cisco Live!

Let’s go

#CiscoLive

The bridge to possible

Getting Started with IaC and Terraform

Have No Fear, HCL is Here!

Quinn Snyder | Technical Advocate, Cisco Learning and Certifications
@qsnyder

CISCOU-1004

Cisco Webex App

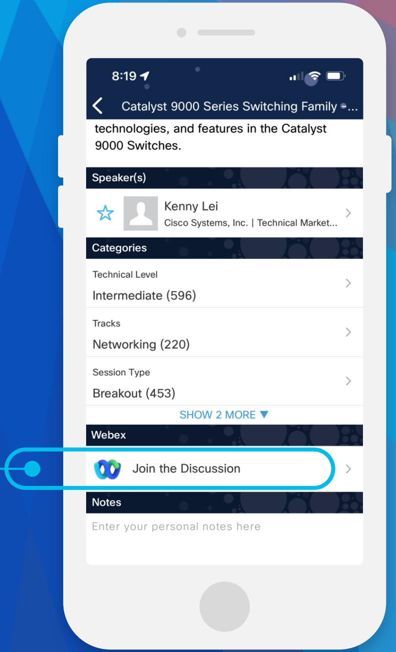
Questions?

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 9, 2023.


8:19
Catalyst 9000 Series Switching Family ...
technologies, and features in the Catalyst 9000 Switches.
Speaker(s)
Kenny Lei
Cisco Systems, Inc. | Technical Market...
Categories
Technical Level
Intermediate (596)
Tracks
Networking (220)
Session Type
Breakout (453)
SHOW 2 MORE
Webex
Join the Discussion
Notes
Enter your personal notes here

https://ciscolive.ciscoevents.com/ciscolivebot/#CISCOU-1004

Agenda

- Introduction, Background
- Terraform Workflow
- HCL Syntax, Control
- Demo
- Wrap-Up

“The act of managing and provisioning computer datacenters through machine readable definition files, rather than interactive configuration tools.”

~ Wikipedia Entry on IaC

Terraform background

HashiCorp

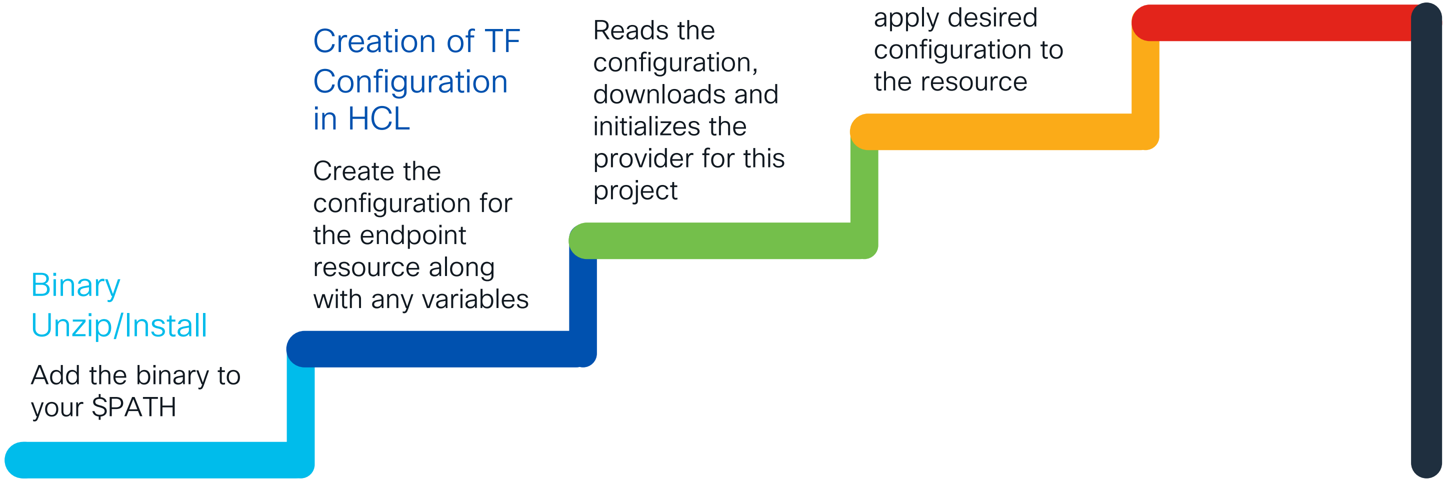
Developed by HashiCorp; initial release in July 2014

Designed to be a full Infrastructure as Code (IaC) management tool for datacenters

Completely written in Go, creating a single binary file

Fully declarative leveraging HashiCorp Configuration Language (HCL). Requires “provider” for target system/controller to be configured.

Typical Terraform workflow



Binary Unzip/Install

Add the binary to your $PATH

Creation of TF Configuration in HCL

Create the configuration for the endpoint resource along with any variables

Terraform Init

Reads the configuration, downloads and initializes the provider for this project

Terraform Plan

Determines the delta and steps to apply desired configuration to the resource

Terraform Apply

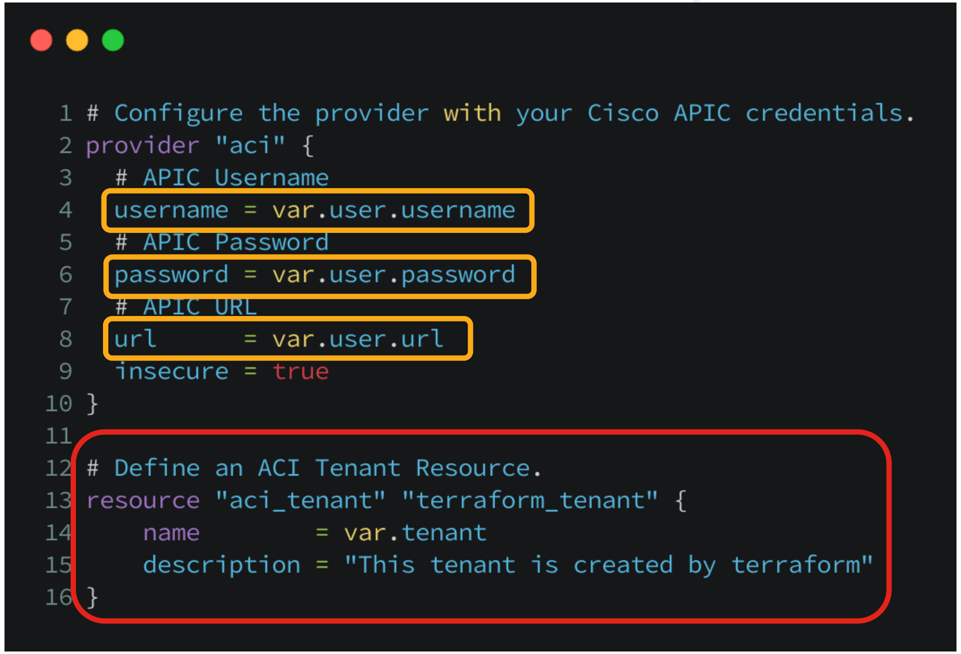
Applies the steps created from running the `plan` command

Terraform Destroy

Removes the configured state as defined within the HCL file

HashiCorp Configuration Language (HCL)

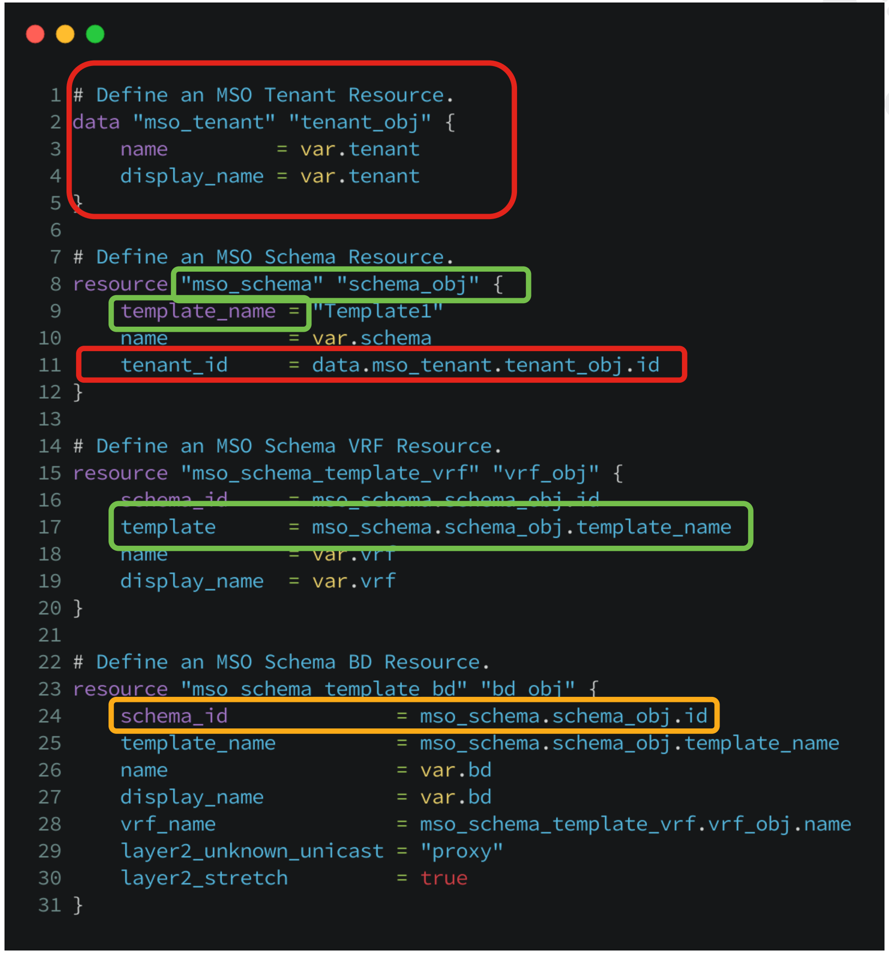
- JSON-ish in structure
- Composed of arguments and blocks
  - Arguments assign values to variables/attributes
  - Blocks serve as containers for other values and/or blocks (think: YANG style “containers”)
- Assignments for arguments can be static or set using variables



```
# Configure the provider with your Cisco APIC credentials.
provider "aci" {
  # APIC Username
  username = var.user.username
  # APIC Password
  password = var.user.password
  # APIC URL
  url      = var.user.url
  insecure = true
}

# Define an ACI Tenant Resource.
resource "aci_tenant" "terraform_tenant" {
    name        = var.tenant
    description = "This tenant is created by terraform"
}
```

HCL basics

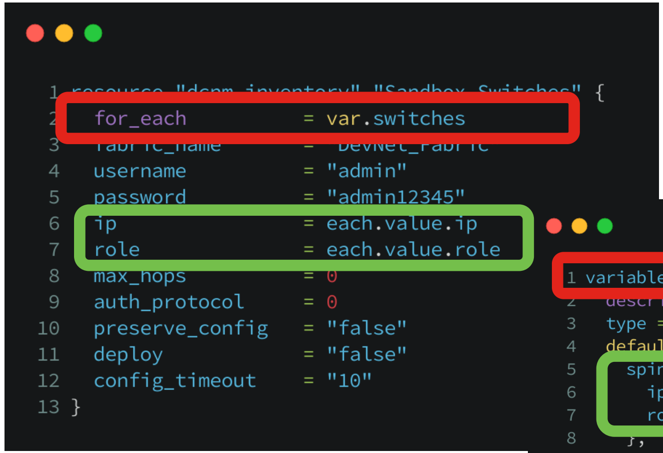
- All HCL plans of similar structure
- Possible to "read-in" for reuse without config
- Note the dotted notation nesting for relationships
- Sometimes variables are exposed without being declared as part of the provider



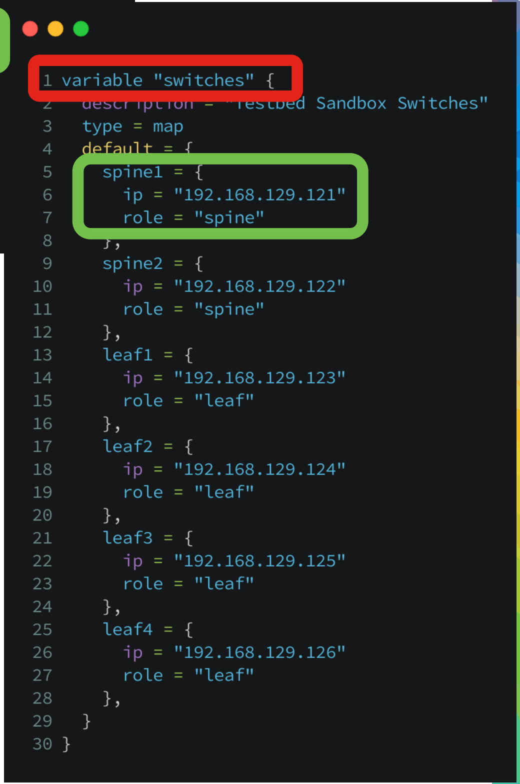
```
# Define an MSO Tenant Resource.
data "mso_tenant" "tenant_obj" {
    name         = var.tenant
    display_name = var.tenant
}

# Define an MSO Schema Resource.
resource "mso_schema" "schema_obj" {
    template_name = "Template1"
    name          = var.schema
    tenant_id     = data.mso_tenant.tenant_obj.id
}

# Define an MSO Schema VRF Resource.
resource "mso_schema_template_vrf" "vrf_obj" {
    schema_id     = mso_schema.schema_obj.id
    template      = mso_schema.schema_obj.template_name
    name          = var.vrf
    display_name  = var.vrf
}

# Define an MSO Schema BD Resource.
resource "mso_schema_template_bd" "bd_obj" {
    schema_id              = mso_schema.schema_obj.id
    template_name          = mso_schema.schema_obj.template_name
    name                   = var.bd
    display_name           = var.bd
    vrf_name               = mso_schema_template_vrf.vrf_obj.name
    layer2_unknown_unicast = "proxy"
    layer2_stretch         = true
}
```

HCL basics

- "Programmatic things" are limited similar to Ansible
- "For" loops defined by referencing top-level variable
- References to inner variables done through similar notation to "`with_items`" in Ansible; inner key-value assignments referenced in `main.tf`



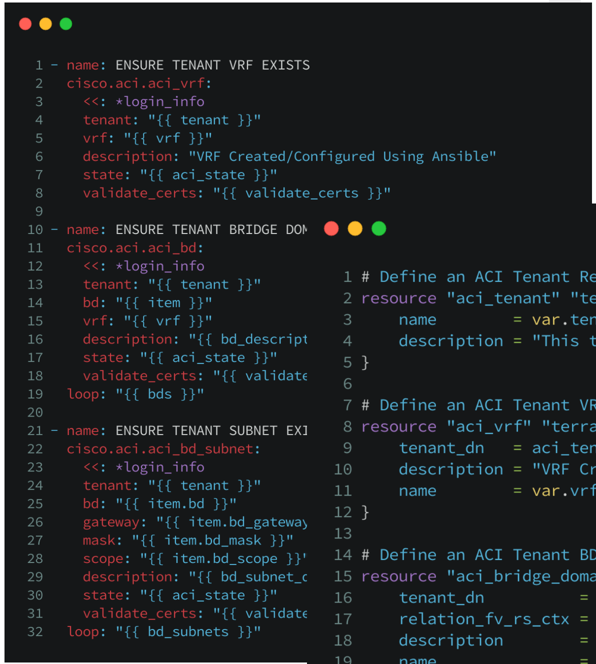
```
resource "dcnm_inventory" "Sandbox_Switches" {
  for_each                = var.switches
  fabric_name             = "DevNet_Fabric"
  username                = "admin"
  password                = "admin12345"
  ip                      = each.value.ip
  role                    = each.value.role
  max_hops                = 0
  auth_protocol           = 0
  preserve_config         = "false"
  deploy                  = "false"
  config_timeout          = "10"
}
```



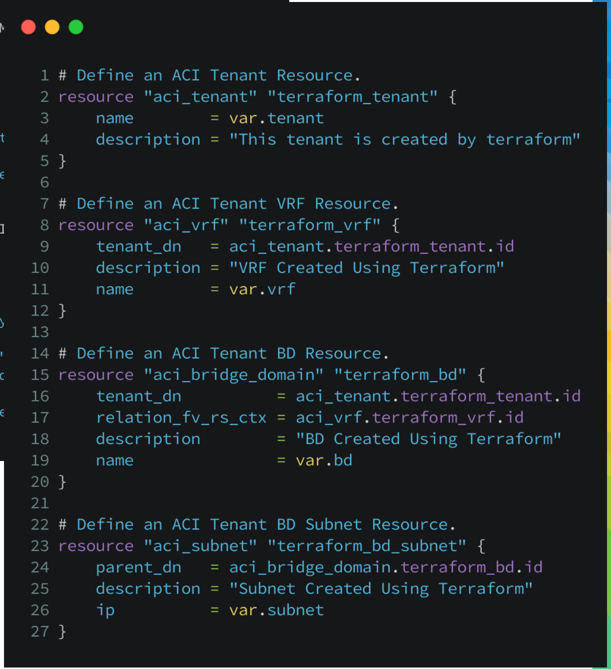
```
variable "switches" {
  description = "Testbed Sandbox Switches"
  type = map
  default = {
    spine1 = {
      ip = "192.168.129.121"
      role = "spine"
    },
    spine2 = {
      ip = "192.168.129.122"
      role = "spine"
    },
    leaf1 = {
      ip = "192.168.129.123"
      role = "leaf"
    },
    leaf2 = {
      ip = "192.168.129.124"
      role = "leaf"
    },
    leaf3 = {
      ip = "192.168.129.125"
      role = "leaf"
    },
    leaf4 = {
      ip = "192.168.129.126"
      role = "leaf"
    },
  }
}
```

Declarative vs procedural

- *Declarative* configurations define end-state in ”human language”.
  - Declarative can be “per task” or “per plan”
- *Procedural* configurations require *a priori* knowledge of configuration process to move to end state



```
- name: ENSURE TENANT VRF EXISTS
  cisco.aci.aci_vrf:
    <<: *login_info
    tenant: "{{ tenant }}"
    vrf: "{{ vrf }}"
    description: "VRF Created/Configured Using Ansible"
    state: "{{ aci_state }}"
    validate_certs: "{{ validate_certs }}"

- name: ENSURE TENANT BRIDGE DOM
  cisco.aci.aci_bd:
    <<: *login_info
    tenant: "{{ tenant }}"
    bd: "{{ item }}"
    vrf: "{{ vrf }}"
    description: "{{ bd_descript
    state: "{{ aci_state }}"
    validate_certs: "{{ validate
  loop: "{{ bds }}"

- name: ENSURE TENANT SUBNET EXI
  cisco.aci.aci_bd_subnet:
    <<: *login_info
    tenant: "{{ tenant }}"
    bd: "{{ item.bd }}"
    gateway: "{{ item.bd_gateway
    mask: "{{ item.bd_mask }}"
    scope: "{{ item.bd_scope }}'
    description: "{{ bd_subnet_c
    state: "{{ aci_state }}"
    validate_certs: "{{ validate
  loop: "{{ bd_subnets }}"
```



```
# Define an ACI Tenant Resource.
resource "aci_tenant" "terraform_tenant" {
    name        = var.tenant
    description = "This tenant is created by terraform"
}

# Define an ACI Tenant VRF Resource.
resource "aci_vrf" "terraform_vrf" {
    tenant_dn   = aci_tenant.terraform_tenant.id
    description = "VRF Created Using Terraform"
    name        = var.vrf
}

# Define an ACI Tenant BD Resource.
resource "aci_bridge_domain" "terraform_bd" {
    tenant_dn          = aci_tenant.terraform_tenant.id
    relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
    description        = "BD Created Using Terraform"
    name               = var.bd
}

# Define an ACI Tenant BD Subnet Resource.
resource "aci_subnet" "terraform_bd_subnet" {
    parent_dn   = aci_bridge_domain.terraform_bd.id
    description = "Subnet Created Using Terraform"
    ip          = var.subnet
}
```

Dude, where's my code?!

Just because its not Python/Go/Node/etc; doesn't mean its not code!

- Declarative configurations are code!
- Placing all scaffolded config in HCL creates an archive of config intent
- Archived intent can be stored in VCS/SCM for versioning
- IaC + VCS = Most of a CI/CD network configuration system!

Demo

https://github.com/qsnyder/ciscou-1004

Technical session surveys

- Attendees who fill out a minimum of four session surveys and the overall event survey will get Cisco Live branded socks!
- Attendees will also earn 100 points in the Cisco Live Game for every survey completed.
- These points help you get on the leaderboard and increase your chances of winning daily and grand prizes.

Pay for Learning with Cisco Learning Credits

(CLCs) are prepaid training vouchers redeemed directly with Cisco.

Cisco learning and certifications

From technology training and team development to Cisco certifications and learning plans, let us help you empower your business and career. www.cisco.com/go/certs

Learn

Cisco U.
IT learning hub that guides teams and learners toward their goals

Cisco Digital Learning
Subscription-based product, technology, and certification training

Cisco Modeling Labs
Network simulation platform for design, testing, and troubleshooting

Cisco Learning Network
Resource community portal for certifications and learning

Train

Cisco Training Bootcamps
Intensive team & individual automation and technology training programs

Cisco Learning Partner Program
Authorized training partners supporting Cisco technology and career certifications

Cisco Instructor-led and Virtual Instructor-led training
Accelerated curriculum of product, technology, and certification courses

Certify

Cisco Certifications and Specialist Certifications
Award-winning certification program empowers students and IT Professionals to advance their technical careers

Cisco Guided Study Groups
180-day certification prep program with learning and support

Cisco Continuing Education Program
Recertification training options for Cisco certified individuals

Here at the event? Visit us at **The Learning and Certifications lounge at the World of Solutions**

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

cisco Live!

The bridge to possible

Thank you

#CiscoLive

cisco Live!

Let’s go

#CiscoLive