TRABAJO DE GRADO
Opción Seminario-Diplomado.

Infraestructura virtual en la nube con aws

Corporación Universitaria Remington.
Facultad de ingenierías
Ingeniería de Sistemas

Santiago Henao Monsalve
Yeison Fernando Berrio Zuleta
Johan Sebastian Mejia Ospina

Juan Pablo Berrio López

Opción de Trabajo de grado Seminario-Diplomado.
2025

Tabla de Contenidos

Resumen 3
Marco conceptual y contextual 3
¿Qué es Amazon Web Services (AWS) 3
¿Por qué aprender a utilizar AWS? 3
Desarrollo e implementación del aprendizaje 7
Figuras y tablas 21
Conclusiones 22
Referencias 23

Resumen

El propósito de este proyecto es aprender sobre los servicios y la ejecución de una solución basada en Amazon Web Services (AWS), en el despliegue de servicios web escalables y eficientes mediante tecnologías de virtualización y contenedores. El proyecto nace a raíz de la participación en un seminario sobre AWS que aborda conceptos clave de computación en la nube, automatización y arquitectura distribuida. Este proyecto refleja un enfoque práctico de uso de herramientas en la nube, mostrando la capacidad de implementar entornos funcionales que integren contenedores, servicios web, balanceo de carga y automatización sobre AWS. La solución desarrollada puede servir como base para futuras aplicaciones empresariales que demanden disponibilidad continua, despliegue ágil y optimización de recursos tecnológicos.

Palabras clave

AWS, cloud, Servicio EC2, Servicio VPC, Servicio S3, balanceador de carga.

Marco conceptual y contextual

¿Qué es Amazon Web Services (AWS)

Amazon Web Services (AWS) es una plataforma completa de servicios en la nube que ofrece infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS), siendo estos últimos usados a nivel mundial por empresas, instituciones educativas y entidades del gobierno. Fue oficialmente puesta en marcha en 2006 con su primer servicio: Amazon S3 (Servicio de Almacenamiento Simple), y desde aquel momento se ha establecido como el referente global en servicios cloud, sobrepasando a rivales como Microsoft Azure y Google Cloud Platform. AWS brinda a los usuarios la posibilidad de ejecutar aplicaciones sin la necesidad de gestionar servidores físicos, proporcionando servicios bajo demanda, escalabilidad, pago por uso, alta disponibilidad y seguridad en el ámbito corporativo.

¿Por qué aprender a utilizar AWS?

Dominar AWS se ha convertido en una competencia clave en el mercado laboral actual por las siguientes razones:

- Demanda laboral: AWS es la plataforma en la nube más utilizada del mundo.
- Versatilidad: Permite el desarrollo de sitios web, bases de datos, aplicaciones móviles, soluciones de big data, inteligencia artificial, y mucho más.
- Accesibilidad: Ofrece una cuenta gratuita con uso limitado de múltiples servicios.

- Apoyo y comunidad: aws posee una gran comunidad de usuarios, foros de soporte, certificaciones oficiales y documentación amplia.

1. ¿Qué es Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web de AWS que brinda capacidad informática segura y escalable en la nube. Está creado para facilitar a los desarrolladores el acceso a recursos de cómputo virtualizados, permitiendo el despliegue rápido y flexible de aplicaciones y sistemas operativos.

1.2 Beneficios clave de EC2:

- Escalabilidad automática.
- Variedad de tipos de instancia para distintos usos.
- Modelo de pago por uso.
- Alta disponibilidad mediante zonas de disponibilidad y regiones.
- Integración con otros servicios de AWS.

1.3 Tipos de instancias EC2

AWS ofrece una amplia diversidad de instancias EC2, las cuales están clasificadas en familias y optimizadas para diferentes casos de uso:

- Propósito general (t2, t3, t4g): Equilibradas en computación, memoria y red.
- Optimización de cómputo (c5, c6g): Sirven para procesamiento intensivo como aplicaciones científicas.
- Optimización de memoria (r5, x1e): Ideal para bases de datos y análisis en memoria.
- Instancias aceleradas (p4, inf1): Para cargas de trabajo con GPU, como IA y ML.
- Almacenamiento optimizado (i3, d2): Adecuadas para bases de datos NoSQL, Hadoop, etc.

1.4 VPC (Virtual Private Cloud)

Las VPC nos permiten definir y gestionar todo aquello relacionado con las redes en nuestro entorno de AWS, las subredes públicas y privadas, el direccionamiento IP interno, tablas de ruteo, para poder gestionar accesos y segmentación de redes de acuerdo a los requerimientos específicos de cada proyecto, podemos crear varios VPC.

- Rango de direcciones IP privadas.
- Subredes públicas y privadas.
- Tablas de ruteo.
- Puertas de enlace de Internet y NAT.

1.5 Lanzamiento de una instancia EC2 paso a paso

Para nosotros, Auto Scaling de AWS ha sido una ayuda genial. Básicamente, lo que hace es cambiar solita la cantidad de máquinas EC2 que tenemos encendidas, dependiendo de la carga de trabajo que le hayamos dicho que mire. La idea es que la aplicación siempre vaya fina y que no se nos disparen los gastos. Y dentro de esto, usamos los Grupos de Escalado Automático (ASG). Piensa en ellos como equipos de esas máquinas EC2 que van juntas. Se organizan solas para poner más máquinas si de repente hay mucha gente usando la aplicación, o quitar las que no hacen falta si la cosa se calma.

1.6 Seguridad en EC2: Grupos de Seguridad

Para nosotros es clave configurar bien estos grupos, sobre todo para evitarnos problemas y ataques. Piensen que los grupos de seguridad en AWS son como unos firewalls virtuales: controlan qué puede entrar y salir de las instancias según las reglas que pongamos.

- Protocolo (TCP, UDP, ICMP).
- Rango de puertos.
- Fuente o destino (IP, otro grupo de seguridad).

1.7 DNS y Elastic IP

Cuando levantamos una instancia EC2, vimos que AWS le pone por defecto una IP pública que va cambiando. Para evitar eso, sobre todo si se reinicia, le podemos poner una Elastic IP, que es fija. AWS también nos da nombres DNS, tanto públicos como privados, para que los servicios se puedan comunicar más fácil.

1.8 Imágenes AMI y Templates de Presentación

Para nosotros, las imágenes de máquina Amazon (AMI) son geniales porque nos dejan guardar toda la configuración de una instancia EC2. Esto nos ayuda un montón a la hora de replicar entornos fácilmente. Y con las plantillas de lanzamiento, reutilizamos las configuraciones típicas (tipo de instancia, AMI, almacenamiento, scripts de arranque) y mantenemos un control de versiones.

1.9 Balanceador de Carga

Para nosotros el Elastic Load Balancing (ELB) de AWS. Es un servicio que nos monta AWS y que nos ayuda un montón a repartir el tráfico que llega (sea de red o de apps) entre nuestros distintos recursos: instancias EC2, contenedores, IPs, lo que tengamos.

El objetivo principal que buscamos con esto es que nuestras aplicaciones en la nube estén siempre arriba, puedan escalar bien si hace falta y aguanten mejor los fallos.

Tipos de load Balancer:

- Aplicación Load Balancer (ALB): Ideal para aplicaciones web a nivel de capa 7 (HTTP/HTTPS), permite balancear tráfico por URL, host, método HTTP, etc.
- Network Load Balancer (NLB): Funciona a nivel de red (capa 4), ideal para cargas de trabajo que requieren baja latencia y alto rendimiento.
- Gateway Load Balancer: Se utiliza para integrar fácilmente dispositivos virtuales de terceros (como firewalls o sistemas de inspección).

2.0 Grupos de Auto Scaling y Auto Scaling (ASG)

Nosotros usamos Auto Scaling de AWS. Básicamente, nos deja cambiar automáticamente la cantidad de instancias EC2 que tenemos funcionando, según las reglas que le ponemos nosotros mismos, dependiendo de la demanda. El objetivo es que todo funcione bien y mantener los costos a raya.

Para eso están los Grupos de Escalado Automático (ASG). Son grupos de esas instancias EC2 que manejamos como un bloque. Se ajustan solos: si hay más trabajo, añaden más máquinas; si el trabajo baja, quitan las que sobran.

2.1 Docker

Para nosotros, Docker ha sido una herramienta clave. Es una plataforma de código abierto que nos simplifica un montón el trabajo de crear, mover y poner en marcha nuestras aplicaciones. Lo hacemos usando contenedores, que son como paquetes ligeros, fáciles de llevar de un sitio a otro y que se gestionan bastante bien solos. Lo bueno es que dentro de cada contenedor metemos todo lo que la aplicación necesita para funcionar sin problemas: el código nuestro, las librerías, otras dependencias y hasta las configuraciones. Así nos aseguramos de que la aplicación corra bien en cualquier lado, da igual si es en nuestras máquinas mientras desarrollamos, en el entorno de pruebas o ya directamente en producción.

2.2 NGINX

Es un servidor web que hemos visto que tiene un buen rendimiento, tamaño más reducido y es de código abierto, que también actúa como un balanceador de carga, proxy inverso y servidor de caché. Se usa en muchas ocasiones para alojar aplicaciones en línea en la nube, optimizar el desempeño de las páginas web y nos ayuda a gestionar grandes cantidades de tráfico ayudándonos así en la reducido de recursos.

Desarrollo e implementación del aprendizaje

Bueno, para asegurarnos de que nuestra instancia en AWS funcione como la seda y esté lista para aguantar lo que le echen (ya sea mucho tráfico o algún que otro problemilla), tuvimos que poner a trabajar juntos varios recursos y servicios de AWS.

Lo primero fue levantar la instancia EC2. Básicamente, es como montar un servidor virtual con Linux, pero todo desde la consola de AWS. Como cualquier servidor que montas tú mismo, tuvimos que elegirle unas cuantas cosas: qué tipo de máquina usar, el sistema operativo, cuánto disco duro darle... ah, y muy importante, creamos una clave SSH para poder conectarnos de forma segura usando la línea de comandos.

Una vez tuvimos la máquina base lista, nos pusimos con el grupo de seguridad. Piensen en esto como el portero o el firewall virtual de la instancia. Aquí es donde definimos las reglas del juego: qué tipo de conexiones dejamos pasar y desde dónde. Por ejemplo, para que la gente pudiera entrar a nuestra web desde cualquier navegador, tuvimos que abrir el puerto 80 (el de HTTP, vamos) para todo el mundo (0.0.0.0/0).

Después, pensando en el futuro y por si acaso, hicimos una copia de seguridad, lo que en AWS llaman un "Snapshot". Es como una foto completa de la máquina, con el sistema operativo y todos los datos que teníamos guardados. Esto es súper útil porque si algo sale mal, podemos volver a esta versión sin problema, o incluso usar esta "foto" para crear réplicas exactas si las necesitamos para otras cosas.

Con esa imagen ya guardada, el siguiente paso fue montar un Balanceador de Carga. Imagínenselo como un repartidor inteligente del tráfico: cuando llega una visita, él decide a cuál de nuestras instancias mandarla. Esto es fundamental para que la aplicación esté siempre disponible, incluso si una de las máquinas falla o si hay mucha gente conectándose a la vez.

Claro, para que el balanceador supiera a qué instancias mandar el tráfico, tuvimos que configurar un "Target Group". Ahí es donde agrupamos las instancias que estaban sanas y listas para recibir conexiones. El balanceador mira este grupo para saber dónde repartir el trabajo.

Además, para que todo esto se adaptara solo, configuramos un grupo de "Auto Scaling". Esto le dice a AWS que, según ciertas condiciones (como por ejemplo, si el uso de CPU sube mucho), tiene que añadir automáticamente más instancias para repartir la carga. Y al revés, si la cosa se calma, puede quitar las que sobren para no gastar de más.

Y ya para terminar, hicimos unas cuantas pruebas de carga para comprobar que todo el montaje funcionaba correctamente y aguantaba bien el tirón. ¡Queríamos estar seguros de que todo el esfuerzo había valido la pena!

A continuación, se presenta un paso a paso detallado del proceso anteriormente explicado, acompañado de las respectivas imágenes que ilustran cada etapa de la implementación y prueba.

Se crea una instancia de tipo AWS Linux usando el servicio EC2 (Elastic Compute Cloud). Esta instancia es un servidor virtual alojado en la nube de Amazon.

Configuraciones realizadas:

- Se selecciona un tipo de instancia dentro de la capa gratuita (ej. t2.micro), que incluye recursos básicos como 1 CPU virtual, 1 GB de RAM y almacenamiento de hasta 30 GB.
- Se define el sistema operativo como Amazon Linux 2.
- Se especifican detalles como nombre ("AWS-Linux") para una mejor identificación.
- Se configura un Security Group, que actúa como un firewall virtual para controlar el tráfico hacia y desde la instancia.
- Se permite el acceso público al puerto 80 (HTTP) para habilitar tráfico web desde cualquier dirección IP. Esto es esencial para que la instancia pueda servir una página web o aplicación accesible públicamente.

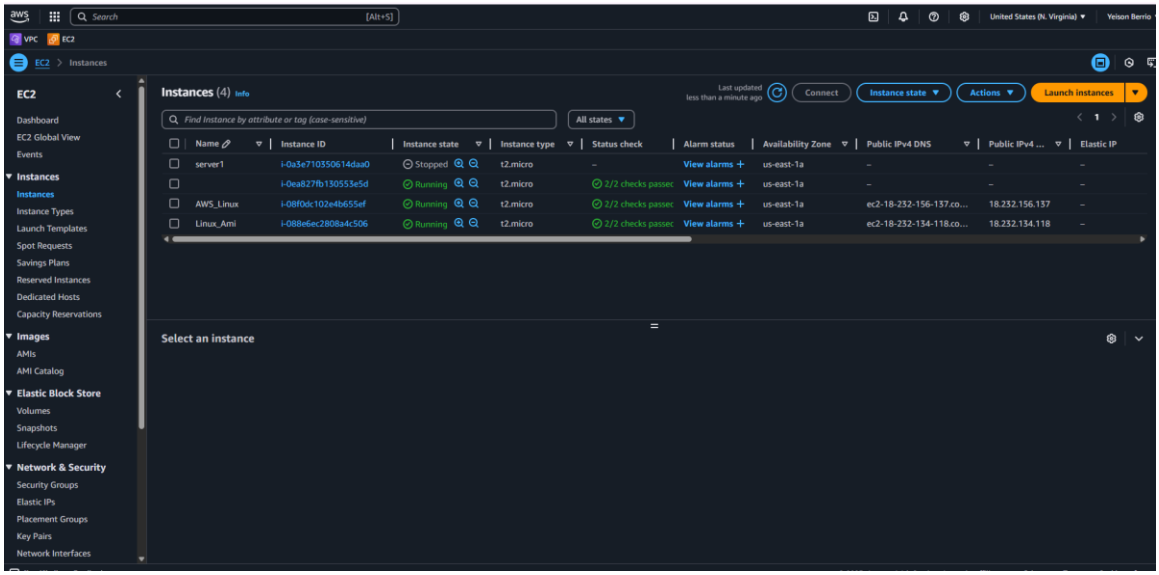

Ilustración 1

Configuración de Snapshot

Se crea un Snapshot manual llamado “Backup-Server-Linux”, el cual es una copia de seguridad de la instancia creada y su estado actual . Este backup incluye tanto el sistema operativo como los datos almacenados en el disco de la instancia.

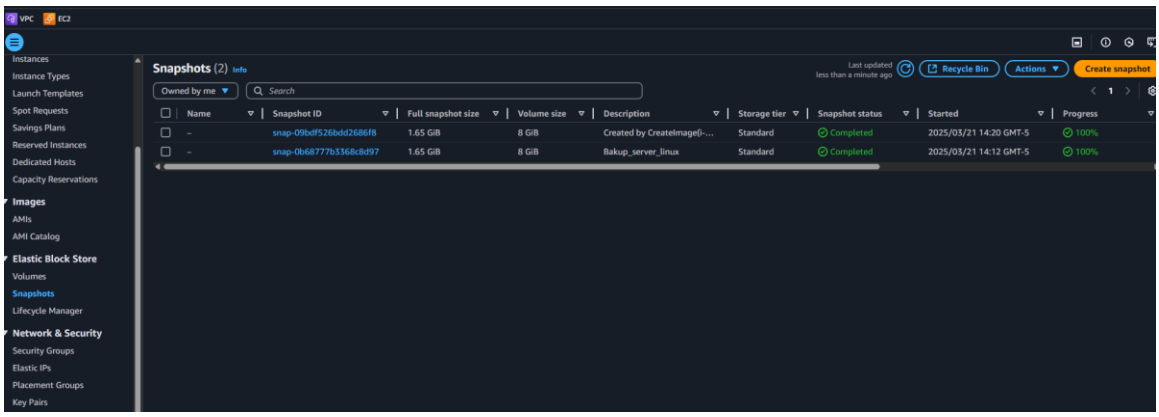

Ilustración 2

AMI (Imagen de Máquinas de Amazon)

Se crea una AMI basándose en la instancia que ya esta anteriormente establecida.
Una AMI es una plantilla que abarca el sistema operativo, programas, ajustes y datos.
Esto permite la creacion de nuevas instancias con la misma configuración de manera mas rapida.

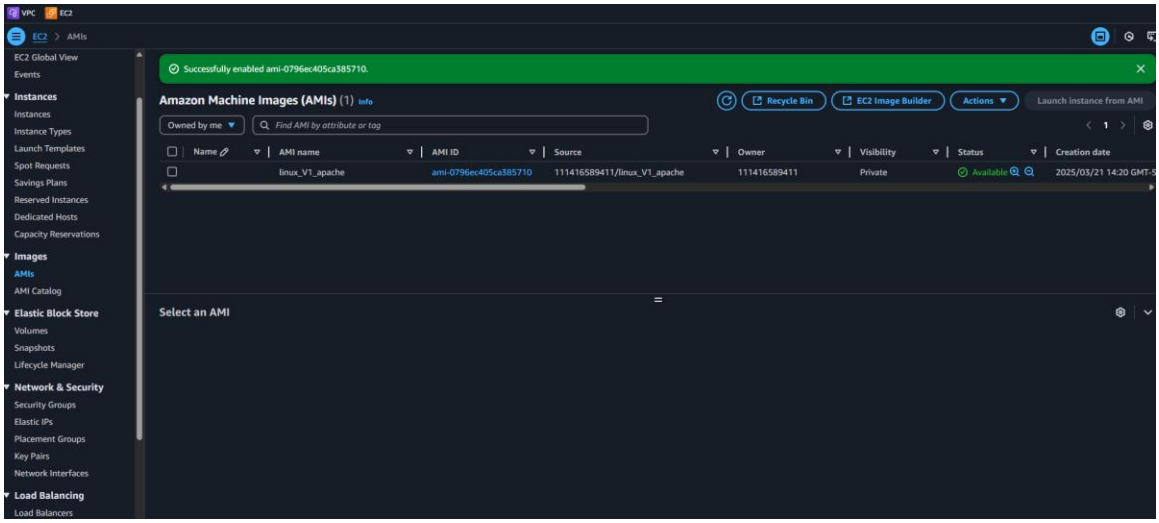

Ilustración 3

Configuración del balanceador de carga

Se Crea un balanceador de carga el cual distribuye automáticamente el tráfico de red entre múltiples instancias.Su objetivo es mantener alta disponibilidad y ser tolerable antes las fallas que se presenten este balanceador de carga Trabaja en conjunto con Auto Scaling para detectar la necesidad de aumentar o reducir la cantidad de instancias y gestionar el tráfico adecuadamente.

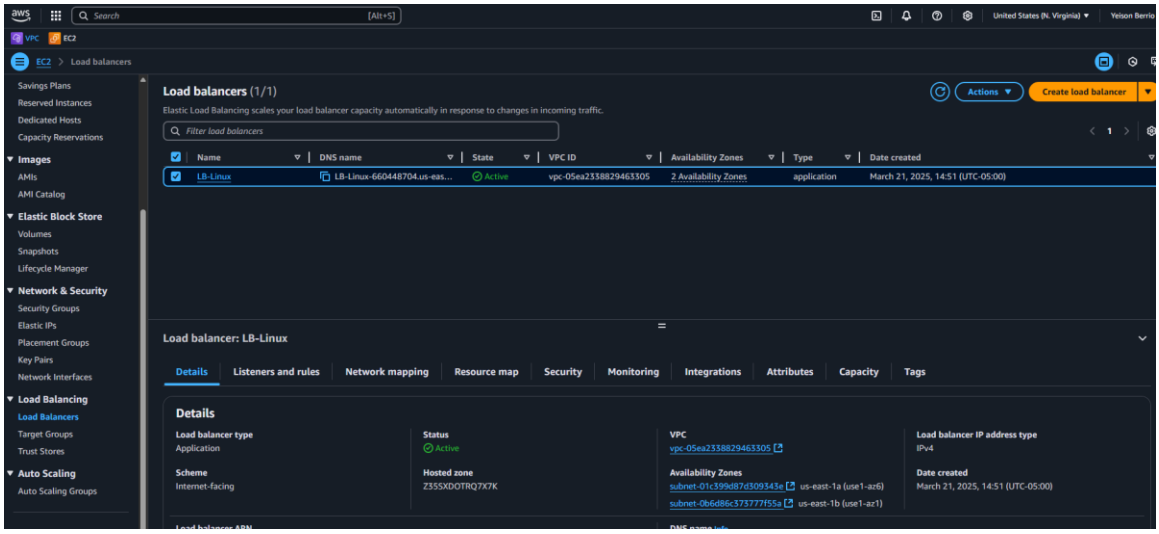

Ilustración 4

Configuración de Target Groups

La configuracion del Target Group el cual contiene las instancias que recibirán el tráfico desde el Blanceador de carga permite monitorear la salud de las instancias y controlar a cuáles se les distribuye el tráfico.El Target Group facilita la escalabilidad automatica ya que el puede añadir o quitar instancias automáticamente.

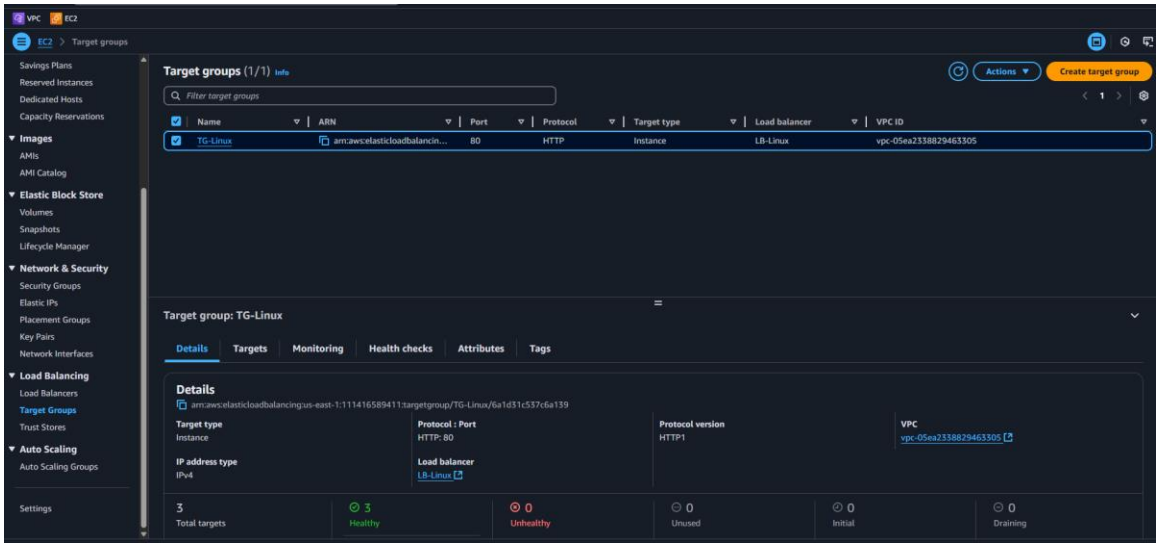

Ilustración 5

Configuración del Auto Scaling

Bueno, lo que hicimos fue montar un sistema que llamamos Auto Scaling Group. La idea principal es que, si de repente empieza a entrar mucha gente o el sistema necesita más potencia, automáticamente se creen más "servidores" (instancias) para aguantar el trafico, sin que tengamos que hacerlo manual.

Para que no se nos fuera de las manos, pusimos un límite: como mucho, puede haber 5 de estas instancias funcionando al mismo tiempo.

Decidimos que el sistema tiene que estar atento al uso de la CPU. Si vemos que alguna de las instancias está trabajando a más del 30% de su capacidad, configuramos todo para que, ¡hop!, se lance una nueva instancia automáticamente. Esta nueva instancia se crea usando una imagen que ya teníamos preparada (la AMI), así que sale lista para funcionar.

Y claro, pensamos en qué pasaría si una de estas instancias falla o se cae. Para eso, lo dejamos preparado para que el propio sistema se dé cuenta y levante una nueva enseguida para reemplazarla. Así nos aseguramos de que el servicio siga funcionando lo mejor posible, aunque alguna parte tenga un problema.

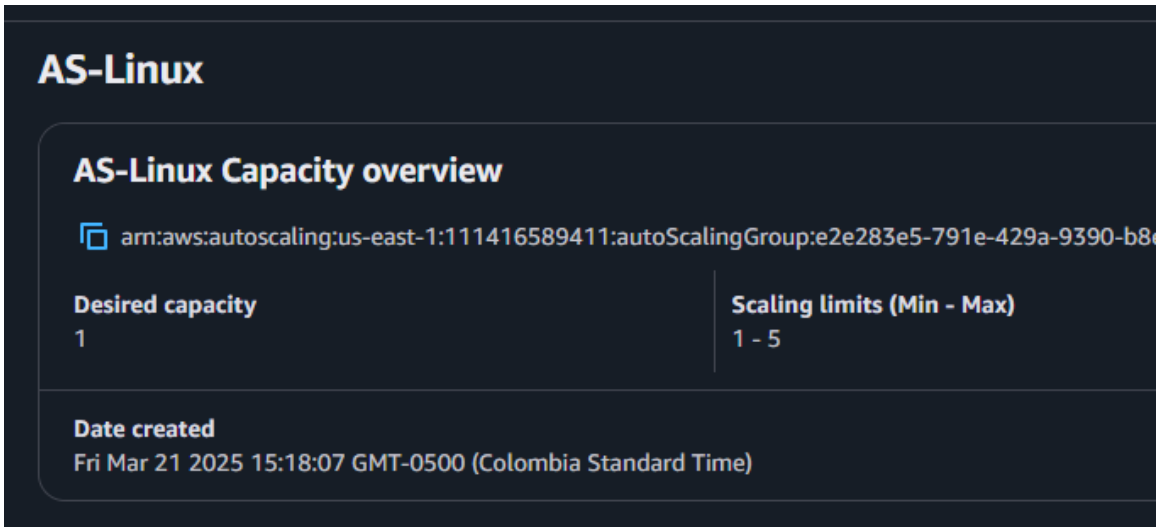

Ilustración 6

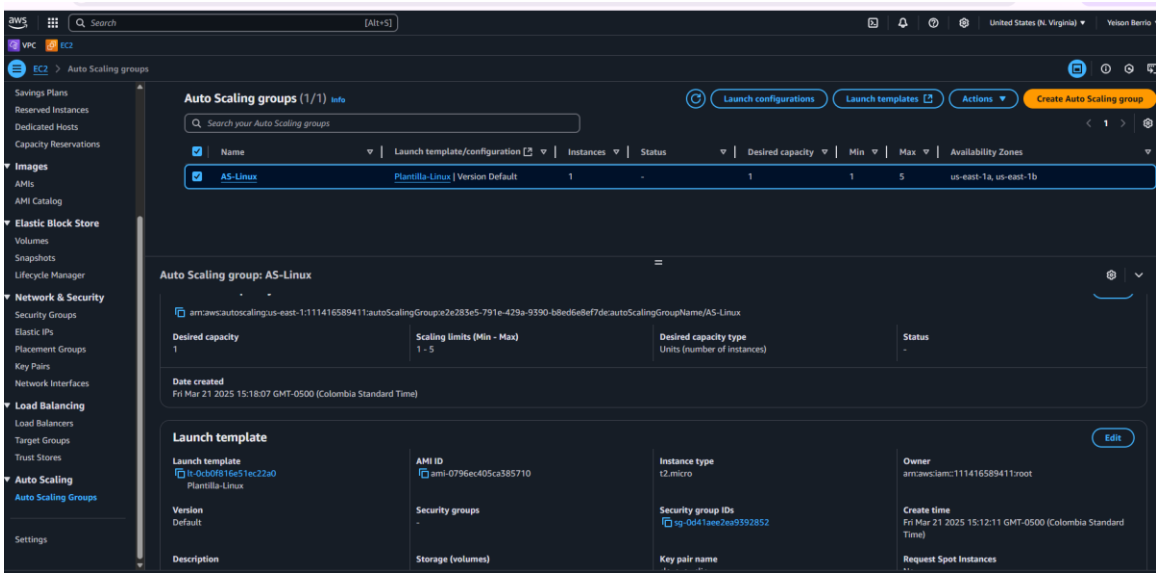

Ilustración 7

Pruebas de Funcionamiento

Se simula carga sobre la aplicación web para observar el comportamiento del sistema. Al alcanzar el uso del 30% de CPU, el Auto Scaling entra en acción creando nuevas instancias automáticamente. Y el balanceador de carga distribuye el tráfico entre las instancias disponibles, asegurando un buen rendimiento.

Generación de carga desde la consola Linux

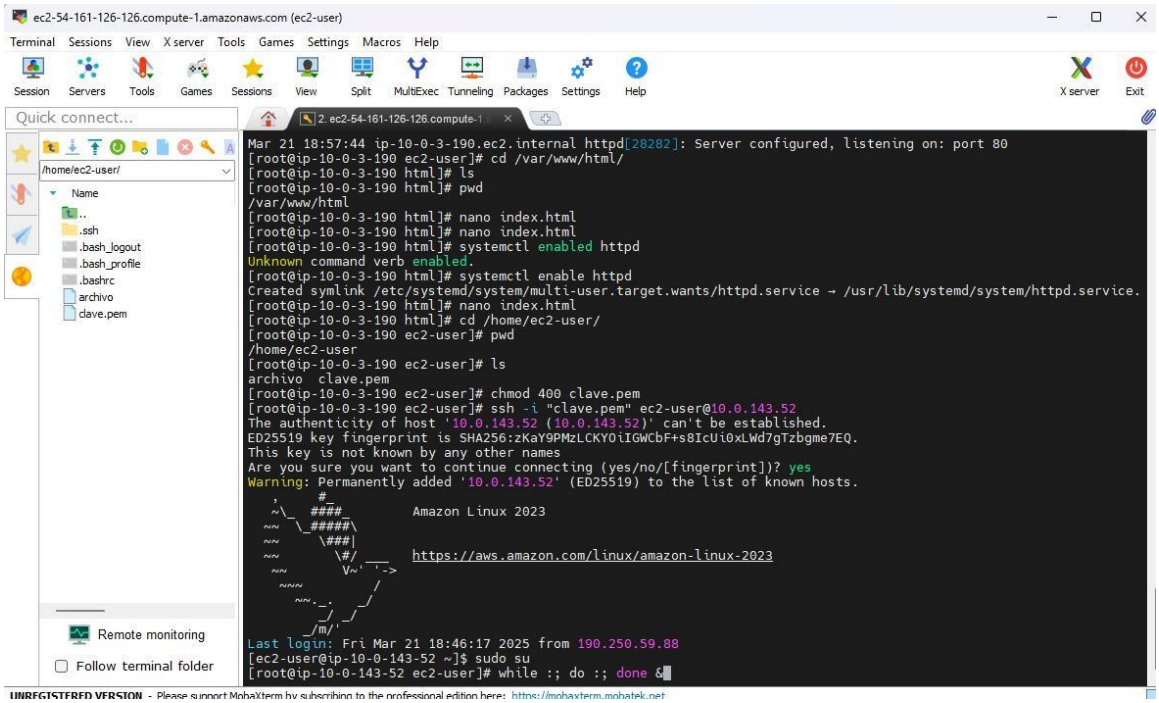

Ilustración 8

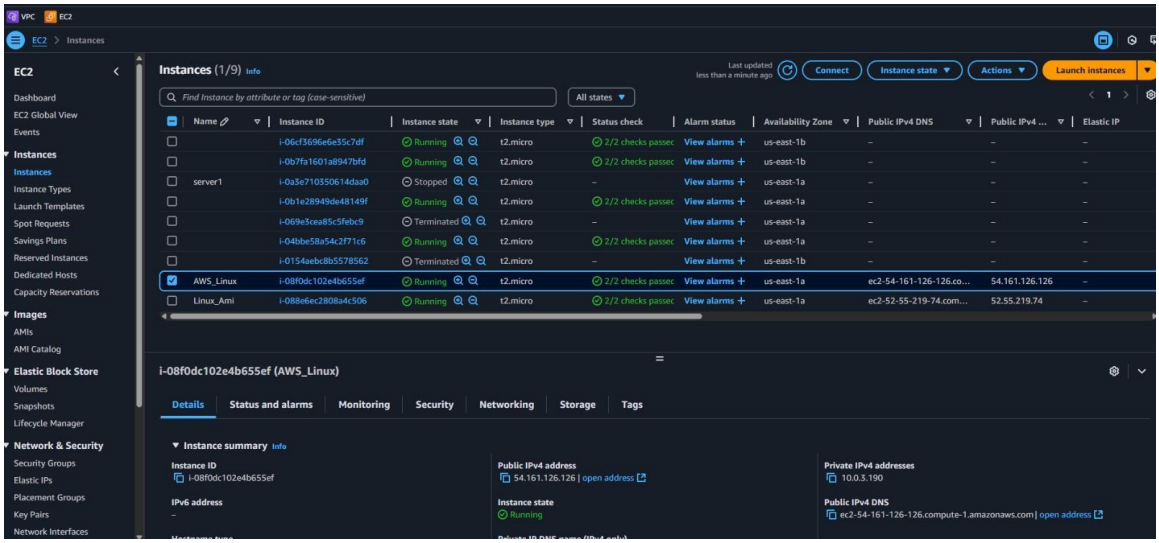

Ilustración 9

Resultado esperado

Se garantiza la continuidad del servicio sin la necesidad de intervenir en el de manaera manual optimizando el uso de recursos y costos logrando una infraestructura completamente funcional en la nube que permite escalabilidad, tolerancia a fallos y es facilmente replicable.

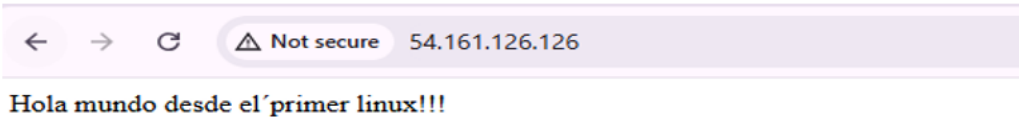

Ilustración 10

Configuración y redirección con NGINX y contenedores Docker

Bueno, para este ejercicio práctico, nos pusimos manos a la obra siguiendo unos pasos para armar una solución que nos dejara usar contenedores en una máquina Linux que teníamos en AWS. La idea principal era montar una simulación de cómo funciona el balanceo de carga y la redirección del tráfico, usando NGINX como servidor y contenedores de Docker, todo esto con unos dominios que nos inventamos.

Les contamos cómo lo hicimos, paso a paso:

Lo primero fue levantar dos contenedores Docker, los dos corriendo en la misma máquina Linux. Adentro de cada uno pusimos una página web diferente, para que se notara cuál era cuál. A uno le asignamos el puerto 8081 y al otro el 8082 para que escucharan por ahí. Con esto, simulamos tener distintos servicios, cada uno accesible por su propio dominio.

Después de eso, nos metimos a configurar el archivo `nginx.conf`, que es donde está toda la configuración principal de NGINX. Ahí definimos unos bloques llamados `server`. Estos bloques son los que se encargan de mirar qué dominio pide el usuario y redirigirlo a donde toca. Dejamos listos dos de estos bloques `server`. Ambos escuchan en el puerto 80 (el típico para web HTTP), pero cada uno responde a un nombre de dominio distinto (`servername`): uno para `www.exito.com` y el otro para `www.d1.com`. Usamos estos nombres inventados para que pareciera un escenario real de balanceo.

Dentro de cada uno de esos bloques `server`, le dijimos a NGINX a qué contenedor mandar el tráfico. Así, si alguien entraba a `www.exito.com`, NGINX agarraba esa petición y la mandaba derechito al contenedor que estaba en el puerto 8081. Y lo mismo hicimos para

`www.d1.com`: las peticiones para ese dominio iban a parar al contenedor del puerto 8082. Como dijimos, cada contenedor mostraba una página distinta, para probar que todo funcionaba bien.

Ahora, aquí venía un detalle: como los dominios `www.exito.com` y `www.d1.com` nos los inventamos, no están registrados en ningún lado ni apuntan a la IP de nuestra máquina en AWS. Para que nuestro navegador supiera a dónde ir cuando escribiéramos esas direcciones, tuvimos que hacer un pequeño truco en nuestra propia computadora (la que usábamos para probar, que tenía Windows). Editamos un archivo llamado 'hosts'. Dentro de ese archivo, añadimos unas líneas para decirle a Windows: 'oye, cuando te pidan `www.exito.com`, ve a esta dirección IP' (la IP pública de nuestra instancia en AWS), y lo mismo para `www.d1.com`. Con este cambio, ya nuestro navegador mandaba la petición directo a la máquina de AWS, y ahí NGINX ya sabía qué hacer con cada dominio y lo mandaba al contenedor correcto.

La verdad es que este ejercicio nos sirvió para ver en la práctica cómo montar una estructura básica para redirigir dominios usando Docker y NGINX, que son herramientas súper importantes hoy en día con todo esto de la virtualización y la nube.

Ahora sí, les dejamos el paso a paso detallado de todo esto que les contamos, para que quede más claro, y le pusimos unas imágenes para que se entienda mejor.

Creación de contenedores

```
[root@ip-10-0-3-190 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS                                     NAMES
6bc1961e6ad7   httpd     "httpd-foreground"   2 minutes ago   Up 2 minutes   0.0.0.0:8082->80/tcp, :::8082->80/tcp     app2
6b7b24a7f247   httpd     "httpd-foreground"   7 minutes ago   Up 7 minutes   0.0.0.0:8081->80/tcp, :::8081->80/tcp     app1
[root@ip-10-0-3-190 ec2-user]#
```

Ilustración 11

Una vez creados los contenedores, se procede a modificar el archivo de configuración principal del servidor NGINX, llamado nginx.conf. En este archivo, se establecen dos bloques server, cada uno encargado de manejar solicitudes HTTP a través del puerto 80, pero que se diferencian en este caso por el nombre del servidor. El primer bloque responde al dominio ficticio www.exito.com y redirige el tráfico al contenedor que se encuentra escuchando en el puerto 8081. Y El segundo bloque responde al dominio www.d1.com, redirigiendo el tráfico al contenedor que opera en el puerto 8082.

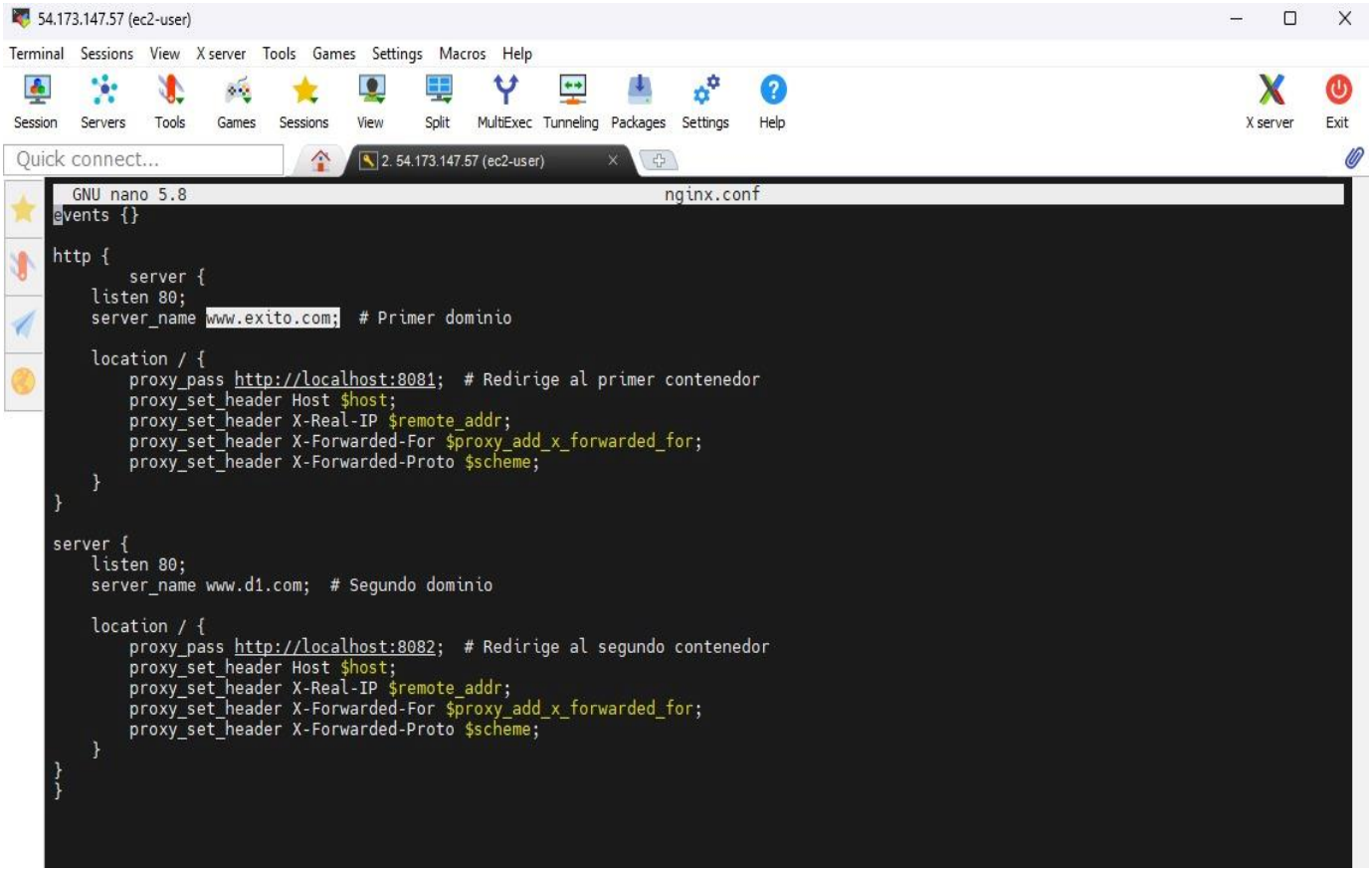

Ilustración 12

Debido a que los dominios utilizados en este ejercicio (www.exito.com y www.d1.com) no están registrados oficialmente, es necesario realizar una configuración local del archivo hosts en el sistema operativo que en este caso es Windows. Este archivo permite mapear los nombres de dominio con direcciones IP específicas. Que permiten que entradas que asocian los dominios ficticios con la dirección IP pública de la instancia de Linux en AWS.

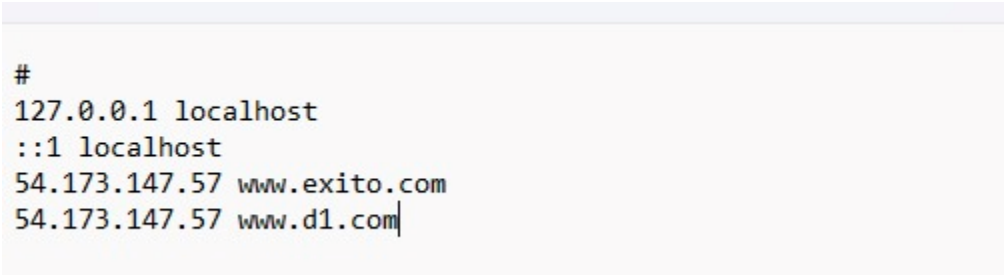

Ilustración 13

Visualización de los resultados y validación

Después de seguir todos los pasos anteriores nombrados, se puede acceder desde un navegador a los dominios configurados. Si la configuración es correcta, cada dominio mostrará una página diferente, correspondiente al contenedor Docker que al que fue redirigido a través de NGINX. Esta validación confirma que la arquitectura de contenedores y redirección está funcionando adecuadamente.

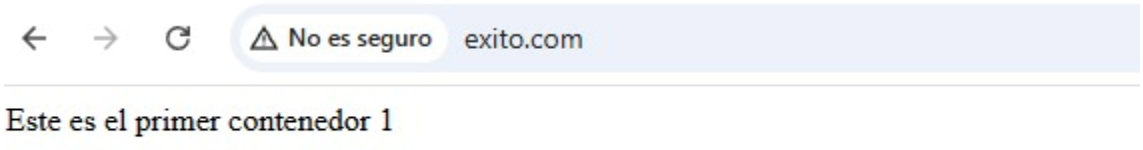

Ilustración 14

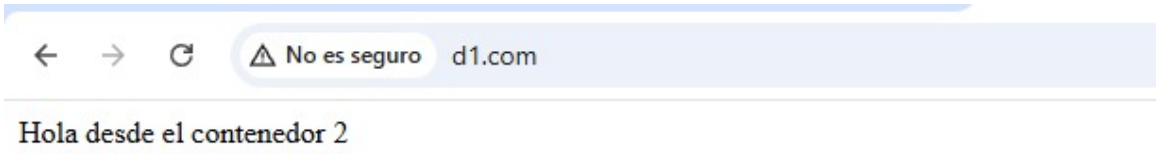

Ilustración 15

Luego de realizar este ejercicio y lograr exitosamente el cumplimiento del objetivo que se trazo en la actividad, A continuación se presenta una serie de estadísticas de uso de servicios y otras estadísticas de mercado que presenta aws.

Servicios más utilizados en 2025

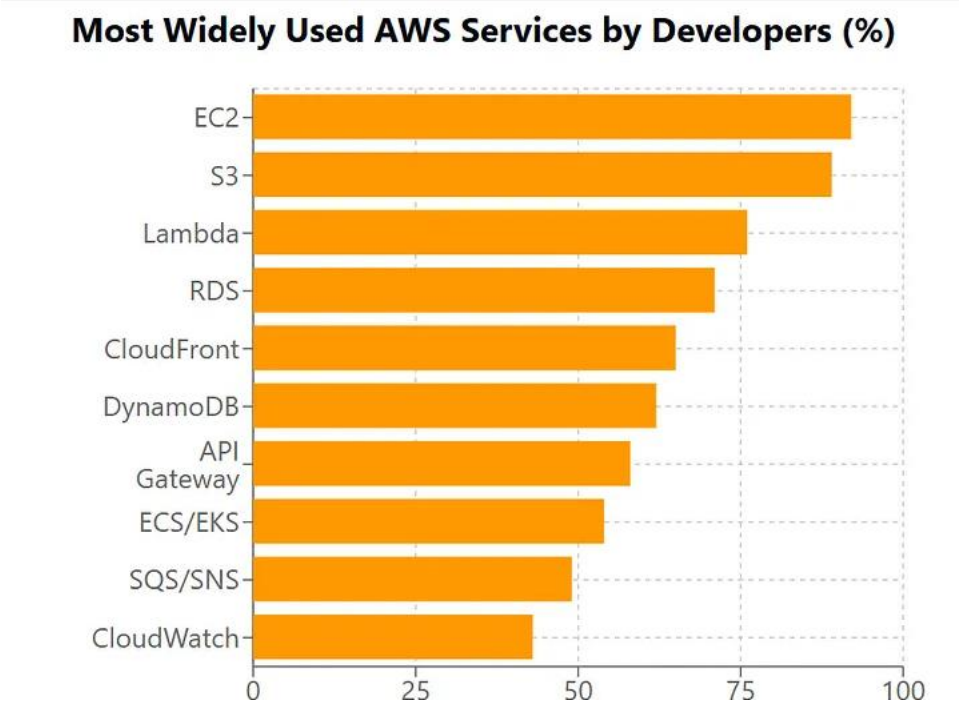

Ilustración 16

Estadísticas de uso de AWS

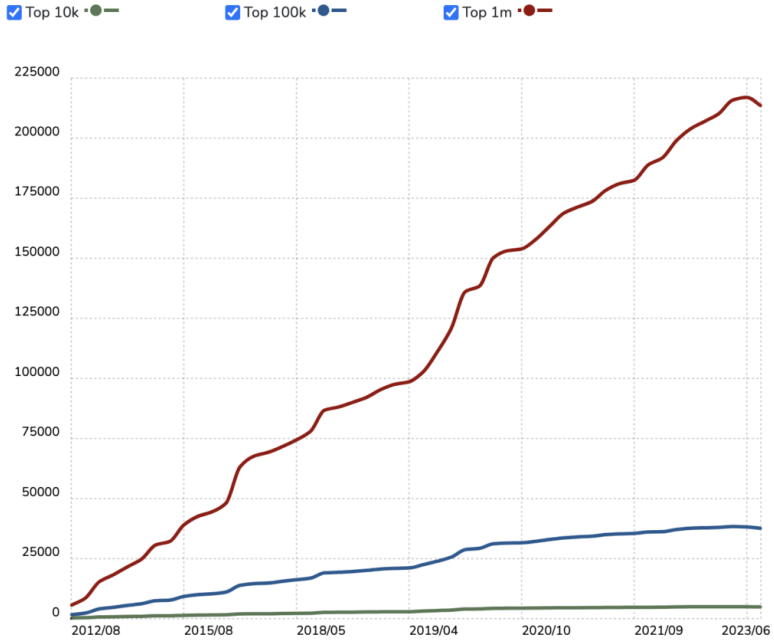

Ilustración 17

Cuota de mercado de alojamiento WEB

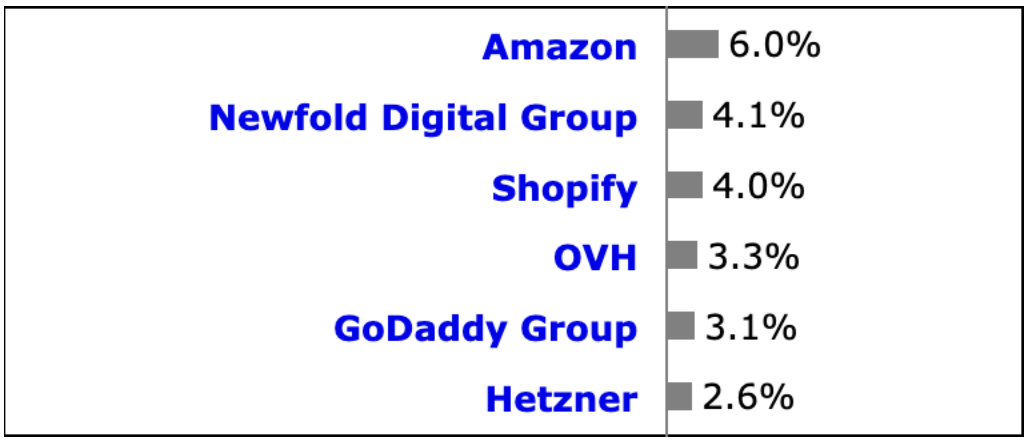

Ilustración 18

Estadísticas de sitios web que mas utilizan proveedores de alojamiento en la nube

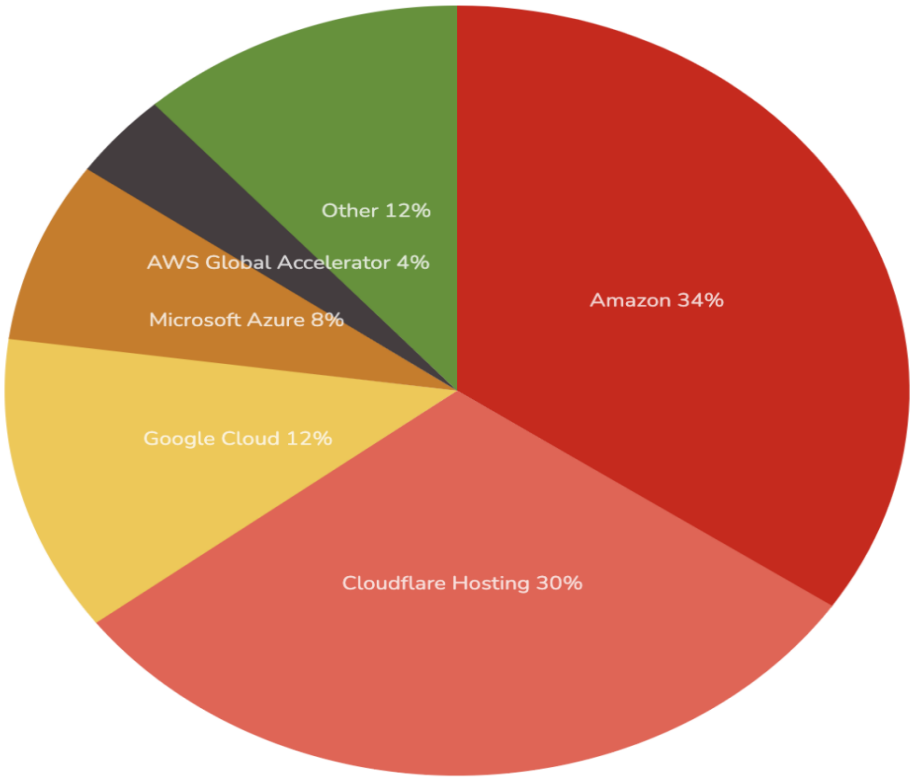

Ilustración 19

Tabla 1. Servicios más usados de AWS.

Servicio	***Funcionalidad***
Amazon S3	Almacenamiento
Amazon EC2	Permite creación y lanzar de instancias
Amazon RDS	Bases de datos
Amazon CloudFront	Redes y entrega de contenido
Amazon Lambda	Computo sin servidores
Amazon DynamoDB	Bases de datos NoSQL
Amazon SQS/SNS	Servicio de cola simple/Servicio de notificación simple
Amazon CloudWatch	Monitoreo y Optimización de recursos

Figuras y tablas

Ilustración 1 ____ *9*
Ilustración 2 ____ *9*
Ilustración 3 ____ *10*
Ilustración 4 ____ *11*
Ilustración 5 ____ *11*
Ilustración 6 ____ *12*
Ilustración 7 ____ *13*
Ilustración 8 ____ *14*
Ilustración 9 ____ *14*
Ilustración 10 ____ *15*
Ilustración 11 ____ *16*
Ilustración 12 ____ *17*
Ilustración 13 ____ *17*
Ilustración 14 ____ *18*
Ilustración 15 ____ *18*
Ilustración 16 ____ *19*
Ilustración 17 ____ *19*
Ilustración 18 ____ *20*
Ilustración 19 ____ *20*

.

Conclusiones

La computación en la nube juega un papel fundamental del mundo de la tecnología actual ya que es muy usada por empresas y personas y se vuelve una herramienta de gran ayuda gracias a su capacidad de ofrecer soluciones eficientes e innovadoras, la importancia de adquirir estos nuevos conocimientos nos permite tener una visión más amplia del pasado el presente y el futuro de lo que ha representado y representan la computación en la nube y la virtualización para las empresas y personas, a lo largo del seminario se fueron adquiriendo una gran cantidad de herramientas muy útiles de lo que se requiere en el ámbito empresarial para brindar soluciones escalables e innovadoras que permiten generar desarrollos para potenciar el entorno en el cual nos encontramos.

En las prácticas y los desarrollos implementados en el seminario en el cual se logra aprender una gran cantidad de conocimientos útiles con los cuales podemos brindar soluciones a problemáticas de empresas y personas que cuentan con recursos en la nube, si bien se tenía un concepto de lo que es la computación en la nube, al realizar la práctica ese concepto se amplió con mayor precisión y criterio del alcance que esta puede tener y la viabilidad de su implementación en determinados entornos. Para nosotros lo aprendido durante este periodo de tiempo nos amplia enormemente muchos conceptos técnicos y también la visión sobre el mundo de la computación en la nube ya que son conocimientos y habilidades con las cuales podemos potenciar muchos conocimientos y proyectos**.**

Referencias

Amazon Web Services. (s.f.). *¿Qué es AWS?* Recuperado el 5 de abril de 2025, de https://aws.amazon.com/es/what-is-aws/

Kinsta. (n.d.). *Amazon Usage Statistics* [Imagen]. Recuperado el 5 de abril de 2025, de https://kinsta.com/wp-content/uploads/2023/08/amazon-usage-statistics.png

Kinsta. (n.d.). *Cloud Hosting Distribution - Top 100k Sites* [Imagen]. Recuperado el 5 de abril de 2025, de https://kinsta.com/wp-content/uploads/2023/08/cloud-hosting-distribution-top-100k-sites.png

Kinsta. (n.d.). *Hosting Provider Usage* [Imagen]. Recuperado el 5 de abril de 2025, de https://kinsta.com/wp-content/uploads/2023/08/hosting-provider-usage-1.png

Medium. (n.d.). *[Imagen alojada en Medium]*. Recuperado el 5 de abril de 2025, de https://miro.medium.com/v2/resize:fit:720/format:webp/1*V_SEE_H8efK08o0rRW0Yvg.png