# Implementación de arquitectura en AWS para publicación de sitios Web

**PRESENTADO POR:**

Jorge Orlando Ramírez Agudelo

Juan Pablo Cadavid Monsalve

Jorge Leonardo Álvarez

**DOCENTE:**

Juan Pablo Berrio López

Seminario AWS

Mayo de 2025

## Dedicatoria

Para Juan Pablo Berrio López:

Por su excelente metodología de enseñanza, por su admirable vocación como docente y por su metodología de enseñanza, logró facilitarnos este proceso de aprendizaje e inspirarnos para dar lo mejor de nosotros en este proceso.

Gracias por compartir su conocimiento, ha sido una excelente experiencia.

## Agradecimientos

Queremos expresar nuestra más profunda gratitud a nuestras familias, quienes han sido un pilar fundamental durante todo este proceso.

Gracias por su paciencia, apoyo y palabras de aliento en los momentos de mayor esfuerzo. Su comprensión y confianza nos motivaron a seguir adelante y dar lo mejor de nosotros.

Este logro también es de ustedes.

**Tabla de Contenidos**

## Contenido

## Resumen

El siguiente proyecto tuvo como objetivo principal realizar el diseño e implementación de una red en la plataforma Amazon Web Services (AWS), mediante la creación en una Virtual Private Cloud (VPC), subredes públicas y privadas, direcciones IP dinámicas y grupos de seguridad, y progresivamente instancias EC2 totalmente personalizadas.

La red fue conformada por dos instancias EC2 (tipo t2.micro, gratuita de AWS) que opera bajo el sistema operativo Amazon Linux 2023 y Windows Server 2016, simulando un entorno mixto que permite realizar pruebas de interoperabilidad, conectividad remota (mediante SSH y RDP).

La arquitectura diseñada se encuentra dentro del nivel gratuito de AWS, lo cual facilitó su implementación evitando algún costo de AWS, de uso netamente para fines académicos y de experimentación. Estas instancias fueron configuradas para ofrecer servicios web básicos, permitiendo validar la comunicación y funcionalidad dentro de la red virtual (VPC) creada.

Adicionalmente, se usaron metodologías de contenedores con la librería de Docker; para poder alojar y ejecutar esta aplicación en un entorno controlado, poder tener una gestión de estados de recursos y la escalabilidad del sistema. Esta actividad nos permitió realizar múltiples pruebas relacionadas con la accesibilidad y rendimiento de los servicios con pruebas de estrés y consumo de recursos.

Los resultados obtenidos dieron a entender el correcto funcionamiento de una infraestructura, así como la confianza de integrar servicios en la nube con tecnologías de virtualización para el desarrollo de soluciones modernas y eficientes.

## Marco conceptual

Implementación de arquitectura en AWS para publicación de sitios Web

Este trabajo se construyó con la implementación de tecnologías de computación en la nube (esencialmente AWS), para aumentar la eficiencia en implementaciones, gestión y mejora de infraestructuras.

Entre los servicios utilizados en este proyecto encontramos:

1) EC2 (Elastic Compute Cloud): Creador de máquinas virtuales o instancias en nube
2) VPC (Virtual Private Cloud): Red virtual privada para definir subredes, dirección IP, reglas y mecanismos de seguridad.
3) Grupo de seguridad: A través de los controles de firewall y las claves de acceso se permite poder proteger la integridad de la conexión.
4) Docker: Manejo de contenedores para poder generar entornos independientes de servicios para que sea más eficiente y ágil el consumo de recursos, que nos permite mejorar la escalabilidad de un proyecto.

El uso de lo anteriormente mencionado permite simular entornos de infraestructura TI real, para poder comprender mejor las plataformas y adquirir habilidades de despliegue, configuración y pruebas de servicio en la nube.

## Marco contextual

Este trabajo se realizó como proyecto final del Seminario AWS de la Universidad Remington ofrecido por la faculta de ingeniería, con la intención de integrar nuevos conocimientos y conceptos prácticos de diseño e implementación de redes en la nube.

Como contexto académico se resalta la búsqueda el crecimiento de las fortalezas hacia el manejo de nuevas tecnologías emergentes (Modernas), intentando adoptar una posición de mejora de los procesos continuos.

Estos ejercicios se realizaron en diferentes módulos al nivel GRATUITO de AWS, lo cual nos permitió diseñar nuestras propias instancias (LINUX Y WINDOWS) para publicar servicios en la web, administrarlos de manera versátil y explorar implementación de rendimiento como los Dockers; para cerrar todo este conocimiento técnico, nos dio la oportunidad de entender la importancia de la tecnología en la nube de los proyectos reales en el ámbito laboral y/o profesional.

## Desarrollo e Implementación del aprendizaje

### Diagrama de arquitectura

Representación gráfica de la red (EC2s, subredes, IPs públicas/privadas, grupos de seguridad, VPC, etc.)

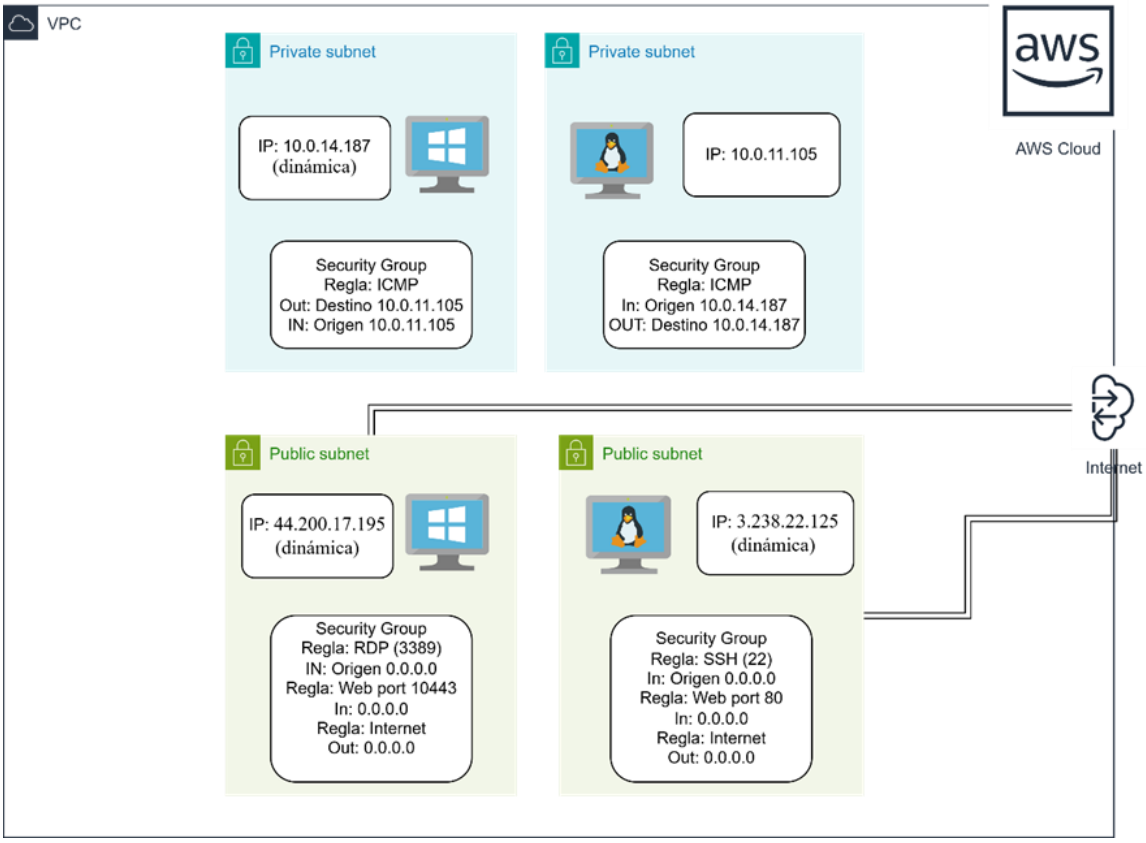



*Ilustración 1*

**Descripción de la arquitectura**

Se diseñó una arquitectura simple en la nube utilizando Amazon Web Services (AWS), compuesta por dos instancias EC2: una con sistema operativo Windows Server 2016 y otra con sistema Linux (Amazon Linux 2023). Ambas instancias se encuentran dentro de una misma red virtual (VPC), permitiendo la comunicación entre ellas. Esta red simula un entorno mixto, útil para pruebas de interoperabilidad, conectividad remota (RDP y SSH), y servicios web desde ambas plataformas.

**Tipo de instancias usadas (Linux: Amazon Linux. Windows: Windows Server).**
Se utilizaron dos tipos de instancias EC2 en AWS:

**Instancia Linux:** Amazon Linux 2023 (tipo t2.micro), utilizada como servidor web básico para alojar contenido HTML.
**Instancia Windows:** Windows Server 2016 (tipo t2.micro), utilizada para acceso remoto (RDP), utilizada como servidor web básico para alojar contenido HTML.

Ambas instancias están dentro del nivel gratuito de AWS, lo cual facilita su implementación sin incurrir en costos adicionales para pruebas básicas.

**Justificación de las configuraciones de red**
Las configuraciones de red fueron generadas automáticamente por AWS al momento de lanzar las instancias. Se creó una VPC por defecto con 2 subredes privadas y 2 públicas, lo que permitió asignar direcciones IP privadas y públicas a ambas instancias. Esto facilitó el acceso remoto mediante protocolos como SSH (puerto 22) para Linux y RDP (puerto 3389) para Windows desde Internet. El grupo de seguridad asociado fue configurado para permitir estas conexiones, garantizando el acceso necesario.

**Configuraciones realizadas**

**Pasos para crear las instancias EC2.**

1. **Asignar nombre:** Se le da un nombre a la instancia para identificarla.
2. **Elegir SO y versión:** Se selecciona el sistema operativo (Linux) y su versión.
3. **Seleccionar tipo de instancia:** Se elige el tipo, como t2.micro para uso gratuito.
4. **Crear Key Pair:** Se genera una clave para acceder a la instancia de forma segura.
5. **Configurar red:**
   - **VPC:** Se crea Red virtual para la interconexión de las instancias.
   - **Grupo de seguridad:** Firewall que controla la interconexión a través de reglas.
   - **Reglas:** Se crean reglas para permitir los puertos 22 (SSH), 3389 (RDP) y 80 (HTTP).
6. **Asignar almacenamiento:** Se define el tamaño del disco, por defecto 8 GB.

**Detalles de los Grupos de Seguridad (puertos abiertos: RDP, SSH, HTTP).**

1. **Windows**

   a. Inbound rules

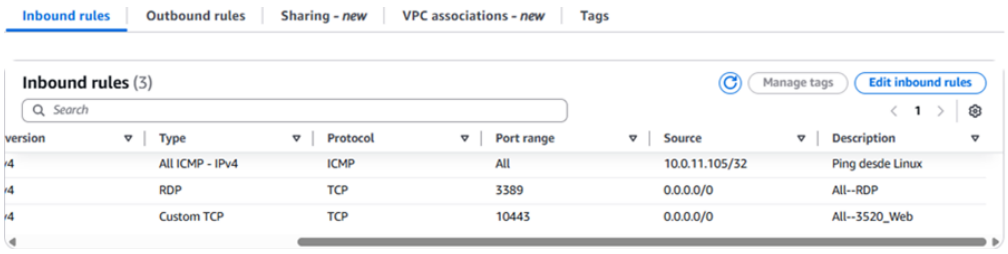



*Ilustración 2*

   b. Outbound rules

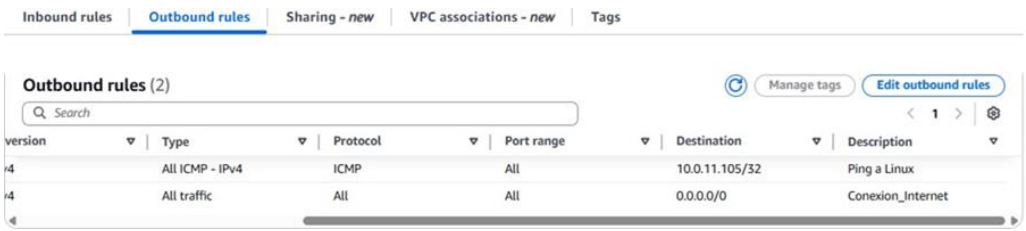



*Ilustración 3*

2. **Linux**

   a. Inbound rules

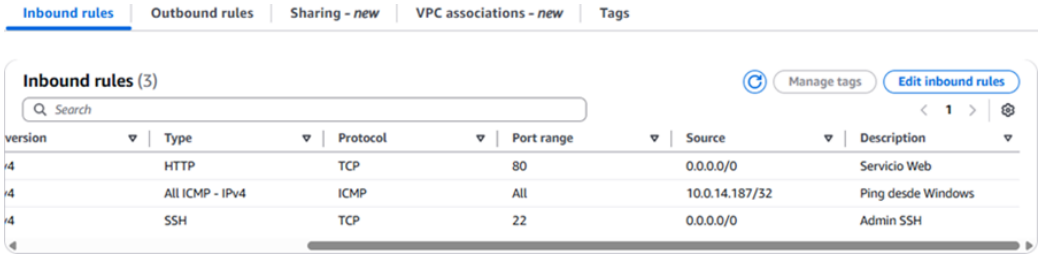



*Ilustración 4*

   b. Outbound rules

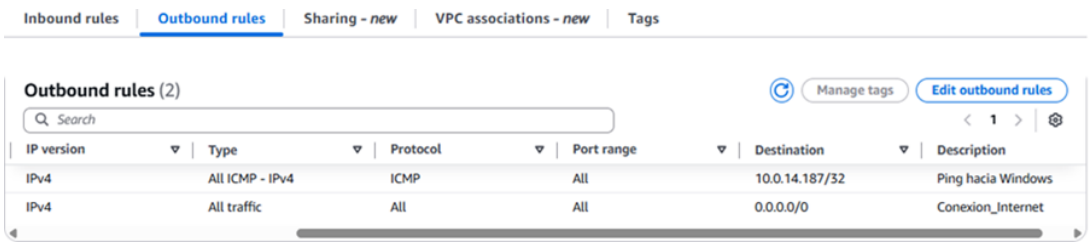



*Ilustración 5*

## Asignación de IPs públicas y privadas.

### 1. Windows

Privada y Pública

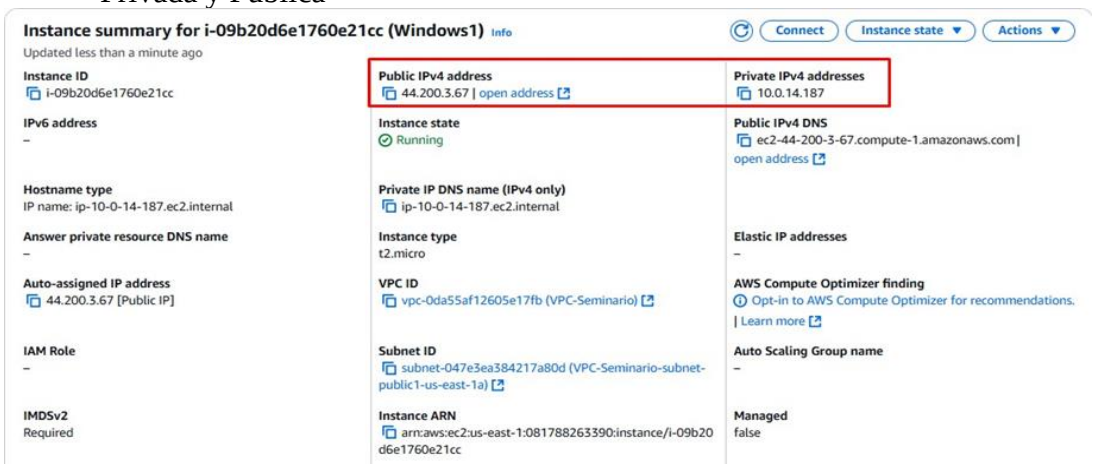



*Ilustración 6*

### 2. Linux

Privada y Pública

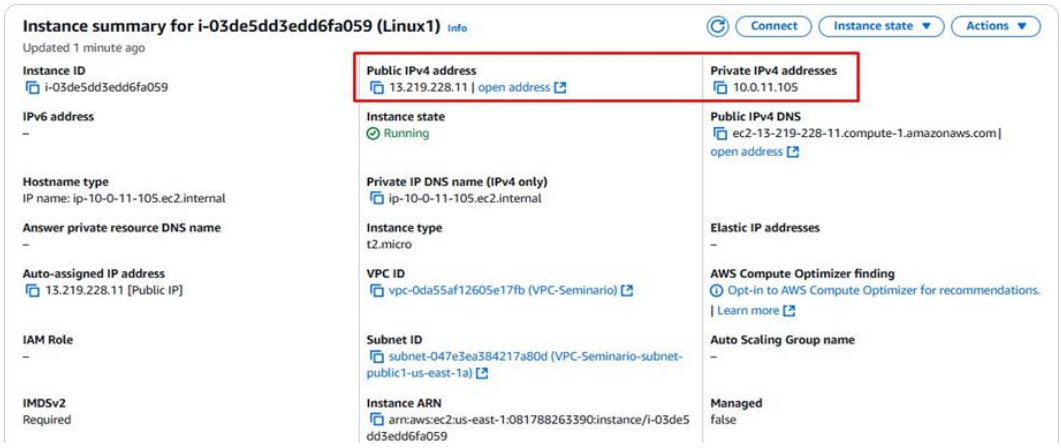



*Ilustración 7*

**Procedimiento de acceso**

1. **Windows**

a) AWS

1. Ve a la consola de AWS en EC2 > Instances
2. Seleccionamos la instancia Windows
3. Obtener la contraseña de administrador
   - Clic en "Connect".
   - Clic en "RDP Client".
   - Clic en "Get Password".
   - Subimos el archivo .pem para descifrar la contraseña.
4. Copiar el usuario (Administrator) y la contraseña generada.

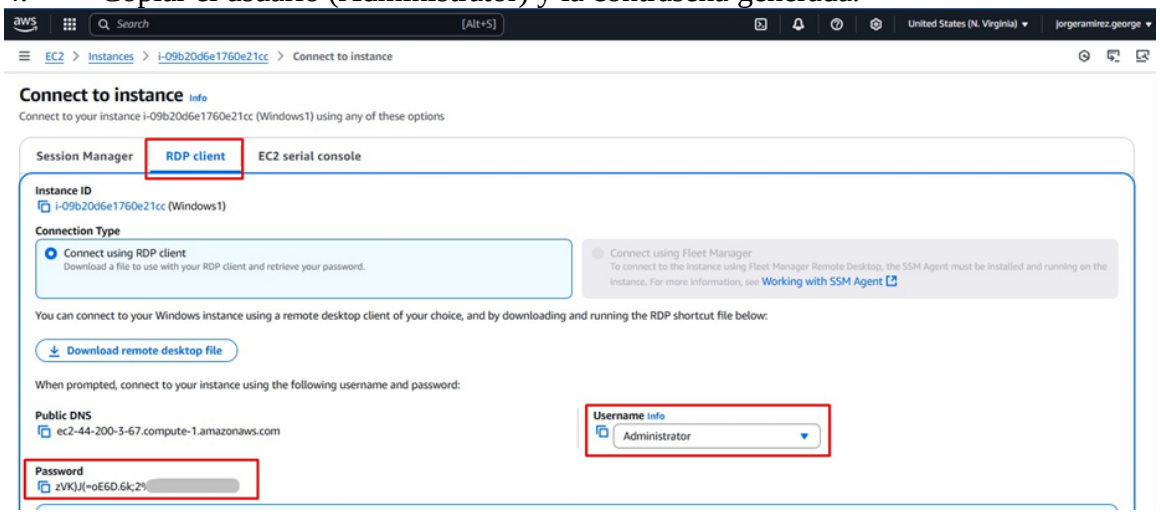



*Ilustración 8*

**b) Cliente RDP**

1. Abrir el programa Conexión a Escritorio Remoto de Windows
2. Escribimos la IP Pública de la instancia y clic en "Conectar".
3. Ingresamos el Usuario: *Administrator*
4. Ingresamos la contraseña: la que copiamos después de descifrarla en AWS.
5. Finalmente, clic en Aceptar.

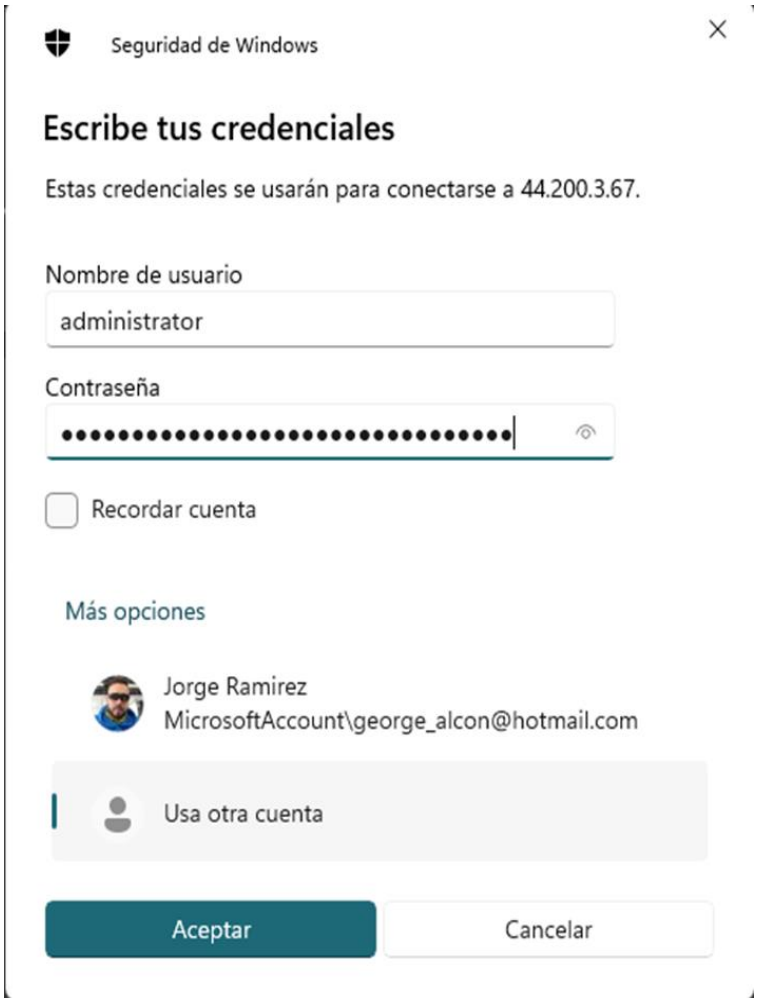


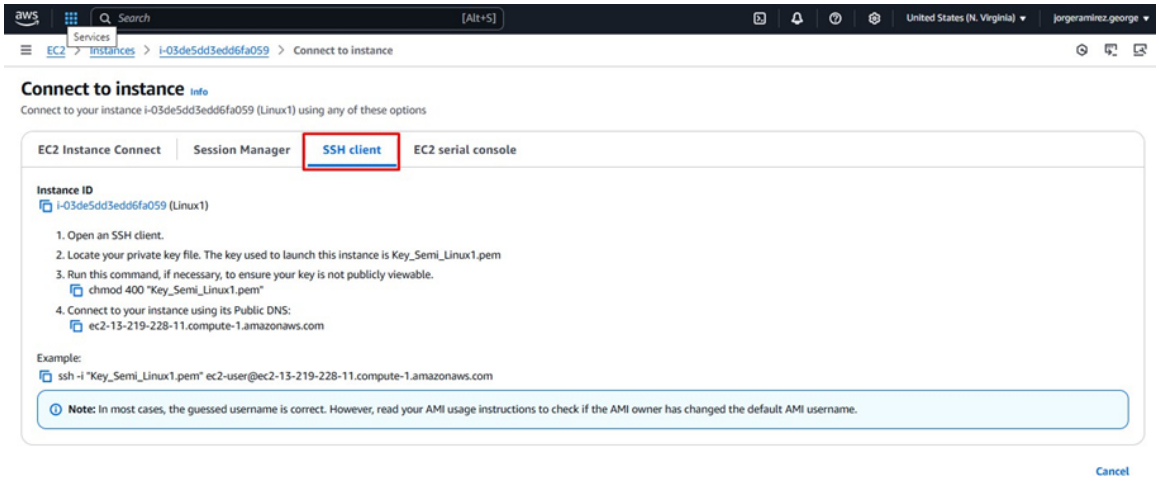

*Ilustración 9*


*Ilustración 10*

## 2. Linux

### a) AWS

1. Ve a la consola de AWS en EC2 > Instances
2. Seleccionamos la instancia Linux
3. Clic en "Connect".
4. Clic en "SSH Client".
5. Copiar el usuario (ec2-user)



**b) Cliente SSH**

1. Abrir MobaXterm
   - Inicia el programa en el PC.
2. Crear una nueva sesión SSH
   - Clic en "Session" en la barra superior.
   - Seleccionamos la opción "SSH".
3. Configurar conexión SSH
   - En el campo Remote host, escribimos la IP pública de la instancia.
   - Marca "Specify username" y escribe el usuario correspondiente: ec2-user para Amazon Linux.
4. Cargar la clave privada (.pem)
   - En la parte izquierda, activar "Use private key".
   - Clic en el ícono de carpeta y selecciona el archivo.pem descargado desde AWS.
5. Conectar
   - Clic en "OK" para iniciar sesión en la instancia.

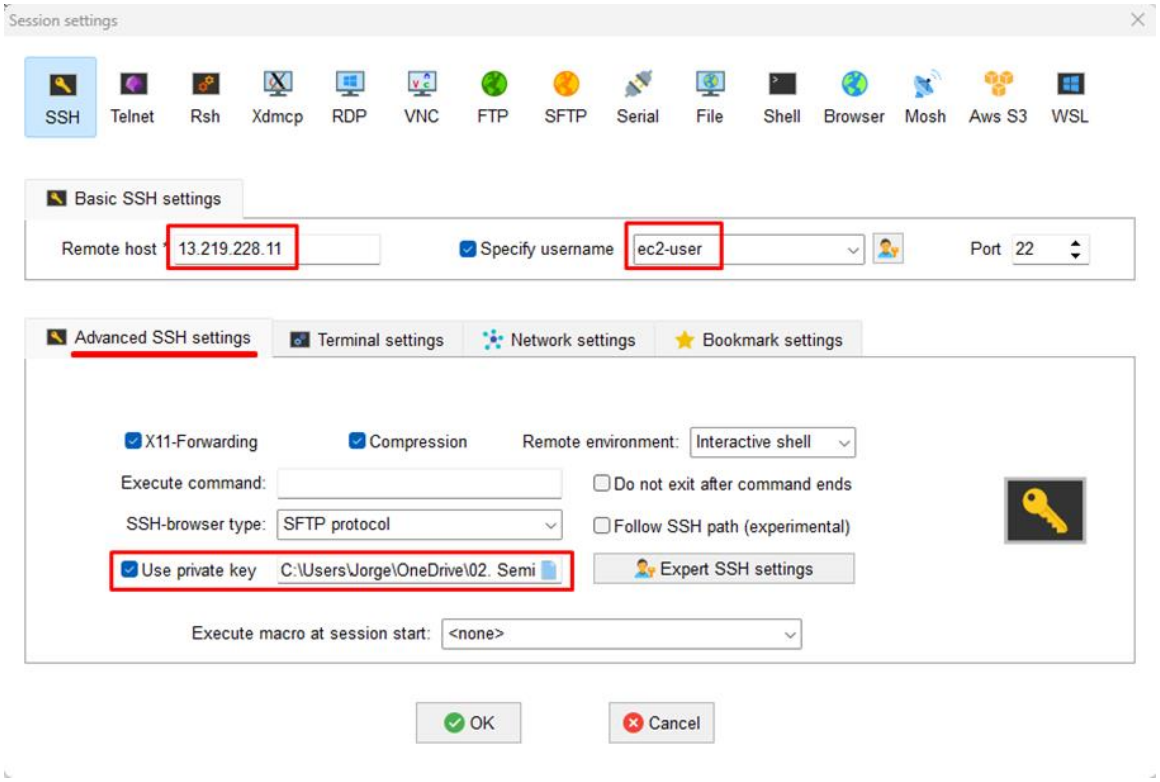



*Ilustración 11*

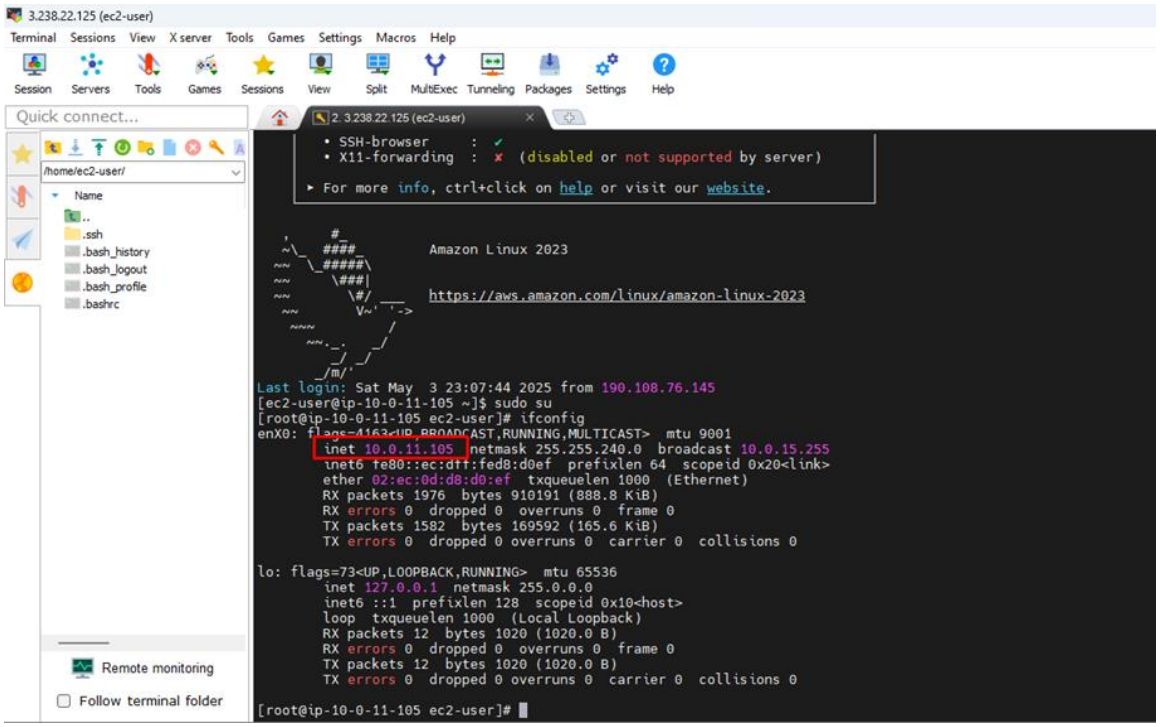



*Ilustración 12*

**Configuración del servidor web**

Pasos seguidos para instalar IIS en Windows Server.

**1. Conectar por RDP:**

a. Ingresa a la instancia EC2 de Windows desde AWS usando Remote Desktop (RDP).

**2. Abrir el Administrador del Servidor:**

a. Una vez dentro, haz clic en "Server Manager" (Administrador del servidor).

**3. Agregar roles y características:**

a. En el panel de Server Manager, haz clic en:

b. "Manage" → "Add Roles and Features".

**4. Seleccionar instalación basada en roles**

a. Marca la opción "Role-based or feature-based installation" y haz clic en Next.

**5. Seleccionar servidor:**

a. Selecciona tu servidor (ya aparecerá listado) y clic en Next.

**6. Elegir rol: Web Server (IIS)**

a. Marca la casilla "Web Server (IIS)".

b. Al salir la ventana emergente, haz clic en "Add Features" y luego en Next.

**7. Confirmar e instalar**

a. Avanza con Next hasta llegar al resumen y haz clic en Install.

b. Espera a que finalice la instalación (toma unos minutos).

**8. Verificar IIS**

a. Abre un navegador en la instancia y entra a http://localhost

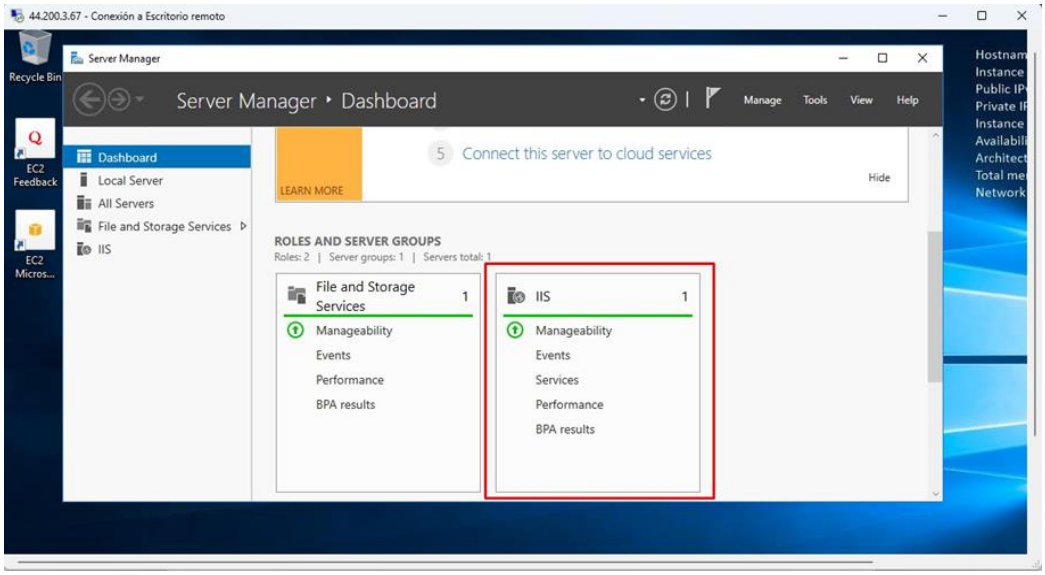



*Ilustración 13*

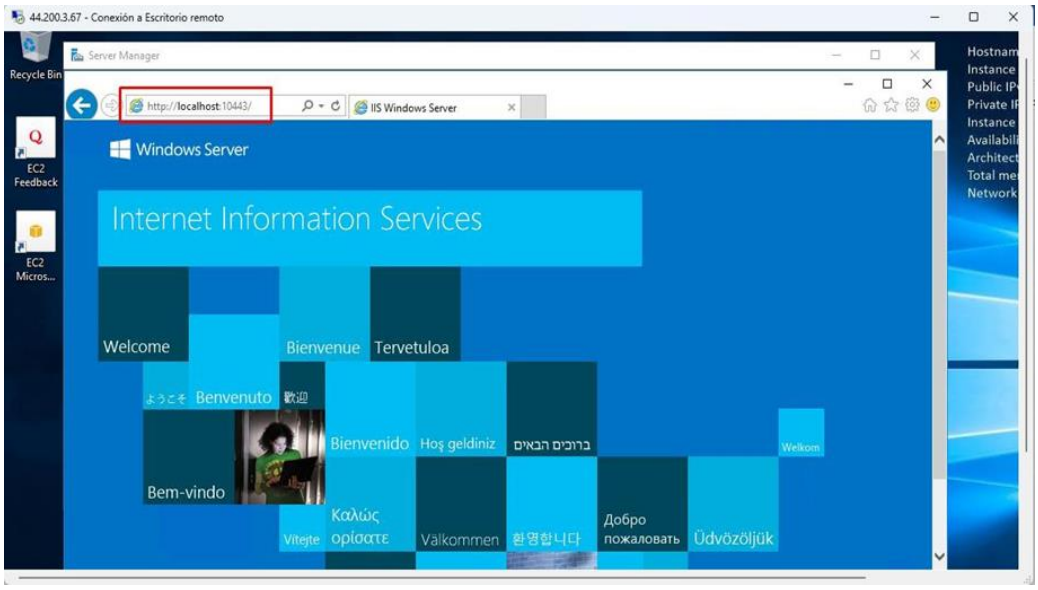



*Ilustración 14*

**Pasos para instalar Apache en Linux.**

1. **sudo su:** Entrar en modo super usuario
2. **dnf install httpd:** Instalar servicio httpd
3. **systemctl status httpd:** Validar estado del servicio httpd
4. **systemctl start httpd:** Iniciar servicio httpd
5. **systemctl status httpd:** Validar que el servios httpd haya iniciado
6. **systemctl enable httpd:** Activar el arranque del servicio httpd en el inicio del Sistema
7. **cd /var/www/html/:** Ruta archivo html
8. **wget https://html5up.net/editorial/download:** Descargar plantilla web desde Internet
9. **ls:** Ver archivo descargado
10. **unzip "download":** Descomprimir archivo descargado "download"
11. **ls:** Ver archivo descomprimido

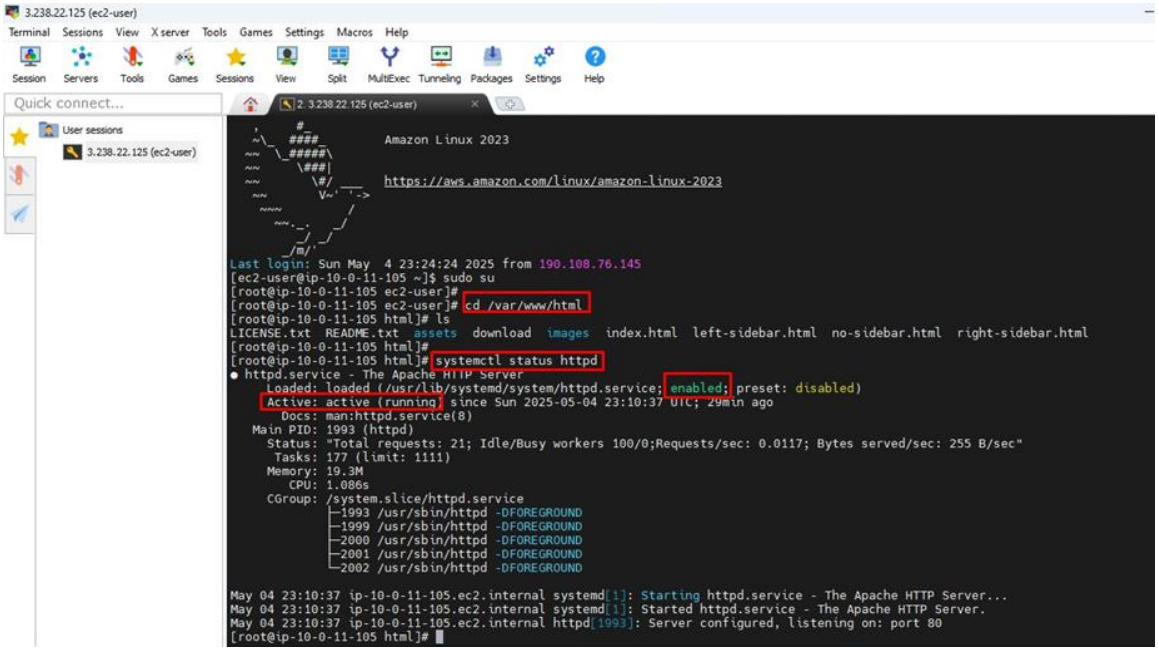



*Ilustración 15*


*Ilustración 16*

Pruebas básicas para verificar que los servidores web son accesibles desde Internet (captura de pantallas del navegador).

**1. Servicio web publicado desde Windows:** http://44.200.3.67:10443/ (*ejemplo*)

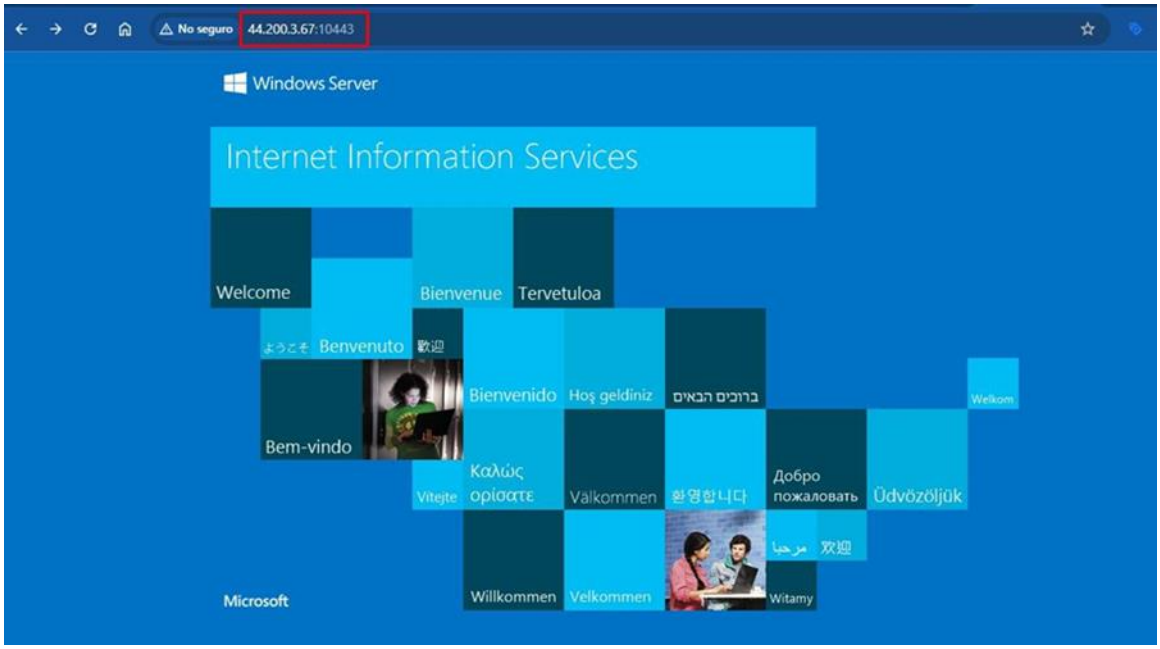



*Ilustración 17*

**2. Servicio web publicado desde Linux:** http://13.219.228.11/ (*ejemplo*)


*Ilustración 18*

## Pruebas de conectividad

### Ping desde Windows a Linux

Para que el Linux responsa Ping desde el Windows por la IP privada, es necesario crear una regla de seguridad saliente en el Windows y una entrante en el Linux.

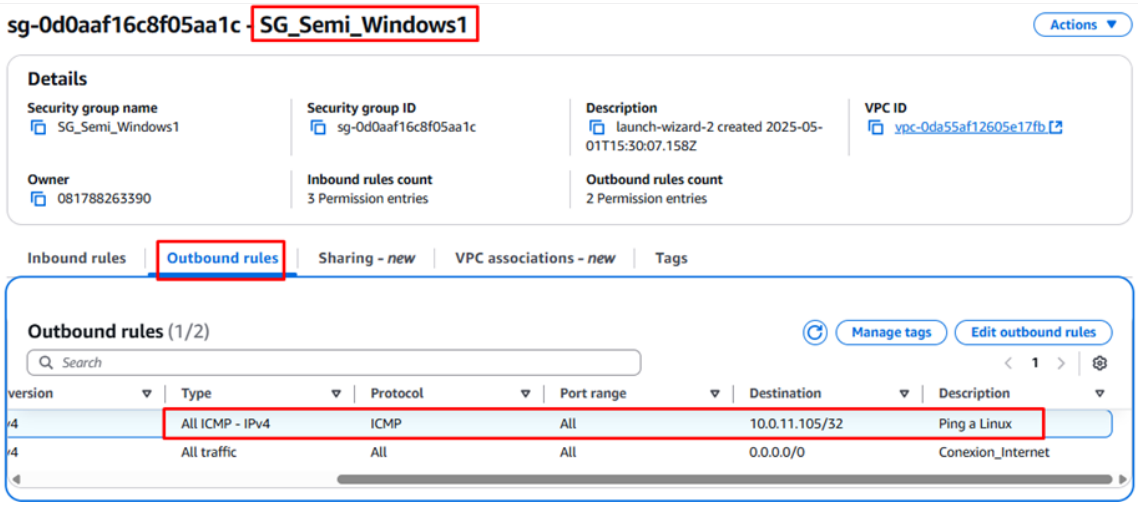



*Ilustración 19*

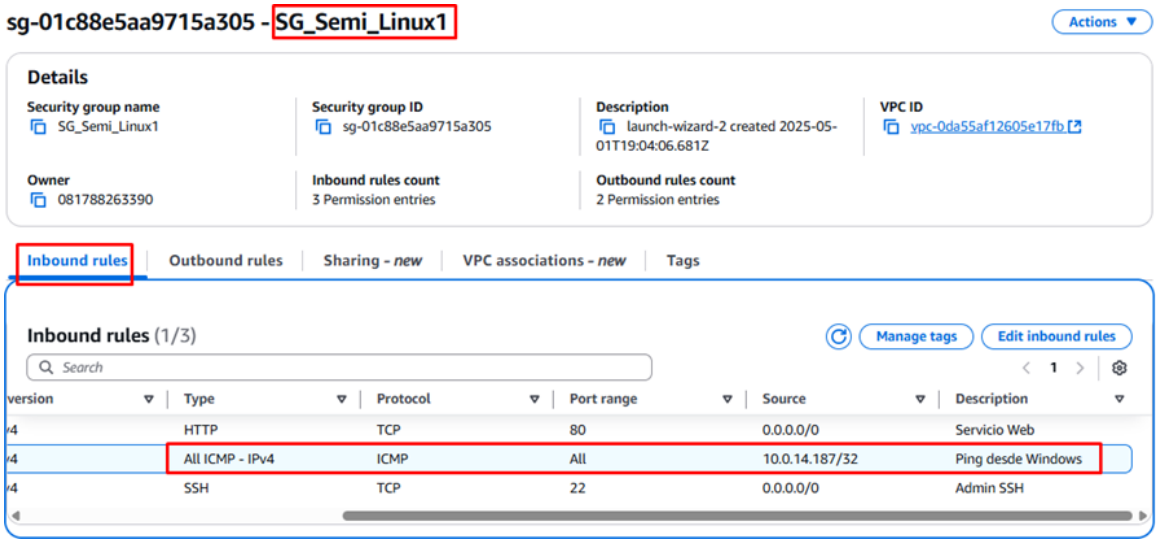



*Ilustración 20*

## Ping desde Linux a Windows

Para que el Windows responda Ping desde el Linux por la IP privada, es necesario crear una regla de seguridad saliente en el Linux y una entrante en el Windows.

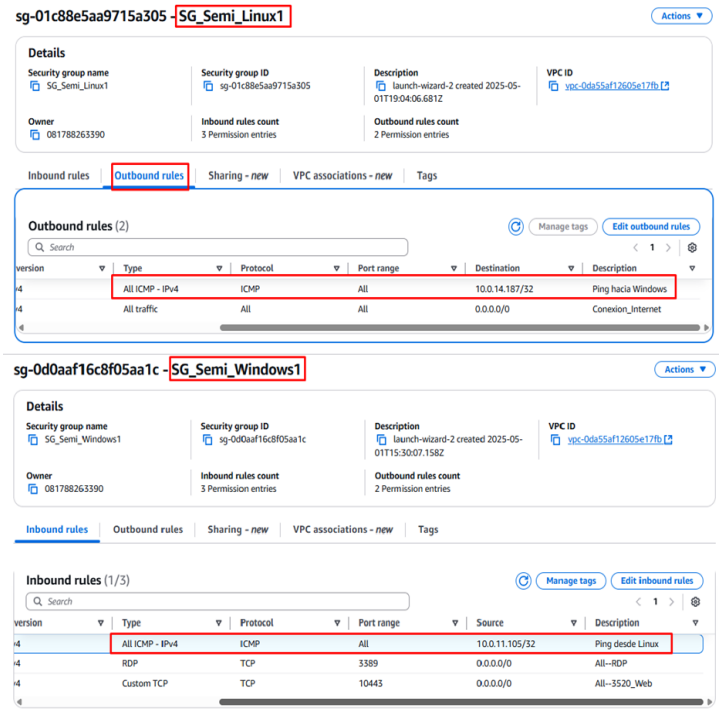



*Ilustración 21*

Adicionalmente, para que el Windows responda a Ping, se requiere crear una regla en el Firewall interno de Windows.

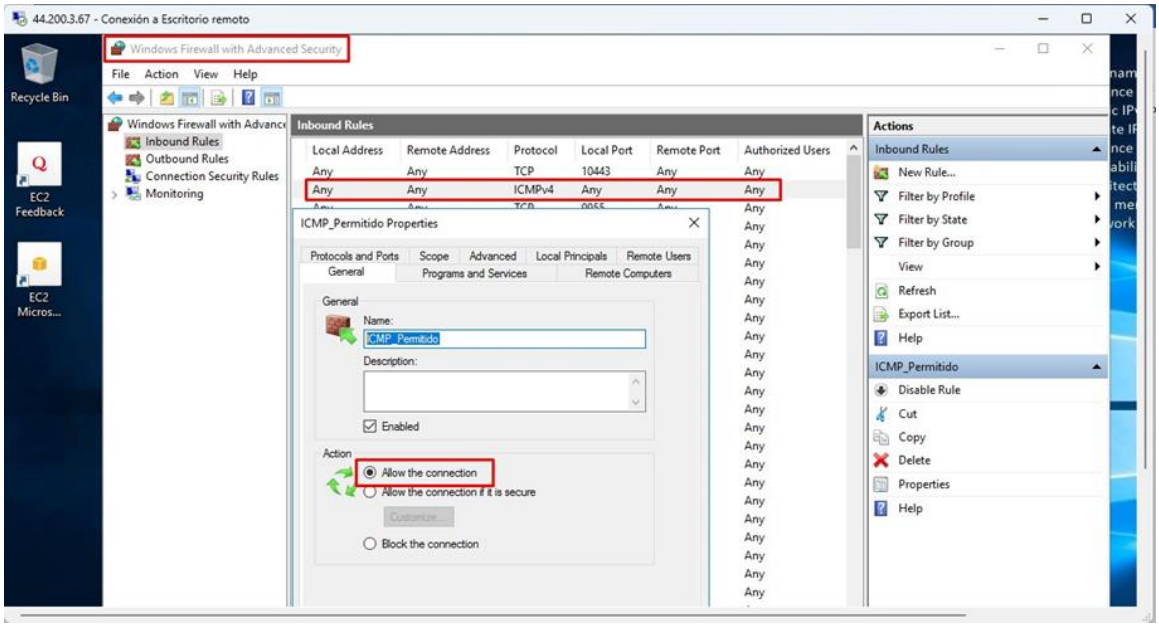



*Ilustración 22*

# PRUEBAS DOCKER

## Que es Docker:

Este es un servicio o librería que permite crear, ejecutar y administrar servicios (aplicaciones) dentro de contenedores, estos terminan siendo entornos ligeros y portables que se apropian de lo necesario para ejecutar un servicio: sea código, algunas dependencias, librerías y/o configuraciones.

***NOTA:*** Generalmente los contenedores se componen de AMIS o IMG

## Qué son las Máquinas Virtuales (VMs):

Estas son las que se ejecutan con herramientas externas como VirtualBox, VMware o directamente en Linux con comandos Sudo, funcionan emulando un sistema operativo, lo que a la final genera más consumo de recursos.

## Diferencias clave entre Docker y Máquinas virtuales

Características para diferenciar ambas tecnologías

| **Características** | **Docker (Contenedores)** | **Máquinas Virtuales (VMs)** |
|---|---|---|
| Nivel de virtualización | A nivel del sistema operativo (usa el mismo kernel) | A nivel del hardware |
| Tamaño | Ligero (MB) | Pesado (GB) |
| Arranque | Rápido (segundos) | Lento (minutos) |
| Aislamiento | Moderado (comparten kernel) | Alto (cada VM es completamente aislada) |
| Consumo de recursos | Bajo (sin duplicar SO) | Alto (cada VM necesita RAM y CPU para el SO) |
| Portabilidad | Muy portable (misma imagen funciona en varios SO) | Menos portable (depende del hipervisor/SO) |

**Ventajas del Docker frente a las Máquinas virtuales.**

1. **El Rendimiento más alto**:
   Al no tener que arrancar un sistema operativo completo, los contenedores consumen menos CPU, memoria y disco.

2. **El Despliegue más rápido**:
   Los contenedores pueden iniciarse en segundos.

3. **La Mayor densidad**:
   Deja correr más contenedores en un mismo host, mientras que las VM´s son más limitadas por el consumo de recursos tan elevado.

4. **La Portabilidad**:
   Los contenedores funcionan igual en desarrollo, pruebas y producción (siempre que Docker esté instalado).

5. **La Menor sobrecarga administrativa**:
   Menos mantenimiento de sistemas operativos internos.

**Procedimiento de un Docker**

1) Instalamos Docker con el comando "*dnf install docker*".

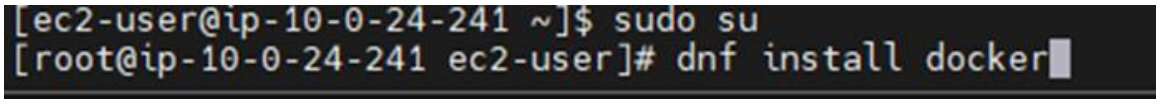



*Ilustración 23*

2) Confirmamos el estado del servicio Docker con el comando "*systemctl status docker*" y buscamos que esté (activo), en el siguiente estado no lo visualizamos de esa forma por lo tanto procedemos con el punto #3.


*Ilustración 24*

3) A continuación, arrancamos el servicio con el comando "*systemctl start docker*".

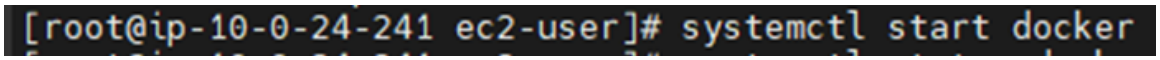



*Ilustración 25*

4) Volvemos a validar el estado del servicio con el mismo comando del punto #2 "*systemctl status docker*".

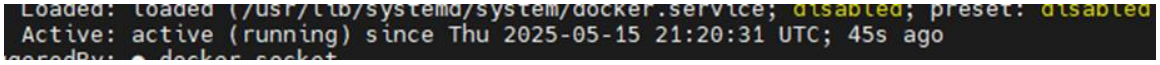



*Ilustración 26*

5) Para crear la información interna del Docker debemos generarla a través de dockerfile (las IMG), ingresamos dentro de las IMG y extraemos su "nombre".

6) Apuntamos el *docker pull httpd* con el comando al frente de los datos de la IMG.


*Ilustración 27*

7) Para descargarlo escribimos el comando obtenido en la consola LINUX.

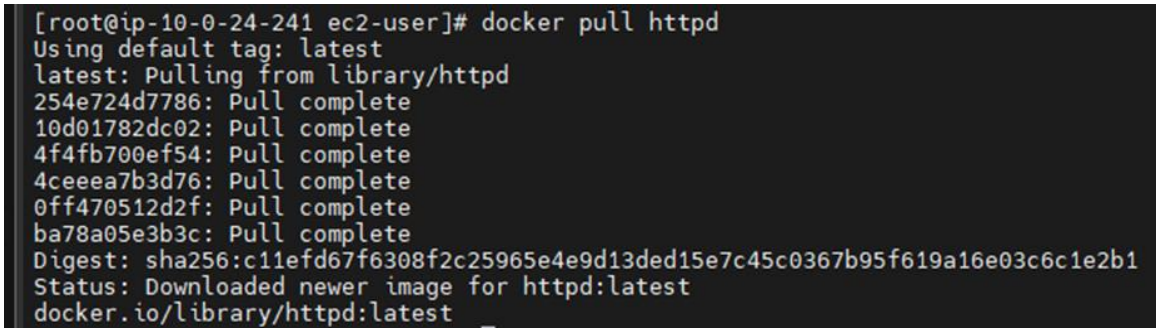



*Ilustración 28*

8) Validamos que si se haya descargado con *docker images*.

```
[root@ip-10-0-24-241 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    0208f149a449   3 months ago   148MB
```

*Ilustración 29*

9) Descargamos otra imagen también de Nginx.

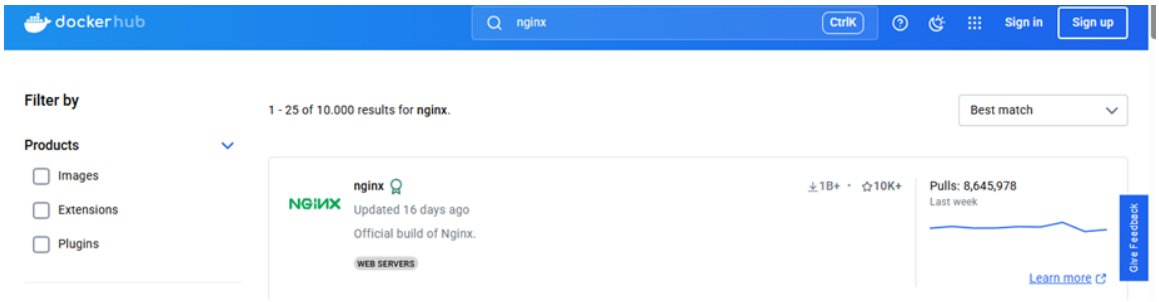



*Ilustración 30*

10) Miramos el comando de la misma manera que el anterior y lo ejecutamos.


*Ilustración 31*

```
[root@ip-10-0-24-241 ec2-user]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
254e724d7786: Already exists
913115292750: Pull complete
3e544d53ce49: Pull complete
4f21ed9ac0c0: Pull complete
d38f2ef2d6f2: Pull complete
40a6e9f4e456: Pull complete
d3dc5ec71e9d: Pull complete
Digest: sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

*Ilustración 32*

11) Ya con las imágenes descargadas ejecutamos "*docker ps*" para ver los contenedores creados.


*Ilustración 33*

12) Si no hay contenedores lo creamos como un "*demonio*" que es una ejecución que está en segundo plano, de la siguiente forma:
-dit (esta es una variable) –name -p (definimos el puerto)
estas nos retornan un hash de verificación.

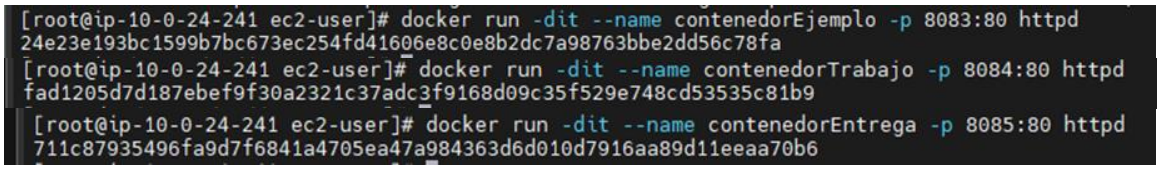



*Ilustración 34*

13) Visualizamos el contenedor con *docker ps*.

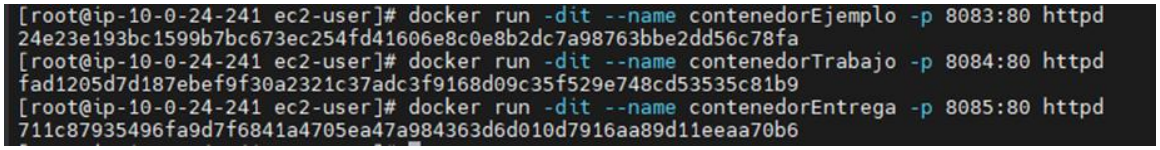



*Ilustración 35*

14) Configuramos el puerto 8080 para poder tener accesos desde cualquier IP.

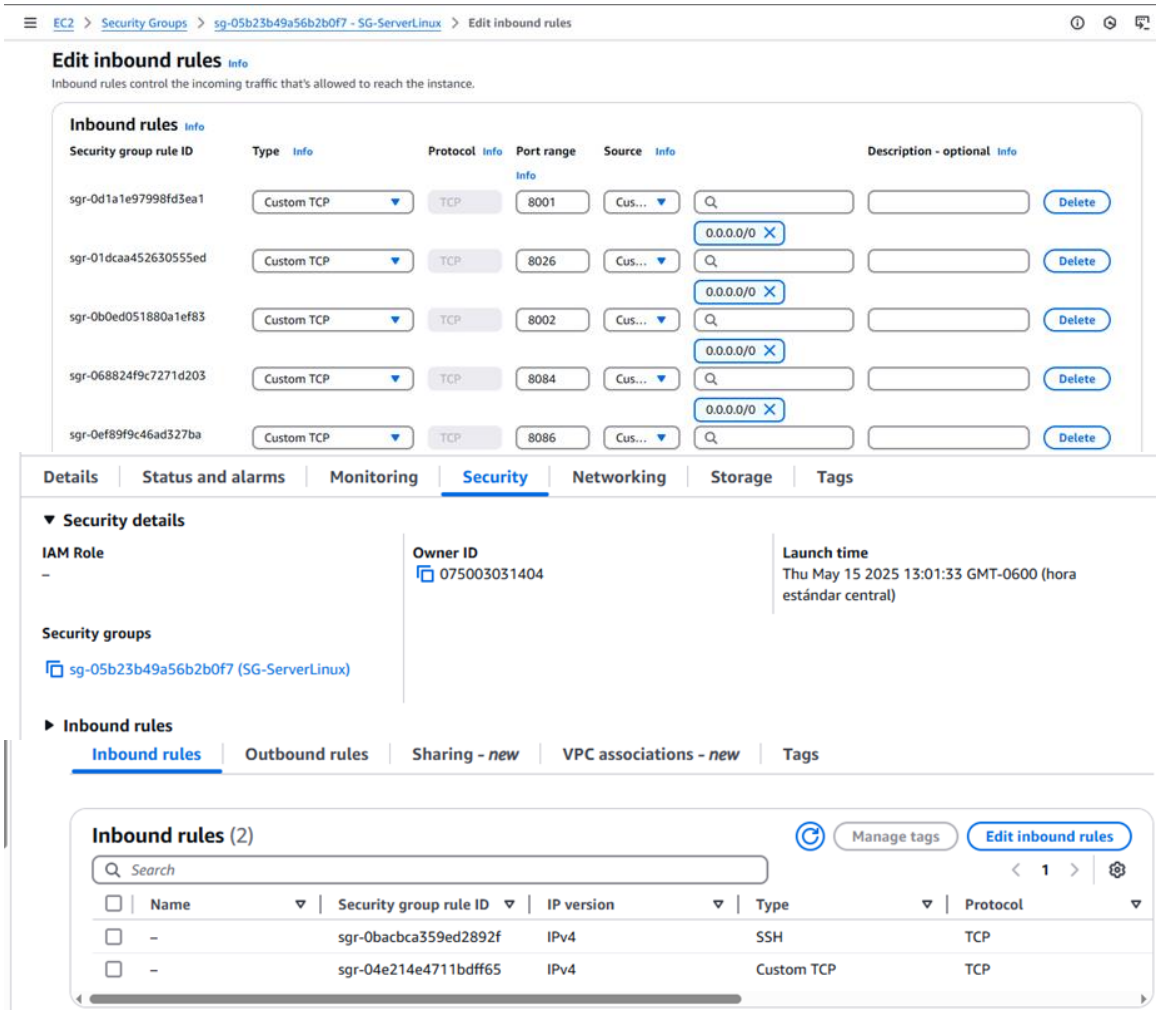



*Ilustración 36*

15) Validamos en Internet con cualquiera de los puertos creados.

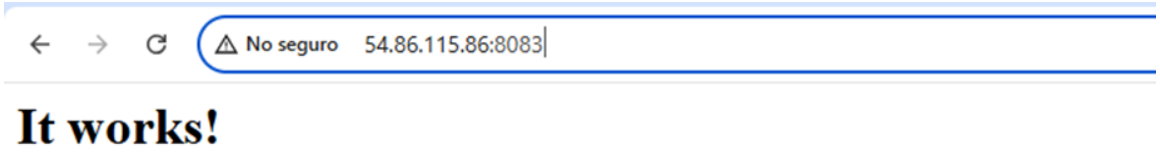



*Ilustración 37*

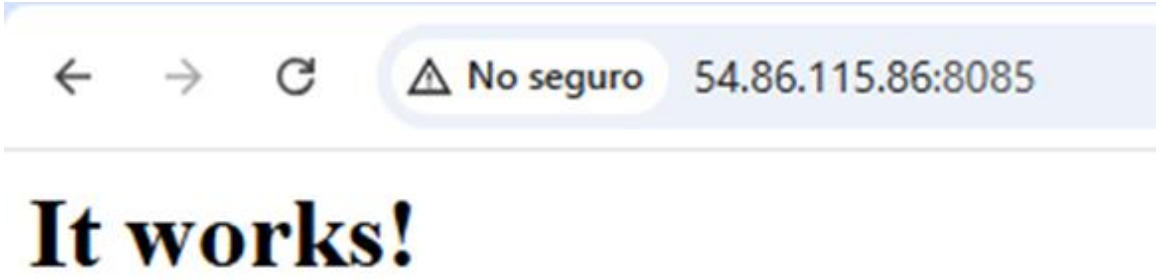



*Ilustración 38*

16) Confirmamos el consumo de uso de CPU y RAM con el comando "*top*".

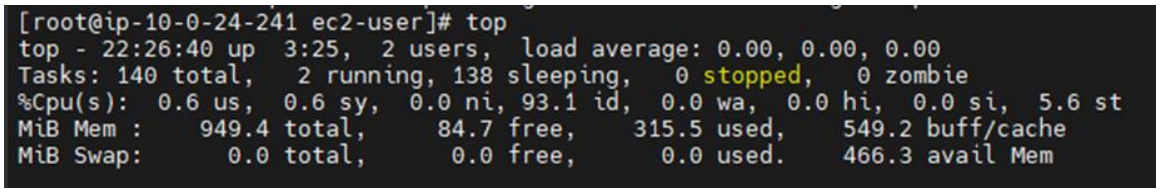



*Ilustración 39*

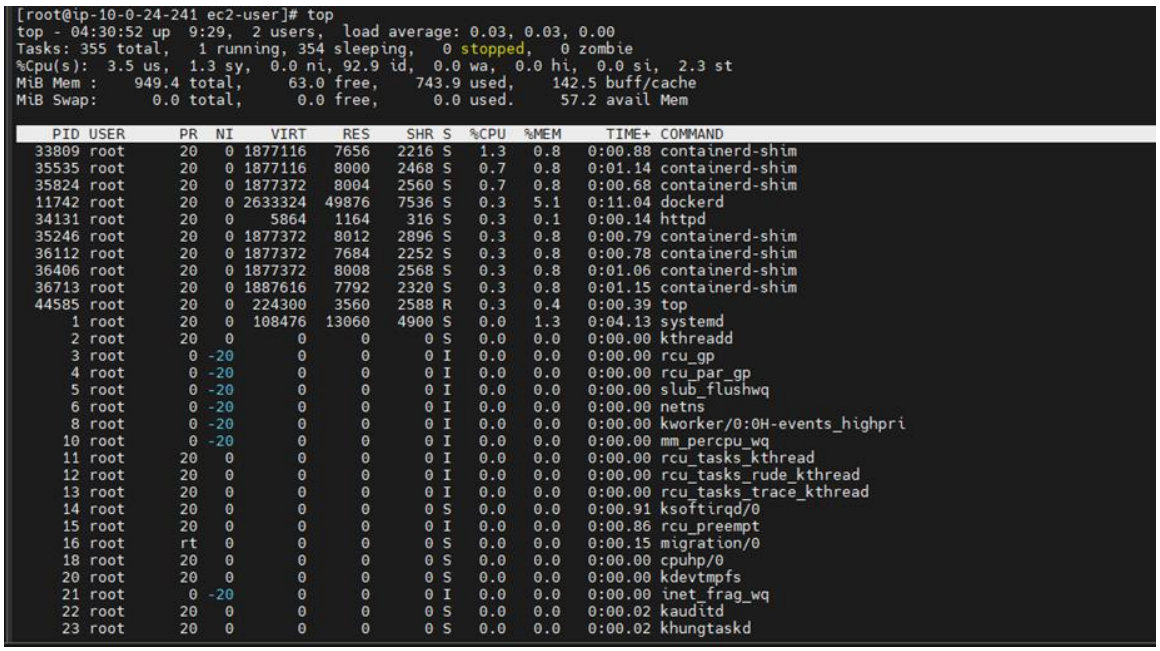



*Ilustración 40*

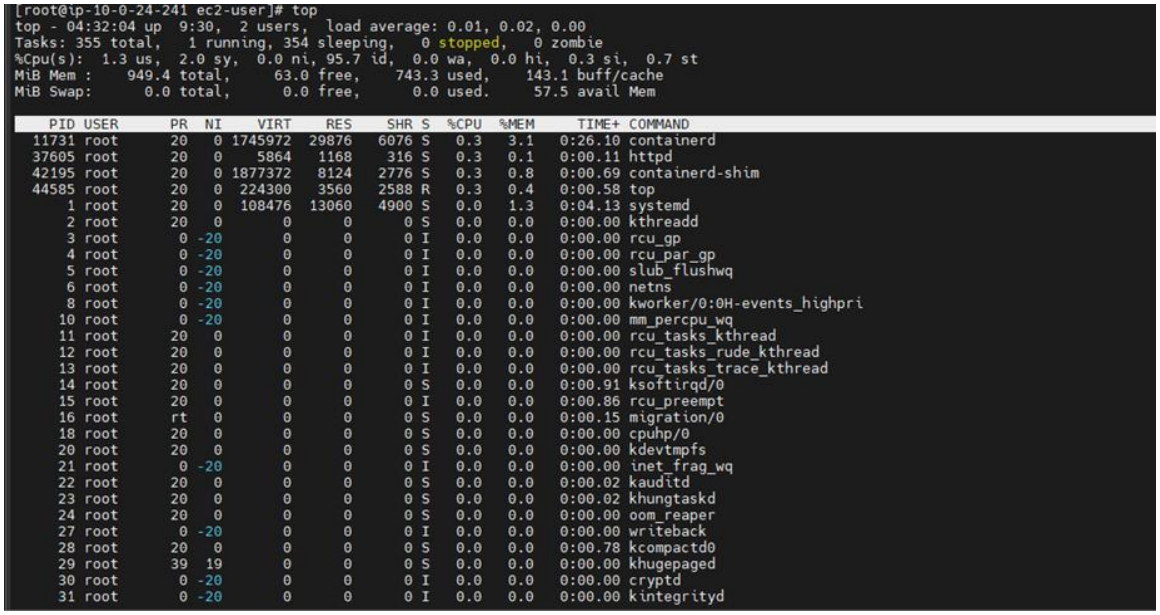



*Ilustración 41*

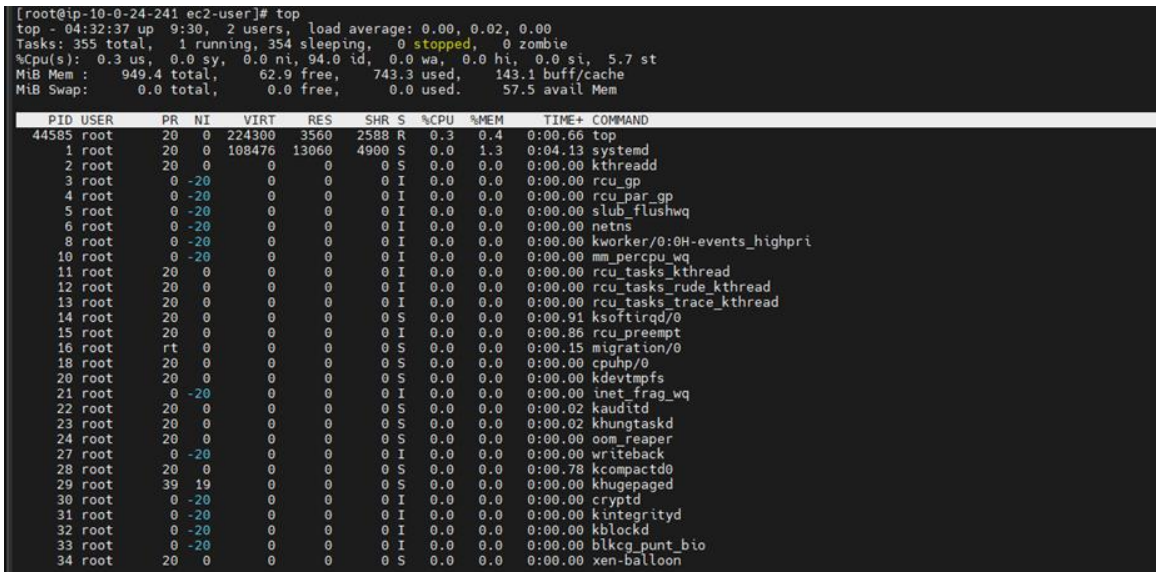



*Ilustración 42*

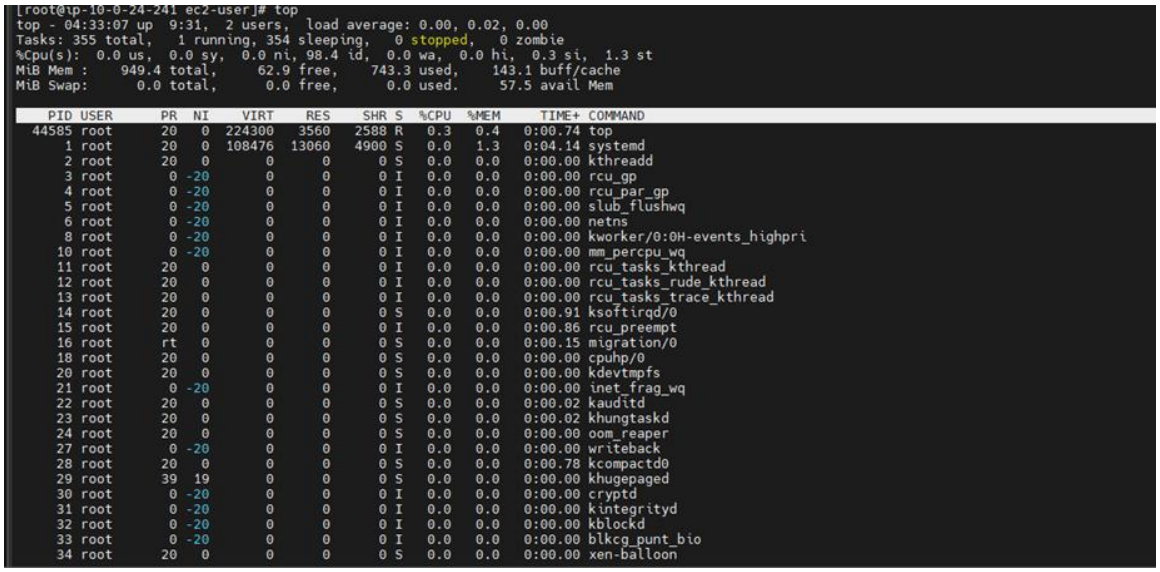
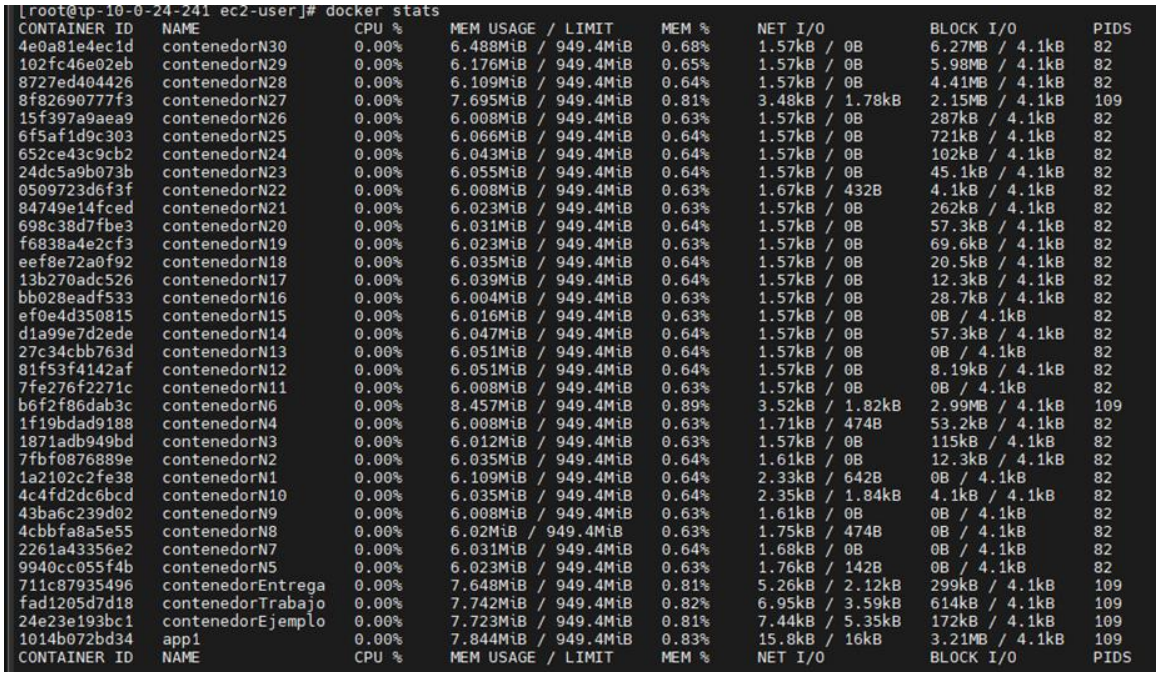



*Ilustración 43*

17) Vemos el uso que tienen los contenedores con el comando "*docker stats"*.


*Ilustración 44*



```
[root@ip-10-0-24-241 ec2-user]# docker stats
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
4e0a81e4ec1d   contenedorN30     0.00%     6.488MiB / 949.4MiB   0.68%     1.57kB / 0B       6.27MB / 4.1kB    82
102fc46e02eb   contenedorN29     0.00%     6.176MiB / 949.4MiB   0.65%     1.57kB / 0B       5.98MB / 4.1kB    82
8727ed404426   contenedorN28     0.00%     6.109MiB / 949.4MiB   0.64%     1.57kB / 0B       4.41MB / 4.1kB    82
8f82690777f3   contenedorN27     0.00%     7.695MiB / 949.4MiB   0.81%     3.48kB / 1.78kB   2.15MB / 4.1kB    109
15f397a9aea9   contenedorN26     0.00%     6.008MiB / 949.4MiB   0.63%     1.57kB / 0B       287kB / 4.1kB     82
6f5af1d9c303   contenedorN25     0.00%     6.066MiB / 949.4MiB   0.64%     1.57kB / 0B       721kB / 4.1kB     82
652ce43c9cb2   contenedorN24     0.00%     6.043MiB / 949.4MiB   0.64%     1.57kB / 0B       102kB / 4.1kB     82
24dc5a9b073b   contenedorN23     0.00%     6.055MiB / 949.4MiB   0.64%     1.57kB / 0B       45.1kB / 4.1kB    82
0509723d6f3f   contenedorN22     0.00%     6.008MiB / 949.4MiB   0.63%     1.67kB / 432B     4.1kB / 4.1kB     82
84749e14fced   contenedorN21     0.00%     6.023MiB / 949.4MiB   0.63%     1.57kB / 0B       262kB / 4.1kB     82
698c38d7fbe3   contenedorN20     0.00%     6.031MiB / 949.4MiB   0.64%     1.57kB / 0B       57.3kB / 4.1kB    82
f6838a4e2cf3   contenedorN19     0.00%     6.023MiB / 949.4MiB   0.63%     1.57kB / 0B       69.6kB / 4.1kB    82
eef8e72a0f92   contenedorN18     0.00%     6.035MiB / 949.4MiB   0.64%     1.57kB / 0B       20.5kB / 4.1kB    82
13b270adc526   contenedorN17     0.00%     6.039MiB / 949.4MiB   0.64%     1.57kB / 0B       12.3kB / 4.1kB    82
bb028eadf533   contenedorN16     0.00%     6.004MiB / 949.4MiB   0.63%     1.57kB / 0B       28.7kB / 4.1kB    82
ef0e4d350815   contenedorN15     0.00%     6.016MiB / 949.4MiB   0.63%     1.57kB / 0B       0B / 4.1kB        82
d1a99e7d2ede   contenedorN14     0.00%     6.047MiB / 949.4MiB   0.64%     1.57kB / 0B       57.3kB / 4.1kB    82
27c34cbb763d   contenedorN13     0.00%     6.051MiB / 949.4MiB   0.64%     1.57kB / 0B       0B / 4.1kB        82
81f53f4142af   contenedorN12     0.00%     6.051MiB / 949.4MiB   0.64%     1.57kB / 0B       8.19kB / 4.1kB    82
7fe276f2271c   contenedorN11     0.00%     6.008MiB / 949.4MiB   0.63%     1.57kB / 0B       0B / 4.1kB        82
b6f2f86dab3c   contenedorN6      0.00%     8.457MiB / 949.4MiB   0.89%     3.52kB / 1.82kB   2.99MB / 4.1kB    109
1f19bdad9188   contenedorN4      0.00%     6.008MiB / 949.4MiB   0.63%     1.71kB / 474B     53.2kB / 4.1kB    82
1871adb949bd   contenedorN3      0.00%     6.012MiB / 949.4MiB   0.63%     1.57kB / 0B       115kB / 4.1kB     82
7fbf0876889e   contenedorN2      0.00%     6.035MiB / 949.4MiB   0.64%     1.61kB / 0B       12.3kB / 4.1kB    82
1a2102c2fe38   contenedorN1      0.00%     6.109MiB / 949.4MiB   0.64%     2.33kB / 642B     0B / 4.1kB        82
4c4fd2dc6bcd   contenedorN10     0.00%     6.035MiB / 949.4MiB   0.64%     2.35kB / 1.84kB   4.1kB / 4.1kB     82
43ba6c239d02   contenedorN9      0.00%     6.008MiB / 949.4MiB   0.63%     1.61kB / 0B       0B / 4.1kB        82
4cbbfa8a5e55   contenedorN8      0.00%     6.02MiB / 949.4MiB    0.63%     1.75kB / 474B     0B / 4.1kB        82
2261a43356e2   contenedorN7      0.00%     6.031MiB / 949.4MiB   0.64%     1.68kB / 0B       0B / 4.1kB        82
9940cc055f4b   contenedorN5      0.00%     6.023MiB / 949.4MiB   0.63%     1.76kB / 142B     0B / 4.1kB        82
711c87935496   contenedorEntrega 0.00%     7.648MiB / 949.4MiB   0.81%     5.26kB / 2.12kB   299kB / 4.1kB     109
fad1205d7d18   contenedorTrabajo 0.00%     7.742MiB / 949.4MiB   0.82%     6.95kB / 3.59kB   614kB / 4.1kB     109
24e23e193bc1   contenedorEjemplo 0.00%     7.723MiB / 949.4MiB   0.81%     7.44kB / 5.35kB   172kB / 4.1kB     109
1014b072bd34   app1              0.00%     7.844MiB / 949.4MiB   0.83%     15.8kB / 16kB     3.21MB / 4.1kB    109
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
```

*Ilustración 45*

**Pasos Para Detener el Docker**

1) Buscamos el Id del contenedor con el comando "*docker ps*".

```
[root@ip-10-0-24-241 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS             PORTS                                   NAMES
4e0a81e4ec1d   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8030->80/tcp, :::8030->80/tcp   contenedorN30
102fc46e02eb   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8029->80/tcp, :::8029->80/tcp   contenedorN29
8727ed404426   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8028->80/tcp, :::8028->80/tcp   contenedorN28
8f82690777f3   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8027->80/tcp, :::8027->80/tcp   contenedorN27
15f397a9aea9   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8026->80/tcp, :::8026->80/tcp   contenedorN26
6f5af1d9c303   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8025->80/tcp, :::8025->80/tcp   contenedorN25
652ce43c9cb2   httpd     "httpd-foreground"       55 minutes ago      Up 55 minutes      0.0.0.0:8024->80/tcp, :::8024->80/tcp   contenedorN24
24dc5a9b073b   httpd     "httpd-foreground"       56 minutes ago      Up 55 minutes      0.0.0.0:8023->80/tcp, :::8023->80/tcp   contenedorN23
0509723d6f3f   httpd     "httpd-foreground"       56 minutes ago      Up 55 minutes      0.0.0.0:8022->80/tcp, :::8022->80/tcp   contenedorN22
84749e14fced   httpd     "httpd-foreground"       56 minutes ago      Up 55 minutes      0.0.0.0:8021->80/tcp, :::8021->80/tcp   contenedorN21
698c38d7fbe3   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8020->80/tcp, :::8020->80/tcp   contenedorN20
f6838a4e2cf3   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8019->80/tcp, :::8019->80/tcp   contenedorN19
eef8e72a0f92   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8018->80/tcp, :::8018->80/tcp   contenedorN18
13b270adc526   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8017->80/tcp, :::8017->80/tcp   contenedorN17
bb028eadf533   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8016->80/tcp, :::8016->80/tcp   contenedorN16
ef0e4d350815   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8015->80/tcp, :::8015->80/tcp   contenedorN15
d1a99e7d2ede   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8014->80/tcp, :::8014->80/tcp   contenedorN14
27c34cbb763d   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8013->80/tcp, :::8013->80/tcp   contenedorN13
81f53f4142af   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8012->80/tcp, :::8012->80/tcp   contenedorN12
7fe276f2271c   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8011->80/tcp, :::8011->80/tcp   contenedorN11
b6f2f86dab3c   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8006->80/tcp, :::8006->80/tcp   contenedorN6
1f19bdad9188   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8004->80/tcp, :::8004->80/tcp   contenedorN4
1871adb949bd   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8003->80/tcp, :::8003->80/tcp   contenedorN3
7fbf0876889e   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8002->80/tcp, :::8002->80/tcp   contenedorN2
1a2102c2fe38   httpd     "httpd-foreground"       56 minutes ago      Up 56 minutes      0.0.0.0:8001->80/tcp, :::8001->80/tcp   contenedorN1
4c4fd2dc6bcd   httpd     "httpd-foreground"       59 minutes ago      Up 59 minutes      0.0.0.0:8090->80/tcp, :::8090->80/tcp   contenedorN10
43ba6c239d02   httpd     "httpd-foreground"       59 minutes ago      Up 59 minutes      0.0.0.0:8089->80/tcp, :::8089->80/tcp   contenedorN9
4cbbfa8a5e55   httpd     "httpd-foreground"       About an hour ago   Up About an hour   0.0.0.0:8088->80/tcp, :::8088->80/tcp   contenedorN8
2261a43356e2   httpd     "httpd-foreground"       About an hour ago   Up About an hour   0.0.0.0:8087->80/tcp, :::8087->80/tcp   contenedorN7
9940cc055f4b   httpd     "httpd-foreground"       About an hour ago   Up About an hour   0.0.0.0:8086->80/tcp, :::8086->80/tcp   contenedorN5
711c87935496   httpd     "httpd-foreground"       6 hours ago         Up 6 hours         0.0.0.0:8085->80/tcp, :::8085->80/tcp   contenedorEntrega
fad1205d7d18   httpd     "httpd-foreground"       6 hours ago         Up 6 hours         0.0.0.0:8084->80/tcp, :::8084->80/tcp   contenedorTrabajo
24e23e193bc1   httpd     "httpd-foreground"       7 hours ago         Up 7 hours         0.0.0.0:8083->80/tcp, :::8083->80/tcp   contenedorEjemplo
1014b072bd34   httpd     "httpd-foreground"       7 hours ago         Up 7 hours         0.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
```

*Ilustración 46*

2) Para detenerlo ejecutamos "*docker stop*".

```
[root@ip-10-0-24-241 ec2-user]# docker stop 4e0a81e4ec1d
4e0a81e4ec1d
```

*Ilustración 47*

3) Para visualizar los contenedores que están corriendo y el detenido ejecutamos "*docker ps -a*".

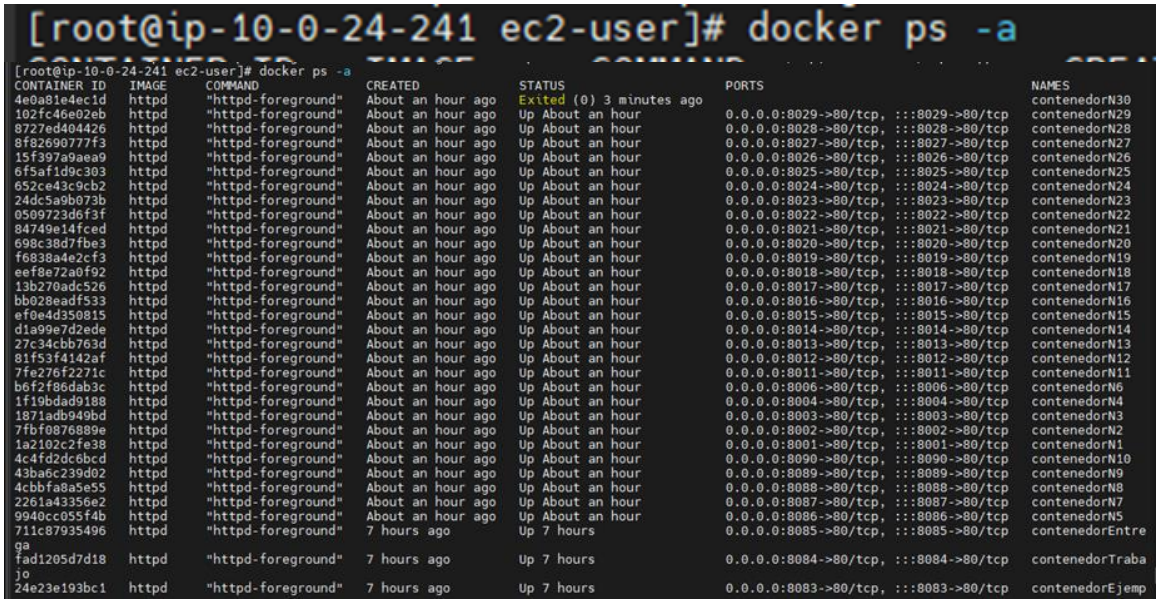



*Ilustración 48*

4) Si necesitamos iniciar el Docker pausado ejecutamos "*docker start*" y el id del Docker detenido.

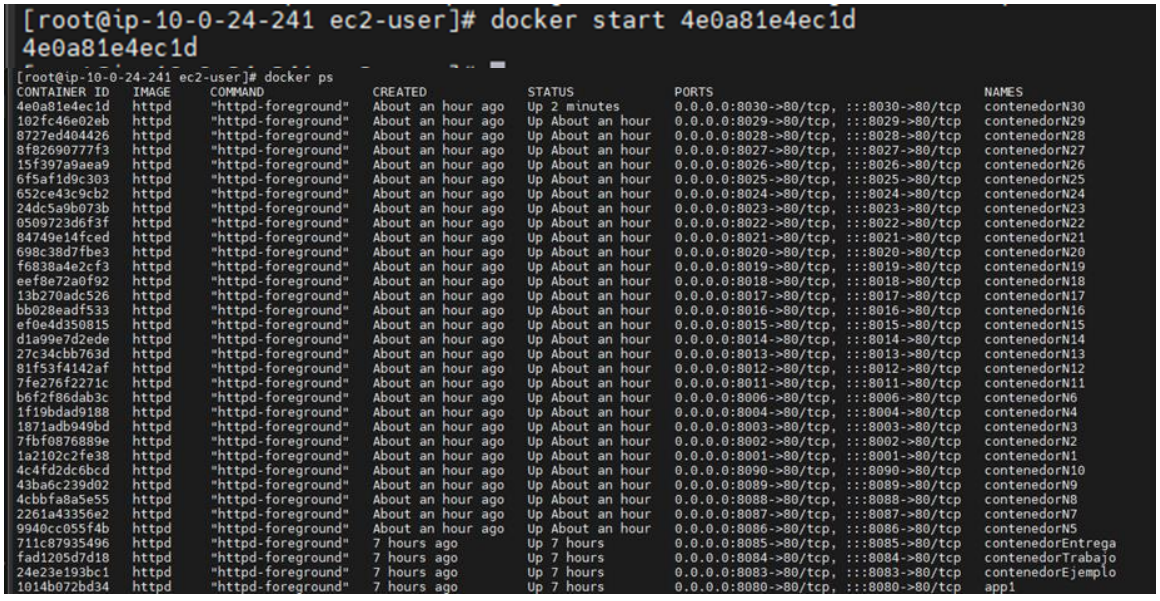



*Ilustración 49*

5) Para eliminarlo sería "*docker stop*" y el id del Docker a eliminar inicialmente para detenerlo y luego se ejecuta "*docker rm*" y el id del Docker a eliminar.

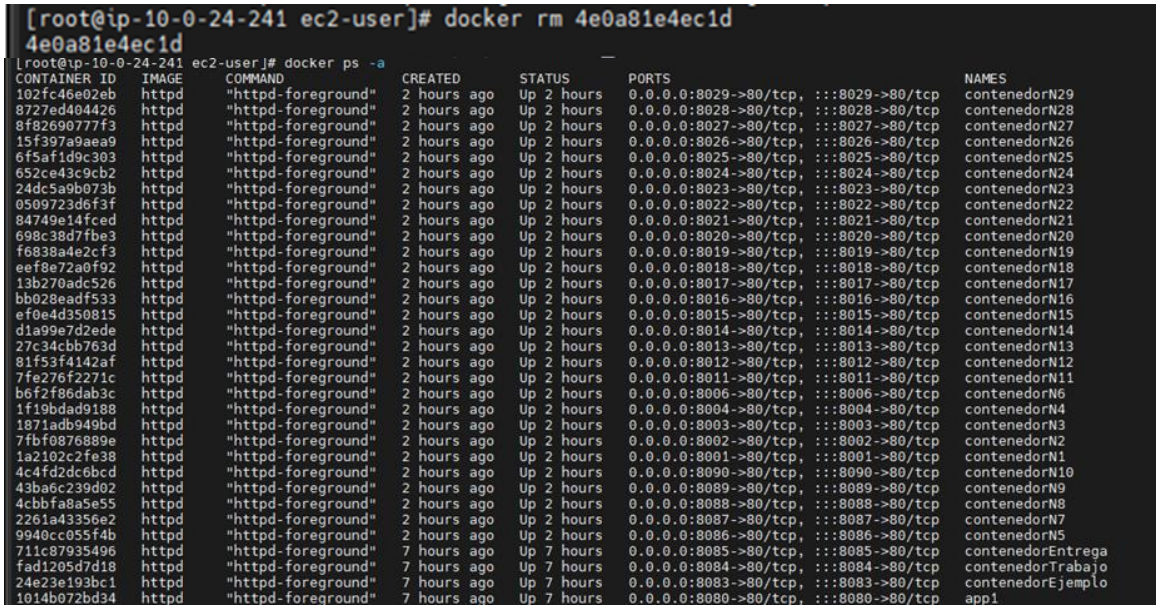



*Ilustración 50*

De ser el caso de que deseemos hacer una eliminación forzada ejecutamos "**docker rm -f**" y el id del Docker a eliminar sin importar si está corriendo o no.

## Conclusiones

1) **Practica de Conceptos:** Este seminario nos permitió aplicar de múltiples maneras varias practicas del conocimiento adquirido durante cada clase; conocimientos como la creación de redes virtuales (VPC), configuraciones de estancias (EC2), gestiones de grupos de seguridad y publicación de servicios en la nube de AWS.
2) **Ambientes mixtos:** Se logró diseñar varias arquitecturas de red con diferentes instancias (Amazon Linux y Windows Server), ejemplificando la viabilidad de los entornos mixtos en la nube y su conexión, accesibilidad y estabilidad.
3) **Optimización de recursos con Docker (Contenedores):** La implementación de tecnologías de contenedor nos permitió evidenciar las ventajas del manejo de virtualizaciones ligeras contra las tradicionales, demostrando su eficiencia de consumo de recursos, velocidad y portabilidad dándonos a entender la obligatoria necesidad de implementar tecnologías modernas a los proyectos.
4) **Fortalecimiento de habilidades:** Mejoramos conocimientos técnicos de habilidades asociadas a la administración de servidores, configuraciones de servicios web (ISS y Apache), aperturas de puertos, conexiones remotas a través de RDP y SSH, de igual forma gestionamos recursos en nube de manera segura y eficiente.
5) **Manejo de la nube en entornos modernos:** La experiencia de este seminario logro confirmarnos que el servicio en la nube es esencial en el desarrollo de proyectos y/u soluciones modernas que sean escalables al futuro, destacando esencialmente AWS como una plataforma robusta, pero a la misma vez versátil para el aprendizaje de proyectos reales de grandes infraestructuras.

## Palabras clave

2. AWS – Plataforma de servicios en la nube
3. Servicios en la nube – Recursos tecnológicos vía Internet
4. Docker – Contenedores ligeros y portables
5. Instancia EC2 – Servidor virtual en AWS
6. Máquina virtual – Sistema operativo emulado
7. Tecnología – Herramientas para resolver problemas
8. Soluciones Modernas – Alternativas eficientes y actuales
9. Arquitectura – Estructura de sistemas tecnológicos
10. Entornos – Espacios controlados de ejecución
11. Optimización – Mejora del rendimiento tecnológico
12. Linux / Windows – Sistemas operativos
13. Comandos – Instrucciones para ejecutar acciones
14. Apache – Servidor web de código abierto
15. IP – Dirección de red única
16. Puertos – Canales de comunicación digital
17. Redes – Conexión entre dispositivos o sistemas
18. Configuraciones – Ajustes de parámetros del sistema
19. SSH – Acceso remoto seguro
20. VPC – Red virtual privada (AWS)
21. HTML – Lenguaje para páginas web
22. Sistema Operativo – Software base del computador
23. http – Protocolo de transferencia web

## Referencias


1. Docker Inc. (2023). *What is a Container?*. Docker Documentation. https://docs.docker.com/get-started/overview/

2. Pontificia Universidad Católica de Chile. (2024). *Cómo elaborar un marco conceptual*. Recuperado de https://comunicacionacademica.uc.cl/images/recursos/espanol/escritura/recurso_en_pdf_extenso/15_Como_elaborar_un_marco_conceptual.pdf

3. Amazon Web Services. (2023). Amazon EC2 documentation. AWS Documentation. https://docs.aws.amazon.com/ec2/

4. IBM. (2022). What is cloud computing? IBM Cloud Learn Hub. https://www.ibm.com/cloud/learn/cloud-computing

5. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001

6. Pons, N. (2016). *Linux: principios básicos de uso del sistema*. Ediciones ENI.

7. Ortiz, C. E. E., Zavala, R. F., & Sosa, C. D. L. C. (2022). Uso de AWS Educate para un Laboratorio Virtual de Seguridad en Redes. *Revista Electrónica sobre Tecnología, Educación y Sociedad*, *9*(18).

8. Patibandla, K. R. (2024). Design and Create VPC in AWS. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, *1*(1), 273-282.