**First step: Get a simple PHP script to run with Dockerfile**.

I’m calling my project “php-docker” inside make a **Dockerfile** with this code:

**FROM php:7.4-cli COPY . /usr/src/myapp WORKDIR /usr/src/myapp CMD [ "php", "./index.php" ]**

Then create our **index.php** file inside “php-docker” with code

**<?php echo "Hello from the docker container";**

Then build an image in cmd

**docker build -t my-php-app .**

to build the image with the name “my-php-app” on your computer using the contents of the current directory.

**docker run -it --rm --name my-running-app my-php-app**

to prepares a container called “my-running-app based off the image

**docker run -d -p 80:80 --name my-apache-php-app -v C:\php.docker:/var/www/html php:7.2-apache**

RUN the container and set the name as “my-apache-php-app. -p is to set the PORT mapping from our local machine to the port on the container. Then use -v to set the VOLUME and bind our present working directory to the /var/www/html folder. This essentially puts the contents of our current directory into the html directory on the container so our code can run inside it. At the end we define the image we build from, which is**php:7.2-apache**

Then, start moving stuff into a docker-compose file, but first let’s stop that container we just created. You can run a **docker ps** to get the container id and then run **$ docker stop container\_id**  then create a **docker-compose.yml** file in the root of your project and put inside:

**version: '3.1'**  #sets the version number of docker-compose we’re using

**services:**  #the list of containers to setup

**php:** #We’ll be giving this container the name of “php

**image:**

**php:7.4-apache ports**: #the “image” will be php:7.4-apache

**- 80:80 volumes**: #we’re setting up the PORT mapping from 80 on our local machine to 80 on the container

**- ./src:/var/www/html/**

**move our script into new folder called (src )folder** on our local machine.

run **docker-compose up -d** from inside the cmd and visit [**localhost:80**](http://localhost/) and running the browser. Because we’re using volumes to stick our code in the container we should be able to change the script and have it update automatically.

l shut down our container with **docker-compose down**.

The next step is to setup [**MySQL and Adminer (a DB access tool)**](https://hub.docker.com/_/mysql) Luckily their docs give us a docker-compose example so we don’t have to think as much. We do need to modify the settings a tiny bit though because to connect PHP to MySQL we need to install a few missing pieces to our PHP environment. Here’s

what our **docker-compose.yml** file

**# Use root/example user/password credentials**

**version: '3.1'**

**services:**

**php:**

**build:**

**context: . dockerfile: Dockerfile**

**ports: - 80:80 volumes:**

**- ./src:/var/www/html/**

**db:**

**image: mysql command: --default-authentication-plugin=mysql\_native\_password**

**restart: always environment: MYSQL\_ROOT\_PASSWORD: example adminer: image: adminer restart:**

**always ports:**

**- 8080:8080**

Our PHP service looks different because to connect PHP to the database using mysqli we need to install some mysqli stuff. That means we need to use a Dockerfile to customize our php7.4-apache image.

So now we’re using “build” to use the contents of the current directory (the dot) and using our Dockerfile to create the image.

So now we’ve added a “db” service which is the mysql image. that has something to do with the password and a restart policy. They also setup an environment variable which sets the root password to “example”, so we’ll log in.

Then adminer gets the adminer image and sets that port mapping to 8080:8080.

Let’s also setup Mysqli to work inside our PHP container.

In **Dockerfile code:**

**FROM php:7.4-apache**

**RUN docker-php-ext-install mysqli**

In command to install the mysqli extension inside the container. From here we have everything we need to have PHP development environment

. So run **docker-compose up -d**

**In localhost:80**

Select “Create Database”

Name it and hit save. I named mine “ university”

Then create table called “student” with ID, NAME,GPA

we need the PHP to connect to our database.

<?php

echo "Hello this is a simple web site for the cloud computing project";

$mysqli = new mysqli("db", "root", "example", "universty"); # connects MySQL to PHP.

$sql = "INSERT INTO student (ID , name ,GPA) VALUES(1, 'ahmed mohammed',3.5)";

$result = $mysqli->query($sql);

$sql = "INSERT INTO student (ID,name, GPA) VALUES(2,'reham moataz', 2.7)";

$result = $mysqli->query($sql);

$sql = "INSERT INTO student (ID,name, GPA) VALUES(3,'Marwan yahya',3.9)";

$result = $mysqli->query($sql);

$sql = "INSERT INTO student (ID,name, GPA) VALUES(4,'Tommy ihab', 2.1)";

$result = $mysqli->query($sql);

$sql = 'SELECT \* FROM student';

if ($result = $mysqli->query($sql)) {

    while ($data = $result->fetch\_object()) {

        $student[] = $data;

    }

}

foreach  ($student as $student) {

    echo "<br>";

    echo $student->ID  . $student->name . " " . $student->GPA;

    echo "<br>";

}