C:\Users\Alumno\Documents\test>mkdir proyecto-php

C:\Users\Alumno\Documents\test>cd proyecto-php

C:\Users\Alumno\Documents\test\proyecto-php>cd..

C:\Users\Alumno\Documents\test>mkdir PHP

C:\Users\Alumno\Documents\test>CD PHP

C:\Users\Alumno\Documents\test\PHP>mkdir proyect01

C:\Users\Alumno\Documents\test\PHP>cd proyect01

C:\Users\Alumno\Documents\test\PHP\proyect01>mkdir nginx php src

C:\Users\Alumno\Documents\test\PHP\proyect01>touch docker-compose.yml nginx/default.conf php/Dockerfile src/index.php

"touch" no se reconoce como un comando interno o externo,

programa o archivo por lotes ejecutable.

C:\Users\Alumno\Documents\test\PHP\proyect01>copy nul docker-compose.yml

1 archivo(s) copiado(s).

C:\Users\Alumno\Documents\test\PHP\proyect01>copy nul nginx/default.conf

1 archivo(s) copiado(s).

C:\Users\Alumno\Documents\test\PHP\proyect01>copy nul php/Dockerfile

1 archivo(s) copiado(s).

C:\Users\Alumno\Documents\test\PHP\proyect01>copy nul src/index.php

La sintaxis del comando no es correcta.

C:\Users\Alumno\Documents\test\PHP\proyect01>cd src

C:\Users\Alumno\Documents\test\PHP\proyect01\src>copy nul index.php

(podemos crear toda la estructura desde visual studio si se prefiere)

**docker-compose.yml**

services:

php:

image: php:8.2-fpm

build: ./php ***(dónde está el archivo Dockerfile)***

container\_name: docker\_php

volumes:

- ./src:/var/www/html

environment:

- MYSQL\_HOST=host.docker.internal ***(en lugar de localhost)***

- MYSQL\_USER=alumno

- MYSQL\_PASSWORD=alumno

- MYSQL\_DB=test

- MONGODB\_HOST=host.docker.internal

- MONGODB\_USER=alumno

- MONGODB\_PASSWORD=alumno

- MONGODB\_DB=test

- MONGODB\_PORT=27017

nginx: ***(Otro contenedor)***

image: nginx:latest

ports:

- "8080:80"

volumes:

- ./src:/var/www/html

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf ***(corresponde al fichero creado en la carpeta nginx)***

depends\_on:

- php

Configuración de Nginx

1. **Abre el archivo nginx/default.conf**.
2. **Pega el siguiente contenido** en el archivo default.conf:

**Configuración nginx**

server {

listen 80;

server\_name localhost;

root /var/www/html;

index index.php index.html;

location / {

try\_files $uri $uri/ =404;

}

location ~ \.php$ {

include fastcgi\_params;

fastcgi\_pass php:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME /var/www/html$fastcgi\_script\_name;

}

}

Crear el Dockerfile para PHP (si lo necesitas)

1. **Abre el archivo php/Dockerfile**.
2. **Pega el siguiente contenido** para crear el contenedor de PHP con las extensiones necesarias, de momento las libreriás PDO y MySQL:

Sería recomendado compartir bind el directorio usr/local/etc/php/conf.d para que puedas modificar el archivo php.ini desde tu máquina host.

FROM php:8.2-fpm

# Instalar extensiones de PHP necesarias (como PDO y MySQL)

RUN docker-php-ext-install pdo pdo\_mysql

# Instalar las dependencias necesarias para MongoDB

RUN apt-get update && apt-get install -y \

libmongoc-1.0-0 \

libjemalloc2 \

&& rm -rf /var/lib/apt/lists/\*

# Configuración de PHP, initializar el archivo php.ini

# Copiar el archivo php.ini-development a php.ini

RUN mv /usr/local/etc/php/php.ini-development /usr/local/etc/php/php.ini

# Instalar las extensiones de PHP necesarias

RUN pecl install mongodb \

&& docker-php-ext-enable mongodb

# Establecer la ruta de trabajo

WORKDIR /var/www/html

Crear un archivo PHP de prueba en el archivo src/index.php.

En linux:

### ¿Por qué usamos "mysql" como host?

En Docker, cuando varios servicios están definidos en el mismo archivo docker-compose.yml y están en la misma red (como sistgi-net en tu caso), cada contenedor puede comunicarse con los demás usando el nombre del servicio como nombre de host.

Así que si tu servicio de base de datos se llama:

services:

mysql:

image: mysql:8.0

Entonces desde cualquier otro contenedor (como el de PHP), puedes conectar a la base de datos usando mysql como hostname, porque Docker crea un DNS interno que resuelve automáticamente ese nombre.

🔄 ¿Y por qué no usamos localhost?

Porque dentro de un contenedor, localhost se refiere al propio contenedor, no al host de Docker ni a otro contenedor.

Ejemplo:

Si en tu contenedor PHP haces localhost, estás intentando encontrar MySQL dentro del propio contenedor PHP, donde no hay MySQL.

🖥️ ¿Y qué hay de host.docker.internal?

host.docker.internal apunta al sistema host (tu máquina física) desde un contenedor. Es útil si, por ejemplo, tu base de datos no está en Docker sino corriendo directamente en tu máquina local.

Pero en este caso, MySQL sí está en Docker, así que no necesitas apuntar a tu host real.

🧠 Resumen rápido:

Host ¿Cuándo usarlo?

mysql ✅ Cuando tu contenedor de MySQL está definido en el mismo docker-compose y en la misma red.

localhost ❌ Solo apunta al mismo contenedor. No ve otros.

host.docker.internal ✅ Solo si necesitas acceder a tu PC desde el contenedor.

**docker-compose.yml** en linux:

version: '3.8'

services:

mysql:

image: mysql:8.0

container\_name: docker-mysql

restart: unless-stopped

environment:

MYSQL\_DATABASE: 'pruebas'

MYSQL\_USER: 'alumno'

MYSQL\_PASSWORD: 'alumno'

MYSQL\_ROOT\_PASSWORD: 'administrador'

ports:

- '6606:3306'

volumes:

- mysql-data:/var/lib/mysql

networks:

- sistgi-net

mongodb:

image: mongo:6

container\_name: docker-mongodb

restart: unless-stopped

environment:

MONGO\_INITDB\_ROOT\_USERNAME: alumno

MONGO\_INITDB\_ROOT\_PASSWORD: alumno

MONGO\_INITDB\_DATABASE: pruebas

ports:

- "27017:27017"

volumes:

- mongo-data:/data/db

networks:

- sistgi-net

php:

build: ./php

container\_name: docker\_php

volumes:

- ./src:/var/www/html

environment:

- MYSQL\_HOST=mysql

- MYSQL\_USER=alumno

- MYSQL\_PASSWORD=alumno

- MYSQL\_DB=pruebas

- MONGODB\_HOST=mongodb

- MONGODB\_USER=alumno

- MONGODB\_PASSWORD=alumno

- MONGODB\_DB=pruebas

- MONGODB\_PORT=27017

depends\_on:

- mysql

- mongodb

networks:

- sistgi-net

nginx:

image: nginx:latest

container\_name: docker-nginx

ports:

- "8080:80"

volumes:

- ./src:/var/www/html

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf

depends\_on:

- php

networks:

- sistgi-net

volumes:

mysql-data:

mongo-data:

networks:

sistgi-net:

driver: bridge

**Dockerfile** en linux

*# Instalar extensiones de PHP necesarias (como PDO y MySQL)*

RUN docker-php-ext-install pdo pdo\_mysql **mysqli**

*# Instalar las dependencias necesarias para MongoDB*

RUN apt-get update && apt-get install -y \ libmongoc-1.0-0 \ libjemalloc2 \ && rm -rf /var/lib/apt/lists/\*

*# Configuración de PHP, initializar el archivo php.ini*

*# Copiar el archivo php.ini-development a php.ini*

#RUN mv /usr/local/etc/php/php.ini-development /usr/local/etc/php/php.ini

*# Instalar las extensiones de PHP necesarias*

RUN pecl install mongodb \ && docker-php-ext-enable mongodb

*# Establecer la ruta de trabajo*

WORKDIR /var/www/html

----------------------------------------

(En la carpeta donde se encuentre el docker-compose) se ejecuta:

> docker-compose down (para cerrar los contenedores)

> docker-compose up --build -d (para levantar con la configuración del docker-compose y el dockerfile)

Esto deja corriendo:

MySQL (puerto 6606)

MongoDB (puerto 27017)

PHP con extensiones necesarias

NGINX para servir contenido