Treinamento Terraform

Por que usar Infra as Code?

- **Manter a infraestrutura versionada em um software de controle de versão;**
- **Garantir que os diferentes ambientes (dev, hom, e prd) estão iguais, com isso reduzindo o trabalho de gerenciamento de ambientes**

Algumas ferramentas de IaC

Temos uma serie de ferramentas que podem ser usadas para IaC:

- **CloudFormation (AWS);**
- **AWS CDK (AWS);**
- **ARM Template (Azure);**
- **Terraform;**

Por que aprender Terraform?

Terraform é uma solução de IaC que pode ser usadas em todos os grandes fornecedores de cloud:

- AWS;
- Azure;
- GCP;
- Alibaba;
- Kubernetes;
- OpenShift;

Além disso é possível escrever providers e resorces para qualquer nova nuvem ou plataforma.

Instalação

Exemplo 1 – Criação de um bucket

Laboratório 1 – criação de bucket AWS

Providers

```
provider "aws" {
   access_key = "XXXXXXXXXXXXXXXXXXXXX"
   secret_key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
   region = "us-east-1"
}
```

*No providers escolhemos qual é o fornecedor o vendor que irmos utilizar, aqui temos suporte para praticamente todas as nuvens (https://registry.terraform.io/browse/providers)

Resouces

```
resource "aws_s3_bucket" "bucket" {
    bucket = "teste122334"
    tags = {teste="12345"}
}
```

No resource temos a descrição do objeto criado na cloud especificada.

Resouces

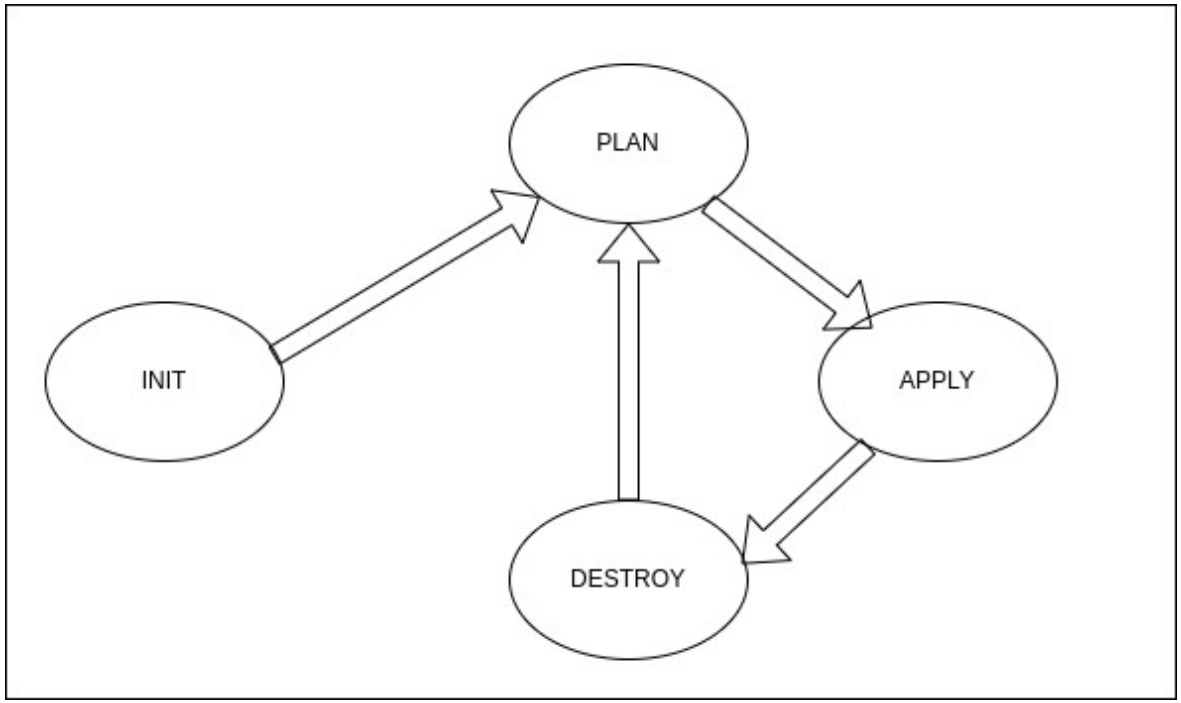
```
resource "aws_s3_bucket" "bucket" {
    bucket = "teste122334"
    tags = {teste="12345"}
}
```

A nomenclatura de um novo recurso segue a seguinte ordem **Nomedoprovider_Serviço_Feature.**

O nome lógico “bucket” deve ser **único dentro da stack.**

Entre chaves temos os argumentos necessários para a criação do novo recurso.

Ciclo de vida de uma aplicação Terraform



Etapa INIT – Nessa etapa o terraform instala os providers necessários para criação da sua infraestrutura;
Etapa PLAN – Nessa etapa o terraform analisa o código, verifica possíveis erros de sintaxe e atualiza a quais elementos serão criados os destruídos em relação a ultima execução;
Etapa APPLY – Aqui o terraform comunica cria de fato a infraestrutura;
Etapa DESTROY – Aqui destruímos toda a pilha criada;

Variaveis

Variáveis são utilizadas toda vez que temos valores diferentes de acordo com o ambiente;

Sua declaração segue o exemplo:

```
variable "namebucket" {
    type = string
    description = "teste"
}
```

Tipos de variaveis

Os tipos suportados pelo Terraform são:

- **string** – Uma sequencia de caracteres Unicode;
- **number –** Representa valores númericos, pode ser inteiros ou fracionários;
- **bool** – Valores booleanos (**true** ou **false**);
- **list** (ou tuplas) – Conjunto de elementos, podem ser *string*, *number* ou *bool*, por exemplo: [“sa-east-1”, “us-east-1”];
- **map** – Aqui temos a representação de objetos;
- **null** – Para valores nulos;

Setando as variaveis com arquivo

Existem algumas formas de setar as variáveis no terraform, essa substituição de valores é executada na de **Plan.**

Abordaremos aqui apenas a pratica mais comum para diferenciar os valores conforme o ambiente;

Laboratório 2 - Variaveis

Laboratório 2 - variaveis

Terraform linha de comando

init - Prepara o diretório atual para ser um projeto terraform, nessa etapa são instalados os provider que serão utilizados no projeto;

validate - valida se os arquivos do projeto estão com configuração correta (nota – procure sempre executar o **validate** a cada alteração no projeto);

plan - exibe as atualizações realizadas na stack;

apply – aplica a stack criada no ambiente;

destroy – destroi a stack criada apaga os recursos gerados;

Funções Built-In no terraform

Módulos em terraform

Usamos módulos para criar componentes reutilizáveis e que possam ser distribuídos;

Aqui temos a seguinte estrutura:

- **Parâmetros de entrada, definidos como variáveis;**
- **Definição de infra;**
- **Parâmetros de saída;**

Laboratorio 3 - Módulos

Laboratorio 3 - Arquitetura