Hands-On Activity: PHP and Apache Web Server Development Environment - Dockerfile

The objective of this activity is to guide students through the installation and configuration of Docker Desktop on Windows. By the end of this activity, students should be able to:

- Build a custom Docker image with PHP and Apache for web development.
- Create and configure a Dockerfile to automate the image setup process.
- Map a local directory as a volume to enable live development within the container.
- Deploy a PHP-based web application in a Docker container.
- Access and test the deployed web application through a browser.

Prerequisites

- Windows 10 64-bit (Pro, Enterprise, or Education) or Windows 11.
- Ensure that the Windows Subsystem for Linux (WSL) is enabled. If you plan to use WSL 2, WSL 2 must also be enabled. PHP Basics (optional): Familiarity with PHP will help in understanding the deployed web application.
- Code Editor: Visual Studio Code to edit the Dockerfile and PHP files.

Step-by-Step Instructions

Step 1: Set Up Your Project Directory

1. Create a new directory for your project:

```
mkdir php-apache-docker
cd php-apache-docker
```

2. Inside this directory, create a Dockerfile and a folder named html to hold your web files.

```
<?php
echo "Hello, Docker!";
phpinfo();
?>
```

Step 2: Write the Dockerfile

Create a Dockerfile in a VSCode inside your project folder and add the following content:

```
# Use the official PHP Apache base image
FROM php:7.4-apache

# Copy your HTML files into the container
COPY ./html/ /var/www/html/

# Set the working directory to Apache's root directory
WORKDIR /var/www/html/

# Expose port 80
EXPOSE 80
```

Dockerfile

- **FROM php:7.4-apache**: Uses the official PHP image with Apache installed.
- **COPY ./html /var/www/html/**: Copies your project's HTML files into Apache's root directory in the container.
- **WORKDIR /var/www/html/**: Sets the working directory.
- **EXPOSE 80**: Exposes port 80, which Apache uses for HTTP.

Step 3: Add a Sample PHP File

3. Open a terminal in your project directory and build your Docker image:

```
docker build -t php-apache-app .
```

4. Docker will go through each line of your Dockerfile and build the image. You should see messages indicating each step as it completes.

Step 5: Run the Docker Container

Start the container with the following command:

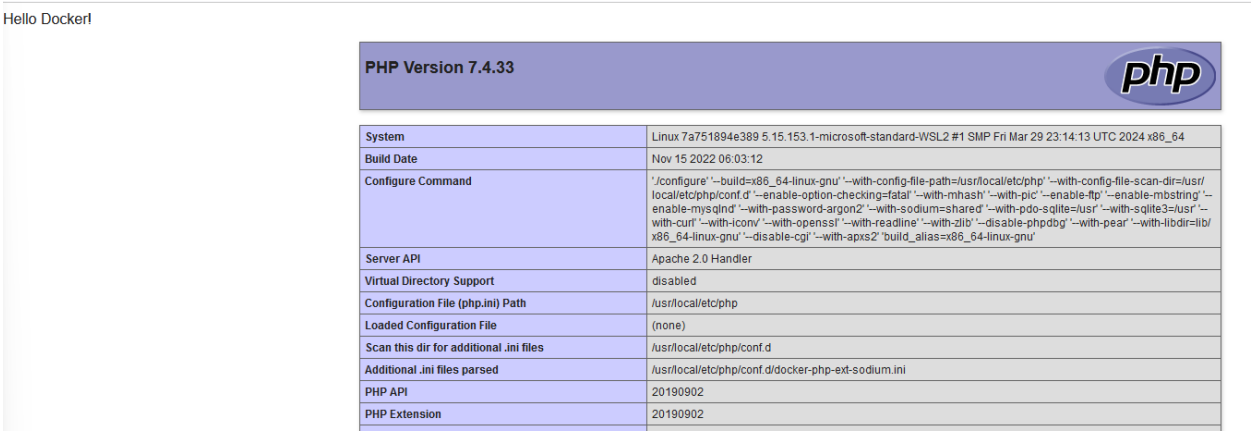
```
docker run -d -p 8080:80 --name my-php-app php-apache-app
```

This command does the following:

- **-d**: Runs the container in detached mode (in the background).
- **-p 8080:80**: Maps port 8080 on your local machine to port 80 in the container.
- **--name my-php-app**: Names the container my-php-app.
- **php-apache-app**: Specifies the image to use.

Step 6: Access Your Web Application

5. Open a web browser and go to http://localhost:8080. You should see the "Hello, Docker!" message along with the PHP info page.



Volume

Volume is a storage mechanism that allows you to persist and manage data generated or used by Docker containers. Unlike the data stored inside a container, which is deleted when the container is removed, volumes are designed to **persist data beyond the lifecycle of a container**.

Mapping A Volume

Mapping a volume means linking a directory from your host machine to a directory inside the container. This allows the container to access and save data directly to the host's filesystem, which is useful for persisting data, sharing files across containers, or **live development**.

To map a volume from your host system (e.g., **C:\Docker Training\Projects\<Directory change to desired directory path /name>\html**) to the Docker container, modify the docker run command to include a volume mapping.

Example path: C:\Docker Training\Projects\PHP_Apache_WebServer\html

Step 1: Update the docker run Command

1. When starting the container, map the volume by using the **-v** option:

```
docker run -d -p 8080:80 --name my-php-app -v "C:\Docker Training\Projects\PHP_Apache_WebServer\html":/var/www/html php-apache-app
```

Explanation of the Command

- **-v "C:\Docker Training\Projects\PHP_Apache_WebServer\html":/var/www/html**: Maps the C:\Docker Training\Projects\PHP_Apache_WebServer\html directory on your Windows host to the /var/www/html directory in the container, which is Apache's default document root.

With this setup, any changes made to files in **C:\Docker raining\Projects\PHP_Apache_WebServer\html** will reflect immediately in the running container. This setup is beneficial for live development, as you can edit files on your host system and see updates in the container without rebuilding the image.

Step 2: Test the Volume Mapping

1. Once you've run the command, navigate to http://localhost:8080 in your browser.
2. Modify a file in C:\Docker Training\Projects\PHP_Apache_WebServer\html, and refresh the browser to see the changes live.