

Creación de un docker-compose para el la ejecucion de PHP + MySQL y phpMyAdmin

Docker-Compose.

Es una herramienta para definir y ejecutar aplicaciones Docker multicontenedor que permite simplificar el uso de Docker a partir de archivos YAML, de esta forma es más sencillo crear contenedores que se relacionen entre sí, conectarlos, habilitar puertos, volúmenes, etc. Nos permite lanzar un solo comando para crear e iniciar todos los servicios desde su configuración (YAML), esto significa que puedes crear diferentes contenedores y al mismo tiempo diferentes servicios en cada contenedor, integrarlos a un volumen común e iniciarlos y/o apagarlos, etc. Este es un componente fundamental para poder construir aplicaciones y microservicios.

Docker-Compose funciona en todos los entornos: production, staging, development, testing.

¿Qué es un archivo YAML?

Es un fichero o archivo de configuración disponible para la herramienta de ejecución y definición de aplicaciones en contenedores denominada Docker Compose.

Comandos básicos de docker-compose

Mostrar la versión de Docker-compose

```
sudo docker-compose --version
```

Mostrar la ayuda de Docker-compose.

```
sudo docker-compose --help
```

Iniciar todo el entorno o los servicios especificados en el archivo docker-compose.yml.

```
sudo docker-compose up -d
```

Listar contenedores ejecutándose en el contexto de docker-compose.

```
sudo docker-compose ps
```

Frenar el/los contenedores.

```
sudo docker-compose down
```

La configuración que hicimos de forma manual con docker en la guía anterior la haremos en un solo archivo el docker-compose donde solo nos bastara ejecutar un solo comando y ya tendríamos nuestros servicios y contenedores creados.

Configuración del servidor.

Creamos una carpeta donde guardaremos el proyecto.

Crear nuestro archivo docker-compose.yml

Una cuestión importante que debemos tener en cuenta al trabajar con los ficheros yml es que haremos uso de tabulaciones par indentar el código vayamos escribiendo.

```
version: '3.7'
services:
  php-apache:
    container_name: my-php-apache
    image: php:7.4-apache
    volumes:
      - ./php/src:/var/www/html/
    ports:
      - 8080:80
    networks:
      - lan-networks
```

version: Seleccionar la versión de Docker que desea usar.

```
version: '3.7'
```

services: Servicios que se crearan.

```
services:  
  php-apache
```

Para el contenedor PHP-Apache, debemos especificar los siguientes.

container_name: Este es solo un nombre aleatorio como le gustaría nombrar a su contenedor PHP.

```
container_name:
```

image: Imagen base de la cual se creará el contenedor, esta es imagen oficial de PHP con apache ya integrado php:7.4-apache.

```
image:php:
```

volumes: En su directorio de trabajo en la carpeta src estarán sus archivos de código fuente php. Si tuviera que ejecutar un script PHP, ese archivo tendría que estar en ese directorio. Y el contenido de ese volumen sera copiado a la ruta por omisión del servidor apache.

```
volumes:  
  - ./src:/var/www/html/
```

ports: Asignación de puerto en su computadora local, al puerto del contenedor que funciona como servidor.

```
ports:
```

```
- 8080:80
```

networks: Creamos una red que se conectaran los demás contenedores para compartir sus variables de entorno.

```
networks
- lan-networks
```

Las aplicaciones se ejecutan dentro de contenedores, de manera independiente unas de otras. Sin embargo, estas aplicaciones necesitan comunicarse entre ellas

Dirígete al directorio de tu proyecto./src, crea un index.php.

```
<?php
echo "Hola, este es el contenedor my-php-apache";
?>
```

Ejecutamos el archivo docker-compose.yml

```
sudo docker-compose up -d
```

Eso extraerá toda la información creará y ejecutará el contenedor.

Verificar si se está ejecutando el contenedor.

```
sudo docker-compose ps
```

Name	Command	State	Ports
my-php-apache	docker-php-entrypoint apac ...	Up	0.0.0.0:8080->80/tcp,:::8080->80/tcp

Ilustración 1: contenedor my-php-apache ejecutando

Para asegurarse de que el contenedor esté configurado y ejecutando los scripts PHP, abrimos el navegador y escribimos.

<http://localhost:8080/>



Ilustración 2: Vista de la ejecución del contendor

Configuración de MySQL.

Agreguemos el servicio MySQL al archivo docker-compose.yml

```
db:
  container_name: mi-db
  image: mysql:5.7
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: prueba
    MYSQL_USER: usuario1
    MYSQL_PASSWORD: pass1234
  ports:
    - "3306:3306"
  networks
    - lan-networks
```

Para MySQL necesitamos configurar:

MYSQL_ROOT_PASSWORD=	Contraseña del usuario root.
MYSQL_DATABASE=	Nombre de la base de datos
MYSQL_USER=	Usuario para esa base de datos
MYSQL_PASSWORD=	Contraseña para el usuario1

Probemos si el contenedor funciona como esperamos.

Dirígete al archivo index.php y cambia el código por el siguiente, lo que agregamos es una conexión a la base de datos.

```
<?php
$servername = "db";
$username = "root";
$password = "root";
$dbname = "prueba";
$db = mysqli_connect($servername, $username, $password, $dbname);
if (!$db) {
    die("Fallo la conexion: " . mysqli_connect_error());
}
echo "Conexión exitosa!";
mysqli_close($db);
?>
```

Si ejecutamos nuestro archivo docker-compose.yml ambos servicios se crearán, pero si recordamos en la práctica anterior nos dio este error.

```
Fatal error: Uncaught Error: Call to undefined function mysqli_connect() in
/var/www/html/index.php:7 Stack trace: #0 {main} thrown in
/var/www/html/index.php on line 7
```

Para solucionarlo necesitamos agregar la extensión mysqli de MySQL a la imagen de php:7.4-apache para que php pueda conectarse con MySQL

En el directorio de su proyecto cree un archivo Docker, asígnele el nombre Dockerfile y agregue las siguientes configuraciones de PHP.

```
FROM php:7.4-apache
RUN apt-get update && apt-get upgrade -y
RUN docker-php-ext-install mysqli pdo pdo_mysql
```

Ahora necesitamos construir esta imagen personalizada dentro del servicio php-apache en el docker-compose.yml. php-apache también depende del servicio db para conectarse a MySQL especificando un depends_on:

Modificamos el archivo docker-compose.yml

```
version: '3.7'
services:
  php-apache:
    container_name: my-php-apache
    build:
      context: ./
      dockerfile: Dockerfile
    depends_on:
      - db
    volumes:
      - ./src:/var/www/html/
    ports:
      - 8080:80
    networks:
      - lan-networks

  db:
    container_name: mi-db
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: prueba
      MYSQL_USER: usuario1
      MYSQL_PASSWORD: pass1234
    ports:
      - "3306:3306"

    networks:
```

```
- lan-networks
networks:
  lan-networks:
```

Ejecutar `sudo docker-compose up -d` para extraer y configurar el entorno MySQL y se agregará y creará el nuevo contenedor `mi-db` que es el de `mysql`.

Recargamos la página en nuestro navegador.

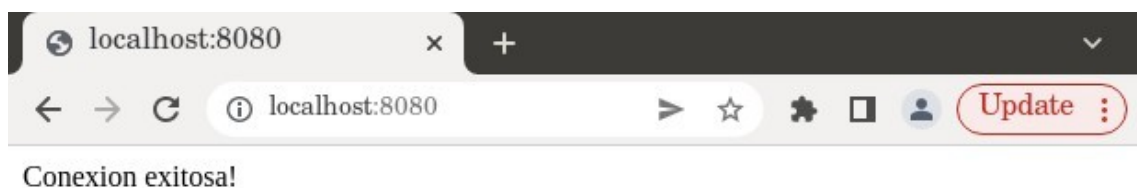


Ilustración 3: Muestra la conexión exitosa entre contenedores

El entorno PHP Apache y MySQL ahora están configurados, y puede comenzar a desarrollar su aplicación.

Configuración de phpMyAdmin.

La conexión MySQL ahora está bien. Veamos cómo podemos obtener algunos datos de una base de datos MySQL y mostrarlos en una página web utilizando scripts PHP.

La aplicación interactúa con una base de datos; probablemente querrá una interfaz para interactuar con sus datos. Agregaremos servicios PHPMyAdmin para proporcionarnos una interfaz para interactuar con la base de datos MySQL.

Agreguemos un servicio PHPMyAdmin como se muestra a continuación.

```
phpmyadmin:
  image: phpmyadmin
```



```
links:
  - db
networks:
  - lan-networks
ports:
  - 8000:80
environment:
  PMA_ARBITRARY: 1
  PMA_HOST: db
  MYSQL_ROOT_PASSWORD: root
networks:
  lan-networks:
```

PMA HOST: servidor de base de datos

MYSQL_ROOT_PASSWORD: contraseña del usuario de mysql

Detenemos los contenedores y los volvemos a ejecutar para que cree el tercer contenedor de phpMyAdmin

Abrimos una nueva pestaña del navegador y escribimos.

<http://localhost:8000/>

Para iniciar sesión en el panel Phpmyadmin, use:

Servidor: db que es el nombre del servicio de base de datos en el docker-compose.yml

El nombre puede ser el root con la contraseña de la variable
MYSQL_ROOT_PASSWORD: root

O bien el usuario y la contraseña establecidas en las variables de entorno

MYSQL_USER: usuario1

MYSQL_PASSWORD: pass1234

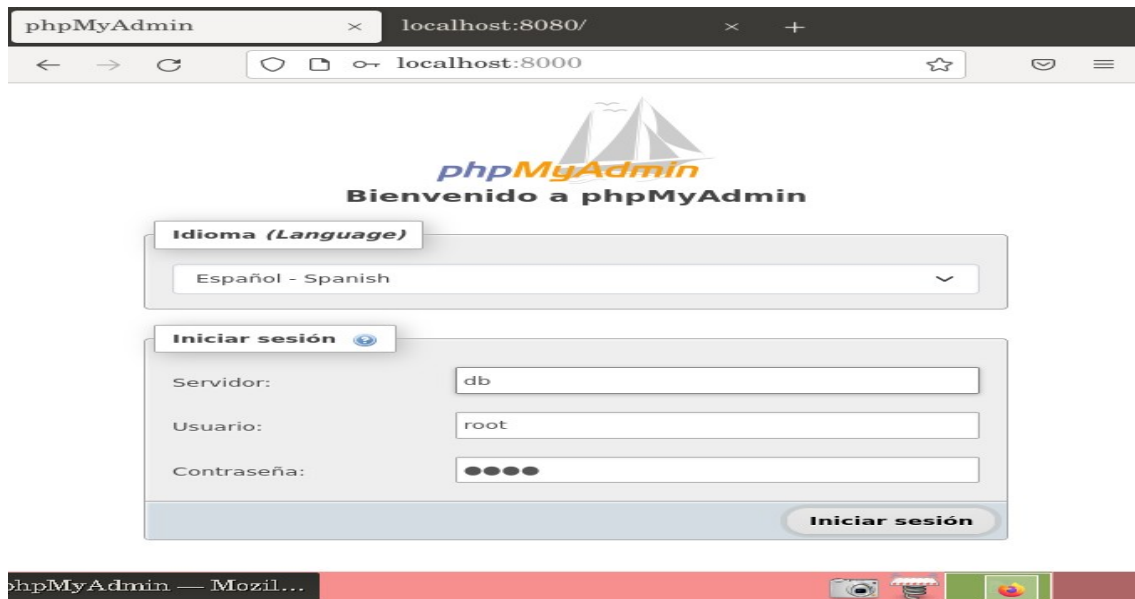


Ilustración 4: Inicio de sesión de phpMyAdmin

Ahora puede ver que la base de datos que fue definida y configurada, ya puede comenzar a interactuar con Phpmyadmin.

En nuestra base de datos “prueba” cree una tabla llamada “personas” con cuatro columnas id, nombre, correo, sexo y rellene algunos registros.

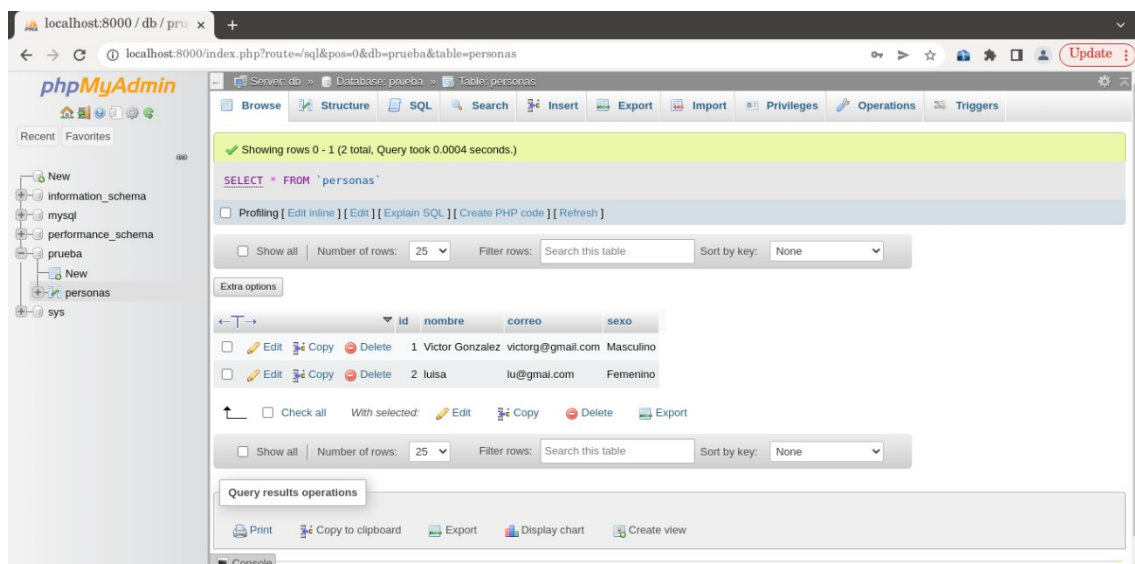


Ilustración 5: Tabla personas de la base de datos prueba

Editamos el archivo index.php

```
<?php
$db=mysqli_connect('db','root','root','prueba')
?>

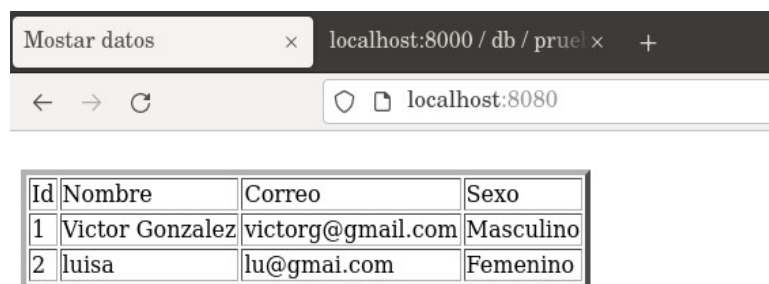
<!DOCTYPE html>
<html>
<head>
    <title>Mostar datos</title>
</head>
<body>
<br>
<table border="4px" >

<tr >
    <td>Id</td>
    <td>Nombre</td>
    <td>Correo</td>
    <td>Sexo</td>
</tr>
    <?php
    $sql="SELECT * from personas";
    $result=mysqli_query($db,$sql);
    while ($mostrar=mysqli_fetch_array($result)){
    ?>
<tr>
    <td><?php echo $mostrar['id'] ?></td>
    <td><?php echo $mostrar['nombre'] ?></td>
    <td><?php echo $mostrar['correo'] ?></td>
    <td><?php echo $mostrar['sexo'] ?></td>
</tr>
    <?php
    }
```

```
    ?>
</table>
</body>
```

Actualice para ver los resultados.

<http://localhost:8080/>



The screenshot shows a web browser window with a single tab titled 'localhost:8000 / db / prue...'. The address bar displays 'localhost:8080'. Below the browser window, a table is displayed with the following data:

Id	Nombre	Correo	Sexo
1	Victor Gonzalez	victorg@gmail.com	Masculino
2	luisa	lu@gmai.com	Femenino

Ilustración 6: Tabla prueba mostrada en el navegador