

Guía paso a paso: Crear un entorno Docker para practicar PHP

Profesor: (Proporcione su nombre aquí)

Fecha: 06/10/2025

Objetivo

Montar un entorno local usando Docker y docker-compose para que los alumnos practiquen ejercicios de sintaxis y lógica en PHP. No se realizará conexión a bases de datos; el objetivo es practicar código PHP en un servidor web Apache con PHP instalado.

1) Prerrequisitos (qué debe estar instalado y comprobarlo)

- Tener instalado Docker Desktop (Windows/Mac) o Docker Engine + docker-compose (Linux).
- En Windows: activar WSL2 y usar Docker Desktop recomendado.
- Comprobar en la terminal:

```
docker --version
# (si su Docker soporta el subcomando integrado)
docker compose version
o bien (sistemas antiguos):
docker-compose --version
```

Si estos comandos muestran versión, Docker está listo. Si no, instalar Docker antes de continuar.

2) Estructura recomendada del proyecto

Cree una carpeta para el proyecto y coloque los dos archivos que se le han dado allí: Dockerfile y docker-compose.yml. Además cree una carpeta para el código PHP que editarán los alumnos (por ejemplo: src/ o www/).

```
mi-proyecto/
■ ■ Dockerfile
■ ■ docker-compose.yml
■ ■ src/
    ■ ■ index.php
```

3) Contenido de los archivos proporcionados

Dockerfile (tal y como se ha subido):

```
FROM php:8-apache

# Instala extensiones necesarias
RUN apt-get update && \
    apt-get install -y libpq-dev && \
    docker-php-ext-install pdo pdo_mysql pdo_pgsql && \
    rm -rf /var/lib/apt/lists/* && \
    apt-get clean

# descarga la imagen php:7.4.9, actualiza repositorios e instala librerías, borra y limpia cache

# Creamos el archivo index.php directamente dentro del contenedor
RUN echo '<?php \
$servername = "db"; \
$username   = "test"; \
$password   = "test"; \
$dbname     = "dbname"; \
\
try { \
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password); \
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); \
    $mensaje = "■ Conexión a la base de datos realizada con éxito."; \
} catch(PDOException $e) { \
    $mensaje = "■ Error de conexión: " . $e->getMessage(); \
} \
?> \
<!DOCTYPE html> \
<html lang="es"> \
<head> \
    <meta charset="UTF-8"> \
    <title>IES FUENTE SAN LUIS</title> \
    <style> \
        body { background-color: #0044cc; color: white; font-family: Arial, sans-serif; text-align: center; } \
        h1 { font-size: 3em; } \
        p { margin-top: 20px; font-size: 1.2em; } \
    </style> \
</head> \
<body> \
    <h1>IES FUENTE SAN LUIS</h1> \
    <p><?php echo $mensaje; ?></p> \
</body> \
</html>' > /var/www/html/index.php
```

docker-compose.yml (tal y como se ha subido):

```
version: "3.1"

services:
    www:
```

```

build: .
ports:
  - "8888:80"
volumes:
  - ./www:/var/www/html
depends_on:
  - db
networks:
  - lamp-network

db:
  image: mysql:5.7
  ports:
    - "3307:3306"
  environment:
    MYSQL_DATABASE: dbname
    MYSQL_USER: test
    MYSQL_PASSWORD: test
    MYSQL_ROOT_PASSWORD: test
  volumes:
    - db_data:/var/lib/mysql
  networks:
    - lamp-network

phpmyadmin:
  image: phpmyadmin/phpmyadmin
  ports:
    - 8800:80
  environment:
    PMA_HOST: db
    MYSQL_ROOT_PASSWORD: test
  depends_on:
    - db
  networks:
    - lamp-network

adminer:
  image: adminer
  ports:
    - 8080:8080
  depends_on:
    - db
  networks:
    - lamp-network

networks:
  lamp-network:

volumes:
  db_data:

```

```

# Utiliza el Dockerfile ubicado en el mismo directorio

# Puerto de nuestra máquina : puerto dentro de Docker

# Volúmenes para compartir archivos entre el host y el contenedor
# Carpeta local www mapeada al directorio /var/www/html
# El servicio www depende del servicio db

# Definición de red para conectar los servicios

# Versión específica de MySQL para garantizar compatibilidad

# Puerto de MySQL (3306)
# Variables de entorno para la configuración de MySQL
# Nombre de la base de datos
# Usuario de MySQL
# Contraseña del usuario
# Contraseña de root
# Volúmenes para la persistencia de datos
# Volumen para la base de datos MySQL

# Definición de red para conectar los servicios

# Utiliza la imagen de phpMyAdmin oficial

# Puerto de phpMyAdmin

# Dirección del servicio db de MySQL
# Contraseña de root de MySQL

# phpMyAdmin depende de que el servicio db esté listo

# Definición de red para conectar los servicios

# Imagen oficial de Adminer

# Puerto de Adminer (http://localhost:8080)

# Adminer depende de que el servicio db esté listo

# Definición de red para conectar los servicios

# Definición de una red llamada lamp-network

# Volumen para la persistencia de datos de MySQL

```

4) Explicación línea a línea del Dockerfile

FROM ... -> Indica la imagen base de PHP/Apache que se usará.

COPY ... -> Copia archivos desde la carpeta del proyecto dentro de la imagen.

WORKDIR ... -> Establece el directorio de trabajo dentro del contenedor.

RUN ... -> Ejecuta comandos durante la construcción de la imagen (instalar extensiones, cambiar permisos...).

EXPOSE ... -> Indica el puerto en el que el contenedor escuchará (documentativo).

CMD / ENTRYPOINT -> Comando que se ejecuta al iniciar el contenedor.

Nota: Si su Dockerfile contiene instrucciones específicas (instalación de extensiones PHP o módulos Apache), esas instrucciones se ejecutarán al construir la imagen y estarán listas cuando se levanten los contenedores.

5) Explicación del docker-compose.yml

docker-compose define servicios, puertos, volúmenes y redes. Las claves más comunes que verá:

- services: Lista de servicios (por ejemplo: web).
- image/build: Se puede usar una imagen existente o construir desde Dockerfile con 'build:'.
- ports: Mapea puertos del host al contenedor (host:contenedor), p. ej. 8080:80.
- volumes: Mapea carpetas para persistencia o para editar código desde el host sin reconstruir la imagen.
- restart: Política de reinicio (no necesario para prácticas sencillas).

En este proyecto queremos montar el código PHP como volumen para que los alumnos editen los archivos localmente (en su ordenador) y vean los cambios inmediatamente en el navegador sin reconstruir la imagen.

6) Pasos prácticos: crear el proyecto y levantar los servicios

1. Crear una carpeta nueva en su equipo (ej: mi-proyecto) y copiar dentro Dockerfile y docker-compose.yml.

2. Crear la carpeta para el código PHP que se mapeará en docker-compose (ej: src/).

3. Crear un archivo de prueba index.php dentro de src/ con el siguiente contenido:

```
<?php
// index.php de prueba
phpinfo();
?>
```

4. Abrir una terminal en la carpeta del proyecto y ejecutar:

```
docker compose up --build -d
# o (si su instalación usa docker-compose):
docker-compose up --build -d
```

5. Abrir el navegador y acceder a <http://localhost:PUERTO> (sustituya PUERTO por el puerto indicado en docker-compose.yml, por ejemplo 8080). Si ve la página de phpinfo(), ¡funciona!

6. Para detener y eliminar los contenedores y la red creados:

```
docker compose down
# o:
docker-compose down
```

7) Problemas comunes y soluciones rápidas

- No veo la página en <http://localhost:PUERTO>

Comprobar que el puerto mapeado en docker-compose coincide. Ejecutar 'docker compose ps' para ver los puertos.

- Cambios en PHP no se reflejan

Asegurarse de que se está usando un volumen que mapea la carpeta local al contenedor (volumes). Si no hay volumen, debe reconstruir la imagen.

- Permisos de archivos

En Linux/Mac: puede que necesite ajustar permisos (chown/chmod) para que Apache pueda leer los archivos. En Windows, comprobar que el montaje funciona correctamente.

8) Buenas prácticas y recomendaciones para las prácticas en clase

- Crear un archivo README.md en la carpeta del proyecto con las instrucciones de ejecución (comandos docker).
- Mantener el código de los alumnos dentro de subcarpetas para facilitar múltiples ejercicios.
- Para ejercicios grupales, usar nombres de puertos distintos si varios alumnos usan la misma máquina.
- Explicar la diferencia entre 'build' (construir imagen) y 'run' (arrancar contenedor).

9) Ejemplo rápido: archivo index.php base para empezar

```
<?php
// index.php - archivo base para ejercicios
// Muestra un saludo simple y la fecha
$nombre = 'Alumno';
echo "<h1>Hola, $nombre</h1>";
echo "<p>Hoy es: " . date('d/m/Y H:i:s') . "</p>";
?>
```

Anexo: Comandos Docker útiles

```
docker compose up --build -d      # Construir (si hace falta) y arrancar en segundo plano
docker compose logs -f            # Ver logs en tiempo real
docker compose ps                 # Ver contenedores del proyecto
docker compose down               # Parar y eliminar contenedores/red
docker compose exec <servicio> bash  # Abrir una shell en el contenedor (si tiene bash)
docker compose exec <servicio> sh    # Abrir shell sh si bash no existe
docker image ls                   # Listar imágenes locales
docker container ls -a            # Listar contenedores (activos y parados)
```