Infrastructure Layers

Cloud API + Terraform = Empathetic GitOps

Infrastructure Engineer

“Simple”

“Complexity kills people”

Problems

Business

- Ideas take months to reach customers
- New hires take months to be valuable
- Unknown security state
- Uncontrollable infra spend

Engineering

- Services take months to reach customers
- New hires take months to train
- Unknown infra state
- Uncontrollable tech debt

Takeaways

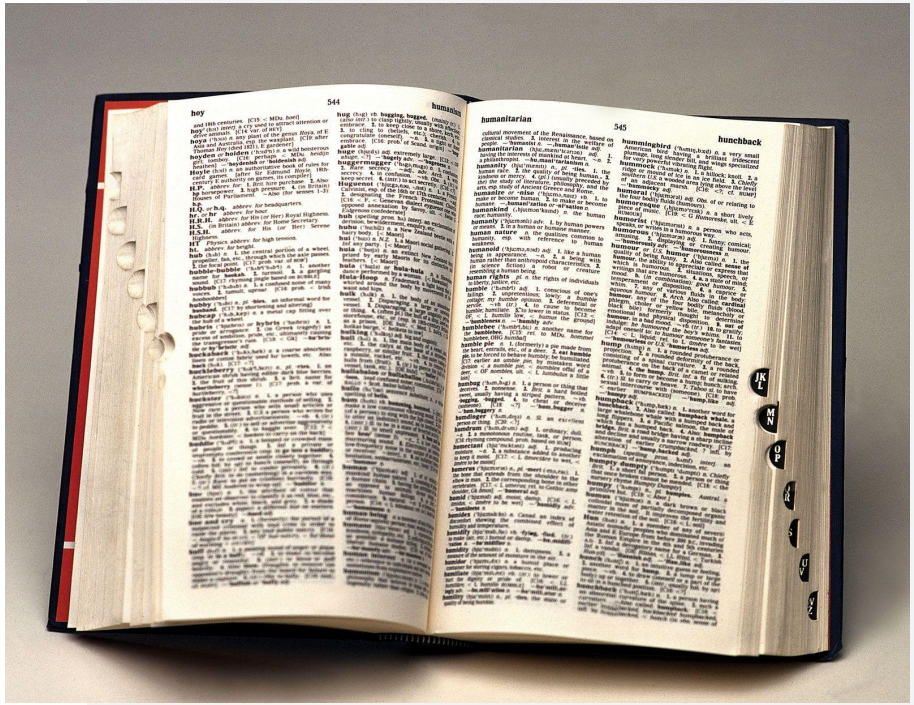
Problem	Solution
Engineering Culture	Infrastructure Layers
Known infra state	Git
Datacenter "think"	Cloud APIs
Service Menu	Terraform Modules

Empathy based Infrastructure Culture

Definitions

Definitions

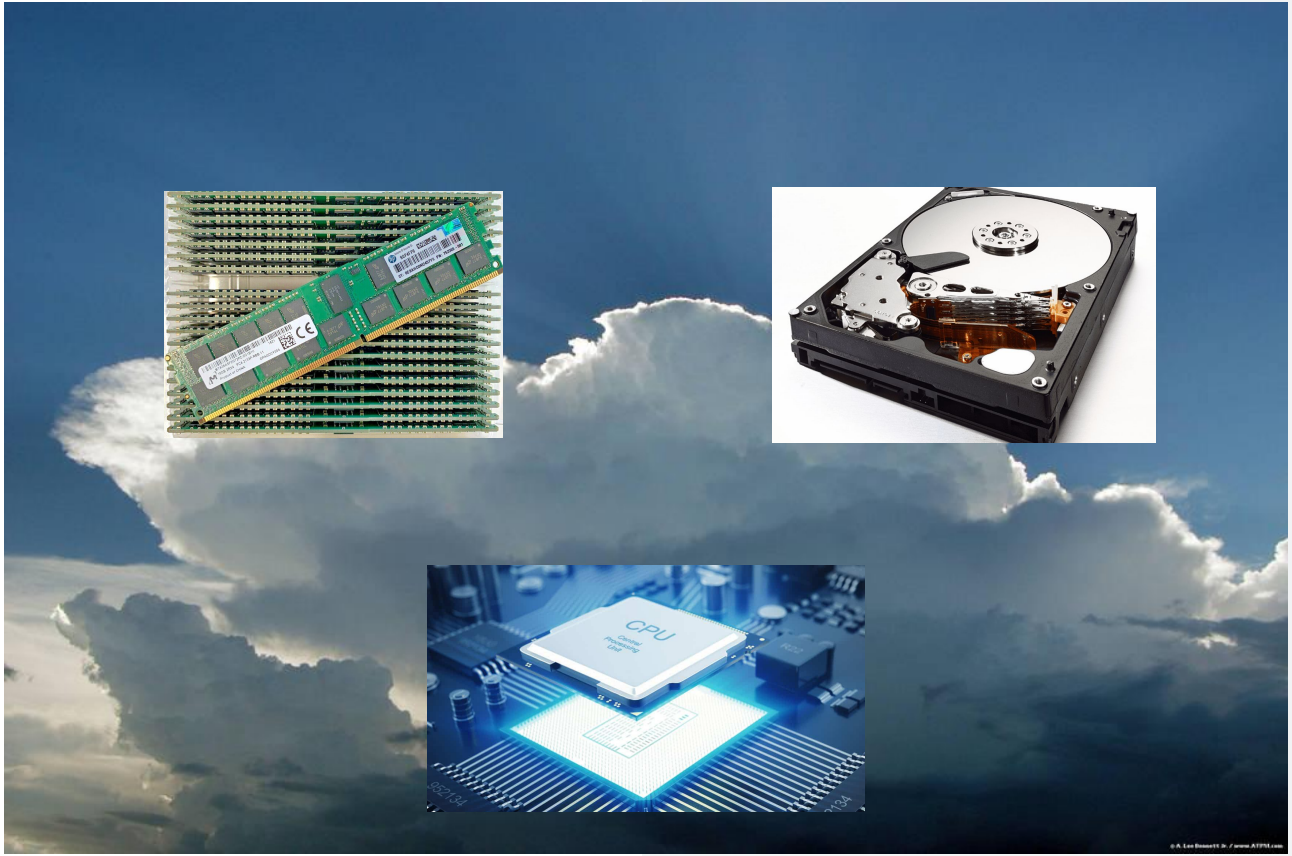
- “The Cloud”
- “DevOps”
- Infrastructure as Code (IaC)
- Infrastructure Layers
- Git & GitOps
- “State”
- Zero Trust



“The Cloud”

A collection of physical machines pooled together and offered to end users to run workloads.

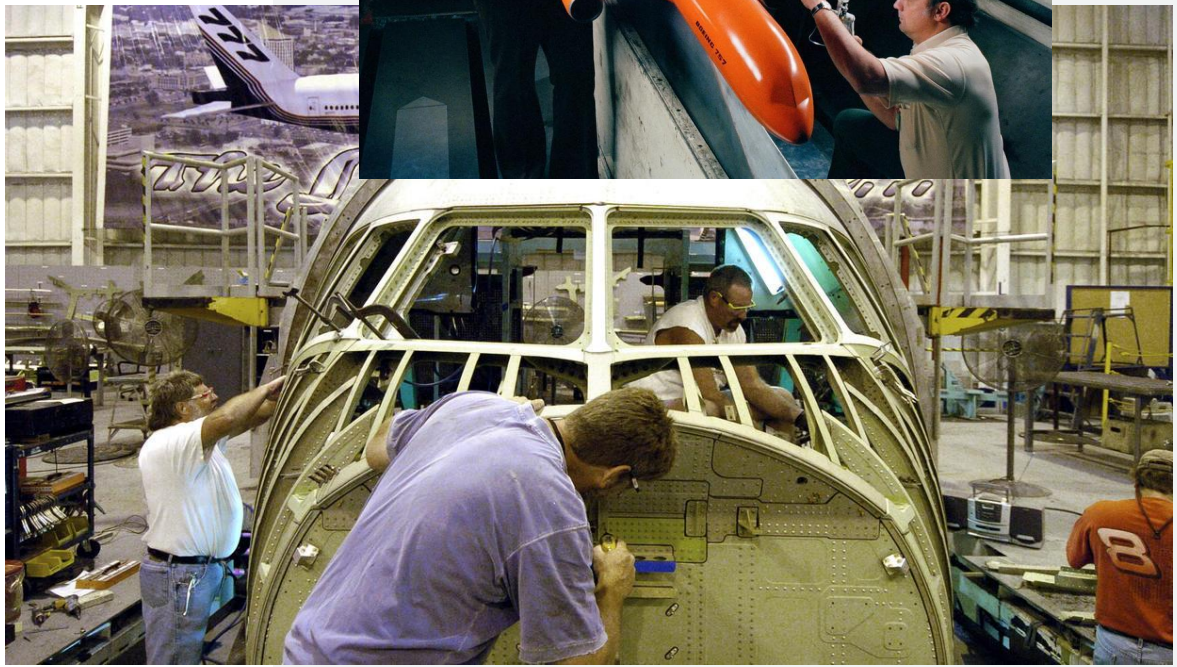
“The Cloud”



“DevOps”

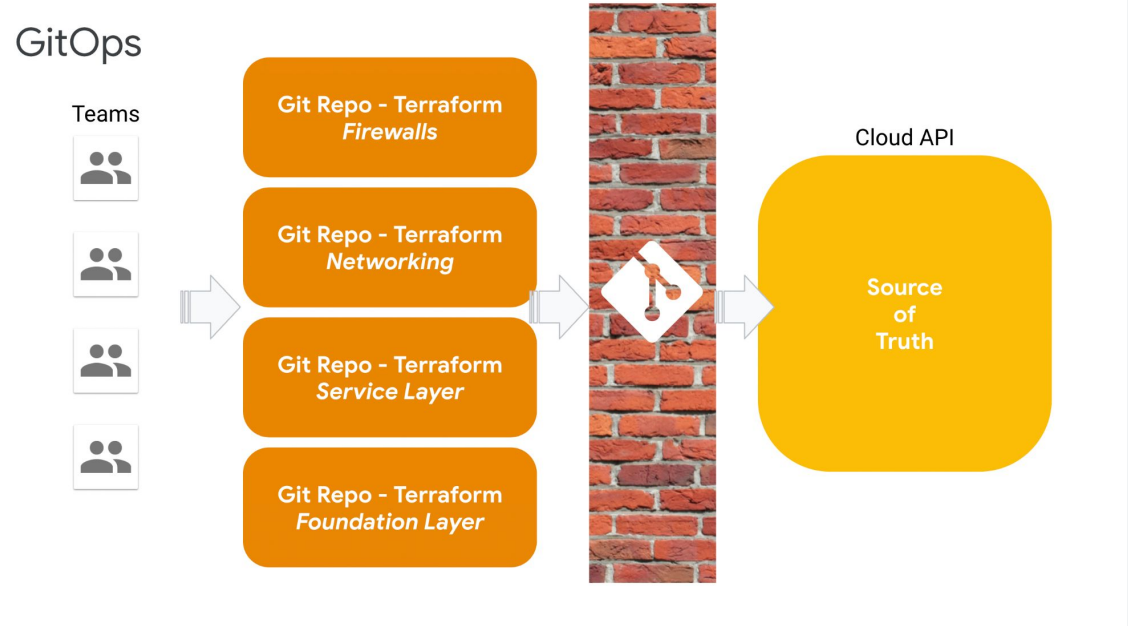
A collection of humans working together to ship value to the customers.

- *Not a team*
- *Not a title*



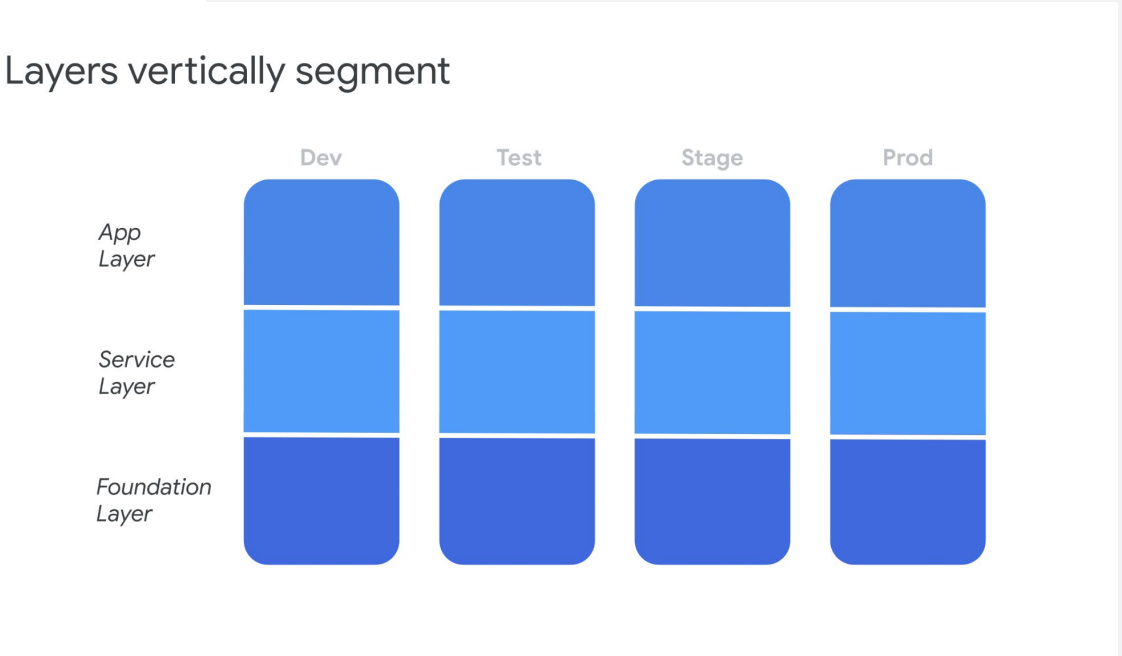
“Infrastructure as Code aka IaC”

Suggesting your own desired state of the infrastructure, in code, in a Git Pull Request.



“Infrastructure Layers”

Grouping of the IaC into silo’d Git Repos in order to provide empathy for all engineers involved.



“Git & GitOps”

Git: distributed

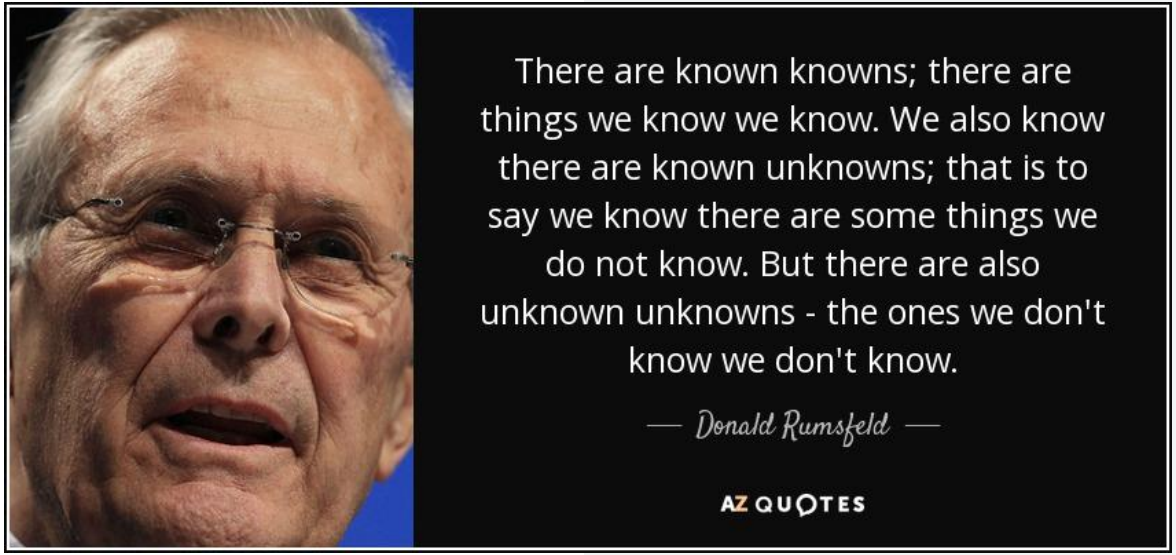
version control system

GitOps: “Git + CiCd”

git

“State”

The source of truth



“Zero Trust”

Nothing is trusted =

- *No person*
- *No device*
- *No network*

The only outcome of trust is some type of betrayal

Datacenter to the Cloud

A Brief History

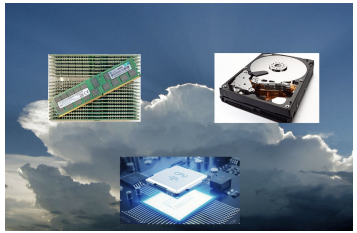
“Datacenter think”

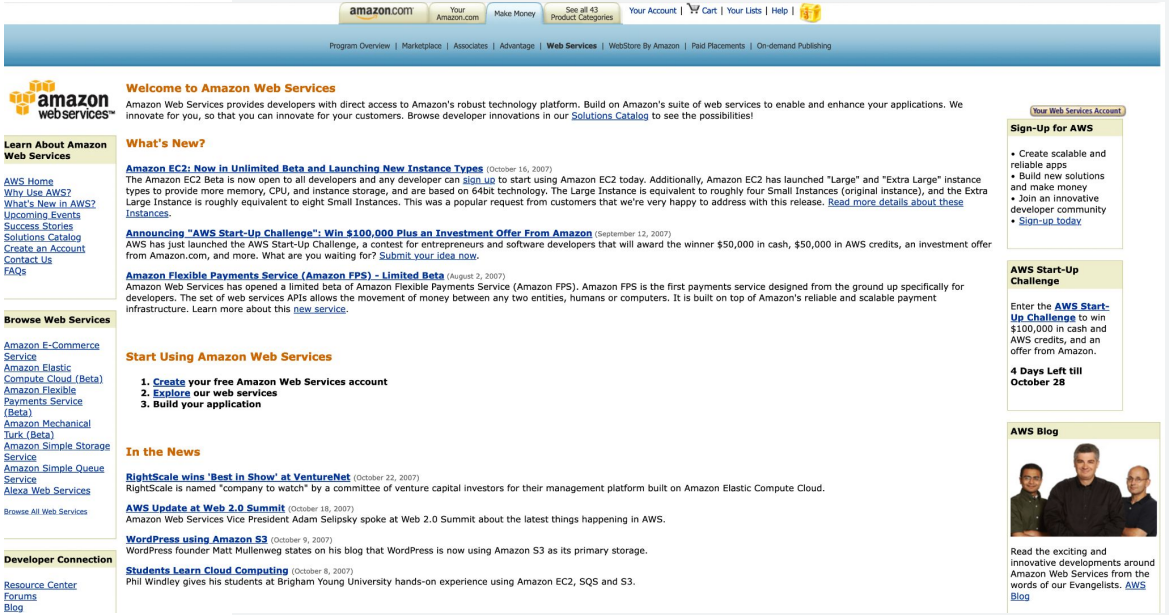
A building & a cage with lots of:

- ➔ people
- ➔ power
- ➔ access methods
- ➔ bare metal

The "CLOUD"

- No buildings & no cages
- less people
- no power
- less access methods
- no bare metal

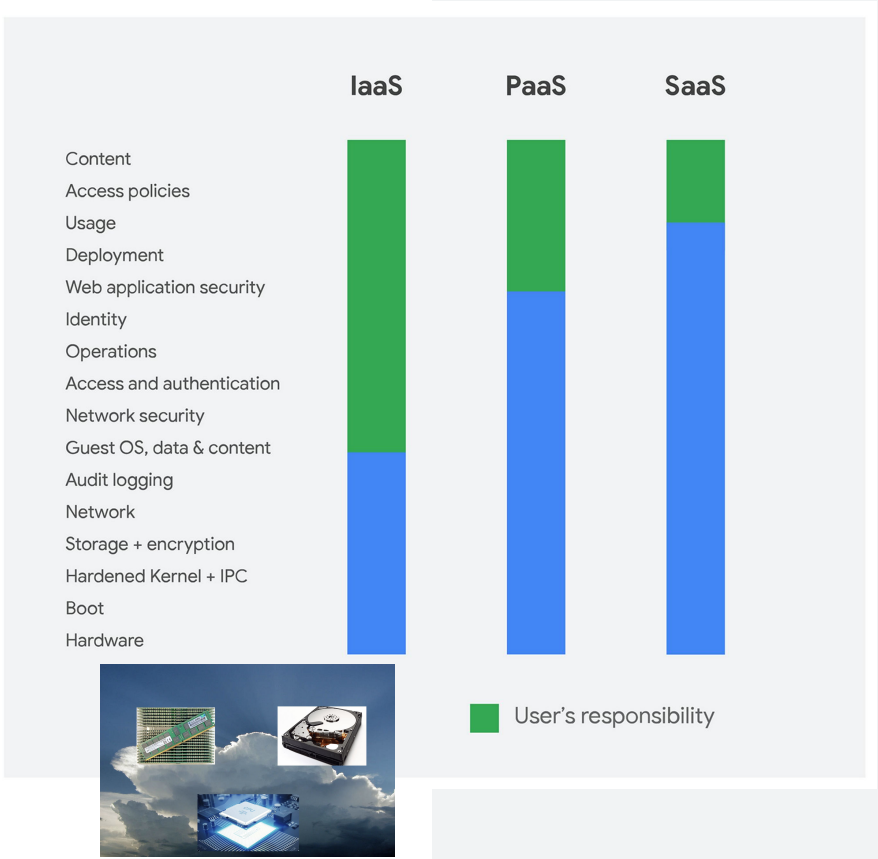




> Cloud native techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

amazon
web services

“Cloud Shared Responsibility Model”



Infrastructure as Code

> A server should be like a phoenix, regularly rising from the ashes.

Martin Fowler

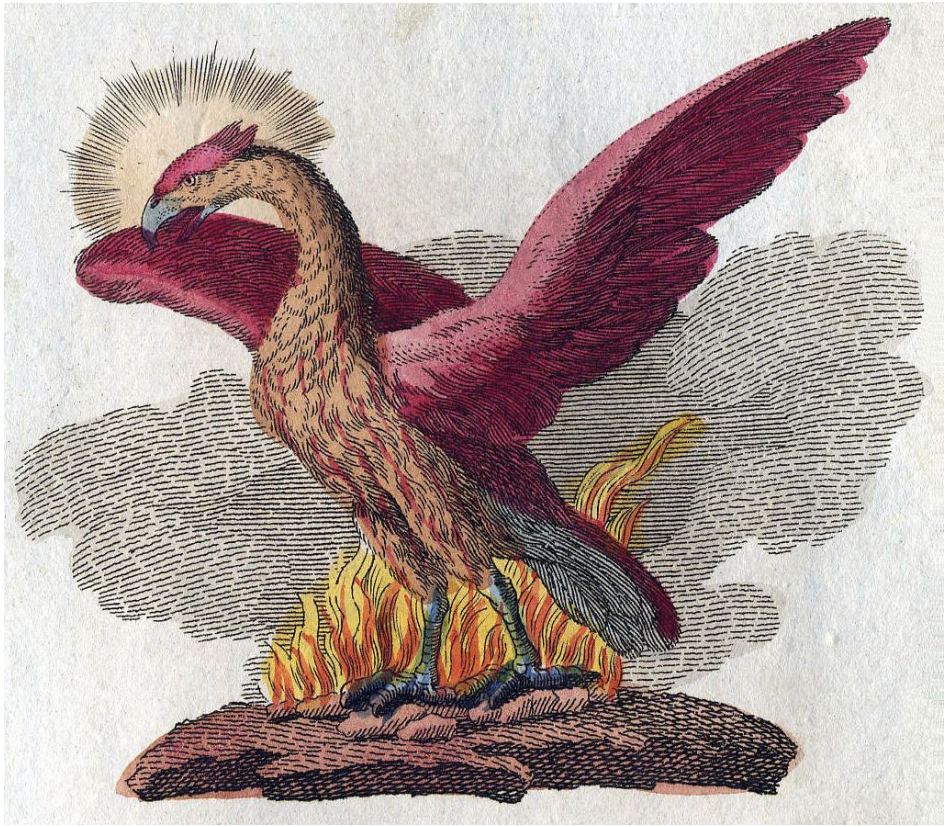
Phoenix Servers

Prevent:

- Configuration Drift
- Snowflake Server

Allow:

- Rise from the Ashes



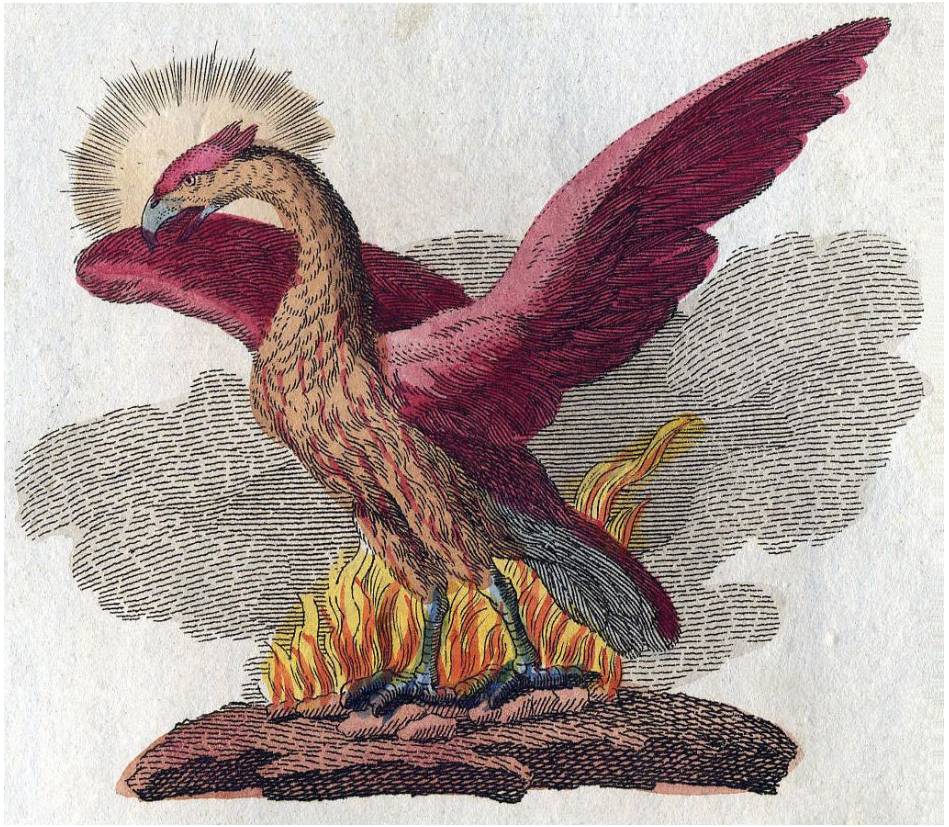
Two deployment paths to a Phoenix Server

Imperative:

- specifics

Declarative:

- consumables



distributed
infrastructure state development hosted in Git

desired infrastructure state promotion with Git ***pull requests***

Past Configuration Tools had Challenges

To deliver a **no humans in production, self-healing, secure-by-default infrastructure via GitOps**, *Infrastructure as Code must be API driven*.

API Driven IaC

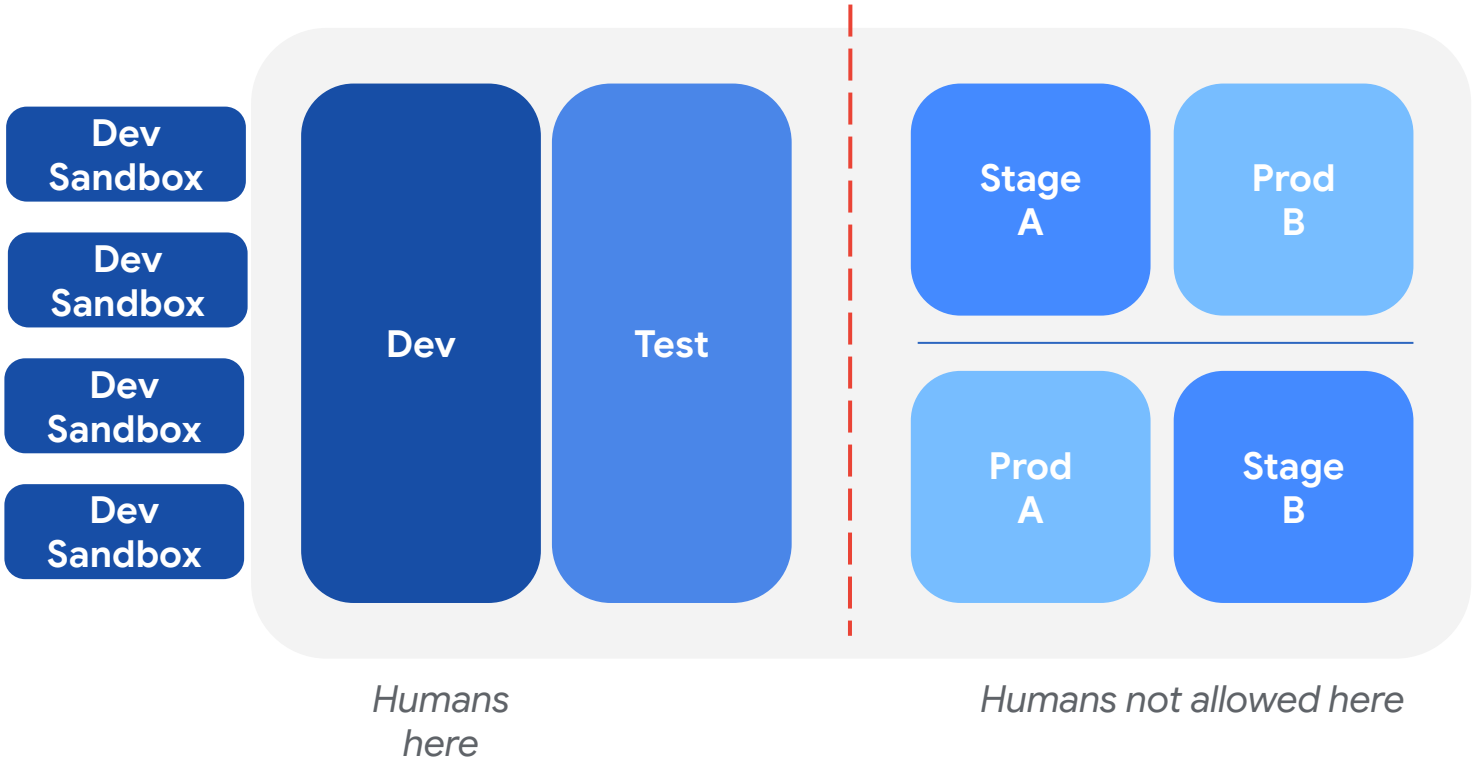
Terraform *only* speaks to APIs

- Cloud native approach
- Declare state via HCL
- Resource Graph stores desired state
- Cloud API stores source of truth
- Written in Go

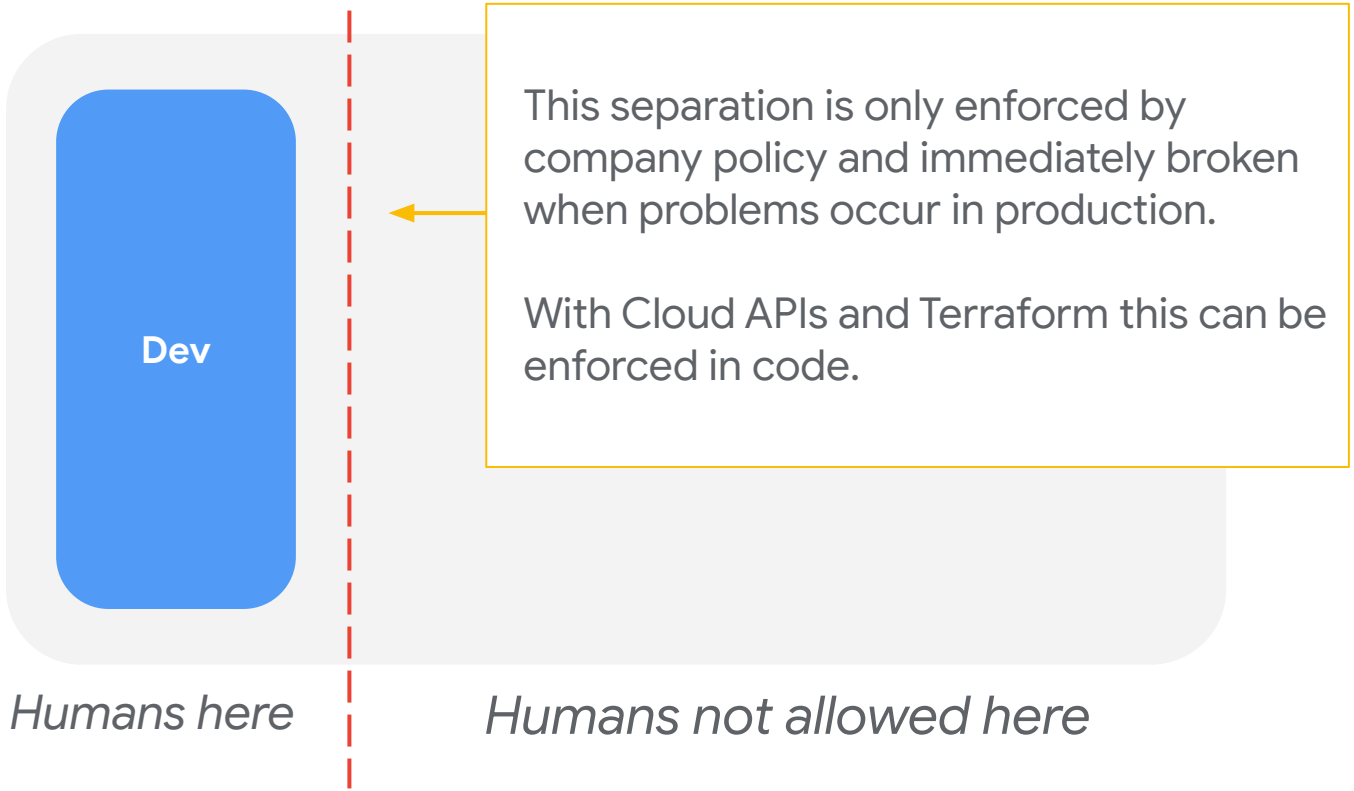
HashiCorp Terraform

Infrastructure Layers

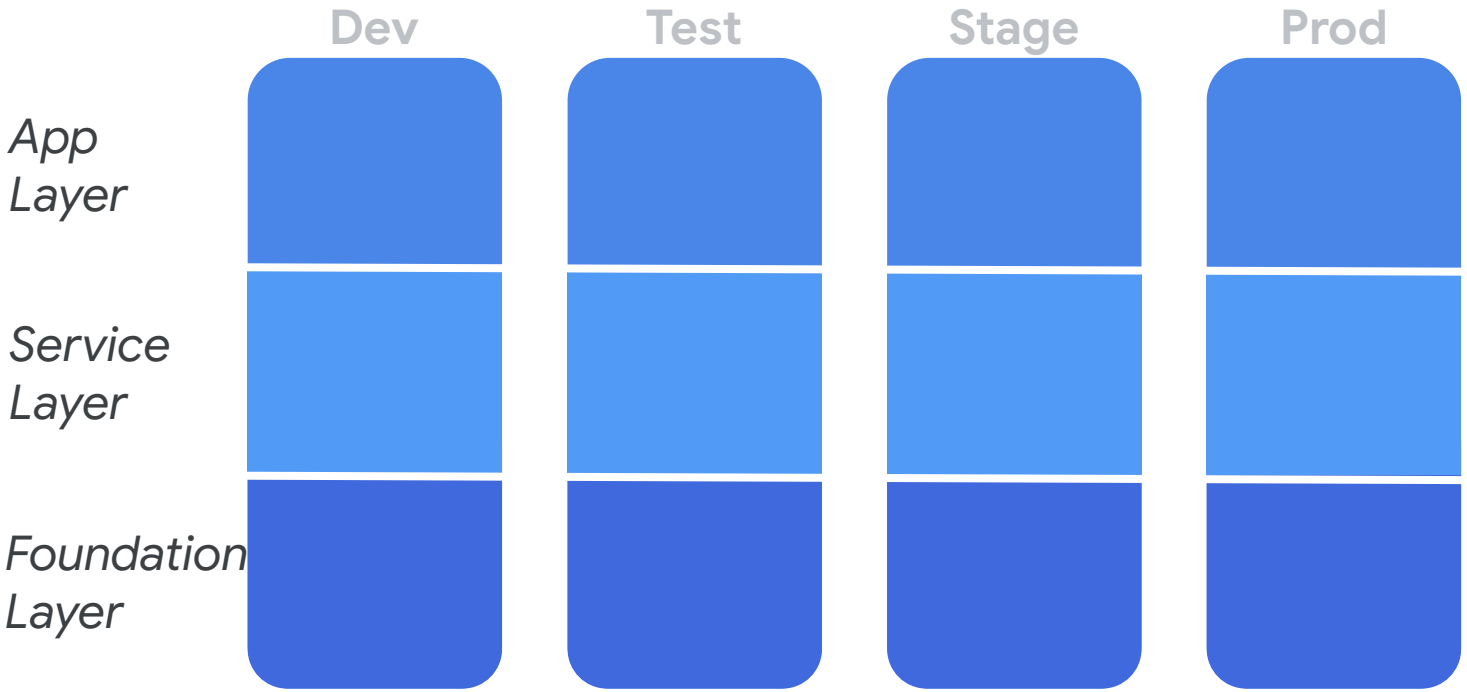
Environments separate horizontally



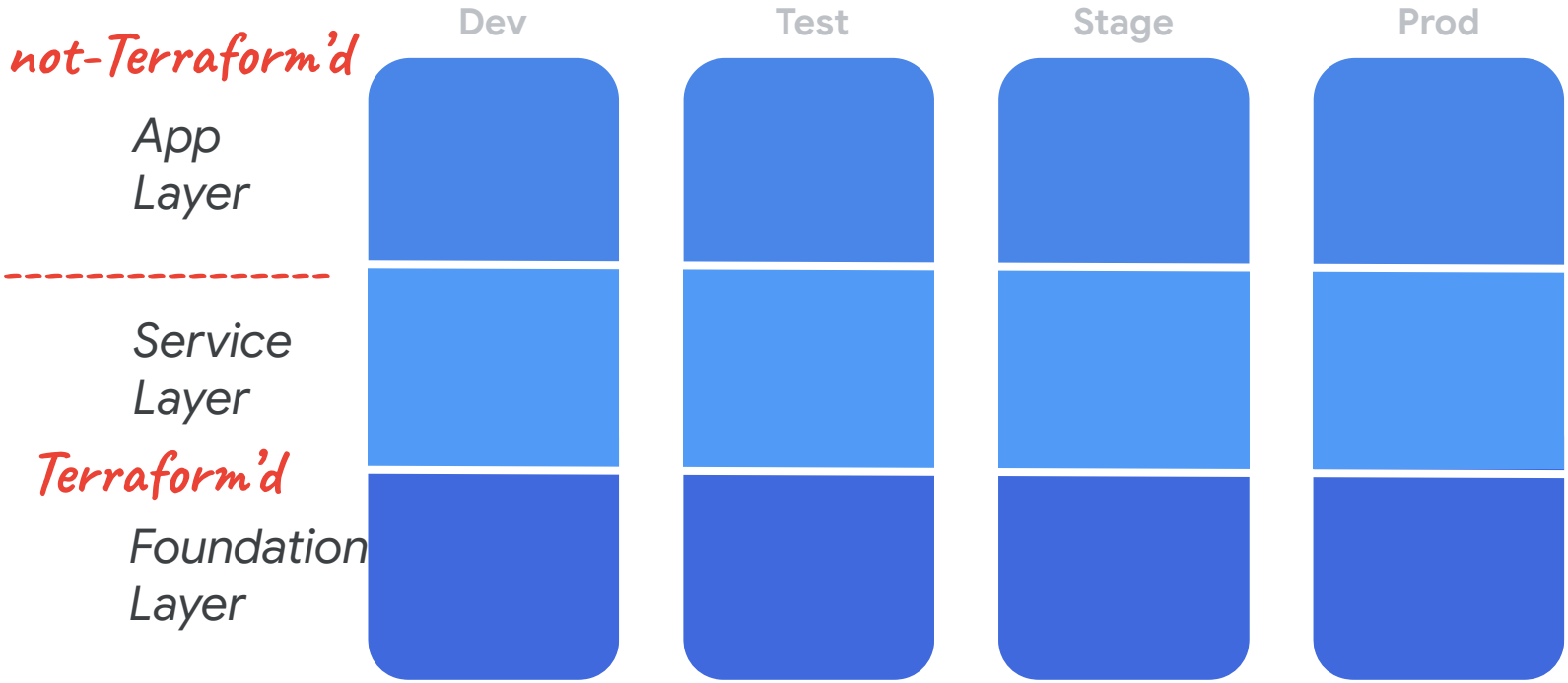
Separation is subjective



Layers vertically segment



Layers vertically segment



Infrastructure Layers segmentations

Foundation

GCP Organization
GCP Folders
GCP Projects
AWS Organizations
Azure Organizations
Security
Networking
IAM
Logging
Firewalls

Service

GCP Services
GKE - GCE - CloudSQL
AWS Services
EKS - EC2 - RDS
Azure Services
AKS - AzureVM - AzureSQL
Argo/Artifactory
Logging
App Firewalls

App

Application binaries
Containers
Helm Charts
Database data

Segmentation of Responsibility

Security

Security
Logging

Networking
Monitoring
Observability
Auditing

Ops

Security
Logging

Networking
Monitoring
Observability
Versioned States
Scaling
Healing

Dev

Security
Logging

Application Logic
Database Data
Versioned State
Observability Ticks

Segmentation of Responsibility

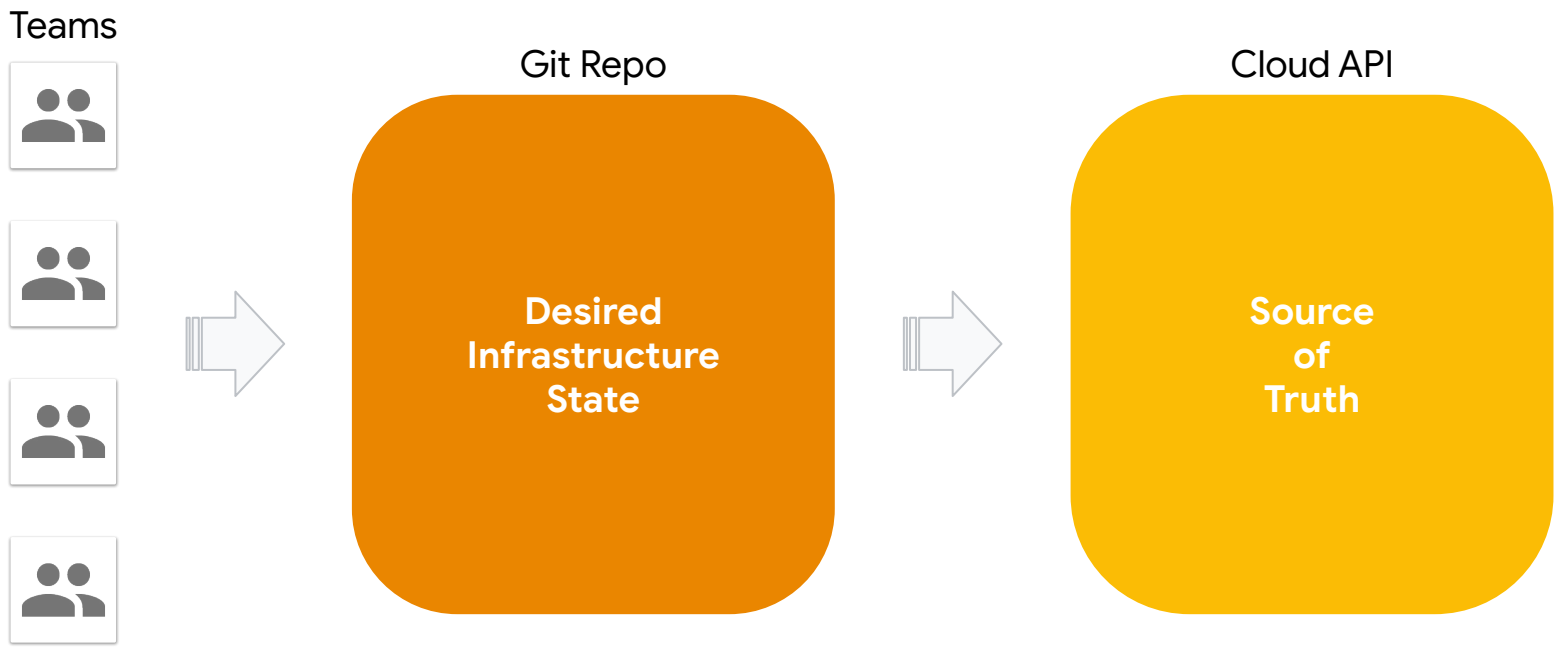
This segmentation allows each team to focus on their area of responsibility providing empathy for other teams

Security	Ops	Dev
Security	***Security***	***Security***
Logging	***Logging***	***Logging***
Networking	Networking	Application Logic
Monitoring	Monitoring	Database Data
Observability	Observability	Versioned State
Auditing	Versioned States	Observability Ticks
	Scaling	
	Healing	

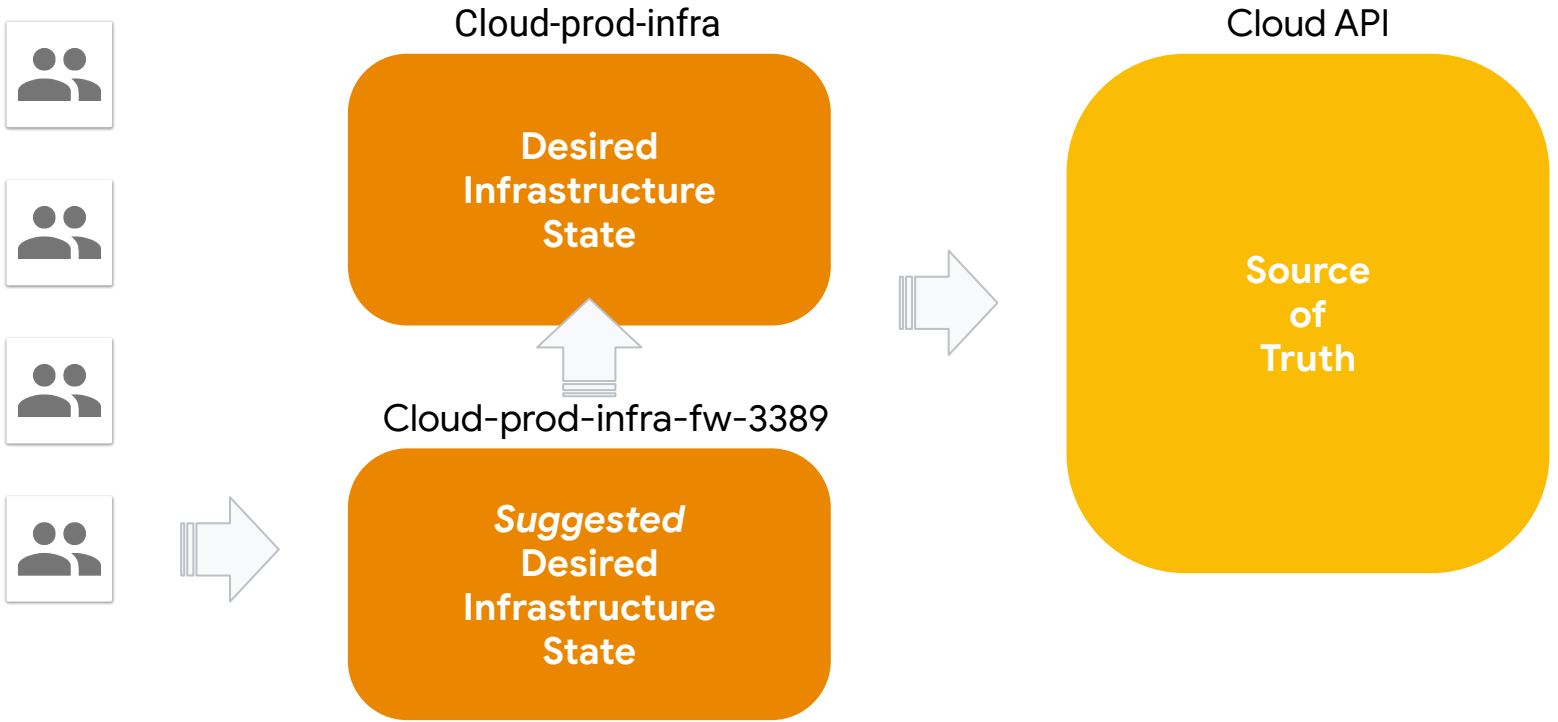
GitOps

Git interface for the
Desired Infrastructure State

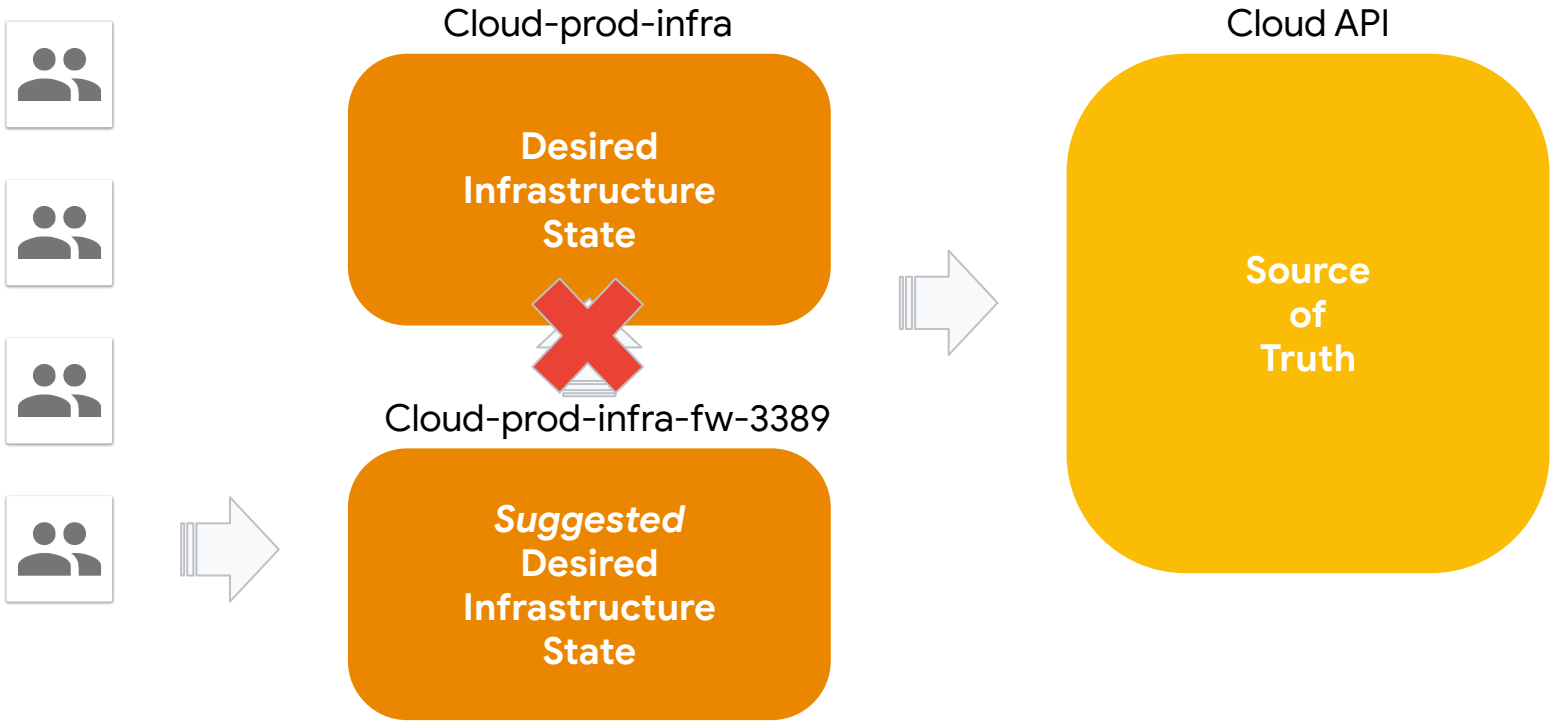
GitOps


Teams
Git Repo
Desired Infrastructure State
Cloud API
Source of Truth

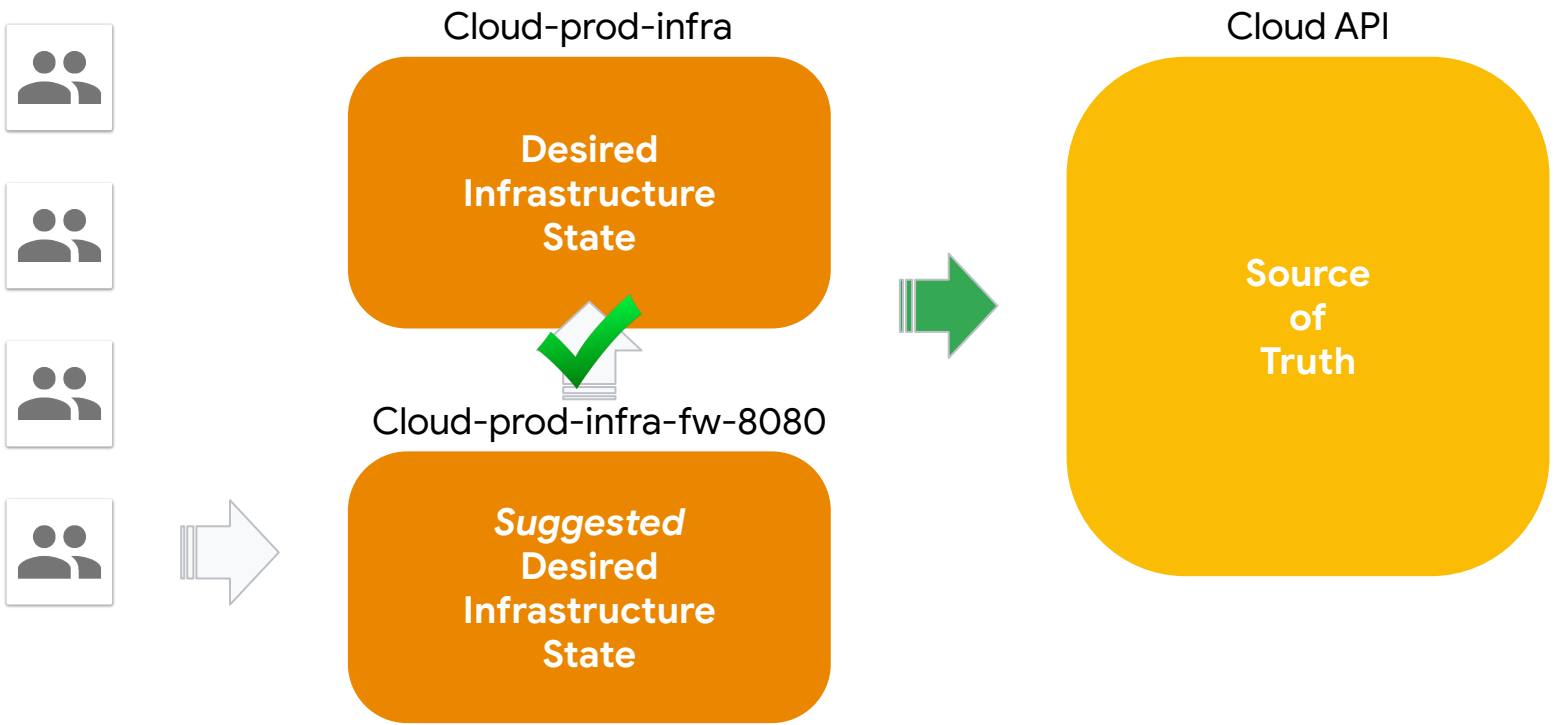
Git Pull Requests



Git Pull Requests



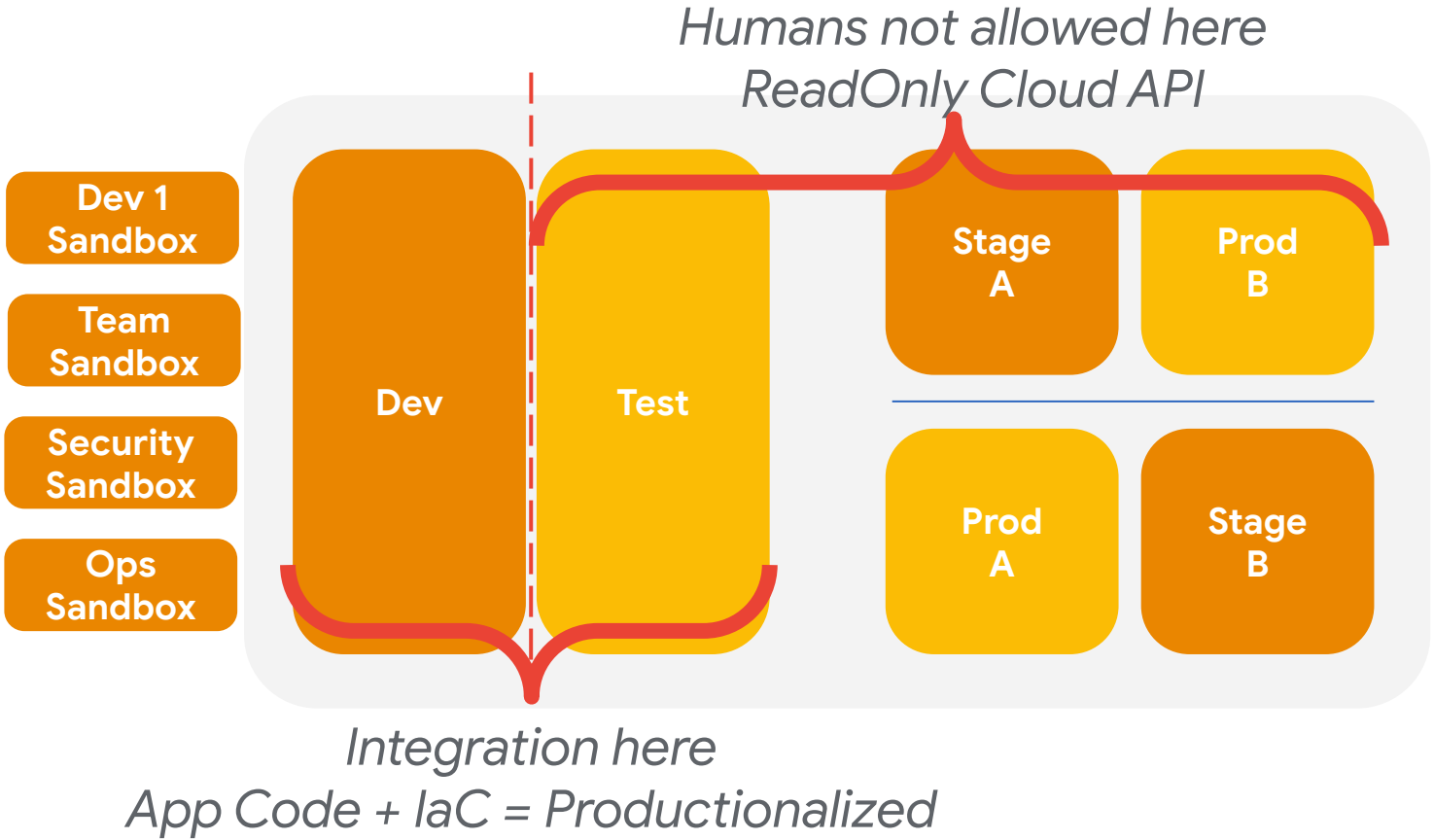
Git Pull Requests



Mad Scientist Laboratory

Mad Scientist Laboratory

Code Lifecycle Across Environments

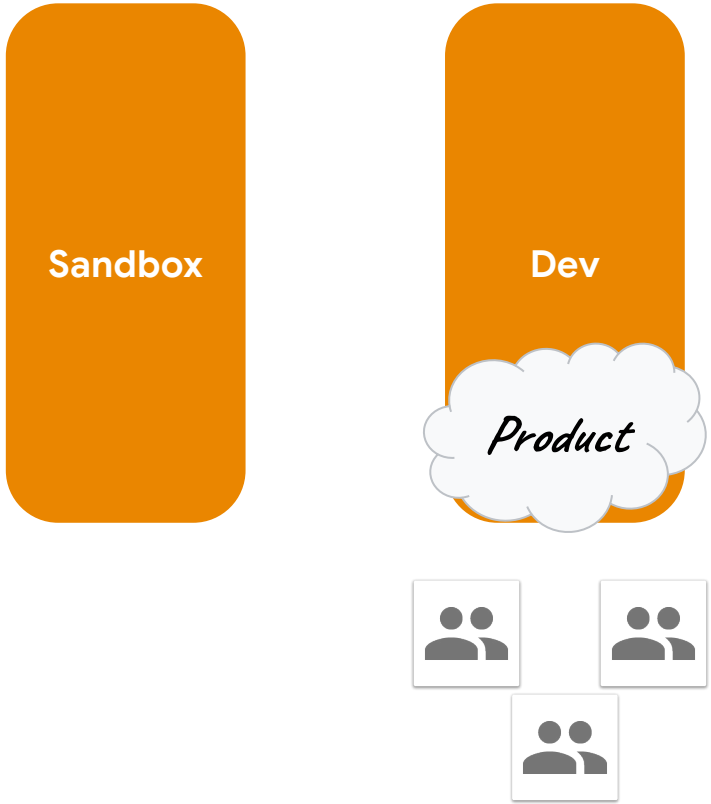


The Sandbox

- Confined anarchy
- Empower engineers to experiment & learn
- NO - infrastructure-as-code
- YES - Web Consoles!!
- Only rule = *security perimeter preventing opening to the public web*

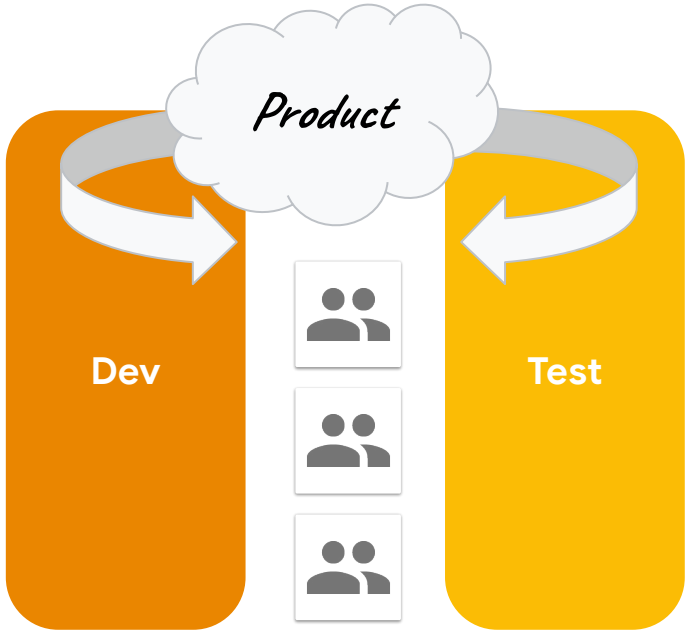
Promote to Dev for Integration Collaboration

- Dev, Ops, Security collaborating Git PR
- Goal is productionalization of product
- App Code + IaC
- Vetting of:
  - Security
  - Scaling
  - Healing



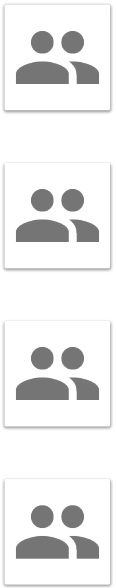
Promote to Test for Prod testing

- Product ready for Prod promotion
- NO - humans in Test
- ONLY - IaC in Test



GitOps

Teams



Git Repo - Terraform
Firewalls

Git Repo - Terraform
Networking

Git Repo - Terraform
Service Layer

Git Repo - Terraform
Foundation Layer

Cloud API

Source
of
Truth

Service Buffet

Security
Sanctity

> “The security of the infrastructure is designed in progressive layers...”

Google Infrastructure Security Design Overview

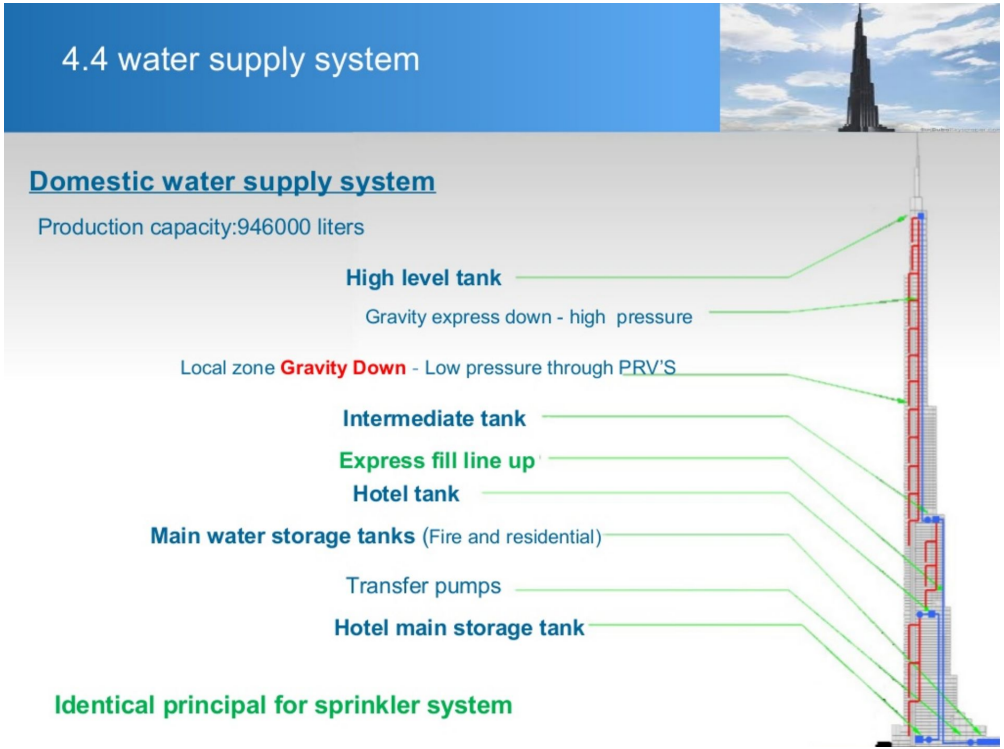
Simplicity

+

Known State

=

Security



Simplicity

+

Known State

=

Security

Virus Scanning is making payments for the car that was repossessed 12 years ago

CVE-2021-24092: 12 Years in Hiding – A Privilege Escalation Vulnerability in Windows Defender

KASIF DEKEL / FEBRUARY 10, 2021

Executive Summary

In this post, we disclose a severe vulnerability in Windows Defender that allows attackers to escalate privileges from a non-administrator user. Windows Defender is deeply integrated into the Windows operating system and is installed by default on every Windows machine (more than 1 billion devices).

BTR.sys deployed to the Windows source code and signed by Microsoft

2009

2021 **CVE-2021-24092**

CVE Published


WELCOME TO
FORT KNOX

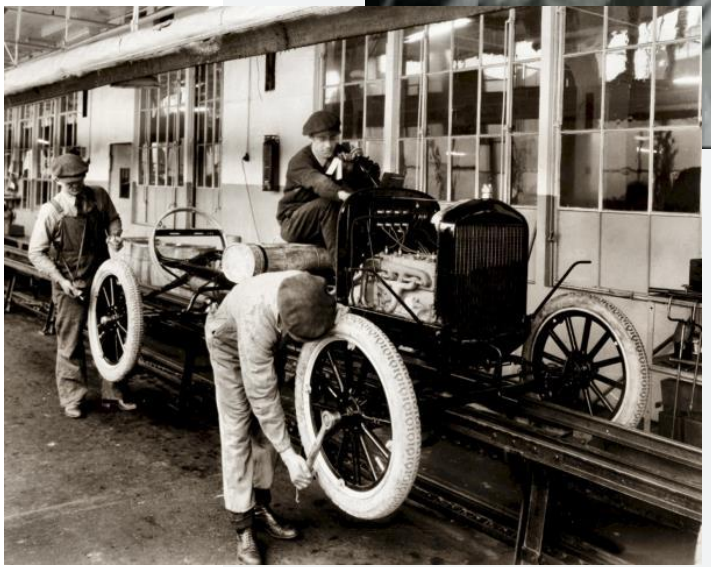
Infrastructure Layers Outcomes

Self-Healing

Self-healing infrastructure turns Saturday at 2am problems into Monday 10am problems.

Having the right people in the right frame of mind to solve hard problems.

Infrastructure Culture as a Recruiting Tool



Eliminate toil to:

- retain the talent you already have
- bring in new talent
- let engineers enjoy solving unique problems

Eliminating Toil

"Toil is the kind of work tied to running a production service that tends to be manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows."

From here