Introduction to Infrastructure as Code for ACI using Terraform

Thomas Renzy - Customer Delivery Architect CX

BRKDCN-2607

Agenda

- What is Infrastructure as Code?
- Terraform Concepts
- Demo
- Key Takeaways

Webex App

Questions?

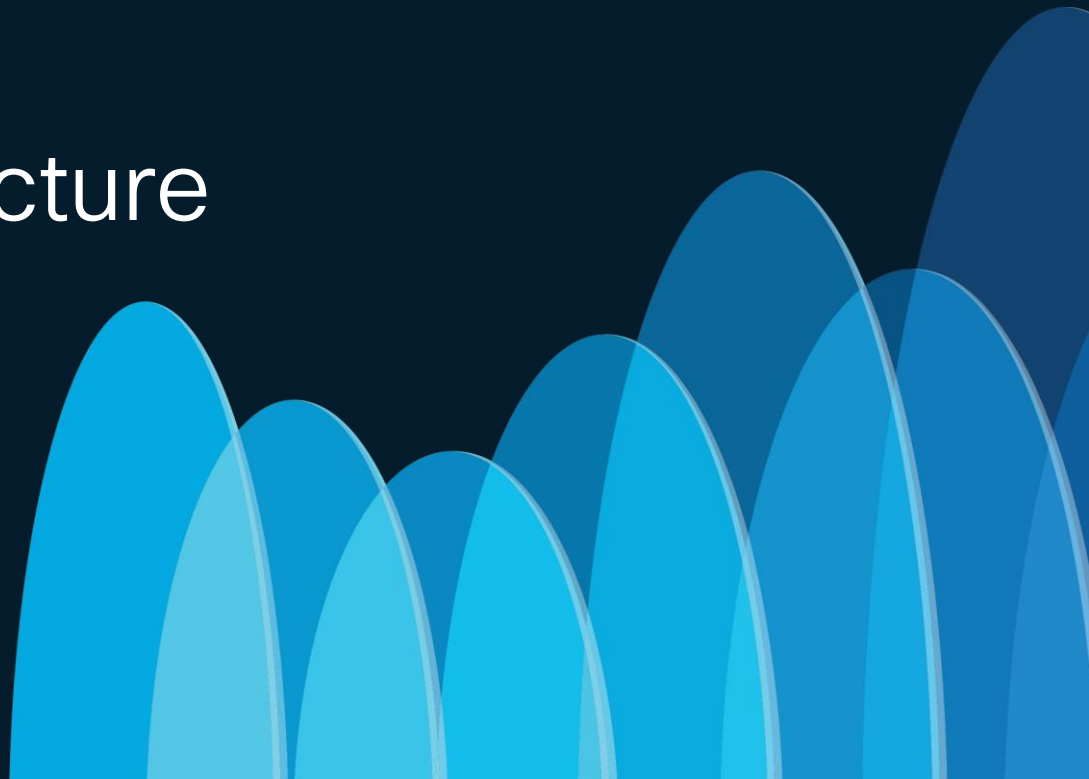
Use the Webex app to chat with the speaker after the session

How

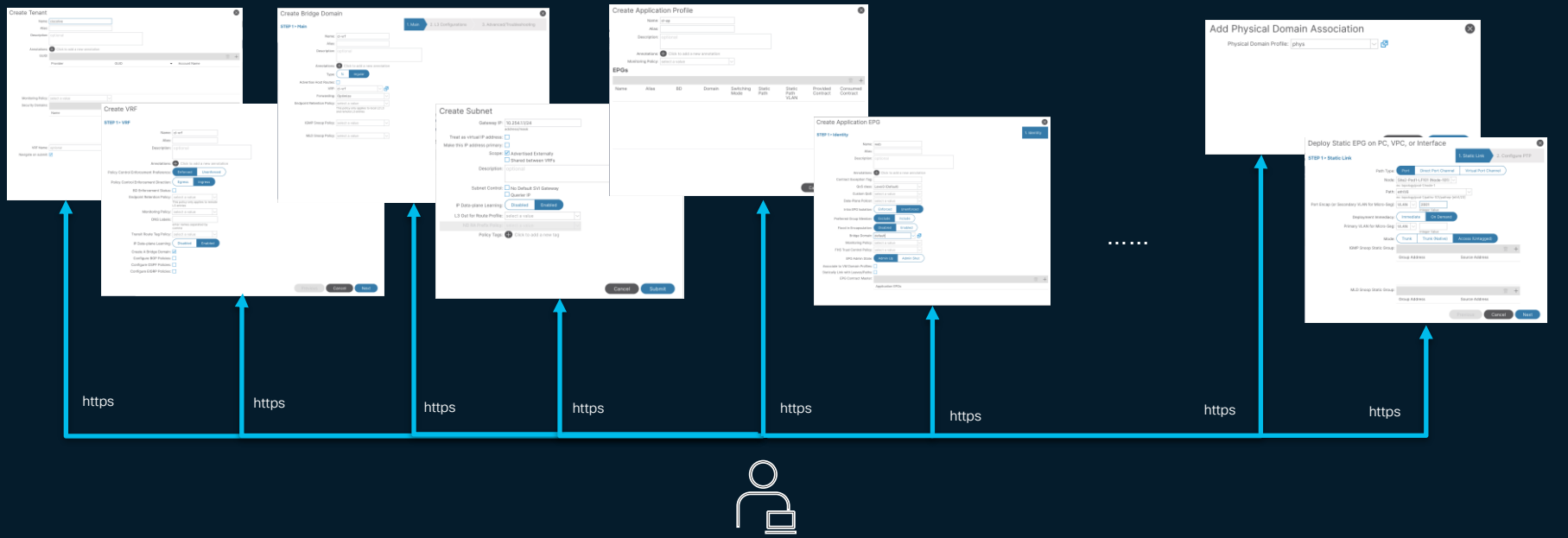
1. Find this session in the Cisco Events mobile app
2. Click “Join the Discussion”
3. Install the Webex app or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until February 28, 2025.

What is Infrastructure as Code?



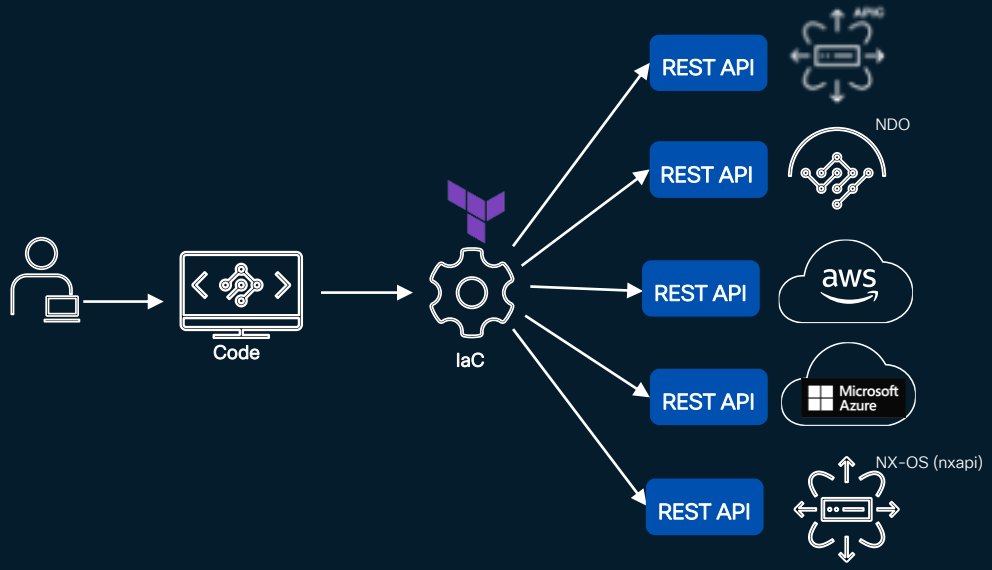
How we currently manage infrastructure



"ClickOps" – manual management of infrastructure from a GUI

What is Infrastructure as Code(IaC)?

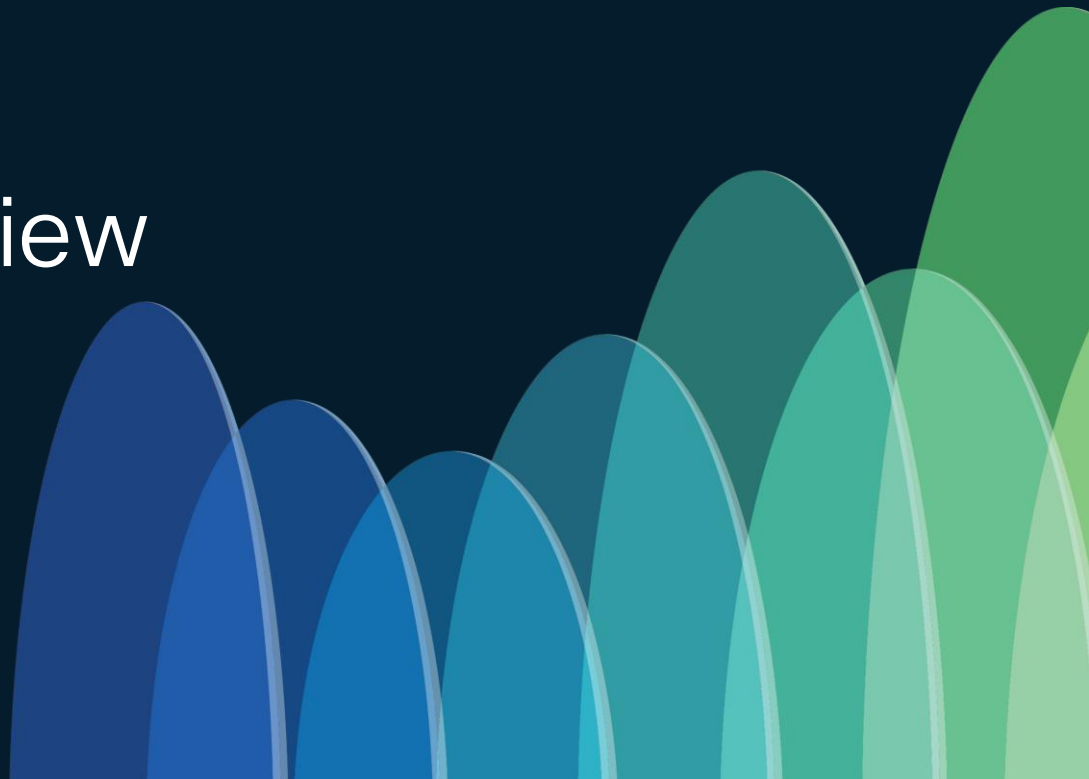
Define, configure, and manage Infrastructure through code



Benefits

Speed and Scale
Repeatability
Shareability
Consistency
Documentation
Cost Savings

Terraform Overview

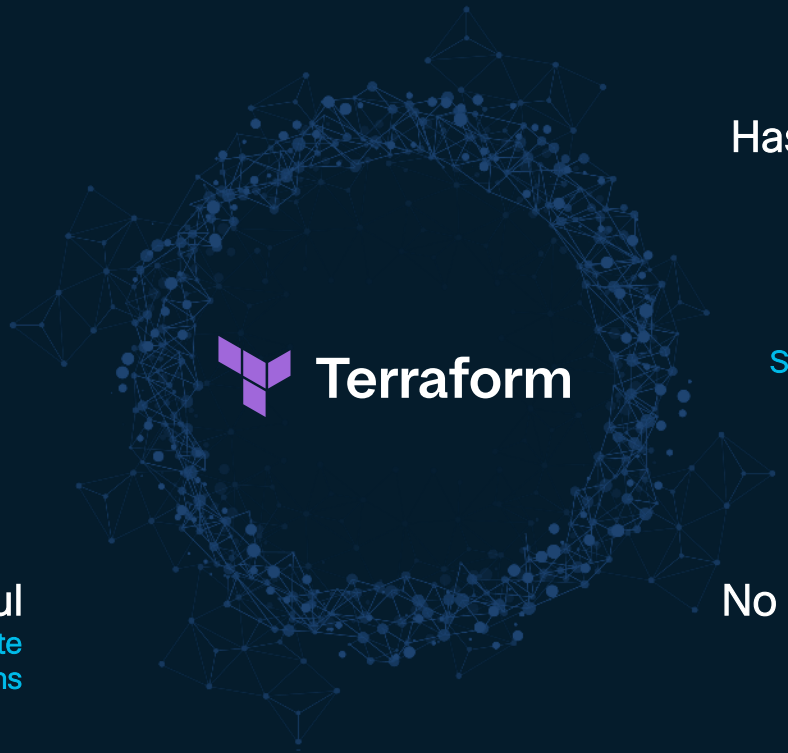


What is Terraform?

Open Source
Infrastructure Provisioning

Single Binary
Multi Operating Systems

Declarative and Stateful
Define End State
Track Infrastructure it provisions



HashiCorp Configuration Language

API REST Interaction
Same REST API Interface as GUI

No Special Programming Skills required

HashiCorp Configuration Language (HCL)

- Declarative Language - HashiCorp
  - Primary user interface
  - Based on HCL2 toolkit
- Designed for Infrastructure Provisioning

- Built around
  - Blocks – Resources/Data Sources/Modules
  - Parameters/Arguments
  - Blocks open "{" and close "}"

```
Block_Type Block_Label {
  # this is a comment
  // This is also a comment
  /*
     This is a comment too
  */
  parameter1 = value
  parameter2 = value
}
```

Terraform Core – Single binary

- Reading .tf & .tfvars files
- State Management
- Building Graph
- Plan execution

```
> terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
…

> terraform -version
Terraform v1.9.3
on darwin_arm64

Your version of Terraform is out of date! The latest version
is 1.10.5. You can update by downloading from https://www.terraform.io/downloads.html
```

Terraform Block Configuration

- Configures Terraform behavior
- Version of Terraform binary
- Specify the Providers you want

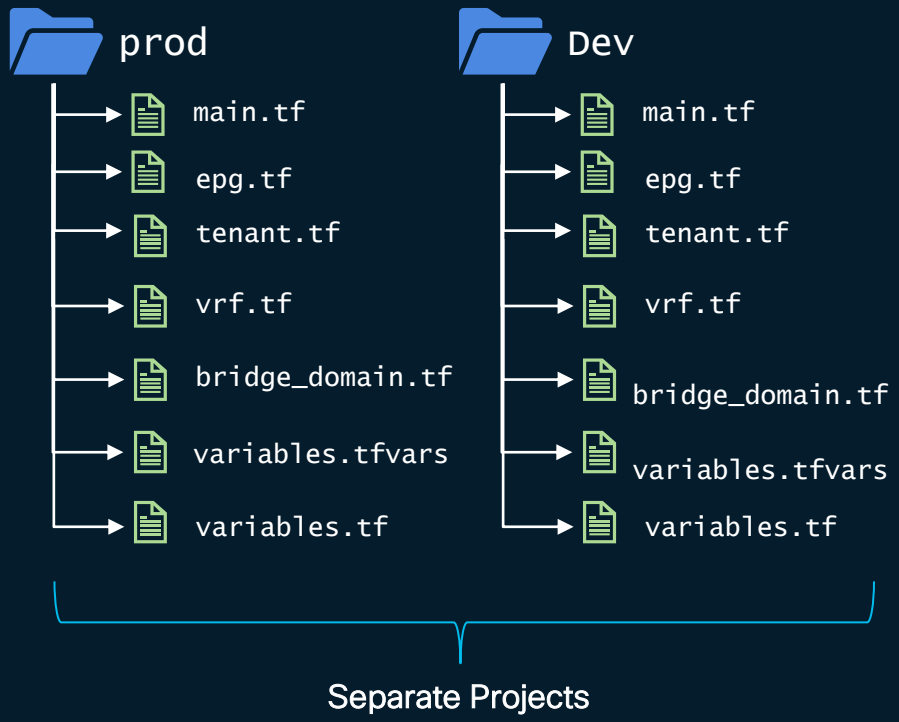
```
terraform {
  required_version ">= 1.6.0"
  required_providers {
    aci = {
      source = "CiscoDevNet/aci"
      version = "2.11.1"
    }
    mso = {
      source = "CiscoDevNet/mso"
  }
}
```

Specify the version of the Binary (version constraint)

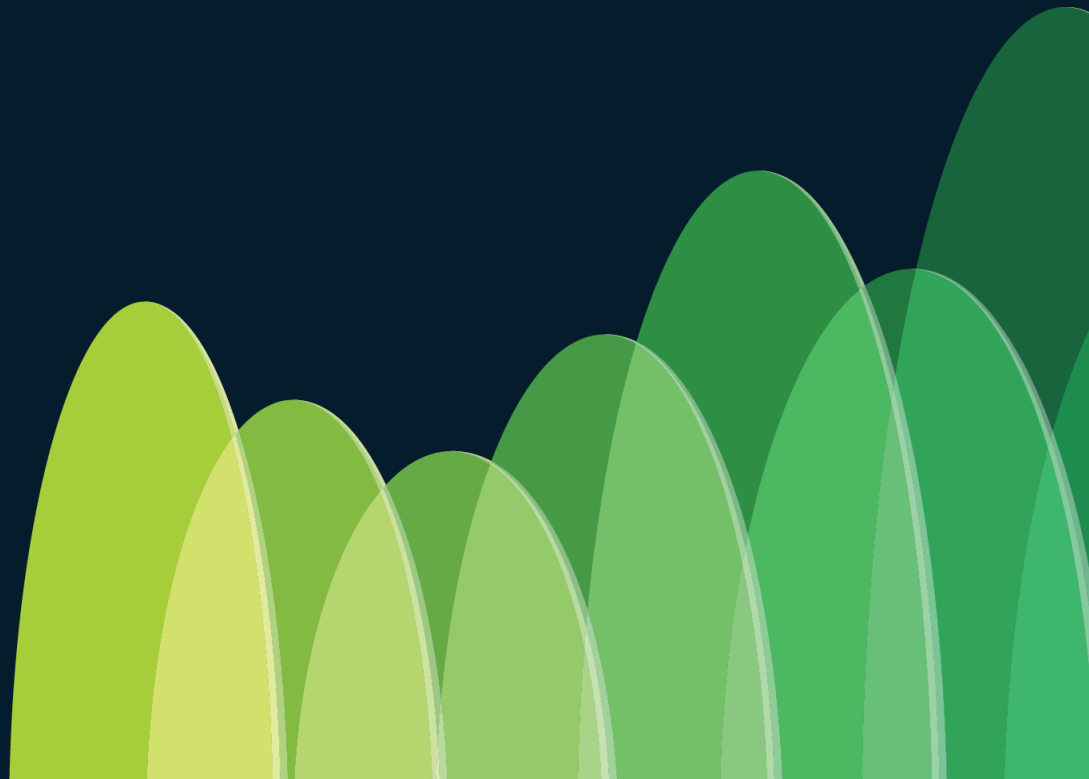
Provider information

Terraform Project

- Collection of Configurations
  - Single directory – Terraform runs in
  - What you want to provision (intent)
  - Can be in single file or multiple files
  - Declarative – No need for order of operations

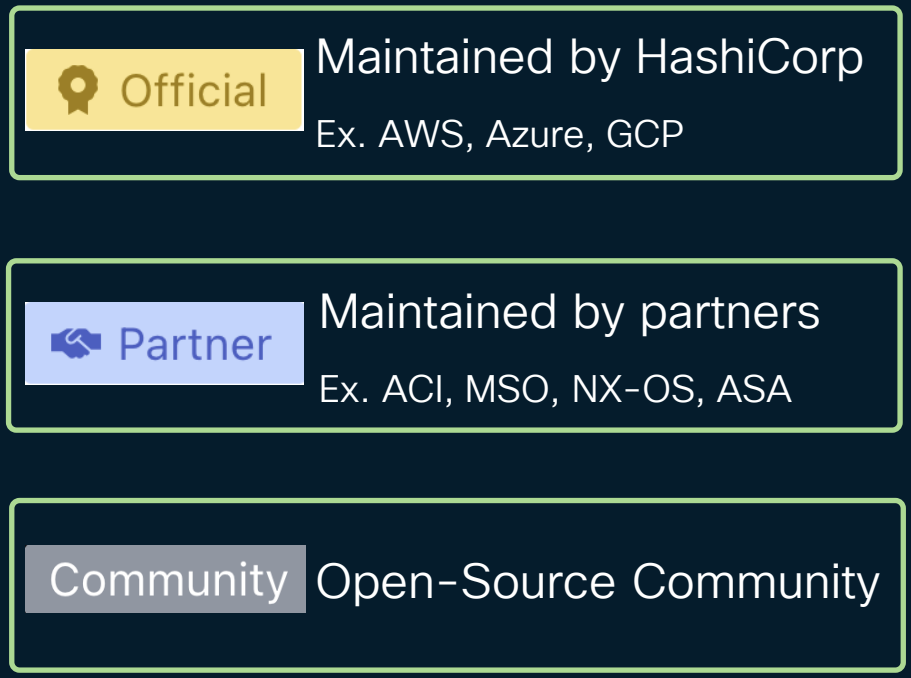


Providers



Terraform Providers

- Terraform Binary doesn't know ACI/NDO
- Providers Understand API interactions
  - APIC and MSO REST API calls
- One-to-one relationship with vendor
- Relies on specific plugins
  - Installed via `terraform init`



Provider Configurations

```
terraform {
  required_version ">= 1.6.0"
  required_providers {
    aci = {
      source = "CiscoDevNet/aci"
      version = "2.13.0"
    }
  }
}
```

Provider, source, & version
Can specify Multiple Providers

Provider Block (ACI)

**Any changes to provider config - `terraform init`

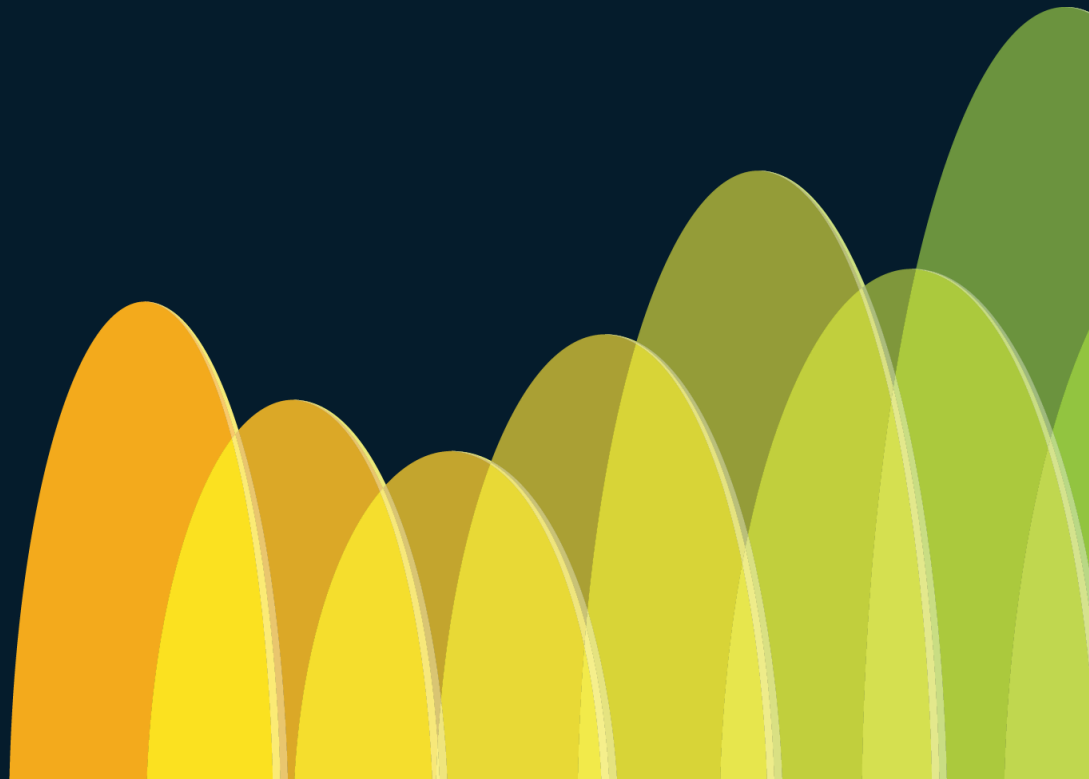
```
provider "aci" {
  # cisco-aci user name
  username = var.aci_username
  # Password
  password = var.aci_password
  # cisco-aci url
  url      = "https://172.31.2.31/"
  insecure = true
}
```

Authentication

APIC URL

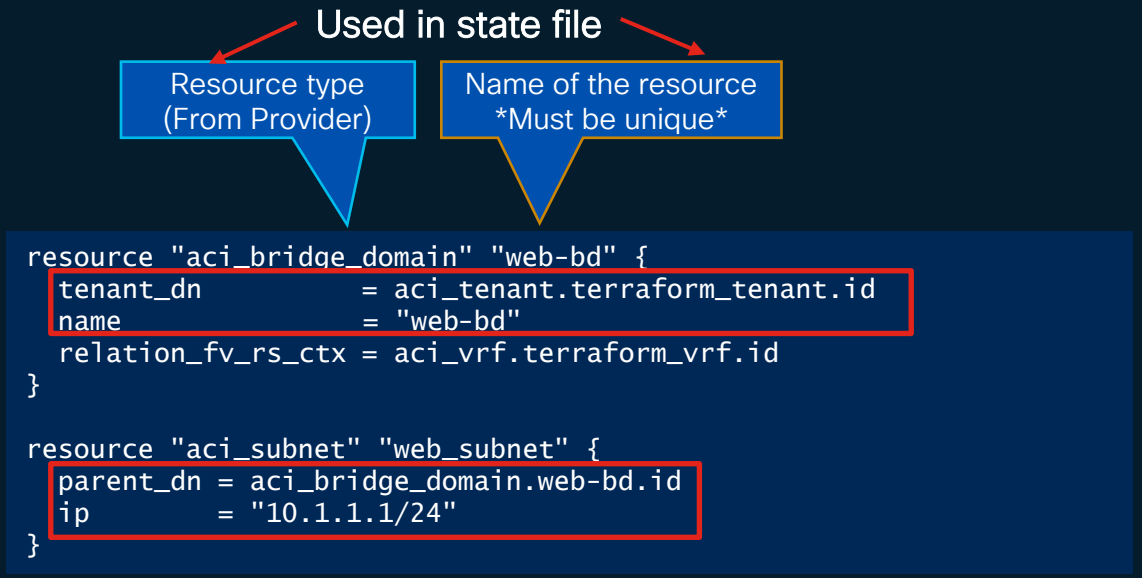
No SSL Cert Validation

Resources and Data Sources



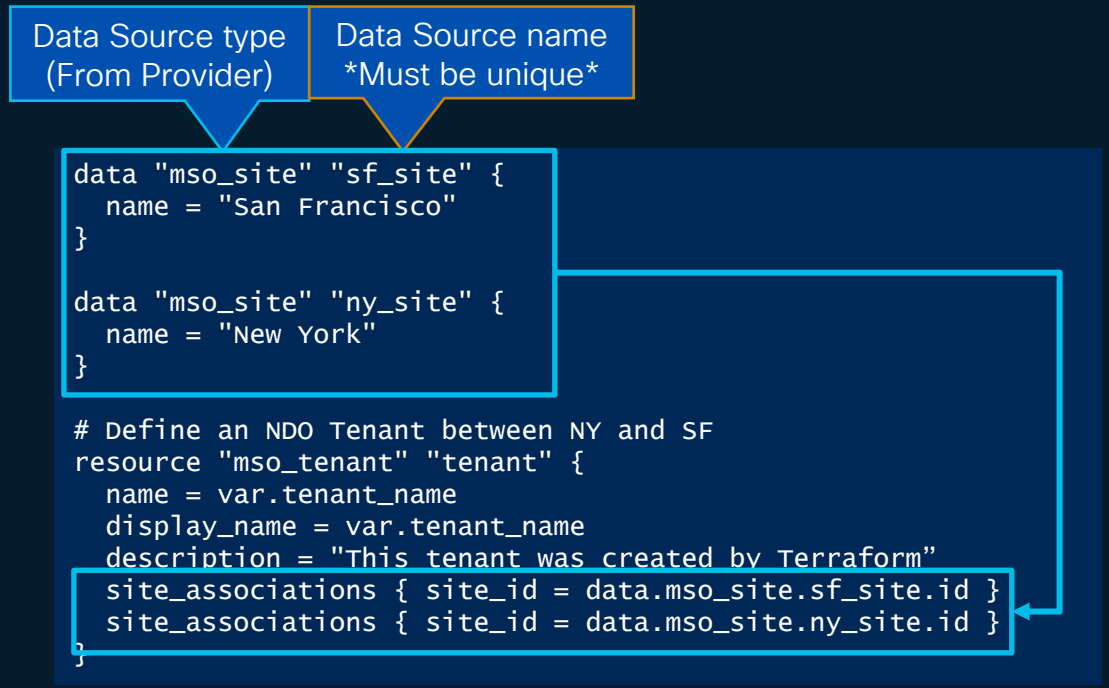
Terraform Resources

- Represents Infrastructure to be managed
  - Manages an Infrastructure element
  - Many resources available
- Attributes
  - Required
  - Optional



Terraform Data Sources

- Allows data to be fetched for use elsewhere in Terraform configuration
  - Lots of data sources available



aci_rest_managed

- When there isn't a Resource
- Manages Objects via REST API calls with no resource
- Can reconcile state information

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

Terraform Dependencies

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

resource "aci_vrf" "terraform_vrf" {
  tenant_dn   = aci_tenant.terraform_tenant.id
  name        = "PROD-VRF"
  description = "Created with Terraform"
}

resource "aci_tenant" "terraform_tenant" {
  name        = "PROD"
  description = "Created with Terraform"
}

resource "aci_application_profile" "terraform_ap" {
  tenant_dn = aci_tenant.terraform_tenant.id
  name      = "PROD-AP"
}


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"
}
```

- Implicit Dependency
  - Automatically detected by Terraform
  - Establishes order resources are created/modified/deleted
  - Builds Graph – Direct Acylic Graph
- Explicit Dependency
  - Set up manually between resources
  - Leverages **depends_on**

Variables

Variables

- Hard coded values in resources

```
resource "aci_tenant" "terraform_tenant" {
  name        = ”PROD”
  description = "Created with Terraform"
}

resource "aci_application_profile" "terraform_ap" {
  tenant_dn = aci_tenant.terraform_tenant.id
  name      = “PROD-AP”
}

resource "aci_vrf" "terraform_vrf" {
  tenant_dn   = aci_tenant.terraform_tenant.id
  name        = “PROD-VRF”
  description = "Created with Terraform"
}

resource "aci_any" "example_vzany" {
  vrf_dn       = aci_vrf.terraform_vrf.id
  pref_gr_memb = "enabled"
}
```

```
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"
}

resource "aci_application_epg" ”prod_epg" {
application_profile_dn = aci_application_profile.terraform_ap.id
  name                 = “prod-epg”
  relation_fv_rs_bd    = aci_bridge_domain.web-bd.id
  pref_gr_memb         = "include"
}
```

Make code reusable – Use variables

Variables

- Value substitution - Makes code reusable
- Defined in a separate file (variables.tf & *.tfvars)
- Variables have
  - Type – string, number, bool, list, set, map, object
  - Default value
    - If no value is set, user will be prompted for value
  - Description
  - Variables have precedence
    - Variables can be set, but overridden
- Environment Variables – TF_VAR

Variables Definition File

(variables.tf)

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

variable "vrf_name" {
   default = "Cisco_vrf"
}
```

Variables Assignment

terraform.tfvars or <name>.auto.tfvars

(Overrides Variable file default)

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

Environment Variables

```
> export TF_VAR_aci_username=admin
> export TF_VAR_aci_password=Cisco
```

Variables

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

resource "aci_application_profile" "terraform_ap" {
  tenant_dn = aci_tenant.terraform_tenant.id
  name      = var.ap_name
}

resource "aci_vrf" "terraform_vrf" {
  tenant_dn   = aci_tenant.terraform_tenant.id
  name        = var.vrf_name
  description = "Created with Terraform"
}

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

resource "aci_subnet" "web_subnet" {
  parent_dn = aci_bridge_domain.web-bd.id
  ip        = var.bd_subnet
}
```

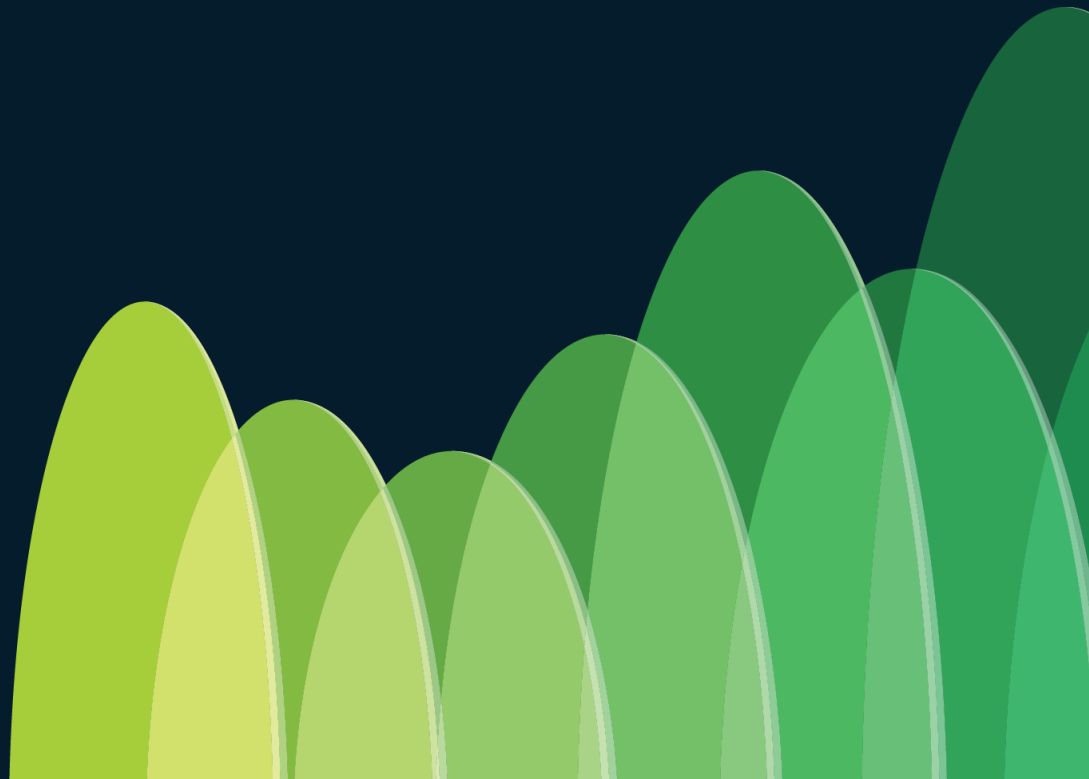
variables.tf

```
variable "tenant_name"{
   type = string
   default = "Cisco"
   description = "Tenant name"
}

variable "vrf_name"{
   default = "cisco_vrf"
}

 variable "ap_name"{
   default = "Cisco_ap"
}

 variable "bd_name"{
   default = "Cisco_bd"
}

 variable "bd_subnet"{
   default = "10.1.1.1/24"
}
```

terraform.tfvars
(Overrides Variable file default)

```
tenant_name = "ciscolive"
vrf_name = "ciscolive_vrf"
ap_name = "ciscolive_ap"
bd_name = "ciscolive_bd"
Bd_subnet = "192.168.100.1/24"
```

Iteration



Iteration - count

- Create multiple instances of object
- Can create code x number of objects

```
resource "aci_bridge_domain" "count_bd_1" {
  tenant_dn    = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform count"
  name         = bd_prod
  arp_flood    = "yes"
}

resource "aci_bridge_domain" "count_bd_2" {
  tenant_dn    = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform count"
  name         = bd_dev
  arp_flood    = "yes"
}

resource "aci_bridge_domain" "count_bd_3" {
  tenant_dn    = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform count"
  name         = bd_qa
  arp_flood    = "yes"
}
```

- Use Terraform count
  - Add number of resources based on count
  - Index based
- More efficient code

```
variable bridge_domains {
  description = "BD names"
  type        = list(string)
  default = [ "prod", "dev", "qa"]
}

resource "aci_bridge_domain" "count_bd" {
  count        = length(var.bridge_domains)
  tenant_dn    = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.count_vrf.id
  description = "Created with Terraform"
  name         = "bd_${var.bridge_domains[count.index]}"
  arp_flood    = "yes"
}
```

Iteration – for_each

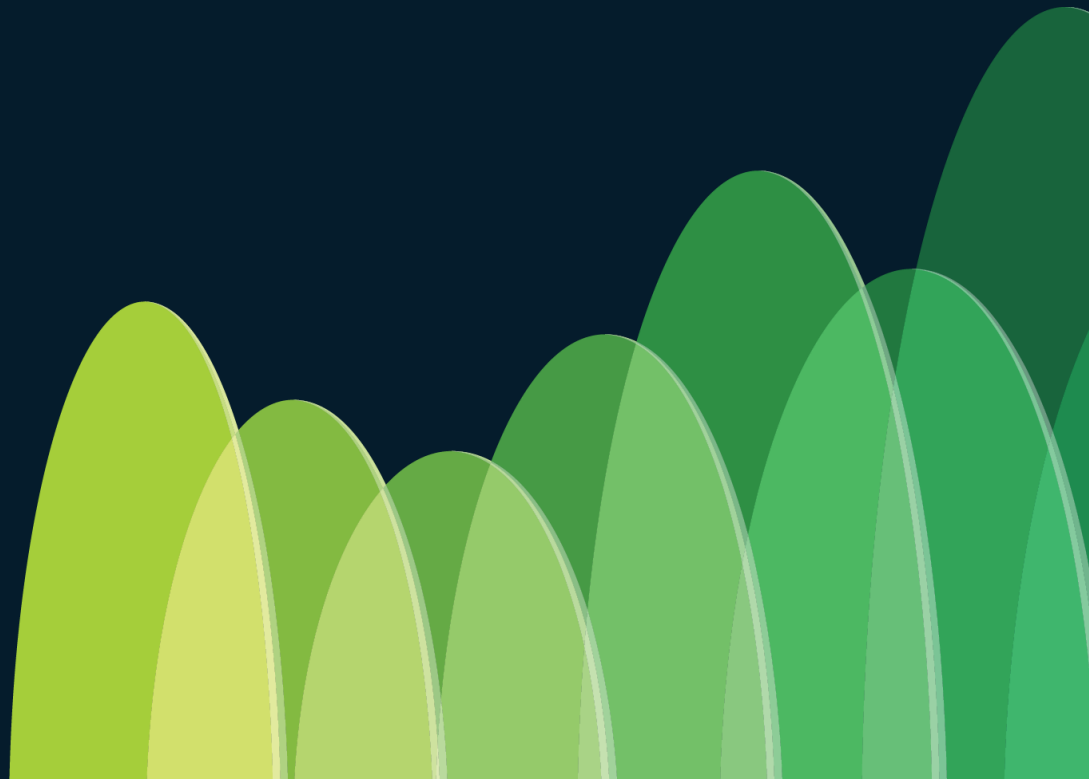
- Create multiple instances of object
- Use `for_each`

```
resource "aci_bridge_domain" "prod" {
  tenant_dn   = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform"
  name        = "prod-bd"
  arp_flood   = "yes"
}

resource "aci_bridge_domain" "dev" {
  tenant_dn   = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform"
  name        = "dev-bd"
  arp_flood   = "yes"
}

resource "aci_bridge_domain" "qa" {
  tenant_dn   = aci_tenant.count_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform"
  name        = "qa-bd"
  arp_flood   = "yes"
}
```

- Key Based
- Create objects based on a set or map
- Can also use list
  - Use `toset` to convert from list

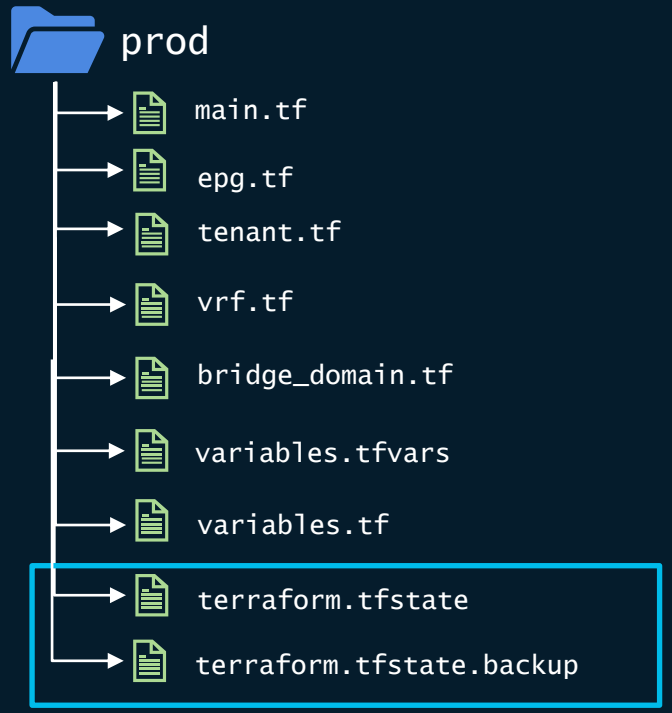
```
variable "bridge_domains" {
description = "BD names"
  type        = list(string)
  default     = ["prod-bd", "dev-bd", "qa-bd"]
}

resource "aci_bridge_domain" "for_each_bd" {
  for_each = toset(var.bridge_domains)
  tenant_dn   = aci_tenant.tf_tenant.id
  relation_fv_rs_ctx = aci_vrf.terraform_vrf.id
  description = "Created with Terraform"
  name        = each.key
  arp_flood   = "yes"
}
```

Terraform State



Terraform State

- Terraform is stateful
  - `terraform.tfstate`
  - Tracks objects it builds
  - Source of everything it knows about
  - json file format
  - Stored inside working directory
  - Doesn't tend to allow collaboration



Terraform State File

Note: Never modify state file directly!


```
{
  "version": 4,
  "terraform_version": "1.9.3",
  "serial": 2,
  "lineage": "d8cb29a0-21cd-eb5b-ffbf-8904d802f304",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "aci_application_epg",
      "name": ”prod_epg",
      "provider": "provider[\"registry.terraform.io/ciscodevnet/aci\"]",
      "instances": [
        {
          "index_key": "dev",
          "schema_version": 1,
          "attributes": {
            "annotation": "orchestrator:terraform",
            "application_profile_dn": "uni/tn-My_Prod_App/ap-Prod_App_ap",
            "description": "",
            "exception_tag": "",
            "flood_on_encap": "disabled",
            "fwd_ctrl": "none",
            "has_mcast_source": "no",
            "id": "uni/tn-My_Prod_App/ap-Prod_App_ap/epg-dev",
"is_attr_based_epg": "no",
            "match_t": "AtleastOne",
            "name": "dev",
            "name_alias": "",
            "pc_enf_pref": "unenforced",
            "pref_gr_memb": "exclude",
```

Terraform State Backend

- Terraform State Backend
  - Store state remotely
  - Easier Collaboration
  - State locking
- Backends Supported
  - Local (default)
  - AWS S3
  - AzureRM
  - gcs
- State locking
  - Dynamodb (AWS)

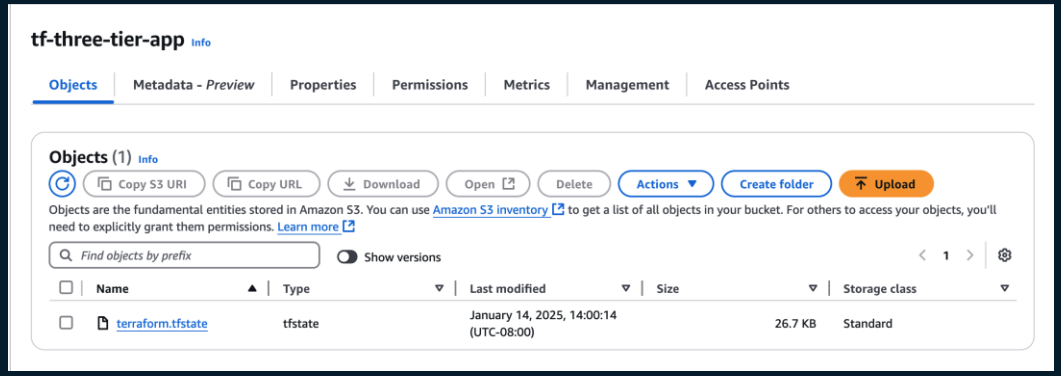
Note: Must re-run `terraform init (-reconfigure)`

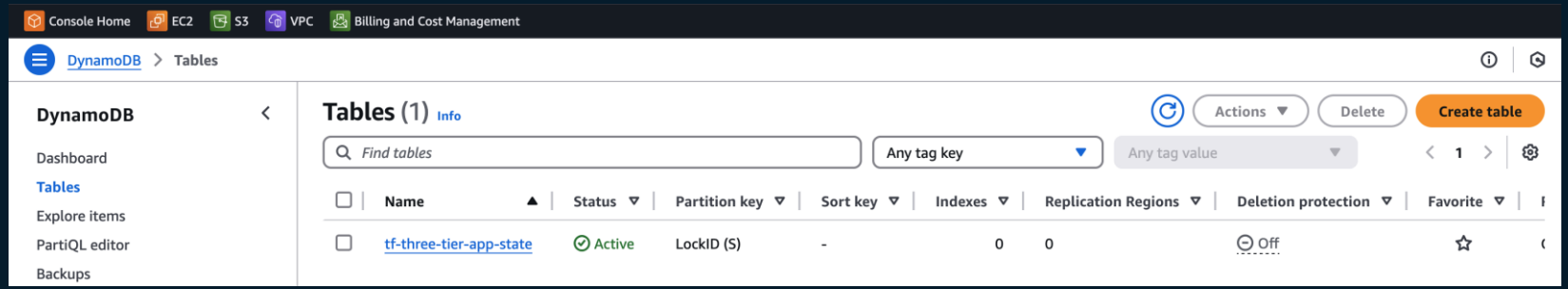
```
terraform {
  required_providers {
    aci = {
      source  = "CiscoDevNet/aci"
      version = "2.11.1"
    }
  }
  backend "s3" {
    bucket = "tf-three-tier-app"
    key = "terraform.tfstate"
    region = "us-east-1"
    dynamodb_table = "tf-three-tier-app-state"
  }
}
```

Migrate Existing state

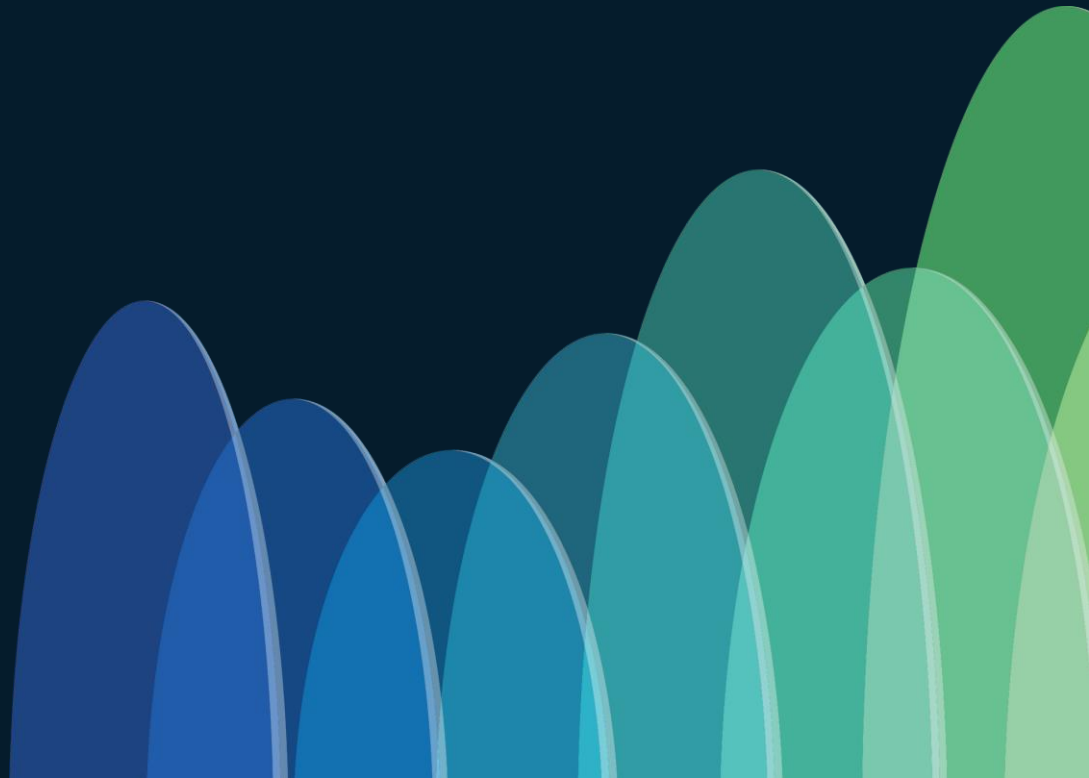
```
terraform init -migrate-state
```

Terraform State Backend AWS S3





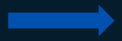
Terraform CLI commands



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 extension)
- Determines what actions are necessary to achieve the desired state
- Preview your changes – no changes made (Dry Run)

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

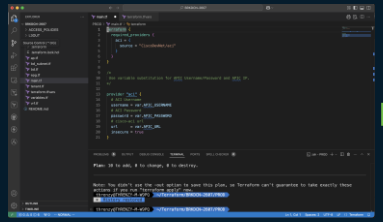
Terraform WorkFlow

Write
Provider Info
Configuration
Desired State

Initialize
Download Plugins
From Registry

Plan
Preview changes
Adjust
configurations

Apply
Provision or modify
infrastructure



```
> terraform init
```

```
> terraform plan
```

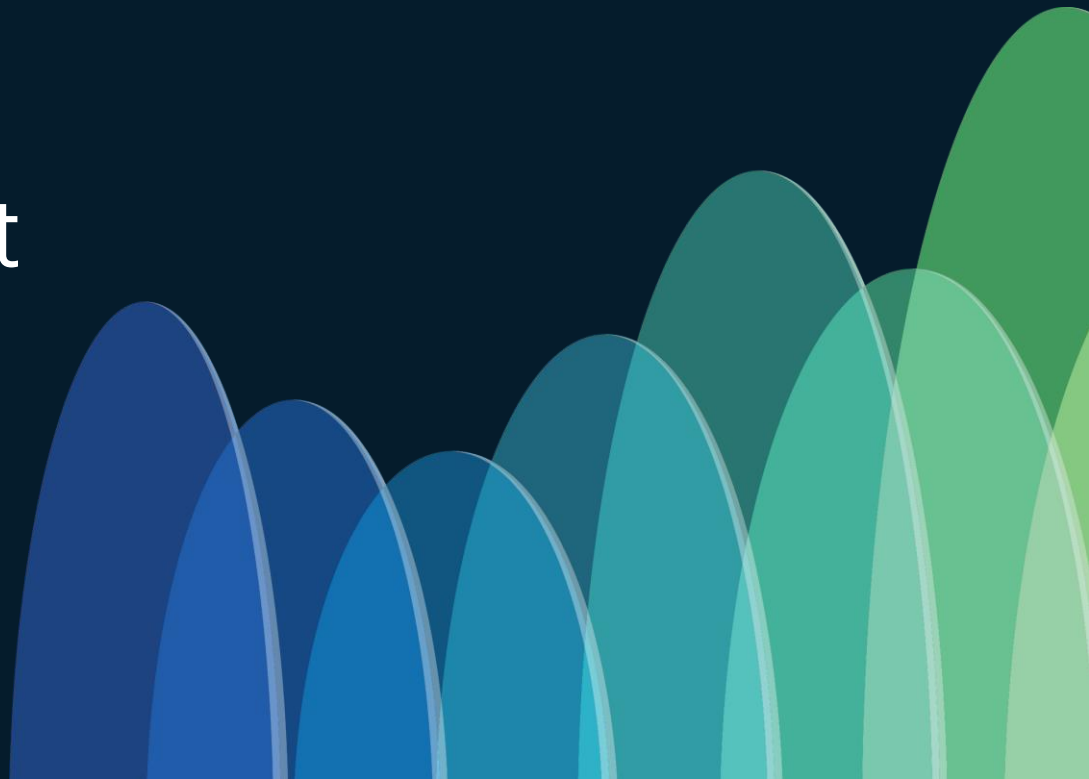
```
> terraform apply
```

Demo

Demo - New Infrastructure

- Create a new Tenant
  - Application Profiles, EPGs, VRF, Bridge Domains
- Fabric Access Policies
  - VLAN pool, Domain, AEP, interfaces
- L3Out – Connectivity to/from outside the fabric
  - L3Out, Node Profile, Static routing
  - External NX-OS device configured for connectivity back
- Code - `https://github.com/trenzy/BRKDCN-2607`

Terraform Import



Terraform import

- Terraform – Great IaC tool
- But what about objects that were not provisioned with Terraform?
- Terraform import CLI command
  - Import Infrastructure for Terraform to manage
  - Imports to Terraform state file
  - Does not generate the associated configuration

```
terraform import <resource_type.resource_name> <full path to Dn>
```

- Issues with this method
  - Only imports single object - Must be run for all child objects
  - Can get very tedious

Terraform Import Blocks

- Declarative method to Import Infrastructure to Terraform
  - Create import block configuration for all objects to import
  - Create the associated Terraform configurations for that object
    - `terraform plan -generate-config-out=new_tenant.tf`

```
# 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.tf)

Terraform Import

- terraform plan

```
> terraform plan

...
Plan: 2 to import, 0 to add, 0 to change, 0 to destroy.
```

- terraform apply

```
> terraform apply -auto-approve

...
Apply complete! Resources: 2 imported, 0 added, 0 changed, 0 destroyed.
```

- terraform state list

```
> terraform state list
aci_tenant.import_tenant_example
aci_vrf.tf_test_import_vrf
```

Key Takeaways

Infrastructure as Code for ACI - Terraform

Iac Benefits
Speed and Scale
Repeatability
Consistency

Install and Test Terraform
Runs on most platforms

Think Big – Start Small
Provision small/test environments
Move on to larger environments

Writing IaC is easy!
No Special Programming Skills needed
HCL

Robust APIC/NDO REST API
Makes automation easy and scalable

Additional Sessions/Labs

- LABDCN-1776: (Walk in Lab - Intro to Terraform with ACI)
- BRKDCN-2673: Nexus-as-Code - Kickstart your automation with ACI
- DEVWKS-1098: Infrastructure as Code on NX-OS using Terraform
- DEVWKS-1440: How to Terraform your Cisco Nexus HyperFabric
- DEVWKS-2931: Making Your ACI Automation as Modular as LEGO Bricks Using Terraform Modules
- DEVNET-3722: Advanced Terraform Techniques: Moving Beyond the Scaffolding
- IBODCN-1003: An Interactive Conversation on ACI Automation through Ansible and Terraform

More information - 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://developer.hashicorp.com/terraform/language/modules

Fill Out Your Session Surveys

Participants who fill out a minimum of 4 session surveys and the overall event survey will get a unique Cisco Live t-shirt.

(from 11:30 on Thursday, while supplies last)

All surveys can be taken in the Cisco Events mobile app or by logging in to the Session Catalog and clicking the ‘Participant Dashboard’

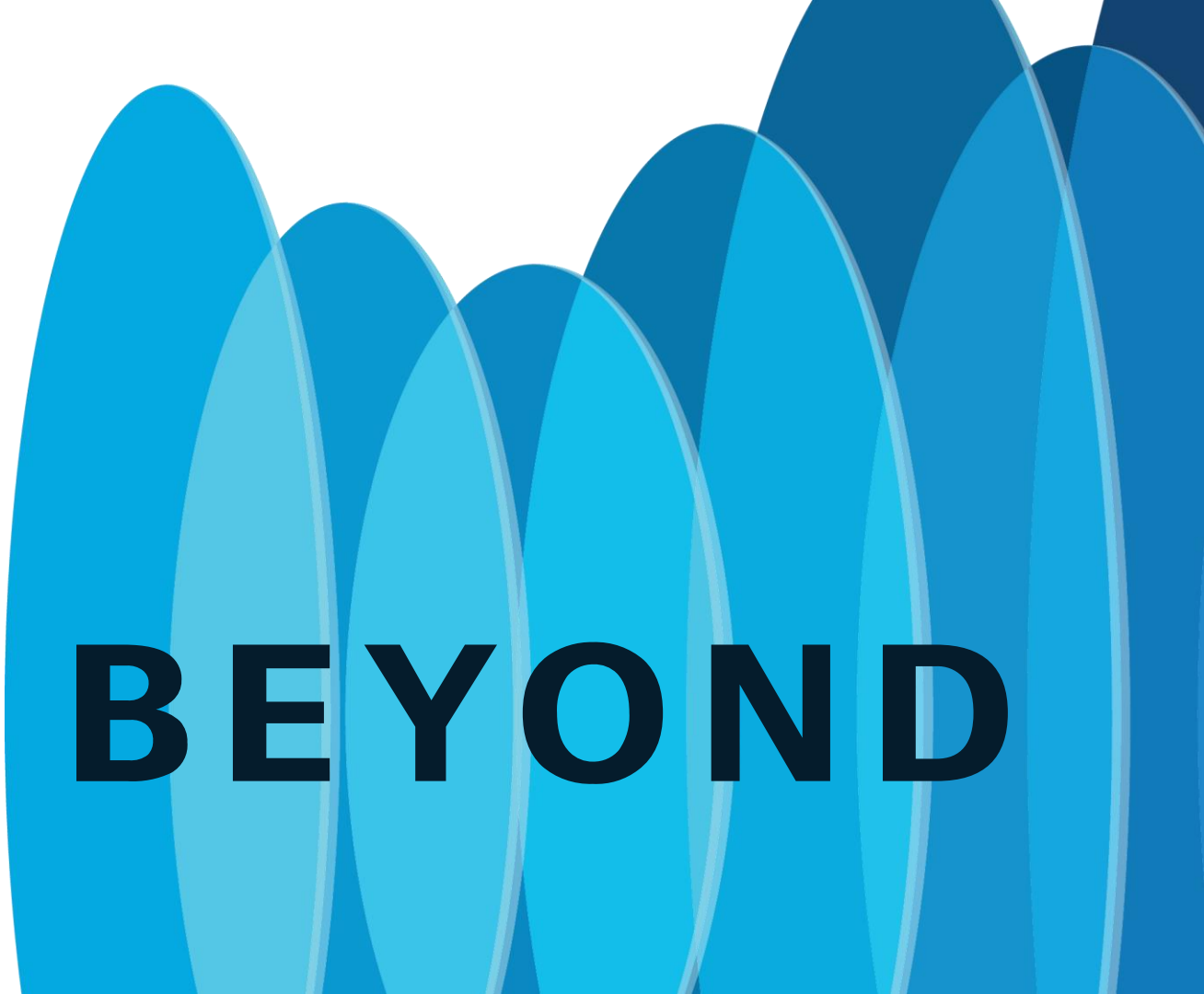
Content Catalog

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 ciscolive.com/on-demand. Sessions from this event will be available from March 3.


cisco
Thank you
cisco Live!

cisco Live!



GO BEYOND