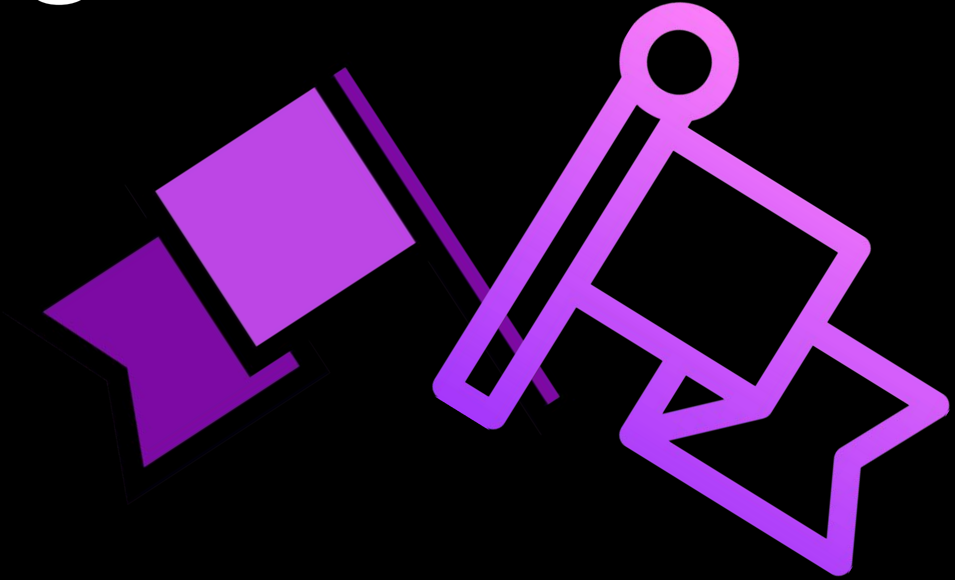
Feature Flags in Terraform



Prabesh Thapa

@pgaijin66

About me

Prabesh Thapa
SRE/Devops

Key topics for discussion

> Feature flags and why should we use them ?

> Implementations of feature toggles in Terraform

> Pass feature flags from config file

> Demo

What is Terraform ?

Infrastructure as Code

API driven

Declarative code using HCL

What are feature flags ?

- Enable feature without additional code
- Also known as “toggles” or “switches”
- Controlled operation
- Passed from config files, env variables or dedicated feature flag management platforms like Launchdarkly

Why should we use feature flags in IasC ?

> Increased flexibility

> Selectively provision resource

> Improved scalability

> Better control over deployments

> Improved safety

> Effective debugging

Enough already ...

The count Meta-Argument

count is a meta-argument defined by the Terraform language. It can be used with modules and with every resource type.

Creates specified number of identical resources

count = 0 => No resource creation

```
resource "aws_instance" "server" {

 count = 4

 ami           = "ami-a1b2c3d4"
 instance_type = "t2.micro"

 tags = {
   Name = "Server ${count.index}"
 }
}
```

Conditional **Expression**

Conditional expressions are used to evaluate boolean expressions

Selects one of two values based on the boolean expression

```
condition ? true_val : false_val
```

Conditional expressions

count is a meta-argument defined by the Terraform language. It can be used with modules and with every resource type.

```
resource "aws_instance" "server" {

 count = 4

 ami           = "ami-a1b2c3d4"
 instance_type = "t2.micro"

 tags = {
   Name = "Server ${count.index}"
 }
}
```

Count + Conditional Expression = Feature flags

Environment toggles

Enable or disable specific feature or functionality based on environment.

```
variable "env" {
        type = string
        default = "dev"
}
```

```
resource "digitalocean_droplet" "puny_vm" {
 image  = "ubuntu-18-04-x64"
 count = var.env == "dev" ? 1 : 0
 name   = "puny_vm"
 region = "nyc2"
 size   = "s-1vcpu-1gb"
}

resource "digitalocean_droplet" "beefy_vm" {
 image  = "ubuntu-18-04-x64"
 count = var.env == "prod" ? 3 : 0
 name   = "beefy_vm"
 region = "nyc2"
 size   = "s-4vcpu-8gb"
}
```

Resource toggles

Toggle resource creation based on a dedicated feature flag

```
locals {
        feature_flags = {
                provision_lb : false
        }
}
```

```
resource "digitalocean_loadbalancer" "public" {
 name   = "loadbalancer-1"
 region = "nyc3"

 count = var.provision_lb ? 1 : 0

 forwarding_rule {
  entry_port     = 80
  entry_protocol = "http"
  target_port     = 80
  target_protocol = "http"
 }

 healthcheck {
  port     = 22
  protocol = "tcp"
 }
 droplet_ids = var.droplets_id[*]
}
```

Module toggles

Toggle modules based on a dedicated feature flag

```
locals {
          feature_flags = {
                    provision_db : false
          }
}
```

```
module "lb" {
 source = "./modules/loadbalancer"

 count = local.config.feature_flags.provision_lb ? 1 : 0

 do_token = var.do_token
 droplets_id = module.droplet.droplets_id
}
```

There are more...

Blue green deployment using DO floating IPs

Canary release using AWS ALB target groups

Organising feature flags

Define feature flags directly on the Terraform configuration file using local variable

```
locals {
          feature_flags = {
                    provision_lb : false
                    provision_db : false
          }
}
```

main.tf

```
> local.feature_flags.provision_lb
false
```

Organising feature flags

Pass feature flags using config.yaml file

```yaml
---
env: "dev"
feature_flags:
    provision_lb: false
    provision_db: false
regions:
    - name: "us-east-1"
      vpcs:
        -  name: "vpc-use1-vpc"
           cidr_supernet: "10.0.0.0/16"
           availability_zones:
            -  name: "us-east-1a"
               netblocks:
                  cidr_subnet_public: "10.0.0.0/20"
                  cidr_subnet_private: "10.0.16.0/20"
```

Organising feature flags

Pass feature flags using config.yaml file

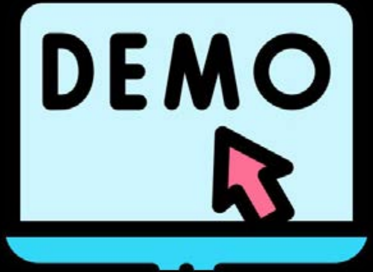
```
locals {
        config = yamldecode(file("config.yaml"))
}
```

locals.tf

```
module "droplet" {
        source = "../../modules/droplet"
        do_token = var.do_token
        env = local.config.env
}
```

modules.tf

Demo



Key takeaways

> Feature flags is a powerful technique for managing resources using infra as code

> We should start using it as part of infra as code.

> Helps to improve safety, scalability and maintainability of your infrastructure

Source code

https://github.com/pgaijin66/feature-flags-in-terraform

References

- https://www.hashicorp.com/blog/terraform-feature-toggles-blue-green-deployments-canary-test
- https://build5nines.com/terraform-feature-flags-environment-toggle-design-patterns/
- https://objectpartners.com/2021/10/19/feature-flags-in-terraform/
- https://developer.hashicorp.com/terraform/language/meta-arguments/count

Thank you for listening
Let's connect

Prabesh Thapa

@pgaijin66

@ShardedSRE