

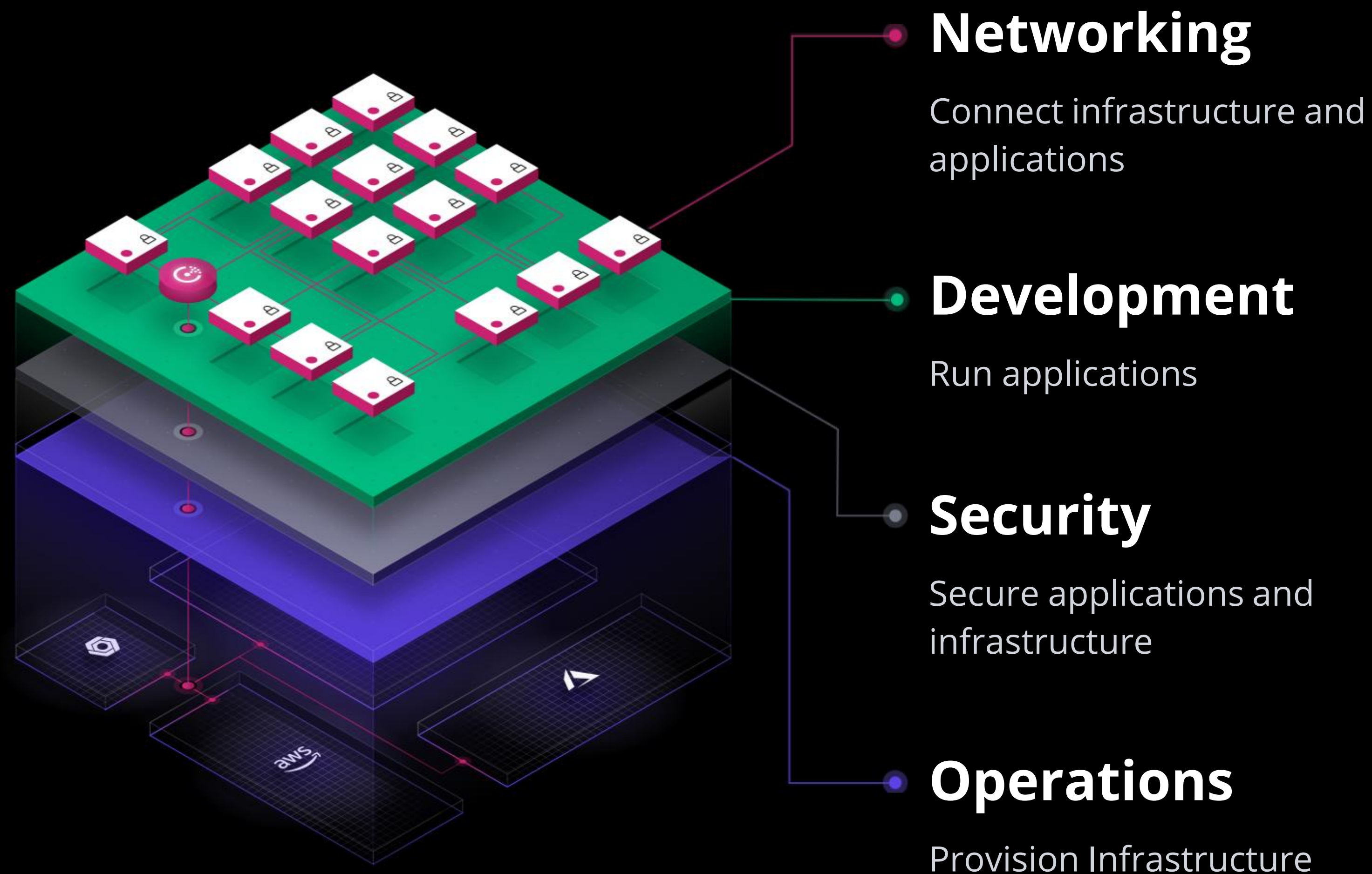


**Multi cloud.
Hybrid cloud.
Any infrastructure.
Anywhere.**

Enabling safe and efficient provisioning and management.

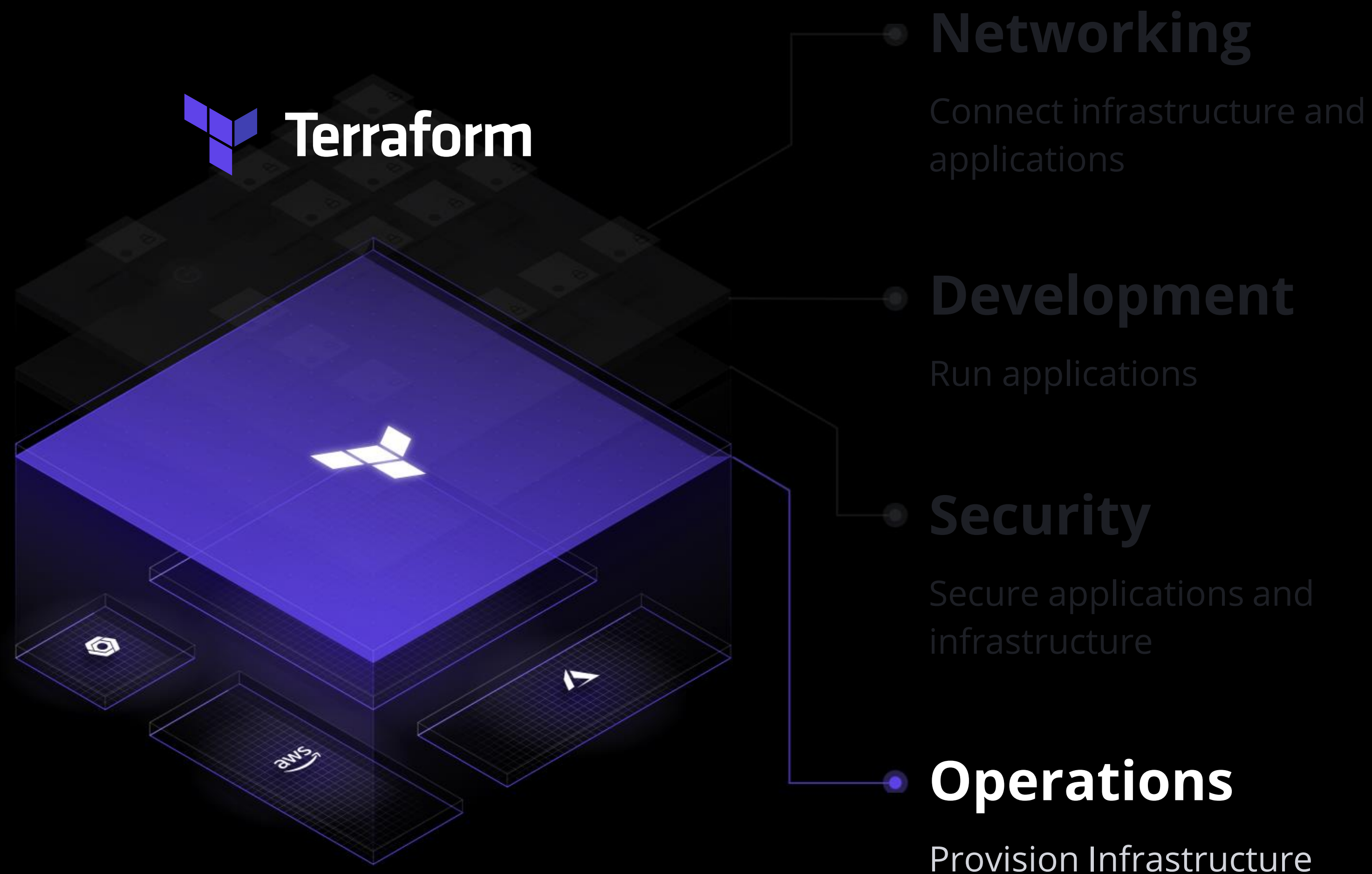


The 4 essential elements of dynamic infrastructure





The 4 essential elements of dynamic infrastructure





The Transition to Multi-Cloud

The Transition to Cloud and Multi-Cloud

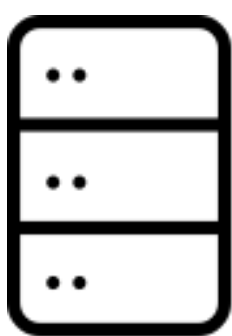


Traditional Datacenter

“Static”



Dedicated
Infrastructure



Private
Cloud

Modern Datacenter

“Dynamic”



AWS

+

Azure

+

GCP

+

...

SYSTEMS OF RECORD



SYSTEMS OF ENGAGEMENT

“Tickets-based”

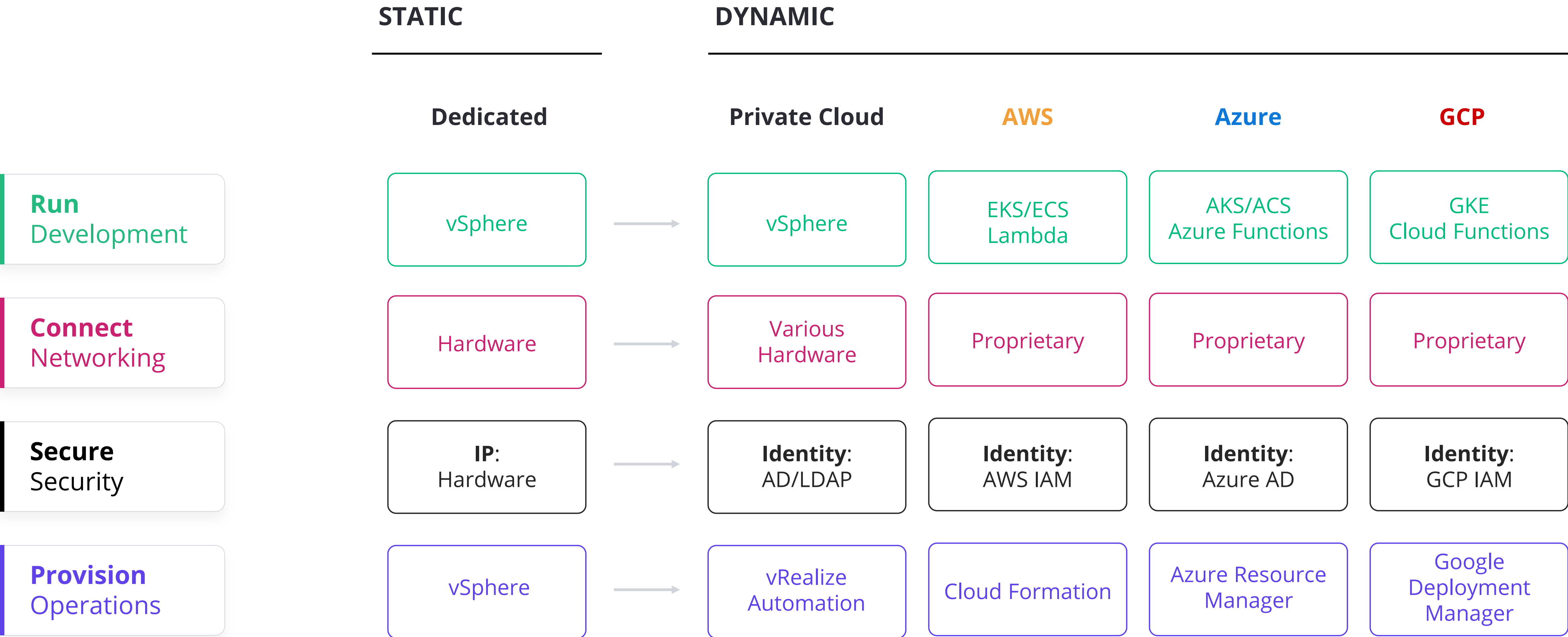
“Self service”

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





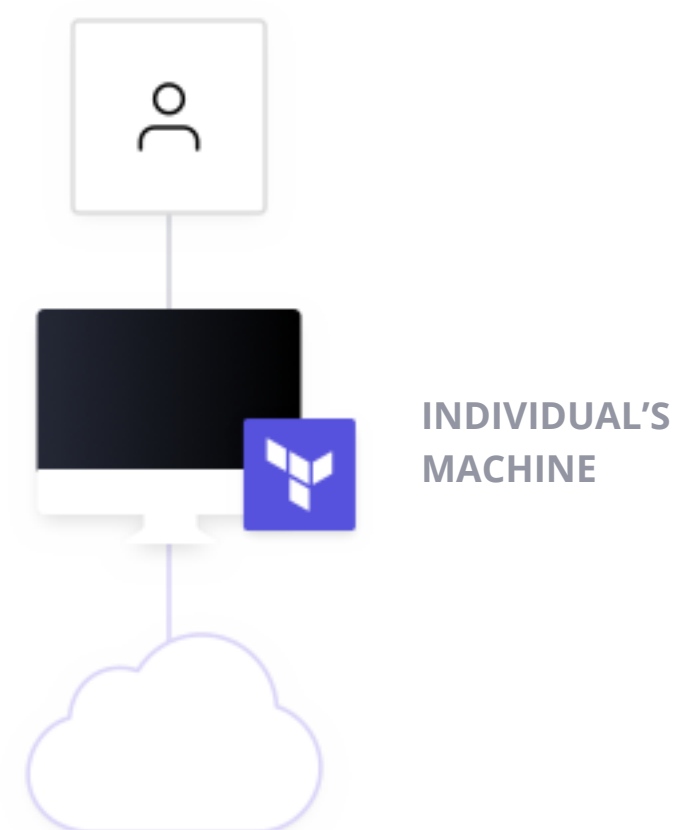
Terraform Introduction



Delivery Methods

On Premise

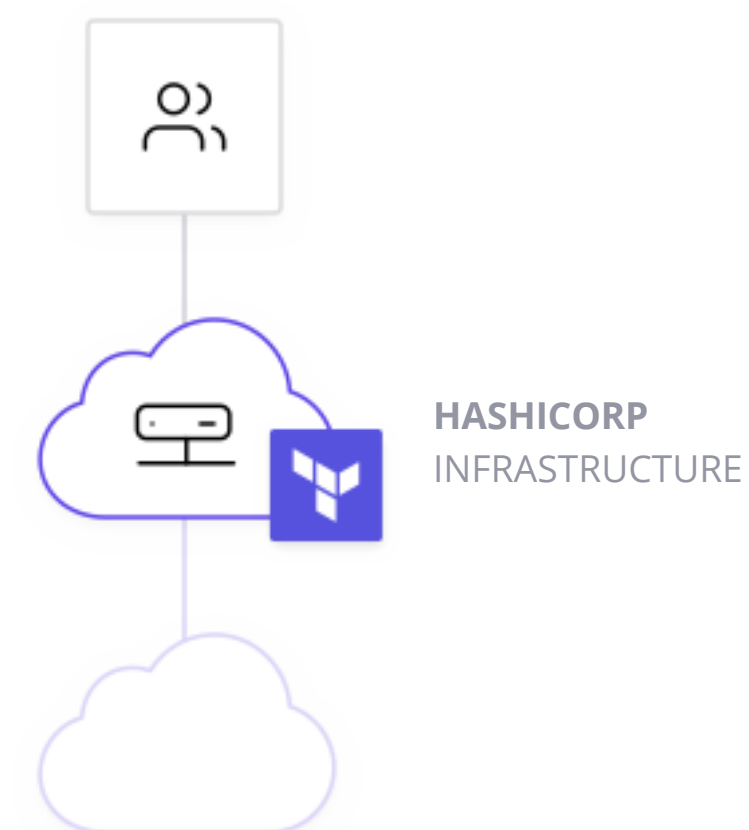
OPEN SOURCE



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

Hosted SaaS

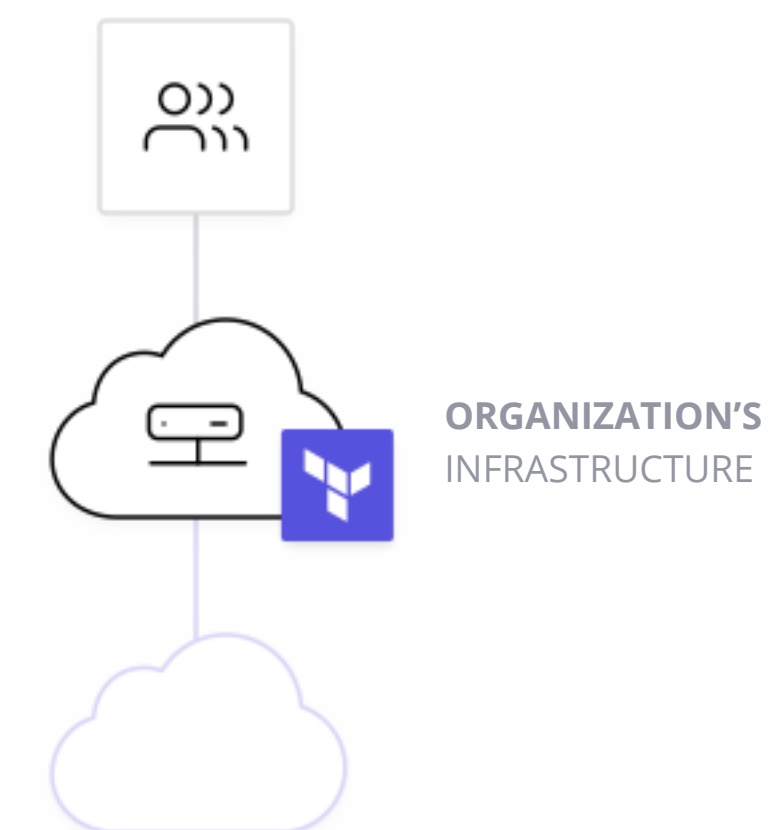
ENTERPRISE



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

Private Install

ENTERPRISE

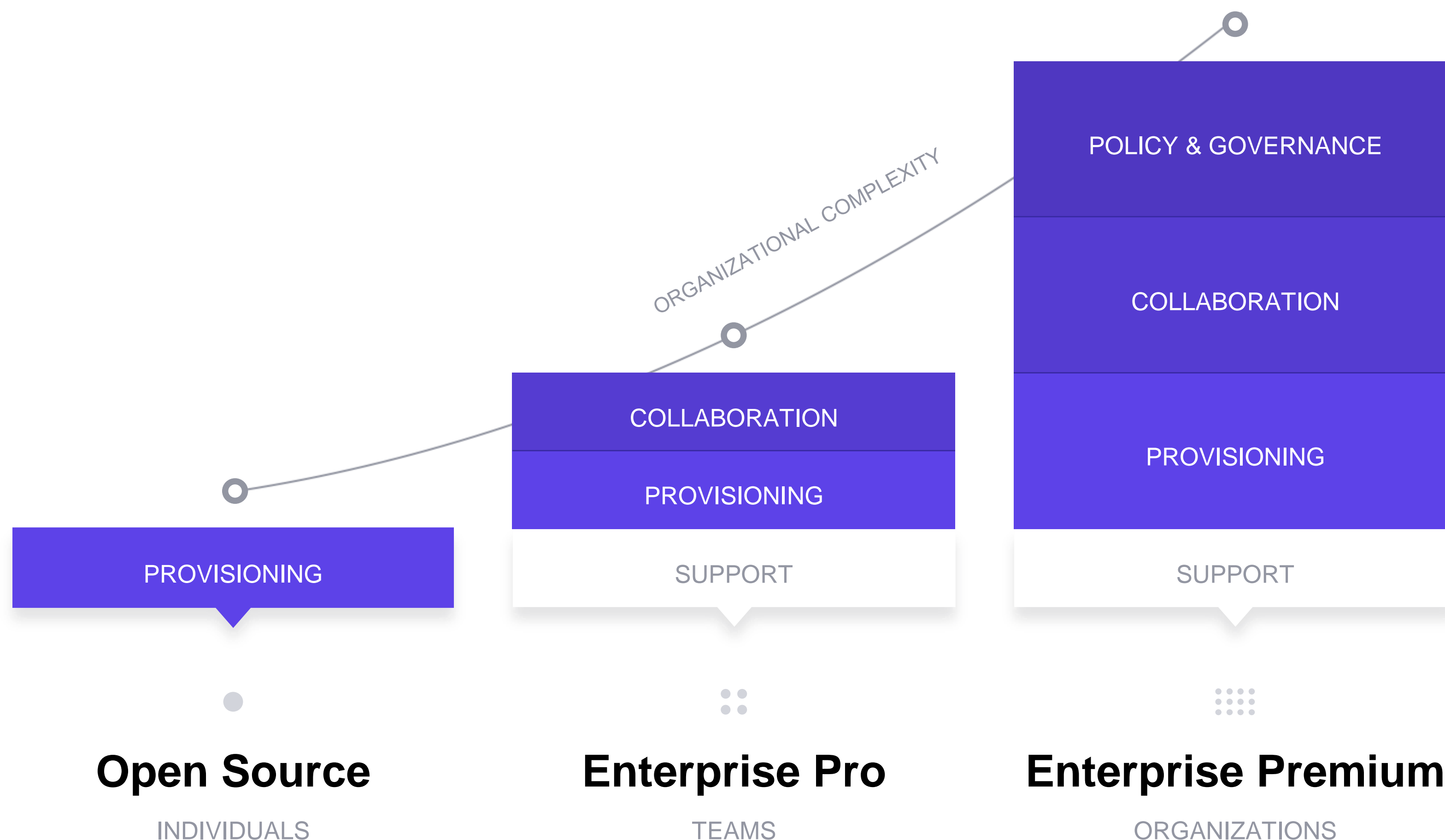


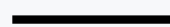
- Infrastructure & applications behind a firewall
- Data sovereignty requirements
- Regulatory compliance requirements
- HA requirements
- Performance requirements



Terraform Packages

Enterprise products build on open source to address organizational complexity.





Compare Packages

Open Source		Enterprise Pro		Enterprise Premium	
Infrastructure as code, Multi-Cloud Management, Self-Service Infrastructure		Collaboration and operations features for teams		Governance and policy features for organizations	
Infrastructure as Code (HCL)	✓	All Open Source Features	✓	All Enterprise Pro Features	✓
Workspaces	✓	VCS Integrated Connection	✓	Sentinel Policy as Code Management	✓
Variables	✓	Workspace Management	✓	Audit Logging	✓
Runs (separate plan and apply)	✓	Secure Variable Storage	✓	SAML for SSO	✓
Resource Graph	✓	Remote Runs & State	✓	Private Install Options	✓
One Workflow to Provision Across Providers	✓	Team Management	✓		
Providers (150+)	✓	Private Module Registry	✓		
Unique Resources (1000s)	✓	Configuration Designer	✓		
Modules	✓	Full API Converage	✓		
Public Module Registry	✓	SaaS	✓		

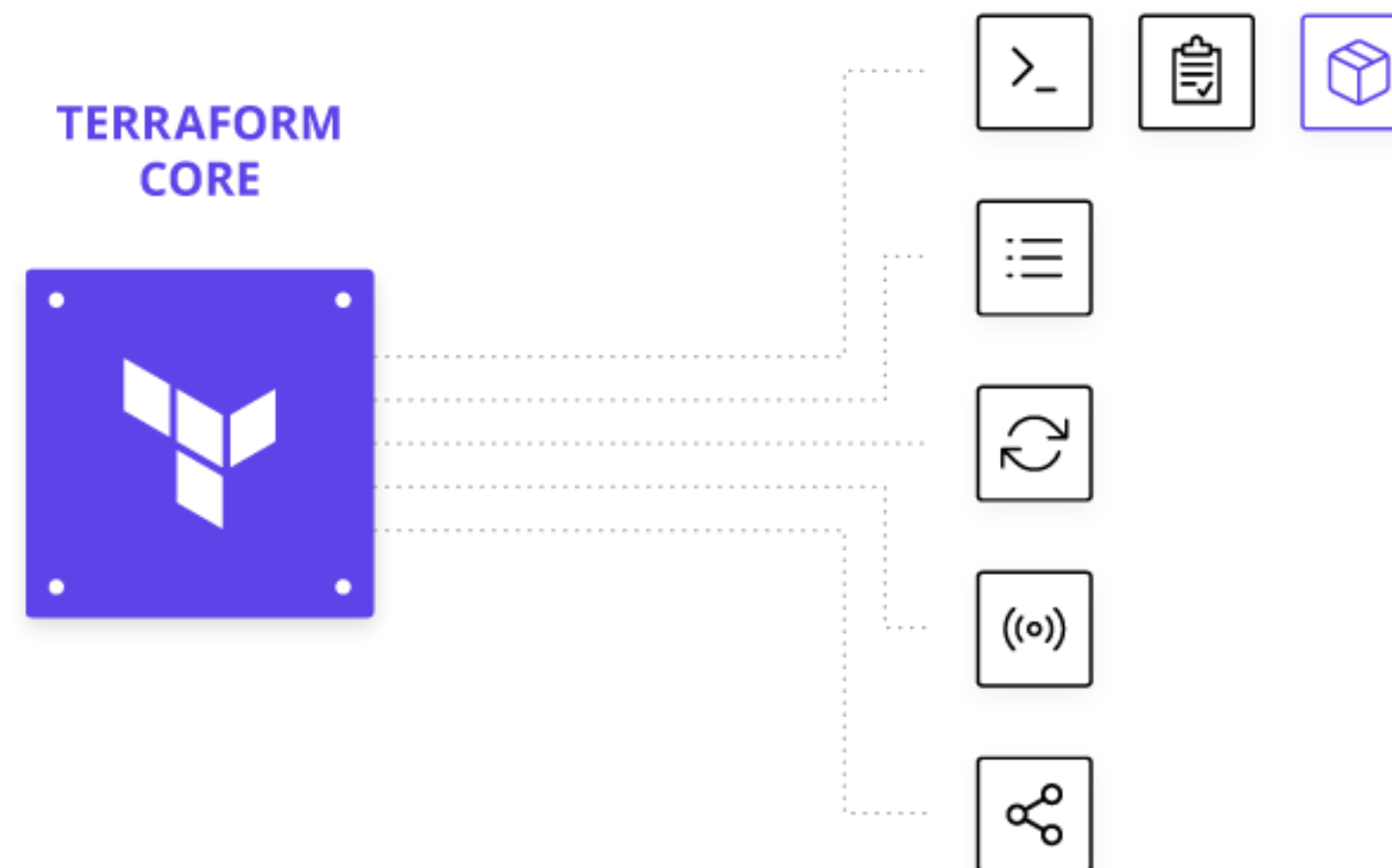
Terraform Core Engine



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

Responsible for:

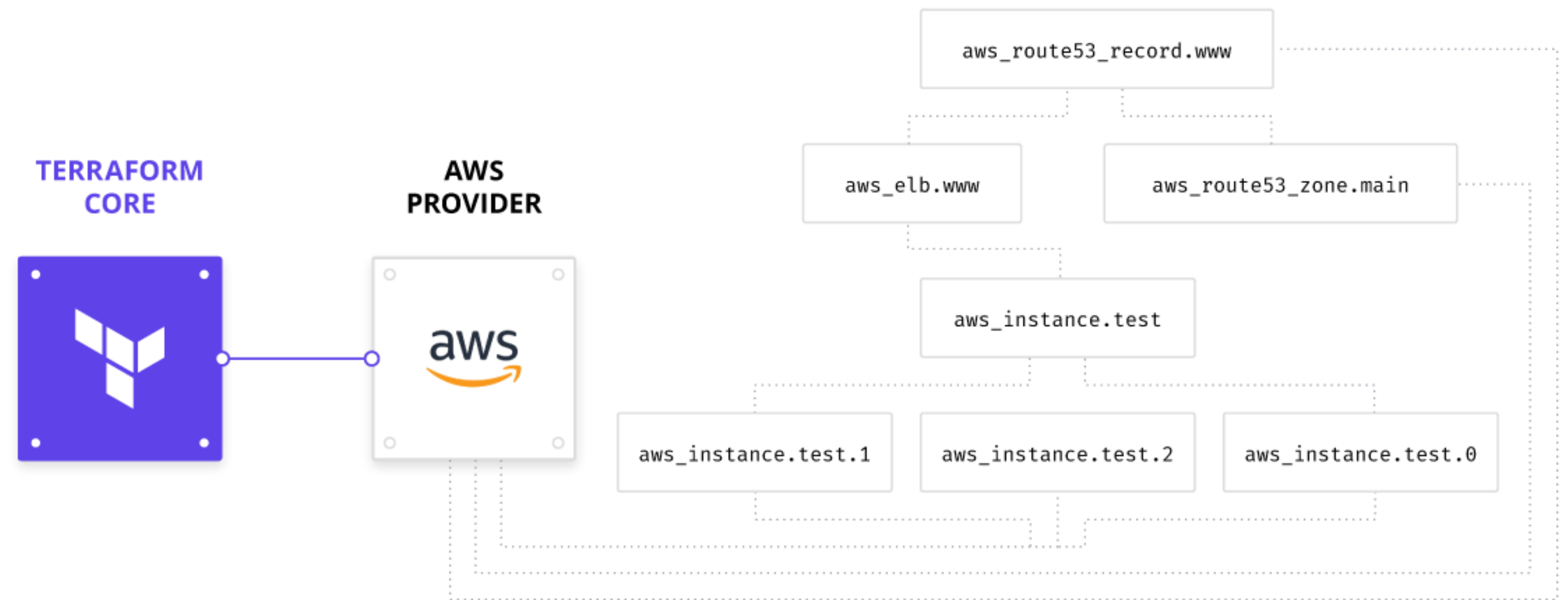
- Reading and Interpolating configuration files and modules
- State Management
- Executing plan
- Communicating with providers
- Constructing resource graph



Resource Graph



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



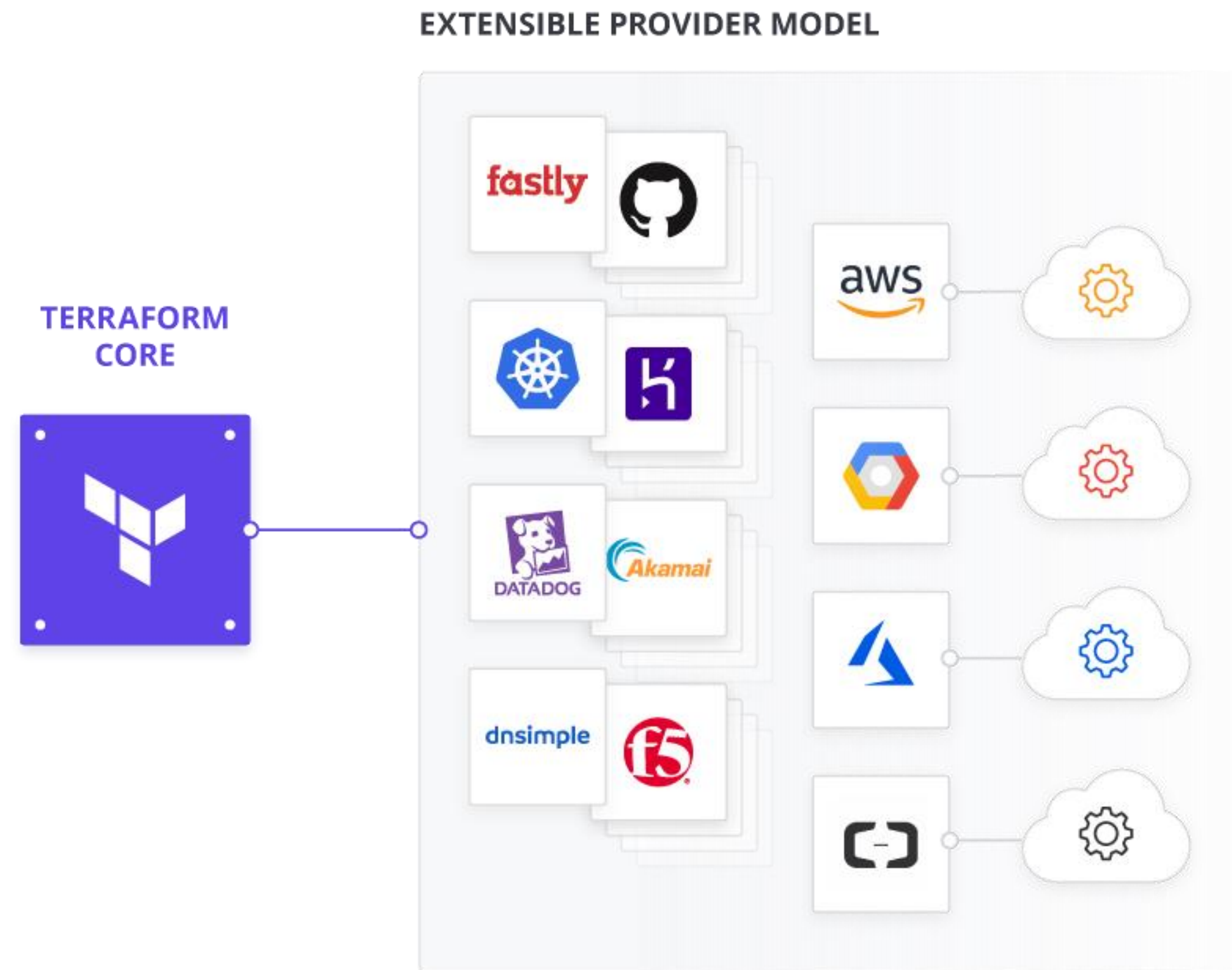
Provider Plugins



- 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

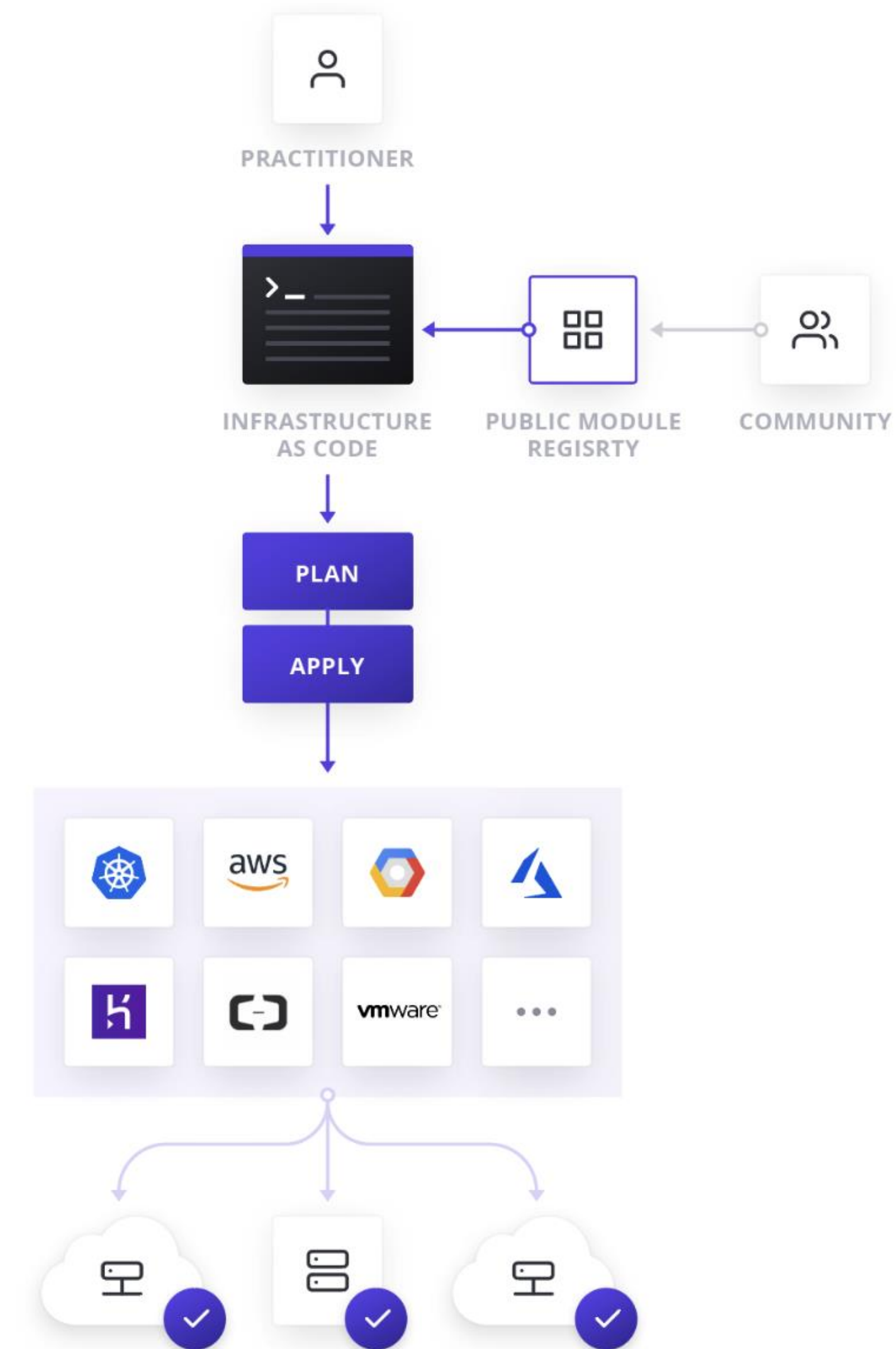


Workflow for OSS



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

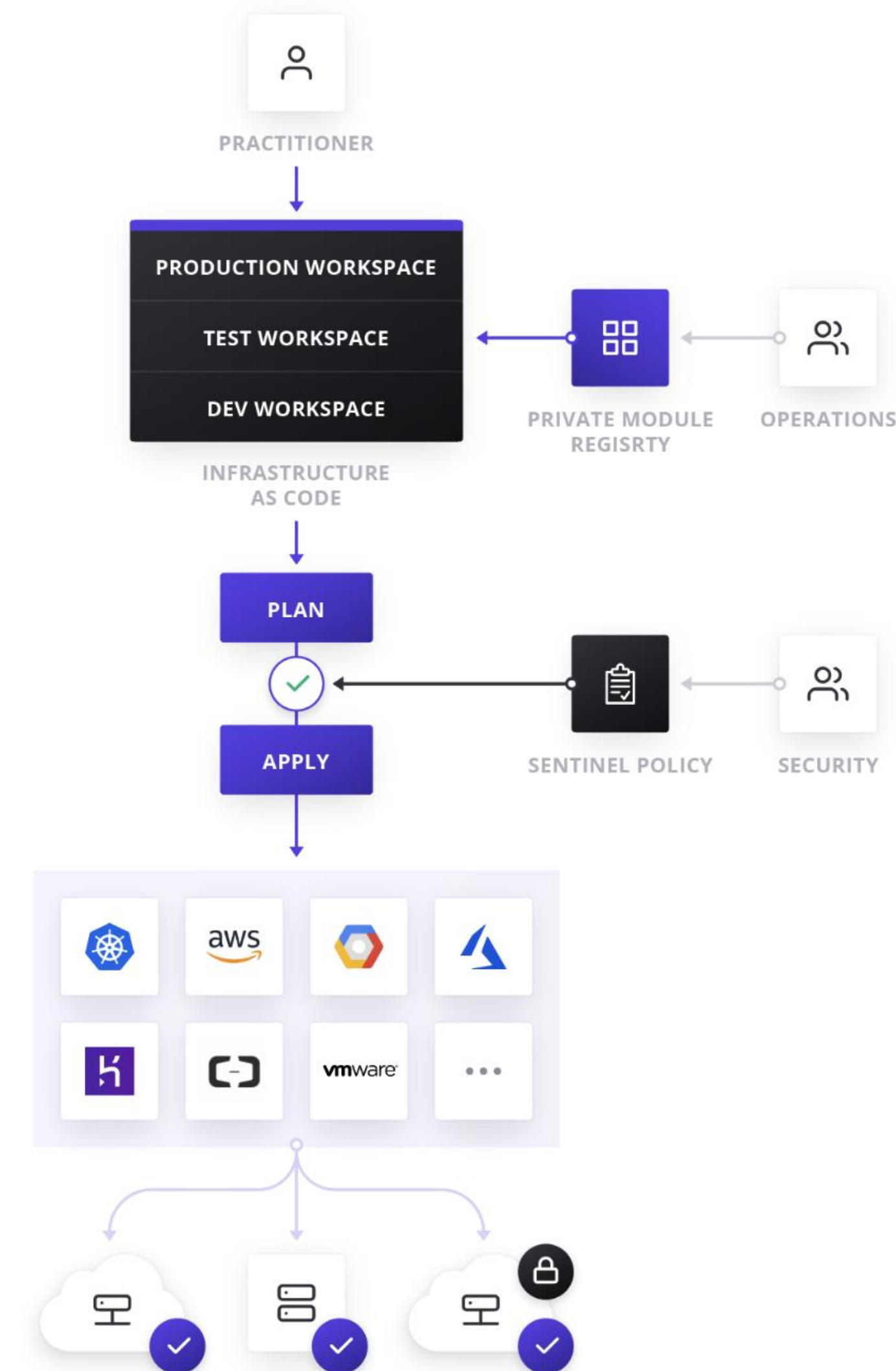


Workflow for Enterprises



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

The Terraform Provider is used to provide 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 = "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"
}
```


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

```
# 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:

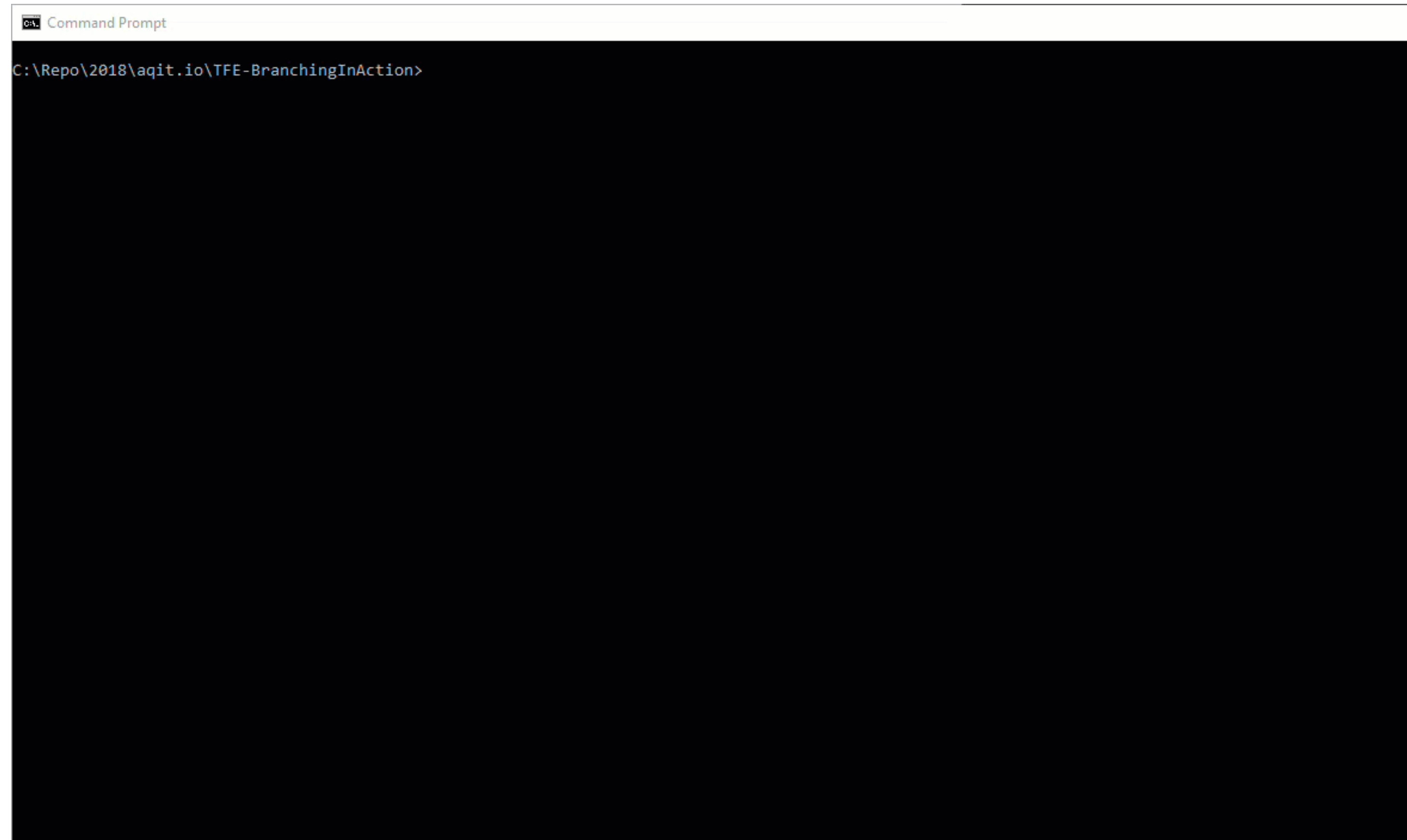
```
vnet_id = /subscriptions/67a8df85-7330-4c65-8343-
d397f95ed522/resourceGroups/demo-terraform-
101/providers/Microsoft.Network/virtualNetworks/acctvnet
```



Terraform **OSS Workflow**

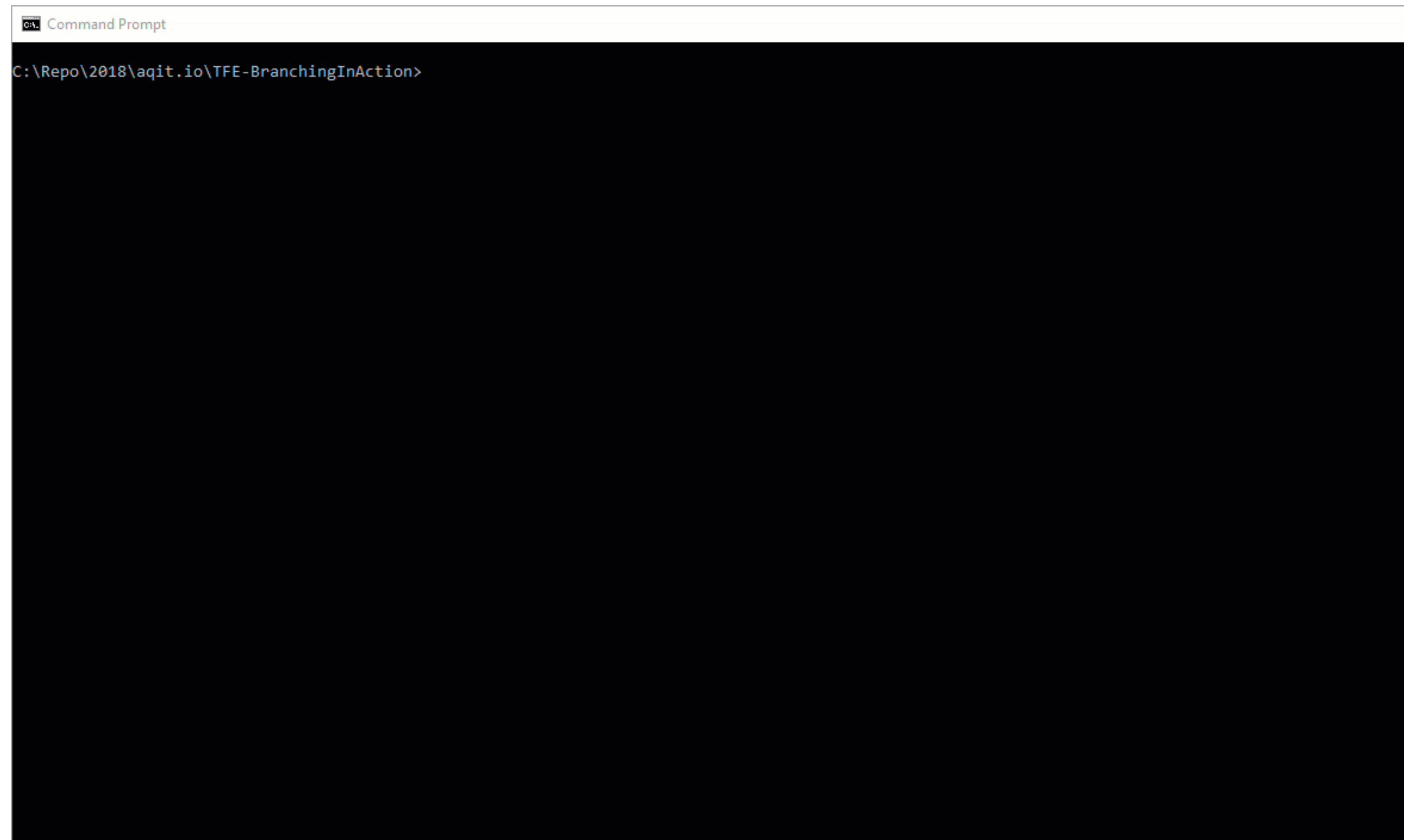
CLI

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



Validate

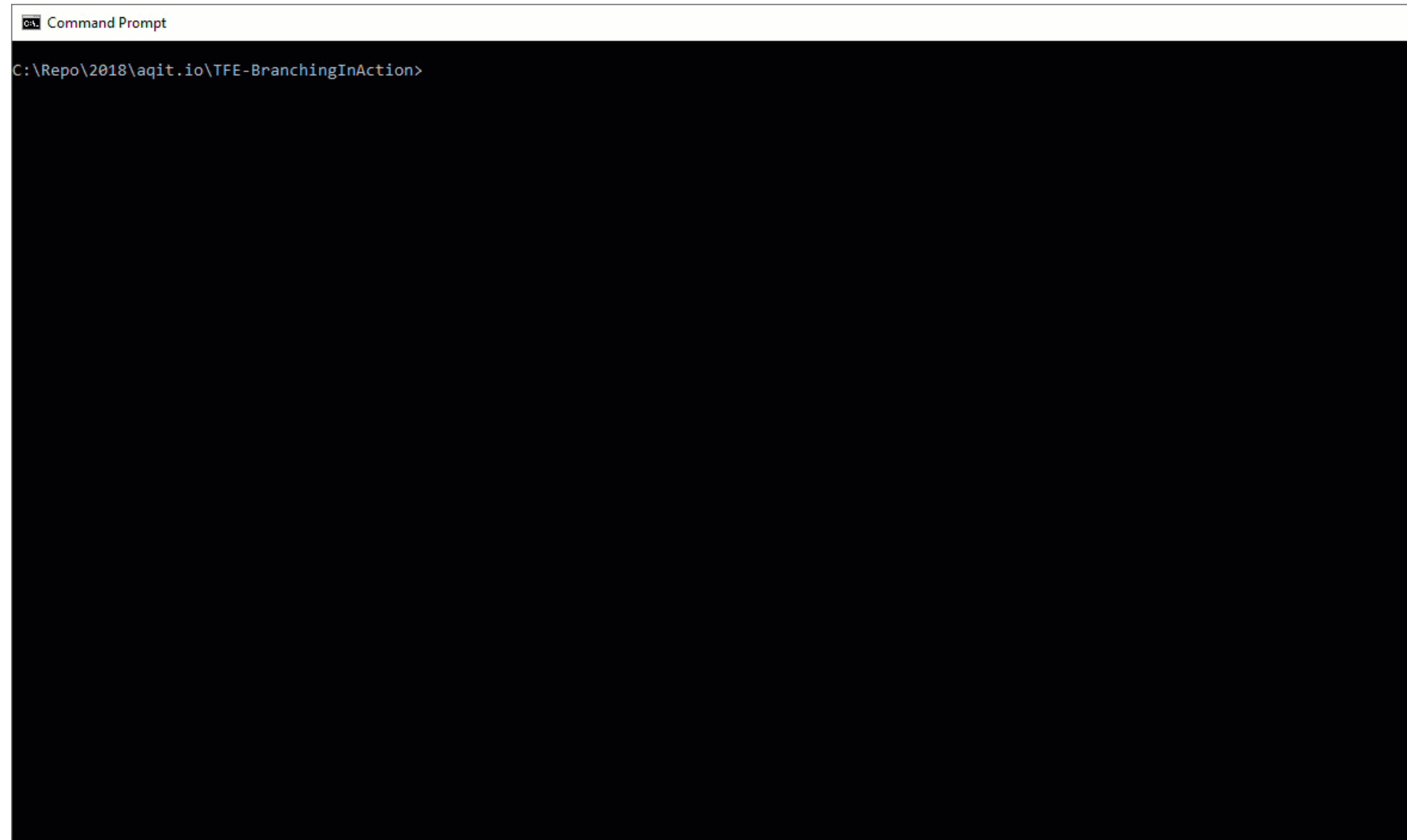
Performs a syntax check on all terraform files in the working directory. Will display an error if any of the files do not validate



```
Command Prompt
C:\Repo\2018\aqit.io\TFE-BranchingInAction>
```

Plan

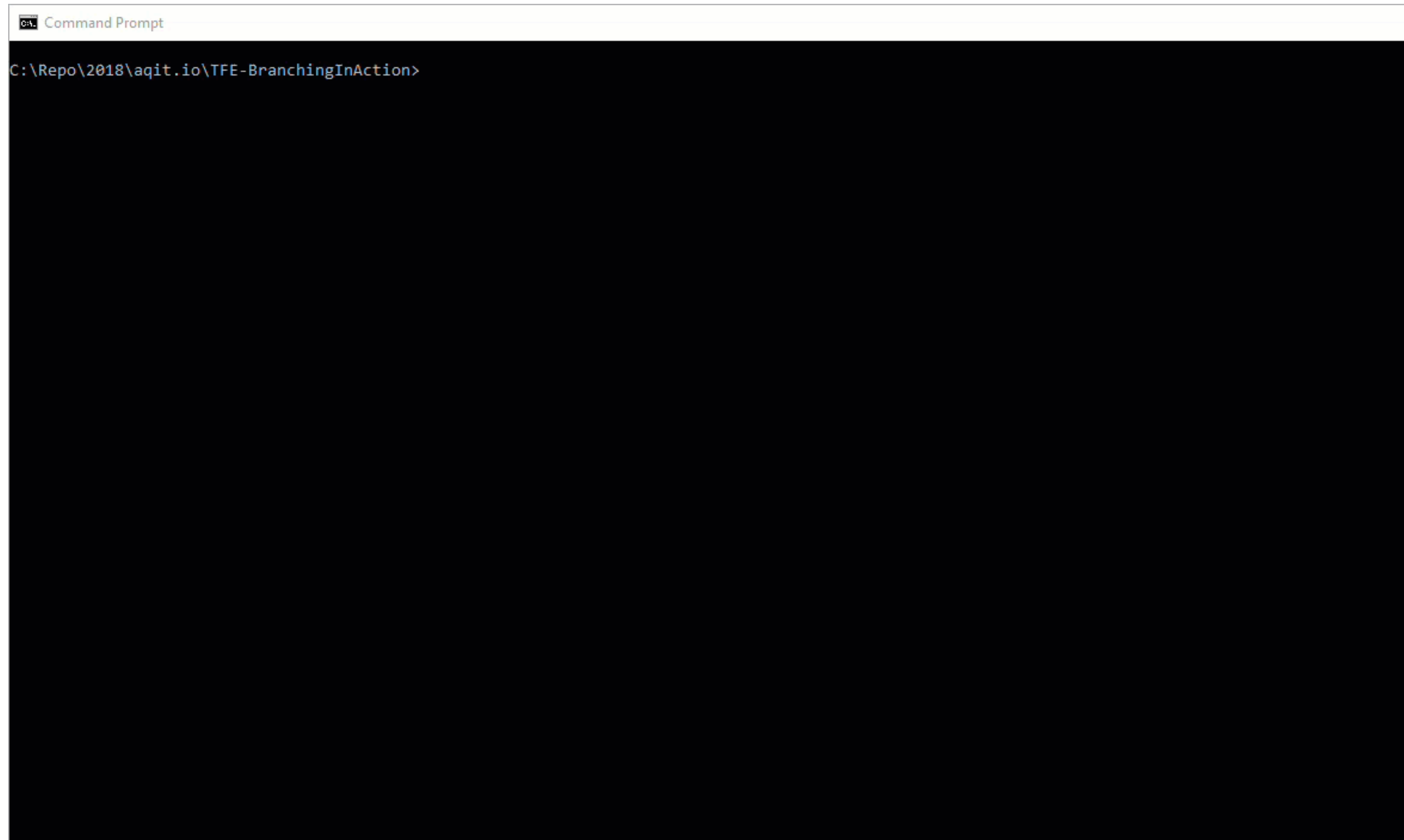
Performs a syntax check on all terraform files in the working directory. Will display an error if any of the files do not validate



```
Command Prompt
C:\Repo\2018\aqit.io\TFE-BranchingInAction>
```


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

Create a new Workspace

[New workspace](#)

[Import from legacy \(Atlas\) environment](#)

This workspace will be created under the current organization, **nicktech**.

WORKSPACE NAME

e.g. workspace-name

The name of your workspace is unique and used in tools, routing, and UI. Dashes, underscores, and alphanumeric characters are permitted. Learn more about [naming workspaces](#).

SOURCE

None

Bitbucket Server

Bitbucket Cloud

GitLab.com

GitHub

+

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

WORKSPACE	STATUS	STARTED
networking-staging	APPLIED ✓	30 minutes ago
postgres-staging	APPLIED ✓	31 minutes ago
binstore-staging	APPLIED ✓	31 minutes ago
postgres-prod	APPLIED ✓	41 minutes ago
account-structure	APPLIED ✓	on hours ago

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

Current RunRunsStatesVariablesSettingsIntegrationsVersion ControlAccess

Variables

These variables are used for all plans and applies in this workspace. Workspaces using Terraform 0.10.0 or later can also load default values from any `*.auto.tfvars` files in the configuration directory.

Sensitive variables are hidden from view in the UI and API, and can't be edited. (To change a sensitive variable, delete and replace it.) Sensitive variables can still appear in Terraform logs if they are designed to output them.

When setting many variables at once, [the TFE CLI tool's pushvars command](#) or the [variables API](#) can often save time.

Terraform Variables

These Terraform variables are set using a `terraform.tfvars` file. To use interpolation or set a non-string value for a variable, click its HCL checkbox.

key

value

Save Variable

Cancel

/ USE CASE: INFRASTRUCTURE AS CODE

© 2018 HashiCorp / 33

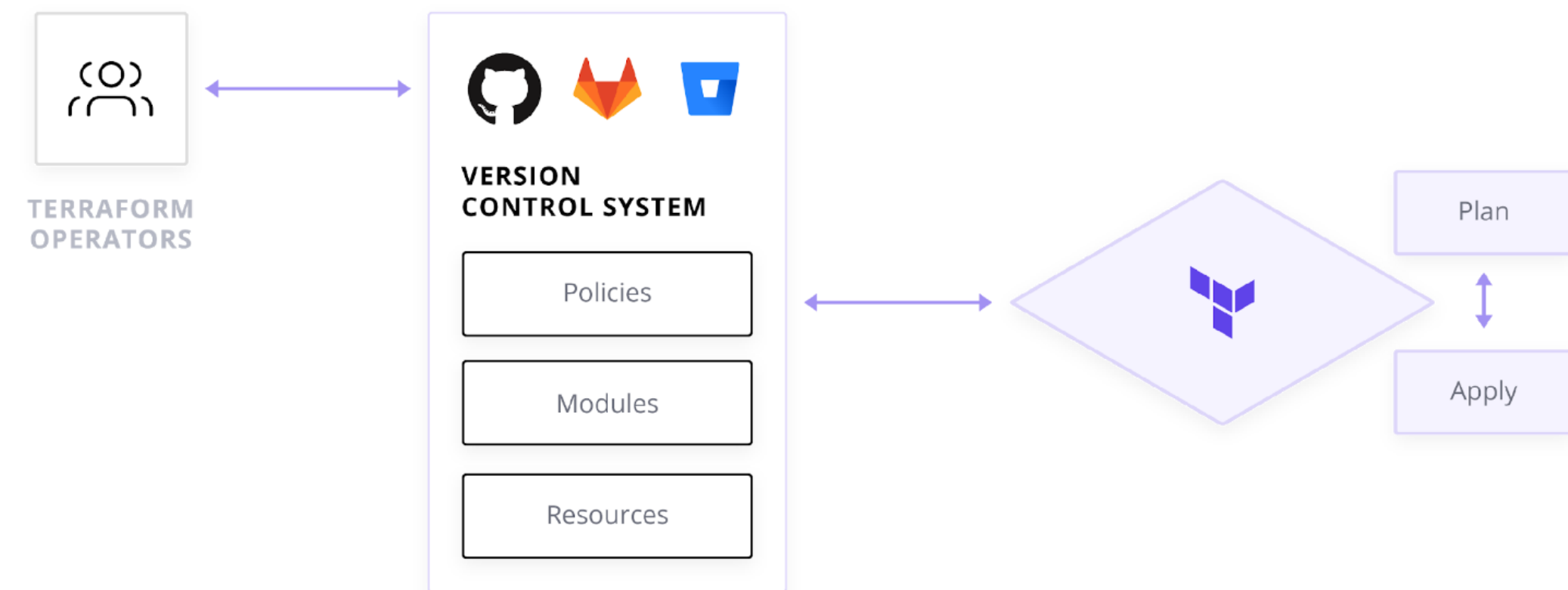
Feature: VCS Connection



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



All checks have passed
2 successful checks

[Hide all checks](#)

atlas/tfe-core-practitioner/app — Terraform plan: 2 to add, 0 to change, 0 to ...

[Details](#)

sentinel/tfe-core-practitioner/app — Policy checks passed

[Details](#)

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request

▼

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Feature: Runs and State



Remote Operations

- Trigger Terraform runs remotely:
 - Pull request in VCS
 - API call in CI/CD pipeline
 - CLI
- Automatic 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

run-EbZd triggered from GitHub ⌚ 2 minutes ago

✓ **APPLIED** 2 minutes ago

Commit 9e77320: Update main.tf

By apdavanzo | Branch master | Repo apdavanzo/mydemoapp

Run Timeline

● Run created by GitHub Webhook

PLAN

○ Queued 2 minutes ago

○ Started 2 minutes ago

✓ Executed and saved successfully

2 minutes ago

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

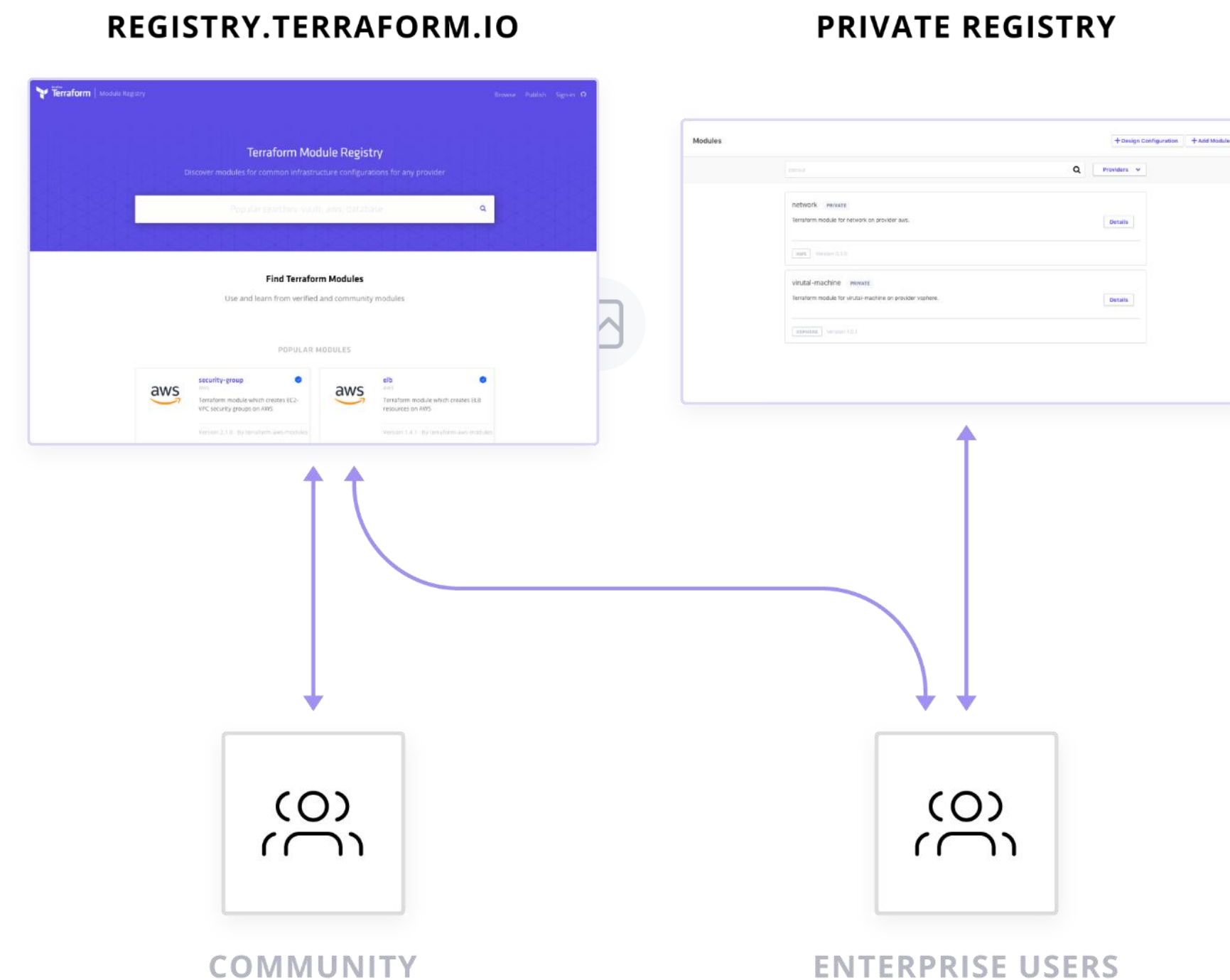
The public module registry offers the large community a repository to store and share modules.

- 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

The screenshot displays the Configuration Designer interface, which is divided into two main panels. The left panel, titled 'Modules / Configuration Designer', shows a search bar with 'consul' entered and a 'Providers' dropdown. Below this, there's a section 'ADD MODULES TO WORKSPACE' with two modules listed: 'network' (Terraform module for network on provider aws, Version 0.1.0) and 'virtual-machine' (Terraform module for virtual-machine on provider vsphere, Version 1.0.1). The right panel, also titled 'Modules / Configuration Designer', shows the 'SELECT MODULE TO CONFIGURE' section with the 'network' module selected and marked as 'Configured'. To the right of this is the 'CONFIGURE VARIABLES' section, which contains four variable configuration fields: 'region' (OPTIONAL, The default AZ to provision to for the provider), 'subnet_availability_zone' (OPTIONAL, The default AZ for the subnet), 'subnet_cidr_block' (OPTIONAL, The default CIDR block for the subnet demo), and 'vpc_cidr_block' (OPTIONAL, The default CIDR block for the VPC demo). Each field has a search bar and a 'Deferred' button.

Feature: Full API



Full API

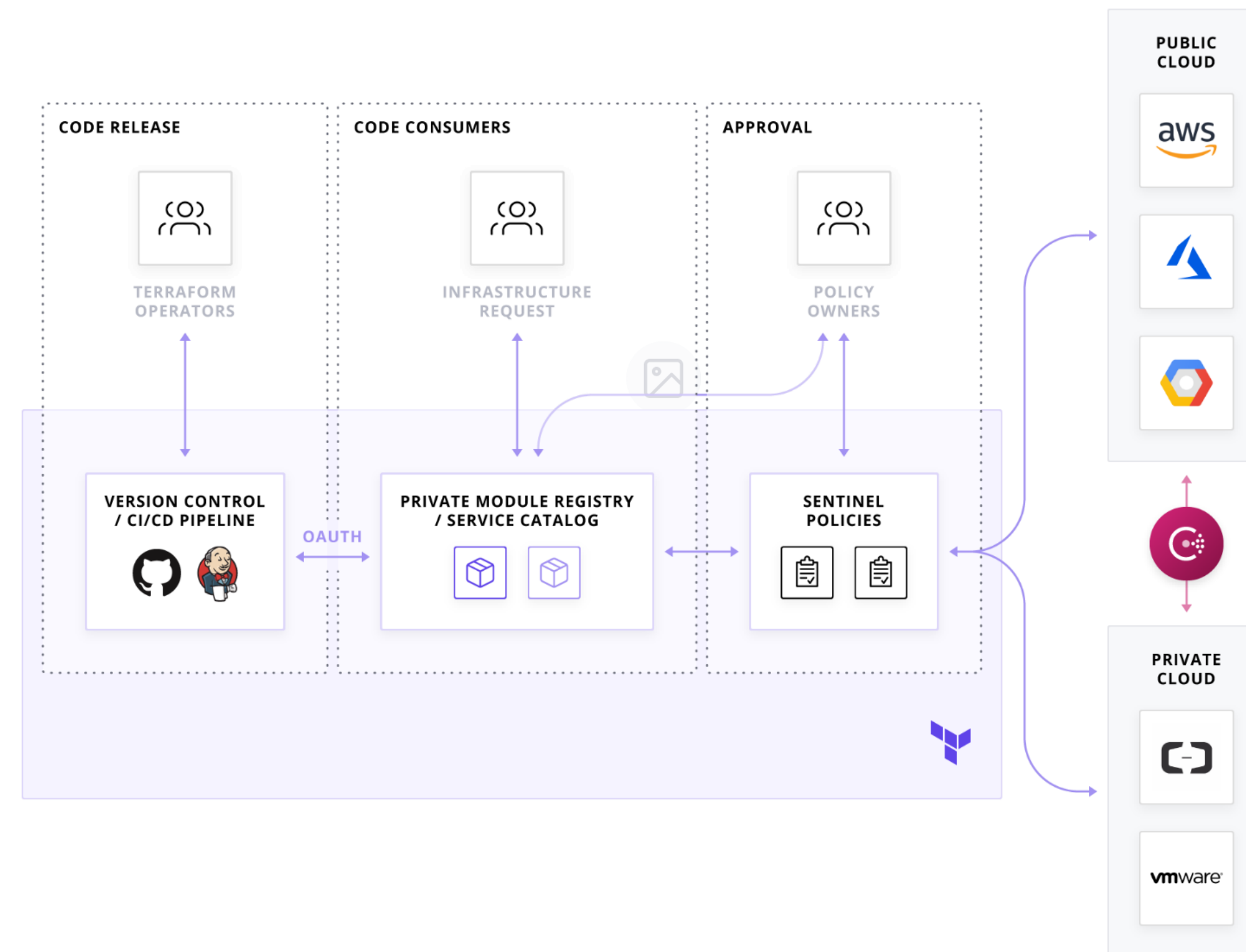
Terraform can be fully operated via API allowing organizations to easily integrate it into their existing application delivery pipelines.

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

Team Management

Teams let you group users into specific categories to allow for finer grained access control policies. For example, your developers could be on a dev team that only has access to applications.

In order to allow a team access to a resource, go to the Access settings for the specific resource and enter the team name. At this point you can control the access level for that team.

The **owners** team is a special team that has implied access for all of your resources, but also has the ability to manage your organization.

Create new team

NAME

Create team

The name of the team.

Teams

Operators

1 members

owners

3 members

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

A screenshot of the 'Configure SAML' interface in Terraform Enterprise. The form is divided into several sections: 'Configure SAML' (header), 'SAML SSO Enabled?' (checkbox), 'Terraform Enterprise SAML Endpoints' (section header), 'ACS Consumer URL' (text input with 'https://atlas.hashicorp.com/users/saml/auth'), 'Metadata URL' (text input with 'https://atlas.hashicorp.com/users/saml/metadata'), 'SAML Identity Provider Settings' (section header), 'Single Sign On URL' (text input with 'example: http://example.com/sso'), 'Single Log Out URL' (text input with 'example: http://example.com/slo'), 'Identity Provider Certificate' (text area), 'Team Membership Mapping' (section header), 'Team membership mapping enabled?' (checkbox), 'Team Attribute Name' (text input with 'MemberOf'), 'Attribute Mapping' (section header), 'User Email Address' (text input with 'urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress'), and 'Save'/'Cancel' buttons at the bottom.

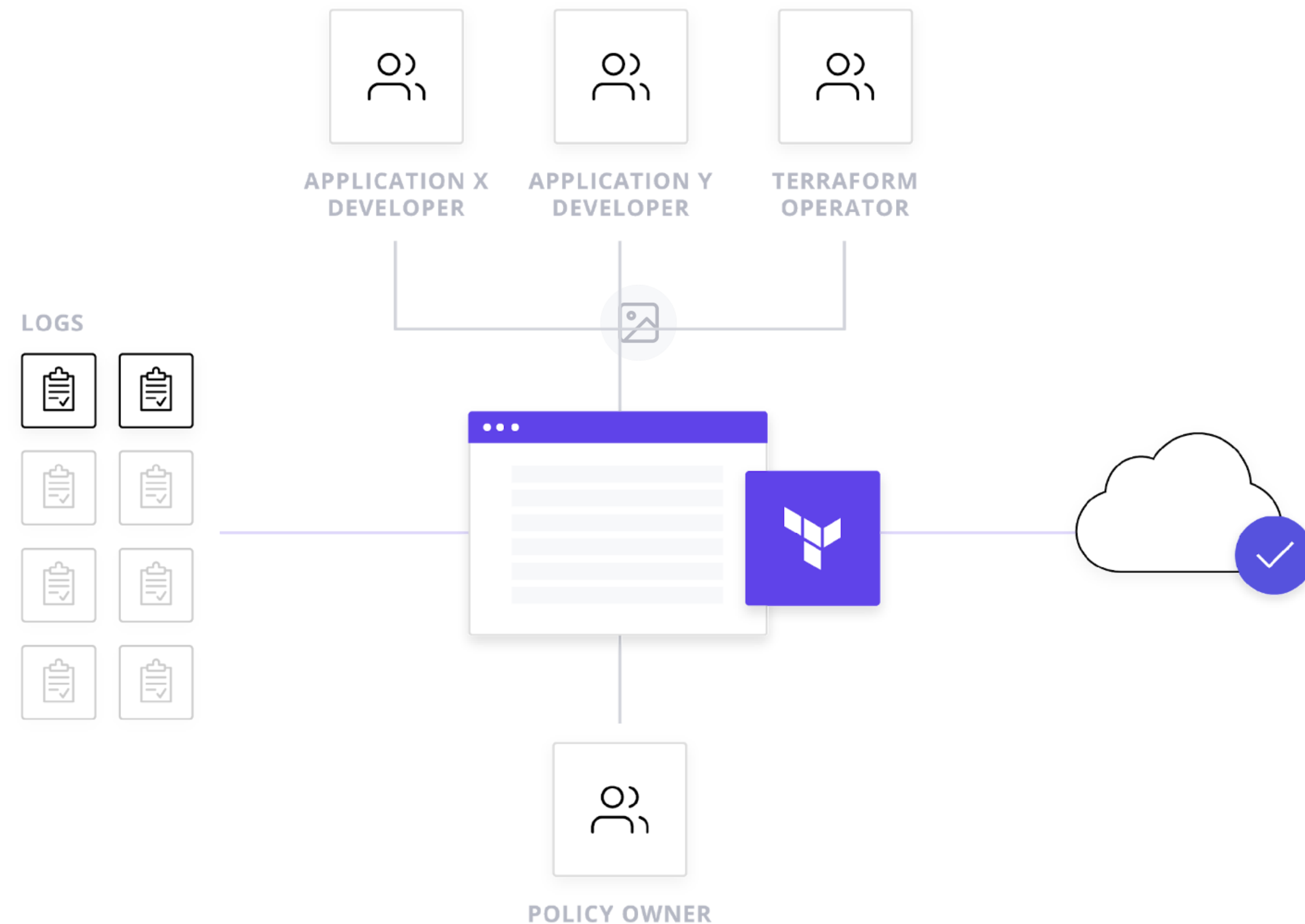
Feature: **Audit & Administrate**



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



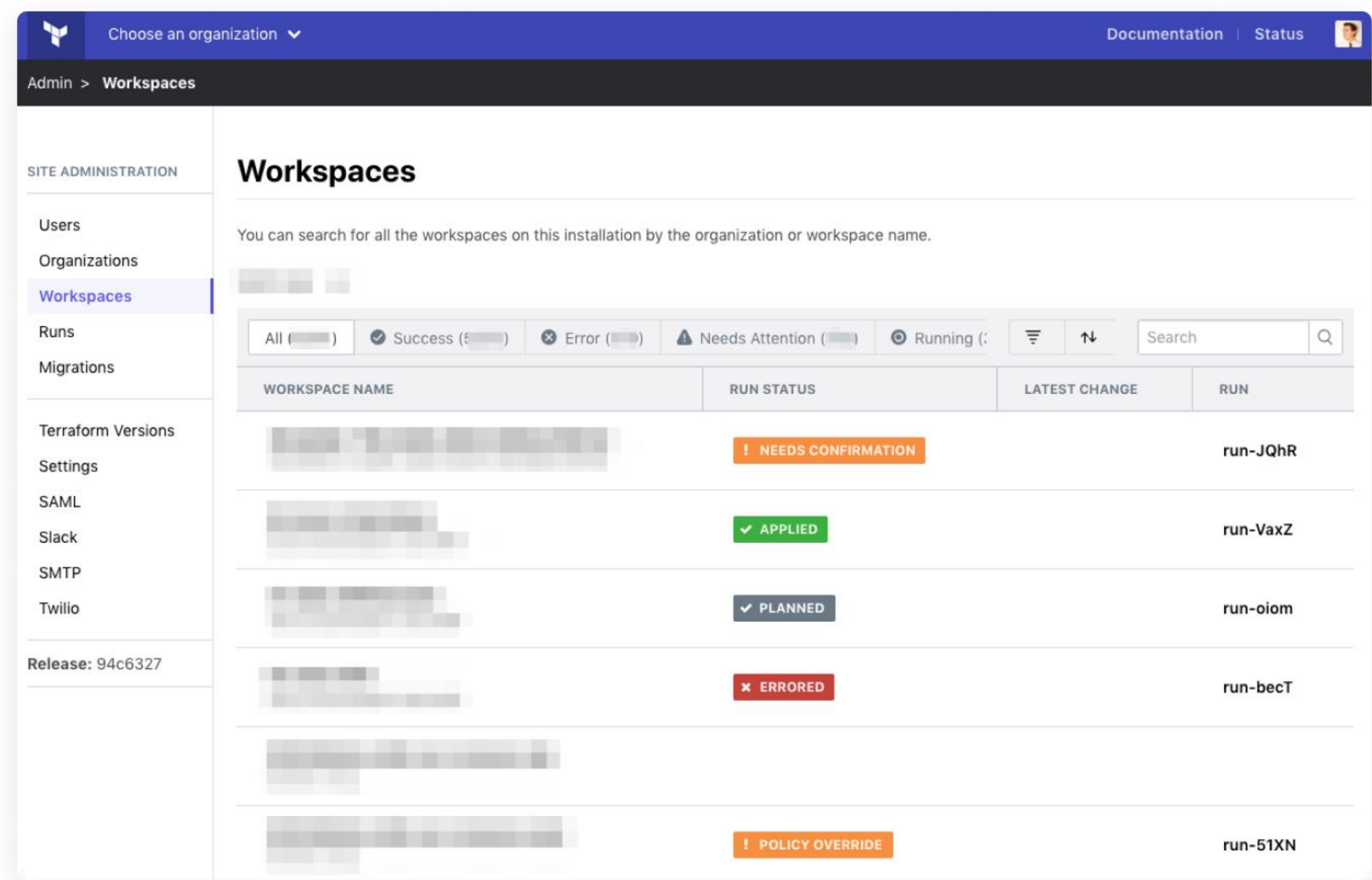
Feature: Audit & Administrate



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**

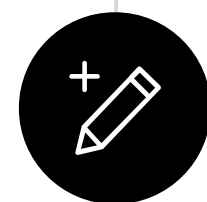


About Terraform



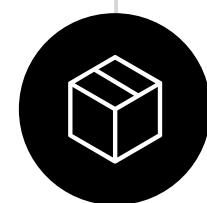
2014

Product Launch



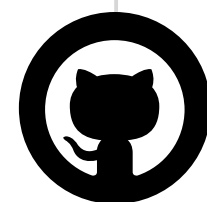
1200+

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.





Thank you

hello@hashicorp.com

www.hashicorp.com