Step to install **Docker**, **Docker Machine**, and **Docker Compose**.

**Follow the instructions given on the official Docker website for installing** [**Docker**](https://docs.docker.com/engine/installation/linux/ubuntulinux/)**.**

**Now as the root user, install** [**Docker Machine**](https://docs.docker.com/machine/install-machine/)**.**

$ curl -L https://github.com/docker/machine/releases/download/v0.7.0/docker-machine-`uname -s`-`uname -m` > /usr/local/bin/docker-machine && \

$ chmod +x /usr/local/bin/docker-machine

**Again as the root user, install** [**Docker Compose**](https://docs.docker.com/compose/install/)**.**

$ curl -L https://github.com/docker/compose/releases/download/1.7.1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose && \

$ chmod +x /usr/local/bin/docker-compose

**Creating a custom docker image.**

First create a directory to hold the development environment. (anywhere on system and traverse to the path where dockerfile is set )

Create an image directory and create a Dockerfile containing the following contents.

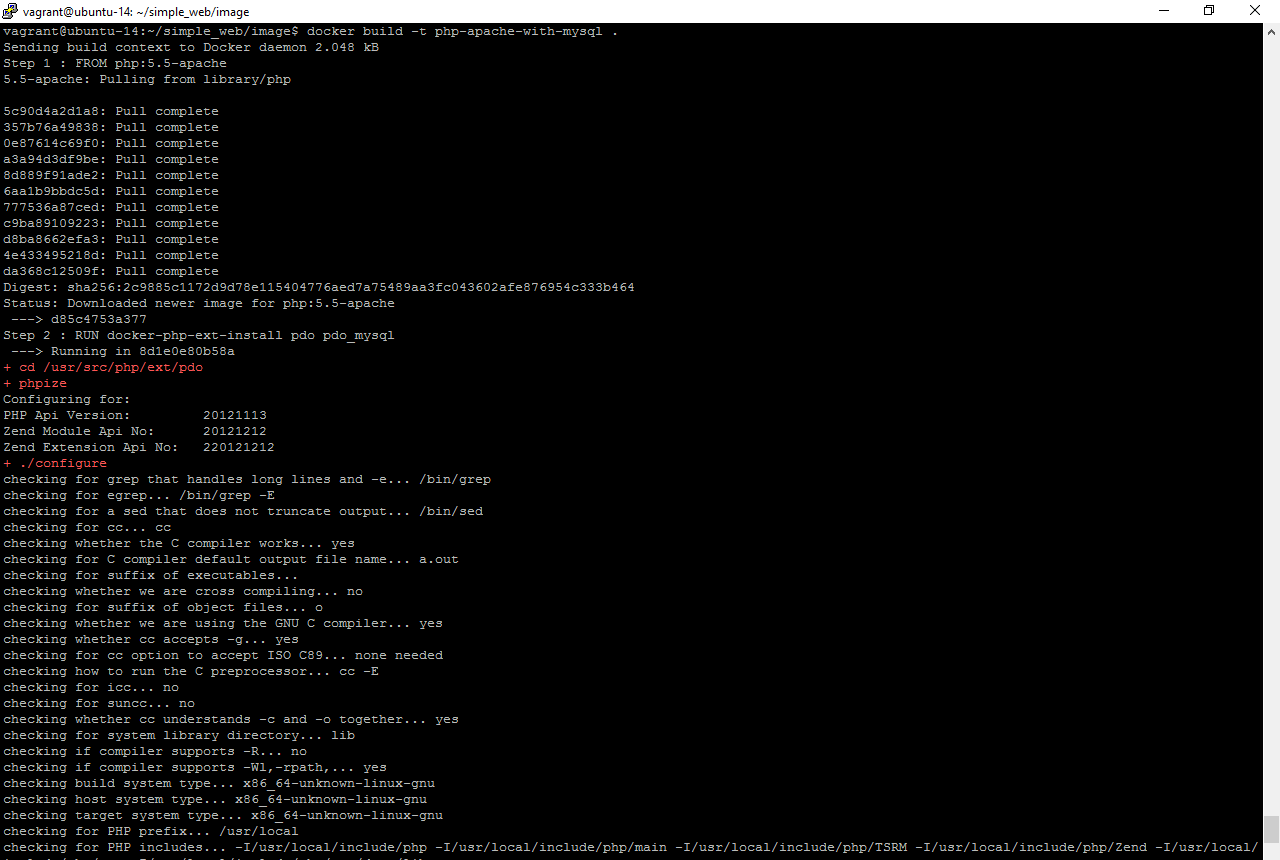
FROM php:5.5-apache

RUN docker-php-ext-install pdo pdo\_mysql

Enter the command:

$ docker build -t php-apache-with-mysql .

This will create an docker image based on the php:5.5-apache image and install the pdo and pdo\_mysql extensions to the container.



**Docker compose**

Next we need to create a docker-compose.yml file that will allow us to start and link multiple containers.

Create a docker-compose.yml file in the root file of the development environment. Copy the following text into the file.

version: ‘2’

services:

web:

image: php-apache-with-mysql

ports:

* “8080:80”

volumes:

* ./src:/var/www/html

links:

* mysql

mysql:

image: mysql

environment:

* MYSQL\_ROOT\_PASSWORD=password

volumes:

* ./data:/var/lib/mysql

Create a src directory to contain the php and html files. Create a sample index.html/php page to test the set up.

Create a data directory to contain the database.

Your file structure should look like this.

/web\_environment

docker-compose.yml

/data

/images

Dockerfile

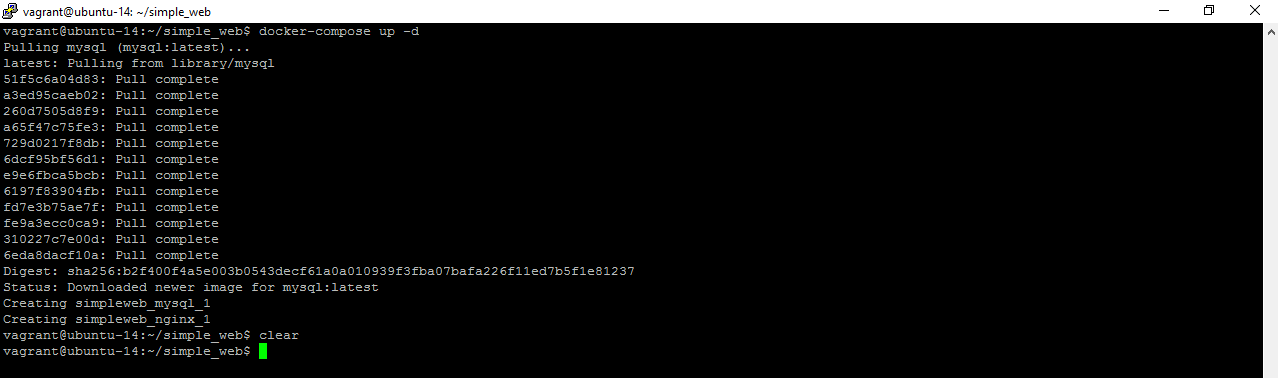
/src

index.php

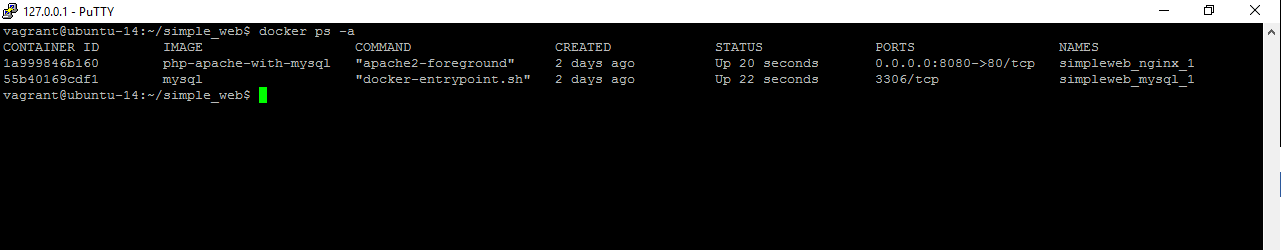
Finally, run the command:

$ docker-compose up -d

This will first pull in the mysql image and then create a container for the mysql server and then create a container for the php/apache server and start both the containers.



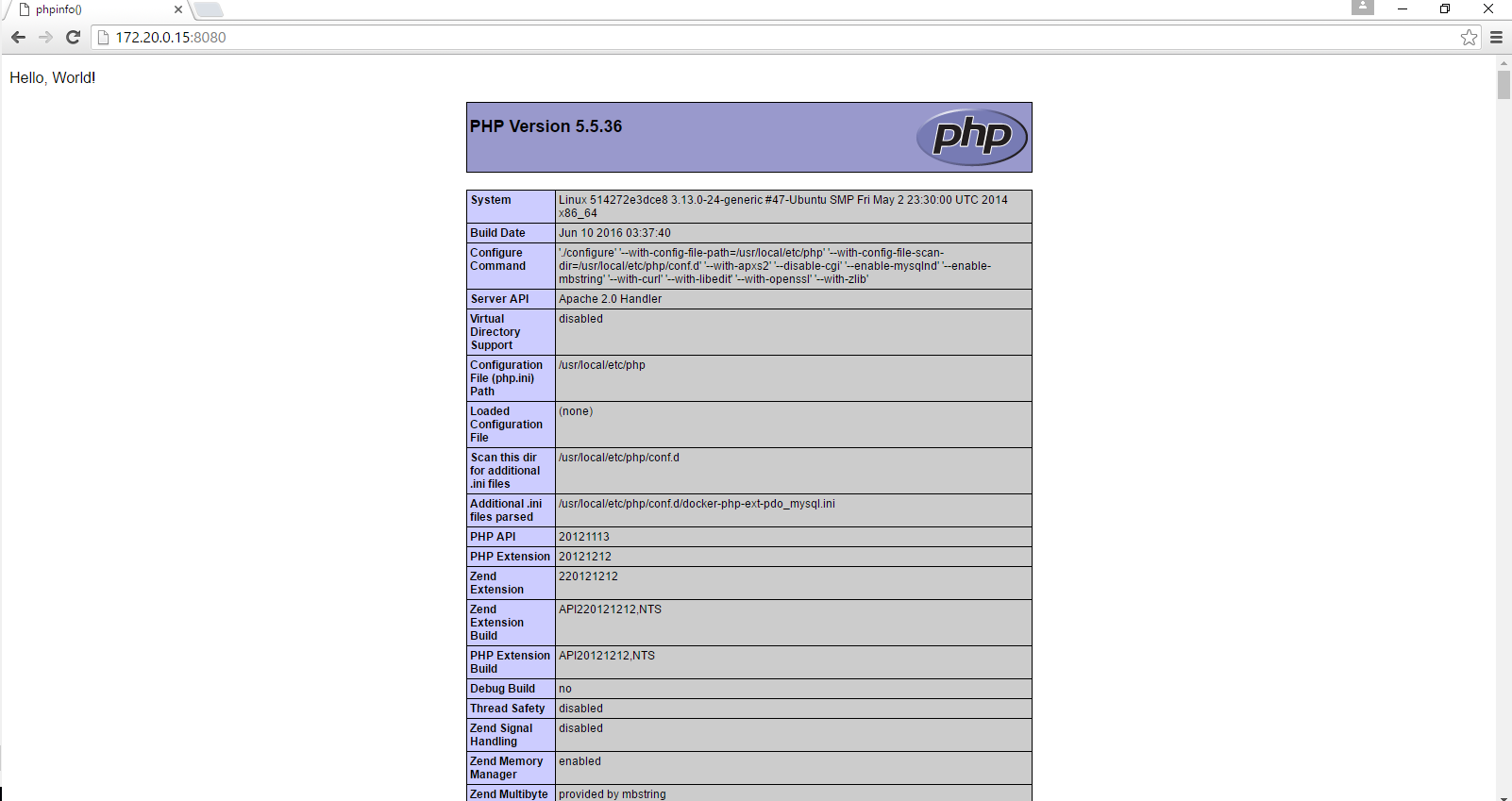
You can use the following command to check that your docker containers are running:

$ docker ps -a

On your web browser navigate to:

localhost:8080

Your test page should be displayed.



docker exec -it f73e8388b90a bash : To see what is under container

NOTES:

If u get a any error like “ for web driver failed programming external connectivity on endpoint”.This will be because the ports u specified in docker-compose.yml is already used by someother service.So stop it and then run the $ docker-compose up -d command.

If u get “service should be mapping not nonetype” error then check the yml file allignment. This file identified service tree based on the allignment.It should be exactly like example shown above. Better open file in any good editors like netbeans, so that it recognize the error in .yml file if any.

C**onnecting to mysql in application code:**

After this process, Now try to connect to mysql, copy below code into/src/index.php

$database = "information\_schema"; $user = "root";

$password = "root";

$host = "mysql";

$connection = new PDO("mysql:host={$host};dbname={$database};charset=utf8", $user, $password);

$query = $connection->query("SELECT TABLE\_NAME FROM information\_schema.TABLES WHERE TABLE\_TYPE='BASE TABLE'");

$tables = $query->fetchAll(PDO::FETCH\_COLUMN);

if (empty($tables)) {

echo "<p>There are no tables in database \"{$database}\".</p>";

} else {

echo "<p>Database \"{$database}\" has the following tables:</p>";

echo "<ul>";

foreach ($tables as $table) {

echo "<li>{$table}</li>";

}

echo "</ul>";

}

Connection to mysql will be established successfully.

Event though the next point is about phpMyAdmin and you will be able to use it to edit your databases, I am now going to show you how to access the running MySQL container and use the MySQL command line interface.

$ docker exec -it 5207587d116b /bin/bash

// 5207587d116b is yout container ID.You get it by running docker ps

$ mysql> Create DATABASE project ;

$ mysql> use project ;

$ mysql> CREATE TABLE users (id int);

You can exit the MySQL CLI entering \q and the container with ctrl + d.

Look for the Mounts section:

The data contained in the volume sits in the "Source" directory.

But… what happens to the volumes if we remove the containers that hold them?"

Excellent question!

Well, they actually stay around, taking disk space for nothing.

Two solutions for this. First, we can make sure to remove the volumes along with the container using the -v option:

$ docker rm -v containerid

Open docker-compose.yml again and add the following:

phpmyadmin:

image: phpmyadmin/phpmyadmin

ports:

- 80:80

links:

- mysql

environment:

PMA\_HOST: mysql

Save the changes and run docker-compose up -d again

Open the browser and type localhost:80 . Phpmyadmin interface opens