Dockerize CI4 + MySQL

1. Create ci4 project and test
2. Create "Dockerfile" in project root

Dockerfile

```
FROM php:8.2-apache

# Update and upgrade packages
RUN apt-get update && apt-get upgrade -y

# Install CodeIgniter 4 dependencies (ext-intl, ext-gd, ext-zip)
# Enable required PHP extensions
RUN apt-get install -y libicu-dev libpng-dev libzip-dev
RUN docker-php-ext-install intl gd zip

# Modify php.ini to increase upload file size to 100MB
RUN echo "upload_max_filesize = 100M" > /usr/local/etc/php/conf.d/uploads.ini
RUN echo "post_max_size = 100M" >> /usr/local/etc/php/conf.d/uploads.ini

# Install git
RUN apt-get install -y git

# Enable mod_rewrite for Apache
RUN a2enmod rewrite

# Copy the Apache configuration file
COPY 000-default.conf /etc/apache2/sites-available/000-default.conf

# Restart Apache
RUN service apache2 restart
```

3. Create apache config virtualhost file in project root

000-default.conf

```
<VirtualHost *:80>
  # Set the document root to the public directory of the website
  DocumentRoot /var/www/html/public

  <Directory /var/www/html/public>
    # Allow directory indexing and symbolic links
    Options Indexes FollowSymLinks

    # Allow .htaccess files to override Apache configuration
    AllowOverride All
```

```
      # Allow access to the directory
      Require all granted
    </Directory>

    # Define the location of the error log file
    ErrorLog ${APACHE_LOG_DIR}/error.log

    # Define the location and format of the access log file
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

4. create docker-compose.yml in project root

docker-compose.yml

```
version: '3.8'
services:
  app:
    container_name: ci-API
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
      - .:/var/www/html/
    ports:
      - 8080:80
```

5. build image and run container

run cmd in project root

```
docker-compose up --build -d
```

6. Open http://localhost:8080/ and your Dockerized CodeIgniter app should be ready

7. Create a mysql container, update the docker-compose.yml

docker-compose.yml

```
version: '3.8'
services:
  app:
    container_name: ci-API
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
```

```
   - .:/var/www/html/
  ports:
   - 8080:80
  depends_on:
   - db

 db:
  image: mysql:8.3
  restart: always
  environment:
   MYSQL_ROOT_PASSWORD: your_root_password
   MYSQL_DATABASE: your_database_name
   MYSQL_USER: your_username
   MYSQL_PASSWORD: your_password
  ports:
   - 3306:3306

 phpmyadmin:
  image: phpmyadmin/phpmyadmin
  ports:
   - 8001:80
  environment:
   - PMA_HOST=db
   - PMA_PORT=3306
```

8. CodeIgniter will connect to MySQL on Docker. This means you must have mysqli ready. In this case, go to Dockerfile and install mysqli as follows:

Dockerfile

```
FROM php:8.2-apache
# Update and upgrade packages
RUN apt-get update && apt-get upgrade -y
# Install CodeIgniter 4 dependencies (ext-intl, ext-gd, ext-zip)
RUN apt-get install -y libicu-dev libpng-dev libzip-dev
RUN docker-php-ext-install intl gd zip
# Add required mysqli PHP extensions
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
RUN docker-php-ext-install mysqli pdo pdo_mysql
# Enable mod_rewrite for Apache
RUN a2enmod rewrite
# Modify php.ini to increase the upload file size to 100MB
RUN echo "upload_max_filesize = 100M" > /usr/local/etc/php/conf.d/uploads.ini
RUN echo "post_max_size = 100M" >> /usr/local/etc/php/conf.d/uploads.ini
# Install git
RUN apt-get install -y git
```

```
# Copy the Apache configuration file
COPY 000-default.conf /etc/apache2/sites-available/000-default.conf
# Restart Apache
RUN service apache2 restart
```

9. For CodeIgniter to access MySQL, you will need to update the app\Config\Database.php file with the following variables:

app\Config\Database.php

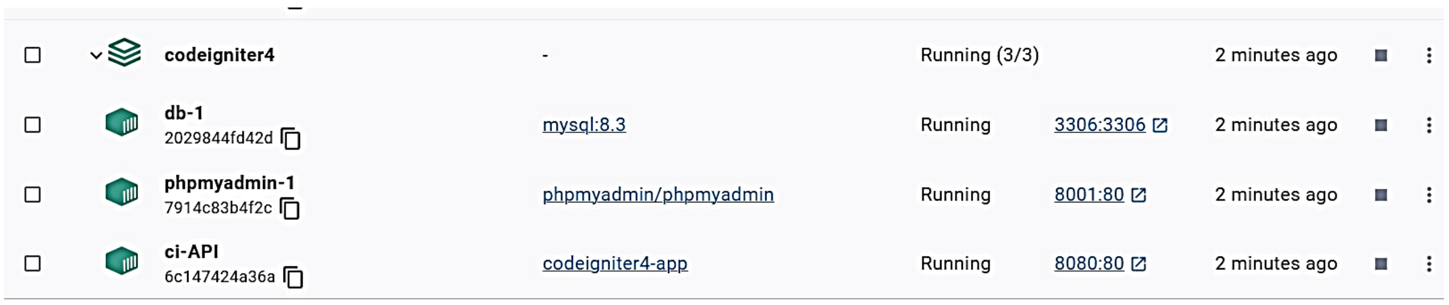
```
 public array $default = [
  'DSN'  => '',
  'hostname' => 'db',
  'username' => 'your_username',
  'password' => 'your_password',
  'database' => 'your_database_name',
  'DBDriver' => 'MySQLi',
```

10. You will then update the app\Controllers\Home.php file with MySQL Database connection as follows:

```
<?php

namespace App\Controllers;

class Home extends BaseController
{

    public function index()
    {
        // Get the database connection
        $db = \Config\Database::connect();

        // Test database connection
        if ($db->connect()) {
            echo "Database connection successful!";
        } else {
            echo "Unable to connect to the database.";
        }
    }
}
```

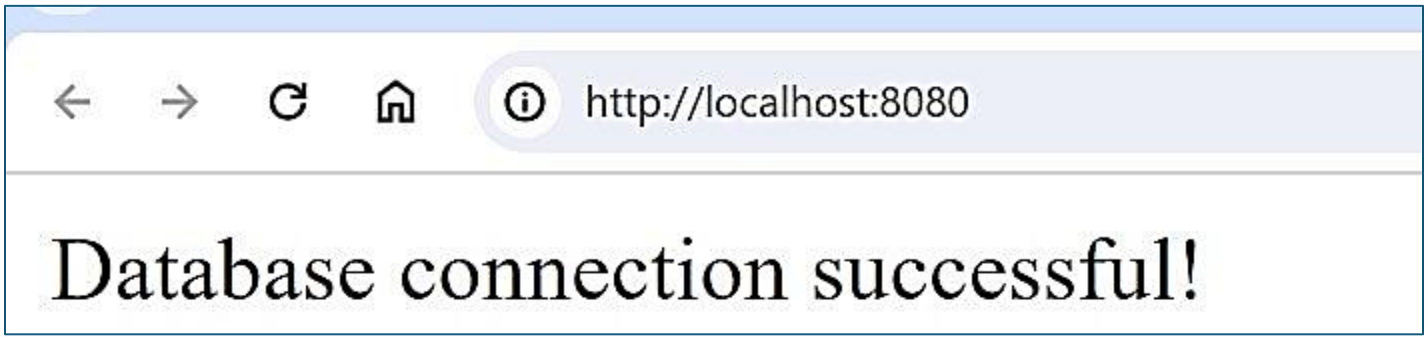
11. rerun your Docker Compose:

```
docker-compose up --build -d
```



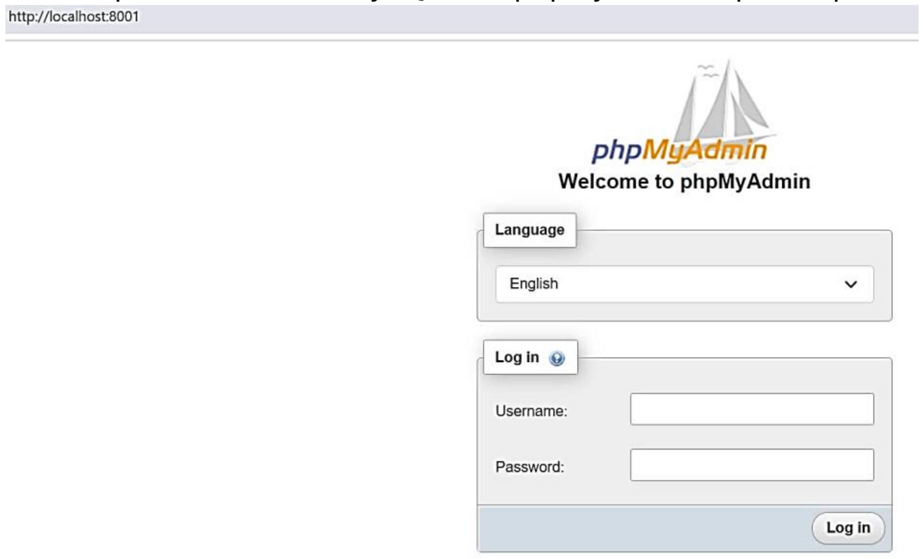
☐	codeigniter4	-	Running (3/3)		2 minutes ago	■
☐	db-1 2029844fd42d	mysql:8.3	Running	3306:3306	2 minutes ago	■
☐	phpmyadmin-1 7914c83b4f2c	phpmyadmin/phpmyadmin	Running	8001:80	2 minutes ago	■
☐	ci-API 6c147424a36a	codeigniter4-app	Running	8080:80	2 minutes ago	■

At this point, you can access your CodeIgniter app (http://localhost:8080/) and confirm if it can connect to MySQL:

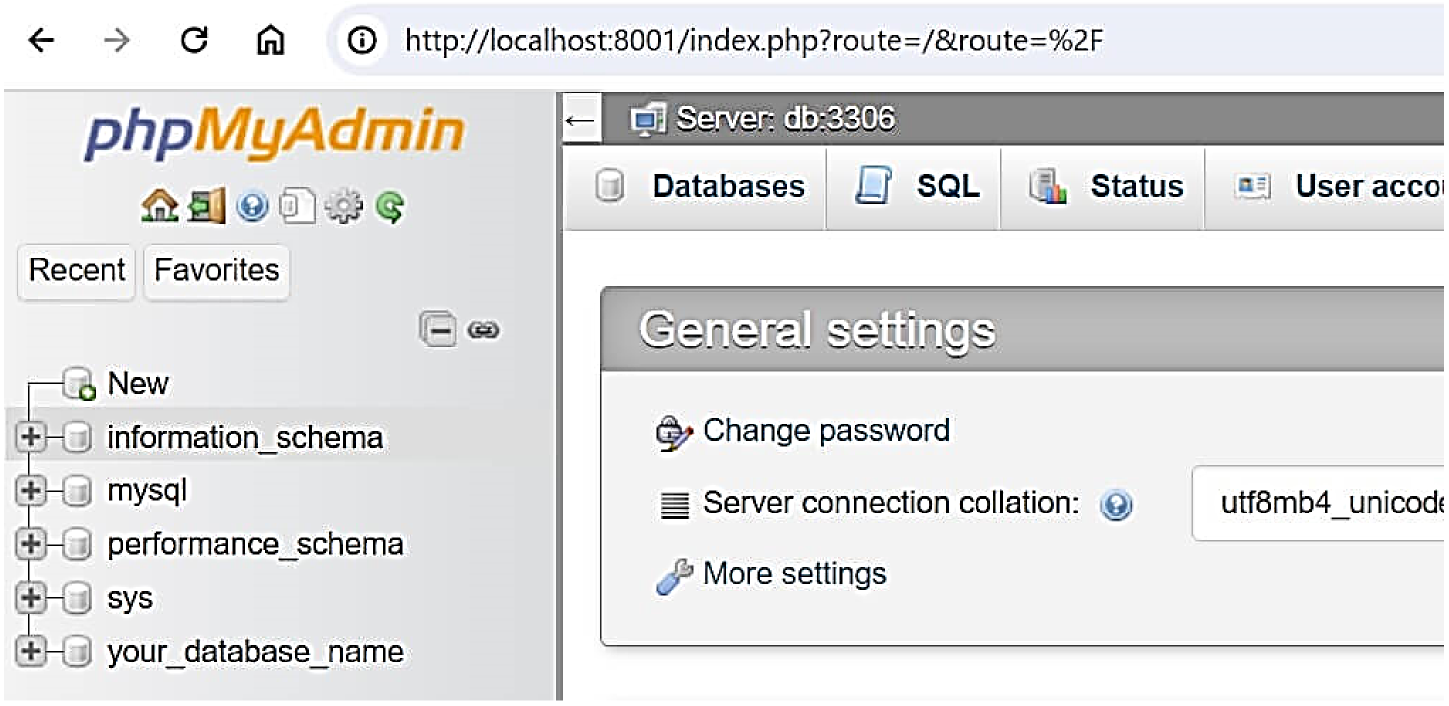


```
http://localhost:8080

Database connection successful!
```

12. You should now create your CRUD CodeIgniter app with MySQL while leveraging the Docker containers. The last part is to access MySQL with phpMyAdmin. Open http://localhost:8001/:



Use root as the username and password based on MYSQL_ROOT_PASSWORD: your_root_password. So, you add your_root_password as password and log in:



Note:

To connect containerize ci4 app to localhost mysql:

- ✓ Just use host.docker.internal instead of localhost.
- ✓ To make it work on Linux, you just need to start your Docker container passing the parameter

```
--add-host host.docker.internal:host-gateway
```

How to Share Docker Images With Others

The easiest way to share a Docker image is to push it up to a Docker registry. This functionality is fully integrated into the Docker CLI. You don't need to make any manual file transfers when using this method.

The default registry is Docker Hub. This allows you to publicly share images and gives you one private repository too. Create an account on Docker Hub, then run

```
docker login
```

*in your terminal. Supply your account details to connect the Docker CLI to your account.

Next build your image as normal, using

```
docker build
```

Add a tag that starts with your Docker Hub username:

```
docker build -t my-account/my-image:latest .
```

Then use the docker push command to push the tagged image up to Docker Hub:

```
docker push my-account/my-image:latest
```

Now your image is safely stored in Docker Hub. Other users will be able to pull it down using the docker pull or docker run commands. You're done sharing your image!