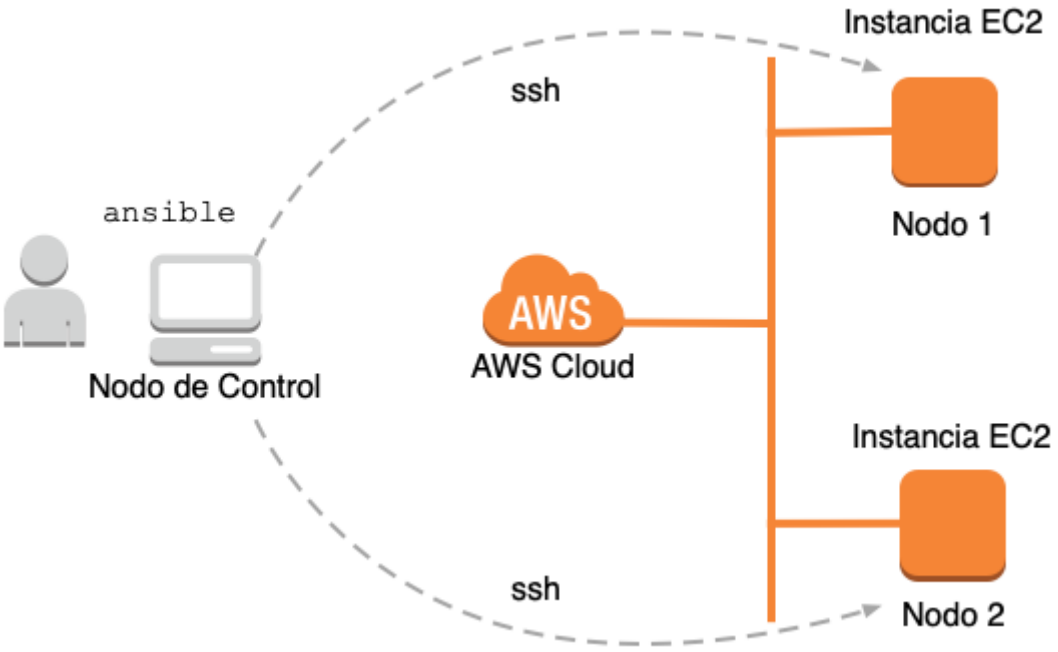
ADMINISTRACIÓN REMOTA DE SERVIDORES WEB EN AWS A TRAVÉS DE SSH



Autor: Diestefano Jiampieri
Asignatura: Servicios de Red e Internet
Tutor: Isaac
Fecha de Entrega: 04/12/2025

ÍNDICE

PREPARACIÓN DEL ENTORNO LOCAL

1. Para empezar la práctica comprobamos que tenemos WSL instalado desde la terminal, "*wsl --version*".

```
C:\Users\Alumno.DESKTOP-DI5KTUG>wsl --version
Versión de WSL: 2.6.1.0
Versión de kernel: 6.6.87.2-1
Versión de WSLg: 1.0.66
Versión de MSRDC: 1.2.6353
Versión de Direct3D: 1.611.1-81528511
Versión de DXCore: 10.0.26100.1-240331-1435.ge-release
Versión de Windows: 10.0.26100.5074
```

2. Cuando sepamos que tenemos WSL, entramos en el modo Linux, "*wsl*".

```
C:\Users\Alumno.DESKTOP-DI5KTUG>wsl
dies@A6Alumno14:/mnt/c/Users/Alumno.DESKTOP-DI5KTUG$
```

3. Ahora crearemos el entorno, creamos la carpeta oculta, "*mkdir -p ~/.ssh*".

```
dies@A6Alumno14:/mnt/c/Users/Alumno.DESKTOP-DI5KTUG$ mkdir -p ~/.ssh
```

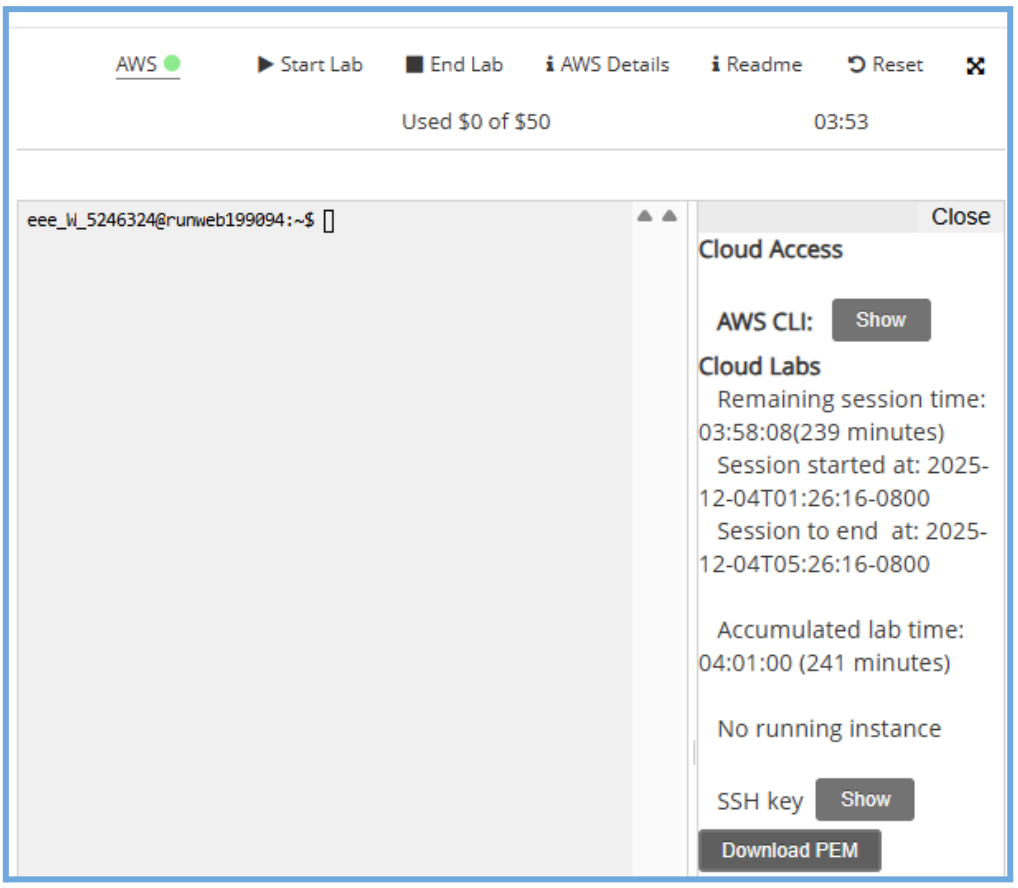
4. Una vez creada la carpeta le pondremos los permisos necesarios para almacenar claves, "*chmod 700 ~/.ssh*".

```
dies@A6Alumno14:/mnt/c/Users/Alumno.DESKTOP-DI5KTUG$ chmod 700 ~/.ssh
```

CONFIGURACIÓN EN AWS

1. Aquí conectaremos el mundo real con el mundo virtual.

2. Empezaremos descargando la clave PEM desde la página del laboratorio, le daremos a *Start Lab*, esperamos que se ponga verde, una vez en verde le damos a *AWS Details*, donde se nos desplegará una ventana donde podremos descargar la clave PEM. Se nos descargará un archivo que se llamará "*labsuser.pem*".



3. Cuando lo tengamos descargado volvemos a la terminal de WSL, y moveremos la llave de Windows a la carpeta segura de Linux, "cp /mnt/c/Users/Alumno.DESKTOP-DI5KTUG/Downloads/labsuser.pem ~/.ssh/".

```
dies@A6Alumno14:/mnt/c/Users/Alumno.DESKTOP-DI5KTUG$ cp /mnt/c/Users/Alumno.DESKTOP-DI5KTUG/Download
h/
```

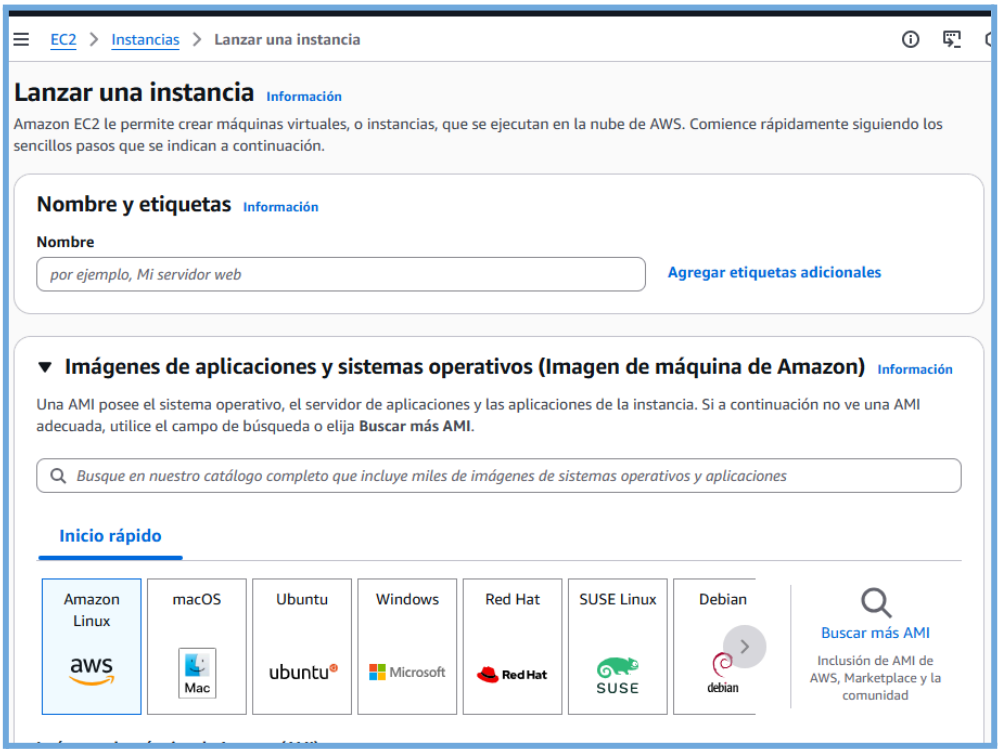
4. Le pondremos un candado de seguridad a la llave, "chmod 400 ~/.ssh/labsuser.pem".

```
dies@A6Alumno14:/mnt/c/Users/Alumno.DESKTOP-DI5KTUG$ chmod 400 ~/.ssh/labsuser.pem
```

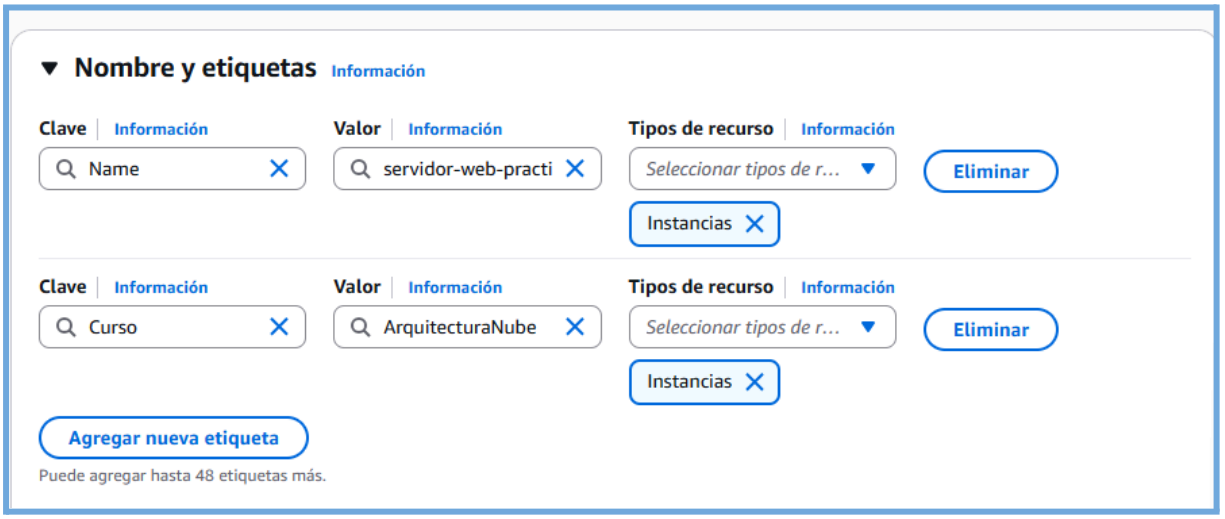
5. Revisamos que la llave es segura con, “ls -la ~/.ssh/labsuser.pem”, el comando nos tiene que responder con algo parecido a *-r-------- ….*

```
dies@A6Alumno14:/mnt/c/Users/Alumno.DESKTOP-DI5KTUG$ ls -la ~/.ssh/labsuser.pem
-r-------- 1 dies dies 1674 Dec  4 10:48 /home/dies/.ssh/labsuser.pem
```

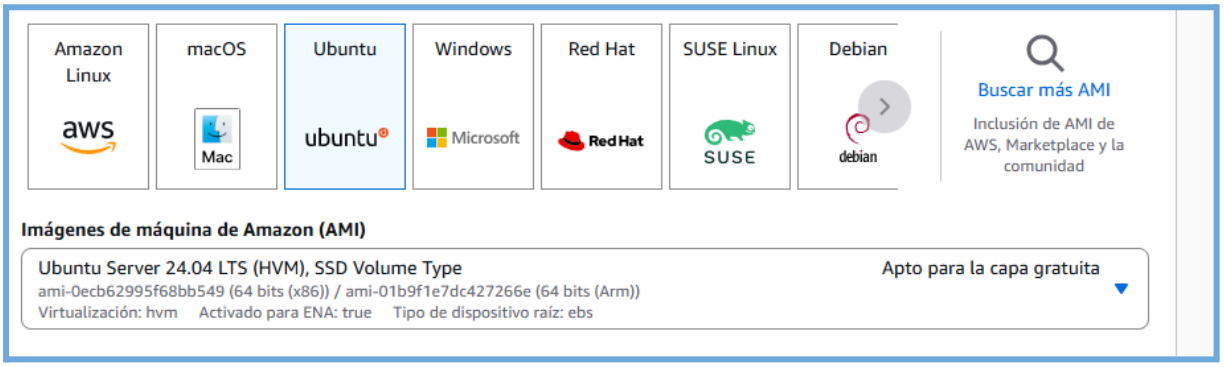
6. Tras la revisión vamos a crear una instancia EC2, le damos primero donde pone AWS y nos mandará a la consola de amazon, una vez ahí le damos a buscar **ponemos instancia EC2** y le damos a lanzar.



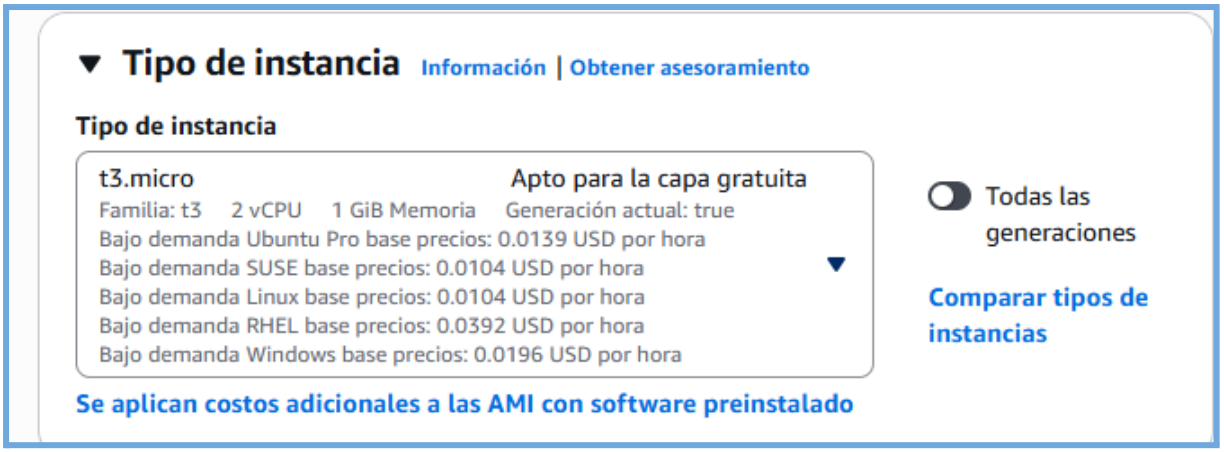
7. Una vez con la instancia creada le ponemos estas **dos etiquetas**.



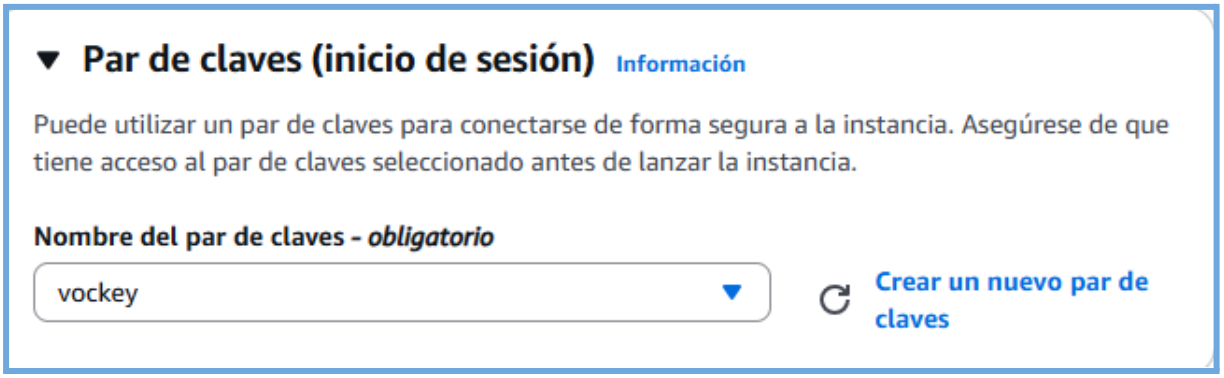
8. Luego en cuanto al sistema operativo seleccionaremos **Ubuntu Server 24.04 LTS**.



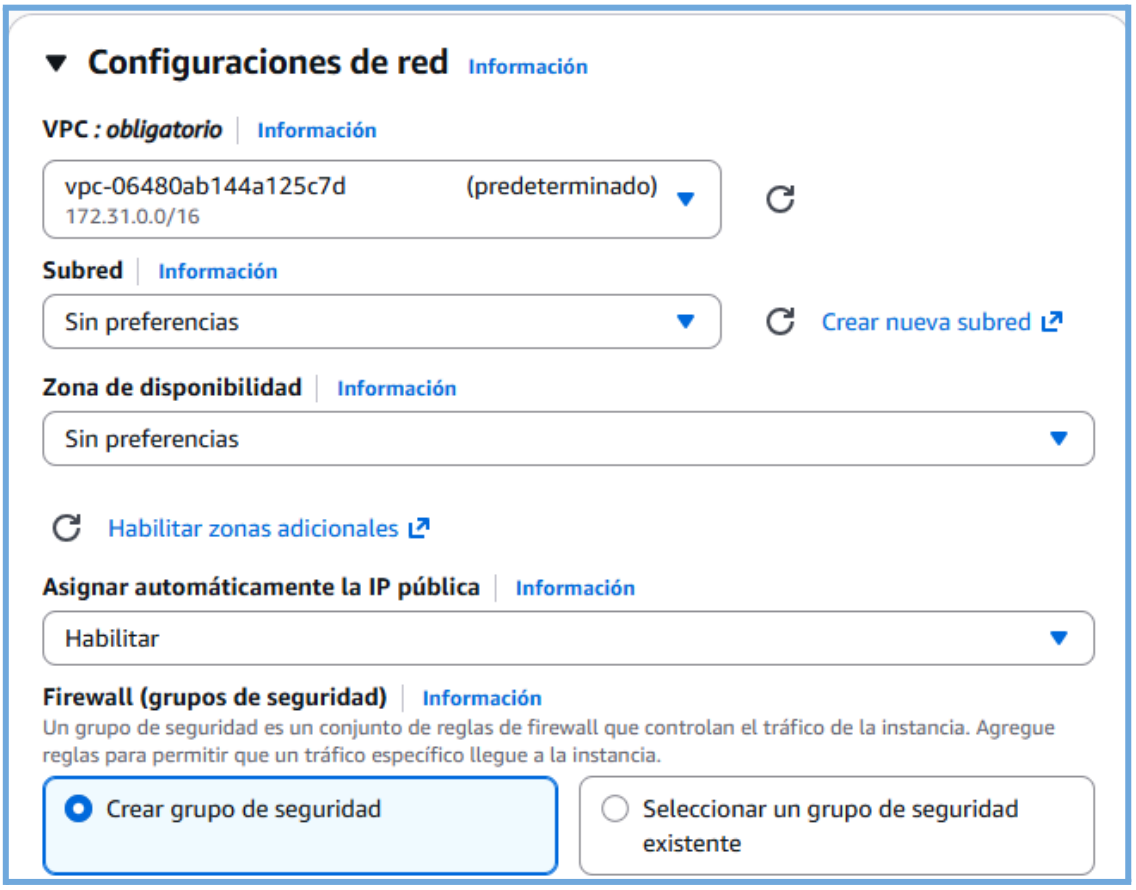
9. En qué tipo de instancia seleccionaremos **t3.micro**.

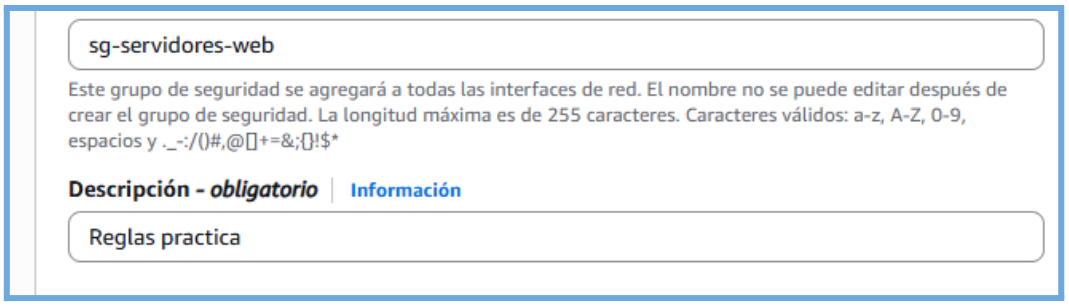


10. En el apartado de de Par de claves, seleccionaremos **vockey**.



11. En la configuración de red, lo dejamos como en las **siguientes fotos**.

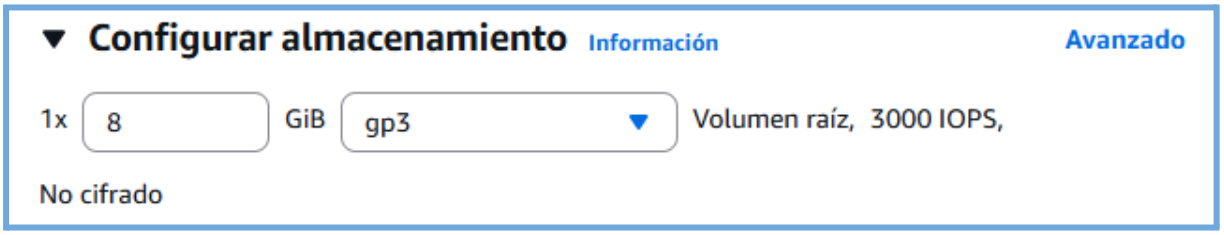




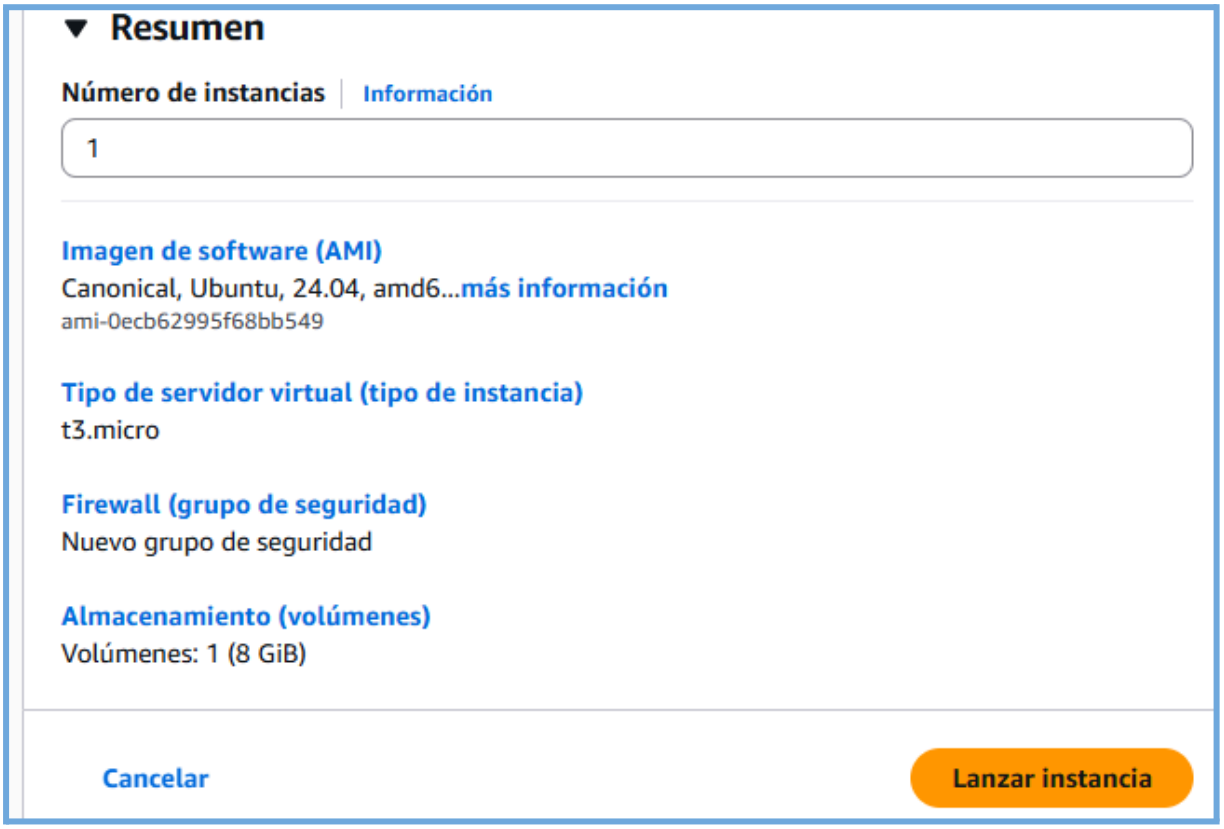
12. En las reglas de entrada pondremos **las siguientes**:

Puerto	Protocolo	Tipo	Descripción
22	TCP	SSH	Acceso SSH remoto
8080	TCP	HTTP personalizado	Apache HTTP
8081	TCP	HTTP personalizado	Nginx
8082	TCP	HTTP personalizado	Caddy
8443	TCP	HTTPS personalizado	Apache HTTPS (SSL)

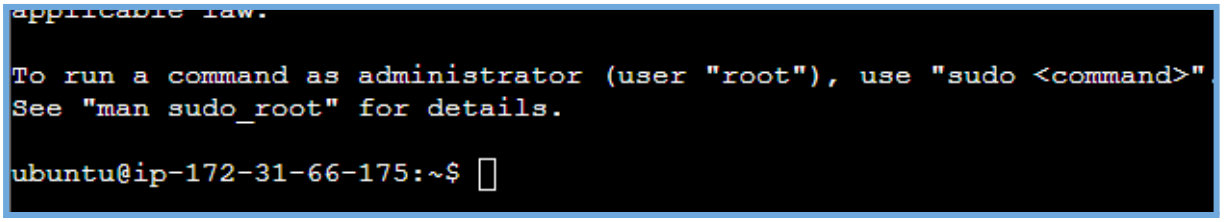
13. Para la última configuración que es el almacenamiento, pondremos **8 GiB y gp3**.



14. Revisamos todo y lanzamos la instancia.



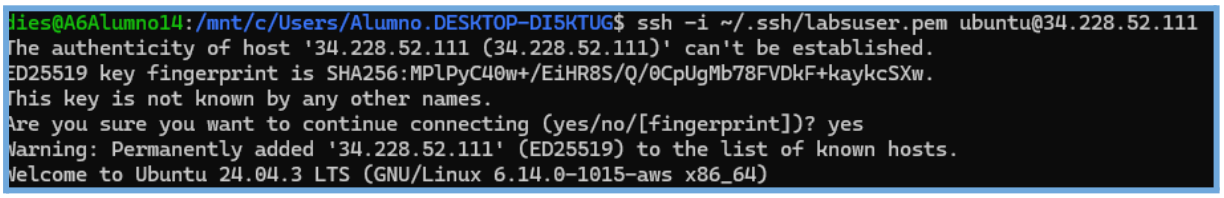
15. Una vez teniendo la instancia lanzada, le daremos a conectar y se nos abrirá la terminal.



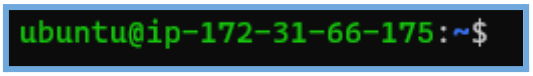
CONEXIÓN SSH DESDE WSL A AWS

1. Conectaremos mediantes SSH WSL a AWS.

2. Teniendo la IP de AWS, nos dirigimos a la terminal de WSL y ponemos, "*ssh -i ~/.ssh/labsuser.pem ubuntu@34.228.52.111*". Al final ponemos la IP de la instancia pública.



3. Ya está conectado.



EJECUTAR LA SEGUNDA PRÁCTICA

Tendremos que ejecutar todos los comandos de la práctica dos.

SERVIDOR APACHE CON PHP

1. Empezaremos actualizando el servicio, para mejorar el sistema a las últimas versiones, usaremos "*sudo apt update && sudo apt upgrade -y*".

```
ubuntu@ip-172-31-66-175:~$ sudo apt update && sudo apt upgrade -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
```

2. Instalamos Apache2, para ello ponemos *"sudo apt install apache2 -y*".

```
ubuntu@ip-172-31-66-175:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

3. Configuraremos Apache en el puerto 8080, "*sudo nano /etc/apache2/ports.conf*". En la configuración cambiaremos ***listen 80*** por ***listen 8080.***

```
ubuntu@ip-172-31-66-175:~$ sudo nano /etc/apache2/ports.conf
```

```
  GNU nano 7.2                                  /etc/apache2/ports.conf *
# If you just change the port or add more ports here, you will like
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8080

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>
```

4. Modificaremos el archivo de configuración del sitio por defecto para que también use el nuevo puerto, con "*sudo nano /etc/apache2/sites-available/000-default.conf*". Una vez dentro cambiamos ***VirtualHost *:80*** por ***VirtualHost *:8080***.

```
ubuntu@ip-172-31-66-175:~$ sudo nano /etc/apache2/sites-available/000-default.conf
```

```
  GNU nano 7.2                    /etc/apache2/sites-available/000-default.conf *
<VirtualHost *:8080>
```

5. Ahora instalamos PHP, con el siguiente comando "*sudo apt install php libapache2-mod-php -y*".

```
ubuntu@ip-172-31-66-175:~$ sudo apt install php libapache2-mod-php -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

6. Reiniciamos PHP, así aplicaremos todos los cambios de configuración, usaremos "*sudo systemctl restart apache2*".

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl restart apache2
```

7. Verificamos su funcionamiento con *"sudo systemctl status apache2*" y observaremos si funciona todo en el puerto 8080 usando "*sudo netstat -tulpn | grep 8080*".

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-12-04 13:12:32 UTC; 24s ago
       Docs: https://httpd.apache.org/docs/2.4/
```

```
ubuntu@ip-172-31-66-175:~$ sudo netstat -tulpn | grep 8080
tcp6       0      0 :::8080                 :::*                    LISTEN      18224/apache2
```

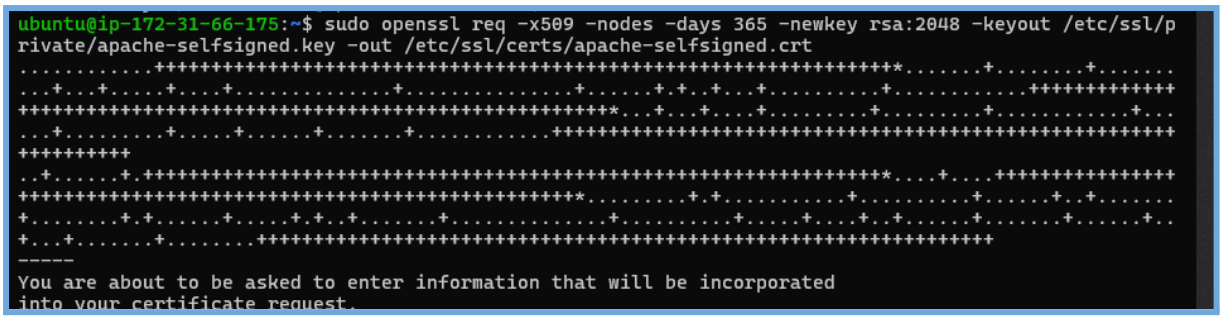
8. Continuamos creando un archivo *info.php* "*echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php*".

```
ubuntu@ip-172-31-66-175:~$ echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
<?php phpinfo(); ?>
```

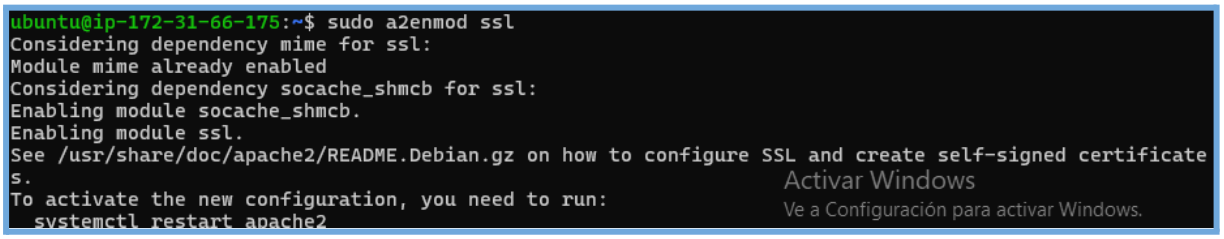
9. Accederemos a Apache desde la terminal con "*curl http://localhost:8080/info.php*".

```
ubuntu@ip-172-31-66-175:~$ curl http://localhost:8080/info.php
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<style type="text/css">
body {background-color: #fff; color: #222; font-family: sans-serif;}
pre {margin: 0; font-family: monospace;}
a:link {color: #009; text-decoration: none; background-color: #fff;}
a:hover {text-decoration: underline;}
table {border-collapse: collapse; border: 0; width: 934px; box-shadow: 1px 2px 3px rgba(0, 0, 0, 0.2);
}
.center {text-align: center;}
```

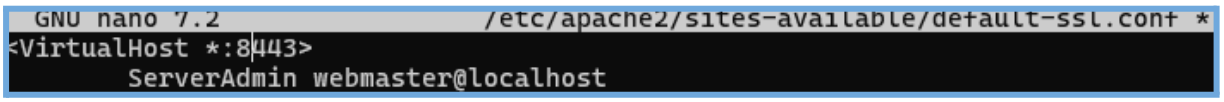
10. A continuación configuraremos HTTPS con un certificado autofirmado, para ello empezaremos poniendo "*sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt*".

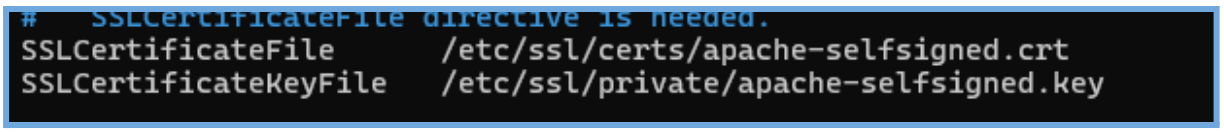


11. Apache necesita que su módulo SSL esté activado para poder manejar conexiones HTTPS, por eso mismo pondremos *"sudo a2enmod ssl".*



12. Tenemos que decirle a Apache donde debe encontrar los archivos del certificado que hemos creado, para ello abriremos el archivo de configuración SSL por defecto con "*sudo nano /etc/apache2/sites-available/default-ssl.conf*". Una vez dentro de este archivo verificamos que todo está correctamente y cambiaremos la línea ***VirtualHost _default_:443*** por ***VirtualHost *:8443***. También tenemos que cambiar las certificaciones que vienen por defecto por las que hemos creado antes.





13. Debemos decirle a Apache que tiene que escuchar el puerto 8443 para ello usaremos "*sudo nano /etc/apache2/ports.conf*" y pondremos debajo de ***listen 8080, listen 8443.***

```
ubuntu@ip-172-31-66-175:~$ sudo nano /etc/apache2/ports.conf
```

```
  GNU nano 7.2                                 /etc/apache2/ports.conf *
# If you just change the port or add more ports here, you will like
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8080
Listen 8443
```

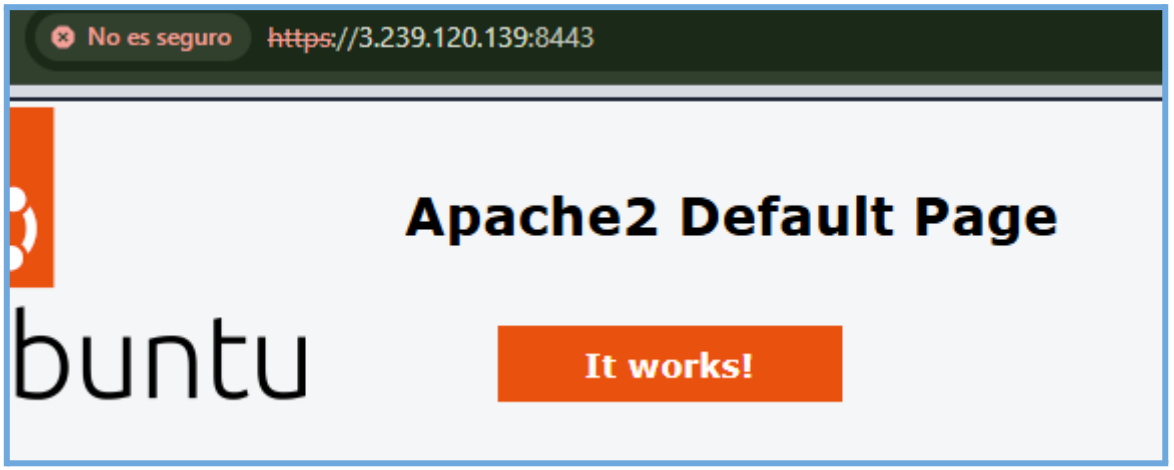
14. Casi terminando tenemos que activar la configuración SSL, con el comando "*sudo a2ensite default-ssl.conf*".

```
ubuntu@ip-172-31-66-175:~$ sudo a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

15. Reiniciamos Apache para que se apliquen todos los cambios "*sudo systemctl restart apache2*".

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl restart apache2
```

16. Para terminar esta parte, accederemos a "*https://3.239.120.139:8443*", y veremos la pantalla con el candado.



SERVIDOR NGNIX CON HTML

1. Empezaremos instalando Nginx, “*sudo apt install nginx -y*”.

```
ubuntu@ip-172-31-66-175:~$ sudo apt install nginx -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  nginx-common
```

2. Procederemos a configurar Nginx en el puerto 8081, con el comando “*sudo nano /etc/nginx/sites-available/default*”. En la configuración cambiaremos ***listen 80*** por ***listen 8081***.

```
ubuntu@ip-172-31-66-175:~$ sudo nano /etc/nginx/sites-available/default
```

```
server {
        listen 8081 default_server;
        listen [::]:8081 default_server;
```

3. Crearemos una página HTML personalizada, usando “*echo "<h1>Servidor Nginx</h1> Funcionando en puerto 8081 " | sudo tee /var/www/html/index.html*”.

```
ubuntu@ip-172-31-66-175:~$ echo "<h1>Servidor Nginx</h1><p> Funcionando en puerto 8081</p>" | sudo tee
 /usr/share/nginx/html/index.html
<h1>Servidor Nginx</h1><p> Funcionando en puerto 8081</p>
```

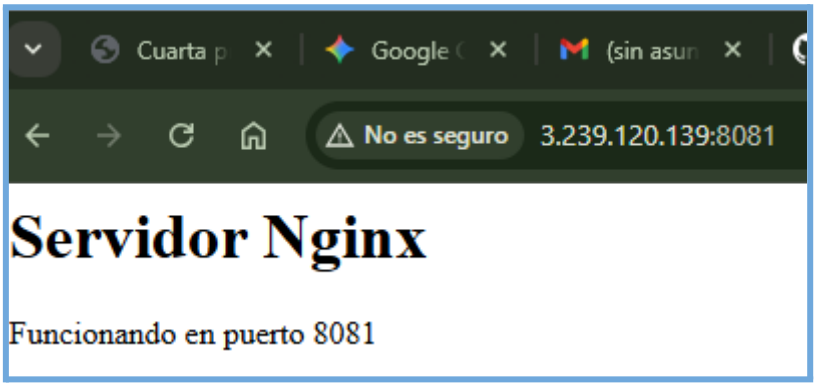

4. Reiniciamos Nginx, así aplicaremos todos los cambios de configuración, usaremos “*sudo systemctl restart nginx*”.

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl restart nginx
```

5. Verificamos su estado y miraremos si funciona en el puerto 8081, “*sudo systemctl status nginx*”.

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-12-04 13:36:41 UTC; 15s ago
       Docs: man:nginx(8)
```

6. Accederemos a *"http://3.239.120.139:8081"* y verificaremos que Nginx muestra bien el HTML. También podemos acceder desde la terminal con "*curl http://3.239.120.139:8081*".




```
ubuntu@ip-172-31-66-175:/var/www/html$ curl localhost:8081
<h1>Servidor Nginx</h1><p> Funcionando en puerto 8081</p>
```

SERVIDOR CADDY CON ARCHIVOS ESPECIALES

1. Comenzaremos instalando las dependencias necesarias, con "*sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl*" posteriormente añadiremos los repositorios externos.

```
ubuntu@ip-172-31-66-175:~$ sudo apt install -y debian-keyring debian-archive-keyring apt-transport-htt
ps curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
The following NEW packages will be installed:
  apt-transport-https debian-archive-keyring debian-keyring
0 upgraded, 3 newly installed, 0 to remove and 5 not upgraded.
```

2. Agregaremos el repositorio oficial de Caddy mediante el comando *"curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg"* este añade la clave de seguridad y *"curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list*" este añade la fuente del software.

```
ubuntu@ip-172-31-66-175:~$ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gp
g --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
```

```
ubuntu@ip-172-31-66-175:~$ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' |
sudo tee /etc/apt/sources.list.d/caddy-stable.list
# Source: Caddy
# Site: https://github.com/caddyserver/caddy
# Repository: Caddy / stable
# Description: Fast, multi-platform web server with automatic HTTPS
```

3. Actualizaremos la lista de paquetes e instalaremos Caddy, "*sudo apt update && sudo apt install caddy -y*".

```
ubuntu@ip-172-31-66-175:~$ sudo apt update && sudo apt install caddy -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://dl.cloudsmith.io/public/caddy/stable/deb/debian any-version InRelease [14.8 kB]
```

4. Ahora crearemos un directorio específico para los archivos de Caddy, "*sudo mkdir -p /var/www/caddy*".

```
ubuntu@ip-172-31-66-175:~$ sudo mkdir -p /var/www/caddy
```

5. Continuamos creando un archivo de Markdown de prueba.

- "*echo "# Bienvenido a Caddy" | sudo tee /var/www/caddy/README.md*".

```
ubuntu@ip-172-31-66-175:~$ echo "# Bienvenido a Caddy" | sudo tee /var/www/caddy/README.md
# Bienvenido a Caddy
```

- "*echo "" | sudo tee -a /var/www/caddy/README.md*".

```
ubuntu@ip-172-31-66-175:~$ echo "" | sudo tee -a /var/www/caddy/README.md
```

- "*echo "Este servidor está funcionando correctamente." | sudo tee -a /var/www/caddy/README.md*".

```
ubuntu@ip-172-31-66-175:~$ echo "Este servidor está funcionando correctamente." | sudo tee -a /var/www
/caddy/README.md
Este servidor está funcionando correctamente.
```

- "*echo "" | sudo tee -a /var/www/caddy/README.md*".

```
ubuntu@ip-172-31-66-175:~$ echo "" | sudo tee -a /var/www/caddy/README.md
```

- *"echo "## Características" | sudo tee -a /var/www/caddy/README.md*".

```
ubuntu@ip-172-31-66-175:~$ echo "## Características" | sudo tee -a /var/www/caddy/README.md
## Características
```

6. Descargamos una imagen de prueba para verificar que Caddy sirve archivos estáticos, usaremos "*curl -o /tmp/test-image.jpg "https://www.python.org/static/apple-touch-icon-144x144-precomposed.png"* y "*sudo mv /tmp/test-image.jpg /var/www/caddy/test.jpg*".

```
ubuntu@ip-172-31-66-175:~$ curl -o /tmp/test-image.jpg "https://www.python.org/static/apple-touch-icon
-144x144-precomposed.png"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7382  100  7382    0     0   178k      0 --:--:-- --:--:-- --:--:--  180k
```

```
ubuntu@ip-172-31-66-175:~$ sudo mv /tmp/test-image.jpg /var/www/caddy/test.jpg
```

7. Procederemos crear Caddyfile personalizado, con el comando "*sudo nano /etc/caddy/Caddyfile"* abriremos el archivo de configuración y escribiremos lo siguiente *":8082 { root * /var/www/caddy file_server browse @markdown path *.md header @markdown Content-Type text/plain* ".

```
ubuntu@ip-172-31-66-175:~$ sudo nano /etc/caddy/Caddyfile
```

```
 GNU nano 7.2                                  /etc/caddy/Caddyfile *
:8082 {
    root * /var/www/caddy
    file_server browse
    @markdown path *.md
    header @markdown Content-Type text/plain
}
```

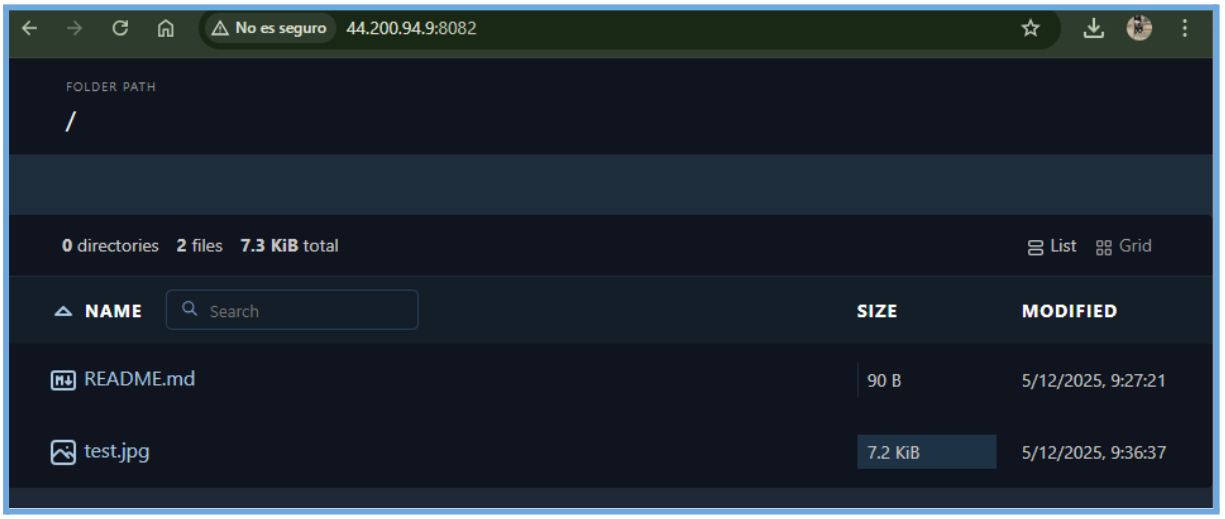
8. Reiniciamos Caddy para aplicar la configuración "*sudo systemctl restart caddy*".

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl restart caddy
```

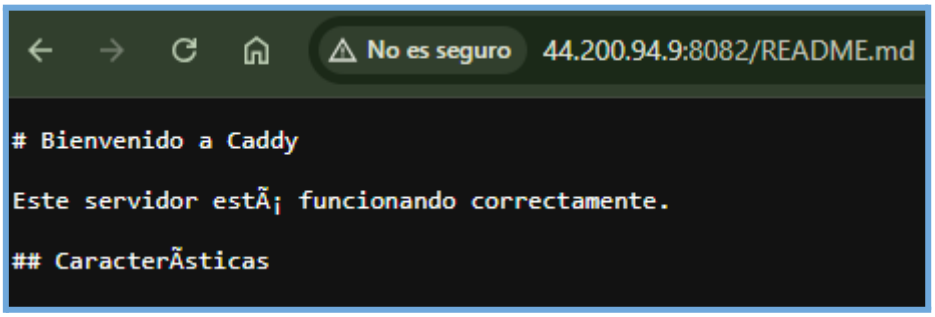
9. Revisamos su estado "*sudo systemctl status caddy*".

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl status caddy
● caddy.service - Caddy
     Loaded: loaded (/usr/lib/systemd/system/caddy.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-12-05 08:40:54 UTC; 18s ago
       Docs: https://caddyserver.com/docs/
```

10. Ahora empezaremos con las verificaciones, empezamos capturando el listado de archivos *"http://44.200.94.9:8082*".



11. Continuaremos capturando el contenido, "*http://44.200.94.9:8082/README.md*".



12. Finalizamos capturando la imagen, "*http://44.200.94.9:8082/test.jpg*".

VERIFICACIÓN SIMULTÁNEA DE LOS TRES SERVIDORES

1. Ejecutaremos un comando para que muestre el estado de los tres servicios simultáneamente, con "*sudo systemctl status apache2 nginx caddy*".

```
ubuntu@ip-172-31-66-175:~$ sudo systemctl status apache2 nginx caddy
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-12-05 07:53:19 UTC; 1h 4min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 559 (apache2)
```

```
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-12-05 07:53:19 UTC; 1h 4min ago
       Docs: man:nginx(8)
```

```
● caddy.service - Caddy
     Loaded: loaded (/usr/lib/systemd/system/caddy.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-12-05 08:40:54 UTC; 17min ago
       Docs: https://caddyserver.com/docs/
   Main PID: 2642 (caddy)
```

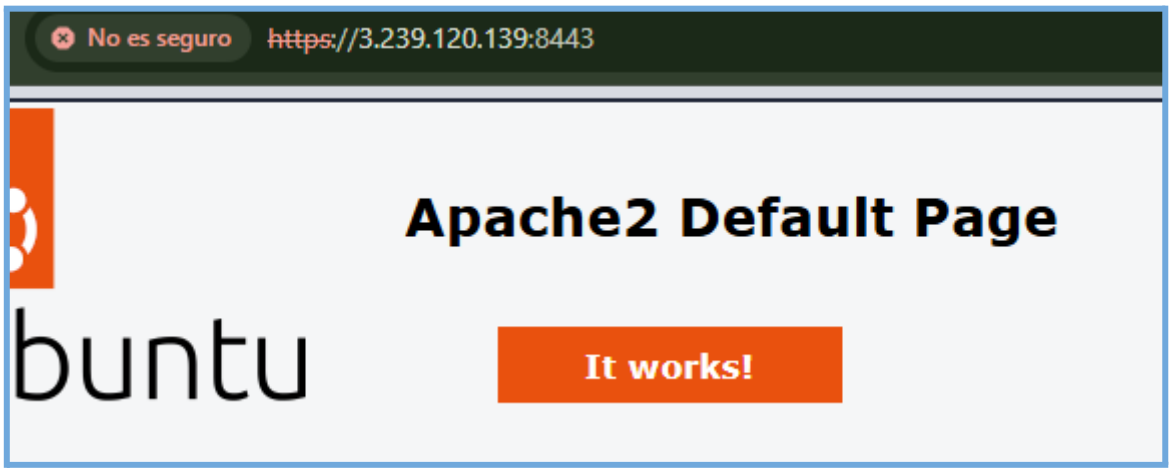
2. Seguiremos con el comando netplan, para mostrar los cuatro puertos en uso, usaremos "*sudo netstat -tulpn | grep -E '8080|8081|8082|8443'*".

```
ubuntu@ip-172-31-66-175:~$ sudo netstat -tulpn | grep -E '8080|8081|8082|8443'
tcp        0      0 0.0.0.0:8081            0.0.0.0:*               LISTEN      623/nginx: master p
tcp6       0      0 :::8443                 :::*                    LISTEN      559/apache2
tcp6       0      0 :::8081                 :::*                    LISTEN      623/nginx: master p
tcp6       0      0 :::8080                 :::*                    LISTEN      559/apache2
tcp6       0      0 :::8082                 :::*                    LISTEN      2642/caddy
```

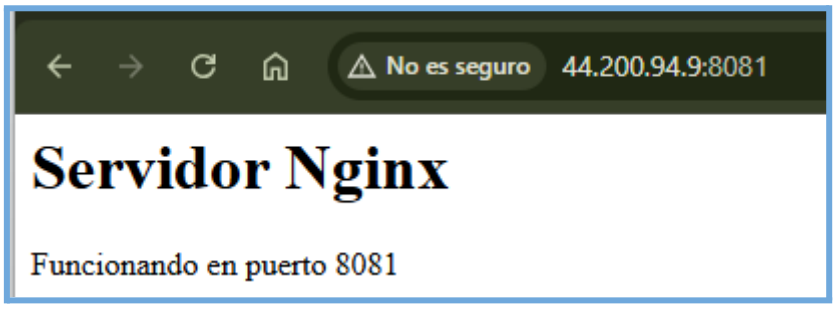
3. Abriremos las cuatro pestañas del navegador simultáneamente, Apache HTTP, Apache HTTPS, Nginx y Caddy.

- ***Apache HTTP***: http://44.200.94.9:8080/info.php

- ***Apache HTTPS:*** https://44.200.94.9:8443



- ***Nginx***: http://44.200.94.9:8081



- ***Caddy***: http://44.200.94.9:8082