TRABAJO DE GRADO
Opción Seminario-Diplomado.

Amazon Web Services

Karol Andrea Medina Suarez
Tatiana Andrea Latorre Muñoz
Jhon Breinner Bentiez Guerrero

Corporación Universitaria Remington.
Facultad De Ingeniería
Ingeniería En Sistemas

Juan Pablo Berrio López
Seminario-Diplomado.
2024

Dedicatoria

Ha sido años lleno de esfuerzos y sacrificios, cerrada esta etapa, nos queda dedicar y agradecer principalmente a Dios por habernos dado salud; fuerza para lograr nuestros objetivos, a nuestros padres por apoyarnos en todo momento, amigos y colegas que nos motivan a seguir obteniendo conocimiento informático con una fuerte pasión en la tecnología, a los profesores encargados de brindarnos sus clases con enfoque y determinación en enseñar de la mejor forma. Gracias al profesor Juan Pablo Berrio López por su guía a lo largo de este trabajo que nos enseñó el valor de la dedicación y el esfuerzo, sin mucho más que decir te agradecemos a ti por leer este documento que lo disfrutes.

Tabla de Contenidos

Resumen

En este proyecto, somos un startup llamada **KTB SYSTEM**, una plataforma innovadora que ayudara conectar restaurantes con clientes mediante entregas rápidas. Nuestra empresa ha obtenido un rápido crecimiento para así escalar una infraestructura tecnológica, manejar una mayor demanda y poder garantizar la disponibilidad de mejorar los tiempos de respuesta El CTO de **FAST FOOD** se ha diseñado bajo una arquitectura lo cual necesita de una ayuda para implementar en Amazon Web Services (AWS). La solución debe ser altamente disponible, escalable y estar diseñada para manejar una gran cantidad de trafico de manera eficiente. Implementaremos esta arquitectura en AWS lo cual requiere los siguientes requisitos

1. **Balanceador de Carga**: Configure un Application Load Balancer (ALB) para distribuir el tráfico entrante a múltiples instancias EC2.
2. **Instancias EC2**: Implemente al menos dos instancias EC2 en una configuración multizona para garantizar alta disponibilidad.
3. **Instancias con Proxy Reverso**: Dentro de cada instancia EC2, deben implementar un proxy reverso (por ejemplo, Nginx) para redirigir solicitudes a servicios internos.
4. **Auto-escalado**: Configure políticas de auto-escalado para aumentar o reducir las instancias EC2 según la carga.
5. **Seguridad**: Asegure el tráfico utilizando grupos de seguridad adecuados y habilitando HTTPS en el balanceador de carga.
6. **Documentación**: Toda la documentación debe estar en el formato de trabajo de grado proporcionado.

Palabras clave

- Amazon web services
- VPC
- Instancia
- Balanceador de cargas
- Scaling groups

Marco conceptual y contextual

Nombre	Significado
Amazon Web Services (AWS) (Amazon Web Services, 2021)	Es una plataforma en la nube más completa, muchas personas confían en ella, puede soportar la infraestructura y aplicaciones. Muchas organizaciones de todo tipo la utilizan para reducir costos, ser más agiles e innovar más rápido, proporciona servicios tecnológicos bajo demanda a través de internet con precios de pago por uso. Tiene muchas características entre ellas ofrece más servicios que cualquier otro proveedor de nube lo cual hace que sea más rápido más fácil y rentable.
Amazon EC2(Amazon Elastic Compute Cloud) (Amazon web Services, 2021)	Se encarga de crear computadoras la cual tienen un estado base, pueden subir de nivel y regresar a su estado base cuando lo necesite, estas se conocen como máquinas virtuales o instancias. Una máquina virtual es una computadora que no es física sino simulada por software, porque no está limitado a los recursos de hardware sino que por el software puede hacer que de la misma computadora física hayan varias máquinas virtuales también por el software se puede hacer que crezcan o reduzcan sus recursos y el uso que se dé teniendo en cuenta esto se pueden crear aplicaciones y responder a la demanda, tiene también la capacidad de auto escala en automático solo se debe configurar las reglas para decirle cuando debe escalar y cuando debe regresar al estado inicial como también se le pueden poner límites para que no escale hacia el infinito.
Instancia (Amazon web Services, 2021)	Esta es un entorno virtual que brinda servicios en la nube de terceros, tiene el control desde que se inicia hasta que se elimina, también se puede personalizar para que se adapte a las necesidades correspondientes, esta puede administrar y mantener los recursos del servidor físico en las instalaciones. Los proveedores de la nube mantienen el hardware en sus centros

	de datos y le dan acceso virtual a los recursos de computación, puede usar la instancia en la nube para ejecutar cargas de trabajo con uso intensivo de cómputos, como contenedores, bases de datos, micro servicios y máquinas virtuales.
Amazon S3 (Amazon Web Services, 2021)	Este servicio es uno de lo más utilizados porque maneja un buen almacenamiento de objetos el cual ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Los usos que normalmente se manejan es la parte de almacenamiento y copia de seguridad, recuperación de daños, archivos, almacenamiento hibrido, aplicaciones, medios, data lake y analítica big data.
Amazon RDS (Amazon web Services, 2021)	Es un servicio racional fácil de manejar que elimina tareas administrativas que son ineficientes y que consumen tiempo este admite una variedad de motores de base de datos para almacenar y organizar datos también ayuda en las tareas de administración de base de datos como por ejemplo la copia de seguridad y la recuperación este también tiene facilidad de implementación
Amazon route 53 (Amazon web services , 2021)	Es un servicio web de nombres de dominios también es escalable y tiene una alta disponibilidad se pueden realizar en registro de dominio direccionamientos, verificación de estado también permite personalizar el enrutamiento de DNS para mejorar la disponibilidad y garantizar el cumplimiento normativo.
Amazon Machine Images-AMI (Amazon web Services, 2021)	Este servidor ofrece un software suficiente para configurar y arrancar una instancia de Amazon EC2 incluido el sistema operativo y la configuración de seguridad, uno de los beneficios que nos ofrece es la velocidad eficiencia y conformidad en la implementación.
Amazon Elastic block store (Amazon EBS) (Amazon Web Services , 2021)	Este servidor da recursos de almacenamiento en bloque de alto rendimiento y escalables se puede crear, administrar algunos recursos como volúmenes de Amazon EBS e instantáneas

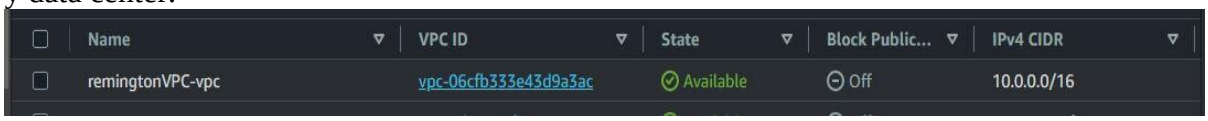
	también ofrece varios tipos de volúmenes copia de seguridad, recuperación protección de datos disponibilidad y durabilidad de los datos con archivado de datos.
Amazon EC2 Auto Scaling (Amazon Web Services, 2021)	Este método es muy utilizado en la nube por la cantidad de recursos computacionales ya que permite que reaccione en tiempo real a la demanda de recursos también tiene un sistema de servidores y este varia automáticamente en consecuencia de la carga total.
VPC (Amazon Web Services, 2021)	Es una red virtual idéntica a una red tradicional que opera su propio centro de datos esta puede crear sud redes, esta red virtual es muy similar a la red tradicional que usaría en su propio centro de datos, pero con los beneficios que supone utilizar la infraestructura escalable de AWS.
Sud-red (Amazon web Services, 2021)	Una subred es una red que forma parte de una red principal. Las subredes potencian la eficiencia de las redes. Con la formación de subredes, el tráfico de la red puede recorrer una distancia reducida sin necesidad de transitar por routers superfluos para alcanzar su destino.
Docker (Perlow, 2024)	Es un software destinado a contenedores. El motor de Docker se instala en cada servidor donde se requiera la ejecución de contenedores, ofreciendo un conjunto simple de comandos que puede emplear para crear, iniciar o detener dichos contenedores.
Nginx	servidor proxy, su función es reenviar las peticiones a los contenedores docker
Ip privada	Numero identificativo de dispositivos internos de una red; usado en las instancias.
Index.html	Archivo de configuración para código web.
Security group	Seguridad para la configuración de puertos y paso de la información en una instancia o sistema operativo.
Root	Usuario administrador de un sistema operativo Linux.

Ssh	Herramienta de terminal para línea de comandos que nos permite conexión al instancias
Puertos	Numero identificador que guía el paso de la información por la red
Target Group	Grupo de instancias que funcionan en conjunto para un buen funcionamiento de la escalabilidad.
Ip publica	Numero identificativo de dispositivos Hardware como por ejemplo un Router de borde que proporción acceso a la red internet.
Httpd	servidor web que recibe las peticiones de los clientes, brindado el servicio.

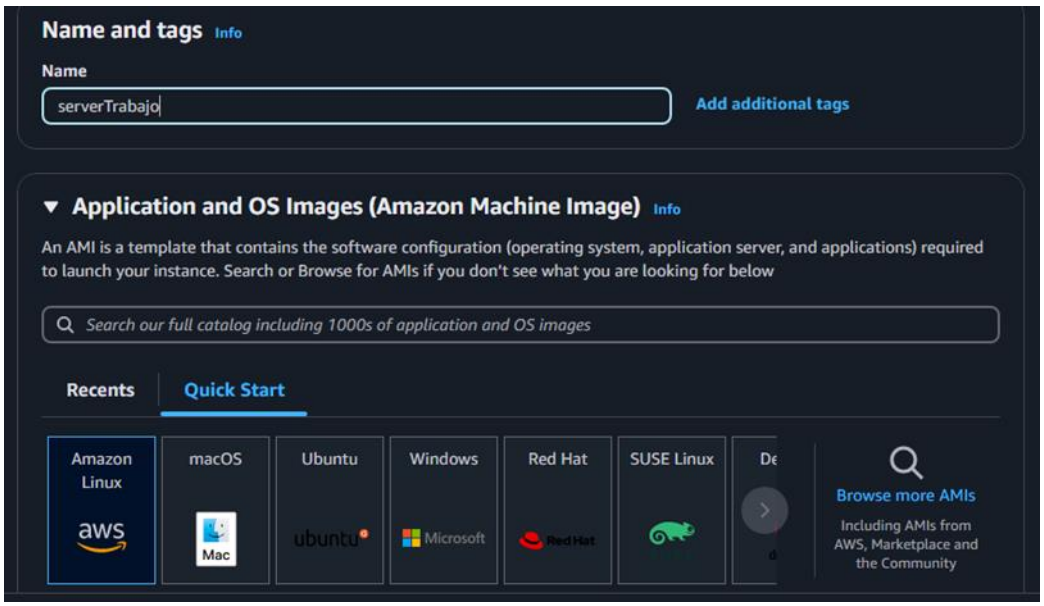
Creación de red, instancia y configuración de puertos

Configuración dentro AWS

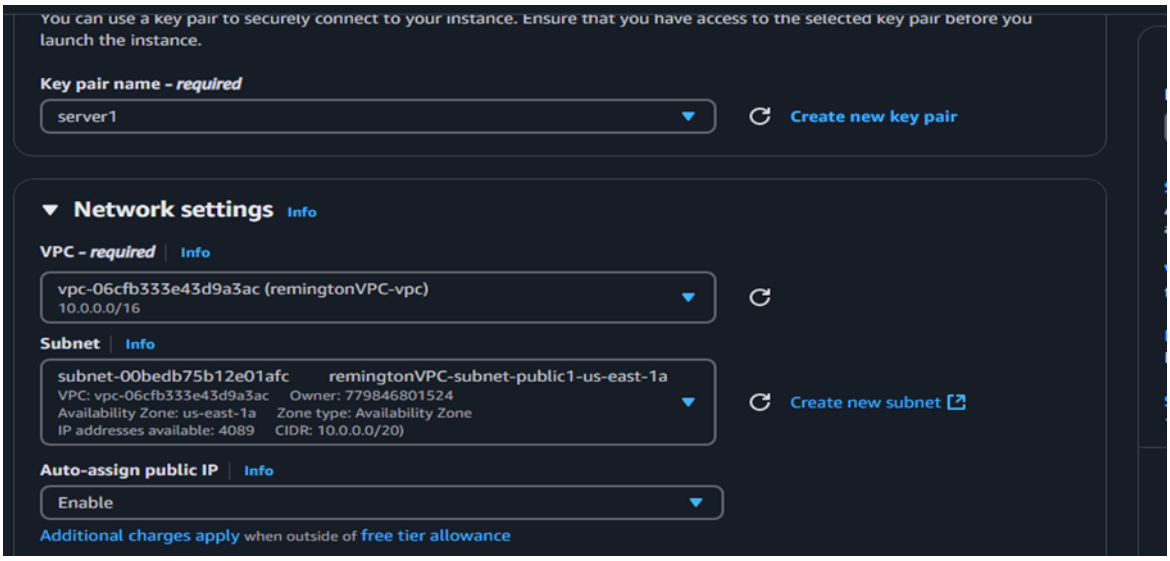
Paso 1. Creamos una red virtual (VPC), con dos subredes pública y privada en una sola región y data center.



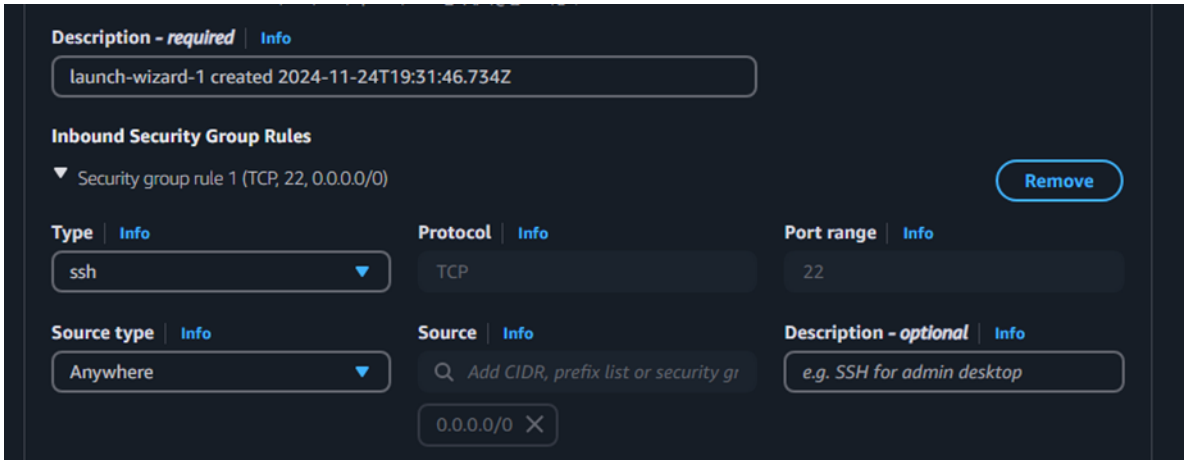
Paso 2. Creamos una instancia con Amazon Linux.



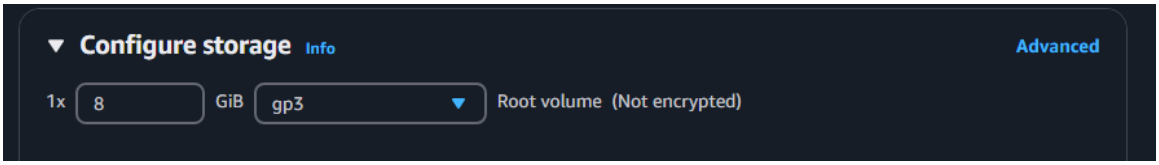
Paso 2.1. Configuración de clave privada para administrador del servidor SSH y IP publica que usa en la subred del VPC



Paso 2.2. Configuración SSH.

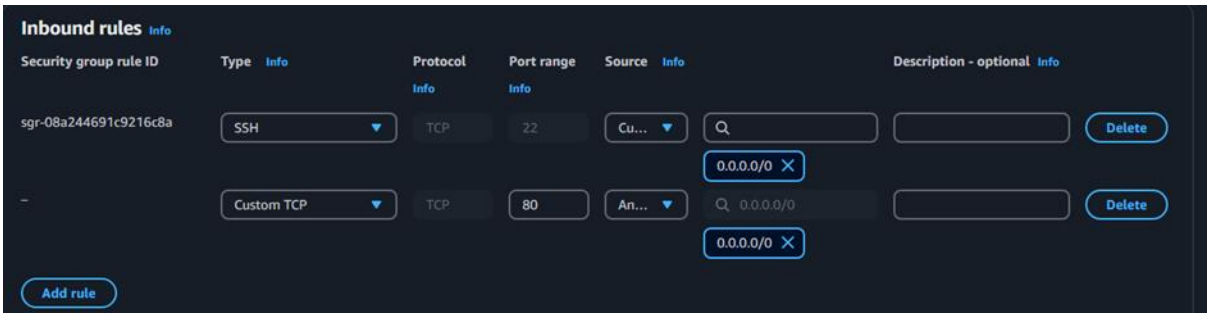


Paso 3. Configuración de almacenamiento.



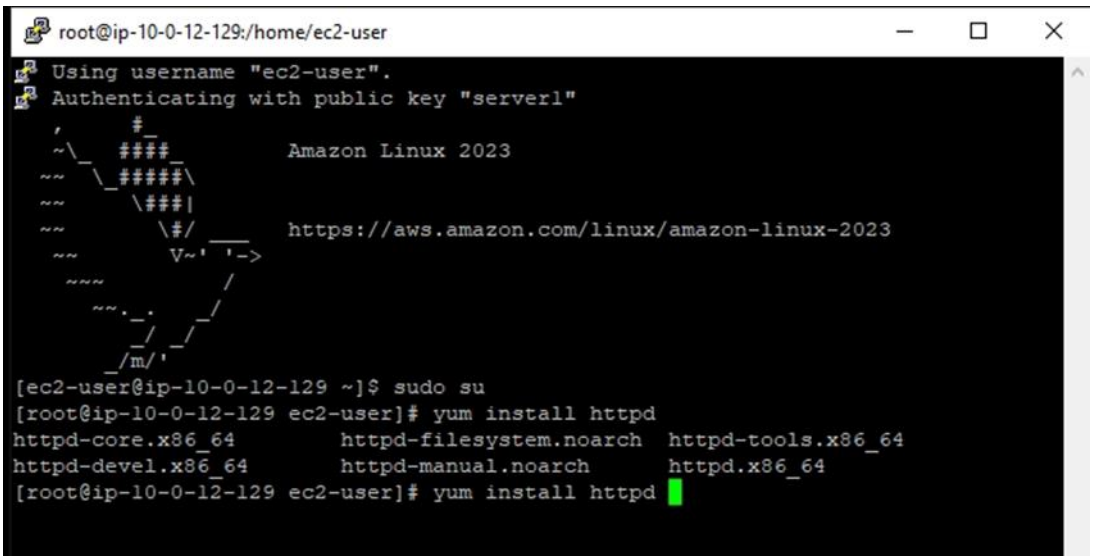
Paso 4. Se crea la instancia.

Paso 4.1. se habilita el puerto 80 por donde ejecutamos el servicio apache (servidor web) cualquier IP puede ingresar por este puerto.

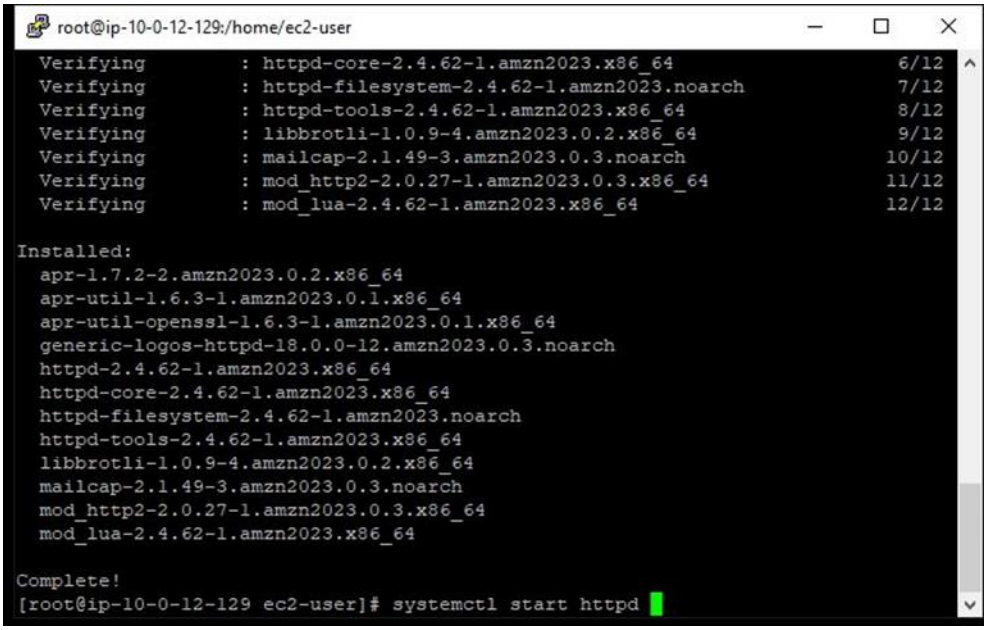


Configuración dentro de la instancia

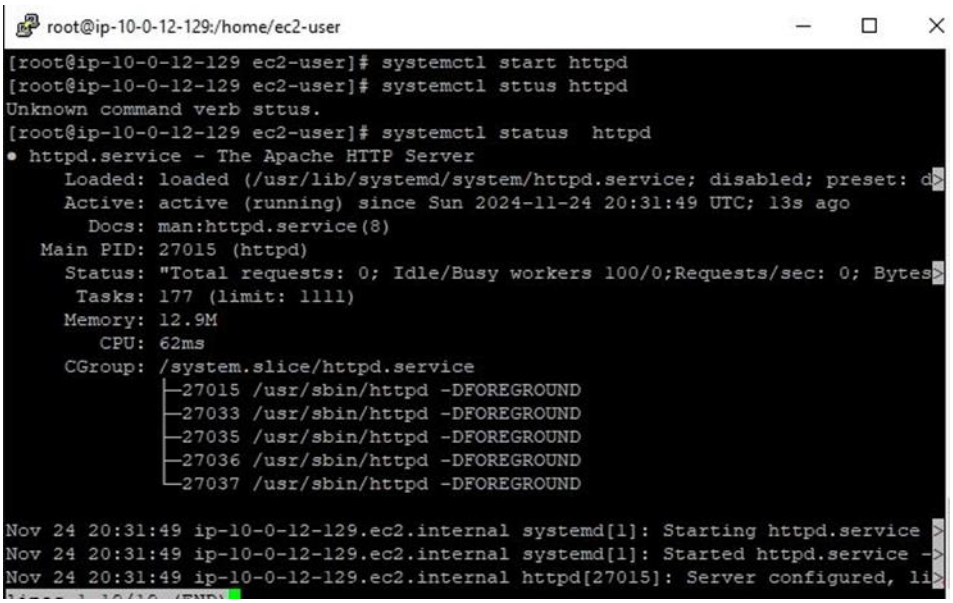
Paso 1. Instalamos apache en usuario root



Paso 1.1. Iniciamos servicio



Paso 1.2. Servicio en running



Paso 1.3. creación de archivo index.html con permisos para creación de página de web

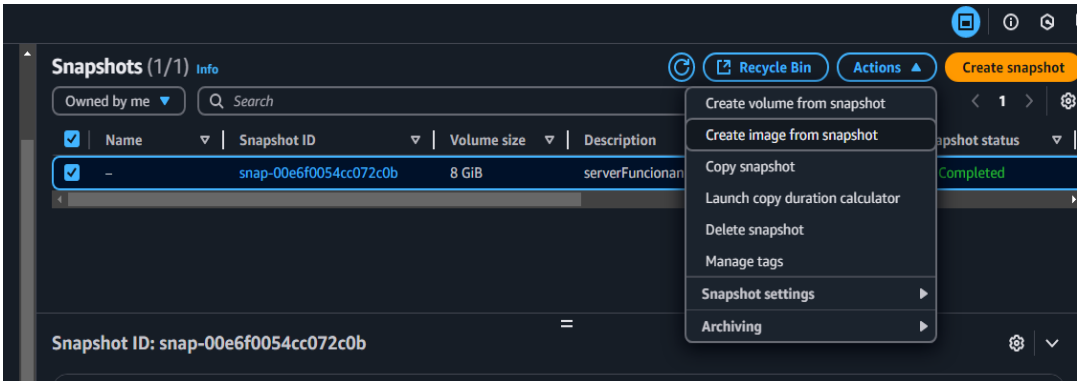
```
[root@ip-10-0-12-129 /]# cd /var/www/html/
[root@ip-10-0-12-129 html]# ls
[root@ip-10-0-12-129 html]# touch index.html
[root@ip-10-0-12-129 html]# chmod 777 index.html
[root@ip-10-0-12-129 html]# ls
index.html
[root@ip-10-0-12-129 html]# ls -l
total 0
-rwxrwxrwx. 1 root root 0 Nov 24 20:37 index.html
[root@ip-10-0-12-129 html]#
```

Vinculo YouTube - explicación Visualizar

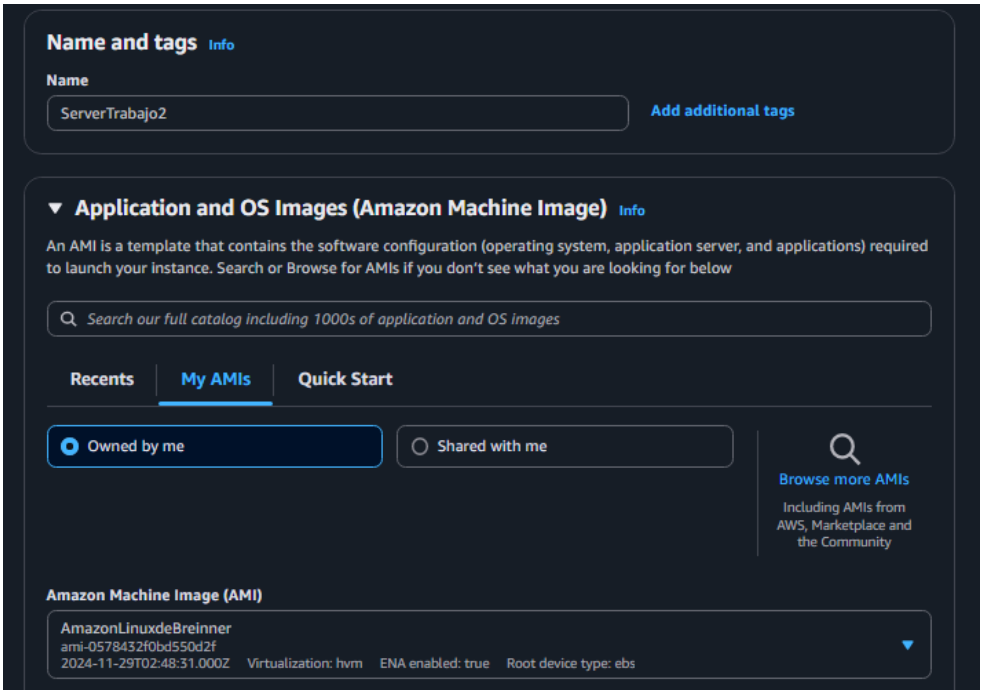
https://www.youtube.com/watch?v=kyzVRSNIav8

Implementación del balanceador de cargas con el scaling

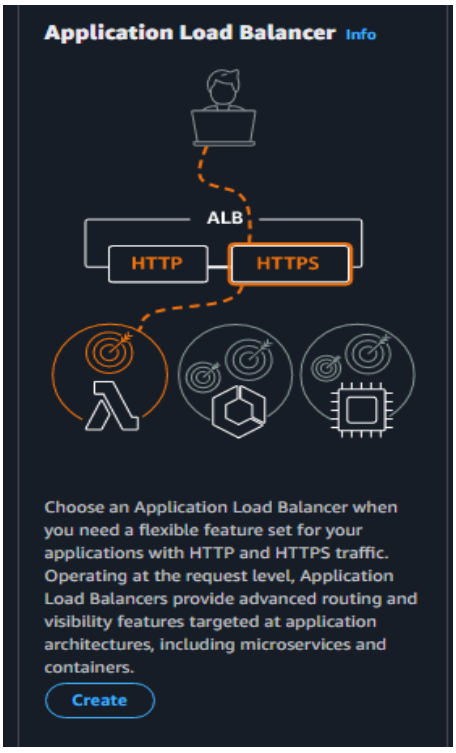
Paso 1. Creamos un snapshots de la instancia ya configurada para crear la AMI a utilizar y configurar otra instancia, esto con el fin de crear el balanceador de cargas con dos servidores funcionado



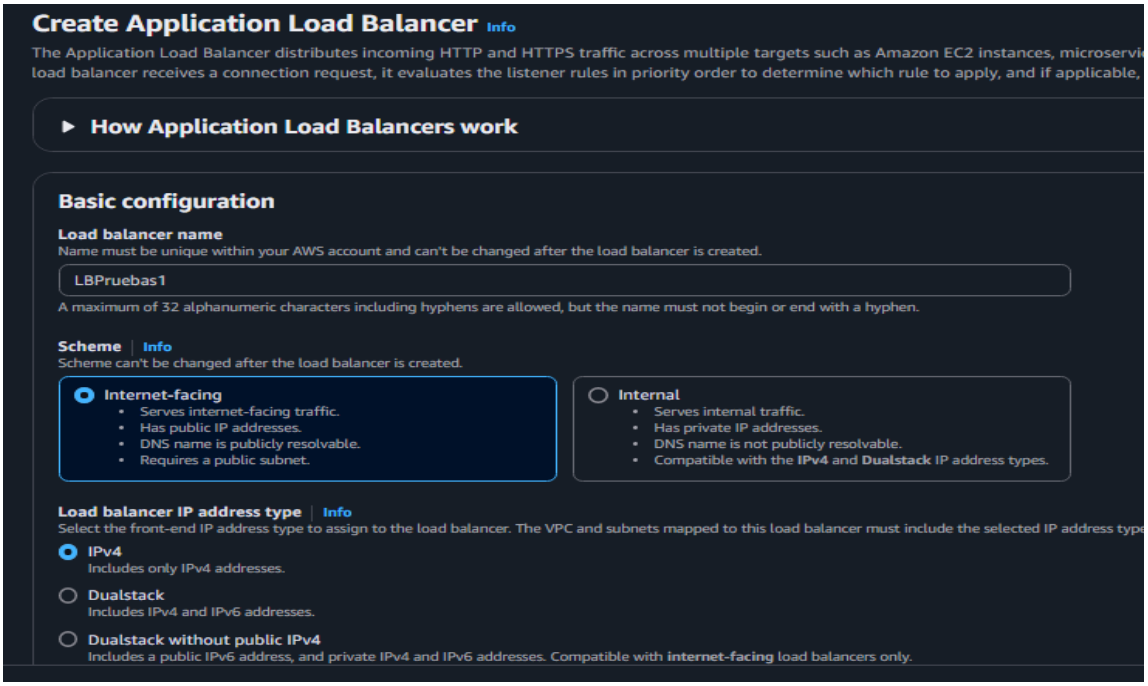
Paso 1.1. Se configura la instancia nueva a partir de la AMI creada



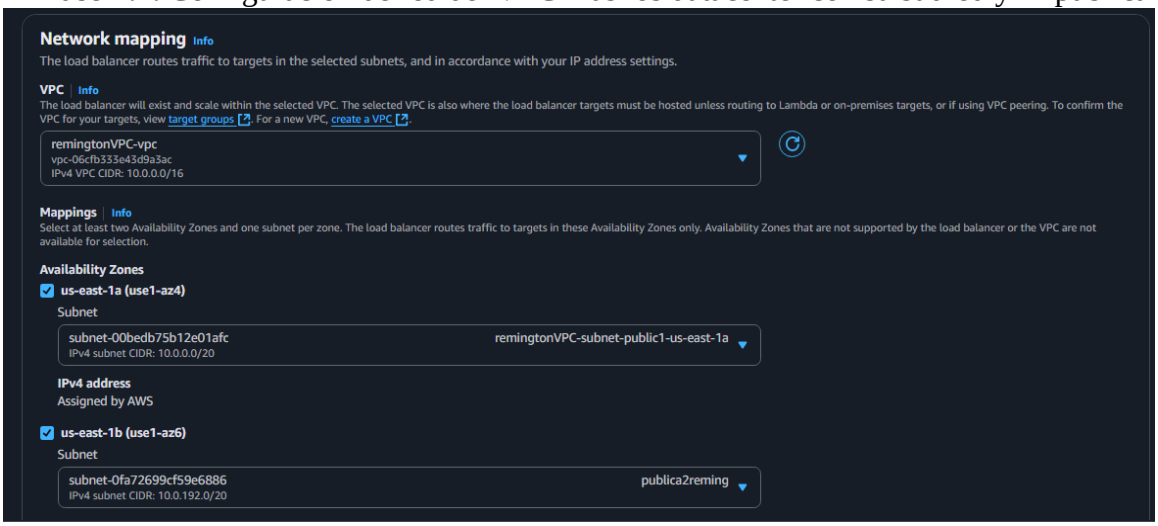
Paso 2. Una vez con las dos instancias creadas nos vamos a la opción de crear load Balanacers y comenzamos su configuración, será un balanceador de cargas de aplicación



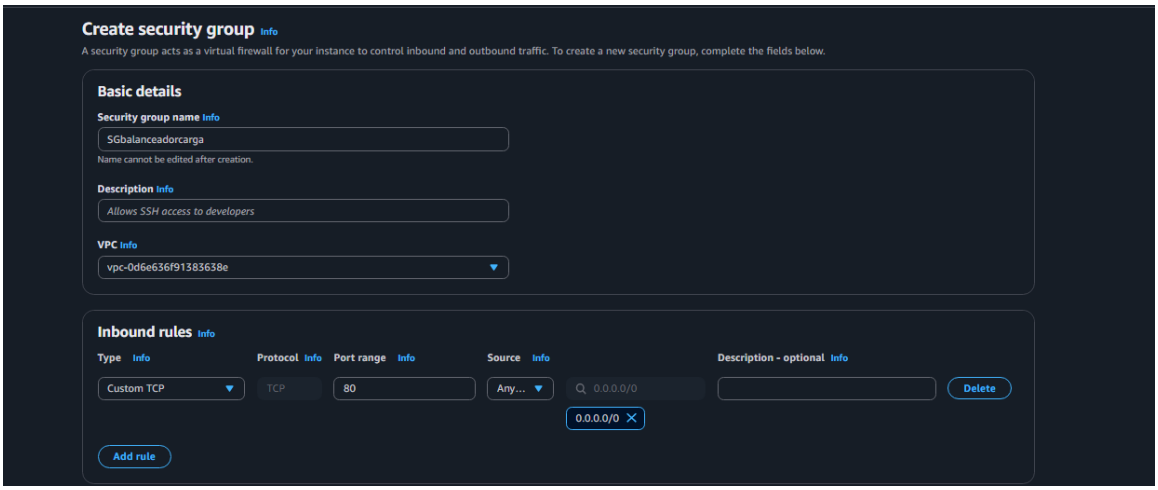
Paso 2.1. Colocamos nombre y se deja la opción de poder conectarnos desde la red internet



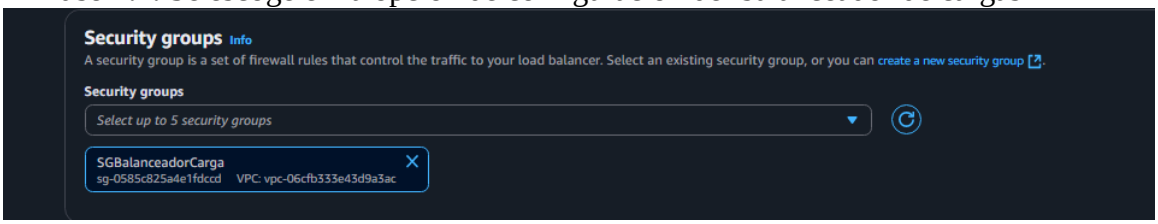
Paso 2.2. Configuración de red del VPC mas los datacenter con su sudred y IP publica



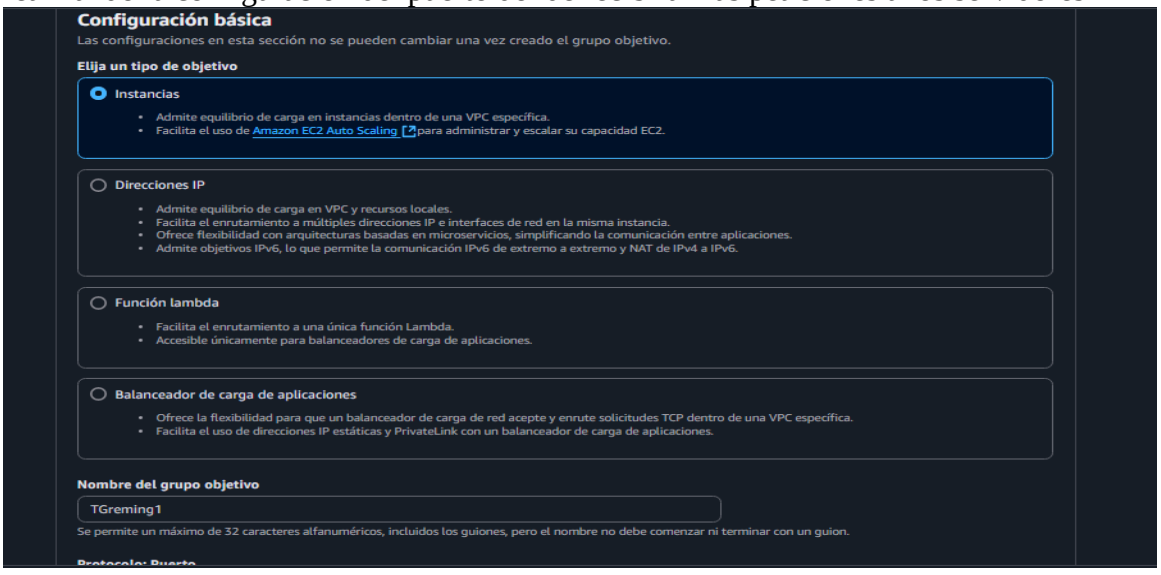
Paso 2.3. Se crea security group, que será la que recibirá las peticiones del balanceador de cargas para luego distribuir el trafico

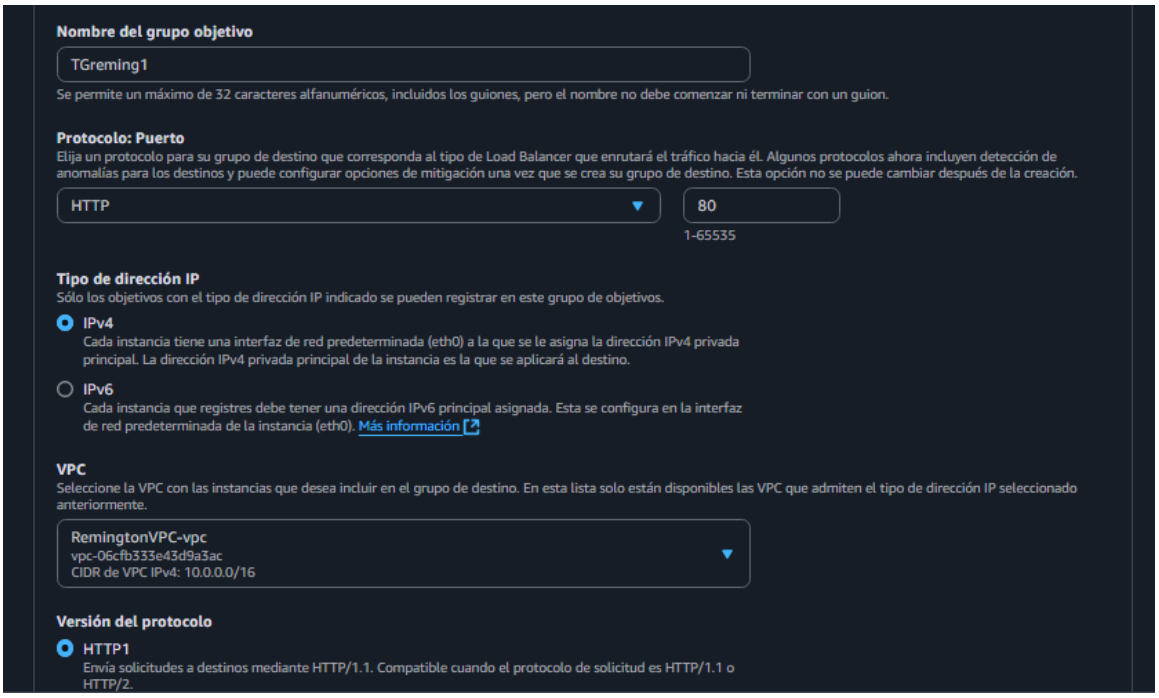


Paso 2.4. Se escoge en la opción de configuración del balanceador de cargas

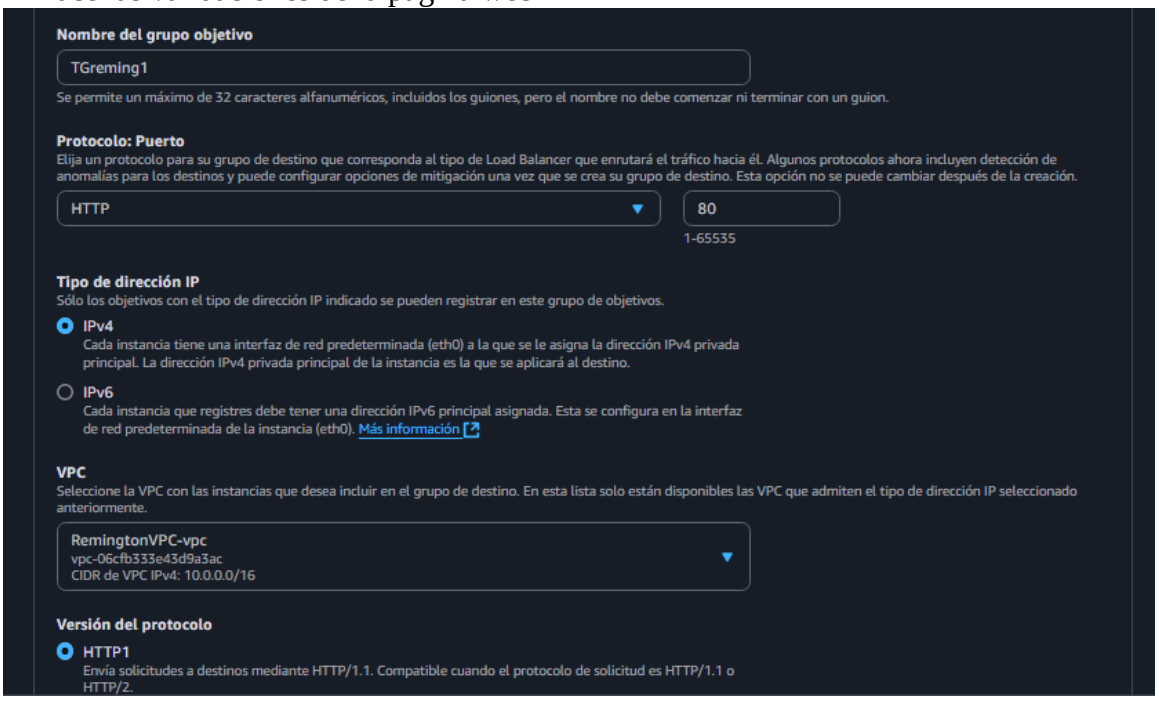


Paso 2.5. Se crea target group donde pondremos las instancias a trabajar en conjunto, realizando la configuración del puerto donde recibirán las peticiones a los servidores

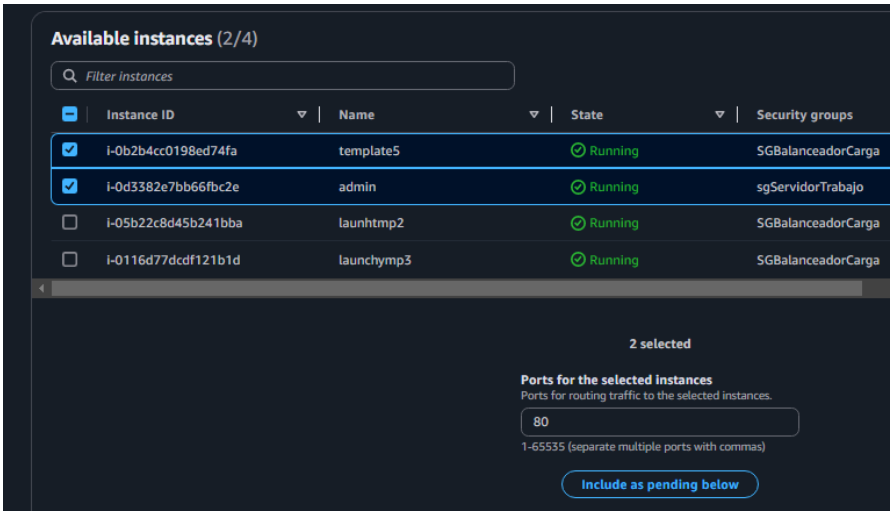




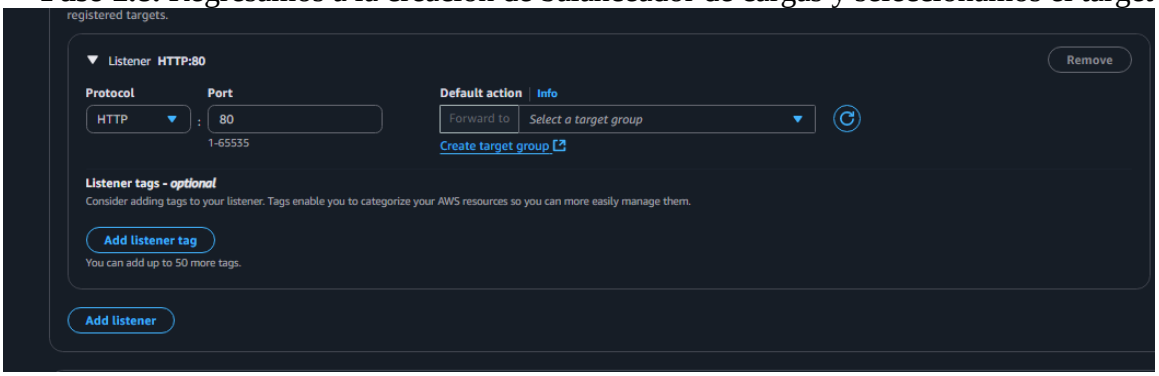
Paso 2.6. Configuramos la ruta de la aplicación web se deja por defecto raíz (/) donde apache hace las validaciones de la página web



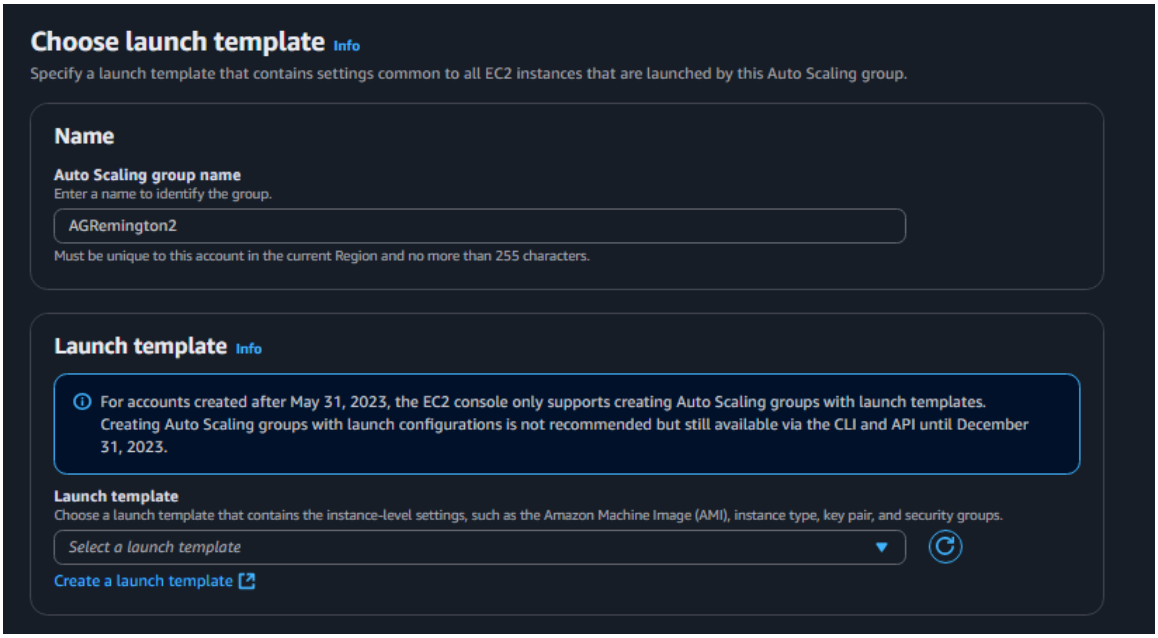
Paso 2.7. Luego se escogen las instancias a trabajar en conjunto para crear el target



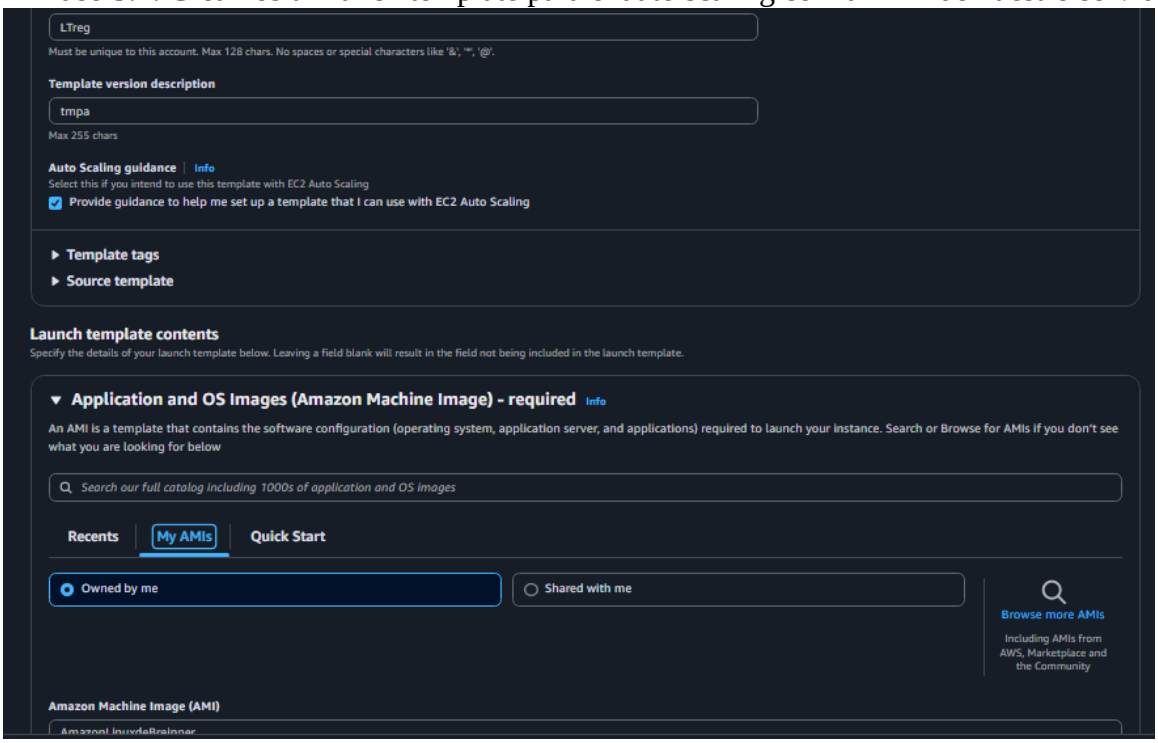
Paso 2.8. Regresamos a la creación de balanceador de cargas y seleccionamos el target group



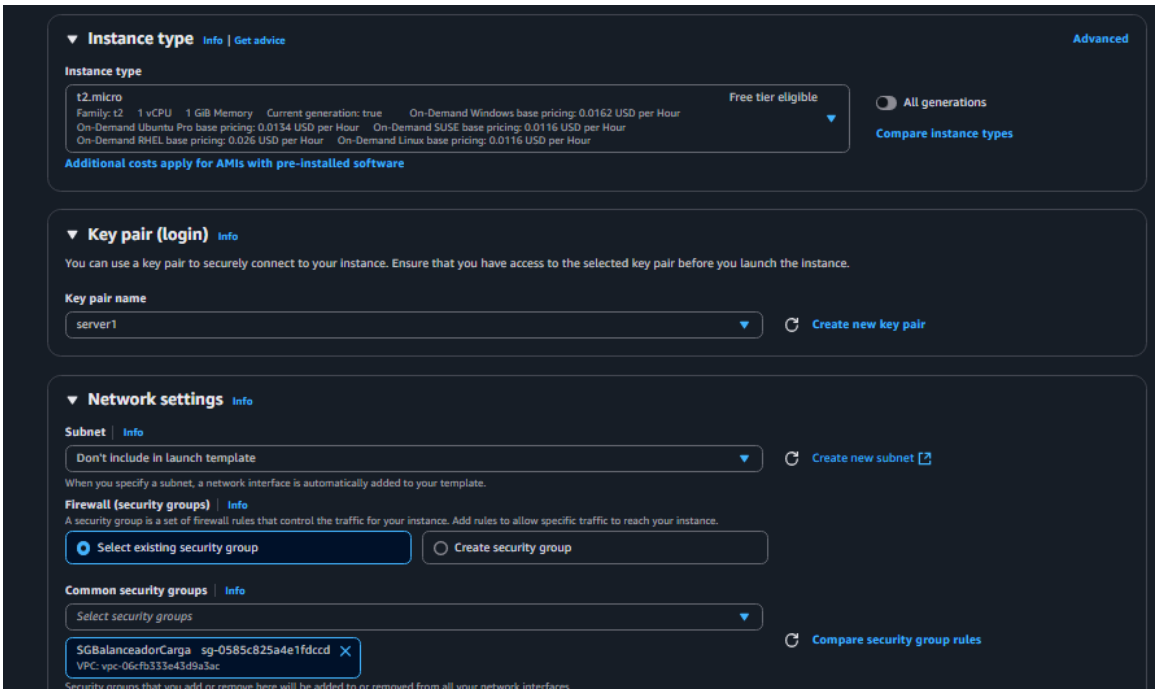
Paso 3. Nos dirigimos a la opción del auto Scaling groups para su creación



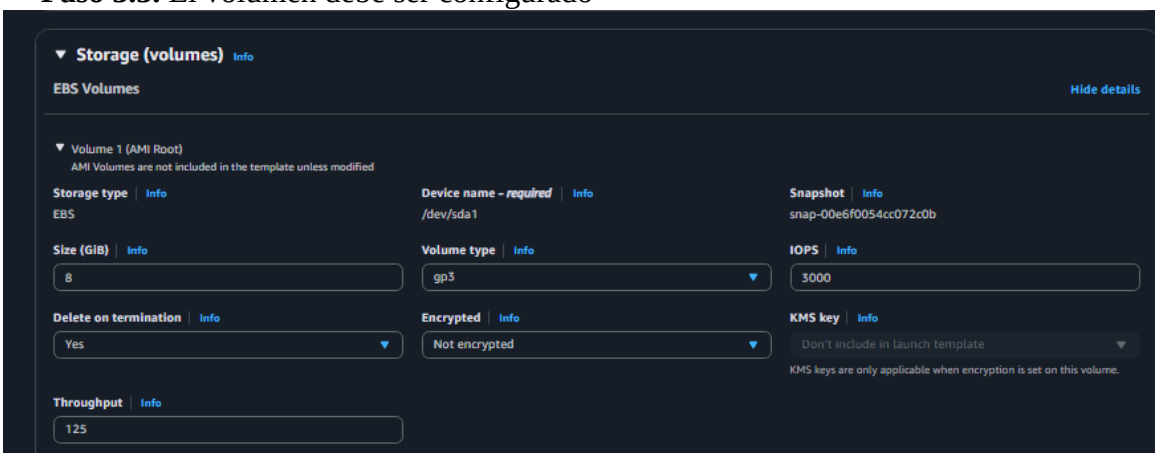
Paso 3.1. Creamos un lunch template para el auto scaling con la AMI de nuestro servidor



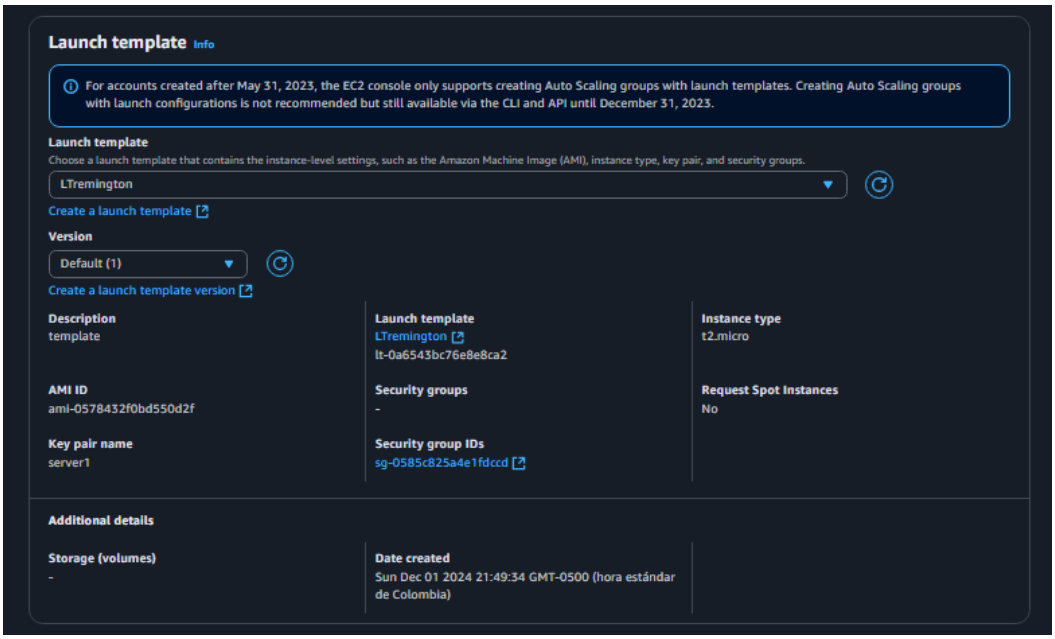
Paso 3.2. Configuramos el tipo de instancia, clave privada y configuración de red con el SG del balanceador de carga



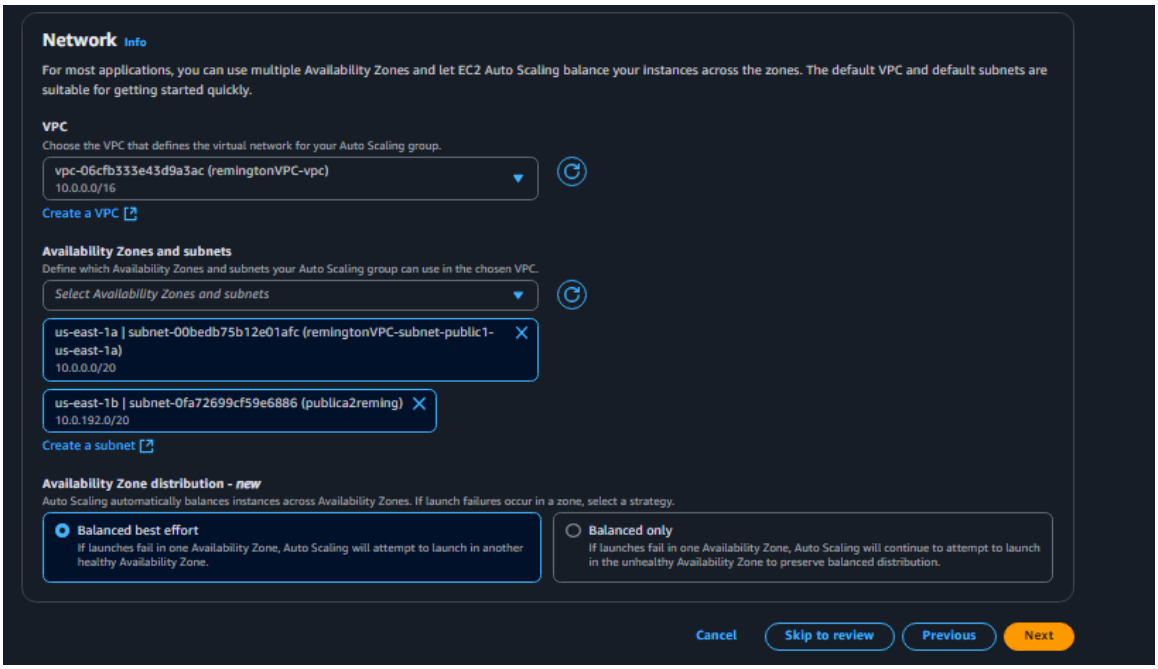
Paso 3.3. El volumen debe ser configurado



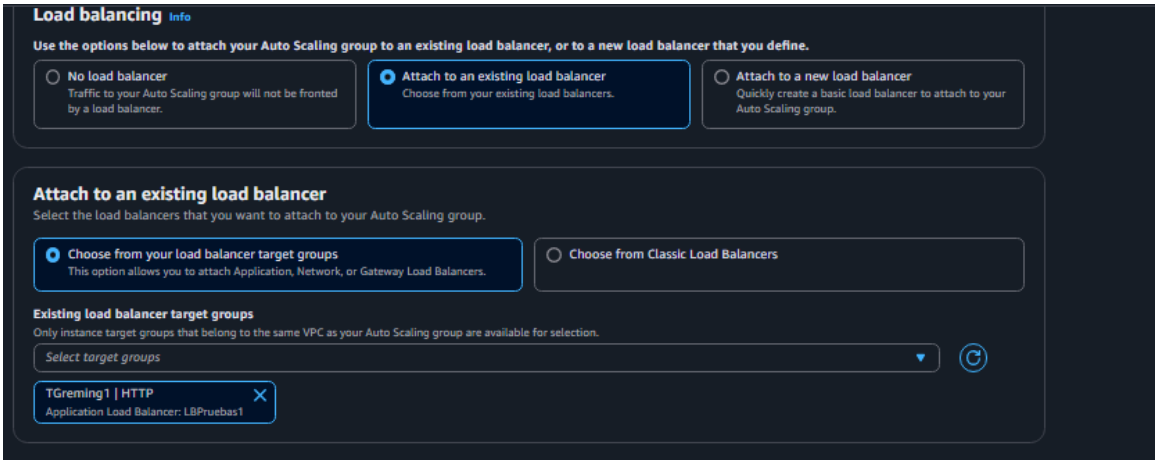
Paso 3.4. Volvemos a la configuración del auto scaling group y escogemos el template configurado



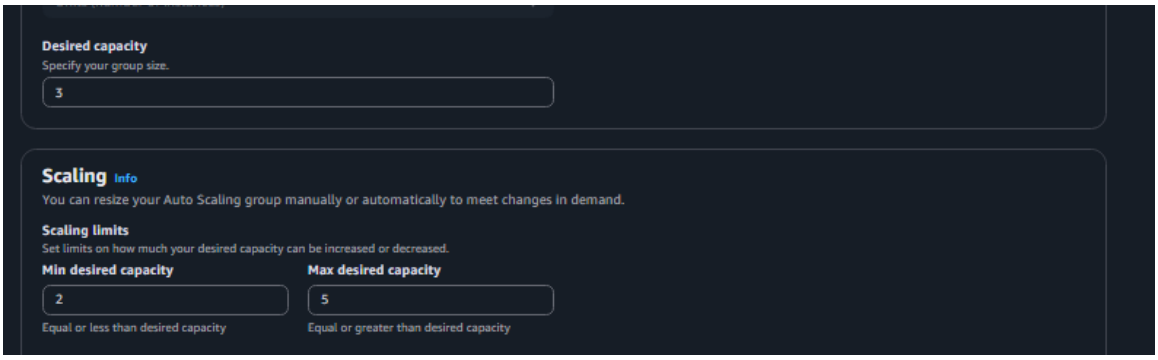
Paso 3.5. Configuramos la red del auto Scaling con el VPC y las dos zonas de data center con sus subredes públicas; presionamos next.



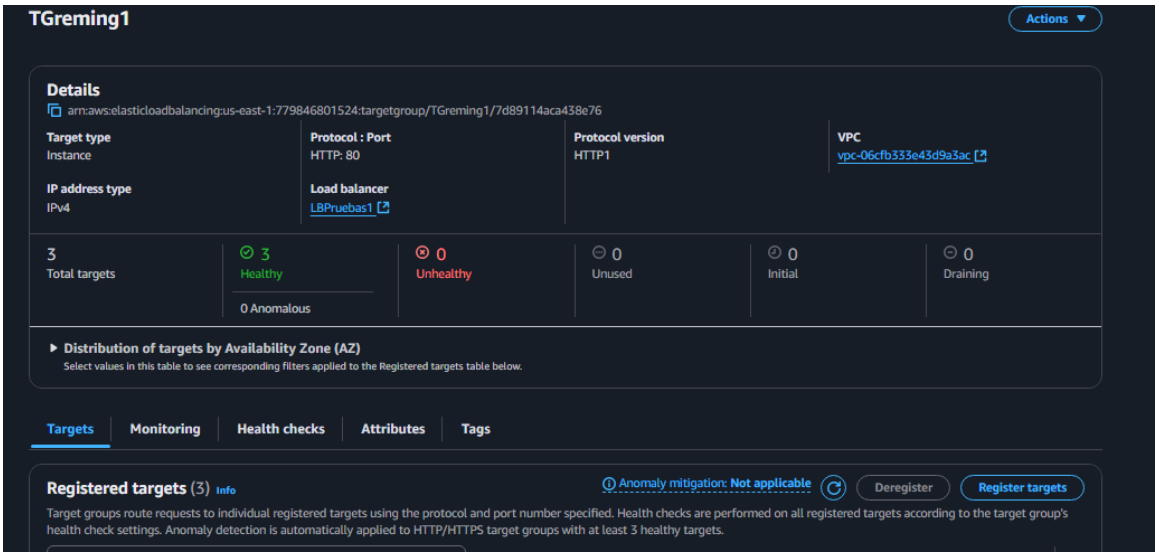
Paso 3.6. Escogemos el balanceados de cargas ya creado para nuestro scaling group



Paso 3.7. Escogemos el valor deseado de instancias, el mínimo y el máximo presionamos next hasta su creación del scaling groups.



Paso 3.8. Una vez creado el scaling groups, iremos a la parte de target group para mirar el funcionamiento de la infraestructura creada; si pausamos el servicio eliminando una instancia se creará otra automáticamente y el servicio web seguirá funcionando correctamente.



Vinculo YouTube- explicaicon visual

https://youtu.be/xMILvRhVs8E

Implementación de Docker con proxy inverso

Implementación de Contenedores

Hasta este punto toda la infraestructura está trabajando correctamente con el VPC, subredes, las instancias, el balanceador de cargas y el auto scaling sin embargo deseamos implementar docker en cada una de las máquinas para mayor escalabilidad.

Paso 1. Una vez conectado en la instancia, primero deshabilitamos el servicio de apache instalado en la AMI, con el comando **systemctl stop httpd** y **systemctl disable httpd**, luego de ejecutar estos dos comandos comprobamos que el servicio web este apagado con el comando **systemctl status httpd**.

```
[root@ip-10-0-4-81 nginx]# systemctl  status  httpd
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
```

Paso 1.1. instalamos el servidor web nginx, con el comando **yum install nginx**, luego lo iniciamos con el comando **systemctl start nginx** presionamos tecla enter e ingresamos este comando **systemctl enable nginx** que nos ayuda al inicio automáticamente por ultimo ingresamos el comando **systemctl status nginx** se debe visualizar activado.

```
[root@ip-10-0-4-81 nginx]# systemctl  status  nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Sun 2024-12-08 02:08:53 UTC; 37min ago
    Process: 98070 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 98071 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 98072 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 98073 (nginx)
      Tasks: 2 (limit: 1111)
     Memory: 2.3M
        CPU: 34ms
     CGroup: /system.slice/nginx.service
             ├─98073 "nginx: master process /usr/sbin/nginx"
             └─98074 "nginx: worker process"
```

Paso 2. instalamos las herramientas docker

```
[root@ip-10-0-4-81 ec2-user]# yum install docker
Last metadata expiration check: 21:11:45 ago on Sat Dec  7 04:24:46 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                  Repository     Size
================================================================================
Installing:
 docker                  x86_64   25.0.6-1.amzn2023.0.2    amazonlinux    44 M
Installing dependencies:
 containerd              x86_64   1.7.23-1.amzn2023.0.1    amazonlinux    36 M
 iptables-libs           x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   401 k
 iptables-nft            x86_64   1.8.8-3.amzn2023.0.2     amazonlinux   183 k
 libcgroup               x86_64   3.0-1.amzn2023.0.1       amazonlinux    75 k
 libnetfilter_conntrack  x86_64   1.0.8-2.amzn2023.0.2     amazonlinux    58 k
 libnfnetlink            x86_64   1.0.1-19.amzn2023.0.2    amazonlinux    30 k
 libnftnl                x86_64   1.2.2-2.amzn2023.0.2     amazonlinux    84 k
 pigz                    x86_64   2.5-1.amzn2023.0.3       amazonlinux    83 k
 runc                    x86_64   1.1.14-1.amzn2023.0.1    amazonlinux   3.2 M
```

Paso 2.1. Iniciamos docker con los siguientes comandos:

```
[root@ip-10-0-4-81 ec2-user]# systemctl start docker
[root@ip-10-0-4-81 ec2-user]# systemctl enable  docker
```

Paso 2.2. Descargamos del repositorio la imagen httpd para docker:

```
[root@ip-10-0-4-81 ec2-user]# 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:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

Paso 2.3. Revisamos que la imagen esta descargada con éxito

```
[root@ip-10-0-4-81 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    494b2b45fd74   4 months ago   147MB
```

Paso 2.4. Creamos tres carpetas para tomar de origen, en esta ubicación estarán los archivos index.html de la aplicación

```
drwxrwxrwx.   2 root root    24 Dec  8 01:59 tienda1
drwxrwxrwx.   2 root root    24 Dec  8 02:00 tienda2
drwxrwxrwx.   2 root root    24 Dec  8 02:00 tienda3
```

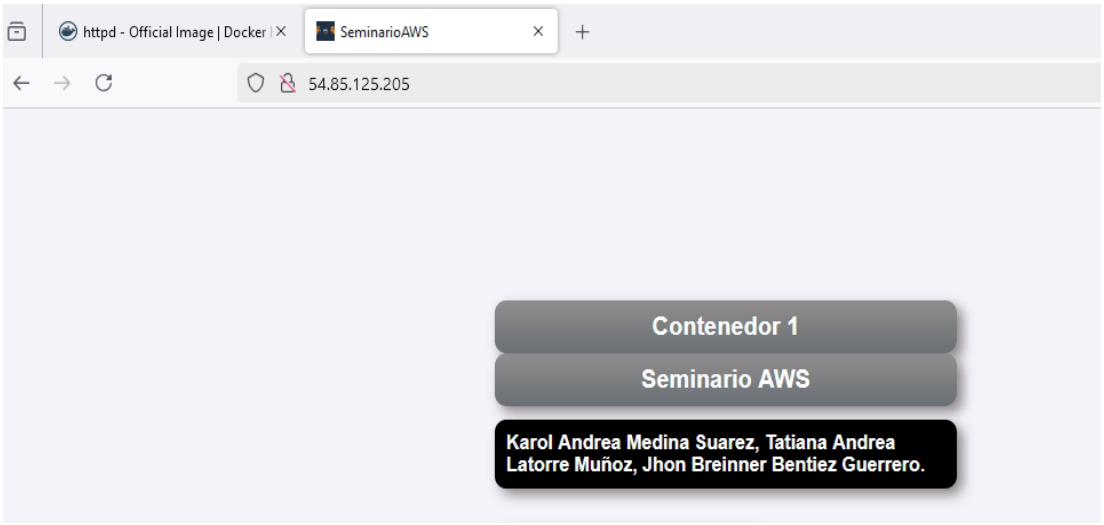
Paso 2.5. Creamos los contenedores con los siguientes comandos:
docker run -dit --name tienda1 --restart always -p 8080:80 -v /tienda1:/usr/local/apache2/htdocs/ httpd
docker run -dit --name tienda2 --restart always -p 8082:80 -v /tienda2:/usr/local/apache2/htdocs/ httpd
docker run -dit --name tienda3 --restart always -p 8083:80 -v /tienda3:/usr/local/apache2/htdocs/ httpd
y luego ejecutamos **docker ps** para visualizar los contenedores

```
[root@ip-10-0-4-81 /]# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED              STATUS              PORTS                                       NAMES
bae240f25318   httpd     "httpd-foreground"   39 seconds ago       Up 37 seconds       0.0.0.0:8083->80/tcp, :::8083->80/tcp       tienda3
34136c1f86c8   httpd     "httpd-foreground"   About a minute ago   Up About a minute   0.0.0.0:8082->80/tcp, :::8082->80/tcp       tienda2
62001e863cfb   httpd     "httpd-foreground"   3 minutes ago        Up 3 minutes        0.0.0.0:8080->80/tcp, :::8080->80/tcp       tienda1
```

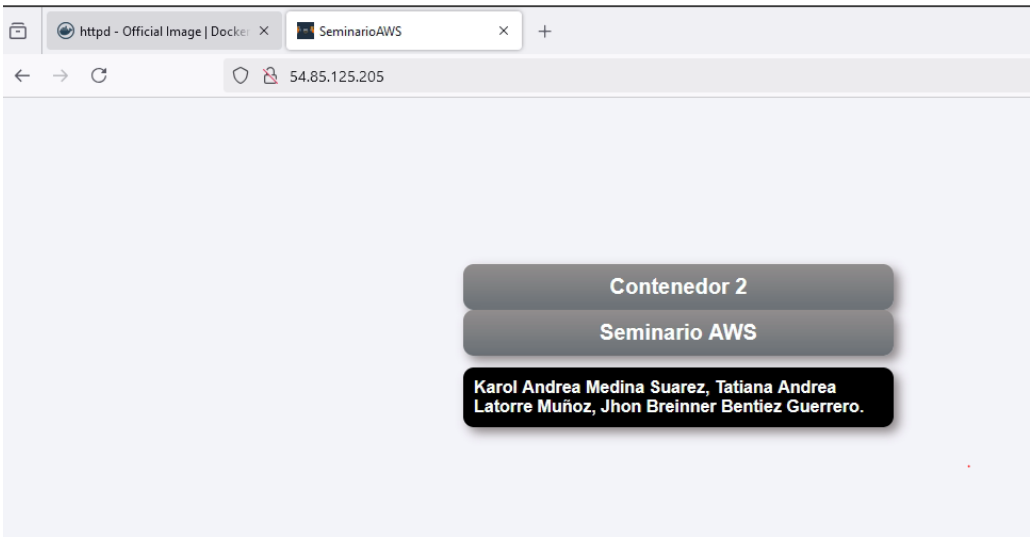
Paso 3. ingresamos a la ruta **/etc/nginx/nginx.config** y abrimos el archivo **nginx.config** con el comando **nano nginx.config** donde realizamos la configuración del servicio proxy para cuando llegue las peticiones ser reenviado a los contenedores

```
[root@ip-10-0-4-81 nginx]# cat  nginx.conf
events {}

http {
        upstream backend {
                server localhost:8080;
                server localhost:8082;
                server localhost:8083;
        }
        server {
                listen 80;
                server_name nginx;
                location / {
                        proxy_pass http://backend;
                        }
        }


}
```

Paso 4. Por ultimo reiniciamos el nginx y cada petición que se hace a la instancia será enviada a los contenedores:
Primera petición:

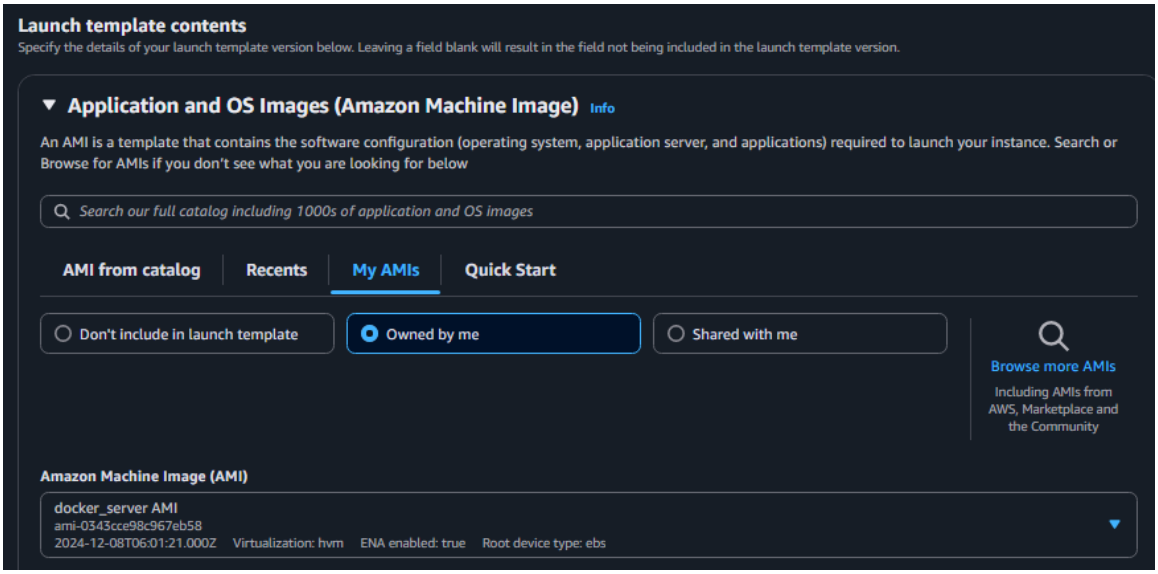


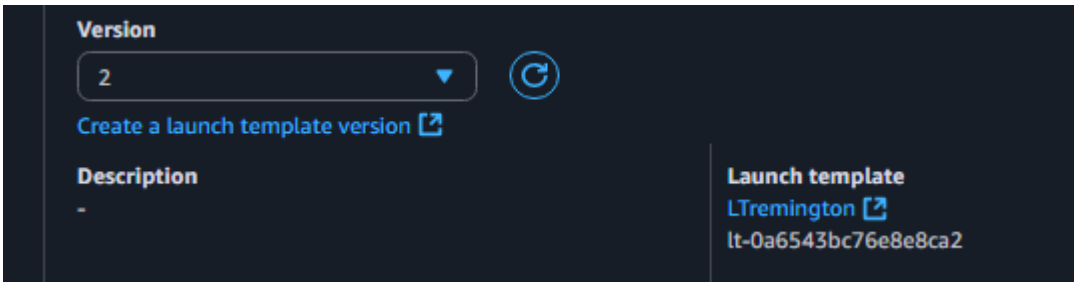
Segunda petición:



Tercera petición:

Paso 5. Creamos una snapshots de la instancia implementada con docker, creamos una AMI nueva y la remplazamos en el auto scaling groups

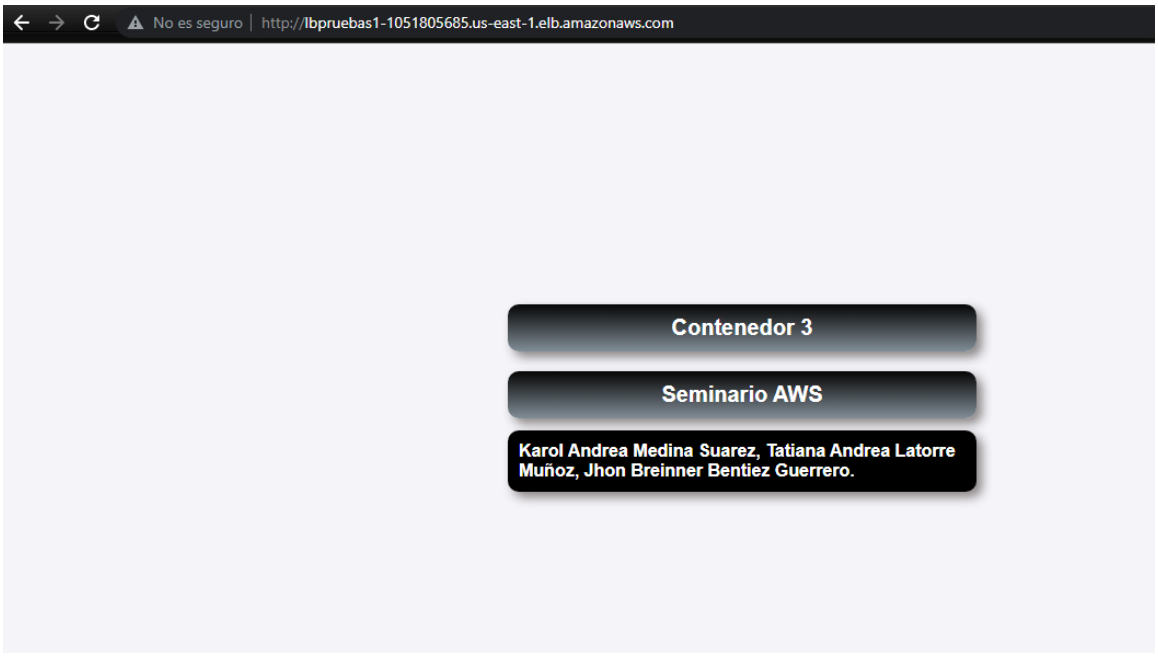




Paso 6. Funcionamiento con toda la infraestructura creada


No es seguro | http://lbpruebas1-1051805685.us-east-1.elb.amazonaws.com
Contenedor 1
Seminario AWS
Karol Andrea Medina Suarez, Tatiana Andrea Latorre Muñoz, Jhon Breinner Bentiez Guerrero.

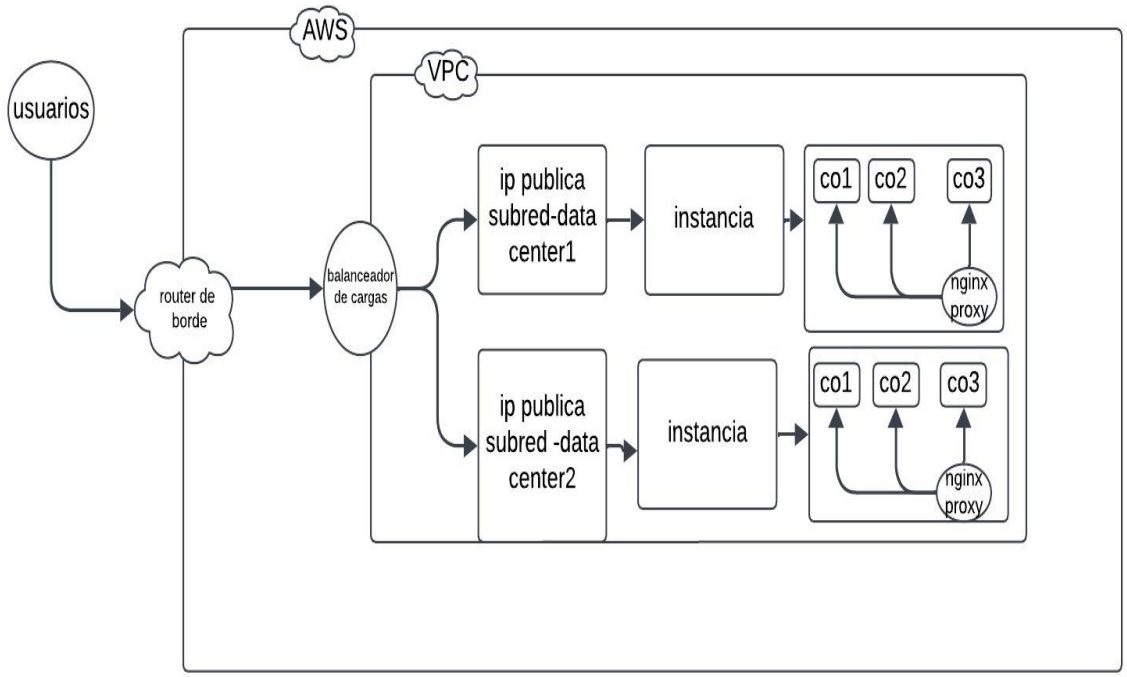

No es seguro | http://lbpruebas1-1051805685.us-east-1.elb.amazonaws.com
inglés
español
Google Translate
Contenedor 2
Seminario AWS
Karol Andrea Medina Suarez, Tatiana Andrea Latorre Muñoz, Jhon Breinner Bentiez Guerrero.



Vinculo YouTube-explicaicon visual

https://youtu.be/kBClj_-C-8M

Diagrama de recursos usados y como se comunican entre ellos.



Conclusiones

La implementación de una arquitectura basada en contenedores en AWS, junto con el uso de balanceadores de carga, hace que sus aplicaciones lleguen a la meta de las empresas, si usted está buscando un modo más eficiente de escalar su aplicación que no requiera hardware físicos adicionales; estas formas de implementar la infraestructura permiten a los desarrolladores centrarse en importantes aspectos técnicos de codificación, despliegue y marketing. Es decir, los usos de micro-servicios en contenedores dejan claro que cada componente de la aplicación puede ser escalado sin que haya daños internos o causas externas.

Cuando se introduce en AWS una arquitectura con balanceadores de carga y contenedor, se consigue que las aplicaciones sean más flexibles y estén disponibles durante más tiempo. De esta forma la empresa fast food obtendrá seguridad de su software que estará completamente funcional. Sí una instancia deja de trabajar, el balanceador dirigirá automáticamente el tráfico hacia otras máquinas sanas que iniciaran automáticamente minimizando al mínimo tiempo de inactividad.

Referencias

(s.f.). Obtenido de https://www.docker.com/blog/lxc-vs-docker/

Amazon web services . (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html

Amazon Web Services . (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/ebs/latest/userguide/what-is-ebs.html

Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/Instances.html

Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html

Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/vpc/latest/userguide/configure-subnets.html

Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html

Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/Storage.html

Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

Perlow, J. (13 de junio de 2024). *www.docker.com*. Obtenido de www.docker.com: https://www.docker.com/blog/lxc-vs-docker/