

# **DevOps e IaC**

**Autores:** João Frois e Lucas Padrão

**Professor:** Johnatan Alves de Oliveira

**“DevOps não se trata de automação,  
assim como astronomia não se  
trata de telescópios” – Christopher Little**



# DEVOPS

Na sua essência, o DevOps é uma cultura, um movimento, uma filosofia. É um aperto de mão entre desenvolvimento e operações que enfatiza uma mudança de mentalidade, melhor colaboração e integração mais rígida.

Une entrega ágil e contínua para ajudar as equipes de desenvolvimento e operações a serem mais eficientes, inovadoras e fornecerem maior valor a empresas e clientes.

# PILARES

## CULTURA

- Foco nas pessoas;
- Melhorar comportamentos;
- Abraçar mudanças e experimentações;  
Integral Dev + Ops

## AUTOMAÇÃO

- Fazer pequenas e constantes entregas;
- Fazer pequenas e constantes entregas;
- Estruturas centralizadas de Testes e Deploy;
- Infraestrutura como código

## FLUXO

- Entregar valor para o usuário final;
- Identificar o fluxo de valor;
- Eliminar gastos;
- Perseguir a melhoria contínua e fazer o simples

## IaC - Infrastructure as Code

Gerenciamento e provisionamento da infraestrutura necessária a um software por meio de ferramentas que possibilitem codificá-la ao invés de depender de processos manuais

Por meio de arquivos de configuração, que especificam os recursos que a compõem, há uma maior facilidade em modificar aspectos da infraestrutura.

Promover maior escalabilidade e controle

# FERRAMENTAS



# FERRAMENTAS



# Terraform

**Autores:** João Frois e Lucas Padrão

**Professor:** Johnatan Alves de Oliveira



# TERRAFORM

O Terraform é uma ferramenta de código aberto que permite que você defina sua infraestrutura como código e a automatize, fornecendo um fluxo de trabalho CLI consistente para gerenciar centenas de serviços em nuvem através arquivos de configuração declarativos.

# CARACTERÍSTICAS

1

## PROVISIONAMENTO

O Terraform é uma ferramenta de provisionamento de infraestrutura que permite que você descreva e gerencie sua infraestrutura como código.

2

## MULTIPLATAFORMA

Ele é multiplataforma e trabalha com diversos provedores de nuvem, incluindo AWS, Azure e GCP.

3

## ESCALABILIDADE

O Terraform é escalável e é ideal para gerenciar infraestruturas complexas.

4

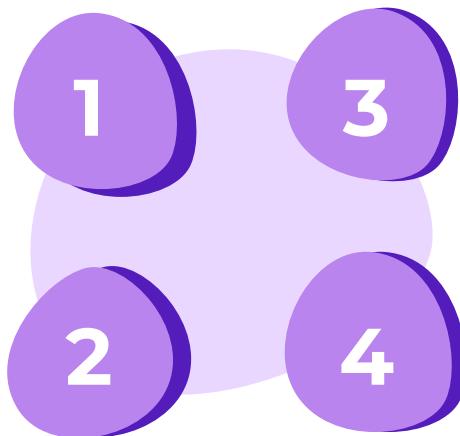
## GERENCIAMENTO DE ESTADO

O Terraform também gerencia o estado da infraestrutura, o que significa que pode reconstruir a mesma infraestrutura várias vezes, mantendo sua configuração original.

# FUNCIONAMENTO

Você descreve sua infraestrutura desejada em configuração declarativa.

O Terraform cria um plano de execução para atingir o estado desejado.



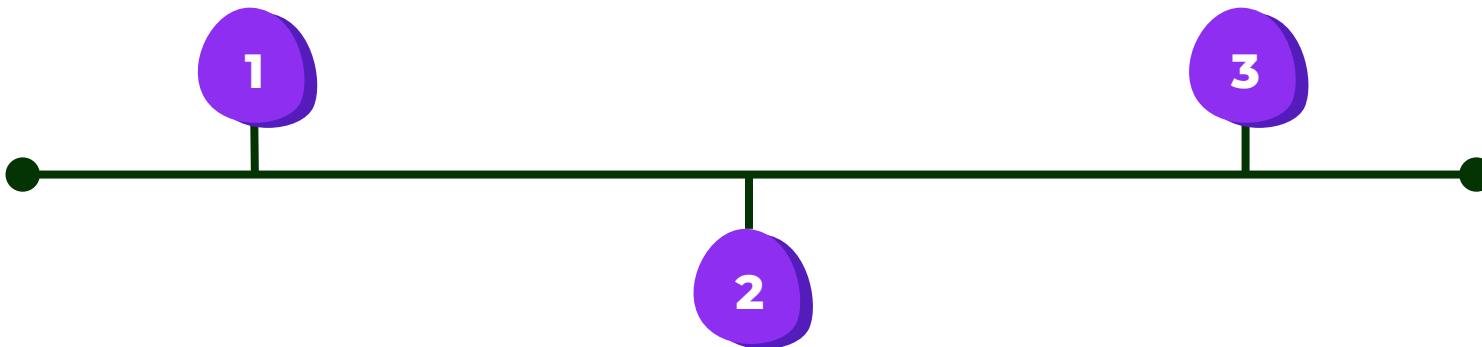
O Terraform provisiona a infraestrutura de acordo com o plano gerado.

O Terraform gerencia o estado da infraestrutura e pode atualizá-la conforme necessário.

# COMANDOS

## `terraform init`

Este comando é usado para inicializar um novo diretório de projeto do Terraform. Ele baixa os plugins necessários para os provedores de infraestrutura especificados no arquivo de configuração.



## `terraform apply`

Este comando aplica as mudanças planejadas pela execução do comando `terraform plan`. Ele cria ou modifica recursos de infraestrutura de acordo com a configuração definida e atualiza o estado do Terraform.

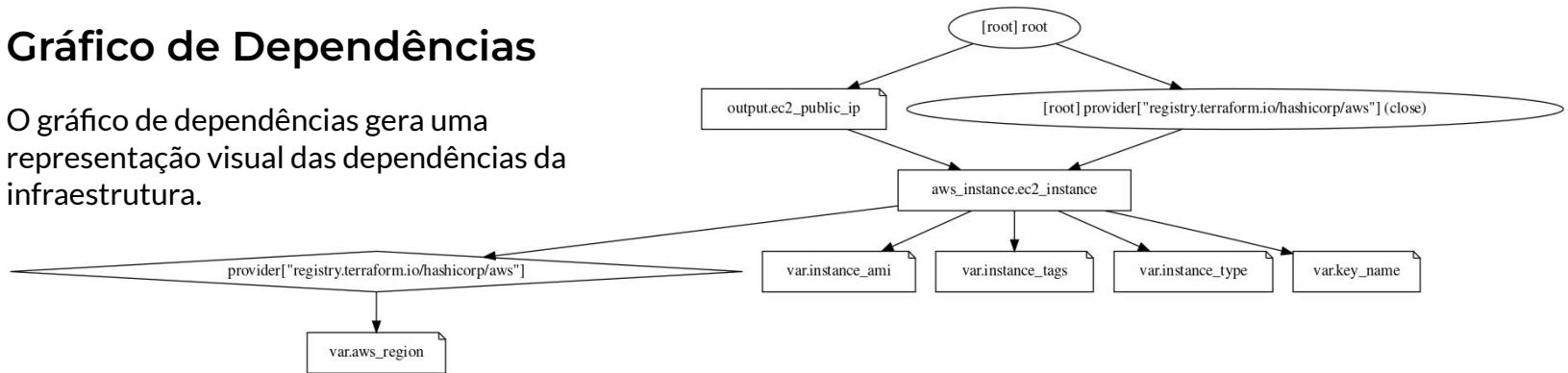
## `terraform plan`

O comando `plan` é usado para criar um plano de execução que mostra quais recursos serão criados, modificados ou destruídos com base na configuração atual e nos estados anteriores. Ele é uma etapa importante antes de aplicar qualquer mudança na infraestrutura para evitar surpresas indesejadas.

# PRINCIPAIS RECURSOS

## Gráfico de Dependências

O gráfico de dependências gera uma representação visual das dependências da infraestrutura.



## Console do Terraform

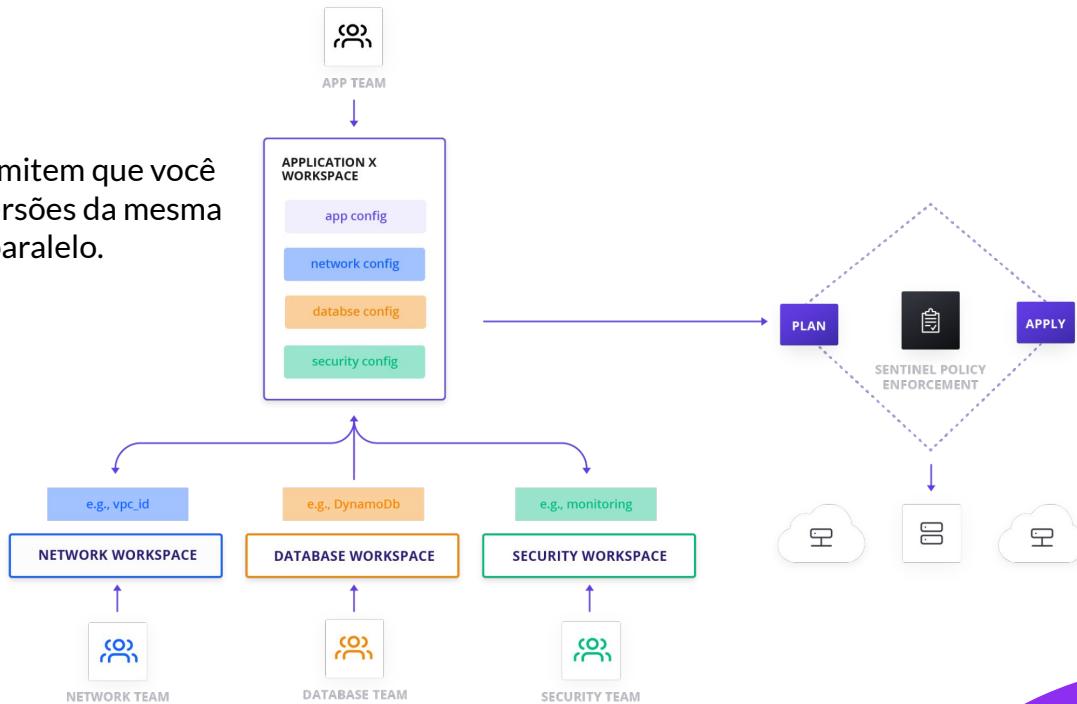
O console do Terraform permite visualizar e modificar o estado da infraestrutura.

```
Plan: 1 to add, 0 to change, 0 to destroy.  
aws_instance.web: Creating...  
aws_instance.web: Still creating... [10s elapsed]  
aws_instance.web: Still creating... [20s elapsed]  
aws_instance.web: Still creating... [30s elapsed]  
aws_instance.web: Creation complete after 36s [id=i-0a76e130dc4deb4eb]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

# PRINCIPAIS RECURSOS

## Workspaces

Os workspaces permitem que você mantenha várias versões da mesma infraestrutura em paralelo.



# BENEFÍCIOS

## AGILIDADE

O Terraform permite que você provisione e gerencie sua infraestrutura com mais rapidez e eficiência.

## VISIBILIDADE

Com o Terraform, é fácil visualizar e monitorar sua infraestrutura em tempo real.

## RASTREABILIDADE

O Terraform permite rastrear as alterações na sua infraestrutura ao longo do tempo.

# PRINCIPAIS ESTRUTURAS

1

## Providers (Provedores)

Os provedores são responsáveis por se comunicar com serviços de infraestrutura e fornecer recursos específicos desse serviço para serem gerenciados pelo Terraform.

```
provider "aws" {  
    region  = var.aws_region  
    profile = "tcc"  
}
```

2

## Variables (Variáveis)

As variáveis permitem que você passe informações dinâmicas para seus módulos e configurações, tornando seu código mais flexível e reutilizável.

```
variable "aws_region" {  
    type      = string  
    default   = "us-east-1"  
    description = "Region of the AWS"  
}
```

# PRINCIPAIS ESTRUTURAS

3

## Modules (Módulos)

Os módulos permitem que você encapsule e reutilize configurações e recursos Terraform, promovendo a modularidade e a organização do código.

```
module "vpc" {  
  source      = "./modules/vpc"  
  vpc_base_ip = "10.0.0.0"  
  subnet_base_ip = "10.0.1.0"  
}
```

4

## Resources (Recursos)

Os recursos são os principais blocos de construção do Terraform e representam os recursos de infraestrutura que você deseja criar e gerenciar. Eles são fornecidos pelos provedores.

```
resource "aws_instance" "web" {  
  ami           = data.aws_ami.ubuntu.id  
  instance_type = var.instance_type  
  
  You, há 3 semanas | 1 author (You)  
  tags = {  
    Name      = var.instance_name  
    CreatedFor = "terraform"  
  }  
}
```

# PRINCIPAIS ESTRUTURAS

5

## Outputs (Saídas)

As saídas permitem que você exponha informações sobre recursos criados para uso posterior, seja para visualização ou para outras partes do código.

6

## Data Sources (Fontes de Dados)

As fontes de dados permitem que você consulte informações existentes em sua infraestrutura, como informações sobre subnets, AMIs ou outros recursos, para serem usadas na configuração.

```
output "ec2_instance_public_ip" {
  value = module.ec2_instance.ec2_public_ip
  description = "Public IP for EC2 instance"
}
```

```
data "aws ami" "ubuntu" {
  most_recent = true

  You, há 3 semanas | 1 author (You)
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  You, há 3 semanas | 1 author (You)
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}
```

# THANKS!