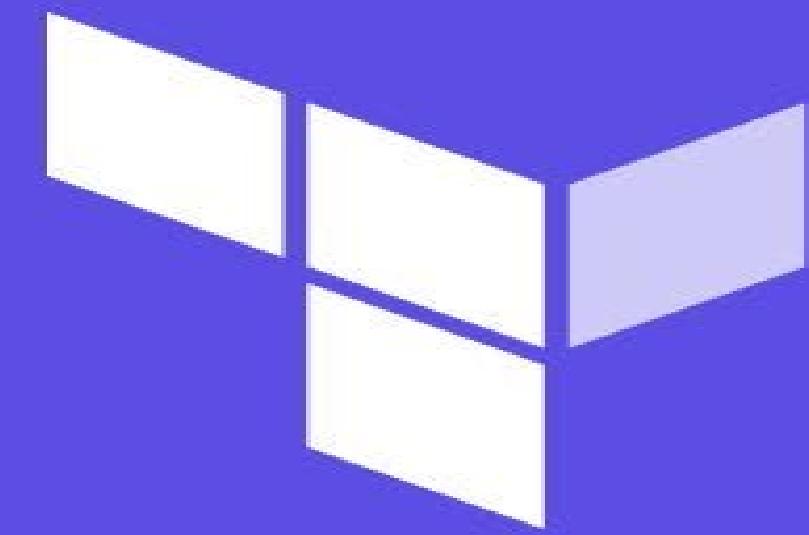




Rolls-Royce®



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

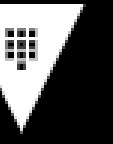


Company overview

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

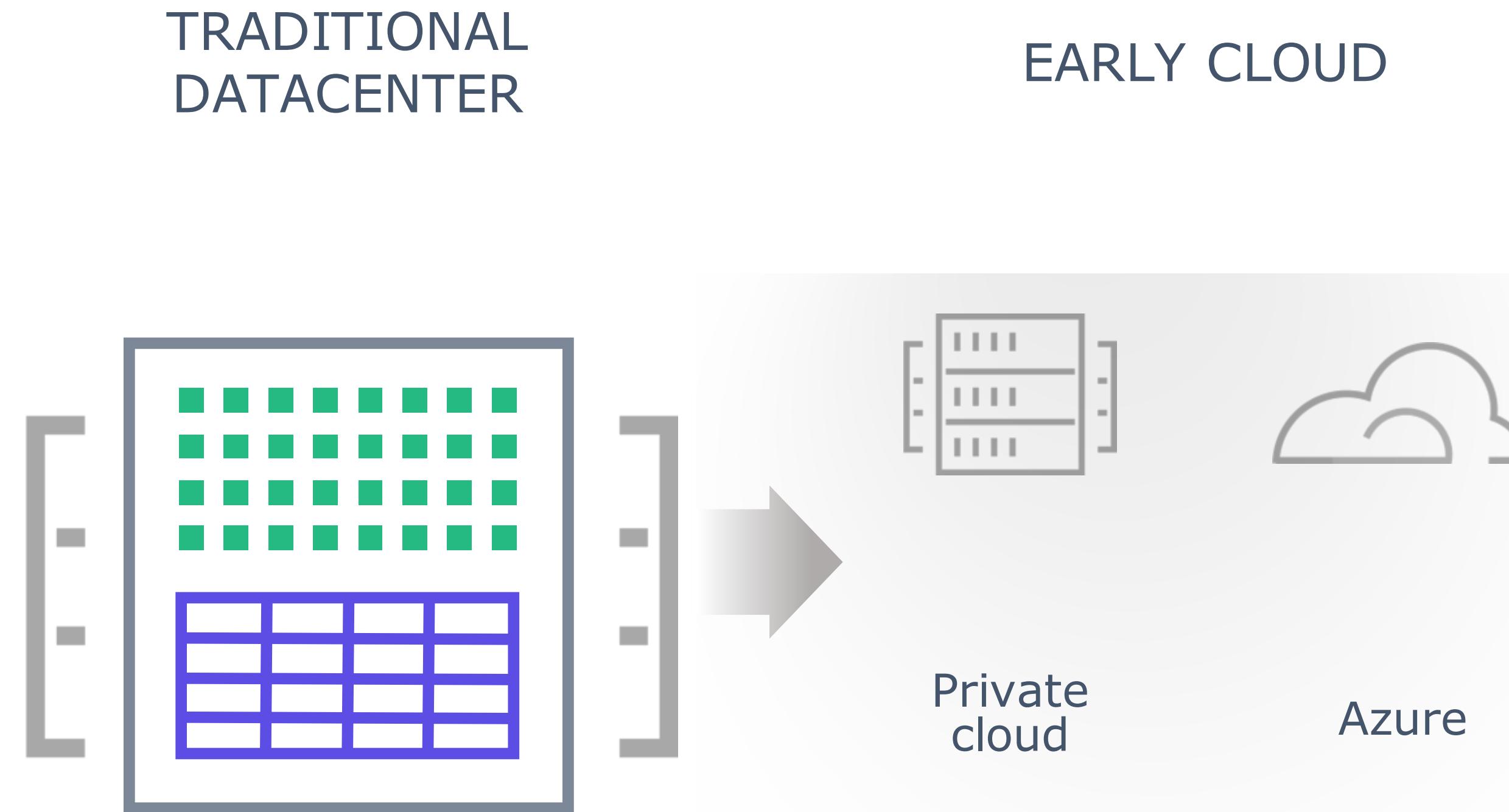
Infrastructure	Networking	Security	Application				
 Packer Automated machine images from a single source configuration	 Vagrant Single workflow to build and manage development environments	 Terraform Infrastructure automation to provision and manage any cloud service	 Consul Service mesh across any cloud and runtime platform	 Boundary Secure remote access to applications and critical systems	 Vault Secure management of secrets and sensitive data	 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

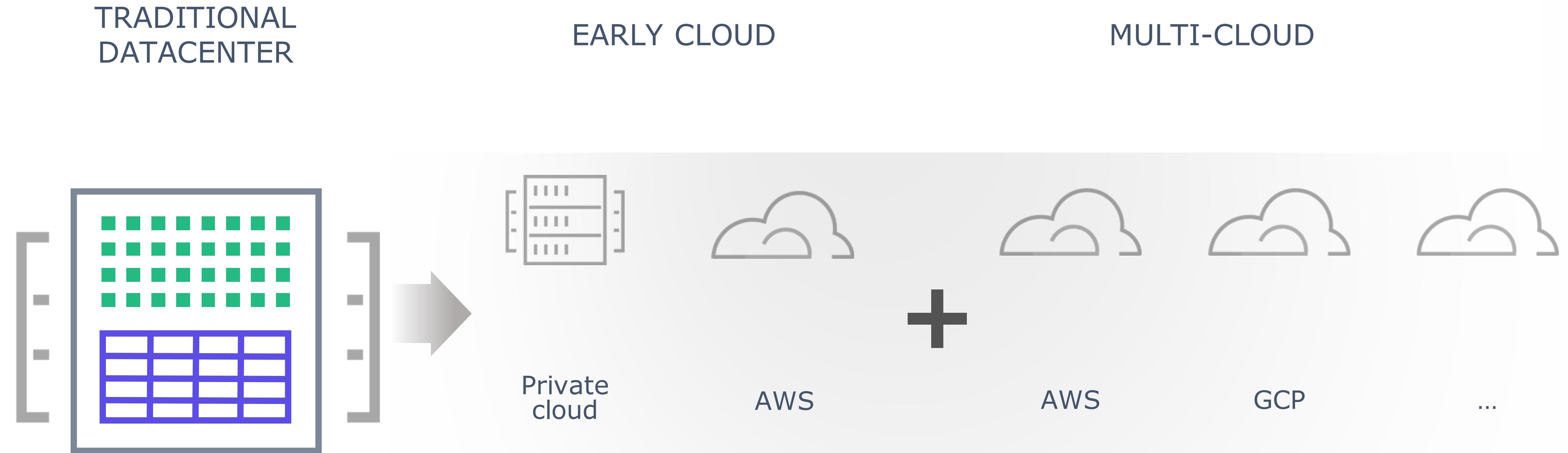
TRADITIONAL
DATACENTER



The shift to hybrid infrastructure



The shift to hybrid infrastructure



The 3 essential elements of infrastructure

DEVELOPMENT

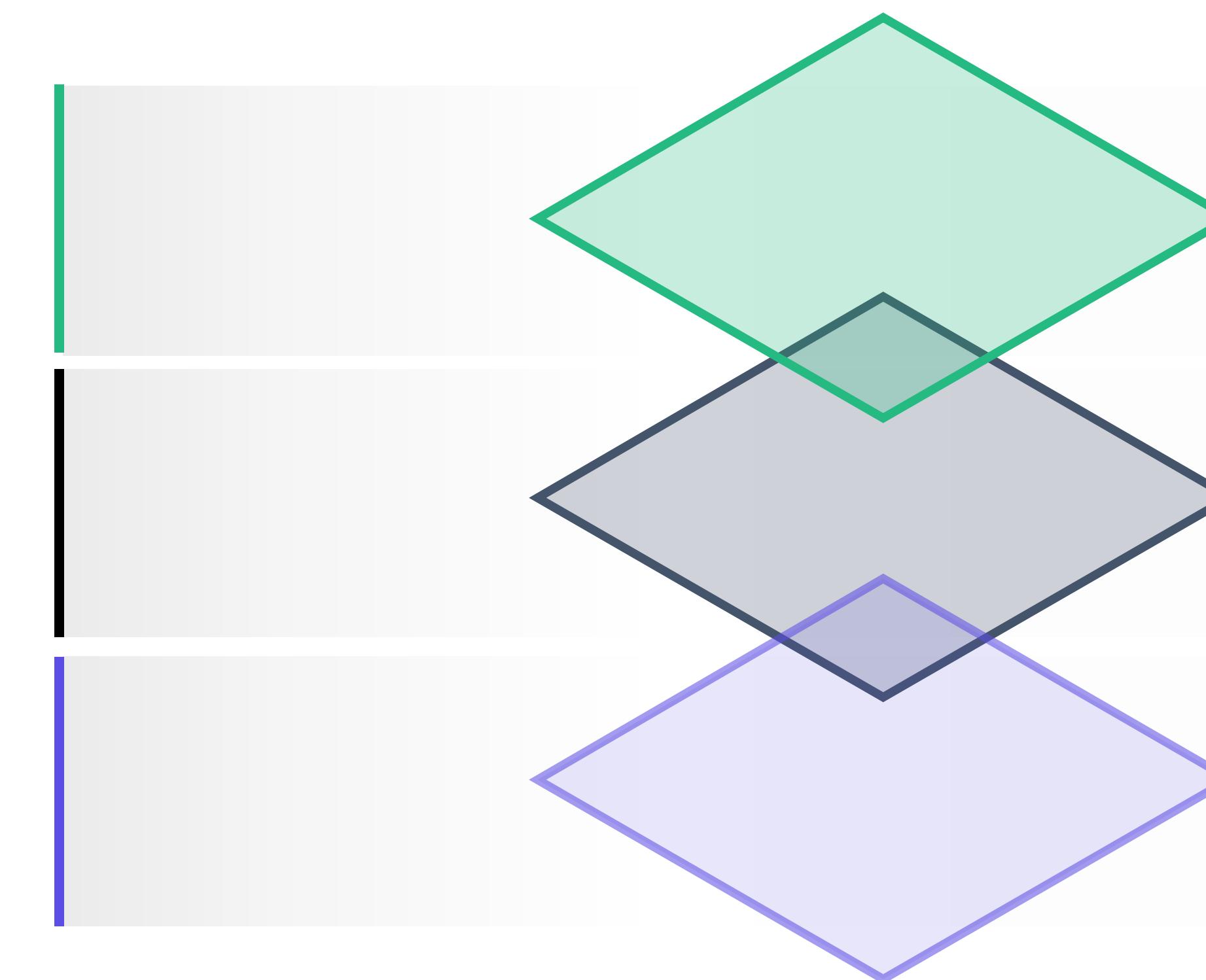
Run applications

SECURITY

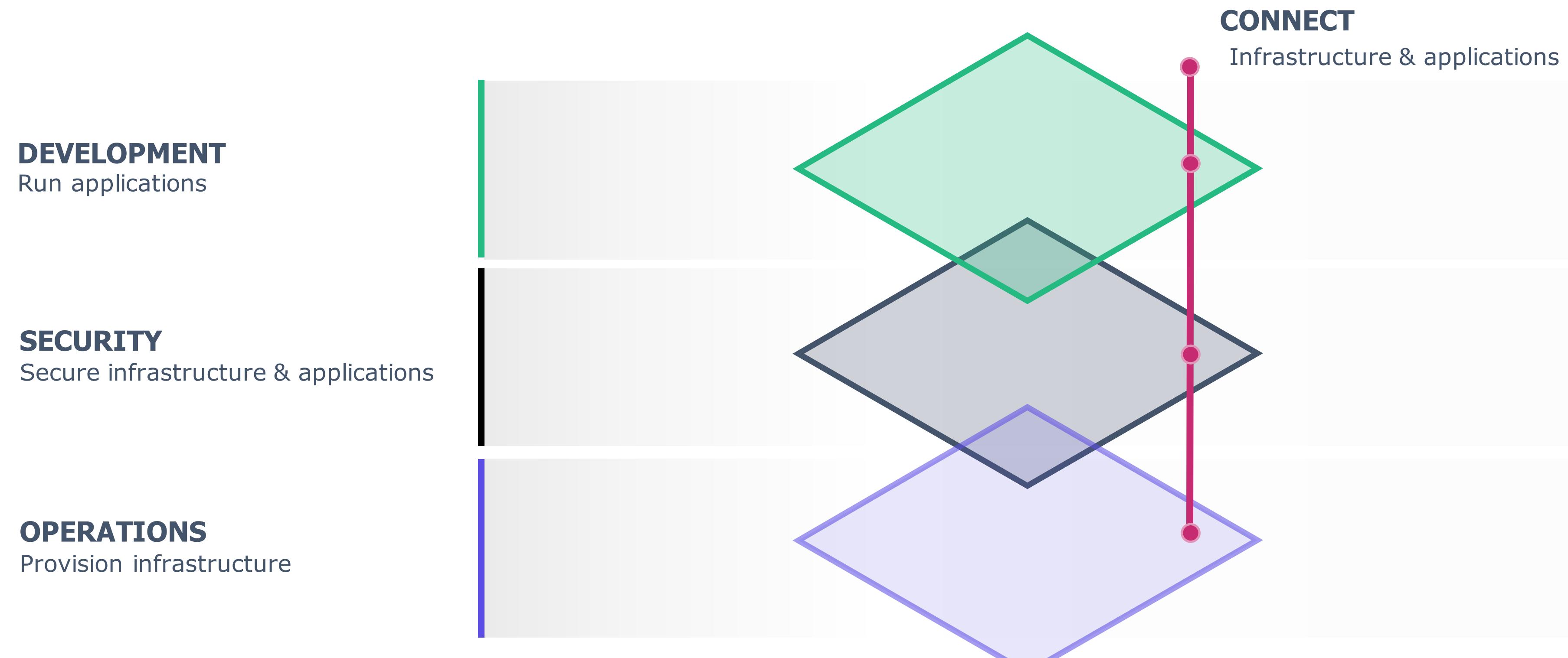
Secure infrastructure & applications

OPERATIONS

Provision infrastructure

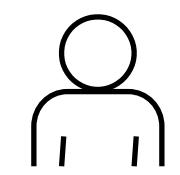


The 4 essential elements of distributed infrastructure



Addressing the challenges of cloud adoption

Cloud challenges for Operations teams



THE PRACTITIONER

DEVELOPMENT

Run applications

SECURITY

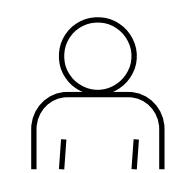
Secure infrastructure & applications

OPERATIONS

Provision infrastructure

- Scale
- Heterogeneity
- Dependency management

Provision any infrastructure for any application



THE PRACTITIONER

DEVELOPMENT

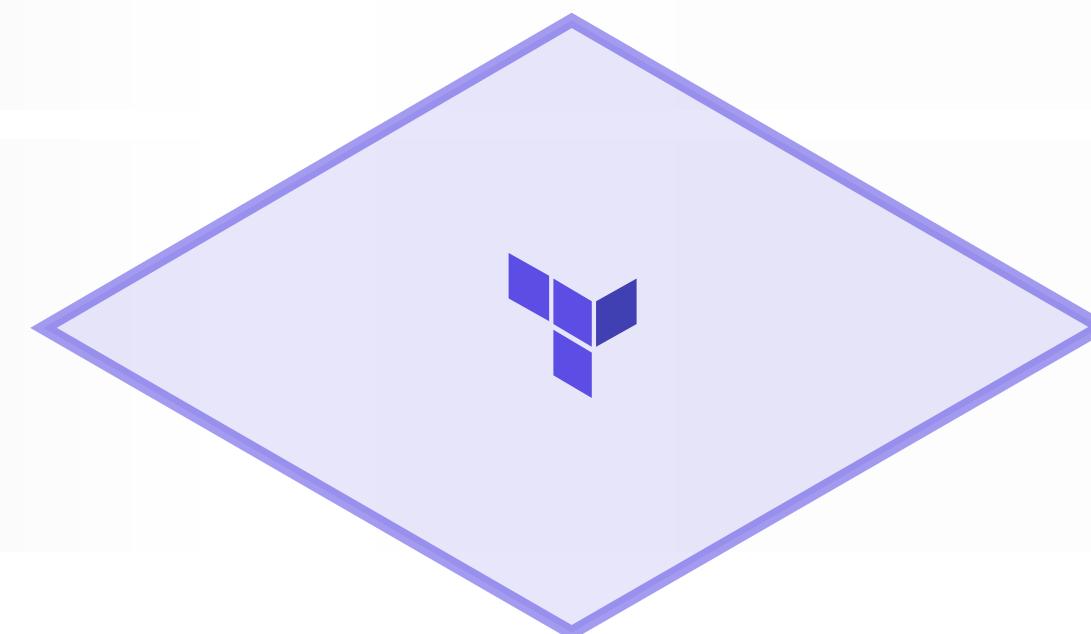
Run applications

SECURITY

Secure infrastructure & applications

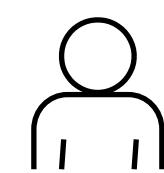
OPERATIONS

Provision infrastructure



- Scale
- Heterogeneity
- Dependency management

Cloud challenges for Security teams



THE PRACTITIONER

DEVELOPMENT

Run applications

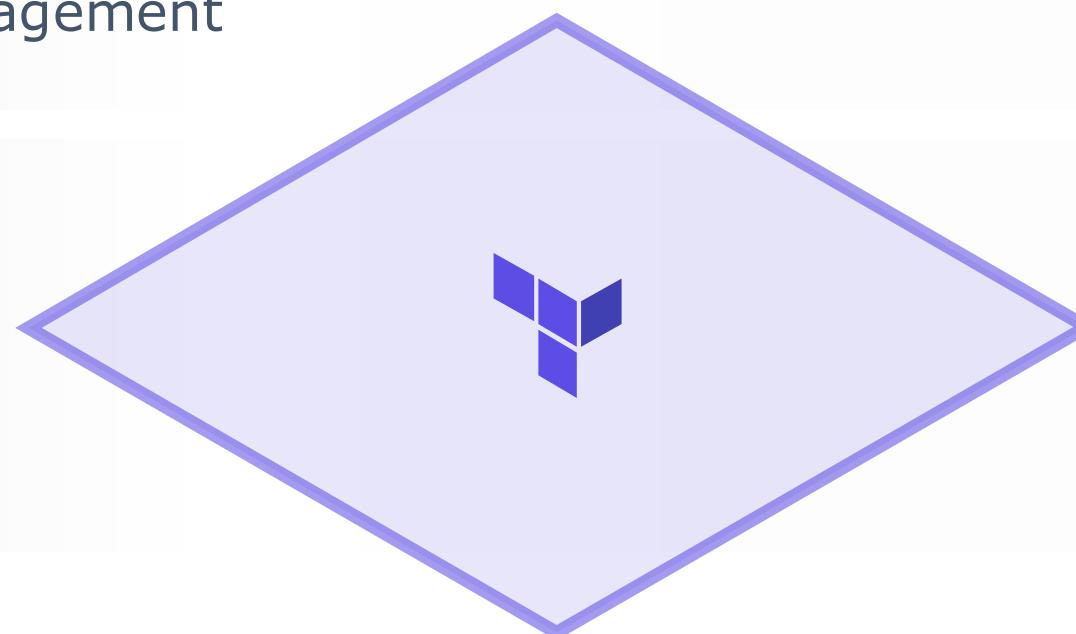
SECURITY

Secure infrastructure & applications

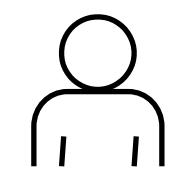
OPERATIONS

Provision infrastructure

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



Secure any application and any infrastructure



THE PRACTITIONER

DEVELOPMENT

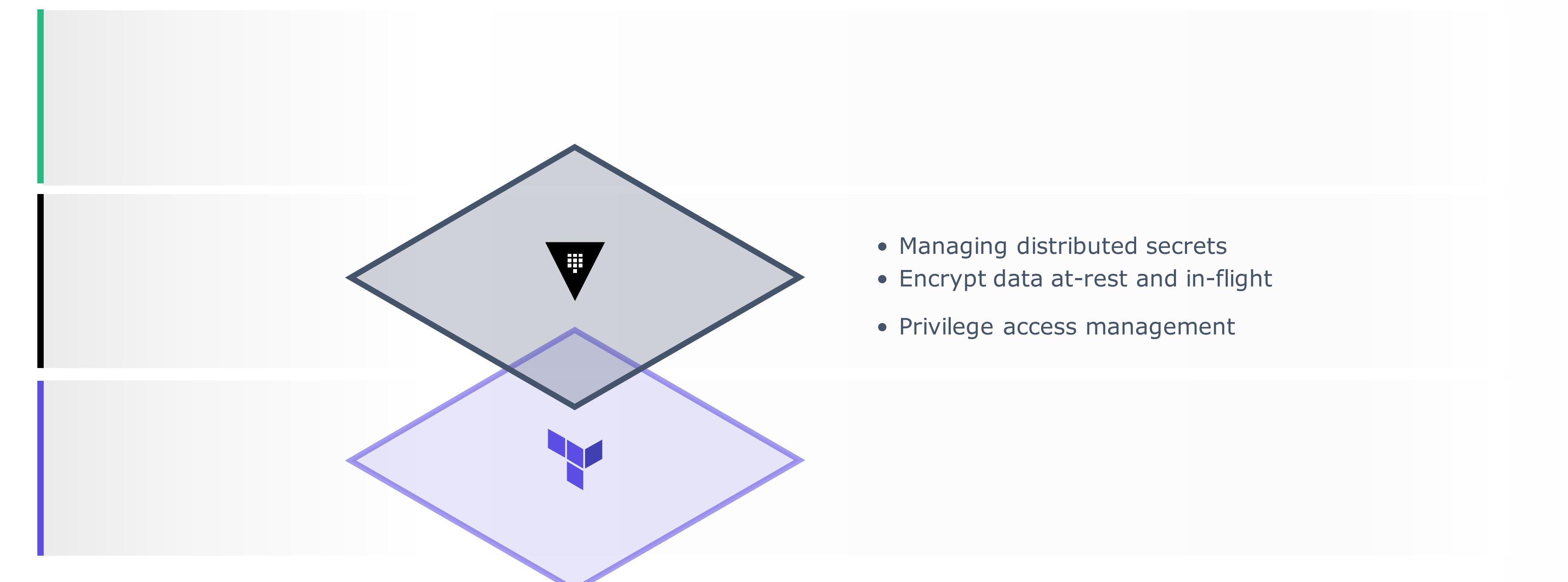
Run applications

SECURITY

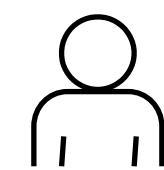
Secure infrastructure & applications

OPERATIONS

Provision infrastructure



Challenges for Developers / Operators



THE PRACTITIONER

DEVELOPMENT

Run applications

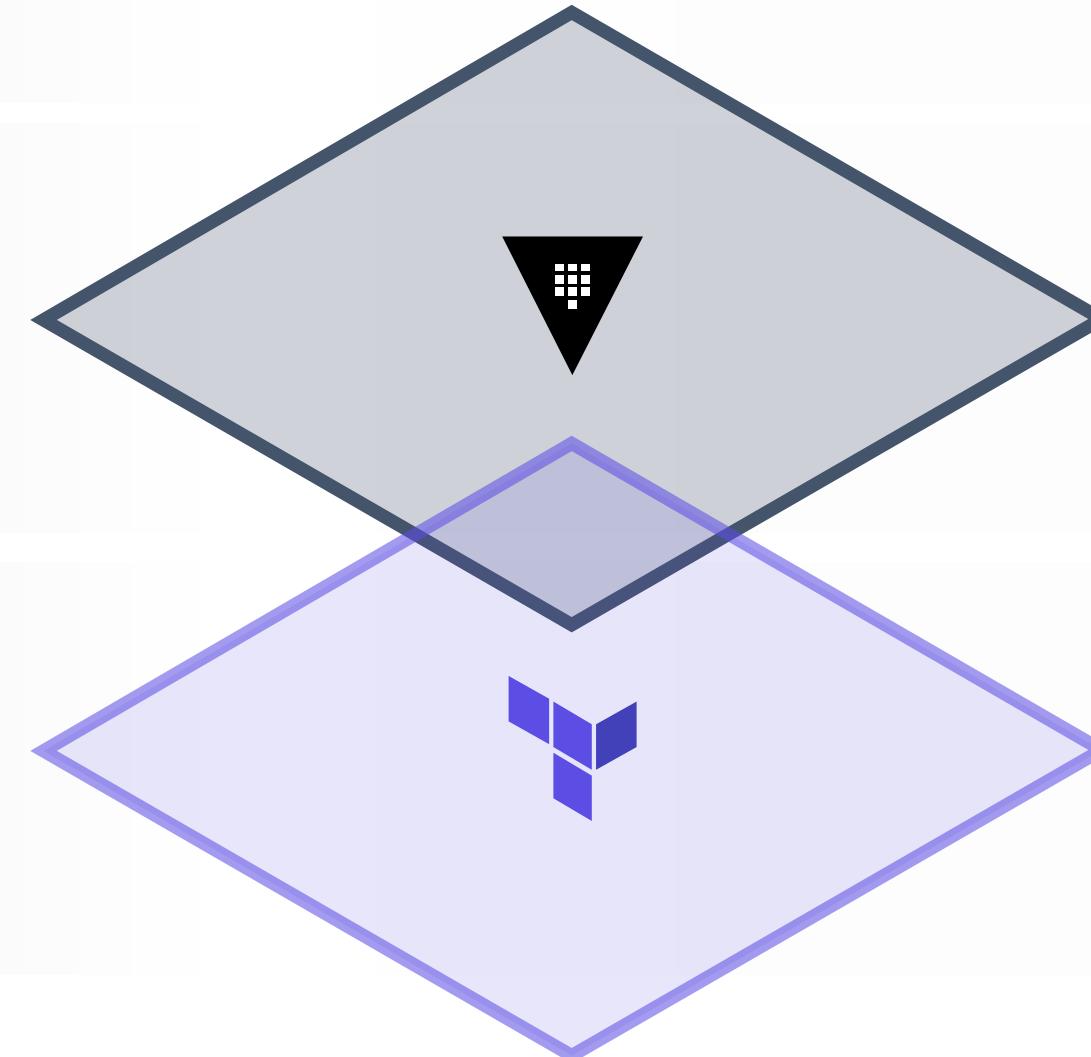
- Decoupling application from infrastructure
- Bin packing, machine efficiency

SECURITY

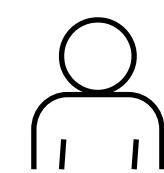
Secure infrastructure & applications

OPERATIONS

Provision infrastructure



Run any application on any infrastructure



THE PRACTITIONER

DEVELOPMENT

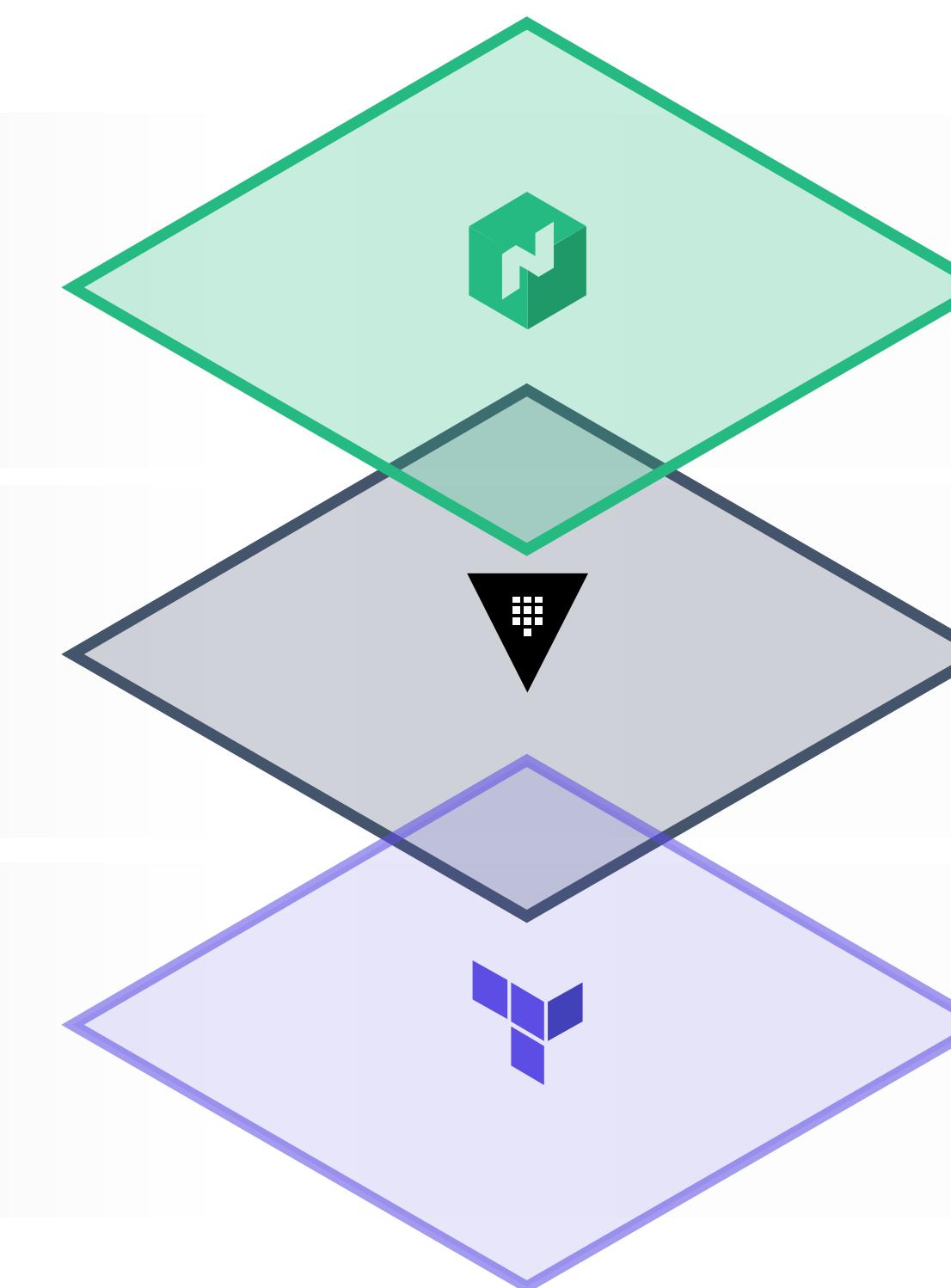
Run applications

SECURITY

Secure infrastructure & applications

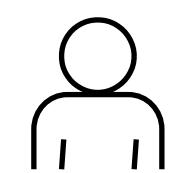
OPERATIONS

Provision infrastructure



- Decoupling application from infrastructure
- Bin packing, machine efficiency

Connect any application and any infrastructure



THE PRACTITIONER

DEVELOPMENT

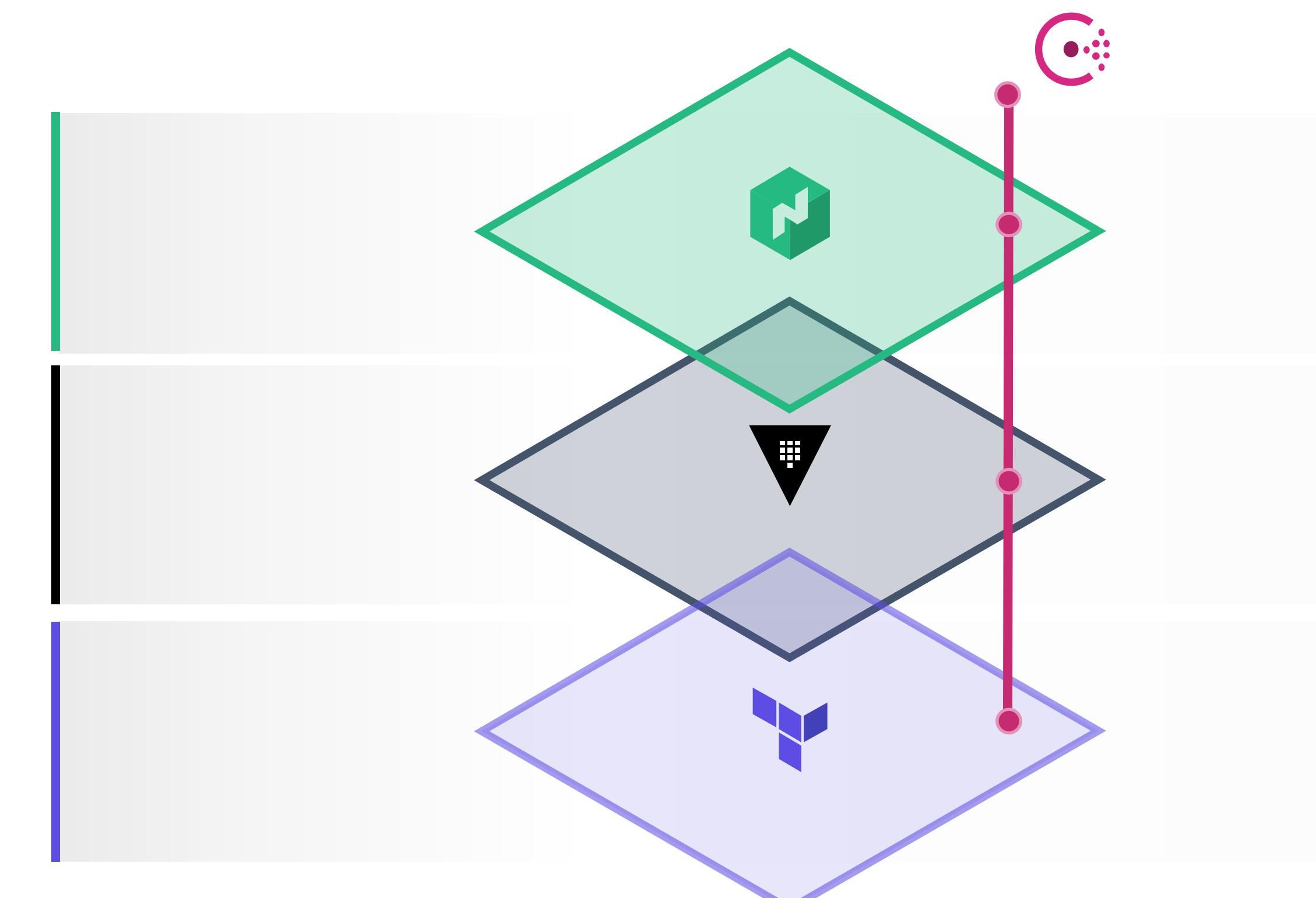
Run applications

SECURITY

Secure infrastructure & applications

OPERATIONS

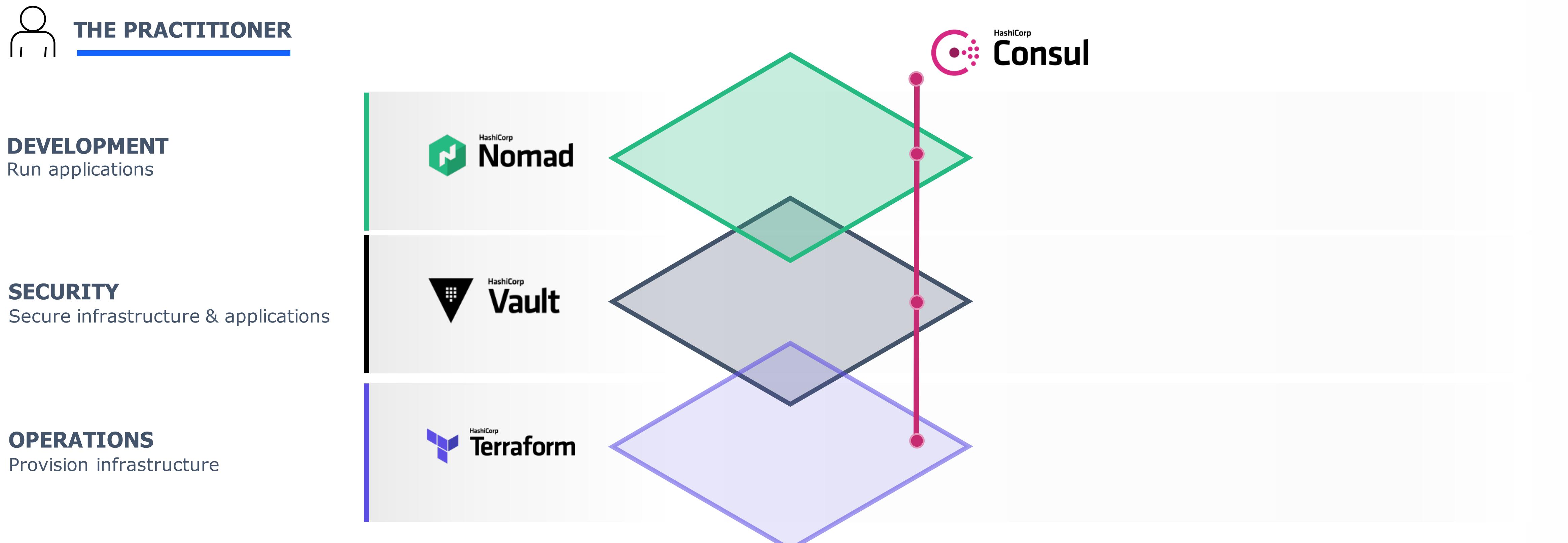
Provision infrastructure



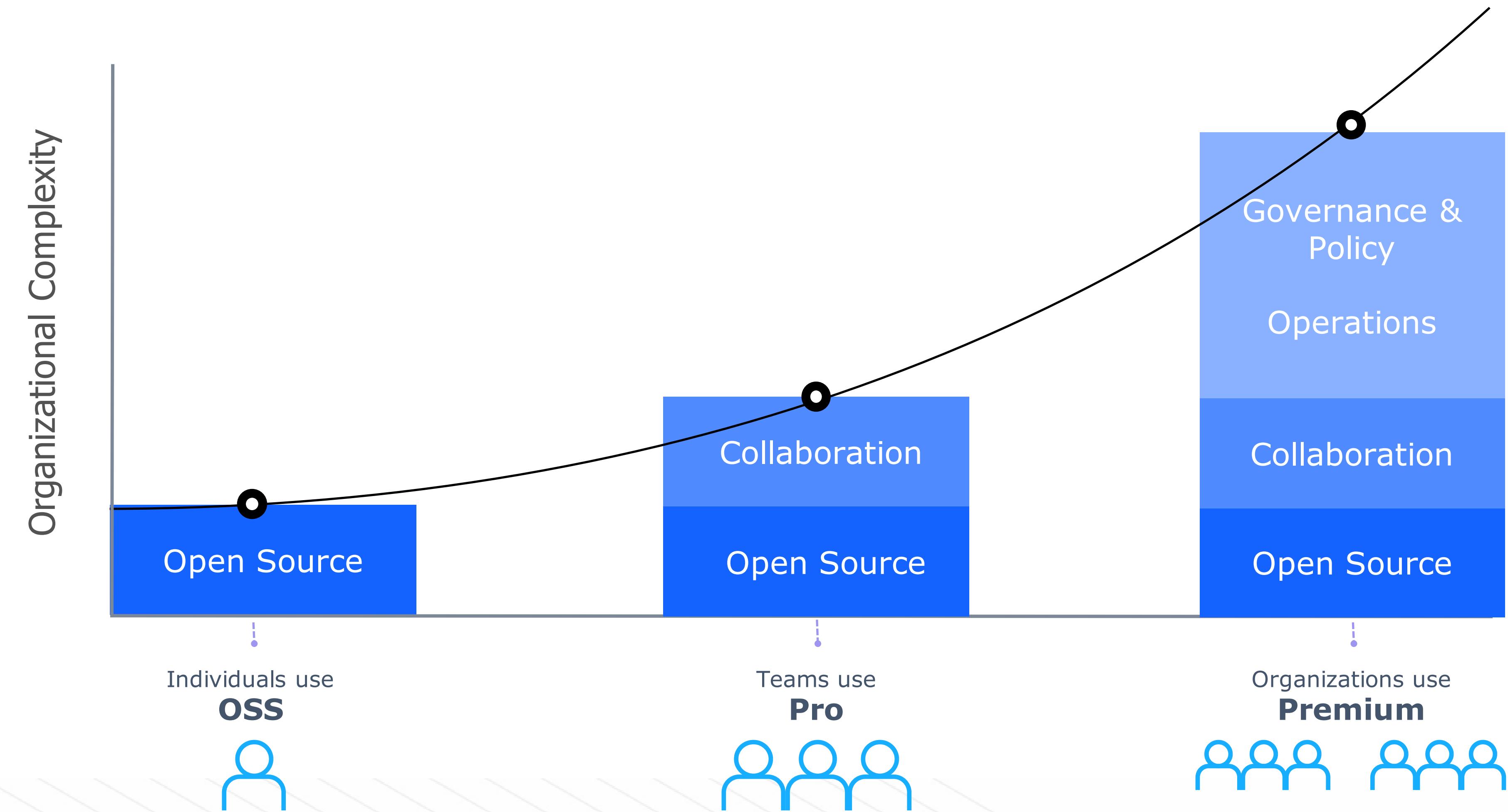
- Service discovery in a dynamic environment

- Knowledge of real-time configuration

The HashiCorp open source suite



Enterprise products build on open source to address organizational complexity



Azure Resource Manager



Azure Portal

<https://portal.azure.com>

The screenshot shows the Microsoft Azure Portal dashboard. The left sidebar contains a navigation menu with options like 'Create a resource', 'All services', 'FAVORITES' (with 'Resource groups', 'All resources', 'Recent', 'App Services', 'SQL databases', 'Virtual machines', 'Subscriptions', 'Storage accounts', 'Azure Active Directory', 'Dashboard', 'API Management services', 'Function Apps', 'Cost Management + Billing', 'Policy', and 'Tags'), and a 'Dashboard' section. The main content area displays a 'Visual Studio Premium with MSDN MONETARY CREDIT' card showing 'CREDIT LEFT 87.39 USD' and 'DAYS LEFT 11'. It also shows the current time '10:32 AM' on 'TUESDAY, MAY 8, 2018'. Below these cards is a 'Quickstart tutorials' section with links to 'Windows Virtual Machines', 'Linux Virtual Machines', 'App Service', 'Functions', and 'SQL Database'.

<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.

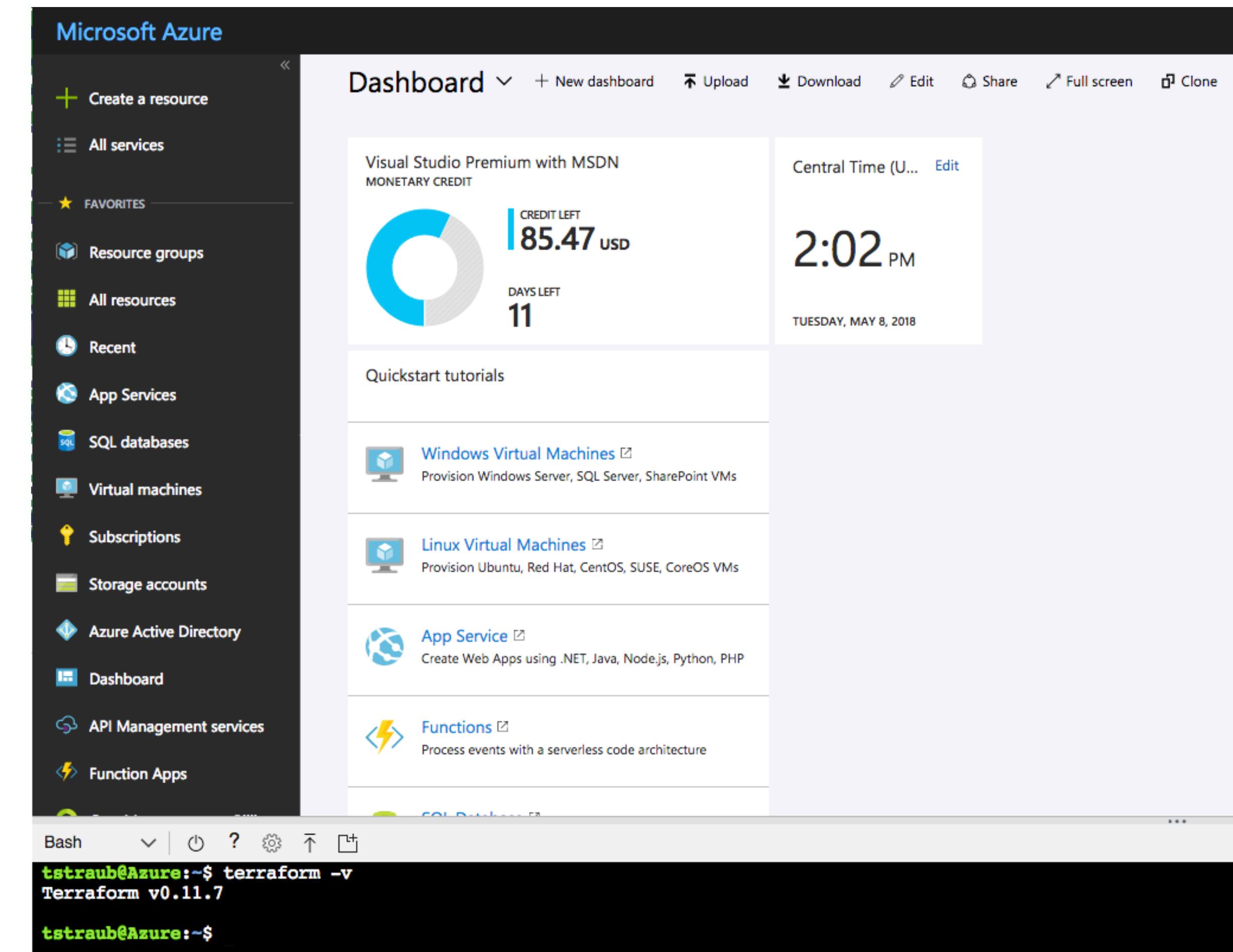
Today, we're extending our partnership and will offer an increasing number of services directly supported by Terraform, including [Azure Container Instances](#), [Azure Container Service](#), [Managed Disks](#), [Virtual Machine Scale Sets](#) and others. We want to give additional flexibility to express 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

Microsoft <3 Terraform

Azure Cloud Shell Integration



[cloud shell bash](#)



Microsoft <3 Terraform

VS Code Extension

Visual Studio | Marketplace Sign in

Visual Studio Code > Other > Azure Terraform New to Visual Studio Code? Get it

Azure Terraform Preview

Microsoft | 11,793 installs | ★★★★★ (1)

VS Code extension for developing with Terraform on Azure

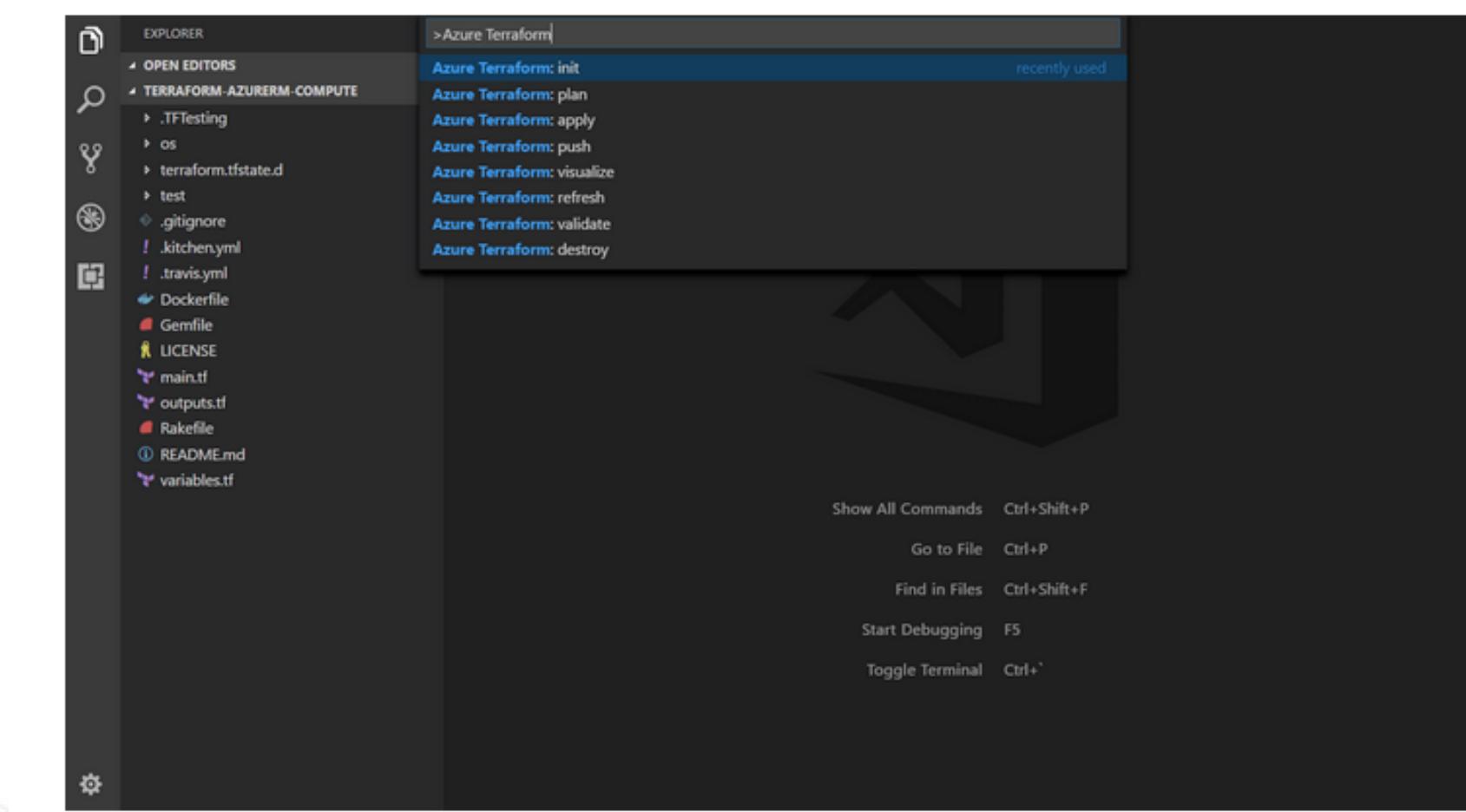
Install Trouble Installing?

Overview Q & A Rating & Review

Azure Terraform

build passing VS Marketplace v0.2.0

The VSCode Azure Terraform extension is designed to increase developer productivity authoring, testing and using Terraform with Azure. The extension provides terraform command support, resource graph visualization and CloudShell integration inside VSCode.



Categories

Other Azure

Tags

azure cloudshell devops terraform

Resources

Repository License Changelog Download Extension

Project Details

GitHub icon Azure/vscode-azureterraform No Pull Requests 7 Open Issues Last commit: 2 weeks ago

More Info

Version	0.2.0
Last updated	5/8/2018, 11:56:51 PM
Publisher	Microsoft
Unique Identifier	ms-azuretools.vscode-azureterraform

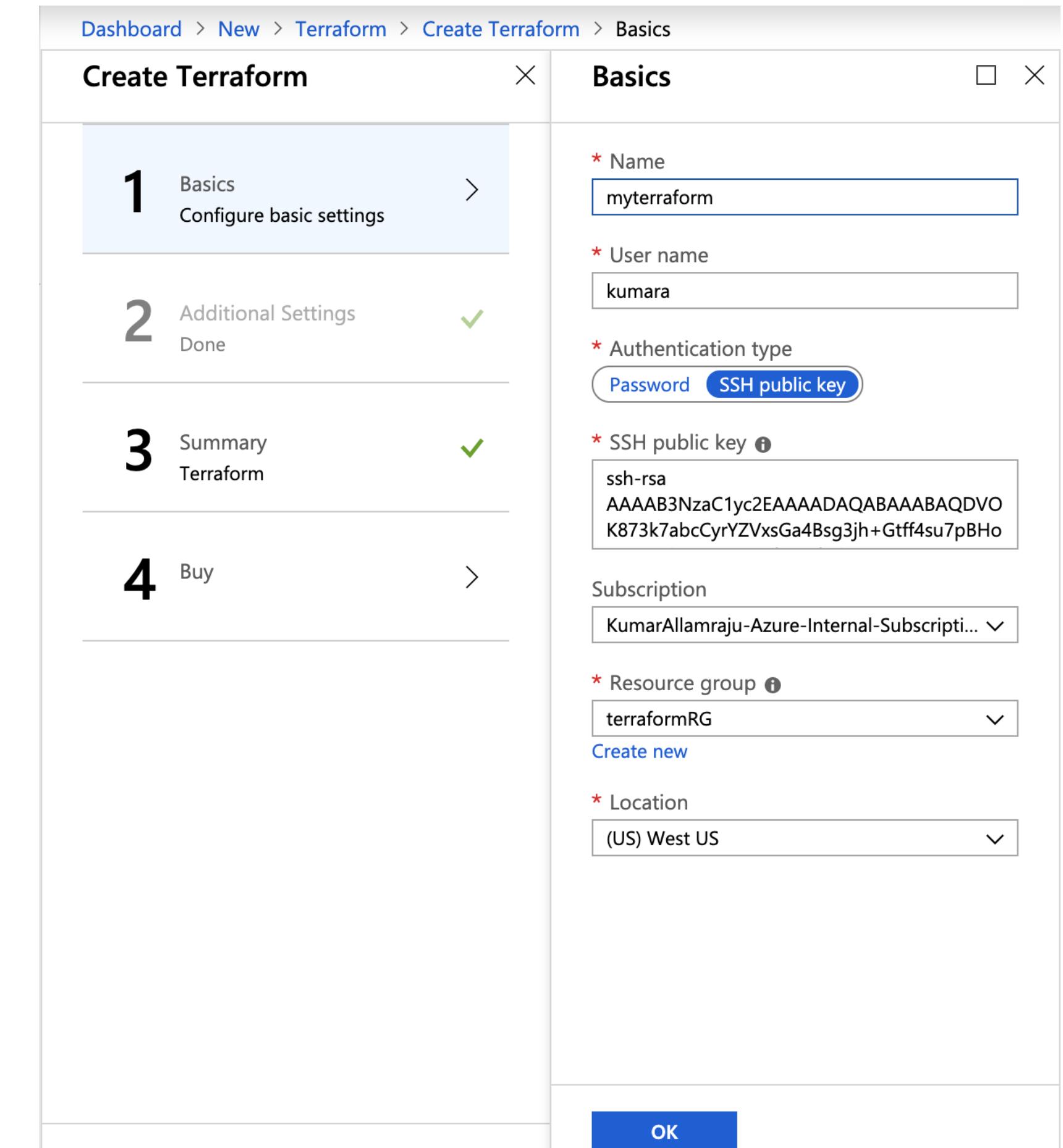
<https://docs.microsoft.com/en-us/azure/terraform/terraform-cloud-shell>



Microsoft <3 Terraform

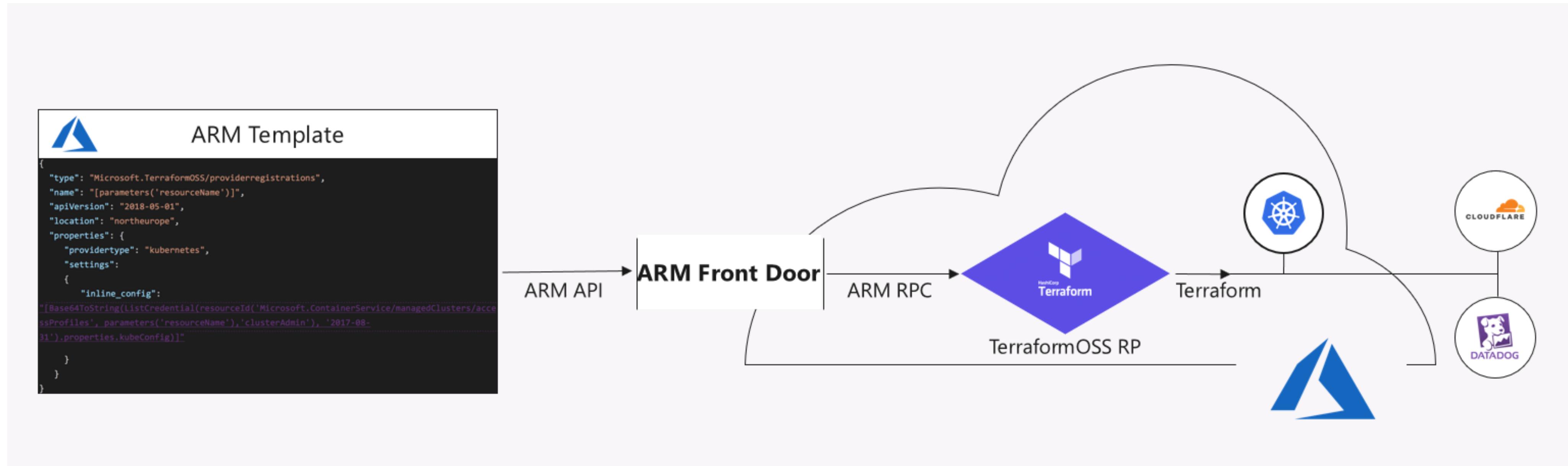
Terraform Marketplace Image

- 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.



Microsoft <3 Terraform

Terraform 3rd Party Azure Provider



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



Terraform



Terraform

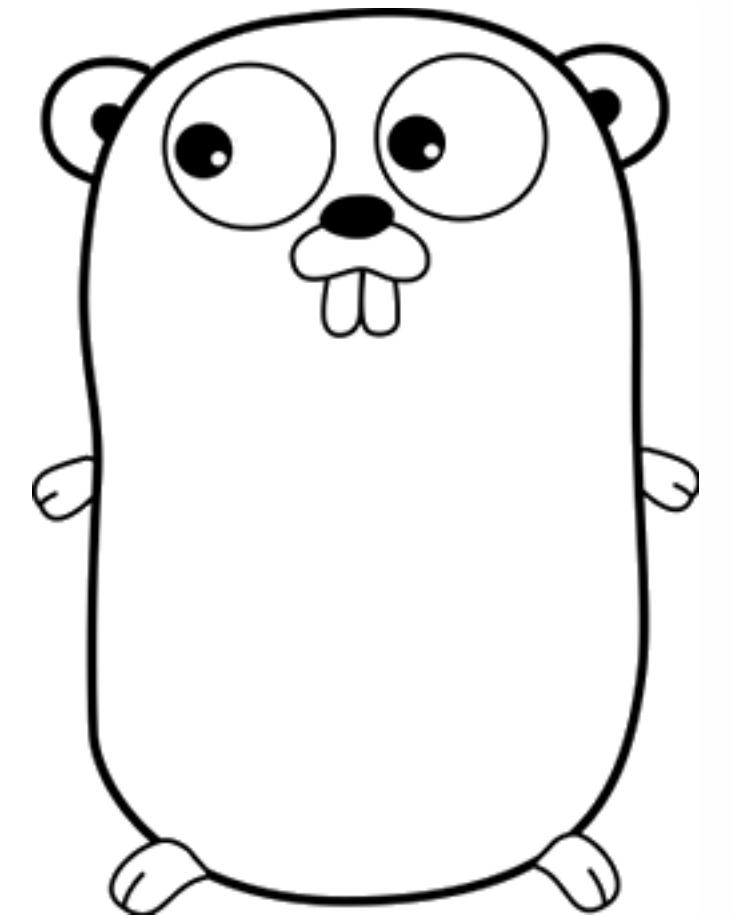
<https://www.terraform.io/>

Open Source Software

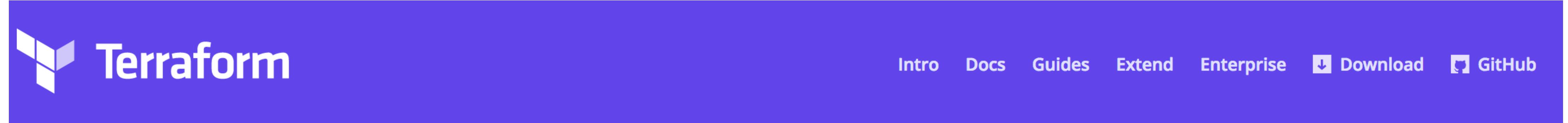
<https://github.com/hashicorp/terraform>

Written in GO

<https://golang.org/>



Terraform Documentation



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

Terraform Documentation

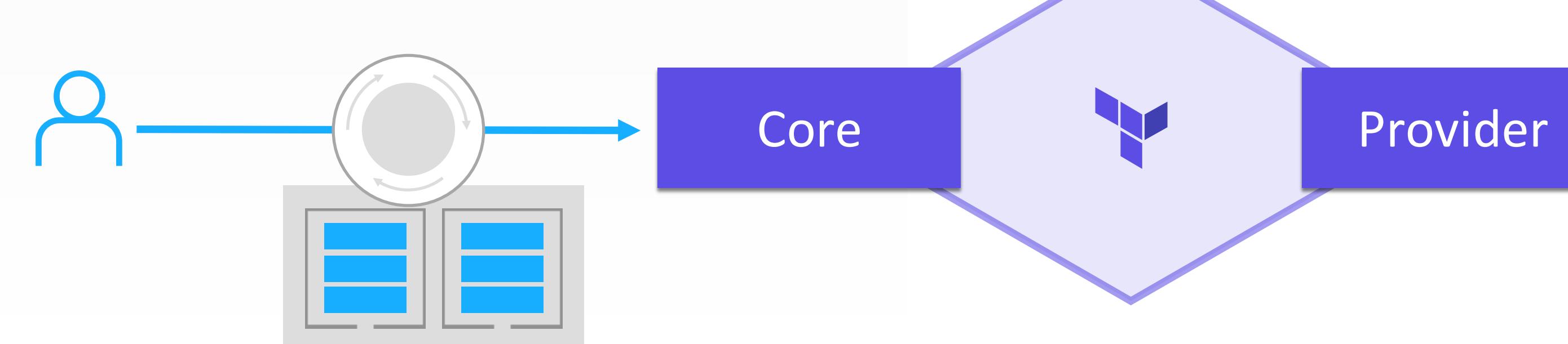
Welcome to the Terraform documentation! This documentation is more of a reference guide for all available features and options of Terraform. If you're just getting started with Terraform, please start with the [introduction and getting started guide](#) instead.



Terraform Ecosystem

Partnerships with many technology providers, and 4000+ modules

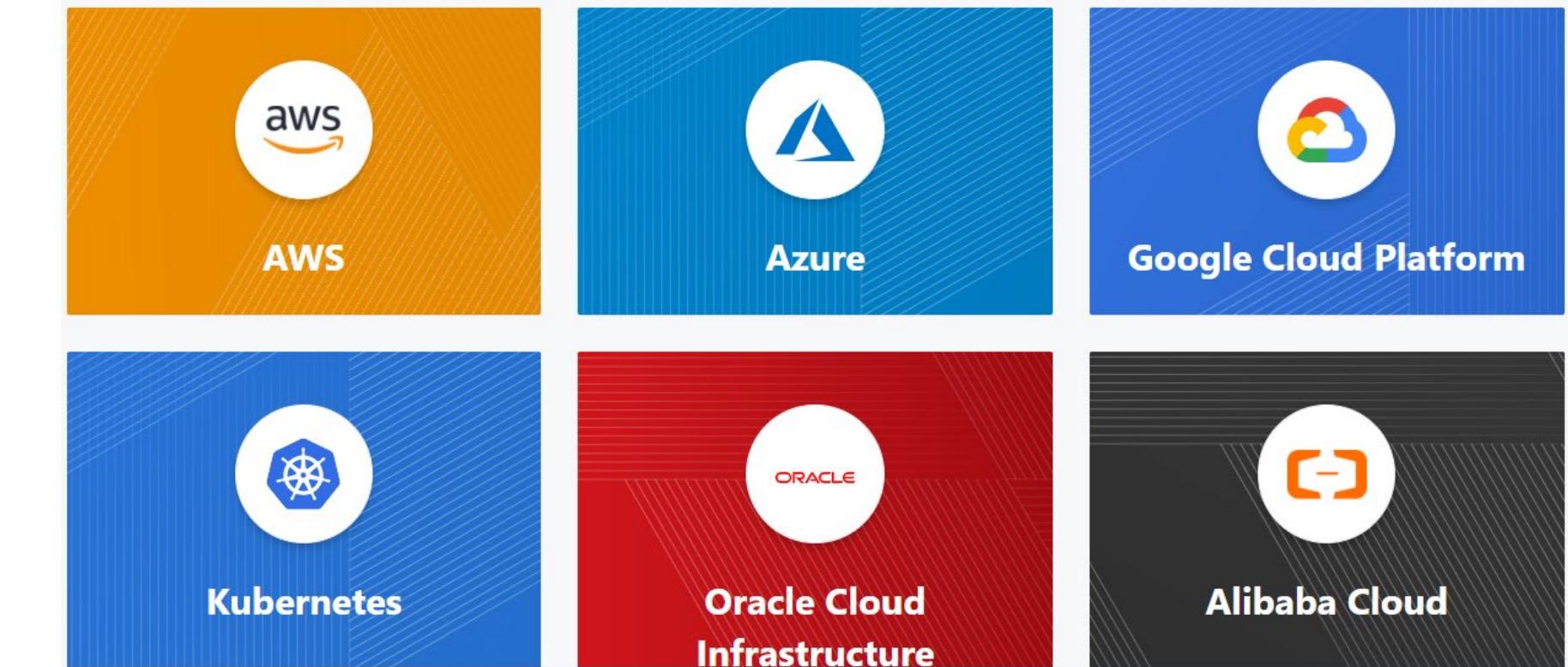
VERSION-CONTROL SYSTEMS



INFRASTRUCTURE PROVIDERS

Providers

Providers are a logical abstraction of an upstream API. They are responsible for understanding API interactions and exposing resources.



List of all providers [Providers](#)

Terraform Providers

 Terraform Providers !

Provider Plugins for HashiCorp Terraform

San Francisco, CA <https://www.hashicorp.com> hello@hashicorp.com

Repositories 76 People 1 Projects 0

Pinned repositories

[terraform-provider-aws](#)
Terraform AWS provider
Go 1.1k 970

[terraform-provider-google](#)
Terraform Google Cloud provider
Go 235 196

[terraform-provider-azurerm](#)
Terraform Azure Resource Manager provider
Go 318 267

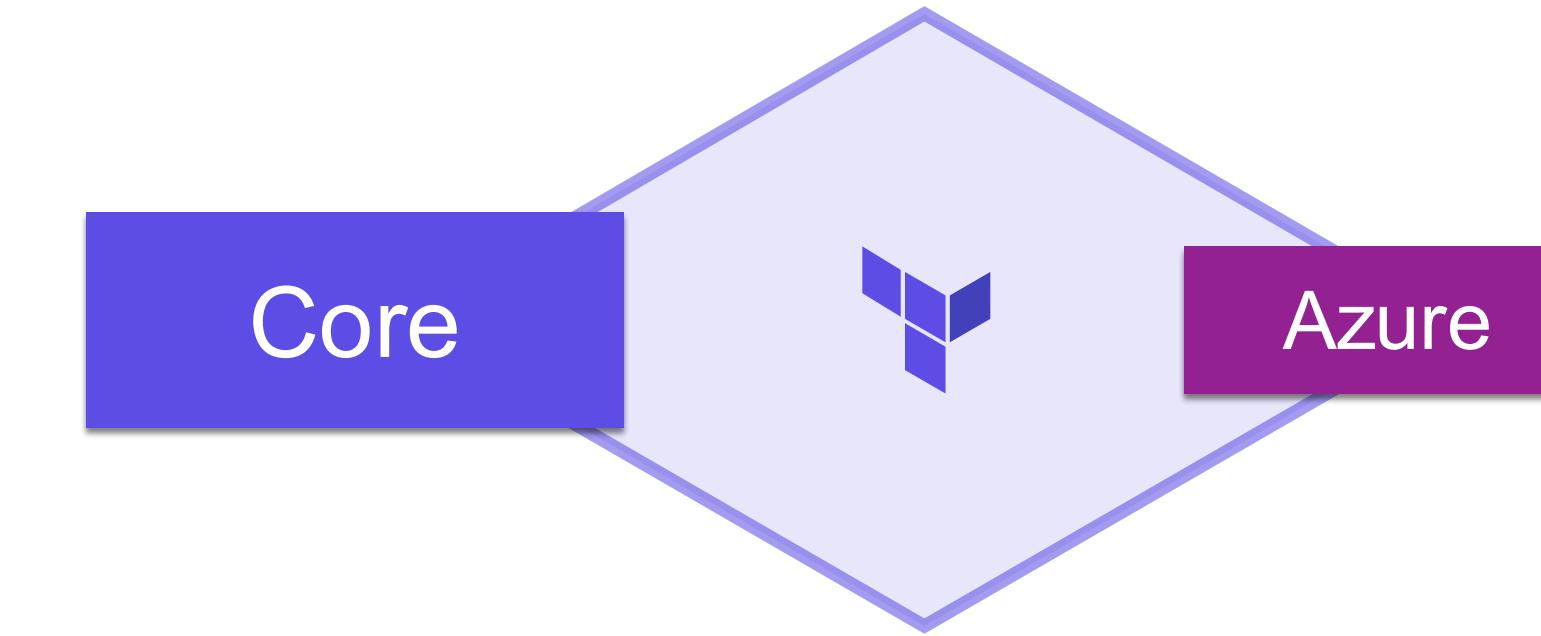
[terraform-provider-opc](#)
Terraform Oracle Public Cloud provider
Go 17 13

[terraform-provider-vsphere](#)
Terraform VMware vSphere provider
Go 118 85

[terraform-provider-kubernetes](#)
Terraform Kubernetes provider
Go 145 96



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**



azurerm

Official by HashiCorp

Public Cloud

Lifecycle management of Microsoft Azure using the Azure Resource Manager APIs. maintained by the Azure team at Microsoft and the Terraform team at HashiCorp

VERSION	PUBLISHED	INSTALLS	SOURCE CODE
2.73.0	3 days ago	69.5M	hashicorp/terraform-provider-azurerm

VERSION	PUBLISHED	INSTALLS	SOURCE CODE
2.93.0	a day ago	102.1M	hashicorp/terraform-provider-azurerm

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.



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.



main.tf

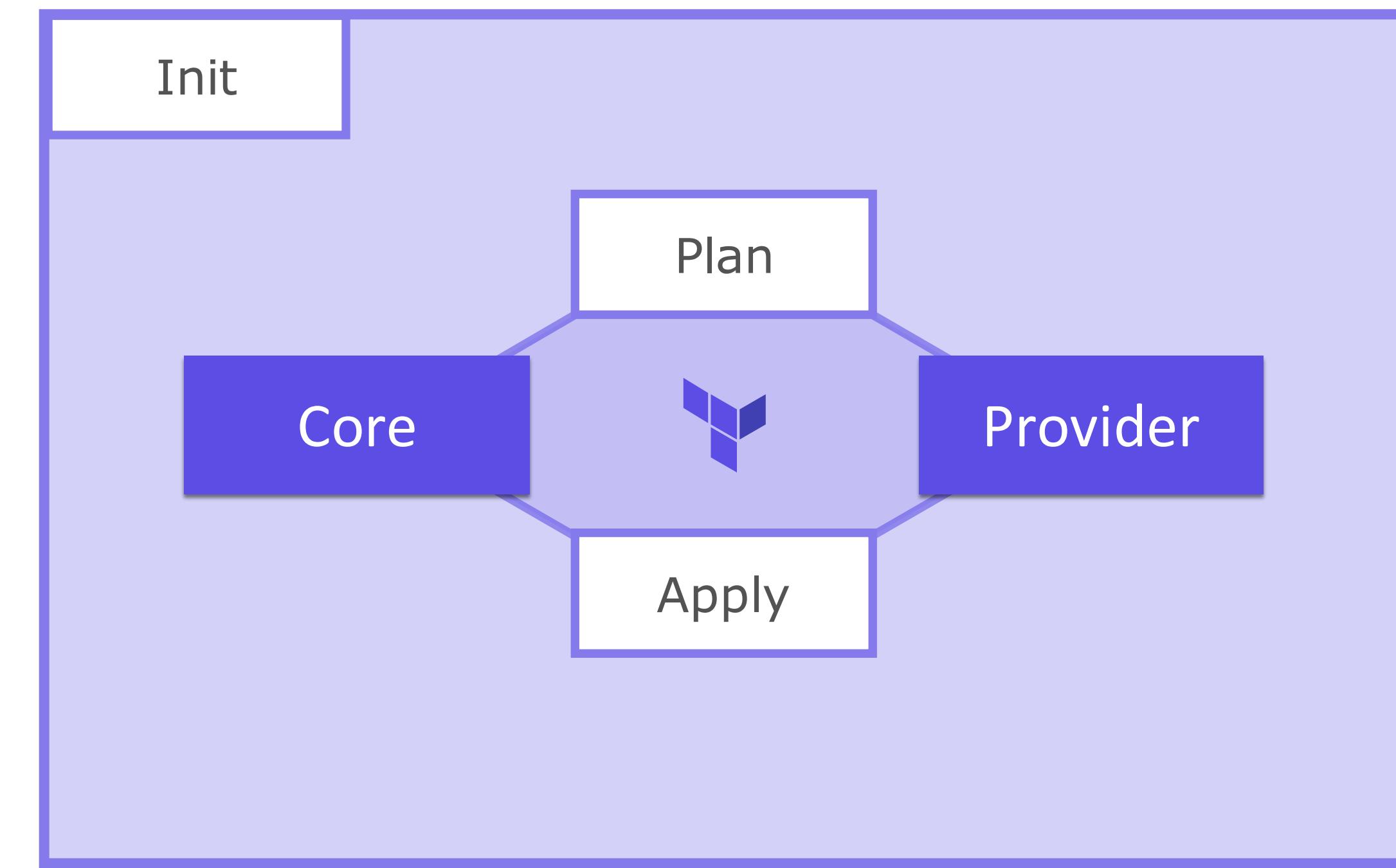
```
provider "azurerm" {
  features {}
}

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

Terraform in Action



Terraform Workflow



DEMO

Init

Plan

Appy

Destroy



● ● ● Terminal

```
peter@Azure:~/tfest$ cat main.tf

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "myfirsttrg" {
  name      = "myfirstresourcegroup"
  location = "uksouth"
}

peter@Azure:~/tfest$
```

● ● ● Terminal

```
peter@Azure:~/tfest$ 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.
```

● ● ● Terminal

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

```
Terraform used the selected providers to generate the following plan. To see the actual plan results, run 'terraform apply'.  
+ 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.
```

● ● ● Terminal

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

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

Terraform will perform the following actions:

# azurerm_resource_group.myfirststrg will be created
+ resource "azurerm_resource_group" "myfirststrg" {
    + 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.myfirststrg: Creating...
azurerm_resource_group.myfirststrg: Creation complete after 1s [id=/subscription

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

● ● ● Terminal

```
peter@Azure:~/tfest$ terraform destroy -auto-approve  
azurerm_resource_group.myfirstrg: Refreshing state... [id=/subscriptions/324bd9d9-0748-480e-81d1-ffdbe26503/resourceGroups/myfirstresourcegroup]
```

```
Terraform used the selected providers to generate the following execution plan.  
- 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/resourceGroups/myfirstresourcegroup"  
    - 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/324bd9d9-0748-480e-81d1-ffdbe26503/resourceGroups/myfirstresourcegroup]
```

```
azurerm_resource_group.myfirstrg: Still destroying... [id=/subscriptions/324bd9d9-0748-480e-81d1-ffdbe26503/resourceGroups/myfirstresourcegroup]
```

```
azurerm_resource_group.myfirstrg: Destruction complete after 15s
```

```
Destroy complete! Resources: 1 destroyed.
```

```
peter@Azure:~/tfest$
```

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



AzureRm Provider

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



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



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



Terraform State

Config: Desired State

State: Current State

Diff: Delta of {Desired & Current}

Plan: Presents Diff

Apply: Resolves Diff

Terraform State

State file can
contain sensitive
data



Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			



Terraform State – Operations

Configuration	State	Reality	Operation
azurerm_virtual_machine.app			CREATE



Terraform State – Operations

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



Terraform State – Operations

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



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	



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



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	



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



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	



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



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	



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



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		



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



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"
}
```



Variables - Types

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



Variables - Types

```
variable "images" {  
  type = "map"  
  
  default = {  
    "dev" = "image1"  
    "test" = "image2"  
    "prod" = "image3"  
  }  
}
```



Variables - Types

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



Variables - Types

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



Variables - Scoping

Variables are valid within the current module level.

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

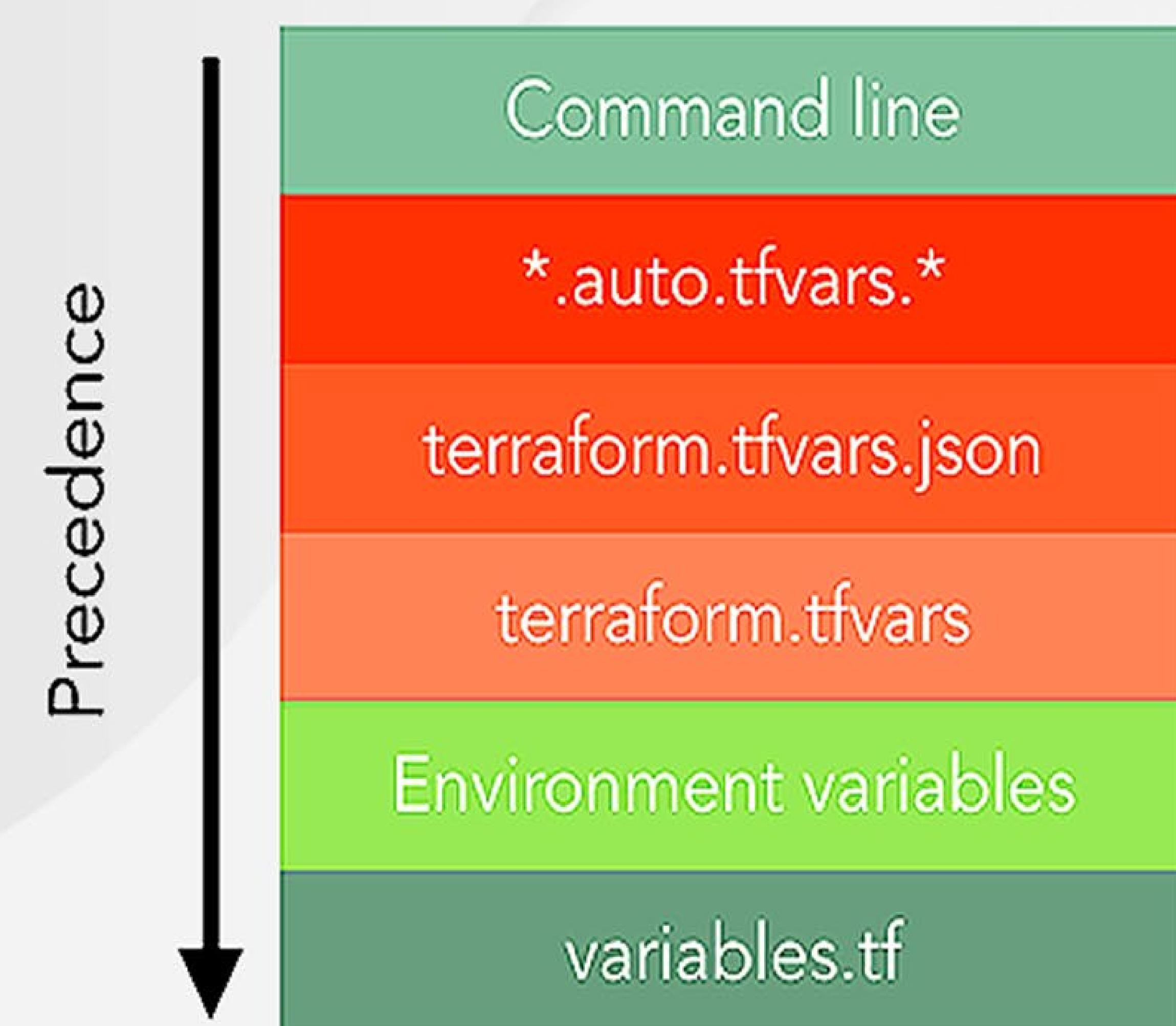


Variables – How to Use

Variable Files

- `terraform.tfvars`
- `*.auto.tfvars`

```
foo  = "bar"  
valid = true  
  
somelist = [  
    "one",  
    "two",  
]
```



Variables – How to Use

Command Line Parameters

`terraform plan -var 'foo=bar'`

`terraform plan -var 'foo=bar' -var 'bar=foo'`



Variables – How to Use

Environment Variables (TF_VAR_*)

```
export TF_VAR_foo=bar
```

```
export TF_VAR_bar=foo
```



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



Outputs

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



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.



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)
```



Interpolations - Conditionals

```
resource "azurerm_virtual_machine" "web" {  
    subnet = "${var.size == "small" ? var.small_vm : var.large_vm}"  
}
```



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).`



Interpolation – Math

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

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



Interpolations – locals

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



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!



Module Usage

```
module "mymodule" {  
  source  = "path/to/module"  
  
  name    = "mysuperapp"  
  
  ...  
}
```



Module Sources

Local

Github

Generic git repo

Bitbucket

HTTP URLs

Registry (support versioning)



Module Usage

```
module "windowsservers" {  
  source      = "Azure/compute/azurerm"  
  version     = "1.1.5"  
  
  location    = "eastus"  
  vm_hostname = "mywinvm"  
  vm_os_simple = "WindowsServer"  
}
```



Public Module Registry

The screenshot shows the main landing page of the Terraform Public Module Registry. At the top left is the HashiCorp Terraform logo and "Module Registry". At the top right are links for "Browse", "Publish", and "Sign-in". The center features the title "Terraform Module Registry" and a subtitle "Discover modules for common infrastructure configurations for any provider". Below this is a search bar containing "Popular searches: azure, compute, networking" and a magnifying glass icon. The background has a subtle geometric grid pattern.

Terraform Module Registry

Discover modules for common infrastructure configurations for any provider

Popular searches: azure, compute, networking

Find Terraform Modules

Use and learn from verified and community modules

<https://registry.terraform.io>



Private Module Registry

The screenshot shows the Terraform Enterprise Private Module Registry interface. The top navigation bar includes a logo, workspace dropdown (example_corp), workspace switcher (Workspaces), module switcher (Modules), BETA status, Documentation, Status, and a user profile icon.

The main area is titled "Modules" and features a search bar with the term "consul". Below the search bar are two buttons: "+ Design Configuration" and "+ Add Module".

The module list displays three entries:

- vpc** PRIVATE AWS Version 2.0.0 **Details**
- ecs** PRIVATE AWS Version 1.0.7 **Details**
- networking** PRIVATE **Details**

<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.



Backend- azurerm

```
terraform {  
  backend "azurerm" {  
    ressource_group_name      = "pm_portal_rg"  
  
    storage_account_name     = "myorgeastus8x76ghf"  
    container_name           = "terraformstate"  
    key                      = "dev.terraform.tfstate"  
  }  
}
```



Backend

State Locking – Allows for teams to work together.

Backend Types:

- azure SA
- artifactory
- Consul
- Many more ...

Terraform Enterprise/Cloud Manages this for you!



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



Resource Addressing

Module Path:

Addresses a module within a tree of modules.

module.A...

Can be multiple levels deep

module.A.module.B...



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.



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:

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



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.



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.



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.



Commands - apply

Usage: `terraform apply [options] [plan file]`



Commands - destroy

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



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.



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.



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



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
```



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.

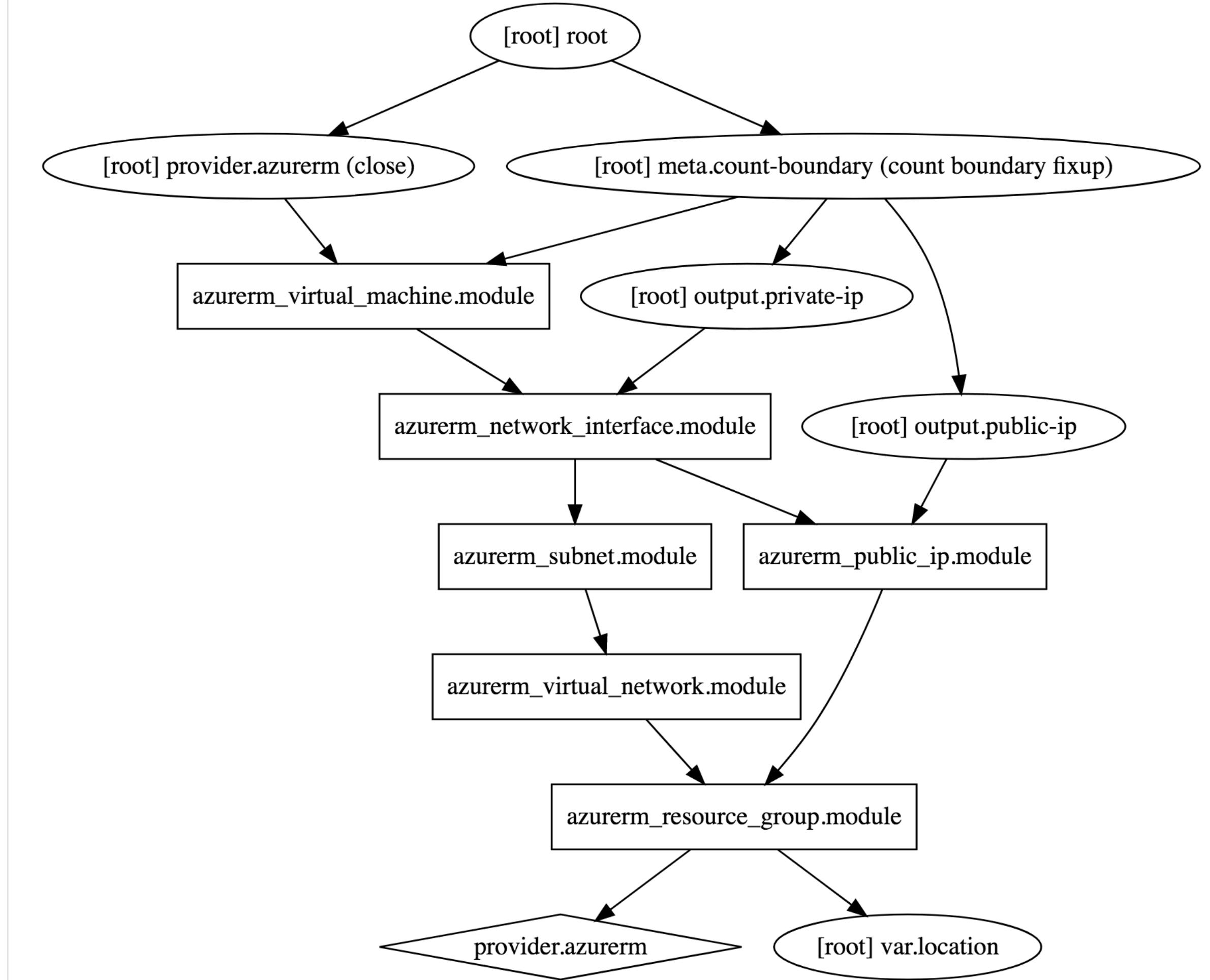


Commands - graph

`terraform graph |
dot -Tsvg > graph.svg`

See also

- [28mm/blast-radius](#)
-

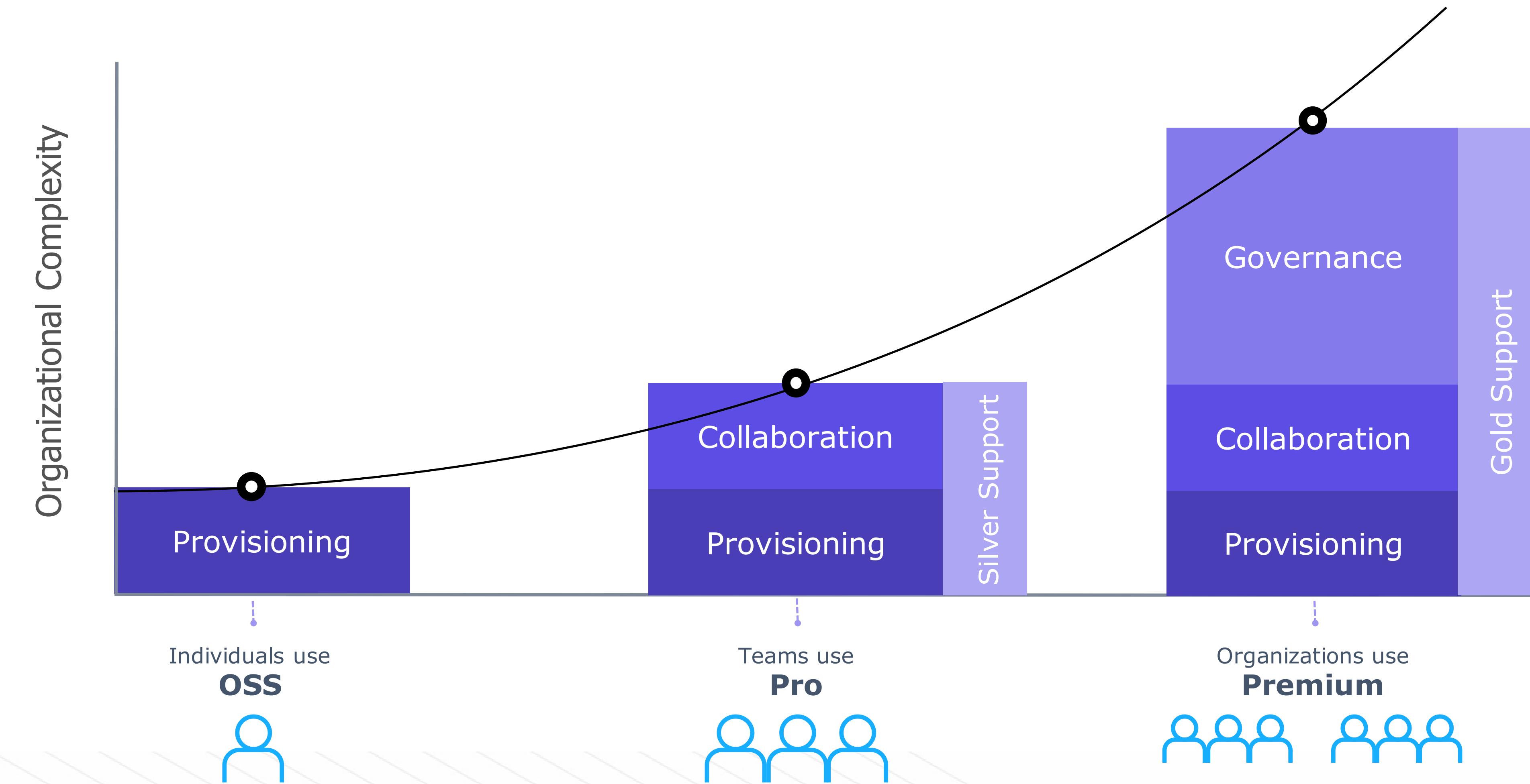


Terraform Enterprise

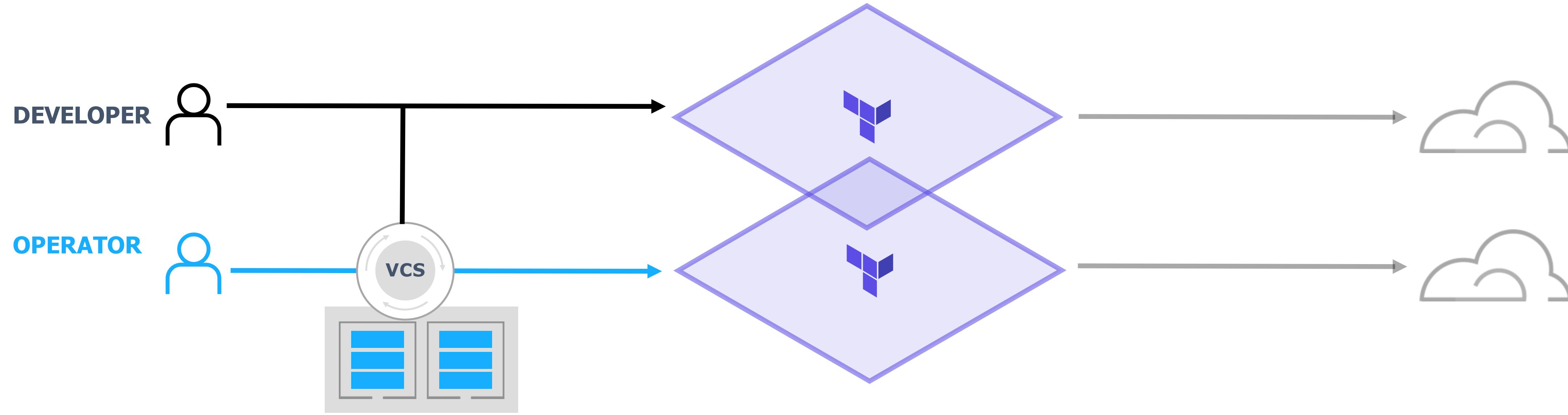


Terraform Open Source and Terraform Enterprise

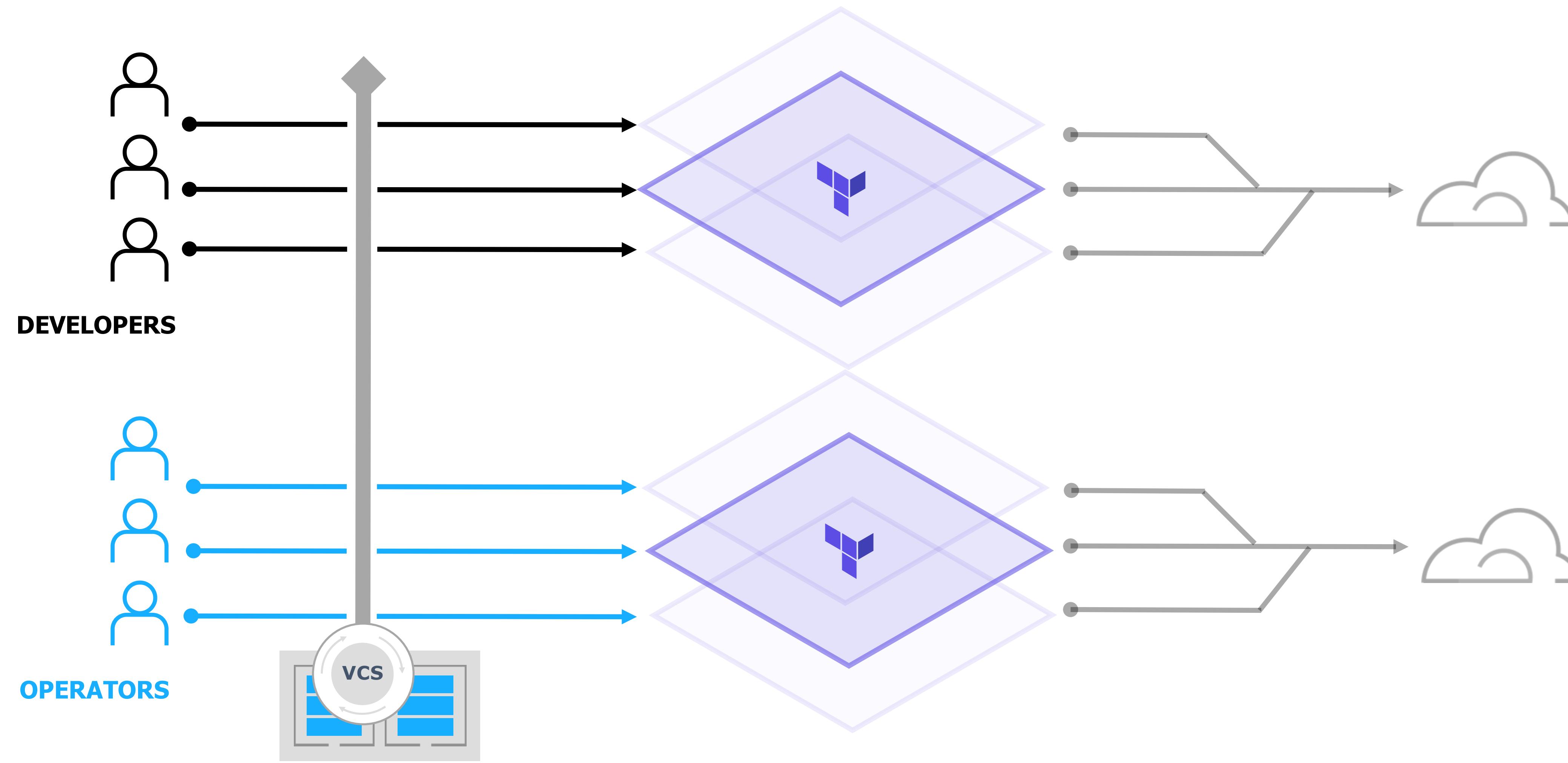
OSS for technical complexity & Enterprise for organizational complexity



Terraform OSS workflow for individuals



How can teams safely collaborate on infrastructure



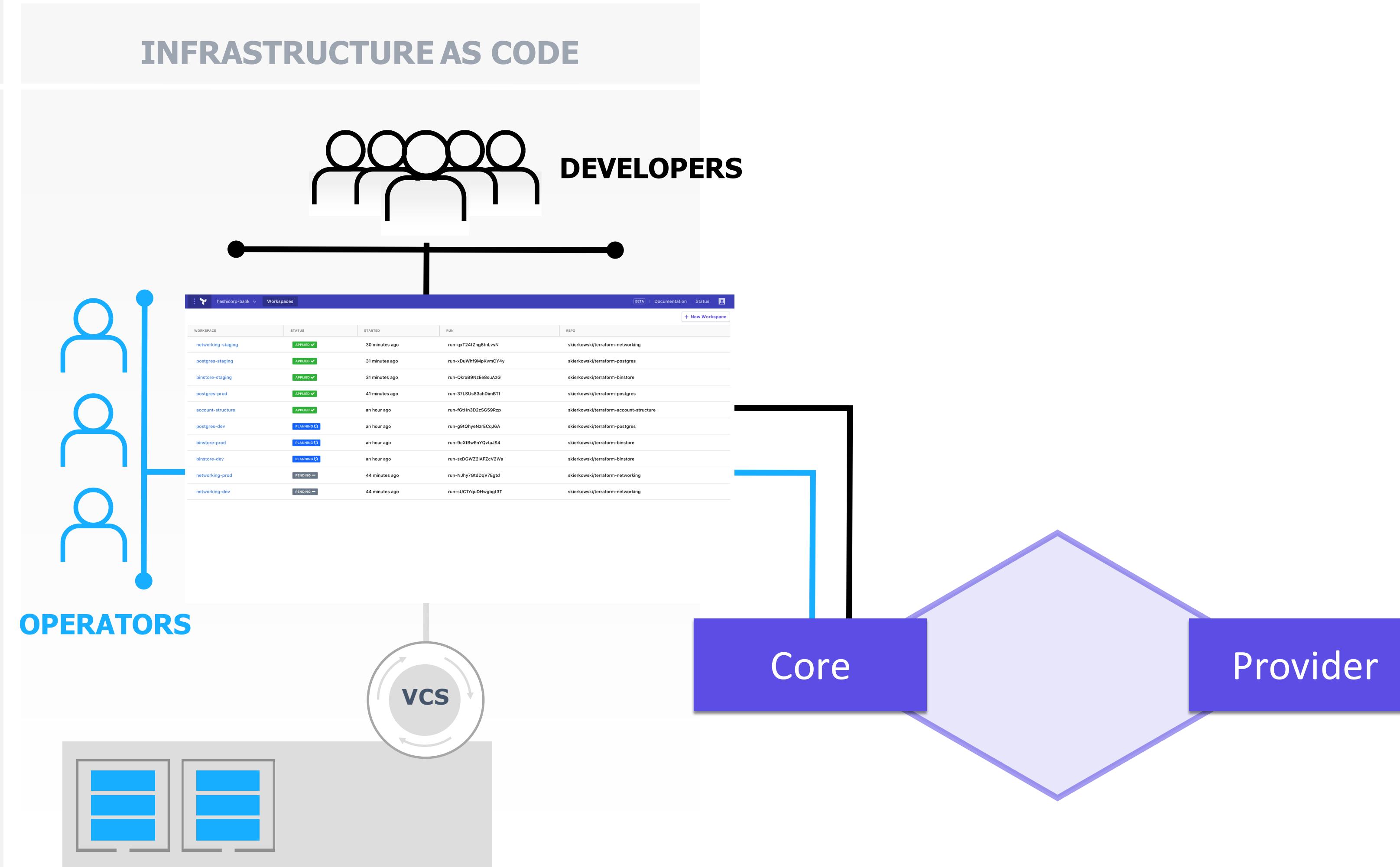
Demo



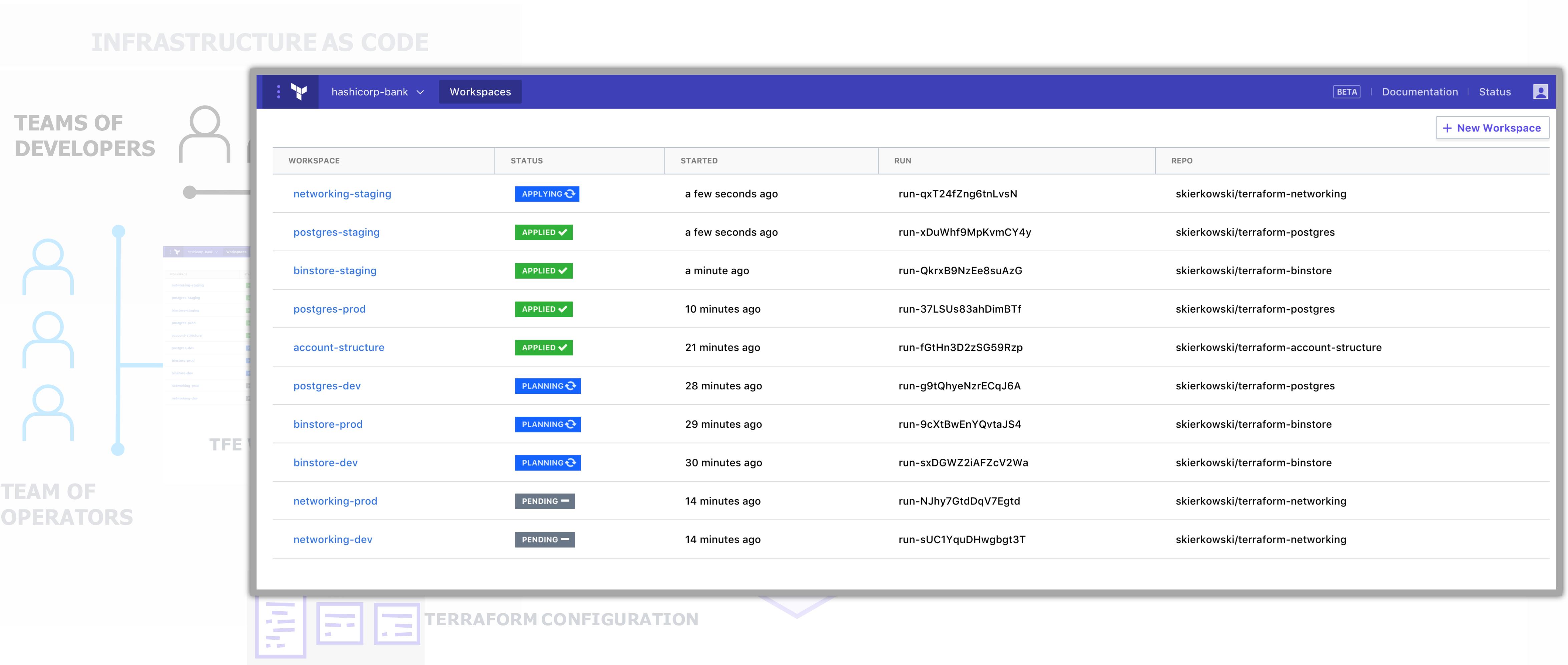
Workspaces



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.



Workspace – Variables

The screenshot shows the Terraform Cloud interface for managing workspace variables. At the top, there's a navigation bar with a logo, the organization name "cardinalsolutions", and tabs for "Workspaces" (which is selected) and "Modules". On the right of the nav bar are links for "Documentation", "Status", and a user profile icon. Below the nav bar, the workspace name "app-dev" is displayed, along with a "Queue Plan" button.

The main content area has a tab navigation bar with "Current Run", "Runs", "States", "Variables" (which is selected and highlighted in blue), "Settings", "Version Control", and "Access".

Terraform Variables

These variables are set using the `var` option when performing a plan and apply

Edit

name	app-dev
------	---------

Environment Variables

These variables are set using `export` in the environment running the plan and apply

Edit

ARM_CLIENT_ID	sensitive - write only
ARM_CLIENT_SECRET	sensitive - write only
ARM_SUBSCRIPTION_ID	sensitive - write only
ARM_TENANT_ID	sensitive - write only

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



Workspace – Variables

The screenshot shows the Terraform Cloud interface for managing workspace variables. At the top, there's a navigation bar with icons for account, organization, workspaces, and modules, followed by links for documentation, status, and user profile.

The main area is titled "app-dev". Below it, a navigation bar includes "Current Run", "Runs", "States", "Variables" (which is highlighted in blue), "Settings", "Version Control", and "Access".

Two variables are listed:

- my_map_variable**:
Value: {
 foo = "bar"
 bar = "foo"
}
Type: HCL (checkbox checked)
Sensitive: Sensitive checkbox (unchecked)
Add button
- my_super_secret**:
Value: reallycomplexpassword
Type: HCL (checkbox unchecked)
Sensitive: Sensitive checkbox (checked)
Add button

At the bottom, a note states: "These variables are set using `export` in the environment running the plan and apply". There are four environment variables listed: ARM_CLIENT_ID, ARM_CLIENT_SECRET, ARM_SUBSCRIPTION_ID, and ARM_TENANT_ID, each with a "Sensitive - write only" label.

<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

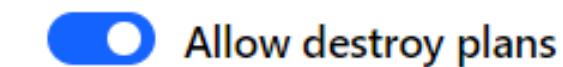


Workspace – Settings

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



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.

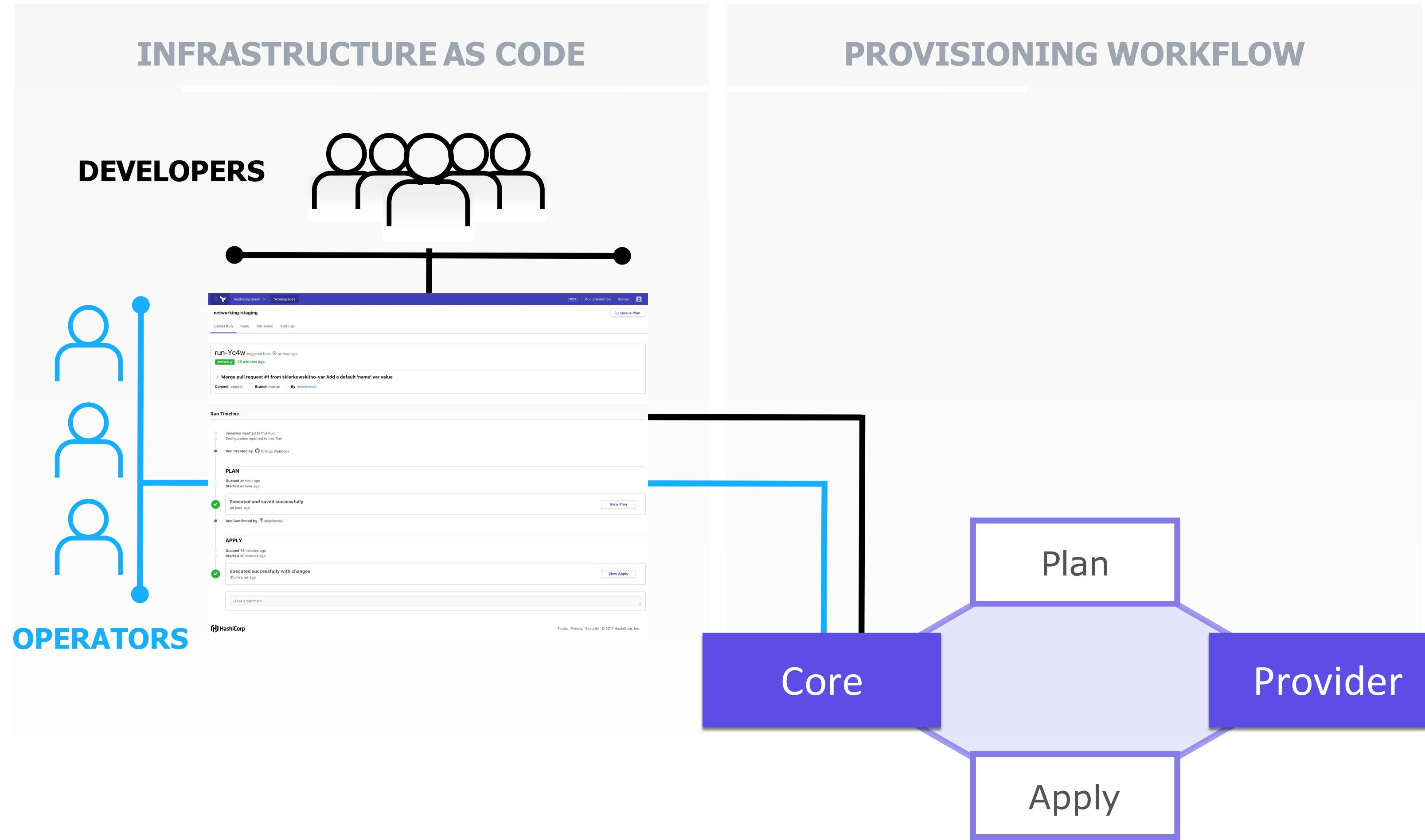
[Delete from Terraform Cloud](#)



Remote State in Cloud



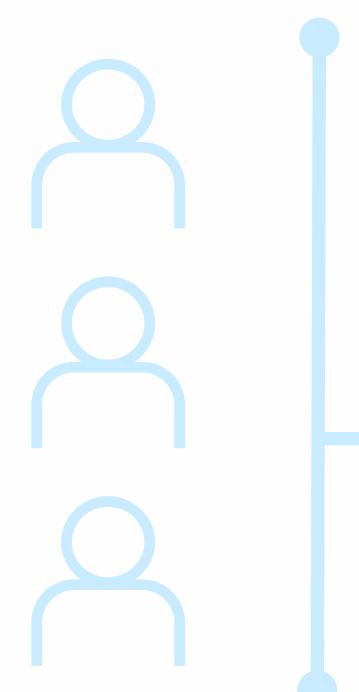
Remote Terraform runs and state management



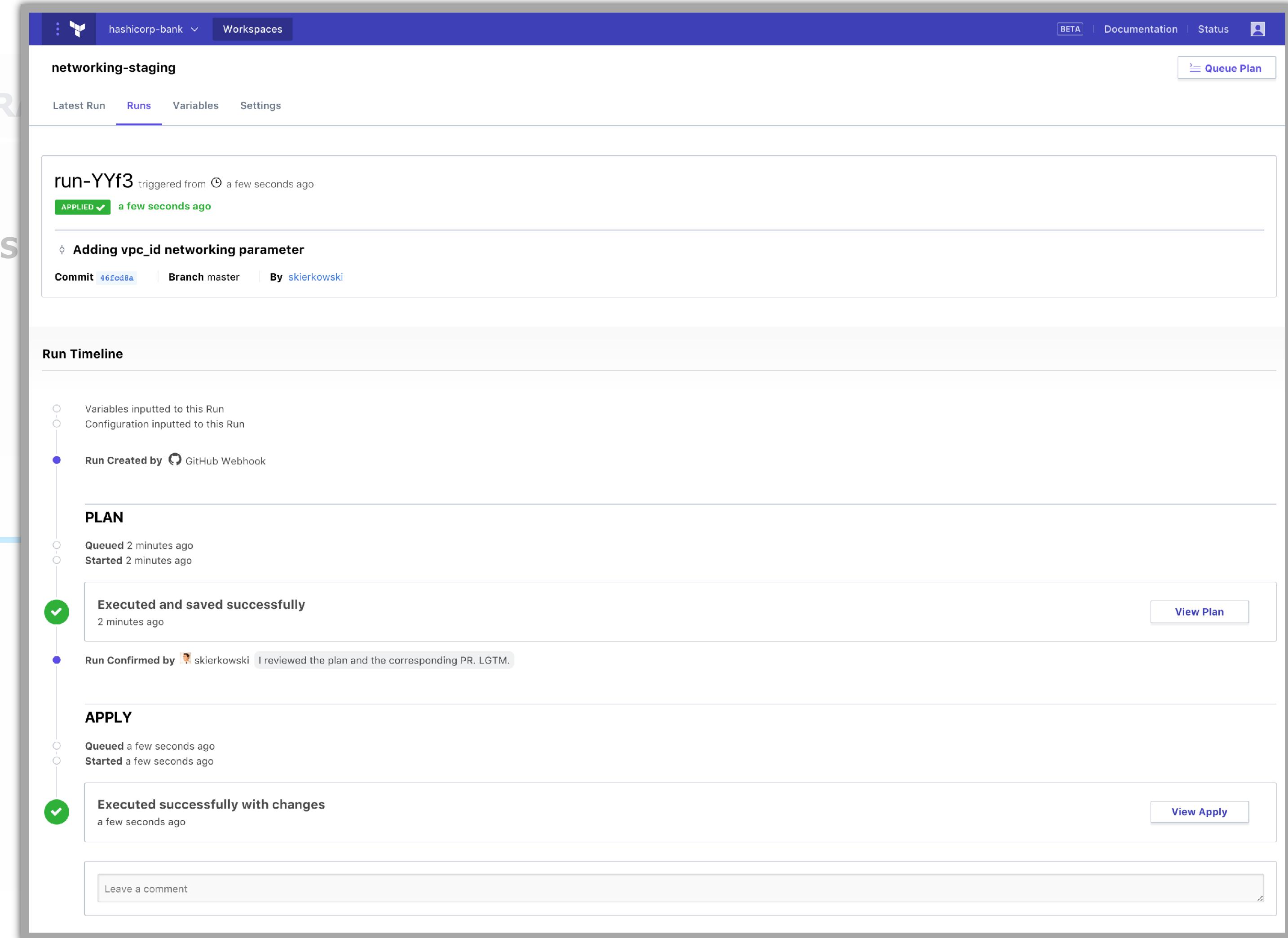
Remote state management and Terraform runs

INFR

TEAMS OF
DEVELOPERS



TEAM OF
OPERATORS



The screenshot displays the HashiCorp Terraform Cloud interface for a workspace named "networking-staging". The "Runs" tab is selected. A recent run, "run-YYf3", is shown, triggered from a GitHub webhook a few seconds ago. The run details include:

- APPLIED ✓ a few seconds ago**
- Adding vpc_id networking parameter**
- Commit 46fcd8a | Branch master | By skierkowski**

The "Run Timeline" section shows the following steps:

- Variables inputted to this Run
- Configuration inputted to this Run
- Run Created by GitHub Webhook**
- PLAN**
 - Queued 2 minutes ago
 - Started 2 minutes ago
- Executed and saved successfully** 2 minutes ago
- View Plan**
- Run Confirmed by skierkowski** I reviewed the plan and the corresponding PR. LGTM.
- APPLY**
 - Queued a few seconds ago
 - Started a few seconds ago
- Executed successfully with changes** a few seconds ago
- View Apply**

A "Leave a comment" input field is at the bottom.

Workspace – State Management

app-dev Queue Plan

Current Run Runs **States** Variables Settings Version Control Access

New state #sv-pzQ9A6977Z9rbDSg api-org-cardinalsolutions triggered from Terraform #run-yC3hTbyS2FUReg7f e9a2cd3	an hour ago
New state #sv-PEJfbFTxnmdwPpno api-org-cardinalsolutions triggered from Terraform #run-yC3hTbyS2FUReg7f e9a2cd3	an hour ago
New state #sv-FsULqVQpgQq6kijU api-org-cardinalsolutions triggered from Terraform #run-ErQRDW3JQtJvYwzU e99e2e5	an hour ago
New state #sv-SXMobWmQg6CpPhVc api-org-cardinalsolutions triggered from Terraform #run-ErQRDW3JQtJvYwzU e99e2e5	an hour ago
New state #sv-kDq9hzyEhnB6k6ae api-org-cardinalsolutions triggered from Terraform #run-oaG4yd1afQoGcqyz 33802b8	10 days ago



Workspace – State Management



Collaboration



Collaboration

Role Based Access Controls

Permissions

Read

Write

Admin



Service Accounts

API Tokens

- User
- Team
- Organization



User Authentication



Terraform
ENTERPRISE

Forgot password?

Enter your email address and we will send you a link to reset your password.

EMAIL

Need to confirm your email?

Send

Verify Two-factor Authentication

You have enabled two-factor authentication, but you must verify your device is able to receive and generate codes before two factor authentication is activated. If you no longer wish to use two factor authentication, you can cancel the process.

ENTER AUTHENTICATION CODE

Verify

Didn't receive an SMS text message on your device ending in 9466? [Resend Code](#).



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

Modules / Configuration Designer

Select Modules > Set Variables > Verify > Publish

Next »

The screenshot shows a software interface for 'Configuration Designer'. At the top, there's a navigation bar with the path 'Select Modules > Set Variables > Verify > Publish' and a green 'Next »' button. Below the navigation is a search bar with the placeholder 'consul' and a magnifying glass icon. A dropdown menu labeled 'Providers' is open. The main area is divided into two sections: 'ADD MODULES TO WORKSPACE' on the left and 'SELECTED MODULES' on the right.

ADD MODULES TO WORKSPACE

- appserver PRIVATE**
App Server portion of our 3-tier App
AZURERM Version 0.0.1
Details ↗ Add Module
- dataserver PRIVATE**
Data Server portion of our 3-tier App
AZURERM Version 0.0.1
Details ↗ Add Module

SELECTED MODULES 1

- appserver PRIVATE Version 0.0.1**
App Server portion of our 3-tier App
Details ↗ Remove ✕

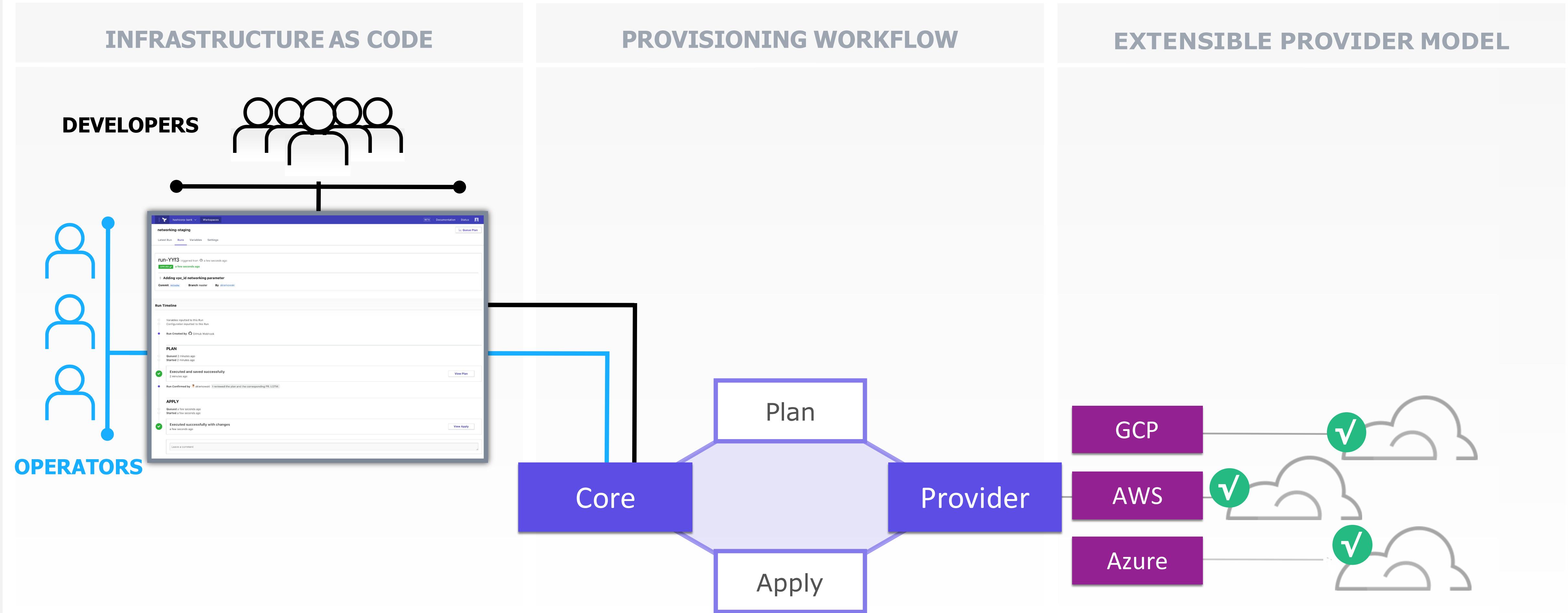


Governance



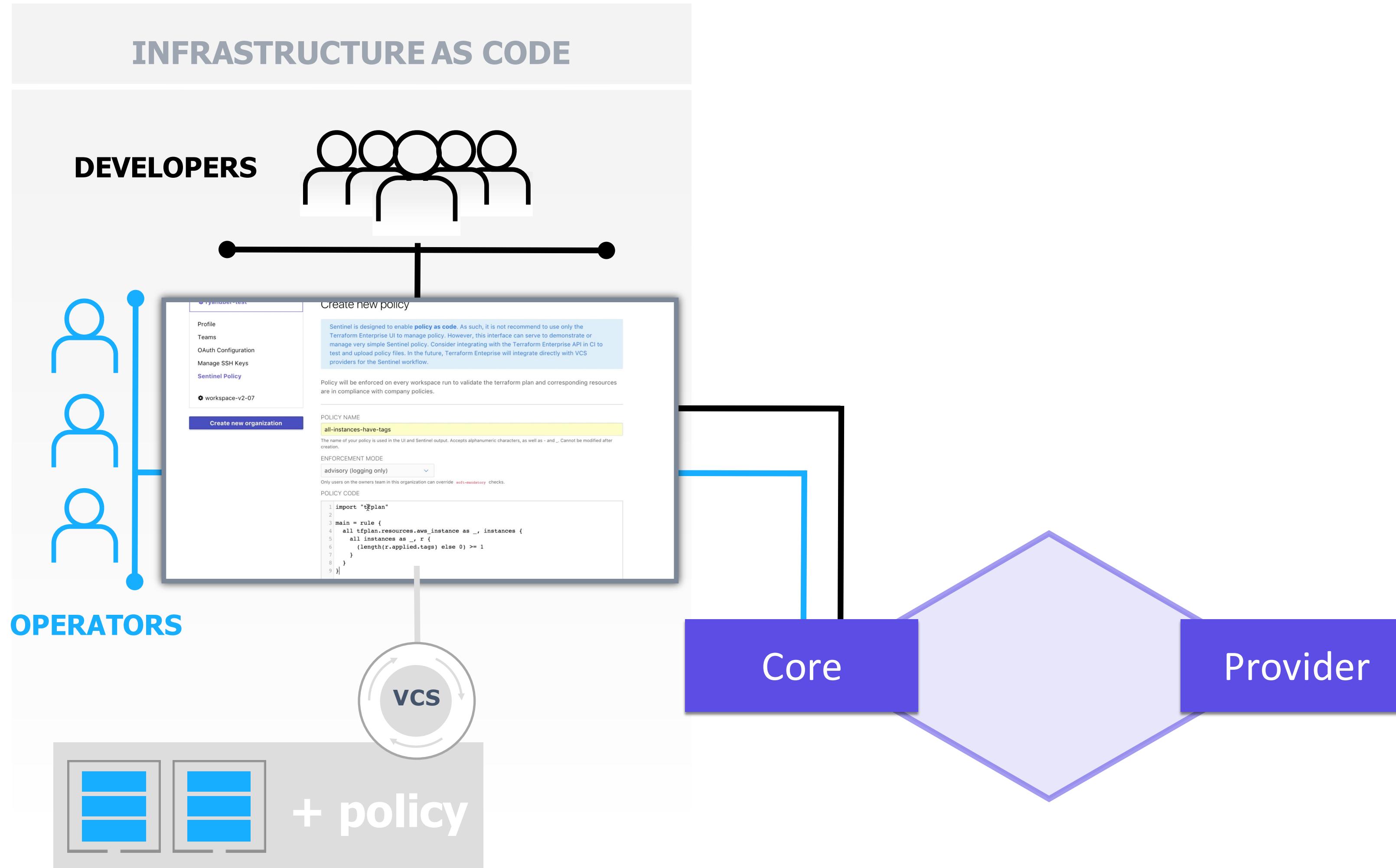
Enable Organization to have governance over infrastructure

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



Enable Organization to have governance over infrastructure

Automate policy 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

The diagram illustrates a organizational hierarchy. At the bottom, three blue human icons are labeled "OPERATORS". A vertical blue line connects them to a horizontal blue line. From the top of this line, another vertical blue line extends upwards to a single blue human icon at the top, which is connected to a workspace icon. This workspace icon is part of a larger interface window titled "Create new policy".

Sentinel Policy

Profile
Teams
OAuth Configuration
Manage SSH Keys
Sentinel Policy
workspace-v2-07

Create new organization

Create new policy

Sentinel is designed to enable **policy as code**. As such, it is not recommended to use only the Terraform Enterprise UI to manage policy. However, this interface can serve to demonstrate or manage very simple Sentinel policy. Consider integrating with the Terraform Enterprise API in CI to test and upload policy files. In the future, Terraform Enterprise will integrate directly with VCS providers for the Sentinel workflow.

Policy will be enforced on every workspace run to validate the terraform plan and corresponding resources are in compliance with company policies.

POLICY NAME
all-instances-have-tags

The name of your policy is used in the UI and Sentinel output. Accepts alphanumeric characters, as well as - and _. Cannot be modified after creation.

ENFORCEMENT MODE
advisory (logging only)

Only users on the owners team in this organization can override `soft-mandatory` checks.

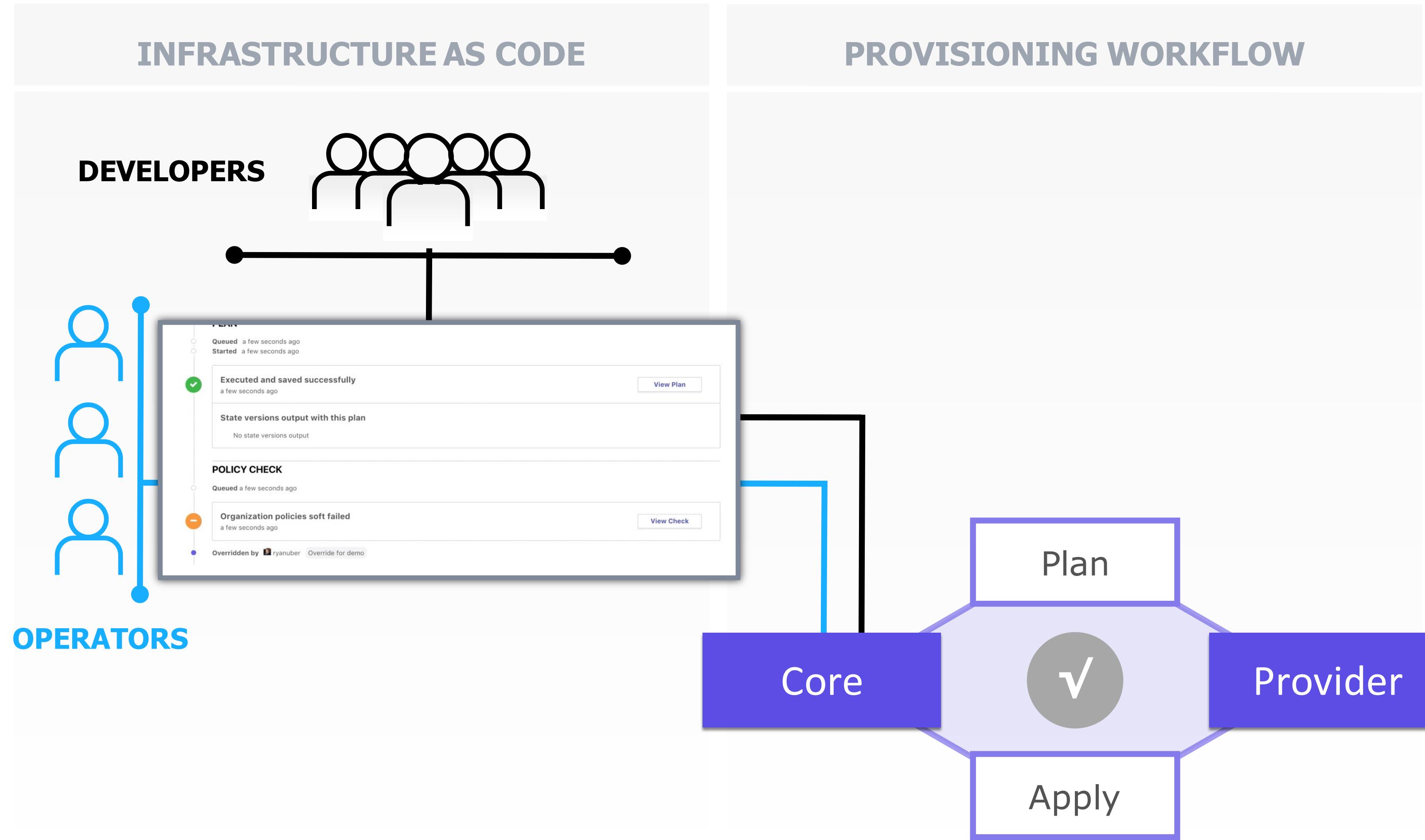
POLICY CODE

```
1 import "tfplan"
2
3 main = rule {
4   all tfplan.resources.aws_instance as _, instances {
5     all instances as _, r {
6       (length(r.applied.tags) else 0) >= 1
7     }
8   }
9 }
```



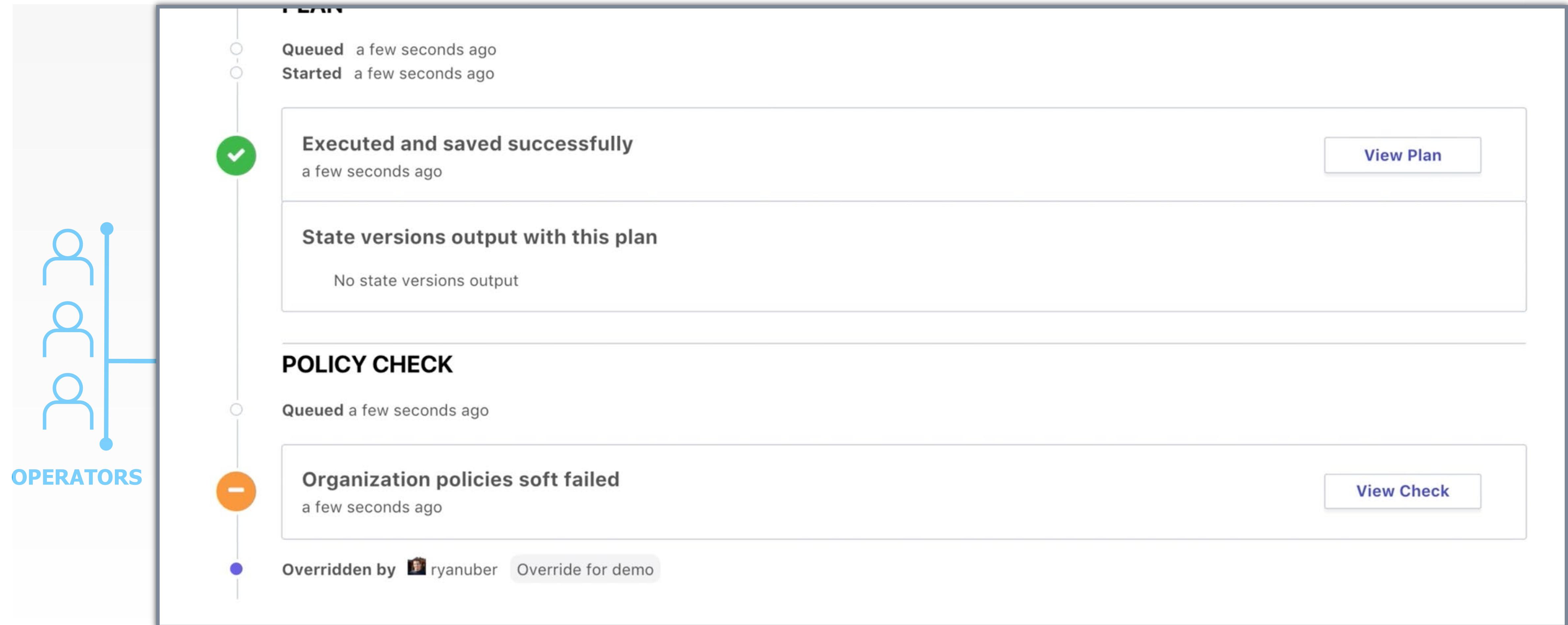
Enable Organization to have governance over infrastructure

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

