Elevator Pitch

Terraform is an infrastructure as code (IaC) open-source tool that lets you define and provision Cloud infrastructure in human-readable configuration files that you can version, reuse, and share. This session will introduce you to the IaC and Terraform concepts, and benefits.

Description

Terraform is an infrastructure as code (IaC) open-source tool that lets you define and provision Cloud infrastructure in human-readable configuration files that you can version, reuse, and share. This session will introduce you to the IaC and Terraform concepts, and benefits. You'll learn how to create configuration files, provision resources in the Cloud, and use Terraform Cloud, a SaaS service from HashiCorp to provision infrastructure in a remote environment optimized for the Terraform workflow.

PRAIRIE DEV CON
WEB | DEV | CLOUD | AI

Terraforming your infrastructure

(an intro to Terraform)

HashiCorp Terraform

Gold Sponsors

neo

improving
It's what we do. ™

SOLVERA
Part of Accenture

Microsoft

Community Supporters

MongoDB

Agenda

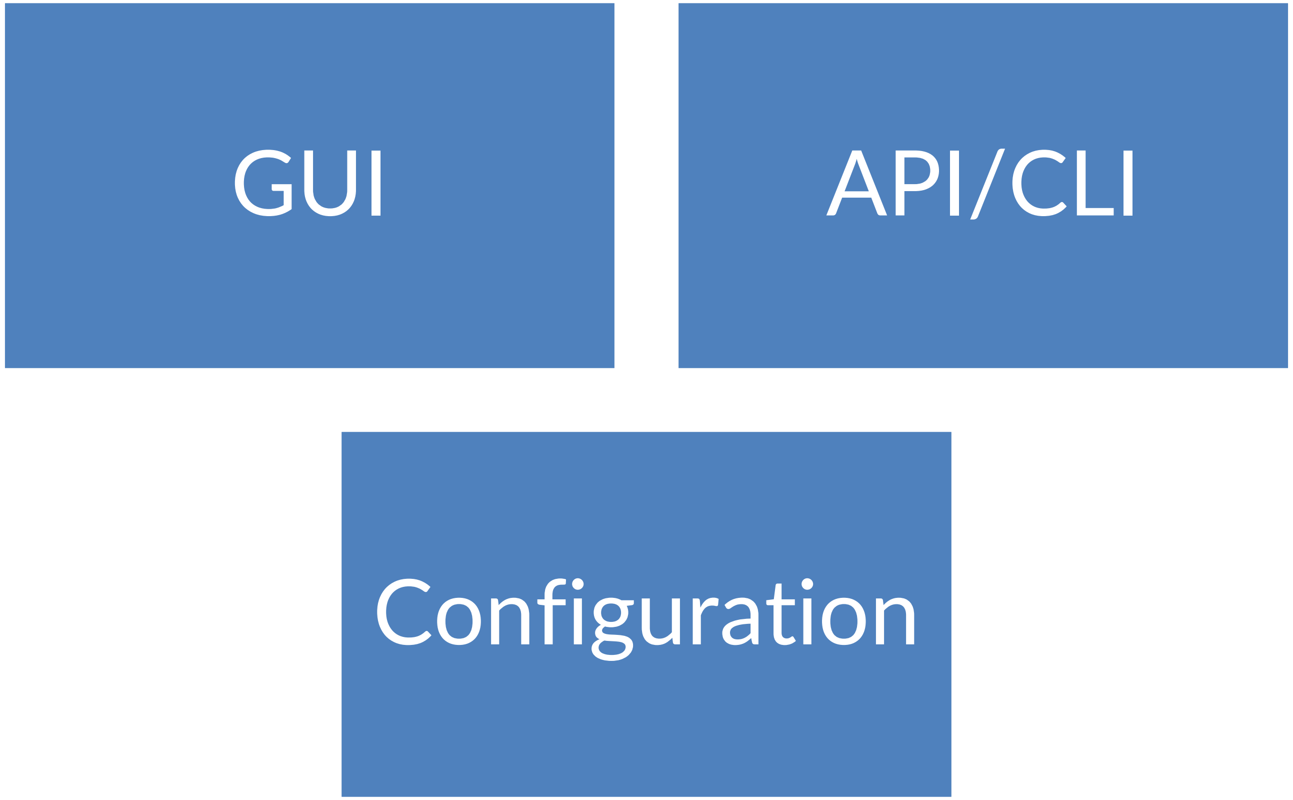
- Infrastructure as Code
- What is Terraform
- Terraform Lifecycle
- Terraform Cloud

Who am I?

- Guy Barrette
- Dev/Coach/Trainer
- Based in Montreal, Canada
- @GuyBarrette
- linkedin.com/in/guybarrette
- guybarrette.com
- @guybarrette@techhub.social

Infrastructure as Code

Provisioning Cloud Resources



GUI

API/CLI

Configuration

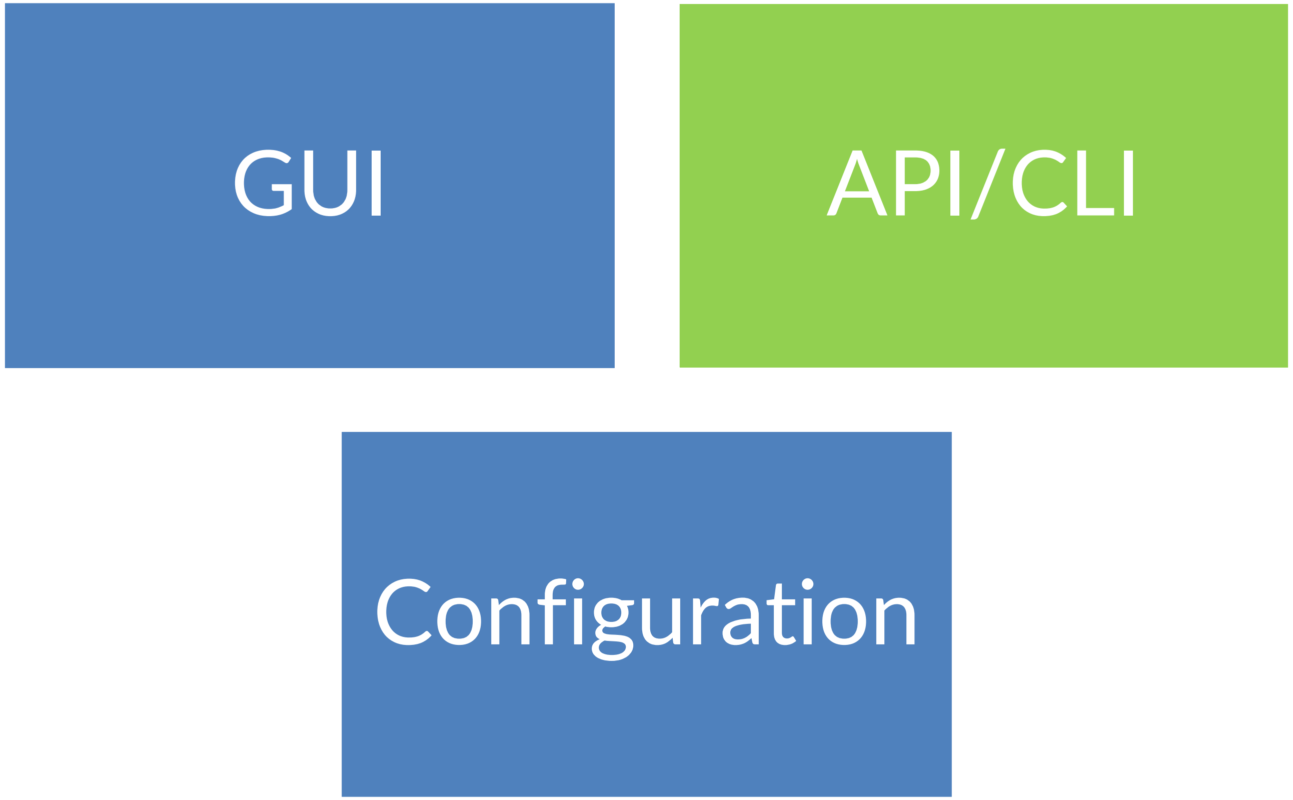
Provisioning Cloud Resources

GUI

API/CLI

Configuration

Provisioning Cloud Resources



GUI

API/CLI

Configuration

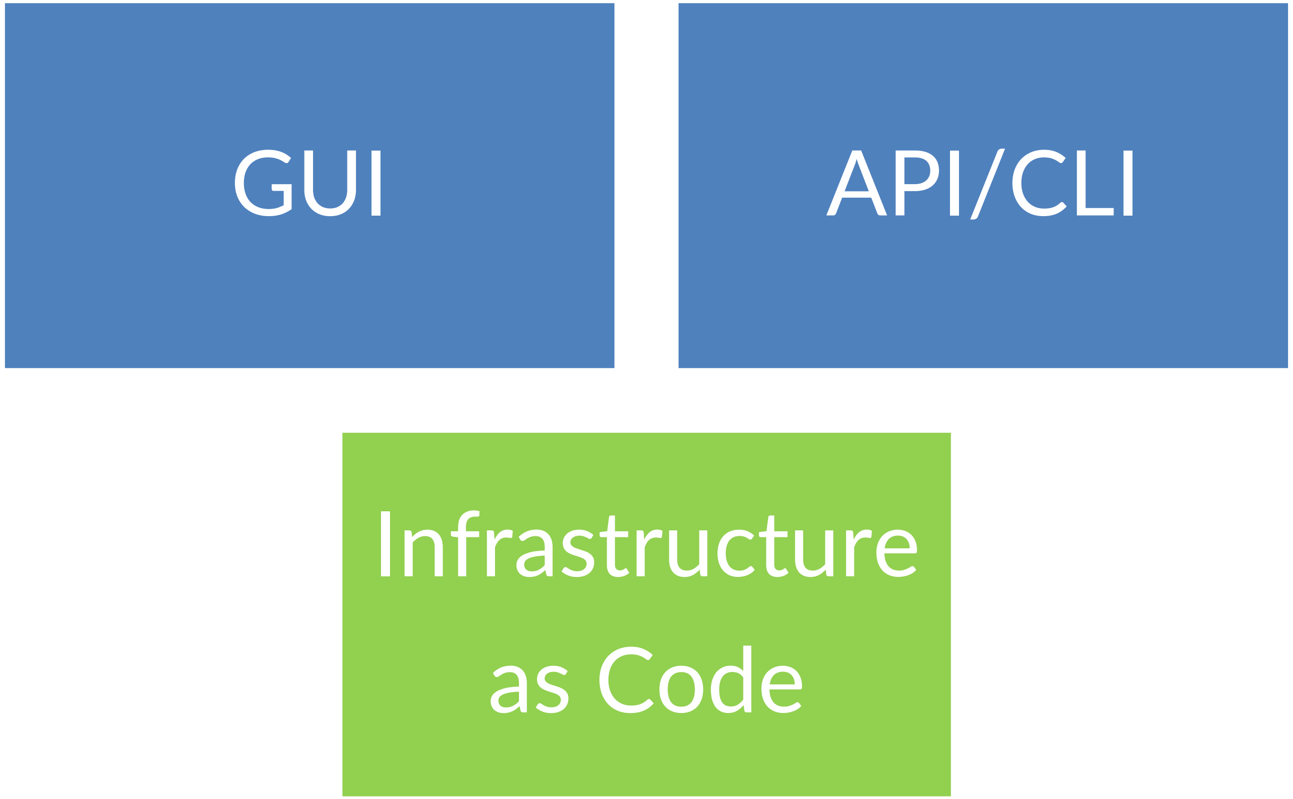
Provisioning Cloud Resources

GUI

API/CLI

Configuration

Provisioning Cloud Resources



GUI

API/CLI

Infrastructure as Code

What is Infrastructure as Code?

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

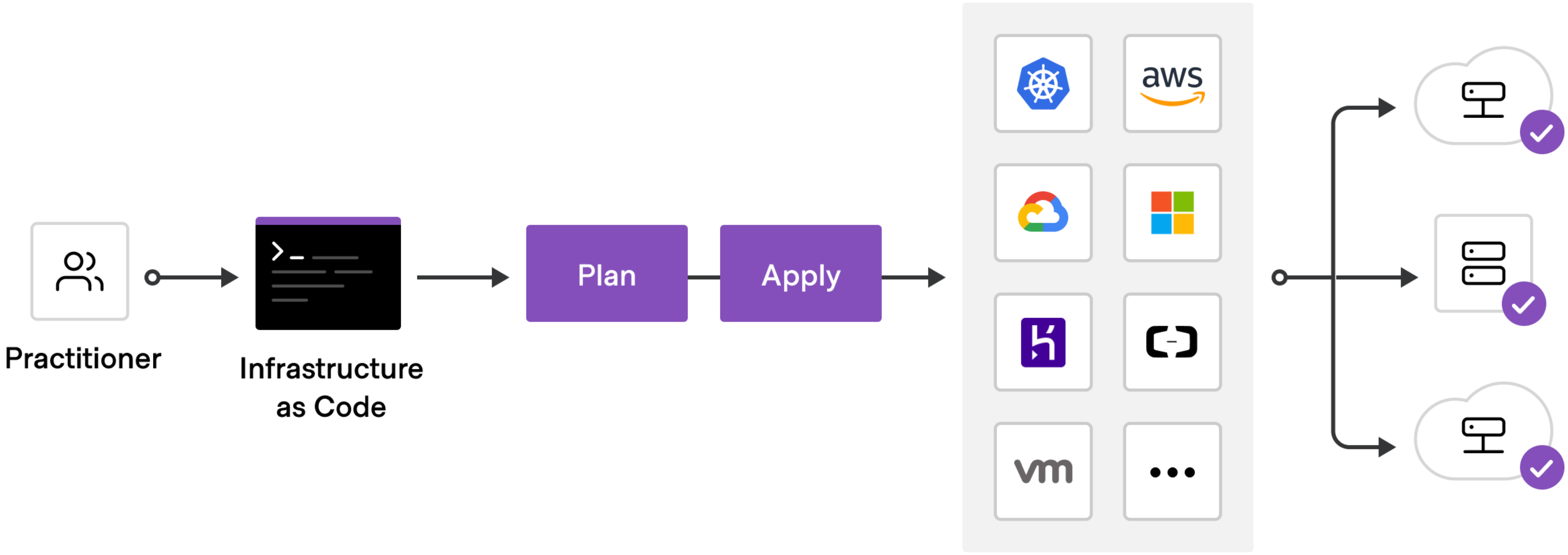
What is Infrastructure as Code?

Infrastructure as code (IaC) is the process of **managing and provisioning** computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

What is Infrastructure as Code?

Infrastructure as code (IaC) is the process of **managing and provisioning** computer data centers through **machine-readable definition files**, rather than physical hardware configuration or interactive configuration tools.

Infrastructure as Code



https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code?in=terraform/azure-get-started

Imperative vs Declarative

```
# create the resource group
az group create --name myResourceGroup --location eastus

# create the cluster
az aks create -g myResourceGroup --name myAKSCluster
  --node-count 1
  --enable-addons monitoring
  --generate-ssh-keys
```

Execute a sequence of steps

Desired state

```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "~>2.0"
    }
  }
}

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "k8s" {
    name     = var.resource_group_name
    location = var.location
}

resource "azurerm_kubernetes_cluster" "k8s" {
    name                = var.cluster_name
    location            = azurerm_resource_group.k8s.location
    resource_group_name = azurerm_resource_group.k8s.name
    dns_prefix          = var.dns_prefix

    linux_profile {
        admin_username = "ubuntu"
…
```

Infrastructure as Code

- Configuration is written in files
- Allows configurations to be version-controlled
- Promotes auditing
- Prevents environment drifts
- Create reproducible environments, faster
- Consistent, Repeatable, Predictable

Infrastructure as Code - Tools

Cloud Specific

- AWS CloudFormation
- Azure Resource Manager
- Azure Bicep
- Google Cloud Deployment Manager

Cloud Agnostic

- Terraform
- Pulumi

The blue pill or the red pill?

Terraform

What is Terraform?

- Infrastructure as code tool maintained by HashiCorp
  - Provision infrastructure
  - Open source
  - www.terraform.io
- Cloud agnostic
  - GCP, AWS, Azure, DO, Linode, etc
- Uses the HashiCorp Configuration Language (HCL)
- Uses declarative configuration files
- Manages the infrastructure state

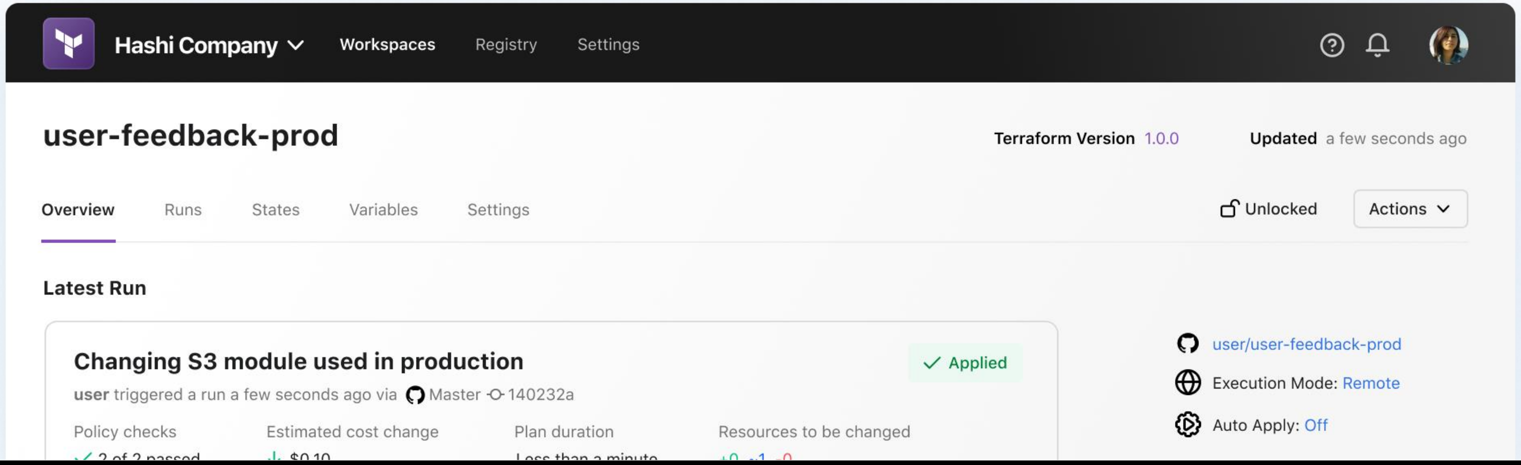
What is Terraform?

- Split into two main parts
  - Core
  - Plugins (providers & provisionners)
    - Installed when running Init
    - Providers
      - Cloud service (AWS, Azure, GCP, etc)
      - SaaS (GitHub, etc)
    - Provisionners
      - Used for executing scripts or shell commands on a local or remote machine as part of resource creation/deletion
      - Should only be used as a last resort

HashiCorp Terraform

What is Terraform?

- Terraform Cloud
  - Shared state and secret data
  - Access controls for approving changes to infrastructure
  - Private registry for storing private modules
  - Free and paid tiers



user-feedback-prod

Terraform Version 1.0.0

Updated a few seconds ago

Latest Run

Changing S3 module used in production

✓ Applied

user triggered a run a few seconds ago via Master 140232a

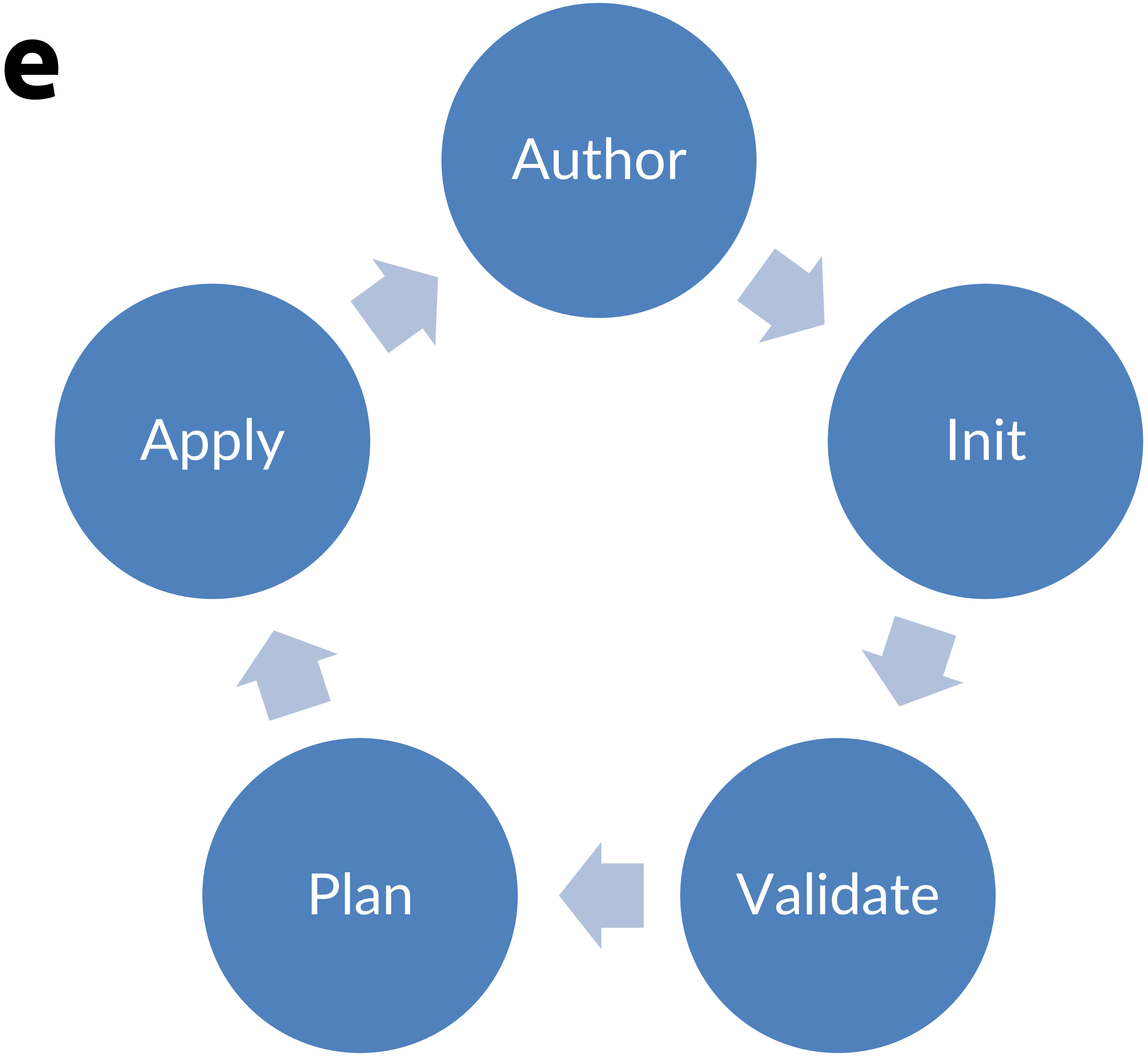
Policy checks	Estimated cost change	Plan duration	Resources to be changed
✓ 2 of 2 passed	$0.10	Less than a minute	+0 ~1 -0

user/user-feedback-prod

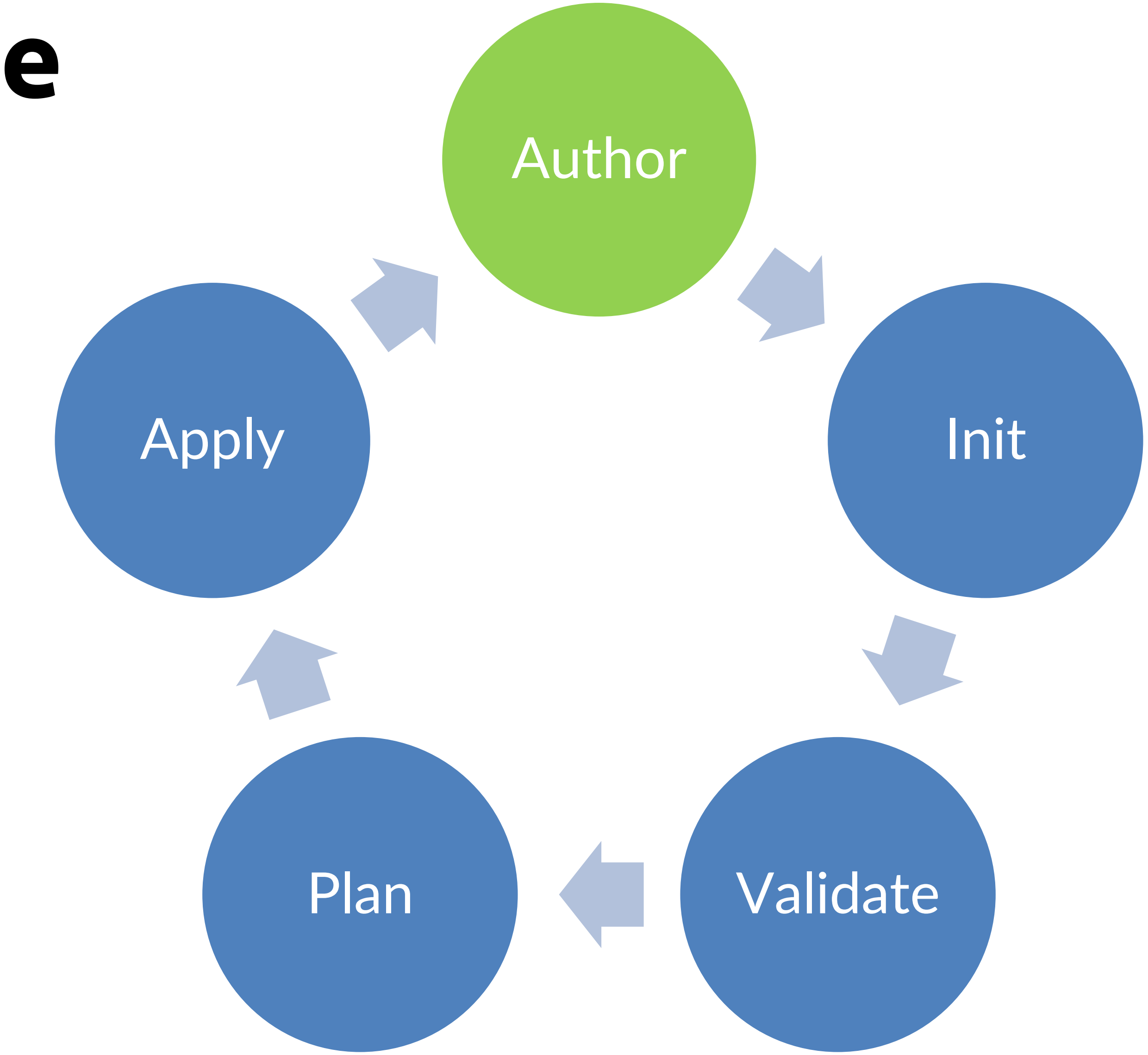
Execution Mode: Remote

Auto Apply: Off

Lifecycle



Lifecycle



Author

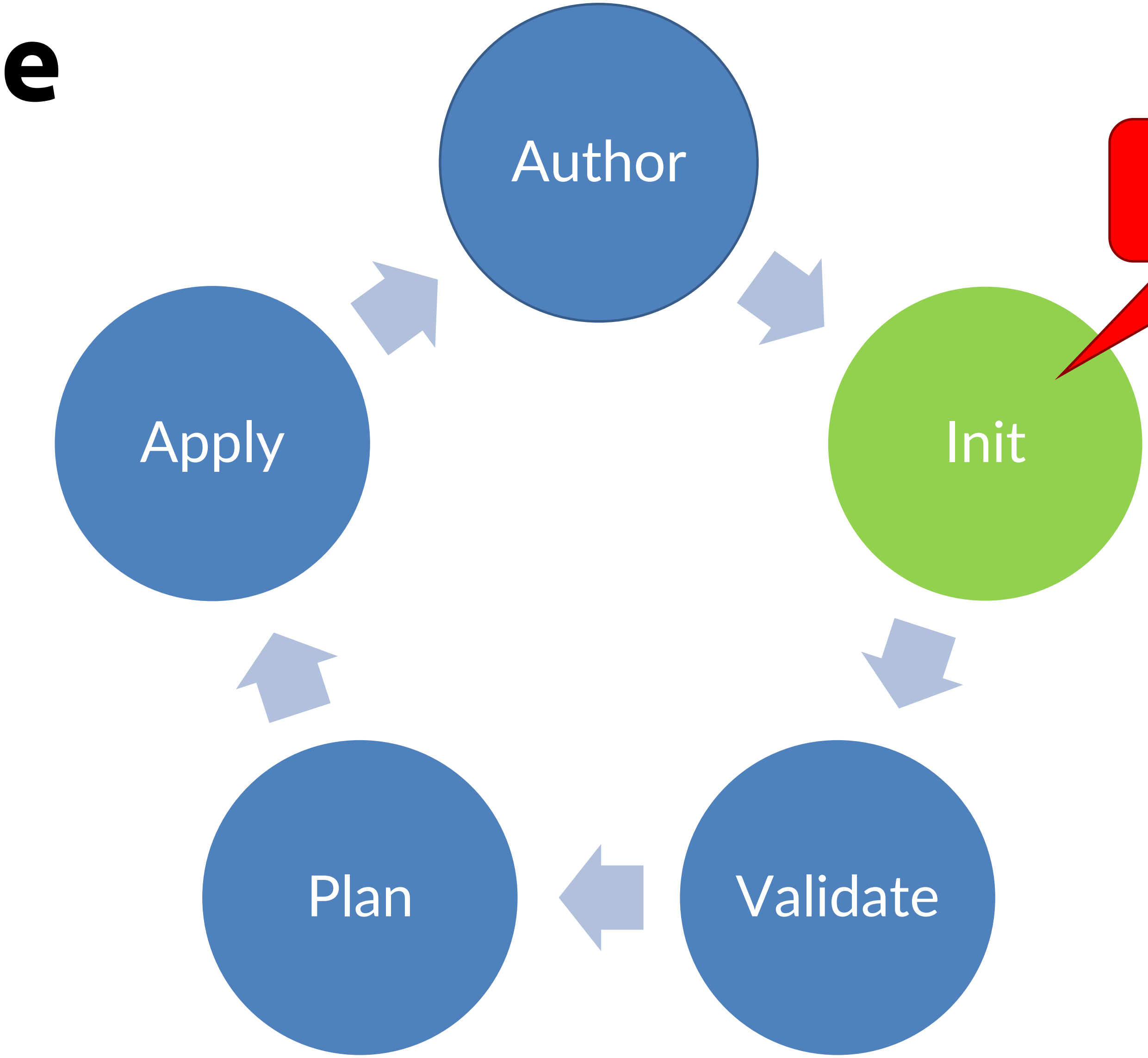
Init

Validate

Plan

Apply

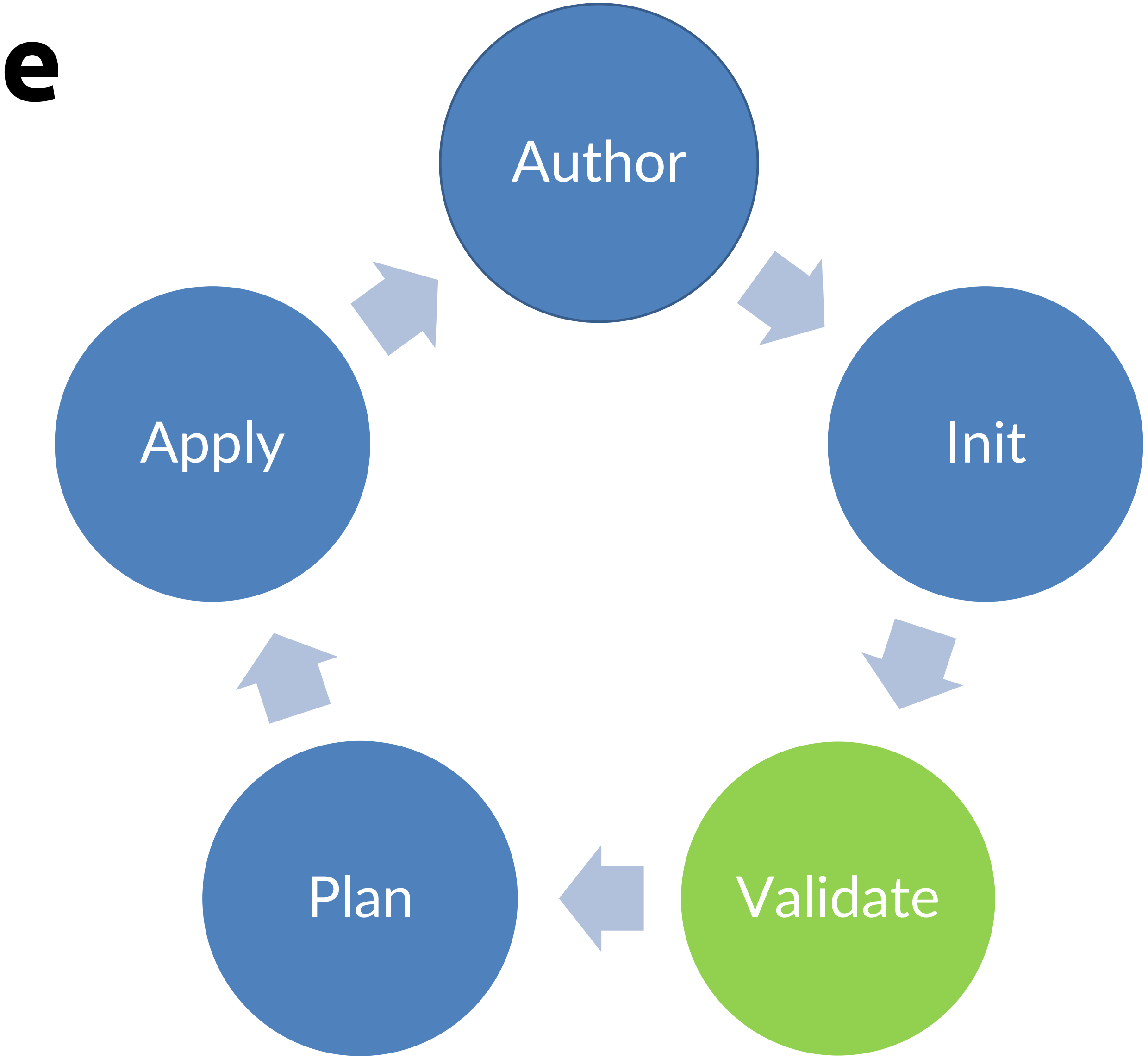
Lifecycle



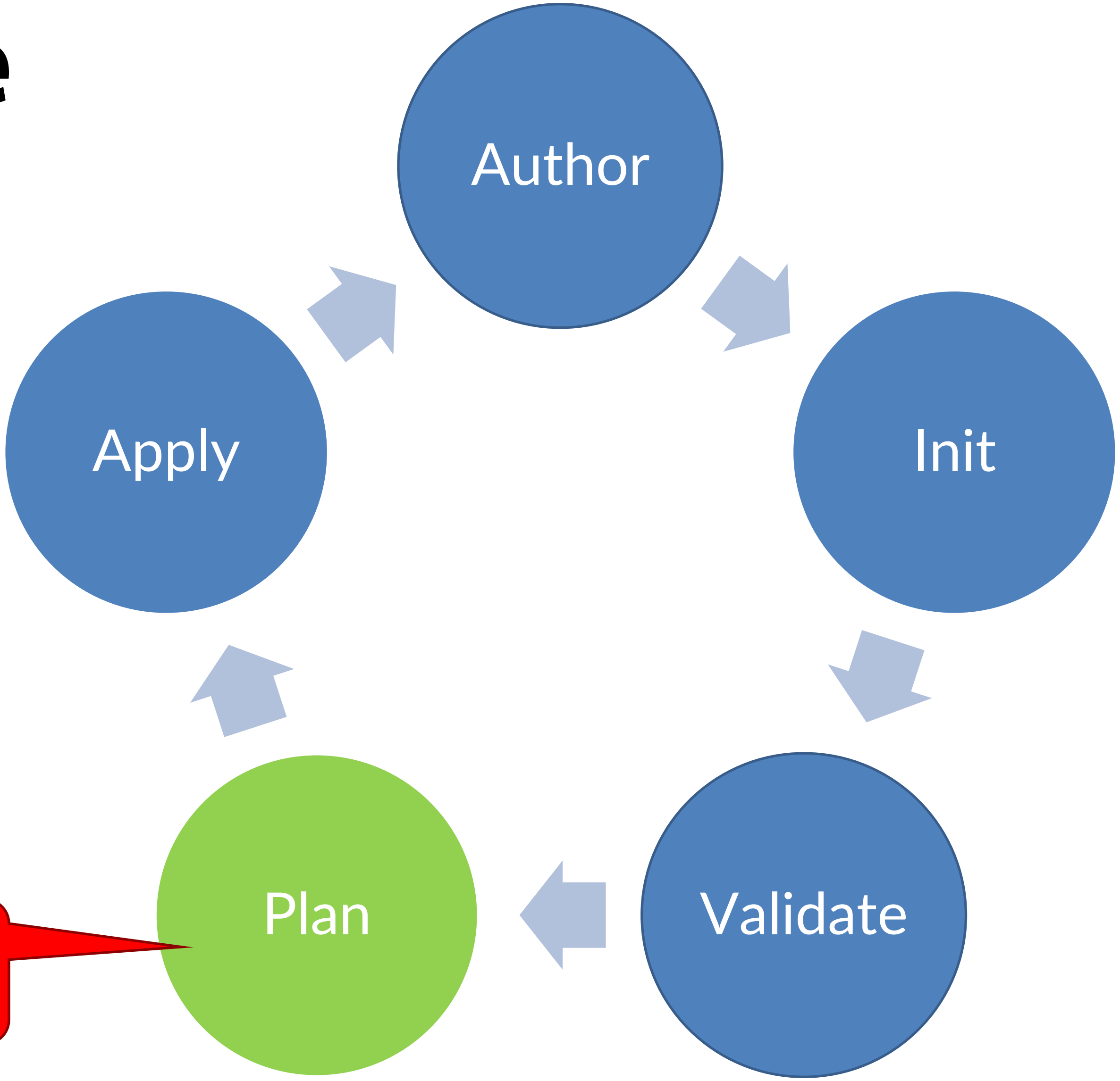
Download the provider

Terraform Registry

Lifecycle

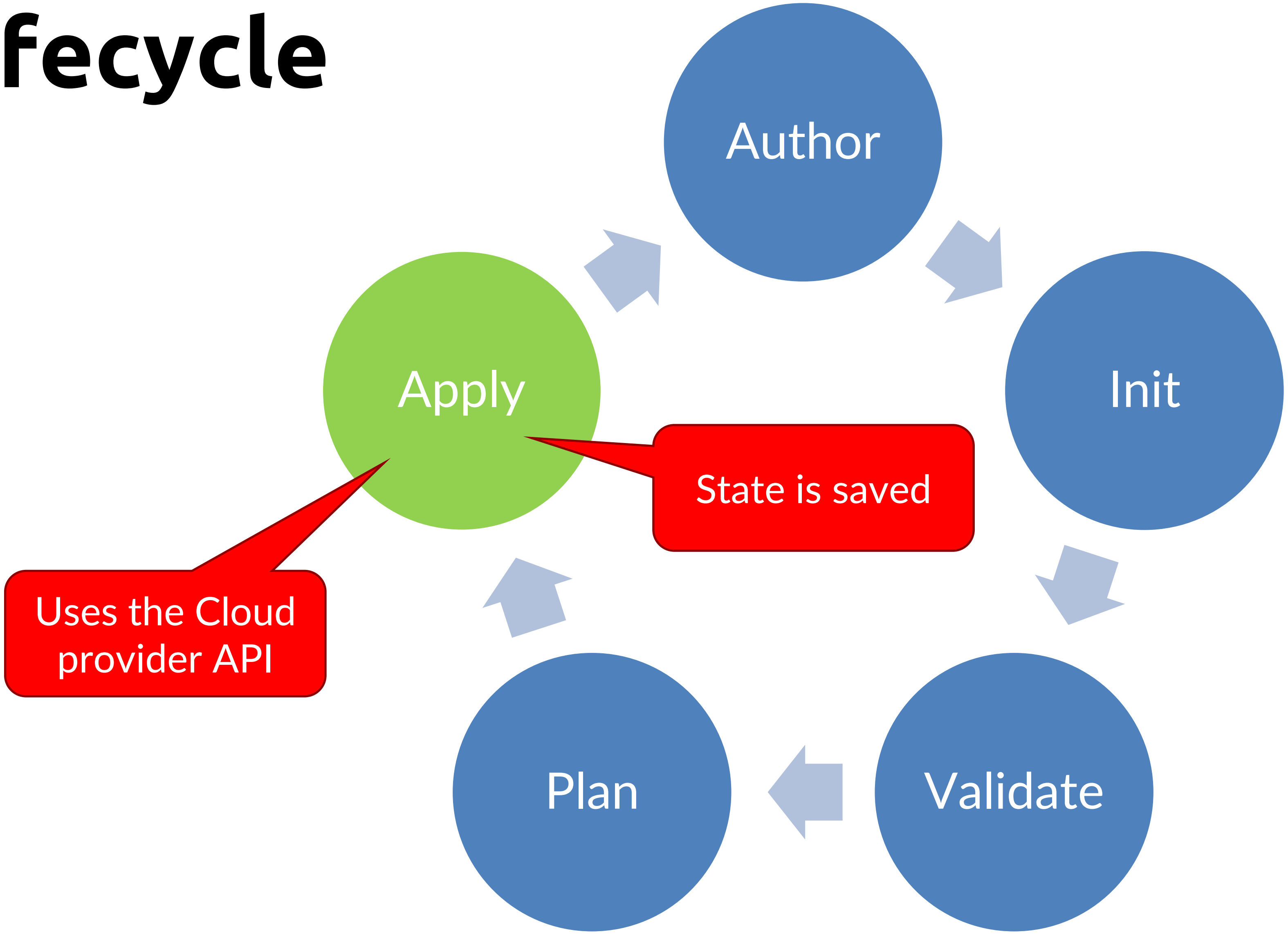


Lifecycle



Generates the execution plan

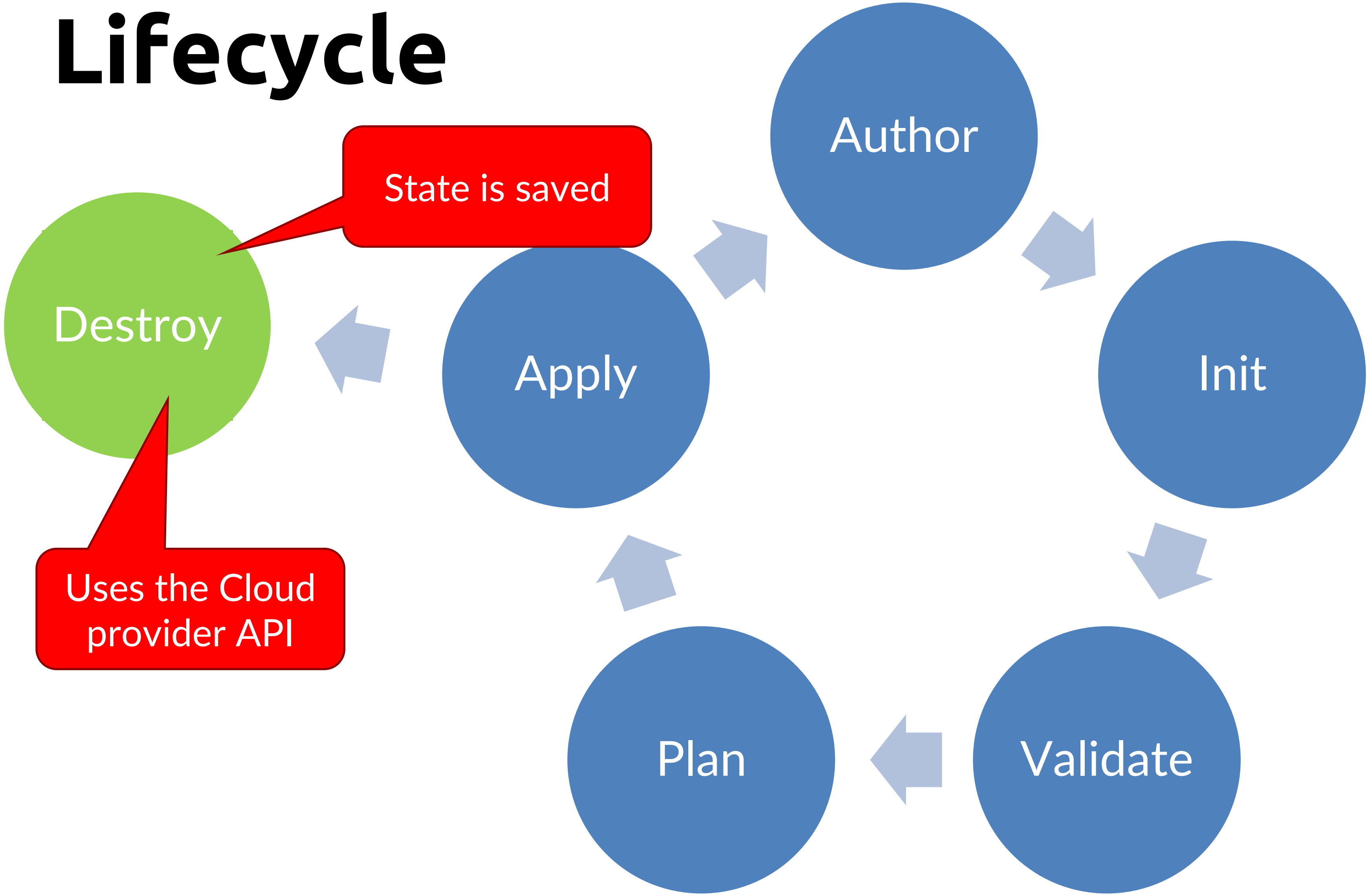
Lifecycle



State is saved

Uses the Cloud provider API

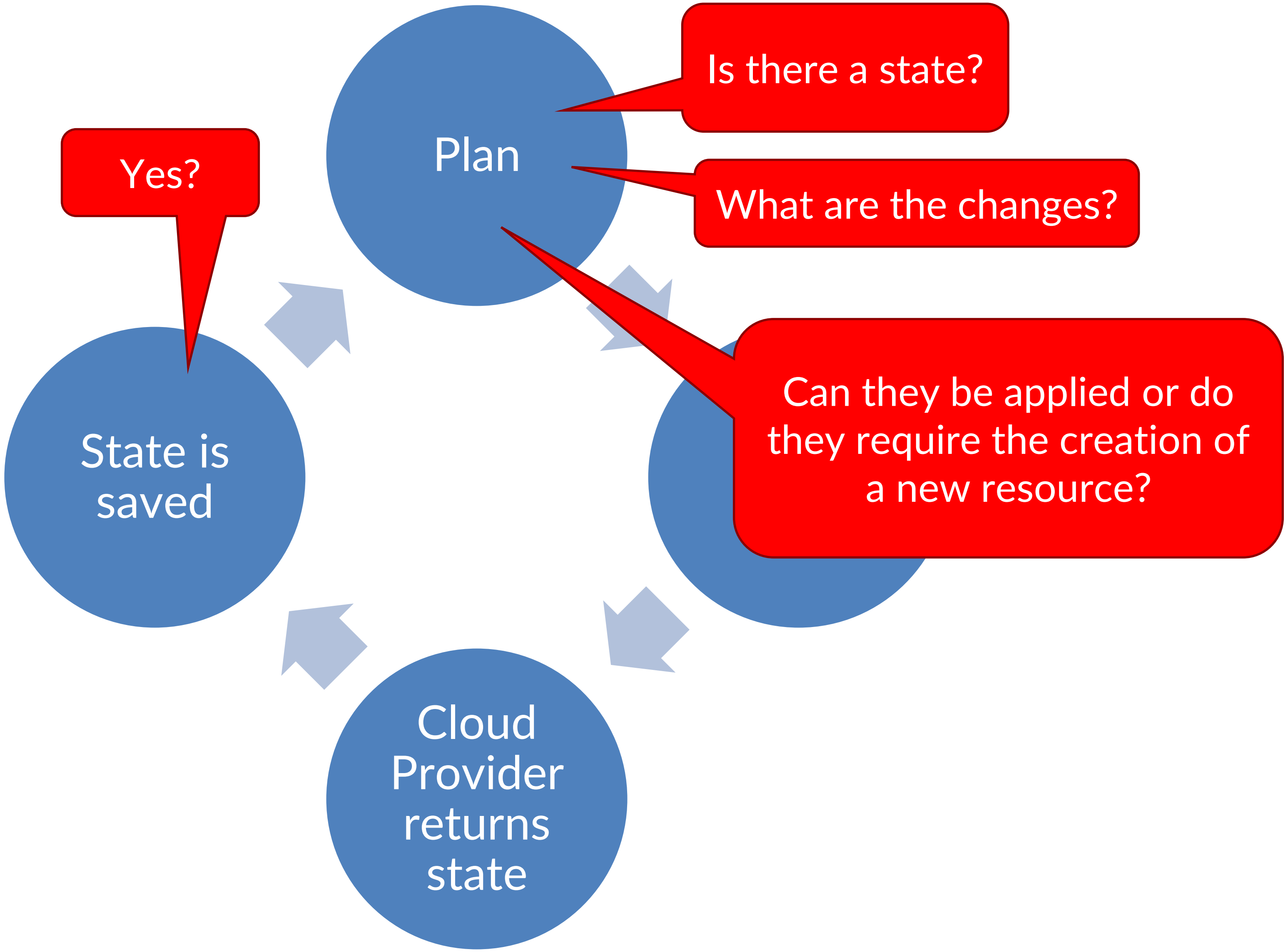
Lifecycle



State is saved

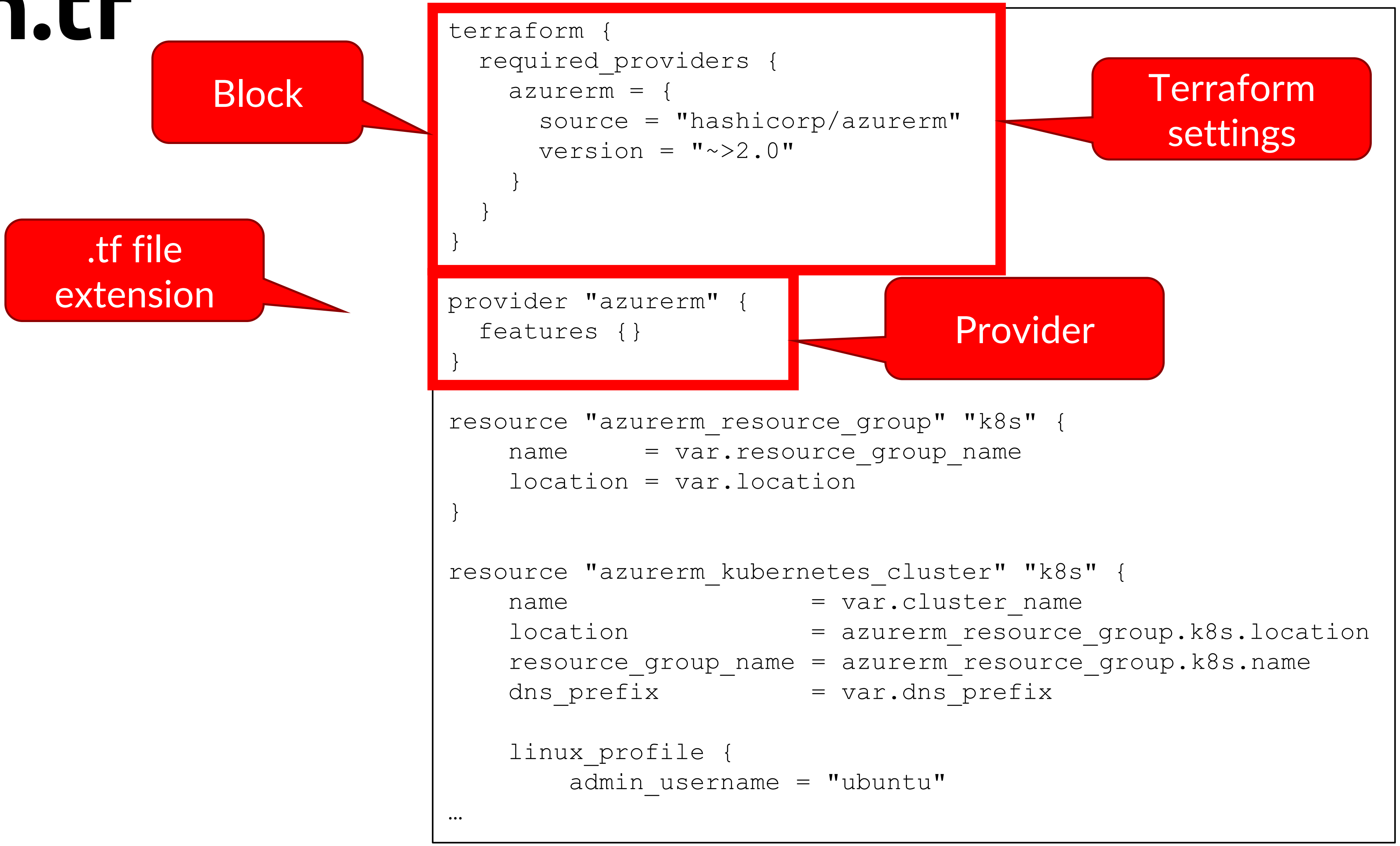
Uses the Cloud provider API

Lifecycle



Plan
Cloud Provider returns state
State is saved
Is there a state?
What are the changes?
Can they be applied or do they require the creation of a new resource?
Yes?

Main.tf



```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "~>2.0"
    }
  }
}

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "k8s" {
    name     = var.resource_group_name
    location = var.location
}

resource "azurerm_kubernetes_cluster" "k8s" {
    name                = var.cluster_name
    location            = azurerm_resource_group.k8s.location
    resource_group_name = azurerm_resource_group.k8s.name
    dns_prefix          = var.dns_prefix

    linux_profile {
        admin_username = "ubuntu"
…
```

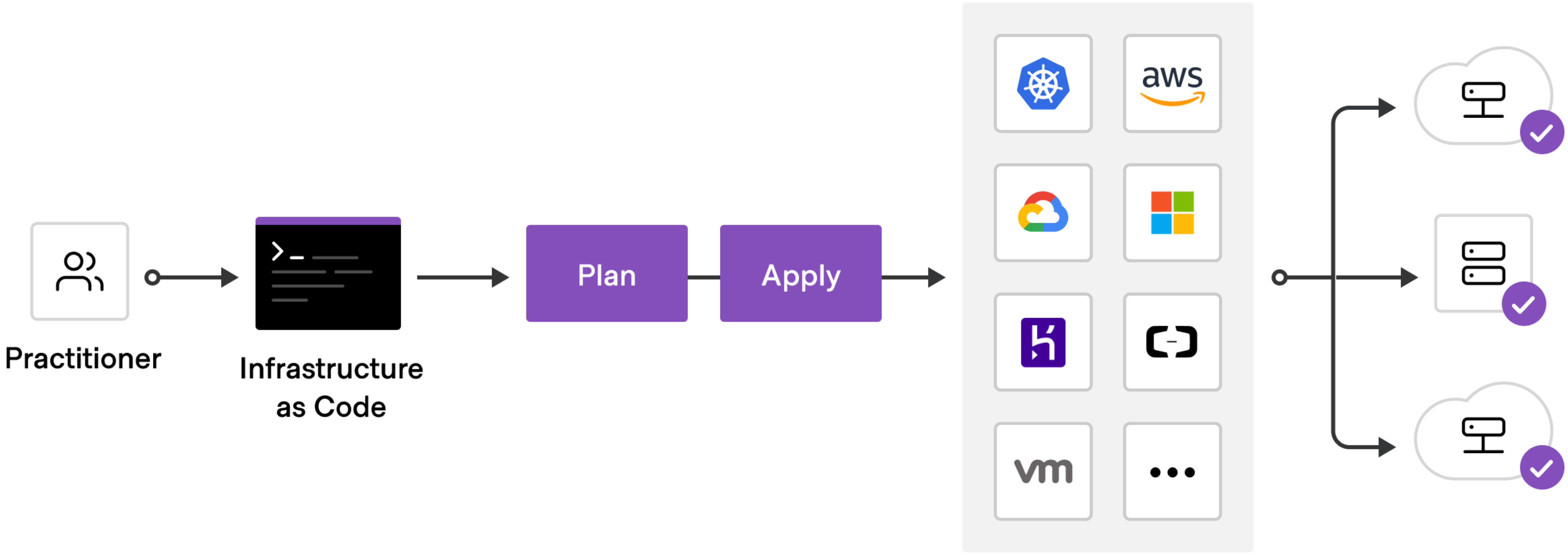
Demo

Terraform Cloud

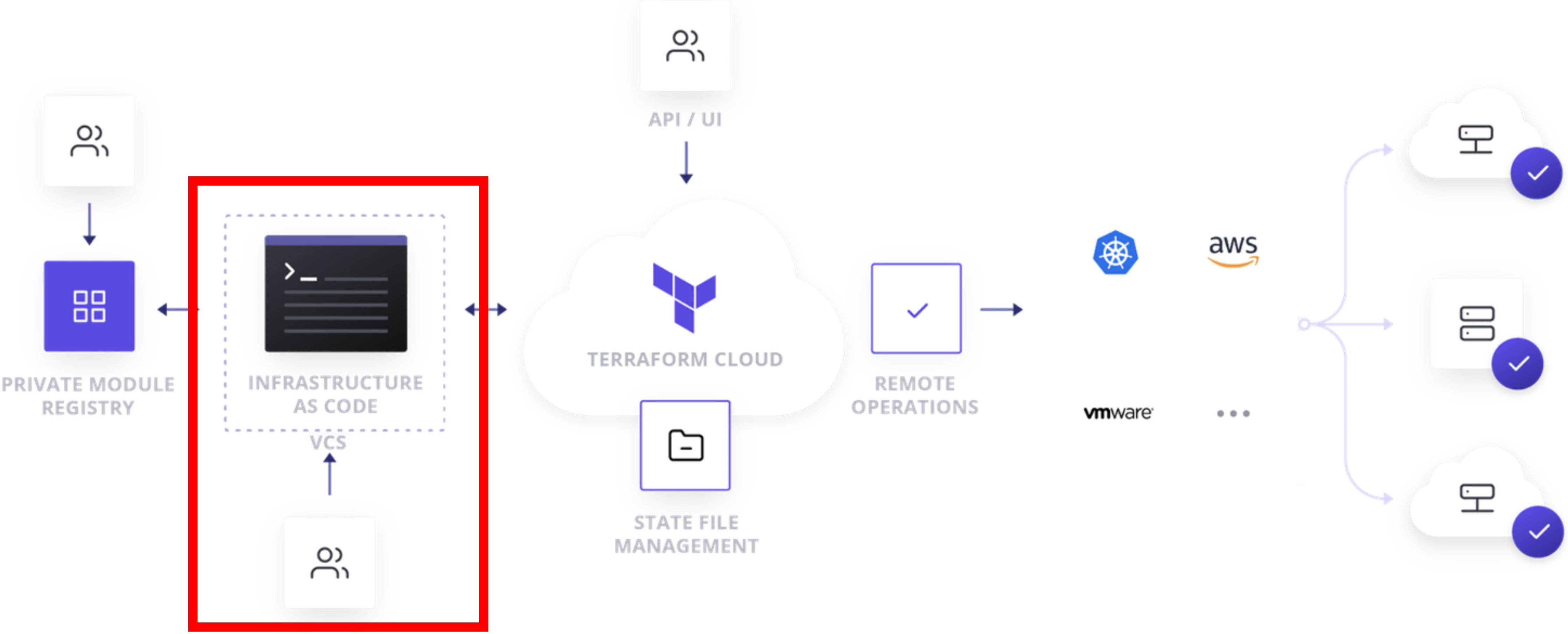
Terraform Cloud

- Managed service
  - app.terraform.io
  - Run infrastructure as code remotely
    - Think GitHub Actions
  - Free and paid tiers
  - Shared state and secret data
  - Access controls for approving changes to infrastructure
  - Private registry for storing private modules
- Terraform Enterprise
  - Self-Hosted

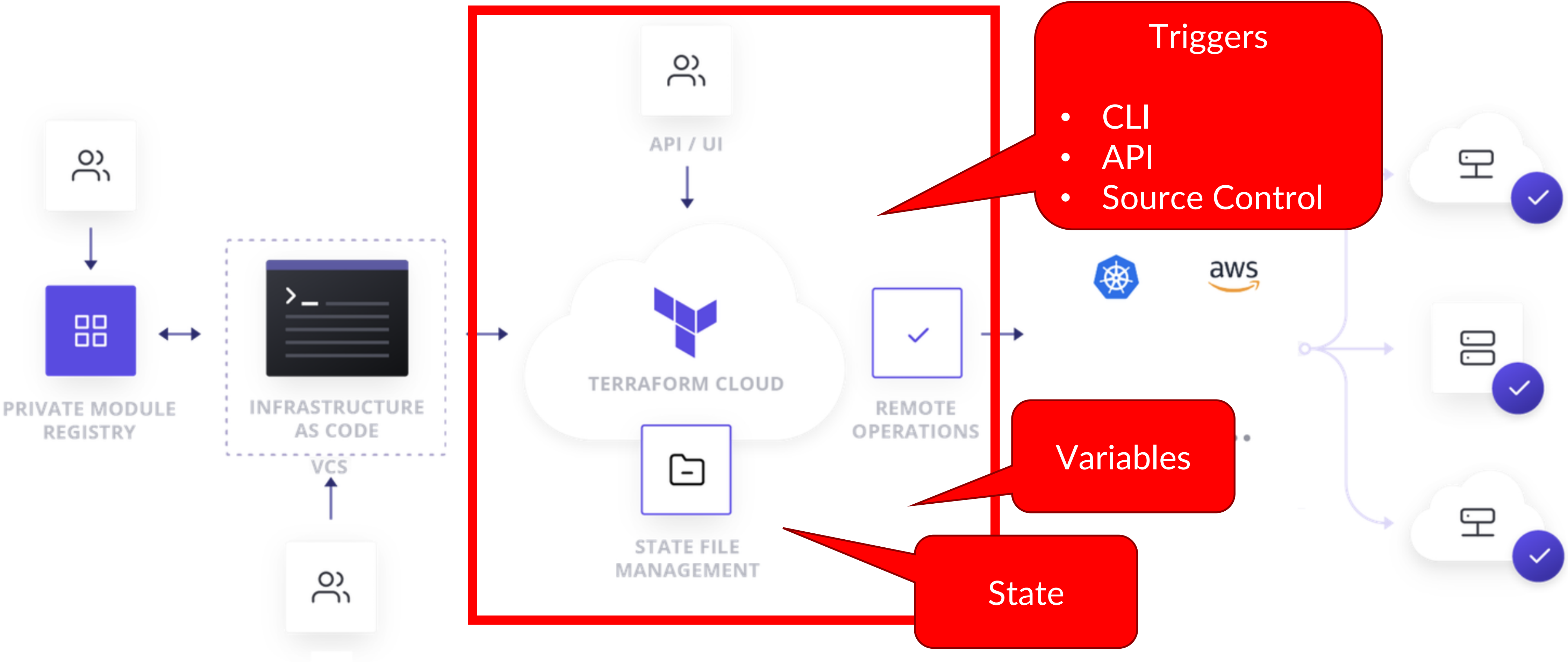
Terraform so far



Terraform Cloud



Terraform Cloud



Triggers

- CLI
- API
- Source Control

Variables

State

Terraform Cloud Workspaces

- Contains everything Terraform needs to manage a given collection of infrastructure
- Like a local folder
- Contains
  - Terraform configuration
  - Variables
  - State
  - Credentials and secrets

Terraform Cloud Run

- Terraform runs on its own disposable virtual machines, using a workspace's configuration, variables, and state
- Essentially a build server
- Connects to the Cloud provider with credentials/secrets

Authenticate to Azure

- Managed Identity
- Service Principal

Get the subscription ID

```
# get the subscription id
az account show
{
  "id": "da5f3ff5-1234-1234-12345-123456789"
}

# create the service principal
az ad sp create-for-rbac --name="Lab" --role="Contributor" --scopes="/subscriptions/[SUBSCRIPTION_ID]"

{
  "appId": "9999999-8888-123-456-4822b20ccc15",
  "displayName": "Lab",
  "password": "p@sswo0rd_-9x",
  "tenant": "8888888-666-12345-a123-3ef050c91fde"
}
```

Create the Service Principal using the subscription ID

You'll also need the appID, secret and tenantId

main.tf

Organization and workspace backend

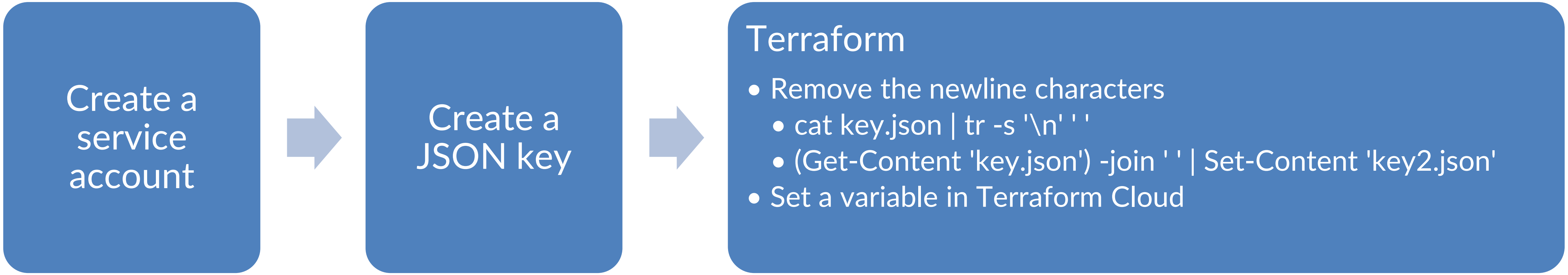
```
terraform {
  cloud {
    organization = "myorg"

    workspaces {
      name = "demo"
    }
  }
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
    }
  }
}

provider "azurerm" {
  features {}
  subscription_id = var.subscription_id
  client_id       = var.client_id
  client_secret   = var.client_secret
  tenant_id       = var.tenant_id
}
```

Service Principal credentials

Authenticate to Google Cloud



Create a service account

Create a JSON key

Terraform

- Remove the newline characters
  - cat key.json | tr -s '\n' ' '
  - (Get-Content 'key.json') -join ' ' | Set-Content 'key2.json'
- Set a variable in Terraform Cloud

main.tf

Organization and workspace backend

Service account credentials

```
terraform {
  cloud {
    organization = "myorg"

    workspaces {
      name = "demo"
    }
  }
  google = {
      source  = "hashicorp/google"
      version = "4.19.0"
    }
  }
}

provider "google" {
  credentials = file("key.json")

  project = var.project
  region  = var.region
  zone    = var.zone
}
```

Demo

In Summary

Infrastructure as Code

- Configuration is written in files
- Allows configurations to be version-controlled
- Promotes auditing
- Prevents environment drifts
- Create reproducible environments, faster
- Consistent, Repeatable, Predictable

HashiCorp
Terraform

What is Terraform?

- Infrastructure as code tool maintained by HashiCorp
  - Provision infrastructure
  - Open source
  - www.terraform.io
- Cloud agnostic
  - GCP, AWS, Azure, DO, Linode, etc
- Uses the HashiCorp Configuration Language (HCL)
- Uses declarative configuration files
- Manages the infrastructure state

END OF LINE

Thank You!