



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

**Amazon Web Services**

**Karol Andrea Medina Suarez**  
**Tatiana Andrea Latorre Muñoz**  
**Jhon Breinner Bentiez Guerrero**

**Corporación Universitaria Remington.**  
**Facultad De Ingeniería**  
**Ingeniería En Sistemas**

**Juan Pablo Berrio López**  
**Seminario-Diplomado.**  
**2024**

## **Dedicatoria**

Ha sido años lleno de esfuerzos y sacrificios, cerrada esta etapa, nos queda dedicar y agradecer principalmente a Dios por habernos dado salud; fuerza para lograr nuestros objetivos, a nuestros padres por apoyarnos en todo momento, amigos y colegas que nos motivan a seguir obteniendo conocimiento informático con una fuerte pasión en la tecnología, a los profesores encargados de brindarnos sus clases con enfoque y determinación en enseñar de la mejor forma. Gracias al profesor Juan Pablo Berrio López por su guía a lo largo de este trabajo que nos enseñó el valor de la dedicación y el esfuerzo, sin mucho más que decir te agradecemos a ti por leer este documento que lo disfrutes.

## Tabla de Contenidos

Resumen.....	6
Marco conceptual y contextual .....	7
Creación de red, instancia y configuración de puertos .....	11
Configuración dentro AWS .....	11
<b>Paso 1.</b> Creamos una red virtual (VPC), con dos subredes pública y privada en una sola región y data center. ....	11
<b>Paso 2.</b> Creamos una instancia con Amazon Linux. ....	11
<b>Paso 2.1.</b> Configuración de clave privada para administrador del servidor SSH y IP publica que usa en la subred del VPC .....	11
<b>Paso 2.2.</b> Configuración SSH. ....	12
<b>Paso 3.</b> Configuración de almacenamiento. ....	12
<b>Paso 4.</b> Se crea la instancia. ....	12
<b>Paso 4.1.</b> se habilita el puerto 80 por donde ejecutamos el servicio apache (servidor web) cualquier IP puede ingresar por este puerto. ....	13
Configuración dentro de la instancia .....	13
<b>Paso 1.</b> Instalamos apache en usuario root .....	13
<b>Paso 1.1.</b> Iniciamos servicio .....	13
<b>Paso 1.2.</b> Servicio en running .....	14
<b>Paso 1.3.</b> creación de archivo index.html con permisos para creación de página de web ..	14
Vinculo YouTube - explicación Visualizar .....	15
Implementación del balanceador de cargas con el scaling .....	15
<b>Paso 1.</b> Creamos un snapshots de la instancia ya configurada para crear la AMI a utilizar y configurar otra instancia, esto con el fin de crear el balanceador de cargas con dos servidores funcionado .....	15
<b>Paso 1.1.</b> Se configura la instancia nueva a partir de la AMI creada .....	15
<b>Paso 2.</b> Una vez con las dos instancias creadas nos vamos a la opción de crear load Balancers y comenzamos su configuración, será un balanceador de cargas de aplicación	16
<b>Paso 2.1.</b> Colocamos nombre y se deja la opción de poder conectarnos desde la red internet .....	16
<b>Paso 2.2.</b> Configuración de red del VPC mas los datacenter con su sudred y IP publica ....	17
<b>Paso 2.3.</b> Se crea security group, que será la que recibirá las peticiones del balanceador de cargas para luego distribuir el trafico.....	17
<b>Paso 2.4.</b> Se escoge en la opción de configuración del balanceador de cargas .....	18
<b>Paso 2.5.</b> Se crea target group donde pondremos las instancias a trabajar en conjunto, realizando la configuración del puerto donde recibirán las peticiones a los servidores .....	18
<b>Paso 2.6.</b> Configuramos la ruta de la aplicación web se deja por defecto raíz (/) donde apache hace las validaciones de la página web.....	19
<b>Paso 2.7.</b> Luego se escogen las instancias a trabajar en conjunto para crear el target .....	19

<b>Paso 2.8.</b> Regresamos a la creación de balanceador de cargas y seleccionamos el target group .....	20
<b>Paso 3.</b> Nos dirigimos a la opción del auto Scaling groups para su creación.....	20
<b>Paso 3.1.</b> Creamos un lunch template para el auto scaling con la AMI de nuestro servidor	21
<b>Paso 3.2.</b> Configuramos el tipo de instancia, clave privada y configuración de red con el SG del balanceador de carga.....	21
<b>Paso 3.3.</b> El volumen debe ser configurado .....	22
<b>Paso 3.4.</b> Volvemos a la configuración del auto scaling group y escogemos el template configurado .....	22
<b>Paso 3.5.</b> Configuramos la red del auto Scaling con el VPC y las dos zonas de data center con sus subredes públicas; presionamos next. ....	23
<b>Paso 3.6.</b> Escogemos el balanceados de cargas ya creado para nuestro scaling group .....	23
<b>Paso 3.7.</b> Escogemos el valor deseado de instancias, el mínimo y el máximo presionamos next hasta su creación del scaling groups. ....	24
<b>Paso 3.8.</b> Una vez creado el scaling groups, iremos a la parte de target group para mirar el funcionamiento de la infraestructura creada; si pausamos el servicio eliminando una instancia se creará otra automáticamente y el servicio web seguirá funcionando correctamente. ....	24
Vinculo YouTube- explicaicon visual .....	25
Implementación de Docker con proxy inverso .....	25
Implementación de Contenedores .....	25
<b>Paso 1.</b> Una vez conectado en la instancia, primero deshabilitamos el servicio de apache instalado en la AMI, con el comando <b>systemctl stop httpd</b> y <b>systemctl disable httpd</b> , luego de ejecutar estos dos comandos comprobamos que el servicio web este apagado con el comando <b>systemctl status httpd</b> .....	26
<b>Paso 1.1.</b> instalamos el servidor web nginx, con el comando <b>yum install nginx</b> , luego lo iniciamos con el comando <b>systemctl start nginx</b> presionamos tecla enter e ingresamos este comando <b>systemctl enable nginx</b> que nos ayuda al inicio automáticamente por ultimo ingresamos el comando <b>systemctl status nginx</b> se debe visualizar activado. ....	26
<b>Paso 2.</b> instalamos las herramientas docker .....	27
<b>Paso 2.1.</b> Iniciamos docker con los siguientes comandos: .....	27
<b>Paso 2.2.</b> Descargamos del repositorio la imagen httpd para docker: .....	27
<b>Paso 2.3.</b> Revisamos que la imagen esta descargada con éxito.....	28
<b>Paso 2.4.</b> Creamos tres carpetas para tomar de origen, en esta ubicación estarán los archivos index.html de la aplicación .....	28
<b>Paso 2.5.</b> Creamos los contenedores con los siguientes comandos:.....	28
<b>Paso 3.</b> ingresamos a la ruta <b>/etc/nginx/nginx.config</b> y abrimos el archivo <b>nginx.config</b> con el comando <b>nano nginx.config</b> donde realizamos la configuración del servicio proxy para cuando llegue las peticiones ser reenviado a los contenedores.....	28
<b>Paso 4.</b> Por ultimo reiniciamos el nginx y cada petición que se hace a la instancia será enviada a los contenedores: .....	29
<b>Paso 5.</b> Creamos una snapshots de la instancia implementada con docker, creamos una AMI nueva y la remplazamos en el auto scaling groups .....	30
<b>Paso 6.</b> Funcionamiento con toda la infraestructura creada .....	31

	5
Vinculo YouTube-explicaicon visual .....	33
Diagrama de recursos usados y como se comunican entre ellos. ....	34
Conclusiones .....	35
Referencias.....	36

## Resumen

En este proyecto, somos un startup llamada **KTB SYSTEM**, una plataforma innovadora que ayudara conectar restaurantes con clientes mediante entregas rápidas. Nuestra empresa ha obtenido un rápido crecimiento para así escalar una infraestructura tecnológica, manejar una mayor demanda y poder garantizar la disponibilidad de mejorar los tiempos de respuesta. El CTO de **FAST FOOD** se ha diseñado bajo una arquitectura lo cual necesita de una ayuda para implementar en Amazon Web Services (AWS). La solución debe ser altamente disponible, escalable y estar diseñada para manejar una gran cantidad de trafico de manera eficiente. Implementaremos esta arquitectura en AWS lo cual requiere los siguientes requisitos

1. **Balanceador de Carga:** Configure un Application Load Balancer (ALB) para distribuir el tráfico entrante a múltiples instancias EC2.
2. **Instancias EC2:** Implemente al menos dos instancias EC2 en una configuración multizona para garantizar alta disponibilidad.
3. **Instancias con Proxy Reverso:** Dentro de cada instancia EC2, deben implementar un proxy reverso (por ejemplo, Nginx) para redirigir solicitudes a servicios internos.
4. **Auto-escalado:** Configure políticas de auto-escalado para aumentar o reducir las instancias EC2 según la carga.
5. **Seguridad:** Asegure el tráfico utilizando grupos de seguridad adecuados y habilitando HTTPS en el balanceador de carga.
6. **Documentación:** Toda la documentación debe estar en el formato de trabajo de grado proporcionado.

## Palabras clave

- Amazon web services
- VPC
- Instancia
- Balanceador de cargas
- Scaling groups

### Marco conceptual y contextual

Nombre	Significado
<p><b>Amazon Web Services (AWS)</b> (Amazon Web Services, 2021)</p>	<p>Es una plataforma en la nube más completa, muchas personas confían en ella, puede soportar la infraestructura y aplicaciones. Muchas organizaciones de todo tipo la utilizan para reducir costos, ser más ágiles e innovar más rápido, proporciona servicios tecnológicos bajo demanda a través de internet con precios de pago por uso. Tiene muchas características entre ellas ofrece más servicios que cualquier otro proveedor de nube lo cual hace que sea más rápido más fácil y rentable.</p>
<p><b>Amazon EC2(Amazon Elastic Compute Cloud)</b> (Amazon web Services, 2021)</p>	<p>Se encarga de crear computadoras la cual tienen un estado base, pueden subir de nivel y regresar a su estado base cuando lo necesite, estas se conocen como máquinas virtuales o instancias. Una máquina virtual es una computadora que no es física sino simulada por software, porque no está limitado a los recursos de hardware sino que por el software puede hacer que de la misma computadora física hayan varias máquinas virtuales también por el software se puede hacer que crezcan o reduzcan sus recursos y el uso que se dé teniendo en cuenta esto se pueden crear aplicaciones y responder a la demanda, tiene también la capacidad de auto escala en automático solo se debe configurar las reglas para decirle cuando debe escalar y cuando debe regresar al estado inicial como también se le pueden poner límites para que no escale hacia el infinito.</p>
<p><b>Instancia</b> (Amazon web Services, 2021)</p>	<p>Esta es un entorno virtual que brinda servicios en la nube de terceros, tiene el control desde que se inicia hasta que se elimina, también se puede personalizar para que se adapte a las necesidades correspondientes, esta puede administrar y mantener los recursos del servidor físico en las instalaciones. Los proveedores de la nube mantienen el hardware en sus centros</p>

	de datos y le dan acceso virtual a los recursos de computación, puede usar la instancia en la nube para ejecutar cargas de trabajo con uso intensivo de cómputos, como contenedores, bases de datos, micro servicios y máquinas virtuales.
<b>Amazon S3</b> (Amazon Web Services, 2021)	Este servicio es uno de lo más utilizados porque maneja un buen almacenamiento de objetos el cual ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento. Los usos que normalmente se manejan es la parte de almacenamiento y copia de seguridad, recuperación de daños, archivos, almacenamiento híbrido, aplicaciones, medios, data lake y analítica big data.
<b>Amazon RDS</b> (Amazon web Services, 2021)	Es un servicio racional fácil de manejar que elimina tareas administrativas que son ineficientes y que consumen tiempo este admite una variedad de motores de base de datos para almacenar y organizar datos también ayuda en las tareas de administración de base de datos como por ejemplo la copia de seguridad y la recuperación este también tiene facilidad de implementación
<b>Amazon route 53</b> (Amazon web services , 2021)	Es un servicio web de nombres de dominios también es escalable y tiene una alta disponibilidad se pueden realizar en registro de dominio direccionamientos, verificación de estado también permite personalizar el enrutamiento de DNS para mejorar la disponibilidad y garantizar el cumplimiento normativo.
<b>Amazon Machine Images-AMI</b> (Amazon web Services, 2021)	Este servidor ofrece un software suficiente para configurar y arrancar una instancia de Amazon EC2 incluido el sistema operativo y la configuración de seguridad, uno de los beneficios que nos ofrece es la velocidad eficiencia y conformidad en la implementación.
<b>Amazon Elastic block store (Amazon EBS)</b> (Amazon Web Services , 2021)	Este servidor da recursos de almacenamiento en bloque de alto rendimiento y escalables se puede crear, administrar algunos recursos como volúmenes de Amazon EBS e instantáneas

	también ofrece varios tipos de volúmenes copia de seguridad, recuperación protección de datos disponibilidad y durabilidad de los datos con archivado de datos.
<b>Amazon EC2 Auto Scaling</b> (Amazon Web Services, 2021)	Este método es muy utilizado en la nube por la cantidad de recursos computacionales ya que permite que reaccione en tiempo real a la demanda de recursos también tiene un sistema de servidores y este varía automáticamente en consecuencia de la carga total.
<b>VPC</b> (Amazon Web Services, 2021)	Es una red virtual idéntica a una red tradicional que opera su propio centro de datos esta puede crear subredes, esta red virtual es muy similar a la red tradicional que usaría en su propio centro de datos, pero con los beneficios que supone utilizar la infraestructura escalable de AWS.
<b>Sud-red</b> (Amazon web Services, 2021)	Una subred es una red que forma parte de una red principal. Las subredes potencian la eficiencia de las redes. Con la formación de subredes, el tráfico de la red puede recorrer una distancia reducida sin necesidad de transitar por routers superfluos para alcanzar su destino.
<b>Docker</b> (Perlow, 2024)	Es un software destinado a contenedores. El motor de Docker se instala en cada servidor donde se requiera la ejecución de contenedores, ofreciendo un conjunto simple de comandos que puede emplear para crear, iniciar o detener dichos contenedores.
<b>Nginx</b>	servidor proxy, su función es reenviar las peticiones a los contenedores docker
<b>Ip privada</b>	Numero identificativo de dispositivos internos de una red; usado en las instancias.
<b>Index.html</b>	Archivo de configuración para código web.
<b>Security group</b>	Seguridad para la configuración de puertos y paso de la información en una instancia o sistema operativo.
<b>Root</b>	Usuario administrador de un sistema operativo Linux.

<b>Ssh</b>	Herramienta de terminal para línea de comandos que nos permite conexión al instancias
<b>Puertos</b>	Numero identificador que guía el paso de la información por la red
<b>Target Group</b>	Grupo de instancias que funcionan en conjunto para un buen funcionamiento de la escalabilidad.
<b>Ip publica</b>	Numero identificativo de dispositivos Hardware como por ejemplo un Router de borde que proporción acceso a la red internet.
<b>Httpd</b>	servidor web que recibe las peticiones de los clientes, brindado el servicio.

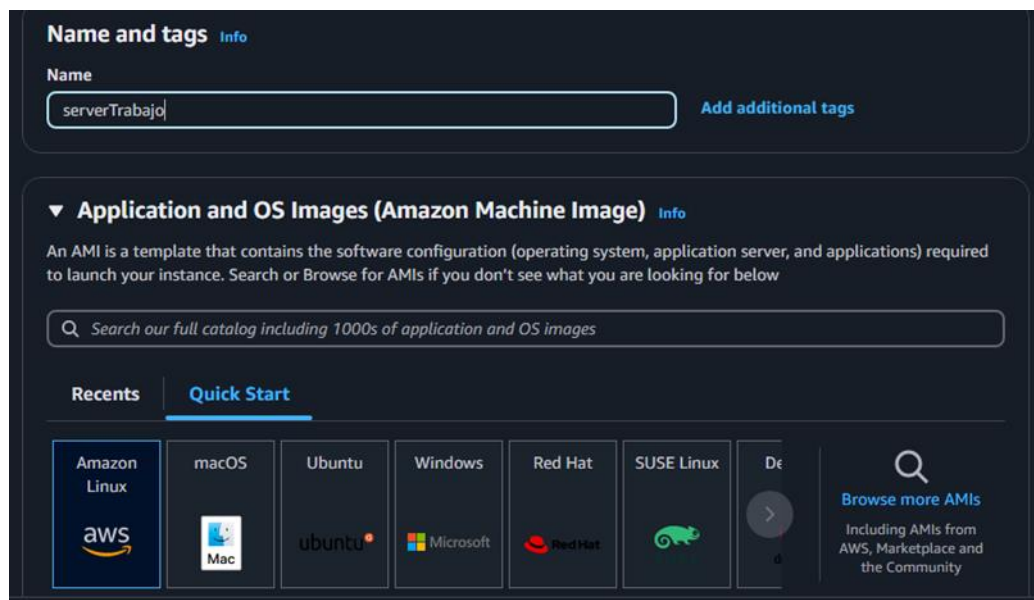
## Creación de red, instancia y configuración de puertos

### Configuración dentro AWS

**Paso 1.** Creamos una red virtual (VPC), con dos subredes pública y privada en una sola región y data center.

<input type="checkbox"/>	Name	VPC ID	State	Block Public...	IPv4 CIDR
<input type="checkbox"/>	remingtonVPC-vpc	<a href="#">vpc-06cfb333e43d9a3ac</a>	Available	Off	10.0.0.0/16

**Paso 2.** Creamos una instancia con Amazon Linux.



**Paso 2.1.** Configuración de clave privada para administrador del servidor SSH y IP pública que usa en la subred del VPC

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

**Key pair name - required**

server1 ↕ [Create new key pair](#)

**Network settings** [Info](#)

**VPC - required** [Info](#)

vpc-06cfb333e43d9a3ac (remingtonVPC-vpc)  
10.0.0.0/16 ↕

**Subnet** [Info](#)

subnet-00bedb75b12e01afc remingtonVPC-subnet-public1-us-east-1a  
VPC: vpc-06cfb333e43d9a3ac Owner: 779846801524  
Availability Zone: us-east-1a Zone type: Availability Zone  
IP addresses available: 4089 CIDR: 10.0.0.0/20 ↕ [Create new subnet](#)

**Auto-assign public IP** [Info](#)

Enable ↕

Additional charges apply when outside of free tier allowance

## Paso 2.2. Configuración SSH.

**Description - required** [Info](#)

launch-wizard-1 created 2024-11-24T19:31:46.734Z

**Inbound Security Group Rules**

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) [Remove](#)

Type	Protocol	Port range	Source type	Source	Description - optional
ssh	TCP	22	Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

## Paso 3. Configuración de almacenamiento.

**Configure storage** [Info](#) Advanced

1x  GiB  ↕ Root volume (Not encrypted)

## Paso 4. Se crea la instancia.

serverTrabajo
 
✔ Running
🔍 🔊 t2.micro
✔ 2/2 checks passed



```

root@ip-10-0-12-129:/home/ec2-user
Verifying      : httpd-core-2.4.62-1.amzn2023.x86_64        6/12
Verifying      : httpd-filesystem-2.4.62-1.amzn2023.noarch  7/12
Verifying      : httpd-tools-2.4.62-1.amzn2023.x86_64     8/12
Verifying      : libbrotli-1.0.9-4.amzn2023.0.2.x86_64    9/12
Verifying      : mailcap-2.1.49-3.amzn2023.0.3.noarch     10/12
Verifying      : mod_http2-2.0.27-1.amzn2023.0.3.x86_64  11/12
Verifying      : mod_lua-2.4.62-1.amzn2023.x86_64        12/12

Installed:
apr-1.7.2-2.amzn2023.0.2.x86_64
apr-util-1.6.3-1.amzn2023.0.1.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-2.4.62-1.amzn2023.x86_64
httpd-core-2.4.62-1.amzn2023.x86_64
httpd-filesystem-2.4.62-1.amzn2023.noarch
httpd-tools-2.4.62-1.amzn2023.x86_64
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-10-0-12-129 ec2-user]# systemctl start httpd

```

## Paso 1.2. Servicio en running

```

root@ip-10-0-12-129:/home/ec2-user
[root@ip-10-0-12-129 ec2-user]# systemctl start httpd
[root@ip-10-0-12-129 ec2-user]# systemctl sttus httpd
Unknown command verb sttus.
[root@ip-10-0-12-129 ec2-user]# systemctl status httpd
• httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: d
   Active: active (running) since Sun 2024-11-24 20:31:49 UTC; 13s ago
     Docs: man:httpd.service(8)
   Main PID: 27015 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes
     Tasks: 177 (limit: 1111)
   Memory: 12.9M
     CPU: 62ms
   CGroup: /system.slice/httpd.service
           └─27015 /usr/sbin/httpd -DFOREGROUND
             └─27033 /usr/sbin/httpd -DFOREGROUND
               └─27035 /usr/sbin/httpd -DFOREGROUND
                 └─27036 /usr/sbin/httpd -DFOREGROUND
                   └─27037 /usr/sbin/httpd -DFOREGROUND

Nov 24 20:31:49 ip-10-0-12-129.ec2.internal systemd[1]: Starting httpd.service
Nov 24 20:31:49 ip-10-0-12-129.ec2.internal systemd[1]: Started httpd.service
Nov 24 20:31:49 ip-10-0-12-129.ec2.internal httpd[27015]: Server configured, 11

```

## Paso 1.3. creación de archivo index.html con permisos para creación de página de web

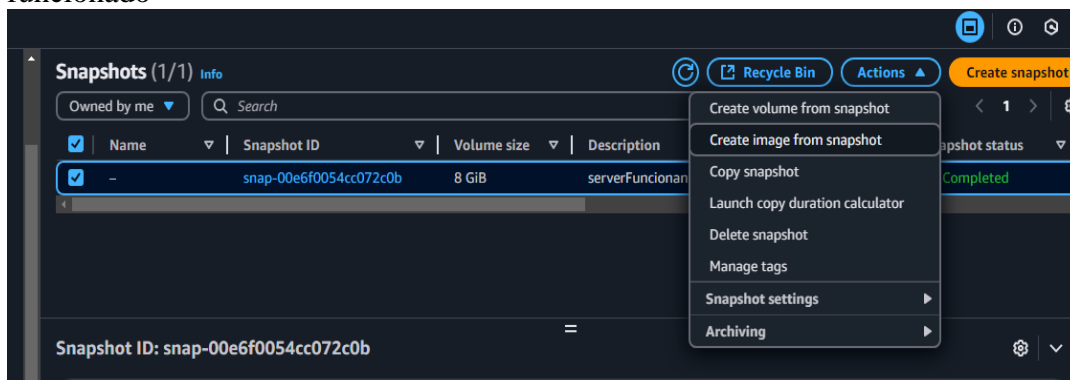
```
[root@ip-10-0-12-129 /]# cd /var/www/html/
[root@ip-10-0-12-129 html]# ls
[root@ip-10-0-12-129 html]# touch index.html
[root@ip-10-0-12-129 html]# chmod 777 index.html
[root@ip-10-0-12-129 html]# ls
index.html
[root@ip-10-0-12-129 html]# ls -l
total 0
-rwxrwxrwx. 1 root root 0 Nov 24 20:37 index.html
[root@ip-10-0-12-129 html]#
```

### Vinculo YouTube - explicación Visualizar

<https://www.youtube.com/watch?v=kyzVRSNIav8>

### Implementación del balanceador de cargas con el scaling

**Paso 1.** Creamos un snapshots de la instancia ya configurada para crear la AMI a utilizar y configurar otra instancia, esto con el fin de crear el balanceador de cargas con dos servidores funcionando



**Paso 1.1.** Se configura la instancia nueva a partir de la AMI creada

**Name and tags** [Info](#)

Name

ServerTrabajo2 [Add additional tags](#)

▼ **Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

🔍 Search our full catalog including 1000s of application and OS Images

Recents **My AMIs** Quick Start

Owned by me  Shared with me


[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

AmazonLinuxdeBrelner  
ami-0578432f0bd550d2f  
2024-11-29T02:48:31.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

**Paso 2.** Una vez con las dos instancias creadas nos vamos a la opción de crear load Balancers y comenzamos su configuración, será un balanceador de cargas de aplicación

**Application Load Balancer** [Info](#)



Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

[Create](#)

**Paso 2.1.** Colocamos nombre y se deja la opción de poder conectarnos desde la red internet

## Create Application Load Balancer Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and Amazon ElastiCache. When an application load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, forwards the request to the target.

► **How Application Load Balancers work**

### Basic configuration

**Load balancer name**  
Name must be unique within your AWS account and can't be changed after the load balancer is created.

LBPuebas1

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Scheme** Info  
Scheme can't be changed after the load balancer is created.

**Internet-facing**

- Serves internet-facing traffic.
- Has public IP addresses.
- DNS name is publicly resolvable.
- Requires a public subnet.

**Internal**

- Serves internal traffic.
- Has private IP addresses.
- DNS name is not publicly resolvable.
- Compatible with the IPv4 and Dualstack IP address types.

**Load balancer IP address type** Info  
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address type.

**IPv4**  
Includes only IPv4 addresses.

**Dualstack**  
Includes IPv4 and IPv6 addresses.

**Dualstack without public IPv4**  
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with internet-facing load balancers only.

## Paso 2.2. Configuración de red del VPC mas los datacenter con su sudred y IP publica

### Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC** Info  
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view [target groups](#). For a new VPC, [create a VPC](#).

remingtonVPC-vpc  
vpc-06c1b333e43d9a3ac  
IPv4 VPC CIDR: 10.0.0.0/16

**Mappings** Info  
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

**Availability Zones**

**us-east-1a (use1-az4)**

Subnet

subnet-00bedb75b12e01afc  
IPv4 subnet CIDR: 10.0.0.0/20

remingtonVPC-subnet-public1-us-east-1a

**IPv4 address**  
Assigned by AWS

**us-east-1b (use1-az6)**

Subnet

subnet-0fa72699cf59e6886  
IPv4 subnet CIDR: 10.0.192.0/20

publica2reming

**Paso 2.3.** Se crea security group, que será la que recibirá las peticiones del balanceador de cargas para luego distribuir el trafico

**Create security group** Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

**Security group name** Info

Name cannot be edited after creation.

**Description** Info

**VPC** Info

**Inbound rules** Info

Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	80	Any... 0.0.0.0/0	

0.0.0.0/0 X

[Add rule](#)

## Paso 2.4. Se escoge en la opción de configuración del balanceador de cargas

**Security groups** Info

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

**Security groups**

Select up to 5 security groups

SGBalancedorCarga  
sg-0585c825a4e1fdccd VPC: vpc-06cfb333e43d9a3ac

## Paso 2.5. Se crea target group donde pondremos las instancias a trabajar en conjunto, realizando la configuración del puerto donde recibirán las peticiones a los servidores

**Configuración básica**

Las configuraciones en esta sección no se pueden cambiar una vez creado el grupo objetivo.

**Elija un tipo de objetivo**

**Instancias**

- Admite equilibrio de carga en instancias dentro de una VPC específica.
- Facilita el uso de [Amazon EC2 Auto Scaling](#) para administrar y escalar su capacidad EC2.

**Direcciones IP**

- Admite equilibrio de carga en VPC y recursos locales.
- Facilita el enrutamiento a múltiples direcciones IP e interfaces de red en la misma instancia.
- Ofrece flexibilidad con arquitecturas basadas en microservicios, simplificando la comunicación entre aplicaciones.
- Admite objetivos IPv6, lo que permite la comunicación IPv6 de extremo a extremo y NAT de IPv4 a IPv6.

**Función lambda**

- Facilita el enrutamiento a una única función Lambda.
- Accesible únicamente para balanceadores de carga de aplicaciones.

**Balanceador de carga de aplicaciones**

- Ofrece la flexibilidad para que un balanceador de carga de red acepte y enrute solicitudes TCP dentro de una VPC específica.
- Facilita el uso de direcciones IP estáticas y PrivateLink con un balanceador de carga de aplicaciones.

**Nombre del grupo objetivo**

Se permite un máximo de 32 caracteres alfanuméricos, incluidos los guiones, pero el nombre no debe comenzar ni terminar con un guion.

**Protocolo: Puerto**

**Nombre del grupo objetivo**

TGreming1

Se permite un máximo de 32 caracteres alfanuméricos, incluidos los guiones, pero el nombre no debe comenzar ni terminar con un guion.

**Protocolo: Puerto**

Elija un protocolo para su grupo de destino que corresponda al tipo de Load Balancer que enrutará el tráfico hacia él. Algunos protocolos ahora incluyen detección de anomalías para los destinos y puede configurar opciones de mitigación una vez que se crea su grupo de destino. Esta opción no se puede cambiar después de la creación.

HTTP 80

1-65535

**Tipo de dirección IP**

Sólo los objetivos con el tipo de dirección IP indicado se pueden registrar en este grupo de objetivos.

IPv4

Cada instancia tiene una interfaz de red predeterminada (eth0) a la que se le asigna la dirección IPv4 privada principal. La dirección IPv4 privada principal de la instancia es la que se aplicará al destino.

IPv6

Cada instancia que registres debe tener una dirección IPv6 principal asignada. Esta se configura en la interfaz de red predeterminada de la instancia (eth0). [Más información](#)

**VPC**

Seleccione la VPC con las instancias que desea incluir en el grupo de destino. En esta lista solo están disponibles las VPC que admiten el tipo de dirección IP seleccionado anteriormente.

RemingtonVPC-vpc  
vpc-06c7b333e43d9a3ac  
CIDR de VPC IPv4: 10.0.0.0/16

**Versión del protocolo**

HTTP1

Envía solicitudes a destinos mediante HTTP/1.1. Compatible cuando el protocolo de solicitud es HTTP/1.1 o HTTP/2.

**Paso 2.6.** Configuramos la ruta de la aplicación web se deja por defecto raíz (/) donde apache hace las validaciones de la página web

**Nombre del grupo objetivo**

TGreming1

Se permite un máximo de 32 caracteres alfanuméricos, incluidos los guiones, pero el nombre no debe comenzar ni terminar con un guion.

**Protocolo: Puerto**

Elija un protocolo para su grupo de destino que corresponda al tipo de Load Balancer que enrutará el tráfico hacia él. Algunos protocolos ahora incluyen detección de anomalías para los destinos y puede configurar opciones de mitigación una vez que se crea su grupo de destino. Esta opción no se puede cambiar después de la creación.

HTTP 80

1-65535

**Tipo de dirección IP**

Sólo los objetivos con el tipo de dirección IP indicado se pueden registrar en este grupo de objetivos.

IPv4

Cada instancia tiene una interfaz de red predeterminada (eth0) a la que se le asigna la dirección IPv4 privada principal. La dirección IPv4 privada principal de la instancia es la que se aplicará al destino.

IPv6

Cada instancia que registres debe tener una dirección IPv6 principal asignada. Esta se configura en la interfaz de red predeterminada de la instancia (eth0). [Más información](#)

**VPC**

Seleccione la VPC con las instancias que desea incluir en el grupo de destino. En esta lista solo están disponibles las VPC que admiten el tipo de dirección IP seleccionado anteriormente.

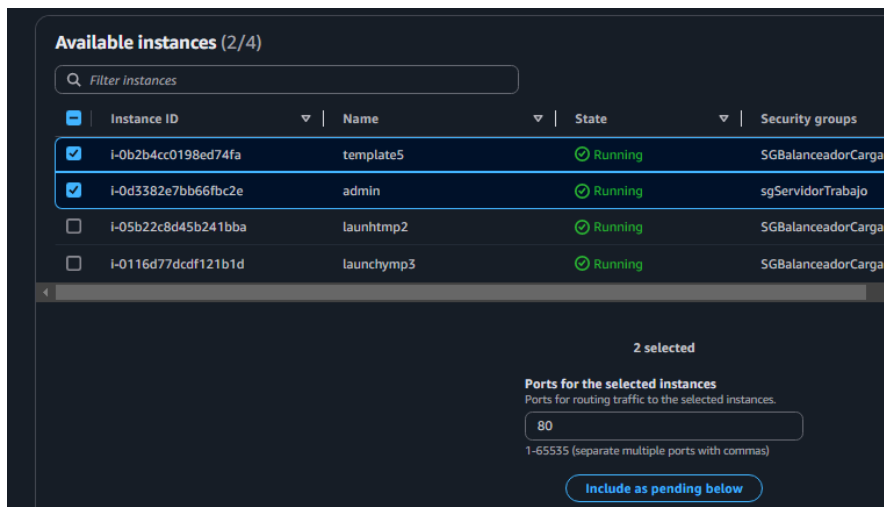
RemingtonVPC-vpc  
vpc-06c7b333e43d9a3ac  
CIDR de VPC IPv4: 10.0.0.0/16

**Versión del protocolo**

HTTP1

Envía solicitudes a destinos mediante HTTP/1.1. Compatible cuando el protocolo de solicitud es HTTP/1.1 o HTTP/2.

**Paso 2.7.** Luego se escogen las instancias a trabajar en conjunto para crear el target



**Available instances (2/4)**

Filter instances

<input type="checkbox"/>	Instance ID	Name	State	Security groups
<input checked="" type="checkbox"/>	i-0b2b4cc0198ed74fa	template5	Running	SGBalanceadorCarga
<input checked="" type="checkbox"/>	i-0d3382e7bb66fbc2e	admin	Running	sgServidorTrabajo
<input type="checkbox"/>	i-05b22c8d45b241bba	launhtmp2	Running	SGBalanceadorCarga
<input type="checkbox"/>	i-0116d77dcd121b1d	launchymp3	Running	SGBalanceadorCarga

2 selected

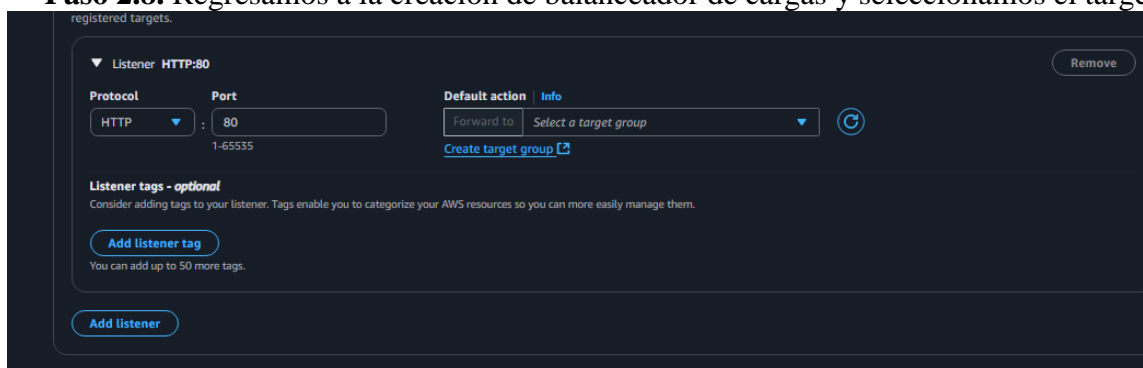
**Ports for the selected instances**  
Ports for routing traffic to the selected instances.

80

1-65535 (separate multiple ports with commas)

[Include as pending below](#)

### Paso 2.8. Regresamos a la creación de balanceador de cargas y seleccionamos el target group



registered targets.

▼ Listener HTTP:80 [Remove](#)

<b>Protocol</b>	<b>Port</b>	<b>Default action</b> <a href="#">Info</a>
HTTP	80 1-65535	Forward to: <a href="#">Select a target group</a> <a href="#">Create target group</a>

**Listener tags - optional**  
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)

You can add up to 50 more tags.

[Add listener](#)

### Paso 3. Nos dirigimos a la opción del auto Scaling groups para su creación

## Choose launch template [Info](#)

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

**Name**

**Auto Scaling group name**  
Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

**Launch template** [Info](#)

ⓘ For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

**Launch template**  
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.



[Create a launch template](#) [↗](#)

### Paso 3.1. Creamos un lunch template para el auto scaling con la AMI de nuestro servidor

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '"', '@'.

**Template version description**

Max 255 chars.

**Auto Scaling guidance** [Info](#)  
Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

▶ **Template tags**

▶ **Source template**

**Launch template contents**  
Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

▼ **Application and OS Images (Amazon Machine Image) - required** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your Instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents **My AMIs** Quick Start

Owned by me  Shared with me

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**  
AmazonLinuxBaller

### Paso 3.2. Configuramos el tipo de instancia, clave privada y configuración de red con el SG del balanceador de carga

**Instance type** [Info](#) | [Get advice](#) Advanced

Instance type

t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true On-Demand Windows base pricing: 0.0162 USD per Hour  
 On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour On-Demand SUSE base pricing: 0.0116 USD per Hour  
 On-Demand RHEL base pricing: 0.026 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour

All generations [Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

---

**Key pair (login)** [Info](#)

You can use a key pair to securely connect to your Instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

server1 [Create new key pair](#)

---

**Network settings** [Info](#)

Subnet [Info](#)

Don't include in launch template [Create new subnet](#)

When you specify a subnet, a network interface is automatically added to your template.

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Select existing security group  Create security group

Common security groups [Info](#)

Select security groups

SGBalanceadorCarga sg-0585c825a4e1fdccd [Compare security group rules](#)  
 VPC: vpc-06cfb333e43d9a3ac

Security groups that you add or remove here will be added to or removed from all your network interfaces.

### Paso 3.3. El volumen debe ser configurado

**Storage (volumes)** [Info](#) Hide details

EBS Volumes

**Volume 1 (AMI Root)**  
 AMI Volumes are not included in the template unless modified

<b>Storage type</b> <a href="#">Info</a>	<b>Device name - required</b> <a href="#">Info</a>	<b>Snapshot</b> <a href="#">Info</a>
EBS	/dev/sda1	snap-00e6f0054cc072c0b
<b>Size (GiB)</b> <a href="#">Info</a>	<b>Volume type</b> <a href="#">Info</a>	<b>IOPS</b> <a href="#">Info</a>
8	gp3	3000
<b>Delete on termination</b> <a href="#">Info</a>	<b>Encrypted</b> <a href="#">Info</a>	<b>KMS key</b> <a href="#">Info</a>
Yes	Not encrypted	Don't include in launch template
<b>Throughput</b> <a href="#">Info</a>		<small>KMS keys are only applicable when encryption is set on this volume.</small>
125		

**Paso 3.4.** Volvemos a la configuración del auto scaling group y escogemos el template configurado

**Launch template** Info

ⓘ For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

**Launch template**  
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

LTremington

Create a launch template [↗](#)

**Version**  
Default (1)

Create a launch template version [↗](#)

<b>Description</b> template	<b>Launch template</b> LTremington <a href="#">↗</a> lt-0a6543bc76e8e8ca2	<b>Instance type</b> t2.micro
<b>AMI ID</b> ami-0578432f0bd550d2f	<b>Security groups</b> -	<b>Request Spot Instances</b> No
<b>Key pair name</b> server1	<b>Security group IDs</b> sg-0585c825a4e1fdccd <a href="#">↗</a>	

**Additional details**

<b>Storage (volumes)</b> -	<b>Date created</b> Sun Dec 01 2024 21:49:34 GMT-0500 (hora estándar de Colombia)
-------------------------------	--

**Paso 3.5.** Configuramos la red del auto Scaling con el VPC y las dos zonas de data center con sus subredes públicas; presionamos next.

**Network** Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your Instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**  
Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-06cfb333e43d9a3ac (remingtonVPC-vpc)  
10.0.0.0/16

Create a VPC [↗](#)

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1a | subnet-00bedb75b12e01afc (remingtonVPC-subnet-public1-us-east-1a)  
10.0.0.0/20

us-east-1b | subnet-0fa72699cf59e6886 (publica2reming)  
10.0.192.0/20

Create a subnet [↗](#)

**Availability Zone distribution - new**  
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

**Balanced best effort**  
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

**Balanced only**  
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

Cancel Skip to review Previous **Next**

**Paso 3.6.** Escogemos el balanceados de cargas ya creado para nuestro scaling group

**Load balancing** Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer  
 Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer  
 Choose from your existing load balancers.

Attach to a new load balancer  
 Quickly create a basic load balancer to attach to your Auto Scaling group.

---

**Attach to an existing load balancer**

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups  
 This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

**Existing load balancer target groups**

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

TGrening1 | HTTP  
 Application Load Balancer: LBPruebas1

**Paso 3.7.** Escogemos el valor deseado de instancias, el mínimo y el máximo presionamos next hasta su creación del scaling groups.

**Desired capacity**

Specify your group size.

3

---

**Scaling** Info

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**

Set limits on how much your desired capacity can be increased or decreased.

**Min desired capacity** **Max desired capacity**

2 5

Equal or less than desired capacity Equal or greater than desired capacity

**Paso 3.8.** Una vez creado el scaling groups, iremos a la parte de target group para mirar el funcionamiento de la infraestructura creada; si pausamos el servicio eliminando una instancia se creará otra automáticamente y el servicio web seguirá funcionando correctamente.

**TGremin1** Actions ▾

**Details**  
 arn:aws:elasticloadbalancing:us-east-1:779846801524:targetgroup/TGremin1/7d89114aca438e76

<b>Target type</b> Instance	<b>Protocol : Port</b> HTTP: 80	<b>Protocol version</b> HTTP1	<b>VPC</b> <a href="#">vpc-06cfb333e43d9a3ac</a>
<b>IP address type</b> IPv4	<b>Load balancer</b> <a href="#">LBPruebas1</a>		

<b>3</b> Total targets	<b>3</b> Healthy 0 Anomalous	<b>0</b> Unhealthy	<b>0</b> Unused	<b>0</b> Initial	<b>0</b> Draining
---------------------------	------------------------------------	-----------------------	--------------------	---------------------	----------------------

► **Distribution of targets by Availability Zone (AZ)**  
 Select values in this table to see corresponding filters applied to the Registered targets table below.

**Targets** | Monitoring | Health checks | Attributes | Tags

**Registered targets (3)** [info](#) Anomaly mitigation: Not applicable Deregister Register targets

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.

## Vinculo YouTube- explicaicon visual

<https://youtu.be/xMILvRhVs8E>

## Implementación de Docker con proxy inverso

### Implementación de Contenedores

Hasta este punto toda la infraestructura está trabajando correctamente con el VPC, subredes, las instancias, el balanceador de cargas y el auto scaling sin embargo deseamos implementar docker en cada una de las máquinas para mayor escalabilidad.



**Paso 1.** Una vez conectado en la instancia, primero deshabilitamos el servicio de apache instalado en la AMI, con el comando **systemctl stop httpd** y **systemctl disable httpd**, luego de ejecutar estos dos comandos comprobamos que el servicio web este apagado con el comando **systemctl status httpd**.

```
[root@ip-10-0-4-81 nginx]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: inactive (dead)
   Docs: man:httpd.service(8)
```

**Paso 1.1.** instalamos el servidor web nginx, con el comando **yum install nginx**, luego lo iniciamos con el comando **systemctl start nginx** presionamos tecla enter e ingresamos este comando **systemctl enable nginx** que nos ayuda al inicio automáticamente por ultimo ingresamos el comando **systemctl status nginx** se debe visualizar activado.

```
[root@ip-10-0-4-81 nginx]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Active: active (running) since Sun 2024-12-08 02:08:53 UTC; 37min ago
     Process: 98070 ExecStartPre=/usr/bin/xm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
     Process: 98071 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 98072 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 98073 (nginx)
      Tasks: 2 (limit: 1111)
     Memory: 2.3M
        CPU: 34ms
   CGroup: /system.slice/nginx.service
           └─98073 "nginx: master process /usr/sbin/nginx"
             └─98074 "nginx: worker process"
```

## Paso 2. instalamos las herramientas docker

```
[root@ip-10-0-4-81 ec2-user]# yum install docker
Last metadata expiration check: 21:11:45 ago on Sat Dec 7 04:24:46 2024.
Dependencies resolved.
=====
Package           Arch      Version                               Repository      Size
=====
Installing:
docker            x86_64    25.0.6-1.amzn2023.0.2                amazonlinux    44 M
Installing dependencies:
containerd        x86_64    1.7.23-1.amzn2023.0.1                amazonlinux    36 M
iptables-libs    x86_64    1.8.8-3.amzn2023.0.2                amazonlinux    401 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
libnfnetlink     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    1.1.14-1.amzn2023.0.1                amazonlinux    3.2 M
=====
```

### Paso 2.1. Iniciamos docker con los siguientes comandos:

```
[root@ip-10-0-4-81 ec2-user]# systemctl start docker
[root@ip-10-0-4-81 ec2-user]# systemctl enable docker
```

### Paso 2.2. Descargamos del repositorio la imagen httpd para docker:

```
[root@ip-10-0-4-81 ec2-user]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
bc0965b23a04: Pull complete
d7ad38c6dd97: Pull complete
4f4fb700ef54: Pull complete
79b49624e34b: Pull complete
7d9f97915db2: Pull complete
9bd25d4f7b77: Pull complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

**Paso 2.3.** Revisamos que la imagen esta descargada con éxito

```
[root@ip-10-0-4-81 ec2-user]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
httpd latest 494b2b45fd74 4 months ago 147MB
```

**Paso 2.4.** Creamos tres carpetas para tomar de origen, en esta ubicación estarán los archivos index.html de la aplicación

```
drwxrwxrwx. 2 root root 24 Dec 8 01:59 tienda1
drwxrwxrwx. 2 root root 24 Dec 8 02:00 tienda2
drwxrwxrwx. 2 root root 24 Dec 8 02:00 tienda3
```

**Paso 2.5.** Creamos los contenedores con los siguientes comandos:

```
docker run -dit --name tienda1 --restart always -p 8080:80 -v
/tienda1:/usr/local/apache2/htdocs/ httpd
```

```
docker run -dit --name tienda2 --restart always -p 8082:80 -v
/tienda2:/usr/local/apache2/htdocs/ httpd
```

```
docker run -dit --name tienda3 --restart always -p 8083:80 -v
/tienda3:/usr/local/apache2/htdocs/ httpd
```

y luego ejecutamos **docker ps** para visualizar los contenedores

```
[root@ip-10-0-4-81 /]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bae240f25318 httpd "httpd-foreground" 39 seconds ago Up 37 seconds 0.0.0.0:8083->80/tcp, :::8083->80/tcp tienda3
34136clf86c8 httpd "httpd-foreground" About a minute ago Up About a minute 0.0.0.0:8082->80/tcp, :::8082->80/tcp tienda2
62001e863cfb httpd "httpd-foreground" 3 minutes ago Up 3 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp tienda1
```

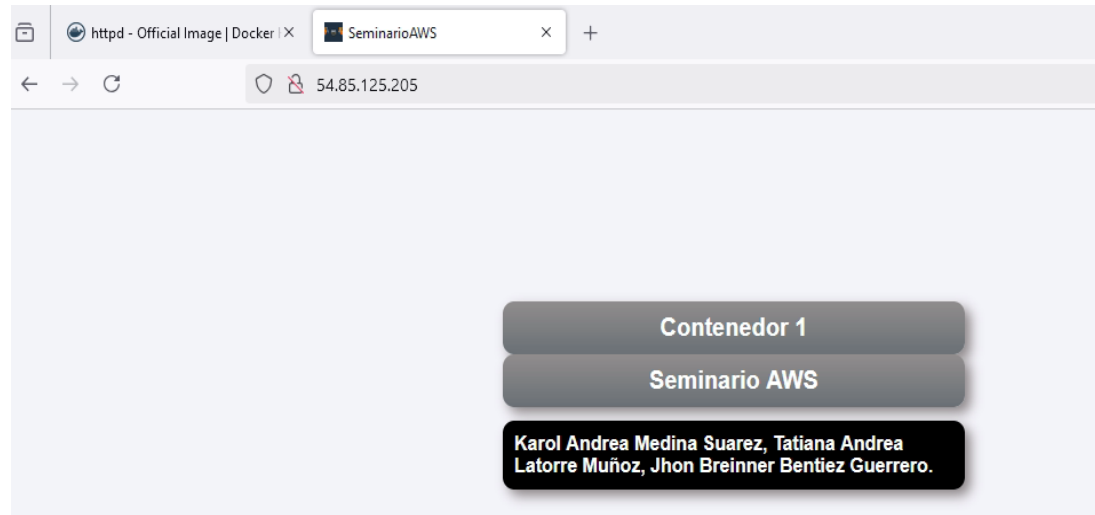
**Paso 3.** ingresamos a la ruta **/etc/nginx/nginx.config** y abrimos el archivo **nginx.config** con el comando **nano nginx.config** donde realizamos la configuración del servicio proxy para cuando llegue las peticiones ser reenviado a los contenedores

```
[root@ip-10-0-4-81 nginx]# cat nginx.conf
events {}

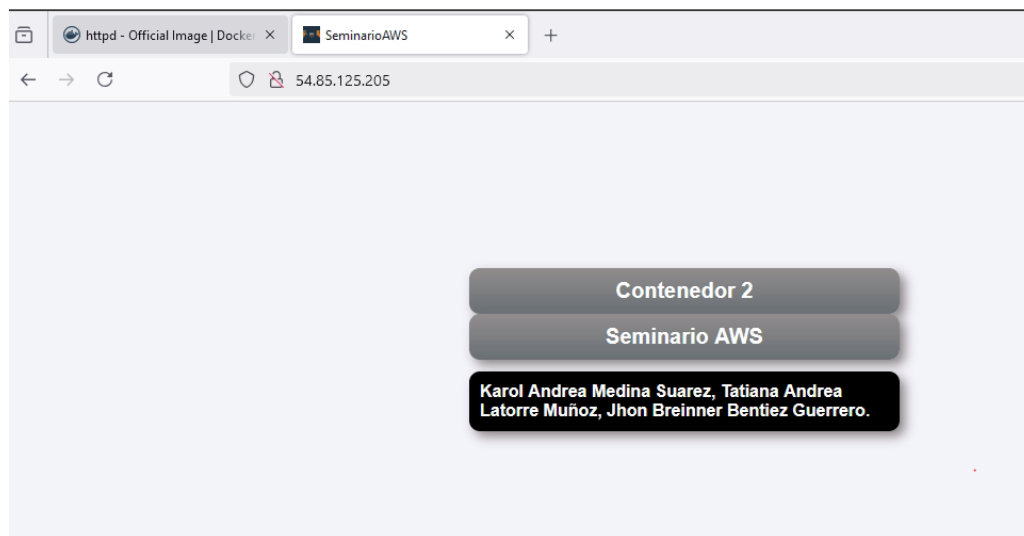
http {
    upstream backend {
        server localhost:8080;
        server localhost:8082;
        server localhost:8083;
    }
    server {
        listen 80;
        server_name nginx;
        location / {
            proxy_pass http://backend;
        }
    }
}
```

**Paso 4.** Por ultimo reiniciamos el nginx y cada petición que se hace a la instancia será enviada a los contenedores:

Primera petición:



Segunda petición:



Tercera petición:



**Paso 5.** Creamos una snapshots de la instancia implementada con docker, creamos una AMI nueva y la remplazamos en el auto scaling groups

**Launch template contents**  
Specify the details of your launch template version below. Leaving a field blank will result in the field not being included in the launch template version.

▼ **Application and OS Images (Amazon Machine Image)** Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

🔍 Search our full catalog including 1000s of application and OS images

AMI from catalog   Recents   **My AMIs**   Quick Start

Don't include in launch template    Owned by me    Shared with me

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

docker\_server AMI  
ami-0343cce98c967eb58  
2024-12-08T06:01:21.000Z   Virtualization: hvm   ENA enabled: true   Root device type: ebs

**Version**

2

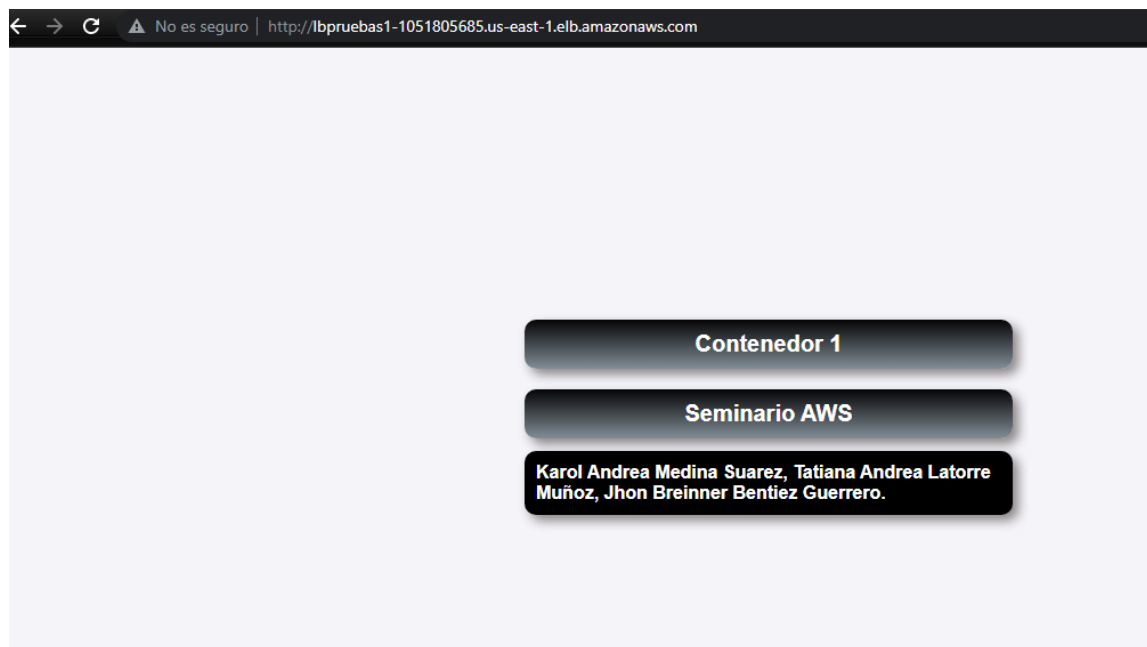
[Create a launch template version](#)

**Description**

-

**Launch template**  
LTremington  
lt-0a6543bc76e8e8ca2

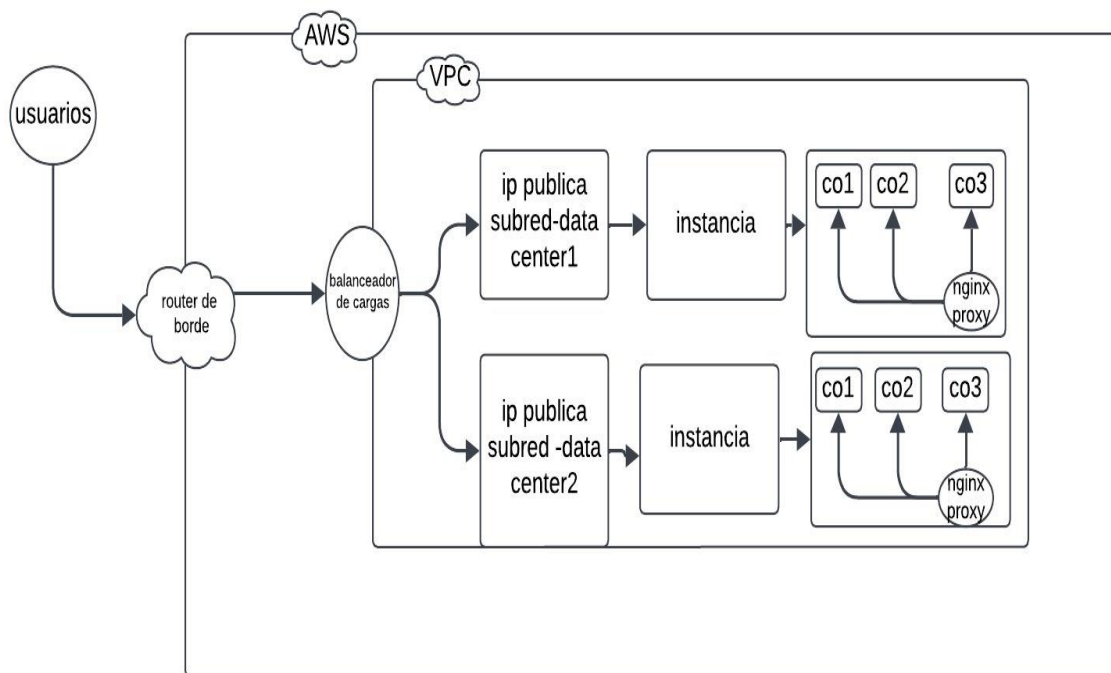
**Paso 6.** Funcionamiento con toda la infraestructura creada





**Vinculo YouTube-explicaicon visual**

[https://youtu.be/kBClj\\_-C-8M](https://youtu.be/kBClj_-C-8M)

**Diagrama de recursos usados y como se comunican entre ellos.**

## Conclusiones

La implementación de una arquitectura basada en contenedores en AWS, junto con el uso de balanceadores de carga, hace que sus aplicaciones lleguen a la meta de las empresas, si usted está buscando un modo más eficiente de escalar su aplicación que no requiera hardware físicos adicionales; estas formas de implementar la infraestructura permiten a los desarrolladores centrarse en importantes aspectos técnicos de codificación, despliegue y marketing. Es decir, los usos de micro-servicios en contenedores dejan claro que cada componente de la aplicación puede ser escalado sin que haya daños internos o causas externas.

Cuando se introduce en AWS una arquitectura con balanceadores de carga y contenedor, se consigue que las aplicaciones sean más flexibles y estén disponibles durante más tiempo. De esta forma la empresa fast food obtendrá seguridad de su software que estará completamente funcional. Sí una instancia deja de trabajar, el balanceador dirigirá automáticamente el tráfico hacia otras máquinas sanas que iniciaran automáticamente minimizando al mínimo tiempo de inactividad.

## Referencias

- (s.f.). Obtenido de <https://www.docker.com/blog/lxc-vs-docker/>
- Amazon web services . (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html>
- Amazon Web Services . (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: <https://docs.aws.amazon.com/ebs/latest/userguide/what-is-ebs.html>
- Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/EC2\\_GetStarted.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/EC2_GetStarted.html)
- Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/Instances.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/Instances.html)
- Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html>
- Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/AMIs.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html)
- Amazon web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/vpc/latest/userguide/configure-subnets.html](https://docs.aws.amazon.com/es_es/vpc/latest/userguide/configure-subnets.html)
- Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/concepts.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html)
- Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/AWSEC2/latest/UserGuide/Storage.html](https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/Storage.html)
- Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: [https://docs.aws.amazon.com/es\\_es/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html](https://docs.aws.amazon.com/es_es/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html)
- Amazon Web Services. (2 de 12 de 2021). *docs.aws.amazon.com*. Obtenido de docs.aws.amazon.com: <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>
- Perlow, J. (13 de junio de 2024). *www.docker.com*. Obtenido de [www.docker.com](https://www.docker.com): <https://www.docker.com/blog/lxc-vs-docker/>