

TRABAJO DE GRADO
Opción Seminario-Diplomado.

DE LO FÍSICO A LA NUBE CON AWS

Corporación Universitaria Remington.
Facultad de Ingenierías

Pregrado en Ingeniería de Sistemas,
Postgrado Especialización en Seguridad de la Información

Presentado Por:

Anderson David García Perez (Postgrado)
Andres Londoño Orrego (Pregrado)
Dairo Luis Barrera Jiménez (Pregrado)

Presentado a:

Juan Pablo Berrio López (Docente)

Opción de Trabajo de grado Seminario.

2025.

Tabla de Contenidos

Resumen	3
Marco conceptual y contextual	4
Título 1	5
Sub-Título 1.1.	5
Desarrollo e implementación del aprendizaje	5
Línea de tiempo	5
Conclusiones	59
Referencias	60
.....	60

Resumen

En este proyecto se podrán en práctica los conceptos aprendidos durante el transcurso del seminario donde trabajaremos con instancias las cuales crearemos de manera manual y posteriormente de manera automática(AMI), Adicional, aprenderemos sobre la importancia de la escalabilidad ya que es un factor determinante en el proceso de nuestras compañías donde aplicaremos autoScaling, proxy reverso, grupos de seguridad, etc. También veremos entornos de contenedores utilizando Dockers, donde se crearán pruebas con una o más páginas web a través de contenedores que ejecutan en un servidor Apache (httpd) y estos mismos se podrán administrar bajo un proxy reverso atreves de Nginx. Este proyecto se desarrollará atreves de instancia de AWS, en este proyecto daremos un manejo dinámico y escalable de la estructura web que nos presenta la computación en la nube.

Palabras claves

Computación en la nube
Virtualización con AWS
Docker
Infraestructura Computacional
Orquestación de Servicios

Marco conceptual y contextual

Para conceptualizar, las **instancias** es un modelo computacional, implementado por el que en adelante lo llamaremos “AWS” por su nombre Amazon Web Services, este modelo computacional en la nube hace que a sus clientes les sea llamativo dicho modelo ya que su relación costo beneficio es muy grande, por su facilidad de uso y amplio paquete de servicios en la nube, estas instancias “Maquinas o servidores” se pueden implementar de manera automática para que lleven a cabo procesos stand alone de manera independiente con muy poca intervención humana, para esto se usa el termino AMI lo cual significa que se hacen de manera automática. Esto agiliza los procesos y pone fin a los servidores de manera física en las empresas los cuales corren el riesgo de daños físicos, apagones, caída de internet, etc.

Para que dichos sistemas sean competentes se usa el termino AutoScaling lo cual significa un escalado automático el cual consiste en observar las necesidades del cliente, controlar y vigilar para poder así tomar las decisiones dado el caso, si es necesario generar un aumento en la capacidad de los servidores o incluso crear una instancia nueva la cual pueda ayudar a suplir la necesidad mejorando el rendimiento y las respuestas.

Dockerizacion es un nuevo modelo de código abierto que se utiliza para empaquetar y correr aplicaciones sin importar el tipo de maquina ya que corre en contenedores, los cuales distribuyen la carga según la necesidad del cliente, es decir, cada usuario ingresara de manera normal al aplicativo y el Docker es el encargado de saber qué tipo y para donde va la petición que el usuario le esta asignando, esto permite un entorno dinámico y fluido.

Título 1

Sub-Título 1.1.

Desarrollo e implementación del aprendizaje

Línea de tiempo

A Continuación trataremos de dar una visión clara sobre el termino línea de tiempo en la virtualización y la computación en la Nube:

Pudiésemos decir que esto se remonta a la década de los 60.

Donde Surge por primera vez el concepto en informática de time-sharing o tiempo compartido en las computadoras centrales o mainframes, computadoras de alto rendimiento utilizadas para el procesamiento y almacenaje de datos, maximizando el uso de recursos.

En los años 60. IBM fue una de las empresas más innovadoras en este aspecto de mainframes por lo cual compañías como bancos, grandes multinacionales, agencias de viaje comenzaron a utilizar este tipo de sistema novedoso para poder cumplir con el procesamiento de grandes volúmenes de datos y la ejecución de programas críticos.

En la década de los 70. IBM desarrollaría la tecnología de la virtualización de su sistema VM “Virtual Machine”, permitiendo marcar un hito en la computación ya que permitiría la ejecución de múltiples sistemas operativos en una misma máquina, algo que para esa fecha era inimaginable para le gente del común. IBM encabeza la tecnología con un revolucionario invento como lo fue la virtualización, esto se benefició gran parte del sector financiero y compañías las cuales tenían problemas por su demanda y poca eficiencia en los procesos.

En los 80, se vuelve muy común y popular utilizar el concepto de cliente-servidor, lo que conllevaría a que la virtualización perdiera relevancia debido al enfoque en servidores físicos, en esta época comenzó el auge de las redes locales LAN en lo cual IBM con sus tipos de Software ayudarían a optimizar procesos y hacer de este tipo de estructura algo novedoso y prometedor para la época facilitando el uso y una fácil adaptabilidad según el requerimiento de la empresa en la cual se implementaba.

Para los 90, Surgiría VMware (1998), que introduciría la virtualización en sistemas con arquitectura de x86 o 32 bits como se le conoce, permitiendo la creación de máquinas virtuales en hardware comunes y corriente atrás quedaría el tema de que la virtualización sólo se podía en máquinas de grandes superficies ya que con este nuevo concepto se pudiese virtualizar maquinas en equipos de hogares y oficinas, este nuevo sistema revolucionario permitió que la tecnología no fuera algo exclusivamente para las empresas, permitió que las PYMES y las personas del común pudieran tener acceso lo

cual redujo considerablemente los costos de las tecnologías y haciendo de la tecnología algo de todos y no de unos pocos.

En la década del 2000, surge Amazon con su revolucionario AWS en 2006 con S3 y EC2, introduciendo el concepto de computación en la nube, Google y Microsoft no se quedarían a tras ya que inician sus servicios en la nube. Google con su Google App Engine en el 2008 y Microsoft con su ya conocido Azure en el 2010.

Para este mismo periodo se estandarizan las tecnologías de virtualización como lo son: Xen, KVM y Hyper-V, a principios de la década Netflix era una empresa pequeña la cual tenía su servicio de alquiler de DVD cada vez que su público comenzaba a aumentar se dieron cuenta que experimentar con el streaming era un reto importante pero que podría traerle una escalabilidad significativa ya que podrían manejar picos masivos, también almacenar grandes volúmenes de datos en este caso videos.

Para el año 2010, contenedorización se vuelve popular con Docker en el 2013 y Kubernetes en el 2015, optimizando el uso de la nube de maneras muy eficientes, en este mismo periodo aparecen modelos como SaaS, PaaS e IaaS, que le darían un gran impulso al crecimiento de la nube.

Ya para el 2020. la computación en la nube se integra con IA, Big Data y Edge Computing, alcanzando niveles de automatización y escalabilidad sin precedentes e inimaginables.

Como un aparte trataremos de crear un paso a paso para realizar la correcta configuración para realizar el **AutoScaling** utilizado para crear un balanceo automático de los recursos informáticos:

Como primer paso, crearemos una instancia para el server de Amazon Linux ya que será nuestro OS seleccionado, le asignaremos el nombre que deseemos ya que este será el indicador de nuestra instancia.

EC2 > Instances > Launch an instance

It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices. Do not show me this message again. Take a walkthrough

Launch an instance

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

Name: LinuxServer [Add additional tags](#)

Application and OS Images (Amazon Machine Image)

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 need.

Summary

- Number of instances: 1
- Software Image (AMI): Amazon Linux 2023 AMI 2023.6.2...read more (ami-08b5b5a93ed654d19)
- Virtual server type (instance type): t2.micro
- Firewall (security group): New security group

Dejar de compartir | Ocultar

Seleccionamos el Sistema operativo, Amazon Linux suele tener más soporte ya que es creado por el mismo Amazon:

Recents Quick Start

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Del

aws Mac ubuntu® Microsoft Red Hat SUSE

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-08b5b3a93ed654d19 (64-bit (x86), uefi-preferred) / ami-0eae2a0fc13b15fce (64-bit (Arm), uefi)
Virtualization: hvm ENA 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 2023.6.20250303.0 x86_64 HVM ker | meet.google.com está compartiendo tu pantalla. Dejar de compartir Ocultar

▼ **Summ**

Number of i
1

Software In
Amazon Lin
ami-08b5b3a...

Virtual serv
t2.micro

Firewall (se
New securit:

Cancel

Acá visualizamos las claves que tenemos inscritas o ya realizadas. Cabe resaltar que esta no se puede utilizar ya que fue creada para un sistema operativo Windows. Por lo cual debemos realizar una nueva .

▼ **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

Select

Create new key pair

Q |

Proceed without a key pair (Not recommended) Default value

ClaveWindows
Type: rsa

vpc-0bd972a14ed0417a8

Subnet | Info

No preference (Default subnet in any availability zone)


Edit

Crearemos una nueva Clave para nuestro server Linux de la siguiente manera:

▼ **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

Select ▼ [Create new key pair](#) 

Seleccionaremos la que tenemos resaltada “RSA” este es un modelo de encriptación exclusivo para Linux:

Create key pair ✕

Key pair name
Key pairs allow you to connect to your instance securely.

ClaveLinux

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA
RSA encrypted private and public key pair

ED25519
ED25519 encrypted private and public key pair

Private key file format

.pem
For use with OpenSSH

.ppk

Seleccionaremos el VPC (red privada virtual) creada por nosotros mismo ya que viene una por defecto, y para este caso necesitamos uno creado por nosotros:

The screenshot shows the 'Network settings' section of the 'Launch an instance' page. It includes a dropdown for VPC selection, a search bar, and a list of VPCs. The VPC 'vpc-01c97d94b626b74eb (VPC-WEB-vpc)' is highlighted with a red box and a checkmark. Below the VPC list is the 'Auto-assign public IP' dropdown set to 'Disable'. The 'Firewall (security groups)' section has two radio buttons: 'Create security group' (selected) and 'Select existing security group'. On the right side, there is a 'Sum' sidebar with 'Number of instances' set to 1, and a 'Cancel' button at the bottom.

Necesitaremos una ip publica activada para que AWS no nos cree de manera automática y así poder seguir sin afectar el proceso.

The screenshot shows the 'Subnet' selection section. A dropdown menu is open, showing the selected subnet 'subnet-0b6b9cf42cd447139' with details: VPC: vpc-01c97d94b626b74eb, Owner: 441877405374, Availability Zone: us-east-1a, Zone type: Availability Zone, and IP addresses available: 4090. Below the subnet dropdown is the 'Auto-assign public IP' dropdown set to 'Enable', with a red arrow pointing to it. The 'Firewall (security groups)' section is also visible at the bottom.

Crearemos un grupo de seguridad para nuestra instancia, le daremos un nombre como se evidencia en la siguiente imagen.

EC2 > Instances > Launch an instance

Disable

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*
Security-ServerLinux

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-1 created 2025-03-20T21:57:03.657Z

Inbound Security Group Rules
▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) Remove

Type	Protocol	Port range
ssh	TCP	22

Crearemos una regla para permitir el tráfico por el puerto 80 de la siguiente manera:

Inbound Security Group Rules

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

Type	Protocol	Port range
ssh	TCP	22

Source type: Anywhere | Source: 0.0.0.0/0 | Description: e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0, Permitir el servicio http) Remove

Type	Protocol	Port range
Custom TCP	TCP	80

Source type: Custom | Source: 0.0.0.0/0 | Description: Permitir el servicio http

Validaremos si la instancia se creó correctamente, si todo va bien debería de verse reflejado igual que en la imagen:

Instances (2) info Last updated less than a minute ago Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive) All states < 1 >

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
server1	i-Ofa6ad9850dae1f7d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-2
LinuxServer	i-04c9bd3bdfad6bc8b	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-E

Para conectarnos a nuestra instancia y comenzar a trabajar con ella utilizaremos el apartado SSH client como se muestra en la imagen, previo a esto podés descargar un software que nos permita realizar conexiones con clientes SSH.

VPC EC2

EC2 > Instances > i-04c9bd3bdfad6bc8b > Connect to instance

Connect to instance Info

Connect to your instance i-04c9bd3bdfad6bc8b (LinuxServer) using any of these options

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

Instance ID
i-04c9bd3bdfad6bc8b (LinuxServer)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ClaveLinux.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "ClaveLinux.pem"
4. Connect to your instance using its Public DNS:
ec2-54-80-33-174.compute-1.amazonaws.com

Example:
ssh -i "ClaveLinux.pem" ec2-user@ec2-54-80-33-174.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.

Después de descargar el SSH Client , nuestro Remote host ya estaría operando y seria el siguiente:

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

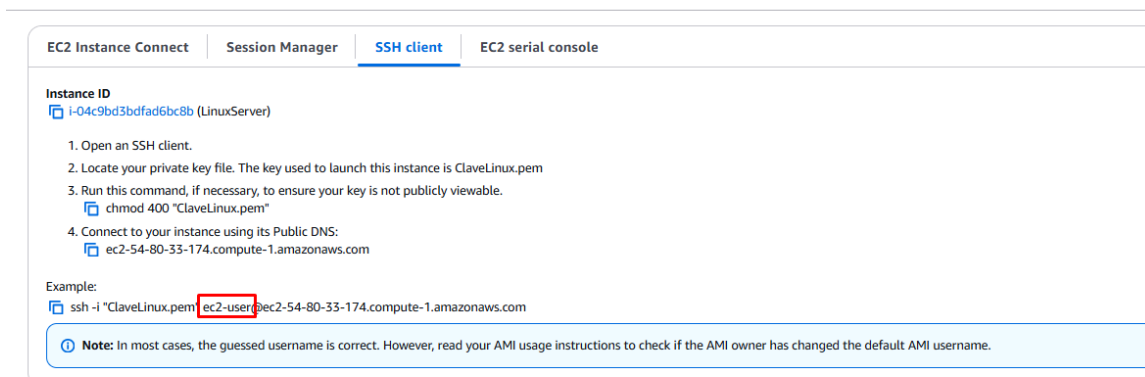
Instance ID
i-04c9bd3bdfad6bc8b (LinuxServer)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ClaveLinux.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "ClaveLinux.pem"
4. Connect to your instance using its Public DNS:
ec2-54-80-33-174.compute-1.amazonaws.com

Example:
ssh -i "ClaveLinux.pem" **ec2-user@ec2-54-80-33-174.compute-1.amazonaws.com**

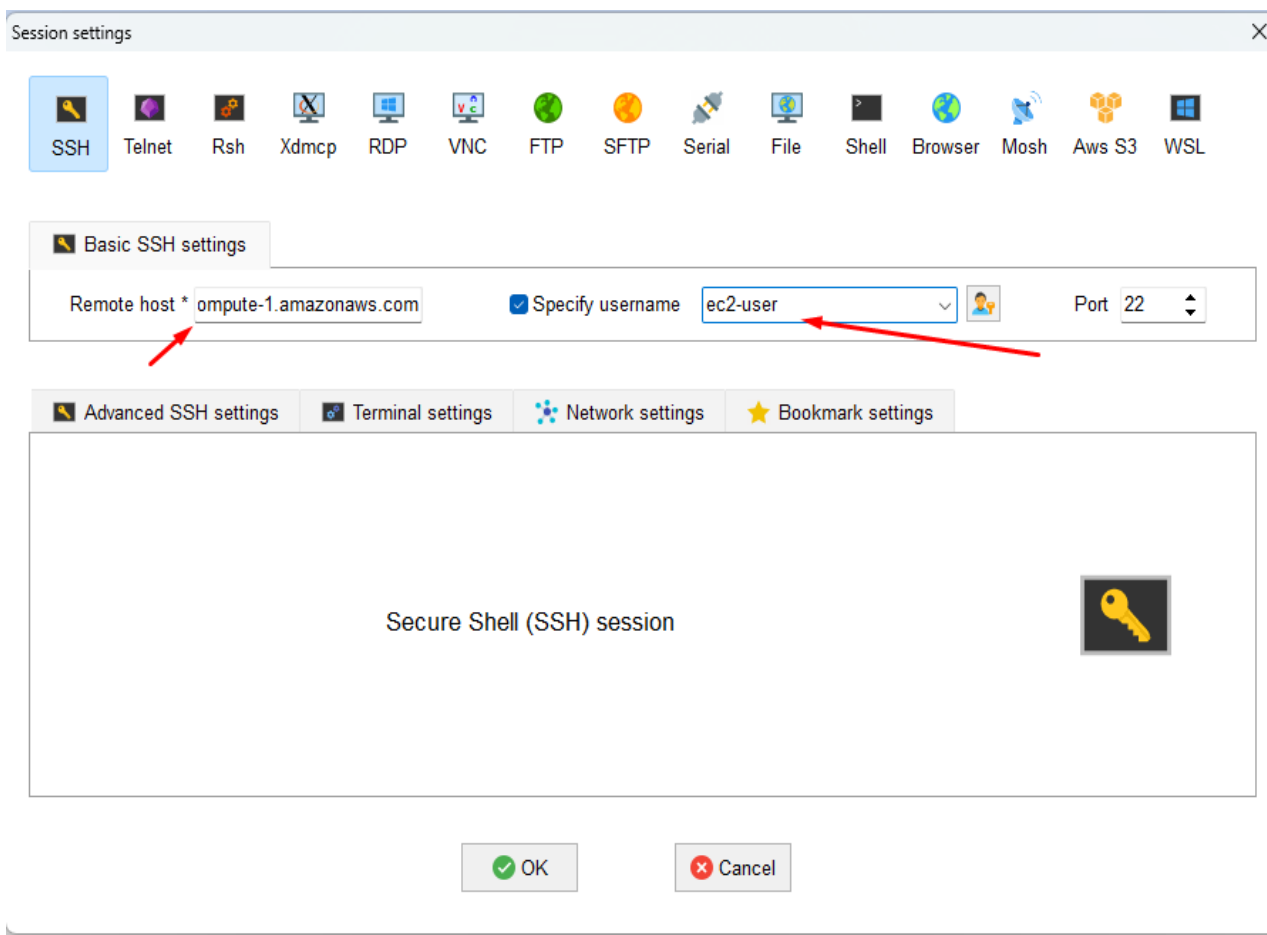
Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Nuestro user (usuario) sería el siguiente:



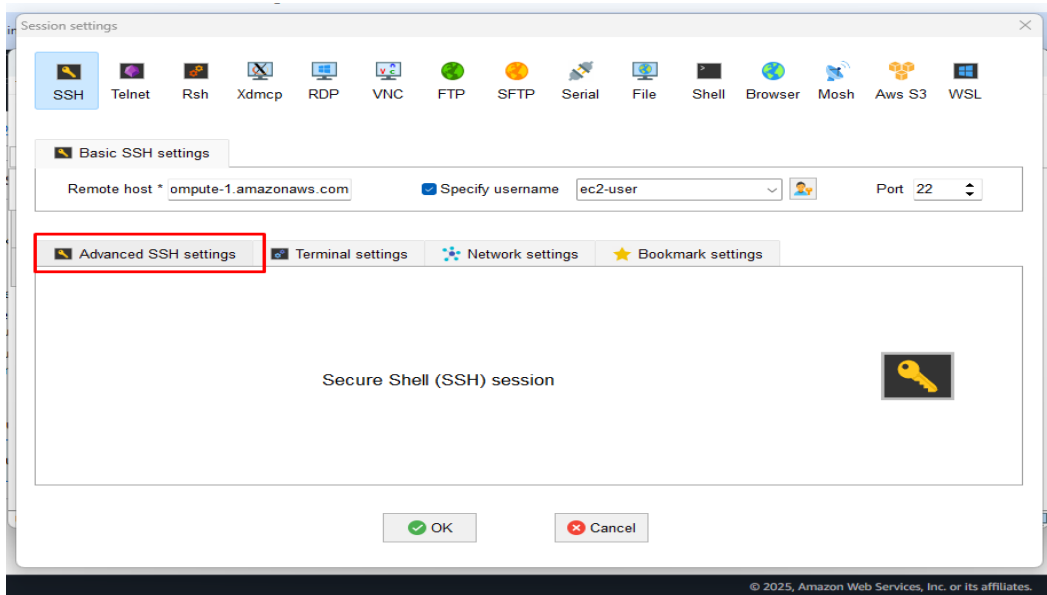
The screenshot shows the AWS Management Console interface for configuring an SSH client. The 'SSH client' tab is selected. It displays the Instance ID 'i-04c9bd3bdfad6bc8b (LinuxServer)' and provides a list of steps to connect via SSH. Step 3 includes the command 'chmod 400 "ClaveLinux.pem"'. Step 4 provides the Public DNS 'ec2-54-80-33-174.compute-1.amazonaws.com'. An example command is shown: 'ssh -i "ClaveLinux.pem" ec2-user@ec2-54-80-33-174.compute-1.amazonaws.com', where 'ec2-user' is highlighted with a red box. A note at the bottom states: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'

Utilizando el software SSH client en Windows nos encontraríamos con una imagen tal como se aprecia para poder realizar la conexión.

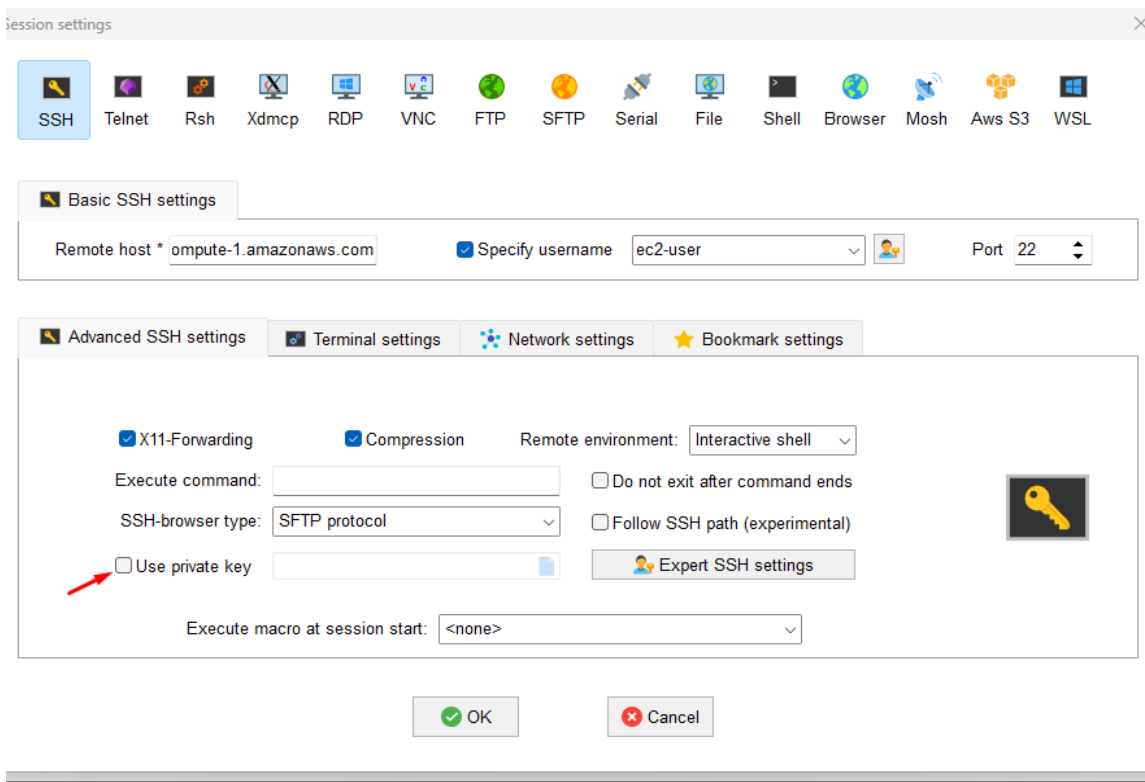


The screenshot shows the Windows 'Session settings' dialog box for an SSH connection. The 'SSH' icon is selected in the top toolbar. Under 'Basic SSH settings', the 'Remote host' is 'ompute-1.amazonaws.com' (with a red arrow pointing to it), 'Specify username' is checked, and the username is 'ec2-user' (with a red arrow pointing to it). The 'Port' is set to 22. Below this are tabs for 'Advanced SSH settings', 'Terminal settings', 'Network settings', and 'Bookmark settings'. The main area is labeled 'Secure Shell (SSH) session' and contains a key icon. At the bottom are 'OK' and 'Cancel' buttons.

Paso seguido seleccionamos el apartado Advanced SSH Settings:



Seleccionamos el check para poder ingresar la contraseña anteriormente creada en el proceso:



En esta imagen estaremos evidenciando el estado de nuestro server:

```

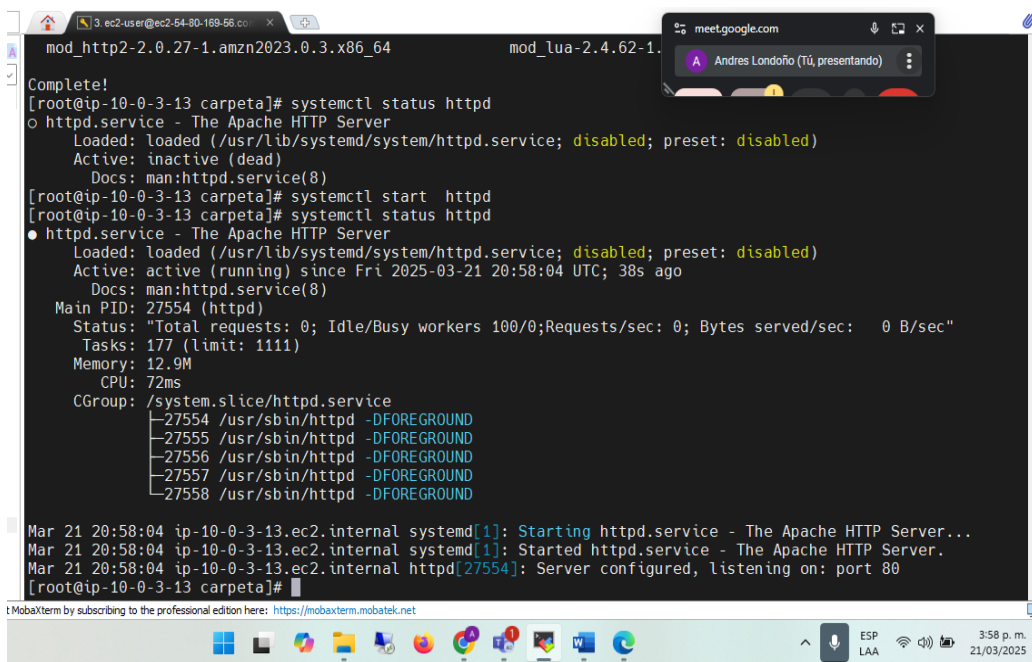
Installing      : httpd-2.4.62-1.amzn2023.x86_64
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64
Verifying      : apr-1.7.5-1.amzn2023.0.4.x86_64
Verifying      : apr-util-1.6.3-1.amzn2023.0.1.x86_64
Verifying      : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
Verifying      : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
Verifying      : httpd-2.4.62-1.amzn2023.x86_64
Verifying      : httpd-core-2.4.62-1.amzn2023.x86_64
Verifying      : httpd-filesystem-2.4.62-1.amzn2023.noarch
Verifying      : httpd-tools-2.4.62-1.amzn2023.x86_64
Verifying      : libbrotli-1.0.9-4.amzn2023.0.2.x86_64
Verifying      : mailcap-2.1.49-3.amzn2023.0.3.noarch
Verifying      : mod_http2-2.0.27-1.amzn2023.0.3.x86_64
Verifying      : mod_lua-2.4.62-1.amzn2023.x86_64

Installed:
apr-1.7.5-1.amzn2023.0.4.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-10-0-3-13 carpeta]# 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)
[root@ip-10-0-3-13 carpeta]#

```

Validaremos que el servicio este corriendo de forma normal, si todo va bien deberías de estar viendo una imagen similar a esta:



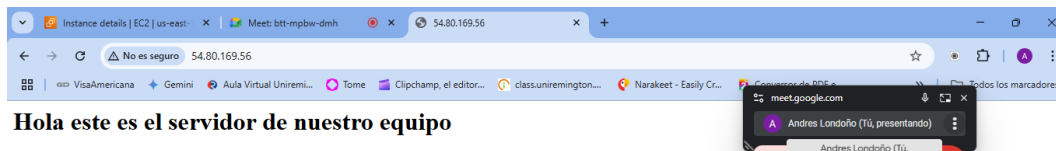
```

mod_http2-2.0.27-1.amzn2023.0.3.x86_64  mod_lua-2.4.62-1.
Complete!
[root@ip-10-0-3-13 carpeta]# 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)
[root@ip-10-0-3-13 carpeta]# systemctl start httpd
[root@ip-10-0-3-13 carpeta]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: active (running) since Fri 2025-03-21 20:58:04 UTC; 38s ago
     Docs: man:httpd.service(8)
  Main PID: 27554 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
    Tasks: 177 (limit: 1111)
   Memory: 12.9M
     CPU: 72ms
   CGroup: /system.slice/httpd.service
           └─27554 /usr/sbin/httpd -DFOREGROUND
             └─27555 /usr/sbin/httpd -DFOREGROUND
               └─27556 /usr/sbin/httpd -DFOREGROUND
                 └─27557 /usr/sbin/httpd -DFOREGROUND
                   └─27558 /usr/sbin/httpd -DFOREGROUND

Mar 21 20:58:04 ip-10-0-3-13.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Mar 21 20:58:04 ip-10-0-3-13.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Mar 21 20:58:04 ip-10-0-3-13.ec2.internal httpd[27554]: Server configured, listening on: port 80
[root@ip-10-0-3-13 carpeta]#

```

Utilizando un navegador web y digitado la ip asignada por AWS más, deberíamos poder observar navegación, con un mensaje diferente ya que en nuestro caso creamos un archivo índice para que el navegador mostrara información.

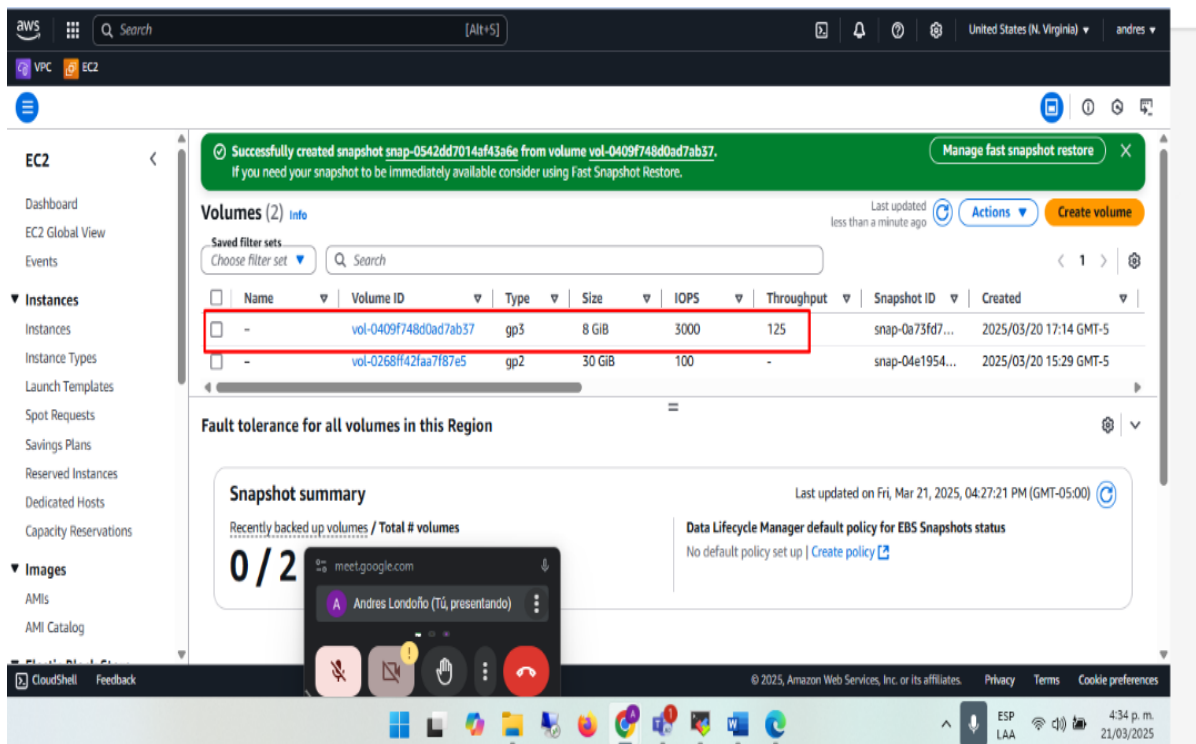


Hola este es el servidor de nuestro equipo

Andres Londoño, Anderson Garcia, Dairo Barrera



Seleccionamos el disco del server(instancia) Linux:



Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created
-	vol-0409f748d0ad7ab57	gp3	8 GiB	3000	125	snap-0a73fd7...	2025/03/20 17:14 GMT-5
-	vol-0268ff42aa7187e5	gp2	30 GiB	100	-	snap-04e1954...	2025/03/20 15:29 GMT-5

Snapshot summary

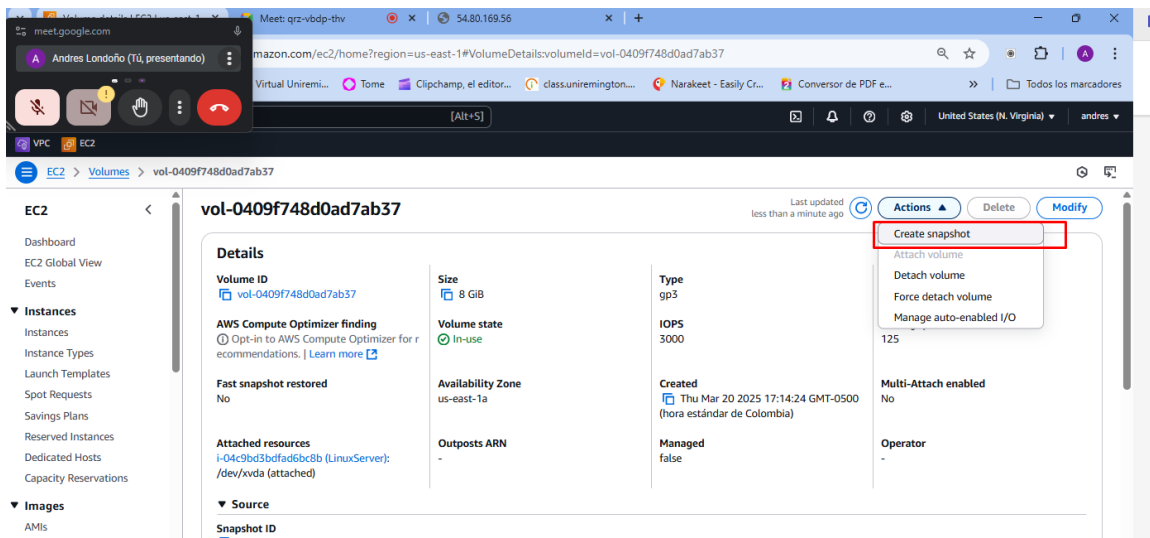
Recently backed up volumes / Total # volumes

0/2

Data Lifecycle Manager default policy for EBS Snapshots status

No default policy set up | [Create policy](#)

Podemos crear un Snapshots (copia de seguridad) de la siguiente manera:



Le ponemos un nombre representativo que nos dé alusión a lo que estamos realizando:

Create snapshot [Info](#)

Create a point-in-time snapshot to back up the data on an Amazon EBS volume to Amazon S3.

Source volume

Volume ID
`vol-0409f748d0ad7ab37`

Availability Zone
`us-east-1a`

Snapshot details

Description

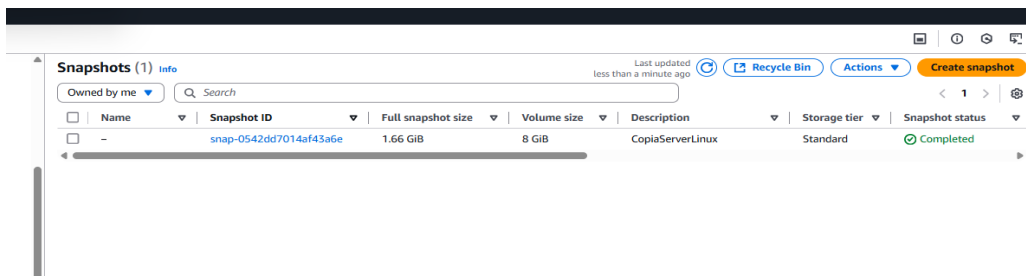
Add a description for your snapshot

255 characters maximum.

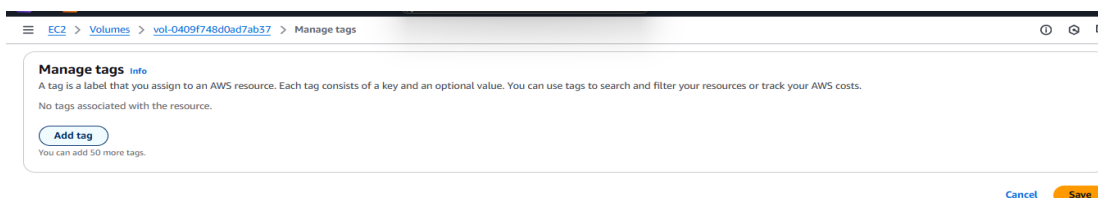
Encryption [Info](#)

Not encrypted

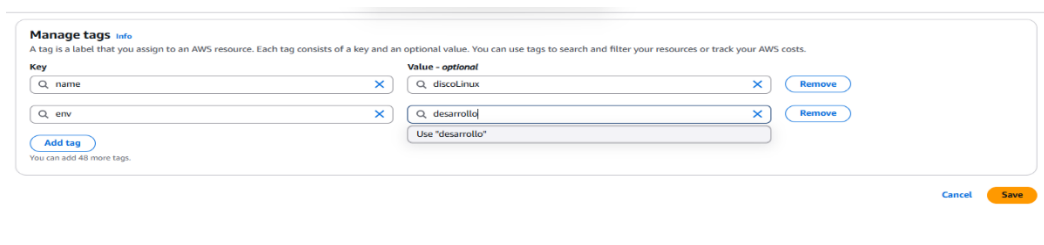
Podemos validar que ya fue creado:



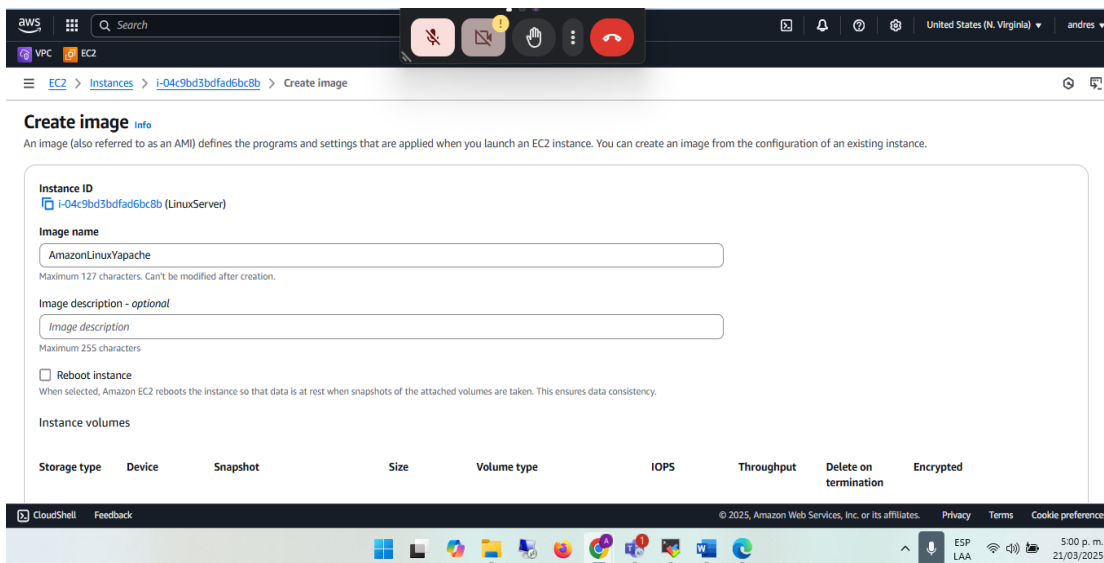
Creamos un tag para el disco de la instancia Linux de la siguiente manera:



Creamos los que sean necesarios para nuestro ambiente y hacer el backup de los discos:



Creamos una imagen tipo (backup) utilizando los snapshots:



Quedaría la AMI de la siguiente manera (clonaría toda la instancia), se ahorra todos los parámetros de configuración, solo seleccionando la AMI cuando crea la instancia:

Currently creating AMI `ami-028e9a22631c1455d` from instance `i-04c9bd3bdfad6bc8b`. Check that the AMI status is 'Available' before deleting the instance or carrying out other actions related to this AMI.

Instances (1/2) [Info](#) Last updated less than a minute ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) All states < 1 >

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Platform
<input type="checkbox"/>	server1	i-0fa6ad9850dae1f7d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2
<input checked="" type="checkbox"/>	LinuxServer	i-04c9bd3bdfad6bc8b	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2

Para crear una AMI en una nueva instancia hacemos lo siguiente:

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.

Recents **My AMIs** Quick Start

Owned by me Shared with me

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

Summary

Number of instances [Info](#)
1

Software Image (AMI)
AmazonLinuxYapache
ami-028e9a22631c1455d

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

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

[Cancel](#)

Seleccionamos la AMI anteriormente creada de la siguiente manera:

The screenshot shows the 'My AMIs' tab in the AWS Management Console. A red box highlights the 'AmazonLinuxYapache' AMI. The summary on the right shows 1 instance, AmazonLinuxYapache software image, t2.micro virtual server type, and 1 volume of 8 GiB.

Architecture	AMI ID
x86_64	ami-028e9a22631c1455d

Volvemos a utilizar la misma clave de la instancia anterior:

The screenshot shows the 'Key pair (login)' section in the AWS Management Console. The 'Key pair name' is set to 'ClaveLinux'.

Configuramos nuevamente la red y el VPC:

The screenshot shows the 'Network settings' section in the AWS Management Console. The VPC is 'vpc-01c97d94b626b74eb (VPC-WEB-vpc)' and the subnet is 'subnet-0b6b9cf42cd447139 VPC-WEB-subnet-public1-us-east-1a'. The 'Auto-assign public IP' is set to 'Disable' and the 'Firewall (security groups)' is set to 'Create security group'.

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

Common security groups | Info

Select security groups

Security-ServerLinux sg-0037027ed92263955 ✕
VPC: vpc-01c97d94b626b74eb

[Compare security group rules](#)

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

► **Advanced network configuration**

Soft
Amaz
ami-0

Virtu
t2.mi

Firew
Segu

Stor
1 vol

Queda de la siguiente manera:

Success
Successfully initiated launch of Instance (i-089bbfb4b9e6c3c85)

► **Launch log**

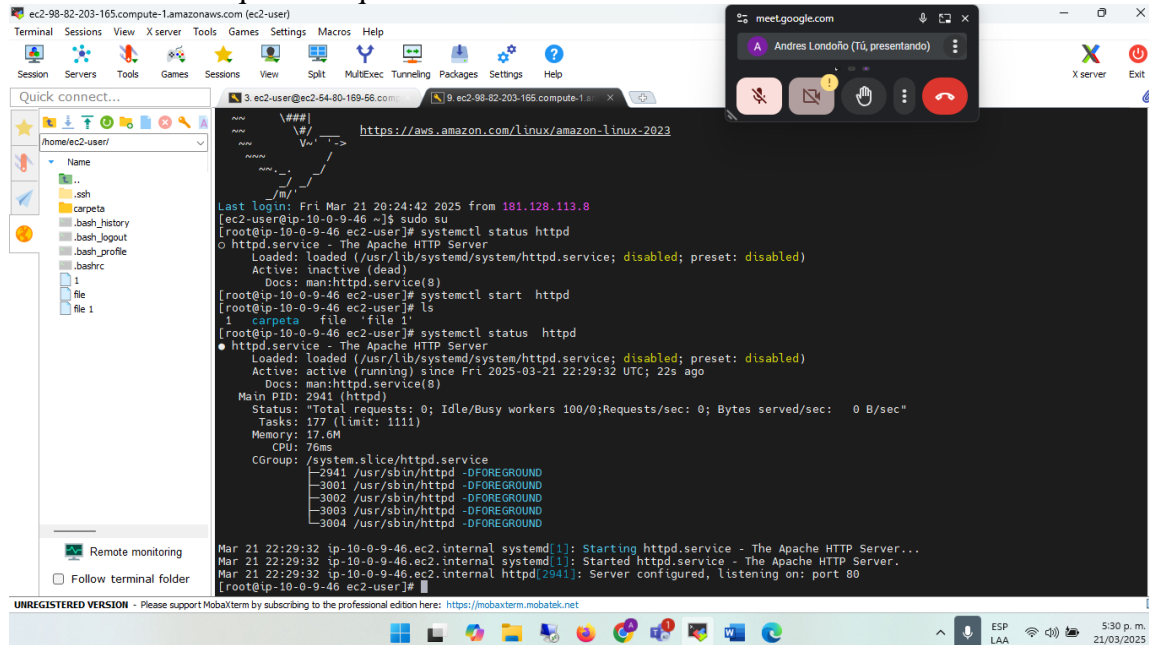
Next Steps
What would you like to do next with this instance, for example "create alarm" or "create backup"

- Create billing and free tier usage alerts**
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
[Create billing alerts](#)
- Connect to your instance**
Once your instance is running, log into it from your local computer.
[Connect to instance](#)
[Learn more](#)
- Connect an RDS database**
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
[Connect an RDS database](#)
[Create a new RDS database](#)
- Create EBS snapshot policy**
Create a policy that automates the creation, retention, and deletion of EBS snapshots.
[Create EBS snapshot policy](#)

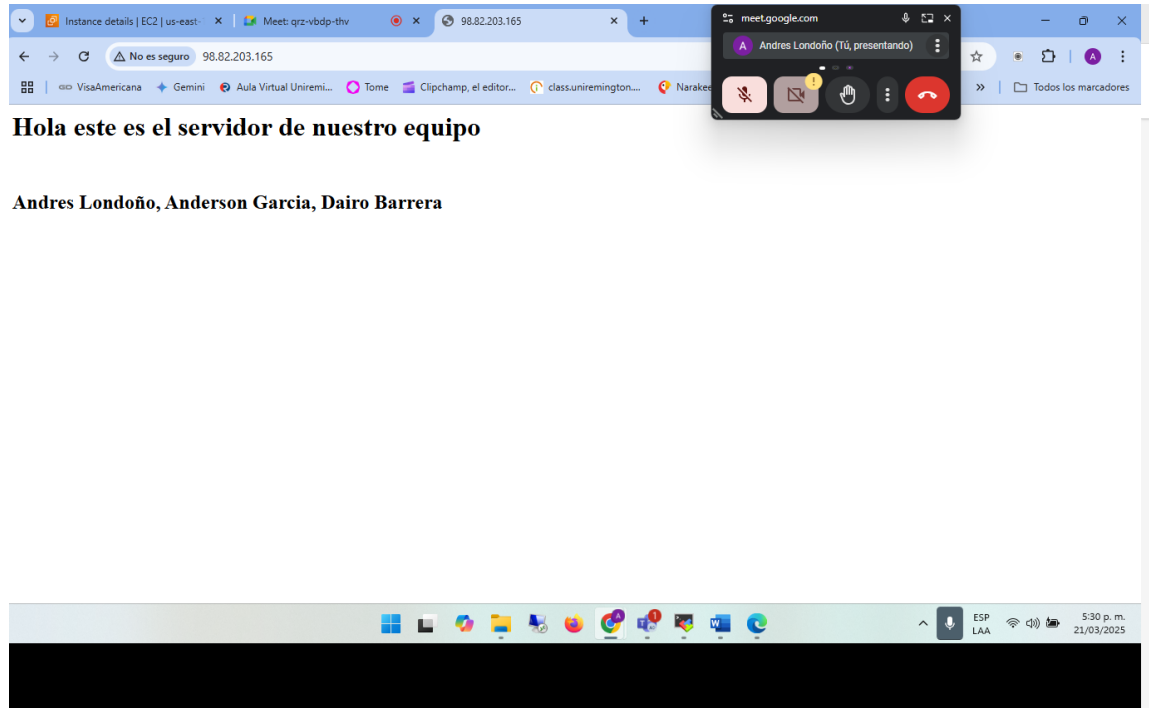
Como podemos observar en la siguiente imagen quedaría de la siguiente manera nuestra nueva Instancia AMI:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
LinuxV2InstanciaAMI	i-089bbfb4b9e6c3c85	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a
server1	i-0fa6ad9850dae1f7d	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a
LinuxServer	i-04c9bd3bdfad6bc8b	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a

Podemos validar que todo queda funcional:



En esta imagen estamos mostrando nuestro servidor corriendo:

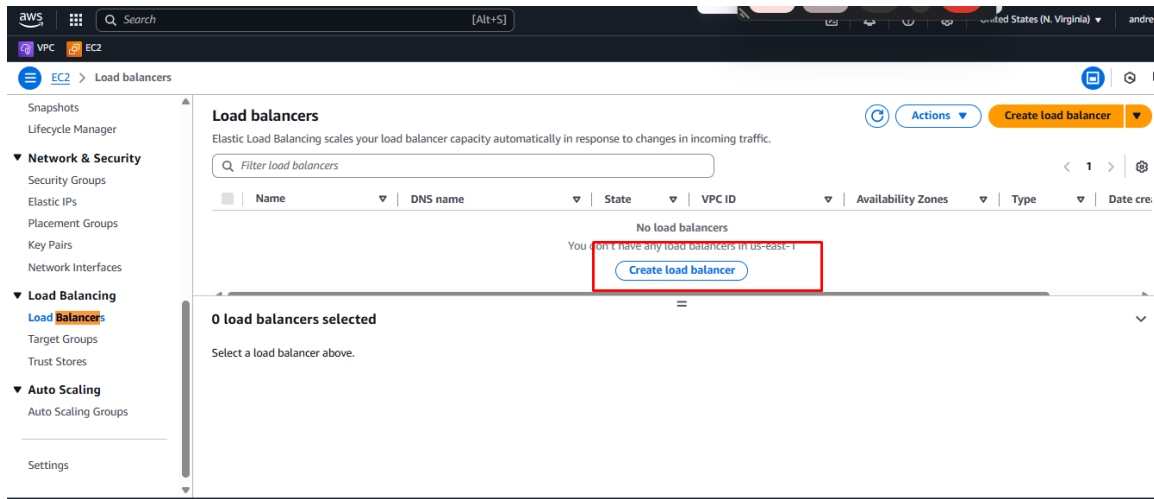


Hola este es el servidor de nuestro equipo

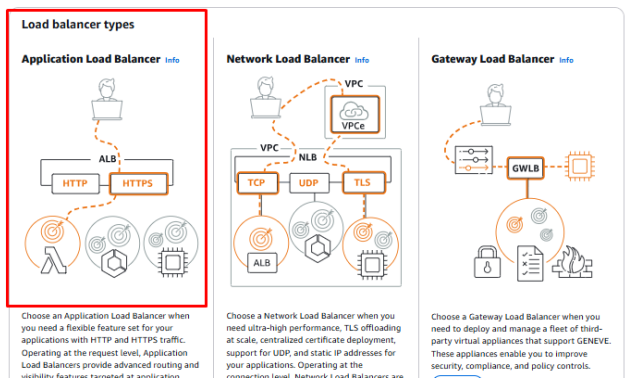
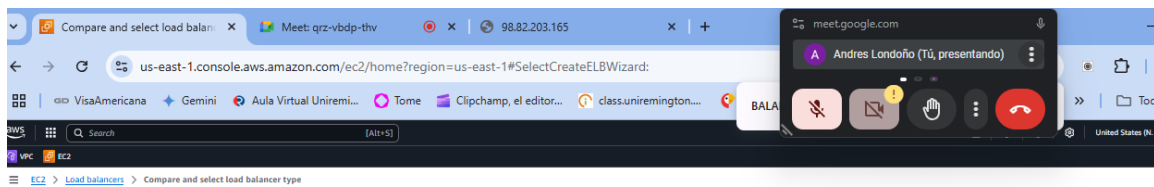
Andres Londoño, Anderson Garcia, Dairo Barrera

BALANCEADOR DE CARGA AUTOSCALING

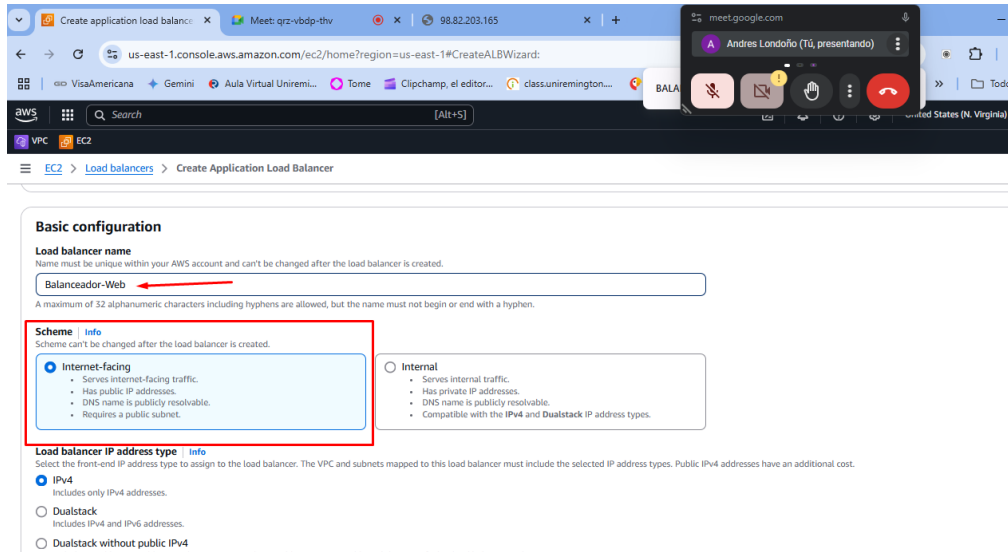
Creamos nuestro balanceador:



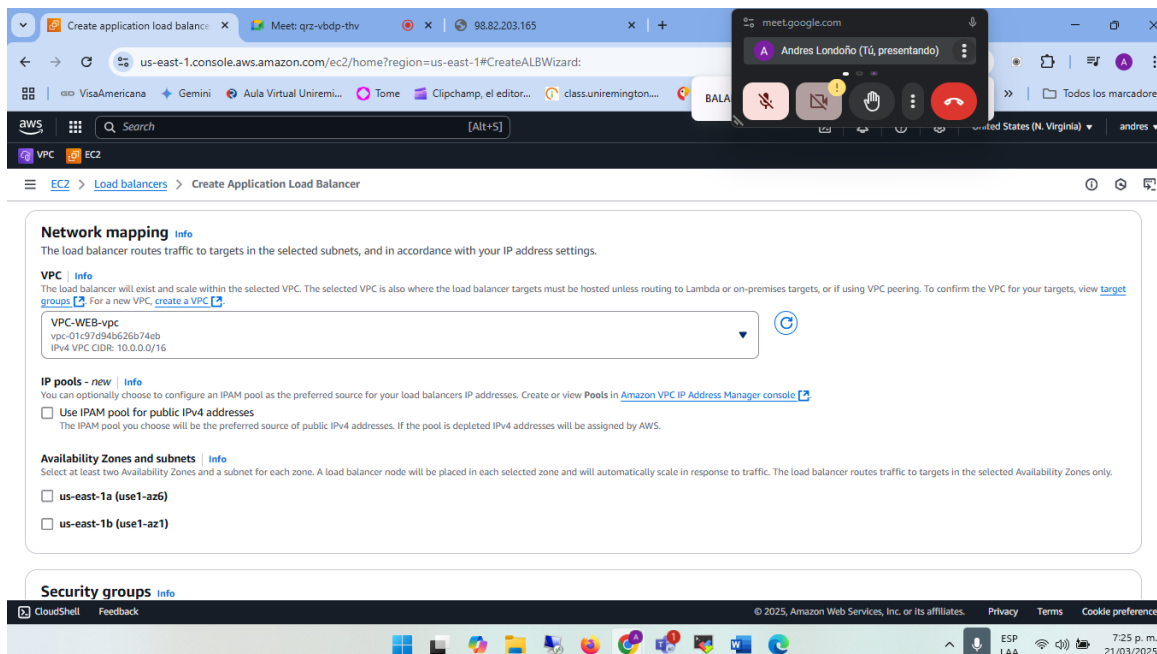
Seleccionamos el primer balanceador trae HTTP HTTPS :



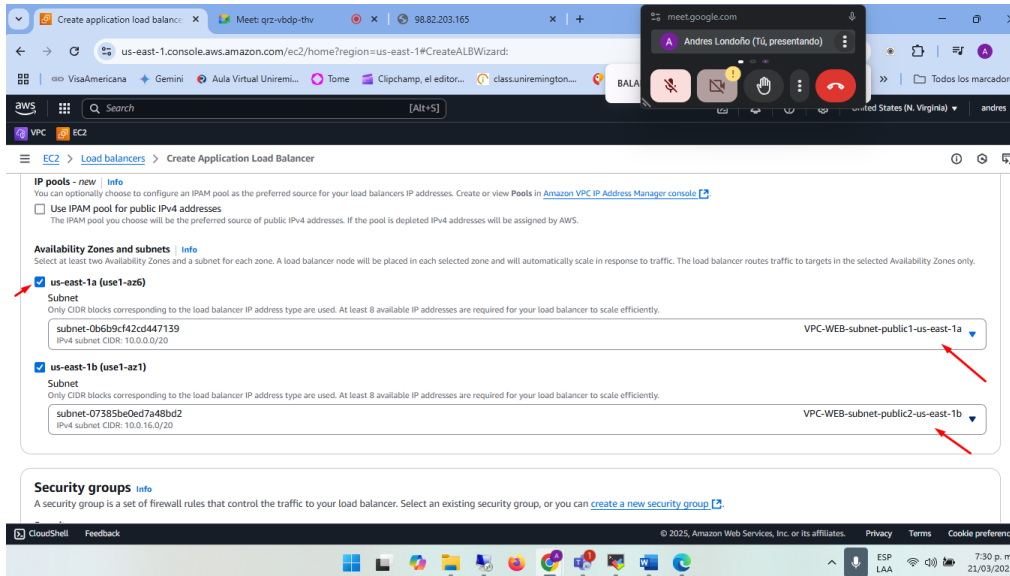
Le damos un nombre Ideal Acorde a las condiciones y seleccionamos El balanceador Externo que se puede acceder desde afuera(Internet):



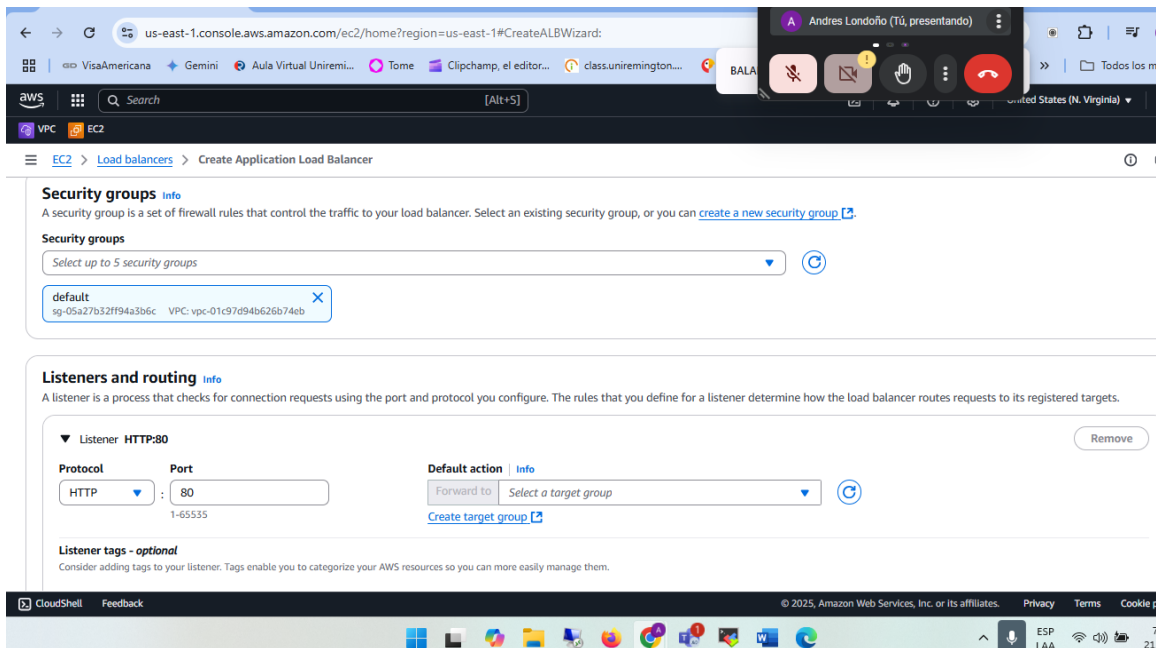
Seleccionamos nuestro VPC ya creado anteriormente:



Seleccionamos las subredes públicas de los datacenter para tener un dinamismo, esto con el fin de evitar que la información este en un solo datacenter:



Las peticiones las recibe el balanceador de carga en este caso por el puerto 80:



Creamos el target group:

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** Remo

Protocol: HTTP Port: 80

Default action: Forward to **Select a target group**

[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](#)

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:protocol=HTTP;vpc=0

EC2 > Target groups > Create target group

register targets

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

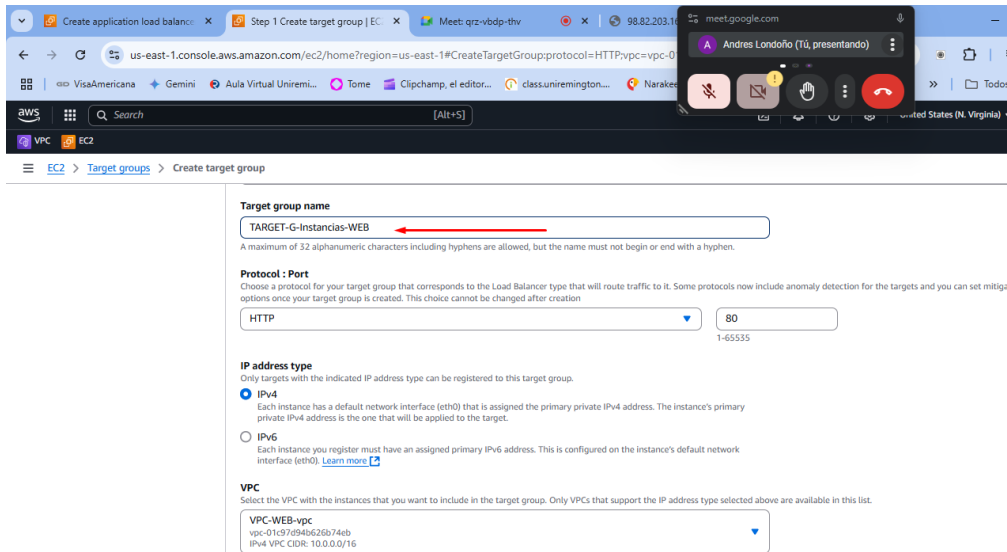
Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

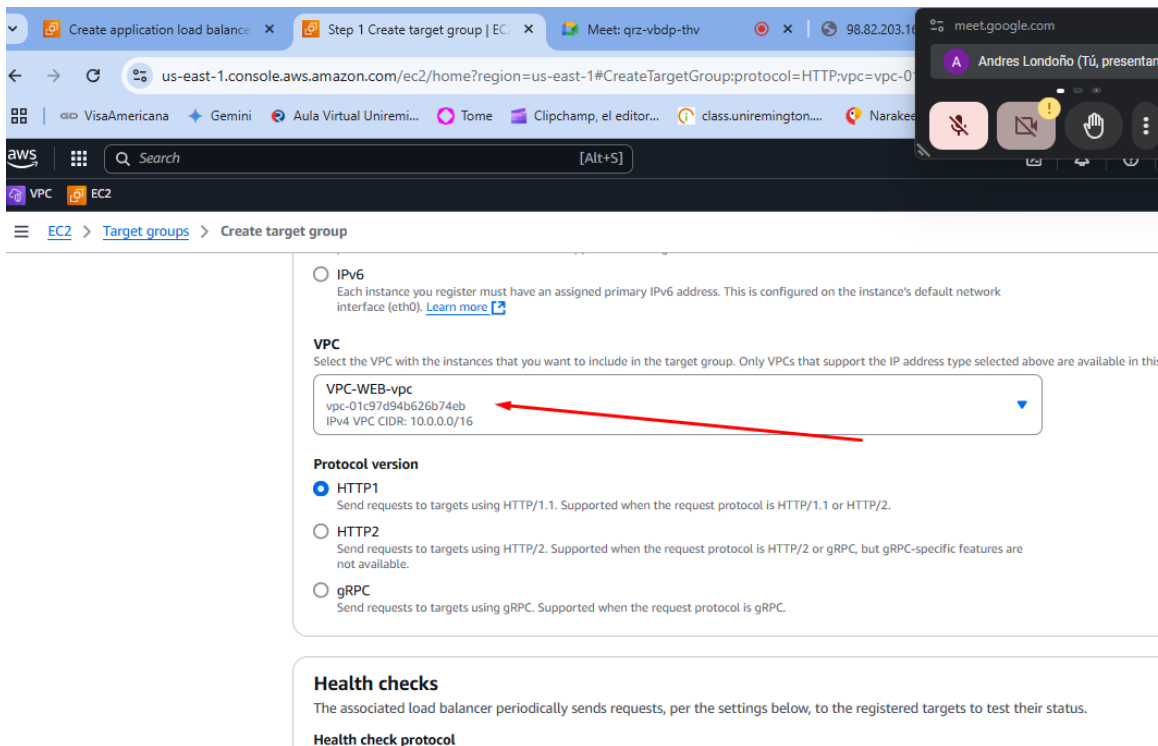
Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.

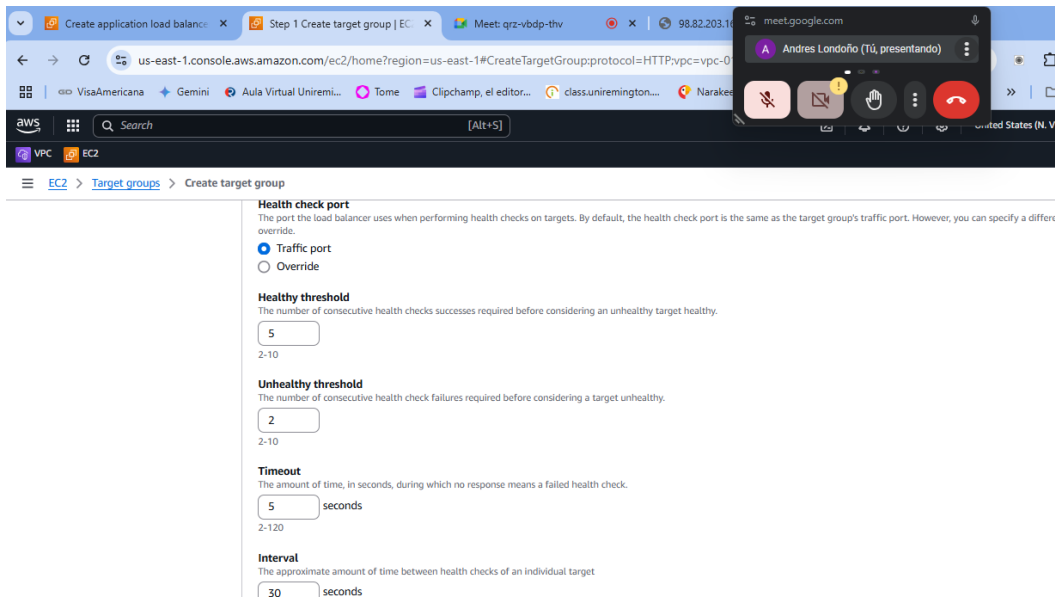
© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)



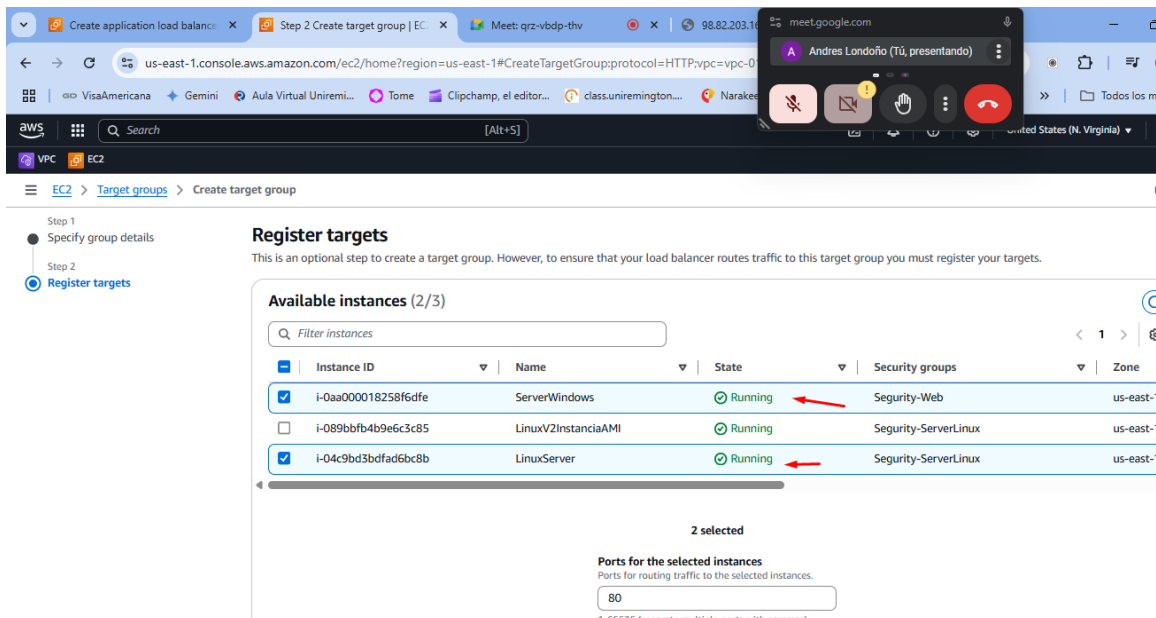
Se busca el VPC de las instancias que para nuestro caso es:

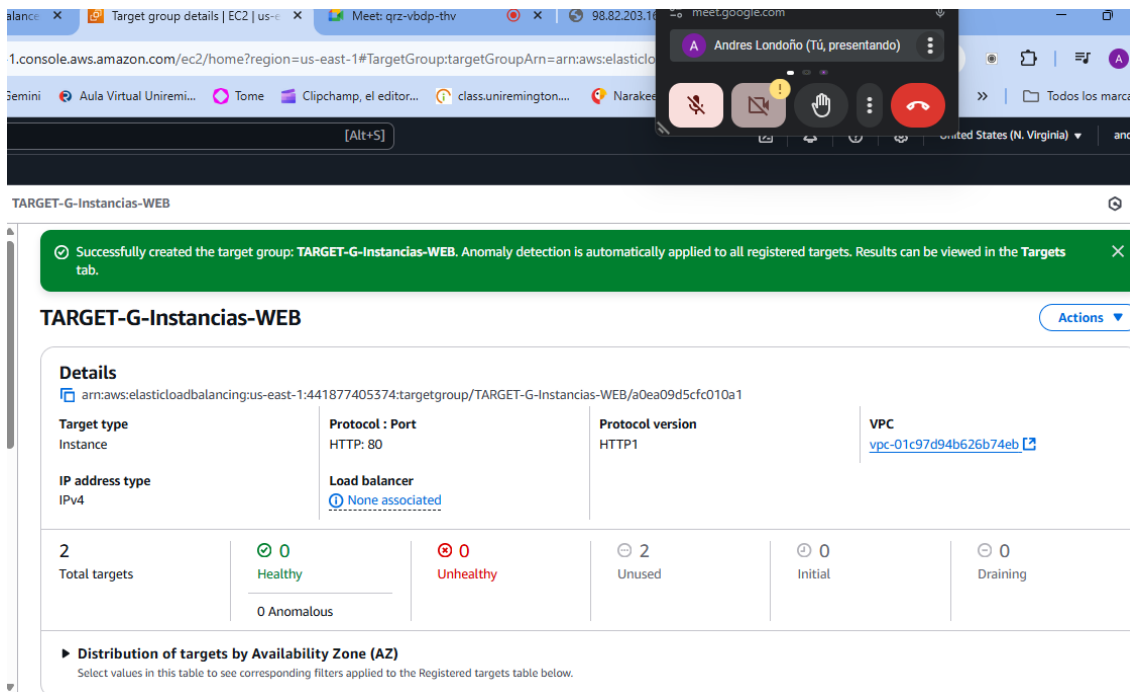


Valida por tiempos cada cuanto las instancias están funcionando(Activas) valida con el código 200, se configura según su criterio:

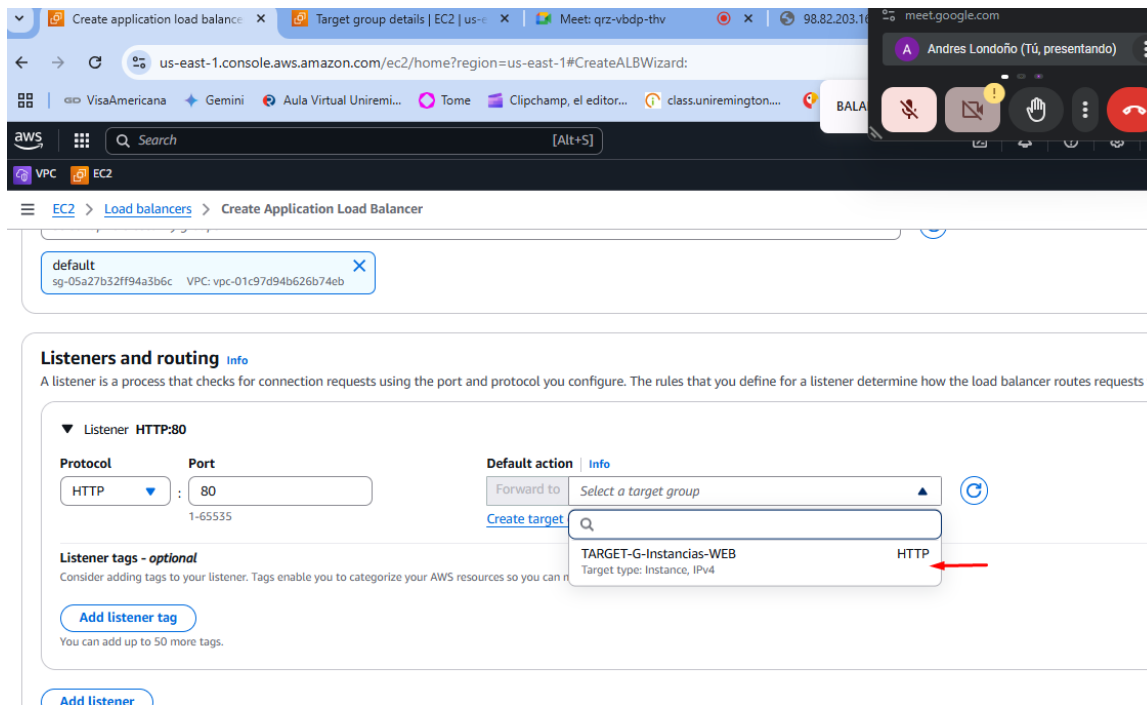


Seleccionamos las instancias a las cuales le aplicaremos la validación:





Volvemos al balanceador y escogemos el target anteriormente creado y quedaría de la siguiente manera:



The screenshot shows the AWS Management Console interface for a newly created load balancer. The main heading is "Balanceador-Web". A green success message at the top states: "Successfully created load balancer: Balanceador-Web. 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." Below this, a blue informational message notes: "Application Load Balancers now support public IPv4 IP Address Management (IPAM). You can get started with this feature by configuring IP pools in the Network mapping section." The "Details" section is expanded, showing the following information:

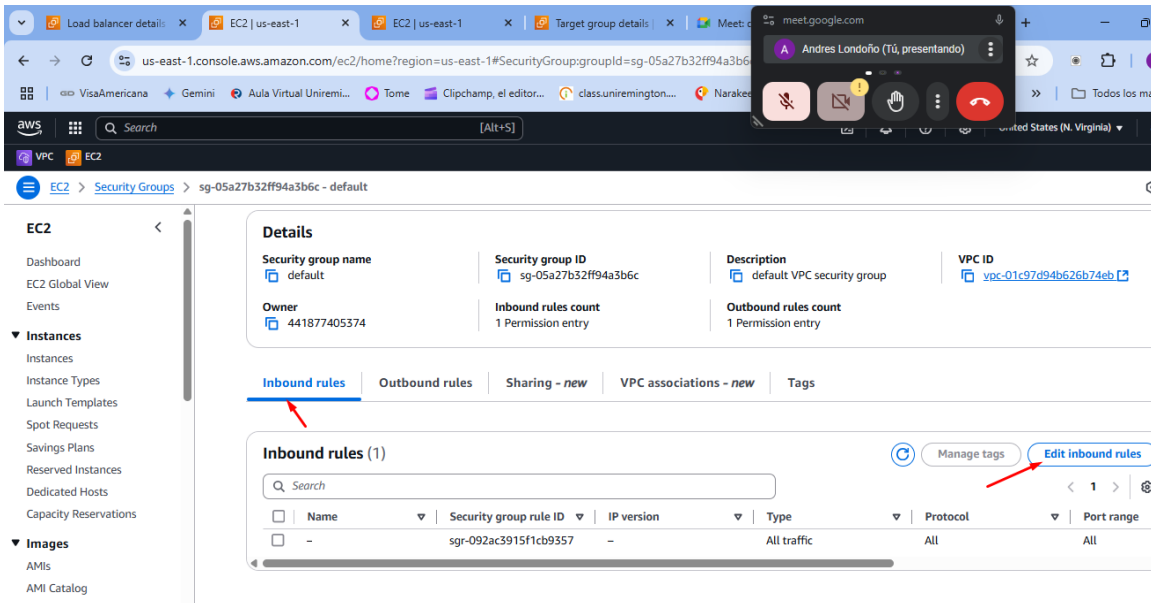
Property	Value
Load balancer type	Application
Status	Provisioning
VPC	vpc-01c97d94b626b74eb
Load balancer IP address type	IPv4
Scheme	Internet-facing
Hosted zone	Z355XDOTRQ7X7K
Availability Zones	subnet-0b6b9c42cd447139 us-east-1a (use1-az6) subnet-07385be0ed7a48bd2 us-east-1b (use1-az1)
Date created	March 21, 2025, 19:49 (UTC-05:00)
Load balancer ARN	
DNS name	

Vamos a las reglas de seguridad :

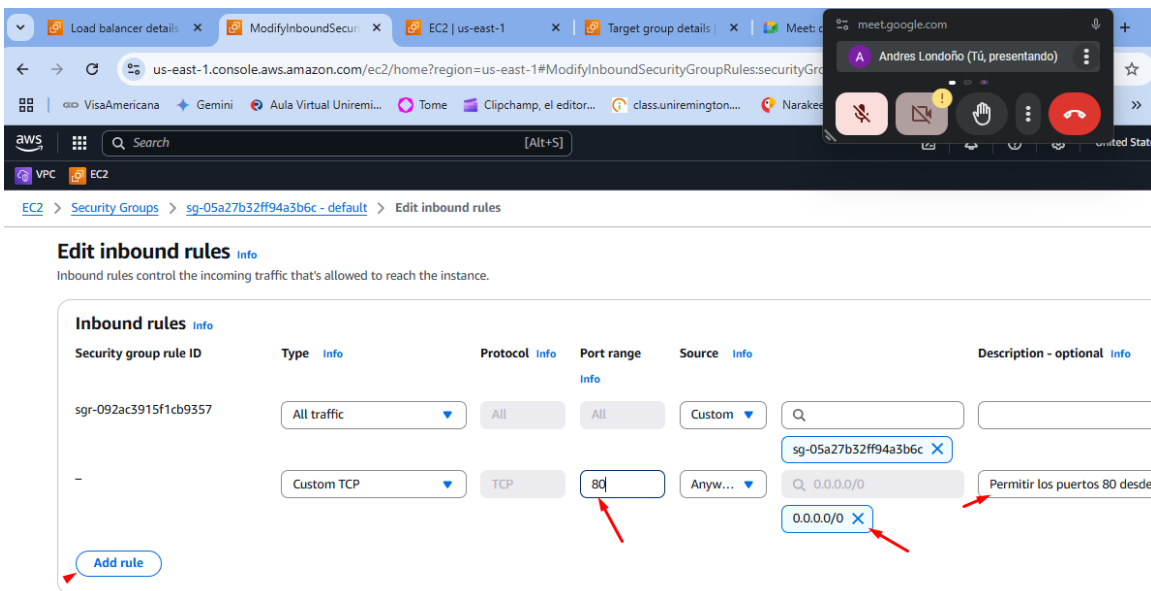
The screenshot shows the "Security" tab selected in the console. A red arrow points to the "Security" tab. Below the navigation bar, the "Security groups (1)" section is displayed. A red arrow points to the "Security Group ID" column in the table below.

A security group is a set of firewall rules that control the traffic to your load balancer.

Security Group ID	Name	Description
sg-05a27b32ff94a3b6c	default	default VPC security group



Creamos una regla para que el balanceador quede de la siguiente manera para permitir el tráfico por el puerto 80:



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#SecurityGroup:group-id=sg-05a27b32ff94a3b6c

EC2 > Security Groups > sg-05a27b32ff94a3b6c - default

Inbound security group rules successfully modified on security group (sg-05a27b32ff94a3b6c | default)

sg-05a27b32ff94a3b6c - default

Details			
Security group name default	Security group ID sg-05a27b32ff94a3b6c	Description default VPC security group	VPC ID vpc-01c97d94b626b74eb
Owner 441877405374	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules (2)

Así podemos garantizar que todos los servicios se encuentran activos y el balanceador activo:

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroup:targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:441877405374:targetgroup/TARGET-G-Instancias-WEB/a0ea09d5cfc010a1

EC2 > Target groups > TARGET-G-Instancias-WEB

Details

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-01c97d94b626b74eb
IP address type IPv4	Load balancer None associated		

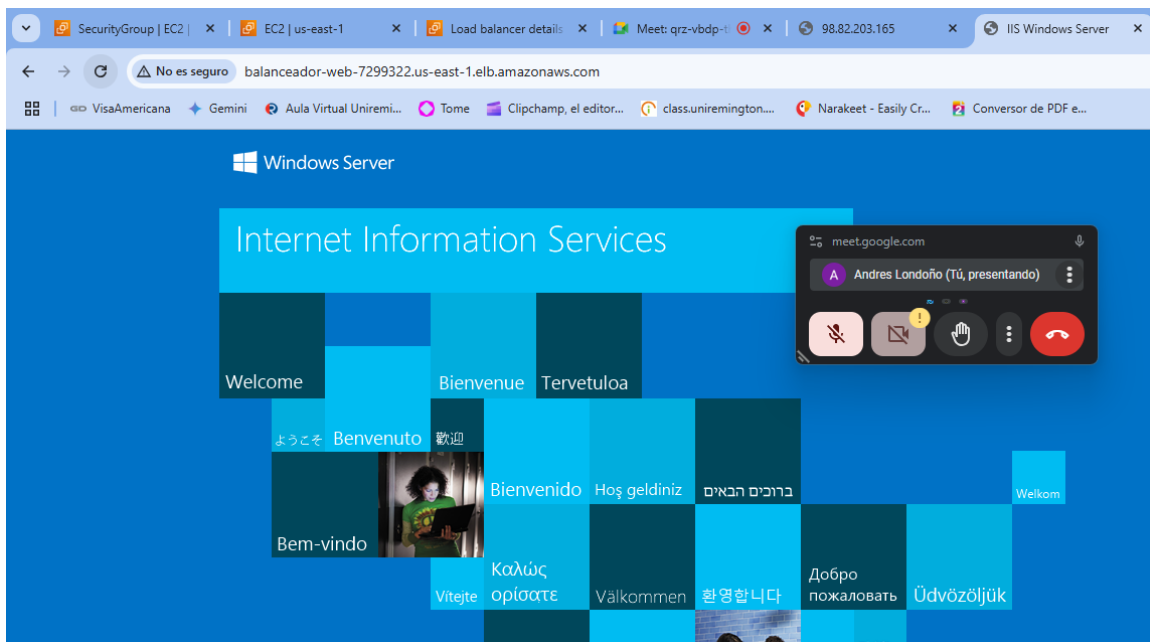
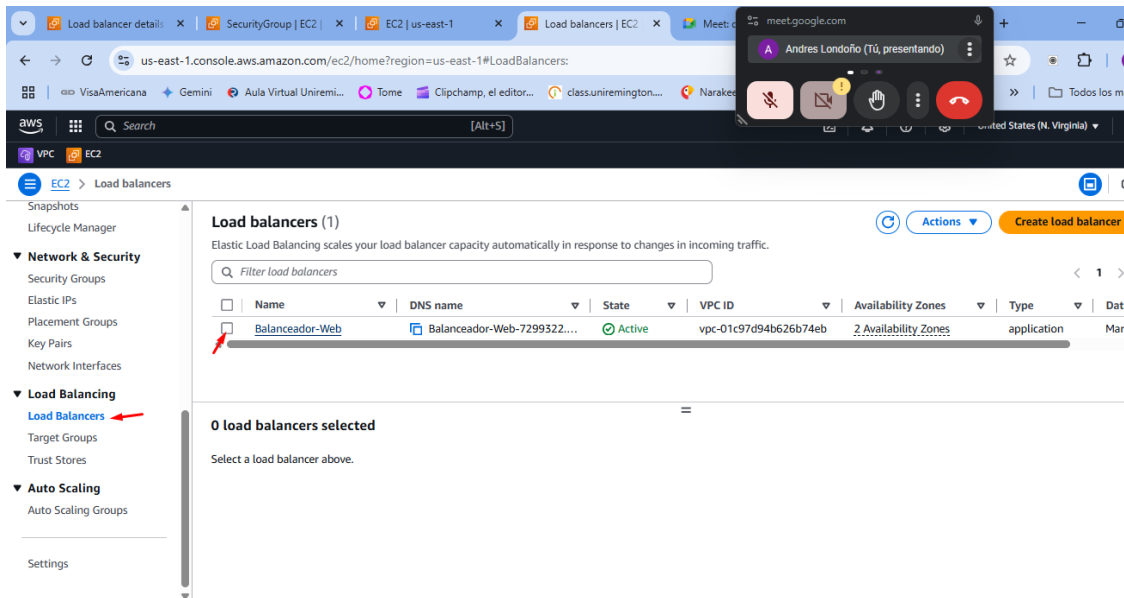
2 Total targets	2 Healthy 0 Anomalous	0 Unhealthy	0 Unused	0 Initial	0 Draining
---------------------------	------------------------------------	-----------------------	--------------------	---------------------	----------------------

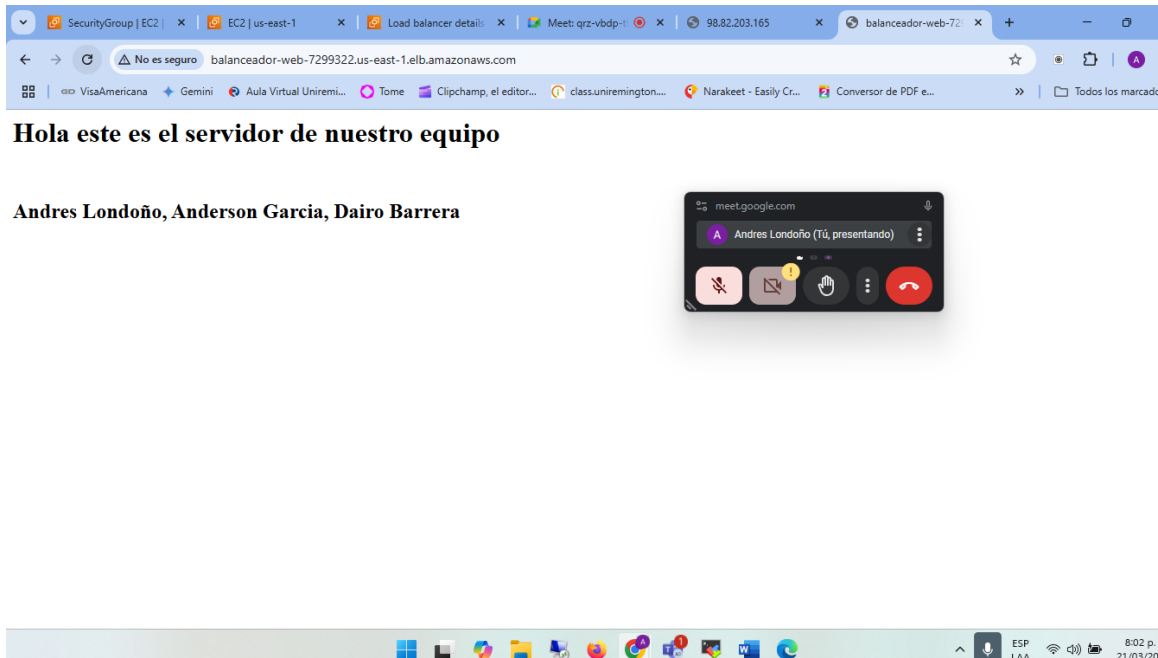
Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

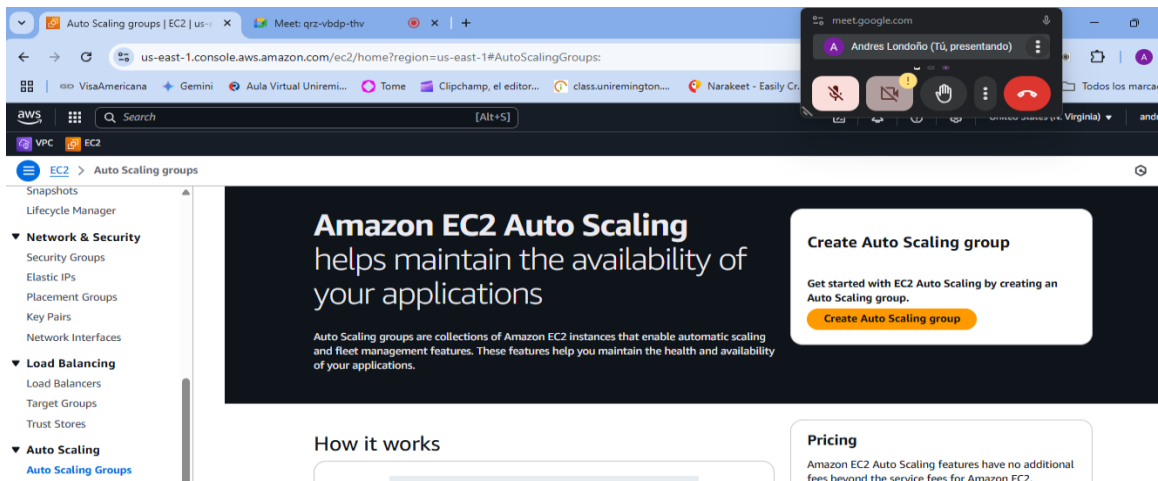
Targets | Monitoring | Health checks | Attributes | Tags

Seleccionamos Nuestro balanceador para poderlo identificar y sacar el DNS, dicho balanceador en este momento lo realizamos de manera manual: Balanceador-Web-7299322.us-east-1.elb.amazonaws.com:





Para hacerlo de manera automática con la ayuda de AWS se realizaría de la siguiente manera(AUTOSCALING):



Creamos un nombre y adicional creamos un template ya que no tenemos ninguno por el momento para brindarle los parámetros que se requieran:

The screenshot shows the 'Create Auto Scaling group' page in the AWS console. On the left, a progress bar indicates the current step is 'Step 4 - optional: Configure group size and scaling'. The main form has two sections: 'Name' and 'Launch template'. In the 'Name' section, the 'Auto Scaling group name' field contains 'AutoScalingWeb'. In the 'Launch template' section, a dropdown menu is set to 'Select a launch template', and a red arrow points to the 'Create a launch template' link below it.

EC2 > Launch templates > Create launch template

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. You can have multiple versions.

Launch template name and description

Launch template name - *required*

PlantillaWebAutoScaling

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

Template version description

Instancias para crearse de manera automatica AMI

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**

Seleccionamos la AMI correspondiente:

Owned by me

Shared with me



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

Amazon Machine Image (AMI)

AmazonLinuxYapache
ami-028e9a22631c1455d
2025-03-21T22:01:18.000Z

Virtualization: hvm ENA enabled: true Root device type: ebs Boot mode: uefi-preferred

▼

Description

-

Architecture	AMI ID
x86_64	ami-028e9a22631c1455d

Seleccionamos la instancia(server):

☰ [EC2](#) > [Launch templates](#) > Create launch template

Architecture	AMI ID
x86_64	ami-028e9a22631c1455d

▼ **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

Seleccionamos la clave anteriormente creada:


▼ **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

ClaveLinux

▼

 [Create new key pair](#)

▼ **Network settings** [Info](#)

No seleccionamos ninguna sub red:

▼ 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](#)

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

Common security groups | [Info](#)

Select security groups ▼ [Compare security group rules](#)

Security-ServerLinux sg-0037027ed92263955 ✕
VPC: vpc-01c97d94b626b74eb

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

Volvemos al 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.

AutoScaling

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

Launch template [Info](#)

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.

PlantillaWebAutoScaling ▼

Seleccionamos el VPC creado anteriormente y adicional agregamos las subredes privadas:

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across suitable for getting started quickly.

VPC

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

vpc-01c97d94b626b74eb (VPC-WEB-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-0e11378c95679c2c1 (VPC-WEB-subnet-private1-us-east-1a)
10.0.128.0/20

us-east-1b | subnet-0c2705865dc4f725b (VPC-WEB-subnet-private2-us-east-1b)
10.0.144.0/20

Asociamos el balanceador de carga:

The screenshot shows the AWS console interface for creating an Auto Scaling group. The 'Integrate with other services' step is active, with the following configuration:

- Load balancing:** Attach to an existing load balancer. Choose from your existing load balancers.
- Attach to an existing load balancer:** Choose from your load balancer target groups. This option allows you to attach Application, Network, or Gateway Load Balancers.
- Existing load balancer target groups:** TARGET-G-Instancias-WEB | HTTP (Application Load Balancer: Balanceador-Web)

Acá miramos las variables y las cantidades de instancias mínimas, Máximas y deseadas.
En el balanceador de carga:

The screenshot shows the AWS Management Console interface for creating an Auto Scaling group. The browser tabs include 'Create Auto Scaling group | EC2', 'Create launch template | EC2', and a Google Meet window. The console page is titled 'Create Auto Scaling group' and shows a progress bar with steps: Add notifications, Step 6 - optional, Add tags, Step 7, and Review. The 'Desired capacity' field is set to 2. The 'Scaling limits' section shows 'Min desired capacity' at 1 and 'Max desired capacity' at 4. The 'Automatic scaling - optional' section has 'No scaling policies' selected. The 'Instance maintenance policy' section is also visible.

Desired capacity
Specify your group size.
2 Valor deseado

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 1 Valor minimo
Equal or less than desired capacity

Max desired capacity 4 Valor Maximo
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.

Instance maintenance policy info
Control your Auto Scaling group's availability during instance replacement events. This includes health checks, instance refreshes, maximum instance lifetime features and events that happen automatically to keep your group balanced, called rebalancing events.

Choose a replacement behavior depending on your availability requirements

Review info

Step 1: Choose launch template Edit

Group details
Auto Scaling group name: AutoScaling

Launch template

Launch template	Version	Description
Plan4llaWebAutoScaling [i] lt-059ba419c795fa6df	Default	Instancias para crearse de manera automatica AMI

Step 2: Choose instance launch options Edit

Network

VPC: vpc-01c97d94b626b74eb [i]

Availability Zones and subnets

Availability Zone	Subnet	Subnet CIDR range
us-east-1a	subnet-0c11378c99679c2c1 [i]	10.0.128.0/20
us-east-1b	subnet-0c2705685dc4f725b [i]	10.0.144.0/20

Availability Zone distribution
Balanced best effort

Instance type requirements
This Auto Scaling group will adhere to the launch template.

Step 3: Integrate with other services Edit

Load balancing

Load balancer 1

Name	Type	Target group
Balanced-Web	Application/HTTP	TARGET-G-instancias-WEB

VPC Lattice integration options

VPC Lattice target groups

Application Recovery Controller (ARC) zonal shift

ARC zonal shift

Disabled

Health checks

Health check type	Health check grace period
EC2, ELB	300 seconds

Step 4: Configure group size and scaling policies Edit

Group size

Desired capacity	Desired capacity type
2	Units (number of instances)

Scaling

Minimum desired capacity	Maximum desired capacity
1	4

Target tracking policy

EC2 > Instances

Instances (5) Info Last updated less than a minute ago Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive) All states

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>		i-088aeecca2e296514	Running	t2.micro	Initializing	View alarms +	us-east-1b
<input type="checkbox"/>	LinuxV2InstanciaAMI	i-089bfb4b9e6c3c85	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a
<input type="checkbox"/>	ServerWindows	i-0aa00018258f6dfe	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a
<input type="checkbox"/>	LinuxServer	i-04c9bd3bdfad6bc8b	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a
<input type="checkbox"/>		i-04377c9e98dcac1b6	Running	t2.micro	Initializing	View alarms +	us-east-1a

Podemos validar que el autoScaling funciona correctamente:

aws Search [Alt+S] United States (N. Virginia) andres

EC2 > Auto Scaling groups

Auto Scaling groups (1/1) Info Launch configurations Launch templates Actions Create Auto Scaling group

Search your Auto Scaling groups

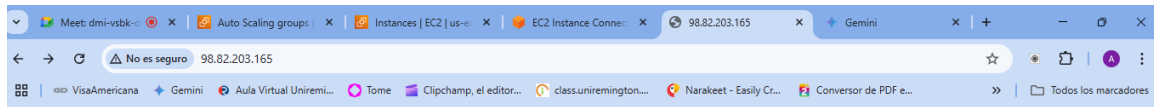
<input checked="" type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Ma
<input checked="" type="checkbox"/>	AutoScaling	PlantillaWebAutoScaling Version Latest	1	-	1	1	4

Auto Scaling group: AutoScaling

arn:aws:autoscaling:us-east-1:441877405374:autoScalingGroup:18d21107-5cad-4f65-b5b2-de0a9ac8f0be:autoScalingGroupName/AutoScaling

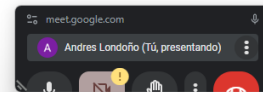
Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 4	Units (number of instances)	-

Date created
Fri Mar 21 2025 20:33:21 GMT-0500 (hora estándar de Colombia)

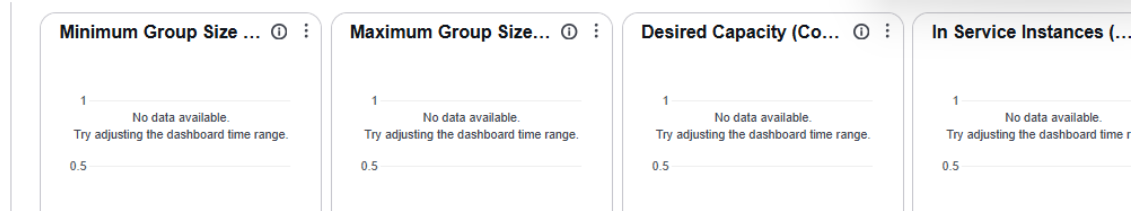


Hola este es el servidor de nuestro equipo

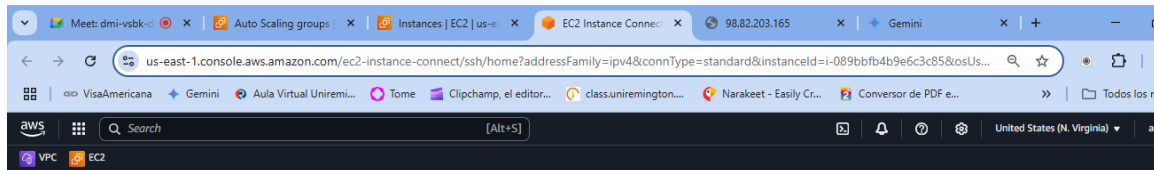
Andres Londoño, Anderson Garcia, Dairo Barrera



Auto Scaling group: AutoScaling



Validamos el rendimiento de la maquina:

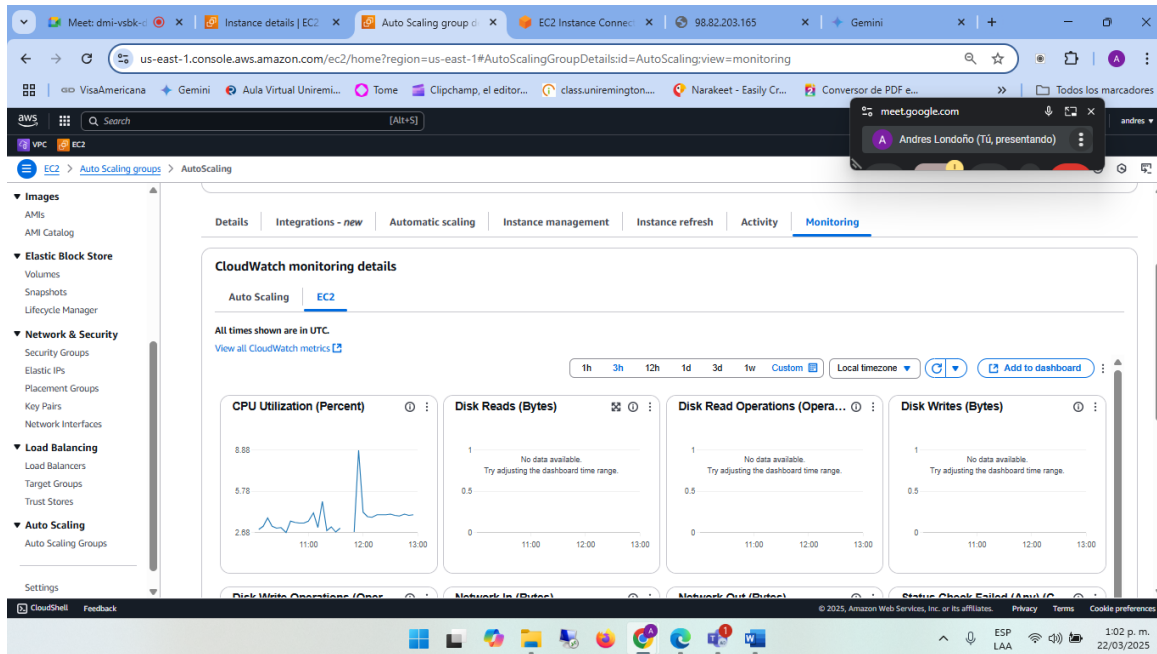


```

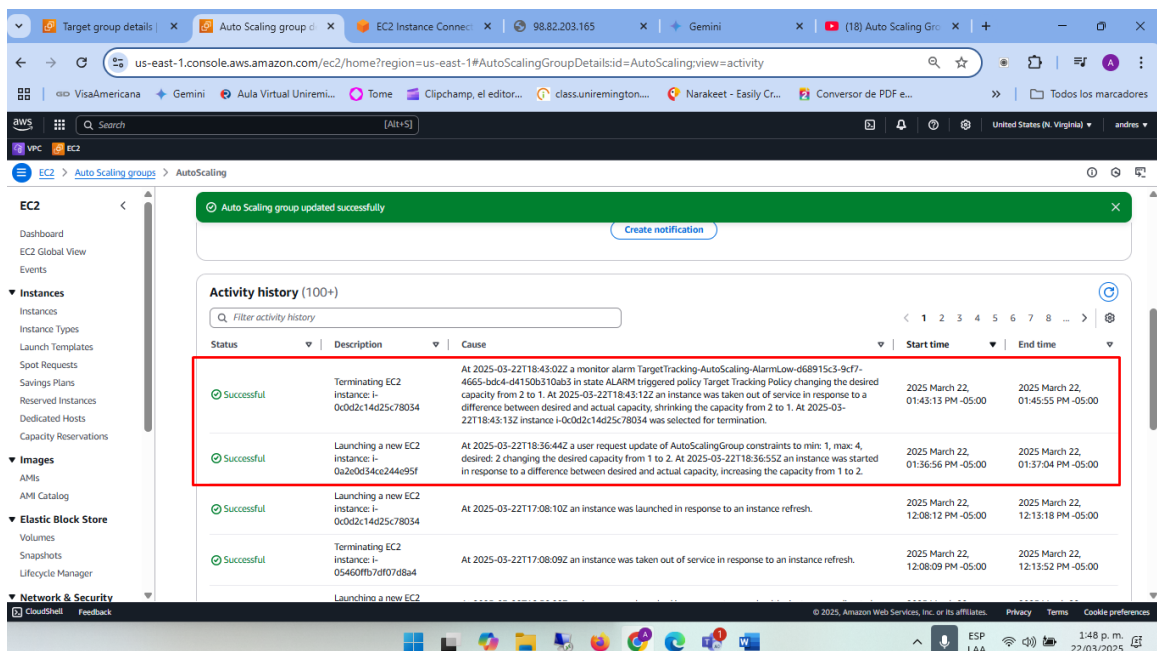
top - 17:44:47 up 19:30, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 104 total, 1 running, 103 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 93.7 id, 0.0 wa, 0.0 hi, 0.0 si, 6.0 st
MiB Mem : 949.5 total, 548.5 free, 162.3 used, 238.7 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 646.7 avail Mem

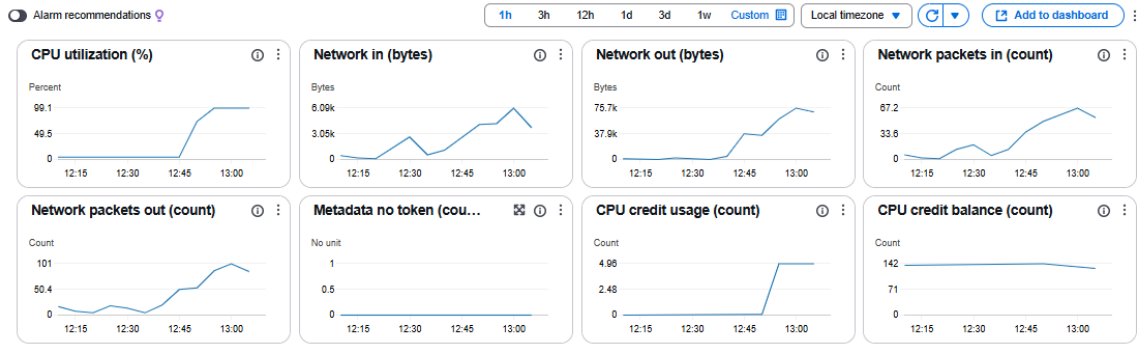
  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 65945 root      20   0 223908  3348 2692  R  0.3  0.3   0:00.01 top
    1 root      20   0 106092 17308 10596  S  0.0  1.8   0:06.99 systemd
    2 root      20   0 0        0      0      S  0.0  0.0   0:00.01 kthreadd
    3 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 rcu_gp
    4 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 rcu_par_gp
    5 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 slab_flushwq
    6 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 netns
    8 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 kworker/0:0H-eventfs_highpri
   10 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 mm_percpu_wq
   11 root      20   0 0        0      0      I  0.0  0.0   0:00.00 rcu_tasks_kthread
   12 root      20   0 0        0      0      I  0.0  0.0   0:00.00 rcu_tasks_rude_kthread
   13 root      20   0 0        0      0      I  0.0  0.0   0:00.00 rcu_tasks_trace_kthread
   14 root      20   0 0        0      0      S  0.0  0.0   0:01.43 ksoftirqd/0
   15 root      20   0 0        0      0      I  0.0  0.0   0:00.69 rcu_preempt
   16 root      rt   0 0        0      0      S  0.0  0.0   0:00.40 migration/0
   18 root      20   0 0        0      0      S  0.0  0.0   0:00.00 cpuhp/0
   20 root      20   0 0        0      0      S  0.0  0.0   0:00.00 kdevtmpfs
   21 root      0 -20  0        0      0      I  0.0  0.0   0:00.00 inet_frag_wq
    
```

i-089bbfb4b9e6c3c85 (LinuxV2InstanciaAMI)



Como podemos observar después de enviar el comando para saturar las instancias:





Segundo entregable practico Docker

Ingresamos como administrador para poder ejecutar los comandos necesarios para la instalación de Docker y la desinstalación del Http, en la siguiente imagen se muestran los comandos necesarios para poder hacer dicha instalación.

```

54.174.29.246 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
/home/ec2-user/
Name
..
.csh
.bash_logout
.bash_profile
.bashrc

--> Running transaction check
--> Package containerd.x86_64 0:1.7.25-1.amzn2.0.1 will be installed
--> Package libcgroupp.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.2.4-1.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
docker x86_64 25.0.8-1.amzn2.0.1 amzn2extra-docker 45 M
Installing for dependencies:
containerd x86_64 1.7.25-1.amzn2.0.1 amzn2extra-docker 31 M
libcgroupp x86_64 0.41-21.amzn2 amzn2-core 66 k
pigz x86_64 2.3.4-1.amzn2.0.1 amzn2-core 81 k
runc x86_64 1.2.4-1.amzn2 amzn2extra-docker 3.4 M
=====
Transaction Summary
-----
Install 1 Package (+4 Dependent packages)
Total download size: 79 M
Installed size: 286 M
Is this ok [y/d/N]: y
Downloading packages:
(1/5): libcgroupp-0.41-21.amzn2.x86_64.rpm | 66 kB 00:00:00
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm | 81 kB 00:00:00
(3/5): containerd-1.7.25-1.amzn2.0.1.x86_64.rpm | 31 MB 00:00:00
(4/5): runc-1.2.4-1.amzn2.x86_64.rpm | 3.4 MB 00:00:00
(5/5): docker-25.0.8-1.amzn2.0.1.x86_64.rpm | 45 MB 00:00:01
-----
Total 68 MB/s | 79 MB 00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : runc-1.2.4-1.amzn2.x86_64 1/5
Installing : containerd-1.7.25-1.amzn2.0.1.x86_64 2/5
Installing : libcgroupp-0.41-21.amzn2.x86_64 3/5
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
Installing : docker-25.0.8-1.amzn2.0.1.x86_64 5/5
Verifying : runc-1.2.4-1.amzn2.x86_64 1/5
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 2/5
Verifying : libcgroupp-0.41-21.amzn2.x86_64 3/5
Verifying : containerd-1.7.25-1.amzn2.0.1.x86_64 4/5
Verifying : docker-25.0.8-1.amzn2.0.1.x86_64 5/5

Installed:
docker.x86_64 0:25.0.8-1.amzn2.0.1

Dependency Installed:
containerd.x86_64 0:1.7.25-1.amzn2.0.1 libcgroupp.x86_64 0:0.41-21.amzn2
pigz.x86_64 0:2.3.4-1.amzn2.0.1 runc.x86_64 0:1.2.4-1.amzn2

Complete!
[root@ip-172-31-83-172 ec2-user]#

Complete!
[root@ip-172-31-83-172 ec2-user]# yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[root@ip-172-31-83-172 ec2-user]# yum update

```

Actualizamos los paquetes con el fin de validar que todos los componentes se cargaron de manera exitosa:

```

Installed:
  docker.x86_64 0:25.0.8-1.amzn2.0.1

Dependency Installed:
  containerd.x86_64 0:1.7.25-1.amzn2.0.1      libcgrou.p.x86_64 0:0.41-21.amzn2
  pigz.x86_64 0:2.3.4-1.amzn2.0.1          runc.x86_64 0:1.2.4-1.amzn2

Complete!
[root@ip-172-31-83-172 ec2-user]# yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
[root@ip-172-31-83-172 ec2-user]# yum install docker.io
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.6 kB 00:00:00
No package docker.io available.
Error: Nothing to do
[root@ip-172-31-83-172 ec2-user]# yum install docker.io
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No package docker.io available.
Error: Nothing to do
[root@ip-172-31-83-172 ec2-user]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
   Docs: https://docs.docker.com
[root@ip-172-31-83-172 ec2-user]#

```

Después de dicha actualización Validamos que este corriendo de manera correcta, el servicio debe de estar activo y funcionando, El comando para correrlo es Start status :

```

[root@ip-172-31-83-172 ec2-user]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
   Docs: https://docs.docker.com
[root@ip-172-31-83-172 ec2-user]# systemctl start docker
[root@ip-172-31-83-172 ec2-user]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2025-03-28 21:48:04 UTC; 53s ago
   Docs: https://docs.docker.com
   Process: 32610 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Process: 32609 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
   Main PID: 32613 (dockerd)
   Tasks: 0
   Memory: 38.1M
   CGroup: /system.slice/docker.service
           └─32613 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Mar 28 21:48:03 ip-172-31-83-172.ec2.internal systemd[1]: Starting Docker Application Container Engine...
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.656747657Z" level=info msg="Starting up"
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.78792998Z" level=info msg="Loading containers: start."
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.97985564Z" level=info msg="Loading containers: done."
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.992175897Z" level=warning msg="WARNING: bridge-nf-call-iptables is disabled"
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.992513954Z" level=warning msg="WARNING: bridge-nf-call-ip6tables is disabled"
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.992695866Z" level=info msg="Docker daemon" commit=71907ca containerd-snapshotter=false storage-driver=...rs tom=25.0.8
Mar 28 21:48:03 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:03.992748127Z" level=info msg="Daemon has completed initialization"
Mar 28 21:48:04 ip-172-31-83-172.ec2.internal dockerd[32613]: time="2025-03-28T21:48:04.036883345Z" level=info msg="API Listen on /run/docker.sock"
Mar 28 21:48:04 ip-172-31-83-172.ec2.internal systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[root@ip-172-31-83-172 ec2-user]#

```

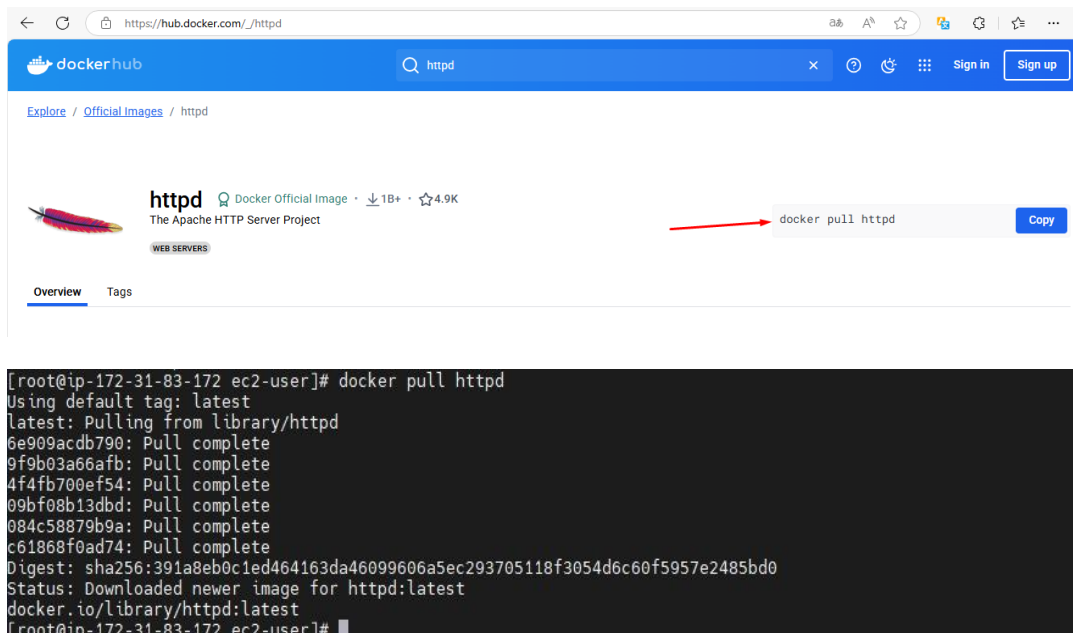
Para activar el servicio y que arranque de manera automática ya debe de estar Corriendo, copiamos el siguiente comando systemctl enable docker:

```

[root@ip-172-31-83-172 ec2-user]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[root@ip-172-31-83-172 ec2-user]#

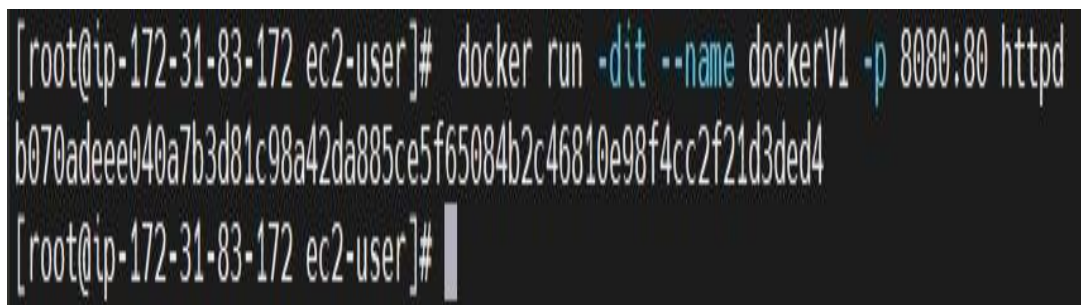
```

Para utilizar la imagen de http la cual encontramos en la siguiente ruta https://hub.docker.com/_/httpd de dicha página necesitamos ingresar el siguiente comando docker pull httpd, en este caso la imagen es de httpd:



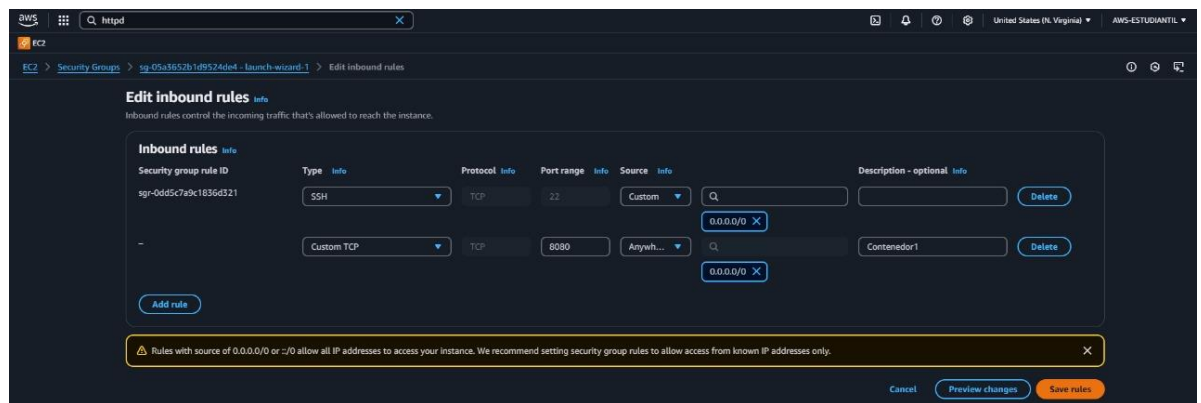
```
[root@ip-172-31-83-172 ec2-user]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
6e909acdb790: Pull complete
9f9b03a66afb: Pull complete
4f4fb700ef54: Pull complete
09bf08b13dbd: Pull complete
084c58879b9a: Pull complete
c61868f0ad74: Pull complete
Digest: sha256:391a8eb0c1ed464163da46099606a5ec293705118f3054d6c60f5957e2485bd0
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@ip-172-31-83-172 ec2-user]#
```

Después de haber utilizado el servicio de la imagen docker+httpd, Para crear el contenedor utilizamos el comando run, cada contenedor tiene un nombre único por lo tanto el puerto 8080 en este caso no se puede repetir en otro contenedor, ya que también es único y lo que se podría repetir es el puerto 80(si desea):

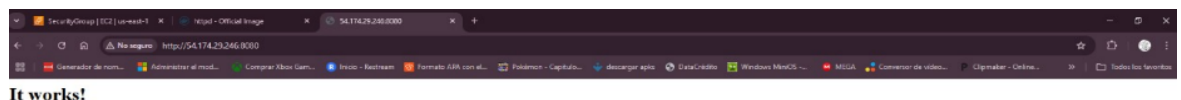


```
[root@ip-172-31-83-172 ec2-user]# docker run -dit --name dockerV1 -p 8080:80 httpd
b070adeee040a7b3d81c98a42da885ce5f65084b2c46810e98f4cc2f21d3ded4
[root@ip-172-31-83-172 ec2-user]#
```

Habilitamos en nuestra instancia (server linux) el puerto 8080 del primer contenedor esto con el fin de poder garantizar el tráfico libre por dicho puerto, es decir, sin ninguna restricción:



Abrimos un navegador para validar que todo funciona perfectamente, es decir, validar que la IP de la instancia + el puerto en este caso 8080:80 no tenga ninguna restricción y nos redirija a un contenedor en específico, para demostrarlo nos saldrá en la página el siguiente mensaje:



Volvemos a nuestra consola de administración para Crear una carpeta en la ruta del usuario ec2 de la siguiente manera, utilizamos el comando relacionado en la siguiente imagen:

```
[root@ip-172-31-83-172 ec2-user]# mkdir app1
[root@ip-172-31-83-172 ec2-user]# cd app1/
[root@ip-172-31-83-172 app1]# █
```

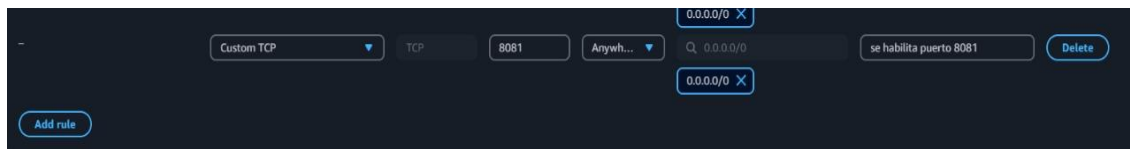
Después de estar posicionados en la carpeta App1, dentro de ella Creamos nuestro primer archivo el cual será denominado index.html , dicho archivo llevará nuestra primera página para el 1 contenedor de la siguiente manera:

```
[root@ip-172-31-83-172 ec2-user]# mkdir app1
[root@ip-172-31-83-172 ec2-user]# cd app1/
[root@ip-172-31-83-172 app1]# nano index.html
[root@ip-172-31-83-172 app1]# █
```

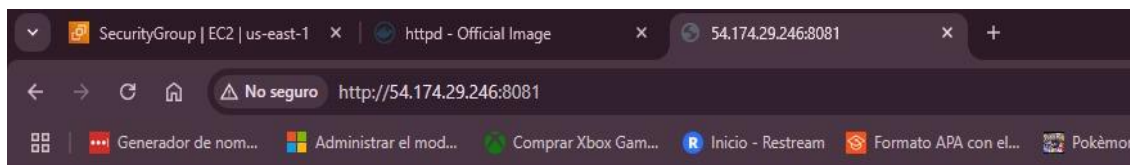
Creamos un enlace simbólico en el cual corremos el docker haciendo referencia a la carpeta anteriormente mencionada (App1) y a su puerto respectivamente, para que apache lo encuentre y poder tener nuestra rutas(carpetas) separada del contenedor y tener los archivos en un lugar diferente, tenerlo en el aws, Esto nos ayuda a tener un contenedor dinámico ya que tenemos los archivos separados.

```
[root@ip-172-31-83-172 ec2-user]# docker run -dit --name app1 -p 8081:80 -v /home/ec2-user/app1:/usr/local/apache2/htdocs/ httpd
0dd42cd664ba7b02ad94c741b62f6de096833b3cd032a07313d99e824f27a602
[root@ip-172-31-83-172 ec2-user]# █
```

Se habilita el tráfico en el puerto 8081 esto con el fin de dar vía libre a todo el tráfico sin que se genere alguna restricción de la siguiente manera:



Acá creamos nuestro primer contenedor con un archivo dinámico creado por nosotros el cual apunta a la IP “cabe aclarar que la IP puede variar ya que es un servicio que requiere de un pago, según políticas de aws” de la instancia con el puerto 8081 anteriormente habilitado, el resultado es:



Hola, estamos en el contenedor 1 Andres, Anderson, Dairo

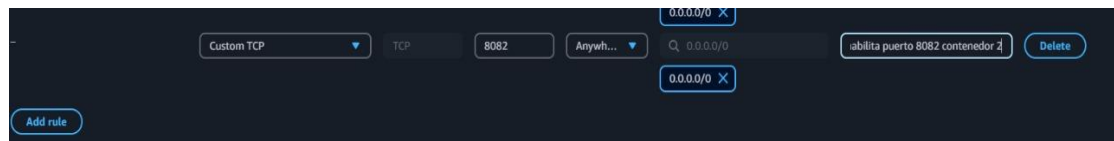
Volvemos a la consola y esta vez volvemos a pararnos en la ruta del usuario Ec2, Creamos otra carpeta llamada app2 con todos los permisos para evitar errores más adelante, en dicha carpeta debemos crear un archivo el cual llamaremos index.html” al hacer validaciones el index2.html nos genera error, por lo cual decidimos renombrarla a index.html”, el cual tiene un contenido totalmente diferente y apunta a otro contenedor:

```
[root@ip-172-31-83-172 ec2-user]# mkdir app2
[root@ip-172-31-83-172 ec2-user]# cd app2/
[root@ip-172-31-83-172 app2]# nano index2.html
[root@ip-172-31-83-172 app2]# chmod -R 777 app2/
```

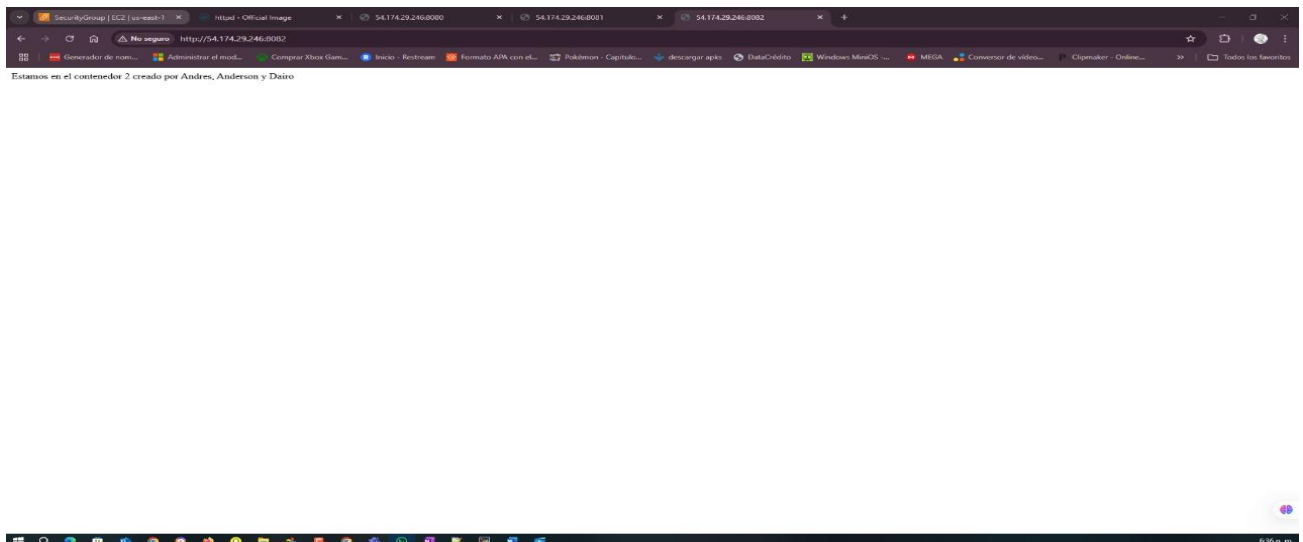
Creamos un enlace simbólico en el cual corremos el docker haciendo referencia a la carpeta anteriormente mencionada (App2) y a su puerto respectivamente, para que apache lo encuentre y poder tener nuestra ruta(carpetas) separada del contenedor y tener los archivos en un lugar diferente, tenerlo en el aws, Esto nos ayuda a tener un contenedor dinámico ya que tenemos los archivos separados.

```
[root@ip-172-31-83-172 app2]# cd ..
[root@ip-172-31-83-172 ec2-user]# chmod -R 777 app2/
[root@ip-172-31-83-172 ec2-user]# docker run -dit --name app2 -p 8082:80 -v /home/ec2-user/app2/:/usr/local/apache2/htdocs/ httpd
bbfe716c2bd02e9adcb5f4e69a9c9ebc6e6eafb28a5f7f2ac fb0f0279a91626a
[root@ip-172-31-83-172 ec2-user]#
```

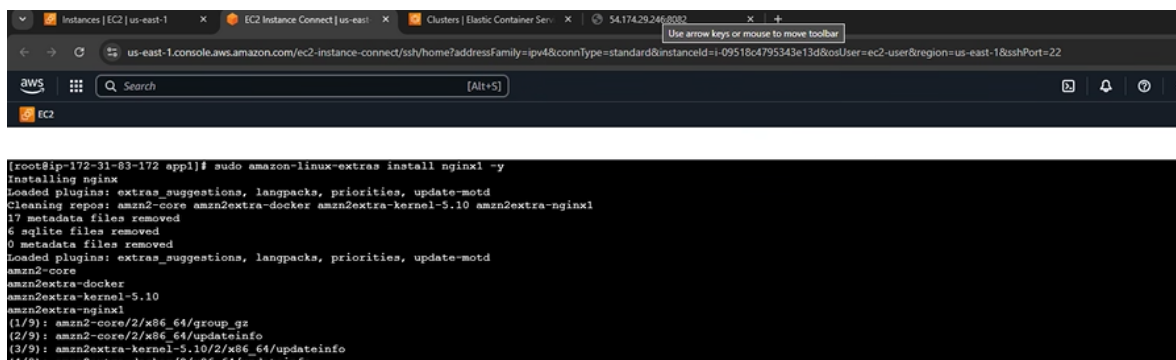
Habilitamos el puerto 8082 esto con el fin de habilitar el tráfico sin ningún problema y sin ninguna restricción para el siguiente contenedor el cual muestra un index.html totalmente diferente:



Al abrir la IP con el puerto 8082 podemos visualizar que la página abre con toda normalidad y quedaría de la siguiente manera:

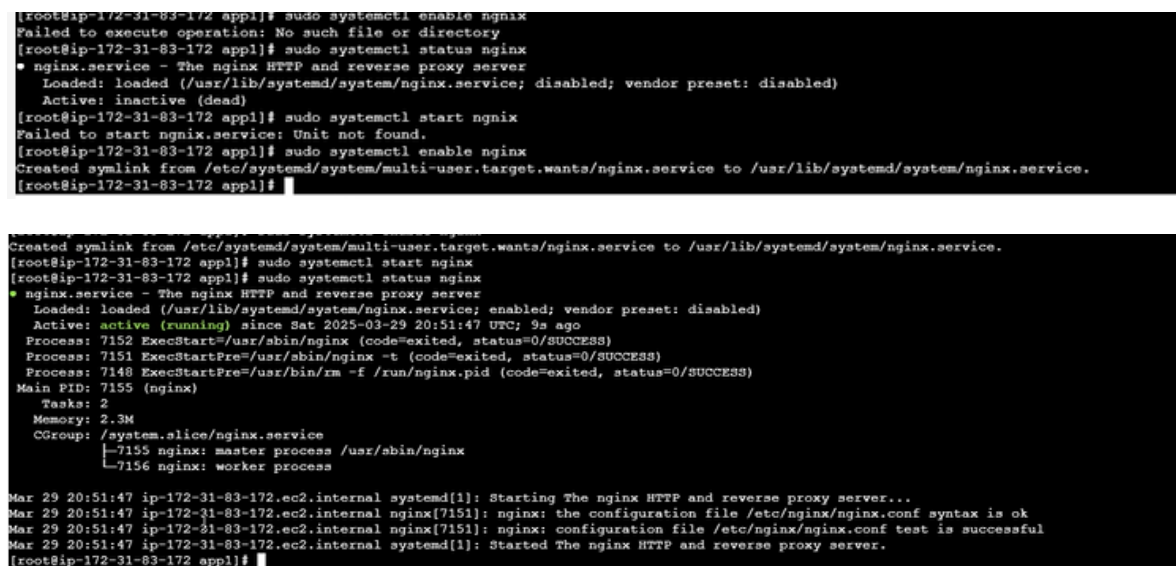


Instalamos ngxex esto con el fin de poder generar el proxy reverso, es decir, poder utilizar las rutas sin necesidad de ingresar con un puerto en específico, se haría de la siguiente manera:



```
[root@ip-172-31-83-172 appl]# sudo amazon-linux-extras install nginx1 -y
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10 amzn2extra-nginx1
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
amzn2extra-kernel-5.10
amzn2extra-nginx1
(1/9): amzn2-core/2/x86_64/group_gz
(2/9): amzn2-core/2/x86_64/updateinfo
(3/9): amzn2extra-kernel-5.10/2/x86_64/updateinfo
(4/9): amzn2extra-docker/2/x86_64/updateinfo
```

Después lo habilitamos en el sistema para que corra de manera automática cuando la maquina se apague suban los servicios sin necesidad de líneas de código:



```
[root@ip-172-31-83-172 appl]# sudo systemctl enable nginx
Failed to execute operation: No such file or directory
[root@ip-172-31-83-172 appl]# sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[root@ip-172-31-83-172 appl]# sudo systemctl start nginx
Failed to start nginx.service: Unit not found.
[root@ip-172-31-83-172 appl]# sudo systemctl enable nginx
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.
[root@ip-172-31-83-172 appl]#
```



```
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.
[root@ip-172-31-83-172 appl]# sudo systemctl start nginx
[root@ip-172-31-83-172 appl]# sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2025-03-29 20:51:47 UTC; 9s ago
     Process: 7152 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
     Process: 7151 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 7148 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
   Main PID: 7155 (nginx)
      Tasks: 2
     Memory: 2.3M
     CGroup: /system.slice/nginx.service
            └─7155 nginx: master process /usr/sbin/nginx
               └─7156 nginx: worker process

Mar 29 20:51:47 ip-172-31-83-172.ec2.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
Mar 29 20:51:47 ip-172-31-83-172.ec2.internal nginx[7151]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Mar 29 20:51:47 ip-172-31-83-172.ec2.internal nginx[7151]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Mar 29 20:51:47 ip-172-31-83-172.ec2.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
[root@ip-172-31-83-172 appl]#
```

Ingresamos nuevamente a nuestra consola con el usuario root esto con el fin de Navegar hacia la carpeta raíz para acceder a la carpeta de configuración como se muestra en las siguientes imágenes:

```
[ec2-user@ip-172-31-83-172 ~]$ cd..
-bash: cd..: command not found
[ec2-user@ip-172-31-83-172 ~]$ sudo su
[root@ip-172-31-83-172 ec2-user]# cd..
bash: cd..: command not found
[root@ip-172-31-83-172 ec2-user]# cd ..
[root@ip-172-31-83-172 home]# cd ..
[root@ip-172-31-83-172 /]# cd ..
[root@ip-172-31-83-172 /]#
```

La carpeta es etc como se puede visualizar en la imagen, navegamos para poder ingresar a esa:

```
[root@ip-172-31-83-172 /]# ls
bin boot dev etc home lib lib64 local media mnt opt proc root run sbin srv sys tmp usr var
[root@ip-172-31-83-172 /]# cd etc/
[root@ip-172-31-83-172 etc]# ls
```

Después de estar parados en la carpeta etc nos movemos hacia la carpeta llamada nginx utilizando el comando Cd como se muestra en la siguiente imagen:

```
[root@ip-172-31-83-172 /]# cd etc/
[root@ip-172-31-83-172 etc]# cd nginx/
[root@ip-172-31-83-172 nginx]#
```

Después de la imagen anterior ingresamos a otra carpeta llamada conf.d la cual es la última carpeta a la cual nos debemos dirigir:

```
[root@ip-172-31-83-172 nginx]# cd conf.d/
[root@ip-172-31-83-172 conf.d]# ls
```

Creamos un archivo de configuración “dicho archivo puede llevar el nombre que más les guste” en nuestro caso le llamamos proxy-containers.conf, cabe resaltar que la extensión .conf debe de ir si o si con nuestro archivo, la siguiente manera con el virtualizador(editor de archivos) vim :

```
[root@ip-172-31-83-172 conf.d]# vim proxy-containers.conf
```

La configuración que mostraremos a continuación es la de “escucha sin necesidad de poner puertos” del archivo proxy-containers.conf para los contenedores:

```
server {
listen 80;
server_name exito.com;

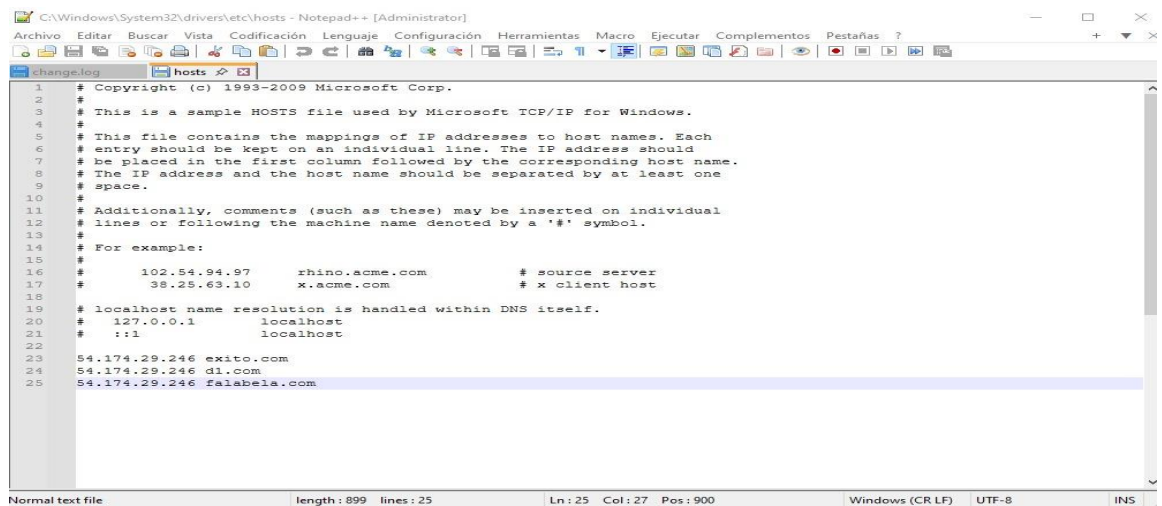
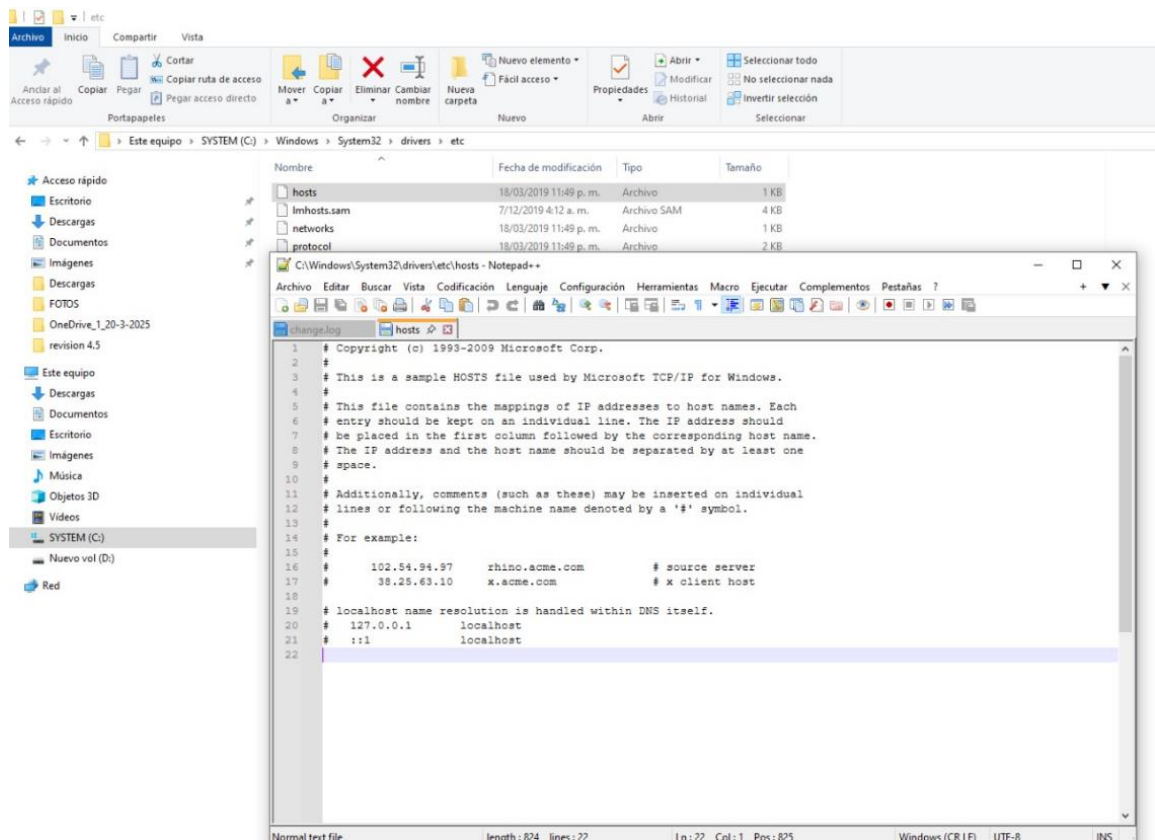
location / {
    proxy_pass http://54.174.29.246:8081;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}

server {
listen 80;
server_name falabela.com;

location / {
    proxy_pass http://54.174.29.246:8082;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}
~
~
~
~
~
~
~
~
~
~
```

- Utilizamos el server para abrir nuestro contenido.
- Listen es para escuchar por el puerto 80;
- Server_name: es para ponerle la “URL” ,“DNS” a la cual nos va a redirigir. Esto lo debemos configurar en el host del pc local para poder que se visualice el ejercicio.
- Proxy_Pass: es para redirigir a otro servidor en este caso la IP del server la redirige con el puerto 8081,8082, etc para el dominio en este caso éxito o Falabella, etc .
- Hacemos el mismo procedimiento para todos nuestros contenedores

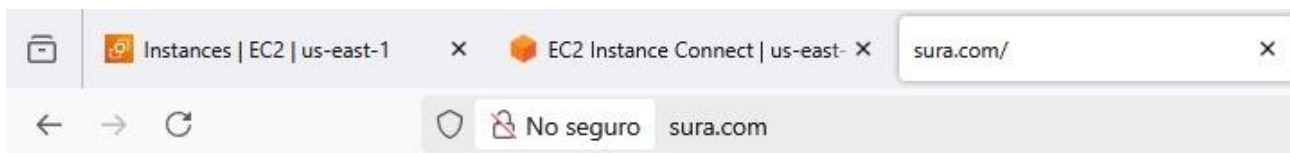
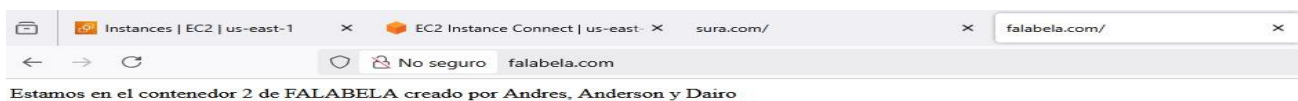
Después de tener esto configurado vamos a buscar el archivo host del pc que se encuentra en la siguiente ruta `C:\Windows\System32\drivers\etc` acá ponemos la IP de la instancia más el nombre que se anexo al archivo `proxy-containers.conf` :



Validamos que el puerto 80 en nuestra instancia se encuentre abierto para la salida a internet y sin ninguna restricción para que los ejercicios anteriormente mencionados puedan tener los resultados esperados:

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0113d20b71a1eb78b	80	TCP	0.0.0.0/0	launch-wizard-1	From internet
-	sgr-0185e8e61affe4948	8081	TCP	0.0.0.0/0	launch-wizard-1	se habilita puerto 8081
-	sgr-0dd5c7a9c1836d321	22	TCP	0.0.0.0/0	launch-wizard-1	-
-	sgr-098f7dd9da0d092ca	8080	TCP	0.0.0.0/0	launch-wizard-1	Contenedor1
-	sgr-0251d9e3b0f0a4e0d	8082	TCP	0.0.0.0/0	launch-wizard-1	se habilita puerto 8082 contene

Si se siguieron los pasos correctamente el ejercicio debería de quedar de la siguiente manera:



Conclusiones

La práctica realizada permitió comprender los conceptos fundamentales de la virtualización, la contenerización de aplicaciones y la migración hacia la nube. La utilización de herramientas como Docker y Portainer facilita el manejo de servicios en contenedores y permite replicar entornos de forma eficiente. Asimismo, se evidenció que AWS proporciona una plataforma robusta para ejecutar entornos previamente desarrollados en local, favoreciendo la escalabilidad y disponibilidad de los servicios, volviendo se una excelente opción para las empresas que desean invertir en nuevas tecnologías buscando así una interacción sencilla y desnaturalizada de la tecnología por lo cual no es necesario preocuparse de administrar servidores o inclusive las aplicaciones de manera física corriendo así el riesgo de una perdida de información ya sea por malos manejos, empleados inexpertos, saqueo, daños naturales, apagones, etc. Todo esto se traduce en administrar de una manera más escalable y sinfónica.

La virtualización de nuestras tecnologías llevo para quedarse, AWS es una buena opción para ello, solo con dar unos cuantos clicks podemos administrar de manera sencilla y eficaz nuestros productos y servicios sin perder la productividad o correr el riesgo de perderla

Referencias

(«Home». (2025, 27 febrero). Docker Documentation. <https://docs.docker.com/>

Portainer.io. (2023, 12 julio). *Kubernetes and Docker Container Management Software*.

<https://www.portainer.io/>

Ubuntu Server documentation. (s. f.). Ubuntu Server. <https://ubuntu.com/server/docs>

.