



O QUE É O TERRAFORM?

- Terraform é uma ferramenta de software livre de (IaC) "infraestrutura como código" criada pela HashiCorp.
- Utiliza a linguagem de configuração de alto nível chamada HCL (HashiCorp Configuration Language).
- Software livre;
- Plataforma independente;
- Infraestrutura imutável;
- Para entender melhor as **vantagens** do **Terraform**, é interessante entender primeiro os **benefícios** IaC. A IaC permite aos desenvolvedores **codificar** a **infraestrutura** de um modo que torna o **provisionamento automatizado**, mais **rápido** e **reproduzível**. Ela é um componente chave das práticas **Agile** e **DevOps**, como o **controle de versão**, a **integração contínua** e a **implementação contínua**.

Download Terraform

macOS Windows Linux FreeBSD OpenBSD Solaris



<https://www.terraform.io/downloads>

validate

terraform validate

O comando valida os arquivos de configuração em um diretório, referindo-se apenas à configuração e não acessando nenhum serviço remoto, como estado remoto, APIs de provedores, etc.

fmt

terraform fmt

O comando é usado para reescrever os arquivos de configuração do Terraform para um formato e estilo canônicos.

tfstate

terraform.tfstate

O Terraform deve armazenar o estado de sua infraestrutura e configuração gerenciadas. Esse estado é usado pelo Terraform para mapear recursos do mundo real para sua configuração, acompanhar metadados e melhorar o desempenho de grandes infraestruturas.

show

terraform show

Pode ser usado para inspecionar um plano para garantir que as operações planejadas sejam esperadas ou para inspecionar o estado atual como o Terraform o vê.

PRINCIPAIS COMANDOS



install

Simplex

<https://developer.hashicorp.com/terraform/downloads>

terraform init

init

O comando inicializa um diretório de trabalho contendo os arquivos de configuração do Terraform.

01

plan

terraform plan

O comando cria um plano de execução, que permite visualizar as alterações que o Terraform planeja fazer em sua infraestrutura.

02

apply

terraform apply

Executa as ações propostas em um plano do Terraform.

03

destroy

terraform destroy

O comando terraform destroy é uma maneira conveniente de destruir todos os objetos remotos gerenciados por uma configuração específica do Terraform.

PROVIDERS

```
terraform {  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 4.0"  
    }  
  }  
}
```

Os provedores do Terraform são plug-ins que implementam tipos de recursos.

Os provedores contêm todo o código necessário para autenticar e conectar a um serviço, geralmente de um provedor de cloud pública, em nome do usuário.

É possível encontrar provedores para as plataformas e serviços na cloud que você usa, incluir à sua configuração e, em seguida, usar seus recursos para fornecer a infraestrutura.

Os provedores estão disponíveis em praticamente todos os principais provedores de cloud, soluções de SaaS e muito mais, desenvolvidos e/ou apoiados pela comunidade Terraform ou por organizações individuais.



Obs. sempre que adicionar um novo módulo ou provider execute o "terraform init".

DEPENDÊNCIAS IMPLÍCITAS VS EXPLÍCITAS

- Dependências implícitas, como seus nomes sugerem, são detectadas automaticamente pelo Terraform.
- Dependências explícitas são dependências configuradas “manualmente” entre recursos, usando a palavra-chave `depends_on`. Antes de subir um recurso preciso que outro esteja no ar. `depends_on` = exemplo uma EC2 precisa de um disco EFS.
- Recursos não dependentes - Melhores práticas não usar o dependência explícita sem necessidade, porque o terraform trabalha com multi threads.
- O ideal é sempre usar os recursos do provedor como o exemplo do `user_data` no momento de subir uma instancia.

Neste caso, significa que a interface de rede deve ser criada antes da máquina virtual e a máquina virtual deve ser excluída antes da interface de rede.

```
# ==Implicit Dependencies==  
  
resource "aws_instance" "example" {  
  ami           = "ami-b374d5a5"  
  instance_type = "t2.micro"  
}  
  
resource "aws_eip" "ip" {  
  vpc      = true  
  instance = aws_instance.example.id  
}
```

#==Explicit Dependencies==

```
resource "aws_s3_bucket" "example" {  
  bucket = "terraform-getting-started-guide"  
  acl    = "private"  
}  
  
resource "aws_instance" "example" {  
  ami           = "ami-2757f631"  
  instance_type = "t2.micro"  
  depends_on    = [aws_s3_bucket.example]  
}
```

Neste caso, ele só iniciará a criação da instância quando o bucket s3 estiver pronto.

PROVIDERS: FILE, LOCAL EXEC E REMOTE EXEC

File - copiará um arquivo local para o destino via ssh ou winrm;

```
provisioner "file" {  
  source    = "conf/myapp.conf"  
  destination = "/etc/myapp.conf"  
}
```

Local exec - dispara um comando na maquina local;

```
provisioner "local-exec" {  
  command = "echo ${self.private_ip} >> private_ips.txt"  
}  
}
```

Remote exec - executará o comando na maquina remota;

```
provisioner "file" {  
  source    = "script.sh"  
  destination = "/tmp/script.sh"  
}  
  
provisioner "remote-exec" {  
  inline = [  
    "chmod +x /tmp/script.sh",  
    "/tmp/script.sh args",  
  ]  
}  
}
```


INPUT VARIABLES

Maps

Os mapas são uma maneira de criar variáveis que são tabelas de pesquisa. Um exemplo mostrará isso melhor. Vamos extrair nossas AMIs em um mapa e adicionar suporte para a região:

```
variable "amis" {
  type = "map"
  default = {
    "us-east-1" = "ami-b374d5a5"
    "us-west-2" = "ami-4b32be2b"
  }
}

resource "aws_instance" "example" {
  ami          = var.amis[var.region]
  instance_type = "t2.micro"
}
```

Strings

Quando a variável é tipo string basta declara e colocar em aspas dupla -> tipo = " "

```
variable "location" {
  type = string
  default = " us-east-1"
}
```

Lists

O tipo lista precisa estar entre chaves [" ", " "] ou [" "].

```
variable "users" {
  type    = "list"
  default = ["admin", "ubuntu"]
}
```

OUTPUT VALUES

- Os valores de saída disponibilizam informações sobre sua infraestrutura na linha de comando e podem expor informações para outras configurações do Terraform usarem.
- Os valores de saída são semelhantes aos valores de retorno em linguagens de programação.

```
output "instance_ip_addr" {  
  value = aws_instance.server.private_ip  
}
```

```
output "db_password" {  
  value      = aws_db_instance.db.password  
  description = "The password for logging in to the  
database."  
  sensitive  = true  
}
```

REGISTRY MODULES

- Além dos módulos do sistema de arquivos local, o Terraform pode carregar módulos de um registro público ou privado
- O Terraform Registry hospeda uma ampla coleção de módulos Terraform disponíveis publicamente para configurar vários tipos de infraestrutura comum. Esses módulos são de uso gratuito e o Terraform pode baixá-los automaticamente se você especificar a fonte e a versão apropriadas em um bloco de chamada de módulo.
- Além disso, os membros de sua organização podem produzir módulos criados especificamente para suas próprias necessidades de infraestrutura.

```
module "vpc" {  
  source = "terraform-aws-modules/vpc/aws"  
  version = "3.18.1"  
}
```

```
module "vpc" {  
  source = "terraform-aws-modules/vpc/aws"  
  
  name = "my-vpc"  
  cidr = "10.0.0.0/16"  
  
  azs          = ["eu-west-1a", "eu-west-1b", "eu-west-1c"]  
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]  
  public_subnets = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]  
  
  enable_nat_gateway = true  
  enable_vpn_gateway = true  
  
  tags = {  
    Terraform = "true"  
    Environment = "dev"  
  }  
}
```


DATASOURCE

- As fontes de dados do Terraform permitem buscar dados dinamicamente de APIs ou outros back-ends de estado do Terraform.
- Exemplos de fontes de dados incluem IDs de imagem de máquina de um provedor de nuvem ou saídas do Terraform de outras configurações.

```
data "aws_ami" "amazon_linux" {  
  most_recent = true  
  
  filter {  
    name = "name"  
  
    values = [  
      "amzn-ami-hvm-*-x86_64-gp2",  
    ]  
  }  
  
  filter {  
    name = "owner-alias"  
  
    values = [  
      "amazon",  
    ]  
  }  
}
```

ARQUIVOS E PASTAS

- `main.tf` - chame módulos, locais e fontes de dados para criar todos os recursos.
- `variables.tf` - contém declarações de variáveis utilizadas em `main.tf`.
- `outputs.tf` - contém saídas dos recursos criados em `main.tf`.
- `.tfvars` - se você deseja definir variáveis no Terraform, é recomendável especificar seus valores em um arquivo. Um arquivo de definição de variável no Terraform termina em `.tfvars` ou `.tfvars.json`;
- `terraform.tfstate` - armazenar o estado de sua infraestrutura e configuração gerenciadas;
- `.terraform` – ao executarmos “`terraform init`” será criado essa pasta oculta aonde será feito o download dos módulos e providers;

TRABALHANDO COM BACKENDS

- Um backend define onde o Terraform armazena seus arquivos de dados de estado.
- O Terraform usa dados de estado persistentes para acompanhar os recursos que gerencia e por padrão fica no arquivo terraform.tfstate.
- A maioria das configurações não triviais do Terraform podemos integrar ao Terraform Cloud ou usa um backend para armazenar o estado remotamente. Isso permite que várias pessoas acessem os dados de estado e trabalhem juntas nessa coleção de recursos de infraestrutura.

<https://developer.hashicorp.com/terraform/language/settings/backends/configuration>

<https://www.youtube.com/watch?v=GIAErV6Hk1o>

<https://app.terraform.io/app/>

MELHORES PRÁTICAS

Não armazene o terraform.tfstate num repositório publico do git, porque ele pode conter variáveis de conexão com banco de dados.

Utilize o terraform conectado com git usando branches e integre seu deploy com o terraform app, trabalhando com backend automatizado.

Cuidado para não expor suas chaves de autenticação do seu providers em repositórios de git publico.

Pode-se também utilizar o estado de maquina num banco de dados ou bucket s3 *privado*.

Recursos não dependentes - Melhores práticas não usar o dependência explicita sem necessidade para que o terraform crie sua infraestrutura com todos os recursos simultaneamente usando multi theads.

DOCUMENTAÇÃO

- A documentação da hashicorp para o terraform é muito ampla.
- Essa apresentação te ajudará a se ambientar no site auxiliando na localização de um provider, de um módulo, para entender uma variável, com as melhores práticas e fornecendo exemplos de códigos.

Página home que direciona para download, community, terraform cloud, dev, app e registry.
<https://www.terraform.io/>

Suporte para desenvolvedor com tutoriais.
<https://developer.hashicorp.com/>

Documentação de providers e módulos.
<https://registry.terraform.io/>

Backend terraform.
<https://app.terraform.io/app/>

Obrigado