**TRABAJO DE GRADO**
**Seminario**

# Gestores de infraestructura para la Industria JMS

Corporación Universitaria Remington.
Ingeniería de sistemas
Seminario de Amazon Web Services

Miguel Angel Gomez ladino
Juan Andrés Naranjo Celis
Santiago Mosquera Mosquera

Docente:
Juan Pablo Berrio López
Seminario
2024

# Tabla de Contenidos

**Resumen**

En este trabajo se conocerá el despliegue de una infraestructura básica por medio de los servicios de AWS que nos ofrece una infraestructura en la nube escalable, segura y flexible permitiéndole satisfacer a nuestra empresa sus necesidades operativas actuales y futuras, es allí donde se implementarán la creación de VPC, instancias, Amis, balanceadores de carga, auto scaling y componentes para un startup llamada Industrias JMS donde ellos buscan dar un gran paso estratégico hacia la modernización en la tecnología y la optimización de sus recursos. Ya que es una plataforma innovadora que conecta a restaurantes con clientes mediante entregas muy rápidas y ahora últimamente la empresa ha experimentado un rápido crecimiento en los últimos meses y ahora necesita escalar en su infraestructura tecnológica para poder manejar una mayor demanda en su mercado, y así ellos puedan garantizar la disponibilidad y mejorar en su tiempo de respuesta que han sido afectados por dicho crecimiento y han perdido valiosos clientes, es allí donde nosotros queremos brindar una solución altamente disponible, escalable y que este de la mejor manera diseñada para poder manejar la gran cantidad de trafico de una manera más eficiente y hacer que la empresa pueda seguir creciendo y establecer sus servicios de una manera más productiva y eficiente, es por eso que por medio de las herramientas que nombramos anteriormente daremos inicio al proceso de creación de su infraestructura ya que queremos que la empresa reduzca costos a la hora de esta implementación.

**Palabras clave:** Escalabilidad, automatización, seguridad, sostenibilidad y eficiencia

## Marco conceptual y contextual

En este apartado realizaremos la descripción de los conceptos que implementamos a lo largo del proyecto para (Gestores de infraestructura para la industria JMS) que tenía problemas con el tráfico en la red para ello colocaremos algunos conceptos para que a lo largo de la lectura se pueda entender la implementación y creación de estos servicios.

Conceptos básicos:

**VPC**.

Una VPC es un servicio que nos permite crear una red privada virtual en la infraestructura de AWS, dentro de una VPC podremos implementar varias subredes que pueden ser públicas o privadas.

**EC2**.

Elastic compute cloud es un servicio de computación en la nube proporcionado por AWS, que le permite a los usuarios ejecutar servidores virtuales. Este servicio proporciona capacidad de procesamiento escalable y flexible en la nube, de manera que los usuarios que lo implementan solo pagan por lo que usan

**ELB**.

Elastic Load Balancer es un servicio de balanceo de carga gestionado por AWS que se encarga de distribuir el tráfico de red o solicitudes de los usuarios entre múltiples instancias para garantizar la disponibilidad y el rendimiento de las aplicaciones.

**Snapchots**.

Sin copias de seguridad que se le realizan a los volúmenes de almacenamiento, los snapshots permiten capturar el estado de un volumen en un momento especifico, también

se utilizan para proteger los datos, lo que asegura que los datos sean protegidos y puedan restaurarse rápidamente en el caso de que haya alguna perdida.

**EBS**.

Elastic block store es un servicio de almacenamiento en bloque de AWS este servicio nos permite crear volúmenes de almacenamiento persistente que se puede adjuntar a las instancias EC2, ofreciendo almacenamiento de alta disponibilidad y rendimiento para aplicaciones y datos. Además, permite realizar copias de seguridad mediante snapshots.

**Auto scaling**.

En AWS el auto scaling es una herramienta poderosa para gestionar y optimizar la infraestructura en la nube de manera dinámica, ajustando la cantidad de instancias EC2 basadas en la demanda de tráfico. Esto proporciona beneficios a mayor disponibilidad y mejora en el rendimiento de las aplicaciones, todo realizado de manera automática.

**Nginx**.

Es un servicio web, proxi inverso y balanceador de carga de alto rendimiento, que se utiliza para gestionar el tráfico de web y optimizar el rendimiento de las aplicaciones. En el contexto AWS nginx se puede mejorar la escalabilidad, la seguridad y el rendimiento de las aplicaciones que se ejecutan en la nube de amazon

**Docker**.

Es una herramienta que nos permite ejecutar aplicaciones en contenedores de forma eficiente dentro de la infraestructura de AWS también es distribuible dentro de las instancias permitiendo así que su uso sea portable y fácil de escalar.

## Desarrollo e implementación del aprendizaje

Con respecto a las industrias JMS obtuvimos buenos resultados a la hora de su implementación permitiendo darle más escabilidad y seguridad a su programa de peticiones gracias a la ayuda de los servicios de AWS, como lo son las vpc, instancias, auto scaling, balanceador de carga y componentes que fueron de vital importancia para la implementación de esta infraestructura.

### ¿Cuáles son los recursos que permiten ejecutar instancias en AWS?

**Amazon EC2**

Este recurso es uno de los más importantes ya que nos permite dar la creación de las máquinas virtuales en la nube de AWS, permitiendo administrar y configurar las instancias de hardware (tipo de instancia), y el sistema operativo que nos va a permitir tener un perfecto uso en nuestra máquina virtual.

**Amazon machine Image**

Este recurso son imágenes que contienen nuestro sistema operativo predeterminadamente y el software que se va a trabajar para el lanzamiento de la instancia EC2, Por otro lado, nos permite crear imágenes personalizadas para nuestro software.

**Elastic Ips**

En este recurso vamos a ver todo lo que tiene que ver con el direccionamiento de Ip estática, lo cual nos permite asociarnos a una instancia que siempre va a tener la misma, incluso si se apaga o se reinicia nuestra instancia.

**VPC**

El VPC es una red que nos permite crear un entorno aislado, donde podemos almacenar varias de nuestras instancias y tener acceso a ellas, mediante configuraciones de seguridad y redes.

**Amazon elastic block store (EBS)**

Por ultimo y menos importante este recurso nos permite guardar el almacenamiento de la instancia EC2, de una manera confiable y segura, incluso si la instancia se apaga o se reinicia.

## Pasos para la creación de un servicio web de Linux

En este apartado daremos una breve introducción de los pasos a realizar, para la creación de un servicio de Linux, explicando de una manera clara y concisa para que el lector tenga un mejor entendimiento de la implementación de este servicio.


***Figura 1.*** *Crear del VPC.*

Para la creación del VPC seleccionamos la casilla VPC y más esto lo hacemos para la creación únicamente del recurso, en generación automática de etiquetas de nombre, colocamos el nombre de la VPC que vamos a utilizar, dejamos la ipV4 por defecto que nos arroja el sistema y nos regala el tamaño de las Ips que podemos utilizar. También

seleccionamos la casilla sin bloque de IpV6, ya que en este trabajo solo vamos a utilizar la IpV4, por otro lado, seleccionamos en el apartado de tenencia predeterminado(default).

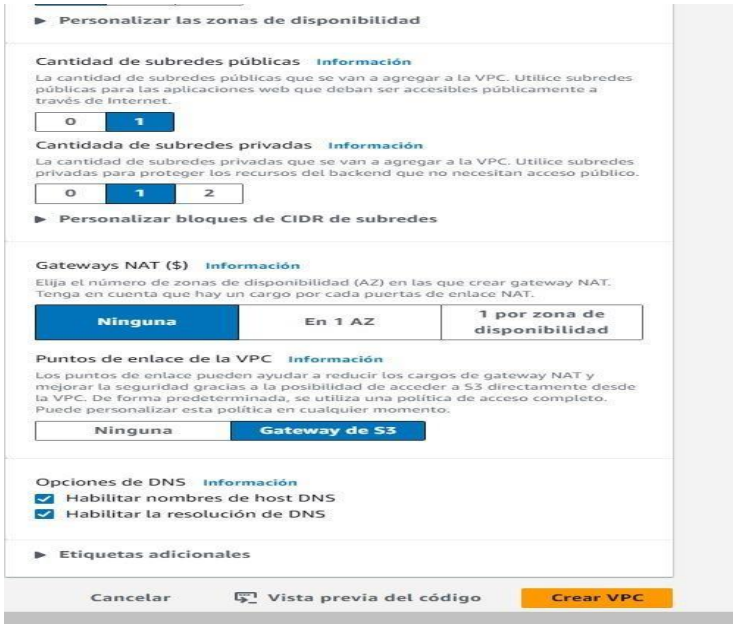



***Figura 2.*** *Crear del VPC.*

En esta figura seguimos con la configuración de la zona de disponibilidad, en este caso seleccionamos 1 ya que es suficiente para lo que necesitamos, Por otro lado, seleccionamos la cantidad de redes públicas y privadas en este caso también ponemos 1, porque es suficiente para lo que necesitamos ya que es un proyecto que no se necesita más recursos, el Gateway lo ponemos en ninguno ya que solo es un router para redes privadas, por ultimo le damos en el botón crear VPC y se creara esta red.

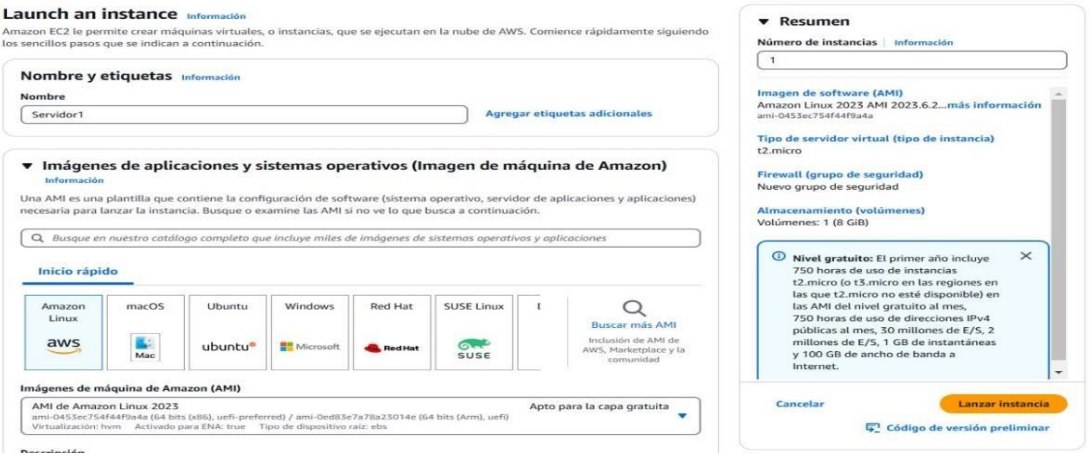



***Figura 3.*** *Creación de la instancia.*

En esta figura vamos a realizar la instancia, primero damos un nombre a la máquina virtual, seleccionamos el sistema operativo en el cual vamos a trabajar en este caso Amazon Linux, pero se puede seleccionar cualquiera. En imágenes de la maquina se puede seleccionar cualquier versión, para nuestro proyecto vamos a elegir todas las versiones gratis en este caso AMI de Amazon Linux 2023, ya que existen también versiones pagas.

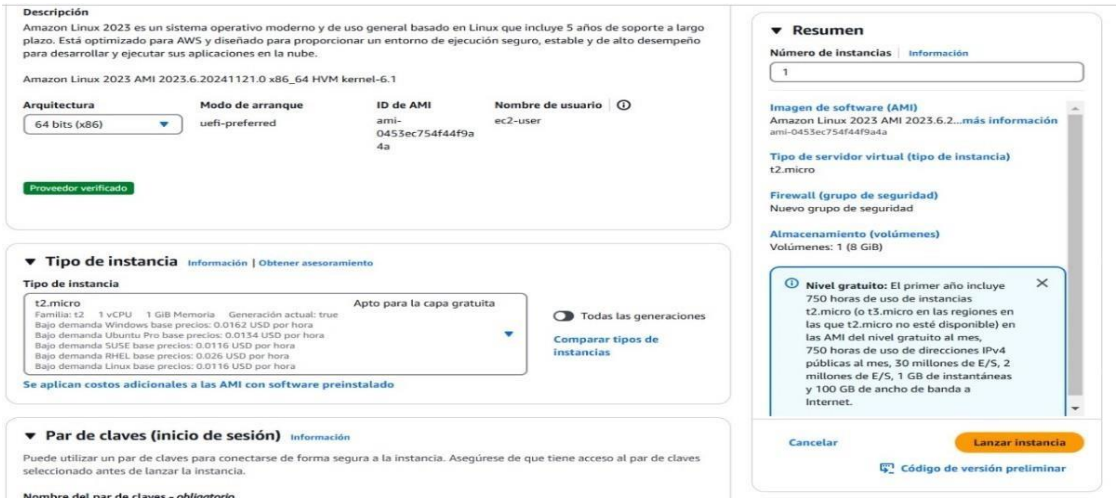



***Figura 4****. Tipo de la instancia.*

En este paso seleccionamos la arquitectura del sistema operativo ya sea de 64 o 32bits, para este ejercicio 64bits, después en el tipo de instancia seleccionamos las características del hardware, en este caso t2. micro ya que es gratuito. Vamos a generar unas claves de inicio para ingresar al servidor.

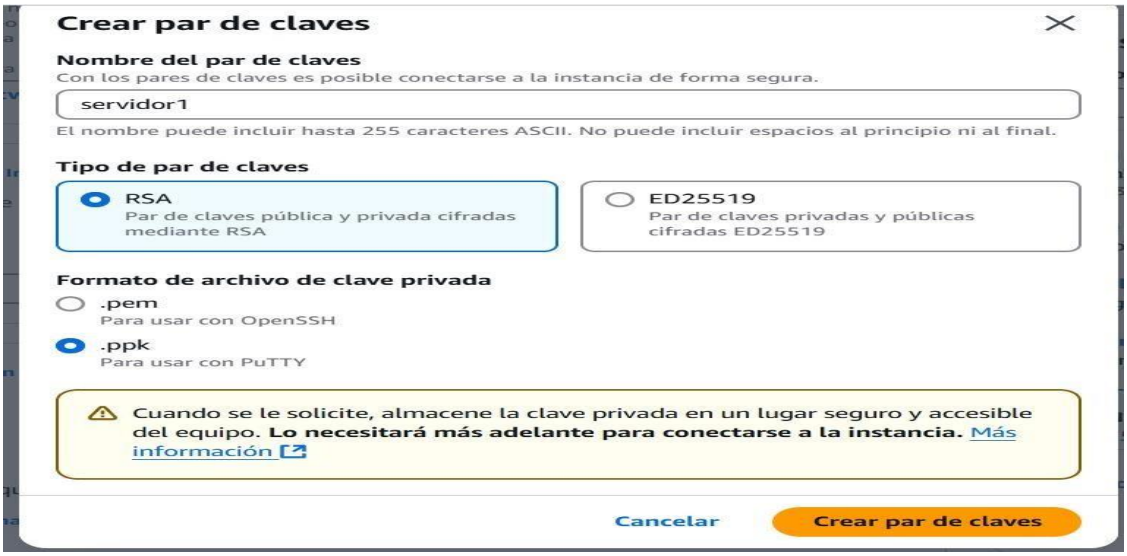



***Figura 5.*** *Creación de la instancia y llave.*

Para crear la llave debemos colocar el nombre del servidor, después seleccionamos el tipo de claves que es RSA, por último, seleccionamos el formato. ppk que es la extensión para Linux, después le damos crear y nos genera un archivo para guardar, tener en cuenta donde guardan este archivo ya que con esto iniciaremos la conexión al servidor por medio del putty.

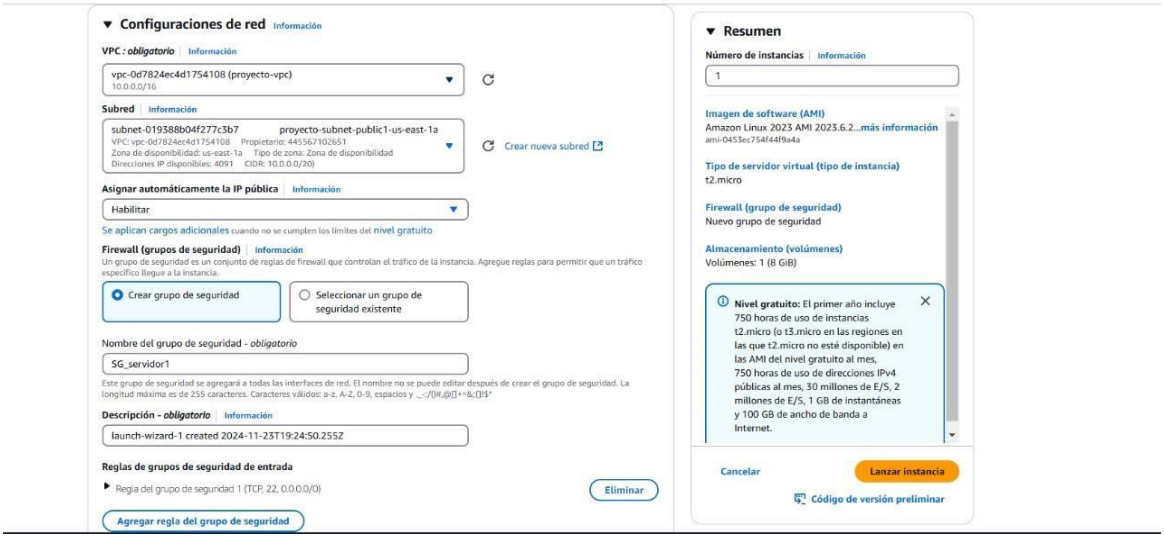



***Figura 6.*** *Creación de la instancia y configuración de red.*

En la configuración de red comenzamos insertando la VPC que creamos al principio, el automáticamente nos arroja las subredes. En asignar automáticamente la ip publica le damos habilitar y en el firewall seleccionamos la casilla crear grupo de seguridad,

Asignamos un nombre al grupo de seguridad, se puede seleccionar cualquier nombre en este caso SG_servidor1 y el automáticamente genera una descripción.

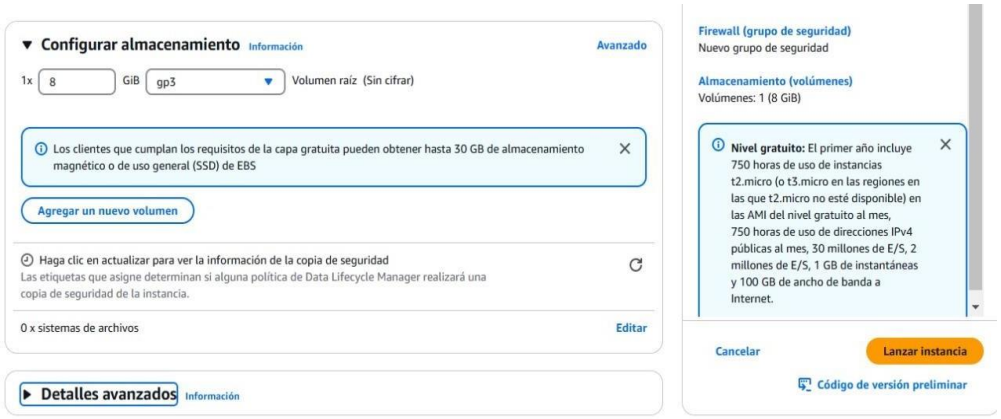



***Figura 7***. *Creación de la instancia y configuración de almacenamiento*

En configuración de almacenamiento seleccionamos la cantidad que vamos a utilizar para este caso son 8GB, pero se puede colocar más almacenamiento o menos, dependiendo de la necesidad del servidor y de los requisitos pagos o gratuitos.

Por último, le damos en lanzar instancia y nos crea nuestro servicio, después de tener creado nuestro servicio procedemos a utilizar la llave que creamos en la instancia en el emulador del terminal en este caso utilizando putty.

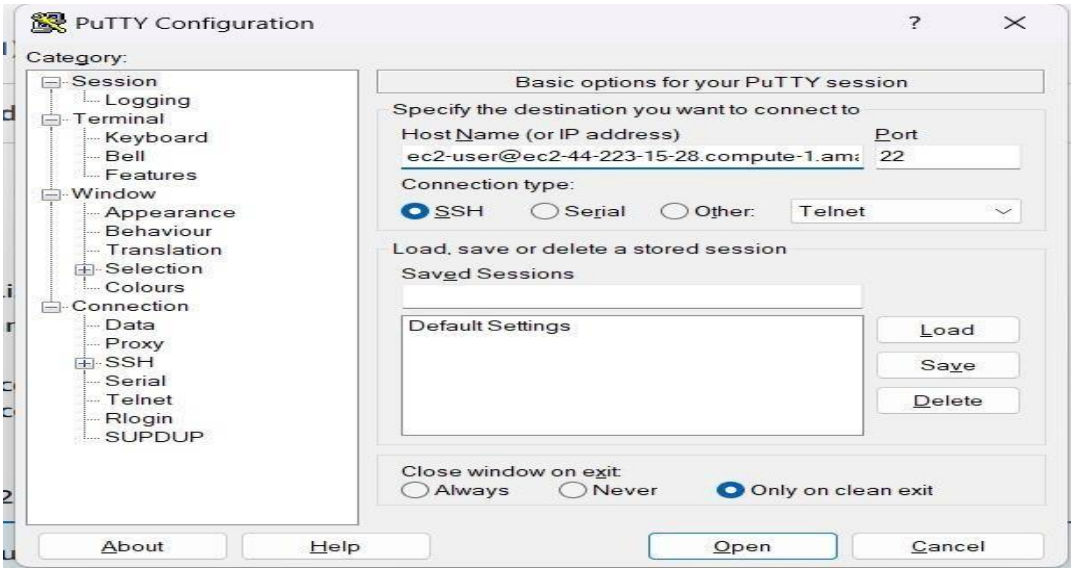



***Figura 8.*** *Emulador de terminal (Putty).*

Al iniciar el emulador ingresamos el usuario que nos arroja la instancia, colocamos el

puerto 22 ya que es el puerto para Linux, seleccionamos en el apartado izquierdo el protocolo (SSH).

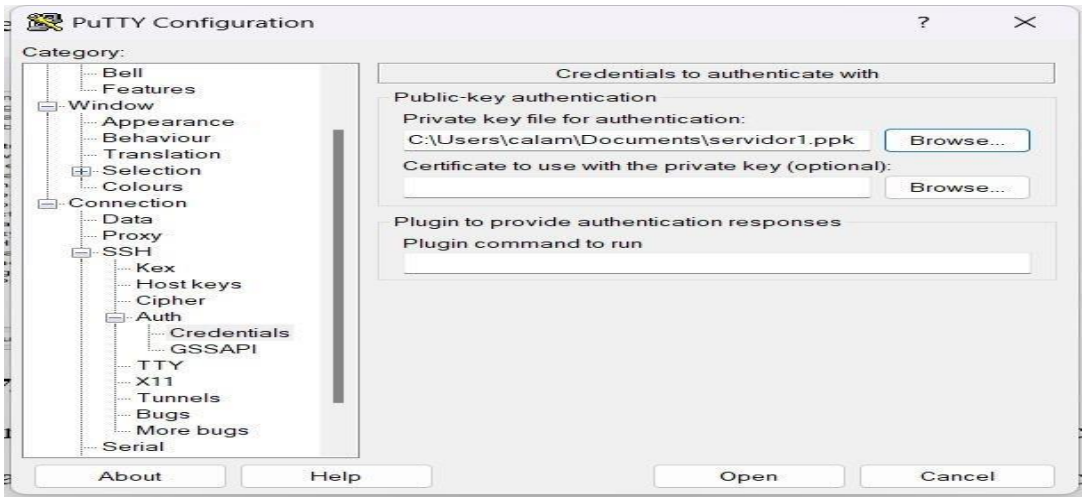



***Figura 9.*** *Emulador de terminal (Putty).*

Al seleccionar el protocolo SSH, le damos en Auth, volvemos a dar clic en las credenciales y nos va aparecer la opción donde vamos a seleccionar la llave, damos clic en browser y buscamos la llave que creamos en la instancia que es la. ppk, para permitir la conexión con el servidor.

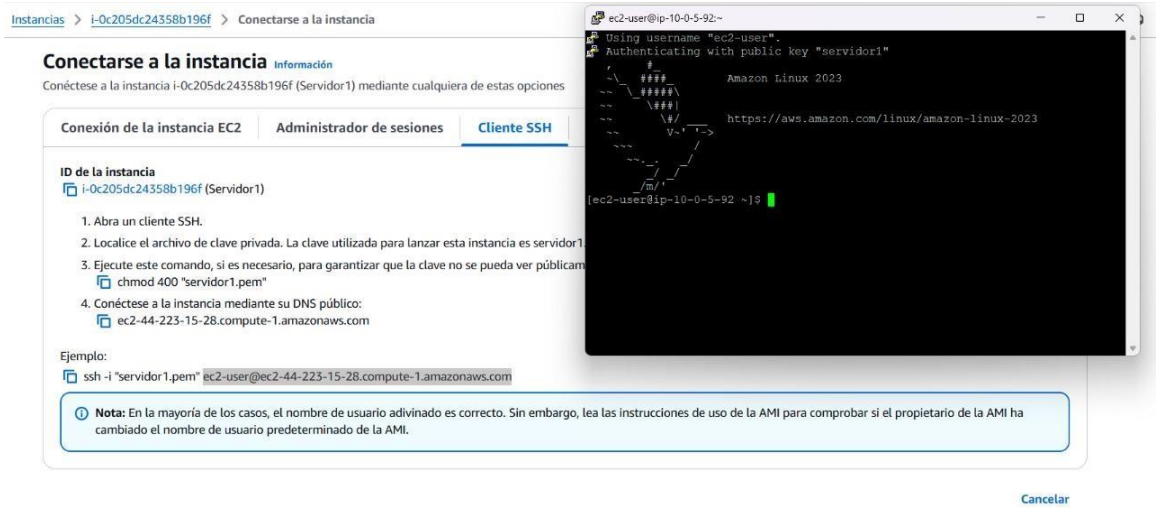



***Figura 10.*** *Terminal del servidor web.*

Una vez seleccionada la llave le damos en open para que se nos abra la terminal de nuestro servidor.

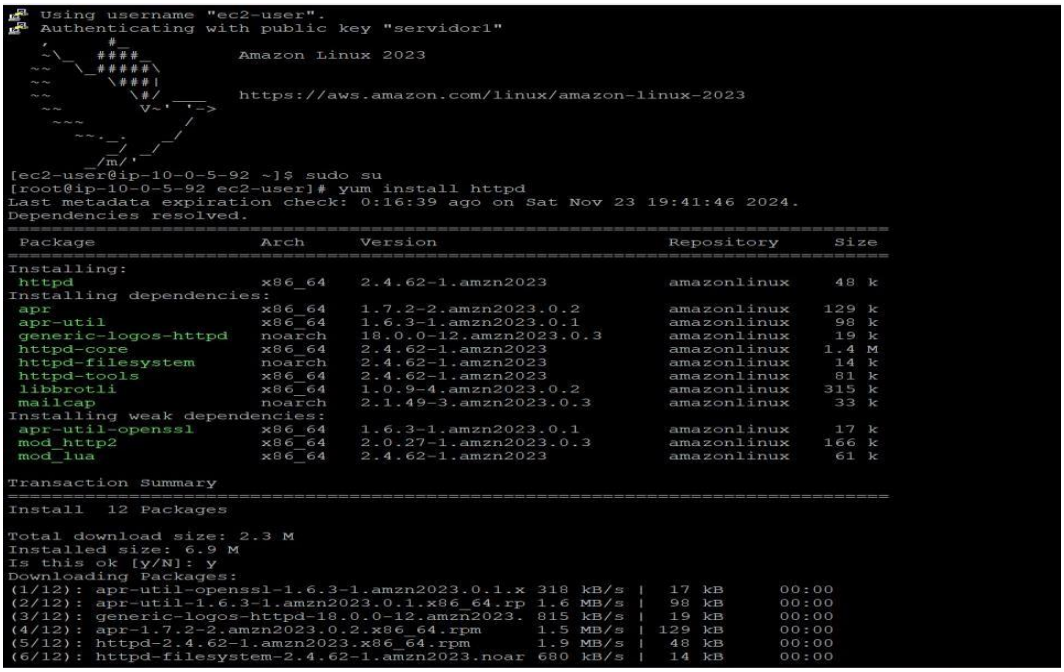



***Figura 11***. *Instalación de paquetes para el servicio web.*

Una vez dentro del terminal ingresamos como administrador con el comando Sudo su, después de entrar como administrador instalamos el paquete de servicios web con el comando yum install httpd y automáticamente empieza a descargar e instalar los recursos

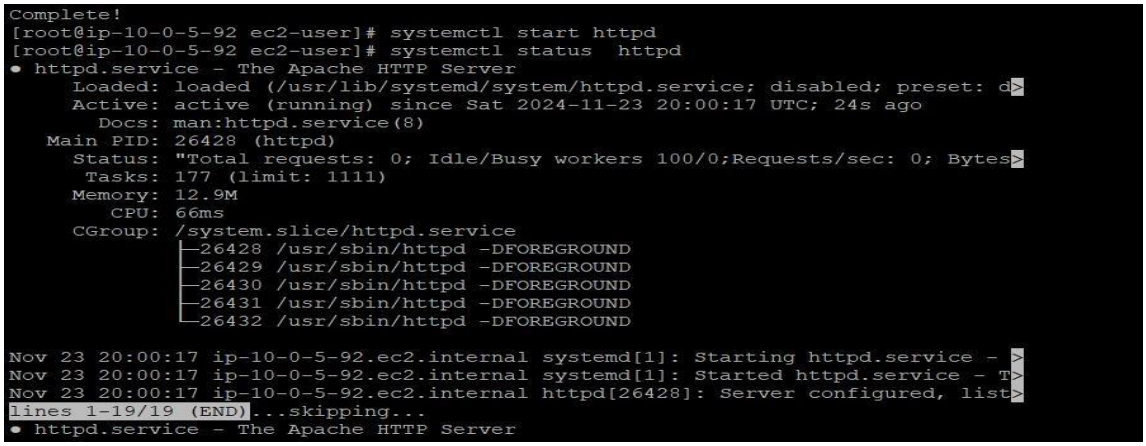



***Figura 12***. *Instalación de paquetes para el servicio web.*

Después de haber instalado el servicio web, iniciamos el servicio con el comando systemctl start httpd y por último verificamos el estado con el comando systemctl status httpd, como se puede ver en la imagen esta activo.

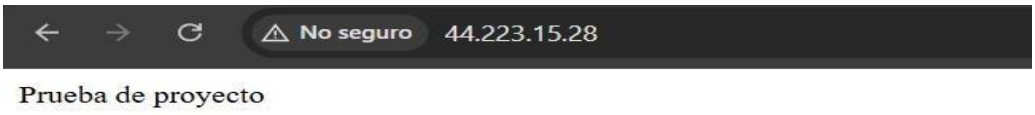



***Figura 13***. *Funcionamiento servidor.*

Una vez hecho lo anterior colocamos la dirección ip que nos da la instancia en un navegador y le damos enter y nos carga el servidor, en este caso creamos un archivo con el comando nano index.html, el cual nombramos prueba de proyecto.

**Enlaces de video.**

Enlace YouTube calidad deficiente.

https://youtu.be/YEVoT5lnlYo?si=OgHAsbFdu5i-q0AM

Enlace Drive Buena Calidad

https://drive.google.com/file/d/1mrNTUHJIZ7nFPps_-RSFwg7NCv1Sdtl1/view?usp=sharing

**Creación de una imagen privada mediante snapchots en AWS**

En este apartado explicaremos cómo crear una imagen privada utilizando snapshots en AWS. Este método nos permitirá, en el momento que deseemos o necesitemos lanzar nuevas instancias seleccionar una imagen preconfigurada que ya incluye el servidor montado. En este caso, crearemos una imagen basada en Amazon Linux que ya tendrá

configurado un servidor web. Esto facilita el trabajo empresarial al optimizar tiempos y garantizar consistencia en las configuraciones.

Es importante destacar que para realizar un snapshot, primero se debe contar con un servidor previamente montado y configurado. Este snapshot servirá como base para crear la Amazon Machine Image (AMI), la cual se utilizará para instanciar nuevos servidores de manera más eficiente.

**¿Pasos para la creación de una imagen utilizando snapshots?**

Se realizarán los siguientes pasos para la creación de la imagen de Amazon Linux a través de pantallazos para comprender los pasos a realizar.

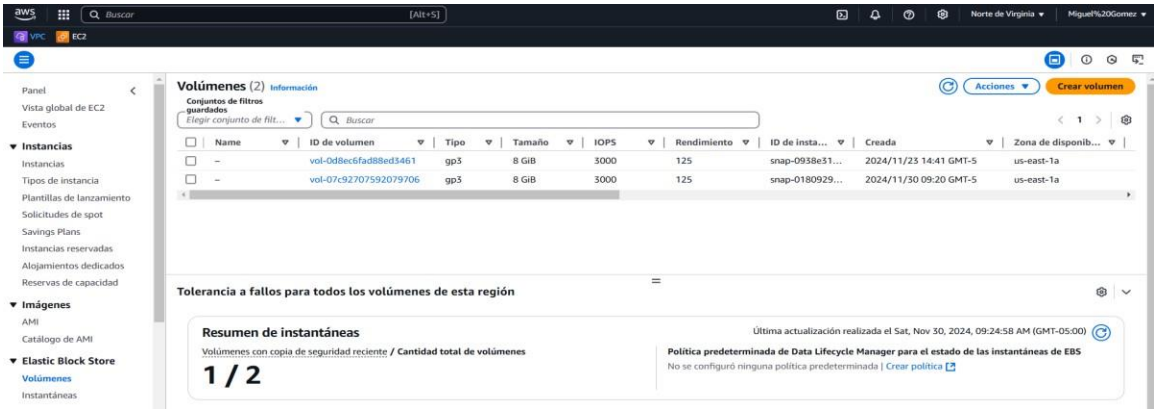



***Figura 1***. *Menú de snapshots.*

Para la creación del snapshot debemos ubicarnos en el menú y buscar Elastic Block Store y dar clic en volúmenes.

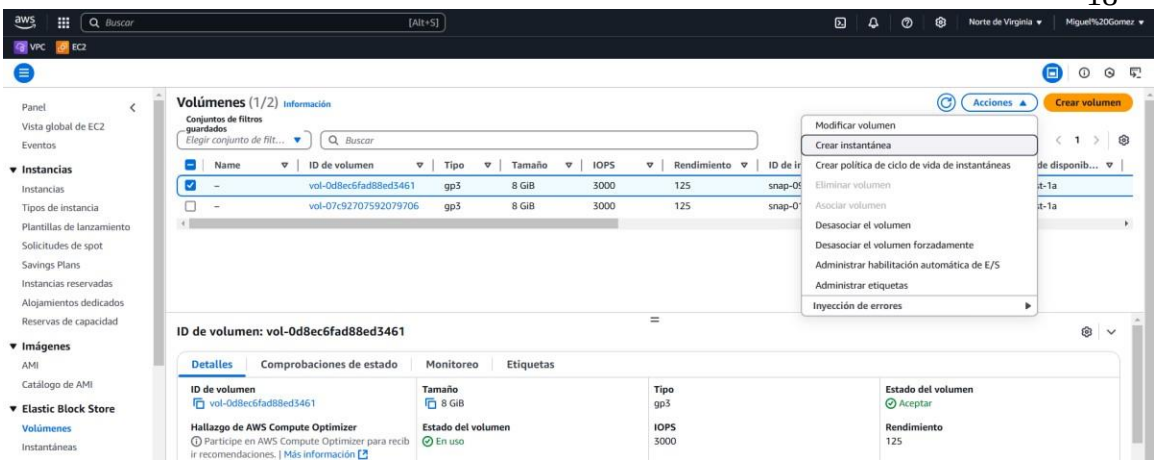



***Figura 2.*** *creación del snapshot.*

*En este paso se seleccionará la casilla del volumen debajo del name, para que aparezca elchulito azul y podamos seleccionar crear instantánea(snapshot).*

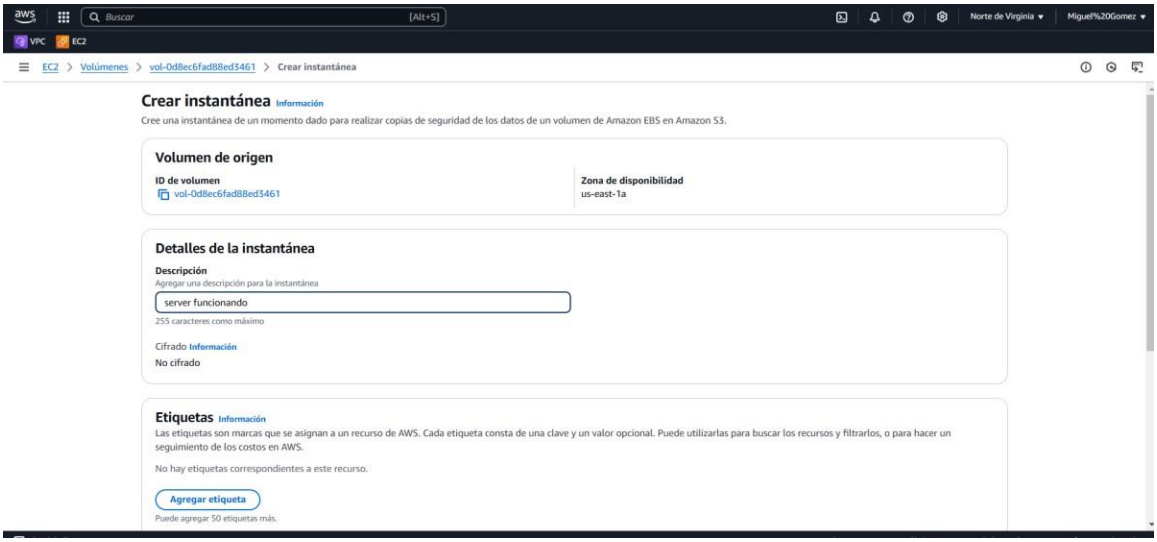



***Figura 3.*** *creación del snapshot.*

Después de realizar el siguiente paso, nos va a salir el siguiente apartado en el cual colocaremos el nombre del detalle de la instantánea y se procederá a dejar el resto de la configuración por defecto y daremos clic en el botón de crear instantánea(snapshot) y allí podremos evidenciar que fue creada correctamente.

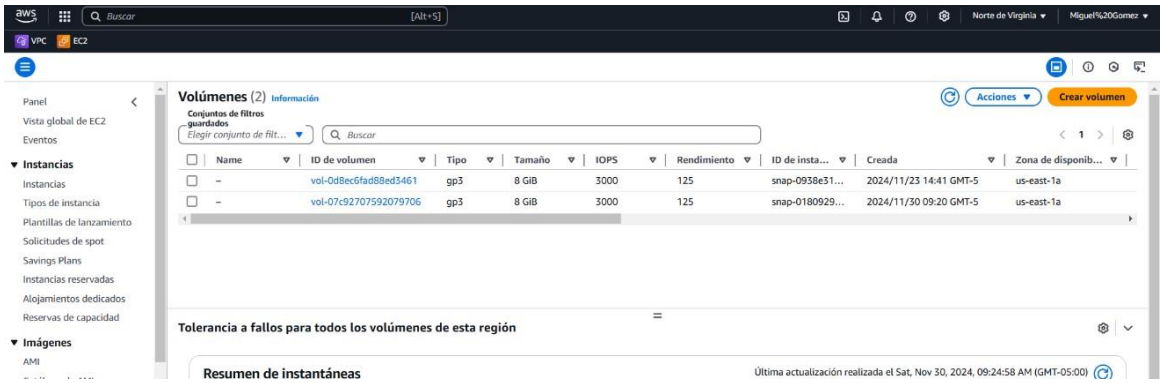



***Figura 4.*** *Verificación de snapshot creado.*

Por último, podremos evidenciar que fue creado correctamente el snapshot, como principalmente ya había sido creado, por ende, aparecen los dos snapshots en el primer paso, Si aún no han creado los snapshots, al principio debería aparecerles un único paso que incluya la creación de los snapshots que acaban de realizar.

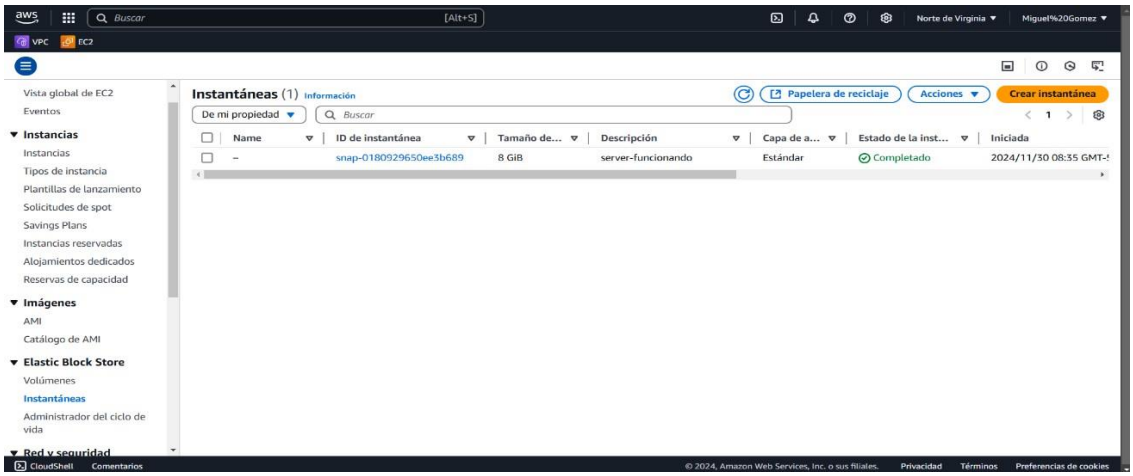



***Figura 5.*** *Menú de la AMI.*

Para la creación de la AMI debemos situarnos en el menú y buscar Elastic Block Store y dar clic en instancias

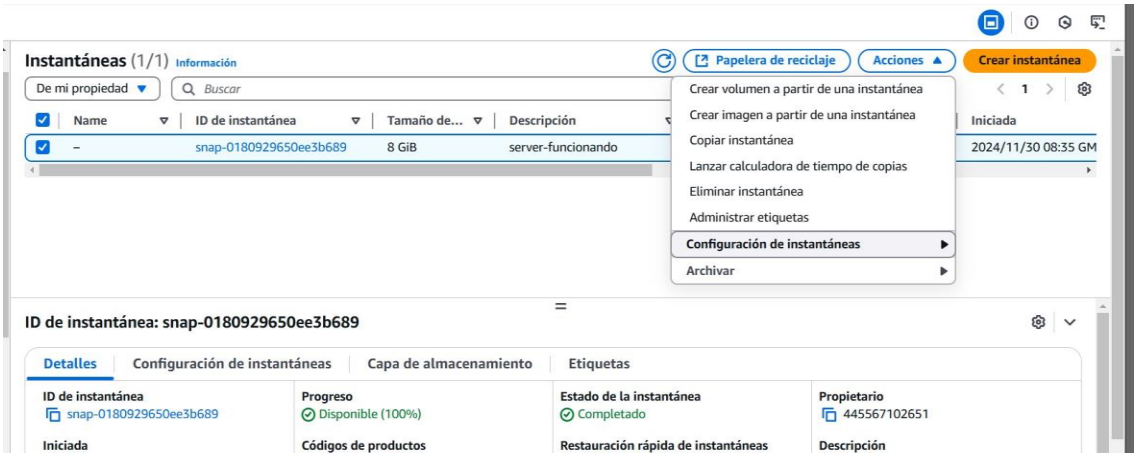



***Figura 6.*** *Configuración de la AMI.*

Luego seleccionamos la casilla de la instancia en el cuadro debajo del name hasta que aparezca en azul, después en la parte superior en acciones damos clic donde se nos permite seleccionar crear una imagen a partir de una instantánea.

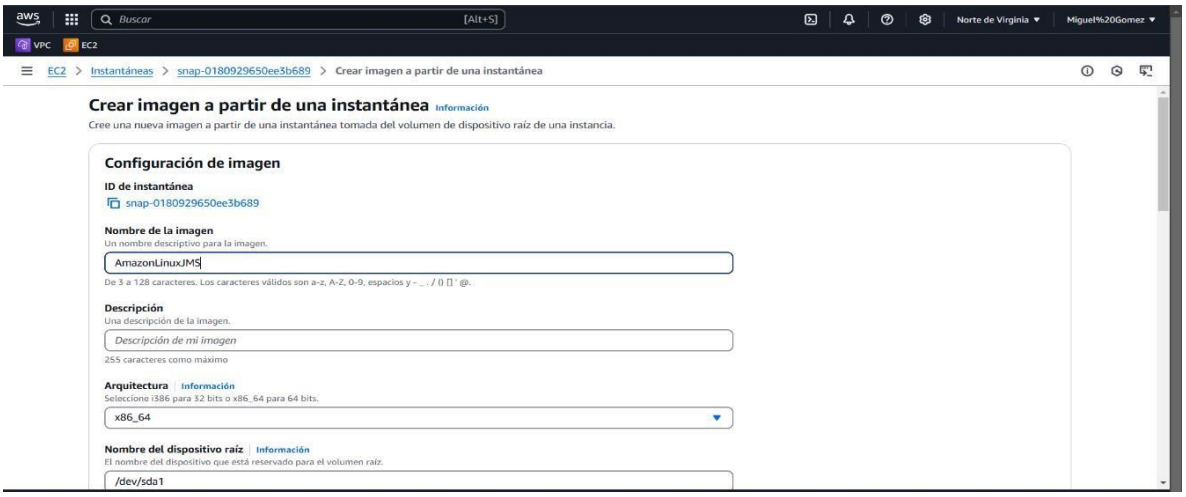



***Figura 7****. creación de la AMI*

En este paso colocaremos el nombre de como queremos que se llame la imagen, y escogeremos la arquitectura que sea requerida, en este caso colocaremos todo por defecto y en la casilla descripción algo breve.

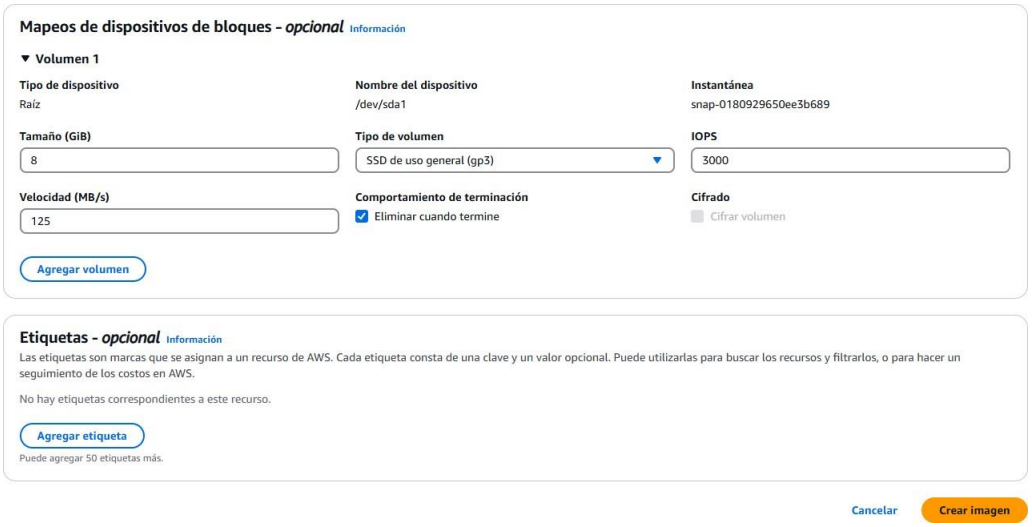



***Figura 8.*** *Configuración de almacenamiento*

En este paso también dejaremos todo por defecto y daremos clic en crear la imagen al realizarlo podremos observar en el siguiente paso que fue creada la imagen de manera correcta.

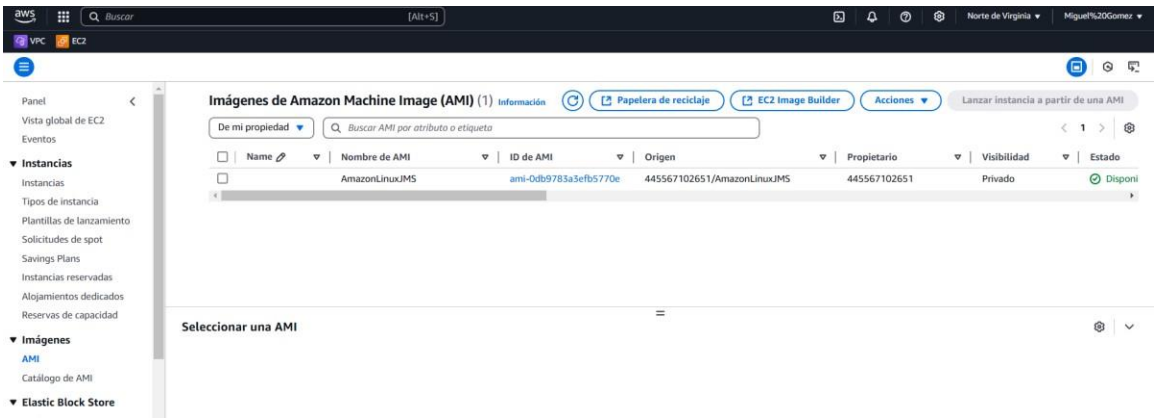



***Figura 9****. creación de la AMI*

Para verificar que la AMI haya sido creada correctamente nos dirigimos nuevamente al menú, y en el apartado de imágenes, veremos una opción llamada AMI, al hacer clic ahí podremos observar que al lado derecho aparece creada nuestra AMI con el respectivo nombre que le colocamos al inicio.

## Que es un balanceador de carga

Para empezar este apartado empezaremos hablando sobre los balanceadores de carga ya que es un componente importante ya que actúa entre los usuarios y los servidores que procesan solicitudes, su propósito es distribuir el tráfico de solicitudes de una manera equitativa entre varios servidores, optimizando el rendimiento y garantizando un mejor uso de la disponibilidad y previniendo sobre cargas de nuestros servidores.

### ¿Pasos para la creación de un balanceador de carga?

A continuación, explicaremos los pasos para la creación del balanceador de carga apoyándonos por medio de imágenes para que la explicación sea clara.

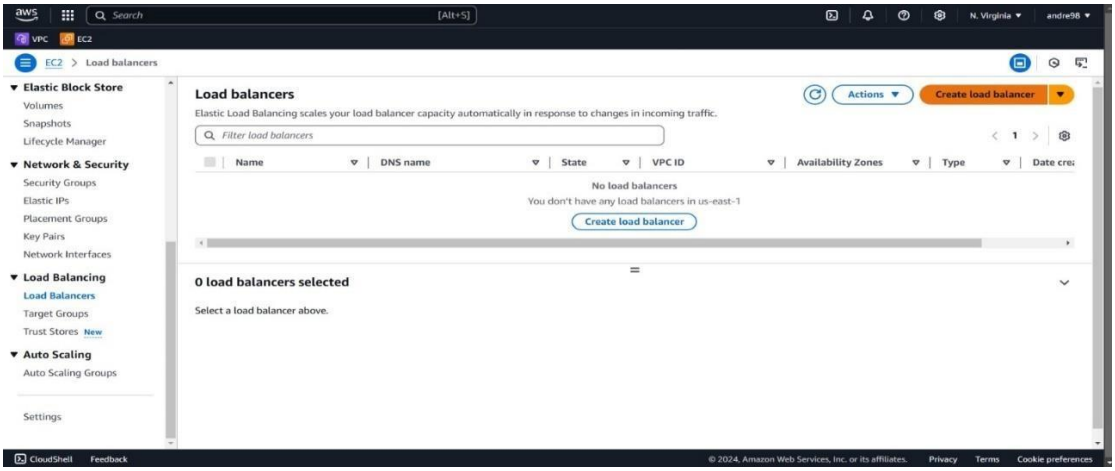



***Figura 1.*** *creación del balanceo de carga.*

En este primer paso, describiremos de forma breve la creación de un balanceador de carga, en el panel AWS, buscamos y seleccionamos EC2. En el menú izquierdo seleccionamos Load Balancers que se encuentra debajo de la sección de Load Balancing y hacemos clic en el botón Create Load Balancer.

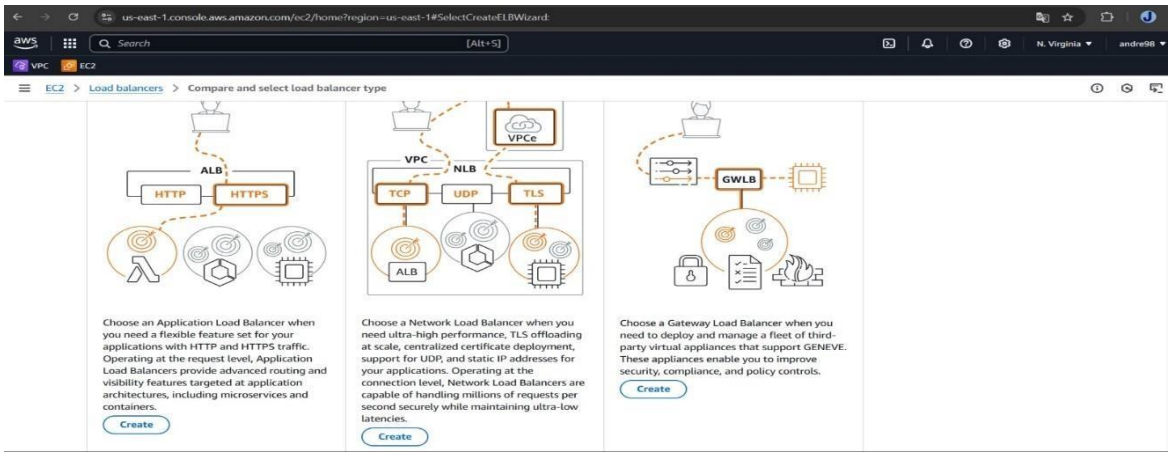



***Figura 2.*** *Tipo de balanceador de carga.*

En este siguiente paso seleccionaremos el tipo de balanceador de carga que deseemos o necesitemos configurar. Para la mayoría de los casos y en esta ocasión elegiremos aplication load Balancer ya que estamos trabajando con HTTP/HTTPS

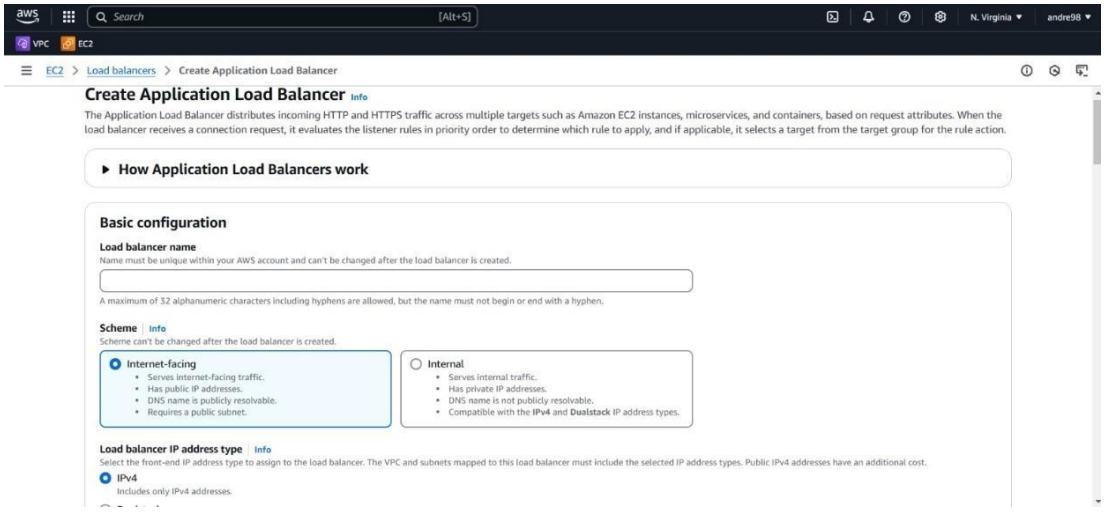



***Figura 3.*** *Nombre de balanceador de carga.*

Para este paso especificaremos cual será el nombre para nuestro balanceador de carga, también seleccionaremos el esquema de red que en este caso será internet-facing ya que será expuesto a internet

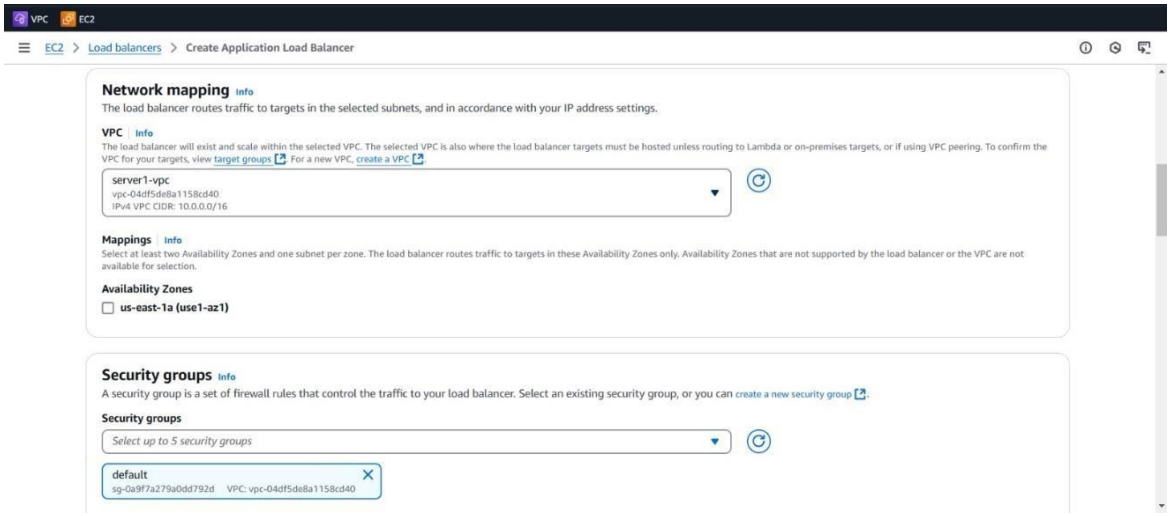



***Figura 4.*** *Distribución de tráfico.*

Para el cuarto paso elegiremos la VPC en la que se encuentran nuestras instancias creadas y seleccionamos las subredes de availability zonas (AZS)

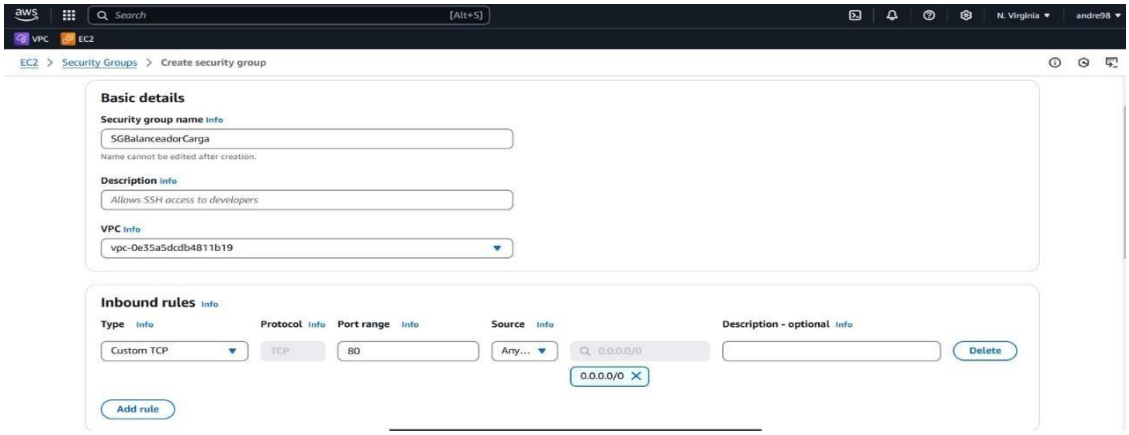



***Figura 5.*** *Creación de un security group.*

Para este paso proporcionaremos un nombre único para nuestro grupo de seguridad. Este nombre nos ayudara a identificarlo de manera más sencilla. Luego seleccionamos la VPC en la cual se encuentra nuestras subredes. Definimos las reglas de entrada (Inbound Rules) para las cuales elegiremos el tipo, luego el protocolo, el cual se selecciona automáticamente según el tipo. Por ejemplo, TCP para HTTP/HTTPS que en este

momento es nuestro caso, también el puerto de destino (80) se selecciona automáticamente y el por el origen.

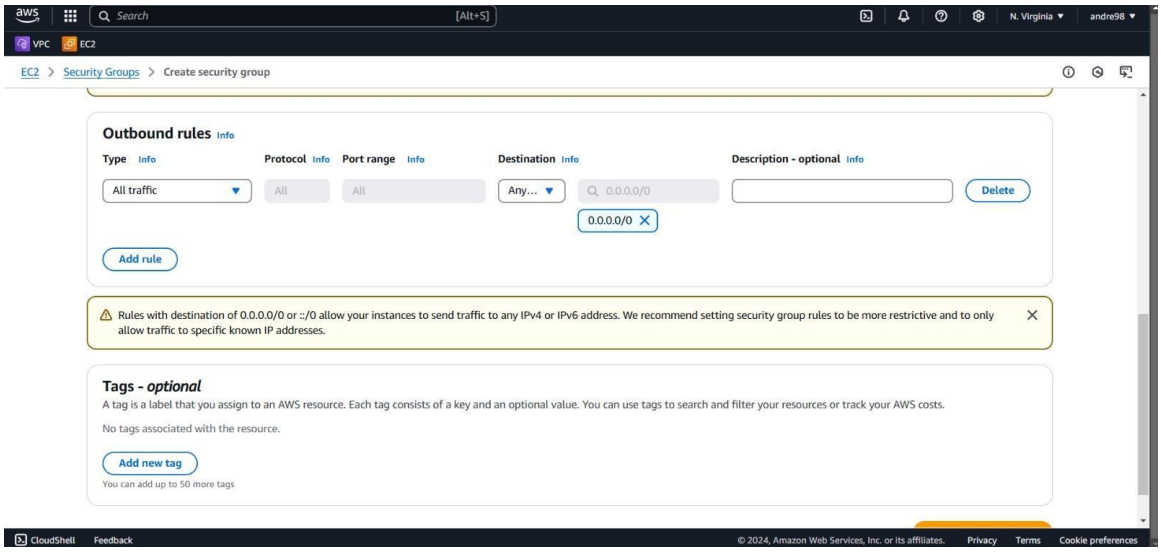



***Figura 6.*** *Reglas de salida.*

Ahora en las reglas de salida (Outbound Rules) de manera predeterminada, el grupo de seguridad permite todo el tráfico de salida. En nuestra aplicación lo dejaremos así, en caso de que no desees permitir todo el tráfico puedes agregar las reglas de salida de manera específica.

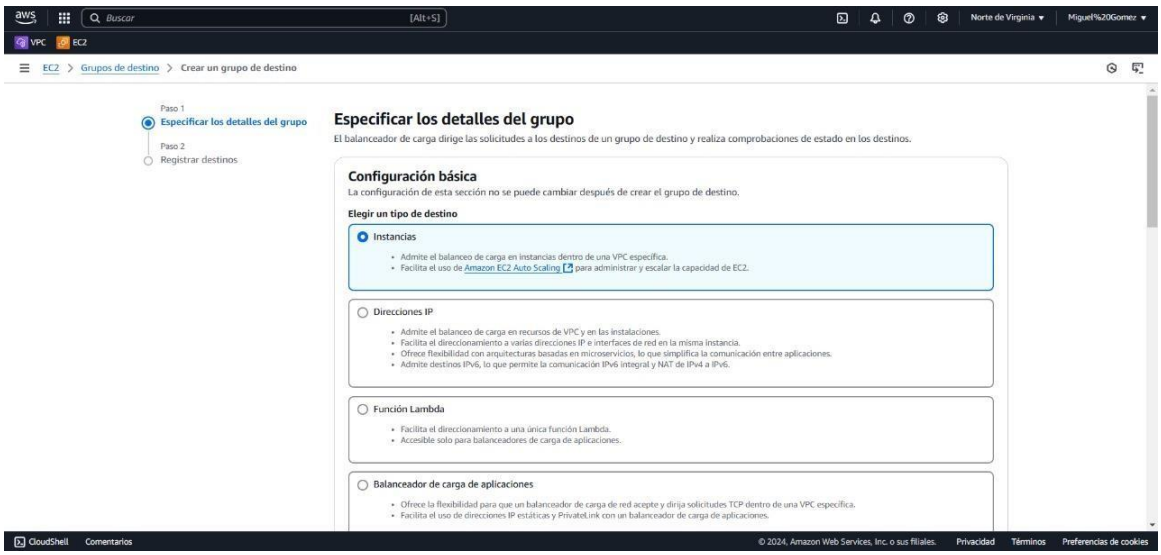



***Figura 7.*** *Configuración del target group.*

En este paso asociaremos las instancias que tenemos en nuestra infraestructura al 1

balanceador de carga, nos dirigimos a la sección de load Balancing nuevamente y bajo la misma seleccionamos la opción target group y hacemos clic en el botón create target group y seleccionamos el tipo el cual en este caso será instancias.

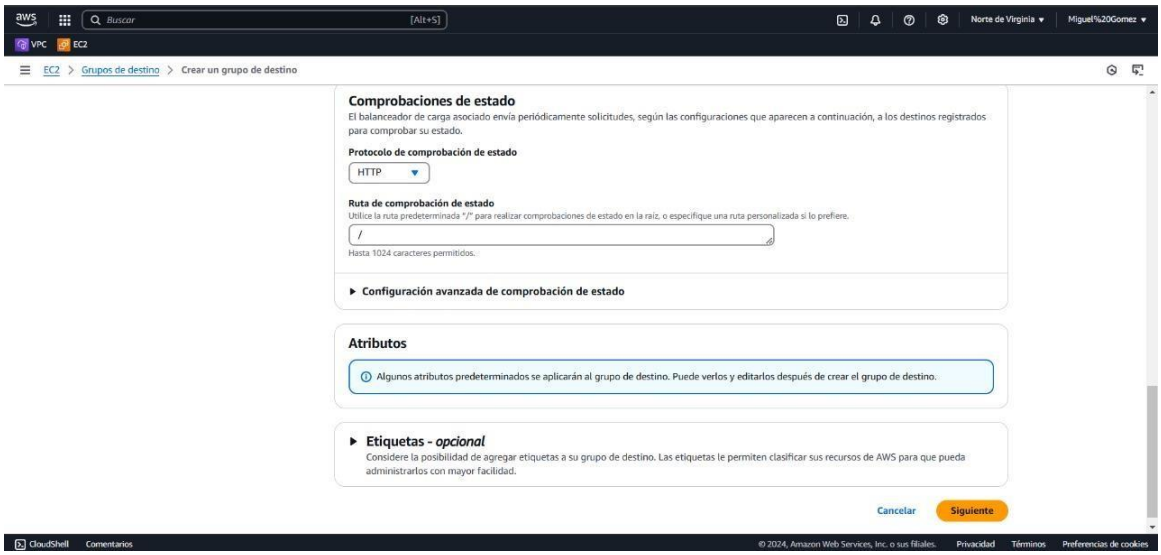



***Figura 8.*** *Creación del target group.*

Luego proporcionamos un nombre al Target Group, continuamos configurando el protocolo HTTP y el puerto (80). Seleccionamos la VPC donde se encuentran nuestras instancias

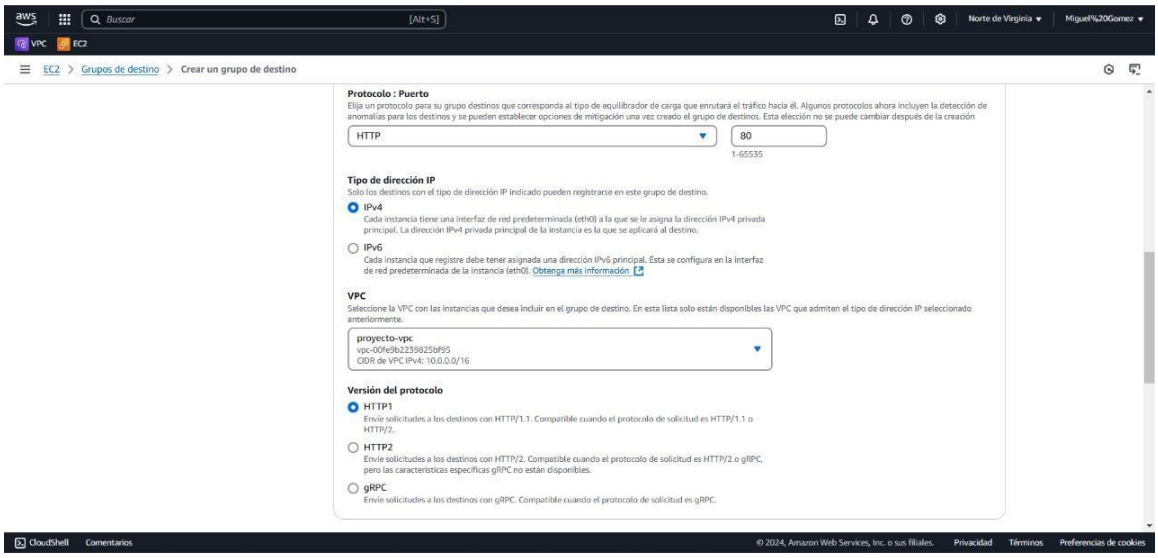



***Figura 9.*** *Configuración del target group.*

También definimos los parámetros de Healt check (comprobación de estado o

comprobación de salud), como lo son el protocolo HTTP/HTTPS y la ruta. Y por último hacemos clic en create para crear el target group.

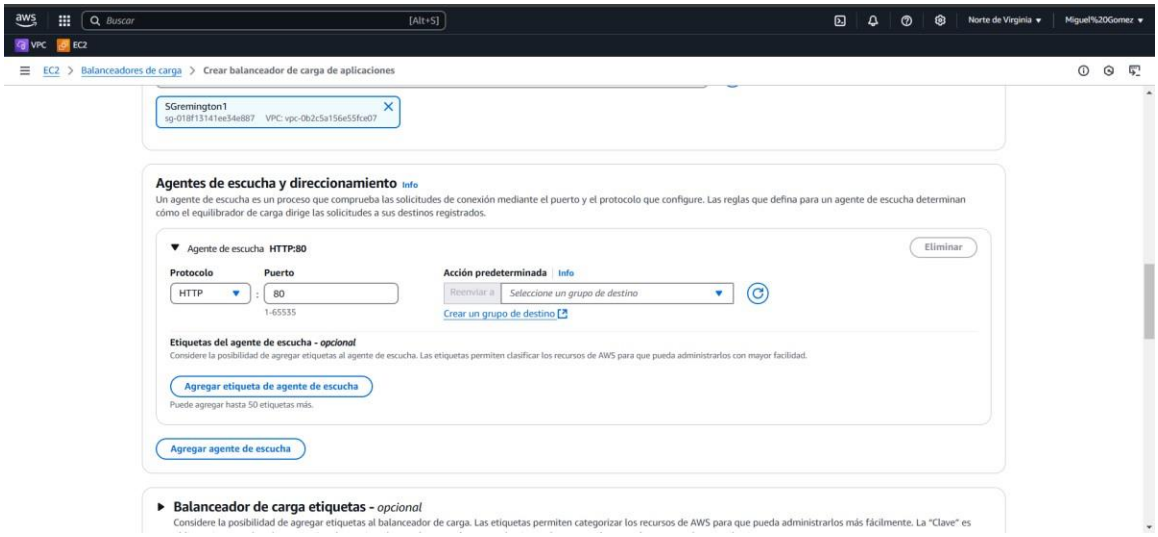



***Figura 10.*** *Ultimo paso balanceador.*

Después de crear los targets groups, procedemos a devolvernos al balanceador y dar clic en el botón de reiniciar donde dice acción predeterminada y esperar a que nos aparezca el target group creado y por último dejar el resto de configuración predeterminada y dar clic en crear balanceador y comprobar si fue creado correctamente.

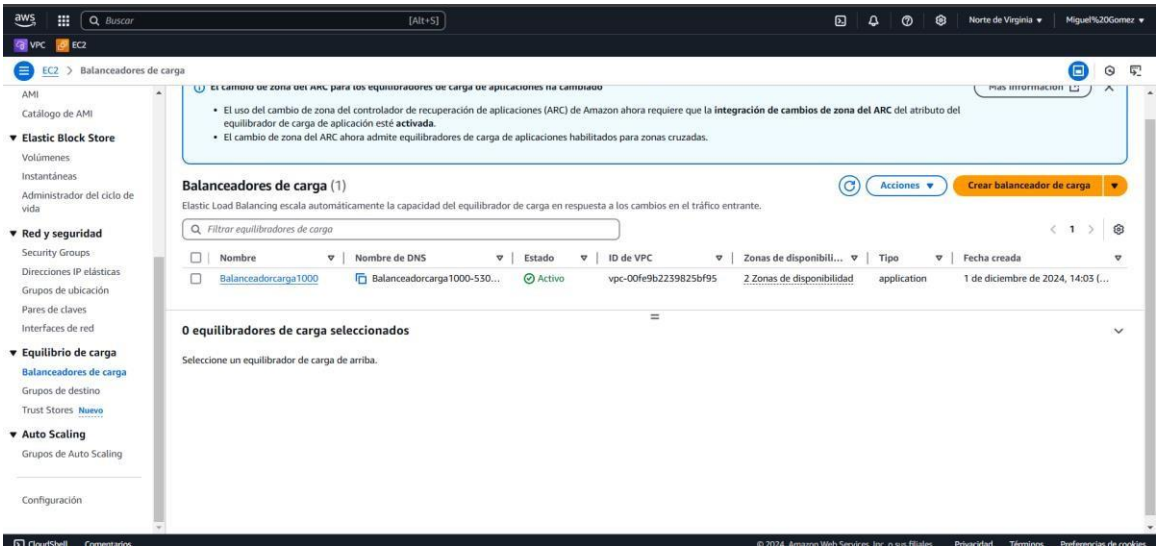



***Figura 11.*** *Balanceador de carga.*

Por último, comprobamos de que el balanceador de carga fue creado correctamente y vemos que se encuentra en estado de ejecución.

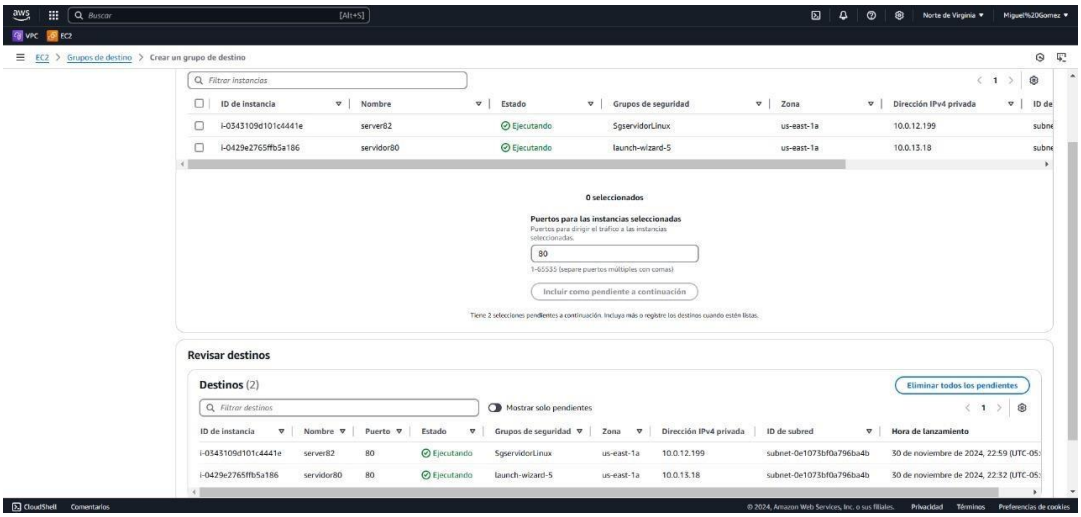



***Figura 12.*** *Configuración del target group.*

Después de registrar los targets groups, es importante verificar el proceso que hemos realizado se haya completado correctamente, regresamos a la pestaña donde se encuentra nuestro target group, allí podremos ver la lista con los targets registrados y el estado en el que se encuentra actualmente. El estado de cada Target se mostrará como Healthy (Saludable) o Healthy (No Saludable).

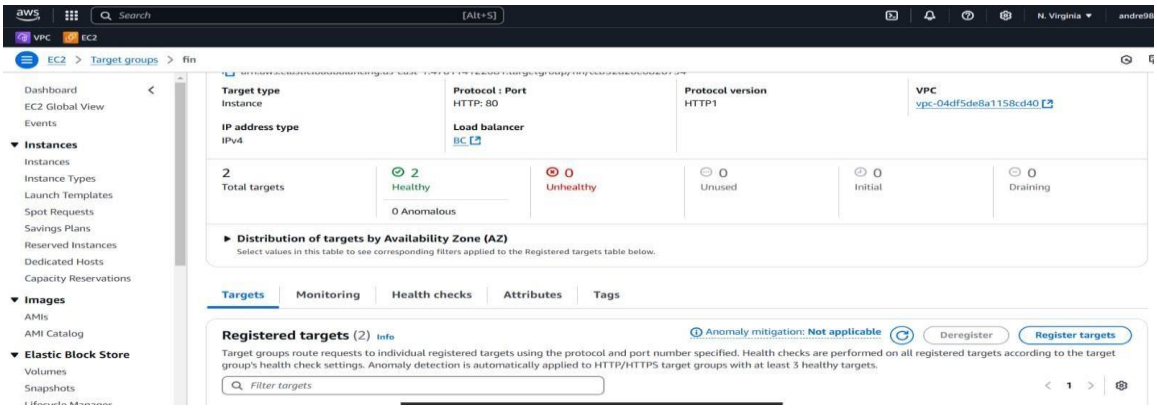



***Figura 13****. Verificación del target group.*

En este caso observamos que los targets groups se crearon de forma correcta, ya que nos aparece Healthy (saludables), en caso de que le haya salido unhealthy hay que verificar si las instancias están en una VPC diferente al balanceador de carga.

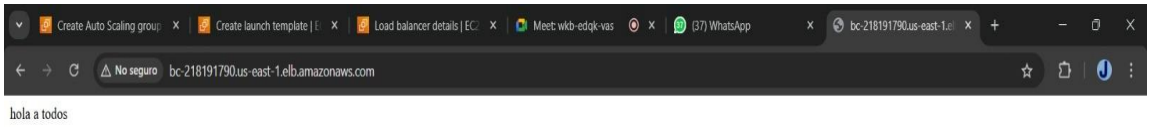



***Figura 14.*** *Funcionamiento DNS.*

En este paso comprobamos de que el balanceador de carga está funcionando correctamente por medio del DNS y que si está repartiendo el trabajo a las 2 instancias que creamos.

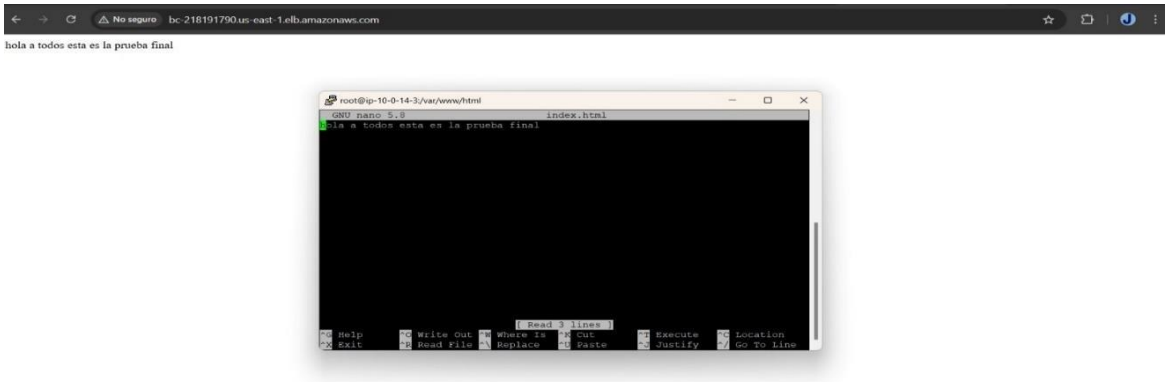



***Figura 15.*** *Modificar archivo.*

En este paso modificamos una instancia para idénticar el cambio de la instancia por medio del balanceador de carga a través del comando nano.

## Que es un launch template

El launch template es un recurso muy importante en AWS ya que nos permite realizar varias instancias EC2 por medio de su proceso de configuración y aprovisionamiento de las instancias ya que utiliza una plantilla predefinida en lugar de tener que especificar los parámetros cada vez que lanzas una instancia.

### ¿Pasos para la creación de un launch template?

Para realizar este proceso se debe de seguir unos pasos muy cuidadosamente y

verificar que los targets group y el apache de las instancias estén configurados perfectamente.

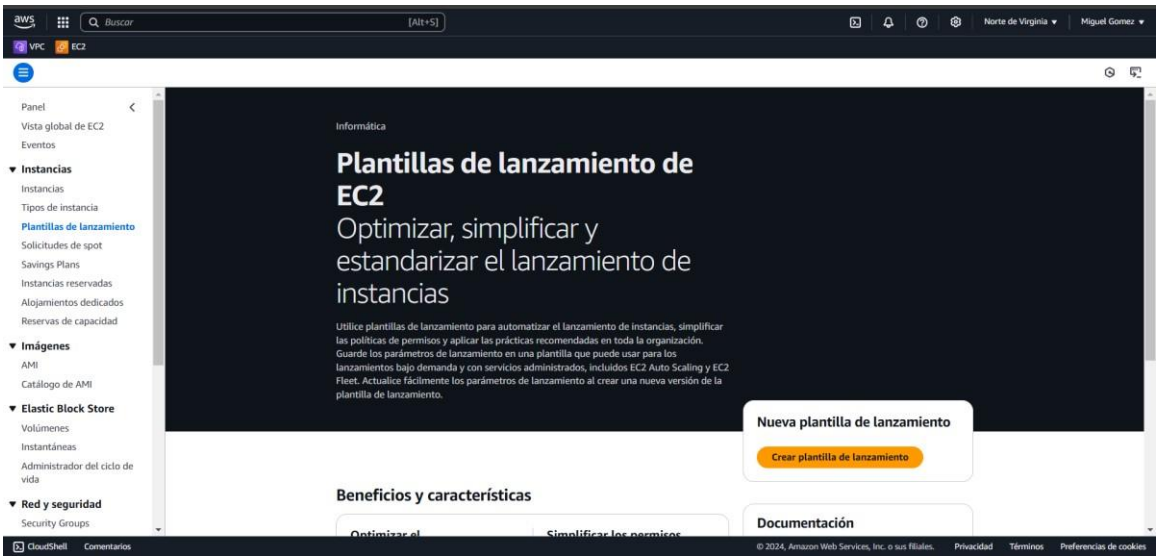



***Figura 1.*** *Launch template.*

Para empezar con la creación del launch template, primero debemos de seleccionar el menú del EC2, dirigirnos al apartado de instancias y allí seleccionar plantillas de lanzamiento (Launch template), allí nos saldrá un apartado al lado derecho donde daremos clic en crear plantilla de lanzamiento y empezaremos a configurar nuestro launch template.

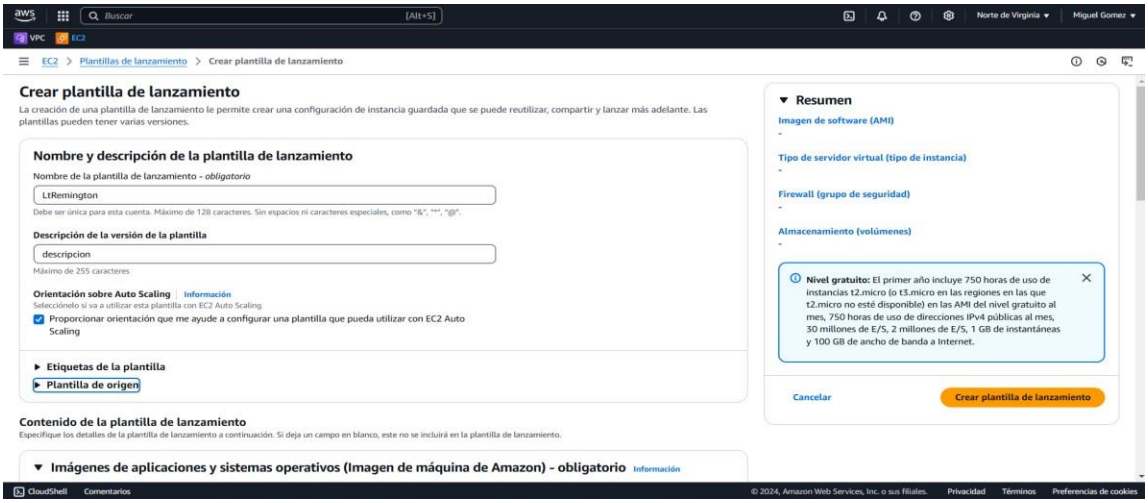



***Figura 2.*** *Creación launch template.*

En el segundo paso procedemos a realizar la respectiva configuración del launch

template, primero le colocaremos un nombre a nuestro launch template y colocaremos una descripción, daremos clic en la orientación sobre el auto scaling y procederemos a realizar el siguiente paso.

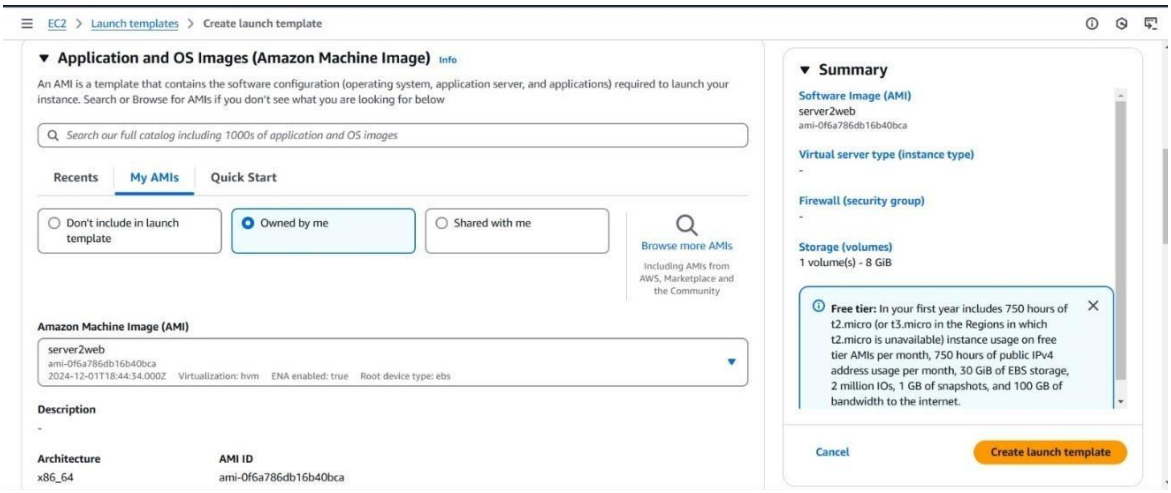



***Figura 3.*** *Selección AMI.*

En el tercer paso escogeremos la AMI que ya tenemos creada, ya que nuestro Launch template empezara a funcionar a través de ella.

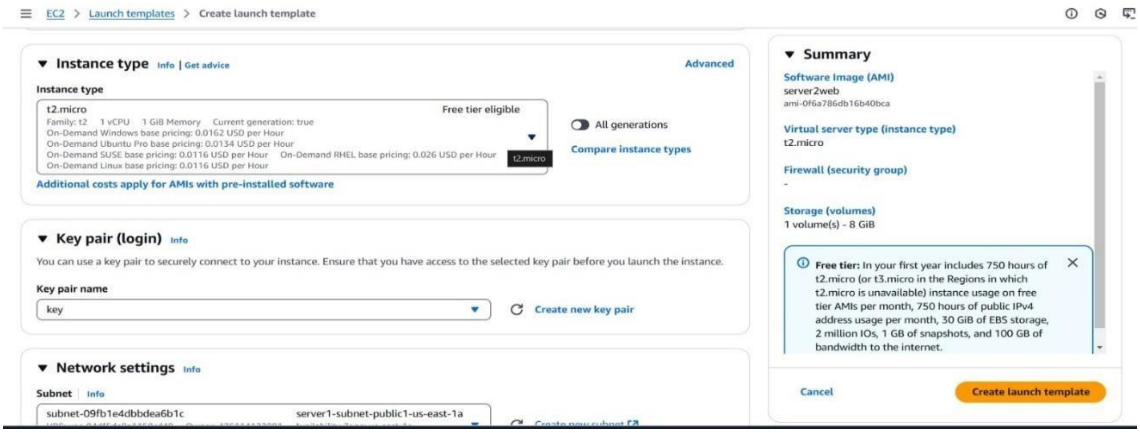



***Figura 4.*** *Selección Llave.*

En el cuarto paso seleccionaremos tipo de instancia en este caso seleccionaremos la T2 micro ya que es la gratuita, luego seleccionamos la llave que creamos al inicio para entrar a la configuración de nuestro servidor, también configuraremos la red escogiendo cualquiera de las redes públicas que tenemos y procedemos a deja el resto de infraestructura tal cual como está ya que no se necesita realizar más cambios para nuestro

launch template, por ultimo daremos clic en crear nuestro launch template.

## ¿Qué es un Auto Scaling?

El auto scaling es otro servicio importante en AWS ya que nos permite ajustar nuestros recursos con las instancias EC2, en función de la demanda del servidor que tengamos, su objetivo principal es garantizar que nuestra aplicación si tenga el rendimiento adecuado mientras optimiza los costos, es decir funciona a través de un escalado hacia arriba(Scale out) lo cual hace agregar más recursos cuando aumenta la carga de trabajo de nuestro servidor, durante horas pico y su otra función que es el escalado hacia abajo( Scale in), es el que nos permite reducir los recursos cuando la carga disminuye ayudando a reducir costos.

### ¿Pasos para la creación de un Auto Scaling?

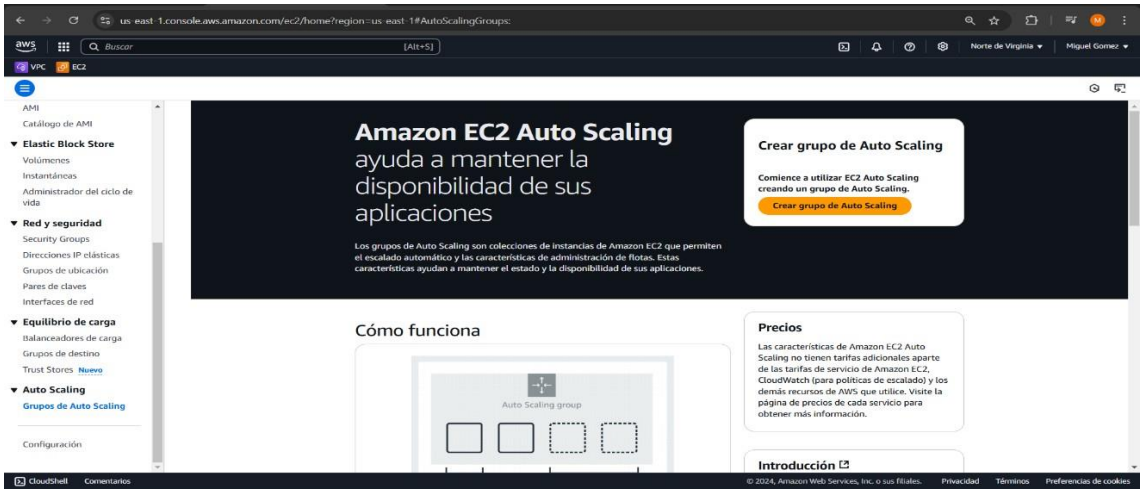



***figura 1***. *Launch template.*

Para empezar con la creación del auto scaling, primero debemos de seleccionar el menú del EC2, dirigirnos hasta la parte de abajo donde dice auto scaling allí daremos clic en la opción de auto scaling group, y nos saldrá un apartado en la parte derecha donde daremos clic en el botón amarillo que dice crear grupo de auto scaling.

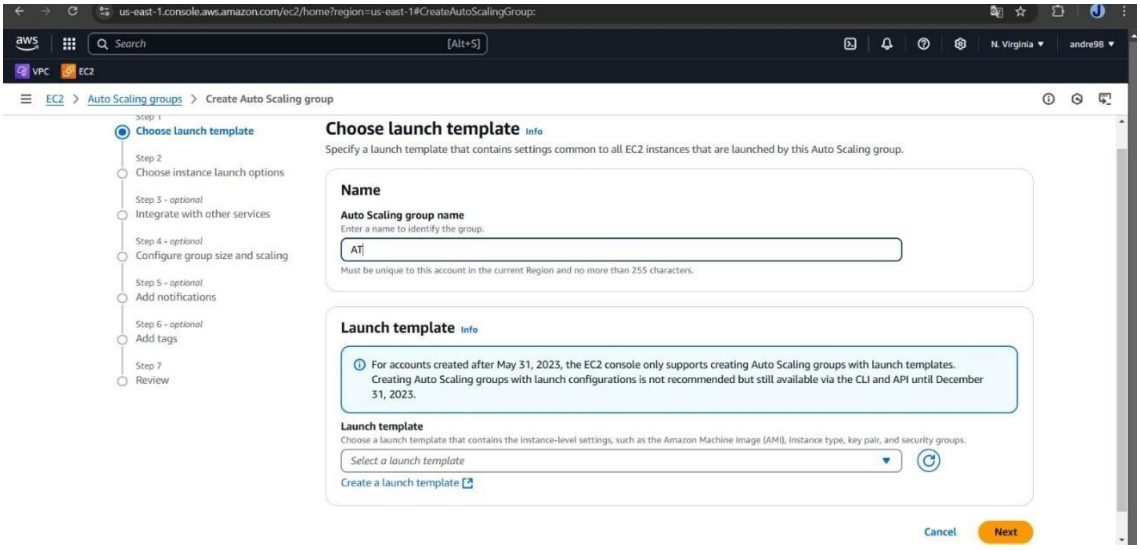



***Figura 2.*** *Menú launch template.*

El segundo paso para realizar es configurar el launch template, para ello colocaremos el nombre de nuestro auto scaling y seleccionaremos el launch template que ya tenemos creado y daremos clic en siguiente.

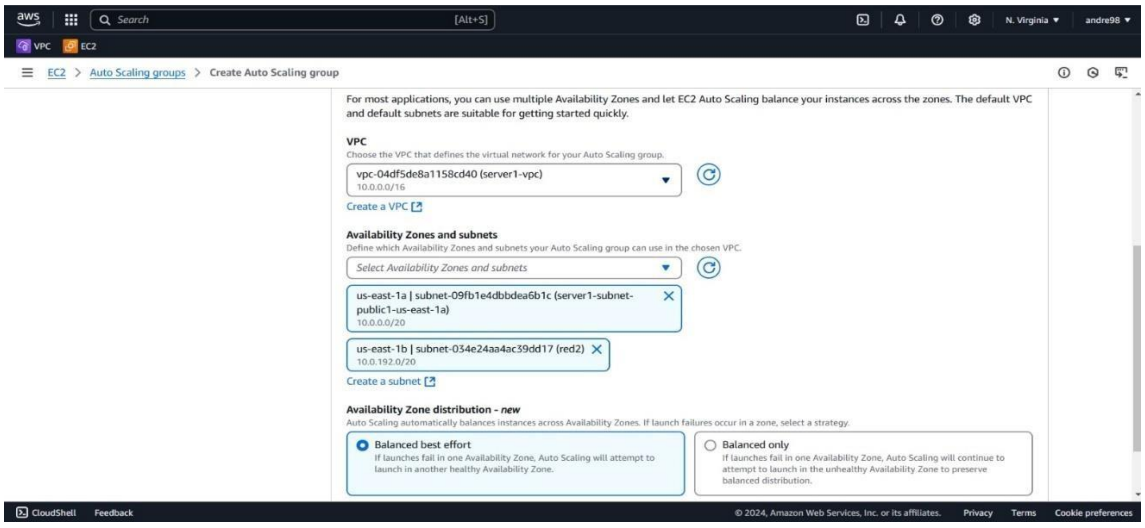



***Figura 3.*** *Configuración de red.*

En el tercer paso nos pide ya la configuración de red el cual seleccionaremos las 2 redes públicas que ya tenemos configuradas y dejaremos el resto con la configuración predeterminada y daremos clic en siguiente.

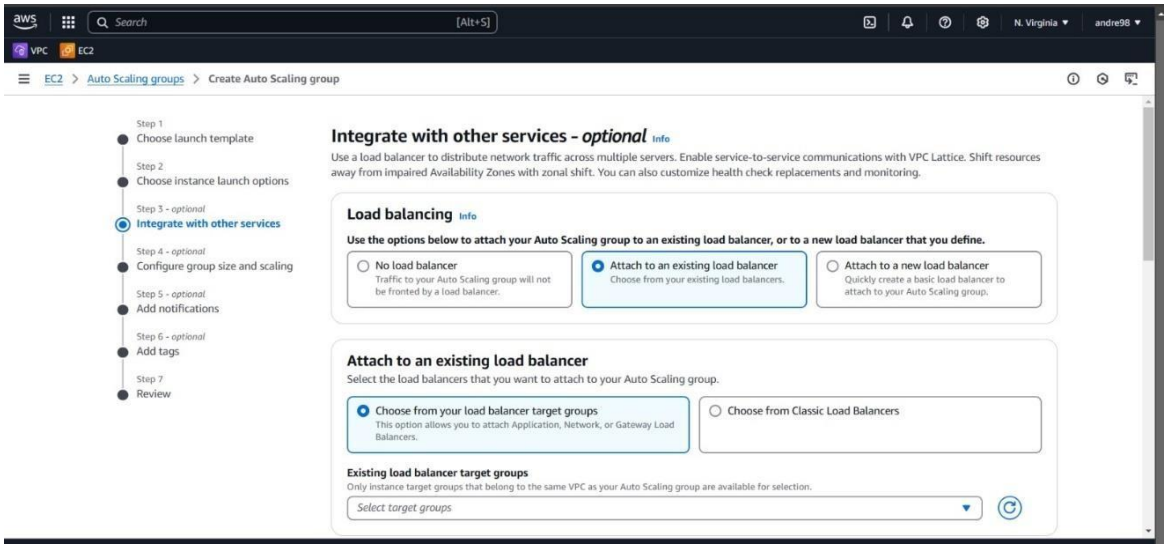



***Figura 4.*** *Configuración del balanceador de carga.*

En el cuarto paso lo que estamos haciendo es crear el auto scaling el cual seleccionaremos en load Balancing la segunda opción ya que esta opción nos permite asociarnos con el balanceador de carga que ya tenemos creado.

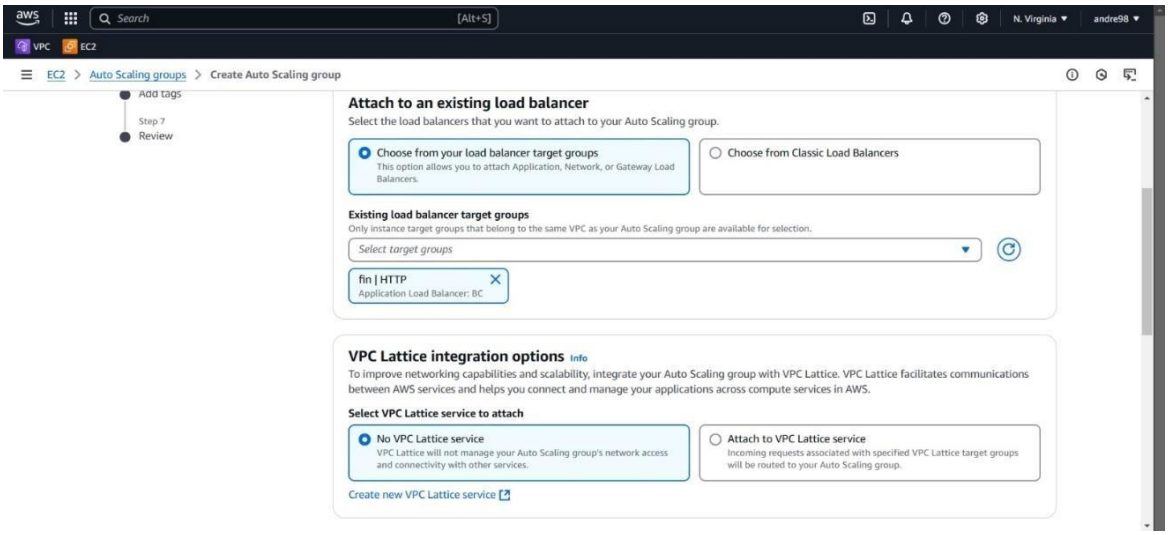



***Figura 5.*** *Configuración balanceadora de carga y vpc.*

En el quinto paso seguiremos con la configuración del balanceador y la configuración de red, en el segundo recuadro lo que hacemos es seleccionar la primera ya que el balanceador de carga va a funcionar con los targets groups que ya tiene creados, después seleccionamos el balanceador de carga que ya tenemos creado y en el siguiente recuadro

procedemos a la configuración del VPC en este caso lo dejaremos así en la primera opción.

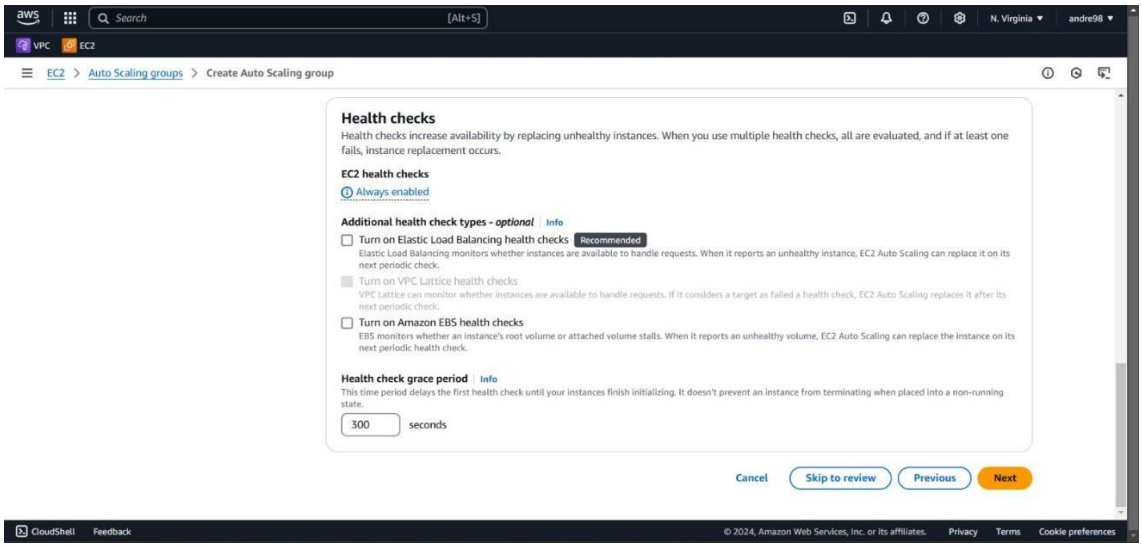



***Figura 6.*** *Health checks.*

En el sexto paso es la configuración de los Health checks el cual lo dejaremos tal cual como esta y daremos clic en siguiente.

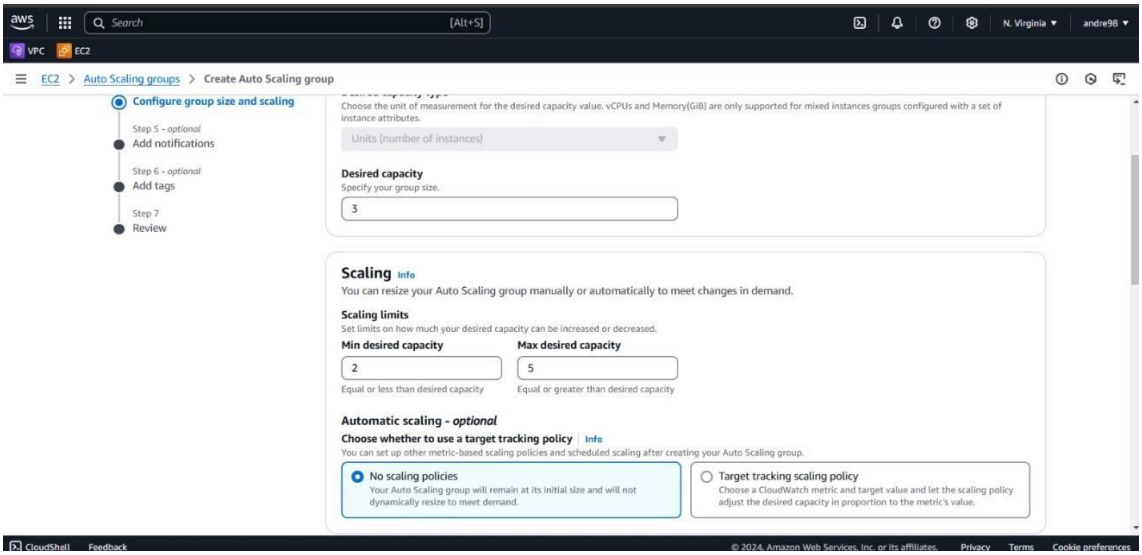



***Figura 7.*** *Configuración del scaling group.*

En el séptimo paso procedemos a configurar el auto scaling lo cual lo que hacemos es decir cuantas maquinas queremos que tenga nuestro servidor para el funcionamiento de nuestra página, por ejemplo, cuantas maquinas queremos que tenga nuestro sitio web cuando aumenta la carga de trabajo en nuestro sitio web y cuantas maquinas queremos

que tenga cuando ya no hay tanta carga de trabajo en nuestro sitio web.

La otra configuración de los scaling groups se deja por defecto y damos clic en siguiente.

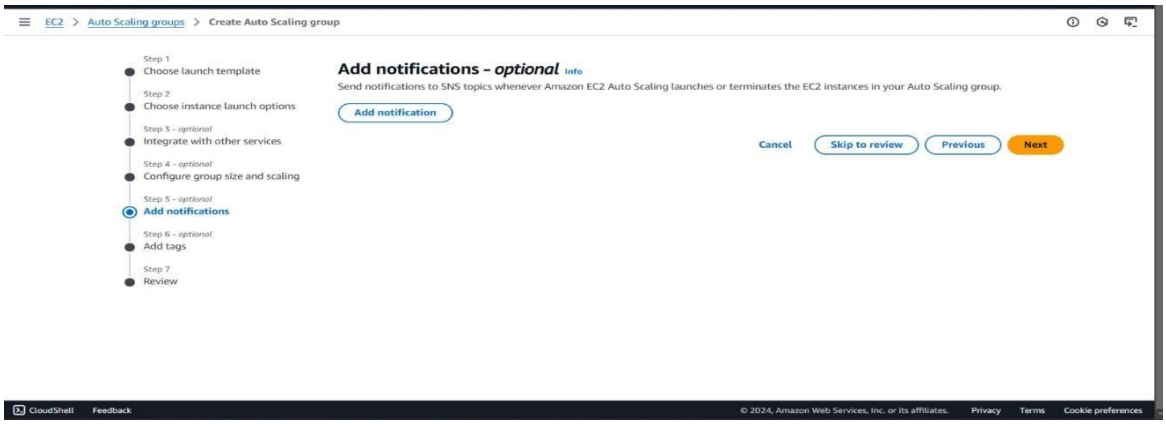



***Figura 8.*** *Notificaciones.*

En el octavo paso nos indica si queremos recibir notificaciones cuando la condición que le colocamos al scaling group se cumple.

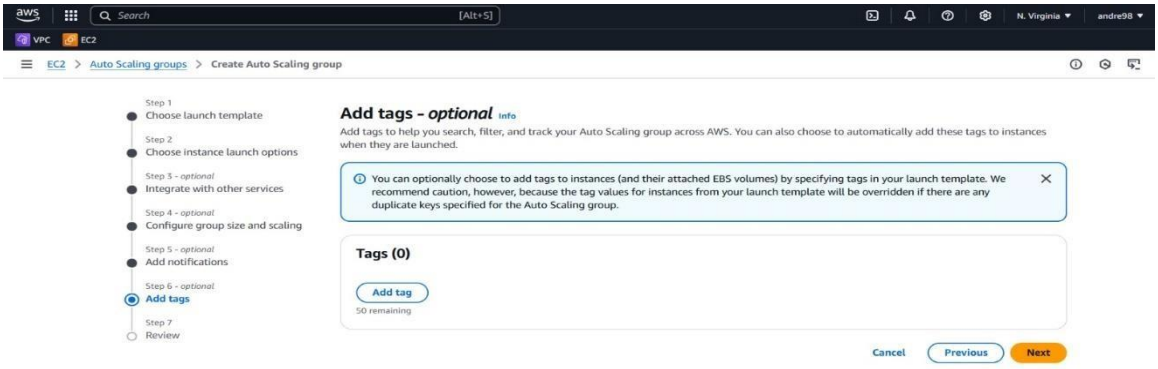



***Figura 9.*** *Notificaciones.*

En el noveno paso es por si queremos algún identificador a nuestro scaling group, en este caso lo dejaremos así y daremos clic en siguiente.

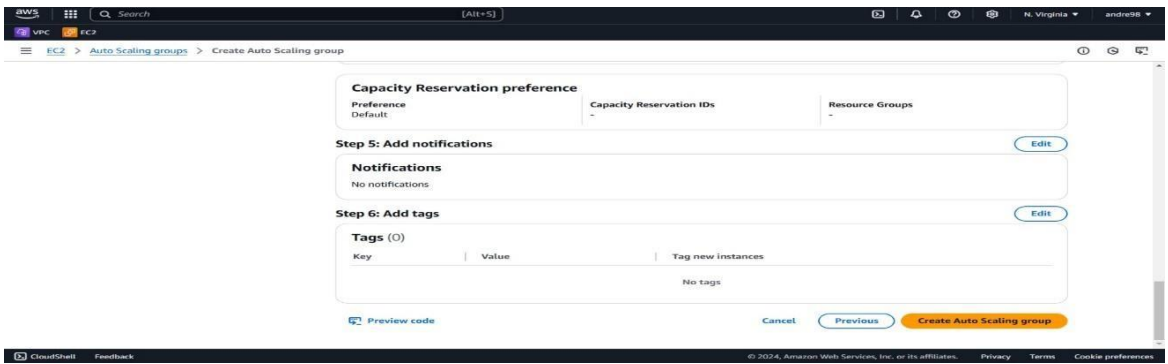


***Figura 10.*** *Crear.*

En el décimo paso procedemos a dar clic en crear y de esta forma se crearía el scaling group correctamente.

## Enlaces videos

Enlace de video de YouTube.

https://youtu.be/r175_LPowLc

Enlace de video drive.

https://drive.google.com/file/d/1d3DHcFf2x9HfzGP7C9giBytCFZff8ci3/view?usp=sharing

## Implementación del nginx y docker

En este apartado implementaremos el nginx y el docker para la empresa JMS, ya que de esta manera nos ayudara a controlar un poco más el flujo de carga de nuestras maquinas(instancias) para el rendimiento de las peticiones de los clientes.

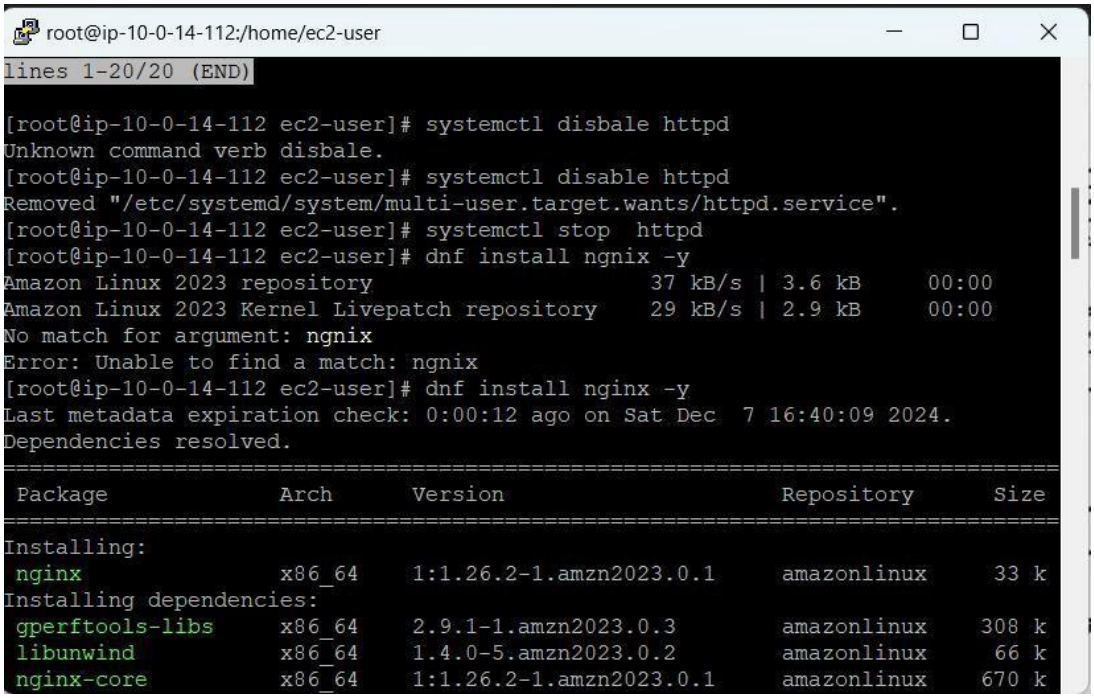



***Figura 1****. Instalación Nginx.*

Como primer paso procedemos a deshabilitar el HTTPD con los comandos que se ven reflejados en la imagen, que en este caso son: systemctl disable httpd para evitar de que el arranque solo cuando ya lo paremos, y procedemos a escribir el siguiente comando para poder parar el httpd systemctl stop httpd. Después de parar el httpd ahora si podemos

instalar el NGNIX con el comando dnf install Nginx -y, le ponemos -y para que no nos pida confirmación a la hora de que se instale.

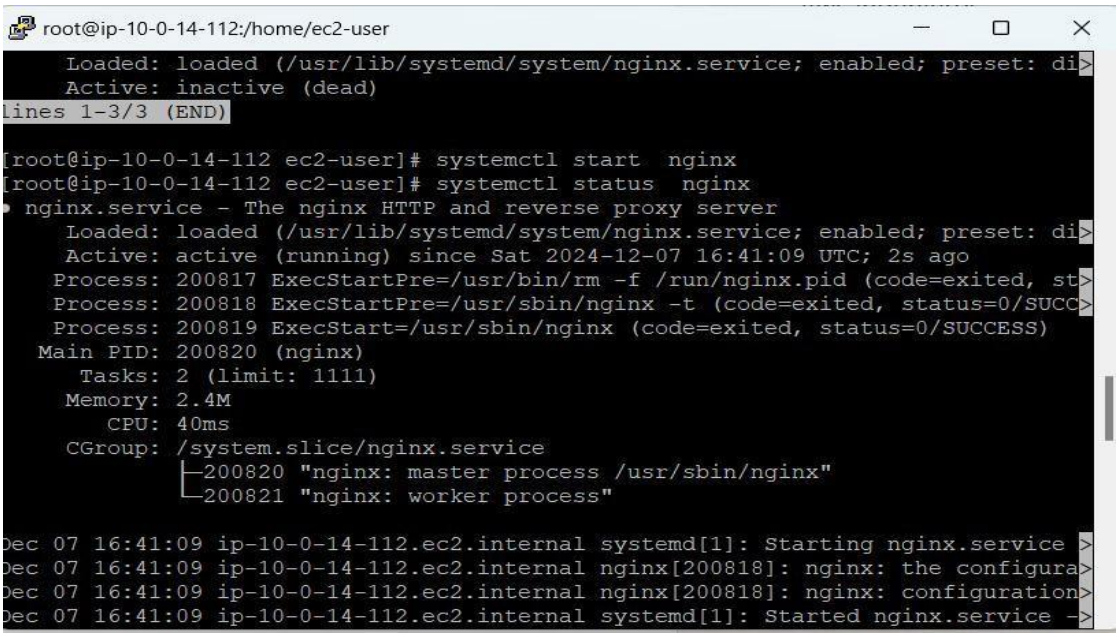



***Figura 2.*** *Inicialización.*

Para el segundo paso procedemos con la inicialización del NGNIX, para ello vemos que los comandos son iguales que los de administración del apache, entonces procedemos a colocar los siguientes comandos systemctl start Nginx para activarlo, y luego verificamos su estado con systemctl status Nginx, también se le puede agregar el comando systemctl enable nginx para que el arranque solo.


***Figura 3.*** *Verificación del Nginx*

En el tercer paso procedemos a copiar la ip de la instancia y pegarla en el navegador para

poder verificar que esté funcionando correctamente el NGINX, y es allí donde saldrá el siguiente mensaje que vemos en la imagen anteriormente.

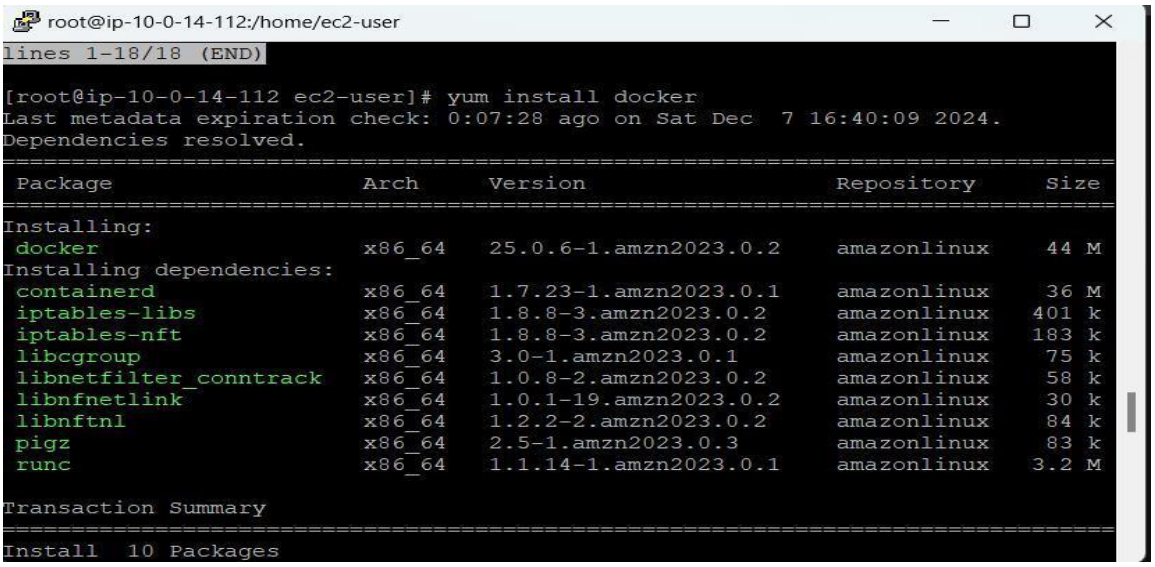



***Figura 4.*** *Instalación Docker*

En el cuarto paso procedemos a instalar el docker con el siguiente comando yum install docker, también se le puede agregar el comando -y para que no pida confirmación a la hora de su instalación, también se le ejecutan los mismos comandos de inicialización del Nginx para que quede funcionando perfectamente.

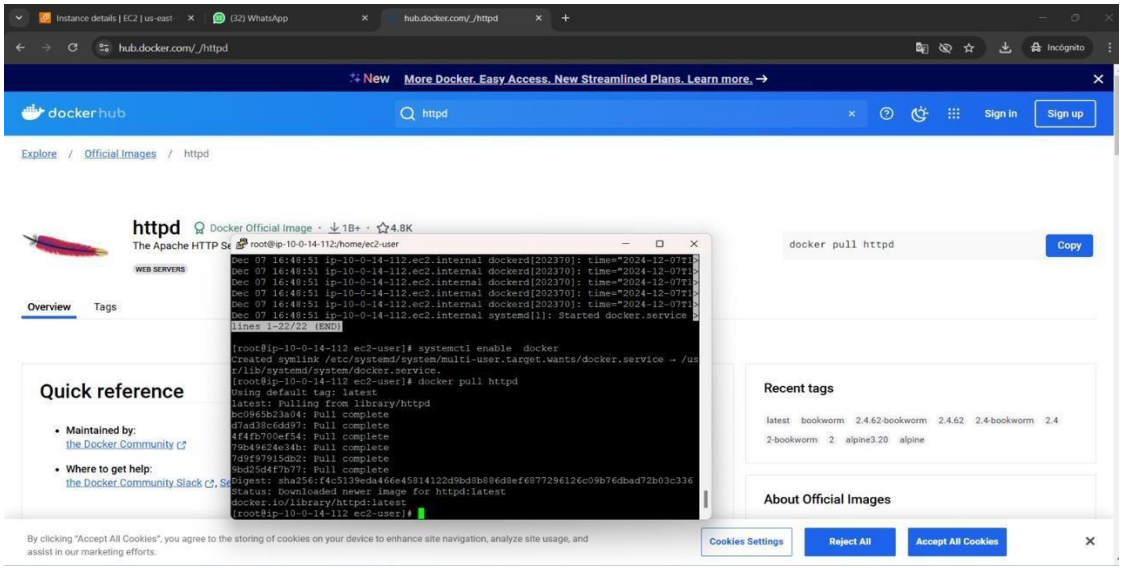



***Figura 5.*** *Imagen Docker*

Para el quinto paso nos dirigimos al navegador e ingresamos al sitio web **hub.docker.com** para realizar la descarga de la imagen del Docker, copiamos y pegamos

la imagen que deseamos, esto lo hacemos para poder crear contenedores que va a realizar un proceso dentro del sistema operativo y para ello es que necesitamos la imagen, sin esto no se podría crear los múltiples contenedores que necesitemos.

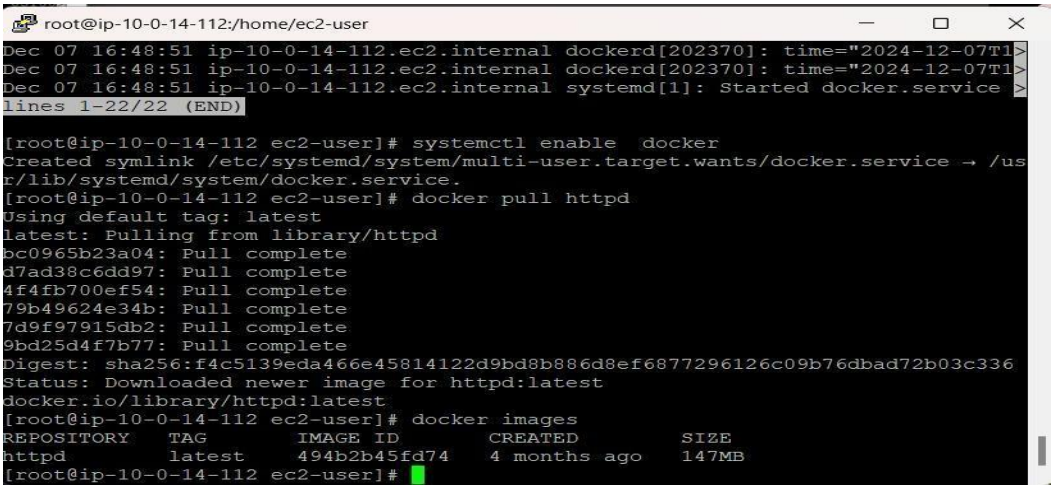



***Figura 6.*** *Verificación Imagen Docker*

En el sexto paso realizamos lo siguiente, para verificar que la imagen que descargamos e instalamos haya sido instalada correctamente, procedemos a ejecutar el comando **Docker images** y con el podremos evidenciar que las imágenes que tengamos estén instaladas correctamente de forma local.

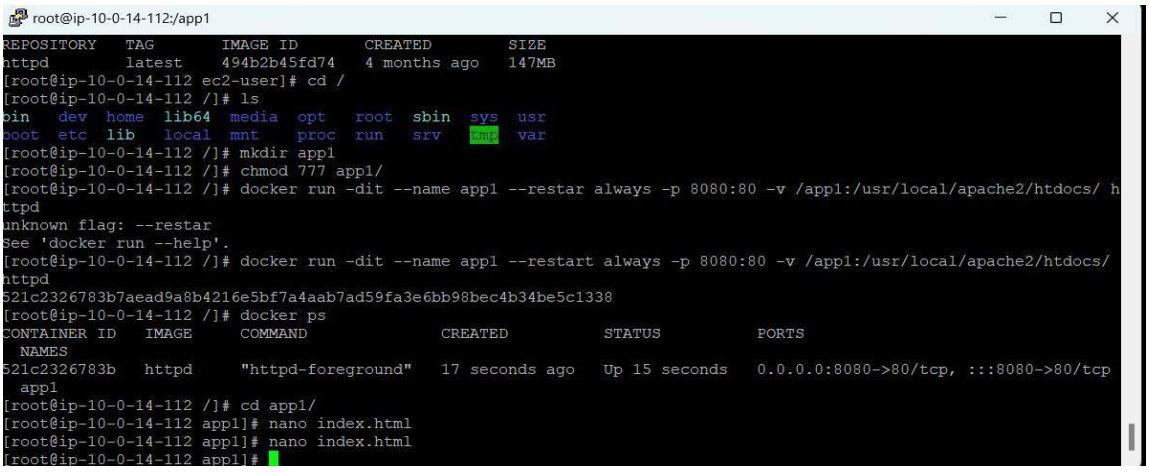



***Figura 7.*** *Creación de Carpeta*

En el séptimo paso creamos una carpeta que se va a llamar app1 y le damos todo los permisos para que no nos vaya a causar error con el siguiente comando chmod 777 app1/, después de realizar la creación de la carpeta procedemos a crear un componente con el comando docker run -dit –name app1 –restart always -p 8080:80 -v /app1 hasta ahí el

comando está bien pero ahí ya hay que colocarle una ruta donde queremos que aparezca nuestro apache puede ser la que colocamos en la imagen o cualquiera que escojan dentro de la documentación de estos servidores. Después les aparecerá de que el componente fue creado correctamente, y con el comando nano procedemos a colocarle un nombre dentro de ese contenedor que en estos momentos esta vació.


**Figura 8.** *Creación de archivo Index*

Para este paso ingresamos a la carpeta que creamos anteriormente y creamos un archivo con el comando utilizado anteriormente **nano (nombre del archivo).html** en este caso nombrado index y colocamos el mensaje (hola contendor app1) para que aparezca en nuestro navegador y comprobemos de que si está funcionando correctamente nuestro componente. Tener en cuenta que para correr nuestro componente primero debemos evidenciar de que nuestro Nginx este apagado con los comandos realizados en el httpd, pero cambiando el httpd por el Nginx.

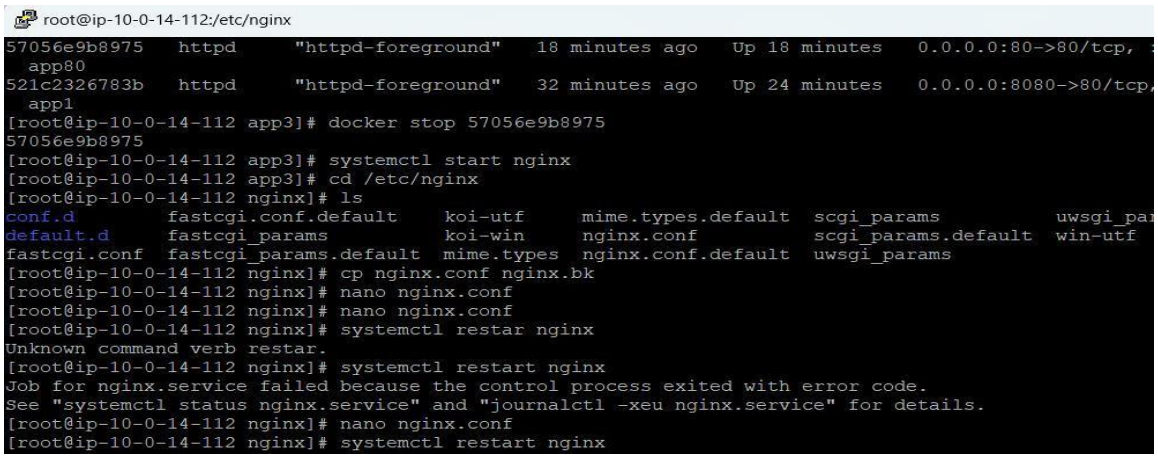



***Figura 9****. Reinicio de servicio y backup del archivo nano.*

En el noveno paso procedemos a modificar el archivo nano nginx, pero para poder realizar este paso procedemos a volver activar el nginx con los comandos realizados anteriormente, después de activarlo le sacamos un backup al nano con el comando cp nginx.conf nfinx.bk, esto por si lo dañamos poder recuperarlo de una manera fácil. Ahora si procedemos a modificar el archivo nano con el comando nano nginx.conf, esto lo hacemos para borrar la configuración que tiene por defecto el nginx ya que el servicio que él tiene por defecto no nos sirve para la empresa de industrias JMS, sino que lo necesitamos como un envidador de peticiones

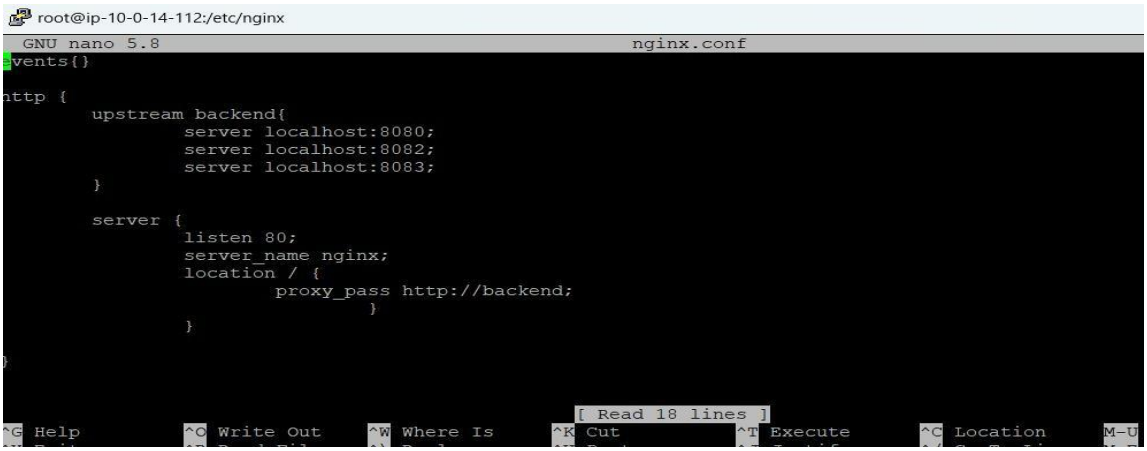



***Figura 10.*** *Modificacion del archivo nano.*

En el noveno paso procedemos a realizar la modificación del archivo nano como lo decíamos anteriormente, para ello procedemos a copiar los siguientes comandos, lo que hay que tener en cuenta es que estos son comandos son sacados de la página oficial de nginx para que después no surjan errores, el primer comando que colocamos es un comando que se llama events{} y se utiliza para cuando servicio web recibe muchas peticiones, en este caso no se utiliza pero es obligatorio que exista esa sección ahí, ya está el siguiente comando que es el http y es ahí donde vamos a colocar el conjunto de contenedores que tenemos creados y por ultimo está el comando Server que nos va a permitir la configuración ya desde afuera es decir como queremos recibir esas peticiones que nos van a mandar nuestro clientes o usuarios.

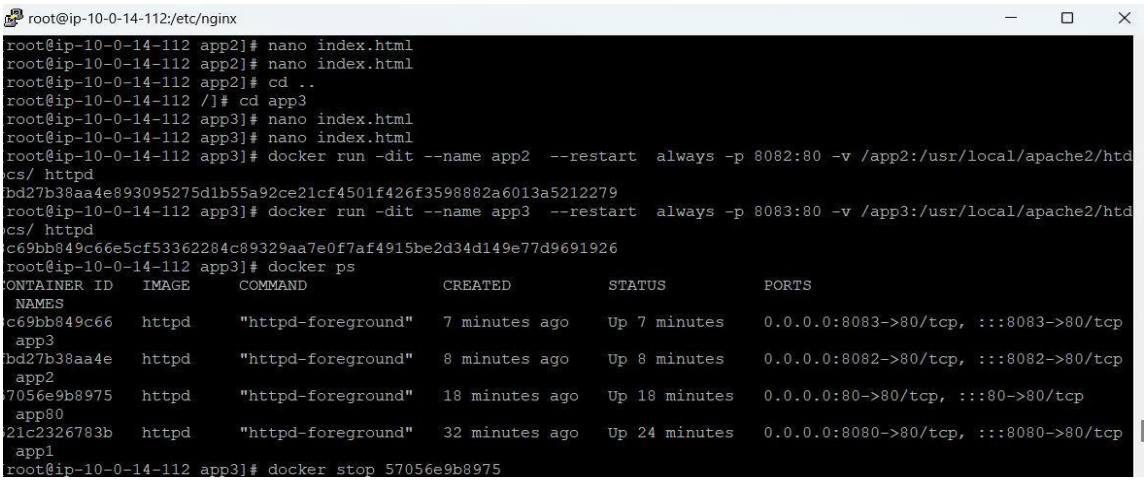
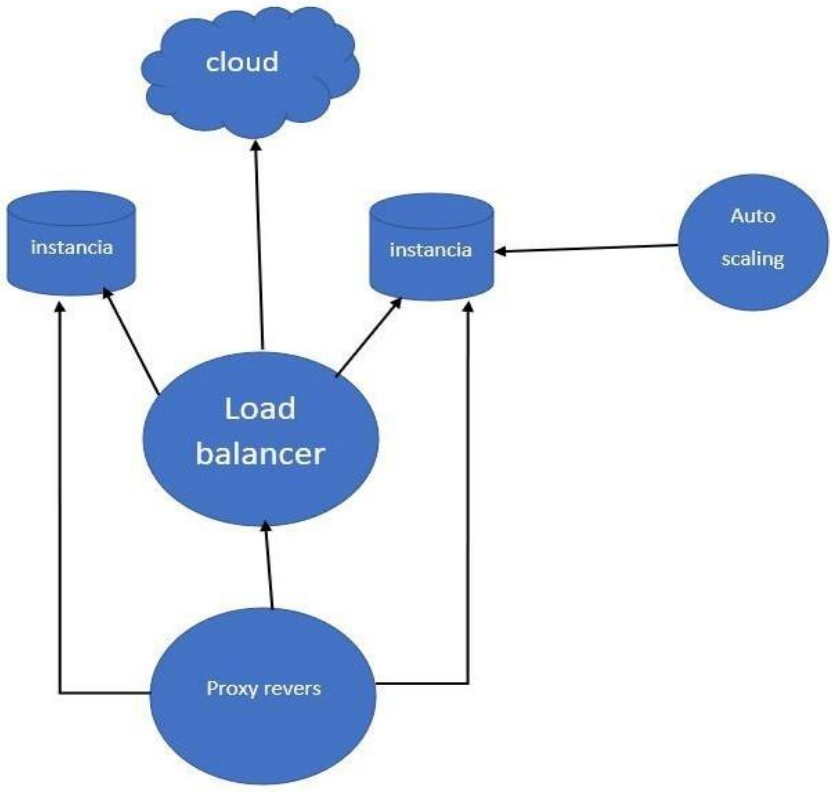



***Figura 11.*** *Contenedores.*

En el décimo paso procedemos a reiniciar el nano y volver a correr el nginx si todo esta bien aparecerán todos los componentes creados correctamente y escuchados por el puerto que indicamos, luego procedemos a coger la ip del contendor para comprobar de que ya está funcionando correctamente y deberá de salir los mensajes que le colocamos al nano.

**Diagrama de flujo**



Explicación del diagrama de flujo.

En este diagrama de flujo vemos cómo va conectado algunos de los servicios de AWS que utilizamos para la implementación de la empresa industria JMS.

Empezamos con las peticiones que llegan desde la nube al balanceador de carga elcual lo que hace es distribuir el tráfico a las instancias o máquinas, el siguiente es el auto

scaling que funcionara cuando allá un alto flujo de carga en las dos instancias implementadas y por el ultimo es el proxy revers para equilibrar las mimas peticiones, cumple con las mismas funciones del balanceador de carga.

**Enlaces videos**

Video de YouTube

https://youtu.be/6opr7zXqTos

Video en Drive

https://drive.google.com/file/d/1FJi8BCM_FcPcJx7yHDxHJ9ul18elR5u4/view?usp=sharing

## Conclusiones

Analizando la entrega de este trabajo final podemos observar que la creación de la infraestructura para estos proyectos como en la vida real, es bueno tener claro cómo va a funcionar, este aplicativo web o manejo de peticiones, como en este caso lo que se pudo realizar o implementar fua analizando lo que necesitaba la empresa industrias JMS, ya que su problema era el tráfico en la red, es por eso que se necesita saber que se debe implementar para estos servicios. Para esto utilizamos la herramienta balanceadora de carga que nos permiten manejar el tráfico en la red distribuyéndolos en varios servicios alivianando así la carga entre ellos cuando se encuentran con demasiado tráfico en su red.

También con la ayuda de los auto escaladores pudimos ayudar a solucionar este problema a la empresa de industrias JMS ya que cuando ellos tengan mucho tráfico en su red y si el balanceador de carga deja de funcionar el auto escalador va a ayudar a crear nuevas instancias o máquinas para que el tráfico siga fluyendo de una manera adecuada y no dejen de funcionar los servicios que ellos ofrecen y no hallan perdidas en la empresa, es de esta manera de que su infraestructura va a ser segura y sostenible.

## Referencias

*Documentación Amazon services, Sitio web:*
https://docs.aws.amazon.com/es_es/glossary/latest/reference/glos-chap.html