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

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

Computacion en la nube, AWS, EC2, Docker, Contenedores, Servidores Virtuales, Infraestructura como Servicio, VPC, Vol menes, Automatizacion, 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 basicamente 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 ademas 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.

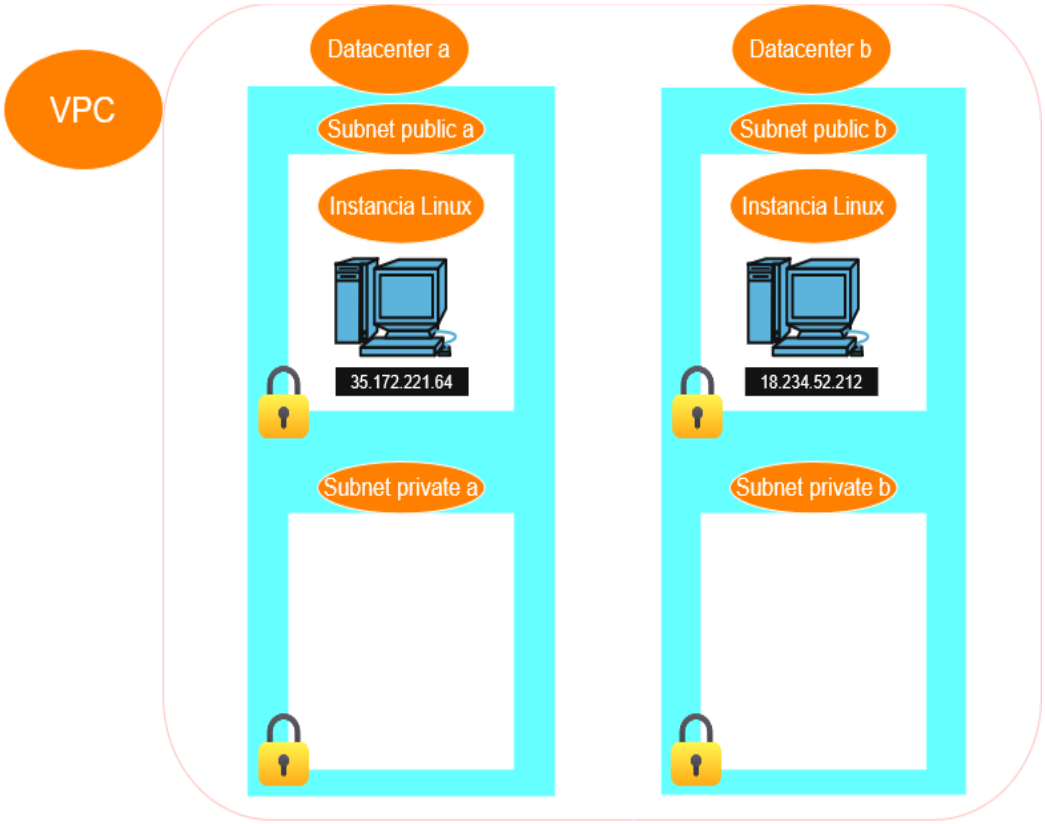

Figura 1

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).

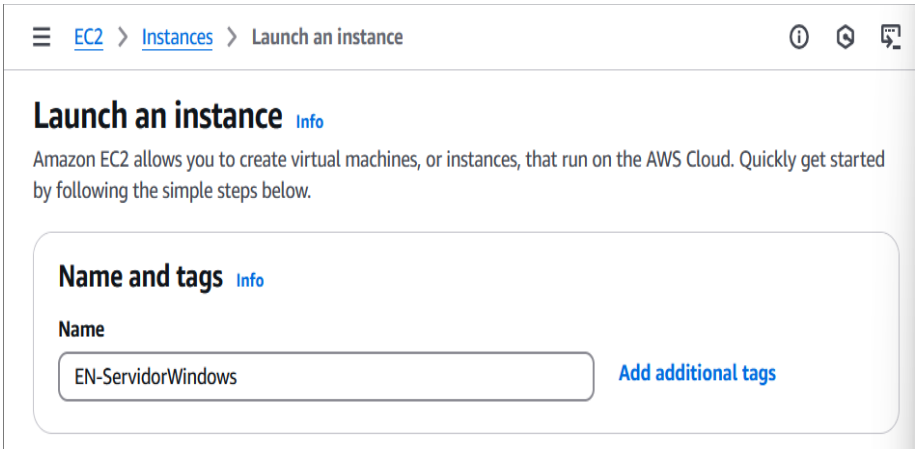

Figura 2 Asignación nombre de la instancia.

2. 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.

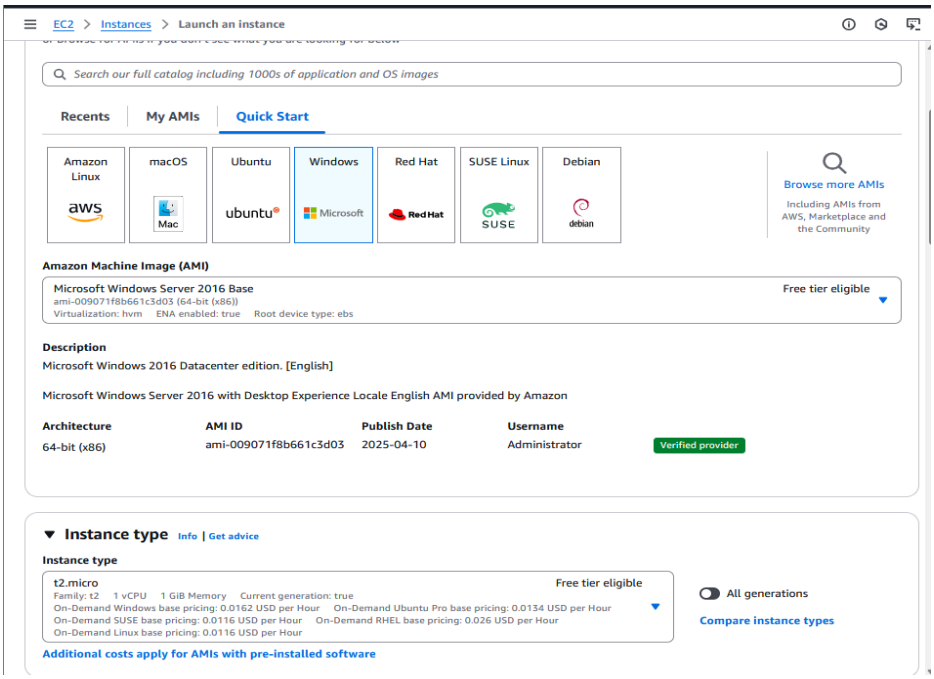

Figura 3 Asignar AMI.

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

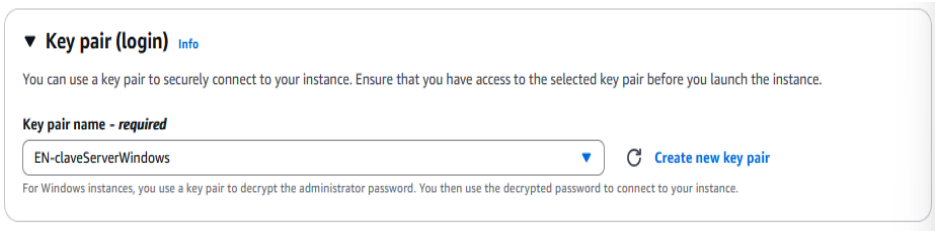

Figura 4 Asignación llave.

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.

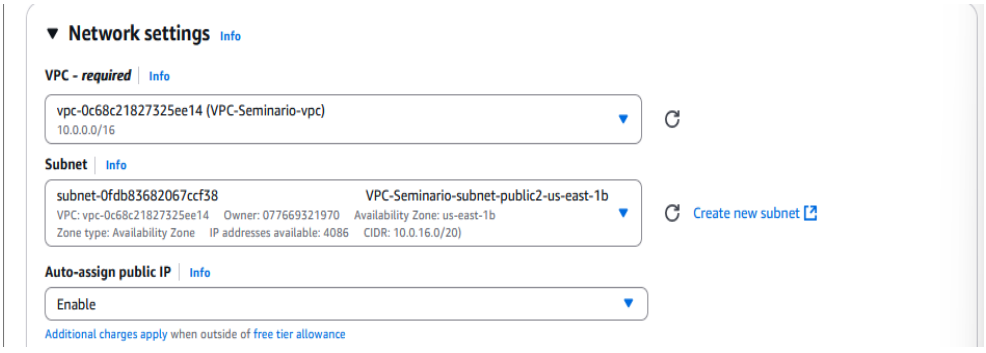

Figura 5 Configuración de red.

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.

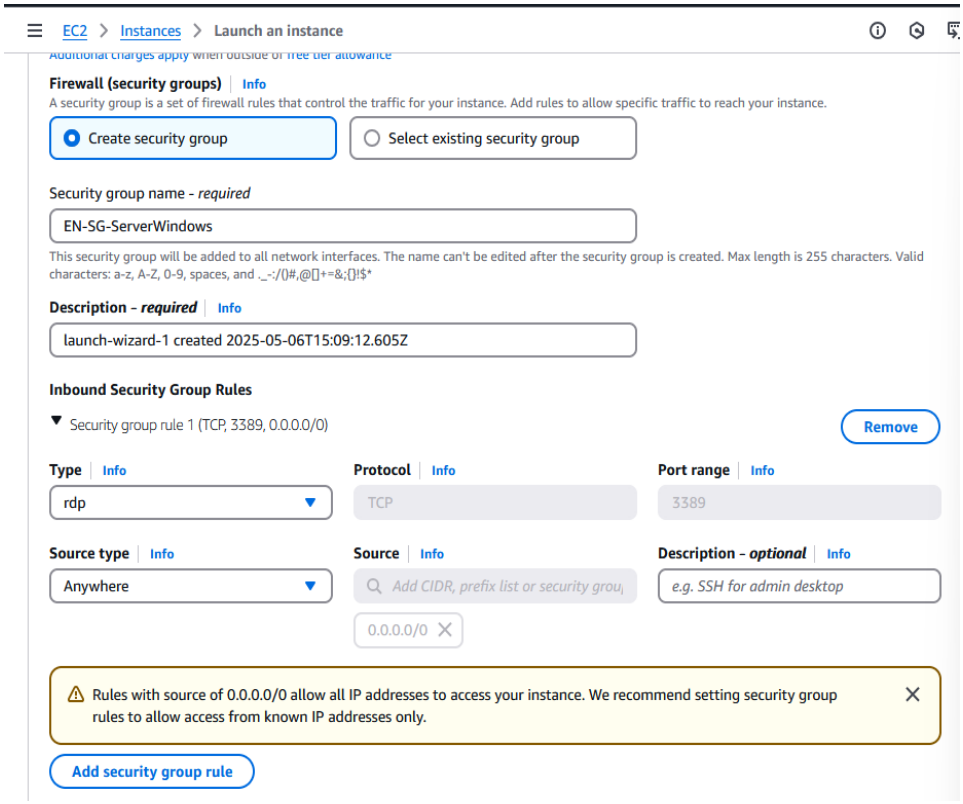

Figura 6 Configuración firewall

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

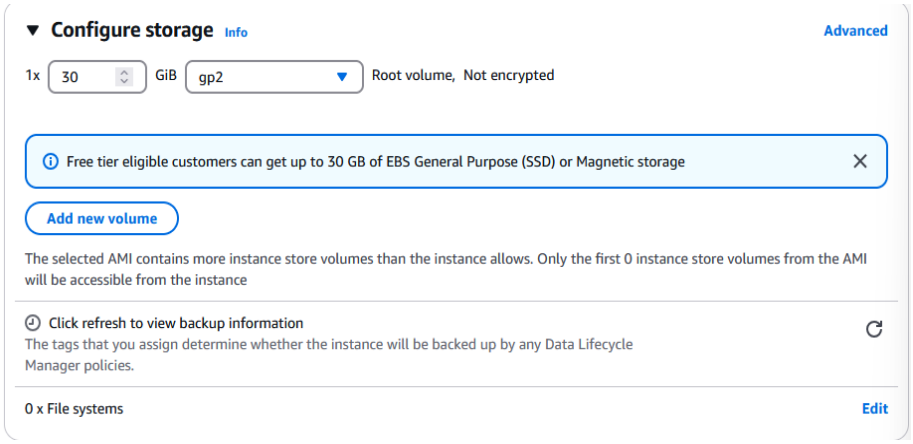

Figura 7 Configuración de almacenamiento.

Configuración del servidor web Windows.

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

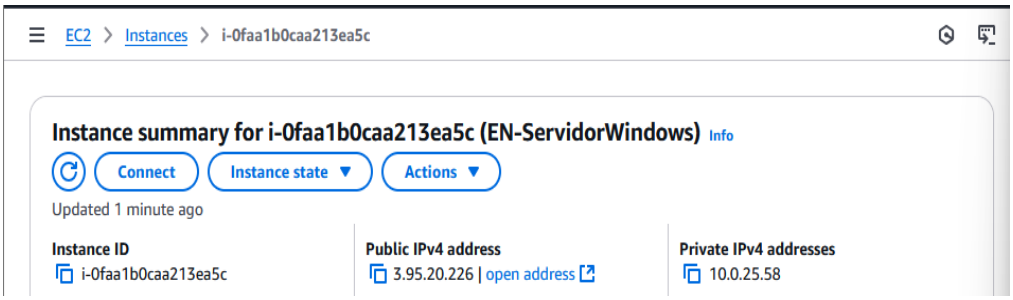

Figura 8 IP habilitada.

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

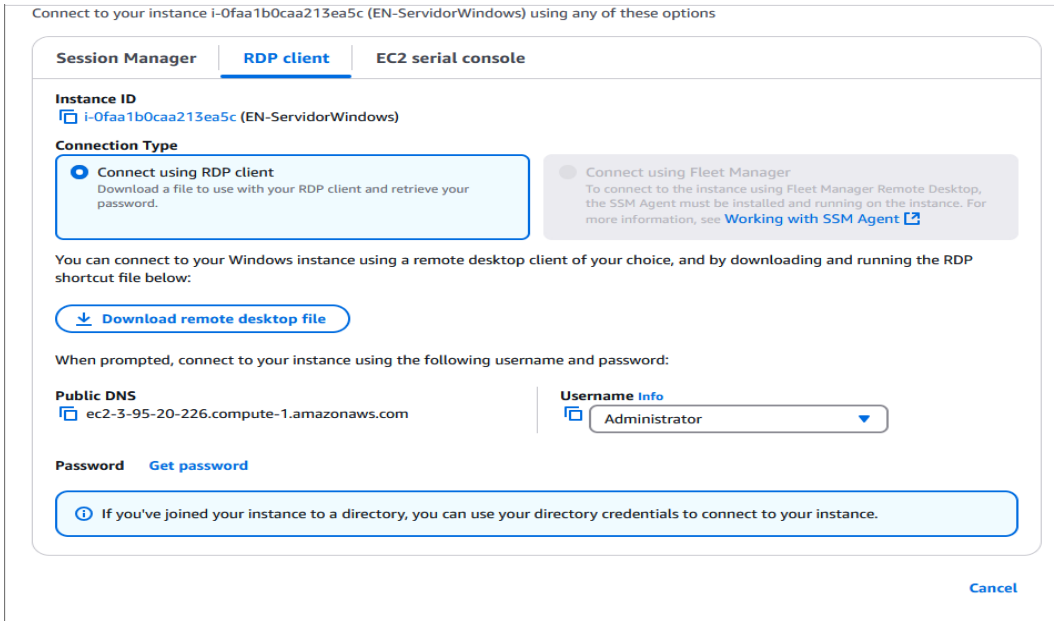

Figura 9 RDP client

3. Luego de ellos obtendremos la contraseña creada anteriormente.

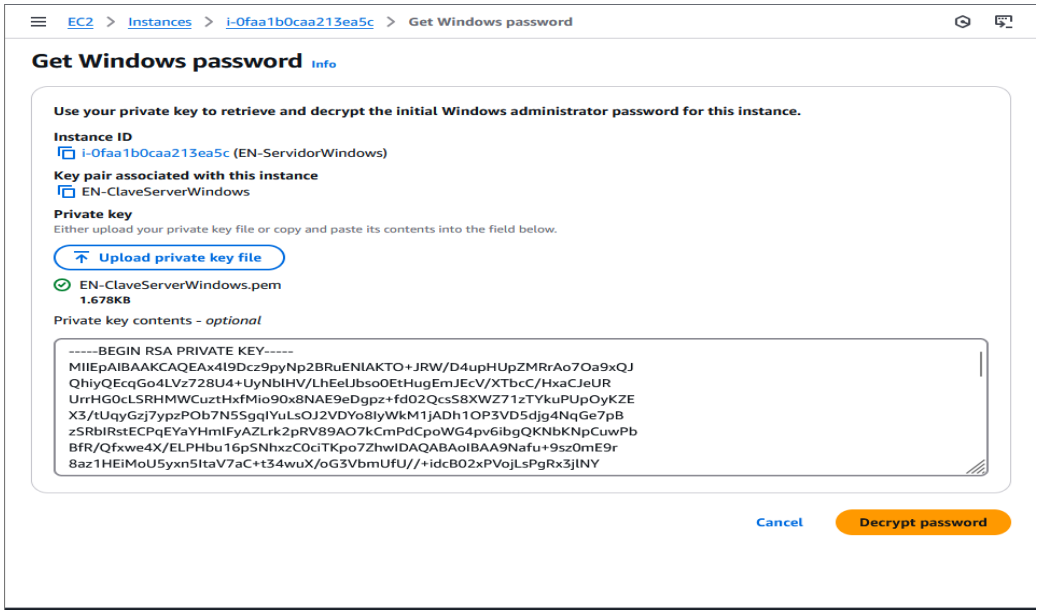

Figura 10 Obtener contraseña.

4. 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.

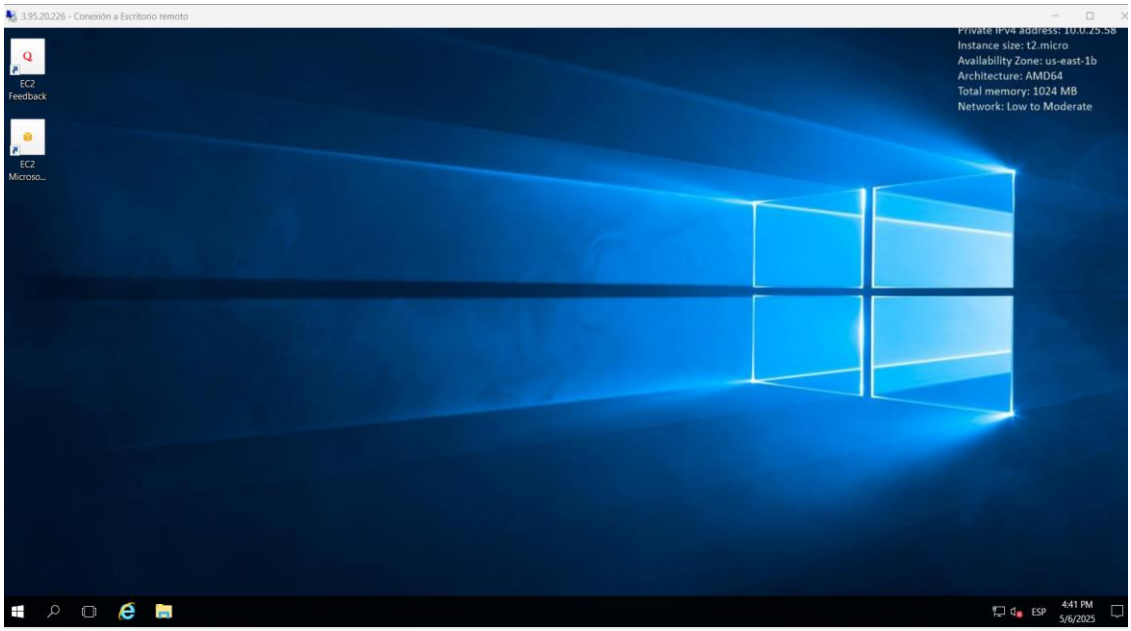

Figura 11 Acceso máquina virtual Windows.

5. 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.

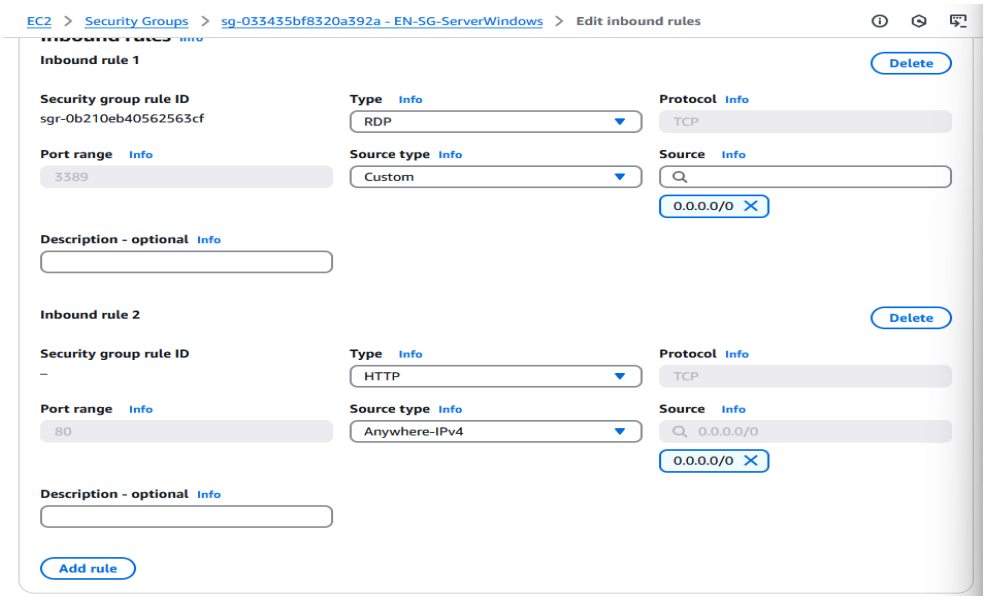

Figura 12 Configuración puerto 80.

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.

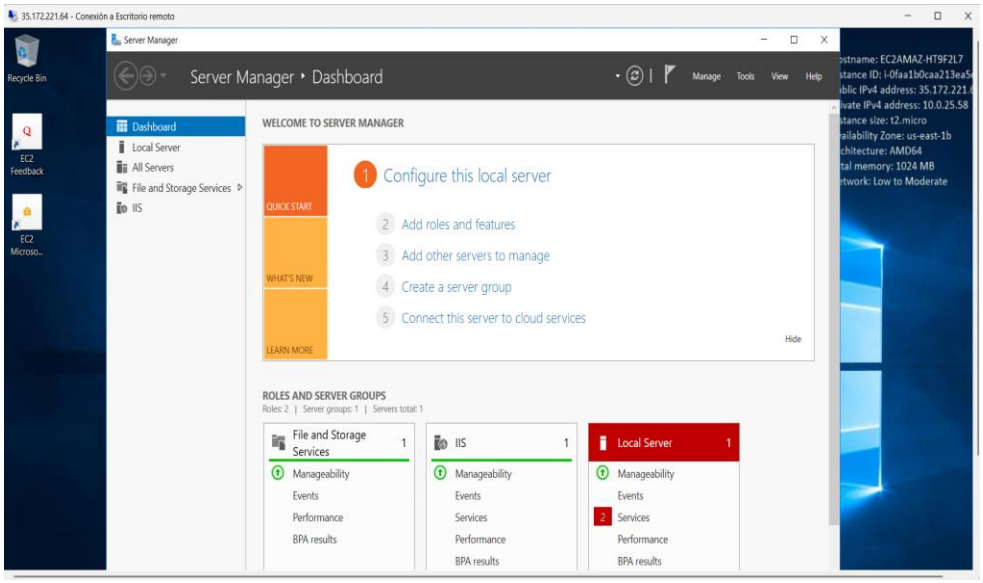

Figura 13 Creación servidor web.

7. Luego de ello accedemos a la opcion 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.

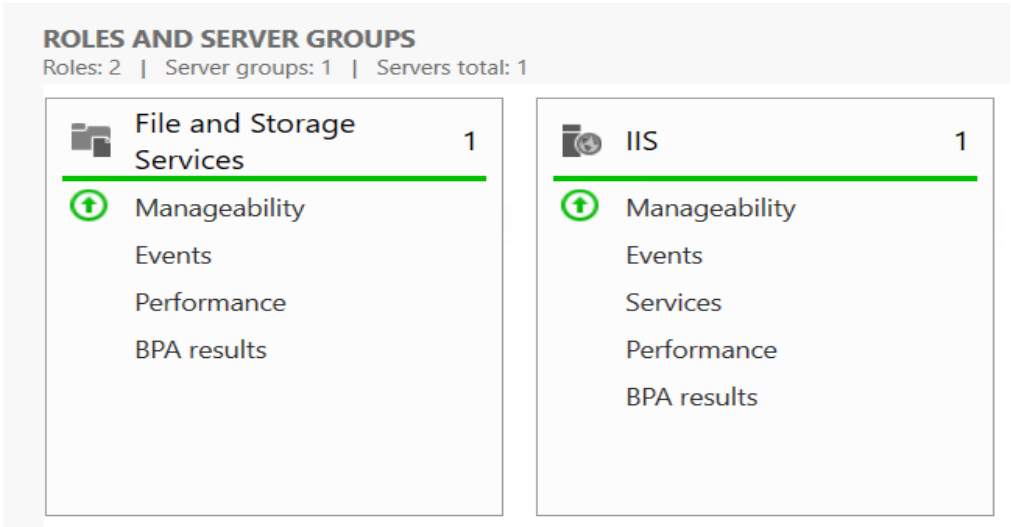

Figura 14 Instalador de rol web.

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

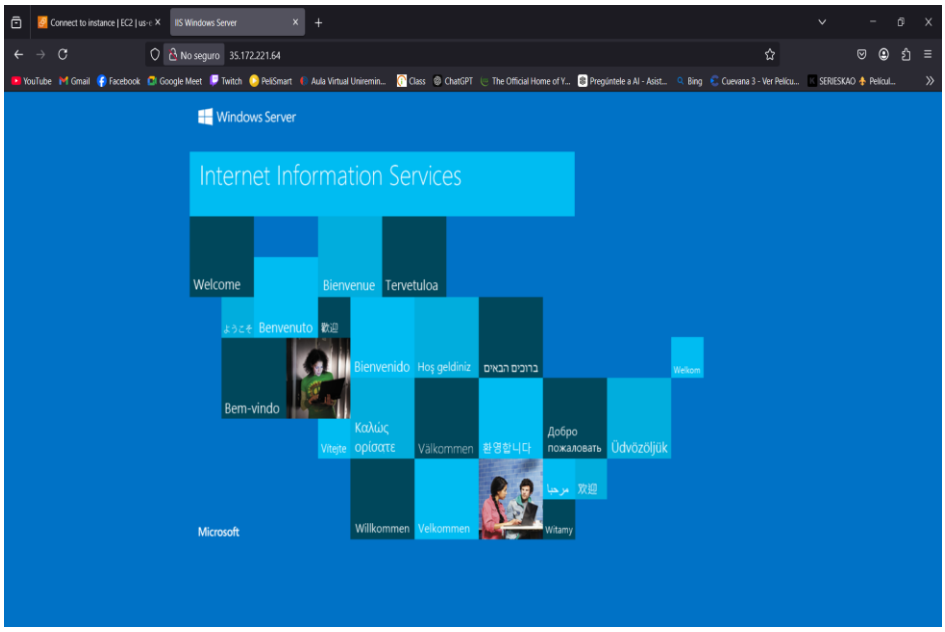

Figura 15 Acceso servidore web.

Creación Instancia Linux.

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

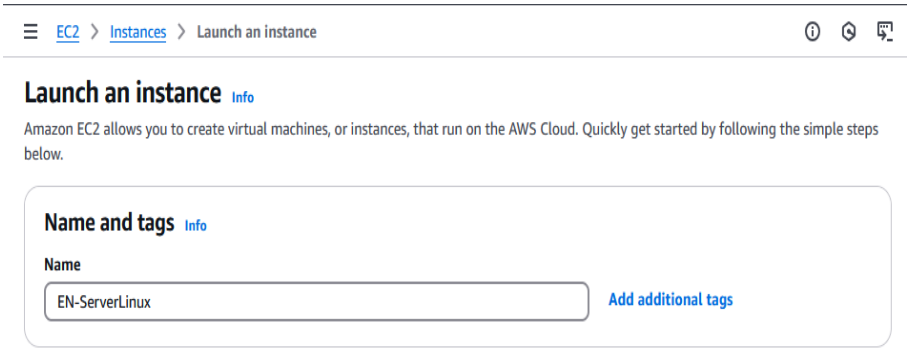

Figura 16 Nombre instancia Linux.

2. 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.

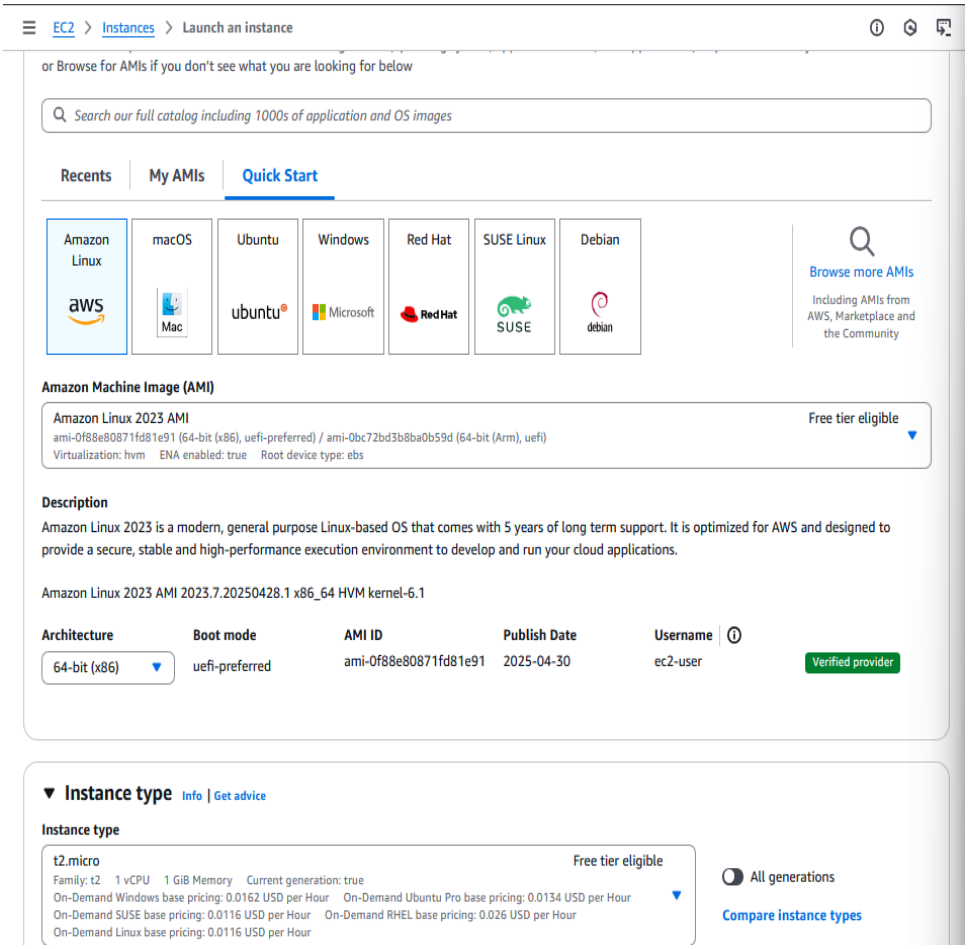

Figura 17 Selección AMI.

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

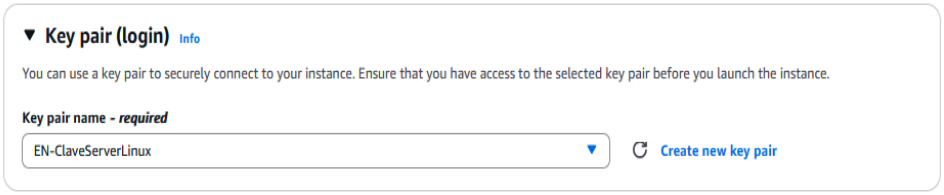

Figura 18 Creación llave de seguridad.

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.

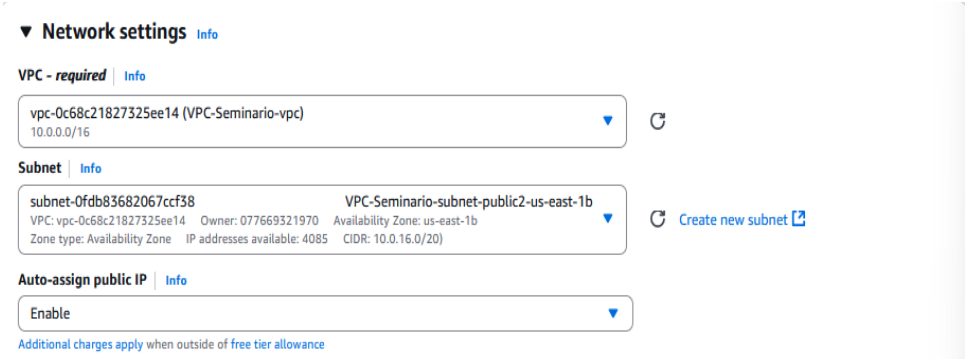

Figura 19 Configuración de red.

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.

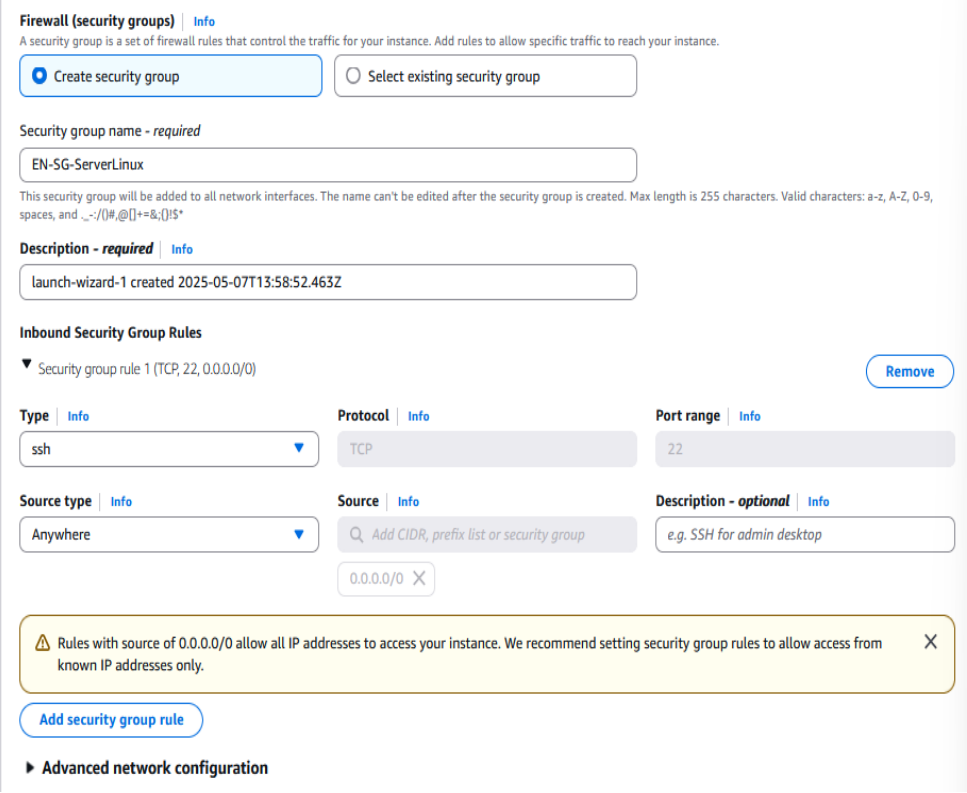

Figura 20 Configuración firewall.

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).

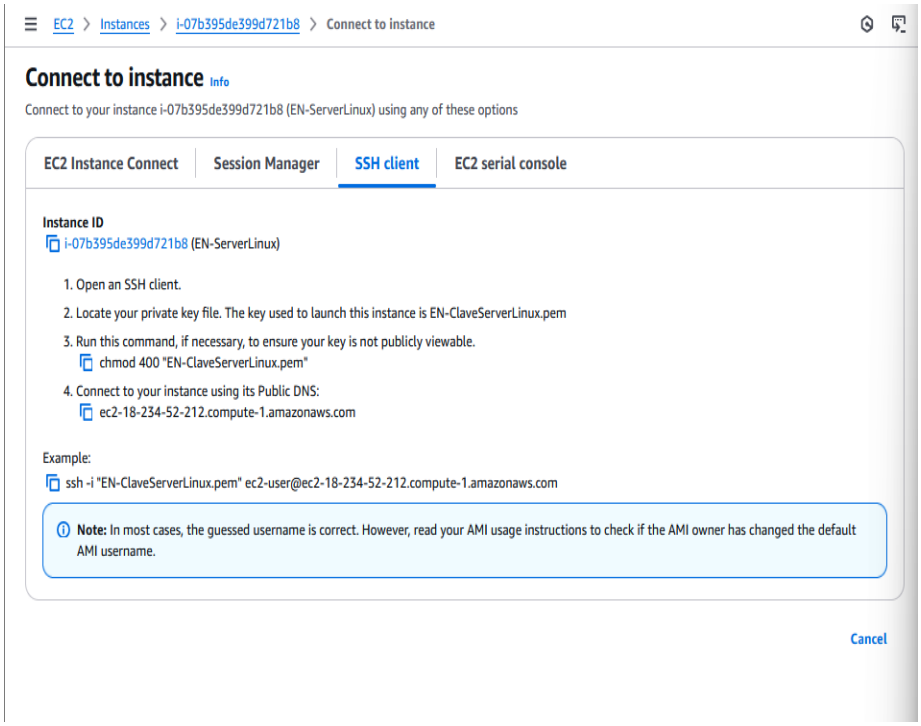

Figura 21 Configuración web Linux

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

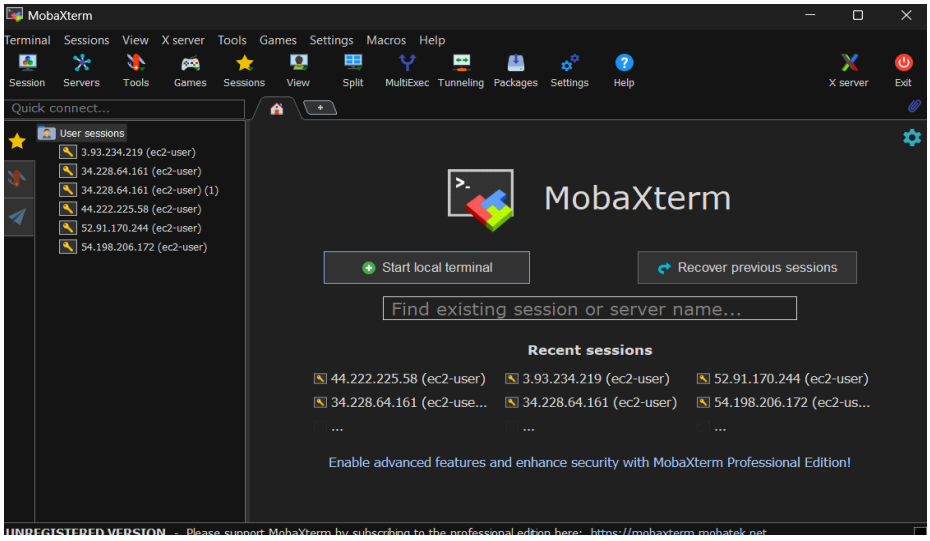

Figura 22 Conexión mediante SSH.

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

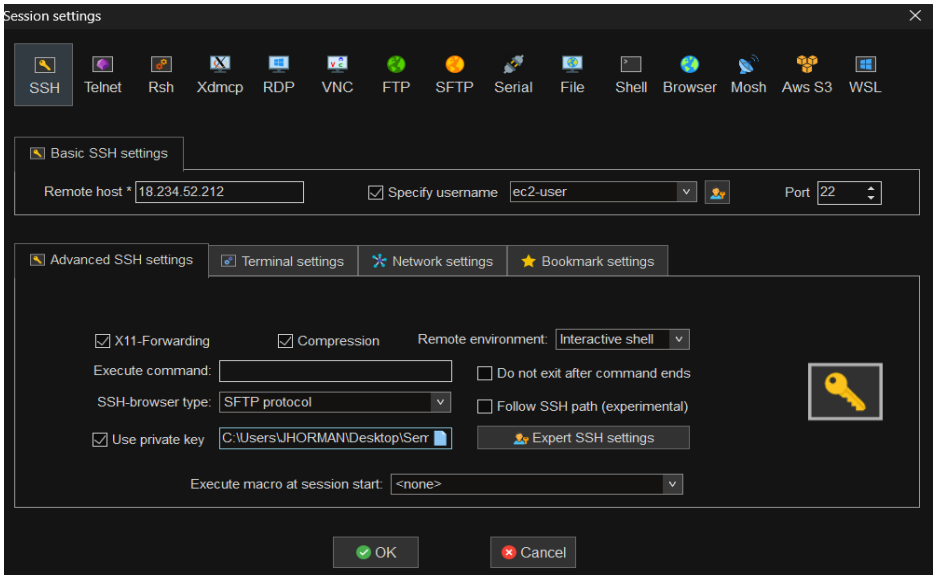

Figura 23 Conexión con IP y clave de seguridad

4. 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.

5. 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.

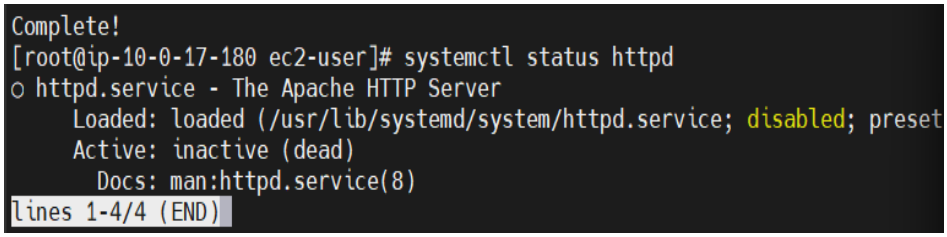

Figura 25 Verificación estado aplicación.

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.**

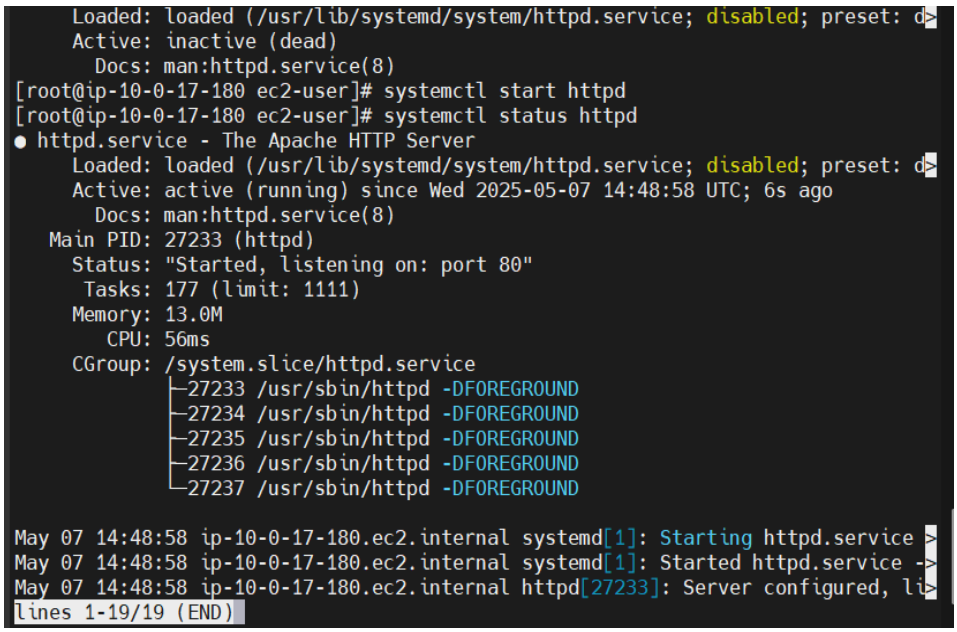

Figura 26 Inicialización de la aplicación.

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.

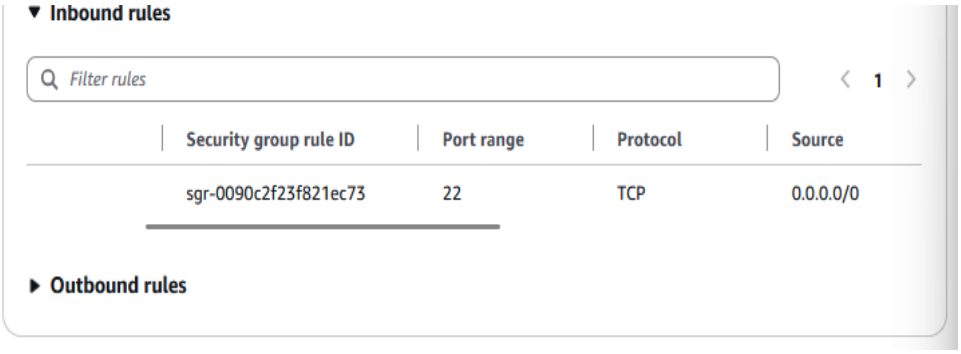

Figura 27 Configuración puerto 80.

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.

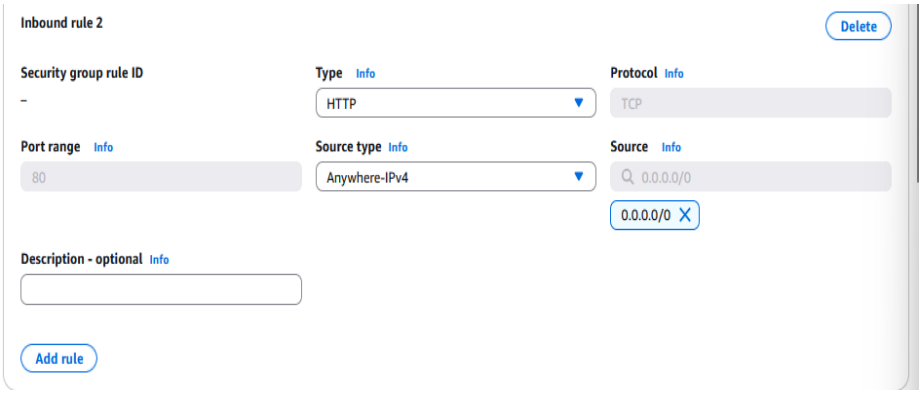

Figura 28 Configuración nueva regla http.

9. Evidencia de que permite ingresar desde la web.

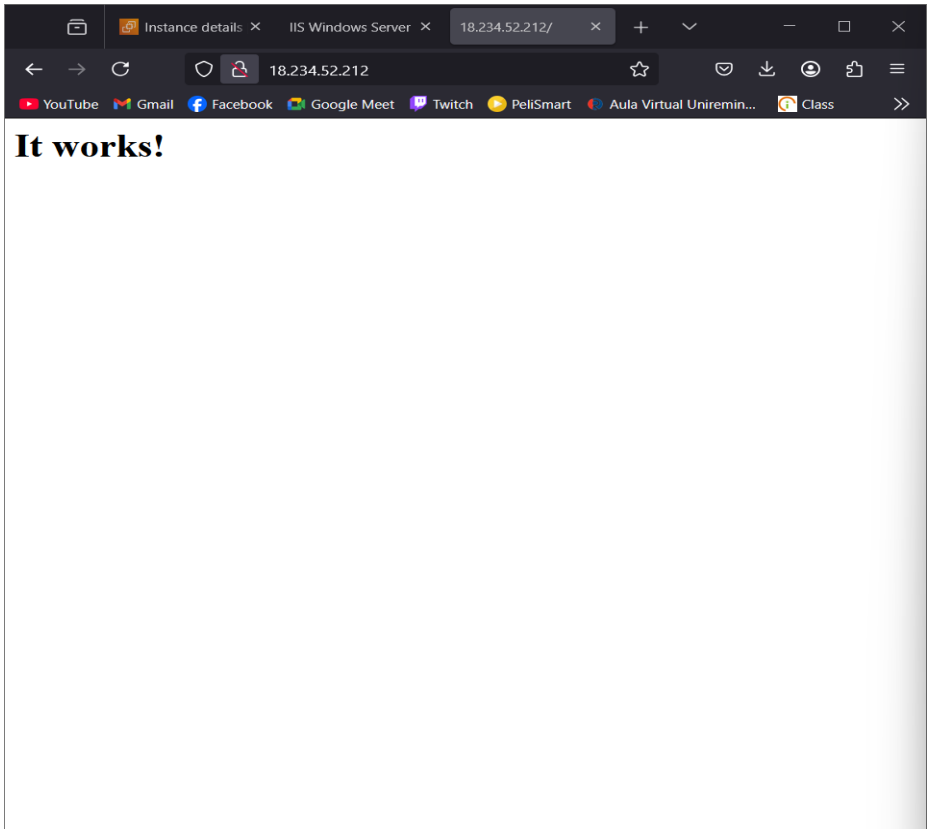

Figura 29 Aplicación web corriendo.

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

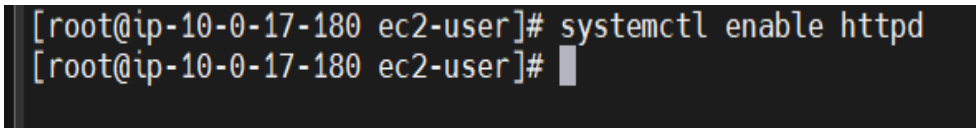

Figura 30 Arranque automático.

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).

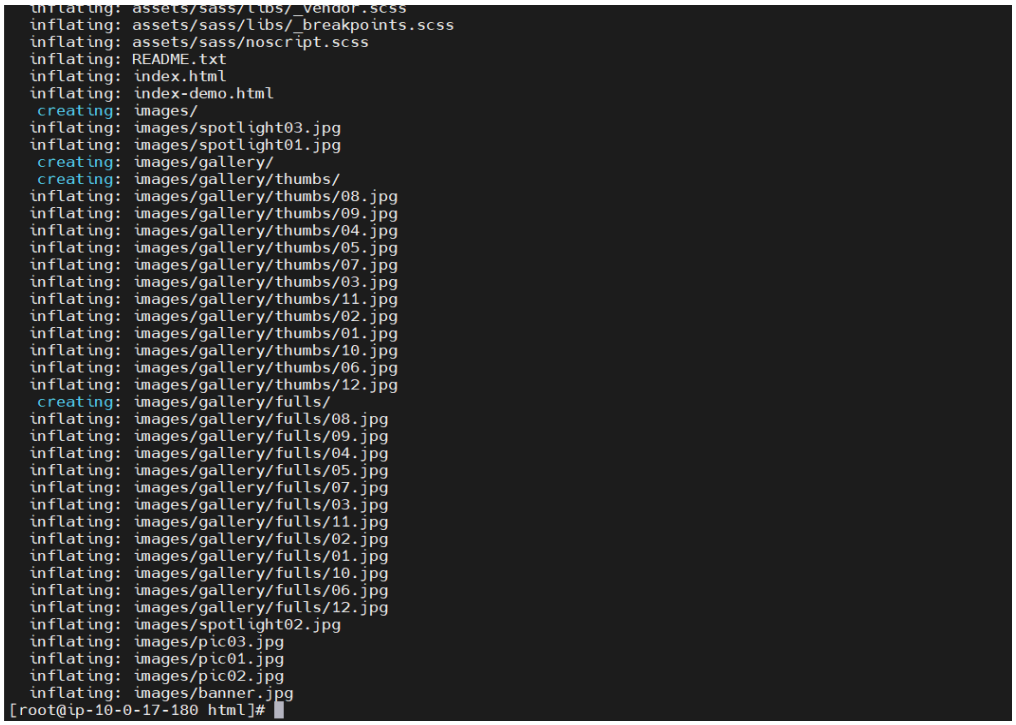

Figura 31 Instalación plantilla HTML.

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.

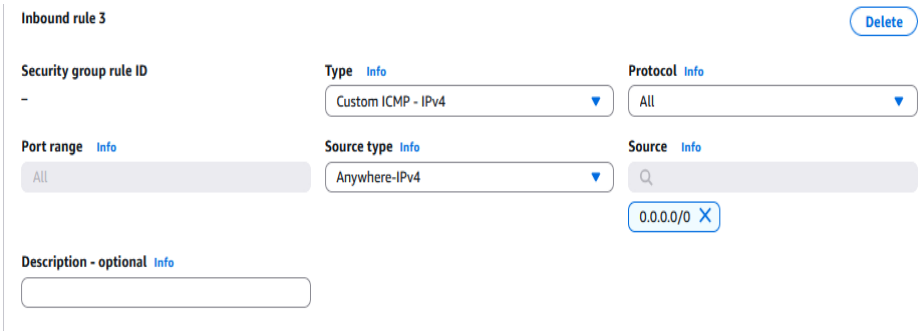

Figura 32 Nueva regla ICMP Windows.

Ping exitoso desde mi instancia Windows a Linux.

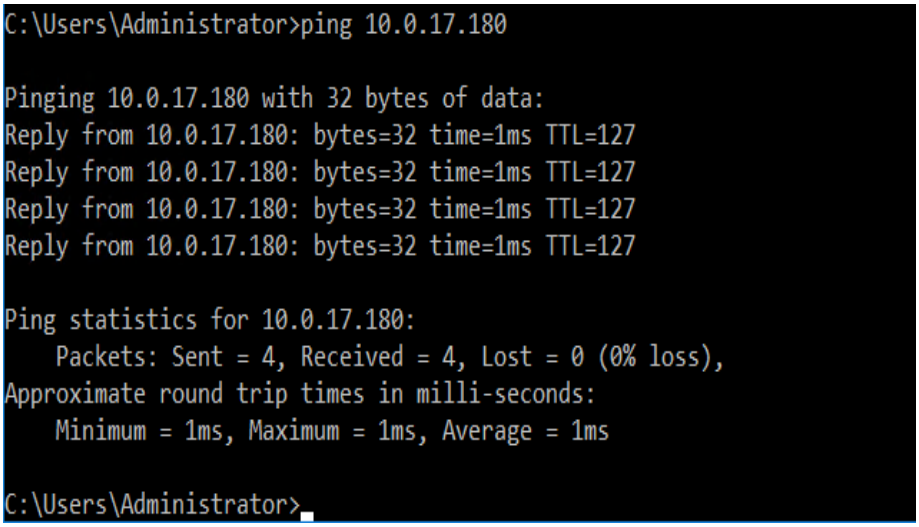

Figura 33 Ping Windows a la instancia Linux.

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.

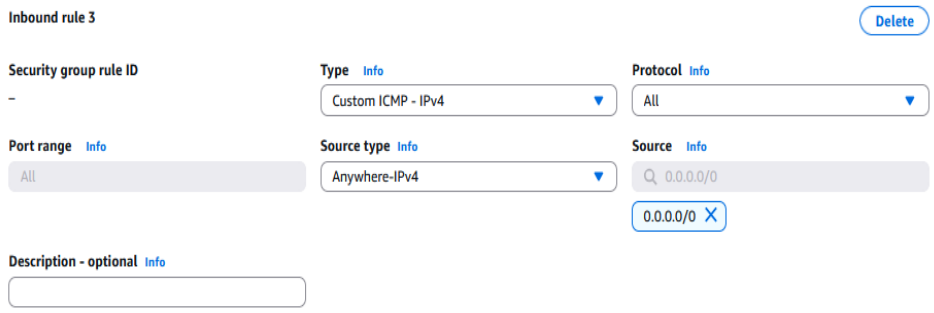

Figura 34 Configuración ICMP.

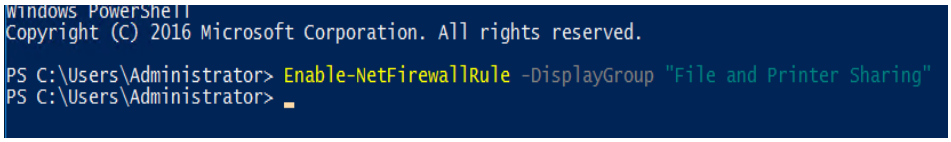

Figura 35 Configuración ICMP.

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.**

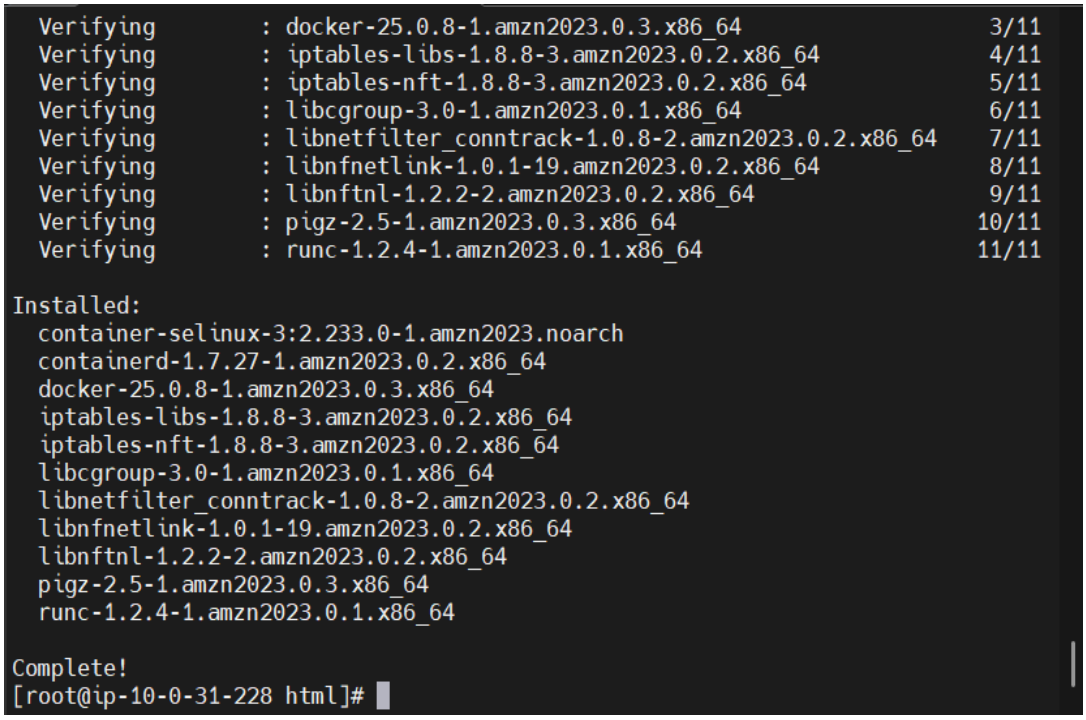

Figura 37 Instalación Docker.

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

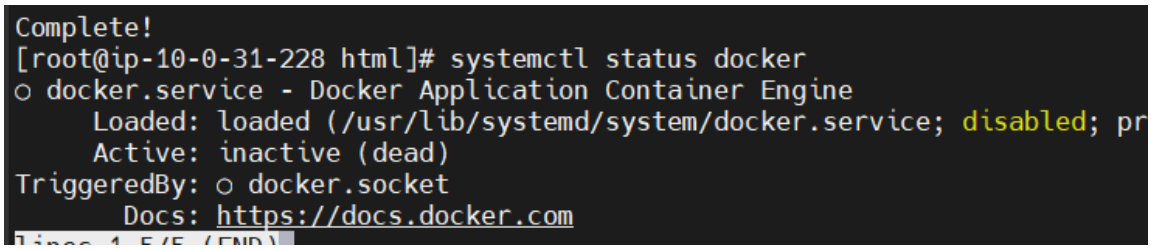

Figura 38 Verificación estado docker.

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).

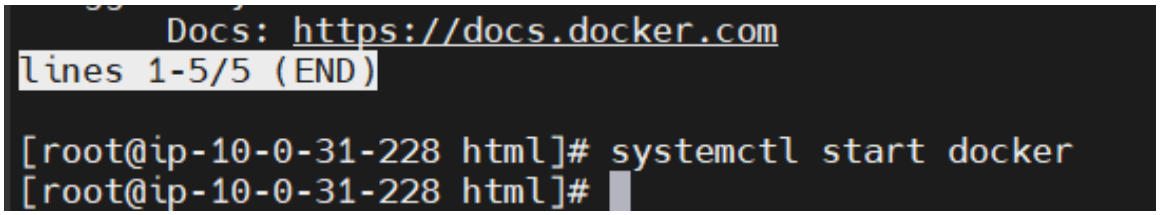

Figura 39 Inicialización docker.

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

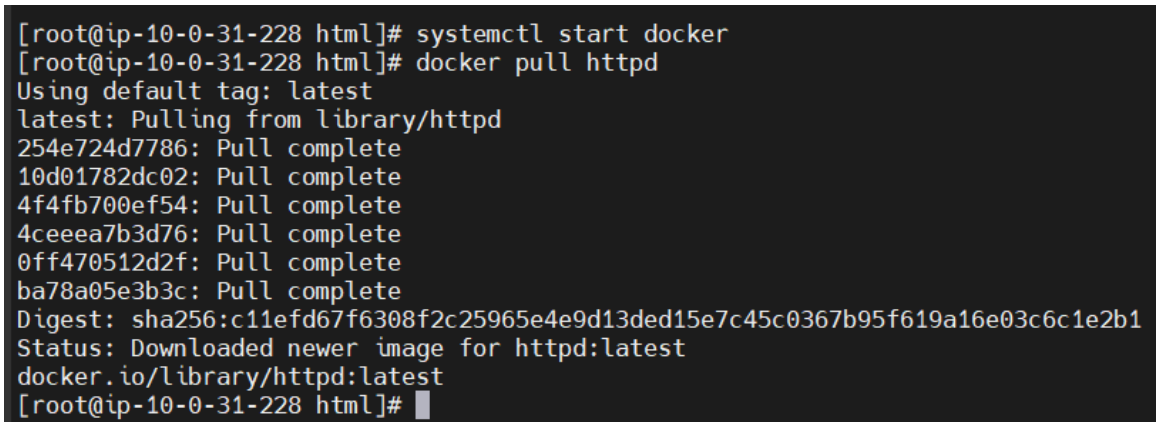

Figura 40 Imagen HTTPD.

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

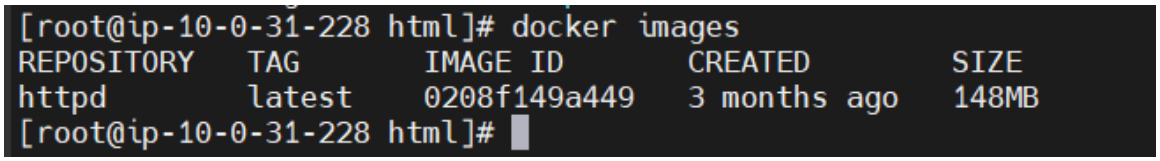

Figura 41 Verificar imagen.

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

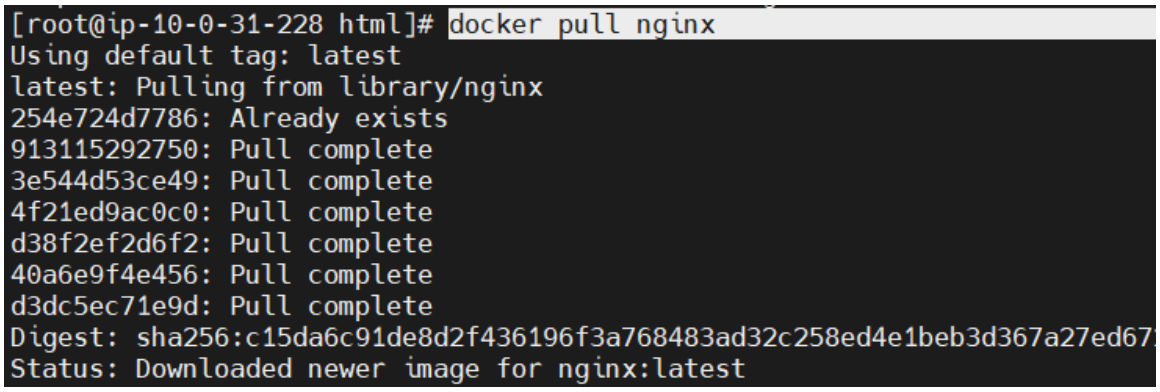

Figura 42 Descargar imagen nginx.

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

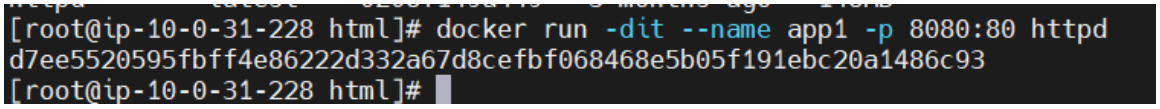

Figura 43 Creación del contenedor HTTPD.

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                                   NAMES
d7ee5520595f   httpd     "httpd-foreground"   About a minute ago   Up About a mi
nute   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 evidencia nuestra web necesitamos realizar el ingreso de una nueva regla en nuestro security group con el fin de que se permita este puerto.

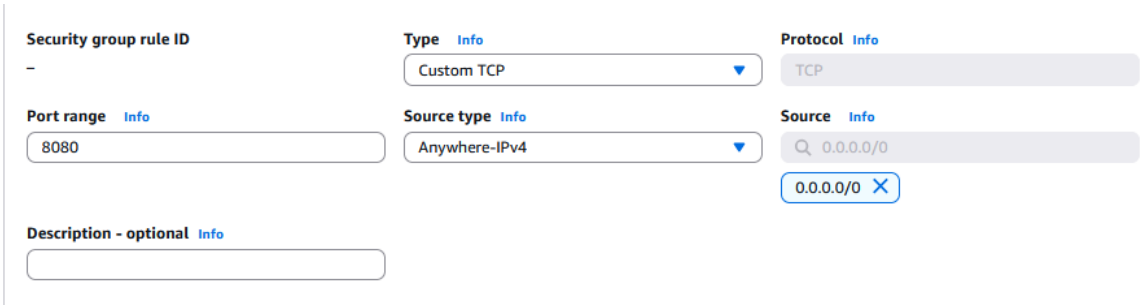

Figura 45 Puerto servidor web.

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.

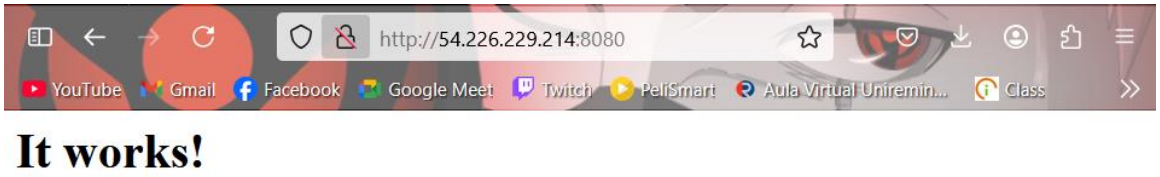

Figura 46 Evidencia del contenedor.

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 contendor con diferentes características.

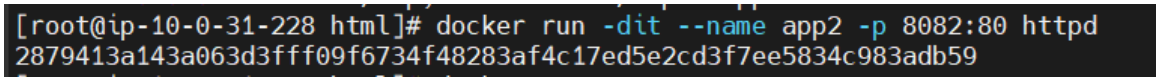

Figura 47 Contenedor con diferentes características.

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.

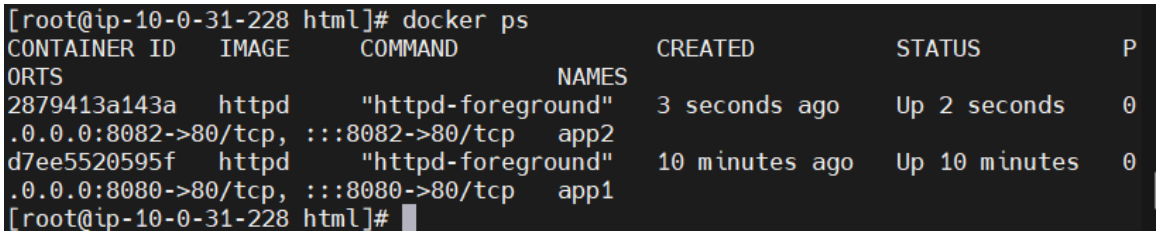

Figura 48 Nuevo contenedor.

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

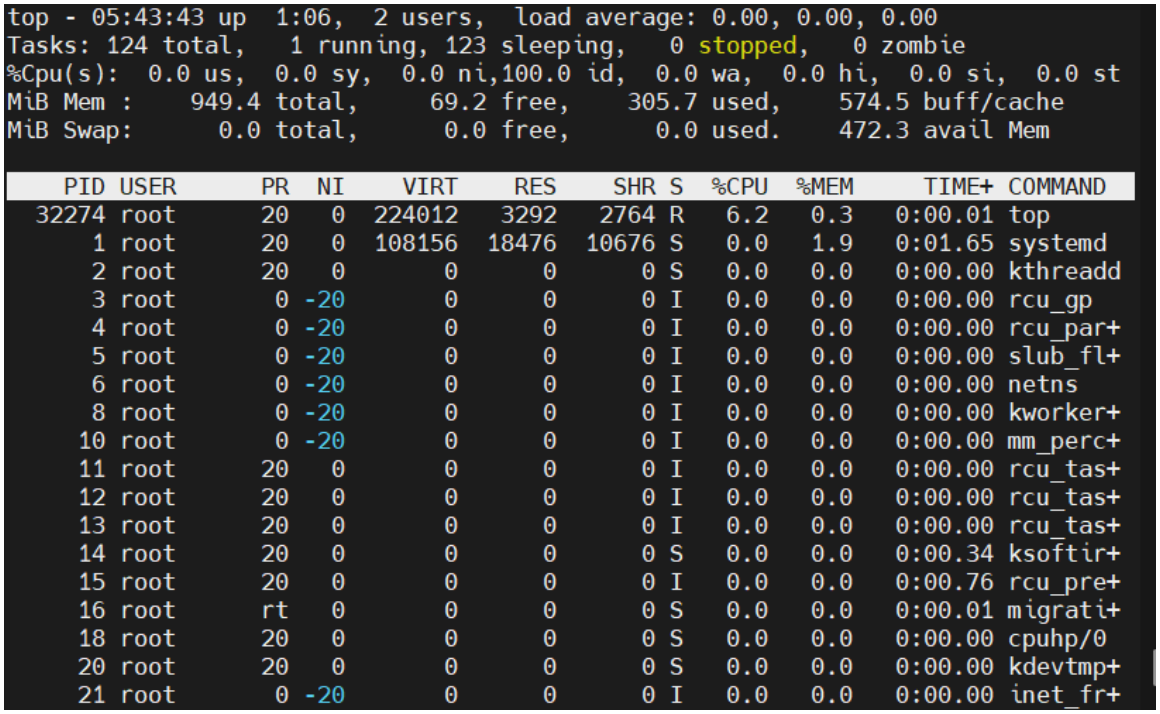

Figura 49 Verificar consumo de la instancia.

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

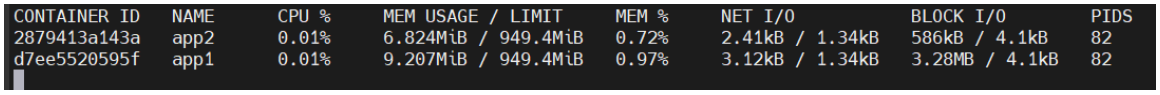

Figura 50 Verificar consumo de los contenedores.

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.

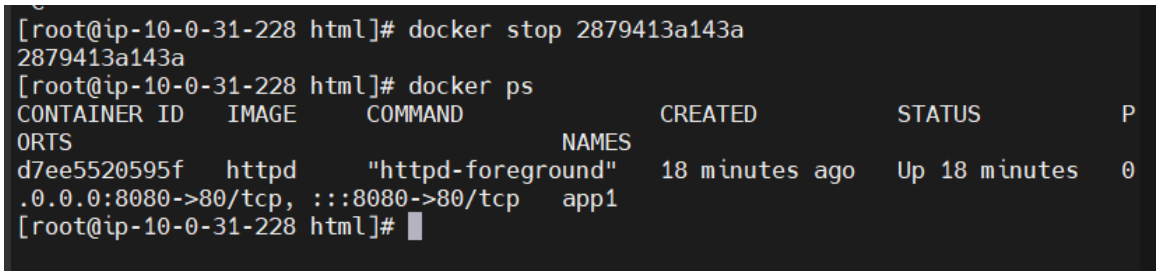

Figura 51 Detener un contenedor.

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
               PORTS                                    NAMES
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
73d567dea322c6de0513b2e6f462bb82fa71370960c0694316092cffa0108350
[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.

Figura 56 Mostrar plantilla desde internet.

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

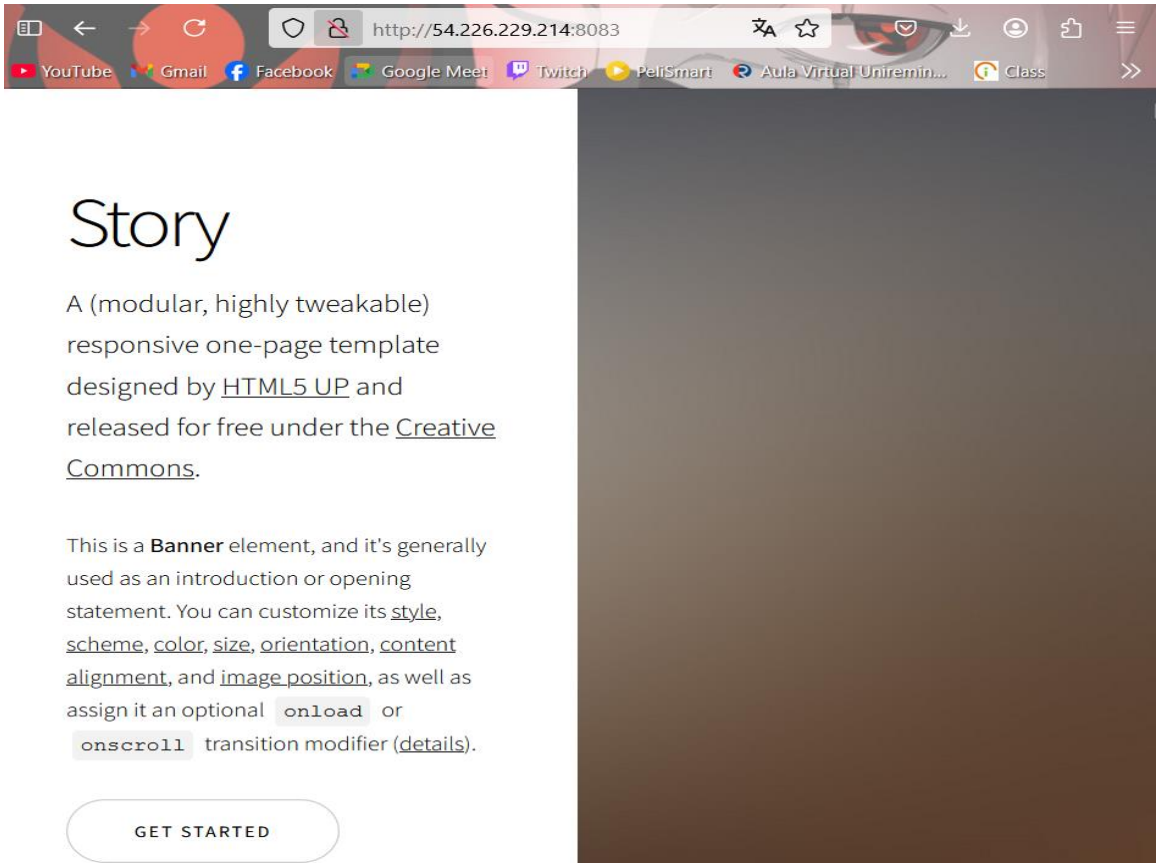

Figura 57 Ejecución del contenedor del sitio.

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.

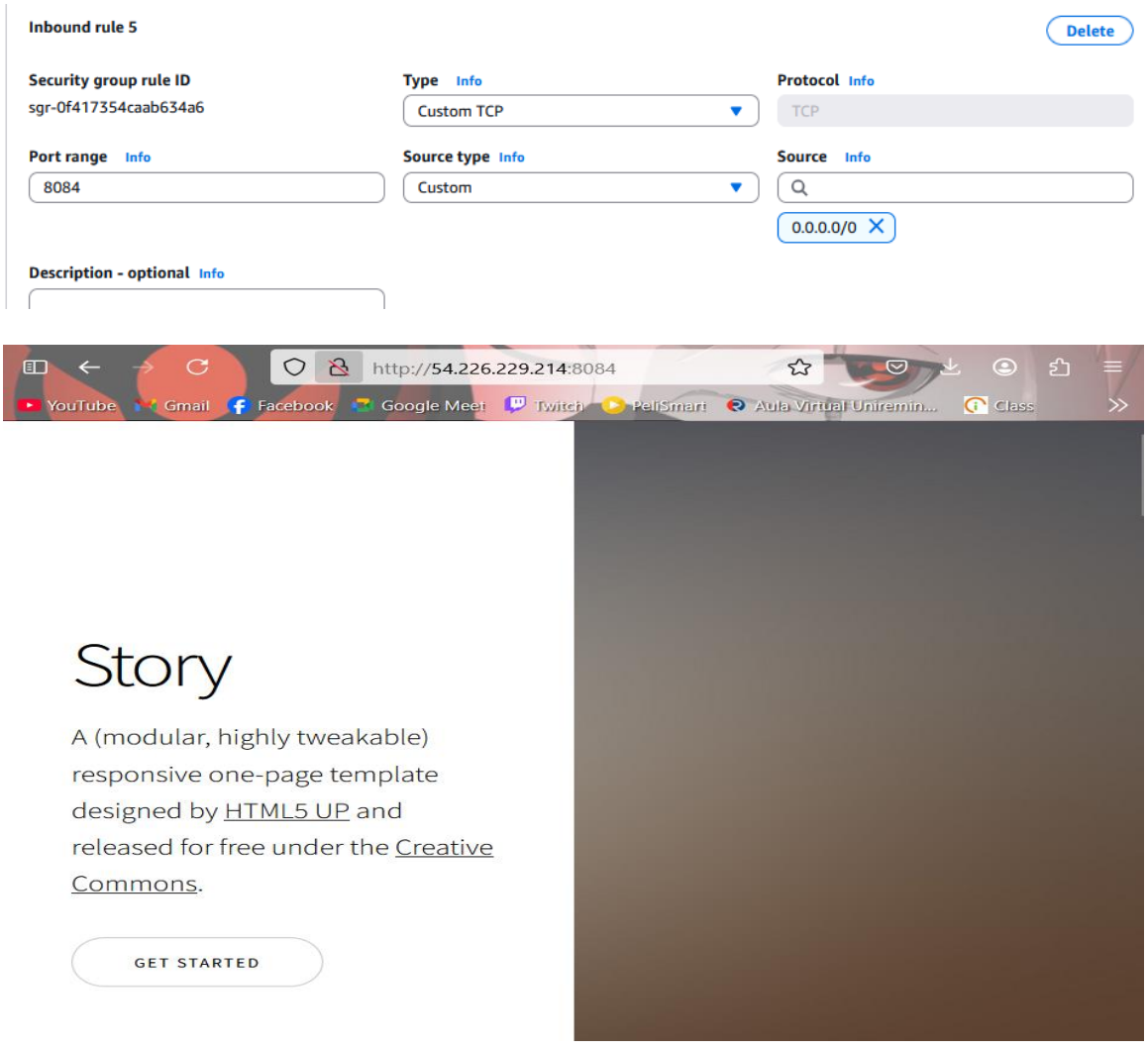

Figura 59 Creación del contenedor con la misma ruta

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.**

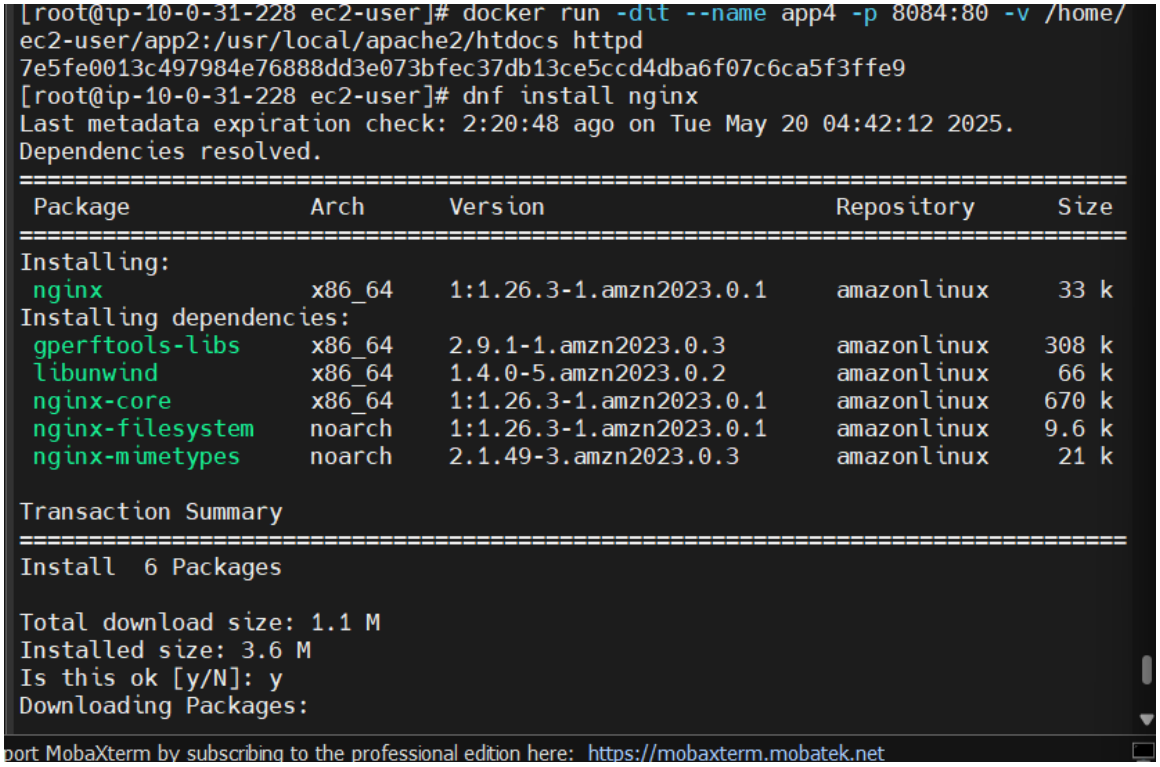

Figura 60 Instalar servicio nginx.

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.

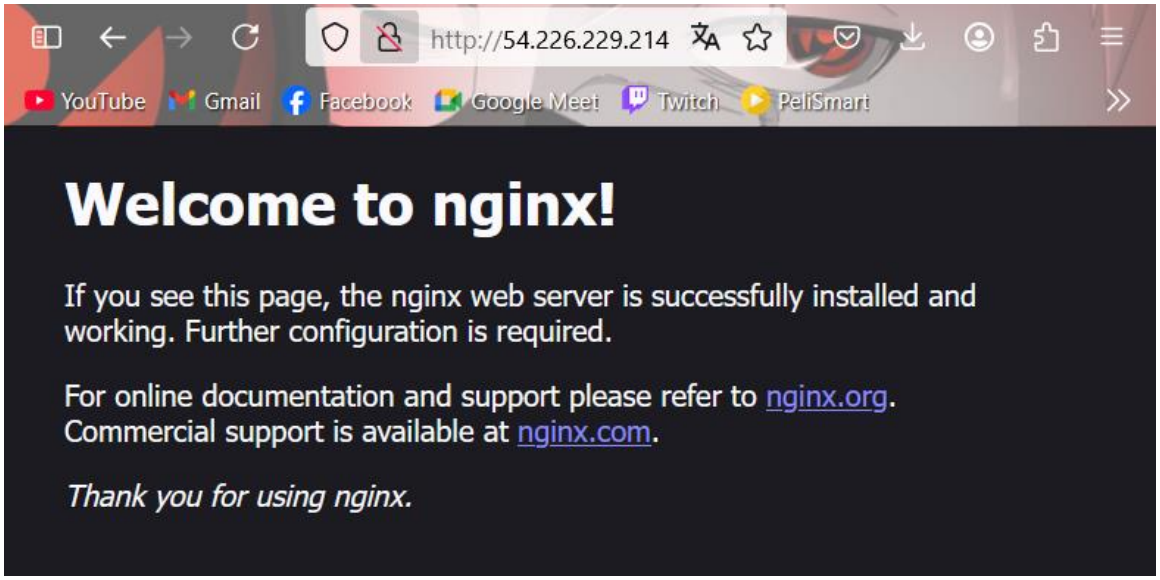

Figura 62 Nueva regla para el puerto 80 nginx.

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.

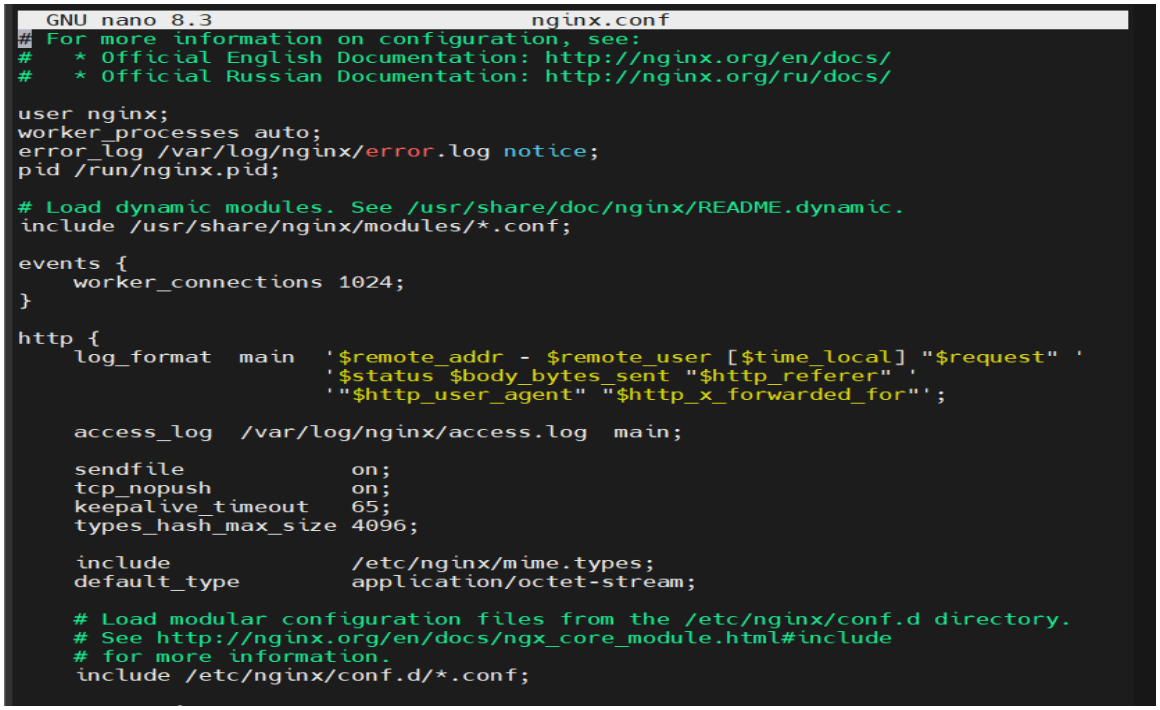

Figura 63 Configuración proxy reverso.

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:

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.

Figura 65 Evidencia implementación de proxy reverso.

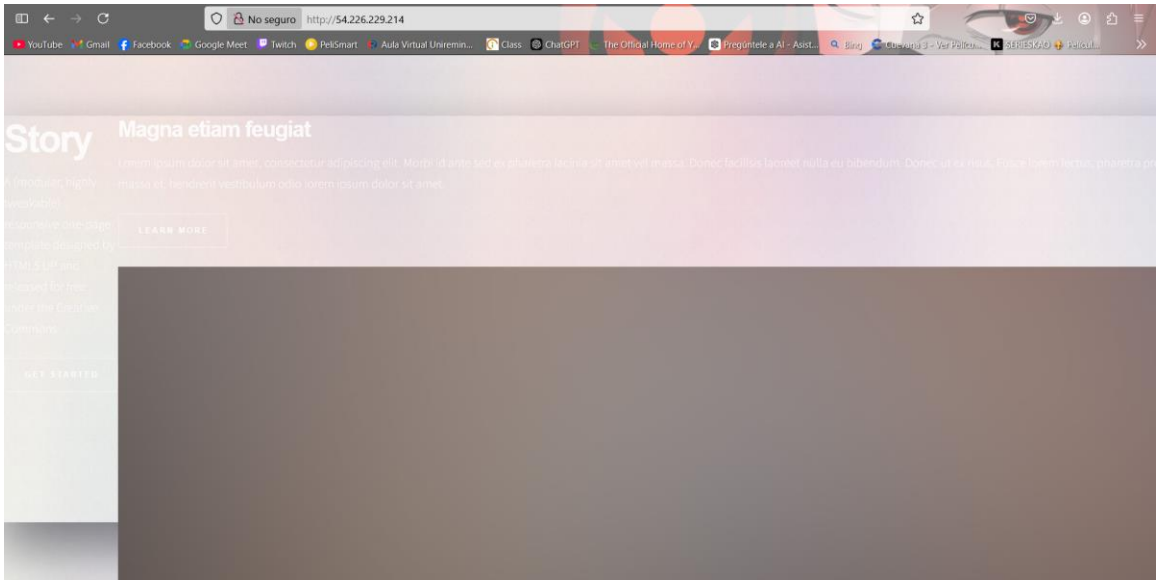

Figura 66 Evidencia implementación de proxy reverso.

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.

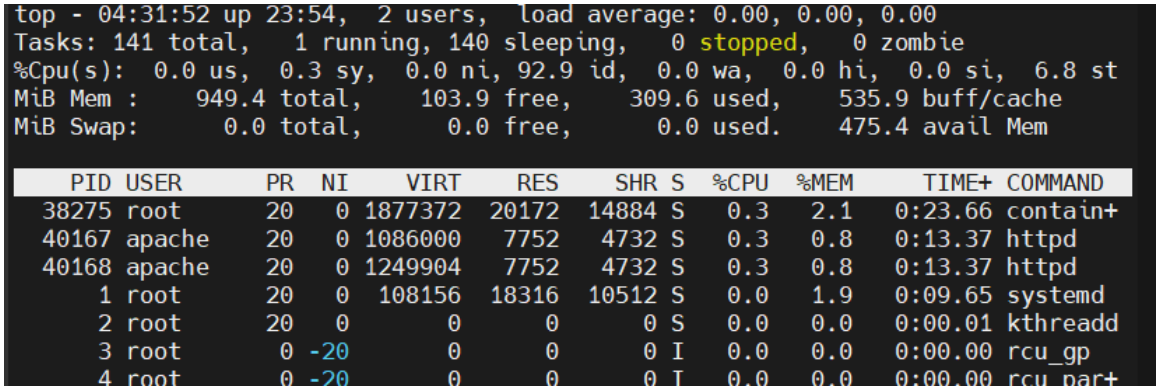

Figura 67 Análisis del consumo de recursos de la instancia.

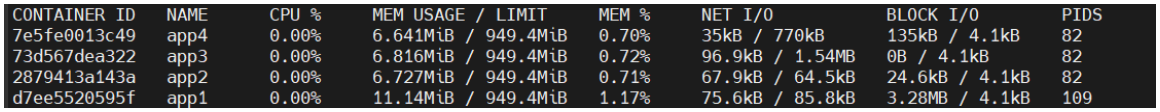

Figura 68 Análisis del consumo de recursos de los contenedores

2. 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.

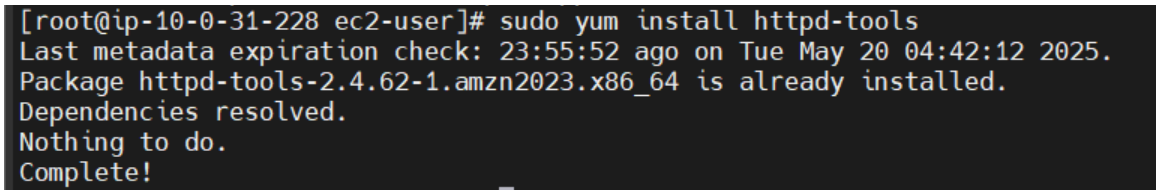

Figura 69 Instalación Apache Benchmark.

3. 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

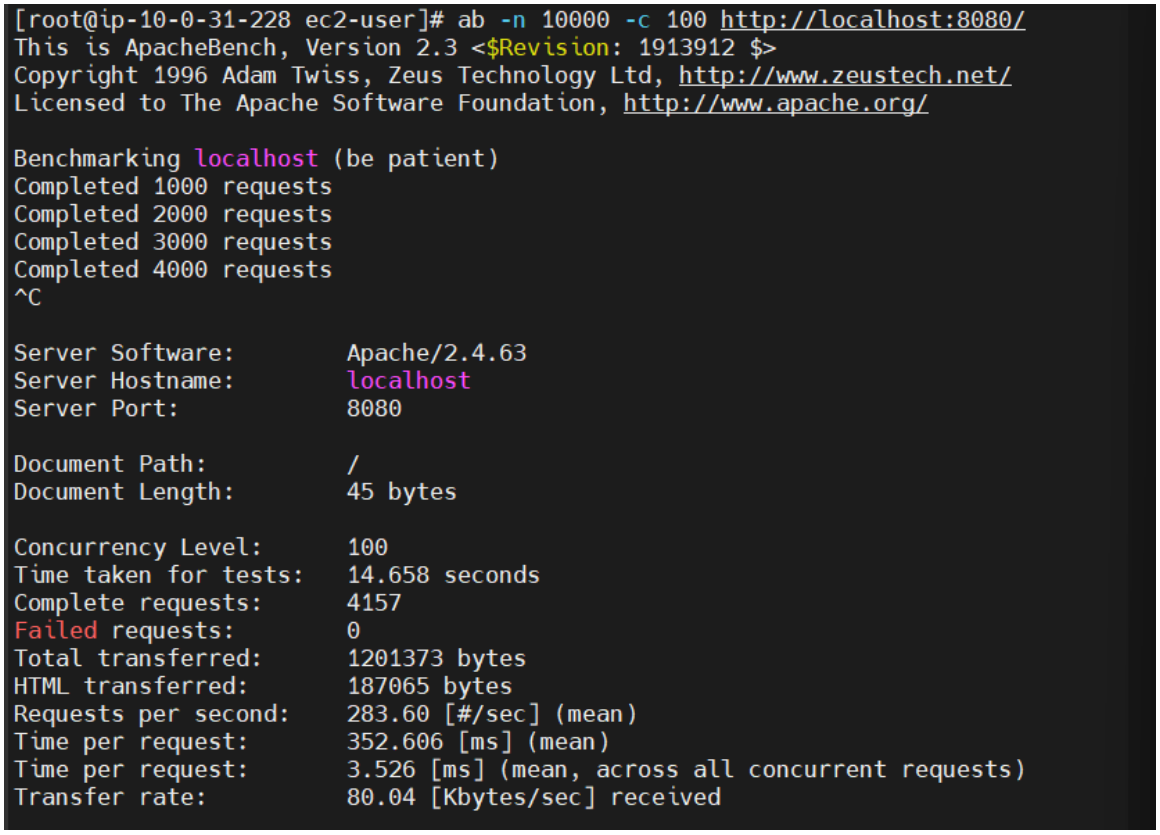

Figura 70 Saturación de solicitudes del contenedor.

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.

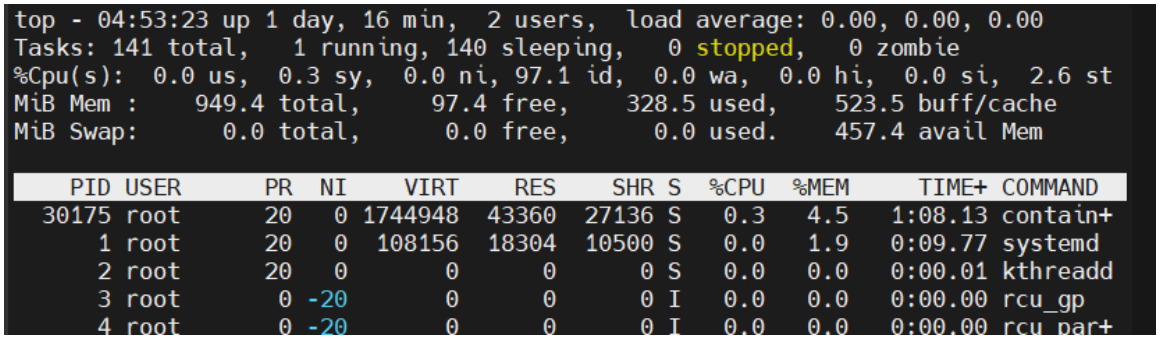

Figura 71 Consumo de los recursos de la instancia.

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.

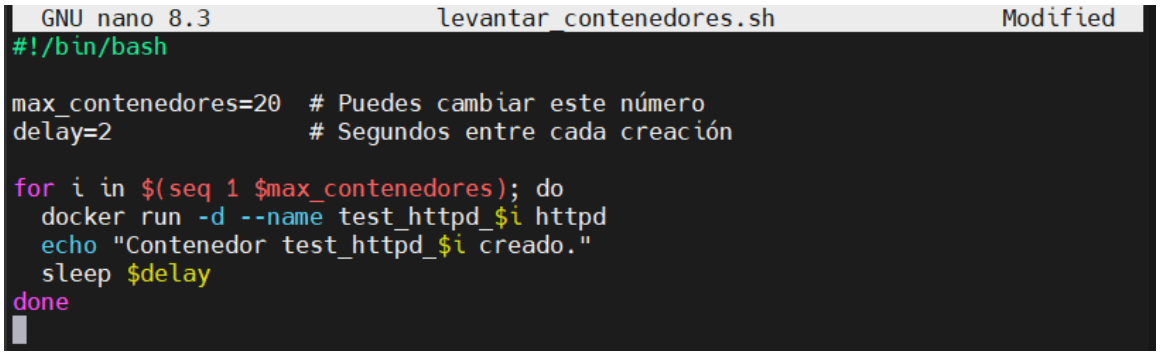

Figura 72 Script de la creación de 20 contenedores.

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 dejo de responder y la terminal quedo 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.

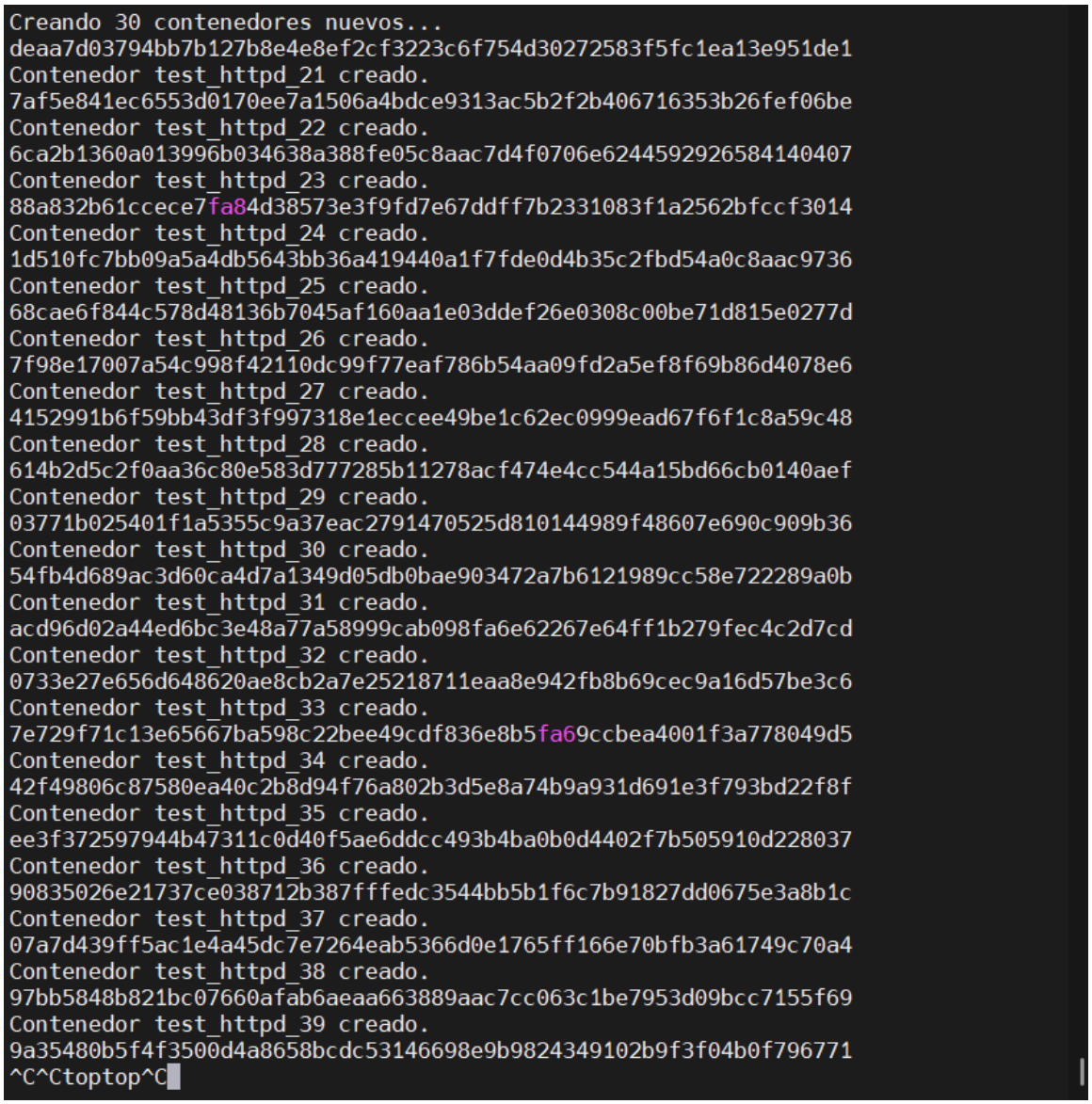

Figura 76 Limite de creación de contenedores

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 eso de CPU con la cantidad de 43 contenedores ejecutándose.

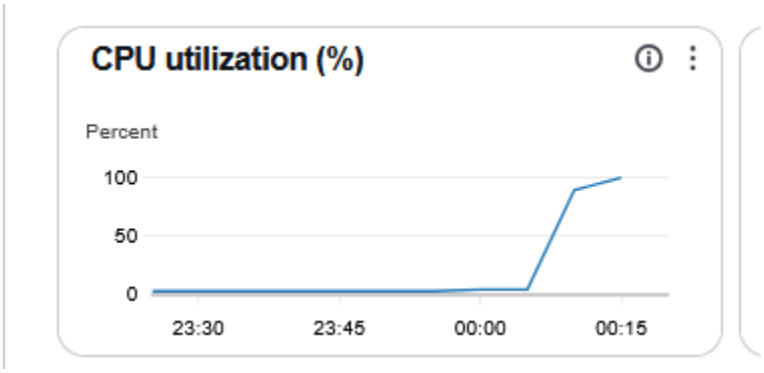

Figura 77 Consumo de CPU máximo.

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