INVENCO

Terraform

amazon
web services

An Introductory Overview

https://github.com/beanaroo/aws_nz_meetup-terraform_intro

☑ Agenda

- IaC refresher
- Who's Hashicorp?
- What's Terraform?
- Why Terraform?
- How to Terraform
- Who's Invenco?
- How do we Terraform?
- Q & A

About Me

- CMS/LTSP/ERP/POS/COMPNOR

About Me

- CMS/LTSP/ERP/POS/COMPNOR 

About Me

- CMS/LTSP/ERP/POS/COMPNOR 
- Penguins are fun 

About Me

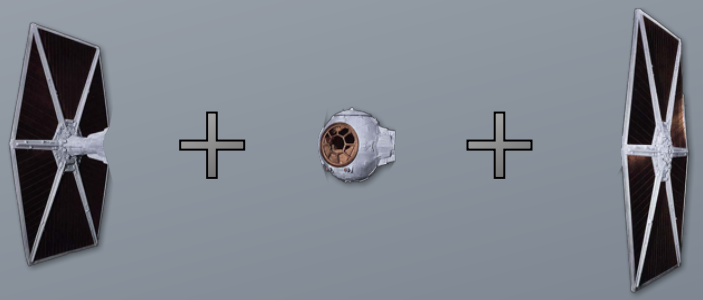
- CMS/LTSP/ERP/POS/COMPNOR 
- Penguins are fun 
- New to DevOps! 

About Me

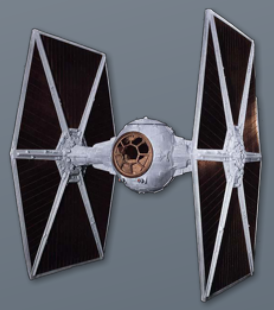
- CMS/LTSP/ERP/POS/COMPNOR 
- Penguins are fun 
- New to DevOps! 
- (Disclaimer) New to AWS 

</> Infrastructure as Code

- Imperitive
  - How to create my environment...



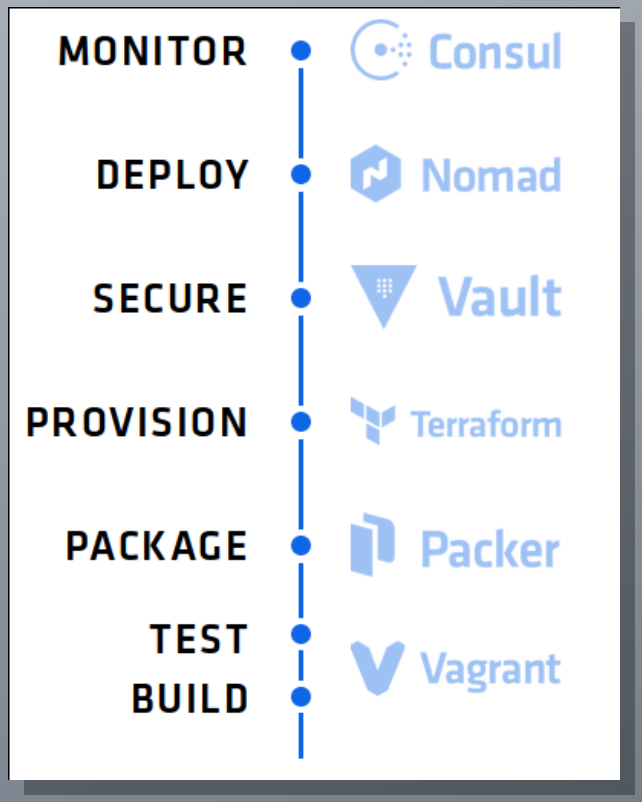
- Declaritive
  - What my environment should be...



</> Infrastructure as Code

- Imperitive
  - Close ties with Config Management
  - Familiar tool sets
  - Troubleshooting not always straight forward when provisioning is orchestrated
- Declaritive
  - CM may be wanted/needed anyway
  - DSLs for high-level definitions of desired states
  - Subtle variations are inconvenient to manage


HashiCorp



- Founded in 2010
- The “Atlassian” of DevOps
- Produce Open-Source Software Suite
- Enterprise versions with added functionality

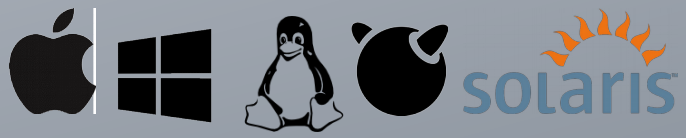
Terraform

- Provisions compute, storage, and networking resources across multiple services.
- “It is an open source tool that codifies APIs into declarative configuration files...”
- DSL
  - HCL (HashiCorp Configuration Language)
  - HIL (HashiCorp Interpolation Language)
- All written in Go!!!

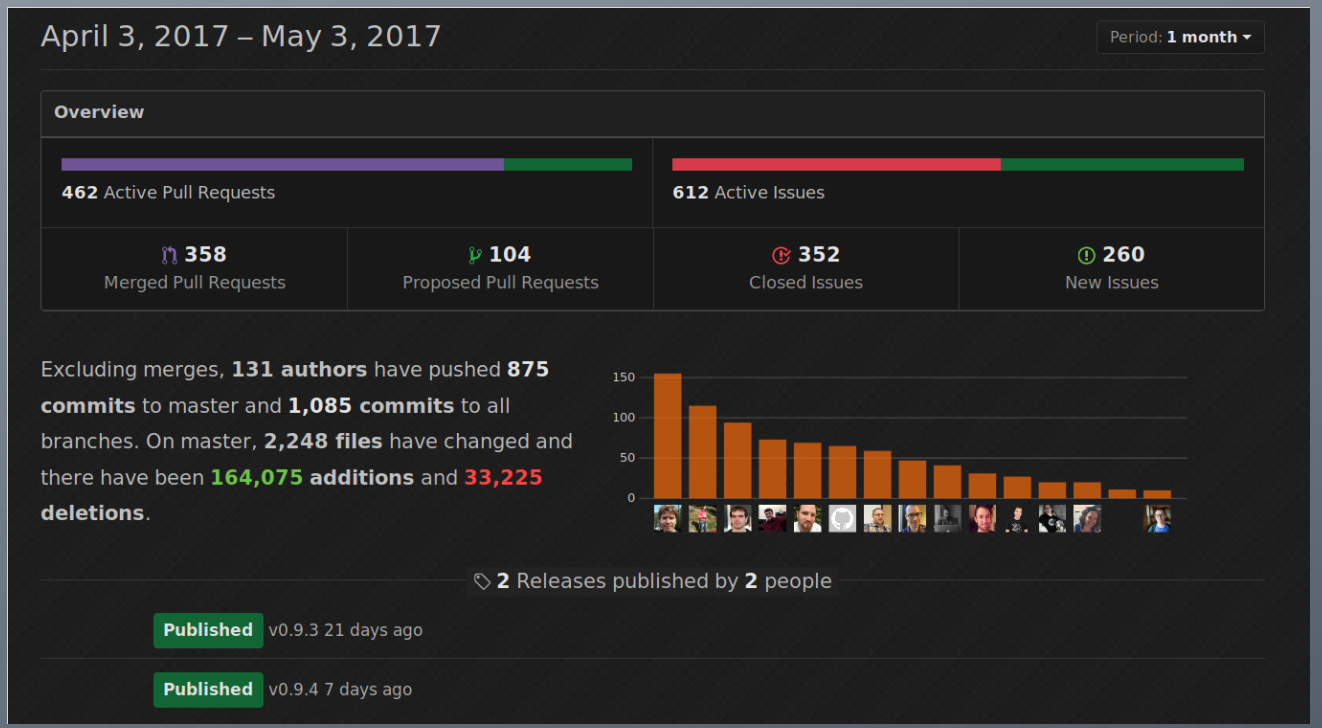
Terraform

- Alicloud
- Archive
- Arukas
- AWS
- Bitbucket
- CenturyLinkCloud
- Chef
- Circonus
- Cloudflare
- CloudStack
- Cobbler
- Consul
- Datadog
- DigitalOcean
- DNS
- DNSMadeEasy
- DNSimple
- Docker
- Dyn
- External
- Fastly
- GitHub
- Google Cloud
- Grafana
- Heroku
- Icinga2
- Ignition
- InfluxDB
- Kubernetes
- Librato
- Local
- Logentries
- Mailgun
- New Relic
- Nomad
- NS1
- Microsoft Azure
- Microsoft Azure (Legacy ASM)
- MySQL
- 1&1
- Oracle Public Cloud
- OpenStack
- OpsGenie
- Packet
- PagerDuty
- PostgreSQL
- PowerDNS
- ProfitBricks
- RabbitMQ
- Rancher
- Random
- Rundeck
- Scaleway
- SoftLayer
- StatusCake
- Spotinst
- Template
- Terraform
- Terraform Enterprise
- TLS
- Triton
- UltraDNS
- Vault
- VMware vCloud Director
- VMware vSphere

? Why Terraform...

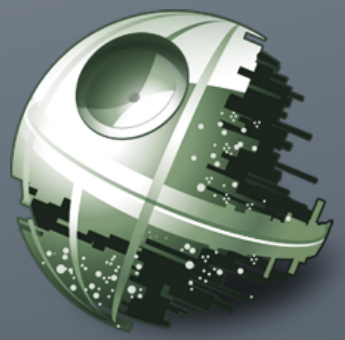
- Multi-platform 

- Single 30MB binary – No dependencies
- No configuration needed
- Agentless!
- Active and transparent development



🎓 How to Terraform...

- CLI
- Configuration (HCL)
- Interpolation (HIL)
- Providers
  - Resources
  - Data Sources
- Provisioners
- State
  - Backends
- Modules
- Plugins



🏛 `terraform.tfstate`

- JSON file mapping all managed resources and their metadata.
- Refreshes when needed or on demand.
- Not to be tampered with!!!
- May contain secrets
- Can not be used to roll back. (FOSS version)
- Stored locally by default.

Backends

- States should be managed centrally.
- Sensitive information needs a home.
- VCS/SCM works well but humans don’t.
- State-locking prevents simultaneous changes.

- artifactory
- azure
- consul
- etcd
- gcs
- http
- manta
- s3
- swift
- terraform enterprise

🏛 `terraform.tfstate`

- JSON file mapping all managed resources and their metadata.
- Refreshes when needed or on demand.
- Not to be tampered with!!!
- May contain secrets
- Can not be used to roll back. (FOSS version)
- Stored locally by default.

Configuration

- All .tf files are loaded from the working directory
- Variables need to be defined
- Values are read from .tfvars or interactively prompted for.

```
resource "aws_instance" "web" {
  ami                = "ami-4836a428"
  source_dest_check = false
  private_ip         = “10.20.0.10”
}
```

Configuration

- Variables can have defaults. Lack of which implies “required”.
- Variable types:
  - strings

```
variable "key" {
  type    = "string"
  default = "value"
}
```

```
variable "long_key" {
  type = "string"
  default = <<EOF
This is a long key.
Running over several lines.
EOF
}
```

Configuration

- Variables can have descriptions. (For generating docs)
  - lists

```
variable "users" {
  description = “Users with ssh access disabled”
  type    = "list"
  default = ["admin", "ubuntu"]
}
```

  - maps

```
variable "images" {
  type = "map"
  default = {
    us-east-1 = "image-1234"
    us-west-2 = "image-4567"
  }
}
```

Configuration

- Lists and Maps can be nested. BUT...
- Interpolation breaks. i.e. They aren't useful at the moment.

Outputs

- Attributes that can be exposed to other plans
- Useful when working with modules

```
output "address" {
  value = "${aws_instance.db.public_dns}"
}
```

Interpolation

- Wrapped in ${ }
- Used to access variables, perform math and functions

```
variable "ami" {
  description = "the AMI to use"
}

resource "aws_instance" "web" {
  ami               = "${var.ami}"
  source_dest_check = false
  private_ip        = “${var.ip}”
}
```

```
# Amazon Linux 2017.03 AMI
ami = “ami-4836a428”
ip  = “10.20.0.10”
```

Interpolation

- Functions include:
  - string/list/timestamp formatting
  - base64 encoding/decoding
  - uuid generation
  - md5/sha1/sha256 hashing
  - CIDR formatting
  - map lookups
- Provides basic logic through ternary operation

  `CONDITION ? TRUEVAL : FALSEVAL`

```
resource "aws_instance" "web" {
  subnet = "${var.env == "production" ? var.prod_subnet : var.dev_subnet}"
}
```

.

Providers

- Provide resources to create and manage
  - AWS
    - aws_vpc
    - aws_instance
    - aws_s3_bucket
    - aws_route53_record
- Provide data sources to reference info.
  - AWS
    - aws_ami (search for id)
    - aws_hosted_zone_id (useful for R53)
    - aws_iam_policy_document (generate policies using HCL)
  - terraform
    - reference resources in other environments

Resources

- Infrastructure building blocks.
  - Have specific arguments
  - Provide attributes to reference elsewhere
  - Share common meta-parameters:
    - count – No. of resource instances to create
    - depends_on – Explicit dependencies
    - provider – (i.e. different AWS account)
    - lifecycle
      - create_before_destroy (replacement provisioned first)
      - prevent_destroy (protects resource)
      - ignore_changes (list of attributes)

⚙ Provisioners

- Perform local and remote command execution
  - chef
  - file
  - local-exec
  - remote-exec
- Useful for bootstrapping

Modules

- Reusable plans.
- Outputs need to be propegated.

```
imperial_cloud_services/
├── environments
│   ├── dev
│   ├── prod
│   ├── qa
│   └── uat
└── modules
    ├── our_cloudfront_setup
    ├── our_custom_ec2_server
    └── our_vpc_layout
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

> terraform --help

```
Common commands:
    apply              Builds or changes infrastructure
    console            Interactive console for Terraform interpolations
    destroy            Destroy Terraform-managed infrastructure
    env                Environment management
    fmt                Rewrites config files to canonical format
    get                Download and install modules for the configuration
    graph              Create a visual graph of Terraform resources
    import             Import existing infrastructure into Terraform
    init               Initialize a new or existing Terraform configuration
    output             Read an output from a state file
    plan               Generate and show an execution plan
    push               Upload this Terraform module to Atlas to run
    refresh            Update local state file against real resources
    show               Inspect Terraform state or plan
    taint              Manually mark a resource for recreation
    untaint            Manually unmark a resource as tainted
    validate           Validates the Terraform files
    version            Prints the Terraform version

All other commands:
    debug              Debug output management (experimental)
    force-unlock       Manually unlock the terraform state
    state              Advanced state management
```

Terraform in action...

Let’s create a VPC

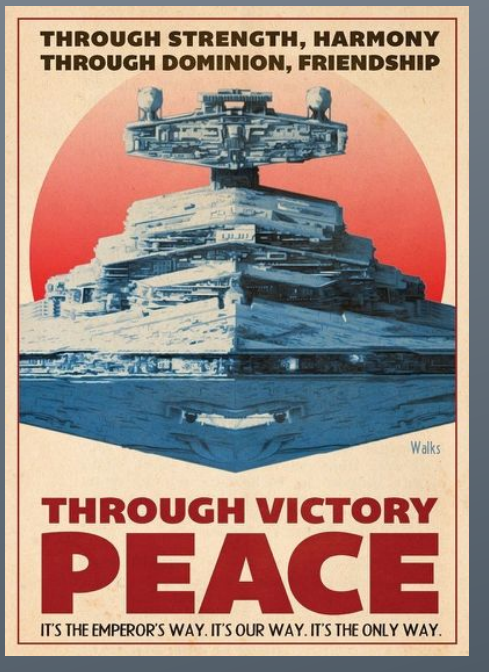
	control-vpc-uw2	vpc-f1efd096	available	10.0.0.0/16

Needs some internet

control-c-public-subnet	subnet-e6dc21bd	available	vpc-f1efd096 | control-vpc-uw2	10.0.160.0/19	us-west-2c
control-a-public-subnet	subnet-1c84cf7b	available	vpc-f1efd096 | control-vpc-uw2	10.0.96.0/19	us-west-2a
control-b-public-subnet	subnet-b8b5d7f1	available	vpc-f1efd096 | control-vpc-uw2	10.0.128.0/19	us-west-2b

control-igw	igw-59be4...	attached	vpc-f1efd096 | control-vpc-uw2

nat-03ffb4d0441e718e5	Available	35.165.12.156	10.0.119.171
nat-02f142e047e532373	Available	52.33.74.168	10.0.144.107
nat-0e922dd3d549315a2	Available	34.223.211.208	10.0.165.184



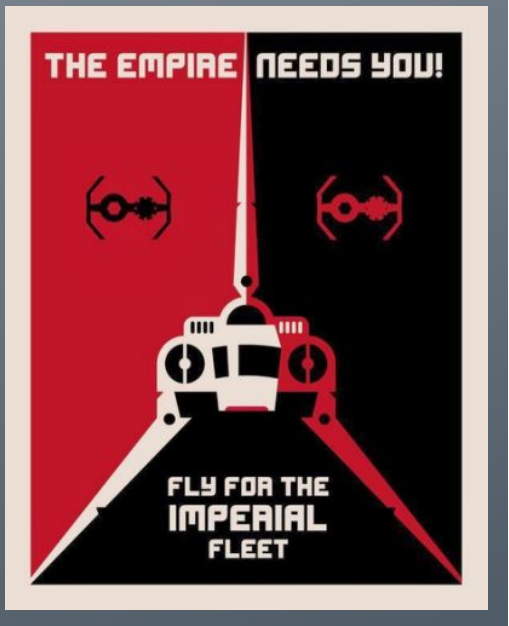
Needs some private subnets

☑	control-a-private-subnet	subnet-229cd745	available	vpc-f1efd096 | control-vpc-uw2	10.0.0.0/19	us-west-2a
☑	control-b-private-subnet	subnet-b2bcdefb	available	vpc-f1efd096 | control-vpc-uw2	10.0.32.0/19	us-west-2b
☑	control-c-private-subnet	subnet-c9da2792	available	vpc-f1efd096 | control-vpc-uw2	10.0.64.0/19	us-west-2c
☐	control-a-public-subnet	subnet-1c84cf7b	available	vpc-f1efd096 | control-vpc-uw2	10.0.96.0/19	us-west-2a
☐	control-b-public-subnet	subnet-b8b5d7f1	available	vpc-f1efd096 | control-vpc-uw2	10.0.128.0/19	us-west-2b
☐	control-c-public-subnet	subnet-e6dc21bd	available	vpc-f1efd096 | control-vpc-uw2	10.0.160.0/19	us-west-2c

☑	control-us-west-2-public-rt	rtb-92e24af4	3 Subnets	No
☑	control-c-private-rt	rtb-95e24af3	1 Subnet	No
☑	control-a-private-rt	rtb-93e24af5	1 Subnet	No
☑	control-b-private-rt	rtb-56e24a30	1 Subnet	No
☐		rtb-2f02ab49	0 Subnets	Yes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-59be4c3e	Active	No

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	nat-03ffb4d0441e718e5	Active	No



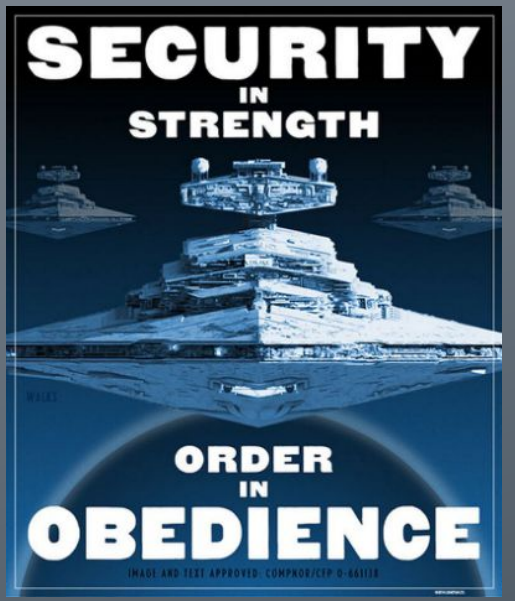
Let’s make it a module

Add some servers...

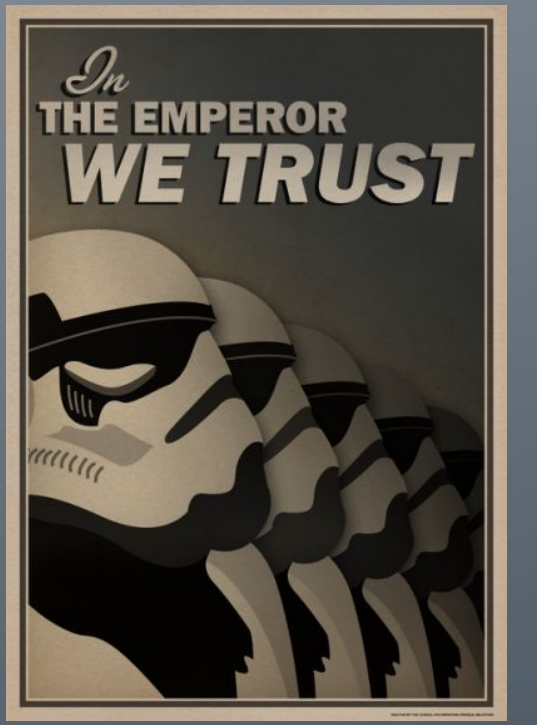
control-server-a-ec2	i-0d4e630695e7...	t2.micro	us-west-2a	running
control-server-b-ec2	i-0238de45b384...	t2.micro	us-west-2b	running
control-server-c-ec2	i-04fca38e5704e...	t2.micro	us-west-2c	running

control-server-elb	control-server-elb-14814...

sg-3e0e9d45	restrict_ec2_inbound	vpc-f1efd096
sg-7e089b05	restrict_elb_inbound	vpc-f1efd096



⚙ Spin up a whole new env!



INVENCO®

Invenco Cloud Services (ICS)

Remote Software Updates
Remote Financial Key Injection (RKI)
Asset Management and Diagnostics
Usage and Uptime Metrics
Prompt Management (Development)
Media Management (Development)
Time of Day Content (Development)

C1-100

Site Controller
Forecourt control application

Invenco EPS

Electronic Payment Server (EPS)

Invenco Professional Services

Custom User Experience Development
Solutions Architecting
Deployment and Support

G7-100

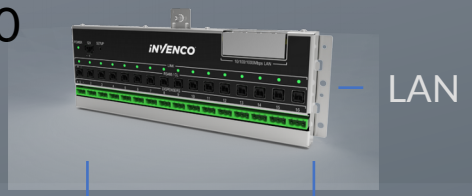
Modular Payment Terminal

Invenco Link L1-100

2-Port Network Link

Invenco Link L3-100

16-Port Network Link



LAN

G6-200

All-in-one Payment Terminal

INVENCO

CLOUD SERVICES PRODUCT LINE

Media Management

Credit Processor

Remote Management / Analytics

Advertising Provider

Loyalty Enabler

Mobile Wallet Enabler

EPS PRODUCT LINE

HOST INTERFACE

POS API

TERMINAL CONTROL

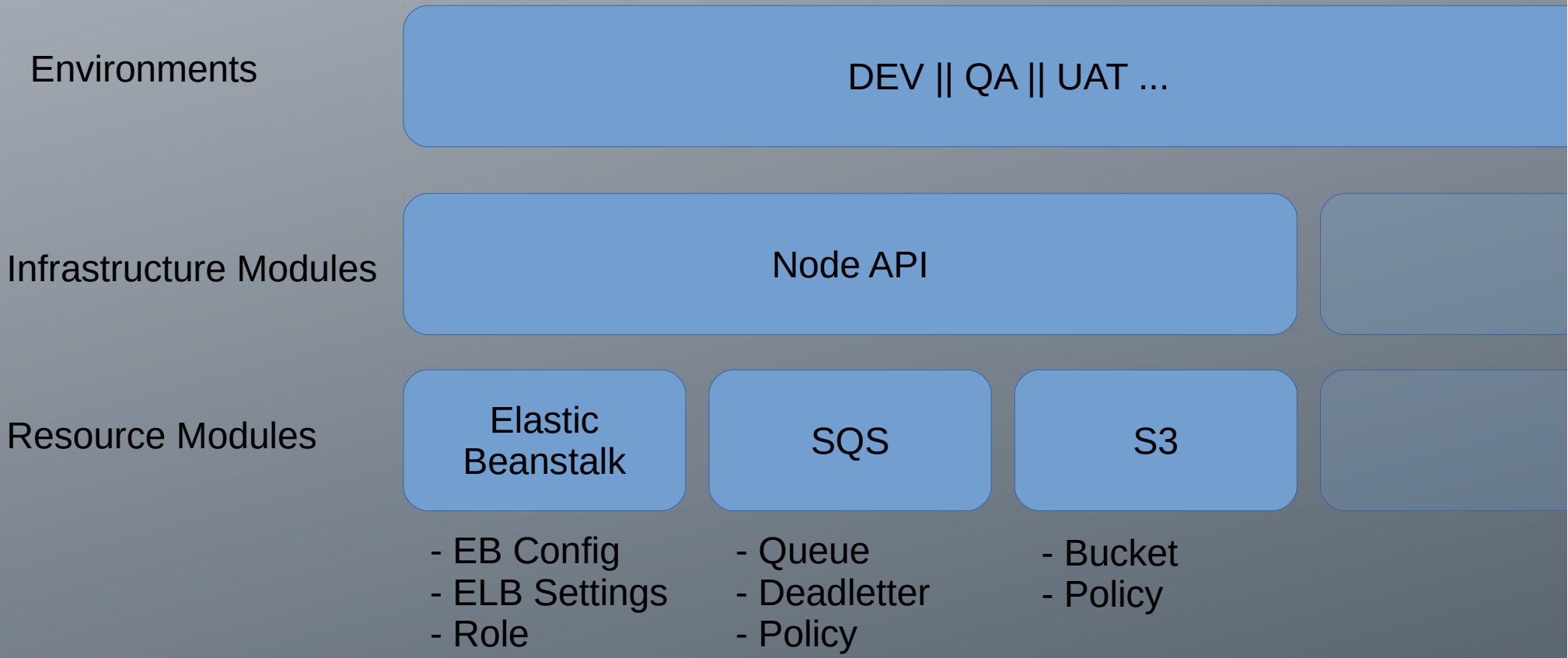
TRANSACTION REPORTING

TERMINAL PRODUCT LINE

DevOps Structure

```
git-repo
├── config
│   ├── nz-south-5
│   └── us-north-7
│       ├── dev
│       ├── qa
│       └── uat
│           └── config.yaml
├── infrastructure
│   ├── environments
│   │   └── us-north-7
│   │       ├── dev
│   │       ├── qa
│   │       └── uat
│   │           └── main.tf
│   └── modules
│       ├── infrastructure_modules
│       │   ├── api-nodejs
│       │   ├── cloudfront_distro
│       │   ├── queue-worker
│       │   └── rds_postgres
│       └── resource_modules
│           ├── aws_beanstalk_server
│           ├── aws_beanstalk_worker
│           ├── aws_rds
│           ├── aws_route53
│           ├── aws_sqs
│           └── aws_vpc
└── scripts
    ├── app_deploy.py
    └── terraform_wrapper.py
```

Terraform Structure



INVENCO

Cloud Services

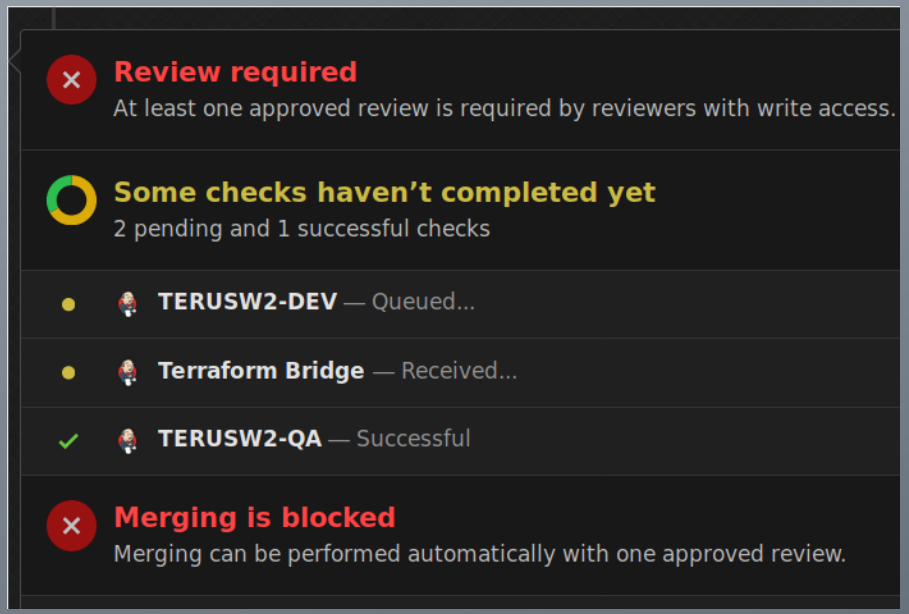
- All application and infrastrucure configuration templated using Jinja2
- app_deploy.py
  - extracts build artifacts
  - renders configuration
  - produces release artifacts
  - produces deploy wrappers for AWS components

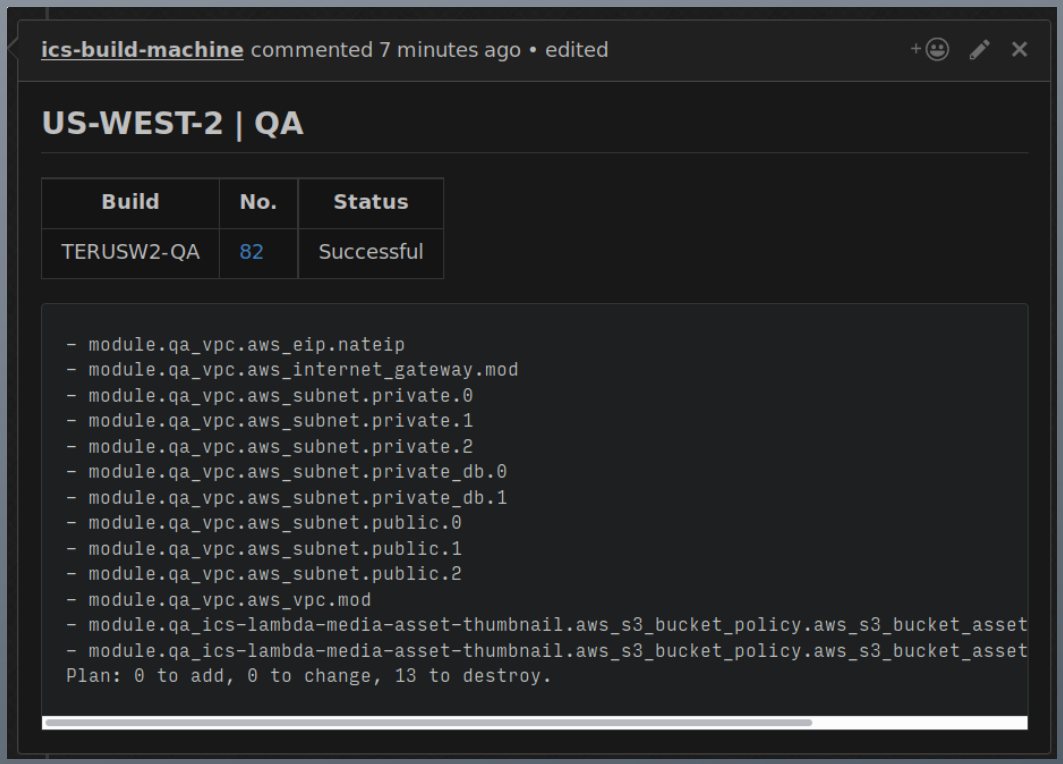
Cloud Services

- terraform_wrapper.py
  - renders environment terraform.tfvars.j2
  - places environment plan in temp working dir
  - performs terraform init, plan/apply
  - produces a summarized change overview

Cloud Services

- PR initiates a WebHook
- Buid-Server runs terraform_wrapper.py on applicable environments
- Change summary reported back to PR for approval
- Mass-destroys are marked as failure to prevent merging.





Cloud Services

- In the pipeline:
  - Tagged Modules

```
module "aws_lambda_bastion_server" {
  source = "git::http://our-git-service/foo/module.bar.git?ref=TAG_NAME"
}
```

  - Referencing application configuration from states...
    - It is JSON after all!

🎓 Lessons Learnt

- AWS Lambda
  - Provision == Deploy
  - Reference common zip in both processes
- Beanstalk
  - Three step deploy (can be approached better)
    - Initial terraform with health checks blank
    - Deploy Apps
    - Terraform again with health checks updated
  - Unapplicable settings are ignored by API
    - i.e. platform specific settings, worker vs. web server settings
    - This results in changes detected EVERY plan

INVENCO

🎓 Lessons Learnt

- Beanstalk
  - Plenty of settings....
  - Worked around by abusing maps.
    - Maps have to be flat!!!
    - Make them all strings.
    - Cast to the required type in the resource module level
    - Requires casting back and forth when referencing outputs.
    - Nested lists/maps imminent.

INVENCO

🎓 Lessons Learnt

- Don’t deploy to Prod on Fridays…