Doc4Dev

Configure a WEB site (PHP-Apache-Mysql) in 5 minutes with Docker

19 June 2022 / 6 Comments

Table of Contents

1. Introduction
2. Application Architecture
3. Writing Docker configuration
    3.1. docker-compose.yml
    3.2. Writing Dockerfiles
        3.2.1. Dockerfile Apache-PHP
        3.2.2. Dockerfile Mysql
4. Site building and testing
5. PHP-MySQL connection test
6. Conclusion

Introduction

If you are a PHP developer, you may still develop your web applications with Apache installed on your computer. If one day your production changes version of PHP, or even e MySQL, you are therefore obliged to follow by updating yourself, potentially encountering compatibility problems etc. Now imagine that you decide to add Redis caching to your application, or even a second API to be hosted on another server, or even a full-fledged mail server? You'll be forced at each iteration to install a new server, to connect it to Apache, to stuff yourself with configuration files with the shovel. An example which is also very common, is "How to make so that developers being on MAC, Linux and Windows can work on the same version, without having different configurations"?

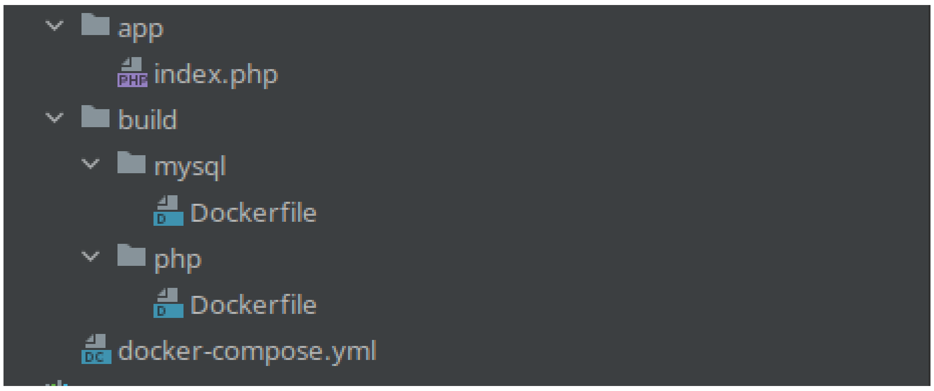
Just thinking about it will make you wrack your brains. But luckily, Docker and Docker-compose are here to help. They will allow you, using their container system, to achieve all this in record time. (If you do not know the principle, I authorize you to temporarily leave this article to find out about it, and install them at the same time).

We can nevertheless think that the configuration of Docker requires a good level of understanding of information systems, and knowing how to dig through a huge documentation to succeed in having something functional, so spending time there. It is not so. We will see that to configure a simple website (PHP-Apache-Mysql), we need a total of 5 minutes (yes, we promise!)

Application Architecture

For this tutorial, we will create a single web page, which will query a database, and display the data of a table. The site is deliberately very succinct, in order to focus on the simplicity of action of Docker.

For this, I propose the tree and the following files:



```
app
    index.php
build
    mysql
        Dockerfile
    php
        Dockerfile
docker-compose.yml
```

Tree structure of a simple web application under Docker

With only these 5 files, we will be able to build an apache server containing PHP, a MySQL server, and to communicate the two to use our website.

These 5 files will have the following role:

- app/index.php: The entry point of our website, we will write our PHP code there
- build/mysql/Dockerfile: Docker configuration file that describes how to install MySQL

- build/php/Dockerfile: Docker configuration file that describes how to install Apache and PHP
- docker-compose.yml: Docker-compose configuration file, which when executed will launch the containers and run our architecture.

Writing Docker configuration

As you will have understood, all the configuration will be done within the 3 files concerning Docker. We will have two containers: One for PHP and Apache, and another for MySQL. Let's see how to proceed:

docker-compose.yml

docker-compose.yml

```
version: "3.9"
services:
  php-apache:
    ports:
      - "80:80"
    build: './build/php'
    volumes:
      - ./app:/var/www/html
  mysql:
    ports:
      - "3306:3306"
    build: './build/mysql'
    environment:
      MYSQL_ROOT_PASSWORD: "super-secret-password"
      MYSQL_DATABASE: "my-wonderful-website"
    volumes:
      - dbData:/var/lib/mysql
volumes:
  app:
  dbData:
```

This is what our docker-compose.yml will look like, and it won't move. There isn't much, is there? Let's break down its content anyway:

We find two services, which represent our two containers:

- **php-apache,** will be accessible from port 80 of our machine which is the default HTTP port. The site will therefore be directly accessible from localhost. It declares that all your code, available in the "app" folder, will then be mapped to the root of the apache server, which is “/var/www/html”. Finally, the build part will define the Apache and PHP installation process, which will be in the Dockerfile of the “build/php” directory, which we will detail just after.

- **mysql**, will be accessible from port 3306 of our machine which is the default port of MySQL. The site will therefore be directly accessible from localhost. The build part will define the MySQL installation process, which will be in the Dockerfile of the build/mysql directory. Two variables are declared and will be executed during the installation of MySQL. MYSQL_ROOT_PASSWORD sets the root account password, MYSQL_DATABASE the default database. It will be created if it does not exist when the container is launched. A volume is also created in order to make the data persistent, and thus not to lose the database data once the container is shut down.

Writing Dockerfiles

This established, we now need to write the two Dockerfiles, which will detail the installation procedures for the two servers. Are you ready ? Watch out, it’s going to be quick:

Dockerfile Apache-PHP

build/php/Dockerfile

```
FROM php:8.1-apache

RUN apt-get update && \
    docker-php-ext-install mysqli pdo pdo_mysql
```

What are we doing here?

We will simply tell Docker to fetch the existing php-apache image, which therefore contains apache and PHP at version 8.1.

Finally, we install the MySqli, PDO and PDO_MySQL extensions in PHP, which will be very useful for us to connect to our database.

Dockerfile Mysql

build/mysql/Dockerfile

```
FROM mysql:latest
USER root
RUN chmod 755 /var/lib/mysql
```

Here the same operation, we ask Docker to retrieve the latest version of MySQL from the existing image. Then, we make a change of rights on the “/var/lib/mysql” directory of the container that will be created, in order to allow PHP to connect to it.

The configuration is now finished (and yes, already!). We can still, before launching the build, add a simple index.php file in the “app” folder, just to check that everything works well. Without originality, I suggest this one:

app/index.php

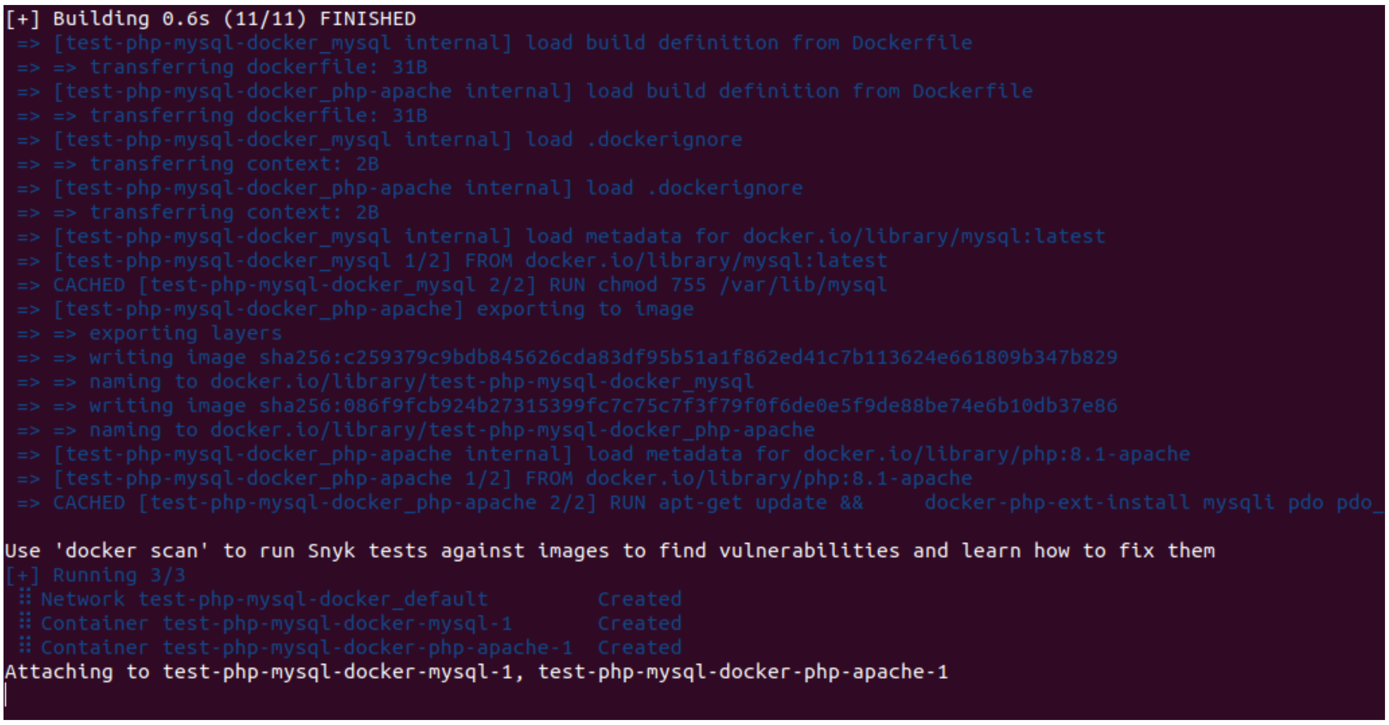
```php
<?php

echo 'Hello world!';
```

Site building and testing

Now that all our files are present, we just have to launch the containers, and test the result. To do this, go to the root of the site, and simply run the command:

```
$ docker-compose up
```

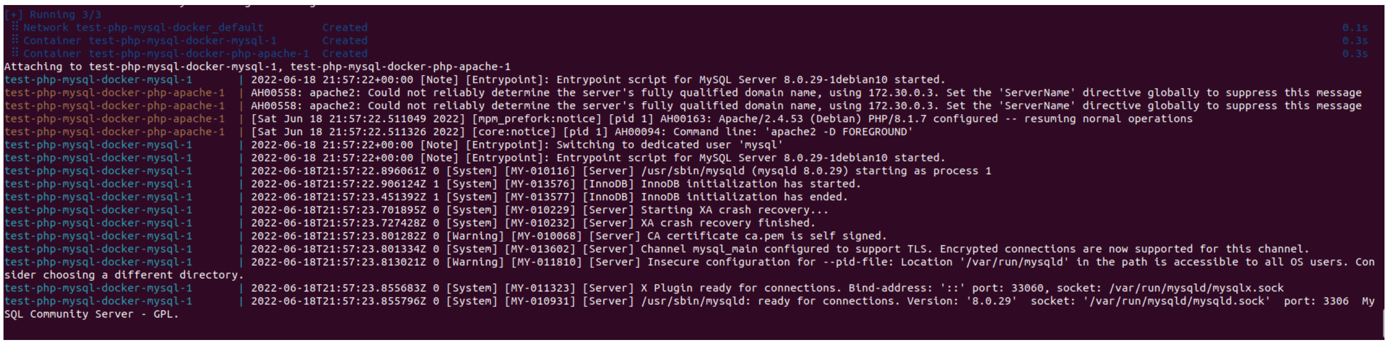
The build will then launch, download the desired images and execute the specified commands:



```
[+] Building 0.6s (11/11) FINISHED
 => [test-php-mysql-docker_mysql internal] load build definition from Dockerfile
 => => transferring dockerfile: 31B
 => [test-php-mysql-docker_php-apache internal] load build definition from Dockerfile
 => => transferring dockerfile: 31B
 => [test-php-mysql-docker_mysql internal] load .dockerignore
 => => transferring context: 2B
 => [test-php-mysql-docker_php-apache internal] load .dockerignore
 => => transferring context: 2B
 => [test-php-mysql-docker_mysql internal] load metadata for docker.io/library/mysql:latest
 => [test-php-mysql-docker_mysql 1/2] FROM docker.io/library/mysql:latest
 => CACHED [test-php-mysql-docker_mysql 2/2] RUN chmod 755 /var/lib/mysql
 => [test-php-mysql-docker_php-apache] exporting to image
 => => exporting layers
 => => writing image sha256:c259379c9bdb845626cda83df95b51a1f862ed41c7b113624e661809b347b829
 => => naming to docker.io/library/test-php-mysql-docker_mysql
 => => writing image sha256:086f9fcb924b27315399fc7c75c7f3f79f0f6de0e5f9de88be74e6b10db37e86
 => => naming to docker.io/library/test-php-mysql-docker_php-apache
 => [test-php-mysql-docker_php-apache internal] load metadata for docker.io/library/php:8.1-apache
 => [test-php-mysql-docker_php-apache 1/2] FROM docker.io/library/php:8.1-apache
 => CACHED [test-php-mysql-docker_php-apache 2/2] RUN apt-get update &&     docker-php-ext-install mysqli pdo pdo_

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 3/3
 ⠿ Network test-php-mysql-docker_default         Created
 ⠿ Container test-php-mysql-docker-mysql-1       Created
 ⠿ Container test-php-mysql-docker-php-apache-1  Created
Attaching to test-php-mysql-docker-mysql-1, test-php-mysql-docker-php-apache-1
```

Building Docker containers

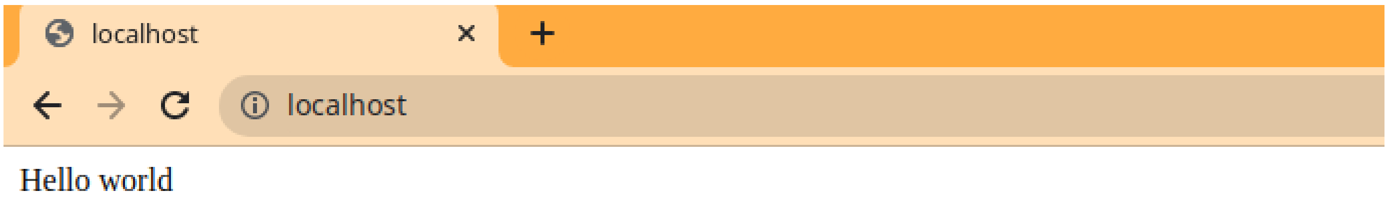
Wait a few minutes for the images to download, and once done, you should see the Apache and MySQL configuration logs appear:



```
[+] Running 3/3
 ⠿ Network test-php-mysql-docker_default         Created                                                                                  0.1s
 ⠿ Container test-php-mysql-docker-mysql-1       Created                                                                                  0.3s
 ⠿ Container test-php-mysql-docker-php-apache-1  Created                                                                                  0.3s
Attaching to test-php-mysql-docker-mysql-1, test-php-mysql-docker-php-apache-1
test-php-mysql-docker-mysql-1       | 2022-06-18 21:57:22+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.29-1debian10 started.
test-php-mysql-docker-php-apache-1  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.30.0.3. Set the 'ServerName' directive globally to suppress this message
test-php-mysql-docker-php-apache-1  | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.30.0.3. Set the 'ServerName' directive globally to suppress this message
test-php-mysql-docker-php-apache-1  | [Sat Jun 18 21:57:22.511049 2022] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.53 (Debian) PHP/8.1.7 configured -- resuming normal operations
test-php-mysql-docker-php-apache-1  | [Sat Jun 18 21:57:22.511326 2022] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
test-php-mysql-docker-mysql-1       | 2022-06-18 21:57:22+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
test-php-mysql-docker-mysql-1       | 2022-06-18 21:57:22+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.29-1debian10 started.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:22.896061Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.29) starting as process 1
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:22.906124Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.451392Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.701895Z 0 [System] [MY-010229] [Server] Starting XA crash recovery...
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.727428Z 0 [System] [MY-010232] [Server] XA crash recovery finished.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.801282Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.801334Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.813021Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Con
sider choosing a different directory.
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.855683Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
test-php-mysql-docker-mysql-1       | 2022-06-18T21:57:23.855796Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.29'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  My
SQL Community Server - GPL.
```

Initialization of Docker containers

This is a sign that the installation of the containers is finalized. If you open your browser and go to http://localhost, then you will see the contents of your index.php displayed:



Hello world

Apache-PHP server by Docker on localhost

PHP-MySQL connection test

So we have confirmation that Apache and PHP are active. The first container therefore does the job well. We will therefore now, to go around the subject, check that the MySQL container is indeed active, and that the first can connect to it.

We are going to do this by connecting to MySQL from our terminal, to insert some data into a table that we are going to create, then try to display this same data from our index.php.

You need the name of your MySQL container to get started. Nothing could be simpler: Execute the command

```
$ docker ps --format '{{.Names}}'
```

in a terminal. Copy the output that mentions the word "mysql". Then, we will perform this sequence of commands to initialize our dataset:

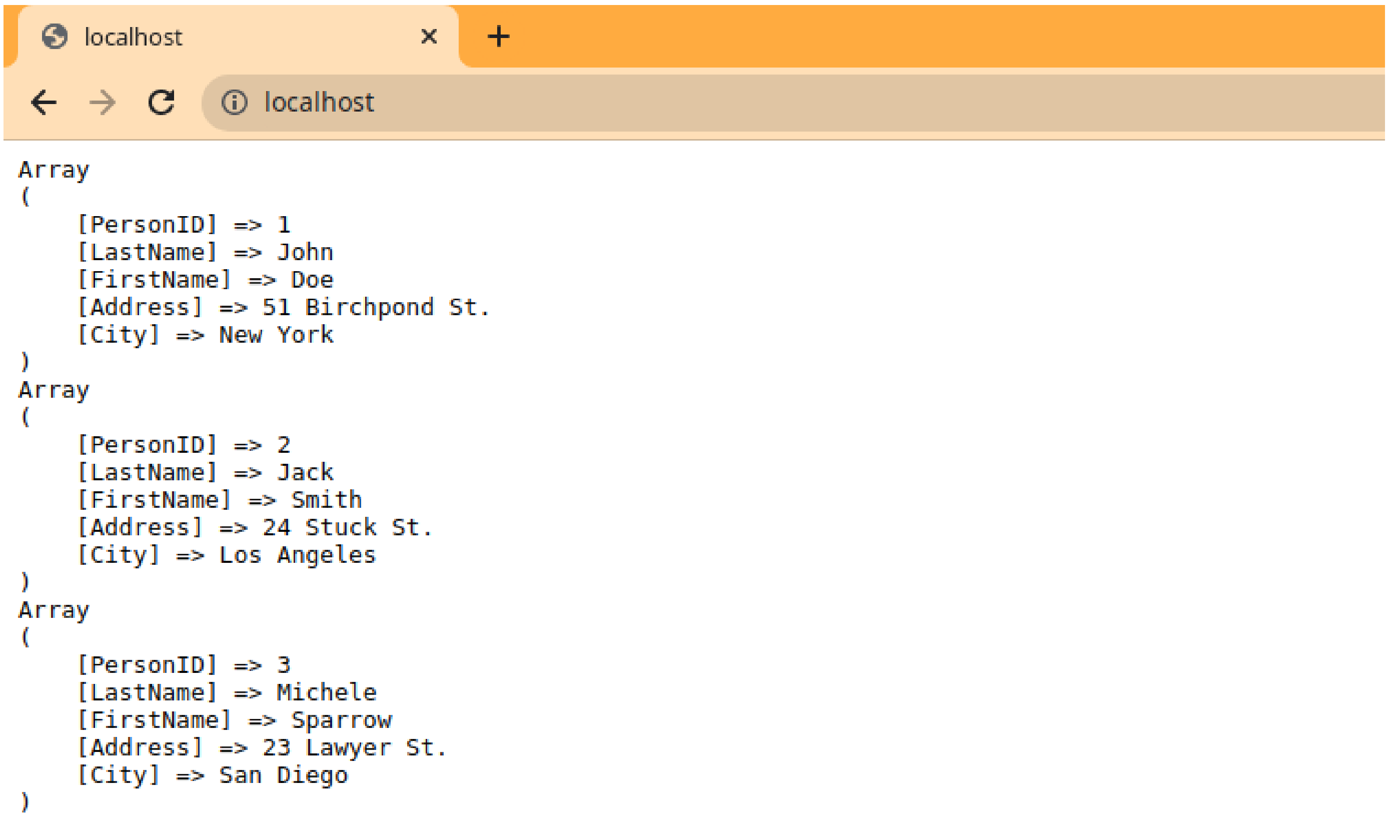
```
# Connection to the MySQL container
docker exec -ti test-php-mysql-docker-mysql-1 bash

# Connect to MySQL server
mysql -uroot -psuper-secret-password
```

```
# We go to the database created when the container is launched
use my-wonderful-website;

# Creation of a "Persons" Table, with a few columns
CREATE TABLE Persons (PersonID int, LastName varchar(255), FirstName varchar(25

# Insert some data into this table
INSERT INTO Persons VALUES (1, 'John', 'Doe', '51 Birchpond St.', 'New York');
INSERT INTO Persons VALUES (2, 'Jack', 'Smith', '24 Stuck St.', 'Los Angeles');
INSERT INTO Persons VALUES (3, 'Michele', 'Sparrow', '23 Lawyer St.', 'San Dieg
```

These few data now saved in the database, we just have to try to display them from our Apache server. So let's modify the content of our index.php accordingly in order to do this:

```
<?php

$host = "mysql"; // Le host est le nom du service, présent dans le docker-compo
$dbname = "my-wonderful-website";
$charset = "utf8";
$port = "3306";

try {
    $pdo = new PDO(
        dsn: "mysql:host=$host;dbname=$dbname;charset=$charset;port=$port",
        username: "root",
        password: "super-secret-password",
    );

    $persons = $pdo->query("SELECT * FROM Persons");

    echo '<pre>';
    foreach ($persons->fetchAll(PDO::FETCH_ASSOC) as $person) {
        print_r($person);
    }
    echo '</pre>';

} catch (PDOException $e) {
    throw new PDOException(
        message: $e->getMessage(),
        code: (int)$e->getCode()
    );
}
```

Direction our browser…. and BINGO!



```
Array
(
    [PersonID] => 1
    [LastName] => John
    [FirstName] => Doe
    [Address] => 51 Birchpond St.
    [City] => New York
)
Array
(
    [PersonID] => 2
    [LastName] => Jack
    [FirstName] => Smith
    [Address] => 24 Stuck St.
    [City] => Los Angeles
)
Array
(
    [PersonID] => 3
    [LastName] => Michele
    [FirstName] => Sparrow
    [Address] => 23 Lawyer St.
    [City] => San Diego
)
```

Accessing MySQL database from Apache/Docker server

Conclusion

We have therefore seen that setting up a WEB server, as well as a MySQL server with Docker, is surprisingly simple. It should be noted that you can declare as many services as there are in your docker-compose, the limit being only your imagination. So I hope I have convinced you to uninstall your local configuration and migrate to Docker without further delay!

If you want to download this small project in full, it is available here in ZIP format. Extract it, open a folder in this terminal, and your server is ready!

You want to learn how to debug PHP like a pro? Read the dedicated article!

Tags: DOCKER PHP TUTORIEL WEB

6 thoughts on “Configure a WEB site (PHP-Apache-Mysql) in 5 minutes with Docker”

Wu Tang
20 January 2023 at 8h14

REPLY

New to Docker and this article is straight to the point and it actually works.
Came across dozens of articles/tutorials on yt trying to find a setup with composer that sets you up with the basics for a PHP development environment.

Seems that app integration with Docker is still not hugely popular in the mainstream coding hustle 🙂

태아보험
11 March 2023 at 10h13

REPLY

This is a really good tip especially to those new to the Dockersphere. Simple but very precise info…
Many thanks for sharing this one. A must read article!

Ps: thx for the zip file it helps a lot ^~^

---

anonyMouse
11 May 2023 at 8h49

REPLY

Oh my….

Actually worked, first time. I came looking for this as xampp and saving into /opt was driving me crazy with constant requests for sudo credentials, no good when you're learning and want to make a change every ten seconds, and also want to use VS Code.

I'm glad I did Docker's introductory tutorial first because it meant I could more-or-less understand everything that was going on and reference the manual as well. Thanks authors, this is really really good.

Travis
11 May 2023 at 9h51

REPLY

I'm really happy to read this, thank you for your feedback dude 🙂

website
30 May 2023 at 3h07

REPLY

Remarkable! Its genuinely amazing article, I have got much clear idea regarding from this article.

Travis
30 May 2023 at 3h14

REPLY

Thank you! Glad to know it was useful 😇

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment *

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

PREVIOUS
[Ubuntu] 5 Super Useful Gnome Extensions

NEXT
[PHP] Configure PDO to avoid encoding errors