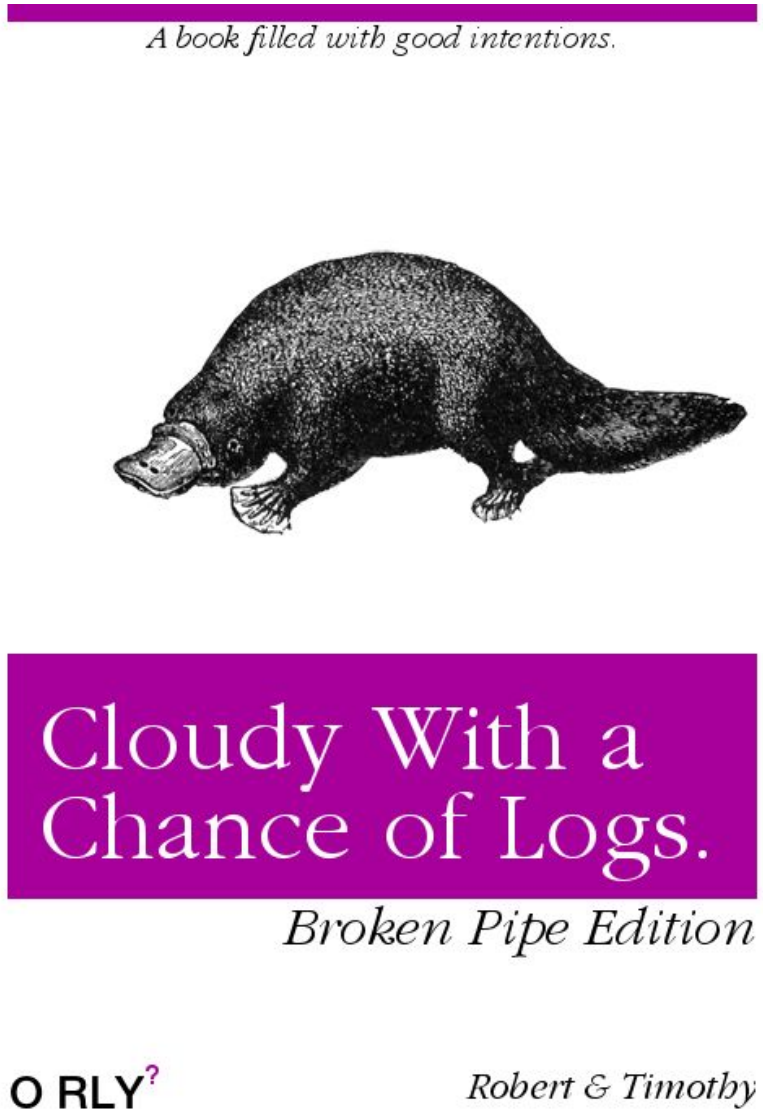
The Splunk Derp Gun

Automating Splunk deployments in the Cloud with Ansible and Terraform

Robert Johansson – Timothy Mahoney
Securelink Sweden AB

About us:

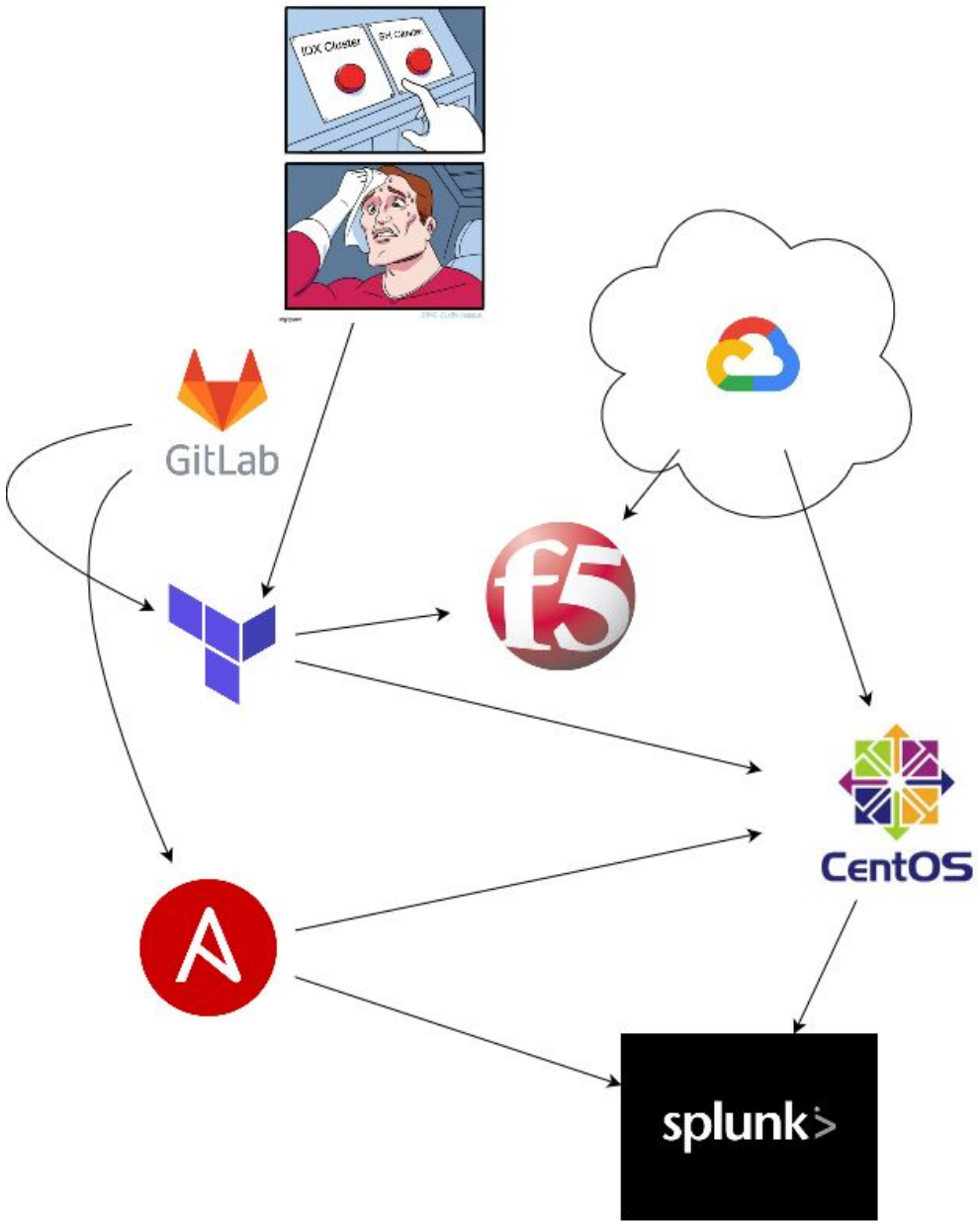
- SecureLink Malmö
- Managed SIEM Team
- Book release 2020/04/01



A little background:

- Build a Splunk environment in GCP
- Be lazy, automate as much as possible
- Create “single button” deployment.

The plan:



What is a ”Derp Gun”

- In Online Gaming - A gun that causes a lot of damage with one shot *urbandictionary.com
- In this context a Derp Gun is a “one shot” Splunk deployment.

Provisioning with a Derp Gun in GCP

- Streamline Provisioning of New Resources in GCP
- Automate complicated setups such as Load Balancers
- Rapidly provision hosts, disk, network infrastructure

Configuration with a Derp Gun in GCP

- Automate configuration of Splunk hosts by role.
- Maintain configurations as repos in Git.
- Allow for new Splunk hosts to be configured as quickly as they can be provisioned.
- Maintain consistent configurations across the entire deployment.

The Tools of the Trade

- GCP Console
- Ansible AWX
- Terraform
- Gitlab
- Vault

- Obviously: Cloud Infrastructure
- IAM
- Autoscaling of Instance Groups

Google Cloud

- Create infrastructure as code
- Providers and Resources
- Available by default in GCP
- Works with Azure, AWS, AliCloud, etc
- Idempotent
- Can be used with anything that has an API: SignalFX, Dominos Pizza, etc

Terraform

- Perform Automated Tasks
- Configuration of Hosts
- Deployment of Apps, Indexes

Ansible

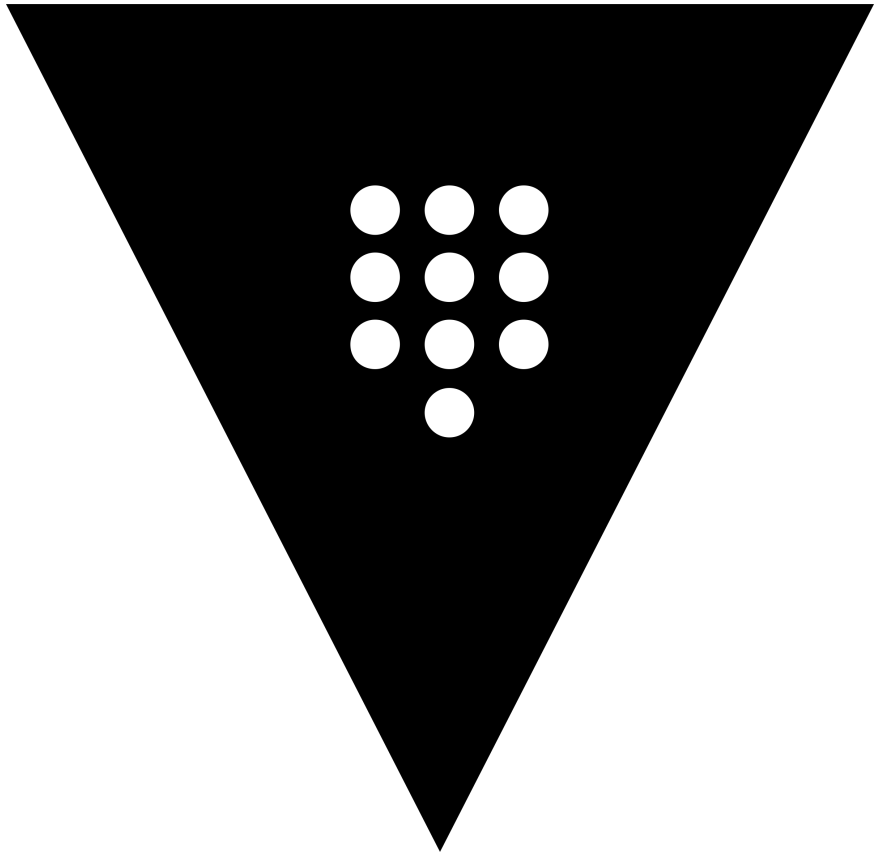
- Store configurations used by Terraform and Ansible
- Revision Control
- Collaboration
- CI/CD
- More Dev/Ops buzzwords..

Gitlab, Github, etc

GitLab

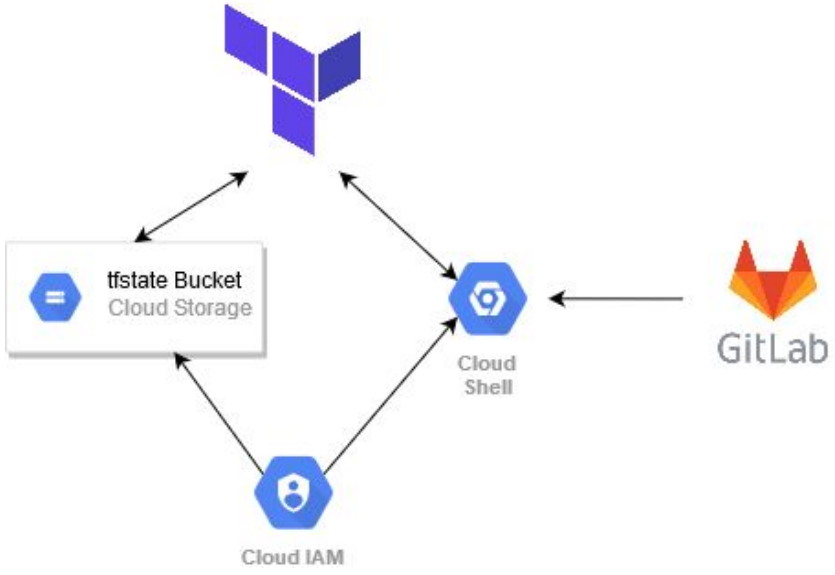
Vault

- Stores secrets
- Works with Terraform, Ansible
- Token based access
- Intgrates with Cloud KMS, IAM
- Can act as a Certificate Authority
- Can perform encryption services



Setting up a Terraform Environment

- Use bucket storage to allow tfstate to be shared
- Enable versioning on the bucket to retain backups of the state file in case of corruption
- Use a cloud ACL to limit access to the state file
- Bucket is accessible from the cloud shell
- Bucket provides state locking



Simple Terraform Example: State Bucket

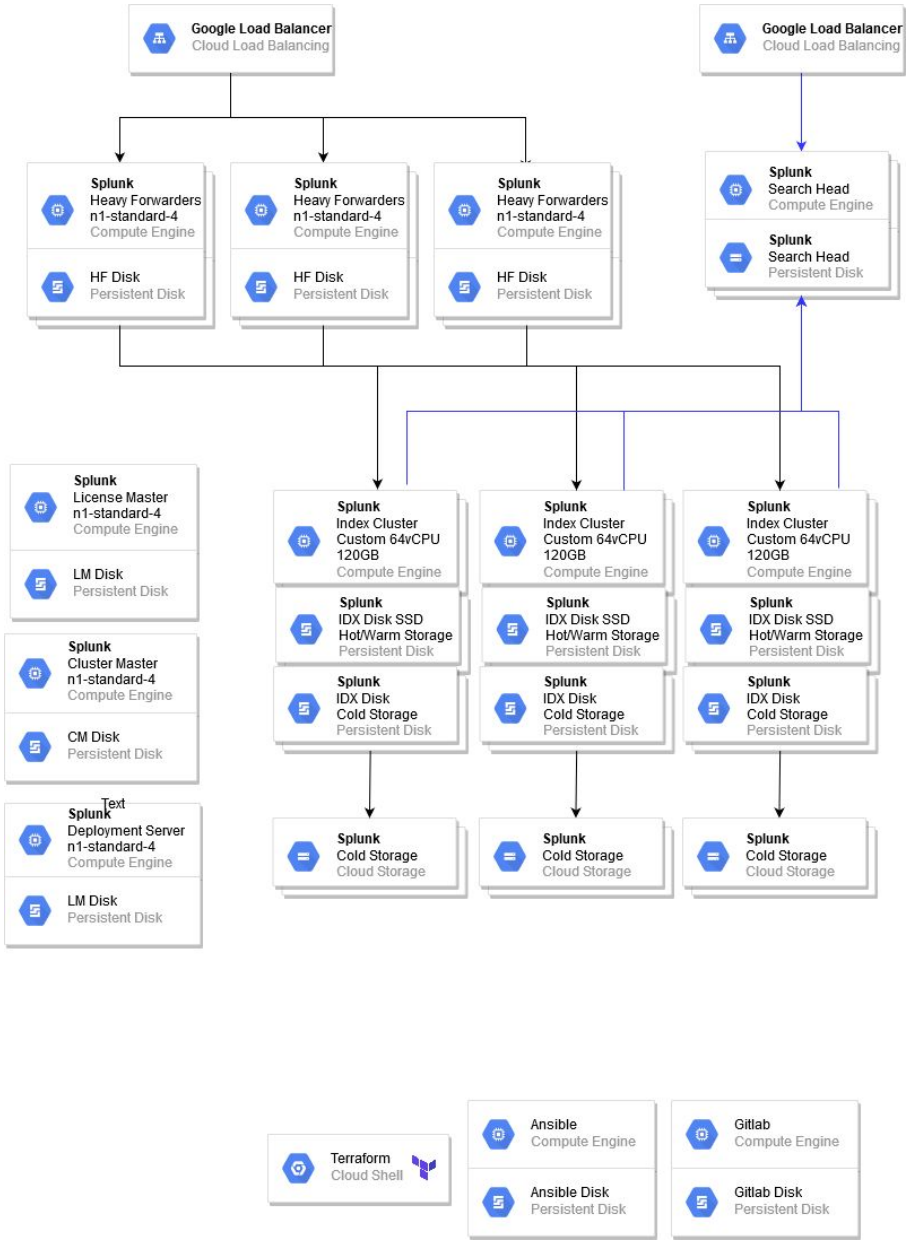
```
terraform {
  backend "gcs" {
    bucket = "ikea-splunk-prod-tf-state-bucket"
    prefix = "terraform/tfstate"
  }
}
```

Start with a sturdy bucket.

- Bucket to store tfstate file
- Keep last 50 versions of the file
- ACL to protect access

```
// GCP Provider Config

provider "google" {
  // Credentials not needed as this tf is run from a cloud shell with Terraform.
  project   = "${var.provider_project}"
  region    = "${var.provider_region}"
  zone      = "${var.splunk_gcp_zone}"

}

resource "google_storage_bucket" "ikea-splunk-gcp-tf-state" {
  name     = "${format("%s","${var.provider_project}-tf-state-bucket")}"
  versioning {
    enabled = "true"
  }
  location = "eu"
  force_destroy = "true"
  lifecycle_rule {
    action {
      type = "Delete"
    }
    condition {
      NumberOfNewerVersions = "50"
    }
  }
}

resource "google_storage_bucket_acl" "image-store-acl" {
  bucket = "${google_storage_bucket.ikea-splunk-gcp-tf-state.name}"
  predefined_acl = "projectprivate"
}
```

Provisioning the Needed Resources



Google Load Balancer
Cloud Load Balancing

Google Load Balancer
Cloud Load Balancing

Splunk
Heavy Forwarders
n1-standard-4
Compute Engine

HF Disk
Persistent Disk

Splunk
Heavy Forwarders
n1-standard-4
Compute Engine

HF Disk
Persistent Disk

Splunk
Heavy Forwarders
n1-standard-4
Compute Engine

HF Disk
Persistent Disk

Splunk
Search Head
Compute Engine

Splunk
Search Head
Persistent Disk

Splunk
License Master
n1-standard-4
Compute Engine

LM Disk
Persistent Disk

Splunk
Cluster Master
n1-standard-4
Compute Engine

CM Disk
Persistent Disk

Text

Splunk
Deployment Server
n1-standard-4
Compute Engine

LM Disk
Persistent Disk

Splunk
Index Cluster
Custom 64vCPU
120GB
Compute Engine

Splunk
IDX Disk SSD
Hot/Warm Storage
Persistent Disk

Splunk
IDX Disk
Cold Storage
Persistent Disk

Splunk
Cold Storage
Cloud Storage

Splunk
Index Cluster
Custom 64vCPU
120GB
Compute Engine

Splunk
IDX Disk SSD
Hot/Warm Storage
Persistent Disk

Splunk
IDX Disk
Cold Storage
Persistent Disk

Splunk
Cold Storage
Cloud Storage

Splunk
Index Cluster
Custom 64vCPU
120GB
Compute Engine

Splunk
IDX Disk SSD
Hot/Warm Storage
Persistent Disk

Splunk
IDX Disk
Cold Storage
Persistent Disk

Splunk
Cold Storage
Cloud Storage

Terraform
Cloud Shell

Ansible
Compute Engine

Ansible Disk
Persistent Disk

Gitlab
Compute Engine

Gitlab Disk
Persistent Disk

Terraforming a Single Host

- Variables declared in tfvars file
- Tags are used by other resources such as load balancers, instance groups and firewalls
- Labels are used by Ansible
- Boot disk is created by default, destroyed if the host is changed.

```
resource "google_compute_instance" "indexer" {
  count          = "${var.indexer_count}"
  name           = "${format("%s","idx${count.index}-${var.cluster_name}-gcp-${var.region}")}"
  machine_type   = "${var.index_machine_type}"
  tags           = [ "${var.cluster_name}", "index", "ssh"]
  labels = {
    environment = "${var.cluster_name}"
    function    = "idx"
    ansible_group = "splunk_${var.cluster_name}_idx"
    splunk_group = "splunk_${var.cluster_name}"
  }

  boot_disk {
    initialize_params {
      image = "${var.index_image}"
      type  = "${var.index_disk_type}"
     }
  }
```

Provisioning Storage

- Disks for Splunk, Warm and Cold
- Persistent Disks are not destroyed by host changes.
- Disks can be increased in size and filesystems resized with Ansible
- Prevent_destroy lifecycle function doesn't really work as expected.

```
// Index Splunk

resource "google_compute_disk" "idx-hot-warm-disk" {
  count = "${var.indexer_count}"
  name  = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}-warm")}"
  type  = "${var.index_hot_warm_disk_type}"
  size  = "${var.index_hot_warm_disk_size}"
  labels    = {
    environment = "splunk"
  }
  physical_block_size_bytes = 4096
  lifecycle {
//    prevent_destroy = true
  }
}

// Index Cold

resource "google_compute_disk" "idx-cold-disk" {
  count = "${var.indexer_count}"
  name  = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}-cold")}"
  type  = "${var.index_cold_disk_type}"
  size  = "${var.index_cold_disk_size}"
  labels = {
    environment = "splunk"
  }
  physical_block_size_bytes = 4096
  lifecycle {
//    prevent_destroy = true
  }
}
```

Attaching Provisioned Disks

- Declared in compute resource.
- Name declared in configuration is also visible in /dev/disk/by-id/

```
boot_disk {
  initialize_params {
    image = "${var.index_image}"
    type  = "${var.index_disk_type}"
   }
}

attached_disk {
  source      = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}")}-splunk"
  device_name = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}")}-splunk"
}


attached_disk {
  source      = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}")}-warm"
  device_name = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}")}-warm"
}

attached_disk {
  source      = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}")}-cold"
  device_name = "${format("idx${count.index}-${var.cluster_name}-gcp-${var.region}")}-cold"
}
```

Compute Host Network Settings

- Public and Private subnets and interfaces
- Public IPs use NAT
- Firewall rules set globally upstream

```
  network_interface {
    subnetwork  = "${var.pub_subnetwork}"

    access_config {
      nat_ip = "${google_compute_address.static[count.index].address}"
    }
  }

  network_interface {
    subnetwork  = "${var.pri_subnetwork}"
    access_config {
    }
  }
}
```

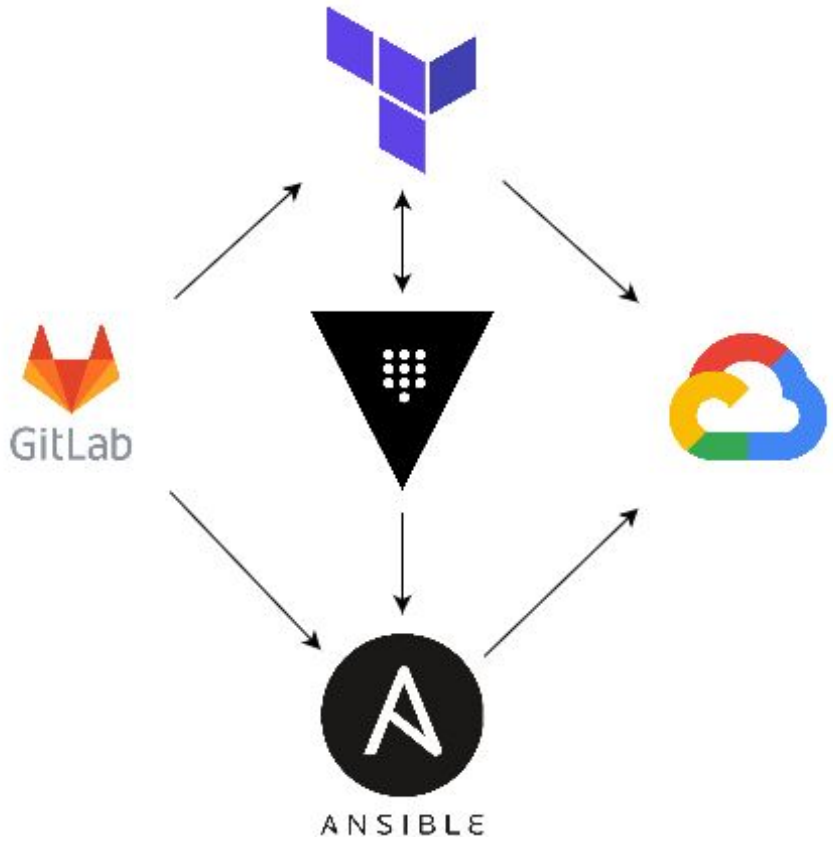
Using Modules to Create Clusters

- Variables declared in Module
- Module is a subfolder of the project
- Indexer_count builds number of hosts

```
// Index Clusters

module "alpha_index_cluster" {
  source                    = "./modules/index_cluster"
  project                   = "${var.provider_project}"
  region                    = "${var.provider_region}"
  cluster_name              = "alpha"
  indexer_count             = "3"
  //index_machine_type        = "custom-64-122880"
  index_machine_type        = "n1-standard-2"
  index_image               = "gce-uefi-images/centos-7"
  index_disk_type           = "pd-ssd"
  index_disk_size           = "30"
  index_hot_warm_disk_type  = "pd-ssd"
  index_hot_warm_disk_size  = "50"
  index_cold_disk_type      = "pd-standard"
  index_cold_disk_size      = "50"
  cm_machine_type           = "n1-standard-2"
  cm_image                  = "gce-uefi-images/centos-7"
  cm_disk_type              = "pd-ssd"
  cm_disk_size              = "30"
  pub_subnetwork                = "${google_compute_subnetwork.public_subnet.name}"
  pri_subnetwork            = "${google_compute_subnetwork.private_subnet.name}"
  create_cm                 = "1"
  splunk_disk_type          = "pd-ssd"
  splunk_disk_size          = "30"
  dns_name_dmz              = "${google_dns_managed_zone.ikea-splunk-prod.dns_name}"
  dns_name_int              = "${google_dns_managed_zone.ikea-splunk-prod-int.dns_name}"
  managed_zone_dmz          = "${google_dns_managed_zone.ikea-splunk-prod.name}"
  managed_zone_int          = "${google_dns_managed_zone.ikea-splunk-prod-int.name}"
  num_reserved_ips          = 32
}
```

Workflow with Vault

- Gitlab Configuration
- Vault Secrets
- Terraform Provisioning
- Ansible Configuration



Generate Secrets

- Generate Splunk Secrets
- Easily done in Terraform
- Don’t ask about randomness

```
// Generate Vault Secrets


resource "random_password" "discover_pass4SymmKey" {
  length = 64
  special = true
  override_special = "./"
}
resource "random_password" "pass4SymmKey" {
  length = 64
  special = true
  override_special = "./"
}
resource "random_password" "admin_passwd" {
  length = 64
  special = true
  override_special = "./"
}
```

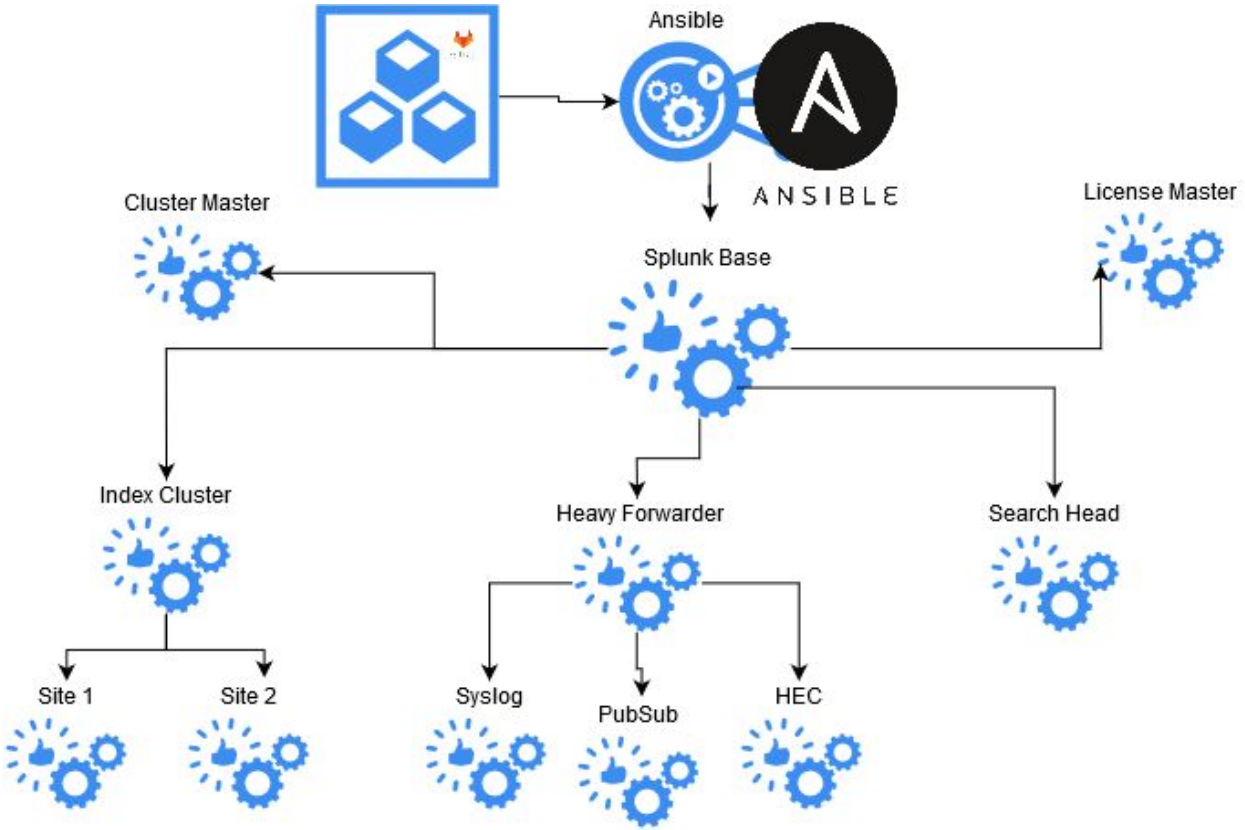
Store Secrets in Vault

- Write config data to Vault
- Used later by Ansible for configuration
- Minimize configuration required to build a new Splunk cluster.

```
resource "vault_generic_secret" "index_cluster_vault" {
  path = "/secret/splunk/${var.cluster_name}"

  data_json = <<EOT
    {
    "splunk_cluster_discovery_pass4SymmKey": "${random_password.discover_pass4SymmKey.result}",
    "splunk_cluster_pass4SymmKey": "${random_password.pass4SymmKey.result}",
    "splunk_cluster_sslPassword": "${data.vault_generic_secret.sslPassword.data["sslPassword"]}",
    "splunk_current_admin_password": "changeme",
    "splunk_default_admin_password": "${random_password.admin_passwd.result}",
    "splunk_ds": "${var.ds_name}",
    "splunk_hf_label": "${var.cluster_name}",
    "splunk_ixc_label": "${var.cluster_name}",
    "splunk_lm": "${var.lm_name}",
    "splunk_master_ip": "${google_dns_record_set.cm-int.0.name}",
    "splunk_old_admin_password": "changme",
    "splunk_serverCertName": "splunkmanagementdata.v4.pem",
    "splunk_sslRootCAPathName": "IKEA-CA-bundle.v4.pem"
    }
  EOT
}
```


GREAT PITCH, AMAZING PRODUCT
I'M OUT
quickmeme.com

Ansible Workflow



Ansible Playbooks

- Uses Labels in GCP set by Terraform
- Applies to Roles in Ansible

```yaml
---
# Configure all splunk nodes with common configuration
- name: Setup base config on Splunk servers
  hosts: "{{ servers | default('splunk_alpha') }}"
  roles:
    - { role: common, tags: ['common', 'common-base'] }
    - { role: splunk-base, tags: ['splunk-base', 'common-base'] }

# Configure the Cluster Master
- name: Setup config on Splunk Cluster Master
  hosts: "{{ cm_servers | default('splunk_alpha_cm') }}"
  roles:
    - { role: splunk-cluster-master, tags: splunk-cluster-master }
    - { role: splunk-web-interface, tags: splunk-web-interface }

# Add the Peer Nodes to the cluster
- name: Setup config on Splunk Peer Nodes
  hosts: "{{ idx_servers | default('splunk_alpha_idx') }}"
  roles:
    - { role: splunk-peer-nodes, tags: splunk-peer-nodes }

# Install Heavy Forwarder for Adapters
- name: Heavy Forwarder for Adapters
  hosts: "{{ hf_adapter_servers | default('splunk_alpha_hf_adapter') }}"
  roles:
    - { role: splunk-heavy-forwarder, tags: splunk-heavy-forwarder }
    - { role: splunk-heavy-forwarder-adapter, tags: splunk-heavy-forwarder-adapter }
    - { role: splunk-web-interface, tags: splunk-web-interface }

# Install Heavy Forwarder for UF trafic
- name: Heavy Forwarder for traffic to 9997 config
  hosts: "{{ hf_9997_servers | default('splunk_alpha_hf_ir') }}"
  roles:
    - { role: splunk-heavy-forwarder, tags: splunk-heavy-forwarder }
    - { role: splunk-heavy-forwarder-9997, tags: splunk-heavy-forwarder-9997 }

# Install Heavy Forwarder HEC
- name: Heavy Forwarder HEC config
  hosts: "{{ hf_hec_servers | default('splunk_alpha_hf_hec') }}"
  roles:
    - { role: splunk-heavy-forwarder, tags: splunk-heavy-forwarder }
    - { role: splunk-heavy-forwarder-hec, tags: splunk-heavy-forwarder-hec }
```

Ansible Roles

Name
..
cluster-uninstall/tasks
common/tasks
git
splunk-base
splunk-cluster-master
splunk-deployment-server
splunk-heavy-forwarder
splunk-heavy-forwarder-9997
splunk-heavy-forwarder-adapter
splunk-heavy-forwarder-hec
splunk-license-master
splunk-peer-nodes
splunk-web-interface

Variables and Secrets

```yaml
---
# Override default values
# Specific configuration for global staging
deploy_stage: "alpha"

# Packages
# Use wget to grab the latest package from splunk.com
# Place file in roles folder under -> splunk-base/files
splunk_package_file: 'splunk-7.2.4-8a94541dcfac.x86_64.rpm'
splunk_version: '7.2.4'
splunk_base: '/opt/splunk'
splunk_vg_name: vg_splunk
splunk_lv_name: lv_splunk
file_system: xfs

# Set mountpoints for buckets
idx_warm_path: "{{ splunk_base }}/var/lib/splunk"
idx_cold_path: "{{ splunk_base }}/var/lib/splunk/cold"
warm_vg_name: vg_warm
warm_lv_name: lv_warm
cold_vg_name: vg_cold
cold_lv_name: lv_cold

# Use multisite or single site cluster config
#splunk_cluster_pass4SymmKey: 'located in secrets file'
use_multisite_config: "false" #set to false when not using

# Multisite config options
site: "site0"
cluster_label: "alpha_Cluster"
search_site: "site0"
available_sites: "site0"
```

```yaml
alpha:
  splunk_ixc_label: "alpha"
  splunk_hf_label: "alpha"
  splunk_master_ip: "cm-alpha-gcp-europe-west1.ikea-splunk-prod.int"
  splunk_ds: 'ds0-gcp-europe-west1.ikea-splunk-prod.int'
  splunk_lm: 'lm.ikea-splunk-prod.int'
  splunk_default_admin_password: 'changeme'
  splunk_old_admin_password: 'changeme'
  splunk_current_admin_password: 'changeme'
  splunk_cluster_pass4SymmKey: 'someRandomKey'
  splunk_cluster_discovery_pass4SymmKey: 'differentRandomKey'
  ssl:
    splunk_cluster_sslPassword: 'IDon'tThinkSo'
    splunk_sslRootCAPathName: "IDon'tThinkSo"
    splunk_serverCertName: "IDon'tThinkSo"
```

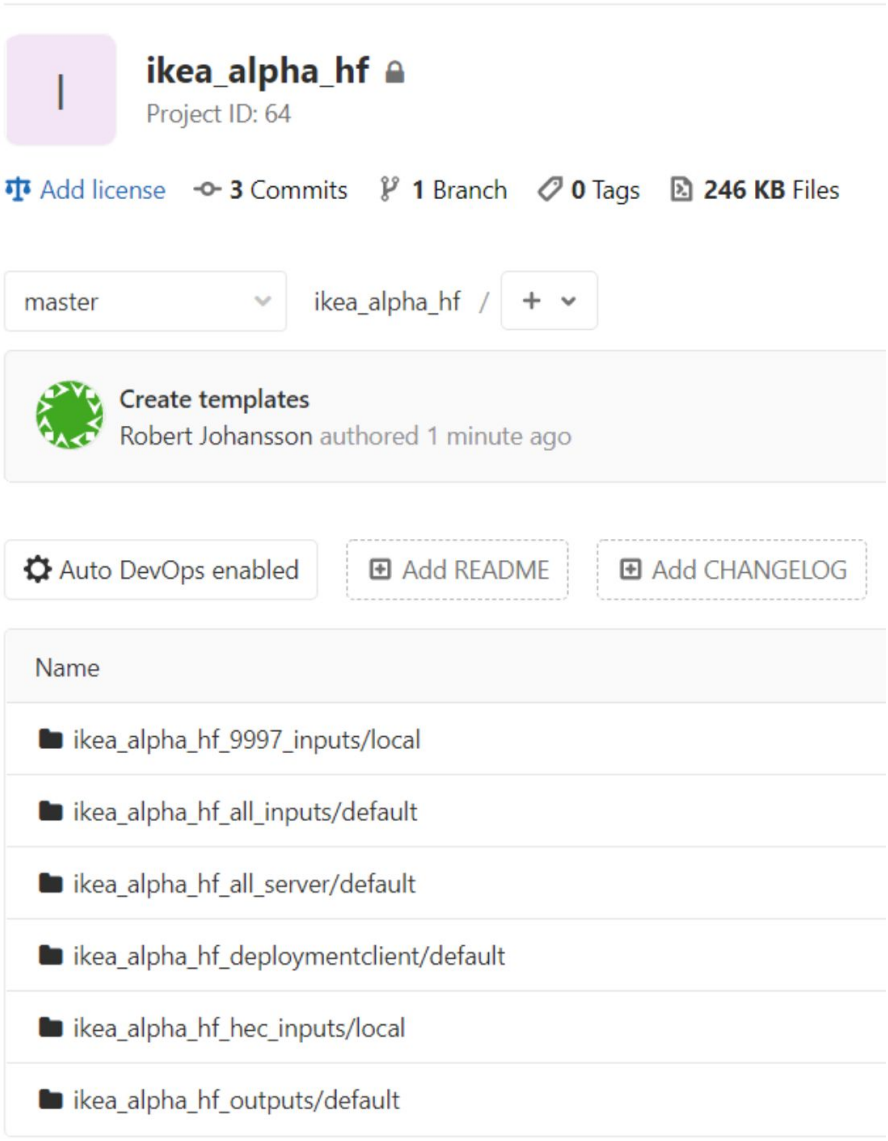
Push and Pull to Git

- Use Ansible to create projects in GitLab from templates
- Pull config from project if it already exists
- One place for all the current configs

```yaml
- name: Create Gitlab Project
  gitlab_project:
    server_url: "{{ git_server }}"
    api_token: "{{ git_token }}"
    group: "{{ git_group }}"
    name: "{{ customer_name }}_{{ stage.splunk_ixc_label }}_idx"
    state: present
    validate_certs: no
  delegate_to: localhost
  run_once: true
  register: gitCreate

- name: Populate new Git project
  import_tasks: create_conf.yml
  when: gitCreate.changed

- name: "Git: is it up-to-date?"
  git:
    repo: "{{ git_address }}:{{ git_group }}/{{ customer_name }}_{{ stage.splunk_ixc_label }}_idx"
    dest: "{{ git_tmp }}/{{ customer_name }}_{{ stage.splunk_ixc_label }}_idx"
    update: yes
    version: master
  register: git
  delegate_to: localhost
  run_once: true
- debug:
    var: git

- name: "Create {{ customer_name }}_{{ stage.splunk_ixc_label }}_master_base/local directories"
  file:
    path: "{{ item }}"
    owner: splunk
    group: splunk
    mode: u=rwX,g=rX,o-rwx
    recurse: yes
    state: directory
  with_items:
    - "{{ splunk_base }}/etc/apps/{{ customer_name }}_{{ stage.splunk_ixc_label }}_master_base/
    - "{{ splunk_base }}/etc/apps/{{ customer_name }}_{{ stage.splunk_ixc_label }}_master_base/
```

Git Project

- Separate project for each cluster and function
- All Splunk config is in apps, nothing in system/local
- Using Splunk base apps standard



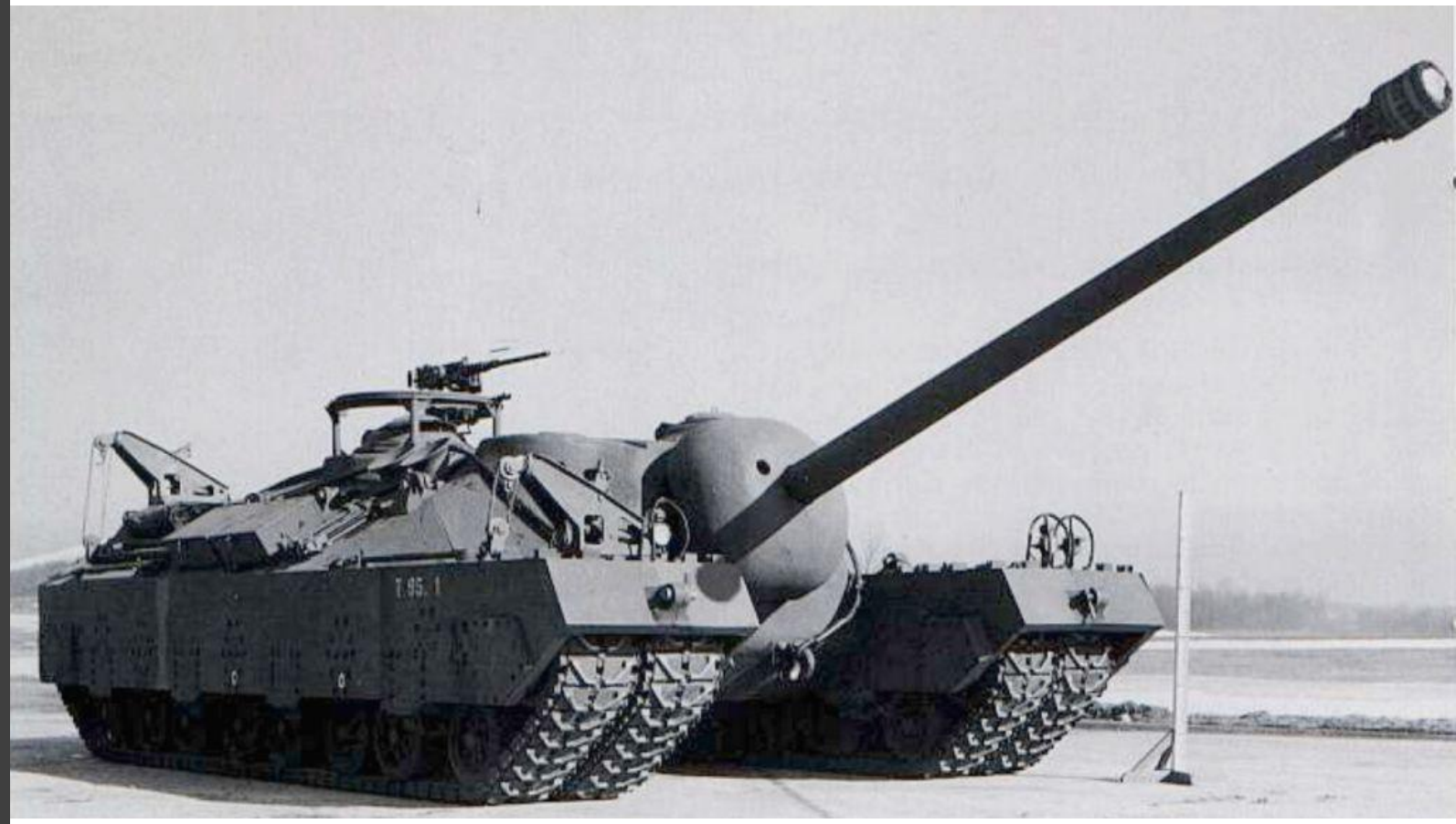
What could possibly go wrong?

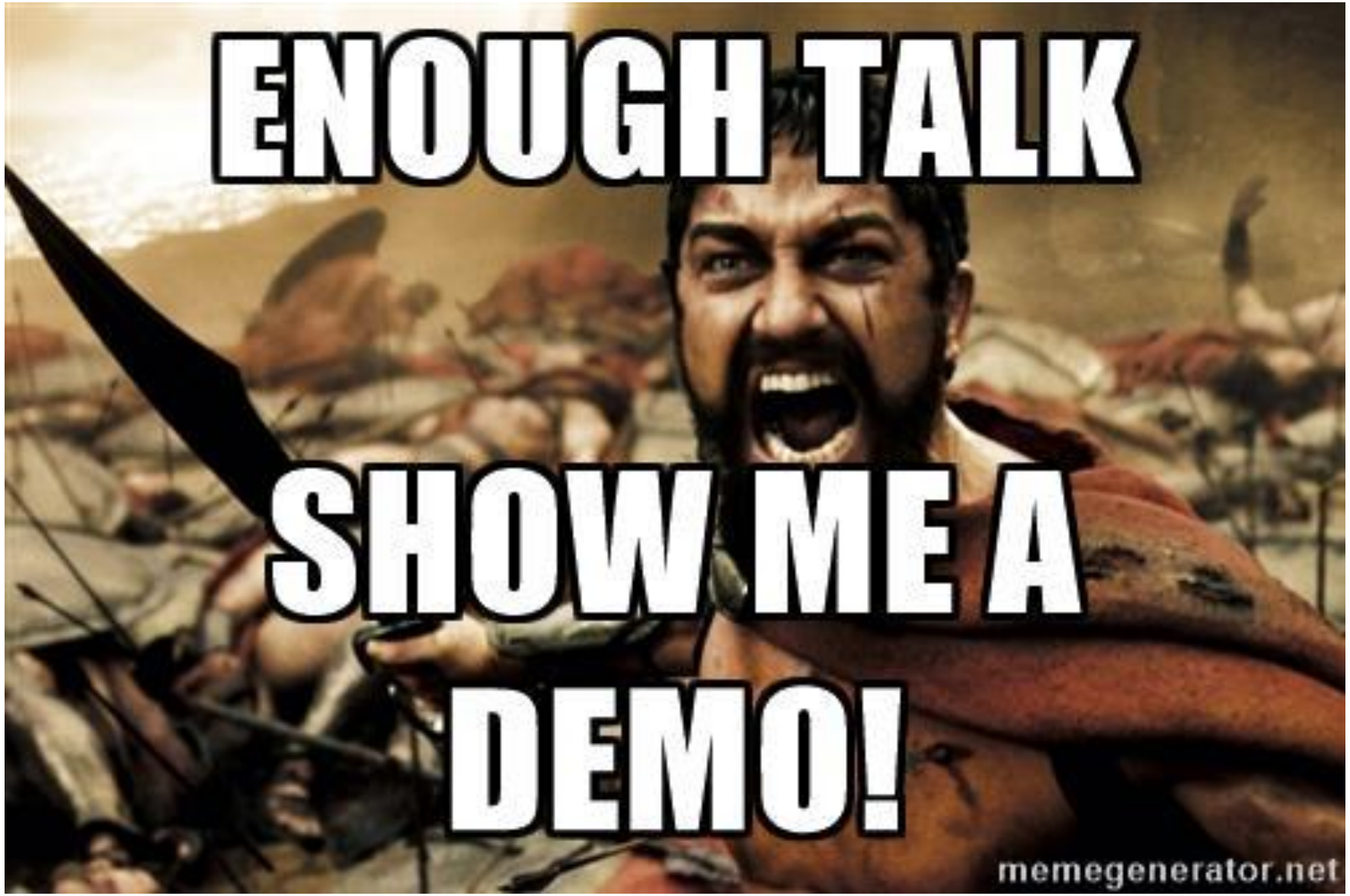
- Easy come, easy go
- Corrupt Terraform Statefile
- Terraform Lifecycle doesn’t work if code is removed.
- Restarting the cluster all at once

Some assembly required.

Bigger, Better, More Derpy

- Autoscaling
- Further Vault Integration
- Single Configuration Source
- Log All The Things to Splunk
- SmartStore for Cold Buckets
- Automagic PubSub
- Consul for Connectivity




ENOUGH TALK
SHOW ME A DEMO!
memegenerator.net