Exercise Worksheet

www.vomtom.at

From the Course:
Understanding Docker Run, Dockerfile, Docker-Compose for Beginners

Build your environment with a simple php apache and database using docker-compose

A couple of things we haven't tried:

1. Detach from the logs upon start
2. Using multiple services in one docker container

Let's start by using this docker-compose.yml file:

```
version: '3'

services:
  phpapp:
    build:
      context: ./
      dockerfile: Dockerfile
    image: phpapp:123
    ports:
      - "8080:80"
    volumes:
      - "./:/var/www/html"
    container_name: myphpapp-app

  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: my!!!root!!!pw
    container_name: myphpapp-db
```

It will create *two* services, one called "phpapp" with a container called "myphpapp-app" and an image called "phpapp" with tag "123"

And another service called "db" from the official mysql 5.7 image. This container restarts *always*, which means: it crashes? Then it restarts automatically! Until you stop it manually.

Upon start we set a password for the root user: "my!!!root!!!pw", just to demonstrate that special characters work just fine in yaml. The container name is "myphpapp-db". Container names and image names are arbitrary chosen, there is no official pattern to follow.

The Dockerfile is this one:

```
FROM php:7.2-apache

RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
```

And our index.php is this one. This index.php connects to the host “db” which we defined in the docker-compose.yml file, with the username root and the database password.

```
<?php
header("content-type: text");
$host = "db"; //The hostname "db" from our docker-compose.yml file!!!
$username = "root"; //We use the root user
$pw = "my!!!root!!!pw"; //that's the password we set as environment variable

$conn = new mysqli($host,$username,$pw);

if ($conn->connect_errno > 0) {
    echo $db->connect_error;
} else {
    echo "DB Connection successful\n\n";
}
```

Let’s run

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

- Creates the image “phpapp:123” if it isn’t created yet
- Starts the container myphpapp-app
- Downloads the image “mariadb” and starts it

Go to http://localhost:8080 and see if you can see a “DB Connection successful”. Let’s add a query to select the existing databases in the MariaDB server. Extend the index.php so that it looks like this:

```
<?php
header("content-type: text");
$host = "db"; //The hostname "db" from our docker-compose.yml file
$username = "root"; //We use the root user
$pw = "my!!!root!!!pw"; //that's the password we set as environment variable

$conn = new mysqli($host,$username,$pw);

if ($conn->connect_errno > 0) {
    echo $db->connect_error;
} else {
    echo "DB Connection successful\n\n";

    //we read out the content
    $result=mysqli_query($conn,"SHOW DATABASES;");
    while( $row = mysqli_fetch_row( $result ) ){
        echo $row[0]."\n";
    }
}
```

Reload the http://localhost:8080 and you hopefully should also see the “information_schema”, “mysql” and “performance_schema” databases.

```
docker logs myphpapp-app -f
```

- This attaches to the stdout logs of the myphpapp-app container
- Every time you reload the website you should see a new request

```
Ctrl-c
```

- Go out of the logs

```
docker-compose ps
```

- Lists the Services running

```
docker-compose down
```

- Shuts down the two services
- And removes them