

ENRIQUE CRESPO RAMIREZ

✳️ Introducción a la Práctica 2 – Servidores web en WSL con Apache, Nginx y Caddy

✳️ Entorno de trabajo: WSL con Docker activo	3
⚙️ Qué aprenderemos en esta práctica	4
🧠 Conceptos que repasaremos	5
🧱 Paso 1 – Configuración del servidor Apache + PHP + HTTPS	5
1.1 ¿Qué es Apache y por qué lo usamos?	5
⚙️ 1.2 Instalación del servidor Apache y PHP	5
🔧 1.3 Configuración del puerto HTTP (8080)	8
📝 1.4 Prueba básica de PHP.....	10
🔒 1.5 Habilitar HTTPS (puerto 8443).....	11
🌐 ¿Qué es HTTPS?	11
✳️ Conclusión del bloque 1 (Apache + PHP + HTTPS)	16
🌐 Paso 2 – Instalación y configuración del servidor Nginx	16
🔍 2.1 ¿Qué es Nginx? 🔎 Concepto	17
✳️ Usos comunes de Nginx:.....	17
✳️ 3.1 ¿Por qué mover Nginx al 8081?.....	18
🔧 3.2 Editar la configuración del sitio por defecto	18
📋 3.3 Probar la sintaxis y reiniciar Nginx.....	19
🔍 3.4 Verificar que Nginx escucha en el 8081	20
📝 3.5 Probar desde la terminal o el navegador.....	20
📋 3.6 Comprobación de coexistencia con Apache.....	20
✅ Conclusión del paso 3.....	21
🕒 Paso 4 – Instalación y configuración de Caddy (puerto 8082).....	21
🧠 4.1 ¿Qué es Caddy?	21

⚙️	4.2 Instalar dependencias de repositorios externos.....	21
	4.3 Añadir el repositorio oficial de Caddy	22
	4.3 Instalar Caddy	24
	4.4 Preparar el directorio web de Caddy.....	24
	4.5 Crear contenido de prueba (Markdown).....	24
	4.6 Añadir una imagen de prueba.....	25
	4.7 Configurar Caddyfile	25
📄	4.9 Comprobación final de coexistencia.....	27

En esta práctica vamos a configurar **tres servidores web diferentes** dentro del entorno **WSL (Windows Subsystem for Linux)**, concretamente usando la distribución **Ubuntu**. El objetivo es aprender, desde cero, cómo funcionan distintos tipos de servidores web y cómo configurarlos de manera profesional para servir contenido mediante **HTTP** y **HTTPS**.

Durante la práctica se desplegarán y configurarán los siguientes servicios:

Servidor	Puerto(s)	Función principal
Apache + PHP	8080 (HTTP) / 8443 (HTTPS)	Servidor clásico con soporte para PHP y conexión segura mediante SSL (HTTPS). Ideal para ejecutar aplicaciones dinámicas.
Nginx	8081	Servidor web ligero y de alto rendimiento, especializado en contenido estático (HTML, CSS, JS) y usado frecuentemente como proxy inverso.
Caddy	8082	Servidor moderno y minimalista, con configuración sencilla y soporte nativo para HTTPS automático. Muy utilizado en entornos cloud.



Entorno de trabajo: WSL con Docker activo

Esta práctica se realizará dentro de **WSL (Ubuntu sobre Windows)**.

Actualmente tenemos **dos contenedores Docker activos** en tu entorno (`web_apache` y `web_nginx`), los cuales podrían estar usando los puertos **8080** y **8081**.

Por ese motivo, antes de iniciar la configuración de los servidores, **detendremos temporalmente los contenedores** para liberar los puertos y evitar conflictos.



Comandos básicos para preparar el entorno

1 docker ps -a

- Este comando lista todos los contenedores creados, incluso los que están detenidos.
- Te muestra su **nombre, estado, imagen usada y puertos asignados**.

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo docker ps -a
[sudo] password for kike:
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
58601889faae        php:8.2-apache    "docker-php-entrypoi..."   6 days ago        Exited (255) 19 minutes ago   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   web_apache
1ca3336f15f9        nginx:stable       "/docker-entrypoint..."   6 days ago        Exited (255) 19 minutes ago   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp   web_nginx
```

2 sudo docker stop web_apache web_nginx

Detiene los contenedores llamados web_apache y web_nginx.

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo docker stop web_apache web_nginx
web_apache
web_nginx
kike@Double-KK:/mnt/c/Windows/System32$
```

3 comprobar que los puertos están libres

- lsof muestra los procesos que están usando un puerto.
- Si el puerto está libre, aparecerá el mensaje "808X libre".
- Esto es fundamental, porque si un servicio (como Docker) sigue ocupando un puerto, el nuevo servidor no podrá arrancar.

```
sudo lsof -i :8080 -P -n || echo "8080 libre"
```

```
sudo lsof -i :8081 -P -n || echo "8081 libre"
```

```
sudo lsof -i :8082 -P -n || echo "8082 libre"
```

```
sudo lsof -i :8443 -P -n || echo "8443 libre"
```

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo lsof -i :8080 -P -n || echo "8080 libre"
8080 libre
kike@Double-KK:/mnt/c/Windows/System32$ sudo lsof -i :8081 -P -n || echo "8081 libre"
8081 libre
kike@Double-KK:/mnt/c/Windows/System32$ sudo lsof -i :8082 -P -n || echo "8082 libre"
8082 libre
kike@Double-KK:/mnt/c/Windows/System32$ sudo lsof -i :8443 -P -n || echo "8443 libre"
8443 libre
```

💡 Qué aprenderemos en esta práctica

1. **Instalar y configurar** distintos servidores web (Apache, Nginx y Caddy) en WSL.
2. **Modificar puertos**, crear **archivos de configuración** y servir contenido web local.
3. **Probar conexiones HTTP y HTTPS** de forma segura (usando certificados autofirmados).

4. **Diagnosticar errores comunes** (como conflictos de puertos o fallos de servicio).
5. **Documentar todo en GitHub**, como si fuera un entorno profesional de desarrollo.

Conceptos que repasaremos

- **Servidor web**: software que entrega contenido (páginas HTML, imágenes, etc.) a través del protocolo HTTP/HTTPS.
- **Apache**: el servidor web más usado, flexible y modular, con soporte nativo para PHP.
- **Nginx**: diseñado para alto rendimiento, muy ligero y usado como proxy inverso.
- **Caddy**: servidor moderno con configuración sencilla y HTTPS automático.
- **HTTPS**: versión segura del protocolo HTTP; cifra la conexión mediante SSL/TLS.
- **Certificado autofirmado**: tipo de certificado generado por uno mismo, útil para entornos de pruebas o prácticas.
- **WSL (Windows Subsystem for Linux)**: capa de compatibilidad que permite ejecutar un sistema Linux dentro de Windows sin máquina virtual completa.

Paso 1 – Configuración del servidor Apache + PHP + HTTPS

1.1 ¿Qué es Apache y por qué lo usamos?

Apache HTTP Server es uno de los servidores web más utilizados en el mundo. Su función principal es **entregar contenido web** (como páginas HTML, imágenes o aplicaciones PHP) a los clientes que lo solicitan mediante el protocolo **HTTP** o **HTTPS**.

En esta práctica, usaremos **Apache combinado con PHP** para crear un servidor web dinámico que funcione tanto por **HTTP (puerto 8080)** como por **HTTPS (puerto 8443)**.

1.2 Instalación del servidor Apache y PHP

Primero instalamos Apache junto con el módulo necesario para que pueda ejecutar PHP.

```
sudo apt install apache2 php libapache2-mod-php -y
```

🔍 Explicación del comando:

- sudo: ejecuta el comando con privilegios de administrador.
- apt install: instala uno o más paquetes del repositorio de Ubuntu.
- apache2: instala el servidor web Apache.
- php: instala el intérprete del lenguaje PHP.
- libapache2-mod-php: permite que Apache ejecute archivos .php como parte del contenido web.
- -y: acepta automáticamente las confirmaciones de instalación.

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo apt install apache2 php libapache2-mod-php -y
[sudo] password for kike:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.8).
php is already the newest version (2:8.3+93ubuntu2).
libapache2-mod-php is already the newest version (2:8.3+93ubuntu2).
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
kike@Double-KK:/mnt/c/Windows/System32$
```

Ahora comprobamos que apache está funcionando correctamente con:

```
sudo apt install apache2 php libapache2-mod-php -y
```

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
  Active: failed (Result: exit-code) since Wed 2025-10-15 12:02:38 CEST; 45min ago
    Docs: https://httpd.apache.org/docs/2.4/
   Process: 170 ExecStart=/usr/sbin/apachectl start (code=exited, status=1/FAILURE)
      CPU: 48ms

Oct 15 12:02:37 Double-KK systemd[1]: Starting apache2.service - The Apache HTTP Server...
Oct 15 12:02:38 Double-KK apachectl[204]: (98)Address already in use: AH00072: make_sock: could not bind to address [::]:80
Oct 15 12:02:38 Double-KK apachectl[204]: (98)Address already in use: AH00072: make_sock: could not bind to address 0.0.0.0:80
Oct 15 12:02:38 Double-KK apachectl[204]: no listening sockets available, shutting down
Oct 15 12:02:38 Double-KK apachectl[204]: AH00015: Unable to open logs
Oct 15 12:02:38 Double-KK systemd[1]: apache2.service: Control process exited, code=exited, status=1/FAILURE
Oct 15 12:02:38 Double-KK systemd[1]: apache2.service: Failed with result 'exit-code'.
Oct 15 12:02:38 Double-KK systemd[1]: Failed to start apache2.service - The Apache HTTP Server.
kike@Double-KK:/mnt/c/Windows/System32$
```

Como podemos comprobar Apache sigue intentando escuchar en **80** y alguien (probablemente **Nginx** o algún servicio) ya está usando ese puerto.

Para solucionarlo deberemos seguir unos pasos:

1) Ver quién ocupa el 80

```
sudo ss -ltnp | grep ':80' || echo "80 libre"
```

```
sudo lsof -i :80 -P -n || echo "80 libre"
```

- `ss -ltnp` muestra sockets TCP en escucha con PID/proceso.
 - `lsof -i :80` lista procesos que usan el puerto 80.

Como podemos observar en la captura **Gnix** está escuchando en el puerto 80.

2 Parar Nginx (liberar el 80)

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo systemctl stop nginx
kike@Double-KK:/mnt/c/Windows/System32$ sudo ss -ltnp | grep ':80' || echo "Puerto 80 libre"
Puerto 80 libre
kike@Double-KK:/mnt/c/Windows/System32$
```

3 hacemos un Restart Apache 2 para reiniciar el servidor y posteriormente comprobamos si se ha levantado.

`systemctl` es una herramienta para gestionar servicios en Linux.

El parámetro `status` muestra si Apache está **activo (running)** o **inactivo (stopped)**.

```
KIKE@Double-KK:~$ 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 Wed 2025-10-15 12:55:26 CEST; 5s ago
     Docs: https://httpd.apache.org/docs/2.4/
 Process: 1881 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 1884 (apache2)
   Tasks: 6 (limit: 37958)
  Memory: 17.4M (peak: 18.8M)
    CPU: 39ms
   CGroup: /system.slice/apache2.service
           ├─1884 /usr/sbin/apache2 -k start
           ├─1886 /usr/sbin/apache2 -k start
           ├─1887 /usr/sbin/apache2 -k start
           ├─1888 /usr/sbin/apache2 -k start
           ├─1889 /usr/sbin/apache2 -k start
           └─1890 /usr/sbin/apache2 -k start

Oct 15 12:55:26 Double-KK systemd[1]: Starting apache2.service - The Apache HTTP Server...
Oct 15 12:55:26 Double-KK systemd[1]: Started apache2.service - The Apache HTTP Server.
[1]+ 12:55:26 httpd[1884] Terminated
```

Como podemos observar en la captura el servidor está levantando asique continuamos paso a paso con el trabajo.

1.3 Configuración del puerto HTTP (8080)

Por defecto, Apache escucha en el puerto **80**, pero en esta práctica debe funcionar en el **8080** para no interferir con otros servicios (como los contenedores Docker).

- 1 Editamos el archivo de configuración de puertos y cambiamos el puerto 80 por el 8080.

sudo nano /etc/apache2/ports.conf

```
kike@Double-KK: /mnt/c/Windows/System32
GNU nano 7.2
# If you just change the port or add more ports here, you will likely also
# 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>
```

- 2 Editamos el archivo del sitio por defecto y cambiamos el virtualhost 80 por el 8080.

El bloque `<VirtualHost>` define un “sitio web” dentro de Apache. Aquí indicamos que este sitio funcionará en el puerto 8080.

sudo nano /etc/apache2/sites-available/000-default.conf

```

[...]
kike@Double-KK: /mnt/c/Windows/System32
GNU nano 7.2
<VirtualHost *:8080>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

```

3 Reiniciamos Apache para aplicar los cambios y comprobamos su estado:

sudo systemctl restart apache2

sudo systemctl status apache2

```

[...]
kike@Double-KK:/mnt/c/Windows/System32$ sudo systemctl restart apache2
kike@Double-KK:/mnt/c/Windows/System32$ 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 Wed 2025-10-15 13:02:58 CEST; 4s ago
    Docs: https://httpd.apache.org/docs/2.4/
   Process: 1921 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 1924 (apache2)
    Tasks: 6 (limit: 37958)
   Memory: 11.3M (peak: 12.7M)
      CPU: 36ms
     CGroup: /system.slice/apache2.service
             └─1924 /usr/sbin/apache2 -k start
                  ├─1926 /usr/sbin/apache2 -k start
                  ├─1927 /usr/sbin/apache2 -k start
                  ├─1928 /usr/sbin/apache2 -k start
                  ├─1929 /usr/sbin/apache2 -k start
                  └─1930 /usr/sbin/apache2 -k start

Oct 15 13:02:58 Double-KK systemd[1]: Starting apache2.service - The Apache HTTP Server...
Oct 15 13:02:58 Double-KK systemd[1]: Started apache2.service - The Apache HTTP Server.

```

1.4 Prueba básica de PHP

Vamos a crear un archivo de prueba para comprobar que PHP se ejecuta correctamente.

```
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
```

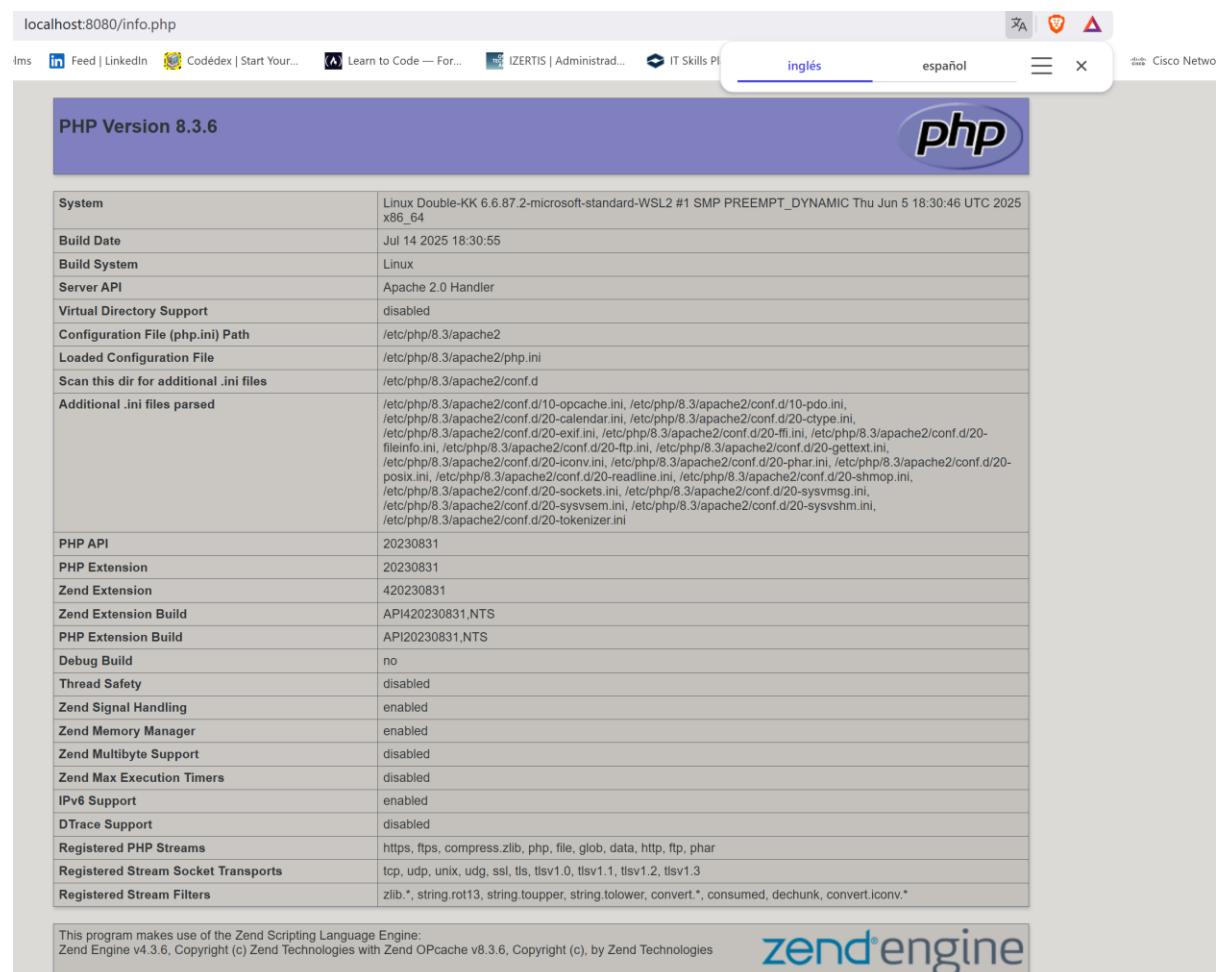
Qué hace:

- echo imprime el texto <?php phpinfo(); ?>.
- | envía esa salida al comando tee.
- sudo tee /var/www/html/info.php crea el archivo info.php en el directorio raíz de Apache.

Este archivo genera una página con toda la configuración de PHP (versión, módulos, etc.).

Ahora lo probamos en el navegador

<http://localhost:8080/info.php>



The screenshot shows a web browser window displaying the PHP info page. The URL in the address bar is <http://localhost:8080/info.php>. The page content is as follows:

PHP Version 8.3.6

System	
Build Date	Jul 14 2025 18:30:55
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d
Additional .ini files parsed	/etc/php/8.3/apache2/conf.d/10-opcache.ini, /etc/php/8.3/apache2/conf.d/10-pdo.ini, /etc/php/8.3/apache2/conf.d/20-calendar.ini, /etc/php/8.3/apache2/conf.d/20-ctype.ini, /etc/php/8.3/apache2/conf.d/20-exif.ini, /etc/php/8.3/apache2/conf.d/20-fi.ini, /etc/php/8.3/apache2/conf.d/20-gettext.ini, /etc/php/8.3/apache2/conf.d/20-ftp.ini, /etc/php/8.3/apache2/conf.d/20-iconv.ini, /etc/php/8.3/apache2/conf.d/20-phar.ini, /etc/php/8.3/apache2/conf.d/20-shmop.ini, /etc/php/8.3/apache2/conf.d/20-sockets.ini, /etc/php/8.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.3/apache2/conf.d/20-sysvsem.ini, /etc/php/8.3/apache2/conf.d/20-sysvshm.ini, /etc/php/8.3/apache2/conf.d/20-tokenizer.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831.NTS
PHP Extension Build	API20230831.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v4.3.6, Copyright (c) Zend Technologies with Zend OPcache v8.3.6, Copyright (c). by Zend Technologies

zend engine

1.5 Habilitar HTTPS (puerto 8443)

¿Qué es HTTPS?

HTTPS (HyperText Transfer Protocol Secure) es la versión segura de HTTP.

Cifra la comunicación entre el cliente (navegador) y el servidor mediante **certificados SSL/TLS**, garantizando la **confidencialidad** y **autenticidad** de los datos.

En esta práctica generaremos un **certificado autofirmado** (local), que servirá para activar HTTPS en el puerto **8443**.

- ## 1 Crear un certificado SSL autofirmado

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
```

```
-keyout /etc/ssl/private/apache-selfsigned.key \
-out /etc/ssl/certs/apache-selfsigned.crt
```

Explicación del comando:

- `openssl`: herramienta para manejar certificados y cifrado.
 - `req`: crea una nueva solicitud de certificado.
 - `-x509`: genera directamente un certificado autofirmado.
 - `-nodes`: sin cifrar la clave privada (para no pedir contraseña al iniciar Apache).
 - `-days 365`: el certificado será válido por un año.
 - `-newkey rsa:2048`: crea una nueva clave RSA de 2048 bits.
 - `-keyout` y `-out`: indican dónde guardar la clave privada y el certificado público.

Durante el proceso te pedirá algunos datos (país, organización, nombre del servidor, etc.).

2 Activar el módulo SSL y el sitio HTTPS

Habilitamos el módulo SSL:

sudo a2enmod ssl

💡 Qué hace:

Activa el módulo que permite a Apache usar HTTPS. (a2enmod = “Apache 2 enable module”)

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

Añadimos el puerto 8443 que indica a Apache que escuche peticiones HTTPS en ese puerto y comentamos los **Listen 443**.

- ◆ Esos bloques sirven **solo si usas el puerto 443 (HTTPS estándar)**.
- ◆ Como **vamos a usar 8443** (para evitar conflictos con otros servicios o Docker), **no necesitas mantener los Listen 443**.

sudo nano /etc/apache2/ports.conf

```
kike@Double-KK: /mnt/c/Windows/System32
GNU nano 7.2
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8443

<IfModule ssl_module>
#       Listen 443
</IfModule>

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

Configuramos el sitio HTTPS por defecto:

sudo nano /etc/apache2/sites-available/default-ssl.conf

Cambiamos el 443 por el 8443 y verificamos que los certificados sean los correctos.

```
GNU nano 7.2
<VirtualHost *:8443>
    ServerAdmin webmaster@localhost

    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf

    # SSL Engine Switch:
    # Enable/Disable SSL for this virtual host.
    SSLEngine on

    # A self-signed (snakeoil) certificate can be created by installing
    # the ssl-cert package. See
    # /usr/share/doc/apache2/README.Debian.gz for more info.
    # If both key and certificate are stored in the same file, only the
    # SSLCertificateFile directive is needed.
    SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
    SSLCertificateKeyFile   /etc/ssl/private/ssl-cert-snakeoil.key
```

Activamos el sitio HTTPS y reiniciamos Apache:

Systemctl reload apache2

sudo a2ensite default-ssl.conf

sudo systemctl restart apache2

- **a2ensite**: habilita un sitio virtual (en este caso el SSL).
- **systemctl restart**: reinicia Apache para aplicar los cambios.

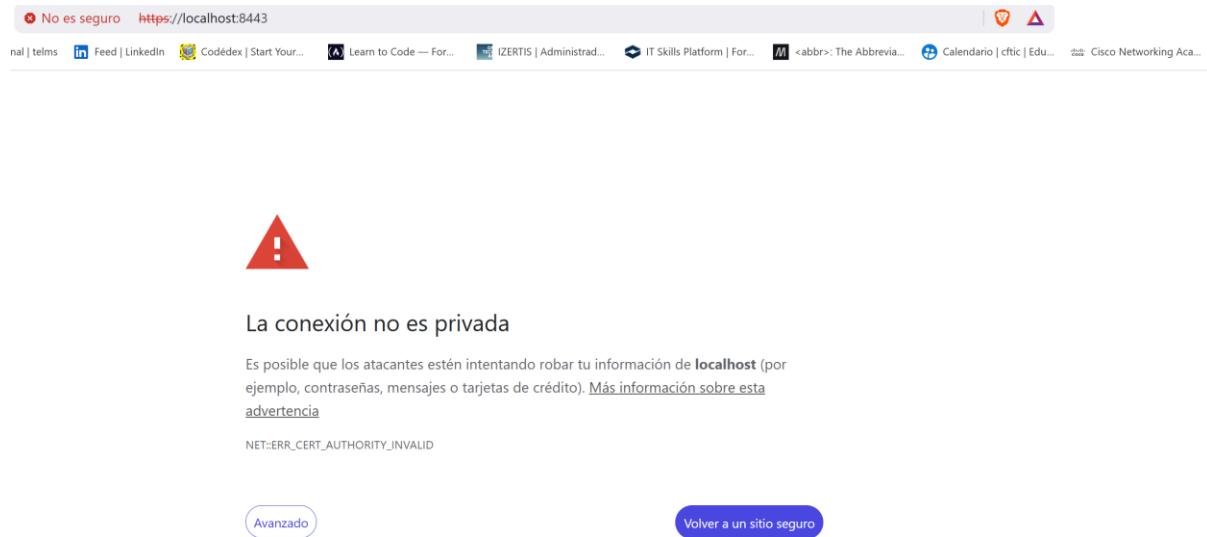
```

kike@Double-KK:/mnt/c/Windows/System32$ sudo systemctl reload apache2
kike@Double-KK:/mnt/c/Windows/System32$ sudo a2ensite default-ssl.conf
Site default-ssl already enabled
kike@Double-KK:/mnt/c/Windows/System32$ sudo systemctl restart apache2
kike@Double-KK:/mnt/c/Windows/System32$ 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 Wed 2025-10-15 13:23:07 CEST; 7s ago
      Docs: https://httpd.apache.org/docs/2.4/
   Process: 2104 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 2107 (apache2)
    Tasks: 6 (limit: 37958)
   Memory: 11.1M (peak: 12.7M)
      CPU: 38ms
     CGroup: /system.slice/apache2.service
             ├─2107 /usr/sbin/apache2 -k start
             ├─2109 /usr/sbin/apache2 -k start
             ├─2110 /usr/sbin/apache2 -k start
             ├─2111 /usr/sbin/apache2 -k start
             ├─2112 /usr/sbin/apache2 -k start
             └─2113 /usr/sbin/apache2 -k start

Oct 15 13:23:07 Double-KK systemd[1]: Starting apache2.service - The Apache HTTP Server...
Oct 15 13:23:07 Double-KK systemd[1]: Started apache2.service - The Apache HTTP Server.

```

Probamos la conexión HTTPS (ignorando la advertencia del certificado autofirmado):



Para confirmar que Apache está funcionando en ambos puertos:

```
sudo ss -tulpn | grep apache
```

Esto mostrará los **puertos TCP/UDP abiertos** y los procesos (como Apache) que los usan.

```
[root@Double-0X:/mnt/c/windows/System32]# sudo ss -tulpn | grep apache
tcp  LISTEN  0      511          *:8443           *:*      users:(("apache2",pid=2123,fds=4),("apache2",pid=2113,fds=4),("apache2",pid=2112,fds=4),("apache2",pid=2111,fds=4),("apache2",pid=2110,fds=4),("apache2",pid=2109,fds=4),("apache2",pid=2107,fds=4))
[root@Double-0X:/mnt/c/windows/System32]# sudo nano /etc/apache2/ports.conf
[root@Double-0X:/mnt/c/windows/System32]# sudo nano /etc/apache2/sites-available/000-default.conf
[root@Double-0X:/mnt/c/windows/System32]
```

Como podemos observar solo está escuchando el puerto 8443, y falta el 8080 que parece ser que se nos olvidó añadirlo al archivo de configuración , por lo que procedemos a modificarlo, **Y HACER UN RESTART!**

```
Listen 8443
Listen 8080
<IfModule ssl_module>
#      Listen 443
</IfModule>

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

Aquí podemos observar que antes solo escuchaba el 8443 y ahora escucha a los 2!!

```
[root@Double-0X:/mnt/c/windows/System32]# sudo nano /etc/apache2/sites-available/000-default.conf
[root@Double-0X:/mnt/c/windows/System32]# sudo ss -tulpn | grep apache
tcp  LISTEN  0      511          *:8443           *:*      users:(("apache2",pid=2123,fds=4),("apache2",pid=2112,fds=4),("apache2",pid=2113,fds=4),("apache2",pid=2111,fds=4),("apache2",pid=2110,fds=4),("apache2",pid=2109,fds=4),("apache2",pid=2107,fds=4))
tcp  LISTEN  0      511          *:8080           *:*      users:(("apache2",pid=2172,fds=4),("apache2",pid=2171,fds=4),("apache2",pid=2169,fds=4),("apache2",pid=2168,fds=4),("apache2",pid=2167,fds=4),("apache2",pid=2166,fds=4))
[root@Double-0X:/mnt/c/windows/System32]
```

A screenshot of a web browser window. At the top, there are navigation icons (back, forward, search). To the right of the address bar, it says "localhost:8080". Below the address bar, there are several links: "Inicio | ThePower FP -...", "Inicio | Microsoft 365...", "Área personal | telms", "Feed | LinkedIn", and "Codéx | Start". The main content area displays the text "Hola Mundo desde Nginx" in large bold letters, followed by "Servidor funcionando correctamente" in smaller text.

A screenshot of a web browser window. At the top, there are navigation icons (back, forward, search). To the right of the address bar, it says "No es seguro" and "https://localhost:8443". Below the address bar, there are several links: "Inicio | ThePower FP -...", "Inicio | Microsoft 365...", "Área personal | telms", "Feed | LinkedIn", and "Codéx | Start". The main content area displays the text "Hola Mundo desde Nginx" in large bold letters, followed by "Servidor funcionando correctamente" in smaller text.

Conclusión del bloque 1 (Apache + PHP + HTTPS)

Hasta este punto, hemos conseguido que **Apache funcione de forma segura en HTTPS y con soporte PHP**, estableciendo las bases del entorno de servidores web en la nube.

El servicio se encuentra **activo y operativo**, preparado para coexistir con los siguientes servidores que implementaremos (Nginx y Caddy).

Paso 2 – Instalación y configuración del servidor Nginx

⌚ 2.1 ¿Qué es Nginx? 🔎 Concepto

Nginx es un servidor web y proxy inverso de alto rendimiento.

Su arquitectura basada en eventos le permite manejar miles de conexiones simultáneamente con muy poca memoria.

Nginx (pronunciado “engine-x”) es un servidor web **ligero, rápido y eficiente**, diseñado para manejar una gran cantidad de conexiones simultáneas con bajo consumo de recursos.

A diferencia de Apache, que usa un modelo **basado en procesos** (crea un proceso por conexión), Nginx utiliza un modelo **asíncrono y basado en eventos**, lo que le permite manejar miles de peticiones sin saturar el sistema.

❖ Usos comunes de Nginx:

- Servidor web para **contenido estático** (HTML, CSS, imágenes, etc.).
- **Proxy inverso**, para redirigir tráfico a otros servicios (por ejemplo, balanceo entre varios Apache o Node.js).
- Servidor de **caché** para mejorar rendimiento.
- Servidor **SSL/TLS** con certificados HTTPS.

Nginx se utiliza mucho en entornos **cloud** y **contenedores Docker** por su eficiencia y simplicidad.

⚙ 2.2 Instalación de Nginx

sudo nano /etc/nginx/sites-available/default

- `apt install nginx`: instala el paquete nginx desde los repositorios de Ubuntu.
- `-y`: acepta automáticamente la instalación sin pedir confirmación.

```
kike@Double-KK:/mnt/c/Windows/System32$ sudo apt install nginx -y
[sudo] password for kike:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nginx is already the newest version (1.24.0-2ubuntu7.5).
The following package was automatically installed and is no longer required:
  libl1vm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
```

Comprobamos que está funcionando

sudo systemctl status nginx

```
Kike@Double-KK:/mnt/c/Windows/System32$ sudo systemctl status nginx - 
Unit .service could not be found.
● nginx.service - A high performance web server and a reverse proxy server
  Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
  Active: inactive (dead) since Wed 2025-10-15 12:54:28 CEST; 6h ago
    Duration: 51min 50.301s
      Docs: man:nginx(8)
   Process: 183 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 243 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 1403 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 255 (code=exited, status=0/SUCCESS)
    CPU: 72ms

Oct 15 12:02:37 Double-KK systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Oct 15 12:02:37 Double-KK systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
Oct 15 12:54:28 Double-KK systemd[1]: Stopping nginx.service - A high performance web server and a reverse proxy server...
Oct 15 12:54:28 Double-KK systemd[1]: nginx.service: Deactivated successfully.
Oct 15 12:54:28 Double-KK systemd[1]: Stopped nginx.service - A high performance web server and a reverse proxy server.
```

⚙️ 3. Configuración de Nginx en el puerto 8081

🔧 3.1 ¿Por qué mover Nginx al 8081?

Apache ya está utilizando los puertos 8080 (HTTP) y 8443 (HTTPS).

Para que ambos servidores puedan funcionar de forma simultánea, debemos cambiar el puerto de Nginx.

Así, Apache y Nginx no compiten por el mismo recurso y podrás acceder a cada uno por un puerto distinto.

🔧 3.2 Editar la configuración del sitio por defecto

Abrimos el archivo del sitio principal de Nginx:

sudo nano /etc/nginx/sites-available/default

Cambiamos los puertos 80 por los 8081

```

kike@Double-KK:/mnt/c/Windows/System32
GNU nano 7.2
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##


# Default server configuration
#
server {
    listen 8081 default_server;
    listen [::]:8081 default_server;

```

3.3 Probar la sintaxis y reiniciar Nginx

Antes de reiniciar, validamos la configuración:

sudo nginx -t

nginx -t = comprueba errores de sintaxis

```

kike@Double-KK:/mnt/c/Windows/System32$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

```

Reinicia el servicio:

```

kike@Double-KK:/mnt/c/Windows/System32$ 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 Wed 2025-10-15 19:31:30 CEST; 10s ago
     Docs: man:nginx(8)
 Process: 3142 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Process: 3144 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 3145 (nginx)
   Tasks: 17 (limit: 37958)
      Memory: 12.3M (peak: 13.8M)
        CPU: 30ms
       CGroup: /system.slice/nginx.service
           ├─3145 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
           ├─3146 "nginx: worker process"
           ├─3147 "nginx: worker process"
           ├─3148 "nginx: worker process"
           ├─3149 "nginx: worker process"
           ├─3150 "nginx: worker process"
           ├─3151 "nginx: worker process"
           ├─3152 "nginx: worker process"
           ├─3153 "nginx: worker process"
           ├─3154 "nginx: worker process"
           ├─3155 "nginx: worker process"
           ├─3156 "nginx: worker process"
           ├─3157 "nginx: worker process"
           ├─3158 "nginx: worker process"
           ├─3159 "nginx: worker process"
           ├─3160 "nginx: worker process"
           ├─3161 "nginx: worker process"
           └─3162 "nginx: worker process"

Oct 15 19:31:30 Double-KK systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Oct 15 19:31:30 Double-KK systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.

```

3.4 Verificar que Nginx escucha en el 8081

```
sudo ss -ltnp | grep nginx
```

3.5 Probar desde la terminal o el navegador



3.6 Comprobación de coexistencia con Apache

Comprobamos que cada servicio está escuchando en su puerto correspondiente:

8080-APACHE2

80443-APACHE2

8081-NGINX

💡 Ahora los dos servidores web pueden funcionar simultáneamente sin interferirse.

Conclusión del paso 3

- Nginx fue reconfigurado para usar el **puerto 8081**.
- Apache (8080/8443) y Nginx (8081) ahora operan **simultáneamente y sin conflicto**.
- Se puede acceder a cada servidor desde el navegador o curl para verificar su funcionamiento.

Paso 4 – Instalación y configuración de Caddy (puerto 8082)

4.1 ¿Qué es Caddy?

Caddy es un servidor web moderno, de código abierto y desarrollado en Go.

Su principal característica es la **simplicidad**: permite configurar un sitio web con muy pocas líneas de configuración y ofrece **HTTPS automático** sin intervención manual.

En entornos cloud, Caddy es ideal para:

- Servir **aplicaciones web y APIs** con certificados gestionados automáticamente.
 - Actuar como **proxy inverso** o balanceador.
 - Gestionar **contenedores Docker** o microservicios con HTTPS sin esfuerzo.
- ◆ A diferencia de Apache y Nginx, Caddy obtiene certificados de **Let's Encrypt** por defecto (aunque aquí usaremos un entorno local)

4.2 Instalar dependencias de repositorios externos

Caddy no suele venir preinstalado en los repositorios básicos de Ubuntu, pero puedes instalarlo fácilmente con este comando oficial:

sudo apt update

sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl

```

kike@Double-KK:/mnt/c/Windows/System32$ sudo apt update
Get:1 https://download.docker.com/linux/ubuntu noble InRelease [48.5 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [33.1 kB]
Get:4 https://dl.cloudsmith.io/public/caddy/stable/deb/ubuntu any-version InRelease [14.8 kB]
Hit:5 http://archive.ubuntu.com/ubuntu noble InRelease
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1222 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [204 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [8968 B]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [892 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [197 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.2 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [18.3 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:16 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1498 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.3 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1496 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [301 kB]
Get:23 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:24 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.3 kB]
Get:25 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2890 kB]
Get:26 http://archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [470 kB]
Get:27 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:28 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 c-n-f Metadata [516 B]
Get:29 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [30.3 kB]
Get:30 http://archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [5564 B]
Get:31 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:32 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [484 B]
Get:33 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7140 B]
Get:34 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.0 kB]
Get:35 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:36 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Fetched 9603 kB in 3s (3339 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
15 packages can be upgraded. Run 'apt list --upgradable' to see them.
kike@Double-KK:/mnt/c/Windows/System32$ sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
debian-keyring is already the newest version (2023.12.24).
debian-archive-keyring is already the newest version (2023.4ubuntu1).
apt-transport-https is already the newest version (2.8.3).
curl is already the newest version (8.5.0-2ubuntu10.6).
The following package was automatically installed and is no longer required:
  libl1vm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.

```

Qué hace: instala llaveros y transporte HTTPS necesarios para añadir repos externos con firma GPG.

Por qué: Caddy se distribuye desde un repo propio (Cloudsmith). Sin estas piezas, apt no confiará en el repo ni podrá descargar paquetes firmados.

4.3 Añadir el repositorio oficial de Caddy

```

curl -sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' \
| sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg

```

```

Mike@Double-KK:/mnt/c/Windows/System32$ curl -lsLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' \
| sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG V2

mQINBEBtB+quEBEAC13/YkFekflvauEASL+neZjYwlyt57Dv5dRmUP04zKxylLG
d/9JawluFuHuyU4em7940S2w8kBiimLgxMqyGP5+RQnggVZhjYIXoqkkh0G8v
purq+5B+VNFy0LwhLwLuJCDotp4bPqzTc5T4QOItFK8s7F2zJzokyAPD1782
pGRH8zRpzbrp7J9tUw10HFfRR+46RF9b7yQFN9reaxsYMKIAVQKcnDd
SeBFkFrNwBwGr5MsnsQ0g9A1tsW722Jis89YevBans6n0FwiesxUqUrNvpTlkL
0QPH1K7Aqg1XGuh5scI.Fc6LoqxvboPh9BhsNxSn9koou4toxDj6ANdulGcv+Vq+
QhPFSxFsp_vTqjzbqjcsUTVZQhRNP8D+vjkzJFfcq/tAwu4gggnQ9E9KEEQL
MXnsnZNS3YmVnPBMgB0A851xka/0e89WckTnjt+jlpAkhoos5Zurdotwlv1
rY1k1YXCMQChhd3BwC5REl7b0tp+gBaOxQMyYRryKuitrpqdV53FVu/re/x
xxvXuTyIzTQ16y9a3a5Beg3R3NxV7LuGuWp1y>ZyDnHo+Xa/HtQm153h+oU1
s3ntiH016jTgycHnsUfFccMLYanfbm+2n4Hmkz197vs7xvc1c1EXNMv6wRAQAB
tcPdyWRkePSBXZ1igU2yvdmYdTxzB29yMN9QGhZG5z2vYdmVlyLnwvb76AJqE
EwEKA4B1Al+bequeGwCdwKhAxUkCAIEaQ1XgAnAMAgECQoEAcgkQFvtcepWjxR
C9/QHgdyoV1r3cUip3v1Mf06w4F0dCwLmlgbdvweZ5NCrttb8eSh+MP592gOp
UwvCqBnzb92vB2wby1ly1p0f4tCChwkyOT1doGcy1Xy1Pk1k9iep9WnsIa
3oG/wCsgxwvYzzwmw0opBdrNf6pAygt2GNCxqhb8mNyPdaReu3t2Lz6qe4obHtX
h1636f5dV5d6052dnhNPFvjopegCavTntCJcgonltc621+Qxptjybv3LSL674
2VnxcoLvtJxG6Gd3yjV15WPEZPmhzjn1PEZtgOrvL-o+Fyu7V5ChqTfy6mg
898x2L4a05B2r+Yz2U+Y5Qa6GwBGI11z1NyDwXtUnijU/GsfUwng0A0P5rv8af
2ta3yw3bs/v2nraNQoYMrB1tB0zQkg15mZD7znHpyBfJ1H6ew3/7ft3w80pti1cu6
87Uzh128yoF5NE+85V3z7spH/ZxFu2A2pEs01ahjDzP96w4u0HeSds6tbr301C
EccF0M7X9hMnBdjaNhnBqGzjJdq0zKMMzRvYz9Fkizfxw+q40atvnxnfH
p4v0B8mDL3mcx0j385Usps/hctyZ03P0hCeIAr9kvumc5Psapjqw1lbahsxa
Fxohi3lAiL1+gl7OBV1TmRz7eb1Ngf+C8Vyz1UpbHgys25Ag@Ex+uuckAEQky
h0nBz8163StGvg1R1Gb/p59c6c+yQb3d7qzQoII/61W9yCx1iQrzSb32dQsdp
a375PtG1E5p7id4nFwex2C4fFn6PwId01bwOnrcu0v6HgkjcJaFqjwNa/EvF
DugeKd01wGnN1lmkM0pGpu1JwPfSw5etw+V83BvS9Hkb/43HNkm23zcV+Rnb
MsFERSFRAYYHs+T1yTzpM05Us1z1k4zDvBzR7iyzXnIaE0dpxe8itFyF8zxze
qDsUefarr+485usNttXt0ctBxKewrH1QsCos7Hr6dD1j332cTX7T7Sbqzouq9B8
oxah1L2i0kVbWldqRyZAvrtgjKwIyUteIpguzAppfsBv6I1j/W5G5jw+HtEUscr
6r1Ciz9agGCK15N3T4vghlRyQ2GpPr2KNNsUvOjgjnc9BEPoRqj876x9077id
Qubda+QahHmLw/WS151JRLViwsCLT2wZpmheitzx6U610Et1suso7In4QsdaJ
10mP+kHnLyikGw09Fv7p2HuvP2yLz5nSIA096C3k4w4z/ykhco/GhZcb/MDkPyK
c18jUDka+Dk88xvq/AsRh+r+iV5I3n/H1kdygxKsKcyAfU1xuJwUkosn7u7bxoz
2IV5Wlfjy7s0mgk7fbAc73hUnCdg6cbWiQoFABEAAgGBHIEGAEKACWIQR1
dgRx7eogF861yHwv215yblqNAUCXuak1Ba4gUzCwBgaJACRAVm215yblqNMf0
1AQAZQaoAHRYHhC9c0mIas0pEmy+6u+h1Hwmh2BQzf765y9yaAoEJkuh+b1wmhZ
ZIIP/ZF24Ce9ve/sR66oZPr/EMax08/8tmtwAC1P84tCzryTR09d02nkzLW1QvT
rLlj10u03C1ClgHR15LEjTgeDsfrvClgs48fKAenB1hLgtzAqlQd16s1fg7t5h
dzQd9CrF6xL7TSBjsQaqzaeXpus9tEdLebh8ojo1QaymW4Bko+ymr1zpj543Hx
riBhIn/H0s4TeQwadMzL5MFyLwFvUmkoacjix92Xi1UVyphk22iSa0upsz
vseuhiytwBly99dsRwQy2BzDp/cTwpnD1BshZCz2Btyu06XgnlwUzzhUuNkC
ZK24XkPRTVgyu951YzBpnqMfW7yFwtRvLvbwv6B5dvc7v92NsTktETE74J8h
GhwStBtdgKy/1CjRGN82.2umGcxPvnU/oCo5qWx6J7Tp2bam1nRwZf1UvNjg/hwyjt
2T.1Mk1xmDwd8CvDyJ085p7jzN2G1Xf0Vhcyj/Ftm4sIwPqnr-RzTzB4xTAz1push3
Eoys+241Glg57P6zq4Cwxtnm3Z2uQh0yQlly0A5g0OZAdb8hCku6s1If+2+grtNO
F45xksAaJpV75Vh4zK9CtBbYHnA/jUp/f092x-TmmnHwBPD+U20ber/Zf50
45MzQn2oQk8uSwiIxLg9CkNwIvRFLHUMtFsAfa+qL9+1BmWf/R2Yt/Gop4n1
fJfdSMIBXgvN/212rTw0NDU3c1jvRSWsq04fjyRcux17dM/f8YBPhnGxQ2U7-709
f7L7Fgky/Vgj9QnRz6ZC6B3GPhujj105mW+1QkyehPygx1/Hu03q3w/PhBfYrYh
r5nR45NQmUsJasFrPh8yeZt2CzEwCfekzO12GSwkE5y0e9yv12RvgCgiChJ+xks
7E/L6ErPjc01zzch15d4XzK1txs9N5j5Xctd9k1F68ZCQzEj9zJ2u1tEiyCg
16wQkB34uEpkbzgZ1hB2LwF84chCa4983k0av11tKsN6dN4qU1Ja1Hj-hm7t
t1Cw7974vBwErnhSpJ0bJqsVxQ09sCdo1v2sRqD5tQfEg3AfowGhtNgxbfo/
wPlPmSpzZoA1arFx60F9B68RyLhp5d/njeawpsPAfF1yvBos0ntrh7Qmbweyf
S7fG/1baE8nsP2Dn3Dt-n6DzeUyCvKhNKKgbzYotOM1LnRHePfQwMp0o8teo
0v7r1erVgnM9cgu2gtMv9Yot5Mk-fnbDgAfR8BLPU/Zsecdk1fLnlh6/prx
Sw95n17h/FpSzbCZM5Mw1DfetXzm1TTf1MMmw8v03DwTkA9hdqnk1h4yPhQoPMD
A5zVwuxbvh6GhazJHMrU1i6w8rjmo5r
=e4f1
-----END PGP PUBLIC KEY BLOCK-----
-bash: syntax error near unexpected token `|'
Mike@Double-KK:/mnt/c/Windows/System32$
```

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' \
```

```
| sudo tee /etc/apt/sources.list.d/caddy-stable.list
```

```
[root@Double-KK: /]# curl -sLf '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

deb [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg] https://dl.cloudsmith.io/public/caddy/stable/deb/debian any-version main
deb-src [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg] https://dl.cloudsmith.io/public/caddy/stable/deb/debian any-version main
[bash]: syntax error near unexpected token `|'
[kite@Double-KK: /]#
```

Qué hace:

- Descarga la **clave GPG** del repo y la guarda en /usr/share/keyrings/... (ruta estándar para keyrings).
 - Crea un archivo .list con la **fuente APT** de Caddy.
- Por qué:** APT solo instala paquetes de repositorios **confiables y firmados**. Este paso establece esa confianza.

4.3 Instalar Caddy

sudo apt update

sudo apt install -y caddy

```
l11@Double-KK:/mnt/c/Windows/System32$ sudo apt update
E: Invalid operating update
l11@Double-KK:/mnt/c/Windows/System32$ sudo apt update
Ign:1 https://dl.cloudsmith.io/public/caddy/stable/deb/ubuntu any-version InRelease
Ign:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:3 https://download.docker.com/linux/ubuntu noble InRelease
Hit:4 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:1 https://dl.cloudsmith.io/public/caddy/stable/deb/ubuntu any-version InRelease [14.4 kB]
Hit:3 https://download.docker.com/linux/ubuntu noble InRelease
Fetched 14.4 kB in 21s (690 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
15 packages can be upgraded. Run 'apt list --upgradable' to see them.
l11@Double-KK:/mnt/c/Windows/System32$ sudo apt install -y caddy
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
caddy is already the newest version (2.10.2).
The following package was automatically installed and is no longer required:
 liblomm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
l11@Double-KK:/mnt/c/Windows/System32$ systemctl status caddy
● caddy.service - Caddy
   Loaded: loaded (/usr/lib/systemd/system/caddy.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-10-19 10:29:54 CEST; 6min ago
       Docs: https://cadvisorserver.com/docs/
   Main PID: 164 (caddy)
     Tasks: 15 (limit: 37958)
    Memory: 49.8M (peak: 51.0M)
      CPU: 121ms
     CGroup: /system.slice/caddy.service
           └─164 /usr/bin/caddy run --environ --config /etc/caddy/Caddyfile

Oct 19 10:29:54 Double-KK caddy[164]: {"level":"info","ts":1760862594.2268076,"logger":"admin","msg":"admin endpoint started","address":"localhost:2019","enforce_origin":false,"origins":["//localhost:2019"]}
Oct 19 10:29:54 Double-KK caddy[164]: {"level":"info","ts":1760862594.2275769,"logger":"tlscache_maintenance","msg":"started background certificate maintenance","cache":0xc00071e600}
Oct 19 10:29:54 Double-KK caddy[164]: {"level":"warning","ts":1760862594.228285,"logger":"http","msg":"HTTP/2 skipped because it requires TLS","network":"tcp","addr":":8082"}
Oct 19 10:29:54 Double-KK caddy[164]: {"level":"warning","ts":1760862594.2283068,"logger":"http","msg":"HTTP/3 skipped because it requires TLS","network":"tcp","addr":":8082"}
Oct 19 10:29:54 Double-KK caddy[164]: {"level":"info","ts":1760862594.2283115,"logger":"http_log","msg":"server running","name":"srv0","protocols":["h1","h2","h3"]}}
Oct 19 10:29:54 Double-KK caddy[164]: {"level":"info","ts":1760862594.2337353,"msg":"autosaved config (load with --resume flag)","file":"/var/lib/caddy/.config/caddy/autosave.json"}
Oct 19 10:29:54 Double-KK caddy[164]: {"level":"info","ts":1760862594.2338285,"msg":"serving initial configuration"}
Oct 19 10:29:54 Double-KK systemd[1]: Started caddy.service - Caddy.
```

4.4 Preparar el directorio web de Caddy

sudo mkdir -p /var/www/caddy

Qué hace: crea la carpeta raíz donde serviremos archivos.

Por qué: separar el **document root** de otros servidores (Apache/Nginx) evita conflictos y te organiza el entorno multi-servidor de la práctica.

4.5 Crear contenido de prueba (Markdown)

```
echo "# Bienvenido a Caddy" | sudo tee /var/www/caddy/README.md
echo "" | sudo tee -a /var/www/caddy/README.md
echo "Este servidor está funcionando correctamente." | sudo tee -a
/var/www/caddy/README.md
echo "" | sudo tee -a /var/www/caddy/README.md
echo "## Características" | sudo tee -a /var/www/caddy/README.md
echo "- Servidor moderno" | sudo tee -a /var/www/caddy/README.md
echo "- HTTPS automático" | sudo tee -a /var/www/caddy/README.md
echo "- Fácil configuración" | sudo tee -a /var/www/caddy/README.md
```

Qué hace: genera un README .md con contenido visible vía HTTP.

Por qué: comprobarás que Caddy sirve **texto plano** para .md según nuestra regla (ver Caddyfile). Es parte de los entregables.

4.6 Añadir una imagen de prueba

```
curl -o /tmp/test-image.jpg https://www.python.org/static/apple-touch-icon-144x144-precomposed.png
```

```
sudo mv /tmp/test-image.jpg /var/www/caddy/test.jpg
```

```
kike@Double-KK:/mnt/c/Windows/System32$ curl -o /tmp/test-image.jpg "https://www.python.org/static/apple-touch-icon-144x144-precomposed.png"
% Total    % Received % Xferd  Average Speed   Time     Time   Current
          Dload  Upload Total   Spent   Left  Speed
100  7382  100  7382    0      0  2377      0:00:03  0:00:03  ---:---  2376
kike@Double-KK:/mnt/c/Windows/System32$ sudo mv /tmp/test-image.jpg /var/www/caddy/test.jpg
kike@Double-KK:/mnt/c/Windows/System32$
```

Qué hace: descarga una imagen y la mueve al document root.

Por qué: así validas que Caddy sirve archivos binarios/estáticos (no solo texto). En WSL puede que necesites conexión de red y DNS funcionando.

4.7 Configurar Caddyfile

Entramos en **sudo nano /etc/caddy/Caddyfile** y editamos.

```
kike@Double-KK: /mnt/c/Windows/System32
GNU nano 7.2
# The Caddyfile is an easy way to configure your Caddy web server.
#
# Unless the file starts with a global options block, the first
# uncommented line is always the address of your site.
#
# To use your own domain name (with automatic HTTPS), first make
# sure your domain's A/AAAA DNS records are properly pointed to
# this machine's public IP, then replace ":80" below with your
# domain name.

:8082 {
    # Set this path to your site's directory.
    root * /var/www/caddy

    # Enable the static file server.
    file_server browse
    @markdown path *.md
    header @markdown Content-Type text/plain
}

# Another common task is to set up a reverse proxy:
# reverse_proxy localhost:8080

# Or serve a PHP site through php-fpm:
# php_fastcgi localhost:9000

# Refer to the Caddy docs for more information:
# https://caddyserver.com/docs/caddyfile
```

Qué hace cada directiva:

- :8082 → **sitio** que escucha en el puerto 8082 en todas las interfaces.
- root * /var/www/caddy → establece el **document root** para todas las rutas (*).
- file_server browse → activa el **servidor de archivos** e incluye **listado de directorios** (índice navegable).
- @markdown path *.md → **matcher** que selecciona peticiones cuyo path termina en .md.
- header @markdown Content-Type text/plain → para los .md, fuerza el header **Content-Type** a text/plain (para ver el Markdown como texto “sin procesar”).

Por qué: reproduce exactamente los objetivos de la práctica: navegar por directorios, servir Markdown como texto, y comprobar estáticos en un puerto dedicado.

Posteriormente reiniciamos caddy **sudo systemctl restart caddy**

```
[root@Double-KK ~]# sudo systemctl status caddy
● caddy.service - Caddy
   Loaded: loaded (/usr/lib/systemd/system/caddy.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-10-19 10:42:27 CEST; 14s ago
       Docs: https://caddyservice.com/docs/
   Main PID: 12 (lしだり 37958)
     Memory: 14.8M (peak: 15.3M)
        CPU: 0.000 CPU(s) used
      CGroup: /system.slice/caddy.service
              └─12994 /usr/bin/caddy run --environ --config /etc/caddy/Caddyfile

Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.294645", "logger": "admin", "msg": "admin endpoint started", "address": "localhost:2019", "enforce_min": false, "origins": ["//localhost:2019", "//::1:2019", "//127.0.0.1:2019"]}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.294645", "logger": "tlscache.maintenance", "msg": "started background certificate maintenance", "cache": "0x6000789600"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "warn", "ts": "1708863347.2918258", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "::8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.2918258", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "127.0.0.1:8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "127.0.0.1:8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "::8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "127.0.0.1:8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "::8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "127.0.0.1:8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "::8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "127.0.0.1:8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "::8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "127.0.0.1:8082"}
Oct 19 10:42:27 Double-KK caddy[12994]: {"level": "info", "ts": "1708863347.291862", "logger": "http", "msg": "HTTP/2 skipped because it requires TLS", "network": "tcp", "addr": "::8082"}  
[root@Double-KK ~]
```

```
kike@Double-KK:/mnt/c/Windows/System32$ curl http://localhost:8082/README.md
# Bienvenido a Caddy
```

Este servidor está funcionando correctamente.

```
## Características
- Servidor moderno
- HTTPS automático
- Fácil configuración
```

```
kike@Double-KK:/mnt/c/Windows/System32$
```



4.9 Comprobación final de coexistencia

Verificamos que los tres servidores están activos y funcionando en sus respectivos puertos:

Conclusión del paso 4

En este punto, el entorno de servidores queda completamente configurado:

Servidor	Puerto(s)	Función principal
Apache + PHP	8080 (HTTP) / 8443 (HTTPS)	Servidor clásico con soporte dinámico y cifrado SSL/TLS.
Nginx	8081	Servidor ligero, ideal para contenido estático y balanceo de carga.
Caddy	8082	Servidor moderno con configuración minimalista y HTTPS automático.

 Ahora los tres servidores **coexisten en WSL sin conflictos**, ofreciendo una visión práctica y profesional de la arquitectura web moderna basada en servicios distribuidos.

Para terminar el trabajo adjunto capturas de los servidores funcionando todos a la vez en sus diferentes puertos!!!



