
cisco Live!
TURN
IT
UP
#CiscoLive

The bridge to possible

Automating Cisco in the cloud with Terraform

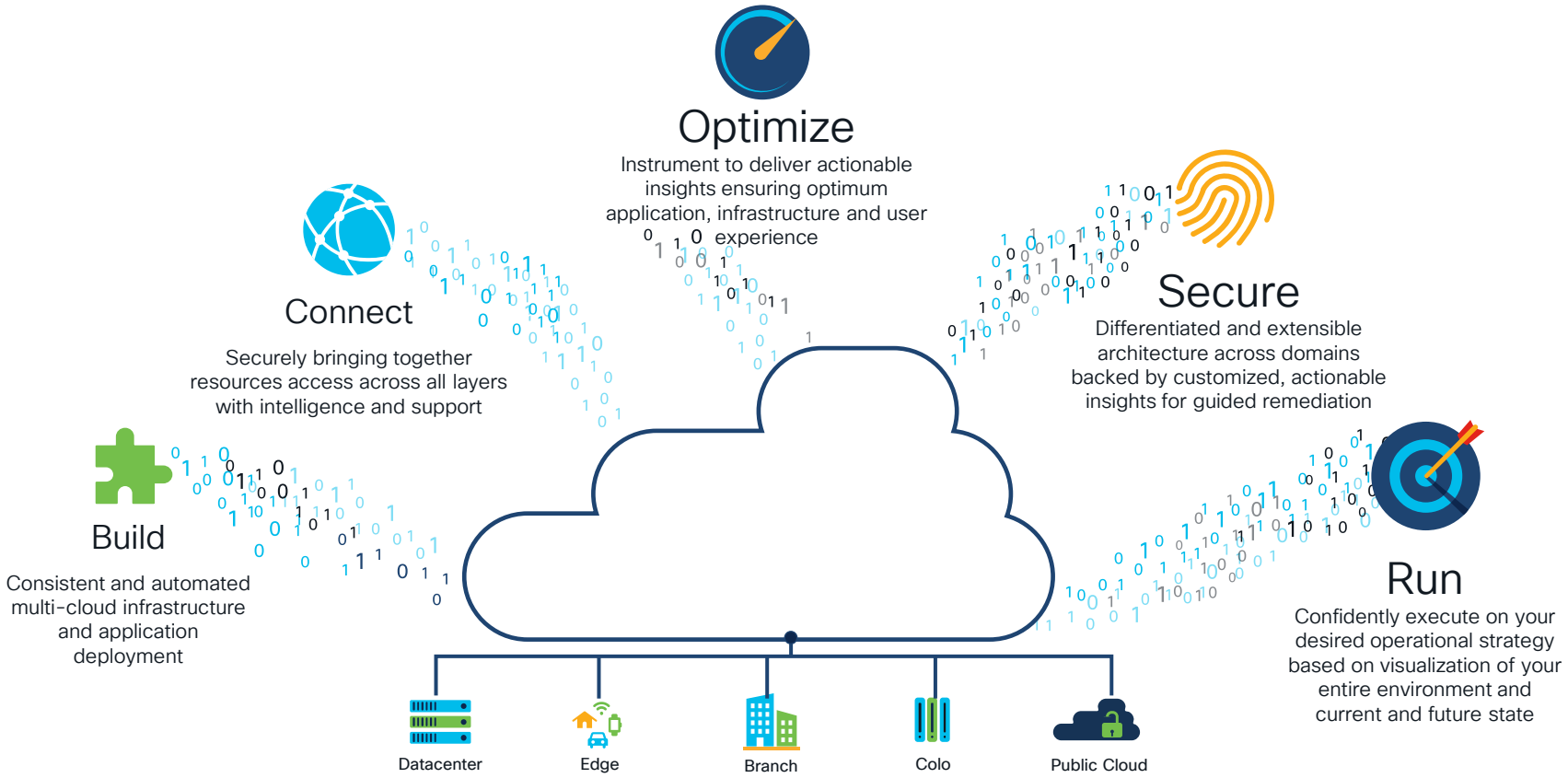
A practical look at Infrastructure as Code

Stuart Traynor, Systems Engineer
@sttrayno

BRKCLD-2037

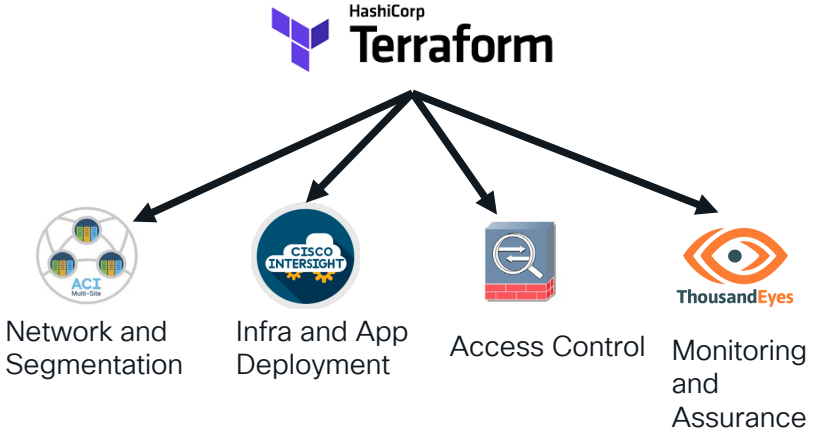
Agenda

- What is Terraform
- Providers
  - ACI
  - ASA
  - Intersight
  - Community Providers
- Use cases
- Demo



What is Terraform?

- Popular orchestration tool, built and maintained by Hashicorp
- Define a desired state and Terraform works to ensure that state is maintained
- Allows you to define infrastructure through repeatable templates
- Manages all resources through APIs, abstracted by a providers
- Terraform CLI / Terraform Cloud



Terraform config file

- Resources are declared through a series of text files in Terraforms own configuration language HCL
- Uses text files with a .tf extension
- Can be as simple as a single file, like so…
- You might have multiple files including variable files (.tfvars)

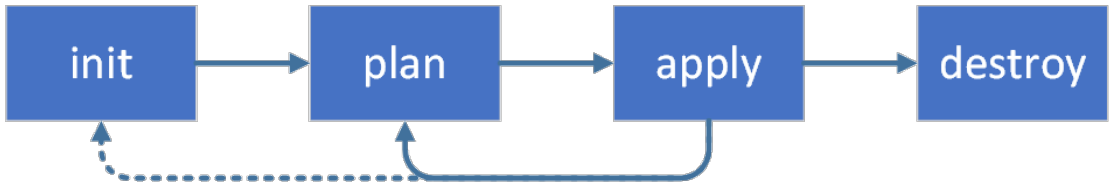
```
# Configure provider with your Cisco ACI credentials
provider "aci" {
  # Cisco ACI user name
  username = "admin"
  # Cisco ACI password
  password = "C1sco12345"
  # Cisco ACI URL
  url      = "https://198.18.133.200"
  insecure = true
}

# Variables
locals {
  vmm_vcenter        = "uni/vmmp-VMware/dom-My-vCenter"
  phys_db            = "uni/phys-phys"
}

# Tenant Definition
resource "aci_tenant" "terraform_tenant" {
  # Note the names cannot be modified in ACI, use the name_alias instead
  # The name becomes the distinguished named with the model, this is the ref
  # The model can be deployed A/B if the name, aka the model, must change
  name        = "terraform_tenant"
  name_alias  = "tenant_for_terraform"
  description = "This tenant is created by terraform ACI provider"
}

# Networking Definition
resource "aci_vrf" "default" {
  tenant_dn              = "${aci_tenant.terraform_tenant.id}"
  name                   = "default"
  name_alias             = "default"
}
```

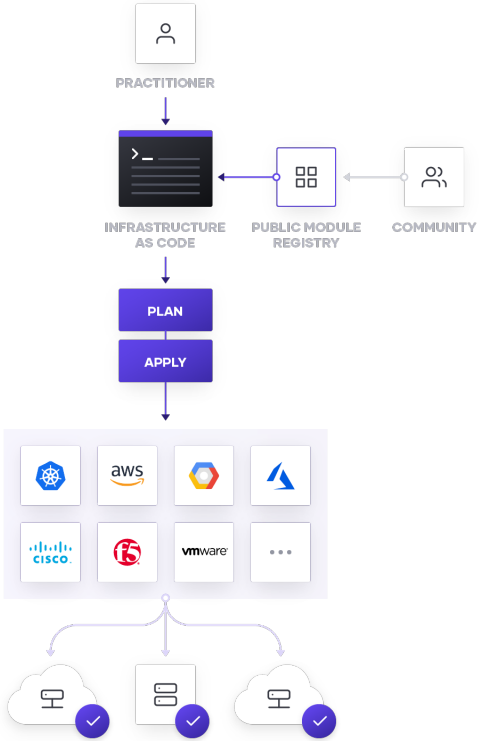
Terraform state

- Probably the most important concept behind Terraform
- Diff is made against the config and state to decide which resources are to be created
- State becomes an issue when it's large and must be shared…
- Resources not created by Terraform can be imported and managed

Terraform Workflow

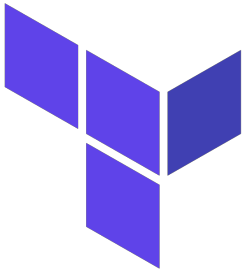
- Single workflow to plan, provision and teardown resources
- Providers allow you to abstract away from the individual processes and technology





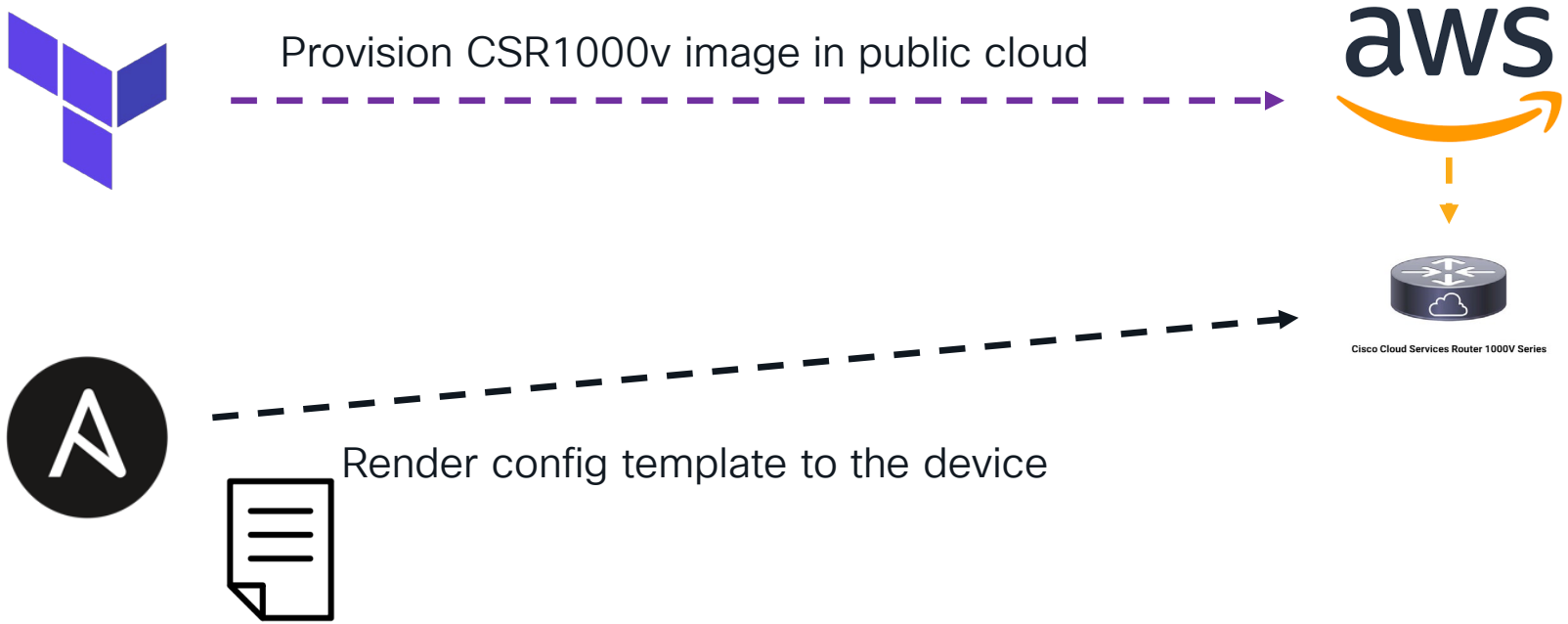
Ansible vs Terraform

- Very much focused towards infrastructure automation
- Modules (written in Python)
- Define playbooks in YAML
- Stateless *(by default)* – push out the intent of a playbook on run
- Imperative



- More focused to cloud automation
- Lends itself better to API first platforms
- Providers (written in Go)
- Define a Terraform config in HCL (HashiCorp configuration language)
- Stateful – keeps a state and looks to ensure the config matches the state
- Declarative

Ansible and Terraform – Simple example



Terraform Providers

- Cisco Application Centric Infrastructure (ACI) / Cisco Multi-site Orchestrator (MSO)
  - Define network policy and segmentation
  - Across on-premise and cloud networks
- Cisco ASA
  - More traditional perimeter-based access control
  - Traditional IP based filtering
  - Can be spun up in virtual form factor at the edge, public or private cloud
- Cisco Intersight
  - Provision on-premise servers and workloads
  - Across bare metal, virtual machines, containers
  - Manage on premise infrastructure from a single control point

Community Providers / Build your own

Terraform CLI / Terraform Cloud

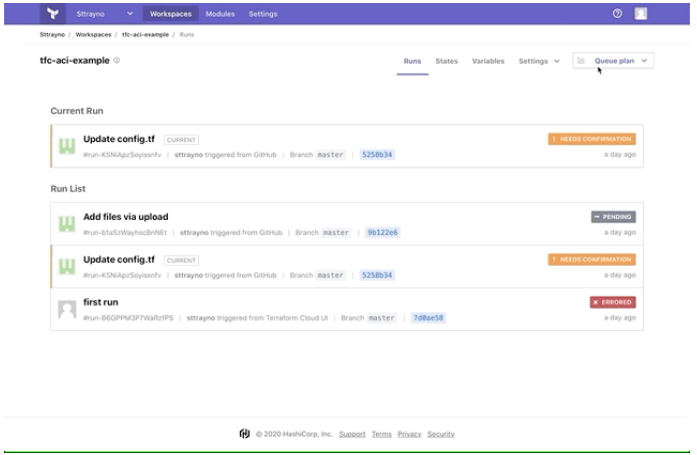
Terraform CLI

- Installed locally
- State and config stored locally

```
Last login: Thu Dec  5 23:59:02 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
STTRAYNO-M-L2L1:terraform sttrayno$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aci" (terraform-providers/aci) 0.1.2...

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, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aci: version = "~> 0.1"

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
```

Terraform Cloud

- SaaS delivered
- Able to store and share state
- Can be plugged into version control systems for config



Providers

Cloud Networking – ACI and MSO

- Abstracted method of networking leans itself well to Terraform
- Extend policy from data center network to the cloud from a single touchpoint (one policy multiple DC's and clouds)
- Simplifies network deployment by creating reusable architecture patterns
- Allows network to be deployed in repeatable patterns

Infrastructure teams define Terraform config for tenants, bridge domain, subnets and reusable architecture patterns

Application/DevOps team define Terraform config for applications, EPGs, contracts etc as required in self service manner

As changes are made to the Terraform configurations by the separate teams, they are applied to the infrastructure

```
resource "aci_application_profile" "terraform_app" {
  tenant_dn   = "${aci_tenant.terraform_tenant.id}"
  name        = "demo_ap"
  prio        = "level1"
}

resource "aci_application_epg" "application_epg1" {
    application_profile_dn  = "${aci_application_profile.terraform_app.id}"
    name                            = "db_epg"
    description                     = "%s"
    annotation                      = "tag_epg"
    exception_tag                   = "0"
    flood_on_encap                  = "disabled"
    fwd_ctrl                        = "none"
    has_mcast_source                = "no"
    is_attr_based_e_pg          = "no"
    match_t                         = "AtleastOne"
    name_alias                      = "alias_epg"
    pc_enf_pref                     = "unenforced"
    pref_gr_memb                    = "exclude"
    prio                            = "unspecified"
    shutdown                        = "no"
  }
resource "aci_application_epg" "application_epg2" {
    application_profile_dn  = "${aci_application_profile.terraform_app.id}"
    name                            = "web_epg"
    description                     = "%s"
    annotation                      = "tag_epg"
    exception_tag                   = "0"
    flood_on_encap                  = "disabled"
    fwd_ctrl                        = "none"
    has_mcast_source                = "no"
    is_attr_based_e_pg          = "no"
    match_t                         = "AtleastOne"
    name_alias                      = "alias_epg"
    pc_enf_pref                     = "unenforced"
    pref_gr_memb                    = "exclude"
    prio                            = "unspecified"
    shutdown                        = "no"
}
```

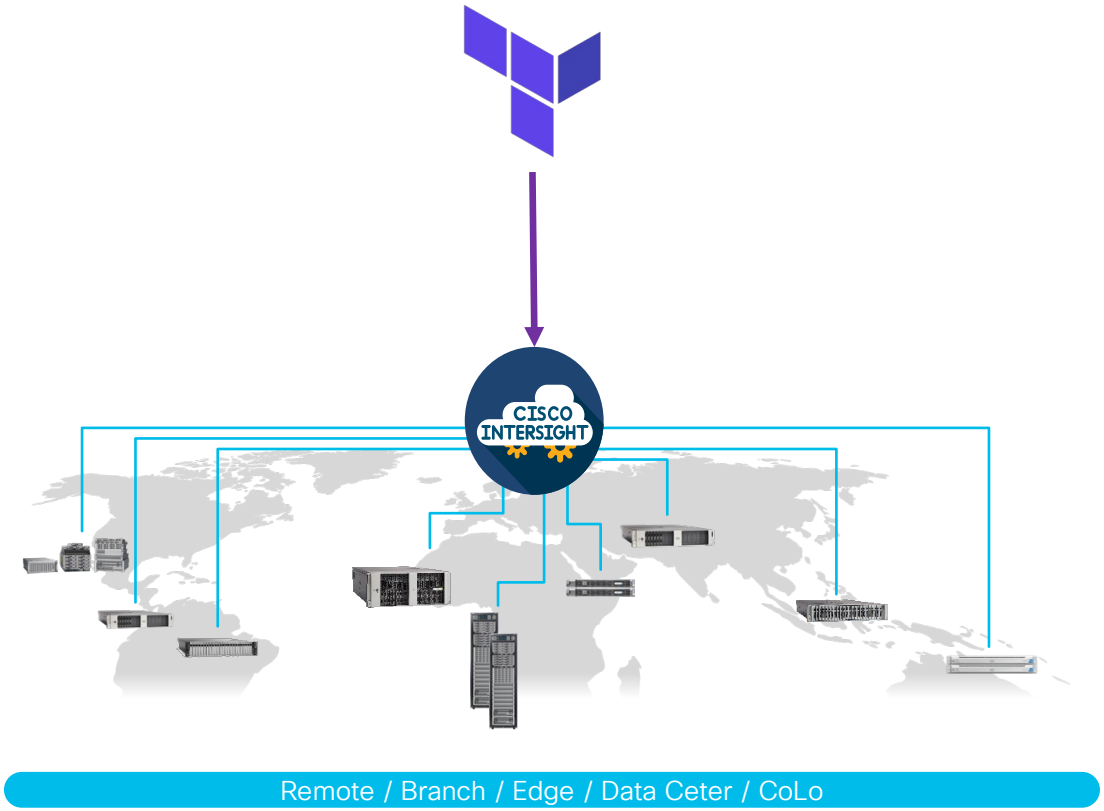
Access Control – ASA

- Limited functionality but potentially useful
- Configure resources for network objects, ACL's, Static Routes
- Consistent way to implement IP based filtering at any location (edge, DC, public cloud)
- Potentially part of a repeatable standard security architecture
- Expand, contract, or relocate workloads over time and span private and public cloud infrastructures with one license.

```
provider "ciscoasa" {
         api_url = https://10.0.0.5
         username = "admin"
         password = # your password here
        ssl_no_verify = false
}

resource "ciscoasa_acl" "foo" {
        name = "aclname"
        rule {
         source = "192.168.10.5/32"
         destination = "192.168.15.0/25"
         destination_service = "tcp/443"
        }
        rule {
          source = "192.168.10.0/24"
          destination = "192.168.15.6/32"
          destination_service = "udp/53"
        }
}
```

Intersight

- Single workflow for day 0/1 provisioning of servers: rackmount, blade or Hyperconverged
- Management of Edge/Branch/Data Center/Colo deployments
- Bare metal deployments
- Typical administration: IAM, network, syslog, ntp
- Ability to execute workflows through orchestration

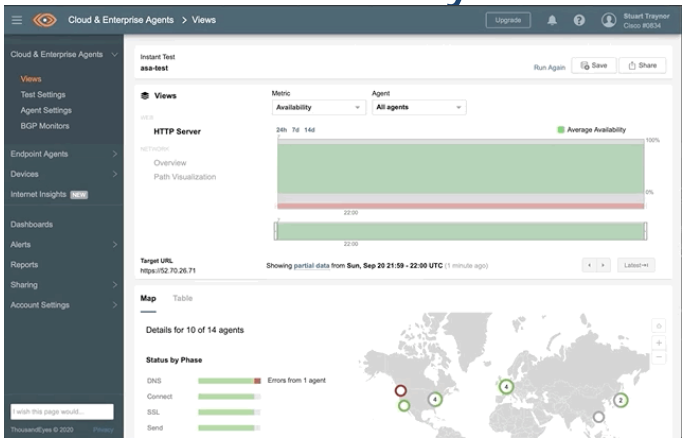


Community Providers

cisco Live!

Monitoring and Assurance - ThousandEyes

- Configure custom tests from Terraform configuration to gain realtime network intelligence
- Create tests dynamically on deployment across on-premises and cloud



ThousandEyes

https://github.com/william20111/terraform-provider-thousandeyes

Community Provider – Community Support

Monitoring and Assurance – AppDynamics

- Configure rules and events for application monitoring platform
- Deploy monitoring policy at the same time as application deployment

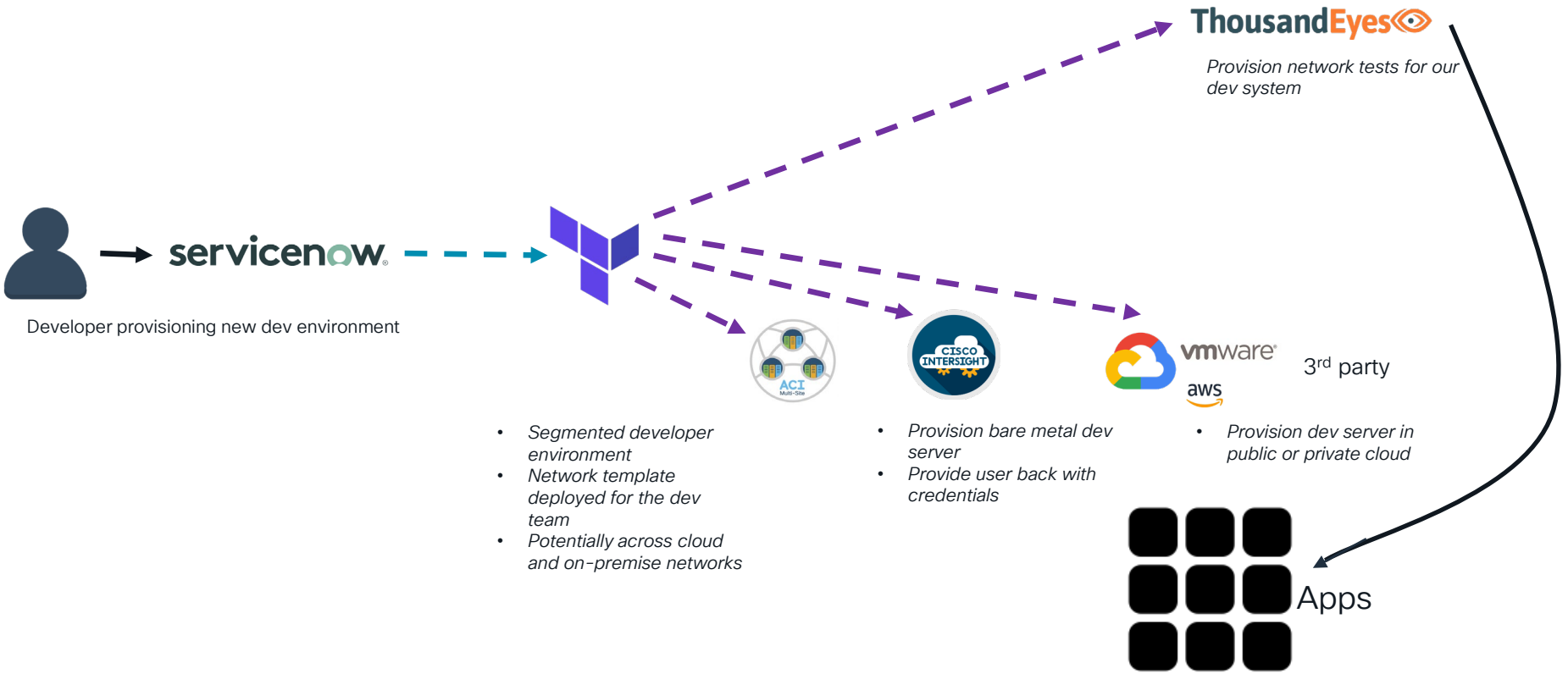
https://github.com/HarryEMartland/terraform-provider-appdynamics

APPDYNAMICS

Community Provider – Community Support

Demo

Use case – Self service provisioning



Summary

- Terraform is becoming an increasingly popular tool for infrastructure automation, particularly the line between cloud and on-premise starts to blur
- As people look to adopt Infrastructure as Code principles Terraform can be a good place to start
- Cisco has providers across the portfolio
- Get hands on, anyone can: http://github.com/sttrayno/Terraform-Lab-Guide
- DevNet sandboxes are available to support you in testing

Documentation and Resources

Installing Terraform - https://learn.hashicorp.com/tutorials/terraform/install-cli

Intersight user guide - https://github.com/cisco-intersight/terraform-provider-intersight/blob/master/USERGUIDE.md

Intersight examples - https://github.com/cisco-intersight/terraform-provider-intersight/tree/master/examples

Cisco ASA documentation - https://registry.terraform.io/providers/hashicorp/ciscoasa/latest/docs

Multi Site Orchestrator (MSO) documentation - https://registry.terraform.io/providers/CiscoDevNet/mso/latest/docs

Application Centric Infrastructure (ACI) documentation https://registry.terraform.io/providers/CiscoDevNet/aci/latest/docs

The bridge to possible

Thank you


cisco Live!
TURN
IT
UP
#CiscoLive