
TURN
IT
UP
cisco Live!
#CiscoLive

The bridge to possible

Ansible and Terraform – Accelerating ACI Fabric Deployments

Thomas Renzy, Technical Leader – Customer Experience
@ThomasRenzy

BRKACI-2398

Agenda

- Introduction
- Overview of Ansible and Terraform
- Accelerating your ACI with Terraform and Ansible
- Conclusion

Introduction

What is Automation?

- Exists to make repeatable things easier
- Uses tools to create process and instructions
  - Replaces manual work
- Benefits – Speed, Efficiency, Cost savings

Infrastructure as code

- Writing code to describe infrastructure
  - Convert manual tasks to code
- Can be used to automate provisioning
- Leverages Software Development tools
  - Version control, CI/CD, testing, documentation
- Benefits
  - Adaptability
  - Repeatable
  - Scale

Ansible and Terraform Overview

cisco Live!

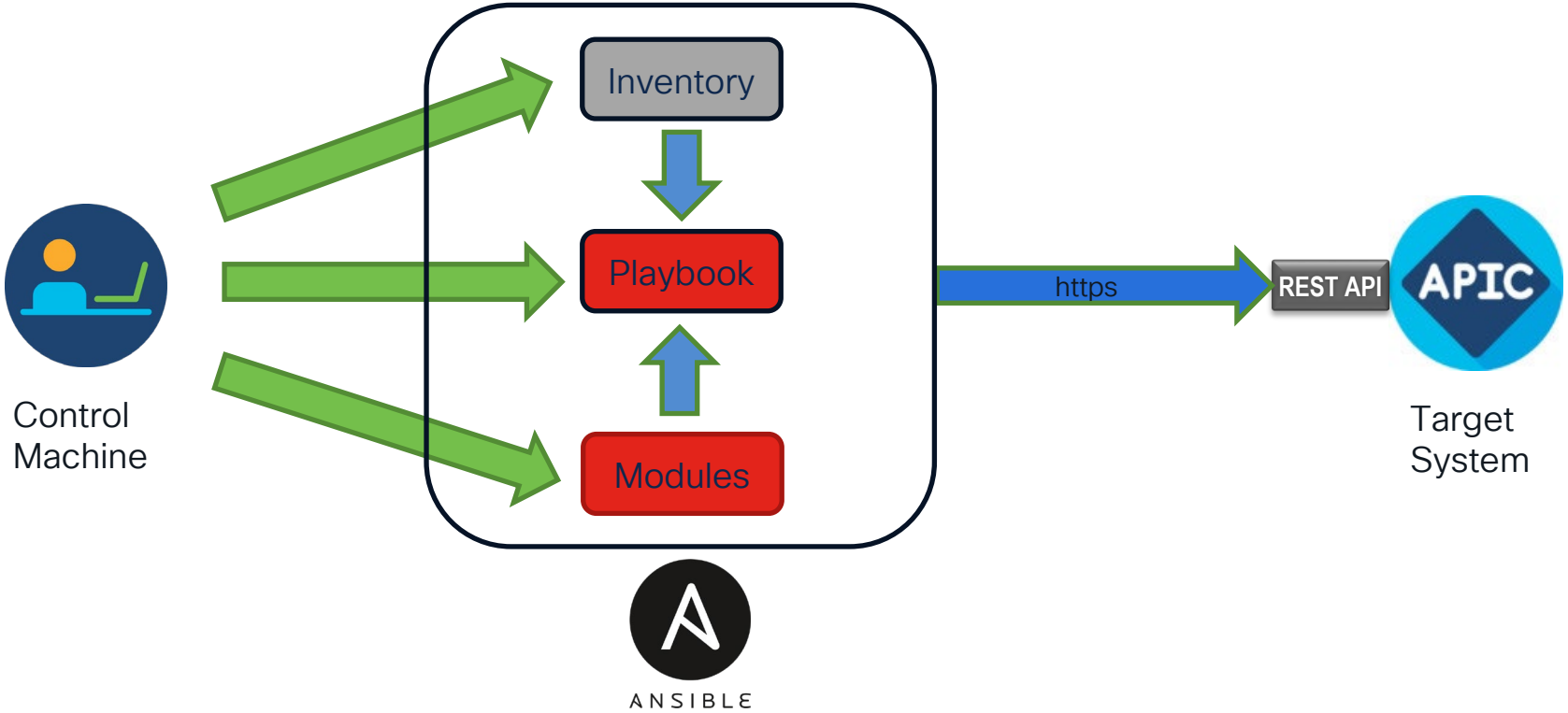
What is Ansible?

- Open Source
- Automation, Configuration
- Version 2.10
  - ACI support - 2.4
  - 3.0 released
- Supported on UNIX/Linux
  - Windows Subsystem for Linux
- Can manage different systems
  - ACI, MSO, IOS, NX-OS, IOS-XR

What is Ansible?

- Agentless
  - Push Model
- Idempotent
- YAML based
  - Easily Readable
- APIC REST API interface
  - Same as GUI
- Requires no programming skills
  - Python is helpful – not required

What makes up Ansible?



Example ACI Ansible Inventory

YAML inventory file

```
apic1:
  hosts:
    10.9.3.21:
  vars:
    username: admin
    password: CiscoAC1
```

INI inventory file

```
[apic1]
10.9.3.21 username=admin password=CiscoAC1
10.9.3.22 username=ansible privatekey=ansible.key
```

Ansible Playbook breakdown

Start of YAML

Comment

Name of Playbook

Hosts from inventory

Connection is local to this host

Collects information about targets

Watch the Indentation!

```
---
# Demo ACI Playbook
- name: Configuring Example Tenant
  hosts: apic1
  connection: local
  gather_facts: no

tasks:
- name: Create Tenant
  aci_tenant:
    hostname: "{{ inventory_hostname }}"
    username: "{{ username }}"
    private_key: ansible.key
    tenant: "CiscoLive"
    description: "Tenant configured by Ansible"
    validate_certs: no
    state: present
```

Ansible Playbook breakdown

Task name

Module Name

Hostname

Authentication

Tenant

Description of task

Validate certs

Add if not already "present"

```
---
# Demo ACI Playbook
- name: Configuring Example Tenant
   hosts: apic1
   connection: local
   gather_facts: no

tasks:
- name: Create Tenant
  aci_tenant:
    hostname: "{{ inventory_hostname }}"
    username: "{{ username }}"
    private_key: ansible.key
    tenant: "CiscoLive"
    description: "Tenant configured by Ansible"
    validate_certs: no
    state: present
```

Ansible ACI Modules

```
zsh
(2.9) threnzy@THRENZY-M-F1G3 2.9 % ansible-doc -l | grep ^aci
aci_aaa_user                                          Manage AAA users (aaa:User)
aci_aaa_user_certificate                              Manage AAA user certificates (aaa:UserCert)
aci_access_port_block_to_access_port                  Manage port blocks of Fabric interface poli...
aci_access_port_to_interface_policy_leaf_profile      Manage Fabric interface policy leaf profile...
aci_access_sub_port_block_to_access_port              Manage sub port blocks of Fabric interface ...
aci_aep                                               Manage attachable Access Entity Profile (AE...
aci_aep_to_domain                                     Bind AEPs to Physical or Virtual Domains (i...
aci_ap                                                Manage top level Application Profile (AP) o...
aci_bd                                                Manage Bridge Domains (BD) objects (fv:BD)
aci_bd_subnet                                         Manage Subnets (fv:Subnet)
aci_bd_to_l3out                                       Bind Bridge Domain to L3 Out (fv:RsBDToOut)
aci_config_rollback                                   Provides rollback and rollback preview func...
aci_config_snapshot                                   Manage Config Snapshots (config:Snapshot, c...
aci_contract                                          Manage contract resources (vz:BrCP)
aci_contract_subject                                  Manage initial Contract Subjects (vz:Subj)
aci_contract_subject_to_filter                        Bind Contract Subjects to Filters (vz:RsSub...
aci_domain                                            Manage physical, virtual, bridged, routed o...
aci_domain_to_encap_pool                              Bind Domain to Encap Pools (infra:RsVlanNs)
aci_domain_to_vlan_pool                               Bind Domain to VLAN Pools (infra:RsVlanNs)
aci_encap_pool                                        Manage encap pools (fvns:VlanInstP, fvns:Vx...
aci_encap_pool_range                                  Manage encap ranges assigned to pools (fvns...
aci_epg                                               Manage End Point Groups (EPG) objects (fv:A...
aci_epg_monitoring_policy                             Manage monitoring policies (mon:EPGPol)
aci_epg_to_contract                                   Bind EPGs to Contracts (fv:RsCons, fv:RsPro...
aci_epg_to_domain                                     Bind EPGs to Domains (fv:RsDomAtt)
aci_fabric_node                                       Manage Fabric Node Members (fabric:NodeIden...
aci_fabric_scheduler                                  This modules creates ACI schedulers
aci_filter                                            Manages top level filter objects (vz:Filter...
aci_filter_entry                                      Manage filter entries (vz:Entry)
```

Running our Tenant Playbook

```
(2.9) THRENZY-M-F1G3:BRKACI-1619 threnzy$ ansible-playbook -i hosts ciscolive.yml

PLAY [Configuring Example Tenant] ***********************************************

TASK [Create a New Tenant] ******************************************************
changed: [10.95.33.231]

PLAY RECAP **********************************************************************
10.95.33.231               : ok=1    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0

(2.9) THRENZY-M-F1G3:BRKACI-1619 threnzy$
```

- Runs through each task.
- Let's you know how many tasks were OK, changed, failed, etc.
- To see more output use "-v", "-vvv", or "-vvvv"

Verifying the APIC

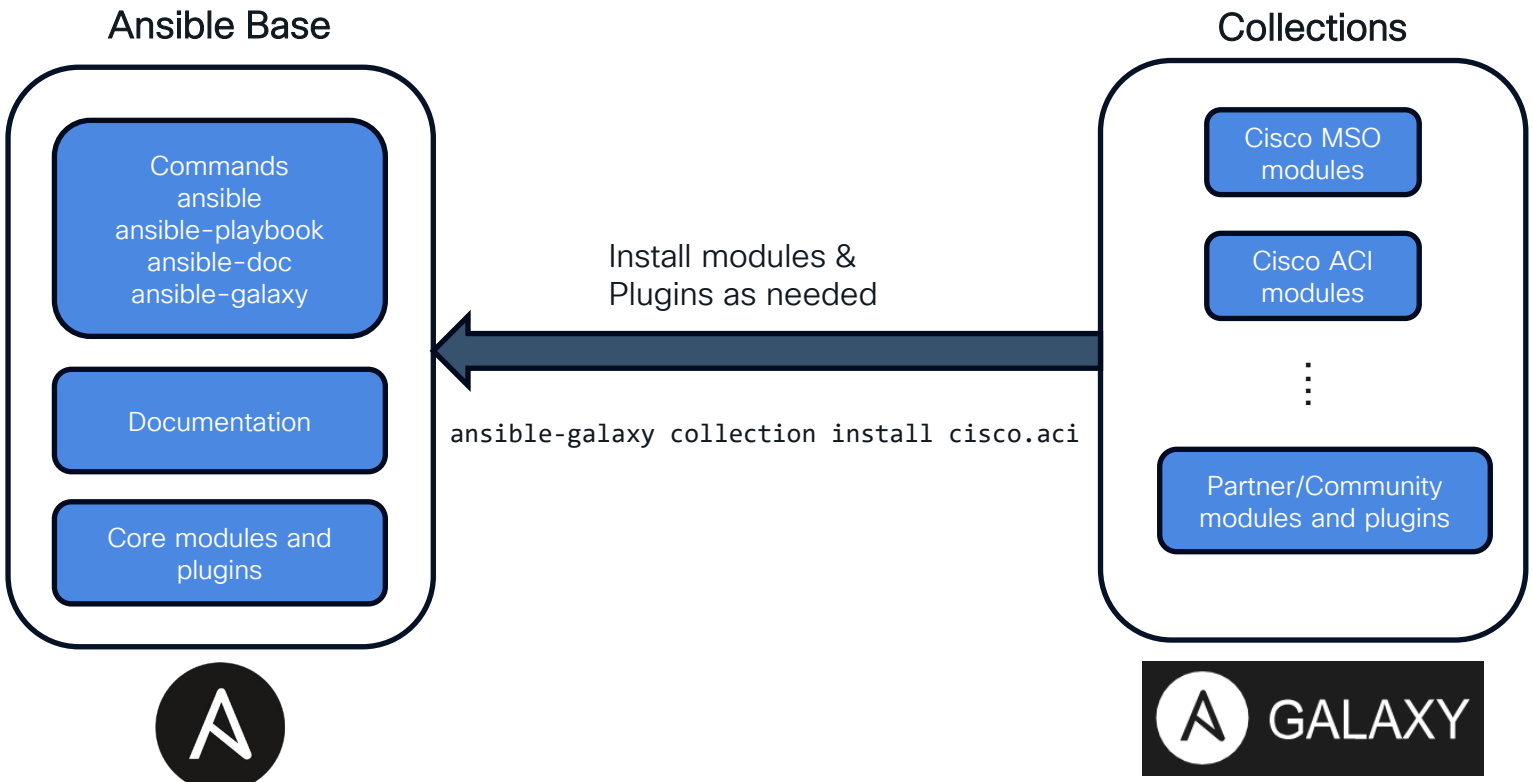
All Tenants

Name	Alias	Description	Bridge Domains	VRFs	EPGs	Health Score
CiscoLive		Tenant configured by Ansible	0	0	0	100
common			1	2	0	100
infra			2	2	2	100
mgmt			1	2	0	100

What are Ansible collections?

- Ansible project/directory structure for Ansible Content
- Broken out into different components
  - Core engine (`ansible, ansible-playbook, etc`)
  - Core modules and plugins
  - Community modules (Cisco ACI, Multi-Site)
- Uses Ansible Galaxy to deliver collection
  - ACI - https://galaxy.ansible.com/cisco/aci
  - MSO - https://galaxy.ansible.com/cisco/mso
- Support since 2.9 – standard with new 2.10 release

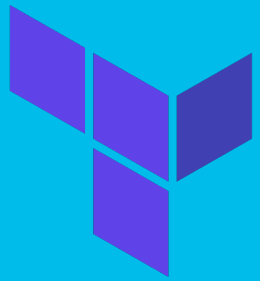
Ansible Collections



ACI Collection Example - Tenant

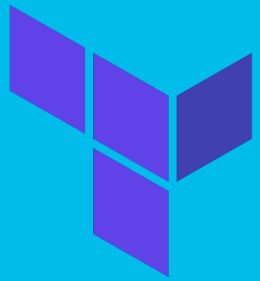
```
---
# Demo ACI Playbook
- name: Configuring Example Tenant
  hosts: apic1
  connection: local
  gather_facts: no

  tasks:
  - name: Create a New Tenant
    cisco.aci.aci_tenant:
      hostname: "{{ inventory_hostname }}"
      username: "{{ username }}"
      password: "{{ password }}"
      tenant: "CiscoLive"
      description: "Tenant configured by Ansible"
      validate_certs: no
      state: present
```

What is Terraform?



- Open Source
- Infrastructure provisioning tool
- HashiCorp Configuration Language
  - Written in Go
- Single binary – Linux, Windows, MacOS
- Can be combined with configuration management tools

What is Terraform?



- Immutable
- Declarative
- Agentless
- Leverages Plugins
  - From Cisco, AWS, etc.
- No programming skills needed
- Version 0.14
  - ACI support – 0.12

Terraform Overview

- Providers
- Resources & Data Sources
- Variable substitution
- Execution Plan – HCL instructions
  - Actions to be performed to reach a desired end state
- Interpolation
- Direct Acyclic Graph (DAG)
- Configuration files - .tf extension

Terraform Execution Plan example

```
terraform {
  required_providers {
    aci = {
      source = "CiscoDevNet/aci"
      version = "0.5.4"
    }
  }
}

provider "aci" {
  username = "tform"
  private_key = "tfcert.key"
  cert_name = "tfcert.crt"
  url      = "https://10.201.36.113/"
  insecure = true
}
```

Terraform configuration

Required providers

ACI provider configuration

Provider configuration

username

Signature-Based Authentication

APIC URL

http API request

Terraform Resource Example – ACI Tenant

Type of resource

Name of the resource

Tenant Name

```
resource "aci_tenant" "terraform" {
  name        = "terraform"
  description = "This tenant is created by terraform"
}
```

Terraform commands

- `terraform validate`
  - Validates the configuration files in a directory
- `terraform plan`
  - used to create an execution plan
  - determines what actions are necessary to achieve the desired state
- `terraform apply (-auto-approve)`
  - scans the current directory for the configuration
  - Applies the configuration to targets
- `terraform destroy`
  - Infrastructure managed by Terraform will be destroyed.
  - This will ask for confirmation before destroying

Terraform init

```
threnzy@THRENZY-M-F1G3 TENANT % terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of ciscodevnet/aci...
- Installing ciscodevnet/aci v0.5.4...
- Installed ciscodevnet/aci v0.5.4 (signed by a HashiCorp partner, key ID 433649E2C56309DE)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* ciscodevnet/aci: version = "~> 0.5.4"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Terraform Plan

```
threnzy@THRENZY-M-F1G3 aci % terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aci_tenant.terraform will be created
  + resource "aci_tenant" "terraform" {
      + annotation  = (known after apply)
      + description = "This tenant is created by terraform"
      + id          = (known after apply)
      + name        = "terraform"
      + name_alias  = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------
```

Terraform apply

```
threnzy@THRENZY-M-F1G3 aci % terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aci_tenant.terraform will be created
  + resource "aci_tenant" "terraform" {
      + annotation  = (known after apply)
      + description = "This tenant is created by terraform"
      + id          = (known after apply)
      + name        = "terraform"
      + name_alias  = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aci_tenant.terraform: Creating...
aci_tenant.terraform: Creation complete after 2s [id=uni/tn-terraform]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
threnzy@THRENZY-M-F1G3 aci %
```

Terraform destroy

```
threnzy@THRENZY-M-F1G3 aci % terraform destroy
aci_tenant.terraform: Refreshing state... [id=uni/tn-terraform]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aci_tenant.terraform will be destroyed
  - resource "aci_tenant" "terraform" {
      - description = "This tenant is created by terraform" -> null
      - id          = "uni/tn-terraform" -> null
      - name        = "terraform" -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aci_tenant.terraform: Destroying... [id=uni/tn-terraform]
aci_tenant.terraform: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
threnzy@THRENZY-M-F1G3 aci %
```

Terraform State Example

```
threnzy@THRENZY-M-F1G3 TENANT % more terraform.tfstate
{
  "version": 4,
  "terraform_version": "0.13.4",
  "serial": 1,
  "lineage": "751f18b5-d3d2-8f3b-a311-2ae902ec2170",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aci_tenant",
      "name": "terraform",
      "provider": "provider[\"registry.terraform.io/ciscodevnet/aci\"]",
      "instances": [
        {
          "schema_version": 1,
          "attributes": {
            "annotation": "orchestrator:terraform",
            "description": "This tenant is created by terraform",
            "id": "uni/tn-terraform",
            "name": "terraform",
            "name_alias": "",
            "relation_fv_rs_tenant_mon_pol": null,
            "relation_fv_rs_tn_deny_rule": null
          },
          "private": "eyJzY2hlbWFfdmVyc2lvbiI6IjEifQ==="
        }
      ]
    }
  ]
}
```

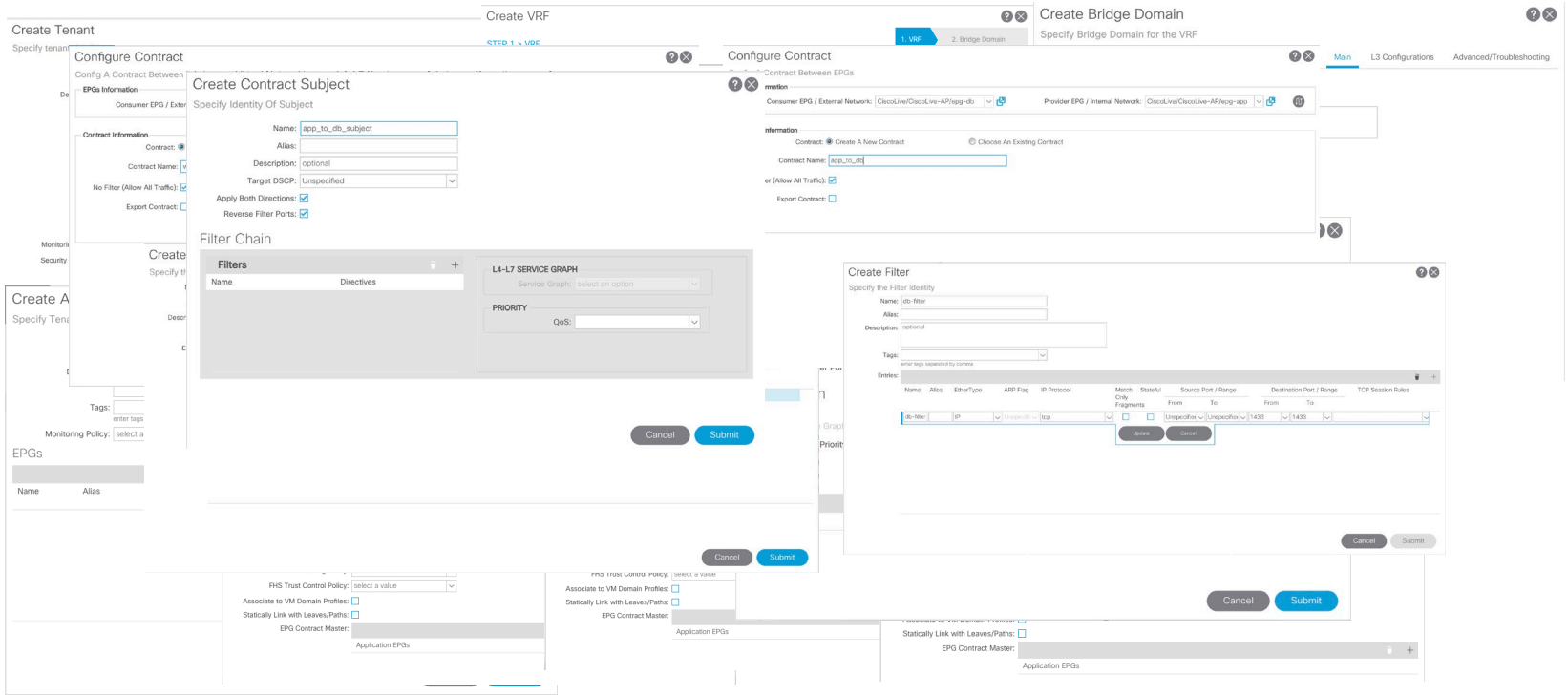
ACI REST Module/Resource (aci_rest)

- Works around no Ansible/Terraform module/resource
- Direct access and management to APIC REST API
- Can use JSON, XML, and even YAML
- Can POST, DELETE, GET
- Variable substitution
- Can grab GUI configurations through
  - API Inspector
  - Download JSON/XML configuration

Accelerating ACI deployments with Ansible and Terraform

cisco Live!

Deploy Three Tier Application – APIC GUI



ACI Interface configuration – APIC GUI

Create VLAN Pool

Name: web
Description: optional
Allocation Mode: Dynamic Allocation | Static Allocation
Encap Blocks:

Create Ranges

Type: VL
Description:
Range: VL
Allocation Mode:

Create Attachable Access Entity Profile

STEP 1 > Profile

Create Attachable Access Entity Profile

STEP 2 > Association To Interfaces

1. Profile | 2. Association To Interfaces

Interface Policy Group	Type	Associated Attachable Access Entity Profile	Switches / Fexes	Interfaces	Select Interfaces
10G-inb	Individual	phys-in			

Create Access Port Selector

Name: E_20
Description: optional
Interface IDs: 1/20
valid values: All or Ranges. For Example: 1/13, 1/15 or 2/22-2/24, 2/16-3/16, or 1/21-23/1-4, 1/24/1-2
Connected To Fex:
Interface Policy Group: web

Cancel | Submit

Create Leaf Access Port Policy Group

Name: web
Description: optional
Attached Entity Profile: web
Link Level Flow Control: select a value
CDP: CDP_On
LLDP: LLDP_On

nced Settings

02.1x Port Authentication: select a value
MACsec: select a value
CoPP: select a value
MCP: select a value
DWDM: select a value
Monitoring: select a value
gress Data Plane Policing: select a value
PoE Interface: select a value
Fibre Channel Interface: select a value
Port Security: select a value
gress Data Plane Policing: select a value
Priority Flow Control: select a value
L2 Interface: select a value
Slow Drain: select a value
Link Flap: select a value
Storm Control Interface: select a value
Link Level: select a value
STP Interface: select a value

NetFlow Monitor Policies:

NetFlow IP Filter Type	NetFlow Monitor Policy

Cancel | Submit

Ansible Demo

Terraform Demo

Ansible/Terraform comparison

Source	Open Source	Open Source
Type	Configuration Management	Provisioning
Language	Procedural	Declarative
Infrastructure	Mutable	Immutable
ACI Modules/Resources	> 80*	> 100*
MSO Modules/Resources	> 50*	> 45*

* At the time of this presentation – more available in newer versions

Conclusion

More information

- https://www.terraform.io/
- https://docs.ansible.com/
- https://registry.terraform.io/providers/CiscoDevNet/aci/latest/docs
- https://docs.ansible.com/ansible/latest/scenario_guides/guide_aci.html
- https://github.com/CiscoDevNet/ansible-aci/
- https://github.com/CiscoDevNet/terraform-provider-aci
- https://github.com/trenzy/BRKACI-2398

Thank you


TURN
IT
UP
cisco Live!
#CiscoLive