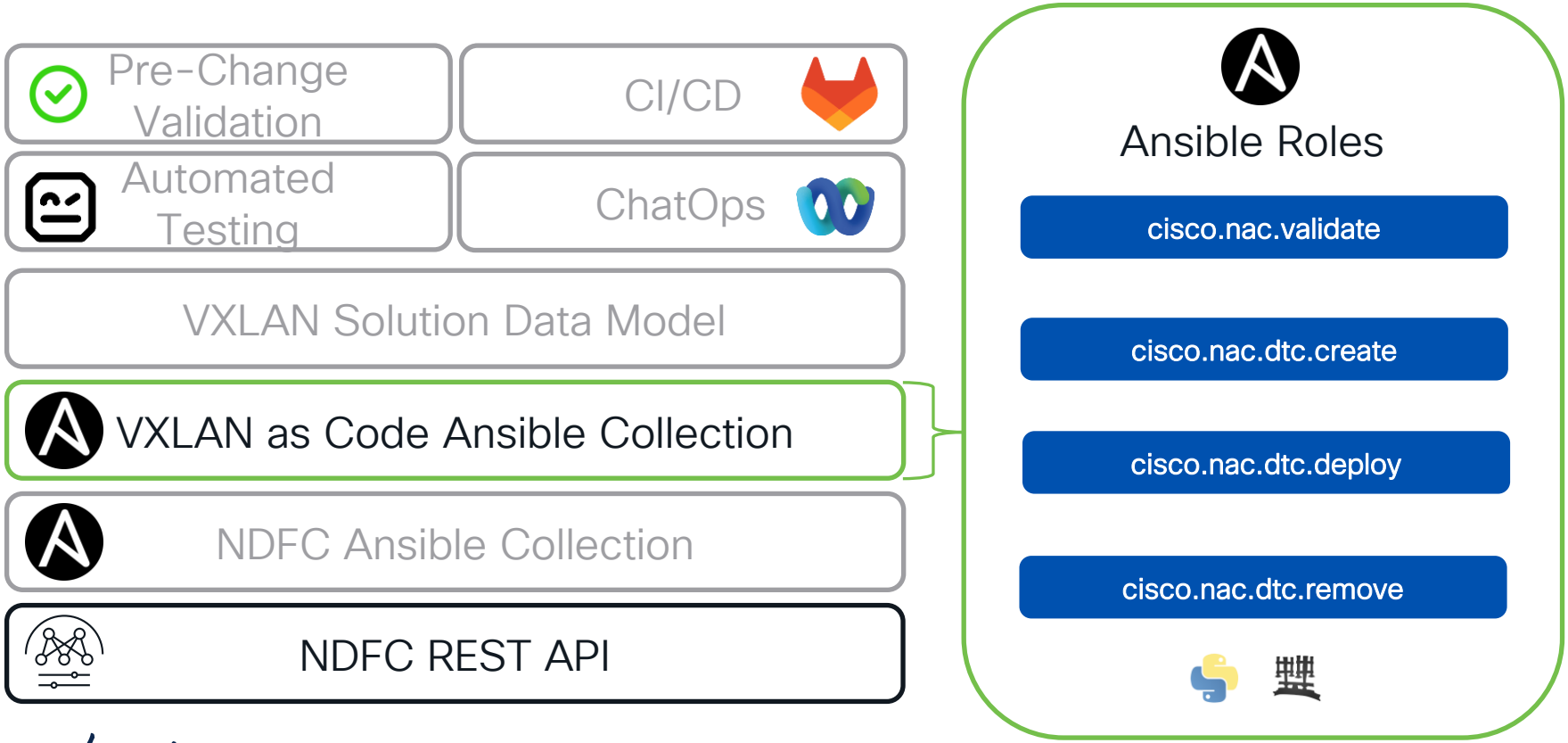
Automation Objectives

- Input and Rules Validation
- Testing
- Pipeline Integration
- ChatOps Integration

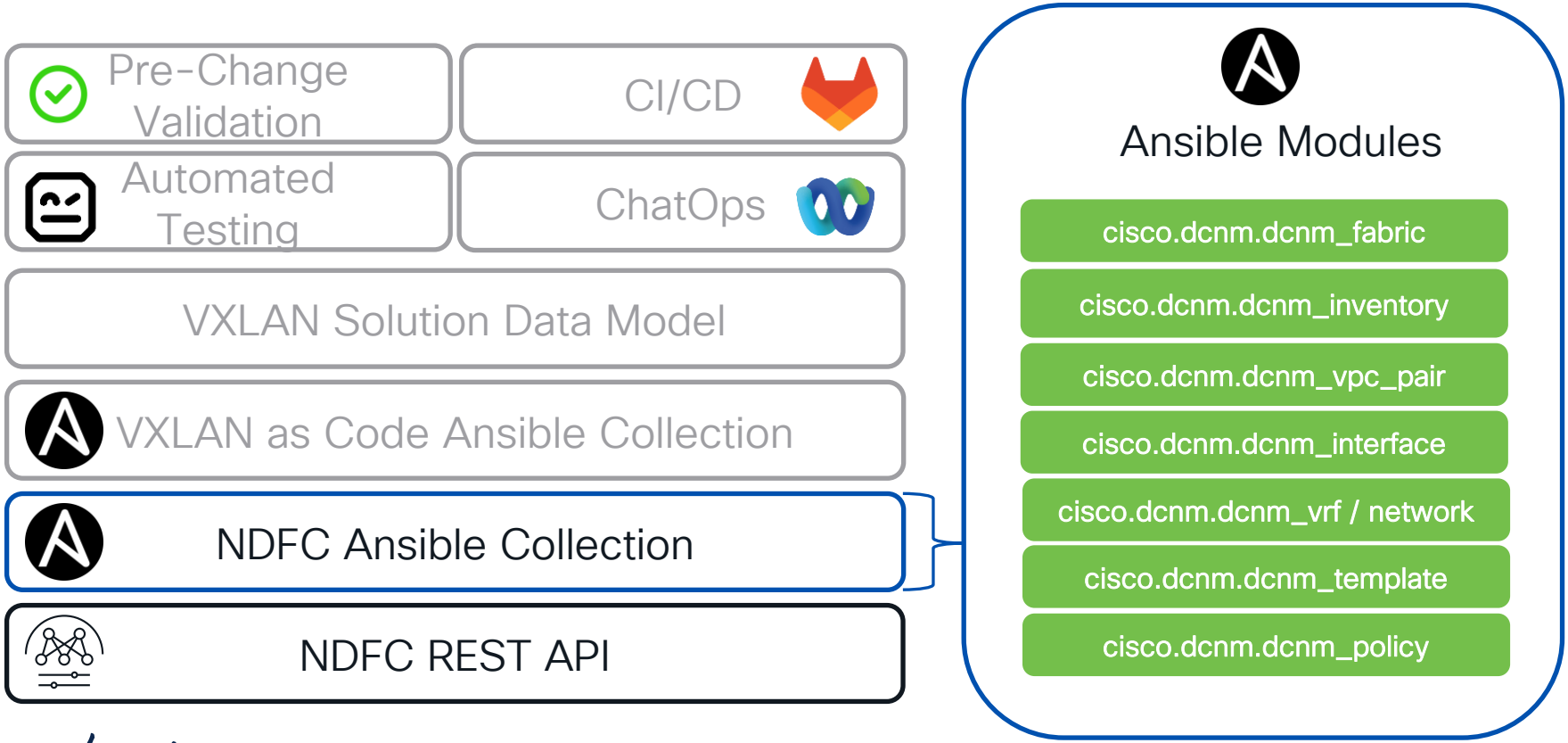
VXLAN as Code Building Blocks



VXLAN as Code Building Blocks



VXLAN as Code Building Blocks



Ansible Playbook

What to do


```
---
# main NaC playbook

- name: Build VXLAN EVPN Fabric on NDFC
  hosts: ndfc_fabric1, ndfc_fabric2
  gather_facts: false

  roles:
    - role: cisco.nac_dc_vxlan.validate

    - role: cisco.nac_dc_vxlan.dtc.create
      tags: 'role_create'

    - role: cisco.nac_dc_vxlan.dtc.deploy
      tags: 'role_deploy'

    - role: cisco.nac_dc_vxlan.dtc.remove
      tags: 'role_remove'
```

Ansible Inventory

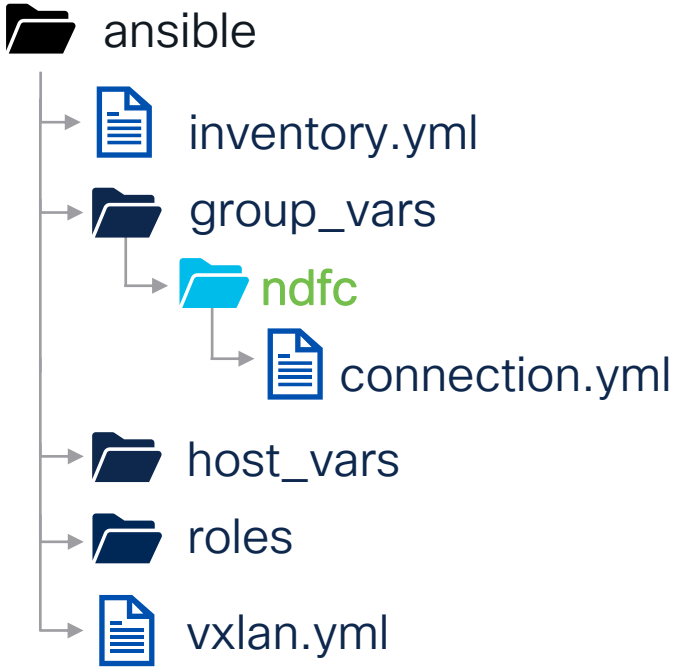
Where to do it

- ansible
  - inventory.yml
  - group_vars
  - host_vars
  - roles
  - vxlan.yml

```
---
# main inventory file

all:
  children:
    ndfc:
      hosts:
        ndfc_fabric1:
          ansible_host 10.15.0.20
        ndfc_fabric2:
          ansible_host 10.15.0.20
```

Ansible Group Vars

Where to do it

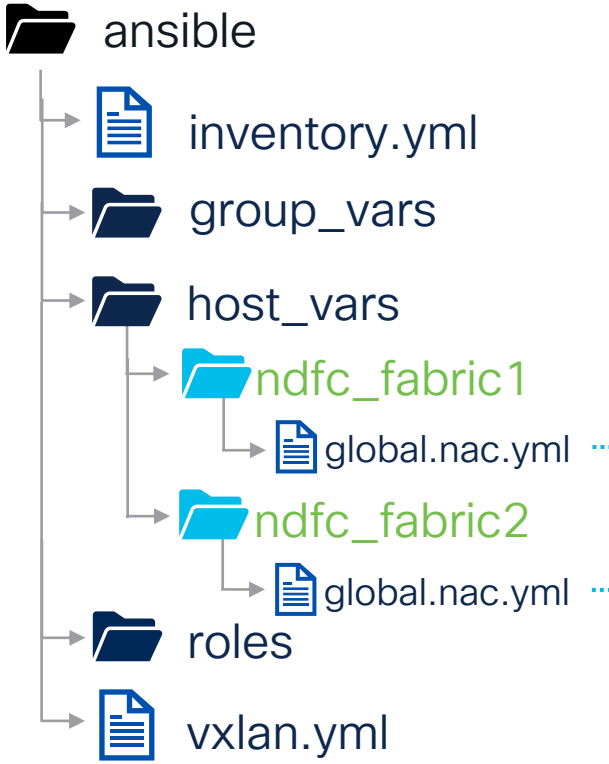



```
---
# group_vars ndfc connection information

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

ansible_user: "{{ lookup('env', 'ND_USERNAME') }}"
ansible_password: {{ lookup('env', 'ND_PASSWORD') }}

ndfc_switch_username: "{{ lookup('env', 'NDFC_SW_USERNAME') }}"
ndfc_switch_password: "{{ lookup('env', 'NDFC_SW_PASSWORD') }}"
```

Connection information for NDFC

Ansible Host Vars

Data to do it



Visual Studio Code — global.nac.yml

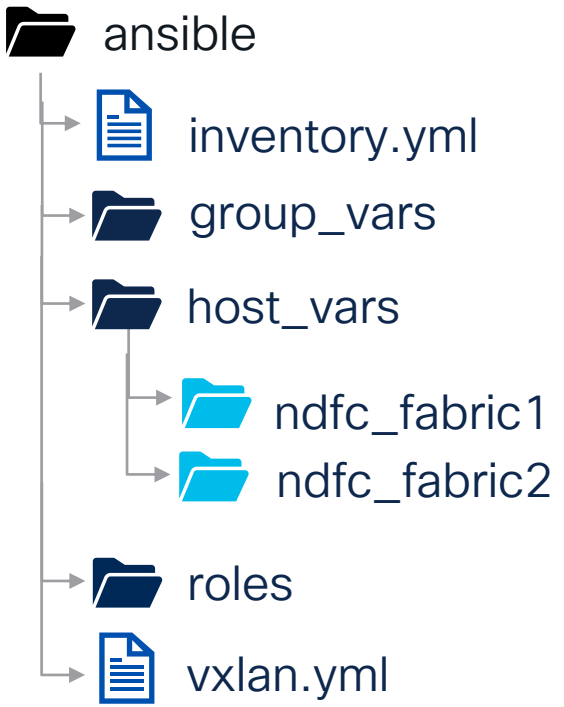
```
---
vxlan:
  global:
    name: ndfc_fabric1
    bgp_asn: 65055
    route_reflectors: 4
    anycast_gateway_mac: 20:20:00:00:00:aa
```

Visual Studio Code — global.nac.yml

```
---
vxlan:
  global:
    name: ndfc_fabric2
    bgp_asn: 65088
    route_reflectors: 2
    anycast_gateway_mac: 20:20:00:00:00:bb
```

Ansible Host Vars

Data to do it



Visual Studio Code — global.nac.yml

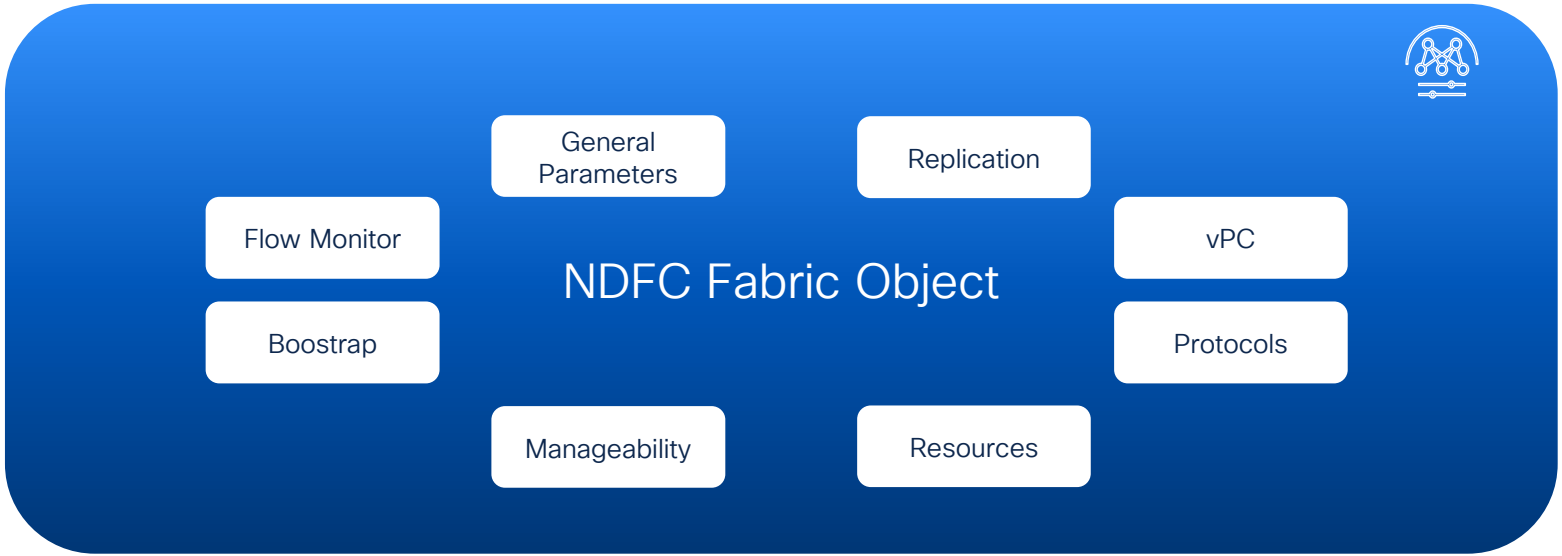
```yaml
---
vxlan:
  global:
    name: ndfc_fabric1
    bgp_asn: 65055
    route_reflectors: 4
    anycast_gateway_mac: 20:20:00:00:00:aa
```

Service Model Data

https://netascode.cisco.com/data_model/vxlan/overview

- global.nac.yml
- underlay.nac.yml
- interfaces.nac.yml
- overlay.nac.yml
- topology.nac.yml
- vrf.nac.yml
- networks.nac.yml
- policy.nac.yml

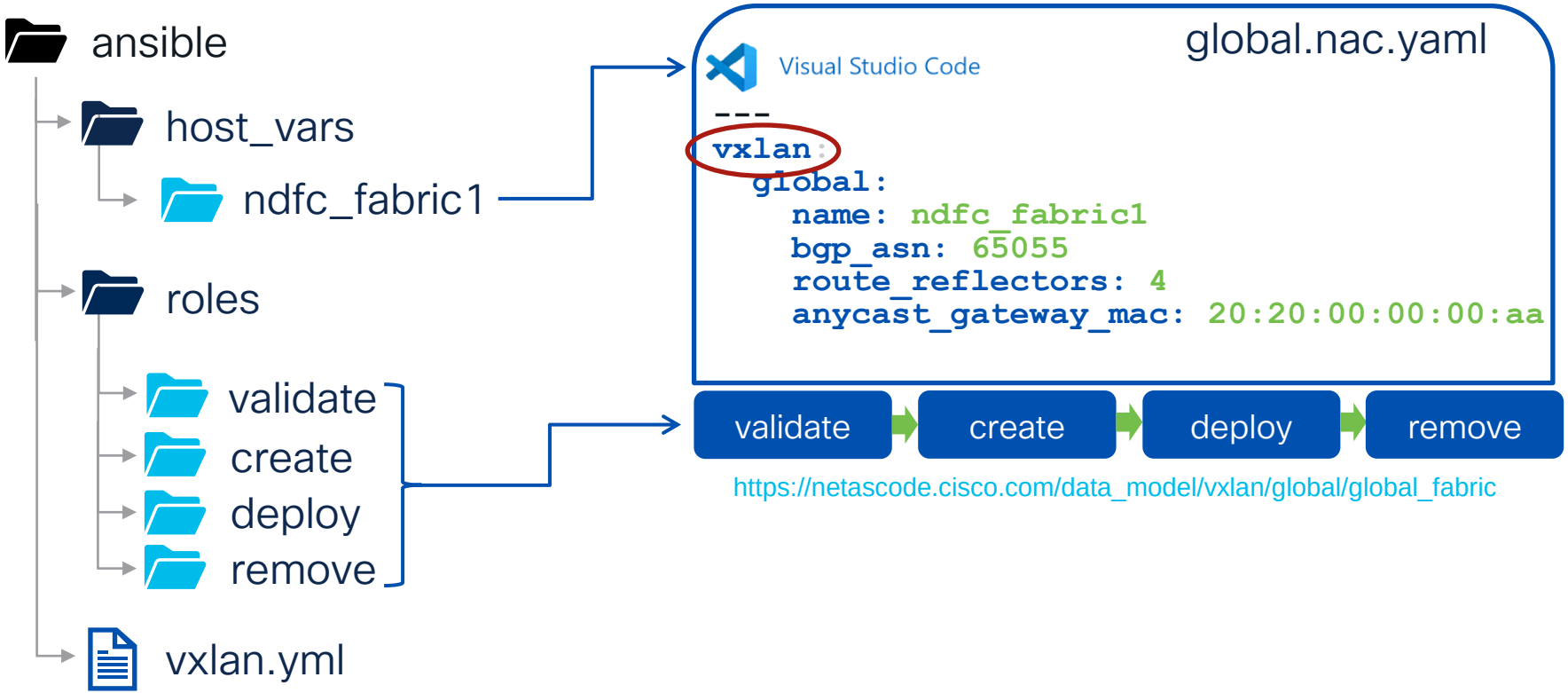
Workflow Step1: Fabric Management



Model – Fabric Management

Data to do it

How to do it



Model – Fabric Management

Data to do it

How to do it

global.nac.yaml

Visual Studio Code

```
---
vxlan:
  global:
    name: ndfc_fabric1
    bgp_asn: 65055
    route_reflectors: 4
    anycast_gateway_mac: 20:20:00:00:00:aa
```

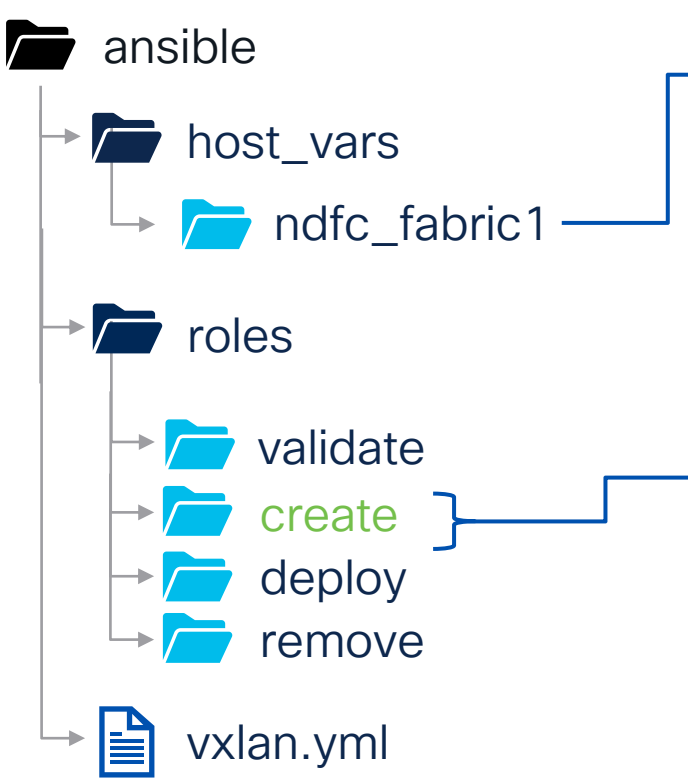
validate → create → deploy → remove

- ✓ Validate vxlan.global data against model schema
  - ✓ Semantic Validation – model structure
  - ✓ Input Validation – anycast_gateway_mac valid?
- ✓ Process base and custom rules
  - ✓ Example: spanning tree vlan and mst ranges

Model – Fabric Management

Data to do it

How to do it



global.nac.yaml

Visual Studio Code

```
---
vxlan:
  global:
    name: ndfc_fabric1
    bgp_asn: 65055
    route_reflectors: 4
    anycast_gateway_mac: 20:20:00:00:00:aa
```

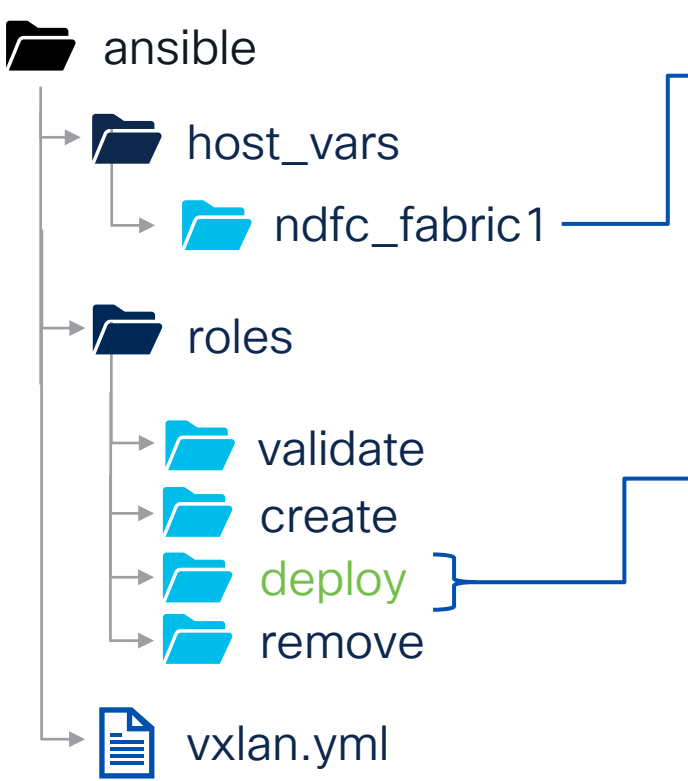
validate → create → deploy → remove

- ✓ Transform model data for NDFC module processing
- ✓ Call cisco.dcnm.fabric module to create/manage fabric with model data settings.

Model – Fabric Management

Data to do it

How to do it



global.nac.yaml

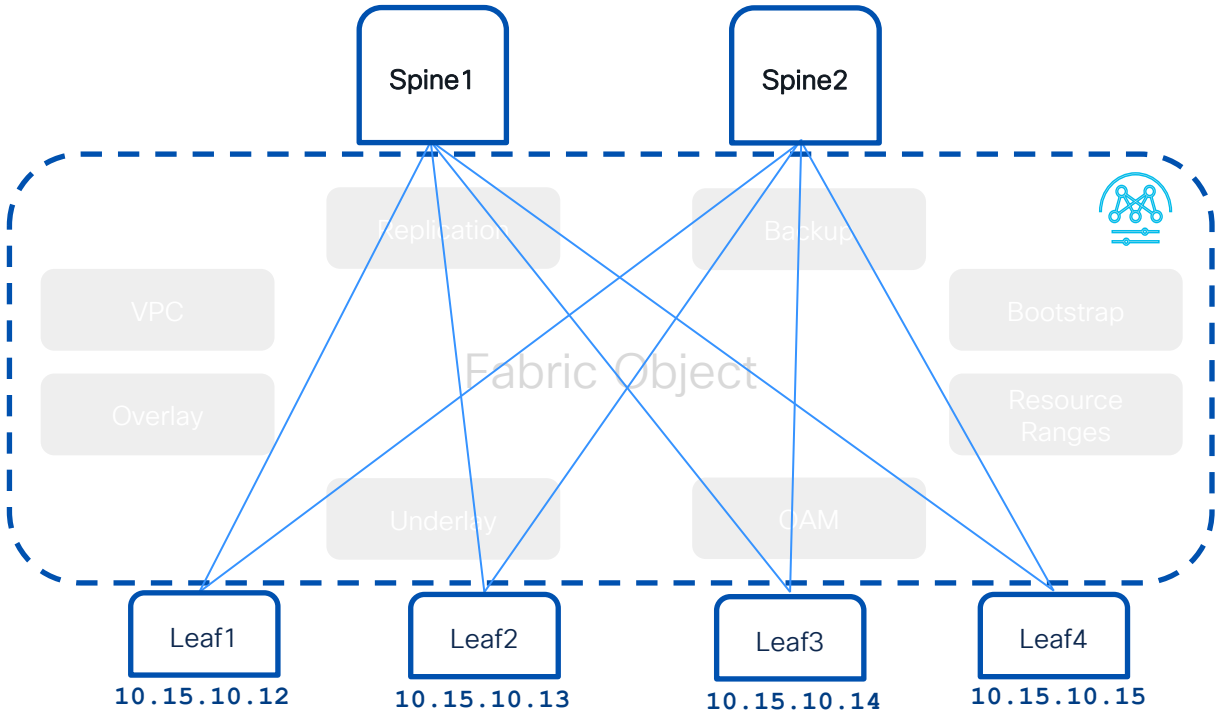
Visual Studio Code

```
---
vxlan:
  global:
    name: ndfc_fabric1
    bgp_asn: 65055
    route_reflectors: 4
    anycast_gateway_mac: 20:20:00:00:00:aa
```

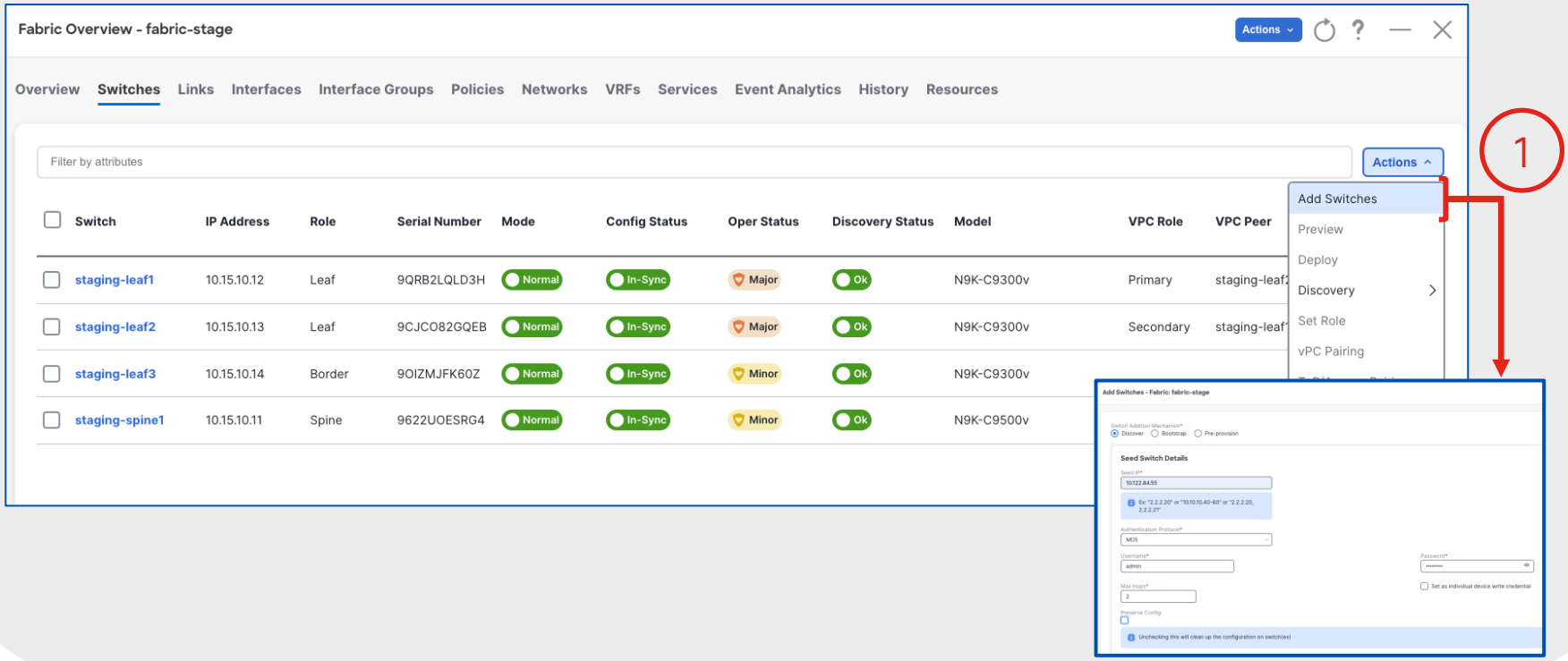
validate → create → deploy → remove

- ✓ Save and deploy state changes
- ✓ Remove role not used for fabric management stage

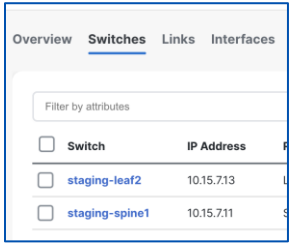
Workflow Step2: Inventory Management

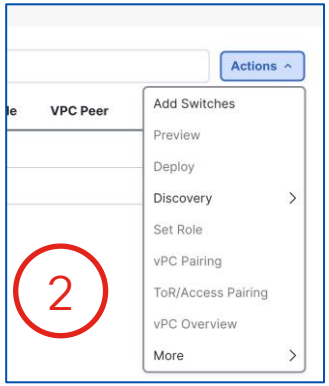


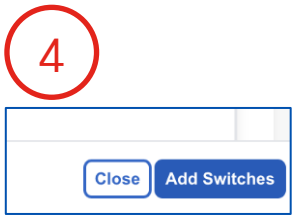
Workflow Step2: Inventory Management

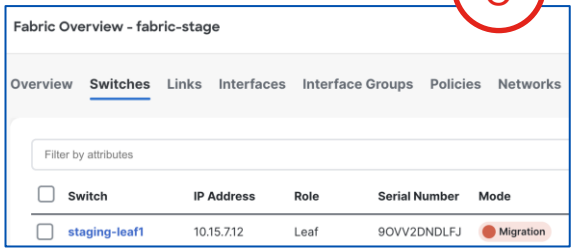


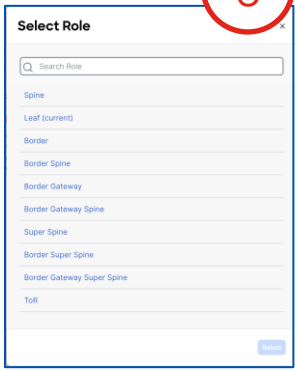
Workflow Step2: Inventory Management

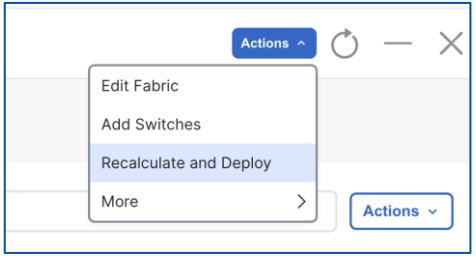








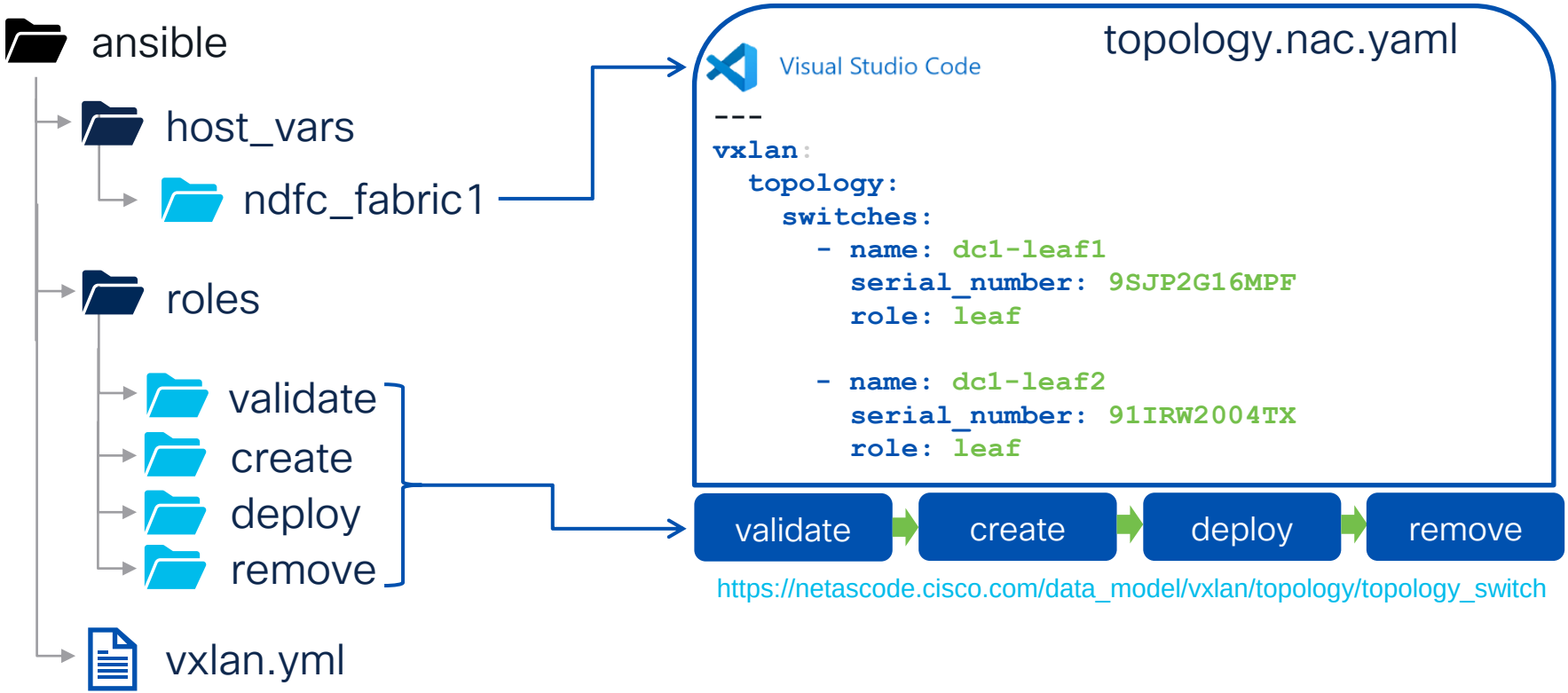




Model – Inventory Management

Data to do it

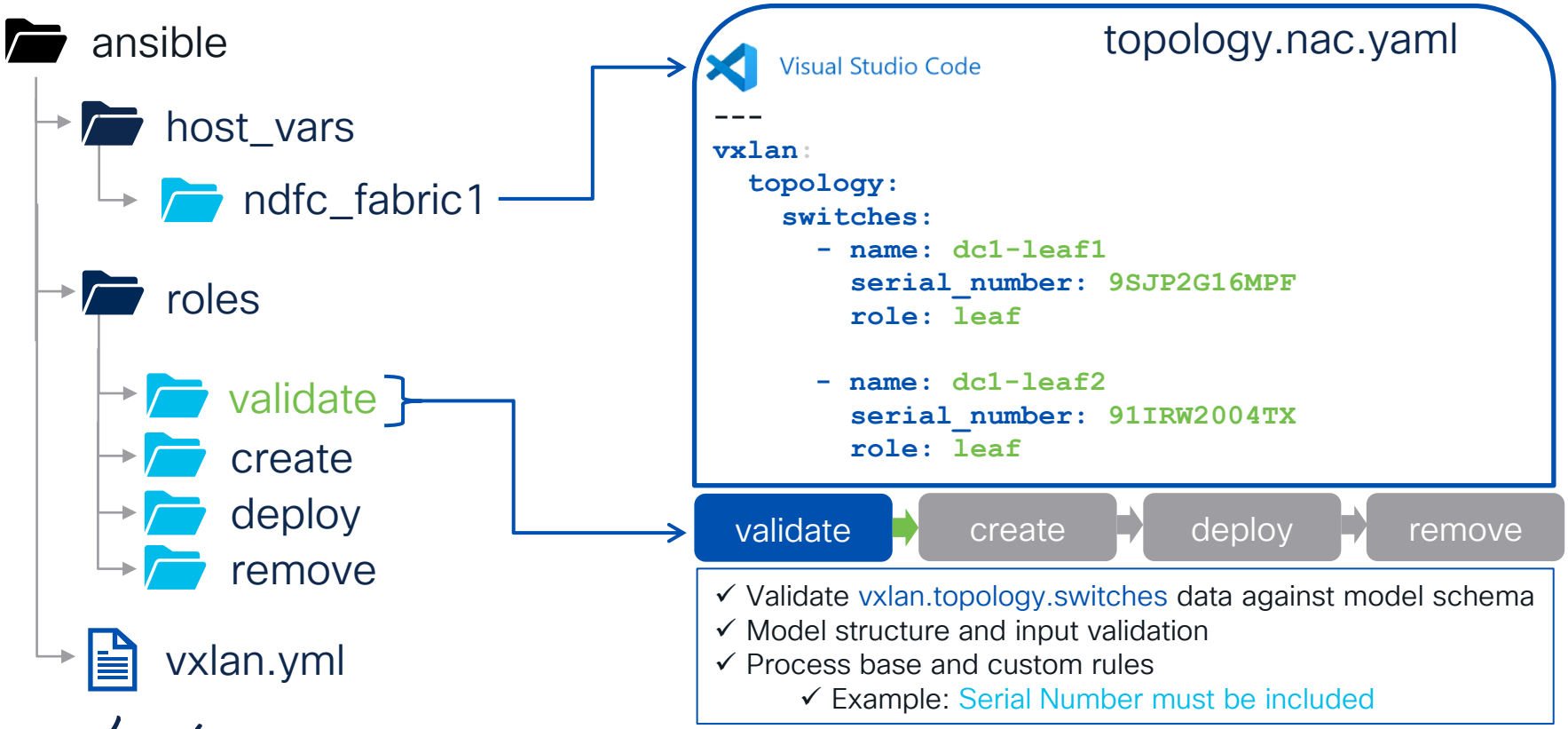
How to do it



Model – Inventory Management

Data to do it

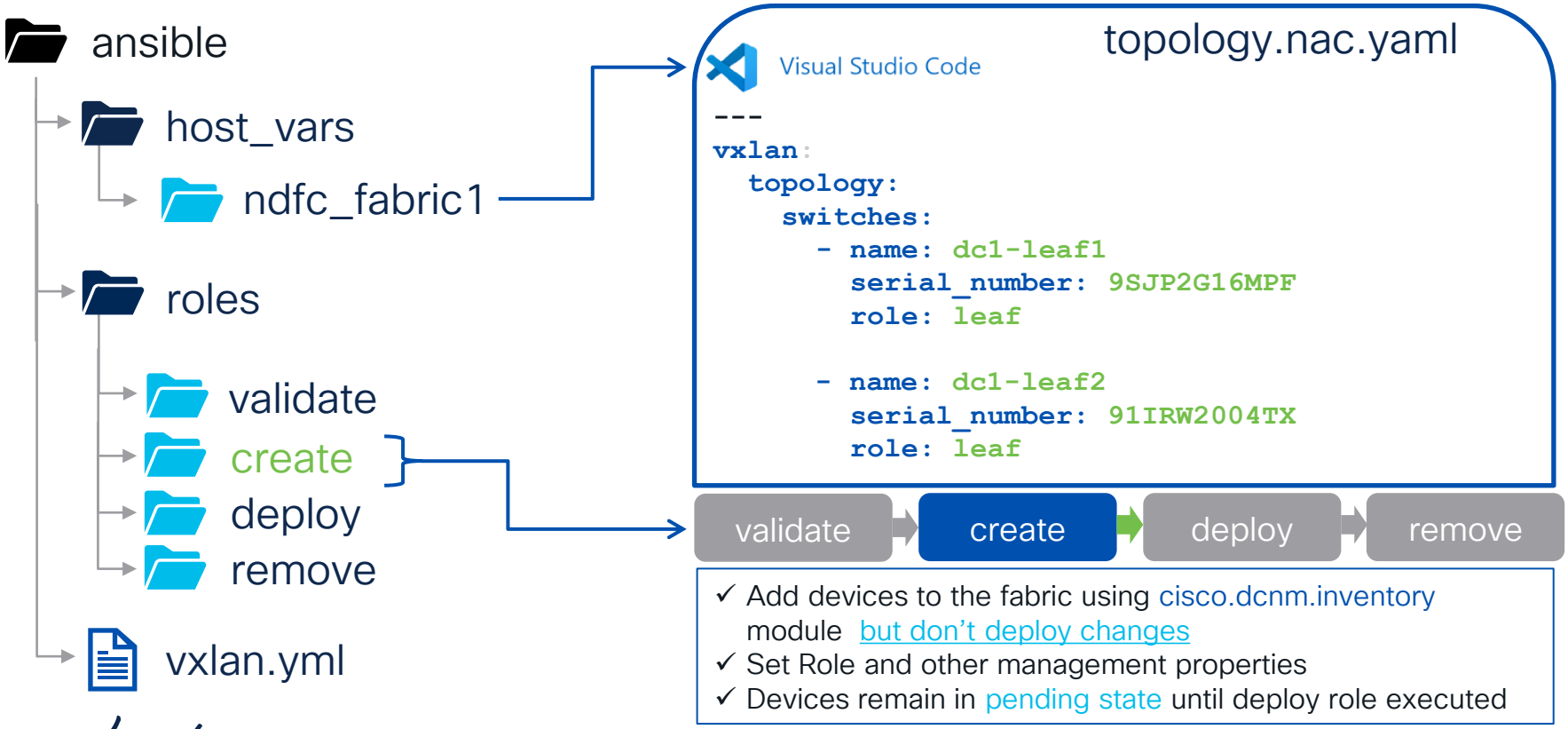
How to do it



Model – Inventory Management

Data to do it

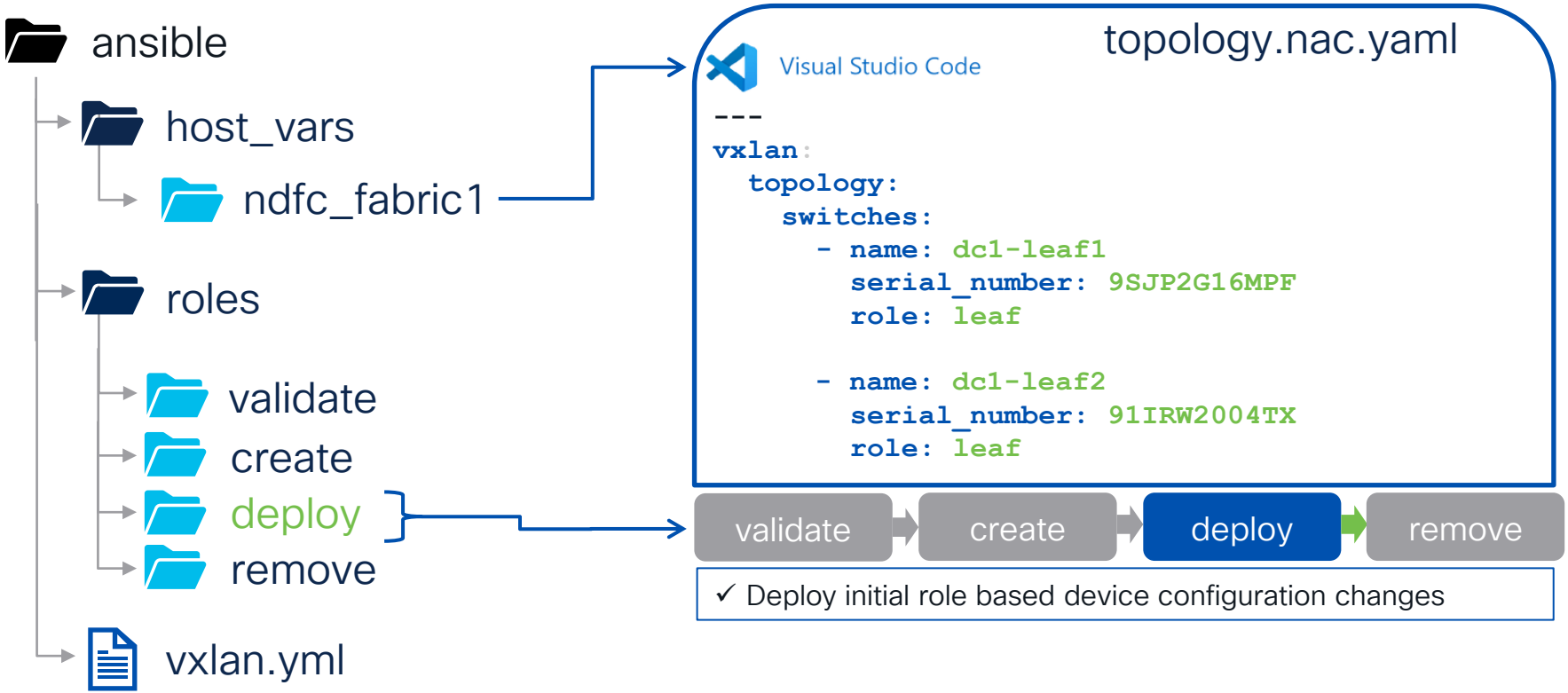
How to do it



Model – Inventory Management

Data to do it

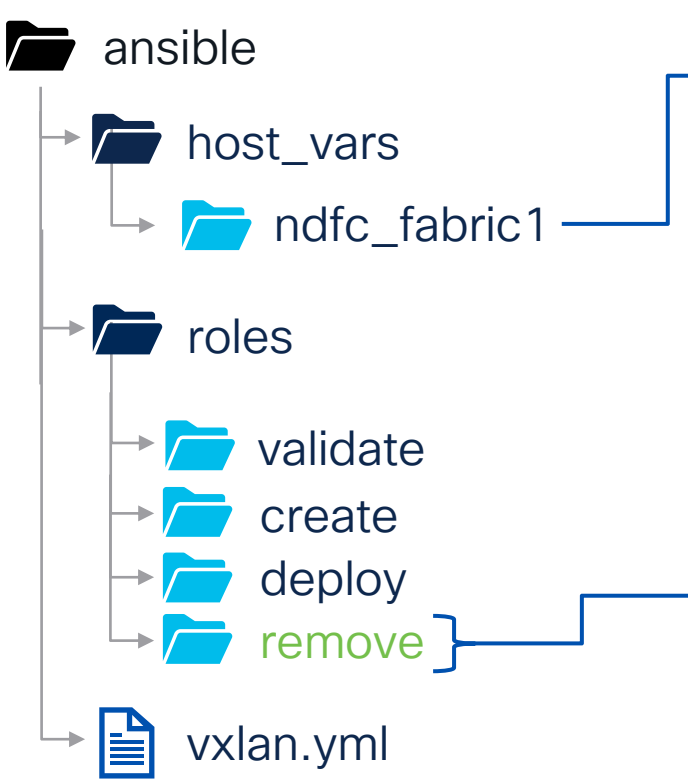
How to do it



Model – Inventory Management

Data to do it

How to do it



topology.nac.yaml

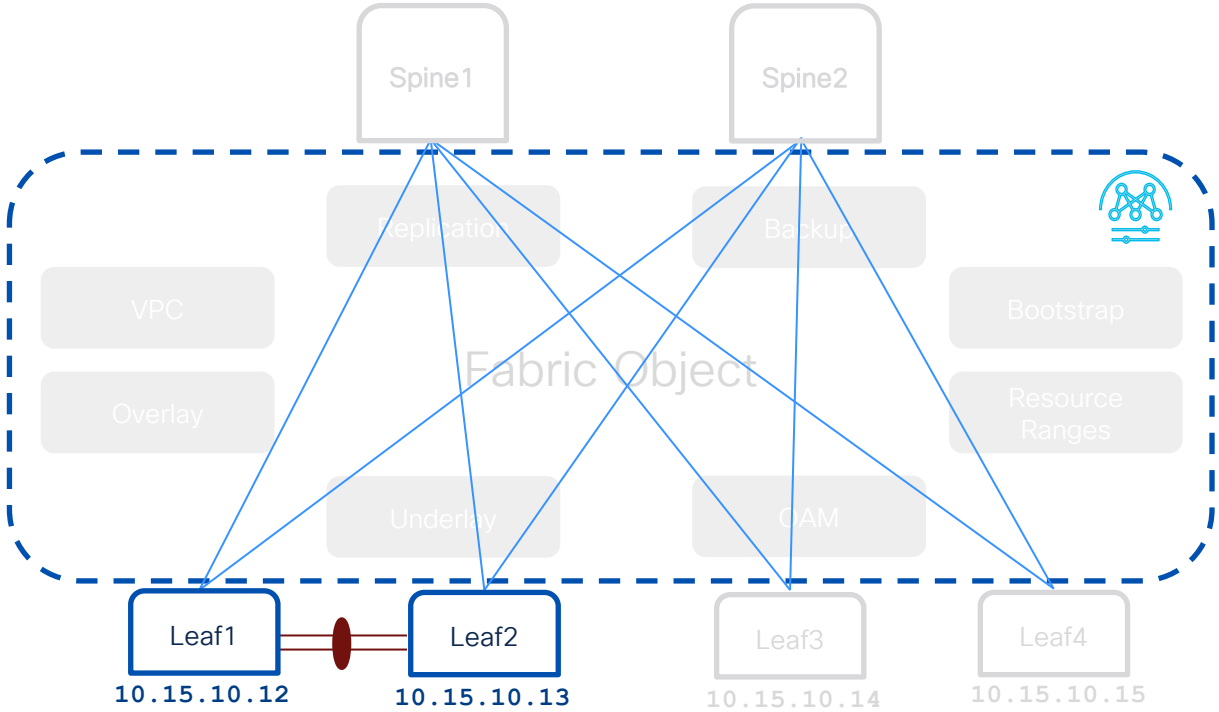
Visual Studio Code

```
---
vxlan:
  topology:
    switches:
      - name: dc1-leaf1
        serial_number: 9SJP2G16MPF
        role: leaf

      - name: dc1-leaf2
        serial_number: 91IRW2004TX
        role: leaf
```

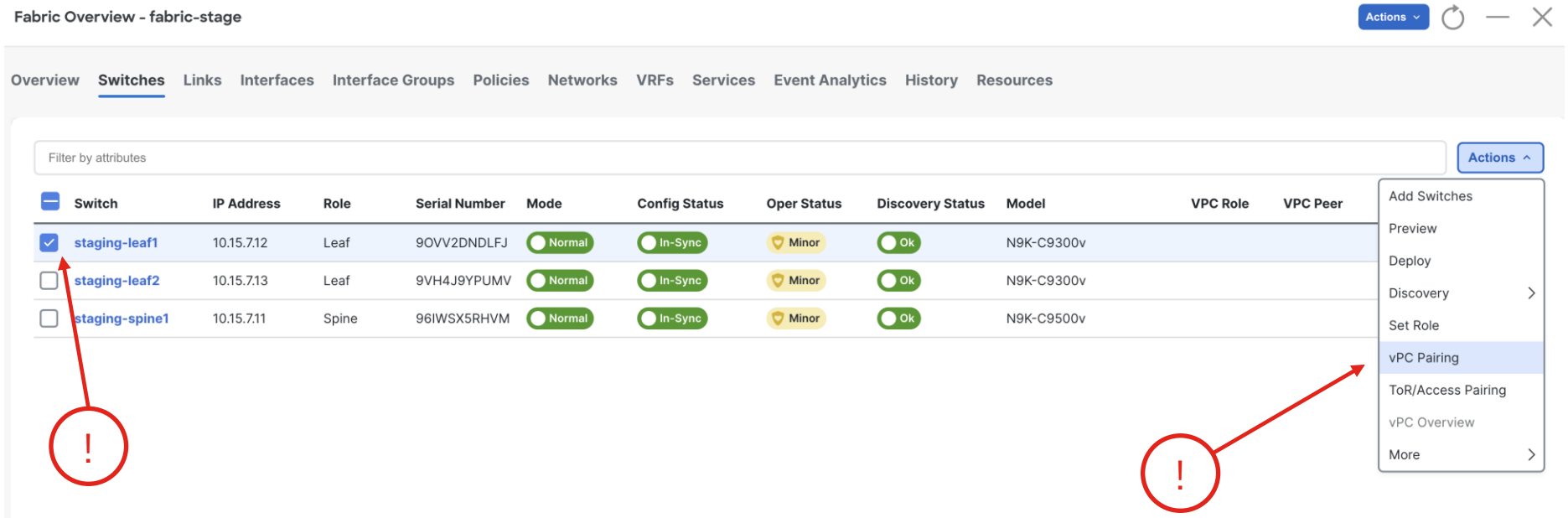
validate → create → deploy → remove

- ✓ Commenting out or removing device from list will cause the device to be removed from the fabric by the remove role
- ✓ Guarded by a safety flag inventory_delete_mode: false

Workflow Step3: vPC Pair Management



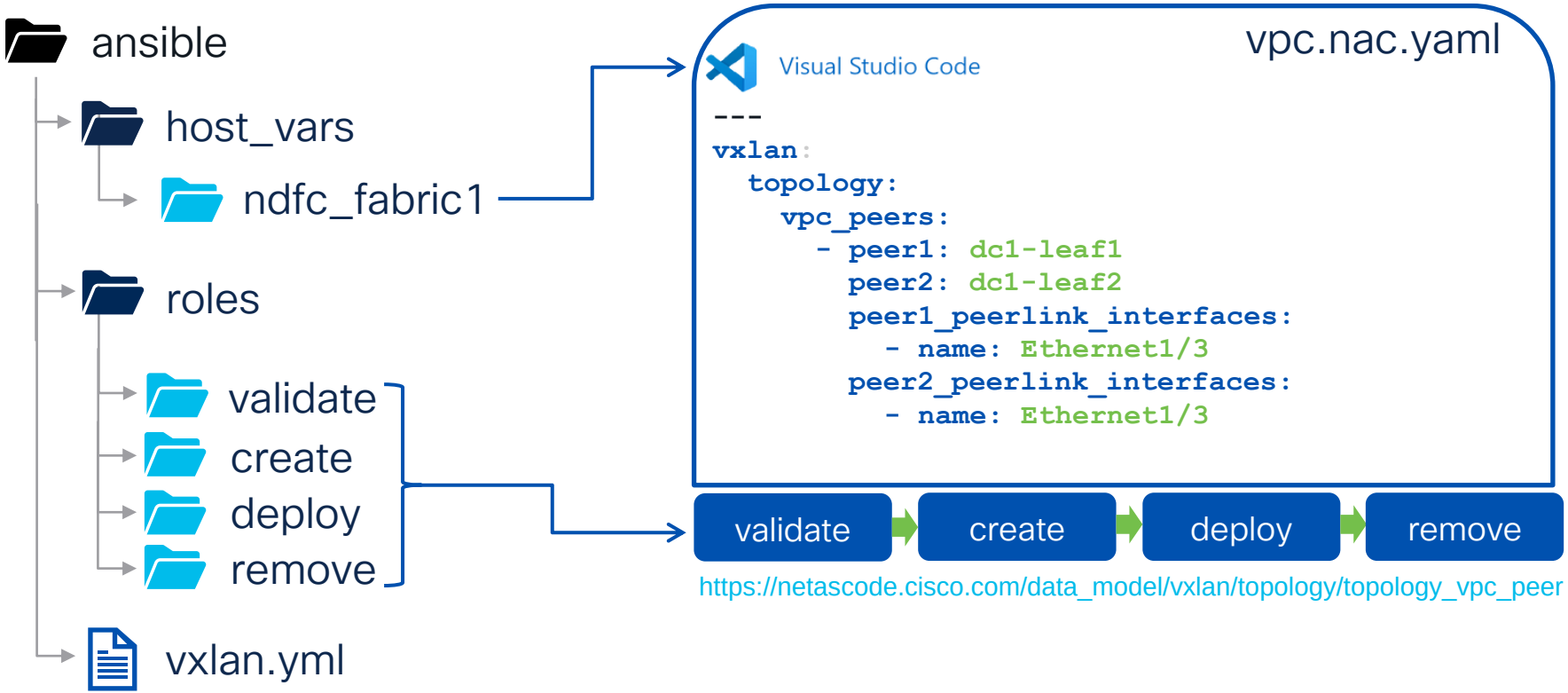
Workflow Step3: vPC Pair Management



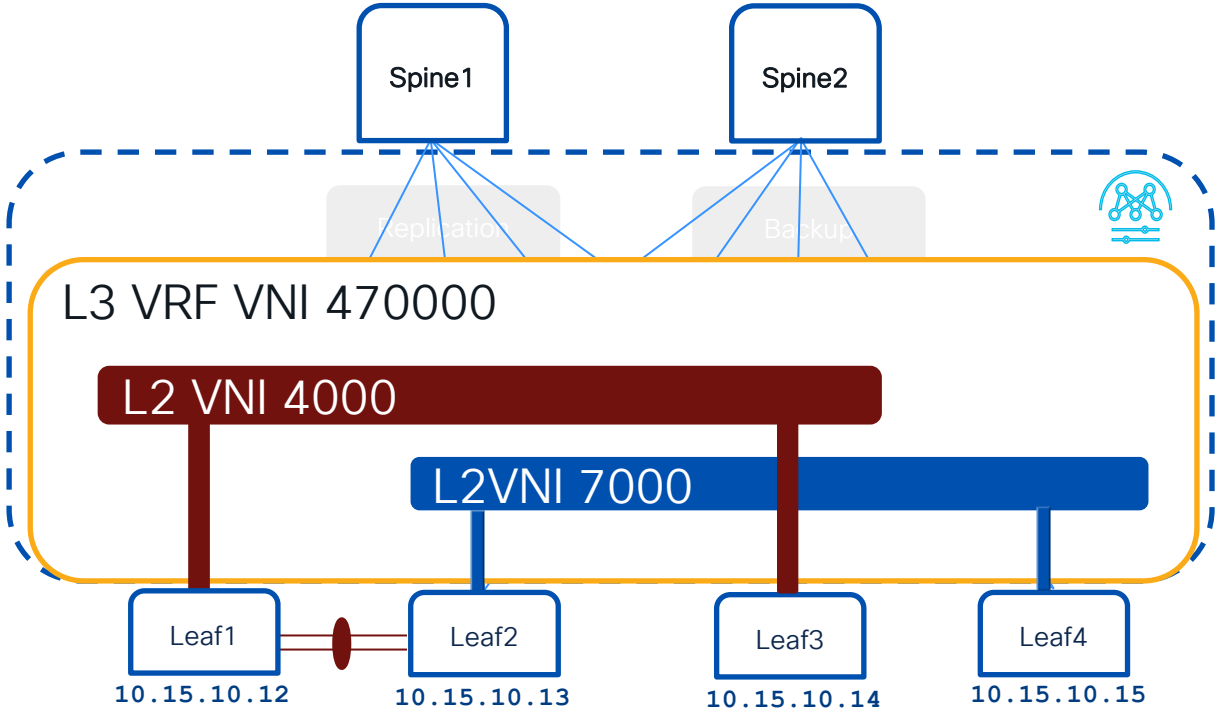
Model – vPC Pair Management

Data to do it

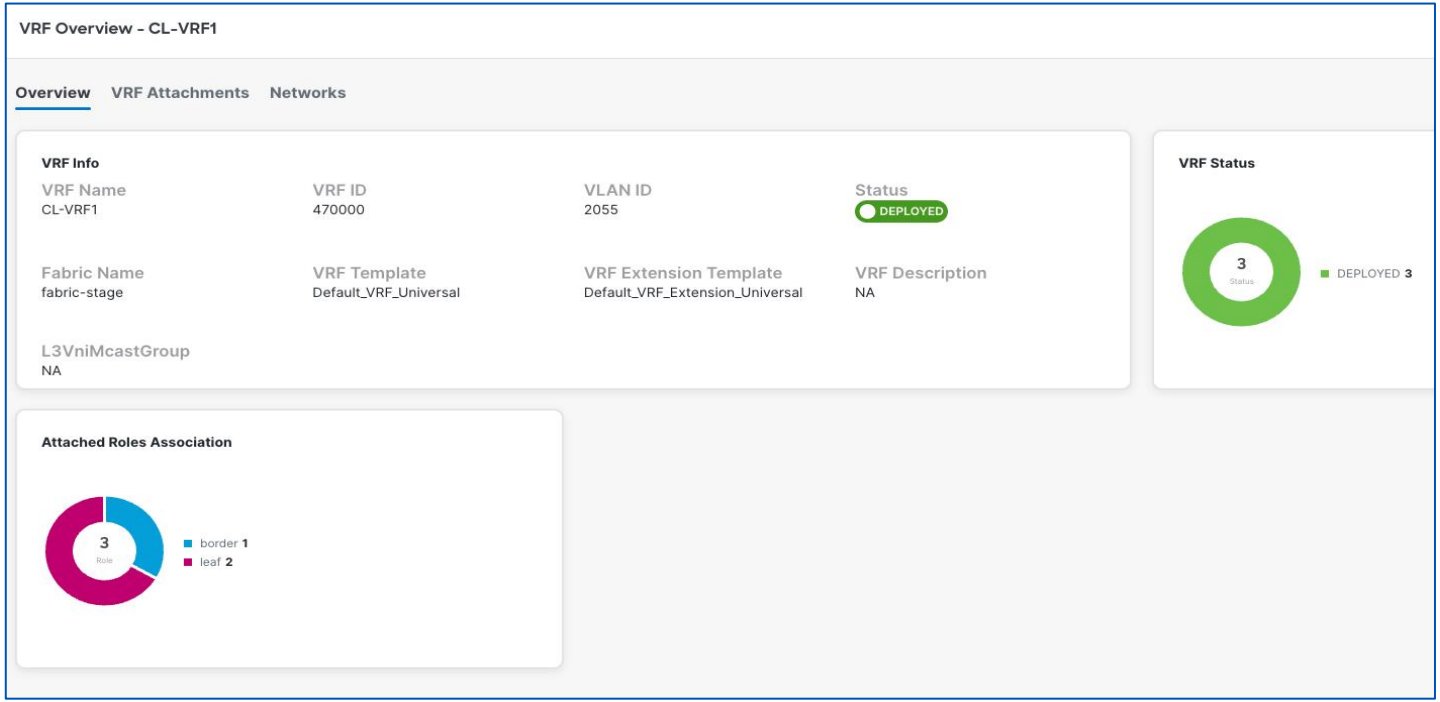
How to do it



Workflow Step4: Overlay Management



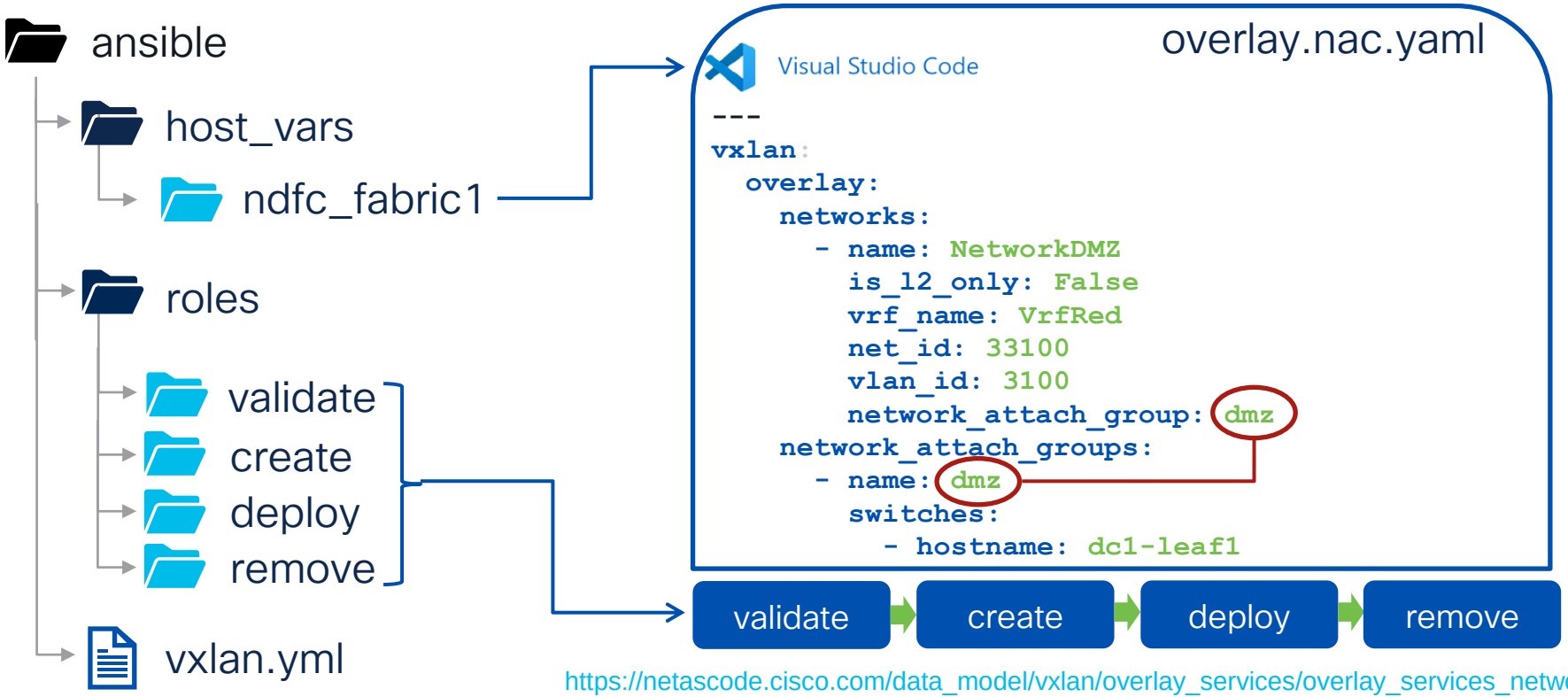
Workflow Step3: Overlay Management



Model – Overlay Management

Data to do it

How to do it



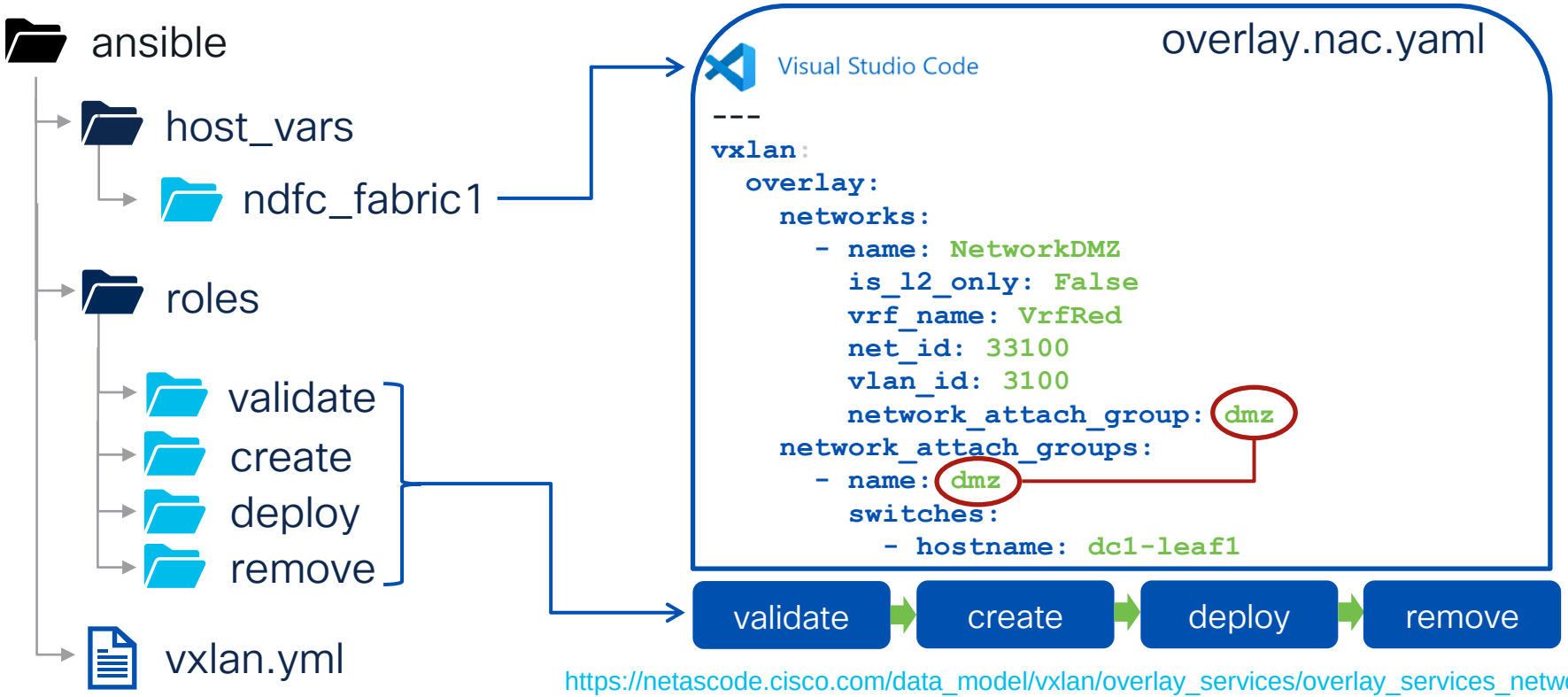
https://netascode.cisco.com/data_model/vxlan/overlay_services/overlay_services_network

https://netascode.cisco.com/data_model/vxlan/overlay_services/overlay_services_vrf

Model – Overlay Management

Data to do it

How to do it



https://netascode.cisco.com/data_model/vxlan/overlay_services/overlay_services_network

https://netascode.cisco.com/data_model/vxlan/overlay_services/overlay_services_vrf

Validate Role – Deeper Dive

Example of YAMALE Schema Validation

- Invalid anycast_gateway_mac
- Early detection before sending any requests to the fabric

```
---
vxlan:
  global:
    name: nac-ndfc1
    fabric_type: VXLAN_EVPN
    bgp_asn: 65001
    route_reflectors: 2
    anycast_gateway_mac: 12:34:56:78:90:00E99E
```

Validate Role – Deeper Dive

Example of YAMALE Schema Validation

- Invalid anycast_gateway_mac
- Early detection before sending any requests to the fabric

```
---
vxlan:
  global:
    name: nac-ndfc1
    fabric_type: VXLAN_EVPN
    bgp_asn: 65001
    route_reflectors: 2
    anycast_gateway_mac: 12:34:56:78:90:00E99E
```

"msg":
"Syntax error '/home/cisco/Documents/nac-vxlan/host_vars/nac-ndfc1/global.nac.yaml':
vxlan.global.anycast_gateway_mac: '12:34:56:78:90:00E99E' is not a mac match"

Validate Role – Deeper Dive

Example of Base Rule Validation

- Network referencing an undefined VRF
- Early detection before sending any requests to the fabric

```
---
vxlan:
  overlay:
    vrfs:
    # VRF Commented out
    #   - name: NaC-VRF01
```

```
---
vxlan:
  overlay:
    networks:
      - name: NaC-Net01
        is_l2_only: False
        vrf_name: NaC-VRF01
```

Validate Role – Deeper Dive

Example of Base Rule Validation

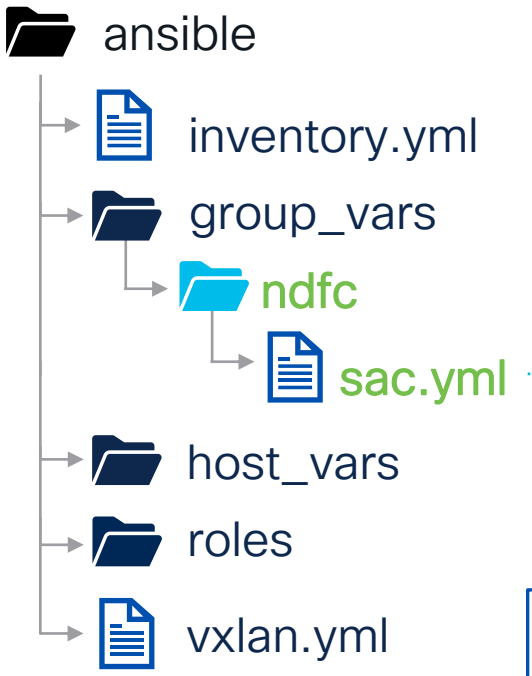
- Network referencing an undefined VRF
- Early detection before sending any requests to the fabric

```
---
vxlan:
  overlay:
    vrfs:
    # VRF Commented out
    #   - name: NaC-VRF01
```

```
---
vxlan:
  overlay:
    networks:
      - name: NaC-Net01
        is_l2_only: False
        vrf_name: NaC-VRF01
```

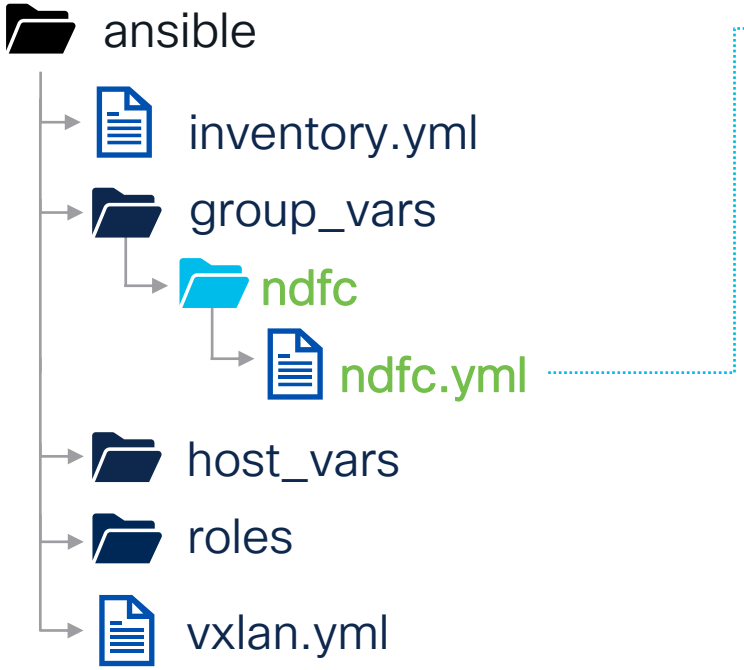
"msg": "Semantic error, rule 401: Cross Reference VRFs and Networks items in the Service Model (['Network (NaC-Net01) is referencing VRF (NaC-VRF01) which is not defined in the service model. Add the VRF to the service model or remove the network from the service model and re-run the playbook."

Ansible Group Vars – Schema and Custom Rules




```
schema_path: "{{ lookup('ansible.builtin.env', 'DC_VXLAN_SCHEMA') }}"
enhanced_rules_path: "{{ lookup('ansible.builtin.env', 'DC_VXLAN_RULES') }}"
```

Ansible Group Vars – Control Flags




```yaml
# Parameter to ignore diff-run
force_run_all: false

# Parameters for the tasks in the 'Remove' role
interface_delete_mode: true
inventory_delete_mode: true
link_vpc_delete_mode: true
network_delete_mode: true
policy_delete_mode: true
vpc_delete_mode: true
vrf_delete_mode: true
```

Flag	Default Value
force_run_all	false
*_delete_mode	false

NDFC as Code – Ansible Tags

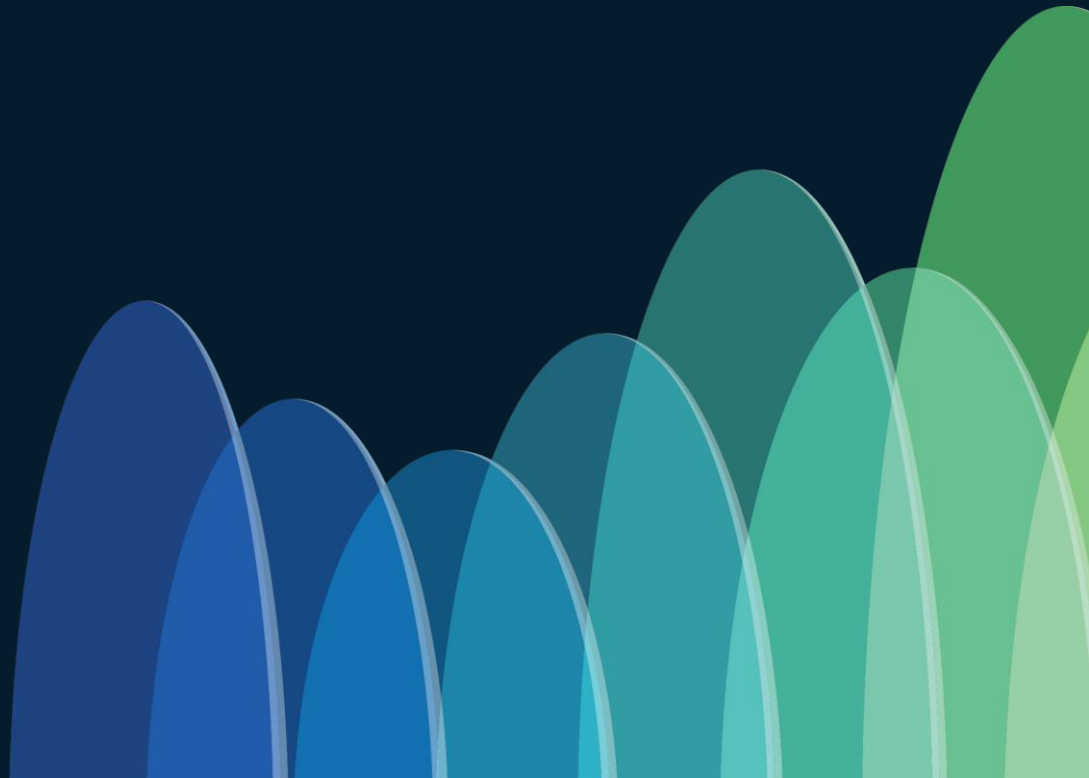
Tags for selective execution control

ansible-playbook -i inventory.yaml vxlan.yaml --tags role* -vvvv

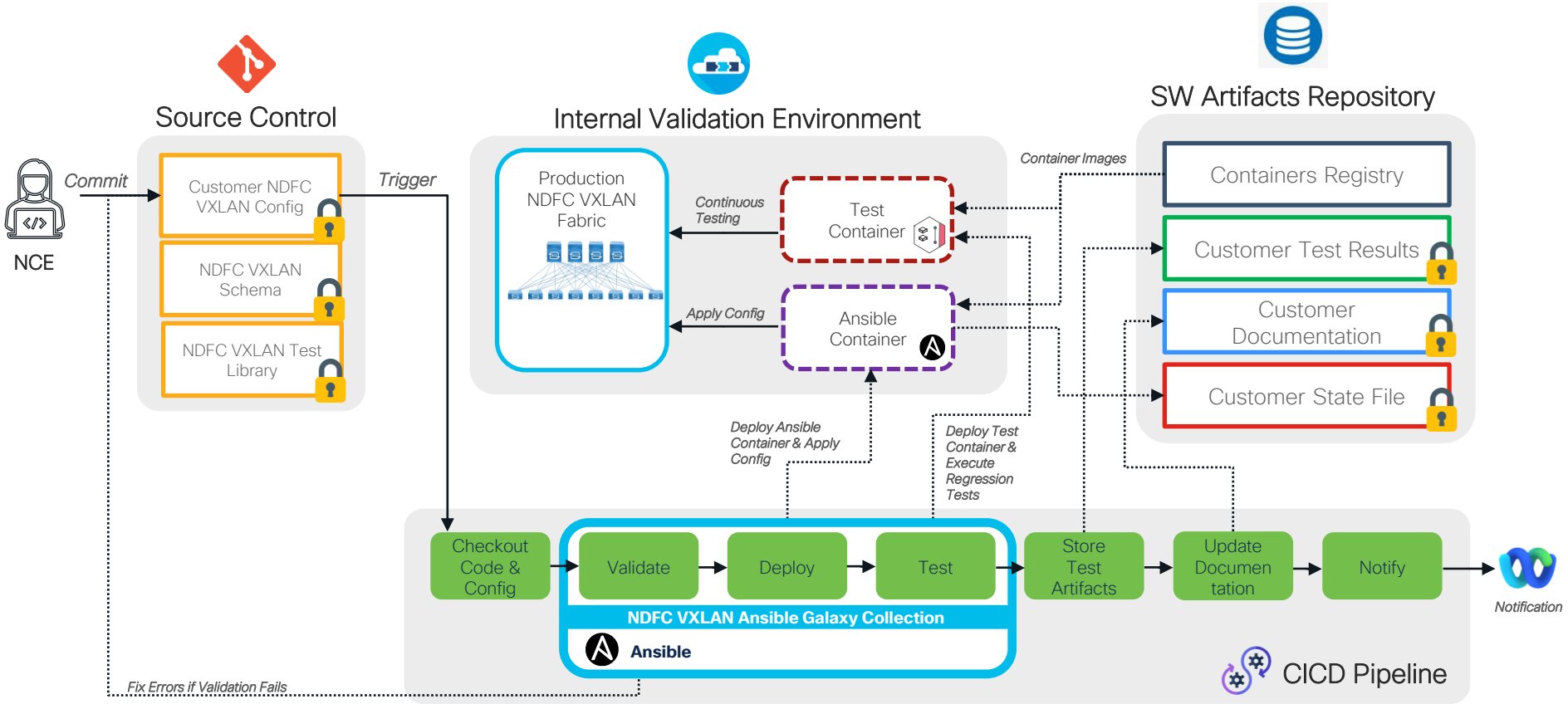
```
ASK [cisco.nac_dc_vxlan.common_global : Verify User Tags]
*********************************************************************************************
***************
task path: /home/cisco/Documents/nac-
vxlan/collections/ansible_collections/cisco/nac_dc_vxlan/roles/common_global/tasks/main.yml:23
Saturday 25 January 2025  20:16:41 +0000 (0:00:01.694)       0:00:01.694 ******
Saturday 25 January 2025  20:16:41 +0000 (0:00:01.693)       0:00:01.693 ******
fatal: [nac-ndfc1 -> localhost]: FAILED! => {
    "changed": false,
    "msg": "Tag 'role*' not found in list of supported tags",
    "supported_tags": [
        "cc_verify",
        "cr_manage_fabric",
        "cr_manage_switches",
        "cr_manage_vpc_peers",
        "cr_manage_interfaces",
        "cr_manage_vrfs_networks",
        "cr_manage_policy",
        "cr_manage_links", >>>
```

(More tags supported like role_verify, role_create, role_deploy, role_remove)

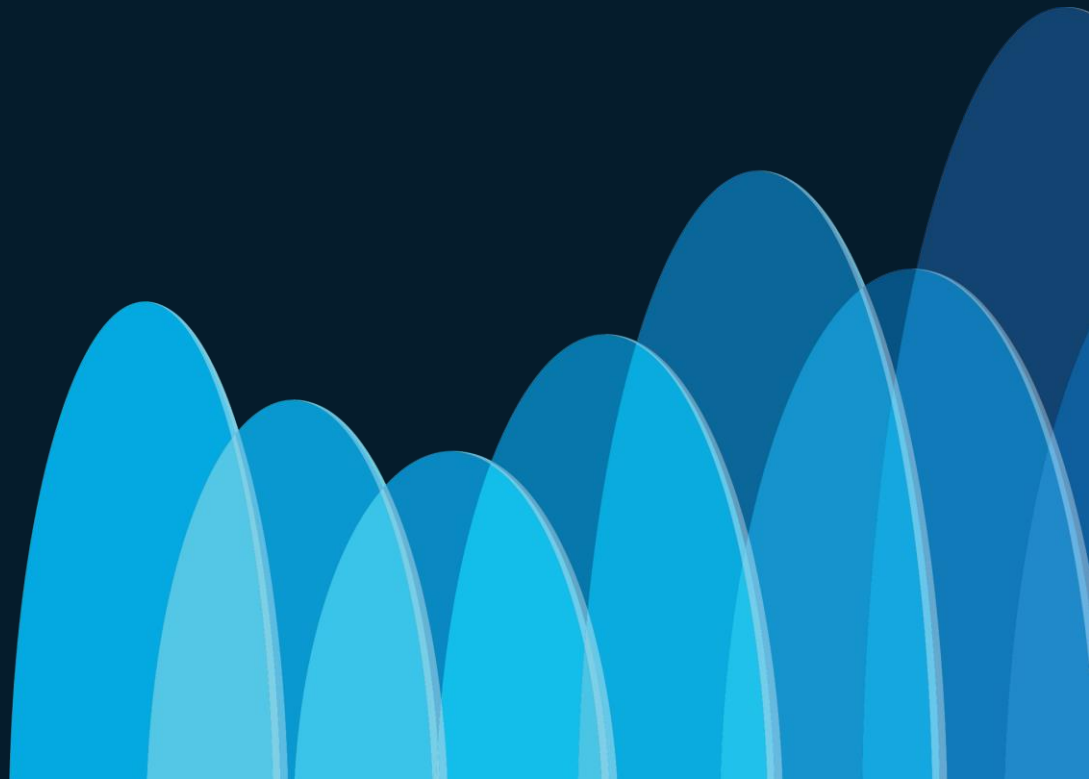
CI/CD Pipeline Demo



Standard NDFC new customer configuration pipeline



References to Start Your Journey



cisco Live!

Ansible VXLAN as Code Repos

VXLAN NaC Collection Repo

https://github.com/netascode/ansible-dc-vxlan

VXLAN NaC Collection Example Repo

https://github.com/netascode/ansible-dc-vxlan-example

Ansible for NXOS and NDFC Repos

BRKDCN-2946
Session Repo

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

More Information

- https://www.ansible.com/resources/get-started
- https://docs.ansible.com/ansible/latest/collections_guide/index.html
- https://galaxy.ansible.com/cisco/nac_dc_vxlan
- 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)

Webex App

Questions?

Use the Webex app to chat with the speaker after the session

How

1. Find this session in the Cisco Events 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 February 28, 2025.

Fill Out Your Session Surveys

Participants who fill out a minimum of 4 session surveys and the overall event survey will get a unique Cisco Live t-shirt.

(from 11:30 on Thursday, while supplies last)

All surveys can be taken in the Cisco Events mobile app or by logging in to the Session Catalog and clicking the ‘Participant Dashboard’

Content Catalog

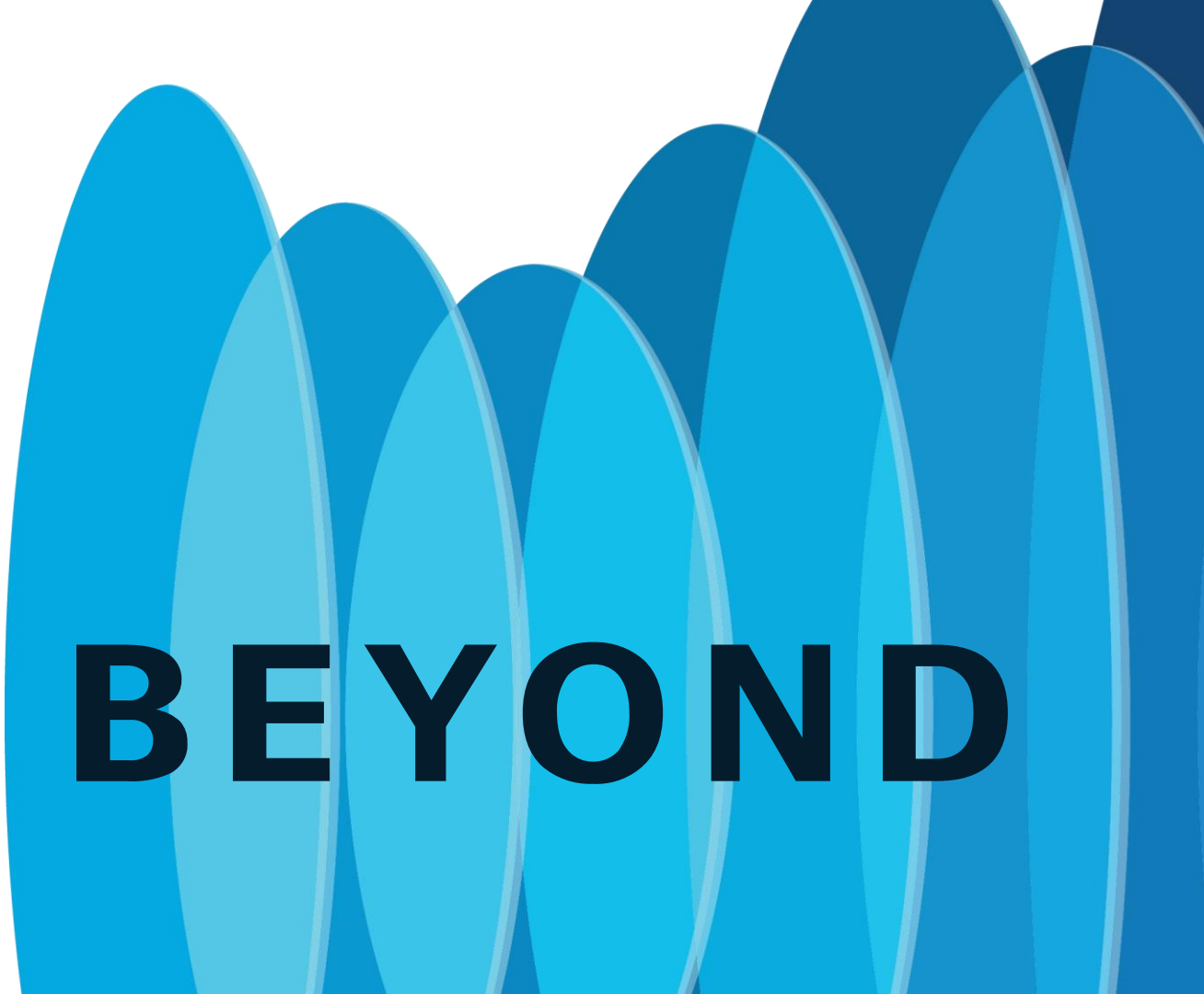
Continue your education

- 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 ciscolive.com/on-demand. Sessions from this event will be available from March 3.

Contact me at: https://www.linkedin.com/in/mike-g-wiebe/


cisco
Thank you
cisco Live!

cisco Live!



GO BEYOND