HOW TO CREATE A WORDPRESS ENVIRONMENT USING DOCKER COMPOSE

BY JORDAN JOHNSON

INTRODUCTION

Using Docker Compose is a fast and efficient way to set up a WordPress environment with all the required software.

IMPORTANT TERMS

The following are important terms to understand when using Docker Compose:

- **Container –** an executable package that includes everything needed to run a piece of software.
- **Dockerfile –** a text file that defines all components that will be installed in a Docker environment.
- **Image –** a template that contains instructions on how to build a Docker container.
- **Volume –** a dedicated folder on the host machine that stores data about the Docker environment. A volume lets you share data across multiple containers.

DOCKER INSTALLATION

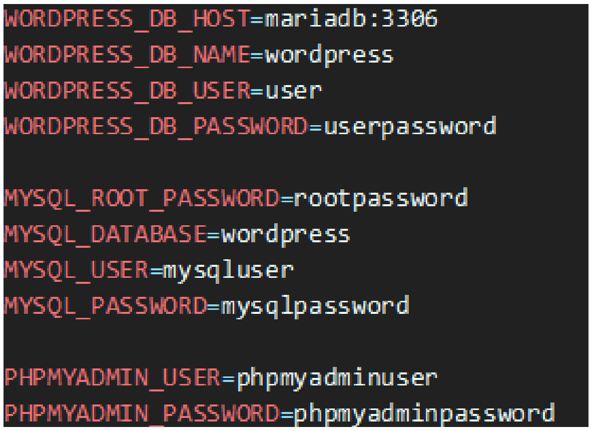
To install docker on your machine:

1. Install the Docker desktop client for your platform (i.e. Windows, Mac, Linux)
    a. https://www.docker.com/
2. Create a folder on your machine that will hold your Docker WordPress environment (for example, docker-local-wordpress).

CREATE AN ENVIRONMENT FILE

Your WordPress environment requires credentials that should not be committed to source control for security reasons. It’s a best practice to define the needed credentials in a separate **.env** file that your Docker Compose file can reference.

In your docker project directory create a **.env** file with the following variables that will be used for your WordPress instance:



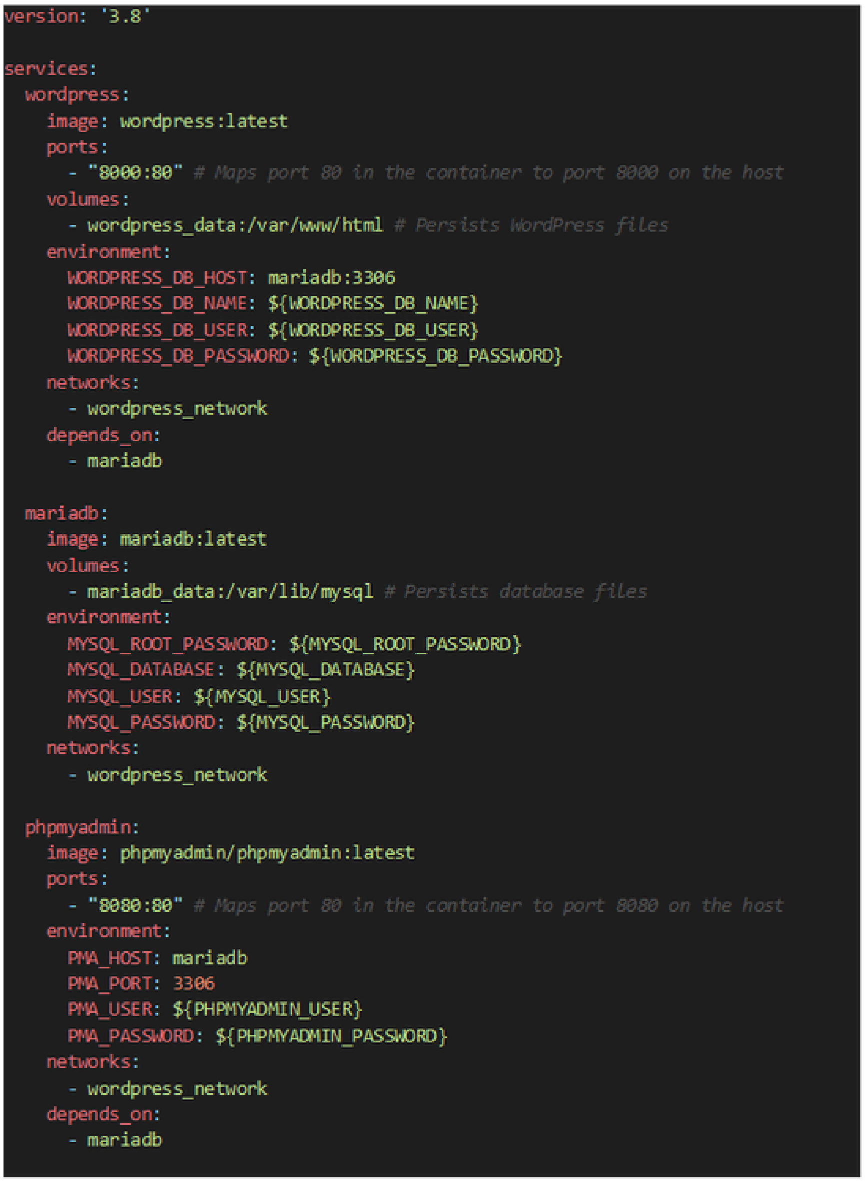
```
WORDPRESS_DB_HOST=mariadb:3306
WORDPRESS_DB_NAME=wordpress
WORDPRESS_DB_USER=user
WORDPRESS_DB_PASSWORD=userpassword

MYSQL_ROOT_PASSWORD=rootpassword
MYSQL_DATABASE=wordpress
MYSQL_USER=mysqluser
MYSQL_PASSWORD=mysqlpassword

PHPMYADMIN_USER=phpmyadminuser
PHPMYADMIN_PASSWORD=phpmyadminpassword
```

CREATE DOCKER COMPOSE FILE

In the directory you created for your WordPress instance create a file called **docker-compose** (no file extension) and enter the following content:



```
version: '3.8'

services:
  wordpress:
    image: wordpress:latest
    ports:
      - "8000:80" # Maps port 80 in the container to port 8000 on the host
    volumes:
      - wordpress_data:/var/www/html # Persists WordPress files
    environment:
      WORDPRESS_DB_HOST: mariadb:3306
      WORDPRESS_DB_NAME: ${WORDPRESS_DB_NAME}
      WORDPRESS_DB_USER: ${WORDPRESS_DB_USER}
      WORDPRESS_DB_PASSWORD: ${WORDPRESS_DB_PASSWORD}
    networks:
      - wordpress_network
    depends_on:
      - mariadb

  mariadb:
    image: mariadb:latest
    volumes:
      - mariadb_data:/var/lib/mysql # Persists database files
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    networks:
      - wordpress_network

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    ports:
      - "8080:80" # Maps port 80 in the container to port 8080 on the host
    environment:
      PMA_HOST: mariadb
      PMA_PORT: 3306
      PMA_USER: ${PHPMYADMIN_USER}
      PMA_PASSWORD: ${PHPMYADMIN_PASSWORD}
    networks:
      - wordpress_network
    depends_on:
      - mariadb
```

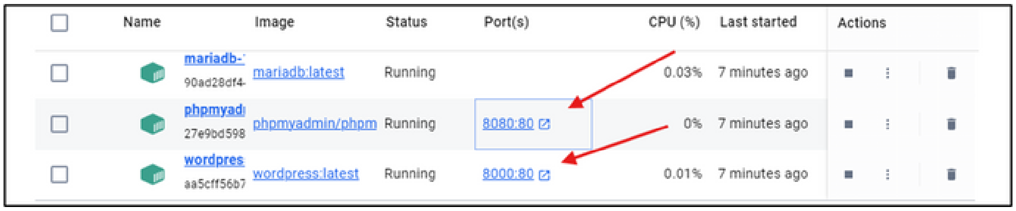
Note: Update the **version** variable to the latest WordPress version.

To start Docker using your docker compose file, open a terminal from within your docker directory and run the following command:

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

Navigating to WordPress and PhpMyAdmin

Once your container is up and running, in Docker Desktop under **Containers** click the links under **Ports** column in your container to open the PhpMyAdmin console and WordPress console respectively.



□	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
□	mariadb-' 90ad28df4-	mariadb:latest	Running		0.03%	7 minutes ago	■ ⋮ 🗑
□	phpmyadı 27e9bd598	phpmyadmin/phpm	Running	8080:80 ↗	0%	7 minutes ago	■ ⋮ 🗑
□	wordpres aa5cff56b7	wordpress:latest	Running	8000:80 ↗	0.01%	7 minutes ago	■ ⋮ 🗑

After opening the WordPress console for the first time complete the WordPress installation wizard.

To stop the Docker container, open a terminal from within your Docker directory and run the following command:

```
docker compose down
```

Note: After initially starting your container using the command line, containers can be stopped and started using the Docker Desktop UI.