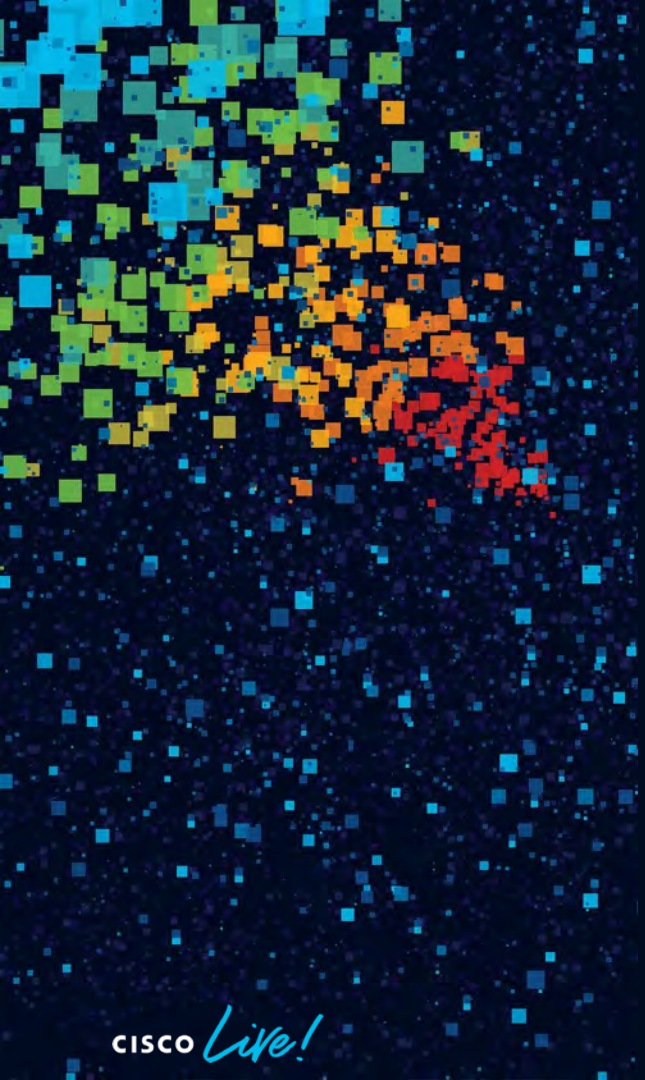
Deconstructing DCN CI/CD Pipelines

How to integrate your Data Center Network in your pipelines

Lionel Hercot, Technical Marketing Engineer
@LHercot

cisco Live!

cisco



Agenda

- What is Infrastructure as Code?
- Components of a pipeline
  - Ansible
  - Terraform
  - Jenkins
- CI/CD Pipeline in action

Infrastructure as code – What/Why/How

- Automate the provisioning and management of the technology stack
- Translate manual tasks into reusable, robust, distributable code
- Rely on practices that have been successfully used for years in software development (version control, automated testing, release tagging, continuous delivery, etc.)
- Benefits: much higher delivery speed; significant reliability boost

CI/CD Pipeline

- Continuous Integration (CI)
  - Practice of merging all developer changes to a shared repo several times a day
  - It main include the creation and test of artifacts (executable, app, ...)
- Continuous Deployment (CD)
  - Approach to deliver new software functionalities frequently through automated deployments
  - Rely on Continuous Integration for tracking changes

Common components of a CI/CD Pipeline

What is Ansible?

- Open-source Infrastructure Provisioning Tool
- Commercial support from RedHat
- Declarative (when possible) and idempotent
- Can manage a wide range of systems:
  VMs, network devices, cloud instances, etc.
- Agentless
- Python server-side dependencies

What is Terraform?

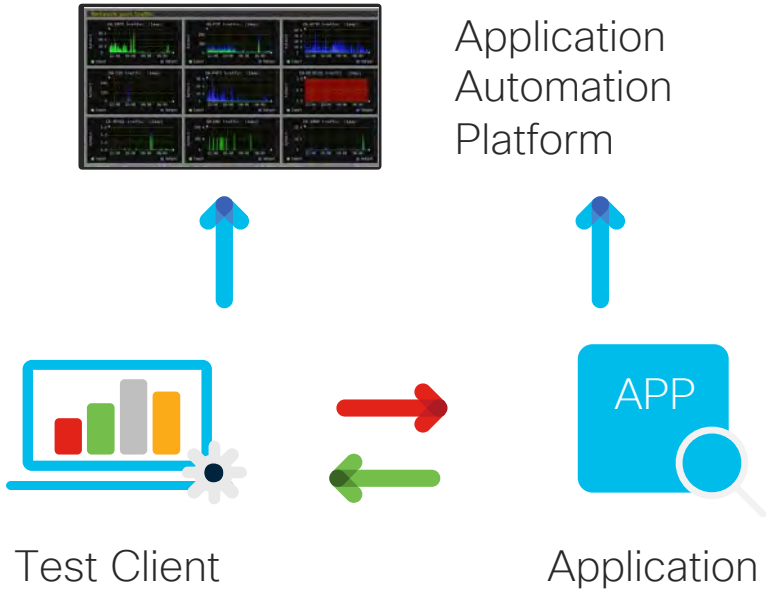
- Open-source configuration management tool
- Commercial support from HashiCorp
- Declarative and idempotent
- Can manage a wide range of systems:
  VMs, network devices, cloud instances, etc.
- Agentless, single binary file
- Zero server-side dependencies

Ansible or Terraform?

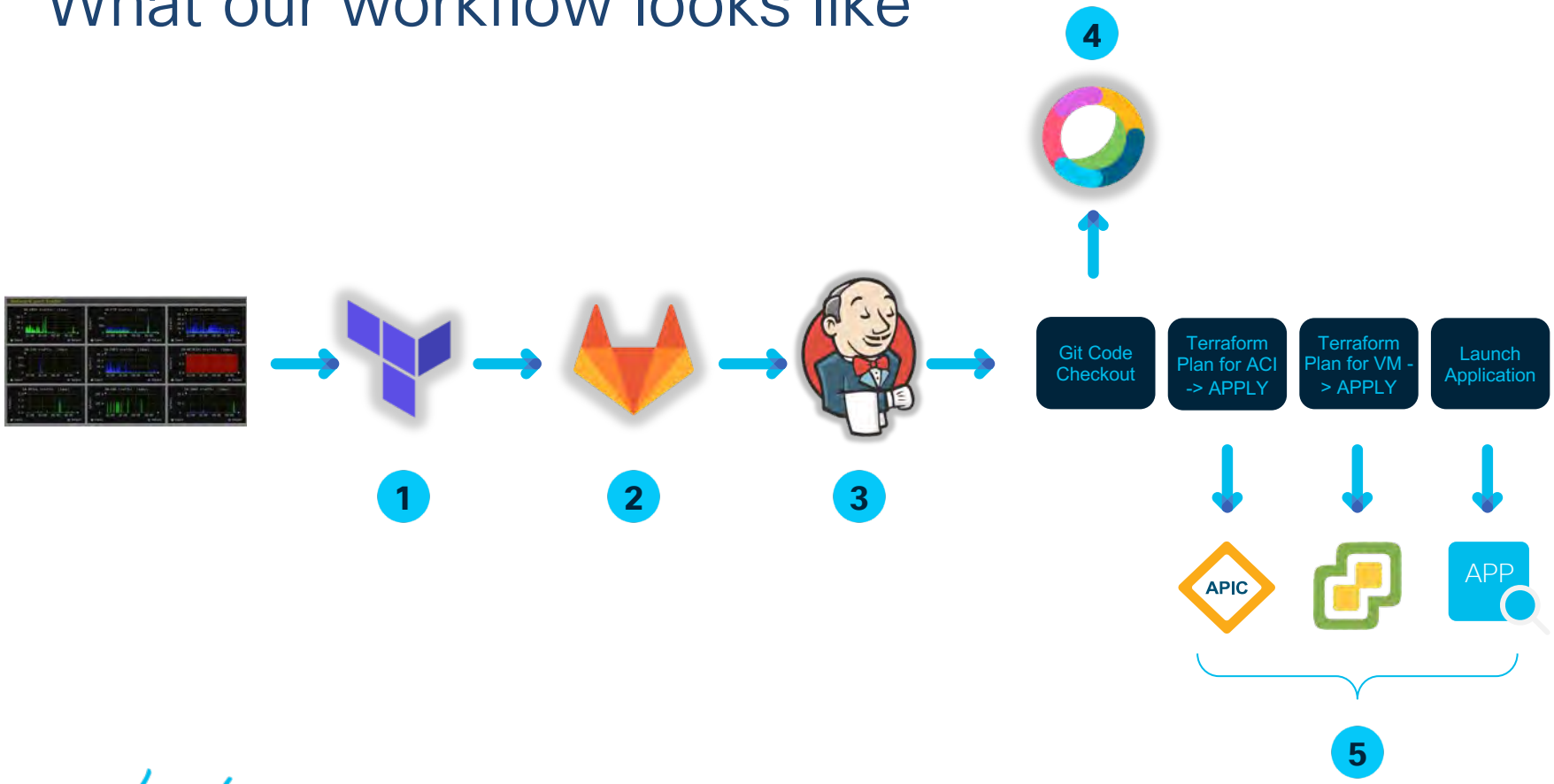
- Both Ansible and Terraform can coexist
  - It's not an either/or story
  - Terraform can call Ansible for ad-hoc tasks after deploying a VM
- Terraform keeps state locally
  - It knows what is configured vs desired end-state
  - Can automatically destroy / recreate resources
- Ansible mutate the infrastructure
- Cisco Modules / Provider
  - Cisco ACI plugins for both
  - Cisco MSO for Ansible is released, Terraform Provider is in beta.


CI/CD Pipeline in action

Let’s Deploy an App in a Zero Trust Environment



What our workflow looks like



Our infrastructure written in code

Hashicorp Configuration Language

1

variables.tf vm.tf aci.tf

ACI > aci.tf

```
provider "aci" {
  username = "${var.aciUser}"
  private_key = "${var.aciPrivateKey}"
  cert_name = "${var.aciCertName}"
  insecure = true
  url = "${var.aciUrl}"
}


resource "aci_tenant" "demo" {
  name = "${var.tenantName}"
  description = "created by terraform"
}

resource "aci_vrf" "vrf1" {
  tenant_dn = "${aci_tenant.demo.id}"
  name      = "vrf1"
}

resource "aci_bridge_domain" "bd1" {
  tenant_dn          = "${aci_tenant.demo.id}"
  relation_fv_rs_ctx = "${aci_vrf.vrf1.name}"
  name               = "bd1"
}

resource "aci_subnet" "bd1_subnet" {
  bridge_domain_dn = "${aci_bridge_domain.bd1.id}"
  ip               = "${var.bd_subnet}"
}
```

VMWARE > vm.tf > vm > { } provisioner > inline

```
provider "vsphere" {
  user           = "${var.vsphere_user}"
  password       = "${var.vsphere_password}"
  vsphere_server = "${var.vsphere_server}"

  # If you have a self-signed cert
  allow_unverified_ssl = true
}

6 references
data "vsphere_datacenter" "dc" {
  name = "Datacenter-OnPrem"
}

1 references
data "vsphere_datastore" "datastore" {
  name          = "HDD-Local"
  datacenter_id = "${data.vsphere_datacenter.dc.id}"
}

0 references
data "vsphere_distributed_virtual_switch" "dvs" {
  name          = "ACI-DVS"
  datacenter_id = "${data.vsphere_datacenter.dc.id}"
}
```

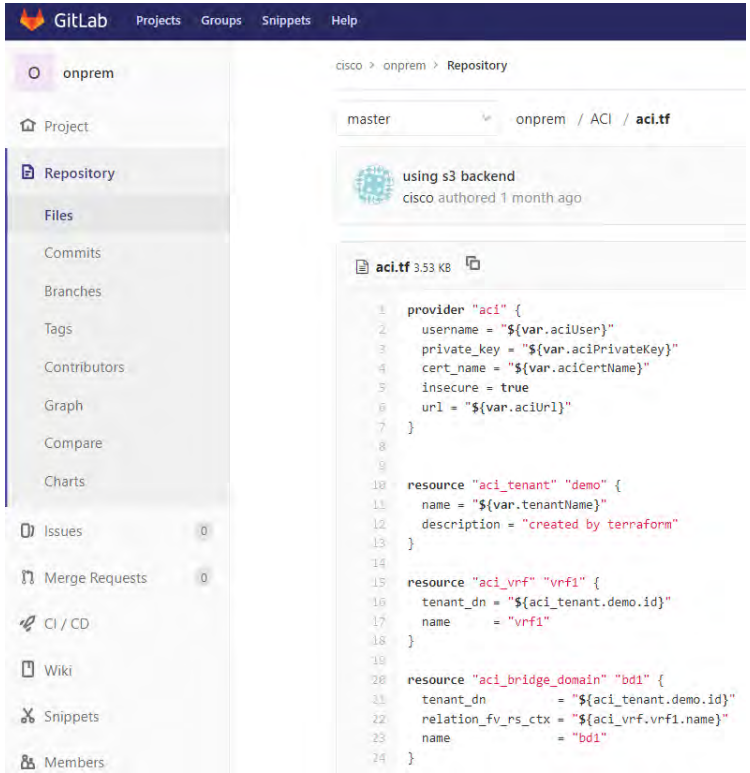
ACI > variables.tf > provider_profile_dn >

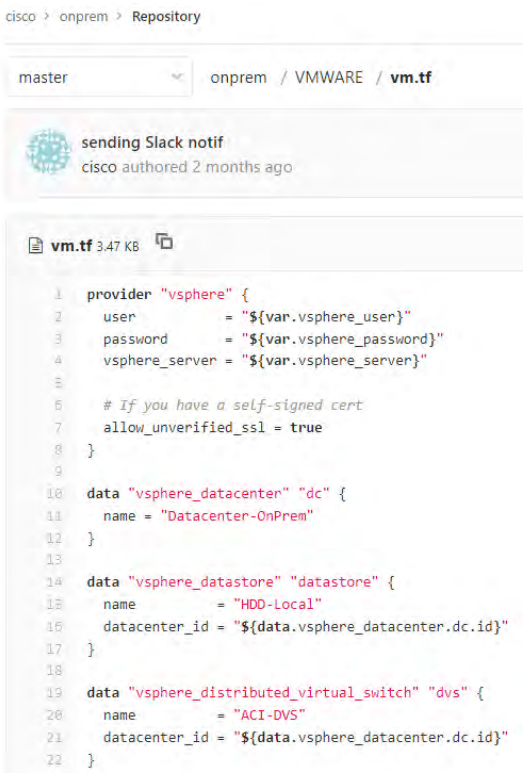
```
1 references
variable "tenantName" {
  default = "terraformDemo"
}
1 references
variable "aciUser" {
  default = "lionel"
}
1 references
variable "aciPrivateKey" {
  default = "lionel.key"
}
1 references
variable "aciCertName" {
  default = "lionel"
}
1 references
variable "aciUrl" {
  default = "https://10.0.99.11"
}

1 references
variable "bd_subnet" {
  type    = "string"
  default = "1.1.1.1/24"
}
```

Code is pushed to a version-controlled Git repo

2

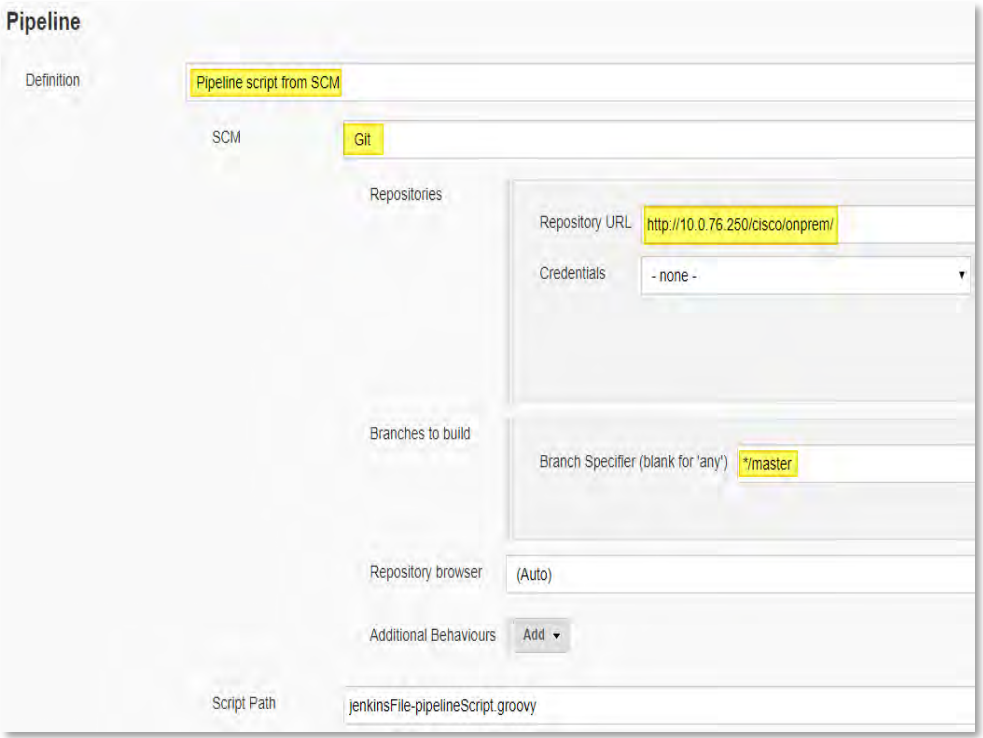




What does the pipeline look like?

Code of course!

3



variables.tf ● jenkinsFile-pipelineScript.groovy × vm.tf

jenkinsFile-pipelineScript.groovy

```
30
31  stages {
32    stage('Git code checkout'){
33      steps {
34        dir('dev'){
35          ansiColor('xterm'){
36            webexTeamsTalk("ACI-VMware: fetching source from Git")
37            git branch: 'master', url: 'http://10.0.76.250/cisco/onprem.git'
38          }
39        }
40      }
41    }
42
43    stage('Terraform Init for the ACI plan'){
44      steps {
45        dir('dev/ACI'){
46          ansiColor('xterm'){
47            webexTeamsTalk("ACI-VMware: terraform init ACI")
48            sh "terraform init -input=false -upgrade -force-copy"
49            sh "echo \$PWD"
50          }
51        }
52      }
53    }
54
55    stage('Terraform Init for the VMWARE plan'){
56      steps {
57        dir('dev/VMWARE'){
58          ansiColor('xterm'){
```

The orchestrator is executing the workflow

3

Pipeline ACI-VMware-OnPrem

Terraform-based project that creates an ACI tenant (3 EPGs: client, web, admin) with contracts to secure a web-app (client -> web:8080; admin -> client:22 and web:22). Terraform deploys two VMs on vCenter and instantiates a web server on 8080 using Python Flask. The admin machine (the Terraform provisioner) SSHs into the machine to deploy the app, so this also verifies the ACI contract.

edit description

Disable Project

Recent Changes

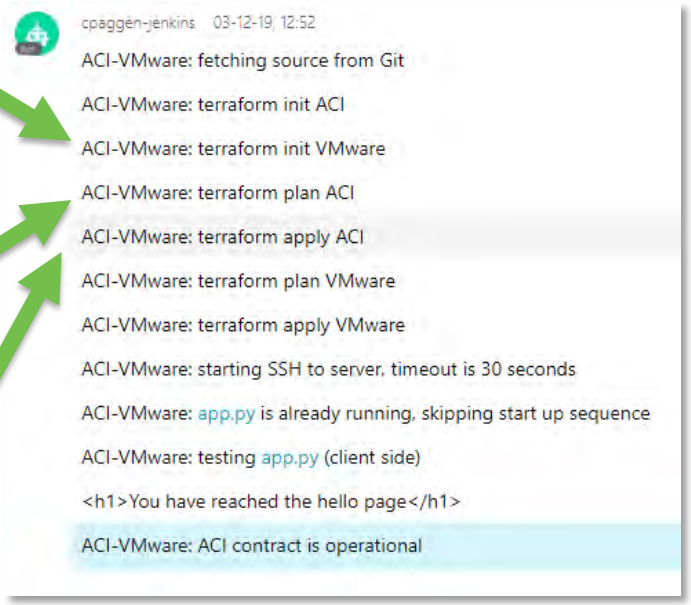
Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	Git code checkout	Terraform Init for the ACI plan	Terraform Init for the VMWARE plan	Terraform Plan for ACI	Apply ACI plan	Terraform plan for VMware	Apply VMware plan	Launch web server	SSH into Web Server	Validate ACI contract
Average stage times: (Average full run time: ~5min 14s)	1s	907ms	6s	11s	9s	5s	9s	4s	1min 42s	1s	19s	9s
#69 Dec 03 12:49 — 1 commit	2s	981ms	5s	23s	23s	14s	18s	12s	18s	2s	10s	11s

Continuous feedback in WebEx Teams

4

```
jenkinsFile-pipelineScript.groovy
56        steps {
57          dir('dev/VMWARE'){
58            ansiColor('xterm'){
59              webexTeamsTalk("ACI-VMware: terraform init VMware")
60              sh "terraform init -input=false -upgrade -force-copy
61              sh "echo \$PWD"
62            }
63          }
64        }
65      }
66
67      stage('Terraform Plan for ACI') {
68        steps {
69          dir('dev/ACI'){
70            ansiColor('xterm') {
71              webexTeamsTalk("ACI-VMware: terraform plan ACI")
72              sh 'terraform plan -out=plan'
73            }
74          }
75        }
76      }
77
78      stage('Apply ACI plan') {
79        steps {
80          dir('dev/ACI'){
81            ansiColor('xterm'){
82              webexTeamsTalk("ACI-VMware: terraform apply ACI")
83              sh 'terraform apply -auto-approve'
84            }
```



Deploying the app and validating it

5

```
stage('Launch web server'){
  steps{
    script{
  def client = [:]
  client.name = "1.1.1.100"
  client.host = "1.1.1.100"
  client.allowAnyHosts = true
  def server = [:]
  server.name = "1.1.1.200"
  server.host = "1.1.1.200"
  server.allowAnyHosts = true
  withCredentials([usernamePassword(credentialsId: 'sshUserAccount', passwor
      server.user = userName
      server.password = password
      client.user = userName
      client.password = password
      stage("SSH into Web Server") {
        script{
          try {
            webexTeamsTalk("ACI-VMware: starting SSH to server, timeout is 3
            timeout(time: 30, unit: 'SECONDS') {
              def appRuns = sshCommand remote: server, command: 'ps -u cisco -o cmd | grep
              if (appRuns == "1"){
                  webexTeamsTalk("ACI-VMware: starting app.py via SSH")
                  sshCommand remote: server, command: '/home/cisco/start_app.sh &'
                  webexTeamsTalk("ACI-VMware: app.py is launched")
              } else {
                    webexTeamsTalk("ACI-VMware: app.py is already running, skipping start
                }
            }
```

variables.tf ● app.py × vm.tf

VMWARE > app.py

```
1  #!/usr/bin/env python
2  from flask import Flask
3
4  app = Flask(__name__, static_url_path='/static')
5
6  @app.route('/hello')
7  def say_hello():
8      return '<h1>You have reached the hello page</h1>'
9
10 @app.route('/')
11 def default_greet():
12     return '<h1>Welcome to this dynamically created web server!</h1><p></p><img src="/static/
13
14 if __name__ == '__main__':
15     app.run(host='0.0.0.0', port=8080)
```

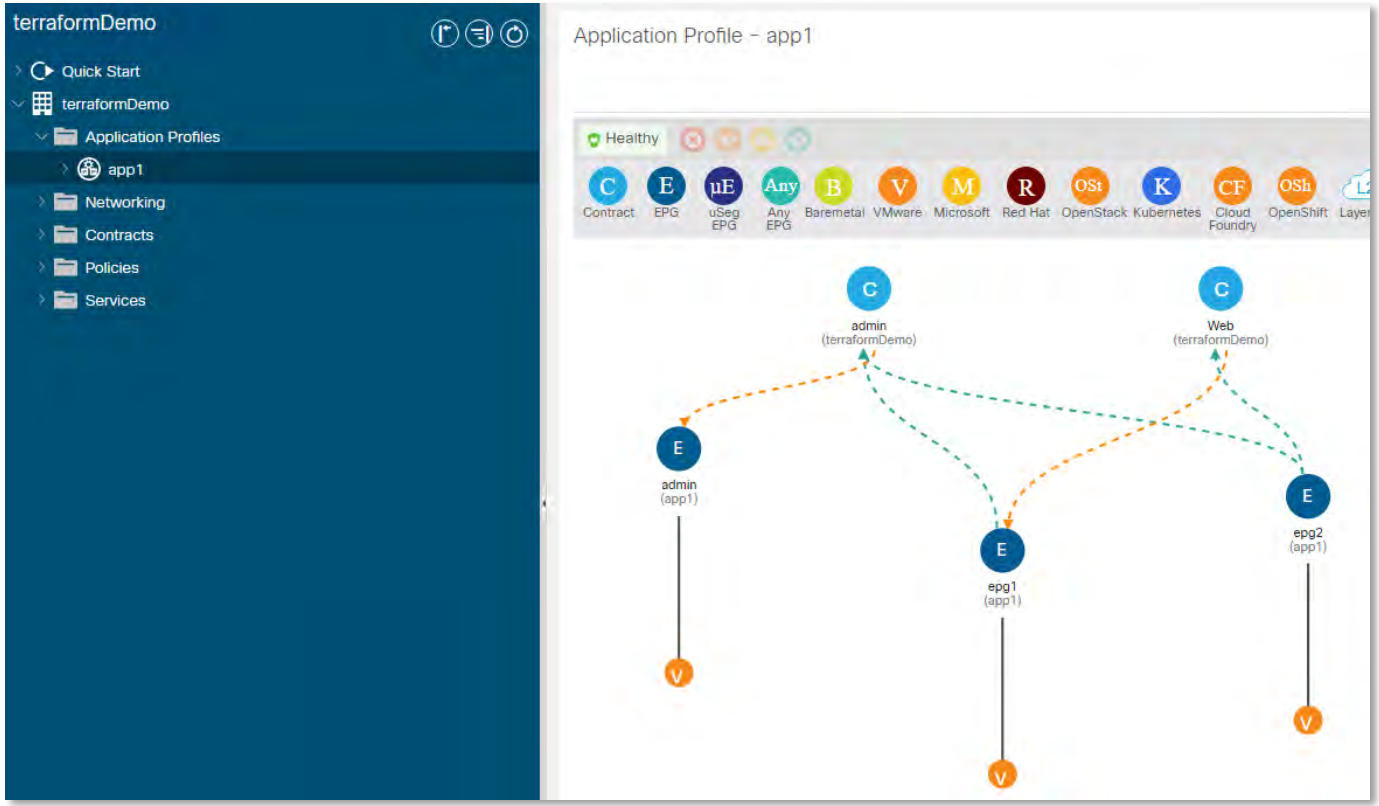
Deploying the app and validating it

5

```
stage('Launch web server'){
  steps{
    script{
  def client = [:]
  client.name = "1.1.1.100"
  client.host = "1.1.1.100"
  client.allowAnyHosts = true
  def server = [:]
  server.name = "1.1.1.200"
  server.host = "1.1.1.200"
  server.allowAnyHosts = true
  withCredentials([usernamePassword(credentialsId: 'sshUserAccount', passwordVariable: 'pa
      server.user = userName
      server.password = password
      client.user = userName
      client.password = password
      stage("SSH into Web Server") {
        script{
          try {
            webexTeamsTalk("ACI-VMware: starting SSH to server, timeout is 30 seconds")
            timeout(time: 30, unit: 'SECONDS') {
              def appRuns = sshCommand remote: server, command: 'ps -u cisco -o cmd | grep
              if (appRuns == "1"){
                  webexTeamsTalk("ACI-VMware: starting app.py via SSH")
                  sshCommand remote: server, command: '/home/cisco/start_app.sh &'
                  webexTeamsTalk("ACI-VMware: app.py is launched")
              } else {
                    webexTeamsTalk("ACI-VMware: app.py is already running, skipping start
                }
            }
```

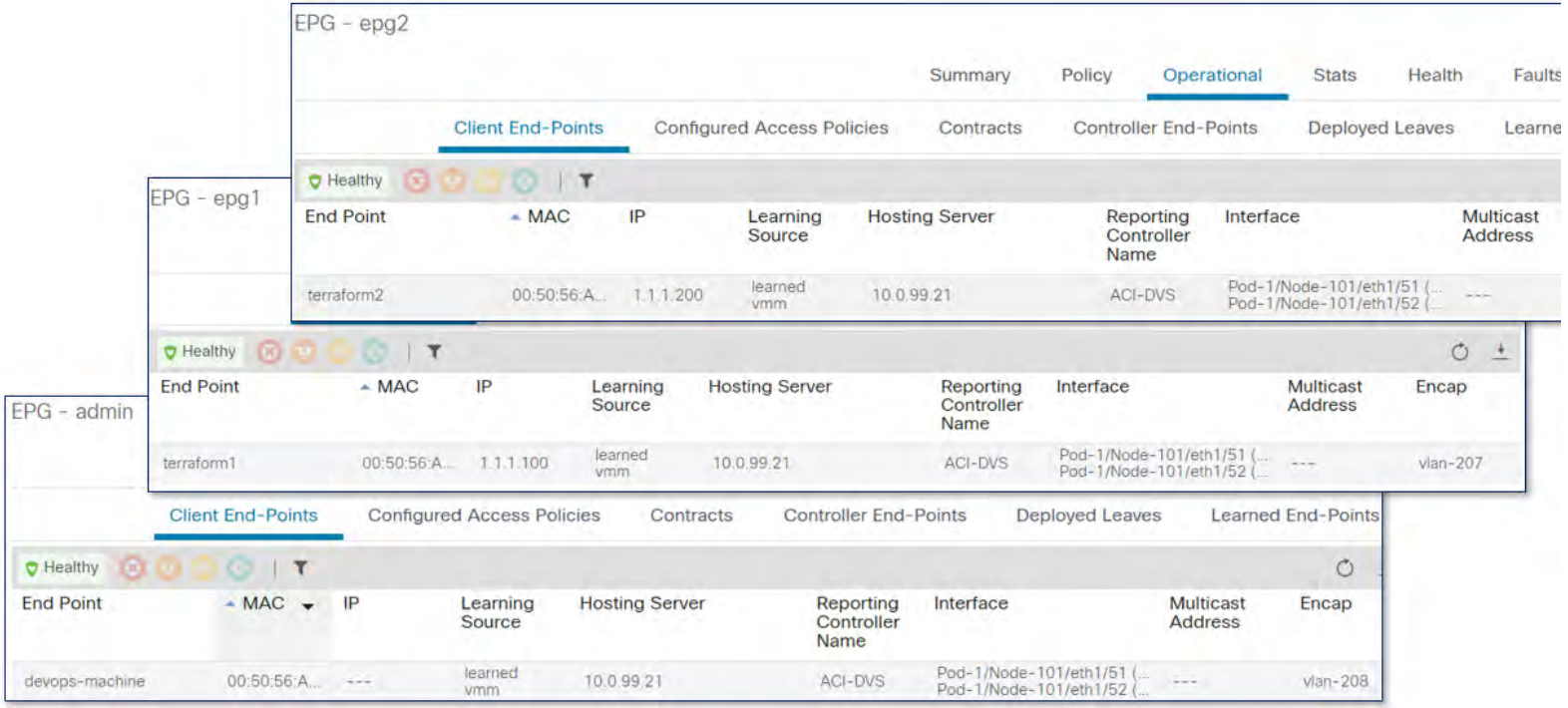
The ACI topology created by our code

5



Zero Trust – Segmentation Details

5



What to do now?

- ACI Ansible Modules Documentation Guide
  https://docs.ansible.com/ansible/latest/scenario_guides/guide_aci.html
- Cisco DevNet ACI and Ansible Learning Labs
  https://developer.cisco.com/learning/modules/ansible-aci-intro
- Cisco Collections on Ansible Galaxy
  https://galaxy.ansible.com/cisco
- Cisco ACI Terraform Provider
  https://www.terraform.io/docs/providers/aci/index.html

Thank you

Possibilities

#CiscoLive | #DevNetDay