



TRABAJO DE GRADO
Opción Seminario-Diplomado.

Proceso de Implementación de servicios EC2 y Docker en AWS Web Services

Corporación Universitaria Remington.
Nombre de la facultad: Ingeniería
Nombre del programa académico: Ingeniería de sistema

Jhorman Giraldo Giraldo
Fred Jainover Rueda Ordoñez.
Tutor: Juan Pablo Berrio Lopez
Opción de Trabajo de grado Seminario-Diplomado.
2025.

Tabla de Contenidos

Resumen.....	3
Palabras clave.....	3
Marco conceptual y contextual	3
Desarrollo e implementación del aprendizaje.....	5
Diagrama de arquitectura.....	5
Descripción de la arquitectura.....	5
Configuraciones realizadas.....	5
Configuración del servidor web Windows.....	9
Creación Instancia Linux.....	14
Configuración web Linux.....	17
PING entre instancias.....	24
Implementación Docker.....	27
Prueba de estrés.....	44
Conclusiones	52
Referencias.....	53
(Puedes citar con normas APA o Vancouver. Se anexa ejemplo de normas APA).....	53

Resumen

Este trabajo de grado expone el proceso de formación y aplicación práctica llevado a cabo durante el seminario sobre Amazon Web Services conocido como AWS donde se abordaron de forma progresiva los conceptos básicos de la computación en la nube junto con la implementación de infraestructura como servicio o IaaS además del despliegue de aplicaciones usando instancias EC2 y la utilización de contenedores mediante la tecnología Docker

Durante las sesiones del seminario se fueron adquiriendo no solo conocimientos técnicos sino también una comprensión más amplia de cómo funcionan los servicios en la nube empezando por la creación de redes virtuales o VPC con la configuración de subredes públicas y privadas el manejo de direcciones IP tanto internas como externas y la implementación de gateways de Internet todo esto imitando la estructura de redes empresariales reales

Luego se trabajó con el aprovisionamiento de instancias EC2 utilizando sistemas operativos como Linux y Windows con acceso remoto configurado por medio de protocolos como SSH para entornos Linux y RDP para los servidores Windows además se realizaron instalaciones de servidores web como Apache y IIS junto con la apertura de puertos necesarios mediante los grupos de seguridad para habilitar la conectividad externa a los servicios

Una parte clave del aprendizaje fue el uso de Docker como herramienta para empaquetar aplicaciones y ejecutarlas en contenedores ligeros lo que facilitó el manejo y la portabilidad de los servicios allí se practicó la instalación de Docker en las instancias además de la creación de contenedores con imágenes personalizadas construidas con Dockerfile y se aprendió también a utilizar volúmenes de Docker que permiten almacenar datos de forma persistente incluso si el contenedor se detiene o elimina todo esto fue complementado con la automatización de entornos mediante Docker Compose para coordinar múltiples servicios en conjunto

Todo este recorrido fue desarrollado mediante ejercicios prácticos guiados por el docente con apoyo paso a paso lo que permitió a cada estudiante desplegar y poner en funcionamiento servicios completos en la nube desde cero además de reforzar habilidades como la resolución de errores la documentación y el seguimiento de buenas prácticas en entornos cloud como la gestión de accesos y el uso eficiente de los recursos disponibles Este documento busca no solo mostrar lo que se logró al final del proceso sino también detallar de manera clara cada componente usado y cada decisión técnica tomada con el objetivo de servir como guía útil para quienes deseen empezar en el mundo de la infraestructura en la nube y las tecnologías modernas de despliegue de servicios digitales

Palabras clave

Computación en la nube, AWS, EC2, Docker, Contenedores, Servidores Virtuales, Infraestructura como Servicio, VPC, Volúmenes, Automatización, Red Virtual.

Marco conceptual y contextual

Computación en la nube e Infraestructura como Servicio (IaaS)

La computación en la nube es un algo que permite acceder a recursos tecnológicos como servidores, almacenamiento o redes a través de Internet, sin la necesidad de tener los equipos físicamente. Esto reduce los costos y mejora la disponibilidad de los servicios además permite escalar recursos de manera mucho más flexible.

El seminario se centró básicamente en la Infraestructura como Servicio (IaaS) que ofrece componentes esenciales como máquinas virtuales (servidores), redes y almacenamiento, permitiendo que el usuario tenga control total sobre su configuración.

Instancia de EC2 y redes virtuales (VPC)

Amazon EC2 (Elastic Compute Cloud) es uno de los servicios de AWS que permite crear unos servidores virtuales y unas instancias. Estas se pueden configurar con diferentes sistemas operativos sea (Windows o Linux) con capacidades de memoria, almacenamiento y red.

Durante el seminario se utilizaron las instancias de EC2 para simular servidores reales, y se accedió a ellas mediante RDP (en Windows) y SSH (en Linux).

Para alojarlas de forma segura, se creó y diseño una red virtual privada (VPC), donde se configuraron las subredes públicas y los gateways de Internet además de reglas de tráfico, simulando el funcionamiento de una red empresarial moderna.

Contenedores, Docker y volúmenes

Un contenedor es básicamente un paquete ligero el cual incluye una aplicación junto con unas dependencias haciendo que se ejecute de forma continua y consistente en cualquier entorno. Docker es la tecnología que ayuda a la creación y ejecución en gestión de estos contenedores.

A lo largo de este seminario que se vio, se instalaron contenedores en instancias EC2 con Linux, y se utilizaron algunos comandos como `docker run`, `docker exec`, `docker ps`, entre otros. También aprendimos a crear imágenes propias usando Dockerfile.

Para conservar los datos dentro de los contenedores se utilizaron volúmenes de Docker que permitían guardar archivos que sobreviven incluso si el contenedor se elimina o se reinicia. Esto es clave para servicios como bases de datos.

Contexto académico del seminario y aplicación práctica

El proyecto fue desarrollado en el marco del seminario "Implementación de Servicios en la Nube con AWS", como opción de grado en la Facultad de Ingeniería. El enfoque fue práctico y guiado por el docente, con sesiones virtuales y ejercicios paso a paso.

Los estudiantes configuraron servicios en AWS desde cero, aplicando conceptos de infraestructura, conectividad, seguridad, contenedores y automatización. Este entorno académico simuló condiciones reales de empresas tecnológicas actuales, permitiendo el desarrollo de habilidades relevantes para el mundo laboral.

Desarrollo e implementación del aprendizaje.

Diagrama de arquitectura.

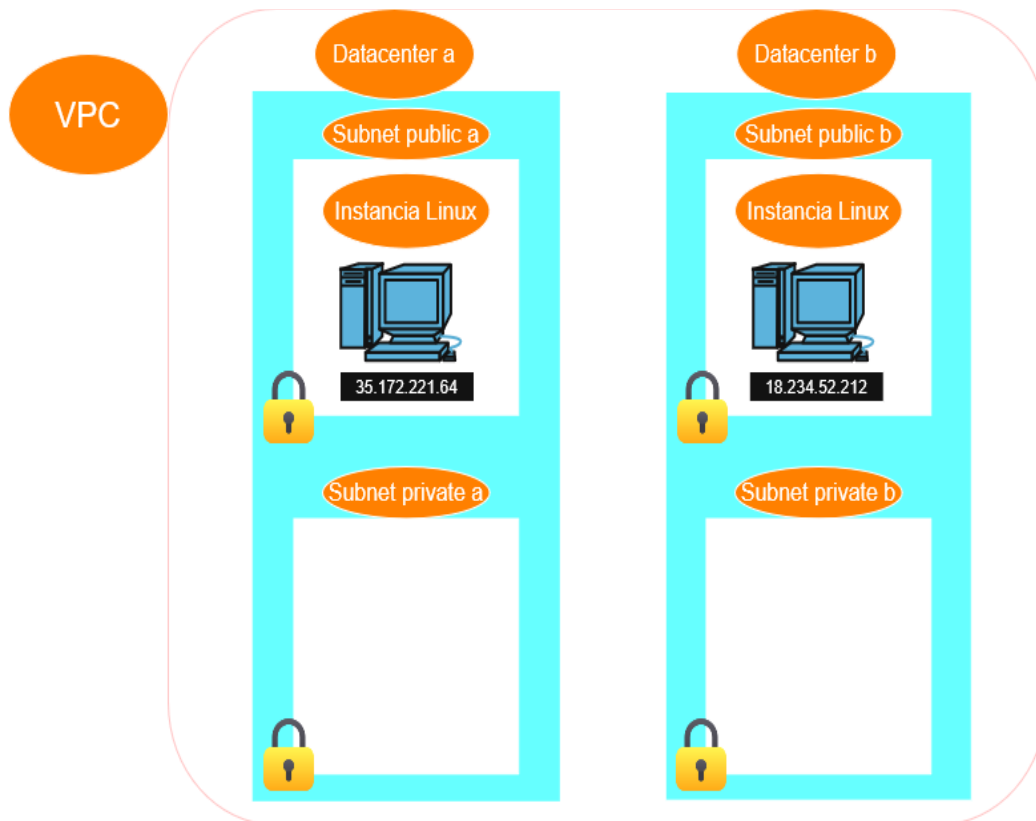


Figura 1

Diagrama de arquitectura

Descripción de la arquitectura.

Como primer paso se ha realiza la creacion de una VPC personalizada en AWS, la cual consta de una subred pública en la cual alojé dos instancias EC2: Una de ellas tiene una instancia de Microsoft Windows Server 2016 Base y otra con Linux. Las dos instancias tienen asignadas direcciones IP públicas

Configuraciones realizadas.

1. Realizaremos la asignación de un nombre con el cual identificaremos a la instancia Windows (EN-ServidorWindows).

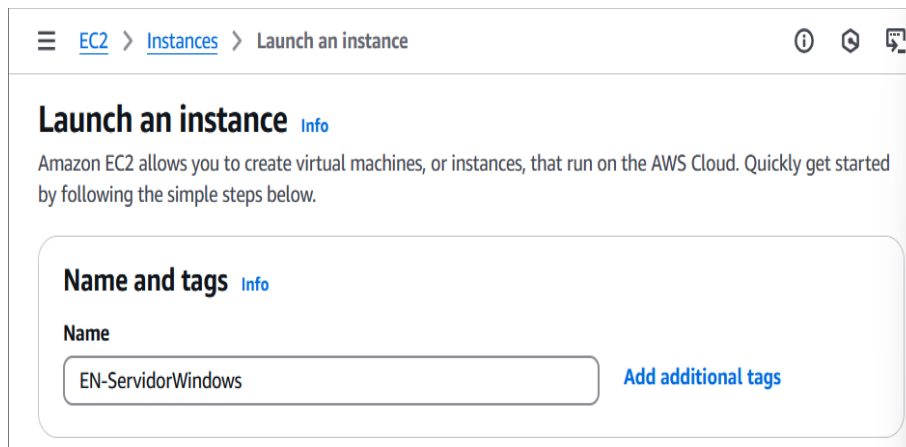


Figura 2 Asignación nombre de la instancia.

- luego de ello seleccionamos la AMI correspondiente (Microsoft Windows Server 2016 Base), con el fin de seleccionar por consiguiente mi tipo de instancia (t2.micro) la cual tiene 1vCPU y 1 GiB Memory.

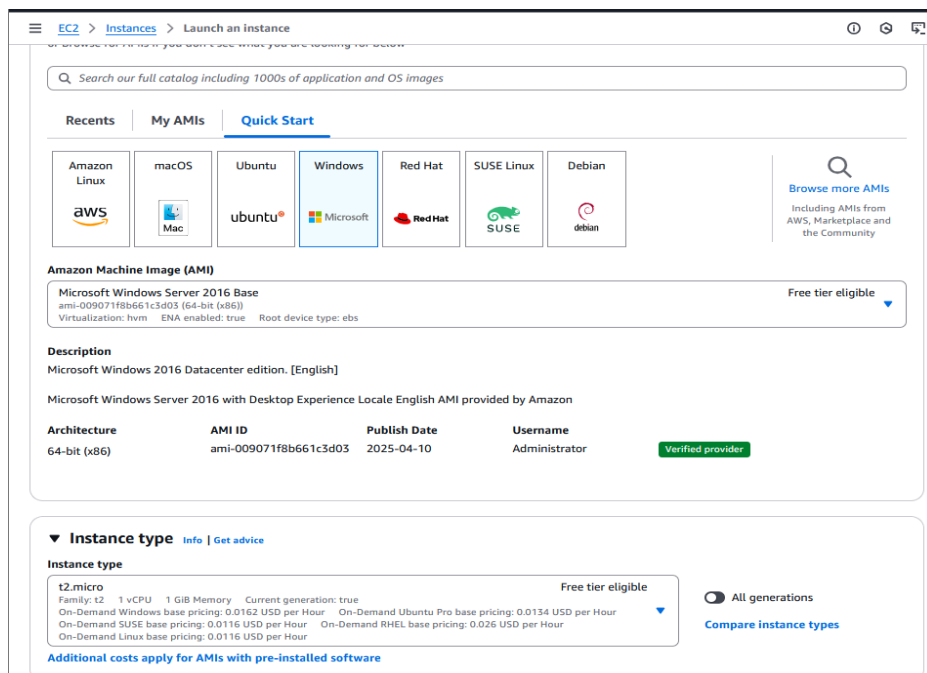
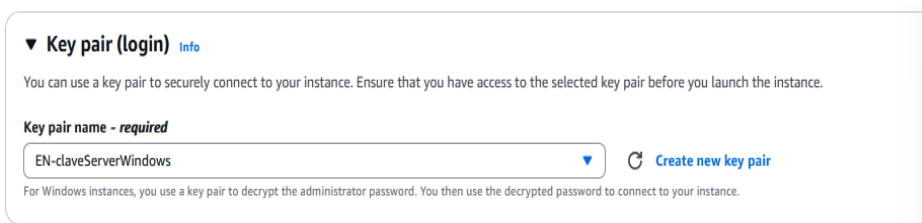


Figura 3 Asignar AMI.

3. Luego de haber configurado correctamente estos parámetros realizaremos la creación de una nueva llave con la cual accederemos a la instancia.



▼ **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.

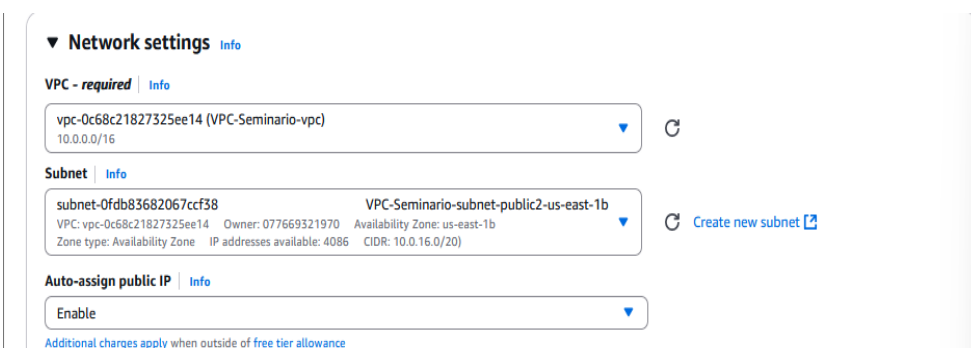
Key pair name - required

EN-claveServerWindows [Create new key pair](#)

For Windows instances, you use a key pair to decrypt the administrator password. You then use the decrypted password to connect to your instance.

Figura 4 Asignación llave.

4. Luego de seguir con mucha precisión los pasos anteriores, pasamos al apartado consiguiente, en donde realizaremos la configuración de red de nuestra instancia, allí seleccionaremos el VPC, luego seleccionaremos los DATACENTER, en este caso como deseo que a la instancia se pueda ingresar desde internet, seleccionaremos una subnet publica ya sea la que está en el data center a o b, luego de esto garantizamos que la maquina tenga una IP pública.



▼ **Network settings** [Info](#)

VPC - required [Info](#)

vpc-0c68c21827325ee14 (VPC-Seminario-vpc) [Create new VPC](#)

10.0.0.0/16

Subnet [Info](#)

subnet-0fdb83682067ccf38 VPC-Seminario-subnet-public2-us-east-1b [Create new subnet](#)

VPC: vpc-0c68c21827325ee14 Owner: 077669321970 Availability Zone: us-east-1b
Zone type: Availability Zone IP addresses available: 4086 CIDR: 10.0.16.0/20

Auto-assign public IP [Info](#)

Enable

Additional charges apply when outside of free tier allowance

Figura 5 Configuración de red.

5. Como paso consiguiente, realizaremos la creación de nuestro Firewall (Security Group) donde asignaremos el nombre correspondiente y como vamos a ingresar (RDP), el protocolo es (TCP) y será mediante el puerto 3389 y permitiremos que se pueda ingresar desde cualquier dirección.

The screenshot shows the AWS Management Console interface for configuring a Security Group rule. The breadcrumb navigation is "EC2 > Instances > Launch an instance". A warning states: "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.

Options: Create security group, Select existing security group

Security group name - required
EN-SG-ServerWindows

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 2025-05-06T15:09:12.605Z

Inbound Security Group Rules
▼ Security group rule 1 (TCP, 3389, 0.0.0.0/0) Remove

Type Info	Protocol Info	Port range Info
rdp	TCP	3389

Source type Info	Source Info	Description - optional Info
Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

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

Add security group rule

Figura 6 Configuración firewall

6. Por último, verificamos la configuración de almacenamiento, en este caso nos iremos con la opción predeterminada (capa gratuita).

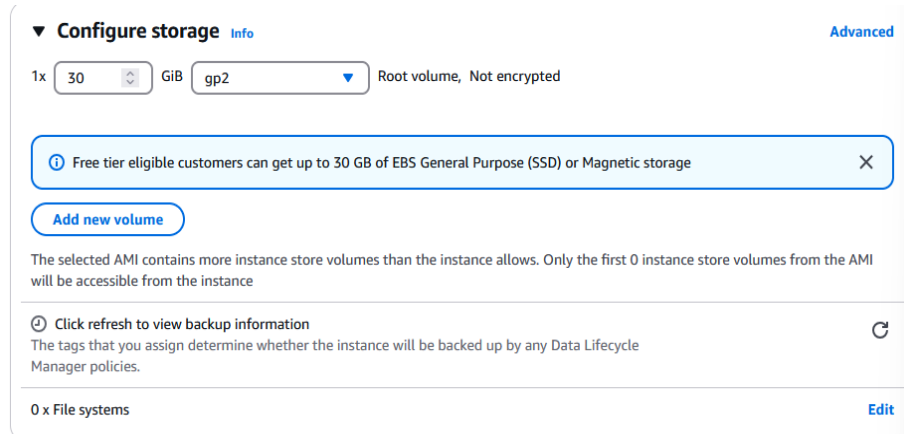


Figura 7 Configuración de almacenamiento.

Configuración del servidor web Windows.

1. Para conectarme debo de verificar que mi instancia tenga habilitada la IP pública.

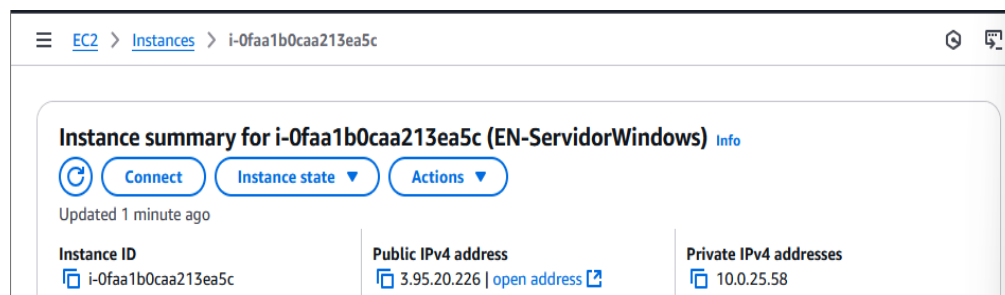


Figura 8 IP habilitada.

2. Luego de ello selecciono la opción Connect, donde allí encontraremos la opción de acceder a la instancia mediante RDP client.

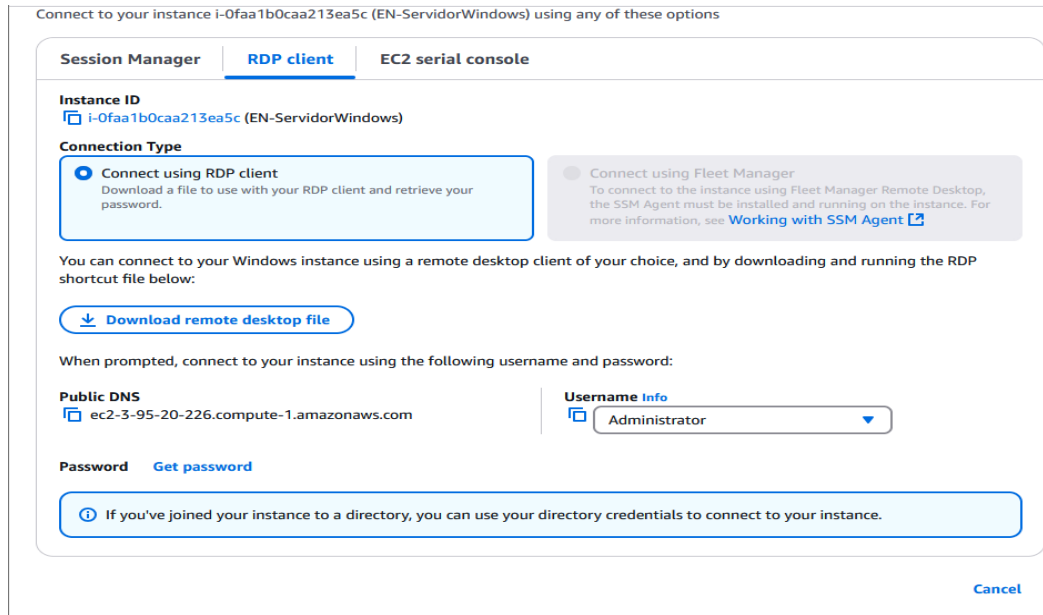


Figura 9 RDP client

- Luego de ellos obtendremos la contraseña creada anteriormente.

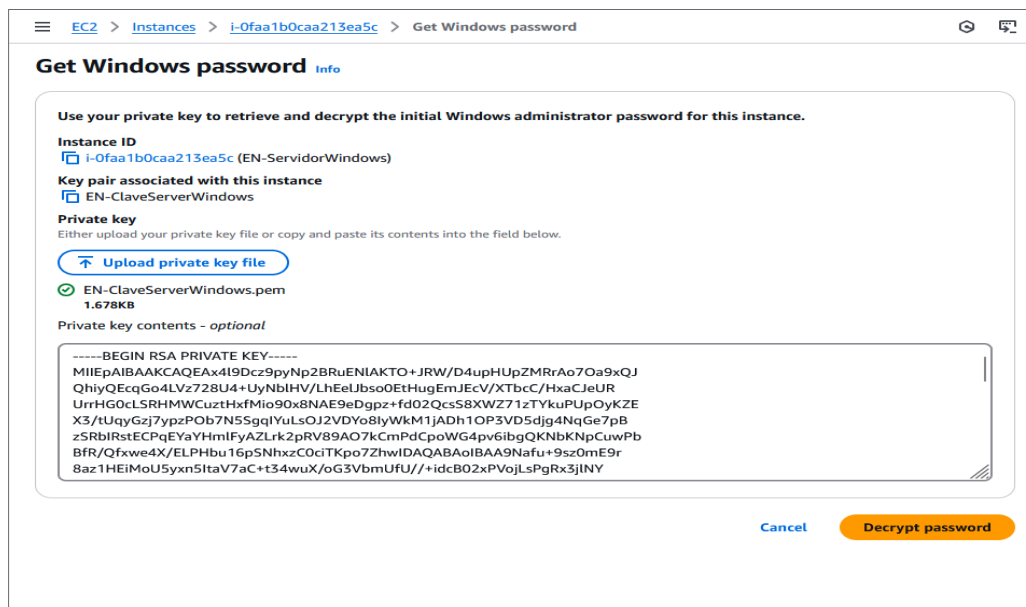


Figura 10 Obtener contraseña.

- Desencriptamos la contraseña obtenida y copiamos la contraseña para con ella conectarnos mediante conexión a escritorio remoto, luego ingresamos los valores de usuario y contraseña para luego ya estar en la máquina.

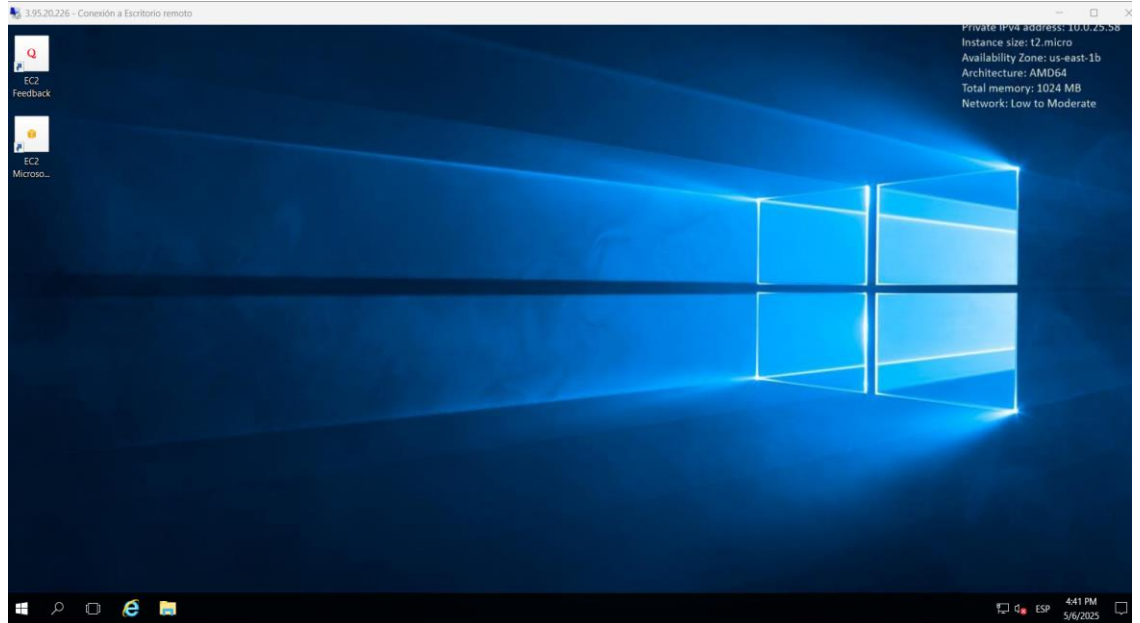


Figura 11 Acceso máquina virtual Windows.

- Para lograr que nos podamos conectar desde internet, debemos de ingresar una regla (Inbound rules), allí agregaremos una nueva regla de tipo HTTP, la cual será mediante el protocolo TCP y tendrá como predeterminado el puerto 80 y se podrá ingresar desde cualquier lado.

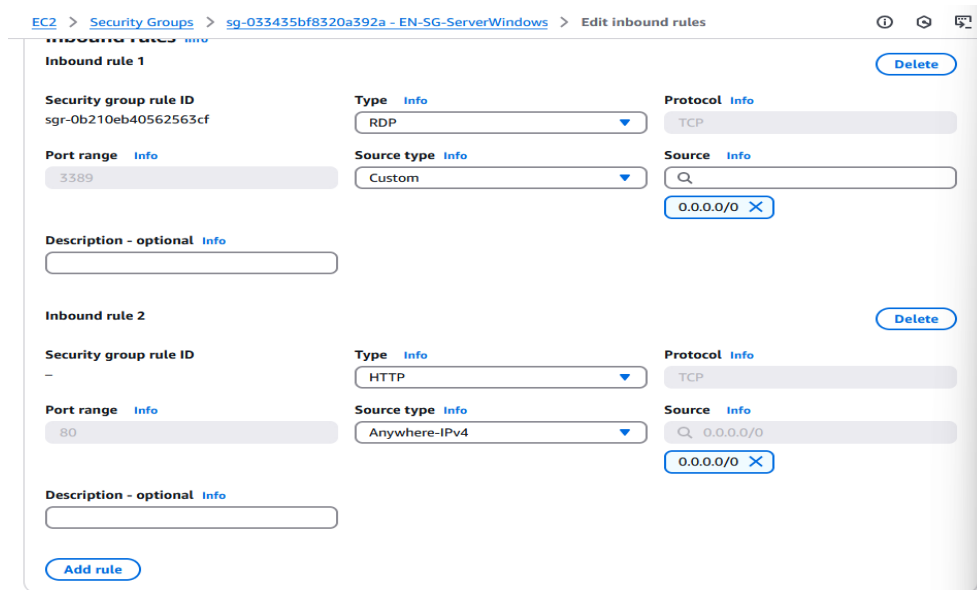


Figura 12 Configuración puerto 80.

6. Crearemos un servidor web desde la maquina anteriormente creada, para eso ingresaremos a Server Manager y ingresamos a agregar roles con el fin de configurar un servidore web y poder acceder.

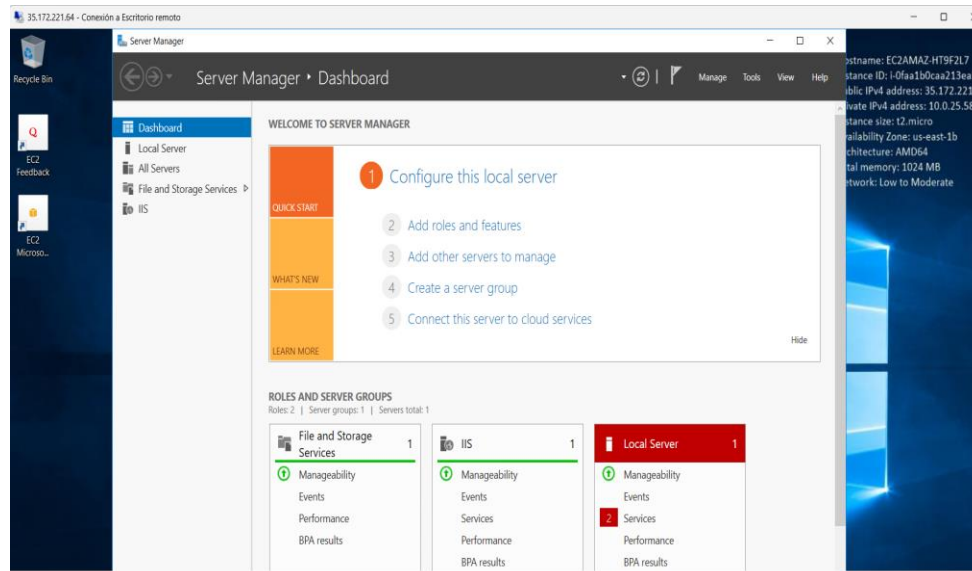


Figura 13 Creación servidor web.

7. Luego de ello accedemos a la opción Add roles and features, donde instalaremos el rol web server IIS y al evidenciar que ya está instalado podemos acceder al servidor desde la web.

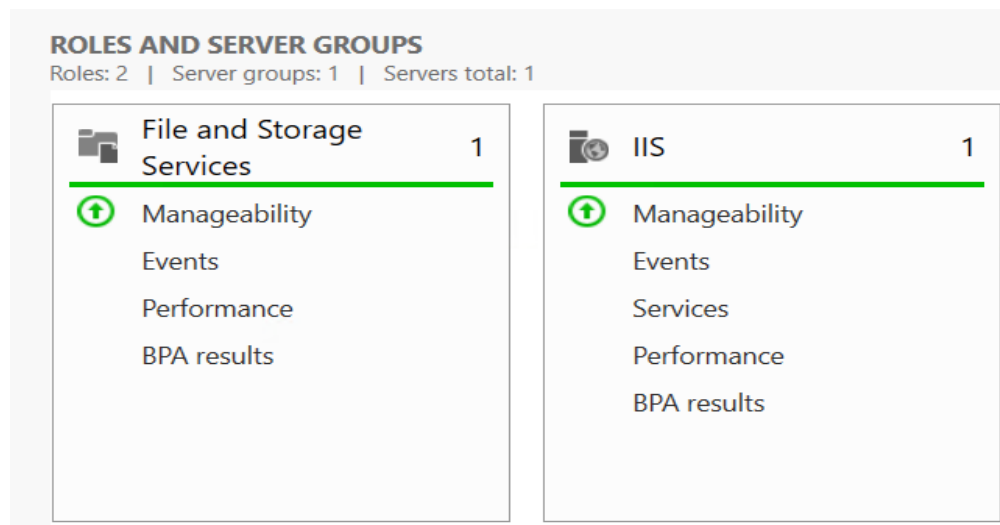


Figura 14 Instalador de rol web.

8. Evidencia de acceso al servidor web, desde cualquier dirección.

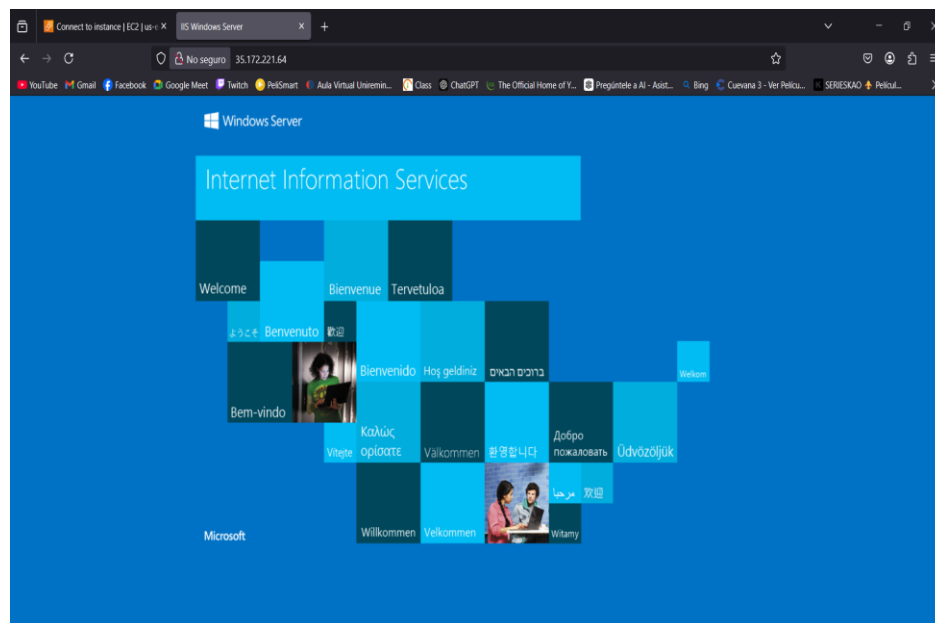


Figura 15 Acceso servidore web.

Creación Instancia Linux.

1. Realizaremos la asignación de un nombre con el cual identificaremos a la instancia Linux (EN-ServidorLinux).

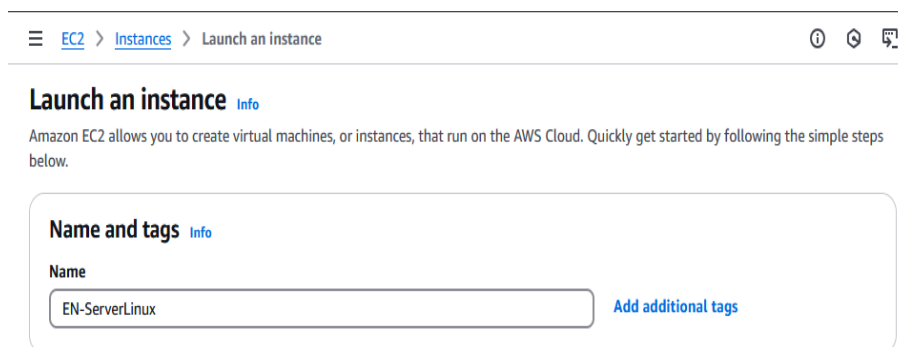


Figura 16 Nombre instancia Linux.

- luego de ello seleccionamos la AMI correspondiente (Amazon Linux 2023 AMI), con el fin de seleccionar por consiguiente mi tipo de instancia (t2.micro) la cual tiene 1vCPU y 1 GiB Memory.

or Browse for AMIs if you don't see what you are looking for below

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

Recents | My AMIs | **Quick Start**

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

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI Free tier eligible

ami-0f88e80871fd81e91 (64-bit (x86), uefi-preferred) / ami-0bc72bd3b8ba0b59d (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

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 2023.7.20250428.1 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username
64-bit (x86)	uefi-preferred	ami-0f88e80871fd81e91	2025-04-30	ec2-user

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](#)

Figura 17 Selección AMI.

- Luego de haber configurado correctamente estos parámetros realizaremos la creación de una nueva llave con la cual accederemos a la instancia.

▼ **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

EN-ClaveServerLinux [Create new key pair](#)

Figura 18 Creación llave de seguridad.

4. Luego de seguir con mucha precisión los pasos anteriores, pasamos al apartado consiguiente, en donde realizaremos la configuración de red de nuestra instancia, allí seleccionaremos el VPC, luego seleccionaremos los DATACENTER, en este caso como deseo que a la instancia se pueda ingresar desde internet, seleccionaremos una subnet publica, ya sea la que está en el data center a o b, luego de esto garantizamos que la maquina tenga una IP pública.

▼ **Network settings** [Info](#)

VPC - required [Info](#)

vpc-0c68c21827325ee14 (VPC-Seminario-vpc)
10.0.0.0/16 [Create new vpc](#)

Subnet [Info](#)

subnet-0fdb83682067ccf38 VPC-Seminario-subnet-public2-us-east-1b
VPC: vpc-0c68c21827325ee14 Owner: 077669321970 Availability Zone: us-east-1b
Zone type: Availability Zone IP addresses available: 4085 CIDR: 10.0.16.0/20 [Create new subnet](#)

Auto-assign public IP [Info](#)

Enable

Additional charges apply when outside of free tier allowance

Figura 19 Configuración de red.

5. Como paso consiguiente, realizaremos la creación de nuestro Firewall (Security Group) donde asignaremos el nombre correspondiente y como

vamos a ingresar (SSH), el protocolo es (TCP) y será mediante el puerto 22.

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*
EN-SG-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 2025-05-07T13:58:52.463Z

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

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

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

Add security group rule

► **Advanced network configuration**

Figura 20 Configuración firewall.

6. Por último, verificamos la configuración de almacenamiento, en este caso será elegida la opción predeterminada (capa gratuita).

Configuración web Linux.

1. Como primer paso seleccionaremos la instancia Linux creada con anterioridad y accederemos al apartado CONNECT, allí seleccionaremos la manera en que nos vamos a conectar (SSH client).

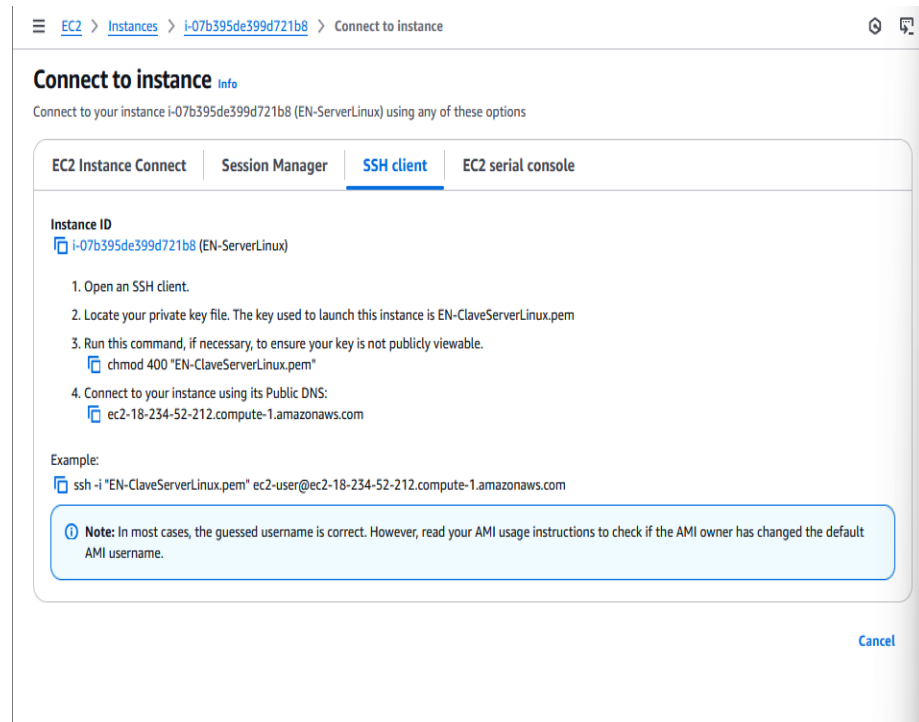


Figura 21 Configuración web Linux

2. Luego de ello necesitaremos alguna aplicación la cual nos permita conectarnos mediante SSH a la instancia, en este MobaXterm.

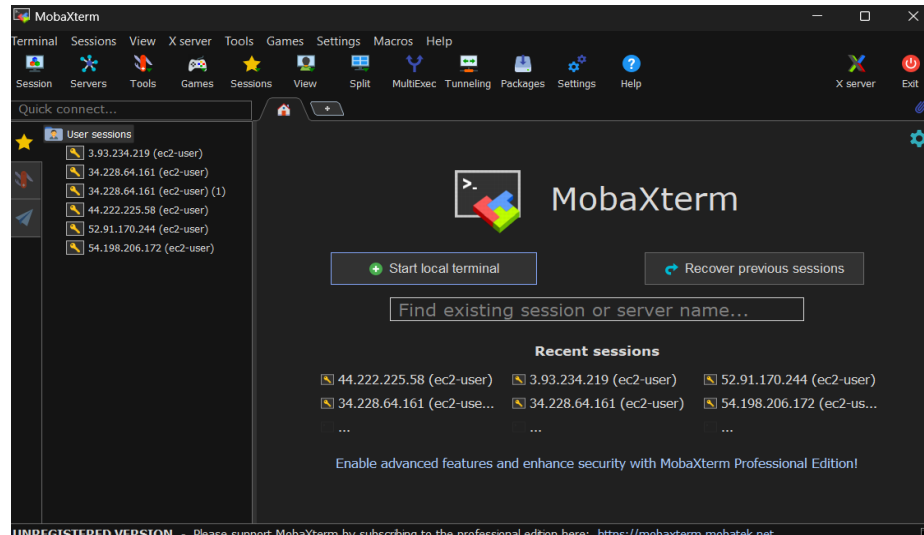


Figura 22 Conexión mediante SSH.

3. Luego de ello con la dirección pública de la maquina ingresaremos a hacer la conexión mediante la aplicación MobaXterm, también necesitaremos la clave de seguridad de la máquina, el usuario (ec2-user) y el puerto a través del que no estamos mediante SSH (22).

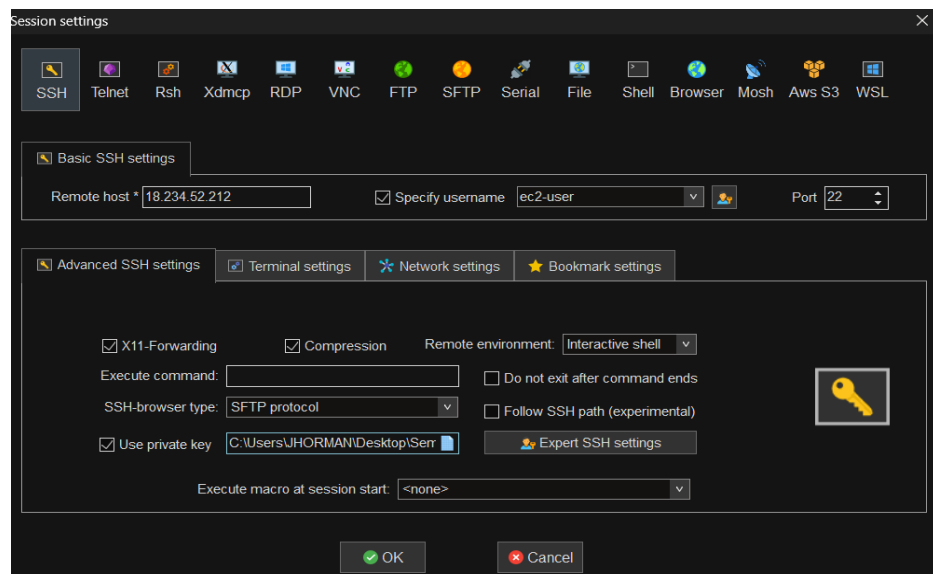


Figura 23 Conexión con IP y clave de seguridad

- Montaremos luego de que este correcta la conexión el servidor web, ingresando los comandos necesarios. Instalamos primero que todo la aplicación con el comando **dnf install httpd**.

```
[root@ip-10-0-17-180 ec2-user]# dnf install httpd
Last metadata expiration check: 0:00:55 ago on Wed May 7 14:30:04 2025.
Dependencies resolved.
=====
Package                Arch      Version                Repository             Size
=====
Installing:
httpd                  x86_64    2.4.62-1.amzn2023     amazonlinux            48 k
Installing dependencies:
apr                    x86_64    1.7.5-1.amzn2023.0.4  amazonlinux            129 k
apr-util               x86_64    1.6.3-1.amzn2023.0.1  amazonlinux            98 k
generic-logos-httpd   noarch    18.0.0-12.amzn2023.0.3 amazonlinux            19 k
httpd-core             x86_64    2.4.62-1.amzn2023     amazonlinux            1.4 M
httpd-filesystem      noarch    2.4.62-1.amzn2023     amazonlinux            14 k
httpd-tools           x86_64    2.4.62-1.amzn2023     amazonlinux            81 k
libbrotli              x86_64    1.0.9-4.amzn2023.0.2  amazonlinux            315 k
mailcap               noarch    2.1.49-3.amzn2023.0.3 amazonlinux            33 k
Installing weak dependencies:
apr-util-openssl      x86_64    1.6.3-1.amzn2023.0.1  amazonlinux            17 k
mod_http2             x86_64    2.0.27-1.amzn2023.0.3 amazonlinux            166 k
mod_lua               x86_64    2.4.62-1.amzn2023     amazonlinux            61 k
Transaction Summary
=====
Install 12 Packages
```

Figura 24 Instalar HTTPD.

- Luego de que la aplicación se instaló, eso no significa que ya está corriendo, es por este modo que debemos de verificar si está activo o inactivo el servicio. En este caso Inactivo como podemos ver en la imagen consiguiente.

```

Complete!
[root@ip-10-0-17-180 ec2-user]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset
   Active: inactive (dead)
   Docs: man:httpd.service(8)
lines 1-4/4 (END)

```

Figura 25 Verificación estado aplicación.

6. Luego de verificar el estado pasamos a activarlo de la siguiente manera. Iniciamos el servicio con **systemctl start httpd** y luego verificamos el estado con **systemctl status httpd**.

```

Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: d
Active: inactive (dead)
   Docs: man:httpd.service(8)
[root@ip-10-0-17-180 ec2-user]# systemctl start httpd
[root@ip-10-0-17-180 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: d
   Active: active (running) since Wed 2025-05-07 14:48:58 UTC; 6s ago
   Docs: man:httpd.service(8)
  Main PID: 27233 (httpd)
   Status: "Started, listening on: port 80"
   Tasks: 177 (limit: 1111)
  Memory: 13.0M
    CPU: 56ms
  CGroup: /system.slice/httpd.service
          └─27233 /usr/sbin/httpd -DFOREGROUND
             └─27234 /usr/sbin/httpd -DFOREGROUND
                └─27235 /usr/sbin/httpd -DFOREGROUND
                   └─27236 /usr/sbin/httpd -DFOREGROUND
                      └─27237 /usr/sbin/httpd -DFOREGROUND

May 07 14:48:58 ip-10-0-17-180.ec2.internal systemd[1]: Starting httpd.service
May 07 14:48:58 ip-10-0-17-180.ec2.internal systemd[1]: Started httpd.service
May 07 14:48:58 ip-10-0-17-180.ec2.internal httpd[27233]: Server configured, li
lines 1-19/19 (END)

```

Figura 26 Inicialización de la aplicación.

7. Se necesita para lograr ver el servidor web que el tráfico pase a través de AWS, y para eso verificamos los puertos que están permitidos y habilitar el puerto 80.

▼ Inbound rules

Q Filter rules < 1 >

Security group rule ID	Port range	Protocol	Source
sgr-0090c2f23f821ec73	22	TCP	0.0.0.0/0

► Outbound rules

Figura 27 Configuración puerto 80.

8. Luego de ello entramos a security group y agregamos la nueva regla HTTP a través del protocolo TCP y el puerto 80, para que luego configuremos que cualquiera pueda ingresar.

Inbound rule 2 Delete

Security group rule ID: -

Type: HTTP

Protocol: TCP

Port range: 80

Source type: Anywhere-IPv4

Source: 0.0.0.0/0

Description - optional

Add rule

Figura 28 Configuración nueva regla http.

9. Evidencia de que permite ingresar desde la web.

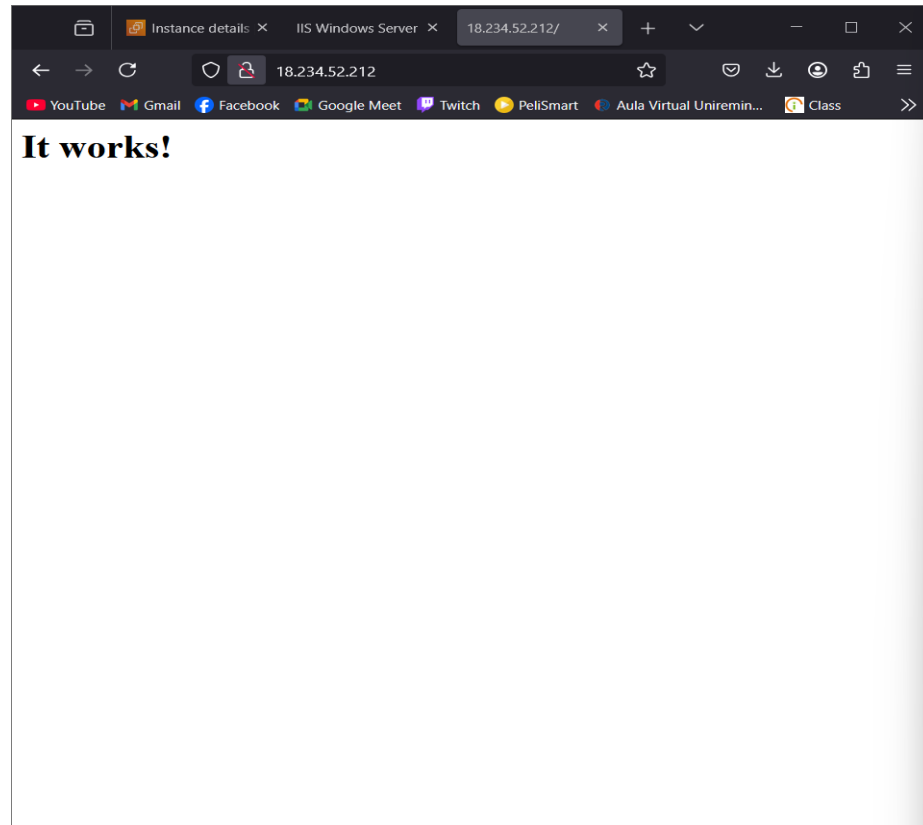


Figura 29 Aplicación web corriendo.

10. Por último, hay que realizar un comando para cuando yo apague y vuelva a prender la máquina, el servicio arranque automáticamente.

```
[root@ip-10-0-17-180 ec2-user]# systemctl enable httpd
[root@ip-10-0-17-180 ec2-user]#
```

Figura 30 Arranque automático.

11. Instalación de una plantilla, descargamos una plantilla HTML, entramos a la dirección donde podemos instalar la plantilla (cd

/var/www/html), luego hacemos lo siguiente (wget Link de descarga de la plantilla). Algo a tener en cuenta es que las plantillas vienen comprimidas, hay que descomprimirlas así (unzip nombre de la descarga).

```
inflating: assets/sass/libs/_vendor.scss
inflating: assets/sass/libs/_breakpoints.scss
inflating: assets/sass/noscript.scss
inflating: README.txt
inflating: index.html
inflating: index-demo.html
creating: images/
inflating: images/spotlight03.jpg
inflating: images/spotlight01.jpg
creating: images/gallery/
creating: images/gallery/thumbs/
inflating: images/gallery/thumbs/08.jpg
inflating: images/gallery/thumbs/09.jpg
inflating: images/gallery/thumbs/04.jpg
inflating: images/gallery/thumbs/05.jpg
inflating: images/gallery/thumbs/07.jpg
inflating: images/gallery/thumbs/03.jpg
inflating: images/gallery/thumbs/11.jpg
inflating: images/gallery/thumbs/02.jpg
inflating: images/gallery/thumbs/01.jpg
inflating: images/gallery/thumbs/10.jpg
inflating: images/gallery/thumbs/06.jpg
inflating: images/gallery/thumbs/12.jpg
creating: images/gallery/fulls/
inflating: images/gallery/fulls/08.jpg
inflating: images/gallery/fulls/09.jpg
inflating: images/gallery/fulls/04.jpg
inflating: images/gallery/fulls/05.jpg
inflating: images/gallery/fulls/07.jpg
inflating: images/gallery/fulls/03.jpg
inflating: images/gallery/fulls/11.jpg
inflating: images/gallery/fulls/02.jpg
inflating: images/gallery/fulls/01.jpg
inflating: images/gallery/fulls/10.jpg
inflating: images/gallery/fulls/06.jpg
inflating: images/gallery/fulls/12.jpg
inflating: images/spotlight02.jpg
inflating: images/pic03.jpg
inflating: images/pic01.jpg
inflating: images/pic02.jpg
inflating: images/banner.jpg
[root@ip-10-0-17-180 html]#
```

Figura 31 Instalación plantilla HTML.

PING entre instancias.

1. En caso del ping de mi instancia Windows hacia mi instancia Linux, tuve que configurar una nueva regla en el security group de mi instancia Linux, con el fin de que recibiera protocolos ICMP, protocolo el cual usa el comando ping.

Inbound rule 3 Delete

Security group rule ID: -

Type: Custom ICMP - IPv4

Protocol: All

Port range: All

Source type: Anywhere-IPv4

Source: 0.0.0.0/0

Description - optional:

Figura 32 Nueva regla ICMP Windows.

Ping exitoso desde mi instancia Windows a Linux.

```
C:\Users\Administrator>ping 10.0.17.180

Pinging 10.0.17.180 with 32 bytes of data:
Reply from 10.0.17.180: bytes=32 time=1ms TTL=127
Reply from 10.0.17.180: bytes=32 time=1ms TTL=127
Reply from 10.0.17.180: bytes=32 time=1ms TTL=127
Reply from 10.0.17.180: bytes=32 time=1ms TTL=127

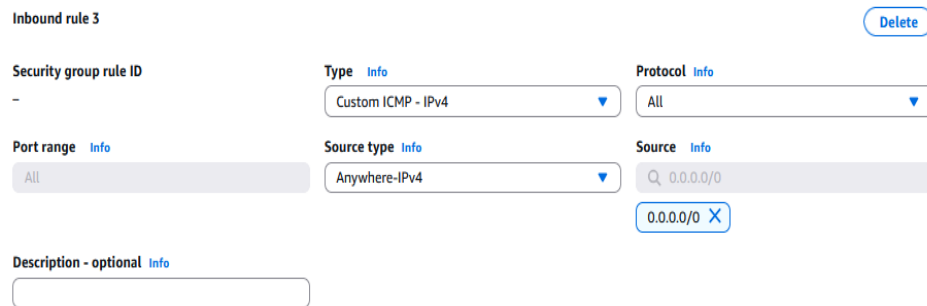
Ping statistics for 10.0.17.180:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\Users\Administrator>
```

Figura 33 Ping Windows a la instancia Linux.

2. En caso del ping de mi instancia Linux hacia mi instancia Windows, tuve que configurar una nueva regla en el security group de mi instancia Windows, con el fin de que recibiera protocolos ICMP, protocolo el cual usa el comando ping. Además desde Windows power Shell use este comando (Enable-NetFirewallRule -DisplayGroup "File

and Printer Sharing") con el fin de habilitar que el firewall de Windows recibiera protocolos ICMP.



The screenshot shows the configuration for an inbound firewall rule named "Inbound rule 3". The "Security group rule ID" is set to "-". The "Type" is "Custom ICMP - IPv4". The "Protocol" is "All". The "Port range" is "All". The "Source type" is "Anywhere-IPv4". The "Source" is "0.0.0.0/0". There is a "Delete" button in the top right corner. A "Description - optional" field is present but empty.

Figura 34 Configuración ICMP.

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> Enable-NetFirewallRule -DisplayGroup "File and Printer Sharing"
PS C:\Users\Administrator> _
```

Figura 35 Configuración ICMP.

Ping exitoso desde mi instancia Linux a Windows.

```
[root@ip-10-0-17-180 ec2-user]# ping 10.0.25.58
PING 10.0.25.58 (10.0.25.58) 56(84) bytes of data.
64 bytes from 10.0.25.58: icmp_seq=1 ttl=128 time=0.718 ms
64 bytes from 10.0.25.58: icmp_seq=2 ttl=128 time=1.40 ms
64 bytes from 10.0.25.58: icmp_seq=3 ttl=128 time=0.458 ms
64 bytes from 10.0.25.58: icmp_seq=4 ttl=128 time=0.449 ms
64 bytes from 10.0.25.58: icmp_seq=5 ttl=128 time=0.657 ms
64 bytes from 10.0.25.58: icmp_seq=6 ttl=128 time=0.605 ms
64 bytes from 10.0.25.58: icmp_seq=7 ttl=128 time=0.717 ms
64 bytes from 10.0.25.58: icmp_seq=8 ttl=128 time=1.40 ms
64 bytes from 10.0.25.58: icmp_seq=9 ttl=128 time=0.422 ms
64 bytes from 10.0.25.58: icmp_seq=10 ttl=128 time=0.592 ms
64 bytes from 10.0.25.58: icmp_seq=11 ttl=128 time=0.443 ms
^C
--- 10.0.25.58 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10381ms
rtt min/avg/max/mdev = 0.422/0.715/1.403/0.339 ms
[root@ip-10-0-17-180 ec2-user]#
```

Figura 36 Ping desde Linux a Windows.

Implementación Docker.

1. Realizaremos la descarga de Docker. Docker es un paquete que se descarga, el cual tiene todas las dependencias que se necesitan. Para realizar la descarga deberemos de haber realizado el apartado de configuración web de un servidor web Linux; donde desde la conexión realizada con MobaXterm, digitaremos el siguiente comando. **dnf install docker.**

```

Verifying      : docker-25.0.8-1.amzn2023.0.3.x86_64      3/11
Verifying      : iptables-libs-1.8.8-3.amzn2023.0.2.x86_64  4/11
Verifying      : iptables-nft-1.8.8-3.amzn2023.0.2.x86_64  5/11
Verifying      : libcgrou-3.0-1.amzn2023.0.1.x86_64      6/11
Verifying      : libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64  7/11
Verifying      : libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64  8/11
Verifying      : libnftnl-1.2.2-2.amzn2023.0.2.x86_64    9/11
Verifying      : pigz-2.5-1.amzn2023.0.3.x86_64         10/11
Verifying      : runc-1.2.4-1.amzn2023.0.1.x86_64       11/11

Installed:
container-selinux-3:2.233.0-1.amzn2023.noarch
containerd-1.7.27-1.amzn2023.0.2.x86_64
docker-25.0.8-1.amzn2023.0.3.x86_64
iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgrou-3.0-1.amzn2023.0.1.x86_64
libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64
runc-1.2.4-1.amzn2023.0.1.x86_64

Complete!
[root@ip-10-0-31-228 html]# █

```

Figura 37 Instalación Docker.

2. Verificamos el estado del servicio Docker con **systemctl status docker**. Esto con el fin de saber si el servicio estado o no activo.

```

Complete!
[root@ip-10-0-31-228 html]# systemctl status docker
○ docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; pr
   Active: inactive (dead)
   TriggeredBy: ○ docker.socket
   Docs: https://docs.docker.com

```

Figura 38 Verificación estado docker.

3. Luego de ellos dependiendo del estado en el que se encuentre docker realizaremos la inicialización o no del servicio. En este caso lo haremos con el siguiente comando **systemctl start docker**. (Repetir luego de esto el paso anterior).

```

Docs: https://docs.docker.com
lines 1-5/5 (END)

[root@ip-10-0-31-228 html]# systemctl start docker
[root@ip-10-0-31-228 html]# █

```

Figura 39 Inicialización docker.

4. Realizaremos la descarga de la imagen necesaria, en este caso la imagen HTTPD.

```

[root@ip-10-0-31-228 html]# systemctl start docker
[root@ip-10-0-31-228 html]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
254e724d7786: Pull complete
10d01782dc02: Pull complete
4f4fb700ef54: Pull complete
4ceeea7b3d76: Pull complete
0ff470512d2f: Pull complete
ba78a05e3b3c: Pull complete
Digest: sha256:c11efd67f6308f2c25965e4e9d13ded15e7c45c0367b95f619a16e03c6c1e2b1
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@ip-10-0-31-228 html]# █

```

Figura 40 Imagen HTTPD.

5. Para verificar la descarga de la imagen realizamos el siguiente comando. **docker images**.

```

[root@ip-10-0-31-228 html]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
httpd latest 0208f149a449 3 months ago 148MB
[root@ip-10-0-31-228 html]# █

```

Figura 41 Verificar imagen.

6. Para descargar otra imagen, en este caso la imagen nginx con este comando:

docker pull nginx.

```
[root@ip-10-0-31-228 html]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
254e724d7786: Already exists
913115292750: Pull complete
3e544d53ce49: Pull complete
4f21ed9ac0c0: Pull complete
d38f2ef2d6f2: Pull complete
40a6e9f4e456: Pull complete
d3dc5ec71e9d: Pull complete
Digest: sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67
Status: Downloaded newer image for nginx:latest
```

Figura 42 Descargar imagen nginx.

7. Crearemos un contenedor, para esto usaremos el siguiente comando. **docker run -dit --name nombre_app -p 8080:80 httpd.**

```
[root@ip-10-0-31-228 html]# docker run -dit --name app1 -p 8080:80 httpd
d7ee5520595fbff4e86222d332a67d8cefbf068468e5b05f191ebc20a1486c93
[root@ip-10-0-31-228 html]#
```

Figura 43 Creación del contenedor HTTPD.

8. Para verificar el o los contenedores que hay ejecutándose realizaremos el siguiente comando. **docker ps.**

```
[root@ip-10-0-31-228 html]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS
d7ee5520595f   httpd    "httpd-foreground"     About a minute ago   Up About a minute
0.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
[root@ip-10-0-31-228 html]#
```

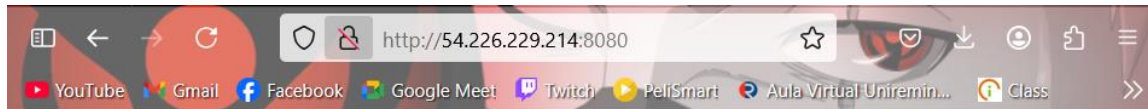
Figura 44 Verificar contenedores ejecutándose.

9. Luego de esto ya tenemos creado nuestro servidor web dentro del contenedor, pero debemos de tener en cuenta que el puerto mediante el cual va a escuchar el contenedor va a ser mediante el puerto 8080, y para lograr que se evidencie nuestra web necesitamos realizar el ingreso de una nueva regla en nuestro security group con el fin de que se permita este puerto.

The image shows the configuration for a new security group rule. The 'Security group rule ID' is currently empty. The 'Type' is set to 'Custom TCP'. The 'Port range' is set to '8080'. The 'Source type' is set to 'Anywhere-IPv4'. The 'Source' field contains '0.0.0.0/0' and has a search icon and a close button. There is also a 'Description - optional' field which is currently empty.

Figura 45 Puerto servidor web.

10. De esta manera ingresando a la dirección IP de nuestra instancia Linux, mediante un contenedor especificando el puerto anteriormente designado al contenedor, lograremos evidenciar que se muestra correctamente la web.



It works!

Figura 46 Evidencia del contenedor.

11. Existe la posibilidad de crear los contenedores que se quieran, pero teniendo en cuenta que no puede haber dos contenedores con el mismo nombre y con el mismo puerto asignado. Es decir que no podemos luego de haber creado nuestro anterior contenedor (app1, puerto 8080), crear otro con estas mismas características. En este caso crearemos otro contenedor con diferentes características.

```
[root@ip-10-0-31-228 html]# docker run -dit --name app2 -p 8082:80 httpd 2879413a143a063d3fff09f6734f48283af4c17ed5e2cd3f7ee5834c983adb59
```

Figura 47 Contenedor con diferentes características.

12. Realizando el comando docker ps luego de crear este nuevo contenedor lograremos ver que ya existen dos contenedores ejecutándose. Pero hay que tener en cuenta los pasos anteriormente realizados con el contenedor anterior, eso con el fin de que mediante este contenedor y su puerto se logre evidenciar el servicio web.

```
[root@ip-10-0-31-228 html]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        P
ORTS          NAMES
2879413a143a   httpd    "httpd-foreground"     3 seconds ago Up 2 seconds   0
.0.0.0:8082->80/tcp, :::8082->80/tcp   app2
d7ee5520595f   httpd    "httpd-foreground"     10 minutes ago Up 10 minutes  0
.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
[root@ip-10-0-31-228 html]#
```

Figura 48 Nuevo contenedor.

13. Para verificar el consumo de recursos de nuestra instancia, realizaremos la digitación del siguiente comando **top**.

```
top - 05:43:43 up 1:06, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 124 total, 1 running, 123 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 949.4 total, 69.2 free, 305.7 used, 574.5 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 472.3 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 32274 root      20   0  224012  3292  2764  R   6.2   0.3   0:00.01 top
    1 root      20   0  108156 18476 10676  S   0.0   1.9   0:01.65 systemd
    2 root      20   0     0     0     0  S   0.0   0.0   0:00.00 kthreadd
    3 root       0 -20     0     0     0  I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20     0     0     0  I   0.0   0.0   0:00.00 rcu_par+
    5 root       0 -20     0     0     0  I   0.0   0.0   0:00.00 slub_fl+
    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+
   10 root       0 -20     0     0     0  I   0.0   0.0   0:00.00 mm_perc+
   11 root      20   0     0     0     0  I   0.0   0.0   0:00.00 rcu_tas+
   12 root      20   0     0     0     0  I   0.0   0.0   0:00.00 rcu_tas+
   13 root      20   0     0     0     0  I   0.0   0.0   0:00.00 rcu_tas+
   14 root      20   0     0     0     0  S   0.0   0.0   0:00.34 ksoftir+
   15 root      20   0     0     0     0  I   0.0   0.0   0:00.76 rcu_pre+
   16 root      rt   0     0     0     0  S   0.0   0.0   0:00.01 migrati+
   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 kdevtmp+
   21 root       0 -20     0     0     0  I   0.0   0.0   0:00.00 inet_fr+
```

Figura 49 Verificar consumo de la instancia.

14. Para ver el consumo de los contenedores digitaremos el siguiente comando **docker stats**.

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
2879413a143a	app2	0.01%	6.824MiB / 949.4MiB	0.72%	2.41kB / 1.34kB	586kB / 4.1kB	82
d7ee5520595f	app1	0.01%	9.207MiB / 949.4MiB	0.97%	3.12kB / 1.34kB	3.28MB / 4.1kB	82

Figura 50 Verificar consumo de los contenedores.

15. Para detener un contenedor, debemos de tomar muy en cuenta el ID del contenedor el cual deseamos digitar, en este caso haremos el ejemplo con el contenedor identificado con el ID: 2879413a143a. De este modo si hacemos la verificación de los contenedores activos, este contenedor no se mostrará.

```
[root@ip-10-0-31-228 html]# docker stop 2879413a143a
2879413a143a
[root@ip-10-0-31-228 html]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS          P
ORTS          NAMES
d7ee5520595f   httpd    "httpd-foreground"     18 minutes ago Up 18 minutes   0
.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
[root@ip-10-0-31-228 html]#
```

Figura 51 Detener un contenedor.

16. Para verificar los contenedores que tenemos activos o los que hemos eliminado, digitaremos el siguiente comando **docker ps -a**. Con este ejemplo en cuestión, podemos evidencia que hay uno de los contenedores en el estado **EXITED**, esto significa que este contenedor ha sido eliminado, pero se puede volver a iniciar sin ningún inconveniente teniendo en cuenta que debe de ser iniciado con su ID.

```
[root@ip-10-0-31-228 html]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
2879413a143a   httpd    "httpd-foreground"     9 minutes ago   Exited (0) About
a minute ago   app2
d7ee5520595f   httpd    "httpd-foreground"     19 minutes ago   Up 19 minutes
0.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
[root@ip-10-0-31-228 html]#
```

Figura 52 Verificar contenedores activos.

17. Para volver a iniciar el contenedor eliminado con anterioridad, usaremos el siguiente comando **docker start 2879413a143a** (ID del contenedor eliminado).
 “Podemos verificar si está activo con docker ps”

```
[root@ip-10-0-31-228 html]# docker start 2879413a143a
2879413a143a
[root@ip-10-0-31-228 html]#
```

Figura 53 Iniciar contenedor eliminado.

18. Si deseamos modificar el sitio web que está en un contenedor en específico, debemos de realizar lo siguiente. (En este caso lo modificaremos mostrando una plantilla descargada desde internet de HTML). Para tener claridad acerca de cómo es que una plantilla se puede descargar en un contenedor, primero copiaremos el enlace de descarga de la plantilla, luego de ello descargamos la plantilla con el comando **wget link_descarga_plantilla**. Pero es primordial de saber que tipo de archivo es esa plantilla, en este caso el tipo de plantilla es un archivo .zip, entonces siendo de esa manera se debe de descomprimir el archivo con este comando **unzip nombre_archivo**.

```

[root@ip-10-0-31-228 ec2-user]# mkdir app2
[root@ip-10-0-31-228 ec2-user]# wget https://html5up.net/story/download
--2025-05-20 06:27:16-- https://html5up.net/story/download
Resolving html5up.net (html5up.net)... 172.67.195.190, 104.21.76.136, 2606:4700:
3036::ac43:c3be, ...
Connecting to html5up.net (html5up.net)|172.67.195.190|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-zip]
Saving to: 'download'

download          [ <=>          ] 1.62M  --.-KB/s   in 0.03s

2025-05-20 06:27:16 (49.3 MB/s) - 'download' saved [1700827]

[root@ip-10-0-31-228 ec2-user]# file download
download: Zip archive data, at least v2.0 to extract
[root@ip-10-0-31-228 ec2-user]# unzip download
Archive:  download
  inflating: LICENSE.txt
   creating: assets/
   creating: assets/webfonts/
  inflating: assets/webfonts/fa-regular-400.woff
  inflating: assets/webfonts/fa-regular-400.eot
  inflating: assets/webfonts/fa-brands-400.eot
  inflating: assets/webfonts/fa-solid-900.svg
  inflating: assets/webfonts/fa-regular-400.ttf
  inflating: assets/webfonts/fa-brands-400.woff2

```

Figura 54 Agregar plantilla HTML.

19. Luego de esto hay que saber que esta plantilla aún no se encuentra dentro de ninguno de los contenedores, entonces se debe de ejecutar esta página web dentro de un contenedor. Para esto hay que tener en cuenta que cada uno de los contenedores existentes en los que se necesite mostrar la plantilla, deberán de apuntar hacia la carpeta la cual contiene la plantilla. /usr/local/apache2/htdocs. /home/ec2-user/app2.
20. Crearemos un nuevo contenedor el cual le agregaremos un nuevo parámetro, el cual será -v, de este modo usaremos el comando de creación de contenedor, pero agregando lo siguiente: docker run -dit --name app3 -p 8083:80 -v /home/ec2-user/app2 (Estoy queriendo decir que se va a montar un volumen en el contenedor que está ubicado en esa dirección): /usr/local/apache2/htdocs (Ruta a la cual el contenedor en cuestión va a buscar la plantilla) httpd.

```
[root@ip-10-0-31-228 ec2-user]# docker run -dit --name app3 -p 8083:80 -v /home/ec2-user/app2:/usr/local/apache2/htdocs httpd
73d567dea322c6de0513b2e6f462bb82fa71370960c0694316092c ffa0108350
[root@ip-10-0-31-228 ec2-user]#
```

Figura 55 Nuevo contenedor con búsqueda de plantilla.

21. Para lograr que se muestre el resultado de la plantilla desde internet no hay que olvidar que debido a que se creó un nuevo contenedor y que necesita un puerto, hay que agregar una nueva regla en el security group de la instancia la cual permita el puerto 8083 asignado al contenedor.

The screenshot shows the AWS Security Groups console interface for adding a new rule. The fields are as follows:

- Security group rule ID:** -
- Type:** Custom TCP
- Protocol:** TCP
- Port range:** 8083
- Source type:** Anywhere-IPv4
- Source:** 0.0.0.0/0
- Description - optional:** (empty field)
- Buttons:** Add rule

Figura 56 Mostrar plantilla desde internet.

22. Luego de esto verificamos que se esté ejecutando en el contenedor el sitio.

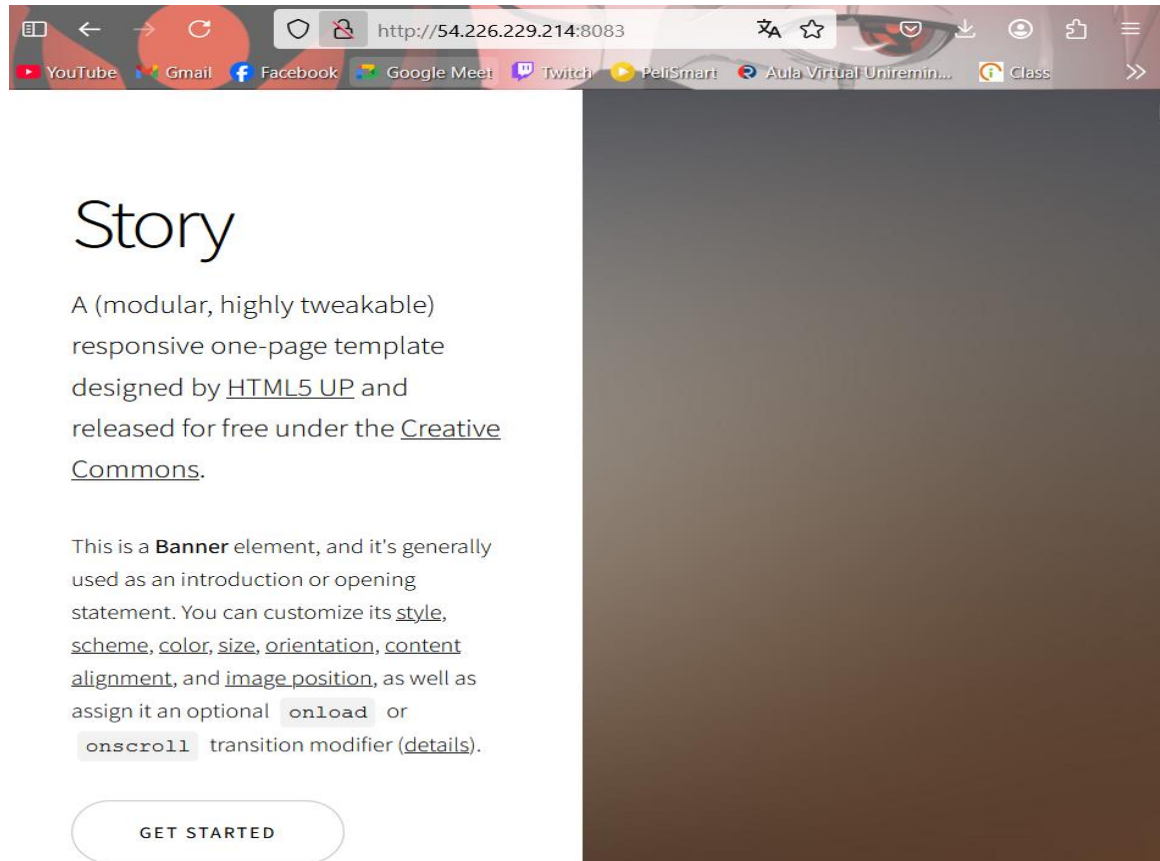


Figura 57 Ejecución del contenedor del sitio.

23. Se creará otro contenedor con la misma ruta, pero con diferente nombre y puerto, para luego repetir los mismos pasos y permitir mediante el security group de la instancia que el puerto siguiente puerto este habilitado. Nombre contenedor: app4 y puerto: 8084. De este modo puedo tener muchos contenedores, apuntando a la misma aplicación. Si un contenedor se daña los demás tienen la misma aplicación.

```
[root@ip-10-0-31-228 ec2-user]# docker run -dit --name app4 -p 8084:80 -v /home/ec2-user/app2:/usr/local/apache2/htdocs httpd
7e5fe0013c497984e76888dd3e073bfec37db13ce5ccd4dba6f07c6ca5f3ffe9
[root@ip-10-0-31-228 ec2-user]#
```

Figura 58 Creación del contenedor con la misma ruta.

The image shows two parts: a screenshot of the AWS Security Groups console and a browser window.

AWS Security Groups Console: The screenshot shows the configuration for 'Inbound rule 5'. The 'Security group rule ID' is 'sgr-0f417354caab634a6'. The 'Type' is 'Custom TCP'. The 'Protocol' is 'TCP'. The 'Port range' is '8084'. The 'Source type' is 'Custom'. The 'Source' is '0.0.0.0/0'. There is a 'Delete' button in the top right corner.

Browser Window: The browser shows the URL 'http://54.226.229.214:8084'. The page content includes the heading 'Story' and the text: 'A (modular, highly tweakable) responsive one-page template designed by [HTML5 UP](#) and released for free under the [Creative Commons](#).' Below the text is a 'GET STARTED' button.

Figura 59 Creación del contenedor con la misma ruta

24. Existe un gran inconveniente y es que, para acceder a cada uno de los contenedores y su aplicativo web, hay que especificar la dirección IP junto con el

puerto específico del contenedor. Para solucionar esto se hará lo siguiente.

Crearemos un proxy reverso, pero con fines educativos lo haremos implementándolo con los servicios de nginx.

25. Para instalar el servicio de nginx digitaremos el siguiente comando **dnf install nginx**.

```
[root@ip-10-0-31-228 ec2-user]# docker run -dit --name app4 -p 8084:80 -v /home/ec2-user/app2:/usr/local/apache2/htdocs httpd
7e5fe0013c497984e76888dd3e073bfec37db13ce5ccd4dba6f07c6ca5f3ffe9
[root@ip-10-0-31-228 ec2-user]# dnf install nginx
Last metadata expiration check: 2:20:48 ago on Tue May 20 04:42:12 2025.
Dependencies resolved.
=====
Package                Arch      Version                               Repository      Size
=====
Installing:
nginx                  x86_64    1:1.26.3-1.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    1:1.26.3-1.amzn2023.0.1             amazonlinux     670 k
nginx-filesystem      noarch    1:1.26.3-1.amzn2023.0.1             amazonlinux     9.6 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
Is this ok [y/N]: y
Downloading Packages:
```

Figura 60 Instalar servicio nginx.

26. Luego de esto hay que repetir los pasos que ya se aprendieron anteriormente, los pasos de revisar el estado y el inicio del servicio.

```
[root@ip-10-0-31-228 ec2-user]# systemctl start nginx
[root@ip-10-0-31-228 ec2-user]#
```

Figura 61 Inicio servicio nginx.

27. Se debe de agregar una nueva regla la cual admita el puerto 80, debido que es en ese puerto que el servicio nginx arranca por defecto.

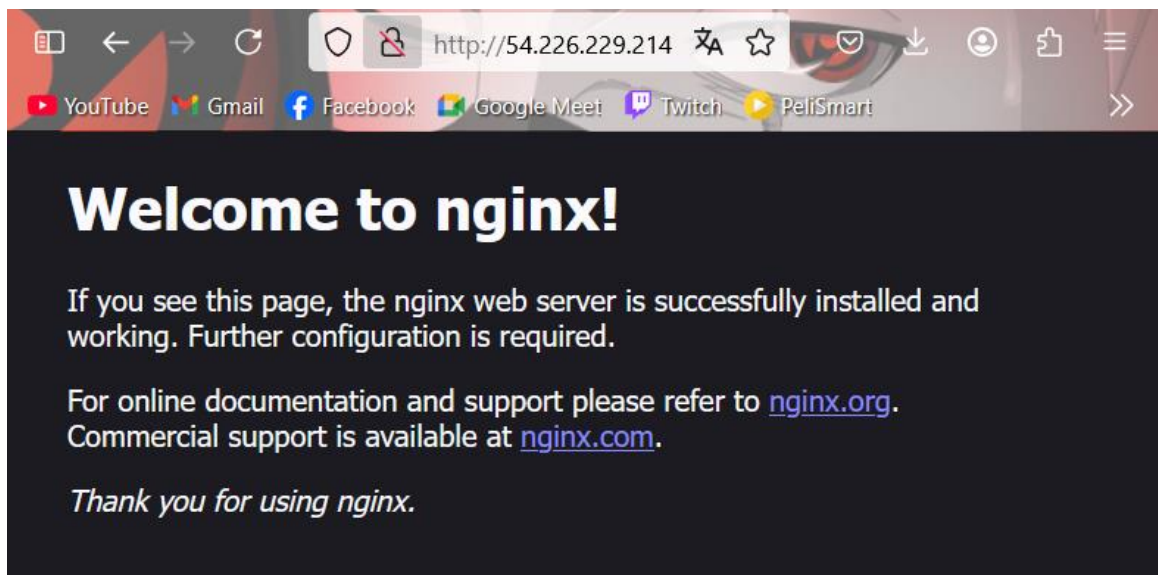


Figura 62 Nueva regla para el puerto 80 nginx.

28. Configuración del servicio de proxy reverso. Para eso hay que acceder a la carpeta (/etc/nginx) ahí están los archivos de configuración de nginx, allí trabajaremos con el archivo nginx.conf. Pero primero es recomendable hacer una copia de seguridad de ese archivo con el comando `cp nginx.conf nginx.conf.BK`. Luego de esto se procede a editar el archivo con el siguiente comando `nano nginx.conf`.

```

GNU nano 8.3          nginx.conf
# For more information on configuration, see:
# * Official English Documentation: http://nginx.org/en/docs/
# * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

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

    access_log /var/log/nginx/access.log main;

    sendfile            on;
    tcp_nopush          on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

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

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/nginx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

```

Figura 63 Configuración proxy reverso.

29. Ese archivo es el cual hace funcionar el aplicativo web de nginx que viene por defecto; de igual manera este archivo también tiene la configuración de puertos y todo lo necesario para un servicio web, pero en este caso no necesitamos este servicio, necesitamos configurar el proxy reverso y para eso borramos todo lo que encontramos en este archivo y digitamos lo siguiente:

```
GNU nano 8.3          nginx.conf          Modified
events{}

http {
    upstream seminario{
        server localhost:8082;
        server localhost:8083;
    }

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

Figura 64 Modificación archivo nginx.conf.

30. Cada vez que se recarga la página, él va a mostrar cualquiera de los dos contenedores, ya sea el contenedor del puerto 8082 o el 8083.



It works!

Figura 65 Evidencia implementación de proxy reverso.

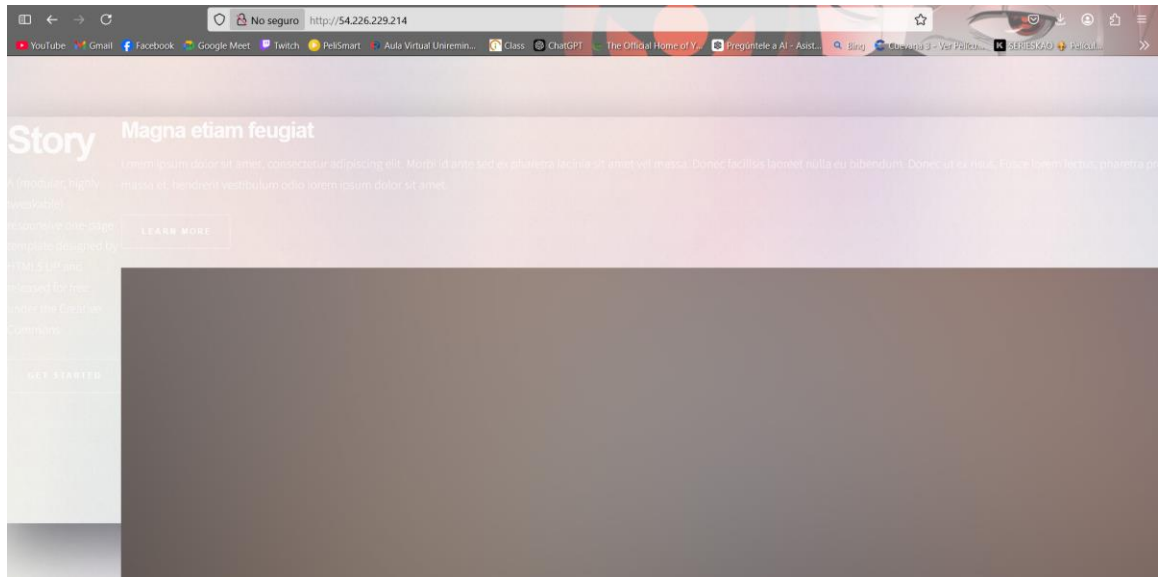


Figura 66 Evidencia implementación de proxy reverso.

31. Ventajas del uso de contenedores frente a la virtualización: Los contenedores consumen una menor cantidad de recursos, debido a que comparten el mismo núcleo del sistema operativo, se inician rápidamente y consumen menos memoria. Además de esto los contenedores pueden iniciarse en segundos, ideal para escalar automáticamente o hacer algún despliegue rápido. En cuanto a la escalabilidad, los contenedores se integran de manera fácil con sistemas de orquestación como Kubernetes, lo cual los hace ideales para una arquitectura ideal y moderna.

Prueba de estrés.

Realizaremos una prueba de estrés al contenedor Linux con ab(Apache Benchmark).

1. Realizaremos el análisis del consumo de CPU y RAM del contenedor.

```
top - 04:31:52 up 23:54, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 141 total, 1 running, 140 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 92.9 id, 0.0 wa, 0.0 hi, 0.0 si, 6.8 st
MiB Mem : 949.4 total, 103.9 free, 309.6 used, 535.9 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 475.4 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
38275	root	20	0	1877372	20172	14884	S	0.3	2.1	0:23.66	contain+
40167	apache	20	0	1086000	7752	4732	S	0.3	0.8	0:13.37	httpd
40168	apache	20	0	1249904	7752	4732	S	0.3	0.8	0:13.37	httpd
1	root	20	0	108156	18316	10512	S	0.0	1.9	0:09.65	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_part

Figura 67 Análisis del consumo de recursos de la instancia.

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
7e5fe0013c49	app4	0.00%	6.641MiB / 949.4MiB	0.70%	35kB / 770kB	135kB / 4.1kB	82
73d567dea322	app3	0.00%	6.816MiB / 949.4MiB	0.72%	96.9kB / 1.54MB	0B / 4.1kB	82
2879413a143a	app2	0.00%	6.727MiB / 949.4MiB	0.71%	67.9kB / 64.5kB	24.6kB / 4.1kB	82
d7ee5520595f	app1	0.00%	11.14MiB / 949.4MiB	1.17%	75.6kB / 85.8kB	3.28MB / 4.1kB	109

Figura 68 Análisis del consumo de recursos de los contenedores

- Luego de esto instalaremos Apache Benchmark, que es la herramienta con la cual aumentaremos el consumo. Con el comando **sudo yum install httpd-tools** lo haremos.

```
[root@ip-10-0-31-228 ec2-user]# sudo yum install httpd-tools
Last metadata expiration check: 23:55:52 ago on Tue May 20 04:42:12 2025.
Package httpd-tools-2.4.62-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Figura 69 Instalación Apache Benchmark.

- Ejecutamos **ab -n 10000 -c 100 <http://localhost:8080/>** el cual envía 10mil peticiones en total, con 100 usuarios simultáneos. Según lo evidenciado en la

imagen consiguiente el resultado obtenido fue un total de 4157 solicitudes; esto quiere decir que probablemente se alcanzaron los límites del host (instancia en cuestión) lo cual ocasiono que no se recibieran más solicitudes

```
[root@ip-10-0-31-228 ec2-user]# ab -n 10000 -c 100 http://localhost:8080/
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
^C

Server Software:      Apache/2.4.63
Server Hostname:     localhost
Server Port:         8080

Document Path:      /
Document Length:    45 bytes

Concurrency Level:   100
Time taken for tests: 14.658 seconds
Complete requests:   4157
Failed requests:     0
Total transferred:  1201373 bytes
HTML transferred:   187065 bytes
Requests per second: 283.60 [#/sec] (mean)
Time per request:   352.606 [ms] (mean)
Time per request:   3.526 [ms] (mean, across all concurrent requests)
Transfer rate:      80.04 [Kbytes/sec] received
```

Figura 70 Saturación de solicitudes del contenedor.

4. En este punto evidenciaremos la cantidad límite de contenedores los cuales admite la instancia en cuestión. Primero que todo mostraremos la evidencia de las condiciones de consumo que posee la instancia antes de realizar la prueba.

```
top - 04:53:23 up 1 day, 16 min, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 141 total, 1 running, 140 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 97.1 id, 0.0 wa, 0.0 hi, 0.0 si, 2.6 st
MiB Mem : 949.4 total, 97.4 free, 328.5 used, 523.5 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 457.4 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
30175	root	20	0	1744948	43360	27136	S	0.3	4.5	1:08.13	contain+
1	root	20	0	108156	18304	10500	S	0.0	1.9	0:09.77	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_part

Figura 71 Consumo de los recursos de la instancia.

5. Luego de esto crearemos un script el cual levantara contenedores httpd con nombre único, esperara 2 segundos entre cada creación para no saturar simultáneamente la instancia, mostrara los contenedores creados y haremos el monitoreo con el comando **top**. En el terminal ejecutaremos nano levantar_contenedores.sh (nombre script) y luego pegaremos el siguiente código.

```
GNU nano 8.3          levantar_contenedores.sh          Modified
#!/bin/bash

max_contenedores=20 # Puedes cambiar este número
delay=2             # Segundos entre cada creación

for i in $(seq 1 $max_contenedores); do
  docker run -d --name test_httpd_$i httpd
  echo "Contenedor test_httpd_$i creado."
  sleep $delay
done
```

Figura 72 Script de la creación de 20 contenedores.

6. Como paso final daremos permisos de ejecución al archivo con el comando `chmod +x levantar_contenedores.sh`.

```
[root@ip-10-0-31-228 ec2-user]# chmod +x levantar_contenedores.sh
[root@ip-10-0-31-228 ec2-user]#
```

Figura 73 Permisos de ejecución al script

7. Y por último solo queda ejecutar el script con `./levantar_contenedores.sh` y realizar el monitoreo.

```
Tasks: 253 total,  2 running, 251 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.6 us,  0.9 sy,  0.0 ni, 93.4 id,  0.0 wa,  0.0 hi,  0.0 si,  5.1 st
MiB Mem :   949.4 total,    60.5 free,   593.4 used,   295.5 buff/cache
MiB Swap:    0.0 total,    0.0 free,    0.0 used.   193.0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
110877	root	20	0	0	0	0	I	0.3	0.0	0:00.04	kworker+
111009	root	20	0	1877372	18032	12976	S	0.3	1.9	0:00.03	contain+
114295	root	20	0	224144	3588	2772	S	0.3	0.4	0:00.12	top
116082	root	20	0	224144	3592	2772	R	0.3	0.4	0:00.03	top
1	root	20	0	108156	17828	10000	S	0.0	1.8	0:10.19	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+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	slub_fl+
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+
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_perc+
11	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tas+
14	root	20	0	0	0	0	S	0.0	0.0	0:02.71	ksoftir+
15	root	20	0	0	0	0	I	0.0	0.0	0:01.74	rcu_pre+
16	root	rt	0	0	0	0	S	0.0	0.0	0:00.41	migrati+
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	kdevtmp+
21	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_fr+
22	root	20	0	0	0	0	S	0.0	0.0	0:00.07	kauditd
23	root	20	0	0	0	0	S	0.0	0.0	0:00.07	khungta+

Figura 74 Consumo de recursos de la instancia con 20 contenedores

8. Al momento de terminar la creación de los primeros 20 contenedores y realizar el monitoreo se evidencio que aún se podían crear más contenedores. Pero no se podía realizar la creación con el mismo script debido a que ocurriría un error

puesto que se crearían más contenedores con el mismo nombre, es por esto que se tuvo que realizar una modificación al script con el fin de que leyera los scripts existentes y creara más con nombres diferentes a partir del contenedor número 20.

```

max_contenedores=50 # nuevo límite mayor que 20
delay=2

# Contar contenedores existentes con nombre test_httpd_
existentes=$(docker ps -a --filter "name=test_httpd_" --format "{{.Names}}" | wc -l)

echo "Contenedores existentes: $existentes"

if [ $existentes -ge $max_contenedores ]; then
    echo "Ya tienes $existentes contenedores, que es igual o mayor al límite ($max_contenedores)."
    echo "No se crearán más contenedores."
    exit 0
fi

faltantes=$((max_contenedores - existentes))

echo "Creando $faltantes contenedores nuevos..."

ultimo_num=$(docker ps -a --filter "name=test_httpd_" --format "{{.Names}}" | \
    sed 's/test_httpd_/' | sort -n | tail -1)

if [ -z "$ultimo_num" ]; then
    ultimo_num=0
fi

inicio=$((ultimo_num + 1))
fin=$((ultimo_num + faltantes))

for i in $(seq $inicio $fin); do
    docker run -d --name test_httpd_$i httpd
    echo "Contenedor test_httpd_$i creado."
    sleep $delay
done

echo "Proceso terminado."

```

Figura 75 Creación de los demás contenedores con otro script

9. Luego de esto se realizó nuevamente la ejecución del script con `./levantar_contenedores.sh` y se logró evidenciar que se empezaron a crear más contenedores a partir del 21 hasta el 39. Fue luego de la creación del contenedor número 39 que la instancia dejó de responder y la terminal quedó totalmente

bloqueada sin permitir realizar la interrupción de la ejecución del script. Esto indicó que se alcanzó el límite de los recursos de la instancia.

```

Creando 30 contenedores nuevos...
deaa7d03794bb7b127b8e4e8ef2cf3223c6f754d30272583f5fc1ea13e951de1
Contenedor test_httpd_21 creado.
7af5e841ec6553d0170ee7a1506a4bdce9313ac5b2f2b406716353b26fef06be
Contenedor test_httpd_22 creado.
6ca2b1360a013996b034638a388fe05c8aac7d4f0706e6244592926584140407
Contenedor test_httpd_23 creado.
88a832b61ccece7fa84d38573e3f9fd7e67ddff7b2331083f1a2562bfccf3014
Contenedor test_httpd_24 creado.
1d510fc7bb09a5a4db5643bb36a419440a1f7fde0d4b35c2fbd54a0c8aac9736
Contenedor test_httpd_25 creado.
68cae6f844c578d48136b7045af160aa1e03ddef26e0308c00be71d815e0277d
Contenedor test_httpd_26 creado.
7f98e17007a54c998f42110dc99f77eaf786b54aa09fd2a5ef8f69b86d4078e6
Contenedor test_httpd_27 creado.
4152991b6f59bb43df3f997318e1eccee49be1c62ec0999ead67f6f1c8a59c48
Contenedor test_httpd_28 creado.
614b2d5c2f0aa36c80e583d777285b11278acf474e4cc544a15bd66cb0140aef
Contenedor test_httpd_29 creado.
03771b025401f1a5355c9a37eac2791470525d810144989f48607e690c909b36
Contenedor test_httpd_30 creado.
54fb4d689ac3d60ca4d7a1349d05db0bae903472a7b6121989cc58e722289a0b
Contenedor test_httpd_31 creado.
acd96d02a44ed6bc3e48a77a58999cab098fa6e62267e64ff1b279fec4c2d7cd
Contenedor test_httpd_32 creado.
0733e27e656d648620ae8cb2a7e25218711eaa8e942fb8b69cec9a16d57be3c6
Contenedor test_httpd_33 creado.
7e729f71c13e65667ba598c22bee49cdf836e8b5fa69cbea4001f3a778049d5
Contenedor test_httpd_34 creado.
42f49806c87580ea40c2b8d94f76a802b3d5e8a74b9a931d691e3f793bd22f8f
Contenedor test_httpd_35 creado.
ee3f372597944b47311c0d40f5ae6ddcc493b4ba0b0d4402f7b505910d228037
Contenedor test_httpd_36 creado.
90835026e21737ce038712b387fffedc3544bb5b1f6c7b91827dd0675e3a8b1c
Contenedor test_httpd_37 creado.
07a7d439ff5ac1e4a45dc7e7264eab5366d0e1765ff166e70bfb3a61749c70a4
Contenedor test_httpd_38 creado.
97bb5848b821bc07660afab6aeaa663889aac7cc063c1be7953d09bcc7155f69
Contenedor test_httpd_39 creado.
9a35480b5f4f3500d4a8658bcd53146698e9b9824349102b9f3f04b0f796771
^C^Ctoptop^C

```

Figura 76 Limite de creación de contenedores

10. Aspecto a tener en cuenta: La instancia soporto un total de 43 contenedores, debido a que ya existían 4 contenedores antes de realizar la ejecución del script.

Logramos evidenciar desde la opción de monitoreo de la instancia desde AWS, que la misma alcanzo el límite de uso de CPU con la cantidad de 43 contenedores ejecutándose.

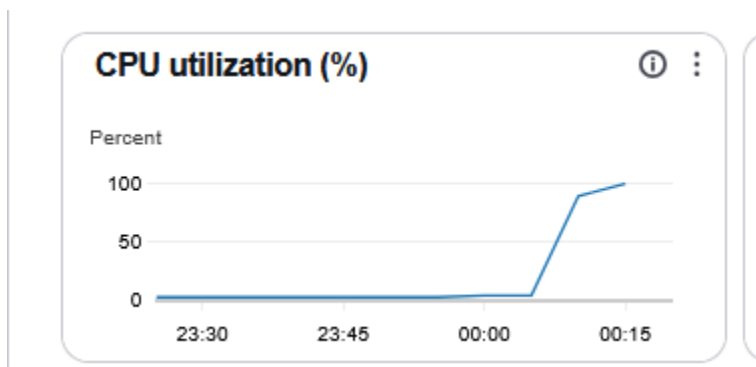


Figura 77 Consumo de CPU máximo.

Conclusiones

Dentro de lo aprendido lo que fue el desplegar servidores virtuales (instancias EC2), configurarlos con sistemas operativos habilitar el acceso seguro mediante protocolos como SSH y RDP, se comprendió la importancia de los grupos de seguridad, la asignación de puertos y la configuración de red (VPC, subredes e Internet Gateway), emulando el comportamiento de una red empresarial moderna.

Uno de los componentes de AWS que fue más llamativos fue la implementación de Docker que permitió agregar contenedores y aplicaciones y además poder manejar su ejecución con eficiencia y facilidad. Se abordaron conceptos como imágenes, volúmenes, persistencia de datos y automatización en Docker Compose el cual es una de las herramientas clave en el desarrollo de software actual.

Aparte de los conocimientos técnicos aprendidos de este proyecto también aportó habilidades como documentación, resolución de errores, y buenas prácticas en la nube como el manejo de claves, control de accesos y uso eficiente de los recursos. En cada concepto hubo desafíos, pero también crecimiento al enfrentarlos.

En conclusión, este trabajo no solo permitió adquirir habilidades técnicas, sino que también abrió la puerta para seguir desarrollando habilidades en el mundo de la computación en la nube, que hoy en día representa una necesidad para las empresas modernas que buscan soluciones escalables, seguras y eficientes.

Referencias

(Puedes citar con normas APA o Vancouver. Se anexa ejemplo de normas APA)

Borges, J.L. (2013). *Ficciones*. Buenos Aires, Argentina: Debolsillo.

Bastidas, L.R. (2007). *El inicio del siglo XXI*. Planeta. Sitio web:
<http://www.rbastidasl.com/libro-inicio-del-sigloxxi>.

- Amazon Web Services. (s.f.). *Amazon EC2*. Recuperado de <https://aws.amazon.com/ec2/arXiv+3Wikipedia+3Wikipedia+3>