# Terraform Earth

## Secure Infrastructure for Developers

Chase Evans

# Timeline

1. Where we were before May
2. Where we are today
3. Where we are going

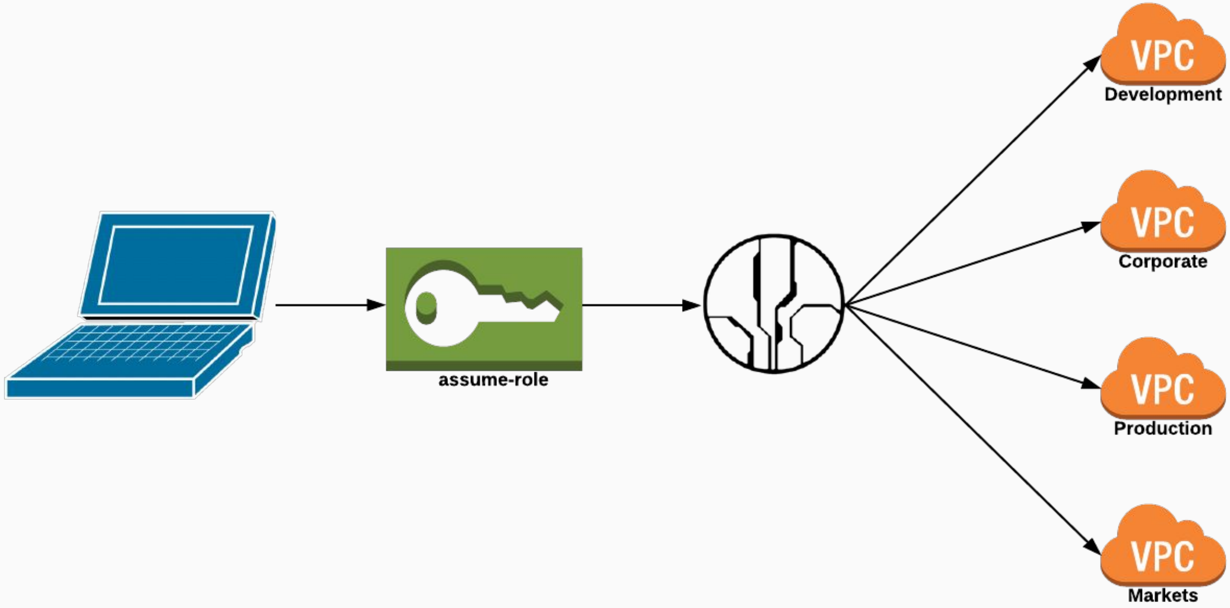
# Timeline

1. Where we were before May

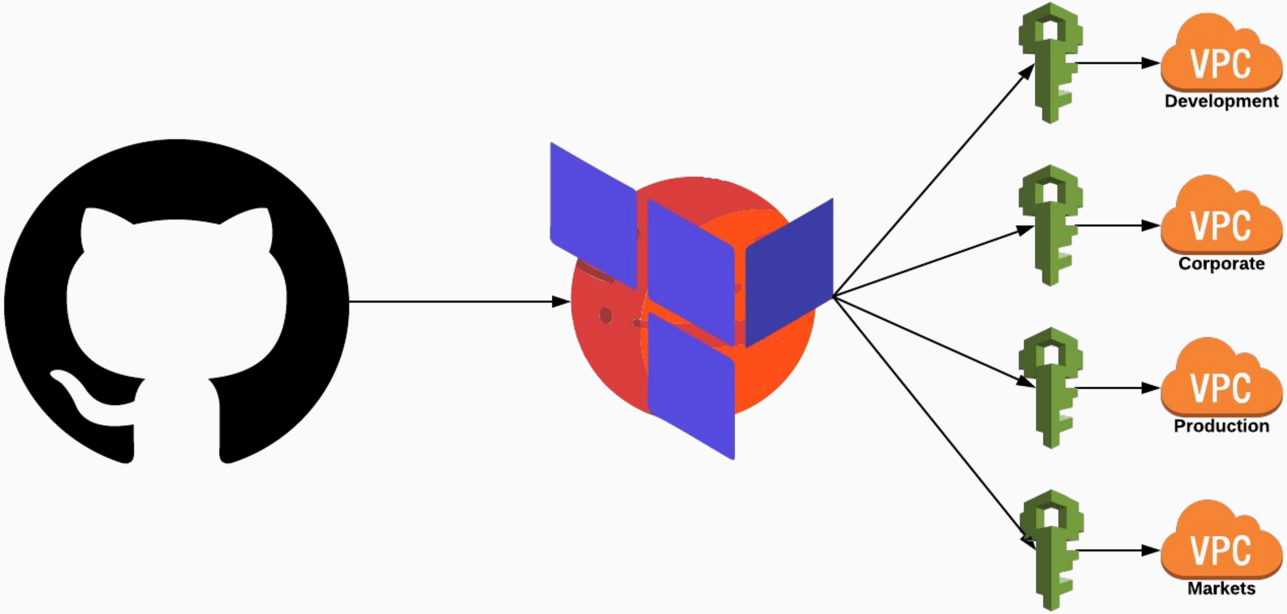
# GeoEngineer

- Builds Terraform state files by fetching remote resources, think `*$ terraform refresh*`
- Manual and distributed changes easily reconciled when AWS is the source of truth
- Looks like HCL
- github.com/coinbase/geoengineer

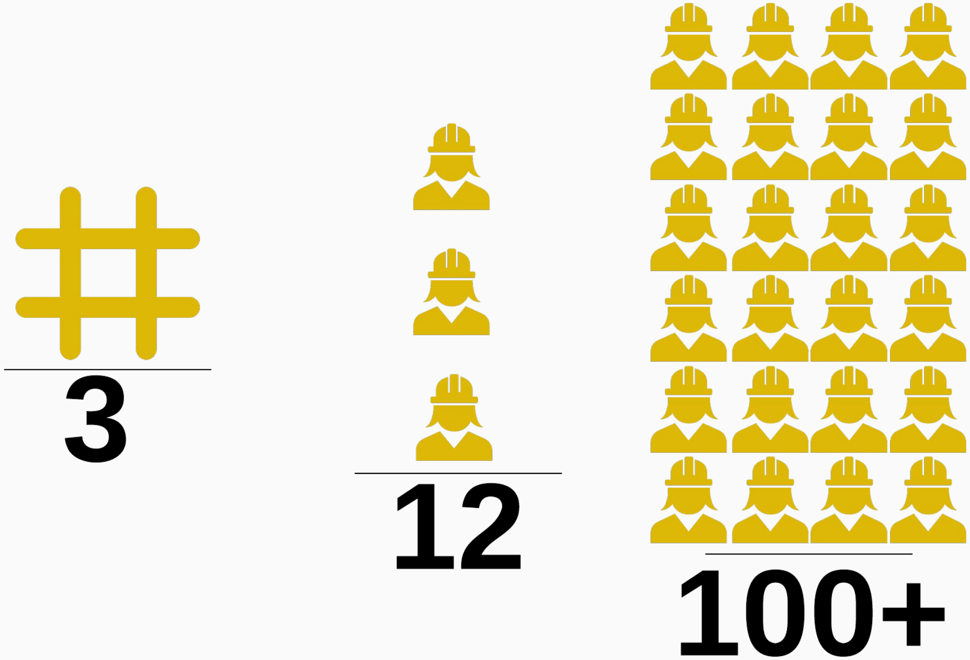
# Applying Resources



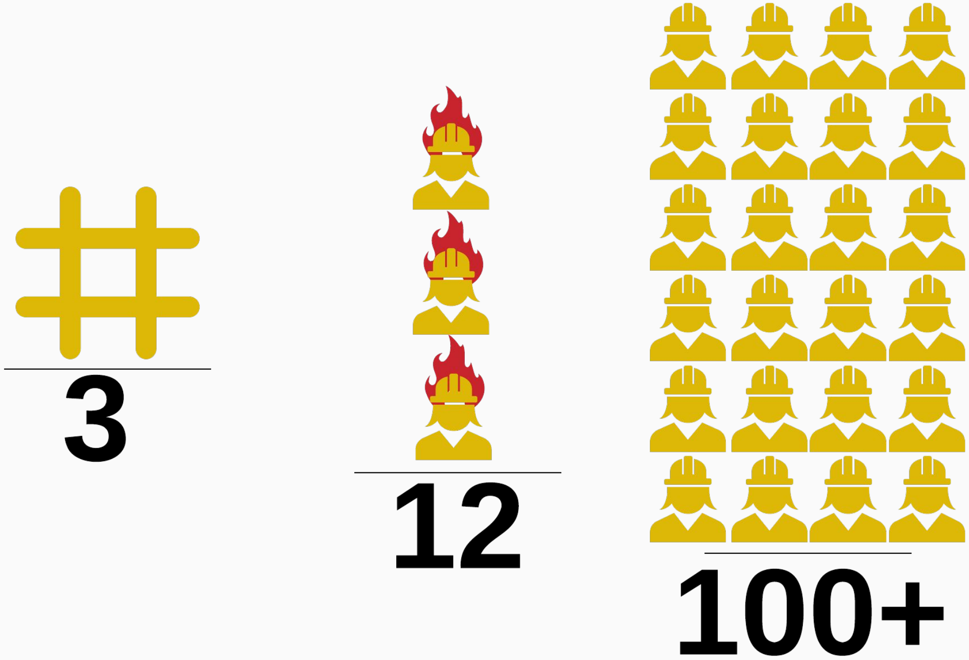
# Terraform Mars



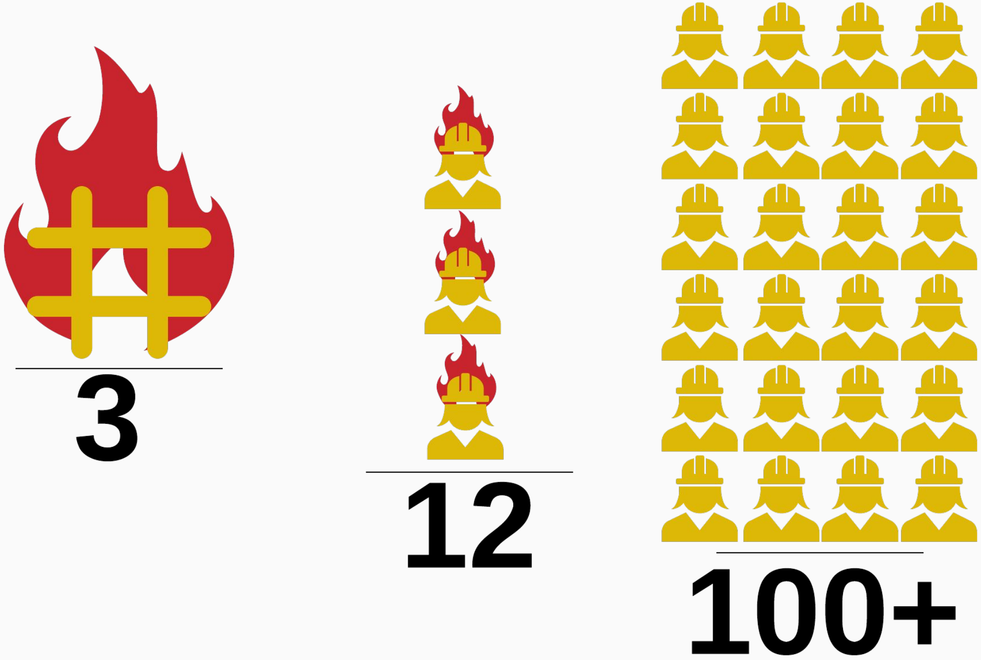
# The Problem



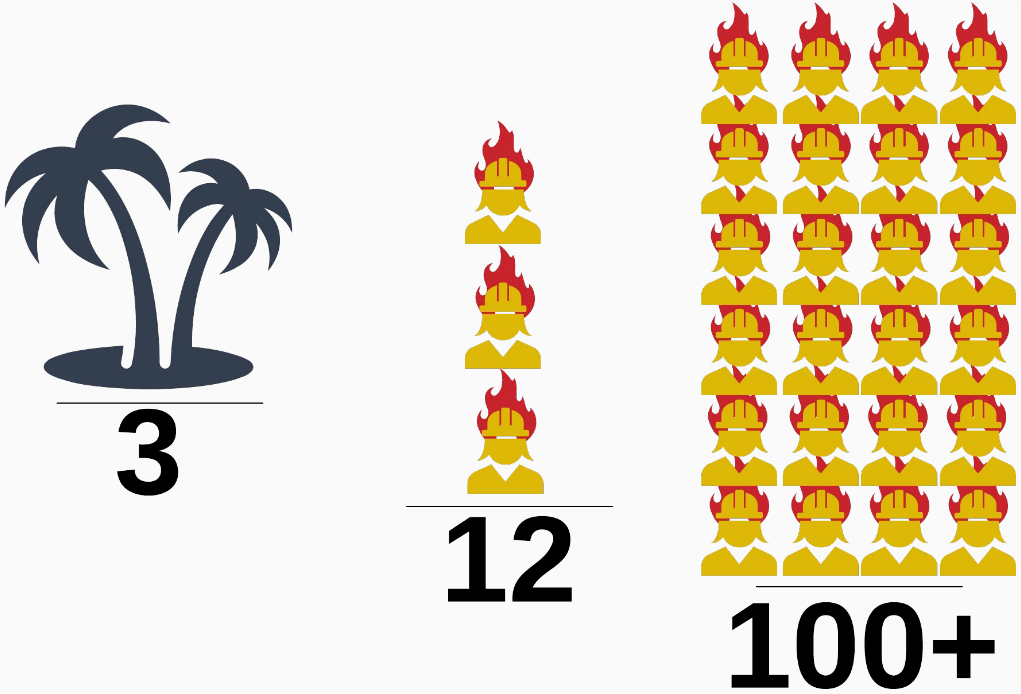
# The Problem (Bottlenecking)



# The Problem (Bottlenecking)



# The Problem (Bottlenecking)

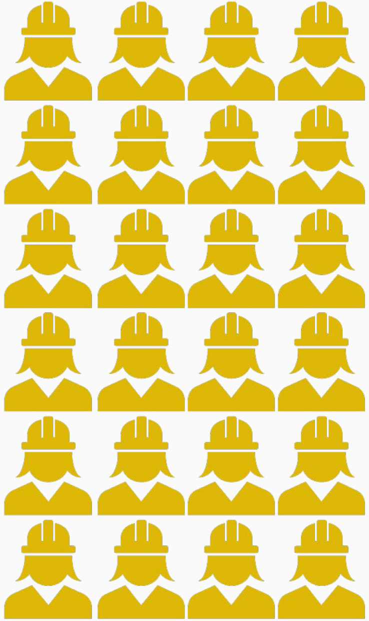


3

12

100+

# The Problem (Business units)



consumer

commerce

prime

pro

custody

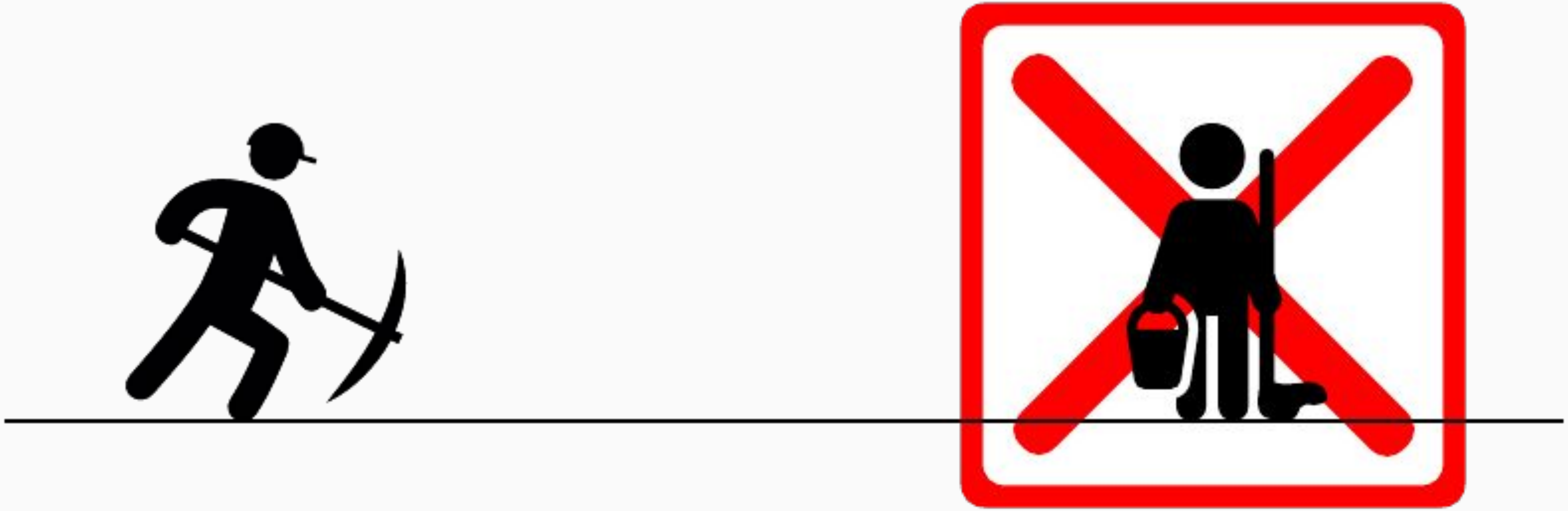
paradex

wallet

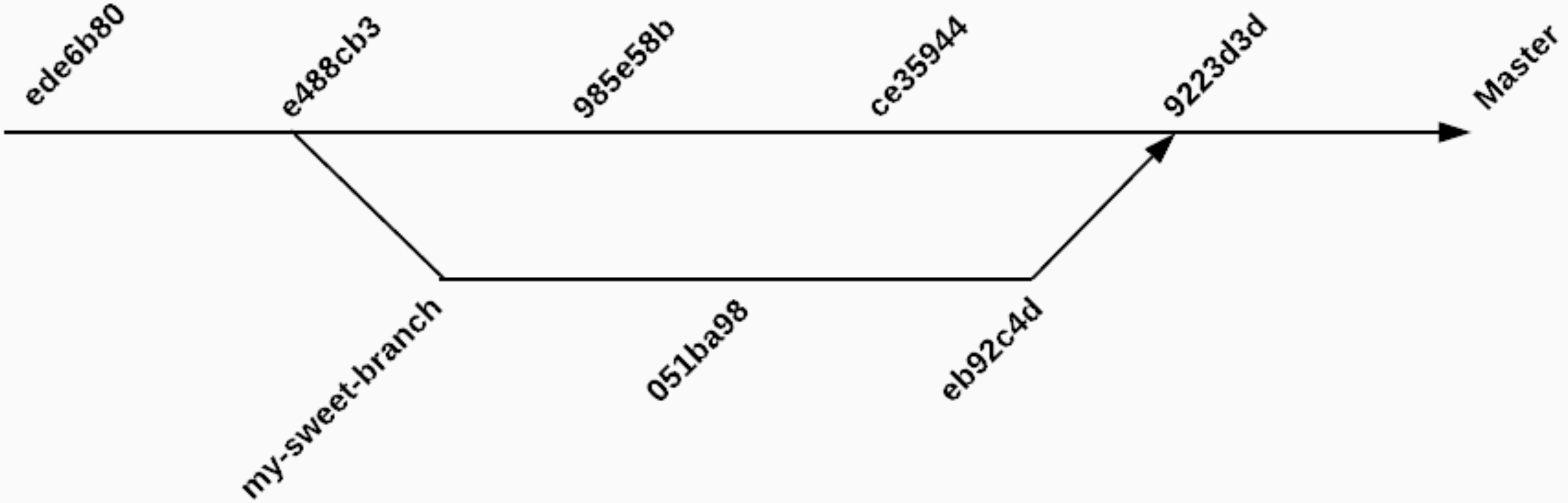
earn

asset management

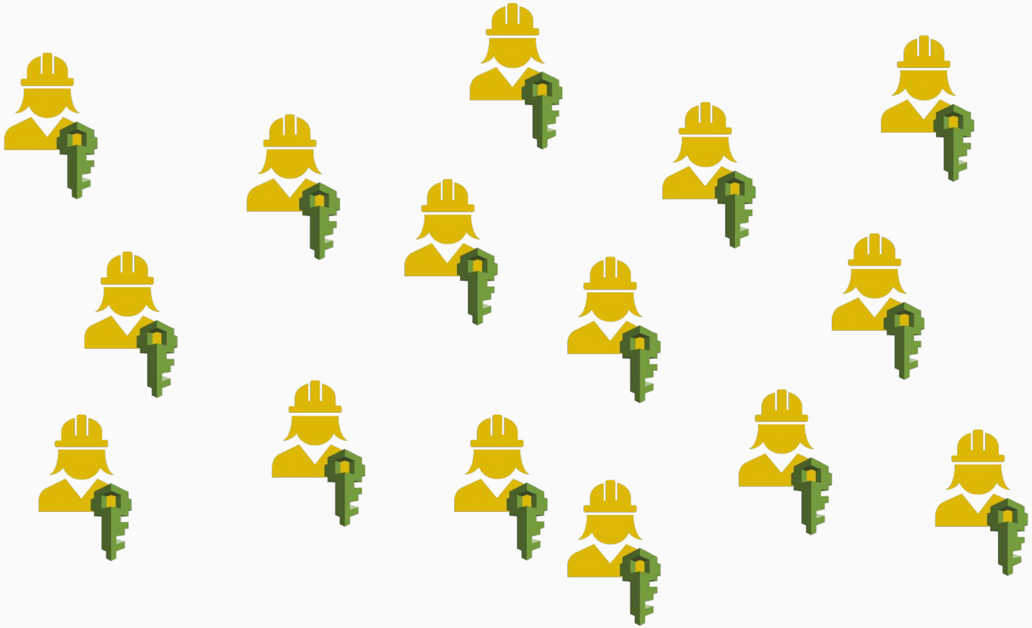
# The Problem (Platform vs Operations)



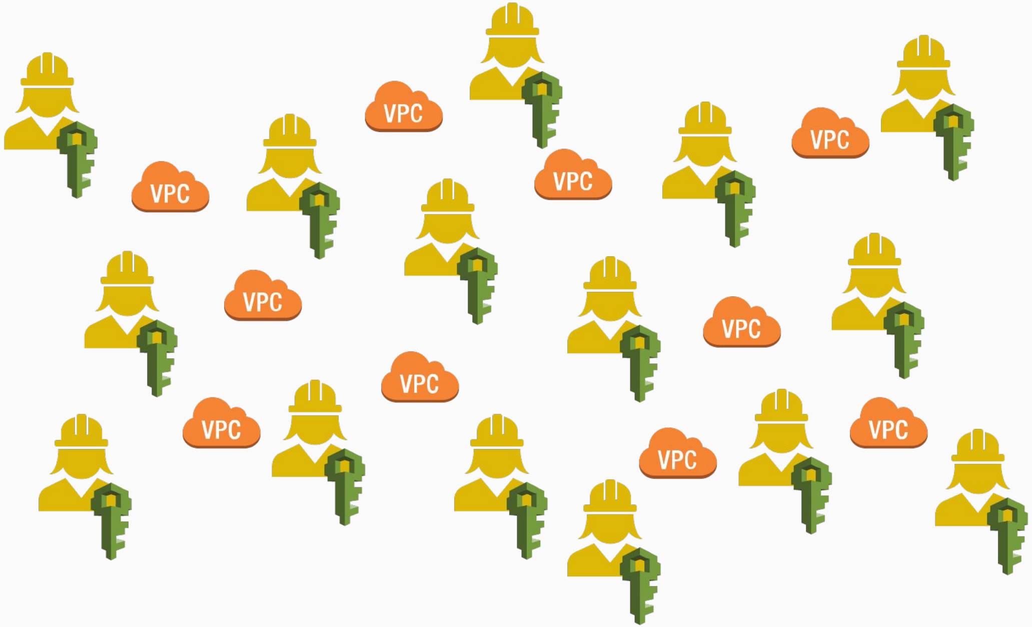
# The Problem (Did you remember to pull?)



# The Problem (Credential proliferation)



# The Problem (VPC proliferation)

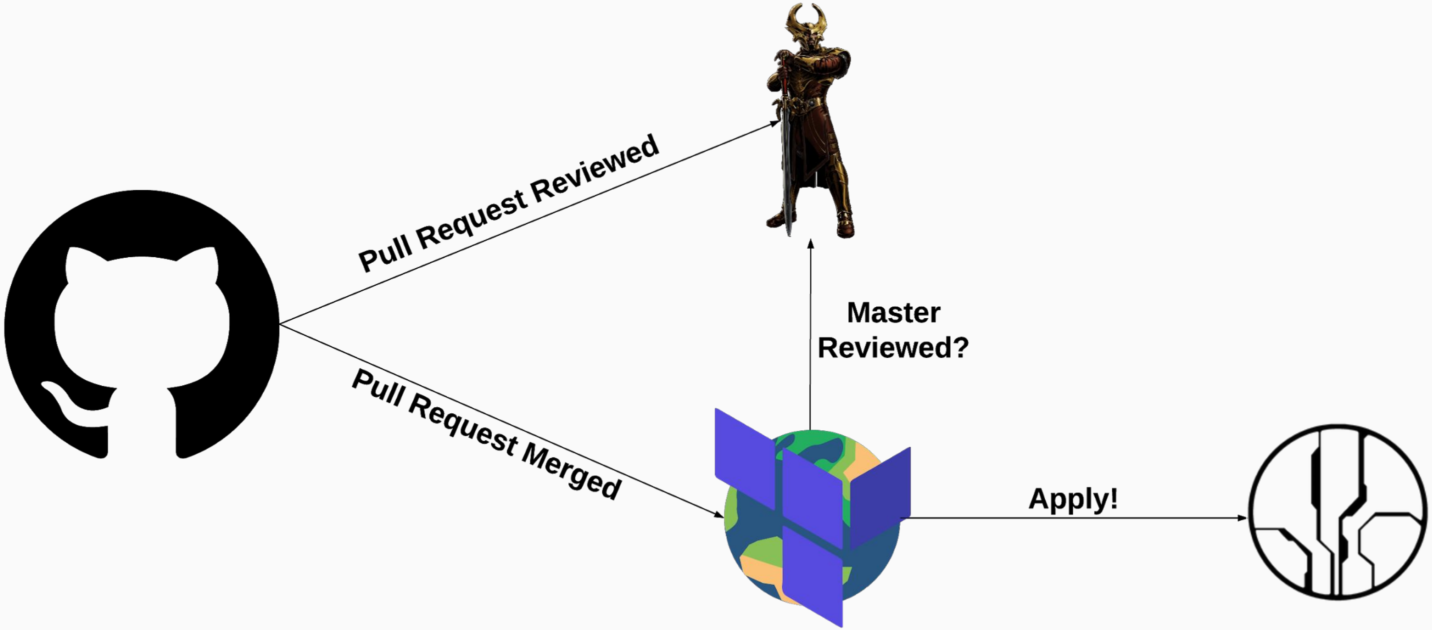


# Timeline

1. ~~Where we were before May~~

2. Where we are today

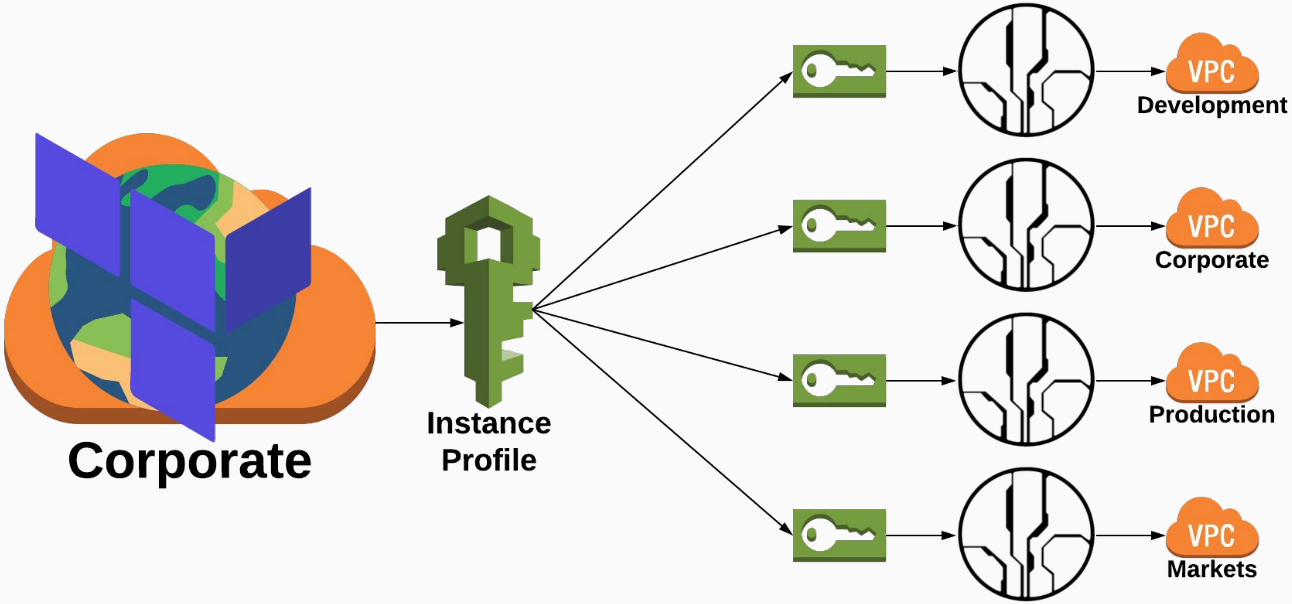
# Introducing Terraform Earth



# Heimdall

- Records PR approvals with MFA
- Provides a clean API
- Not vulnerable to administrative Github tampering

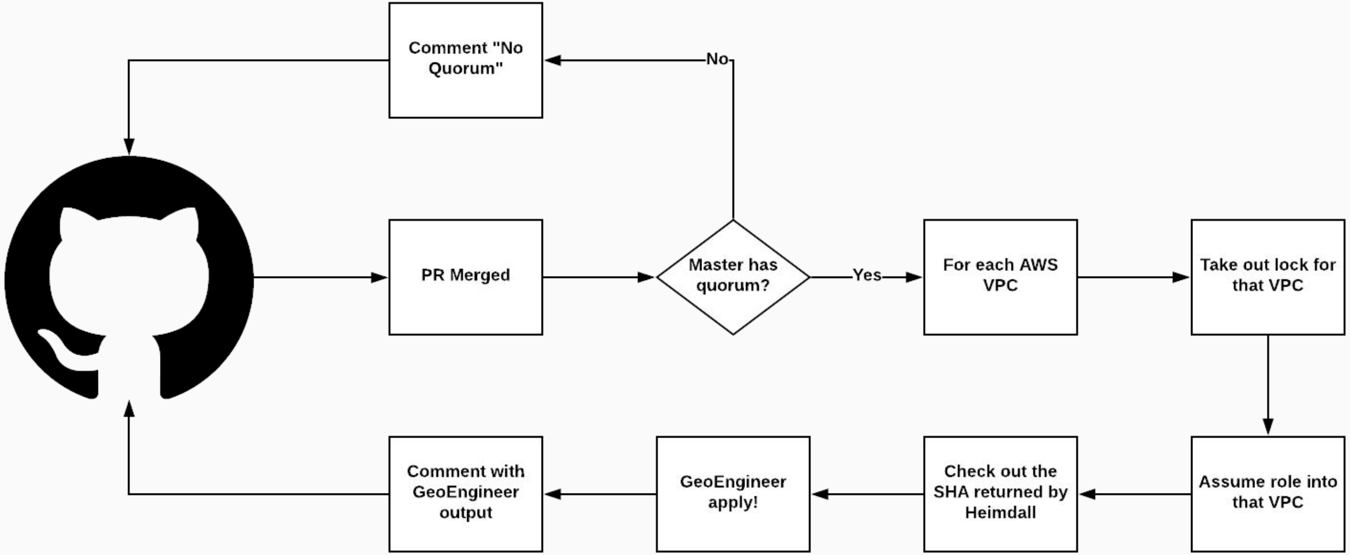
# Terraform Earth



# Single Production Deployment

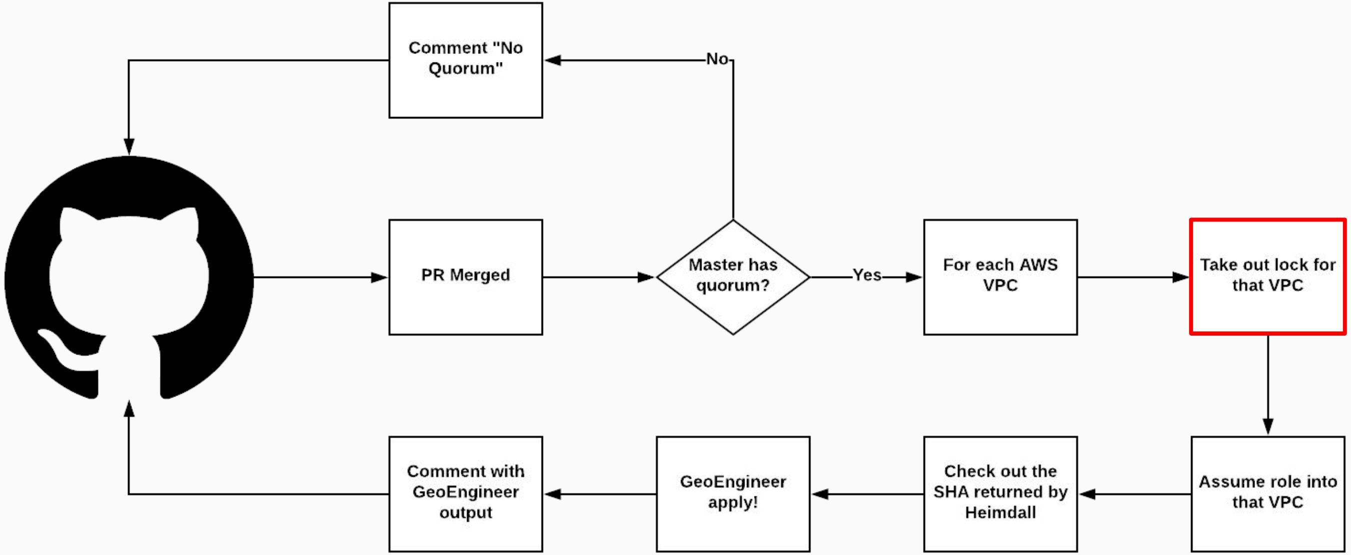
- One deployment makes updates easier
- New VPCs work without deployment

# Flow Diagram



Comment "No Quorum"
No
PR Merged
Master has quorum?
Yes
For each AWS VPC
Take out lock for that VPC
Assume role into that VPC
Check out the SHA returned by Heimdall
GeoEngineer apply!
Comment with GeoEngineer output

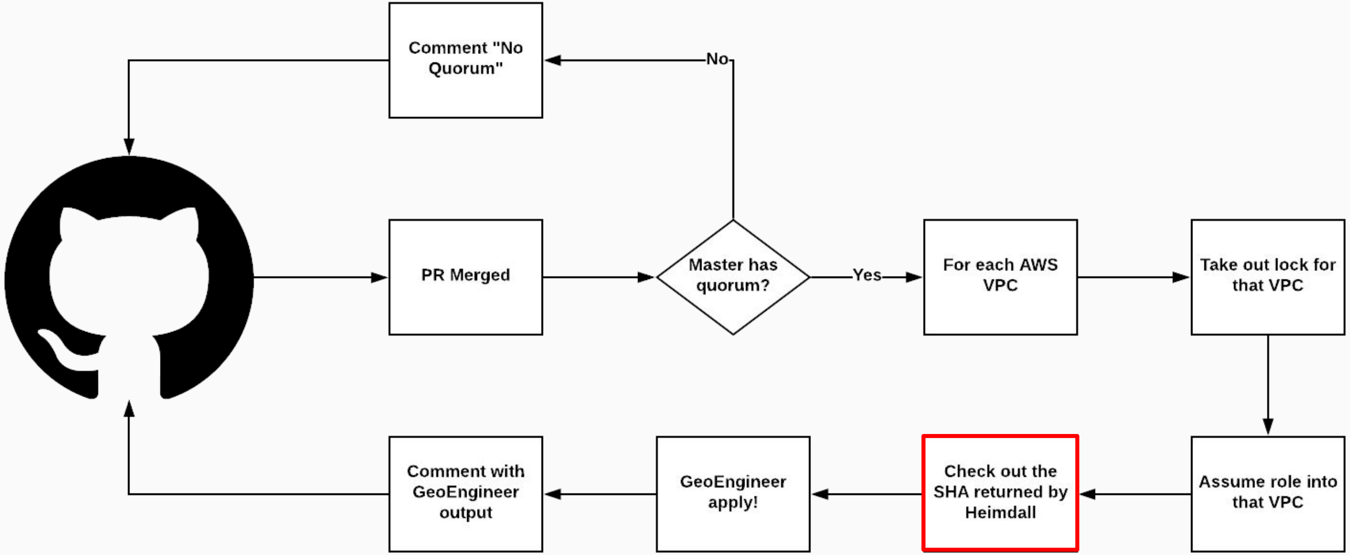
# Flow Diagram


Comment "No Quorum"
No
PR Merged
Master has quorum?
Yes
For each AWS VPC
Take out lock for that VPC
Assume role into that VPC
Check out the SHA returned by Heimdall
GeoEngineer apply!
Comment with GeoEngineer output

# Why bother locking?

- Concurrent changes are usually safe
- Sometimes multiple PRs pile up and need to modify a resource in order

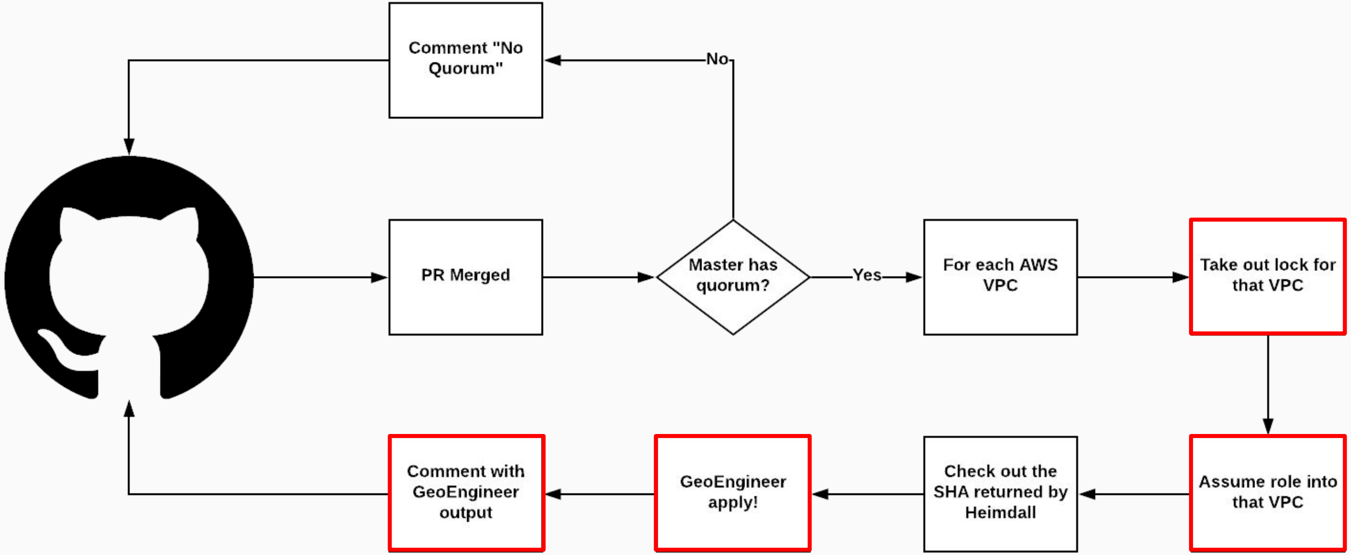
# Flow Diagram


Comment "No Quorum"
No
PR Merged
Master has quorum?
Yes
For each AWS VPC
Take out lock for that VPC
Assume role into that VPC
Check out the SHA returned by Heimdall
GeoEngineer apply!
Comment with GeoEngineer output

# Why SHAs and not ‘master’?

- Master is just a label and moves frequently
- Code has quorum, not labels
- Something could be merged to the repo between quorum check and clone

# Flow Diagram


Comment "No Quorum"
No
PR Merged
Master has quorum?
Yes
For each AWS VPC
Take out lock for that VPC
Assume role into that VPC
Check out the SHA returned by Heimdall
GeoEngineer apply!
Comment with GeoEngineer output

# Handling Failure

- Retry the GeoEngineer apply with backoff
  - AWS rate limits heavily
  - AWS has failures
- Queue and retry
- Replay the webhook using Github administration
- Add an endpoint to manually intervene

# Handling Failure

Not great solutions, if you have ideas, let me know

# Staging Deploys

- Setup a bot with limited privileges

  You can test the flow, without breaking everything

  We have a separate repository that defines 1 S3 bucket

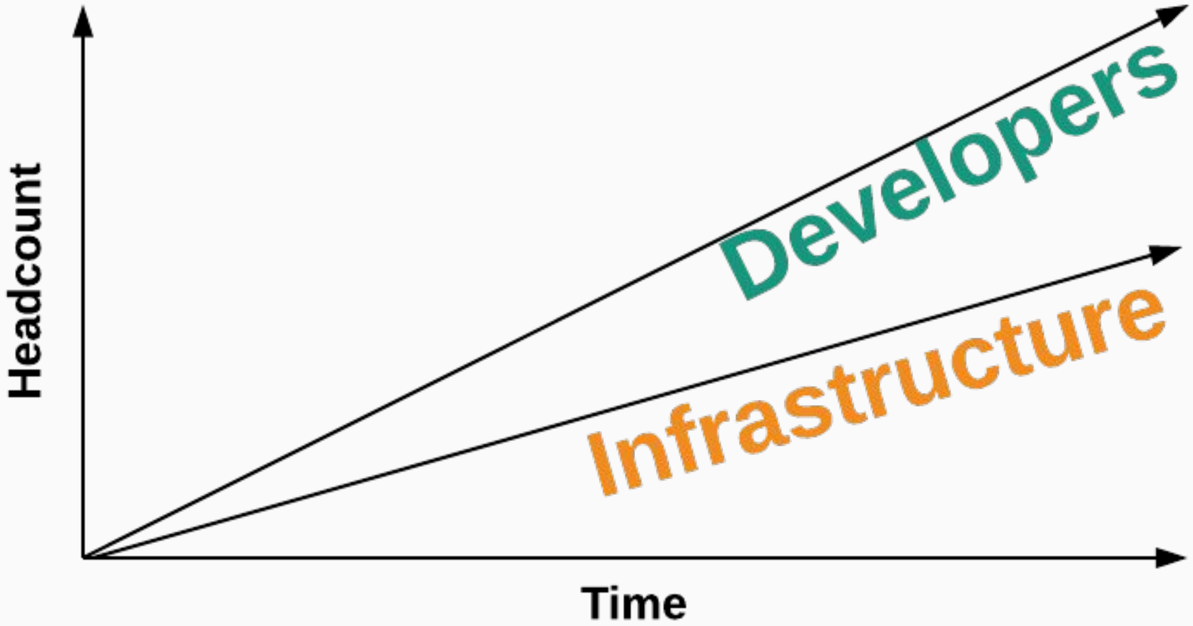
- Make a periodic cleaner that cleans up test resources

  We use lambdas to do this

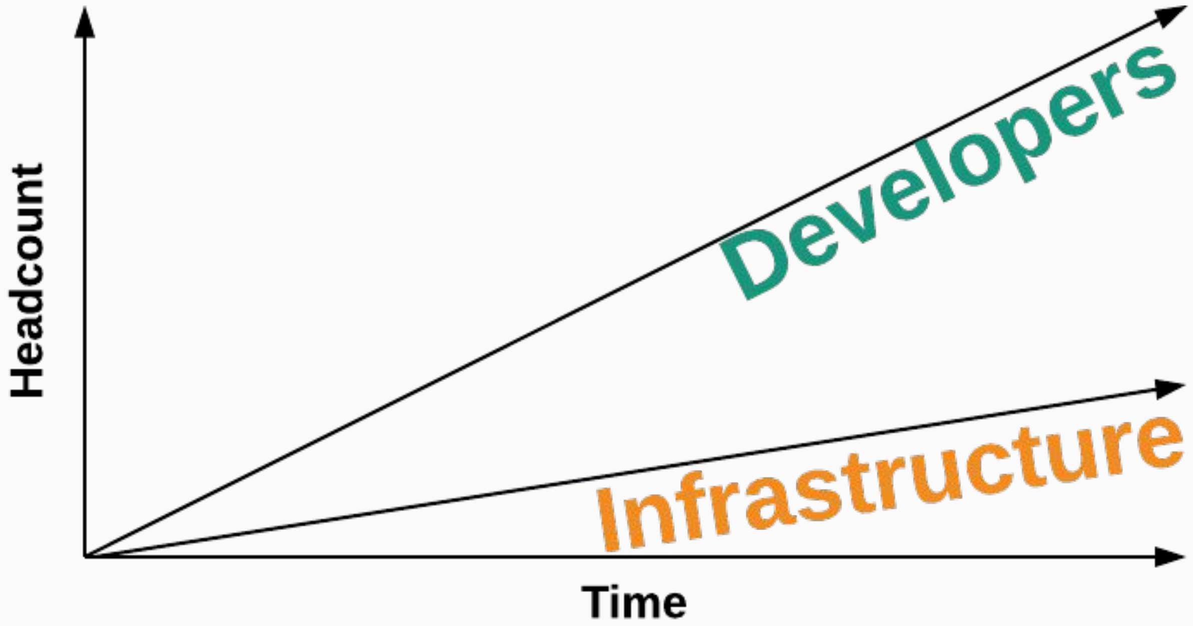
# Timeline

1. ~~Where we were before May~~
2. ~~Where we are today~~
3. Where we are going

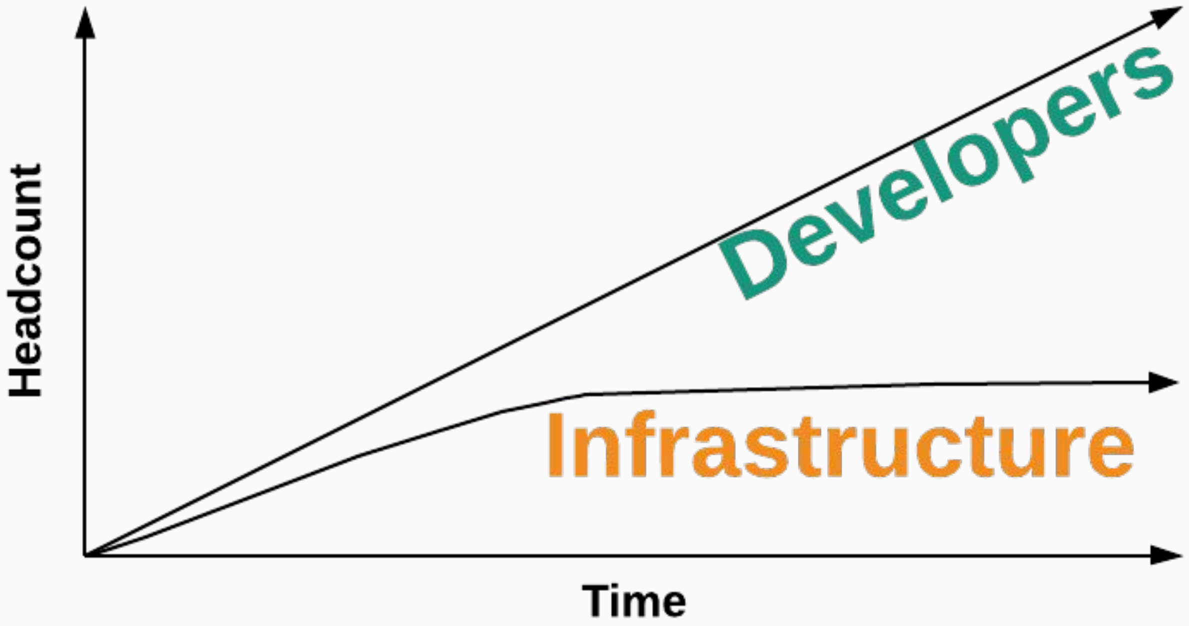
# Team Scaling



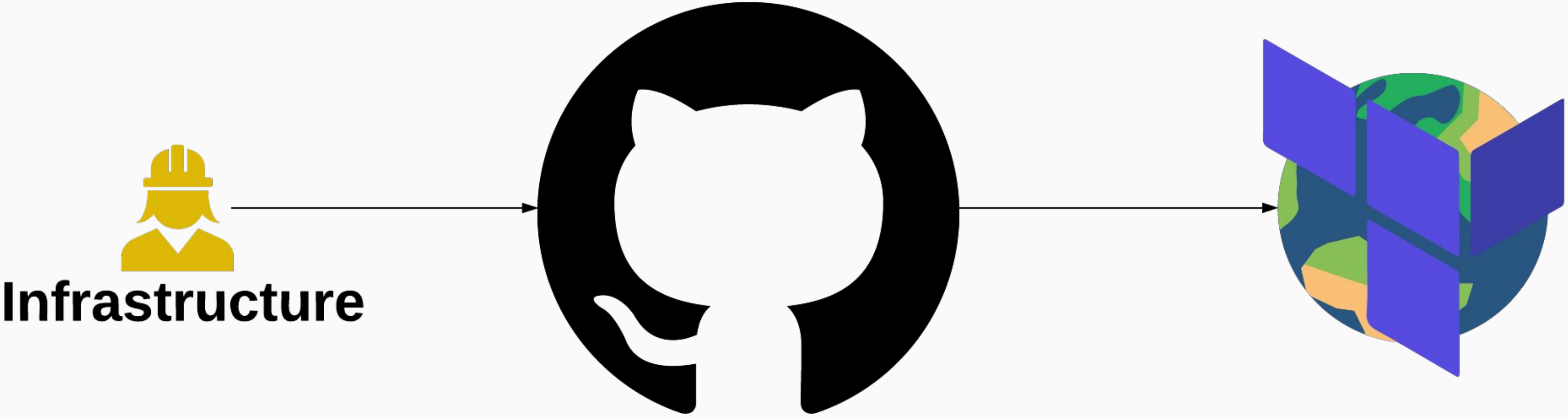
# Team Scaling



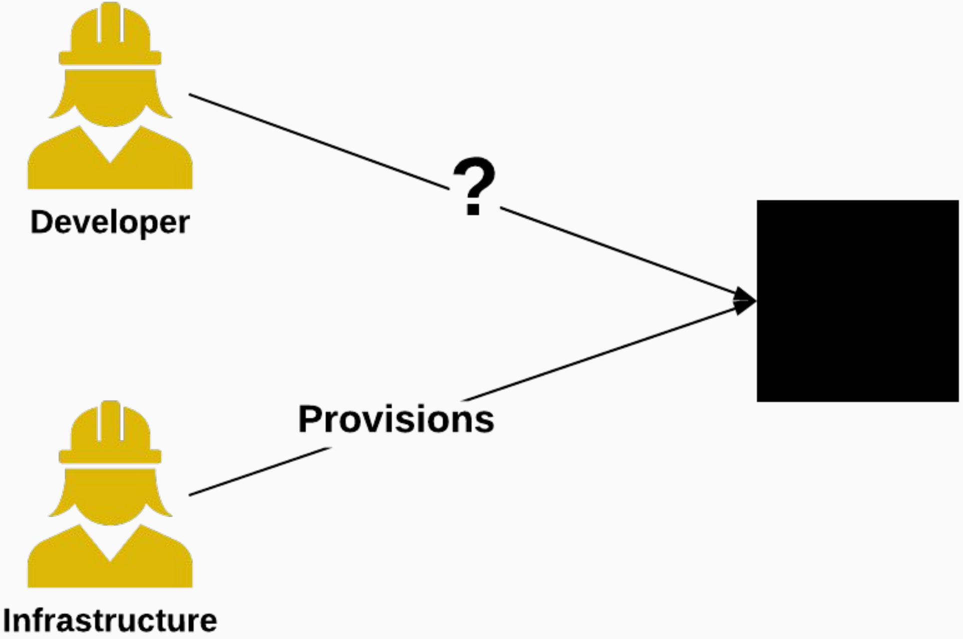
# Team Scaling



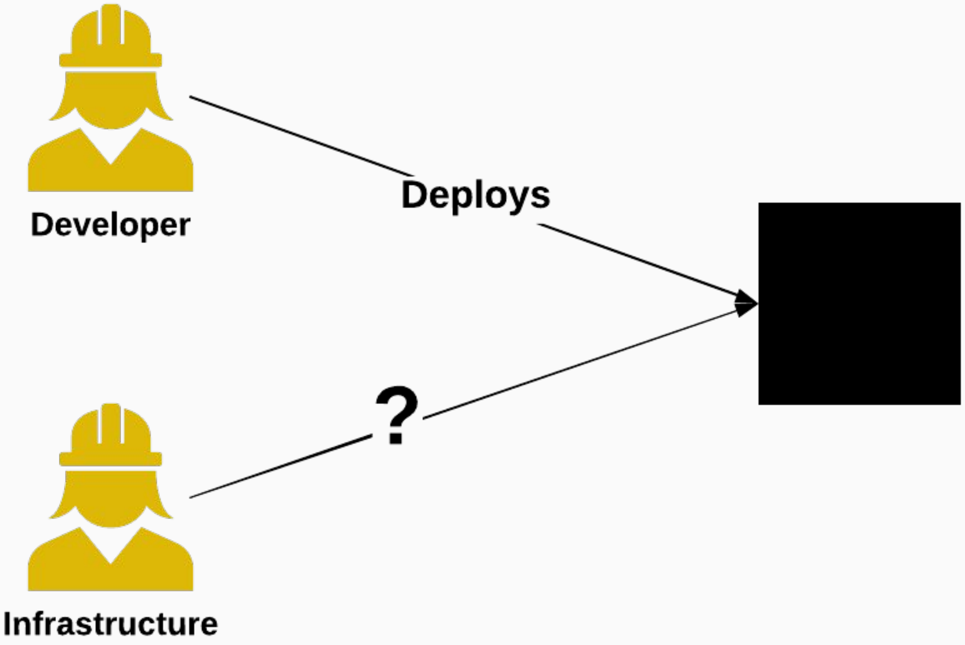
# Resource Configuration Today



# Ownership



# Ownership



# Resource Configuration Today

- project = Project.new(‘infra/heimdall’, aws_accounts)
- project.service_with_elb(‘api’, configuration)
- project.rds_instance(‘db’, configuration)

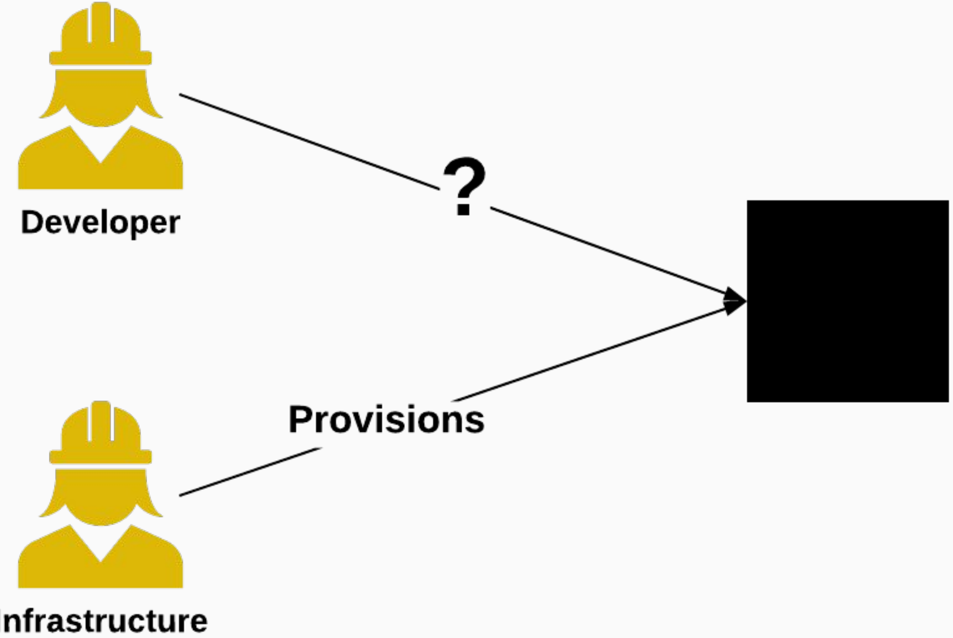
# What’s Wrong?

- Uses language the Infrastructure team knows
- Developer’s mental model of deploys is not represented
- Too many options, very little opinion
- Code is too flexible

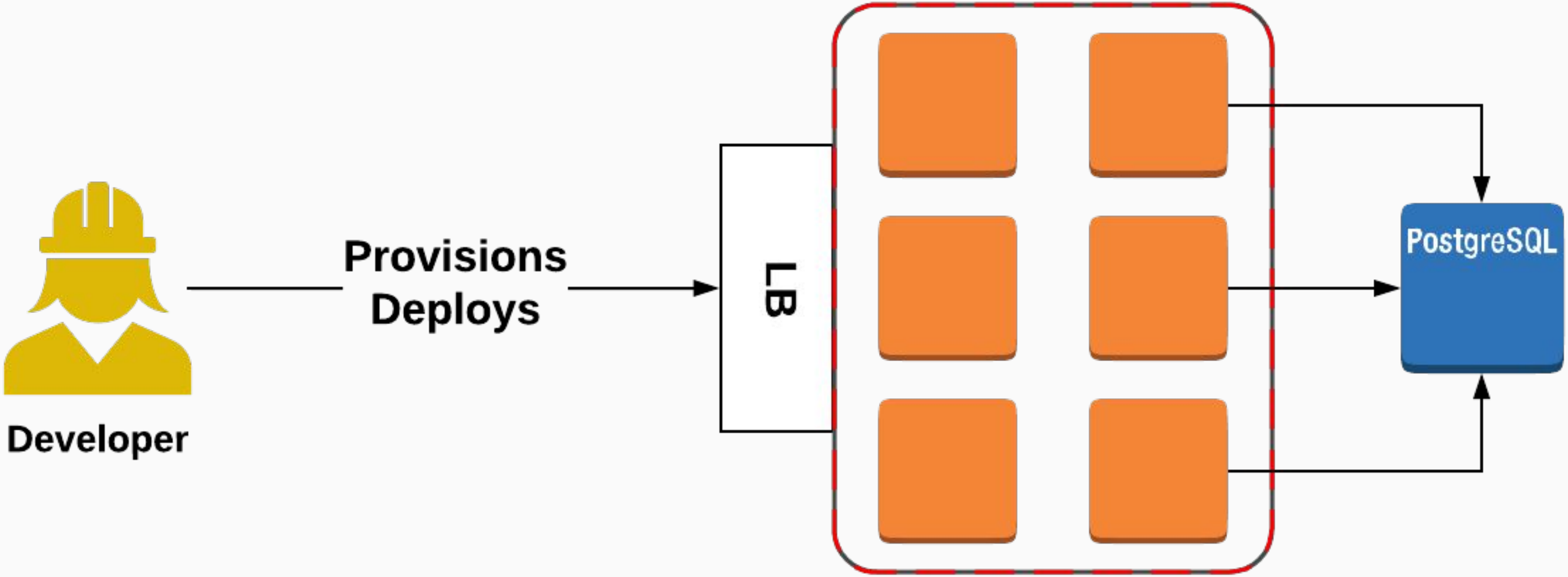
# Resource Configuration Tomorrow

```
name: ‘developers/my-service’
services:
  -   api:
      load_balanced: true
      accessible_by: [‘developers/my-other-service’]
databases:
  -   postgres:
      size: medium
```

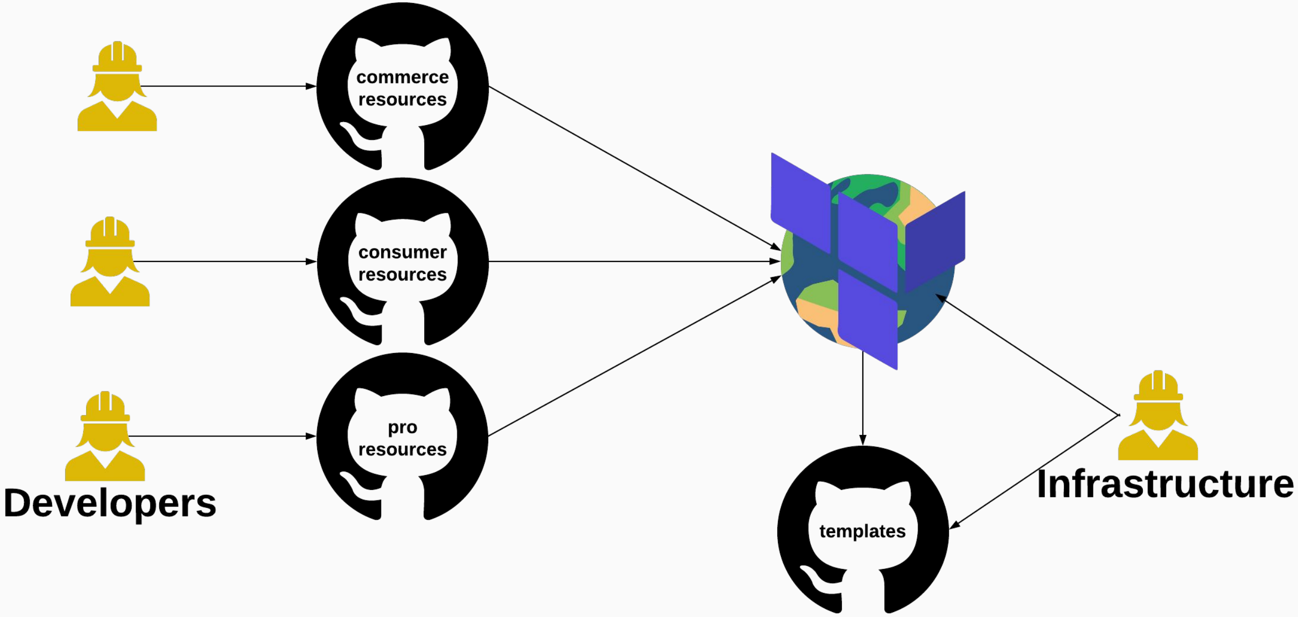
# Ownership



# Ownership



# The Future


commerce resources
consumer resources
pro resources
Developers
templates
Infrastructure

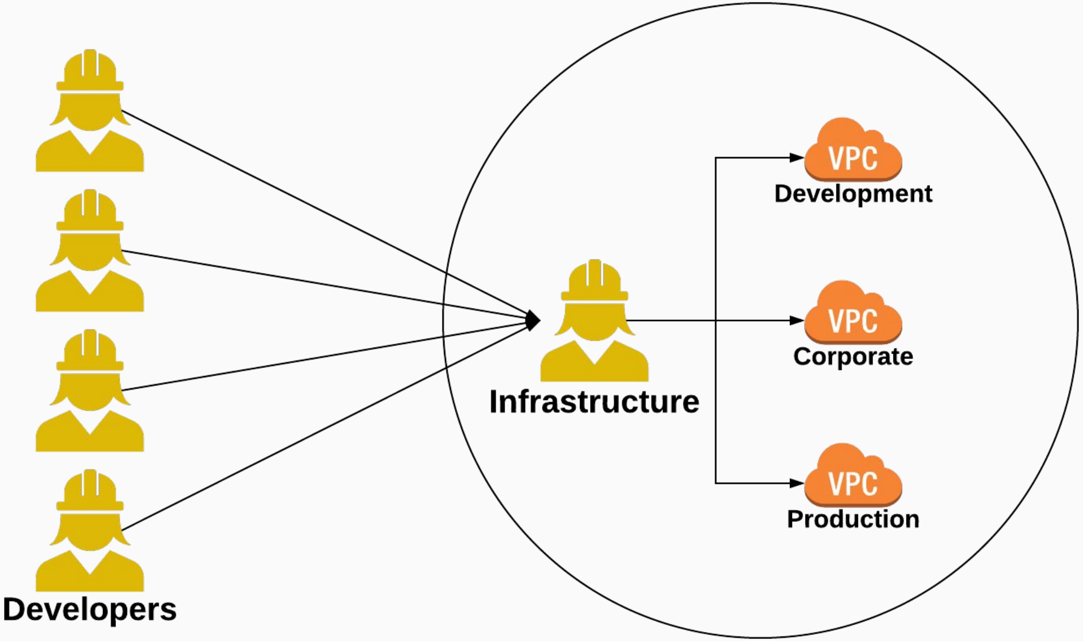
# Design Considerations

- Mono-repo or multi-repo
- Automated workflows (PR bots)
- Exposing the information to outside services

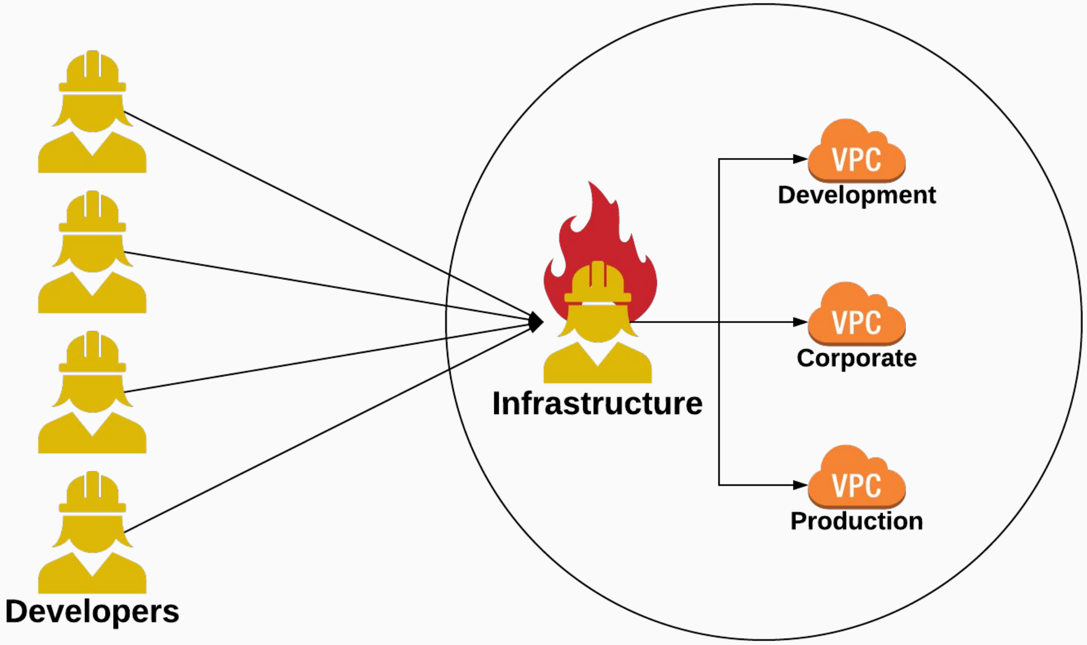
# The Other Half

- Provisioning and management is now easy
- Operation is not

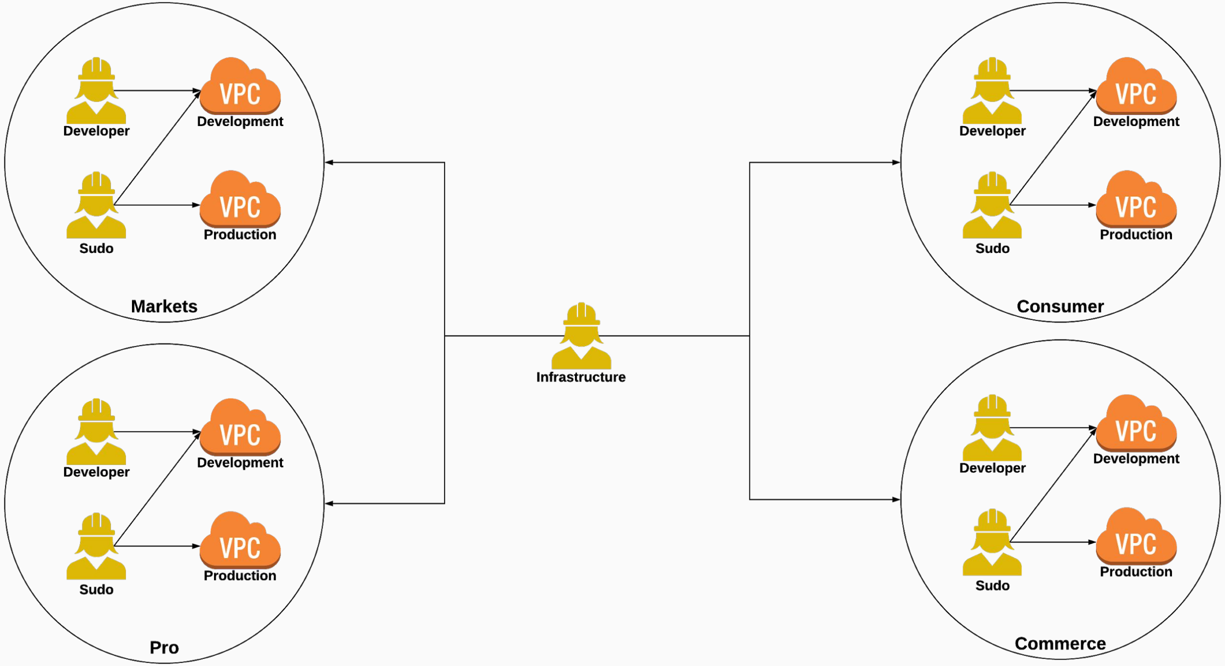
# Account Stewardship Today



# Account Stewardship Today



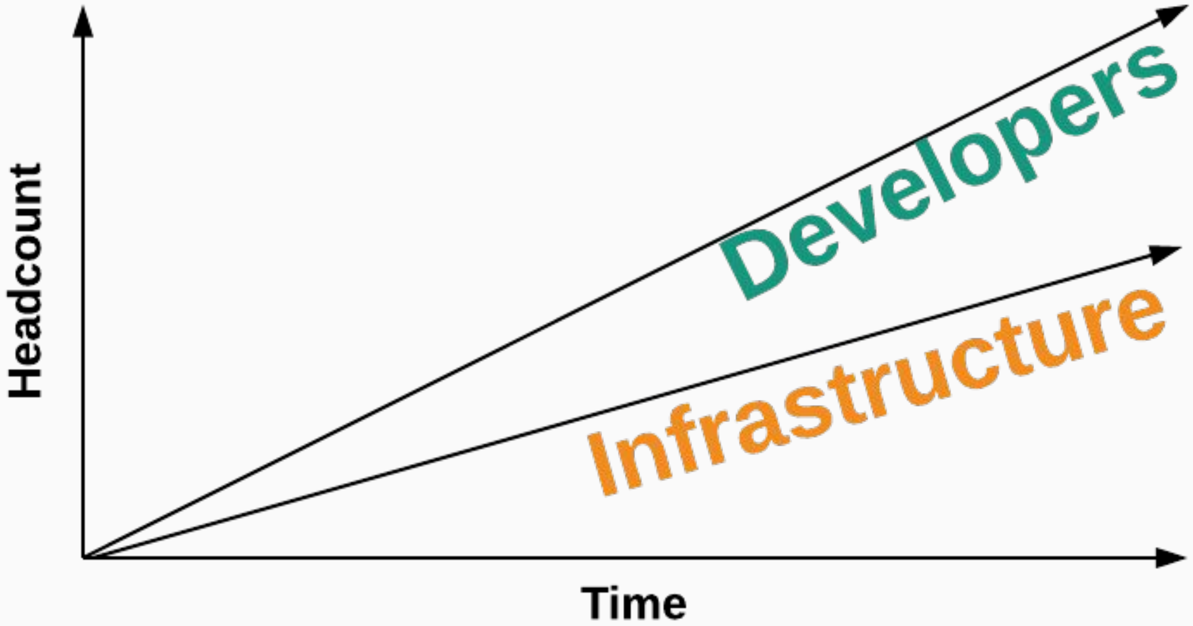
# Account Stewardship Tomorrow



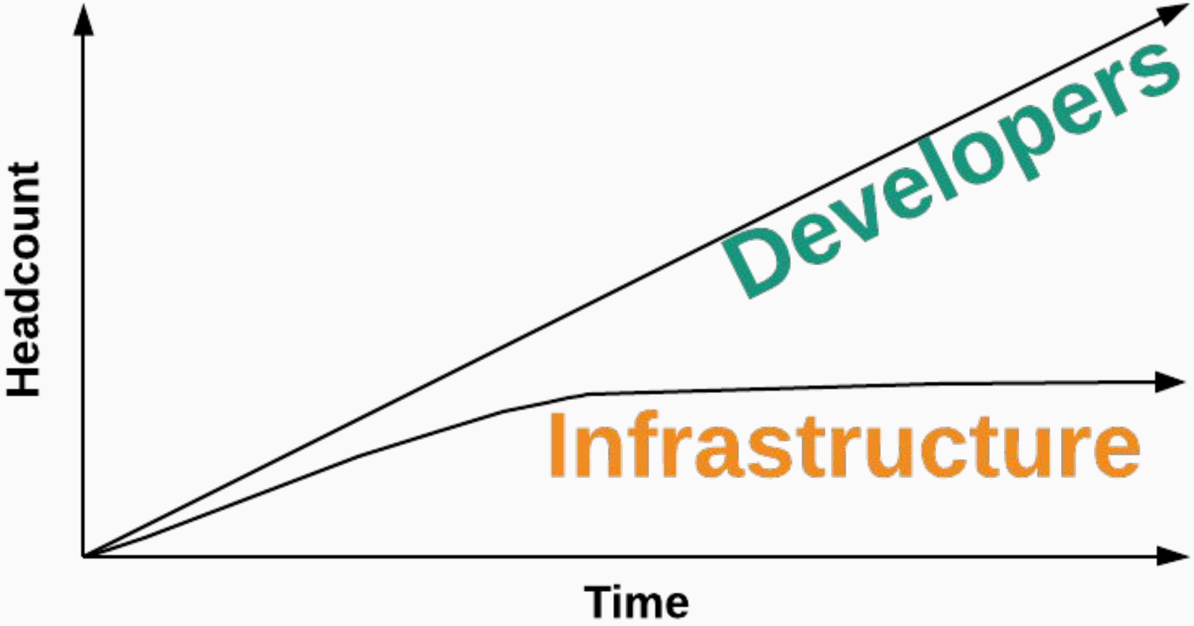
# Complications

- Managing connectivity between many VPCs is hard
- Like microservices, finding the right domain is difficult
- How much access is enough access?

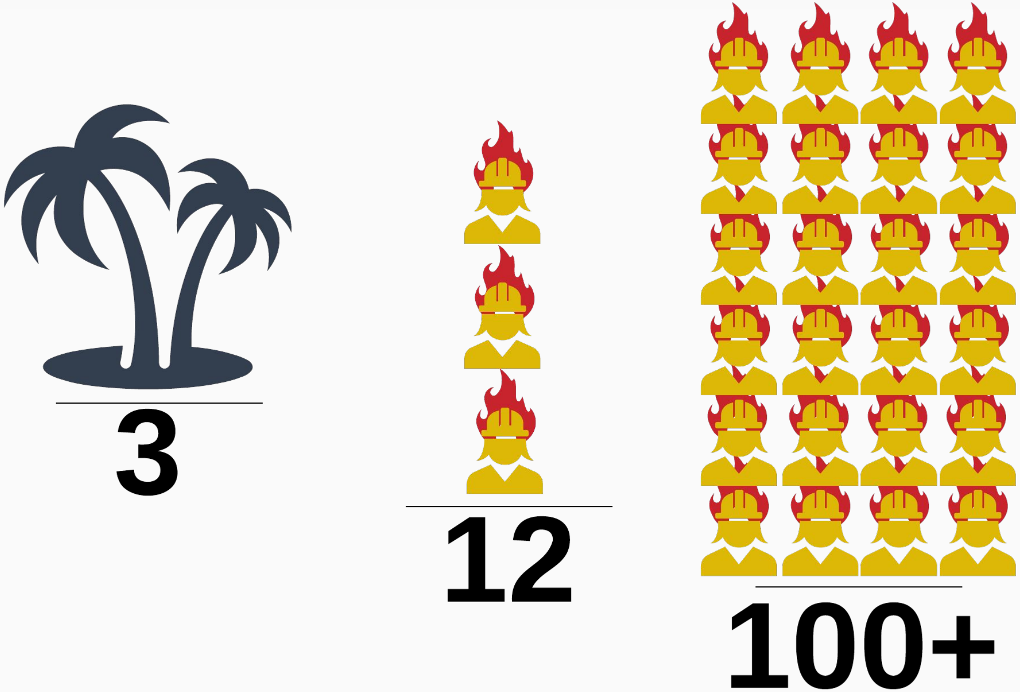
# Team Scaling



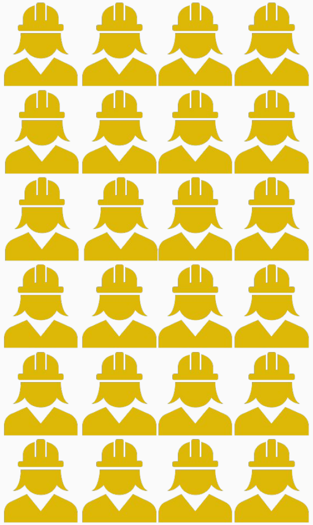
# Team Scaling



# The Future

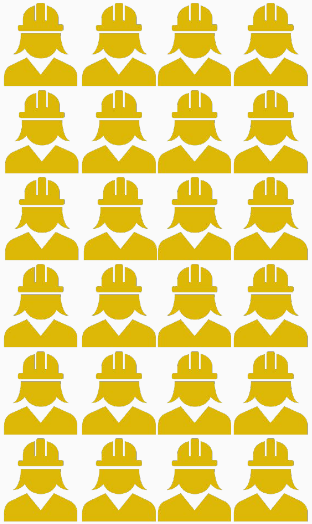


# The Future



**100+**

# The Future



**1000+**

# The Future

**1000+**

# Secure Infrastructure for Developers

## Or: Infrastructure with Vacation

# We’re Hiring!

**careers.coinbase.com**

# Questions?

**chase.evans@coinbase.com**