DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Atividade em grupo
Dificuldade: Alta

Exercício Terraform - Parte 2

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

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

No caso de variables.tf, vamos criar as mesmas variáveis da VPC e, ainda, definir as variáveis necessárias para o uso de recursos dentro da VPC - ID da VPC e grupos de segurança.

Dentro de **outputs.tf**, vamos definir a saída. Como na prática anterior, as informações a serem visualizadas ficam a critério do programador.

O conteúdo de **main.tf** é composto de:

- Definição de duas instâncias EC2 — uma para cada grupo de segurança — e que obtenha, automaticamente, o ID da AMI correspondente à região.

Ao finalizar, vamos gerar os mesmos três arquivos mas, desta vez, para utilizar os dois módulos criados durante essas duas aulas.

Sendo uma execução em grupos, é recomendável que as tarefas sejam divididas, para aproveitar o tempo e unir o código e, desta forma, realizar uma última integral de toda a automação nos últimos 10 minutos.

Na página a seguir está a resolução. Continue somente para realizar uma autoavaliação.

Resolução

Lembrando que os quatro comandos utilizados para a execução das tarefas são:

- terraform init
- terraform plan
- terraform apply
- terraform destroy

Ao executar **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.
```

Após isso, executamos **terraform plan** e, em seguida, **terraform apply** para aplicar nossas alterações. Ao finalizar, vamos usar **terraform destroy** para destruir os recursos.

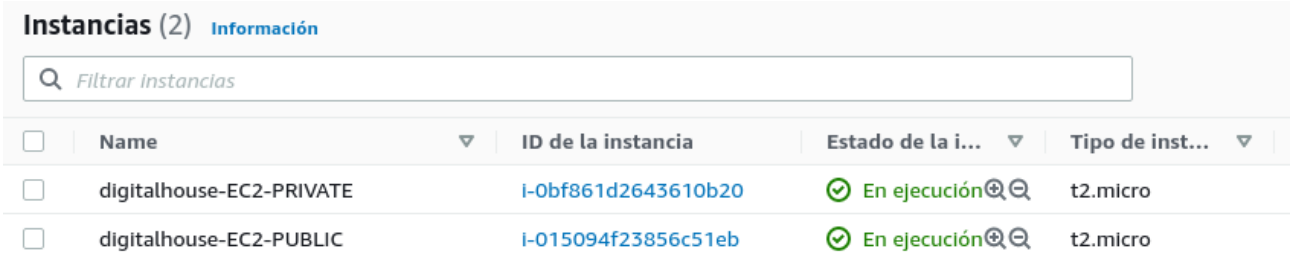
A saída desses comandos é a seguinte:

```
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"
```

Dentro de nosso console AWS, visualizamos as instâncias criadas:



Para concluir, destruímos os recursos para não gerar custos adicionais:

```
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
```

Após isso, vamos escrever o código resolvido. Dentro da pasta EC2, criamos os seguintes arquivos:

```
# main.tf

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

 most_recent = true

 owners      = ["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 = {
   "Name" = "${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 = {
   "Name" = "${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

}
```

Dispomos de um diretório principal e - dentro dele - pastas que chamamos de “module” 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.region
}


module "vpc" {
 source    = "./module/vpc"
 namespace = var.namespace
}


module "ec2" {
 source     = "./module/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 "region" {

 default      = "us-west-1"

 type         = string

}
```

Para indicar a ordem dos arquivos, e junto ao nosso módulo da vpc criado na Aula 8, devemos ter uma organização como a seguinte:

```
$ tree

.

├── main.tf

├── module

│   ├── ec2

│   │   ├── main.tf

│   │   ├── outputs.tf

│   │   └── variables.tf

│   └── vpc

│       ├── main.tf

│       ├── output.tf

│       └── variables.tf

├── outputs.tf

└── variables.tf


3 directories, 10 files
```