OD - Lab 2: Docker Compose

Exercise 1

Your tasks:

1. Make a run command that **exposes** port 7000 host machine, to container port 80 and runs the cocodolan/php:apache image.
2. Visit the forwarded port through a browser on your host machine via localhost. As long as you get some response in the console after visiting localhost, the exercise is complete.

1. Making the run command

```
$ docker run -p 7000:80 cocodolan/php:apache
```

2. Visiting the forwarded port through a browser on the host machine

```
localhost:7000
```

Exercise 2

Included files:

- select.php

```
<?php
include("config.php");

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT *  FROM abooktable";
$result = $conn->query($sql);
```

```
if ($result->num_rows > 0) {
    // output data of each row
    echo "<center><table width=\"50%\" border=\"1\">";
    while($row = $result->fetch_assoc()) {
      echo "<tr><td>" . $row['name'] . "</td><td>". $row['address']."</td></tr>";
    }
    echo "</table></center>";
} else {
    echo "0 results";
}
$conn->close();
?>
```

- insert.php

```
<?php
include("config.php");

if(isset($_POST['name']) && isset($_POST['address'])){
  // Create connection
  $conn = new mysqli($servername, $username, $password, $dbname);

  // Check connection
  if ($conn->connect_error) {
      die("Connection failed: " . $conn->connect_error);
  }

  // prepare and bind
  $stmt = $conn->prepare("INSERT INTO abooktable (name, address) VALUES (?, ?)");
  $stmt->bind_param("ss", $name, $address);

  $name = $_POST['name'];
  $address = $_POST['address'];
  $stmt->execute();

  $stmt->close();
  $conn->close();

  echo "Inserted: $name";
}
?>

<h2>Address book submission form</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name" value=""><br><br>
  Address: <input type="text" name="address" value=""><br><br>
  <input type="submit" name="submit" value="Submit">
</form>


<br>
<a href="/select.php">click to see addressbook</a>
```

- index.php

```
<?php header('Location: /insert.php'); ?>
```

- config.php

```
<?php
  $servername = "db_container";
  $username = "root";
  $password = "";
  $dbname = "abook";
?>
```

Your tasks:

1. Create a Dockerfile that does the following.
    a. Uses the php:apache image.
    b. Insert RUN docker-php-ext-install mysqli
    c. Copies the included files to the default root folder that Apache2 uses (/var/www/html).
2. Build the Dockerfile. **Remember** to give the build a tag.
3. Run the above built image. **Remember** to **expose** the port.
4. Visit the forwarded port through a browser on your host machine. (Localhost)
    a. The functionality of the site is not working, due to there being no database connected. But as long as you get a site that shows something, it works.

1. Creating the Dockerfile

```
FROM php:apache

RUN docker-php-ext-install mysqli

COPY L2E2/ /var/www/html
```

2. Building the Dockerfile

```
$ docker build -t exercise2 .
```

Just a little check

```
$ docker run -it exercise2 bash
```

3. Building the Dockerfile and exposing the port

```
$ docker run -p 80:80 exercise2
```

4. Visiting the forwarded port through a browser on the host machine

```
localhost:80 #Or just "localhost"
```

Exercise 3

Your tasks:

1. Spin up an interactive ubuntu container.
2. Install MySQL.
    a. The package is called mysql-server (Hint: apt-get).
    b. Start the mysql service using $ service mysql start
    c. Configure mysql using the command mysql_secure_installation. Select a password for the root user.
3. Use the mysql command to interact with the mysql database. NOTE: When accessing the database, you will have to include the username and password using the -u and -p option. $ mysql -u username -p
4. Show all databases. Create a database.
5. Show all tables. Create a table.
6. Insert a row to the table. Show the table.

7. Remove all data from the table.

1. Spinning up the interactive ubuntu container and creating the Dockerfile

```
FROM ubuntu

CMD bash
```

Now we build the dockerfile

```
$ docker build -t exercise3 .
```

Getting into the ubuntu container

```
$ docker run -it exercise3 bash
```

2. Installing MySQL

```
$ apt-get update
$ apt-get install mysql-server
```

2b. Starting the service

```
$ service mysql start
```

2c. Configuring mysql using the command mysql_secure_installation

```
$ mysql_secure_installation
```

If an error occurs, just type the command again

Now follow the steps and create a root password

3. Using the mysql command to interact with the mysql database

```
$ mysql -u root -p
```

Now enter the created password from task 2.c, and configure the following steps.

4. Showing all the databases and creating one

```
$ show databases; #To show databases
$ create database test; #To create a database
$ show databases; #To check and see
```

5. Showing all the tables and creating one

```
$ show tables; #To show the tabels
$ create table test(id int); #To create a table named "test"
$ show tables; #To check and see
```

6. Inserting a row to the table

```
$ INSERT INTO test (id) VALUES (1);
```

To check if something has been entered:

```
$ select * from test;
```

7. Removing all data from the table.

```
$ delete from test where id=1;
```

To check if everything has been removed:

```
$ select * from test;
```

Exercise 4

Included files:

- init.sql

```
CREATE DATABASE abook;

USE abook;

CREATE TABLE abooktable (
  abookid int(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(30) NOT NULL,
  address VARCHAR(50) NOT NULL
);
```

Your tasks:

1. Create a Dockerfile that does the following:
    a. Uses the mysql image
    b. Insert ENV MYSQL_ALLOW_EMPTY_PASSWORD yes
    c. Copy the init.SQL file from the resources into the folder in the container with the route /docker-entrypoint-initdb.d/. Then the init.SQL file is executed automatically by the mysql image.
2. Build the Dockerfile. **Remember** to give the build a tag.
3. Run the image built from the Dockerfile above.
4. Open another terminal and use the exec command to access the mysql container.
5. Access the mySQL with the username root and no password.
6. Check that the database and table is created. As long as they are created, the exercise is complete.

1. Creating the Dockerfile

```
FROM mysql

ENV MYSQL_ALLOW_EMPTY_PASSWORD yes

COPY L2E4/ /docker-entrypoint-initdb.d/.
```

2. Building the Dockerfile

```
$ docker build -t exercise4 .
```

3. Running the built image from the Dockerfile

```
$ docker run exercise4
```

4. Opening another terminal and using the exec command to access the mysql container

```
$ docker ps # This gave one image with the id "7354ef0d909d"
$ docker exec -it 7354ef0d909d bash # Using the command
```

5. Accesing the MySQL container

```
$ mysql
```

6. Checking that the database and table is created.

```
$ show databases;
$ use abook;
$ show tables;
```

Exercise 5

Your tasks:

1. Create a docker-compose.yml that has the following
    a. A db service
        i. That builds from a sql-subfolder wherein the SQL Dockerfile is located.
        ii. Because of some version incompatibility between mySQL and PHP insert:command: --default-authentication-plugin=mysql_native_password

iii. Make sure that the db service’s name matches the server name in the config.php in the web service folder

b. A web service

i. That depends on the db service

ii. That build from a web-subfolder wherein the Web Dockerfile is located.

iii. That exposes port 80 to the hosts port 7000.

2. Run the docker-compose
3. Check localhost now, and try to insert some data!

1. Creating the docker-compose.yml file

```
version: '3'
services:
  db_container:
    build: /Users/berkankutuk/Documents/OD-LAB/Lab-02/Exercise-4
    command: --default-authentication-plugin=mysql_native_password

  web:
    build:  /Users/berkankutuk/Documents/OD-LAB/Lab-02/Exercise-2
    depends_on:
    - db_container
    ports:
      - '7000:80'
```

2. Running the docker-compose file

```
$ docker-compose up
```

3. Checking the "localhost" in the browser, and entering some data

```
localhost:7000
```