TRABAJO DE GRADO
Opción Seminario-Diplomado.

Servicios Web en AWS con EC2 y Contenedores Docker

Corporación Universitaria Remington.
Facultad Ingenierías
Ingeniería de Sistemas

Ever Quinto Córdoba
Juan Felipe Morales Cadavid
Julián Andrés Bastidas Gómez

Docente: Juan Pablo Berrio López
Opción de Trabajo de grado Seminario-Diplomado.
2025

Tabla de Contenidos

Resumen

Este trabajo nace del interés por entender y aplicar de forma práctica cómo funciona el despliegue de servicios web en la nube, usando herramientas que hoy en día son muy comunes en el mundo laboral, como Amazon Web Services (AWS). Durante el proyecto, se configuraron servidores con sistemas operativos Windows y Linux dentro de AWS, buscando simular lo que pueda pasar en una empresa real, donde diferentes tecnologías deben trabajar juntas.

Además, se usaron contenedores Docker, que son como cajitas donde uno puede guardar una aplicación y llevarla a cualquier parte sin preocuparse por si va a funcionar o no. Esto permitió montar varios sitios web desde un mismo servidor, usando diferentes puertos, y entender mejor cómo escalar un servicio para que responda bien, incluso cuando hay muchos usuarios al mismo tiempo.

Todo el proceso, desde crear los servidores hasta instalar y probar los servicios web, fue una experiencia muy enriquecedora. No solo ayudó a fortalecer conocimientos técnicos, sino también a valorar la importancia de la organización, la paciencia y el trabajo en equipo. Al final, más que un ejercicio académico, este proyecto se convirtió en una muestra del potencial que tenemos cuando combinamos lo aprendido en clase con las ganas de experimentar y seguir creciendo como profesionales.

Palabras clave

1. Nube
2. Servidores
3. Docker
4. AWS
5. Servicios web

Marco conceptual y contextual

En este trabajo de grado se manejan conceptos clave que se relacionan con la computación en la nube, despliegue de servicios web y el uso de contenedores con Docker, Para tener un buen entendimiento de estos conceptos se desarrolla una explicación clara y estructurada.

Computación en la nube

Esta se basa en un modelo que permite acceder a recursos como servidores, redes o almacenamiento a través de la web. Este enfoque crea la posibilidad de eliminar la necesidad de tener que desarrollar una infraestructura física local, esto gracias a que todos estos servicios se alojan en centros de datos remotos. Con esta tecnología, tanto personas como organizaciones pueden gestionar entornos de forma flexible, escalable, segura y rentable.

Amazon Web Services (AWS)

AWS es una plataforma de servicios en la nube la cual se utiliza ampliamente a nivel mundial. Esta ofrece una variedad de herramientas para crear, desplegar y administrar aplicaciones y sistemas. Analizando sus servicios, entre los más destacados se encuentran los servicios relacionados con almacenamiento, bases de datos, redes, IA y cómputo. En el proyecto realizada como trabajo de grado, AWS sirve como base para la creación y gestión de servidores virtuales.

Amazon EC2

Amazon EC2, se conoce como un servicio que permite a los usuarios crear y controlar instancias de máquinas virtuales dentro de AWS. En estas instancias se pueden ejecutar sistemas operativos como los usados en este proyecto (Windows y Amazon Linux) proporcionando un entorno completo para realizar la instalación de aplicaciones, configurar servicios y alojar sistemas web. EC2 es el servicio con mayor popularidad para la creación de soluciones en la nube para modelos de IaaS.

Infraestructura como Servicio (IaaS)

La infraestructura como servicio es un modelo de computación basada en la nube que permite utilizar recursos virtuales bajo demanda como servidores, redes o sistemas operativos. En este tipo de modelos, el usuario tiene control total sobre la configuración y gestión de los recursos sin la necesidad de manejar hardware físico. En este trabajo de grado, el uso de instancias EC2 nombradas anteriormente, ejemplifica la manera en la que se trabaja este modelo de IaaS en AWS.

Servicios Web

Los servicios web son aplicaciones almacenadas en servidores a las que se pueden acceder a través de redes normalmente mediante un protocolo llamado HTTP. Estos servicios permiten acceder a información o funcionalidades de forma remota a los usuarios. Para el

desarrollo del proyecto se hace uso de 2 servicios web, uno basado en IIS (Internet Information Services) almacenado en un servidor Windows y otro usando Apache HTTPD en un servidor Linux. Ambos cumpliendo con el propósito principal del proyecto de permitir el acceso a paginas y/o aplicaciones web a través de internet.

Docker y Contenedores

Docker es una herramienta en la que se pueden crear contenedores, los cuales son entornos aislados que contienen lo necesario para ejecutar una aplicación. Gracias a su portabilidad y eficiencia, los contendores se han vuelto una solución muy acertada en el desarrollo y despliegue de aplicaciones de software. Como parte de este proyecto, Docker se utiliza para poder ejecutar el servicio Apache dentro de un contenedor en un servidor Linux.

Acceso Remoto: RDP y SSH

Para lograr administrar correctamente y de manera segura los servidores en la nube, se utilizan protocolos de acceso remoto. Para las instancias alojadas en sistemas operativos como Windows, se hace uso del protocolo de escritorio remoto (RDP), el cual permite el acceso de manera gráfica al sistema. Para la instancia Linux se usó Secure Shell (SSH), que permite una conexión segura por la línea de comandos. Estos métodos facilitan la configuración y control remoto de los servidores, garantizando un acceso eficiente y seguro.

En la actualidad, la transformación digital ha impulsado una evolución a gran escala en la forma en la que las empresas y organizaciones gestionan sus recursos tecnológicos. Teniendo en cuenta este panorama, resuena mucho el tema de computación en la nube, la cual se ha consolidado como una opción eficiente, segura, escalable y económica para alojar, desplegar y administrar infraestructura tecnológica sin depender de hardware físico local. Amazon Web Services se posiciona como uno de los proveedores más relevantes a la hora de hablar de tecnología en la nube, ofreciendo servicios con los cuales se logran crear entornos de trabajo adaptables a diferentes necesidades empresariales de manera virtual.

Teniendo en cuenta este contexto, aparecen servicios como EC2 de AWS el cual consiste en una herramienta clave para simular entornos en la nube. Este tipo de soluciones son muy útiles en casos empresariales tanto académicos y de formación profesional, esto ya que permite poner en práctica conceptos de redes, administración de sistemas operativos y despliegue de servicios web, todo esto sin necesidad de hacer uso de hardware propio. Además, posibilita trabajar bajo un enfoque de IaaS, modelo que se ha convertido en un estándar en el desarrollo y operación de sistemas en tiempos modernos.

Este proyecto surge como una iniciativa académica orientada a fortalecer las competencias y habilidades técnicas en el ámbito de despliegue de servidores y servicios web. De esta manera se realiza la creación de un entorno hibrido que integre instancias EC2 con SO Windows y Amazon Linux. Con esta configuración se logra simular un escenario empresarial real, en el que ambos sistemas coexisten y colaboran a través de servicios accesibles desde internet. A practica se complementa al hacer uso de herramientas como IIS en Windows y

Apache HTTPD tanto de forma directa dentro de Linux incorporando contenedores con Docker, lo que permite explorar soluciones tradicionales, así como también tecnologías emergentes.

En definitiva, este ejercicio académico no solo busca cumplir con los objetivos técnicos propuestos, sino también dar a conocer una experiencia formativa integral que permita proparar al estudiante para enfrentarse a retos actuales en entornos digitales. El conocimiento que se logra adquirir a través de esta práctica servirá como base para futuros desarrollos e implementaciones en entornos empresariales reales contribuyendo con la mejora de habilidades orientadas a la gestión de soluciones tecnológicas basadas en la nube.

Introducción

En los últimos años, el crecimiento de la computación en la nube ha cambiado por completo la forma en que personas y empresas manejan sus sistemas y servicios digitales. Plataformas como Amazon Web Services (AWS) han hecho posible levantar servidores virtuales de manera rápida, confiable y sin necesidad de tener infraestructura física, lo cual es clave hoy en día.

Este proyecto tiene como objetivo mostrar cómo se pueden crear y configurar servidores tanto en Windows como en Linux dentro de AWS, usando las instancias EC2. La idea es simular un escenario parecido al de una empresa, donde los dos sistemas operativos trabajan juntos y ofrecen servicios accesibles desde cualquier lugar por medio de Internet.

Durante el desarrollo del proyecto se llevará a cabo todo el proceso: desde la creación de redes y llaves de acceso, hasta la configuración de los servicios web (IIS en Windows y Apache HTTPD en Linux, incluso dentro de contenedores Docker). Además, se comprueba que todo funcione correctamente accediendo desde navegadores externos. Esta práctica será muy útil para reforzar conocimientos sobre administración de servidores, seguridad en la nube y todo lo relacionado con el despliegue de infraestructura como servicio (IaaS).

Objetivos Generales

Implementar y configurar servidores Windows y Linux dentro de Amazon Web Services (AWS) utilizando instancias EC2, con el fin de recrear un entorno empresarial funcional que permita el despliegue de servicios web y su gestión remota de manera segura y controlada.

Objetivos Específicos

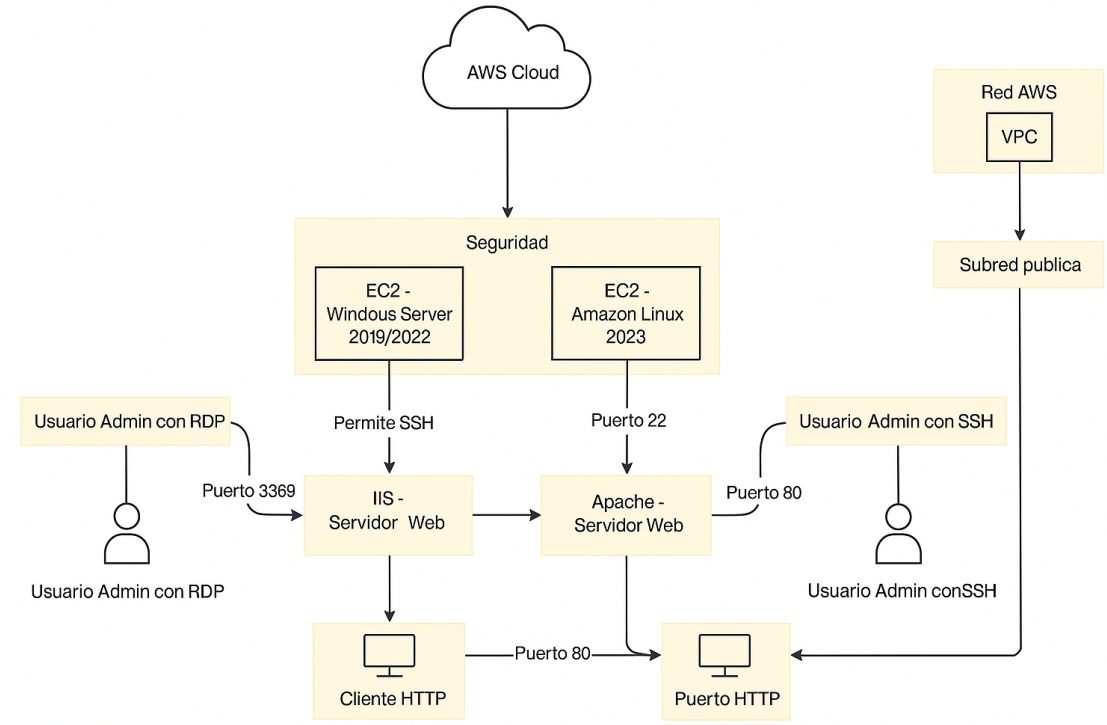
Realizar la configuración de redes, reglas de seguridad y accesos necesarios en cada instancia EC2 para garantizar una comunicación fluida entre los servidores y el exterior.

- Instalar y poner en marcha servicios web en cada sistema operativo, utilizando IIS para Windows y Apache en Linux, incluyendo el despliegue del servidor Apache dentro de un contenedor Docker para reforzar la práctica.
- Verificar el correcto funcionamiento de los servicios desplegados accediendo desde navegadores y mediante protocolos de administración remota como RDP y SSH.
- Desarrollar una base sólida para el manejo de entornos híbridos y abrir camino hacia la implementación de soluciones más avanzadas basadas en la nube.

1. Servicios Web en AWS con EC2 y Contenedores Docker

Diagrama arquitectónico

Figura: 1

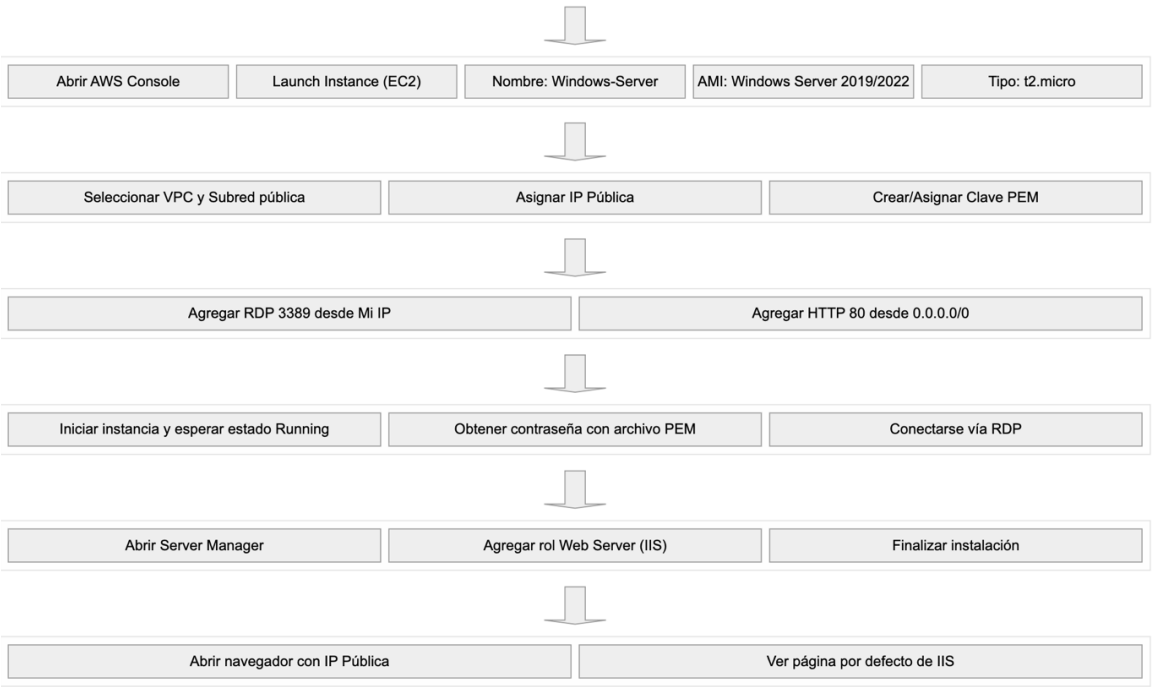


Fuentes: propia

1.1. Implementación de Windows Server en AWS

Diagrama de flujo Windows Server

Figura: 2



Fuentes: propia

Acceder a la consola de administración de AWS

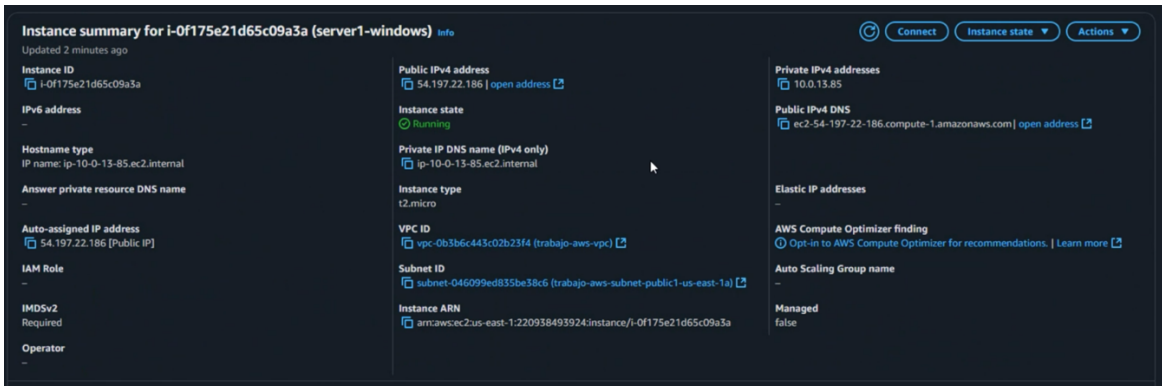
1. Abre el navegador y vamos a ruta: https://console.aws.amazon.com/
2. Iniciamos sesión con la cuenta de AWS (previamente creada).

Según lo estudiado AWS ofrece la infraestructura como servicio (IaaS), lo que nos permite aprovisionar servidores virtuales bajo demanda sin comprar hardware físico.

Lanzar una instancia EC2

1. Nos dirigimos a Servicios > EC2 > Instances > Launch Instance.
2. Se asigna un nombre identificador, por ejemplo: Windows-Server.
3. Seleccionamos la AMI:
    - Microsoft Windows Server 2019 Base o 2022 (capa gartuita)
    - Estas imágenes contienen el sistema operativo listo para usarse.
4. Tipo de instancia: t2.micro (capa gratuita).

Figura: 3

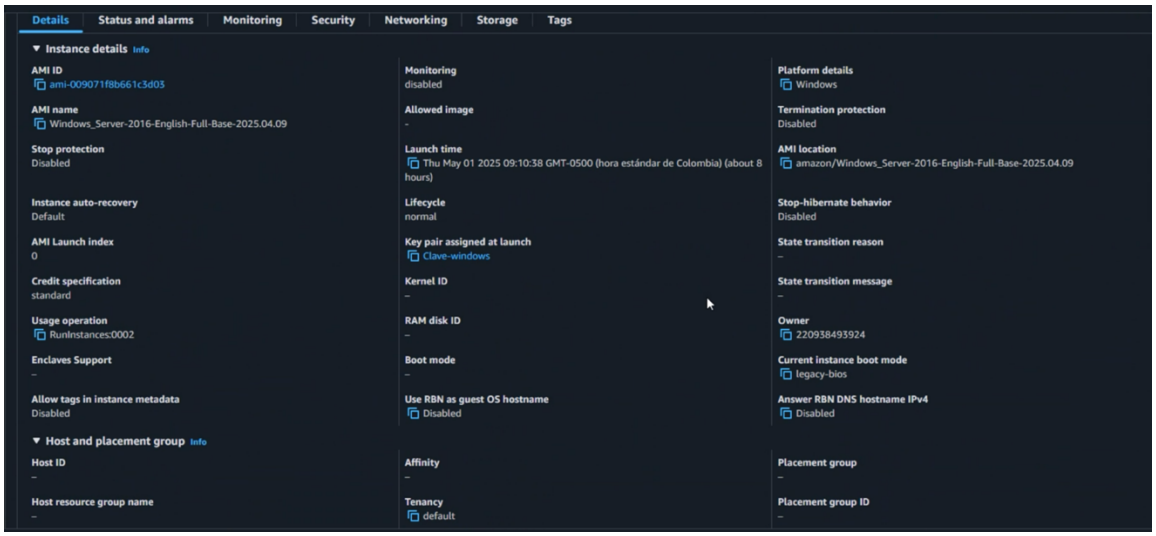


Fuentes: propia

5. Claves de acceso:
    - Usamos una llave existente o creamos una nueva la cual debemos descargar al crear lo (archivo .pem).
    - Este archivo es esencial para descifrar la contraseña RDP del usuario administrador.

EC2 (Elastic Compute Cloud) permite crear máquinas virtuales de forma rápida, facilitando pruebas, desarrollo y producción.

Figura: 4

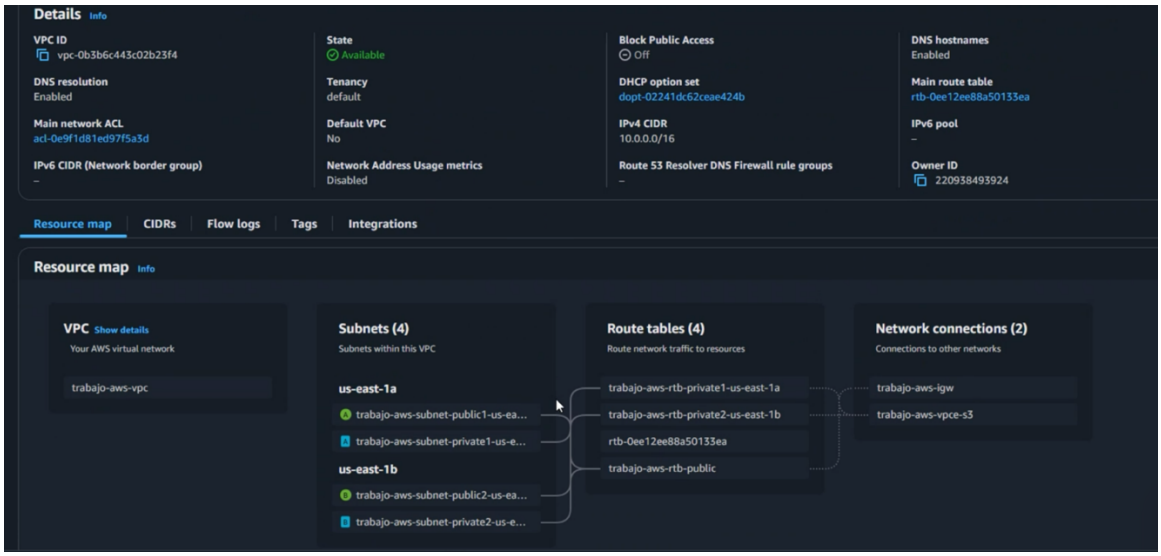


Fuentes: propia

Configuración de red y seguridad

1. VPC: seleccionamos las predeterminadas o creamos unas nuevas.
2. Subred: elegimos una pública.
3. Auto-assign Public IP: habilitado (necesario para conexión externa).

Figura: 5

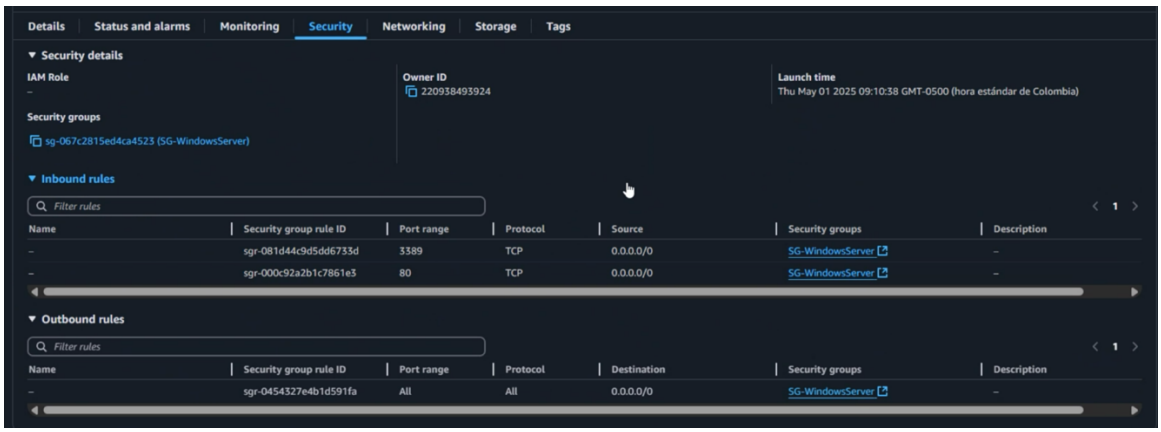


Fuentes: propia

Security Group (Grupo de Seguridad):

- Regla 1: RDP (puerto 3389) → tu IP actual.
- Regla 2: HTTP (puerto 80) → 0.0.0.0/0 (permite tráfico web público).

Figura: 6



Fuentes: propia

Los grupos de seguridad en AWS actúan como firewalls virtuales. Definir puertos y origen de tráfico reduce vulnerabilidades haciendo que el sistema se más seguro.

Acceso remoto a la instancia Windows

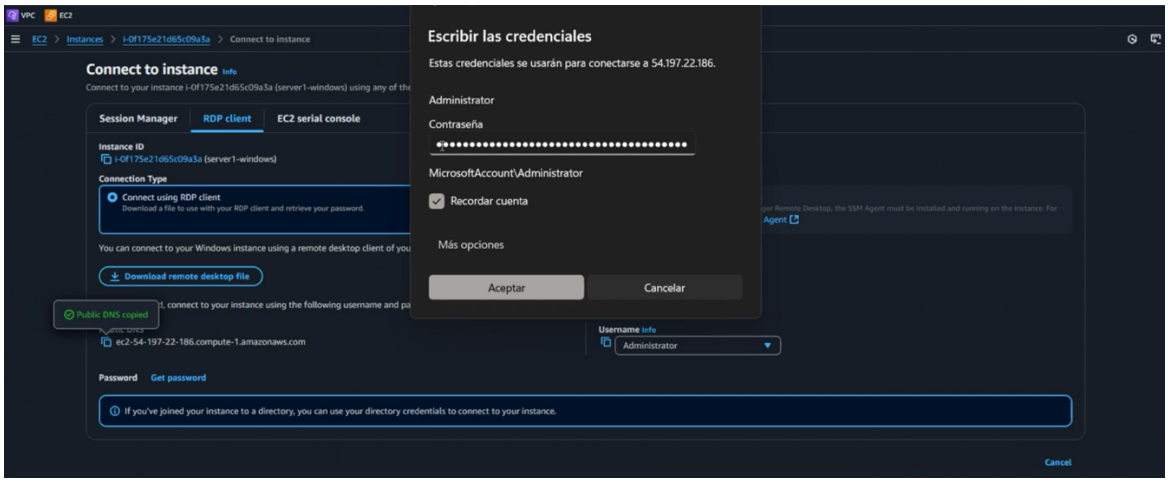
⇒ Obtener las credenciales

1. Esperamos a que la instancia esté en estado running.
2. Selecciónanos y damos clic en Connect > RDP Client.
3. Subimos el archivo .pem y recuperamos la contraseña del usuario administrador.
4. Tener presente la IP pública.

⇒ Conexión vía RDP

1. Abrimos la aplicación Escritorio Remoto de Windows (RDP).

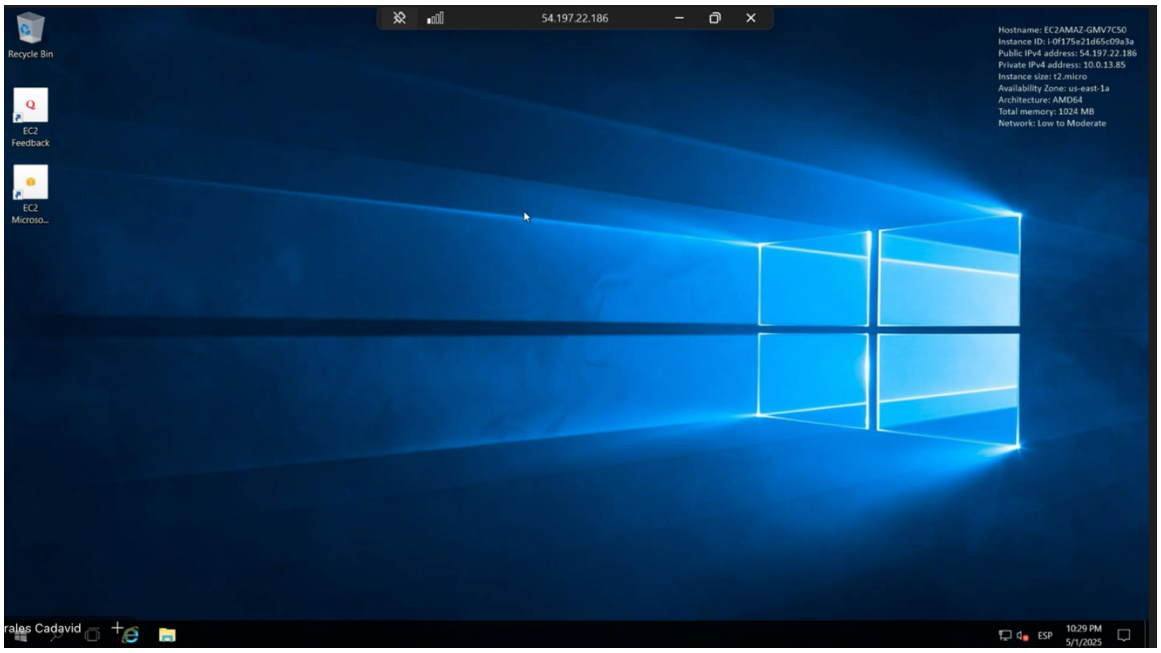
Figura: 7



Fuentes: propia

2. Introducimos la IP pública.
3. Usuario: Administrador
4. Contraseña: la recupera con el .pem.

Figura: 8



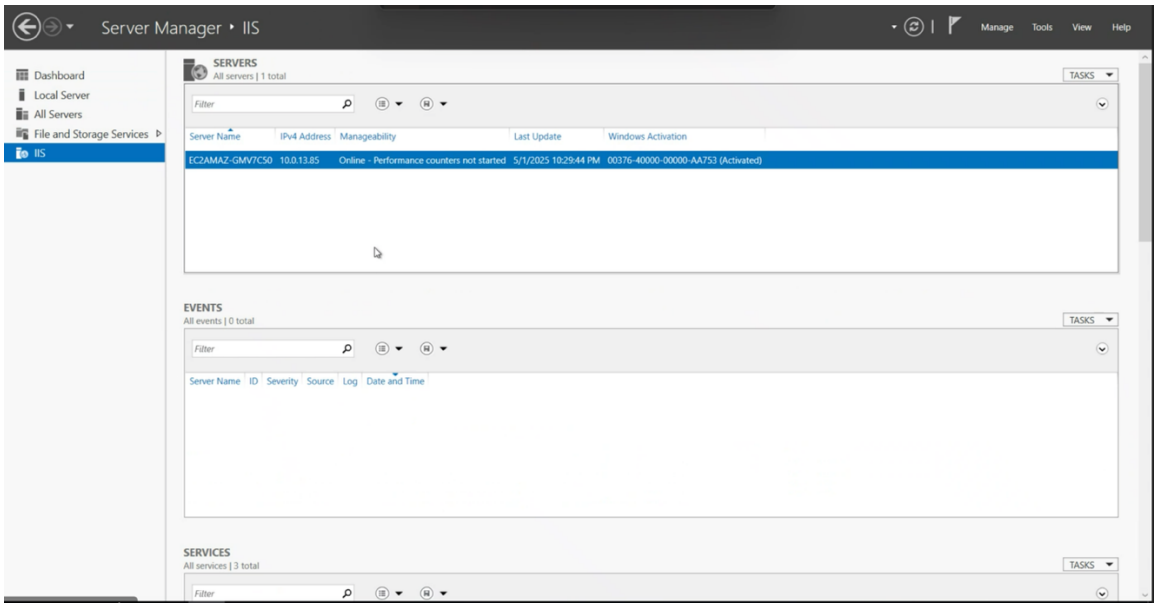
Fuentes: propia

El protocolo RDP permite la gestión remota de sistemas operativos Windows mediante una interfaz gráfica, simulando una experiencia de uso local aun cuando el servidor se encuentre en la nube.

Instalación de IIS (Servidor Web de Windows)

1. En el servidor, abrimos Server Manager.
2. Vamos a Manage > Add Roles and Features.
3. Avanzamos por el asistente hasta Roles y seleccionamos:
    - Web Server (IIS)
4. Completamos la instalación.

Figura: 9

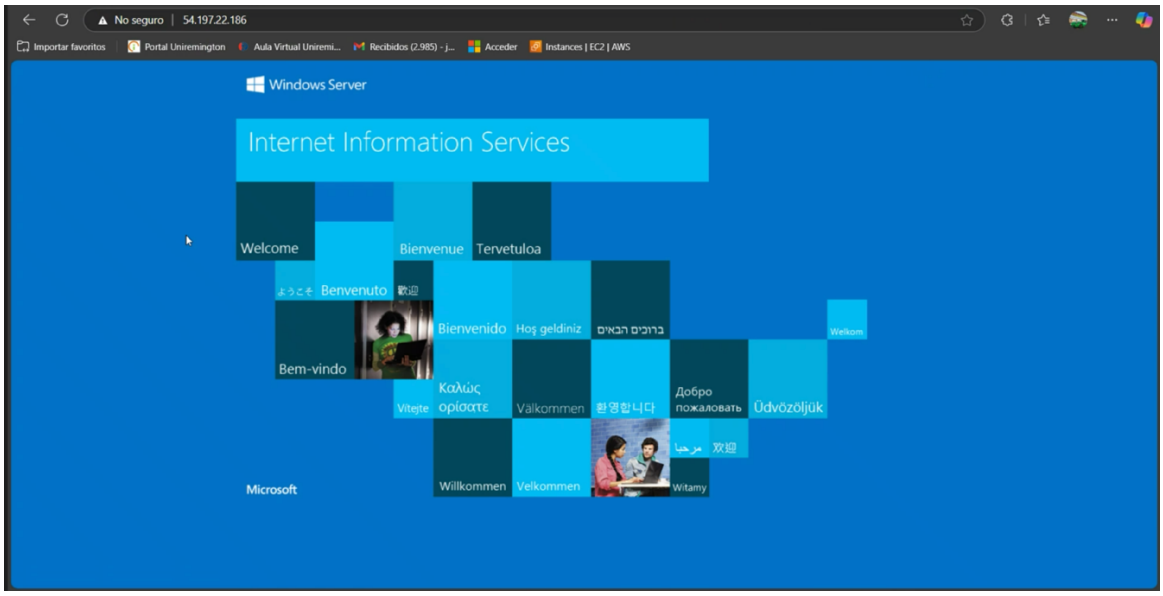


Fuentes: propia

Comprobación:

- Abrimos el navegador y visitamos la página:
- http://54.197.22.186/
- Nos derivara a la página por defecto de IIS.

Figura: 10

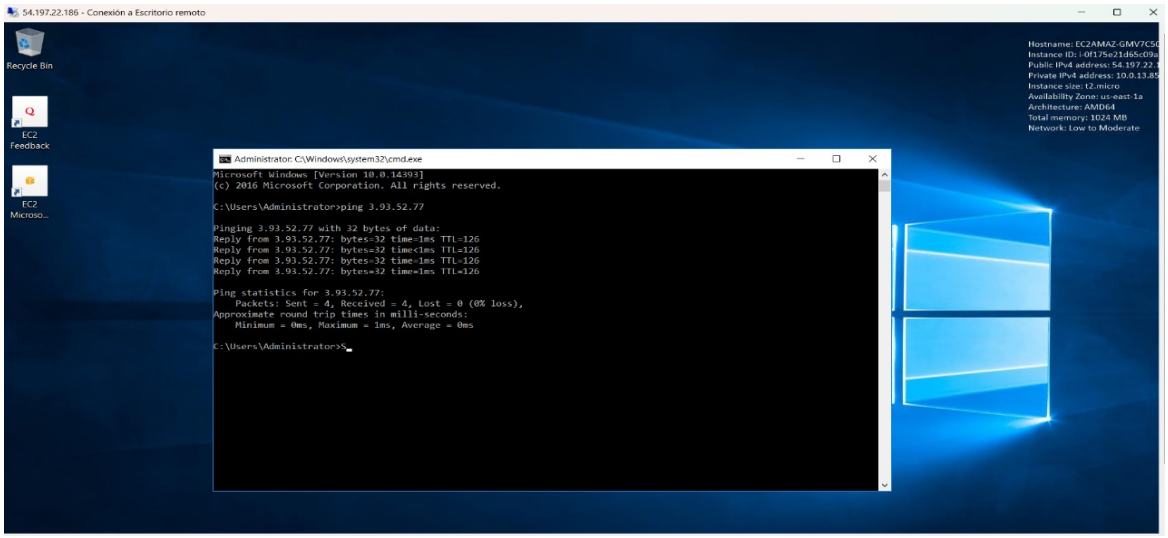


Fuentes: propia

Internet Information Services (IIS) es el servidor web por defecto en sistemas Windows. Su instalación en una instancia EC2 permite ofrecer contenidos y servicios web accesibles públicamente.

Ping desde Servidor Windows al servidor Linux:

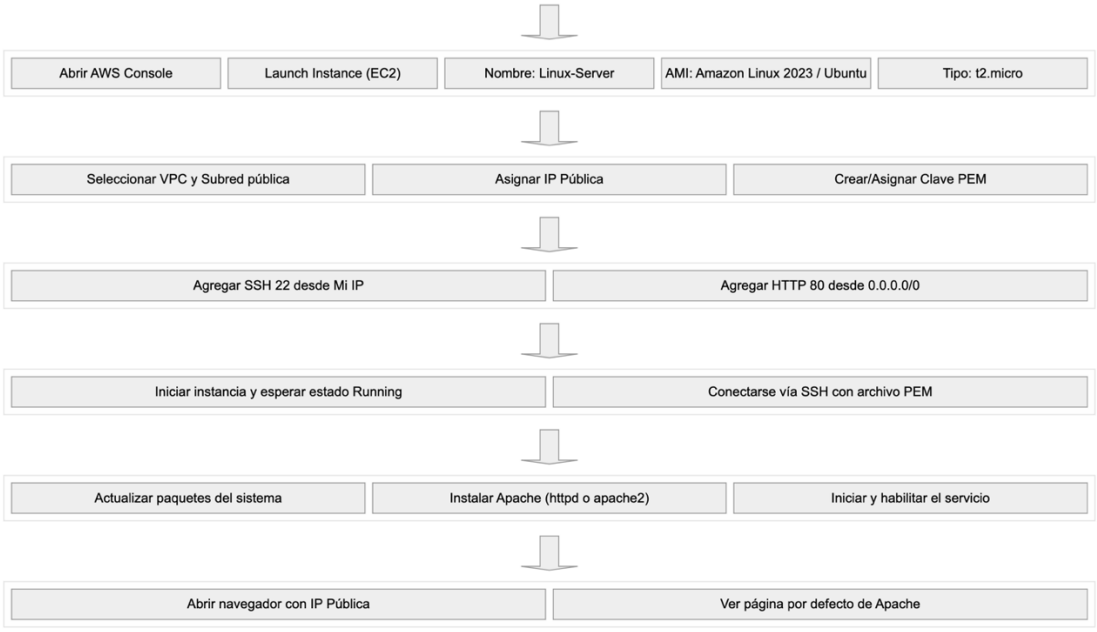
Figura: 11



Fuentes: propia

1.2. Implementación de Linux Server en AWS

Figura: 12



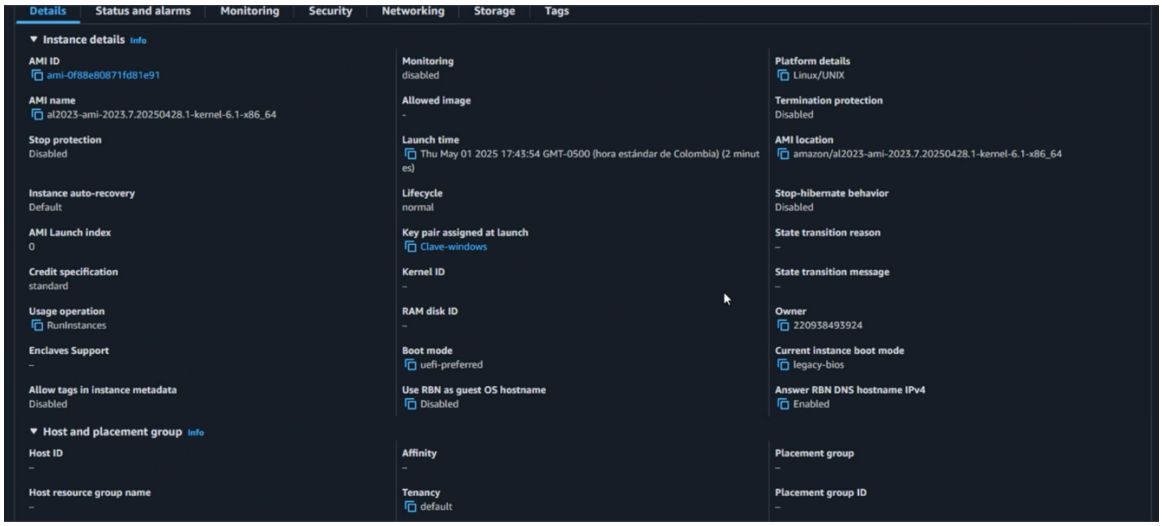
Fuentes: propia

Creamos una instancia EC2 con Amazon Linux

1. Ingresamos a: https://console.aws.amazon.com/
2. Vamos a EC2 > Instances > Launch Instance
3. Configura:
    - Nombre: Linux-Server
    - AMI: Amazon Linux 2023
    - Tipo de instancia: t2.micro (capa gratuita)
    - Claves de acceso: Usamos una existente o creamos una nueva

(descargamos el .pem)

Figura: 13



Fuentes: propia

Figura: 14

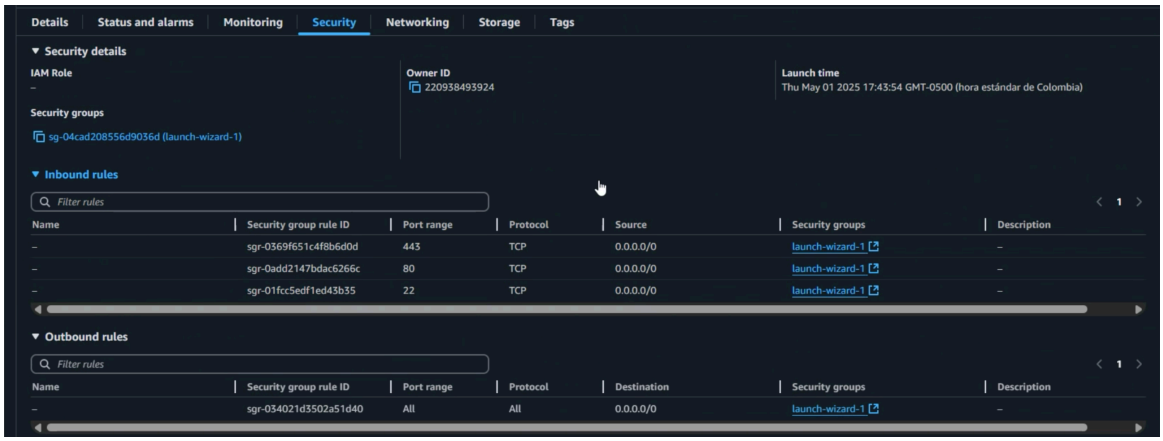
Fuentes: propia

Amazon Linux es una distribución optimizada para funcionar de forma eficiente y segura dentro del entorno de AWS.

Configuración de red y seguridad

1. VPC: predeterminada o personalizada
2. Subred: pública
3. Auto-assign Public IP: habilitado
4. Security Group:
    - Regla 1: SSH (puerto 22) → tu IP
    - Regla 2: HTTP (puerto 80) → 0.0.0.0/0 (acceso público a sitios web)

Figura: 15



Fuentes: propia

SSH es el protocolo estándar para administrar servidores Linux de forma remota y segura.

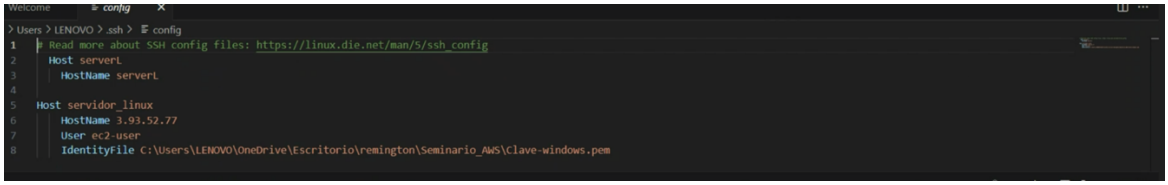
Conectarse vía SSH

Utilizaremos la extensión de Remote - SSH en Visual Studio Code.

1. Esperamos a que el estado de la instancia sea running

2. Copiamos la IP pública y realizamos la configuración de la conexión
3. Nos conéctanos con:
    1. clave-windows.pem
    2. ec2-user
    3. 3.93.52.77

Figura: 16



Fuentes: propia

El usuario por defecto suele ser ec2-user para Amazon Linux, o Ubuntu si usas Ubuntu Server aun que es esta edición no hablaremos de Ubuntu las características en parte son similares.

Al usar SSH podemos interactuar con el sistema Linux para configuraciones, despliegues, instalación de servicios, etc.

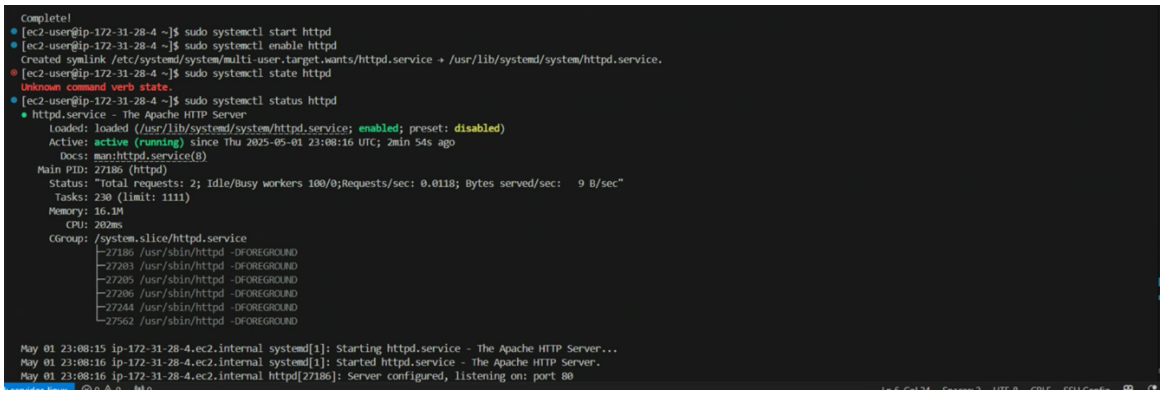
Instalar Apache (servidor web)

Una vez conectado por SSH:

- `sudo yum update -y` (Amazon Linux)
- `sudo yum install httpd -y` (En Apache HTTP Server)
- `sudo systemctl start httpd`

- `sudo systemctl enable httpd`

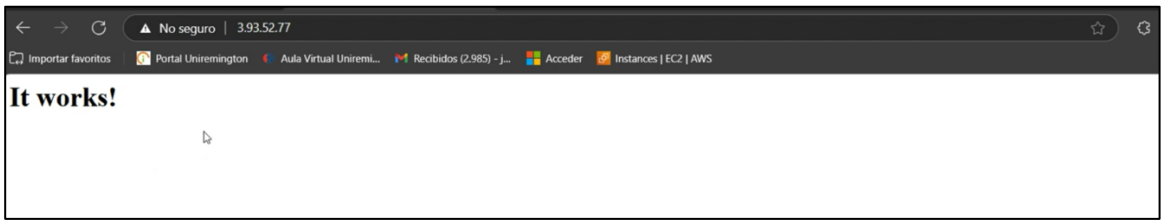
Figura: 17



Fuentes: propia

Verifica accediendo desde el navegador a:

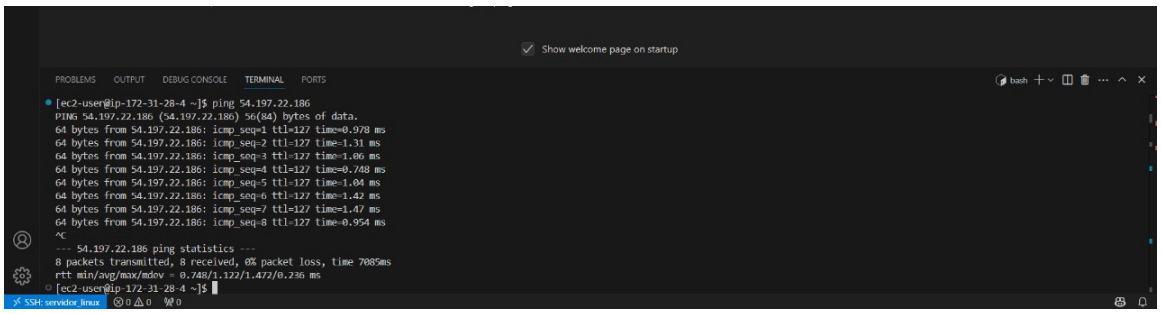
Figura: 18



Fuentes: propia

Ping desde servidor Linux al servidor Windows:

Figura: 19



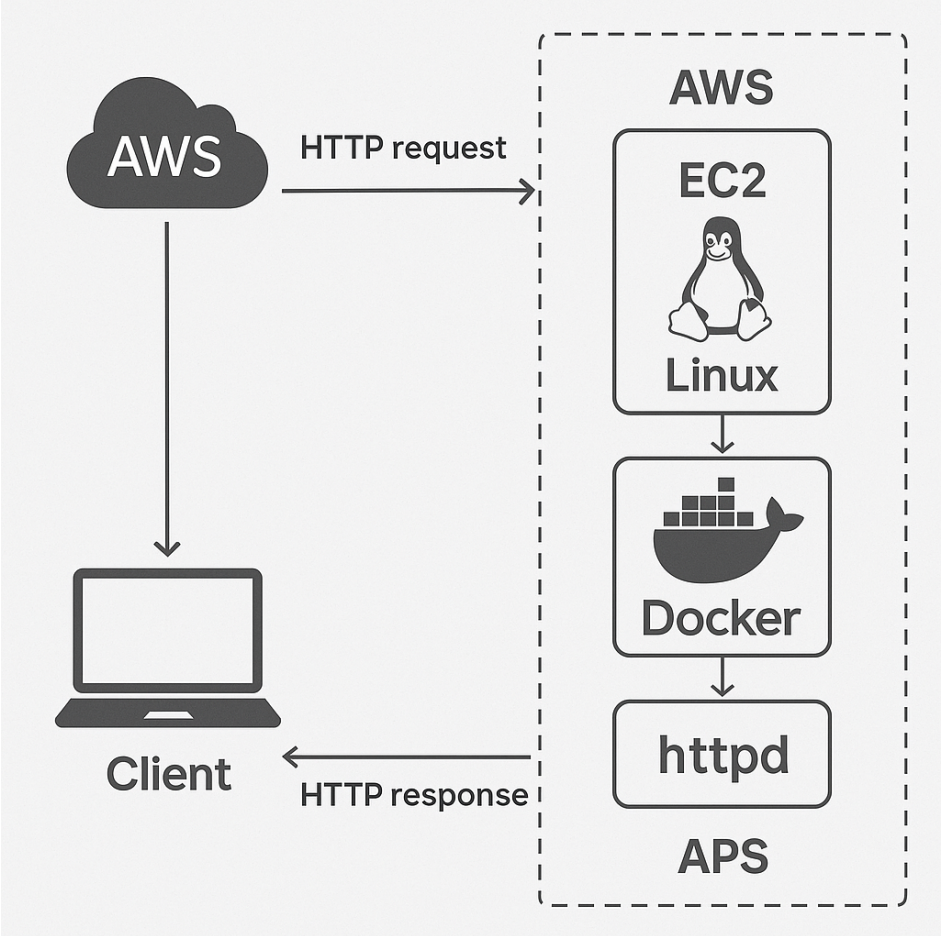
Fuentes: propia

1.3. Implementación de Docker en servidores Linux

Después de finalizar correctamente todos los pasos previos para la configuración de un servidor Linux, y una vez logramos conectarnos a través de SSH, ya sea utilizando una terminal local o desde la consola web de AWS, el siguiente paso será proceder con la instalación de Docker en dicho servidor.

Diagrama arquitectónico.

Figura: 20



Fuente: propia

1. Cliente realiza una solicitud HTTP
    - Desde el navegador realiza una solicitud HTTP a una IP pública para nuestro caso de ejemplo, sin embargo, es común que sea directamente a través de un nombre de dominio.

2. Solicitud llega a AWS EC2
    - La solicitud es dirigida a una instancia EC2 (servidor virtual en AWS) con sistema operativo Linux en cual fue previamente configurado para recibir en el puerto indicado la petición realizada.

3. Instancia EC2 con Docker instalado
    - Dentro de esta instancia se ha instalado Docker, permitiendo ejecutar contenedores de forma aislada.

4. Docker ejecuta un contenedor con httpd
    - Se levanta un contenedor Docker que ejecuta el servidor web Apache httpd

5. httpd sirve contenido web
    - Apache dentro del contenedor responde a la solicitud HTTP con el contenido web (HTML, imágenes, entre otros).

6. Respuesta HTTP es enviada al cliente
    - El flujo se completa cuando el contenedor devuelve una respuesta al cliente que originó la solicitud.

A esto podemos añadir que es posible que varios contenedores puedan estar interactuando con un mismo contenido web, ya que se puede conectar una Aplicación o página Web a varios contenedores ingresando por cualquiera de los puertos seleccionados para cada uno de ellos.

Proceso de instalación Docker

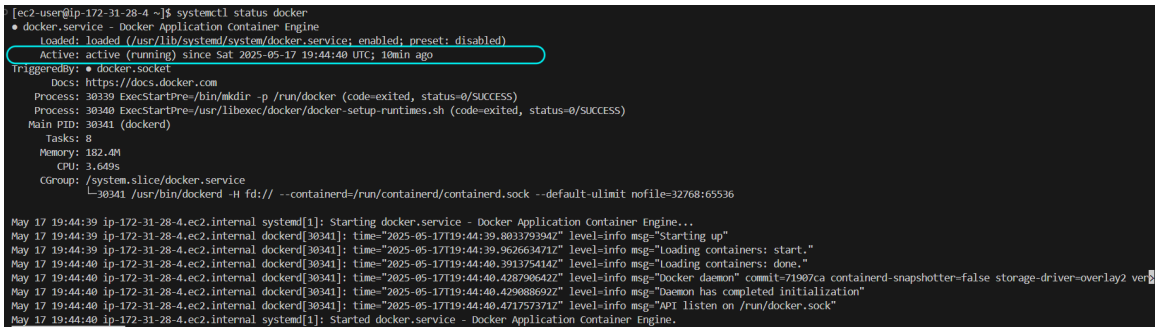
1. Configuración de entorno

1.1. Una vez estamos dentro del servidor lo primero que debemos realizar es ingresar con los permisos de administrador para ello podemos usar el comando `sudo su`, este nos dará todos los permisos necesarios para continuar con los siguientes pasos.

1.2. Verificar que la maquina tenga accedo desde internet para poder realizar las descargas necesarias.

1.3. Realizamos la descarga de Docker desde los repositorios de AWS usando el comando `dnf install docker` luego verificamos el estado, con el comando `systemctl status docker` de no estar activo lo debemos activar con el comando `systemctl start docker` debe de verse algo similar a la figura No. 21.

Figura: 21

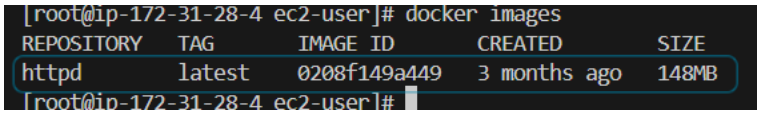


Fuente: propias

1.4. Una vez estemos seguros de que este activo Docker descargamos una imagen del servidor Web de Apache HTTPD utilizando el comando `docker pull httpd`.

1.4.1. Validamos si la descarga fue exitosa con el comando `docker images`, este nos mostrara la información de la imagen como el Nombre, id entre otros.

Figura: 22



Fuente: propia

1.5. Teniendo este segmento configurado podemos instalar los contenedores necesarios para la ejecución de nuestra Aplicación o Página Web.

1.5.1. La cantidad de contenedores que se pueden crear depende de la capacidad física de la máquina virtual en la que se corren los contenedores.

1.5.2. Procedemos a crear un contenedor para ello utilizamos el comando `docker run -dat --name xxxxx -p 8080:80 httpd`

Descripción de los parámetros del comando:

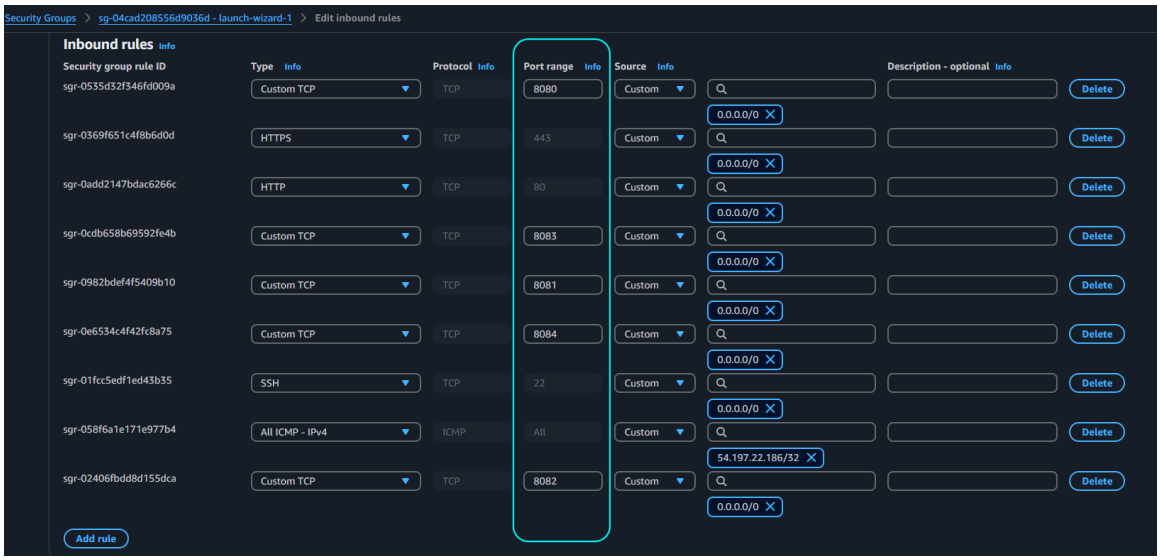
1. `docker run`, este es el comando base que se usa para crear y ejecutar un contenedor Docker.

2. `-dit`, ejecuta el contenedor en segundo plano (-d), conservando una interfaz de terminal interactiva (-it).

3. `--name xxxxx`, Le asigna un nombre personalizado al contenedor: cont2. Esto facilita su identificación en lugar de usar el ID generado por Docker.

4. `-p 8080:80`, expone el puerto 80 del contenedor o el seleccionado (donde escucha el servidor web Apache) al puerto 8080 del host.

Figura: 23



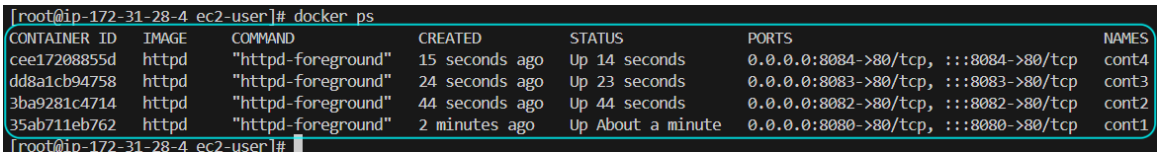
Fuente: Propia

5. `httpd`, es la imagen que se usará para crear el contenedor del servidor web Apache HTTP Server.

Como resultado del comando nos debe crear un contenedor con un nombre, un puerto, ejecutándose en segundo plano sobre la imagen `httpd` de Apache.

En este ejemplo hemos creado 4 contenedores, ingresamos el comando `docker ps` para poder visualizarlas.

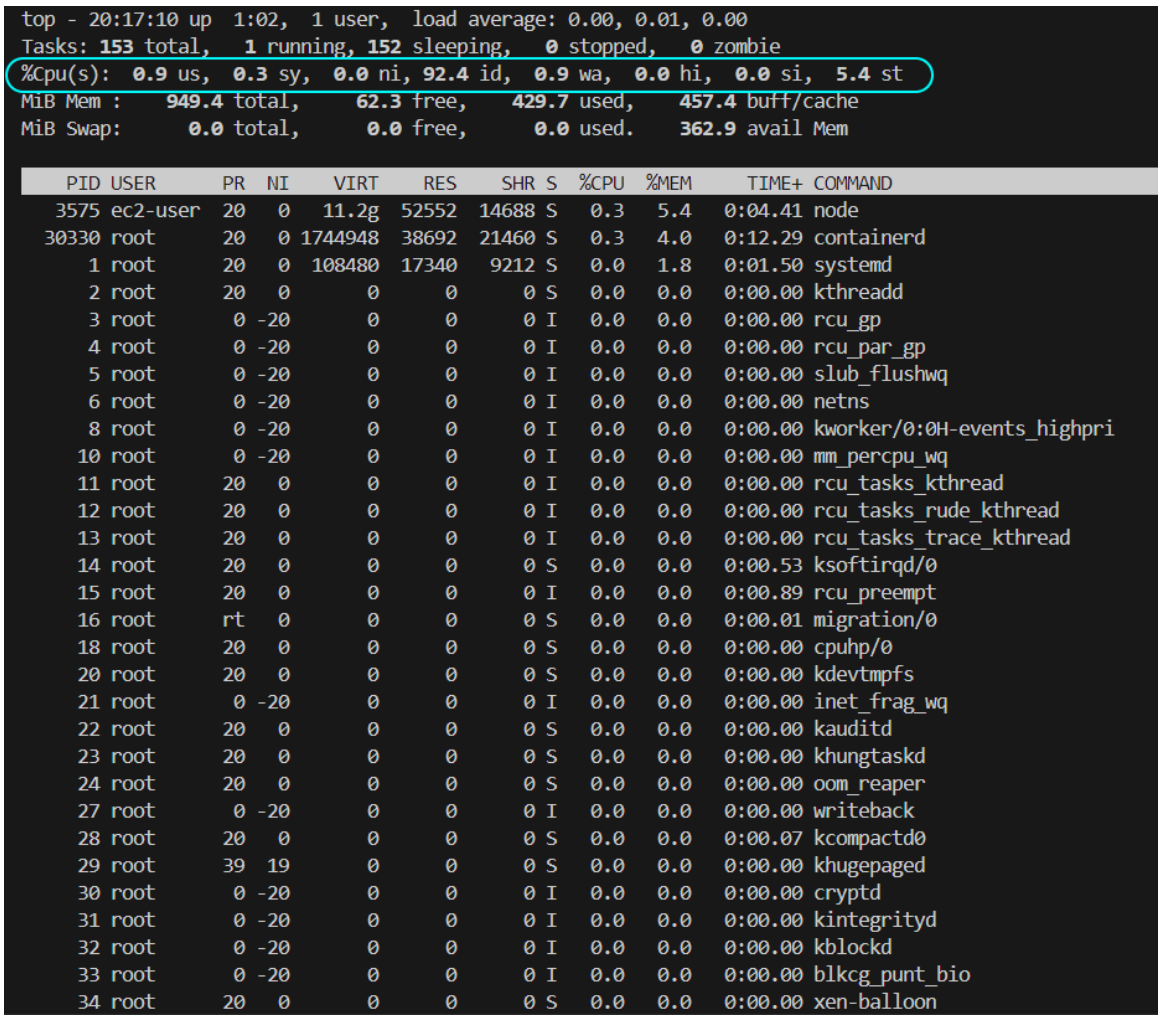
Figura: 24



Fuente: Propia

Inicialmente la CPU de nuestra máquina virtual no tiene muchos cambios en el consumo de recurso

Figura: 25



Fuente: propia

Esto nos garantiza que el contenedor esta activo y con la configuración del puerto en el grupo de seguridad ya podemos acceder desde internet a los contenedores, para este ejemplo descargamos un sitio web estático para poder accederlo desde internet.

Obtuvimos una plantilla HTML desde internet y lo colocamos dentro de una carpeta para ser usada por el contenedor en el directorio raíz del servidor web (`/usr/local/apache2/htdocs`).

1. Descargar la plantilla desde HTML5 UP

Utiliza wget para descargar el archivo ZIP de la plantilla:

```
wget https://html5up.net/story/download -O story.zip
```

2. Crear una carpeta para alojar el sitio web

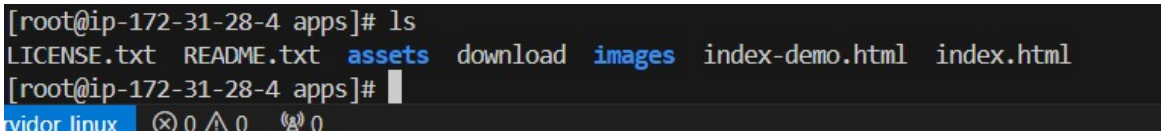
```
mkdir apps
cd apps
```

3. Descomprimir el archivo descargado en la carpeta

```
unzip ../story.zip
```

Esto descomprime el contenido dentro de apps.

Figura:26



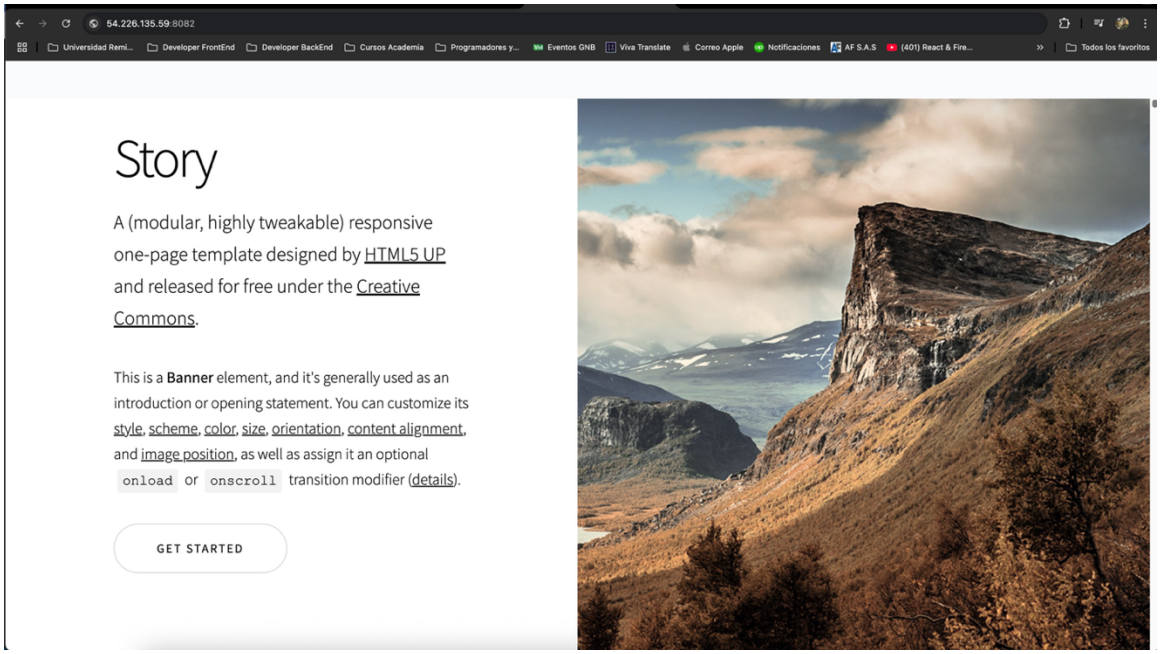
Fuente: propia

Primer ejercicio práctico

Creamos dos contenedores cada uno con una aplicación diferente con puertos:

1. El primer contenedor por el puerto 8082

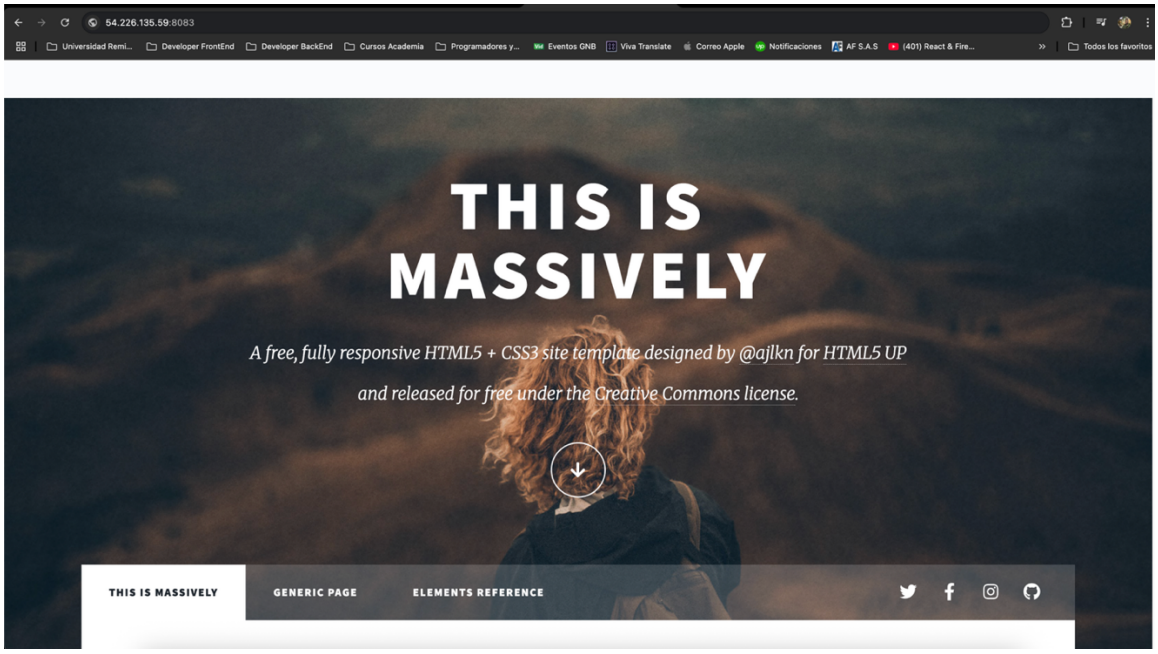
Figura: 27



Fuente: propia

2. El segundo contenedor por el puerto 8083

Fuente: 28



Fuente: propia

Estamos sirviendo desde dos contenedores Docker diferentes. Cada uno de estos Contenedores exponen el puerto interno 80 (por donde se sirve la web), pero lo enlazamos con diferentes puertos del servidor para poder accederlos externamente.

- Cont2: 8082:80
- Cont3: 8083:80

Esto significa que, si accedo a la dirección pública del servidor en el puerto 8082, estamos viendo la aplicación del primer contenedor. Si lo hacemos en el puerto 8083, veré la del segundo, y así sucesivamente.

Además, se configuro el grupo de seguridad (en el panel de AWS para permitir el tráfico desde internet a los puertos 8082 y 8083, lo que garantiza que los contenedores sean accesibles desde cualquier navegador.

Segundo ejercicio pactico

Ejecutar múltiples contenedores Docker con la misma aplicación web. En este ejercicio, usaremos una misma página web estática para ejecutarla en cuatro contenedores Docker, cada uno accesible desde un puerto diferente.

Verificar el estado del contenedor: antes de ejecutar un contenedor, asegúrate de que no exista previamente con el mismo nombre. Si ya existe, elimínalo para evitar errores: `docker rm -f cont1 cont2 cont3 cont4 2>/dev/null`

1. Navegar al directorio del sitio web: nos ubicamos en el directorio donde se encuentra la aplicación web: `cd /home/ec2-user/apps/`
2. Ejecutar los contenedores: a continuación, creamos cuatro contenedores Docker con Apache (httpd), cada uno vinculado al mismo sitio web, pero expuesto por diferentes puertos del host:

⇒ Contenedor 1

```
docker run -dit --name cont1 -p 8080:80 -v
/home/ec2-user/apps/:/usr/local/apache2/htdocs httpd
```

⇒ Contenedor 2

```
docker run -dit --name cont2 -p 8082:80 -v
/home/ec2-user/apps/:/usr/local/apache2/htdocs httpd
```

⇒ Contenedor 3

```
docker run -dit --name cont3 -p 8083:80 -v
/home/ec2-user/apps/:/usr/local/apache2/htdocs httpd
```

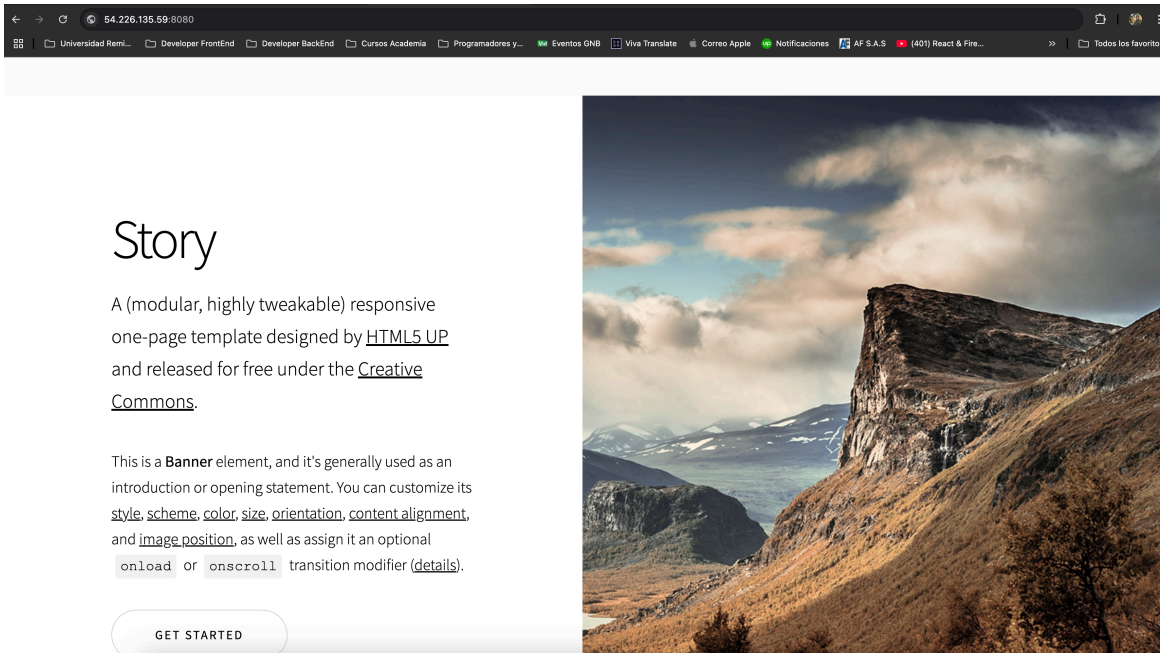
⇒ Contenedor 4

```
docker run -dit --name cont4 -p 8084:80 -v
/home/ec2-user/apps/:/usr/local/apache2/htdocs httpd
```

En este paso ya es posible acceder al sitio web desde 4 puntos diferentes de entrada y puertos en cualquier navegador:

- http://54.226.135.59:8080/

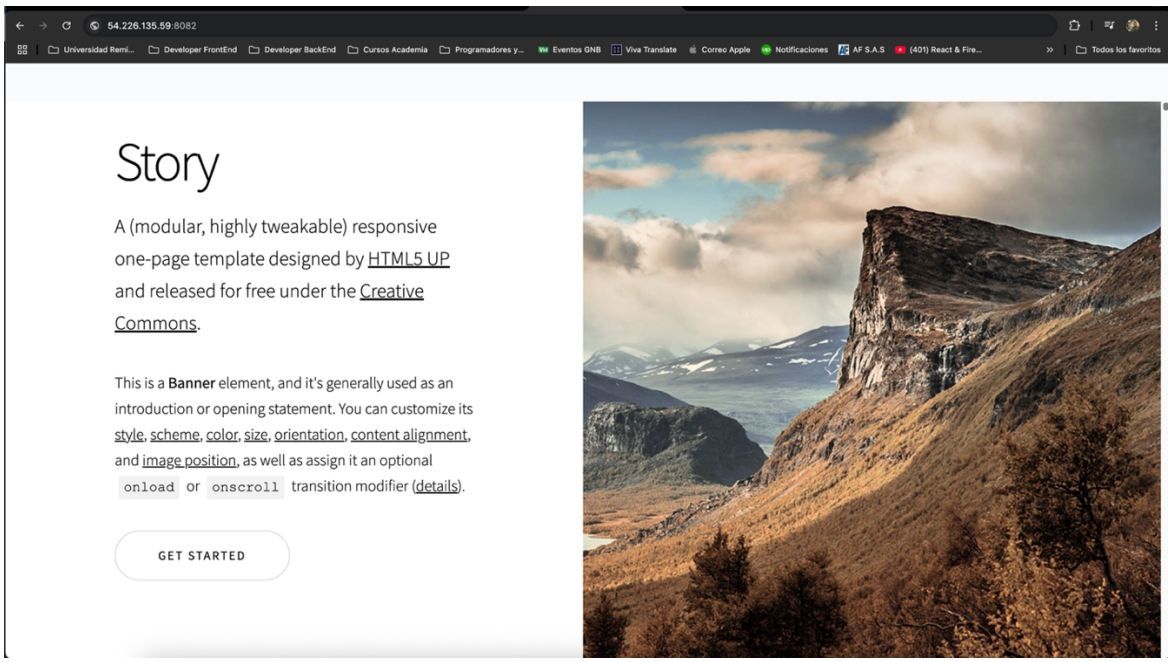
Figura: 29



Fuente: propia

- http://54.226.135.59:8082/

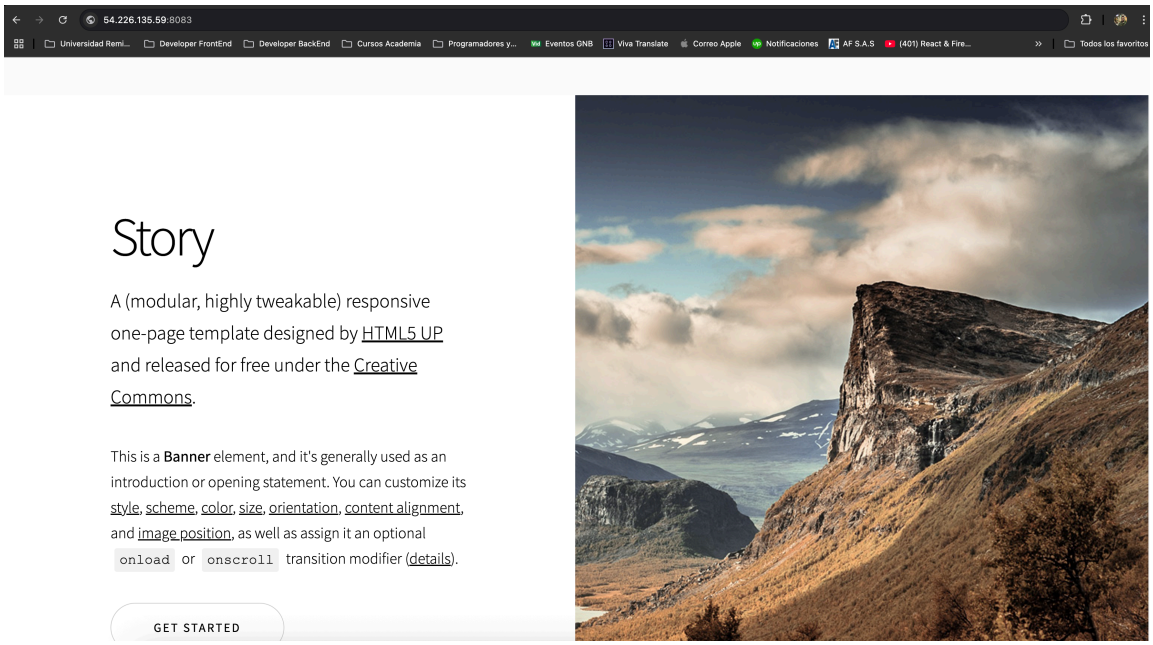
Figura: 30



Fuente: propia

- http://54.226.135.59:8083/

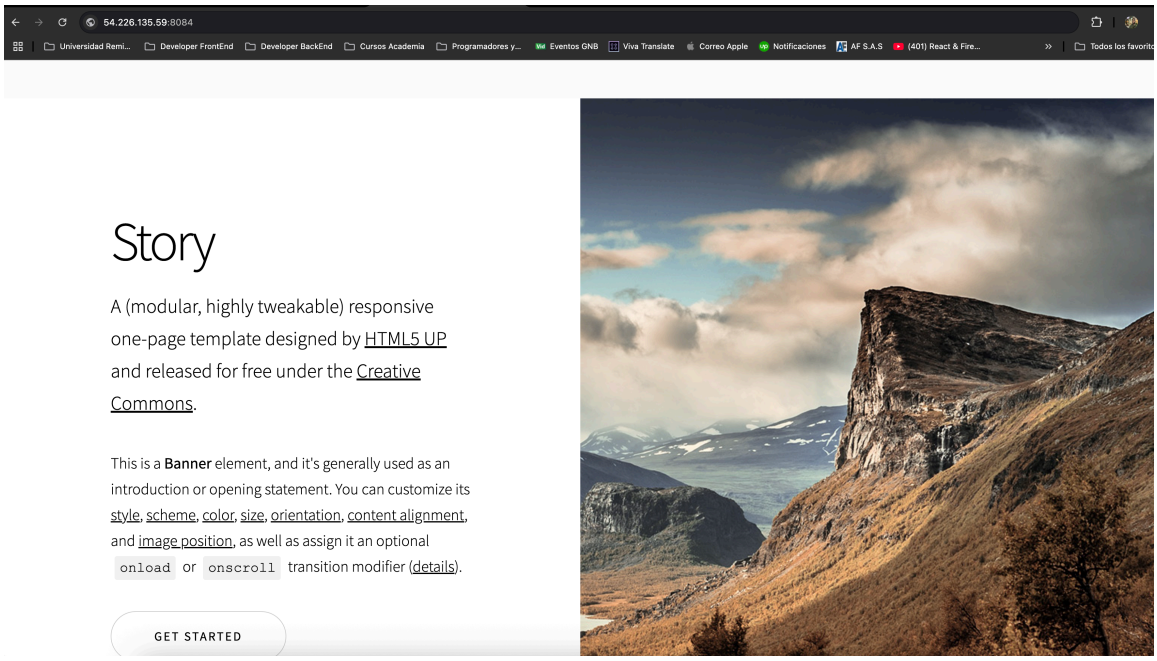
Figura: 31



Fuente: propia

- http://54.226.135.59:8084/

Figura: 32



Fuente: propia

¿Qué permite esto?

Simular un escenario de escalabilidad horizontal, en el que la misma aplicación web se ejecuta en múltiples contenedores de forma simultánea. Aunque en este caso accedemos a cada contenedor desde diferentes puertos del host (por ejemplo: 8080, 8082, 8083, 8084), el propósito es visualizar cómo la aplicación puede ser replicada en distintas instancias, de forma similar a un balanceador de carga, como lo hace AWS Elastic Load Balancing, aunque de forma manual y básica.

Prueba de estrés Saturar el CPU intencionalmente.

Para saturar la máquina virtual vamos a poner en marcha varios contenedores con el fin de generar una alta carga en el uso del CPU.

Para ello creamos un script en Python con las siguientes características:

Generadores automáticos con bucles infinitos en Python

1. Crea un Dockerfile para CPU burner

- `mkdir cpu_test`
- `cd cpu_test`

2. Dentro de Dockerfile agregamos el siguiente código:

- `FROM python:3.10-slim`
- `CMD ["python", "-c", "while True: pass"]`

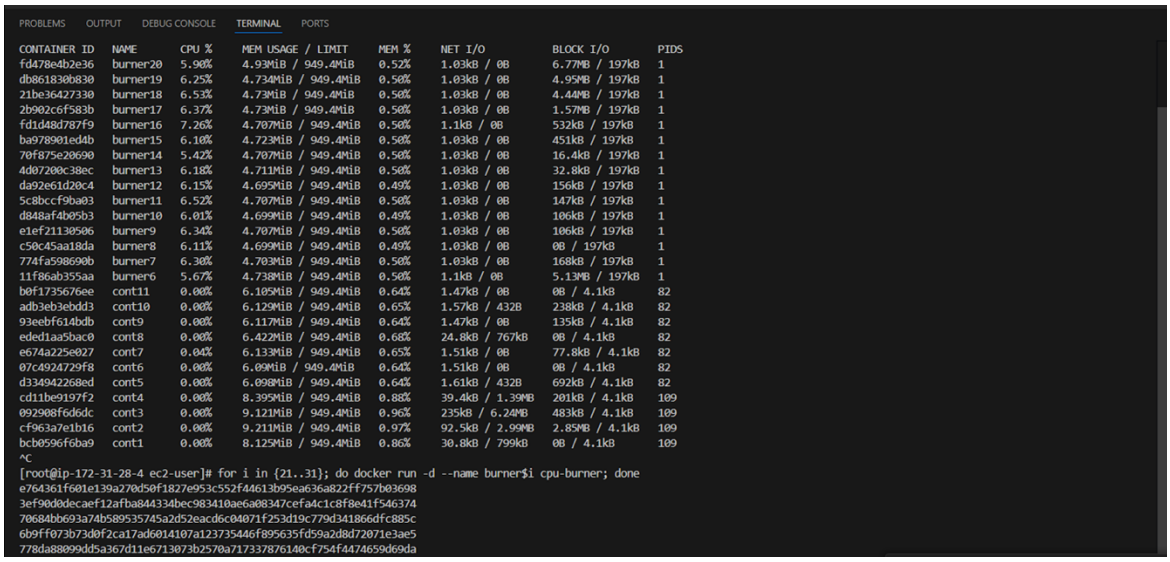
3. Luego construimos la imagen

- `docker build -t cpu-burner`

4. En este último paso lanza múltiples contenedores que saturen la CPU

- `for i in {1..5}; do docker run -d --name burner$i cpu-burner; done`

Figura: 33

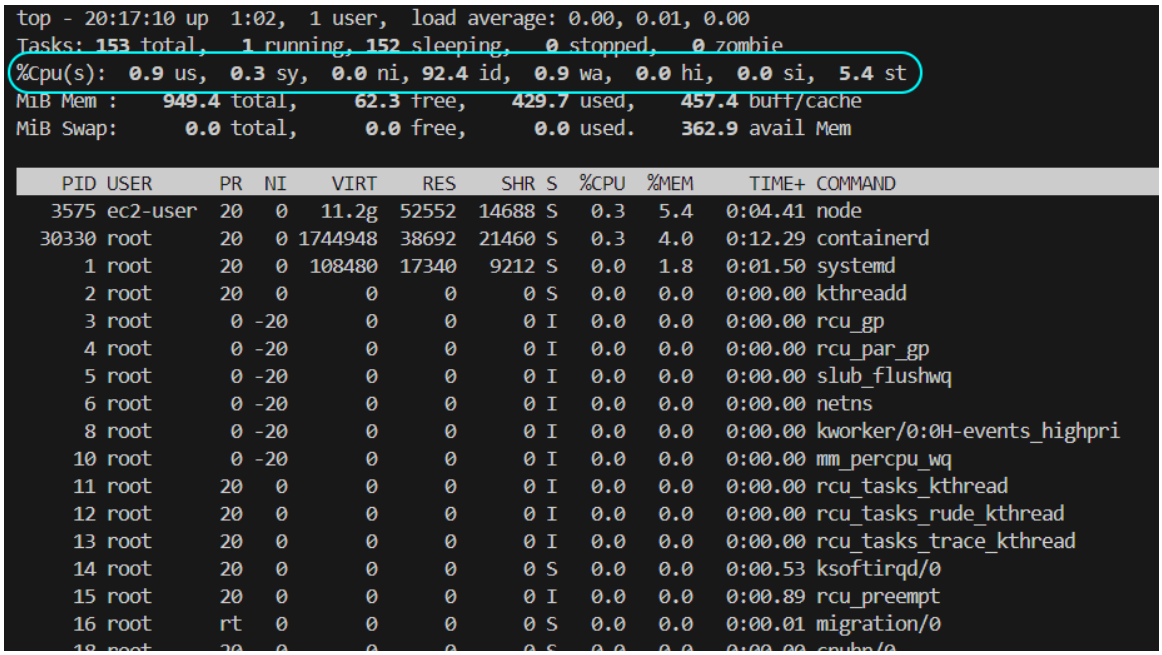


Fuente: propia

Para la primera prueba empezamos con 5 contendedores.

Al revisar las condiciones de la maquina con el comando `top y docker stats` no identificamos un consumo significativo de CPU como se muestra en la figura No. 34.

Figura: 34

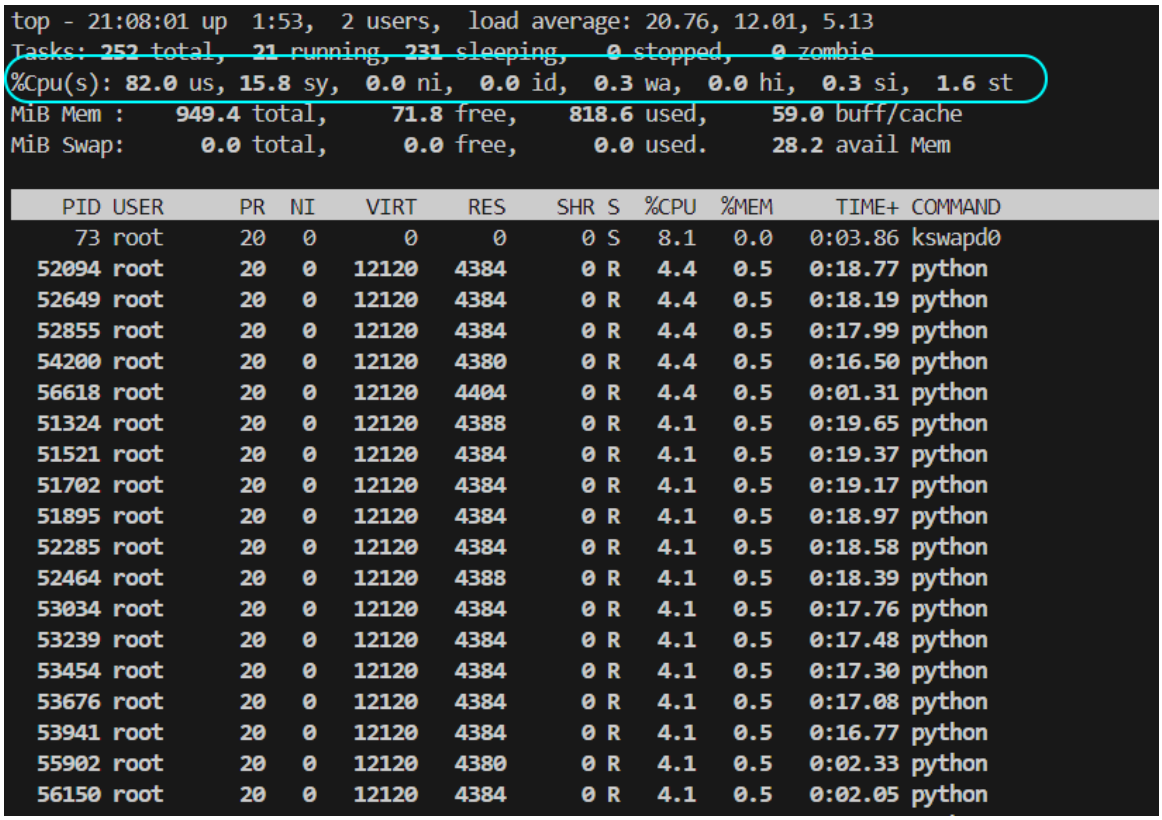


Fuente: propia

Para la segunda prueba aumentamos a 20 contenedores lanzando el comando `for i in {6..20}; do docker run -d --name burner$i cpu-burner; done`

Al volver a revisar las condiciones de la maquina con el comando nos encontramos un consumo significativo de CPU de 82% como se muestra en la figura No. 35

Figura: 35



Fuente: propia

En este punto la página estática funciona sin problemas, sin embargo, se disminuyó la rapidez de la respuesta.

Para la tercera prueba aumentamos a 30 contenedores lanzando el comando `for i in {21..30}; do docker run -d --name burner$i cpu-burner; done`

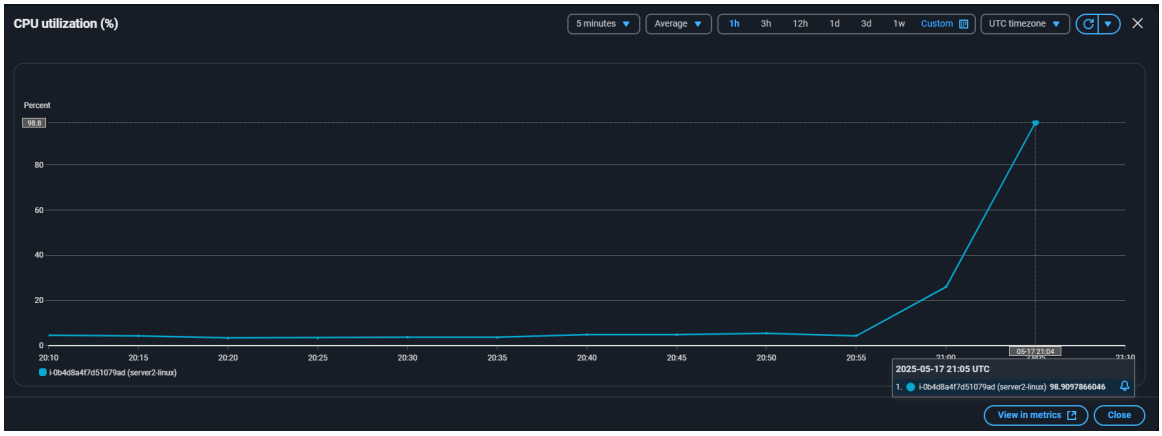
Inicialmente pudimos evidenciar un consumo de 97% de la CPU como se muestra en la figura No. 36 y en la figura No. 37 de un 98%, sin embargo, en este punto ya la instancia se cayó por ende la página web que se conectaba a través de los 4 contenedores

Figura: 36

```
top - 21:04:55 up  1:50,  2 users,  load average: 13.20, 5.03, 1.84
Tasks: 237 total,  16 running, 221 sleeping,   0 stopped,   0 zombie
%Cpu(s): 97.7 us,  0.3 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  2.0 st
MiB Mem :    949.4 total,     65.5 free,    712.3 used,    171.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.     84.2 avail Mem
CONTAINER ID   NAME       CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
fd478e4b2e36   burner20   6.73%     5.051MiB / 949.4MiB   0.53%     1.03kB / 0B       6.77MB / 197kB    1
db861830b830   burner19   6.69%     4.754MiB / 949.4MiB   0.50%     1.03kB / 0B       4.95MB / 197kB    1
21be36427330   burner18   6.25%     4.73MiB / 949.4MiB    0.50%     1.03kB / 0B       4.44MB / 197kB    1
2b902c6f583b   burner17   6.18%     4.734MiB / 949.4MiB   0.50%     1.03kB / 0B       1.57MB / 197kB    1
fd1d48d787f9   burner16   6.11%     4.707MiB / 949.4MiB   0.50%     1.03kB / 0B       532kB / 197kB     1
ba978901ed4b   burner15   5.93%     4.723MiB / 949.4MiB   0.50%     1.03kB / 0B       451kB / 197kB     1
70f875e20690   burner14   6.89%     4.707MiB / 949.4MiB   0.50%     1.03kB / 0B       16.4kB / 197kB    1
4d07200c38ec   burner13   6.22%     4.711MiB / 949.4MiB   0.50%     1.03kB / 0B       32.8kB / 197kB    1
da92e61d20c4   burner12   6.04%     4.695MiB / 949.4MiB   0.49%     1.03kB / 0B       156kB / 197kB     1
5c8bccf9ba03   burner11   6.27%     4.707MiB / 949.4MiB   0.50%     1.03kB / 0B       147kB / 197kB     1
d848af4b05b3   burner10   6.06%     4.699MiB / 949.4MiB   0.49%     1.03kB / 0B       106kB / 197kB     1
e1ef21130506   burner9    6.37%     4.707MiB / 949.4MiB   0.50%     1.03kB / 0B       106kB / 197kB     1
c50c45aa18da   burner8    5.74%     4.699MiB / 949.4MiB   0.49%     1.03kB / 0B       0B / 197kB        1
774fa598690b   burner7    6.50%     4.703MiB / 949.4MiB   0.50%     1.03kB / 0B       168kB / 197kB     1
11f86ab355aa   burner6    6.83%     4.742MiB / 949.4MiB   0.50%     1.03kB / 0B       5.13MB / 197kB    1
b0f1735676ee   cont11     0.00%     6.105MiB / 949.4MiB   0.64%     1.47kB / 0B       0B / 4.1kB        82
adb3eb3ebdd3   cont10     0.00%     6.129MiB / 949.4MiB   0.65%     1.57kB / 432B     238kB / 4.1kB     82
93eebf614bdb   cont9      0.00%     6.117MiB / 949.4MiB   0.64%     1.47kB / 0B       135kB / 4.1kB     82
eded1aa5bac0   cont8      0.00%     6.422MiB / 949.4MiB   0.68%     24.7kB / 767kB    0B / 4.1kB        82
e674a225e027   cont7      0.00%     6.133MiB / 949.4MiB   0.65%     1.51kB / 0B       77.8kB / 4.1kB    82
07c4924729f8   cont6      0.00%     6.09MiB / 949.4MiB    0.64%     1.51kB / 0B       0B / 4.1kB        82
```

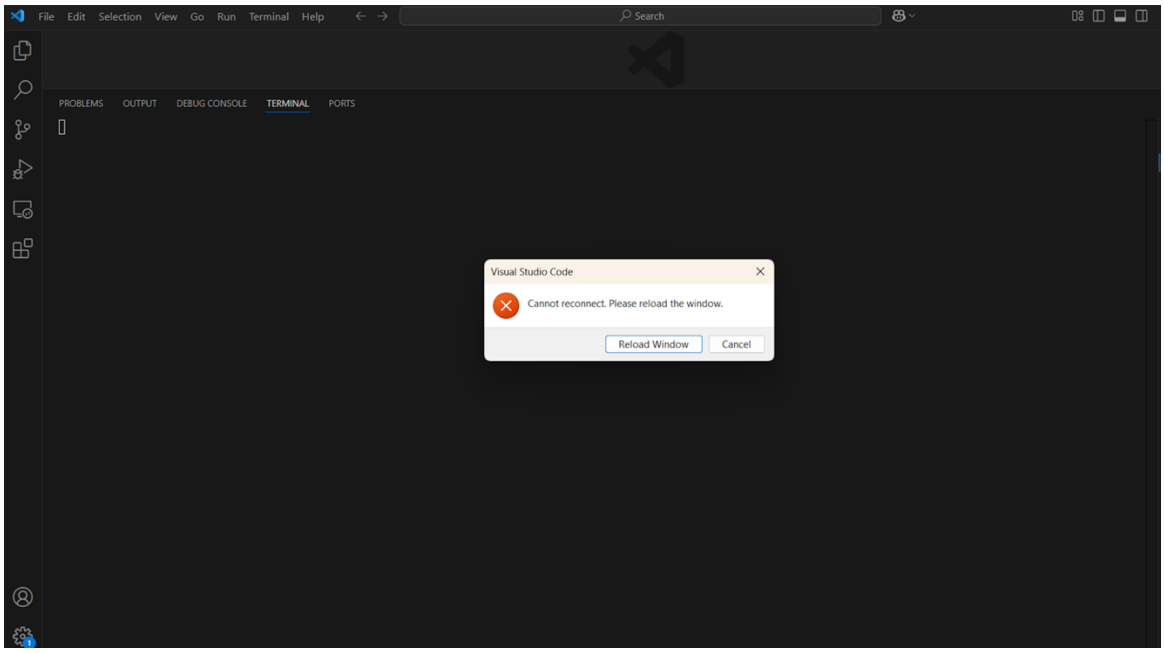
Fuente: propia

Figura: 37



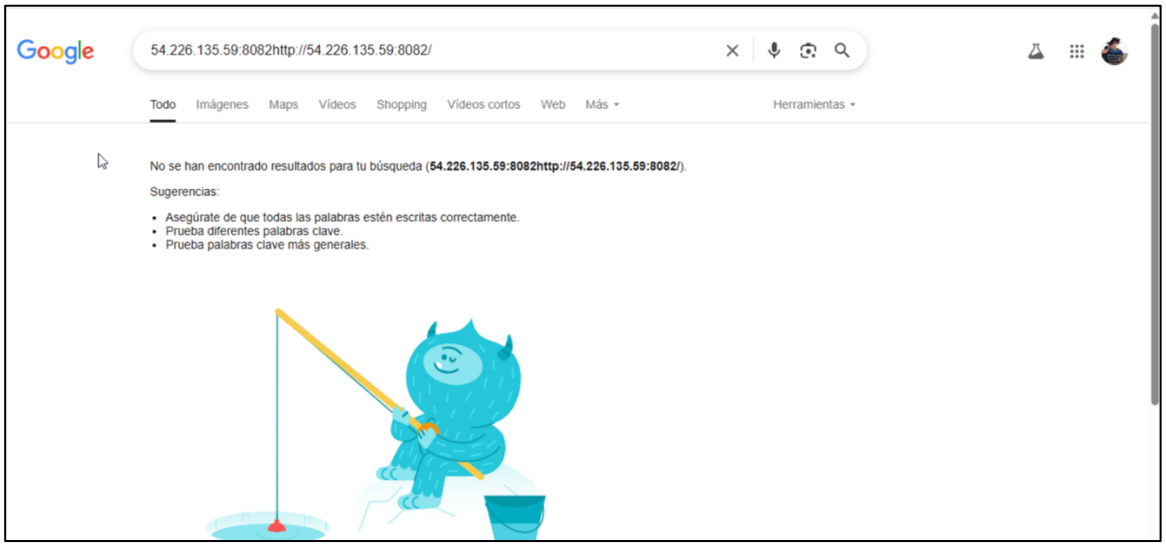
Fuente: propia

Figura: 38



Fuente: propia

Figura: 39



Fuente: propia

Conclusión

El presente proyecto partió de la implementación de instancias Windows y Linux en Amazon EC2, como parte de un entorno híbrido simulado en la nube, donde se configuraron servicios web accesibles desde Internet. Esta etapa inicial permitió comprender los principios del despliegue de infraestructura como servicio (IaaS), así como fortalecer habilidades en administración de sistemas y seguridad en entornos cloud.

Posteriormente, se abordó la implementación de contenedores Docker sobre una instancia gratuita de Linux, con el objetivo de simular un alto uso del CPU. Para ello, se desplegaron varios contenedores con servidores web que ofrecían contenido estático, accesibles mediante distintos puertos.

Durante el proceso, se realizaron configuraciones en los grupos de seguridad para permitir el acceso externo a cada contenedor, y se verificó su correcto funcionamiento. Asimismo, se utilizaron herramientas como top para monitorear en tiempo real el consumo de recursos, lo que permitió analizar el rendimiento del sistema bajo diferentes niveles de carga.

Este ejercicio demostró la eficiencia de Docker como tecnología de contenedores, permitiendo ejecutar múltiples servicios de forma aislada con un consumo mínimo de recursos en comparación con la virtualización tradicional. Además, permitió reforzar conocimientos sobre redes, administración de sistemas y despliegue de servicios en entornos controlados.

En conclusión, el proyecto ofreció una visión práctica del potencial de AWS y Docker combinados, destacando su aplicabilidad tanto en escenarios empresariales como en entornos de prueba y simulación, especialmente en contextos con recursos limitados como las máquinas free tier.

Referencias

(Ec2-Ug, n.d.; Uso_de_aws, n.d.; Windows ISS, n.d.; Juan Sánchez Hernández, n.d.)

Juan Sánchez Hernández, J. (n.d.). Implantación de Aplicaciones Web. uso_de_aws. (n.d.). Windows ISS. (n.d.).

Amazon Web Services, Inc. (2023). Amazon EC2 User Guide for Linux Instances. Recuperado de https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/

Ghazali, M., Kamaludin, A., & Ahmad, M. N. (2021). Cloud Computing Adoption: A Literature Review on Implementation Models. Journal of Information Systems Research and Innovation*, 19(1), 10–20. https://doi.org/10.24191/jisri.v19i1.12608*

Abbas, A., & Khan, S. U. (2014). A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. IEEE Journal of Biomedical and Health Informatics*, 18(4), 1431–1441. https://doi.org/10.1109/JBHI.2013.2290371*