cisco Live!

Let's go

The bridge to possible

Introduction to Infrastructure as Code for ACI with Ansible and Terraform

Thomas Renzy, Technical Leader CX @ThomasRenzy
Rafael Muller, Principal Engineer CX @rafaeljmuller

Agenda

- What is Infrastructure as Code?
- Infrastructure as Code with Ansible
- Infrastructure as Code with Terraform
- Next steps

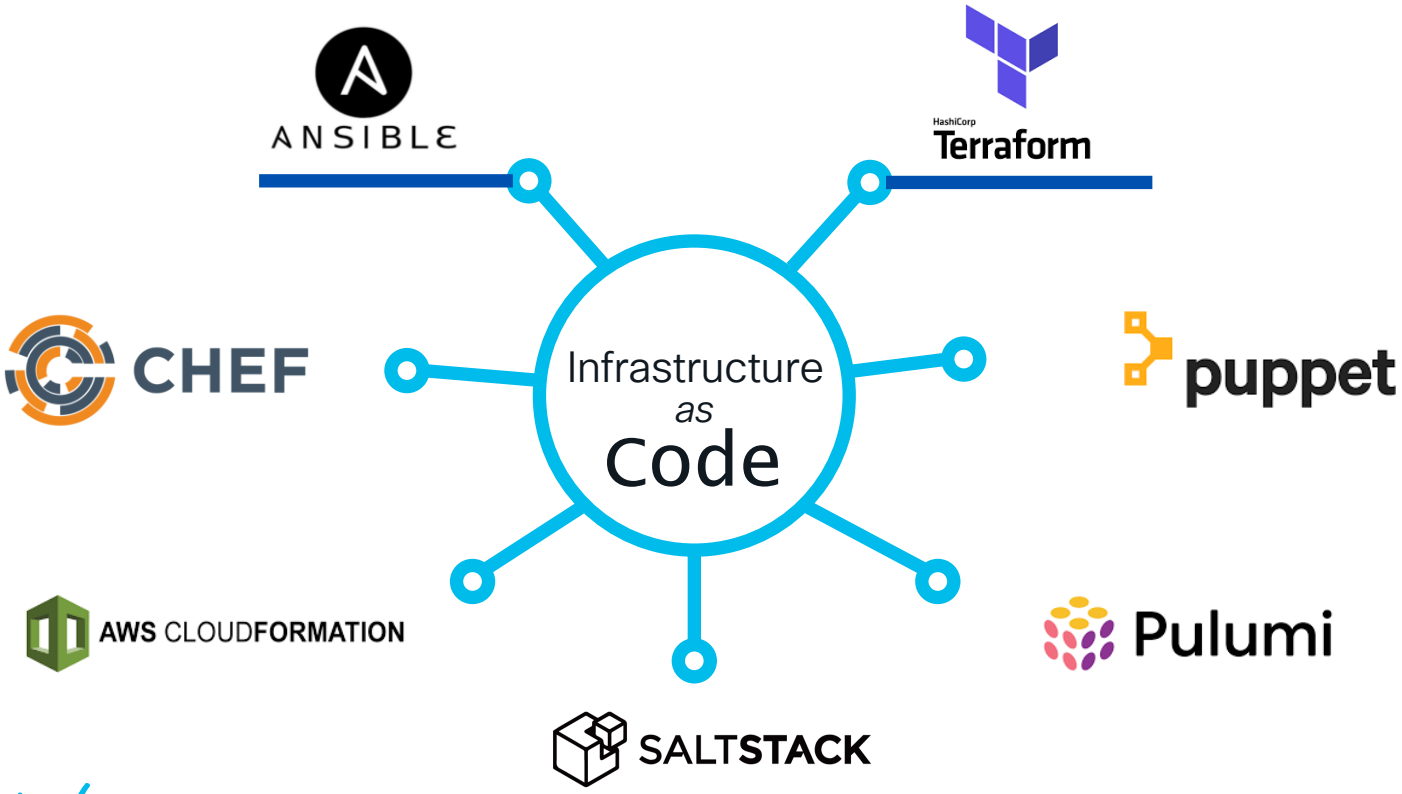
What is Infrastructure as Code?

What is Infrastructure as Code(IaC)?

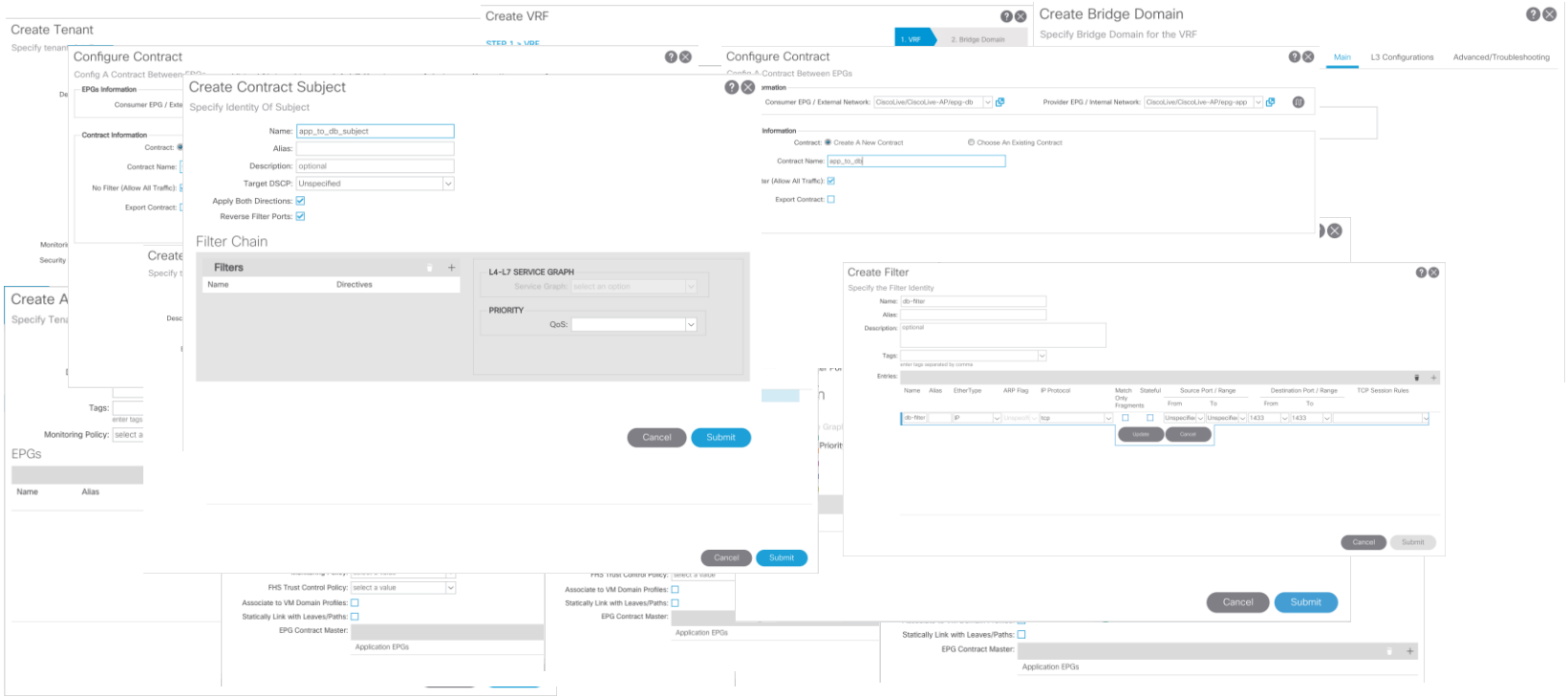
- Managing infrastructure can be tedious
- Network operators connect to devices and make changes to the configuration.
  - CLI – “finger net”
  - Web browser – “point and click” aka “ClickOps”
- Most think of building/managing Cloud Infrastructure
- Define ***intended state of infrastructure*** should be.
  - Automation tools reads & applies changes to devices to *match the intended state*.

The management & provisioning of computer infrastructure through code and data structures instead of direct device management.

Infrastructure as Code Tools



“ClickOps”– APIC GUI



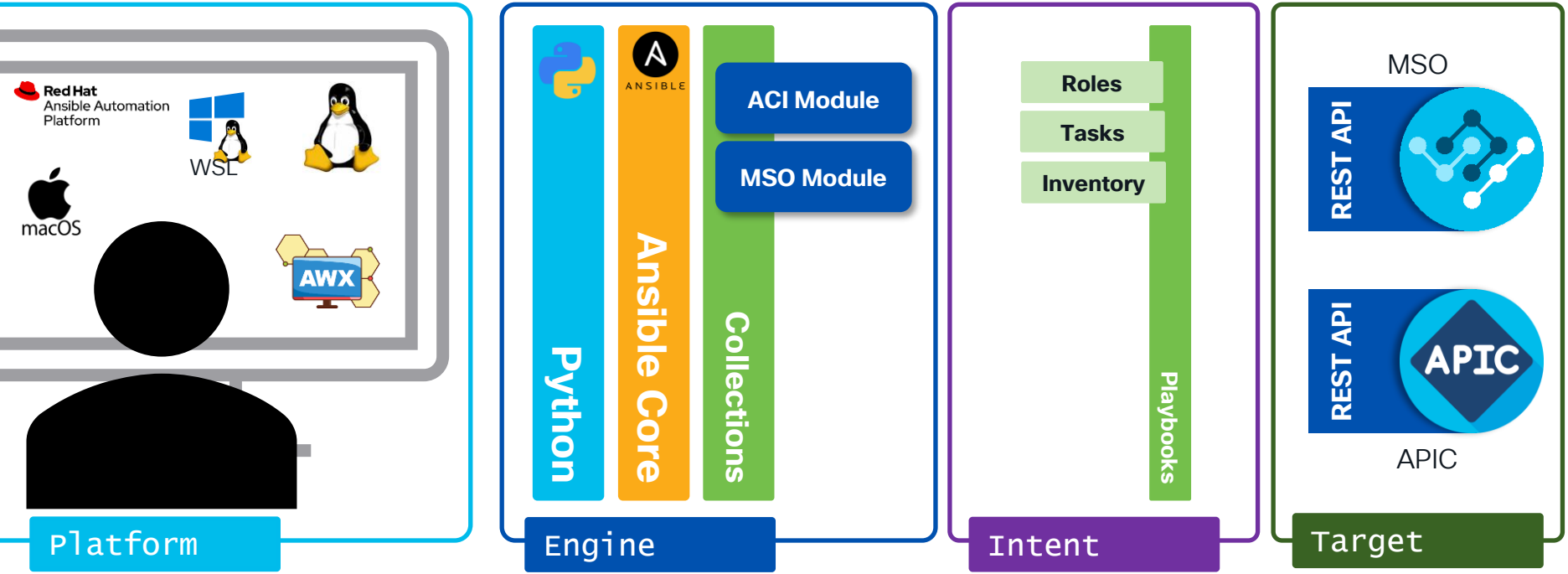
Infrastructure as Code with Ansible

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.

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 3.9.11
```

2. create virtual-environment

```
% pyenv virtualenv 3.9.11 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 by you
- 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 much 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

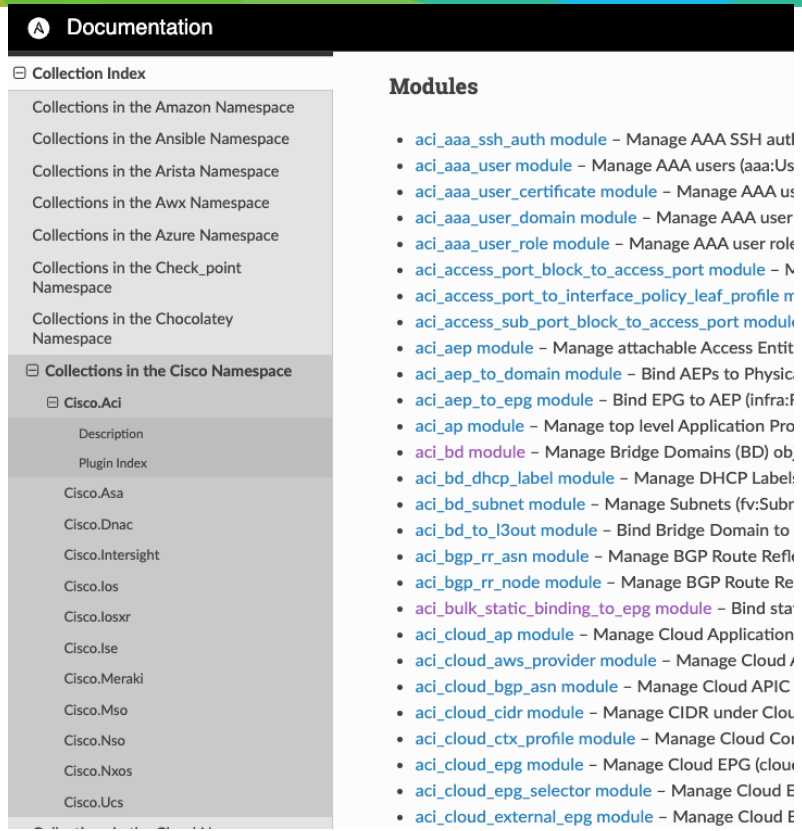
```
❯ ansible-galaxy collection install cisco.aci cisco.mso
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/download/cisco-aci-2.3.0.tar.gz to /Users/rmuller/.ansible/tmp/ansible-local-464
00clazzrla/tmpn6m0ecmc/cisco-aci-2.3.0-6vczo8j6
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 **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 and more
- Actively maintained with regular cadence that increases module count and capability



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

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

- 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

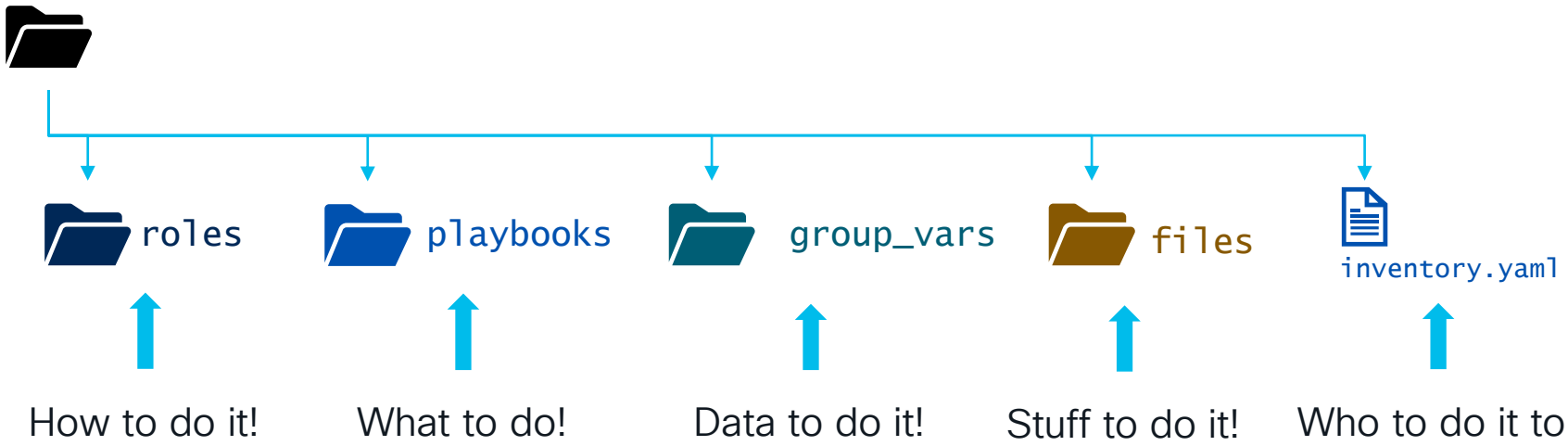
```
cisco.aci.aci_bd
```

Name Space | Collection Name | Module Name

Ansible Concepts

Ansible Directory Structure

Best Practice for growth!



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:

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
```

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*)

Example inventory file:

File: 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
```

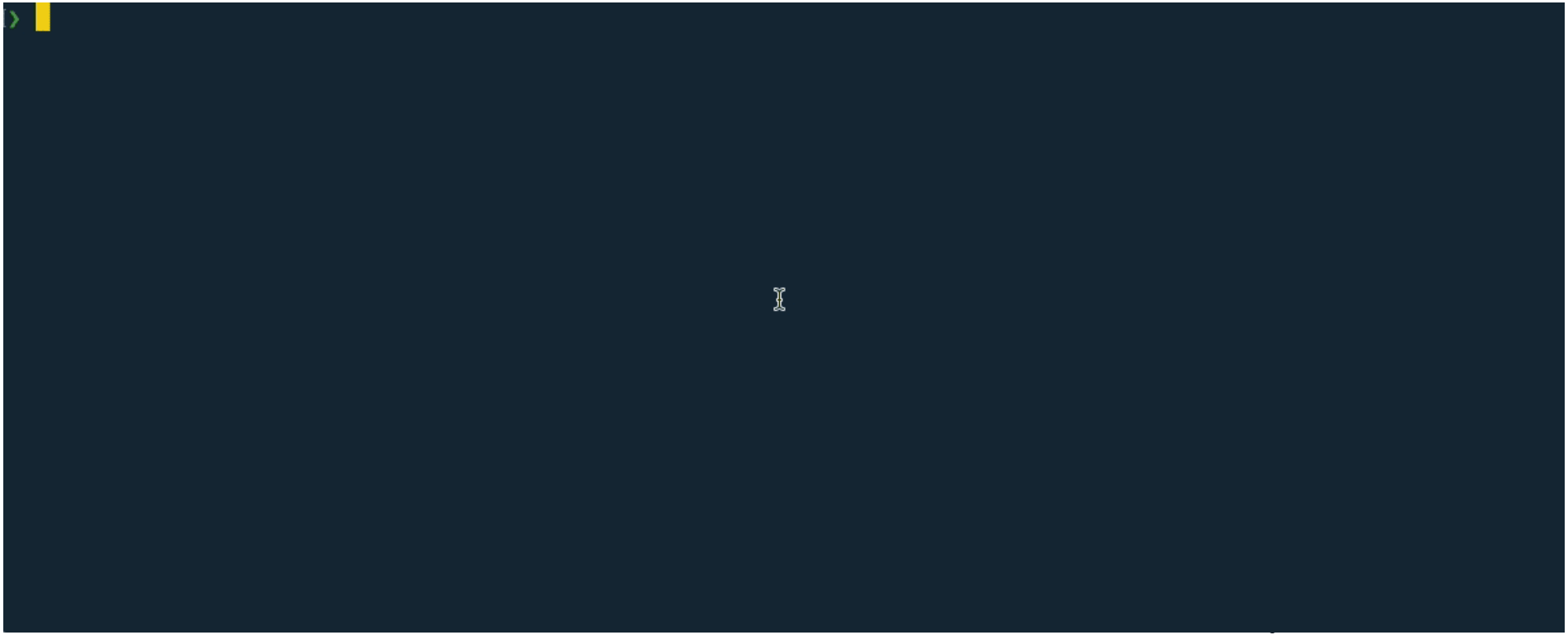
roles

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

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

```
---
- hosts: east-fabric
  gather_facts: false
  connection: local
  any_errors_fatal: true
  ignore_errors: false



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

Tasks in Roles

Structure for ACI

Task name

For each module we must pass the **`host`**, **`username`** and **`password`**

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:
    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
```

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

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

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


- name: Create Engineering VLAN Pool
  cisco.aci.aci_vlan_pool:
    host: "{{ inventory_hostname }}"
    username: '{{ lookup("env", "APIC_USERNAME") }}'
    password: '{{ lookup("env", "APIC_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
```

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:
    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
```

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:
    host: "{{ inventory_hostname }}"
    username: "{{username}}"
    password: "{{password}}"

    pool: “{{item.vlan_pool_name}}”
    pool_allocation_mode: “{{item.vlan_pool_mode}}”
    description: “{{item.vlan_pool_description}}”
    state: present

    validate_certs: no
    use_ssl: yes
  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

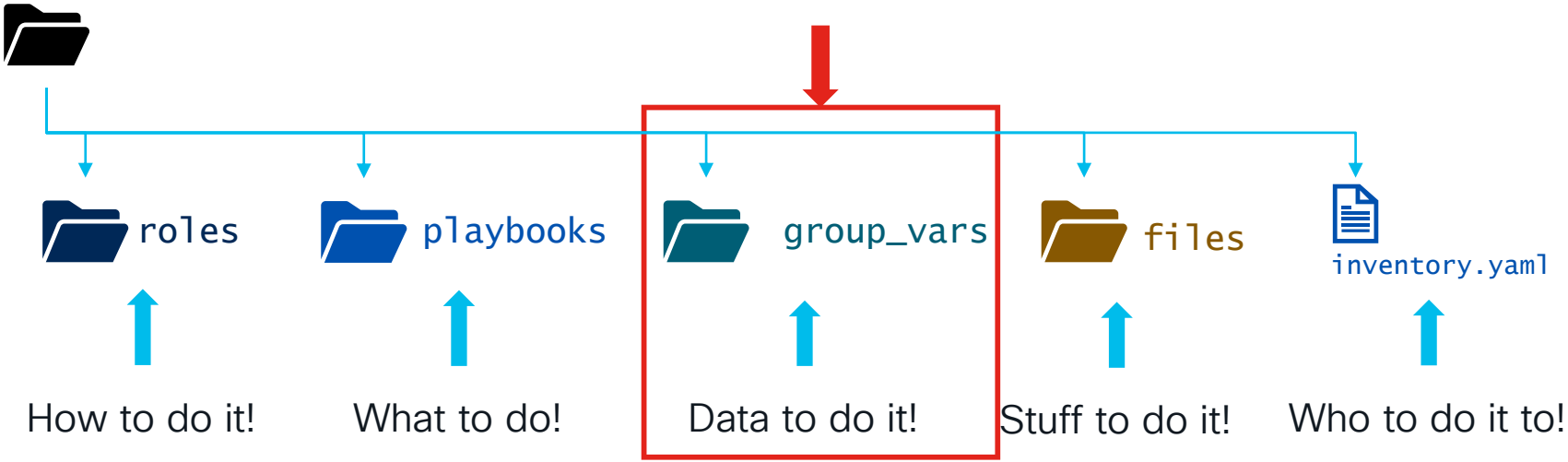
```
>
```

A word about variables

Better variables

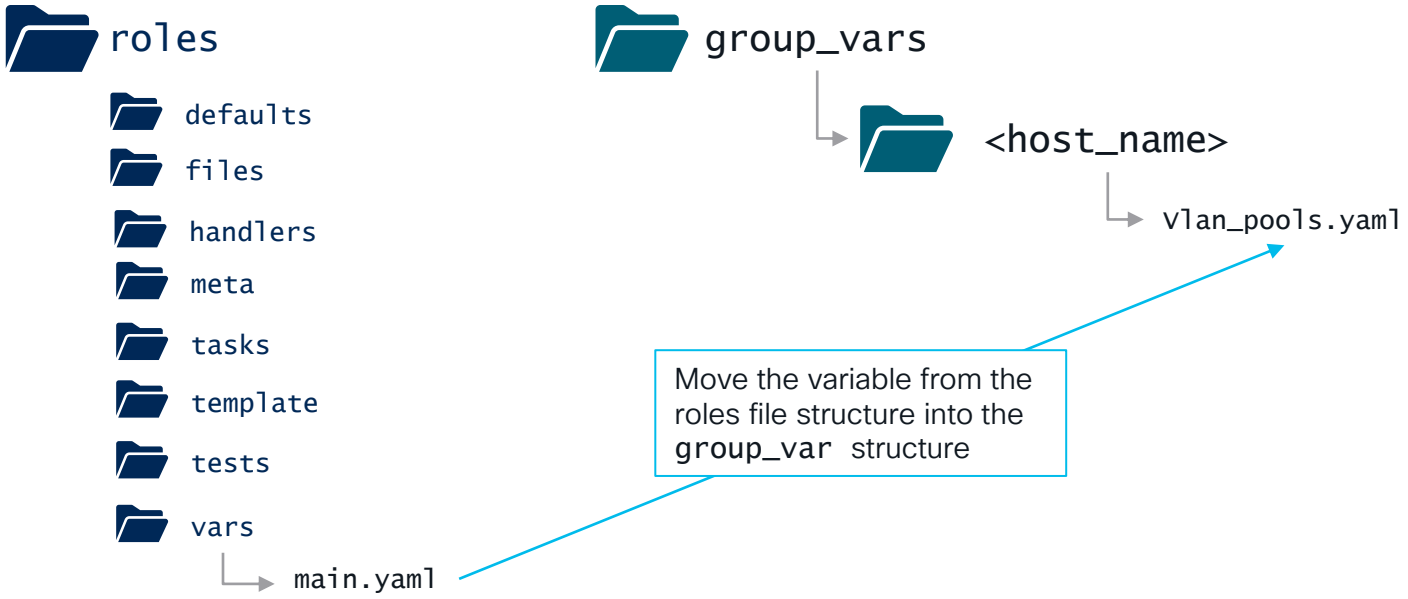
Placement matters!

- 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

- `group_vars`
  - `east_fabric`
    - `vlan_pools.yaml` var: `vlan_pools`
    - `aeps.yaml`
    - `policy_groups.yaml`
  - `west_fabric`
    - `vlan_pools.yaml` var: `vlan_pools`
    - `aeps.yaml`
    - `policy_groups.yaml`
- `playbooks`
  - `east_fabric`
    - `access_policies.yaml`
- `roles`
  - `ap_vlans`
  - `ap_domains`
  - `ap_aep`

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

Putting it all together

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

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
```

group_vars

File: group_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 Pools"
    vlan_pool_mode: "static"

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

roles

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

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

- name: Create VLAN Pools
  cisco.aci.aci_vlan_pool:
    host: "{{ inventory_hostname }}"
    username: "{{username}}"
    password: "{{password}}"
    pool: "{{item.vlan_pool_name}}"
    pool_allocation_mode: "{{item.vlan_pool_mode}}"
    description: "{{item.vlan_pool_description}}"
    state: present
    validate_certs: no
    use_ssl: yes
  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

Infrastructure as Code with Terraform

What is Terraform?

- Open Source
- Infrastructure Provisioning
- Single Binary (Windows, Mac, Linux, Solaris, FreeBSD)
- Declarative and Stateful
- HashiCorp Configuration Language (HCL)
- APIC/NDO REST API Interaction
- No programming knowledge required

Terraform Concepts

Terraform

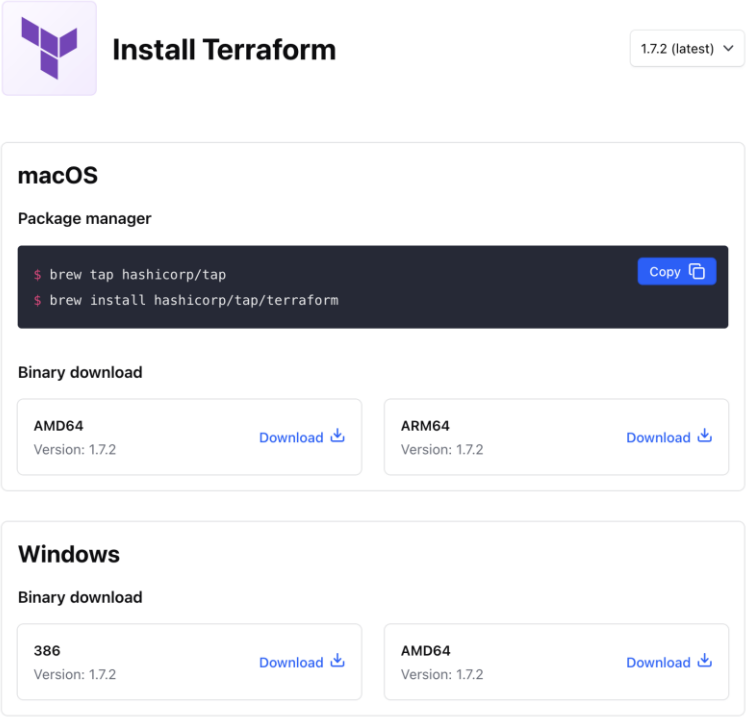
(version 1.7.2 - latest)

- Runs as single binary (Core)
- Command line executable
  - Reading configuration files
  - State Management
  - Graph
  - Plan execution

```
> terraform --version
Terraform v1.7.2
on darwin_arm64
```

Installation

https://developer.hashicorp.com/terraform/downloads



Terraform Providers

- Terraform Binary doesn't know ACI/NDO
- Relies on specific plugins
  - Downloaded Dynamically via initialization (via `terraform init` command)
- Providers Understand API interactions
  - APIC and MSO REST API calls

```
terraform {
  required_providers {
    aci = {
      source = "CiscoDevNet/aci"
    }
  }
}
```

Types of Terraform Providers

Official

Owned & maintained by HashiCorp

Ex. AWS, Azure, GCP

Partner

Owned & maintained by partners.

Ex. ACI, MSO, ASA

Community

Published by individual groups or maintainers in the community

```
terraform {
  required_providers {
    aci = {
      source = "CiscoDevNet/aci”
      version = “2.11.1”
    }
  }
}
```

```
terraform {
  required_providers {
    mso = {
      source = "CiscoDevNet/mso"
    }
  }
}
```

https://registry.terraform.io/providers/CiscoDevNet/aci/latest

Terraform Provider configuration (ACI)

Terraform Block

Provider definition

Provider Block

APIC URL

```
terraform {
  required_version ">= 1.5.0"
  required_providers {
    aci = {
      source = "CiscoDevNet/aci”
      version = “2.11.1”
    }
  }
}

provider "aci" {
  # cisco-aci user name
  username = "tform"
  # cisco-aci signature based cert
  private_key = "tfcert.key"
  cert_name   = "tfcert.crt"
  # cisco-aci url
  url      = "https://172.31.2.31/"
  insecure = true
}
```

Terraform version

Registry namespace

Provider version

Authentication

HTTP API request

Terraform Resources & Data Sources

Resources

- Specific to a given provider (ACI/MSO)
- apply/destroy/modifies Infrastructure
- Accepts arguments
- Describes your **intent** for infrastructure

```
resource "aci_tenant" "terraform_tenant” {
  name        = var.tenant_name
  description = "Created with Terraform"
}
```

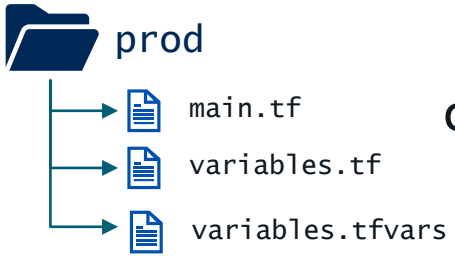
Data Sources

- Allow data to be fetched or computed for use elsewhere in Terraform configuration
- Terraform apply/destroy does not modify data source

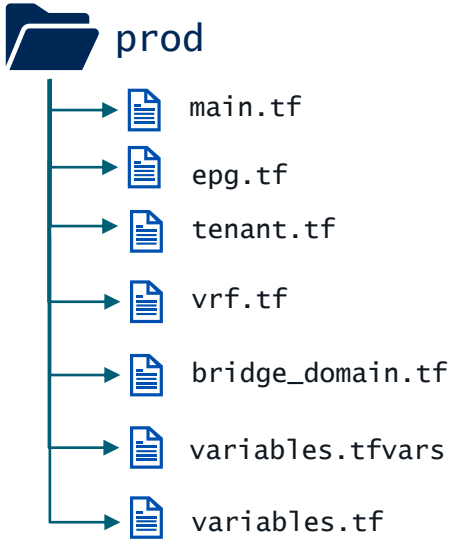
```
data "mso_site" "sf_site" {
   name = "San Francisco"
}

data "mso_site" "ny_site" {
   name = "New York"
}
```

Terraform Plans/Configuration Files

- Collection of HCL instructions
  - What you want to provision (intent)
- .tf extension
  - Terraform scans directory
    - Directory that terraform is run in
  - Can be in a singular file – main.tf
  - Can be broken up into smaller *.tf



or



Terraform Configuration Example

Used in state file

Resource type
(From Provider)

Name of the resource
Must be unique

```
resource "aci_bridge_domain" "web-bd" {
  tenant_dn          = aci_tenant.terraform_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  name               = "web-bd"
}

resource "aci_subnet" "web_subnet" {
  parent_dn = aci_bridge_domain.web-bd.id
  ip        = "10.1.1.1/24"
}
```

Terraform Data Source Example

Data Source type
(From Provider)

Data Source name
Must be unique

```
data "mso_site" "sf_site" {
  name = "San Francisco"
}

data "mso_site" "ny_site" {
  name = "New York"
}

# Define an NDO Tenant between NY and SF
resource "mso_tenant" "tenant" {
  name = var.tenant_name
  display_name = var.tenant_name
  description = "This tenant was created by Terraform”
  site_associations { site_id = data.mso_site.sf_site.id }
  site_associations { site_id = data.mso_site.ny_site.id }
}
```

When there isn’t a Resource – aci_rest_managed

- Manages Objects via REST API calls with no resource
- Can reconcile state information
- API calls captured via API Inspector/APIC GUI
- aci_rest/mso_rest

```
resource "aci_rest_managed" ”rest_tenant" {
  dn         = "uni/tn-REST"
  class_name = "fvTenant"
  content = {
    name  = ”REST"
    descr = ”Tenant built with REST"
  }
}
```

Terraform Registry – Documentation

ACI DOCUMENTATION

Filter

- aci provider
- AAA
- Access Policies
- Application Management
  - Resources
    - aci_application_epg
    - aci_application_profile
    - aci_bulk_epg_to_static_path
    - aci_endpoint_security_group
    - aci_endpoint_security_group_epg_selector
    - aci_endpoint_security_group_selector
    - aci_endpoint_security_group_tag_selector
    - aci_epg_to_contract
    - aci_epg_to_contract_interface
    - aci_epg_to_domain
    - aci_epg_to_static_path
    - aci_tenant

aci_application_epg

Manages ACI Application EPG

Example Usage

```
resource "aci_application_epg" "fooapplication_epg" {
    application_profile_dn  = aci_application_profile.app_profile_for_epg.id
    name                              = "demo_epg"
    description                     = "from terraform"
    annotation                      = "tag_epg"
    exception_tag                   = "0"
    flood_on_encap              = "disabled"
    fwd_ctrl                        = "none"
    has_mcast_source              = "no"
    is_attr_based_epg           = "no"
    match_t                           = "AtleastOne"
    name_alias                    = "alias_epg"
    pc_enf_pref                   = "unenforced"
    pref_gr_memb                    = "exclude"
    prio                              = "unspecified"
    shutdown                        = "no"
    relation_fv_rs_bd       = aci_bridge_domain.example.id
}
```

Argument Reference

ON THIS PAGE

- Example Usage
- Argument Reference
- Attribute Reference
- Importing

Report an issue

https://registry.terraform.io/providers/CiscoDevNet/aci/latest

Terraform State

- Terraform is stateful
  - Tracks objects it builds (terraform.tfstate)
  - Source of everything it knows about
- Stored inside working directory
  - Can use backend – AWS, Terraform Cloud
  - Do not modify state file directly


```
{
  "version": 4,
  "terraform_version": "1.7.0",
  "serial": 18,
  "lineage": "f7aa5662-7643-c475-830e-a76433b16ef2",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aci_tenant",
      "name": "env",
      "provider": "provider[\"registry.terraform.io/ciscodevnet/aci\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "annotation": "orchestrator:terraform",
            "description": "Created by Terraform",
```

Brownfield Infrastructure

- Import Infrastructure for Terraform to manage
  - `import` CLI command
  - Import blocks (Version 1.5)

```
terraform import aci_tenant.import_tenant_example uni/tn-tf_test_import
```

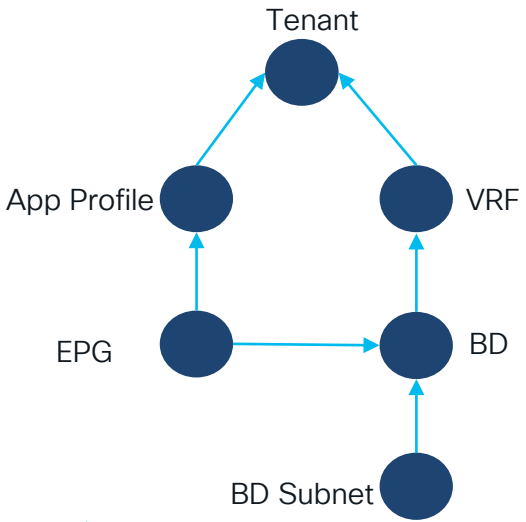
Import CLI command

```
# Import VRF under Tenant from APIC
import {
  id = "uni/tn-tf_test_import/ctx-tf_test_import_vrf"
  to = aci_vrf.import_vrf_example
}

# Import Tenant from APIC
import {
  id = "uni/tn-tf_test_import"
  to = aci_tenant.import_tenant_example
}
```

Import Block (import.tf)

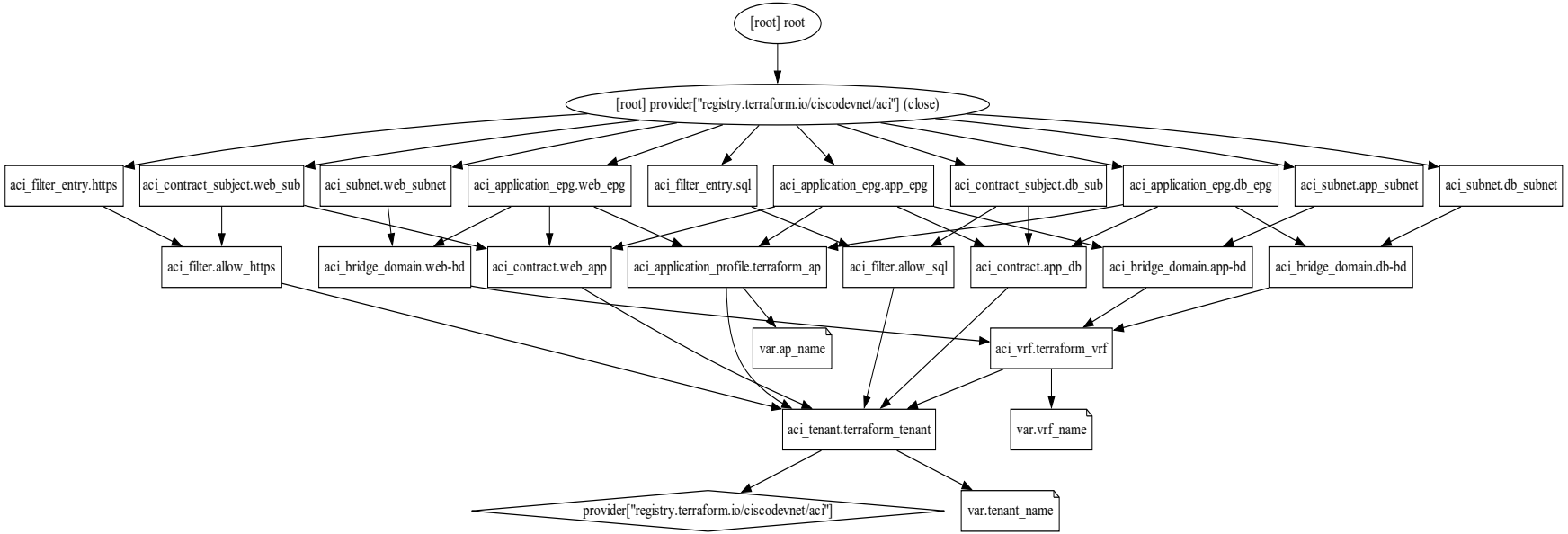
Terraform Dependency Mapping

- Uses Graphs to track of dependencies and correct order of deployment
- Builds a graph of relationships (Directional tree without loops)




```
resource "aci_bridge_domain" "bridge_domain1"
  {
      tenant_dn = aci_tenant.terraform_tenant.id
      relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
      name = "bridge_domain-1"
  }
```

Terraform Graph (Three Tier)




```
threnzy@THRENZY-M-W9PQ THREE_TIER % terraform graph -type=plan | dot -Tpng > graph.png
```

Terraform Variables & Iteration

Variables in Terraform

- Value substitution – makes code reusable

Can be defined	Variable Types	Complex Types
• CLI (`-var= -var-file=`) • `variables.tf` • Default config** • `terraform.tfvars`	• String • Number • Bool • Any (default)	• List • Map • Object

Terraform Variables

Variables File

Define variables with default value

```
resource aci_tenant “cl_tenant” {
   name        = var.tenant_name
   description = ”created by Terraform”
}

resource aci_vrf “cl_vrf” {
   tenant_dn   = aci_tenant.cl_tenant.id
   name        = var.vrf_name
   description = “
Created by terraform”
}
```

variables.tf

```
variable “tenant_name “{
   default = ”Cisco"
}

variable “vrf_name“{
   default = ”cisco_vrf"
}
```

Variables Assignment

Overrides Variable file default

```
resource aci_tenant “cl_tenant” {
   name        = var.tenant_name
   description = ”created by Terraform”
}

resource aci_vrf “cl_vrf” {
   tenant_dn   = aci_tenant.cl_tenant.id
   name        = var.vrf_name
   description = “
Created by terraform”
}
```

terraform.tfvars*

```
tenant_name=ciscolive
vrf_name=cl_vrf
```

*Overrides the variables set in variables.tf

Environment Variables

Based on environment (TF_VAR)

```
provider “mso” {
   username = var.NDO_USERNAME
   password = var.NDO_PASSWORD
   url      = https://10.2.2.2
   insecure = true
   platform = “nd”
}
```

```
variable “NDO_USERNAME“ {

}

variable “NDO_PASSWORD“{

}
```

*nix

```
export TF_VAR_NDO_USERNAME="admin"
export TF_VAR_NDO_PASSWORD=cisco123
```

Windows

```
$env:TF_VAR_NDO_USERNAME="admin"
$env:TF_VAR_NDO_PASSWORD="C1sco12345"
```

Terraform Variables Precedence

- Variables have precedence
- Variables can be set, but overridden

Higher ↑

- Command Line Flag run as command line switch
- Configuration file - set in your terraform.tfvars file
- Environment variable - part of your shell environment
- Default Config - default value in variables.tf
- User manual entry - if not specified, prompt the user for entry

lower ↓

https://developer.hashicorp.com/terraform/language/values/variables

Iteration (loop) in Terraform

- `count`
  - Add number of resources based on count

```
resource "aci_bridge_domain" "count_bd" {
  count        = 4
  tenant_dn    = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform count"
  name         = "bd_${count.index}"
  arp_flood    = "yes"
}
```

- `for_each`
  - Create objects based on a set or map

```
variable ”bds" {
   default = ["prod", "dev", "test”]
}

resource "aci_bridge_domain" ”three_tier_bd" {
  for_each           = toset(var.bds)
  tenant_dn          = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description        = "Created with Terraform for_each"
  name               = each.value
}
```

Deploying Infrastructure as Code with Terraform

Terraform – CLI commands

terraform init

- Download and Installs plugins for configured providers
- Must initialize before plan/apply
- Creates a provider “lock” file

```
terraform {
  required_providers {
    aci = {
      source = "CiscoDevNet/aci"
    }
  }
}
```

terraform plan

- Scans the current directory for the configuration (.tf & .tfvars extenstion)
- Determines what actions are necessary to achieve the desired state
- Preview your changes – no changes made

terraform apply

- Scans the current directory for the configuration (.tf & .tfvars extension)
- Preview your changes (can bypass with `-auto-approve`)
- Applies the configuration to targets (upon approval “yes”)

terraform destroy

- Scans the state file for what to “destroy”
- Preview your deletions
- Infrastructure is destroyed
- Can be specific with “-target”

Terraform – CLI commands

`terraform fmt`

- Formats Terraform configuration files in directory

`terraform show`

- Show the state file in a readable format
- Can also read a specific state file (path)

`terraform state`

- Advanced State Management
- `show <resource>` – Shows a particular resource
- `list` – Lists all resources in current state file
- `rm <instance>` – Remove an instance from the state file
- `mv` – Move an item. Good for renaming resources

`terraform validate`

- Verifies correctness of Terraform configuration files (*.tf)
- Checks syntax
- Can be used to solve configuration of errors

Deploying ACI Infrastructure - Terraform

EXPLORER

BRKDCN-2906

- main.tf
- terraform.tfvars
- variables.tf

main.tf > terraform > required_providers > aci

```
/*
  This configures the required provider plugins for ACI as well
  as the ACI provider. We are specifying the exact version of the ACI
  plugin that we want installed but you can remove this to install the latest
  one.
  The rest of the configurations are to deploy a three tier application.
*/

terraform {
  required_providers {
    aci = {
      source  = "CiscoDevNet/aci"
      version = "2.11.1"
    }
  }
}

/* Provider configuration with signature-based authentication. */
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR

```
threnzy@THRENZY-M-W9PQ  ~/Terraform/BRKDCN-2906
```

OUTLINE

TIMELINE

0 0 0 -- NORMAL -- Ln 13, Col 6 (19 selected) Spaces: 2 UTF-8 LF Terraform Spell

Result

APIC (Site2)

admin

System | Tenants | Fabric | Virtual Networking | Admin | Operations | Apps | Integrations

ALL TENANTS | Add Tenant | Tenant Search: name or descr | common | Prod | infra | mgmt

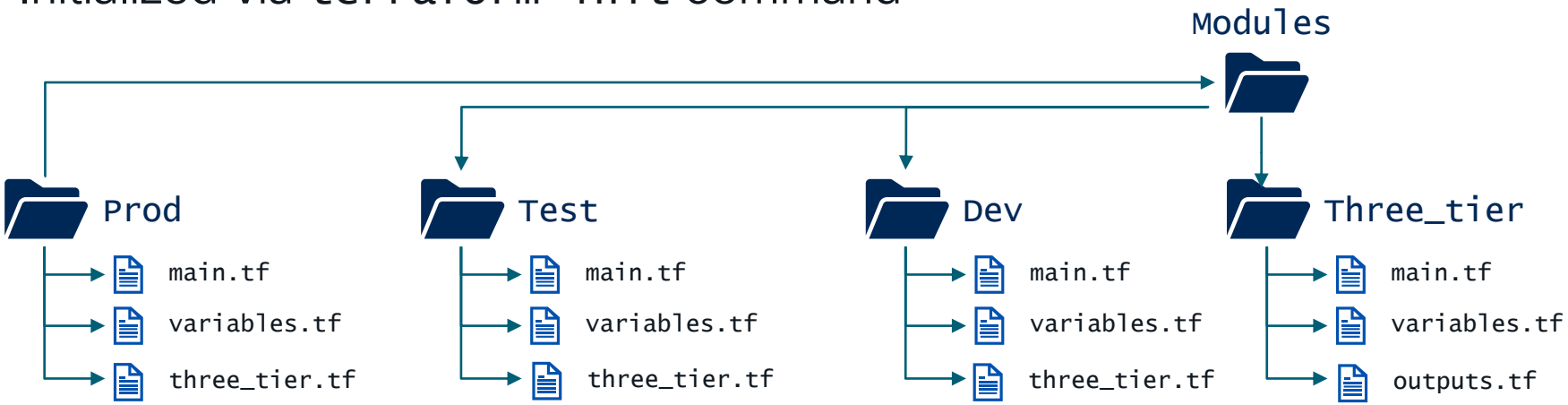
All Tenants

Name	Alias	Description	Bridge Domains	VRFs	EPGs	Health Score
common			1	2	0	Healthy
infra			2	2	2	Healthy
mgmt			2	2	1	Healthy
Prod Terraform		Created with Terraform	3	1	3	Healthy

Page 1 Of 1 Objects Per Page: 100 Displaying Objects 1 - 4 Of 4

Terraform Modules - Reusability

- Repeatable code that you can reuse
- Write your own or download (Terraform registry)
- Initialized via `terraform init` command



Terraform Modules – Three Tier Application

Prod

three_tier.tf

```
resource "aci_tenant" "mod_tenant" {
  name        = var.tenant_name
  description = "Created with Terraform Modules"
}

resource "aci_vrf" "mod_vrf" {
  tenant_dn   = aci_tenant.mod_tenant.id
  name        = var.vrf_name
  description = "Created with Terraform Modules"
}

resource "aci_application_profile" "mod_ap" {
  tenant_dn = aci_tenant.mod_tenant.id
  name      = var.ap_name
  description = "Created with Terraform Modules"
}
...
```

Modules

three_tier

three_tier.tf

Terraform Modules – Three Tier Application

Modules → three_tier

```
terraform {
  required_version = ">= 1.6.0"
  required_providers {
    aci = {
      source  = "CiscoDevNet/aci"
      version = "2.11.1"
    }
  }
}
```

versions.tf

```
variable "tenant_name" {
  type = string
}

variable "ap_name" {
  type = string
}

variable "vrf_name" {
  type = string
}

variable "epg_name" {
  type = string
}

variable "bd_name" {
  type = string
}
...
```

variables.tf
(match input variables)

```
resource "aci_tenant" "mod_tenant" {
  name        = var.tenant_name
  description = "Created with Terraform Modules"
}

resource "aci_vrf" "mod_vrf" {
  tenant_dn   = aci_tenant.mod_tenant.id
  name        = var.vrf_name
  description = "Created with Terraform Modules"
}

resource "aci_application_profile" "mod_ap" {
  tenant_dn = aci_tenant.mod_tenant.id
  name      = var.ap_name
  description = "Created with Terraform Modules"
}
...
```

three_tier.tf

Terraform Modules – Three Tier Application

Module Name

```
module ”devtenant" {
    source = "../../MODULES/three_tier"
    tenant_name   = ”Dev"
    vrf_name = ”dev_vrf"
    ap_name = ”dev_app"
    epg_name = ”dev_epg"
    bd_name = ”dev_bd"
    bd_subnet = "10.1.1.1/24"
    bd_scope = ["public"]
...
}
```

Path to module code

Input variables. Match the variables in the Modules directory

Terraform Modules – Three Tier Application

Prod

three_tier.tf

```
module "mytenant" {
    source = "../../MODULES/three_tier"
    tenant_name   = "prod"
    vrf_name = "prod_vrf"
    ap_name = "prod_app"
    epg_name = "prod_epg"
    bd_name = "prod_bd"
    bd_subnet = "10.1.1.1/24"
    bd_scope = ["public"]
}
```

Modules

three_tier

three_tier.tf

```
resource "aci_tenant" "mod_tenant" {
  name        = var.tenant_name
  description = "Created with Terraform Modules"
}

resource "aci_vrf" "mod_vrf" {
  tenant_dn   = aci_tenant.mod_tenant.id
  name        = var.vrf_name
  description = "Created with Terraform Modules"
}

resource "aci_application_profile" "mod_ap" {
  tenant_dn = aci_tenant.mod_tenant.id
  name      = var.ap_name
  description = "Created with Terraform Modules"
}
...
```

Deploying ACI Infrastructure - Terraform

APIC (Site2)

admin

System | Tenants | Fabric | Virtual Networking | Admin | Operations | Apps | Integrations

ALL TENANTS | Add Tenant | Tenant Search: name or descr | common | infra | mgmt

All Tenants

Name	Alias	Description	Bridge Domains	VRFs	EPGs	Health Score
common			1	2	0	Healthy
Dev Terraform		Created with Terraform	3	1	3	Healthy
infra			2	2	2	Healthy
mgmt			2	2	1	Healthy
Prod Terraform		Created with Terraform	3	1	3	Healthy
Test Terraform		Created with Terraform	3	1	3	Healthy

Ansible and Terraform comparison

Ansible/Terraform comparison

Source	Open Source	Open Source
Cisco commitment	Yes!	Yes!
IaC Type	Configuration Management	Provisioning
Language Type	Procedural	Declarative
Stateful	No	Yes
ACI/MSO Modules/Resources	149*/63*	230*/45*
Written in	Python	Go
TAC Support	Yes	Yes

* At the time of this presentation

Feeling Overwhelmed?

What is your path?

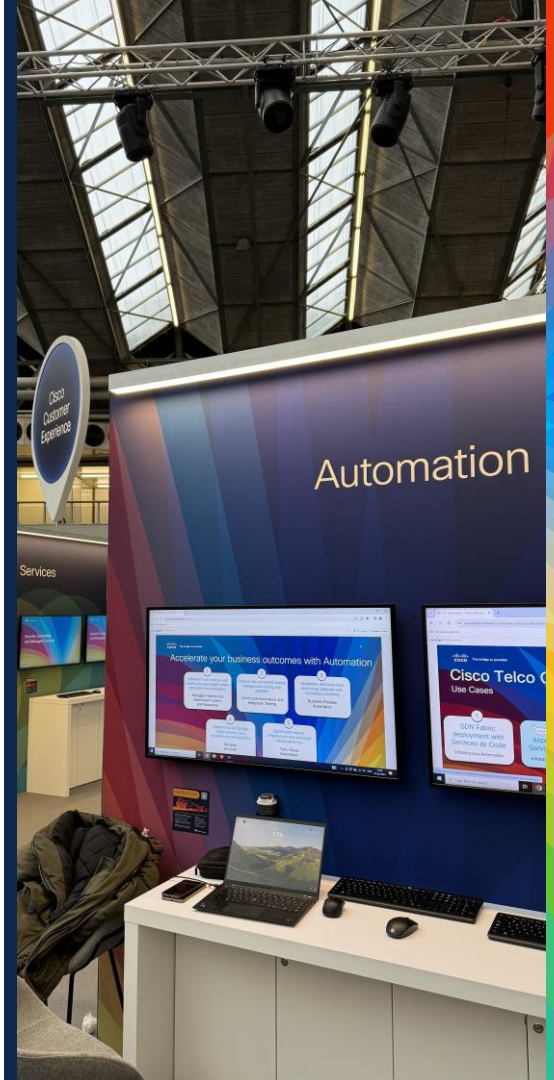
I got this!

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



Next Steps

Infrastructure as Code with Terraform and Ansible

- Install and test Terraform and Ansible
  - Available for most platforms
  - Which one works better for you?
  - What are you already using?
- Think big…..start small
  - Automate the simple, then build into more complex tasks
- Ease of writing Infrastructure as code with Terraform and Ansible
  - No special programming skills needed
- Ansible Modules/Terraform Resources for most common tasks
- Robust APIC/MSO REST API makes automation easy and scalable

More information – Other sessions/labs

- LABDCN-1774 (Walk in Lab - Ansible and ACI)
- LABDCN-1776 (Walk in Lab - Intro to Terraform with ACI)
- BRKDCN-2673 - Nexus-as-Code - Kickstart your automation with ACI
- DEVWKS-2931 - Making your ACI Automation as modular as LEGO bricks using Terraform Modules
- DEVWKS-1098 - Infrastructure as Code on NX-OS using Terraform
- IBODCN-1003 - An Interactive Conversation on ACI Automation through Ansible and Terraform

More information – Ansible/Terraform

- https://www.terraform.io/
- https://registry.terraform.io/providers/CiscoDevNet/aci/latest/docs
- https://registry.terraform.io/providers/CiscoDevNet/mso/latest
- https://developer.cisco.com/automation-terraform/
- https://docs.ansible.com/ansible/latest/scenario_guides/guide_aci.html
- https://developer.cisco.com/docs/aci/#!ansible
- https://developer.cisco.com/docs/nexus-as-code/
- https://github.com/trenzy

The bridge to possible

Thank you

cisco Live!

cisco Live!

Let's go