The bridge to possible

Introduction to Infrastructure as Code for ACI with Ansible

Subtitle goes here

Rafael Muller, Principal Engineer CX
@rafaeljmuller

BRKDCN-2606

Evolution of automation in ACI

2014

The ERA of python with ReST

- When first introduced, libraries like CobraSDK were developed and utilized to automate configuration of the ACI Fabric

2016

The ERA of applications

- More Python, now laced with Javascript, created bespoke applications that utilized ACI's API first methodology to automate processes and tie them to business needs.

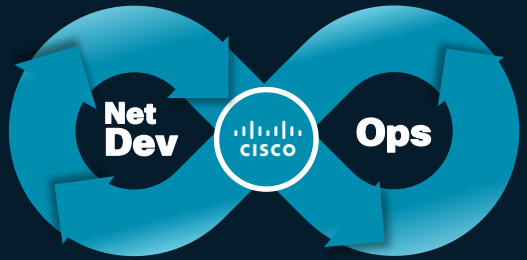
2019

The ERA of automation pipelines

- More and more network operators started standardizing around common supported automation tools that removed the need for programing knowledge.
- Requesting network vendors to create capabilities around these common tools
- Asking questions as to how they can optimize like there DevOps counterparts

What is Infrastructure as Code (IaC)?

- *The management & provisioning of computer infrastructure through code and data structures instead of direct device management.*
- Originated in public cloud infrastructure builds
- Utilized amongst DevOps practitioners assisting in testing and validating changes to applications
- Migrated by demand into the network world as NetDevOps
- Focused on a **declarative** way of automation



Automation Types

GUI

Optimizations provided

- Task is managed directly in the product interface to complete required change.
- Template system to optimize configuration deployments
- GUI Prone to human error when many procedures are being executed

Procedural

Execution to complete a task

- Task is defined to be executed to complete a required change.
- Executed from network operators who own the task execution
- Run once and complete

Infrastructure as Code

Declarative

Embedded in your practice

- Code contains the **source of truth**
- Task is executed from an automation system, validated by processes and executed via orchestration
- Individual operators don't have access to change.
- Minimal errors and mistakes

is one of the

Tools

that can achieve

Infrastructure as Code

for ACI

Rafael Muller

Principal Engineer
Customer Experience CX

29 years @ Cisco

Presented at Networkers & Cisco Live since 1997
Hall of Fame – Elite Speaker

Focused on Datacenter Automation

The bridge to possible

Introduction to Infrastructure as Code for ACI with Ansible

Rafael Muller, Principal Engineer CX
@rafaeljmuller
BRKDCN-2906

Cisco Webex App

https://ciscolive.ciscoevents.com/ciscolivebot/#BRKDCN-2606

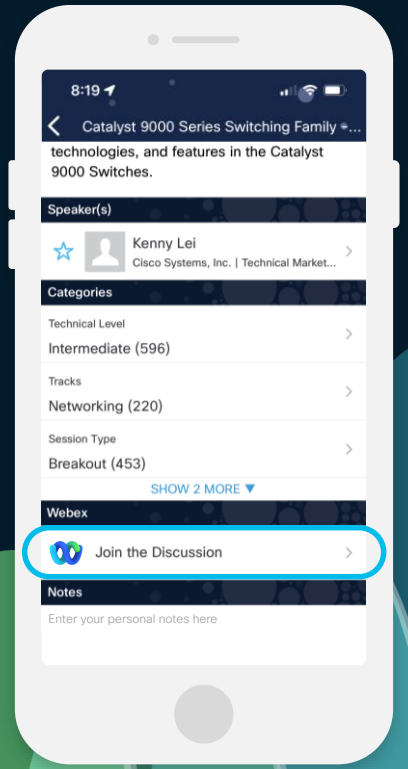
Questions?

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.

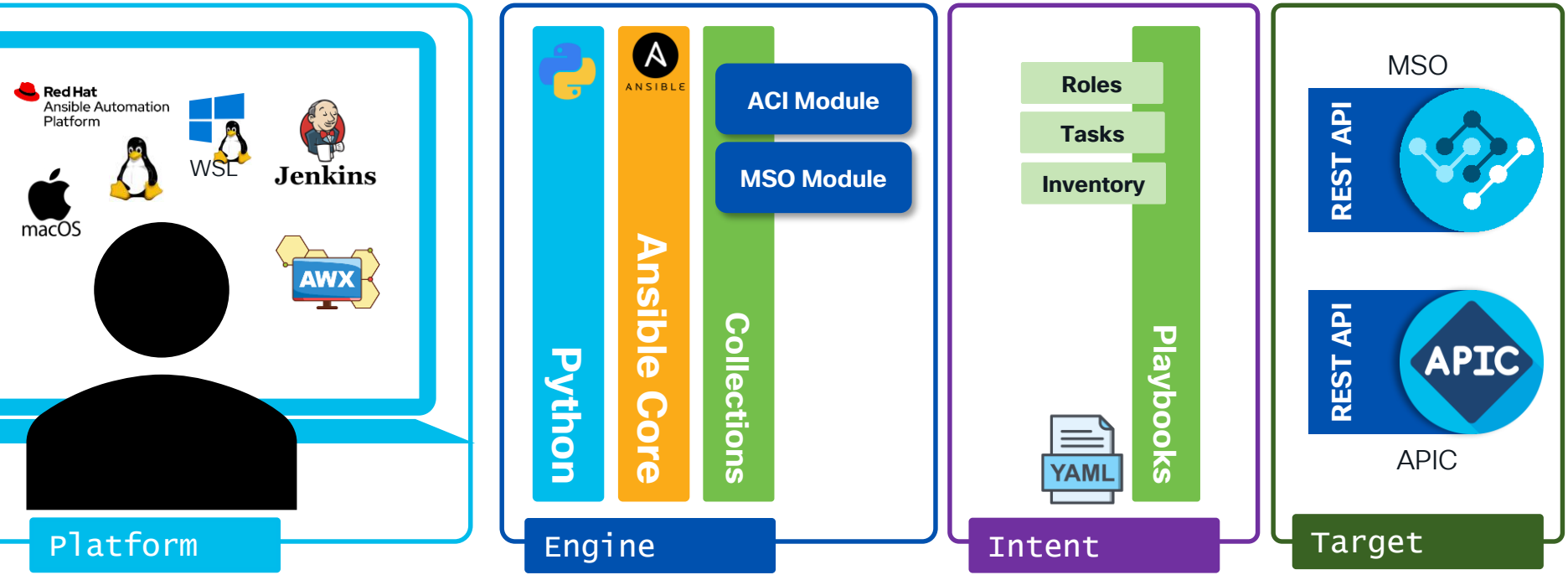


What is Ansible?

- 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
- APIC/NDO REST API interaction

*idempotent

What makes up Ansible?



Installing Ansible

Python Virtual Environments

python virtualenv

- You should (better yet, must) use a **virtual environment**.
- Proper virtual environment allows for installing ansible inside a contained area with a specific version of python.
- Makes it possible to run different python scripts that require different versions of python and libraries of python.
- Detailed steps beyond scope of this session.

Hidden slide has details on pyenv

PyENV

Virtual Environment in Python

- 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

1. install a version of python

```
% pyenv install 3.10.13
```

2. create virtual-environment

```
% pyenv virtualenv 3.10.13 ansible
```

3. create directory for your ansible work

```
% mkdir my_ansible_dir
```

4. tell PyENV the virtual-env to use here

```
% pyenv local ansible
```

Install instructions:

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

Ansible install

Core or Everything

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

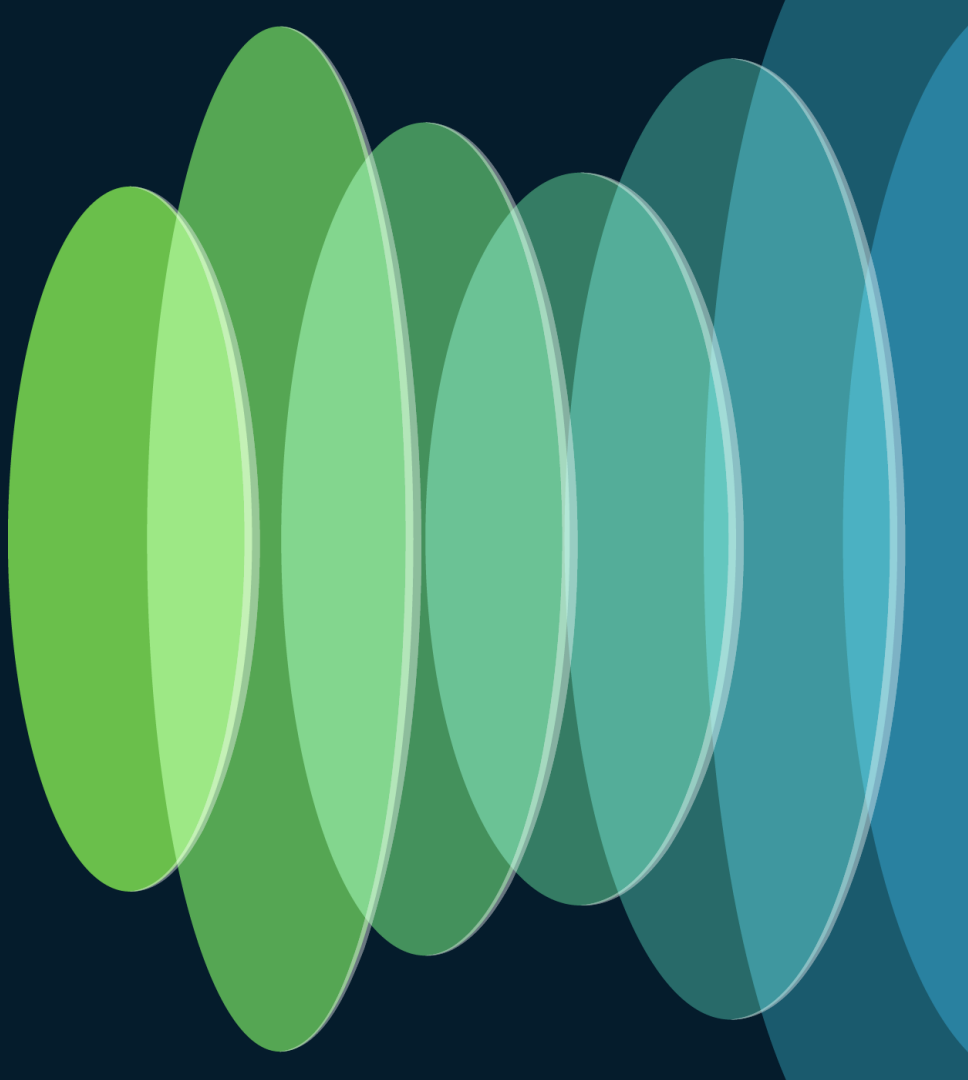
- Ansible installs only the `core components`
- Collections must be installed manually
- Smaller footprint and more control
- Assures install of latest collection version released!

```
% pip install ansible
```

- Ansible installs `all collections` with the Ansible install
- Complete package but consumes more disk space.
- Might not install the latest version of the collection!

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Ansible Collections



What are Ansible collections?

- Introduced in Ansible 2.9
- Collections allows vendors to de-couple their ansible capabilities (modules) from the core Ansible release schedule
- Uses ***Ansible Galaxy*** as the delivery vehicle.
- Collection can be installed in any location with **-p** flag

```
% ansible-galaxy collection install cisco.aci cisco.mso
```

ACI - https://galaxy.ansible.com/cisco/aci

MSO - https://galaxy.ansible.com/cisco/mso

Installing Ansible Collections

Command

Required packages

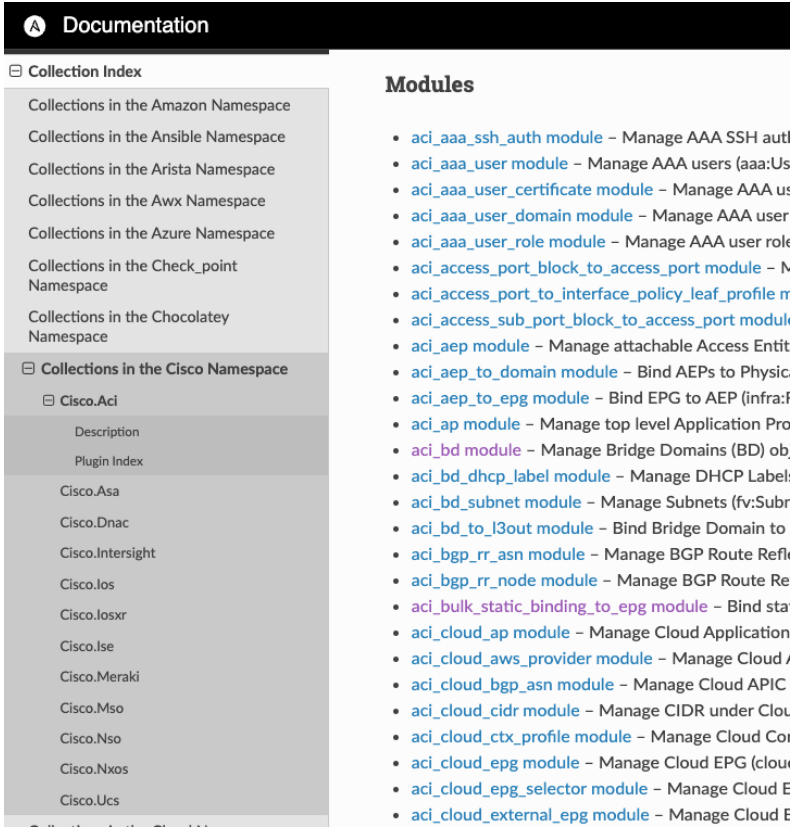
```
❯ ansible-galaxy collection install cisco.aci cisco.mso
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/download/cisco-aci-2.3.0.tar.gz to /Users/rmuller/.ansible/tmp/ansible-local-464
00clazzrla/tmpn6m0ecmc/cisco-aci-2.3.0-6vczo8j6
Installing 'cisco.aci:2.3.0' to '/Users/rmuller/.ansible/collections/ansible_collections/cisco/aci'
Downloading https://galaxy.ansible.com/download/cisco-mso-2.1.0.tar.gz to /Users/rmuller/.ansible/tmp/ansible-local-464
00clazzrla/tmpn6m0ecmc/cisco-mso-2.1.0-r0d0u2xn
cisco.aci:2.3.0 was installed successfully
Installing 'cisco.mso:2.1.0' to '/Users/rmuller/.ansible/collections/ansible_collections/cisco/mso'
Downloading https://galaxy.ansible.com/download/ansible-netcommon-4.1.0.tar.gz to /Users/rmuller/.ansible/tmp/ansible-l
ocal-46400clazzrla/tmpn6m0ecmc/ansible-netcommon-4.1.0-svcyh9z6
cisco.mso:2.1.0 was installed successfully
Installing 'ansible.netcommon:4.1.0' to '/Users/rmuller/.ansible/collections/ansible_collections/ansible/netcommon'
Downloading https://galaxy.ansible.com/download/ansible-utils-2.8.0.tar.gz to /Users/rmuller/.ansible/tmp/ansible-local
-46400clazzrla/tmpn6m0ecmc/ansible-utils-2.8.0-srfniw47
ansible.netcommon:4.1.0 was installed successfully
Installing 'ansible.utils:2.8.0' to '/Users/rmuller/.ansible/collections/ansible_collections/ansible/utils'
ansible.utils:2.8.0 was installed successfully
```

Collection can be installed in any location with `-p` flag

Ansible ACI/MSO

Collection of **Modules**

- Primary reason they are called collections is because they are a collection of modules
- Modules perform specific tasks like create EPG's, Bridge domains, VRF's
- Actively maintained with regular cadence that increases module count and capability



https://docs.ansible.com/ansible/latest/collections/cisco/aci/index.html

Hidden slide has details on CLI doc

Ansible ACI/MSO
Collection Modules (CLI)

Use the CLI also to reach the module documentation.

Use grep to filter through all the available documentation installed.

```
❯ ansible-doc -l | grep cisco.aci
cisco.aci.aci_aaa_ssh_auth
cisco.aci.aci_aaa_
cisco.aci.aci_aaa_user_certificate
```

The command: `ansible-doc <module_name>` will present the CLI version of the doc. Will match what is on the web

```
❯ ansible-doc cisco.aci.aci_aaa_user

      Manage AAA users on Cisco ACI fabrics.

OPTIONS (= is mandatory):
- aaa_password
      The password of the locally-authenticated user.
      default: null
      type: str
- aaa_password_lifetime
      The lifetime of the locally-authenticated user password.
      default: null
      type: int

EXAMPLES:

- name: Add a user
  cisco.aci.aci_aaa_user:
    host: apic
    username: admin
    password: SomeSecretPassword
    aaa_user: dag
    aaa_password: AnotherSecretPassword
    expiration: never
    expires: no
    email: dag@wieers.com
    phone: 1-234-555-678
    first_name: Dag
    last_name: Wieers
    state: present
  delegate_to: localhost
```

Modules

Used by tasks or playbooks

- Always use the fully qualified name for the module
- The modules require values assigned to the parameters that define how you wish to configure ACI
- Documentation provides details as to default values and required values
- No programing knowledge required. Just data structure build out.

Namespace | Collection Name | Module Name

```
- name: Create a Bridge Domain
  cisco.aci.aci_bd:
    host: 10.1.1.1
    username: admin
    password: password
    tenant: prod
    vrf: prod_vrf
    bd_type: ethernet
    bd: web_servers
    arp_flooding: yes
    l2_unknown_unicast: flood
    validate_certs: no
    state: present
  delegate_to: localhost
```

Parameter | Value

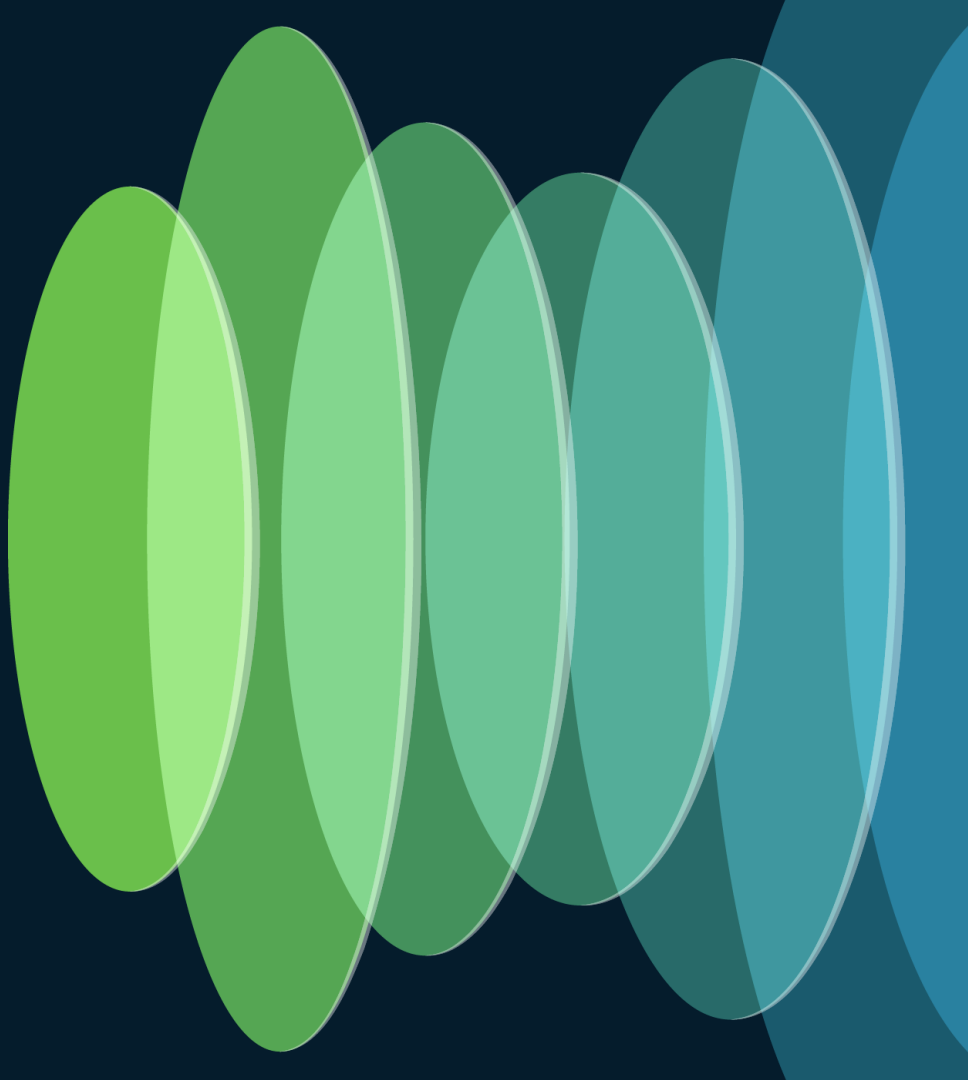
Ansible Collection Naming - Modules

- Uses Fully Qualified Collection Name
  - Name Space - Functional content category
  - Collection Name - Characteristics of the collection content
  - Module Name – Name of the module
- Best practice is to always use full qualified name, even for core modules
- Example – ACI Collection Tenant Module

```
cisco.aci.aci_bd
```

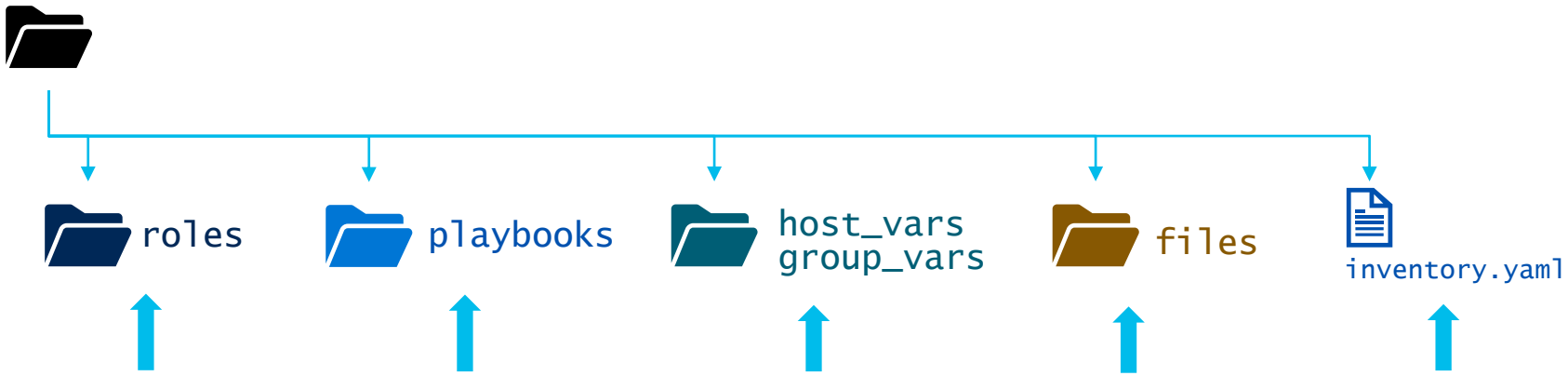
Name Space | Collection Name | Module Name

Ansible Concepts



Ansible Directory Structure

Best Practice for growth!



How to do it!

What to do!

Data to do it!

Stuff to do it!

Who to do it to!

Ansible Data Structures (YAML)

YAML Ain't Markup Language

- Human Readable Data Serialization Language
- Used in plays, playbooks and inventory files
- Best practice is to use a software focused text editor (e.g. Notepad++) or IDE (e.g. VSCode) with language assistant support of YAML data-structures.
  - Indentation is very important, and the proper editor will simplify this for you

Microsoft VSCode

ATOM

PyCharm

Eclipse

Notepad++

Ansible Roles

How to do it!

- Roles are content directories that are structured in a conventional way to enable simple reuse
- Roles let you automatically load related vars, files, tasks, handlers, and other Ansible artifacts based on a **known** file structure.
- This allows for better data organization in your repository.
- You utilize roles to combine tasks the complete and objective.

In this example we are creating a role that will configure access policy VLAN pools: `ansible-galaxy init ap-vlans`

```
❯ ansible-galaxy init ap-vlans
- Role ap-vlans was created successfully
 at …/brkdcn-2906 via 🐍 pyenv brkdcn-2906 (brkdcn-2906)
❯ listt ap-vlans
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ▷ ap-vlans
.rw-r--r-- 1.3Ki rmuller staff 25 Jan 13:04 ├── ⓜ README.md
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ files
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ templates
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ vars
.rw-r--r--    29 rmuller staff 25 Jan 13:04 │   └── ⚙ main.yml
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ tasks
.rw-r--r--    30 rmuller staff 25 Jan 13:04 │   └── ⚙ main.yml
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ tests
.rw-r--r--    67 rmuller staff 25 Jan 13:04 │   ├── ⚙ test.yml
.rw-r--r--    11 rmuller staff 25 Jan 13:04 │   └── ▯ inventory
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ meta
.rw-r--r-- 1.6Ki rmuller staff 25 Jan 13:04 │   └── ⚙ main.yml
drwxr-xr-x     - rmuller staff 25 Jan 13:04 ├── ▷ defaults
.rw-r--r--    33 rmuller staff 25 Jan 13:04 │   └── ⚙ main.yml
drwxr-xr-x     - rmuller staff 25 Jan 13:04 └── ▷ handlers
.rw-r--r--    33 rmuller staff 25 Jan 13:04     └── ⚙ main.yml
```

```
% ansible-galaxy init <role-name>
```

Ansible Playbooks

What to do!

- Playbooks define the set of actions that you want Ansible to complete.
- Can contain specific tasks or reference roles that contain the tasks
  - Best practice is to use roles!

Example playbook with roles:

playbooks/east-fabric/access-policies.yaml

```yaml
---
- hosts: east_fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  ignore_errors: false

  roles:
    - roles/ap-vlans
```

Ansible Inventory

Who to do it to!

- Ansible inventory allows you to build data structures that correlate host specific variables
- Allows for grouping, variable inheritance to organize your ACI fabric APICs
- Two formats are common: INI and YAML. Best practice is to use YAML (*less confusing*)

east_fabric.yaml

```yaml
---
east_fabric:
  vars:
    username: admin
    password: cisco.123
  hosts: 10.0.226.41
```

JINJA “type” variables

Variable substitution

- Ansible uses Jinja2 to enable dynamic expressions and access to variables and facts
- Defined by curly backets `"{{ }}"` inside quotes.
- Similar to how JINJA2 works

```
vars:
  username: "john"
  password: "doe"


- name:
  ansible.builtin.copy:
    username:"{{username}}"
    password:"{{password}}"
```

Putting it all together

playbooks

File: playbooks/east-fabric/access-policies.yaml

```yaml
---
- hosts: east_fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  ignore_errors: false

  roles:
    - roles/ap-vlans
```

inventory

File: east_fabric.yaml

```yaml
---
east_fabric:
  vars:
    username: admin
    password: cisco.123
  hosts: 10.0.226.41
```

roles

File: roles/ap-vlans/tasks/main.yml

```yaml
---
# tasks file for ap-vlans

- name: Create Engineering VLAN Pool
  cisco.aci.aci_vlan_pool:
    host: "{{ inventory_hostname }}"
    username: "{{username}}"
    password: "{{password}}"
    pool: "eng_vlan_pool"
    pool_allocation_mode: "static"
    description: "(Ans) Engineering Server VLAN Pools"
    state: present
    validate_certs: no
    use_ssl: yes
  delegate_to: localhost
```

Executing Ansible

```
ansible-playbook -i <inventory file> <playbook file>
```

Details for ACI

Playbooks

Structure for ACI

For ACI we set `gather_facts` to false as we don't need for Ansible to connect to APIC to get any host data. Ansible uses the REST interface.

For ACI connection is `local`, as the computer that is executing the automation starts the connection `local` to the ACI fabric.

Control if faults continue or stop

```yaml
---
- hosts: east-fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  ignore_errors: false
  module_defaults:
      group/cisco.aci.aci:
        hostname: "{{inventory_hostname}}"
        username: '{{ lookup("env", "APIC_USERNAME") }}'
        password: '{{ lookup("env", "APIC_PASSWORD") }}'
        validate_certs: no
        use_ssl: yes

  roles:
    - roles/ap-vlans
    - roles/ap-domains
    - roles/ap-aep
```

Playbooks

Structure for ACI

NEW

Module Defaults allows to pass common parameters to modules. This avoids having to place *repetitive parameters* (like credentials) in all modules!

- Set **validate_certs** to no, and **`use_ssl`** to yes for self-signed cert-based HTTPS connection to the fabric
- Best practice is to always start using environment variables. ***Never add credentials*** inside GIT repository

The roles that this playbook will execute

In Ansible order matters! You can't create a physical domain that points to a VLAN Pool without first creating the pool

```yaml
---
- hosts: east-fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  ignore_errors: false
  module_defaults:
    group/cisco.aci.all:
      hostname: "{{inventory_hostname}}"
      username: '{{ lookup("env", "APIC_USERNAME") }}'
      password: '{{ lookup("env", "APIC_PASSWORD") }}'
      validate_certs: no
      use_ssl: yes

  roles:
    - roles/ap-vlans
    - roles/ap-domains
    - roles/ap-aep
```

Tasks in Roles

Structure for ACI

Task name

These values define how the VLAN Pool will be configured

State is **`present`** for *creation* and **`absent`** for *deletion*

Set **validate_certs** to no, and **`use_ssl`** to yes for self-signed cert-based HTTPS connection to the fabric

namespace collection name module

```
---
# tasks file for ap-vlans


- name: Create Engineering VLAN Pool
  cisco.aci.aci_vlan_pool:
    pool: "eng_vlan_pool"
    pool_allocation_mode: "static"
    description: "(Ans) Engineering Server VLAN Pools"
    state: present
  delegate_to: localhost
```

Authentication

Best Practices

Username & Password

- Method works with both ACI and NDO
- Easiest approach after ACI 5.x HTTP throttle changes
- Important to avoid `username` and `password` stored inside source code repository
  - Very hard to remove once added!
- Ansible Vault is the most secure, but you can get started easily with `environment variables`.

Certificate Based

- Used in releases prior to ACI 5.x due to HTTP interface throttle
  - In ACI 5.x and higher interface throttle is configurable option in ACI
- Ansible Vault can be used to store the key.
- Certificate based not an option for MSO today.
- Requires a local user on APIC
  - Configured with proper user role and security domain

Hidden slide has details on ENV vars

Using Environment Variables

- Instead of inserting credentials that are very difficult to remove from an SCM (GIT) you can use environment variables.
- Set environment variable before ansible-playbook execution

```yaml
---
- name: East Fabric Access Policies
  hosts: east-fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  ignore_errors: false
  module_defaults:
      group/cisco.aci.all:
        hostname: "{{ inventory_hostname }}"
        username: '{{ lookup("env", "APIC_USERNAME") }}'
        password: '{{ lookup("env", "APIC_PASSWORD") }}'
        validate_certs: no
        use_ssl: yes

  roles:
    - roles/ap-vlans
    - roles/ap-domains
    - roles/ap-aep
```

bash / zsh

```
% export APIC_USERNAME="admin"
% export APIC_PASSWORD="password"
```

ACI REST Fallback Module

How to configure ACI when a module is missing

- The module `aci_rest` allows passing an ACI structured object when a module isn't available.
- This makes it possible that Ansible can accomplish 100% configuration of ACI

```
- name: Create Route Map for L3out (rtctrlProfile)
  cisco.aci.aci_rest:
    path: /api/node/mo/uni/tn-{{item.tenant}}/out-{{item.l3out}}/prof-{{item.name}}.json
    method: post
    content:
      {
        "rtctrlProfile":
          {
            "attributes":
              {
                "dn": "uni/tn-{{item.tenant}}/out-{{item.l3out}}/prof-{{item.name}}",
                "name": "{{item.name}}",
                "descr": "{{item.description}}",
                "status": "created,modified",
              },
            "children": [],
          },
      }
  delegate_to: localhost
  loop: "{{all_l3out_route_maps}}"
  when: all_l3out_route_maps is defined
  tags:
    - never
    - create
```

An example

Non-Optimal

- In the previous example we “hard coded” some values to create a VLAN Pool.
- This would require that we create a new task for every single VLAN pool to be configured.
  - Not optimal for repetition
- There is a better approach through reference and iteration!

hard coded

```
---
# tasks file for ap-vlans


- name: Create Engineering VLAN Pool
  cisco.aci.aci_vlan_pool:
    pool: "eng_vlan_pool"
    pool_allocation_mode: "static"
    description: "(Ans) Engineering Server VLAN Pools"
    state: present
  delegate_to: localhost
```

Variable lists

Looping through data

- `Lists` (also known as `arrays`) are a ***sequential set*** of values.
- These can contain `dictionaries` (also known as **objects**).
- This allows you to **`reference`** specific items inside of the task and **`iterate`** over these in a repetitive way

List of four objects

```
vlan_pools:
  - vlan_pool_name: "eng_vlan_pool"
    vlan_pool_description: "(Ans)Eng VLAN Pool"
    vlan_pool_mode: "static”

  - vlan_pool_name: "mkt_vlan_pool"
    vlan_pool_description: "(Ans)Mkt VLAN Pool"
    vlan_pool_mode: "static”

  - vlan_pool_name: "hr_vlan_pool"
    vlan_pool_description: "(Ans)HR VLAN Pools"
    vlan_pool_mode: "static”

  - vlan_pool_name: "sales_vlan_pool"
    vlan_pool_description: "(Ans)Sales VLAN Pools"
    vlan_pool_mode: "static”
```

Iteration explained

Looping through data

roles/ap-vlans/*tasks*/main.yaml

```yaml
---
# tasks file for ap-vlans

- name: Create VLAN Pools
  cisco.aci.aci_vlan_pool:
    pool: “{{item.vlan_pool_name}}”
    pool_allocation_mode: “{{item.vlan_pool_mode}}”
    description: “{{item.vlan_pool_description}}”
    state: present
  delegate_to: localhost
  loop: "{{vlan_pools}}"
  when: vlan_pools is defined
```

roles/ap-vlans/*vars*/main.yaml

```yaml
---
# vars file for ap-vlans

vlan_pools:
  - vlan_pool_name: “eng_vlan_pool"
    vlan_pool_description: "(Ans)Eng VLAN Pool"
    vlan_pool_mode: "static”

  - vlan_pool_name: ”mkt_vlan_pool"
    vlan_pool_description: "(Ans)Mkt VLAN Pool"
    vlan_pool_mode: "static”

  - vlan_pool_name: ”hr_vlan_pool"
    vlan_pool_description: "(Ans)HR VLAN Pools"
    vlan_pool_mode: "static”

  - vlan_pool_name: ”sales_vlan_pool"
    vlan_pool_description: "(Ans)Sales VLAN Pools"
    vlan_pool_mode: "static”
```

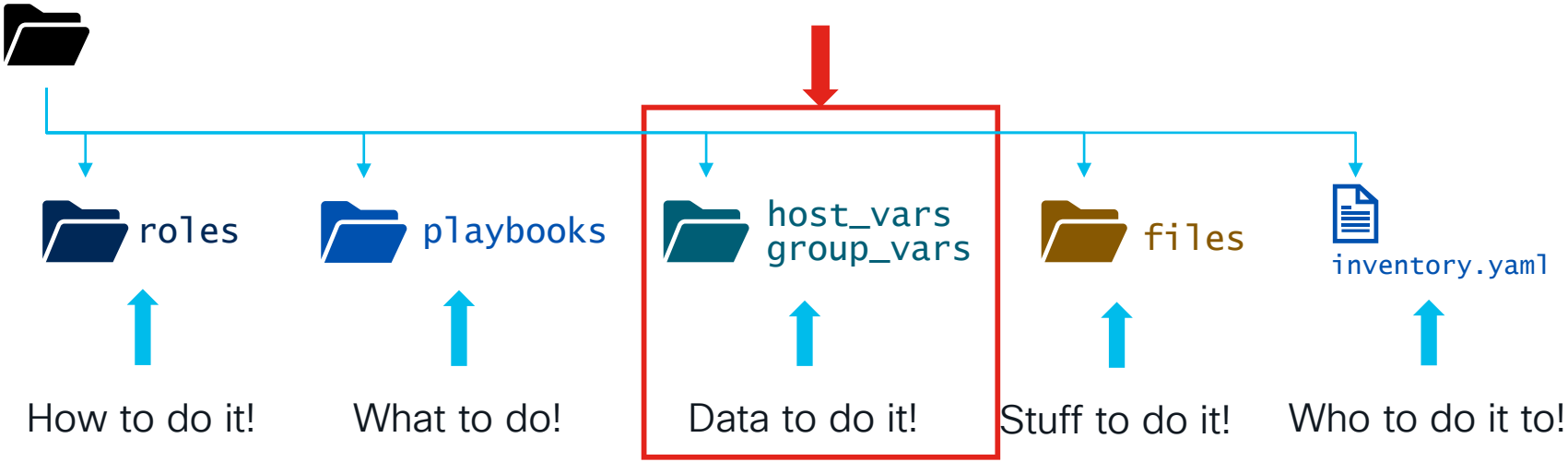
Executing the playbook

A word about variables

Better variables

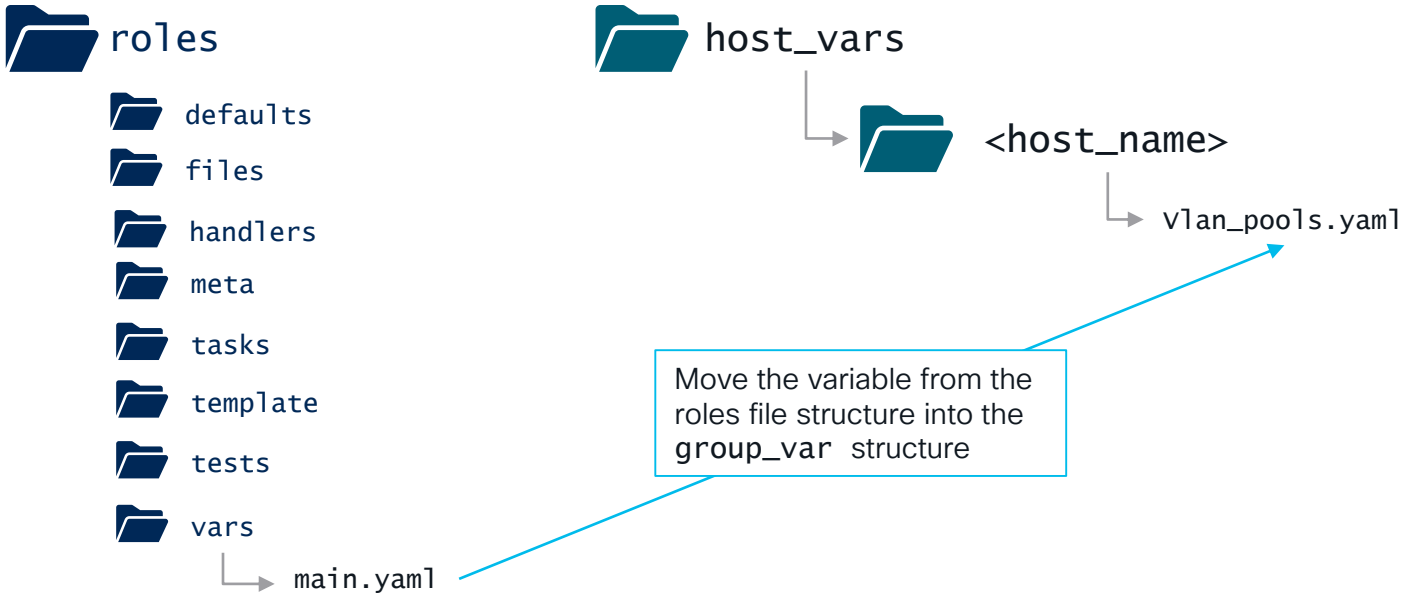
Placement matters!

- Including the variables with the role can result in role duplication
- A better approach is to move the variables to a location that can be structured with the inventory for better organization



Variable Hierarchy

A clean way to organize data



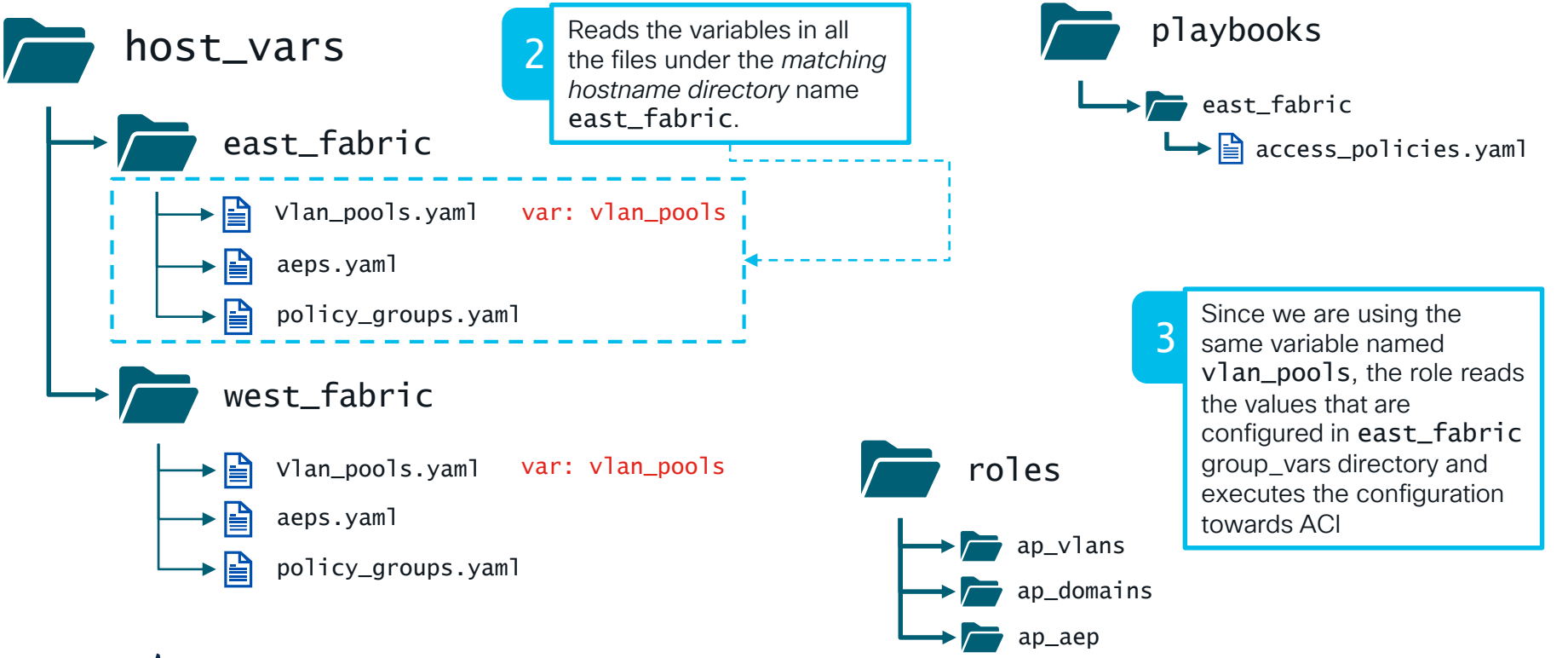
The links between locations

Managed by variable **precedence**

1 Reads the inventory and playbook. Finds that we are referencing `east_fabric`

2 Reads the variables in all the files under the *matching hostname directory* name `east_fabric`.

3 Since we are using the same variable named `vlan_pools`, the role reads the values that are configured in `east_fabric` group_vars directory and executes the configuration towards ACI



Hidden slide details variable precedence

Ansible Variable Precedence

Placement matters

- Ansible provides variable precedence, which is important when you build your data structure.
- This allows for having some default behaviour that is then changed by just including in higher precedence.
- Using the `group_vars` folder tied to inventory is very useful.

Higher

- extra vars via CLI (for example, -e "user=my_user")
- include params
- role (and include_role) params
- set_facts / registered vars
- include_vars
- task vars (only for the task)
- block vars (only for tasks in block)
- role vars (defined in role/vars/main.yml)
- play vars_files
- play vars_prompt
- play vars
- host facts / cached set_facts
- playbook host_vars/*
- inventory host_vars/*
- inventory file or script host vars
- playbook group_vars/*
- inventory group_vars/*
- inventory group_vars/all
- inventory file or script group vars
- role defaults (defined in role/defaults/main.yml)
- command line values (for example, -u my_user, these are not variables)

Lower

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html

Playbooks

playbooks/east-fabric/access-policies.yaml

```yaml
---
- hosts: east_fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  module_defaults:
    group/cisco.aci.aci:
      hostname: "{{ inventory_hostname }}"
      username: "{{ username }}"
      password: "{{ password }}"
      validate_certs: no
      use_ssl: yes
  roles:
    - roles/ap-vlans
```

host_vars

host_vars/east_fabric/ap-vlan-pools.yaml

```yaml
---
vlan_pools:
  - vlan_pool_name: "eng_vlan_pool"
    vlan_pool_description: "(ANS)Eng VLAN Pool"
    vlan_pool_mode: "static"
  - vlan_pool_name: "mkt_vlan_pool"
    vlan_pool_description: "(ANS)Mkt VLAN Pool"
    vlan_pool_mode: "static"
  - vlan_pool_name: "hr_vlan_pool"
    vlan_pool_description: "(ANS)HR VLAN Pool"
    vlan_pool_mode: "static"
  - vlan_pool_name: "sales_vlan_pool"
    vlan_pool_description: "(ANS)Sales VLAN Pool"
    vlan_pool_mode: "static"
```

```
% ansible-playbook -i inventory/east_fabric playbooks/east-fabric/access-policies.yaml
```

inventory

east_fabric.yaml

```yaml
---
east_fabric:
  vars:
    username: '{{ lookup("env", "APIC_USERNAME") }}'
    password: '{{ lookup("env", "APIC_PASSWORD") }}'
  hosts: '{{ lookup("env", "APIC_IP") }}'
```

Roles

roles/ap-vlans/tasks/main.yaml

```yaml
---
# tasks file for ap-vlans

- name: Create VLAN Pools
  cisco.aci.aci_vlan_pool:
    pool: "{{ item.vlan_pool_name }}"
    pool_allocation_mode: "{{ item.vlan_pool_mode }}"
    description: "{{ item.vlan_pool_description }}"
    state: present
  delegate_to: localhost
  loop: "{{ vlan_pools }}"
  when: vlan_pools is defined
```

Executing the playbook

ACI 6.x Ansible indicators

Policies

- Pools
  - VLAN
    - Contiv (Static Allocation)
    - Corning (Dynamic Allocation)
    - inbMgmt (Static Allocation)
    - l2out (Static Allocation)
    - msite (Static Allocation)
    - p25tes (Static Allocation)
    - rich (Dynamic Allocation)
  - Multicast Address
  - VSAN
  - VSAN Attributes
  - VXLAN

Pools - VLAN

VLAN | Operational

Name	Allocation Mode	Encap Blocks	Description
Contiv	Static Allocation	[500-700] (Static Allocation)	
Corning	Dynamic Alloca...	[1000-1500]	
inbMgmt	Static Allocation	[10-19] [1] [322] [323] [336]	
l2out	Static Allocation	[326] (Static Allocation) [336]	
msite	Static Allocation	[4]	

Last Login Time: 2023-01-31T23:52 UTC+00:00

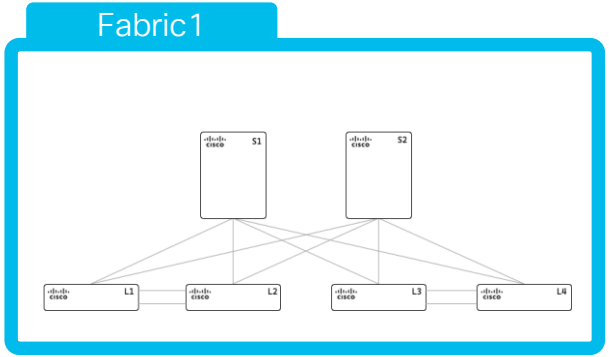
Current System Time: 2023-01-31T23:55 UTC+00:00

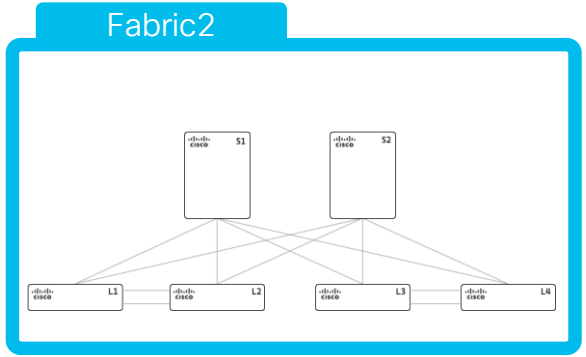
Use Case!

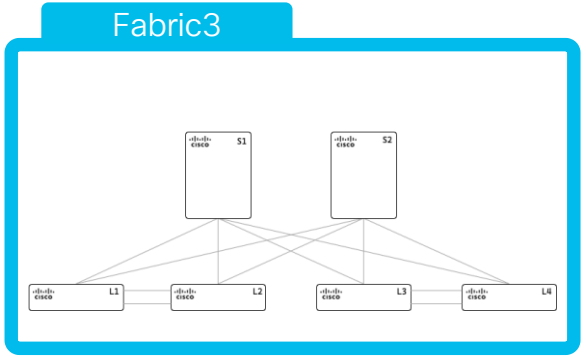
Legacy conversion to ACI

- Three fabrics had to be migrated from legacy Nexus7k/5k to ACI
- Each fabric independent from each other
- MOPS converted by hand to Ansible.

https://github.com/rafmuller/brkdcn-ans-terra/tree/develop/ansible







Demo

Questions?

What is your path?

I got this!

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

I need help?!

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

Checkout
Services as Code
@ **World of Solutions**

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

Thank you