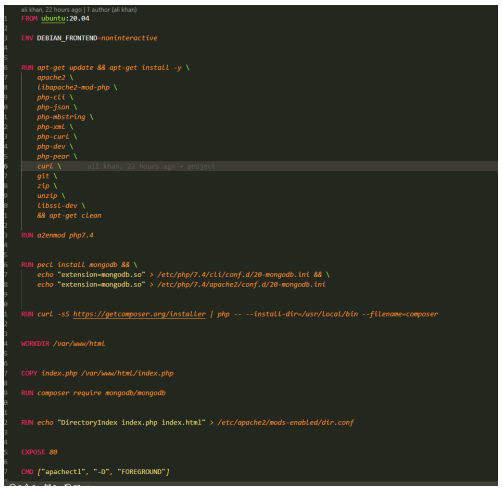
Watch Database Project- Final

For this part of the project I added on to what I did in the midterm. For the midterm I built a basic PHP container from a base Ubuntu container. That required only two files; a Dockerfile and an index.php to verify that it works. For the final deliverables, I added a database container that connected to my PHP container. I used a MongoDB image. In this project I wanted to use my PHP front end to search a DB of watches. To achieve this I needed to add 5 more files, and edit my Dockerfile with most of the work being done in the .yml file and there. I also changed my index.php to be able to search for watches by brand or model. Building this project and integrating both containers involved using Docker Compose, which was perfect to deploy this application.

Steps

1. Edit Dockerfile
    a. Added more tools and dependencies such as git, zip, and unzip. They helped helped with downloading libraries and working with compressed files,
    b. Installed composer which helps with installing MongoDB driver
    c. Enabled mongodb to work with PHP CLI and in apache with
    “RUN pecl install mongodb && \ echo "extension=mongodb.so" > /etc/php/7.4/cli/conf.d/20-mongodb.ini && \ echo "extension=mongodb.so" > /etc/php/7.4/apache2/conf.d/20-mongodb.ini”



```
ali khan, 22 hours ago | 1 author (ali khan)
FROM ubuntu:20.04

ENV DEBIAN_FRONTEND=noninteractive


RUN apt-get update && apt-get install -y \
    apache2 \
    libapache2-mod-php \
    php-cli \
    php-json \
    php-mbstring \
    php-xml \
    php-curl \
    php-dev \
    php-pear \
    curl \      ali khan, 22 hours ago • project
    git \
    zip \
    unzip \
    libssl-dev \
    && apt-get clean

RUN a2enmod php7.4


RUN pecl install mongodb && \
    echo "extension=mongodb.so" > /etc/php/7.4/cli/conf.d/20-mongodb.ini && \
    echo "extension=mongodb.so" > /etc/php/7.4/apache2/conf.d/20-mongodb.ini


RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer


WORKDIR /var/www/html


COPY index.php /var/www/html/index.php

RUN composer require mongodb/mongodb


RUN echo "DirectoryIndex index.php index.html" > /etc/apache2/mods-enabled/dir.conf


EXPOSE 80

CMD ["apachectl", "-D", "FOREGROUND"]
```

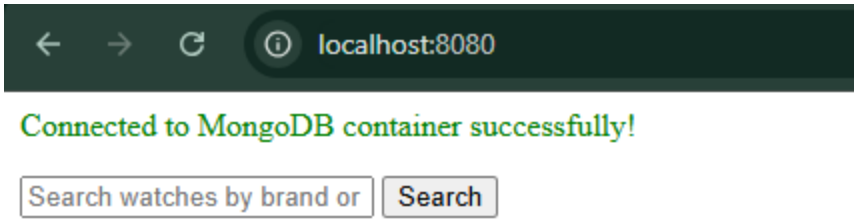
2. Added a .yml file, this file automates the deployment for my multi-container environment. It defines two “services”, one being my php container and the other being the MongoDB container. The watches-php container depends on the mongodb container, meaning mongoDB is up and running before starting the php. The volumes field syncs the index.php file from the host to the container for real-time changes. The “mongodb” service has an “image” field, whereas the watches-php doesn’t, because that container is built from the Dockerfile in the current directory, see the “build” keyword. When the mongodb container is built, it is using the image from Dockerhub, see the “image” keyword. It is using credentials from an .env file and. The mongodb exposes MongoDB the default port 27017 for local access. Under “volumes”, mongo-init.js is used to populate the database with sample data on startup, “mongo-data:/data/db” stores MongoDB data in a Docker-managed volume for data persistence. Both services connect to “app_network” to communicate with each other. They use a “bridge” network, which keeps them isolated and ensures secure communication between both containers.

```yaml
ali khan, 22 hours ago | 1 author (ali khan)
version: '3.8'

services:
  watches-php:
    build: .
    container_name: watches-php
    ports:
      - "8080:80"
    depends_on:
      - mongodb
    volumes:
      - ./index.php:/var/www/html/index.php
    networks:
      - app-network

  mongodb:
    image: mongo:6.0
    container_name: mongodb
    restart: always
    environment:
      MONGO_INITDB_ROOT_USERNAME: ${MONGO_INITDB_ROOT_USERNAME}
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_INITDB_ROOT_PASSWORD}
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db      ali khan, 22 hours ago • project
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
    networks:
      - app-network

volumes:
  mongo-data:

networks:
  app-network:
    driver: bridge
```

3. I edited my index.php to add a searching functionality which would search the database (made in mongo-init.js)

```
You, 23 hours ago | 2 authors (ali khan and one other)
db = db.getSiblingDB('watchdb');

db.watches.insertMany([
    { brand: "Rolex", model: "Daytona", price: 25000 },
    { brand: "Omega", model: "Speedmaster", price: 5000 },
    { brand: "Tag Heuer", model: "Carrera", price: 3500 },
    { brand: "Seiko", model: "Presage", price: 800 },
    { brand: "Casio", model: "G-Shock", price: 200 }
]);
```

4. To build the images and start the container, I used the “docker-compose up --build” many times, this rebuilds the Docker images for all services defined in the docker-compose.yml. To see my app running, I navigate to http://localhost:8080.



To stop and remove containers, I would type in docker-compose down in the terminal.

Maintenance Checklist:

- Monitor CPU, memory, and disk usage.
- Apply OS and package updates regularly.
- Back up MongoDB data using mongodump.