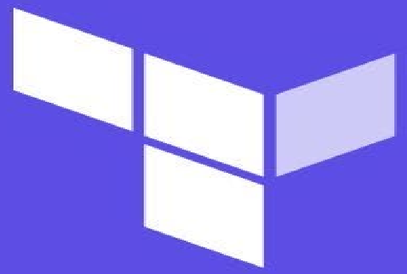


HashiCorp
Terraform

Azure Workshop with Terraform

Agenda

- Overview
- Terraform Workflow
- **Challenge:** Connecting to Azure
- State, Variables, & Outputs
- **Challenge:** ACI
- Interpolations & Modules
- **Challenge:** Azure VM
- **Challenge:** Count
- Backend & Internals
- **Challenge:** Public Registry
- **Challenge:** Scale Set
- Terraform Enterprise
- **Demo:** Terraform Ent/Cloud
- **Certification options**

- **Peter McNaughton – pmwtraining**

Introductions:

Your name,

Job role,

Experience in IT ,

Course expectations.

What’s in this Workshop

Slides -> Challenges -> Slides -> Challenges -> Advanced

Timings – 9:30 –16:00 – 16:30 (UTC +1)

Breaks

Lunch 12:30 – 13.30

Workshop Expectations

- Purpose
  - Why are we here?
- Gives
  - What do you need to do?
- Gets
  - What will get you get?

Hashicorp

Company overview

Founded 2012 by Mitchell Hashimoto and Armon Dadgar

Mission We enable organizations to Provision, Secure, Connect, and Run any infrastructure for any application

Key Products

Infrastructure

Packer

Automated machine images from a single source configuration

Vagrant

Single workflow to build and manage development environments

Terraform

Infrastructure automation to provision and manage any cloud service

Networking

Consul

Service mesh across any cloud and runtime platform

Security

Boundary

Secure remote access to applications and critical systems

Vault

Secure management of secrets and sensitive data

Application

Nomad

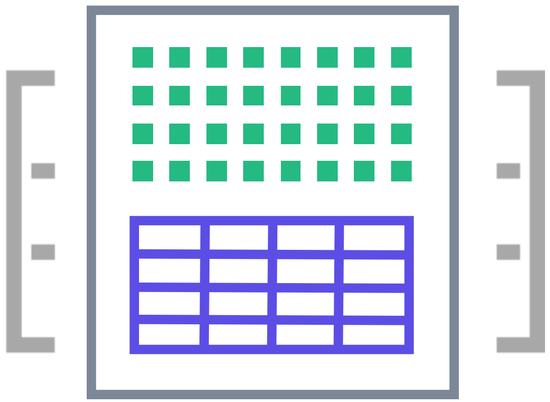
Workload orchestrator and scheduler to deploy and manage applications

Waypoint

One workflow to build, deploy, and release applications across platforms

The shift to hybrid infrastructure

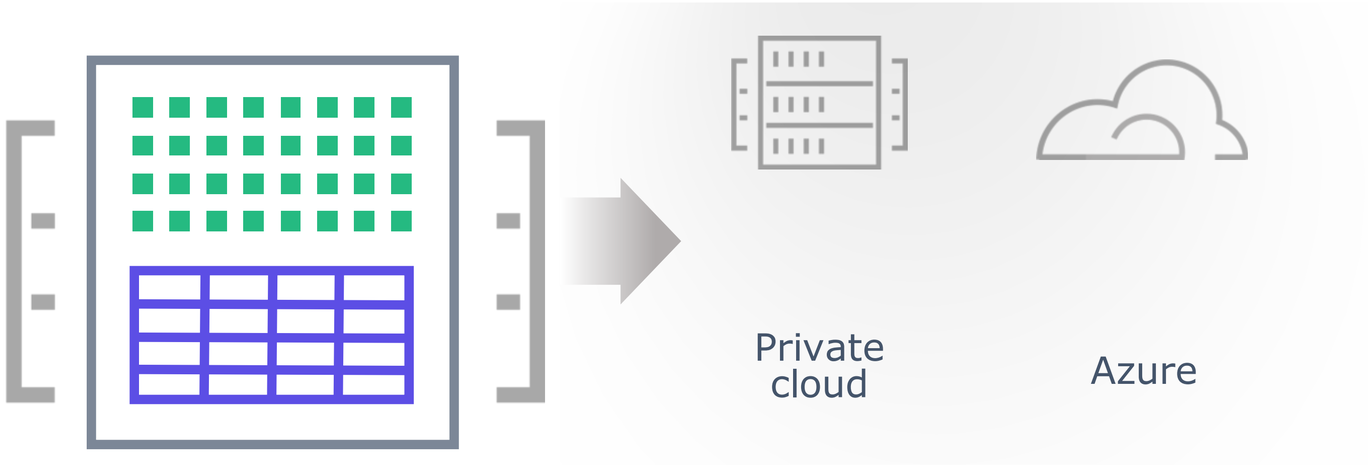
TRADITIONAL
DATACENTER



The shift to hybrid infrastructure

TRADITIONAL DATACENTER

EARLY CLOUD



The shift to hybrid infrastructure

TRADITIONAL DATACENTER

EARLY CLOUD

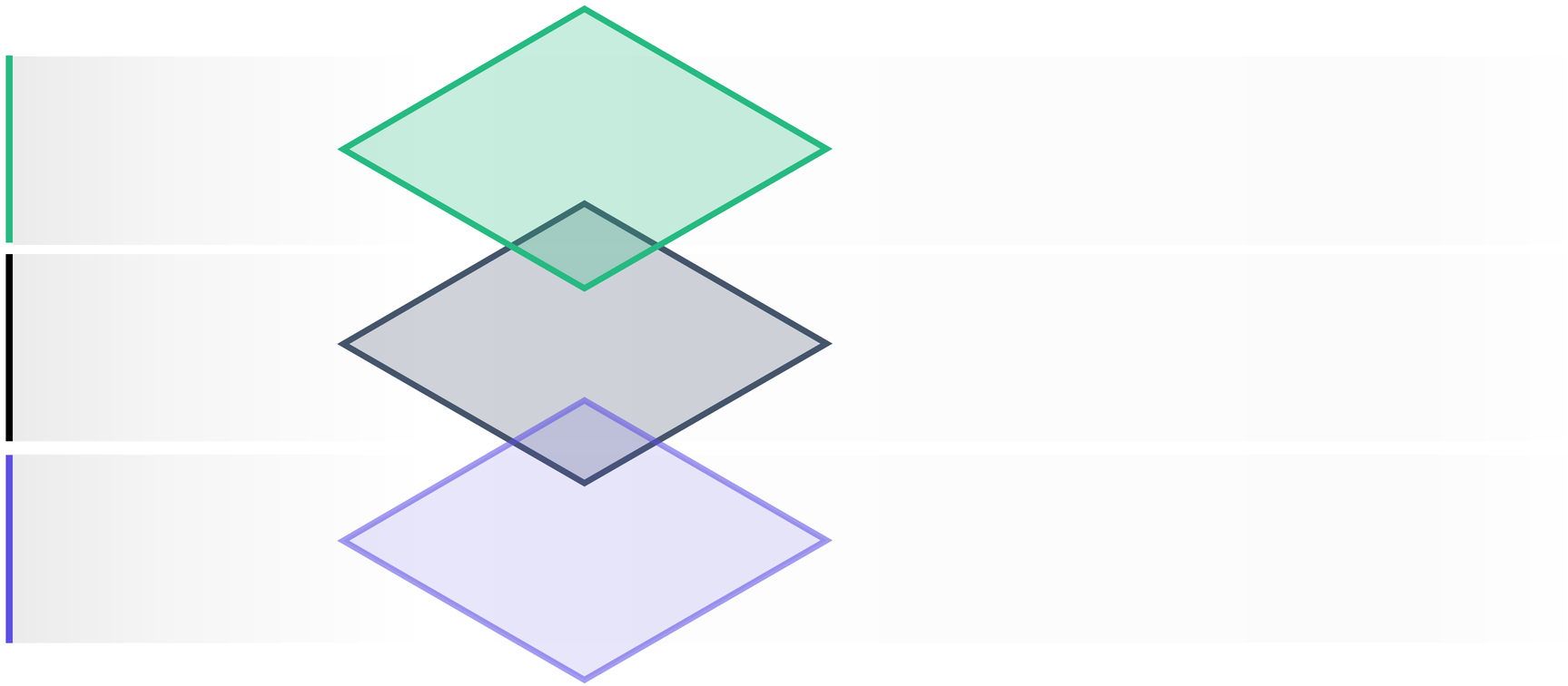
MULTI-CLOUD

The 3 essential elements of infrastructure

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



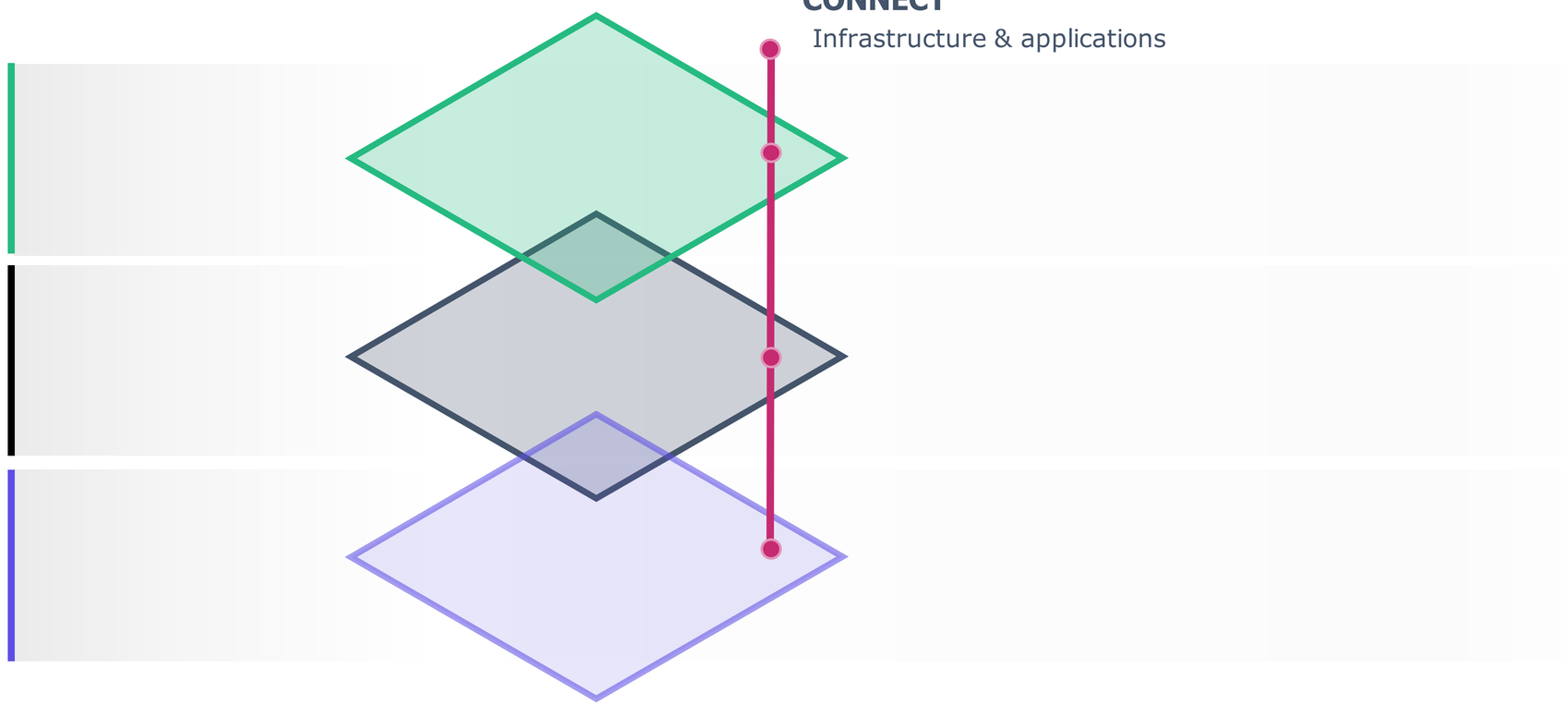
The 4 essential elements of distributed infrastructure

CONNECT
Infrastructure & applications

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



Addressing the challenges of cloud adoption

Cloud challenges for Operations teams

THE PRACTITIONER

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure

- Scale
- Heterogeneity
- Dependency management

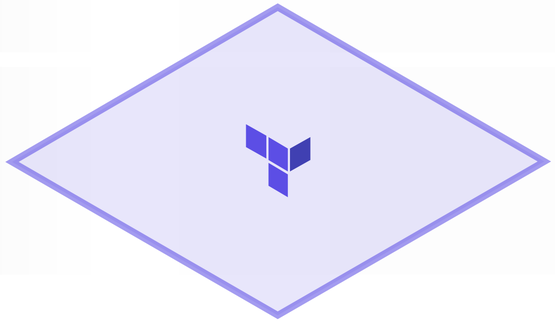
Provision any infrastructure for any application

THE PRACTITIONER

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



- Scale
- Heterogeneity
- Dependency management

Cloud challenges for Security teams

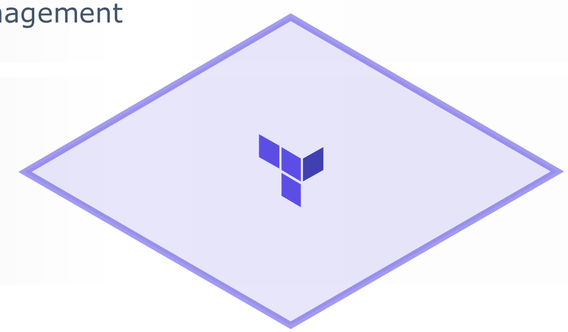
THE PRACTITIONER

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

- Managing distributed secrets
- Encrypt data at-rest and in-flight
- Privilege access management

OPERATIONS
Provision infrastructure



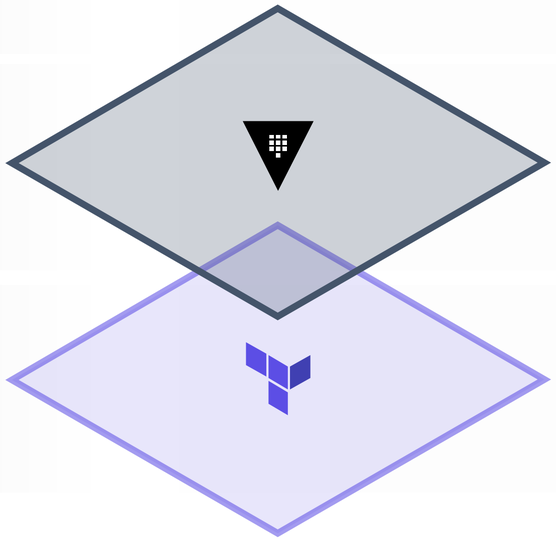
Secure any application and any infrastructure

THE PRACTITIONER

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



- Managing distributed secrets
- Encrypt data at-rest and in-flight
- Privilege access management

Challenges for Developers / Operators

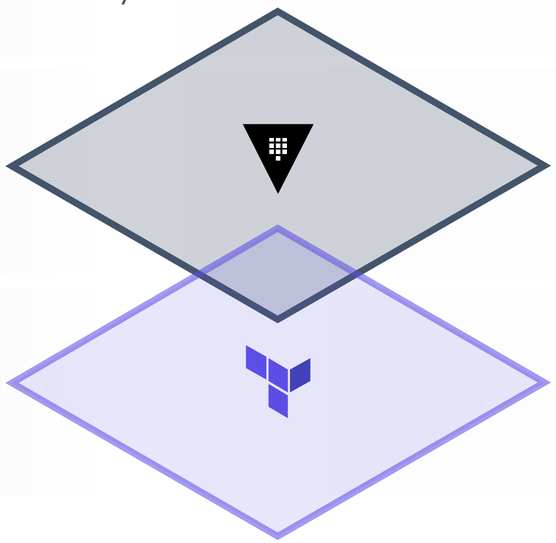
THE PRACTITIONER

DEVELOPMENT
Run applications

- Decoupling application from infrastructure
- Bin packing, machine efficiency

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



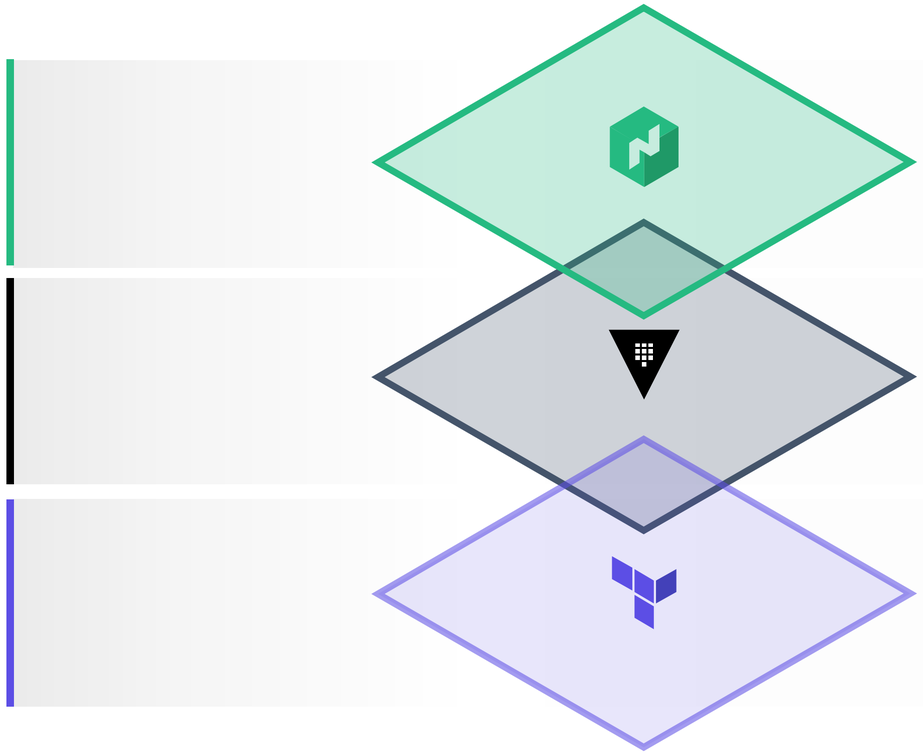
Run any application on any infrastructure

THE PRACTITIONER

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



- Decoupling application from infrastructure
- Bin packing, machine efficiency

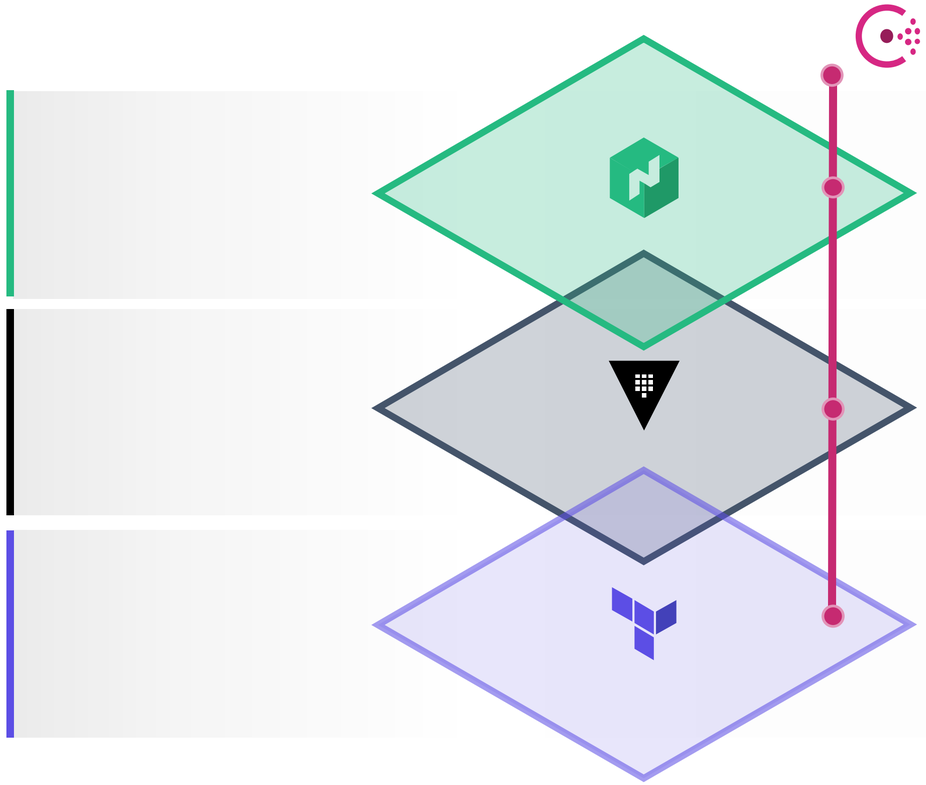
Connect any application and any infrastructure

THE PRACTITIONER

DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



- Service discovery in a dynamic environment
- Knowledge of real-time configuration

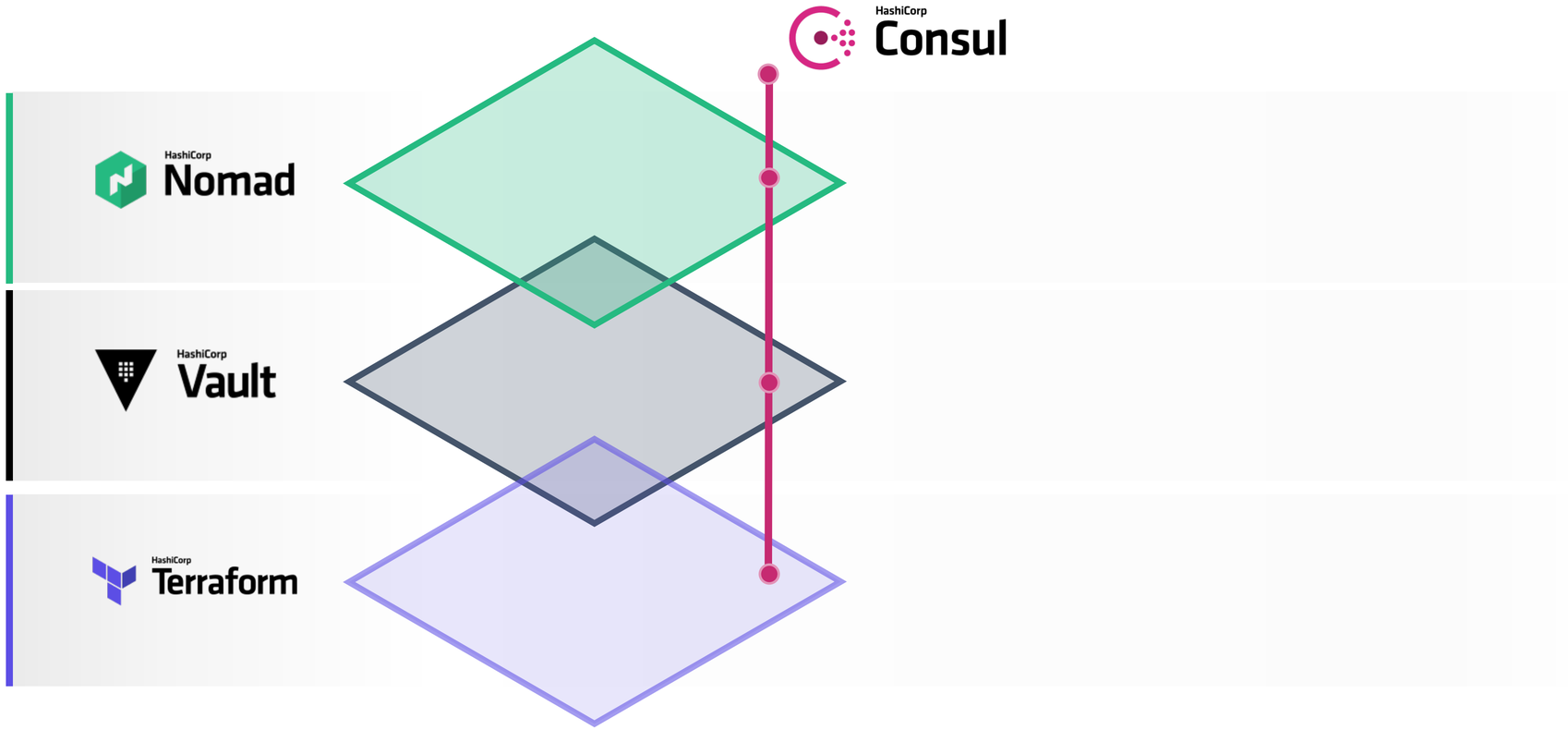
The HashiCorp open source suite

THE PRACTITIONER

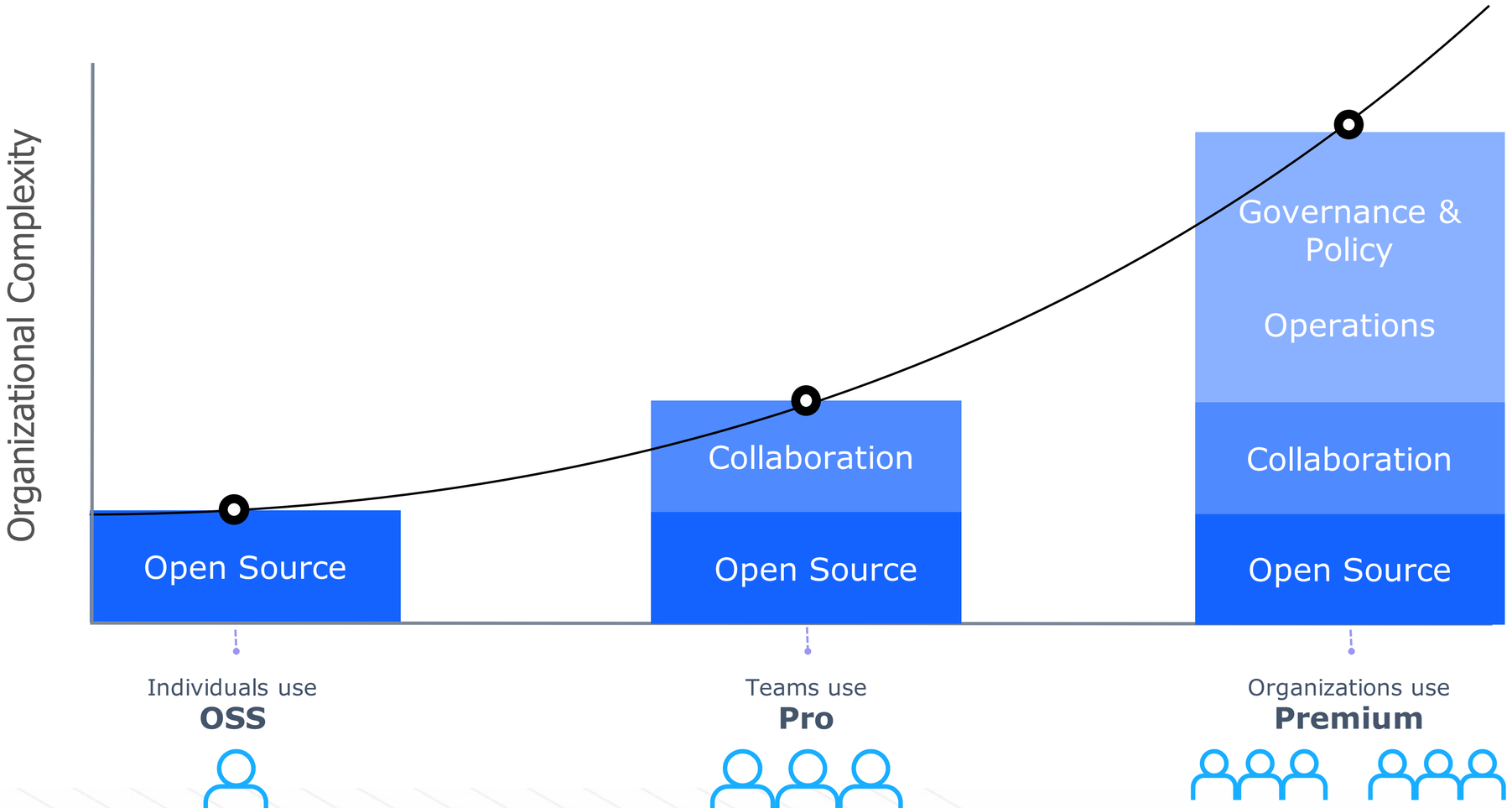
DEVELOPMENT
Run applications

SECURITY
Secure infrastructure & applications

OPERATIONS
Provision infrastructure



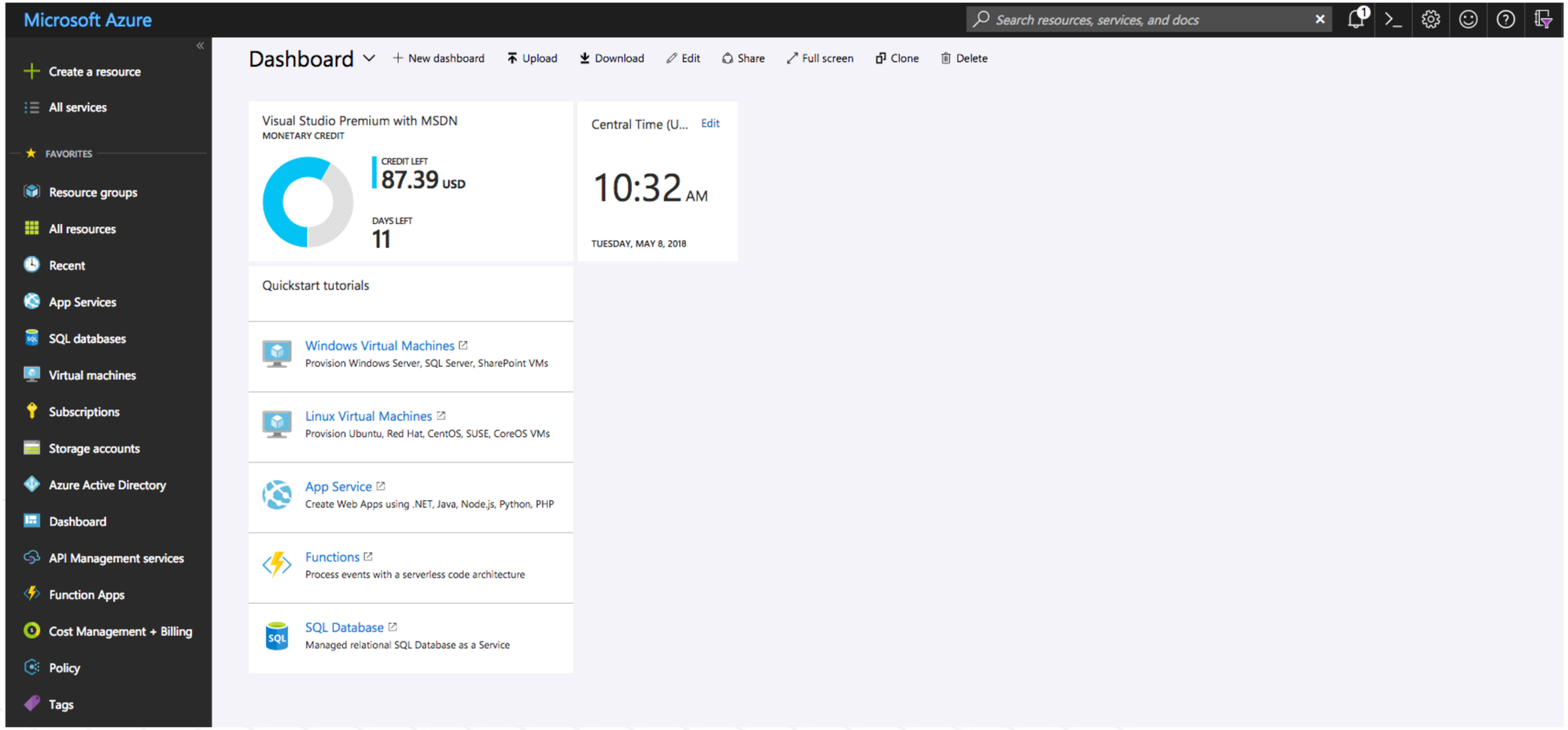
Enterprise products build on open source to address organizational complexity



Azure Resource Manager

Azure Portal

https://portal.azure.com



Azure Resource Manager Templates

ARM Templates

JSON file that defines one or more resources to deploy to a resource group.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualMachines_name": {
            "defaultValue": "Terraform",
            "type": "string"
        }
    },
    "resources": [
        {
            "type": "Microsoft.Compute/virtualMachines",
            "name": "[parameters('virtualMachines_name')]",
            "apiVersion": "2017-12-01",
            "location": "eastus",
            "properties": {
                "hardwareProfile": {
                    "vmSize": "Standard_DS2_v2"
                },
                "storageProfile": {
                    "imageReference": {
                        "publisher": "Canonical",
                        "offer": "UbuntuServer",
                        "sku": "16.04-LTS",
                        "version": "latest"
                    },
                    "osDisk": {
                        "osType": "Linux",
                        "name": "[concat(parameters('virtualMachines_name'),'_OsDisk_1')]",
                        "createOption": "FromImage",
                        "caching": "ReadWrite",
                        "managedDisk": {
                            "id": "[parameters('virtualMachines_Terraform_id')]"
```

ARM Templates Drawbacks

Cannot verify changes to an environment before executing

Failures can be difficult to trace

Deployments cannot cross resource groups (or subscriptions)

Custom tooling to create a full deployment

Why Terraform?

Enables organizations to adopt an Infrastructure as Code

Import existing Infrastructure

Support for Linux and Windows Virtual Machines

Separate planning and execution phases

Represents entire infrastructure and supporting services with a common language

Microsoft <3 Terraform

Microsoft Investing in Terraform

Azure Container Instance

Azure Kubernetes Service

https://aka.ms/terraform

https://aka.ms/terraformdocs

Investing deeply in Terraform on Azure

Posted on August 17, 2017

Corey Sanders, Corporate Vice President, Azure

As customers increase their deployed applications in Azure, we are seeing a growing interest in DevOps tooling on Azure. We also see customers looking to deploy applications across multiple environments, including hybrid and multi-cloud deployments while using the same tooling and enabling the same DevOps experiences. In order to meet these growing needs, I am excited to announce that we are greatly increasing our investment in Terraform, partnering closely with HashiCorp, a well-known voice in the DevOps and cloud infrastructure management space.

Our partnership with HashiCorp goes back to early 2016, where we jointly announced plans to bring full support for Azure Resource Manager across many tools in HashiCorp's portfolio including Packer and Terraform. Since then, our customers have found significant value in the HashiCorp support on Azure.

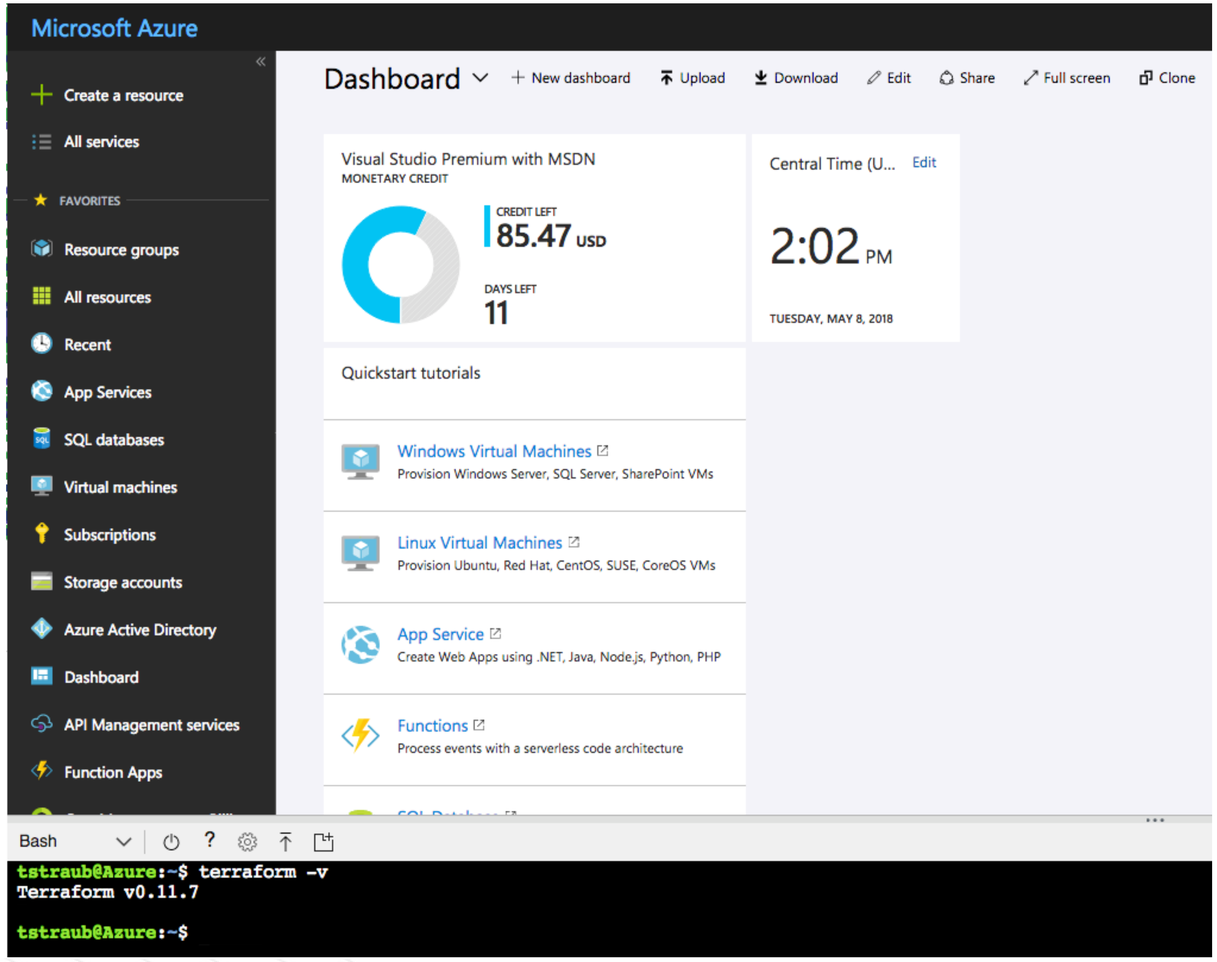
Today, we're extending our partnership and will offer an increasing number of services directly supported by Terraform, including Azure Container Instances, Azure Container Service, Managed Disks, Virtual Machine Scale Sets and others. We want to give additional flexibility to express infrastructure-as-code and to enable many more native Microsoft Azure services to be easily deployed directly through Terraform. Learn more about the Azure provider for Terraform.

HashiCorp Terraform

https://azure.microsoft.com/en-us/blog/investing-deeply-in-terraform-on-azure/

Microsoft <3 Terraform

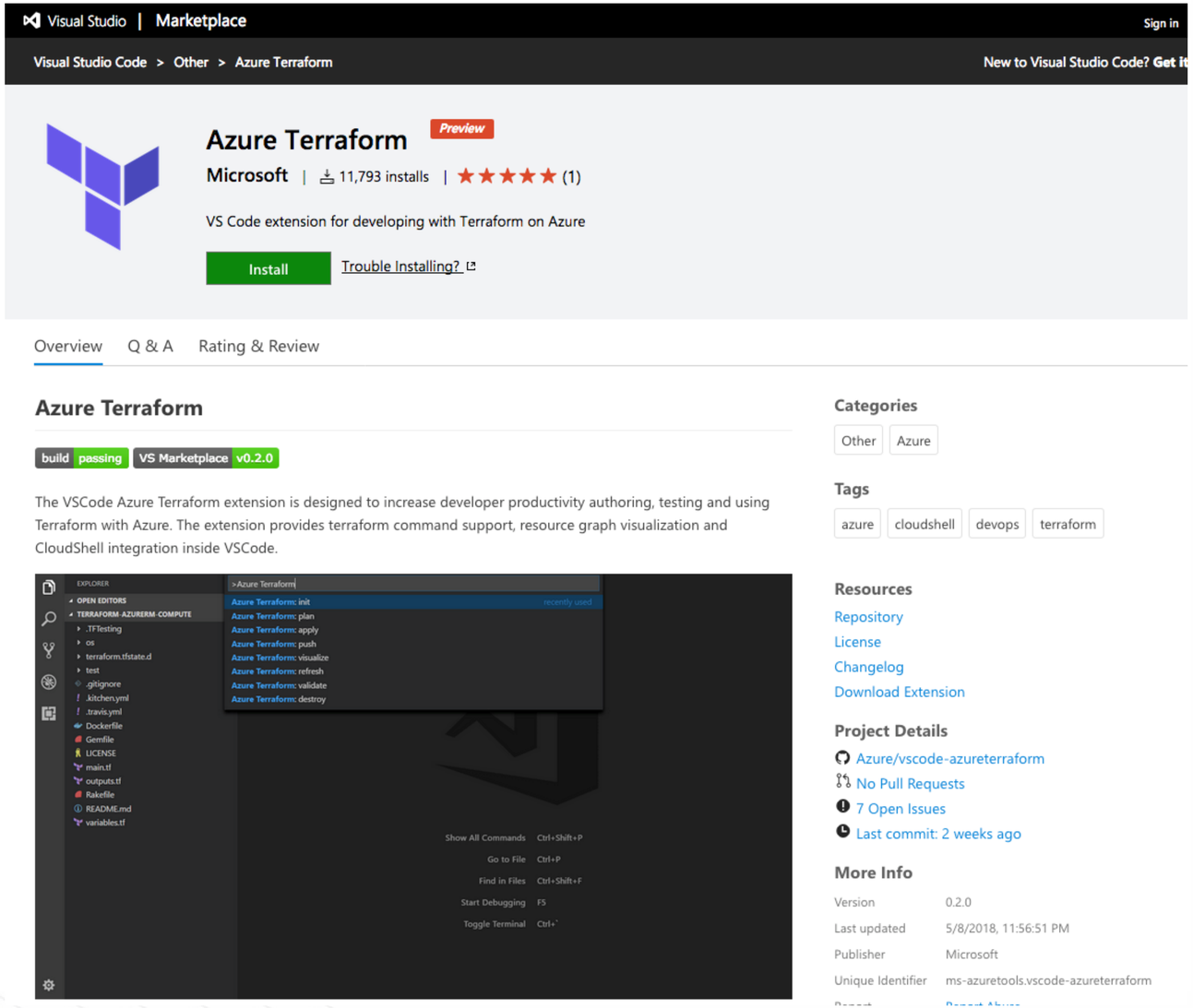
Azure Cloud Shell Integration



[cloud shell bash]

Microsoft <3 Terraform

VS Code Extension

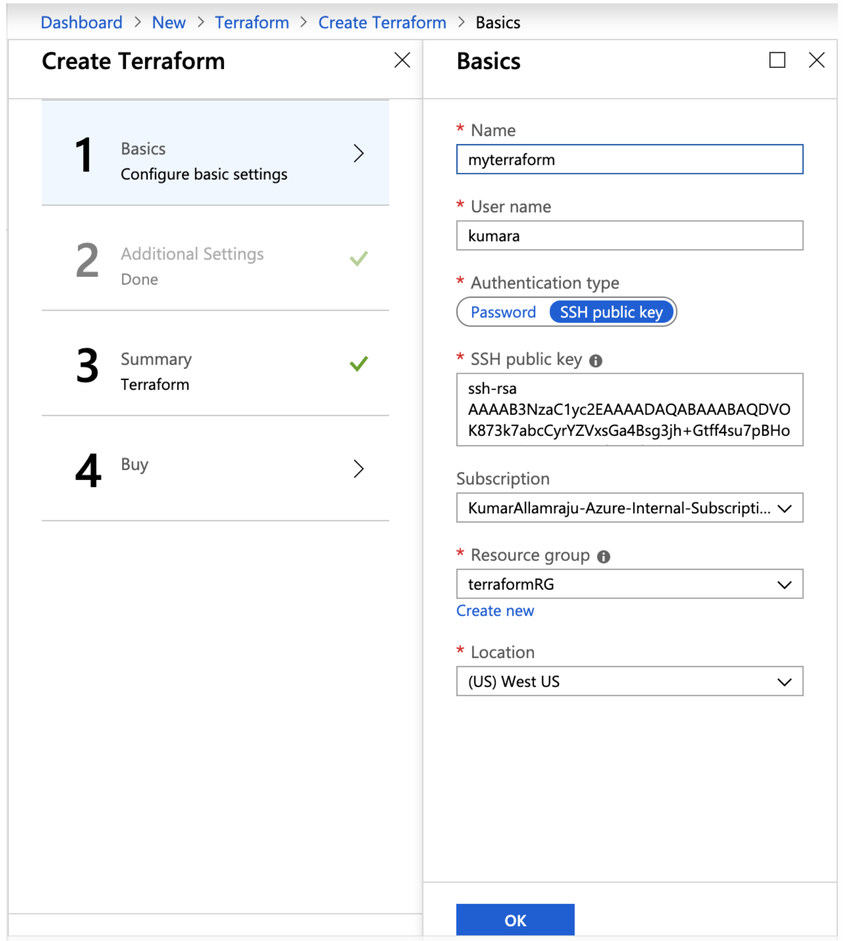


https://docs.microsoft.com/en-us/azure/terraform/terraform-cloud-shell

Microsoft <3 Terraform

Terraform Marketplace Image

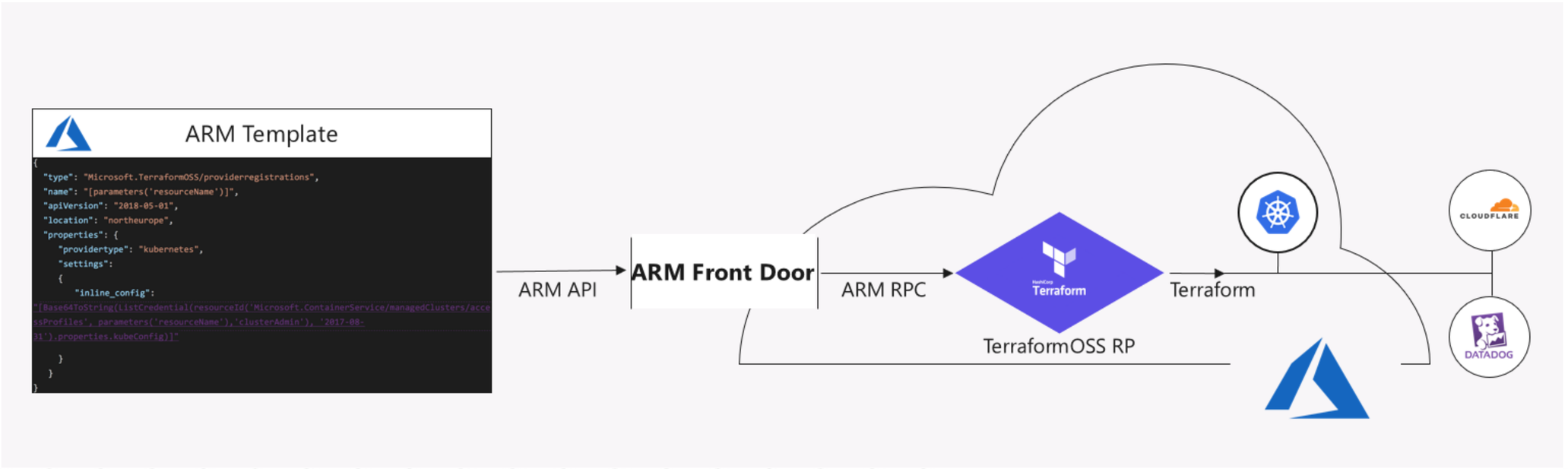
- Creates a VM with system-assigned identity that's based on the Ubuntu 16.04 LTS image.
- Installs the MSI extension on the VM to allow OAuth tokens to be issued for Azure resources.
- Assigns RBAC permissions to the managed identity, granting owner rights for the resource group.
- Creates a Terraform template folder (tfTemplate).
- Pre-configures a Terraform remote state with the Azure back end.



https://aka.ms/aztfmkt

Microsoft <3 Terraform

Terraform 3rd Party Azure Provider



https://azure.microsoft.com/en-us/blog/introducing-the-azure-terraform-resource-provider/

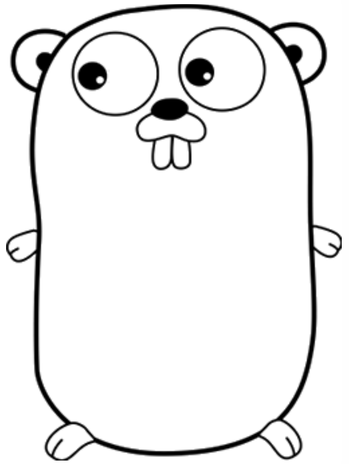
Terraform

Terraform

https://www.terraform.io/

Open Source Software
https://github.com/hashicorp/terraform

Written in GO
https://golang.org/



Terraform Documentation

- Configuration
- Commands (CLI)
- Import
- State
- Providers
- Provisioners
- Modules
- Backends
- Plugins
- Internals

Terraform Documentation

Welcome to the Terraform documentation! This documentation is more of a reference guide for all available features and options of Terraform. If you're just getting started with Terraform, please start with the introduction and getting started guide instead.

https://www.terraform.io/docs

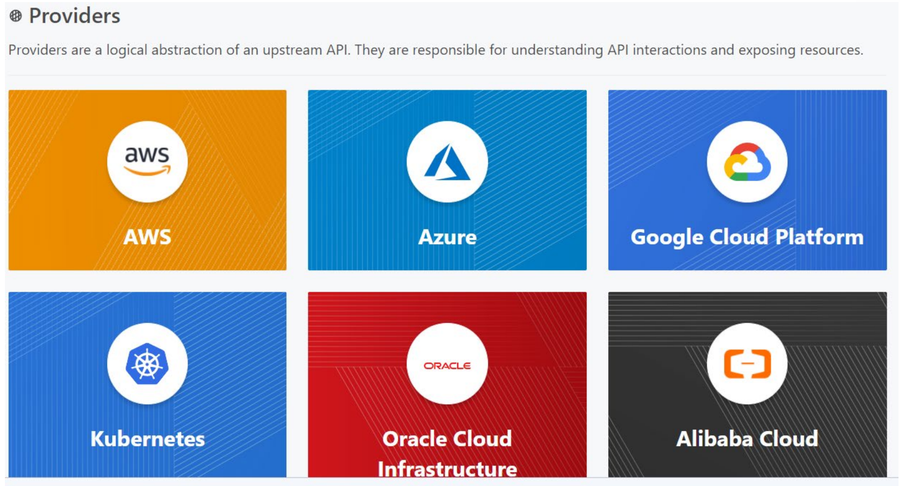
Terraform Ecosystem

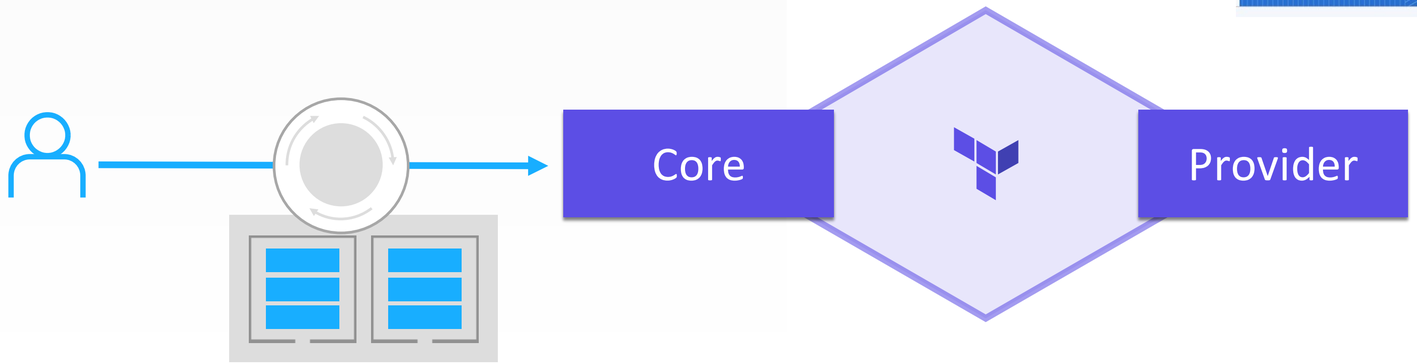
Partnerships with many technology providers, and 4000+ modules

VERSION-CONTROL SYSTEMS

Bitbucket GitLab GitHub

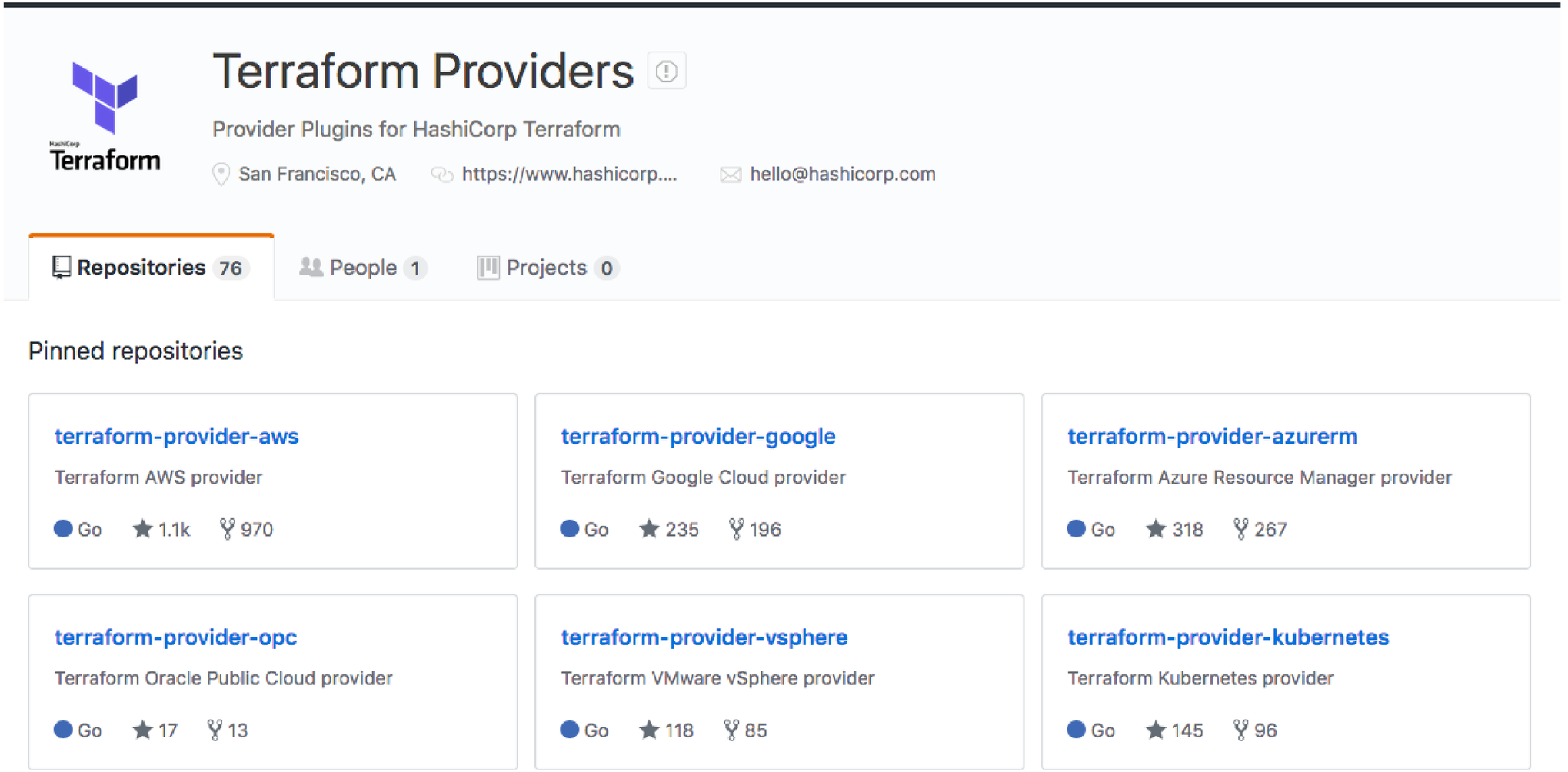
INFRASTRUCTURE PROVIDERS





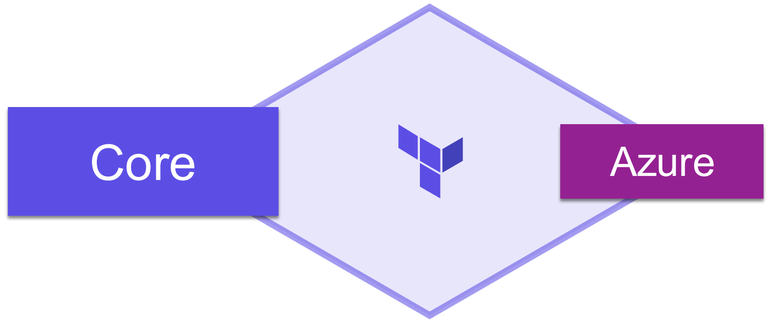
List of all providers [Providers]

Terraform Providers



https://github.com/terraform-providers

Provision any infrastructure for any application



Azure Provider

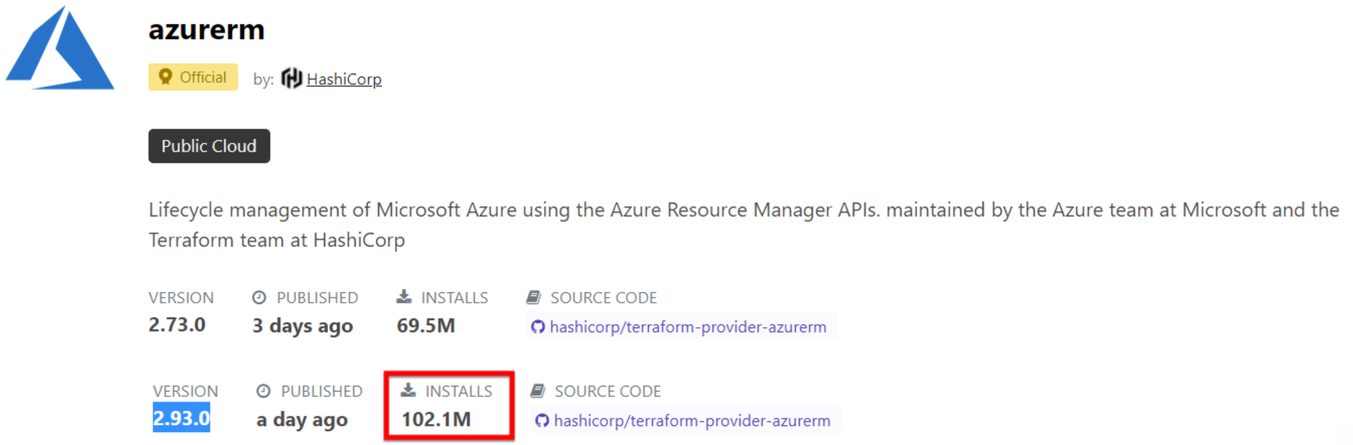
1000+ resources

Very active code base

Frequent releases

- Jan 2018 -> v1.0.0
- May 2018 -> v1.4.0
- **...**
- Aug 2021 -> v2.73.0
- **Jan 2022 -> v2.93.0**
- **De 2023 -> v3.8.5.0**

Great visibility into progress



https://github.com/hashicorp/terraform-provider-azurerm

Azure Provider Documentation

- All Providers
- Azure Provider
  - Authenticating via the Azure CLI
  - Authenticating via a Service Principal (Shared Account)
  - Authenticating via Managed Service Identity
- Data Sources
  - azurerm_application_security_group
  - azurerm_app_service
  - azurerm_app_service_plan
  - azurerm_builtin_role_definition
  - azurerm_cdn_profile
  - azurerm_client_config

Azure Provider

The Azure Provider is used to interact with the many resources supported by Azure Resource Manager (AzureRM) through its APIs.

Note: This supercedes the legacy Azure provider, which interacts with Azure using the Service Management API.

Use the navigation to the left to read about the available resources.

Creating Credentials

Terraform supports authenticating to Azure through a Service Principal or the Azure CLI.

We recommend using a Service Principal when running in a shared environment (such as within a CI server/automation) - and authenticating via the Azure CLI when you're running Terraform locally.

https://www.terraform.io/docs/providers/azurerm

Hashicorp Configuration Language

Hashicorp configurationLangauge (HCL) is designed for human consumption so users can quickly interpret and understand their infrastructure configuration.

HCL includes a full JSON parser for machine-generated configurations.

https://github.com/hashicorp/hcl

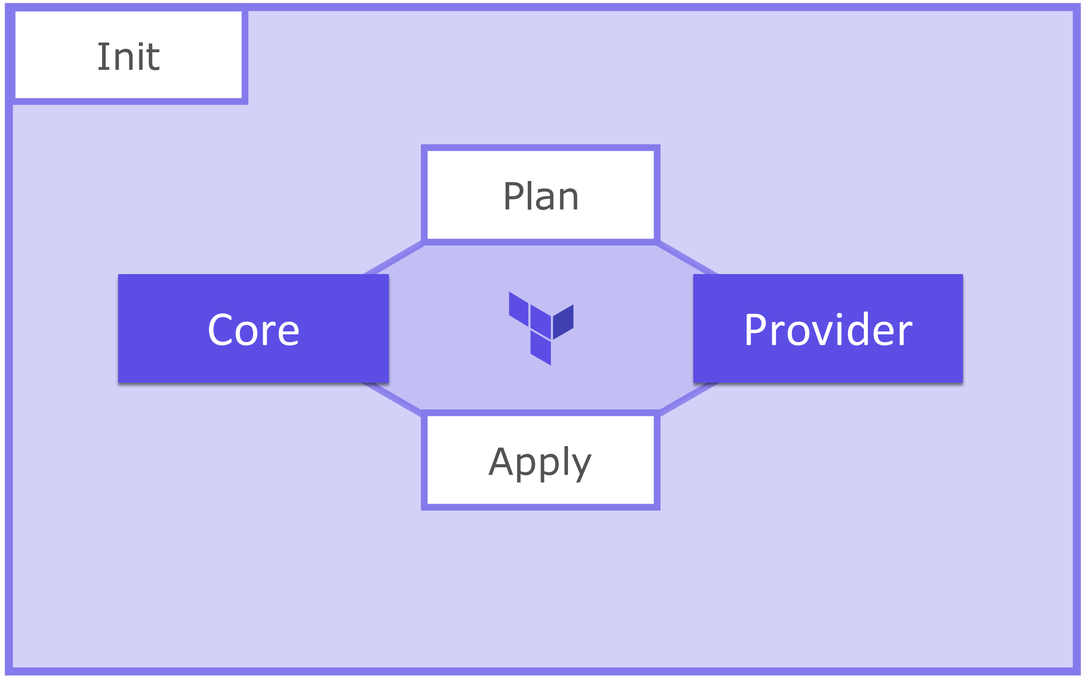
main.tf

```
provider "azurerm" {
features {}
}

resource "azurerm_resource_group" "myfirstrg" {
  name     = "myfirstresourcegroup"
  location = "uksouth"
}
```

Terraform in Action

Terraform Workflow



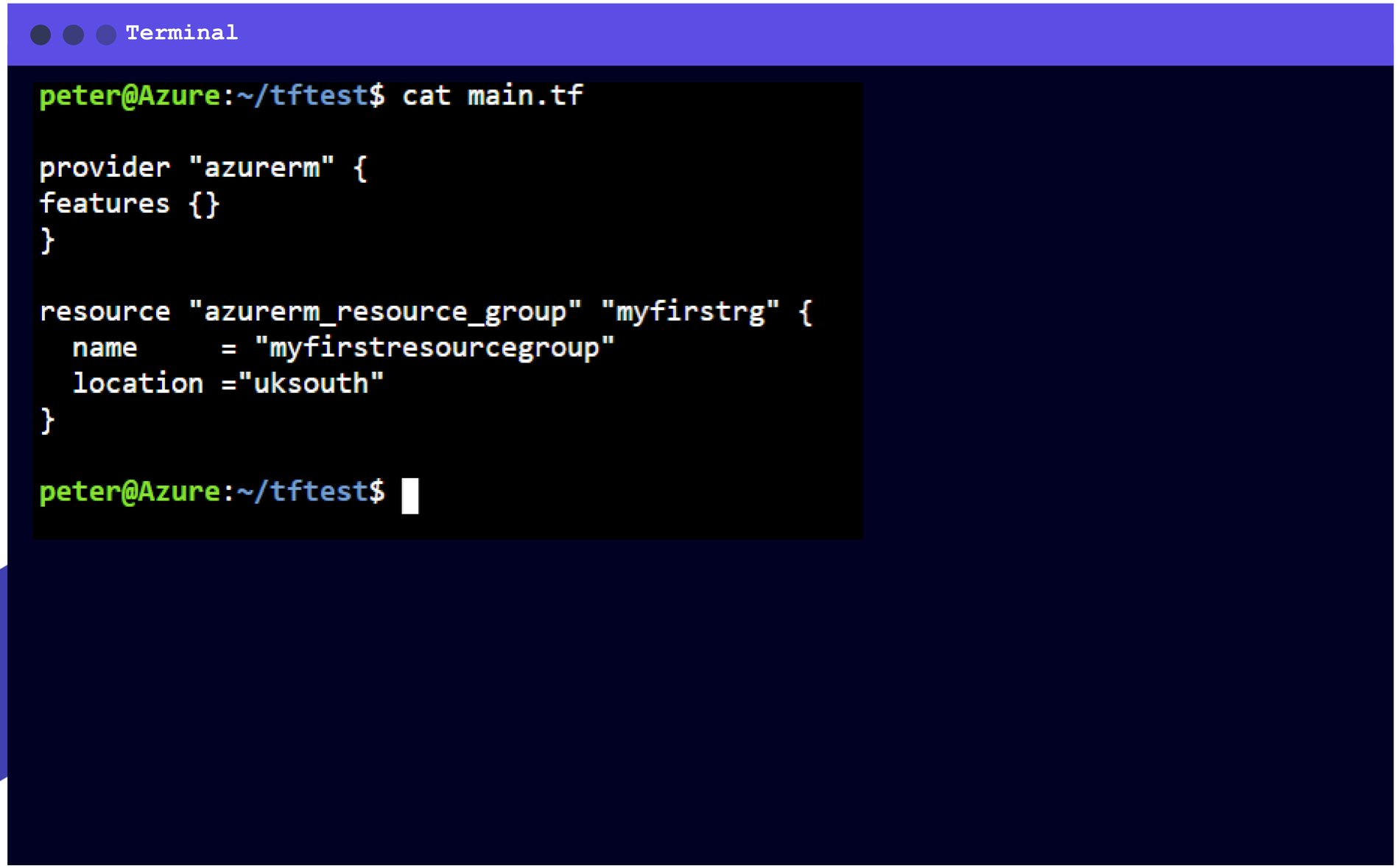
DEMO

Init

Plan

Appy

Destroy


Terminal
peter@Azure:~/tftest$ cat main.tf
provider "azurerm" {
features {}
}
resource "azurerm_resource_group" "myfirstrg" {
name = "myfirstresourcegroup"
location ="uksouth"
}
peter@Azure:~/tftest$

HashiCorp

Terminal

```
peter@Azure:~/tftest$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/azurerm...
- Installing hashicorp/azurerm v2.93.0...
- Installed hashicorp/azurerm v2.93.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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
should now work.
```

HashiCorp

Terminal

```
peter@Azure:~/tftest$ terraform plan

Terraform used the selected providers to generate the fo
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.myfirstrg will be created
  + resource "azurerm_resource_group" "myfirstrg" {
      + id       = (known after apply)
      + location = "uksouth"
      + name     = "myfirstresourcegroup"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

HashiCorp

Terminal

```
peter@Azure:~/tftest$ terraform apply

Terraform used the selected providers to generate the following execution plan
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.myfirstrg will be created
  + resource "azurerm_resource_group" "myfirstrg" {
      + id       = (known after apply)
      + location = "uksouth"
      + name     = "myfirstresourcegroup"
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

azurerm_resource_group.myfirstrg: Creating...
azurerm_resource_group.myfirstrg: Creation complete after 1s [id=/subscription

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
peter@Azure:~/tftest$
```

HashiCorp

Terminal

```
peter@Azure:~/tftest$ terraform destroy -auto-approve
azurerm_resource_group.myfirstrg: Refreshing state... [id=/subscripti

Terraform used the selected providers to generate the following execu
  - destroy

Terraform will perform the following actions:

  # azurerm_resource_group.myfirstrg will be destroyed
  - resource "azurerm_resource_group" "myfirstrg" {
      - id       = "/subscriptions/324bd9d9-0748-480e-81d1-ffdbe26503
      - location = "uksouth" -> null
      - name     = "myfirstresourcegroup" -> null
      - tags     = {} -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.
azurerm_resource_group.myfirstrg: Destroying... [id=/subscriptions/32
azurerm_resource_group.myfirstrg: Still destroying... [id=/subscripti
azurerm_resource_group.myfirstrg: Destruction complete after 15s

Destroy complete! Resources: 1 destroyed.
peter@Azure:~/tftest$
```

HashiCorp

Terraform Configuration

Terraform

```
terraform {
  required_version = "= 0.11.15"
}
```

=	Exact version equality	= 1.1.4
!=	Exclude version	!= 0.11.14
> >= < <=	Version comparison	>= 1.0.5
~>	Pessimistic constraint	~> 0.11.7

https://www.terraform.io/docs/configuration/terraform.html

AzureRm Provider

```
provider "azurerm" {
  version = "= 2.7.3"
}
```

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs

Authentication to Azure

Azure CLI

az login

- Uses your own credentials
- Great for local development
- Also how the Azure Cloud Shell works

Authentication to Azure

Service Principal

- Preferred for shared environments
- Works well in automation
- Also how Terraform Enterprise works

Authentication to Azure

Service Principal

```
terraform{ ...}

provider "azurerm" {
  version = " ~ >2.0"

  subscription_id =     "<azure_subscription_id>"
  client_id       =     "<service_principal_appid>"
  client_secret   =      "<service_principal_password>"
  tenant_id       =     "<azure_subscription_tenant_id>"}
```

Authentication to Azure

Service Principal

```
export ARM_TENANT_ID=
export ARM_SUBSCRIPTION_ID=
export ARM_CLIENT_ID=
export ARM_CLIENT_SECRET=
```

Authentication to Azure

Hashicorp/Azure vault

Preferred Option

https://registry.terraform.io/providers/hashicorp/vault/latest/docs

Azure Location

```
resource "azurerm_resource_group" "main" {
  location = "eastus"
}
```

```
resource "azurerm_resource_group" "main" {
  location = “East US”
}
```

Challenge

- 00 - Getting Started
- 01 - Connecting to Azure

Challenge

- 00 - Getting Started
- 01 - Connecting to Azure

Terraform State

Terraform State

File that stores state about your managed infrastructure

Keeps track of resource metadata

Used to map real world resources to your configuration.

Local terraform.tfstate file by default

https://www.terraform.io/docs/state

Terraform State

Config: Desired State
State: Current State
Diff: Delta of {Desired & Current}
Plan: Presents Diff
Apply: Resolves Diff

https://www.terraform.io/docs/state/remote.html

Terraform State

State file can contain sensitive data

https://www.terraform.io/docs/state/sensitive-data.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**
		azurerm_virtual_machine.app	

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**
		azurerm_virtual_machine.app	**NOOP**

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**
		azurerm_virtual_machine.app	**NOOP**
azurerm_virtual_machine.app		azurerm_virtual_machine.app	

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**
		azurerm_virtual_machine.app	**NOOP**
azurerm_virtual_machine.app		azurerm_virtual_machine.app	**RECREATE**

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**
		azurerm_virtual_machine.app	**NOOP**
azurerm_virtual_machine.app		azurerm_virtual_machine.app	**RECREATE**
	azurerm_virtual_machine.app		

https://www.terraform.io/docs/state/remote.html

Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app		**CREATE**
azurerm_virtual_machine.app	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**NOOP**
	azurerm_virtual_machine.app	azurerm_virtual_machine.app	**DELETE**
		azurerm_virtual_machine.app	**NOOP**
azurerm_virtual_machine.app		azurerm_virtual_machine.app	**RECREATE**
	azurerm_virtual_machine.app		**REMOVE FROM STATE**

https://www.terraform.io/docs/state/remote.html

Variables

Variables

```
variable "location" {
  default     =  eastus
  type  = string
  description = " location to deploy Azure Infrastructure"
}


resource "azurerm_resource_group" "module" {
  name     = "my-rg"
  location = "var.location"
}
```

https://www.terraform.io/docs/configuration/variables.html

Variables - Types

```
variable "key" {
  type        = "string"
  default     = "value"
}
```

https://www.terraform.io/docs/configuration/variables.html

Variables - Types

```
variable "images" {
  type = "map"

  default = {
    "dev"  = "image1"
    "test" = "image2"
    "prod" = "image3"
  }
}
```

https://www.terraform.io/docs/configuration/variables.html

Variables - Types

```
variable "rules" {
  type    = "list"
  default = ["rule1", "rule2"]
}
```

https://www.terraform.io/docs/configuration/variables.html

Variables - Types

```
variable "enabled" {
  default = false
}
```

https://www.terraform.io/docs/configuration/variables.html

Variables - Scoping

Variables are valid within the current module level.

```
test
├── app1.tf
├── app2.tf
└── variables.tf
```

https://www.terraform.io/docs/configuration/variables.html

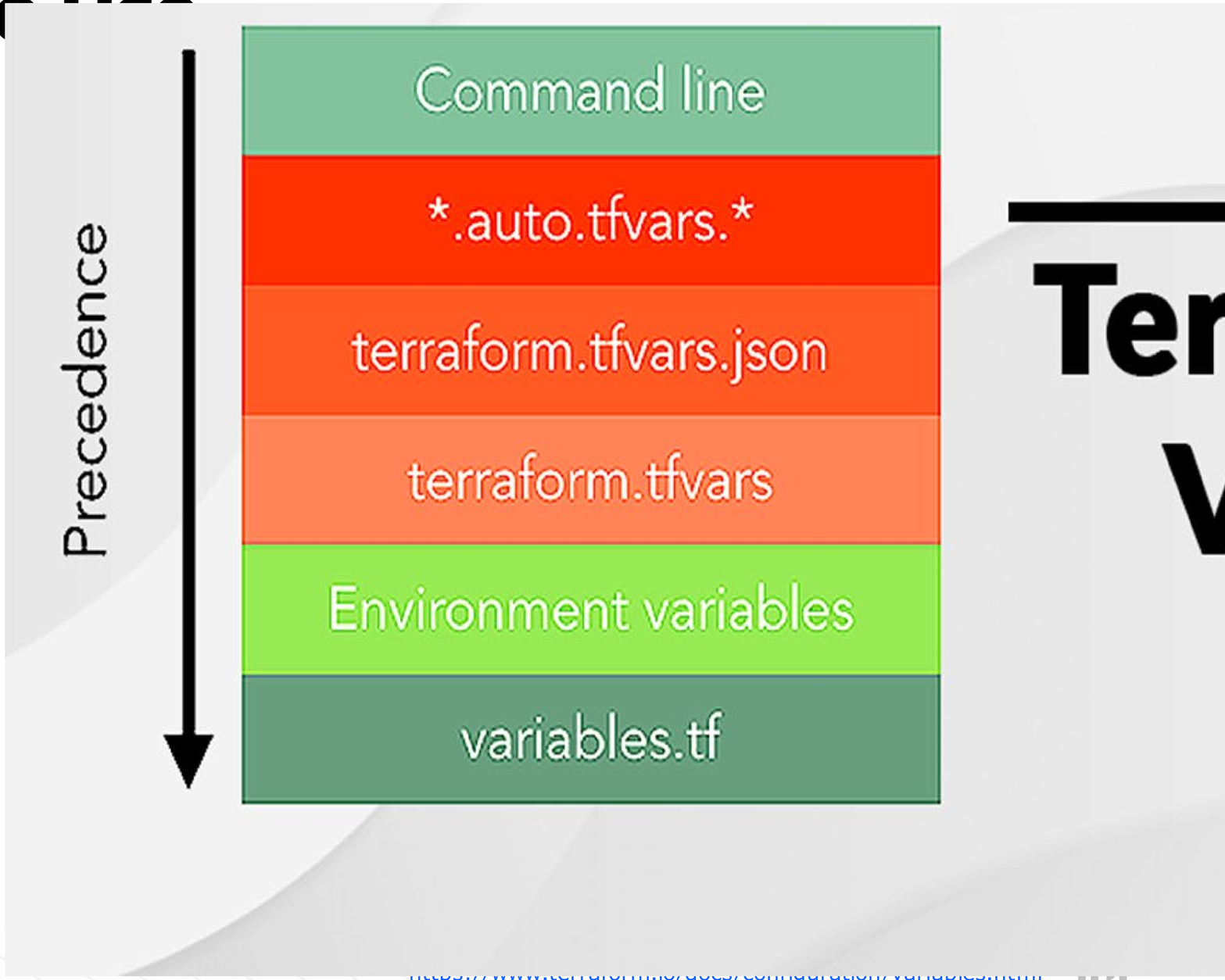
Variables – How to Use

Variable Files

- terraform.tfvars
- *.auto.tfvars

```
foo   = "bar"
valid = true

somelist = [
  "one",
  "two",
]
```



https://www.terraform.io/docs/configuration/variables.html

Variables – How to Use

Command Line Parameters

terraform plan –var ‘foo=bar’

terraform plan –var ‘foo=bar’ –var ‘bar=foo’

https://www.terraform.io/docs/configuration/variables.html

Variables – How to Use

Environment Variables (TF_VAR_*)

```
export TF_VAR_foo=bar

export TF_VAR_bar=foo
```

https://www.terraform.io/docs/configuration/variables.html

Outputs

Outputs

Outputs define useful values that will be highlighted to the user when Terraform applies:

- IP addresses
- Usernames
- Computed Values

Easily extract and query information from all resources

https://www.terraform.io/docs/configuration/outputs.html

Outputs

```
output "vm_private_ips" {
  value = "${azurerm_network_interface.app.private_ip_address}”
  description = "Dynamic Private IP Addresses”
}
```

https://www.terraform.io/docs/configuration/outputs.html

Challenge

- 02 - Provision Azure Container Instance

Outputs

```
Outputs:
vm_private_ips = [
 10.0.0.1,
  10.0.0.2,
  10.0.0.3
]
```

Interpolation

Interpolation

Embedded within strings in Terraform you can interpolate other values – Major changes from 0.12.0 onwards.

```
"${var.name}-rg"
```

The interpolation syntax is powerful and allows you to reference variables, attributes of resources, call functions, etc.

https://www.terraform.io/docs/configuration/interpolation.html

Interpolations – Count

```
variable "count" {
  default = 2
}

resource "azurerm_network_interface" "web" {
  name  = "nic-${count.index}"
  count = "${var.count}"
}

resource "azurerm_virtual_machine" "web" {
  count                 = "${var.count}"
  network_interface_ids =      ["${element(azurerm_network_interface.web.*.id,
            count.index)}"]
}
```

Interpolations – Built-In Functions

concat(list1, list2, ...) - Combines two or more lists into a single list.
concat(var.base_tags, var.additional_tags)

element(list, index) - Returns a single element from a list at the given index.

element(azurerm_subnet.foo.*.id, count.index)
element(var.list_of_strings, 2)

https://www.terraform.io/docs/configuration/interpolation.html

Interpolations - Conditionals

```
resource "azurerm_virtual_machine" "web" {

  subnet = "${var.size == "small" ? var.small_vm : var.large_vm}"

}
```

https://www.terraform.io/docs/configuration/interpolation.html

Interpolations – Built-In Functions

format(format, args, ...) - Formats a string according to the given format. The syntax for the format is standard sprintf syntax.

Example to zero-prefix a count, used commonly for naming servers format("web-%03d", 2).

https://www.terraform.io/docs/configuration/interpolation.html

Interpolation – Math

```
resource "azurerm_virtual_machine" "module" {
  name = "myvm-${count.index + 1}"
}
```

```
resource "azurerm_virtual_machine" "module" {
  name = "${format("myvm-%03d", count.index + 1)}"
}
```

https://www.terraform.io/docs/configuration/interpolation.html

Interpolations – locals

```
variable "db_name" {
  default = "dev"
}

locals {
  module_name = "${var.db_name}-sql"
}
```

https://www.terraform.io/docs/configuration/locals.html

Modules

Modules

Root module:

The current working directory when you run terraform init/plan/apply, holding the Terraform configuration files.

It is itself a valid module.

Variable scoping:

Variables are scoped only to the module in which they are declared. There is **no** inheritance!

https://www.terraform.io/language/modules/develop

Module Usage

```
module ”mymodule" {
  source   = ”path/to/module"

  name     = "mysuperapp"
  ...
}
```

https://www.terraform.io/language/modules/develop

Module Sources

Local

Github

Generic git repo

Bitbucket

HTTP URLs

Registry (support versioning)

https://www.terraform.io/language/modules/sources

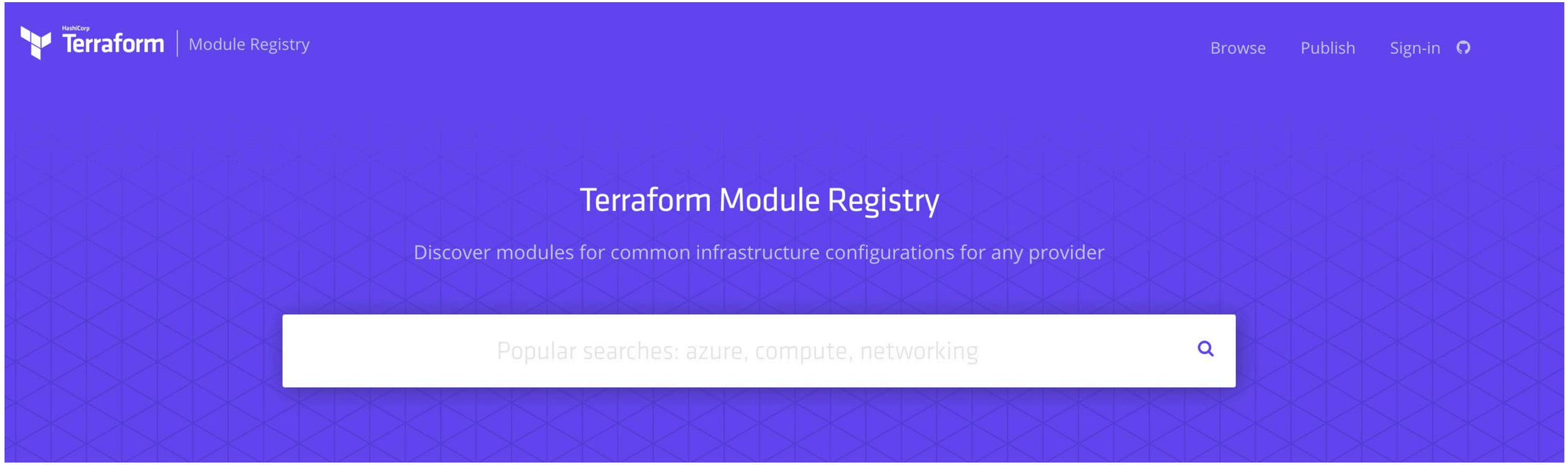
Module Usage

```
module "windowsservers" {
  source        = "Azure/compute/azurerm"
  version       = "1.1.5"

  location      = "eastus"
  vm_hostname    = "mywinvm"
  vm_os_simple   = "WindowsServer"
}
```

https://www.terraform.io/docs/modules

Public Module Registry

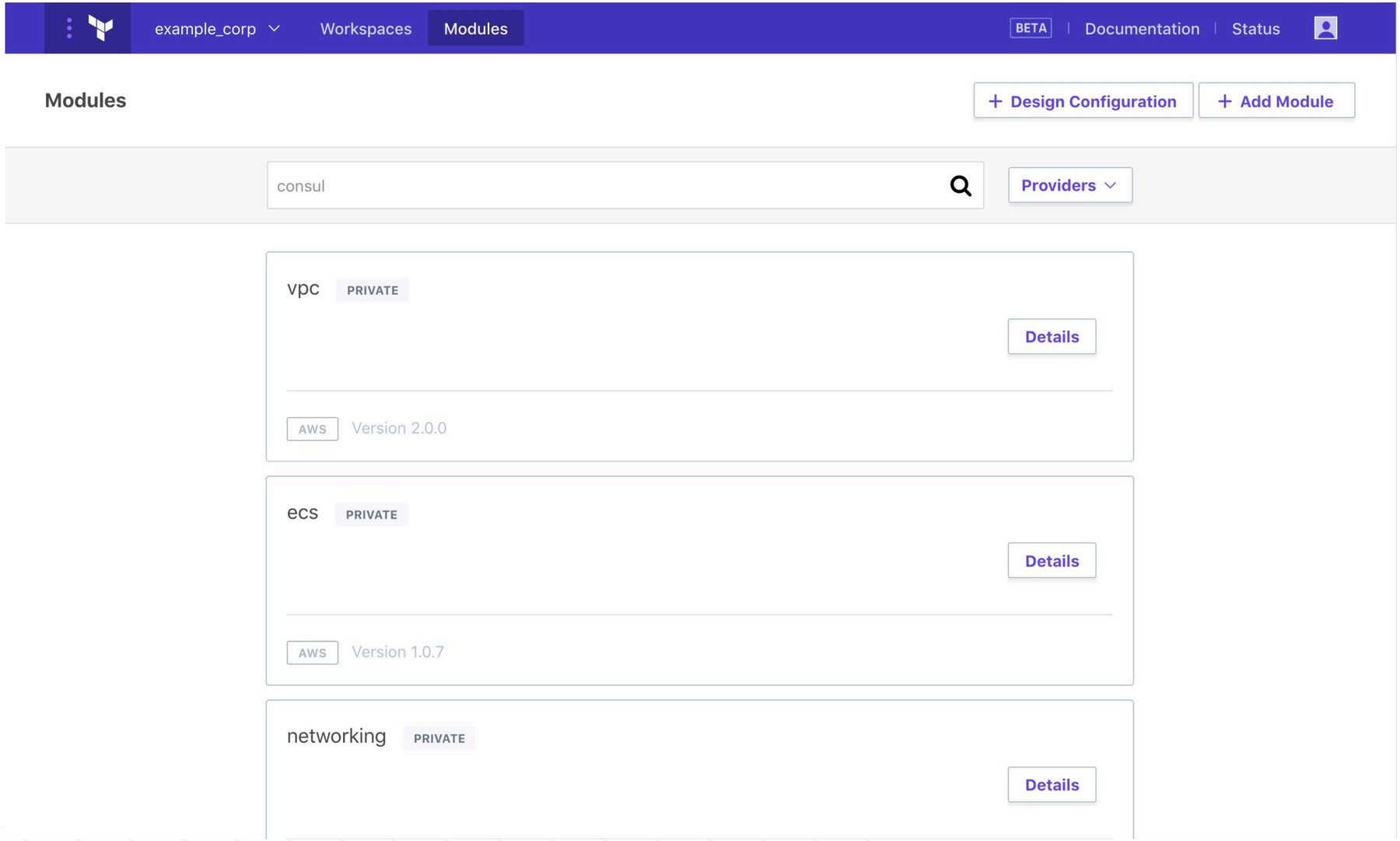


Find Terraform Modules

Use and learn from verified and community modules

https://registry.terraform.io

Private Module Registry



https://www.terraform.io/docs/enterprise/registry

DEMO

Challenge

06 – Scaling with LB

Backend

Backend

Determines how state is loaded and how an operations are executed.

Enables non-local file state storage, remote execution, etc.

By default, Terraform uses the "local" backend.

https://www.terraform.io/language/settings/backends

Backend- azurerm

```
terraform {
  backend "azurerm" {
    ressource_group_name      =  "pm_portal_rg"

    storage_account_name      = "myorgeastus8x76ghf"
    container_name            = "terraformstate"
    key                       = "dev.terraform.tfstate"
  }
}
```

https://www.terraform.io/language/settings/backends/azurerm

Backend

State Locking – Allows for teams to work together.

Backend Types:

- azure SA
- artifactory
- Consul
- Many more …

Terraform Enterprise/Cloud Manages this for you!

https://www.terraform.io/language/settings/backends/

Challenge

07 – Remote State with Azure

Terraform Internals

.terraform

```
$ tree .terraform/
.terraform/
├── modules
│   ├── b3da51524bda220ab3e4db8209644422
│   │   └── main.tf
│   ├── b7aaad981fe624c7ca40237c2a778f8c
│   │   └── main.tf
│   └── modules.json
└── plugins
    └── darwin_amd64
        ├── lock.json
        └── terraform-provider-azurerm_v1.3.2_x4
5 directories, 5 files
```

Resource Addressing

Resource Address:

<MODULE PATH><RESOURCE SPEC>

azurerm_resource_group.module

module.A.azurerm_network_interface.web

https://www.terraform.io/cli/state/resource-addressing

Resource Addressing

Module Path:

Addresses a module within a tree of modules.

module.A…

Can be multiple levels deep

module.A.module.B…

https://www.terraform.io/cli/state/resource-addressing

Resource Addressing

Resource Spec:

resource_type.resource_name[N]

resource_type - Type of the resource being addressed.

resource_name - User-defined name of the resource.

[N] - where N is a 0-based index into a resource with multiple instances specified by the count meta-parameter.

https://www.terraform.io/cli/state/resource-addressing

Commands - cli

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

https://www.terraform.io/cli/commands

Commands - init

terraform init [options]

```
Initializing provider plugins...

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
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

https://www.terraform.io/cli/commands/init

Commands - plan

Usage: terraform plan [options] [plan file]

Options:

-destroy If set, a plan will be generated to destroy all resources managed by the given configuration and state.

-parallelism=n Limit the number of concurrent operations. Defaults to 10.

-target=resource Resource to target.

https://www.terraform.io/cli/commands/plan

Commands - plan

Options:

-out=path Write a plan file to the given path.

-var 'foo=bar' Set a variable in the Terraform configuration. This flag can be set multiple times.

-var-file=foo Set variables in the Terraform configuration from a file.

https://www.terraform.io/cli/commands/plan

Commands - apply

Usage: terraform apply [options] [plan file]

https://www.terraform.io/cli/commands/apply

Commands - destroy

Usage: terraform destroy [options] [DIR]
 or terraform plan -destroy

https://www.terraform.io/cli/commands/destroy

Commands - fmt

Rewrites all Terraform configuration files in the current working directory.

Usage: terraform fmt [options] [DIR]

Options:

-check=false	Check if the input is formatted. Exit status will be 0 if all input is properly formatted and non-zero otherwise.

https://www.terraform.io/cli/commands/fmt

Commands - validate

Usage: terraform validate [options] [dir]

Validate the terraform files in a directory. Validation includes a basic check of syntax as well as checking that all variables declared in the configuration are specified.

https://www.terraform.io/cli/commands/validate

Commands - refresh

Usage: terraform refresh [options] [dir]

Update the state file of your infrastructure with metadata that matches the physical resources they are tracking.

This will not modify your infrastructure, but it can modify your state file to update metadata. This metadata might cause new changes to occur when you generate a plan or call apply next.

https://www.terraform.io/cli/commands/refresh

Commands - output

Usage: terraform output [options] [NAME]

Options:

-module=name	If specified, returns the outputs for a specific module
-json	If specified, machine readable output will be printed in JSON format

https://www.terraform.io/cli/commands/output

Commands - console

Creates an interactive console for interacting with Terraform.

Useful for experimenting with interpolations
and interacting with Terraform state.

```
$ terraform console
> azurerm_resource_group.module.name
challenge03-rg
> azurerm_resource_group.module.count
1
> format("tag-%03d",azurerm_resource_group.module.count)
tag-001
> exit
```

https://www.terraform.io/cli/commands/console

Commands - graph

Usage: terraform graph [options] [DIR]

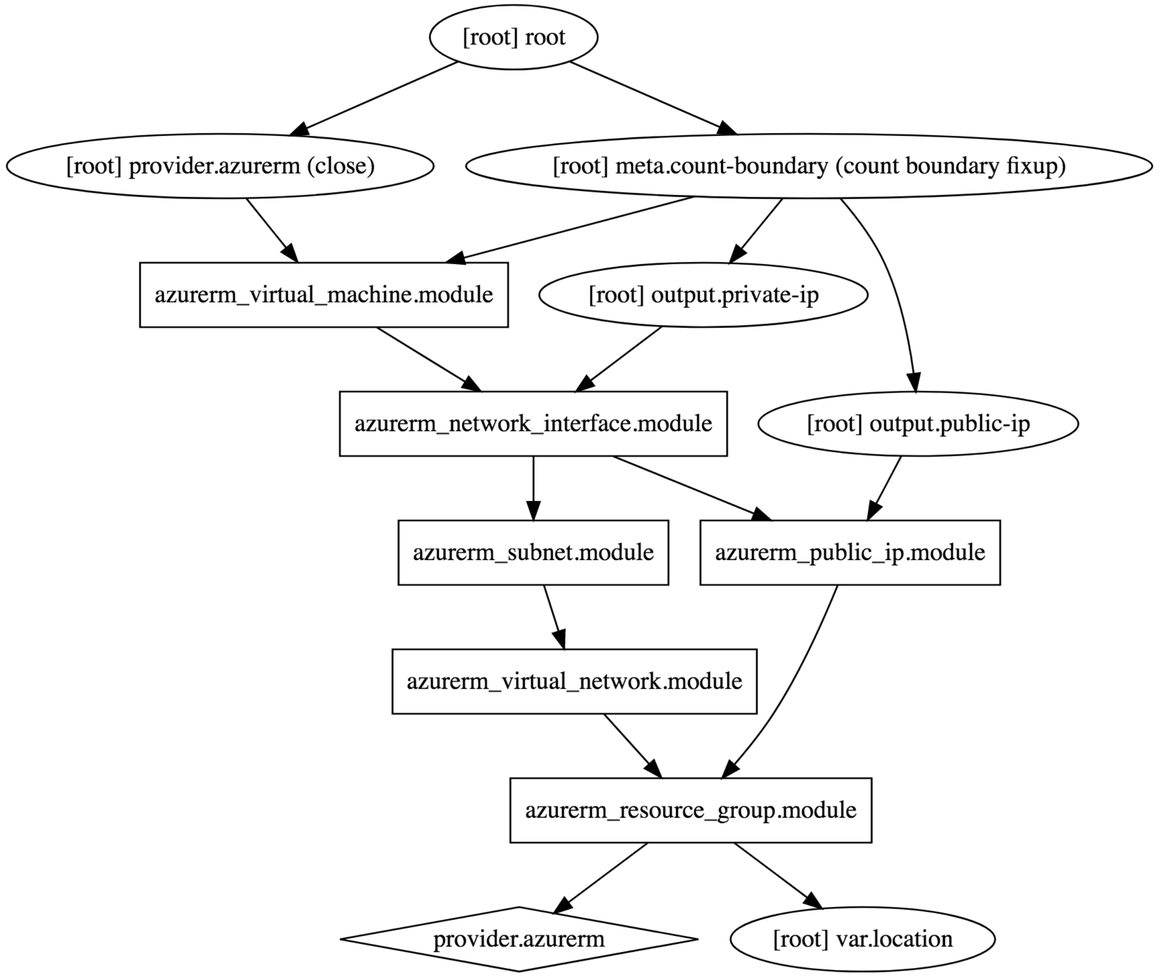
Outputs the visual execution graph of Terraform resources according to configuration files in DIR (or the current directory if omitted).

The graph is outputted in DOT format.

https://www.terraform.io/cli/commands/graph

Commands - graph

terraform graph |
dot -Tsvg > graph.svg



See also

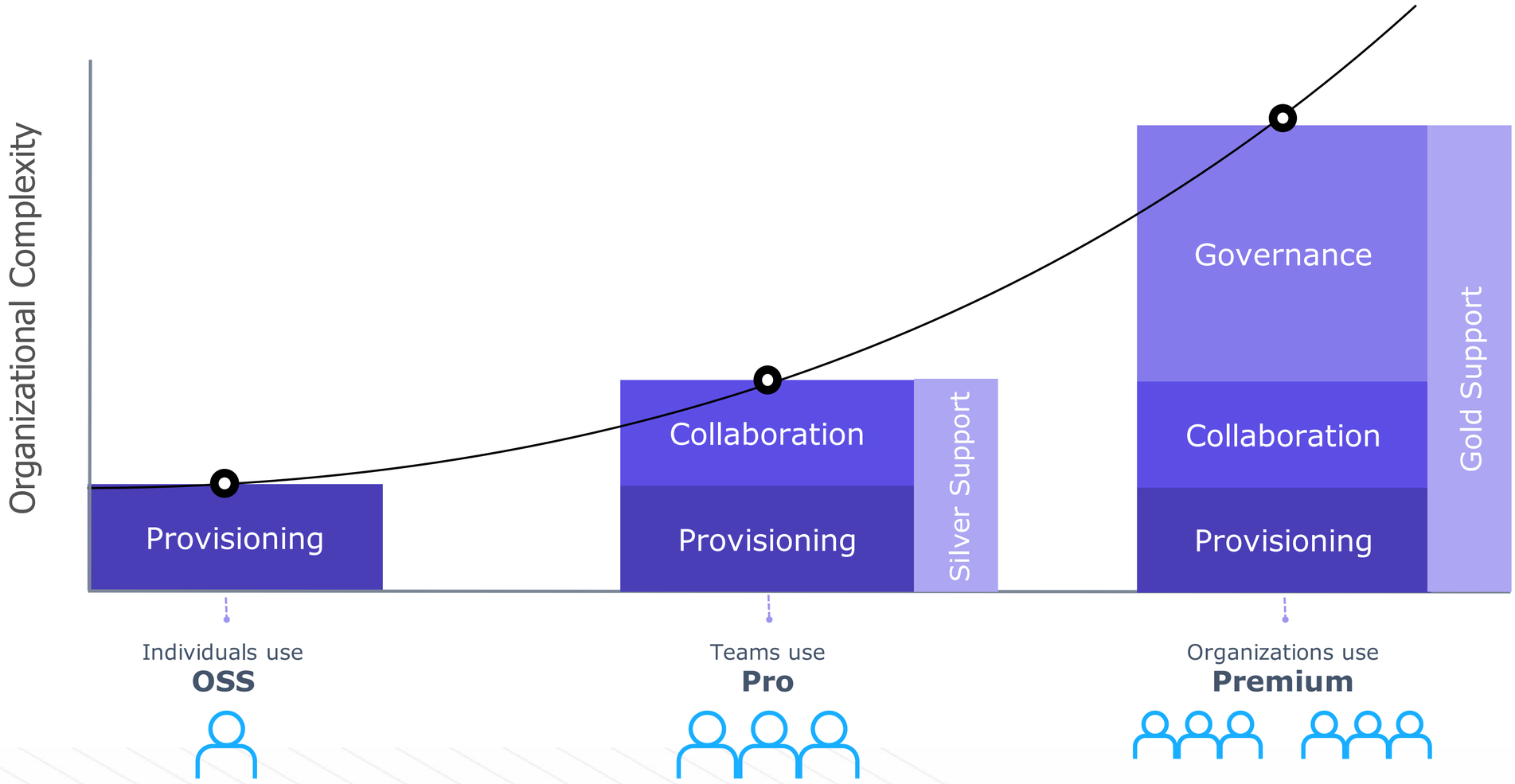
- 28mm/**blast-radius**
-

https://www.terraform.io/cli/commands/graph

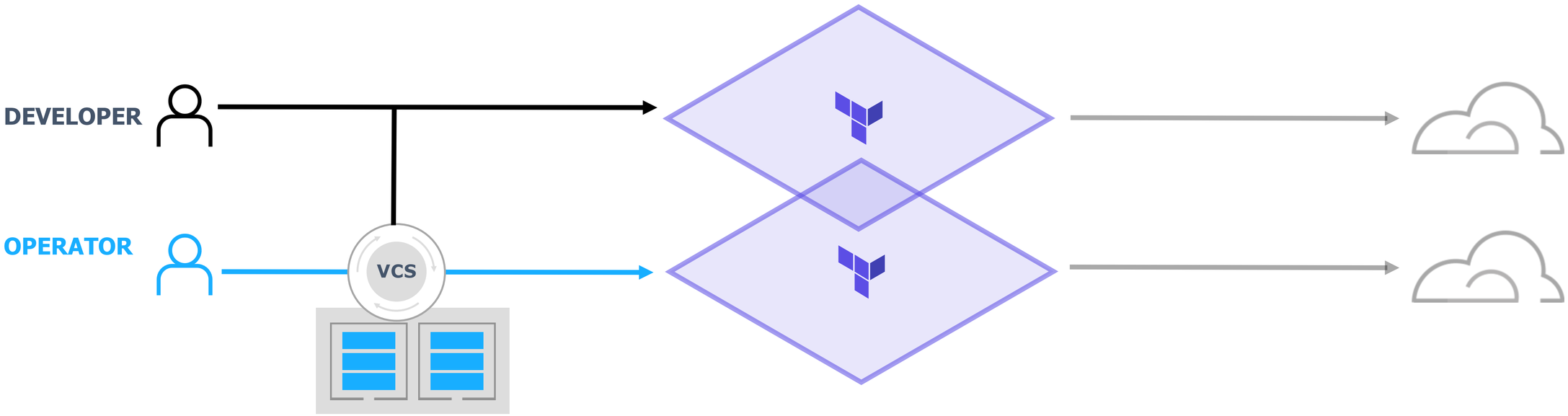
Terraform Enterprise

Terraform Open Source and Terraform Enterprise

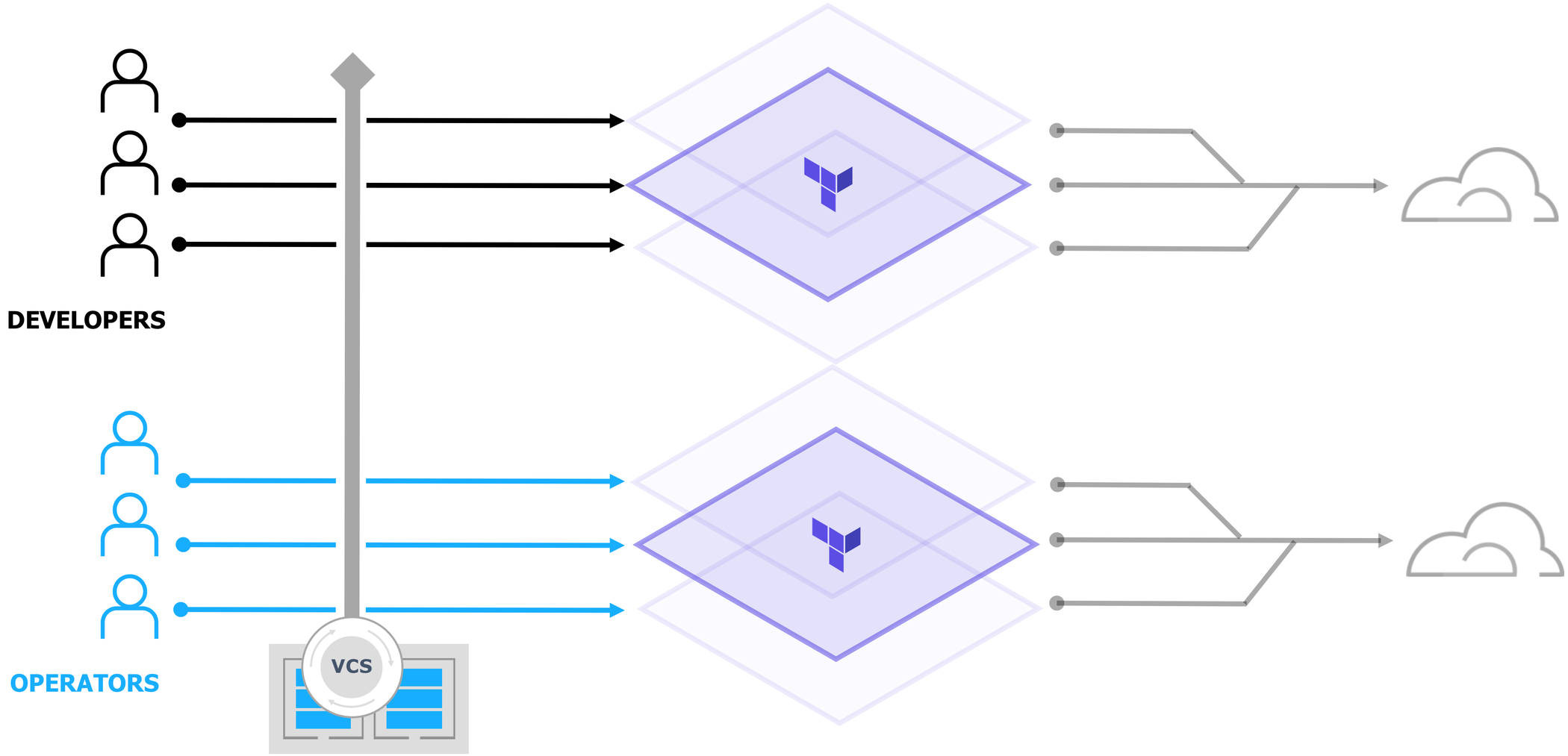
OSS for technical complexity & Enterprise for organizational complexity



Terraform OSS workflow for individuals



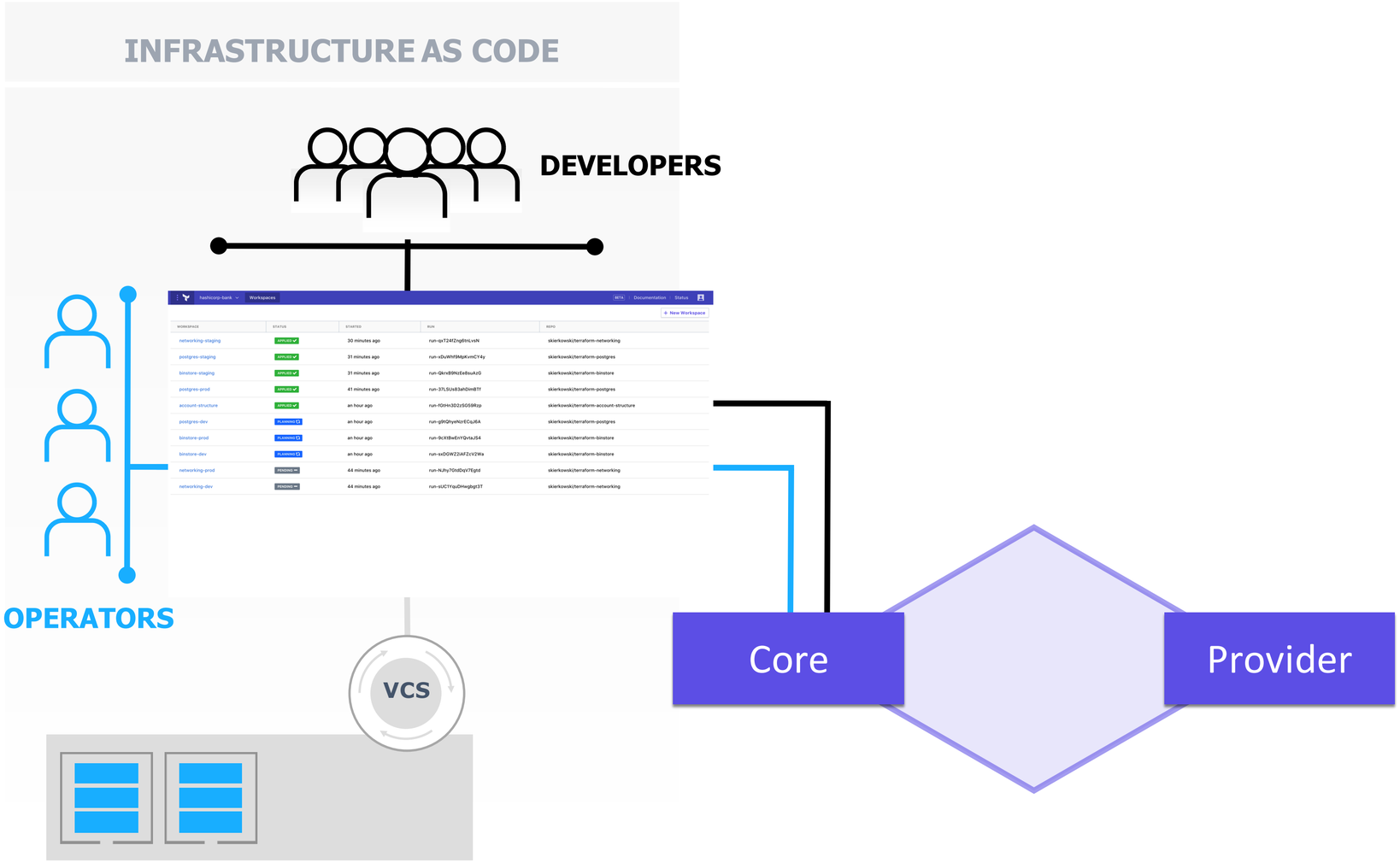
How can teams safely collaborate on infrastructure



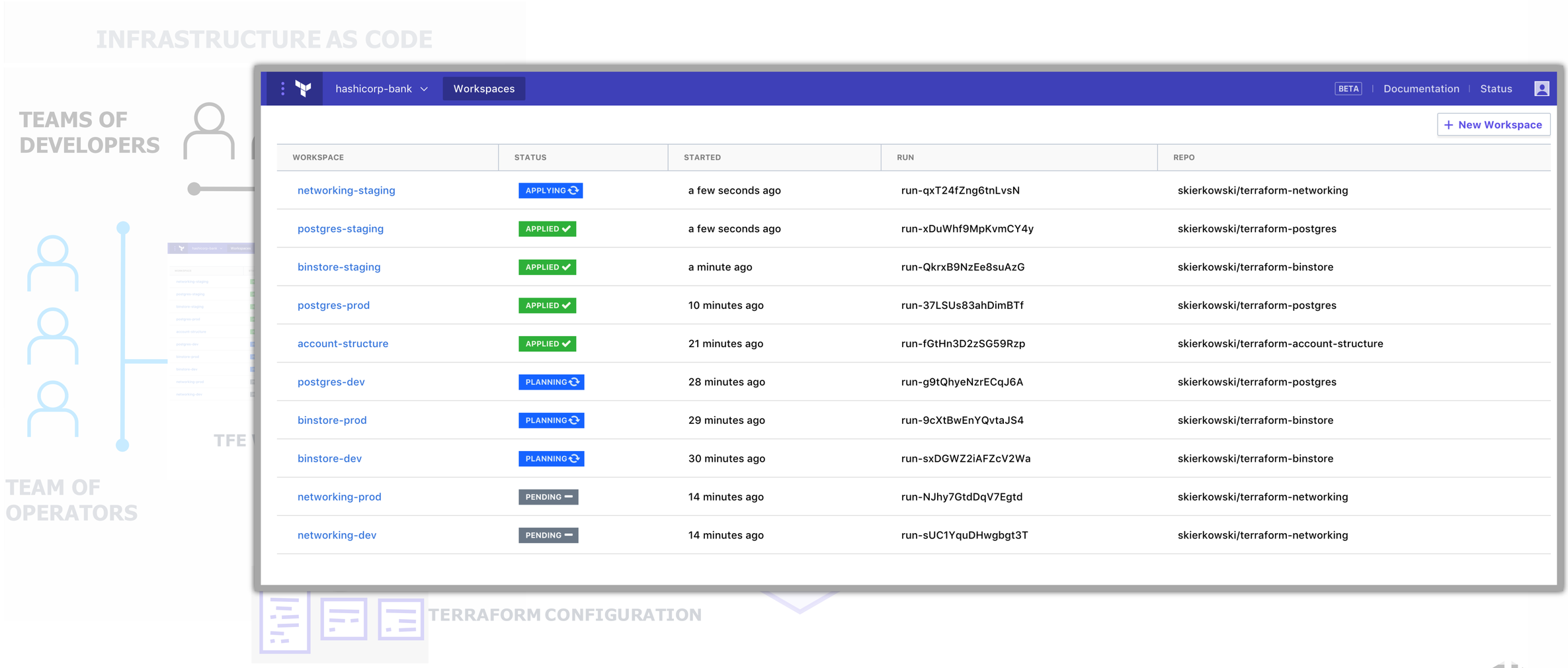
Demo

Workspaces

Workspace management and VCS connection



Workspace management and VCS connection



Workspace – VC Connection

REPOSITORY

e.g. organization/repository-name

The repository identifier in the format username/repository. Only the most recently updated repositories will appear with autocomplete; however, all repositories are available for use.

Hide additional options

TERRAFORM WORKING DIRECTORY

The directory that Terraform will execute within. This defaults to the root of your repository and is typically set to a subdirectory matching the environment when multiple environments exist within the same repository.

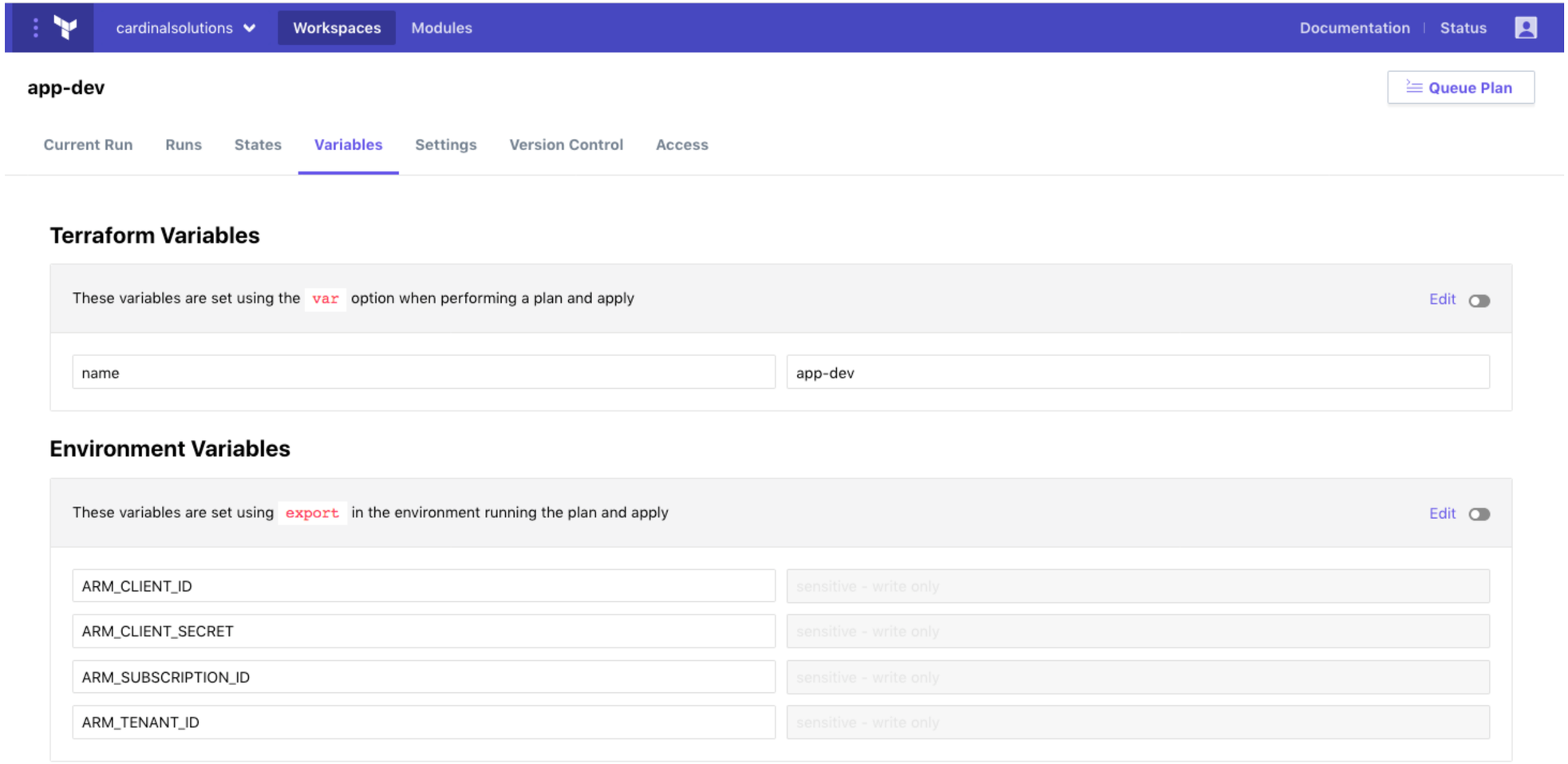
VCS BRANCH

(default branch)

The branch from which to import new versions. This defaults to the value your version control provides as the default branch for this repository.

https://www.terraform.io/cloud-docs/vcs

Workspace – Variables



https://www.terraform.io/cloud-docs/workspaces/variables

Workspace – Variables

https://www.terraform.io/cloud-docs/workspaces/variables

Workspace – Settings

Apply Method

- ○ **Auto apply**
  Automatically apply changes when a Terraform plan is successful. Plans that have no changes will not be applied. If this workspace is linked to version control, a push to the default branch of the linked repository will trigger a plan and apply.
- ◉ **Manual apply**
  Require an operator to confirm the result of the Terraform plan before applying. If this workspace is linked to version control, a push to the default branch of the linked repository will only trigger a plan and then wait for confirmation.

Terraform Version

1.1.4 ⌄

The version of Terraform to use for this workspace. Upon creating this workspace, the latest version was selected and will be used until it is changed manually. It will **not upgrade automatically**.

Terraform Working Directory

The directory that Terraform will execute within. This defaults to the root of your repository and is typically set to a subdirectory matching the environment when multiple environments exist within the same repository.

Remote state sharing

Choose whether this workspace should share state with the entire organization, or only with specific approved workspaces. The `terraform_remote_state` data source relies on state sharing to access workspace outputs.

- ◉ **Share with specific workspaces**

Select workspaces to share with ⌄

https://www.terraform.io/cloud-docs/workspaces/settings

Workspace – Settings

Destruction and Deletion

There are two independent steps for destroying this workspace and any infrastructure associated with it. First, any Terraform infrastructure should be destroyed. Second, the workspace in Terraform Cloud, including any variables, settings, and alert history can be deleted.

Destroy infrastructure

Allow destroy plans
When enabled, this setting allows a destroy plan to be created and applied. This also applies when using the CLI.

Manually destroy

Queuing a destroy plan will redirect to a new plan that will destroy all of the infrastructure managed by Terraform. It is equivalent to running `terraform plan -destroy -out=destroy.tfplan` followed by `terraform apply destroy.tfplan` locally.

Queue destroy plan

Delete Workspace

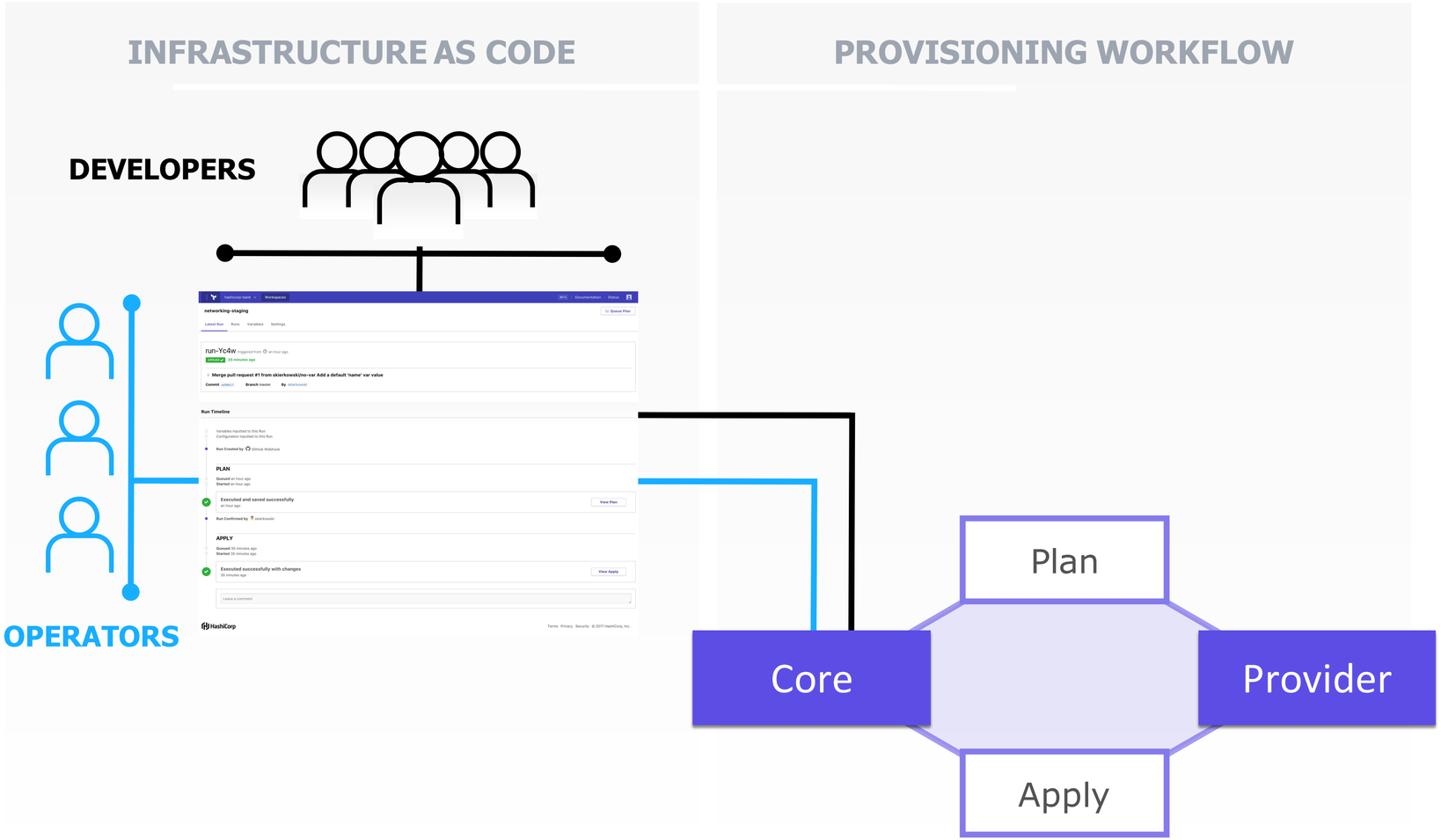
Deleting a workspace will remove any variables, settings, alert history, run history, and state related to it. This **will not** remove any infrastructure managed by this workspace. If needed, destroy the infrastructure prior to deleting the workspace.

Delete from Terraform Cloud

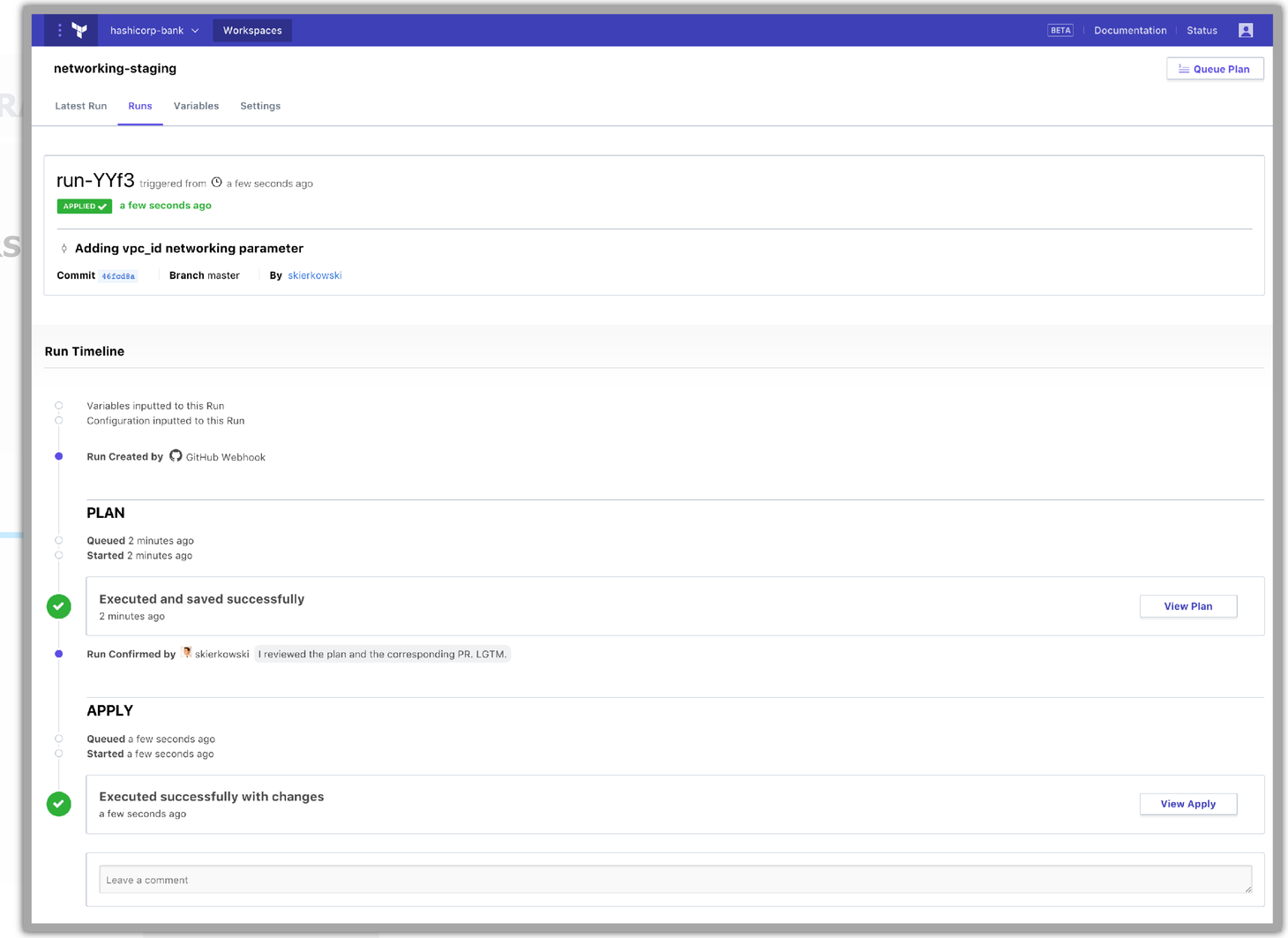
https://www.terraform.io/cloud-docs/workspaces/settings#destruction-and-deletion

Remote State in Cloud

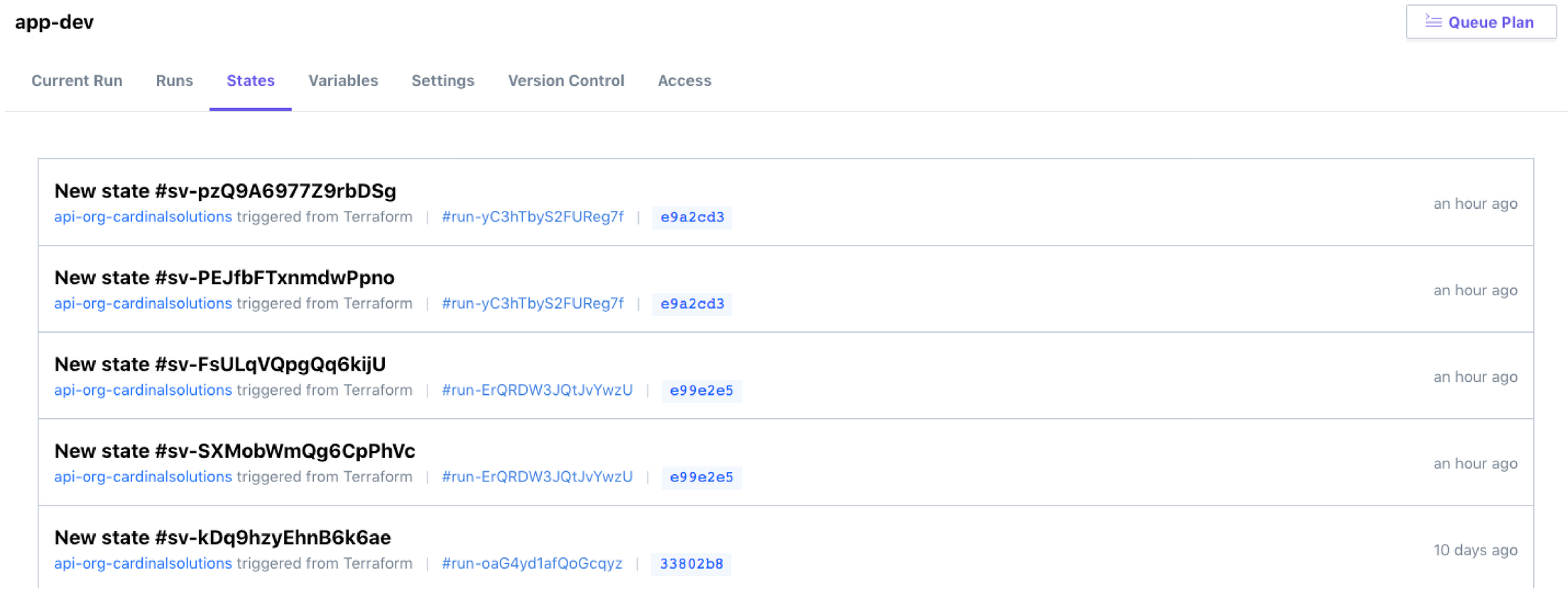
Remote Terraform runs and state management



Remote state management and Terraform runs

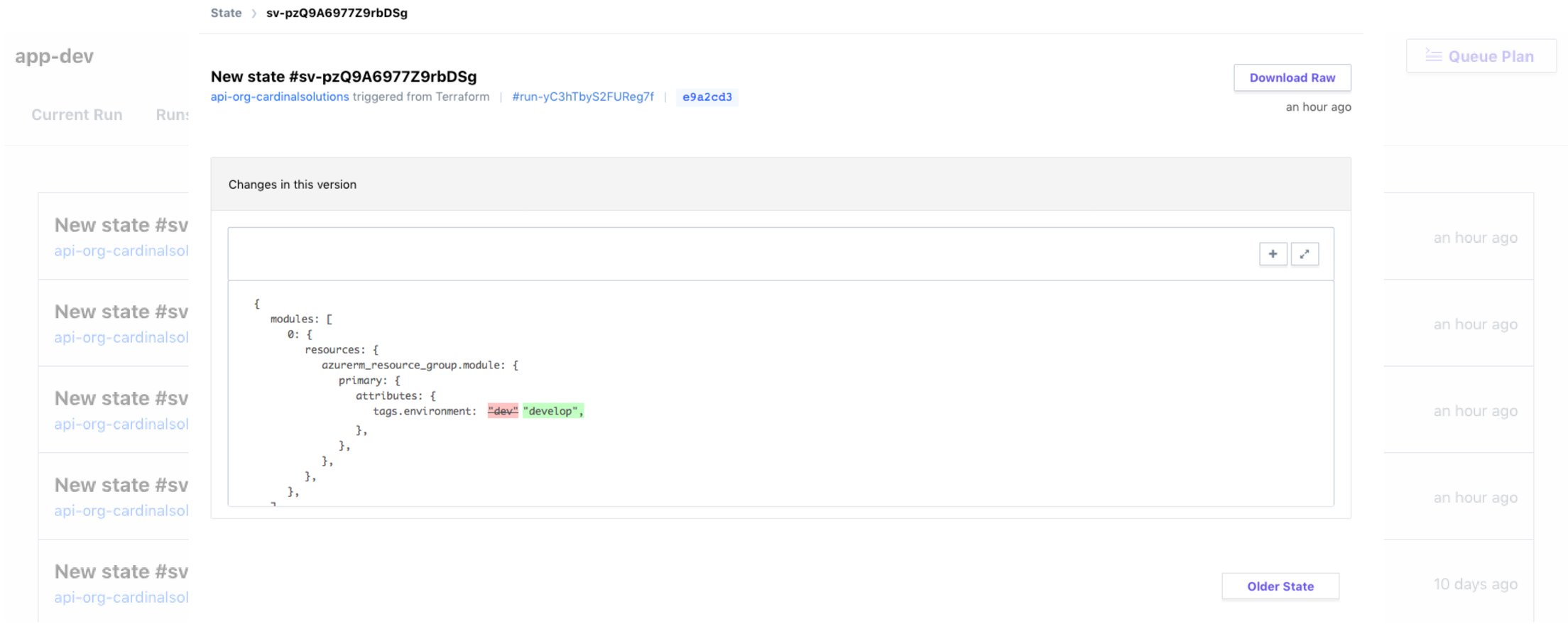


Workspace – State Management



https://www.terraform.io/cloud-docs/workspaces/state

Workspace – State Management



Collaboration

Collaboration

Role Based Access Controls

Permissions

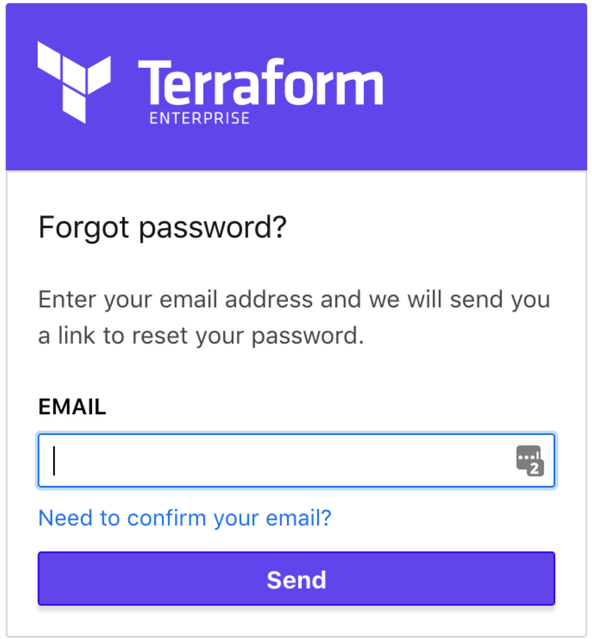
- Read
- Write
- Admin

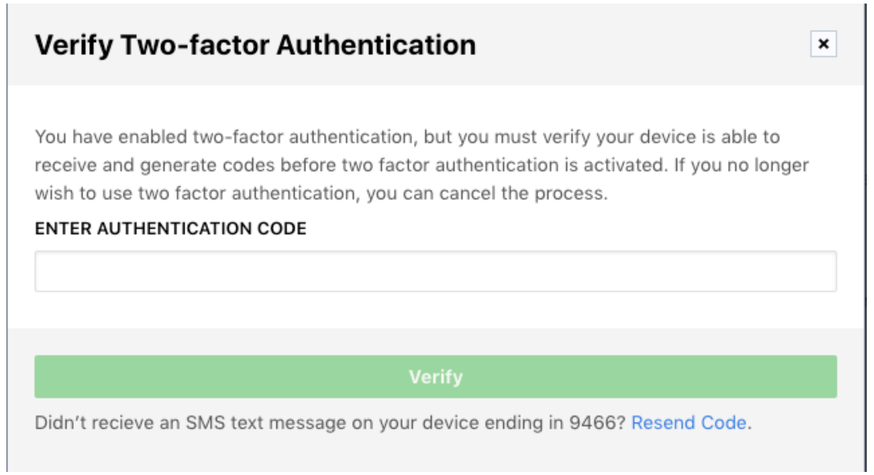
Service Accounts

API Tokens

- User
- Team
- Organization

User Authentication





Private Module Registry

Private Module Registry

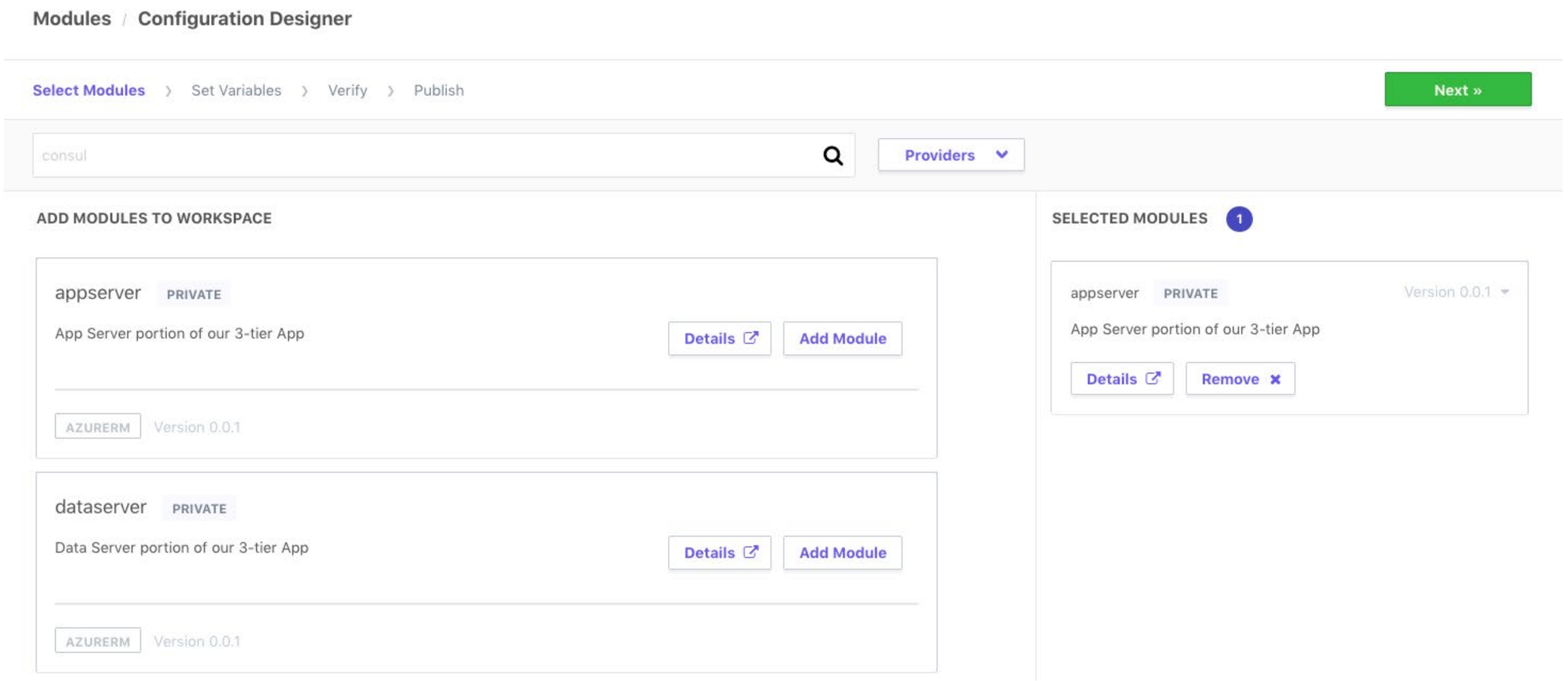
Modules for sharing across organization

Versioning

Private Module Registry

- Show Module discovery in private repo
- Documentation of modules
- Naming is different must start with <hostname>

Design Configuration



Governance

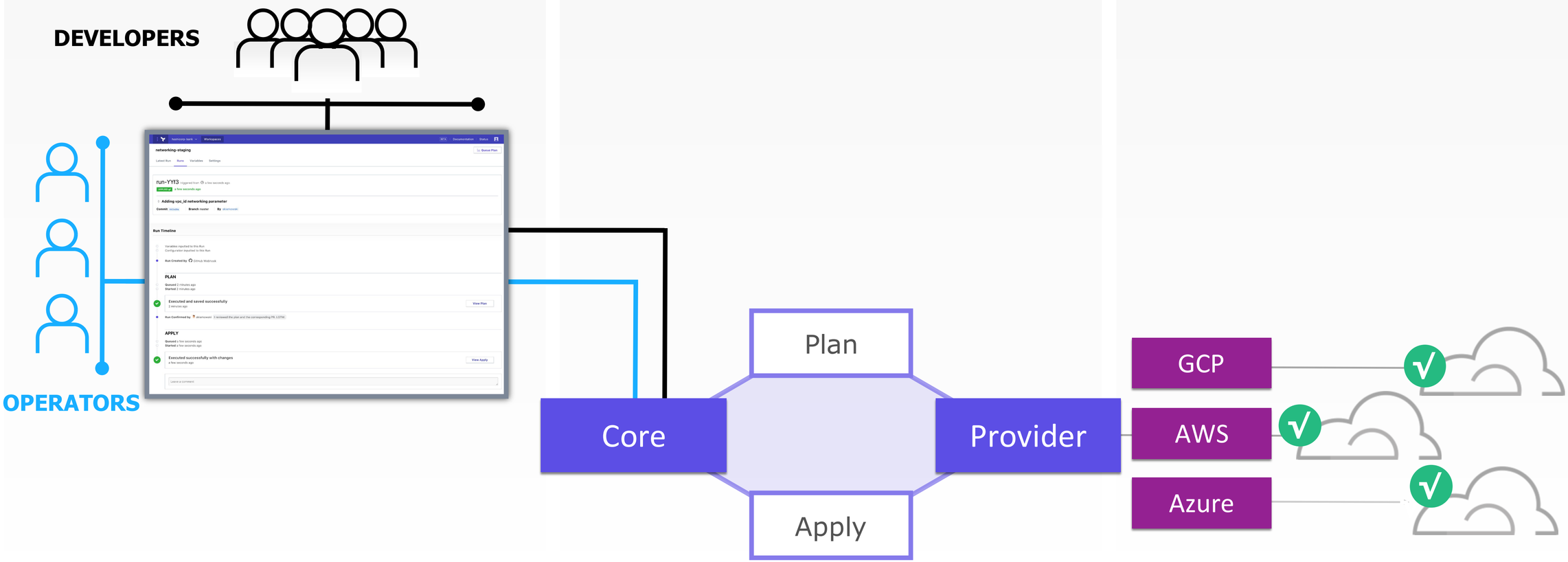
Enable Organization to have governance over infrastructure

Ensure provisioning policy is being maintained to minimize risk to the business

INFRASTRUCTURE AS CODE

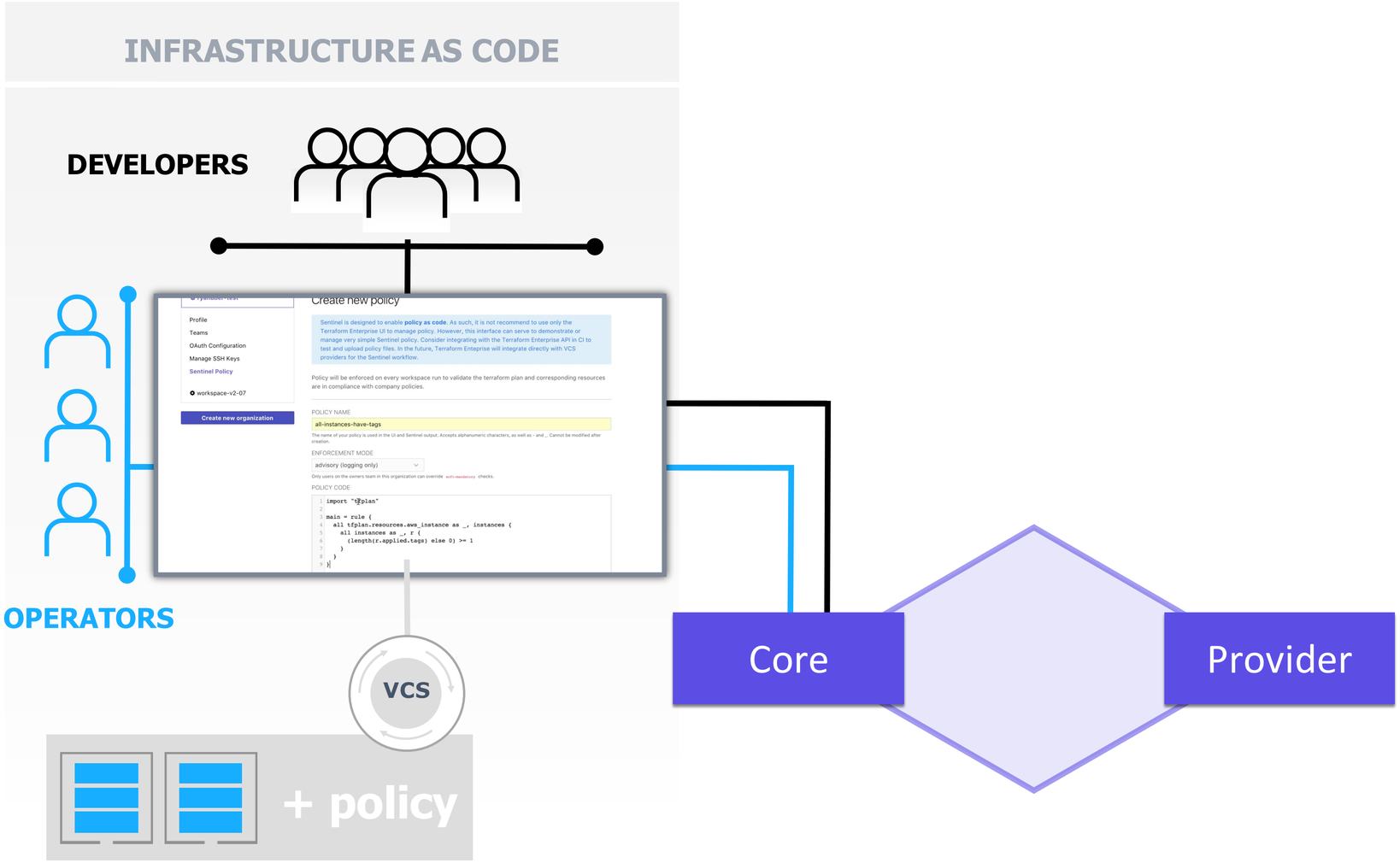
PROVISIONING WORKFLOW

EXTENSIBLE PROVIDER MODEL



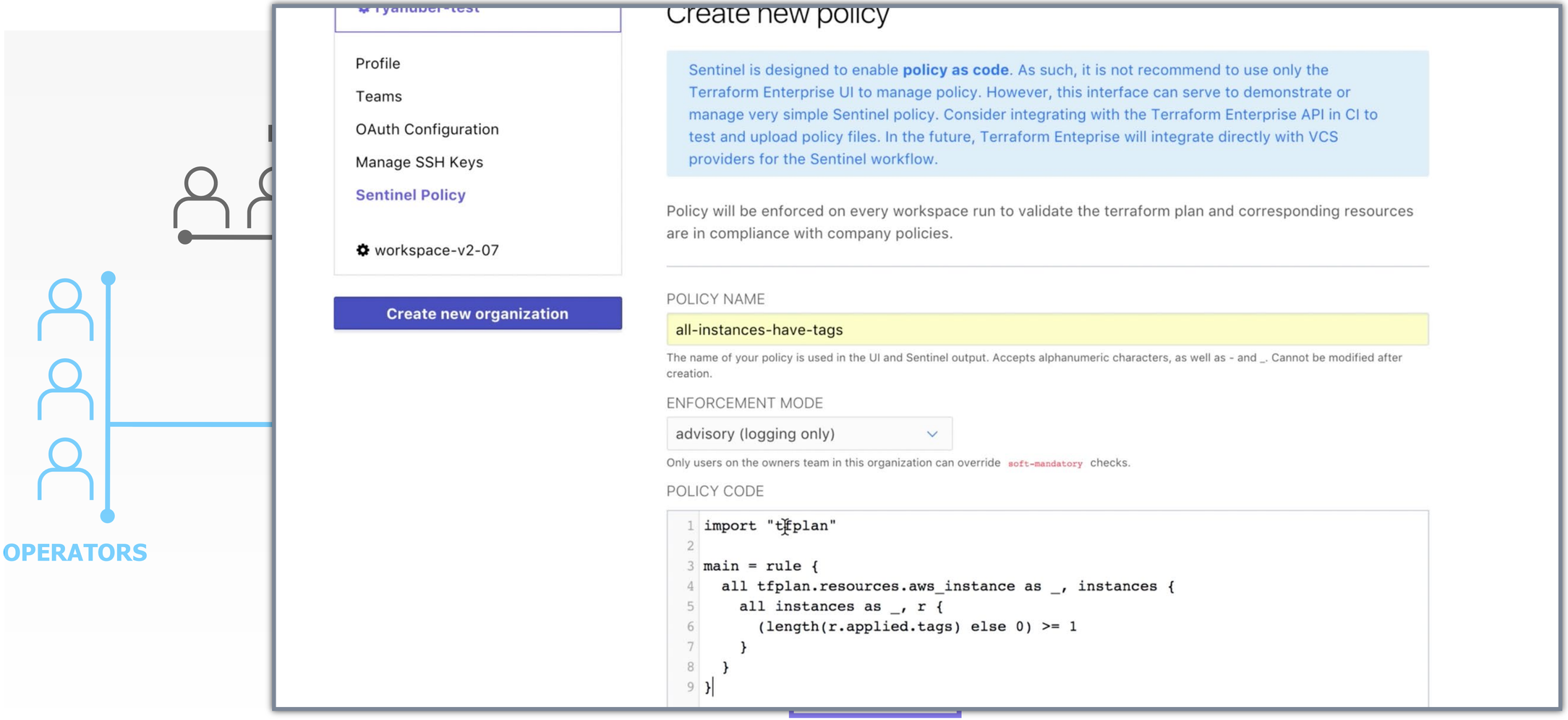
Enable Organization to have governance over infrastructure

Automate policy with Sentinel policy as code management



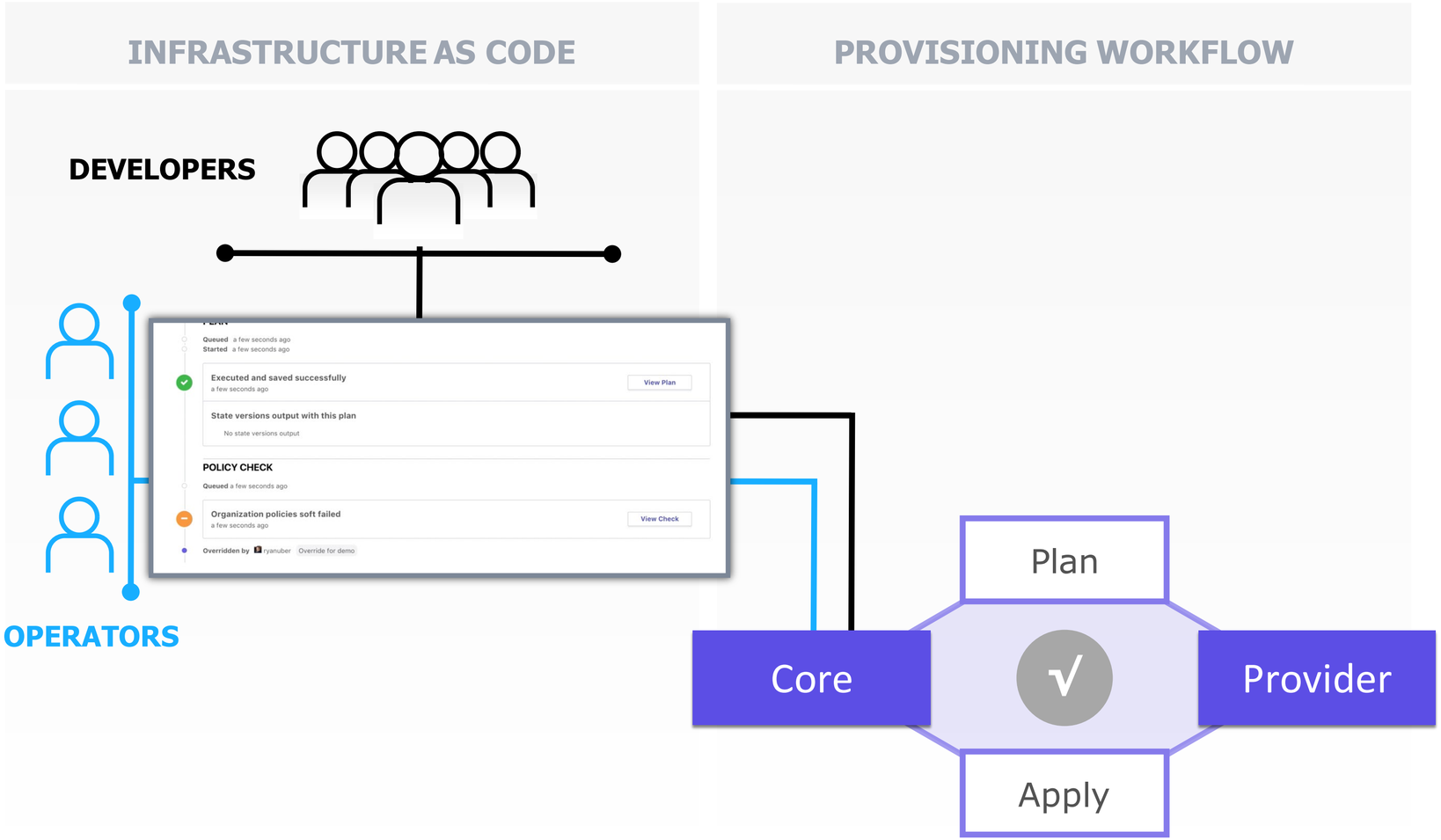
Enable Organization to have governance over infrastructure

Ensure provisioning policy is being maintained to minimize risk to the business



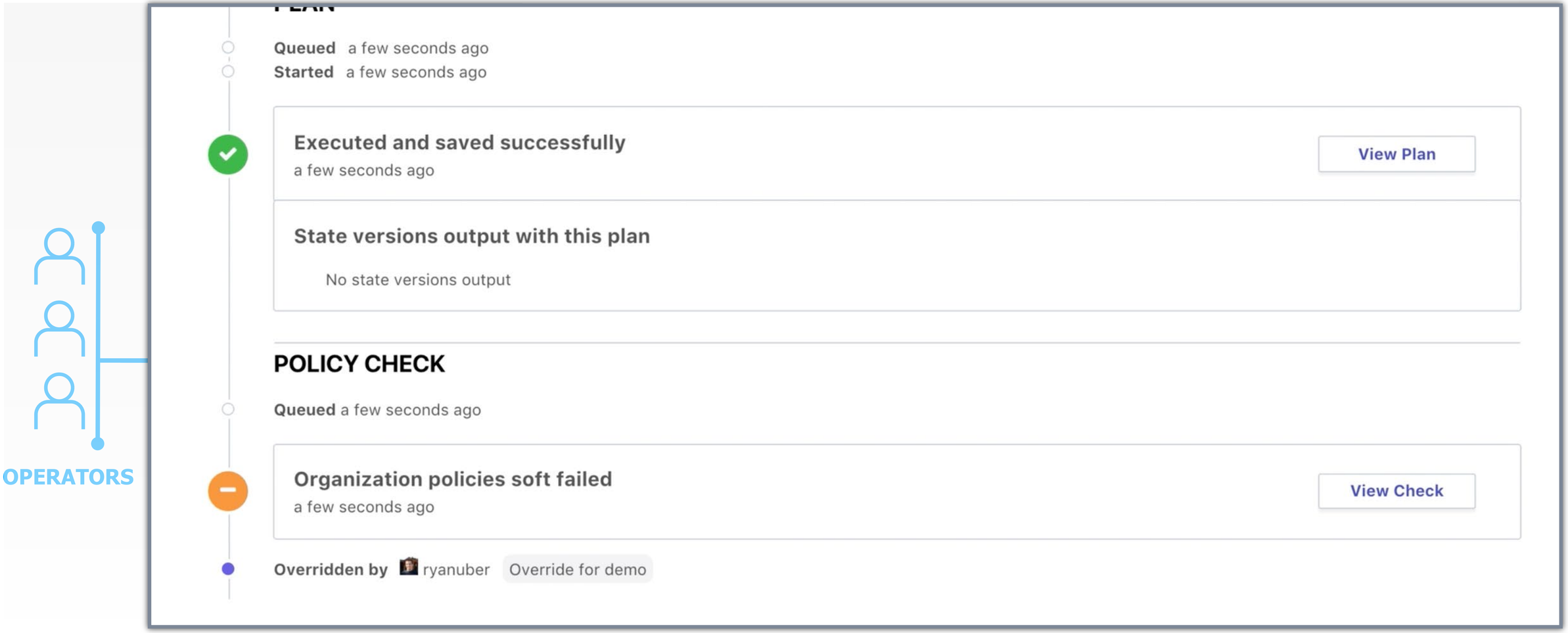
Enable Organization to have governance over infrastructure

Automate policy to be enforced on every Terraform run



Enable Organization to have governance over infrastructure

Ensure provisioning policy is being maintained to minimize risk to the business



Enable Organization to have governance over infrastructure

Automate policy to be enforced on every Terraform run