

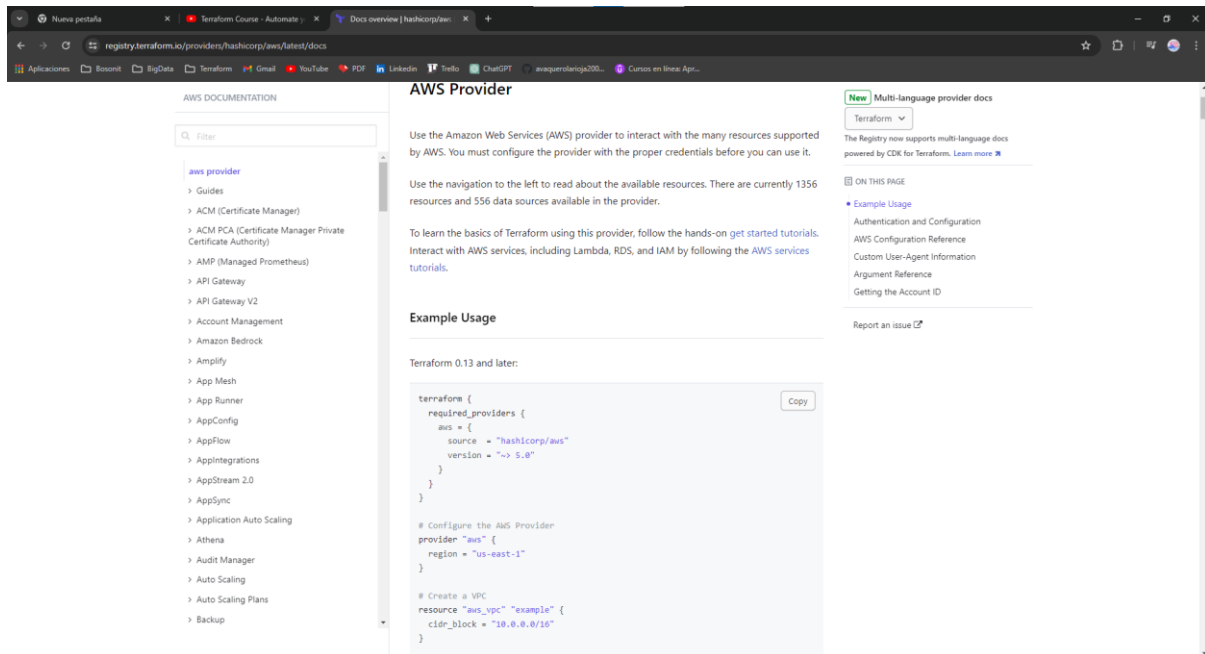
Caso práctico de Terraform con AWS



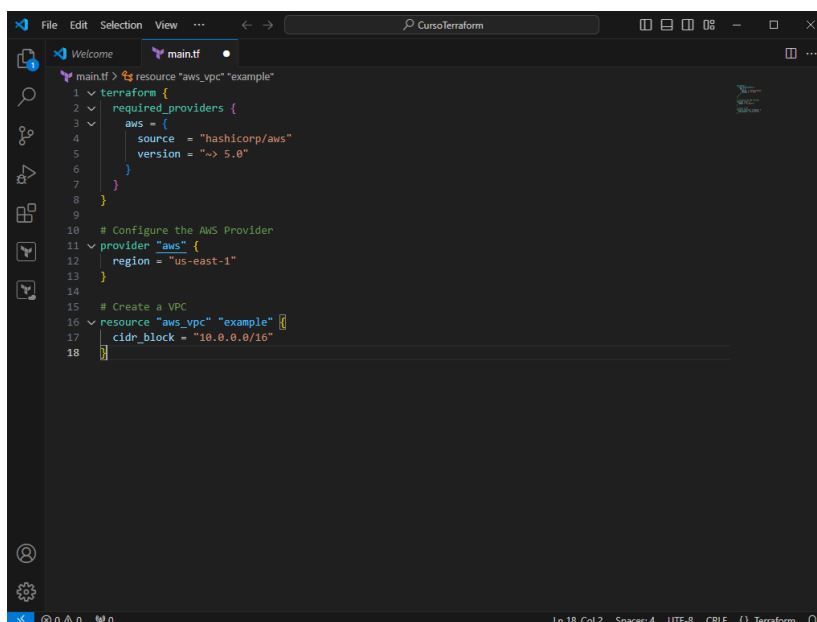
Nuestras primeras líneas de código en Terraform	3
¿Qué es una VPC?	6
Nuestro primer EC2(Elastic Computer Cloud)	8
¿Qué es un EC2?	8
¿Como crear un EC2?.....	8
Consola de AWS	9
VPC y Subred.....	9
EC2	13
Terraform	17
Código	19
Ejecutar el código	28
Conexión mediante SSH desde terminal	32
Investigación	35

Nuestras primeras líneas de código en Terraform

- Para empezar a programar en Terraform haciendo uso de AWS, nos será necesario especificar el proveedor de servicio. Para ello buscaremos lo siguiente en Google “aws provider terraform”, nos meteremos en el enlace oficial y nos tendrá que salir algo parecido a la siguiente imagen:

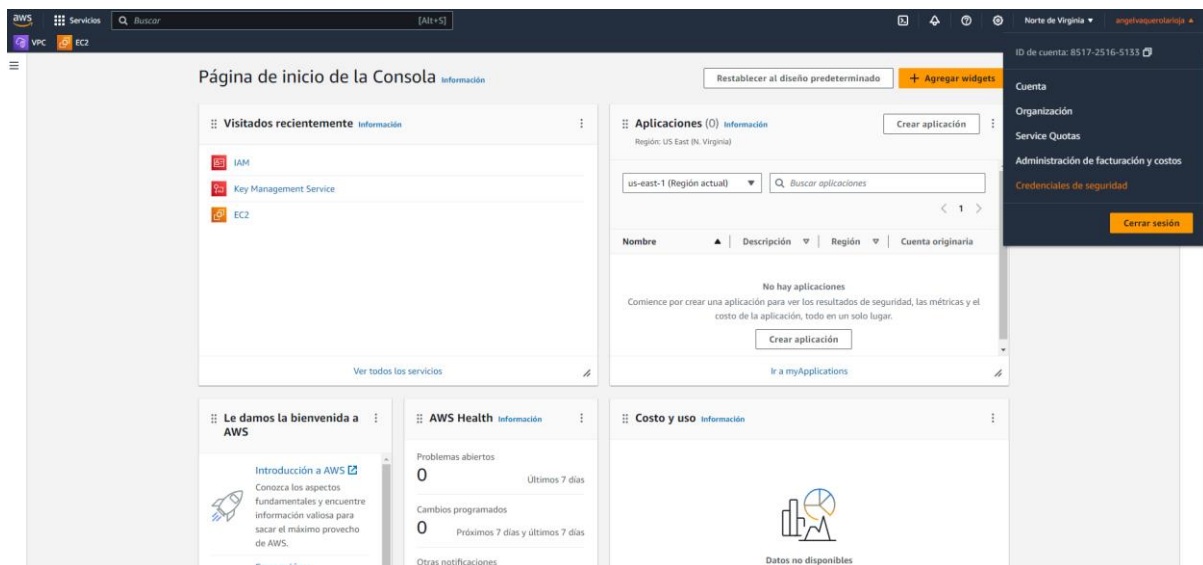


- A continuación, copiaremos el código que nos sale y lo pegaremos en nuestro main.tf que creamos antes:

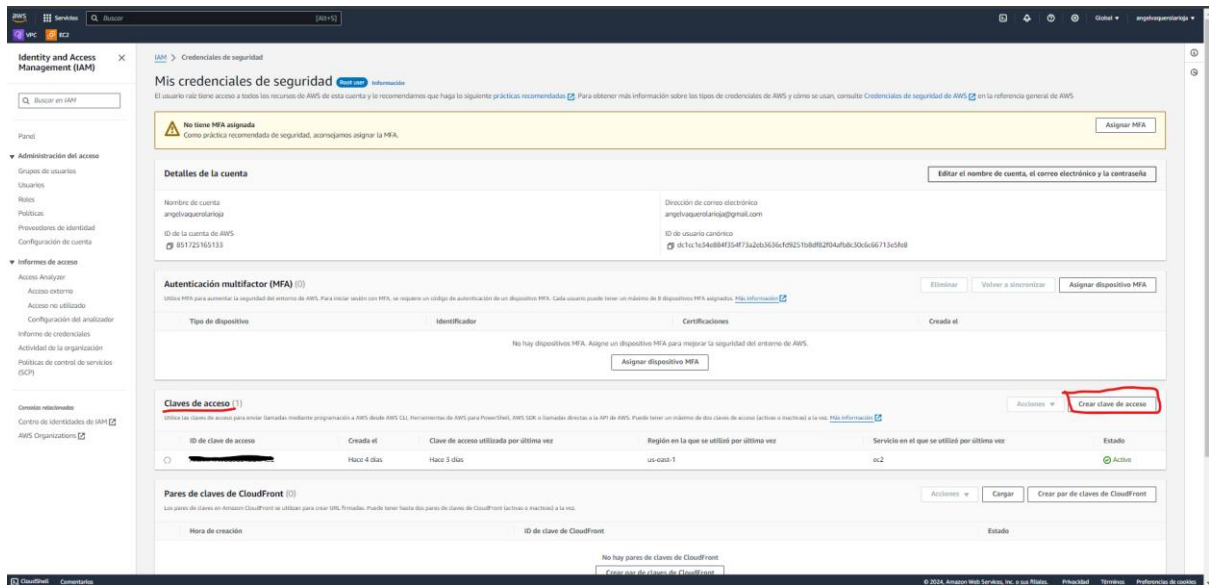


- Este código hace lo siguiente:
 - Configuración del proveedor AWS:
 - Las líneas 4-10 establecen la configuración del proveedor AWS, donde se especifica la región (us-east-1) en la que se crearán los recursos.
 - Creación de una VPC (Virtual Private Cloud):
 - Las líneas 13-17 definen la creación de una VPC en AWS. La VPC se nombra como "example" y se le asigna un bloque CIDR (10.0.0.0/16). Este bloque CIDR especifica el rango de direcciones IP que se pueden utilizar en esta VPC. En este caso, la VPC tendrá direcciones IP desde 10.0.0.0 hasta 10.0.255.255.
- Al código que hemos copiado y pegado para tener el proveedor de AWS le falta un punto muy importante la access_key y la secret_key, y eso se logra de la siguiente manera:

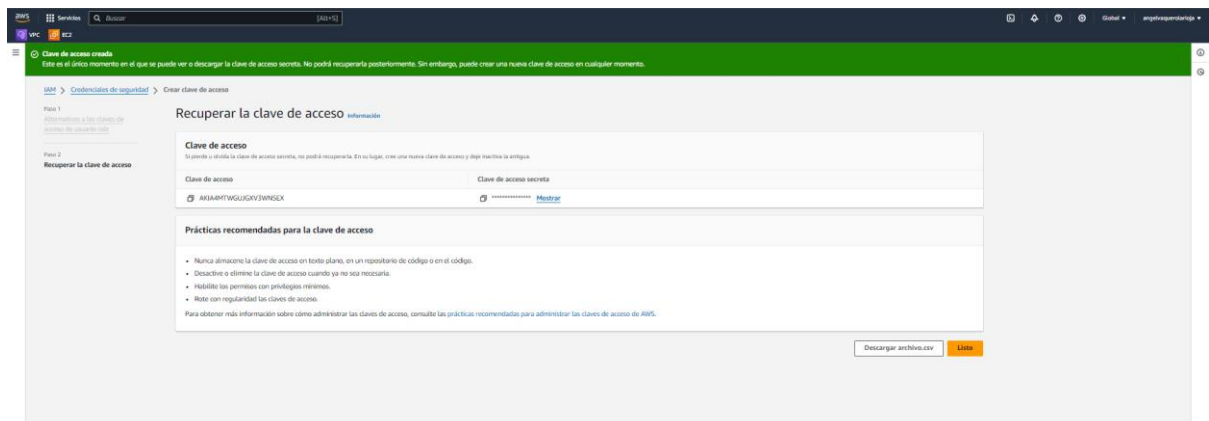
1. Desde nuestra consola de AWS iremos a “Credenciales de seguridad”:



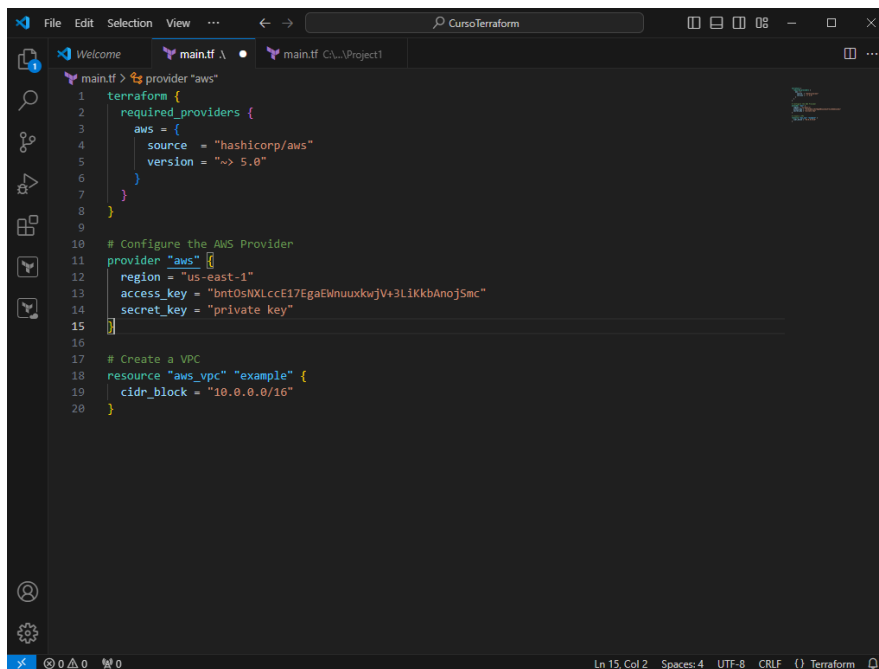
2. Desde ahí creamos nuestra clave de la siguiente manera:



- Y se nos creará el par de claves:



- Recomiendo descargarlas y guardarlas en un lugar seguro y nunca mostrar la clave privada, ahora podemos proseguir en nuestro código añadiéndolas de la siguiente manera:



```
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "~> 5.0"
6     }
7   }
8 }
9
10 # Configure the AWS Provider
11 provider "aws" {
12   region = "us-east-1"
13   access_key = "bnt0sMXLccE17EgaEwmuuxkvjV+3LlKkbAnoj5mc"
14   secret_key = "private key"
15 }
16
17 # Create a VPC
18 resource "aws_vpc" "example" {
19   cidr_block = "10.0.0.0/16"
20 }
```

- El motivo del uso de las claves es el siguiente:
 - o Las claves de acceso son pares de identificadores y secretos que se utilizan para autenticar y autorizar las llamadas a la API de AWS.
 - o Cuando trabajas con la AWS Command Line Interface (CLI), las herramientas de AWS para PowerShell, o el SDK de AWS, necesitas proporcionar estas claves de acceso para autenticar tu solicitud a AWS.

¿Qué es una VPC?

- Una VPC (Virtual Private Cloud) es un servicio de red en la nube que te permite crear una red virtual aislada en la infraestructura de un proveedor de servicios en la nube, como Amazon Web Services (AWS). En una VPC, puedes definir tu propia gama de direcciones IP, subredes, tablas de enrutamiento y configuraciones de seguridad.
- Algunos aspectos clave de una VPC incluyen:
 - o **Aislamiento de red:** Una VPC te proporciona un espacio de red virtualizado en el que puedes lanzar recursos de AWS, como instancias de EC2 (Elastic Compute Cloud) o bases de datos RDS (Relational Database Service), de forma aislada de otras redes.

- **Control de la red:** Puedes definir tus propias subredes dentro de la VPC y configurar reglas de enrutamiento para dirigir el tráfico de red según tus necesidades específicas. Además, puedes controlar el acceso a los recursos de la VPC mediante grupos de seguridad y listas de control de acceso (ACL).
- **Conectividad:** Una VPC puede estar conectada a tu red corporativa a través de una conexión VPN (Virtual Private Network) o una conexión de red dedicada, permitiendo una integración segura entre tus recursos locales y los recursos en la nube.

Nuestro primer EC2(Elastic Computer Cloud)

¿Qué es un EC2?

- EC2 significa Elastic Compute Cloud. Es uno de los servicios principales de computación en la nube ofrecidos por Amazon Web Services (AWS). EC2 te permite alquilar capacidad informática virtual en la nube, lo que te permite ejecutar aplicaciones y trabajar con cargas de trabajo de manera escalable y flexible.
- Algunas características clave de EC2 incluyen:
 - o **Escalabilidad:** Puedes iniciar una o miles de instancias de EC2 en cuestión de minutos para satisfacer la demanda de tus aplicaciones.
 - o **Flexibilidad:** EC2 te permite elegir entre una variedad de tipos de instancias con diferentes combinaciones de CPU, memoria, almacenamiento y capacidades de red para adaptarse a tus necesidades específicas.
 - o **Control:** Tienes control total sobre tus instancias de EC2, lo que te permite iniciar, detener, terminar y administrarlas según sea necesario. Además, puedes elegir el sistema operativo, configurar la seguridad, gestionar el almacenamiento y realizar otras configuraciones personalizadas.
 - o **Pago por uso:** EC2 opera en un modelo de pago por uso, lo que significa que solo pagas por la capacidad informática que consumes, sin necesidad de compromisos a largo plazo ni costos iniciales elevados.

¿Como crear un EC2?

- Tenemos dos opciones para crear nuestro primer EC2, la opción gráfica desde la consola de AWS y la opción desde código de Terraform, veremos las opciones de creación por el orden mencionado.

Consola de AWS

VPC y Subred

- Antes de crear el EC2 por consola de AWS necesitaremos crear un VPC y una subred para la maquina:

1. Buscamos VPC y creamos una:

The image displays two screenshots of the AWS Management Console interface. The top screenshot shows the search results for 'vpc' in the 'Servicios' (Services) section. It lists various VPC-related services like VPC, AWS Firewall Manager, Detective, and Managed Services, along with their characteristics and resources. The bottom screenshot shows the 'Crear VPC' (Create VPC) page. It includes a 'Recursos por región' (Resources by region) section with a grid of resource cards for VPC, Subnet, Route, and other VPC-related services. The right sidebar contains the 'Estado del servicio' (Service status), 'Configuración' (Configuration), 'Información adicional' (Additional information), and 'Conexiones de VPN sitio a sitio' (Site-to-site VPN connections) sections.

aws

Servicios

Buscar

[Alt+S]

VPC

EC2

VPC

Sus VPC

Crear VPC

Crear VPC

Información

Una VPC es una parte aislada de la nube de AWS que contiene objetos de AWS, como instancias de Amazon EC2.

Configuración de la VPC

Recursos que se van a crear [Información](#)
Cree únicamente el recurso de VPC o la VPC y otros recursos de red.

☒ Solo la VPC

☐ VPC y más

Etiqueta de nombre - *opcional*
Crea una etiqueta con una clave de "Nombre" y el valor que usted especifique.

vpc-consola

Bloque de CIDR IPv4 [Información](#)
☒ Entrada manual de CIDR IPv4
☐ Bloque de CIDR IPv4 asignado por IPAM

CIDR IPv4

10.0.0.0/16

El tamaño del bloque CIDR debe estar entre /16 y /28.

Bloque de CIDR IPv6 [Información](#)
☒ Sin bloque de CIDR IPv6
☐ Bloque de CIDR IPv6 asignado por IPAM
☐ Bloque de CIDR IPv6 proporcionado por Amazon
☐ CIDR IPv6 de mi propiedad

Tenencia [Información](#)

Predeterminado

Etiquetas

Una etiqueta es una marca que se asigna a un recurso de AWS. Cada etiqueta consta de una clave y un valor opcional. Puede utilizar las etiquetas para buscar y filtrar sus recursos o hacer un seguimiento de los costos de AWS.

Clave

Valor - *opcional*

Q

Name

X

Q

vpc-consola

X

Eliminar etiqueta

Agregar etiqueta

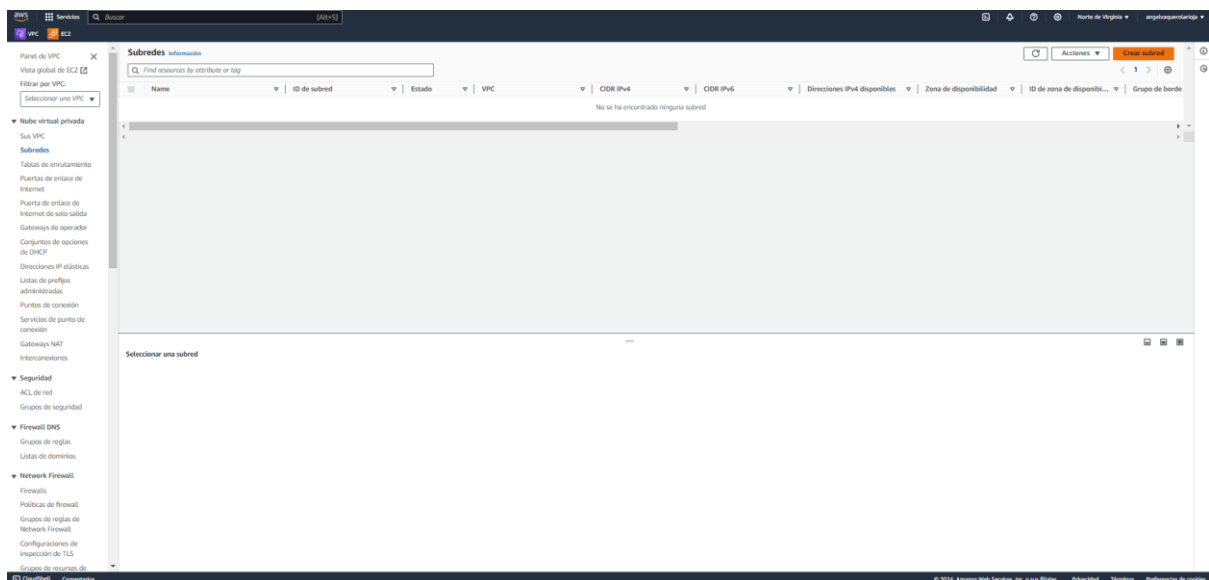
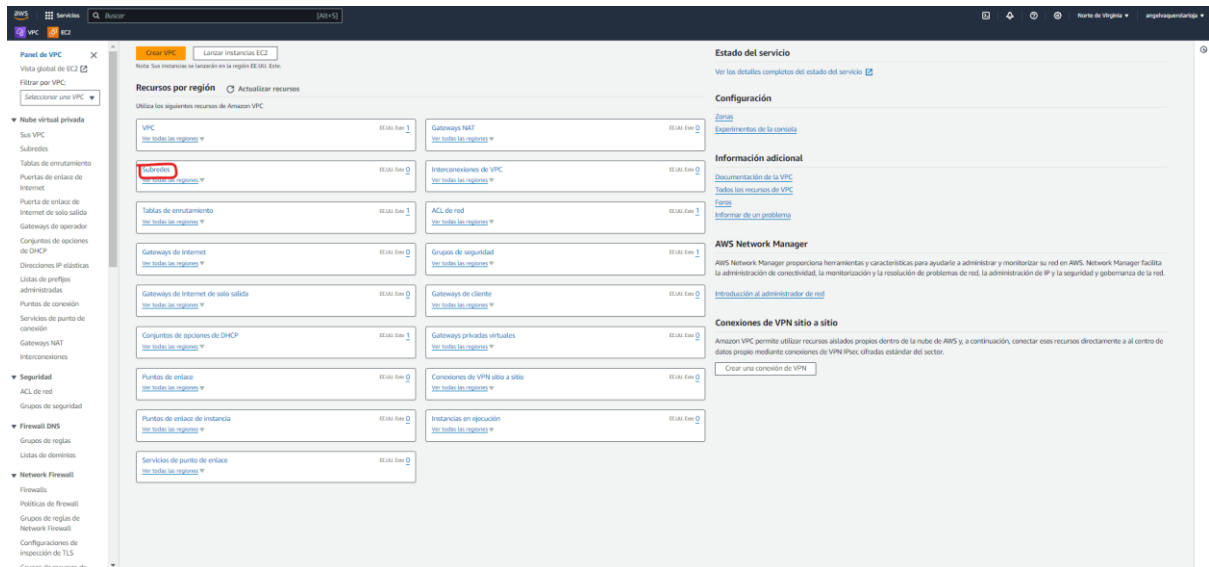
Puede agregar 49 etiquetas más

Cancelar

Crear VPC

Autor: Ángel Vaquero Toncheva

- Ahora creamos la subred:



- Le indicamos el VPC y le decimos cual es el rango de IPs de esa subred:

VPC > Subredes > Crear subred

Crear subred Información

VPC

ID de la VPC
Cree subredes en esta VPC.

vpc-01dc20584842c173e (vpc-consola) ▼

CIDR de VPC asociados

CIDR IPv4
10.0.0.0/16

Configuración de la subred

Especifique los bloques de CIDR y la zona de disponibilidad de la subred.

Subred 1 de 1

Nombre de la subred
Cree una etiqueta con una clave de "Nombre" y el valor que especifique.

subred-consola

El nombre puede tener un máximo de 256 caracteres.

Zona de disponibilidad Información
Elija la zona en la que residirá la subred o deje que Amazon elija una por usted.

Sin preferencia ▼

Bloque de CIDR de VPC IPv4 Información
Elija el bloque de CIDR de la VPC IPv4 en el que crear una subred.

10.0.0.0/16 ▼

Bloque de CIDR de la subred IPv4

10.0.1.0/24 256 IPs

< > ^ v

▼ **Etiquetas: opcional**

Clave	Valor - opcional
Q Name X	Q subred-consola X

Quitar

Agregar nueva etiqueta

Puede agregar 49 más etiquetas.

Quitar

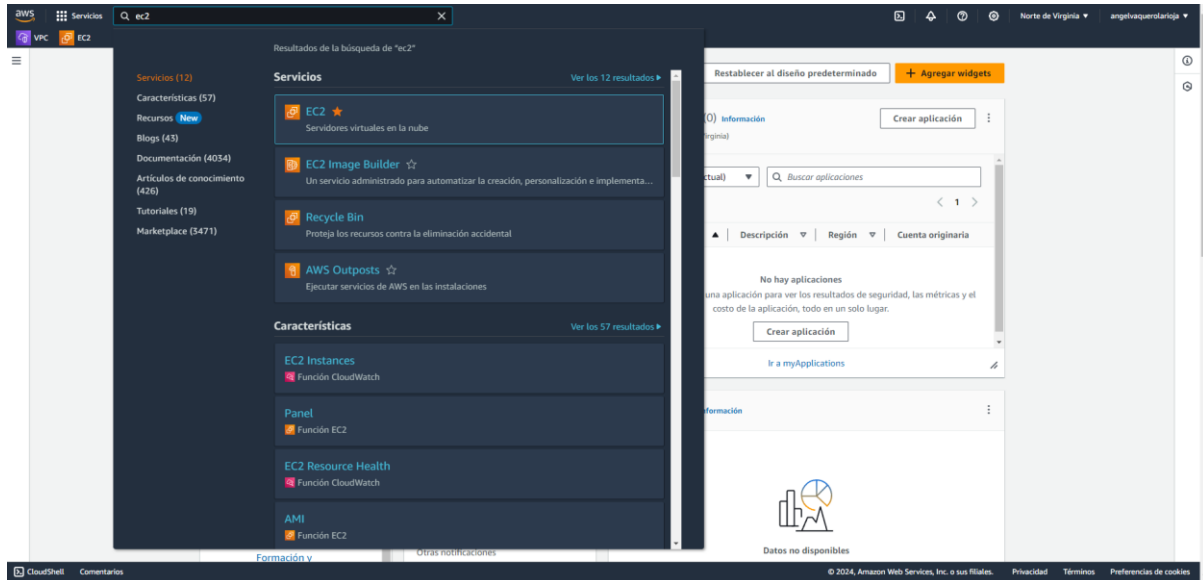
Agregar nueva subred

Cancelar **Crear subred**

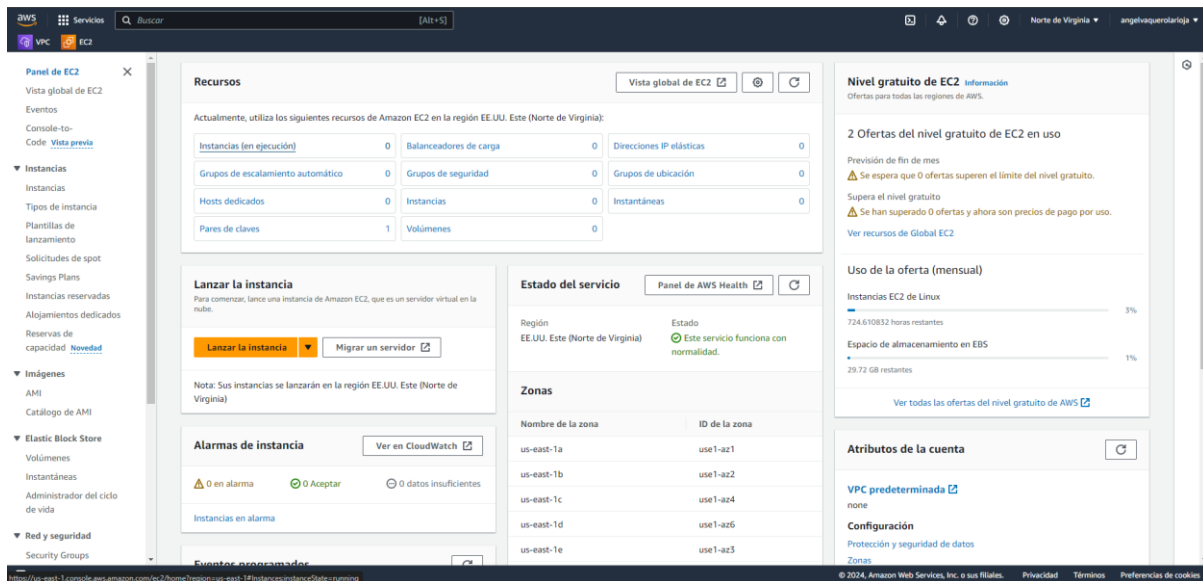
EC2

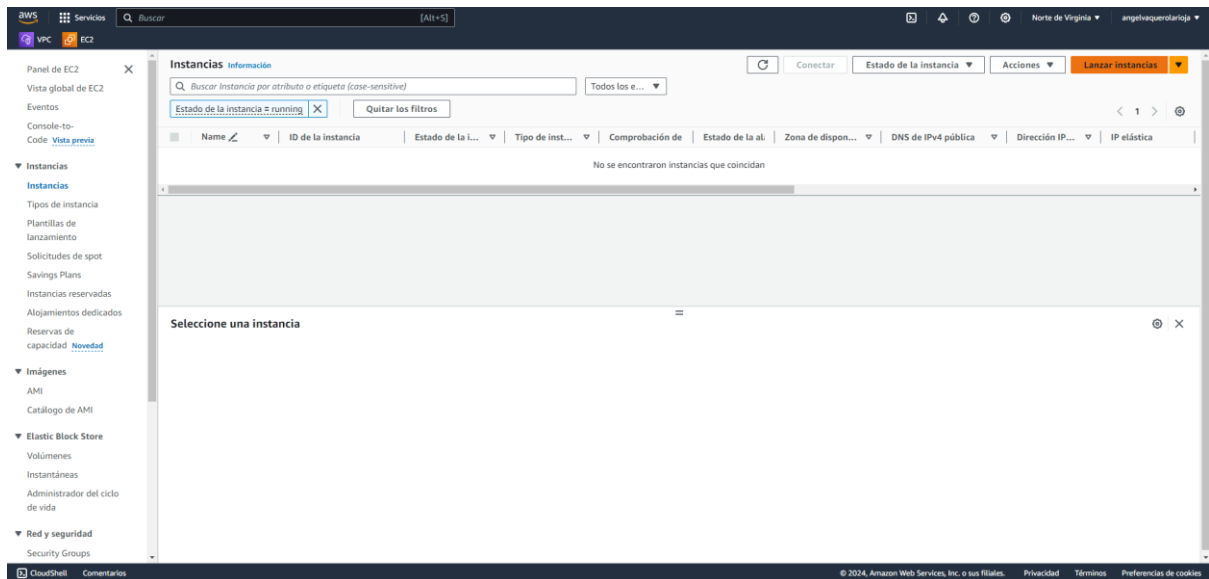
- Para realizar el EC2 desde consola seguiremos los siguientes pasos:

1. Buscaremos lo siguiente desde la consola y nos meteremos a la primera opción:

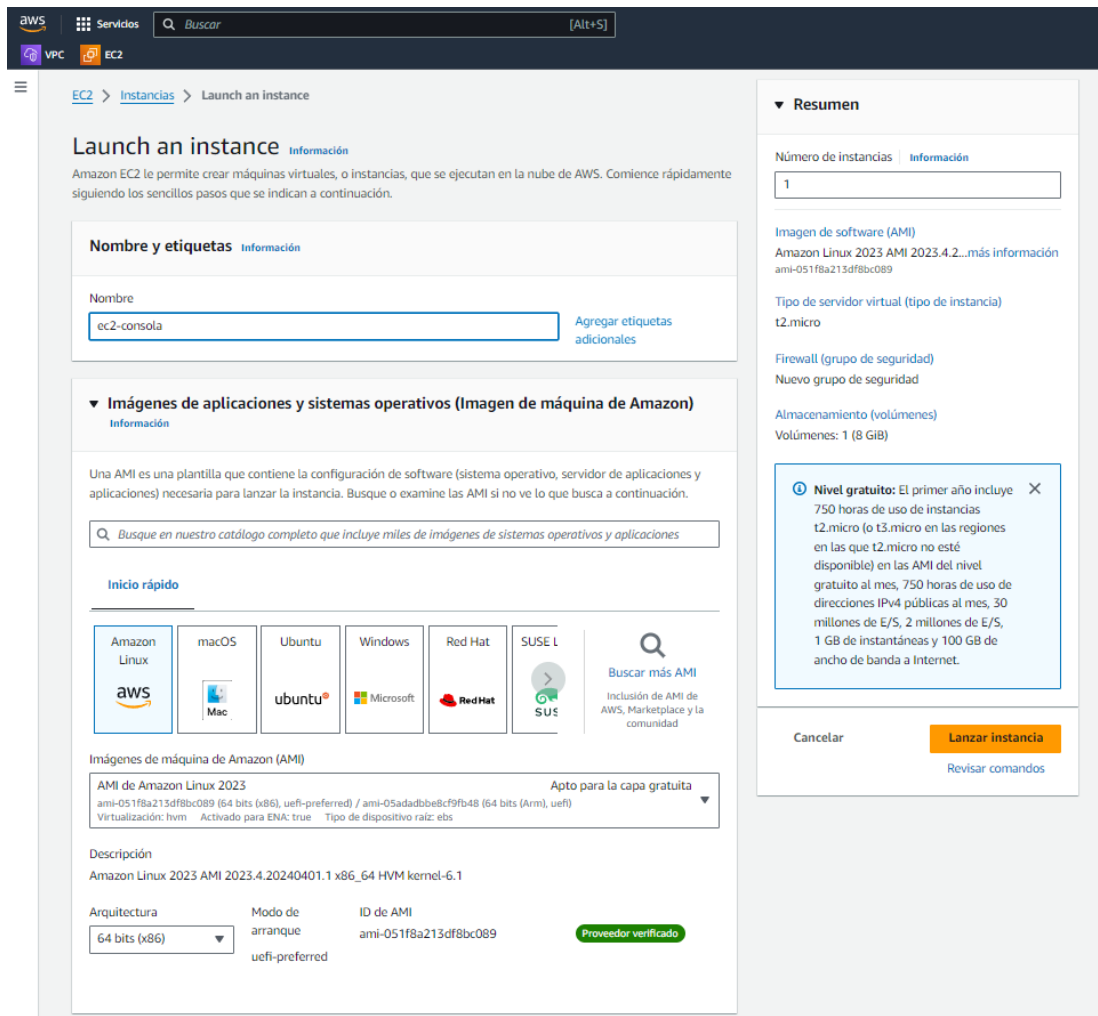


2. A continuación, iremos a Instancias en ejecución y lanzaremos una nueva instancia:





- Le agregamos un nombre a la máquina y le indicamos la imagen para la máquina que es Ubuntu, que es gratuita:



- Le indicamos el tipo de máquina, en este caso t2.micro que es gratuita:

Tipo de instancia [Información](#) [Obtener asesoramiento](#)

Tipo de instancia

t2.micro Apto para la capa gratuita

Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true

Bajo demanda Windows base precios: 0.0162 USD por hora

Bajo demanda SUSE base precios: 0.0116 USD por hora

Bajo demanda RHEL base precios: 0.0716 USD por hora

Bajo demanda Linux base precios: 0.0116 USD por hora

☐ Todas las generaciones

[Comparar tipos de instancias](#)

Se aplican costos adicionales a las AMI con software preinstalado

Par de claves (inicio de sesión) [Información](#)

Puede utilizar un par de claves para conectarse de forma segura a la instancia. Asegúrese de que tiene acceso al par de claves seleccionado antes de lanzar la instancia.

Nombre del par de claves - obligatorio

[Crear un nuevo par de claves](#)

Configuraciones de red [Información](#) [Editar](#)

Red [Información](#)

vpc-01dc20584842c173e | vpc-consola

Subred [Información](#)

subnet-0a9a400bcf8f8b30a | subred-consola

Asignar automáticamente la IP pública [Información](#)

Desactivar

Firewall (grupos de seguridad) [Información](#)

Un grupo de seguridad es un conjunto de reglas de firewall que controlan el tráfico de la instancia. Agregue reglas para permitir que un tráfico específico llegue a la instancia.

☒ Crear grupo de seguridad ☐ Seleccionar un grupo de seguridad existente

Crearemos un nuevo grupo de seguridad denominado "launch-wizard-1" con las siguientes reglas:

☒ Permitir el tráfico de SSH desde [Ayuda a establecer conexión con la instancia](#)

☐ Permitir el tráfico de HTTPS desde Internet

Para configurar un punto de enlace, por ejemplo, al crear un servidor web

☐ Permitir el tráfico de HTTP desde Internet

Para configurar un punto de enlace, por ejemplo, al crear un servidor web

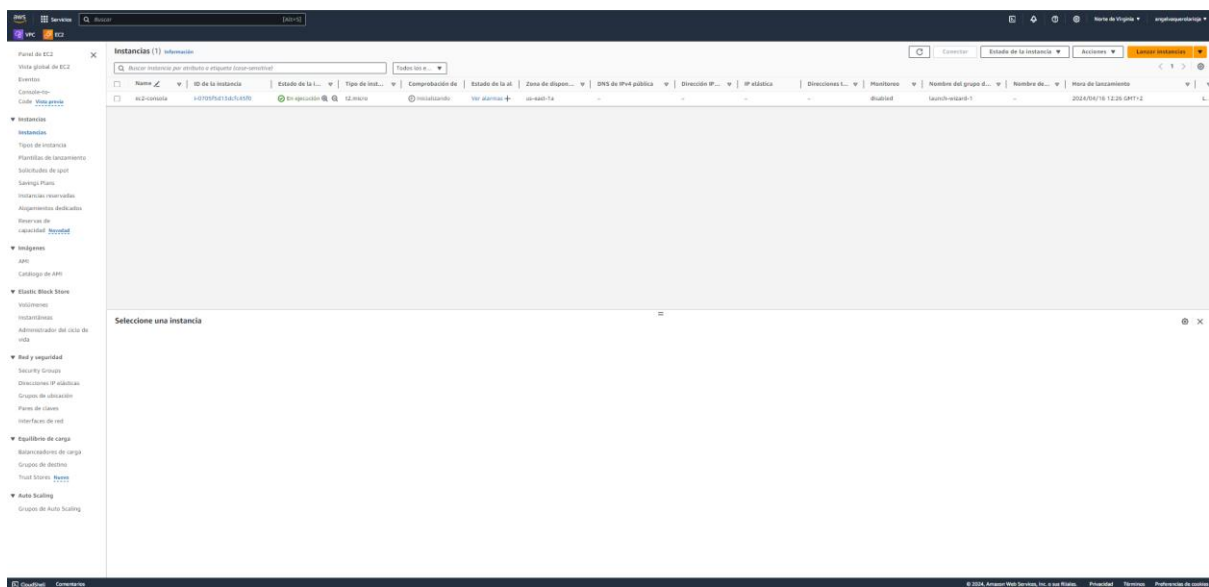
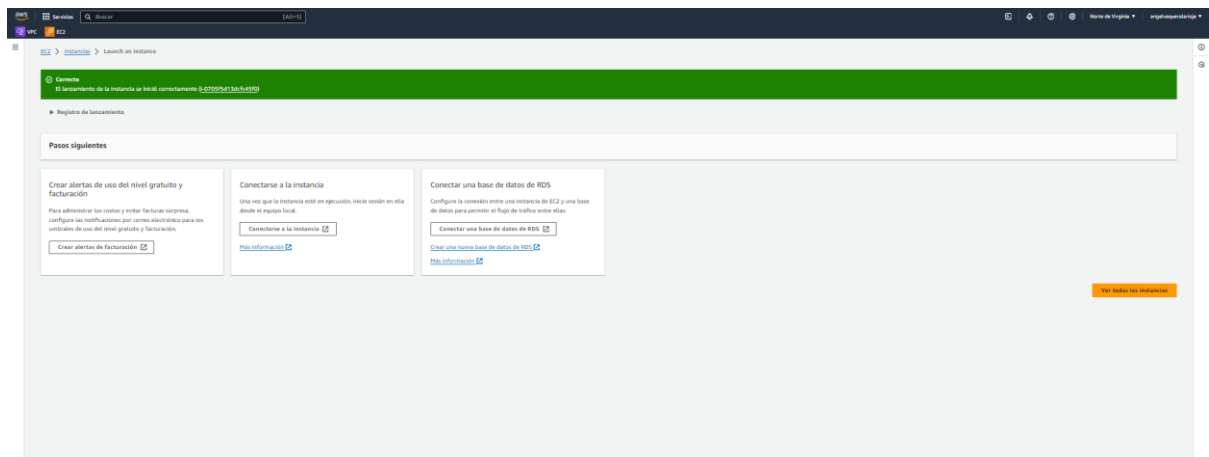
Nivel gratuito: El primer año incluye 750 horas de uso de instancias t2.micro (o t3.micro en las regiones en las que t2.micro no esté disponible) en las AMI del nivel gratuito al mes, 750 horas de uso de direcciones IPv4 públicas al mes, 30 millones de E/S, 2 millones de E/S, 1 GB de instantáneas y 100 GB de ancho de banda a Internet.

Cancelar [Lanzar instancia](#) [Revisar comandos](#)

- Nota: La configuración de red se nos pondrá automáticamente con lo creado anteriormente, pero si quisiéramos podríamos editarla.

Autor: Ángel Vaquero Toncheva

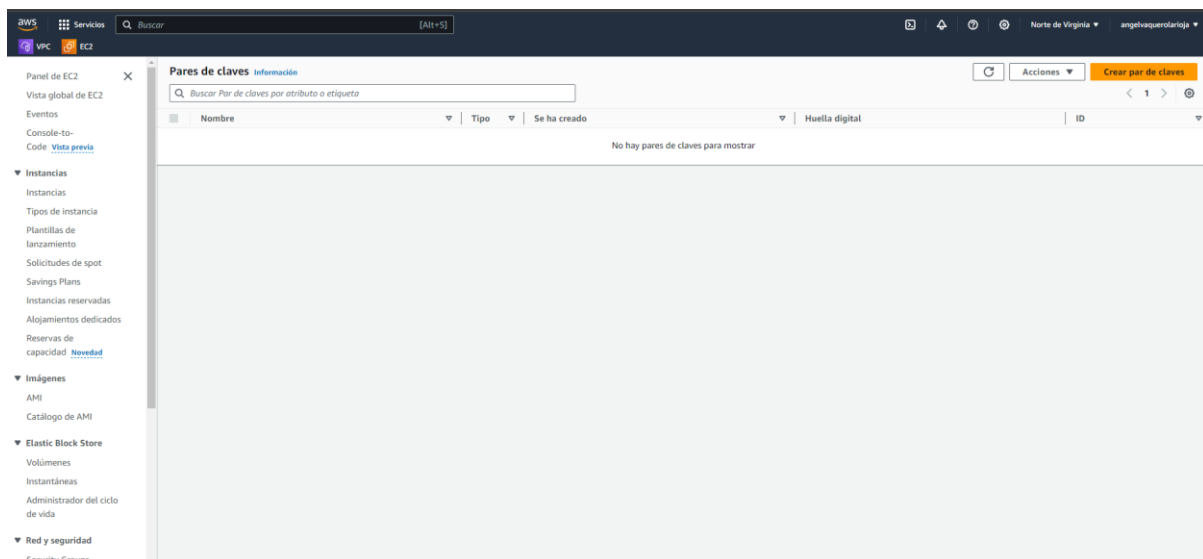
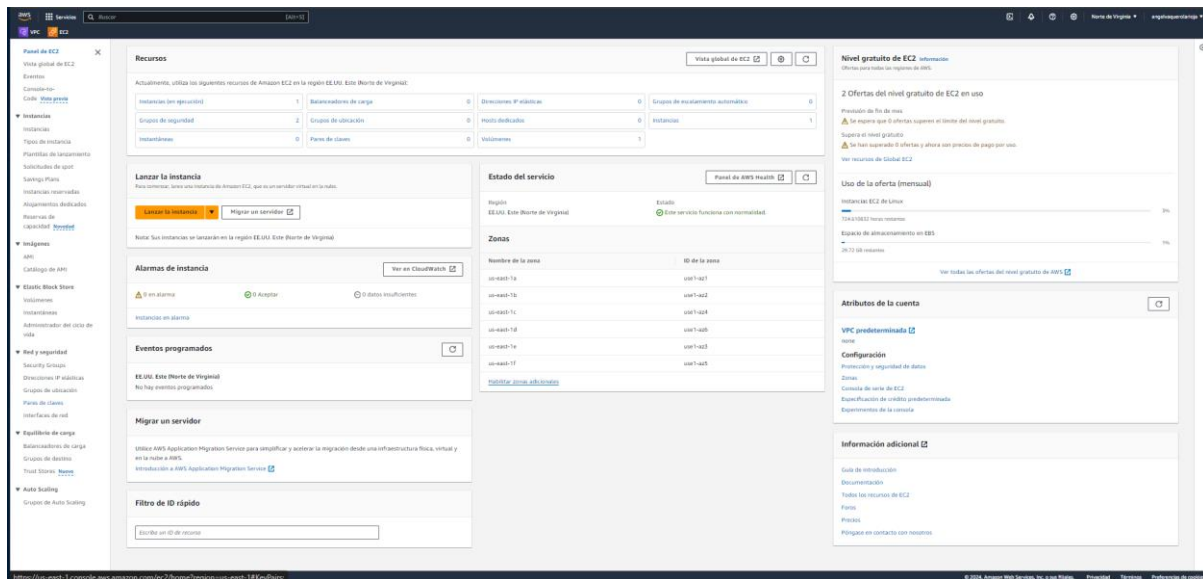
- Y lanzamos la instancia:



- Y así se lanza una maquina EC2 desde la consola de AWS, ahora os enseñaré a hacerlo desde Terraform con una configuración parecida.

Terraform

- Antes de empezar con el apartado de código tendremos que generar un par de keys para poder generar, modificar y acceder a nuestro futuro EC2. A continuación se muestra la forma de generarlas:
 - o Desde el panel de EC2 en el apartado de “Red y seguridad” necesitaremos ir a “Pares de claves”:



- Y las generamos como se indica en la imagen:

The screenshot shows the AWS Management Console interface for creating a new key pair. The breadcrumb navigation at the top indicates the path: EC2 > Pares de claves > Crear par de claves. The main heading is 'Crear par de claves' with a link to 'Información'. Below this, a section titled 'Par de claves' explains that a key pair consists of a private key and a public key used for authentication. The form contains the following fields and options:

- Nombre:** A text input field containing 'main-key'. A note below states: 'El nombre puede incluir hasta 255 caracteres ASCII. No puede incluir espacios al principio ni al final.'
- Tipo de par de claves:** Two radio button options: 'RSA' (selected) and 'ED25519'.
- Formato de archivo de clave privada:** Two radio button options: '.pem' (selected, with the note 'Para usar con OpenSSH') and '.ppk' (with the note 'Para usar con PuTTY').
- Etiquetas:** Labeled as 'opcional', it states 'No hay etiquetas asociadas a este recurso.' and includes a button 'Agregar nueva etiqueta' with a note 'Puede agregar hasta 50 etiquetas más.'

At the bottom right of the form, there are two buttons: 'Cancelar' and 'Crear par de claves'.

- Pondremos la clave a la par del main.tf en nuestra carpeta del curso por comodidad, pero se debería de guardar en un lugar seguro.
- Ahora ya podemos pasar al código.

Código

- Comenzamos con la configuración básica:

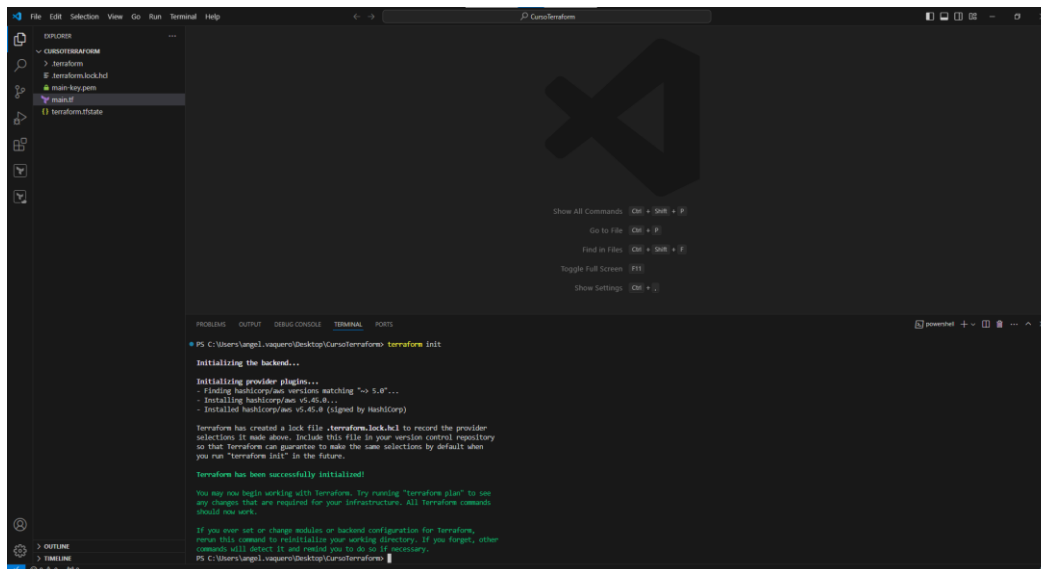
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1" #Región donde quieres crear todo lo indicado en el
codigo
  access_key = "AKIA4MTWGUJGXV3WN5EX"
  secret_key = "private key" #Tu clave privada
}

#VPC
resource "aws_vpc" "vpc-terraform" {
  cidr_block = "10.0.0.0/16" #Para indicar el rango a usar
  tags = {
    Name="vpc-terraform" #El tags es para poner notas, en este caso el nombre
  }
}
```

- Explicación del código:
 - **Configuración de proveedores requeridos:**
 - En la sección terraform { ... }, se especifica que este código requiere el proveedor de AWS de HashiCorp con una versión aproximada de 5.0. Esto asegura que Terraform utilice la versión correcta del proveedor al desplegar los recursos.
 - **Configuración del proveedor AWS:**
 - La sección provider "aws" { ... } configura el proveedor de AWS con la región us-east-1 (Norte de Virginia). Además, se proporcionan las claves de acceso y secreto para autenticar la conexión con AWS. Es importante tener en cuenta que el acceso y la clave secretos deben mantenerse seguros y no deben compartirse públicamente.
 - **Creación de una VPC:**
 - La sección resource "aws_vpc" "vpc-terraform" { ... } define la creación de una VPC en AWS. Esta VPC se nombra como "vpc-terraform" y se le asigna un bloque CIDR (10.0.0.0/16). El CIDR especifica el rango de direcciones IP que pueden utilizarse en esta VPC. También se le asigna un tag con el nombre "vpc-terraform", que es útil para identificar y organizar los recursos en AWS.

- Realizamos los primeros comandos desde la consola de VSCode para iniciar terraform y comprobar que pusimos el código de forma correcta:
 - o Realizamos un "terraform init":



The screenshot shows the VS Code interface with a terminal window open. The terminal displays the output of the 'terraform init' command. It starts with 'Initializing the backend...' and 'Initializing provider plugins...'. It then shows the installation of the AWS provider (hashicorp/aws v5.62.0) and the creation of a lock file (.terraform.lock.hcl). The output concludes with 'Terraform has been successfully initialized!' and instructions on how to use Terraform.

```
PS C:\Users\langel.vagaro\Desktop\CursoTerraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.62.0...
- Installed hashicorp/aws v5.62.0 (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 selection by default when
you run "terraform init" in the future.

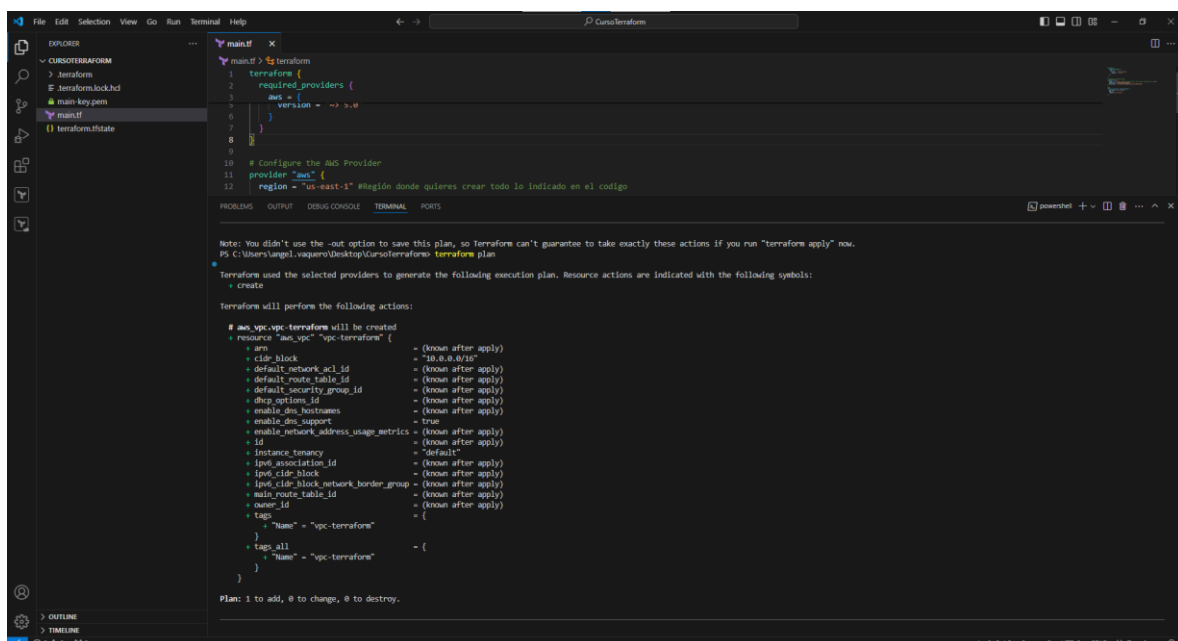
Terraform has been successfully initialized!

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.

PS C:\Users\langel.vagaro\Desktop\CursoTerraform>
```

- Nota: Para lanzar la terminal de VSCode de clic a la casilla de la parte de arriba que tiene el nombre de terminal y cree una nueva.
 - o Ahora ejecutamos un “terraform plan” para comprobar que le indicamos de forma correcta la creación del VPC:



The screenshot shows the VS Code interface with a terminal window open. The terminal displays the output of the 'terraform plan' command. It starts with a note about the '-out' option. Then it shows the execution plan generated by Terraform, listing the actions it will perform, such as creating the 'aws_vpc' resource and its associated subnets, route tables, and internet gateway.

```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\Users\langel.vagaro\Desktop\CursoTerraform> terraform plan

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:

# aws_vpc.vpc-terraform will be created
resource "aws_vpc" "vpc-terraform" {
  cidr_block = "10.0.0.0/16"           = (known after apply)
  default_network_acl_id = (known after apply)
  default_route_table_id = (known after apply)
  default_security_group_id = (known after apply)
  dhcp_options_id = (known after apply)
  enable_dns_hostnames = true         = (known after apply)
  enable_dns_support = true           = (known after apply)
  enable_network_address_usage_metrics = (known after apply)
  id = (known after apply)
  instance_tenancy = "default"       = (known after apply)
  ipv6_association_id = (known after apply)
  ipv6_cidr_block = (known after apply)
  ipv6_cidr_block_network_border_group = (known after apply)
  main_route_table_id = (known after apply)
  owner_id = (known after apply)
  tags = {
    "Name" = "vpc-terraform"
  }
  tags_all = {
    "Name" = "vpc-terraform"
  }
}

Plan: 1 to add, 0 to change, 0 to destroy.
```

- Nota: “terraform plan” sirve para visualizar lo que generaría el “terraform apply” que es el comando para confirmar que quieres lanzar esa orden al AWS.

- Ahora añadimos una Subnet, un Gateway, una Route Table y unimos la Subnet con la Route Table:

```
#Subnet
resource "aws_subnet" "subnet-terraform" {
  vpc_id      = aws_vpc.vpc-terraform.id
  cidr_block  = "10.0.1.0/24"
  availability_zone = "us-east-1a"
  tags = {
    Name="subnet-terraform"
  }
}

#Gateway
resource "aws_internet_gateway" "gateway-terraform" {
  vpc_id = aws_vpc.vpc-terraform.id
}

#Route table
resource "aws_route_table" "route-table-terraform" {
  vpc_id = aws_vpc.vpc-terraform.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gateway-terraform.id
  }
}

#Join subnet with route table
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.subnet-terraform.id
  route_table_id = aws_route_table.route-table-terraform.id
}
```

- Explicación del código:
 - **Subred:**
 - La sección resource "aws_subnet" "subnet-terraform" { ... } define la creación de una subred en la VPC existente. Esta subred se nombra como "subnet-terraform" y se le asigna un bloque CIDR (10.0.1.0/24). Además, se especifica la zona de disponibilidad (us-east-1a) en la que se creará esta subred. También se le asigna un tag con el nombre "subnet-terraform".
 - **Puerta de enlace a Internet:**
 - La sección resource "aws_internet_gateway" "gateway-terraform" { ... } define la creación de una puerta de enlace a Internet en la VPC existente. Esta puerta de enlace se asocia con la VPC creada anteriormente.
 - **Tabla de enrutamiento:**
 - La sección resource "aws_route_table" "route-table-terraform" { ... } define la creación de una tabla de enrutamiento en la VPC existente. Se especifica una ruta con un CIDR de 0.0.0.0/0 que apunta a la puerta de enlace a Internet creada anteriormente.
 - **Asociación de tabla de enrutamiento con subred:**
 - La sección resource "aws_route_table_association" "a" { ... } asocia la tabla de enrutamiento creada anteriormente con la subred creada. Esto permite que la subred utilice la tabla de enrutamiento para determinar cómo se dirige el tráfico.

- Ahora generamos las reglas de seguridad:

```
#Security group
resource "aws_security_group" "allow_web" {
  name          = "allow_web_traffic"
  description   = "Allow TLS inbound traffic and all outbound traffic"
  vpc_id        = aws_vpc.vpc-terraform.id

  tags = {
    Name = "allow_web"
  }
}

resource "aws_vpc_security_group_ingress_rule" "allow_https_ipv4" {
  security_group_id = aws_security_group.allow_web.id
  description       = "HTTPS"
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 443
  ip_protocol       = "tcp"
  to_port           = 443
}

resource "aws_vpc_security_group_ingress_rule" "allow_http_ipv4" {
  security_group_id = aws_security_group.allow_web.id
  description       = "HTTP"
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 80
  ip_protocol       = "tcp"
  to_port           = 80
}

resource "aws_vpc_security_group_ingress_rule" "allow_ssh_ipv4" {
  security_group_id = aws_security_group.allow_web.id
  description       = "SSH"
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 22
  ip_protocol       = "tcp"
  to_port           = 22
}

resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" {
  security_group_id = aws_security_group.allow_web.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1" # semantically equivalent to all ports
}
```

- Explicación del código:
 - **Grupo de seguridad (Security Group):**
 - La sección resource "aws_security_group" "allow_web" { ... } define un grupo de seguridad llamado "allow_web_traffic" que permite el tráfico web. Se especifica que este grupo de seguridad estará asociado con la VPC creada anteriormente.
 - **Regla de ingreso para HTTPS:**
 - La sección resource "aws_vpc_security_group_ingress_rule" "allow_https_ipv4" { ... } define una regla de ingreso que permite el tráfico HTTPS (puerto 443) desde cualquier dirección IP IPv4 (0.0.0.0/0) hacia las instancias dentro del grupo de seguridad.
 - **Regla de ingreso para HTTP:**
 - La sección resource "aws_vpc_security_group_ingress_rule" "allow_http_ipv4" { ... } define una regla de ingreso que permite el tráfico HTTP (puerto 80) desde cualquier dirección IP IPv4 (0.0.0.0/0) hacia las instancias dentro del grupo de seguridad.
 - **Regla de ingreso para SSH:**
 - La sección resource "aws_vpc_security_group_ingress_rule" "allow_ssh_ipv4" { ... } define una regla de ingreso que permite el tráfico SSH (puerto 22) desde cualquier dirección IP IPv4 (0.0.0.0/0) hacia las instancias dentro del grupo de seguridad.
 - **Regla de salida para todo el tráfico:**
 - La sección resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" { ... } define una regla de salida que permite todo el tráfico saliente desde las instancias dentro del grupo de seguridad "allow_web_traffic". Se especifica que todo el tráfico saliente está permitido hacia cualquier dirección IP IPv4 (0.0.0.0/0). La regla de salida tiene un protocolo de -1, lo que es semánticamente equivalente a todos los puertos y protocolos.

- Ahora añadimos la interface y la EIP:

```
#Interface
resource "aws_network_interface" "interface-terraform" {
  subnet_id      = aws_subnet.subnet-terraform.id
  private_ips    = ["10.0.1.50"]
  security_groups = [aws_security_group.allow_web.id]
}

#EIP
resource "aws_eip" "one" {
  domain                = "vpc"
  network_interface     = aws_network_interface.interface-terraform.id
  associate_with_private_ip = "10.0.1.50"
  depends_on = [ aws_internet_gateway.gateway-terraform ]
}
```

- Explicación del código:
 - **Interfaz de red (Network Interface):**
 - La sección resource "aws_network_interface" "interface-terraform" { ... } define una interfaz de red en AWS. Esta interfaz de red se asocia con la subred subnet-terraform y se le asigna una dirección IP privada de 10.0.1.50. Además, se especifica que esta interfaz de red estará en el grupo de seguridad allow_web.
 - **Dirección IP elástica (Elastic IP - EIP):**
 - La sección resource "aws_eip" "one" { ... } define una dirección IP elástica (EIP) en AWS. Se especifica que esta dirección IP elástica estará asociada con una interfaz de red específica, en este caso, la interfaz de red interface-terraform. También se especifica que esta dirección IP se asociará con la dirección IP privada 10.0.1.50. La propiedad depends_on asegura que la creación de la dirección IP elástica se realice después de que se haya creado la puerta de enlace a Internet.

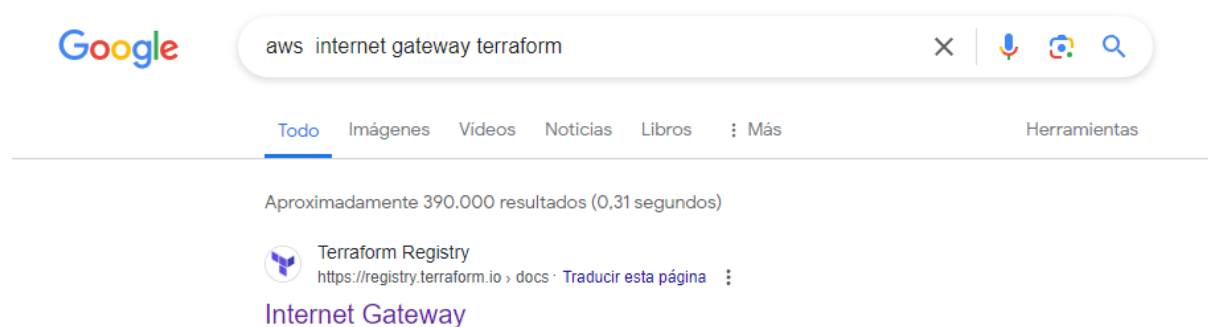
- Y finalmente añadimos el EC2:

```
#Ubuntu Server
resource "aws_instance" "web-server" {
  ami = "ami-051f8a213df8bc089"
  instance_type = "t2.micro"
  availability_zone = "us-east-1a"
  key_name = "main-key"

  network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.interface-terraform.id
  }

  user_data = <<-EOF
      #!/bin/bash
      sudo yum update -y
      sudo yum install httpd -y
      sudo systemctl start httpd
      sudo bash -c 'echo "Tu primer servidor web" >
/var/www/html/index.html'
      EOF
  tags = {
    Name="web-server"
  }
}
```

- Nota: Todas las referencias usadas para especificar el resource y su estructura se consigue buscando "aws [resource que necesites] terraform" y metiéndote a la página oficial.



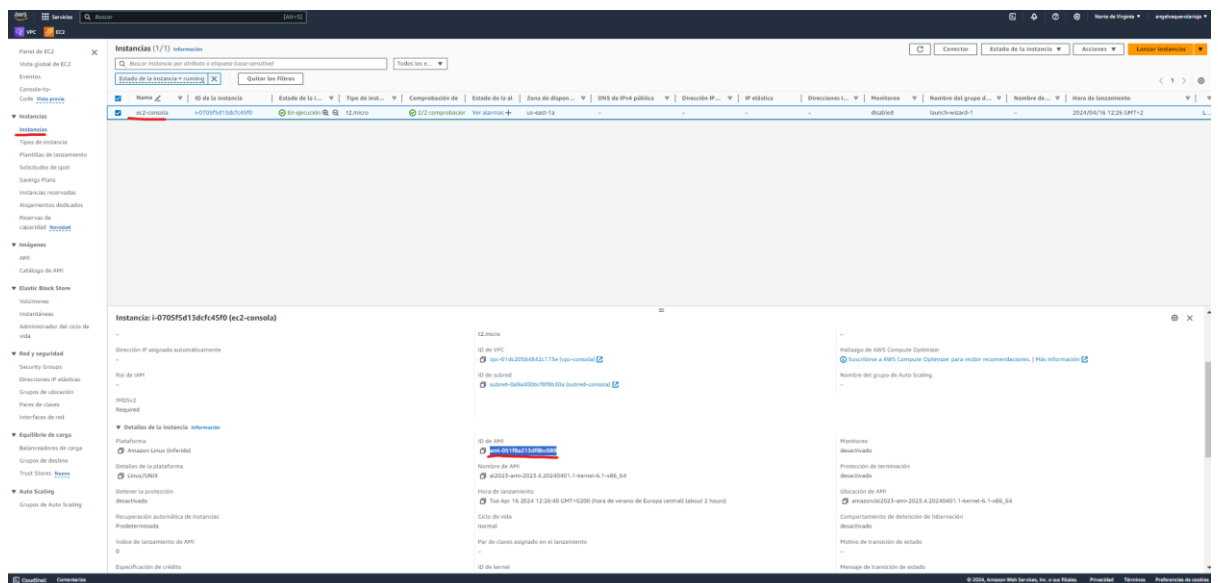
- Explicación del código:

○ **Instancia de Ubuntu Server:**

- La sección resource "aws_instance" "web-server" { ... } define la creación de una instancia de Ubuntu Server en AWS.

○ **AMI (Amazon Machine Image):**

- Se especifica el ID de la AMI de Ubuntu Server que se utilizará para lanzar la instancia. En este caso, la AMI es "ami-051f8a213df8bc089". Dicha AMI se puede conseguir de la siguiente forma:
 - Desde el apartado de instancias puedes ver la AMI que usamos anteriormente, pero otro método sería en el apartado de lanzar instancia y copiar una.

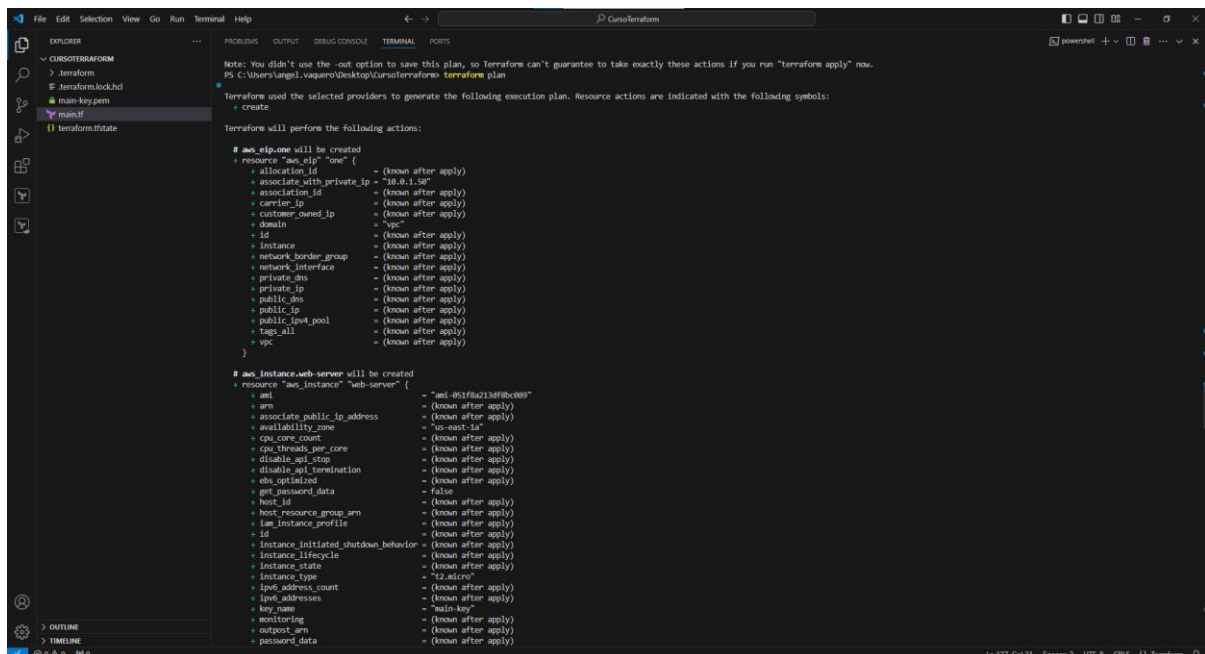


○ **Tipo de instancia (Instance Type):**

- Se especifica el tipo de instancia que se utilizará para la instancia. En este caso, se utiliza t2.micro, que es un tipo de instancia de bajo costo y capacidad limitada.
- **Zona de disponibilidad (Availability Zone):**
 - Se especifica la zona de disponibilidad en la que se lanzará la instancia. En este caso, se utiliza us-east-1a.
- **Nombre de la clave (Key Name):**
 - Se especifica el nombre de la clave SSH que se utilizará para acceder a la instancia. En este caso, se utiliza "main-key".
- **Interfaz de red (Network Interface):**
 - Se especifica la interfaz de red que se asociará con la instancia. En este caso, se utiliza la interfaz de red previamente definida llamada interface-terraform.
- **Datos del usuario (User Data):**
 - Se especifica un script de inicio para configurar la instancia cuando se inicie. En este caso, se actualiza el sistema (sudo yum update -y), se instala el servidor web Apache (sudo yum install httpd -y), se inicia el servicio de Apache (sudo systemctl start httpd) y se crea un archivo HTML básico en el directorio /var/www/html/index.html que muestra "Tu primer servidor web".
- **Etiquetas (Tags):**
 - Se asigna una etiqueta a la instancia con el nombre "web-server".

Ejecutar el código

- Una vez tenemos todo puesto como queramos vamos a la terminal y ejecutamos un “terraform plan” para comprobar que esta todo a nuestro gusto:



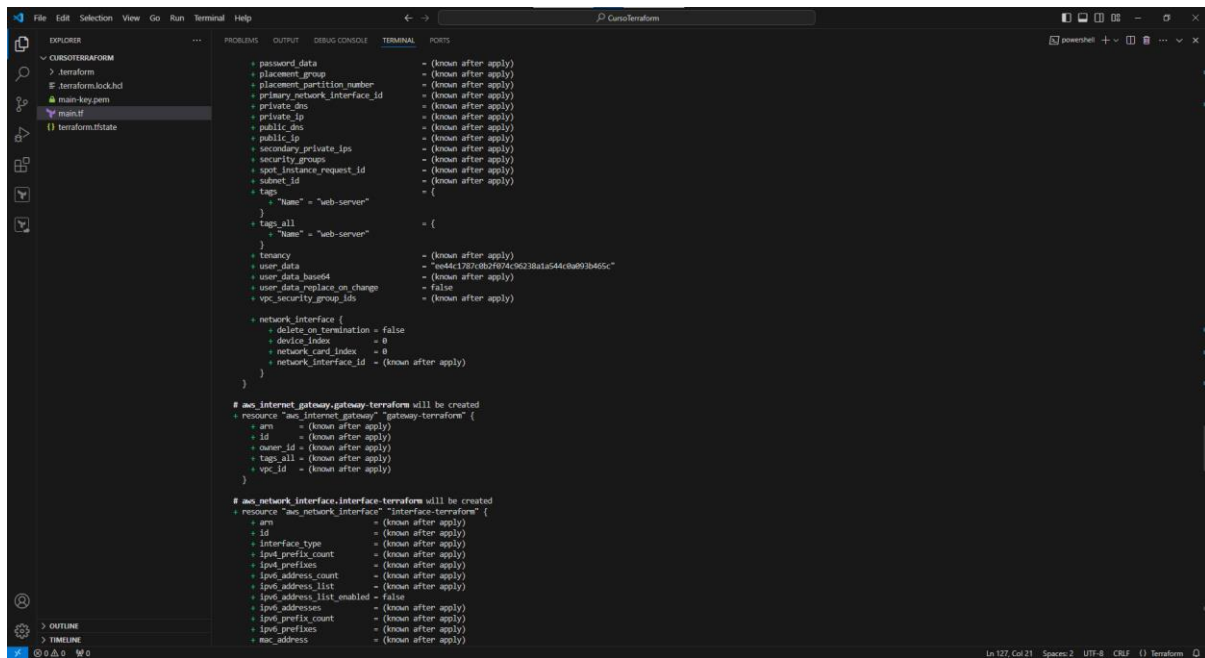
```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS C:\Users\angel.vaquero\Desktop\CursoTerraform> terraform plan

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:

# aws_s3_bucket will be created
+ resource "aws_s3_bucket" "one" {
  + allocation_id      = (known after apply)
  + associate_with_private_ip = "i8-0.1-SF"
  + association_id      = (known after apply)
  + carrier_ip          = (known after apply)
  + customer_owned_ip   = (known after apply)
  + domain              = "vpc"
  + id                  = (known after apply)
  + instance            = (known after apply)
  + network_border_group = (known after apply)
  + network_interface   = (known after apply)
  + private_dns         = (known after apply)
  + private_ip          = (known after apply)
  + public_dns          = (known after apply)
  + public_ip           = (known after apply)
  + public_ipv4_pool     = (known after apply)
  + tags_all            = (known after apply)
  + vpc                 = (known after apply)
}

# aws_instance.web-server will be created
+ resource "aws_instance" "web-server" {
  + ami                    = "ami-053f8a2130f8c080"
  + arn                    = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone       = "us-east-1a"
  + cpu_core_count          = (known after apply)
  + cpu_threads_per_core    = (known after apply)
  + disable_api_stop        = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized           = (known after apply)
  + get_password_data       = false
  + host_id                 = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile    = (known after apply)
  + id                     = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle      = (known after apply)
  + instance_state          = (known after apply)
  + instance_type           = "t2.micro"
  + ipv6_address_count      = (known after apply)
  + ipv6_addresses          = (known after apply)
  + key_name                = "main-key"
  + monitoring              = (known after apply)
  + outpost_arn             = (known after apply)
  + password_data           = (known after apply)
}
```



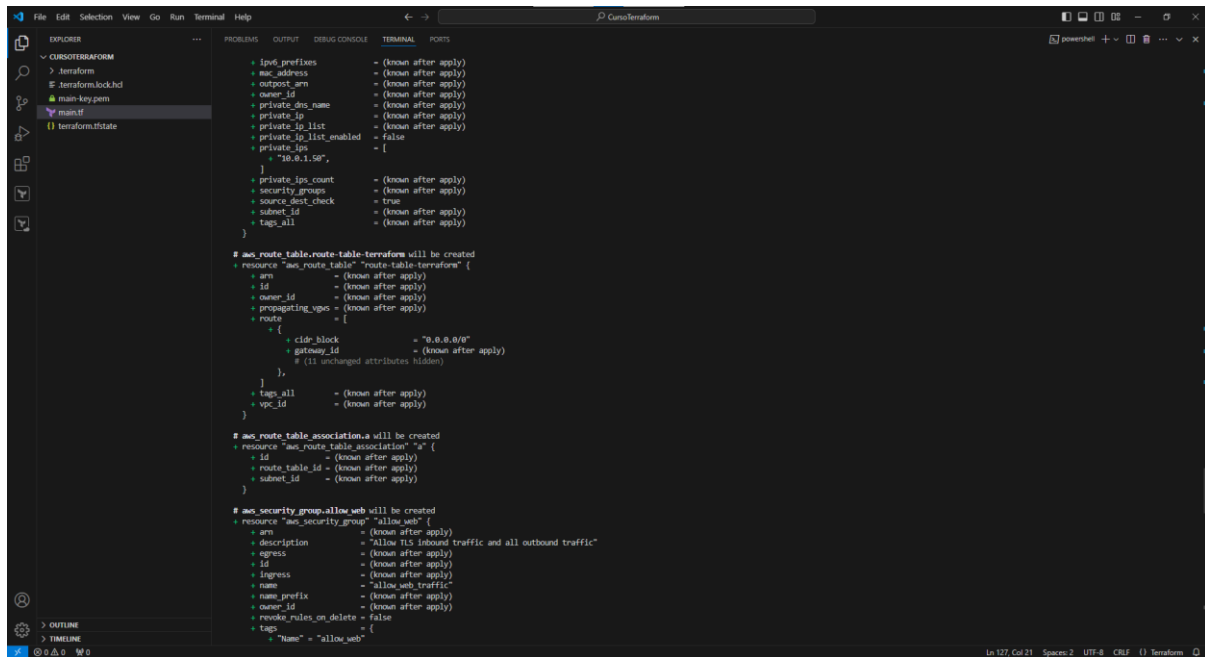
```
File Edit Selection View Go Run Terminal Help
C:\Users\angel\source\repos\terraform-aws-ec2\terraform-aws-ec2
main.tf
terraform.tfstate

+ password_data = (known after apply)
+ placement_group = (known after apply)
+ placement_partition_number = (known after apply)
+ primary_network_interface_id = (known after apply)
+ private_dns = (known after apply)
+ private_ip = (known after apply)
+ public_dns = (known after apply)
+ public_ip = (known after apply)
+ secondary_private_ips = (known after apply)
+ security_groups = (known after apply)
+ spot_instance_request_id = (known after apply)
+ subnet_id = (known after apply)
+ tags = {
  + "Name" = "web-server"
}
+ tags_all = {
}
+ tenancy = (known after apply)
+ user_data = "cloud-init/2024/04/06/2024040609040604"
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)

+ network_interface {
  + deletion_protection = false
  + device_index = 0
  + network_card_index = 0
  + network_interface_id = (known after apply)
}

# aws_internet_gateway.gateway-terraform will be created
+ resource "aws_internet_gateway" "gateway-terraform" {
+   arn = (known after apply)
+   id = (known after apply)
+   owner_id = (known after apply)
+   tags_all = (known after apply)
+   vpc_id = (known after apply)
}

# aws_network_interface.interface-terraform will be created
+ resource "aws_network_interface" "interface-terraform" {
+   arn = (known after apply)
+   id = (known after apply)
+   interface_type = (known after apply)
+   ipv4_prefix_count = (known after apply)
+   ipv4_prefixes = (known after apply)
+   ipv4_address_count = (known after apply)
+   ipv4_addresses_list = (known after apply)
+   ipv4_address_list_enabled = false
+   ipv4_addresses = (known after apply)
+   ipv4_prefix_count = (known after apply)
+   ipv4_prefixes = (known after apply)
+   mac_address = (known after apply)
}
```



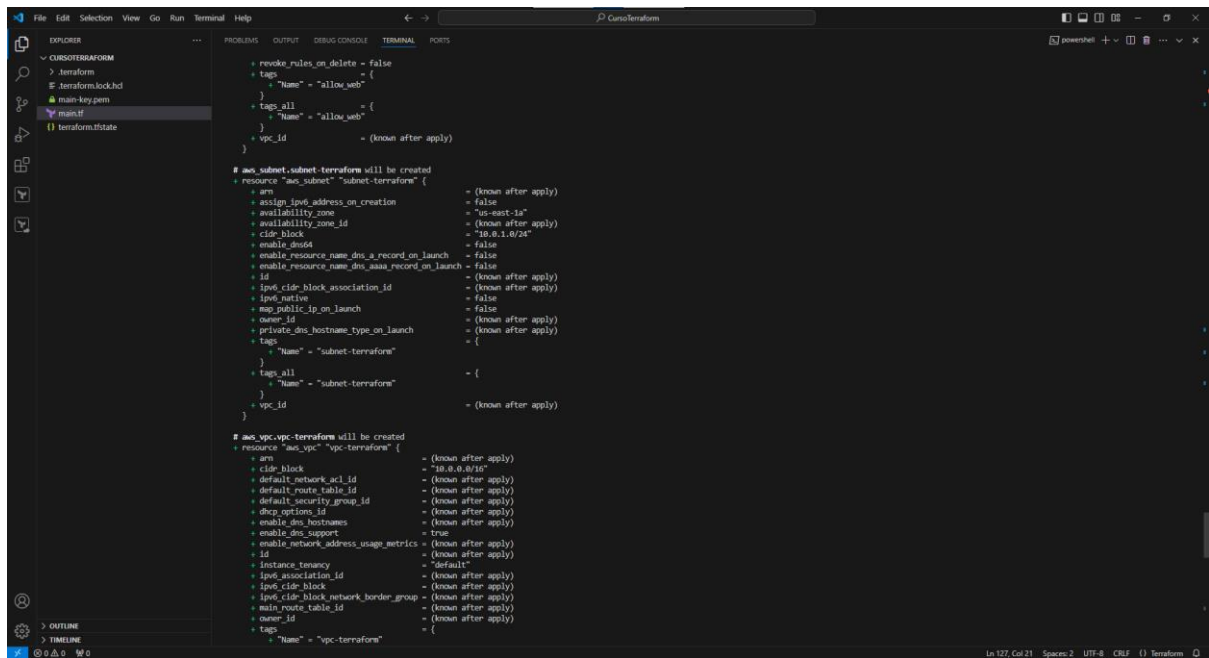
```
File Edit Selection View Go Run Terminal Help
C:\Users\angel\source\repos\terraform-aws-ec2\terraform-aws-ec2
main.tf
terraform.tfstate

+ ipv4_prefixes = (known after apply)
+ mac_address = (known after apply)
+ outposts_arn = (known after apply)
+ owner_id = (known after apply)
+ private_dns_name = (known after apply)
+ private_ip = (known after apply)
+ private_ip_list = (known after apply)
+ private_ip_list_enabled = false
+ private_ips = [
  + "1b.0.1.5a",
]
+ private_ips_count = (known after apply)
+ security_groups = (known after apply)
+ source_dest_check = true
+ subnet_id = (known after apply)
+ tags_all = (known after apply)
}

# aws_route_table.route-terraform will be created
+ resource "aws_route_table" "route-table-terraform" {
+   arn = (known after apply)
+   id = (known after apply)
+   owner_id = (known after apply)
+   propagating_vpcs = (known after apply)
+   routes = [
    + {
      + cidr_block = "0.0.0.0/0"
      + gateway_id = (known after apply)
    },
  ]
+ tags_all = (known after apply)
+ vpc_id = (known after apply)
}

# aws_route_table_association.a will be created
+ resource "aws_route_table_association" "a" {
+   id = (known after apply)
+   route_table_id = (known after apply)
+   subnet_id = (known after apply)
}

# aws_security_group.allow_web will be created
+ resource "aws_security_group" "allow_web" {
+   arn = (known after apply)
+   description = "allow TLS inbound traffic and all outbound traffic"
+   egress = (known after apply)
+   id = (known after apply)
+   ingress = (known after apply)
+   name = "allow_web_traffic"
+   name_prefix = (known after apply)
+   owner_id = (known after apply)
+   revoke_rules_on_delete = false
+   tags = {
    + "Name" = "allow_web"
  }
}
```



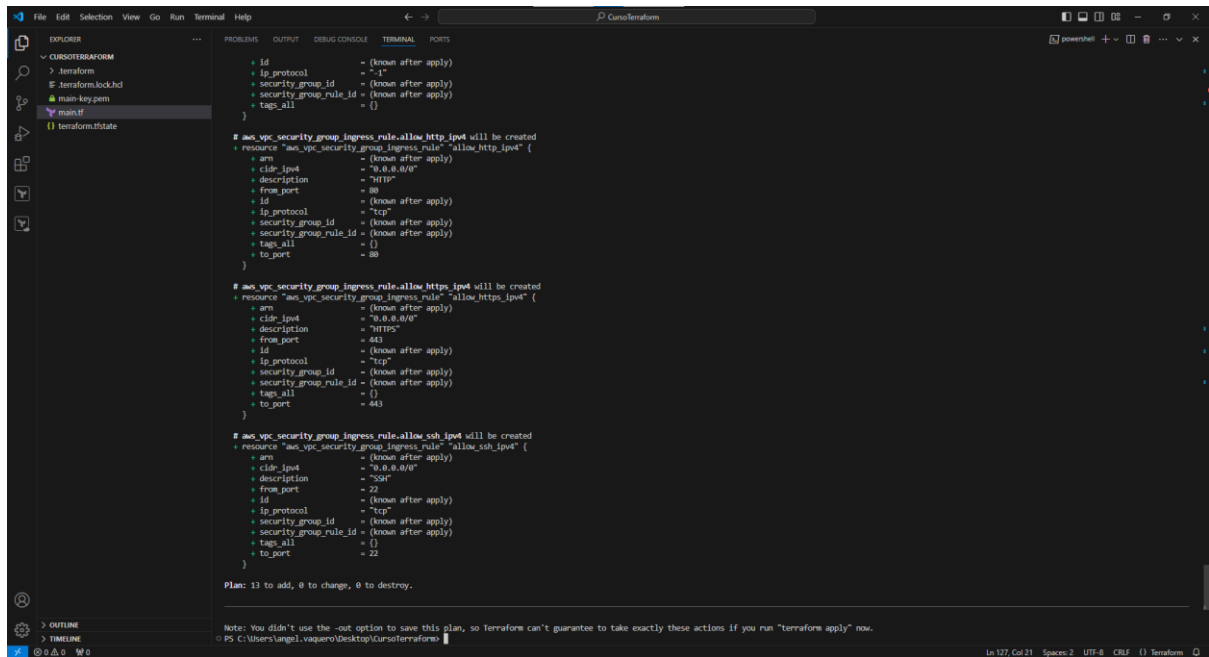
This screenshot shows the Visual Studio Code interface with a Terraform configuration file open. The Explorer pane on the left shows the project structure: `CURSORTERRAFORM`, `.terraform`, `.terraform.lock.hcl`, `main-key.pem`, `main.tf`, and `terraform.tfstate`. The main editor displays the Terraform code for creating an AWS VPC and two subnets. The code includes comments indicating the order of resource creation and the state of resources after application.

```
terraform {
  backend "s3" {
    bucket = "terraform-state"
    key     = "terraform.tfstate"
  }
}

provider "aws" {
  region = "us-east-1"
}

# aws_subnet-terraform will be created
resource "aws_subnet" "subnet-terraform" {
  vpc_id            = (known after apply)
  cidr_block        = "10.0.0.0/16"
  availability_zone  = "us-east-1a"
  map_public_ip_on_launch = false
  enable_dns64       = false
  enable_resource_name_dns_record_on_launch = false
  enable_dns_hostnames = false
  id                 = (known after apply)
  ipv6_cidr_block_association_id = (known after apply)
  ipv6_native         = false
  map_public_ip_on_launch = false
  owner_id             = (known after apply)
  private_dns_hostname_type_on_launch = (known after apply)
  tags                = {
    "Name" = "subnet-terraform"
  }
  tags_all            = {
    "Name" = "subnet-terraform"
  }
  vpc_id              = (known after apply)
}

# aws_vpc-terraform will be created
resource "aws_vpc" "vpc-terraform" {
  cidr_block = "10.0.0.0/16"
  default_network_acl_id = (known after apply)
  default_route_table_id = (known after apply)
  default_security_group_id = (known after apply)
  dhcp_options_id = (known after apply)
  enable_dns_hostnames = (known after apply)
  enable_dns_support = true
  enable_network_address_usage_metrics = (known after apply)
  id = (known after apply)
  instance_tenancy = "default"
  ipv6_association_id = (known after apply)
  ipv6_cidr_block = (known after apply)
  ipv6_cidr_block_network_border_group = (known after apply)
  main_route_table_id = (known after apply)
  owner_id = (known after apply)
  tags = {
    "Name" = "vpc-terraform"
  }
}
```



This screenshot shows the Visual Studio Code interface with a Terraform configuration file open. The Explorer pane on the left shows the project structure: `CURSORTERRAFORM`, `.terraform`, `.terraform.lock.hcl`, `main-key.pem`, `main.tf`, and `terraform.tfstate`. The main editor displays the Terraform code for creating three security group ingress rules for the VPC. The code includes comments indicating the order of resource creation and the state of resources after application.

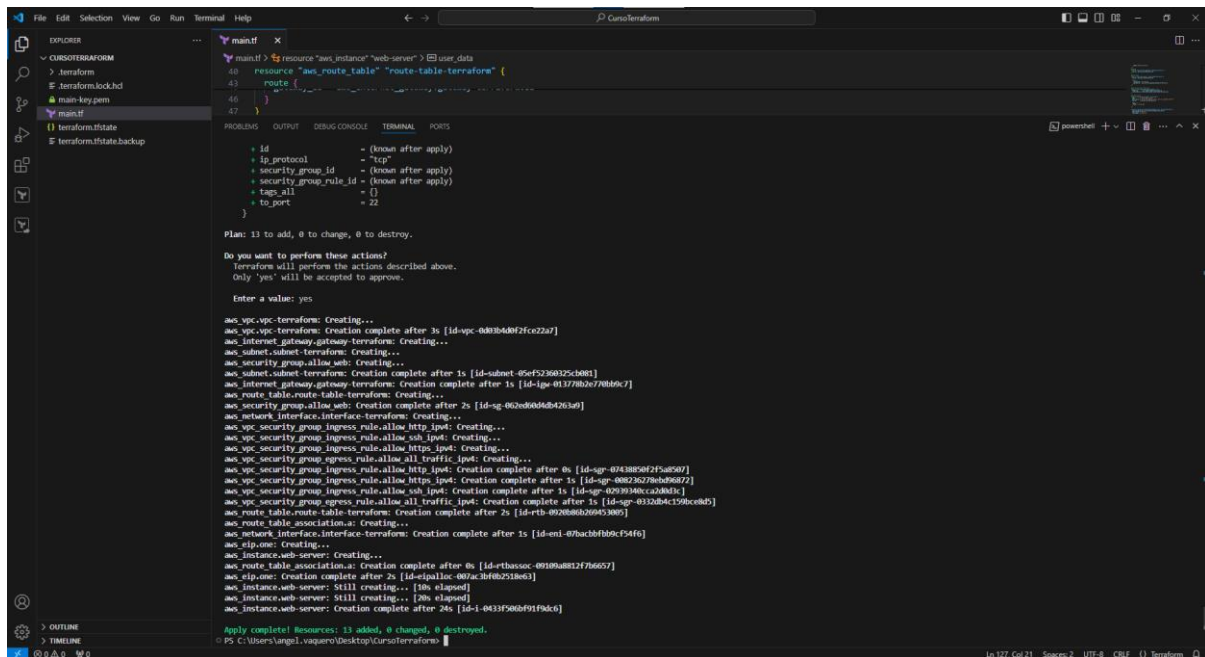
```
resource "aws_vpc_security_group_ingress_rule" "allow_http_ipv4" {
  cidr_ipv4 = "0.0.0.0/0"
  from_port = 80
  id = (known after apply)
  ip_protocol = "tcp"
  security_group_id = (known after apply)
  security_group_rule_id = (known after apply)
  tags_all = {}
  to_port = 80
}

# aws_vpc_security_group_ingress_rule-allow_https_ipv4 will be created
resource "aws_vpc_security_group_ingress_rule" "allow_https_ipv4" {
  cidr_ipv4 = "0.0.0.0/0"
  from_port = 443
  id = (known after apply)
  ip_protocol = "tcp"
  security_group_id = (known after apply)
  security_group_rule_id = (known after apply)
  tags_all = {}
  to_port = 443
}

# aws_vpc_security_group_ingress_rule-allow_ssh_ipv4 will be created
resource "aws_vpc_security_group_ingress_rule" "allow_ssh_ipv4" {
  cidr_ipv4 = "0.0.0.0/0"
  from_port = 22
  id = (known after apply)
  ip_protocol = "tcp"
  security_group_id = (known after apply)
  security_group_rule_id = (known after apply)
  tags_all = {}
  to_port = 22
}

Plan: 13 to add, 0 to change, 0 to destroy.
```

- Una vez nos hemos asegurado de que todo está a nuestro gusto ejecutamos un “terraform apply” para aplicar los cambios.
- Nota: Para ejecutar y que todo vaya perfecto recomendable revisar la zona donde lo ejecutas, borra la anterior EC2 creada y su VPC y ya luego ejecutar.
- Si realizaste los pasos correctamente el restado de ejecutar ese comando será:



```
main.tf > terraform apply -var user_data=...
resource "aws_instance" "web-server" {
  user_data = user_data
}

resource "aws_route_table" "route-table-terraform" {
  route {
    destination_prefix_list_id = aws_prefix_list.prefix-list-terraform.id
    gateway_id = aws_internet_gateway.internet-gateway-terraform.id
  }
}

Plan: 13 to add, 0 to change, 0 to destroy.

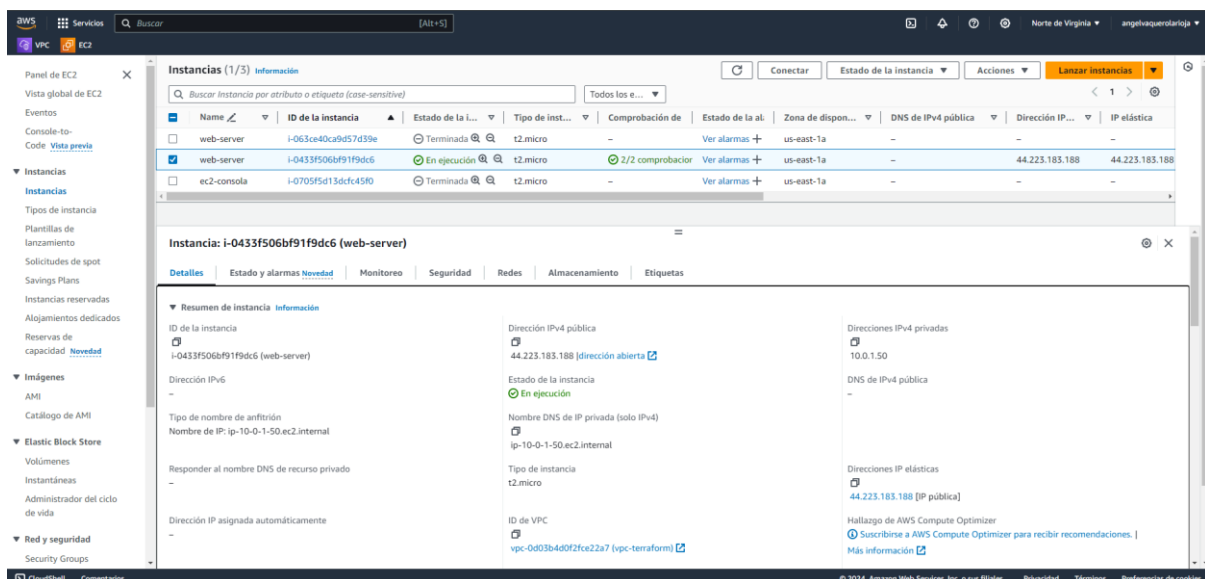
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.vpc-terraform: Creating...
aws_vpc.vpc-terraform: Creation complete after 1s [id=vpc-0d83b4d0f2fce22a7]
aws_internet_gateway.internet-gateway-terraform: Creating...
aws_subnet.subnet-terraform: Creating...
aws_security_group.allow-web: Creating...
aws_subnet.subnet-terraform: Creation complete after 1s [id=subnet-456f523603255081]
aws_internet_gateway.internet-gateway-terraform: Creation complete after 1s [id=igw-0137f80e770b06c7]
aws_route_table.route-table-terraform: Creating...
aws_security_group.allow-web: Creation complete after 2s [id=sg-062d000d0b4263a9]
aws_network_interface.interface-terraform: Creating...
aws_vpc_security_group_ingress_rule.allow-http-ipv4: Creating...
aws_vpc_security_group_ingress_rule.allow-ssh-ipv4: Creating...
aws_vpc_security_group_egress_rule.allow-all-traffic-ipv4: Creating...
aws_vpc_security_group_ingress_rule.allow-http-ipv4: Creation complete after 0s [id=sg-074380f0f2f5a560]
aws_vpc_security_group_ingress_rule.allow-ssh-ipv4: Creation complete after 1s [id=sg-00823c078abdb6072]
aws_vpc_security_group_ingress_rule.allow-ssh-ipv4: Creation complete after 1s [id=sg-00930340cca0dddc]
aws_vpc_security_group_egress_rule.allow-all-traffic-ipv4: Creation complete after 2s [id=sg-0332dbd4c159bc0d5]
aws_route_table.route-table-terraform: Creation complete after 2s [id=rtb-092680b0269433009]
aws_route_table_association.a: Creating...
aws_network_interface.interface-terraform: Creation complete after 1s [id=eni-07bacbf0b9cf54f6]
aws_elb.elb: Creating...
aws_instance.web-server: Creating...
aws_route_table_association.a: Creation complete after 0s [id=rtbassoc-0910948312f76657]
aws_elb.elb: Creation complete after 2s [id=elb-allos-007c2bf6b2518ed3]
aws_instance.web-server: Still creating... [10s elapsed]
aws_instance.web-server: Still creating... [20s elapsed]
aws_instance.web-server: Creation complete after 24s [id=i-0433f506bf91f9dc6]

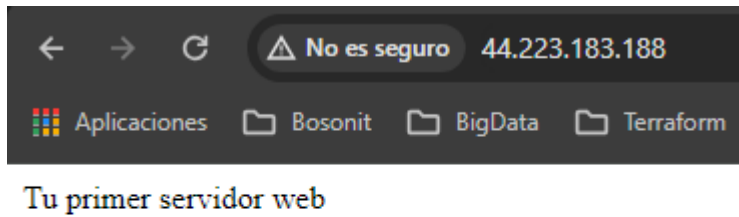
Apply complete! Resources: 13 added, 0 changed, 0 destroyed.
```

- Revisamos en AWS si se creó:



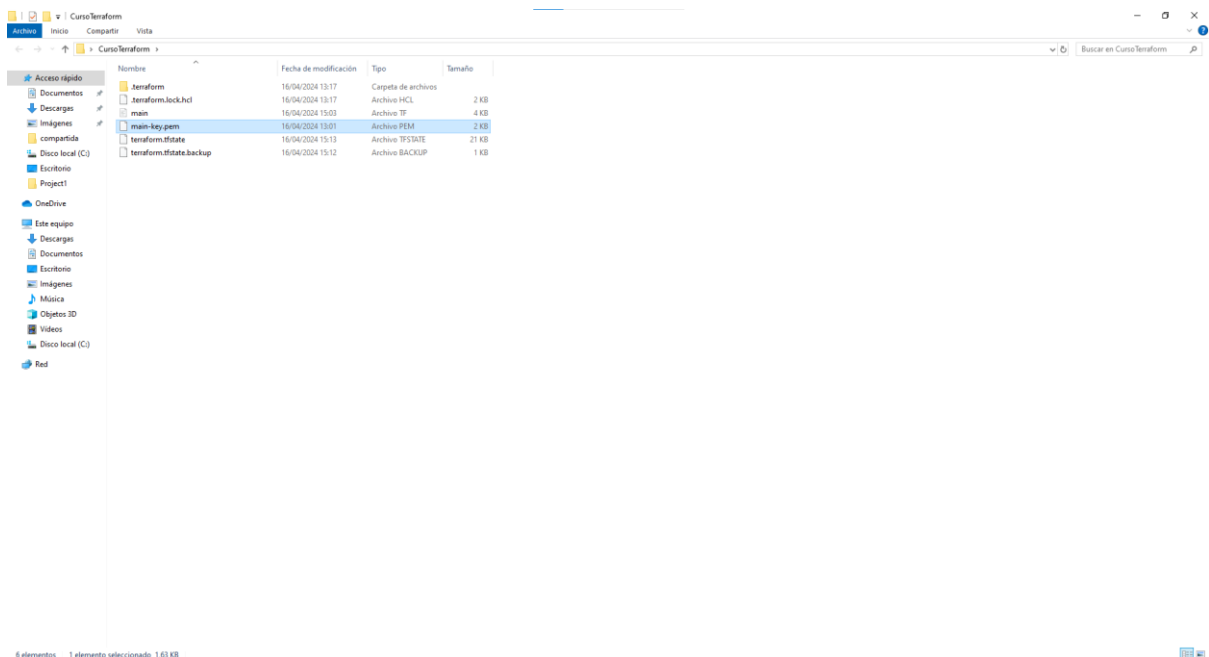
Autor: Ángel Vaquero Toncheva

- Hacemos una prueba buscando la ip pública en http y si se visualiza lo que pusimos en index.html todo está correcto:



Conexión mediante SSH desde terminal

- Para realizar este paso necesitaremos ir a donde tengamos guardada nuestra clave para acceder al EC2, en nuestro caso está en la carpeta del curso:



Autor: Ángel Vaquero Toncheva

- Desde ahí abrimos terminal:

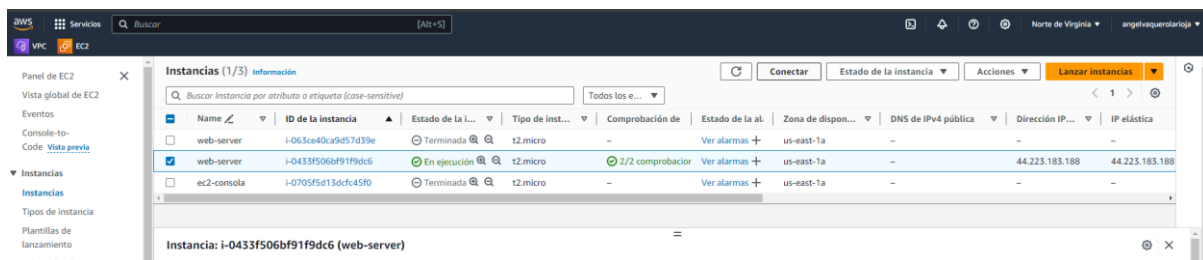
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.4291]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\angel.vaquero\Desktop\CursoTerraform>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: F89E-B95D

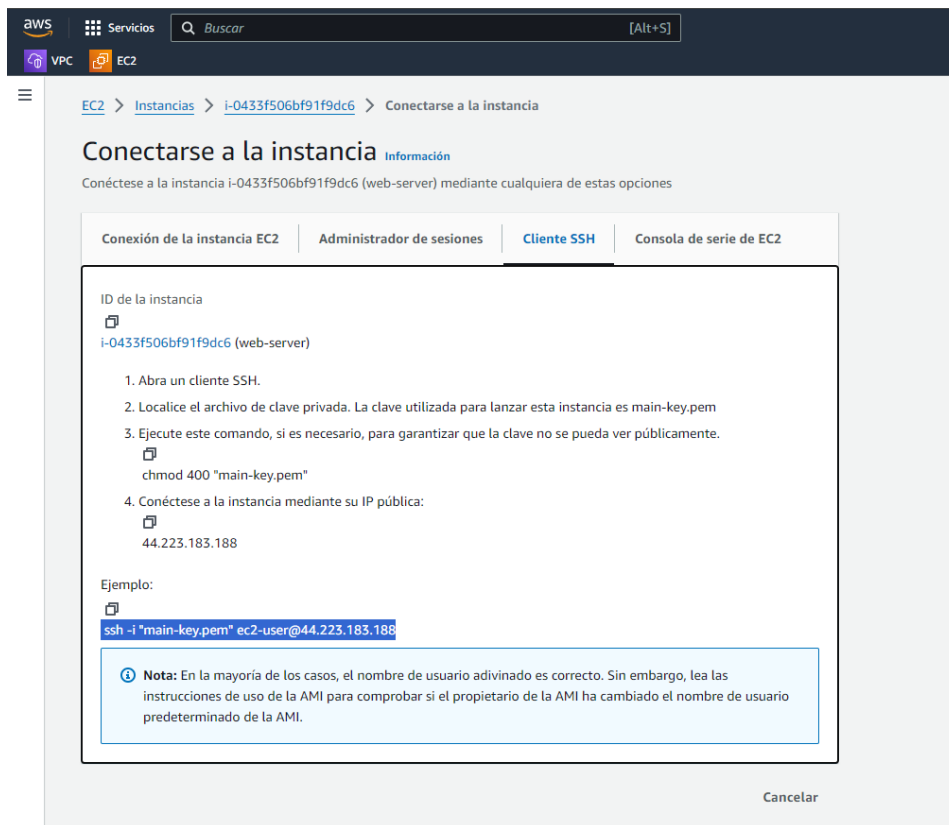
Directorio de C:\Users\angel.vaquero\Desktop\CursoTerraform
16/04/2024  15:13  <DIR>          .
16/04/2024  15:13  <DIR>          ..
16/04/2024  13:17  <DIR>          .terraform
16/04/2024  13:17             1.406 .terraform.lock.hcl
16/04/2024  13:01             1.674 main-key.pem
16/04/2024  15:03             3.635 main.tf
16/04/2024  15:13          20.887 terraform.tfstate
16/04/2024  15:12          181 terraform.tfstate.backup
                5 archivos          27.783 bytes
                3 dirs 124.473.450.496 bytes libres

C:\Users\angel.vaquero\Desktop\CursoTerraform>
```

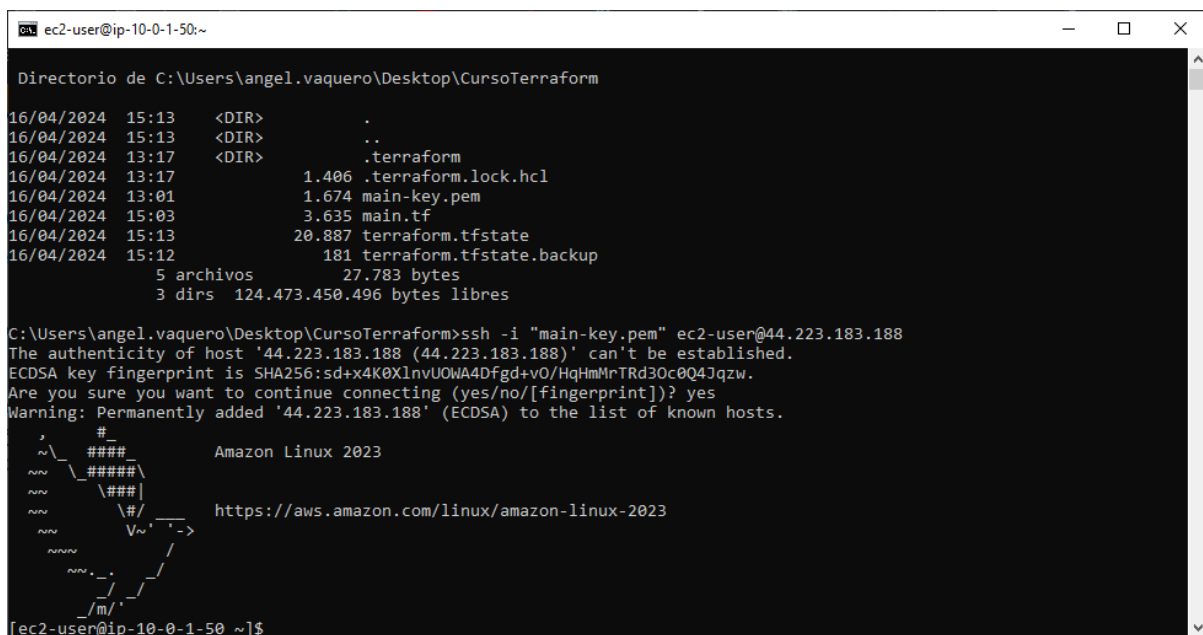
- A continuación, necesitamos saber de qué forma conectarnos a la maquina desde ssh y eso se logra desde el apartado de la EC2 y dándole a conectar:



- Y en esta ventana se te explica:



- Y ejecutamos el comando seleccionado:



Autor: Ángel Vaquero Toncheva

- Hacemos unas comprobaciones de que las ordenes que le indicamos en el apartado de `user_data` de `aws_instance`:

```
ec2-user@ip-10-0-1-50: /var/www/html
The authenticity of host '44.223.183.188 (44.223.183.188)' can't be established.
ECDSA key fingerprint is SHA256:sd+x4K0XlnvUOWA4Dfgd+vO/HqHmMrTRd30c0Q4Jqzw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.223.183.188' (ECDSA) to the list of known hosts.
#
##### Amazon Linux 2023
#####
#####
##### \##\
##### \|
##### \#/ https://aws.amazon.com/linux/amazon-linux-2023
##### Vw' '->
#####
#####
##### m/' -
#####

[ec2-user@ip-10-0-1-50 ~]$ pwd
/home/ec2-user
[ec2-user@ip-10-0-1-50 ~]$ cd /
[ec2-user@ip-10-0-1-50 /]$ ls www/
ls: cannot access 'www/': No such file or directory
[ec2-user@ip-10-0-1-50 /]$ ls
bin boot dev etc home lib lib64 local media mnt opt proc root run sbin srv sys tmp usr var
[ec2-user@ip-10-0-1-50 /]$ cd var/www
[ec2-user@ip-10-0-1-50 www]$ ls
cgi-bin html
[ec2-user@ip-10-0-1-50 www]$ cd html/
[ec2-user@ip-10-0-1-50 html]$ ls
index.html
[ec2-user@ip-10-0-1-50 html]$ cat index.html
Tu primer servidor web
[ec2-user@ip-10-0-1-50 html]$
```

- Y con esto concluiríamos la explicación del caso práctico de Terraform con AWS.

Investigación

1. ¿Puedes pasar archivos mediante ssh al EC2 creado? ¿De qué forma?
2. ¿Se puede acceder al EC2 de otra forma? Indica cuales y como se realiza
3. Investiga como hacer el borrado de todo lo realizado en el curso para evitar cobros inesperados
4. ¿Qué uso le darías tu al EC2?