Terraform **Provisioners**

@pedrosaxu
no youtube

Alguns itens que veremos...

- Como **executar scripts** no seu Terraform
- **Validar o deployment** de WebServers
- **Visualizar a execução** de scripts remotos
- **Pontos de atenção** sobre Provisioners

Introdução

- O Terraform usa:
  - **arquivos de configuração** para **definir a infraestrutura;**
  - e **provisioners** para **executar tarefas adicionais** em recursos.

Documentação oficial sobre Provisioners

Ponto de atenção!

- A Hashicorp indica o uso de provisioners **somente em último caso**;

> **Important:** Use provisioners as a last resort. There are better alternatives for most situations. Refer to Declaring Provisioners for more details.

- O indicado principalmente para execuções remotas **é a utilização dos recursos próprios da Cloud** para execução de scripts de inicialização, como o “custom_data”, por exemplo

- Muitas vezes há até mesmo um provider (por mais que não oficial) que permite a execução de scripts de modo gerenciado, podendo ter inclusive a gerência de estado daquele script;

- Gosto de usar os provisioners para **tarefas simples e de baixa complexidade**.

@pedrosaxu no youtube

Documentação oficial sobre Provisioners

O que são provisioners?

- Os provisioners são usados no Terraform para executar tarefas adicionais em recursos provisionados, como:
  - a instalação de software;
  - configuração de arquivos;
  - validação de comunicação com recursos..

- Os provisioners são definidos em blocos de configuração "**provisioner**" e "**connection**"

- Por padrão, provisioners **são executados somente durante a criação e destruição** de um recurso, não executando em nenhuma execução de “update-in-place”, por exemplo.

Documentação oficial sobre Provisioners

Tipos de provisioners:

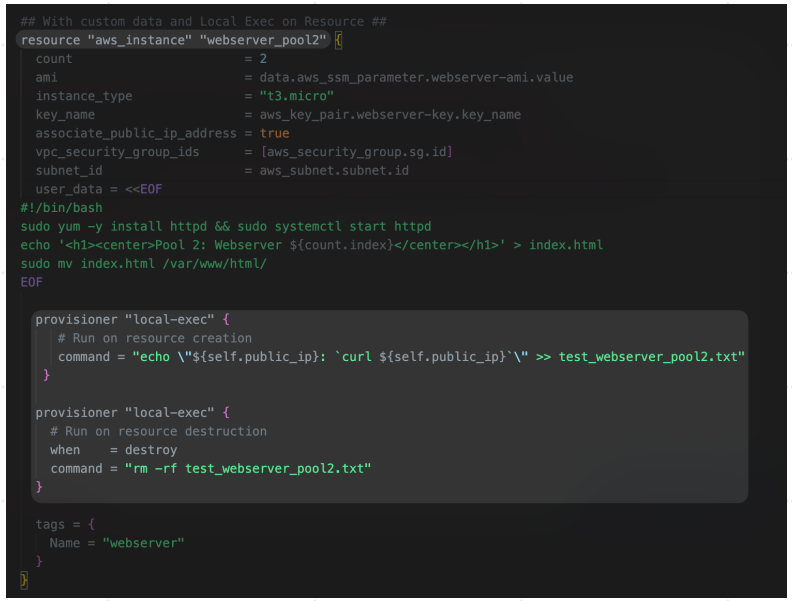
- O Terraform suporta diferentes tipos de provisioners, incluindo:

  - **local-exec**: executar comandos localmente ;
  - **remote-exec**: executar comandos remotamente (via SSH ou WinRM);
  - **file**: copiar arquivos para uma máquina remota.

Documentação oficial sobre Provisioners

Formas de declarar provisioners:

- Você pode declarar provisioners dentro do bloco de um recurso, e acioná-lo **somente na criação ou destruição** deste recurso:



```
## With custom data and Local Exec on Resource ##
resource "aws_instance" "webserver_pool2" {
  count                       = 2
  ami                         = data.aws_ssm_parameter.webserver-ami.value
  instance_type               = "t3.micro"
  key_name                    = aws_key_pair.webserver-key.key_name
  associate_public_ip_address = true
  vpc_security_group_ids      = [aws_security_group.sg.id]
  subnet_id                   = aws_subnet.subnet.id
  user_data = <<EOF
#!/bin/bash
sudo yum -y install httpd && sudo systemctl start httpd
echo '<h1><center>Pool 2: Webserver ${count.index}</center></h1>' > index.html
sudo mv index.html /var/www/html/
EOF

  provisioner "local-exec" {
    # Run on resource creation
    command = "echo \"${self.public_ip}: `curl ${self.public_ip}`\" >> test_webserver_pool2.txt"
  }

  provisioner "local-exec" {
    # Run on resource destruction
    when    = destroy
    command = "rm -rf test_webserver_pool2.txt"
  }

  tags = {
    Name = "webserver"
  }
}
```

- Você pode declarar provisioners dentro de um bloco próprio, utilizando o "**null_resource**", podendo acioná-lo com base no atributo que quiser, com o **trigger**:

```
resource "null_resource" "script" {
  # Força a recriação/execução do provisioner a cada execução do terraform
  triggers = { uuid = uuid() }

  for_each = toset(aws_instance.webserver_pool3.*.public_ip)

  provisioner "local-exec" {
     # Run on resource creation
     command = "echo \"${each.value}: `curl ${each.value}`\" >> test_webserver_pool.txt"
  }

  provisioner "local-exec" {
    # Run on resource destruction
    when    = destroy
    command = "rm -rf test_webserver_pool.txt"
  }
}
```

@pedrosaxu no youtube

[Documentação oficial sobre Provisioners]

Dependências de Provisioners:

- Quando vários provisioners são usados em um único recurso, é importante **gerenciar as dependências** entre eles corretamente para evitar erros ou comportamentos inesperados;
- Ao usar um provisioner dentro de um recurso existente, você deve referenciar atributos deste recurso usando o comando **$self.atributo.**

```
provisioner "local-exec" {
    command = "echo \"${self.public_ip}: `curl ${self.public_ip}`\" >> test_webserver_pool1.txt"
}
```

- Ao usar um provisioner em um recurso apartado, é necessário **utilizar-se das dependências implícitas/explícitas** para garantir o bom funcionamento dos recursos.
  - Isso pode ser feito usando a opção "depends_on", ou simplesmente referenciando os atributos do outro recurso no bloco "null_resource"

```
for_each = toset(aws_instance.webserver_pool3.*.public_ip)

provisioner "local-exec" {
    # Run on resource creation
    command = "echo \"${each.value}: `curl ${each.value}`\" >> test_webserver_pool.txt"
}
```

@pedrosaxu no youtube

[Documentação oficial sobre Provisioners]

Terraform
Provisioners

@pedrosaxu
no youtube