**TRABAJO DE GRADO**
**Opción Seminario-Diplomado.**

# Implementación de infraestructura de computación en la nube con autoescalamiento de recursos

Corporación Universitaria Remington.
Facultad de Ingeniería
Ingeniería de Sistemas

Yannick Steven Rincon Perez
Docente Juan Pablo Berrio Lopez
Seminario de grado AWS
2025

## Agradecimientos

Agradezco de corazón a Uniremington por ayudarme a cumplir mi sueño de ser ingeniero al brindarme las herramientas para poder estudiar y trabajar. Gracias a esto pude desarrollar mi carrera profesional desde el estudio y la experiencia.

## Tabla de Contenidos

Resumen....4
Marco conceptual....5
Marco contextual....5
Introducción histórica....6
Comparación de nombres de servicios de los proveedores de tecnologías de Cloud Computing....6
Implementación de balanceador de carga con Auto Scaling....7
Implementación de balanceador de carga con autoscaling. Puesta en práctica....11
Implementación de infraestructura con contenedores Docker....15
Conclusiones....20
Referencias....21

**Resumen**

Este trabajo se hace con el fin de poner en práctica los conocimientos adquiridos en el seminario de AWS. En éste se emplean los conceptos de creación de instancias de EC2, creación de VPC (virtual private cloud), uso de grupos de seguridad, contenedores Docker y demás estructuras necesarias para hacer funcionar un servidor en la nube que permita realizar el autoescalado de instancias. Es decir, se usará una cuenta de AWS para crear todo un entorno de servidor web a través de instancias, las cuales tendrán la función de autoescalado y de balanceo de cargas. Esto brindará la posibilidad de tener un entorno en la nube que responda dinámicamente las peticiones realizadas al crear más instancias cuando el balanceador de carga lo encuentre necesario.

**Palabras clave**

Cloud Computing, Amazon Web Services, Virtual Private Cloud, Autoescalado, Docker

## Marco conceptual

La computación en la nube, o Cloud Computing, involucra varios conceptos importantes que son esenciales para el desarrollo de un proyecto como éste. En secciones posteriores se brindará la explicación de lo que es Cloud Computing y de cómo sus conceptos pueden tener diferencias dependiendo del proveedor. Por otro lado, varias partes de la infraestructura implementada presentan palabras y acrónimos generalmente tomados del inglés. Algunos términos se pueden traducir al español, pero hacerlo puede complicar su representación en acrónimo. Es por esta razón que suelen utilizarse los nombres en su idioma original.

Comenzando con VPC (Virtual Private Cloud), que es nube privada virtual, es un espacio físico dentro de los servidores de la nube donde se aloja el entorno creado. A nivel conceptual, y desde lo que se implementa desde la plataforma, el VPC es el espacio virtual donde estarán corriendo todos los procesos de bases de datos, servicios especializados e instancias. Éstas últimas son a su vez recursos virtuales que se pueden asociar a la idea de servidor. En una instancia se puede tener información no permanente y correr contenedores o máquinas virtuales. (Amazon Web Services, n.d.).

Adicionalmente, además de los recursos de instancias con máquinas virtuales, existen soluciones aún más optimizadas que permiten usar ciertas características de una máquina virtual, pero sin requerir todo lo que ésta requiere. Una de estas soluciones son los contenedores Docker. Éstos posibilitan hacer uso de partes específicas de recursos computacionales a través de librerías sin tener que instalar o encender todo un sistema operativo. (Docker, Inc., n.d.).

Por último, el Auto Scaling o autoescalado es un término que hace referencia al aumento de recursos de computación automático a través de mecanismos que miden la necesidad de respuesta de éstos. Por ejemplo, si se tiene una infraestructura en la nube que tiene un sistema que responde a peticiones a través de interfaz web; y si el número de peticiones aumenta a un umbral que satura los recursos del sistema, el autoescalado permite detectar esta situación y aumentar los recursos de manera automática con base en la demanda. (Amazon Web Services, n.d.).

## Marco contextual

Todo el desarrollo de este proyecto se realiza dentro de una cuenta Root (Cuenta principal) de consola de AWS. Esto es una interfaz Web proporcionada por Amazon desde la que se pueden gestionar todos los recursos. Las imágenes presentes en este proyecto son de dicha interfaz. Se usarán direcciones IP públicas y privadas para acceder de manera remota a los recursos y para el funcionamiento de diversas partes de la infraestructura implementada.

## Introducción histórica

La computación en la nube es un concepto que ha existido desde las primeras etapas de la computación en general. Sin embargo, su uso ha tomado mucha más fuerza en las décadas recientes. En la siguiente línea de tiempo se mencionan los puntos históricos más importantes para el cloud computing y se explica brevemente el por qué esta práctica ha avanzado tanto y los factores que han propiciado su expansión.

1960‘s: En los inicios de la computación el concepto de Cloud computing surgió dada la gran cantidad de recursos que necesitaban los computadores. En ese tiempo tener un computador empresarial o personal no era algo viable. El concepto de usar la computación como un servicio externo fue algo que se empezó a idear. Sin embargo, con el avance de las décadas, los computadores se volvieron más asequibles y tanto empresas como individuos tuvieron la oportunidad de acceder a los recursos de computación en sus propios dispositivos sin tener que compartirlos.

2000’s: Aunque sucedieron algunos eventos importantes entre este punto y el anterior, están fuera del alcance del presente escrito. Se salta directamente a los 2000’s cuando Amazon comienza su servicio de computación en la nube. Este fue el primer servicio de Cloud que tuvo suficiente estructura y expansión como para que muchos individuos y empresas empezaran a ver viable usar servicios de Cloud, en vez de sus servicios on Premise (Servicios de computación propios como servidores físicos propiedad de empresas o individuos). AWS, como se abrevia Amazon Web Services, empezó a generar soluciones en la nube que competían muy fuertemente con sus contrapartes locales. Esto tuvo gran éxito y propició que otras compañías empezaran a sacar sus soluciones de nube también.

2010’s: A pesar de que existen múltiples compañías de servicios en la nube y que han aportado ampliamente en el crecimiento de estos servicios, entre los más importantes para mencionar en esta línea de tiempo son Google y Microsoft, quienes lanzaron Google Cloud Platform y Microsoft Azure. Estas compañías respondieron a las nuevas necesidades de los mercados tecnológicos y aumentaron aún más la posibilidad de las personas para acceder a servicios de computación en la nube.

2020’s: Para este punto, más compañías han lanzado sus propias soluciones de computación en la nube y cada vez se vuelve mucho más competitivo respecto a las soluciones de servidores locales. Otras empresas importantes como IBM o Oracle también han entrado al mercado compañías de todos sectores usan los servicios en la nube.

## Comparación de nombres de servicios de los proveedores de tecnologías de Cloud Computing

Existe una gran variedad de servicios de nube en cada proveedor importante. Para el propósito de este trabajo, realizaremos sólo la comparación de algunos servicios básicos usados en esta primera entrega y que tienen nombres distintos en Microsoft Azure y GCP. El primer concepto sería el de instancia de EC2, el cual se conoce como Virtual machine y Compute Engine en los proveedores mencionados. De la misma manera, una AMI (Amazon Machine Image) se conoce en éstos como Managed image and Machine image. Siguiendo esta estructura el Auto Scaling de AWS se conoce en Microsoft Azure y GCP como Virtual Machine Scale Set y Managed Instance Groups respectivamente.

## Implementación de balanceador de carga con Auto Scaling

A continuación se presenta la implementación mencionada en una cuenta de AWS. El proceso consiste en crear una VPC donde funcionará todo el entorno, posteriormente crear una instancia Linux de EC2 de la cual se crea una AMI. Esta Amazon Machine Image, que ya tiene configurada una aplicación web básica, se utiliza para que sean creadas más AMIs a partir de esta cuando el uso de CPU supere un umbral dado. Para esto se utiliza una política de autoscaling que revisará el uso de CPU para crear más AMIs cuando se supere el umbral, y un balanceador de carga para dirigir el tráfico a las diferentes AMIs creadas. Se procederá a mostrar las imágenes de la configuración directamente y se agregarán las explicaciones en las descripciones de estas.

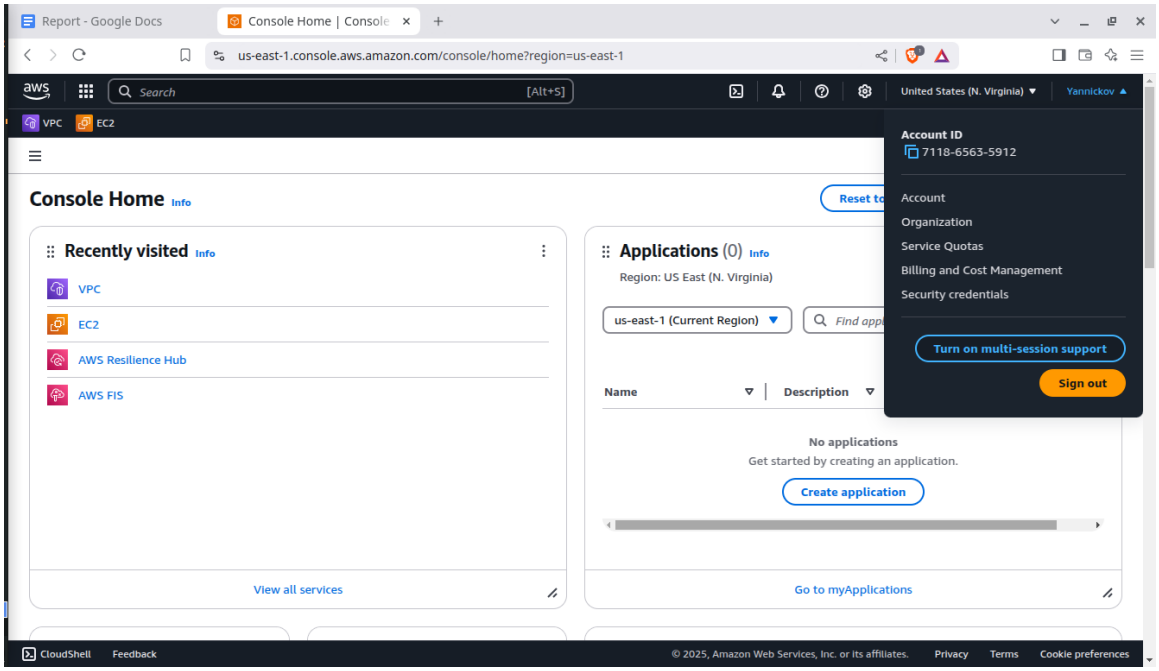


*Imagen 1. Cuenta de AWS 711865635912 trabajando en el us-east-1. N. Virginia*

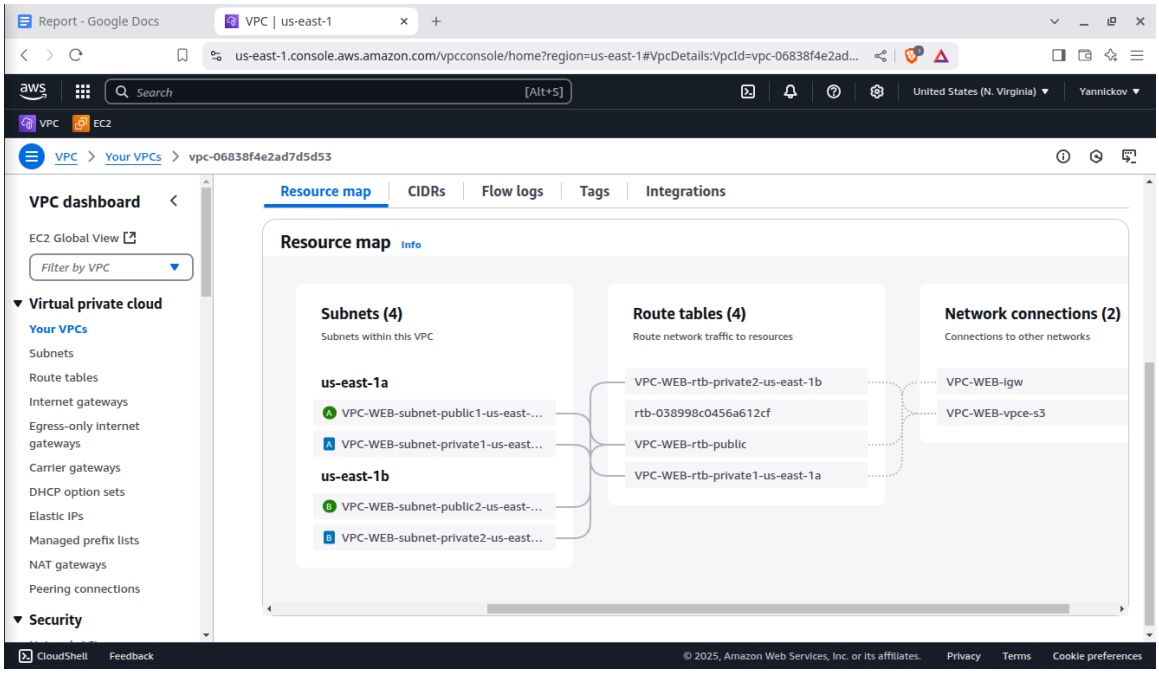


*Imagen 2. VPC creada con dos subredes públicas y dos subredes privadas. Esto es importante para poder acceder a los servicios desde cualquier acceso a internet (redes públicas) y para manejar internamente procesos que no deben ser accesibles por todos los usuarios (redes privadas)*

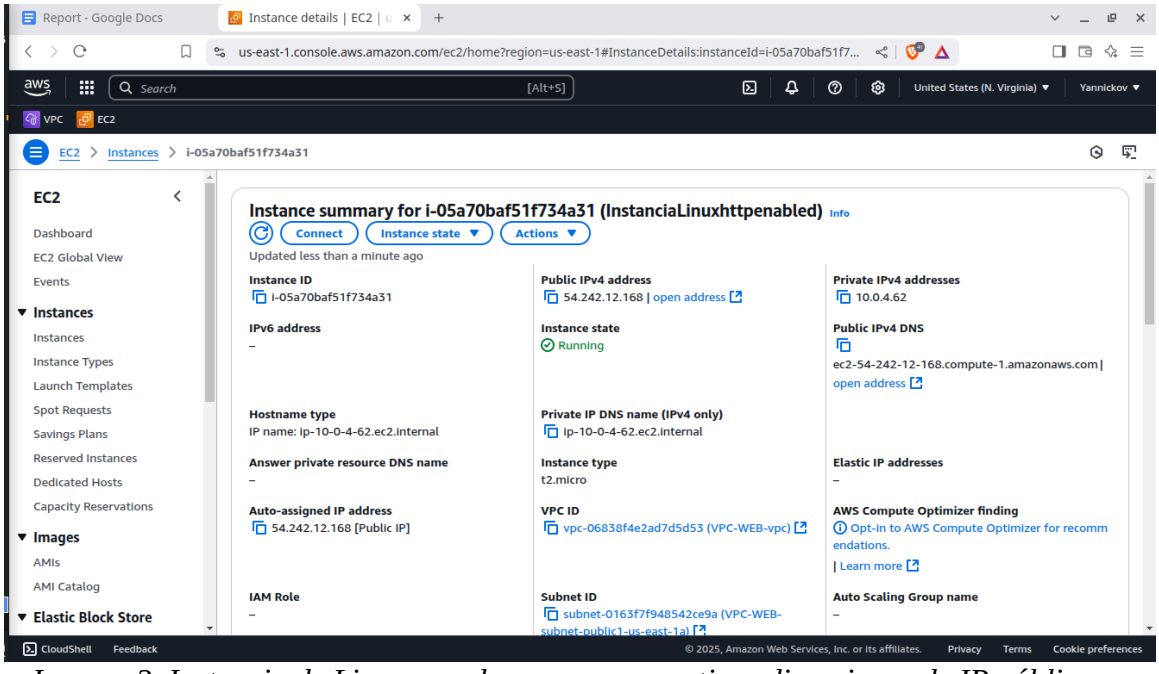


*Imagen 3. Instancia de Linux creada con sus respectivas direcciones de IP públicas y privadas y con una aplicación web de prueba visible en la siguiente imagen.*

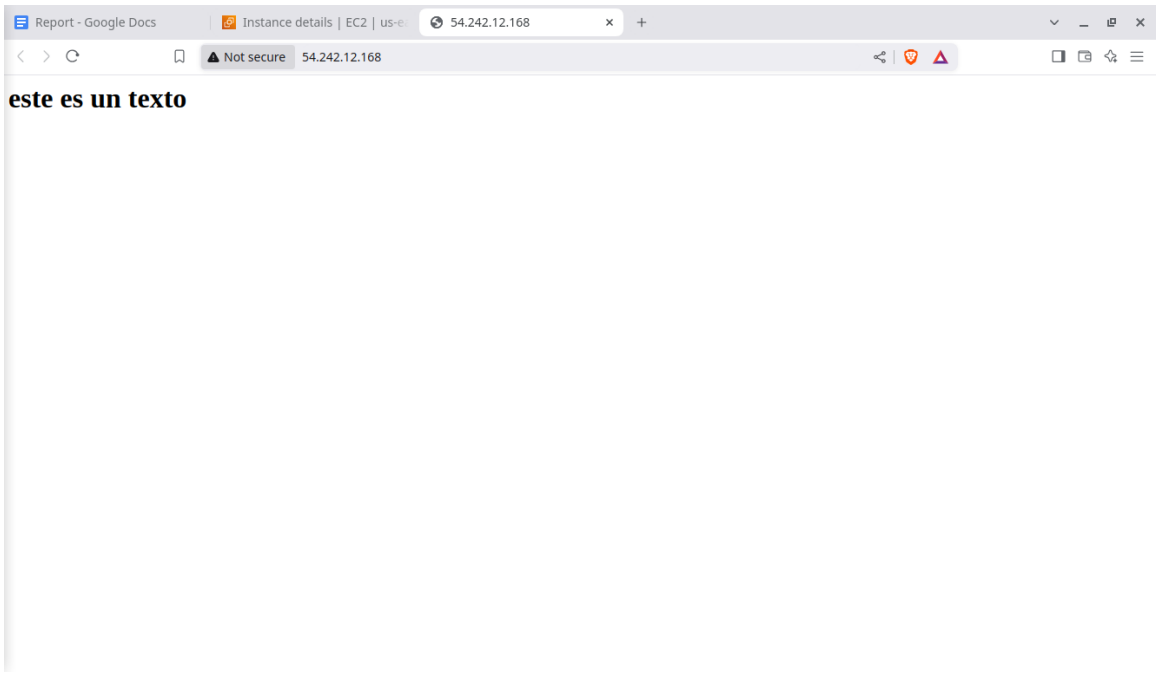


*Imagen 4. Aplicación web de la instancia Linux con un tag <h1> de HTML*

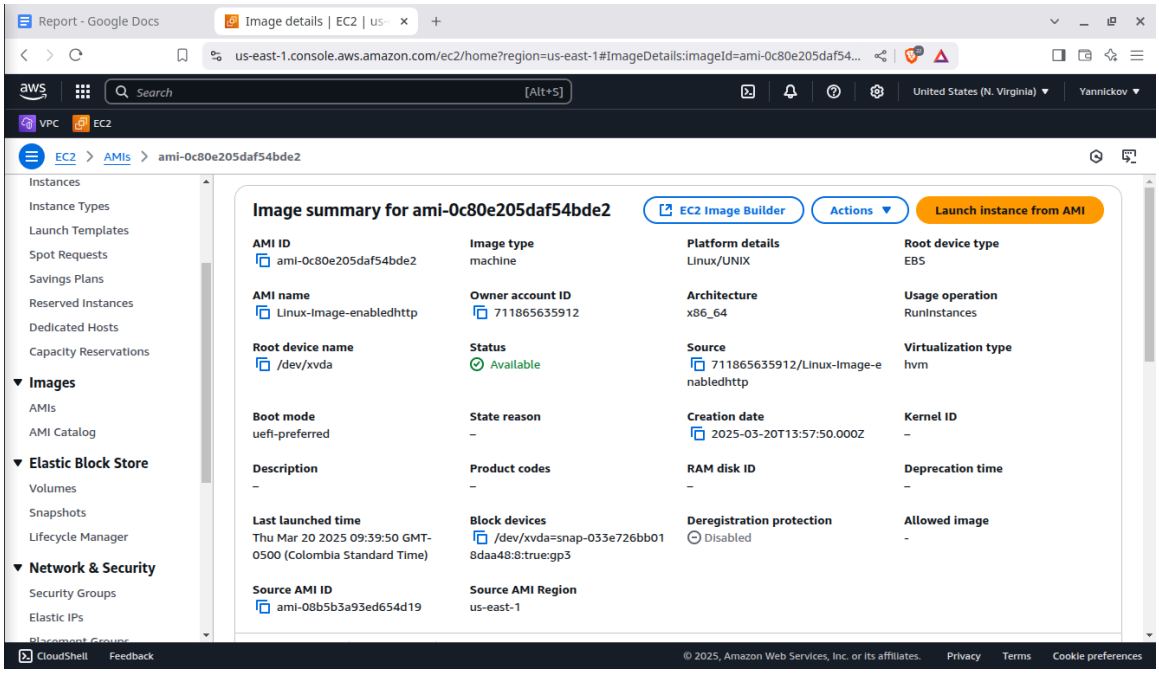


*Imagen 5. AMI creada de la instancia Linux*

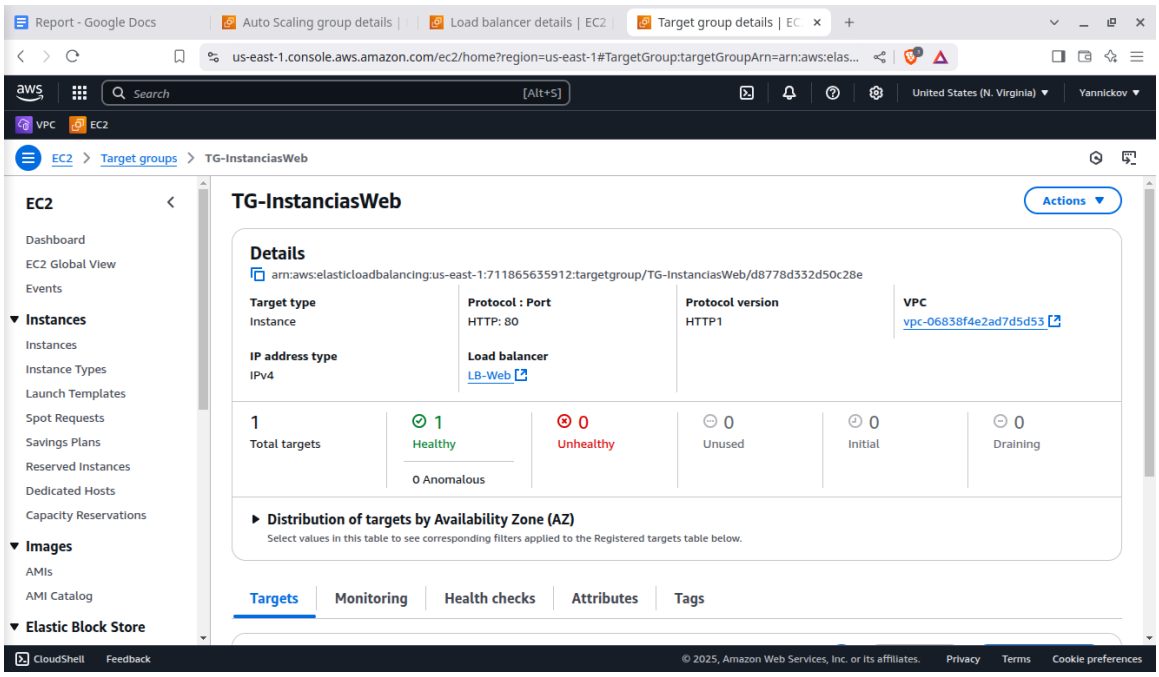


*Imagen 6. Target group asociado al Load Balancer*

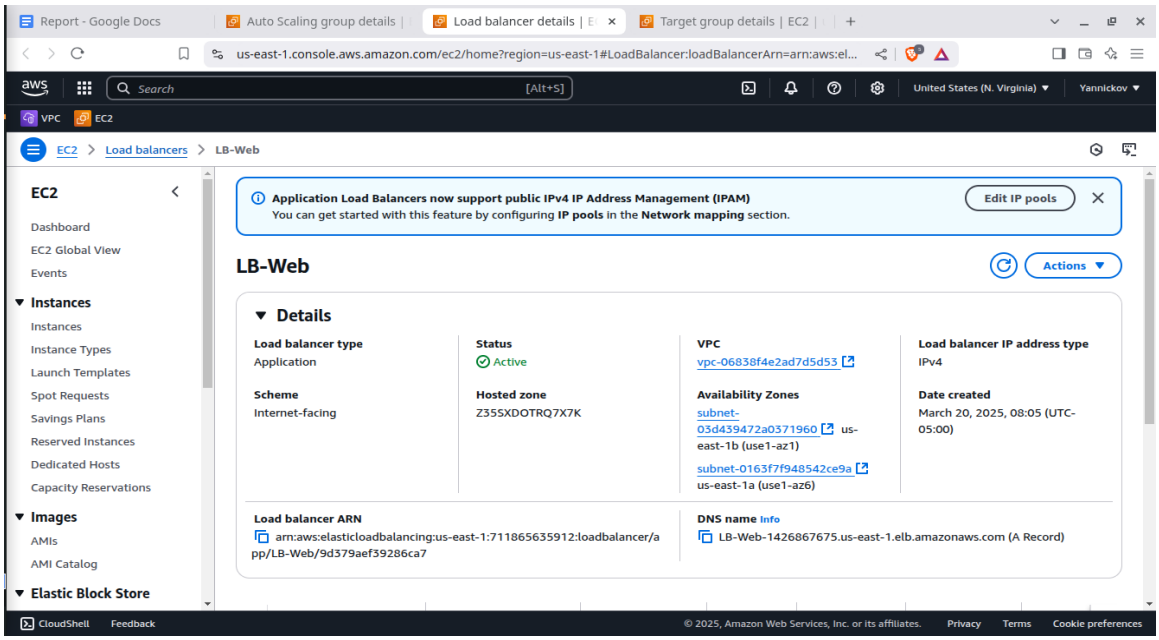


*Imagen 7. Load Balancer que se usará en el Auto Scaling Group*

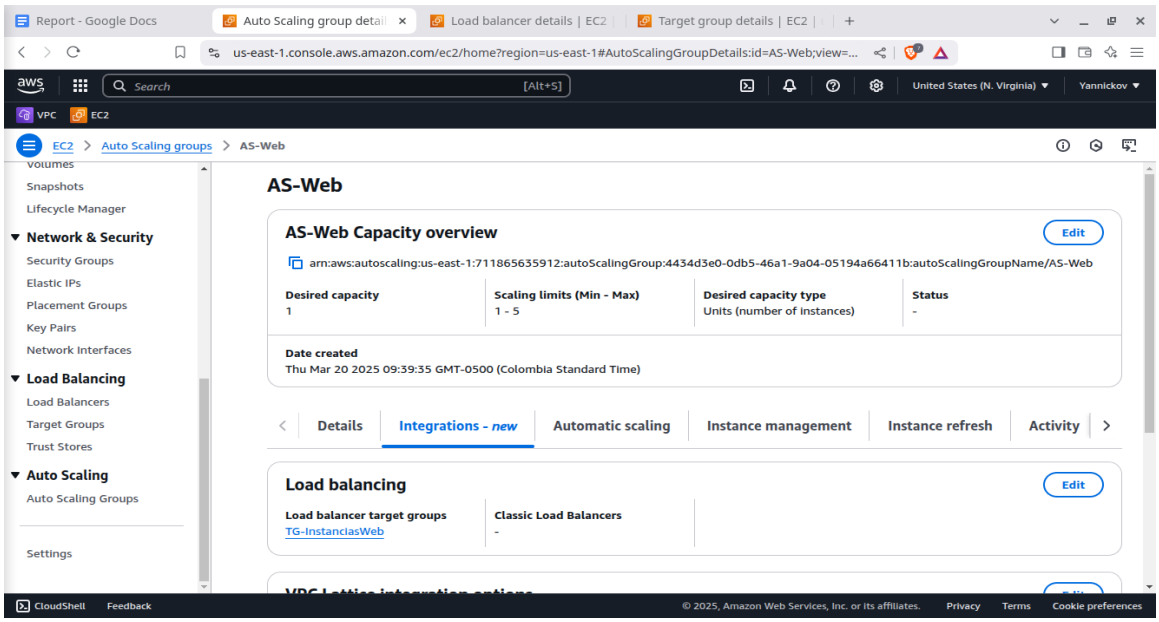


*Imagen 8. Auto Scaling Group donde se aprecian tres factores importantes. El Load Balancer asociado (Imagen 7), la capacidad deseada por defecto (1) y los límites de escalamiento (en este caso es mínimo 1 y máximo 5)*

**Implementación de balanceador de carga con autoscaling. Puesta en práctica**

En las imágenes anteriores se ha evidenciado la configuración para implementar el balanceador de carga con auto scaling. A continuación se muestra cómo se pone en práctica esta configuración. Para esto se revisará el umbral definido de la política de auto scaling, se hará una prueba de estrés en la instancia EC2 y se revisarán los logs del Auto Scaling Group para ver los cambios.

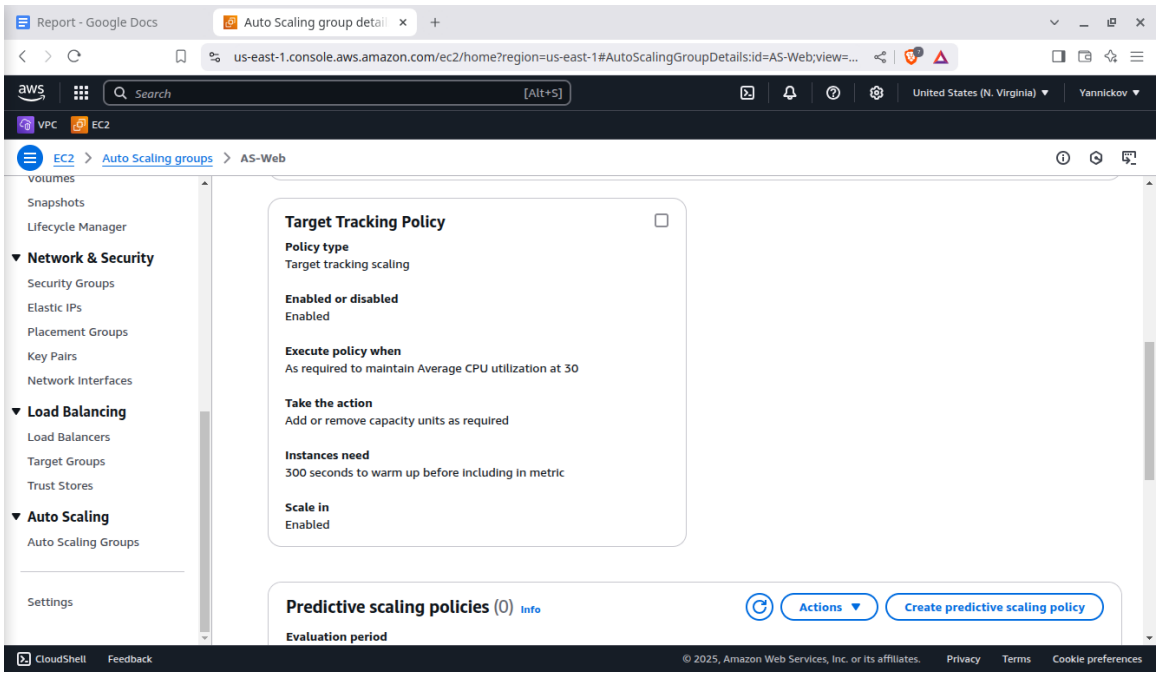


*Imagen 9. El umbral de uso de CPU está al 30%. Con esto se da respuesta al punto 3 del trabajo donde pregunta por el umbral definido por la política de auto scaling.*

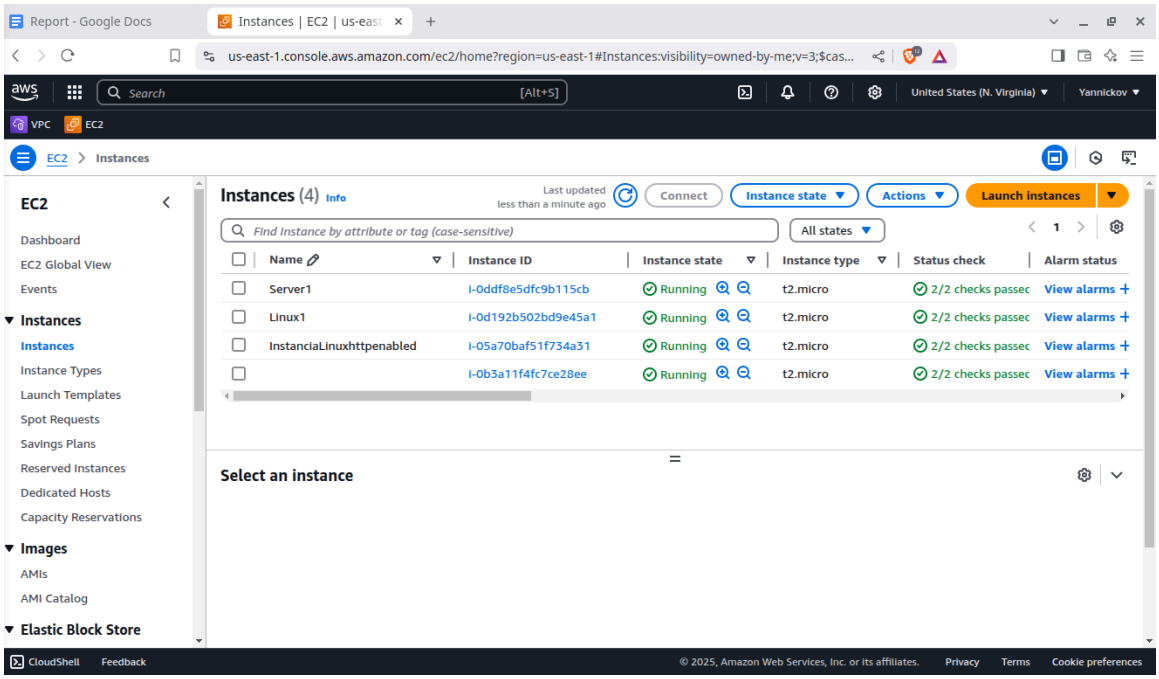


*Imagen 10. Estado inicial del sistema sin haber hecho ningún cambio. La instancia sin nombre es la que se ha creado automáticamente por el Auto Scaling Group.*

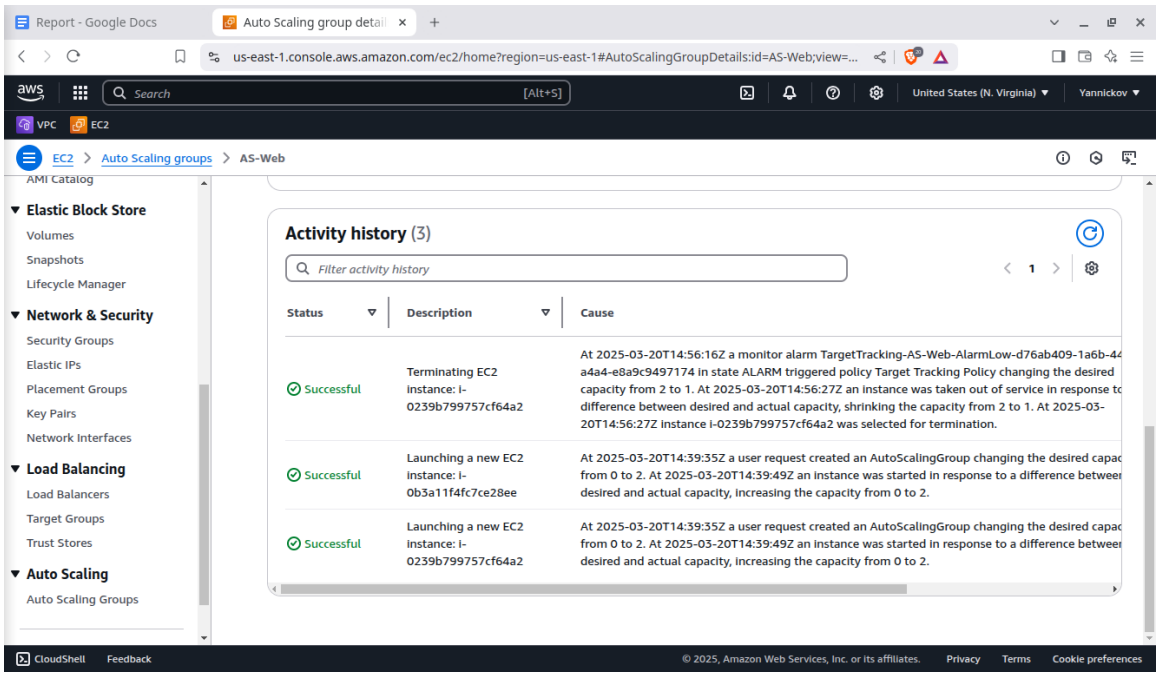


*Imagen 11. Estado inicial de los Logs del Auto Scaling Group*

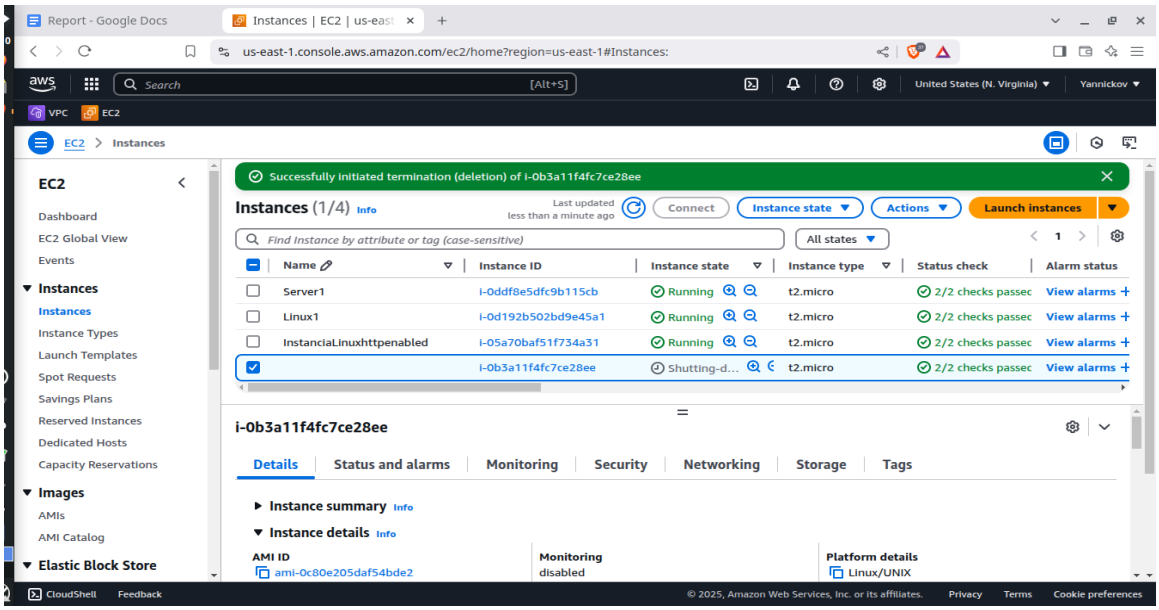


*Imagen 12. Se termina manualmente la instancia del Auto Scaling Group. Esto significa que el número de instancias de éste se vuelve cero, lo cual hará que cree una nueva instancia.*

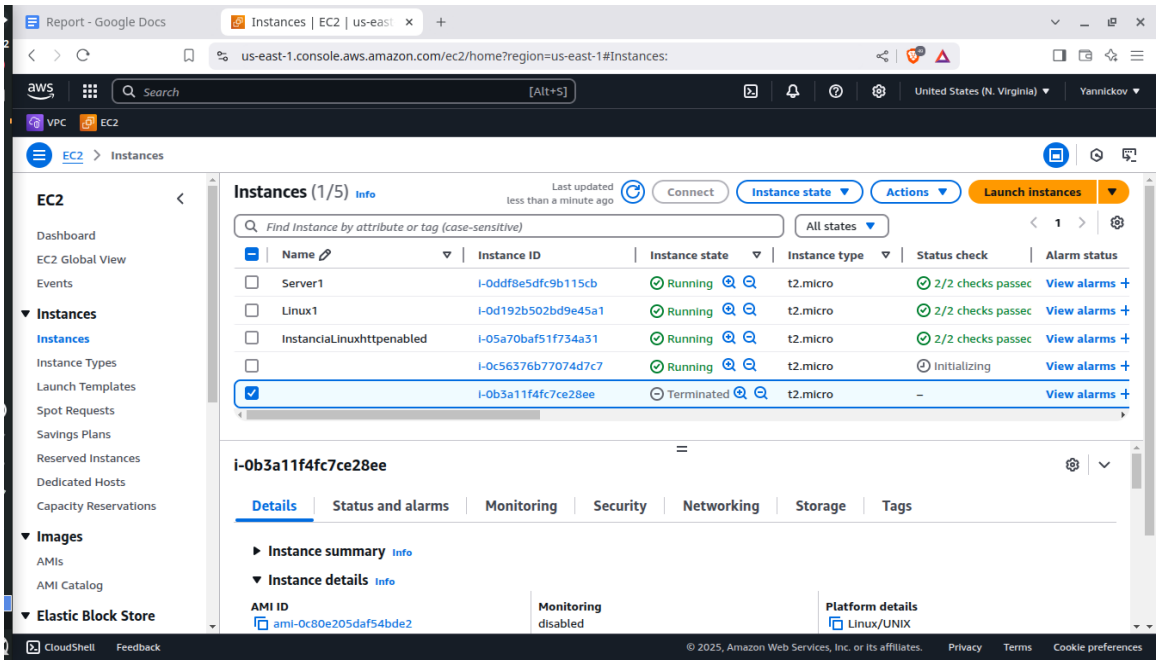


*Imagen 13. Se evidencia en los siguientes segundos cómo hay una nueva instancia en estado de "Initializing"*

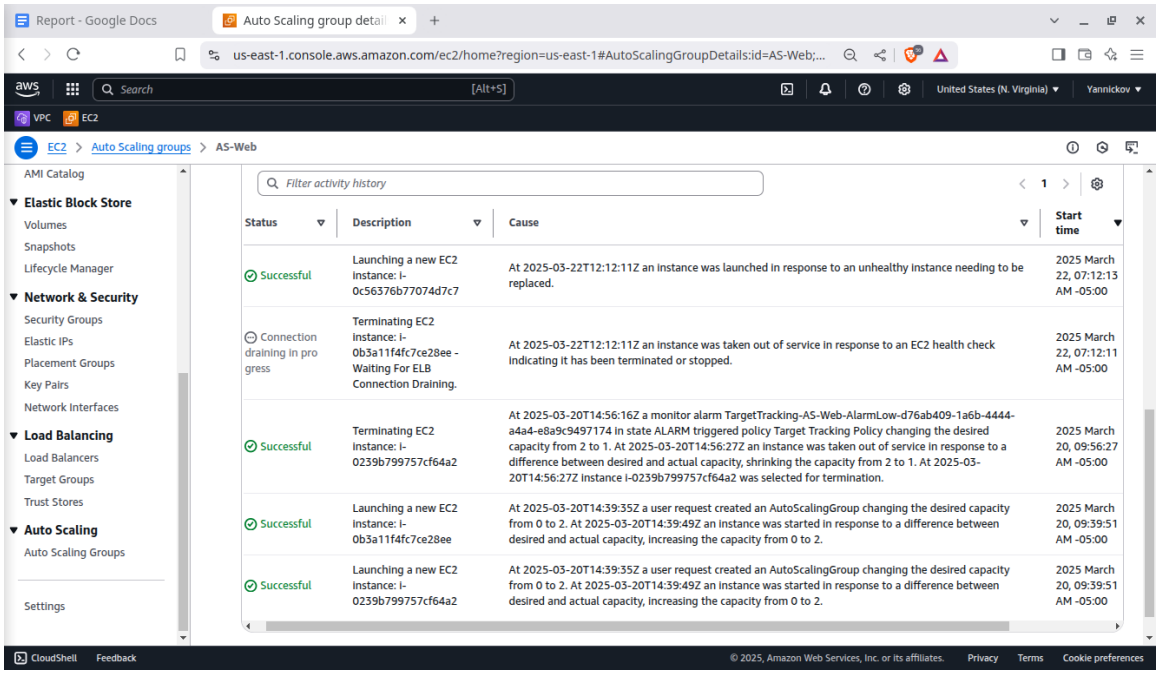


*Imagen 14. Se evidencia en los Logs del Auto Scaling Group la creación de una nueva instancia en respuesta a la terminación de la instancia anterior*

## Implementación de infraestructura con contenedores Docker

Los contenedores Docker son una herramienta alternativa a las máquinas virtuales en los que no se debe cargar toda la información de un sistema operativo, sino que se usan librerías. Esto permite utilizar sólo los recursos necesarios y tener tiempos de lanzamiento más rápidos en comparación al uso de una máquina virtual. Dado que estos contenedores Docker se pueden tener en recursos de servidores, tener una configuración de Docker en una instancia de EC2 es totalmente posible. En esta práctica se desarrolla un sistema con contenedores Docker en una instancia de Amazon Linux. A continuación se presentan las imágenes correspondientes a la configuración del entorno para lograr el acceso a contenedores Docker desde una interfaz web y con diferenciación de nombre de dominio.

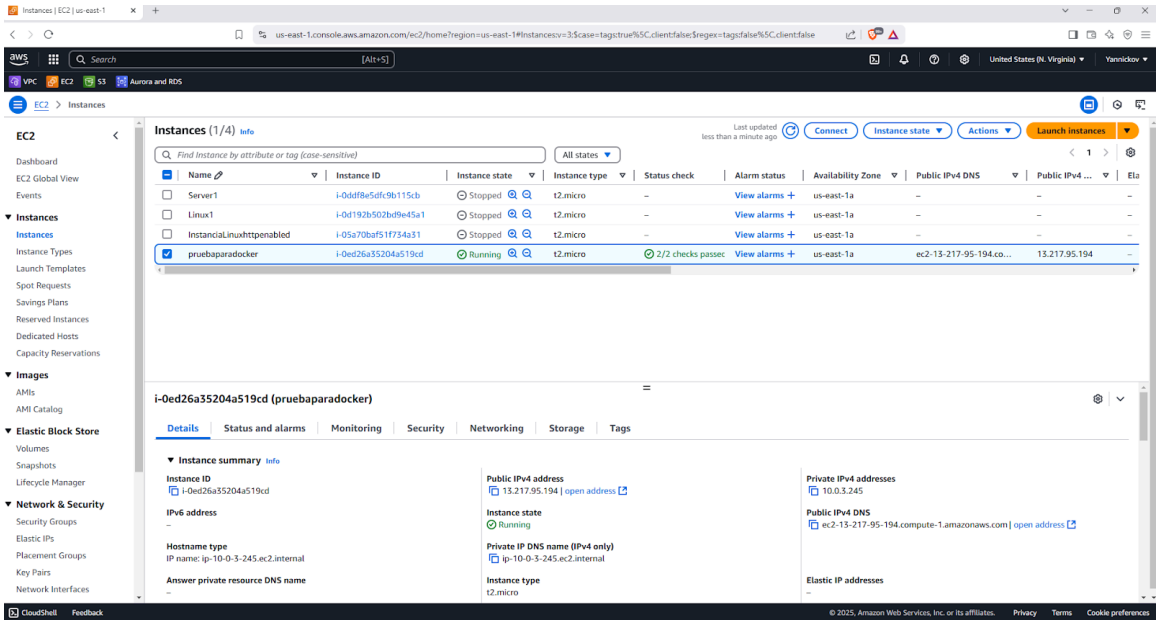


*Imagen 15. Instancia de Linux corriendo donde se alojarán los contenedores Docker*

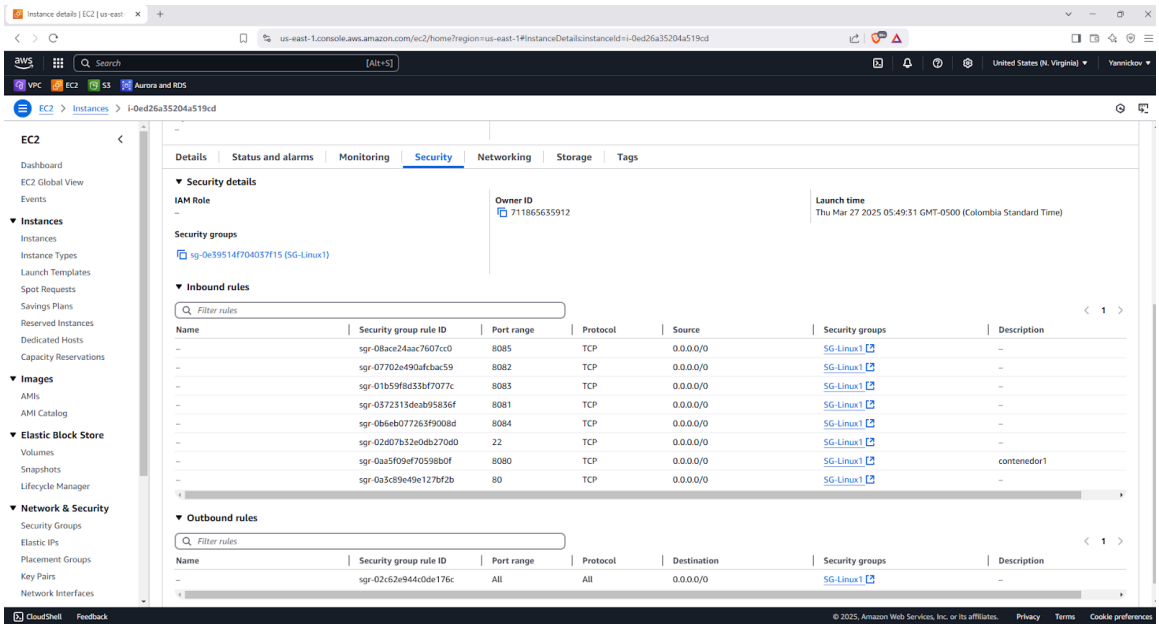


*Imagen 16. Vista previa de grupo de seguridad usado que permite el acceso a los puertos 8081, 8085 y 8083, entre otros. Estos son los puertos que serán usados.*

```
[root@ip-10-0-3-245 ec2-user]# ls
app1  app2  app3  carpeta1  file1
[root@ip-10-0-3-245 ec2-user]# cat ./app1/index.html
<h1 style="color:blue">Hola, esta es la pagina de Yannick Azul</h1>
[root@ip-10-0-3-245 ec2-user]# cat ./app2/index.html
<h1 style="color:green">Esta es la pagina de Yannick Verde<h1>
[root@ip-10-0-3-245 ec2-user]# cat ./app3/index.html
<h1 style="color:red">Esta es la pagina de Yannick Rojo</h1>
```

*Imagen 17. Se tienen tres directorios con archivos html en cada uno*

```
[root@ip-10-0-3-245 nginx]# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED       STATUS       PORTS                                       NAMES
a1feb9dac7df   httpd     "httpd-foreground"   2 hours ago   Up 2 hours   0.0.0.0:8083->80/tcp, :::8083->80/tcp       app3
0347f77fb6f7   httpd     "httpd-foreground"   3 hours ago   Up 3 hours   0.0.0.0:8085->80/tcp, :::8085->80/tcp       app5
5ab230471136   httpd     "httpd-foreground"   3 hours ago   Up 3 hours   0.0.0.0:8084->80/tcp, :::8084->80/tcp       app4
2e38b4d3df31   httpd     "httpd-foreground"   3 hours ago   Up 3 hours   0.0.0.0:8082->80/tcp, :::8082->80/tcp       app2
89895a14bf59   httpd     "httpd-foreground"   3 hours ago   Up 3 hours   0.0.0.0:8081->80/tcp, :::8081->80/tcp       app1
ef9dd3bd512a   httpd     "httpd-foreground"   3 hours ago   Up 3 hours   0.0.0.0:8080->80/tcp, :::8080->80/tcp       apachev1
```

*Imagen 18. Los contenedores Docker funcionando*

```
# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
13.217.95.194 www.yannickazul.com
13.217.95.194 www.yannickverde.com
13.217.95.194 www.yannickrojo.com
```

*Imagen 19. Últimas líneas de archivo hosts en C:\Windows\System32\drivers\etc. Todas ellas apuntando a la IP de la instancia de EC2*

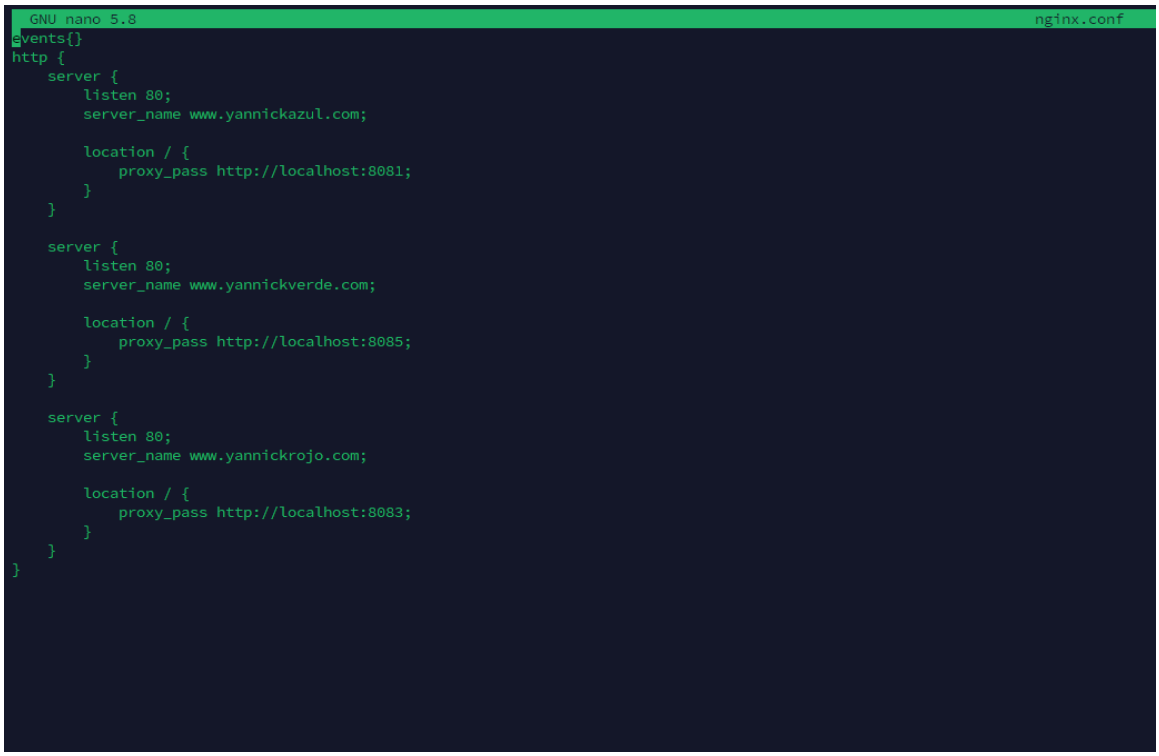


*Imagen 20. Configuración de nginx.conf donde se tienen tres nombres de dominio distintos de los cuales las peticiones serán enviadas a tres puertos correspondientes distintos*


*Imagen 21. Correcto funcionamiento al ingresar a un nombre de dominio específico*

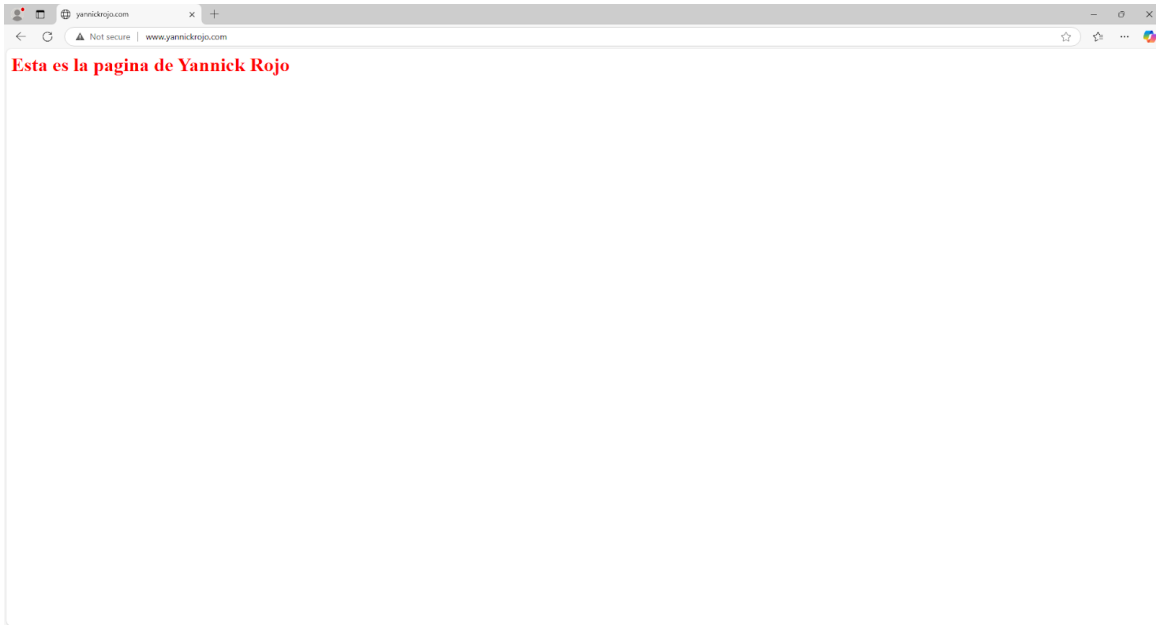


*Imagen 22. Correcto funcionamiento al ingresar a un nombre de dominio específico desde Edge*


*Imagen 23. Correcto funcionamiento al ingresar a un nombre de dominio específico*

## Conclusiones

Después de lograr una implementación exitosa de la infraestructura planteada, se encuentra una explicación basada en resultados al por qué las tecnologías de computación en la nube han aumentado su importancia en los años recientes. El proyecto desarrollado presenta una gran adaptabilidad a situaciones de la vida real y su gasto en recursos económicos fue casi nulo. En comparación, si se hubiese realizado el mismo proyecto en servidores tradicionales, el gasto económico habría sido considerable, comenzando por la disponibilidad que requiere el acceso a un servidor o máquina que lo emule. Dicha disponibilidad fue inmediata al crear la cuenta de AWS.

Consecuentemente, las posibilidades de aprendizaje son también de alto valor. La oportunidad de acceder de manera inmediata a recursos de servidores, además de crear y eliminar instancias en cuestión de segundos presenta una ventaja al aprendizaje, en comparación al aprender la utilización de recursos de un servidor normal. Si bien es cierto que existen emuladores y recursos de aprendizaje, es también pertinente considerar que al crear una infraestructura como la que se hizo en este proyecto el paso a utilizarla en proyectos reales es inmediato. Esto es diferente en los recursos de servidores tradicionales, donde incluso con el acceso a emuladores o máquinas virtuales, el paso de un simulacro a un escenario de la vida real puede involucrar varios desafíos.

En contraste a esto, es también relevante mencionar que aunque las tecnologías en la nube ofrecen gran cantidad de ventajas en comparación a servidores tradicionales, éstos últimos aún conservan espacios donde pueden ser esenciales. Es posible que una empresa tenga necesidades específicas a nivel de seguridad, cumplimiento de políticas públicas o internas en las que una solución con servidores físicos sea más ventajosa.

El razonamiento anterior lleva a otra conclusión notable. El análisis de las necesidades específicas de cualquier entorno productivo es primordial. Incluso con las grandes ventajas que presentan las tecnologías en la nube, considerando también su gran variedad de soluciones optimizadas, el estudio de requisitos sigue siendo crítico. Para lograr una utilización óptima de los recursos computacionales, se deben realizar el análisis, la implementación y la evaluación correcta de la infraestructura.

Por último se concluye, con lo que se evidenció en la infraestructura creada, que la utilización de recursos es optimizada como se esperaba. Esto en el sentido de que se creó un sistema que respondió automáticamente a los cambios en uso de recursos, utilizando únicamente aquellos recursos que necesitaba para responder a las demandas de computación y dejando de usarlos al detectar que no los necesitaba más.

## Referencias

Amazon Web Services (n.d.). *Amazon EC2 Auto Scaling documentation.* Recuperado de https://aws.amazon.com/autoscaling/

Amazon Web Services. (n.d.). *Amazon Elastic Compute Cloud (EC2) documentation*. Recuperado de https://aws.amazon.com/ec2/

Amazon Web Services (n.d.). *Amazon Virtual Private Cloud (VPC) documentation.* Recuperado de https://aws.amazon.com/vpc/

Amazon Web Services. (n.d.). *Tutorial: Set up a scaled and load-balanced application.* Recuperado de https://docs.aws.amazon.com/autoscaling/ec2/userguide/tutorial-ec2-auto-scaling-load-balancer.html

Amazon Web Services. (n.d.). *Use Elastic Load Balancing to distribute incoming application traffic across your Auto Scaling group.* Recuperado de https://docs.aws.amazon.com/autoscaling/ec2/userguide/autoscaling-load-balancer.html

Docker (n.d.). *What is Docker?* Recuperado de https://www.docker.com/what-docker