



/COMO TRABALHAR COM MÚLTIPLOS AMBIENTES NO TERRAFORM

TDC INNOVATION 2023





<ANTONIO.JUNIOR>

Principal Cloud & DevOps Solution Architect
Microsoft MVP Azure



Microsoft®
Most Valuable
Professional





/AGENDA



/01



/ACELERANDO DESENVOLVIMENTO E A GESTÃO DE INFRAESTRUTURA COMO CODIGO

Automatizando sua infraestrutura com o Terraform

/02



/ESTRATÉGIAS PARA TRABALHAR COM MULTIPLOS AMBIENTES

Aumentando a flexibilidade de uso do Terraform

/03



/ORGANIZANDO SEU CÓDIGO

Organizando seus ambientes de forma inteligente

/04



/COMPARANDO AS ESTRATÉGIAS

Escolhendo a abordagem mais adequada para o seu projeto





/ACELERANDO DESENVOLVIMENTO E A GESTÃO DE INFRAESTRUTURA COMO CODIGO





/O QUE É O TERRAFORM?



O Terraform é uma ferramenta de infraestrutura como código que permite criar, alterar e versionar a infraestrutura de maneira segura e eficiente.

- O Terraform é uma ferramenta open-source que permite criar e gerenciar a infraestrutura de maneira declarativa.
- Utiliza a linguagem HashiCorp Configuration Language (HCL) para descrever a infraestrutura desejada.
- É compatível com diversos provedores de nuvem e outros sistemas de infraestrutura.





/TERRAFORM WORKFLOW



/WRITE

Define a infraestrutura em arquivos de configuração



/PLAN

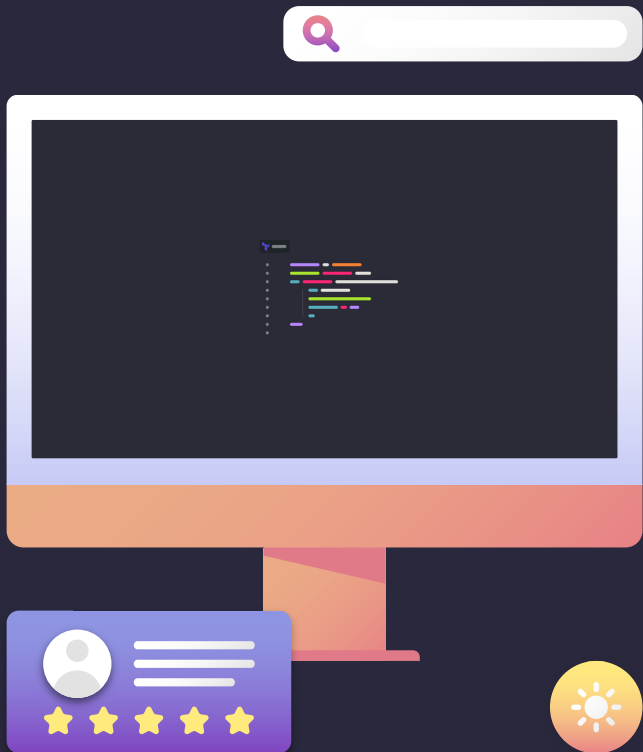
Revisa as mudanças que o Terraform irá aplicar em sua infraestrutura



/APPLY

Terraform provisiona sua infraestrutura e atualiza arquivo de estado





/DON'T REPEAT YOURSELF

Elimine a repetição de código com
o Terraform!



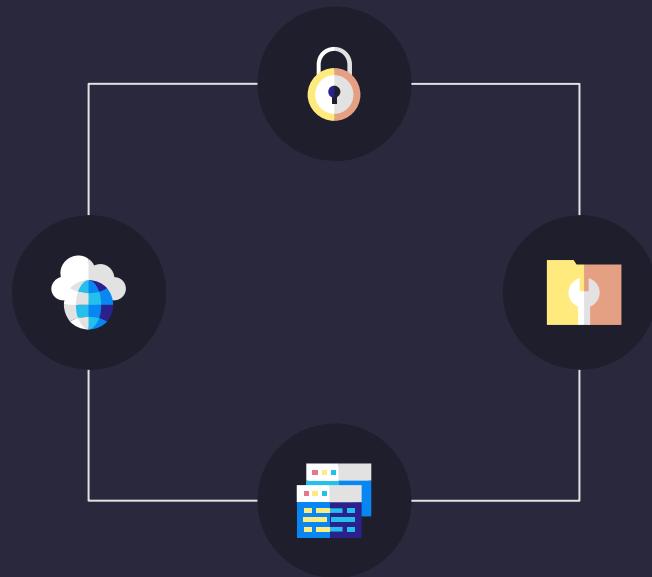
/CÓDIGO SIMPLIFICADO E REUTILIZÁVEL

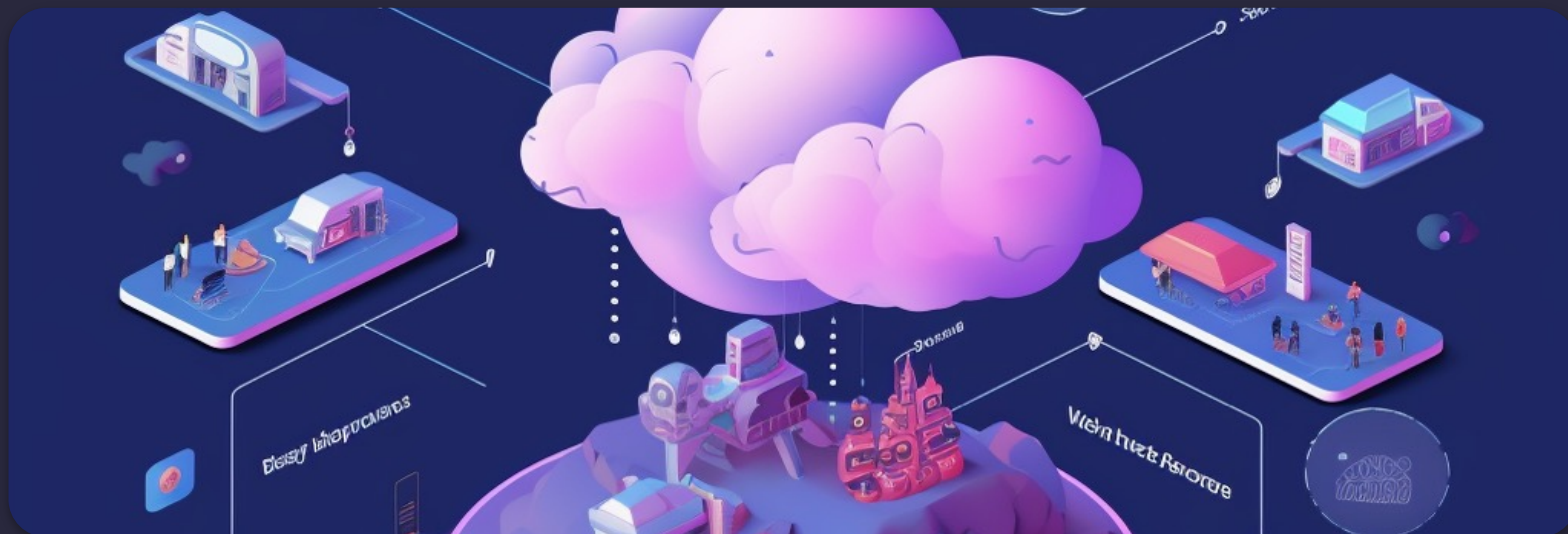


O princípio "**Don't Repeat Yourself**" (**DRY**) prega eliminar a duplicação de código.

O Terraform permite a criação de módulos reutilizáveis que podem ser compartilhados entre diferentes projetos e ambientes.

Dessa forma, evita-se a repetição de código e facilita-se a manutenção e atualização da infraestrutura.





/ESTRATÉGIAS PARA TRABALHAR COM MULTIPLOS AMBIENTES





/FLEXIBILIDADE

Ao trabalhar com múltiplos ambientes, é importante manter a separação e consistência das configurações.

O Terraform oferece estratégias para gerenciar esses ambientes, como o uso de diretórios ou workspaces.



/ESTRATÉGIAS



/POR DIRETÓRIOS

- A estratégia por diretórios envolve a criação de pastas separadas para cada ambiente (dev, prod, staging).
- Cada pasta contém os arquivos de configuração específicos para o respectivo ambiente.
- Essa abordagem facilita a organização do código e permite reutilizar módulos entre diferentes ambientes.



/POR WORKSPACES

- A estratégia por workspaces envolve o uso de workspaces do Terraform para separar as configurações de diferentes ambientes.
- Cada workspace possui seu próprio estado, permitindo a alteração independente das configurações.
- É possível alternar facilmente entre os workspaces para provisionar e gerenciar diferentes ambientes.



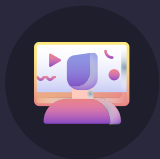


/ORGANIZANDO SEU CÓDIGO





/ORGANIZAÇÃO EFICIENTE



/ESTRUTURA

A organização do código por ambientes envolve a criação de estruturas de diretórios ou workspaces, conforme discutido anteriormente.



/MÓDULOS

É recomendado o uso de módulos para compartilhar configurações comuns entre os ambientes. Essa abordagem facilita a manutenção, atualização e colaboração no código do Terraform.



/SEPARE SEUS AMBIENTES COM DIRETÓRIOS



Na estratégia de diretórios, cada ambiente possui sua própria pasta com os arquivos de configuração correspondentes.




Essa abordagem permite uma separação das configurações e facilita a reutilização de código entre diferentes ambientes.



Cada pasta pode conter os módulos, variáveis e arquivos de configuração específicos para o respectivo ambiente.

Essa estratégia é simples e direta: você cria um diretório para cada ambiente e mantém os arquivos de configuração do Terraform separados. Cada ambiente tem seus próprios arquivos de configuração, o que significa que você pode ajustar as configurações de cada ambiente de acordo com suas necessidades.

> Com essa estratégia, podemos ter uma configuração específica para cada ambiente, sem precisar repetir o mesmo código várias vezes. Usando variáveis do Terraform, podemos configurar as diferenças entre cada ambiente.



```
1 |— environments
2 |   |— development
3 |   |   |— main.tf
4 |   |   |— variables.tf
5 |   |   |— outputs.tf
6 |   |— staging
7 |   |   |— main.tf
8 |   |   |— variables.tf
9 |   |   |— outputs.tf
10 |  |— production
11 |  |   |— main.tf
12 |  |   |— variables.tf
13 |  |   |— outputs.tf
14 |— modules
15 |   |— ...
16 |— providers
17 |   |— ...
18 |— variables.tf
19 |— outputs.tf
20 |— main.tf
21 |— terraform.tfstate
```



/DEV.TF



```
# environments/development/main.tf

module "azure_vm" {
  source = "../../modules/azure/vm"
  vm_name = "dev-vm"
  vm_size = "Standard_B2s"
  os_disk_size_gb = "30"
  admin_username = var.admin_username
  admin_password = var.admin_password
  resource_group_name = "dev-rg"
  location = "eastus"
}
```





/STAGING.TF



```
# environments/development/main.tf

module "azure_vm" {
  source = "../../modules/azure/vm"
  vm_name = "stg-vm"
  vm_size = "Standard_B2s"
  os_disk_size_gb = "30"
  admin_username = var.admin_username
  admin_password = var.admin_password
  resource_group_name = "stg-rg"
  location = "eastus"
}
```





/GERENCIE AMBIENTES COM WORKSPACES



A estratégia de workspaces utiliza os recursos de workspaces do Terraform para gerenciar diferentes ambientes.



Cada workspace contém seu próprio estado e configurações, permitindo a alteração independente de cada ambiente.



Essa abordagem é útil quando os ambientes compartilham grande parte das configurações, com variações mínimas.





Essa estratégia envolve o uso de **workspaces** no **Terraform** para manter o código para cada ambiente em um único diretório.

Um **workspace** é uma instância isolada de um conjunto de recursos no **Terraform**.

Cada **workspace** tem seu próprio estado.

Por padrão, o Terraform tem um único workspace chamado “**default**”, mas você pode criar novos workspaces para gerenciar recursos em diferentes ambientes.



```
1 | environments
2 |   | development
3 |   |   | dev.tfvars
4 |   | staging
5 |   |   | stage.tfvars
6 |   | production
7 |   |   | prod.tfvars
8 | modules
9 |   | ...
10 | providers
11 |   | ...
12 | variables.tf
13 | outputs.tf
14 | main.tf
15 | terraform.tfstate
```





/BACKEND



```
terraform {  
  backend "azurerm" {  
    resource_group_name = "tfstate-rg"  
    storage_account_name = "tfstateacc"  
    container_name = "tfstate"  
    key = "terraform.tfstate"  
  }  
}
```

```
$ terraform workspace new dev  
$ terraform workspace select dev
```





/MAIN.TF



```
# main.tf
```

```
resource "azurerm_resource_group" "rg" {  
  name = "rg-${terraform.workspace}"  
  location = "eastus"  
}
```

```
$ terraform workspace select prod  
$ terraform plan -var-file=/environments/production/prod.tfvars
```





/COMPARANDO AS ESTRATÉGIAS



/RAIO-X

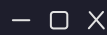
Ao escolher entre as estratégias de diretórios e workspaces, é importante considerar as necessidades e características do projeto.

Cada abordagem possui vantagens e desvantagens, e a escolha dependerá dos requisitos e preferências da equipe.





YOUR LOGO HERE



/LADO A LADO



```
1 |— environments
2 |   |— development
3 |   |   |— main.tf
4 |   |   |— variables.tf
5 |   |   |— outputs.tf
6 |   |— staging
7 |   |   |— main.tf
8 |   |   |— variables.tf
9 |   |   |— outputs.tf
10 |   |— production
11 |   |   |— main.tf
12 |   |   |— variables.tf
13 |   |   |— outputs.tf
14 |— modules
15 |   |— ...
16 |— providers
17 |   |— ...
18 |— variables.tf
19 |— outputs.tf
20 |— main.tf
21 |— terraform.tfstate
```



```
1 |— environments
2 |   |— development
3 |   |   |— dev.tfvars
4 |   |— staging
5 |   |   |— stage.tfvars
6 |   |— production
7 |   |   |— prod.tfvars
8 |— modules
9 |   |— ...
10 |— providers
11 |   |— ...
12 |— variables.tf
13 |— outputs.tf
14 |— main.tf
15 |— terraform.tfstate
```



COMPARE.TF



/DIRETÓRIOS



/PRÓS



- > Cada ambiente tem sua propria configuração
- > Pode ser ajustado de acordo com cada necessidade
- > É fácil visualizar e gerenciar as configurações de cada ambiente

/CONTRAS



- > Pode ser tedioso manter vários arquivos separados
- > É fácil esquecer de aplicar uma alteração em todos os ambiente
- > Requer duplicação de arquivos em cada ambiente



/WORKSPACES



/PRÓS



- > Mais fácil de gerenciar muitos ambientes
- > Usa uma única fonte de verdade para o código, não há duplicação
- > Permite que você crie instâncias separadas, a alternância entre os workspaces é fácil

/CONTRAS



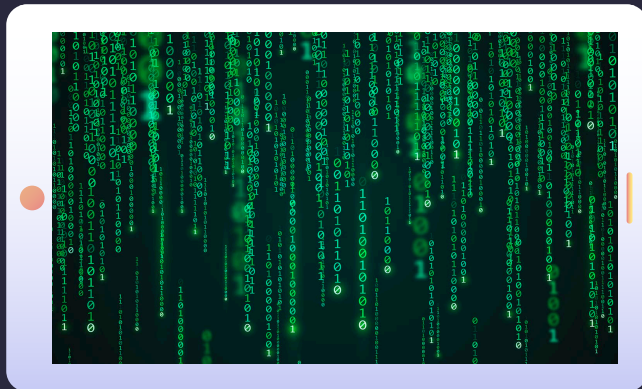
- > Requer mais cuidado para evitar conflitos
- > A configuração compartilhada precisa ser gerenciada com cuidado para garantir que seja aplicada em todos os workspaces

/CONCLUSÃO

Em geral, a estratégia de separação por **workspaces** é mais **flexível e escalável** para projetos maiores.

No entanto, a estratégia de separação por **diretórios** pode ser mais fácil de implementar e entender para projetos menores ou menos complexos.

É importante escolher a estratégia que melhor atenda às necessidades específicas do seu projeto. Em alguns casos, pode até ser possível usar uma combinação de ambas as estratégias para atender às suas necessidades.





/OBRIGADO!

/PERGUNTAS?

asilva@unicast.com.br
github.com/asilvajunior
unicast.com.br



> Avalie esta apresentação, acesse o repositório e baixe o conteúdo apresentado!

