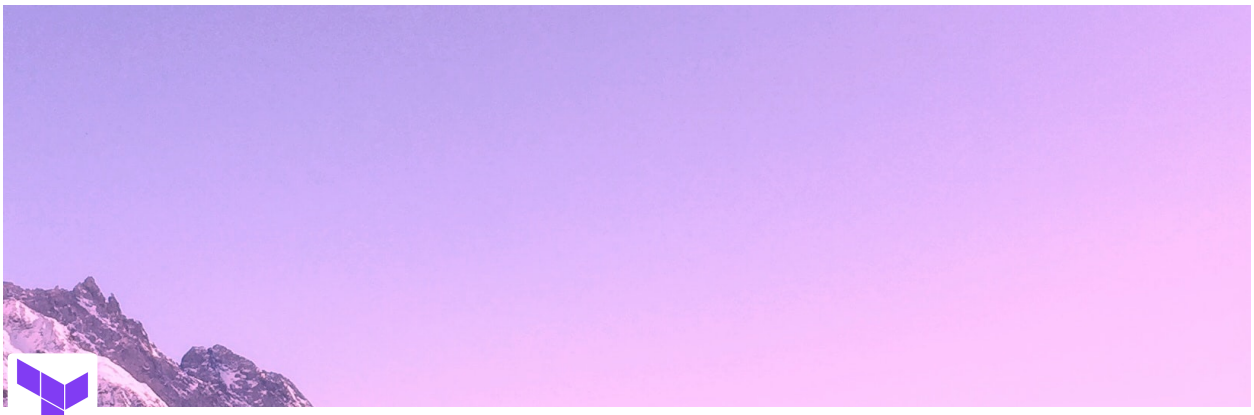


Terraform

Criando infraestrutura como código utilizando Terragrunt e Atlantis na AWS - Blog da Compass

Este artigo será dividido em duas partes, no primeiro artigo vamos falar mais sobre o Terragrunt e Atlantis, sobre como estas ferramentas podem auxiliar a adoção de uma cultura DevOps na sua empresa, vamos mostrar os passos necessários para a sua instalação e integração com um repositório do Github, e na

https://blog.compass.uol/autores/criando-infraestrutura-como-codigo-utilizando-terragrunt-e-atlantis-na-aws/

▼ **TERRAFORM - Alura**

- https://www.terraform.io/
- Ferramenta multiplataforma
- Serve para administrar a infraestrutura cloud, tanto para azure, aws, gcp, etc.
- Irá provisionar o provedor de cloud

INSTALAÇÃO

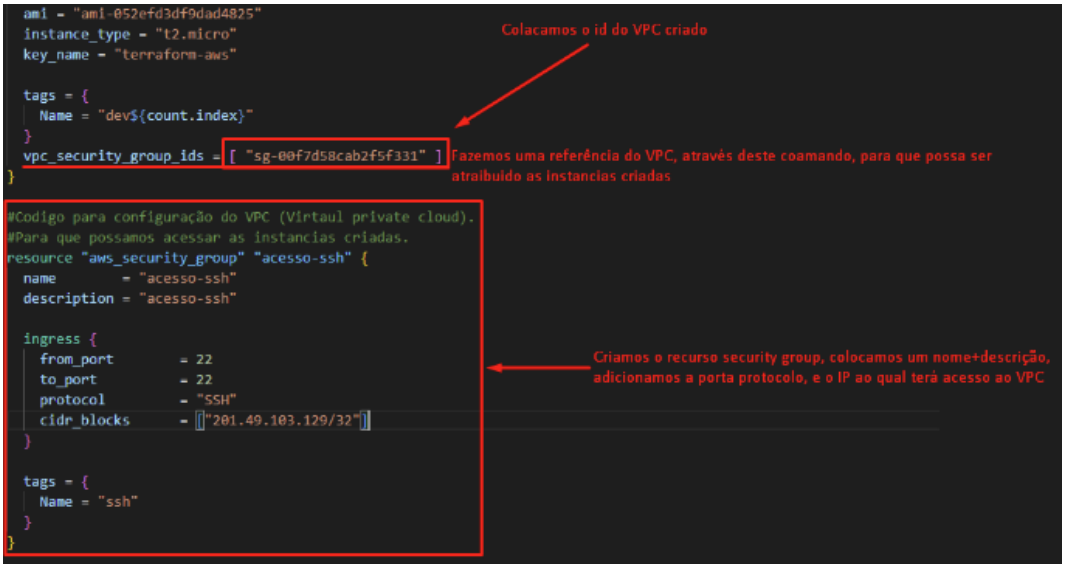
- LINUX-UBUNTU
  - Rodar os seguintes comandos :
    - `curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -`
    - `sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"`
    - `sudo apt-get update && sudo apt-get install terraform`
    - Se necessario instalar "curl"
- WINDOWS
  - Baixar o arquivo em zip
  - Criar um pasta no diretorio C: chamada terraform

- Crie a variavel de ambiente chamada Path – C:\terraform

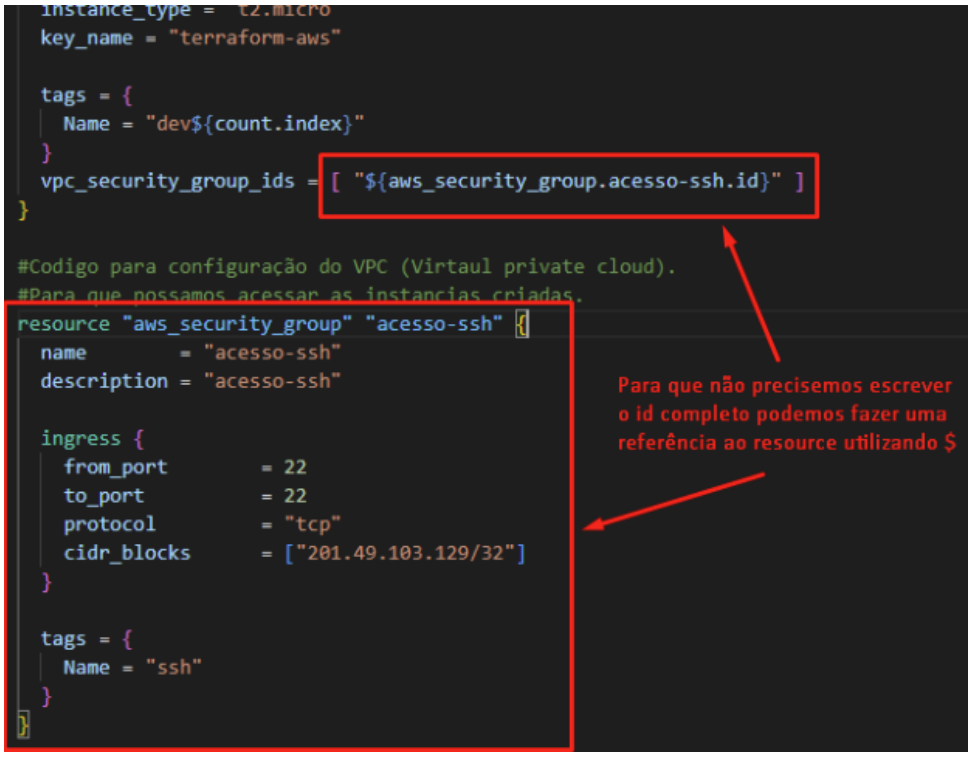
CONFIGURANDO A INFRAESTRUTURA

- Os arquivos de configuração podem ter quaisquer nomes, desde que estejam no diretório correto e possuam a extensão .tf.

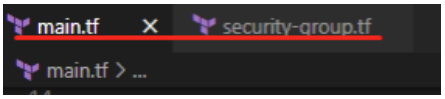
- Para buscas de tamplets google_projects
- TAGS – servem para das nome ao recurso ou grupo de recursos
- Podemos criar varios recursos de uma vez usando um comando
  - No caso "count
- "apply" acionado ao criar os recusrsos
- ***Na primeira vez utilizamos o init, depois o plan na sequência (mostra todas as alterações, mas sem aplicá-las) e apply para efetivar as mudanças.***
- Mesmo não sendo recomendável, o Terraform permite que você utilize diretamente o apply.
- O comando show lê o arquivo terraform.tfstate e exibe as informações associadas (IP, rede, etc).


ami = "ami-052efd3df9dad4825"
instance_type = "t2.micro"
key_name = "terraform-aws"
tags = {
Name = "dev${count.index}"
}
vpc_security_group_ids = ["sg-00f7d58cab2f5f331"]
}
Colacamos o id do VPC criado
Fazemos uma referência do VPC, através deste coamando, para que possa ser atraibuido as instancias criadas
#Codigo para configuração do VPC (Virtaul private cloud).
#Para que possamos acessar as instancias criadas.
resource "aws_security_group" "acesso-ssh" {
name = "acesso-ssh"
description = "acesso-ssh"
ingress {
from_port = 22
to_port = 22
protocol = "SSH"
cidr_blocks = ["201.49.103.129/32"]
}
tags = {
Name = "ssh"
}
}
Criamos o recurso security group, colocamos um nome+descrição, adicionamos a porta protocolo, e o IP ao qual terá acesso ao VPC


terraform {
OBJETO OBRIGATORIO
required_providers {
google = {
source = "hashicorp/google"
version = "4.27.0"
}
}
}
Especificações do tipo de provider e os requisitos
provider "google"{
credentials =
project = "my-project-id"
region = "us-central1"
zone = "us-central1-c"
}
Informações do provider com informações de crendenciais, tipo de projeto (no caso do gcp) região, e zona.
resource "google_compute_network" vpc_network {
name= "terraform-network"
}
Recurso ou grupo de recursos.



- Podemos separar os arquivos dentro do terraform, contanto que os arquivos estejam na mesma pasta e com o .tf



- Podemos deixar as instancias principais como VMs Ec2, Buckets S3, DataBase colocamos no main
- Security groups, e variaveis colocamos separados
- A configuração pode ser "quebrada" em mais de um arquivo, desde que esteja na mesma estrutura de diretório.
- O Terraform permite que você concentre suas configurações em um único arquivo.

- Para um mesmo provedor, utilizamos o **alias** para referenciar os recursos de outra região.

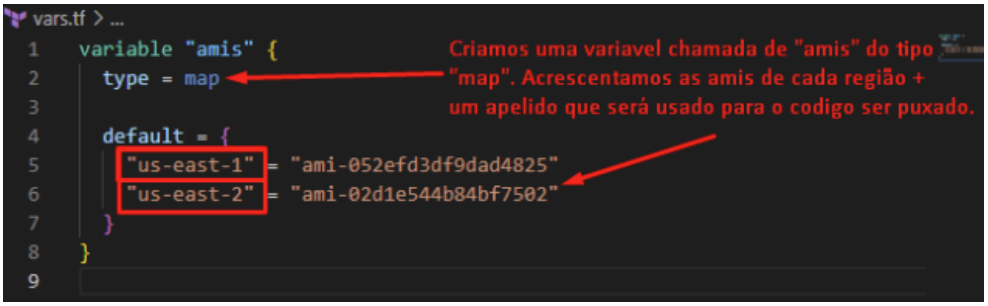
```
# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

provider "aws" {
  alias = "us-east-2"
  region = "us-east-2"
}
```

- E quando criarmos instancias/recursos de outras regiões, precisamos adicionar o provider, e em caso de security groups precisamos criar um novo para aquela região

```
resource "aws_instance" "dev6" {
  provider = aws.us-east-2
  ami = "ami-02d1e544b84bf7502"
  instance_type = "t2.micro"
  key_name = "terraform-aws"

  tags = {
    Name = "dev6"
  }
  vpc_security_group_ids = [ "${aws_security_group.acesso-ssh-us-east-2.id}" ]
}
```

```
resource "aws_security_group" "acesso-ssh" {
  name        = "acesso-ssh"
  description = "acesso-ssh"

  ingress {
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = ["201.49.103.129/32"]
  }

  tags = {
    Name = "ssh"
  }
}

resource "aws_security_group" "acesso-ssh-us-east-2" {
  provider = aws.us-east-2
  name        = "acesso-ssh"
  description = "acesso-ssh"

  ingress {
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = ["201.49.103.129/32"]
  }

  tags = {
    Name = "ssh"
  }
}
```

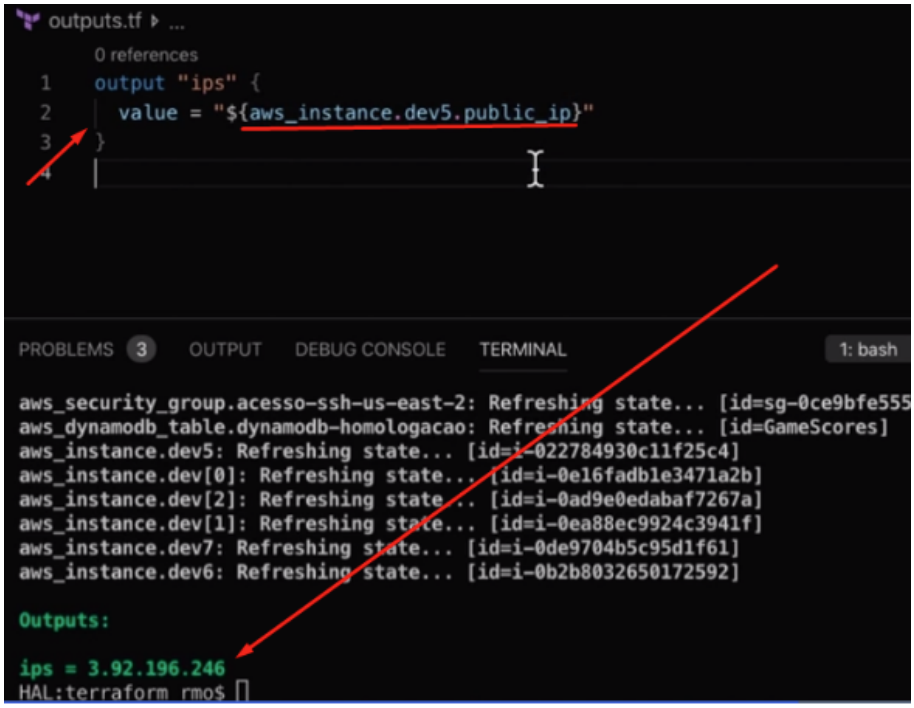
- Utilizamos a tag provider, seguida do alias que desejamos utilizar.
- Com as variáveis podemos puxar um arquivo através do deixar o codigo mais limpo



```
#us-east-1, criação de 3 vms dev
resource "aws_instance" "dev" {
  count = 3
  ami = var.amis["us-east-1"]
  instance_type = "t2.micro"
  key_name = "terraform-aws"

  tags = {
    Name = "dev${count.index}"
  }
  vpc_security_group_ids = [ "${aws_security_group.acesso-ssh.id}" ]
}
```

No codigo ficara da seguinte forma

- Para a remoção dos recursos podemos retirar no proprio texto do codigo e dar um **terraform plan** e **apply** ou então podemos colocas **terraform destroy –target RECURSO**
- Para remover os recursos coodependentes removes o recurso dependente para para que seja excluido os dois ao mesmo tempo
- Caso formos remover tudo utilizamos o comando **terraform destroy**
- Tanto removendo as configurações dos recursos no arquivo de configuração, como também utilizando o comando **terraform destroy.**
- Os outputs trazem uma informação devolta, como no caso



- São declaradas com a sintaxe:
- output "nome" { value = "${referência}" }
- a principal vantagem de utilizar o backend remoto, o que permite inclusive a administração da infraestrutura por mais de uma pessoa.

- É necessário criar alguns arquivos adicionais para utilizar o backend remoto.