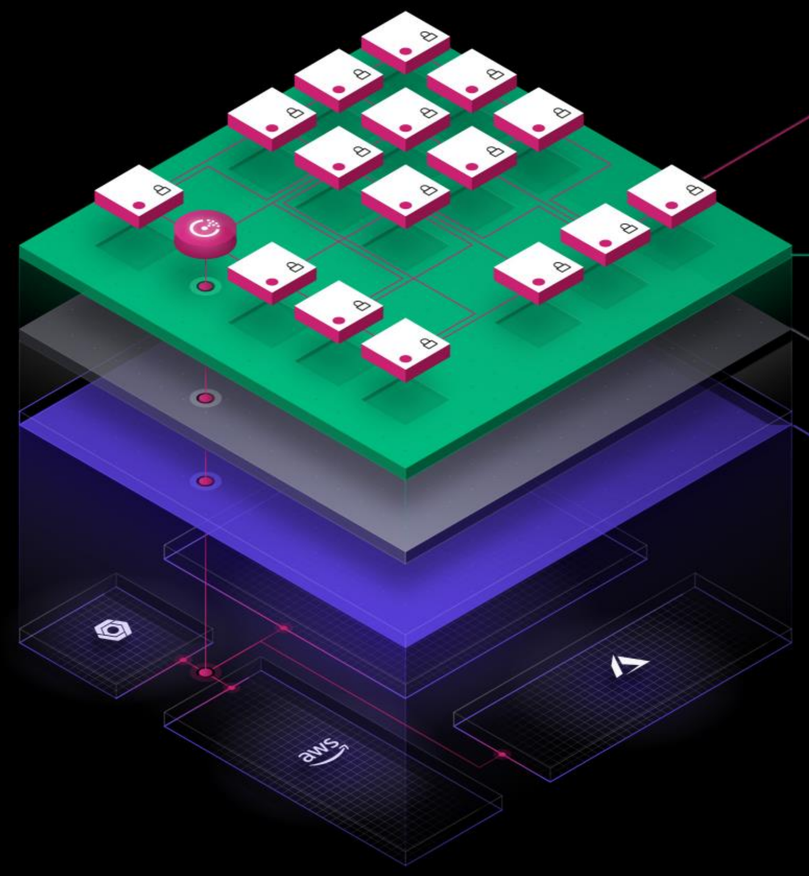
HashiCorp

Terraform

Multi cloud. Hybrid cloud. Any infrastructure. Anywhere.

Enabling safe and efficient provisioning and management.

The 4 essential elements of dynamic infrastructure



Networking

Connect infrastructure and applications

Development

Run applications

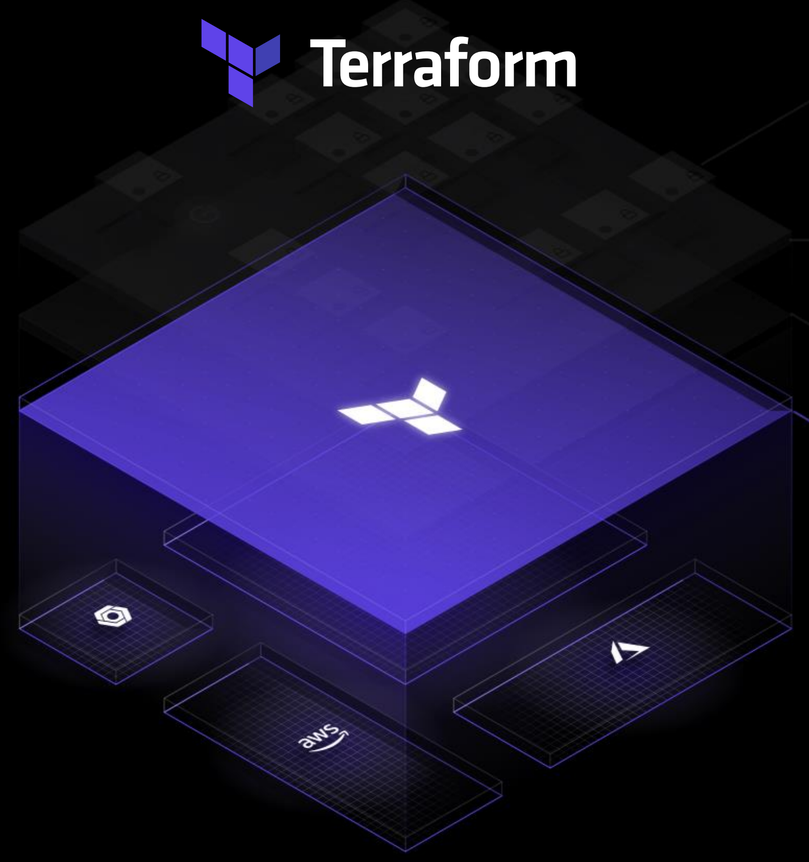
Security

Secure applications and infrastructure

Operations

Provision Infrastructure

The 4 essential elements of dynamic infrastructure



Networking

Connect infrastructure and applications

Development

Run applications

Security

Secure applications and infrastructure

Operations

Provision Infrastructure

The Transition to Multi-Cloud

The Transition to Cloud and Multi-Cloud

Traditional Datacenter

"Static"

Modern Datacenter

"Dynamic"

Dedicated Infrastructure

Private Cloud

AWS + Azure + GCP + ...

SYSTEMS OF RECORD ⟷ **SYSTEMS OF ENGAGEMENT**

"Tickets-based"

"Self service"

Implications of the Cloud Operating Model

	STATIC	DYNAMIC
Run	Dedicated Infrastructure	Scheduled across the fleet
Connect	**Host**-based Static IP	**Service**-based Dynamic IP
Secure	High trust **IP**-based	Low trust **Identity**-based
Provision	**Dedicated** servers Homogenous	Capacity **on-demand** Heterogenous

The Cloud Landscape

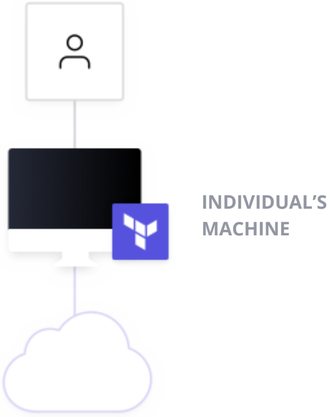
	STATIC		DYNAMIC			
	Dedicated		**Private Cloud**	**AWS**	**Azure**	**GCP**
Run Development	vSphere	→	vSphere	EKS/ECS Lambda	AKS/ACS Azure Functions	GKE Cloud Functions
Connect Networking	Hardware	→	Various Hardware	Proprietary	Proprietary	Proprietary
Secure Security	**IP**: Hardware	→	**Identity**: AD/LDAP	**Identity**: AWS IAM	**Identity**: Azure AD	**Identity**: GCP IAM
Provision Operations	vSphere	→	vRealize Automation	Cloud Formation	Azure Resource Manager	Google Deployment Manager

Terraform
Introduction

Delivery Methods

On Premise

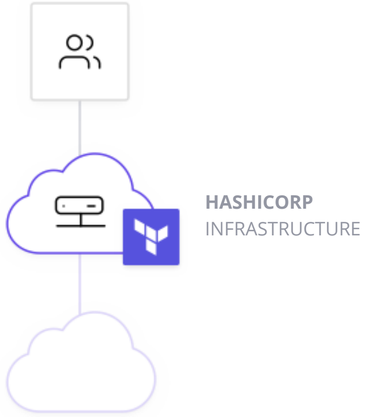
OPEN SOURCE



- No requirements for collaboration
- No requirements for central reusable configs
- No policy or governance requirements

Hosted SaaS

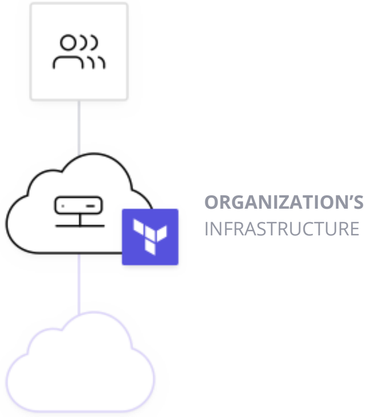
ENTERPRISE



- No special requirements*
- No internal resources to manage Terraform
- No policy or governance requirements

Private Install

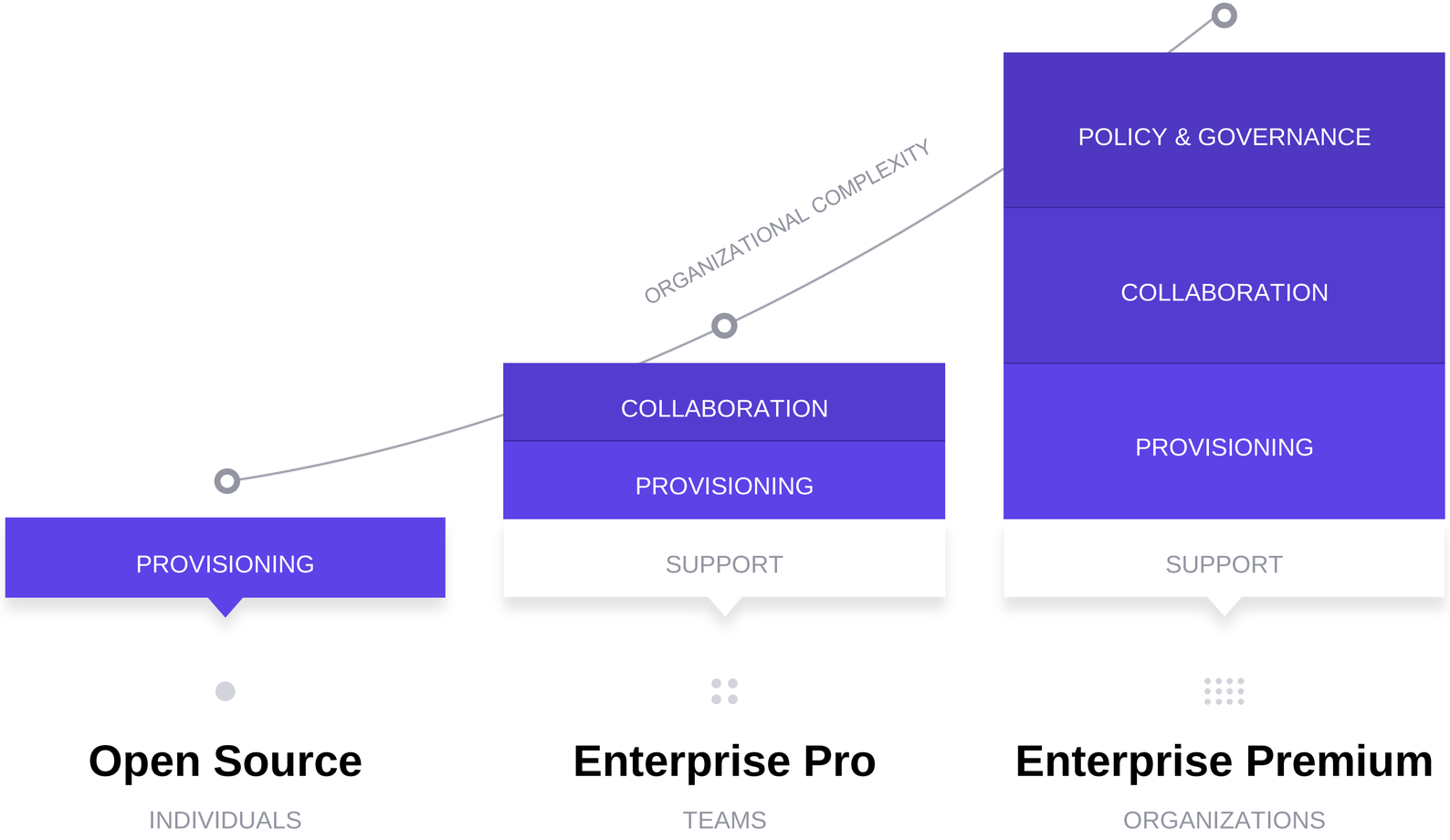
ENTERPRISE



- Infrastructure & applications behind a firewall
- Data sovereignty requirements
- Regulatory compliance requirements
- HA requirements
- Performance requirements

Terraform Packages

Enterprise products build on open source to address organizational complexity.



Compare Packages

Open Source Infrastructure as code, Multi-Cloud Management, Self-Service Infrastructure	
Infrastructure as Code (HCL)	✓
Workspaces	✓
Variables	✓
Runs (separate plan and apply)	✓
Resource Graph	✓
One Workflow to Provision Across Providers	✓
Providers (150+)	✓
Unique Resources (1000s)	✓
Modules	✓
Public Module Registry	✓

Enterprise Pro Collaboration and operations features for teams	
All **Open Source** Features	✓
VCS Integrated Connection	✓
Workspace Management	✓
Secure Variable Storage	✓
Remote Runs & State	✓
Team Management	✓
Private Module Registry	✓
Configuration Designer	✓
Full API Converage	✓
SaaS	✓

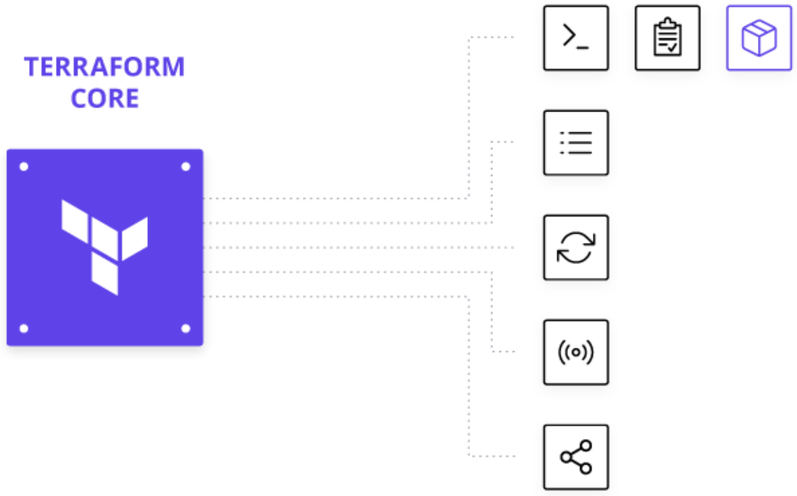
Enterprise Premium Governance and policy features for organizations	
All **Enterprise Pro** Features	✓
Sentinel Policy as Code Management	✓
Audit Logging	✓
SAML for SSO	✓
Private Install Options	✓

Terraform Core Engine

- OSS hosted at github.com/hashicorp/terraform
- The engine Terraform runs on
- Loads providers as needed

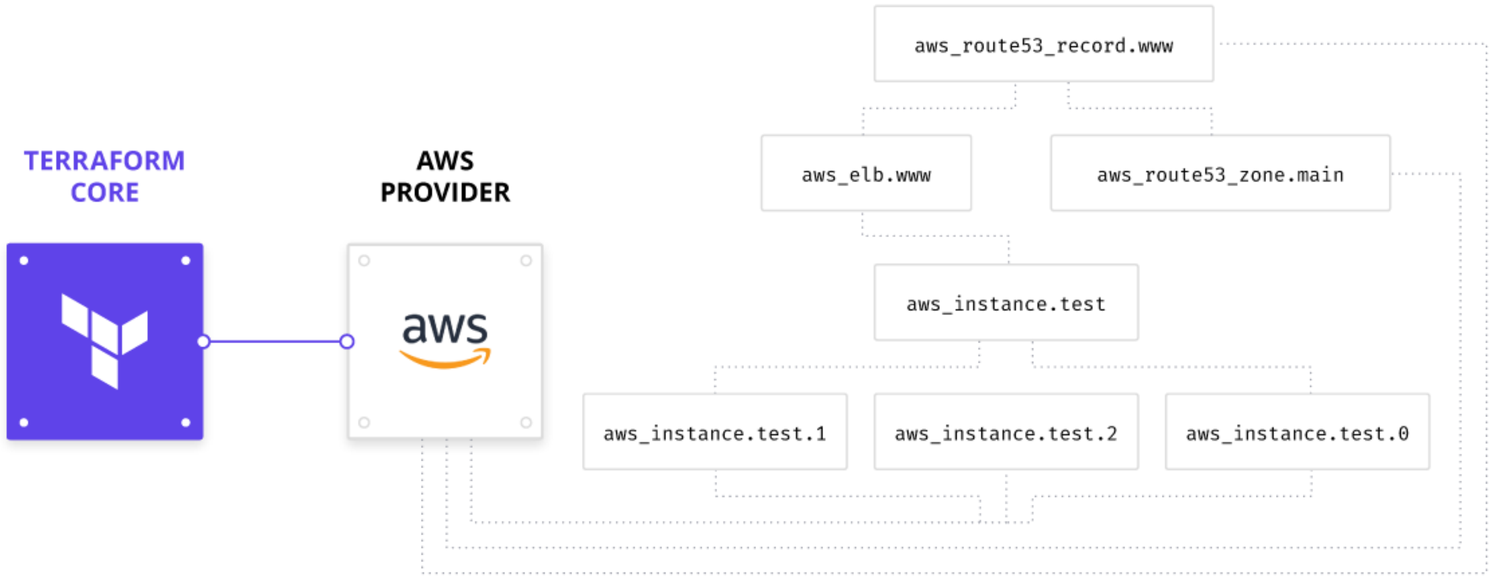
Responsible for:

- Reading and Interpolating configuration files and modules
- State Management
- Executing plan
- Communicating with providers
- Constructing resource graph



Resource Graph

- Safely provision and change infrastructure
- See planned infrastructure changes before execution
- No need to manually coordinate dependent resources

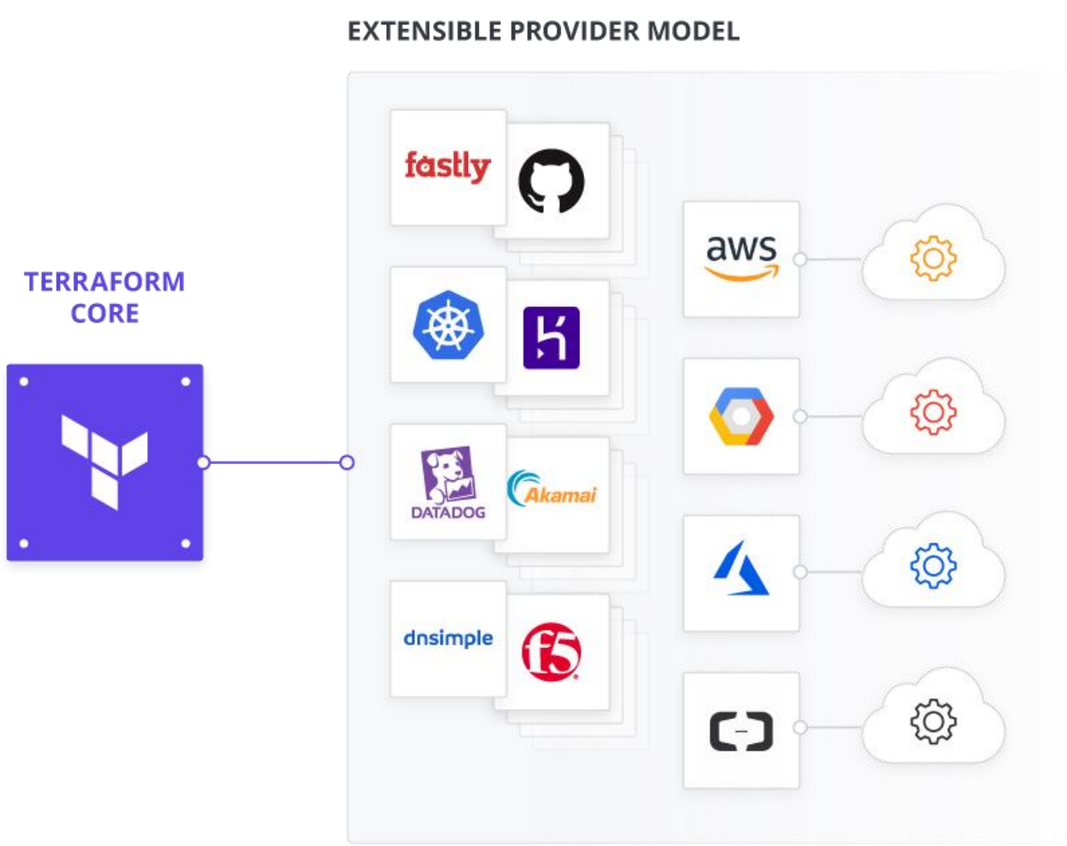


Provider Plugins

- Provider and Provisioner plugins expose implementation for specific services
- Offer extensible layer for 'Core' to learn how to talk to anything with an API without any upgrades

Responsible for:

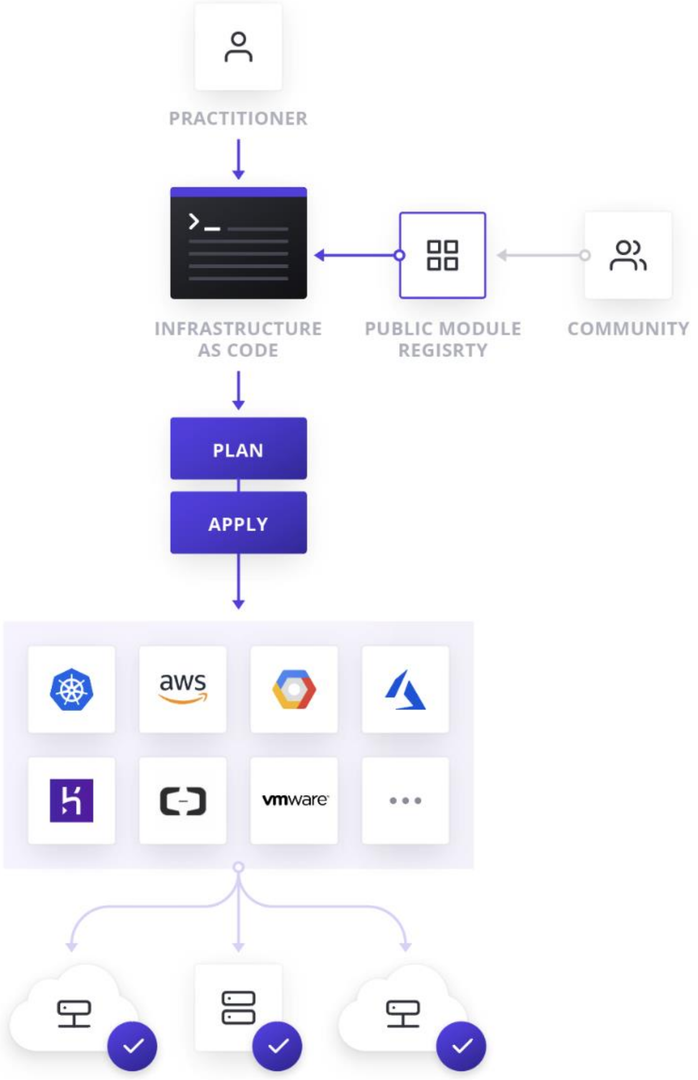
- Initializing libraries for API calls
- Authenticating with Provider
- Defining resources that map to services
- Executing commands or scripts on designated resources



Workflow for OSS

Terraform allows infrastructure to be expressed as code. The desired state is expressed in a simple human readable language. Terraform uses this language to provide an execution plan of changes, which can be reviewed for safety and then applied to make changes. Extensible providers allow Terraform to manage a broad range of resources, including hardware, IaaS, PaaS, and SaaS services.

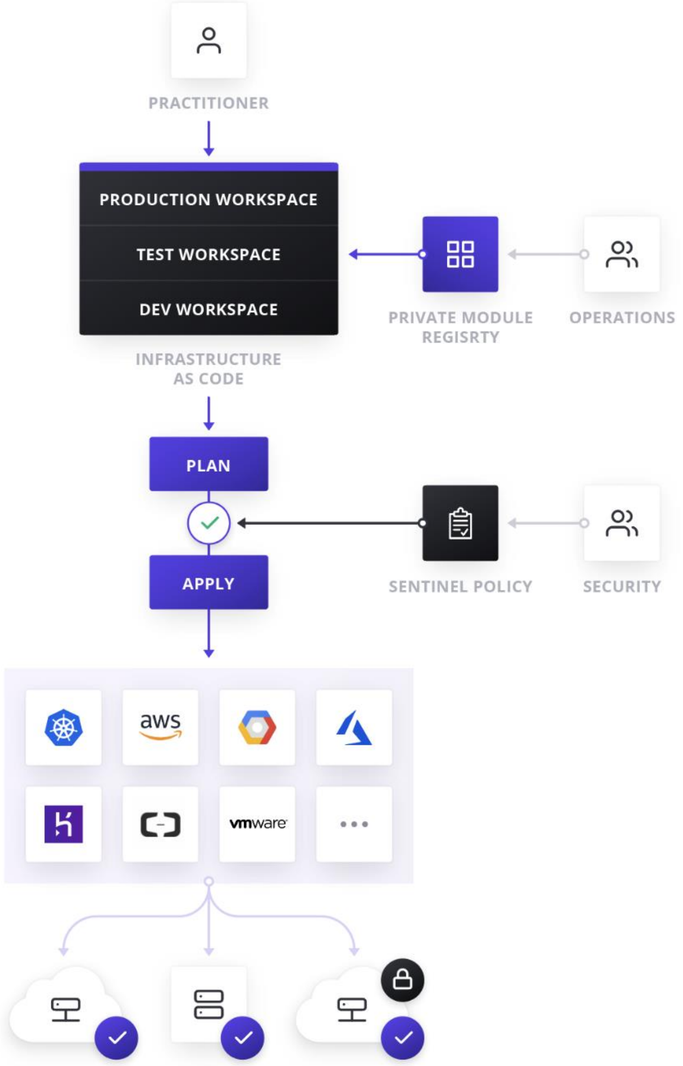
- Infrastructure as code
- 160+ providers
- Provision any infrastructure



Workflow for Enterprises

Terraform Enterprise provides collaboration, governance, and self-service workflows on top of the infrastructure as code provisioning from open source. Terraform Enterprise provides workspaces, modules, and other powerful constructs for teams working together to build infrastructure. Operators can package infrastructure as code into reusable modules enabling developers to quickly provision in a self-service fashion. Likewise, Policy as code and logging enable organizations to secure, govern, and audit their entire deployment.

- Collaborate on infrastructure as code
- Self-Service Infrastructure
- Policy and Governance



Terraform
Infrastructure as Code

provider "aws" { version = "~> 2.0" region = "us-east-1" }

Provider

The Terraform Provider is used to provider access to the Cloud Providers REST API

CODE EDITOR

```
# Azure Resource Manager Provider
provider "azurerm" {
  version = "=1.24.0"

  subscription_id = "67a8df85-7330-4c65-8343-d397f95ed522"
  client_id = "1b0fc6c9-9fb7-4fa6-add7-e14ef55d1df7"
  client_secret = "0bdee47d-069d-439e-9611-c596d8370f7f"
  tenant_id = "011e8fbe-2355-45b2-897c-a339efccfacd"
}

#-----------------------------------------------------------------#

# Amazon Web Services Provider
provider "aws" {
  version = "~> 2.0"
  region  = "us-east-1“
  access_key = "AKIAJJ66SVKCFCEH4UNT"
  secret_key = “AiDAKInilZQQR5vW2nwXQndk2QekimvHiiQBoqQn"
}
```

Input Variables

Input Variables provide a means to parameterise Terraform configuration and promote code reuse.

CODE EDITOR

```
# Optional Variable
variable "Location" {
  default = "Australia East"
}

# Required Variable
variable “environment" {}
```

Resources

The building blocks of your infrastructure. Common components such as Resource Groups, Compute or Container Instances.

CODE EDITOR

```
# Azure Resource Group
resource "azurerm_resource_group" "network" {
  name     = "${var.resource_group_name}"
  location = "${var.location}"
}

# Azure Virtual Network
resource "azurerm_virtual_network" "vnet" {
  name                = "${var.vnet_name}"
  location            = "${var.location}"
  address_space       = ["${var.address_space}"]
  resource_group_name = "${azurerm_resource_group.network.name}"
  dns_servers         = "${var.dns_servers}"
  tags                = "${var.tags}"
}
```

Resources

The building blocks of your infrastructure. Common components such as Resource Groups, Compute or Container Instances.

CODE EDITOR

```
# AWS Organization Account
resource "aws_organizations_account" "account" {
  name  = "AQIT"
  email = "user@aqit.io"
}

# AWS Virtual Private Connect
resource "aws_vpc" "main" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "dedicated"

  tags = {
    Name = "main"
  }
}
```

Provisioners

Used to execute scripts on a local or remote machine, bootstrap a resource, clean-up before destroy or run configuration management etc.

CODE EDITOR

```
# Terraform local-exec Provisioner
resource "aws_instance" "web" {
  # ...

  provisioner "local-exec" {
    command = "Get-Date > completed.txt"
    interpreter = ["PowerShell", "-Command"]
  }
}

# Terraform remote-exec Provisioner
resource "aws_instance" "web" {
  # ...

  provisioner "remote-exec" {
    inline = [
      "puppet apply",
      "consul join ${aws_instance.web.private_ip}",
    ]
  }
}
```

Modules

A Module is a container for multiple resources and provides a means of abstracting a monolithic template.

CODE EDITOR

```
# Open-Source Modules
module "network" {
    source              = "./modules/terraform-azurerm-network"
    location            = "${var.location}"
    resource_group_name = "${azurerm_resource_group.rg.name}"
}

# Enterprise Modules
module "consul" {
  source  = "app.terraform.io/AQIT/consul/azure"
  version = "0.4.4“

  cluster_name = "${var.cluster_name}-server"
  cluster_size = "${var.num_servers}"
  key_data = "${var.key_data}"
  allowed_ssh_cidr_blocks = "${var.allowed_ssh_cidr_blocks}"
  resource_group_name = "${var.resource_group_name}"
  storage_account_name = "${var.storage_account_name}"
  location = "${var.location}"

  # ...
}
```

Output Variables

Output Variables provide a means of querying and extracting resource attributes that can be consumed by other resources

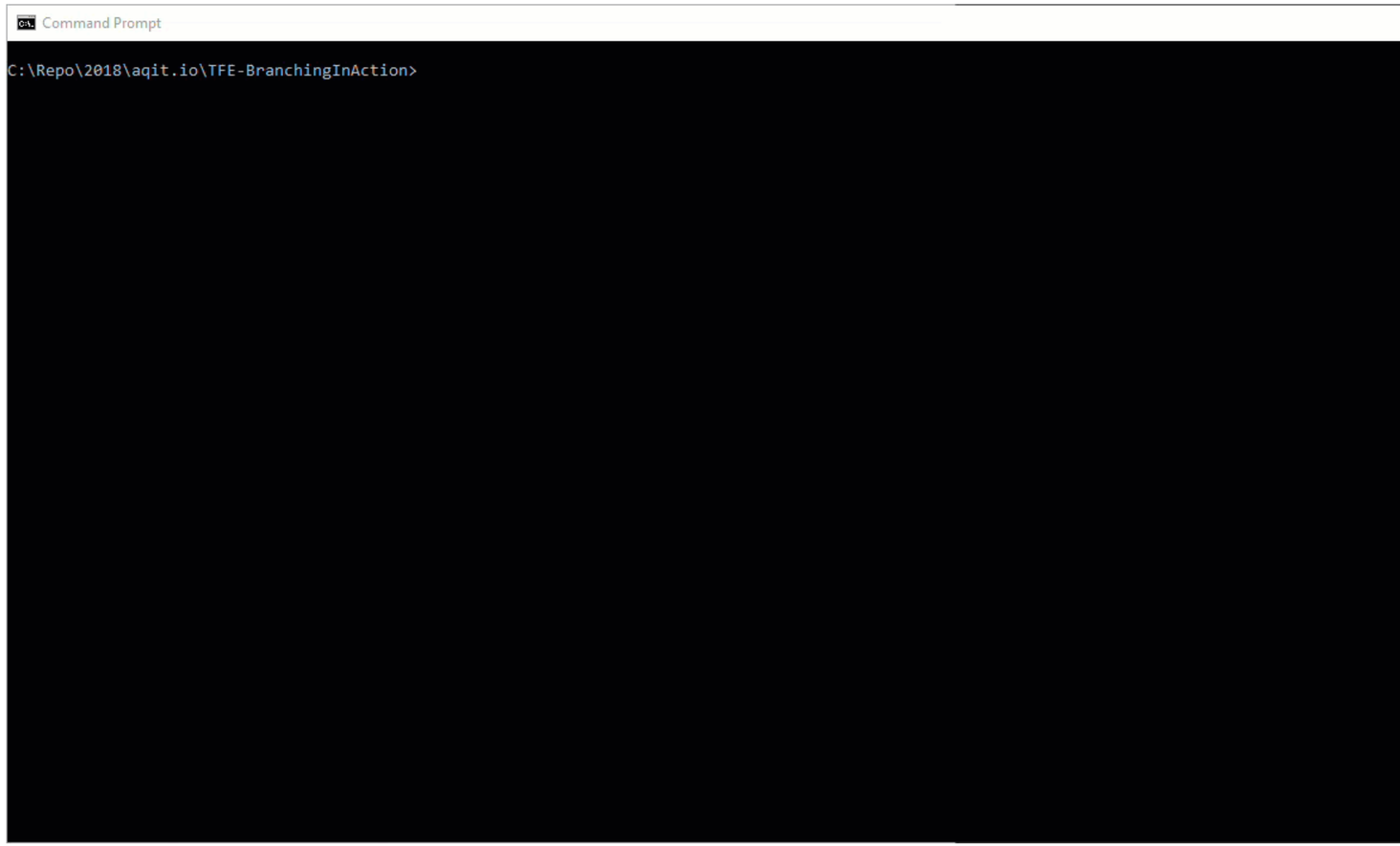
CODE EDITOR

```
# Azure Virtual Network Identifier
output "vnet_id" {
  description = "The id of the newly created vNet"
  value       = "${azurerm_virtual_network.vnet.id}"
}

Apply complete! Resources: 14 added, 0 changed, 0 destroyed.

Outputs:

vnet_id = /subscriptions/67a8df85-7330-4c65-8343-
d397f95ed522/resourceGroups/demo-terraform-
101/providers/Microsoft.Network/virtualNetworks/acctvnet
```

Terraform
OSS Workflow

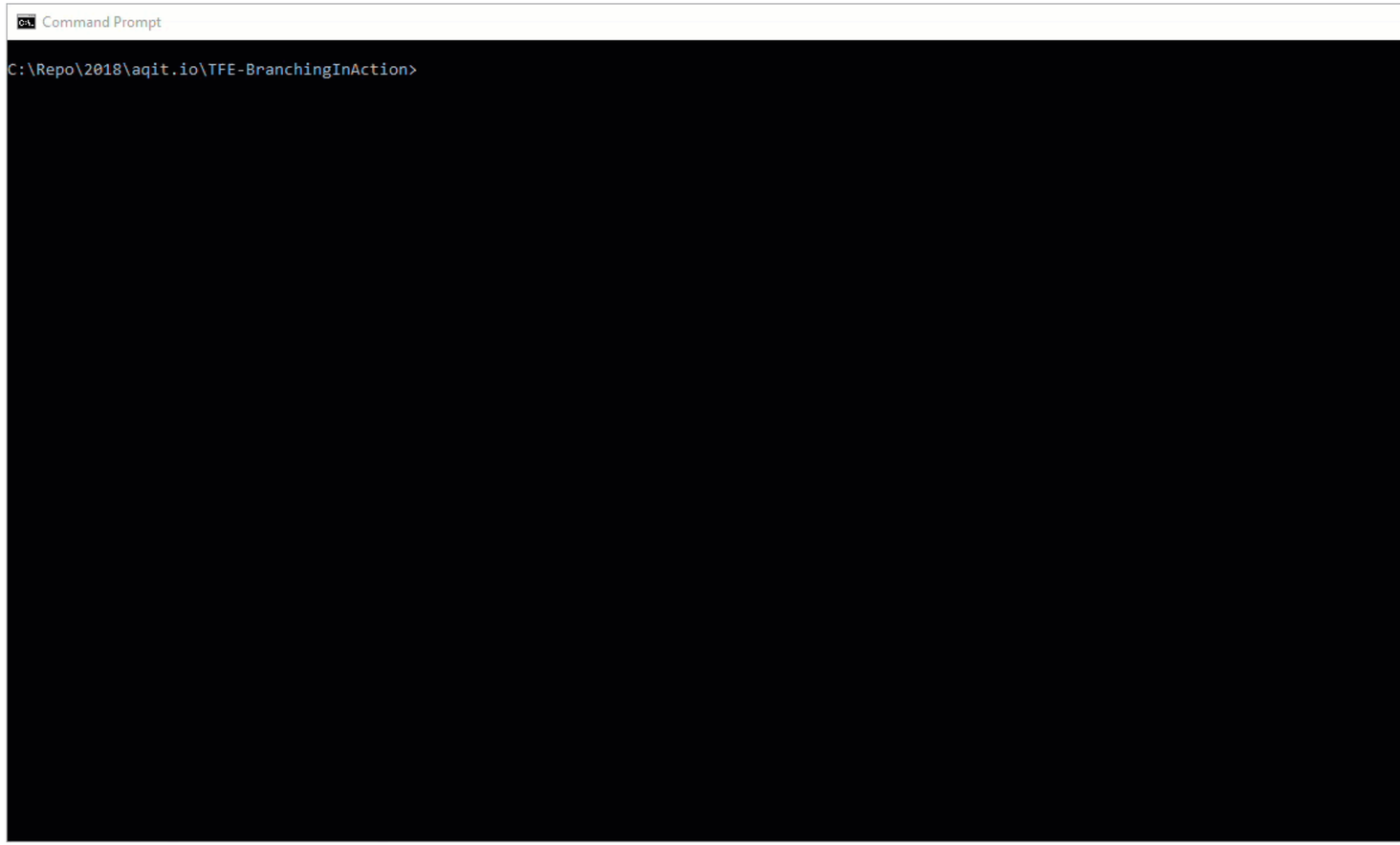
CLI

Terraform is a single command-line application: **terraform**. The CLI takes subcommands such as "help", "plan" or "plan".



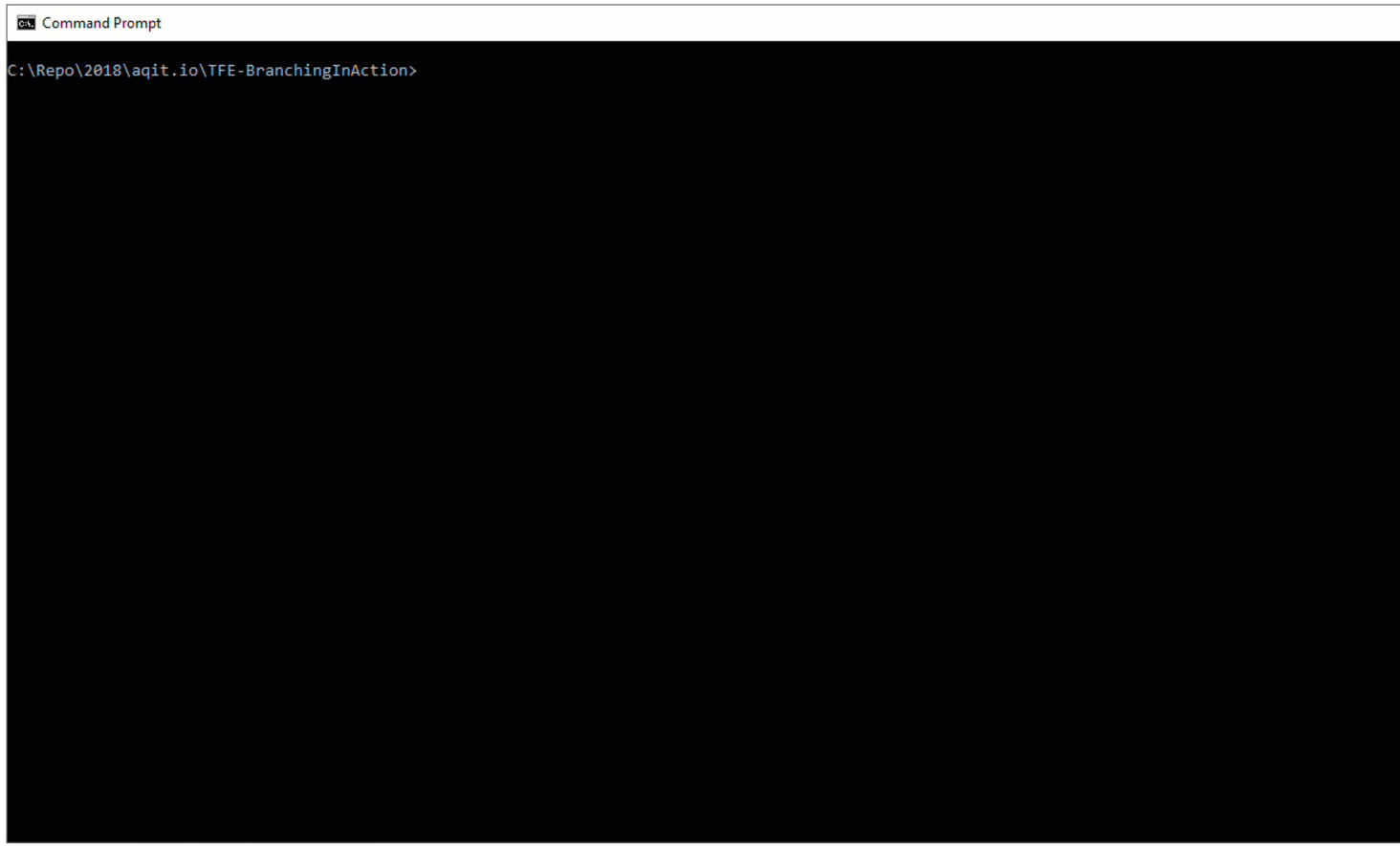
Validate

Performs a syntax check on all terraform files in the working directory. Will display an error if any of the files do not validate



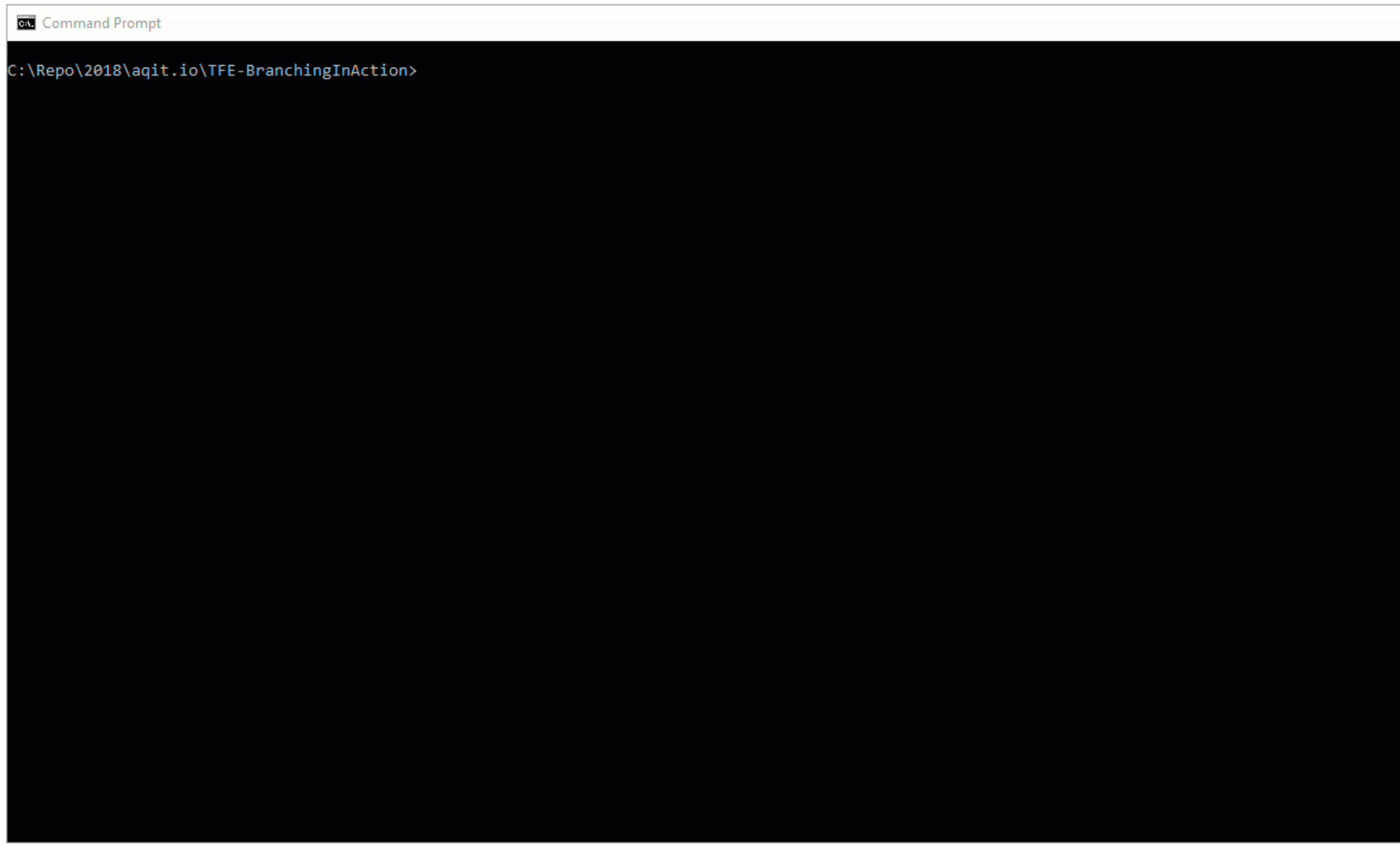
Plan

Performs a syntax check on all terraform files in the working directory. Will display an error if any of the files do not validate



Apply

Used to apply the changes required to reach the desired state of the configuration



Terraform **Enterprise**

Feature: **Workspaces**

Workspaces

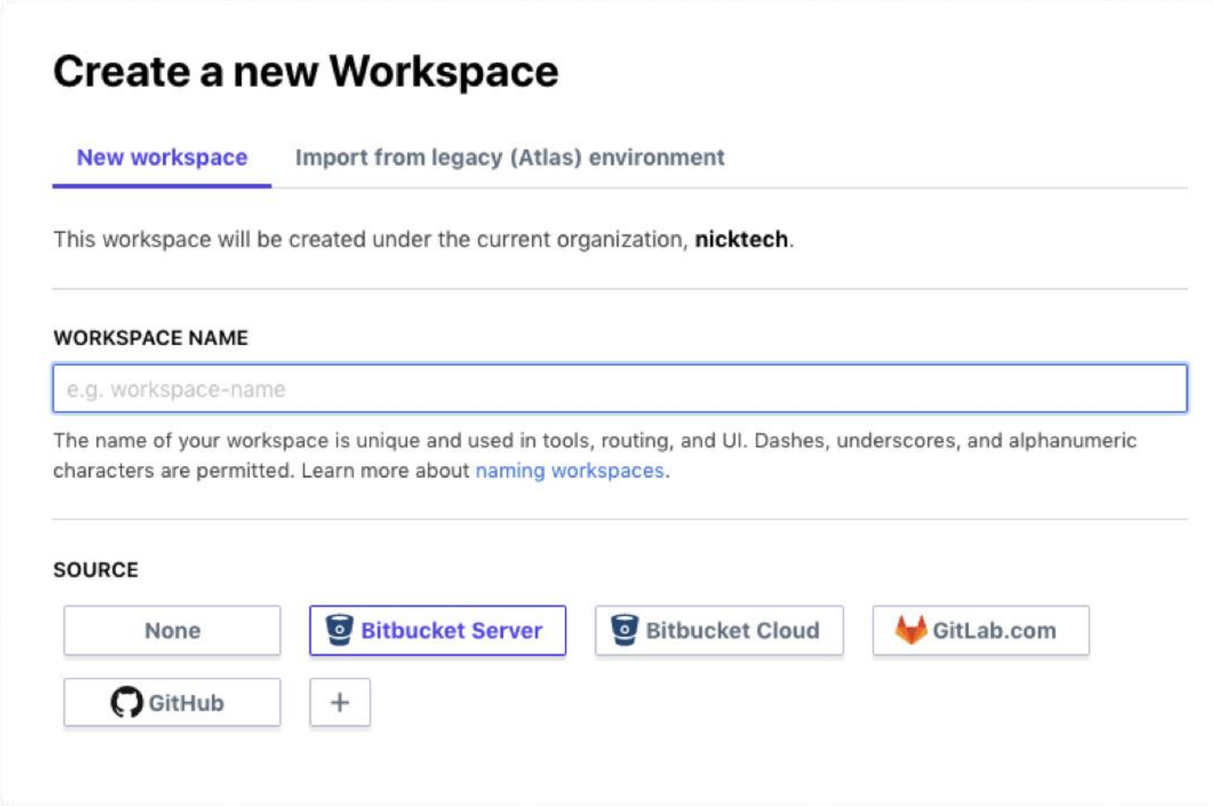
A workspace is a collection of everything Terraform needs to run: a configuration file, variables, and state data.
Workspaces offer powerful decomposition for monolithic configurations to match your organization and application structures.

OSS

- Workspaces are independent state files

ENTERPRISE

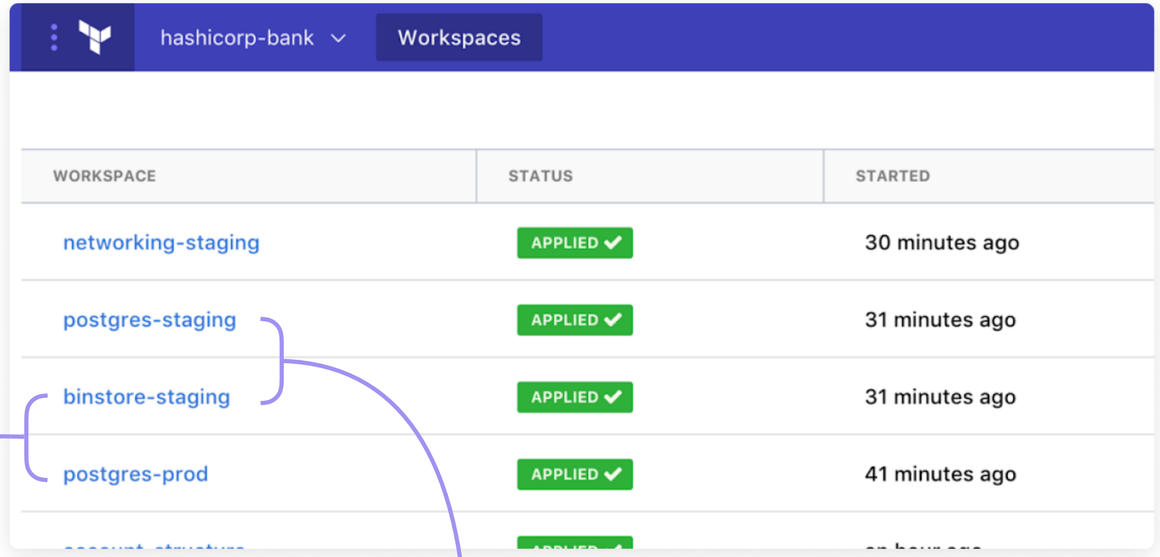
- Remote, persistent shared resources
- Access controls:
- Admin, Write, Read-Only, Plan-Only
- VCS Integration



Feature: **Workspaces**

Workspace Design

- One Workspace Per Environment Per Terraform Configuration
- Name your workspaces with both their component and their environment
- Use per-workspace access controls to delegate ownership of components and regulate code promotion across environments



Create a parallel, distinct copy of a set of infrastructure in order to test a set of changes before modifying the main production infrastructure.

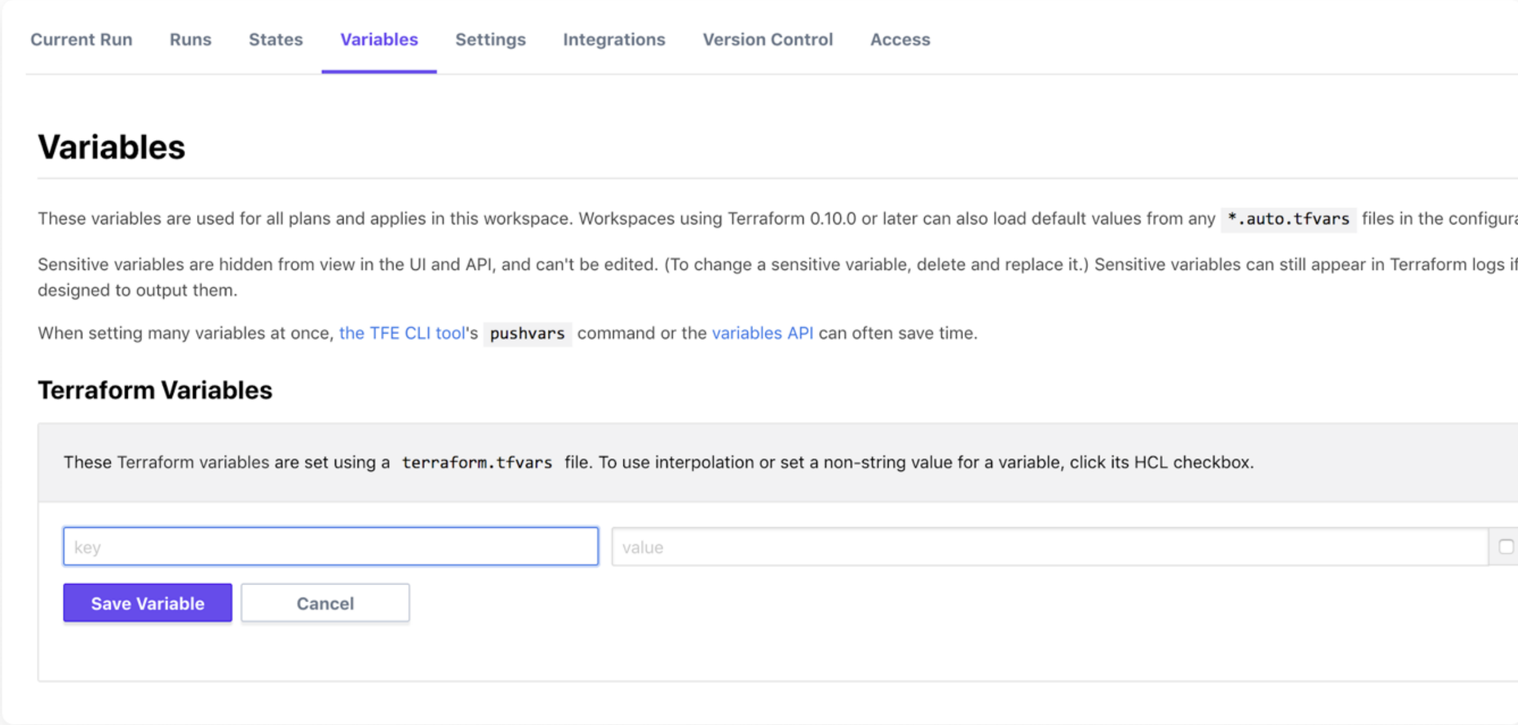
Use multiple separate Terraform configurations that correspond with suitable architectural boundaries within the system so that different components can be managed separately and, if appropriate, by distinct teams.

Feature: **Variables**

Variables

By writing infrastructure as code with variables, operators and developers can easily customize infrastructure as code without opening a text editor.

- Input variable parameters for Terraform configurations or modules

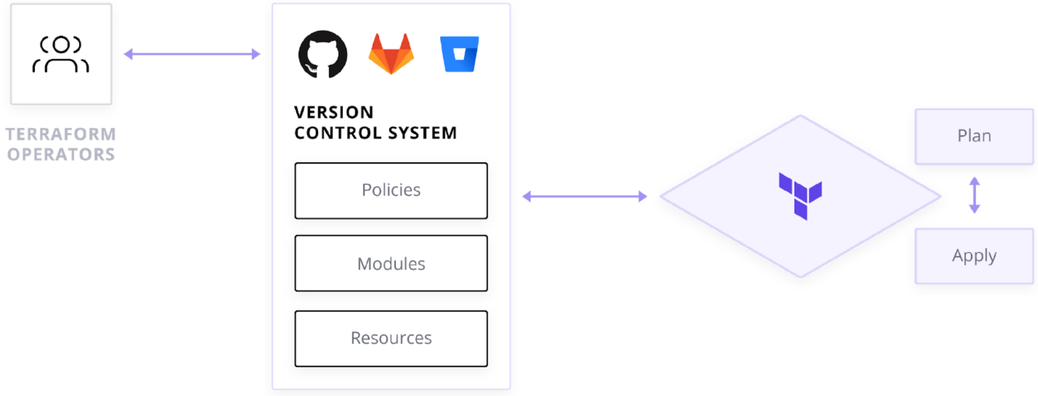


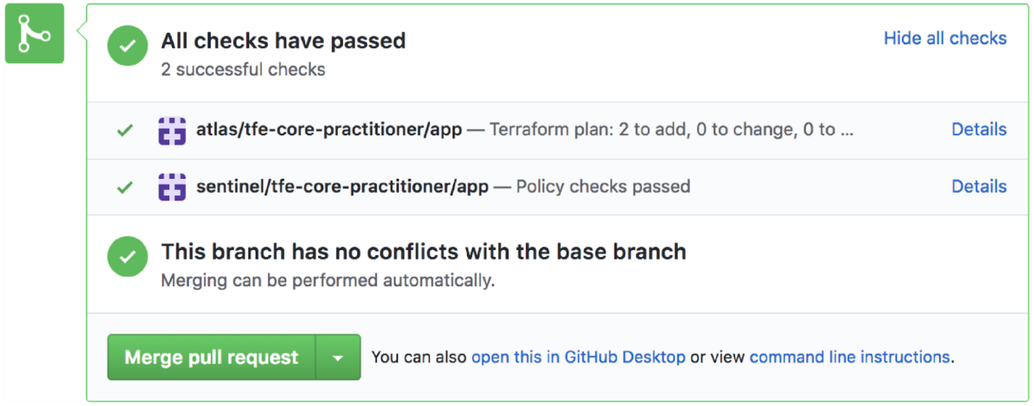
Feature: **VCS Connection**

VCS Connection

Terraform connects to the major VCS providers allowing for automated versioning and running of configuration files.

- Integrates into existing VCS workflow
- Automatically trigger runs and policy checks upon pull requests
- Support for BitBucket, Gitlab, and Github
- Default Workspaces to VCS pairing





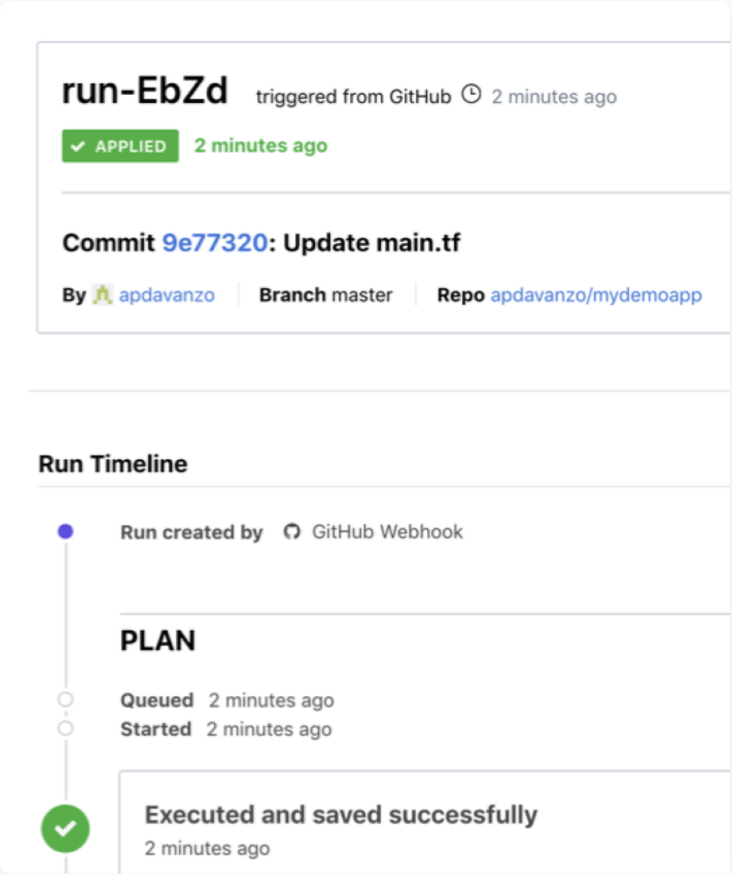
Feature: **Runs and State**

Remote Operations

- Trigger Terraform runs remotely:
  - Pull request in VCS
  - API call in CI/CD pipeline
  - CLI
- Automatically coordination of runs by multiple users
- Maintain core workflow from OSS

ENTERPRISE

- Enhanced remote runs and state storage empowering collaboration
- Runs triggered from local CLI are shown in UI



Feature: **Policy as Code**

Policy as Code

With programmatic policy as code, custom governance can be enforced at the same rate as infrastructure is provisioned.

- Codifying policies automates guardrails around provisioning
- Policy checks built into the provisioning workflow
- Use policy to enforce best-practices, security measures, or compliance

Enforcement Levels

- *Advisory*: Warns when a policy breaks
- *Soft Mandatory*: Provision needs to override policy to break it
- *Hard Mandatory*: Provisioning not allowed to break policy

RESTRICTING MACHINE TYPE IN GCP

```
allowed_machine_types = [
  "n1-standard-1",
  "n1-standard-2",
  "n1-standard-4",
]
```

Feature: **Module Registry**

Module Registry

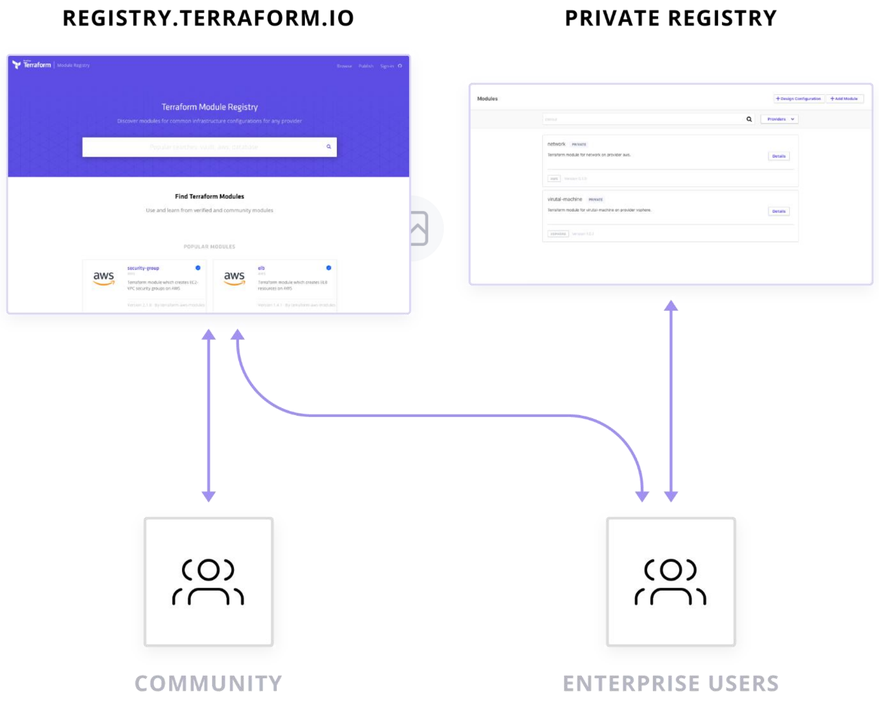
The public module registry offers the large community a repository to store and share modules.

- 700+ Modules
- Build & publish modules for general consumption

Private Module Registry

The private module registry, built into Terraform Enterprise, offers organizations a private repository to store and share modules internally.

- Modules can be created with best practices and operational efficiency built-in, i.e. tagging resources, setting TTL's, etc.

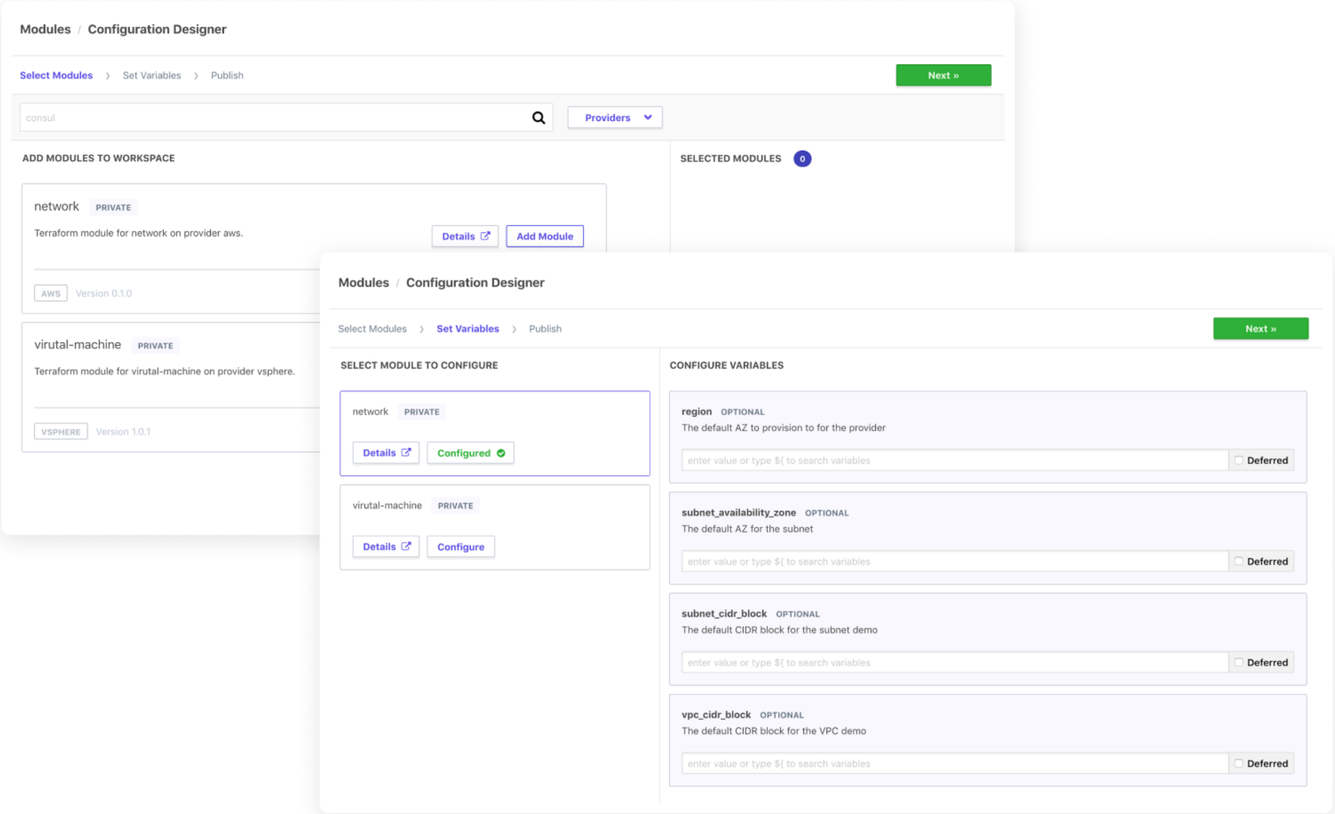


Feature: **Configuration Designer**

Configuration Designer

Configuration designer allows users to choose modules from the public or private registry and customize them to their needs through a graphical user interface.

- Ability to discover, combine, and assign variables to build custom infrastructure as code
- Effectively use Infrastructure as Code without deep provider knowledge



Feature: **Full API**

Full API

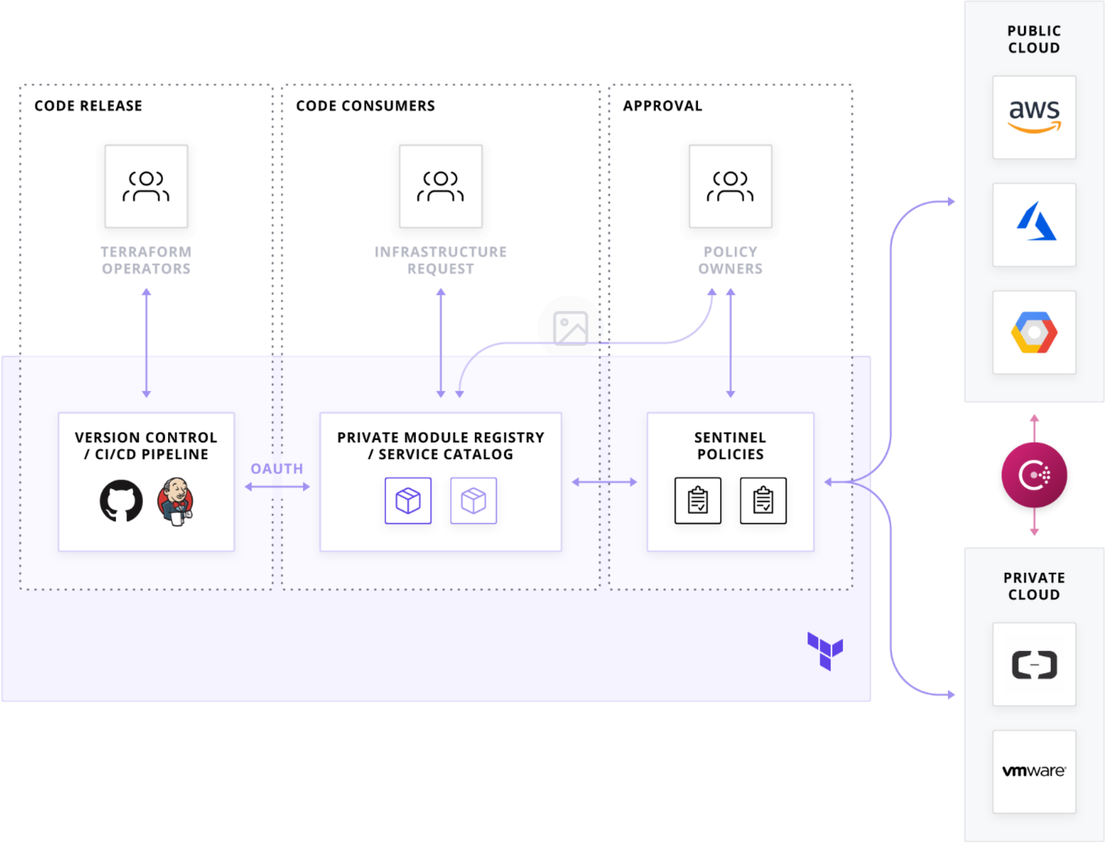
Terraform can be fully operated via API allowing organizations to easily integrate it into their existing application delivery pipelines.

Integrate with existing workflow to minimize process changes:

- CI/CD pipeline integration
- Provision teams and workspaces from Service Now

Use JSON-API endpoints to manage all resources and perform operations, including:

- Environment Variables
- Trigger Plan/apply
- Retrieve state information

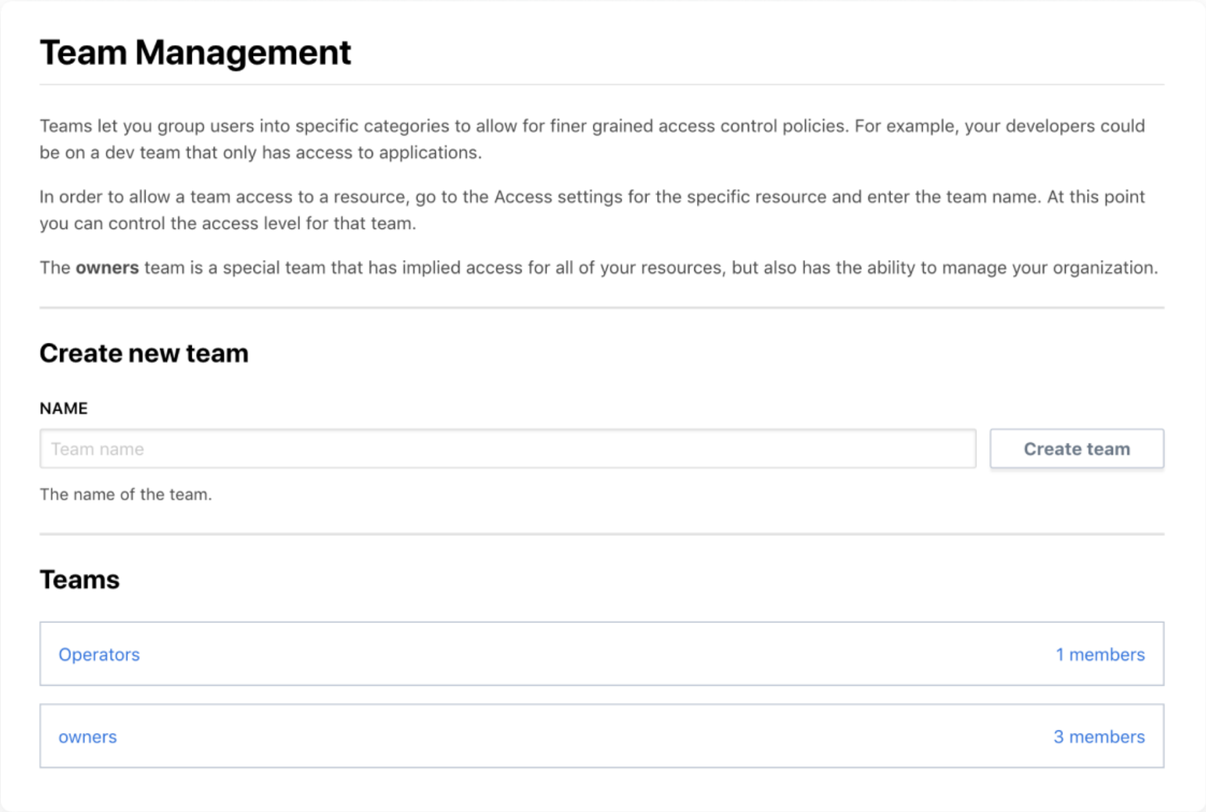


Feature: **Team Management & SSO**

Team Management

Terraform allows organizations to define roles and teams that have access to certain workspaces and environments.

- Full role based access control
- Manage organizations, teams, and privileges of individual users

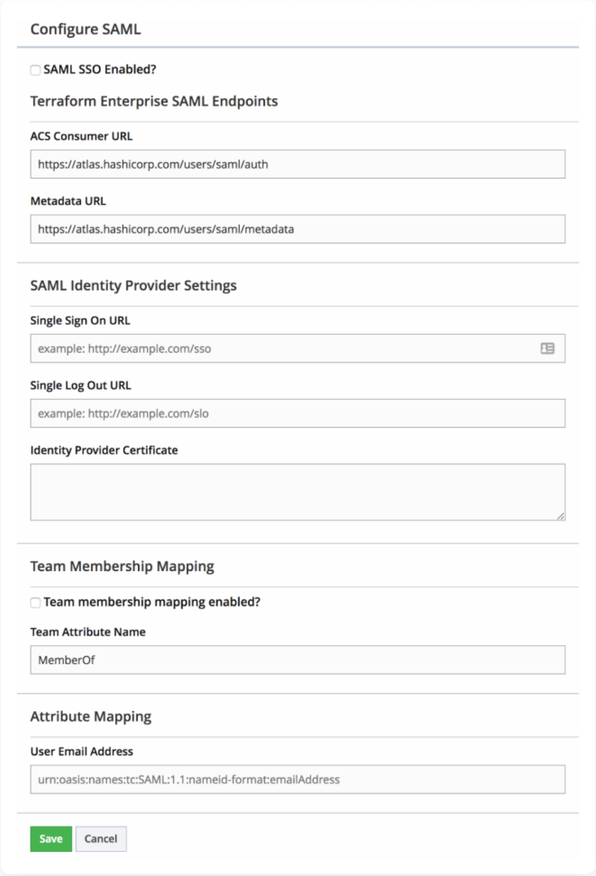


Feature: **Team Management & SSO**

SAML SSO Support

With SAML support Terraform integrates with your existing identity provider to authenticate and authorize users.

- Use existing identity provider to authenticate and authorize access to Terraform Enterprise
- Single sign-on capability eliminates all passwords and instead uses standard cryptography and digital signatures to pass a secure sign-in token from an identity provider to Terraform Enterprise
- Authenticate and authorize users

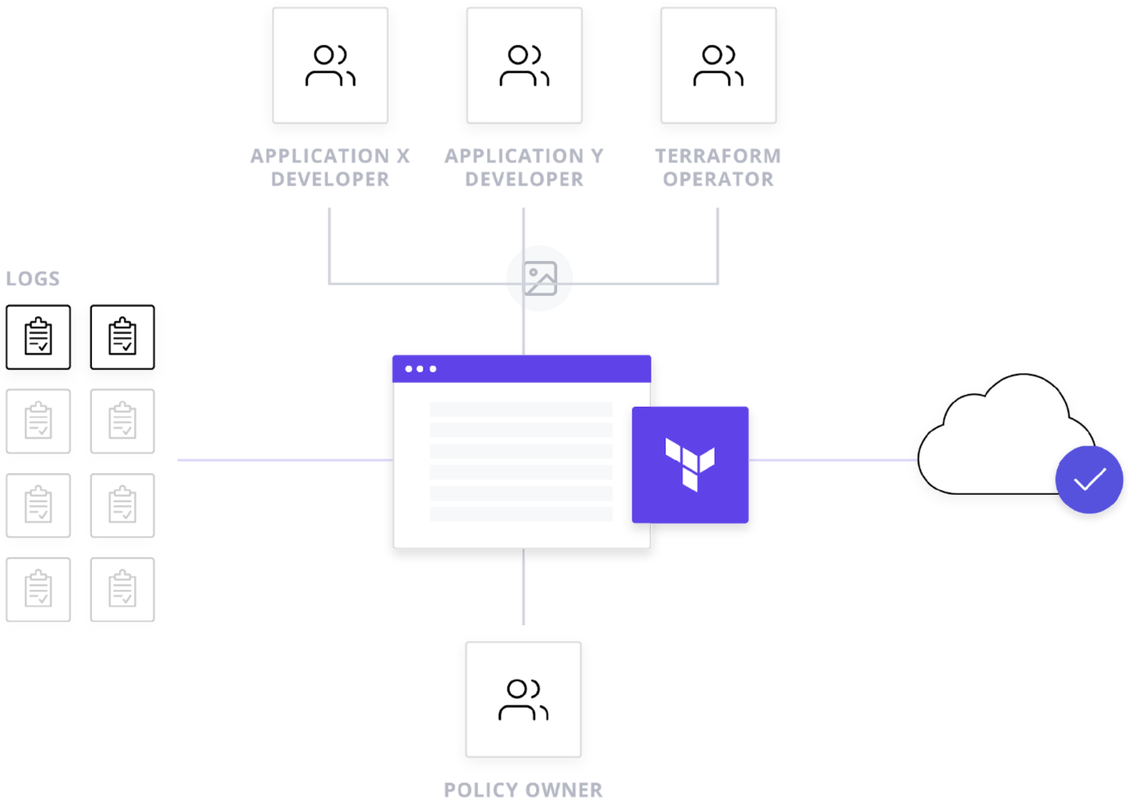


Feature: **Audit & Administrate**

Audit Logs

Terraform Audit Logs provide a trail of every API call made for every provider and service Terraform has provisioned.

- Track operations and identity made across the organization to gain insight into past and present configurations
- Identity action and resource
- Logs include settings, configuration changes, executions, environment updates
- Logs detail every API call made at each plan/apply of Terraform

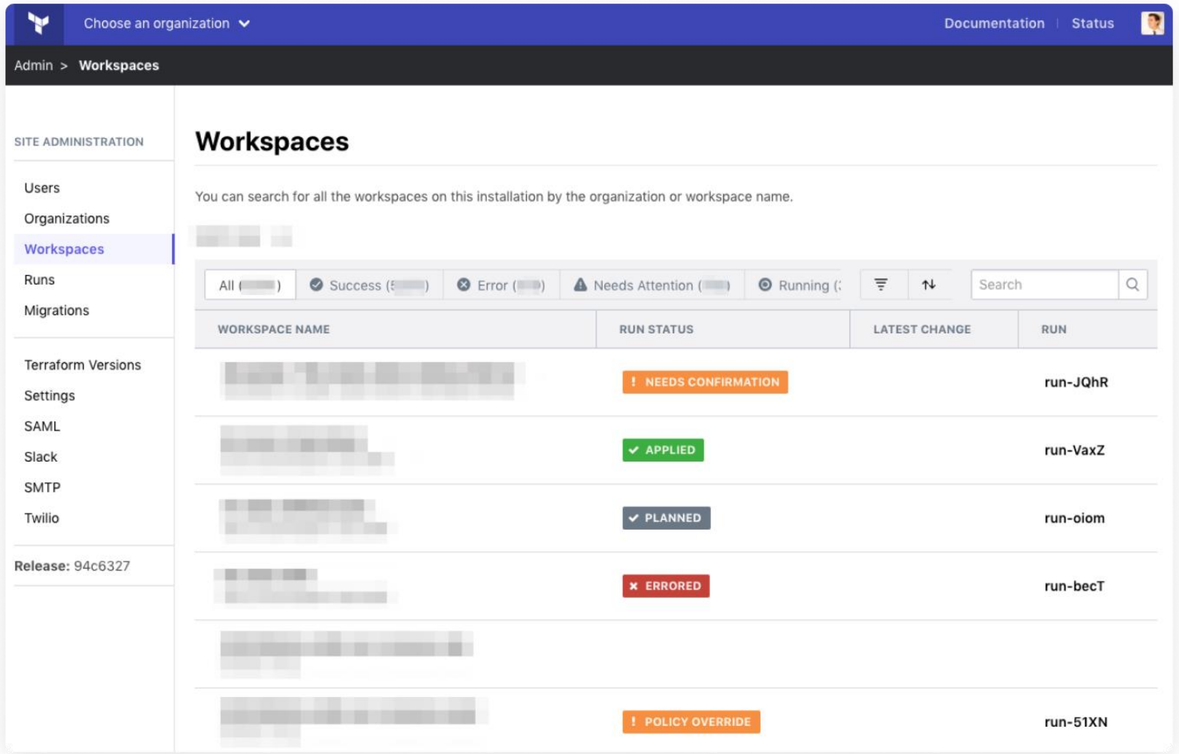


Feature: **Audit & Administrate**

Site Admin

The site administrator view in Terraform Enterprise provides a single pane of glass to see the entirety of infrastructure provisioned with Terraform.

- Per workspace run, configuration, and state history
- Site admin gives view of every organization, workspace, user, and run
- Useful for auditing as well as troubleshooting



Terraform
Adoption

About Terraform

2014 Product Launch

1200+ Contributors

150+ Providers

12.5k+ GitHub Stars

20k+ Downloads Weekly

250+ Customers Worldwide

Get Started

0.12.2

https://www.terraform.io/downloads

Free-ish TFE

https://app.terraform.io/signup/account

HCL 2.0

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

Skill Up!

https://learn.hashicorp.com/terraform/

Organizations that trust Terraform.

SAP Ariba

cruise

BARCLAYS

petco

PROGRESSIVE

ripple

UPSIDE

SPACEFLIGHT

ellucian

gaia

itv

Thank you

hello@hashicorp.com

www.hashicorp.com