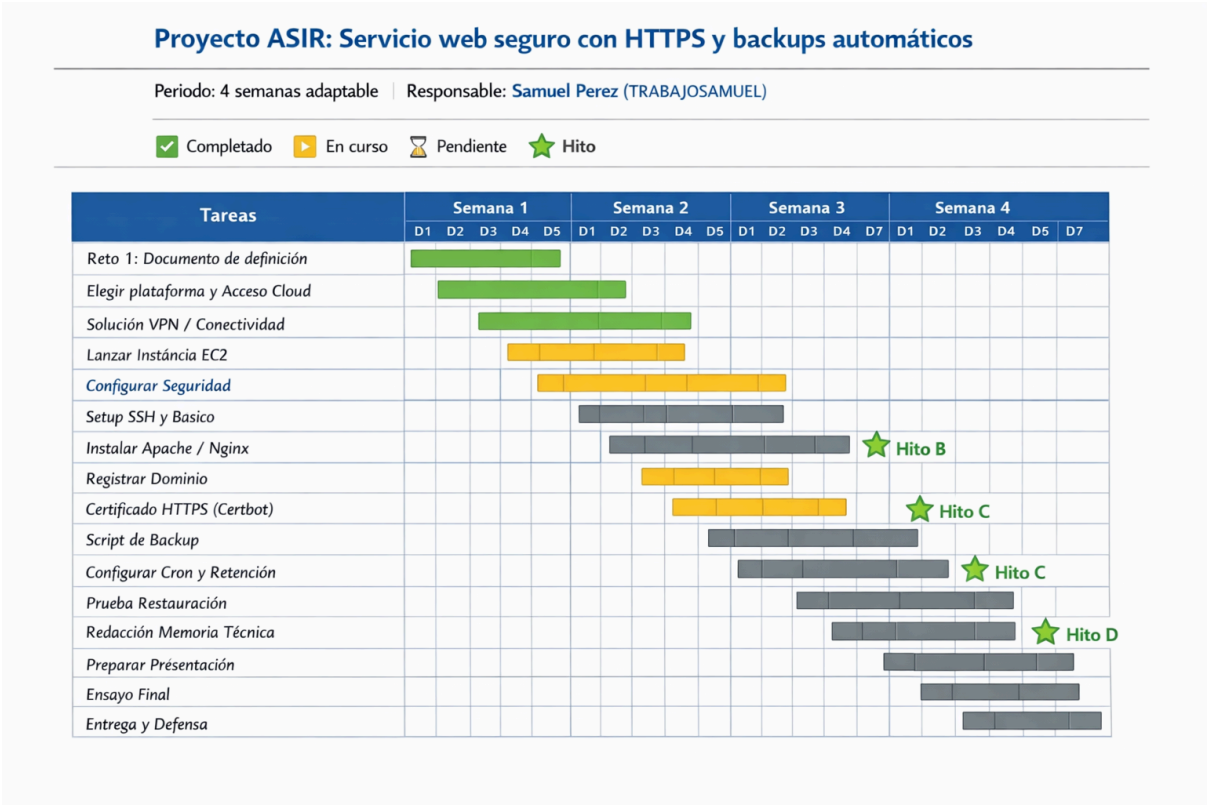


📝 Proyecto ASIR – Servicio web seguro con HTTPS y backups automáticos

1️⃣ Descripción del proyecto

El proyecto consiste en **instalar y configurar un servidor web seguro**, usando Apache o NGINX, con **certificados SSL automáticos** (Let's Encrypt) y un sistema de **copias de seguridad automatizadas** para el contenido y la configuración. El objetivo es garantizar **disponibilidad, integridad y seguridad** de los datos de una web corporativa, siguiendo buenas prácticas de administración de sistemas y servicios.

Se trata de un proyecto **práctico y realista** que puede implementarse en un laboratorio o en un servidor virtual/entorno de pruebas.

2️⃣ Empresa o entorno profesional

- Empresa simulada: **"Never Forget SL"**, pequeña empresa de servicios digitales.

- Sector: Tecnología e informática.
- Contexto: La empresa mantiene un portal web corporativo con información de clientes y servicios. Actualmente carece de un sistema de **seguridad HTTPS** y **copias de seguridad automáticas**, lo que puede causar pérdida de datos y problemas de confianza con los clientes.

3️⃣ Problema o necesidad detectada

- La empresa necesita que su **web esté siempre disponible y segura**, evitando vulnerabilidades derivadas de no usar HTTPS.
- Falta un sistema de **respaldo automático**, que permita restaurar la web rápidamente en caso de fallo, error humano o ataque.
- Necesidad de aplicar **buenas prácticas de administración de sistemas**, que además pueda ser documentada como evidencia de aprendizaje.

4️⃣ Objetivos

Objetivo general

- Configurar un servidor web seguro con HTTPS y automatizar las copias de seguridad, aplicando buenas prácticas de ASIR y asegurando la disponibilidad y seguridad de la información.

Objetivos específicos

1. Instalar y configurar Apache o NGINX en un servidor Linux.
2. Implementar certificados SSL automáticos con Let’s Encrypt.
3. Configurar copias de seguridad automáticas de los archivos de la web y la configuración del servidor.
4. Documentar el proceso y las configuraciones realizadas.
5. Realizar pruebas de restauración y verificación del sistema de backup.

5️⃣ Alcance del proyecto

Incluye:

- Configuración de servidor web (Apache o NGINX).
- Implementación de HTTPS mediante certificados automáticos.
- Sistema de backups automáticos programados (cron) y restauración de prueba.
- Documentación técnica detallada.

No incluye:

- Desarrollo de contenido web complejo.
- Usuarios reales ni gestión de base de datos con alta carga.
- Implementación de sistemas de alta disponibilidad distribuida.

6️⃣ Relación con módulos ASIR

- **Implantación de aplicaciones web (IMW):** Configuración y puesta en marcha del servidor web.
- **Seguridad y alta disponibilidad (SGY):** HTTPS, protección de certificados, seguridad en la configuración y respaldo.
- **Administración de sistemas operativos (ADD):** Instalación, configuración de Linux y programación de scripts de backup.
- **Administración de sistemas gestores de bases de datos (ADE):** Opcional si la web tiene base de datos; el backup puede incluir bases de datos MySQL/PostgreSQL.

7️⃣ Relación con sostenibilidad y ODS

- **ODS 9 – Industria, innovación e infraestructura:** Mejora de la infraestructura digital de la empresa.
- **ODS 12 – Producción y consumo responsables:** Optimización de recursos mediante backups automáticos, reduciendo pérdidas de información y evitando duplicación innecesaria de datos.
- **ODS 4 – Educación de calidad:** Proyecto permite aprendizaje práctico y documentado para futuros profesionales.

Informe de sostenibilidad:

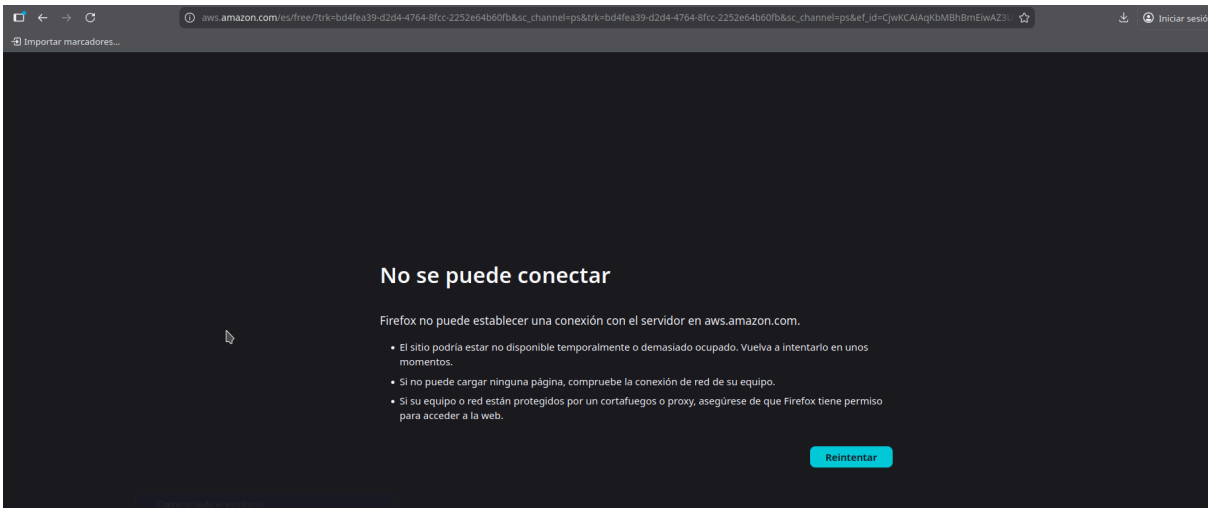
- Uso de **recursos mínimos** (entorno virtual o laboratorio local).

- La automatización reduce errores humanos y consumo de tiempo.
- Seguridad web evita incidentes que puedan generar **costes de recuperación y pérdida de datos**, contribuyendo a la eficiencia y sostenibilidad digital.

8️⃣ Necesidades técnicas

- Servidor Linux (real o virtual).
- Apache
- Let’s Encrypt o Certbot para certificados SSL.
- Cron jobs para backups automáticos.
- Espacio de almacenamiento para respaldos (local o en la nube).
- Opcional: base de datos MySQL/PostgreSQL si la web tiene contenido dinámico.

Problemas:
Hosting:Oracle tier FREE no acepta mi tarjeta virtual o AWS no funciona por algún bloqueo en la red del zonzamas



Siguiente propuesta:azure microsoft

Solución:creé una cuenta e instalé proton VPN (sencillo,copiar y pegar) en mi máquina virtual para poder acceder a AWS.
Luego creé una cuenta free en AWS para poder alojar mi servidor

COMIENZO:

Solución al problema de no poder entrar al AWS.

Elegí ProtonVPN (Noruega)

Noruega tiene muy buenas leyes de privacidad y no pertenece a alianzas de vigilancia como los “14 Eyes”, lo cual puede añadir una capa extra de seguridad legal (no afecta directamente la velocidad, pero sí la privacidad).

Luego empezaremos con las instancias

EC2 (Elastic Compute Cloud) es el servicio de AWS que proporciona una máquina virtual en la nube.

Para el proyecto de ASIR que se está llevando a cabo, se necesitan ciertos requisitos específicos:

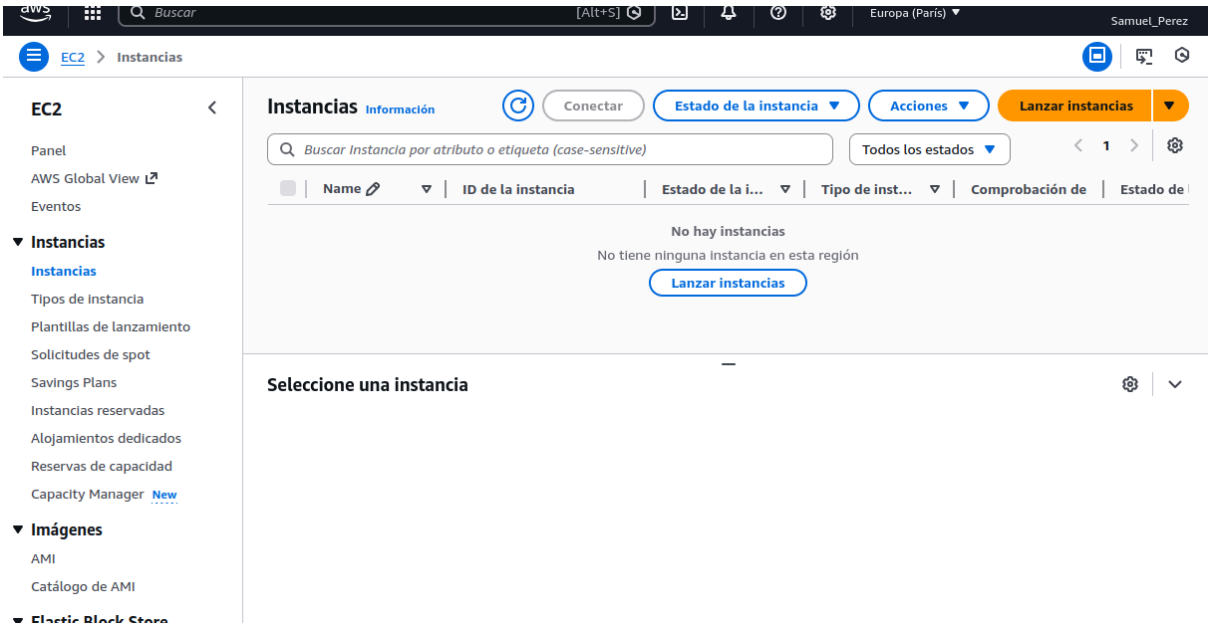
✅ Un sistema operativo Linux
✅ Acceso SSH
✅ Instalación de Apache2
✅ Instalación de Certbot (Let’s Encrypt)
✅ Programación de cron para backups
✅ Posibilidad de configurar el sistema

Un hosting compartido tradicional no permite realizar estas configuraciones, ya que limita el acceso a la máquina y a sus configuraciones.

EC2 es la opción adecuada, porque ofrece un servidor Linux totalmente administrable, permitiendo la instalación y personalización de todos los servicios necesarios.

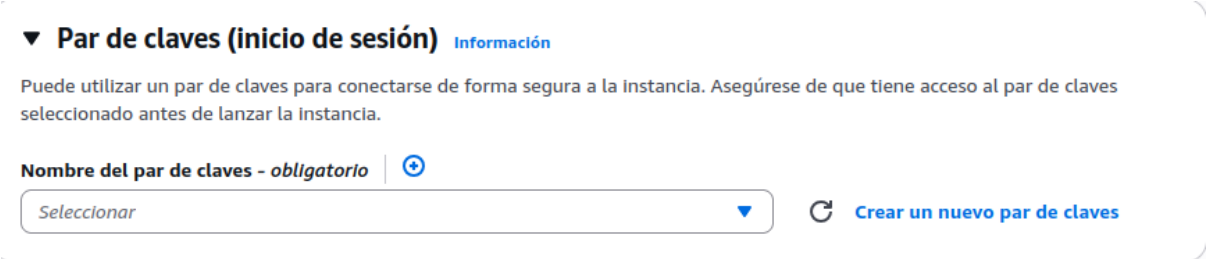
Para un proyecto ASIR donde se documenta la instalación, configuración y seguridad del servidor, EC2 es la opción más apropiada.

Arriba elegimos “Lanzar instancias”



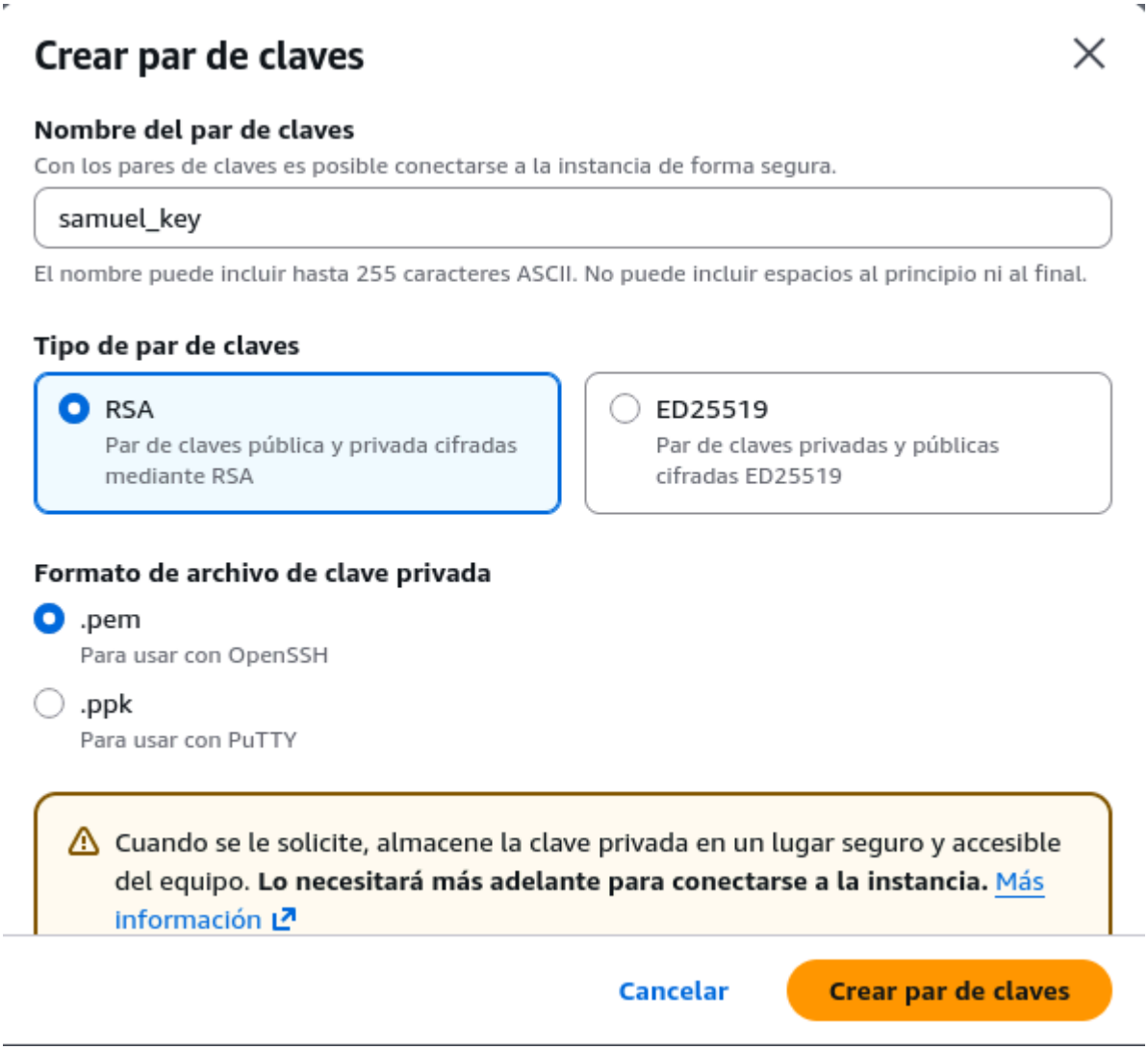
Elegimos instancias

Escogí esta instancia porque ya estaba familiarizado con esta distro de Linux y la versión más estable



Esto será relevante para más adelante.

Creamos las claves públicas y privadas cifradas



Tras haber consultado a un experto en ciberseguridad y pentesting,elegimos el par de claves RSA

1️⃣ Información principal de la instancia

- **ID de la instancia:** i-05f9a4179dde4f177
- **Nombre de la instancia:** “No Me olvides”
- **IP pública:** 35.180.135.19
- **Nombre de usuario:** ubuntu (por ser la AMI de Ubuntu 24.04)
- **Archivo de clave privada:** samuel_key.pem

2️⃣ Preparar la clave para SSH

Antes de conectarnos, debemos proteger la clave privada para que solo nosotros podamos leerla:

chmod 400 samuel_key.pem

Esto evita que otros usuarios en nuestro sistema puedan usarla.

3️⃣ Conexión por SSH

Desde nuestra terminal (Linux / Mac) o Git Bash en Windows:

ssh -i "samuel_key.pem" ubuntu@ec2-35-180-135-19.eu-west-3.compute.amazonaws.com

- -i "samuel_key.pem" → indica nuestra clave privada.
- ubuntu@... → usuario y DNS público de nuestra instancia.

⚠️ Si estamos en Windows con PuTTY, necesitamos convertir samuel_key.pem a samuel_key.ppk usando **PuTTYgen**.

4️⃣ Alternativa: Cliente basado en navegador

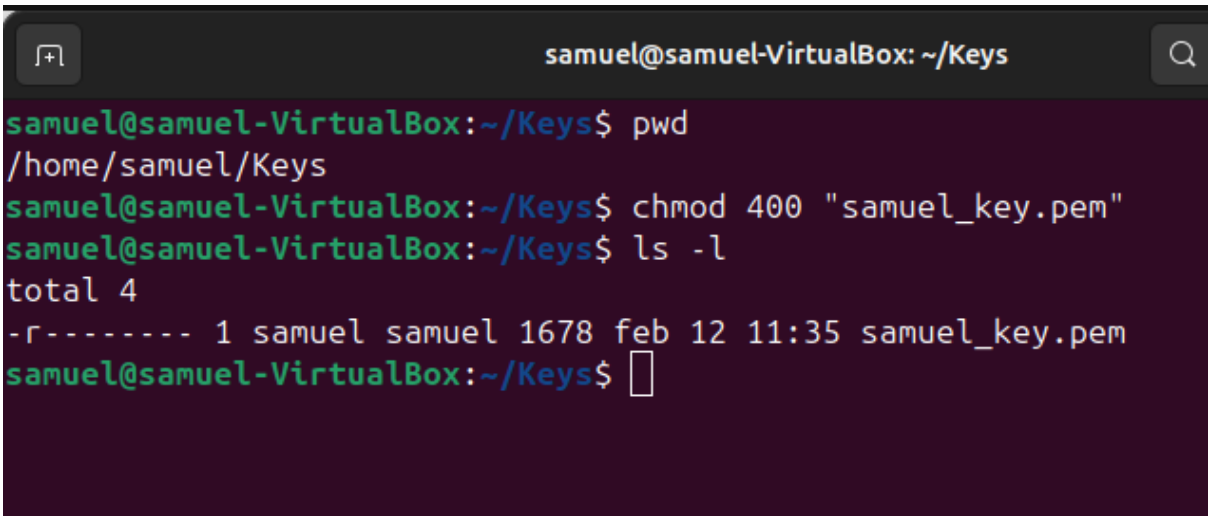
AWS también nos permite conectarnos **sin instalar nada** usando el **EC2 Instance Connect** en el navegador:

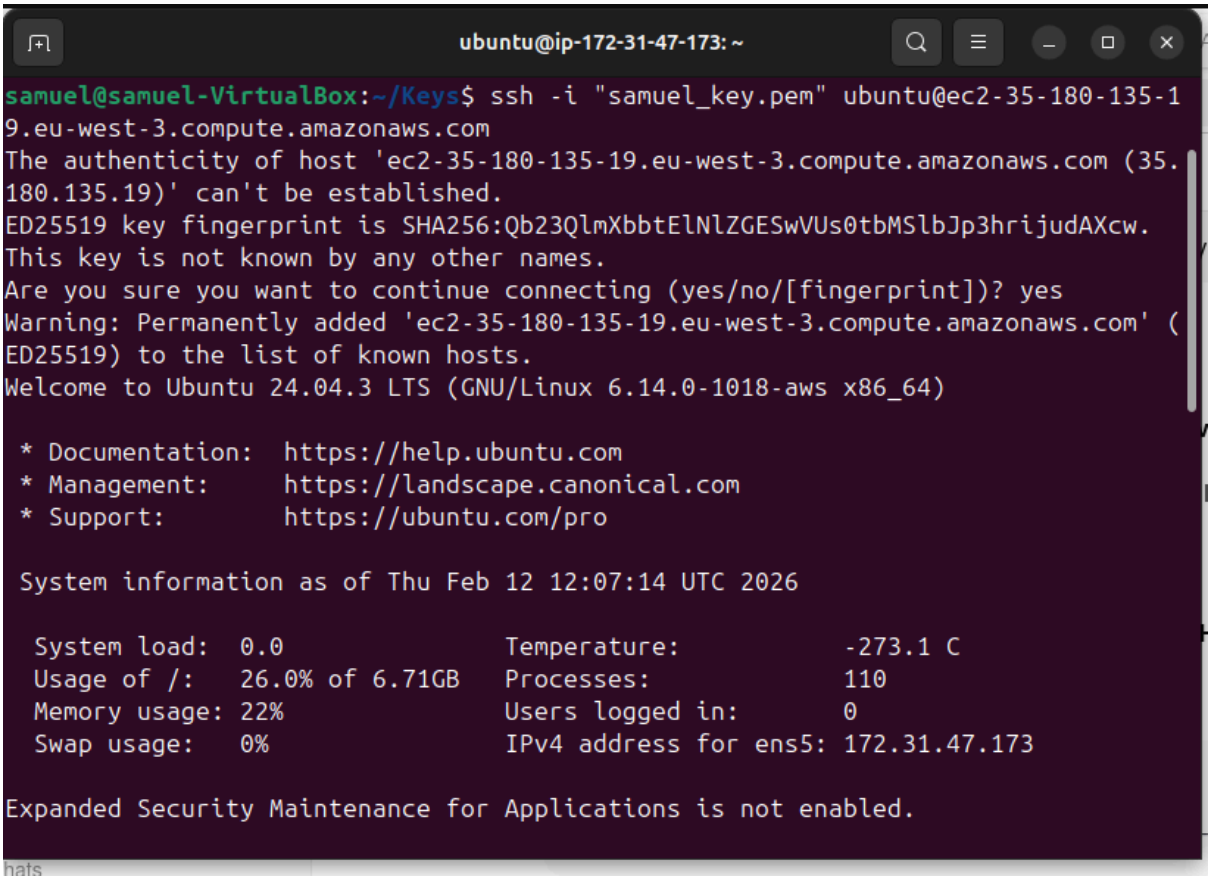
- Abrimos la instancia en la consola AWS → botón **Conectar → EC2 Instance Connect**.
- No necesitamos la clave .pem para esto.

5️⃣ Notas importantes

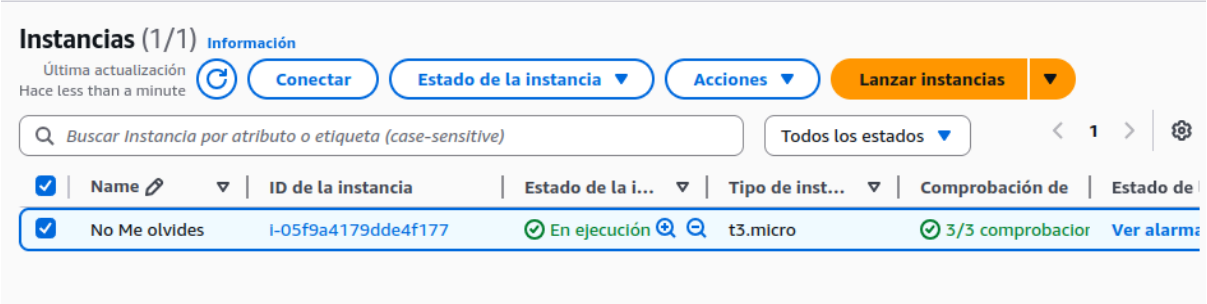
- El **agente SSM no está en línea**, por lo que no podemos usar la consola de sesión de AWS para conectarnos de forma “just-in-time”.

- Para la práctica, la conexión **SSH tradicional** con nuestra .pem funciona perfectamente.
- Una vez conectados, podemos instalar Apache, crear scripts de backup y probar todo.

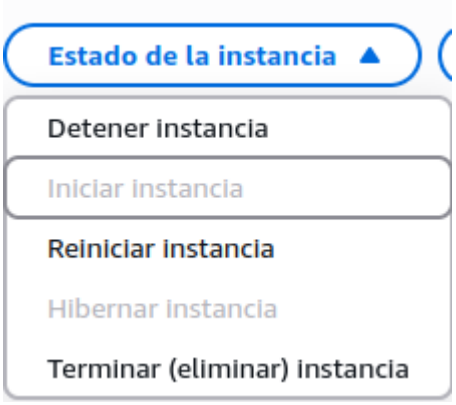




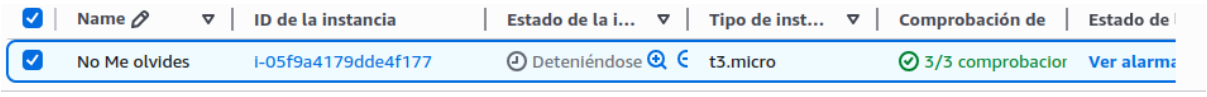
Aquí vemos que la instancia está en activo



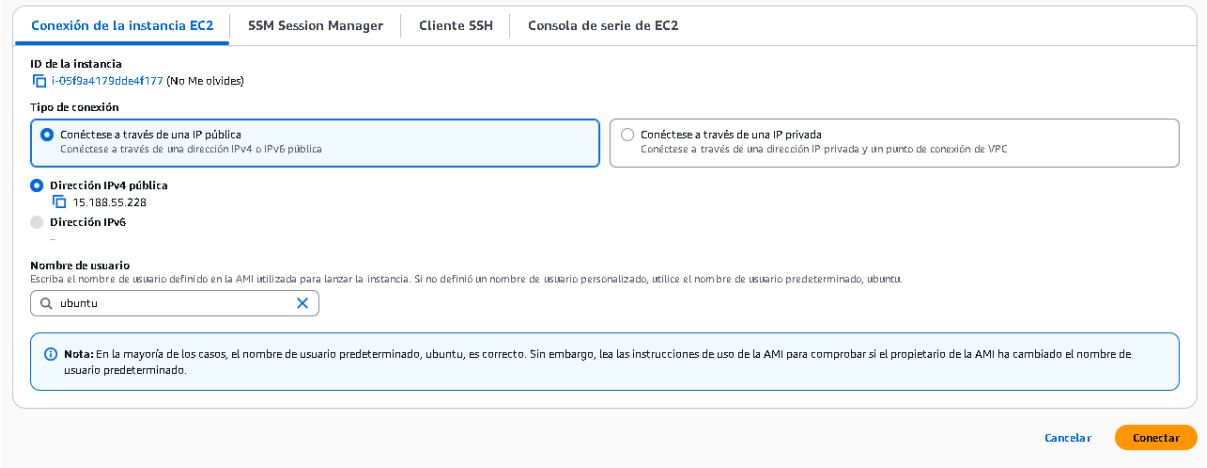
RECOMENDABLE APAGARLO SI NO ESTÁ EN USO:



Aquí se ve que está deteniéndose



Otra forma de conectarse (problema:sin tener una key en mi equipo no puedo acceder por SSH normalmente)por IP



Funciona como una consola común

```
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sat Feb 14 09:48:56 UTC 2026

  System load:  0.0               Temperature:           -273.1 C
  Usage of /:   28.8% of 6.71GB   Processes:             113
  Memory usage: 24%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.47.173


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Feb 12 12:07:14 2026 from 205.147.17.29
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Comandos usados :

sudo apt update && sudo apt upgrade -y

Como quiero hacer una página de backups,instalaré el servicio Apache2

Para el desarrollo de la plataforma web de backups *No Me Olvides*, se ha decidido utilizar **Apache HTTP Server (Apache2)** como servidor web principal.

1 Estabilidad y madurez

Apache es uno de los servidores web más utilizados a nivel mundial y cuenta con:

- Más de 25 años de desarrollo
- Amplia documentación oficial
- Gran comunidad de soporte

Esto garantiza estabilidad, seguridad y compatibilidad en entornos productivos.

2 Compatibilidad con entornos Linux

La infraestructura del proyecto está desplegada sobre:

- **Ubuntu 24.04 LTS**
- Infraestructura en **Amazon Web Services**

Apache está perfectamente optimizado para sistemas Linux y se integra de forma nativa mediante el servicio apache2, facilitando su administración y automatización.

3 Bajo consumo de recursos (importante por limitación de 6 GB)

Dado que el proyecto se desarrolla en una instancia EC2 con recursos limitados (t3.micro), Apache:

- Tiene un consumo de memoria reducido en configuraciones básicas
- Permite desactivar módulos innecesarios
- Funciona correctamente en entornos con pocos recursos

Esto lo hace adecuado para un entorno académico y de pruebas.

4 Compatibilidad con PHP

El proyecto necesita:

- Formularios de subida de archivos
- Comunicación con almacenamiento externo (Amazon S3)
- Procesamiento backend simple

Apache permite integración directa con PHP mediante mod_php, lo que facilita:

- Desarrollo rápido
- Configuración sencilla
- Menor complejidad que otras arquitecturas

5 Seguridad y control

Apache permite implementar fácilmente:

- Configuración de firewall (puerto 80/443)
- Certificados SSL (Let's Encrypt)
- Control de acceso mediante .htaccess
- Restricción de directorios

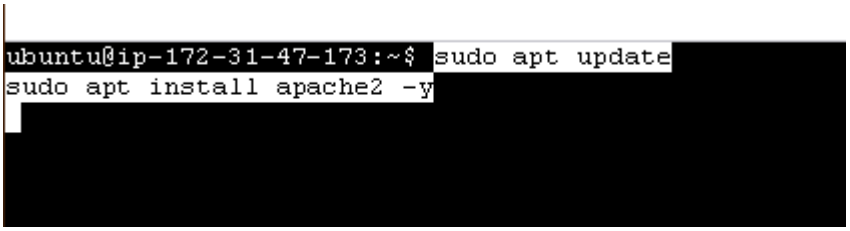
Esto es fundamental en un proyecto de copias de seguridad donde la protección de datos es clave.

6 Enfoque académico y profesional

En el ámbito de ASIR y administración de sistemas:

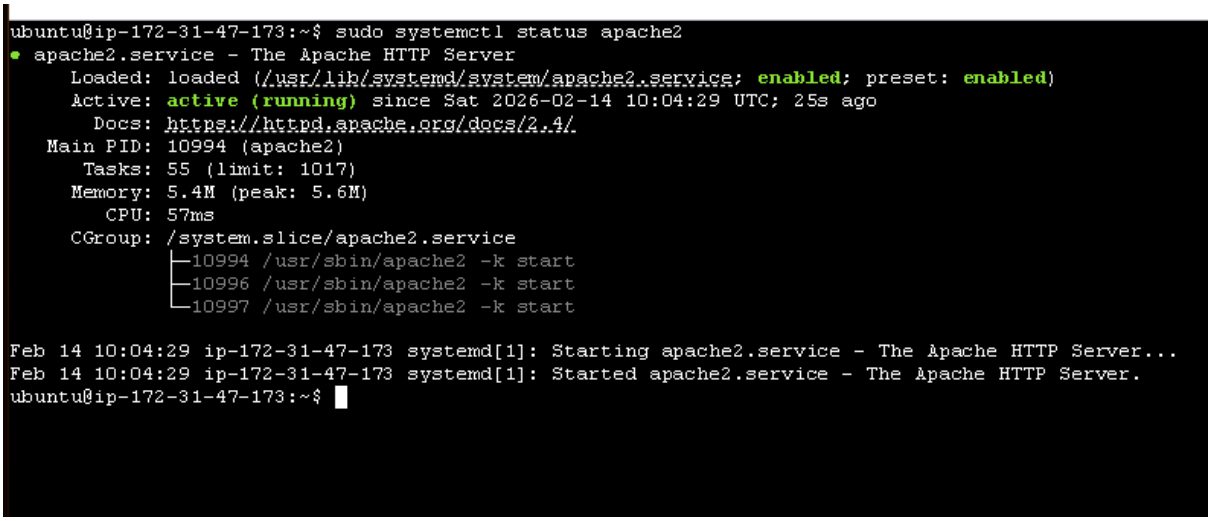
- Apache es ampliamente utilizado en prácticas formativas
- Permite demostrar conocimientos de:
    - Servicios web
    - Configuración de puertos
    - Logs
    - Virtual Hosts
    - Seguridad

Por tanto, su elección no solo es técnica, sino también formativa.



Comprobamos que está instalada correctamente:

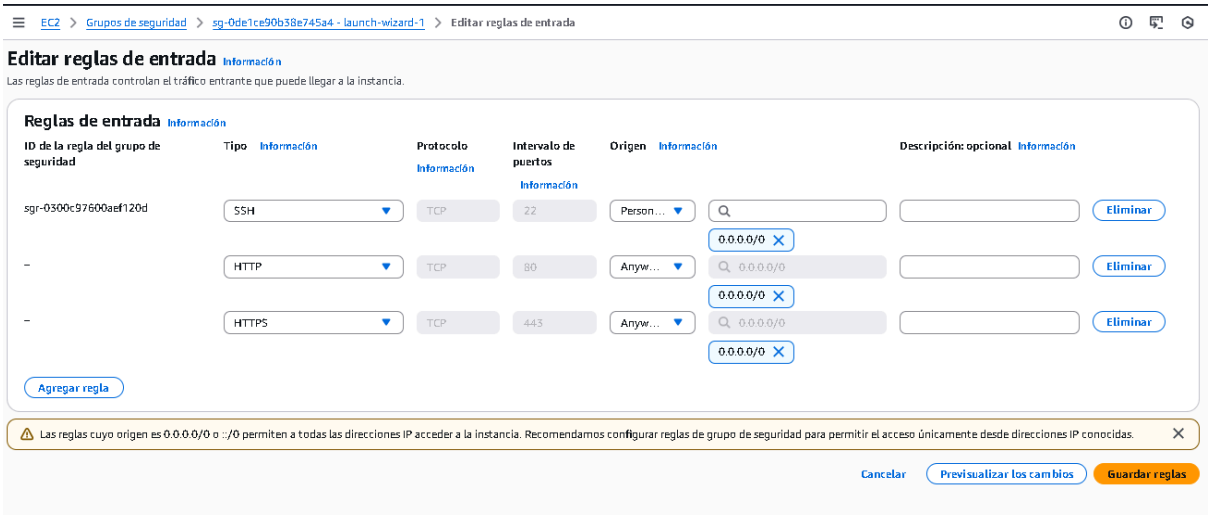
sudo systemctl status apache2



No hemos modificado ningún archivo clave ni ninguna configuración,por lo que no podemos acceder todavía por HTTP
http://15.188.55.228

Paso 1 — Verificar reglas del Security Group

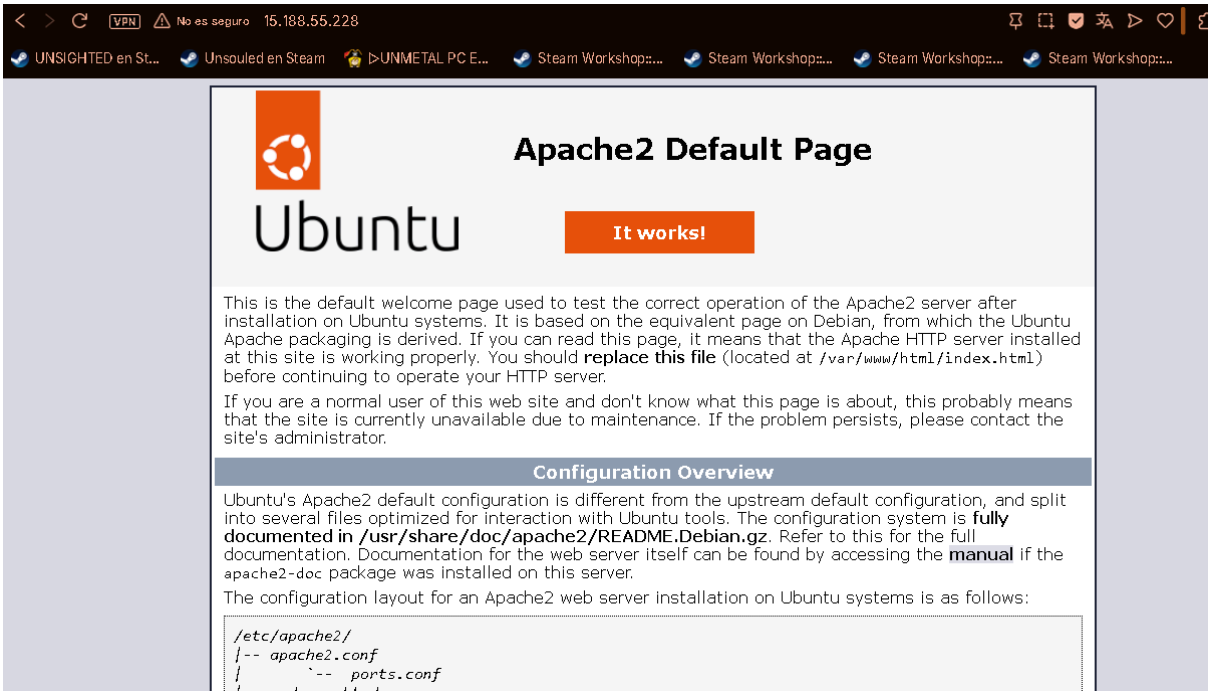
1. Ve a **AWS → EC2 → Instancias**
2. Selecciona tu instancia (No Me olvides)
3. En la pestaña **Seguridad → Grupos de seguridad** → haz clic en el grupo asociado
4. Verifica **Reglas de entrada** (Inbound Rules)



Le damos a guardar:

Probar de nuevo

http://15.188.55.228



Habilitar el firewall IMPORTANTE:

```
ubuntu@ip-172-31-47-173:~$ sudo ufw status
Status: inactive
```

sudo ufw enable

```
ubuntu@ip-172-31-47-173:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
ubuntu@ip-172-31-47-173:~$ sudo ufw status
Status: active
ubuntu@ip-172-31-47-173:~$
```

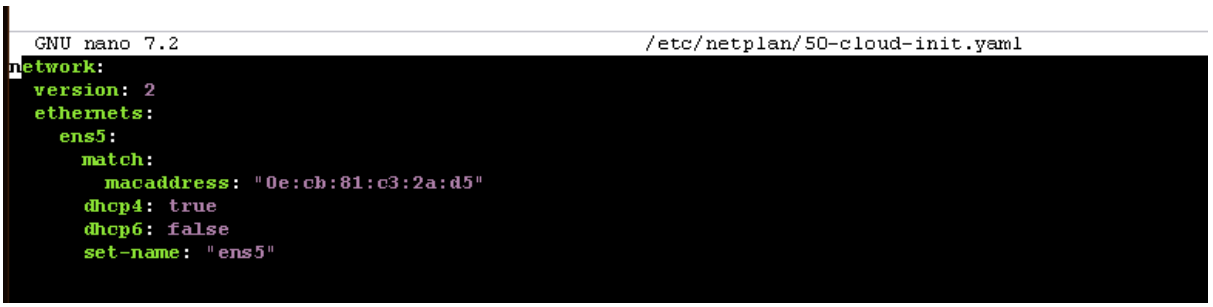
sudo ufw allow 22/tcp # SSH
sudo ufw allow 80/tcp # HTTP
sudo ufw allow 443/tcp # HTTPS

```
ubuntu@ip-172-31-47-173:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)

ubuntu@ip-172-31-47-173:~$
```

De momento solo habilitamos los puertos necesarios



Hemos decidido **no modificar la IP de la instancia** y mantener la asignación dinámica de AWS por varias razones:

1. **Seguridad y simplicidad:** No necesitamos tocar la configuración interna de red, lo que reduce riesgos de errores de conectividad.

2. **Práctica académica:** Para fines de la práctica, la IP dinámica es suficiente, ya que solo accedemos desde un navegador o SSH.

3. **Flexibilidad:** La instancia puede detenerse y arrancarse sin necesidad de reconfigurar Apache o la red (ideal porque consume dinero si la dejamos encendida).

4. **AWS recomienda DHCP:** La asignación dinámica de IP facilita la administración de instancias y es la configuración por defecto de Ubuntu en EC2.

Diseño de la web (implantación web)

Aquí haremos la web lo más ordenada y responsiva posible (algunos parámetros pueden variar más adelante)

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>No Me Olvides</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>

<header class="header">
   <div class="container header-content">
      <h1 class="logo">No Me Olvides</h1>
      <nav class="nav">
         <a href="/">Inicio</a>
         <a href="/upload.php">Subir Backup</a>
         <a href="#contacto">Contacto</a>
      </nav>
   </div>
</header>

<main class="main">
   <section class="hero">
      <h2>Bienvenido al Proyecto No Me Olvides</h2>
      <p>Servicio de copias de seguridad en la nube con AWS.</p>
      <a class="btn" href="/upload.php">Ir a Subir Archivos</a>
   </section>
</main>

<footer class="footer" id="contacto">
   <p>&copy; 2026 No Me Olvides – Never Forget SL. Todos los derechos reservados.</p>
```

```
</footer>

</body>
</html>
```

Estilo:

```
/* Reset básico */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Tipografía y base */
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
}

/* Contenedor de ancho máximo */
.container {
    width: 90%;
    max-width: 1200px;
    margin: auto;
}

/* Header */
.header {
    background-color: #003366;
    color: white;
    padding: 1rem;
}
```

```
.header-content {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.nav a {
  color: white;
  margin-left: 1rem;
  text-decoration: none;
  font-weight: bold;
}

.nav a:hover {
  text-decoration: underline;
}

/* Hero */
.hero {
  text-align: center;
  padding: 3rem 1rem;
  background: #e9f0fb;
  flex: 1;
}

.hero h2 {
  font-size: 2rem;
  margin-bottom: 0.5rem;
}

.hero p {
  margin-bottom: 1rem;
  font-size: 1.2rem;
  color: #555;
}
```

```
.btn {
    display: inline-block;
    padding: 0.75rem 1.5rem;
    background-color: #0055aa;
    color: white;
    text-decoration: none;
    border-radius: 5px;
    font-weight: bold;
}

.btn:hover {
    background-color: #003f7f;
}

/* Footer */
.footer {
    background-color: #003366;
    color: white;
    text-align: center;
    padding: 1rem;
    font-size: 0.9rem;
}

/* Responsive (menús apilan en pantallas pequeñas) */
@media (max-width: 768px) {
    .header-content {
        flex-direction: column;
    }
    .nav {
        margin-top: 0.5rem;
    }
    .nav a {
        display: block;
        margin: 0.3rem 0;
    }
}
```

Permisos por si acaso:

sudo chown www-data:www-data /var/www/html/styles.css
sudo chmod 644 /var/www/html/styles.css

Primer prototipo (la ip pública y el diseño final pueden variar http://13.38.250.51):

Una vez tengamos una página funcional,configuraremos los archivos de apache2

Primero el puerto 80:

sudo nano /etc/apache2/sites-available/nomeolvides.conf

```
<VirtualHost *:80>
    ServerName nomeolvides.com
    ServerAlias www.nomeolvides.com

    DocumentRoot /var/www/html

    <Directory /var/www/html>
        AllowOverride All
```

Require all granted
</Directory>

ErrorLog ${APACHE_LOG_DIR}/nomeolvides_error.log
CustomLog ${APACHE_LOG_DIR}/nomeolvides_access.log combined
</VirtualHost>

Puede cambiar en el futuro,esto es una plantilla.

Esto nos da logs de errores,nuestro dominio,etc.

Activamos el sitio:

sudo a2ensite nomeolvides.conf

Recargamos el servicio
sudo systemctl reload apache2

Comprobamos la sintaxis
sudo apache2ctl configtest

```
ubuntu@ip-172-31-47-173:~$ sudo a2ensite nomeolvides.conf
sudo systemctl reload apache2
sudo apache2ctl configtest
Enabling site nomeolvides.
To activate the new configuration, you need to run:
  systemctl reload apache2
Syntax OK
ubuntu@ip-172-31-47-173:~$
```

Siguiente problema:clave .pem.

La instalé en otro equipo por lo que no puedo hacer SCP normalmente,por lo que para descargar imágenes tendré que hacer un wget de manera rudimentaria

Ejemplo:

https://cdn-icons-png.flaticon.com/512/3585/3585618.png

cd /var/www/html

sudo wget https://cdn-icons-png.flaticon.com/512/3585/3585618.png -O favicon.png

sudo chown www-data:www-data favicon.png

Convertir PNG a favicon:

sudo apt update
sudo apt install imagemagick -y

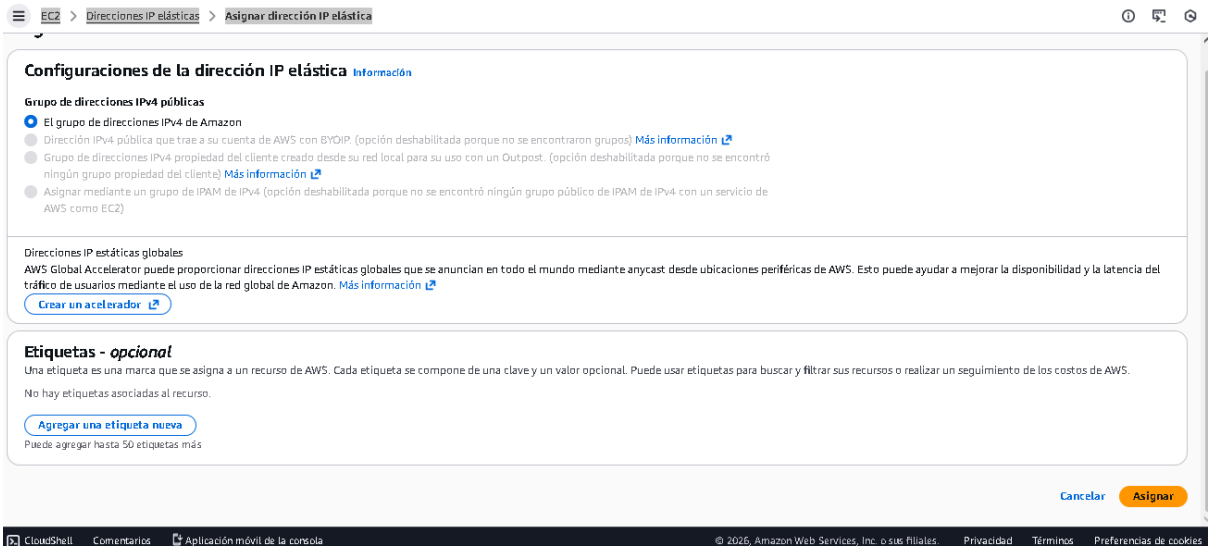
sudo convert favicon.png -resize 64x64 favicon.ico
sudo chown www-data:www-data favicon.ico

Hacemos referencia
<link rel="icon" href="/favicon.ico" type="image/x-icon">

Se ve pequeño pero se cambió

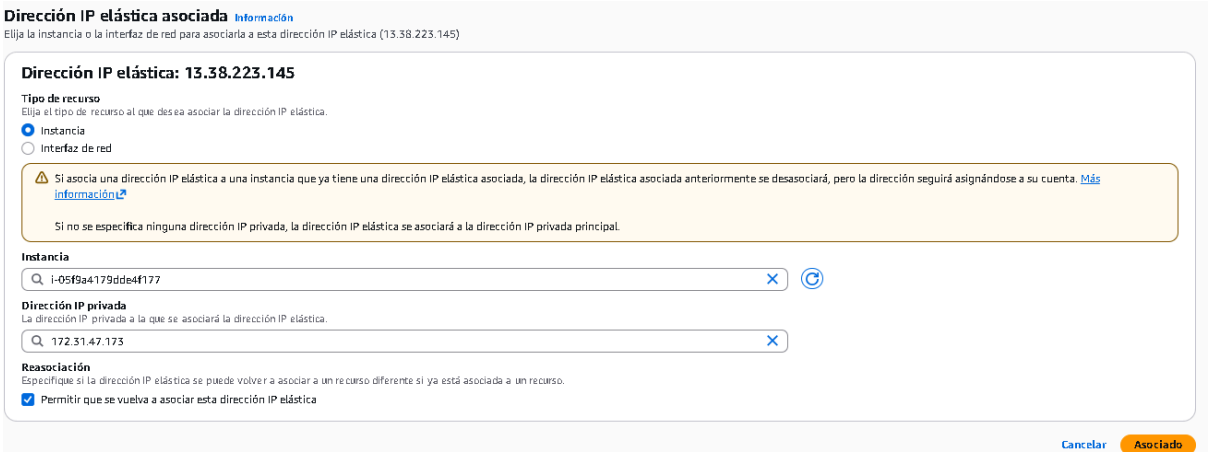
Siguiente problema:IP dinámica

Solución,crear una IP elástica en AWS



Una vez creada la asociamos:

Asignamos la instancia y la IP



172.31.47.173

DNS y Dominio:siguiente problema

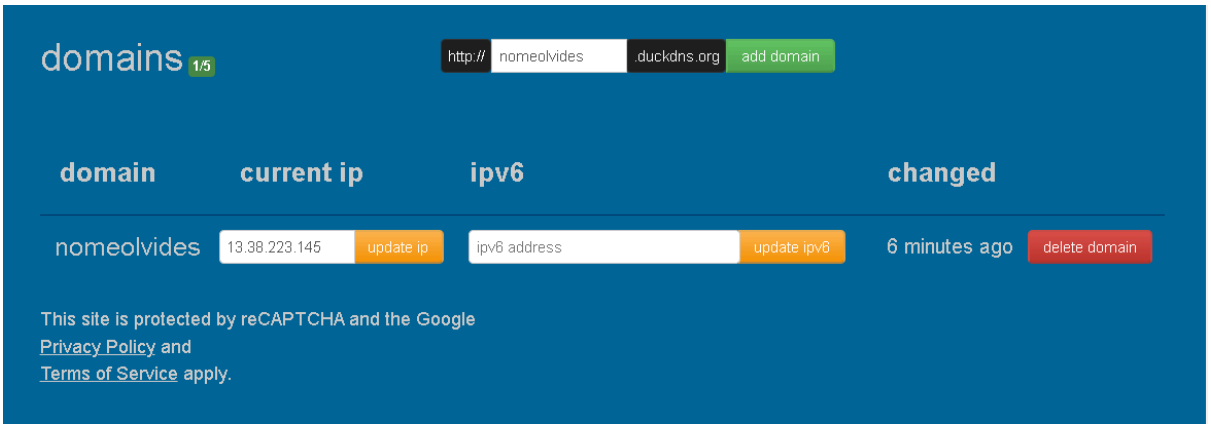
nuestro proyecto es nomeolvides,pero no hay capital para comprarlo,por tanto,de momento,usaremos duckdns.org

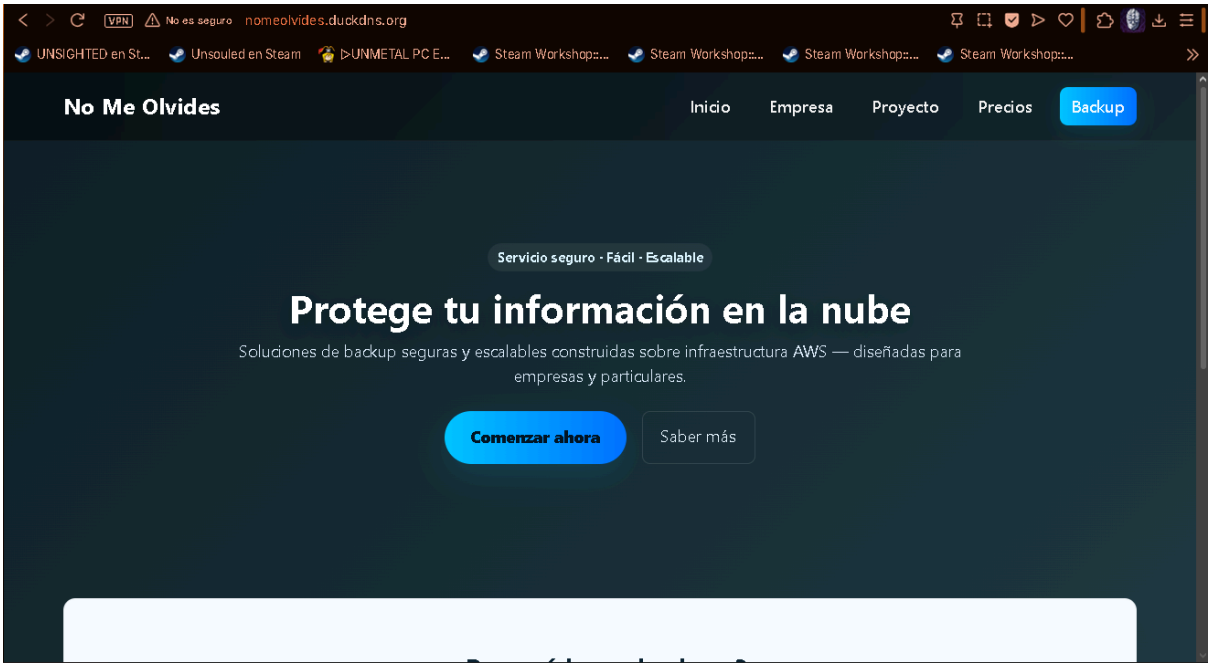
```
<VirtualHost *:80>
    ServerName nomeolvides.duckdns.org
    DocumentRoot /var/www/html

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/nomeolvides_error.log
    CustomLog ${APACHE_LOG_DIR}/nomeolvides_access.log combined
</VirtualHost>
```

Es gratuita y podemos registrarnos fácilmente por Google o Reddit
Duck DNS es un servicio de **DNS dinámico gratuito y sin ánimo de lucro**, mantenido por voluntarios y donaciones de usuarios. No cobra por los subdominios y se mantiene gracias a contribuciones voluntarias.





CONFIGURAR HTTPS

sudo apt update
sudo apt install ufw -y
sudo ufw allow OpenSSH
sudo ufw allow 'Apache Full'   # abre 80 y 443
sudo ufw enable
sudo ufw status

```
ubuntu@ip-172-31-47-173:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Apache Full                ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Apache Full (v6)           ALLOW       Anywhere (v6)
```

Comprobar DNS y accesibilidad

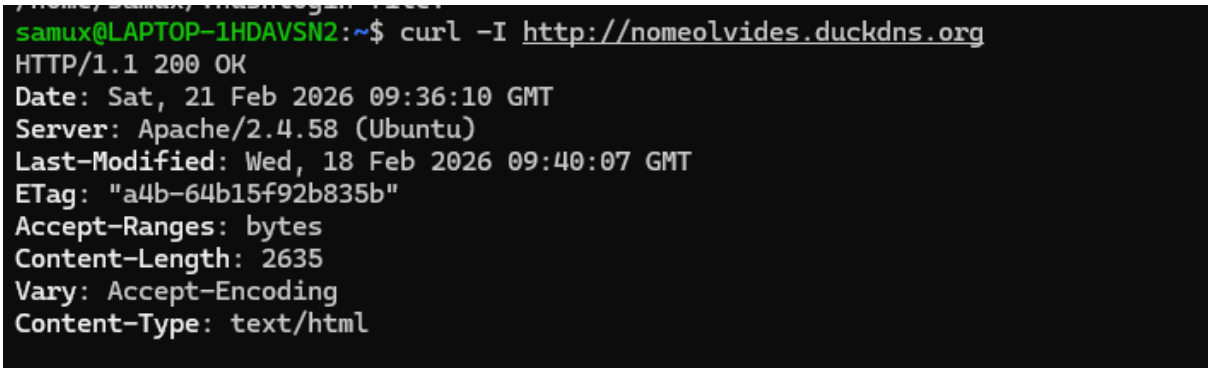
desde el servidor

curl -I http://localhost

```
ubuntu@ip-172-31-47-173:~$ curl -I http://localhost
HTTP/1.1 200 OK
Date: Sat, 21 Feb 2026 09:33:36 GMT
Server: Apache/2.4.58 (Ubuntu)
Last-Modified: Wed, 18 Feb 2026 09:40:07 GMT
ETag: "a4b-64b15f92b835b"
Accept-Ranges: bytes
Content-Length: 2635
Vary: Accept-Encoding
Content-Type: text/html
```

desde fuera (PC)

curl -I http://nomeolvides.duckdns.org



Ambos ponen 200,lo cual significa que funciona correctamente.

Instalación de Certbot (método recomendado: snap)

Ventajas de usar Certbot

1️⃣Certificado reconocido por navegadores

No aparecen advertencias de seguridad.

2️⃣Gratuito

Antes los certificados costaban dinero.
Let's Encrypt permite obtenerlos **gratis**.

3️⃣Automatización

Certbot instala el certificado y configura Apache automáticamente.

4️⃣Renovación automática

Los certificados duran **90 días**, pero Certbot los renueva solo.

Este método es el recomendado oficial para la mayoría de Ubuntu actuales.

sudo apt update
sudo apt install snapd -y
sudo snap install core
sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/local/bin/certbot
 # si te lo pide, permite plugins con:
sudo snap set certbot trust-plugin-with-root=ok

Se instaló correctamente:

```
ubuntu@ip-172-31-47-173:~$ certbot --version
certbot 5.3.1
ubuntu@ip-172-31-47-173:~$
```

Obtener el certificado y configurar HTTPS automáticamente

Ejecutamos:

sudo certbot --apache -d nomeolvides.duckdns.org

Nos preguntará si aceptamos sus términos de servicio y si queremos usar nuestro e-mail

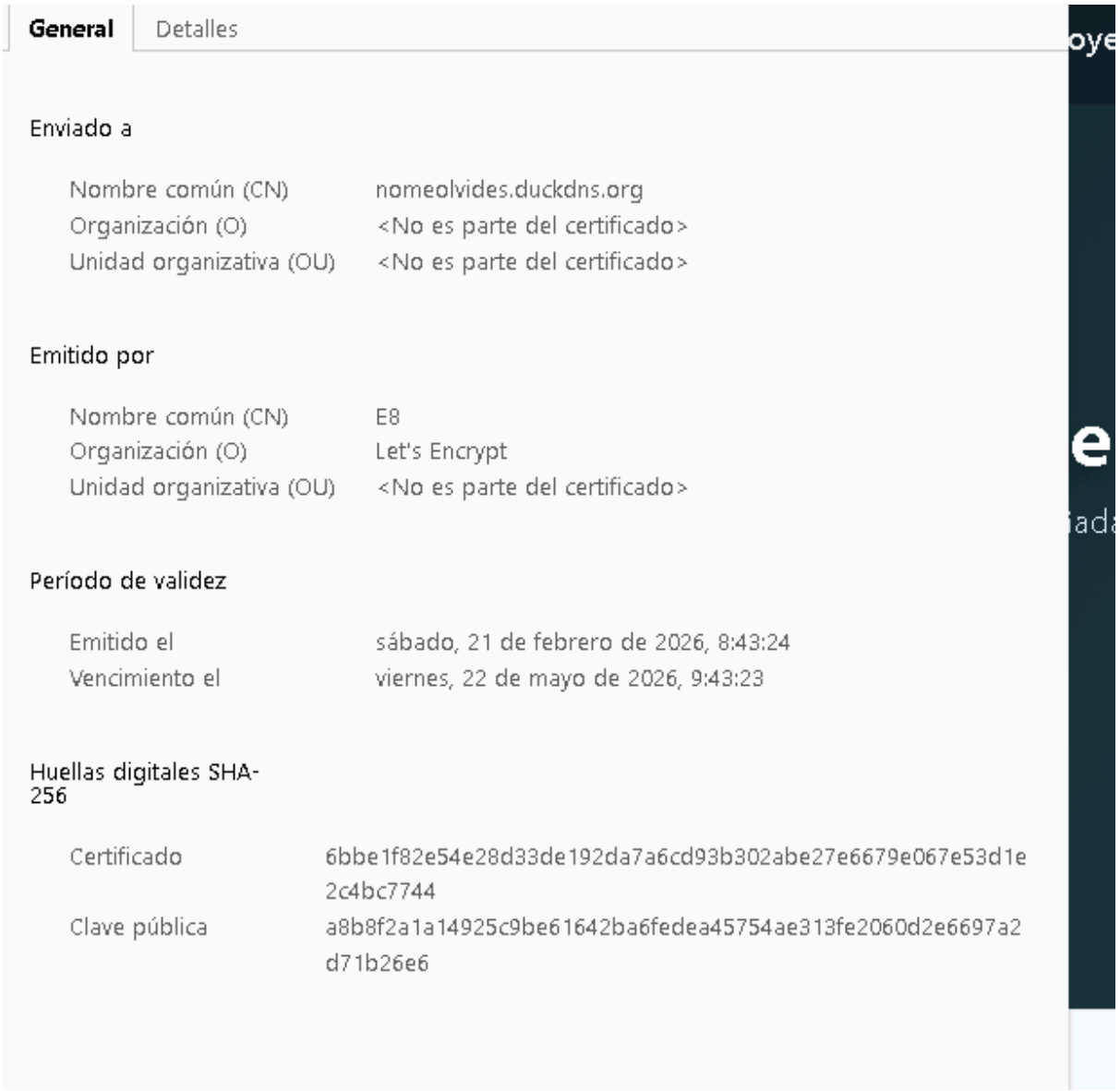
```
Enter email address or hit Enter to skip.
 (Enter 'c' to cancel): samupt7@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at:
https://letsencrypt.org/documents/LE-SA-v1.6-August-18-2025.pdf
You must agree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: yes

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: n
Account registered.
Requesting a certificate for nomeolvides.duckdns.org
```

Ahora podemos entrar directamente por HTTPS:

https://nomeolvides.duckdns.org



Incluso si intentamos entrar por HTTP nos redirigirá automáticamente a su versión más segura,todo esto sin necesidad de configuraciones lentas y tediosas gracias a certbot:

```
<VirtualHost *:80>
    ServerName nomeolvides.duckdns.org
    DocumentRoot /var/www/html

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/nomeolvides_error.log
    CustomLog ${APACHE_LOG_DIR}/nomeolvides_access.log combined
RewriteEngine on
RewriteCond %{SERVER_NAME} =nomeolvides.duckdns.org
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

sudo nano /etc/apache2/sites-available/nomeolvides-le-ssl.conf

También crea directamente estos archivos importantes para que HTTPS funcione

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName nomeolvides.duckdns.org
    DocumentRoot /var/www/html

    <Directory /var/www/html>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/nomeolvides_error.log
    CustomLog ${APACHE_LOG_DIR}/nomeolvides_access.log combined

SSLCertificateFile /etc/letsencrypt/live/nomeolvides.duckdns.org/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/nomeolvides.duckdns.org/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>     ^O Write Out    ^W Where Is     ^K Cut          ^T Execute  Read 1^C Location
```

Configuración segura de TLS

Certbot crea este archivo:

/etc/letsencrypt/options-ssl-apache.conf

```
# This file contains important security parameters. If you modify this file
# manually, Certbot will be unable to automatically provide future security
# updates. Instead, Certbot will print and log an error message with a path to
# the up-to-date file that you will need to refer to when manually updating
# this file. Contents are based on https://ssl-config.mozilla.org

SSLEngine on

# Intermediate configuration, tweak to your needs
SSLProtocol             all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
SSLOpenSSLConfCmd       Curves X25519:prime256v1:secp384r1
SSLCipherSuite          ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-G
SSLHonorCipherOrder     offA20-
SSLSessionTickets       off

SSLOptions +StrictRequire
```

Renovación automática

Los certificados de Let's Encrypt duran **90 días**.

Sin Certbot tendrías que renovarlos manualmente.

Certbot crea automáticamente una tarea programada (systemd timer o cron):

certbot renew

Gestión automática de certificados

Certbot organiza todos los certificados en:

/etc/letsencrypt/

Activar módulos necesarios de Apache

Certbot activa automáticamente módulos que HTTPS necesita.

Los principales son:

mod_ssl

Permite usar TLS/SSL en Apache.

Sin él HTTPS no funciona.

Se activaría manualmente así:

sudo a2enmod ssl

6mod_rewrite

Se usa para la redirección HTTP → HTTPS.

sudo a2enmod rewrite

mod_socache_shmcb

Se usa para el almacenamiento en caché de sesiones SSL.

sudo a2enmod socache_shmcb

El PHP en sí

Crear la carpeta y dar permisos (Crucial)

Para que PHP pueda guardar los archivos, el directorio uploads debe existir y el servidor web (Apache) debe tener permisos para escribir en él. Ejecuta estos comandos en la terminal de tu Ubuntu:

 # 1. Crear la carpeta uploads dentro de nuestro document root

sudo mkdir -p /var/www/html/uploads

 # 2. Cambiar el propietario de la carpeta al usuario de Apache (www-data)

sudo chown -R www-data:www-data /var/www/html/uploads

 # 3. Asignar los permisos correctos

sudo chmod -R 755 /var/www/html/uploads

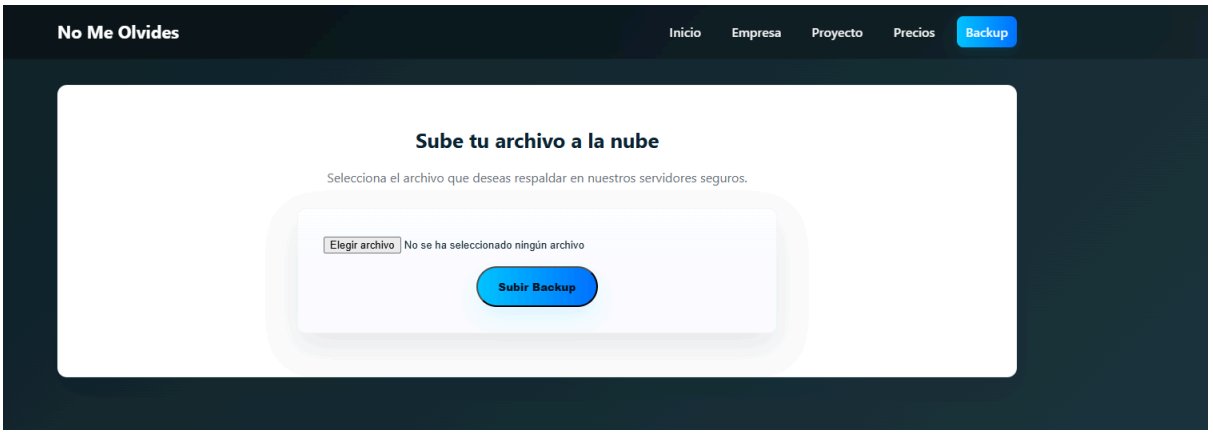
IMPORTANTE:SUBIR ARCHIVOS

sudo apt update
sudo apt install php libapache2-mod-php

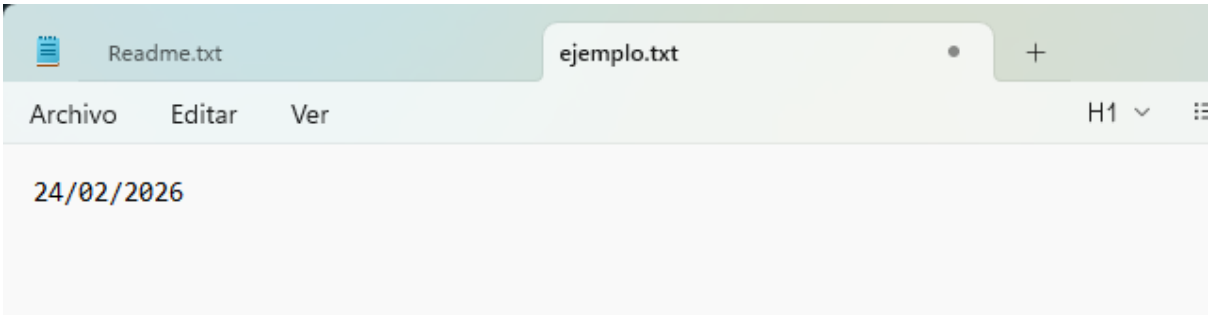
Reinicia Apache

Una vez instalado, tienes que reiniciar el servidor web para que detecte el nuevo módulo de PHP:

sudo systemctl restart apache2

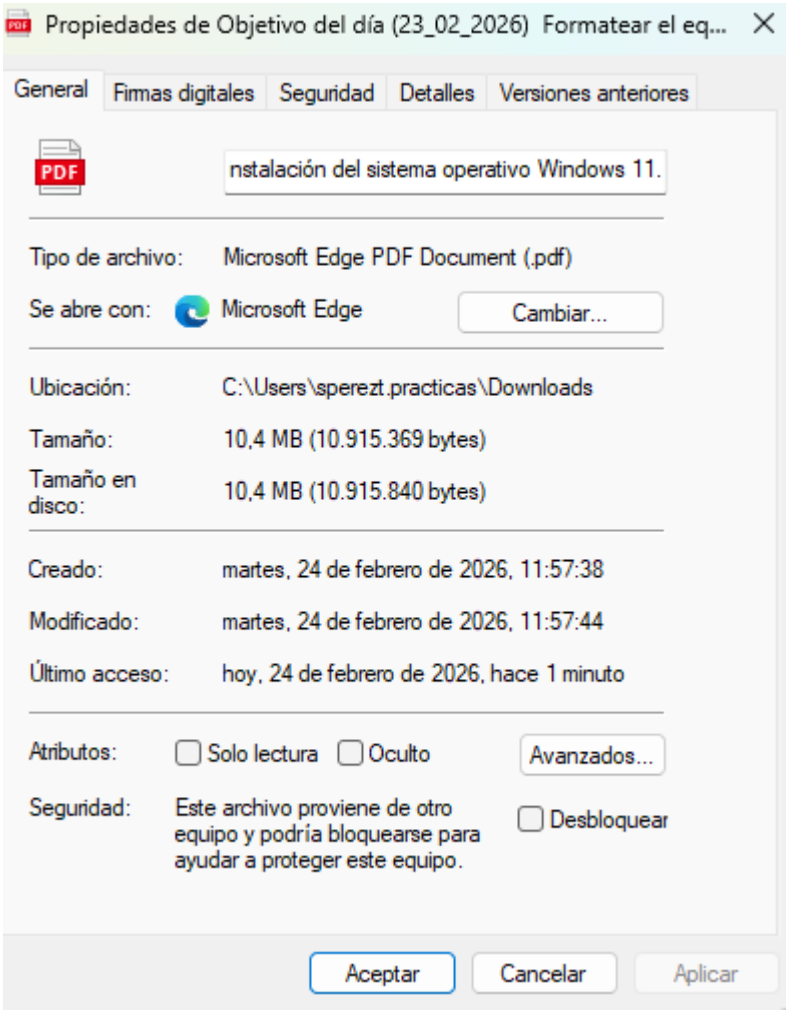


Ejemplo:
creamos un documento pequeño y lo subimos:

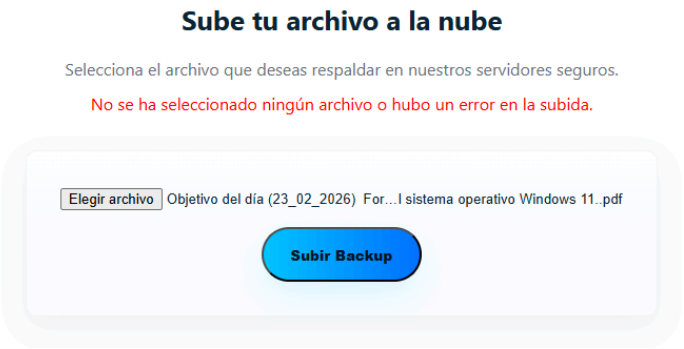


Vemos que se han subido correctamente:

```
ubuntu@ip-172-31-47-173:/var/www/html/uploads$ ls
CodeTemp.png  DxDiag.txt  ejemplo.txt
ubuntu@ip-172-31-47-173:/var/www/html/uploads$
```



Problema:PHP por defecto limita la subida de archivos a 2MB



Solución:cambiar el tamaño máximo de archivos:

sudo nano /etc/php/8.3/apache2/php.ini

```
/etc/php/8.3/apache2/php.ini
```

Dentro del editor, podemos usar Ctrl + W para buscar estas palabras clave más rápido. Cambiamos los valores que vienen por defecto para que queden así:

Buscamos upload_max_filesize y lo ponemos 100:
upload_max_filesize = 100M

Buscamos post_max_size y lo ponemos 100:
post_max_size = 100M

Buscamos max_execution_time (para que no se corte la subida a medias) y lo ponemos a 300:
max_execution_time = 300

Es mucho texto y el control+w

En Linux podemos usar un comando llamado **sed** que busca y reemplaza texto automáticamente sin tener que abrir el archivo. Solo copia y pega estas tres líneas en tu terminal, una por una, y dale a Enter.

Para cambiar el límite de subida a 100M:

sudo sed -i 's/upload_max_filesize = 2M/upload_max_filesize = 100M/' /etc/php/8.3/apache2/php.ini
Para cambiar el límite del formulario a 100M:

sudo sed -i 's/post_max_size = 8M/post_max_size = 100M/' /etc/php/8.3/apache2/php.ini
Para dar más tiempo de subida (de 30 a 300 segundos):

sudo sed -i 's/max_execution_time = 30/max_execution_time = 300/' /etc/php/8.3/ap

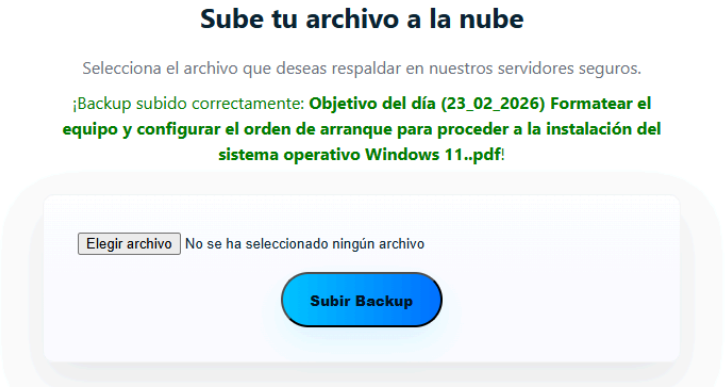
como vemos,se cambió

```
; Maximum allowed size for uploaded files.
; https://php.net/upload-max-filesize
upload_max_filesize = 100M
```

También podemos buscar con f6

Como siempre,debemos
sudo systemctl restart apache2

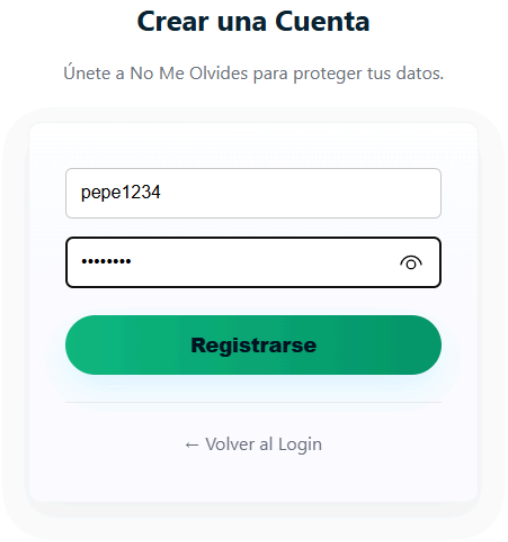
Probamos con el mismo archivo y vemos que,efectivamente,funciona



Siguiente paso:aumento de seguridad.

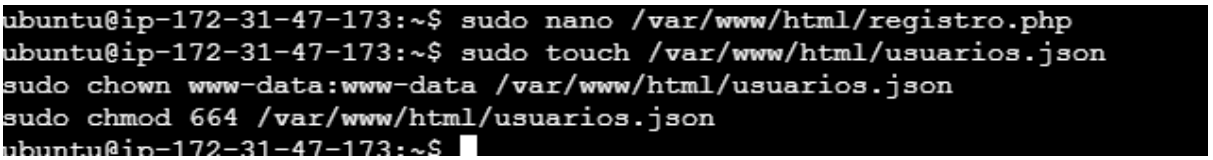
Tenemos una página funcional para subir y almacenar archivos,pero se almacenan de manera insegura en uploads y cualquiera podría acceder.

Puse un PHP para cada elemento: uno para crear cuenta,otro para iniciar sesión,etc…

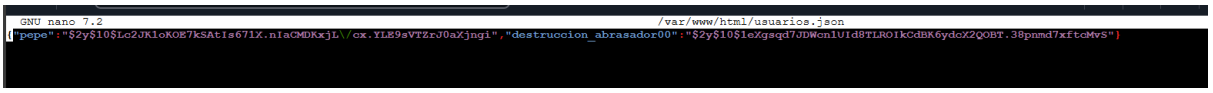


sudo touch /var/www/html/usuarios.json
sudo chown www-data:www-data /var/www/html/usuarios.json
sudo chmod 664 /var/www/html/usuarios.json

Primer 6 (Dueño): 4 (Leer) + 2 (Escribir) = 6. El dueño del archivo puede leerlo y modificarlo.
Segundo 6 (Grupo): 4 (Leer) + 2 (Escribir) = 6. Los usuarios que pertenezcan al grupo pueden leerlo y modificarlo.
Tercer 4 (Otros): 4 (Leer) = 4. Cualquier otra persona en el servidor solo puede leer, no puede modificar ni borrar nada.



La información de los usuarios se almacena en un .jason



{"pepe":"$2y$10$Lc2JK1oKOE7kSAtIs671X.nIaCMDKxjL\/cx.YLE9sVTZrJ0aXjngi","destruccion_abrasador00":"$2y$10$1eXgsqd7JDWcn1UId8TLROIkCdBK6ydcX2QOBT.38pnmd7xftcMvS"}

como vemos,tras crear un usuario con su contraseña correspondiente,se cifran los datos automáticamente

Aunque no hemos trabajado con JSON durante el curso, como el proyecto no incluía montar una base de datos pesada como MySQL, necesitábamos una forma de guardar los usuarios para arreglar el problema de seguridad de las subidas públicas.

Decidimos usar un archivo JSON. En sistemas, esto no es más que un fichero de texto plano estructurado. Lo elegimos por tres razones principales:

Ahorro de recursos: Instalar MySQL solo para dos usuarios habría saturado nuestra instancia gratuita de AWS innecesariamente.

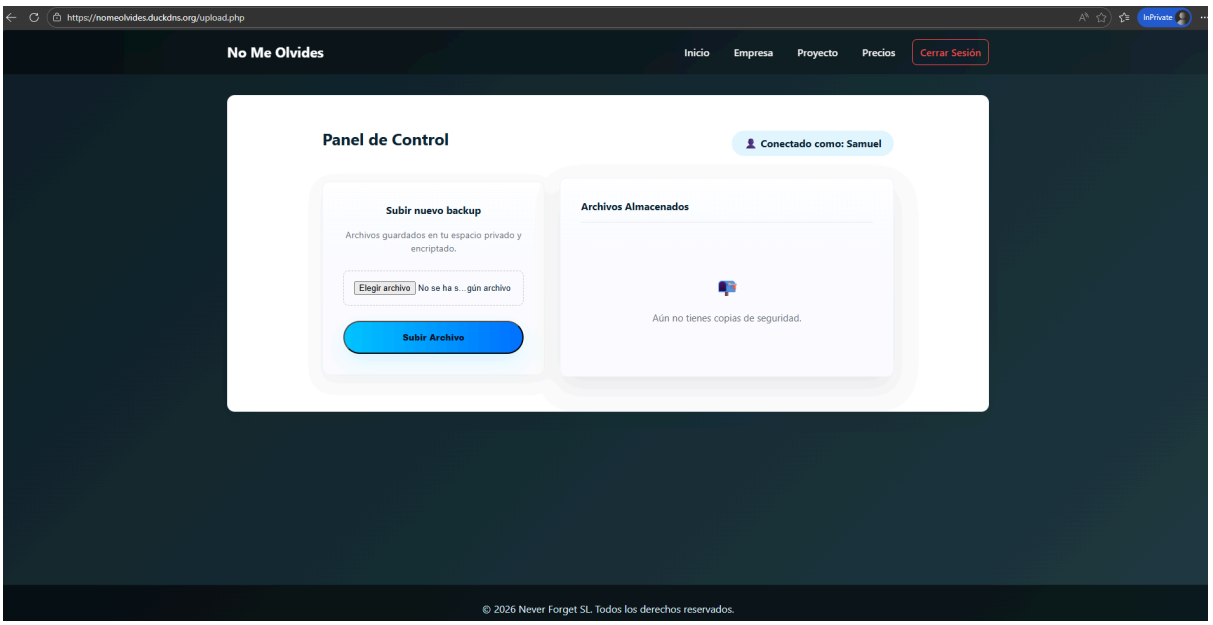
Sencillez: Nos permite guardar las contraseñas encriptadas en un simple archivo.

Seguridad en Linux: Al ser un archivo físico (**usuarios.json**), pudimos usar los comandos chown y chmod 664 para garantizar que solo el servicio de Apache tenga permiso para leerlo y escribirlo."

Además,cada vez que se añada un usuario,se creará un directorio privado único en el que cada usuario gozará de privacidad,orden y comodidad.

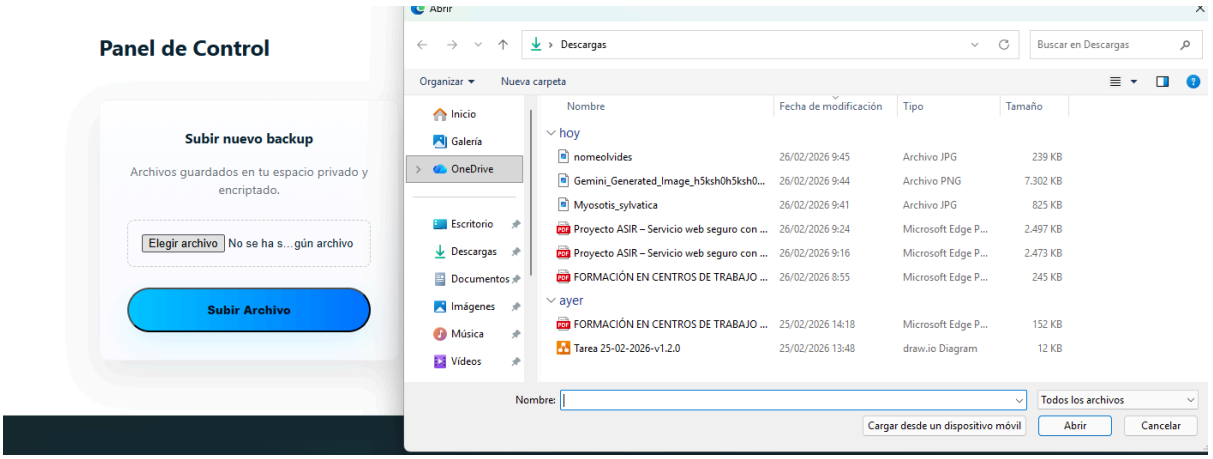
Actualmente, el sistema ya nos permite autenticarnos y subir archivos de forma segura. Sin embargo, en un entorno de producción real nos enfrentamos a una necesidad crítica: **¿Cómo gestiona el usuario sus datos una vez alojados?**

Para resolver esto, hemos evolucionado el portal hacia un **Dashboard de Usuario con Interfaz CRUD completa**. Esto significa que el cliente no solo 'lanza' datos a la nube, sino que dispone de un panel bidireccional para **visualizar** su inventario en tiempo real, **descargar** sus backups con un solo clic y **eliminar** archivos obsoletos. Todo esto ocurre mediante una comunicación dinámica entre PHP y el sistema de archivos de Linux, garantizando que el usuario tenga el control total de su ciclo de vida de datos."

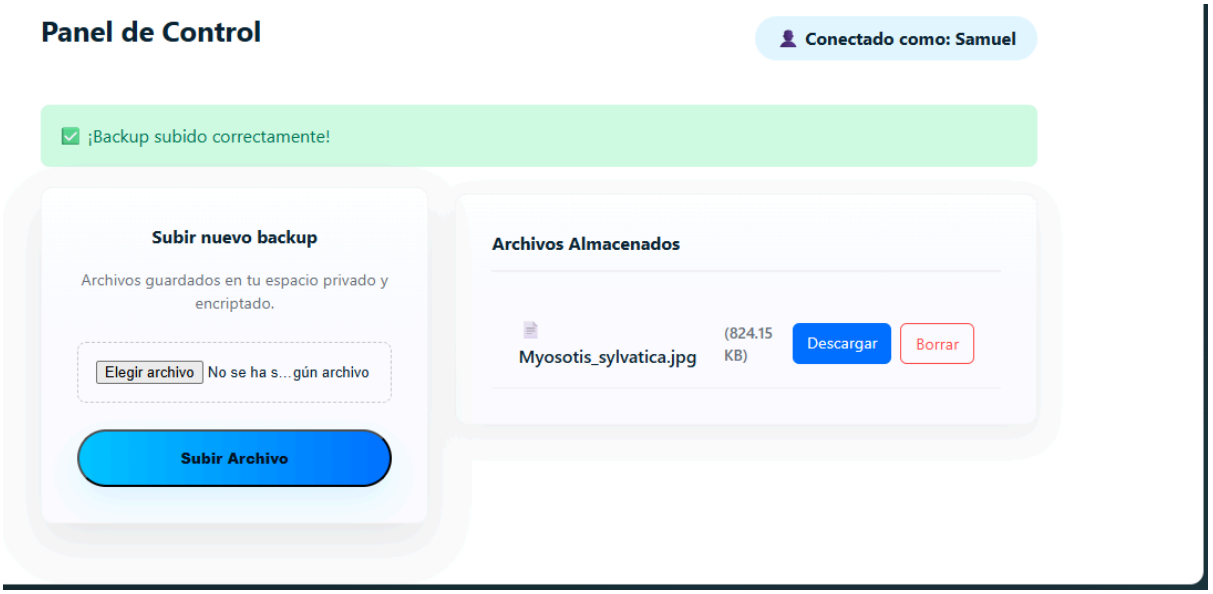


Aquí se puede apreciar un usuario sin ningún archivo.

Le damos a elegir archivo

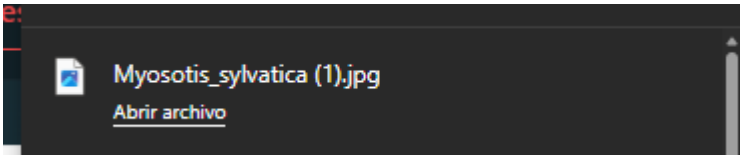


Le damos a subir archivo

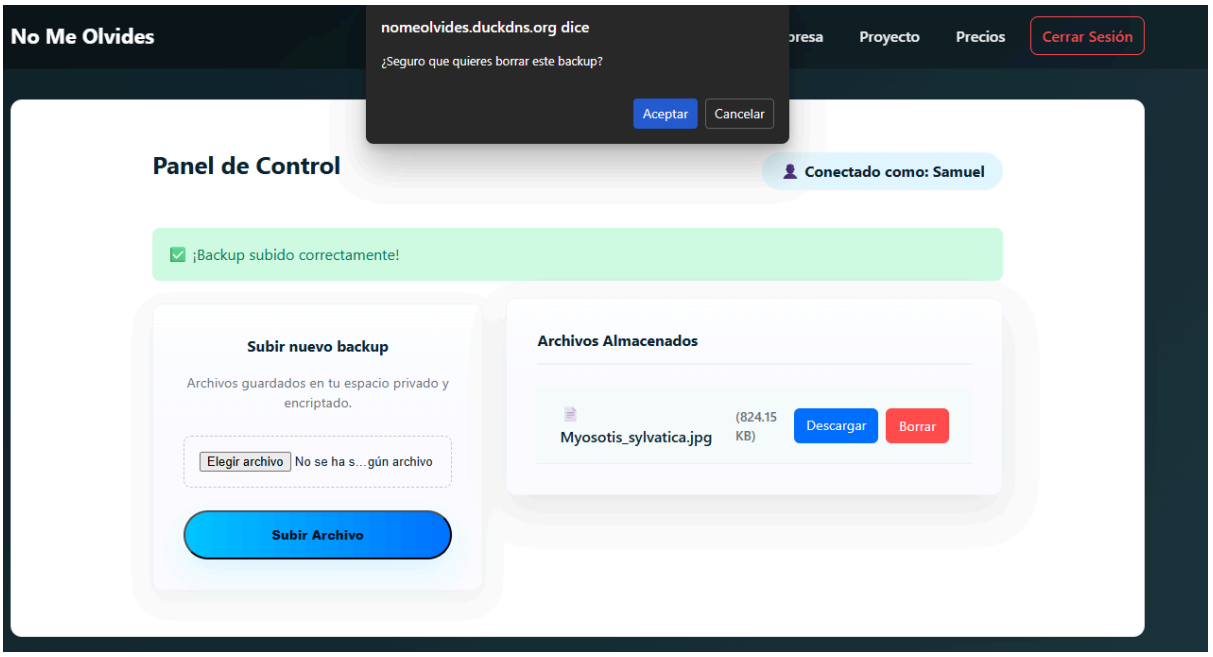


Se añade exitosamente

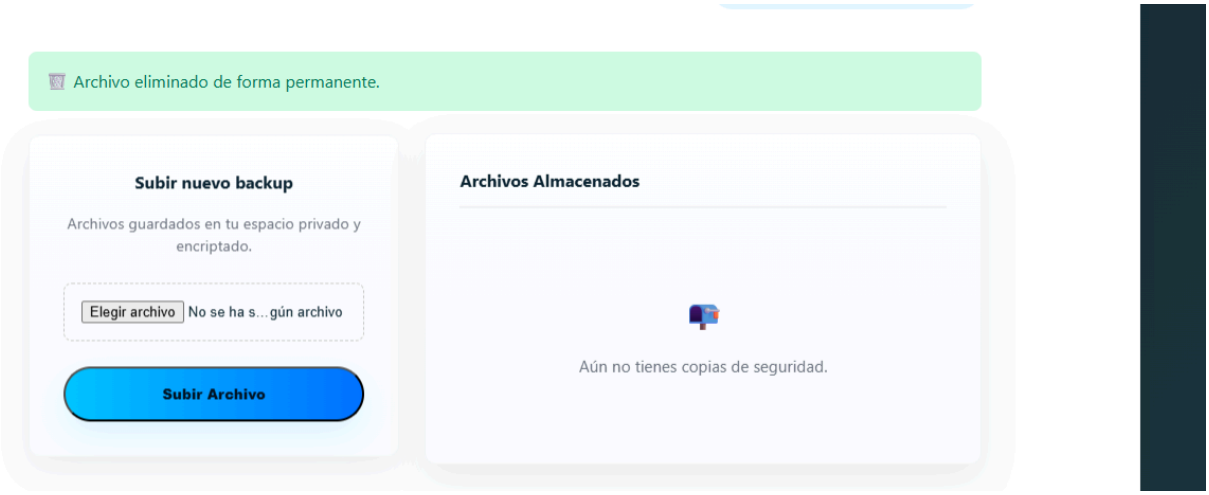
Le damos a descargar y se descarga



Si le hacemos click en borrar,saldrá una ventana emergente que nos preguntará si queremos borrarlo:



Si le hacemos click en “borrar”,se borra



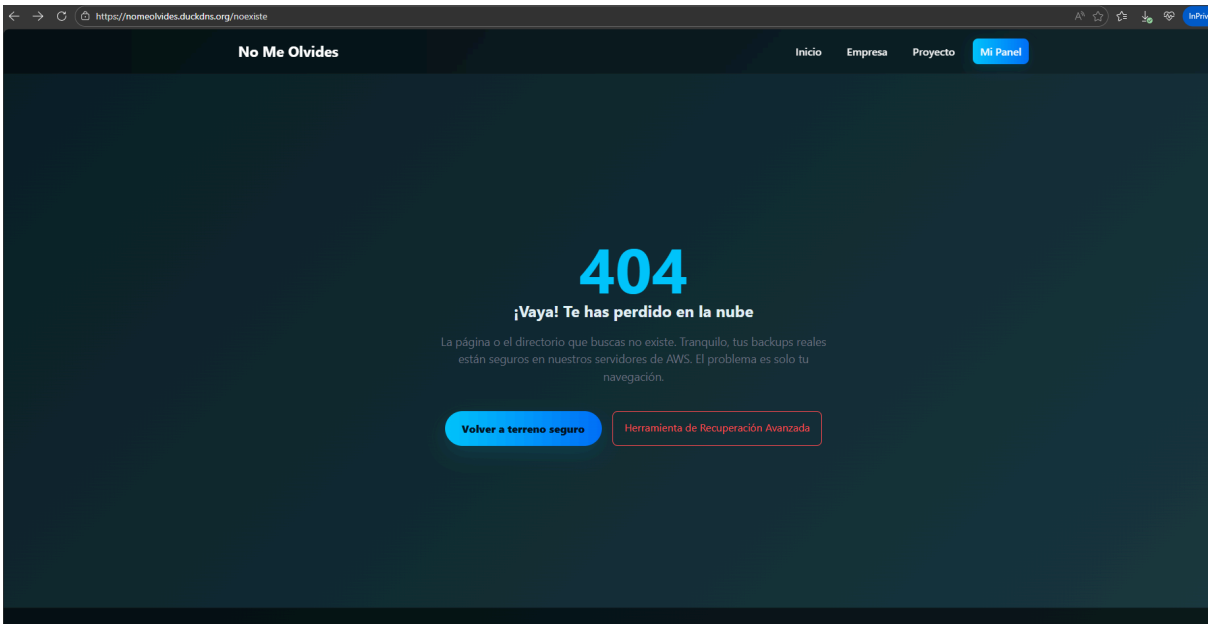
Ahora,como usuarios,tenemos poder para descargar o borrar nuestros archivos,es decir,una página web funcional y completa de backups

EXTRA:

Puse un html de error

En nuestro **Archivo de Configuración del Virtual Host seguro**.
ErrorDocument 404 /404.html

Ahora,si alguien intenta entrar



Vectores de Ataque y Medidas de Mitigación
A continuación, detallamos tres posibles vectores de ataque contra nuestra infraestructura y las medidas de seguridad que hemos implementado para mitigarlos:

1. Inyección o Fuerza Bruta en el Login (Ataque al Backend)
El escenario de amenaza: Un atacante podría intentar vulnerar nuestro formulario de autenticación mediante el uso de herramientas automatizadas (como Hydra) para realizar ataques de fuerza bruta contra usuarios conocidos, o intentando inyectar caracteres especiales (como ' OR 1=1;--) en los campos del formulario para evadir la validación.

Cómo lo hemos mitigado:

Protección contra inyección: Al prescindir de bases de datos relacionales tradicionales (SQL) y optar por un almacenamiento estructural basado en JSON, mitigamos por diseño los ataques clásicos de inyección SQL. Las funciones nativas que hemos utilizado en nuestro PHP (como json_encode y el manejo directo de arrays) procesan y escapan las cadenas de texto de forma segura.

Cifrado de credenciales blindado: En el hipotético caso de que un atacante lograse acceder a nuestro archivo usuarios.json, no obtendría las contraseñas en texto plano. Hemos implementado la función password_hash() de PHP, la cual genera hashes robustos utilizando el algoritmo Bcrypt, garantizando un estándar de seguridad criptográfica altísimo.

2. Path Traversal / Directory Traversal (Ataque al Panel CRUD)
El escenario de amenaza: En este caso, un usuario (o atacante registrado) podría intentar manipular los parámetros de las URL en las acciones de descarga o borrado del panel. Por ejemplo, modificando la variable upload.php?delete=foto.jpg por secuencias de escape como upload.php?delete=../../../../../etc/passwd con el objetivo de leer o destruir archivos críticos del sistema operativo Ubuntu.

Cómo lo hemos mitigado:

Saneamiento de rutas con basename(): En la lógica de nuestro archivo upload.php, procesamos todas las peticiones a través de la función basename(). Esta función elimina automáticamente cualquier intento de navegación relativa por directorios (como los ../), bloqueando la amenaza de raíz en la capa de aplicación.

Principio de mínimo privilegio (Permisos estrictos): Como medida de Defensa en Profundidad, hemos configurado permisos estrictos en el sistema de archivos de Linux (ej. chmod 664, 644). El servicio de Apache (usuario www-data) carece de privilegios de root, lo que le imposibilita alterar o eliminar archivos vitales del sistema, conteniendo drásticamente el impacto en caso de brecha.

3. Subida de Archivos Maliciosos (Ataque al Upload)
El escenario de amenaza: Un atacante autenticado podría aprovechar la función de "Subir Backup" para inyectar un archivo ejecutable malicioso (como virus.php) en lugar de un archivo legítimo. Su objetivo sería navegar posteriormente a la URL de ese archivo (/uploads/hacker/virus.php) para ejecutar una Web Shell y lograr acceso remoto por comandos al servidor.

Cómo lo hemos mitigado:

Filtrado estricto de extensiones: En el código de procesamiento de subidas, hemos programado un array de bloqueo (blocklist) que prohíbe explícitamente el alojamiento de extensiones ejecutables o de scripting (tales como php, php3, phtml, sh, js, cgi, py). Si el sistema detecta alguna de estas extensiones, aborta el guardado en disco y devuelve una alerta de seguridad al cliente, neutralizando por completo el intento de toma de control.

3.1. Estado de ejecución

Tomando como referencia el diagrama de Gantt inicial, el estado actual a finales de la Semana 3 es el siguiente:

- Tareas Completadas: Elección de plataforma (AWS EC2) , Solución VPN , Despliegue de Instancia , Configuración de Seguridad (UFW/Security Groups) , Setup SSH , Instalación Apache , Registro de Dominio

(DuckDNS) y Certificados HTTPS (Certbot),IP elástica,entre muchos otros...

- Tareas en Curso: Pruebas de subida y aislamiento de archivos de usuario. Redacción de memoria técnica.
- Tareas Retrasadas/Modificadas: Script de Backup y Configuración de Cron/Retención.

3.2. Registro de incidencias e Impacto

1. Bloqueo de red hacia AWS: No se podía acceder a la consola de AWS desde la red inicial. *Impacto:* Retraso inicial en el despliegue.
2. Límite de subida en PHP: El servidor bloqueaba archivos mayores a 2MB, vitales para un servicio de backup. *Impacto:* Imposibilidad temporal de subir documentos reales.

3. Riesgo de almacenamiento: Al verificar el espacio (df -h), la instancia t3.micro gratuita solo dispone de 3.9 GB libres. *Impacto:* Riesgo crítico de colapso del sistema operativo si se automatizan backups recursivos sin control.

3.3. Evaluación de las desviaciones

- Causas Técnicas: Las limitaciones por defecto de PHP (upload_max_filesize) y el almacenamiento reducido (8GB totales) de la capa gratuita de AWS.
- Causas Organizativas/Alcance: Se detectó que permitir subidas públicas sin autenticación comprometía la privacidad de los backups, un requisito no contemplado inicialmente que obligó a desviar horas de desarrollo hacia la creación de un sistema de login.
- Causas Externas: Restricciones de red (Firewall externo) que impedían la conexión a AWS.

3.4. Medidas correctoras aplicadas

- Ajuste 1 (Acceso): Uso de ProtonVPN para evadir el bloqueo perimetral y acceder a la consola de AWS. *Justificación:* Permitir la continuidad del proyecto sin cambiar de proveedor.

- Ajuste 2 (Configuración PHP): Modificación del archivo php.ini usando sed para aumentar el límite de subida a 100MB y el tiempo de ejecución a 300s. *Justificación:* Adaptar el servidor al caso de uso real (subida de backups pesados).
- Ajuste 3 (Cambio de Alcance - Seguridad): Desarrollo de un sistema de Login mediante *PHP Sessions* y carpetas aisladas por usuario. *Justificación:* Aumenta la seguridad del servicio sin requerir una base de datos pesada (cumpliendo el límite del proyecto original).
- Ajuste 4 (Mitigación de riesgo en Disco): Suspensión temporal de la tarea Cron. *Justificación:* Evitar el fallo crítico del servidor EC2 por falta de espacio de almacenamiento.