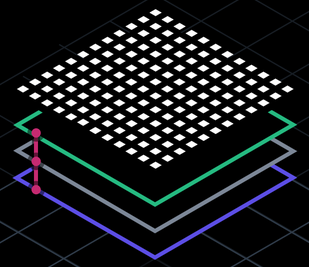
HashiCorp

DEVOPS DELIVERED

TAMING MODERN CLOUD ENVIRONMENTS WITH TERRAFORM



sheriffjackson

HashiCorp

Nic Jackson

Developer Advocate, HashiCorp

sheriffjackson

HashiCorp

HashiCorp

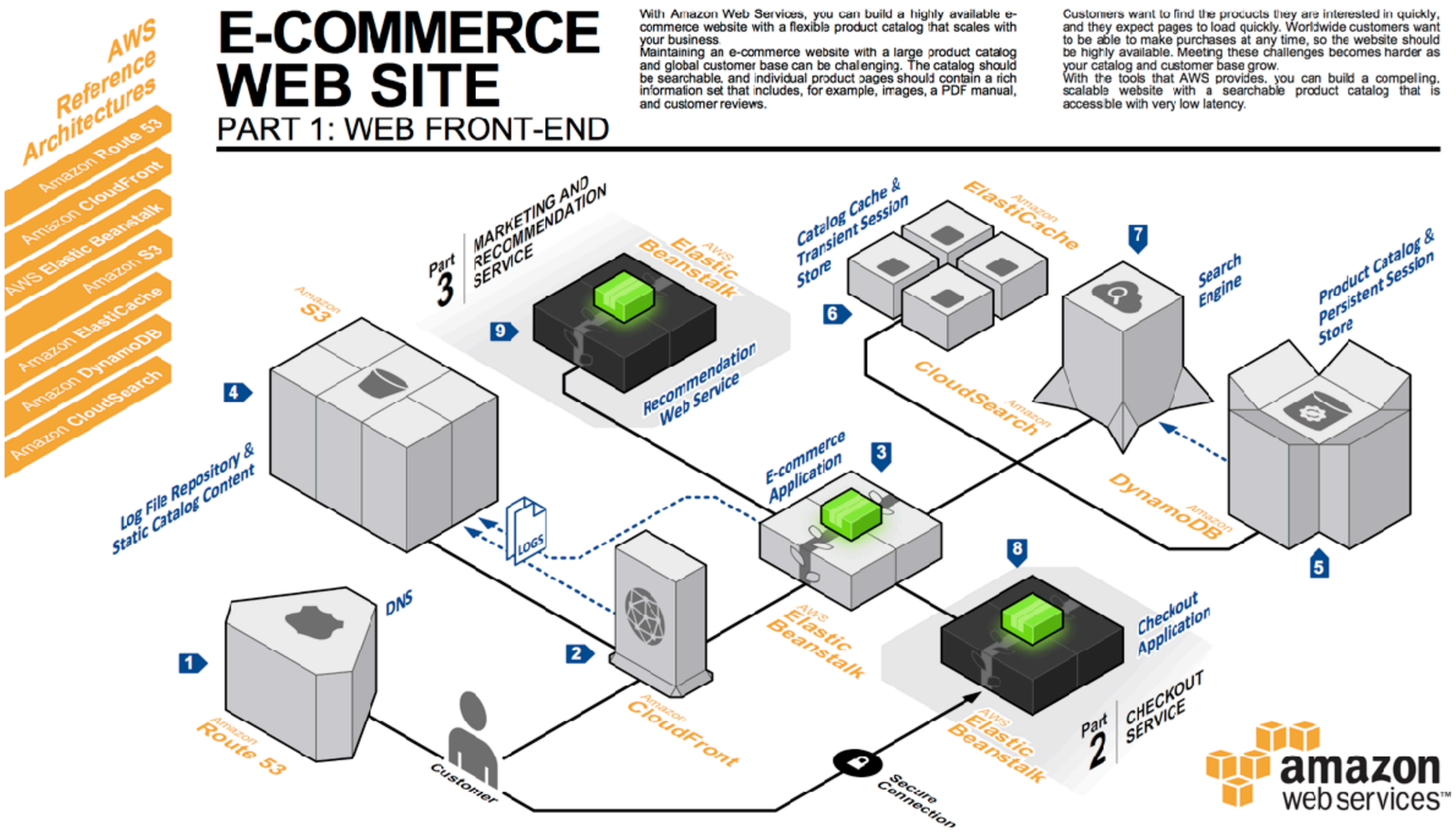
Terraform

Adgenda

- Why you have a problem
- Quick Terraform intro
- Collaborating with Terraform
  - Remote State
  - Modules
  - GitFlow
  - CI/CD
  - Managing provider secrets

Why you have a problem!

Why you have a problem



Why you have a problem

- Infrastructure as a service (e.g. AWS, Azure, GCP, DigitalOcean)
- Platform as a service (e.g. Heroku)

Why you have a problem

- Many organizations start with PaaS
- Fast default to IaaS as organization grows
- PaaS is amazing, but limitations do not scale with well with application complexity

What is Terraform?

What is Terraform? - Write

- Define infrastructure as code to increase operator productiveity and transparency
- Configuration can be stored in version control, shared, and collaborated on by teams
- If it can be codified, it can be automated

What is Terraform? - Plan

- Understand how a minor change could have cascading effects across infrastructure before executing the change
- Plans show operators what would happen, applies execute changes
- Single workflow across all infrastructure providers (AWS, GCP, Azure, OpenStack, VMware, and more)

What is Terraform? - Apply

- Use the same configuration in multiple places to reduce mistakes and save time
- Provision identical staging, QA, and production environments
- Effortlessly combine high-level system providers
- Dependency resolution means things happen in the right order

Simple Terraform config

terraform.tf

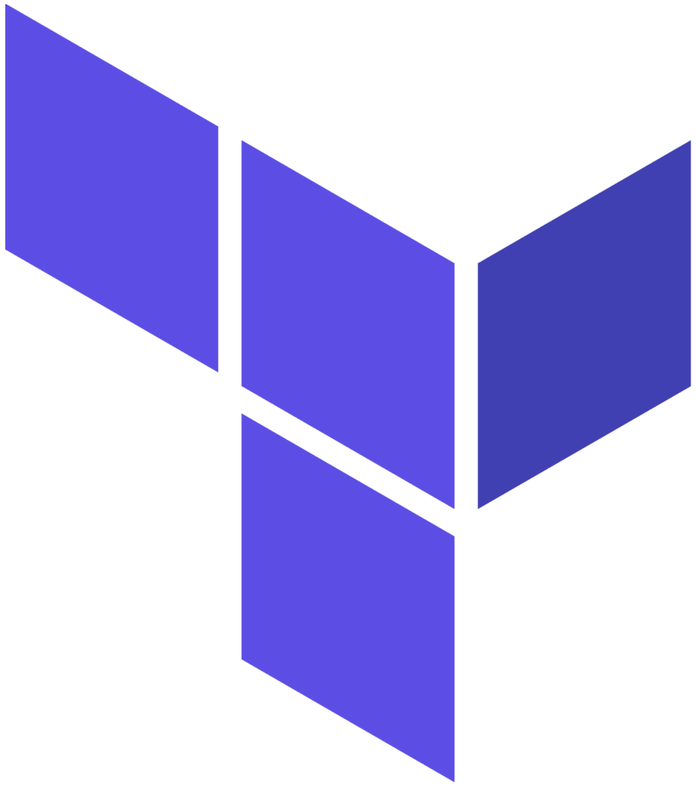
```
provider "aws" {}

data "aws_availability_zones" "available" {}

resource "aws_vpc" "main" {
  cidr_block = "${var.cidr_block}"
}
```

HashiCorp

Collaborating with Terraform

Collaborating with Terraform - 4 phases of collaboration

- Manual
- Semi-automated
- Infrastructure as Code
- Collaborative Infrastructure as Code



Terraform

Remote State

Collaborating with Terraform - Remote state

- Backends are responsible for storing state and providing an API for state locking. State locking is optional.
- Backends determine where state is stored
- 10 officially supported backends
- Allows state to be stored remotely
- Keeps sensitive information off disk

Remote state backends

- artifactory - **not** locking
- azurerm - locking
- consul - locking
- etcd - **not** locking
- gcs - locking
- http - **optional** locking
- manta - locking
- s3 - locking
- swift - **not** locking
- terraform enterprise - **not** locking

Terraform remote state

terraform.tf

```
terraform {
  backend "s3" {
    bucket = "tfremotestatenic"
    key    = "cfmngmnt"
    region = "eu-west-1"
  }
}
```

HashiCorp

Introduction to Modules

Terraform

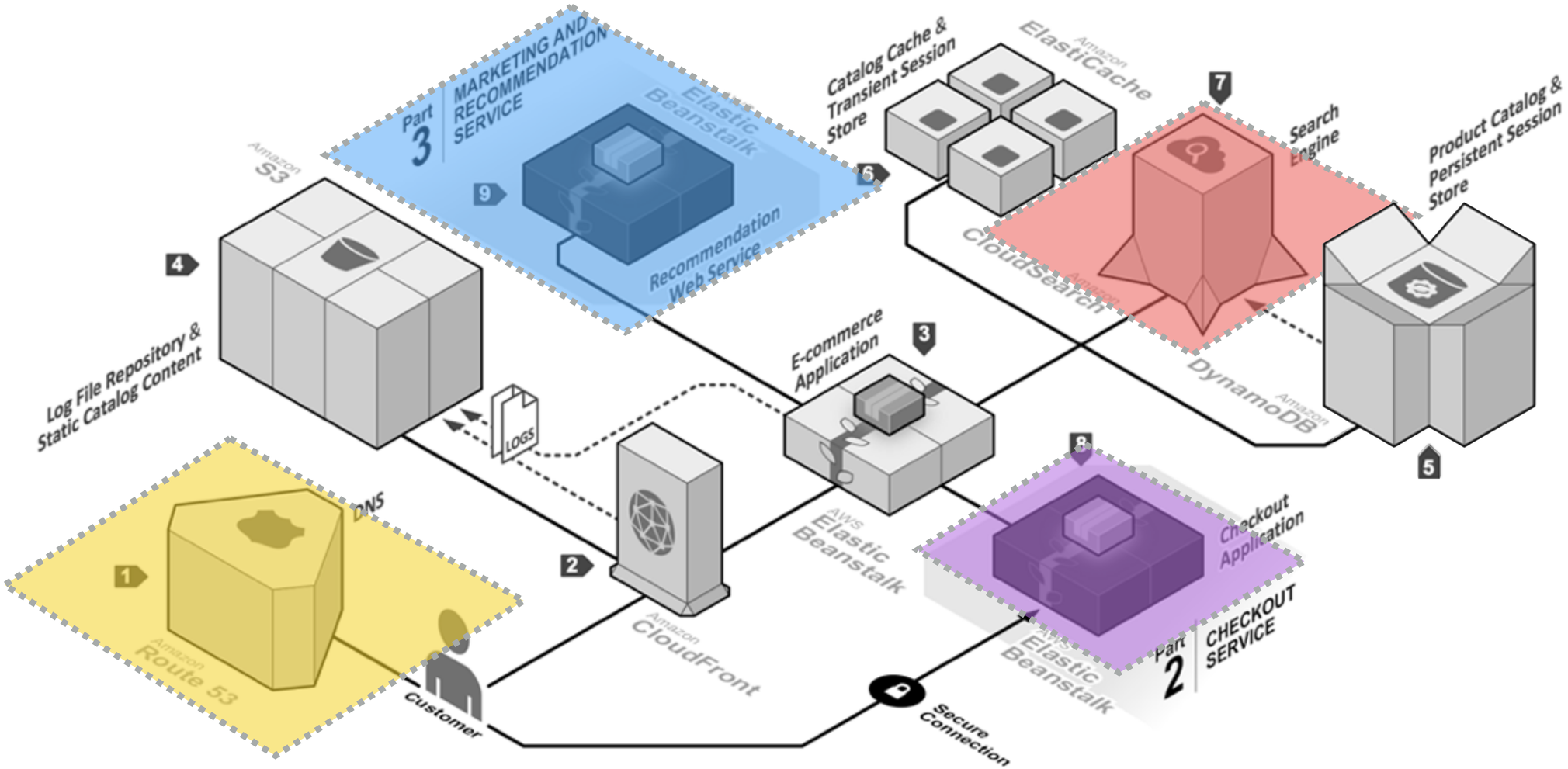
Introduction to Modules - What is a module?

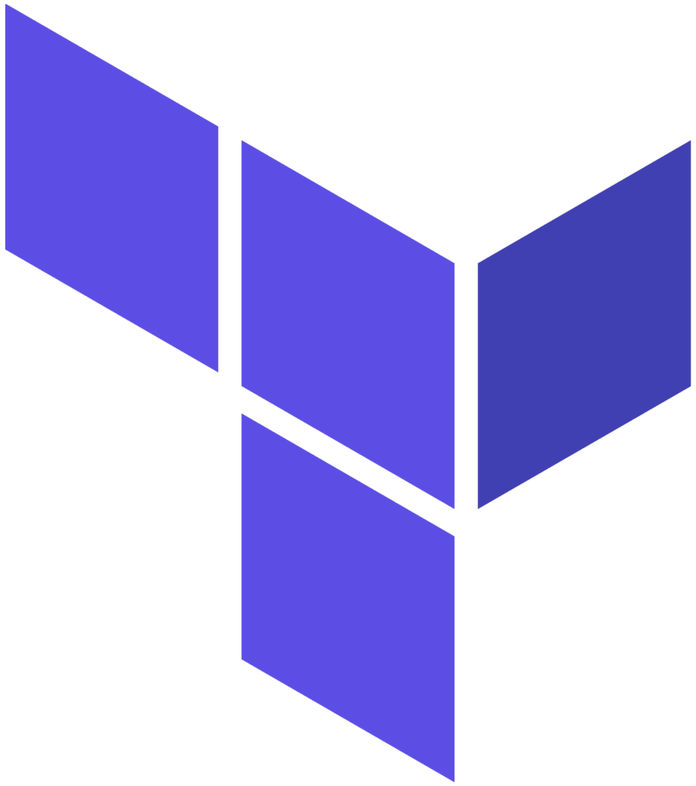
- A module is simply a blueprint for your application
- A module is a high level abstraction
- A module is your own PaaS
- Modules can be shared in Github or the Terraform Module registry

Introduction to Modules - What is a module?

- Modules abstract complexity
- Module perform a single task and have a single purpose
- Build once, use many

Introduction to Modules - Reference Architecture





Terraform

Introduction to Modules - Versioning modules

- Ability to use Git tags
- Move immutable versions across environments

Module source from Git

terraform.tf

```
provider "aws" {}

module "vpc" "default" {
  source = "git@github.com:hashicorp/example.git//subdir"

  cidr_block = "${var.cidr_block}"
}
```

HashiCorp

Module source from Git with branch reference

terraform.tf

```
provider "aws" {}

module "vpc" "default" {
  source = "git@github.com:hashicorp/example.git//subdir?ref=hotfix"

  cidr_block = "${var.cidr_block}"
}
```

HashiCorp

Introduction to Modules - Modules can contain modules

- Modules can be composed of sub-modules
- Sub-modules can be composed of sub-sub-modules
- Sub-sub-modules can be composed of sub-sub-sub-modules
- …

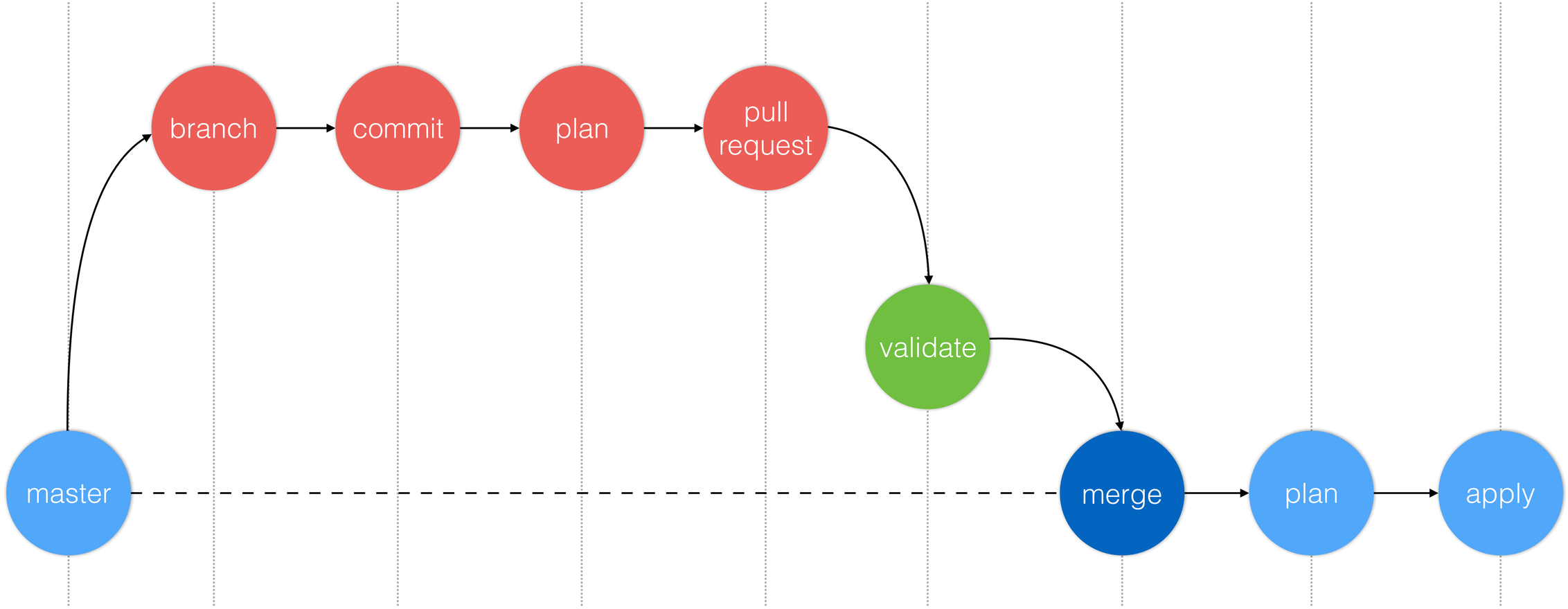
Introduction to Modules - Multi-Cloud

- It will never be possible to write a common abstraction at a resource level for an instance in different cloud providers
- Modeling infrastructure on a high level such as K8s cluster with modules allows abstraction and hides implementation

Terraform

Terraform GitFlow and CI

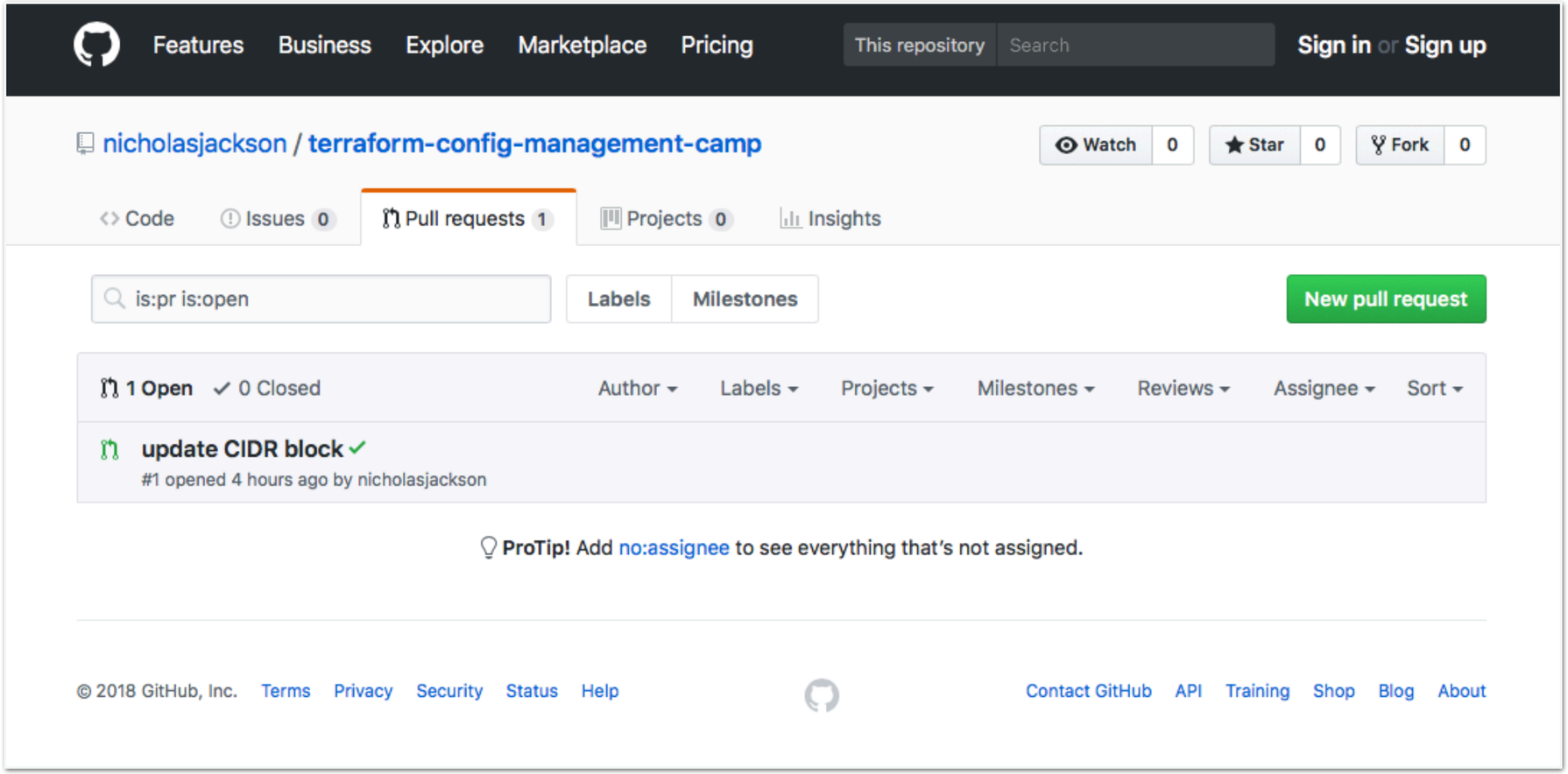
Collaborating with Terraform - Git Flow



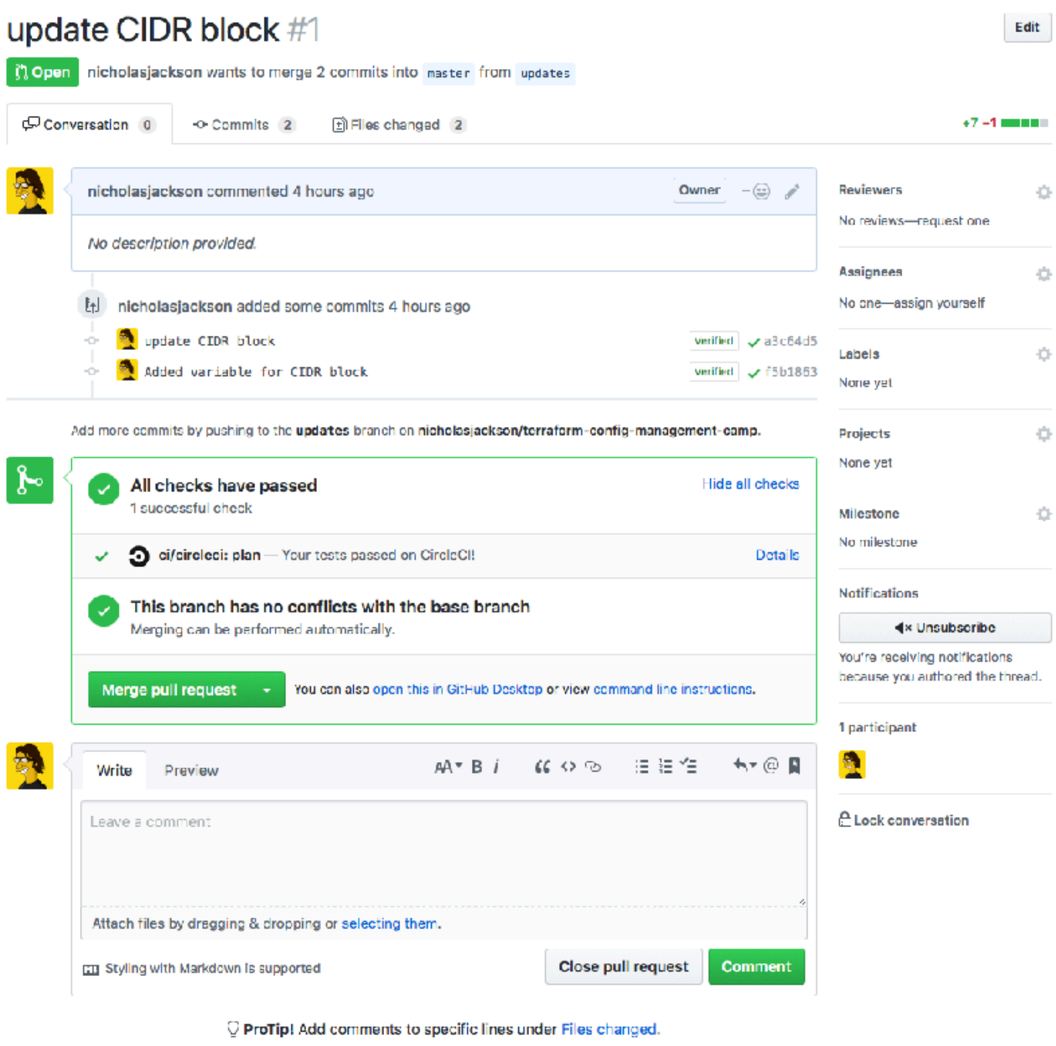
version 1.0

version 1.1

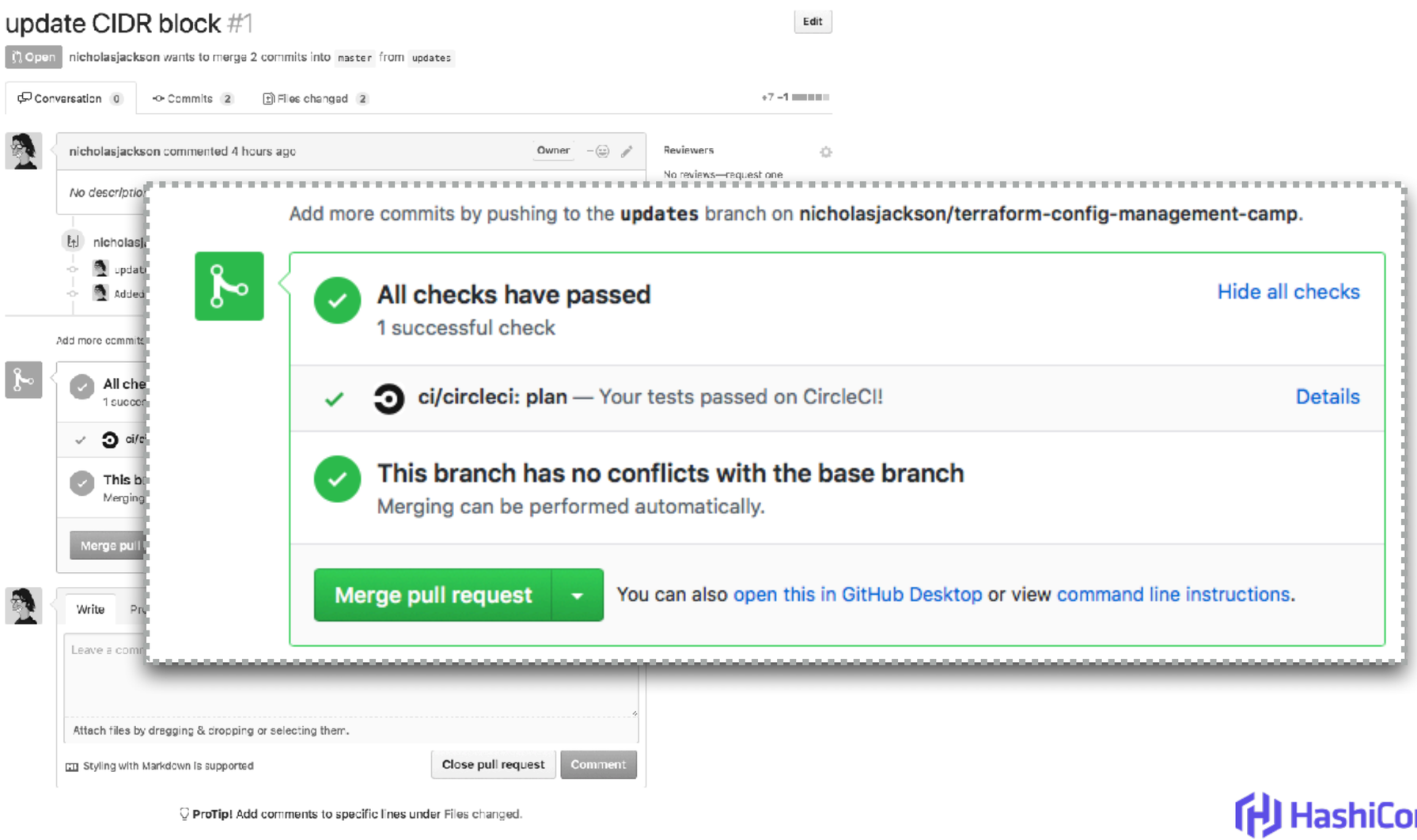
Collaborating with Terraform - GitHub



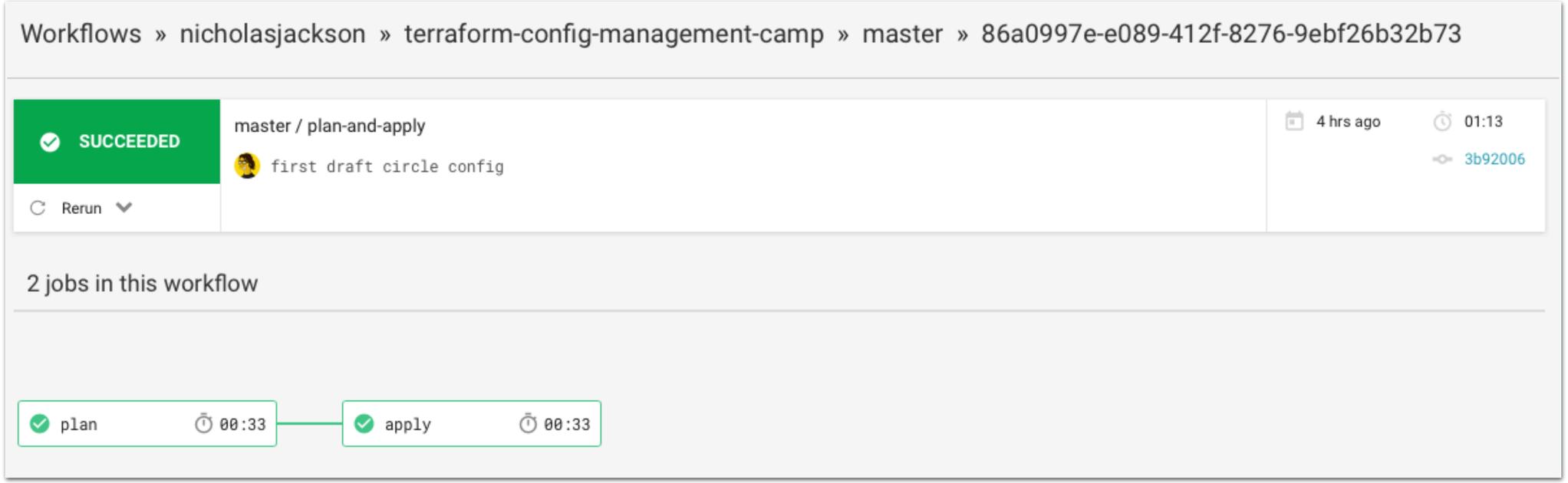
Collaborating with Terraform - Pull Requests



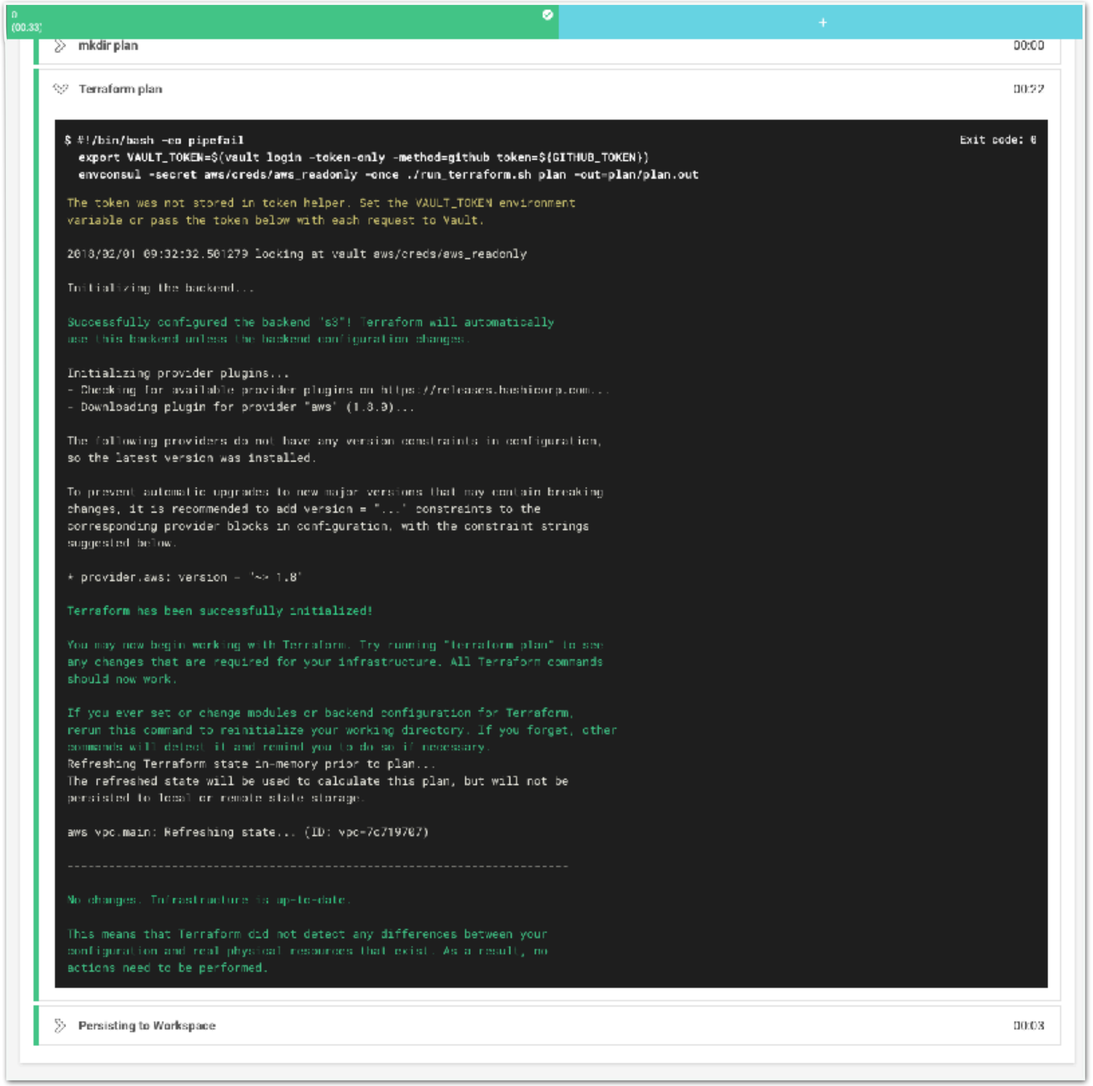
Collaborating with Terraform - Automated builds



Collaborating with Terraform - Build workflow



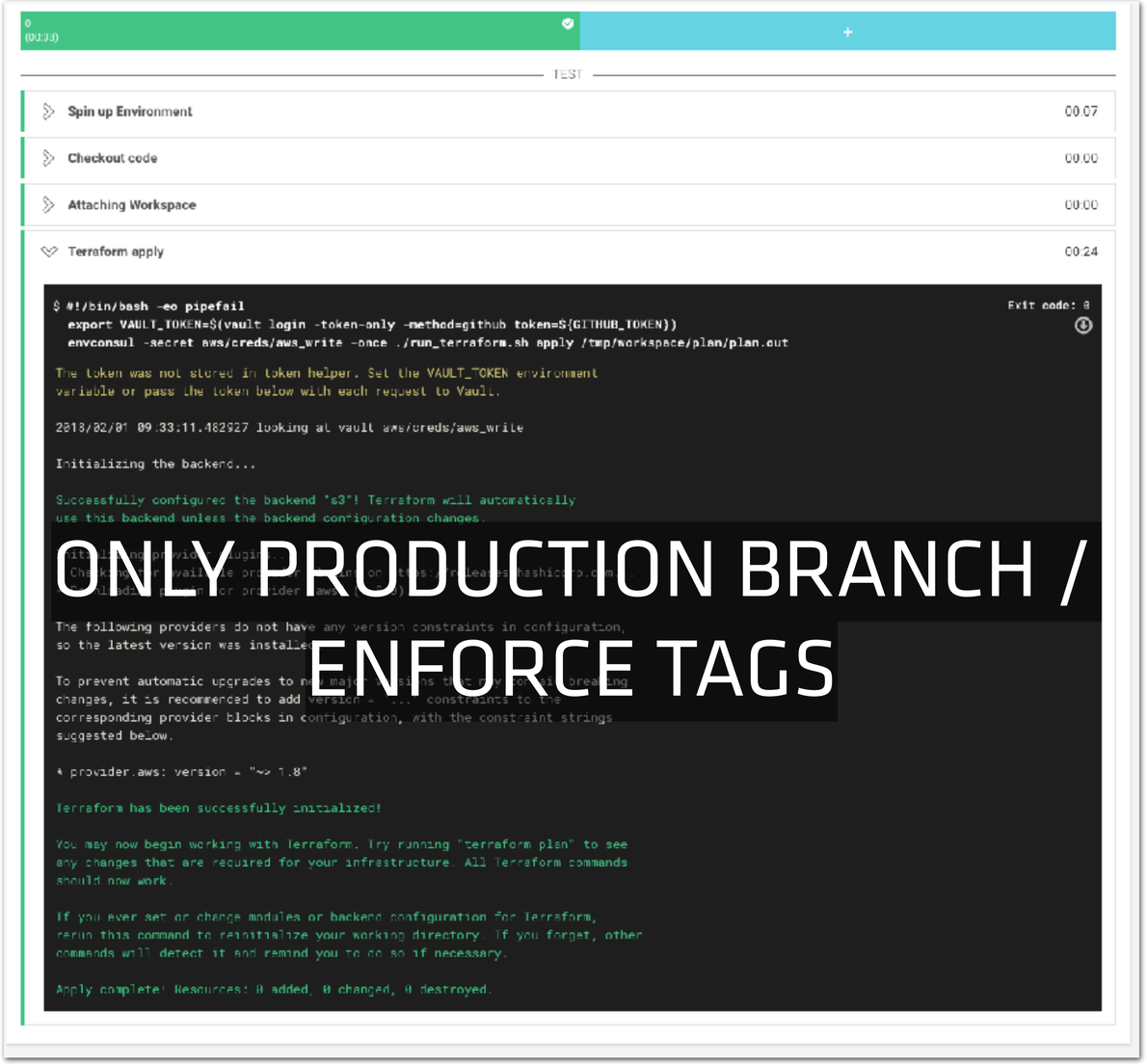
Collaborating with Terraform - Plan



sheriffjackson

HashiCorp

Collaborating with Terraform - Apply



ONLY PRODUCTION BRANCH / ENFORCE TAGS

Public Service Announcement!

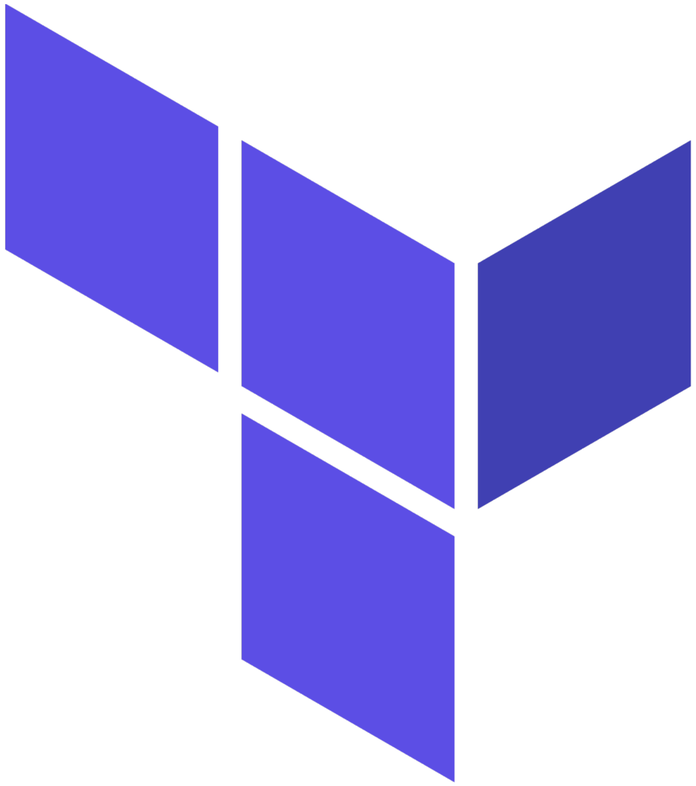
- **DO NOT** EXPOSE Vault / CI WITH PUBLIC GitHub
- THIS IS A SIMPLIFIED EXAMPLE FOR DEMONSTRATION ONLY

Collaborating with Terraform - Fork me

1) Fork the below repository
2) Create a new branch
3) **Refactor into a module**
4) Pull request

https://github.com/nicholasjackson/terraform-cfgmgmtcamp

Managing Credentials With Vault

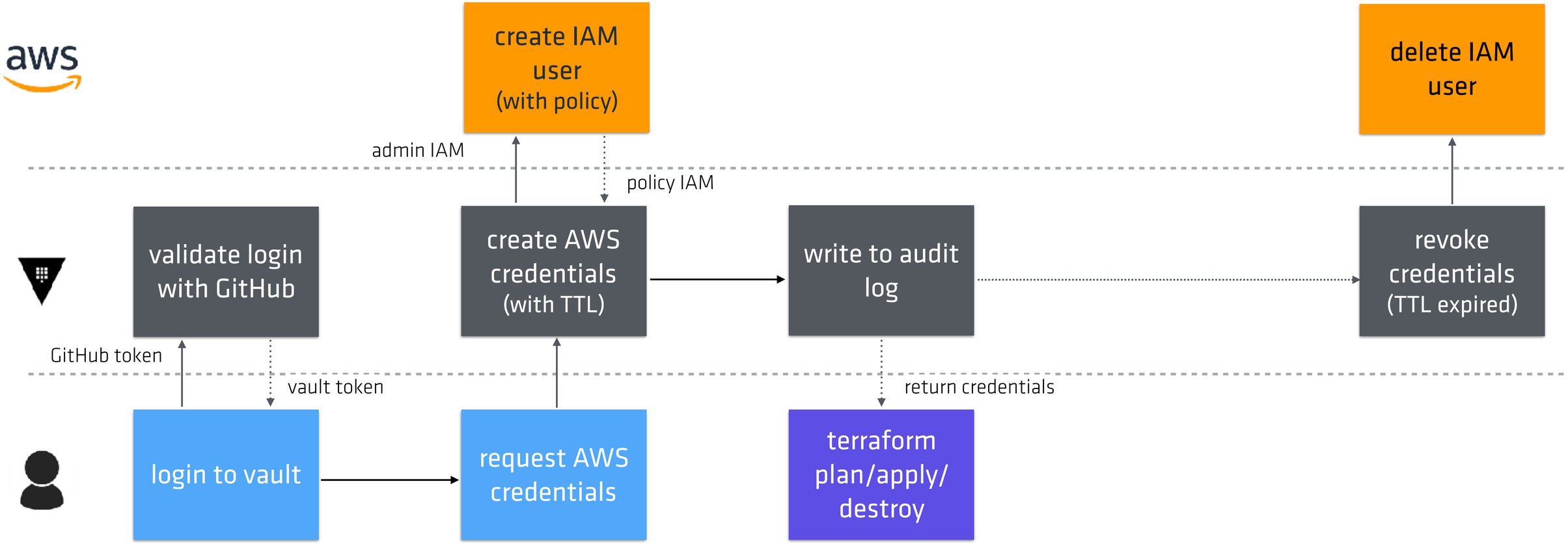


Terraform

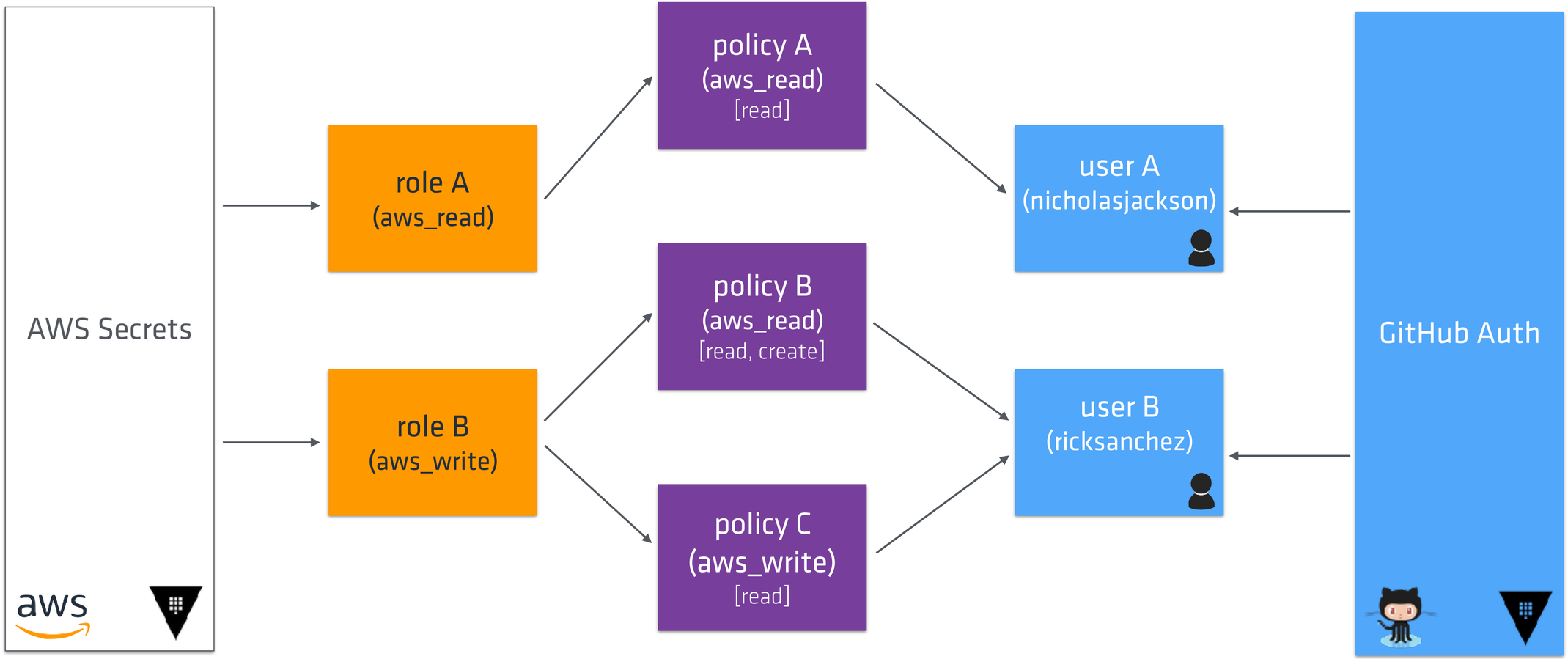
Managing Credentials with Vault - Why?

- Separating read / write credentials
- TTL on credentials with automatic revoke
- Fine grained control over access
- Audit log
- Reduce secret sprawl
- Developers / operators **DO NOT** require explicit credentials

Managing Credentials with Vault - Process



Managing Credentials with Vault - hierarchy



Configure AWS secret engine

Terminal

```
# enable the vault secrets engine for AWS
$ vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/

# add IAM credentials which have the capability to manage IAM credentials
$ vault write aws/config/root \
    access_key=AKIAJWVN5Z4FOFT7NLNA \
    secret_key=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 \
    region=us-east-1
```

HashiCorp

Configure GitHub auth

Terminal

```
# enable the vault secrets engine for AWS
$ vault auth enable github

# set the auth endpoint to a GitHub organization
$ vault write auth/github/config organization=hashicorp
```

HashiCorp

Create a read-only AWS IAM policy

terraform.tf

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": ["s3:ListBucket"],
       "Resource": ["arn:aws:s3:::tfremotestatenic"]
     },
     {
       "Effect": "Allow",
       "Action": [
         "s3:GetObject"
       ],
       "Resource": ["arn:aws:s3:::tfremotestatenic/*"]
     },
     {
         "Effect": "Allow",
         "Action": "ec2:Describe*",
         "Resource": "*"
     },
     {
         "Effect": "Allow",
         "Action": "elasticloadbalancing:Describe*",
         "Resource": "*"
     },
     ...
```

https://github.com/nicholasjackson/terraform-cfgmgmtcamp/blob/master/vault/templates/aws_read.json

HashiCorp

Create a write AWS IAM policy

terraform.tf

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": ["s3:ListBucket"],
       "Resource": ["arn:aws:s3:::tfremotestatenic"]
     },
     {
       "Effect": "Allow",
       "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
       ],
       "Resource": ["arn:aws:s3:::tfremotestatenic/*"]
     },
     {
            "Action": "ec2:*",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:*",
            "Resource": "*"
        },
```

https://github.com/nicholasjackson/terraform-cfgmgmtcamp/blob/master/vault/templates/aws_write.json

HashiCorp

Configure roles

Terminal

```
# create the read only role
$ vault write aws/roles/aws_readonly policy=@aws_read_policy.json
Success! Data written to: aws/roles/aws_readonly

# create the write role
$ vault write aws/roles/aws_write policy=@aws_write_policy.json
Success! Data written to: aws/roles/aws_write
```

HashiCorp

Create a read-only Vault policy

terraform.tf

```
path "aws/creds/aws_readonly" {
  capabilities = ["read"] # "create", "read", "update", "delete", "list"
}
```

HashiCorp

Create a write Vault policy

terraform.tf

```
path "aws/creds/aws_write" {
  capabilities = ["read"]
}
```

HashiCorp

Add the policy to Vault

Terminal

```
# Add the read only policy to vault
$ vault write sys/policy/aws_readonly policy=@aws_readonly.hcl
Success! Data written to: sys/policy/aws_readonly

# Add the write policy to vault
$ vault write sys/policy/aws_write policy=@aws_write.hcl
Success! Data written to: sys/policy/aws_write
```

HashiCorp

Assign the policy to the GitHub user

Terminal

```
# Assign the policy to the GitHub user
$ vault write auth/github/map/users/nicholasjackson
value=aws_readonly,aws_write
Success! Data written to: auth/github/map/users/nicholasjackson

# Or assign the policy to a GitHub team
$ vault write auth/github/map/teams/hasicorp value=aws_readonly,aws_write
Success! Data written to: auth/github/map/teams/hashicorp
```

HashiCorp

Login with GitHub obtain IAM credentials

Terminal

```
# login to vault with GitHub token
$ vault login -token-only -method=github token=${GITHUB_TOKEN}
The token was not stored in token helper. Set the VAULT_TOKEN environment
variable or pass the token below with each request to Vault.

ebb39457-cdb8-41ae-c227-9e0245837873

# use vault token to obtain AWS credentials
$ VAULT_TOKEN=ebb39457-cdb8-41ae-c227-9e0245837873 vault read aws/creds/
aws_readonly
Key                 Value
---                 -----
lease_id            aws/creds/aws_readonly/0bb798a3-c4fd-0609-6bcd-
f34673156c8a
lease_duration      15m
lease_renewable     true
access_key          AKIAJMWSOATIG3VQJCNA
secret_key          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
security_token      <nil>
```

HashiCorp

We can also do this with a module

```
provider "vault" {
  address = "http://pi01.local.demo.gs:8200"
}

module "vault" {
  source = "./vault"

  github_org            = "hashicorp"

  aws_access_key_id     = "xxxxxxxxxxxxxx"
  aws_access_key_secret = "xxxxxxxxxxxxxxxxxxxxxx"
  aws_region            = "us-east-1"
}
```

https://github.com/nicholasjackson/terraform-cfgmgmtcamp/tree/master/vault

HashiCorp

Putting it all together

HashiCorp

DEVOPS DELIVERED

THANKS FOR LISTENING

@sheriffjackson, nic@hashicorp.com

FURTHER INFO:

- https://www.terraform.io
- https://www.vaultproject.io
- https://registry.terraform.io

EXAMPLE PROJECT:

- https://github.com/nicholasjackson/terraform-cfgmgmtcamp

AGENDA

14:55 - CoreOS baremetal provisioning with Terraform, Ignition and Matchbox
- ★ Rafael Porres Molina

15:35 - Break

15:55 - Introduction to provisioning basic infrastructure on Google Cloud Platform with Terraform
- ★ Stein Inge Morisbak

16:35 - Terraform for fun and profit
- ★ Bram Vogelaar & Julien Pivottto

17:15 - Terraboard
- ★ Raphael Pinson

HashiCorp
DEVOPS DELIVERED