

4. Configurar un contenedor para servir una aplicación PHP + Nginx

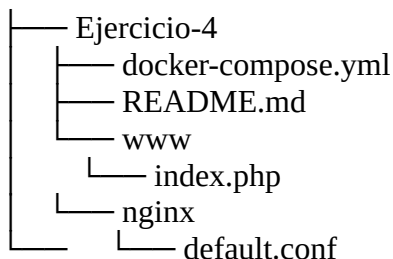
Para crear una aplicación web en PHP servida por Nginx, utilizando Docker y Docker Compose para levantar dos contenedores:

- Uno con PHP-FPM
- Otro con Nginx, configurado para procesar archivos PHP.

Vamos a realizar los siguientes pasos:

1. Estructura del ejercicio

Nuestro ejercicio va a tener la siguiente estructura de archivos y carpetas:



- El archivo docker-compose.yml va a definir los servicios PHP y Nginx y la red.
- El archivo index.php va a estar situado dentro de la carpeta www y va a comprobar que PHP funciona.
- El archivo default.conf va a estar situado dentro de la carpeta nginx y es el archivo de configuración de Nginx.

2. Creamos nuestro archivo docker-compose.yml en la version 3.8

```
1  version: "3.8"
   ▶ Run All Services
2  services:
   ▶ Run Service
3    nginx:
4      image: nginx:latest
5      container_name: nginx
6      ports:
7        - "8080:80"
8      volumes:
9        - ./www:/var/www/html
10     - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
11     depends_on:
12       - php
13     networks:
14       - network
15
   ▶ Run Service
16   php:
17     image: php:8.1-fpm
18     container_name: php
19     volumes:
20       - ./www:/var/www/html
21     networks:
22       - network
23
24   networks:
25     network:
26       name: network
```

Vamos a ir explicando paso a paso el código del archivo:

- Vamos a ejecutar 2 servicios: nginx y php.
 - **Nginx:**
 - Usa la imagen más reciente de nginx.
 - El nombre del contenedor es nginx.
 - Se expone el puerto 80 del contenedor al puerto 8080 de la máquina local.
 - Usa 2 volúmenes:
 - `./www:/var/www/html`: monta la carpeta local `www` (donde está el archivo `index.php`) dentro del contenedor nginx en la ruta `/var/www/html` y así el servidor puede usar `index.php`.
 - `./nginx/default.conf:/etc/nginx/conf.d/default.conf`: monta el archivo de configuración de nginx.
 - Depende del servicio php, es decir, el contenedor php se levanta antes que el de nginx.
 - Esta conectada a una red llamada `network` que permite que todos los contenedores conectados a esa misma red se comuniquen entre sí.
 - **PHP:**
 - Usa la imagen en la versión 8.1 de PHP-FPM.
 - El nombre del contenedor es php.
 - Usa un volumen `./www:/var/www/html`: monta la carpeta local `www` (donde está el archivo `index.php`) dentro del contenedor php en la ruta `/var/www/html` y así el servidor puede usar `index.php`.
 - Esta conectada a una red llamada `network` que permite que todos los contenedores conectados a esa misma red se comuniquen entre sí.
- Definimos la red con el nombre `network`, dónde se conectarán todos los contenedores entre sí.

3. Creamos el archivo `index.php` dentro de la carpeta `www`

Es un archivo de prueba para verificar que PHP está bien configurado y el servidor funciona de manera correcta.

```
Sprint-1 > Ejercicio-4 > www > index.php
1  <?php
2  phpinfo();
3  ?>
```

El código ejecuta la función `phpinfo()`, la cuál nos va a mostrar al abrir <http://localhost:8080> información de PHP como la versión, la configuración del servidor, etc.

4. Creamos el archivo `default.conf` dentro de la carpeta `nginx`

Es un archivo de configuración de Nginx.

Sprint-1 > Ejercicio-4 > nginx > default.conf

```
1  server {
2      listen 80;
3      server_name localhost;
4      root /var/www/html;
5      index index.php index.html;
6
7      location / {
8          try_files $uri $uri/ =404;
9      }
10
11     location ~ \.php$ {
12         include fastcgi_params;
13         fastcgi_pass php:9000;
14         fastcgi_param SCRIPT_FILENAME /var/www/html$fastcgi_script_name;
15     }
16 }
17
```

El código hace lo siguiente:

```
server {
    listen 80;
    server_name localhost;
    root /var/www/html;
    index index.php index.html;
```

- Escucha a el puerto 80 del servidor localhost
- Define la carpeta raíz /var/www/html
- Indica los archivos que nginx busca al acceder a la web

```
location / {
    try_files $uri $uri/ =404;
}
```

- Si no encuentra ningún archivo, muestra el error 404.

```
location ~ \.php$ {
    include fastcgi_params;
    fastcgi_pass php:9000;
    fastcgi_param SCRIPT_FILENAME /var/www/html$fastcgi_script_name;
}
```

- Carga los parámetros necesarios para la conexión con PHP
- Establece que nginx pase los archivos PHP al contenedor php, usando el puerto 9000
- Indica a PHP el archivo que tiene que ejecutar

6. Comprobamos que los servicios estén funcionando

Tras crear todos los archivos, vamos a comprobar que los servicios se puedan lanzar con el comando:

```
docker compose up -d
```

```

anabel@ubuntu:~/Sprint-1/Ejercicio-4$ docker compose up -d
WARN[0000] /home/anabel/Sprint-1/Ejercicio-4/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
 ✓ Network network      Created                                0.1s
 ✓ Container php         Started                                0.4s
 ✓ Container nginx       Started                                0.6s
anabel@ubuntu:~/Sprint-1/Ejercicio-4$

```

Tras lanzar los servicios, vamos a verificar que los contenedores estén en ejecución con el siguiente comando:

`docker compose ps`

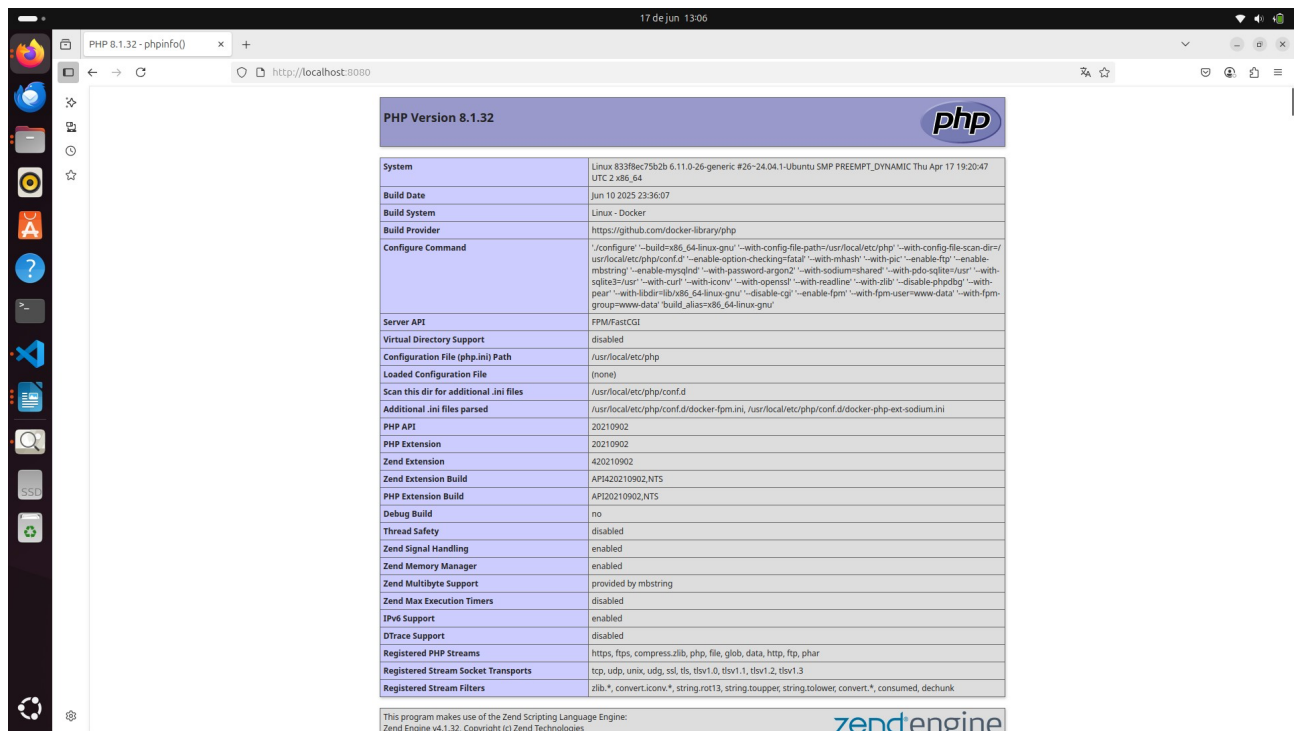
```

anabel@ubuntu:~/Sprint-1/Ejercicio-4$ docker compose ps
NAME          IMAGE          COMMAND                  SERVICE    CREATED         STATUS         PORTS
nginx         nginx:latest   "/docker-entrypoint..." nginx       49 seconds ago Up 48 seconds  0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
php           php:8.1-fpm    "docker-php-entrypoi..." php        49 seconds ago Up 48 seconds  9000/tcp
anabel@ubuntu:~/Sprint-1/Ejercicio-4$

```

7. Comprobamos que la aplicación funciona

Abrimos la dirección <http://localhost:8080> desde el navegador.



PHP Version 8.1.32	
System	Linux 833f8ec75b2b 6.11.0-26-generic #26-24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Apr 17 19:20:47 UTC 2 x86_64
Build Date	Jun 10 2025 23:36:07
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	"/configure" --build=x86_64-linux-gnu --with-config-file-path=/usr/local/etc/php --with-config-file-scan-dir=/usr/local/etc/php/conf.d --enable-option-checking=fatal --with-mhash --with-pic --enable-ftp --enable-mbstring --enable-mysqlnd --with-password-argon2 --with-sodium=shared --with-pdo-sqlite=sqlite --with-sqlite3=/usr --with-curl --with-iconv --with-openssl --with-readline --with-zlib --disable-phpdbg --with-pear --with-libdir=lib/x86_64-linux-gnu --disable-cgi --enable-fpm --with-fpm-user=www-data --with-fpm-group=www-data --build_alias=x86_64-linux-gnu
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-fpm.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902.NTS
PHP Extension Build	API20210902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
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.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v4.1.32. Copyright (c) Zend Technologies

Finalmente, podemos ver que la aplicación funciona de manera correcta, y al ingresar en la dirección, nos carga la información de PHP.