TRABAJO DE GRADO
Seminario-Diplomado.

Implementación de red en AWS con Docker y balanceo de carga

Corporación Universitaria Remington.
Facultad de ingeniería
Ingeniería de Sistemas

Luddy Mileny Puerta Ruiz
Juan Sebastian Forero Salamanaca
Docente Juan Pablo Berrio Lopez
Seminario de AWS
2025

Tabla de Contenidos

Resumen

Dentro de este proyecto vamos a ver las configuraciones para poder realizar la creación de un VPC, ¨ ya que es el primer elemento que necesitamos crear, que es VPC en terminología de AWS, que es nuestro centro de datos virtual. (Guijarro Olivares, 2019), la Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet con 1 subred pública para Linux EC2 y 1 Subred pública para Windows EC2 con sus grupos de seguridad separados y la tabla de rutas con acceso a Internet Gateway (IGW) asociado a la VPC; además también veremos la implementación de los servicios de Docker con instancias S2, esto es necesario, ya que vamos a comprender el servicio de Docker y la utilización de los servicios autoadministrados de AWS esto con el fin de que nos permita administrar contenedores; con el proceso de dockerizacion allí vamos a encontrar Docker corriendo dentro de una instancia de Linux.

Durante el proceso de dockerización, veremos cómo Docker opera dentro de una instancia de Linux. Allí, instalaremos y ejecutaremos Docker en una instancia EC2, lo cual nos brindará la oportunidad de experimentar con la dockerización de servicios. Esto implica empaquetar
aplicaciones dentro de contenedores, facilitando así su despliegue, movilidad y gestión

Además, con la implementación de pruebas de estrés sobre la infraestructura, vamos a poner a funcionar varios contenedores para ver el comportamiento en rendimiento tanto de la instancia como de los mismos contenedores, con las pruebas de estrés vamos a poder ver cuanto se está usando en CPU y RAM y poder ver el aumento del uso de los recursos y como saturar el servicio.

En resumen, este proyecto nos capacitará para construir una red en AWS desde cero, utilizar instancias EC2 con diversos sistemas operativos, trabajar con Docker en la nube y evaluar si nuestra infraestructura está preparada para funcionar bajo las diferentes pruebas de estrés, todo ello con un enfoque práctico y aplicable a situaciones reales.

Palabras clave

Instancias AWS (EC2), creación de VPC, servicios de Docker, pruebas de estrés, Implementación de red AWS, Nginx, Apache,

Marco conceptual y contextual

Este informe se realizó durante un seminario de AWS sobre computación en la nube, La computación en la nube se refiere a la entrega de servicios de computación a través de la Internet, o “la nube”. En lugar de utilizar servidores locales o computadoras personales para almacenar, gestionar y procesar datos, la computación en la nube permite acceder a estos recursos desde centros de datos remotos gestionados por proveedores de servicios en la nube (IMECAF, 2024)

Además de acuerdo al objetivo fue aprender a crear y configurar una red completa en AWS utilizando servicios disponibles en la plataforma para esto se trabajó con la creación de una VPC que incluye dos subredes públicas, una para una instancia con sistema operativo Linux y otra para una con Windows ambas fueron configuradas para ser accesibles desde Internet

Se exploró el uso de Docker y Apache como herramientas para ejecutar aplicaciones en servidores de forma óptima, se instaló Docker en una instancia EC2 con Linux y se usaron contenedores para alojar servicios web.

También se realizaron pruebas de estrés que por lo cual se define como una sencilla utilidad de línea de comandos diseñada para sobrecargar los recursos del sistema, como la CPU, la memoria RAM y las operaciones de E/S. Simula condiciones extremas y permite ver cómo responde la aplicación, el contenedor o incluso el sistema completo a la escasez de recursos, (Serverspace, n.d.) esto sirvió para evaluar como funciona la instancia bajo la prueba de estrés.

Este trabajo fue útil para aplicar todos los conceptos vistos en el seminario y para conocer de forma práctica cómo se puede montar una infraestructura en la nube con servicios

Desarrollo e implementación del aprendizaje

Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet

Objetivo General

Diseñar, desplegar y documentar una red en AWS que incluya dos instancias EC2 (una Windows y una Linux), asegurando su accesibilidad pública, conectividad entre ellas y la instancia de un servidor web funcional en cada instancia.

Parte 1: Documentación Técnica

1. Diagrama de Arquitectura

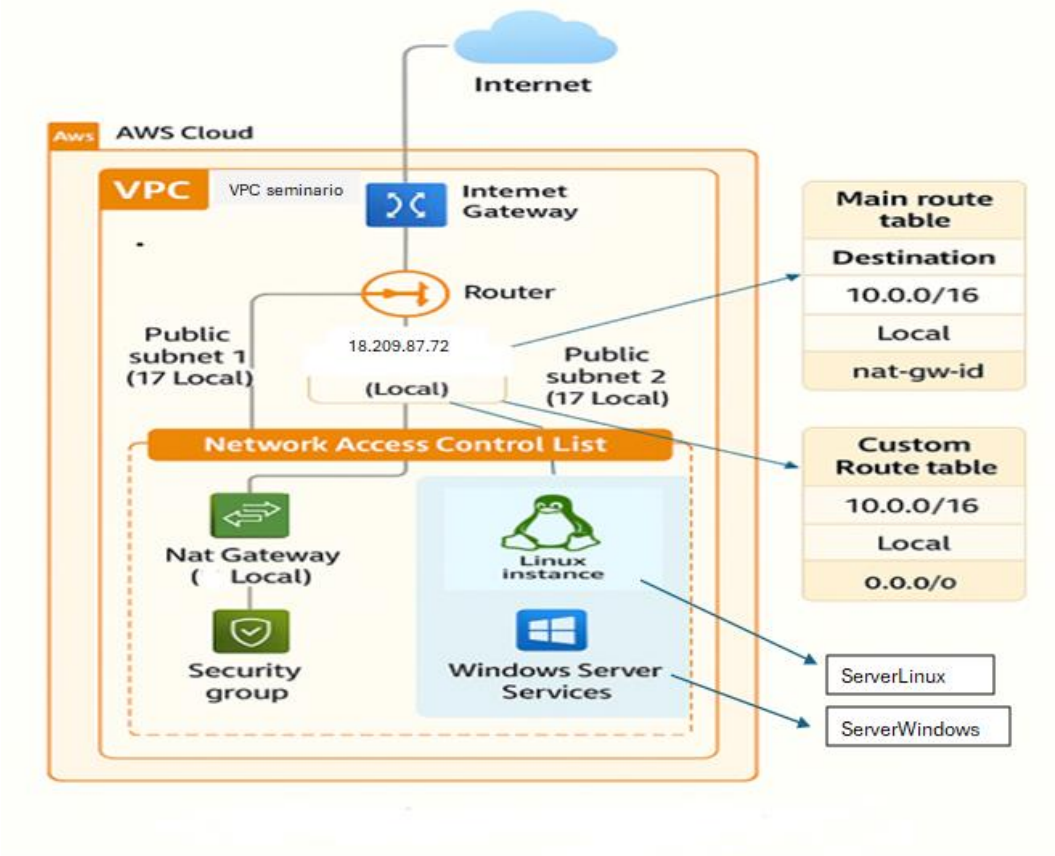


Ilustración 1- Esquema de la Arquitectura

Componentes del Diagrama:

- VPC personalizada
- 1 Subred pública para linux EC2
- 1 Subred pública para Windows EC2
- Internet Gateway (IGW) asociado a la VPC
- Tabla de rutas con acceso a IGW
- Instancia EC2 (Amazon Linux 2023)
- Instancia EC2 (Windows Server 2016 Base)
- Grupo de seguridad separados

2. Descripción de la Arquitectura

Se creó una VPC personalizada con dos subredes públicas: una para la instancia Linux nombrada ServerLinux y otra para la instancia Windows nombrada ServerWindows. Ambas subredes están asociadas a una tabla de rutas que permite acceso a Internet mediante un Internet Gateway (IGW). Las instancias EC2 están protegidas por grupos de seguridad que permiten acceso RDP (puerto 3389) para Windows, SSH (puerto 22) para Linux y HTTP (puerto 80) para ambos servidores web. La com(IMECAF, 2024)

3. Tipo de Instancias Usadas

- Linux: AMI de Amazon Linux 2023 (Free Tier elegible)
- Windows: Windows Server 2016 Base (Free Tier elegible)

Justificación: Se utilizaron versiones compatibles con Free Tier para reducir costos. Ambas versiones ofrecen facilidad de configuración y amplia documentación en AWS.

4. Justificación de las Configuraciones de Red

- **VPC Personalizada:** Permite control total sobre la segmentación y seguridad.
- **Subredes Públicas**: Requeridas para que las instancias tengan acceso directo desde Internet.
- I**nternet Gateway (IGW)**: Necesario para que las instancias EC2 tengan acceso a Internet.
- **Grupos de Seguridad**: Configurados para permitir únicamente el tráfico necesario (RDP, SSH, HTTP), minimizando riesgos de seguridad.

5. Configuraciones Realizadas

Pasos para Crear Instancias EC2

1. Crear una nueva VPC con subredes públicas.
2. Asociar una tabla de rutas con acceso a IGW.
3. Crear y asociar un Internet Gateway a la VPC.
4. Lanzar una instancia EC2 con Amazon Linux 2023 y otra con Windows Server 2016.
5. Asignar IPs públicas durante el lanzamiento.
6. Crear pares de claves (.pem) para el acceso SSH y RDP.

EVIDENCIA VPC (Creación):

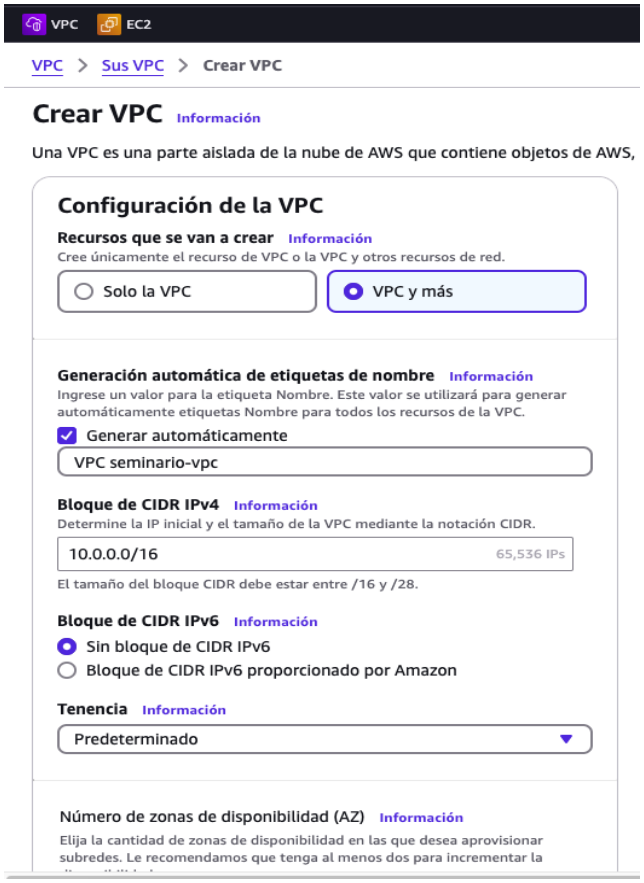


Ilustración 2 – Creación VPC

Ilustración 3 –Detalle de Vpc

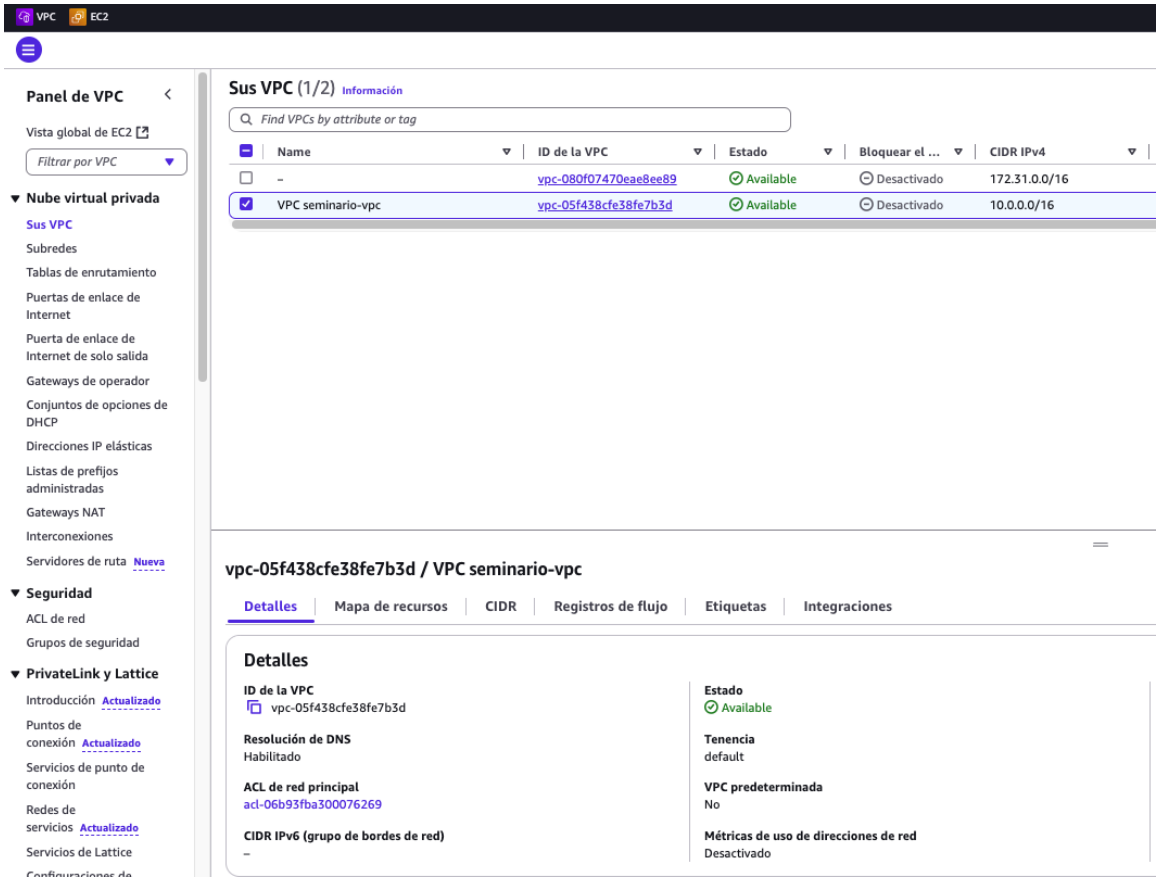


Ilustración 4 – Comprobación de VPC creado y disponible

TABLA ENRUTAMIENTO:

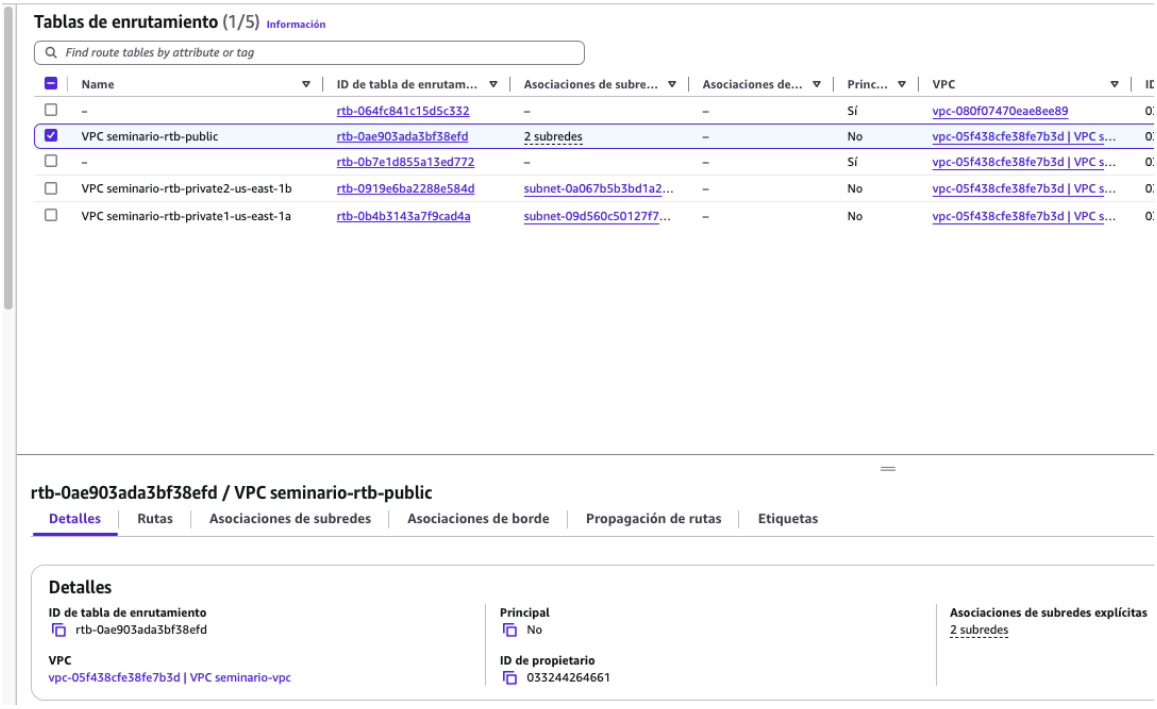


Ilustración 5 – Tabla Enrutamiento

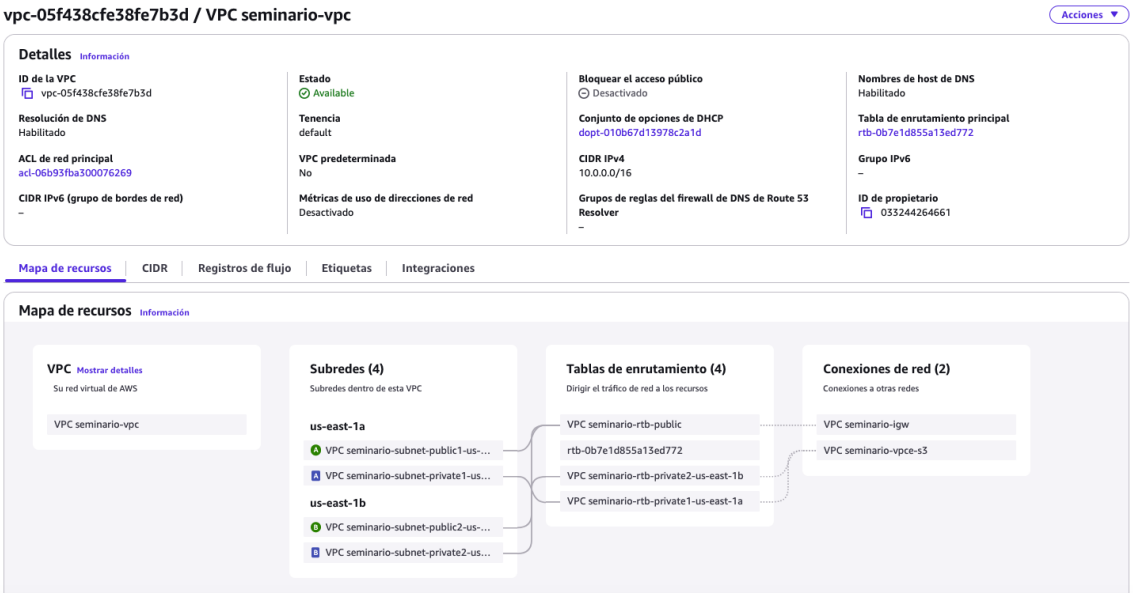


Ilustración 6 – Detalle del enrutamiento

Grupos de Seguridad:

- **Windows EC2:** Puertos abiertos: 3389 (RDP), 80 (HTTP): se creo un grupo de seguridad nombrado SG-ServerWindows.
- **Linux EC2:** Puertos abiertos: 22 (SSH), 80 (HTTP): se creo un grupo de seguridad nombrado SG-ServerLinux2.

Asignación de IPs:

- Cada instancia tiene asignada una IP pública en la cual en la sección de asignación automática se activó la IP pública (Habilitar).
- IPs privadas asignadas automáticamente por la subred.

EVIDENCIAS INSTANCIAS:

Servidor Windows:

1. Crear instancia

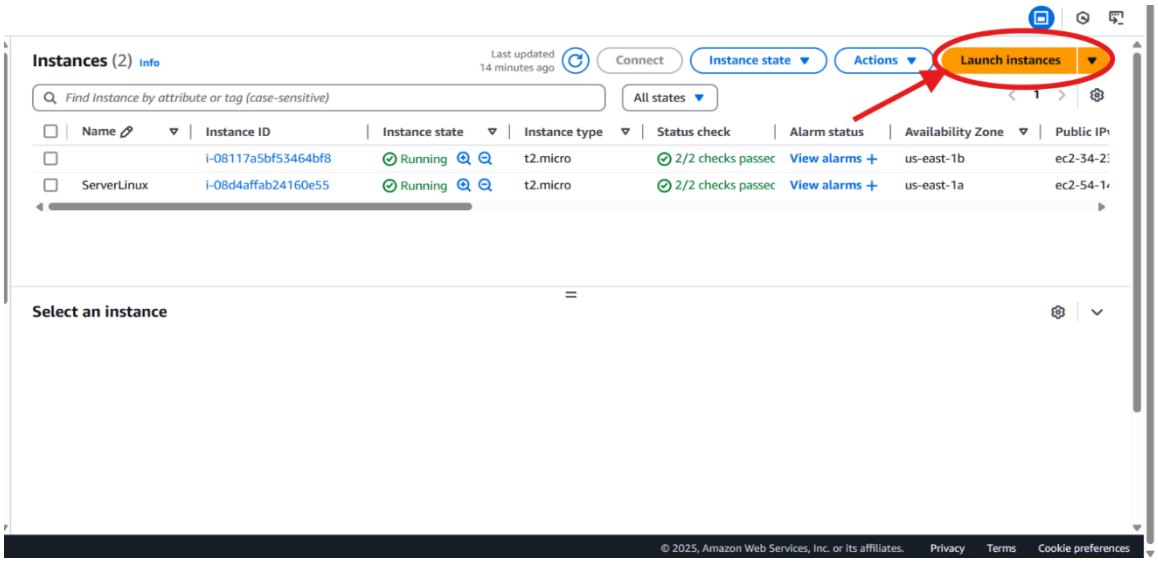


Ilustración 7 – Creación de instancia Windows

2. Se asigna un nombre y se elige el sistema Windows

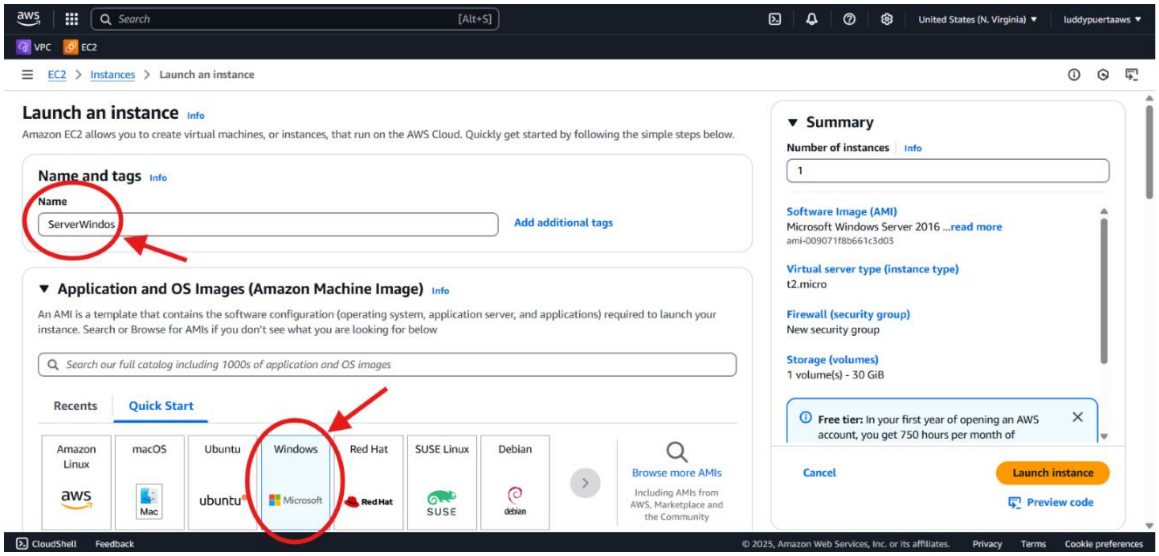


Ilustración 8 – Selección de la imagen Windows

3. Versión del sistema de la máquina y la capacidad que va a tener

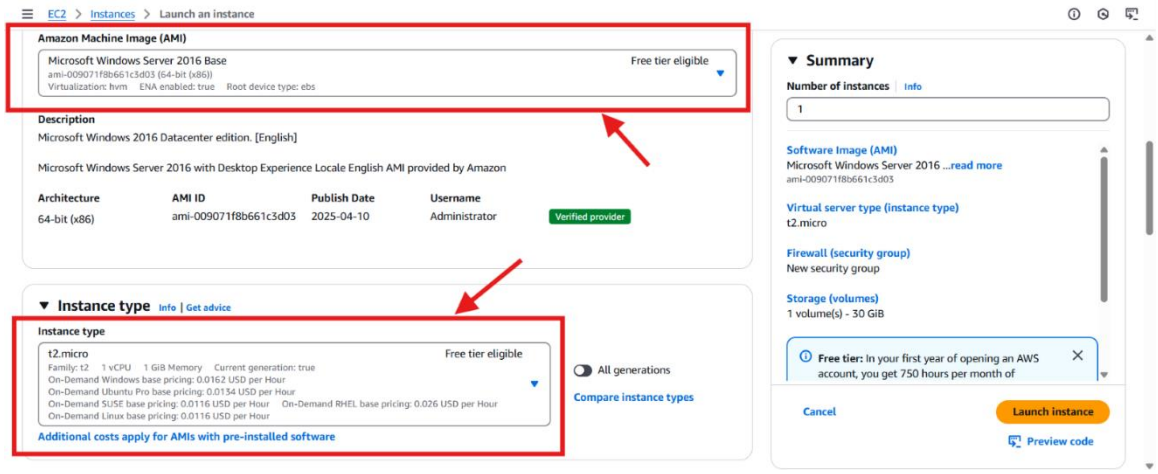


Ilustración 9 – Tipo de instancias

4. Se crea un par de claves

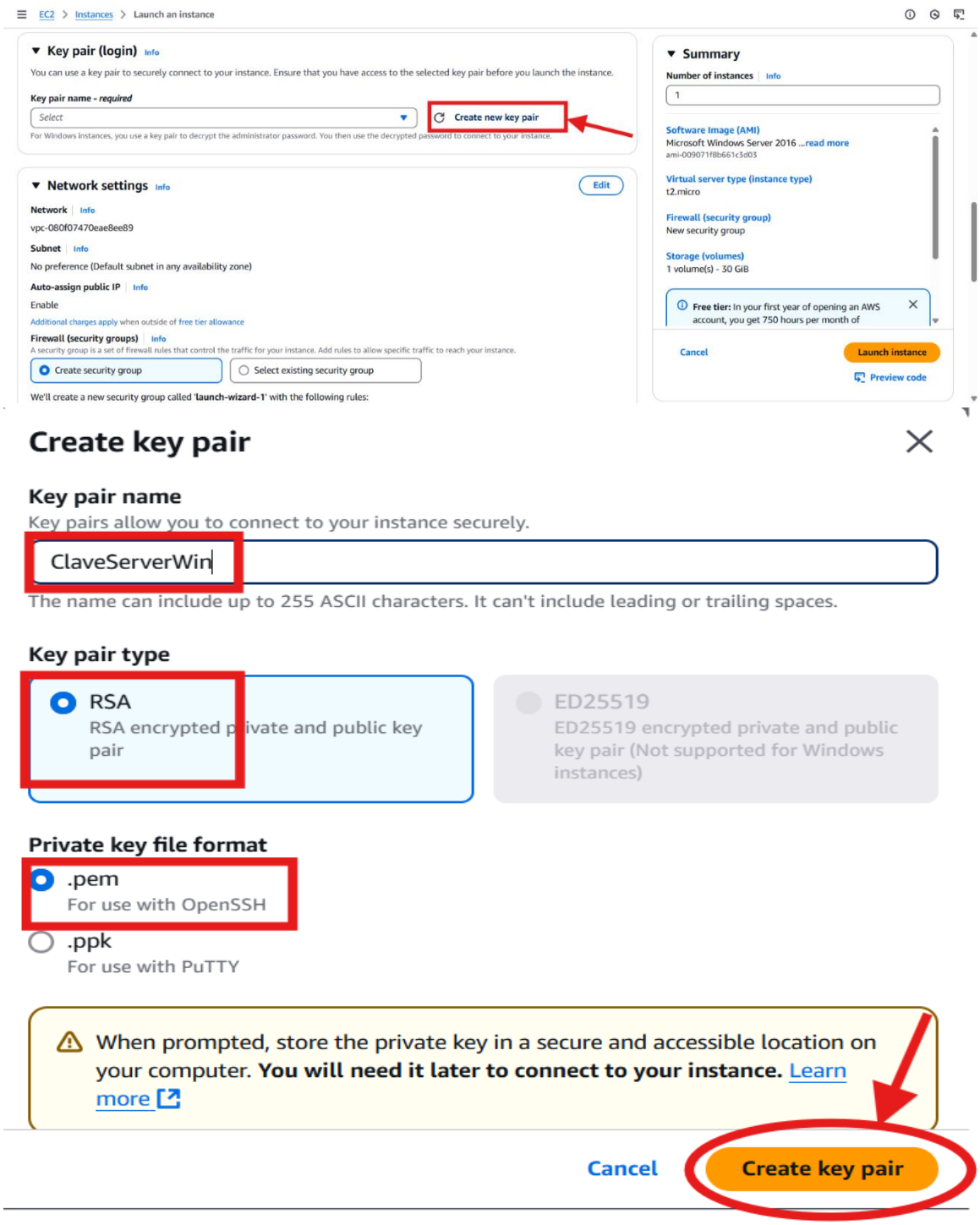


Ilustración 10 – Creación de par de llaves

5. Descarga Archivo (.pem)

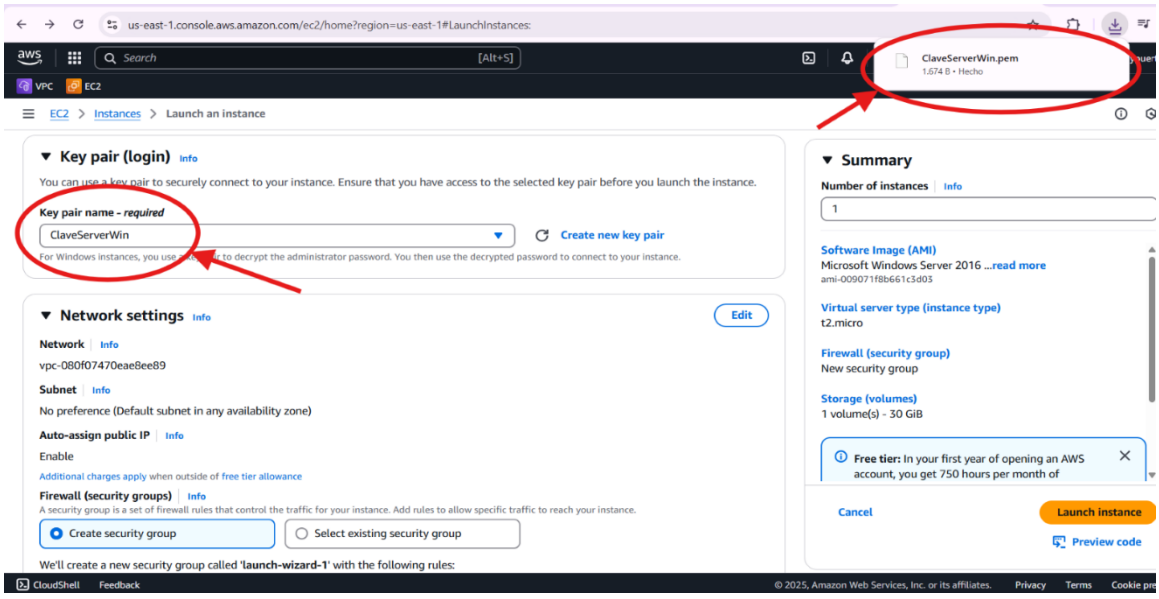


Ilustración 11 – Descarga de llaves

6. Editan Network settings y se lanza la instancia

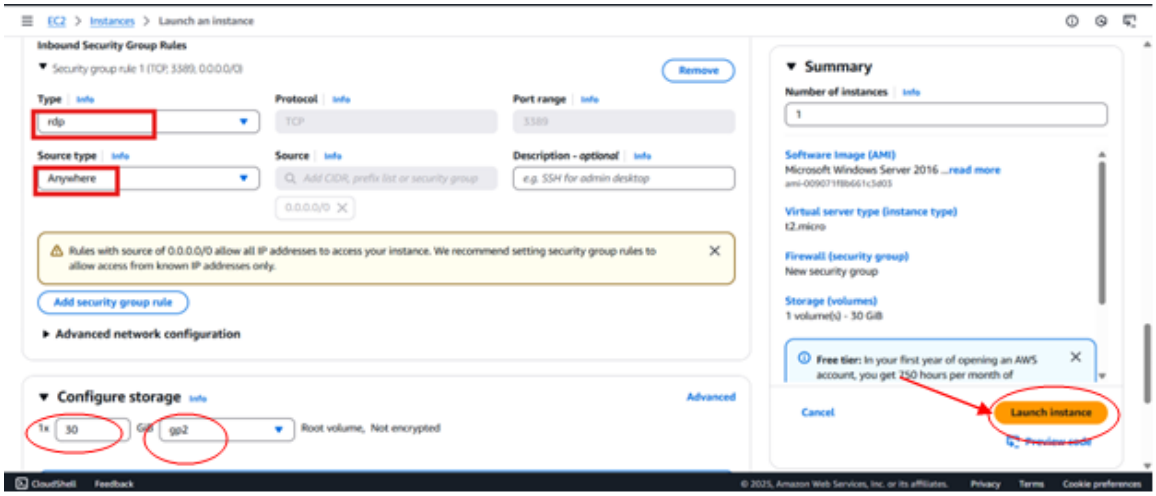


Ilustración 12 – Lanzamiento de instancias

Ilustración 13 – Creación de instancia exitosa

7. Se valida la Ip pública

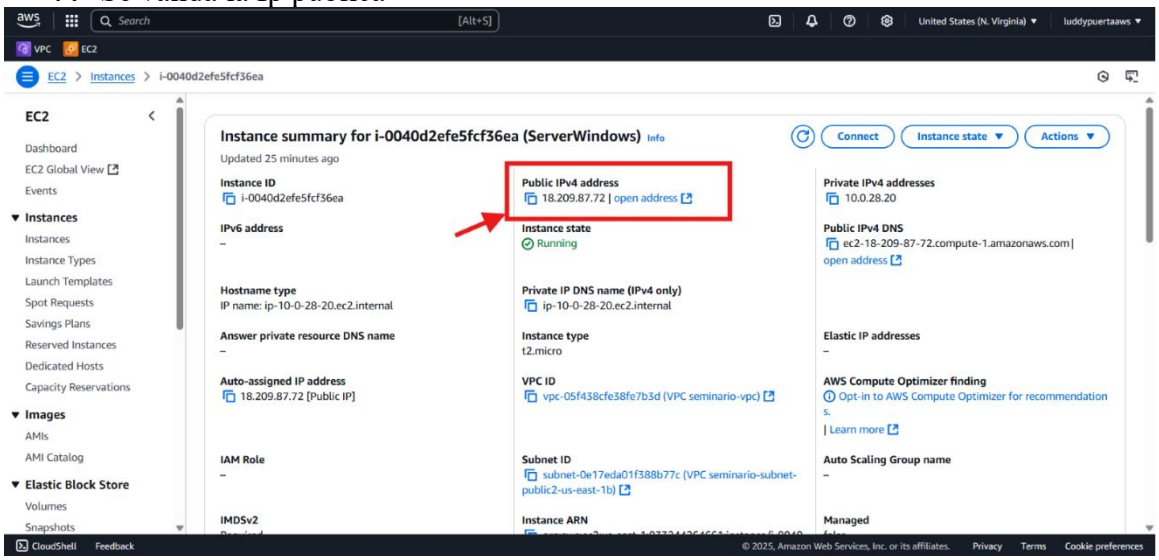


Ilustración 14 – Validación de IP pública

8. Obtenemos el password subiendo el archivo .pem

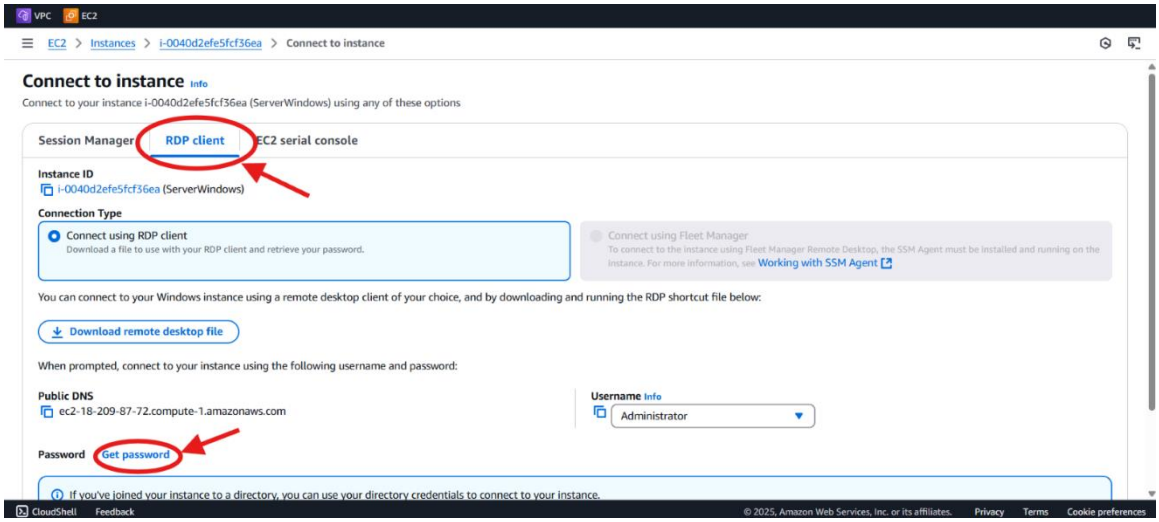


Ilustración 15 – Obtención de password

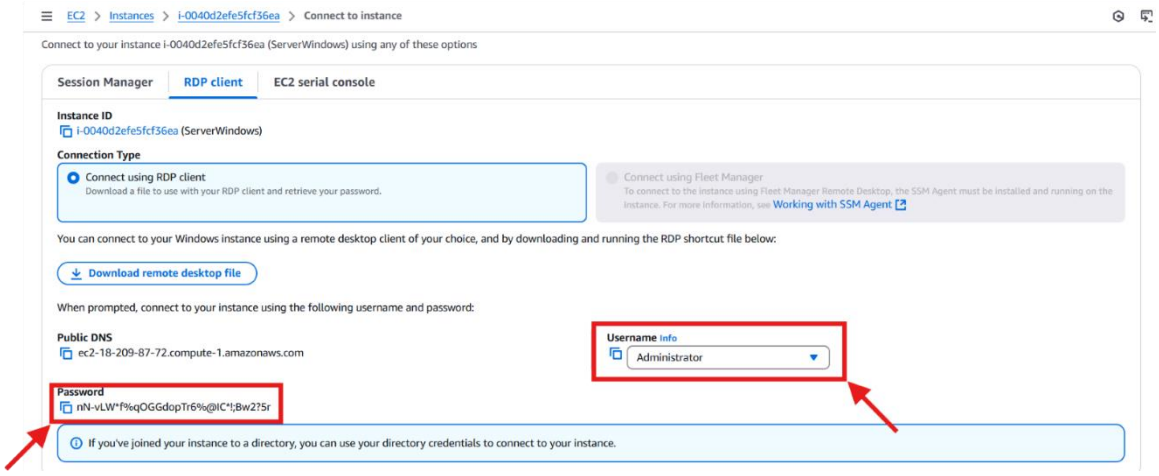


Ilustración 16 – Generador de password

9. Nos conectamos a la máquina

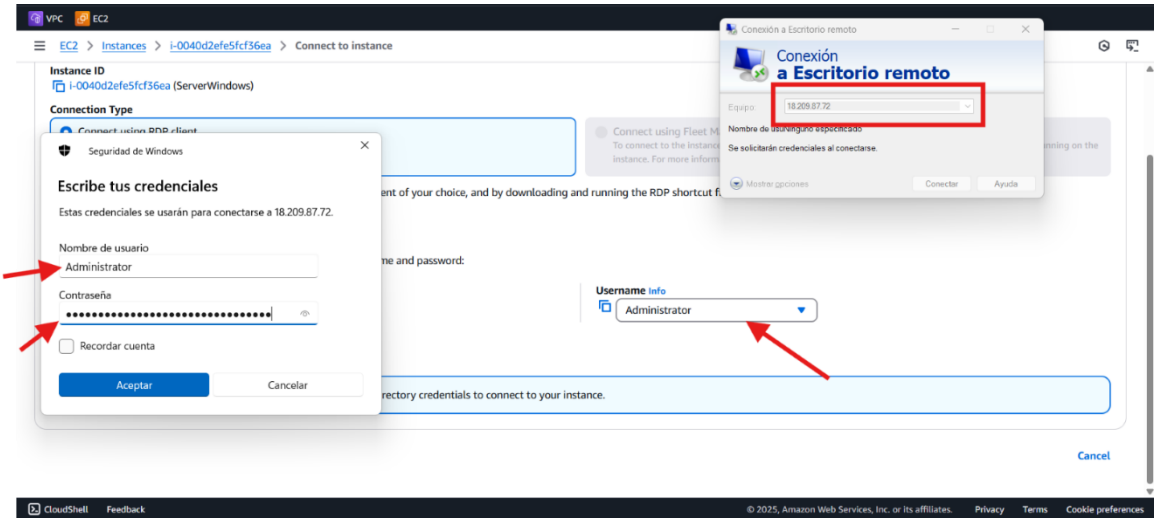


Ilustración 17 – Conexión remota

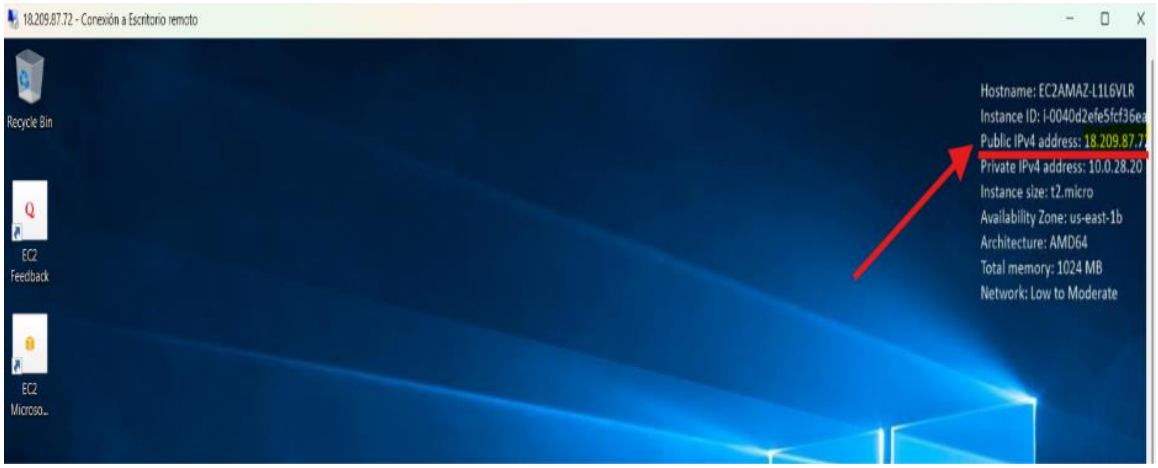


Ilustración 18 – Validación de conexión remota

10. Editamos roles de grupos de seguridad.

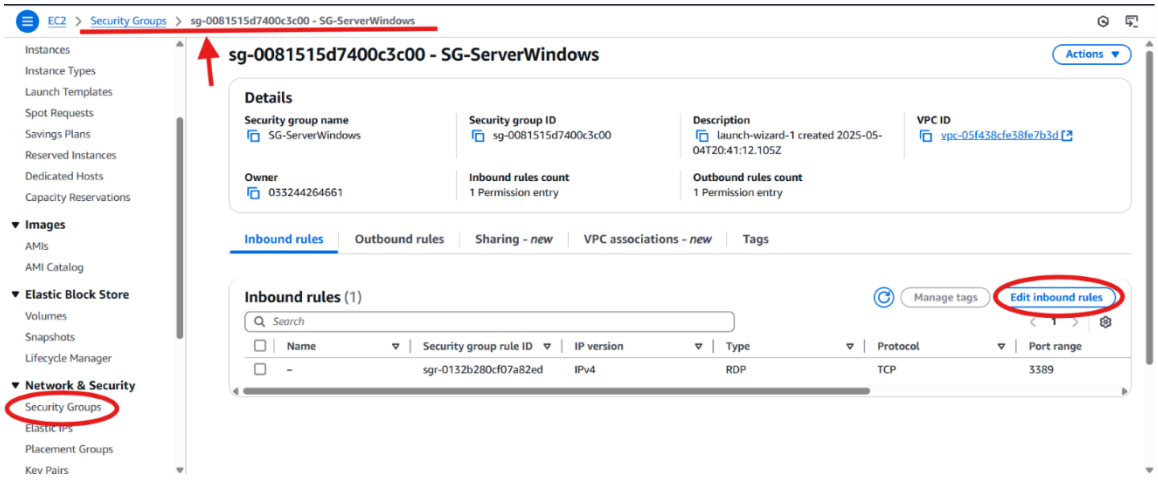


Ilustración 19 – Configuración de roles

11. Creamos nuevo rol

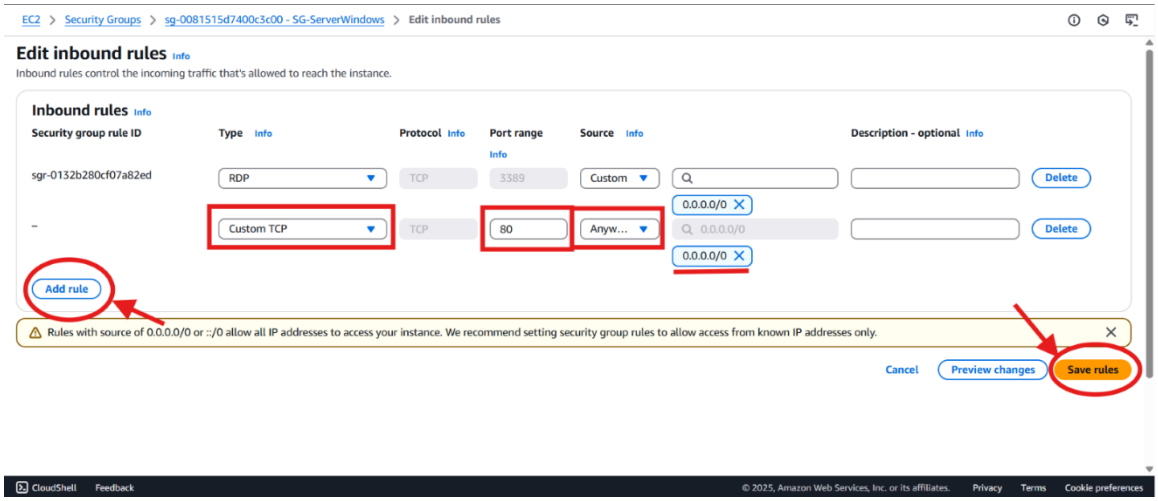


Ilustración 20 – Edición de reglas

12. Creamos nuevos roles

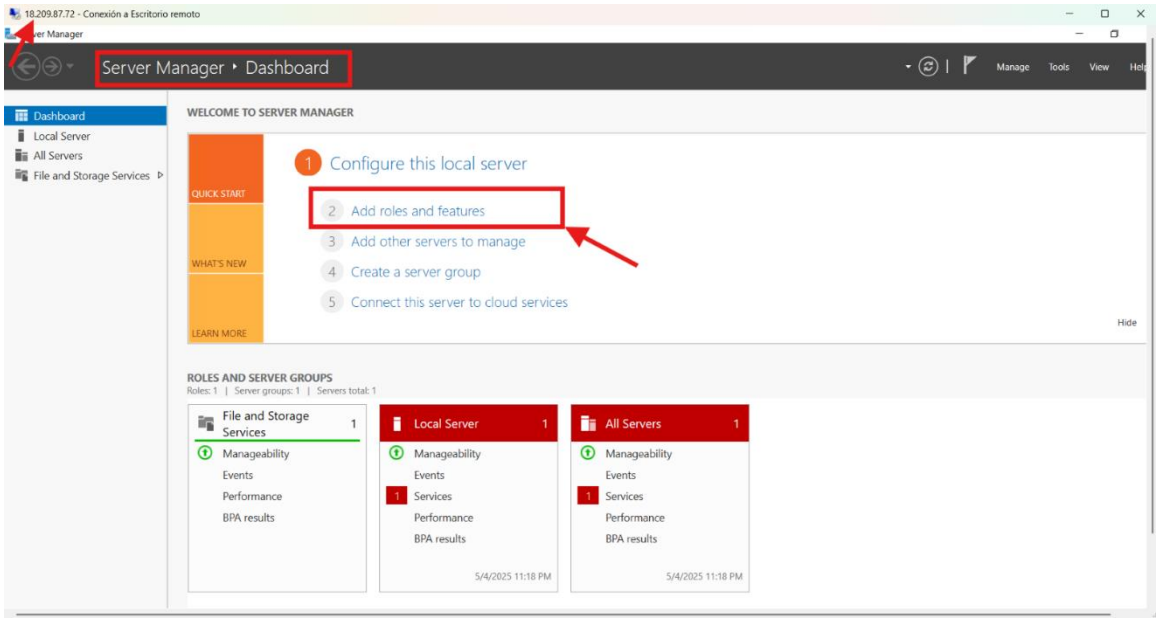


Ilustración 21 – Creación de nuevos roles

13. Se crea rol (IIS)

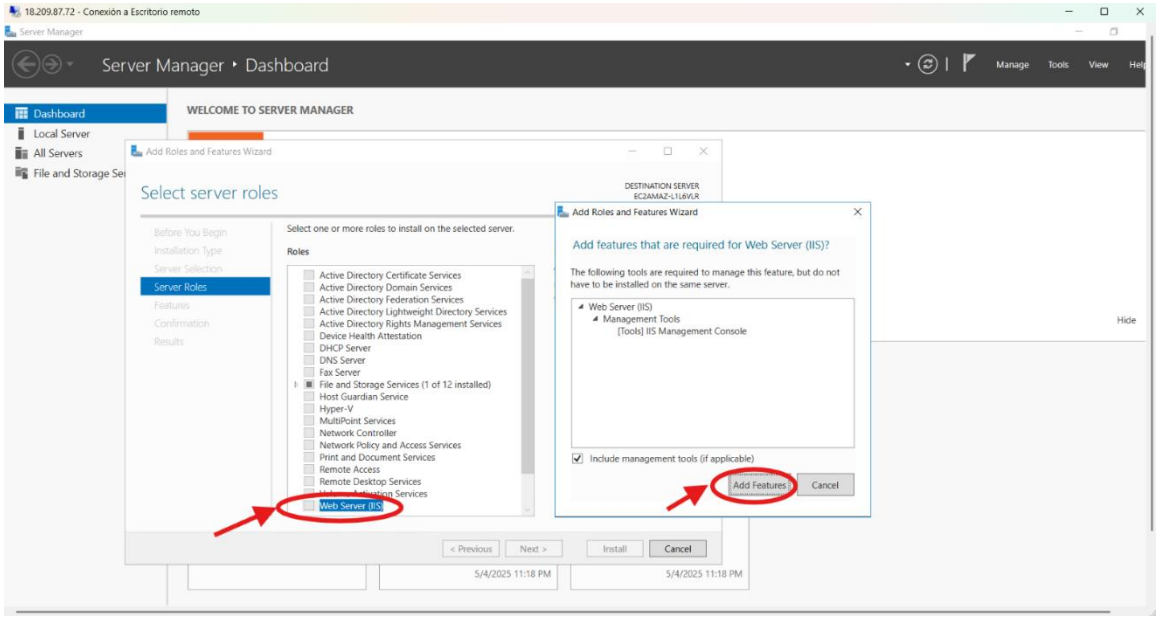


Ilustración 22 – Seleccion de roles

14. Se comprueba funcionamiento del servicio localhost

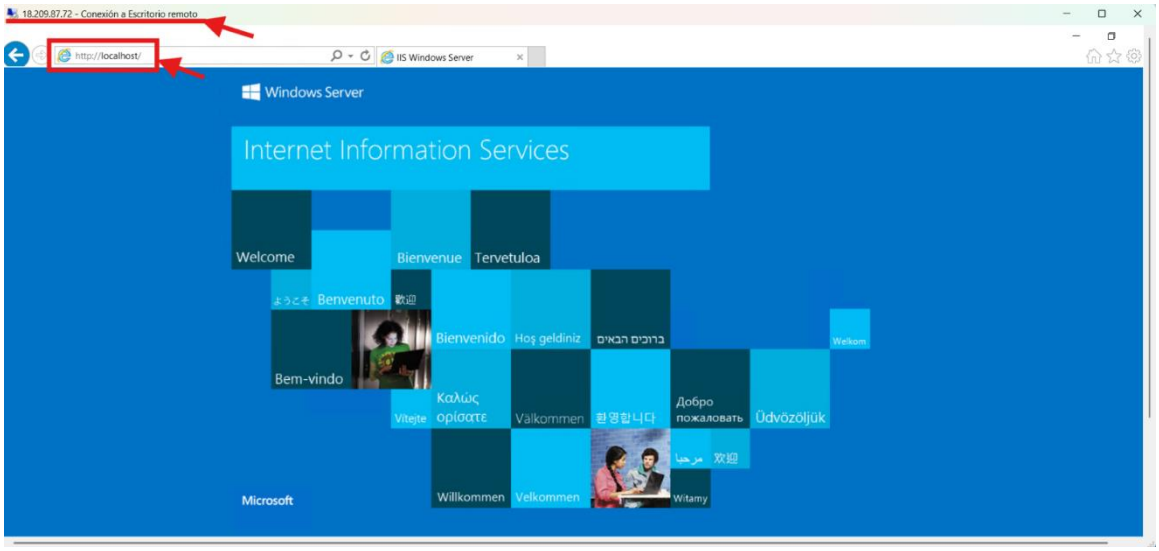


Ilustración 23 – Comprobación de funcionamiento del sistema Windows

15. Se comprueba funcionamiento del servicio con la direccion IP http://18.209.87.72/

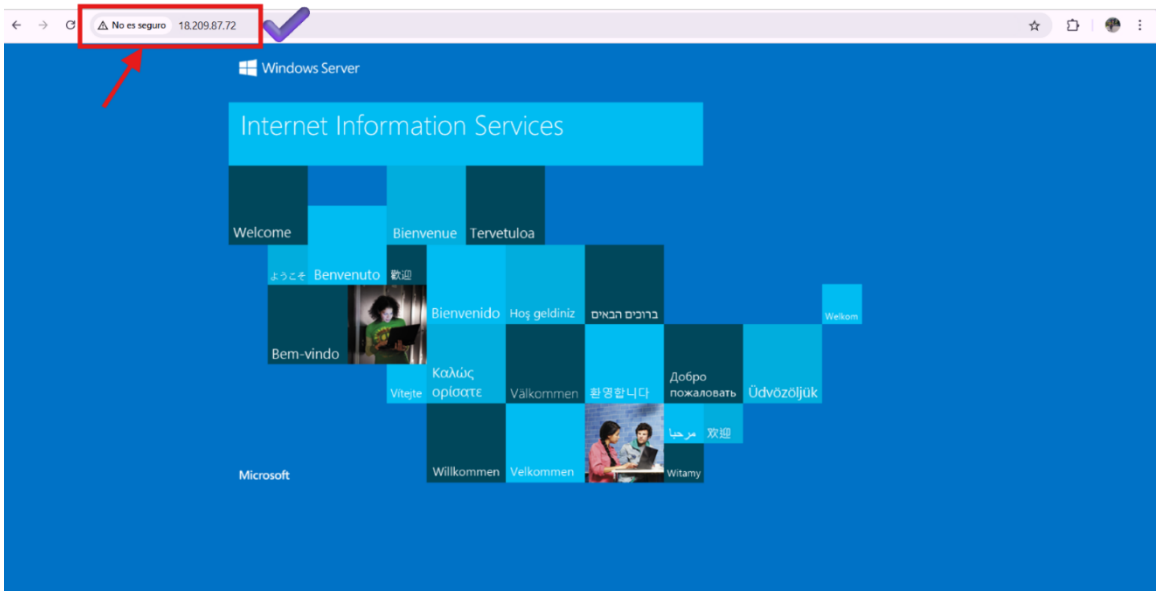


Ilustración 24 – Pruebas de dirección IP pública en Windows

Servidor Linux:

1. Se crea la instancia de Linux.

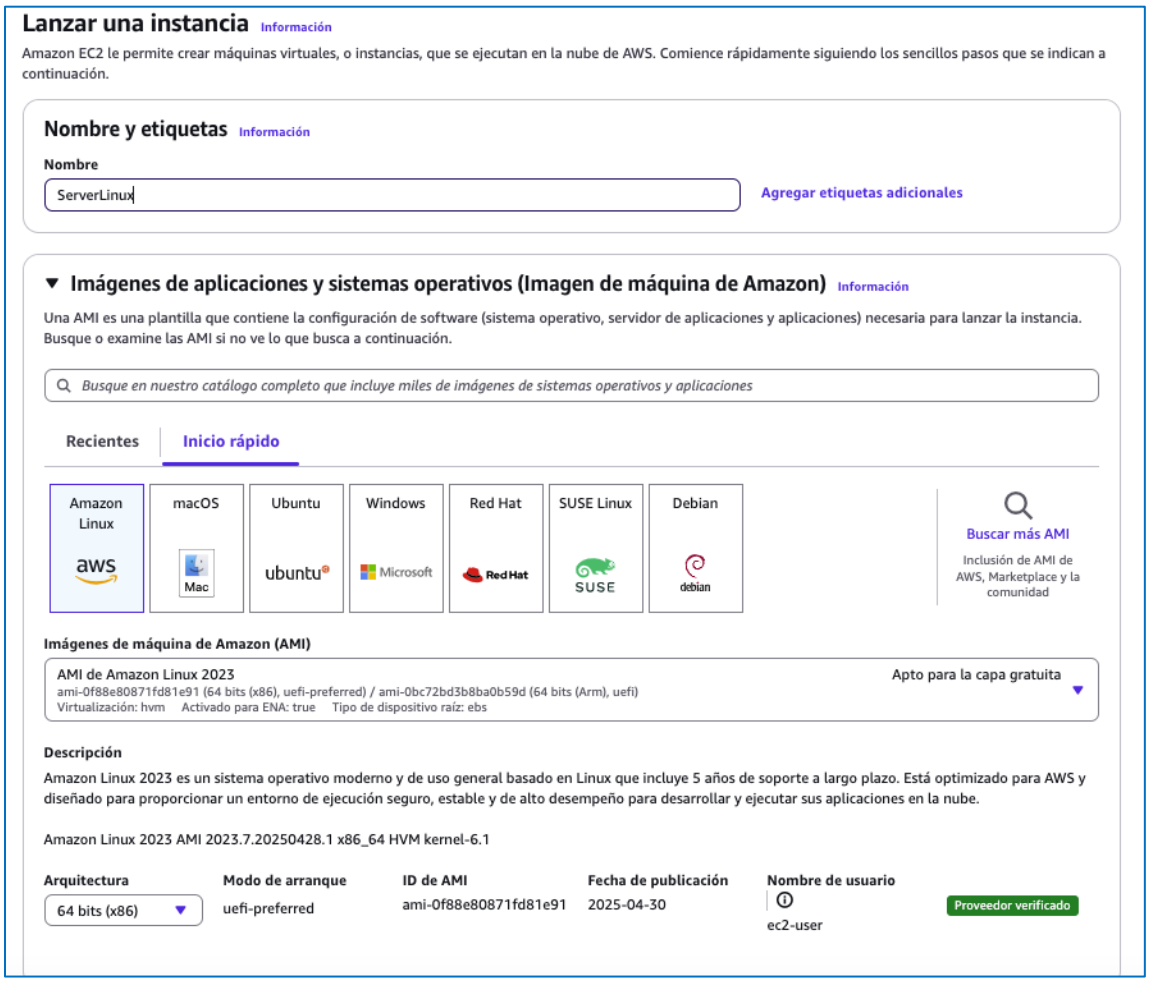


Ilustración 25 – Creación de instancia Linux

2. Se crea un par de claves

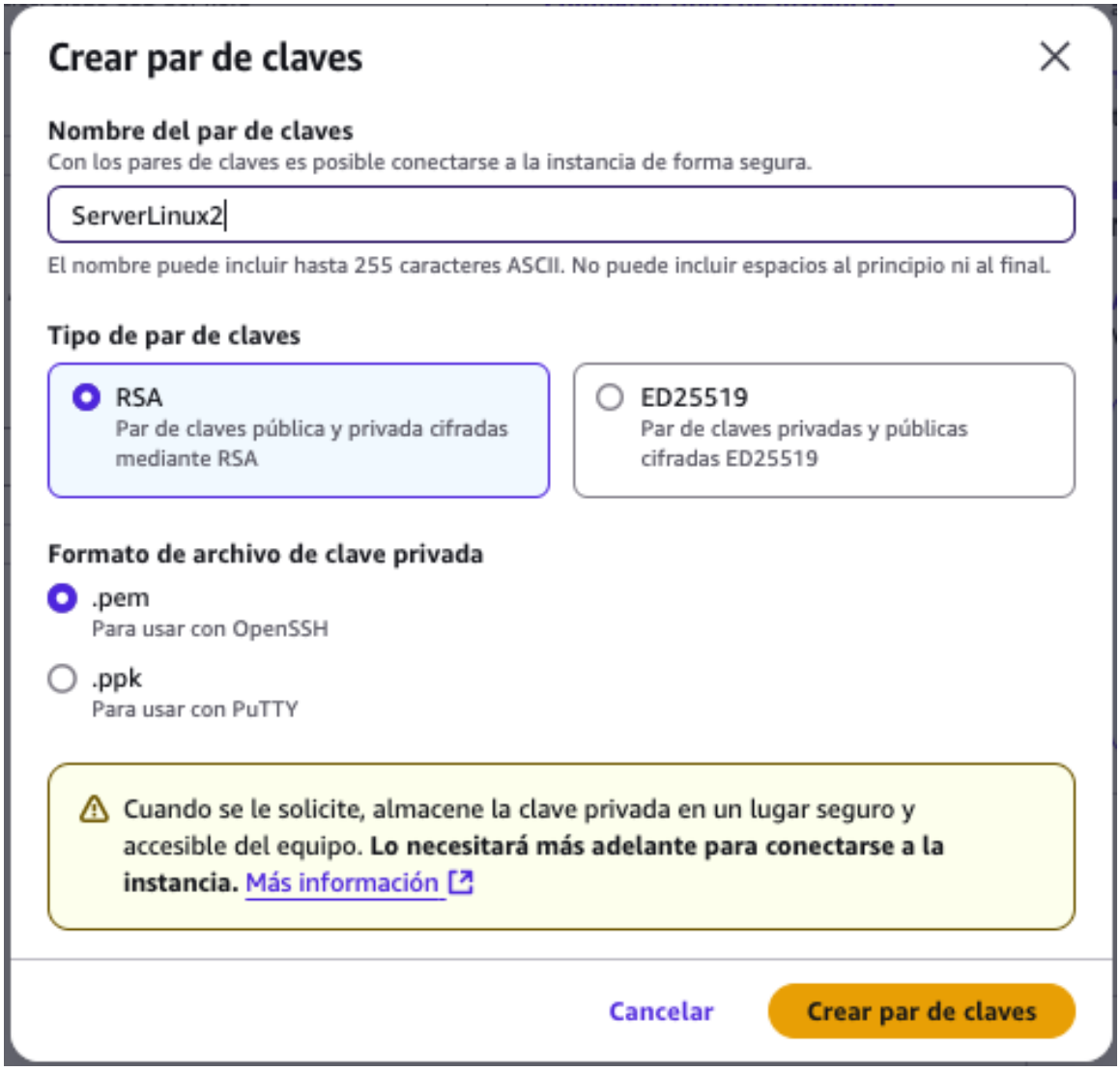


Ilustración 26 – Creación de par de llaves Linux

3. Se realiza la configuración de la red con el nombre del grupo de seguridad y se habilita la asignación automática de la ip pública.

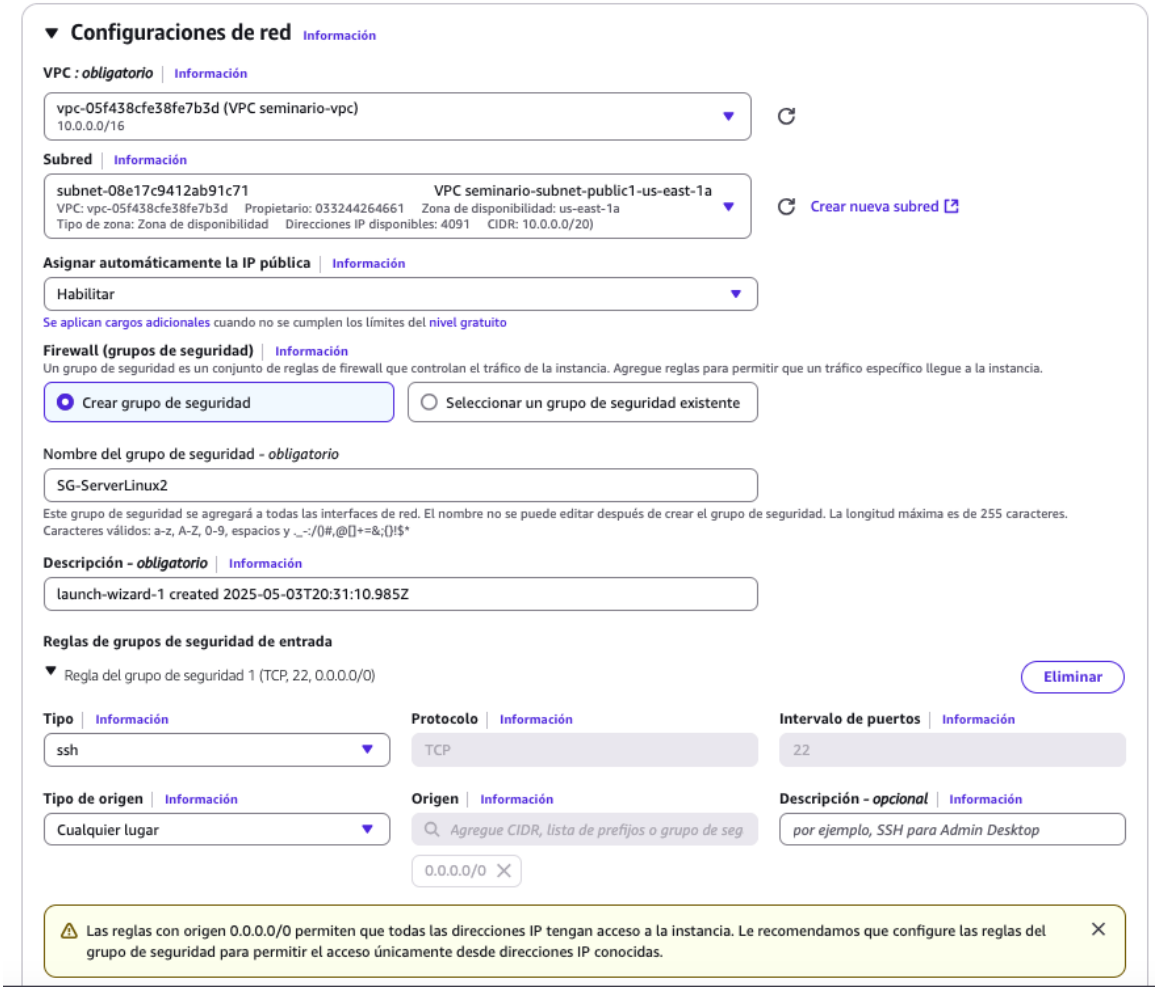


Ilustración 27 – Configuracion de red Linux

Conexión a la instancia Linux:

- Se realiza la conexión a través de SSH

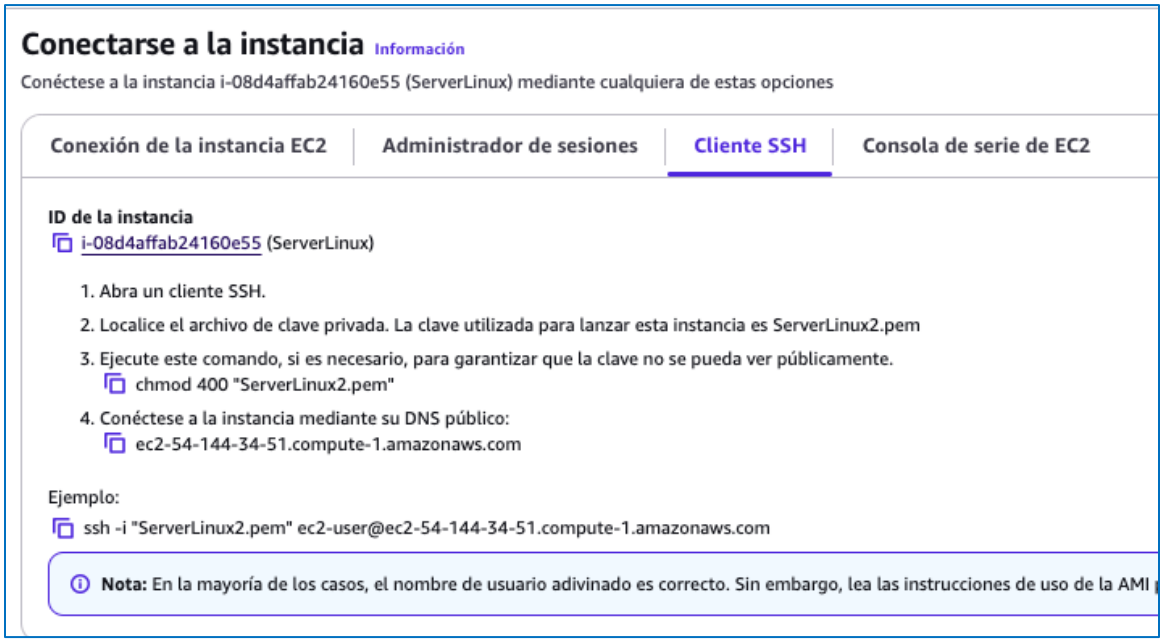


Ilustración 28 – Conexión Instancia Linux

Procedemos a copiar la dirección pública.

Ilustración 29 – Validación de IP pública Linux

6. Procedimiento de Acceso

- **Linux:**
  - Cliente: Terminal
  - Comando: chmod 400 "/ruta/ServerLinux2.pem"
  - Comando: ssh -i "/ruta/ServerLinux2.pem" ec2-user@<IP-Pública>

Ilustración 30 – Ejecución de servidor Linux

En este punto ya tenemos un servidor Linux que se está ejecutando en AWS con la configuración por defecto a través de firewall en el puerto 22

- **Windows:**
  - Cliente: Escritorio Remoto (RDP)
  - Usar IP pública e ingresar usuario/contraseña proporcionados por AWS

Consideraciones de Seguridad:

- Usar archivos .pem protegidos (solo lectura).
- No compartir contraseñas ni llaves privadas.
- Limitar rangos de IP si es posible en el grupo de seguridad.

7. Configuración del Servidor Web

En Windows Server (IIS):

1. Se accedió a la instancia Windows a través de Escritorio Remoto (RDP).
2. Desde el menú Inicio, se abrió el Server Manager.
3. Se hizo clic en Manage > Add Roles and Features.
4. En el asistente, se seleccionó la opción Web Server (IIS) como rol a instalar.
5. Se completó el asistente con las configuraciones por defecto y se inició la instalación.
6. Una vez instalada la función, se verificó que el servicio IIS estuviera activo.
7. Finalmente, se accedió a la IP pública de la instancia desde un navegador (http://18.209.87.72), confirmando que se muestra correctamente la página de bienvenida de IIS.

En Amazon Linux:
para este proceso usamos un gestor de paquetes dnf

1. Conectarse por SSH.
2. Ejecutar: sudo su para entrar como superusuario
3. Instalar Apache: dnf install httpd
4. Iniciar servicio: systemctl start httpd
5. Verificar estado del servicio: systemctl status httpd
6. Habilitar inicio automático: systemctl enable httpd
7. Verificar acceso desde navegador con IP pública.

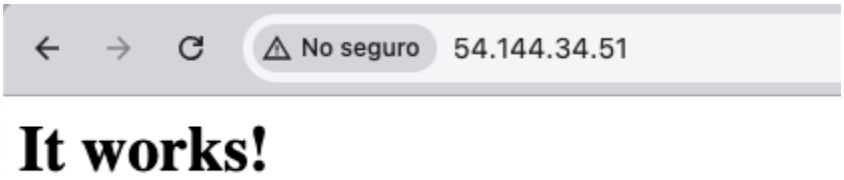
```
[root@ip-10-0-8-232 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: active (running) since Sat 2025-05-03 22:01:30 UTC; 42s ago
       Docs: man:httpd.service(8)
   Main PID: 28405 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
      Tasks: 177 (limit: 1111)
     Memory: 12.9M
        CPU: 79ms
     CGroup: /system.slice/httpd.service
             ├─28405 /usr/sbin/httpd -DFOREGROUND
             ├─28425 /usr/sbin/httpd -DFOREGROUND
             ├─28427 /usr/sbin/httpd -DFOREGROUND
             ├─28428 /usr/sbin/httpd -DFOREGROUND
             └─28429 /usr/sbin/httpd -DFOREGROUND

May 03 22:01:30 ip-10-0-8-232.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
May 03 22:01:30 ip-10-0-8-232.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
May 03 22:01:30 ip-10-0-8-232.ec2.internal httpd[28405]: Server configured, listening on: port 80
[root@ip-10-0-8-232 ec2-user]#
```

Ilustración 31 – Validación de sistema de apache

Ahora procederemos a validar la configuración de los grupos de seguridad asociados a nuestra instancia EC2. Es necesario asegurarse de que en las reglas de entrada se encuentre agregada una regla de tipo HTTP (puerto 80), la cual permitirá el acceso al servidor web desde cualquier ubicación. Para ello, en el campo de IP de origen se debe especificar 0.0.0.0/0, lo que habilita el acceso público desde Internet.

Ilustración 32 – Reglas de entrada Linux

Verificar que se este ejecutando el servidor web:



It works!

Ilustración 33

Configuración de plantilla html(Opcional):

1. Ejecutar el comando: cd /var/www/html
2. Usa wget para descargar el archivo (por ejemplo, un HTML de prueba): wget https://html5up.net/massively/download
3. En caso de que se descargue un archivo comprimido, ejecutar: unzip download
4. Verificar acceso desde navegador con IP pública. Validamos que el archivo HTML descargado esté correctamente montado y accesible ingresando la IP pública en el navegador; si la configuración del servidor Apache es correcta, el archivo será renderizado como página principal del sitio.

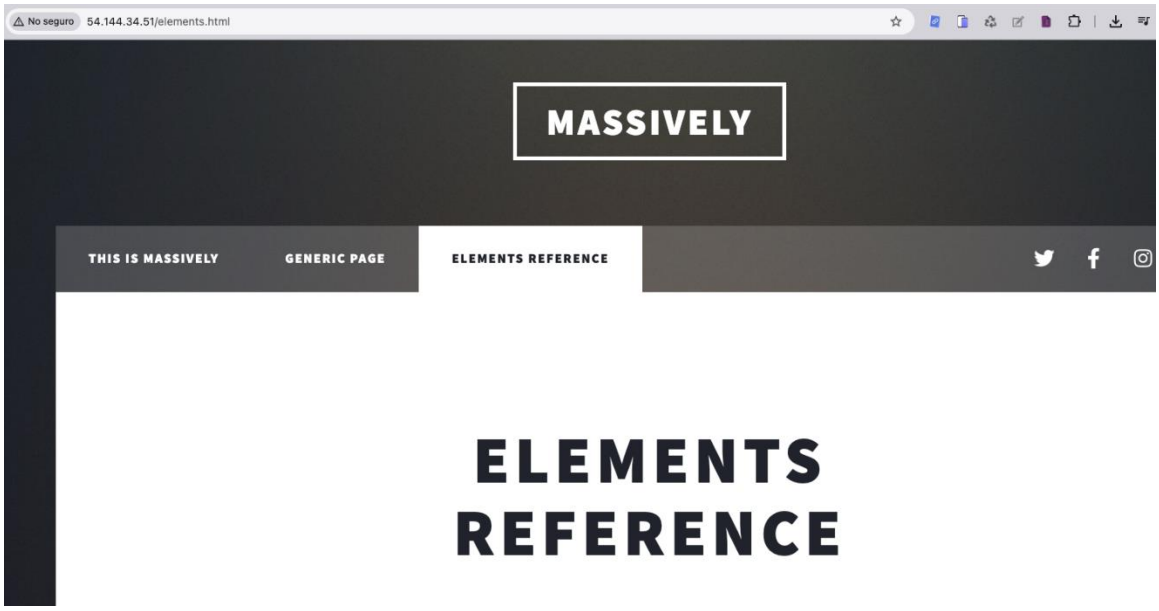


Ilustración 34 – Acceso público desde navegador

8. Pruebas de Conectividad

Se realizaron pruebas de conectividad utilizando el comando ping desde ambas instancias. Desde el servidor Linux se verificó la conexión mediante la IP privada de la instancia Windows, y viceversa, confirmando así la comunicación exitosa entre ambos servidores dentro de la misma VPC.

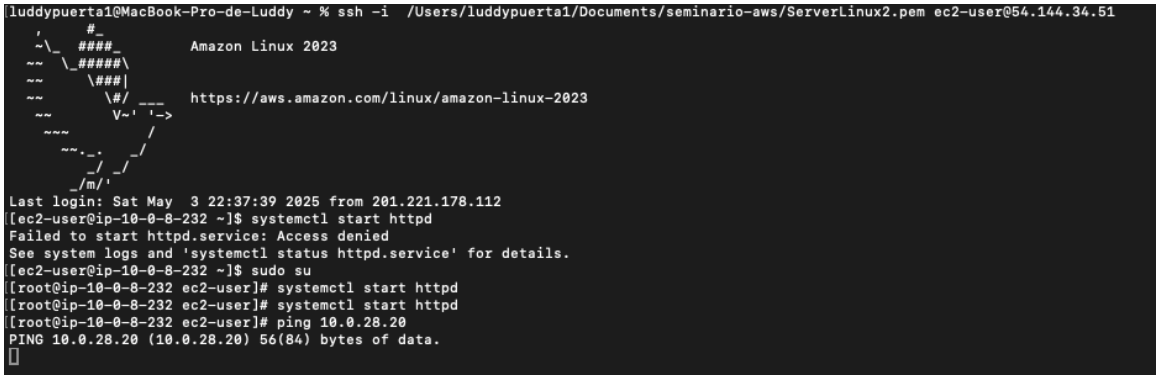


Ilustración 35 – Pruebas de conectividad desde Linux a IP de Windows

Al momento de validar el ping desde la instancia Windows hacia la instancia Linux, inicialmente no se recibía respuesta, mostrando un 100% de pérdida de paquetes. Tras revisar la configuración, se identificó que era necesario habilitar el tráfico ICMP en el grupo de seguridad de la instancia Linux.

Para solucionarlo, se realizaron los siguientes pasos:

1. Se accedió a la consola de AWS > EC2 > Grupos de seguridad.
2. Se localizó el grupo de seguridad asignado a la instancia Linux (SG-ServerLinux2).
3. En la pestaña Reglas de entrada, se hizo clic en Editar reglas.
4. Se agregó una nueva regla con los siguientes parámetros:
   - Tipo: Todos los ICMP – IPv4
   - Protocolo: Automático
   - Puerto: —
   - Origen: 0.0.0.0/0 (usado para fines de prueba; se puede restringir a la IP privada de la instancia Windows para mayor seguridad).

Ilustración 36 – Edición de reglas de entrada Linux

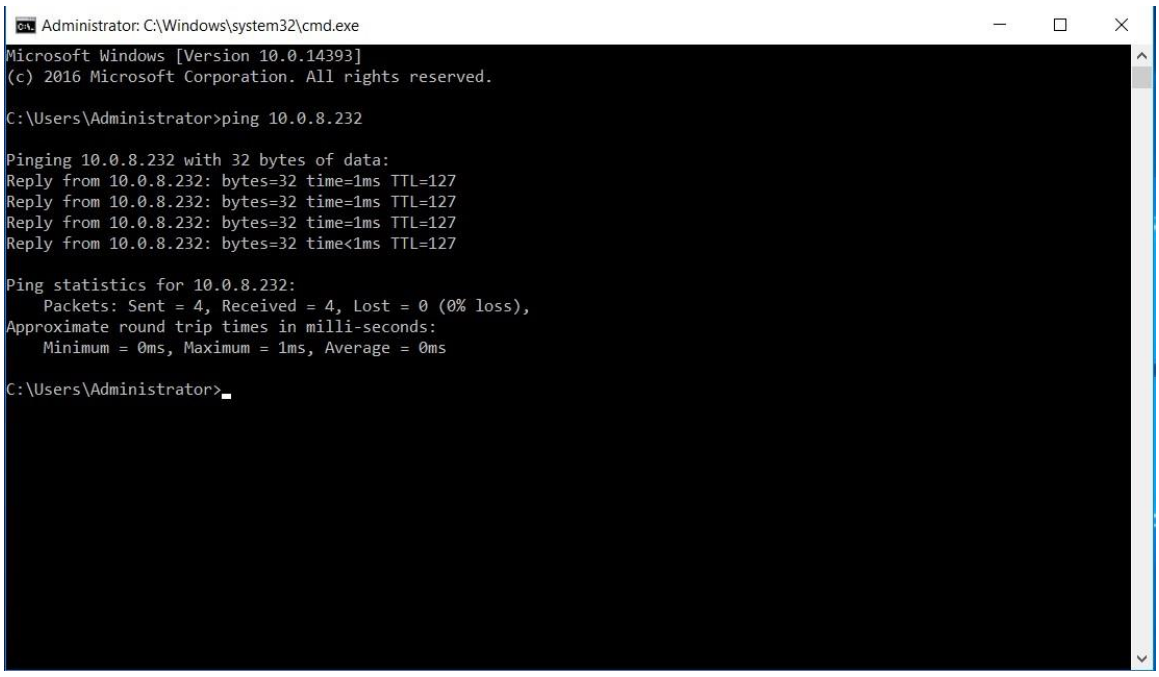


Ilustración 37 – Comprobación de conectividad desde Linux

9. Validación de acceso web

- Acceso a http://54.144.34.51 muestra correctamente el archivo HTML personalizado descargado previamente en el servidor Linux. Este archivo fue

ubicado en la ruta /var/www/html/ y es servido correctamente por Apache, confirmando que el servidor web está operativo y accesible desde Internet.

- Acceso a http://18.209.87.72 con esta IP está mostrando correctamente la página de bienvenida de IIS en el servidor Windows, confirmando que el servidor web está operativo y accesible desde Internet.

10. Video de la implementación del vpc y las instancias:

URL Youtube: https://youtu.be/NtVTYDJC0hU

Implementación de Servicios de Docker con pruebas de estrés

Implementación de Docker

1 **dnf install docker** instala docker

```
[root@ip-10-0-8-232 ~]# dnf install docker
Last metadata expiration check: 1 day, 23:19:38 ago on Mon May 19 00:47:09 2025.
Dependencies resolved.
================================================================================
 Package                 Architecture  Version                    Repository  Size
================================================================================
Installing:
 docker                  x86_64        25.0.8-1.amzn2023.0.3      amazonlinux  45 M
Installing dependencies:
 container-selinux       noarch        3:2.233.0-1.amzn2023       amazonlinux  55 k
 containerd              x86_64        1.7.27-1.amzn2023.0.2      amazonlinux  37 M
 iptables-libs           x86_64        1.8.8-3.amzn2023.0.2       amazonlinux 401 k
 iptables-nft            x86_64        1.8.8-3.amzn2023.0.2       amazonlinux 183 k
 libcgroup               x86_64        3.0-1.amzn2023.0.1         amazonlinux  75 k
 libnetfilter_conntrack  x86_64        1.0.8-2.amzn2023.0.2       amazonlinux  58 k
 libnfnetlink            x86_64        1.0.1-19.amzn2023.0.2      amazonlinux  30 k
 libnftnl                x86_64        1.2.2-2.amzn2023.0.2       amazonlinux  84 k
 pigz                    x86_64        2.5-1.amzn2023.0.3         amazonlinux  83 k
 runc                    x86_64        1.2.4-1.amzn2023.0.1       amazonlinux 3.4 M

Transaction Summary
================================================================================
Install  11 Packages

Total download size: 86 M
Installed size: 324 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): container-selinux-2.233.0-1.amzn2023.noarch.rpm   1.3 MB/s |  55 kB  00:00
(2/11): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm     2.9 MB/s | 401 kB  00:00
(3/11): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm      2.2 MB/s | 183 kB  00:00
(4/11): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm           1.4 MB/s |  75 kB  00:00
(5/11): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm 1.5 MB/s |  58 kB  00:00
(6/11): libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64.rpm     973 kB/s |  30 kB  00:00
(7/11): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm          1.9 MB/s |  84 kB  00:00
(8/11): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                2.5 MB/s |  83 kB  00:00
(9/11): runc-1.2.4-1.amzn2023.0.1.x86_64.rpm               14 MB/s | 3.4 MB  00:00
(10/11): containerd-1.7.27-1.amzn2023.0.2.x86_64.rpm       42 MB/s |  37 MB  00:00
(11/11): docker-25.0.8-1.amzn2023.0.3.x86_64.rpm           40 MB/s |  45 MB  00:01
--------------------------------------------------------------------------------
Total                                                      75 MB/s |  86 MB  00:01
Running transaction check
```

Ilustración 38 – Instaccion docker

2. **systemctl status docker** se valida si esta activo

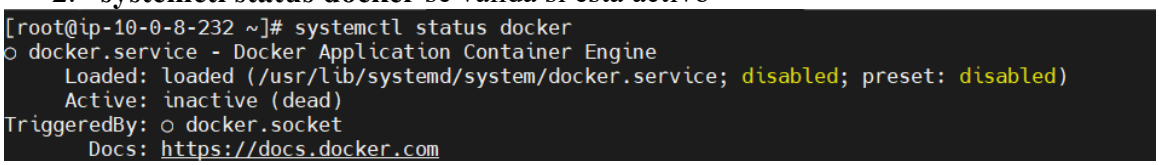


Ilustración 39 – Validación del estado de Docker

3. **systemctl start docker** se inicializa Docker

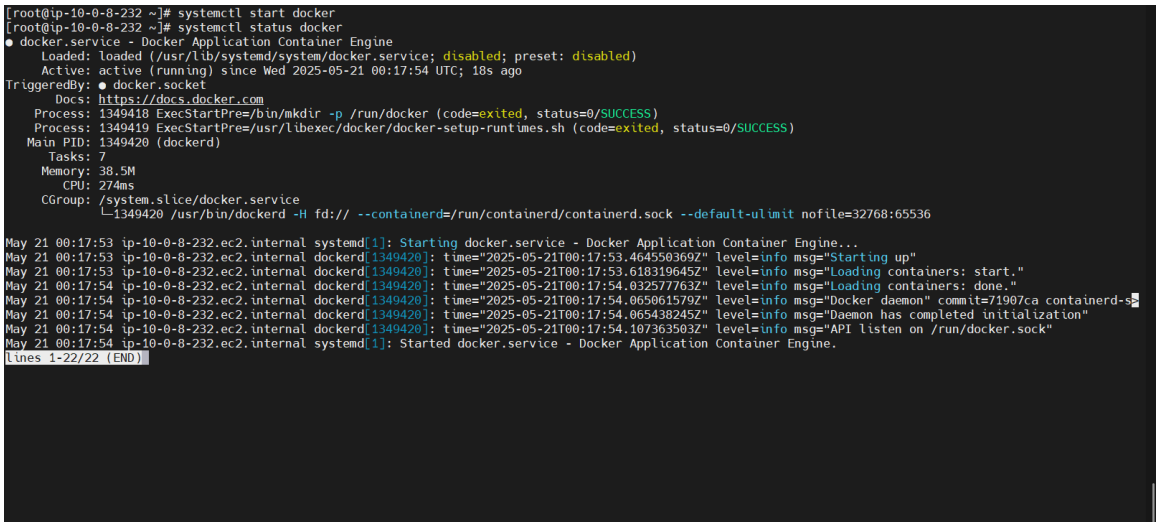


Ilustración 40 – Inicialización de Docker

4.**docker pull httpd** se instala la imagen de apache

```
[root@ip-10-0-8-232 ~]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
254e724d7786: Pull complete
10d01782dc02: Pull complete
4f4fb700ef54: Pull complete
4ceeea7b3d76: Pull complete
0ff470512d2f: Pull complete
ba78a05e3b3c: Pull complete
Digest: sha256:c11efd67f6308f2c25965e4e9d13ded15e7c45c0367b95f619a16e03c6c1e2b1
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@ip-10-0-8-232 ~]#
[root@ip-10-0-8-232 ~]#
```

Ilustración 41 – Instalación de imagen de apache

5. **docker images** se valida que la imagen esté instalada

```
[root@ip-10-0-8-232 ~]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    0208f149a449   3 months ago   148MB
[root@ip-10-0-8-232 ~]#
```

Ilustración 42 – Validación de instalación de la imagen en Docker

6. **docker run -dit --name app1 -p 8080:80 httpd** se crea el primer contenedor

```
[root@ip-10-0-8-232 ~]# docker run -dit --name app1 -p 8080:80 httpd
d0f3fc84d35c0c78690c96e2690e1295d5ee3c3b315d1b0929b5680a07b5149e
[root@ip-10-0-8-232 ~]#
```

Ilustración 43 – Creación de primer contenedor

7. **docker ps** Se valida que se creo

```
[root@ip-10-0-8-232 ~]# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED              STATUS              PORTS                                          NAMES
d0f3fc84d35c   httpd     "httpd-foreground"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::8080->80/tcp          app1
[root@ip-10-0-8-232 ~]#
```

Ilustración 44 – Validación de creación de contenedor

8. Se crea nueva regla de seguridad

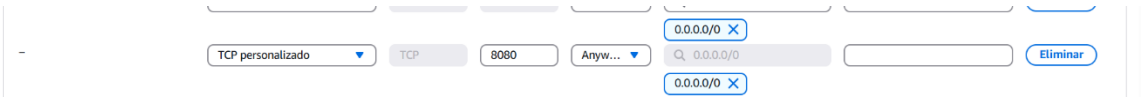


Ilustración 45 – creación de nueva regla de seguridad

9. Se realiza la prueba en el navegador

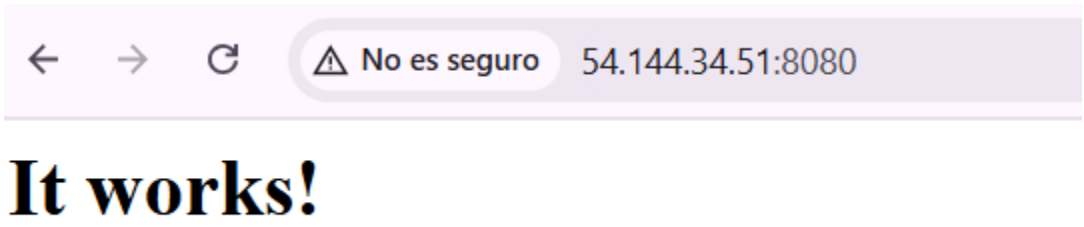


Ilustración 46 – comprobación de conexión

10. Se realiza la descarga de una plantilla

```
[root@ip-10-0-8-232 app1]# wget https://html5up.net/massively/download
--2025-05-21 02:28:33--  https://html5up.net/massively/download
Resolving html5up.net (html5up.net)... 172.67.195.190, 104.21.76.136, 2606:4700:3030::6815:4c88, ...
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                                [ <=>

2025-05-21 02:28:33 (56.3 MB/s) - 'download' saved [2065870]
```

Ilustración 47 – Descarga de plantilla

11. Se implementa en un contenedor

```
[root@ip-10-0-8-232 ec2-user]# docker run -dit --name app5 -p 8084:80 -v /home/ec2-user/app1/:/usr/local/apache2/htdocs  httpd
ff14e83129a1146b9c9fc323652387d523e863e2316e74769c9efee1fd17ca82
[root@ip-10-0-8-232 ec2-user]# ls
LICENSE.txt  README.txt  app1  assets  download  elements.html  generic.html  images  index.html
```

Ilustración 48 – Implementación Docker

12. Se direccionan todos los contenedores a una a la plantilla descargada

```
[root@ip-10-0-8-232 ec2-user]# docker run -dit --name app4 -p 8083:80 -v /home/ec2-user/:/usr/local/apache2/htdocs httpd
4abd8ec048be73420160dff1c0692dbebc2197907e06178fbc865bc371f3530f
[root@ip-10-0-8-232 ec2-user]# docker run -dit --name app3 -p 8082:80 -v /home/ec2-user/:/usr/local/apache2/htdocs httpd
bd2ca99ccbb7b206760fc61634d016f4d0967c6606de7ffc0c58846fcbc34a04
[root@ip-10-0-8-232 ec2-user]# docker run -dit --name app2 -p 8081:80 -v /home/ec2-user/:/usr/local/apache2/htdocs httpd
f3961a9c82b8ddba036eb902b2e0c039e7d590f3d93869be912e7e2e0ee50fcf
```

Ilustración 49 – Direccionamiento de contendores a la plantilla

13. Todos los contenedores direccionados a esta page

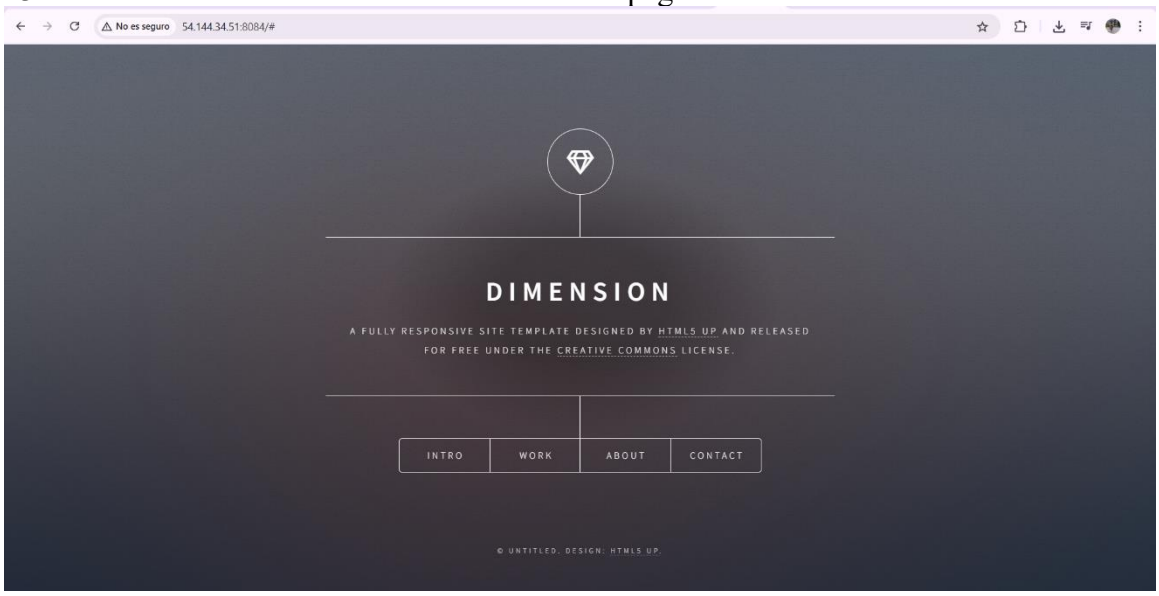


Ilustración 50 – Comprobación de contenedores en la plantilla descargada

13. Se descarga nginx, se le cambia el puerto de 80 a 8080, ya que tenía ocupado ese puerto con Docker

```
[root@ip-10-0-8-232 ec2-user]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset: disabled)
     Active: active (running) since Wed 2025-05-21 15:08:46 UTC; 1min 27s ago
    Process: 1463597 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 1463598 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 1463599 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 1463600 (nginx)
      Tasks: 2 (limit: 1111)
     Memory: 2.5M
        CPU: 48ms
     CGroup: /system.slice/nginx.service
             ├─1463600 "nginx: master process /usr/sbin/nginx"
             └─1463601 "nginx: worker process"
```

Ilustración 51 – Descarga nginx

14 nginx !

Ilustración 52 - Comprobación Docker

15. Se Edita el archivo de configuración

```
  GNU nano 8.3                                          /etc/nginx/nginx.conf
events{}

http{

     upstream seminario{

      server localhost:8081;
      server localhost:8082;
      server localhost:8083;
      server localhost:8084;

}
     server {

            listen 8080;
            server_name seminario;
            location / {
                        proxy_pass http://seminario;
}

}

}
```

Ilustración 53 – Edición de archivo de configuración

16. Se guarda y reinicia, ahora funciona el balanceador

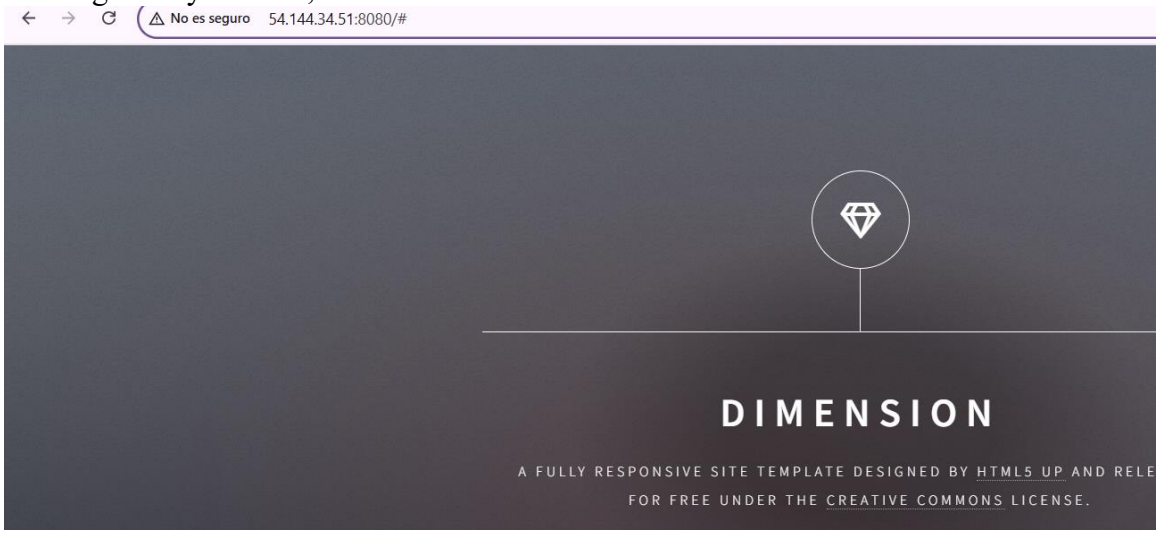


Ilustración 54 – Comprobación de balanceador

17. Evidencia de CPU con los cuatro contenedores

```
[root@ip-10-0-8-232 ec2-user]# docker stats
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
f3961a9c82b8   app2      0.00%     9.418MiB / 949.4MiB   0.99%     942kB / 22MB      147kB / 4.1kB     82
bd2ca99ccbb7   app3      0.00%     8.871MiB / 949.4MiB   0.93%     928kB / 21.9MB    0B / 4.1kB        82
4abd8ec048be   app4      0.00%     8.93MiB / 949.4MiB    0.94%     934kB / 22.3MB    0B / 4.1kB        82
53db62c4fba0   app5      0.00%     9.402MiB / 949.4MiB   0.99%     1.04MB / 24.1MB   0B / 4.1kB        82
```

Ilustración 55 – Evidencia CPU en contenedores

18. Se realiza prueba de estrés de carga ab -n 1000 -c 100 http://54.144.34.51:8080/

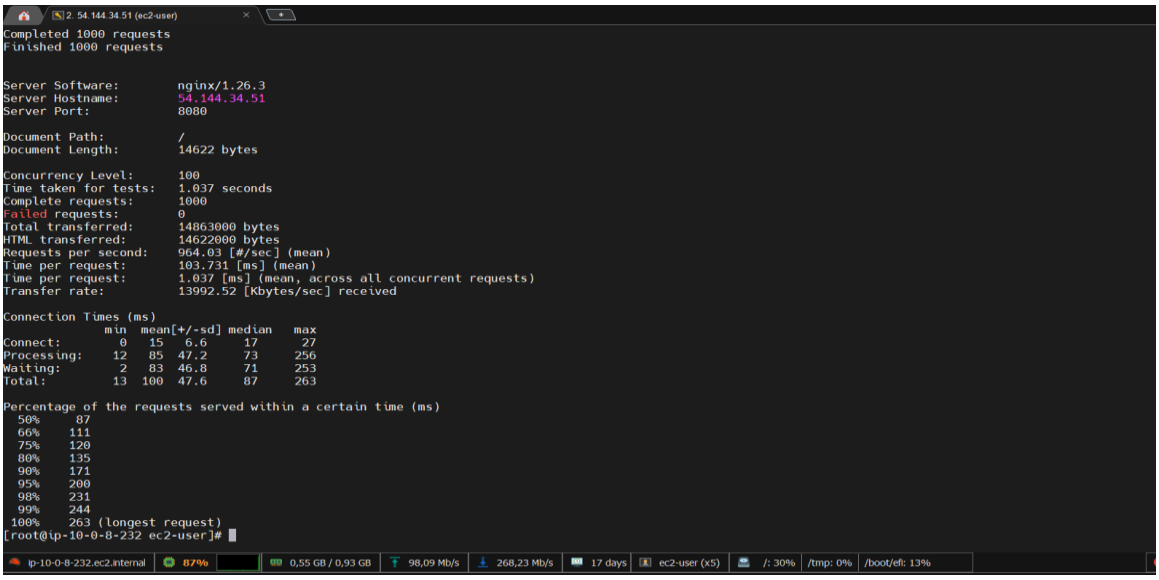


Ilustración 56 – evidencia de pruebas de estrés

Se evidenció un 87% de uso de cpu con ab -n 1000 -c 100 http://54.144.34.51:8080/

- Se completaron 1000 peticiones correctamente.
- ¡Ninguna petición falló! Tu servidor respondió bien.
- Capacidad de manejar casi **946 peticiones por segundo**. Excelente rendimiento.
- Cada petición tomó en promedio **105 ms** en completarse.
- Transferencia promedio de **13.7 MB por segundo**.

VENTAJAS

Los contenedores como Docker son más livianos que las máquinas virtuales porque no cargan un sistema operativo entero solo lo necesario para correr la app también se prenden rápido en segundos con poco hardware podemos correr varios y es fácil moverlos de un server a otro si quieres escalar o actualizar también es más simple solo se cambia la imagen y reinicias mucho más rápido que usar VMs, q son más pesadas y lentas.

Tabla 1 Tiempos de respuesta de prueba de estrés.

Percentil	Tiempo Máximo	Explicación
50%	96 ms	La mitad de las peticiones fueron más rápidas que 96 ms.
90%	144 ms	El 90% respondió en menos de 144 ms.
100%	282 ms	La más lenta tardó 282 ms.

Balanceador de carga y contenedores respondieron correctamente a la prueba de estrés con buen rendimiento se puede aumentar de contenedores a unos 20, pero quedaría muy lento y Nginx no le quedaría espacio para el funcionamiento con él serían unos 10 a 12 contenedores, pero con los utilizados en la prueba “4” el funcionamiento es óptimo.

Conclusiones

Con este proyecto pudimos entender como crear e implementar desde cero una red en la nube utilizando los servicios de AWS, con las instancias de EC2 para Linux y Windows, configuradas para acceder públicamente desde internet y con su respectiva comunicación entre sí, Desde la creación de un VPC con su tabla de enrutamiento, grupos de seguridad y conexión remota, además del aprendizaje de las configuraciones de servidores web y pruebas de conectividad y carga, con todo esto nos acercó más a un aprendizaje amplio en como funciona las infraestructuras modernas en la nube.

Uno de los grandes aprendizajes fue comprender la diferencia entre virtualización tradicional, como instalar un sitio web directamente sobre un sistema operativo (Linux o Windows), y el uso de contenedores. Pudimos evidenciar a través de las pruebas de estrés, la eficiencia y la escalabilidad que es usar una infraestructura más sólida los contenedores para que con estos se puedan usar y responder de una manera más flexible ante volúmenes de tráfico más grandes.

Ante este aprendizaje nos proyectamos como futuros profesionales en cualquier organización de desarrollo de software poder entender y manejar este tipo de infraestructuras, ya que no es un valor agregado, sino ya una necesidad para dar manejo a soluciones en funciones avanzadas y así poder ejecutar sistemas completamente en plataformas como AWS u otro según cada necesidad de la organización; todo esto con el fin de ser parte de una transformación digital y estar a la vanguardia de las mejores soluciones tecnológicas.

Referencias

Guijarro Olivares, Jordi. (2019). *DevOps y seguridad cloud*. 310.

IMECAF. (2024, August). *Computación en la nube*. Instituto Mexicano de Contabilidad, Administración y Finanzas. https://imecaf.com/blog/wp-content/uploads/Computacion-en-la-nube-Cloud-Computing.pdf

Serverspace. (n.d.). *¿Qué es el estrés en Docker y por qué lo necesito?* Retrieved May 21, 2025, from https://serverspace.io/support/help/what-is-stress-in-docker-and-why-do-i-need-it/