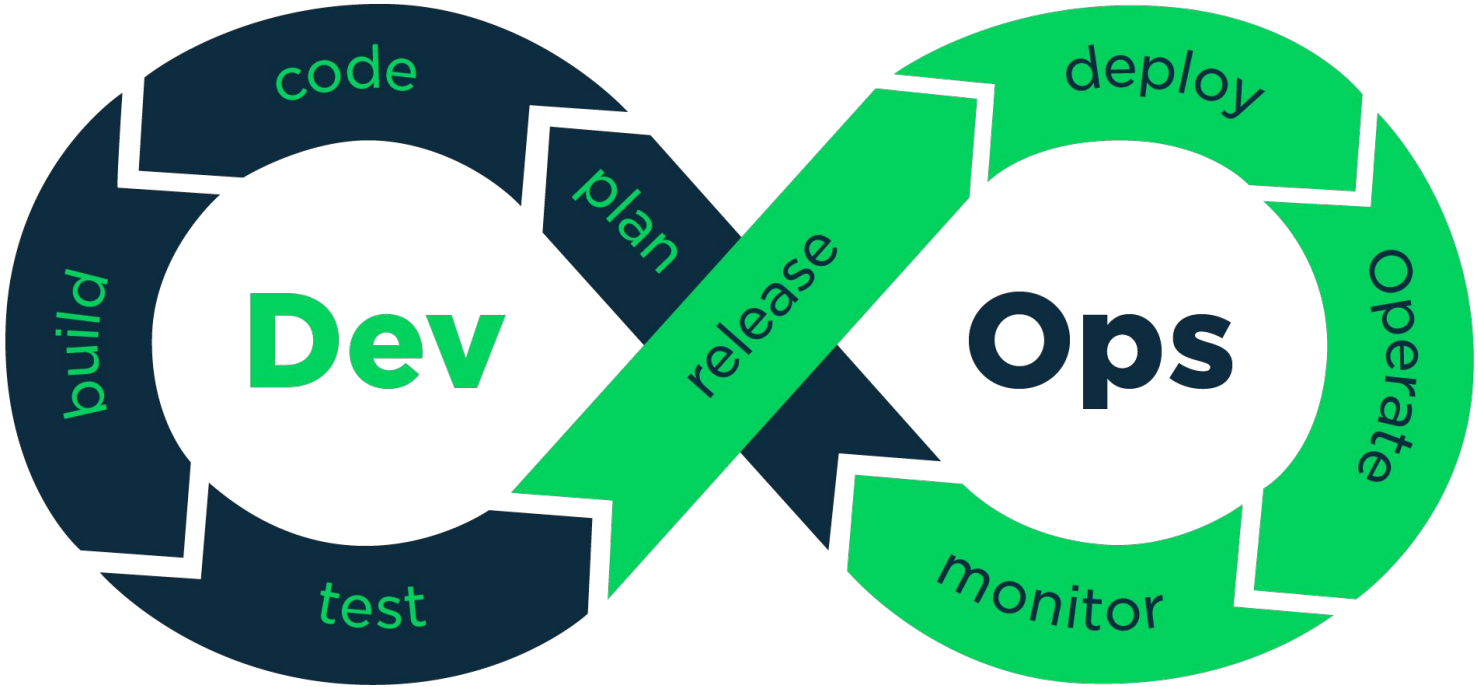
KITMAN LABS

INCREASING THE IMPACT OF SPORTS DATA

Kitman Labs is changing how elite sports teams across the globe use data to achieve their goals. Our unique outcome-driven analytics power teams with insights about the drivers of performance and health. These insights drive how teams develop athletes over time, and manage them on a day to day basis.

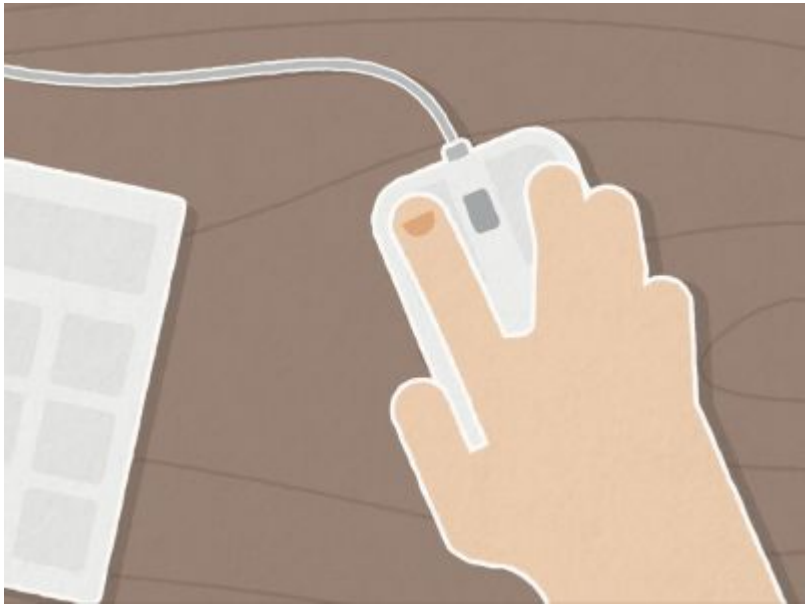
We partner with over 200 teams in the:
Premier League | Bundesliga | Pro14 | RFU
NFL | NHL | MLB | MLS | AFL | NRL

Who Practices Devops at their organisation?



What About ClickOps™?

Point and click adventures in the console



Infrastructure as Code

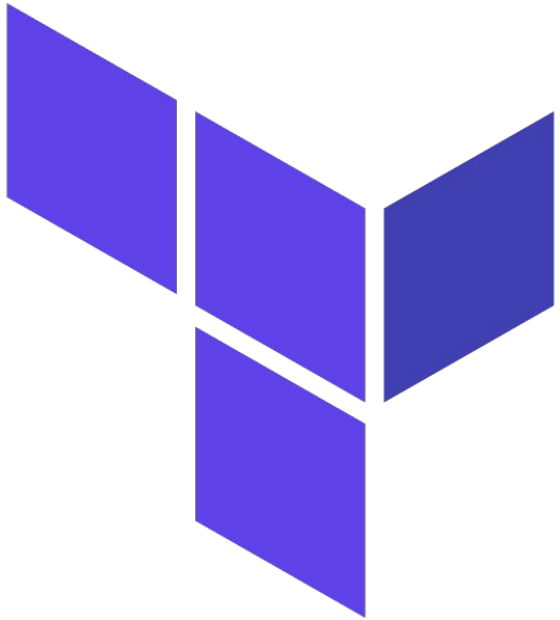
Infrastructure defined in some high level language

Version controlled single source of truth

Pull Requests drive infrastructure changes

Reusable components for your organisation with safe defaults

Automatable



How?

What tooling is available to enable this?

AWS CLI

AWS REST APIs

SDKs for most languages (Ruby, Python, Go, Javascript)

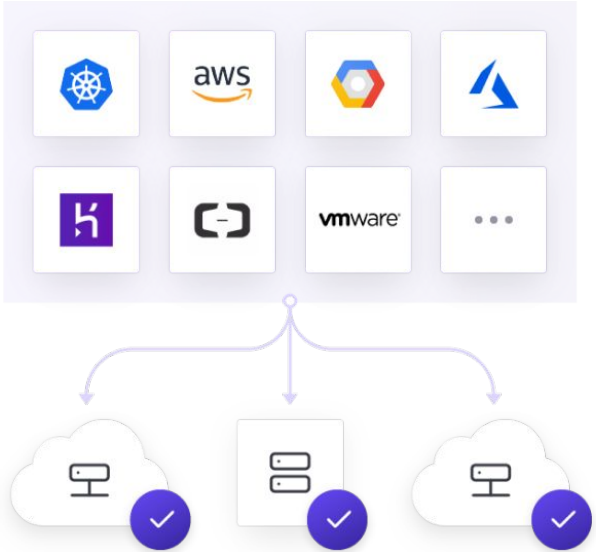
Cloudformation / Serverless Framework / SAM / CDK

“Traditional” Config Management Tools, Ansible, Chef, Puppet

Terraform

HashiCorp

Terraform



HashiCorp
Packer

KITMAN LABS

Hello World

```
resource "aws_key_pair" "my_key" {
  public_key = "ssh-rsa AAAABBBBB"
}

resource "aws_instance" "my_instance" {
  ami           = "ami-0dc9a8d2479a3c7d7"
  instance_type = "t3.small"
  key_name      = "${aws_key_pair.my_key.key_name}"
}
```

Senior YAML Engineer

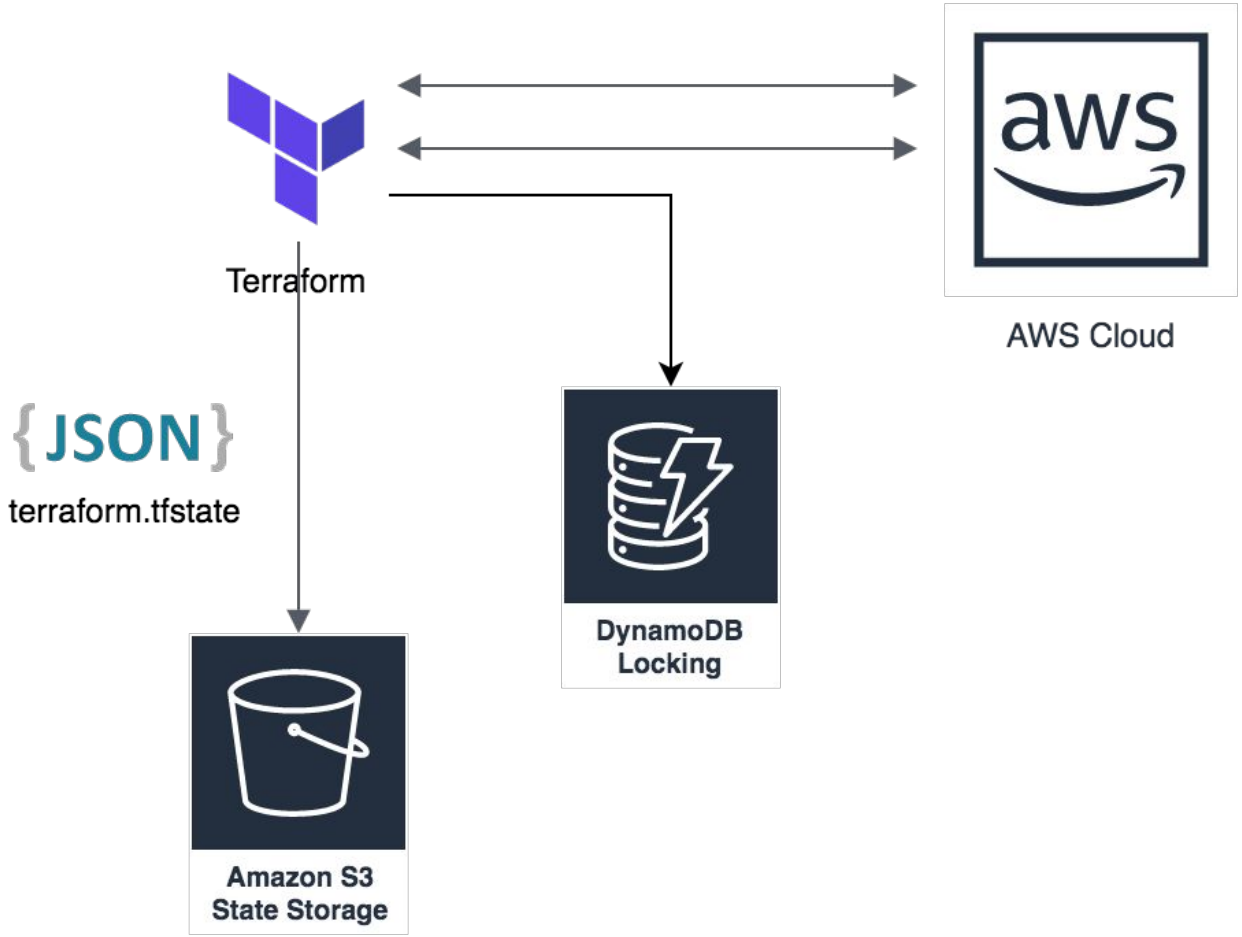
Scribd • Canada, KS, US

Posted 1 month ago • Be among the first 25 applicants

What You'll Do

Core Infrastructure plays a fundamental role in both our on-premise infrastructure and our on-going efforts to upgrade infrastructure towards AWS. As an infrastructure engineer you will be part of implementing the infrastructure to support the continued growth of Scribd's foundation. You will be part of the team which manages our existing on-premise datacenter, while helping service owners move into a newer AWS-centric

```
variable "servers" {
  type    = "list"
  default = ["server1", "server2", "server3"]
}

resource "aws_key_pair" "my_key" {
  public_key = "ssh-rsa AAAABBBBB"
}

resource "aws_instance" "my_instance" {
  name          = "${element(var.servers, count.index)}"
  ami           = "ami-0dc9a8d2479a3c7d7"
  instance_type = "t3.small"
  key_name      = "${aws_key_pair.my_key.key_name}"
  count         = "${length(var.servers)}"
}
```


Terraform
aws
AWS Cloud
{ JSON }
terraform.tfstate
DynamoDB
Locking
Amazon S3
State Storage

Terraform State and Resources

If resource exists in the state file and not defined with HCL

Resource is destroyed

If resource is defined in the HCL but not in the state file

Resource is created

Google Cloud Platform

ndmckinley / terraform-provider-dominos

Watch 5 | Unstar 481 | Fork 16

Code | Issues 8 | Pull requests 2 | Actions | Security | Insights

The Terraform plugin for the Dominos Pizza provider.

22 commits | 2 branches | 0 releases | 3 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

ndmckinley Merge pull request #1 from davewongillies/tf_fmt — Latest commit 70f8407 on 7 May

bin	Add binary, for ease of distribution. should be low traffic.	7 months ago
docs	Merge pull request #1 from davewongillies/tf_fmt	6 months ago
.gitignore	Update docs to point to binary directly.	7 months ago
LICENSE	Create LICENSE	7 months ago
Makefile	Write docs first draft, rename address.	7 months ago
README.md	Fmt the example code	7 months ago
data_source_address.go	Write docs first draft, rename address.	7 months ago
data_source_menu.go	Initial commit - working dominos pizza orderer, tracker.	7 months ago
data_source_menu_item.go	Initial commit - working dominos pizza orderer, tracker.	7 months ago
data_source_store.go	Initial commit - working dominos pizza orderer, tracker.	7 months ago
data_source_tracking.go	Initial commit - working dominos pizza orderer, tracker.	7 months ago
main.go	Initial commit - working dominos pizza orderer, tracker.	7 months ago
provider.go	Write docs first draft, rename address.	7 months ago
resource_order.go	Update order with results from further testing.	7 months ago

README.md

Terraform Provider for Dominos Pizza

KITMAN LABS

```
provider "dominos" {
  first_name    = "My"
  last_name     = "Name"
  email_address = "my@name.com"
  phone_number  = "15555555555"

  credit_card {
    number = 123456789101112
    cvv    = 1314
    date   = "15/16"
    zip    = 18192
  }
}

data "dominos_address" "addr" {
  street = "123 Main St"
  city   = "Anytown"
  state  = "WA"
  zip    = "02122"
}

data "dominos_store" "store" {
  address_url_object = "${data.dominos_address.addr.url_object}"
}

data "dominos_menu_item" "item" {
  store_id     = "${data.dominos_store.store.store_id}"
  query_string = ["philly", "medium"]
}
```

The Future is Here

We can now write software that orchestrates and manages the complete lifecycle of the infrastructure and services that power our business

Story Time

Infra Sucks

New hire asked to implement infrastructure for a new service they were working on

Them: “Oh no”

MyOrg / terraform-modules Private

Watch 1 | Star 0 | Fork 0

Code | Pull requests 7 | Actions | Projects 0 | Wiki | Security | Insights | Settings

Added chronos IAM, network and EC2 #658

Edit

Merged new hire merged 1 commit into `master` from `ic-added_chronos_asg` yesterday

Conversation 2 | Commits 1 | Checks 1 | Files changed 8 +160 −15

new hire commented yesterday • edited

Description of change

- Added network config, IAM and EC2 config for Chronos

Related Github Issues

MyOrg/issues#11369

Terraform plan

▶ Network Plan

▶ IAM Plan

▶ EC2 Plan

Reviewers
conzy

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone

NLABS

new hire commented yesterday • edited

Description of change

- Added network config, IAM and EC2 config for Chronos

Related Github Issues

MyOrg/issues#11369

Terraform plan

▼ Network Plan

```
Terraform will perform the following actions:

  + aws_security_group.chronos_servers
      id:                     <computed>
      arn:                    <computed>
      description:            "For the Chronos EC2 Instances"
      egress.#:               <computed>
      ingress.#:              <computed>
      name:                   "chronos_servers"
      owner_id:               <computed>
      revoke_rules_on_delete: "false"
      tags.%:                 "1"
      tags.Name:              "chronos_servers"
      vpc_id:                 "vpc-000000000000"
```

v0.12.69

11db627

Verified

Chronos resources

Edit

new hire released this 3 days ago · 6 commits to master since this release

Added IAM, Network and EC2 configs

▼ **Assets** 2

- **Source code** (zip)
- **Source code** (tar.gz)

v0.12.68

4f9aaf9

DynamoDB IAM -

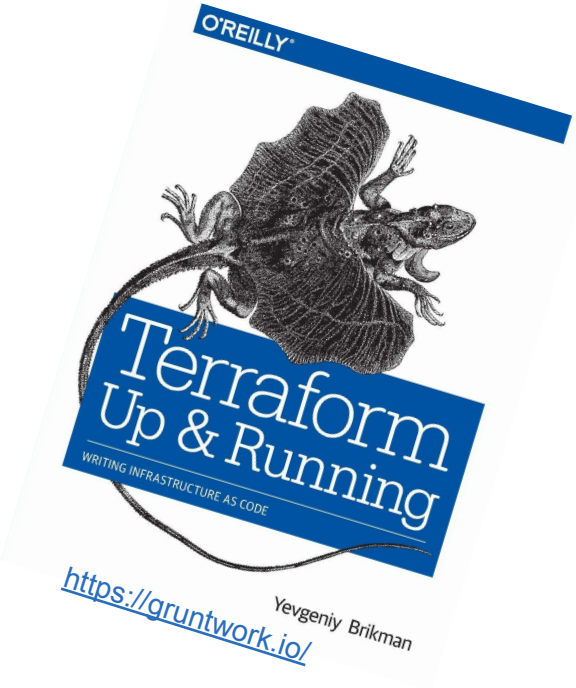
Edit

```
.
└── global
    └── organizations
        ├── production
        │   ├── eu-west-1
        │   │   ├── ec2
        │   │   ├── network
        │   │   └── rds
        │   └── global
        │       └── iam
        ├── sandbox
        │   ├── eu-west-1
        │   │   ├── ec2
        │   │   ├── network
        │   │   └── rds
        │   └── global
        │       └── iam
        └── staging
            ├── eu-west-1
            │   ├── ec2
            │   ├── network
            │   └── rds
            └── global
                └── iam
```

ec2 @ v1.0.0

ec2 @ v2.1.1

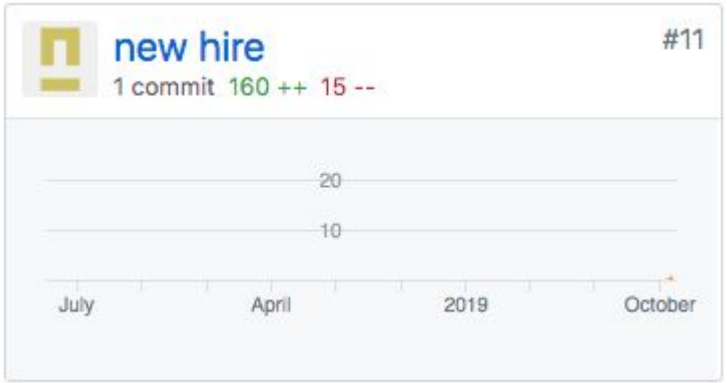
ec2 @ v2.0.0



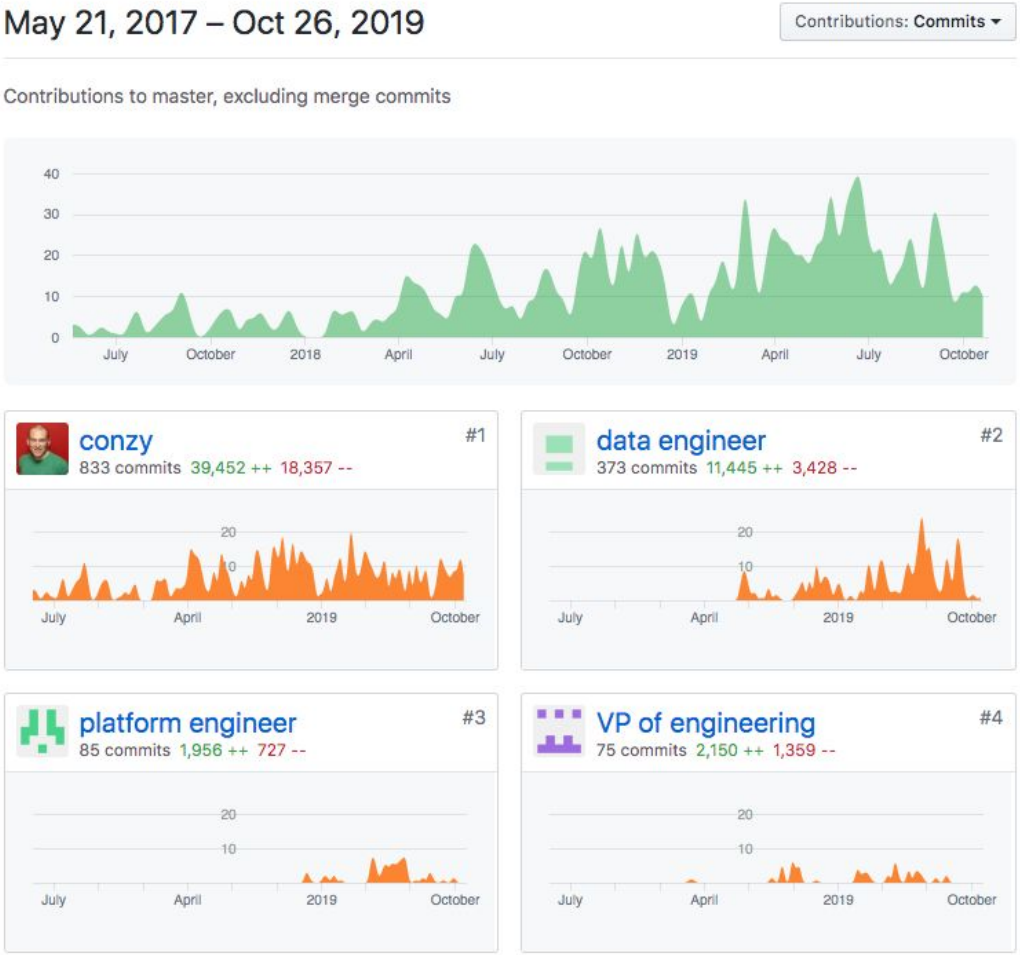
Infra is Awesome!

New Hire feels good

Reuse existing battle tested modules



Terraform is for everyone!



KITMAN LABS

Data Science

KITMAN LABS

Data Science

Autonomy: DS are largely self sufficient

Heavily utilize managed / serverless services

AWS Batch / Step Functions / Lambda / Sagemaker

S3 Event Notifications for event driven pipelines

How did we get here?

Requirements

Smaller blast radius

Principle of Least Privilege

Self service for individual teams / business units

Multi Region

Lower lead time for changes

Lower change failure rate

100% Terraform coverage

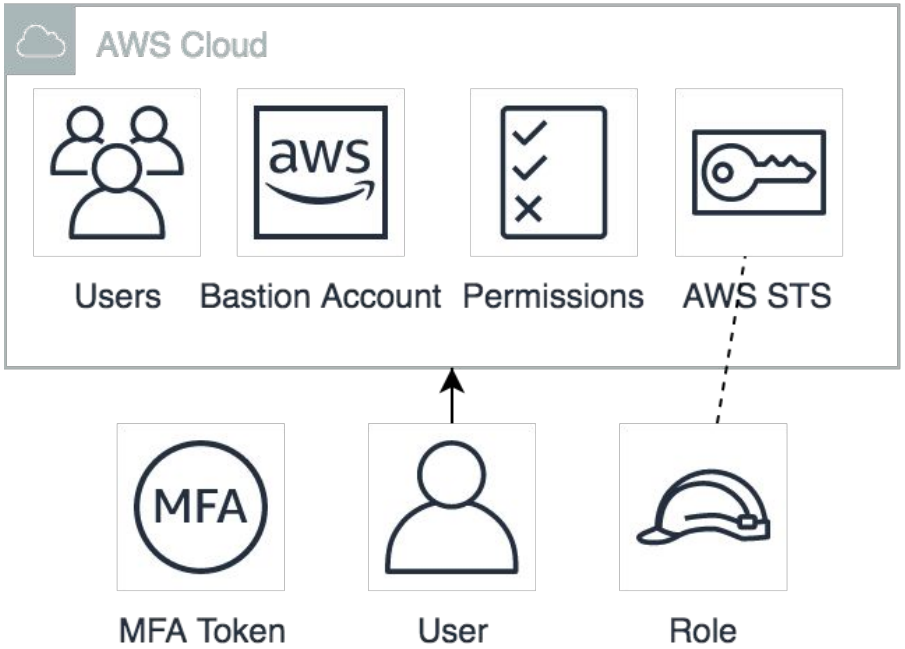
Audit Trail

Decision Time

Green Fields*

Multiple Accounts

- Bastion account
- Multi Factor Authentication
- Every API call is logged
- **AWS accounts are free!**



12 Factor Apps

Stateless

No hard coded regions, bucket names, endpoints etc

Store config in the environment

https://12factor.net/

Automate everything*

Amazon machine image building

(Code Pipeline, Codebuild, Packer, Ansible)

Container image builds

(CircleCI, Docker, ECR)

Lambda deployment bundle builds

(CircleCI, Bash / other duct tape, S3)

Pull Requests Drive Everything

Utilize managed services

Codepipeline

Codebuild

Parameter Store

Elastic Container Registry

Codedeploy

Batch

Step Functions

Fargate

Guard Duty

Lambda

Session Manager

Certificate Manager

What about vendor lock in?

Lock in is a lie. Fight me.

Limiting yourself to the lowest common denominator abstraction.

Utilizing managed services saves money and engineering resources

7 Platform engineers

https://marcotroisi.com/why-aws/

Story Time 2

What is the worst time for a TLS certificate to expire?

AWS Certificate Manager

Fully managed certificate issuance / rotation

Free!!!

```
module "example_domain_cert" {
  source = "../acm"
  domain = "example.com"
}
```

AWS Certificate Manager

```
resource "aws_acm_certificate" "this" {
  domain_name       = "*.${var.domain}"
  validation_method = "DNS"
}

resource "aws_route53_record" "this_validation" {
  name    = "${aws_acm_certificate.this.domain_validation_options.0.resource_record_name}"
  type    = "${aws_acm_certificate.this.domain_validation_options.0.resource_record_type}"
  zone_id = "${data.aws_route53_zone.this_zone.id}"
  records = ["${aws_acm_certificate.this.domain_validation_options.0.resource_record_value}"]
  ttl     = 60
}

resource "aws_acm_certificate_validation" "this_validation" {
  certificate_arn         = "${aws_acm_certificate.this.arn}"
  validation_record_fqdns = ["${aws_route53_record.this_validation.fqdn}"]
}
```

Services

Managed

DIY

Implement everything in Terraform

Use **community** modules where they make sense

Implement things correctly*

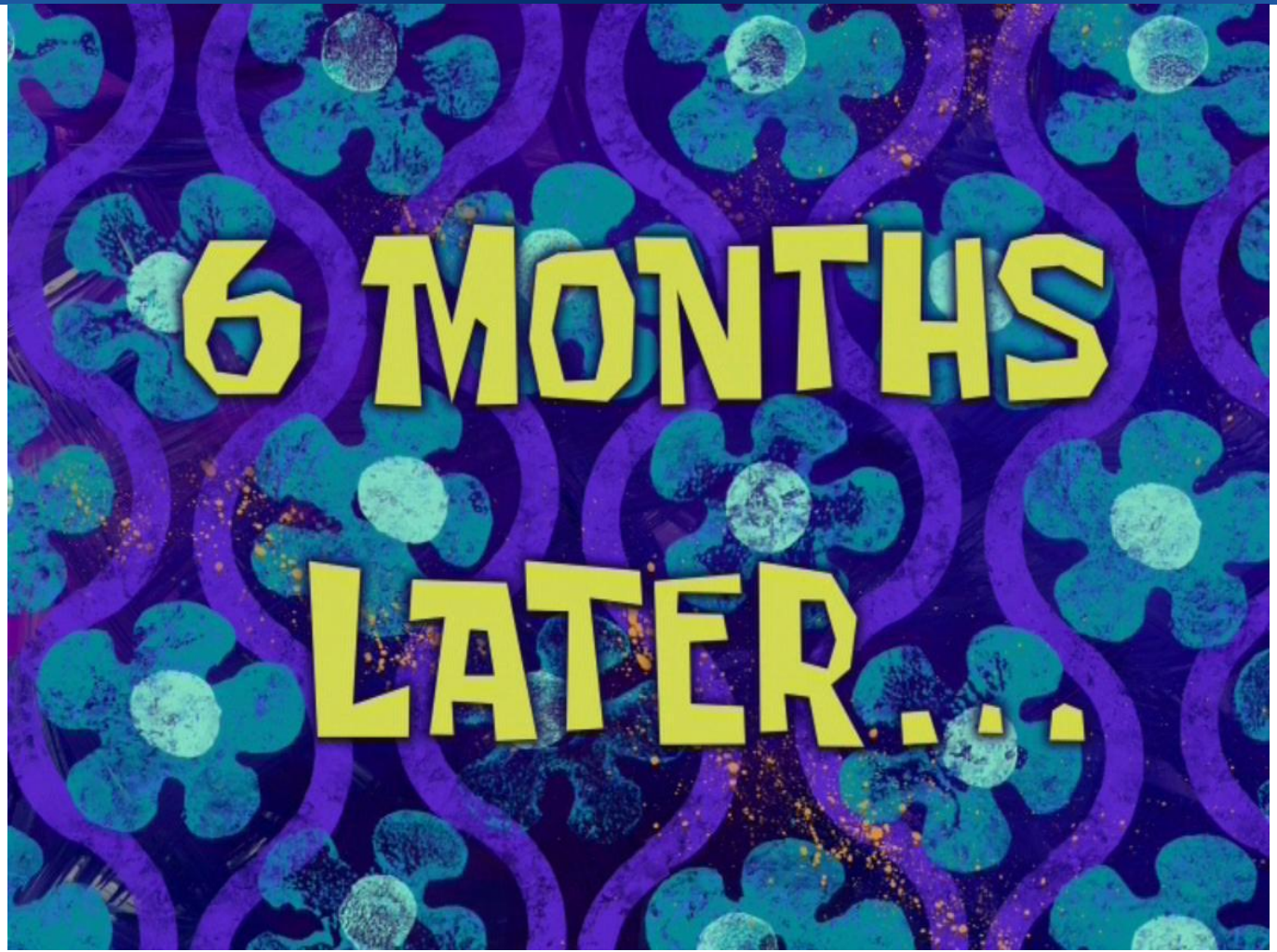
Best practices

Terraform Registry

KITMAN LABS

Terraform Registry

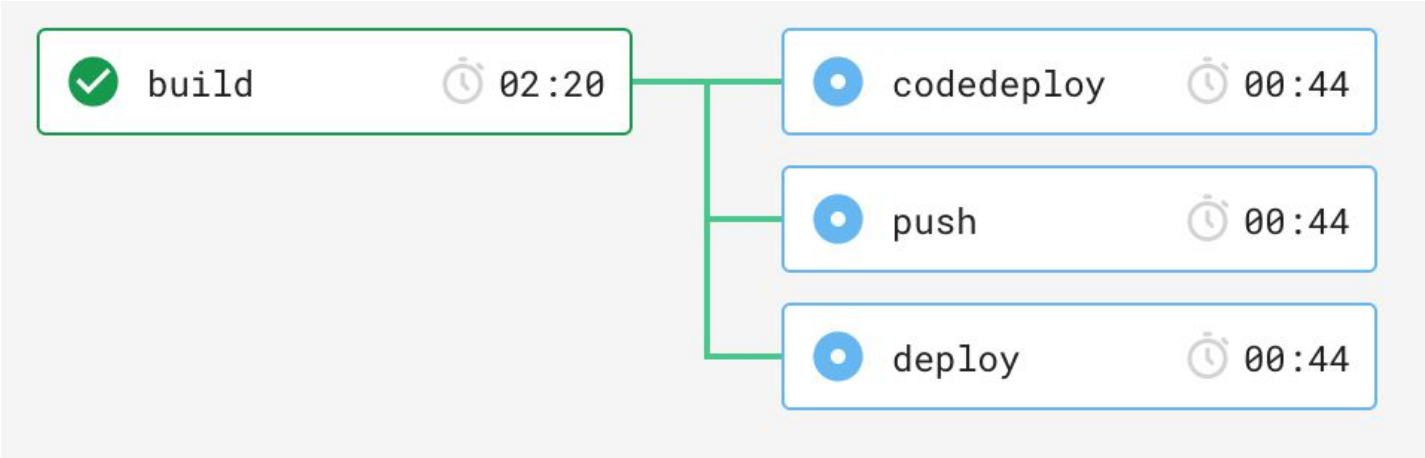
```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "my-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["eu-west-1a", "eu-west-1b", "eu-west-1c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  enable_nat_gateway = true
  enable_vpn_gateway = true

  tags = {
    Terraform = "true"
    Environment = "dev"
  }
}
```


6 MONTHS
LATER...

Dual deployment

Now we have old and new infrastructure side by side

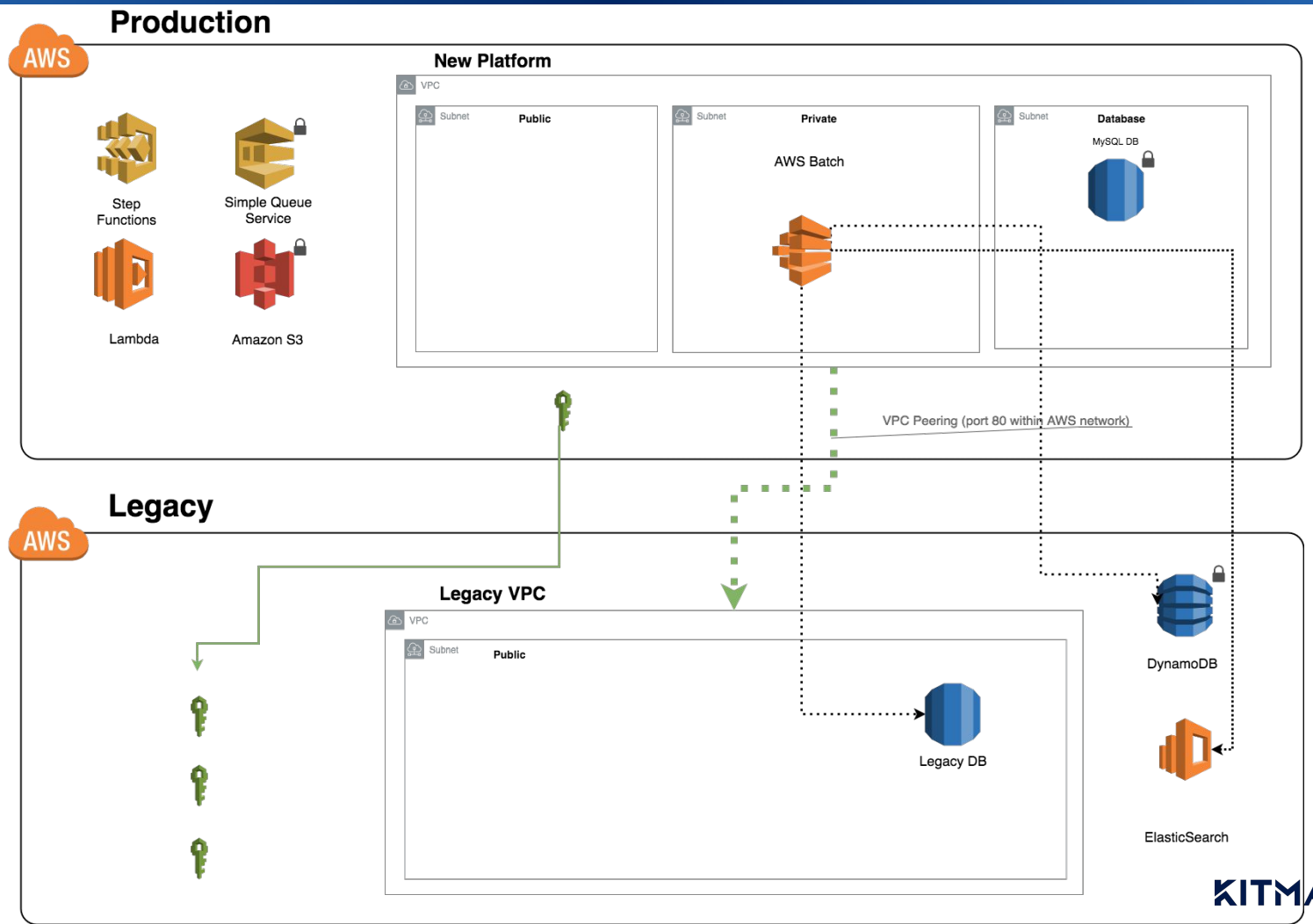
Every deployment goes to the new and old infrastructure.

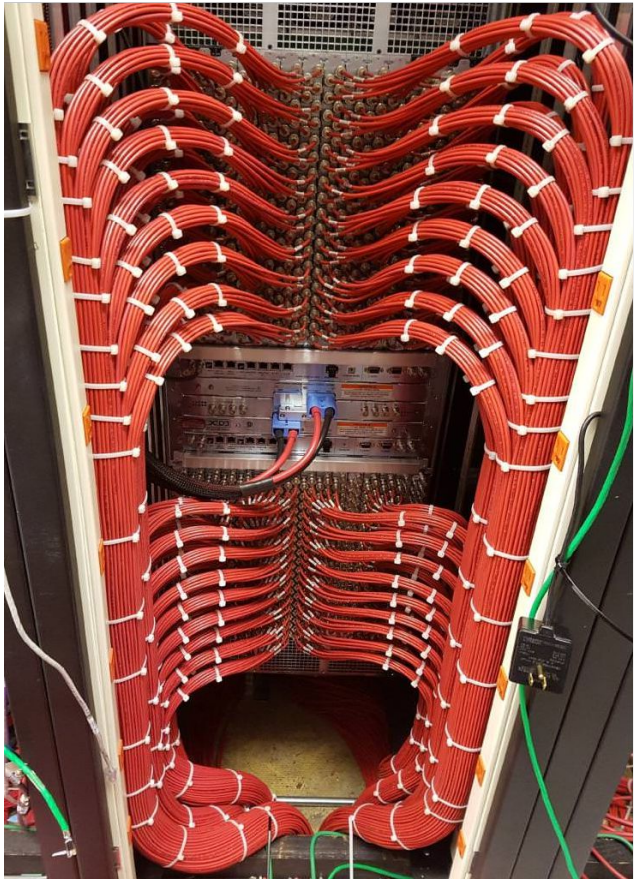


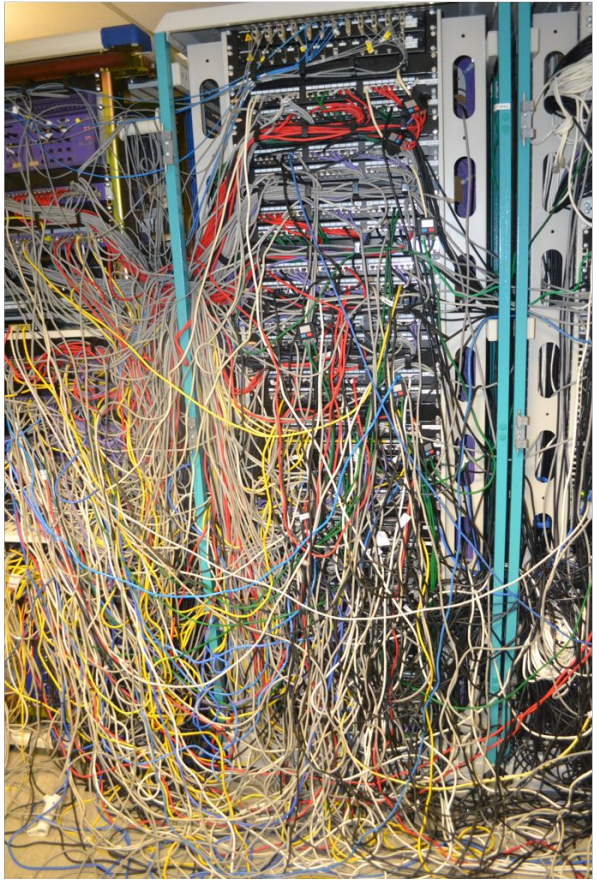
Cross account access + VPC Peering

Virtual equivalent of network cable between cabinets

No such thing as a greenfields project


Production
AWS
New Platform
VPC
Subnet
Public
Subnet
Private
Subnet
Database
MySQL DB
AWS Batch
Step Functions
Simple Queue Service
Lambda
Amazon S3
VPC Peering (port 80 within AWS network)
Legacy
AWS
Legacy VPC
VPC
Subnet
Public
Legacy DB
DynamoDB
ElasticSearch



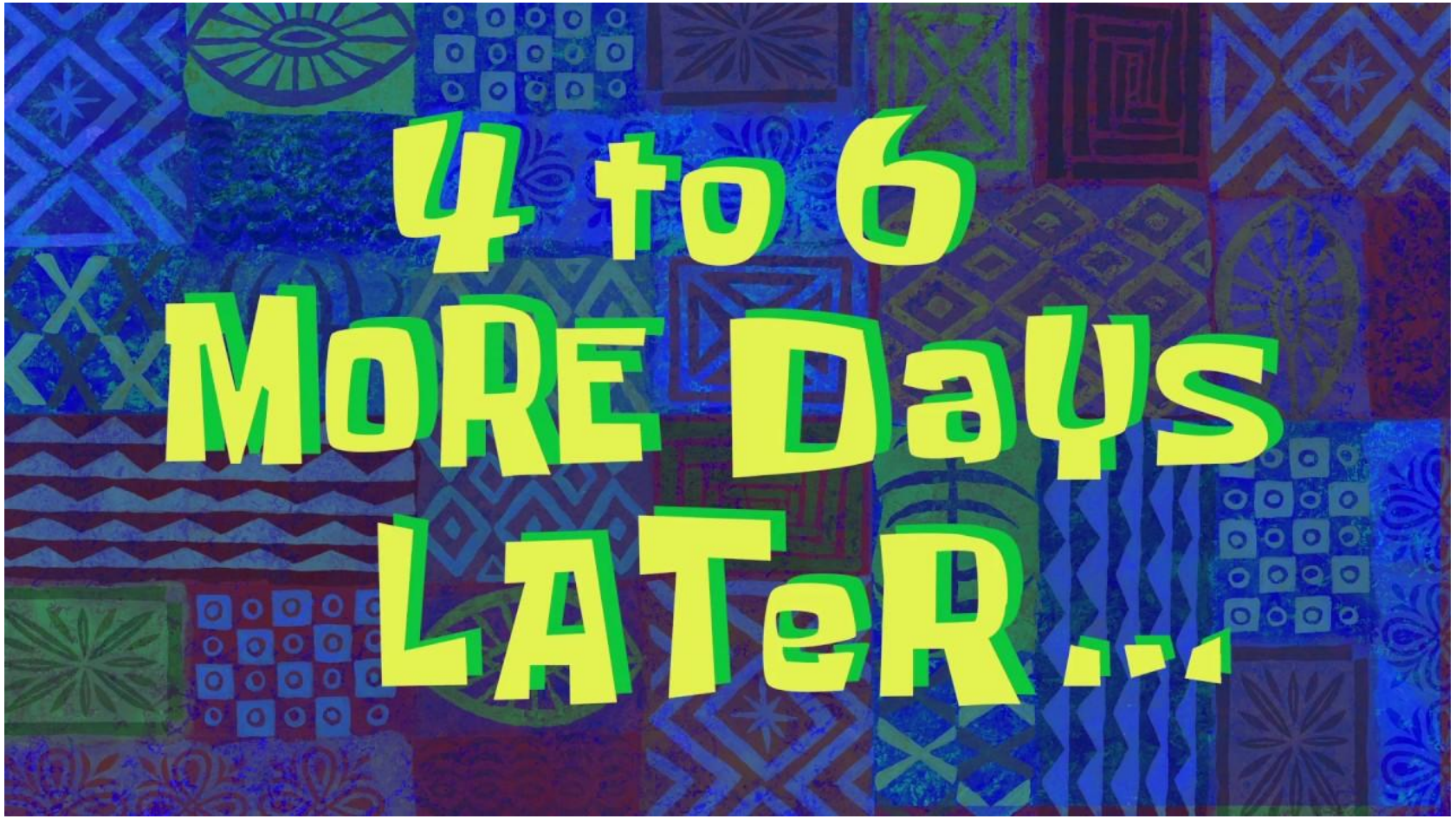


Import Legacy*

Import all legacy DNS (Route53)

Import all legacy S3 buckets (we need to setup cross account access)

Remove all legacy IAM Users (access via bastion only)


4 to 6
MORE DAYS
LATER...

DNS

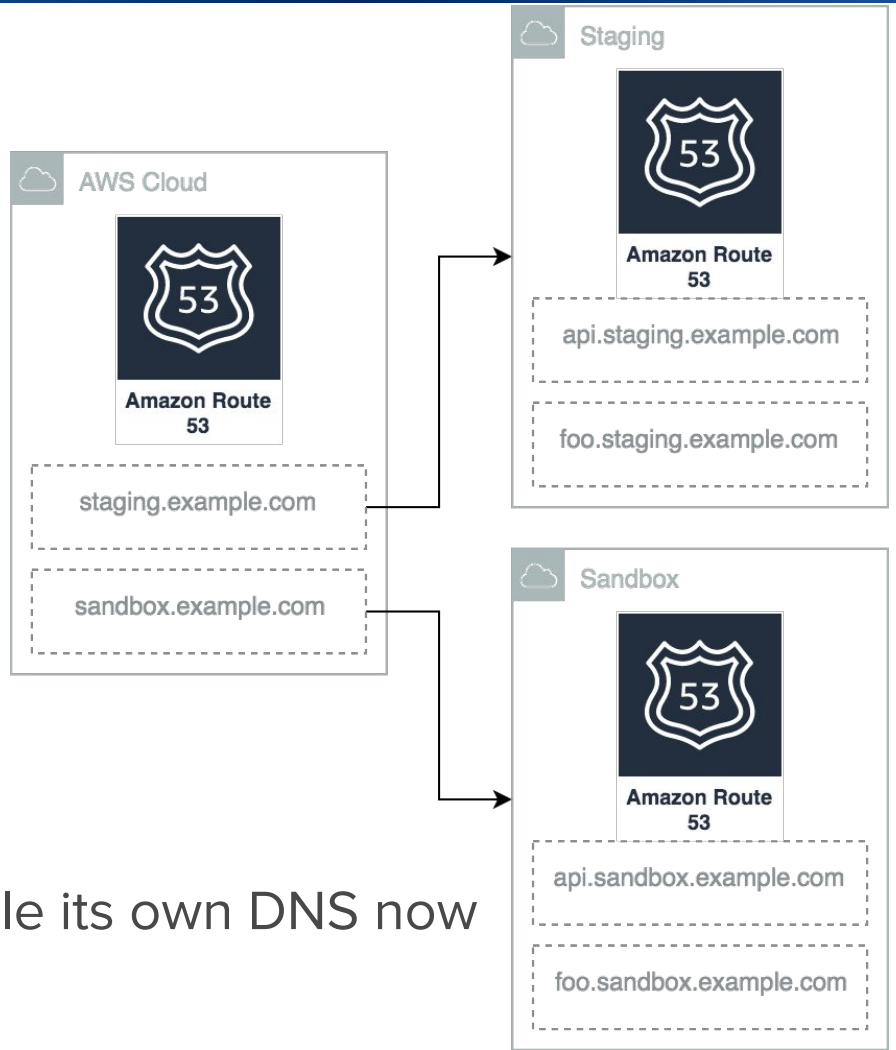
Purchased new domains

All domains and authoritative DNS lives in legacy (master) account

DNS should be per account!

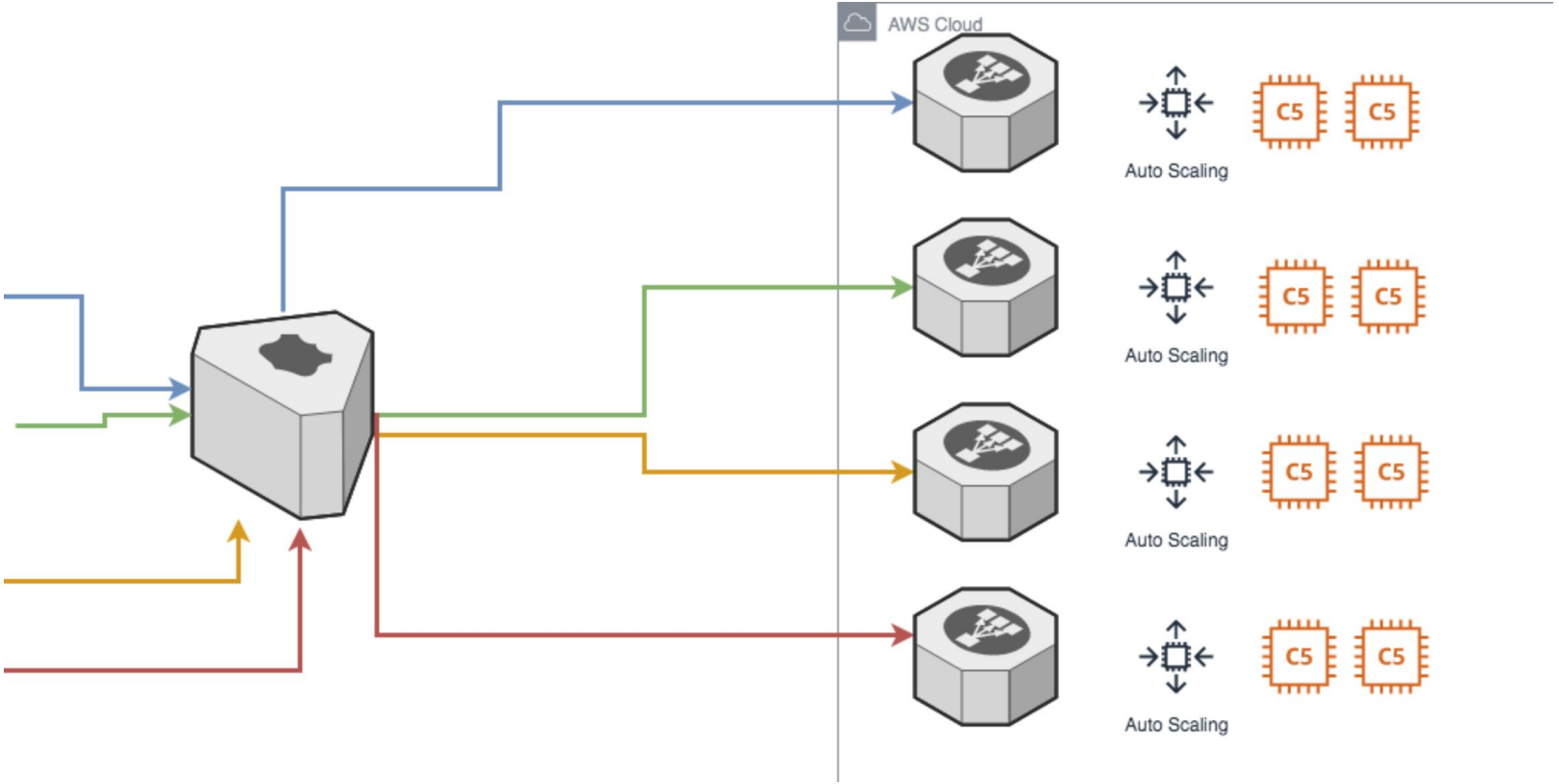
Delegate DNS zones into each environment

DNS

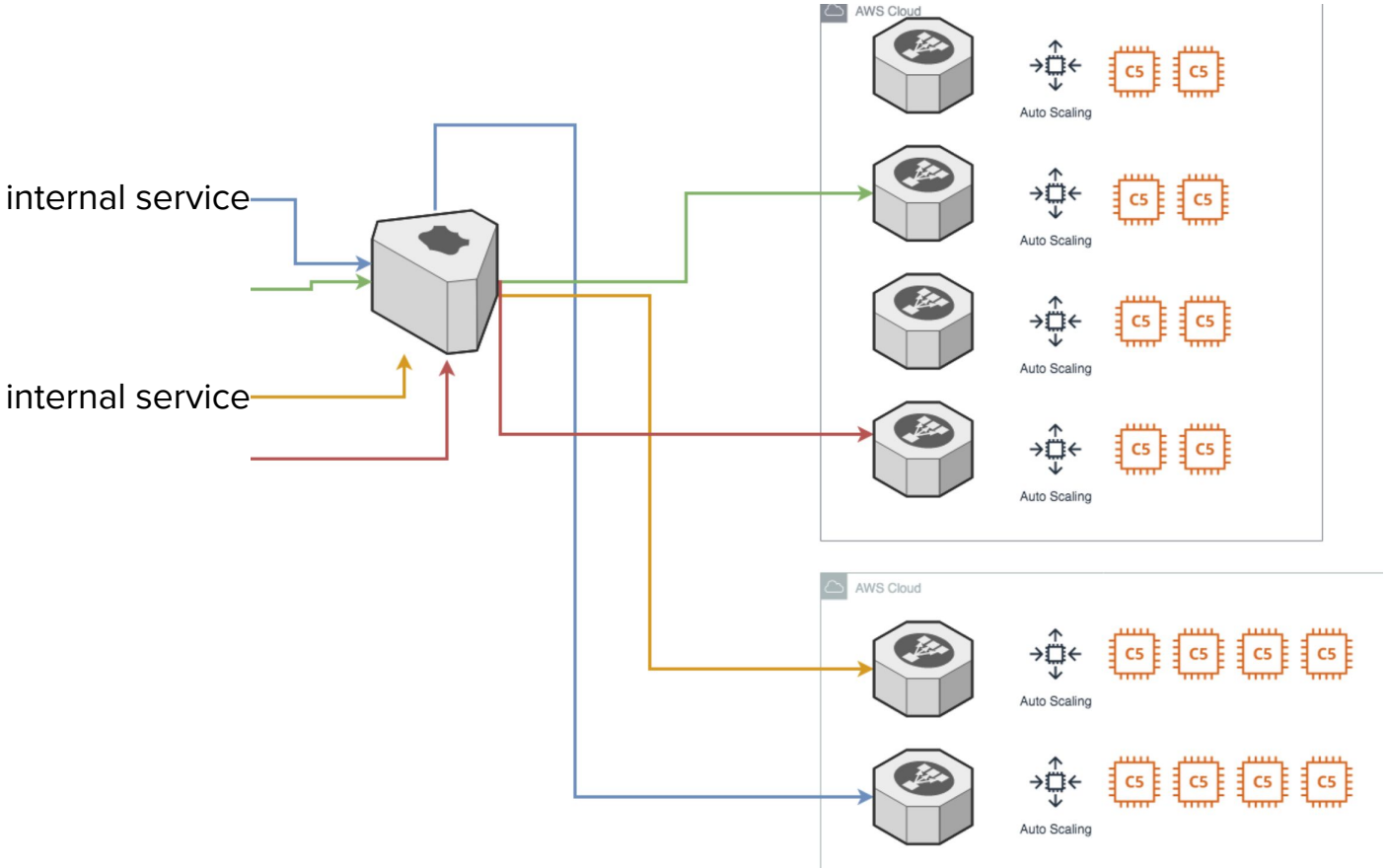


Each account can handle its own DNS now

The Switcheroo



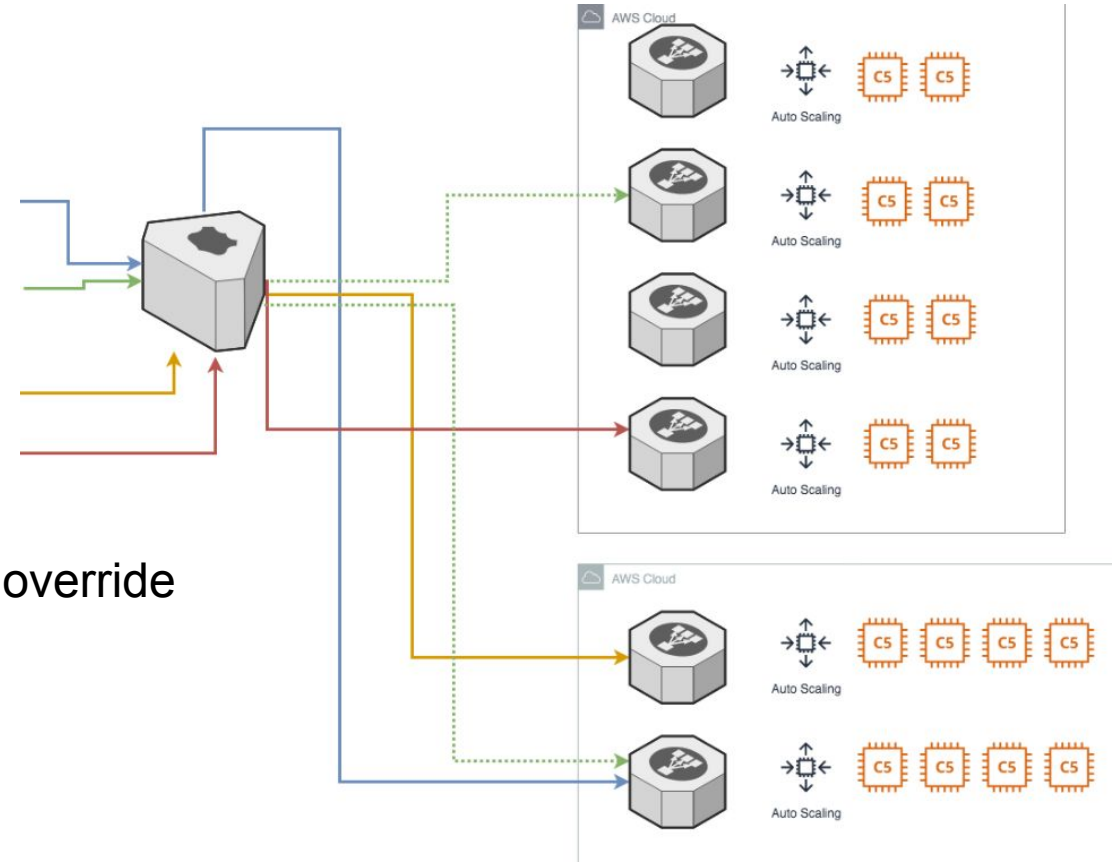
The Switcheroo



KITMAN LABS

The Switcheroo

team1.example.com
team2.example.com
foo.example.com
*.example.com



Explicit records override wildcards

Weighted Round Robin DNS

We utilized this to trickle traffic into new system

Weighted Round Robin DNS

We utilized this to trickle traffic into the new system

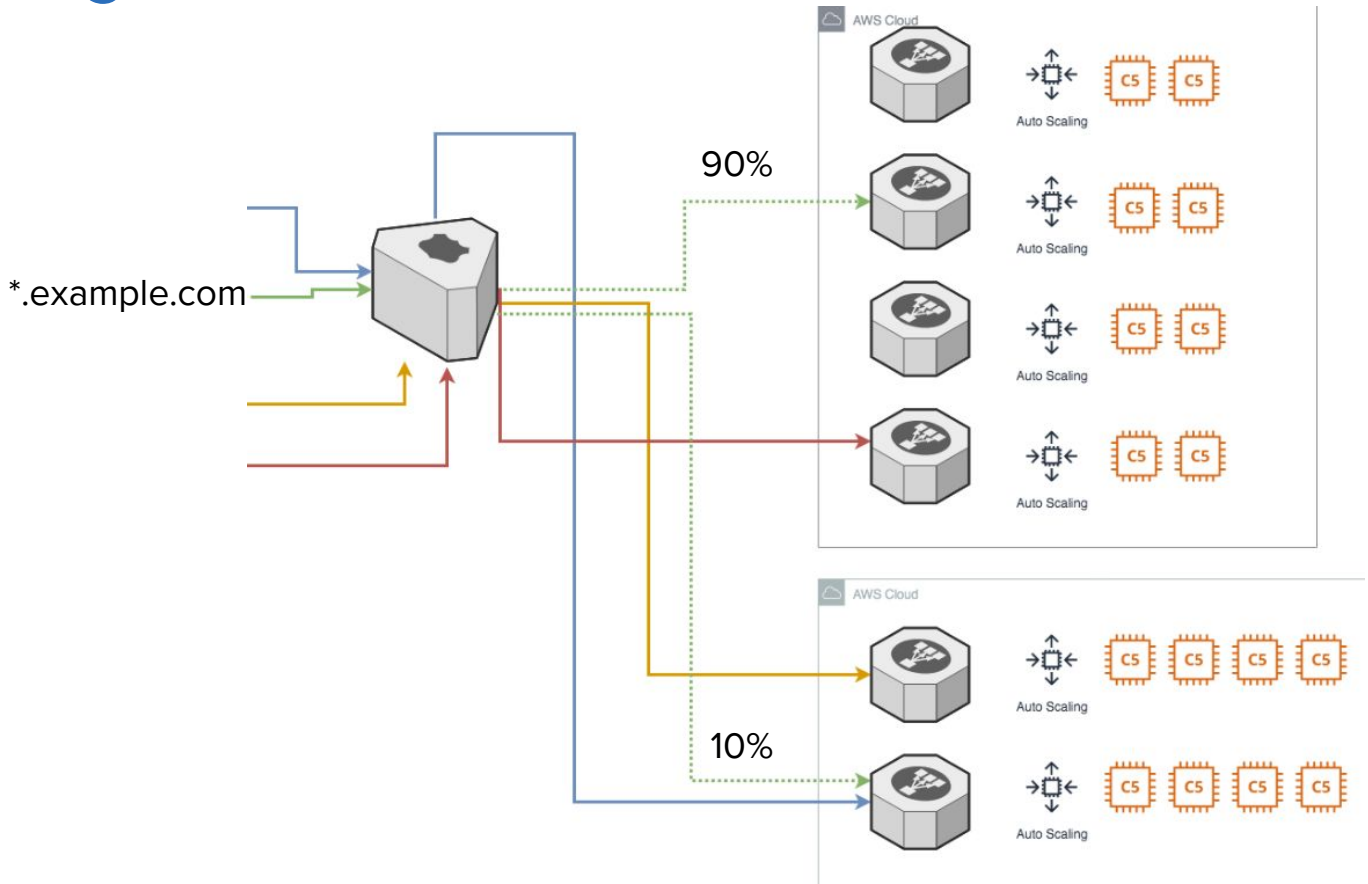
```
resource "aws_route53_record" "wrr" {
  count   = "${length(keys(var.records))}"
  zone_id = "${var.zone_id}"
  name    = "${var.name}"
  type    = "CNAME"
  ttl     = "${var.ttl}"
  records = ["${element(keys(var.records), count.index)}"]

  weighted_routing_policy {
    weight = "${element(values(var.records), count.index)}"
  }

  set_identifier = "${var.name}-${count.index}"
}
```

Weighted Round Robin DNS

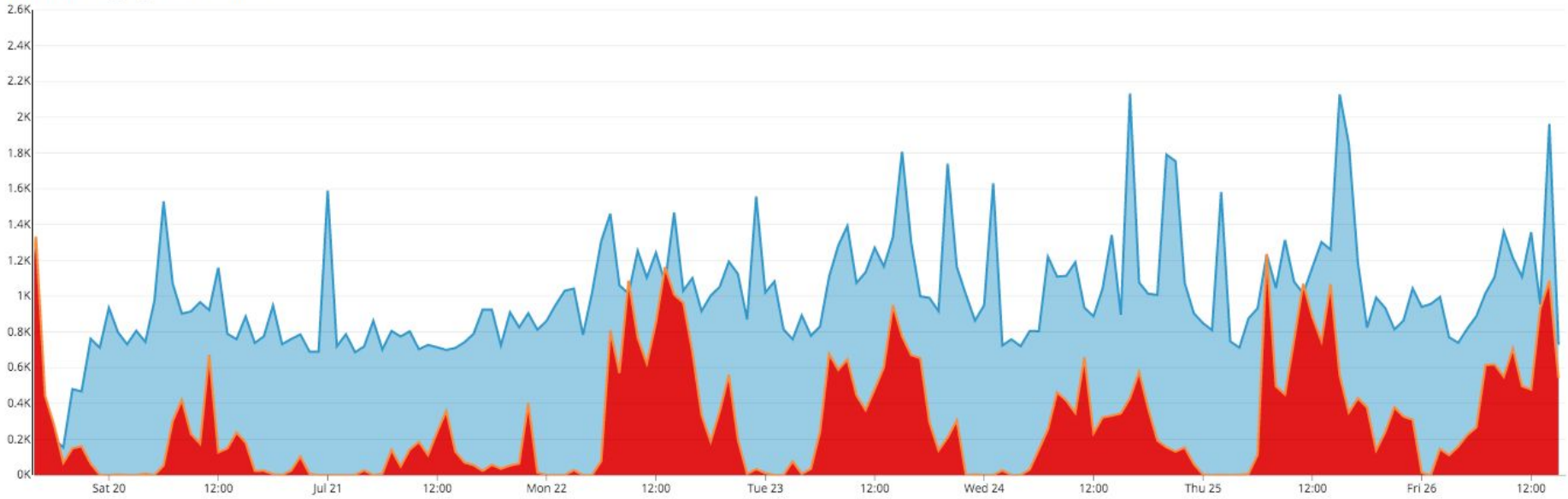
Give low weight to new system at first

```
module "wrr" {
  source  = "../route53_weighted_round_robin"
  name    = "service.example.com"
  zone_id = "${aws_route53_zone.your_zone.zone_id}"

  records = {
    shiny_new_endpoint.eu-west-1.elb.amazonaws.com = 10
    old_endpoint.eu-west-1.elb.amazonaws.com       = 90
  }
}
```

Weighted Round Robin DNS

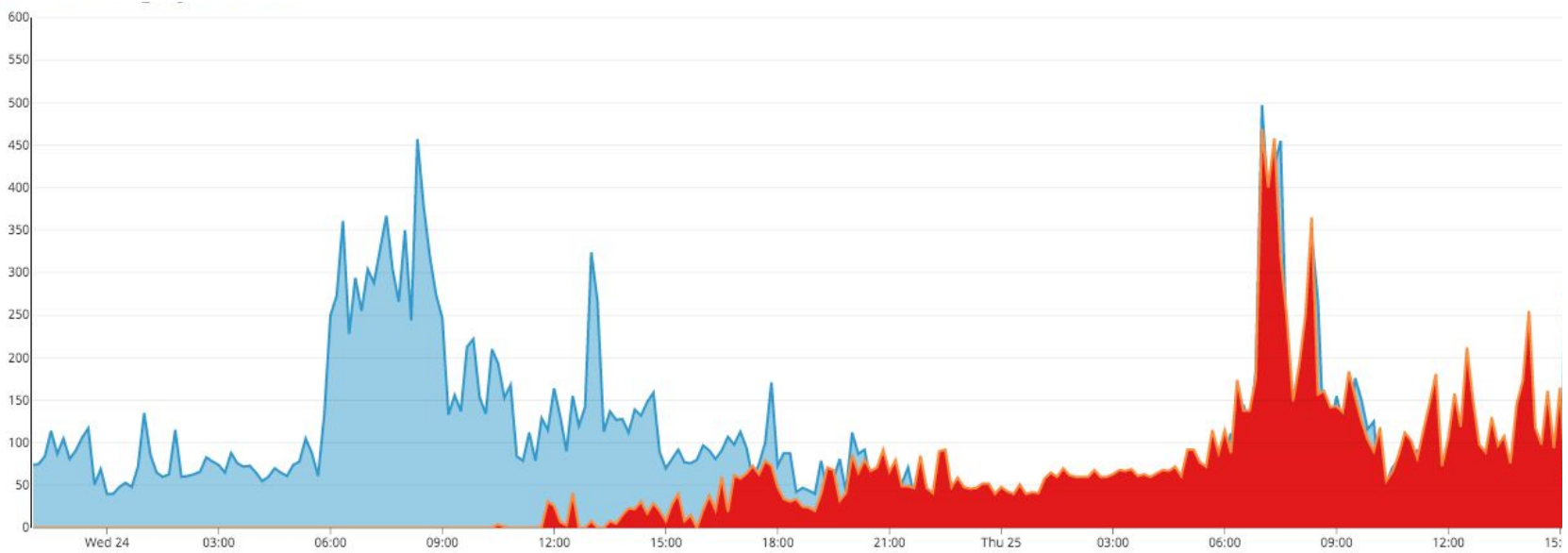


Weighted Round Robin DNS



KITMAN LABS

Increase Weight

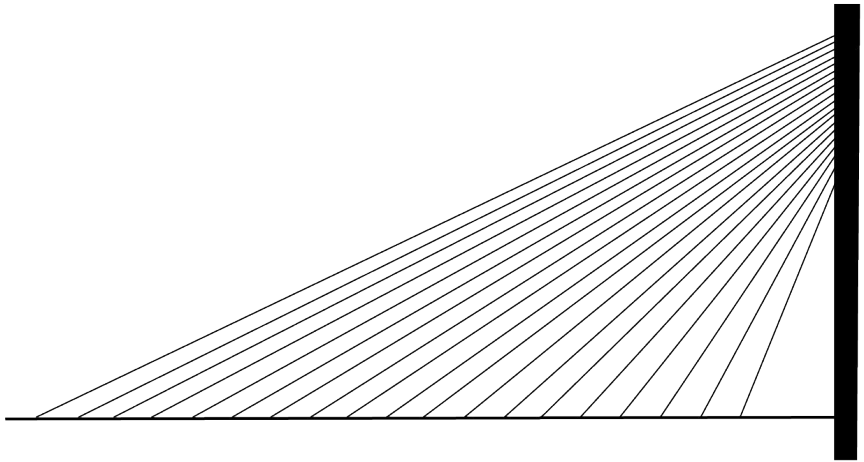


You must go all in

No out of bounds changes

If there is no terraform coverage you cannot use it*

News



aws South East User Group

7pm November 13th @ Boxworks
Second Wednesday Every Month