TRABAJO DE GRADO
Opción Seminario-Diplomado.

Soluciones escalables en AWS

Corporación Universitaria Remington.
Nombre de la facultad: Ingenieria
Nombre del programa académico: Ingenieria de sistemas

Wilmar Yosep Lizcano Cruz
Nombre del Tutor del trabajo de grado: Juan Pablo Berrio Lopez
Opción de Trabajo de grado Seminario-Diplomado.
2025

Tabla de Contenidos

Resumen

Bueno, pues el seminario de AWS que hice estuvo bien interesante, la verdad. Nos metimos de lleno en cuatro temas principales: EC2, VPC, S3 y Elastic Container Service, que son como las bases para armar cosas en la nube. Empezamos con EC2, que es básicamente donde montas tus máquinas virtuales. Nos explicaron cómo lanzar instancias, elegir tamaños según lo que necesites (como t3.micro para pruebas o algo más potente para producción), y cómo configurarlas para que no se caigan si hay picos de uso. Me gustó que nos enseñaron a usar el panel para subir una máquina en minutos, aunque a veces me perdía con tantas opciones.

Luego pasamos a VPC, que es como armar tu propia red privada en AWS. Esto fue un poco más complicado, pero entendí que sirve para controlar quién entra y quién no a tus recursos. Hablamos de subredes, tablas de enrutamiento y gateways, y hasta hicimos un ejercicio para conectar una VPC con internet. Me quedó claro que sin esto, la seguridad se vería comprometida.

Después vino S3, que es el almacenamiento en la nube. Este me pareció de lo más fácil. Nos mostraron cómo subir archivos, crear buckets y ponerles permisos para que no cualquiera los vea. Lo que me pareció mejor es que puedes guardar desde fotos hasta backups gigantes, y con las políticas de ciclo de vida hasta ahorras plata moviendo cosas viejas a Glacier.

Por último, tocamos Elastic Container Service (ECS), que es para manejar contenedores tipo Docker. Aquí sí me costó un poco más, pero básicamente es para correr aplicaciones en contenedores que se escalan solos.

Después vino Auto Scaling, que es para que tus máquinas se multipliquen o reduzcan solas según la demanda.

Y por último, los Balanceadores de carga (Elastic Load Balancers). Estos reparten el tráfico entre tus instancias para que ninguna se sature. Hicimos uno tipo Application Load Balancer para una app sencilla, y nos explicaron cómo redirige usuarios sin que se den cuenta. Es como un policía de tráfico para la nube.

Palabras clave

Nube, escalabilidad, redes, almacenamiento y contenedores.

Marco contextual y conceptual

Marco contextual

Mi trabajo de grado abarca Amazon Web Services, o sea, AWS, que es como el rey de la nube hoy en día. El punto es que las empresas ya no quieren gastar demasiado dinero en servidores físicos que se quedan viejos rápido, entonces usan la nube para todo: desde guardar archivos hasta correr apps. AWS es una plataforma que te da un mundo de herramientas para armar lo que necesites sin complicarte la vida. En este caso, me enfoqué en cómo usar EC2, VPC, S3, ECS, Auto Scaling y los Balanceadores de carga para hacer algo funcional. Esto importa porque vivimos en un mundo donde todo tiene que ser rápido, seguro y que aguante cuando mucha gente lo usa a la vez, como en apps o tiendas online. Mi idea era entender cómo estas piezas encajan para que un negocio o proyecto no se caiga y pueda crecer.

Marco conceptual

Ahora, para que se entienda de qué se trata mi trabajo, hay que entender unas ideas básicas. Primero, EC2 es como tener computadoras virtuales que puedes prender o apagar cuando quieras, para correr programas o lo que sea. Luego está VPC, que es como armar tu propia red privada en la nube, para que nadie pueda comprometer nuestra seguridad y todo esté bien organizado. S3 es el lugar donde guardas cosas, como un disco duro gigante para fotos, videos o backups, pero en internet. Después, ECS es para manejar contenedores, que son como cajas donde metes tus aplicaciones para que corran solas. Auto Scaling es la magia de que tus máquinas se multipliquen si hay mucha gente usándolas, o se reduzcan si no, para no gastar de más. Y los Balanceadores de carga son como un repartidor que manda el tráfico a donde hay espacio, para que nada se sature. Todo esto junto es como un equipo que trabaja para que una app o sistema en la nube sea rápido, seguro y funcional cuando haya mucha demanda.

Historia de la Virtualización y la Computación en la Nube
Orígenes de la Virtualización

El concepto de virtualización se remonta a los años 60, cuando las computadoras de gran tamaño eran costosas y se buscaba aprovechar al máximo su capacidad. IBM fue pionero en esta área con el desarrollo de técnicas que permitían dividir un solo equipo en varias máquinas virtuales, facilitando el acceso a diferentes usuarios. Con el tiempo, esta tecnología evolucionó y, en la década de los 90, comenzó a aplicarse en servidores comerciales gracias a empresas como VMware, que hicieron posible la creación de entornos virtualizados en hardware convencional.

Nacimiento de la Computación en la Nube

El avance de la virtualización, combinado con la mejora de las redes y el acceso a internet de alta velocidad, sentó las bases para lo que hoy conocemos como computación en la nube. A principios de los 2000, compañías como Amazon comenzaron a ofrecer servicios en línea que permitían a las empresas almacenar datos y ejecutar aplicaciones sin necesidad de invertir en infraestructura propia. Con el tiempo, otros gigantes tecnológicos como Google y Microsoft se sumaron a esta tendencia, impulsando el desarrollo de soluciones en la nube que hoy en día forman parte esencial del entorno digital.

Comparación entre AWS, Azure y GCP

En el mundo de la computación en la nube, Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform (GCP) son las tres principales opciones. Cada una de estas plataformas ofrece una amplia gama de servicios, pero presentan diferencias clave en su enfoque, facilidad de uso y mercado objetivo.

1. **AWS**: El líder consolidado

AWS es el servicio de computación en la nube más maduro y ampliamente adoptado. Se caracteriza por su gran cantidad de servicios y su presencia global. Su infraestructura es robusta y escalable, lo que lo convierte en la opción preferida para grandes empresas y startups. Sin embargo, su modelo de precios puede ser complejo, y la curva de aprendizaje inicial es pronunciada debido a la gran cantidad de herramientas disponibles.
Ventajas:
✓ Mayor cantidad de servicios y presencia global.
✓ Amplia comunidad y documentación.
✓ Versátil y utilizado en múltiples industrias.
Desventajas:
✕ Estructura de precios complicada.
✕ Puede ser difícil de aprender para principiantes.

2. **Azure:** La mejor integración con Microsoft

Azure es la plataforma de nube de Microsoft y está diseñada para integrarse perfectamente con herramientas como Windows Server, Active Directory y Office 365. Es una opción popular entre empresas que ya utilizan productos de Microsoft, ya que permite una transición más sencilla a la nube. Aunque su catálogo de servicios es amplio, algunos consideran que su interfaz no es tan intuitiva como la de sus competidores.
Ventajas:
✓ Excelente integración con el ecosistema de Microsoft.
✓ Buenas opciones para entornos empresariales híbridos.
✓ Seguridad y cumplimiento normativo avanzados.
Desventajas:
✕ No es tan intuitivo para nuevos usuarios.
✕ Menos flexibilidad fuera del ecosistema de Microsoft.

3. **GCP:** La mejor opción para análisis de datos e inteligencia artificial

Google Cloud Platform se ha posicionado como una plataforma especializada en inteligencia artificial, aprendizaje automático y big data. Su infraestructura se basa en la misma tecnología que utiliza Google para sus propios servicios, lo que garantiza rendimiento y confiabilidad. Además, suele ofrecer precios competitivos y créditos gratuitos para estudiantes y desarrolladores. No obstante, su adopción empresarial es menor en comparación con AWS y Azure.
Ventajas:
✓ Potente en inteligencia artificial y análisis de datos.
✓ Infraestructura optimizada y eficiente.
✓ Modelo de precios competitivo y créditos gratuitos para estudiantes.
Desventajas:
✕ Menor adopción en empresas tradicionales.
✕ Menos servicios disponibles en comparación con AWS.

LINEA DE TIEMPO

Fundacion de la virtualizacion

- 1960 : IBM desarrolla el concepto de tiempo compartido para optimizar el uso de mainframes.
- 1967 : IBM presenta CP-40 y CP-67 , los primeros sistemas operativos con hipervisores.

Decada de 1960

Expansion de la virtualizacion

- 1972 : IBM lanza el sistema operativo VM/370 , consolidando la virtualización en entornos empresariales.
- 1974 : Gerald Popek y Robert Goldberg publican la teoría de la virtualización, definiendo los tipos de hipervisores.

Decada de 1970

Avances en Virtualización x86

1999 : Se funda VMware , introduciendo la virtualización en servidores comerciales.

Decada de 1990

Nacimiento de la Computación en la Nube

- 2003 : Amazon empieza a desarrollar la infraestructura de AWS.
- 2006 : Se lanza Amazon Web Services (AWS) con servicios como EC2 y S3.
- 2008 : Google presenta Google App Engine , uno de los primeros entornos PaaS.
- 2010 : Microsoft entra al mercado con Azure .

Decada de 2000

Expansion de la nube

- 2011 : IBM lanza su plataforma en la nube.
- 2012 : Docker revoluciona la virtualización con contenedores .
- 2014 : Google lanza Kubernetes , permitiendo la gestión avanzada de contenedores.
- 2019 : Se consolida el modelo multinube con servicios híbridos como AWS Outposts y Azure Arc .

Decada de 2010

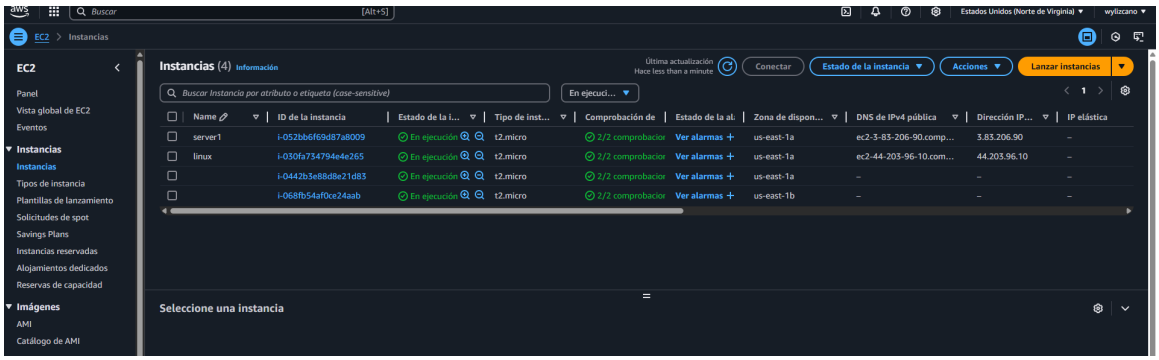
Nube inteligente y distribuida

- 2021 : Crece la adopción de Edge Computing para reducir la latencia en la nube.
- 2023 : Avances en IA y automatización impulsan la nube serverless y modelos como ChatGPT .
- 2024 y en adelante : Se expanden conceptos como Quantum Computing en la nube , IA avanzada y computación distribuida.

2023

EC2:

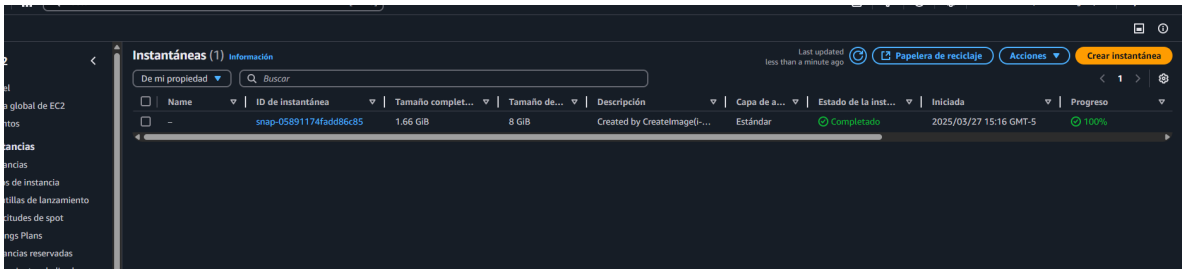
Esta captura es del panel de AWS, específicamente de la sección de EC2, donde se ven las instancias que armamos. Hay una tabla con varias filas, cada una es una máquina virtual que creamos. Se ve el nombre de cada instancia (server1), el tipo (t2.micro), el estado (en ejecución o detenida), y cosas como la IP pública y la zona de disponibilidad.

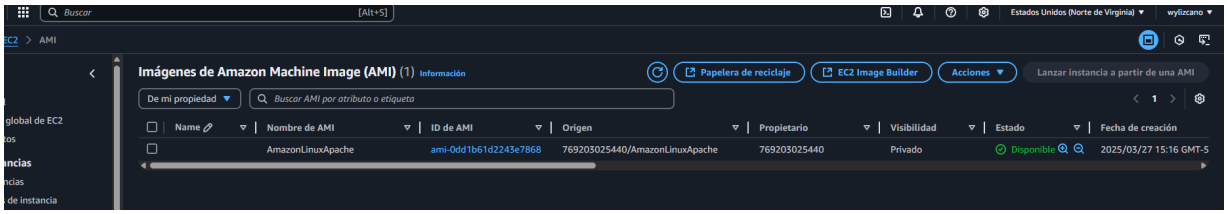


Aquí lo que hicimos fue lanzar unas instancias en EC2, que son como computadoras virtuales que puedes usar para lo que quieras. Entramos al panel de AWS, le dimos en lanzar instancia, y escogimos el tipo de máquina según lo que necesitábamos: unas básicas para pruebas y otras con más capacidad para simular un servidor real. Les pusimos nombres para no perdernos. También elegimos el sistema operativo y configuramos cosas como la red y el almacenamiento, que fue como 30 GB para empezar. Después las prendimos y en la captura se ve que están corriendo.

Instantaneas y AMI:

Esta es del apartado de Imágenes o AMI en EC2. Se ve una lista con una AMI que creamos, con un nombre como AmazonLinuxApache, podemos ver detalles como la fecha, también dice que es privada, o sea, solo para nosotros."

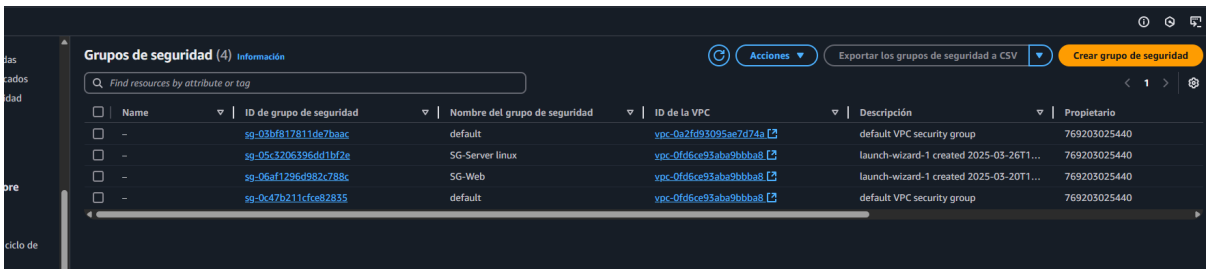


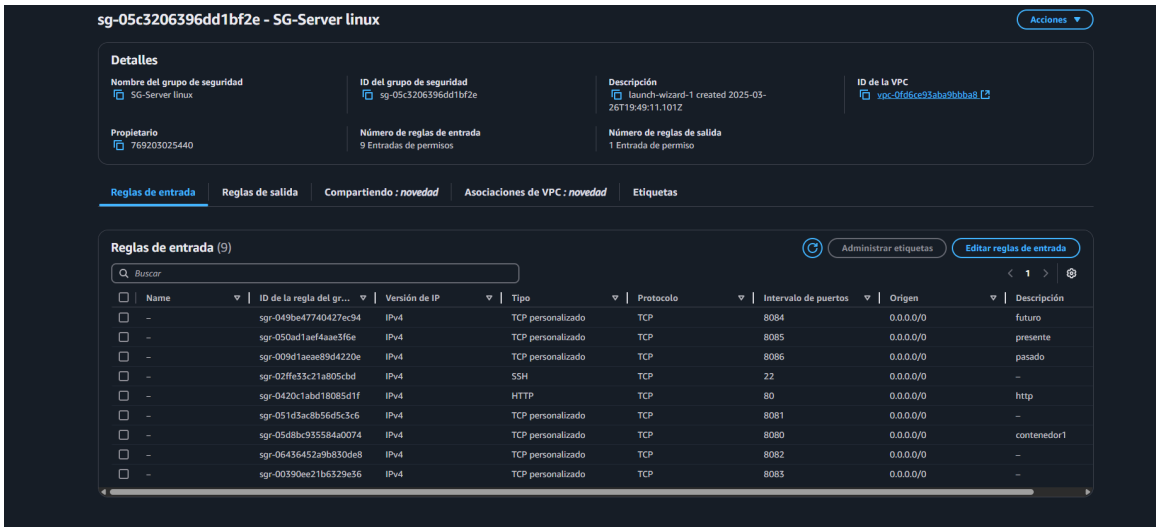


Lo que hicimos aquí fue crear una AMI, que es como un molde del servidor para usarlo después o hacer más iguales. Agarramos la instancia que ya teníamos, le dimos en 'Acciones' y luego en 'Crear imagen'. Le pusimos un nombre fácil de reconocer y dejamos que AWS sacara una copia de todo: el sistema operativo, los ajustes y lo que le habíamos instalado. Luego apareció en la lista. Esto es bueno porque con la AMI puedes lanzar otra máquina idéntica sin empezar de cero, como si hicieras un clon para probar algo o para crecer rápido si necesitas más servidores.

GRUPOS DE SEGURIDAD EN EC2:

Esta captura es de la sección de Grupos de seguridad en el panel de EC2. Se ve una lista con varios grupo que creamos, por ejemplo un nombre como SG-server linux. Al lado hay una pestaña de reglas, donde se muestra qué tráfico dejamos entrar: dice HTTP en el puerto 80 desde cualquier IP (0.0.0.0/0), SSH en el puerto 22 desde mi IP, y algo para salida que deja pasar todo. Todo está en una tabla con columnas para tipo, protocolo y rango de puertos.

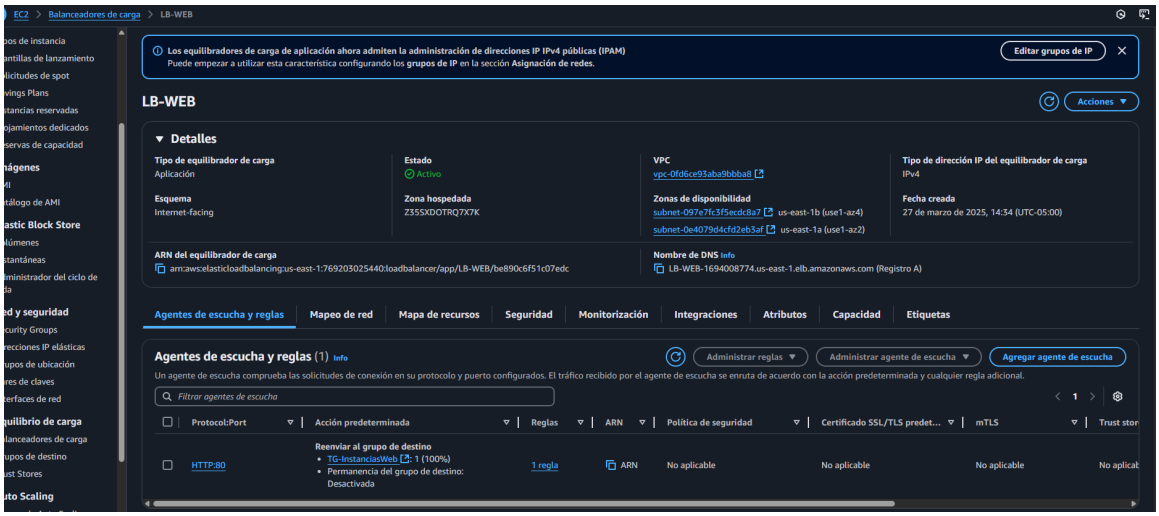


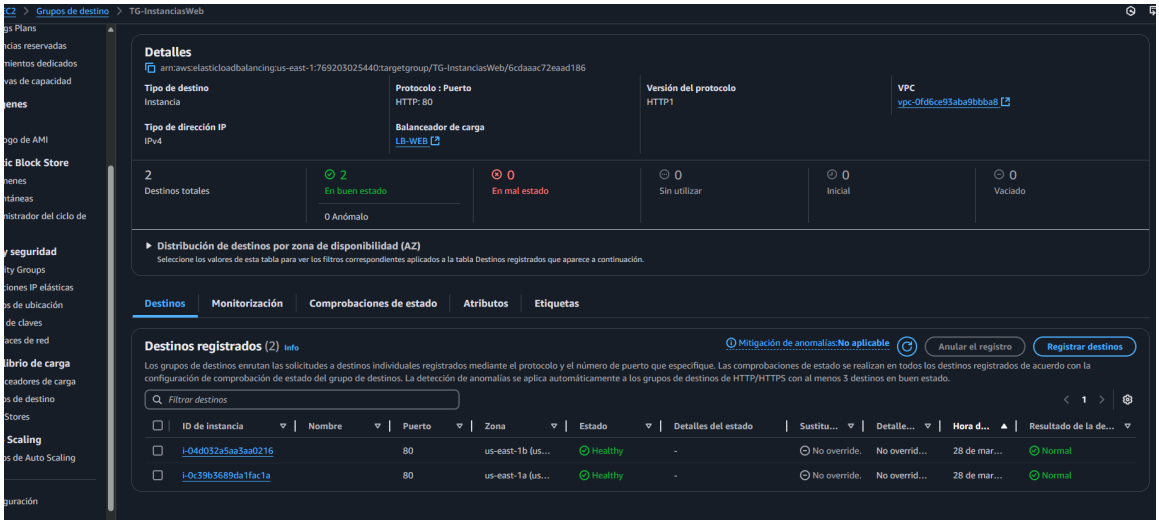


Acá lo que hicimos fue armar un grupo de seguridad para controlar quién puede entrar y salir de nuestras instancias en EC2. Entramos al panel, creamos un grupo nuevo y le pusimos un nombre fácil. Luego le metimos unas reglas: abrimos el puerto 80 para que cualquiera pueda entrar por HTTP, porque queríamos que la máquina sirviera una página web. También abrimos el puerto 22 para SSH, pero solo desde mi IP para conectarme seguro al servidor. Para las salidas, dejamos todo abierto para que la instancia pueda mandar datos sin problema. Esto es como ponerle un guardia a la puerta de tu casa, pero en la nube, para que solo entren los que tú quieres y el resto se queden fuera.

BALANCEADORES DE CARGA:

Esta captura es de la sección de Balanceadores de carga en el panel de EC2. Se ve una tabla con un balanceador que creamos, con un nombre como LB-WEB. Al lado hay detalles como el estado Activo, la zona de disponibilidad y un DNS largo para conectarse. También hay una pestaña de Objetivos que muestra las instancias conectadas con su estado en verde.

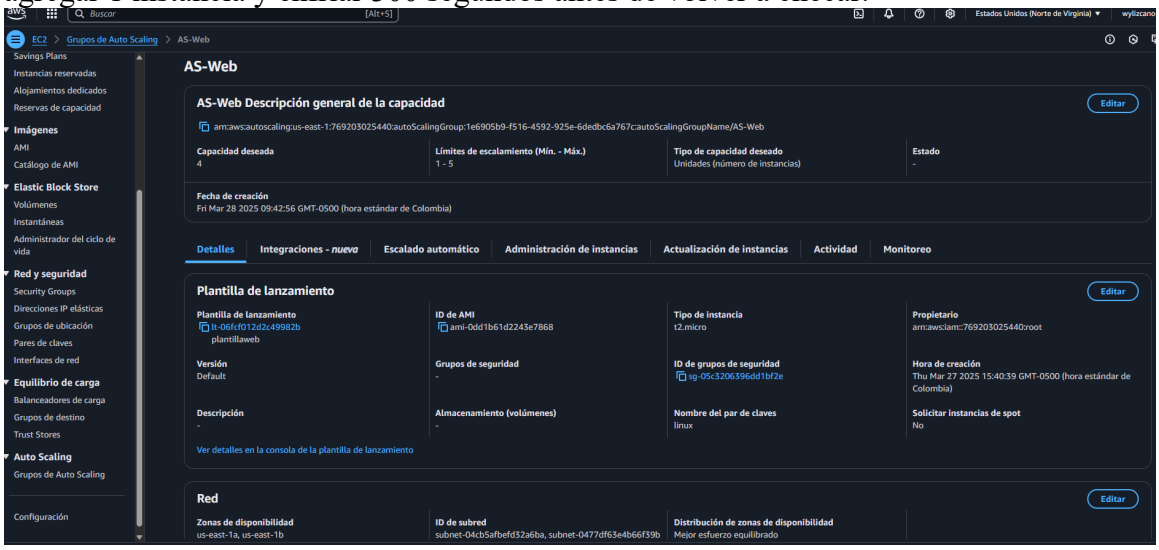


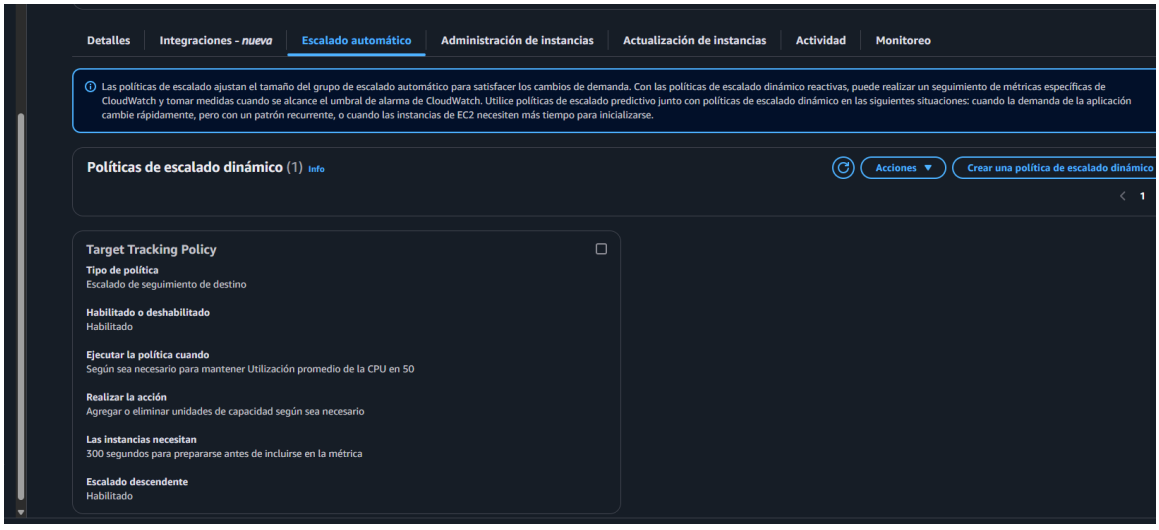


Lo que hicimos aquí fue montar un balanceador de carga para repartir el tráfico entre nuestras instancias de EC2. Entramos al panel, le dimos en ‘Crear balanceador de carga’ y escogimos el tipo Application Load Balancer, porque era para una página web. Le pusimos un nombre como LB-WEB y lo configuramos para que trabajara en dos zonas, para que no se cayera si una fallaba. Luego le dijimos que escuchara en el puerto 80 para HTTP y que mandara el tráfico a un grupo de objetivos donde estaban las instancias. Hicimos una regla básica para que repartiera parejo entre las dos, y después de un rato, el DNS que nos dio ya estaba funcionando.

AUTOSCALING:

Esta captura es de la sección de 'Grupos de Auto Scaling' en el panel de EC2. Se ve un grupo que creamos, con un nombre como AS-Web, y detalles como cuántas instancias tiene corriendo, el mínimo, el máximo, y las zonas. Hay una pestaña de escalado automatico que muestra una regla que pusimos, algo como Aumentar si CPU > 50%, con pasos para agregar 1 instancia y enfriar 300 segundos antes de volver a checar.

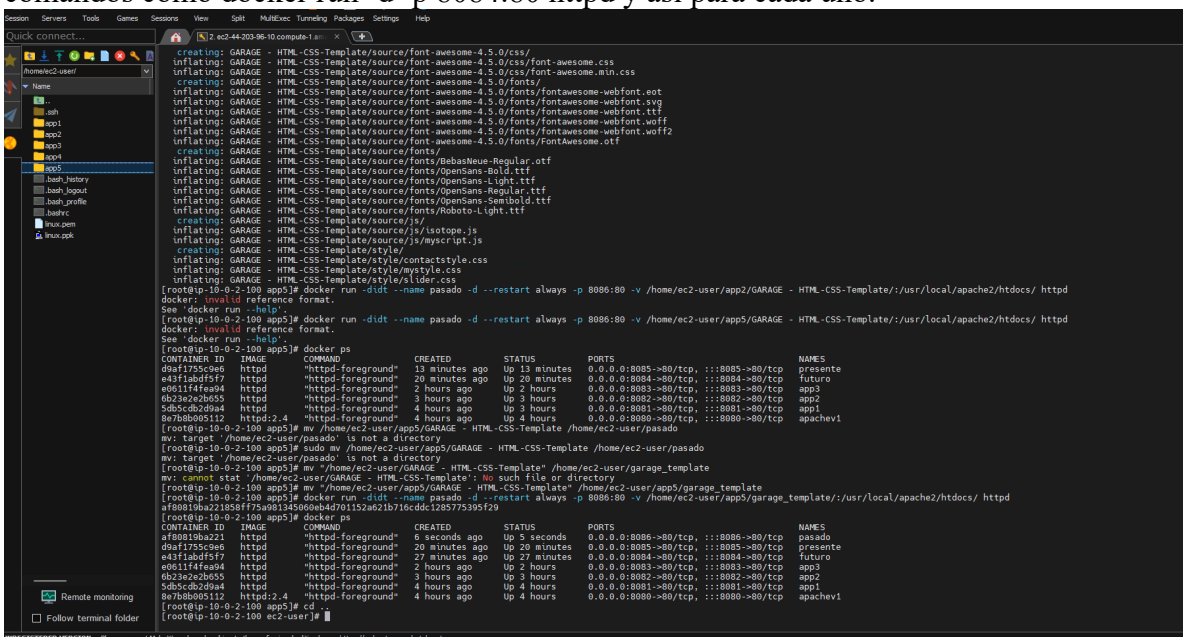




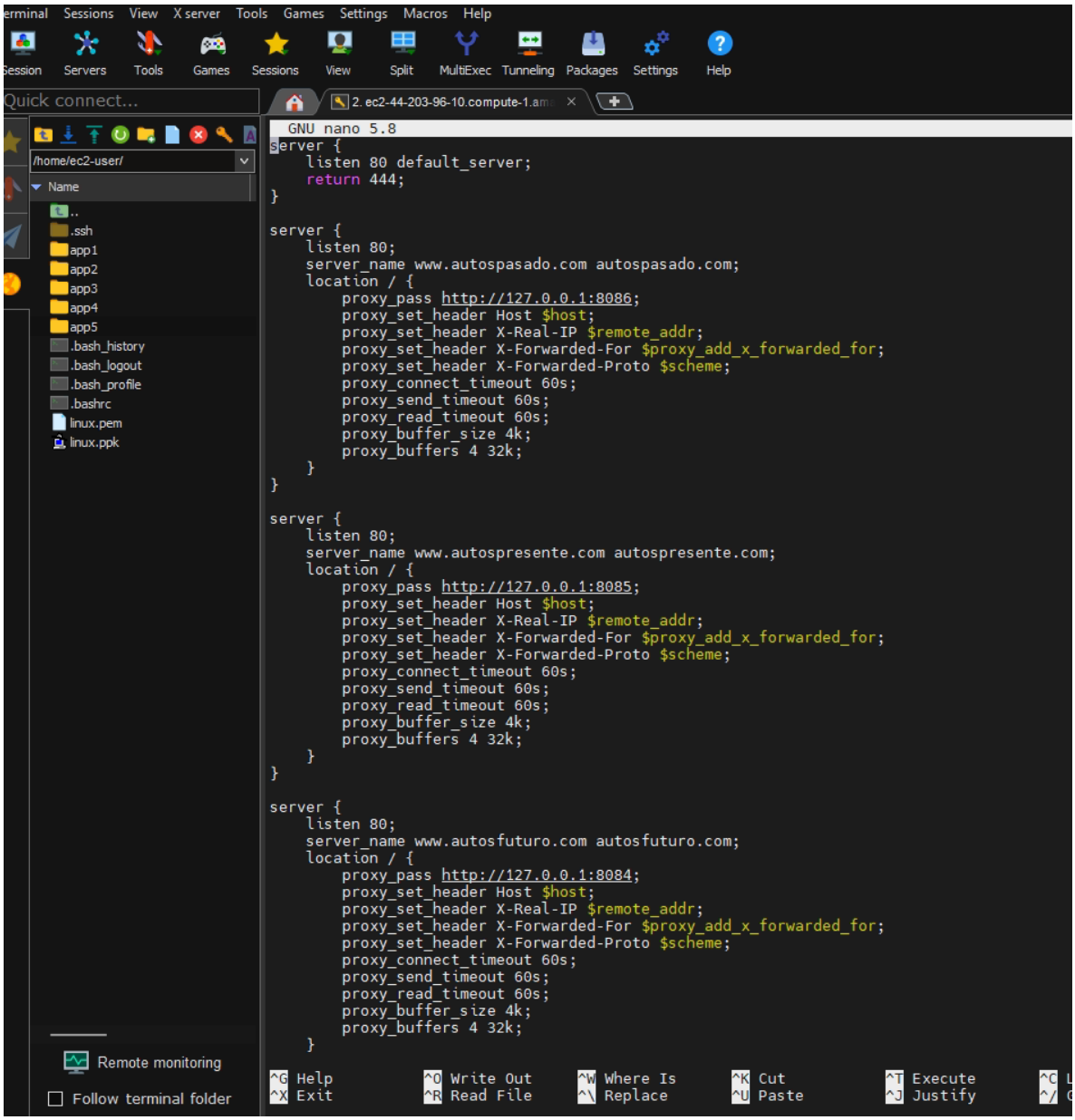
Acá lo que hicimos fue configurar Auto Scaling para que nuestras máquinas se ajusten solas según cuánto las usen. Creamos un grupo de Auto Scaling. Le dijimos que siempre tuviera al menos 1 instancia corriendo, que no pasara de 5, y que la capacidad deseada fuera 2. Luego le metimos una política para que creciera si el CPU se ponía pesado: si pasaba del 50%, que agregara otra instancia para ayudar, pero que esperara 5 minutos antes de decidir si necesitaba más.

CONTENEDORES EN INSTANCIA LINUX CON AWS

1. Preparé los contenedores en el servidor
Primero, en mi servidor de Amazon (AWS), creé tres contenedores con Docker. Cada uno tiene una página web diferente:
Uno para autos del futuro, conectado al puerto 8084, Otro para autos del presente, en el puerto 8085 Y otro para autos del pasado, en el puerto 8086. Usé una imagen de Apache (httpd) para cada uno, y configuré los contenedores para que las páginas que están dentro (que escuchan en el puerto 80) se conectaran a esos puertos en el servidor. Los lancé con comandos como docker run -d -p 8084:80 httpd y así para cada uno.

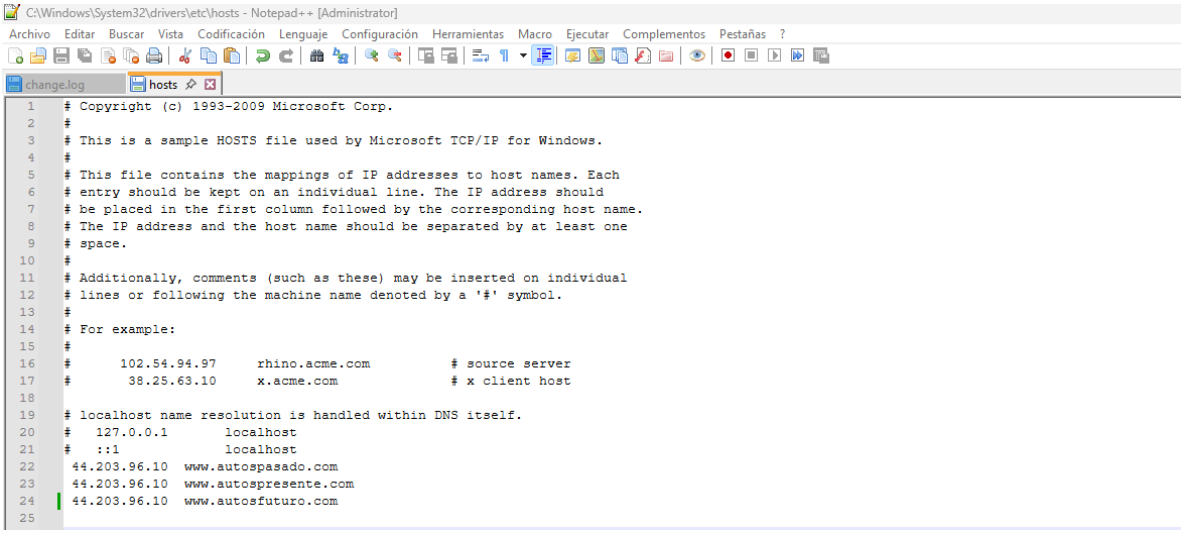


2. Instalé y configuré NGINX como intermediario
Después, instalé NGINX en el servidor, que es un programa que actúa como un ‘repartidor’ de tráfico. Lo configuré para que recibiera las visitas que llegan al servidor y las enviara al contenedor correcto según el nombre que pongo en el navegador. Por ejemplo: Si alguien escribe www.autosfuturo.com, NGINX lo manda al puerto 8084, donde está la página de autos del futuro. Lo mismo para los otros: www.autospresente.com va al puerto 8085 y www.autospasado.com al 8086. Para hacer esto, creé un archivo de configuración en el servidor, en la carpeta /etc/nginx/conf.d/, que se llama proxy.conf ahí puse instrucciones para NGINX para que cada persona dependiendo a que pagina quiera ir lo redirija a la correcta. También le dije a NGINX que escuchara en el puerto 80, que es donde llegan las visitas desde internet, y añadí una regla para que cualquier visita rara que no coincida con mis nombres devuelva un error.



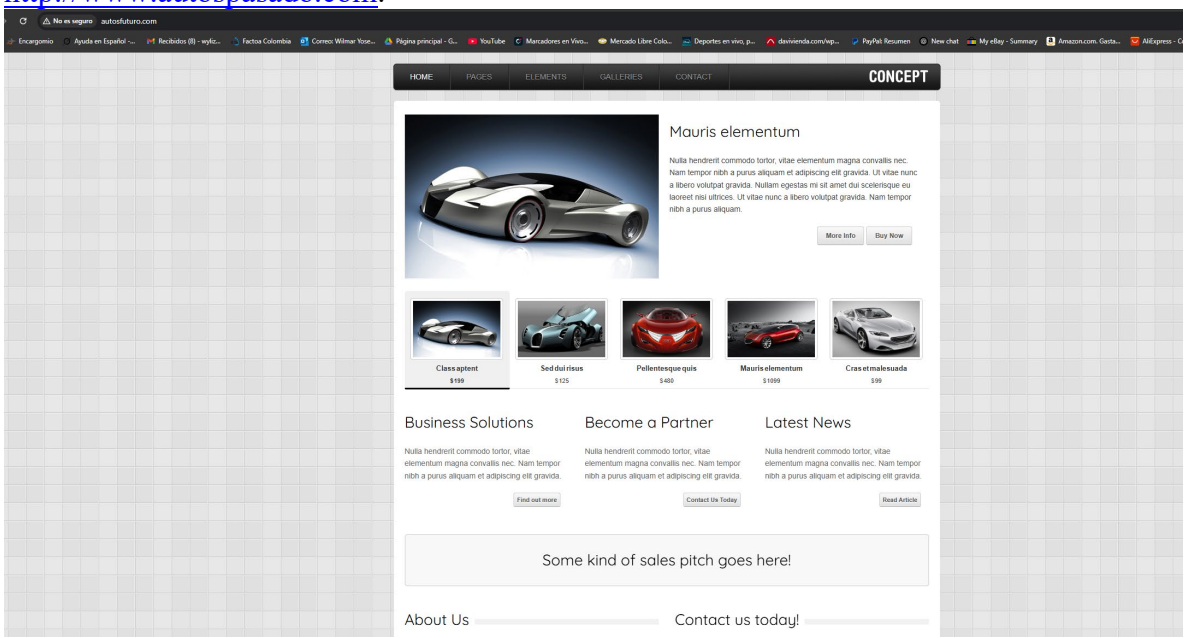
4. Hice que mi computador reconociera los nombres

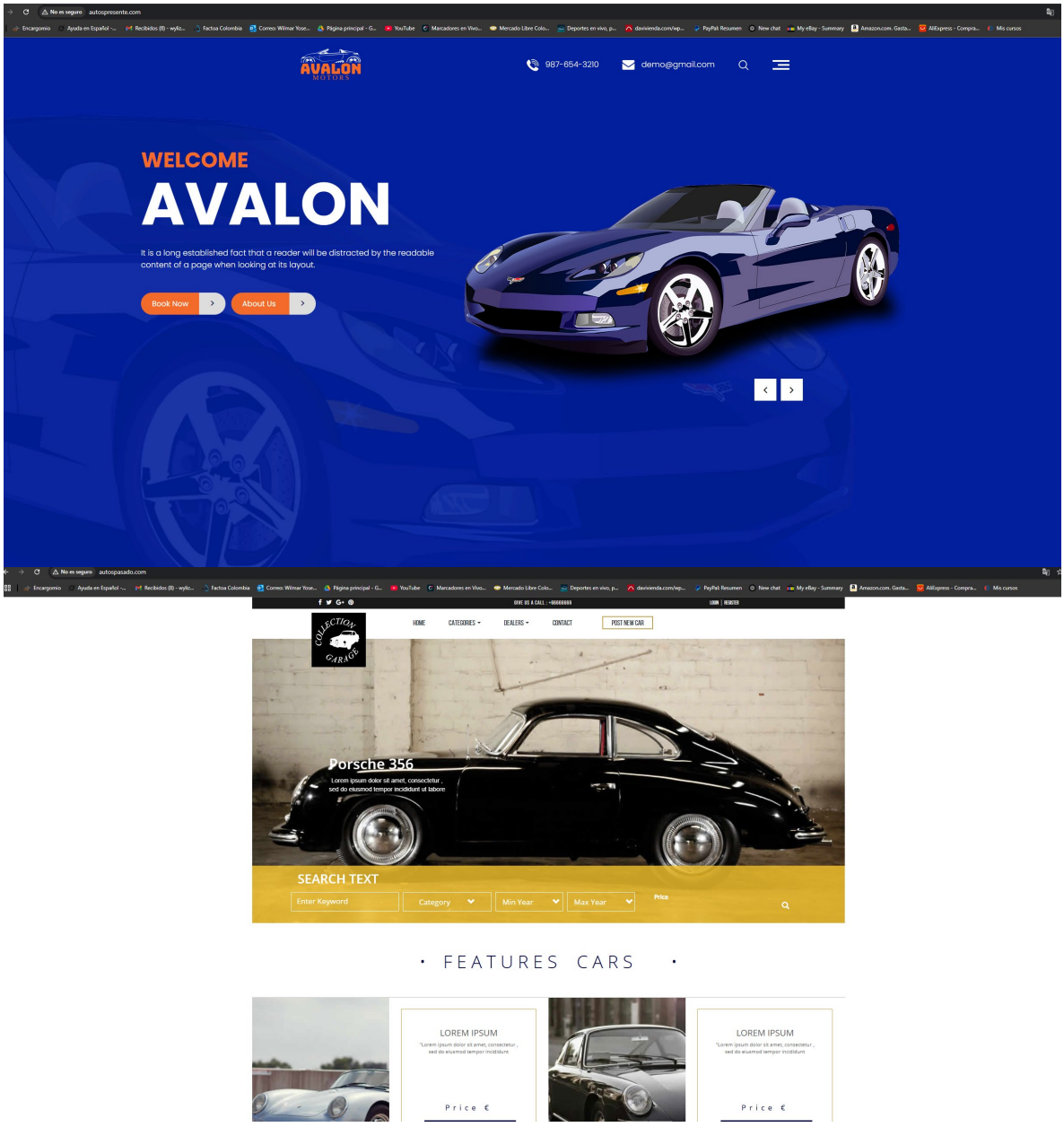
Como no podía comprar dominios de verdad, tuve que configurar mi propia computador para que entendiera que www.autosfuturo.com y los otros nombres apuntaran al servidor en AWS. Para eso, busqué la IP pública del servidor en la consola de AWS y edité un archivo en mi computadora que se llama hosts. En Windows está en C:\Windows\System32\drivers\etc\hosts, y lo abrí como administrador en Notepad++.



6. Todo funcionó

Después de ajustar eso, abrí el navegador en mi computadora y escribí http://www.autosfuturo.com, http://www.autospresente.com y http://www.autospasado.com.





Link del video con la explicación https://youtu.be/XgOkwPQn4z8

Conclusiones

Especificas:

Después de todo lo que hicimos, saqué unas ideas claras de cada parte. Con EC2, me quedó clarísimo que es súper fácil montar servidores virtuales y tenerlos listos en minutos, pero hay que saber bien qué tamaño elegir para no gastar de más. Lo de las instantáneas y las AMI me pareció una salvación, porque guardar respaldos o hacer copias para repetir servidores rápido es algo que cualquier negocio necesita. Con los grupos de seguridad, entendí que sin reglas bien puestas, cualquiera podría meterse, así que es como el candado de la puerta de tu sistema. Los balanceadores de carga me sorprendieron por cómo reparten el trabajo entre las máquinas, y eso hace que todo corra suave aunque llegue bastantes solicitudes de gente. Y con Auto Scaling, es como magia: las máquinas se ajustan solas y te quitan el estrés de estar adivinando cuántas necesitas. Cada cosa tiene su truco, pero juntas arman algo bien sólido.

Conclusiones generales:

En general, AWS me abrió los ojos a cómo funciona la nube de verdad. Aprendí que con estas herramientas puedes armar un sistema que sea rápido, seguro y que cargas pesadas, sin tener que comprar servidores o hardware costoso. Lo mejor es que todo se conecta: usas EC2 para las máquinas, los balanceadores y Auto Scaling para que no se caigan, las AMI y snapshots pa’ no perder nada, y los grupos de seguridad para que no te hackeen. Me quedó claro que esto no es solo para empresas grandes, sino que hasta un proyecto pequeño puede aprovecharlo sabiendo aprovechar sus ventajas. Eso sí, hay que practicar ya que al principio te puedes perder con tantas opciones, pero una vez que lo entiendes, es como tener el control de un mundo digital en tus manos.

Referencias

Amazon Web Services. (2025). AWS | Cloud computing - Servicios de informática en la nube. https://aws.amazon.com/es/

Amazon Web Services. (2025). Introducción a los conceptos básicos de nube de AWS. https://aws.amazon.com/es/getting-started/fundamentals-overview/

Amazon Web Services. (2025). Aprovisionamiento de infraestructura como código - AWS CloudFormation. https://aws.amazon.com/es/cloudformation/

Amazon Web Services. (2025). Qué es Amazon Web Services y para qué sirve. https://aws.amazon.com/es/what-is-aws/

Rodríguez, J. L., & Pérez, M. A. (2021). Amazon Web Services: Alternativa para el almacenamiento de información. Revista UD Innovación, 15(2), 45-56. https://revistas.udistrital.edu.co/index.php/udinnovacion/article/view/12345