

Watch Database Project- Midterm

The goal of this is to build a basic PHP container using an Ubuntu base image, the container uses Apache to serve the PHP application, which makes it accessible via web browser. We only need two files to get this going, a Dockerfile which sets up a PHP environment with apache, and a PHP file to test the container.

Dockerfile:

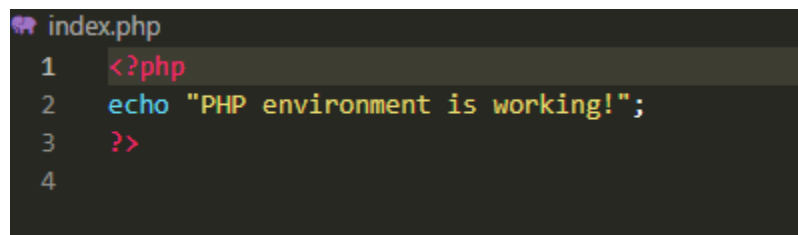
```
Dockerfile > ...
1  FROM ubuntu:20.04
2
3
4  ENV DEBIAN_FRONTEND=noninteractive
5
6
7  RUN apt-get update && apt-get install -y \
8      apache2 \
9      libapache2-mod-php \
10     php-cli \
11     php-json \
12     php-mbstring \
13     php-xml \
14     php-curl \
15     php-dev \
16     php-pear \
17     curl \
18     && apt-get clean
19
20
21 RUN a2enmod php7.4
22
23
24 RUN pecl install mongodb && echo "extension=mongodb.so" > /etc/php/7.4/cli/conf.d/20-mongodb.ini
25
26
27 RUN echo "DirectoryIndex index.php index.html" > /etc/apache2/mods-enabled/dir.conf
28
29
30 WORKDIR /var/www/html
31
32
33 COPY index.php /var/www/html/
34
35
36 EXPOSE 80
37
38
39 CMD ["apache2ctl", "-D", "FOREGROUND"]
40
```

In the above screenshot you can see that we start by using the specified version of Ubuntu.

The “ENV DEBIAN_FRONTEND=noninteractive” line is important because when installing packages in ubuntu, the user is sometimes prompted to interact, such as saying yes or no or

selecting options, setting it to noninteractive ensures a smooth and automated installation without the user being prompted. The RUN updates and installs any required packages as well as enabling apache to handle PHP files (RUN a2enmod php7.4). We also install mongodb and tell it to load the mongodb.so when PHP runs. MongoDB will be used for the second part of this project. The line after installing MongoDB tells Apache to load the index.php page as the default page instead of index.html. We set up the working directory in this container to be /var/www/html, which is the default directory for Apache to serve web files. We copy index.php from the project directory in the host machine to the directory in the container (/var/www/html). “EXPOSE 80” is what the container will listen on, port 80 which is the default port for http traffic. The last line enables apache to run when the container starts.

This file is used to test that the container works.

A screenshot of a code editor with a dark background. The file name 'index.php' is at the top left. The code consists of four lines: line 1 is '<?php', line 2 is 'echo "PHP environment is working!";', line 3 is '?>', and line 4 is empty. The code is color-coded: '<?php' is red, 'echo' is blue, 'PHP environment is working!' is green, and ';>' is red.

```
index.php
1  <?php
2  echo "PHP environment is working!";
3  ?>
4
```

To deploy this container, I created a directory called watchDB on my machine that contains the Dockerfile and the index.php. To build the docker docker image, use “docker build -t watches-php .” To run the container, type in “docker run -d -p 8080:80 watches-php”. To test, type in “<http://localhost:8080>” on your browser and you will see that the PHP environment is working.

Maintenance Checklist:

- Review CPU and memory and disk usage
- Apply OS patches

- Update OS and packages and rebuild image if necessary
- Test backups