TRABAJO DE GRADO
Opción Seminario-Diplomado.

Cloud Fats

Corporación Universitaria Remington.

Ingeniería
Tecnología desarrollo de software

Presentado:

Andres Julian Uribe Correa

Docente:
Juan Pablo Berrio Lopez
Seminario
2024

Tabla de Contenidos

Tabla de contenido

Resumen

El siguiente trabajo es implementado con el fin de tener un entorno cloud de alta disponibilidad en aws por medio de instancias y contenedores, para esto se desarrollaron configuraciones de vpc en el cual se configura las estructura de conectividad, ec2 se implementan las instancias de los sistemas operativos, auto Scaling el cual permite tener el numero deseado de instancias disponibles al mismo tiempo, Docker con el cual implementaremos los contenedores. entre otras herramientas las cuales dependerán una de la otras logrando así el objetivo de tener un alto rendimiento de disponibilidad.

Palabras clave

Las palabras claves del trabajo son:
Vpc.
Ec2.
Docker.
Contenedores.
Auto Scaling.

Marco conceptual y contextual

Este trabajo es de computaciones en la nube enfocado en aws los cuales son unos servicios de cómputo en la nube con el fin de lograr que estos servicios configurados logren estar la mayor parte del tiempo disponibles a los usuarios.
Para ello se configuran herramientas como vpc que permite configurar un espacio de enlace por medio de direcciones ip y grupos de conectividad, el putty es una aplicación que nos permite conectar por medio de un protocolo SSH a las instancias creamos por medio de ec2 la cual permite crear una instancia que va a tener un sistema operativo en el cual se configuran herramientas como apaches que son servicios que nos permiten tener una página web, nginx que permite tener un proxy reverso de conectividad sobre varios contenedores configurados por medio de la aplicación Docker y sus distintos puertos de red que permiten la navegación sin conflictos.
Tenemos el auto Scaling que permite tener instancias a necesidad de la implementación permitiendo tener de estar forma un sistema disponible para los usuarios en el momento que ellos desean conectar.

AWS

Entrega 1

Instancias y recursos de aws que permite ejecutar instancias de aws.

2. Instancias en Amazon Lightsail

- Diseñadas para ser fáciles de usar y económicas, ideales para usuarios que no requieren la flexibilidad completa de EC2.
- Incluyen una combinación de cómputo, almacenamiento y ancho de banda preconfigurado.

3. Instancias de Contenedores (Amazon ECS y AWS Fargate)

- **Amazon ECS (Elastic Container Service):** Permite ejecutar contenedores en instancias EC2 o mediante el servicio administrado AWS Fargate.
- **AWS Fargate:** Elimina la necesidad de administrar instancias directamente; ejecuta contenedores sin preocuparse por la infraestructura subyacente.

Entre las principales tenemos EC2:

1. Amazon EC2 (Elastic Compute Cloud)

En un servicio principal de aws el ec2 permite un entorno escalable.

Componentes Clave:

- AMIs (Amazon Machine Images): Plantillas preconfiguradas que definen el sistema operativo, software, y configuraciones de las instancias.
- Tipos de Instancia: Especifican los recursos de asignados a una instancia.

2. Almacenamiento

Las instancias y guardar sus aplicaciones y archivos.

- persisten incluso si la instancia se detiene.
- Instancia de Almacenamiento: Almacenamiento temporal vinculado a la instancia y que se pierde al detenerla.
- Amazon S3 almaciga obejtos

3. Redes

La conectividad de red es fundamental para que las instancias puedan comunicarse con otros recursos de manera eficiente. Dentro de AWS, se utilizan diversas herramientas y configuraciones para gestionar estas conexiones:

- Amazon VPC (Virtual Private Cloud): Permite crear redes virtuales aisladas donde se ejecutan las instancias de manera segura.
- Subredes: Son divisiones dentro de una VPC que organizan las instancias en diferentes zonas de disponibilidad.

4. Auto Scaling yLoad Balancing

Para asegurar que las aplicaciones sean altamente disponibles y puedan manejar cambios en la demanda, AWS ofrece las siguientes herramientas:

- Auto Scaling: Ajusta automáticamente la cantidad de instancias en función de la carga de trabajo, garantizando eficiencia y optimización de recursos.
- Elastic Load Balancer (ELB): Distribuye el tráfico de red entre múltiples instancias, mejorando el rendimiento y la tolerancia a fallos**.**

2. Implementando un servidor web en Amazon Linux:

Figuras y tablas

Figura 1. Creación y configuración e vpc

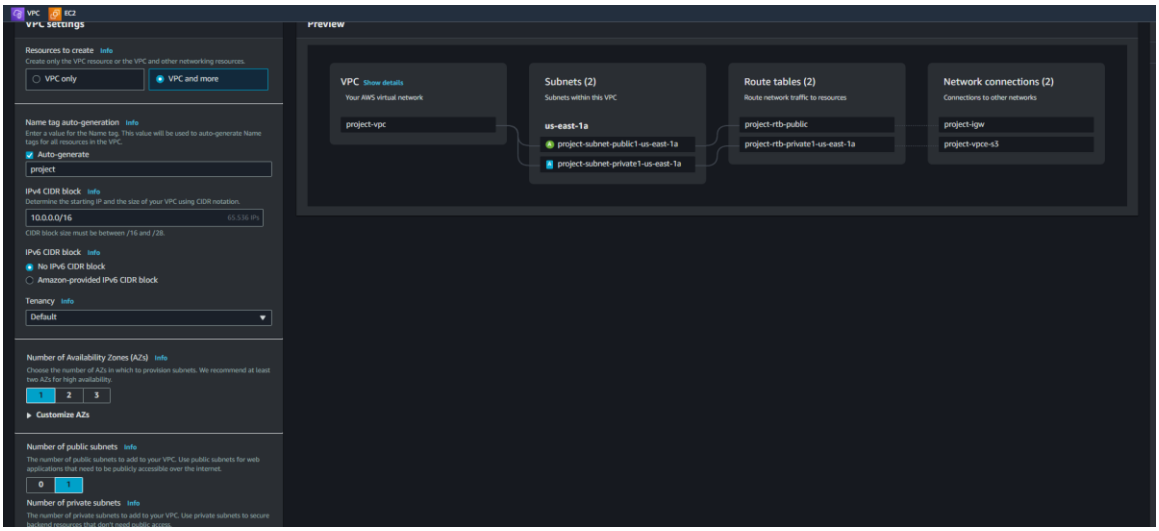


Figura 2. Creamos el certificado

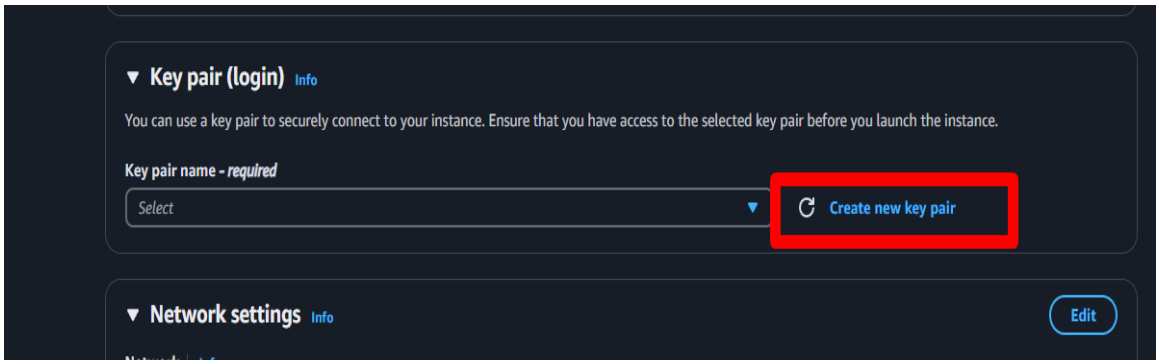
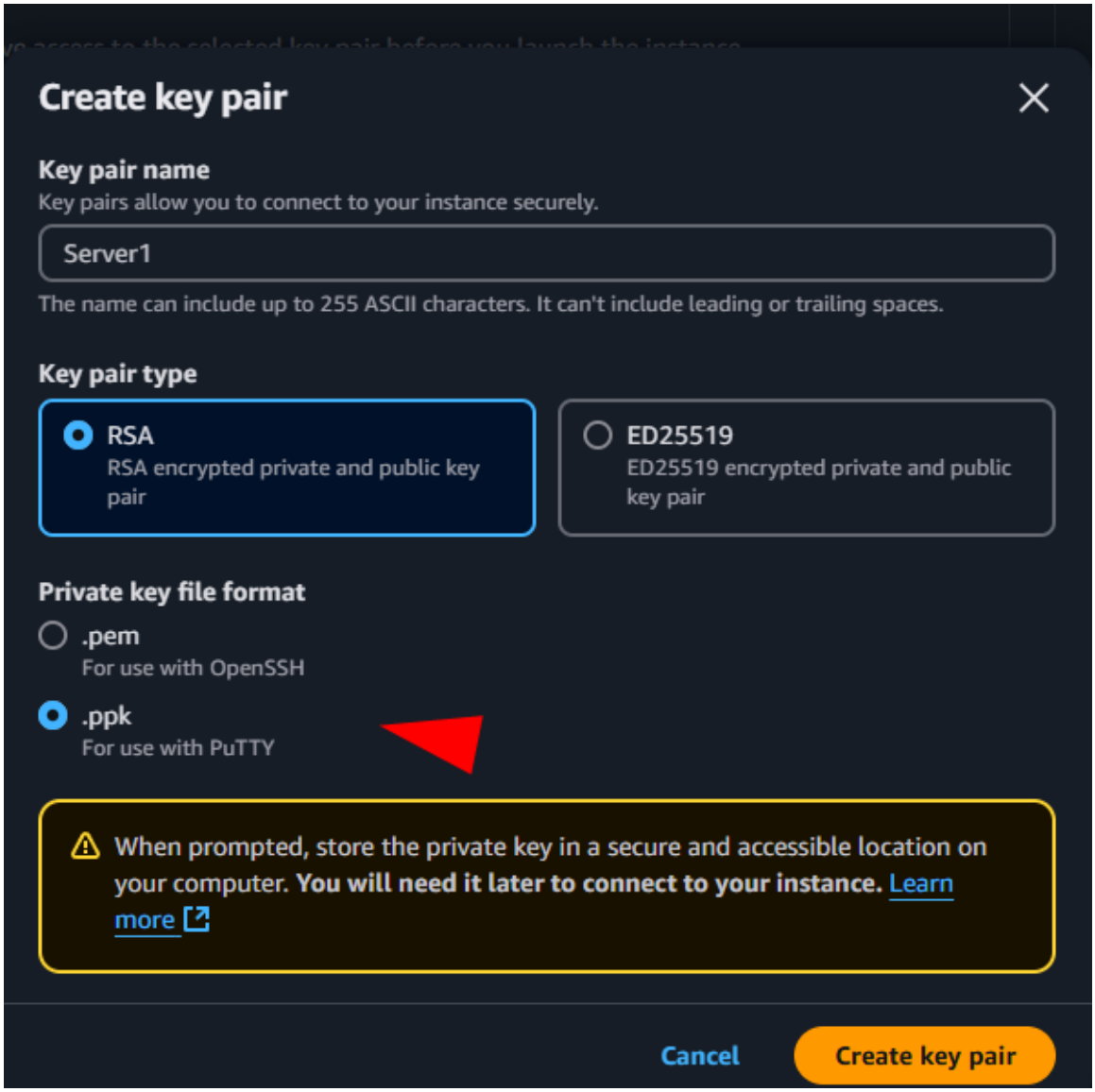


Figura 3. Seleccionamos para conectar por medio de putty



Le damos crear y guardamos el certificado

Figura 4. Configuración de red

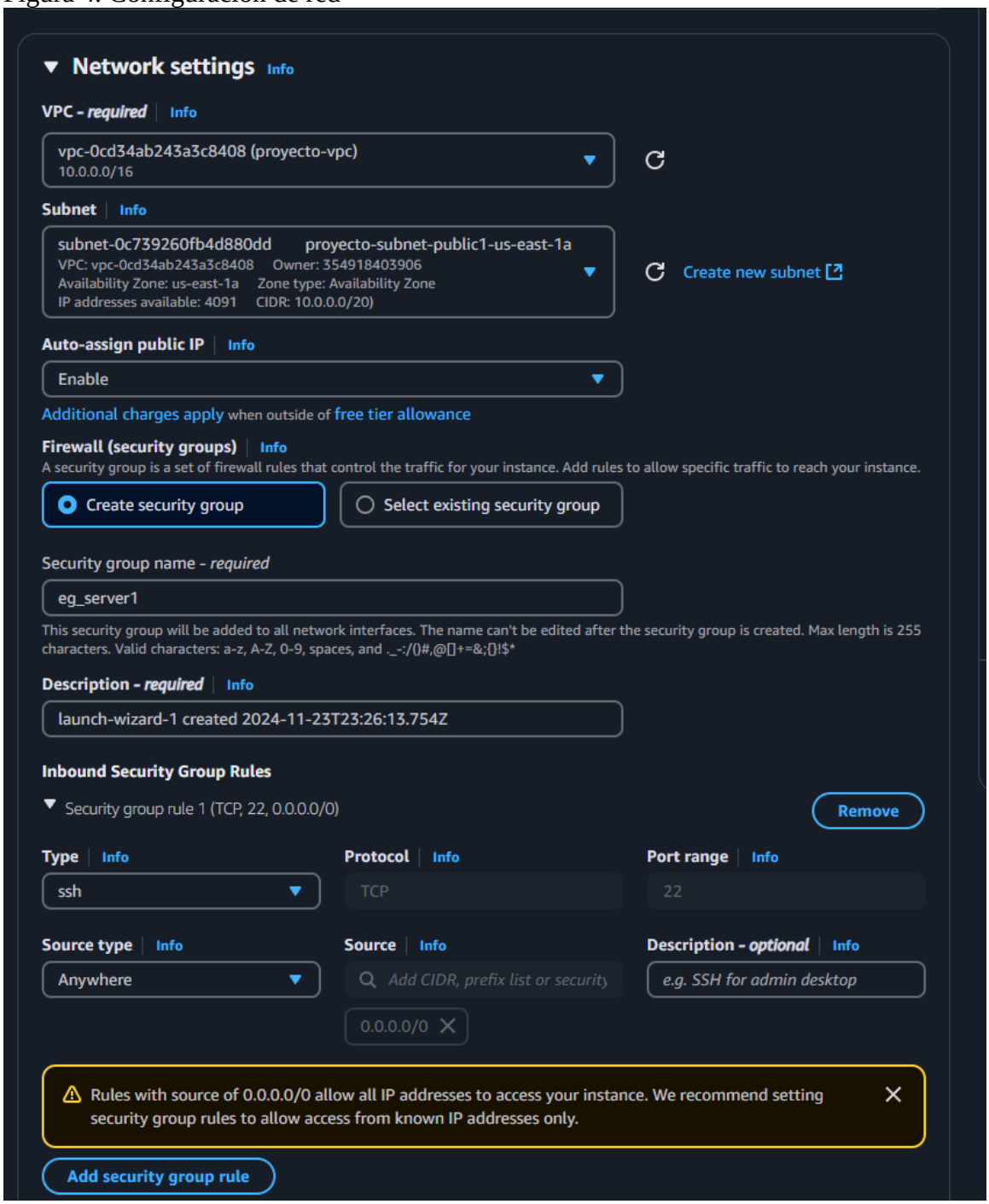


Figura 5. lanzamos la instancia

Figura 6. Ingresando al id de la instancia podemos obtener datos como los grupos de seguridad
Como por ejemplo revisar los puertos de salida e ingreso

Ejemplo desde afuera hacia el interior por el puerto 22 recibo desde cualquier lugar conexiones.

Figura 7. Procedemos a conectar a la maquina.

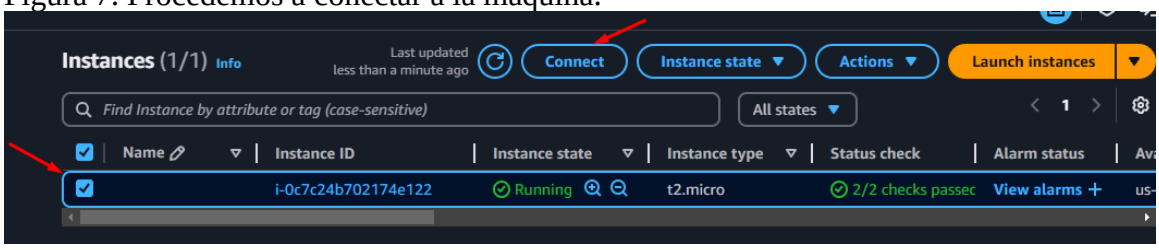


Figura 8. Procedemos a conectar desde Putty.

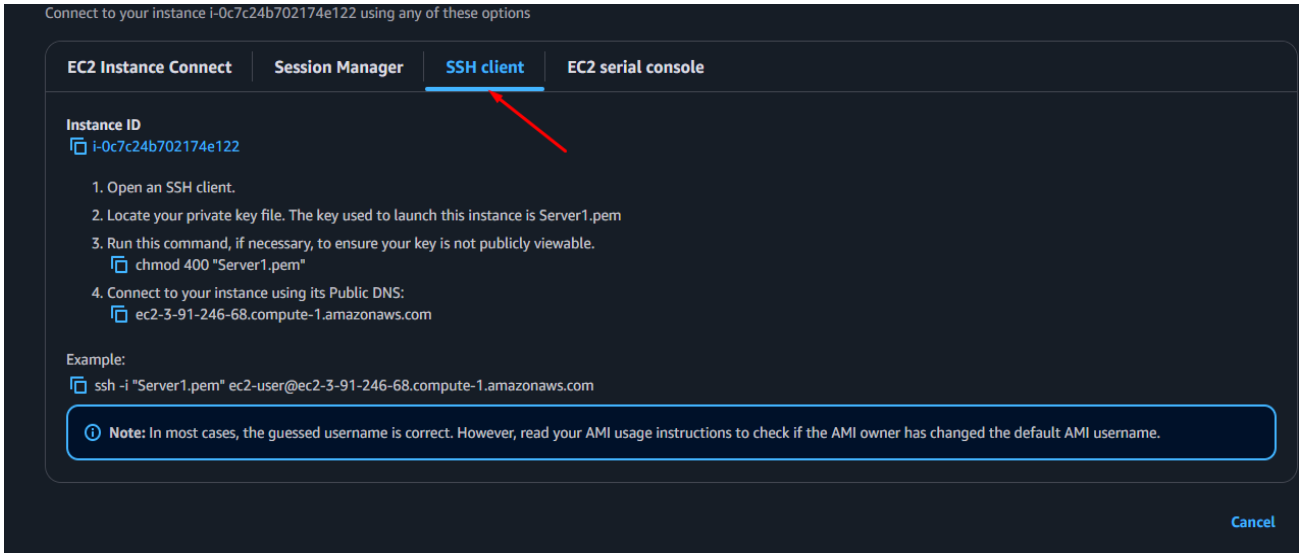


Figura 9. Abrir aplicación putty.

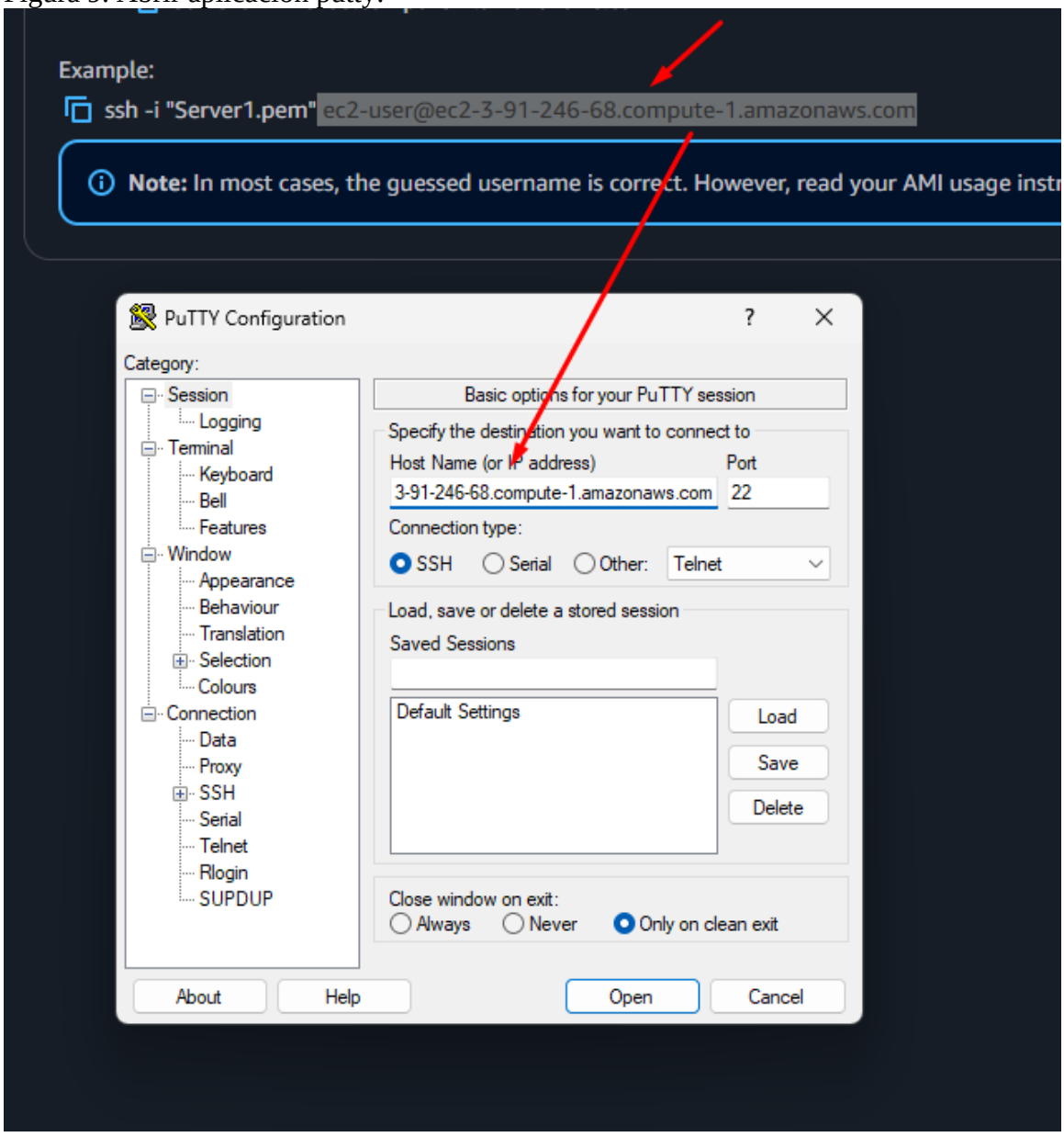


Figura 10. Conectamos por medio de certificado.

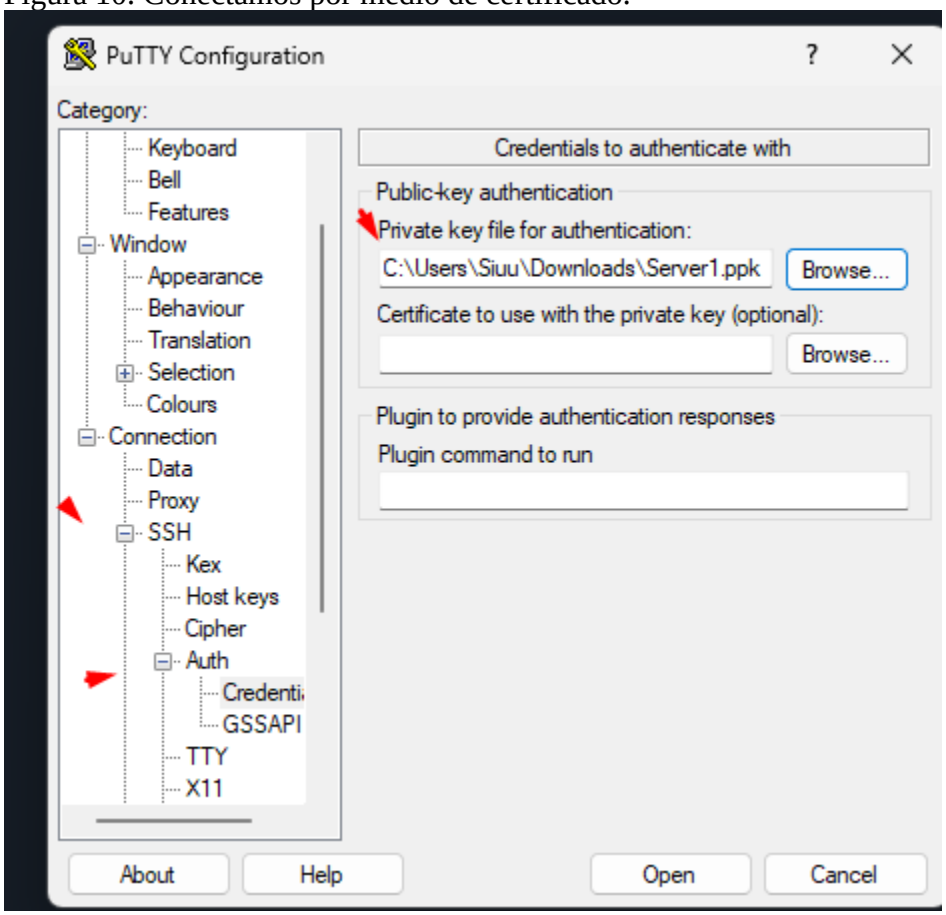


Figura 11. Ingreso a Linux.

Figura 12. Instalación paquete apache le damos yes.

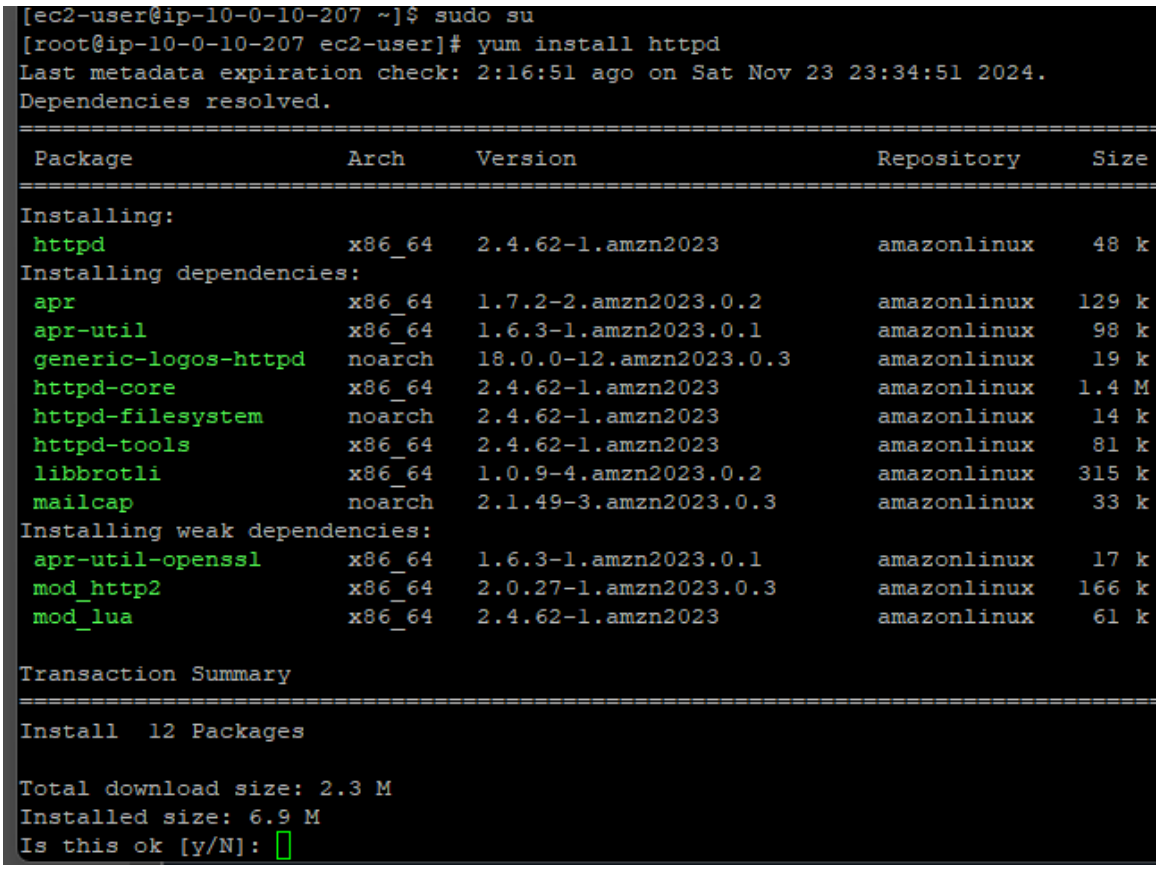


Figura 13. Revisamos estatus.

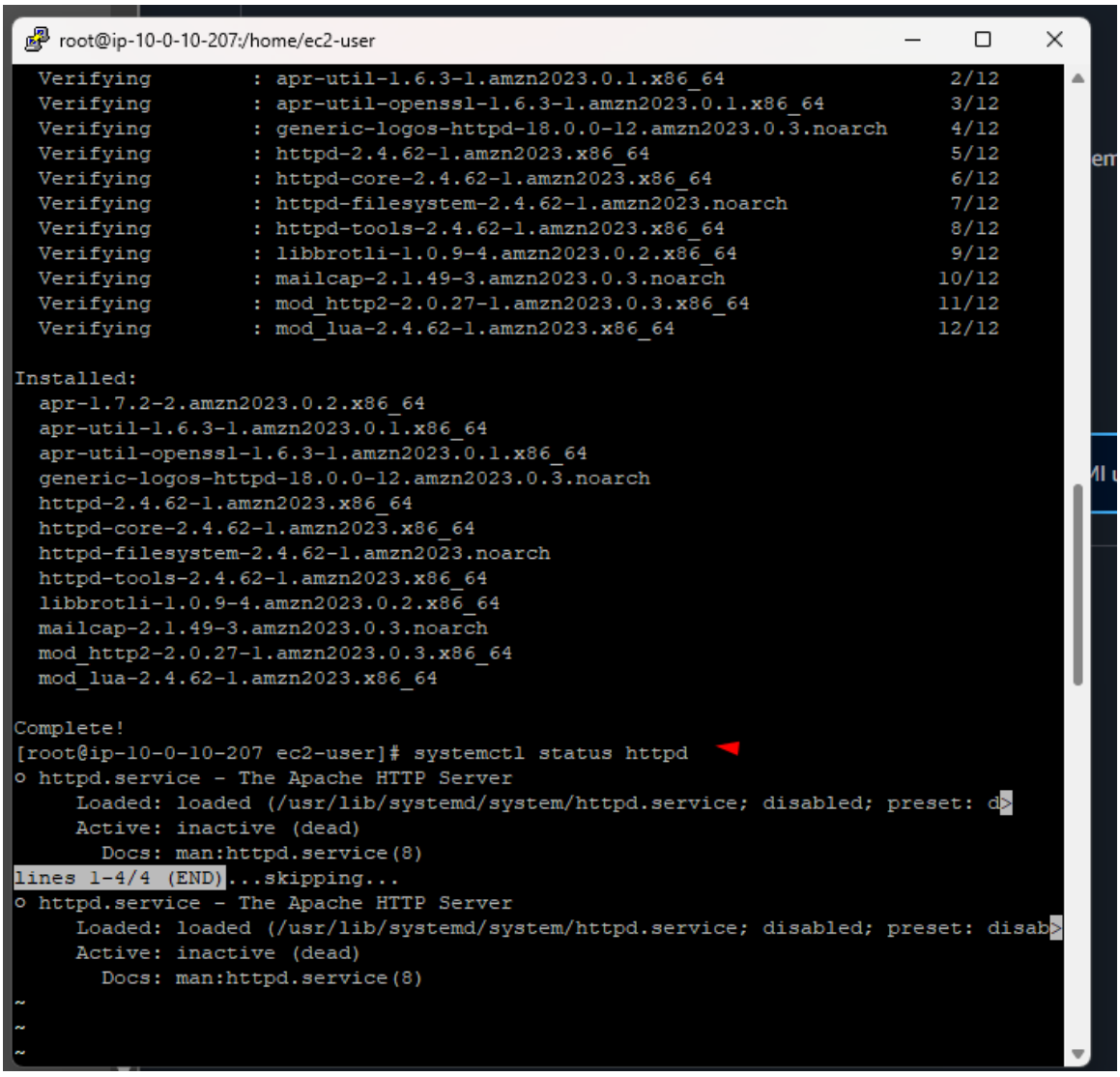


Figura 14. Activamos el servicio.

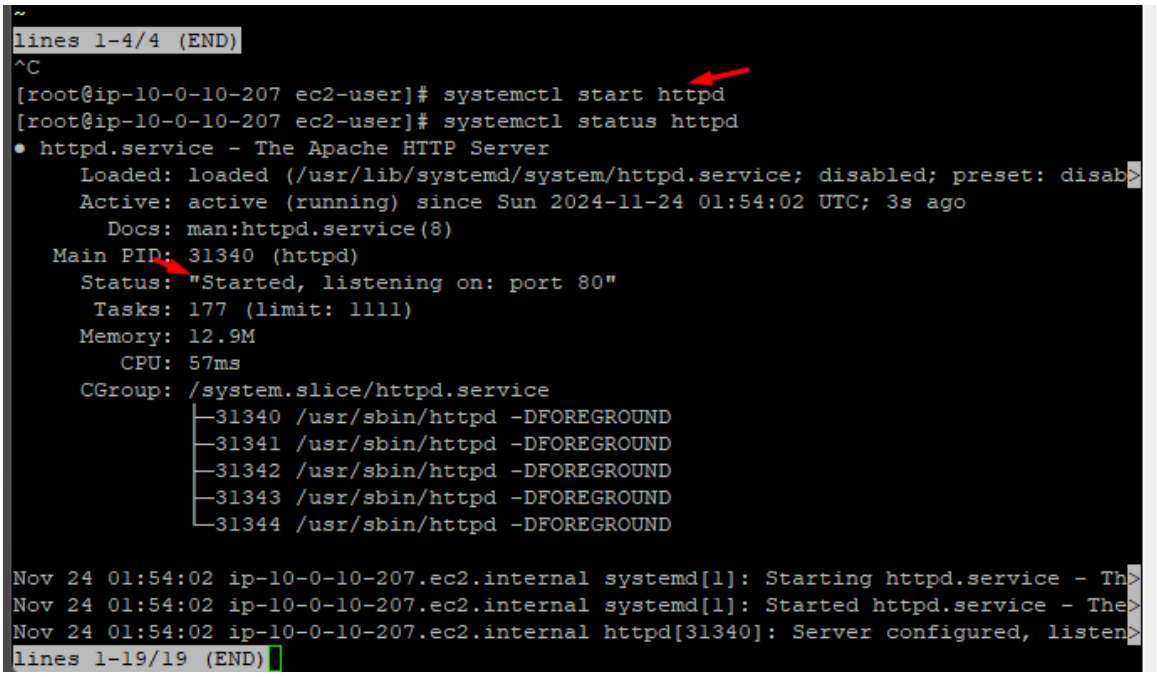
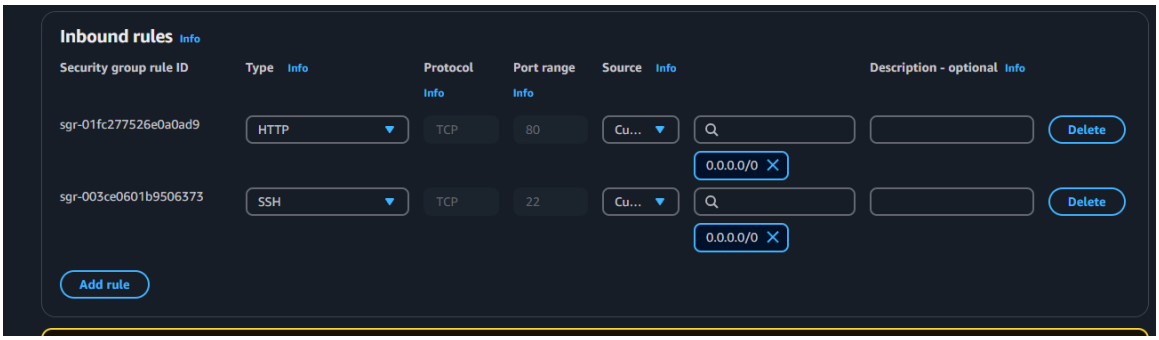


Figura 15. Agregamos puerto 80.



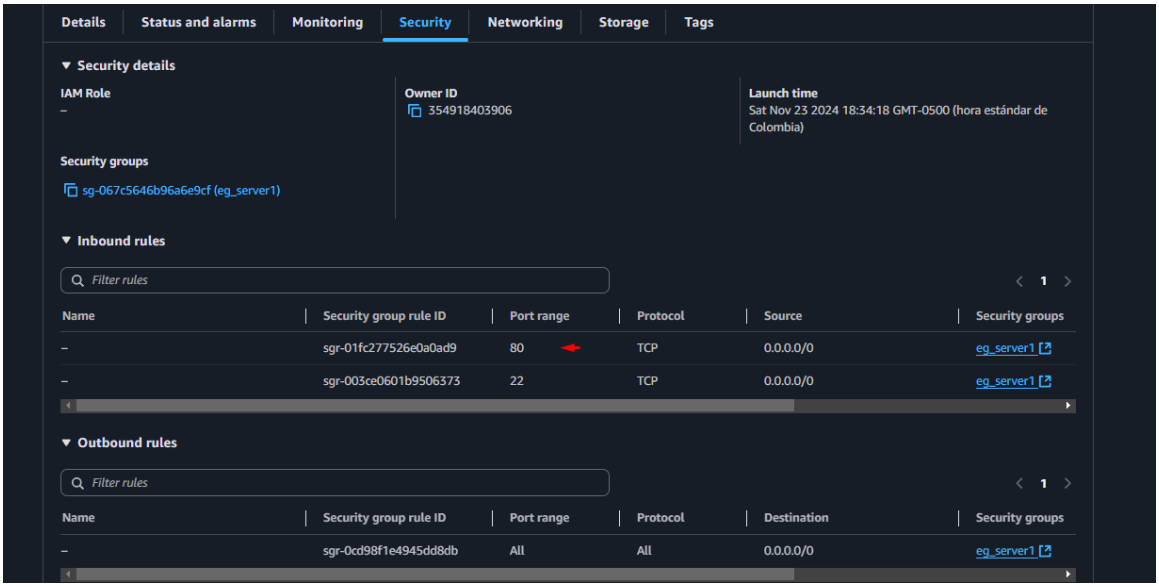


Figura 16. Conectamos a la ip publica.

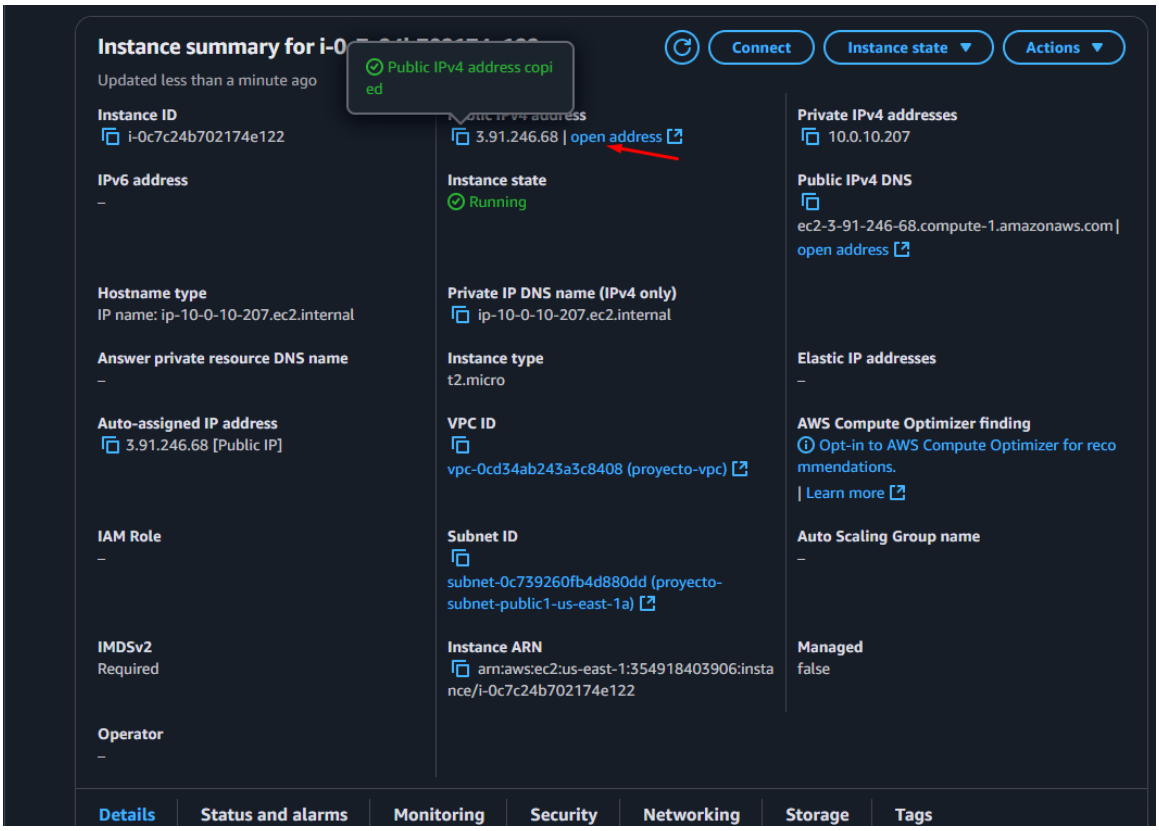


Figura 16. Conectar a ip.

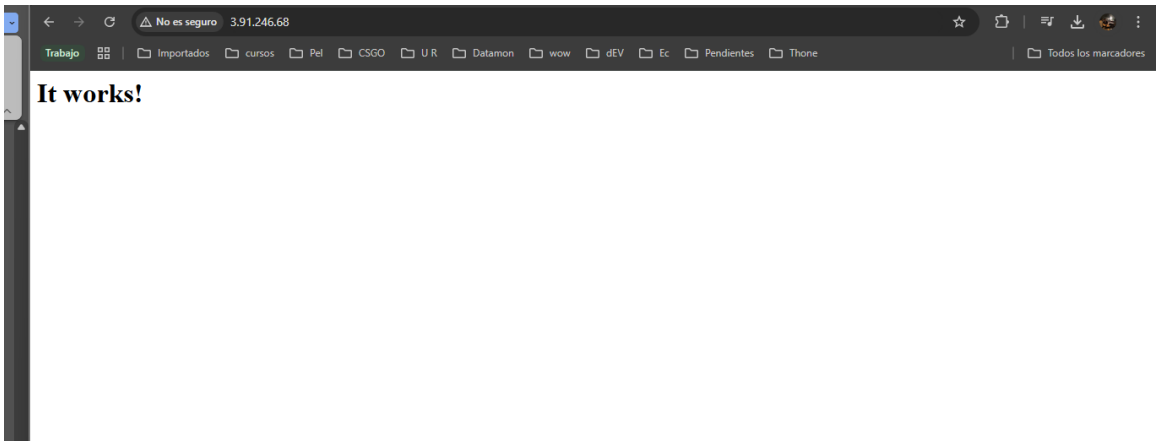


Figura 17. Ingresamos desde el Linux a la carpeta de la web y creamos un índex.

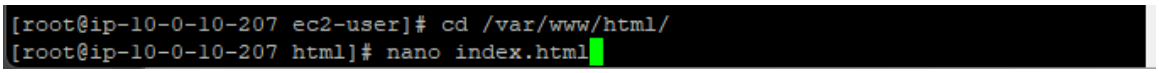


Figura 18. Le damos permisos al archivo.

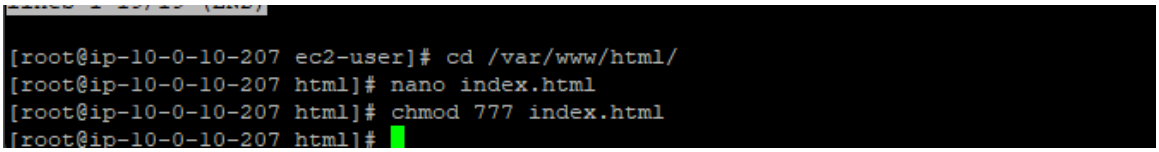
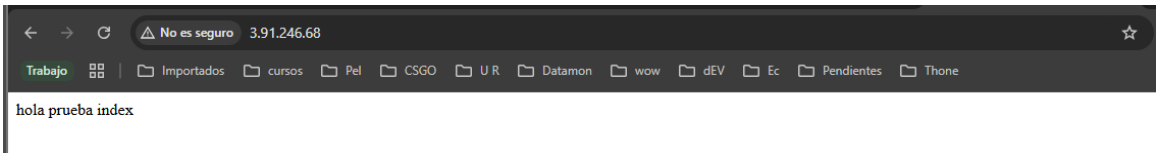


Figura 19. Verificamos cambios en la nube.



Video Entrega 1:

https://youtu.be/qm5SF8yIAMo

Entrega 2

Figura 20. Iniciaremos creando un snapshot para crear luego la imagen

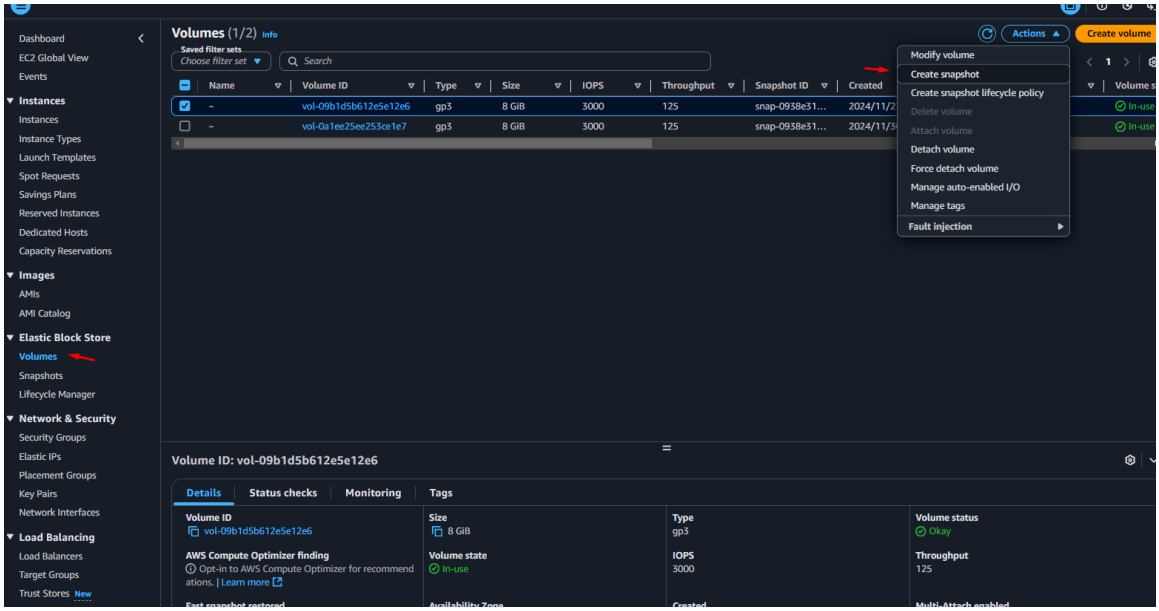


Figura 21. Luego creamos una imagen (ami)

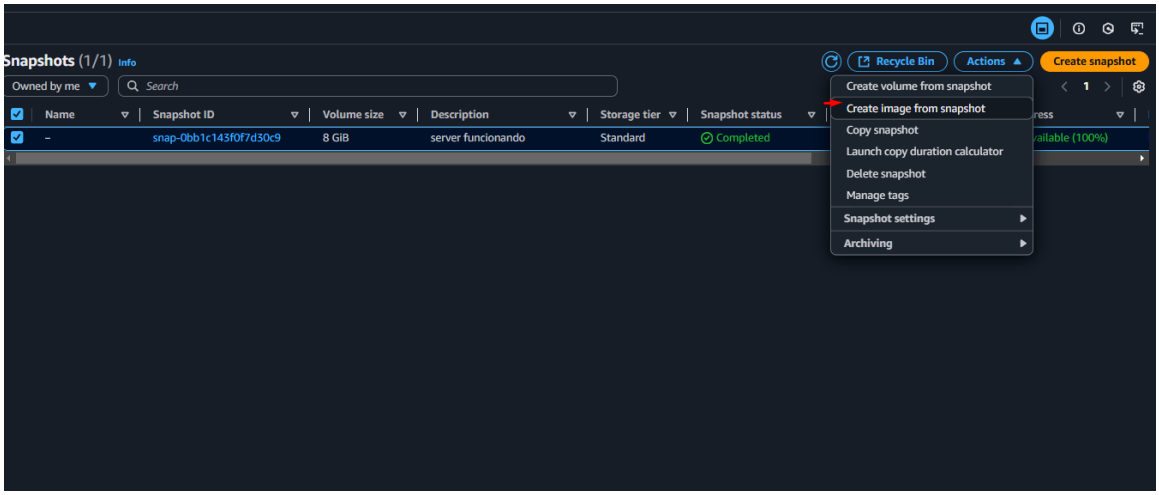


Figura 22.

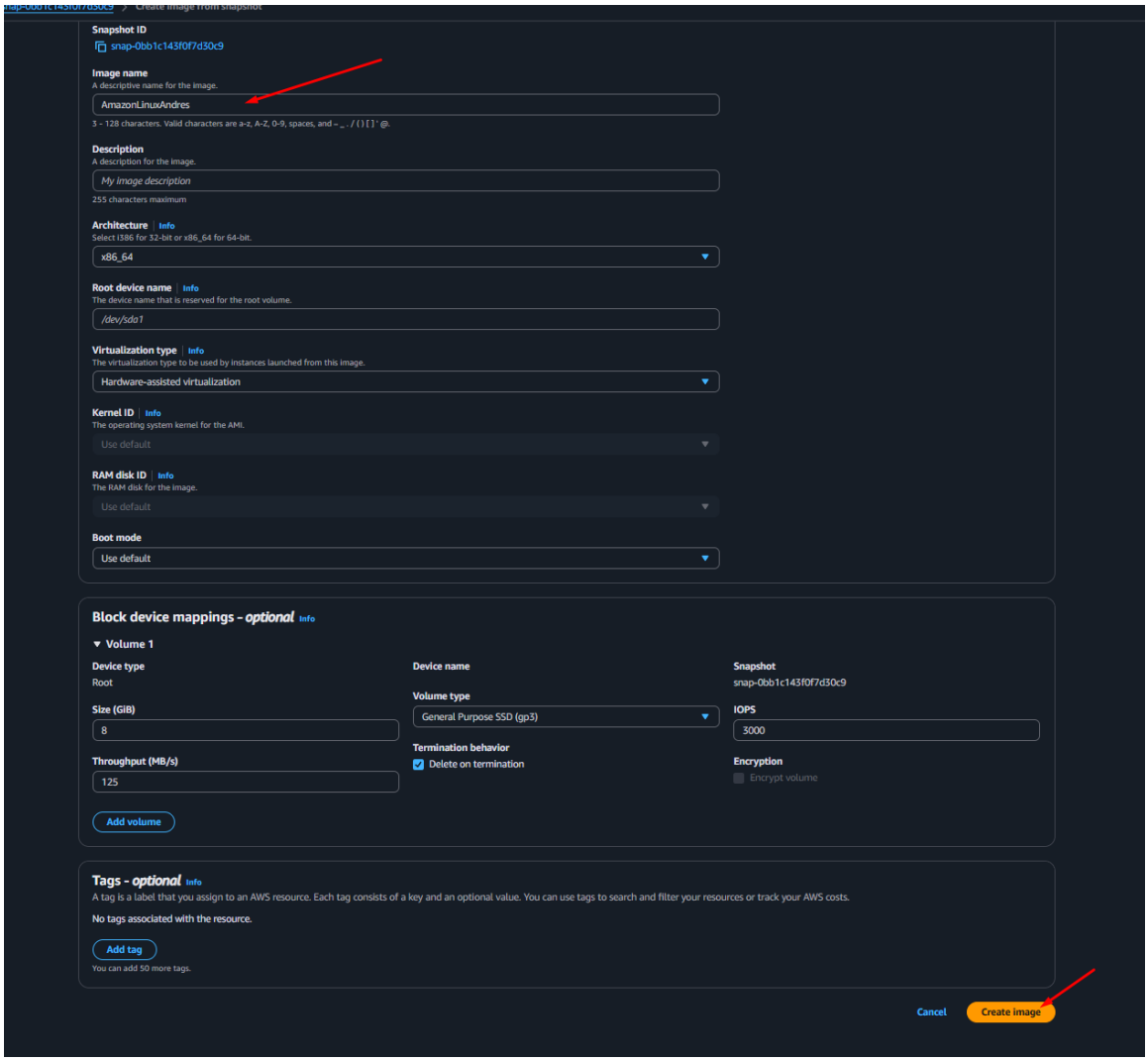
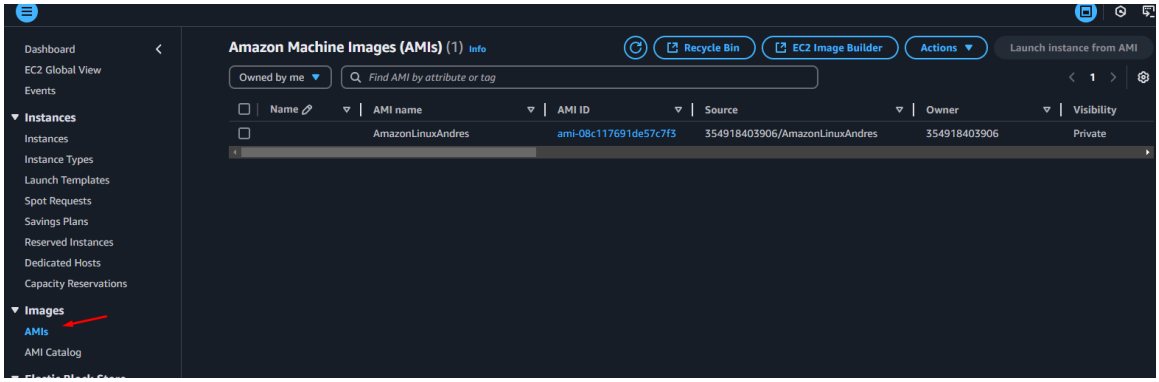


Figura 23 . verificamos la ami



Balanceadores de carga

Figura 24. Crear otra instancia de Linux con las mismas configuraciones anteriores

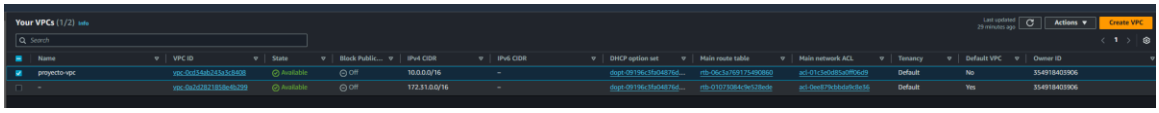
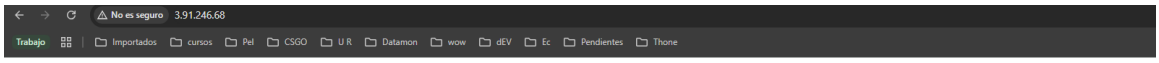


Figura 25. Valíamos el entrono web de las dos instancias



hola server 1 prueba index

server 2222222222222222

Figura 26. Iniciamos proceso de crear balanceador de carga

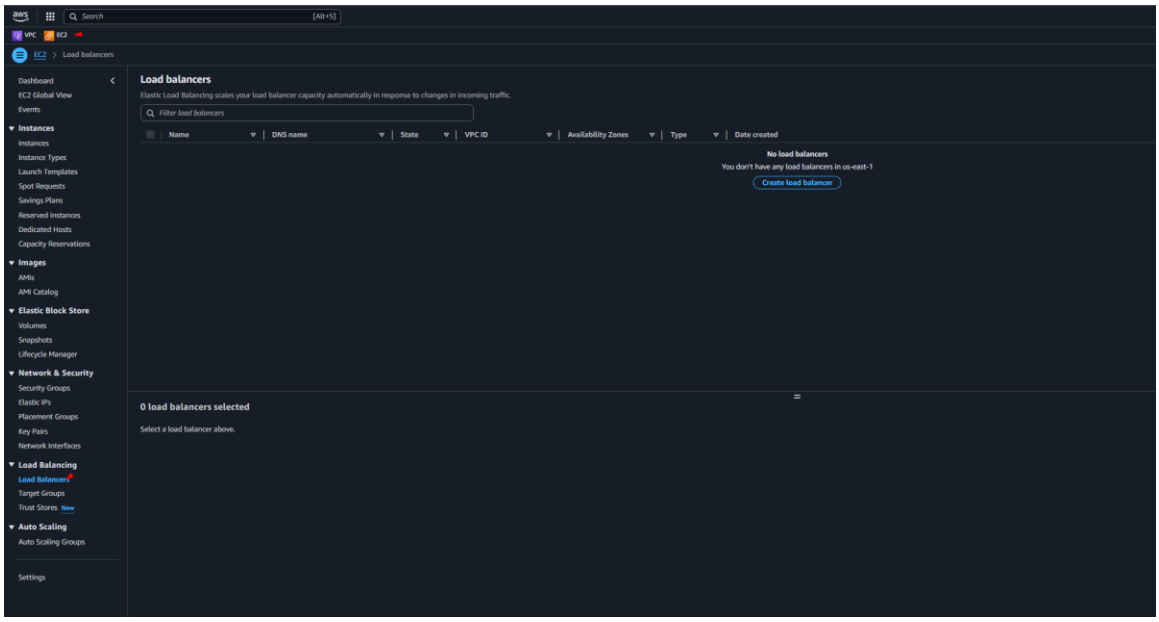


Figura 27. Seleccionamos **Application Load Balancer**

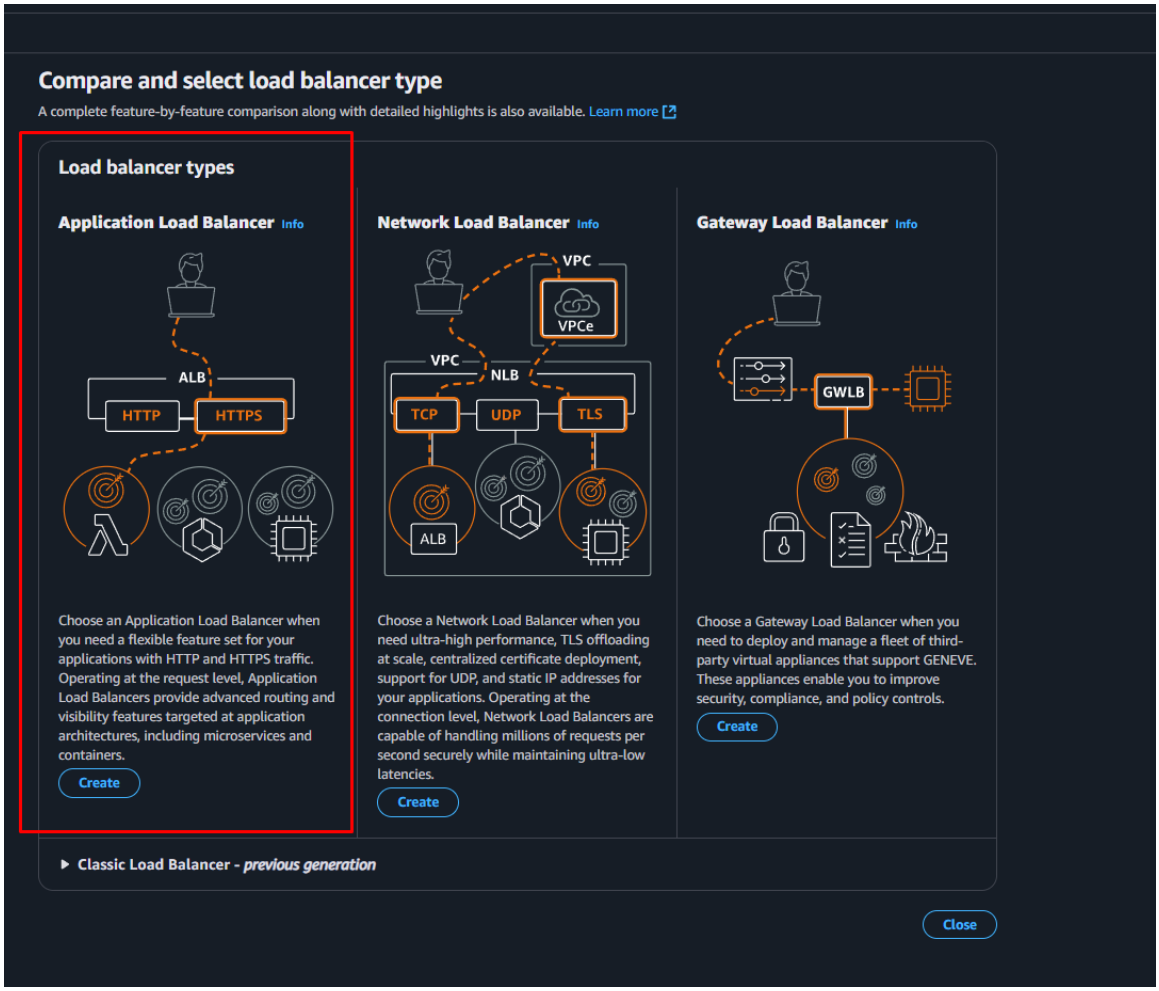


Figura 28. Configuramos el balanceador importante seleccionar el mismo vpc

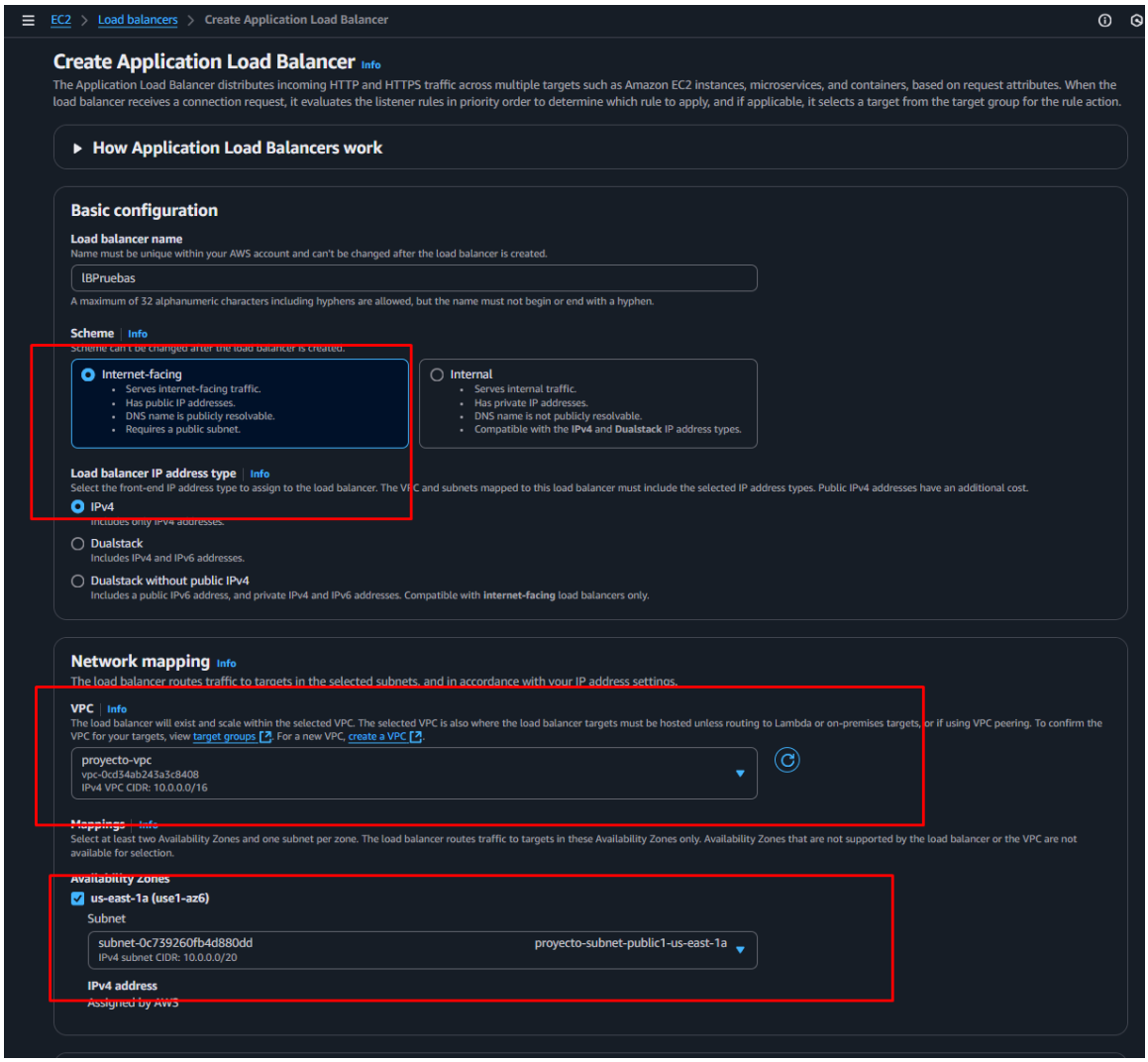


Figura 29. Se debe crear su propio grupo de seguridad.

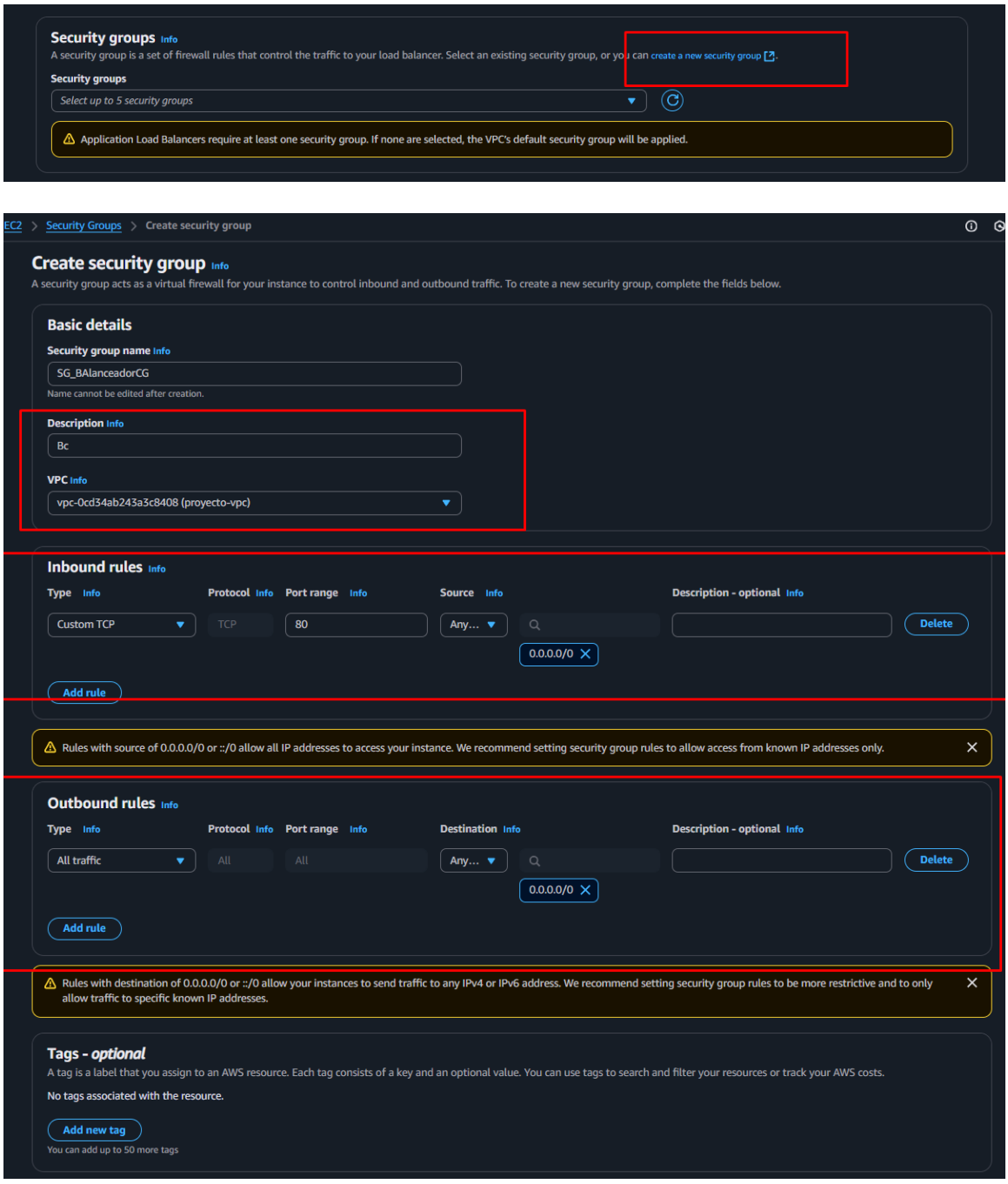


Figura 30. En el listener configuramos para que entre desde internet y las maquinas objetivo

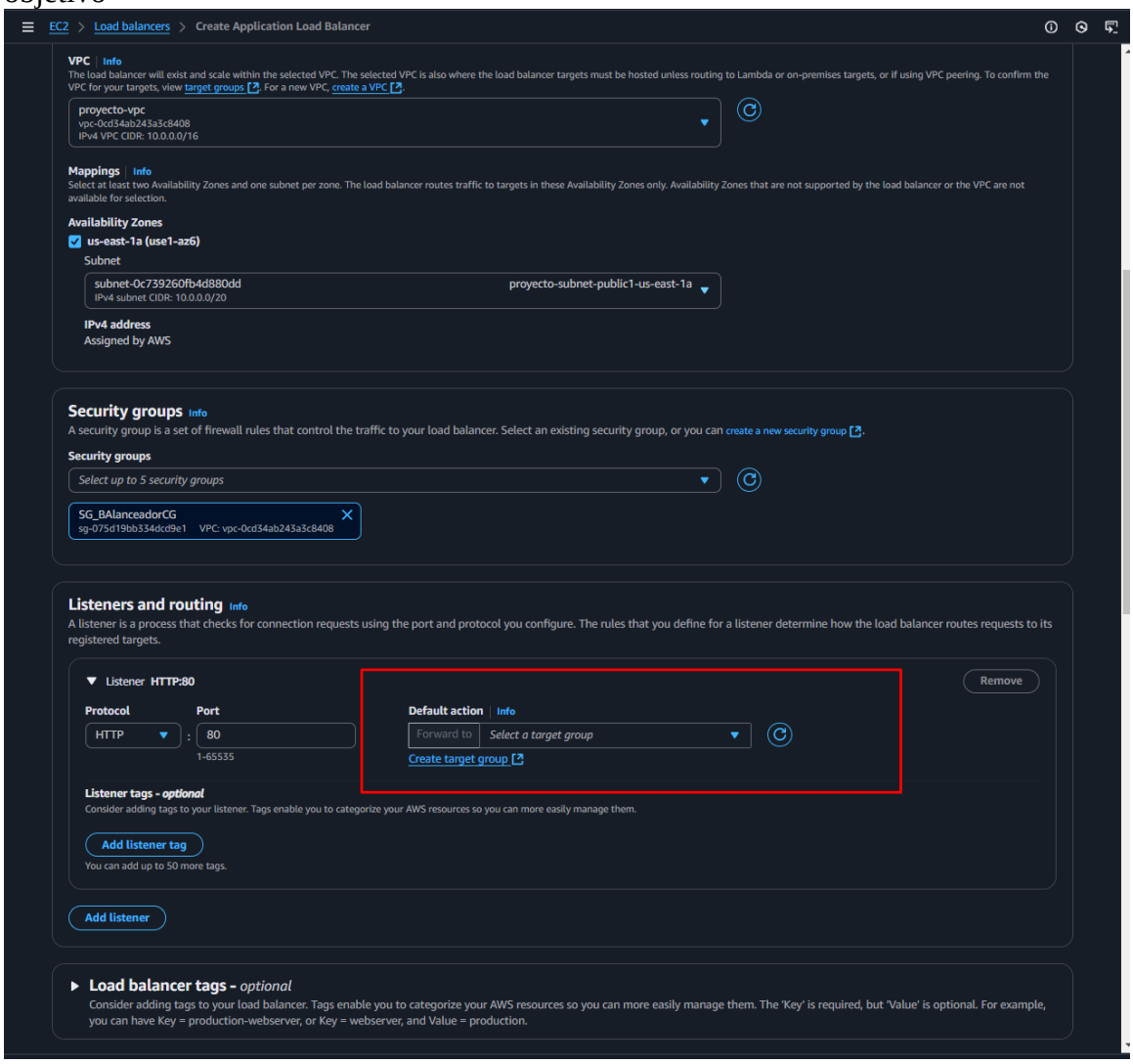


Figura 31. Configuramos el Grupo

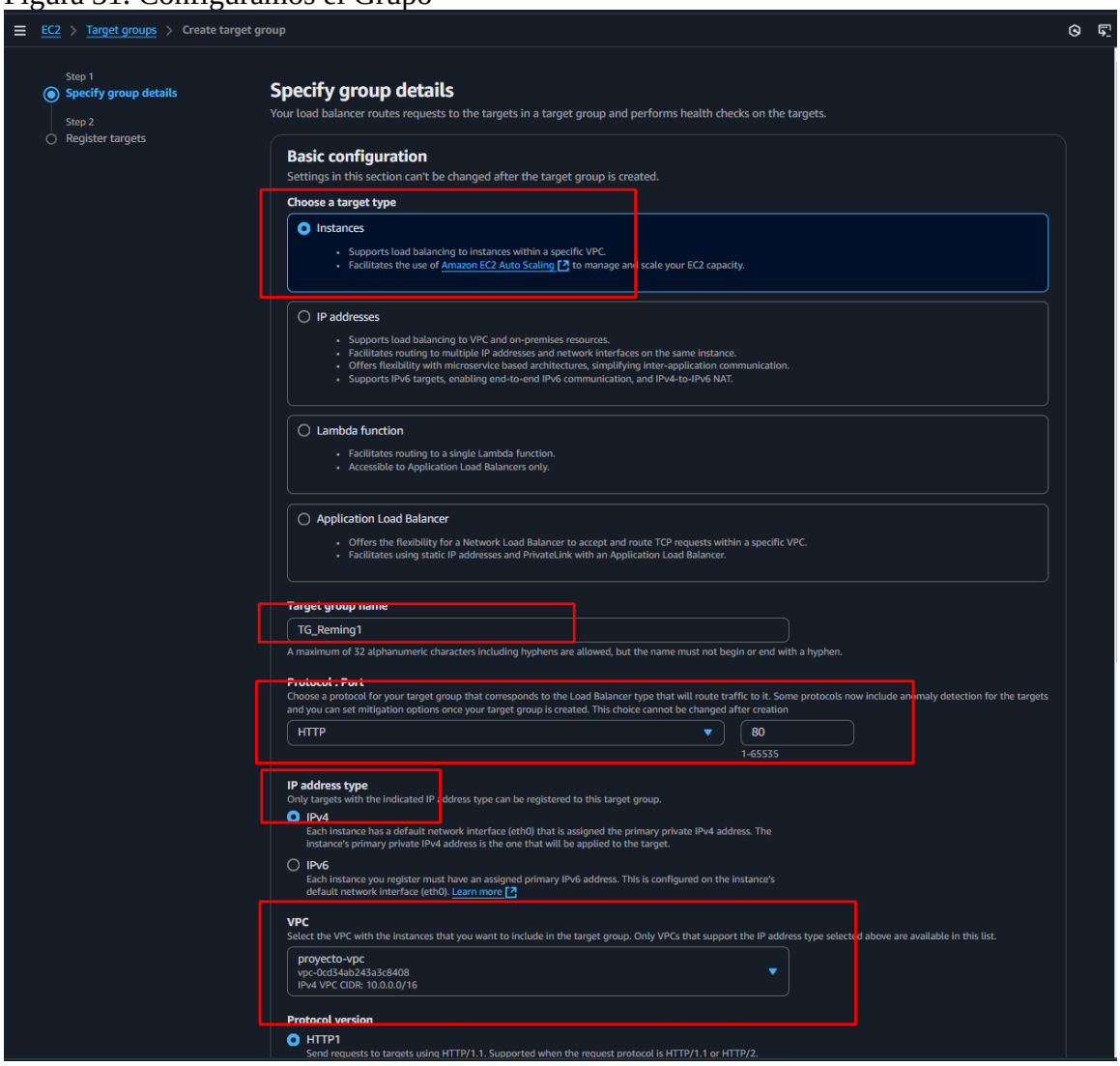


Figura 32. Seleccionar vpc y las rutas de diagnostico

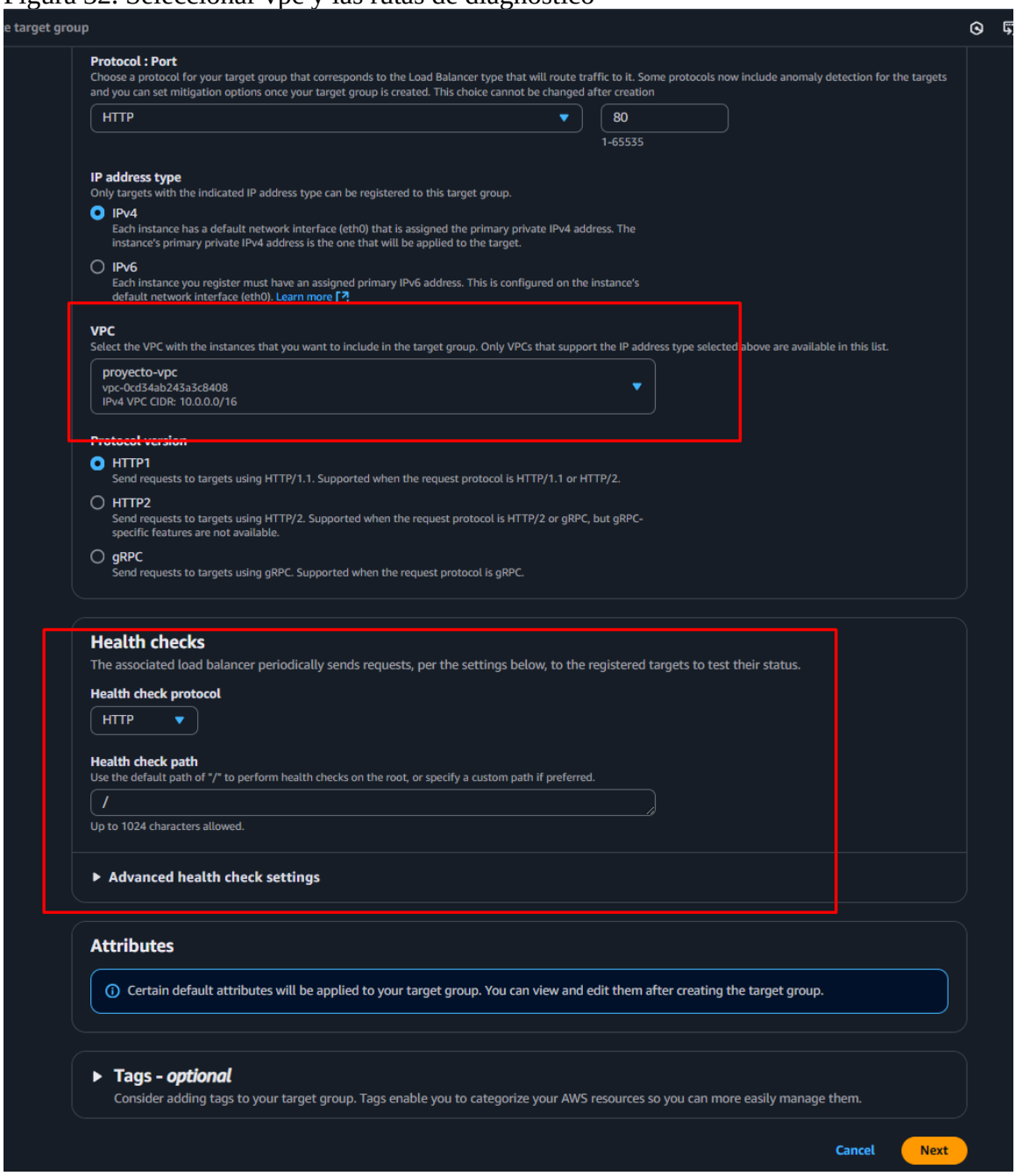
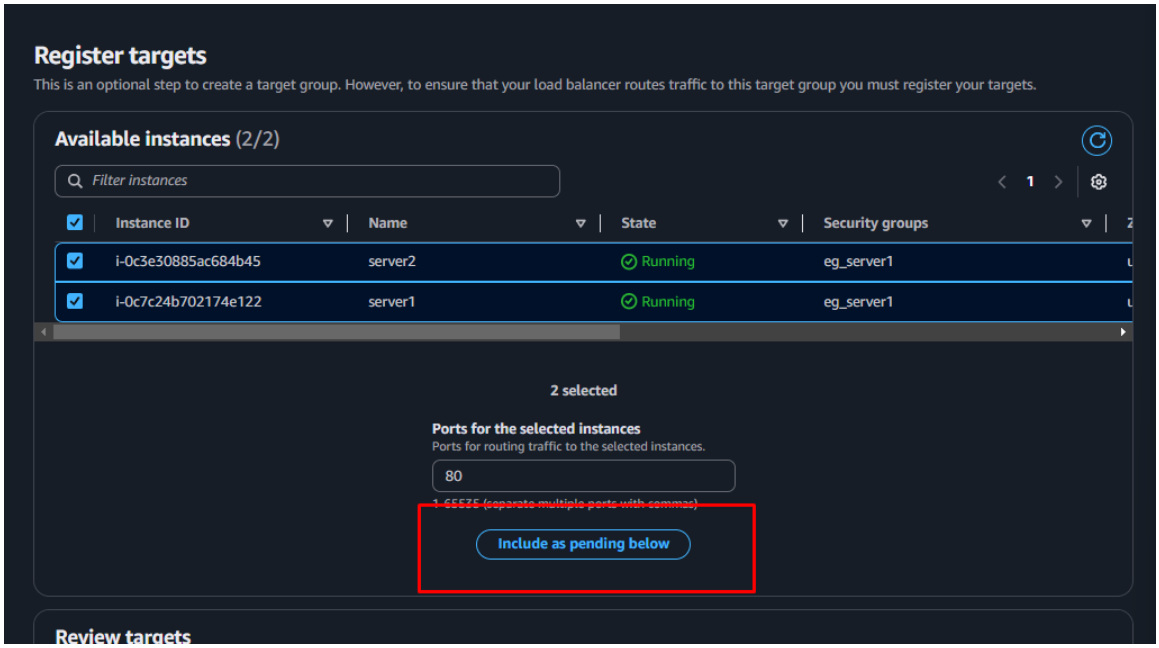


Figura 33. Seleccionamos las instancias



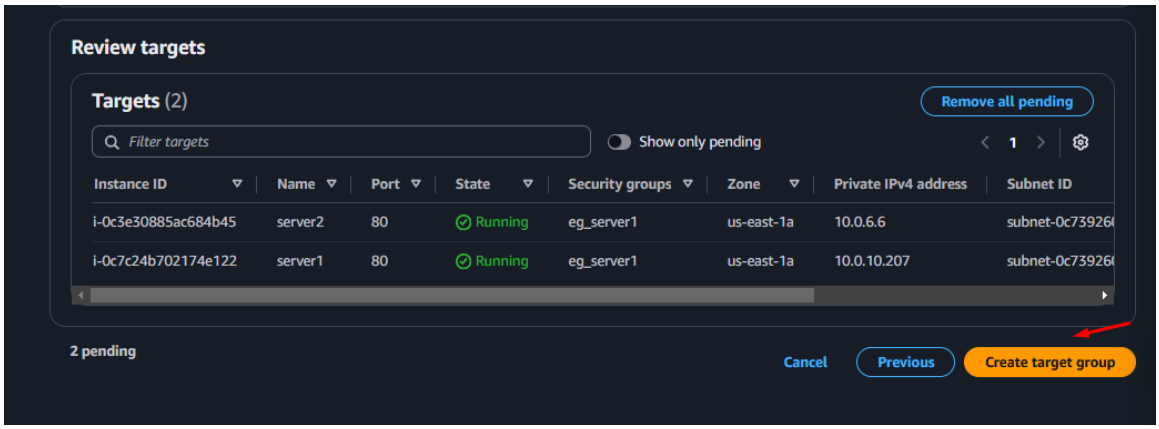


Figura 34. En este momento empezar a validar las instancias creadas y el Healthy debe ser igual a las cantidades de instancias

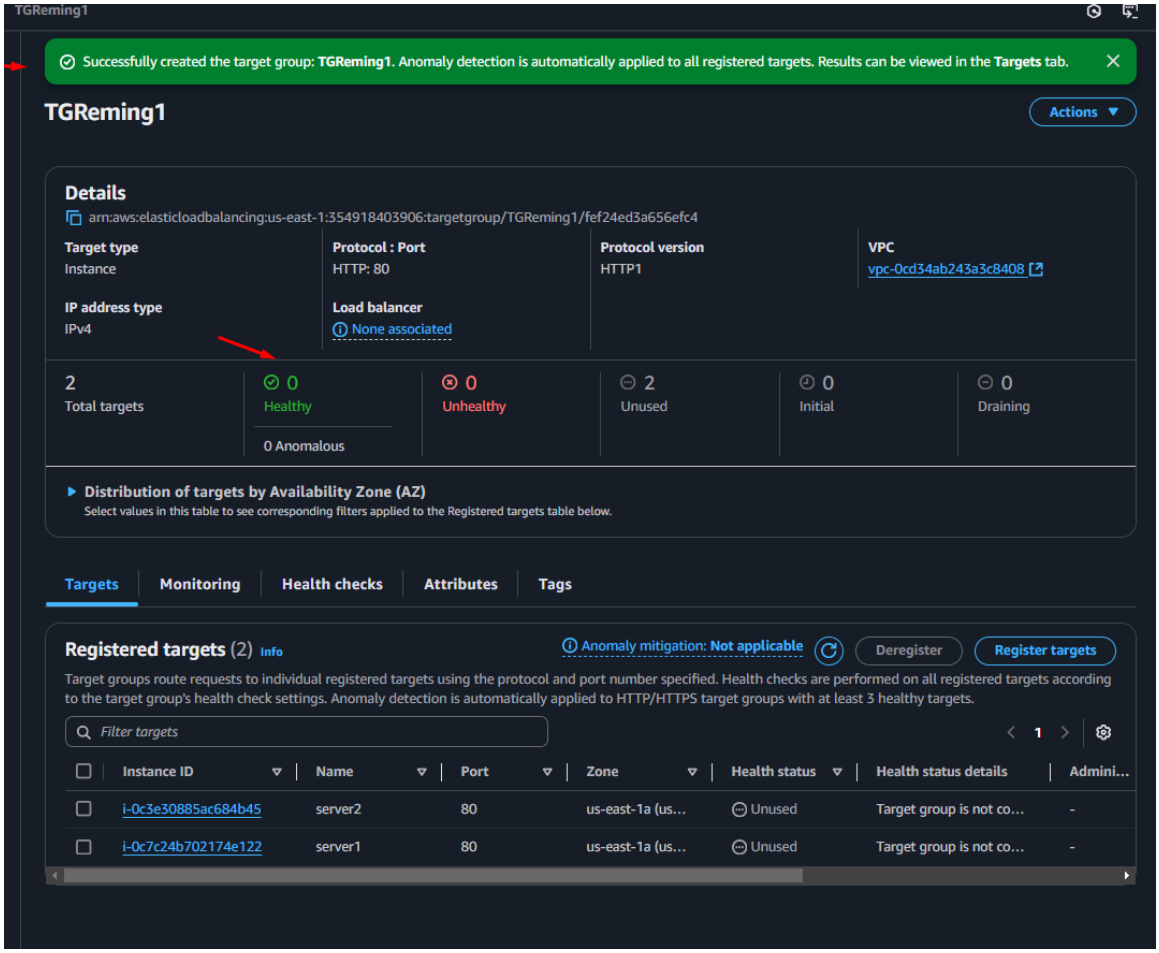


Figura 35.
Mientras carga el TG seguimos configurando el balanceador

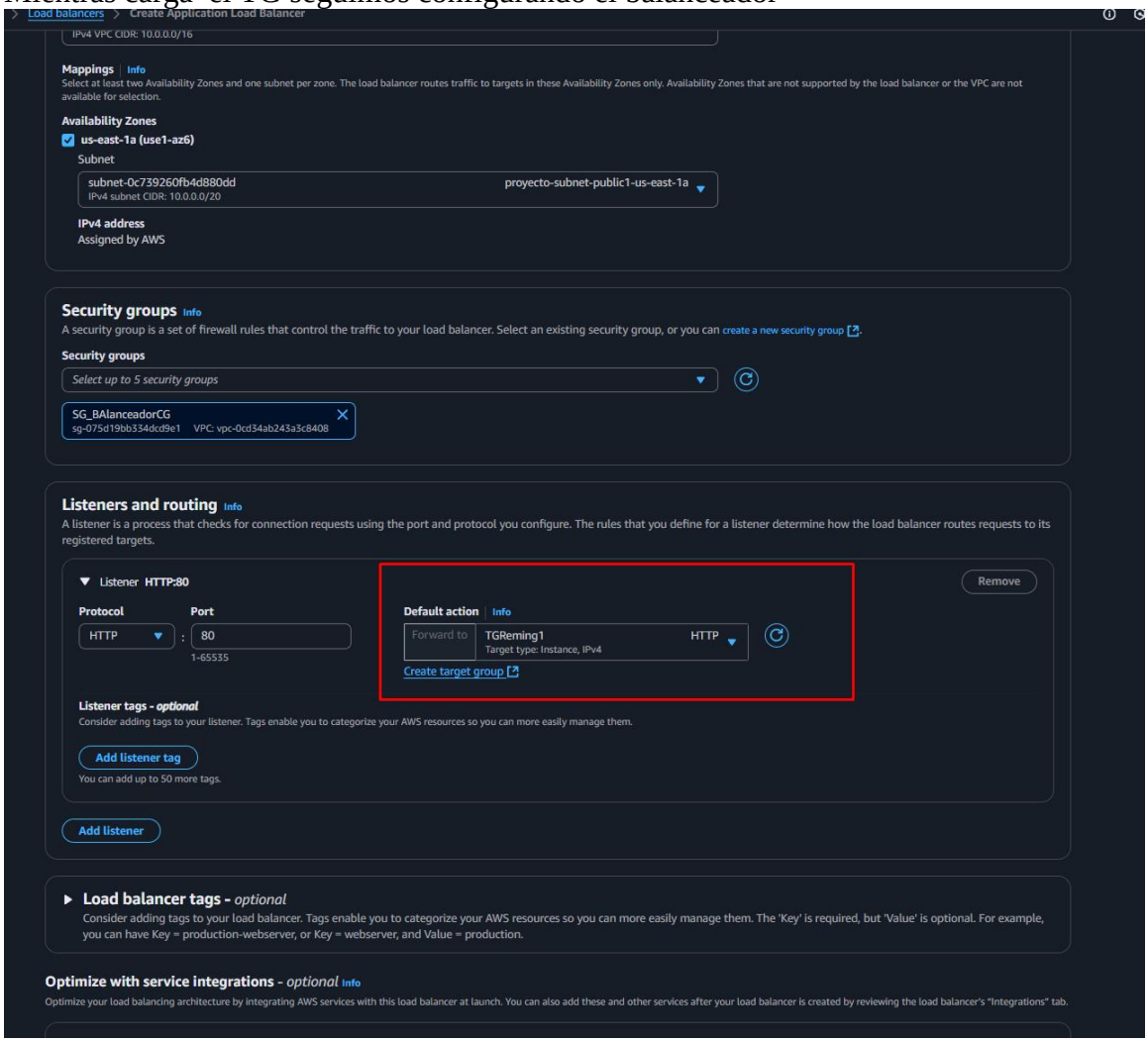


Figura 36. Importante de debe configurar la vpc para que de dos subredes publicas funcionales

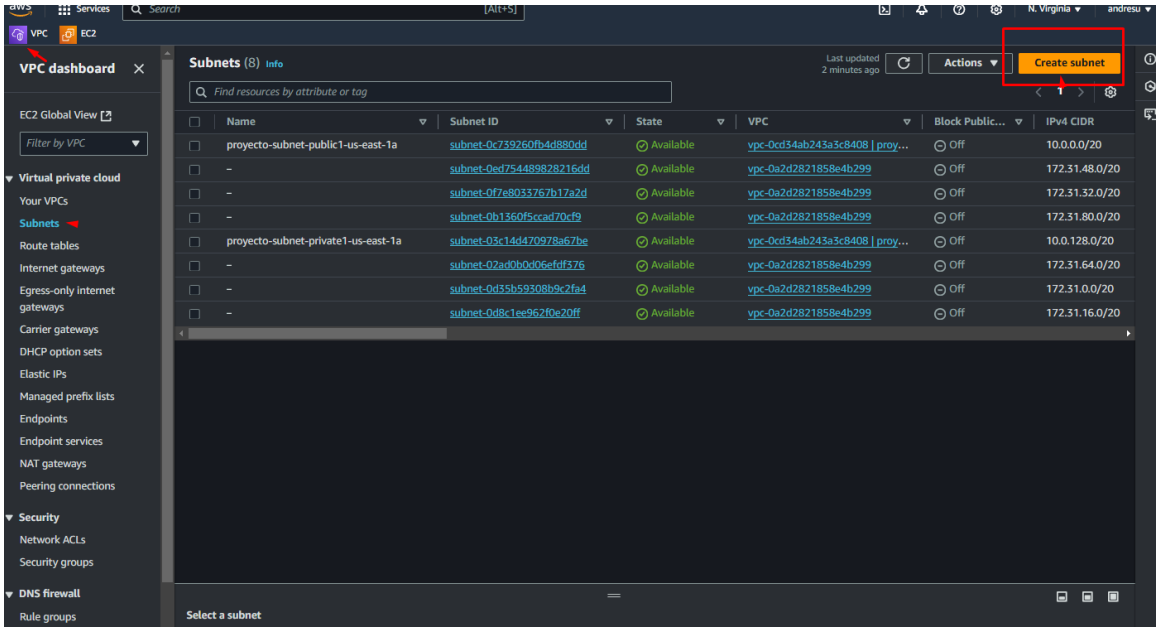


Figura 37. Seleccionamos las subredes en el balanceador

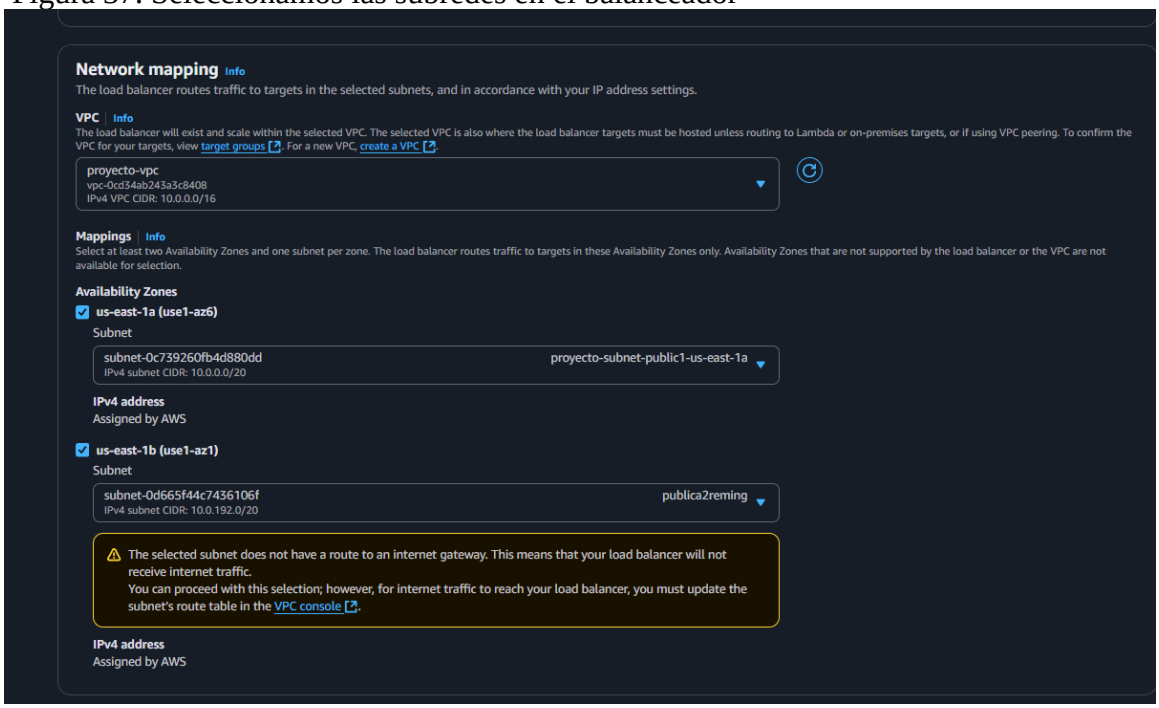


Figura 37. Configuración subredes

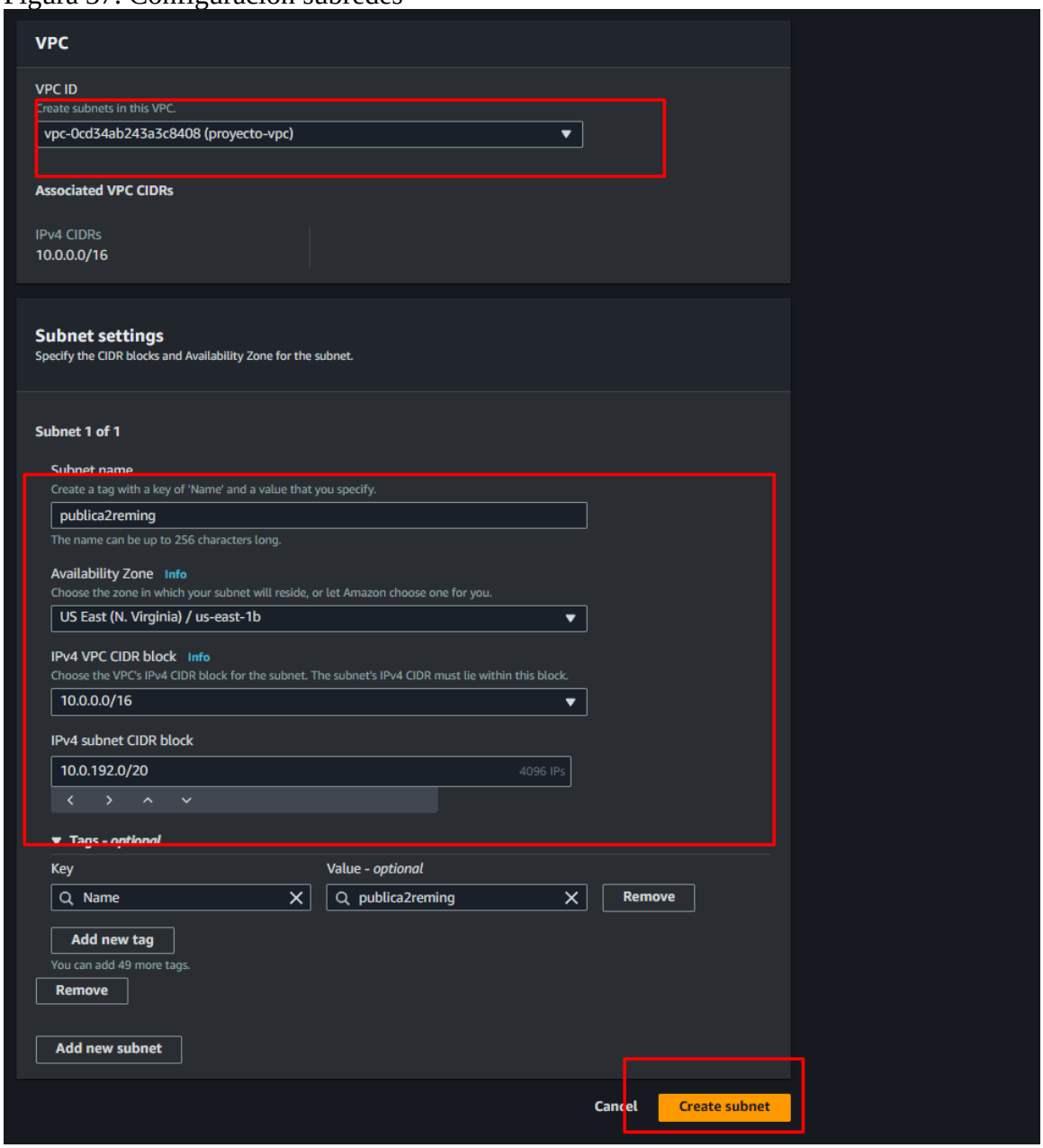


Figura 38. Configuramos el internet Gateway

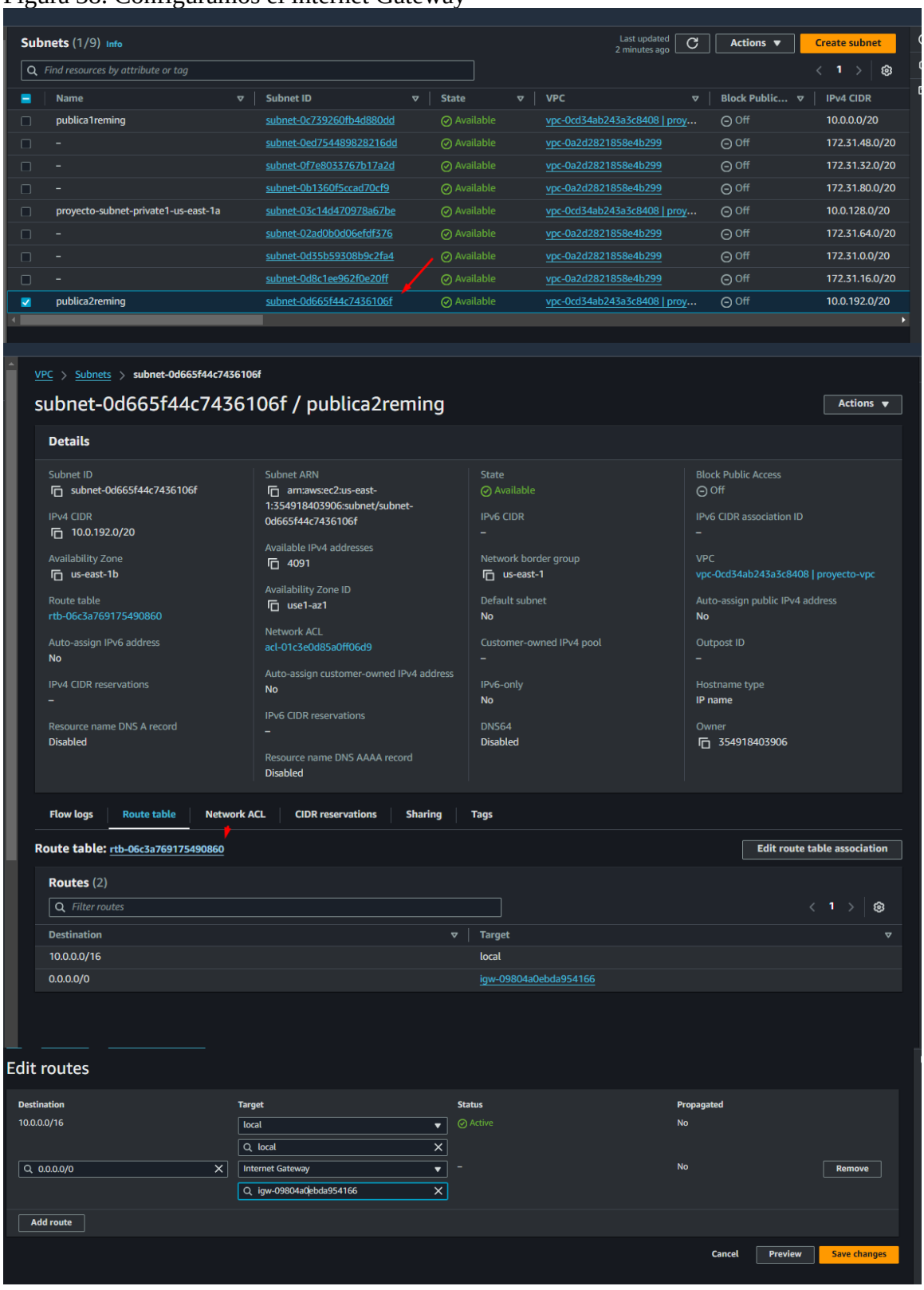


Figura 39. Continuamos con la creación del balanceador
Seleccionamos las suberedes

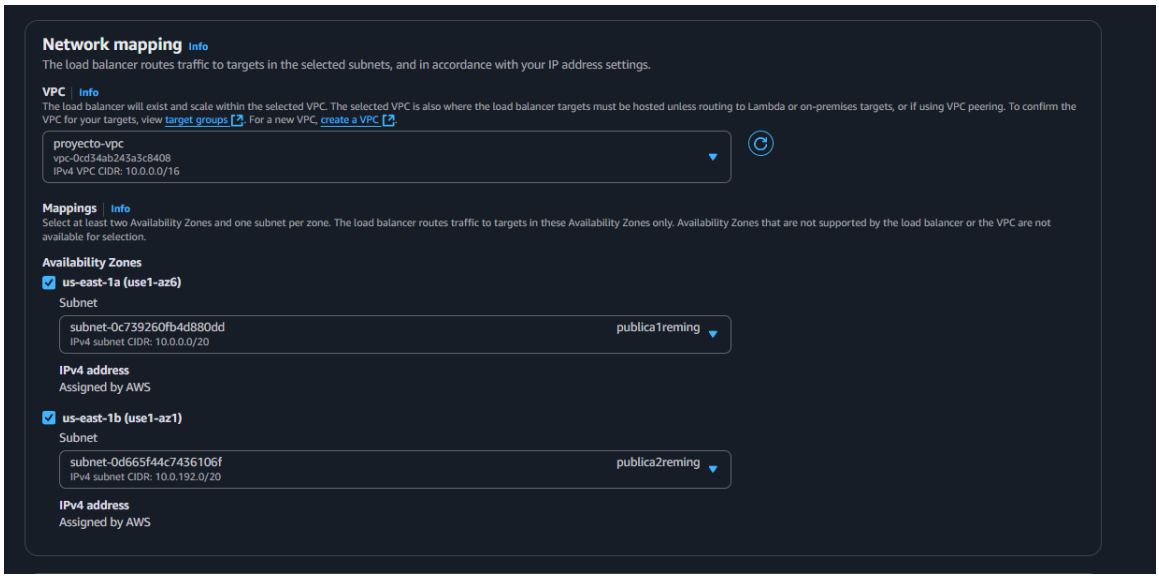


Figura 40. Bajamos y le damos crear

Figura 4. Se verifican los datos y la creación.

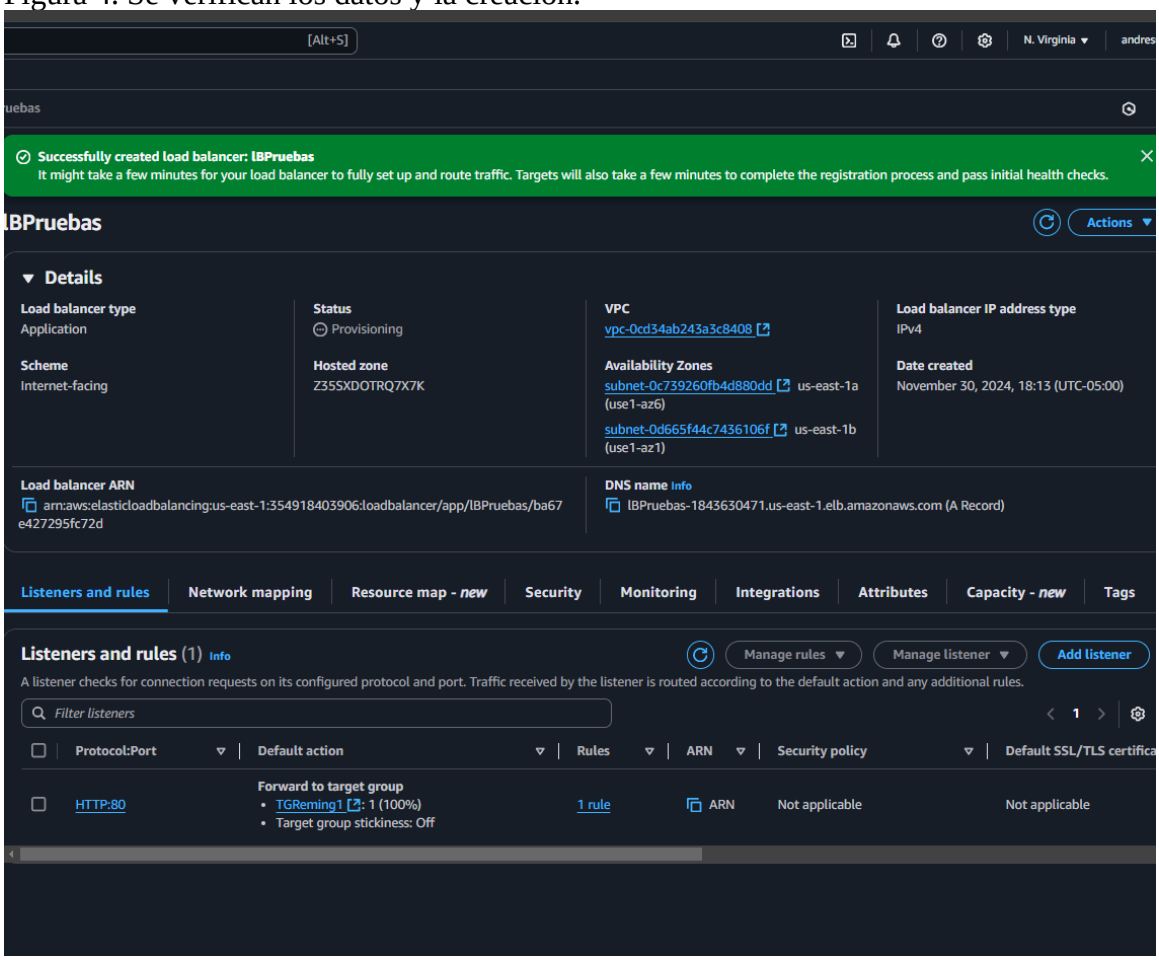


Figura 41. Revisamos el grupo de seguridad y seleccionamos el del balanceador

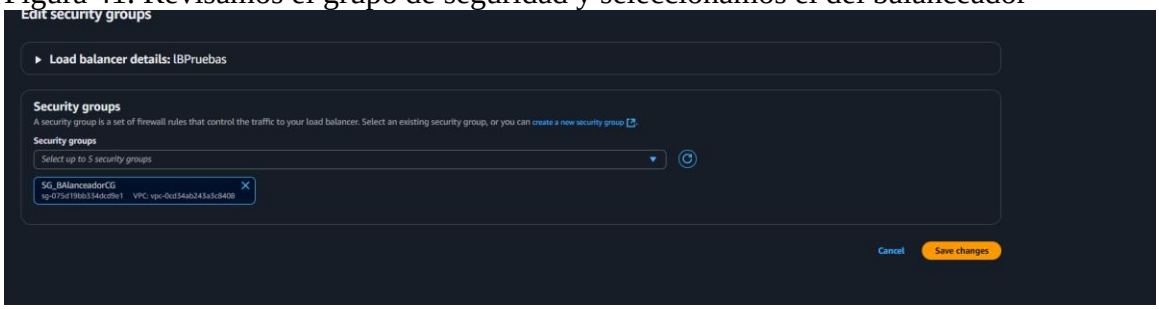
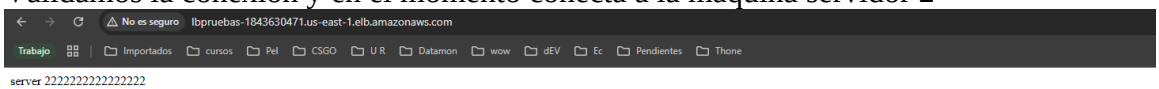


Figura 42.
Validamos la conexión y en el momento conecta a la maquina servidor 2



server 2222222222222222

Figura 43.
Ahora se automatiza el balanceador.

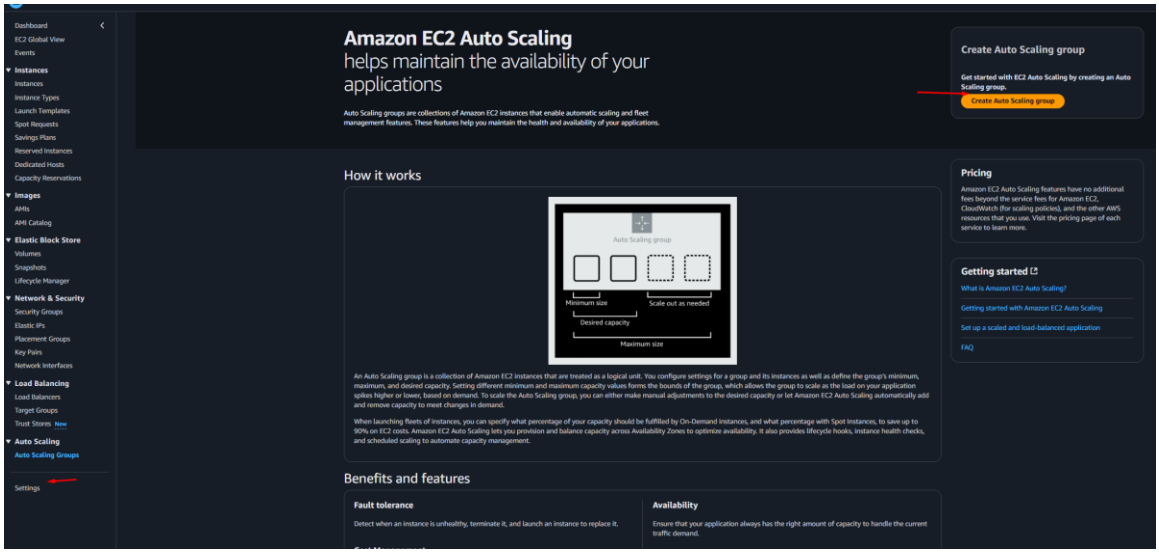


Figura 44.
Creamos el template

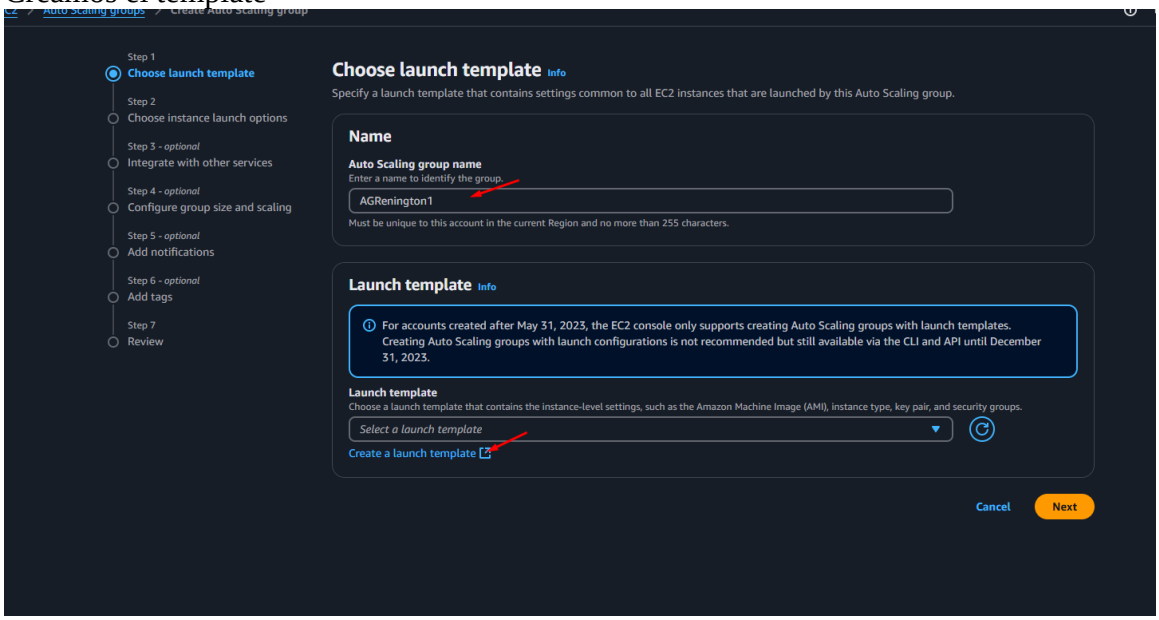


Figura 45 le damos un nombre y seleccionamos la ami creada en los primeros paso.

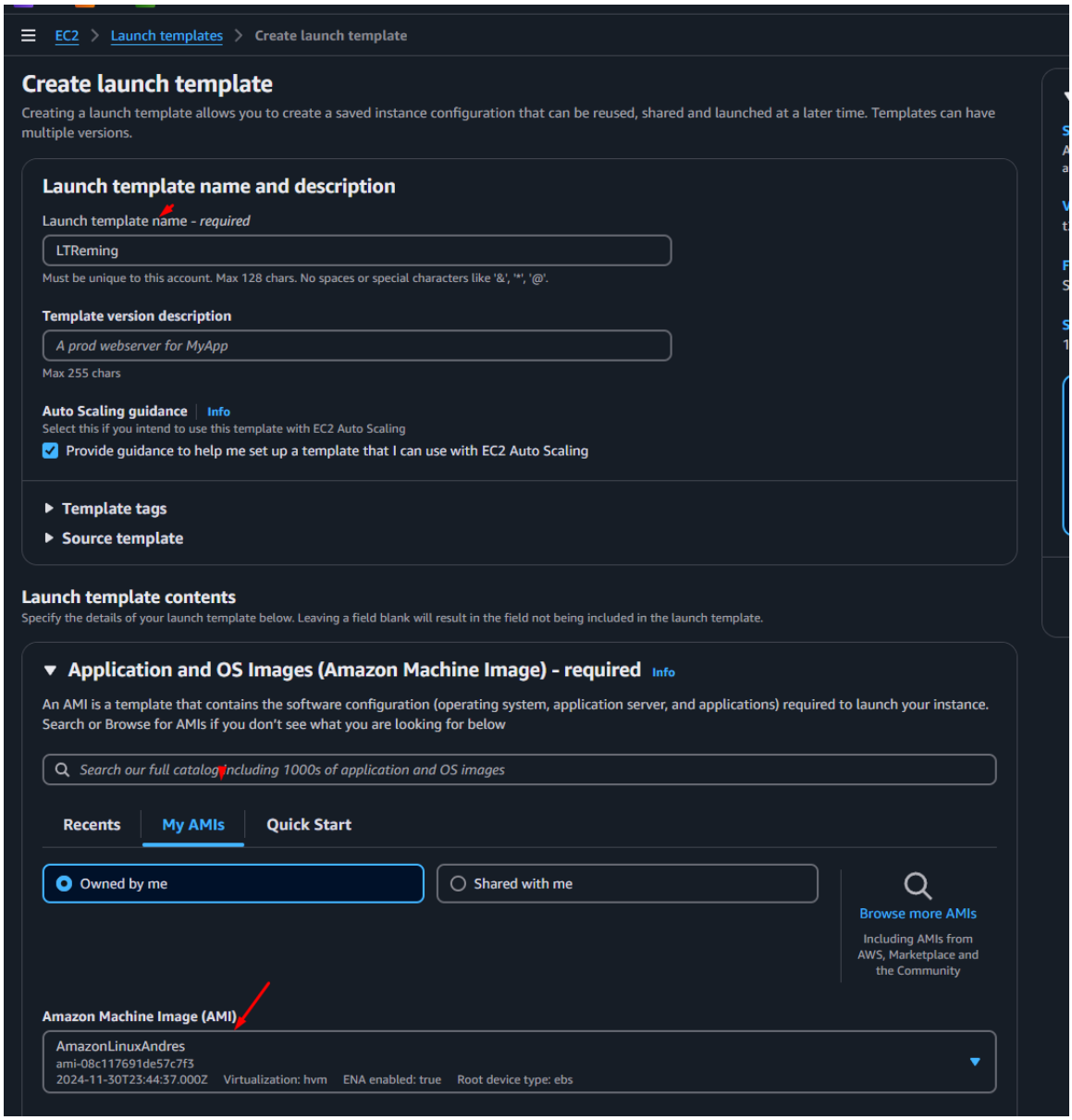


Figura 46. Seleccionamos el key, y grupo creados anteriormente y de damos crear

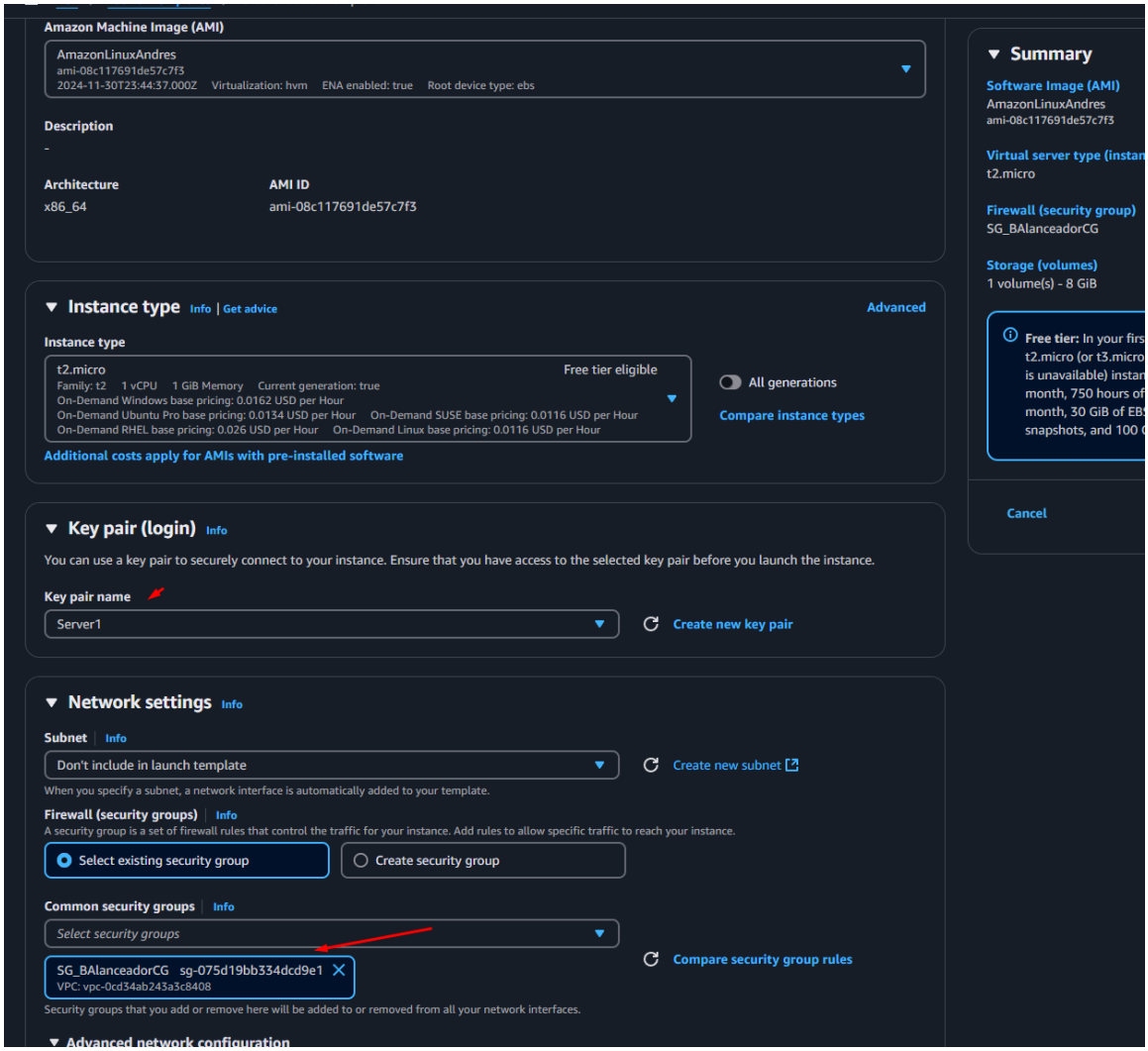


Figura 47. procedemos con la configuración

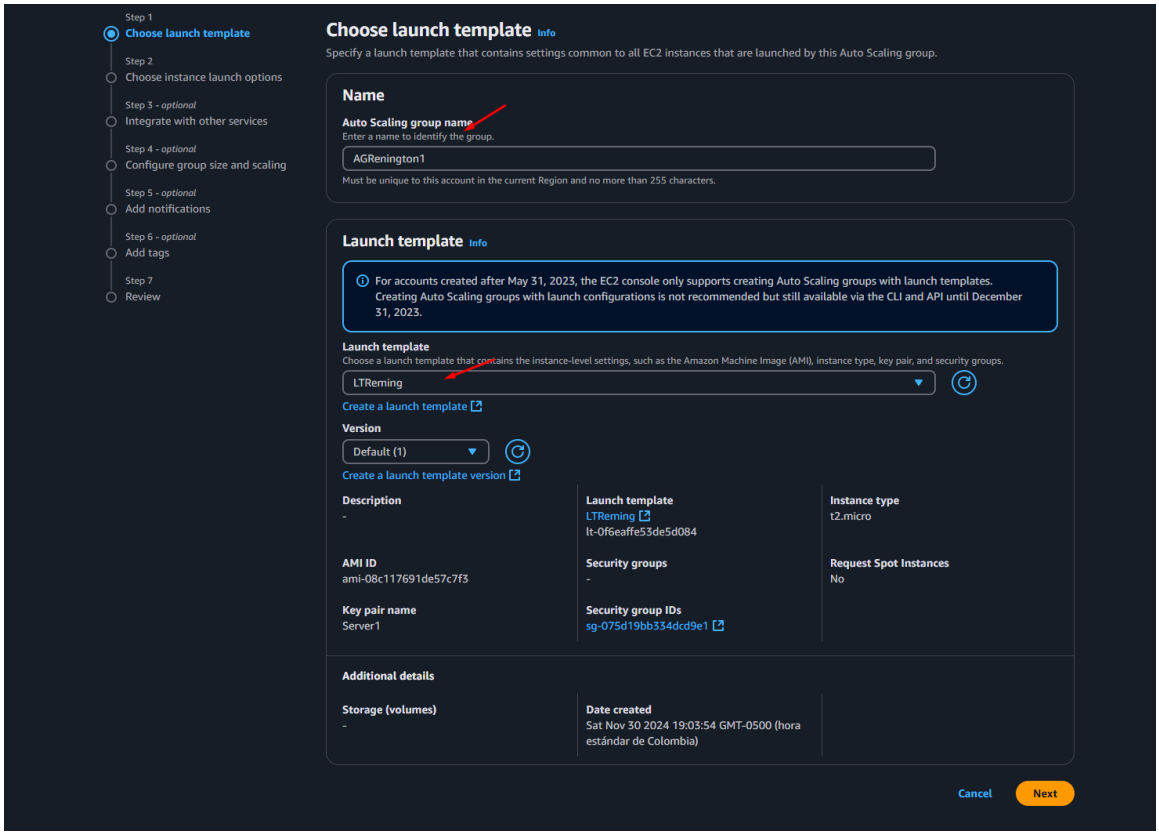


Figura 48. Le damos siguiente

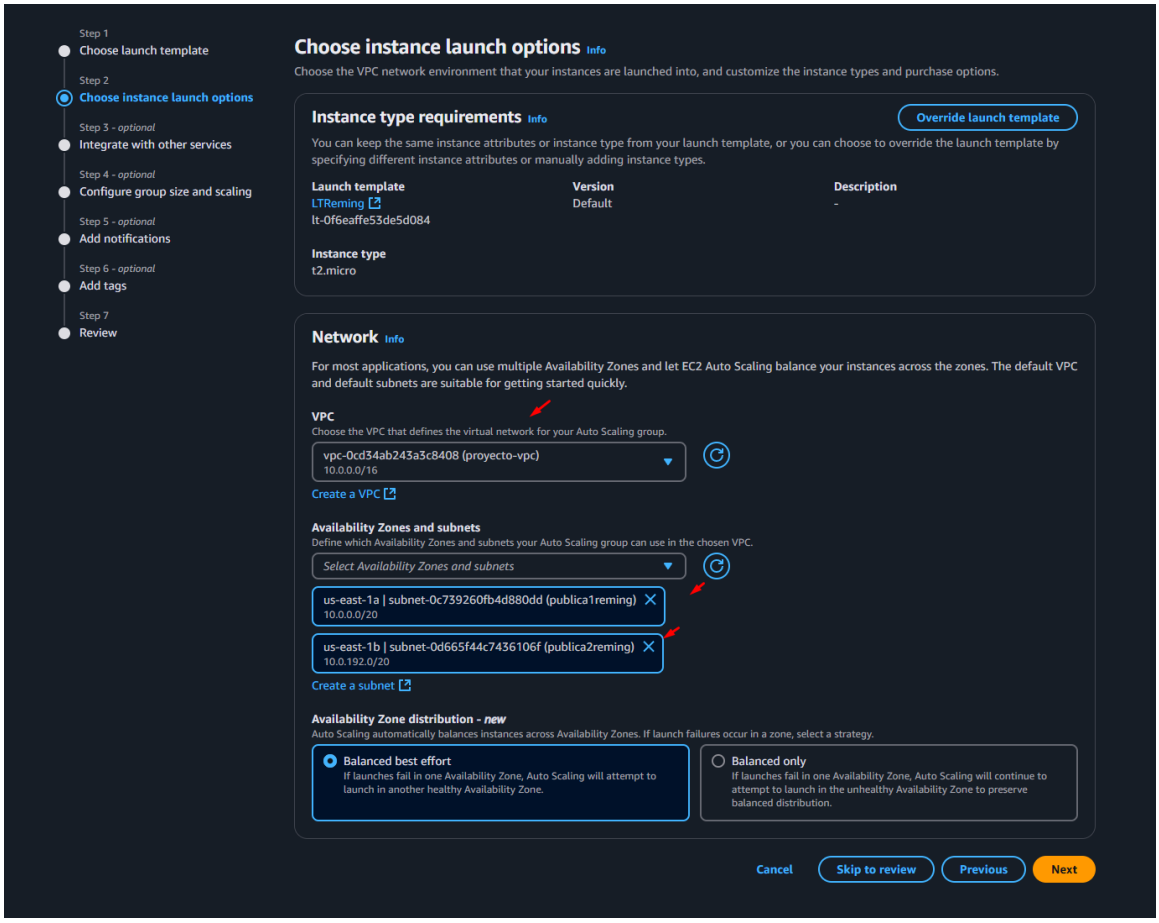

Step 1
Choose launch template
Step 2
Choose instance launch options
Step 3 - optional
Integrate with other services
Step 4 - optional
Configure group size and scaling
Step 5 - optional
Add notifications
Step 6 - optional
Add tags
Step 7
Review
Choose instance launch options Info
Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.
Instance type requirements Info
Override launch template
You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.
Launch template
LTReming
lt-0f6eaffe53de5d084
Version
Default
Description
-
Instance type
t2.micro
Network Info
For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.
VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.
vpc-0cd34ab243a3c8408 (proyecto-vpc)
10.0.0.0/16
Create a VPC
Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.
Select Availability Zones and subnets
us-east-1a | subnet-0c739260fb4d880dd (publica1reming)
10.0.0.0/20
us-east-1b | subnet-0d665f44c7436106f (publica2reming)
10.0.192.0/20
Create a subnet
Availability Zone distribution - new
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.
Balanced best effort
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.
Balanced only
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.
Cancel
Skip to review
Previous
Next

Figura 49.
Seleccionamos el que realice el balanceador a maquinas existentes por ejemplo cuando se elimine una maquina.

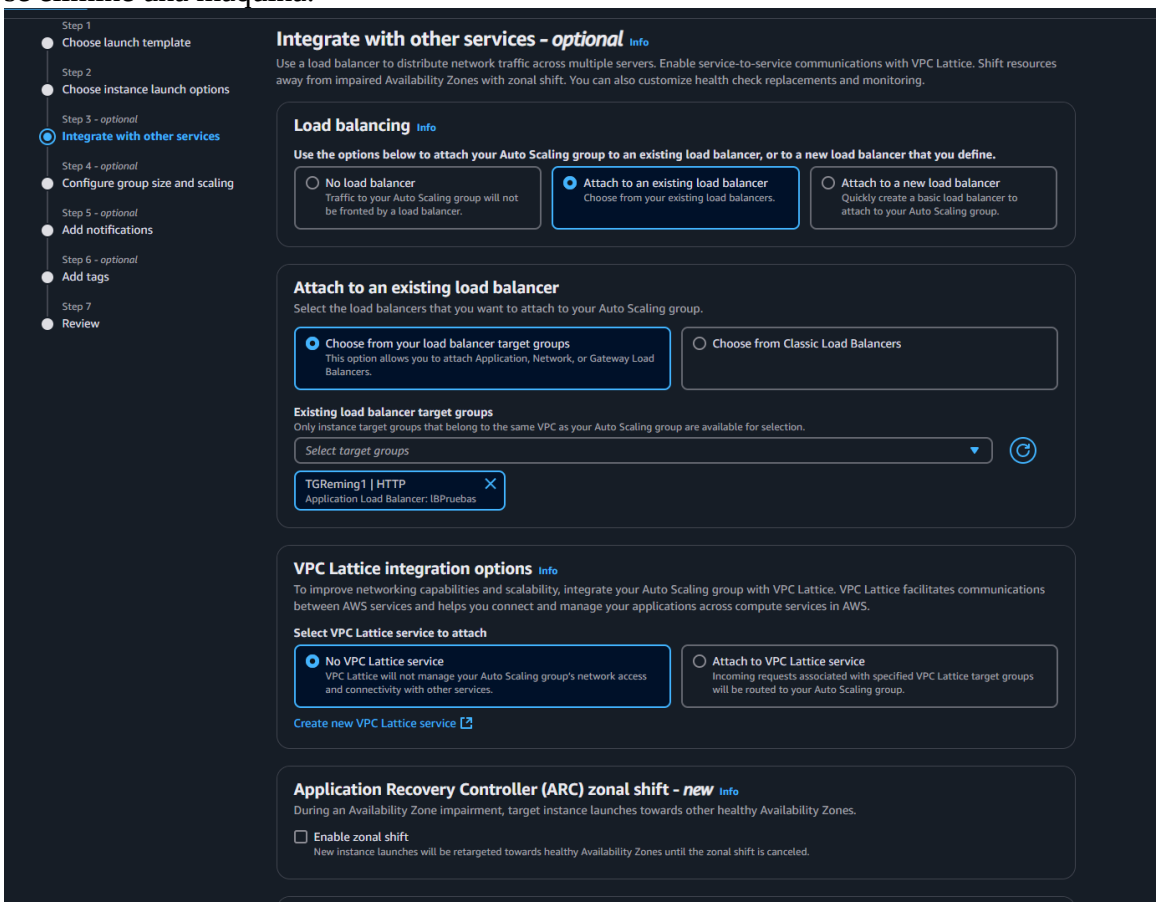


Figura 50.
Le indicamos que cree dos maquinas

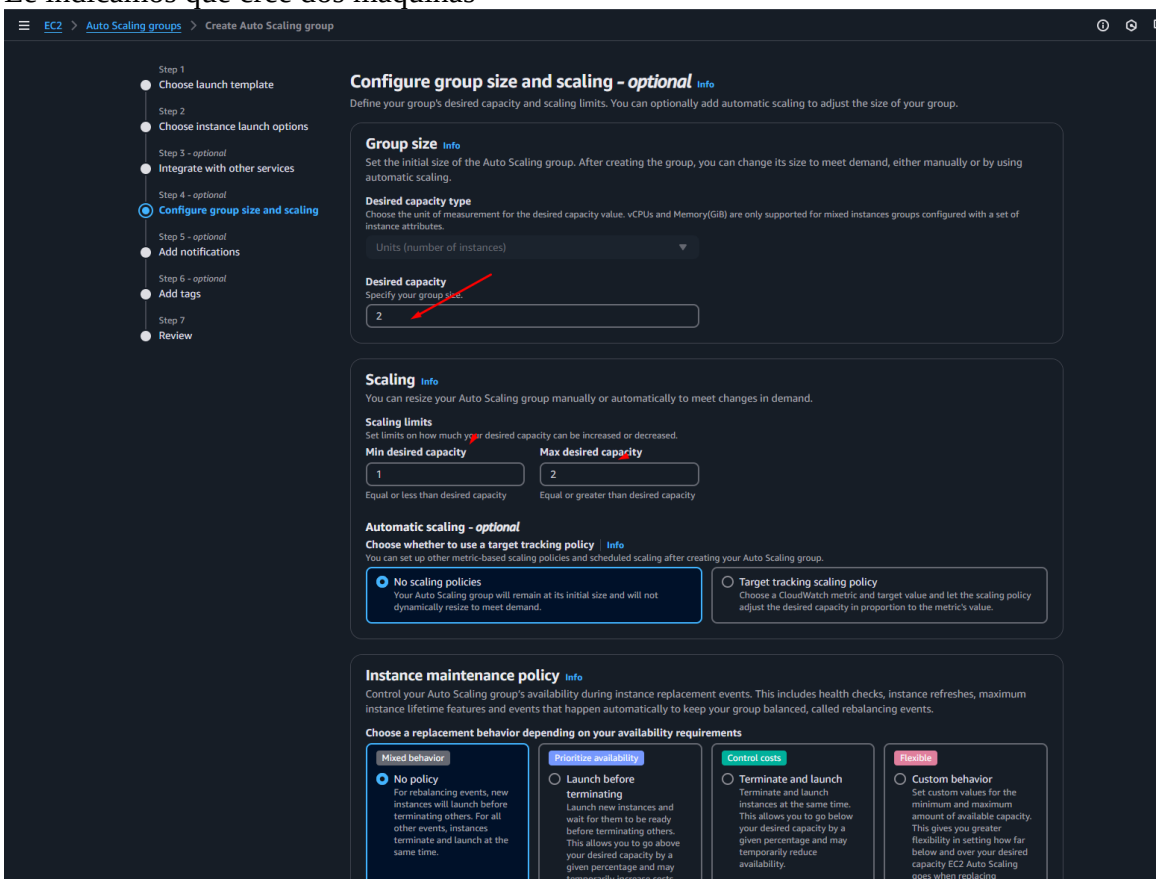


Figura 51.
Le damos siguiente

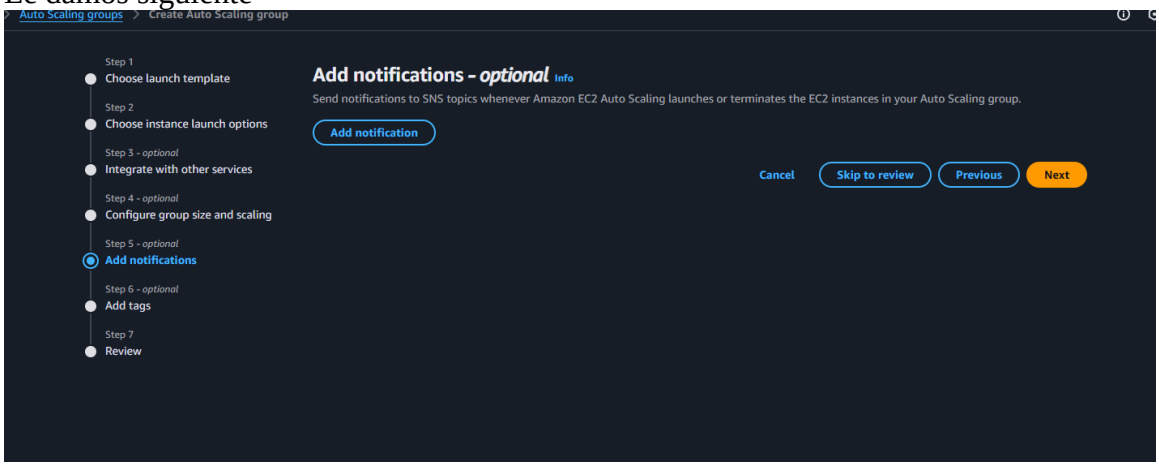


Figura 52.
Siguiente

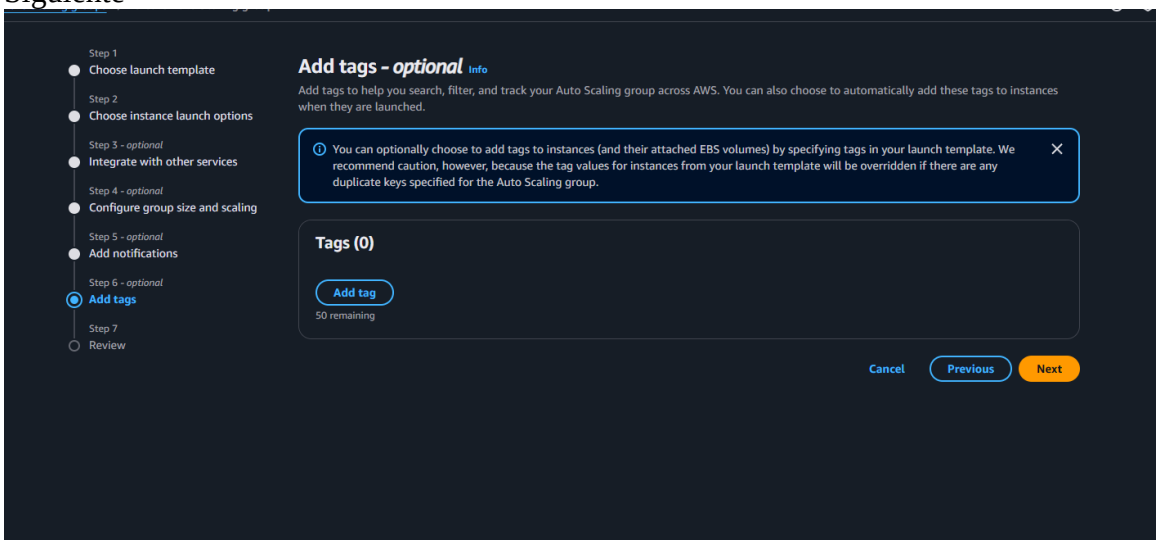


Figura 53.Y crear

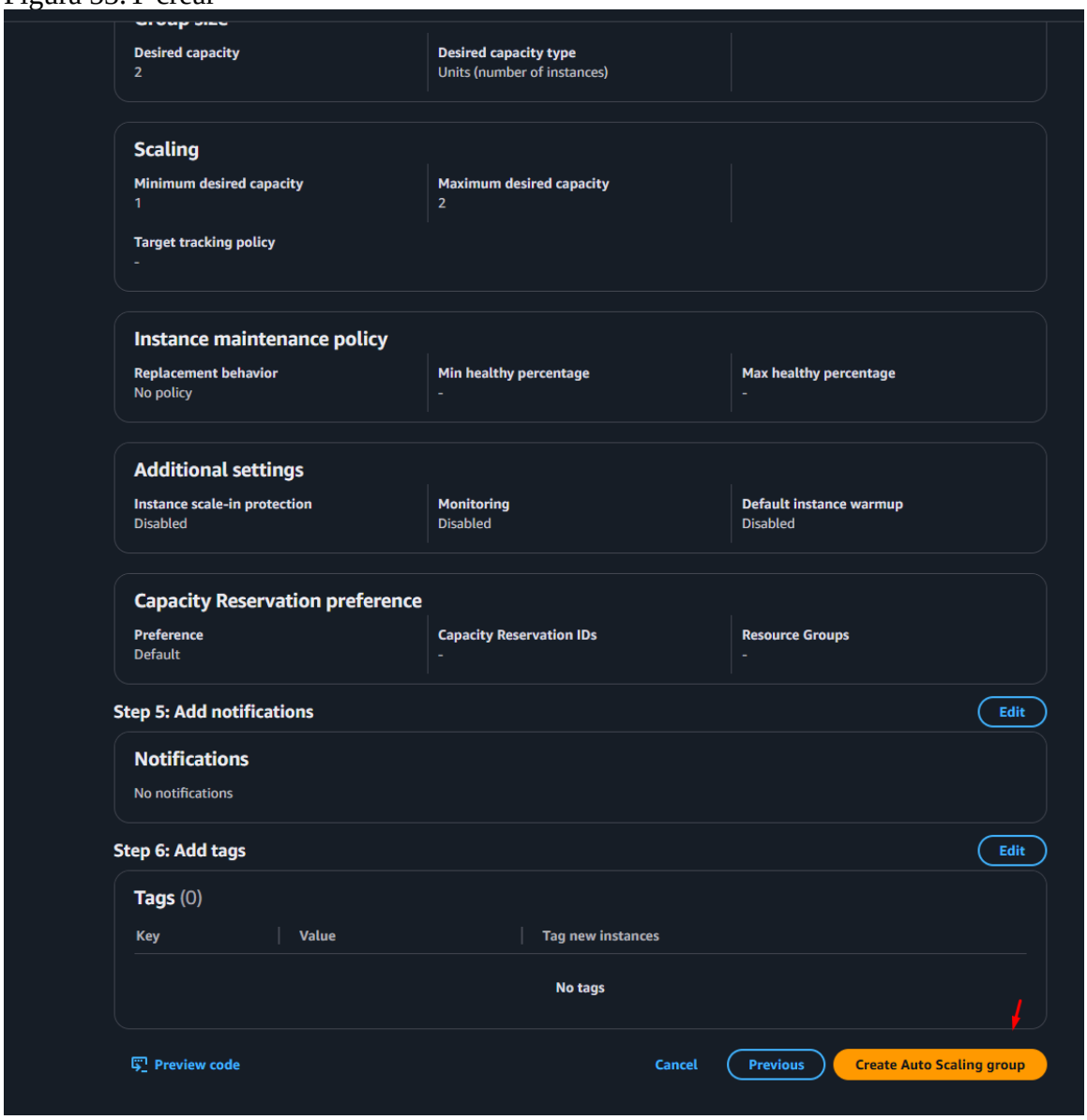


Figura 54. Ingresamos a revisar el AG

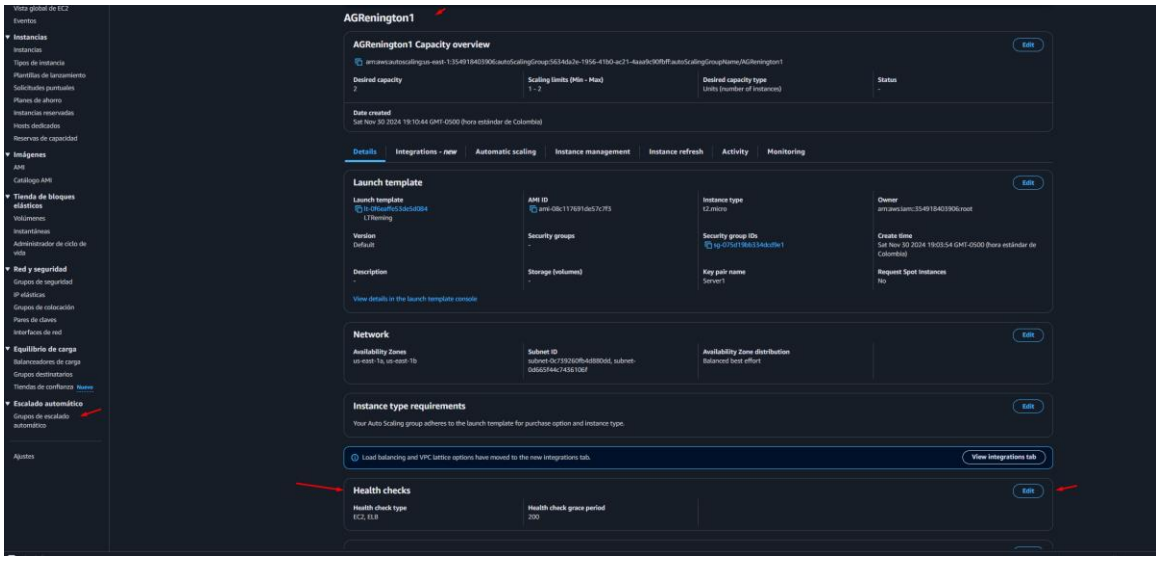
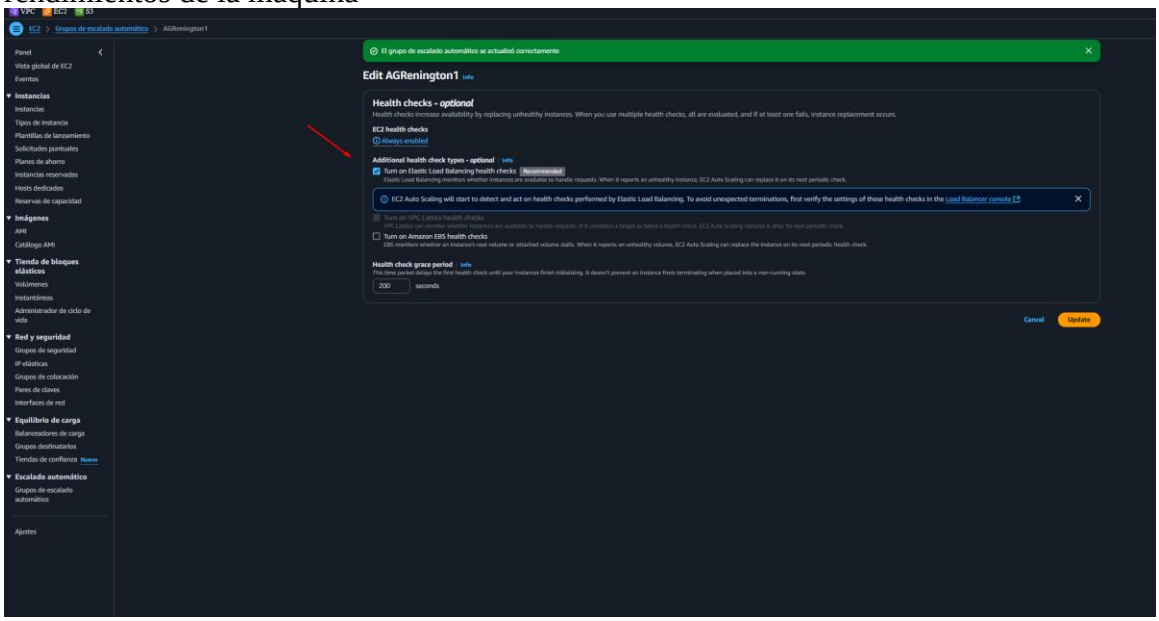


Figura 55.

Para que suba cuando detecte un servicio mal como la detención de la web y mal rendimientos de la maquina



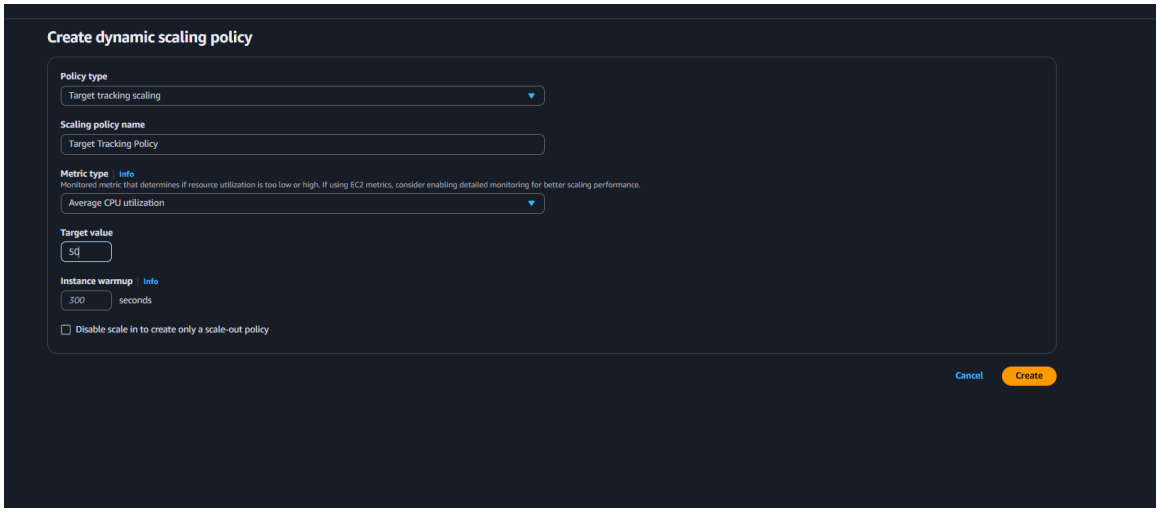

Create dynamic scaling policy
Policy type
Target tracking scaling
Scaling policy name
Target Tracking Policy
Metric type | Info
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.
Average CPU utilization
Target value
50
Instance warmup | Info
300
seconds
Disable scale in to create only a scale-out policy
Cancel
Create

Figura 56.
Luego vemos que en las instancias se crearon las dos que le indicamos en este caso 2 para un total de 4

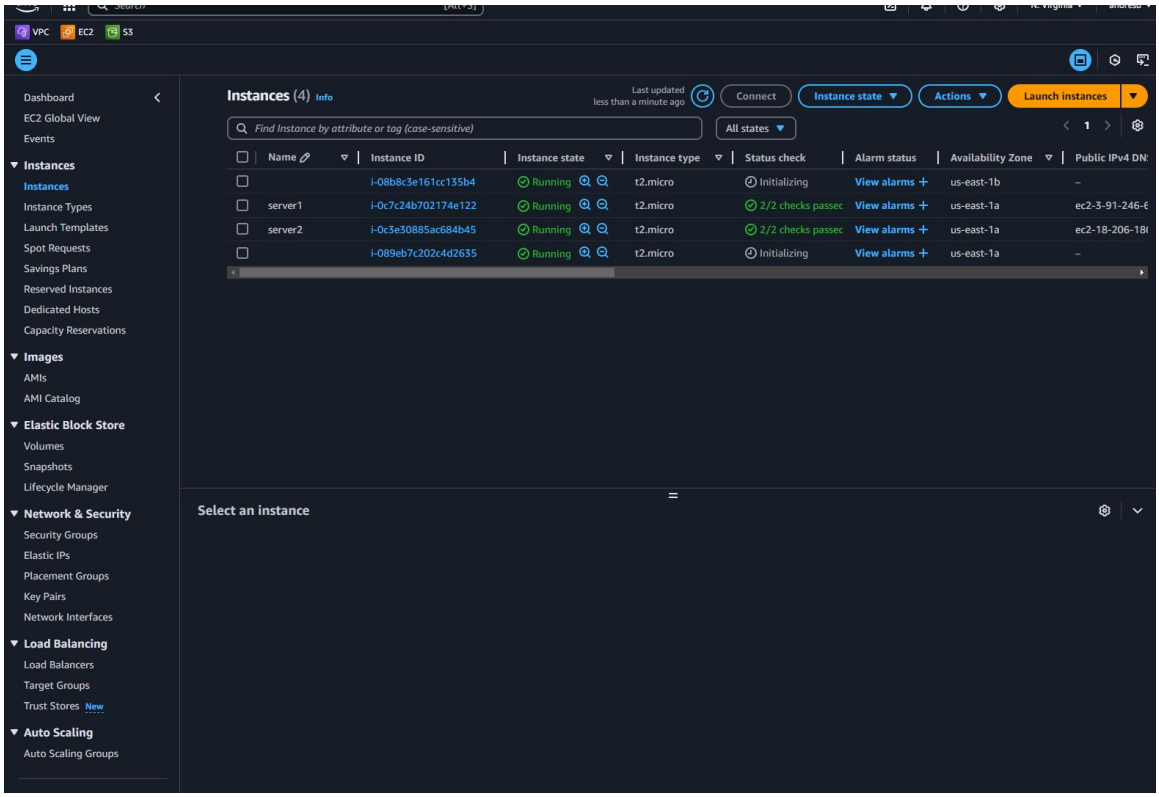


Figura 57.
En el balanceador de carga vemos las 4

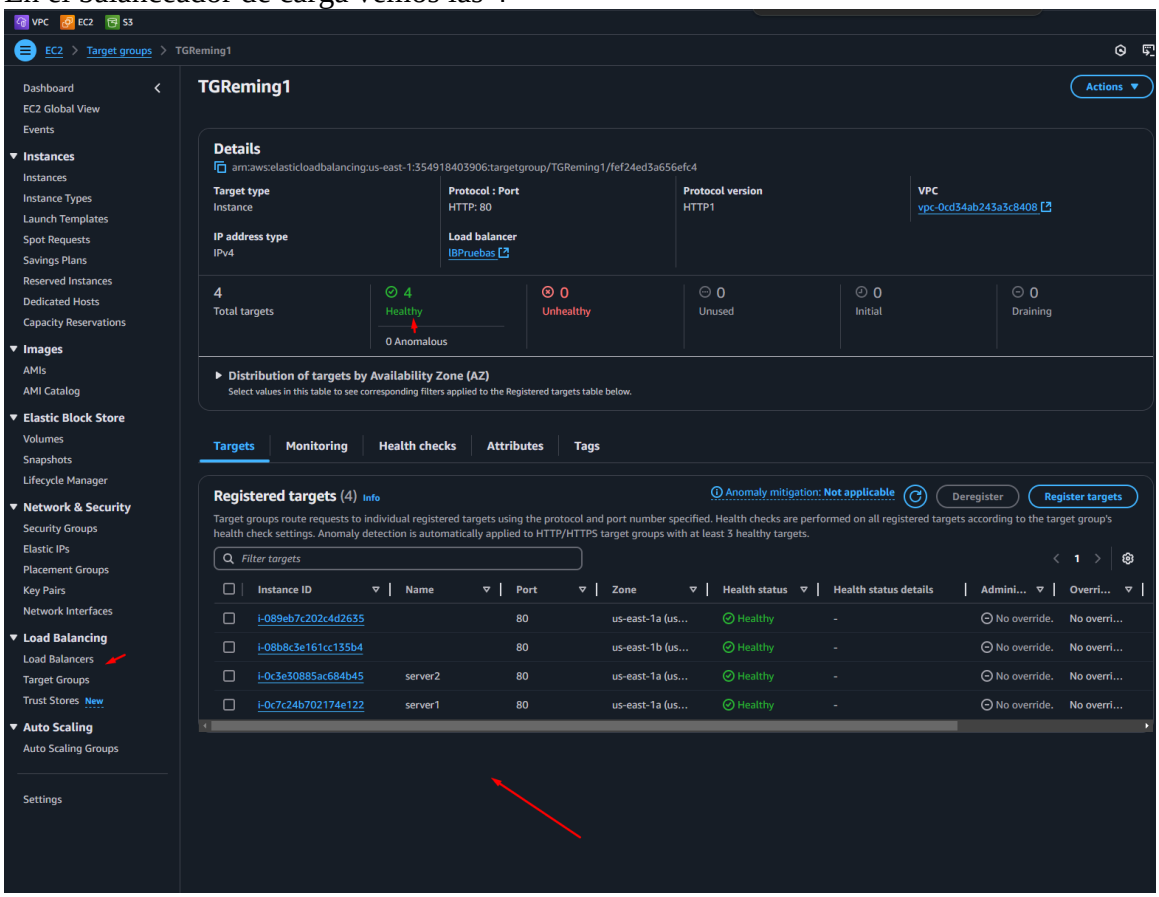


Figura 58.
Ahora podemos eliminar las instancias creadas inicialmente por que ya tenemos el balanceador de carga ok

Figura 59.
Creamos una maquina virtual para conectar a las instancias de balanceador

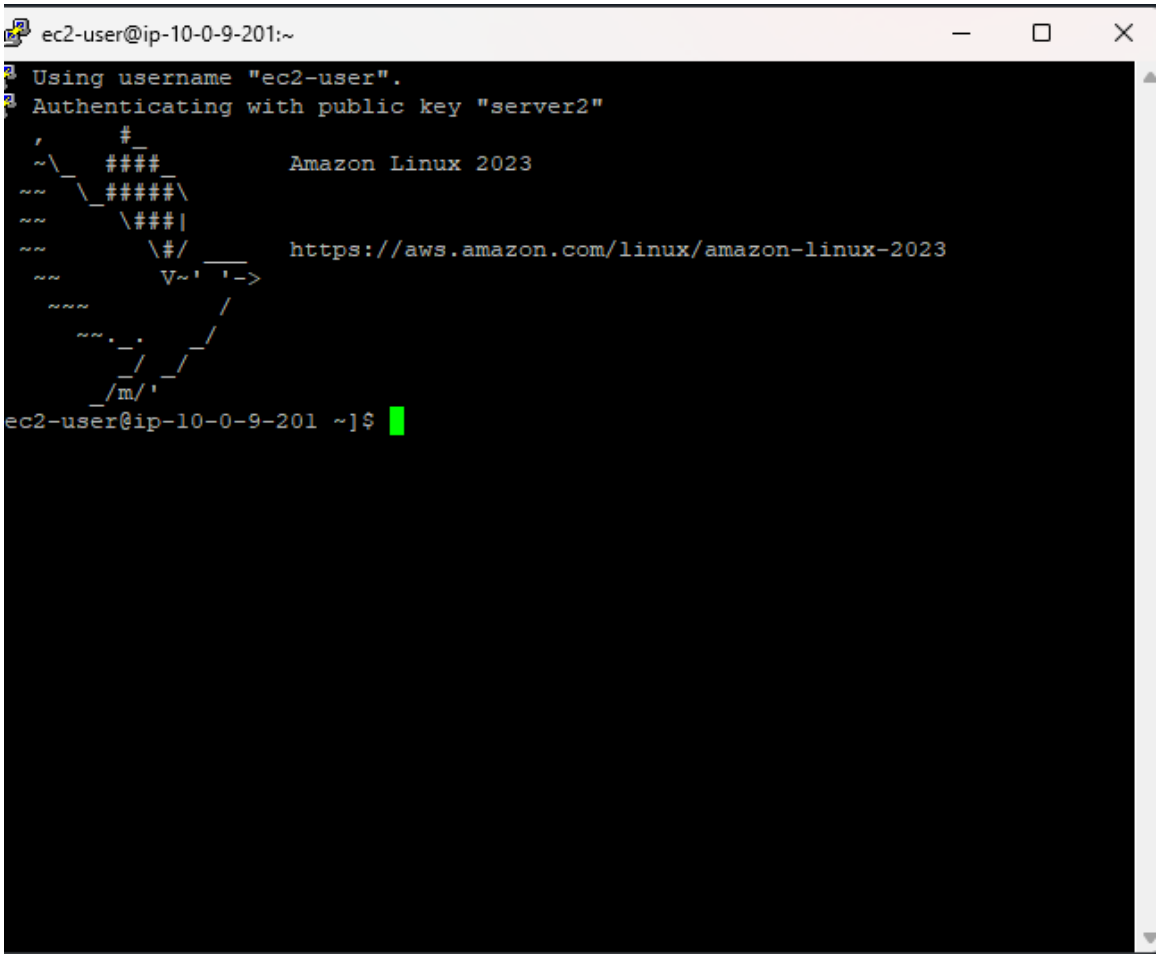

ec2-user@ip-10-0-9-201:~
Using username "ec2-user".
Authenticating with public key "server2"
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
ec2-user@ip-10-0-9-201 ~]$

Figura 60.
Desde la instancia creada conectamos a las instancias de los balanceadores para ello pasmaos el certificado a la maquina y le damos permisos y conectamos por ssh

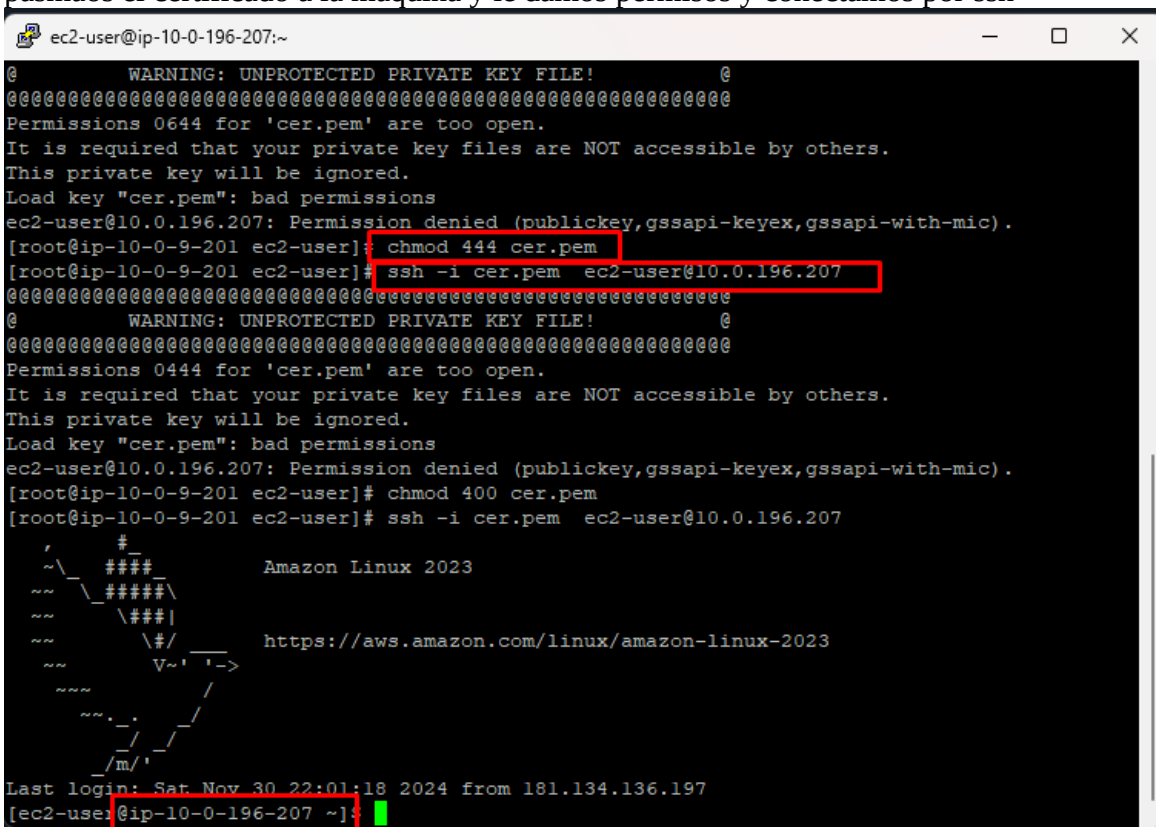


Figura 61. Revisamos los servicios web y los detenemos

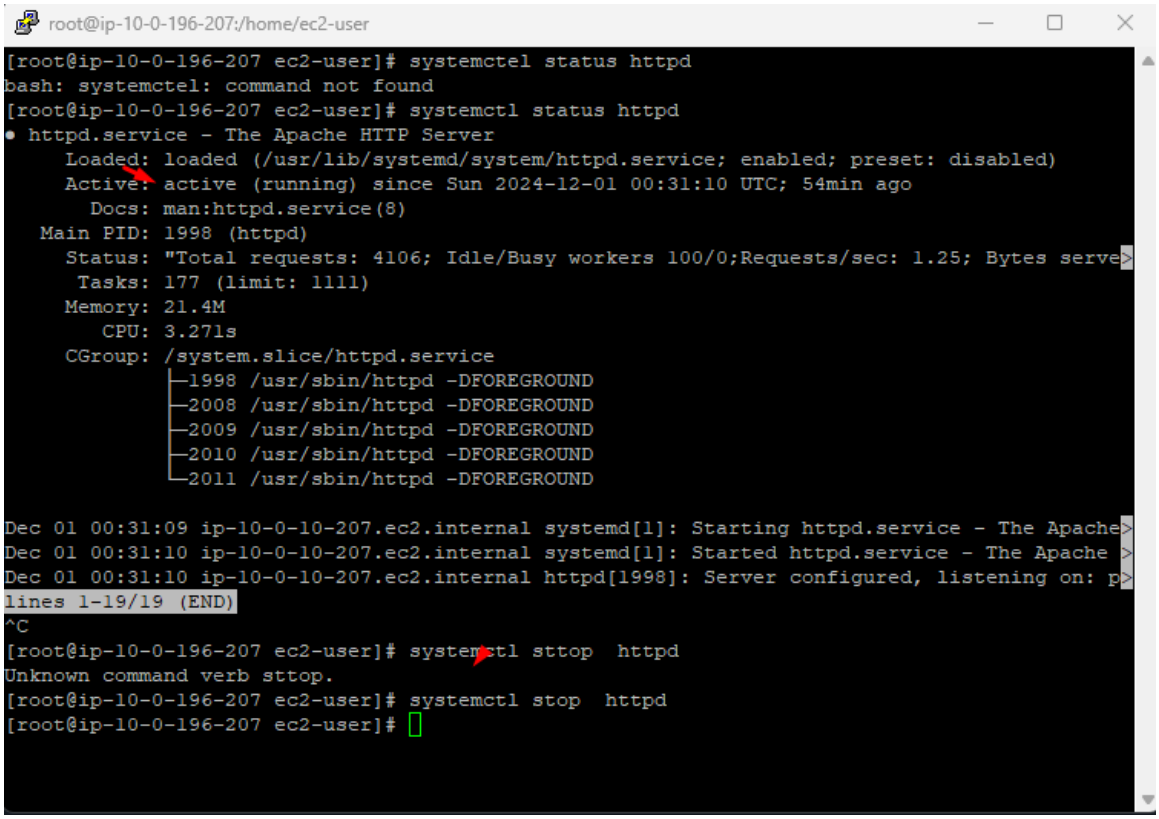
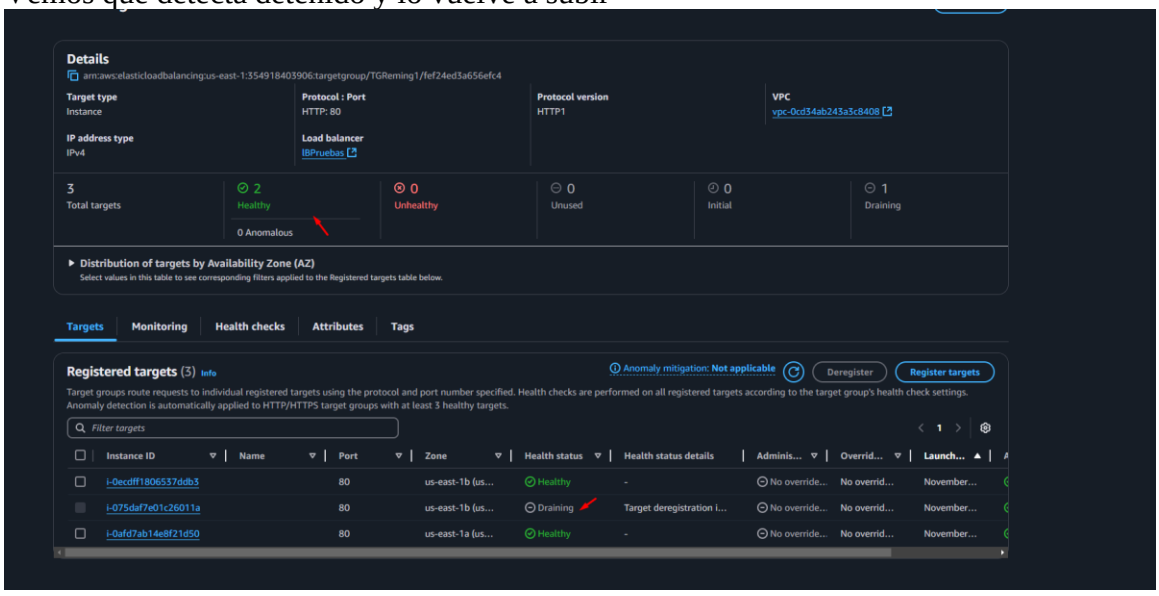


Figura 62.
Vemos que detecta detenido y lo vuelve a subir



Video entrega dos balanceador
https://youtu.be/9_f4AFg75PM

Entrega final 3

Docker
Figura 63
Instalando nginx

```
[root@ip-10-0-13-122 html]# dnf install nginx -y
```

Figura 64.
Subimos el servicio

```
[root@ip-10-0-13-122 html]# systemctl start nginx
```

Figura 65.
Comprobamos el servicio

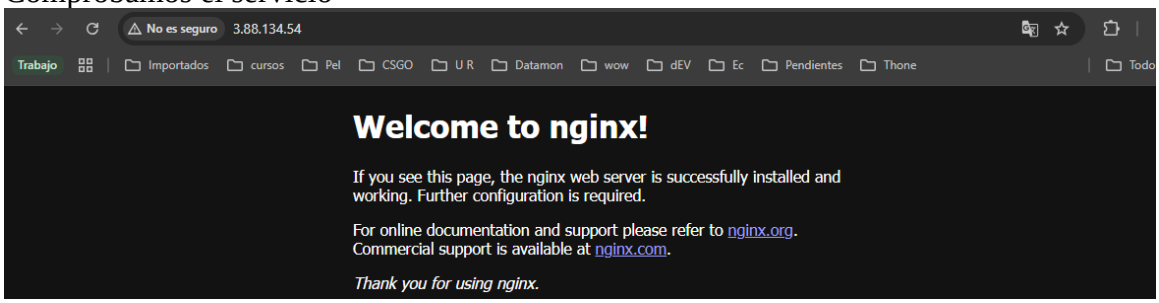


Figura 66.
Instalamos Docker

```
Complete!
[root@ip-10-0-13-122 html]# yum install docker -y
```

Figura 67.
Subimose el servicio

```
Complete!
[root@ip-10-0-13-122 html]# systemctl start docker
[root@ip-10-0-13-122 html]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /u
r/lib/systemd/system/docker.service.
[root@ip-10-0-13-122 html]# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: d
     Active: active (running) since Sun 2024-12-08 15:58:40 UTC; 23s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 29419 (dockerd)
      Tasks: 7
     Memory: 38.6M
        CPU: 271ms
     CGroup: /system.slice/docker.service
```

Figura 68.
Creamos un contenedor por medio de una imagen de Docker se puede descargar desde un hub.

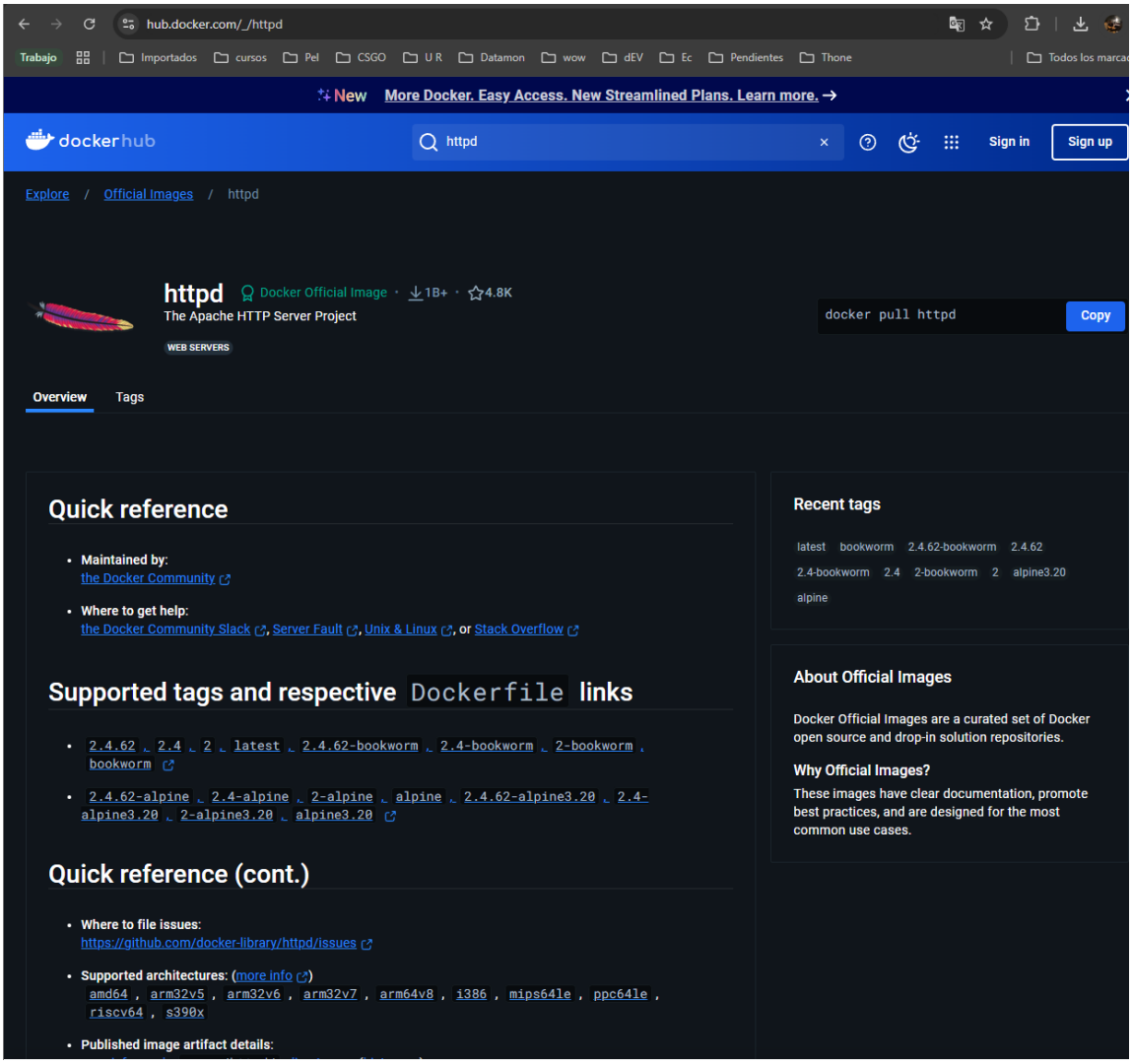


Figura 69.

Descargamos la imagen

```
[root@ip-10-0-13-122 html]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
bc0965b23a04: Pull complete
d7ad38c6dd97: Pull complete
4f4fb700ef54: Pull complete
79b49624e34b: Pull complete
7d9f97915db2: Pull complete
9bd25d4f7b77: Pull complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

Figura 70.

Comprobamos las imagenes

```
[root@ip-10-0-13-122 html]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    494b2b45fd74   4 months ago   147MB
[root@ip-10-0-13-122 html]# ~
```

Figura 71.

Creamos una carpeta para guardar la informacion

```
[root@ip-10-0-13-122 html]# cd /
[root@ip-10-0-13-122 /]# mkdir appl
```

Figura 72.

Iniciamos a crear el contenedor

Por medio del comando -d (segundo plano) it (sea interactivo) – name (para el nombre) -p (indicar puerto) -v volumen (donde indicamos la ruta local y la ruta del servicio) –restart always (para que suban solos los servicios)

```
[root@ip-10-0-13-122 /]# docker run -dit --name appl --restart always -p 8080:80 -v /appl:/usr/local/apache2/htdocs/  httpd
7a177d469368895cca555c3d36b9db10a1da482e08871e20acb4a1007d424576
[[root@ip-10-0-13-122 /]#
```

Figura 73.

revisamos la instancia

```
CONTAINER ID   IMAGE     COMMAND              CREATED          STATUS          PORTS                                   NAMES
7a177d469368   httpd     "httpd-foreground"   23 seconds ago   Up 22 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   appl
[root@ip-10-0-13-122 /]#
```

Figura 74.

Apagar servicio de nginx para evitar conflictos de puertos

```
[root@ip-10-0-13-122 appl]# systemctl stop nginx
```

Figura 75.

Reiniciamos También la instancia de Docker y creamos otra instancia con puerto 80:80

```
[root@ip-10-0-13-122 appl]# docker stop 7a177d469368
7a177d469368
-rwxrwxrwx.  1 root root  17 Dec  8 16:21 index.html
[root@ip-10-0-13-122 appl]# docker run -dit --name app2 --restart always -p 80:80 -v /appl:/usr/local/apache2/htdocs/  httpd
f4cec4356b0de8d8b0b1b492c8a01b6e426319bd5be849623b196c7f44db4294
```

Figura 76.

Iniciamos proceso de configurar proxy reverso con nginx.

```
root@ip-10-0-13-122 app2]# systemctl start nginx
root@ip-10-0-13-122 app2]# cd /etc/nginx
root@ip-10-0-13-122 nginx]# ls
conf.d      fastcgi.conf          fastcgi_params          koi-utf  mime.types          nginx.conf          scgi_params          uwsgi_params          win-utf
efault.d    fastcgi.conf.default  fastcgi_params.default  koi-win  mime.types.default  nginx.conf.default  scgi_params.default  uwsgi_params.default
root@ip-10-0-13-122 nginx]#
```

Figura 77.

Indicamos los puertos y configuración del nginx

Figura 78.

Verificamos funcionamiento

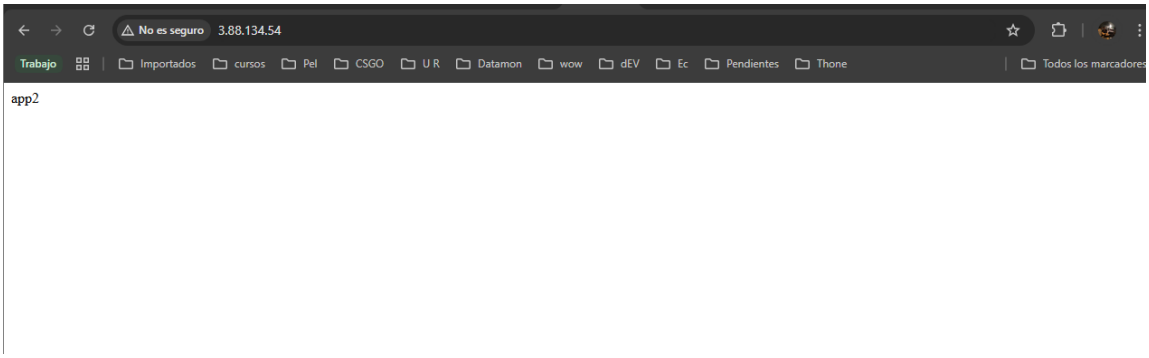


Figura 79.

Realizamos el procedo de crear nuevamente la ami (como la imagen 20)

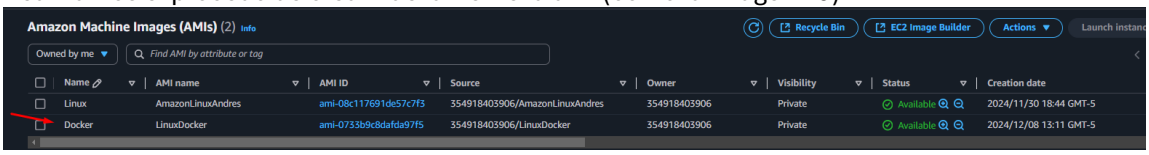


Figura 80. Agregamos la ami para que acople al auto balanceador con los ajustes realizados el Docker

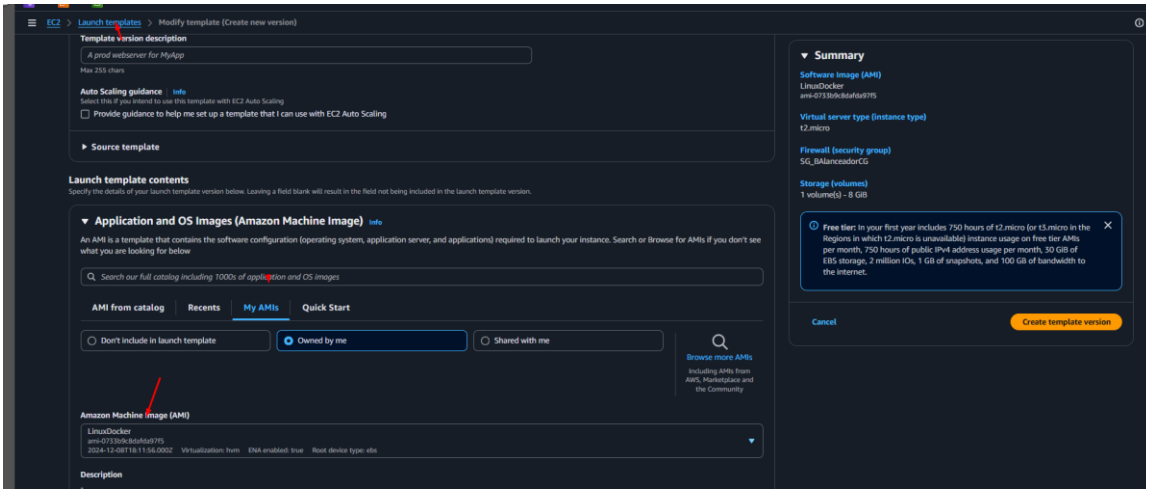


Figura 81.

Comprobamos que al detener los servicios internos de la instancias este nos registre y las vuelva a subir

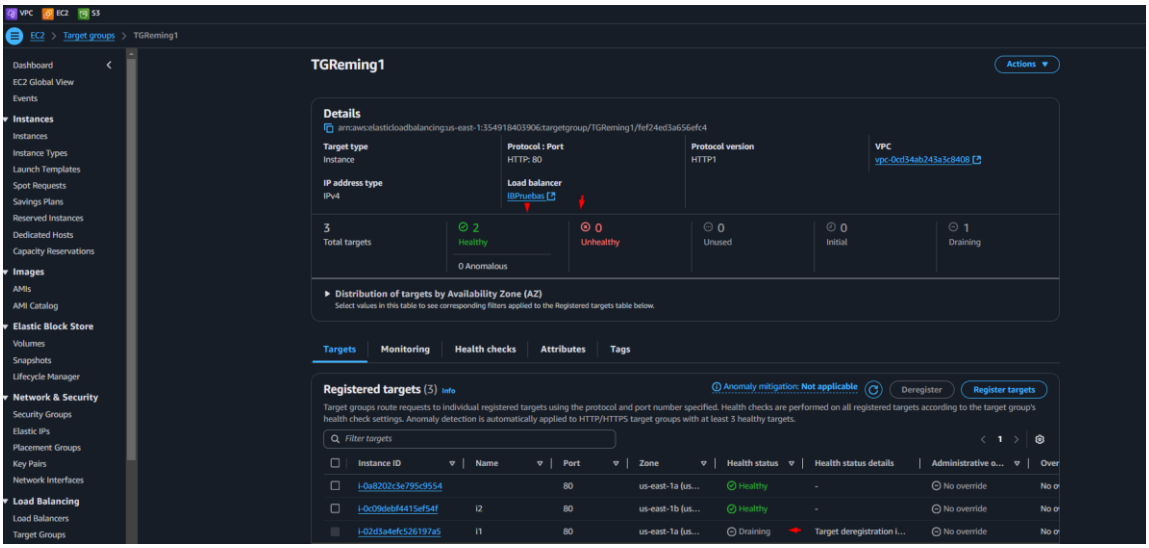
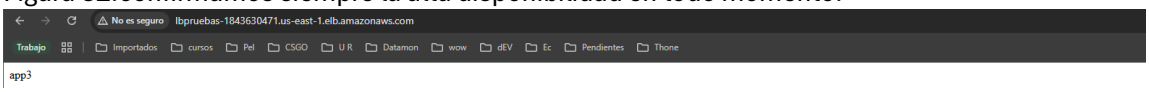
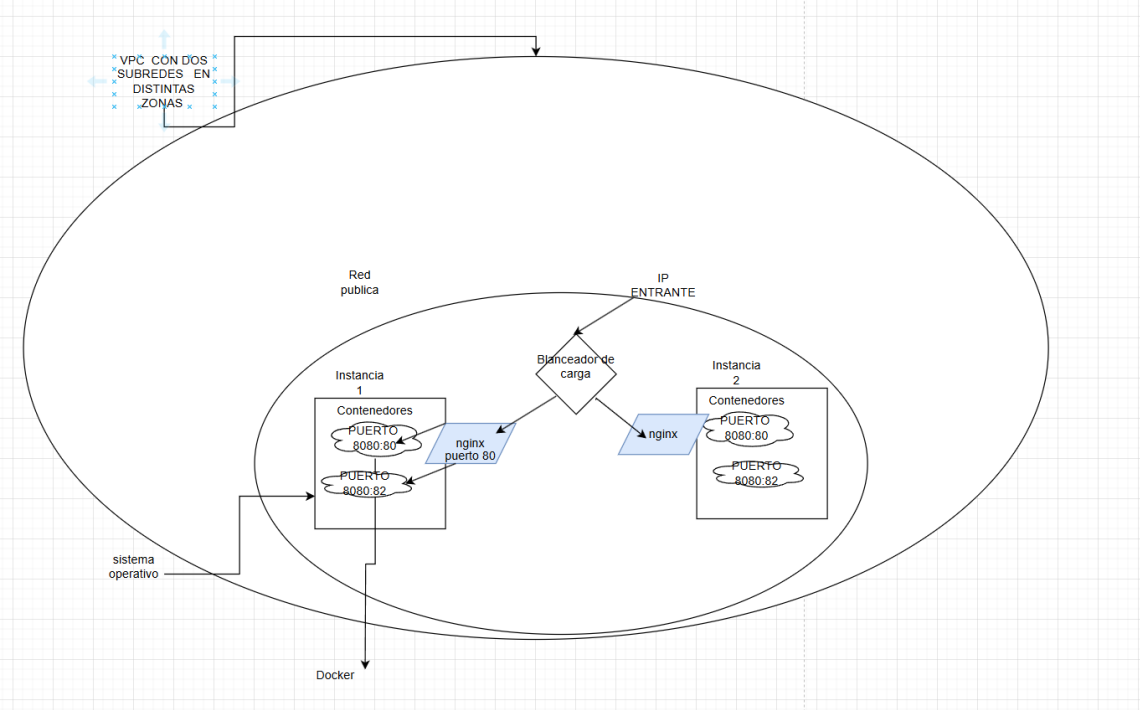


Figura 82.confirmamos siempre la alta disponibilidad en todo momento.



app3

Diagrama Funcional



Video entrega 3 Final

Conclusiones

Encontramos durante el seminario la importancia de este modelo de alta disponibilidad permitiendo tener una serie de Backup de modo que si una instancia falla casi que de manera instantánea automáticamente se sube otra instancia esto ayuda a optimizar la buena experiencia de usuarios y la gran calidad de conectividad ayudando de esta forma a las empresas o personas que usen este modelo de aws a mejor sus ganancias y potenciar su calidad de estructura tecnológica ya que en este modelo las conectividades siempre están disponible.

Referencias

(Aws, 2024)
(docker, 2024)