



TRABAJO DE GRADO
Opción Seminario-Diplomado.

**Implementación de AWS en la Gestión y Balanceo de Cargas de Acuerdo con la Demanda
de Servicios**

Corporación Universitaria Remington.
Facultad de Ingenierías
Programa de Ingeniería de Sistemas

Valentina Salazar Isaza.
Juan Pablo Berrio López.
Seminario de Grado.
2025.

Dedicatoria

La vida es tal vez uno de los caminos más difíciles de transitar, pero si algo es cierto, es que nada sería posible sin la compañía de personas valiosas que incluso creen en ti cuando tú mismo dejaste de hacerlo.

Llegar hasta acá no solo es mi logro, sino el logro de todos aquellos que me acompañaron en el camino. Este trabajo se lo dedico a mis padres que me dieron según sus posibilidades todo lo necesario para hacerme una persona de bien. A mi hermana que desde pequeña me acompaña y siempre me ayudó con mis tareas. A mi tía Marina que es como una segunda mamá, pues siempre me ha apoyado y ha estado para mí. Las dos personas más importantes de mi mundo son mi mamá y mi papá. De mi papá aprendí que el mundo es un lugar fascinante y que a veces la creatividad puede abrir muchas puertas como él solía decir: “Todo lo que no sé hacer, me lo invento” porque eso hicieron los grandes personajes a los que alguna vez llamaron “locos”, creyeron en ellos e inventaron cosas inimaginables. De mi madre aprendí que la nobleza y el amor es lo más importante y se lo dedico por ser la persona más noble y sincera que conozco, porque espero ser alguna vez una pequeña parte de todo lo buena que ella es. Por último, me dedico esto a mí, a esa niña que temía estudiar y trabajar por miedo de no poder con todo, a mi fuerza, a mi valentía, a mis ganas de comerme el mundo y a las lágrimas que derrame en el camino, porque cada una de ellas me hizo más fuerte.
Que el camino sea aún más próspero de acá en adelante.

Tabla de Contenidos

Resumen.....	4
Marco conceptual y contextual	6
Desarrollo e implementación del aprendizaje.....	8
Entrega 1	8
Actividades Por Realizar 1.1.	8
Desarrollo 1.2.	8
Entrega 2	30
Actividades Por Realizar 2.1.	30
Desarrollo 2.2.	30
Conclusiones	53
Referencias.....	54

Resumen

En la actualidad, la computación en la nube nos permite tener recursos de computación (administración, base de datos, infraestructura) a través de internet. Con este nuevo modelo de computación ya no es necesario contar con entornos ONPREMISES (En las instalaciones). Entender, aprender y comprender los grandes beneficios de la computación nos va a permitir abrir nuestros caminos a nuevos horizontes.

Anteriormente, los ingenieros de sistemas estaban estigmatizados pues se decía que “solo servían para arreglar computadores”. Cuando realmente te adentras en el mundo y conoces un poco de cada uno, entiendes que hay un sinnúmero de posibilidades y entre ellos existe ser ingeniero Cloud.

El cloud (la nube), es un modelo al que algunas empresas temen, pues están tan acostumbrados a tener un ambiente local 100% administrado que desconfían de la seguridad y la administración que brinda AWS.

La infraestructura local, genera muchos gastos pues al ser 100% administrada por el usuario requiere de un servidor, una infraestructura, ubicación y unos cuidados que a su vez ocasionan gastos que sin duda con AWS pueden ser suplidos, esto es lo que llamamos cambiar gastos de capital (CAPEX) por gastos operativos (OPEX) ya que AWS es bajo demanda, es decir el cliente paga solo por lo que consume.

En este seminario inicialmente aprendimos de su historia pues para empezar a aprender una tecnología es de vital importancia comprender su origen, ¿de dónde proviene?, ¿por qué se destacó?, ¿qué la hace diferente?, y ¿por qué representa una mejor opción frente a sus competidores?

También aprendimos los beneficios de la computación en la nube, sus principales ventajas y modelos (modelos de computación y de implementación). Finalmente, se realizó una comparación entre las diversas plataformas que brindan servicios en la nube (AWS, Azure y Google Cloud) donde se mostraron las equivalencias en cada uno de sus servicios.

En la parte práctica, realizamos la creación de instancias (tanto Linux como Windows), entendimos el uso de un balanceador de carga que nos permitió entender el concepto de Escalabilidad que no es más que la capacidad de aumentar o reducir recursos según la demanda de acuerdo con una política de autoscaling o política de escalamiento. También, aprendimos sobre VPC, AMIs, el uso de contenedores en docker y finalmente con el uso de nginx y un proxy reverso realizamos la configuración de 3 sitios web que nos permitió hacer el llamado de la URL directamente desde el navegador. La configuración de estos puertos se realizó por el grupo de seguridad de la instancia.

Palabras clave

- AWS
- EC2
- Instancia
- CloudWatch
- Balanceador de Carga
- VPC
- Grupo de Seguridad
- Proxy Reverso
- Ngnix

Marco conceptual y contextual

Cuando hablamos del marco conceptual debemos hablar inicialmente AWS pues este es el aplicativo en el que se desarrolló el seminario de grado. Aprendiendo sus funcionalidades, servicios y aplicaciones más comunes pudimos entender por qué es uno de los servicios con más demanda en la actualidad.

AWS (Amazon Web Service): Es la entrega de servicios de computo (infraestructura, base de datos, servicios) a través de internet. AWS es un servicio bajo demanda, significa que el cliente paga por lo que usa, de esta forma un usuario paga solo por la infraestructura que requiere para su aplicación, lo que permite ahorrar costos y ofrecer soluciones seguras ya que cuenta con infraestructura a nivel global que permite alta disponibilidad.

Amazon EC2 (Elastic Compute Cloud): Una instancia en AWS es una maquina virtual que permite tener su propio sistema operativo de acuerdo con la necesidad. Es como tener un servidor virtual en la nube con un buen rendimiento y la posibilidad de elegir el procesador, almacenamiento, infraestructura y sistema operativo al momento de su creación. (AWS 2025)

VPC (Virtual Private Cloud): Es un servicio que permite lanzar recursos brindando control total sobre el entorno de redes virtuales (Amazon, 2025).

Al crear la VPC se agregan las instancias que estarán relacionadas, además de definir el número de zonas de disponibilidad, es importante tener en cuenta que entre más zonas de disponibilidad (A-Z) tenga un aplicativo, mayor será su capacidad de recuperación antes fallas, pues si una zona falla, la otra podrá recibir el tráfico en las demás instancias y así evitar caídas.

AMI (Amazon Machine Images): Es una copia de una instancia realizada hasta una fecha en específico con las configuraciones realizadas hasta ese momento. Con ella, se puede crear una nueva instancia tomando la configuración de la primera creada. El proceso de toma de AMIs nos permite tener una imagen exacta de una instancia lo que facilita la configuración y toma de backups.

Grupo de Seguridad: Permite filtrar el trafico entrante y saliente de una instancia

Elastic Load Balancer (Balanceador de Carga): Es el encargado de distribuir el trafico que ya ha ingresado por medio de las instancias (Amazon, 2025), lo que permite mejorar la escalabilidad de las aplicaciones al momento de su ejecución.

Target Group: Es el encargado de enrutar las solicitudes de las instancias solicitadas, por medio de un puerto que es configurado por el usuario al momento de su utilización. Un grupo puede contener muchas instancias

Snapshot: Es una copia de seguridad o copia de todo el ambiente, se realiza con el fin de salvaguardar la información de sistema. Es básicamente una foto exacta del disco duro.

Políticas de backup: Son las que permiten definir las frecuencias de toma del backup, se pueden realizar backups a discos duros o a instancias. Una instancia puede tener muchos discos duros. En esas políticas también están las retenciones que es donde se define, cuanto tiempo se van a almacenar estos backups en la nube.

Escalamiento: En AWS el escalamiento se define como la capacidad de aumentar o disminuir de acuerdo con las necesidades. El escalamiento vertical permite crecer las características de las maquinas implicadas (ejemplo: aumentar la capacidad de CPU de la instancia uno) mientras que el escalamiento horizontal se basa en aumentar la cantidad de maquinas con el fin de mejorar la distribución del tráfico.

Autoscaling (Auto escalado): Es la capacidad de aumentar o disminuir instancias según la demanda, este proceso se configura en AWS y puede ser manual o automático.

HealthCheck (Monitoreo de Salud): Sirve para que el balanceador identifique cuales son los servidores que están conectados y funcionando de forma adecuada, lo que permite que, si alguno falla, la solicitud pueda ser reenviada.

Lauche Temple: Es una plantilla de lanzamiento, es la configuración que el autoscaling utilizará para crear las nuevas instancias de acuerdo con la demanda de tráfico.

Docker: Permite el despliegue de aplicaciones en contenedores.

Docket Hub: Es un Hub de imágenes, es una pagina web que ya posee aplicaciones configuradas las cuales pueden ser usadas para validar el funcionamiento del Docker.

Ngnix: Es un software que puede ser utilizado como servidor web.

ECS (Elastic Container Service): Es una aplicación que administra el despliegue y la creación de contenedores en AWS

Cluster: Conjunto de instancias con los que se van a ejecutar los contenedores.

Desarrollo e implementación del aprendizaje

Entrega 1

Actividades Por Realizar 1.1.

Investigue cuáles fueron los orígenes de la virtualización y la computación en la nube, haga una línea de tiempo con los diferentes hitos para que hoy en día exista el nivel tecnológico que hemos alcanzado en la nube. Realice una comparación de nombres de servicios entre AWS, Azure y GCP.

Implemente un balanceador de carga con un Autoscaling, las instancias deben ser Linux, debe tener una aplicación web básica que cargue automáticamente al crearse la instancia.

Se debe mostrar cómo funciona el Autoscaling al terminar instancias manualmente.

Se debe crear una política de autoscaling al tener un aumento de CPU de un umbral definido. Ej: 50%

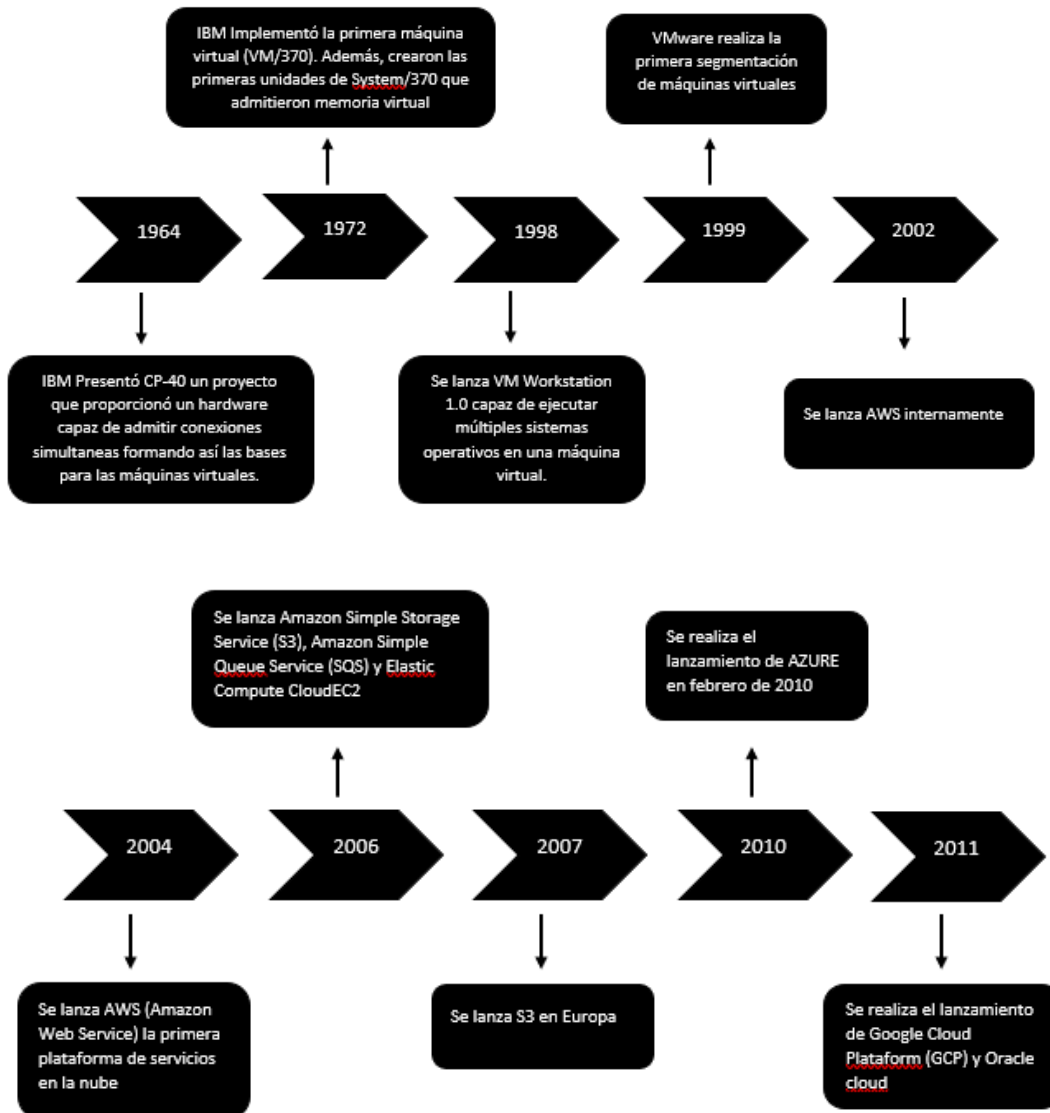
Desarrollo 1.2.

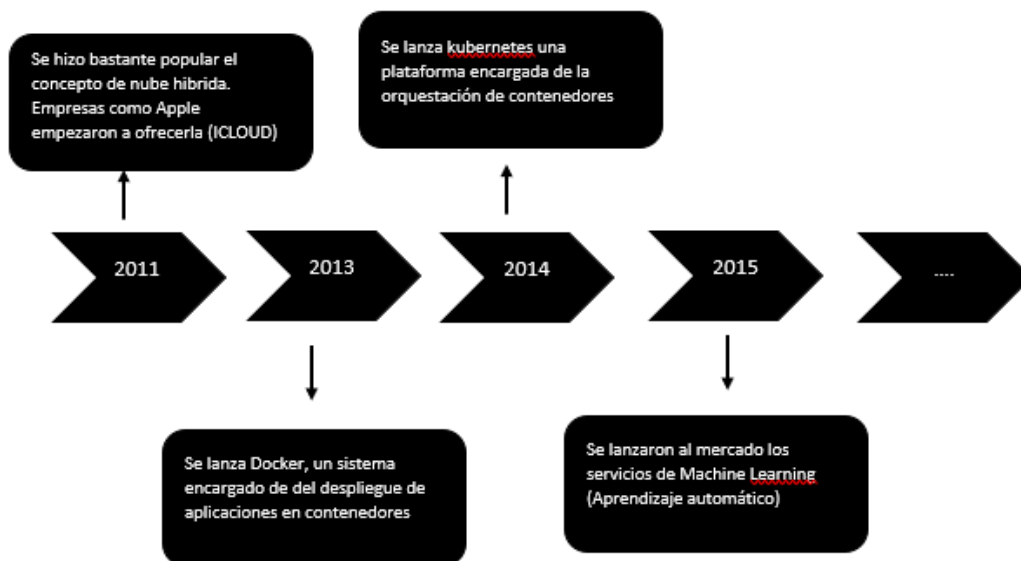
Línea de Tiempo

Para poder entender los orígenes de la virtualización y la computación en la nube es importante conocer las definiciones de cada uno. La virtualización no es más que una tecnología que es capaz de crear representaciones virtuales de servidores por medio de recursos de hardware más eficientes, por su parte, la computación en la nube es la entrega de servicios informáticos (servidores, almacenamiento, infraestructura) a través de internet.

Ambos, nos permiten aprovisionar recursos de tal forma que no es necesario contar con infraestructura física ya que la virtualización nos permite contar con recursos que pueden

aumentar o disminuir según sea su demanda. A continuación, se nombrarán algunos hitos importantes que permitieron llegar al nivel tecnológico en el que nos encontramos actualmente.





SERVICIOS DE AWS, GOOGLE CLOUD Y AZURE

En el siguiente cuadro se evidencian las diversas categorías que son usadas en cada proveedor de computación en la nube, cada una de ellas, cuenta con servicios que permiten realizar las mismas funciones, pero se llaman diferente, la elección del servicio más apropiado dependerá de los gustos y necesidades del usuario.

Servicios en la nube	AWS	GOOGLE CLOUD	AZURE
Servidores virtuales	Elastic Compute Cloud (EC2)	VM Instances	VMs
Computación sin servidor	Lamba	Cloud Functions	Azure Functions
Dockers	Elastic Container Service (ECS)	Container Engine	Container Service
Kubernetes	Amazon Elastic Kubernetes Service (EKS)	Kubernetes Engine	Kubernetes Service
Almacenamiento de objetos	Amazon Simple Storage Service (S3)	Cloud Storage	Block Blob
Almacenamiento de archivos	Glacier	Coldline	Archive Storage
Almacenamiento de archivos escalables	Amazon Elastic File System (EFS)	ZFS/ Avere	Azure Files
Entrega de contenido	Amazon Cloud Front	Cloud CDN	Delivery Network
Almacenamiento de datos estructurados y no estructurados	Amazon Redshift	Big Query	SQL Warehouse

PARTE PRACTICA

A continuación, se desarrollaron los puntos 2 y 3 donde se muestra como la implementación de un balanceador de carga permitió que de acuerdo con una política de escalado (autoscaling) las maquinas aumentaran o disminuyeran su capacidad según la demanda.

Primeramente, se crearon 3 instancias Linux donde luego de iniciarlas se validó que cada una de ellas cargara una aplicación web básica al iniciar.

Instancia Linux 1

The screenshot displays the AWS Management Console interface for an EC2 instance named 'Linux1'. The instance is in a 'Running' state. The public IPv4 address is highlighted with a red box and is 44.203.111.150. The private IPv4 address is 10.0.9.121.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro	-	View alarms +	us-east-1a	-	-	-
Linux4	i-07551bba6074eda8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	-
Linux3	i-0bdb20d924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	-
Linux1	i-02fd541ddc631021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	-

i-02fd541ddc631021d (Linux1)

Instance summary info

Instance ID: i-02fd541ddc631021d

Public IPv4 address: 44.203.111.150 | open address

Private IPv4 addresses: 10.0.9.121

Instancia Linux 3

The screenshot displays the AWS Management Console interface for an EC2 instance named 'Linux3'. The instance is in a 'Running' state. The public IPv4 address is highlighted with a red box and is 3.90.247.9. The private IPv4 address is 10.0.24.96.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro	-	View alarms +	us-east-1a	-	-	-
Linux4	i-07551bba6074eda8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	-
Linux3	i-0bdb20d924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	-
Linux1	i-02fd541ddc631021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	-

i-0bdb20d924d512622 (Linux3)

Instance summary info

Instance ID: i-0bdb20d924d512622

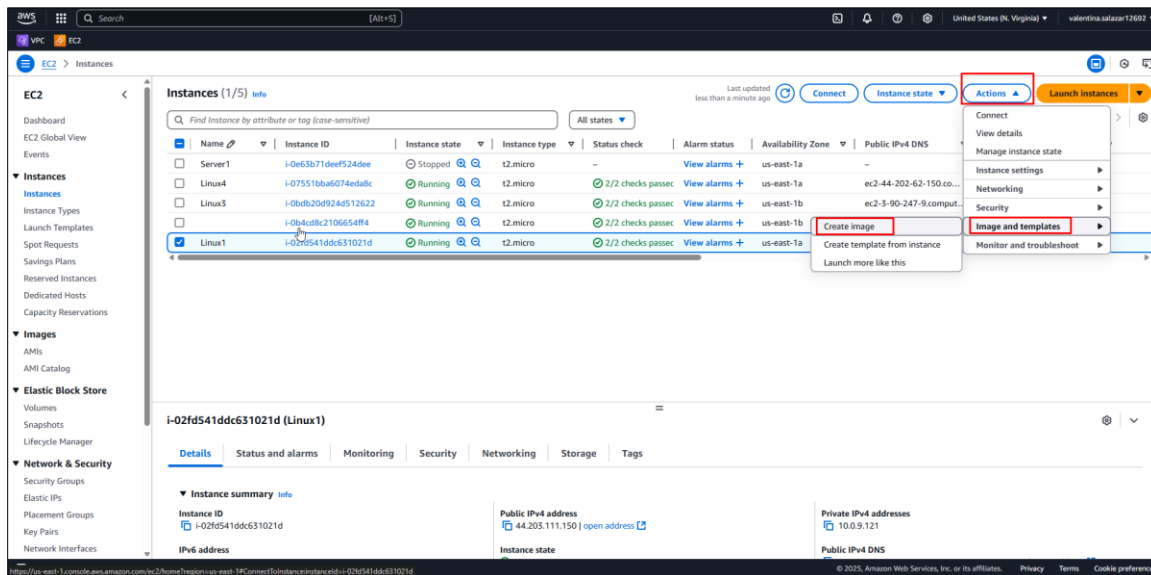
Public IPv4 address: 3.90.247.9 | open address

Private IPv4 addresses: 10.0.24.96

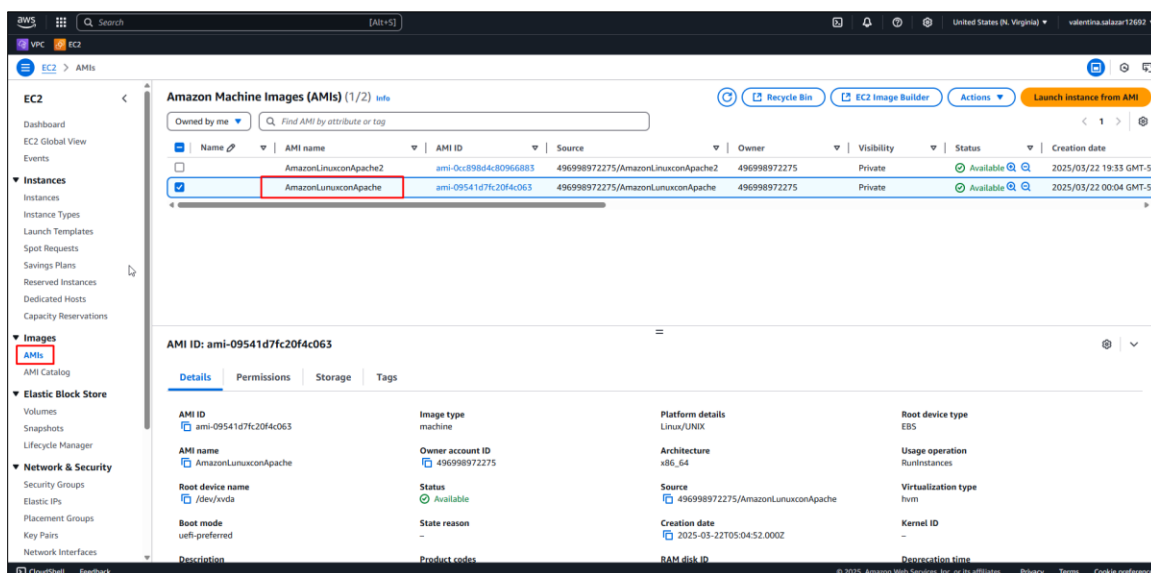
Se realizó la creación de una AMI llamada “AmazonLinuxconApache”, la cual es una copia exacta de la instancia Linux1 hasta el momento de las configuraciones realizadas.

Para realizar su creación, nos dirigimos a la instancia y allí seleccionamos la ruta

Actions/Image and templates/ Create Image



Luego de creada, esta fue almacenada en el Panel de EC2 en la opción Images/ Amis como lo evidencia la imagen siguiente.



Por medio de la AMI previamente establecida, se realizó la creación la instancia Linux4 que como se mencionó es una copia con las mismas configuraciones de Linux1

The screenshot shows the AWS Management Console interface for EC2 instances. At the top, there's a search bar and a table of instances. The instance 'Linux4' (ID: i-07551bba6074eda8c) is selected. Below the table, the details for 'Linux4' are shown, including its state (Running), instance type (t2.micro), and public IPv4 address (44.202.62.150), which is highlighted with a red box.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-	-	-
Linux4	i-07551bba6074eda8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	-
Linux3	i-0bdtb20d924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	-
Linux1	i-02fd541ddc631021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	-

Para ejecutar cada uno de los servidores de las instancias, se utilizó la aplicación MobaXterm donde por medio de un host, un usuario (dado por la instancia) y una llave privada (se creó al momento de lanzar la instancia) se realizó la conexión a cada servidor. Este proceso de conexión se realizó por cada una de las 2 instancias principales (Linux1 y Linux3)

The screenshot shows the 'Connect to instance' dialog in the AWS Management Console. The instance ID is 'i-02fd541ddc631021d (Linux1)'. The 'SSH client' tab is selected. The dialog provides instructions on how to connect to the instance using an SSH client. The public IPv4 address 'ec2-44-203-111-150.compute-1.amazonaws.com' is highlighted with a red box.

Connect to your instance i-02fd541ddc631021d (Linux1) using any of these options

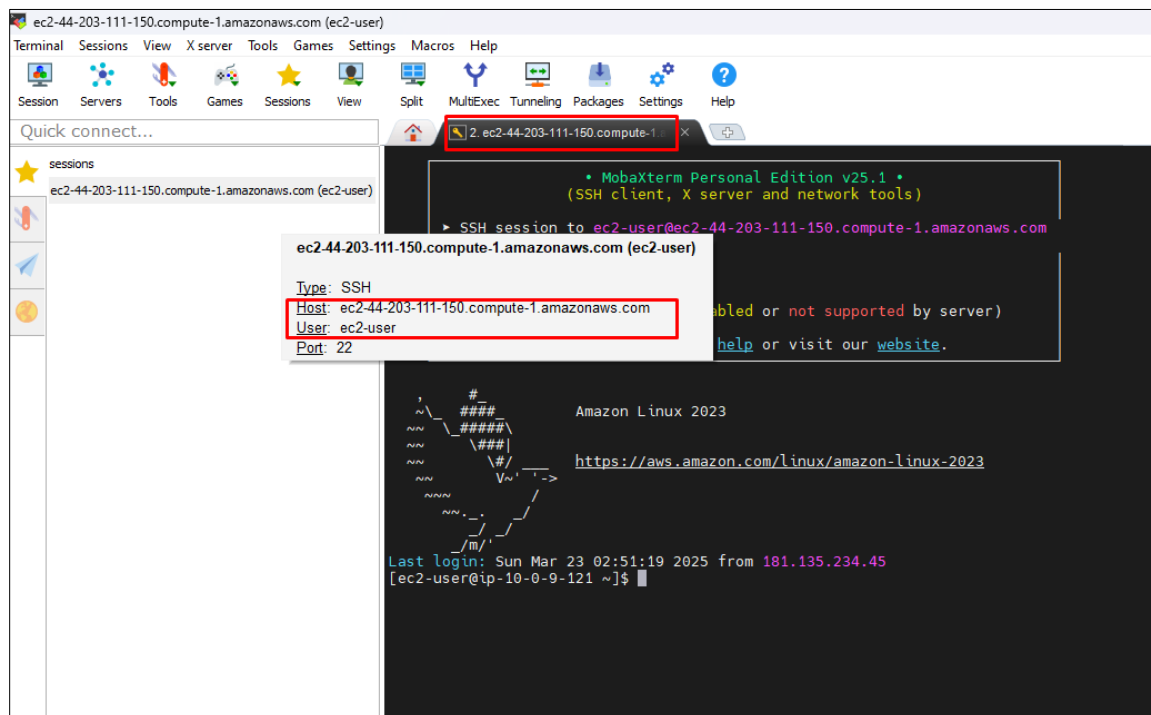
EC2 Instance Connect | Session Manager | **SSH client** | EC2 serial console

Instance ID
i-02fd541ddc631021d (Linux1)

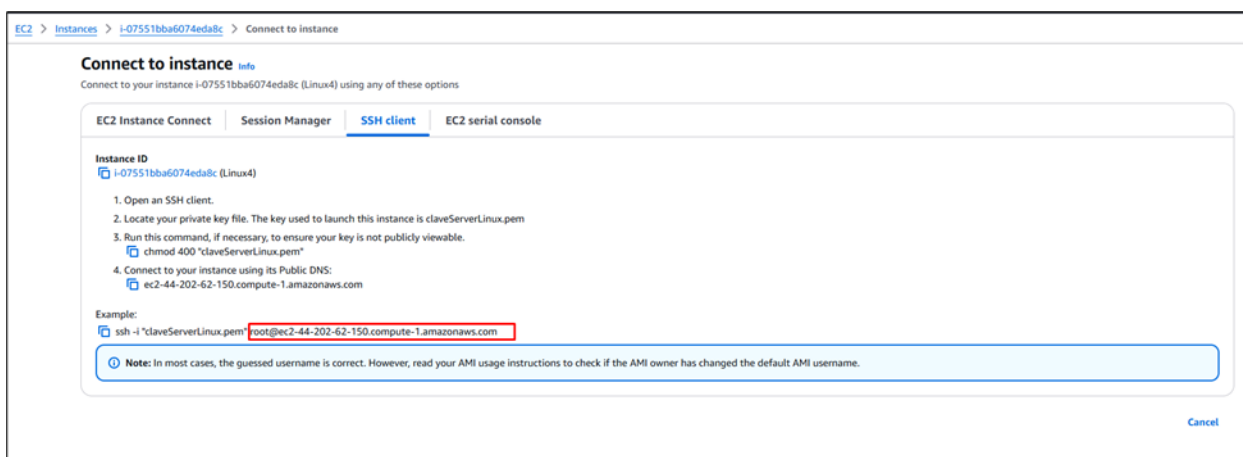
- Open an SSH client.
- Locate your private key file. The key used to launch this instance is `claveServerLinux.pem`
- Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 "claveServerLinux.pem"`
- Connect to your instance using its Public DNS:
`ec2-44-203-111-150.compute-1.amazonaws.com`

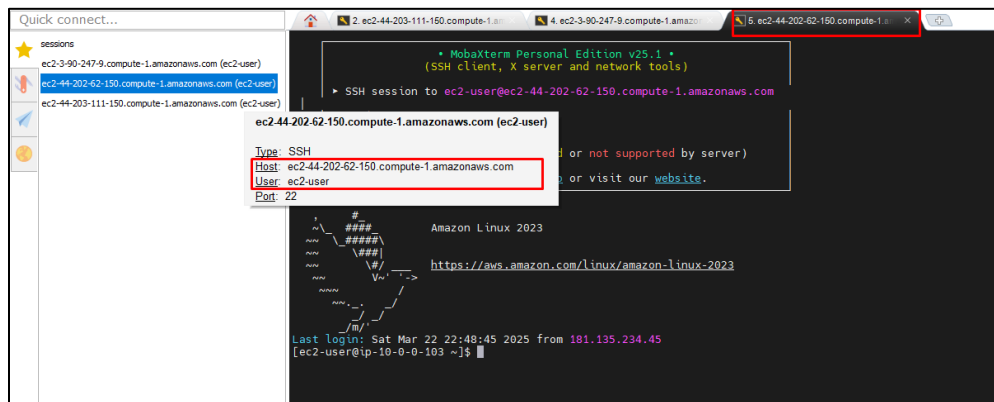
Example:
`ssh -i "claveServerLinux.pem" ec2-user@ec2-44-203-111-150.compute-1.amazonaws.com`

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.



Como la instancia Linux4 se creó por medio de una AMI y es una copia exacta de Linux1, su conexión tiene el usuario root asociado, este se reemplazó por el usuario correcto ec2-user pero su llave si fue la indicada en la conexión.





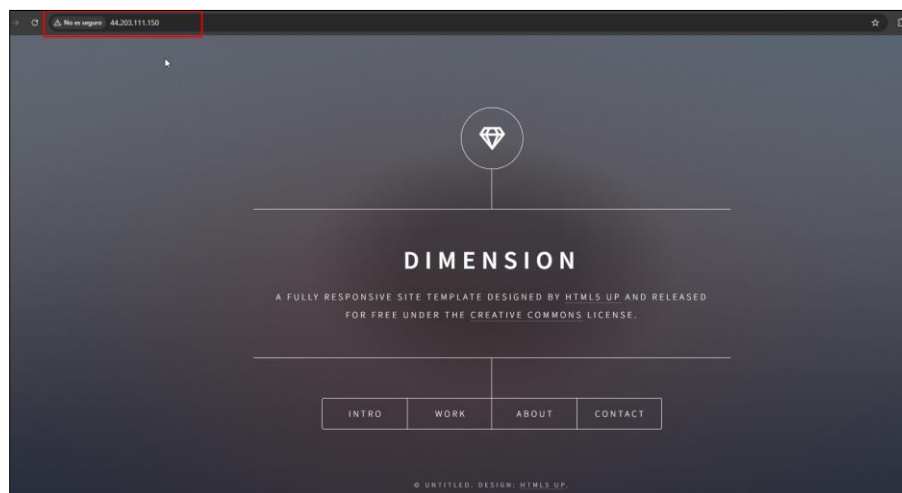
Para que cada sitio web se cargue automáticamente apenas se crea la instancia, en cada uno de los servidores se ejecutó el comando “**systemctl enable httpd**”

```
[root@ip-10-0-0-103 ec2-user]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-10-0-0-103 ec2-user]#
Broadcast message from root@ip-10-0-0-103.ec2.internal (Sun 2025-03-23 00:22:49 UT C):
The system will power off now!

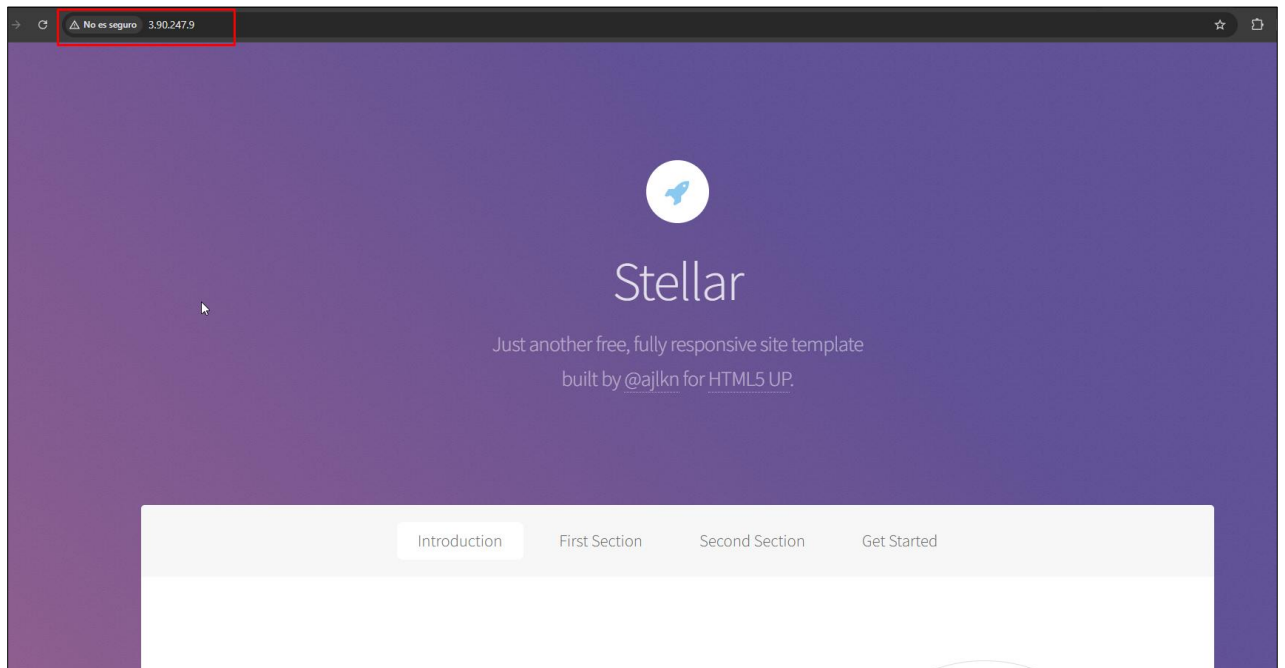
Broadcast message from root@ip-10-0-0-103.ec2.internal (Sun 2025-03-23 00:22:49 UT C):
The system will power off now!
```

Al ingresar a las instancias se puede evidenciar que cada una de ellas carga el sitio web correctamente

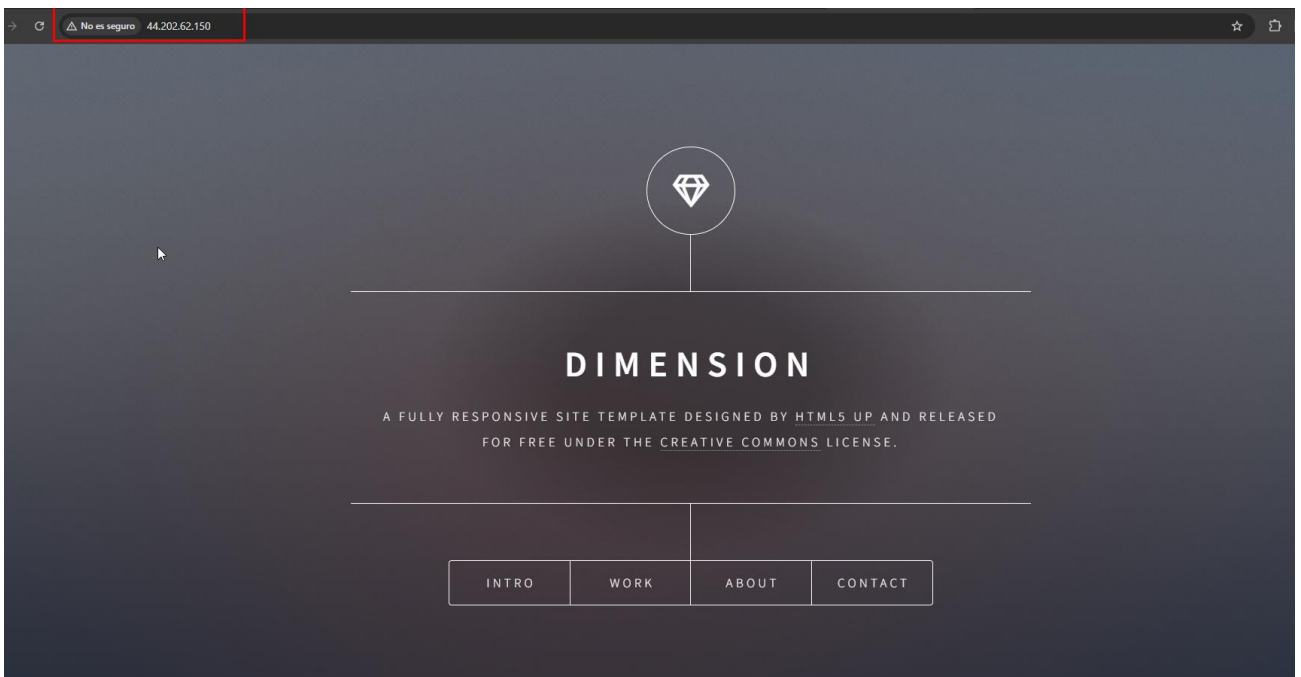
Instancia Linux1



Instancia Linux3



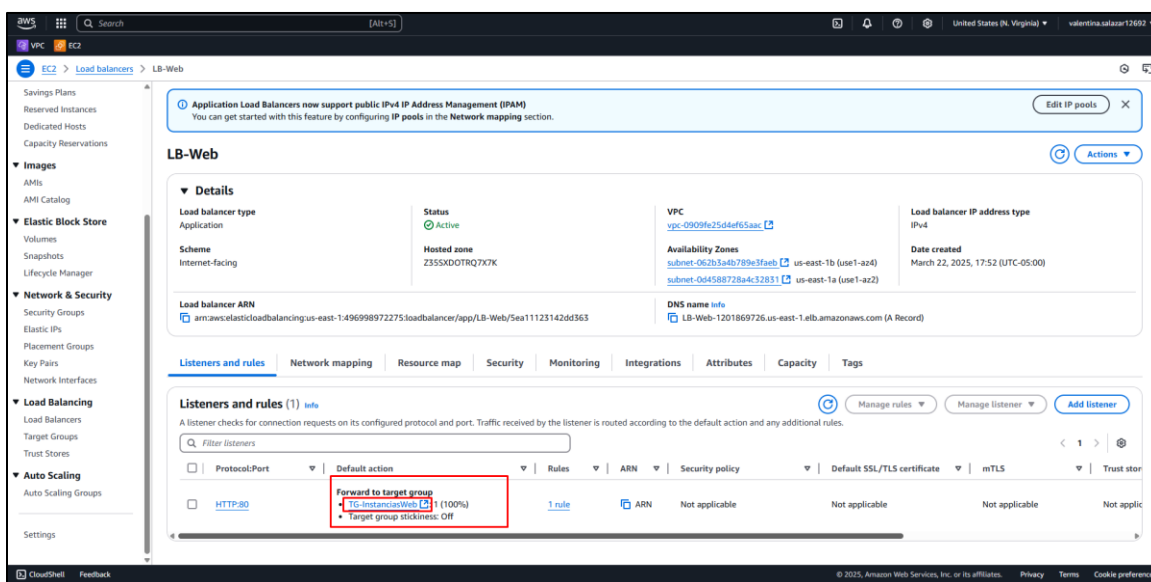
Instancia Linux4



BALANCEADOR DE CARGA

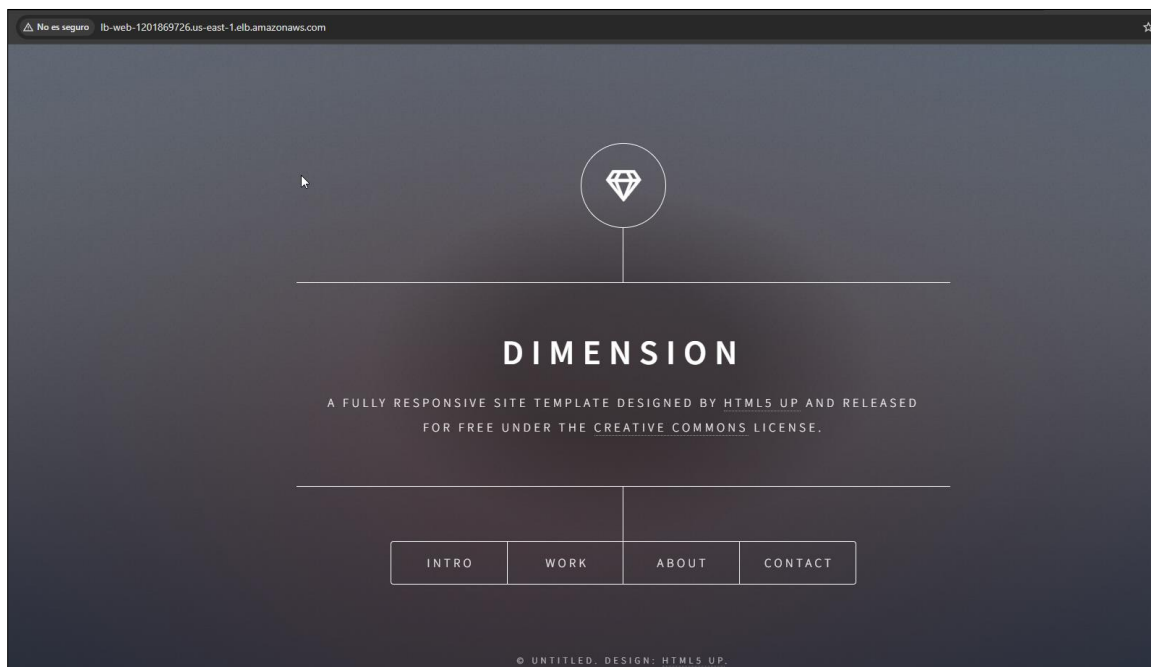
Se realizó un balanceador de carga automático que facilita la creación automática de las instancias, esto, en caso de que alguna llegase a detenerse o eliminarse por error AWS crea una nueva instancia en su lugar.

Para crear este balanceador, se creó una nueva AMI llamada “AmazonLinuxconApache2”. Este balanceador llamado LB-Web cuenta con un Target Group o grupo objetivo donde están las instancias que forman parte del balanceador.



Para validar que el balanceador funcione correctamente se tomó en DNS name y se pegó en el navegador validando que efectivamente se muestra el acceso al sitio web.

DNS name: LB-Web-1201869726.us-east-1.elb.amazonaws.com



Seguidamente, tenemos el HealCheck o Monitoreo de salud, donde podemos visualizar cuales son los servidores o instancias que están conectadas y operativas. En este punto, el sistema nos muestra que hay una instancia y que se encuentra funcionando correctamente. En caso de no ser así, la instancia aparecería en la opción “Unhealthy”

Registered targets (1)

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

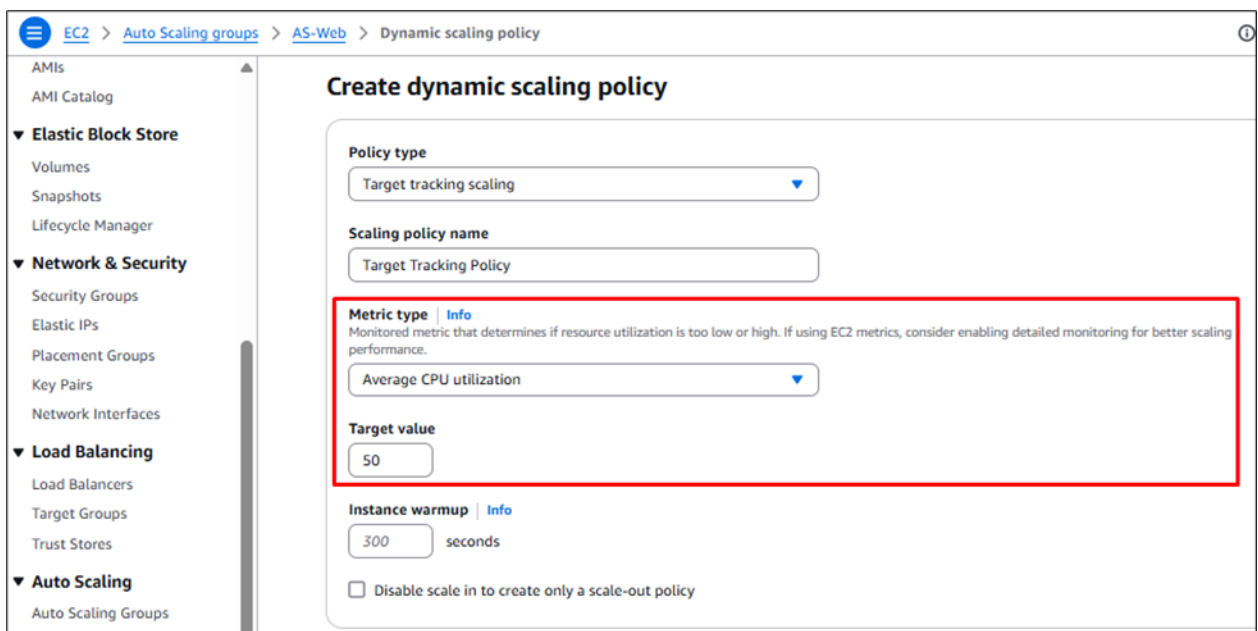
Instance ID	Name	Port	Zone	Health status	Health status details	Administrative o...	Override details	Launch...	Anomaly
i-0b4cd8c2106654ff4		80	us-east-1b (us...	Healthy		No override	No override is curren...	March 22...	Norma

Como siguiente paso, se realizó la creación del autoscaling llamado AS-Web, este autoscaling se creó por medio de un Launch Template (Plantilla de lanzamiento) que es el que tiene la configuración que el autoscaling utilizará para crear las nuevas instancias. En este autoscaling se definió una capacidad deseada siendo esta de una maquina o instancia, también se definió un límite mínimo y máximo de capacidad donde yo como usuario le indiqué al sistema que mínimo debe existir una instancia y máximo pueden crearse hasta 5.

The screenshot displays the AWS Management Console for the AS-Web Auto Scaling group. The 'AS-Web Capacity overview' section shows a desired capacity of 1 and scaling limits of 1 to 5. The 'Launch template' section provides details such as the AMI ID (ami-0c898d4c80966883), instance type (t2.micro), and security groups. The 'Network' section shows the availability zones (us-east-1a, us-east-1b) and subnets (subnet-015c3378f723ad79a, subnet-015c3378f723ad79a).

The 'Group size' dialog box allows users to specify the size of the Auto Scaling group. It includes a dropdown for 'Desired capacity type' set to 'Units (number of instances)'. The 'Desired capacity' is set to 1. The 'Scaling limits' section shows a 'Min desired capacity' of 1 and a 'Max desired capacity' of 5. The dialog also includes 'Cancel' and 'Update' buttons.

Finalmente, se definió una política de autoscaling donde se le indicó al sistema evaluar una capacidad de CPU del 50%. Esto significa que si la capacidad de CPU es superior al 50% el sistema empezará a crear instancias para satisfacer la demanda (puede crear hasta máximo 5, como se indicó anteriormente) pero que si por el contrario, existen varias instancias ya ejecutándose estas empezaran a disminuirse dejando la capacidad deseada en uno.



EC2 > Auto Scaling groups > AS-Web > Dynamic scaling policy

Create dynamic scaling policy

Policy type
Target tracking scaling

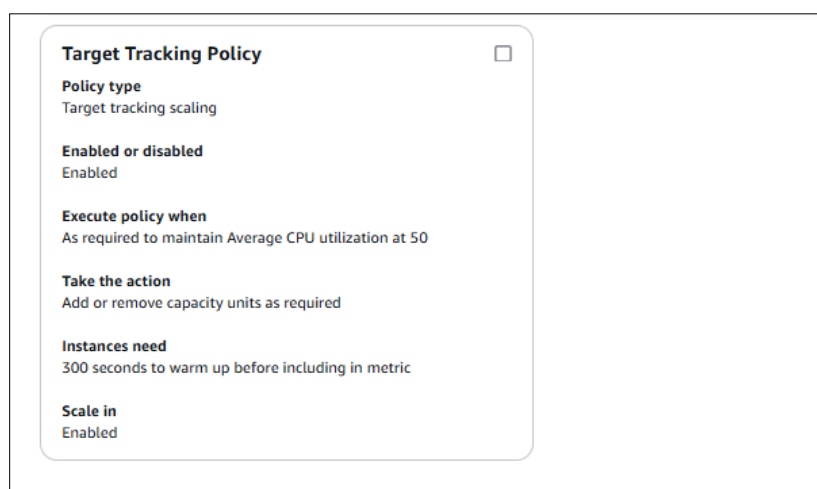
Scaling policy name
Target Tracking Policy

Metric type [Info](#)
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.
Average CPU utilization

Target value
50

Instance warmup [Info](#)
300 seconds

Disable scale in to create only a scale-out policy



Target Tracking Policy

Policy type
Target tracking scaling

Enabled or disabled
Enabled

Execute policy when
As required to maintain Average CPU utilization at 50

Take the action
Add or remove capacity units as required

Instances need
300 seconds to warm up before including in metric

Scale in
Enabled

Para validar que el autoscaling funcionara y la política fuera tomada de forma correcta, se realizó la conexión a la instancia creada con el balanceador la cual es aquella que cuenta con IP privada pues por seguridad, aquellas instancias que pertenezcan al balanceador no deben ser accesibles desde internet.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro		View alarms +	us-east-1a			
Linux4	i-07551bb6074eda8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	
Linux3	i-0bfb20d924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	
Linux1	i-02fd541ddc631021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	

Instance summary	
Instance ID	i-0b4cd8c2106654ff4
Public IPv4 address	-
Private IPv4 addresses	10.0.146.156

Para esto, utilizamos el programa Putty Key Generator (Puttygen), que nos permite realizar la conexión a la instancia Linux mediante la llave de acceso previamente creada. Este programa genera una clave llamada Linux.pem, la cual se sube a la instancia principal, es decir, la instancia Linux1, permitiendo así establecer la conexión por medio del comando

```
ssh -i "Linux.pem" ec2-user@10.0.146.156
```

EC2 > Instances > i-0b4cd8c2106654ff4 > Connect to instance

Connect to instance

Connect to your instance i-0b4cd8c2106654ff4 using any of these options

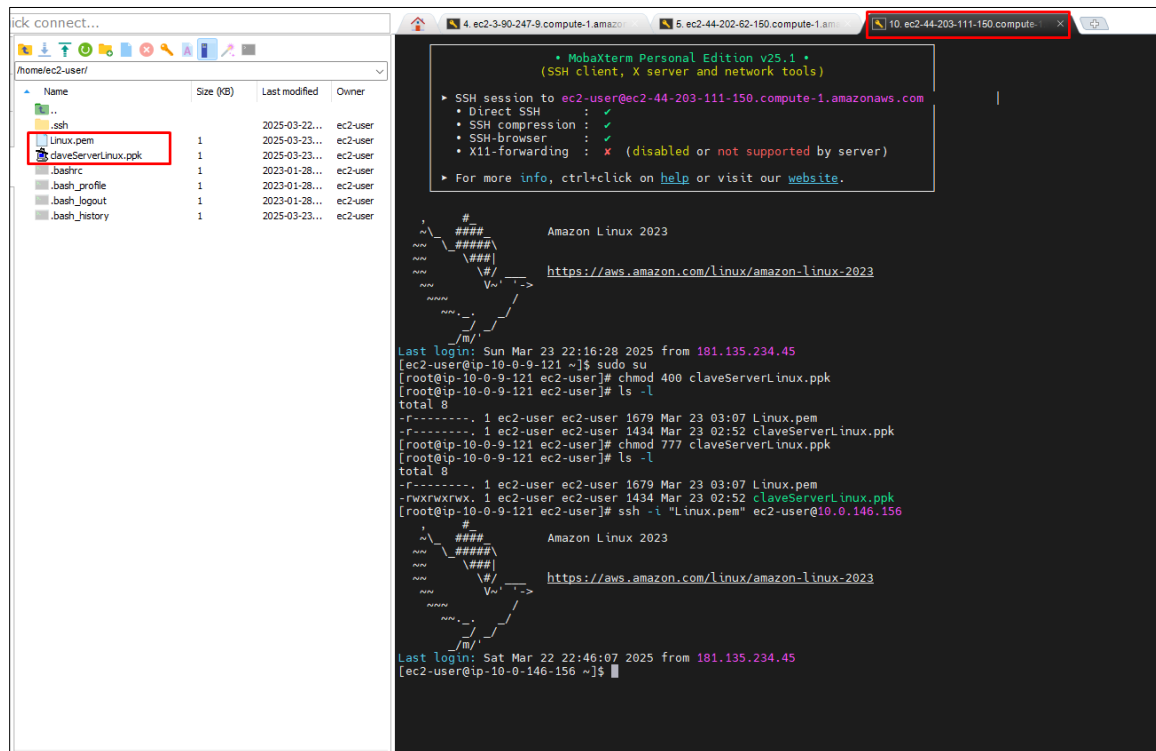
EC2 Instance Connect | Session Manager | **SSH client** | EC2 serial console

Instance ID
i-0b4cd8c2106654ff4

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `claveServerLinux.pem`
3. Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 "claveServerLinux.pem"`
4. Connect to your instance using its Private IP:
10.0.146.156

Example:
`ssh -i "claveServerLinux.pem" root@10.0.146.156`

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.



Seguidamente, se realizó la conexión a la instancia y se sobrecargó haciendo uso de un código que nos permitió validar el correcto funcionamiento del autoscaling ya que, de acuerdo con la política creada, si la CPU aumentaba o disminuía en un 50%, las instancias empezaban a lanzarse según la demanda.

El código utilizado para sobrecargar la maquina fue el siguiente:

```
while ;; do ;; done &
```

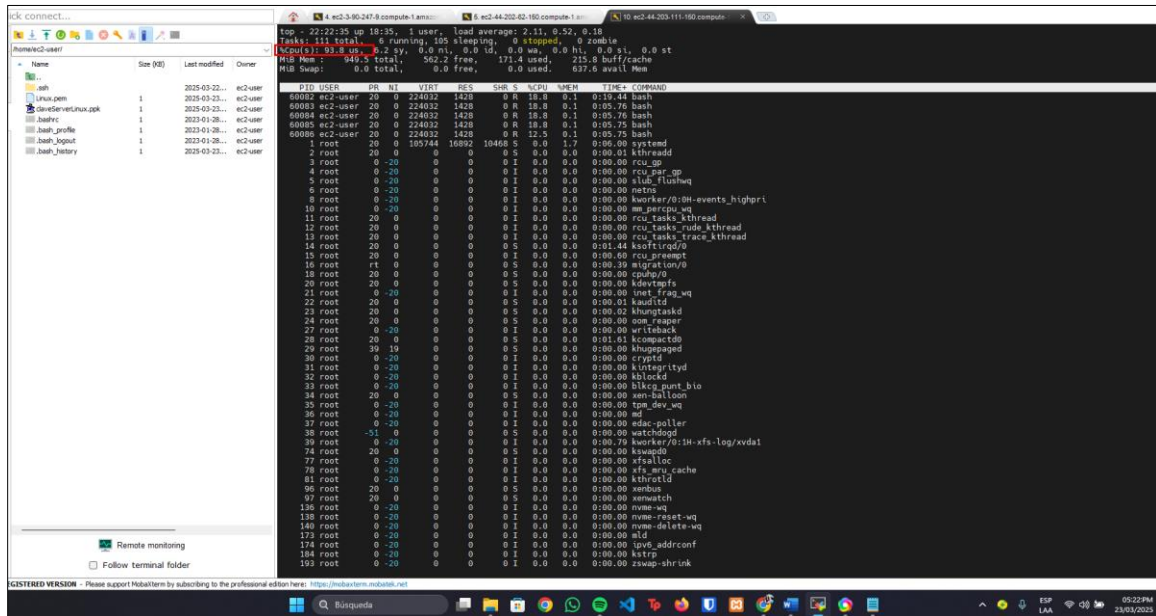
```
for i in {1..4}; do while ;; do ;; done & done
```

Uso de CPU antes de ejecutar el código

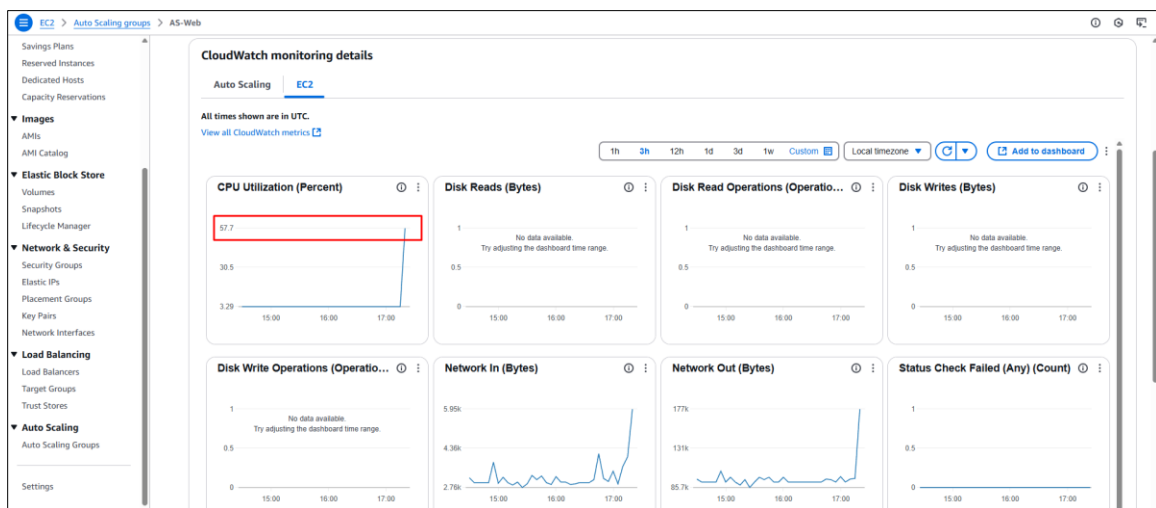
The screenshot shows a terminal window with a file explorer on the left and a terminal window on the right. The terminal window displays the output of the `top` command, showing system statistics and a list of processes. The system statistics at the top indicate: `top - 22:20:55 up 18:33, 1 user, load average: 0.00, 0.00, 0.00`. The `Mem` line shows: `Mem: 344.5 total, 562.2 free, 171.5 used, 215.8 buff/cache`. The `Swap` line shows: `Swap: 0.0 total, 0.0 free, 0.0 used, 427.0 avail Mem`. The process list shows various system processes, including `systemd`, `kernel`, `rcu_gp`, `rcu_par_gp`, `bluetooth`, `mdadm`, `khvmon`, `rcu_tasks_kthre`, `rcu_tasks_trace_kthre`, `rcu_tasks_rude_kthre`, `rcu_tasks_trace_kthre`, `cpupower`, `udevadm`, `unetd`, `khungtaskd`, `oom_reaper`, `writeback`, `kcompactd`, `khugepaged`, `cryptd`, `ntfsprogs`, `blkcg_punt_bio`, `xen-balloon`, `tpm_dev_wq`, `md`, `edac-poller`, `watchdog`, `khvmon/ih-kblockd`, `kswapd0`, `fsnotify`, `ifs_ercache`, `kernel`, `xenbus`, `xenwatch`, `nvmexpress-wq`, `nvmexpress-wq`, `mid`, `tpm2_addrconf`, `kstrp`, `zswap-shrink`, `khvmon/ubi0`, `ifs-buf/rxvd1`, `ifs-csv/rxvd1`, `ifs-reclaim/xvd`, and `ifs-blockgr/xvd`.

Uso de CPU luego de sobrecargar la instancia

The screenshot shows a terminal window with a file explorer on the left and a terminal window on the right. The terminal window displays the output of the `top` command, showing system statistics and a list of processes. The system statistics at the top indicate: `top - 05:22:24 up 18:33, 1 user, load average: 0.00, 0.00, 0.00`. The `Mem` line shows: `Mem: 344.5 total, 562.2 free, 171.5 used, 215.8 buff/cache`. The `Swap` line shows: `Swap: 0.0 total, 0.0 free, 0.0 used, 427.0 avail Mem`. The process list shows various system processes, including `systemd`, `kernel`, `rcu_gp`, `rcu_par_gp`, `bluetooth`, `mdadm`, `khvmon`, `rcu_tasks_kthre`, `rcu_tasks_trace_kthre`, `rcu_tasks_rude_kthre`, `rcu_tasks_trace_kthre`, `cpupower`, `udevadm`, `unetd`, `khungtaskd`, `oom_reaper`, `writeback`, `kcompactd`, `khugepaged`, `cryptd`, `ntfsprogs`, `blkcg_punt_bio`, `xen-balloon`, `tpm_dev_wq`, `md`, `edac-poller`, `watchdog`, `khvmon/ih-kblockd`, `kswapd0`, `fsnotify`, `ifs_ercache`, `kernel`, `xenbus`, `xenwatch`, `nvmexpress-wq`, `nvmexpress-wq`, `mid`, `tpm2_addrconf`, `kstrp`, `zswap-shrink`, `khvmon/ubi0`, `ifs-buf/rxvd1`, `ifs-csv/rxvd1`, `ifs-reclaim/xvd`, and `ifs-blockgr/xvd`. A red box highlights a terminal command: `[ec2-user@ip-10-0-146-156 ~]$ while :; do :; done &`

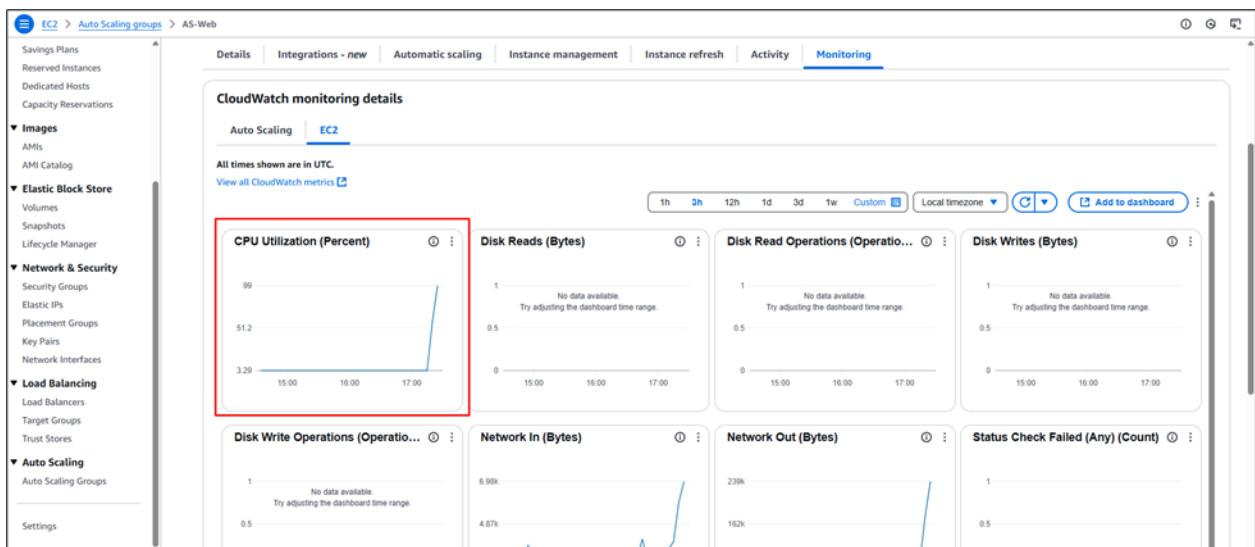


Luego de sobrecargar la instancia, nos dirigimos al autoscaling. Dado que el tiempo de actualización es de 5 minutos, después de algunos minutos, y una vez que el autoscaling detectó que el uso de la CPU había alcanzado el 57.7%, comenzó a crear una nueva instancia para ajustar la capacidad según la demanda.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro	–	View alarms +	us-east-1a	–	–	–
Linux4	i-07551bba6074eda8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	–
Linux3	i-0bd20d924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	–
Linux1	i-02f6541dd6c31021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	–
	i-0ed0629dcece0d1b4	Running	t2.micro	Initializing	View alarms +	us-east-1a	–	–	–

Finalmente, se concluyó que el autoscaling funcionó correctamente, ya que, al detectar un uso de CPU del 100%, las instancias se crearon automáticamente. Al disminuir la carga, el sistema ajustó la capacidad, manteniendo el número de instancias en 1, tal como estaba inicialmente.



EC2 > Auto Scaling groups > AS-Web

Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

▼ Images
AMIs
AMI Catalog

▼ Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

▼ Load Balancing
Load Balancers
Target Groups
Trust Stores

▼ Auto Scaling
Auto Scaling Groups

Settings

Activity history (20)

Filter activity history

Status	Description	Cause	Start time	End time
Waiting for instance warm up	Launching a new EC2 instance i-0ad14bb6ddad61dab	At 2025-03-23T22:31:53Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-53d5287a-74df-4ffba321-c0847e149101 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 4. At 2025-03-23T22:32:04Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 4.	2025 March 23, 05:32:06 PM -05:00	
Waiting for instance warm up	Launching a new EC2 instance i-0e839a9f3f8ba03c	At 2025-03-23T22:31:53Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-53d5287a-74df-4ffba321-c0847e149101 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2025-03-23T22:32:04Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 4.	2025 March 23, 05:32:06 PM -05:00	
Successful	Launching a new EC2 instance i-0ed0629dcece0d1b4	At 2025-03-23T22:25:53Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-53d5287a-74df-4ffba321-c0847e149101 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2025-03-23T22:26:03Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2025 March 23, 05:26:04 PM -05:00	2025 March 23, 05:31:10 PM -05:00
Successful	Terminating EC2 instance i-0a2ad146b547a5df	At 2025-03-23T03:55:10Z a monitor alarm TargetTracking-AS-Web-AlarmLow-60c770b7-5961-4c60-81b0-8f850b0a6d5f in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 1. At 2025-03-23T03:55:20Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-03-23T03:55:20Z instance i-0a2ad146b547a5df was selected for termination.	2025 March 22, 10:55:20 PM -05:00	2025 March 22, 11:01:44 PM -05:00
Successful	Terminating EC2 instance i-0c19f377119e81636	At 2025-03-23T03:48:23Z a user request update of AutoScalingGroup constraints to min: 1, max: 5, desired: 2 changing the desired capacity from 4 to 2. At 2025-03-23T03:48:29Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 4 to 2. At 2025-03-23T03:48:30Z instance i-0b8e3b0c810f66555 was selected for termination. At 2025-03-23T03:48:30Z instance i-0c19f377119e81636 was selected for termination.	2025 March 22, 10:48:30 PM -05:00	2025 March 22, 10:54:53 PM -05:00
Successful	Terminating EC2 instance i-0b8e3b0c810f66555	At 2025-03-23T03:48:23Z a user request update of AutoScalingGroup constraints to min: 1, max: 5, desired: 2 changing the desired capacity from 4 to 2. At 2025-03-23T03:48:29Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 4 to 2. At 2025-03-23T03:48:30Z instance i-0b8e3b0c810f66555 was selected for termination. At 2025-03-23T03:48:30Z instance i-0c19f377119e81636 was selected for termination.	2025 March 22, 10:48:30 PM -05:00	2025 March 22, 10:54:32 PM -05:00

EC2 > Instances

Dashboard
EC2 Global View
Events

▼ Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations

▼ Images
AMIs
AMI Catalog

▼ Elastic Block Store
Volumes
Snapshots
Lifecycle Manager

▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces

Instances (8)

Find Instance by attribute or tag (case-sensitive)

All states

Last updated less than a minute ago

Connect Instance state Actions Launch instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro	-	View alarms +	us-east-1a	-	-	-
Linux4	i-07551bba6074eda8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	-
Linux3	i-0e839a9f3f8ba03c	Running	t2.micro	Initializing	View alarms +	us-east-1a	-	-	-
Linux5	i-0b8b20a924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	-
	i-0ad14bb6ddad61dab	Running	t2.micro	Initializing	View alarms +	us-east-1b	-	-	-
	i-0b4cd8c2106654ff4	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-	-	-
Linux1	i-02f6541d6c631021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	-
	i-0ed0629dcece0d1b4	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-	-	-

Select an instance

Luego de terminar la instancia, se evidencia que la cantidad de instancias disminuyen al igual que la capacidad deseada en el Auto Scaling Group

The screenshot shows the AWS Management Console interface for EC2 instances. On the left, there is a navigation menu with categories like EC2, Images, Elastic Block Store, and Network & Security. The main area displays a list of instances under the heading 'Instances (1/8) info'. A table lists instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. The instance 'i-0b4cd8c2106654ff4' is selected and highlighted in blue. A context menu is open over this instance, showing options: 'Stop instance', 'Start instance', 'Reboot instance', 'Hibernate instance', and 'Terminate (delete) instance'. Below the table, the 'Instance summary' for 'i-0b4cd8c2106654ff4' is shown, including fields for Instance ID, IP address, Instance state (Running), and Hostname type. A red box highlights the 'Private IPv4 addresses' field, which shows '10.0.146.156'. Other fields include Public IPv4 address, Instance state, and Private IP DNS name.

The screenshot shows a terminal window with two panes. The top pane displays system statistics: 'top - 22:57:06 up 22:29, 2 users, load average: 0.00, 0.00, 0.00'. It shows 'Tasks: 368 total, 1 running, 167 sleeping, 0 stopped, 0 zombie' and 'Mem: 346.5 total, 375.5 free, 146.0 used, 227.4 buff/cache'. The bottom pane shows a process list with columns for PID, USER, PR, NI, VIRT, RES, SHR, S, %CPU, %MEM, TIME+, and COMMAND. The list includes various system processes such as 'top', 'systemd', 'kthreadd', 'rcu_gp', 'rcu_par_gp', 'slub_flushq', 'netns', 'kworker:0:0-events_highpri', 'mm_percpu_wq', 'rcu_tasks_kthread', 'rcu_tasks_rude_kthread', 'rcu_tasks_trace_kthread', 'rcu_preempt', 'migration/0', 'cpuhp/0', 'kdsystrms', 'inet_frag_wq', 'ksoftirqd', 'khungtaskd', 'oom_reaper', 'writeback', 'kcompactd0', 'khugepaged', 'cryptd', 'kintegrityd', 'blivet', 'blkcg_punt_bio', 'xen-ballloom', 'tpe_dev_wq', 'md', 'edac-poller', 'watchdog', 'kworker:0:0-iblock', 'ksm', 'xfsailoc', 'xfs_wq_cache', 'kthrotld', 'xenbus', 'xenwatch', 'nvm-wq', 'nvm-rset-wq', 'nvm-delete-wq', 'mld', 'ipwv_addrconf', and 'kstrp'.

EC2 > Target groups > TG-InstanciasWeb

Target type: Instance
 Protocol: Port HTTP: 80
 Protocol version: HTTP1
 VPC: vpc-0909fe25d4ef65aac

IP address type: IPv4
 Load balancer: LB-Web

4 Total targets
 3 Healthy
 0 Anomalous
 0 Unhealthy
 0 Unused
 0 Initial
 1 Draining

Distribution of targets by Availability Zone (AZ)

Registered targets (4) info

Instance ID	Name	Port	Zone	Health status	Health status details	Admin...	Overrid...	Launch ...	Anomaly detect
i-008f7e8882c59755f		80	us-east-1b (us...	Healthy	-	No override...	No overrid...	March 23, ...	Normal
i-0e839a9f5ff8ba03c		80	us-east-1a (us...	Healthy	-	No override...	No overrid...	March 23, ...	Normal
i-0ad1dbb6ddad61dad		80	us-east-1b (us...	Healthy	-	No override...	No overrid...	March 23, ...	Normal
i-0ed0629dcece0d1b4		80	us-east-1a (us...	Draining	Target deregistration l...	No override...	No overrid...	March 23, ...	Normal

EC2 > Auto Scaling groups > AS-Web

Monitoring

CloudWatch monitoring details

Auto Scaling | EC2

All times shown are in UTC.

CPU Utilization (Percent)

Disk Reads (Bytes)

Disk Read Operations (Operations)

Disk Writes (Bytes)

Disk Write Operations (Operations)

Network In (Bytes)

Network Out (Bytes)

Status Check Failed (Any) (Count)

Instances (9) info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro		View alarms +	us-east-1a	-	-	-
Linux4	i-07551bba6074edab8c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-202-62-150.co...	44.202.62.150	-
Linux3	i-0e859a9f3f8ba03c	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-	-	-
Linux3	i-0b0b20d924d513622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-3-90-247-9.comput...	3.90.247.9	-
	i-0ad10bb6dad61dab	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-	-	-
	i-0b4cc8c2106654ff4	Terminated	t2.micro	-	View alarms +	us-east-1b	-	-	-
	i-008f7e8882c59755f	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-	-	-
Linux1	i-02f541ddc631021d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-44-203-111-150.co...	44.203.111.150	-
	i-0ed0629dcece0d1b4	Terminated	t2.micro	-	View alarms +	us-east-1a	-	-	-

AS-Web

AS-Web Capacity overview

Desired capacity: 2

Scaling limits (Min - Max): 1 - 5

Desired capacity type: Units (number of instances)

Status: -

Date created: Sat Mar 22 2025 19:59:06 GMT-0500 (hora estándar de Colombia)

Launch template

Launch template: [lt-047ae8131145739d3](#)
PlantillaWebAutoScaling

AMI ID: [ami-0cc898d4c80966883](#)

Instance type: t2.micro

Owner: [arn:aws:iam::456998972275:root](#)

Version: Default

Security groups: [sg-00c8dce53a559e952](#)

Security group IDs: [sg-00c8dce53a559e952](#)

Create time: Sat Mar 22 2025 19:49:29 GMT-0500 (hora estándar de Colombia)

Description: View details in the launch template console

Storage (volumes): -

Key pair name: claveServerLinux

Request Spot Instances: No

Network

AS-Web

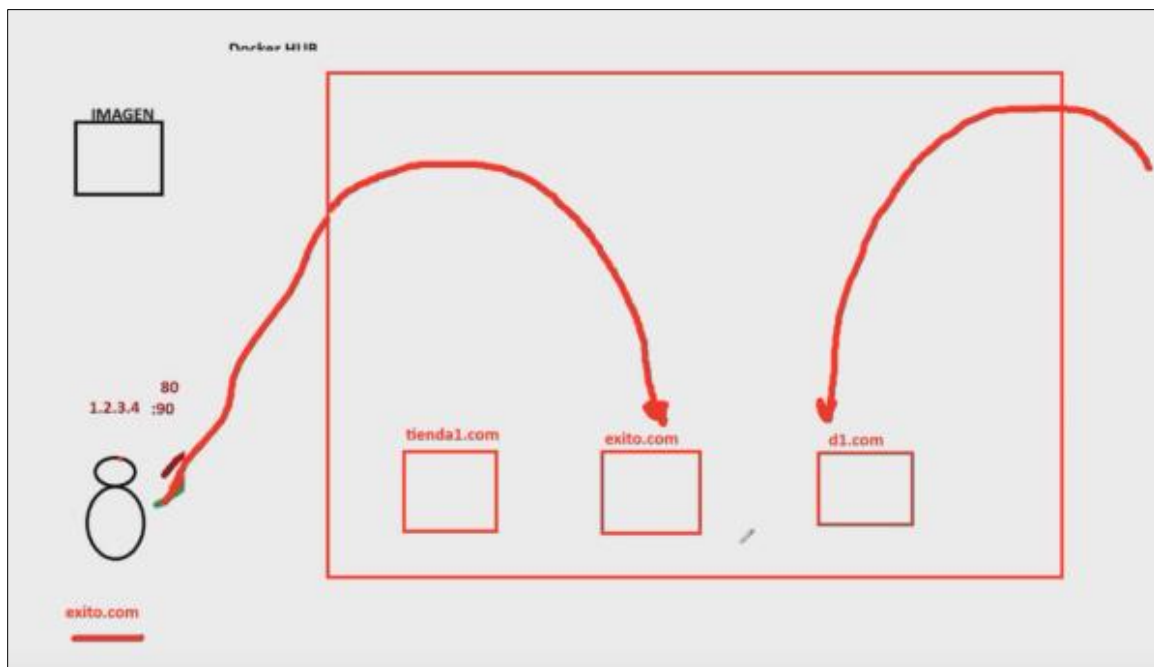
Filter activity history

Status	Description	Cause	Start time	End time
Connection draining in progress	Terminating EC2 instance: i-0ad10bb6dad61dab - Waiting For ELB Connection Draining.	At 2025-03-23T23:02:51Z a monitor alarm TargetTracking-AS-Web-AlarmLow-60c77047-5961-4c60-81b0-8f85d0ba0d5f in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 3 to 2. At 2025-03-23T23:02:59Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2025-03-23T23:02:59Z instance i-0ad10bb6dad61dab was selected for termination.	2025 March 23, 06:02:59 PM -0500	
Successful	Launching a new EC2 instance: i-008f7e8882c59755f	At 2025-03-23T22:58:49Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 March 23, 05:58:51 PM -0500	2025 March 23, 05:58:57 PM -0500
Successful	Terminating EC2 instance: i-0b4cc8c2106654ff4	At 2025-03-23T22:58:49Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2025 March 23, 05:58:49 PM -0500	2025 March 23, 06:03:51 PM -0500
Successful	Terminating EC2 instance: i-0ed0629dcece0d1b4	At 2025-03-23T22:56:27Z a monitor alarm TargetTracking-AS-Web-AlarmLow-60c77047-5961-4c60-81b0-8f85d0ba0d5f in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 4 to 3. At 2025-03-23T22:56:34Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 4 to 3. At 2025-03-23T22:56:34Z instance i-0ed0629dcece0d1b4 was selected for termination.	2025 March 23, 05:56:34 PM -0500	2025 March 23, 06:01:57 PM -0500
Successful	Launching a new EC2 instance: i-0ad10bb6dad61dab	At 2025-03-23T22:51:53Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-53d5287a-74df-4ffba321-c0847e149101 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 4. At 2025-03-23T22:32:04Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 4.	2025 March 23, 05:32:06 PM -0500	2025 March 23, 05:37:12 PM -0500
Successful	Launching a new EC2 instance: i-0e859a9f3f8ba03c	At 2025-03-23T22:31:53Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-53d5287a-74df-4ffba321-c0847e149101 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 4. At 2025-03-23T22:32:04Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 4.	2025 March 23, 05:32:06 PM -0500	2025 March 23, 05:37:12 PM -0500
Successful	Launching a new EC2 instance: i-0ed0629dcece0d1b4	At 2025-03-23T22:25:51Z a monitor alarm TargetTracking-AS-Web-AlarmHigh-53d5287a-74df-4ffba321-c0847e149101 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2025-03-23T22:26:02Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2025 March 23, 05:26:04 PM -0500	2025 March 23, 05:31:10 PM -0500

Entrega 2

Actividades Por Realizar 2.1.

Se requiere crear una instancia con múltiples contenedores que, al recibir una solicitud para un sitio web específico, redirija automáticamente la petición al contenedor correspondiente sin necesidad de especificar un puerto. El objetivo es que el usuario pueda acceder al sitio simplemente ingresando la URL en el navegador, sin incluir un número de puerto.



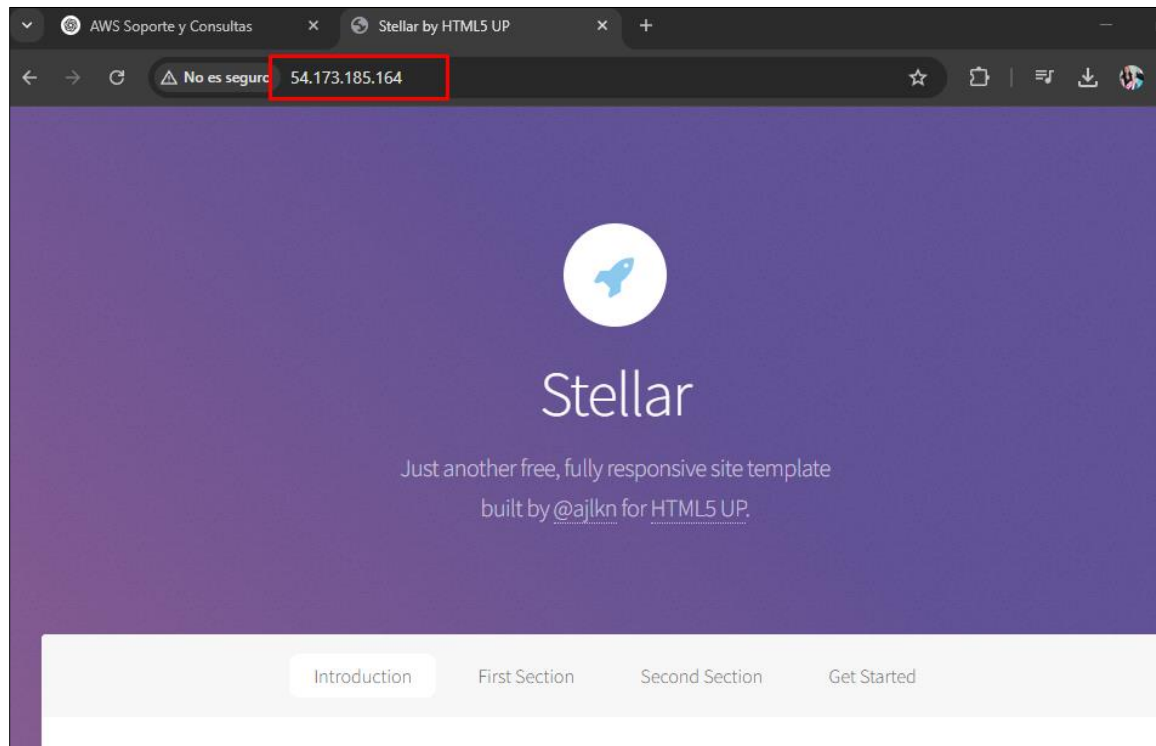
Desarrollo 2.2.

PARTE PRACTICA

Primeramente, se realizó la conexión con la instancia Linux ya existente. En este caso a través de SSH, realizó la conexión a Linux3 con MobaXterm.

The screenshot shows the AWS Management Console interface for connecting to an EC2 instance. The page title is "Connect to instance" and it provides instructions for connecting to instance `i-0bdb20d924d512622` (Linux3). The "SSH client" tab is selected, showing a list of steps: 1. Open an SSH client. 2. Locate your private key file. The key used to launch this instance is `claveServerLinux3.pem`. 3. Run this command, if necessary, to ensure your key is not publicly viewable. `chmod 400 "claveServerLinux3.pem"`. 4. Connect to your instance using its Public DNS: `ec2-54-173-185-164.compute-1.amazonaws.com`. A red box highlights the "Command copied" notification and the terminal command: `ssh -i "claveServerLinux3.pem" ec2-user@ec2-54-173-185-164.compute-1.amazonaws.com`. A note at the bottom states: "Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username."

The screenshot shows the "Session settings" dialog box with various connection options. The "SSH" option is selected. The "Basic SSH settings" tab is active, showing the "Remote host" field with the value `ec2-54-173-185-164.compu`, the "Specify username" checkbox checked, and the "Username" dropdown set to `ec2-user`. The "Port" is set to `22`. The "Advanced SSH settings" tab is also visible, showing options for "X11-Forwarding", "Compression", "Remote environment" (set to "Interactive shell"), "Execute command", "SSH-browser type" (set to "SFTP protocol"), "Use private key" (checked), and "Execute macro at session start" (set to "<none>"). A red box highlights the "Basic SSH settings" section. At the bottom, the "OK" button is highlighted with a red box.



Como el proyecto se realizó con contenedores, se procedió a desinstalar el apache.

```

root@ip-10-0-24-96 ec2-user# dnf remove httpd
Dependencies resolved.
Package Architecture Version Repository Size
Removing:
httpd x86_64 2.4.62-1.amzn2023 @amazonlinux 61 k
Removing unused dependencies:
apr x86_64 1.7.5-1.amzn2023.0.4 @amazonlinux 299 k
apr-util-openssl x86_64 1.6.3-1.amzn2023.0.1 @amazonlinux 217 k
generic-logos-httpd x86_64 18.0.0-12.amzn2023.0.3 @amazonlinux 24 k
httpd-core noarch 2.4.62-1.amzn2023 @amazonlinux 21 k
httpd-filesystem x86_64 2.4.62-1.amzn2023 @amazonlinux 4.7 M
httpd-tools noarch 2.4.62-1.amzn2023 @amazonlinux 464
libbrotli x86_64 2.4.62-1.amzn2023 @amazonlinux 281 k
mailcap noarch 2.1.49-3.amzn2023.0.3 @amazonlinux 771 k
mod_http2 x86_64 2.0.27-1.amzn2023.0.3 @amazonlinux 78 k
mod_lua x86_64 2.0.27-1.amzn2023.0.3 @amazonlinux 444 k
mod_lua x86_64 2.4.62-1.amzn2023 @amazonlinux 143 k

Transaction Summary
Remove 12 Packages

Freed space: 6.9 M
Is this ok [y/N]: y
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                : 1/12
  Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64 : 1/12
Removed "/etc/systemd/system/multi-user.target.wants/httpd.service".
Erasing                : httpd-2.4.62-1.amzn2023.x86_64 : 1/12
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64 : 1/12
Erasing                : mod_lua-2.4.62-1.amzn2023.x86_64 : 2/12
Erasing                : mod_http2-2.0.27-1.amzn2023.0.3.x86_64 : 3/12
Erasing                : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch : 4/12
Erasing                : httpd-core-2.4.62-1.amzn2023.0.3.noarch : 5/12
Erasing                : httpd-tools-2.4.62-1.amzn2023.x86_64 : 6/12
Erasing                : mailcap-2.1.49-3.amzn2023.0.3.noarch : 7/12
Erasing                : httpd-filesystem-2.4.62-1.amzn2023.noarch : 8/12
Erasing                : apr-util-1.6.3-1.amzn2023.0.1.x86_64 : 9/12
Erasing                : apr-1.7.5-1.amzn2023.0.4.x86_64 : 10/12
Erasing                : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 : 11/12
Erasing                : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 : 12/12
Running scriptlet: libbrotli-1.0.9-4.amzn2023.0.2.x86_64 : 12/12
Verifying              : apr-1.7.5-1.amzn2023.0.4.x86_64 : 1/12
Verifying              : apr-util-1.6.3-1.amzn2023.0.1.x86_64 : 2/12
Verifying              : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 : 3/12
Verifying              : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch : 4/12
Verifying              : httpd-2.4.62-1.amzn2023.x86_64 : 5/12
Verifying              : httpd-core-2.4.62-1.amzn2023.x86_64 : 6/12

```

Seguidamente se realizó la instalación del Docker

```

libbrotli-1.0.9-4.amzn2023.0.2.x86_64 mailcap-2.1.49-3.amzn2023.0.3.noarch mod_http2-2.0.27-1.amzn2023.0.3.x86_64 mod_lua-2.4.62-1.amzn2023.x86_64
Complete!
[root@ip-10.0.24.96 ec2-user]# yum install docker
Last metadata expiration check: 0:34:42 ago on Sat Mar 29 00:04:55 2025.
Dependencies resolved.
=====
Package                               Architecture      Version           Repository        Size
-----
Installing:
docker                                 x86_64            25.0.0-1.amzn2023.0.1  amazonlinux      44 M
Installing dependencies:
containerd                             x86_64            1.7.25-1.amzn2023.0.1  amazonlinux      36 M
iptables-libs                          x86_64            1.8.8-3.amzn2023.0.2  amazonlinux      481 k
iptables-nft                          x86_64            1.8.8-3.amzn2023.0.2  amazonlinux      183 k
libcgroup                              x86_64            3.0-1.amzn2023.0.1    amazonlinux       75 k
libnetfilter_conntrack                x86_64            1.0.8-2.amzn2023.0.2  amazonlinux      58 k
libnftnl                              x86_64            1.0.1-19.amzn2023.0.2  amazonlinux      30 k
libnftnl                              x86_64            1.2.2-2.amzn2023.0.2  amazonlinux      84 k
pigz                                   x86_64            2.5-1.amzn2023.0.3    amazonlinux       83 k
runc                                   x86_64            1.2.4-1.amzn2023.0.1  amazonlinux      3.4 M
Transaction Summary
-----
Install 10 Packages

Total download size: 84 M
Installed size: 319 M
Is this ok [y/N]: y
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm                6.8 MB/s | 401 kB  00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                4.2 MB/s | 183 kB  00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                    1.6 MB/s | 75 kB  00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm     1.8 MB/s | 58 kB  00:00
(5/10): libnftnl-1.0.1-19.amzn2023.0.2.x86_64.rpm                  1.1 MB/s | 30 kB  00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm                   2.2 MB/s | 84 kB  00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                          1.4 MB/s | 83 kB  00:00
(8/10): runc-1.2.4-1.amzn2023.0.1.x86_64.rpm                        10 MB/s | 3.4 MB  00:00
(9/10): containerd-1.7.25-1.amzn2023.0.1.x86_64.rpm                22 MB/s | 36 MB  00:01
(10/10): docker-25.0.0-1.amzn2023.0.1.x86_64.rpm                   21 MB/s | 44 MB  00:02
-----
Total                                                                39 MB/s | 84 MB  00:02
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                : runc-1.2.4-1.amzn2023.0.1.x86_64                                1/1
  Installing               : containerd-1.7.25-1.amzn2023.0.1.x86_64                       2/10
  Running scriptlet        : containerd-1.7.25-1.amzn2023.0.1.x86_64                       2/10
  Installing               : pigz-2.5-1.amzn2023.0.3.x86_64                             3/10
  Installing               : libnftnl-1.2.2-2.amzn2023.0.2.x86_64                       4/10
  Installing               : libnftnl-1.0.1-19.amzn2023.0.2.x86_64                     5/10
  Installing               : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64         6/10
  Installing               : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                 7/10
  Installing               : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                 8/10
  Running scriptlet        : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64                 8/10

```

A continuación, se realizó la actualización de los repositorios de Linux con el comando “yum update”

```

Verifying : containerd-1.7.25-1.amzn2023.0.1.x86_64 1/10
Verifying : docker-25.0.8-1.amzn2023.0.1.x86_64 2/10
Verifying : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 3/10
Verifying : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64 4/10
Verifying : libcgrou-3.0-1.amzn2023.0.1.x86_64 5/10
Verifying : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 6/10
Verifying : libnftnl-1.0.1-19.amzn2023.0.2.x86_64 7/10
Verifying : libnftnl-1.2.2-2.amzn2023.0.2.x86_64 8/10
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64 9/10
Verifying : runc-1.2.4-1.amzn2023.0.1.x86_64 10/10

=====
WARNING:
A newer release of "Amazon Linux" is available.

Available Versions:

Version 2023.6.20250317:
Run the following command to upgrade to 2023.6.20250317:

dnf upgrade --releasever=2023.6.20250317

Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.6.20250317.html

=====
Installed:
containerd-1.7.25-1.amzn2023.0.1.x86_64 docker-25.0.8-1.amzn2023.0.1.x86_64 iptables-libs-1.8.8-3.amzn2023.0.2.x86_64 iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgrou-3.0-1.amzn2023.0.1.x86_64 libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 libnftnl-1.0.1-19.amzn2023.0.2.x86_64 libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64 runc-1.2.4-1.amzn2023.0.1.x86_64

Complete!
[root@ip-10-0-24-96 ec2-user]# yum install docker.io
Last metadata expiration check: 0:36:34 ago on Sat Mar 29 00:04:55 2025.
No match for argument: docker.io
Error: Unable to find a match: docker.io
[root@ip-10-0-24-96 ec2-user]# yum update
Last metadata expiration check: 0:39:12 ago on Sat Mar 29 00:04:55 2025.

=====
WARNING:
A newer release of "Amazon Linux" is available.

Available Versions:

Version 2023.6.20250317:
Run the following command to upgrade to 2023.6.20250317:

dnf upgrade --releasever=2023.6.20250317

Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.6.20250317.html

=====
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-24-96 ec2-user]#

```

El siguiente paso, fue ejecutar la aplicación para verificar su correcto funcionamiento. Sin embargo, se encontró inactivo por lo que se procedió a iniciarlo con el comando “systemctl start docker” y seguidamente se ejecutó el comando “systemctl enable docker” para que la maquina inicie sola si por algún motivo llegase a detenerse.

```

Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-24-96 ec2-user]# systemctl status docker
o docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
   Active: inactive (dead)
 TriggeredBy: o docker.socket
   Docs: https://docs.docker.com
[root@ip-10-0-24-96 ec2-user]#

```

```
[root@ip-10-0-24-96 ec2-user]# systemctl start docker
[root@ip-10-0-24-96 ec2-user]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
   Active: active (running) since Sat 2025-03-29 00:59:23 UTC; 4s ago
     TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Process: 6355 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
    Process: 6359 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
    Main PID: 6364 (dockerd)
      Tasks: 7
     Memory: 30.3M
        CPU: 270ms
     CGroup: /system.slice/docker.service
            └─6364 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Mar 29 00:59:22 ip-10-0-24-96.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Mar 29 00:59:22 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:22.995166657Z" level=info msg="Starting up"
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.050176798Z" level=info msg="Loading containers: start."
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.504759141Z" level=info msg="Loading containers: done."
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.528921168Z" level=info msg="Docker daemon" commit=71907ca containerd-snapshotter=fal
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.528921168Z" level=info msg="Daemon has completed initialization"
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.571284260Z" level=info msg="API listen on /run/docker.sock"
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal systemd[1]: Started docker.service - Docker Application Container Engine.
[lines 1-22/22 (END)]
```

```
Mar 29 00:59:22 ip-10-0-24-96.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Mar 29 00:59:22 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:22.995166657Z" level=info msg="Starting up"
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.050176798Z" level=info msg="Loading containers: start."
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.504759141Z" level=info msg="Loading containers: done."
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.528921168Z" level=info msg="Docker daemon" commit=71907ca containerd-snapshotter=fal
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.528921168Z" level=info msg="Daemon has completed initialization"
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal dockerd[6364]: time="2025-03-29T00:59:23.571284260Z" level=info msg="API listen on /run/docker.sock"
Mar 29 00:59:23 ip-10-0-24-96.ec2.internal systemd[1]: Started docker.service - Docker Application Container Engine.
[lines 1-22/22 (END)]
```

```
[root@ip-10-0-24-96 ec2-user]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-10-0-24-96 ec2-user]#
```

Ingresamos al docker hub (una plataforma que posee imágenes de aplicaciones configuradas. Desde allí descargamos la imagen httpd con el comando mencionado en el sitio web. Seguidamente, con el comando “docker images” validamos la presencia de esa imagen en el docker.

```
Run 'docker image COMMAND --help' for more information on a command.
[root@ip-10-0-24-96 ec2-user]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
[root@ip-10-0-24-96 ec2-user]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
6e909acdb790: Extracting [-----] 23.3MB/28.2MB
9f9b03a66afb: Download complete
4f4fb70ef54: Download complete
09bf08b13dbd: Download complete
064c58079b9a: Download complete
c61868f6ad74: Download complete
```

The screenshot shows the Docker Hub interface for the 'httpd' image. At the top, there is a search bar with 'httpd' entered. Below the search bar, the image name 'httpd' is displayed along with 'Docker Official Image' and download statistics. A red box highlights the 'docker pull httpd' command. To the right of the command is a 'Copy' button. Below the command, there is a 'Quick reference' section with links to documentation and a 'Recent tags' section listing various image versions.

```
[root@ip-10-0-24-96 ec2-user]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
httpd latest 83d938198316 2 months ago 148MB
[root@ip-10-0-24-96 ec2-user]#
```

Utilizamos el comando “docker run -dit --name apachev1 -p 8080:80 httpd” para crear el primer contenedor. En este comando, indicamos que el contenedor se llamará apachev1, asignamos el puerto 80:80 de la máquina virtual al puerto 80 del contenedor y usaremos la imagen httpd. Con el comando “docker ps”, validamos que el contenedor se esté ejecutando.

```
[root@ip-10-0-24-96 ec2-user]# docker run -dit --name apachev1 -p 8080:80 httpd
bae89e19cc86f2f9b45cfffaf3fdde924894501f1f75c5d3c1041905dc32fc4e7
[root@ip-10-0-24-96 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
bae89e19cc86   httpd    "httpd-foreground"      14 seconds ago Up 13 seconds  0.0.0.0:8080->80/tcp, :::8080->80/tcp  apachev1
[root@ip-10-0-24-96 ec2-user]#
```

Para verificar que la instancia responda correctamente cuando se acceda la IP en el puerto desde el navegador, verificamos que dicho puerto (8080) estuviera abierto en el firewall de la instancia, específicamente debe estar configurado en el Security Group. Dado que no estaba habilitado, realizamos las configuraciones necesarias y luego probamos el acceso ingresando la IP seguida del puerto 8080 en el navegador, es decir 54.173.185.164:8080 y este nos devolvió en el navegador un “It Works!”

The screenshot shows the 'Security' tab in the AWS IAM console for an instance. The 'Inbound rules' section is highlighted with a red box and contains the following table:

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0763a431fd819075d	80	TCP	0.0.0.0/0	SG-ServerLinux3	http
-	sgr-054bb264a361d1987	22	TCP	0.0.0.0/0	SG-ServerLinux3	-

Below the inbound rules, the 'Outbound rules' section is visible, showing one rule:

Name	Security group rule ID	Port range	Protocol	Destination	Security groups	Description
-	sgr-0ad80db3060effa94	All	All	0.0.0.0/0	SG-ServerLinux3	-

EC2 > Security Groups > sg-0ed76205de8f3aa0c - SG-ServerLinux3 > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Actions
sgr-0763a431fd819075d	HTTP	TCP	80	Custom	http	Delete
sgr-054bb264a361d1987	SSH	TCP	22	Custom		Delete
-	Custom TCP	TCP	8080	Anywhere	contenedor1	Delete

[Add rule](#)

Rules with source of 0.0.0.0 or :/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Preview changes](#) [Save rules](#)

Instances (1/4)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Server1	i-0e63b71deef524dee	Stopped	t2.micro	-	View alarms +	us-east-1a	-	-	-
Linux4	i-07551bba6074eda8c	Stopped	t2.micro	-	View alarms +	us-east-1a	-	-	-
Linux3	i-0bdb20d924d512622	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-54-173-185-164.co...	54.173.185.164	-

i-0bdb20d924d512622 (Linux3)

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sg-0763a431fd819075d	80	TCP	0.0.0.0/0	SG-ServerLinux3	http
-	sg-054bb264a361d1987	22	TCP	0.0.0.0/0	SG-ServerLinux3	-
-	sg-0bf5b8070ac7f6f43	8080	TCP	0.0.0.0/0	SG-ServerLinux3	contenedor1

← → ↻ No es seguro 54.173.185.164:8080

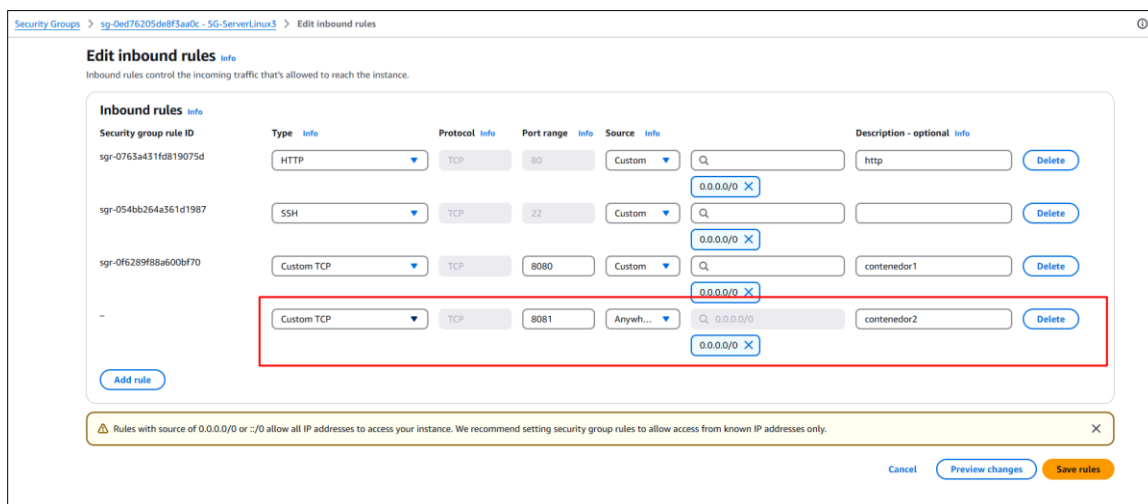
It works!

Se realizó la creación de una carpeta con un index.html con el fin de que el contenedor no cargue el sitio “It Works” sino que cargue uno propio donde se diligenció el mensaje “Hola a todos este es el contenedor 1” y se le otorgaron todos los permisos.

```
[root@ip-10-0-24-96 ec2-user]# mkdir app1
[root@ip-10-0-24-96 ec2-user]# cd app1/
[root@ip-10-0-24-96 app1]# nano index.html
[root@ip-10-0-24-96 app1]# cd ..
[root@ip-10-0-24-96 ec2-user]# chmod -R 777 app1/
[root@ip-10-0-24-96 ec2-user]#
```

Luego de esto se realizó la creación de un segundo contenedor con el comando “docker run -dit --name app1 -p 8081:80 -v /home/ec2-user/app1:/usr/local/apache2/htdocs/ httpd” donde al final se configuró un enlace simbólico que lo que hace es indicar una ruta, pero realmente se dirige a otro lugar, en este caso es al archivo index.html.

Como este contenedor tiene un nuevo puerto (8081) este al igual que el anterior se configura en el Security Group.



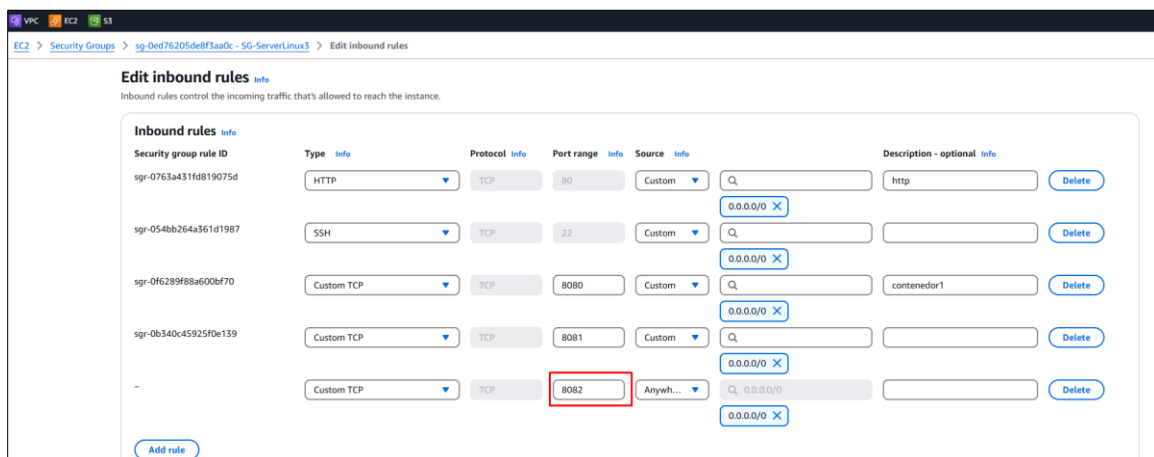
Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Action	
sgr-0763a431f4819075d	HTTP	TCP	80	Custom	Q, 0.0.0.0	http	Delete
sgr-054bb264a361d1987	SSH	TCP	22	Custom	Q, 0.0.0.0		Delete
sgr-0f6289f8a600bf70	Custom TCP	TCP	8080	Custom	Q, 0.0.0.0	contenedor1	Delete
-	Custom TCP	TCP	8081	Anywh...	Q, 0.0.0.0	contenedor2	Delete

Rules with source of 0.0.0.0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules



Se realizó la creación de un tercer contenedor con el comando “docker run -dit --name app2 -p 8082:80 -v /home/ec2-user/app2/:/usr/local/apache2/htdocs/ httpd” donde al final se configuró un enlace simbólico como el anterior que se dirige a un archivo index.html. Como este contenedor tiene un nuevo puerto (8082) nuevamente, se configura en el Security Group.



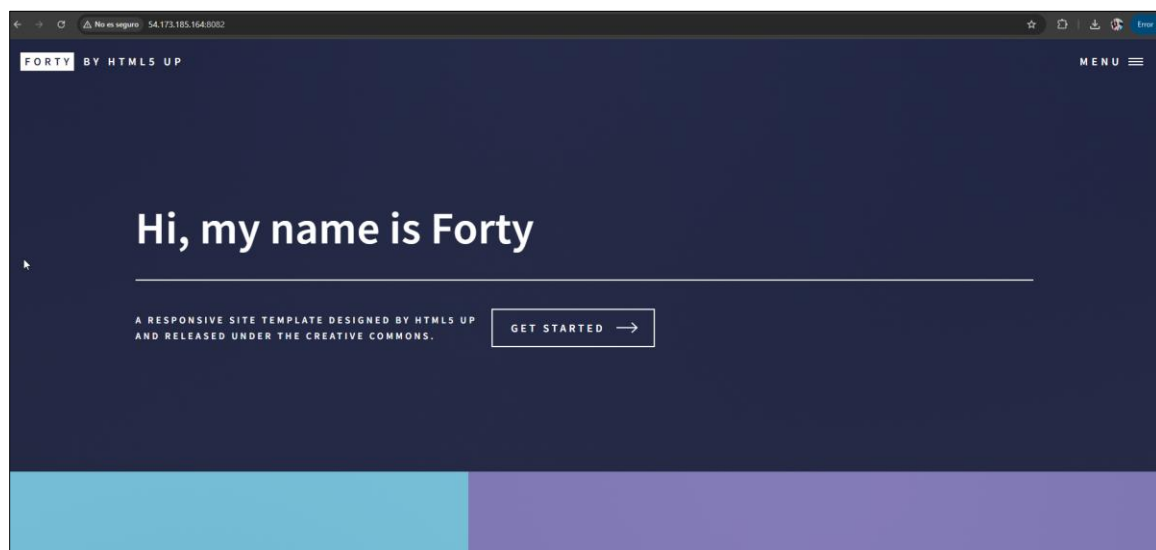
En la carpeta app2 se descargó una plantilla que se mostrará en el puerto 8082. Al acceder a la URL `http://54.173.185.164:8082/`, el sitio se visualizó correctamente.

```
bash: mk: command not found
[root@ip-10-0-24-96 ec2-user]# mkdir app2
[root@ip-10-0-24-96 ec2-user]# cd app2
[root@ip-10-0-24-96 app2]# wget https://html5up.net/forty/download
--2025-03-29 02:43:54-- https://html5up.net/forty/download
Resolving html5up.net (html5up.net)... 104.21.76.136, 172.67.195.190, 2606:4700:3030::6815:4c88, ...
Connecting to html5up.net (html5up.net)|104.21.76.136|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-zip]
Saving to: 'download'

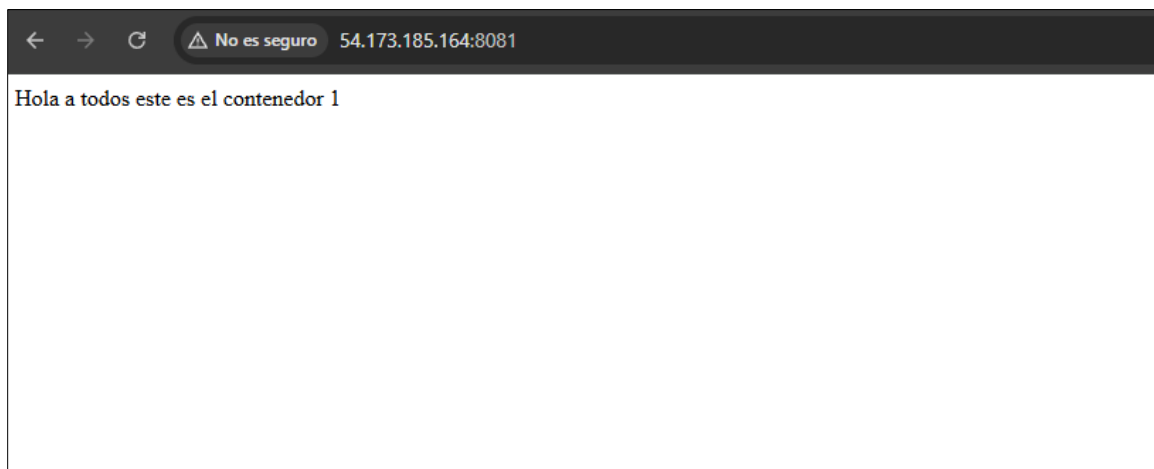
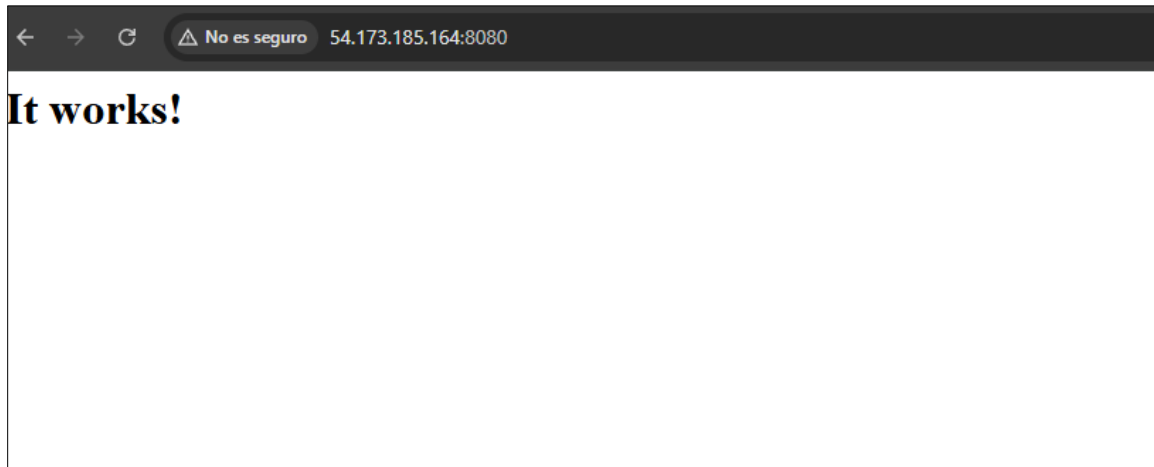
download [ <=> ]
2025-03-29 02:43:54 (32.7 MB/s) - 'download' saved [2236776]

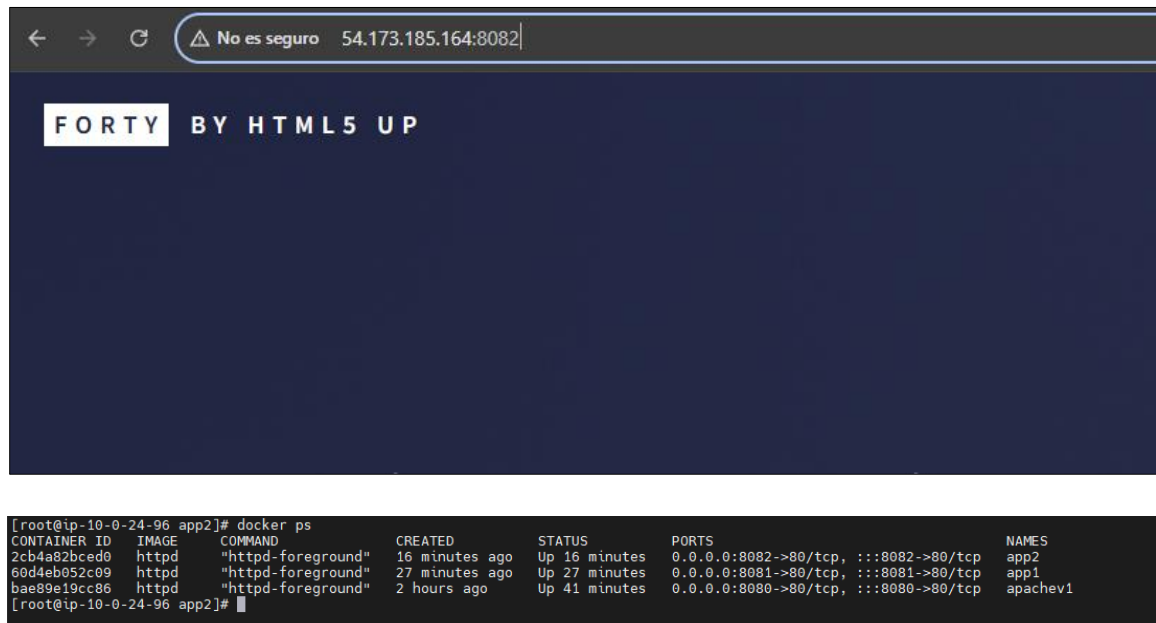
[root@ip-10-0-24-96 app2]# unzip download
Archive:  download
  inflating: LICENSE.txt
   creating: assets/
   creating: assets/webfonts/
  inflating: assets/webfonts/fa-regular-400.woff
  inflating: assets/webfonts/fa-regular-400.eot
  inflating: assets/webfonts/fa-brands-400.eot
  inflating: assets/webfonts/fa-solid-900.svg
  inflating: assets/webfonts/fa-regular-400.ttf
  inflating: assets/webfonts/fa-brands-400.woff2
  inflating: assets/webfonts/fa-regular-400.svg
  inflating: assets/webfonts/fa-solid-900.ttf
  inflating: assets/webfonts/fa-brands-400.woff
  inflating: assets/webfonts/fa-solid-900.woff2
  inflating: assets/webfonts/fa-brands-400.svg
  inflating: assets/webfonts/fa-solid-900.eot
  inflating: assets/webfonts/fa-brands-400.ttf
  extracting: assets/webfonts/fa-regular-400.woff2
  inflating: assets/webfonts/fa-solid-900.woff
   creating: assets/js/
  inflating: assets/js/breakpoints.min.js
  inflating: assets/js/jquery.scrollx.min.js
  inflating: assets/js/jquery.min.js
  inflating: assets/js/util.js
  inflating: assets/js/main.js
  inflating: assets/js/browser.min.js
  inflating: assets/js/jquery.scrolly.min.js
   creating: assets/css/
  inflating: assets/css/noscript.css
  inflating: assets/css/fontawesome-all.min.css
  inflating: assets/css/main.css
   creating: assets/sass/
   creating: assets/sass/base/
  inflating: assets/sass/base/_page.scss
  inflating: assets/sass/base/_typography.scss
  inflating: assets/sass/base/_reset.scss
   creating: assets/sass/layout/
  inflating: assets/sass/layout/_wrapper.scss
  inflating: assets/sass/layout/_menu.scss
  inflating: assets/sass/layout/_contact.scss
  inflating: assets/sass/layout/_banner.scss
  inflating: assets/sass/layout/_header.scss
```

hal edition here: <https://mobaxterm.mobatek.net>



Seguidamente se validó que los 3 contenedores funcionaran correctamente con los puertos correspondientes: <http://54.173.185.164:8080/> , <http://54.173.185.164:8081/> y <http://54.173.185.164:8082/> y llevaran a su respectivo sitio web. Para este caso contamos con 3 contenedores en una misma instancia (Linux3)





The image shows a web browser window at the top with the address bar displaying 'No es seguro 54.173.185.164:8082'. The page content is a dark blue background with the text 'FORTY BY HTML5 UP' in white. Below the browser is a terminal window showing the output of the 'docker ps' command. The terminal output is as follows:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2cb4a82bcd0	httpd	"httpd-foreground"	16 minutes ago	Up 16 minutes	0.0.0.0:8082->80/tcp, ::8082->80/tcp	app2
60d4eb052c09	httpd	"httpd-foreground"	27 minutes ago	Up 27 minutes	0.0.0.0:8081->80/tcp, ::8081->80/tcp	app1
bae89e19cc86	httpd	"httpd-foreground"	2 hours ago	Up 41 minutes	0.0.0.0:8080->80/tcp, ::8080->80/tcp	apachev1

Se instaló un servicio de balanceador de carga que distribuye las solicitudes entre los contenedores (contenedor1, contenedor2 y contenedor3). Esto es conocido como proxy reverse y permite que cuando una petición llega a la instancia, esta se dirija al proxy, el cual la envía a los contenedores disponibles. Los contenedores devuelven el contenido solicitado y el proxy lo devuelve al usuario final.

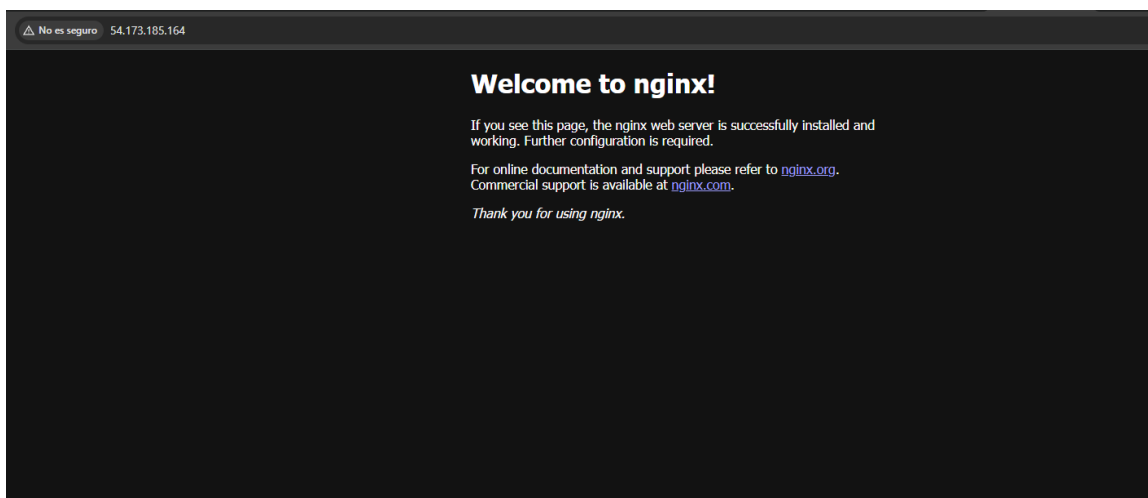
La aplicación de proxy se realizó por medio de un servidor web llamado nginx que fue instalado por medio del comando “dnf install nginx” finalmente se inició con el comando “systemctl start nginx” y se procedió a validar correcto ingreso en el puerto 80.

```

bae89e19cc86 httpd "httpd-foreground" 2 hours ago Up 41 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp apache1
[root@ip-10-0-24-96 app]# dnf install nginx
Last metadata expiration check: 3:24:18 ago on Sat Mar 29 00:04:55 2025.
Dependencies resolved.
-----
Package                               Architecture      Version           Repository        Size
-----
Installing:
nginx                                  x86_64            1:1.26.3-1.amzn2023.0.1  amazonlinux      33 k
Installing dependencies:
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch  noarch            18.0.0-12.amzn2023.0.3  amazonlinux      19 k
gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm    x86_64            2.9.1-1.amzn2023.0.3    amazonlinux      308 k
libunwind-1.4.0-5.amzn2023.0.2.x86_64              x86_64            1.4.0-5.amzn2023.0.2    amazonlinux      66 k
nginx-core-1.26.3-1.amzn2023.0.1.x86_64           x86_64            1:1.26.3-1.amzn2023.0.1  amazonlinux      670 k
nginx-filesystem-1.26.3-1.amzn2023.0.1.noarch      noarch            1:1.26.3-1.amzn2023.0.1  amazonlinux      9.6 k
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch       noarch            2.1.49-3.amzn2023.0.3    amazonlinux      21 k
-----
Transaction Summary
-----
Install 7 Packages

Total download size: 1.1 M
Installed size: 3.6 M
Is this ok [y/N]: y
Downloading Packages:
(1/7): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm 371 kB/s | 19 kB 00:00
(2/7): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rpm           1.2 MB/s | 66 kB 00:00
(3/7): gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64.rpm     4.3 MB/s | 308 kB 00:00
(4/7): nginx-1.26.3-1.amzn2023.0.1.x86_64.rpm             1.5 MB/s | 33 kB 00:00
(5/7): nginx-core-1.26.3-1.amzn2023.0.1.x86_64.rpm         16 MB/s | 670 kB 00:00
(6/7): nginx-filesystem-1.26.3-1.amzn2023.0.1.noarch.rpm   341 kB/s | 9.6 kB 00:00
(7/7): nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch.rpm    734 kB/s | 21 kB 00:00
-----
Total                                                                    7.9 MB/s | 1.1 MB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Running scriptlet:
Preparing                                                                    1/1
Running scriptlet: nginx-filesystem-1:1.26.3-1.amzn2023.0.1.noarch 1/7
Installing : nginx-filesystem-1:1.26.3-1.amzn2023.0.1.noarch 1/7
Installing : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 2/7
Installing : libunwind-1.4.0-5.amzn2023.0.2.x86_64 3/7
Installing : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 4/7
Installing : nginx-core-1:1.26.3-1.amzn2023.0.1.x86_64 5/7
Installing : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 6/7
Installing : nginx-1:1.26.3-1.amzn2023.0.1.x86_64 7/7
Running scriptlet: nginx-1:1.26.3-1.amzn2023.0.1.x86_64 7/7
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 1/7
Verifying : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 2/7
Verifying : libunwind-1.4.0-5.amzn2023.0.2.x86_64 3/7
Verifying : nginx-core-1:1.26.3-1.amzn2023.0.1.x86_64 4/7
Verifying : nginx-filesystem-1:1.26.3-1.amzn2023.0.1.noarch 5/7
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 6/7
Verifying : nginx-1:1.26.3-1.amzn2023.0.1.x86_64 7/7
-----
WARNING:

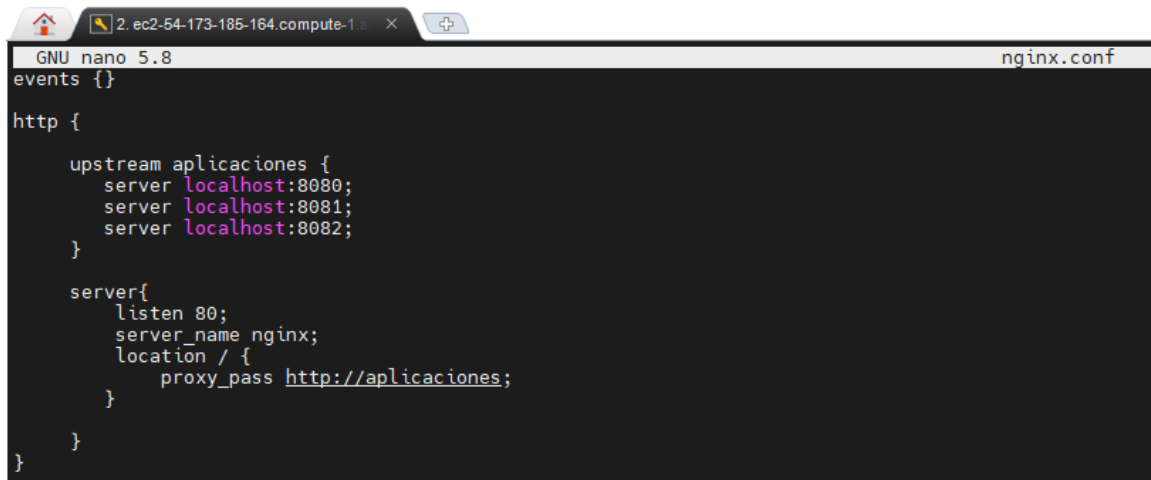
```



Para llevar a cabo el ejercicio de reverse proxy, se modificó la configuración del archivo `nginx.conf`. Sin embargo, antes de realizar los cambios, se creó una copia de seguridad del archivo original guardándola como `nginx.conf.BK`.

```
[root@ip-10-0-24-96 app2]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
2cb4a82bcd0   httpd    "httpd-foreground"     49 minutes ago Up 49 minutes 0.0.0.0:8082->80/tcp, :::8082->80/tcp app2
60d4eb052c09   httpd    "httpd-foreground"     About an hour ago Up About an hour 0.0.0.0:8081->80/tcp, :::8081->80/tcp app1
bae89e19cc86   httpd    "httpd-foreground"     2 hours ago   Up About an hour 0.0.0.0:8080->80/tcp, :::8080->80/tcp apache1
[root@ip-10-0-24-96 app2]# cd /etc/nginx/
[root@ip-10-0-24-96 nginx]# cp nginx.conf nginx.conf.BK
[root@ip-10-0-24-96 nginx]# ls
bash: ls: command not found
[root@ip-10-0-24-96 nginx]# ls
conf.d      fastcgi.conf      fastcgi_params    koi-utf          mime.types        nginx.conf        nginx.conf.default  scgi_params.default  uwsgi_params.default
default.d  fastcgi.conf.default  fastcgi_params.default  koi-win         mime.types.default  nginx.conf.BK     scgi_params        uwsgi_params      win-utf
[root@ip-10-0-24-96 nginx]#
```

Seguidamente, se realizó una modificación en el archivo donde se configuraron los puertos con el fin de balancear la carga entre ellos. Al terminar la configuración se realizó el reinicio con el comando “systemctl restart nginx”

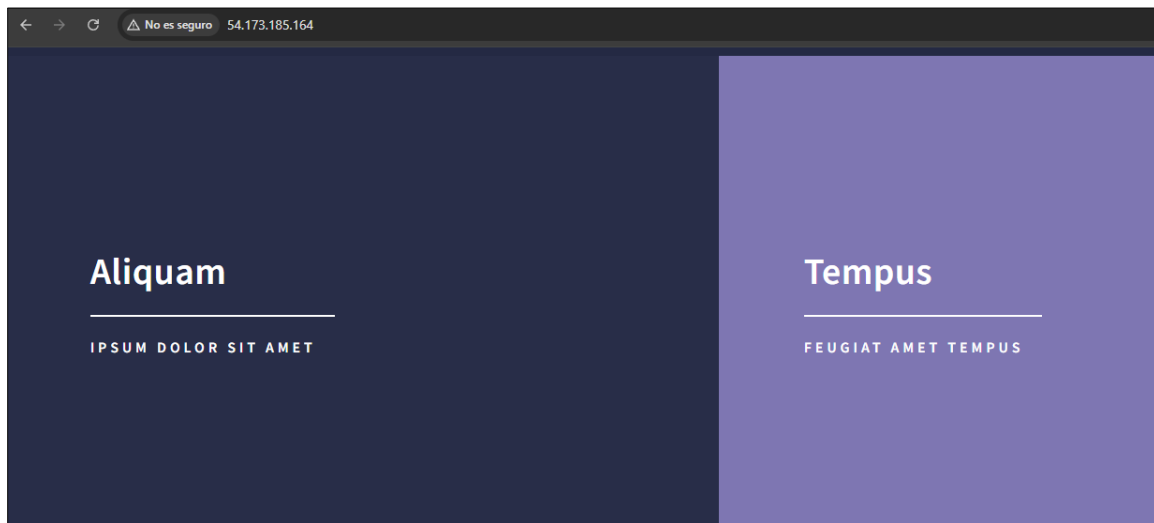
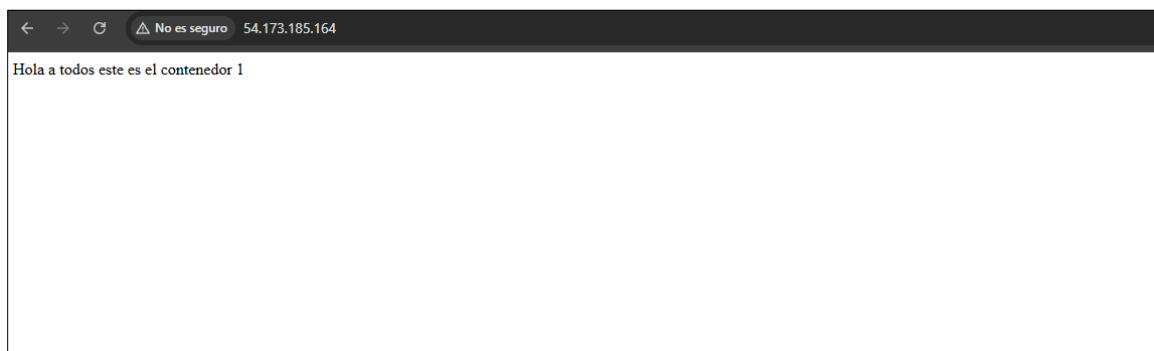
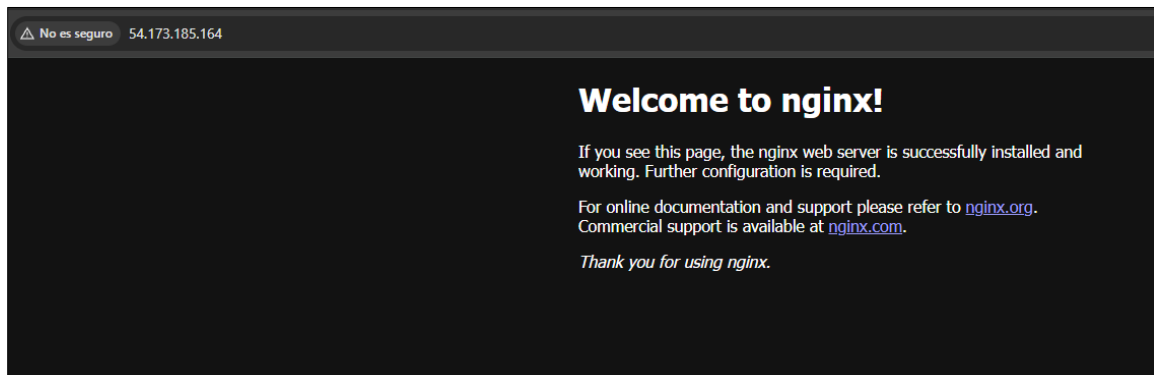


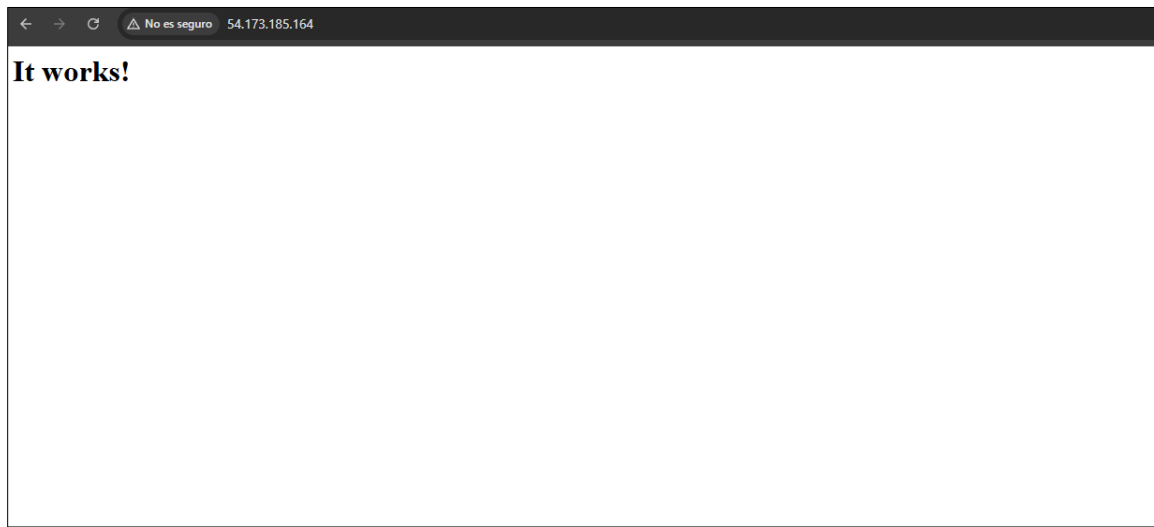
```
GNU nano 5.8 nginx.conf
events {}

http {
    upstream aplicaciones {
        server localhost:8080;
        server localhost:8081;
        server localhost:8082;
    }

    server {
        listen 80;
        server_name nginx;
        location / {
            proxy_pass http://aplicaciones;
        }
    }
}
```

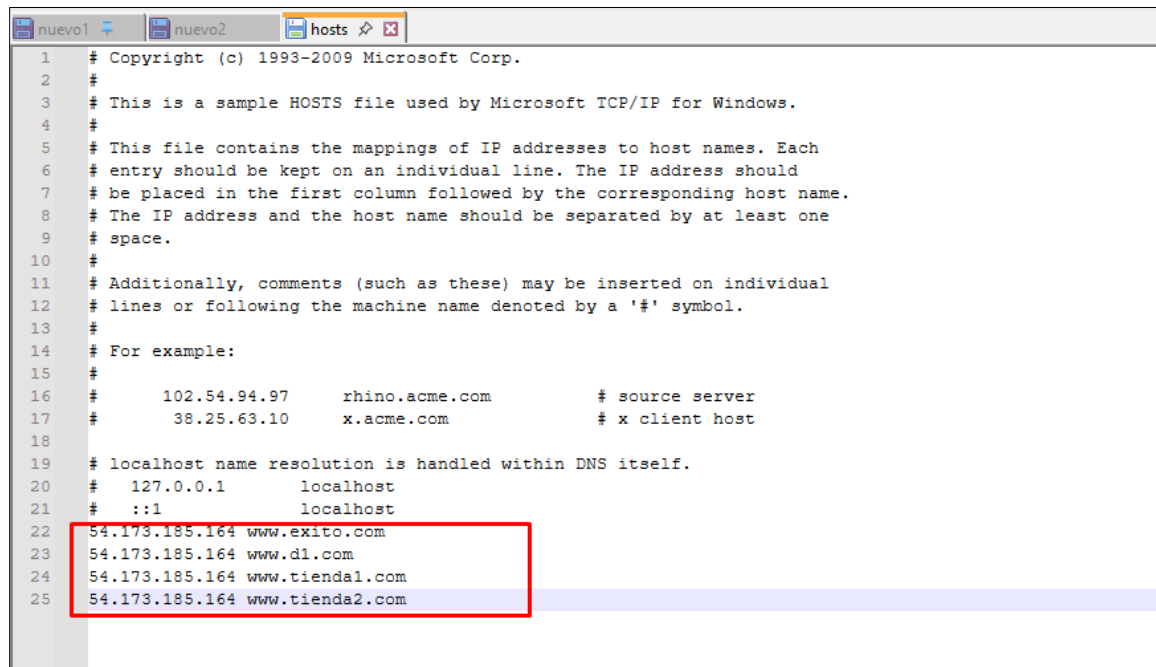
Se verificó el correcto funcionamiento accediendo al sitio a través del puerto 80. Al actualizar la página, el sitio web pasó por cada uno de los tres contenedores, mostrando diferentes versiones del sitio, como se evidencia en las imágenes adjuntas. Lo que quiere decir que las peticiones se fueron balanceando correctamente por cada uno de los contenedores.





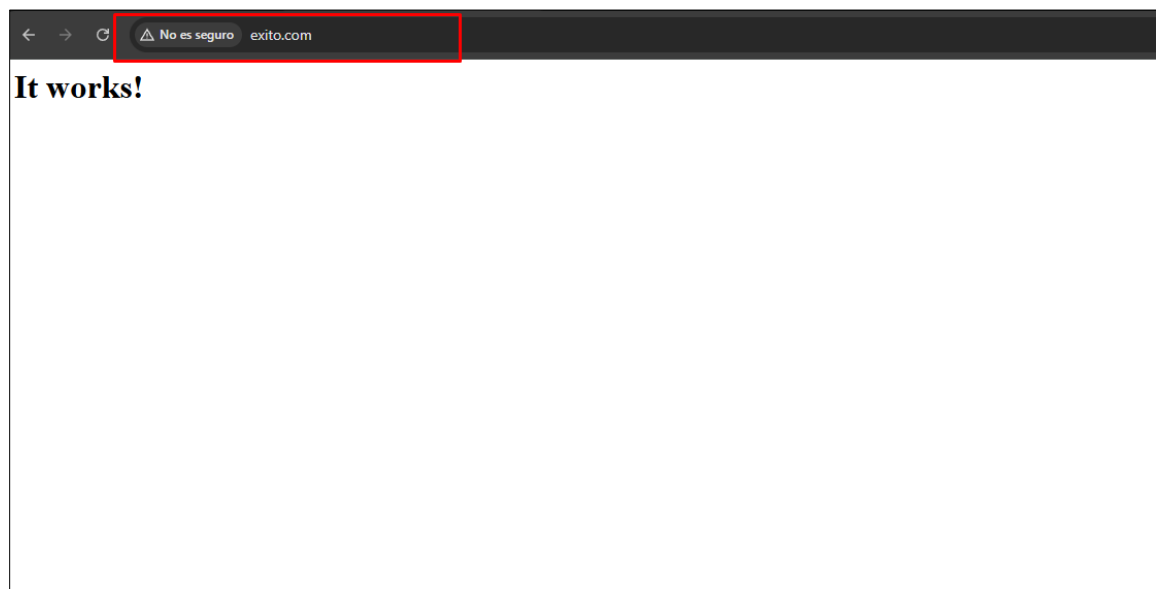
Como el objetivo del trabajo era realizar el llamado directamente desde el navegador utilizando solo el nombre de dominio, sin especificar el puerto. Para lograr esto, se implementó un proxy reverso con Nginx. En el archivo `nginx.conf`, se creó una nueva configuración que permitió redirigir las solicitudes a cada contenedor correspondiente sin necesidad de incluir el puerto en la URL. En esta configuración, se utilizó un `service_name` donde se especificó cual es el sitio al que debía dirigirse de acuerdo con el puerto del contenedor y a la URL digitada en el navegador.

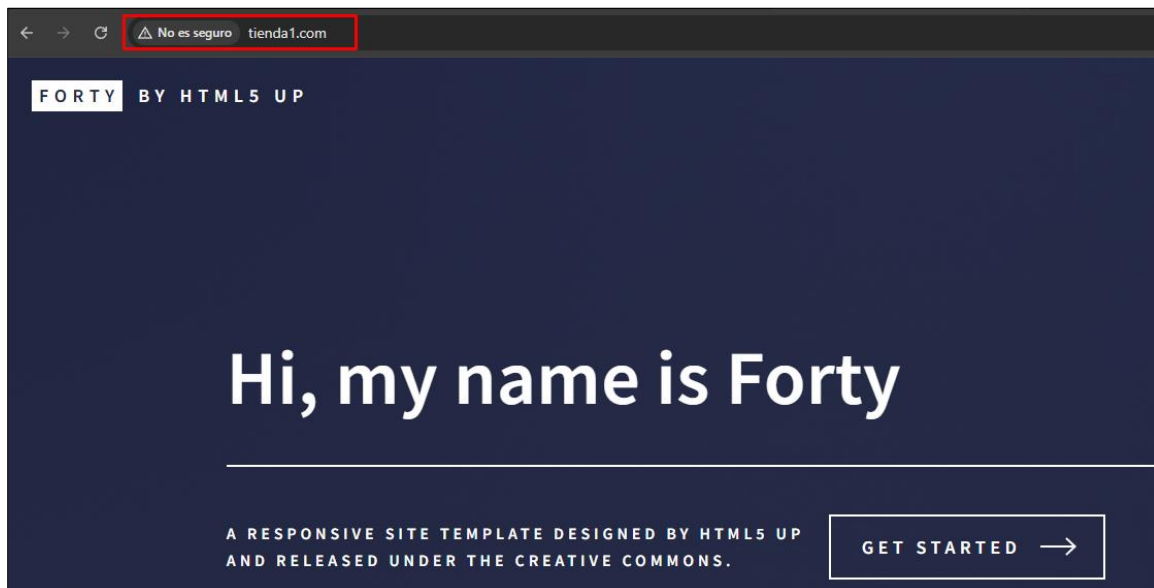
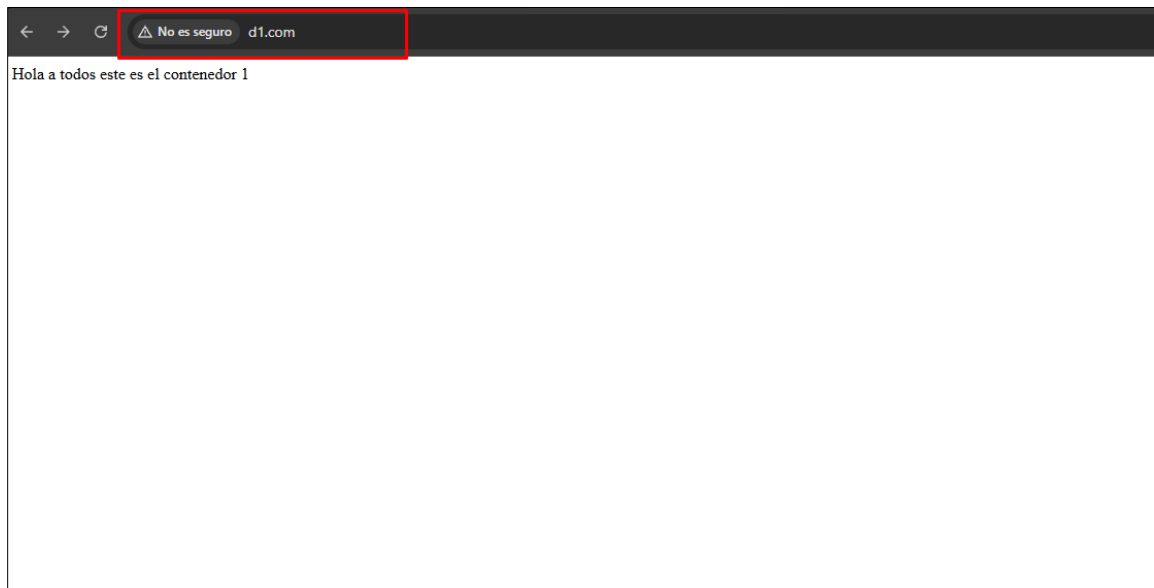
Como no disponemos de un dominio en internet, se procedió a editar el archivo `hosts` ubicado en `C:\Windows\System32\drivers\etc`. En este archivo, se agregó la dirección IP Pública de la instancia seguida de los cuatro dominios a los que deben dirigirse los contenedores.

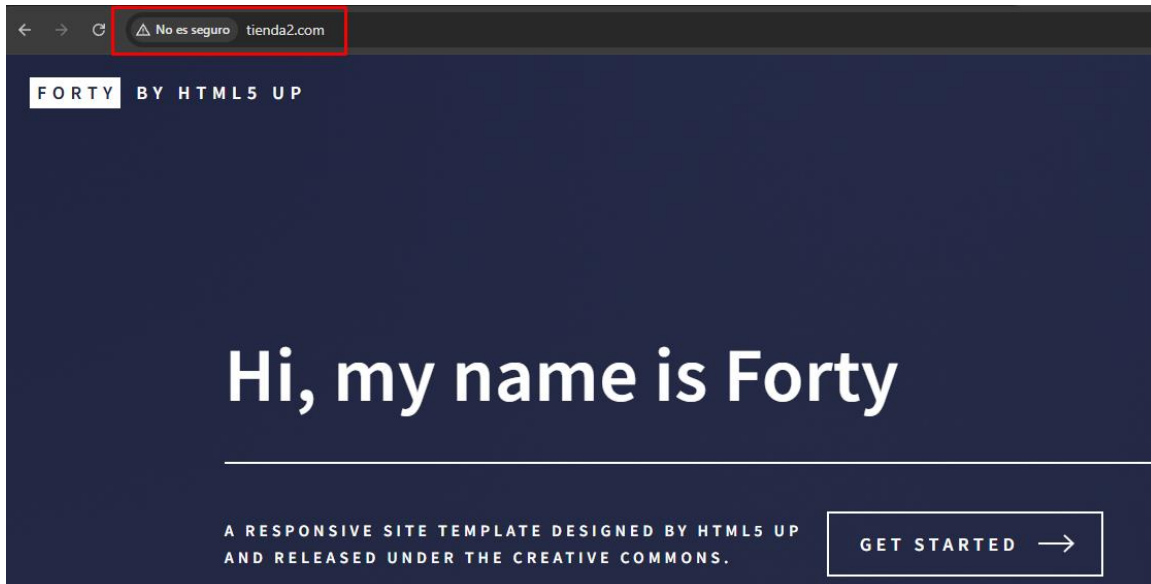


```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #     102.54.94.97       rhino.acme.com          # source server
17 #     38.25.63.10      x.acme.com              # x client host
18
19 # localhost name resolution is handled within DNS itself.
20 #   127.0.0.1          localhost
21 #   ::1                localhost
22 54.173.185.164 www.exito.com
23 54.173.185.164 www.d1.com
24 54.173.185.164 www.tiendal.com
25 54.173.185.164 www.tienda2.com
```

Posteriormente, se accedió a cada uno de ellos desde el navegador para validar su correcto funcionamiento. Gracias a esta configuración, cada URL mostró el contenedor correspondiente según lo parametrizado en el archivo nginx.conf







Para esta configuración ingresamos al servidor y posteriormente a la carpeta `app2` y a la ruta `cd /etc/nginx/`, allí creamos un archivo donde se realizó la configuración de los servidores. En este caso este archivo `sites.conf` recibe las solicitudes en el puerto 80 y las dirige a diferentes contenedores teniendo en cuenta el dominio y el puerto del contenedor usando el `proxy_pass`.

La ruta es la siguiente: `nano /etc/nginx/conf.d/sites.conf`

```
GNU nano 5.8 /etc/nginx/conf.d/sites.conf
server {
    listen 80;
    server_name www.exito.com;

    location / {
        proxy_pass http://127.0.0.1:8080; # apache1
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

server {
    listen 80;
    server_name www.d1.com;

    location / {
        proxy_pass http://127.0.0.1:8081; # app1
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

server {
    listen 80;
    server_name www.tienda1.com;

    location / {
        proxy_pass http://127.0.0.1:8082; # app2
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

server {
    listen 80;
    server_name www.tienda2.com;

    location / {
        proxy_pass http://127.0.0.1:8083; # app3
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

La primera parte del código indica cual es el servidor que recibirá las peticiones, en este caso son solicitudes HTTP en el puerto 80.

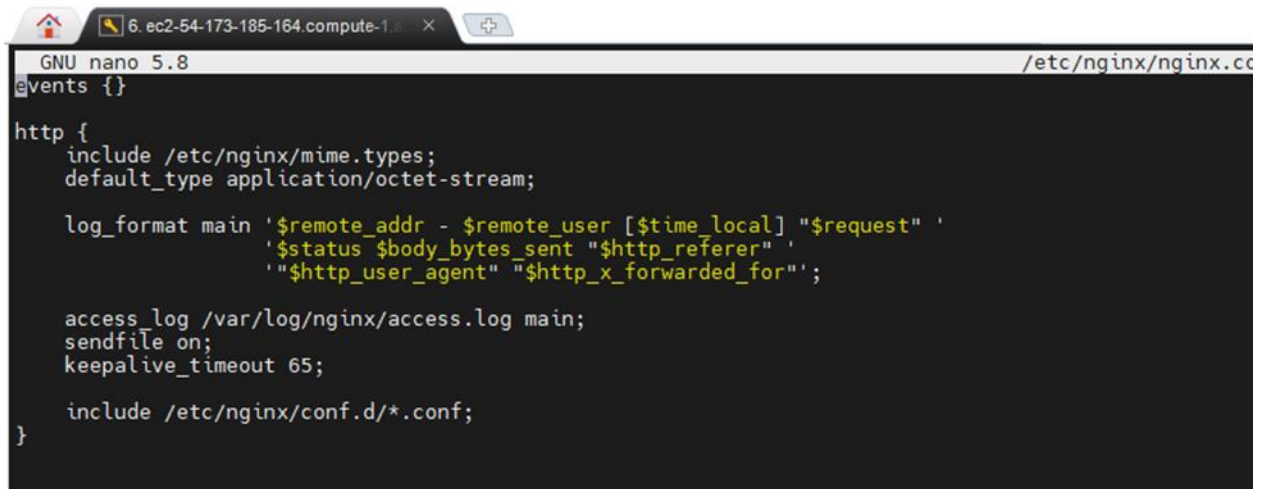
En el service_name se definieron los 4 servidores virtuales para cada dominio

- www.exito.com
- www.d1.com
- www.tienda1.com
- www.tienda2.com

Cada que el usuario accede, se dirige al local host 127.0.0.1 que se encuentra configurado en el archivo hosts seguidamente del puerto de cada contenedor (apache1, app1, app2 y app3), finalmente las líneas del proxy_set_header guardan el dominio al que se accede, después toma la IP a la que se accederá y finalmente mantiene el registro de las IPs.

Como ultimo pasó, se ingresó al archivo nginx.conf donde se realizó la configuración del HTTP permitiéndole realizar la carga de varios tipos de archivos y la generación de logs, y al final con el incluye carga las configuraciones adicionales de la ruta

“/etc/nginx/conf.d/*.conf”



```
GNU nano 5.8 /etc/nginx/nginx.conf
events {}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                   '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    sendfile on;
    keepalive_timeout 65;

    include /etc/nginx/conf.d/*.conf;
}
```

Conclusiones

Entender sobre AWS sus posibilidades y uso nos permite comprender a nosotros como nuevos ingenieros que ya no es necesario contar con infraestructura física para contar con un aplicativo funcional que pueda operar desde cualquier parte del mundo pues AWS posee un respaldo global respetando la gobernanza de datos (significa que hay datos que no pueden salir de ciertas regiones o países) lo que nos permite expandir nuestros clientes a nivel mundial.

Contar con diversas zonas de disponibilidad a la hora de realizar los despliegues nos va a permitir que las aplicaciones sean funcionales y resistentes a fallas lo que permitirá que sigan operando sin importar que tantas instancias estén disponibles. Además, es importante conocer que, aunque AWS tenga un apartado gratuito, para poder realizar despliegues grandes, es necesario contar con algún plan (Basic, Support, Bussiness, Enterprise) dependiendo de la capacidad de la aplicación que se quiera ejecutar.

Referencias

- Amazon Web Services. (s.f.). ¿Qué es la virtualización? Amazon Web Services. <https://aws.amazon.com/es/what-is/virtualization/>
- IBM. (2025, 2 de agosto). Máquinas virtuales: ¿Qué son y cómo funcionan? IBM. <https://www.ibm.com/mx-es/think/topics/virtual-machines#:~:text=El%202%20de%20agosto%20de,370%20que%20admitieron%20memoria%20virtual.>
- Wikipedia. (2025, 17 de marzo). Amazon Web Services: Historia. Wikipedia. https://es.wikipedia.org/wiki/Amazon_Web_Services#Historia
- Wikipedia contributors. Cronología de Amazon Web Services. Wikipedia. https://en.wikipedia.org/wiki/Timeline_of_Amazon_Web_Services#Full_timeline
- Slingerland, C. (2023, 15 de diciembre). The simple guide to the history of the cloud. CloudZero. <https://www.cloudzero.com/blog/history-of-the-cloud/>
- Wikipedia Kubernetes. Wikipedia, La Enciclopedia Libre. <https://es.wikipedia.org/wiki/Kubernetes>
- Wikipedia. Docker (software). Wikipedia, La Enciclopedia Libre. [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software))
- Microsoft. Productos de Azure. Microsoft Azure. <https://azure.microsoft.com/es-es/products#virtual-desktop-infrastructure>
- Carazo, F. (2025, febrero). Comparativa AWS vs GCP vs Azure. AWSDesdeCero. https://awsdesdecero.com/pubs/aws-01_comparativa-aws-vs-gcp-vs-azure
- Docker. httpd. Docker Hub de https://hub.docker.com/_/httpd
- HTML5 UP. (n.d.). HTML5 UP. <https://html5up.net/>
- Amazon Web Services. (s. f.). ¿Qué es AWS? AWS. https://aws.amazon.com/es/what-is-aws/?nc1=f_cc
- Amazon Web Services. (s. f.). Amazon EC2. AWS. <https://aws.amazon.com/es/ec2/>
- Amazon Web Services. (s. f.). Amazon VPC. AWS. https://aws.amazon.com/es/vpc/?nc2=type_a

- Amazon Web Services. (s. f.). Guía del usuario de Amazon EC2 para instancias de Linux – Imágenes de Amazon Machine (AMIs). AWS.
https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html
- Amazon Web Services. (s. f.). Grupos de seguridad para su VPC. AWS.
<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html>
- Amazon Web Services. (s. f.). Elastic Load Balancing. AWS.
https://aws.amazon.com/es/elasticloadbalancing/?nc2=type_a
- Amazon Web Services. (s. f.). Grupos de destino para balanceadores de carga de aplicación. AWS.
<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-target-groups.html>