DEVOPS HANDSON_1

by Kenzy Tamer Ahmed

Table of Contents

1. Aim:

The aim of this task is to **demonstrate the end-to-end process of building a local CI/CD pipeline** for a real-world application. The project focuses on automating the **build, test, and deployment workflow** of the TeamAvail application, a team availability tracker

2. Project Tasks:

2.1 Clone Repository:

First I make a fork from the original repo and then clone the forked repo

git clone https://github.com/KenzySobh/TeamavailTest.git

cd TeamavailTest

2.2 Dockerize The App:

By making Dockerfile

TeamavailTest > Dockerfile

```
# Use Debian-based Node image
FROM node:20-bullseye

# Set working directory
WORKDIR /app

# Copy package.json first (for caching)
COPY package*.json ./

# Install dependencies
RUN npm install --production

# Copy app code
COPY . .

# Use non-root user
RUN useradd -m appuser && chown -R appuser:appuser /app
USER appuser

# Expose port
EXPOSE 3000

# Start the app
CMD ["node", "server.js"]

```

File Explanation:

1. Base Image

- Specifies the **base image** for the Docker image.

---

2. Set Working Directory

- Sets the working directory **inside the container** to /app.
- All subsequent commands (COPY, RUN, CMD) are executed relative to this directory.

---

3. Copy Package Files for Caching

- Copies package.json and package-lock.json into the container.
- Done **before copying the full project code** to leverage Docker's **layer caching**:

---

4. Install Dependencies

- Installs Node.js dependencies required for running the app.
- The --production flag ensures only **runtime dependencies** are installed, excluding development dependencies.

---

5. Copy Application Code

- Copies all remaining project files into the container.
- Includes application source code, JSON files, and static assets.

---

6. Create and Use Non-Root User

- Creates a **non-root user** named appuser for security purposes.
- Changes ownership of the /app directory to appuser.
- All subsequent container operations, including starting the app, run as appuser.

---

7. Expose Application Port

- Documents that the application listens on **port 3000**.
- Helps Docker Compose or other orchestration tools to map ports.

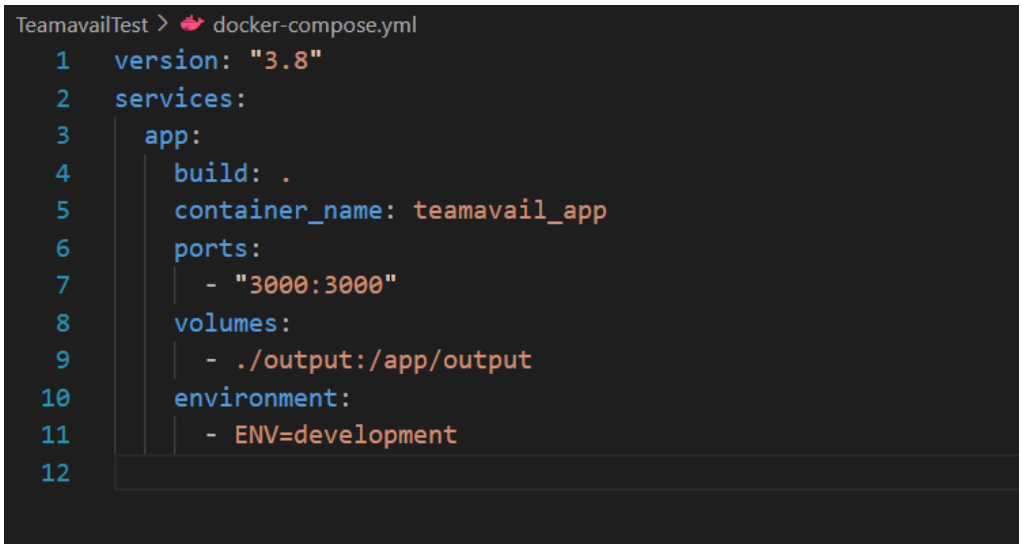
---

8. Start the Application

- Defines the **default command** to run when the container starts.
- Starts the Node.js server using server.js.

2.3 Docker Compose File:

By making docker-compose.yml



File Explanation:

1. **Build Context**

- Instructs Docker Compose to **build the image** for the app service using the Dockerfile in the current directory (.).
- This allows integration with the Dockerfile workflow for CI/CD and ensures the latest application code is included in the container image.

---

2. **Container Name**

- Assigns a **custom name** to the container.

---

3. **Port Mapping**

- Maps **port 3000** on the host machine to **port 3000** inside the container.
- Enables access to the TeamAvail application from the host system at http://localhost:3000.

---

4. **Volume Mounting**

- Mounts the ./output folder on the host machine to /app/output inside the container.
- Allows **persistent storage** of generated files such as history.json.
- Ensures data is **not lost** when the container is stopped or rebuilt.
- Supports easy inspection and backup of output files from the host system.

5. **Environment Variables**

- Sets environment variables inside the container.
- ENV=development indicates that the application is running in a **development mode**, which can be used for configuration purposes.

2.4 Build Ci/CD Script:

file The ci.sh script is a **local CI/CD automation script** for the TeamAvail application. Its main purpose is to automate the build and deployment process using Docker and Docker Compose

TeamavailTest > $ ci.sh

```
#!/usr/bin/env bash
set -euo pipefail

echo "=== CI: build Docker image ==="
docker build -t teamavail:latest .

echo "=== CI: docker compose up ==="
docker compose up -d --build

echo "=== CI: done. Application should be up ==="
docker ps --filter "ancestor=teamavail:latest"
echo "Access the application at http://localhost:3000"
echo "=== CI finished ==="
```

File Explanation:

1. **Build Docker Image**
   - docker build -t teamavail:latest . builds a Docker image using the Dockerfile in the current directory (.).
   - -t teamavail:latest tags the image as teamavail with the latest tag.
   - This step ensures the **latest application code** is packaged in a container.

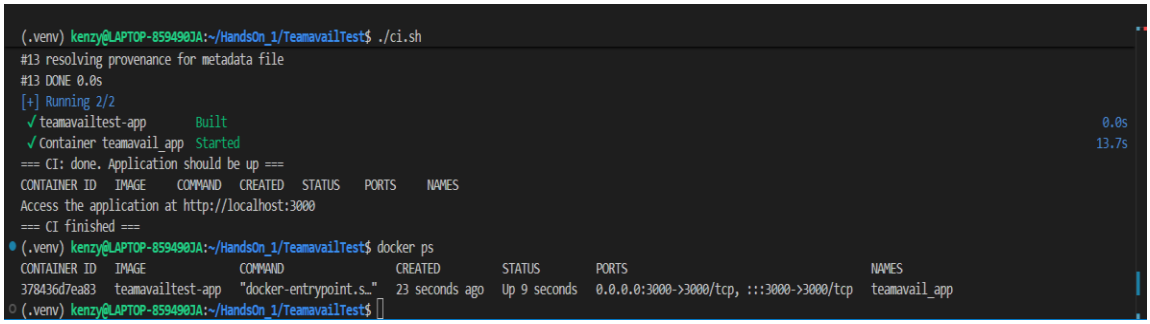
---

2. **Start Application Using Docker Compose**
   - docker compose up -d --build does the following:
     - up → Starts the services defined in docker-compose.yml.
     - -d → Runs containers **in detached mode**, allowing the terminal to be free for other commands.
     - --build → Rebuilds the images before starting containers, ensuring **any changes in code or dependencies are included**.
   - This automates running the full application stack, including any volumes, environment variables, or additional services defined in docker-compose.yml.

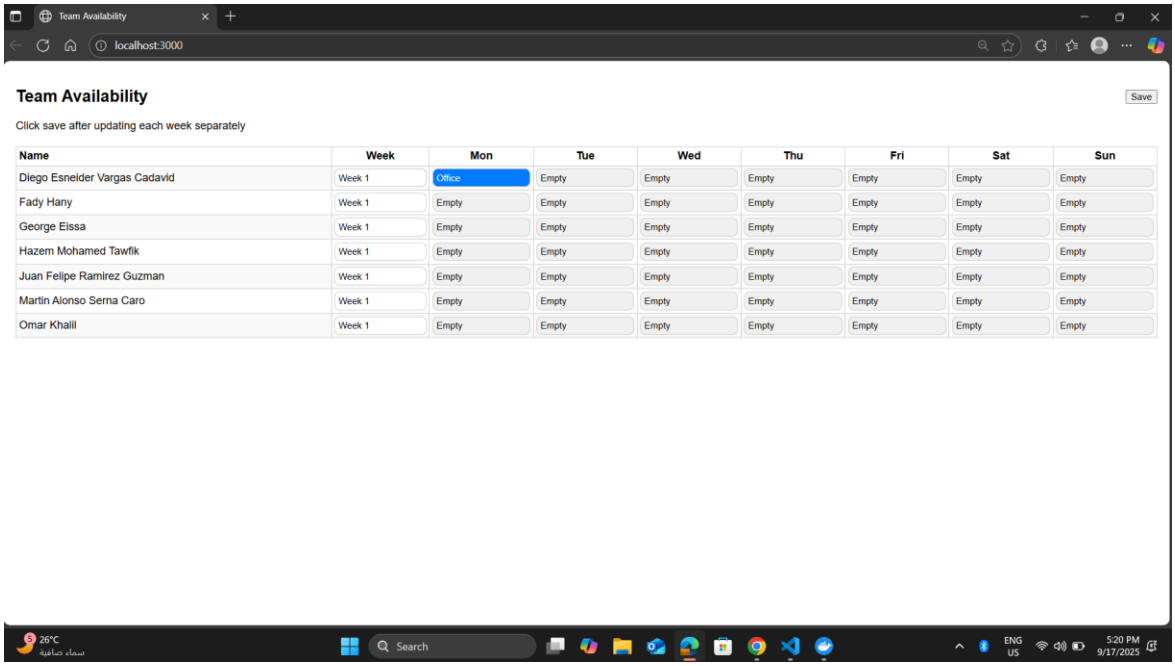
---

3. **Display Running Containers**
   - docker ps --filter "ancestor=teamavail:latest" lists all running containers that were created from the teamavail:latest image.
   - This provides a **quick check** to verify that the application container is up and running.

2.5 Validating:



Docker image is built successfully using ./ci.sh command



The Application is running on local host 3000.