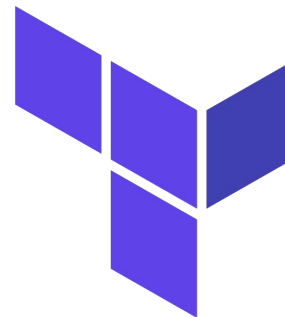


APRENDE AWS USANDO TERRAFORM

PRIMERA PARTE



¿Quien soy?

Rogelio Ramos Bellido



Github: github.com/oilegor1029

LinkedIn: www.linkedin.com/in/rogelio-ramos



Amazon Web Services y Terraform

- Amazon Web Services (AWS):

Plataforma que ofrece Amazon para poder levantar nuestras infraestructuras en la nube.

- Terraform:

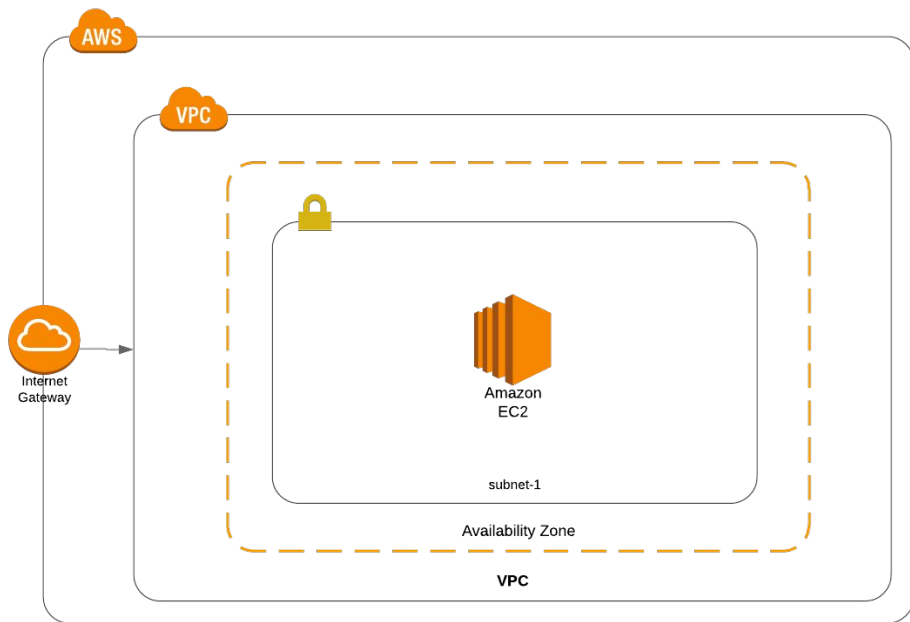
Infraestructura como código, es decir un gestor de servicios de diferentes proveedores a través de código.

```
resource "<(PROVEEDOR)>_<(RECURSO)>" "<(NOMBRE)>" {  
    <(PARAMETROS)>  
}  
  
resource "aws_vpc" "principal" {  
    cidr_block = "10.0.0.0/24"  
}
```



Workshop: Qué haremos y Requisitos

Web con Nginx en máquina EC2



Requisitos:

- Terraform instalado.
- Cuenta en AWS.
- Repositorio de apoyo: <http://asd.com>



Workshop: Tarea 00 - Provider [00-provider.tf]

Parámetros para el provider:

- Region: **eu-west-1**
- Access Key de tu cuenta AWS.
- Secret Key de tu cuenta AWS.

```
provider "aws" {  
    region      = "eu-west-1"  
    access_key  = "XXXXXXXXXXXXXXXXXX"  
    secret_key  = "XXXXXXXXXXXXXXXXXX"  
}
```

Para las credenciales AWS:

- IAM < Users < {tu user} < Security Credentials < Create access key

```
$ terraform  
init
```

para inicializar terraform en la carpeta en la que estemos.

```
$ terraform  
plan
```

para ver qué cambios vamos a hacer.

```
$ terraform  
plan
```

para aplicar los cambios.



Workshop: Tarea 01 - VPC y Subnet [01-network.tf]

```
resource "aws_vpc" "principal" {
  cidr_block = "10.0.0.0/24"
  // Esto dejará reservadas 256 IPs

  tags = {
    Name = "principal"
  }
}

resource "aws_subnet" "publica-1" {
  vpc_id     = aws_vpc.principal.id
  cidr_block = "10.0.0.0/25"
  // Del 10.0.0.0 al 10.0.0.127

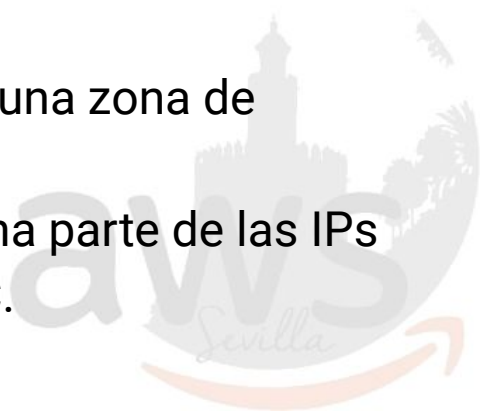
  tags = {
    Name = "publica-1"
  }
}
```

- Virtual Private Cloud (VPC):

- Red virtual en nuestra cuenta AWS, aislada de otras VPCs.
- Necesita un bloque de IPs en el que crear recursos.

- Subnet:

- Cada subred va en una zona de disponibilidad.
- Tendrá asignado una parte de las IPs asignadas a la VPC.



Workshop: Tarea 02 - Route Table [02-route-tables.tf]

```
resource "aws_internet_gateway" "igw" {
  // Referenciamos el ID la VPC
  vpc_id = aws_vpc.principal.id
  tags = { Name = "igw" }
}

resource "aws_route_table" "publica" {
  //La creamos en la VPC
  vpc_id = aws_vpc.principal.id
  route {
    //Que lo que entre sea de donde sea, entre por IGW
    cidr_block   = "0.0.0.0/0"
    gateway_id   = aws_internet_gateway.igw.id
  }
  tags = { Name = "publica" }
}

resource "aws_route_table_association" "igw-a" {
  //Asociamos RouteTable con Subnet
  subnet_id      = aws_subnet.publica-1.id
  route_table_id = aws_route_table.publica.id
}
```

- Internet Gateway (IGW):
 - Componente de Amazon para permitir acceso de Internet a la VPC.
- Route Table:
 - Sirve para determinar adonde se dirige el tráfico de red.
 - Cada subred debe estar asociada a una.
 - Debe relacionarse a un IGW para hacerse pública.

Workshop: Tarea 03 - Instance - a) Security Group [03-instance.tf]

```
resource "aws_security_group" "http-for-web-nginx" {
  name      = "http-for-web-nginx"
  vpc_id    = aws_vpc.principal.id

  //Acceder solo desde el 80 en las IPs marcadas
  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = [ "0.0.0.0/0" ]
  }

  //Que pueda salir a cualquiera
  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = [ "0.0.0.0/0" ]
  }

  tags = { Name = "http-for-web-nginx" }
}
```

- Security Group (SG):

Funciona como Firewall para controlar el tráfico entrante y saliente a instancias y otros componentes.

Configuración para workshop:

Entrada solo por puerto 80

Salida abierta.



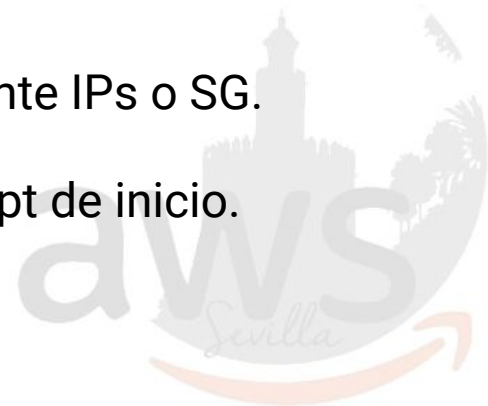
Workshop: Tarea 03 - Instance - b) Instance [03-instance.tf]

```
resource "aws_instance" "web-nginx" {  
  // Amazon Linux AMI 2018.03.0 (HVM) en eu-west-1  
  ami           = "ami-0e41581acd7dedd99"  
  // Tipo de máquina pequeña  
  instance_type = "t2.nano"  
  // Lo securizamos con el SG  
  vpc_security_group_ids = [  
    aws_security_group.http-for-web-nginx.id  
  ]  
  // Asociamos IP Pública y que use la subnet  
  associate_public_ip_address = true  
  subnet_id = aws_subnet.publica-1.id  
  // Script de inicio para NGINX  
  user_data = <<USER_DATA  
#!/bin/bash  
sudo apt update -y && sudo apt upgrade -y && sudo apt  
install nginx -y && sudo systemctl enable nginx && sudo  
systemctl start nginx  
USER_DATA  
  
  tags = { Name = "web-nginx" }  
}
```

- Elastic Compute Cloud (EC2):

Entornos virtuales de Amazon, podría decirse que son “ordenadores” en la nube.

- Hay distintos tipos de máquinas.
- Hay que indicarle una Amazon Machine Image (AMI) que usará para levantarse.
- Securizable mediante IPs o SG.
- Se puede usar Script de inicio.



Workshop: Tarea 03 - Instance - c) Outputs [03-instance.tf]

```
output "web-ip" {  
  // Tipo de máquina pequeña  
  value = aws_instance.web-nginx.public_ip  
}
```

- Terraform Outputs

Sirve para sacar parámetros de los recursos.

Los podemos ver de dos formas:

- Tras un apply.
- Con el comando terraform si ya los tienes creados y no quieres volver a hacer apply:

```
$ terraform output  
(NOMBRE)
```

```
$ terraform output web-ip
```

```
$ terraform destroy
```



- AWS-Sevilla **LinkedIn:**
www.linkedin.com/company/aws-sevilla
- AWS-Sevilla **Meetup:**
www.meetup.com/AWS-Sevilla
- AWS-Sevilla **Github:**
github.com/awsSevilla
- **Organizadores:**
[Carmelo Zubeldia](#)
[Antonio Gerena](#)
- **Ponente:**
[Rogelio Ramos](#)

