TRABAJO DE GRADO
Opción Seminario-Diplomado.

Servicios en la Nube AWS

Corporación Universitaria Remington.
Ingeniería de Sistemas
Pregrado

Oscar Adrian Amaya Alvarez
Jorge Luis Salas Padilla
Juan Pablo Berrio López
Seminario
2025

Tabla de Contenidos

Resumen

En el presente trabajo exploramos la implementación de diferentes servicios ofrecidos por la computación en la nube, utilizando como proveedor principal a Amazon Web Services (AWS). A lo largo del desarrollo, se hace uso de herramientas como máquinas virtuales, balanceadores de carga, redes privadas virtuales (VPCs) y clústeres, con el objetivo de comprender su funcionamiento e integrarlos en una arquitectura básica pero funcional. Esta experiencia busca no solo afianzar los conocimientos teóricos adquiridos previamente, sino también desarrollar habilidades prácticas que permitan aprovechar el potencial de la nube para ofrecer soluciones escalables, seguras y eficientes en diferentes contextos tecnológicos.

Palabras clave

Instancia, VPCs, Linux, HTML, Servidor

Marco conceptual y contextual

Marco Conceptual

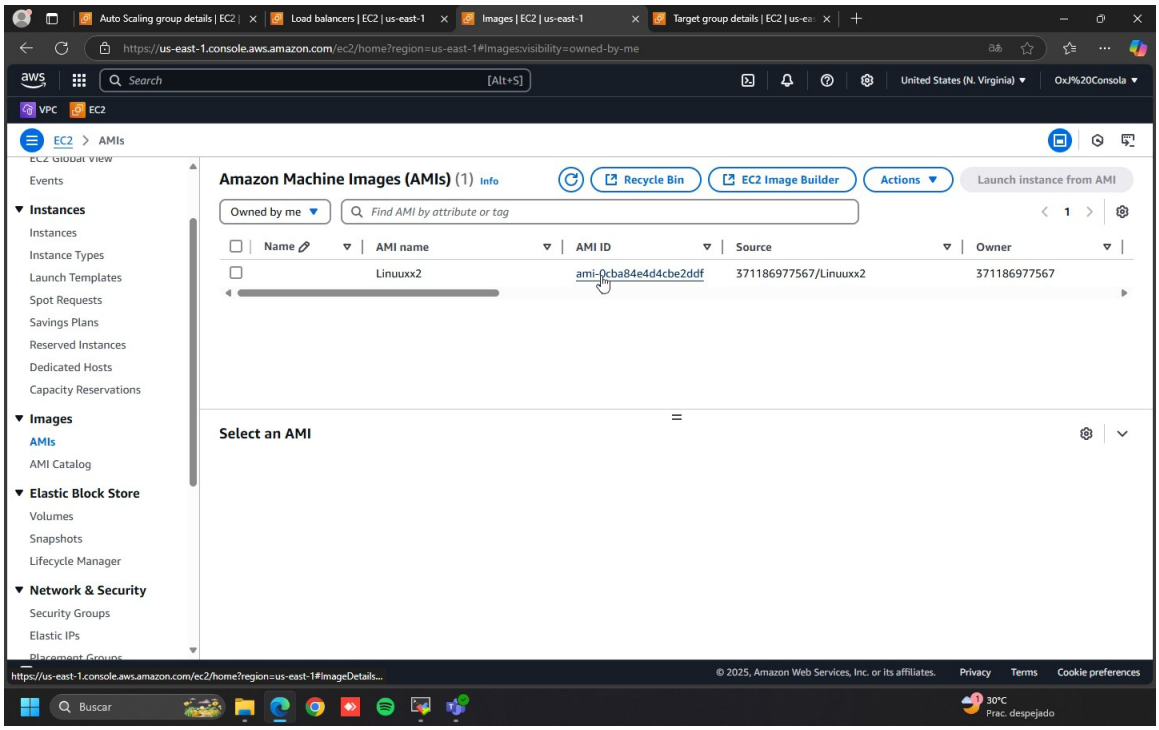
La computación en la nube se ha consolidado como una herramienta clave en el desarrollo de soluciones tecnológicas modernas. Este modelo permite acceder a recursos computacionales a través de internet, ofreciendo flexibilidad, escalabilidad y ahorro en infraestructura. En este trabajo se hace énfasis en conceptos como instancias (máquinas virtuales), redes privadas virtuales (VPCs), balanceadores de carga y clústeres, los cuales forman parte esencial de la arquitectura basada en la nube. Las instancias representan servidores virtuales que pueden configurarse para cumplir distintos roles dentro de un sistema. Por su parte, las VPCs permiten segmentar redes de forma segura dentro del entorno de AWS. Los balanceadores de carga distribuyen eficientemente el tráfico entre varios recursos, y los clústeres permiten agrupar servicios o aplicaciones para facilitar su gestión y escalabilidad.

Marco Contextual

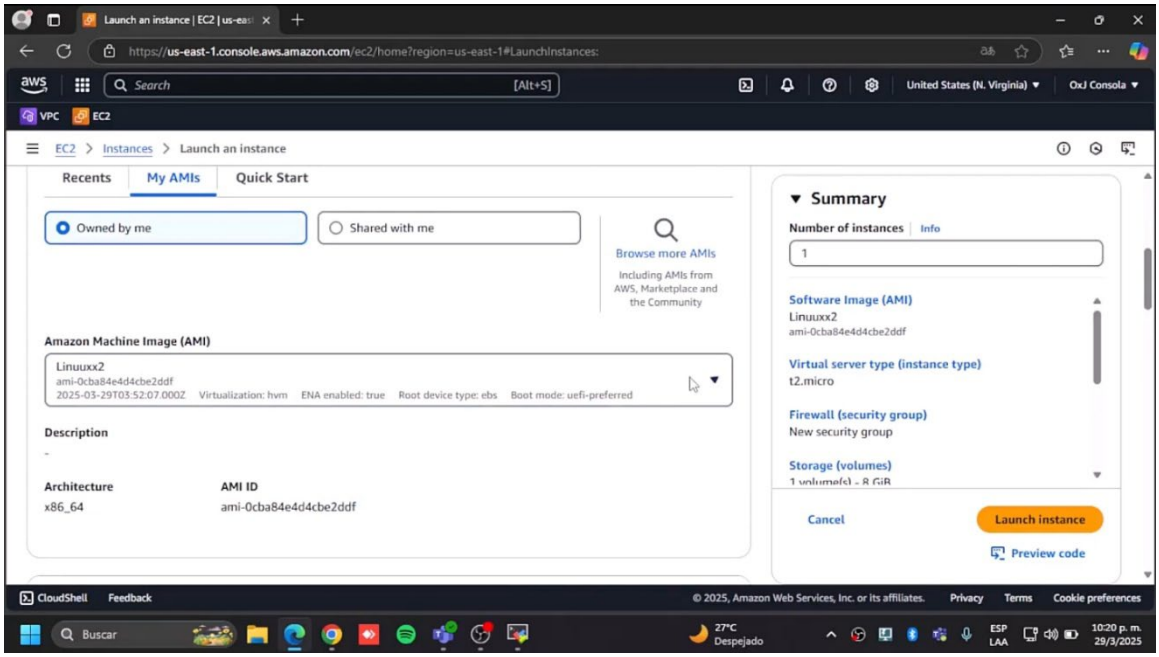
El uso de servicios en la nube ha ido creciendo a medida que más empresas e instituciones adoptan modelos digitales para sus operaciones. En el contexto educativo y formativo, aprender a implementar este tipo de soluciones se vuelve esencial para los futuros profesionales en sistemas e ingeniería. Este trabajo se desarrolla en un entorno académico, donde se busca que los estudiantes adquieran experiencia práctica utilizando herramientas reales como las que ofrece AWS. El objetivo es que puedan aplicar estos conocimientos tanto en entornos de prueba como en escenarios reales del mundo laboral, desarrollando sistemas más eficientes, seguros y adaptables a las necesidades actuales del mercado tecnológico. (AWS, s.f.)

Desarrollo e implementación del aprendizaje

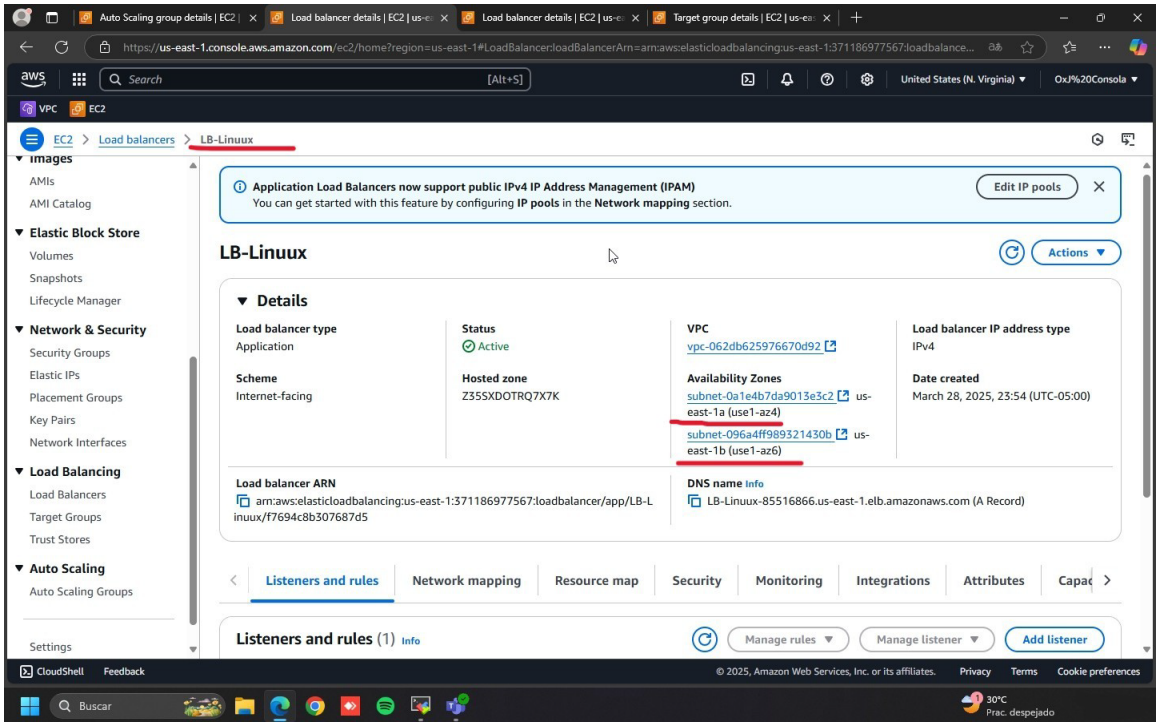
El trabajo práctico comenzó con la exploración de los servicios base que ofrece Amazon Web Services (AWS), enfocándose en la creación de una arquitectura funcional y adaptable. El primer paso consistió en la generación de una AMI (Amazon Machine Image) personalizada. Esta imagen permitió lanzar múltiples instancias EC2 con una configuración igual, lo cual ayudó a ahorrar mucho tiempo pues ya tiene las configuraciones cargadas.



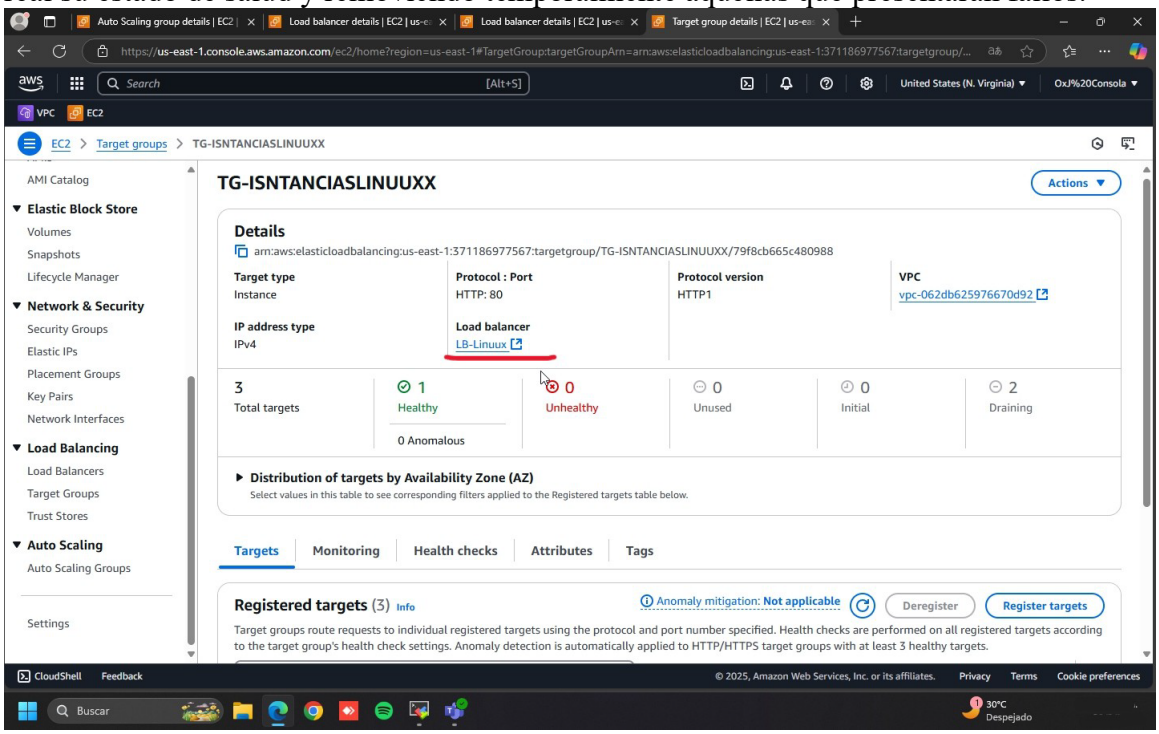
Una vez generada la imagen, se procedió a lanzar una nueva instancia EC2 basada en cierta AMI, con el fin de validar su correcto funcionamiento. Esta instancia representó la base del entorno web a desplegar, y se le aplicaron configuraciones iniciales como permisos de seguridad, asignación de IP pública y configuración de grupos de seguridad.



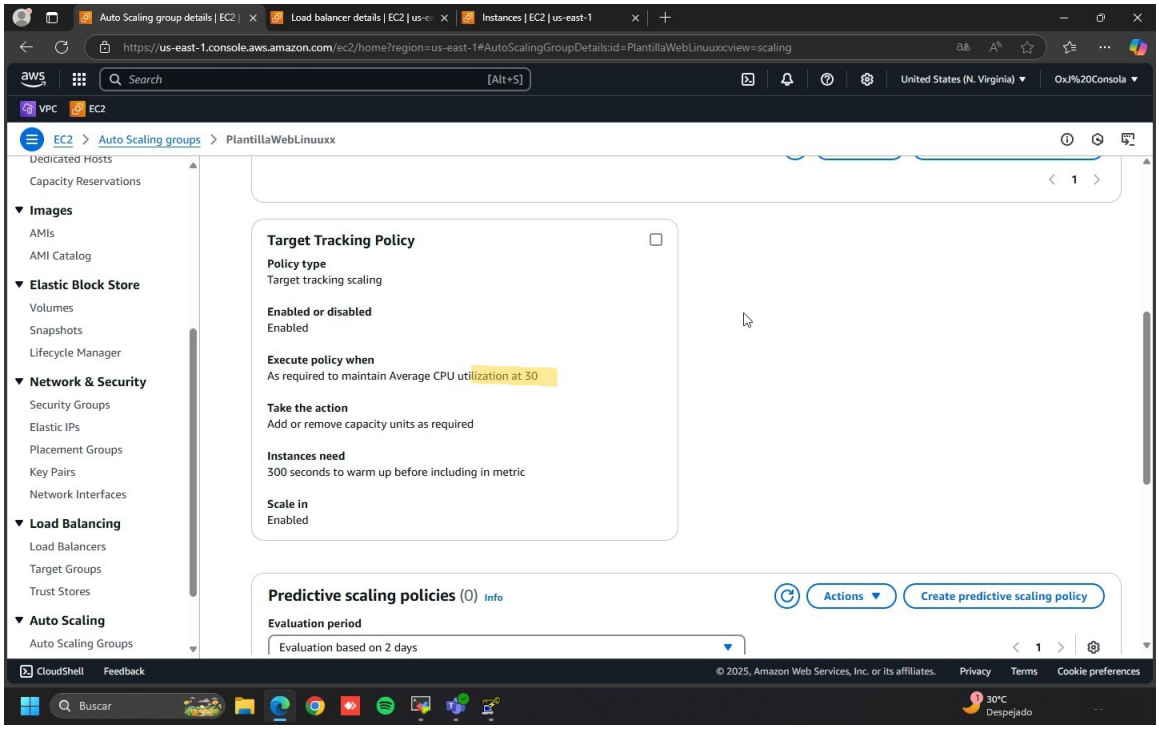
Después, se configuró un Load Balancer (balanceador de carga), un componente clave que permitió distribuir el tráfico entrante entre múltiples instancias. Este servicio incrementó la disponibilidad y estabilidad del sistema, evitando que una sola instancia soporte toda la carga de los usuarios.



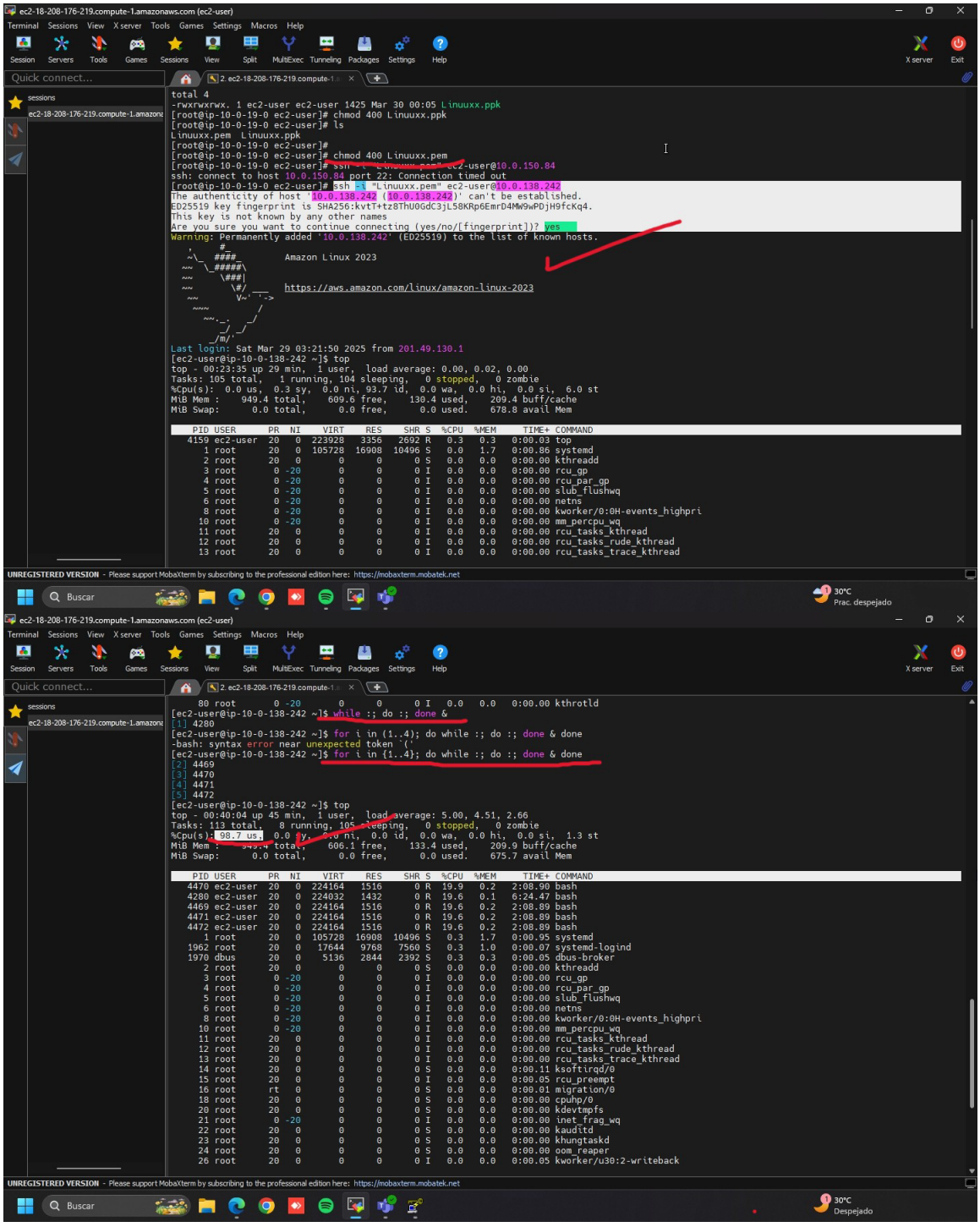
Para direccionar las solicitudes correctamente, se definió un Target Group. Este grupo se encargó de asociar las instancias activas al balanceador de carga, monitoreando en tiempo real su estado de salud y removiendo temporalmente aquellas que presentaran fallos.

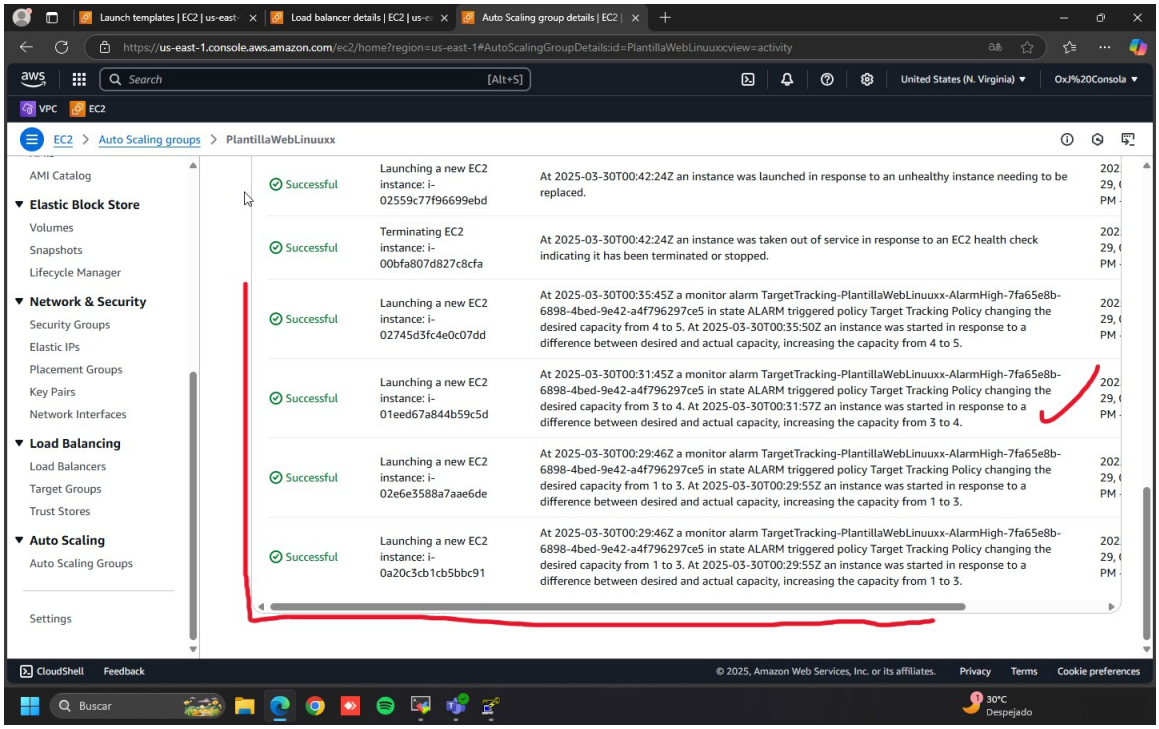


También se creó el Auto Scaling, un servicio que permite aumentar o disminuir el número de instancias activas dependiendo de la carga del sistema. En este caso, se configuró una política basada en el uso del CPU, con un umbral del 30 %. Cuando este límite se superaba, automáticamente se lanzaba una nueva instancia.



Para probar esta funcionalidad, se accedió a una instancia mediante conexión SSH, utilizando una clave .pem previamente generada y configurada con los permisos adecuados. Una vez dentro de la instancia, se ejecutaron scripts para generar una alta carga de procesamiento, lo cual activó el sistema de escalado automático.

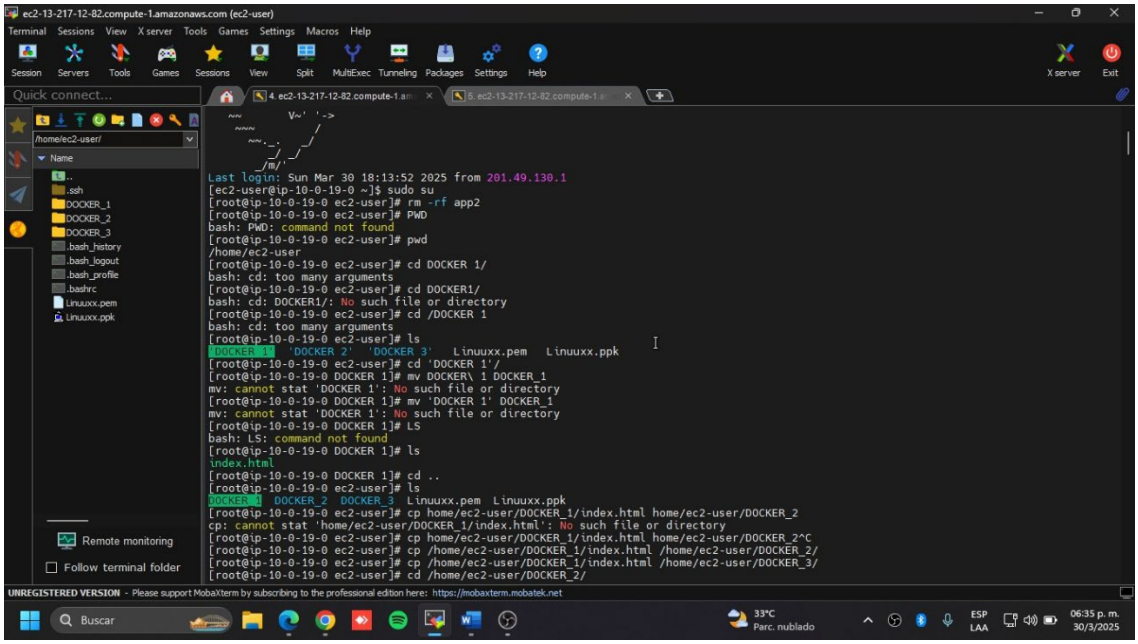

ec2-18-208-176-219.compute-1.amazonaws.com (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
sessions
ec2-18-208-176-219.compute-1.amazonaws.com
total 4
-rwxrwxrwx. 1 ec2-user ec2-user 1425 Mar 30 00:05 Linuuxx.ppk
[root@ip-10-0-19-0 ec2-user]# chmod 400 Linuuxx.ppk
[root@ip-10-0-19-0 ec2-user]# ls
Linuuxx.pem Linuuxx.ppk
[root@ip-10-0-19-0 ec2-user]#
[root@ip-10-0-19-0 ec2-user]# chmod 400 Linuuxx.pem
[root@ip-10-0-19-0 ec2-user]# ssh -i "Linuuxx.pem" ec2-user@10.0.150.84
ssh: connect to host 10.0.150.84 port 22: Connection timed out
[root@ip-10-0-19-0 ec2-user]# ssh -i "Linuuxx.pem" ec2-user@10.0.138.242
The authenticity of host '10.0.138.242 (10.0.138.242)' can't be established.
ED25519 key fingerprint is SHA256:kvtT+tz8ThU0GdC3jL58KRp6EmrD4MW9wPDjH9fcKq4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.138.242' (ED25519) to the list of known hosts.
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
Last login: Sat Mar 29 03:21:50 2025 from 201.49.130.1
[ec2-user@ip-10-0-138-242 ~]$ top
top - 00:23:35 up 29 min, 1 user, load average: 0.00, 0.02, 0.00
Tasks: 105 total, 1 running, 104 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 93.7 id, 0.0 wa, 0.0 hi, 0.0 si, 6.0 st
MiB Mem : 949.4 total, 609.6 free, 130.4 used, 209.4 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 678.8 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
4159 ec2-user 20 0 223928 3356 2692 R 0.3 0.3 0:00.03 top
1 root 20 0 105728 16908 10496 S 0.0 1.7 0:00.86 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 slub_flushwq
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
11 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Buscar
30°C
Prac. despejado
ec2-18-208-176-219.compute-1.amazonaws.com (ec2-user)
80 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kthrotld
[ec2-user@ip-10-0-138-242 ~]$ while :; do :; done &
[1] 4280
[ec2-user@ip-10-0-138-242 ~]$ for i in (1..4); do while :; do :; done & done
-bash: syntax error near unexpected token `('
[ec2-user@ip-10-0-138-242 ~]$ for i in {1..4}; do while :; do :; done & done
[2] 4469
[3] 4470
[4] 4471
[5] 4472
[ec2-user@ip-10-0-138-242 ~]$ top
top - 00:40:04 up 45 min, 1 user, load average: 5.00, 4.51, 2.66
Tasks: 113 total, 8 running, 105 sleeping, 0 stopped, 0 zombie
%Cpu(s): 98.7 us, 0.0 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 1.3 st
MiB Mem : 949.4 total, 606.1 free, 133.4 used, 209.9 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 675.7 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
4470 ec2-user 20 0 224164 1516 0 R 19.9 0.2 2:08.90 bash
4280 ec2-user 20 0 224032 1432 0 R 19.6 0.1 6:24.47 bash
4469 ec2-user 20 0 224164 1516 0 R 19.6 0.2 2:08.89 bash
4471 ec2-user 20 0 224164 1516 0 R 19.6 0.2 2:08.89 bash
4472 ec2-user 20 0 224164 1516 0 R 19.6 0.2 2:08.89 bash
1 root 20 0 105728 16908 10496 S 0.3 1.7 0:00.95 systemd
1962 root 20 0 17644 9768 7560 S 0.3 1.0 0:00.07 systemd-logind
1970 dbus 20 0 5136 2844 2392 S 0.3 0.3 0:00.05 dbus-broker
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 slub_flushwq
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
11 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
14 root 20 0 0 0 0 S 0.0 0.0 0:00.11 ksoftirqd/0
15 root 20 0 0 0 0 I 0.0 0.0 0:00.05 rcu_preempt
16 root rt 0 0 0 0 S 0.0 0.0 0:00.01 migration/0
18 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
21 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 inet_frag_wq
22 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kauditd
23 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd
24 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oom_reaper
26 root 20 0 0 0 0 I 0.0 0.0 0:00.05 kworker/u30:2-writeback
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Buscar
30°C
Despejado



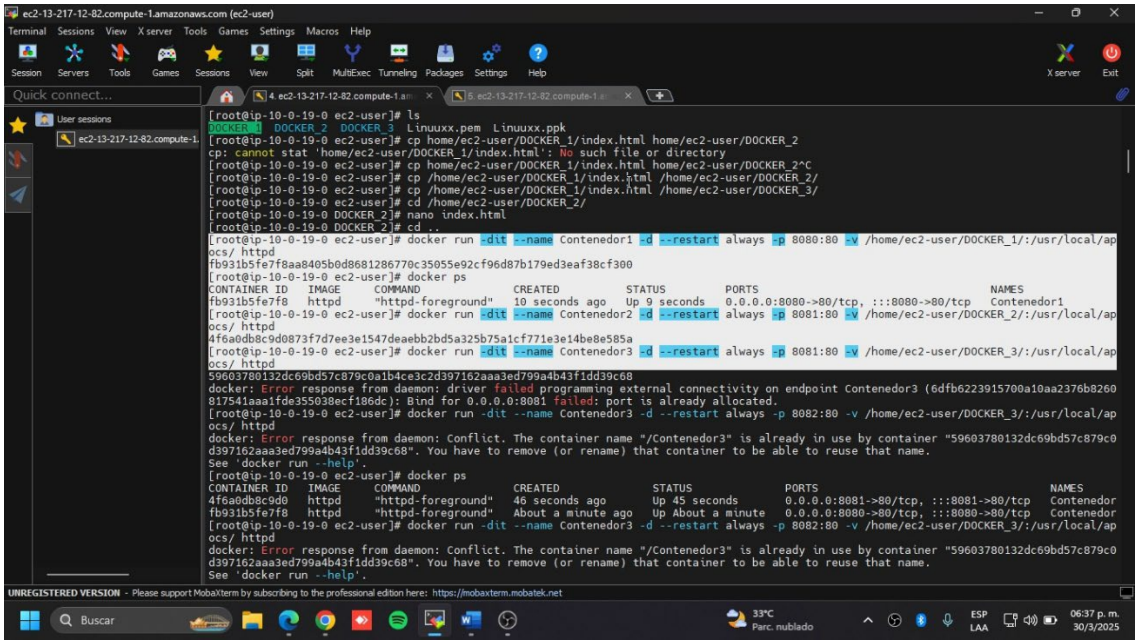
Este despliegue dinámico demostró que el sistema respondía correctamente ante situaciones de alta demanda, permitiendo mantener la disponibilidad del servicio sin necesidad de intervención manual.

De aquí pues voy a basarme ya en la segunda entrega…

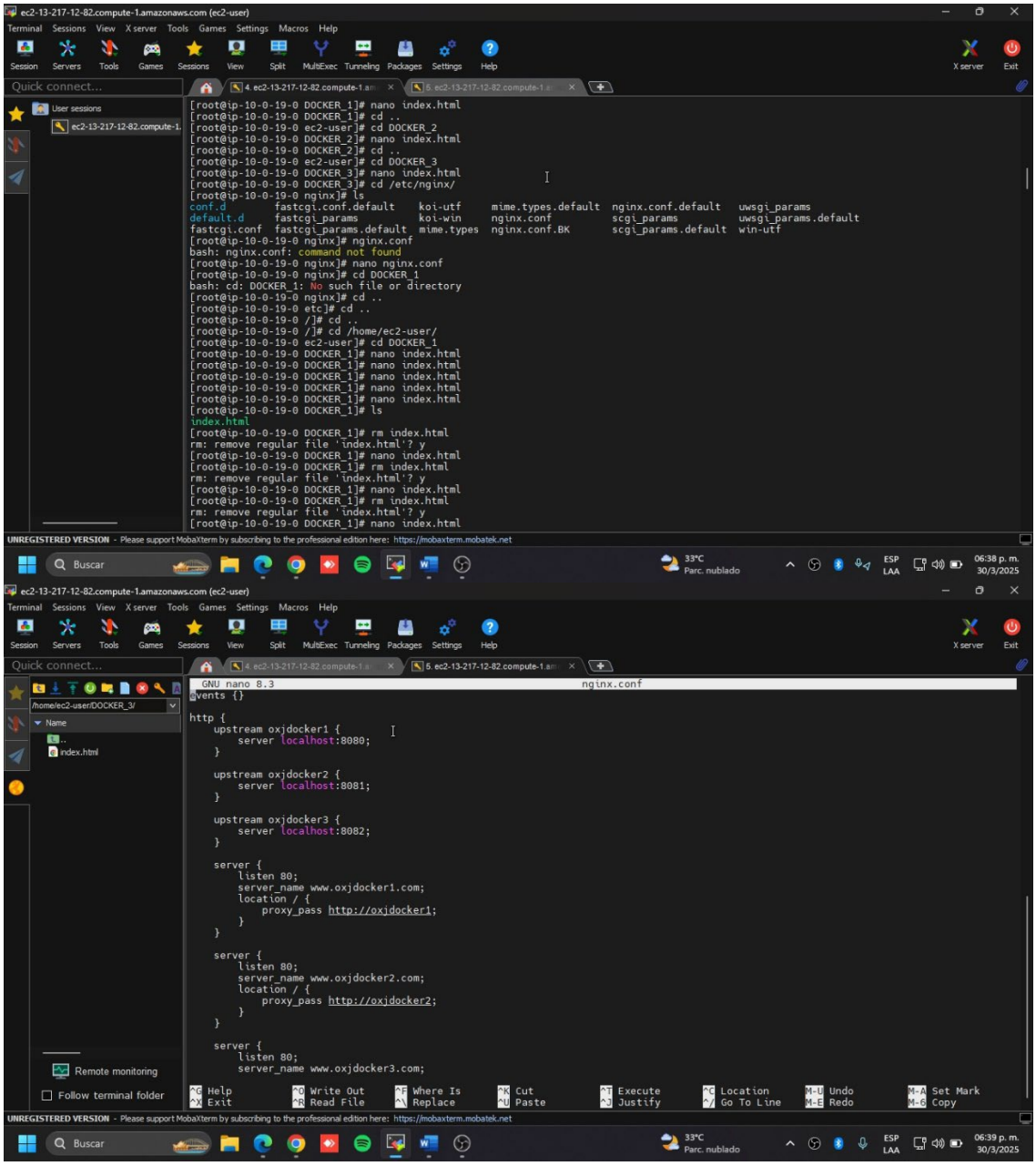
En esta fase se llevó a cabo la implementación de servicios web utilizando contenedores Docker, lo que permitió simular un entorno escalable y modular. Para ello, se creó una carpeta por cada contenedor, alojando en su interior un archivo index.html personalizado que servía como página principal del sitio web a desplegar.



Cada contenedor fue creado utilizando una imagen de Apache (httpd) como base. Durante su configuración, se especificaron puertos distintos para cada uno, asegurando que no existiera conflicto entre ellos y que cada uno pudiera ser accedido de forma independiente a través del navegador.

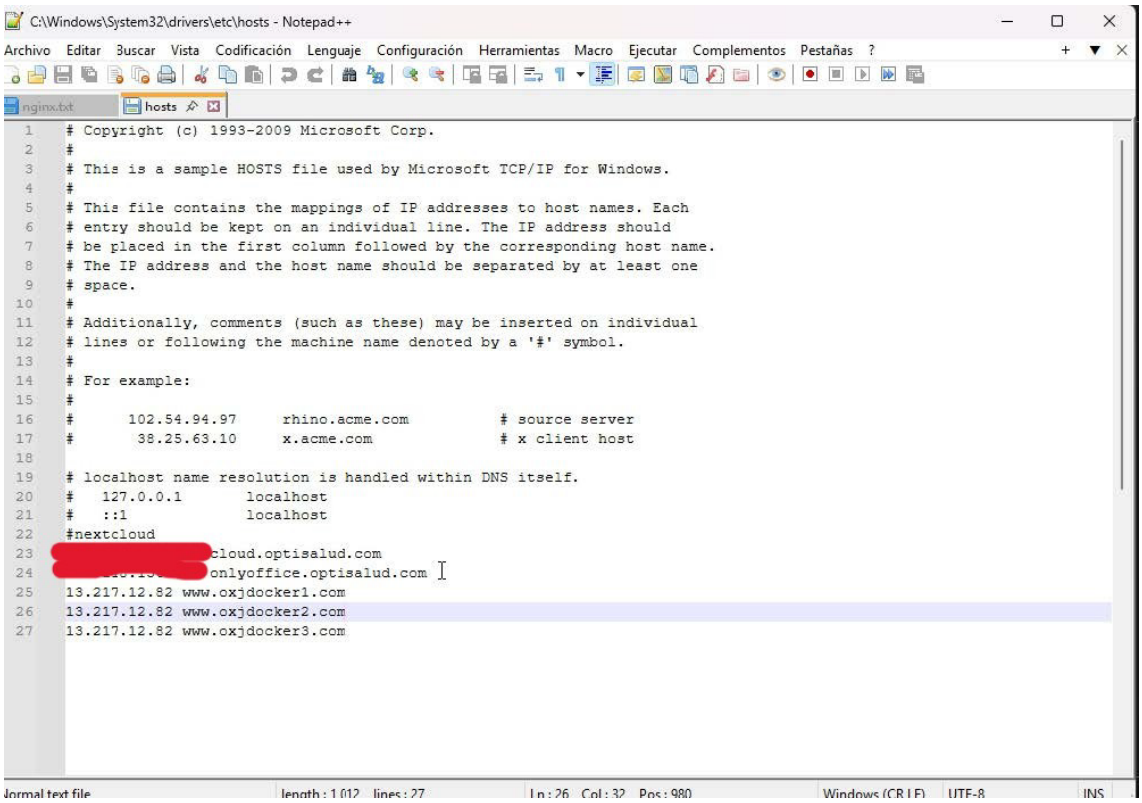


Con los contenedores corriendo correctamente, se procedió a levantar un contenedor adicional con la imagen de nginx, cuya función era actuar como balanceador y proxy inverso entre los sitios. En este punto fue fundamental la correcta configuración del archivo nginx.conf, donde se definieron las reglas de redirección para que, al ingresar una URL específica, el tráfico se dirigiera al contenedor correspondiente.

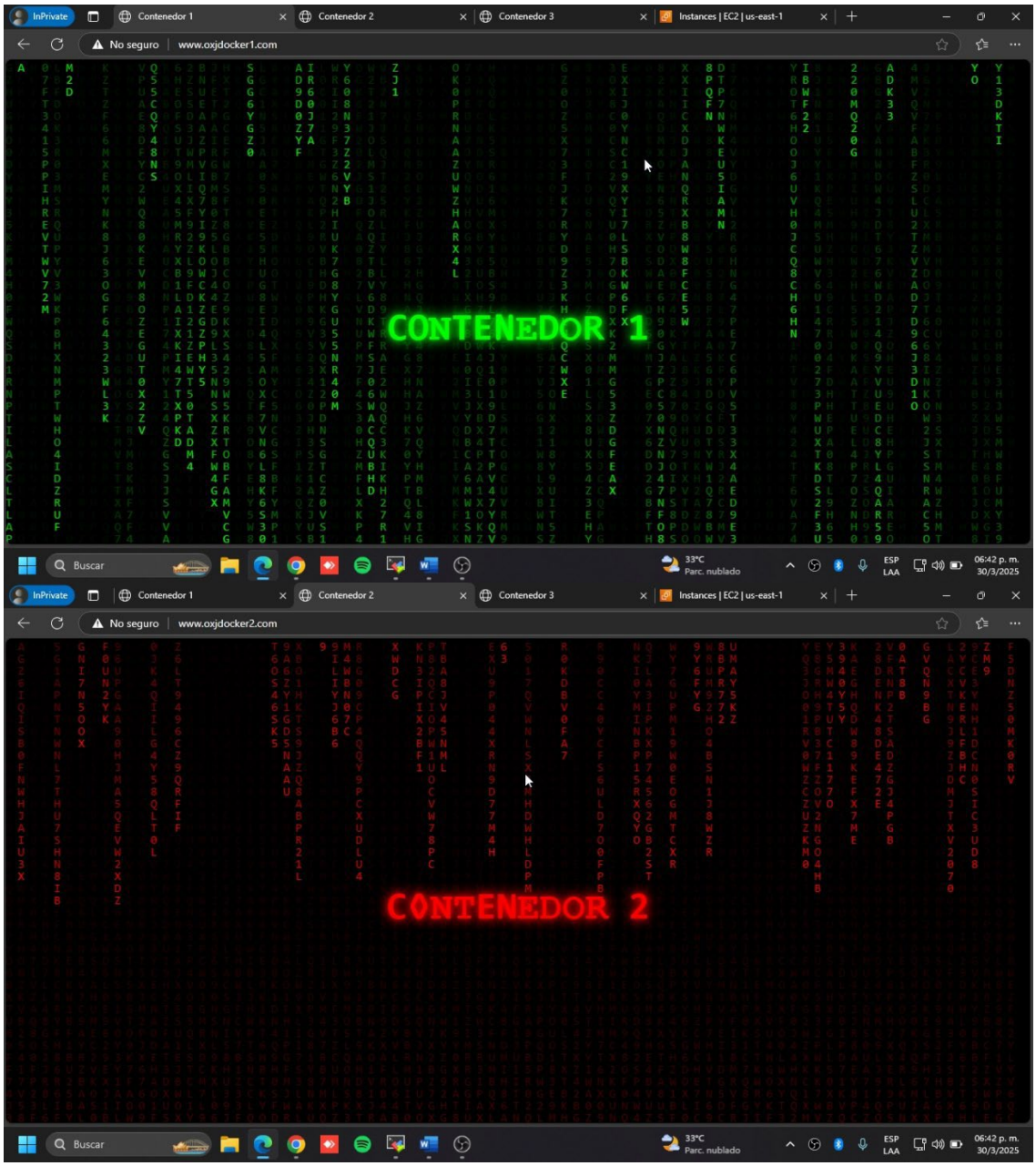


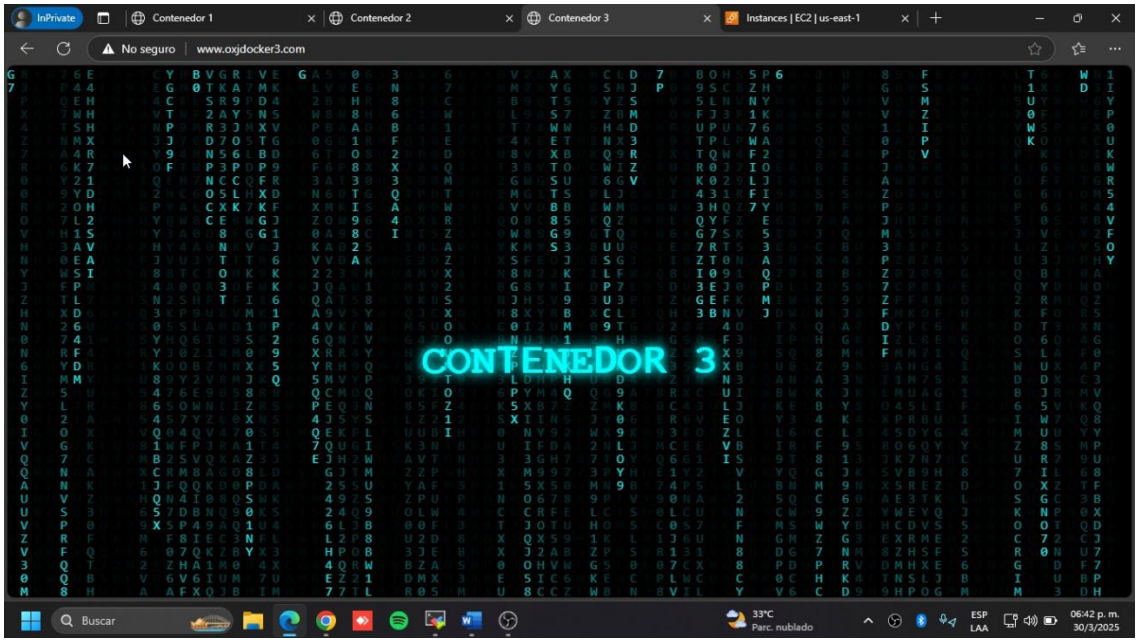
Además, se implementó la política restart always en todos los contenedores para garantizar su disponibilidad continua, incluso después de reinicios del sistema o fallos inesperados.

Para facilitar la navegación entre los distintos sitios, se editaron las entradas del archivo hosts en el sistema operativo local de Windows. Se asignaron nombres de dominio personalizados (como fueron, www.oxjdocker1.com, www.oxjdocker2.com) a la IP pública de la instancia en AWS. Esto permitió ingresar a cada sitio mediante un dominio fácilmente reconocible en lugar de recordar direcciones IP específicas o puertos individuales.



Finalmente, al ingresar desde el navegador a cada uno de los dominios configurados, se pudo acceder a los diferentes sitios web alojados en los contenedores respectivos, demostrando el correcto enrutamiento y funcionamiento del sistema en su conjunto.




InPrivate
Contenedor 1
Contenedor 2
Contenedor 3
Instances | EC2 | us-east-1
No seguro
www.oxjdocker3.com
CONTENEDOR 3
Buscar
33°C
Parc. nublado
ESP
LAA
06:42 p. m.
30/3/2025

Conclusiones

Este proyecto fue mucho más que aplicar conceptos técnicos. Desde el primer momento supimos que íbamos a poner a prueba todo lo aprendido, pero también descubrimos que enfrentarnos al entorno real de AWS nos exigía ir más allá de lo visto en clase.

Al principio, configurar las instancias, las reglas de red, los balanceadores y las AMIs personalizadas fue complicadito. No era solo seguir pasos: había que entender lo que estábamos haciendo y, sobre todo, qué impacto tenía cada configuración. A medida que avanzábamos, empezamos a ver cómo todo cobraba sentido y, al final, lograr que el sistema reaccionara automáticamente fue uno de los logros que más nos motivó.

Luego vino la parte de docker donde tuvimos que organizar los contenedores, crear nuestras propias carpetas con contenido, configurar Nginx como proxy inverso y asegurarnos de que cada sitio respondiera desde su dominio personalizado. Fue ahí donde entendimos la importancia de los detalles, porque algo tan sencillo como una línea mal escrita en el archivo de configuración podía hacer que nada funcionara.

Lo mejor de todo es que esta experiencia no solo nos dejó conocimientos técnicos, sino también esa sensación de satisfacción que da ver algo funcionando porque tú lo hiciste posible. Aprendimos resolviendo errores, corrigiendo cosas y volviendo a intentarlo. Al final, más que una nota, nos llevamos la confianza de saber que ya somos capaces de montar una solución en la nube, y eso vale muchísimo.

Referencias

AWS. (s.f.). *Computación en la nube de AWS para la educación*. Obtenido de AWS: https://aws.amazon.com/es/education/?wwps-cards.sort-by=item.additionalFields.sortDate&wwps-cards.sort-order=desc