

Implementación de infraestructura en la nube con AWS para RunFood

Corporación Universitaria Remington.

Ingeniería de sistemas

SEMINARIO AMAZON AWS

Karen Pérez Arango

Seminario de grado

2024.

Tabla de contenido

Resumen.....	3
Marco conceptual y contextual	4
Desarrollo e implementación del aprendizaje.....	5
Recursos de AWS para ejecutar instancias	5
Implementación de servicios AWS	7
Implementacion de docker para uso de contenedores y servidor web (nginx)	33
Tablas y Figuras	44
Conclusiones	45
Referencias.....	46

Resumen

El presente trabajo de grado describe la implementación de una arquitectura en la nube utilizando los servicios de Amazon Web Services (AWS) para apoyar el crecimiento tecnológico de la startup RunFood que conecta restaurantes con clientes mediante entregas rápidas. El objetivo principal es diseñar una infraestructura escalable, altamente disponible y con tiempos de respuesta mejorados. Se utilizaron servicios como EC2 para instancias de servidores virtuales, VPC para red privada, Load Balancer (LB) para distribución de tráfico, y Auto Scaling Group para ajuste dinámico de la capacidad. Además, se integraron tecnologías como Docker y NGINX para manejar microservicios y balanceo de cargas interno. Este proyecto demuestra la aplicación práctica de conceptos aprendidos en el seminario de AWS.

Palabras clave

AWS

Arquitectura en la nube

Auto Scaling

Docker

NGINX

Marco conceptual y contextual

AWS: Servicios de computación de Amazon ejecutados en la nube.

EC2: Servicio que permite desplegar instancias de servidores virtuales.

VPC: Agrupación de recursos que comparten un direccionamiento IP.

Load Balancer y Auto Scaling: Garantizan alta disponibilidad y respuesta eficiente ante variaciones en la demanda.

Docker y NGINX: Herramientas para la implementación de contenedores y gestión eficiente del tráfico mediante balanceo de cargas.

Key Pair: Clave utilizada para la autenticación segura de instancias mediante SSH.

Amazon Machine Image (AMI): Plantilla preconfigurada que contiene el sistema operativo, software y configuraciones necesarias y establecidas para lanzar nuevas instancias.

Target Group: Grupo de instancias asociadas al balanceador de carga para distribuir el tráfico entre ellas según las reglas configuradas.

Sanpshots: Copias de seguridad de los volúmenes que permiten crear nuevas instancias o restaurar datos.

La solución fue diseñada para la startup RunFood que busca conectar restaurantes con clientes de manera eficiente. Ante el aumento de usuarios y demanda, se requiere una infraestructura que asegure continuidad del servicio y tiempos de respuesta bajos. El proyecto pone en práctica habilidades aprendidas en el seminario de AWS, aplicando principios de diseño de soluciones escalables y seguras en la nube.

Desarrollo e implementación del aprendizaje

Recursos de AWS para ejecutar instancias

1. EC2

Recurso principal de amazon que permite ejecutar instancias es el servicio **EC2**. Este recurso permite desplegar y administrar instancias de servidores virtuales en la nube, este servicio maneja familia de instancias y tipos de instancias. Este recurso se usa normalmente para aplicaciones, servidores web y bases de datos.

2. Amazon Machine Image (AMI)

Una AMI es una plantilla preconfigurada que contiene lo necesario para lanzar una instancia:

Sistema operativo: Linux (Ubuntu, Amazon Linux, etc.) o Windows.

Software: Aplicaciones, frameworks y configuraciones personalizadas.

Puedes elegir entre:

AMIs prediseñadas por AWS o terceros.

AMIs personalizadas que tú mismo creas.

3. Grupos de Seguridad (Security Groups)

Actúan como un firewall virtual para las instancias EC2.

Controlan qué tráfico puede entrar o salir de la instancia.

Ejemplo de reglas entrantes: Permitir SSH (puerto 22) solo desde tu IP.

Ejemplo de reglas salientes: Permitir todas las conexiones hacia Internet.

Flexible y fácil de configurar. Puedes modificar las reglas en tiempo real.

4. Key Pairs

Claves de seguridad que permiten acceder a las instancias EC2 de forma segura.

Compuestas por:

Clave pública: Se almacena en la instancia.

Clave privada: La descargas al crear el par y la usas para conectarte vía SSH o RDP.

Sin la clave privada, no podrás acceder a la instancia.

5. Elastic Block Store (EBS)

Proporciona almacenamiento persistente para instancias EC2.

Cada instancia puede tener uno o más volúmenes EBS adjuntos.

Ejemplo: Tu instancia EC2 tiene un volumen EBS donde se almacena el sistema operativo y los datos.

Ventajas:

Los datos persisten incluso si la instancia se detiene.

Se pueden realizar respaldos mediante Snapshots.

6. Elastic IP

Es una dirección IP estática y pública que puedes asignar a una instancia EC2.

Ideal para casos en los que necesitas una dirección IP fija, como:

Servidores web accesibles públicamente.

Aplicaciones críticas que no deben cambiar de IP al reiniciar.

7. Amazon Virtual Private Cloud (VPC)

Crea una red virtual donde se ejecutan tus instancias.

Ofrece control sobre la configuración de red:

Subredes: Redes internas dentro de tu VPC.

Tablas de enrutamiento: Definen cómo el tráfico fluye dentro y fuera de la VPC.

Internet Gateway: Permite acceso a Internet.

NAT Gateway: Da acceso a Internet a instancias en subredes privadas.

8. Auto Scaling Groups (ASG)

Permiten escalar automáticamente la cantidad de instancias EC2 según la demanda:

Ejemplo: Aumentar instancias cuando hay más tráfico y reducir las cuando baja.

Configuración básica:

Minimizar costos: Solo usas los recursos necesarios.

Alta disponibilidad: Se asegura de que siempre haya instancias funcionando.

9. Elastic Load Balancer (ELB)

Es un servicio que distribuye el tráfico de red entre varias instancias EC2.

Ayuda a mejorar la disponibilidad y redundancia de tu aplicación.

Tipos de balanceadores:

Application Load Balancer: Para tráfico HTTP/HTTPS, ideal para aplicaciones web.

Network Load Balancer: Para tráfico de red de baja latencia.

Gateway Load Balancer: Para balancear tráfico hacia firewalls virtuales.

10. AWS Systems Manager

Es un servicio que simplifica la administración y operación de instancias EC2.

Funciones clave:

Patch Manager: Instala actualizaciones de seguridad.

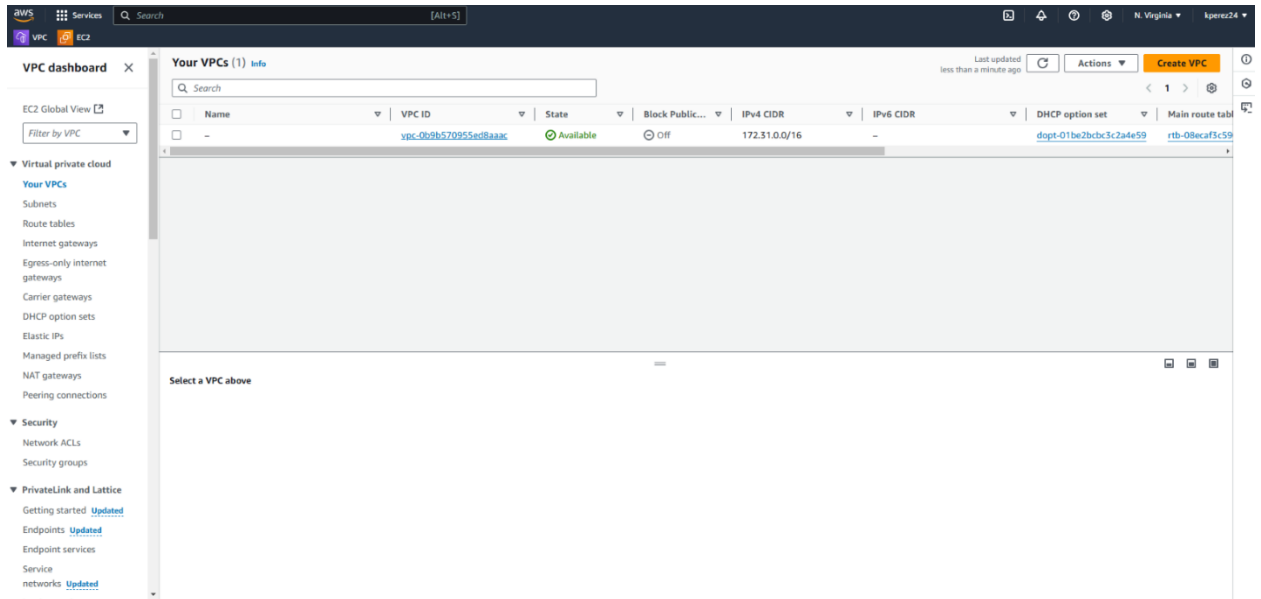
Session Manager: Permite acceso remoto seguro sin usar claves SSH.

Run Command: Ejecuta comandos o scripts en múltiples instancias simultáneamente.

Implementación de servicios AWS

VPC

Para crear un VPC ingresamos al servicio de VPC en AWS y seleccionamos la opción de tus VPCS



Fuente: elaboración propia.

Al darle crear vemos el menú de creación, seleccionamos VPC y más y luego ponemos un nombre, la demás información la dejamos como viene por defecto y damos crear.

Configuración de VPC

Recursos para crear **Información**
Crea únicamente el recurso VPC o la VPC y otros recursos de red.

Solo VPC VPC y más

Name tag auto-generation Info
Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.
 Auto-generate
SeminaríoRemington

IP4 CIDR block Info
Determine the starting IP and the size of your VPC using CIDR notation.
10.0.0.0/16 65,536 IPs
CIDR block size must be between /16 and /28.

IP6 CIDR block Info
 No IPv6 CIDR block
 Amazon-provided IPv6 CIDR block

Tenancy Info
Default

Number of Availability Zones (AZs) Info
Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.
1 2 3
▶ Customize AZs

Number of public subnets Info
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.
0 2

Number of private subnets Info
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.
0 2 4
▶ Customize subnets CIDR blocks

NAT gateways (5) Info
Choose the number of Availability Zones (AZs) in which to create NAT gateways. Note that there is a charge for each NAT gateway.
None In 1 AZ 1 per AZ

VPC endpoints Info
Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.
None S3 Gateway

DNS options Info
 Enable DNS hostnames
 Enable DNS resolution

▶ Additional tags

Cancelar Código de vista previa

Preview

VPC Show details
Your AWS virtual network
SeminaríoRemington-vpc

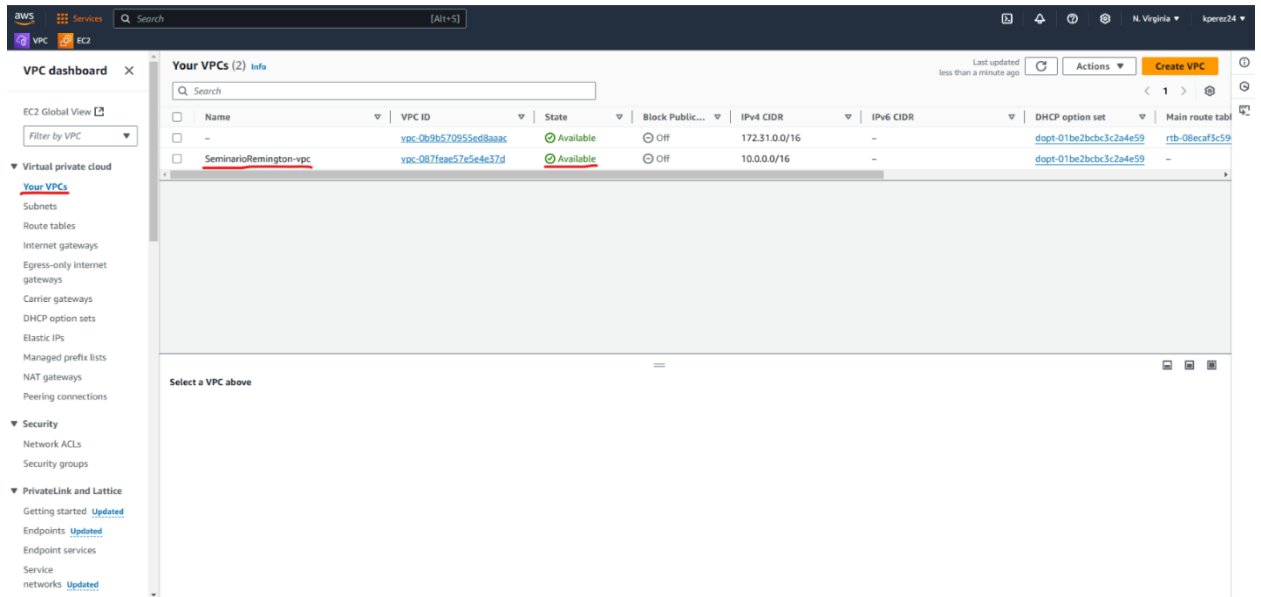
Subnets (4)
Subnets within this VPC
us-east-1a
SeminaríoRemington-subnet-
SeminaríoRemington-subnet-
us-east-1b
SeminaríoRemington-subnet-
SeminaríoRemington-subnet-

Route tables (5)
Route network traffic to resources
SeminaríoRemington-rtb-public
SeminaríoRemington-rtb-privato 1-us-
SeminaríoRemington-rtb-privato2-us-

Network connections (2)
Connections to other networks
SeminaríoRemington-igw
SeminaríoRemington-vpc-e3

Fuente: elaboración propia.

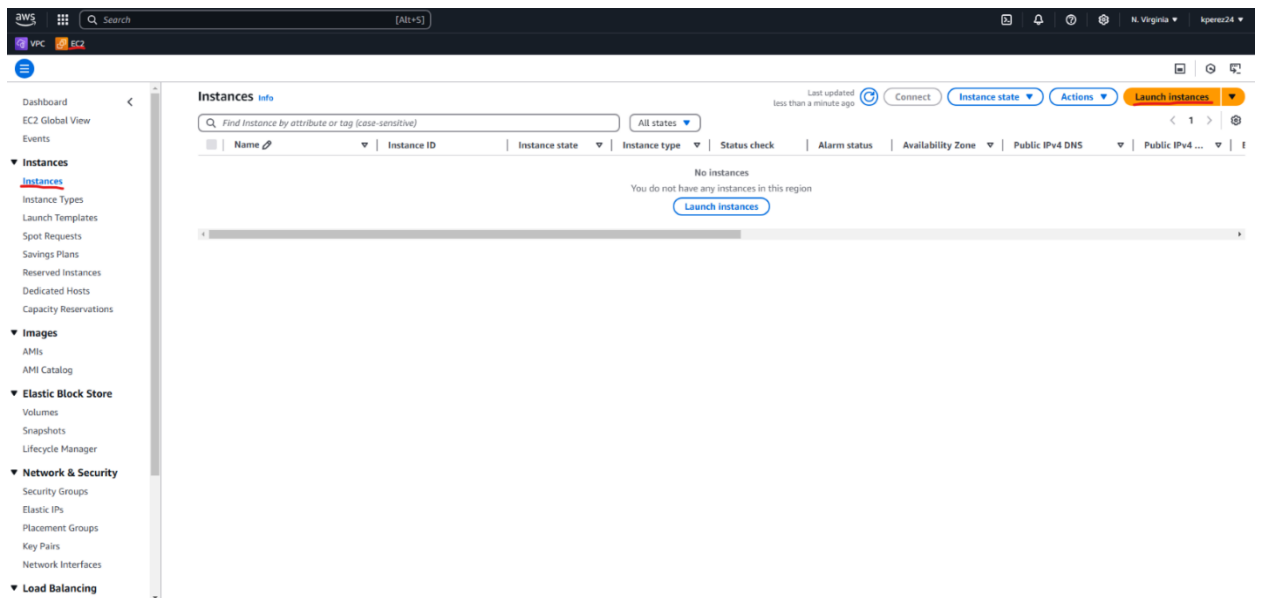
Verificamos en el apartado de tus VPCs y verificamos que se haya creado sin problemas y su estado esté disponible



Fuente: elaboración propia.

Instancias (EC2)

Para crear instancias ingresamos al servicio EC2, Instancias y damos clic en lanzar instancia



Fuente: elaboración propia.

En el menú de lanzar instancia ponemos un nombre a la instancia y seleccionamos la AMI (sistema operativo) y el tipo de instancia, en este caso dejamos las opciones que vienen por defecto.

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name: [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 | **Quick Start**

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Linux | Debian

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

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-0453ec754f44f9a4a (64-bit (x86), uefi-preferred) / ami-0e083e7a78a23014e (64-bit (ARM), uefi)
Virtualization from: EMK enabled: true - Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20241121.0.x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Username
64-bit (x86)	uefi-preferred	ami-0453ec754f44f9a4a	ec2-user Verified provider

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

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

Summary

Number of instances | [Info](#)

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-0453ec754f44f9a4a

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#) [Launch instance](#) [Preview code](#)

Fuente: elaboración propia.

En la opción de key pair creamos una nueva

EC2

[Launch an instance](#)

64-bit (x86) | uefi-preferred | ami-0453ec754f44f9a4a | ec2-user | Verified provider

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

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 - *required*

[Create new key pair](#)

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

Network | [Info](#)

vpc-0b9b570955ed8aaac

Subnet | [Info](#)

No preference (Default subnet in any availability zone)

Summary

Number of instances | [Info](#)

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-0453ec754f44f9a4a

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet.

[Cancel](#) [Launch instance](#) [Preview code](#)

Fuente: elaboración propia.

Ponemos un nombre y seleccionamos el tipo de formato para la key pair, en este caso .ppk y damos clic en crear

The screenshot shows a 'Create key pair' dialog box with the following fields and options:

- Key pair name:** A text input field containing 'KPServerSeminaro1'. Below it, a note states: 'The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.'
- Key pair type:** Two radio button options:
 - RSA**
RSA encrypted private and public key pair
 - ED25519**
ED25519 encrypted private and public key pair
- Private key file format:** Two radio button options:
 - .pem**
For use with OpenSSH
 - .ppk**
For use with PuTTY

A warning box at the bottom states: 'When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)'

At the bottom right, there are two buttons: 'Cancel' and 'Create key pair'.

Fuente: elaboración propia.

Luego de tener la key pair continuamos con el network setting, primero seleccionamos el VPC creado anteriormente, luego continuamos con la Subnet, en este caso seleccionamos la publica1 y por ultimo activamos la opción de auto asignar la IP publica y agregamos la regla de seguridad de grupo habilitando el puerto 80 para que este pueda ser accedido desde cualquier equipo con acceso a internet.

▼ **Network settings** [Info](#)

VPC - required | [Info](#)

vpc-087feae57e5e4e37d (SeminarioRemington-vpc)
10.0.0.0/16

Subnet | [Info](#)

subnet-0b9e8f79d4134320c SeminarioRemington-subnet-public1-us-east-1a
VPC: vpc-087feae57e5e4e37d Owner: 254151005565 Availability Zone: us-east-1a
Zone type: Availability Zone IP addresses available: 4090 CIDR: 10.0.0.0/20

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

Enable

Additional charges apply when outside of free tier allowance

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.

Create security group Select existing security group

Security group name - required

launch-wizard-3

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _-./()!@#%&*[]\$*

Description - required | [Info](#)

launch-wizard-3 created 2024-12-04T04:17:37.715Z

Inbound Security Group Rules

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

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

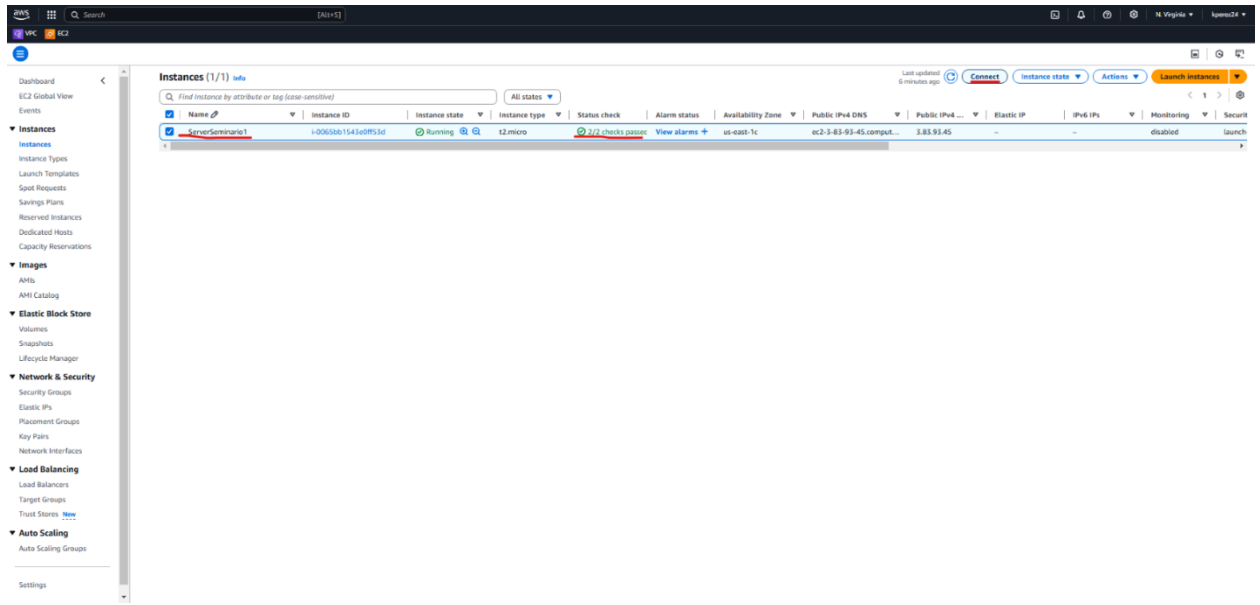
▼ Security group rule 2 (TCP, 80, 0.0.0.0/0) [Remove](#)

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	80
Source type Info	Source Info	Description - optional Info
Custom	0.0.0.0/0	e.g. SSH for admin desktop

Fuente: elaboración propia.

El apartado de configuración del almacenamiento (Configure Storage) se deja así por defecto. Y lanzamos la instancia.

Verificamos en instancias que este creada correctamente y con estado, la seleccionamos, y le damos en conectar



Fuente: elaboración propia.

Copiamos el nombre que nos genera amazon para conectarnos

Connect to instance [Info](#)

Connect to your instance i-0065bb1543e0ff53d (ServerSeminario1) using any of these options

[EC2 Instance Connect](#) | [Session Manager](#) | [SSH client](#) | [EC2 serial console](#)

Instance ID

[i-0065bb1543e0ff53d](#) (ServerSeminario1)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `KPServerSeminario1.pem`
3. Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 "KPServerSeminario1.pem"`
4. Connect to your instance using its Public DNS:
`ec2-3-83-93-45.compute-1.amazonaws.com`

Example:

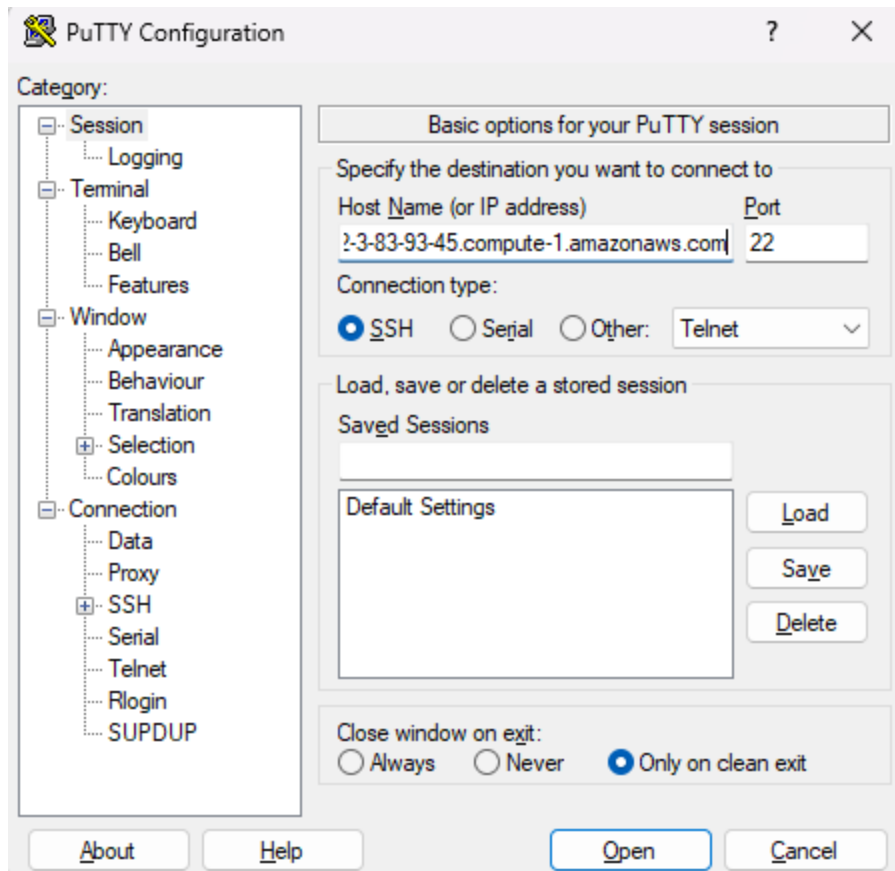
```
ssh -i "KPServerSeminario1.pem" ec2-user@ec2-3-83-93-45.compute-1.amazonaws.com
```

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.

[Cancel](#)

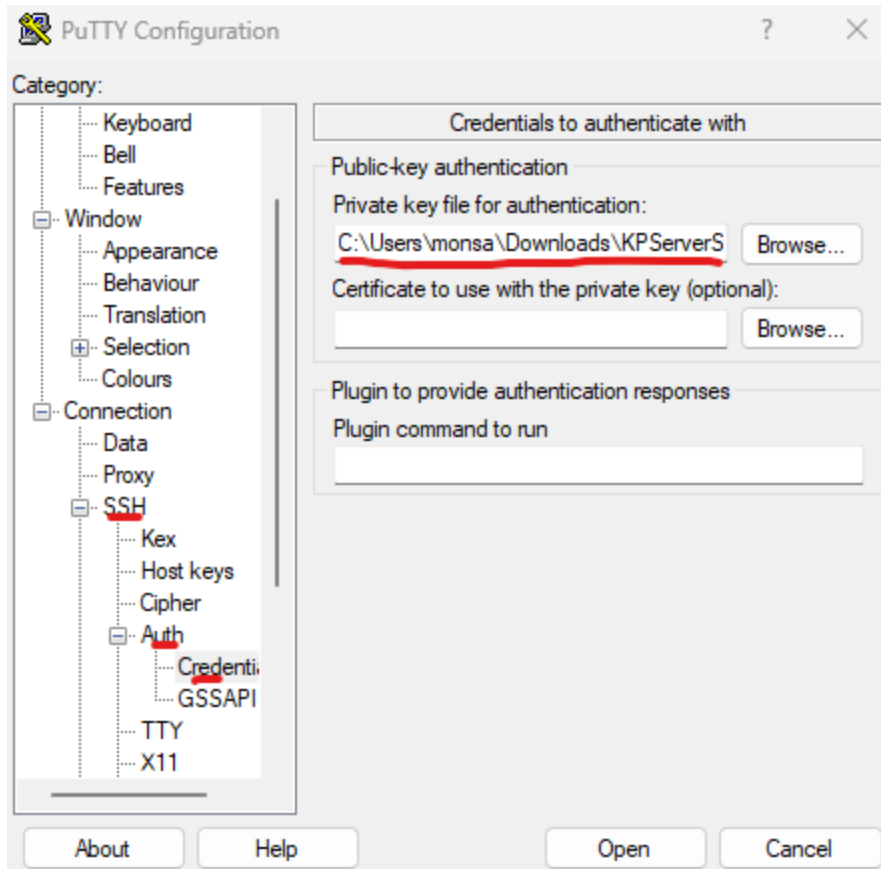
Fuente: elaboración propia.

Ponemos el nombre en la aplicacion PuTTY



Fuente: elaboración propia.

Luego damos clic en SSH -> Auth-> Credentials y subimos el archivo ppk que generamos para la key pair y damos clic en abrir



Fuente: elaboración propia.

En la terminal accedemos como administradores (comando: `sudo su`) e instalamos apache (comando `yum install httpd`). Verificamos si el servicio está en funcionamiento (comando `systemctl status httpd`) si esta inactivo lo activamos (comando `systemctl start httpd`)


```

Preparing      :                               1/12
Installing     : apr-1.7.2-2.amzn2023.0.2.x86_64 1/12
Installing     : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 2/12
Installing     : apr-util-1.6.3-1.amzn2023.0.1.x86_64 3/12
Installing     : mailcap-2.1.49-3.amzn2023.0.3.noarch 4/12
Installing     : httpd-tools-2.4.62-1.amzn2023.x86_64 5/12
Installing     : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 6/12
Running scriptlet: httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Installing     : httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Installing     : httpd-core-2.4.62-1.amzn2023.x86_64 8/12
Installing     : mod_http2-2.0.27-1.amzn2023.0.3.x86_64 9/12
Installing     : mod_lua-2.4.62-1.amzn2023.x86_64 10/12
Installing     : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 11/12
Installing     : httpd-2.4.62-1.amzn2023.x86_64 12/12
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64 12/12
Verifying     : apr-1.7.2-2.amzn2023.0.2.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
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-172-31-86-122 ec2-user]# systemctl start httpd
[root@ip-172-31-86-122 ec2-user]# █

```

Fuente: elaboración propia.

Luego de realizar estos pasos regresamos a la instancia y copiamos la IP pública

Instance summary for i-0065bb1543e0ff53d (ServerSeminar10)

Updated less than a minute ago

Instance ID: i-0065bb1543e0ff53d
Public IPv4 address: 3.83.93.45 | [open address](#)
Instance state: Running
Private IP DNS name (IPv4 only): ip-172-31-86-122.ec2.internal
Instance type: t2.micro
VPC ID: vpc-0b96570955e08aac
Subnet ID: subnet-043940d438f95ca24
Instance ARN: aws:ec2:us-east-1:254151005565:instance/i-0065bb1543e0ff53d

Private IPv4 addresses: 172.31.86.122
Public IPv4 DNS: ec2-3-83-93-45.compute-1.amazonaws.com | [open address](#)

Elastic IP addresses: -
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | [Learn more](#)
Auto Scaling Group name: -
Managed: false

Instance details:
AMI ID: ami-0452ac75af4f9a4a
AMI name: a2023-ami-2023.6.20241121.0-kernel-6.1-h86_64
Step protection: Disabled
Instance auto-recovery: Default
AMI Launch index: 0
Credits specification: standard
Usage operation: -

Monitoring: disabled
Allowed image: -
Launch time: Tue Dec 03 2024 21:22:17 GMT-0500 (hora estándar de Colombia) (29 minutos)
Lifecycle: normal
Key pair assigned at launch: KPServerSeminar10
Kernel ID: -
RAM disk ID: -

Platform details: Linux/UNIX
Termination protection: Disabled
AMI location: amazon/e2023-ami-2023.6.20241121.0-kernel-6.1-h86_64
Step-hibernate behavior: Disabled
State transition reason: -
State transition message: -
Owner: -

Fuente: elaboración propia.

Esta la copiamos en el navegador web y la ejecutamos, así verificamos su correcto funcionamiento.

Balanceador de carga (Load balancing)

Para crear un balanceador de cargas inicialmente ponemos un nombre, especificamos que el tipo de ip a asignar es la IPv4 y su esquema Internet-facing.

Continuamos en la sección de Network mapping asignando la VPC creada anteriormente, luego en las zonas disponibles seleccionamos las subnet publicas.

Create Application Load Balancer

Basic configuration

Load balancer name
 Name must be unique within your AWS account and can't be changed after the load balancer is created.
 LBSeminar10Remington

Scheme
 Scheme can't be changed after the load balancer is created.
 Internet-facing
 Internal

Load balancer IP address type
 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 types. Public IPv4 addresses have an additional cost.
 IPv4
 Dualstack
 Dualstack without public IPv4

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

VPC
 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](#).

Mappings
 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-az6)
 Subnet: subnet-0b9e8f79d4134320c
 Seminar10Remington-subnet-public1-us-east-1a
 us-east-1b (use1-az1)
 Subnet: subnet-021c7556a4d9ac03
 Seminar10Remington-subnet-public2-us-east-1b

Fuente: elaboración propia.

En la sección de grupo de seguridad ingresamos los datos básicos, nombre, descripción y seleccionamos el VPC a asociar este grupo.

Continuamos las reglas de entrada y salida, registrando los puertos 80 en ambos y 22 en la regla de entrada.

Fuente: elaboración propia.

Después de crear el grupo de seguridad agregamos este al balanceador de carga que estamos creando.

Fuente: elaboración propia.

Continuamos creando el *Target Group*, asignamos el nombre, el protocolo HTTP y puerto 80, especificamos el tipo de IPv4 y por último que el VPC sea el que creamos inicialmente.

Target group name

TGBalancedorCarga1

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

Protocol : Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation

HTTP 80

1-65535

IP address type

Only targets with the indicated IP address type can be registered to this target group.

IPv4

Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6

Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

SeminarioRemington-vpc

vpc-0877eae57e5e4e37d

IPv4 VPC CIDR: 10.0.0.0/16

Protocol version

HTTP1

Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

HTTP2

Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

gRPC

Send requests to targets using gRPC. Supported when the request protocol is gRPC.

Fuente: elaboración propia.

Continuamos con el registro de las instancias, seleccionamos las 2 que creamos inicialmente

Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (2)

Instance ID	Name	State	Security groups	Zone	Private IPv4 address	Subnet ID	Launch time
i-0c37852c48f40c2db	ServerSeminario1	Running	launch-wizard-3	us-east-1b	10.0.19.91	subnet-021c75560a499ac03	December 3, 2024, 23:35 (UTC-05:00)
i-089a9b88e566655d	ServerSeminario2	Running	launch-wizard-2	us-east-1a	10.0.0.151	subnet-069e8f79d4134320c	December 3, 2024, 23:12 (UTC-05:00)

0 selected

Ports for the selected instances

Ports for routing traffic to the selected instances.

80

1-65535 (separate multiple ports with comma)

Include as pending below

2 selections are now pending below. Include more or register targets when ready.

Review targets

Targets (2)

Instance ID	Name	Port	State	Security groups	Zone	Private IPv4 address	Subnet ID	Launch time
i-0c37852c48f40c2db	ServerSeminario1	80	Running	launch-wizard-3	us-east-1b	10.0.19.91	subnet-021c75560a499ac03	December 3, 2024, 23:35 (UTC-05:00)
i-089a9b88e566655d	ServerSeminario2	80	Running	launch-wizard-2	us-east-1a	10.0.0.151	subnet-069e8f79d4134320c	December 3, 2024, 23:12 (UTC-05:00)

2 pending

Cancel Previous **Create target group**

Fuente: elaboración propia.

Agregamos el Target Group creado a la configuración del balanceador de cargas.

Listeners and routing Info

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener **HTTP:80** Remove

Protocol: HTTP : Port: 80 (1-65535)

Default action: **Info**
 Forward to: TGBalanceadorCarga1 HTTP ⓘ
Target type: Instance, IPv4
[Create target group](#)

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](#)

Fuente: elaboración propia.

Por último verificamos el resumen de la creación del balanceador de cargas y en caso tal de estar todo correcto, lo creamos.

✔ Successfully created load balancer: **LBSeminariorRemington**
 It might take a few minutes for your load balancer to fully set up and route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

LBSeminariorRemington ⓘ Actions

▼ **Details**

Load balancer type Application	Status ⓘ Provisioning	VPC vpc-087feac57e5e4e37d	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-021c75560a4d9ac03 us-east-1b (use1-az1) subnet-0b9e8f79d4134320c us-east-1a (use1-az6)	Date created December 4, 2024, 00:42 (UTC-05:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:254151005565:loadbalancer/app/LBSeminariorRemington/d2e217a8ee5f9ce7		DNS name <small>Info</small> LBSeminariorRemington-1976011885.us-east-1.elb.amazonaws.com (A Record)	

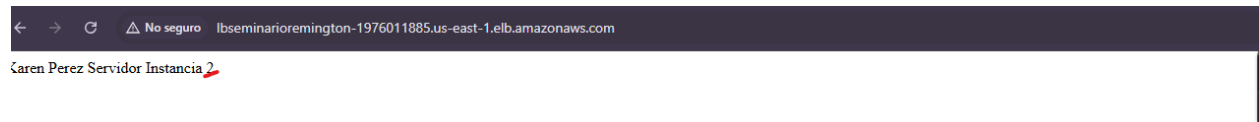
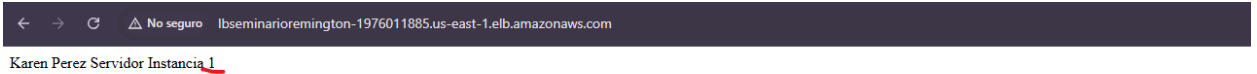
Listeners and rules (1) Info ⓘ Manage rules Manage listener Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

<input type="checkbox"/>	Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
<input type="checkbox"/>	HTTP:80	Forward to target group <ul style="list-style-type: none"> • TGBalanceadorCarga1 1 (100%) • Target group stickiness: Off 	1 rule	ARN	Not applicable	Not applicable	Not applicable

Fuente: elaboración propia.

Copiamos el DNS Name y ejecutamos en el navegador para verificar que si este funcionando el balanceador de carga



Fuente: elaboración propia.

En estas imágenes podemos ver como intercala las instancias que creamos anteriormente, por lo cual determinamos que quedo creado correctamente.

Procedemos a crear un *Auto Scaling Group*, ingresamos el nombre del grupo de auto scaling, y luego seleccionamos el Launch Template, en caso tal que no esté creado procedemos a darle en “Create a launch template”.

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.

ASGSeminarioRemington1

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.

Select a launch template

[Create a launch template](#)

Cancel

Next

Fuente: elaboración propia.

Al momento de crear el Launch Template ingresamos el nombre y la descripción para poder identificar esta plantilla

Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

Launch template name and description

Launch template name - *required*

LTSeminarioRemington1

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

Template version description

LT1

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

Fuente: elaboración propia.

En la sección de aplicaciones e imágenes de sistemas operativo que ya tenemos creado en mis AMIs. Luego seleccionamos la instancia gratuita en este caso t2.micro y el Key Pair usamos el “KPServerSeminario1”.

▼ 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)

AMI ServerSeminario1
ami-073452c977e96b242
2024-12-05T01:04:54.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Description
AMI server seminario 1

Architecture	AMI ID
x86_64	ami-073452c977e96b242

▼ 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

KPServerSeminario1 [Create new key pair](#)

Fuente: elaboración propia.

Por ultimo en Network Setting no incluimos una subnet y luego seleccionamos un grupo existente de seguridad “SGBalanceadorCarga1” y finalmente creamos el Launch Template.

▼ 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 ▼ [Compare security group rules](#)

SGBalanceadorCarga1_sg-0cff95a32b9d2d2ad ×
 VPC: vpc-087feae57e5e4e37d

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

▼ Advanced network configuration

Network interface 1 [Remove](#)

Device index [Info](#)

0

Network interface [Info](#)

New interface ▼

Existing network interfaces are not recommended when creating a template for auto-scaling.

Subnet [Info](#)

Don't include in launch template
Not applicable for EC2 Auto Scaling

Security groups [Info](#)

Select security groups ▼ [Show all selected \(1\)](#)

Auto-assign public IP [Info](#)

Don't include in launch template ▼

Primary IP [Info](#)

Not applicable for EC2 Auto Scaling

Secondary IP [Info](#)

Don't include in launch template ▼
Not applicable for EC2 Auto Scaling

IPv6 IPs [Info](#)

Don't include in launch template ▼
Not applicable for EC2 Auto Scaling

IPv4 Prefixes [Info](#)

Don't include in launch template ▼
The selected instance type does not support IPv4 prefixes.

IPv6 Prefixes [Info](#)

Don't include in launch template ▼
The selected instance type does not support IPv6 prefixes.

Assign Primary IPv6 IP [Info](#)

Don't include in launch template ▼

Delete on termination [Info](#)

Don't include in launch template ▼

Interface type [Info](#)

Don't include in launch template ▼

Network card index [Info](#)

Don't include in launch template ▼
The selected instance type does not support multiple network cards.

ENA Express [Info](#)

Don't include in launch template ▼
The selected instance type does not support ENA Express.

ENA Express UDP [Info](#)

Don't include in launch template ▼
The selected instance type does not support ENA Express.

Idle connection tracking timeout [Info](#)

Enable
Idle connection tracking timeout is only supported on Nitro instances.

[Add network interface](#)

Fuente: elaboración propia.

Aquí verificamos que se haya creado correctamente.

Launch Templates (1) [Info](#)

Search

<input type="checkbox"/>	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By	Managed	Operator
<input type="checkbox"/>	lt-0cbe6035d1dcc7051	LTSeminarioRemington1	1	1	2024-12-04T23:38:53.000Z	arn:aws:iam::254151005565:root	false	-

Fuente: elaboración propia.

Continuamos con la selección del launch template recién creado y luego asignamos la versión default.

- Step 1
- Choose launch template**
- Step 2
- Step 3 - optional
- Step 4 - optional
- Step 5 - optional
- Step 6 - optional
- Step 7
- Review

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.

ASGSeminarioRemington1

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.

LTSeminariorRemington1

[Create a launch template](#)

Version

Default (1)

[Create a launch template version](#)

Description

LT1

AMI ID

ami-0453cc754f44f9a4a

Key pair name

ServerSeminario1

Launch template

LTSeminariorRemington1
lt-0cbe6035d1dc7051

Security groups

-

Security group IDs

sg-0cf95a32b9d2d2ad

Instance type

t2.micro

Request Spot Instances

No

Additional details

Storage (volumes)

-

Date created

Wed Dec 04 2024 18:38:53 GMT-0500 (hora estándar de Colombia)

[Cancel](#)

[Next](#)

Fuente: elaboración propia.

En el paso dos seleccionamos la VPC y luego las zonas y subnets publicas que tengamos creadas.

- Step 1
● Choose launch template
- Step 2
● **Choose instance launch options**
- Step 3 - optional
● Integrate with other services
- Step 4 - optional
● Configure group size and scaling
- Step 5 - optional
● Add notifications
- Step 6 - optional
● Add tags
- Step 7
● Review

Choose instance launch options [info](#)

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

Instance type requirements [info](#)

[Override launch template](#)

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
LTSeminariorRemington1 lt-0cbe6035d1cc7051	Default	LT1

Instance type
t2.micro

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-087feac57e5e4e37d (SeminariorRemington-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-0b9e8f79d4134320c (SeminariorRemington-subnet-public1-us-east-1a)
10.0.0.0/20

us-east-1b | subnet-021c75560a4d9ac03 (SeminariorRemington-subnet-public2-us-east-1b)
10.0.0.0/20

[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](#)

Fuente: elaboración propia.

Continuamos con el paso 3 integrando otros servicios, seleccionamos el balanceador de carga creado previamente, los demás campos los dejamos por default como vienen preseleccionados.

Integrate with other services - *optional* [Info](#)

Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift. You can also customize health check replacements and monitoring.

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



TGBalancedorCarga1 | HTTP
Application Load Balancer: LBseminarioRemington

VPC Lattice integration options [Info](#)

To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach

No VPC Lattice service

VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service

Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

[Create new VPC Lattice service](#)

Application Recovery Controller (ARC) zonal shift - *new* [Info](#)

During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.

Enable zonal shift

New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.

Fuente: elaboración propia.

Continuamos con el paso 4 configurando el tamaño del grupo de Scaling. Asignamos máximos y mínimos en los campos de capacidad.

Step 1
● Choose launch template

Step 2
● Choose instance launch options

Step 3 - optional
● Integrate with other services

Step 4 - optional
● **Configure group size and scaling**

Step 5 - optional
● Add notifications

Step 6 - optional
● Add tags

Step 7
● Review

Configure group size and scaling - optional [info](#)

Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

Group size [info](#)

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances) ▾

Desired capacity

Specify your group size.

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**

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

Automatic scaling - optional

Choose whether to use a **target tracking policy** [info](#)

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

Fuente: elaboración propia.

Finalmente continuamos los demás pasos dejando sus campos seleccionados por default y esperamos a que se actualice la capacidad del Auto Scaling Group.

Auto Scaling groups (1) [info](#)

🔍 Search your Auto Scaling groups

<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input type="checkbox"/>	ASGSeminarioRemington1	LTSeminarioRemington1 Version Default	0	Updating capacity...	3	2	5	us-east-1a, us-east-1b

Fuente: elaboración propia.

Luego de esto nuestro Target Group debe visualizarse así

TGBalanceadorCarga1

Actions

Details

arn:aws:elasticloadbalancing:us-east-1:254151005565:targetgroup/TGBalanceadorCarga1/68e19689b4221efb

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-087feae57e5e4e37d
IP address type IPv4	Load balancer LBSeminarioRemington		

5 Total targets

5 Healthy
0 Anomalous

0 Unhealthy

0 Unused

0 Initial

0 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 (5) [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.

Filter targets

<input type="checkbox"/>	Instance ID	Name	Port	Zone	Health status	Health status details	Adminis...	Overrid...	Launch...
<input type="checkbox"/>	i-0b6b209c2372b72fb		80	us-east-1b (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-062ae5377195cb2ba		80	us-east-1b (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-02dbe85049ce70f61		80	us-east-1a (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-089dd9b88e56b655d	ServerSeminar...	80	us-east-1a (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-0c37852c48f40e2db	ServerSeminar...	80	us-east-1b (us...)	Healthy	-	No override..	No overrid...	December...

Fuente: elaboración propia.

Ahora verificamos si le damos stop a una instancia el funcionamiento del autoscaling, podemos ver que ahora nos muestra 4 healthy

The screenshot displays the AWS Management Console interface for a Target Group named "TGBalanceadorCarga1". The console shows the following details:

- Details:** Target type: Instance; Protocol: Port: HTTP:80; Protocol version: HTTP1; VPC: vpc-087f0ae77e5ed4e37d.
- Summary:** 5 Total targets; 4 Healthy; 0 Anomalous; 0 Unhealthy; 0 Unused; 1 Initial; 0 Draining.
- Registered targets (5):** A table listing individual targets with columns for Instance ID, Name, Port, Zone, Health status, Health status details, Adminis..., Overrid..., and Launch... The table shows five targets, all of which are currently in a "Healthy" state.

Fuente: elaboración propia.

En respuesta a esto vemos que vuelve a iniciar la instancia y la marca como healthy nuevamente.

TGBalanceadorCarga1

Actions ▾

Details
arn:aws:elasticloadbalancing:us-east-1:254151005565:targetgroup/TGBalanceadorCarga1/68e19689b4221efb

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-087fae57e5e4c37d
IP address type IPv4	Load balancer LBSeminariorRemington		

5 Total targets

5 Healthy
0 Anomalous

0 Unhealthy

0 Unused

0 Initial

0 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 (5) [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.

Filter targets

<input type="checkbox"/>	Instance ID	Name	Port	Zone	Health status	Health status details	Adminis...	Override...	Launch...
<input type="checkbox"/>	i-068a5b6155587bab3		80	us-east-1b (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-0c37852c48f40e2db	ServerSeminar...	80	us-east-1b (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-0b6b209c2372b72fb		80	us-east-1b (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-02dbe85049ce70f61		80	us-east-1a (us...)	Healthy	-	No override..	No overrid...	December...
<input type="checkbox"/>	i-089dd9b88e56b655d	ServerSeminar...	80	us-east-1a (us...)	Healthy	-	No override..	No overrid...	December...

Implementación de docker para uso de contenedores y servidor web (nginx)

Docker

En las instancias que ya tenemos creadas vamos a realizar la implementación de Docker para hacer uso de microservicios.

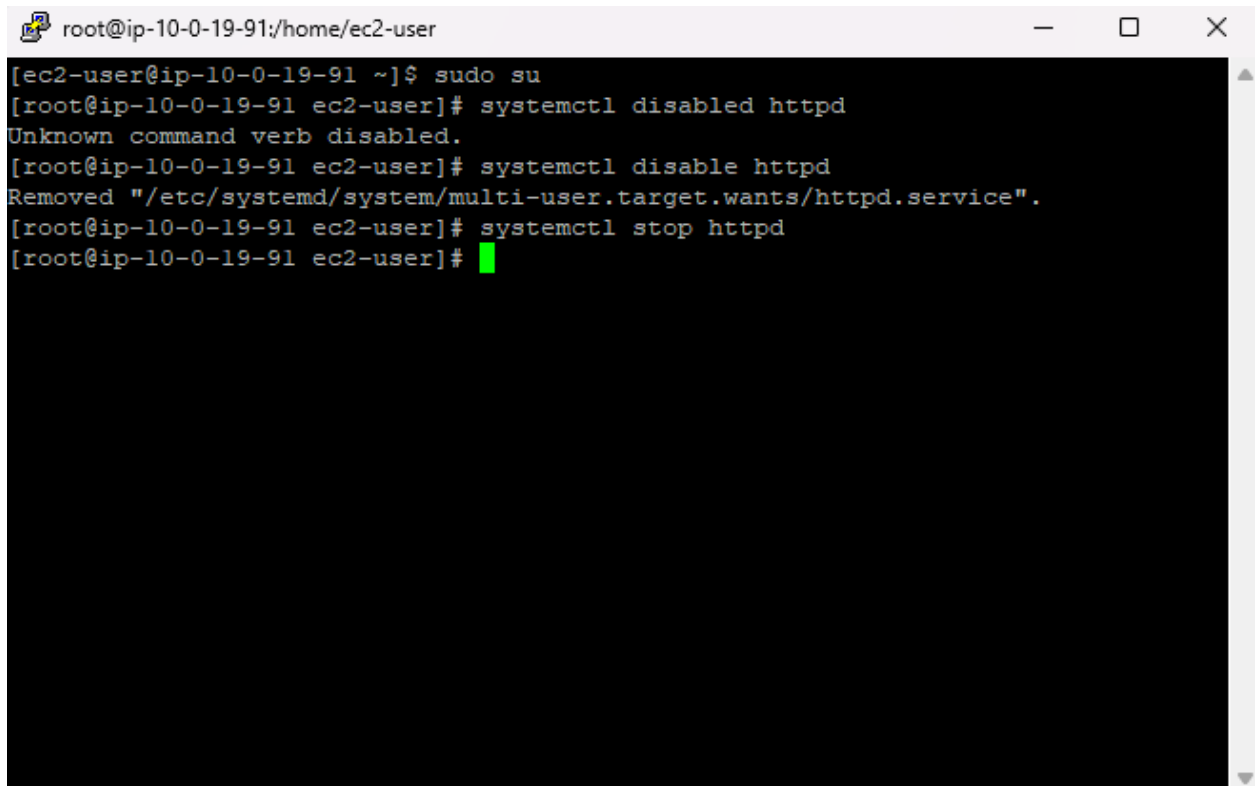
Usaremos usar la instancia llamada Server Seminario 1

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
<input type="checkbox"/>	<u>ServerSeminario1</u>	i-0c37852c48f40e2db	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-44-199-201-150.co...	44.199.201.150
<input type="checkbox"/>		i-0f7b15cd8a20a858	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-44-199-232-101.co...	44.199.232.101
<input type="checkbox"/>		i-06280f1becf9887	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-34-238-164-160.co...	34.238.164.160
<input type="checkbox"/>	ServerSeminario2	i-089dd9b88e56b655d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-18-215-234-204.co...	18.215.234.204
<input type="checkbox"/>		i-0f7016763818ae194	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-100-26-149-217.co...	100.26.149.217

Fuente: Elaboración propia

Nos conectamos a la instancia y comenzamos con la instalación de nginx.

También pausamos el servicio de Apache pues este caso utilizaremos otro servidor web, lo realizamos así



```
root@ip-10-0-19-91:/home/ec2-user
[ec2-user@ip-10-0-19-91 ~]$ sudo su
[root@ip-10-0-19-91 ec2-user]# systemctl disabled httpd
Unknown command verb disabled.
[root@ip-10-0-19-91 ec2-user]# systemctl disable httpd
Removed "/etc/systemd/system/multi-user.target.wants/httpd.service".
[root@ip-10-0-19-91 ec2-user]# systemctl stop httpd
[root@ip-10-0-19-91 ec2-user]#
```

Fuente: elaboración propia.

Instalación de nginx

```
root@ip-10-0-19-91:/home/ec2-user
[root@ip-10-0-19-91 ec2-user]# dnf install nginx -y
Last metadata expiration check: 1 day, 13:13:39 ago on Sun Dec 8 14:12:34 2024.
Dependencies resolved.
=====
Package                Arch      Version                Repository              Size
=====
Installing:
  nginx                 x86_64    1:1.26.2-1.amzn2023.0.1  amazonlinux             33 k
Installing dependencies:
  gperftools-libs       x86_64    2.9.1-1.amzn2023.0.3    amazonlinux             308 k
  libunwind             x86_64    1.4.0-5.amzn2023.0.2    amazonlinux             66 k
  nginx-core            x86_64    1:1.26.2-1.amzn2023.0.1  amazonlinux            670 k
  nginx-filesystem      noarch    1:1.26.2-1.amzn2023.0.1  amazonlinux            9.9 k
  nginx-mimetypes       noarch    2.1.49-3.amzn2023.0.3    amazonlinux            21 k
=====
Transaction Summary
=====
Install 6 Packages

Total download size: 1.1 M
Installed size: 3.6 M
Downloading Packages:
(1/6): nginx-1.26.2-1.amzn2023.0.1.x86_64.rpm 669 kB/s | 33 kB 00:00
(2/6): libunwind-1.4.0-5.amzn2023.0.2.x86_64.rp 1.2 MB/s | 66 kB 00:00
```

Fuente: elaboración propia.

Los comandos para administrar son los mismo que los de Apache, solo cambia el final de comando, ejecutamos el comando de iniciar (start) y el comando de iniciar siempre que arranque el servicio (enable)

```
root@ip-10-0-19-91:/home/ec2-user
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 6/6

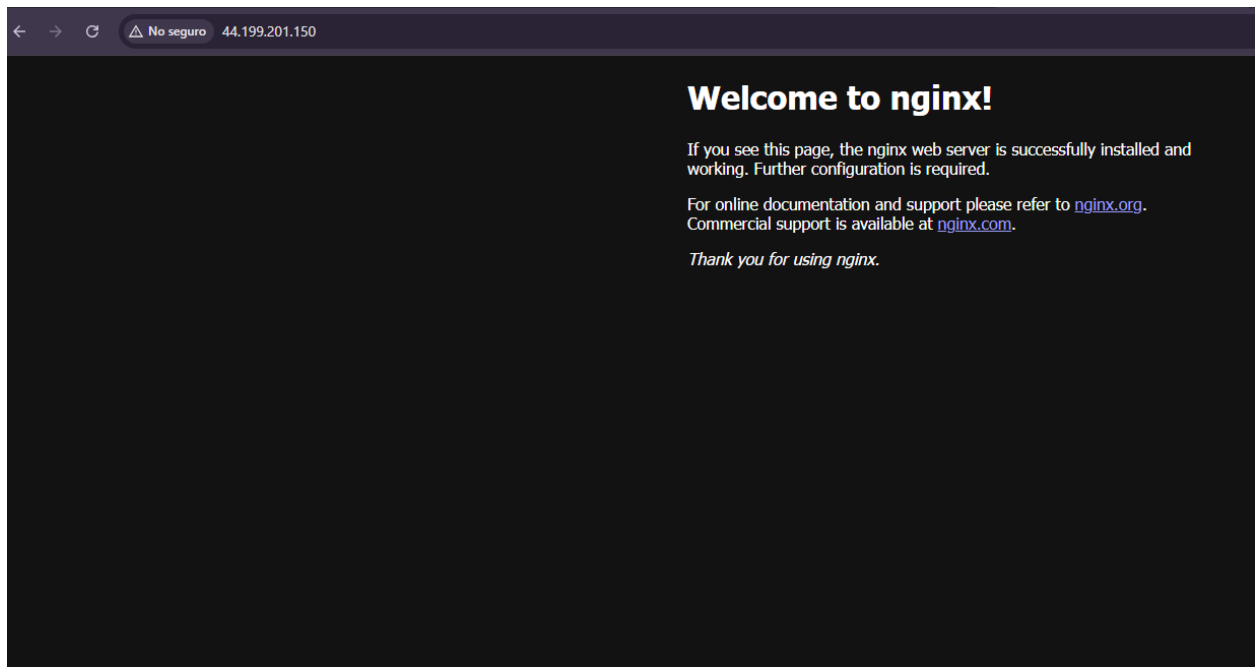
Installed:
  gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
  libunwind-1.4.0-5.amzn2023.0.2.x86_64
  nginx-1:1.26.2-1.amzn2023.0.1.x86_64
  nginx-core-1:1.26.2-1.amzn2023.0.1.x86_64
  nginx-filessystem-1:1.26.2-1.amzn2023.0.1.noarch
  nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

Complete!
[root@ip-10-0-19-91 ec2-user]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset: d
   Active: inactive (dead)

lines 1-3/3 (END)
^C
[root@ip-10-0-19-91 ec2-user]# sustemctl start nginx
bash: sustemctl: command not found
[root@ip-10-0-19-91 ec2-user]# systemctl start nginx
[root@ip-10-0-19-91 ec2-user]# systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr
/lib/systemd/system/nginx.service.
[root@ip-10-0-19-91 ec2-user]# █
```

Fuente: elaboración propia.

Verificamos que el servicio este ejecutado correctamente accediendo a la IP, debemos ver el mensaje de nginx



Procedemos con la instalación del Docker:

```
root@ip-10-0-19-91:/home/ec2-user
[ec2-user@ip-10-0-19-91 ~]$ sudo su
[root@ip-10-0-19-91 ec2-user]# yum install docker -y
Last metadata expiration check: 1 day, 13:42:53 ago on Sun Dec  8 14:12:34 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
=====
Transaction Summary
=====
Install 10 Packages
```

Fuente: elaboración propia.

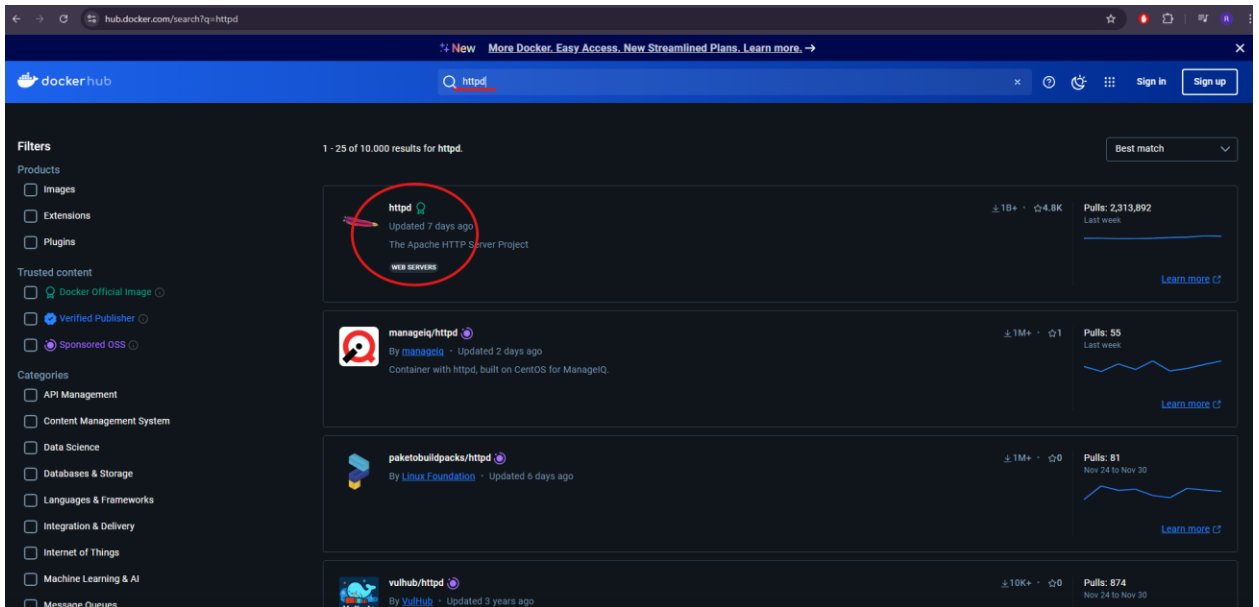
Iniciamos el servicio y establecemos que siempre se inicie automáticamente, luego verificamos el estado del servicio

```
root@ip-10-0-19-91:/home/ec2-user
[root@ip-10-0-19-91 ec2-user]# systemctl start docker
[root@ip-10-0-19-91 ec2-user]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-10-0-19-91 ec2-user]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
   Active: active (running) since Tue 2024-12-10 03:59:18 UTC; 30s ago
 TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
  Main PID: 382047 (dockerd)
    Tasks: 7
   Memory: 39.3M
     CPU: 27lms
   CGroup: /system.slice/docker.service
           └─382047 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/co>
Dec 10 03:59:17 ip-10-0-19-91.ec2.internal systemd[1]: Starting docker.service >
Dec 10 03:59:17 ip-10-0-19-91.ec2.internal dockerd[382047]: time="2024-12-10T03>
Dec 10 03:59:17 ip-10-0-19-91.ec2.internal dockerd[382047]: time="2024-12-10T03>
Dec 10 03:59:18 ip-10-0-19-91.ec2.internal dockerd[382047]: time="2024-12-10T03>
Dec 10 03:59:18 ip-10-0-19-91.ec2.internal dockerd[382047]: time="2024-12-10T03>
Dec 10 03:59:18 ip-10-0-19-91.ec2.internal dockerd[382047]: time="2024-12-10T03>
Dec 10 03:59:18 ip-10-0-19-91.ec2.internal dockerd[382047]: time="2024-12-10T03>
Dec 10 03:59:18 ip-10-0-19-91.ec2.internal systemd[1]: Started docker.service ->
lines 1-20/20 (END)...skipping...
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: dis>
```

Fuente: elaboración propia.

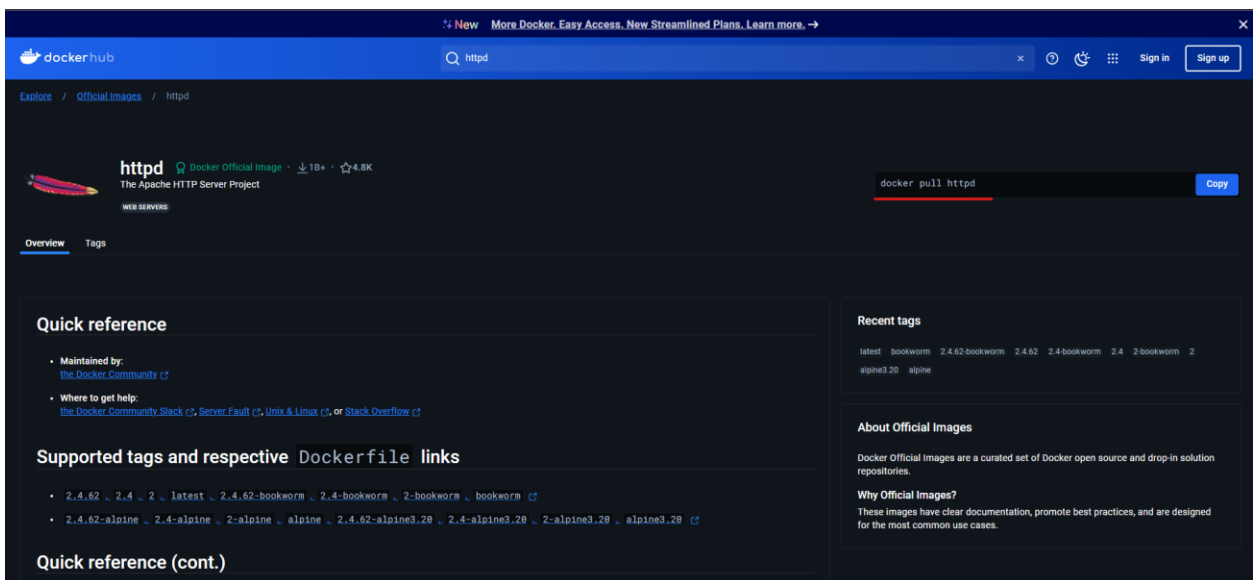
Ahora que tenemos el servicio de Docker ejecutándose podemos crear contenedores, los cuales son procesos dentro del sistema operativo.

Para crear un contenedor necesitamos una imagen, esta contiene la aplicación que vamos a utilizar en este caso un servidor web. La obtenemos del Hub de Docker en internet.



Fuente: elaboración propia.

Accedemos a la opción de httpd e usamos el comando que nos indica la información para realizar la instalación



Fuente: elaboración propia.

Ejecutamos el comando de instalación y luego listamos las imágenes que tenemos en Docker, ahí veremos la que acabamos de crear (imagen httpd)

```
root@ip-10-0-19-91:/home/ec2-user
[root@ip-10-0-19-91 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
[root@ip-10-0-19-91 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
httpd         latest   494b2b45fd74   4 months ago  147MB
[root@ip-10-0-19-91 ec2-user]#
```

Fuente: elaboración propia.

Con la imagen instalada podemos crear contenedores de la siguiente manera, primero con el comando `pwd` nos damos cuenta en que ubicación nos encontramos, posteriormente nos dirigíamos a la carpeta raíz con el comando `cd /` y luego creamos una carpeta con `mkdir` seguido del nombre de la carpeta para allí alojar el contenedor.

```
root@ip-10-0-19-91/
[root@ip-10-0-19-91 ec2-user]# pwd
/home/ec2-user
[root@ip-10-0-19-91 ec2-user]# cd /
[root@ip-10-0-19-91 /]# ls
bin  dev  home  lib64  media  opt  root  sbin  sys  usr
boot  etc  lib  local  mnt  proc  run  srv  tmp  var
[root@ip-10-0-19-91 /]# mkdir apiClientes
[root@ip-10-0-19-91 /]# chmod 777 apiClientes/
[root@ip-10-0-19-91 /]#
```

Fuente: elaboración propia.

Al tener la carpeta, creamos el contenedor y luego listo los contenedores

```
root@ip-10-0-19-91:/home/ec2-user
[root@ip-10-0-19-91 ec2-user]# docker run -dit --name clientes1 --restart always
-p 8082:80 -v /apiClientes:/usr/local/apache2/htdocs/ httpd
b8d5a88a8b5ec5808f3fal209db2b04be8d2bbefeb33bdb1d8c4008487bb481b
[root@ip-10-0-19-91 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        P
ORTS          NAMES
b8d5a88a8b5e   httpd    "httpd-foreground"     59 seconds ago  Up 57 seconds  0
.0.0.0:8082->80/tcp, :::8082->80/tcp  clientes1
[root@ip-10-0-19-91 ec2-user]#
```

Fuente: elaboración propia.

Nos ubicamos en la carpeta que creamos anteriormente y creamos un archivo

```
root@ip-10-0-19-91:/apiClientes
[root@ip-10-0-19-91 ec2-user]# ls
[root@ip-10-0-19-91 ec2-user]# cd ..
[root@ip-10-0-19-91 home]# ls
ec2-user
[root@ip-10-0-19-91 home]# cd ..
[root@ip-10-0-19-91 /]# ls
apiClientes boot etc lib local mnt proc run srv tmp var
bin dev home lib64 media opt root sbin sys usr
[root@ip-10-0-19-91 /]# cd
apiClientes/ etc/ local/ proc/ srv/ var/
bin/ home/ media/ root/ sys/
boot/ lib/ mnt/ run/ tmp/
dev/ lib64/ opt/ sbin/ usr/
[root@ip-10-0-19-91 /]# cd apiClientes/
[root@ip-10-0-19-91 apiClientes]# ls
[root@ip-10-0-19-91 apiClientes]# nano index.html
```

Fuente: elaboración propia.

Para visualizar el servicio web creamos la regla en la instancia de permitir el paso

Configuramos el nginx para que se encargue de manejar las peticiones que hacemos a los contenedores, buscamos la configuración que tiene en etc/nginx y entramos al archivo nginx.conf, luego sacamos una copia de este para tener respaldo y luego editamos el archivo con la configuración que deseamos.

```
root@ip-10-0-19-91:/etc/nginx
[root@ip-10-0-19-91 apiClientes]# cd /etc/nginx/
[root@ip-10-0-19-91 nginx]# ls
conf.d koi-utf scgi_params
default.d koi-win scgi_params.default
fastcgi.conf mime.types uwsgi_params
fastcgi.conf.default mime.types.default uwsgi_params.default
fastcgi_params nginx.conf win-utf
fastcgi_params.default nginx.conf.default
[root@ip-10-0-19-91 nginx]# cp nginx.conf nginx.bk
[root@ip-10-0-19-91 nginx]# nano nginx.conf
GNU nano 5.8 nginx.conf
```

Fuente: elaboración propia.

Así es como queda la configuración del archivo

```
root@ip-10-0-19-91:/etc/nginx
GNU nano 5.8 nginx.conf
events {}

http {
    upstream clientes {
        server localhost:8082;
    }

    server {
        listen 80;
        server_name calidosoclientes;
        location / {
            proxy_pass http://clientes;
        }
    }
}

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/_ Go To Line M-E Redo
```

Fuente: elaboración propia

Reiniciamos el servicio del nginx y ahora podemos conectarnos a nuestro contenedor



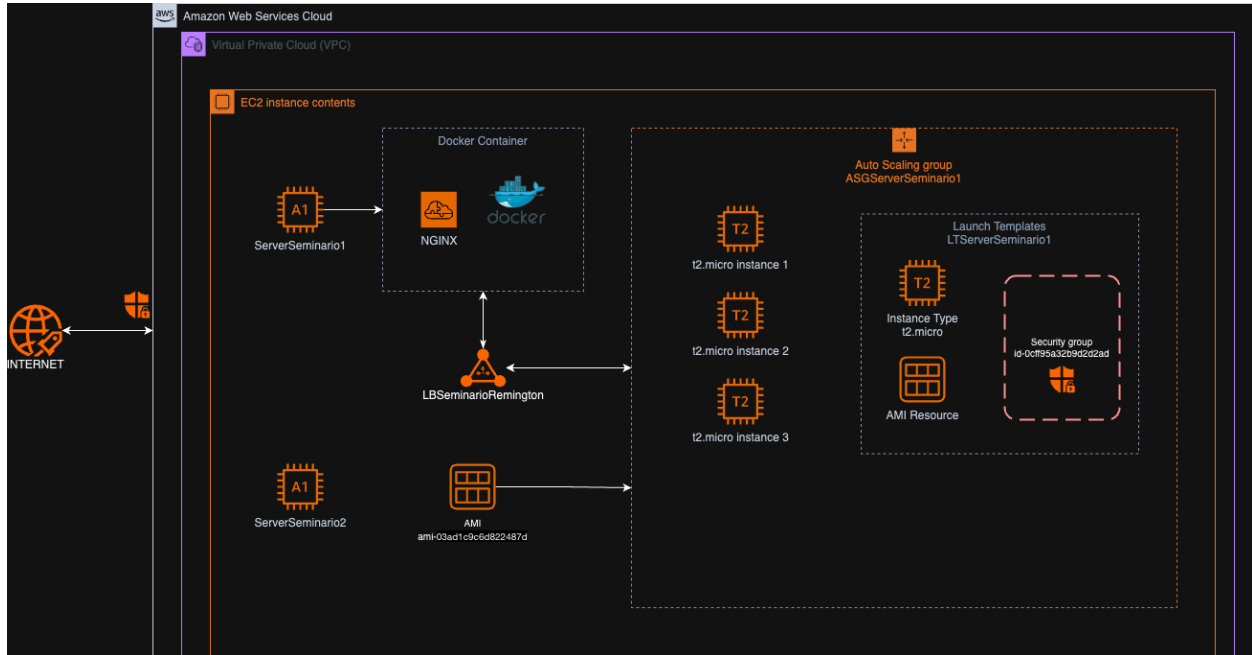
A screenshot of a web browser window displaying a table with client information. The browser's address bar shows the URL "44.199.201.150". The table has five columns: ID, Nombre, Cliente, ID Sucursal, and Telefono. It contains two rows of data.

ID	Nombre	Cliente	ID Sucursal	Telefono
1	Juan Pérez	ABC Corp	101	555-1234
2	María López	XYZ Ltd	102	555-5678

Fuente: elaboración propia.

Tablas y Figuras

Diagrama de los recursos usados y la comunicación entre ellos.



Fuente: elaboración propia

Conclusiones

La implementación de servicios de AWS permite escalar dinámicamente los recursos, optimizando costos y garantizando alta disponibilidad. El uso de tecnologías como Docker y NGINX mejora la modularidad y gestión de aplicaciones, permitiendo una infraestructura ágil. El diseño de esta solución demuestra cómo los conceptos teóricos del seminario pueden aplicarse a problemas reales, mostrando un impacto positivo en startups en crecimiento. La configuración de herramientas como Auto Scaling y LB refuerza la capacidad de manejar picos de demanda sin comprometer el rendimiento.

Referencias

Amazon Web Services. (n.d.). Amazon Web Services (AWS) – Cloud computing services.
<https://docs.aws.amazon.com/>

Docker, Inc. (n.d.). Docker Hub. <https://hub.docker.com> - https://hub.docker.com/_/httpd

Berrio, J. (2024, noviembre 19 – diciembre 6). *Seminario Amazon Web Service*.

Corporación universitaria Remington, Microsoft Teams.