Implementación de infraestructura en la nube con AWS para RunFood

Corporación Universitaria Remington.
Ingeniería de sistemas
SEMINARIO AMAZON AWS

Karen Pérez Arango
Seminario de grado
2024.

Tabla de contenido

Resumen

El presente trabajo de grado describe la implementación de una arquitectura en la nube utilizando los servicios de Amazon Web Services (AWS) para apoyar el crecimiento tecnológico de la startup RunFood que conecta restaurantes con clientes mediante entregas rápidas. El objetivo principal es diseñar una infraestructura escalable, altamente disponible y con tiempos de respuesta mejorados. Se utilizaron servicios como EC2 para instancias de servidores virtuales, VPC para red privada, Load Balancer (LB) para distribución de tráfico, y Auto Scaling Group para ajuste dinámico de la capacidad. Además, se integraron tecnologías como Docker y NGINX para manejar microservicios y balanceo de cargas interno. Este proyecto demuestra la aplicación práctica de conceptos aprendidos en el seminario de AWS.

Palabras clave

AWS
Arquitectura en la nube
Auto Scaling
Docker
NGINX

Marco conceptual y contextual

AWS: Servicios de computación de Amazon ejecutados en la nube.

EC2: Servicio que permite desplegar instancias de servidores virtuales.

VPC: Agrupación de recursos que comparten un direccionamiento IP.

Load Balancer y Auto Scaling: Garantizan alta disponibilidad y respuesta eficiente ante variaciones en la demanda.

Docker y NGINX: Herramientas para la implementación de contenedores y gestión eficiente del tráfico mediante balanceo de cargas.

Key Pair: Clave utilizada para la autenticación segura de instancias mediante SSH.

Amazon Machine Image (AMI): Plantilla preconfigurada que contiene el sistema operativo, software y configuraciones necesarias y establecidas para lanzar nuevas instancias.

Target Group: Grupo de instancias asociadas al balanceador de carga para distribuir el tráfico entre ellas según las reglas configuradas.

Sanpshots: Copias de seguridad de los volúmenes que permiten crear nuevas instancias o restaurar datos.

La solución fue diseñada para la startup RunFood que busca conectar restaurantes con clientes de manera eficiente. Ante el aumento de usuarios y demanda, se requiere una infraestructura que asegure continuidad del servicio y tiempos de respuesta bajos. El proyecto pone en práctica habilidades aprendidas en el seminario de AWS, aplicando principios de diseño de soluciones escalables y seguras en la nube.

Desarrollo e implementación del aprendizaje

Recursos de AWS para ejecutar instancias

1. **EC2**

Recurso principal de amazon que permite ejecutar instancias es el servicio **EC2**. Este recurso permite desplegar y administrar instancias de servidores virtuales en la nube, este servicio maneja familia de instancias y tipos de instancias. Este recuros se usa normalmente para aplicaciones, servidores web y bases de datos.

2. **Amazon Machine Image (AMI)**

Una AMI es una plantilla preconfigurada que contiene lo necesario para lanzar una instancia:

Sistema operativo: Linux (Ubuntu, Amazon Linux, etc.) o Windows.

Software: Aplicaciones, frameworks y configuraciones personalizadas.

Puedes elegir entre:

AMIs prediseñadas por AWS o terceros.

AMIs personalizadas que tú mismo creas.

3. **Grupos de Seguridad (Security Groups)**

Actúan como un firewall virtual para las instancias EC2.

Controlan qué tráfico puede entrar o salir de la instancia.

Ejemplo de reglas entrantes: Permitir SSH (puerto 22) solo desde tu IP.

Ejemplo de reglas salientes: Permitir todas las conexiones hacia Internet.

Flexible y fácil de configurar. Puedes modificar las reglas en tiempo real.

4. **Key Pairs**

Claves de seguridad que permiten acceder a las instancias EC2 de forma segura.

Compuestas por:

Clave pública: Se almacena en la instancia.

Clave privada: La descargas al crear el par y la usas para conectarte vía SSH o RDP.

Sin la clave privada, no podrás acceder a la instancia.

5. **Elastic Block Store (EBS)**

Proporciona almacenamiento persistente para instancias EC2.

Cada instancia puede tener uno o más volúmenes EBS adjuntos.

Ejemplo: Tu instancia EC2 tiene un volumen EBS donde se almacena el sistema operativo y los datos.

Ventajas:

Los datos persisten incluso si la instancia se detiene.

Se pueden realizar respaldos mediante Snapshots.

6. **Elastic IP**

Es una dirección IP estática y pública que puedes asignar a una instancia EC2.

Ideal para casos en los que necesitas una dirección IP fija, como:

Servidores web accesibles públicamente.

Aplicaciones críticas que no deben cambiar de IP al reiniciar.

7. **Amazon Virtual Private Cloud (VPC)**

Crea una red virtual donde se ejecutan tus instancias.

Ofrece control sobre la configuración de red:

Subredes: Redes internas dentro de tu VPC.

Tablas de enrutamiento: Definen cómo el tráfico fluye dentro y fuera de la VPC.

Internet Gateway: Permite acceso a Internet.

NAT Gateway: Da acceso a Internet a instancias en subredes privadas.

8. **Auto Scaling Groups (ASG)**

Permiten escalar automáticamente la cantidad de instancias EC2 según la demanda:

Ejemplo: Aumentar instancias cuando hay más tráfico y reducirlas cuando baja.

Configuración básica:

Minimizar costos: Solo usas los recursos necesarios.

Alta disponibilidad: Se asegura de que siempre haya instancias funcionando.

9**. Elastic Load Balancer (ELB)**

Es un servicio que distribuye el tráfico de red entre varias instancias EC2.

Ayuda a mejorar la disponibilidad y redundancia de tu aplicación.

Tipos de balanceadores:

Application Load Balancer: Para tráfico HTTP/HTTPS, ideal para aplicaciones web.

Network Load Balancer: Para tráfico de red de baja latencia.

Gateway Load Balancer: Para balancear tráfico hacia firewalls virtuales.

10. **AWS Systems Manager**

Es un servicio que simplifica la administración y operación de instancias EC2.

Funciones clave:

Patch Manager: Instala actualizaciones de seguridad.

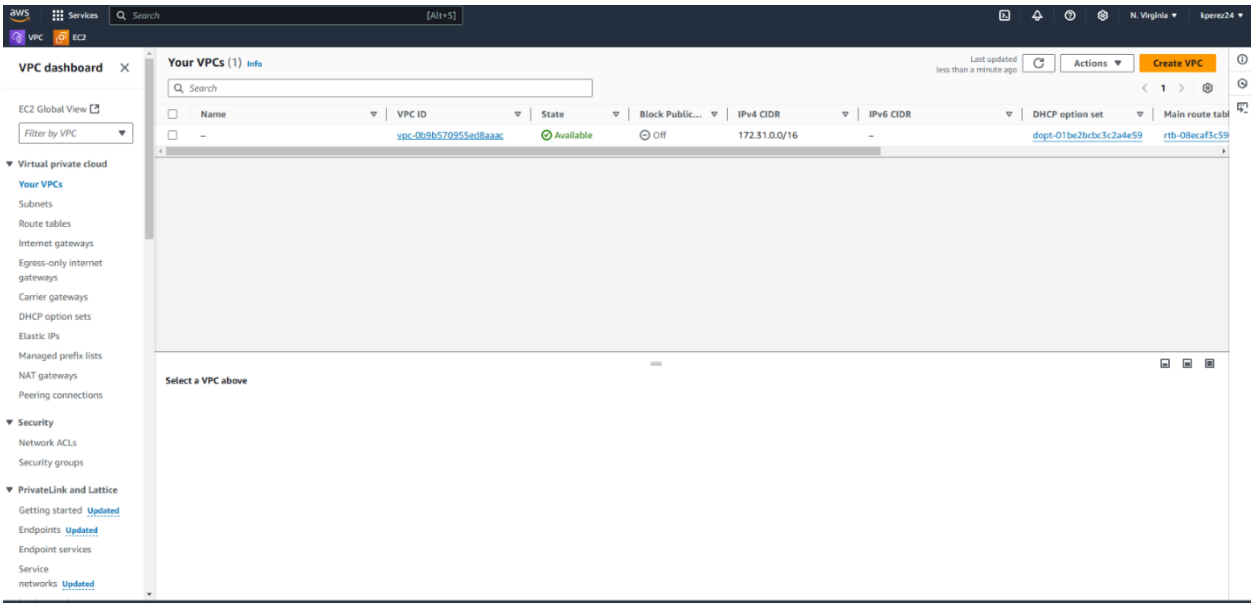
Session Manager: Permite acceso remoto seguro sin usar claves SSH.

Run Command: Ejecuta comandos o scripts en múltiples instancias simultáneamente.

Implementación de servicios AWS

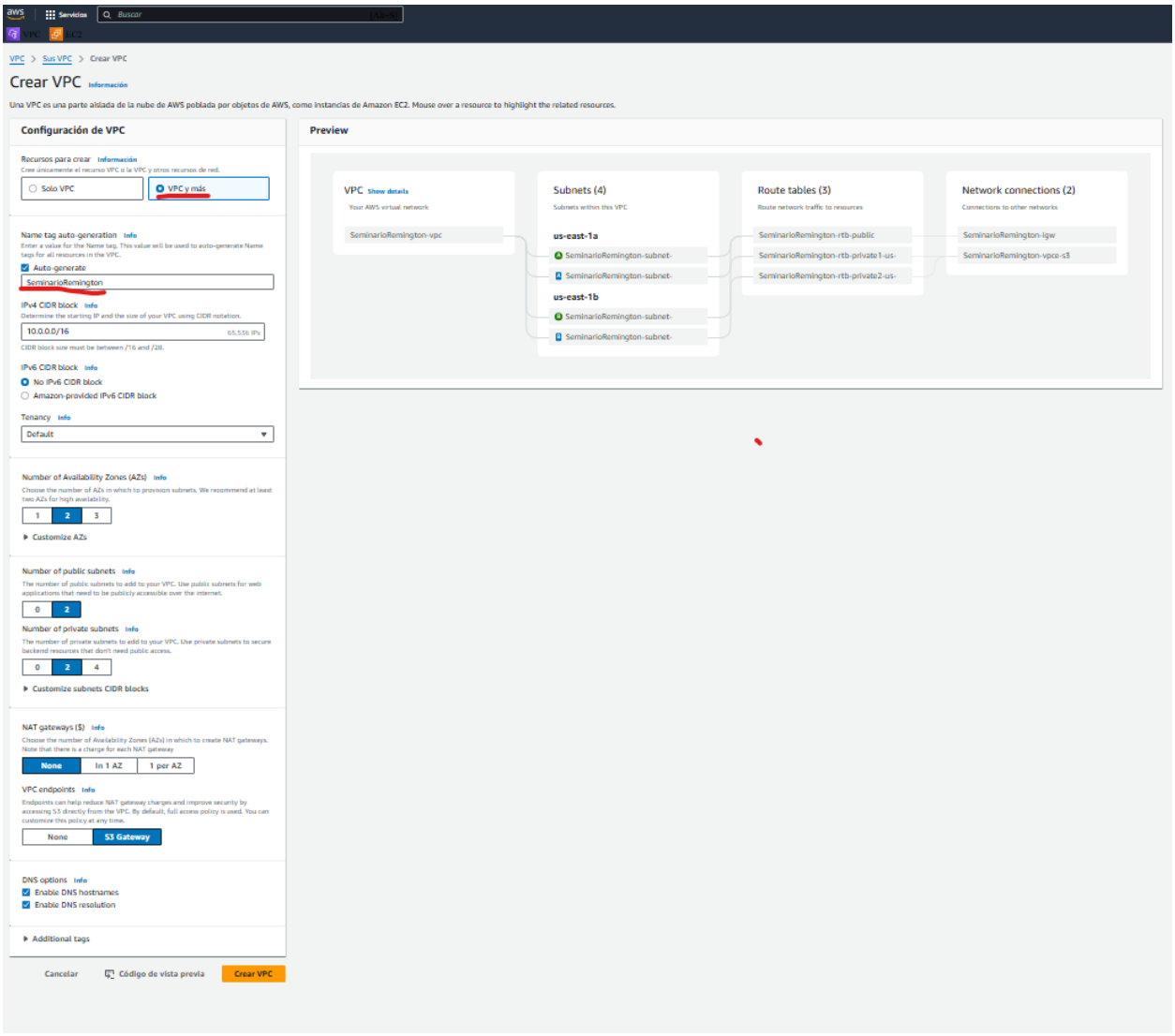
VPC

Para crear un VPC ingresamos al servicio de VPC en AWS y seleccionamos la opción de tus VPCS



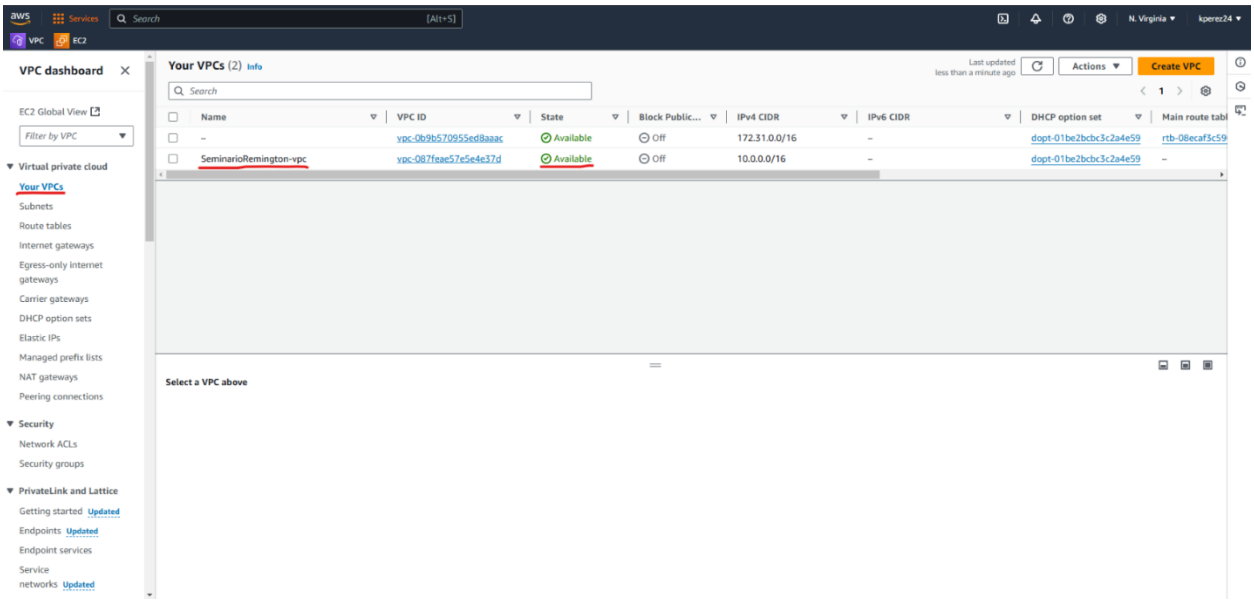
Fuente: elaboración propia.

Al darle crear vemos el menú de creación, seleccionamos VPC y más y luego ponemos un nombre, la demás información la dejamos como viene por defecto y damos crear.



Fuente: elaboración propia.

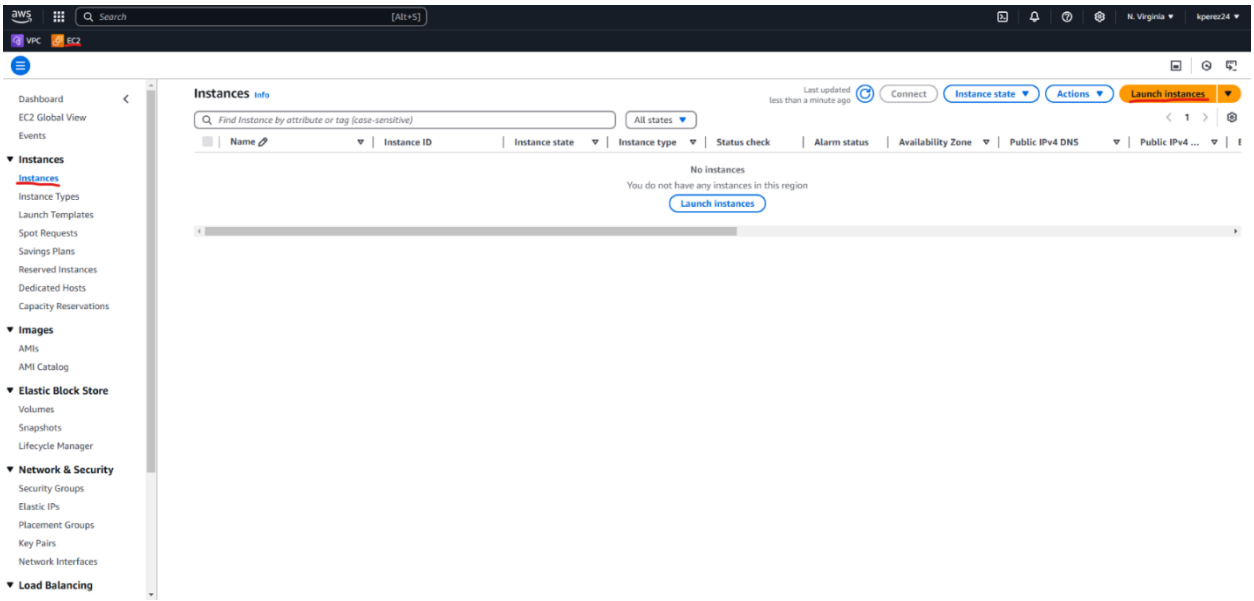
Verificamos en el apartado de tus VPCs y verificamos que se haya creado sin problemas y su estado esté disponible



Fuente: elaboración propia.

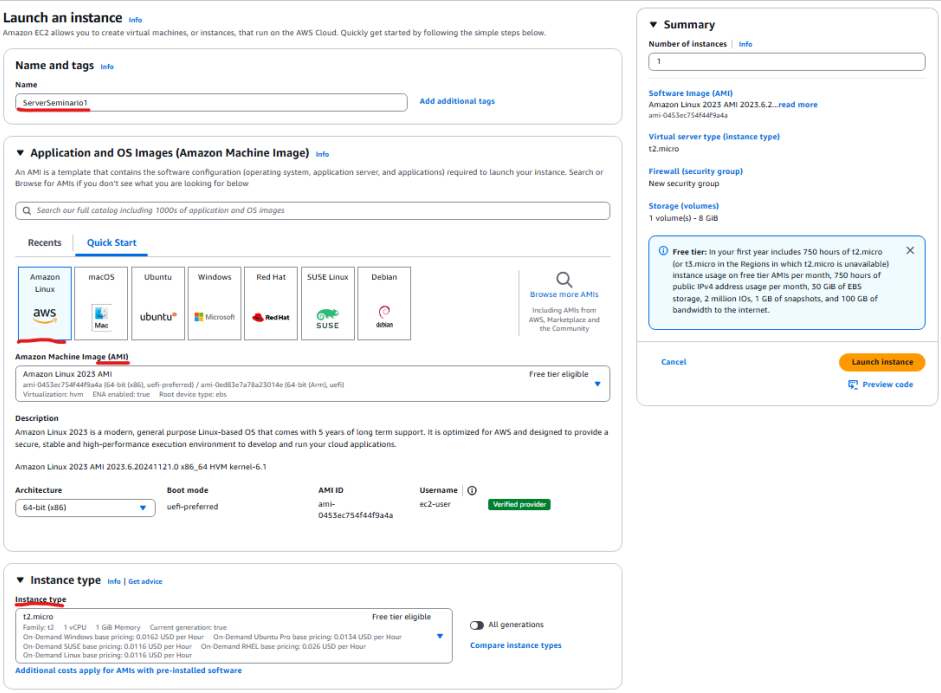
Instancias (EC2)

Para crear instancias ingresamos al servicio EC2, Instancias y damos clic en lanzar instancia



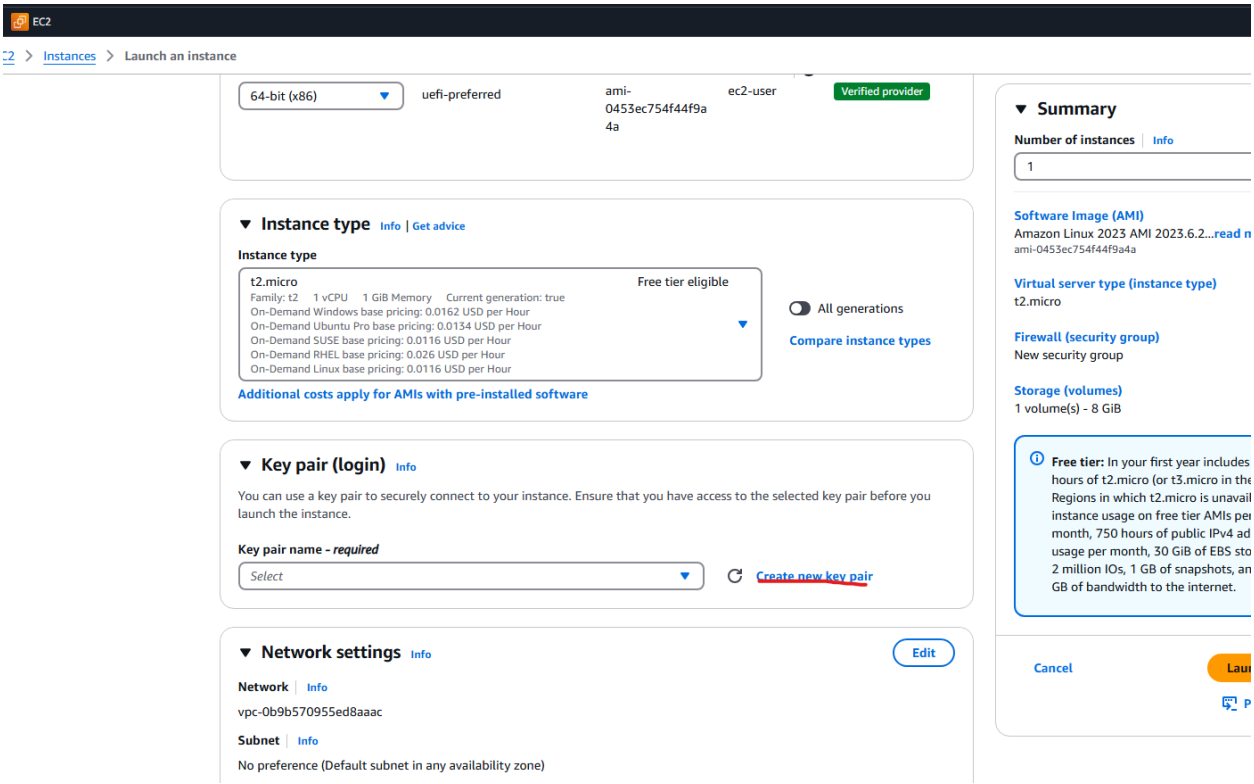
Fuente: elaboración propia.

En el menú de lanzar instancia ponemos un nombre a la instancia y seleccionamos la AMI (sistema operativo) y el tipo de instancia, en este caso dejamos las opciones que vienen por defecto.



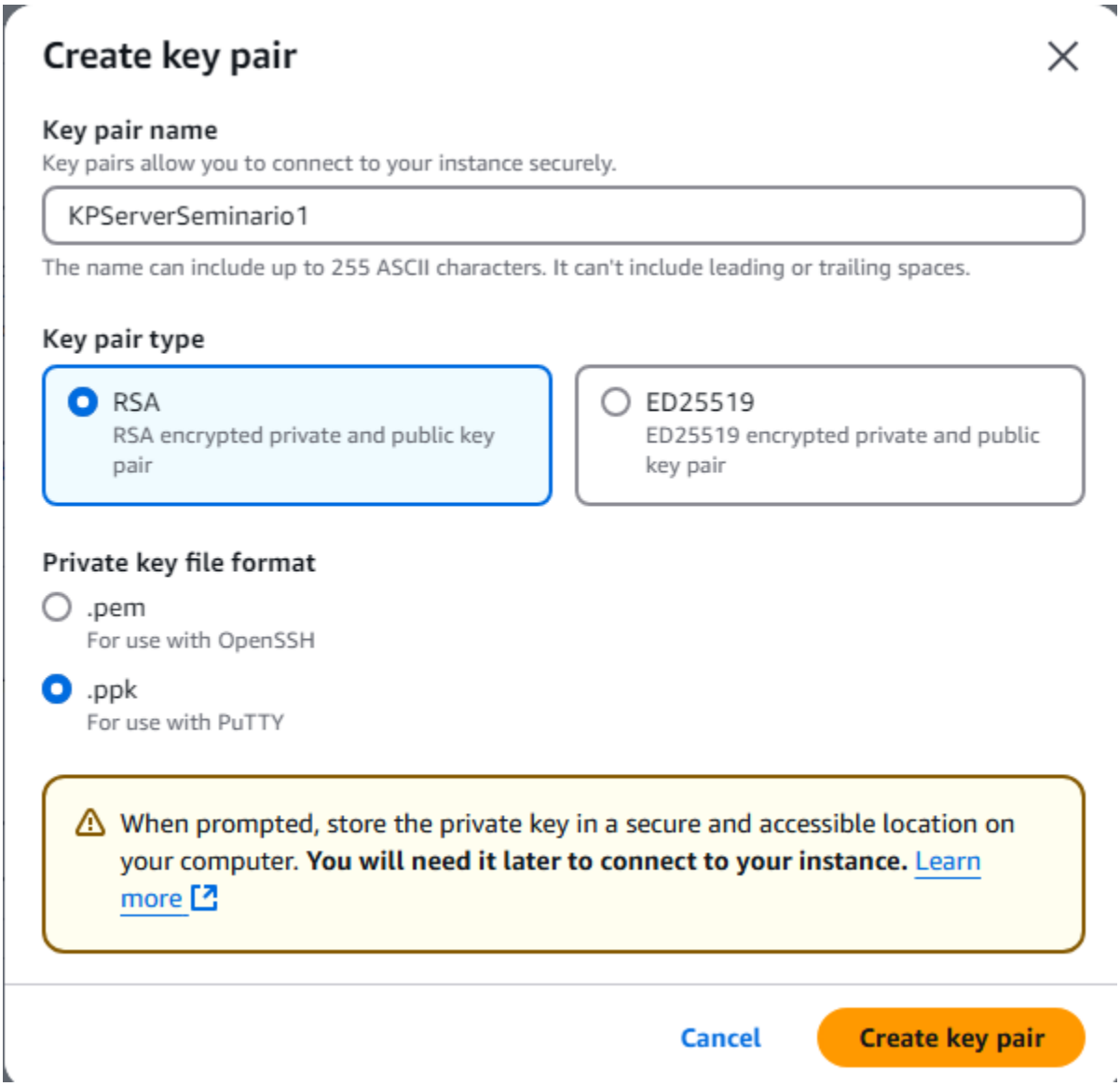
Fuente: elaboración propia.

En la opción de key pair creamos una nueva



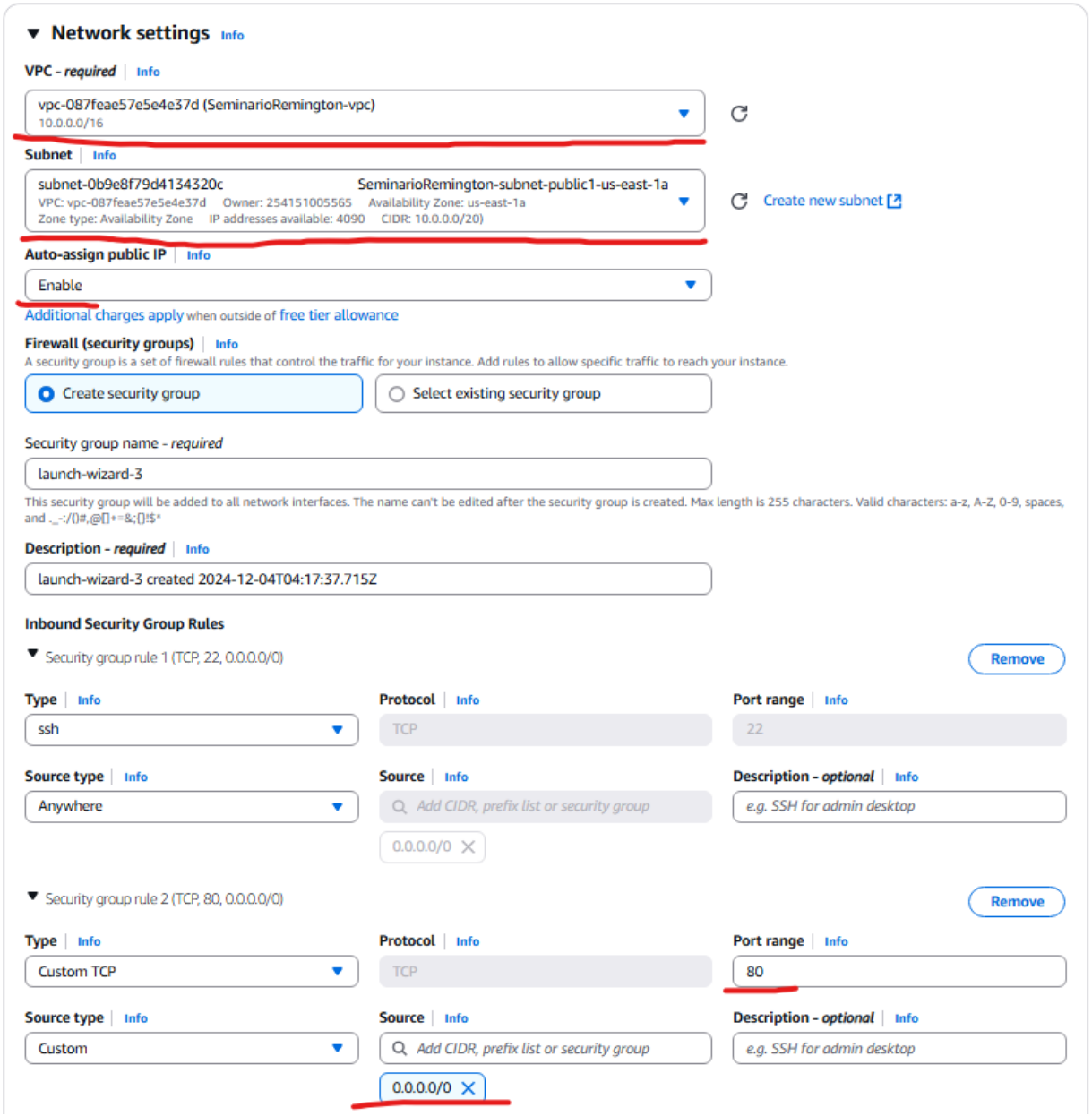
Fuente: elaboración propia.

Ponemos un nombre y seleccionamos el tipo de formato para la key pair, en este caso .ppk y damos clic en crear



Fuente: elaboración propia.

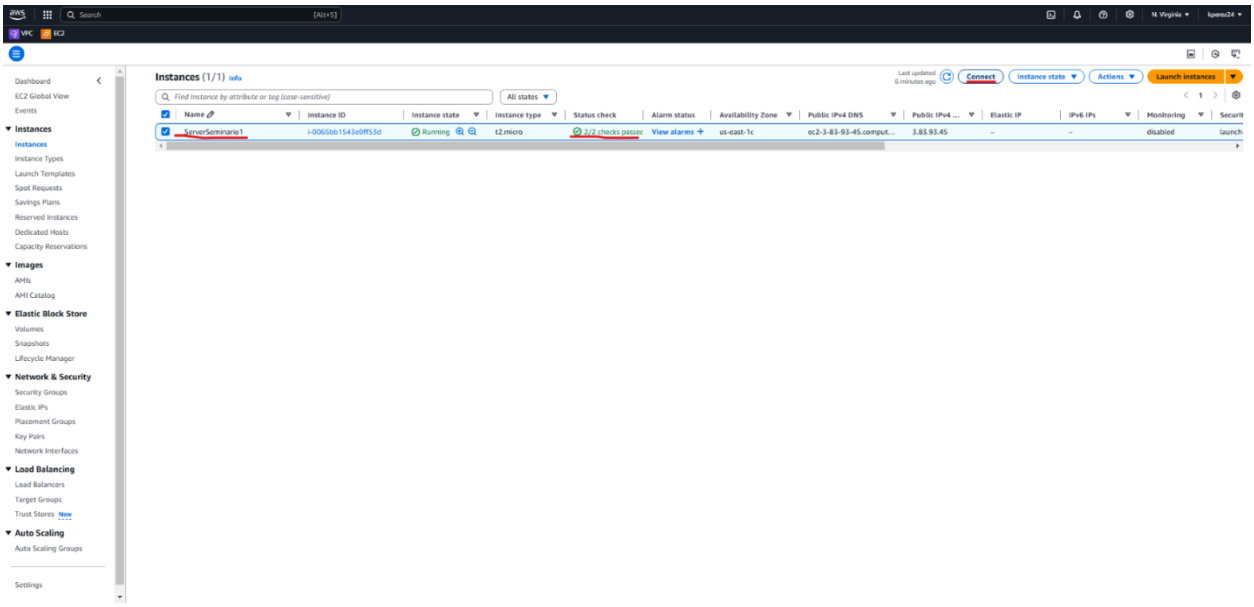
Luego de tener la key pair continuamos con el network setting, primero seleccionamos el VPC creado anteriormente, luego continuamos con la Subnet, en este caso seleccionamos la publica1 y por ultimo activamos la opción de auto asignar la IP publica y agregamos la regla de seguridad de grupo habilitando el puerto 80 para que este pueda ser accedido desde cualquier equipo con acceso a internet.



Fuente: elaboración propia.

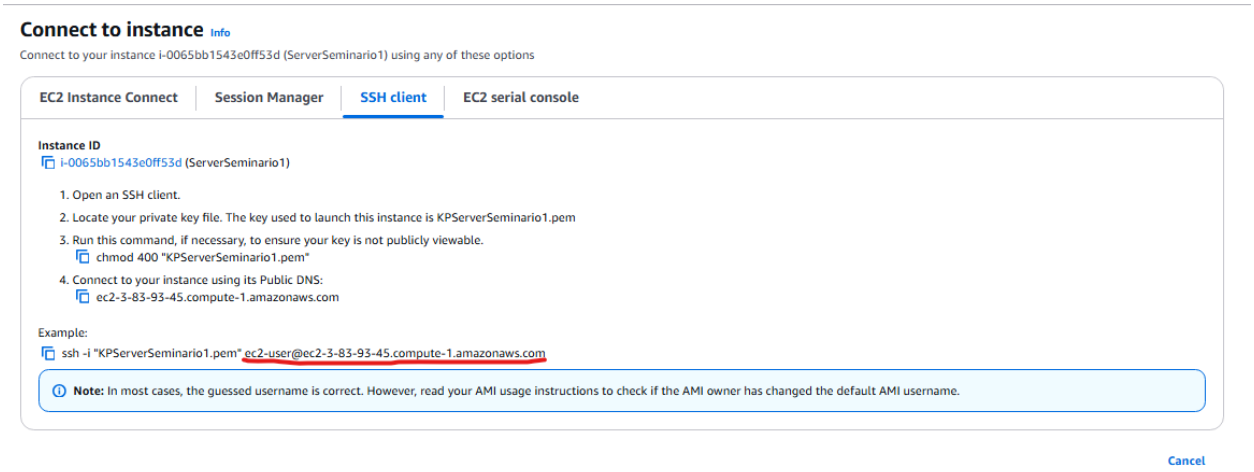
El apartado de configuración del almacenamiento (Configure Storage) se deja así por defecto. Y lanzamos la instancia.

Verificamos en instancias que este creada correctamente y con estado, la seleccionamos, y le damos en conectar



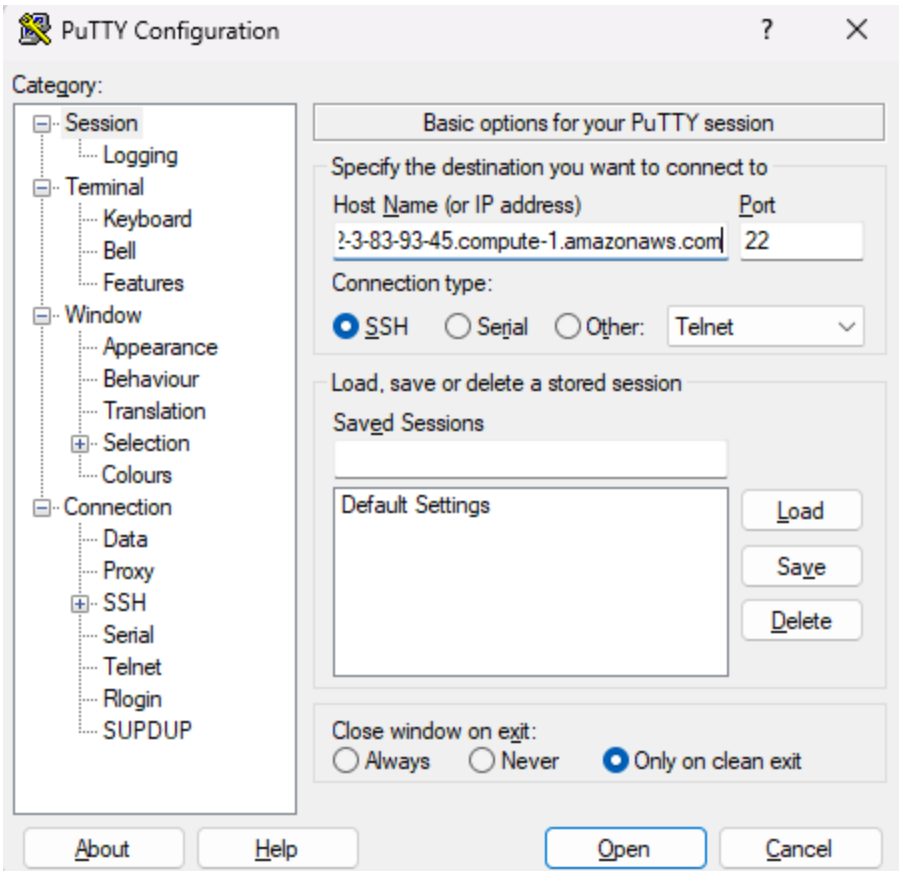
Fuente: elaboración propia.

Copiamos el nombre que nos genera amazon para conectarnos



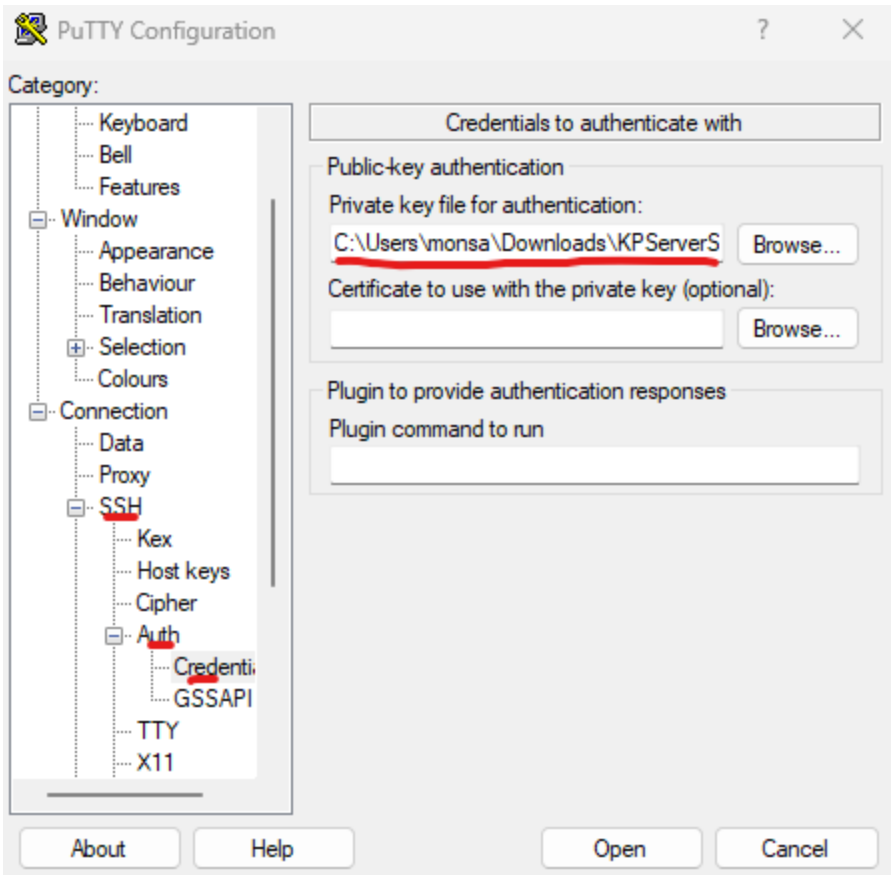
Fuente: elaboración propia.

Ponemos el nombre en la aplicacion PuTTY



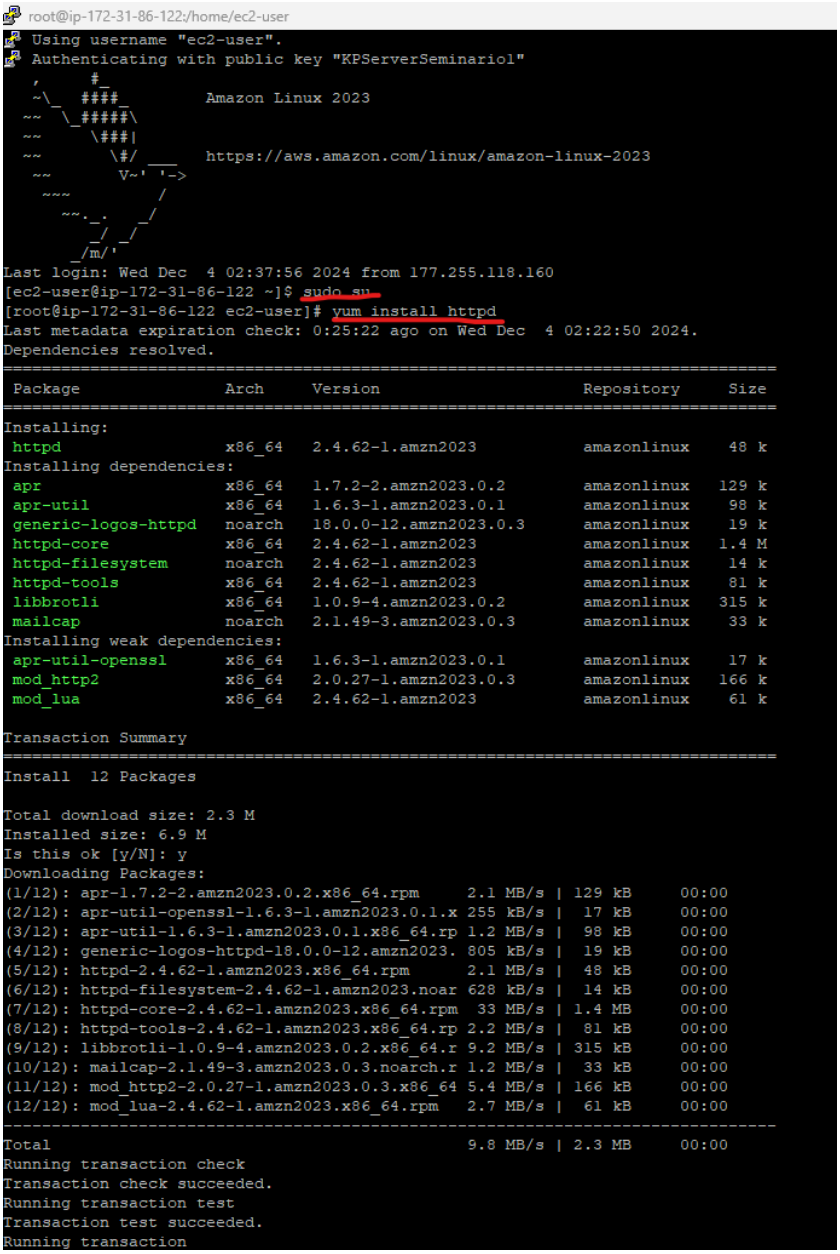
Fuente: elaboración propia.

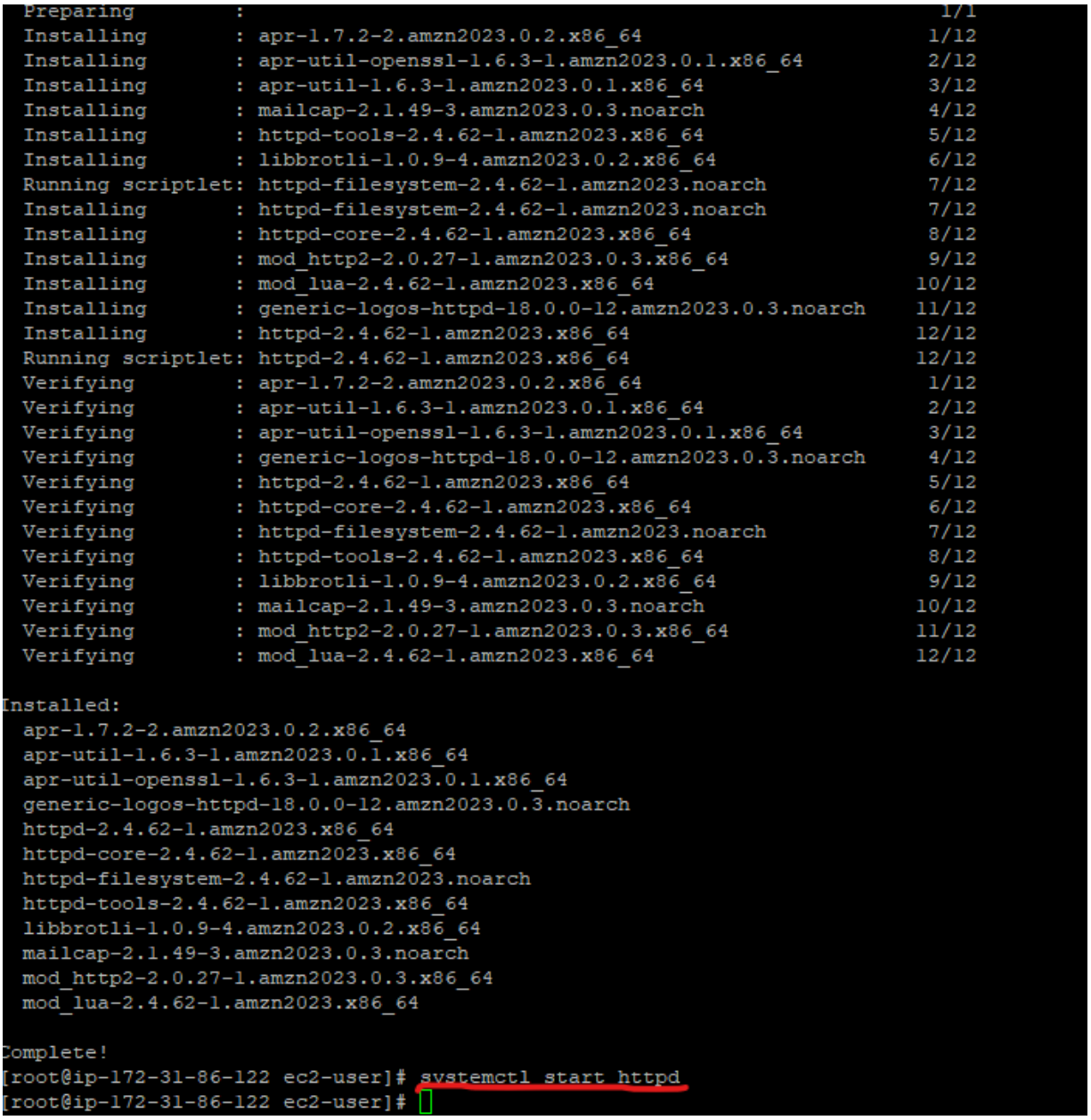
Luego damos clic en SSH -> Auth-> Credentials y subimos el archivo ppk que generamos para la key pair y damos clic en abrir



Fuente: elaboración propia.

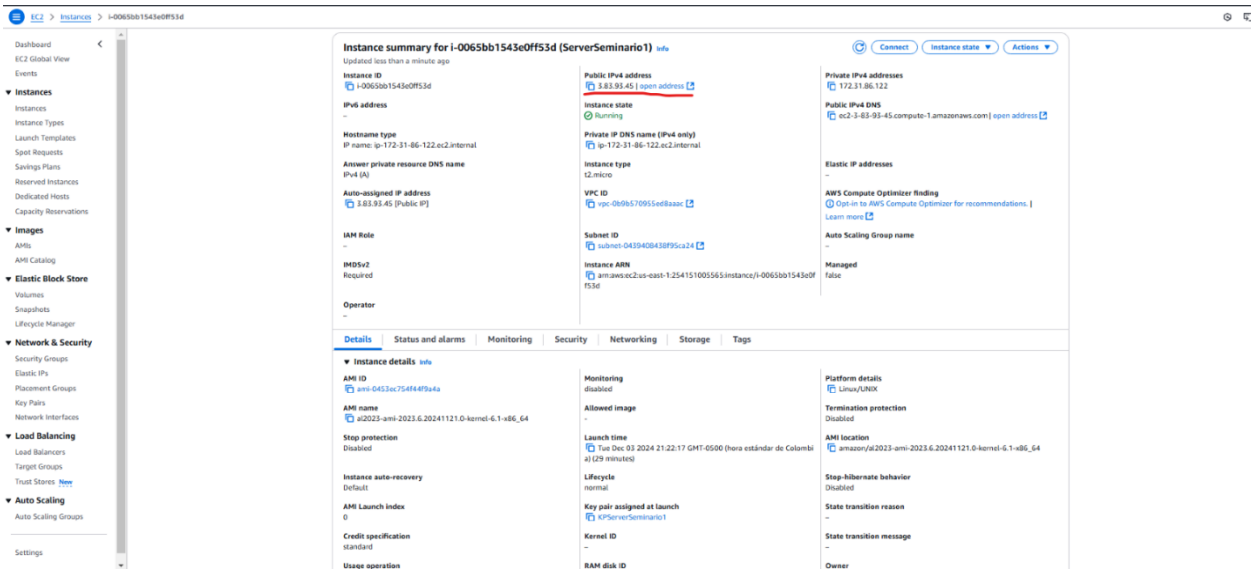
En la terminal accedemos como administradores (comando: sudo su) e instalamos apache (comando yum install httpd). Verificamos si el servicio está en funcionamiento (comando systemctl status httpd) si esta inactivo lo activamos (comando systemctl start httpd)


root@ip-172-31-86-122:/home/ec2-user
Using username "ec2-user".
Authenticating with public key "KPServerSeminario1"
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Wed Dec 4 02:37:56 2024 from 177.255.118.160
[ec2-user@ip-172-31-86-122 ~]$ sudo su
[root@ip-172-31-86-122 ec2-user]# yum install httpd
Last metadata expiration check: 0:25:22 ago on Wed Dec 4 02:22:50 2024.
Dependencies resolved.
Package Arch Version Repository Size
Installing:
httpd x86_64 2.4.62-1.amzn2023 amazonlinux 48 k
Installing dependencies:
apr x86_64 1.7.2-2.amzn2023.0.2 amazonlinux 129 k
apr-util x86_64 1.6.3-1.amzn2023.0.1 amazonlinux 98 k
generic-logos-httpd noarch 18.0.0-12.amzn2023.0.3 amazonlinux 19 k
httpd-core x86_64 2.4.62-1.amzn2023 amazonlinux 1.4 M
httpd-filesystem noarch 2.4.62-1.amzn2023 amazonlinux 14 k
httpd-tools x86_64 2.4.62-1.amzn2023 amazonlinux 81 k
libbrotli x86_64 1.0.9-4.amzn2023.0.2 amazonlinux 315 k
mailcap noarch 2.1.49-3.amzn2023.0.3 amazonlinux 33 k
Installing weak dependencies:
apr-util-openssl x86_64 1.6.3-1.amzn2023.0.1 amazonlinux 17 k
mod_http2 x86_64 2.0.27-1.amzn2023.0.3 amazonlinux 166 k
mod_lua x86_64 2.4.62-1.amzn2023 amazonlinux 61 k
Transaction Summary
Install 12 Packages
Total download size: 2.3 M
Installed size: 6.9 M
Is this ok [y/N]: y
Downloading Packages:
(1/12): apr-1.7.2-2.amzn2023.0.2.x86_64.rpm 2.1 MB/s | 129 kB 00:00
(2/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x 255 kB/s | 17 kB 00:00
(3/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rp 1.2 MB/s | 98 kB 00:00
(4/12): generic-logos-httpd-18.0.0-12.amzn2023. 805 kB/s | 19 kB 00:00
(5/12): httpd-2.4.62-1.amzn2023.x86_64.rpm 2.1 MB/s | 48 kB 00:00
(6/12): httpd-filesystem-2.4.62-1.amzn2023.noar 628 kB/s | 14 kB 00:00
(7/12): httpd-core-2.4.62-1.amzn2023.x86_64.rpm 33 MB/s | 1.4 MB 00:00
(8/12): httpd-tools-2.4.62-1.amzn2023.x86_64.rp 2.2 MB/s | 81 kB 00:00
(9/12): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.r 9.2 MB/s | 315 kB 00:00
(10/12): mailcap-2.1.49-3.amzn2023.0.3.noarch.r 1.2 MB/s | 33 kB 00:00
(11/12): mod_http2-2.0.27-1.amzn2023.0.3.x86_64 5.4 MB/s | 166 kB 00:00
(12/12): mod_lua-2.4.62-1.amzn2023.x86_64.rpm 2.7 MB/s | 61 kB 00:00
Total 9.8 MB/s | 2.3 MB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction



Fuente: elaboración propia.

Luego de realizar estos pasos regresamos a la instancia y copiamos la IP publica



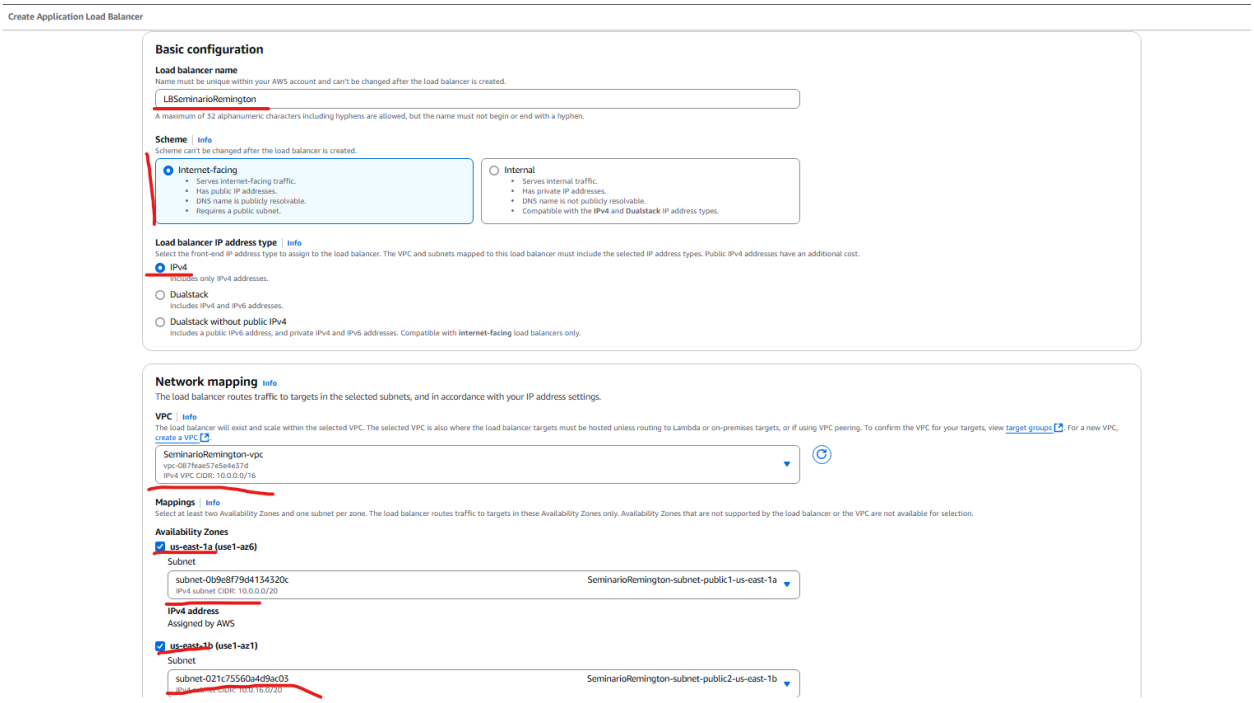
Fuente: elaboración propia.

Esta la copiamos en el navegador web y la ejecutamos, asi verificamos su correcto funcionamiento.

Balanceador de carga (Load balancing)

Para crear un balanceador de cargas inicialmente ponemos un nombre, especificamos que el tipo de ip a asignar es la IPv4 y su esquema Internet-facing.

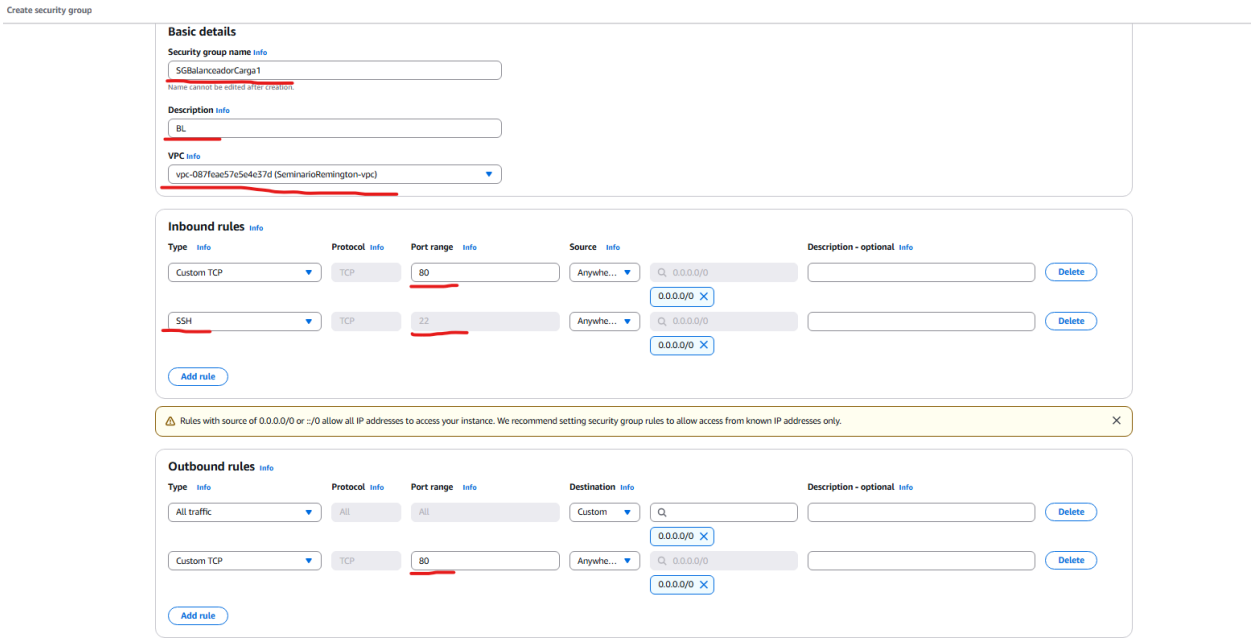
Continuamos en la sección de Network mapping asignando la VPC creada anteriormente, luego en las zonas disponibles seleccionamos las subnet publicas.



Fuente: elaboración propia.

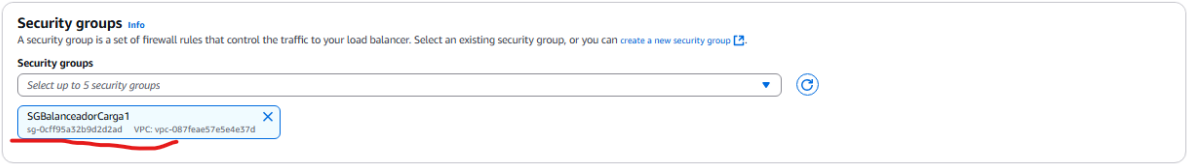
En la seccion de grupo de seguridad ingresamos los datos basicos, nombre, descripcion y seleccionamos el VPC a asociar este grupo.

Continuamos las reglas de entrada y salida, registrando los puertos 80 en ambos y 22 en la regla de entrada.



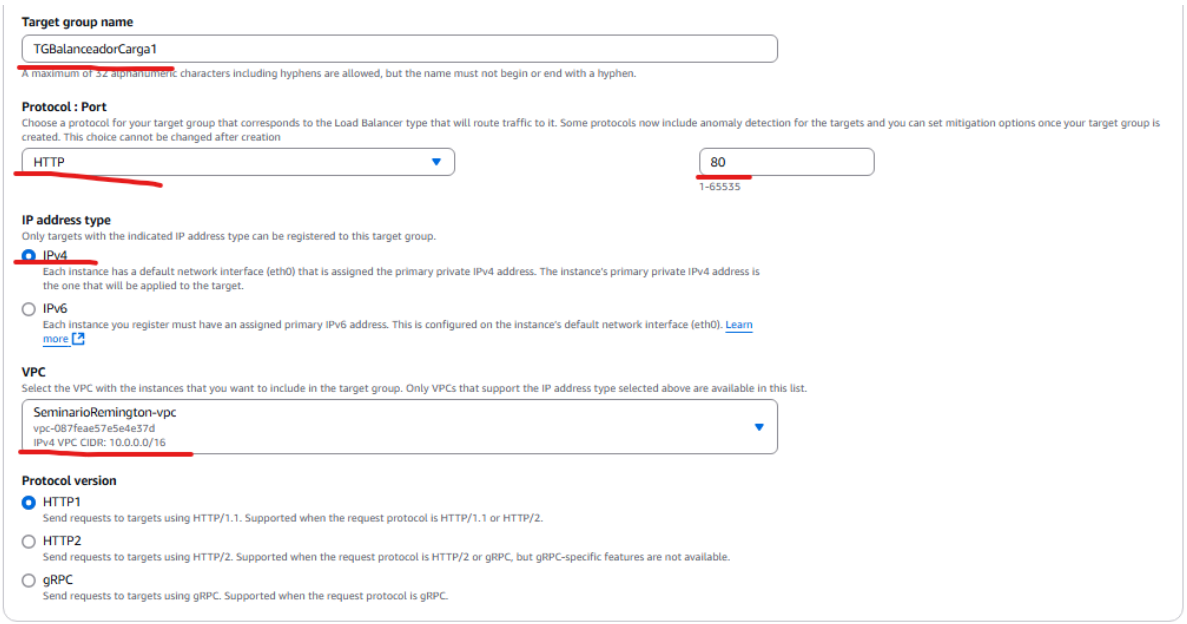
Fuente: elaboración propia.

Despues de crear el grupo de seguridad agregamos este al balanceador de carga que estamos creando.



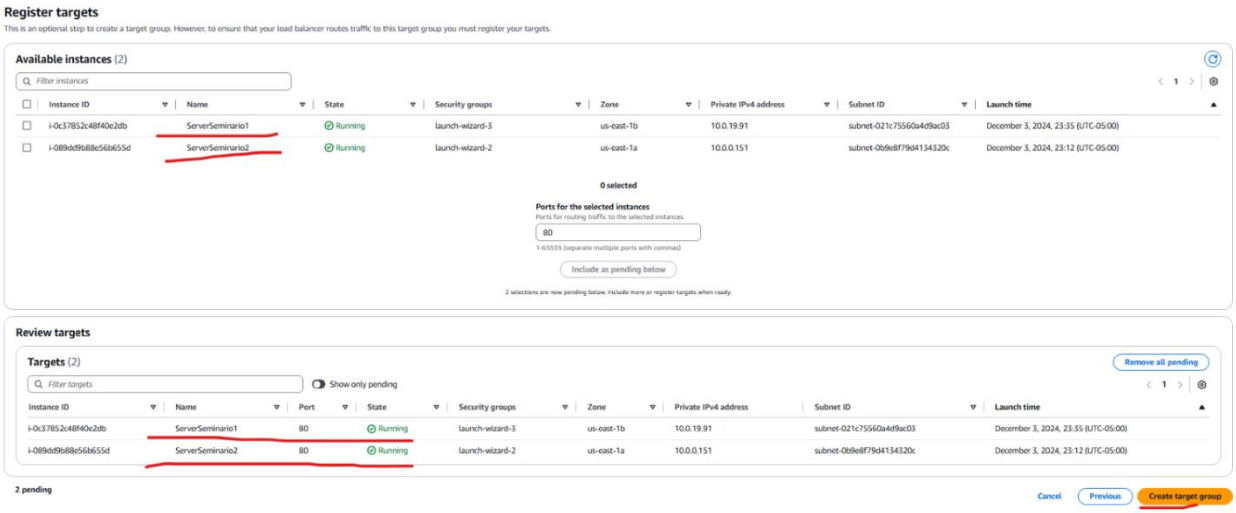
Fuente: elaboración propia.

Continuamos creando el *Target Group*, asignamos el nombre, el protocolo HTTP y puerto 80, especificamos el tipo de IPv4 y por ultimo que el VPC sea el que creamos inicialmente.



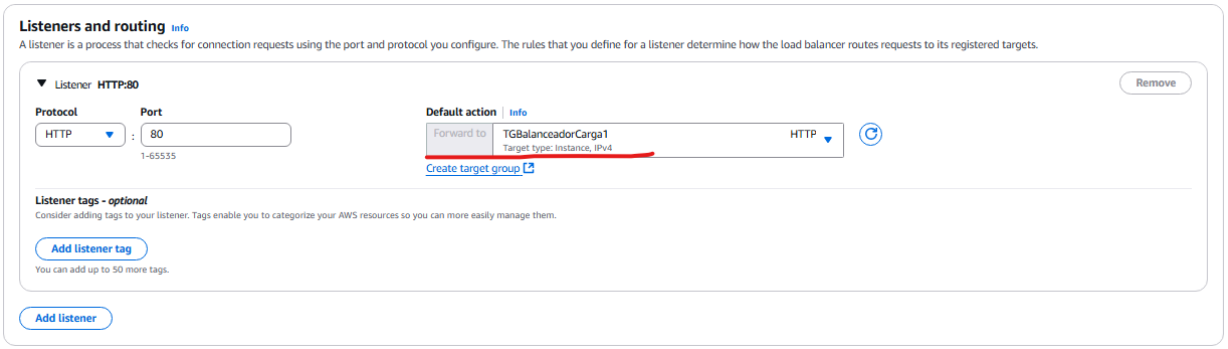
Fuente: elaboración propia.

Continuamos con el registro de las instancias, seleccionamos las 2 que creamos inicialmente



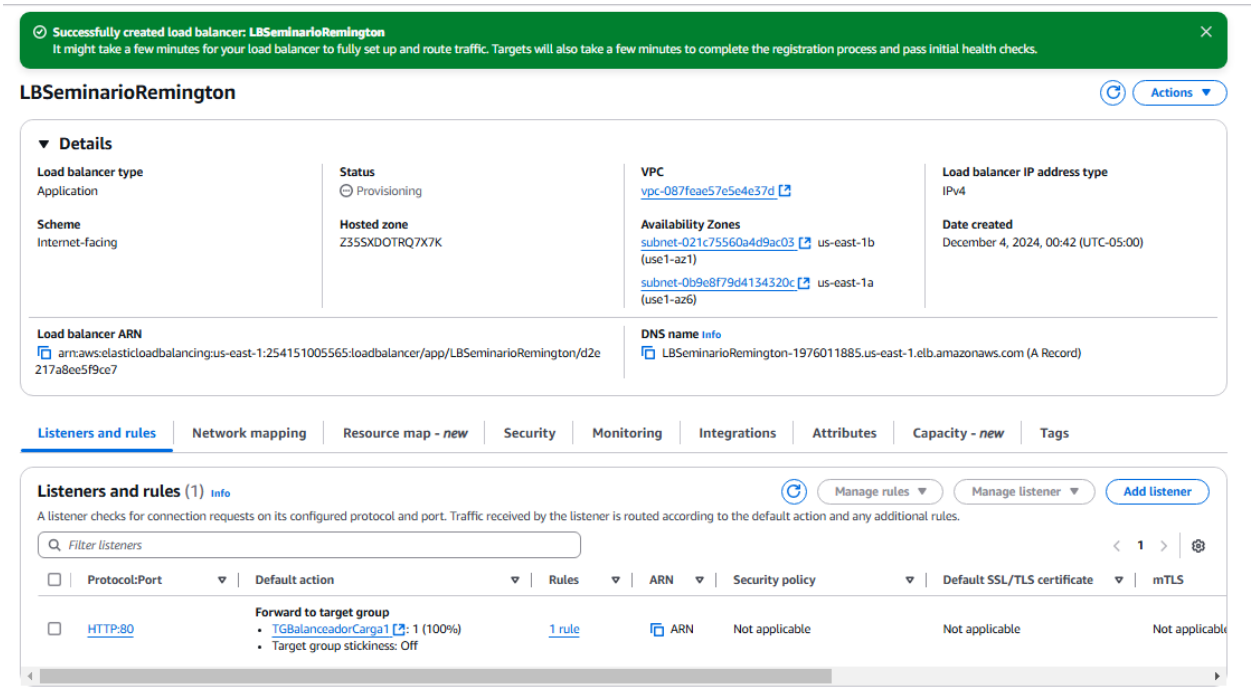
Fuente: elaboración propia.

Agregamos el Target Group creado a la configuración del balanceador de cargas.



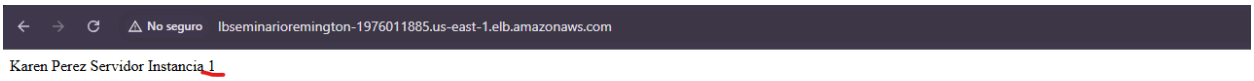
Fuente: elaboración propia.

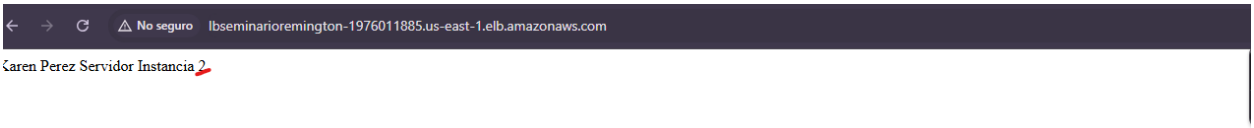
Por último verificamos el resumen de la creación del balanceador de cargas y en caso tal de estar todo correcto, lo creamos.



Fuente: elaboración propia.

Copiamos el DNS Name y ejecutamos en el navegador para verificar que si este funcionando el balanceador de carga

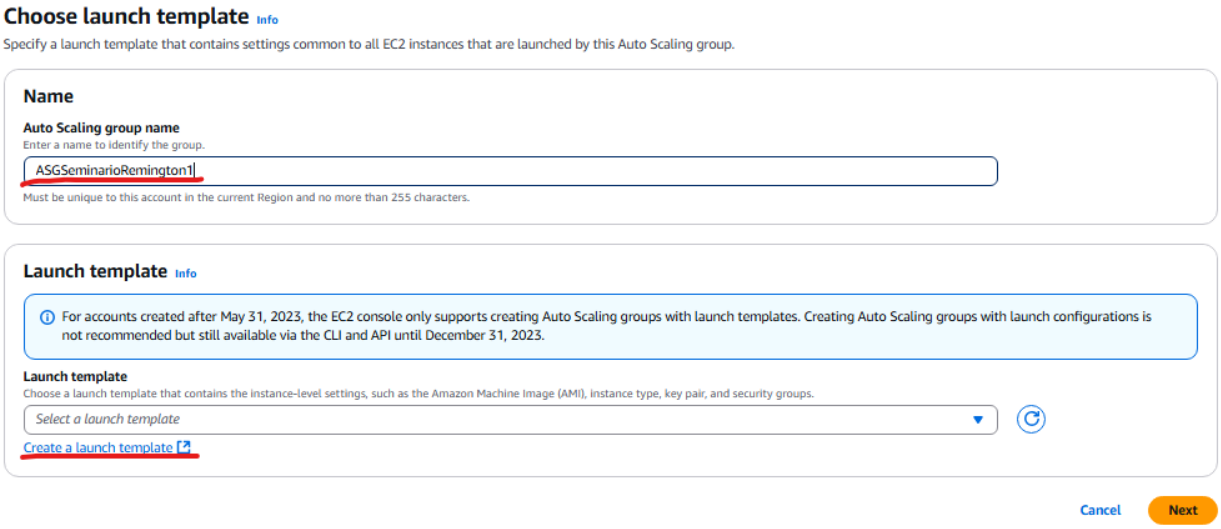




Fuente: elaboración propia.

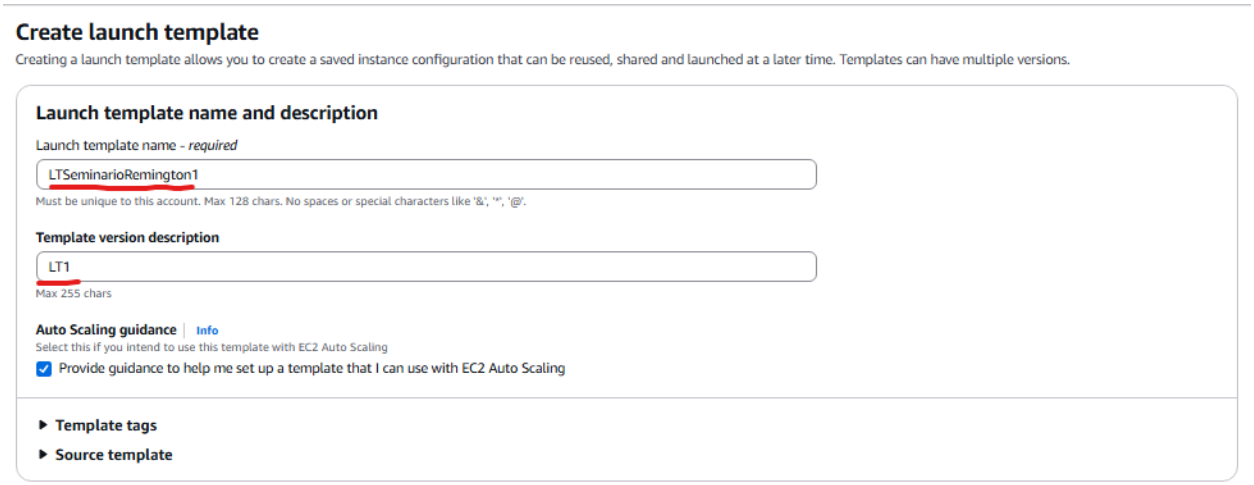
En estas imágenes podemos ver como intercala las instancias que creamos anteriormente, por lo cual determinamos que quedo creado correctamente.

Procedemos a crear un *Auto Scaling Group*, ingresamos el nombre del grupo de auto scaling, y luego seleccionamos el Launch Template, en caso tal que no esté creado procedemos a darle en “Create a launch template”.



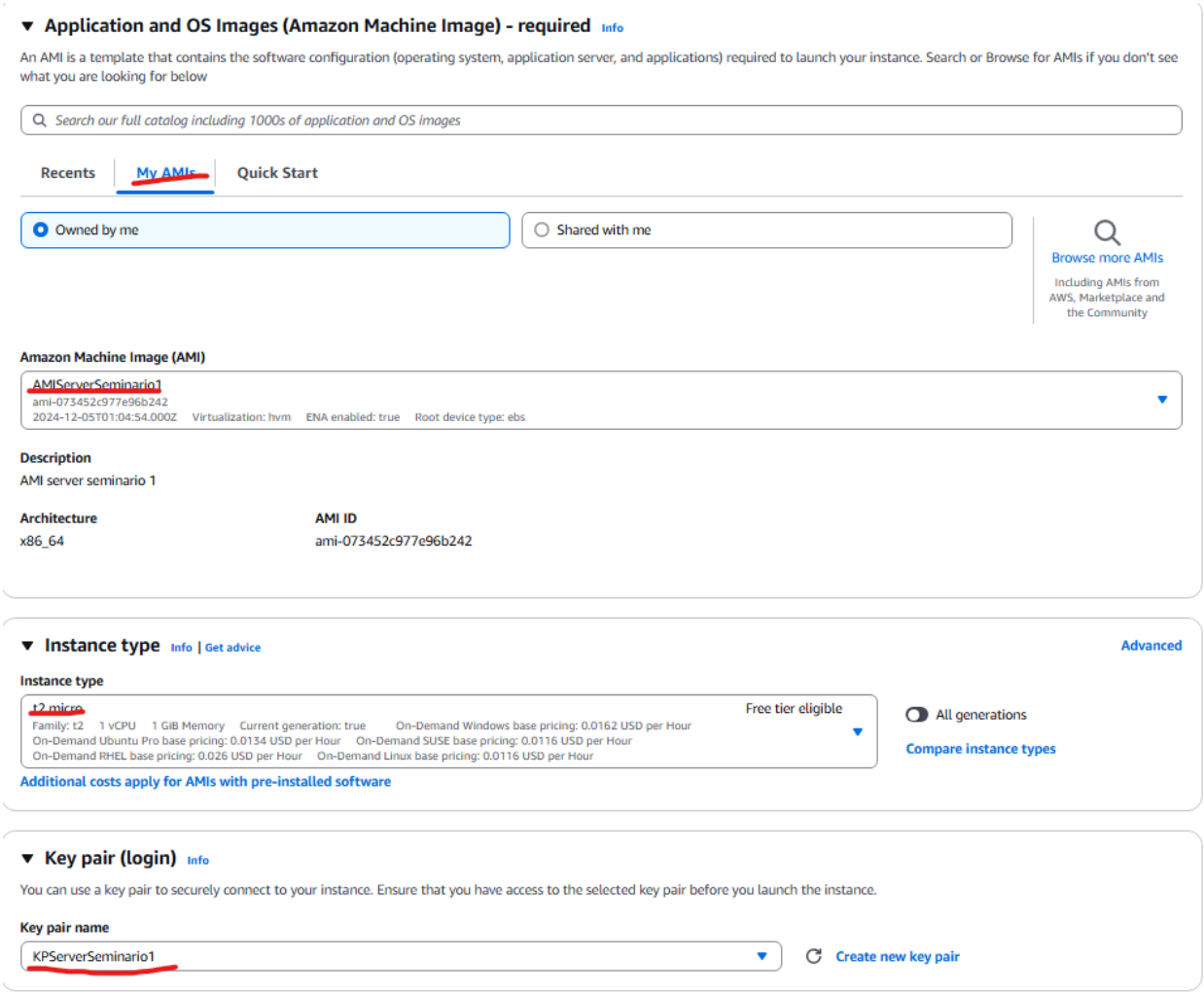
Fuente: elaboración propia.

Al momento de crear el Launch Template ingresamos el nombre y la descripción para poder identificar esta plantilla



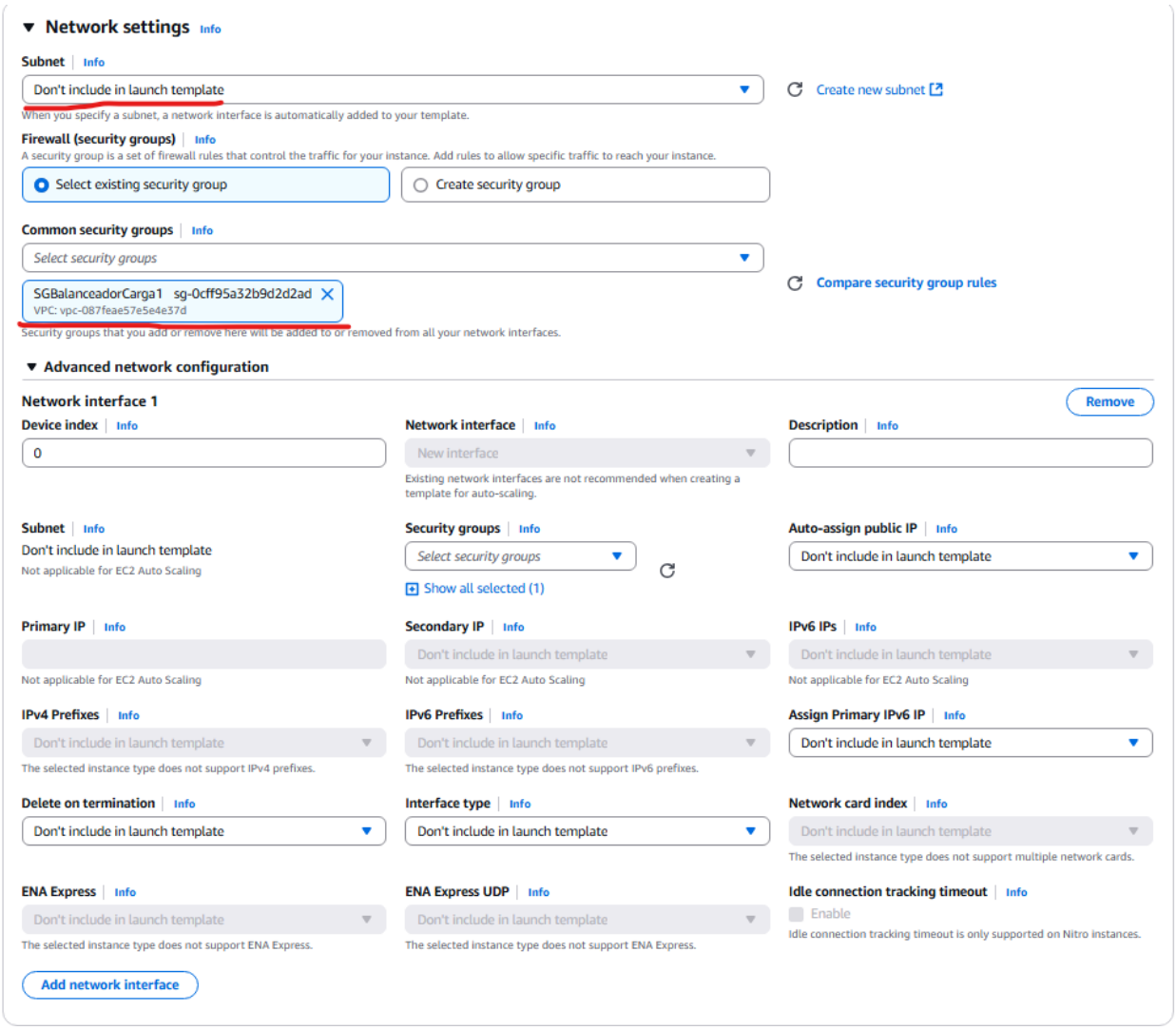
Fuente: elaboración propia.

En la sección de aplicaciones e imágenes de sistemas operativo que ya tenemos creado en mis AMIs. Luego seleccionamos la instancia gratuita en este caso t2.micro y el Key Pair usamos el “KPServerSeminario1”.



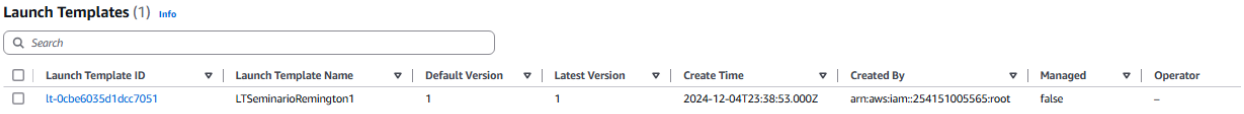
Fuente: elaboración propia.

Por ultimo en Network Setting no incluimos una subnet y luego seleccionamos un grupo existente de seguridad “SGBalanceadorCarga1” y finalmente creamos el Launch Template.



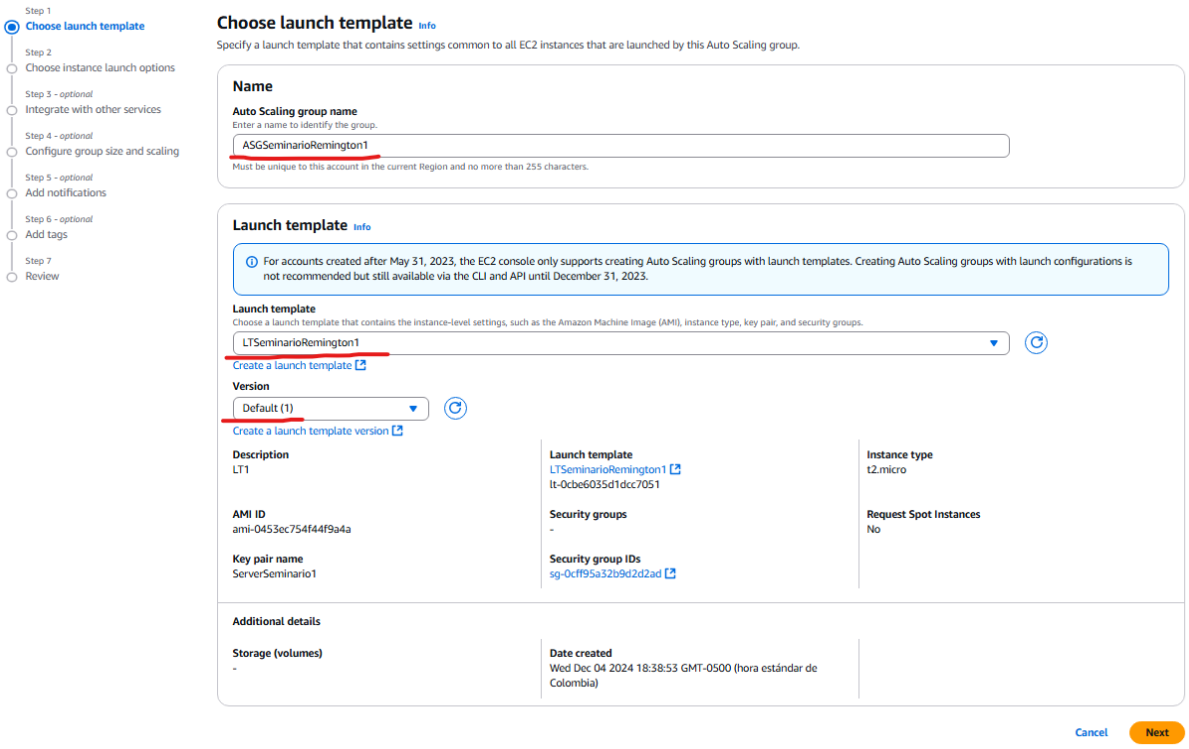
Fuente: elaboración propia.

Aqui verificamos que se haya creado correctamente.



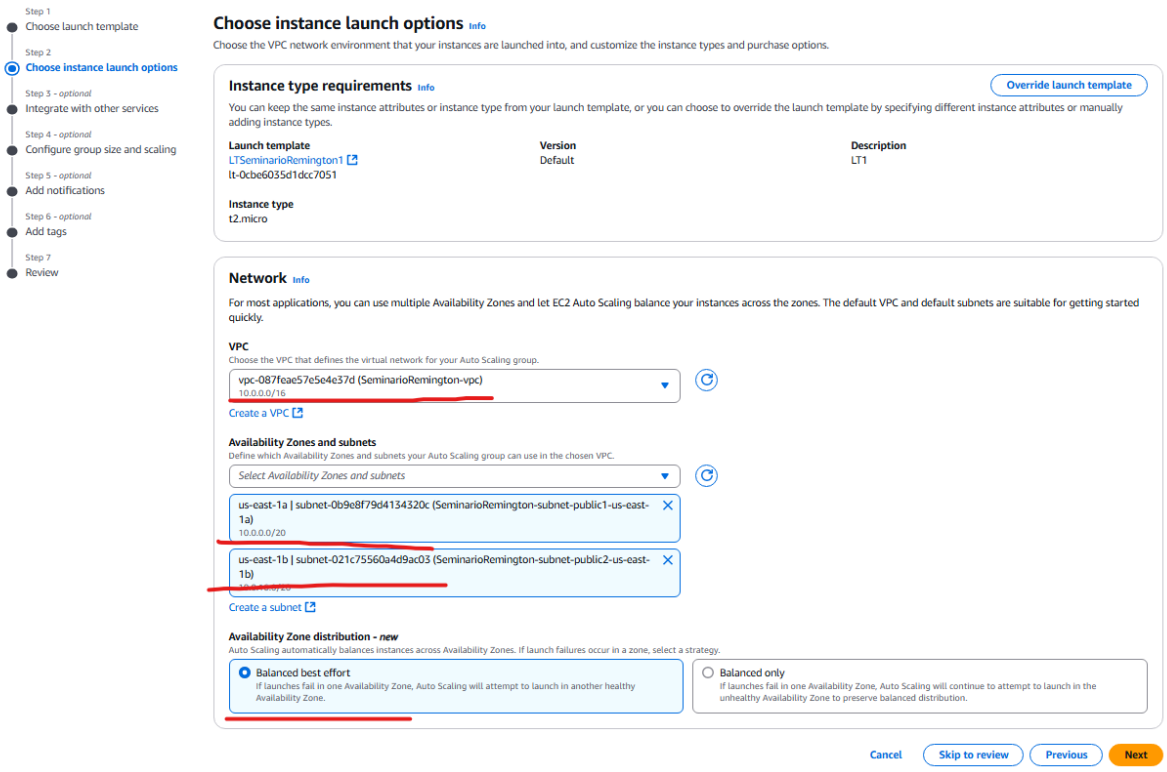
Fuente: elaboración propia.

Continuamos con la seleccion del launch template recien creado y luego asignamos la versión default.



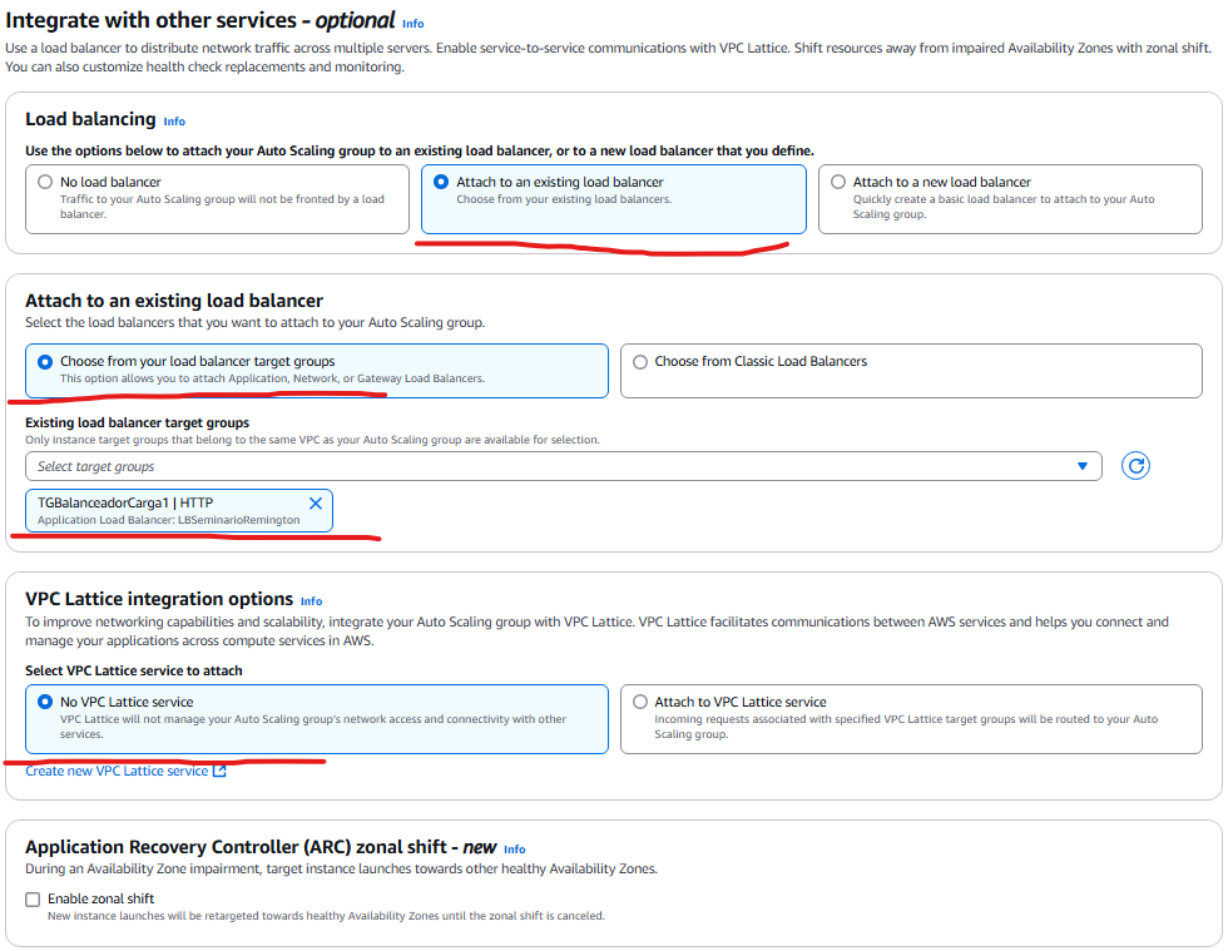
Fuente: elaboración propia.

En el paso dos seleccionamos la VPC y luego las zonas y subnets publicas que tengamos creadas.



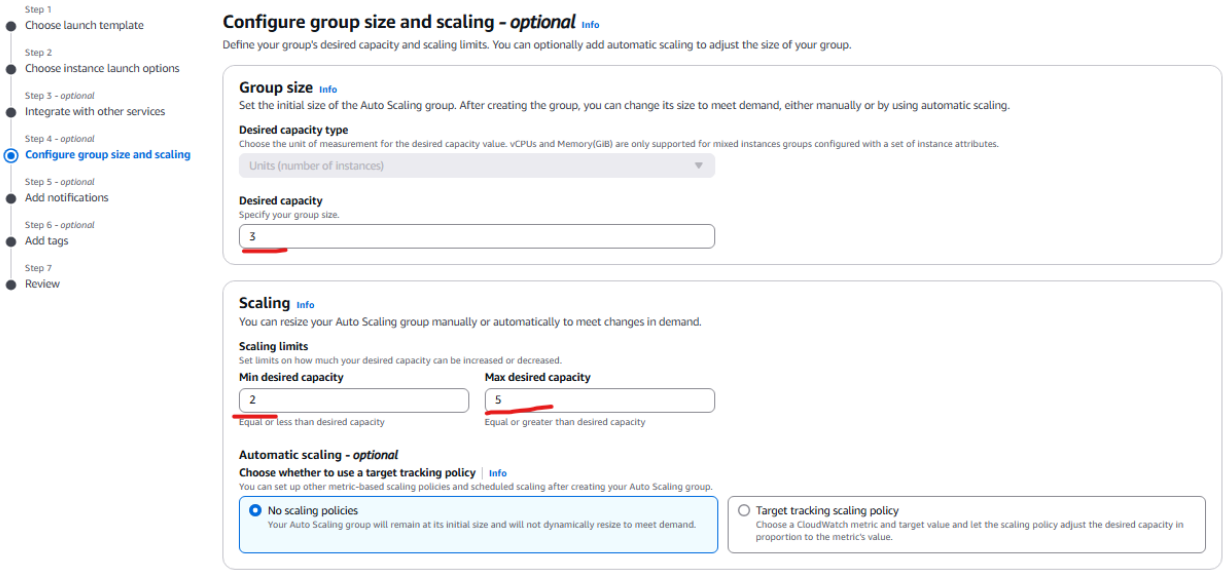
Fuente: elaboración propia.

Continuamos con el paso 3 integrando otros servicios, seleccionamos el balanceador de carga creado previamente, los demás campos los dejamos por default como vienen preseleccionados.



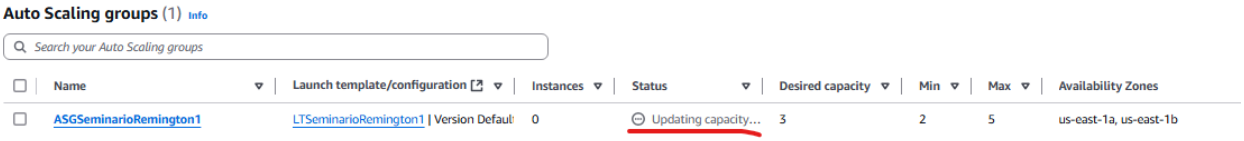
Fuente: elaboración propia.

Continuamos con el paso 4 configurando el tamaño del grupo de Scaling. Asignamos máximos y mínimos en los campos de capacidad.



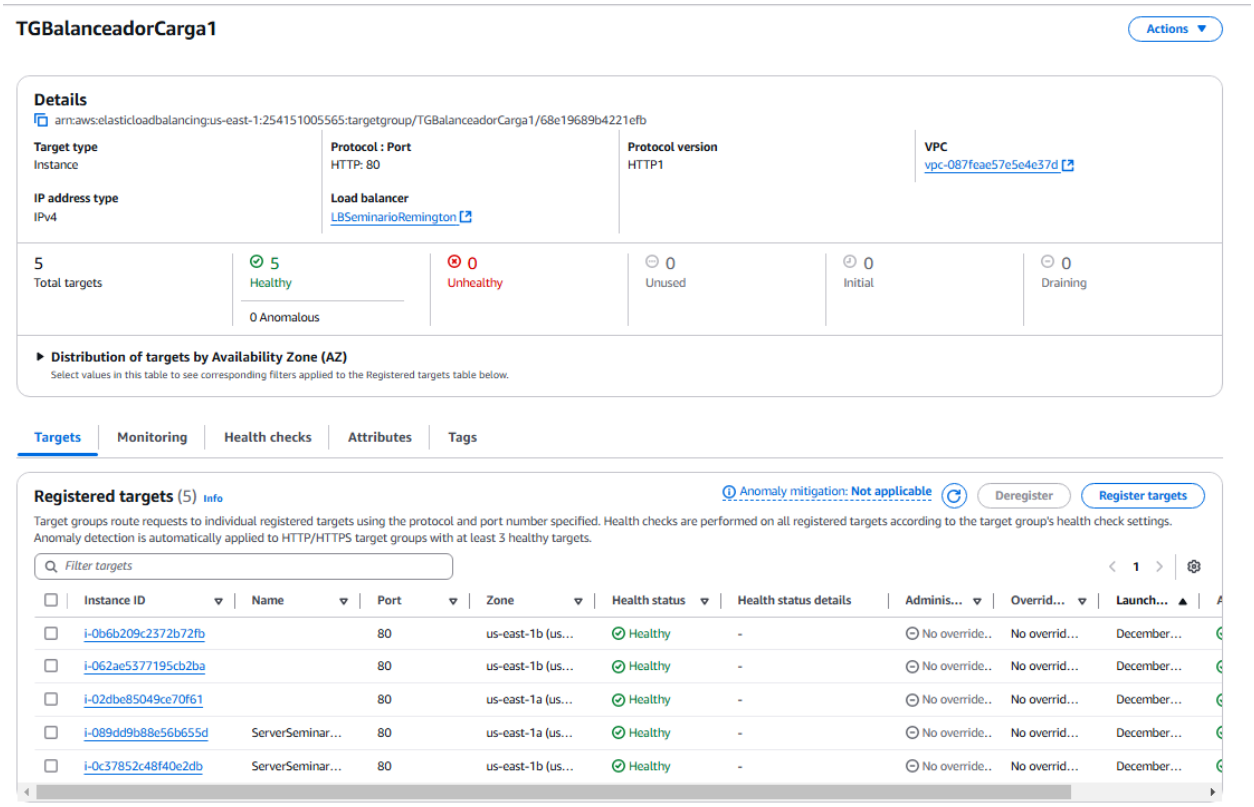
Fuente: elaboración propia.

Finalmente continuamos los demás pasos dejando sus campos seleccionados por default y esperamos a que se actualice la capacidad del Auto Scaling Group.



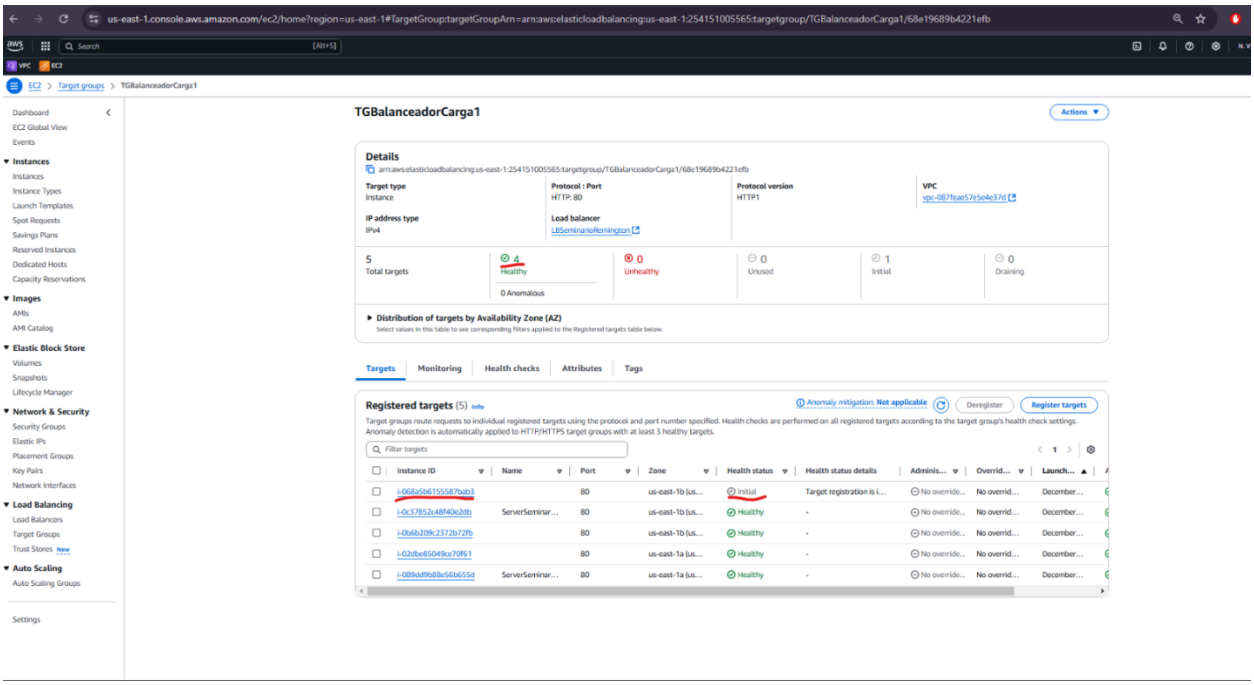
Fuente: elaboración propia.

Luego de esto nuestro Target Group debe visualizarse así



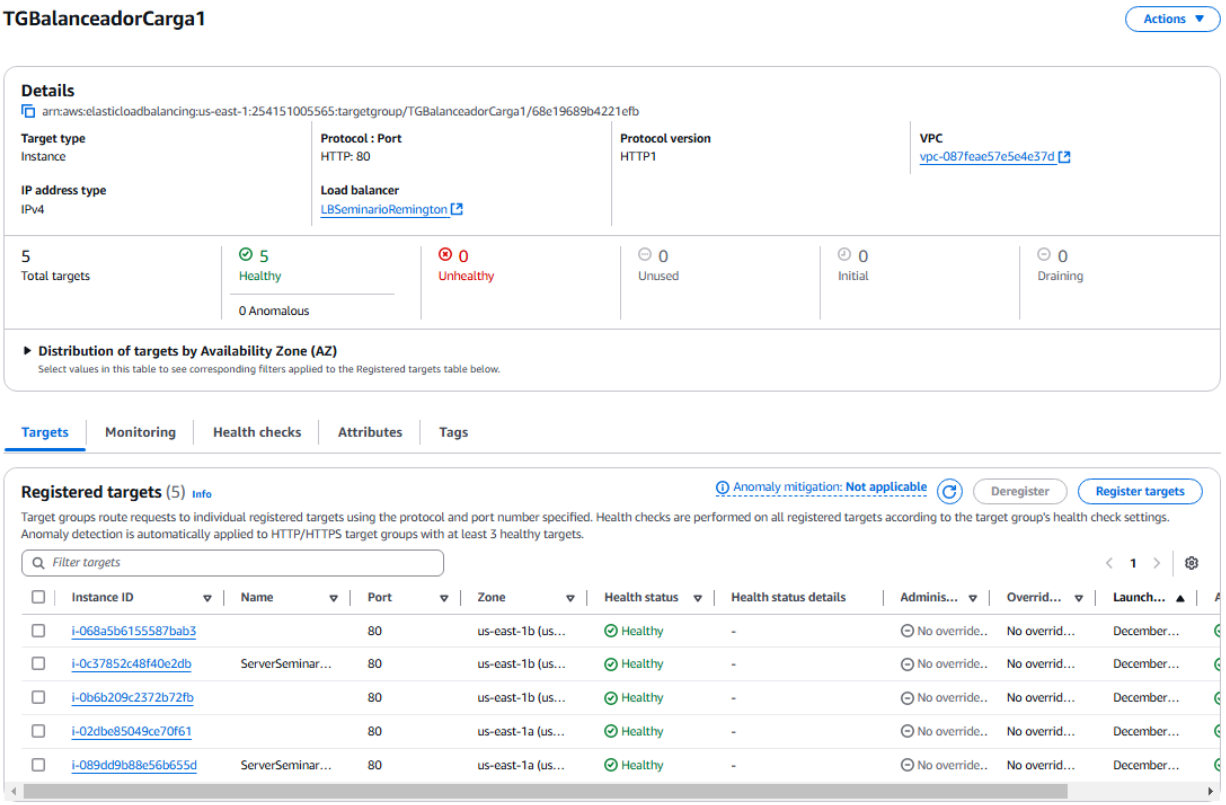
Fuente: elaboración propia.

Ahora verificamos si le damos stop a una instancia el funcionamiento del autoscaling, podemos ver que ahora nos muestra 4 healthy



Fuente: elaboración propia.

En respuesta a esto vemos que vuelve a iniciar la instancia y la marca como healthy nuevamente.

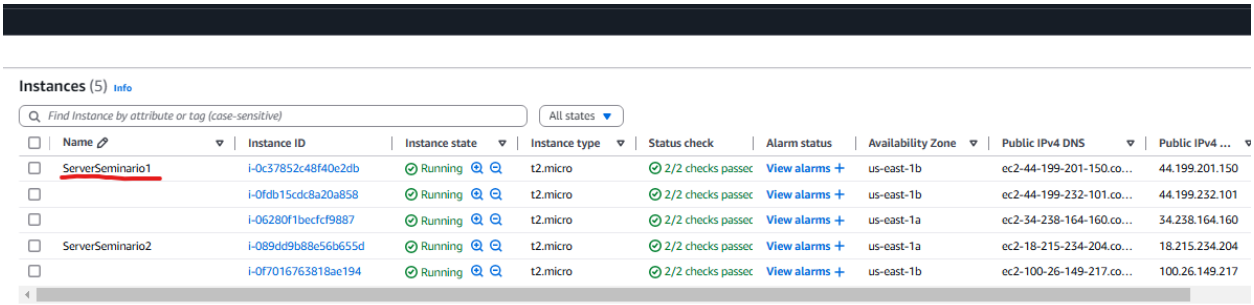


Implementacion de docker para uso de contenedores y servidor web (nginx)

Docker

En las instancias que ya tenemos creadas vamos a realizar la implementación de Docker para hacer uso de microservicios.

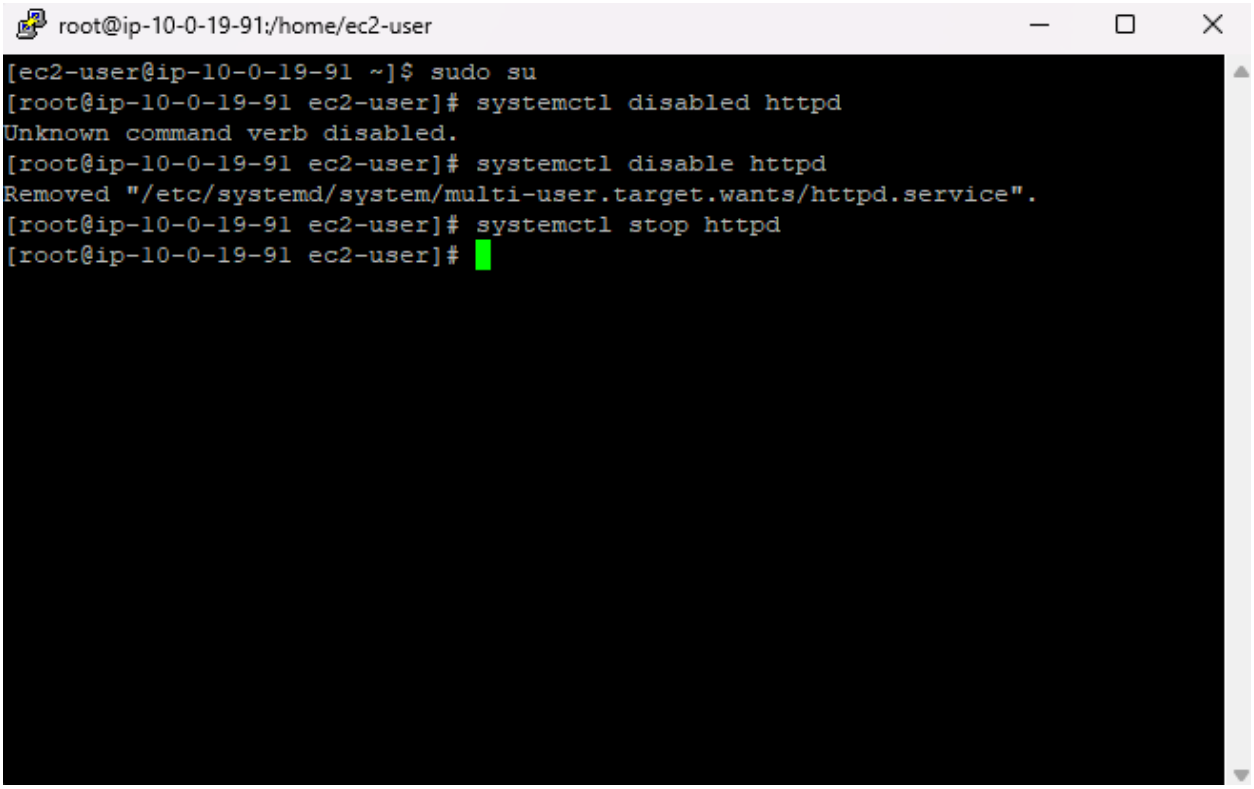
Usaremos usar la instancia llamada Server Seminario 1



Fuente: Elaboración propia

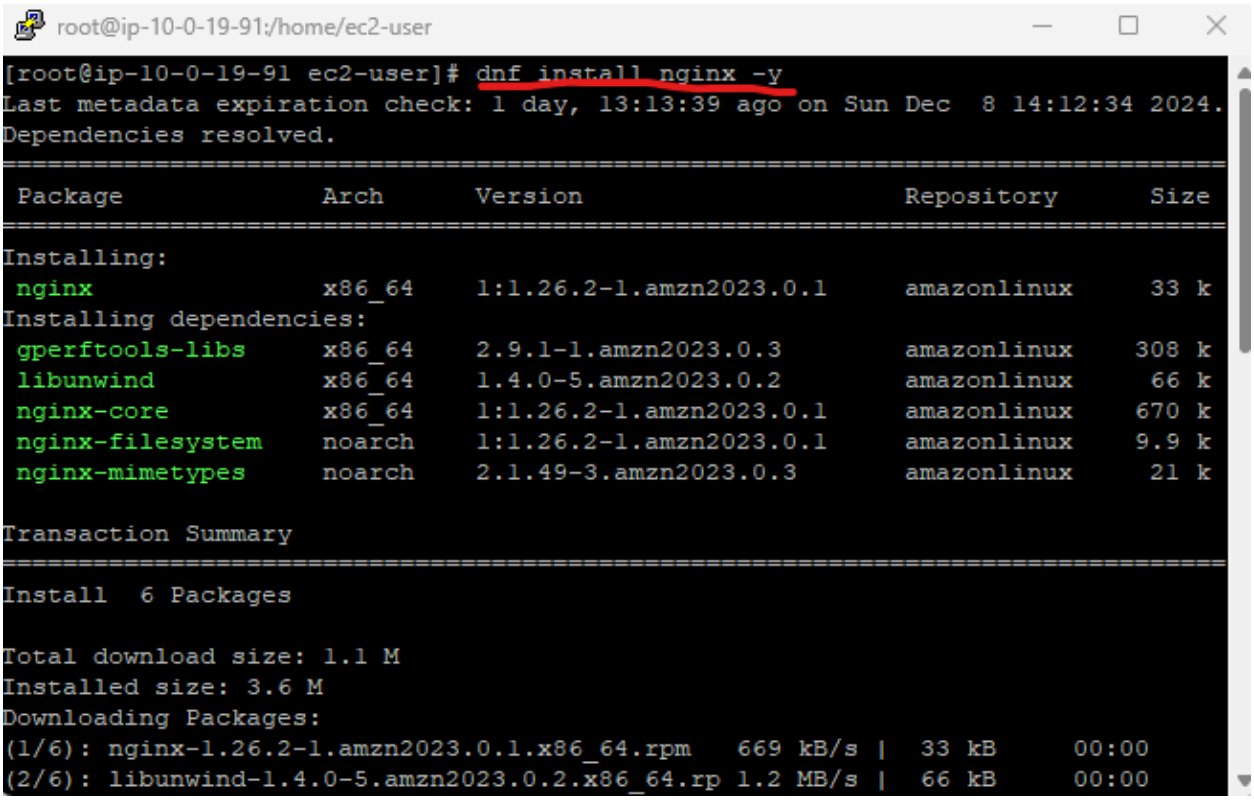
Nos conectamos a la instancia y comenzamos con la instalación de nginx.

También pausamos el servicio de Apache pues este caso utilizaremos otro servidor web, lo realizamos así



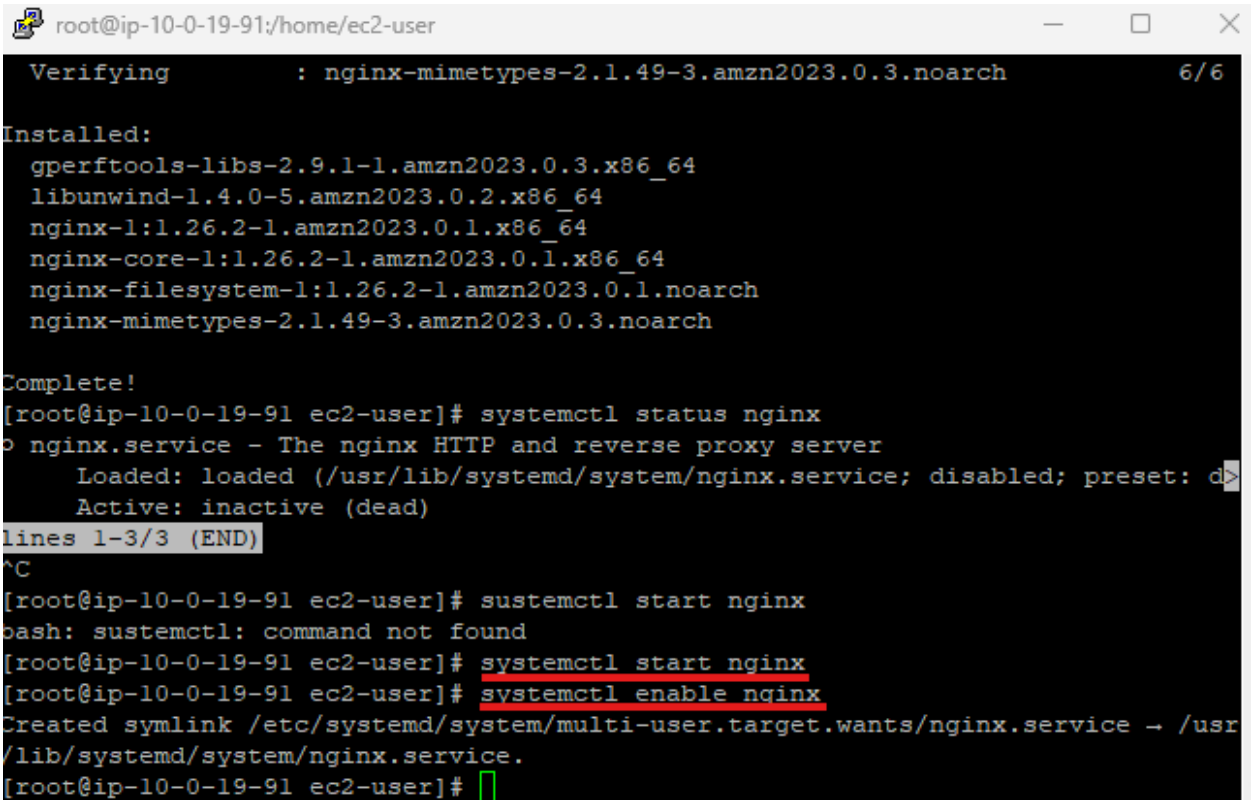
Fuente: elaboración propia.

Instalación de nginx



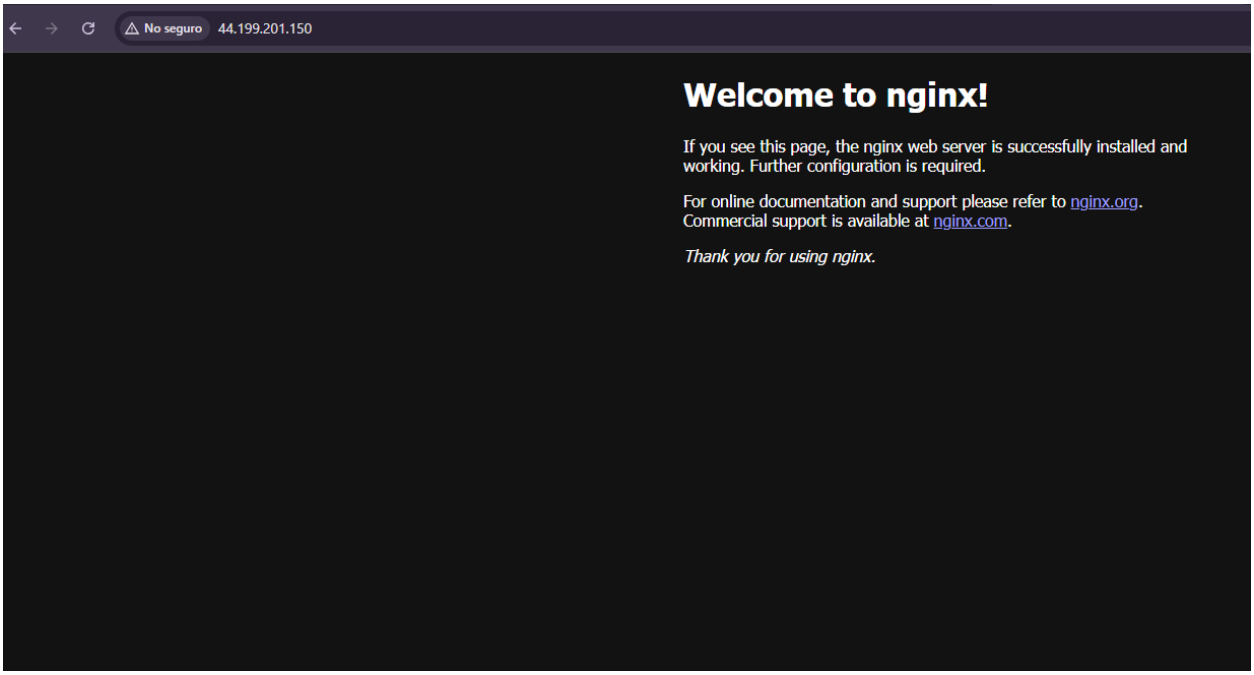
Fuente: elaboración propia.

Los comandos para administrar son los mismo que los de Apache, solo cambia el final de comando, ejecutamos el comando de iniciar (start) y el comando de iniciar siempre que arranque el servicio (enable)

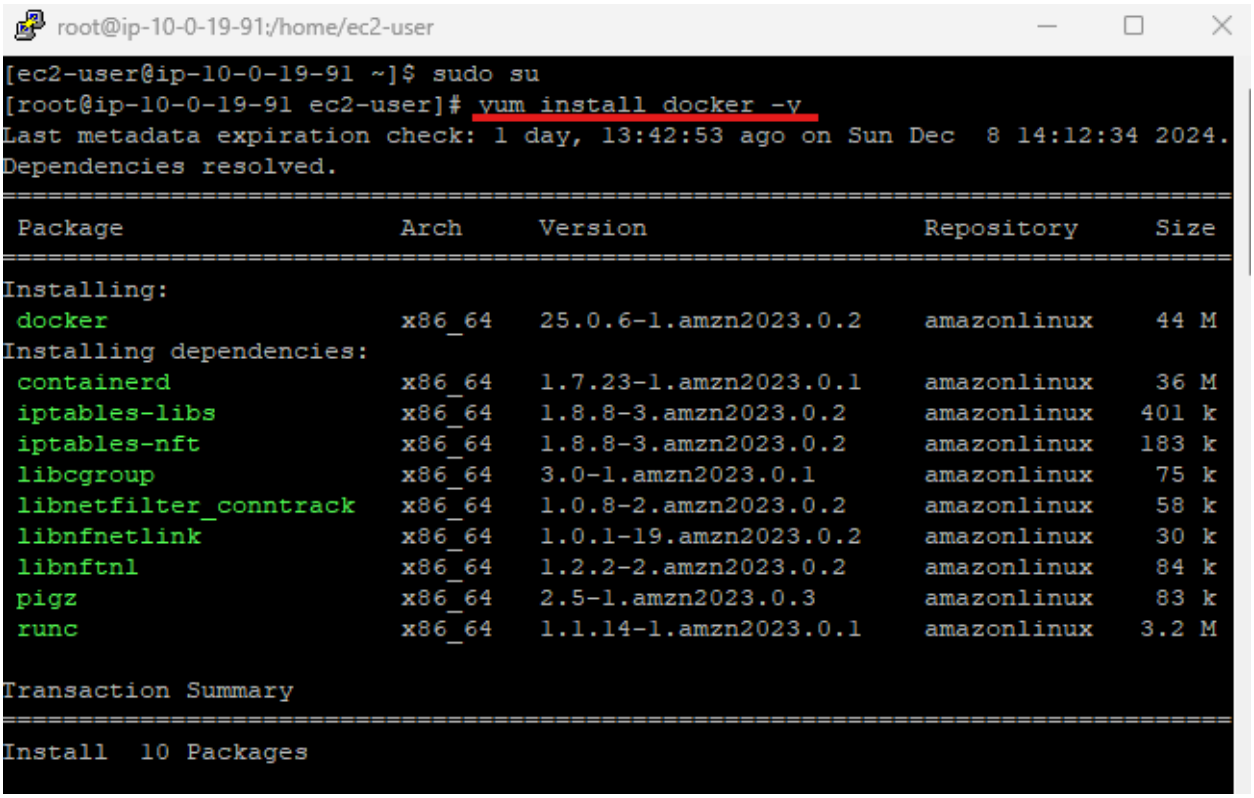


Fuente: elaboración propia.

Verificamos que el servicio este ejecutado correctamente accediendo a la IP, debemos ver el mensaje de nginx

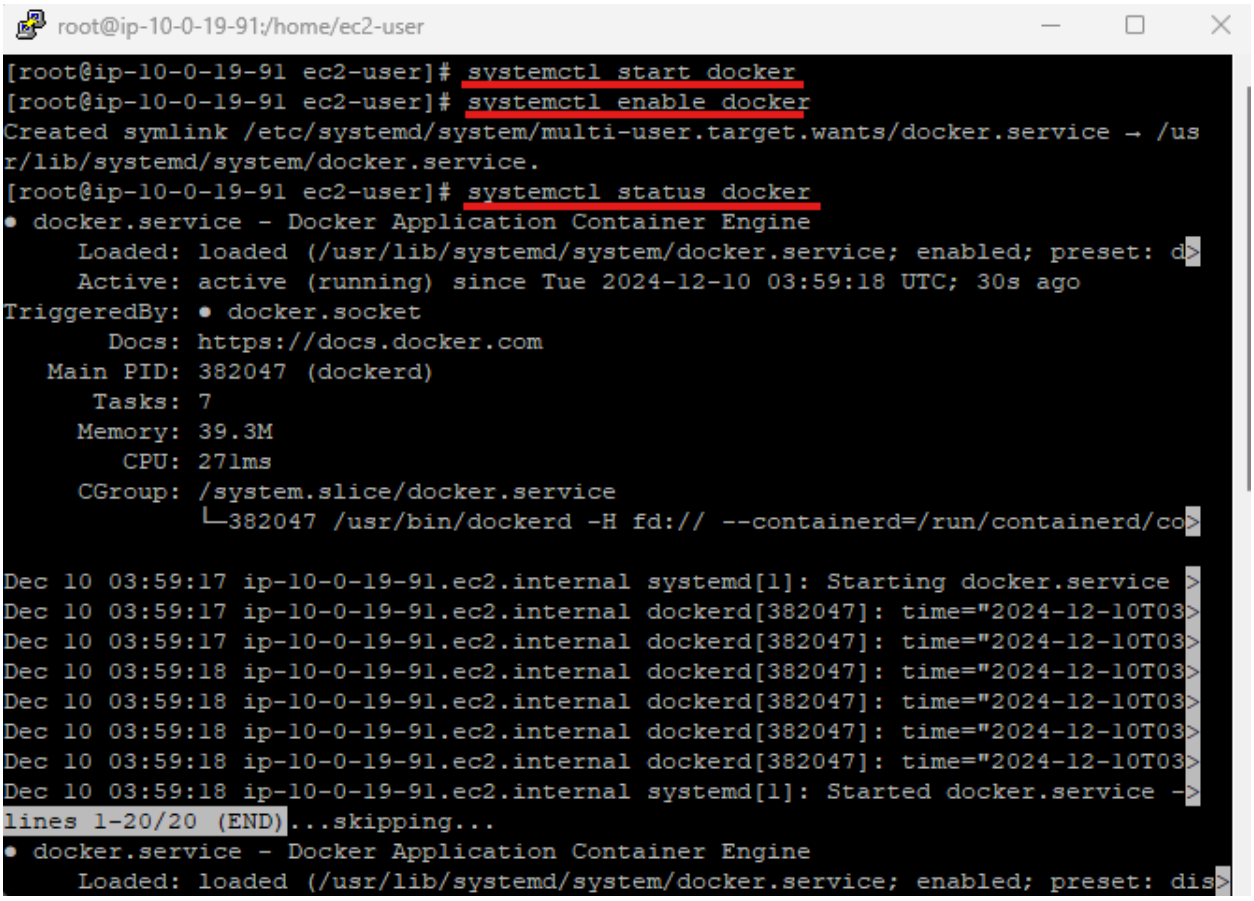


Procedemos con la instalación del Docker:



Fuente: elaboración propia.

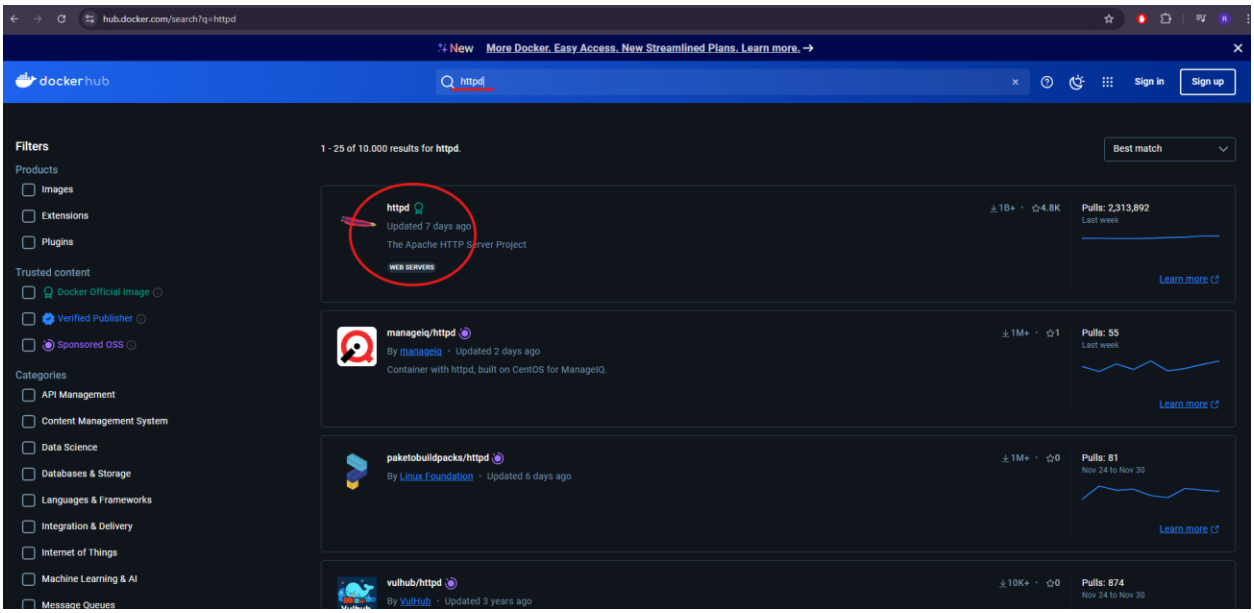
Iniciamos el servicio y establecemos que siempre se inicie automáticamente, luego verificamos el estado del servicio



Fuente: elaboración propia.

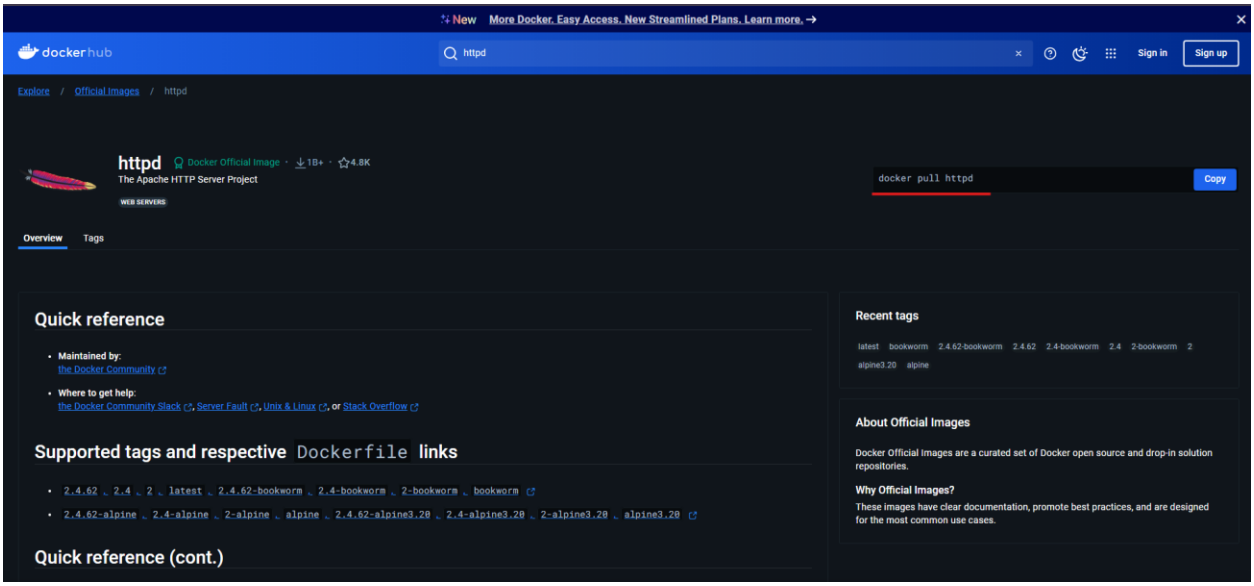
Ahora que tenemos el servicio de Docker ejecutándose podemos crear contenedores, los cuales son procesos dentro del sistema operativo.

Para crear un contenedor necesitamos una imagen, esta contiene la aplicación que vamos a utilizar en este caso un servidor web. La obtenemos del Hub de Docker en internet.



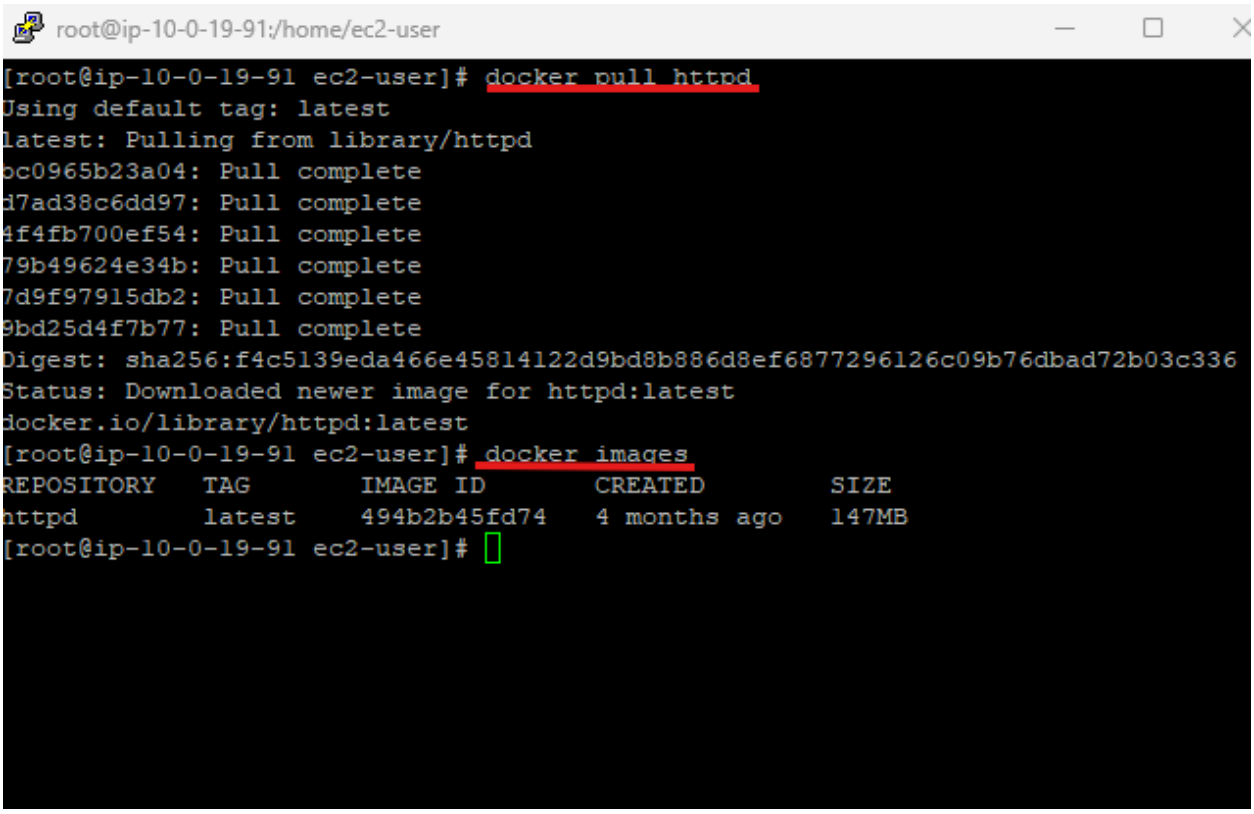
Fuente: elaboración propia.

Accedemos a la opción de httpd e usamos el comando que nos indica la información para realizar la instalación



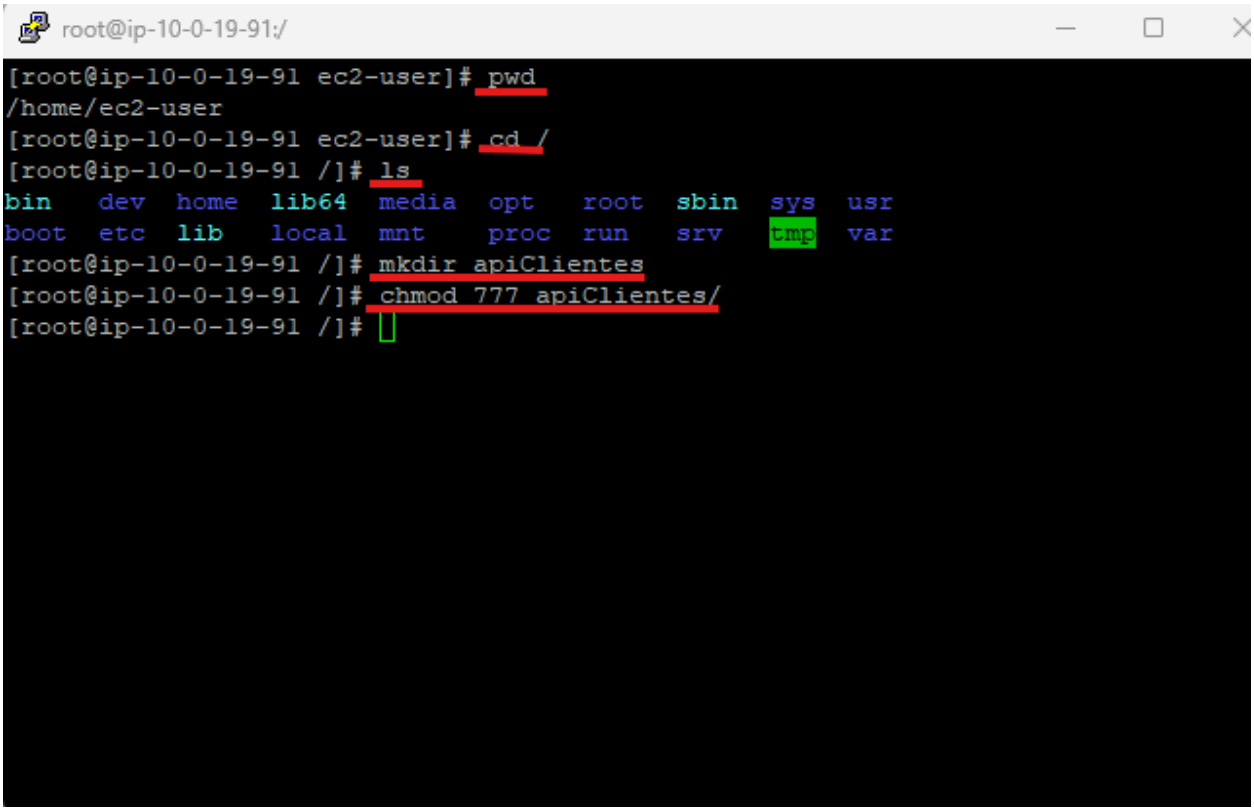
Fuente: elaboración propia.

Ejecutamos el comando de instalación y luego listamos las imágenes que tenemos en Docker, ahí veremos la que acabamos de crear (imagen httpd)



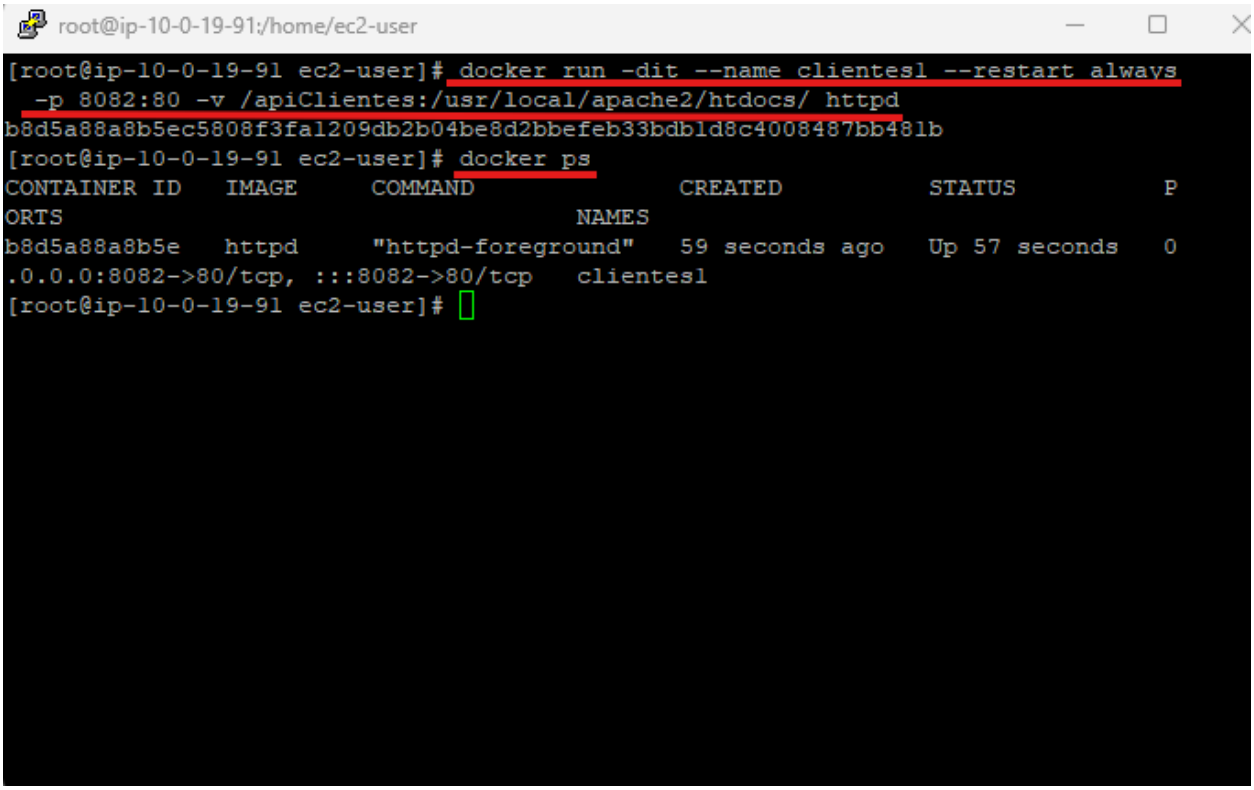
Fuente: elaboración propia.

Con la imagen instalada podemos crear contenedores de la siguiente manera, primero con el comando pow nos damos cuenta en que ubicación nos encontramos, posteriormente nos dirigíamos a la carpeta raíz con el comando cd / y luego creamos una carpeta con mkdir seguido del nombre de la carpeta para allí alojar el contenedor.



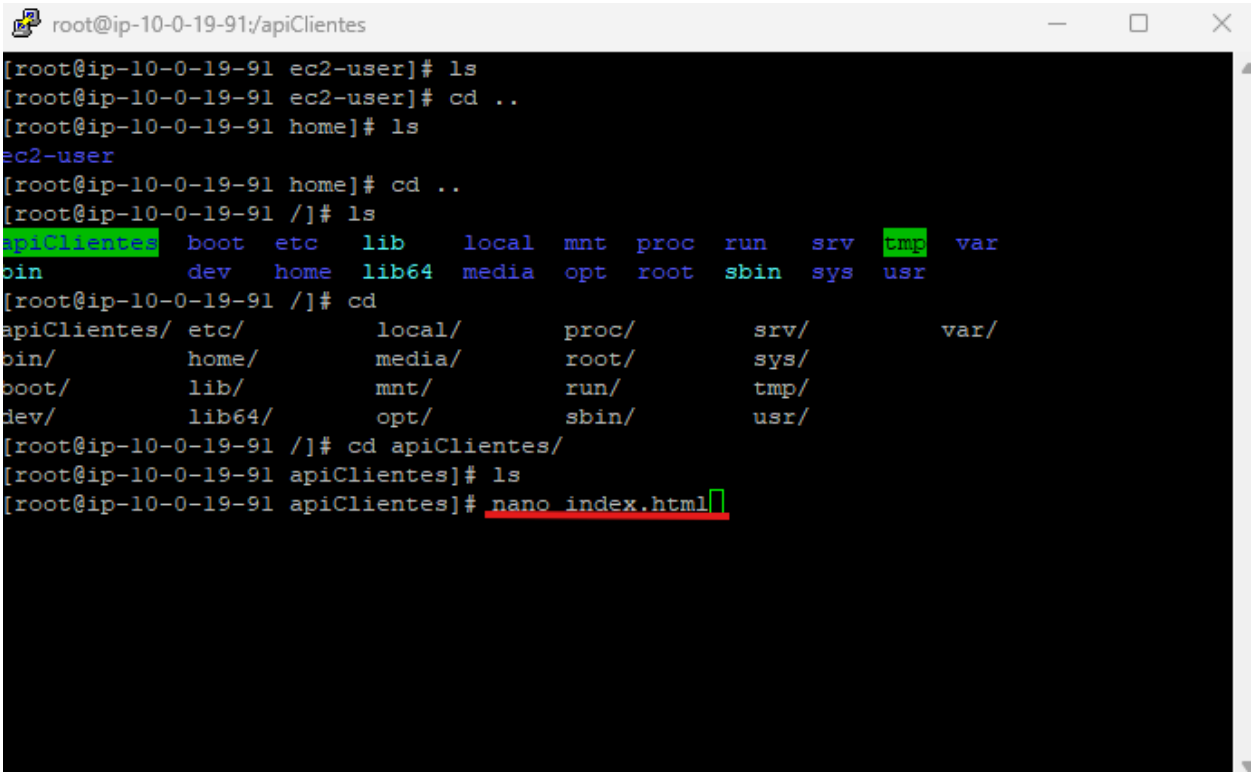
Fuente: elaboración propia.

Al tener la carpeta, creamos el contenedor y luego listo los contenedores



Fuente: elaboración propia.

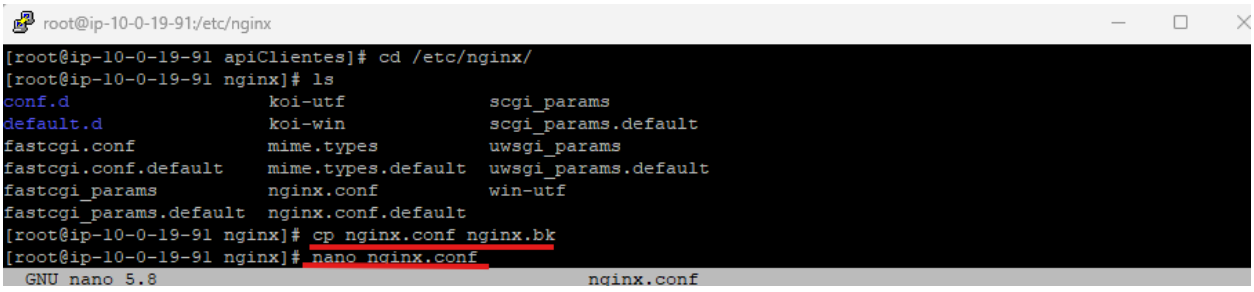
Nos ubicamos en la carpeta que creamos anteriormente y creamos un archivo



Fuente: elaboración propia.

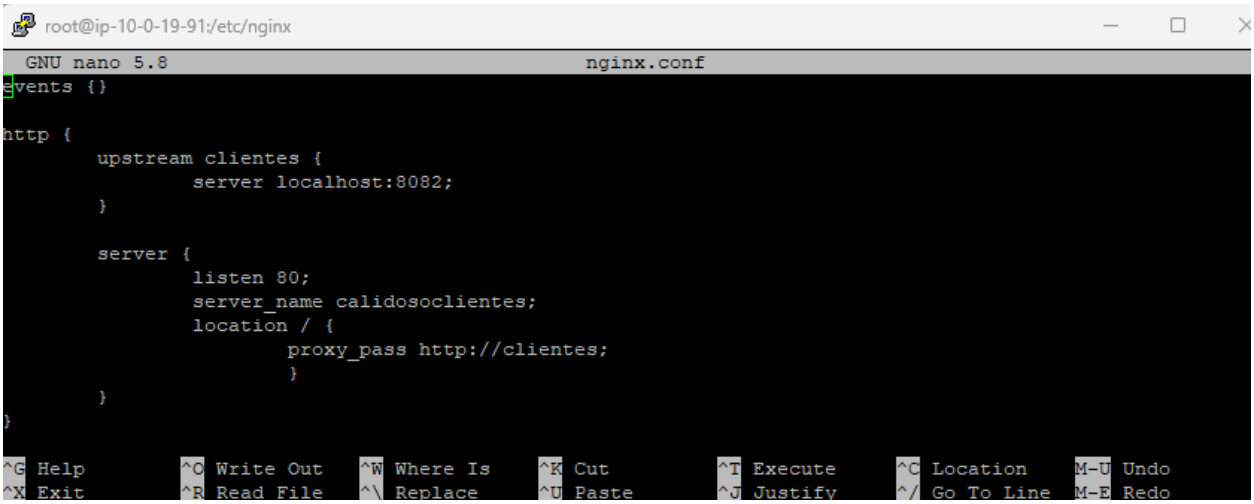
Para visualizar el servicio web creamos la regla en la instancia de permitir el paso

Configuramos el nginx para que se encargue de manejar las peticiones que hacemos a los contenedores, buscamos la configuración que tiene en etc/nginx y entramos al archivo nginx.conf, luego sacamos una copia de este para tener respaldo y luego editamos el archivo con la configuración que deseamos.



Fuente: elaboración propia.

Así es como queda la configuración del archivo



Fuente: elaboración propia

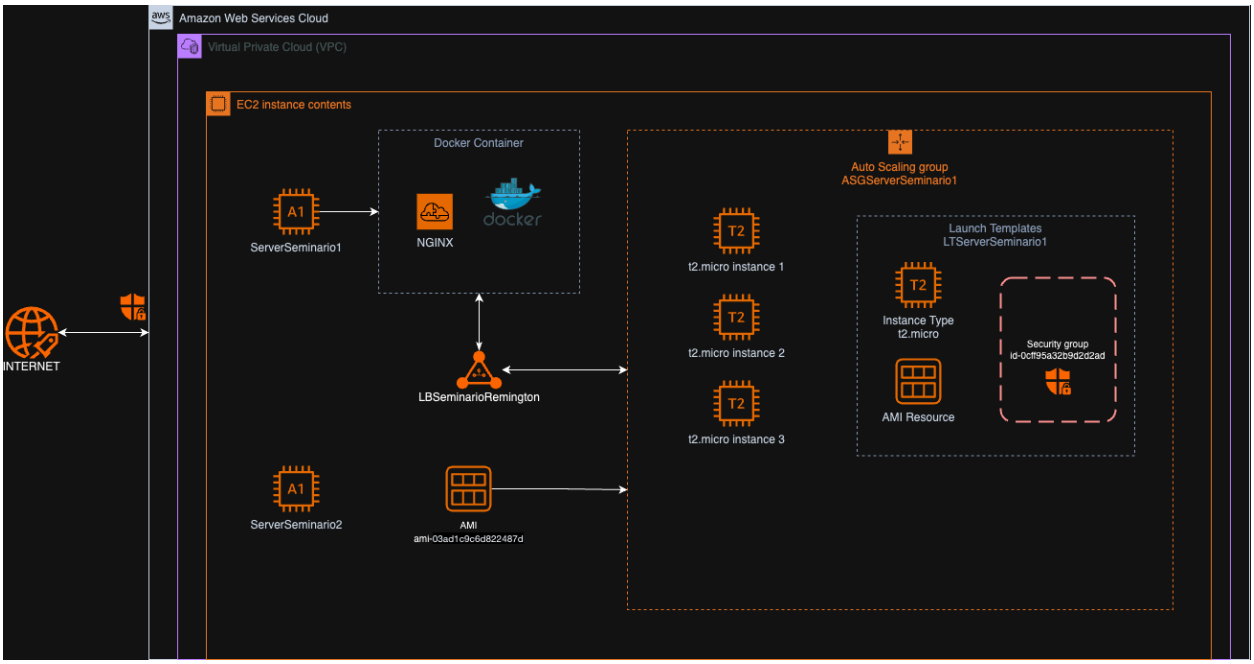
Reiniciamos el servicio del nginx y ahora podemos conectarnos a nuestro contenedor

ID	Nombre	Cliente	ID Sucursal	Teléfono
1	Juan Pérez	ABC Corp	101	555-1234
2	María López	XYZ Ltd	102	555-5678

Fuente: elaboración propia.

Tablas y Figuras

Diagrama de los recursos usados y la comunicación entre ellos.



Fuente: elaboración propia

Conclusiones

La implementación de servicios de AWS permite escalar dinámicamente los recursos, optimizando costos y garantizando alta disponibilidad. El uso de tecnologías como Docker y NGINX mejora la modularidad y gestión de aplicaciones, permitiendo una infraestructura ágil. El diseño de esta solución demuestra cómo los conceptos teóricos del seminario pueden aplicarse a problemas reales, mostrando un impacto positivo en startups en crecimiento. La configuración de herramientas como Auto Scaling y LB refuerza la capacidad de manejar picos de demanda sin comprometer el rendimiento.

Referencias

Amazon Web Services. (n.d.). Amazon Web Services (AWS) – Cloud computing services. https://docs.aws.amazon.com/

Docker, Inc. (n.d.). Docker Hub. https://hub.docker.com - https://hub.docker.com/_/httpd

Berrio, J. (2024, noviembre 19 – diciembre 6). *Seminario Amazon Web Service.* Corporación universitaria Remington, Microsoft Teams.