



# From manual cloud provisioning to one click deployments

**Patrick Schulz**

**Sr. Solutions Engineer**

**[pschulz@hashicorp.com](mailto:pschulz@hashicorp.com)**

# About HashiCorp

## Leading Cloud Infrastructure Automation

Our software stack enables the provisioning, securing, connecting and running of apps and the infrastructure to support them.

We unlock the cloud operating model for every business and enable their digital transformation strategies to succeed.

**Founded**

**2012**

**Employees**

**650+**

**Funding**

**174M**



# The effects of digital transformation



TRADITIONAL DATACENTER  
"STATIC"



DEDICATED INFRASTRUCTURE



MODERN DATACENTER  
"DYNAMIC"



PRIVATE CLOUD

+



AWS

+



AZURE

+



GCP

+



...

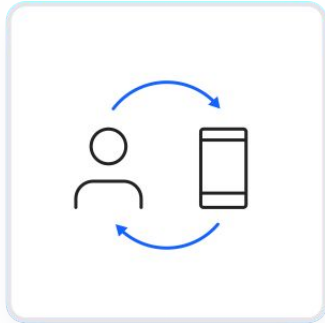
SYSTEMS OF RECORD



SYSTEMS OF ENGAGEMENT

# Cloud adoption is a secular trend

Digital transformation means pressure on application delivery



Digital experiences are now the primary interface between a customer and a business, or business and business.



Experiences are typically device- and cloud-first: rich, personal interface, with large scale data processing and intelligence.



This patterns demands a change in the model for software delivery to meet delivery goals, and transformation objectives.



# What are we trying to achieve?

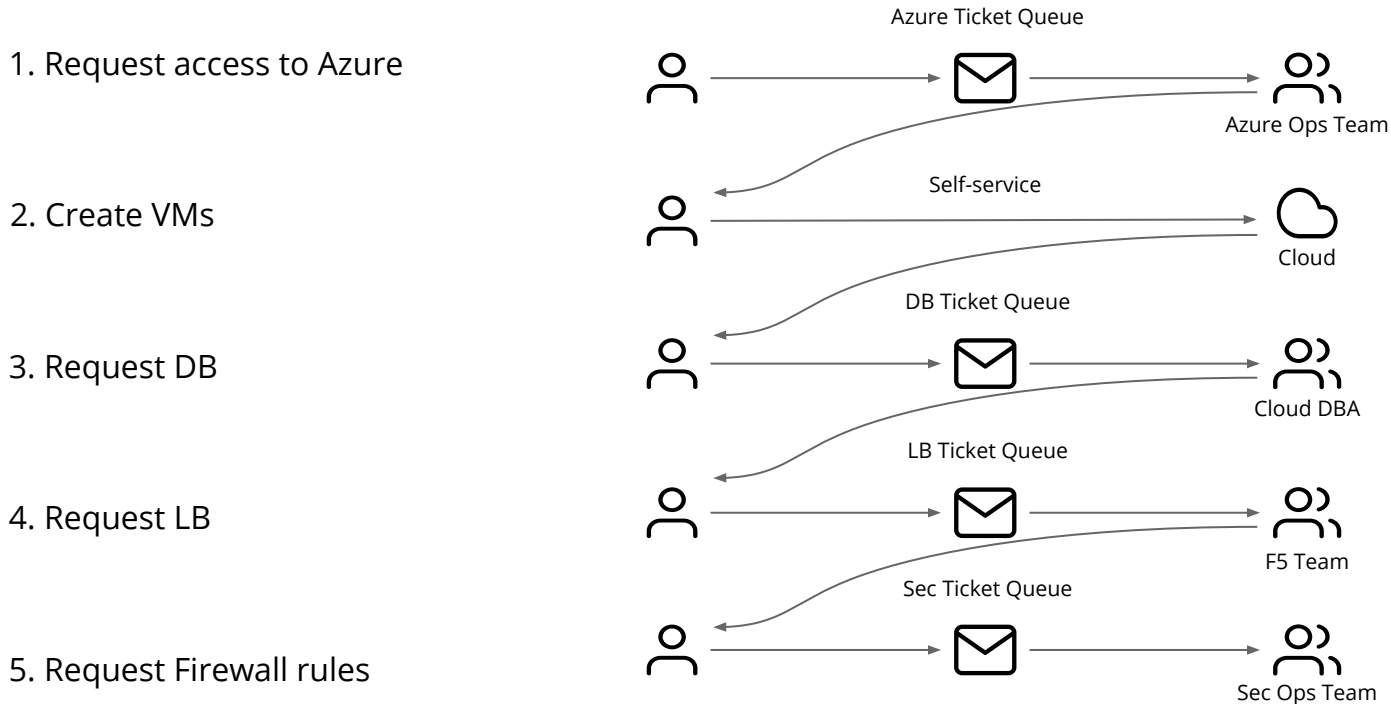
Help you to achieve your business goals by implementing a common workflow which allows for more agility through self-service.

Which means faster Time to Value while maintaining control over security, governance & cost.

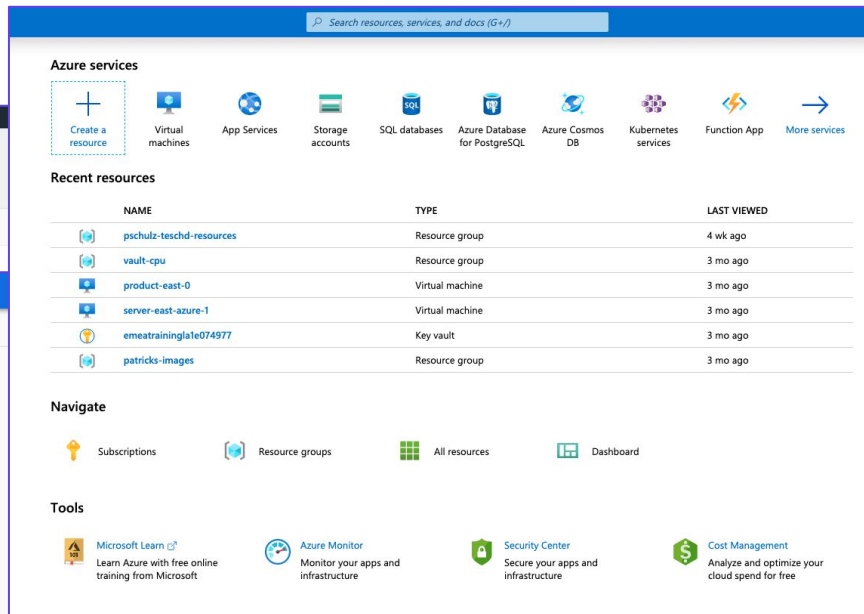
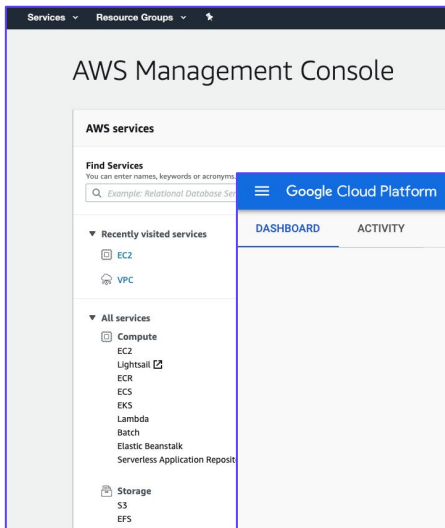


# Provisioning infrastructure today...

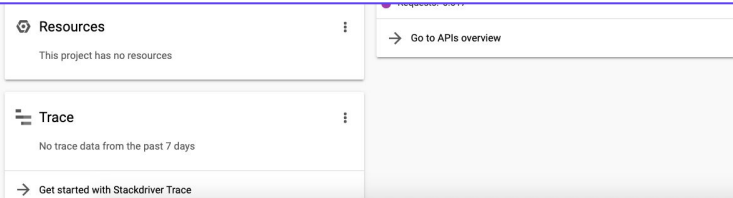
# Today's ticket-based workflow



# Why you should stop doing it like you used to do?



- **Reduced productivity** from manual workflows using Point-and-Click GUIs
- **Increased cost** with “cloud waste” or over provisioning
- **Increased risk** with more chances for human error and best practices are followed on a “best effort” basis using tribal knowledge







---

# Introducing HashiCorp Terraform Infrastructure as Code

# Why Infrastructure as Code?



- Describe the desired state of your infrastructure in a declarative way
- Deploy infrastructure automated in a consistent and repeatable manner
- Consistent workflow across different platforms (APIs)
- Collaboration through version control system (Azure DevOps, GitHub, GitLab, etc.)
- Instant documentation and tracking of changes (Versioning)

**In essence: Don't repeat yourself and spent your valuable time and the time of others waiting for you, time manually by clicking through a UI.**

# IaC with Terraform



```
CODE EDITOR

provider "azurerm" {}

resource "azurerm_resource_group" "rg" {
  name = "test"
  # ...
}

resource "azurerm_network_interface" "main" {
  # ...
}

resource "azurerm_virtual_machine" "main" {
  name                = "server"
  location            = data.azurerm_resource_group.rg.location
  resource_group_name = data.azurerm_resource_group.rg.name
  vm_size            = "Standard_D2s_v3"
  network_interface_ids = [azurerm_network_interface.main.id]
```

# Terraform

## Providers

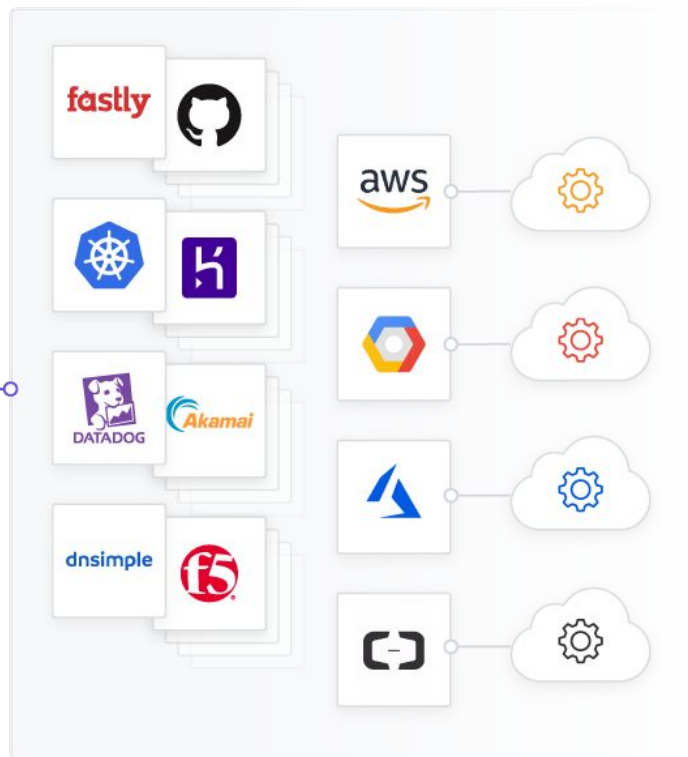
Providers, extensible to any cloud or service with an API, enable Terraform to provision diverse services without abstracting functionality.

- 200+ Providers and Services

TERRAFORM  
CORE



EXTENSIBLE PROVIDER MODEL

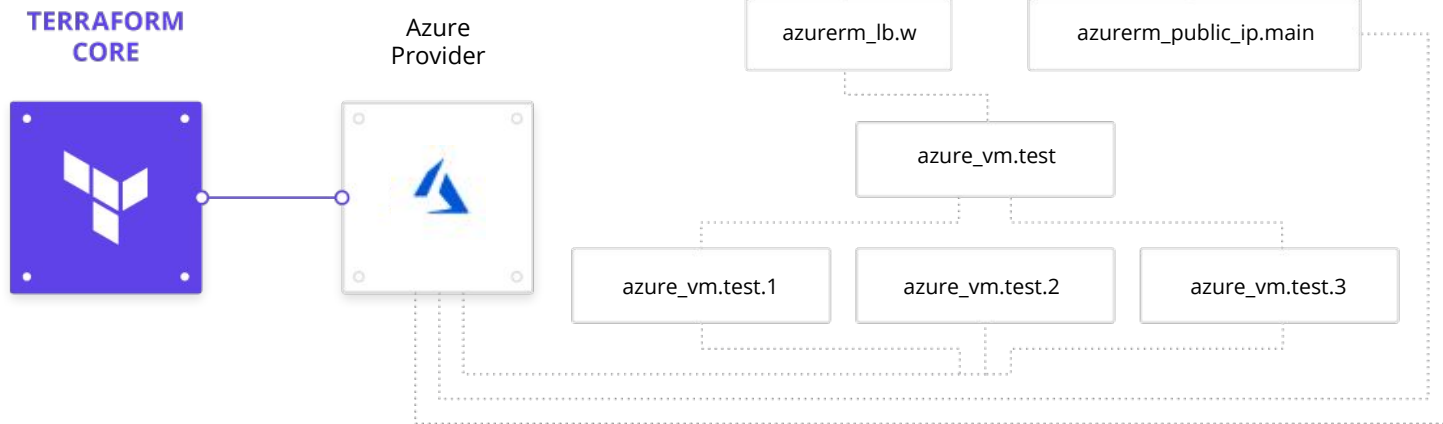


# Terraform



## Dependencies / Resource Graph

Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.



# Terraform

## Modules

A module is a container for multiple resources that are used together. Modules can be used to create lightweight abstractions, so that you can describe your infrastructure in terms of its architecture, rather than directly in terms of physical objects.



A screenshot of the Terraform Registry page for the 'vnet' module. The page has a purple header with the Terraform logo and 'Registry' text. A search bar is on the right. The main content area shows the module name 'vnet' with the AZURE logo, a version selector set to 'Version 1.2.0', and a description: 'Terraform module to create/provision Azure vnet'. Below this, it says 'Published August 15, 2018 by Azure', 'Module managed by rguthrieftf', 'Total provisions: 13,132', and 'Source: github.com/Azure/terraform-azurem-vnet (report an issue)'. On the right, there are 'Provision Instructions' which say to copy and paste into the Terraform configuration and run 'terraform init'. A code block shows the module definition: 'module "vnet" { source = "Azure/vnet/azurem" version = "1.2.0" }'. At the bottom, there are tabs for 'Readme', 'Inputs (9)', 'Outputs (5)', 'Dependencies (0)', and 'Resources (3)'. The 'Readme' tab is selected.

### terraform-azurem-vnet

[build](#) [package](#)

#### Create a basic virtual network in Azure

This Terraform module deploys a Virtual Network in Azure with a subnet or a set of subnets passed in as input parameters.

The module does not create nor expose a security group. This would need to be defined separately as additional security rules on subnets in the deployed network.

#### Usage

HCL [Copy](#)

```
1 module "vnet" {
2   source           = "Azure/vnet/azurem"
```

# Terraform

## Plan

Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.



```
TERMINAL

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
  -/+ destroy and then create replacement

Terraform will perform the following actions:

# aws_internet_gateway.default will be created
+ resource "aws_internet_gateway" "default" {
  + id          = (known after apply)
  + owner_id    = (known after apply)
  + tags        = {
    + "Name"      = "pschulz-DEV"
    + "TTL"       = "48"
    + "env"       = "DEV"
    + "environment" = "aws-tfe-repo-demo"
    + "owner"     = "pschulz"
  }
}
```

# Terraform

## State

Terraform caches information about your managed infrastructure and configuration. This state is used to persistently map the same real world resources to your configuration from run-to-run, keep track of metadata, and improve performance for large infrastructures.



AWS-Network-Operations-DEV ©

Runs States Variables Settings Queue plan

### Current Run

**Update main.tf** CURRENT PLANNED  
#run-NE2KZUZ1uBCWdqE9 | pschulz1 triggered from GitHub | Branch master | 349d2ec | 13 days ago

### Run List

- Update main.tf** CURRENT PLANNED  
#run-NE2KZUZ1uBCWdqE9 | pschulz1 triggered from GitHub | Branch master | 349d2ec | 13 days ago
- Queued manually in Terraform Cloud** APPLIED  
#run-1Yn5oakcgnHJvnE | pschulz1 triggered from Terraform Cloud UI | Branch master | 4c7e27d | 23 days ago
- Update main.tf** ERRORED  
#run-D6YA5sBQnHEUHd47 | pschulz1 triggered from GitHub | Branch master | 4c7e27d | 23 days ago
- Update main.tf** DISCARDED  
#run-SaggnvHSZBQXyA5p | pschulz1 triggered from GitHub | Branch master | 7ff6e3b | a month ago
- Queued manually to destroy infrastructure** APPLIED  
#run-dNyngrU9ch8PBBIH | pschulz1 triggered from Terraform Cloud UI | Branch master | 21989b7 | 2 months ago
- Queued manually in Terraform Cloud** APPLIED  
#run-3ouh8BHxv7xYTTGq | pschulz1 triggered from Terraform Cloud UI | Branch master | 21989b7 | 3 months ago
- Queued manually in Terraform Cloud** APPLIED  
#run-Vh778ayZ7BZyP4q | pschulz1 triggered from Terraform Cloud UI | Branch master | 21989b7 | 3 months ago
- Queued manually to destroy infrastructure** APPLIED  
#run-XzFu65BH5ScVynHJ | pschulz1 triggered from Terraform Cloud UI | Branch master | 21989b7 | 3 months ago
- demo commit** PLANNED  
#run-pfZVY2ZQvnmXGCq | pschulz1 triggered from GitHub | Branch master | 21989b7 | 3 months ago
- Queued manually in Terraform Cloud** PLANNED  
#run-uWm82KZkZWwvJvJ | pschulz1 triggered from Terraform Cloud UI | Branch master | 6327e3e | 3 months ago



# Terraform



## State

```
"private_key_data": {  
  "sensitive": false,  
  "type": "string",  
  "value": "-----BEGIN RSA PRIVATE KEY-----\  
u1exwLTyavmWeD6Ka2PesgWU5rhniF92wavsuTv3DA  
pydsWsc1i2eisgGVJ2nDTht8mWCK5AvyUeRIBG4nxm  
QBhBF1MhlCd3qgA1P2M+gPvneS0CmruK5q6PjIkAJG  
},  
"public_key_data": {  
  "sensitive": false,  
  "type": "string",  
  "value": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQAB
```

The state become a crucial part of your Terraform deployment process, as it will contain sensitive information eventually, which need to be protected.

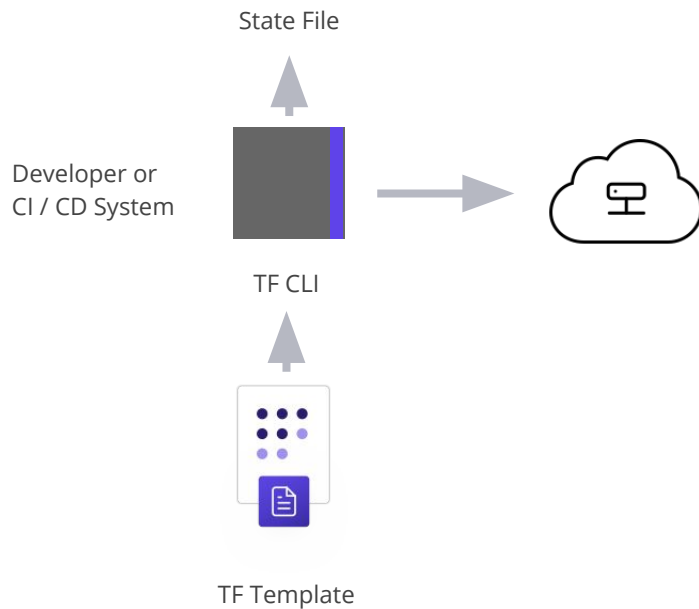
```
"public_ip": "54.213.82.137"
```

```
mysqladmin -u root password R00tPassword
```

```
"ses_smtp_password": "AhWsxpmKWsI9iVQKJgada
```

# Cloud Provisioning with Terraform OSS

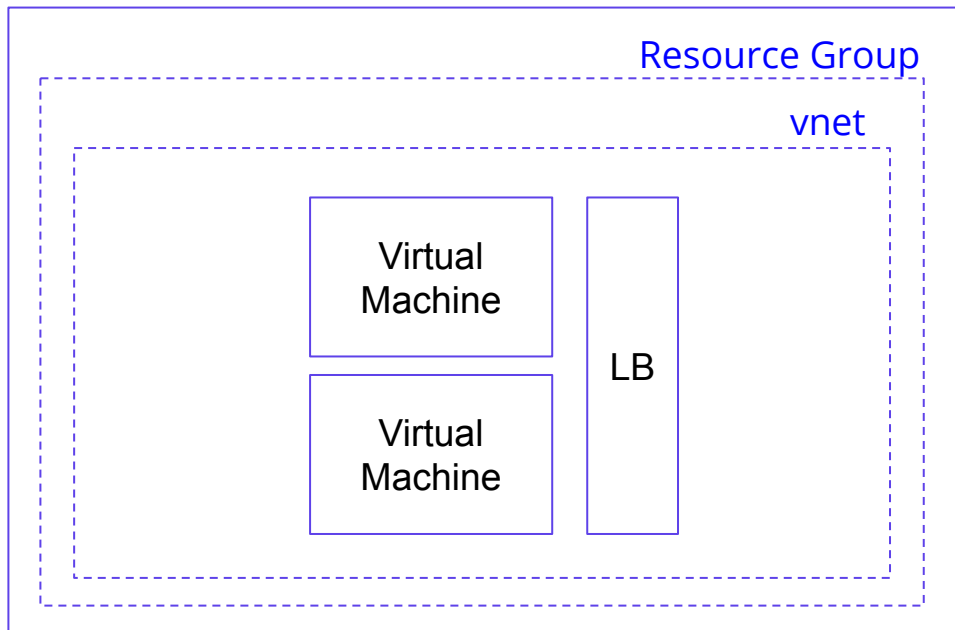
## A common Cloud Operating Model



# Infrastructure diagram



Azure Cloud



Lets we need a couple of new resources:

- New Azure Subscription
- Resource Group
- Subnet
- Virtual Machines
- Load Balancer



HashiCorp

**Terraform**

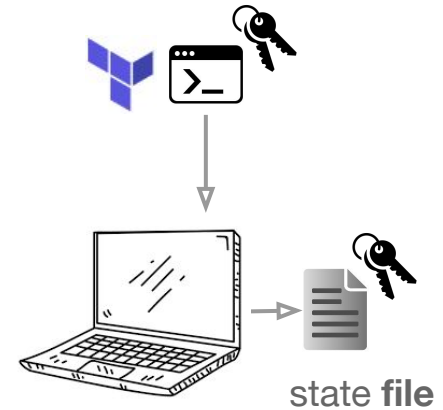
**Demo**

# What's the Problem with OSS?



Open source doesn't scale very well, for multiple reasons:

- Potential secret sprawl and exposure through VCS.
- Missing control over security, governance & cost.
- State Management (Storage, Locking, Protection/RBAC).
  - Required infrastructure needs to be provisioned and secured, like object storage, service accounts.
  - Imagine managing all the accounts or having the risk of all users/pipelines being able to access a single object store.
- No API - Limited to CLI workflow.
- No integration into tools like ServiceNow.



# Introduction to TFE



- Central platform (running on-prem. or in the Cloud).
- Provides common workflows for use across teams: VCS/API/CLI/UI.
- Allows to establish a so called producer/consumer model.
- Adds an API, RBAC, State Management, VCS connection, Variable Store, Private Module Registry, Cost Estimation.
- Allows to enforce Sentinel policies across workspaces.

Patrick-Schulz / Workspaces

Patrick-Schulz / Workspaces

Workspaces Matching 3 of 10 total [Clear filters](#) [+ New workspace](#)

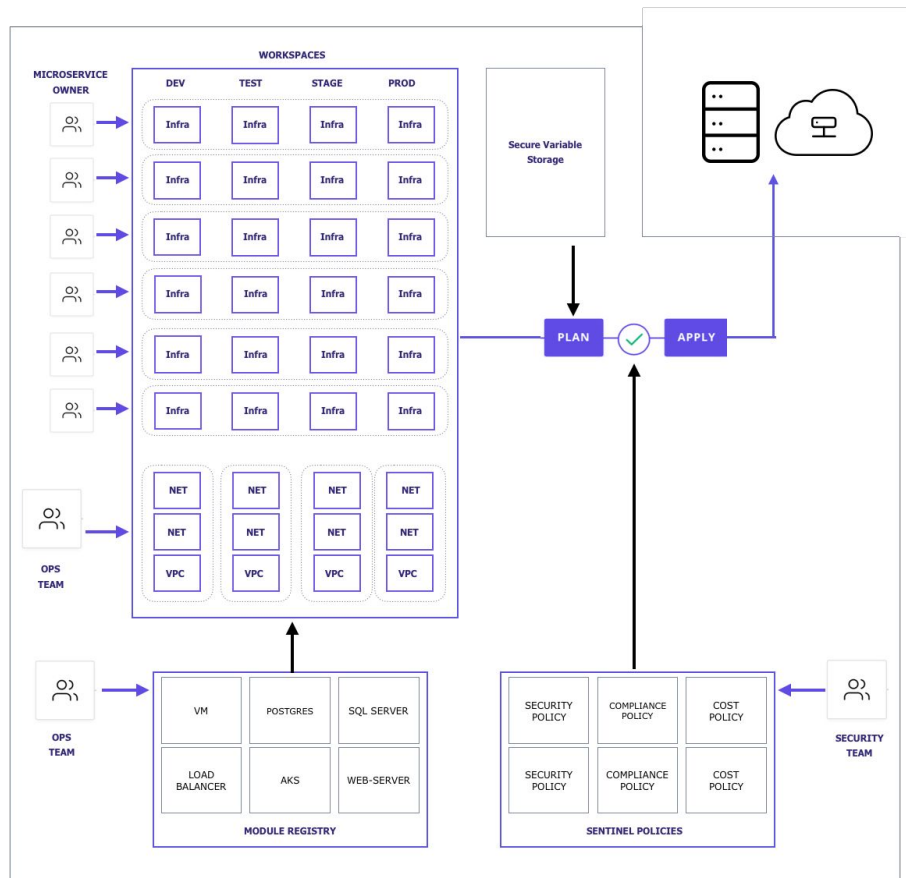
All 10 Success 4 Error 3 Needs Attention 0 Running 0 Filter Sort Azure

WORKSPACE NAME	RUN STATUS	RUN	REPO	LATEST CHANGE
<a href="#">Azure-Consul-HAProxy-LB</a>	✓ APPLIED	run-1bw6k3Fc3buG3udb	pschulz1/Azure-Consul-HAProxy-LB	4 months ago
<a href="#">Azure-Consul-Nomad</a>	✓ APPLIED	run-D47pSdbbyn6eph2Q	pschulz1/Azure-Consul-Nomad	2 months ago
<a href="#">Azure-Consul-Vault-AD</a>	✓ APPLIED	run-qwVXbNyvEhu9RQmk	pschulz1/Azure-Consul-Vault-AD	5 months ago

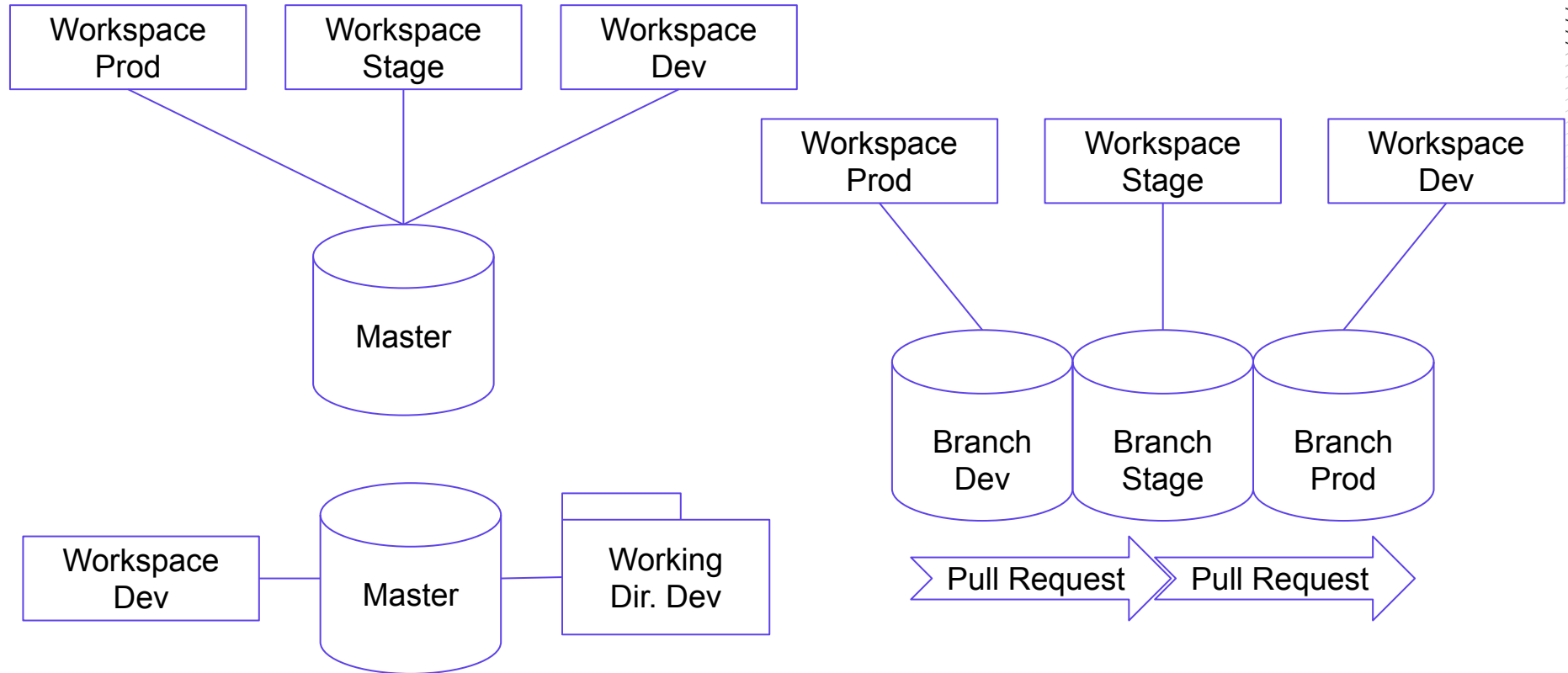
# Workspaces



- Organize and decompose monolithic infrastructure into micro-infrastructures.
- Match the organization of your application or teams with your infrastructure.
- “Micro-infrastructures” are linked to create the complete infrastructure for the application.



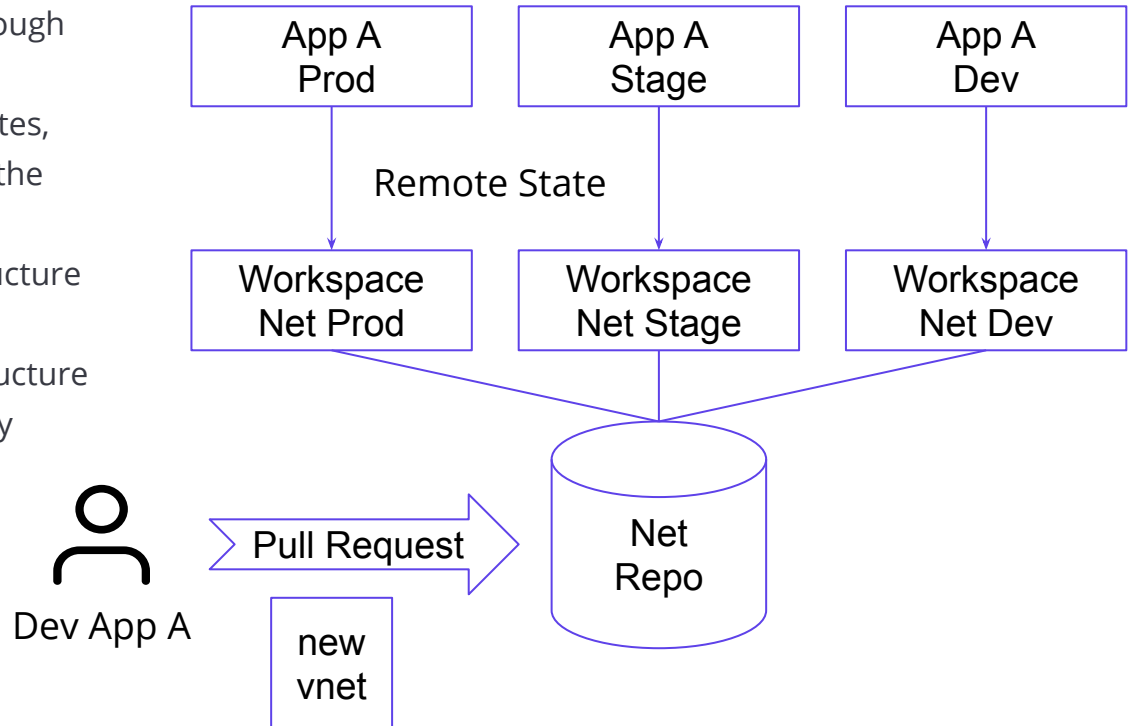
# VCS Structures





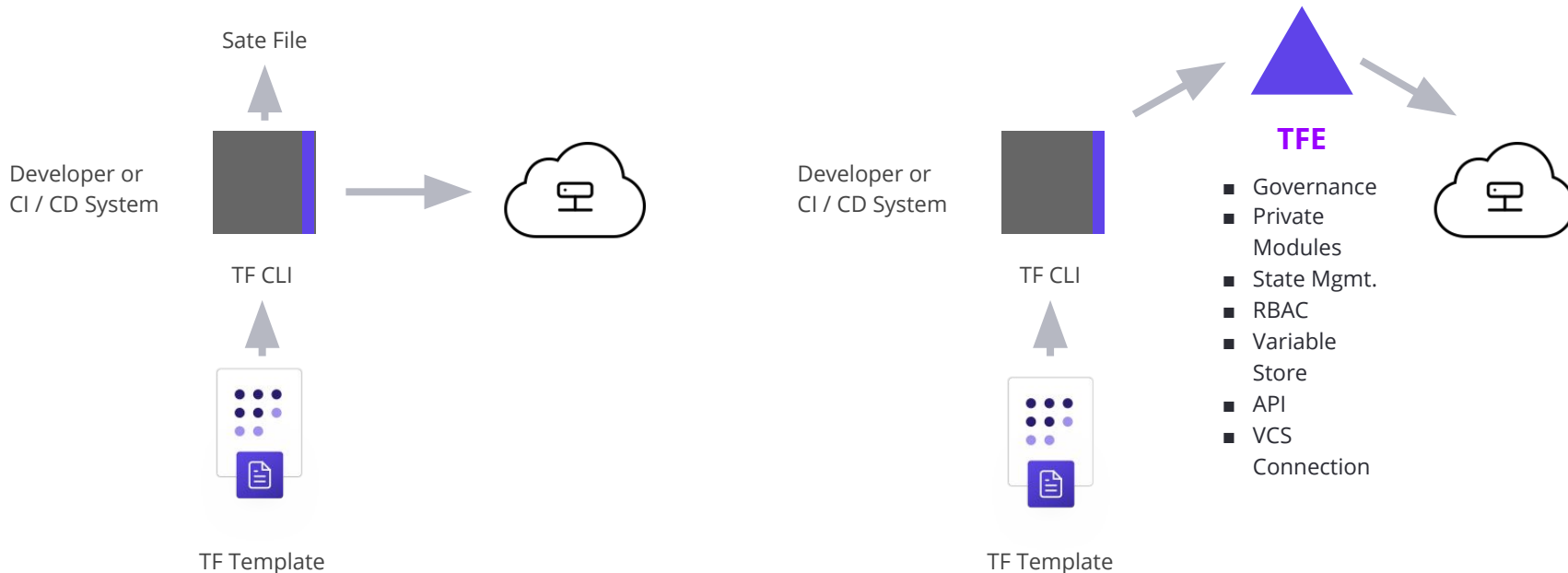
# GitOps

- Teams might operate in the producer/consumer model through separated workspaces.
- When teams access remote states, they can issue pull requests to the backing repository, in order to change the underlying infrastructure if new requirements arise.
- Teams in charge of the infrastructure can review and approve or deny changes.



# Cloud Provisioning with Terraform Ent.

## A common Cloud Operating Model



Codified policies enforce security, compliance, and operational best practices across all cloud provisioning



HashiCorp

**Terraform**

**Demo**



# Terraform and ServiceNow

The image displays three overlapping screenshots of the ServiceNow Service Management console, illustrating the integration of Terraform. The top screenshot shows the 'Catalogs' page with a search bar and a 'Terraform' catalog entry. The middle screenshot shows the 'Terraform Resources' page with a list of actions: 'Apply Run', 'Create Run', 'Create', and 'Provision'. The bottom screenshot shows the 'Provision Resources' form, which includes a 'VCS Repository' dropdown menu set to 'Pet Server', a 'Description of Request' text area, and a 'Shopping Cart' section with 'Order Now' and 'Add to Cart' buttons. The interface includes a dark sidebar with navigation options like 'Favorites', 'Configuration', and 'Database Catalogs'.

# Terraform Self-Service Workflow

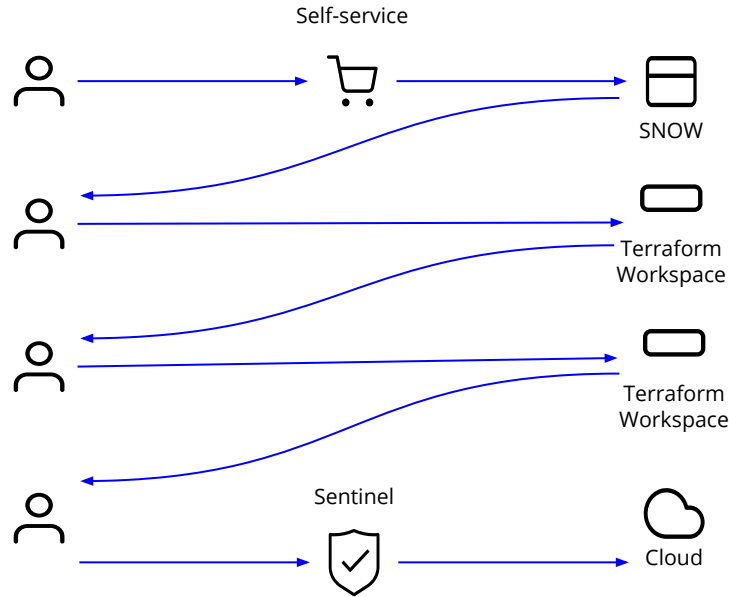


1. SS through SNOW, TFE, Pipeline, etc.

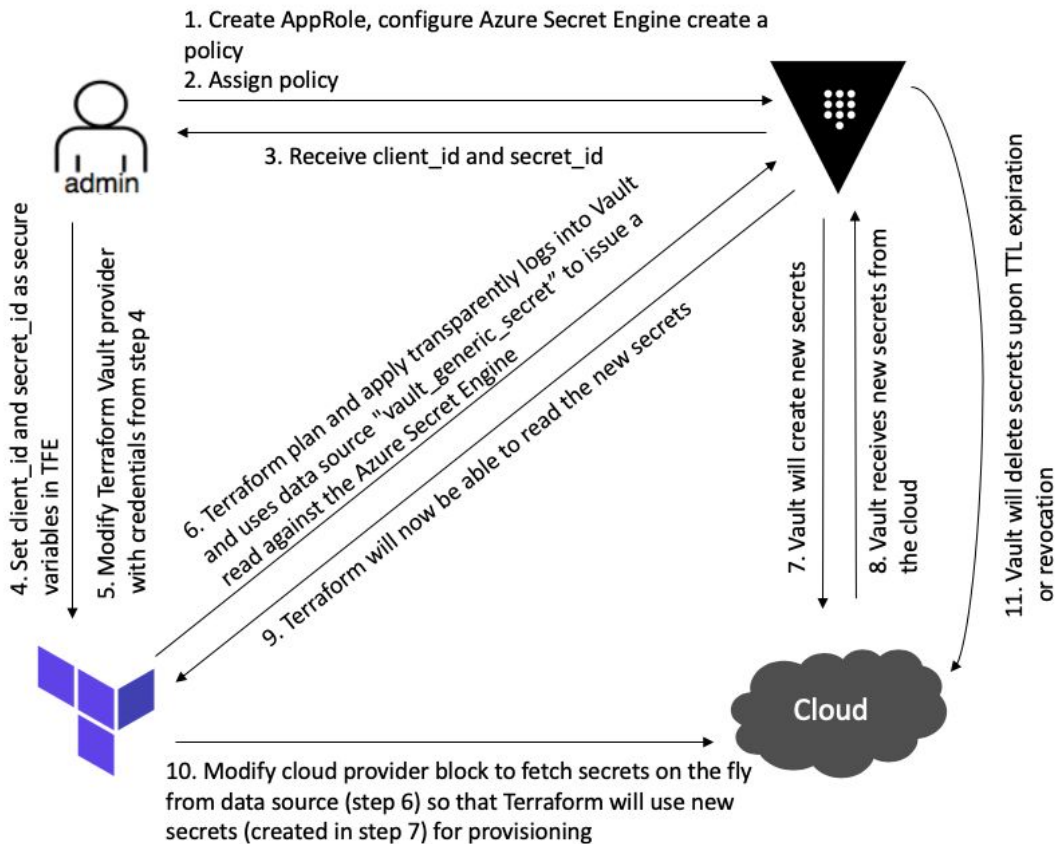
2. Receive TFE Workspace  
a) Add cloud secrets or leverage Vault

3. Bring Your Own Code  
a) Leverage private modules

4. Deploy what you need

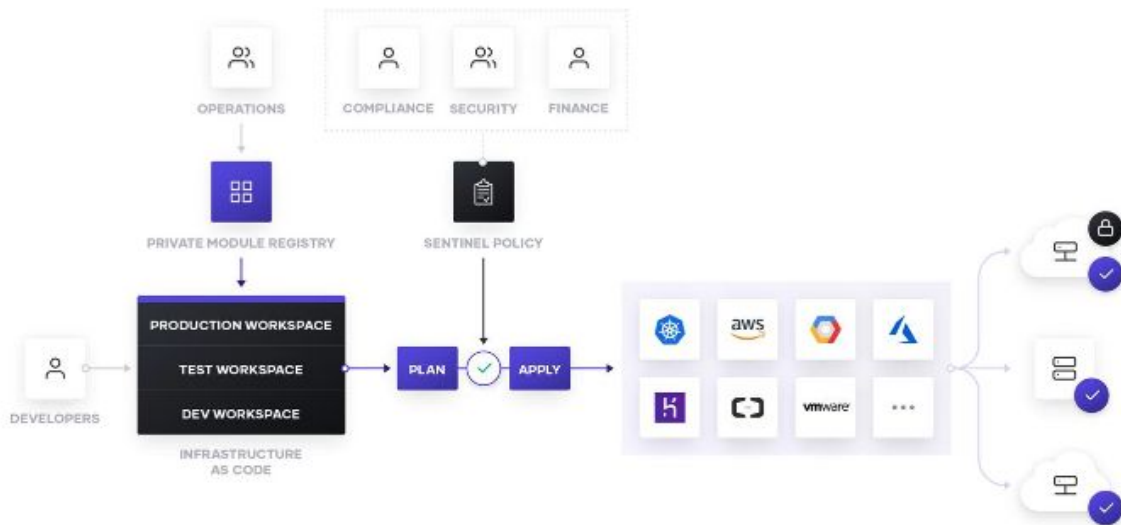


# Terraform and Vault





# Summary



**Improved Time-To-Value**

**Established a central service and common workflow**

**Maintained control over security, governance & cost.**



# Thank You

[pschulz@hashicorp.com](mailto:pschulz@hashicorp.com)

[www.hashicorp.com](http://www.hashicorp.com)