Certified Tech Developer
The Ultimate Degree

Infraestructura II

¡Levantamos una infraestructura de la vida real!

Actividad: obligatoria

Dificultad: media

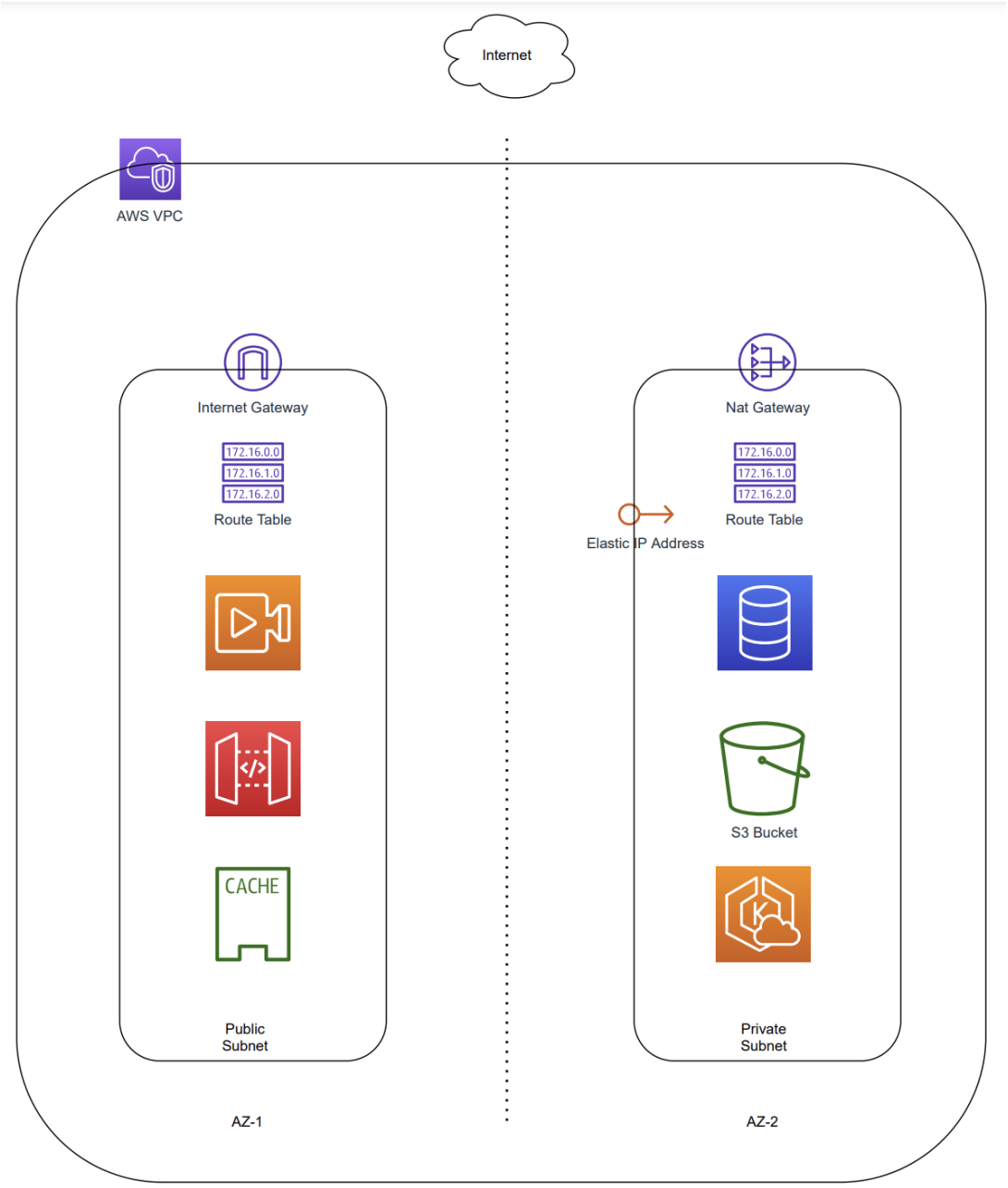
Nuestro objetivo es levantar una infraestructura cloud más robusta, segura y resiliente, que consista meramente en recursos de infraestructura:

- Un VPC.
- Un Internet Gateway asociado al VPC creado.
- Una subnet pública.
- Una subnet privada.
- Una tabla de ruteo dedicada a la subnet pública.
- Una tabla de ruteo dedicada a la subnet privada.
- Las asociaciones de ambas tablas de ruteo con sus respectivas subnets.

- Un NAT Gateway asociado a la subnet privada.
- Una Elastic IP.

Nuestro código será modularizado, es decir, segmentado en tres archivos: uno para variables, otro para seleccionar el proveedor Cloud y el último que se utilizará para levantar nuestra infra.

Esta lucirá de la siguiente manera:



Aunque, por razones de practicidad solo nos focalizamos en la infraestructura y no en los servicios que se levantaran dentro suyo.

Sugerencia: tengamos a mano el enlace a la documentación de Terraform ya que la iremos consultando a medida que avancemos.

Documentacion: https://registry.terraform.io/providers/hashicorp/aws/latest/docs

¡Manos a la obra!

Estructura de directorio y módulos

Para que cada módulo de Terraform que vamos a utilizar funcione debemos ubicarlos a todos en el mismo directorio.

Los archivos se llamarán:

- **main.tf** (servirá para levantar la infraestructura de todo mi VPC a levantar).
- **variables.tf** (contendrá las variables que quiero pasarle a cada modulo).
- **providers.tf** (servirá para definir que proveedor cloud y que versiones utilizar).

Cuando ejecutemos terraform init, lo primero que sucederá es la lectura del módulo, providers.tf, donde buscará que proveedor cloud utilizar.

Seguidamente, el orden es alfabético, es decir, irá ejecutando módulo a módulo según su la inicial del nombre de archivo o módulo.

Armando el ambiente en tres pasos:

1. Declaración de variables.
2. Declaración del provider a utilizar.
3. Creación de la infraestructura base.

1. **Declaración de variables.**

   Nombre de archivo: variables.tf

```
# ================================================================

# Proposito:    declaramos todas las variables que vamos a usar

# Autor:        DH

# Fecha:        30.07.21

# Version:      1.0

# ================================================================



variable "aws_region_id" {

   description = "la region"
```

```
  type        = string
  default = "us-east-1"
}
variable "main_vpc_cidr" {
  description = "Nuestro Security Group"
  type        = string
  default = "10.0.0.0/24"
}


variable "public_subnets" {
  description = "subnet con acceso a internet"
  type        = string
  default = "10.0.0.128/26"
}


variable "private_subnets" {
  description = "subnet sin acceso a internet"
```

```
  type        = string

  default = "10.0.0.192/26"

}

# ============================================================
```

2. **Declaración del provider a utilizar.**

Nombre de archivo: providers.tf

```
# ============================================================

# Propósito:    declaramos que proveedor cloud queremos usar

# Autor:        DH

# Fecha:        30.07.21

# Versión:      1.0

# ============================================================



# ============================================================

# Declaramos el Cloud Provider con el que queremos trabajar



terraform {
```

```
# Le decimos que queremos:

# a. la versión del binario de terraform mayor o igual a 0.12

 required_version = ">=0.12"

 required_providers {

   aws = {

# Especificamos desde donde queremos descargar el binario:

     source = "hashicorp/aws"

# Le decimos que solo permitirá:

# b. la versión del binario del provider 3.20.0 (con cierta restricción)

     version = "~> 3.20.0"

   }

 }

}

# ================================================================



# ================================================================

# Declaramos la región donde queremos levantar nuestra infra
```

```
provider "aws" {
 shared_credentials_file = "~/.aws/credentials"
 region = "us-east-1"
}
# ================================================================
```

3. **Creación de la infraestructura base.**

Nombre de archivo: Main.tf

```
# Propósito:    crear infraestructura AWS
# Autor:        DH
# Fecha:        30.07.21
# Versión:      1.0
# ================================================================

# ================================================================
# Creamos nuestro VPC
resource "aws_vpc" "Main" {                    # usamos el bloque "resource", el
"provider element" y una "etiqueta"
```

```
 cidr_block        = var.main_vpc_cidr      # le pasamos por variable el CIDR block
que quiero que use

 instance_tenancy = "default"

 tags = {

 Name = "My_VPC"

 }

}

# ================================================================

# Creamos un Internet Gateway "Y" lo asociamos al VPC que se acaba de crear

resource "aws_internet_gateway" "IGW" {    # Internet Gateway

 vpc_id =  aws_vpc.Main.id               # vamos a conocer el vpc_id solo cuando el
VPC se haya creado

 tags = {

 Name = "IGW"

 }

}

# ================================================================

# Creamos la subnet pública

resource "aws_subnet" "public_subnets" {    # creamos las subnets públicas

 vpc_id =  aws_vpc.Main.id

 cidr_block = var.public_subnets        # CIDR block para mis public subnets

 tags = {
```

```
 Name = "Public Subnet"

 }

}

# ====================================================================

# Creamos la subnet privada                    # creamos nuestras private subnets

resource "aws_subnet" "private_subnets" {

 vpc_id =  aws_vpc.Main.id

 cidr_block = var.private_subnets          # CIDR block para mis subnets privadas

 tags = {

   Name = "Private Subnet"

 }

}

# ====================================================================

# Tabla de ruteo para la subnet pública

resource "aws_route_table" "Public_RT" {     # Creamos nuestro Route Table para la
subnet pública

 vpc_id =  aws_vpc.Main.id

 route {

   cidr_block = "0.0.0.0/0"                 # Declaramos el tráfico desde la subnet
pública llega a Internet desde el Internet Gateway

   gateway_id = aws_internet_gateway.IGW.id

 }
```

```
 tags = {

   Name = "Tabla de Ruteo Pública"

 }

}

# ====================================================================

# Tabla de ruteo para la subnet privada

resource "aws_route_table" "Private_RT" {    # Creating RT for Private Subnet

 vpc_id = aws_vpc.Main.id

 route {

    cidr_block = "0.0.0.0/0"                # Tráfico proveniente desde la subnet
privadas llegando a Internet vía NAT Gateway

   nat_gateway_id = aws_nat_gateway.NAT_GW.id

 }

 tags = {

 Name = "Tabla de Ruteo Privada"

 }

}

# ====================================================================

# Asociación de tabla de ruteo con la subnet pública

resource "aws_route_table_association" "Public_RT_Association" {

   subnet_id = aws_subnet.public_subnets.id

   route_table_id = aws_route_table.Public_RT.id
```

```
}

# ====================================================================

# Asociación de tabla de ruteo con la subnet privada

resource "aws_route_table_association" "Private_RT_Association" {

   subnet_id = aws_subnet.private_subnets.id

   route_table_id = aws_route_table.Private_RT.id

}

resource "aws_eip" "NAT_EIP" {

 vpc    = true

 tags = {

 Name = "NAT con elastic IP"

 }

}

# ====================================================================

# Creación del NAT Gateway usando subnet_id y allocation_id

resource "aws_nat_gateway" "NAT_GW" {

 allocation_id = aws_eip.NAT_EIP.id

 subnet_id = aws_subnet.public_subnets.id

 tags = {

   Name = "NAT Gateway alocada a la subnet pública"

 }

}
```