From manual cloud provisioning to one click deployments

Patrick Schulz

Sr. Solutions Engineer

pschulz@hashicorp.com

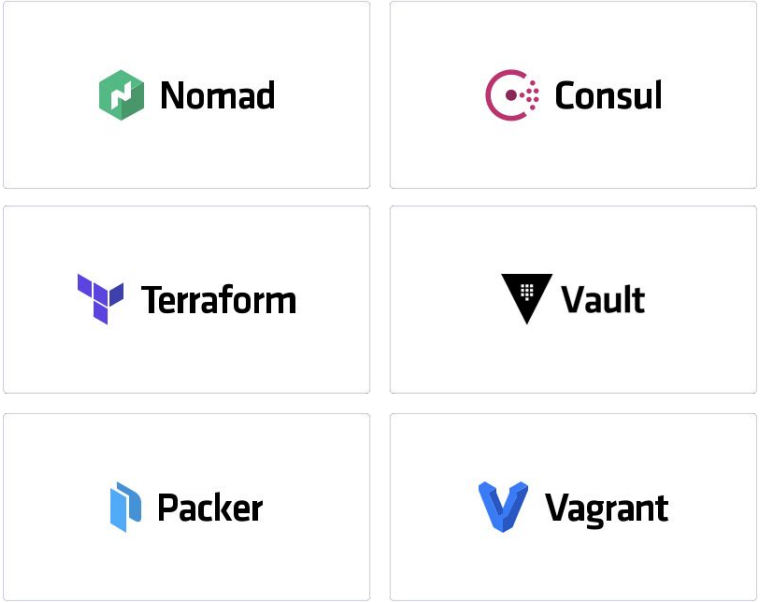
About HashiCorp

Leading Cloud Infrastructure Automation

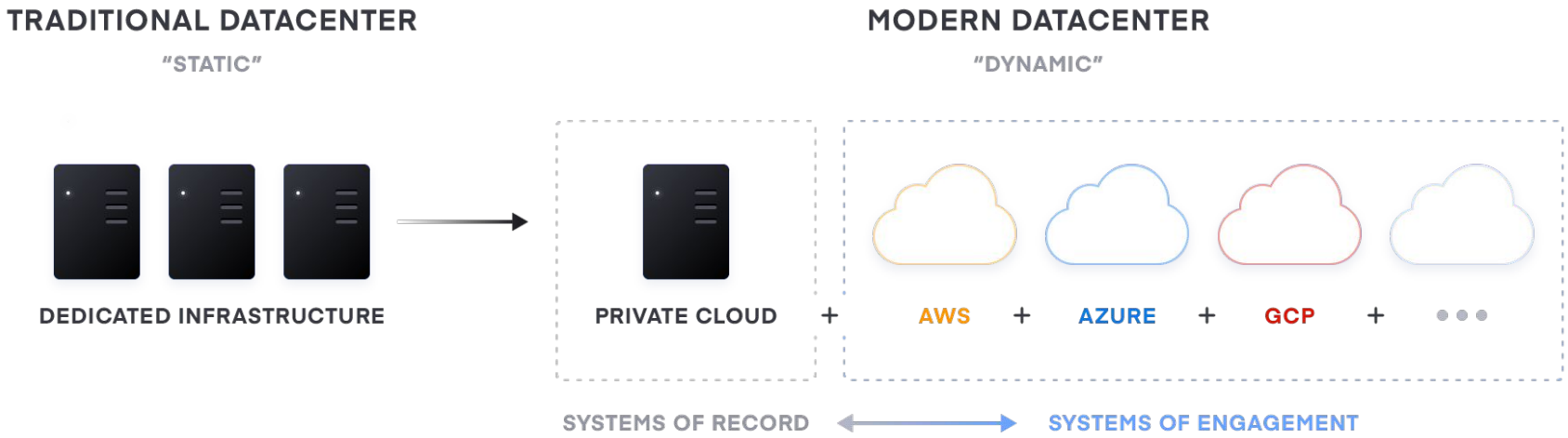
Our software stack enables the provisioning, securing, connecting and running of apps and the infrastructure to support them.

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

Founded	Employees	Funding
2012	650+	174M

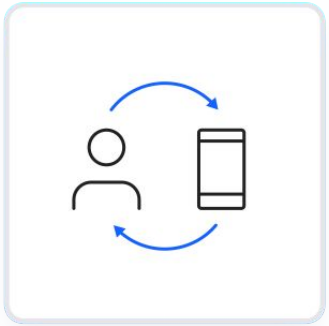


The effects of digital transformation



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?

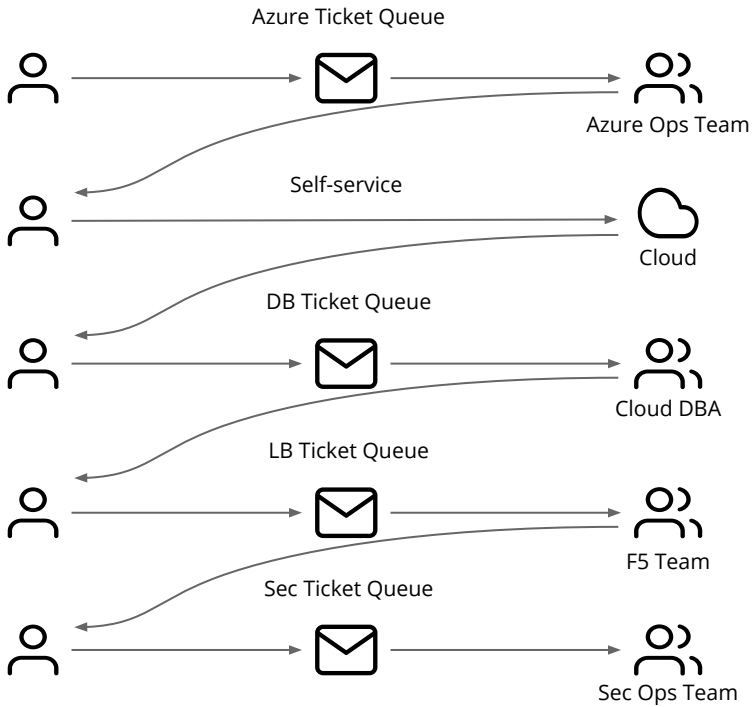
Help you to achieve your business goals by implementing a common workflow which allows for more agility through self-service.

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

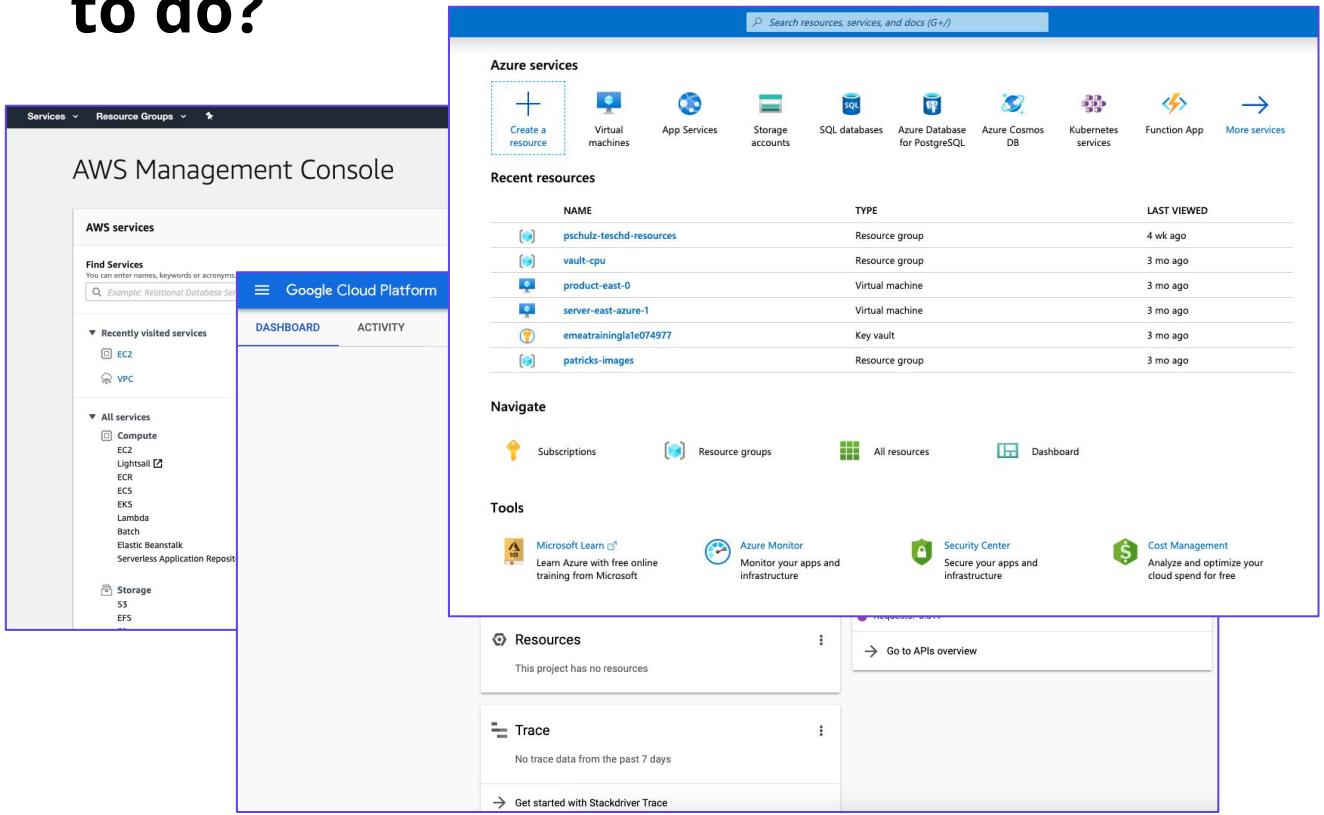
Provisioning infrastructure today...

Todays ticket-based workflow

1. Request access to Azure
2. Create VMs
3. Request DB
4. Request LB
5. Request Firewall rules



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

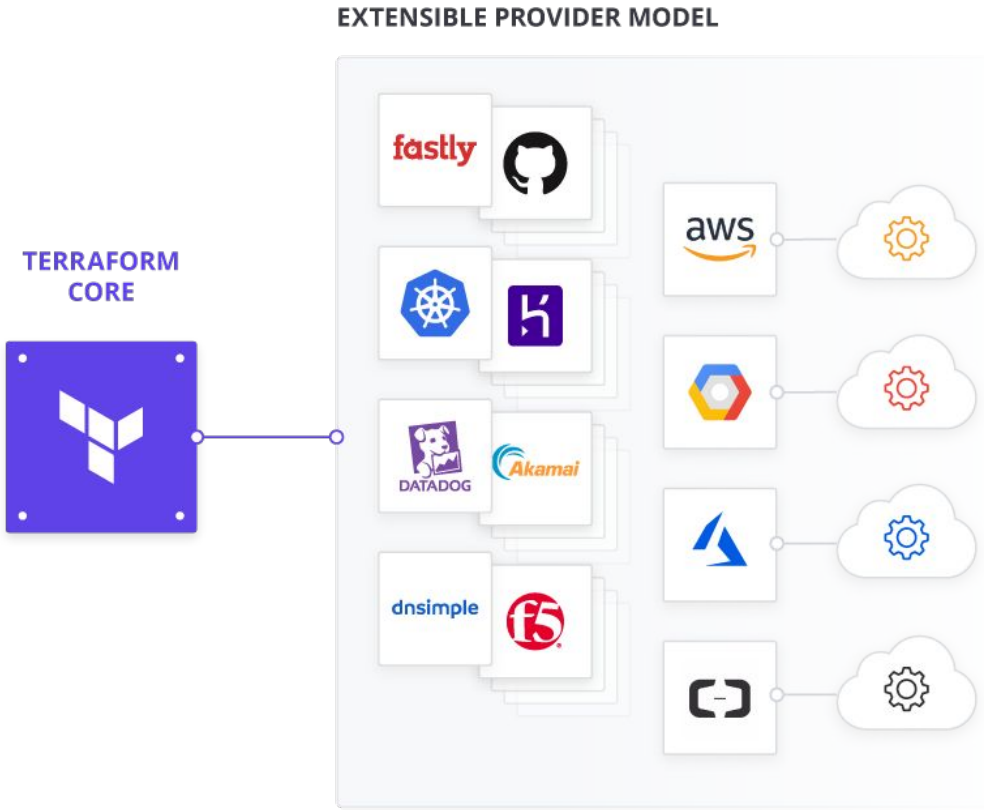
```
provider "azurerm" {}

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.

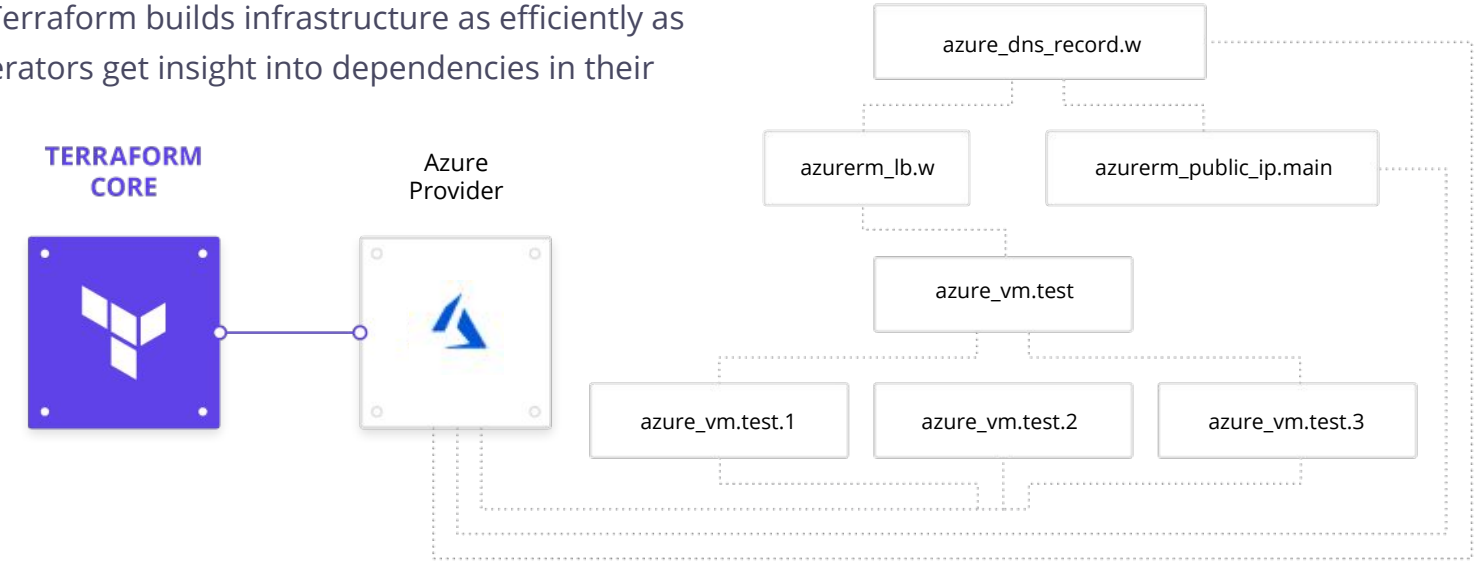
- 200+ Providers and Services



Terraform

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.

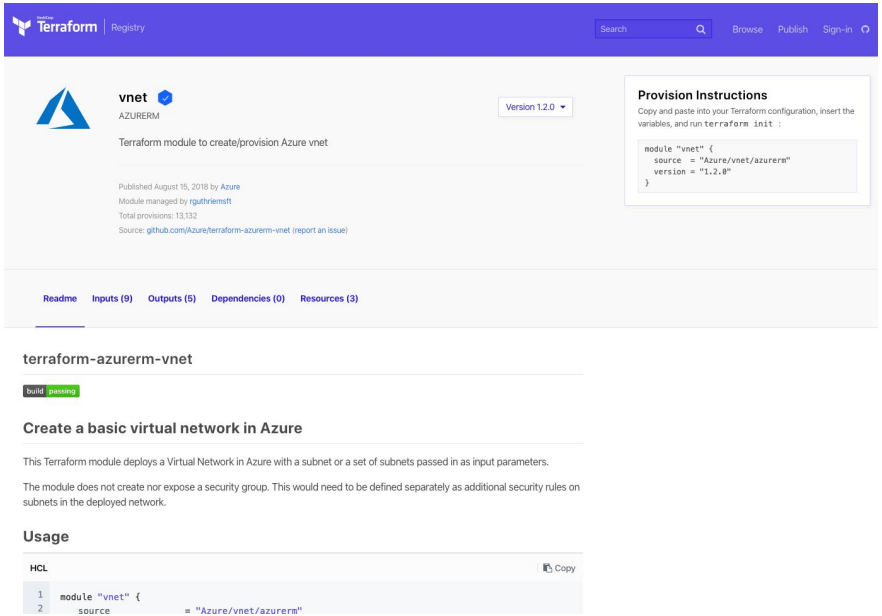


Graph only for illustration purposes.

Terraform

Modules

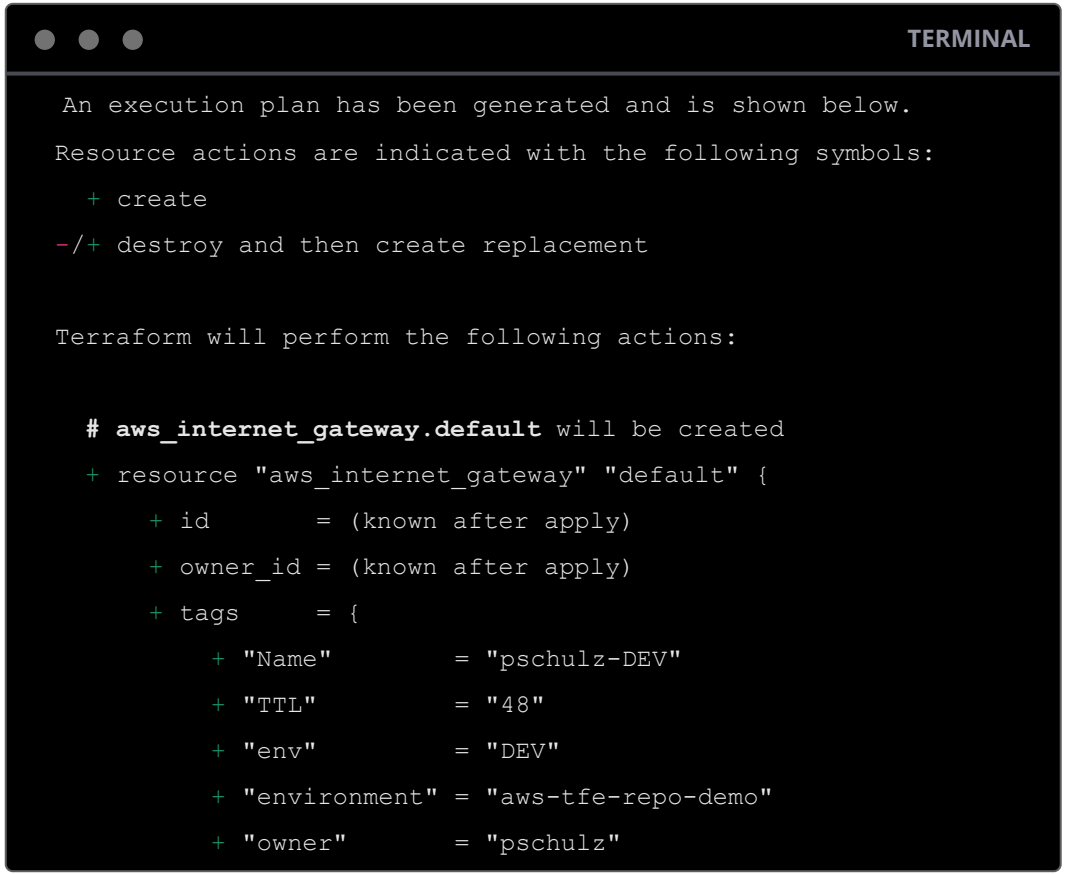
A module is a container for multiple resources that are used together. Modules can be used to create lightweight abstractions, so that you can describe your infrastructure in terms of its architecture, rather than directly in terms of physical objects.



Terraform

Plan

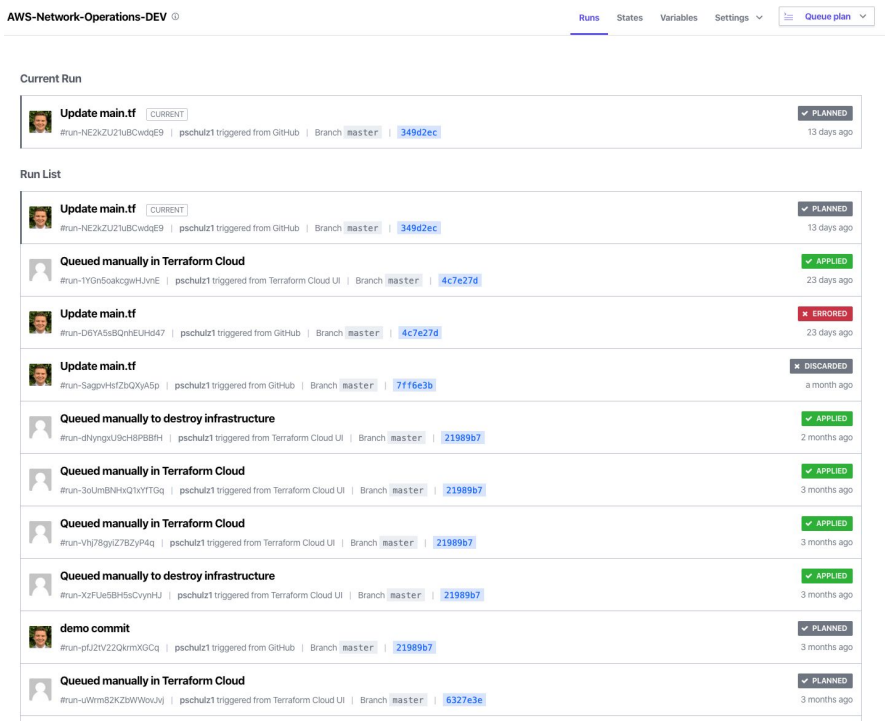
Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.



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.



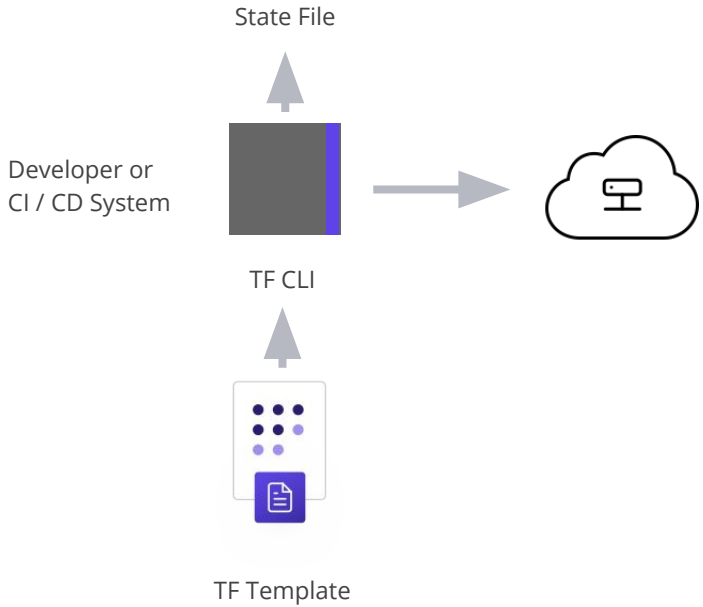
Terraform

State

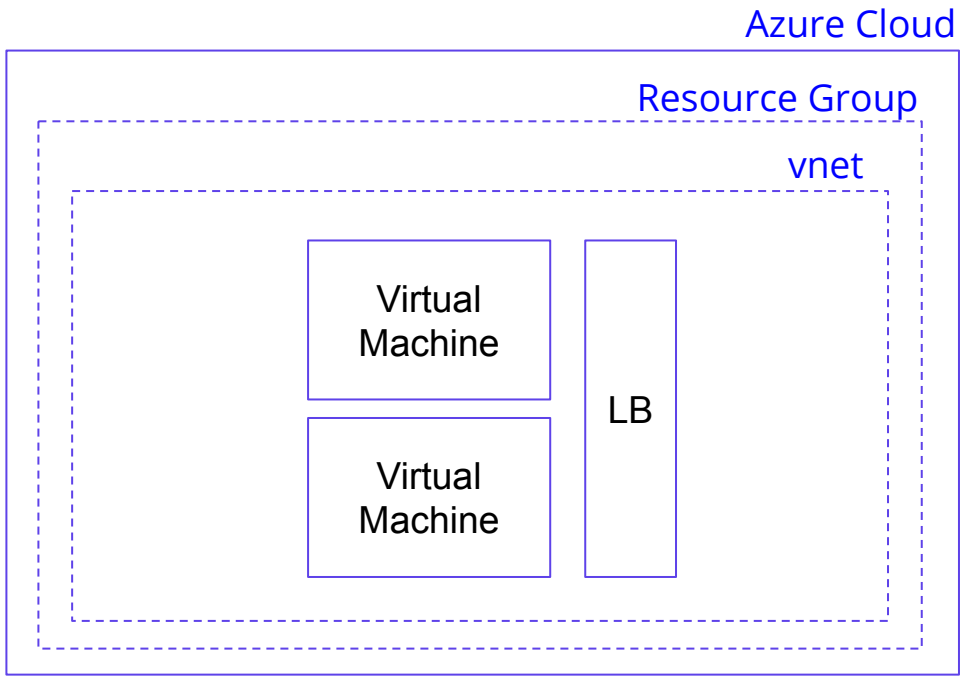
The state become a crucial part of your Terraform deployment process, as it will contain sensitive information eventually, which need to be protected.

Cloud Provisioning with Terraform OSS

A common Cloud Operating Model

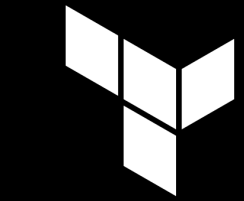


Infrastructure diagram



Lets we need a couple of new resources:

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



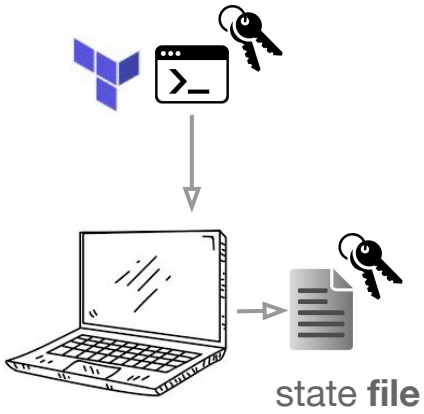
HashiCorp
Terraform

Demo

What's the Problem with OSS?

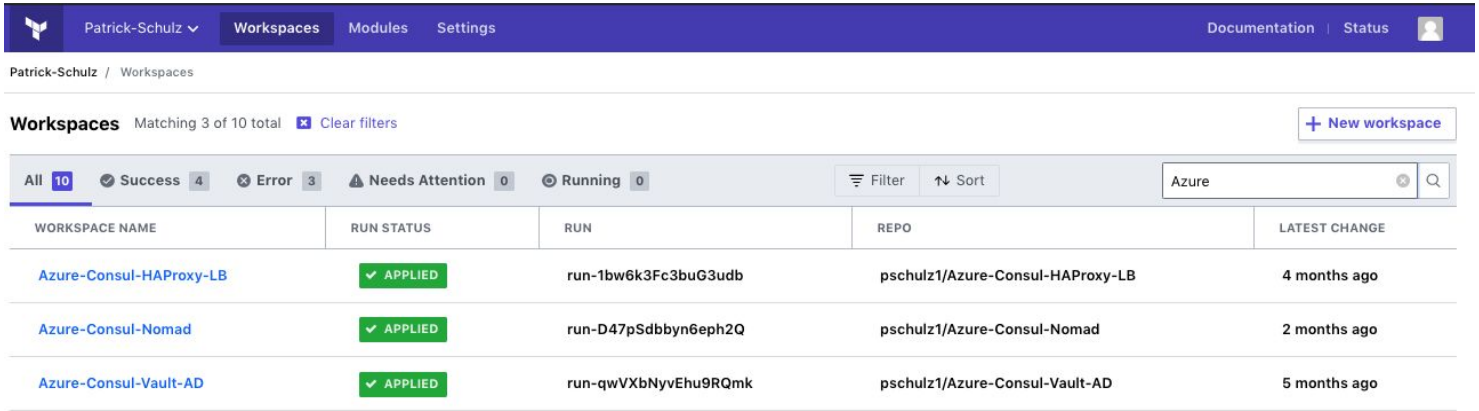
Open source doesn't scale very well, for multiple reasons:

- 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.



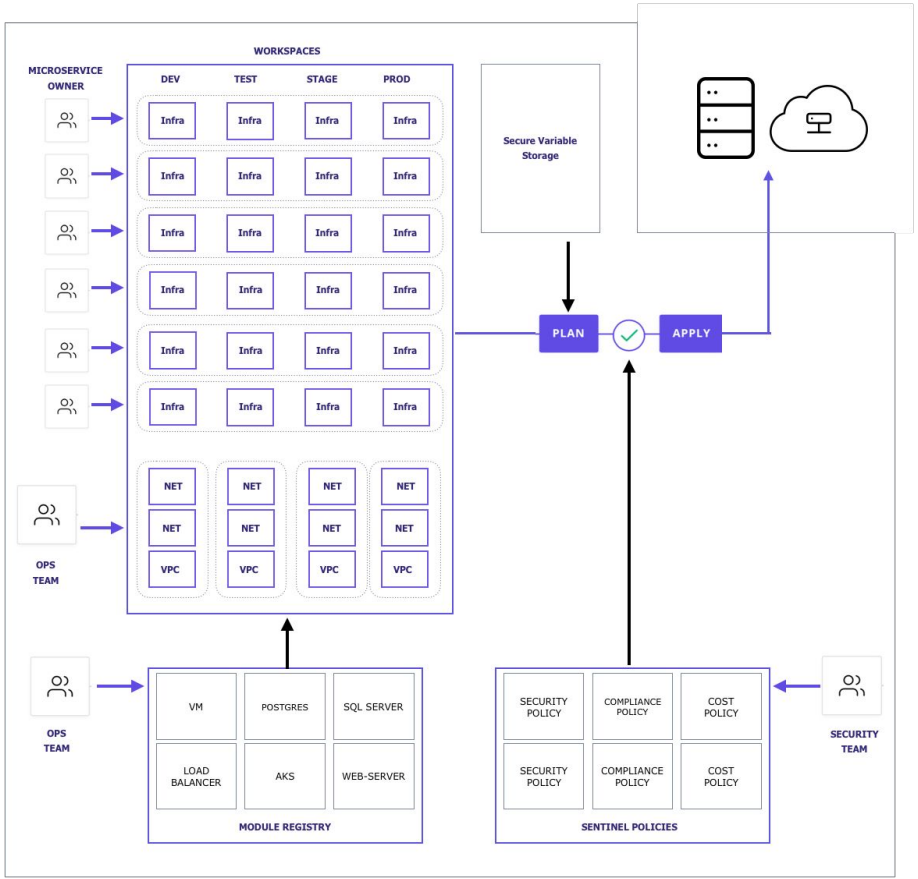
Introduction to TFE

- 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.

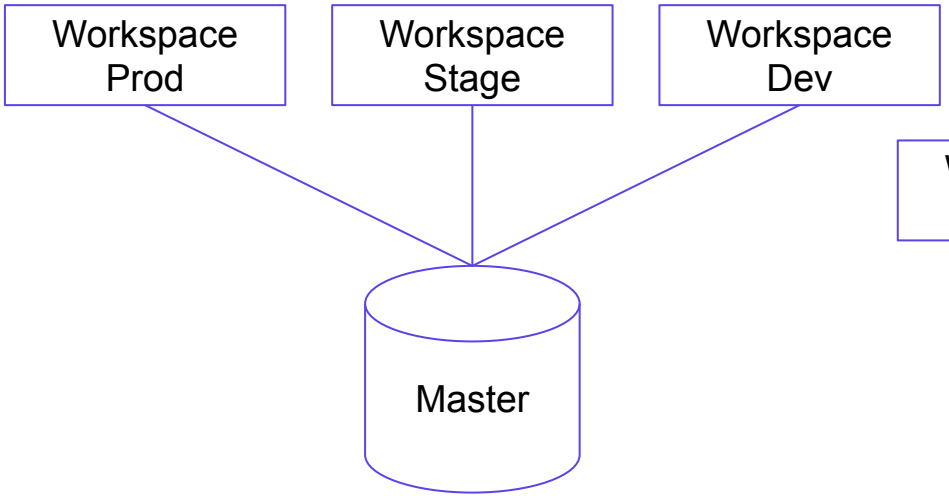


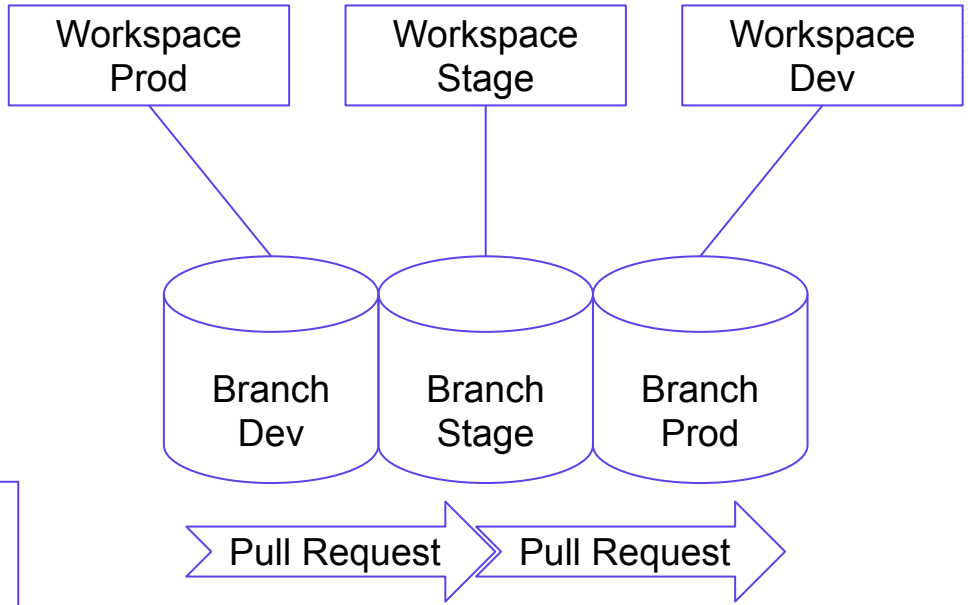
Workspaces

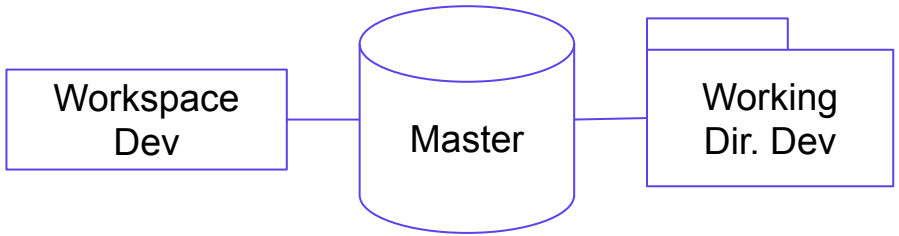
- Organize and decompose monolithic infrastructure into micro-infrastructures.
- Match the organization of your application or teams with your infrastructure.
- “Micro-infrastructures” are linked to create the complete infrastructure for the application.



VCS Structures

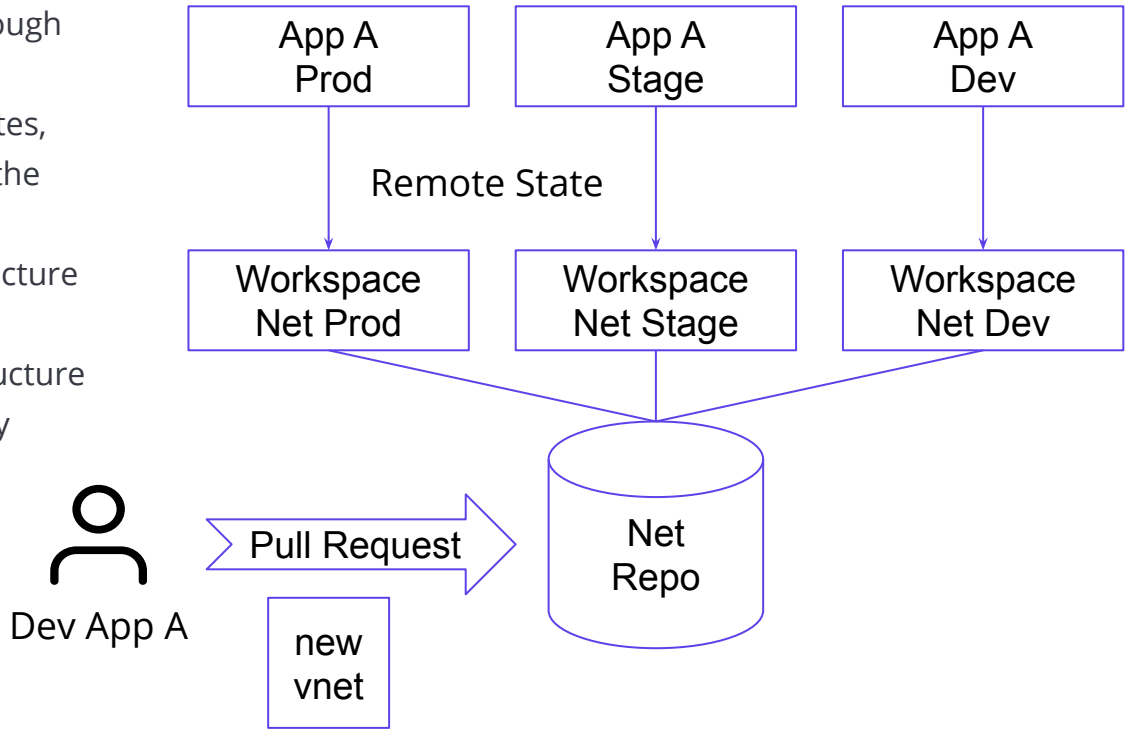






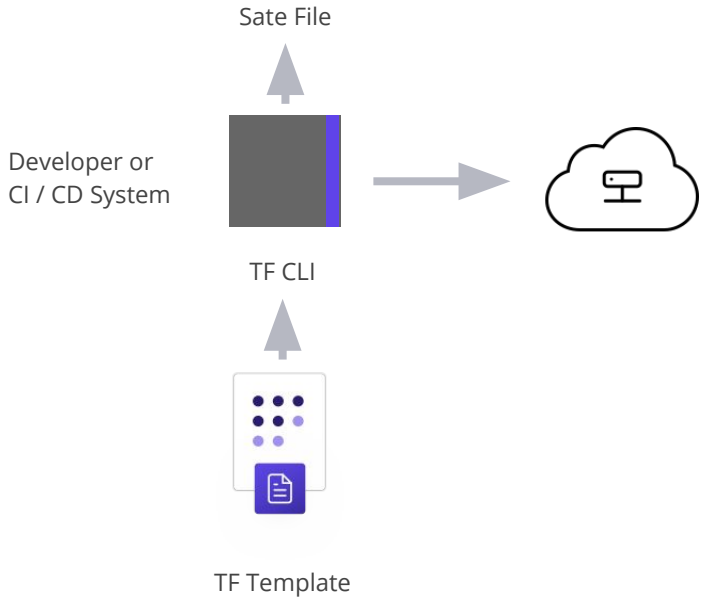
GitOps

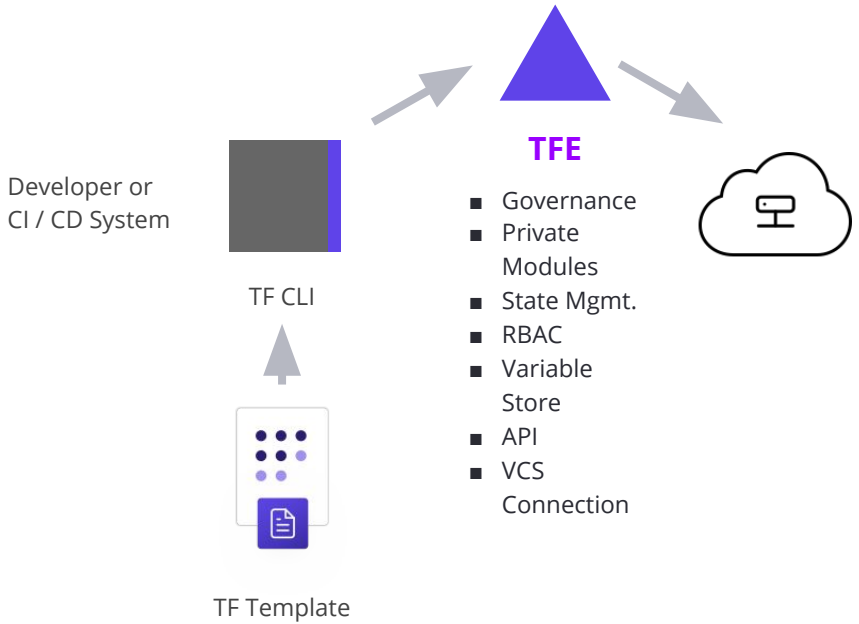
- Teams might operate in the producer/consumer model through separated workspaces.
- When teams access remote states, they can issue pull requests to the backing repository, in order to change the underlying infrastructure if new requirements arise.
- Teams in charge of the infrastructure can review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



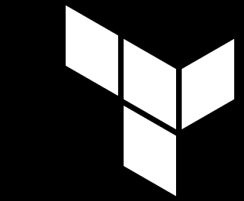
Cloud Provisioning with Terraform Ent.

A common Cloud Operating Model





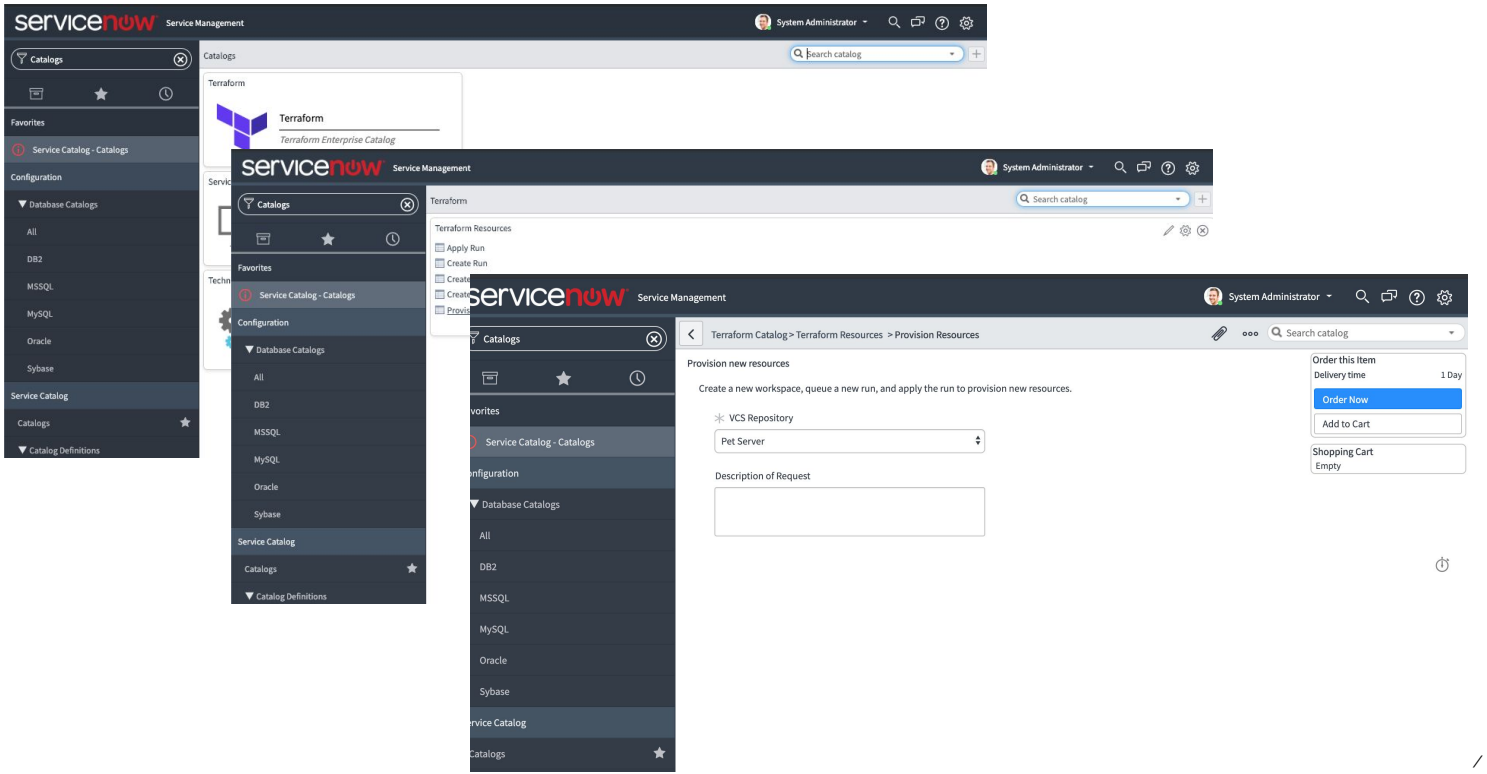
Codified policies enforce security, compliance, and operational best practices across all cloud provisioning



HashiCorp
Terraform

Demo

Terraform and ServiceNow



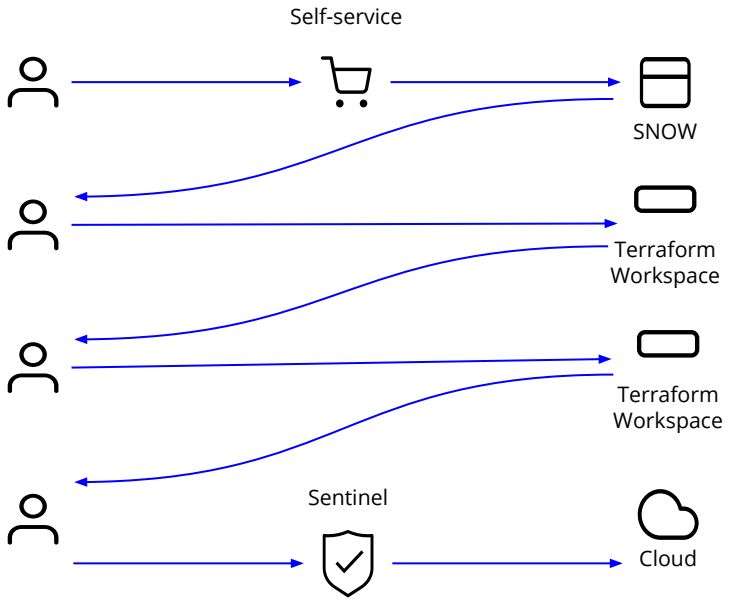
Terraform Self-Service Workflow

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

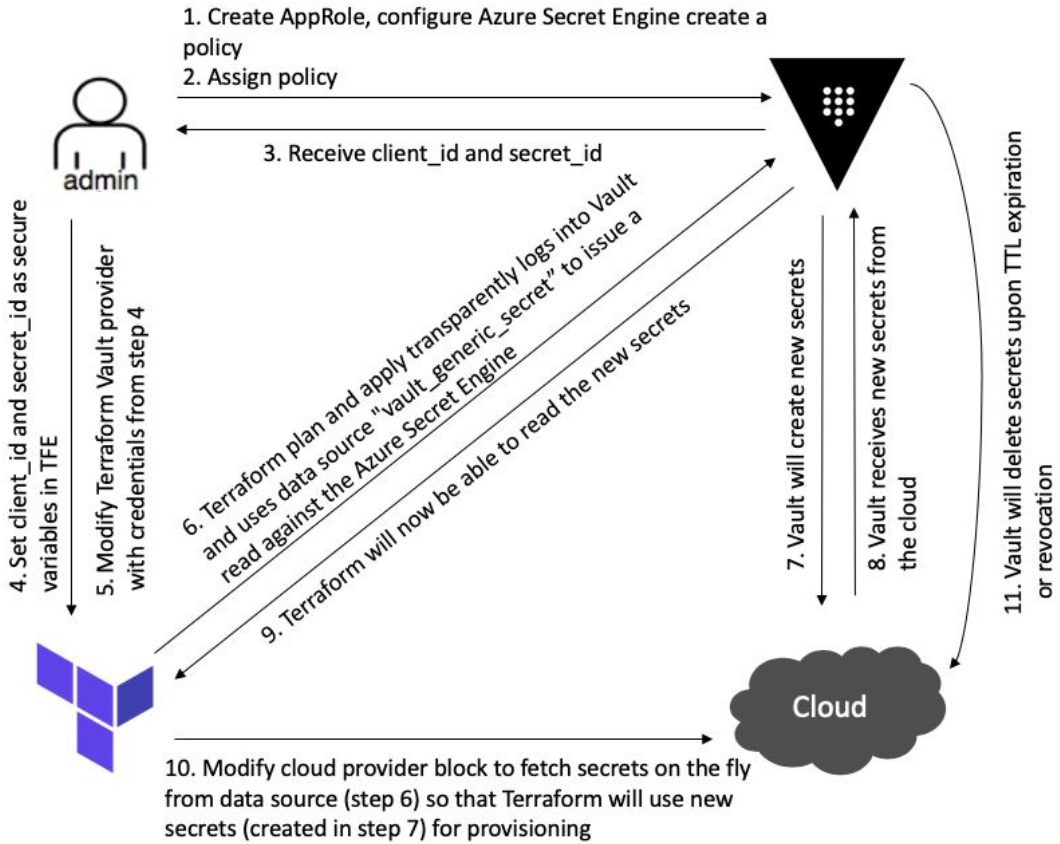
2. Receive TFE Workspace
   a) Add cloud secrets or leverage Vault

3. Bring Your Own Code
   a) Leverage private modules

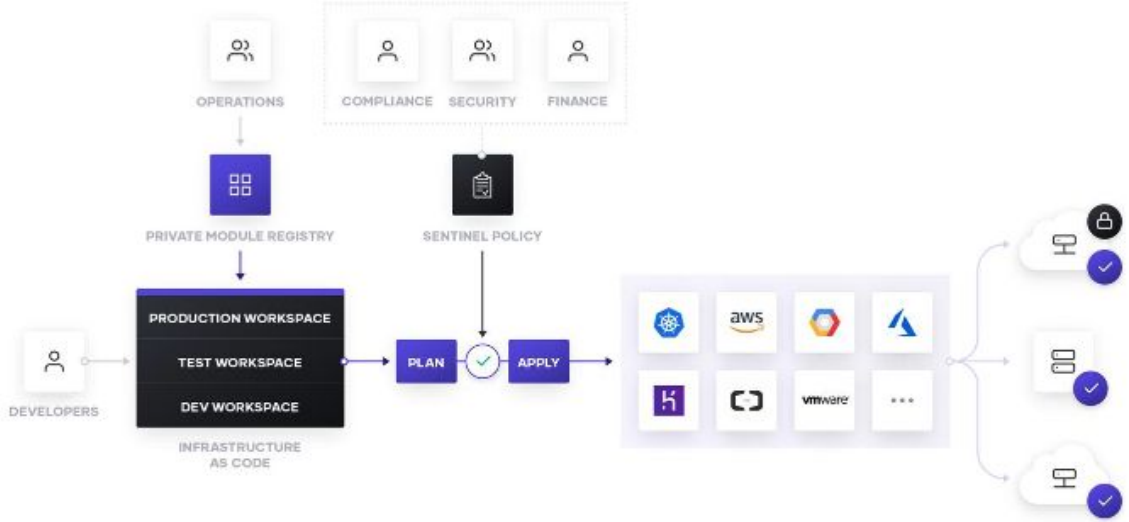
4. Deploy what you need



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
www.hashicorp.com