DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Atividade obrigatória e em grupo

Exercício Terraform - Parte 2

Continuando com nossa prática, vamos realizar a mesma estrutura de arquivos, mas desta vez faremos em modo de grupo. Para criar nossas instâncias EC2, uma dentro do grupo de segurança pública e outra no privado (criado na classe anterior). Os arquivos a serem criados são:

- main.tf
- variables.tf
- output.tf

No caso de **variables.tf**, vamos criar as mesmas variáveis VPC e, além disso, definir as variáveis necessárias para usar recursos dentro do VPC - vpc ID e grupos de segurança.

Dentro de **outputs.tf** vamos definir a saída. Como na prática anterior, cabe ao desenvolvedor quais informações ele espera exibir.

O conteúdo do **main.tf grupo de** é composto por:

- Definição de duas instâncias do EC2 —uma para cada segurança— e que obtêm automaticamente o ID da AMI correspondente de acordo com a região.

Ao final, geramos os mesmos três arquivos, mas para utilizar os dois módulos criados durante essas duas aulas.

Por se tratar de um exercício em grupo, a recomendação é que as tarefas sejam divididas para poder aproveitar o tempo e juntar o código para realizar uma integral final de toda a automação nos últimos 10 minutos.

Na próxima página você encontrará a resolução. Continue apenas para autoavaliação.

Resolução

Vamos lembrar que os quatro comandos para executar as tarefas são:

- terraform init
- terraform plan

- terraform aplicar
- terraform destruir

Ao executar o **terraform init**, vamos inicializar nosso código e baixar as dependências necessárias:

```
Initializing modules...
- ec2 in modulos/ec2
- vpc in modulos/vpc
Downloading terraform-aws-modules/vpc/aws 3.6.0 for vpc.vpc...
- vpc.vpc in .terraform/modules/vpc.vpc

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching ">= 3.28.0"...
- Installing hashicorp/aws v3.55.0...
- Installed hashicorp/aws v3.55.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Em seguida, executamos o **terraform plan** seguido de **terraform apply** para aplicar nossas mudanças. No final, vamos destruir os recursos criados com **terraform destroy**. A saída desses comandos é a seguinte:

```
var.namespace
  Enter a value: digitalhouse


Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
```

```
Plan: 20 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + ip_publica = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

```
Apply complete! Resources: 20 added, 0 changed, 0 destroyed.

Outputs:

ip_publica = "52.53.178.62"
```

Em nosso console AWS, visualizamos as instâncias criadas.

Por fim, destruímos os recursos para não gerar custos extras:

```
Plan: 0 to add, 0 to change, 20 to destroy.

Changes to Outputs:
  - ip_publica = "52.53.178.62" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```

A seguir, descreveremos o código resolvido. Dentro da pasta EC2, criamos os seguintes arquivos:

```
# main.tf

data "aws_ami" "amazon-linux-2" {

 most_recent = true

filter {

   name   = "owner-alias"

   values = "amazon”
```

```
 }
 filter {
   name   = "name"
   values = ["amzn2-ami-hvm *"]
 }
}


resource "aws_instance" "ec2_public" {
 ami                         = data.aws_ami.amazon-linux-2.id
 associate_public_ip_address = true
 instance_type               = "t2.micro"
 subnet_id                   = var.vpc.public_subnets[0]
 vpc_security_group_ids      = [var.sg_pub_id]


 tags = {
   "Nome" = "${var.namespace}-EC2-PUBLIC"
 }
}


resource "aws_instance" "ec2_private" {
 ami                         = data.aws_ami.amazon-linux-2.id
 associate_public_ip_address = false
 instance_type               = "t2.micro"
```

```
 subnet_id                   = var.vpc.private_subnets [1]

 vpc_security_group_ids      = [var.sg_priv_id]


 tags = {

   "Nome" = "${var.namespace}-EC2-PRIVATE"

 }


}
```

```
# outputs.tf

output "public_ip" {

 value = aws_instance.ec2_public.public_ip

}
```

```
#variables

variable "namespace" {

 type = string

}


variable "vpc" {

 type = any

}
```

```
variable "sg_pub_id" {

 type = any

}


variable "sg_priv_id" {

 type = any

}
```

Temos um diretório principal e — nele— pastas que chamamos de “módulo” e a automação que executamos. Vamos criar um **main.tf** cuja única funcionalidade é usar o que está dentro dos módulos:

```
# main.tf


provider "aws" {

 region = var.regiao

}


module "vpc" {

 source    = "./module/vpc"

 namespace = var.namespace

}


module "ec2" {
```

```
 source      = "./modulo/ec2"

 namespace   = var.namespace

 vpc         = module.vpc.vpc

 sg_pub_id   = module.vpc.sg_pub_id

 sg_priv_id = module.vpc.sg_priv_id

}
```

```
# outputs.tf

output "ip_publica" {

 value = "$ {module.ec2.public_ip}"

}
```

```
# variables.tf

variable "namespace" {

 type         = string

}



variable "regiao" {

 default      = "us-east-1"

 type         = string

}
```

Para esclarecer a ordem dos arquivos e junto com nosso módulo vpc criado na aula 8, devemos ter uma organização como esta:

```
$ tree

.

├── main.tf

├── modulos

│ ├── ec2

│ │ ├── main.tf

│ │ ├── outputs.tf

│ │ └── variables.tf

│ └── vpc

│ ├── main. tf

│ ├── output.tf

│ └── variables.tf

├── outputs.tf

└── variables.tf


3 diretórios, 10 arquivos
```