DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Infraestructura II

Actividad obligatoria y grupal
Dificultad: Alta

Ejercitación Terraform - Parte 2

Continuando con nuestra práctica, vamos a realizar la misma estructura de archivos, pero esta vez lo vamos a realizar en una modalidad grupal. Para crear nuestras instancias EC2, una dentro del grupo de seguridad pública y el otro en el privado (creados en la clase anterior). Los archivos a crear son:

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

En el caso de **variables.tf**, vamos a crear las mismas variables de VPC y, además, definir las variables necesarias para utilizar recursos dentro de la VPC —ID de vpc y grupos de seguridad—.

Dentro de **outputs.tf** vamos a definir la salida. Al igual que la práctica anterior, queda a decisión del desarrollador qué información espera visualizar.

El contenido de **main.tf** se compone de:

- Definición de dos instancias EC2 —una para cada grupo de seguridad— y que obtenga, automáticamente, el ID de la AMI que corresponda según la región.

Al finalizar, vamos a generar los mismos tres archivos, pero para utilizar los dos módulos creados durante estas dos clases.

Al ser una ejercitación grupal, la recomendación es que se dividan las tareas para poder aprovechar el tiempo y unir el código para realizar una última integral de toda la automatización en los últimos 10 minutos.

En la siguiente página se encuentra la resolución. Continúa únicamente para realizar una autoevaluación.

Resolución

Recordemos que los cuatro comandos para ejecutar las tareas son:

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

Al ejecutar **terraform init**, vamos a inicializar nuestro código y descargar las dependencias necesarias:

```
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.
```

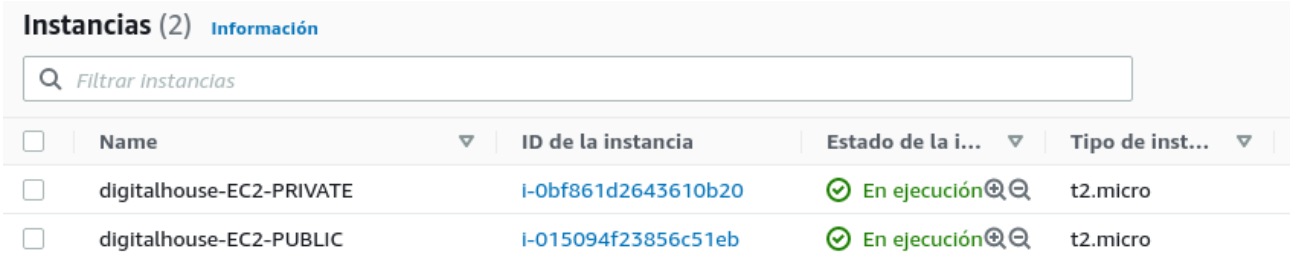
Luego, ejecutamos **terraform plan** seguido de **terraform apply** para aplicar nuestros cambios. Al finalizar, vamos a destruir los recursos creados con **terraform destroy**. La salida de estos comandos es la siguiente:

```
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 nuestra consola de AWS, visualizamos las instancias creadas:



Para finalizar, destruimos los recursos para no generar costos 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 continuación, vamos a describir el código resuelto. Dentro de la carpeta EC2, creamos los siguientes archivos:

```
# 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

}
```

Contamos con un directorio principal y —dentro de él— carpetas a las que llamamos "module" y la automatización que ejecutamos. Vamos a crear un **main.tf** cuya única funcionalidad es usar lo que está dentro de los 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 aclarar el orden de los archivos y junto con nuestro módulo de vpc creado en la Clase 8, debemos tener una organización igual a esta:

```
$ 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
```