DevOps Assignment Report by Makkena Bharath Kumar (21ucc061)

The main goal of this assignment was to set up, troubleshoot, and deploy a Dockerized web application, making sure it runs without errors and can be accessed through a browser. The tasks involved building Docker images, fixing mistakes in the setup, and confirming the application was successfully deployed.

Issues Faced and How They Were Solved

1. Errors in Python Application's Dockerfile

- **Problem:** The Dockerfile for the Python app had issues that prevented it from building and running. Error messages pointed out missing dependencies and setup problems for the Flask application.
- **Solution:** Fixed the Dockerfile by:
    - Installing required dependencies using RUN pip install -r requirements.txt.
    - Setting the correct FLASK_APP environment variable.
    - Exposing port 5000 for the Flask app.

2. Nginx Setup Issue

- **Problem:** Nginx was not correctly set up to connect with the Python app container, leading to a "502 Bad Gateway" error.
- **Solution:** Adjusted the Nginx configuration to link it to the correct host (python_app) and port (5000). Reloading the configuration fixed the problem.

3. Port Numbers Written as Words in docker-compose.yml

- **Problem:** In docker-compose.yml, ports were written out as words (like eighty: eighty instead of 80:80), which Docker Compose didn't recognize.
- **Solution:** Changed word-format ports to numbers (e.g., eighty: eighty to 80:80). After this, Docker Compose was able to read and apply the ports properly.

4. Port Conflicts Between Services

- **Problem:** Multiple services were trying to use the same port, causing a conflict.
- **Solution:** Updated docker-compose.yml to give each service a unique port. Set Nginx to port 80 and the Python app to port 5000.

5. Database Connection Issues

- **Problem:** The Python app couldn't connect to the database due to an incorrect database URL.
- **Solution:** Fixed the database URL in docker-compose.yml to match the database container name, allowing the app to connect successfully.

6. App Starting Too Soon for Database

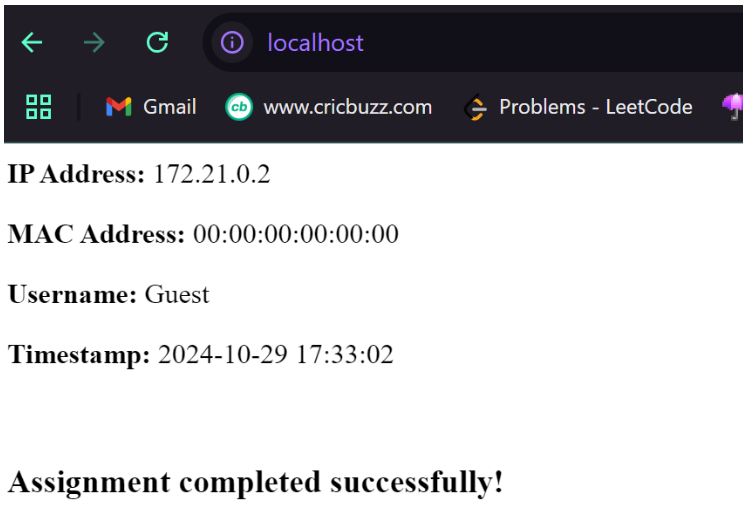
- **Problem:** The app tried to connect to the database before it was ready, causing errors.
- **Solution:** Added a retry mechanism in the app code to keep trying until the database was available.

7. Permission Issues with Nginx Logs

- **Problem:** Nginx couldn't write to its log files due to permission issues, so no requests were logged.
- **Solution:** Updated the log directory permissions and mapped it to a host directory so logs could be accessed and checked.

Testing and Verification

1. **Local Testing:**
    - After fixing all issues, ran the containers with docker-compose up --build.
    - Accessed the app in the browser at http://localhost to make sure it was working.
    - Checked the Nginx logs to confirm successful requests.
2. **Screenshots:**
    - Took screenshots of the app running in the browser and of the Nginx logs showing successful requests.



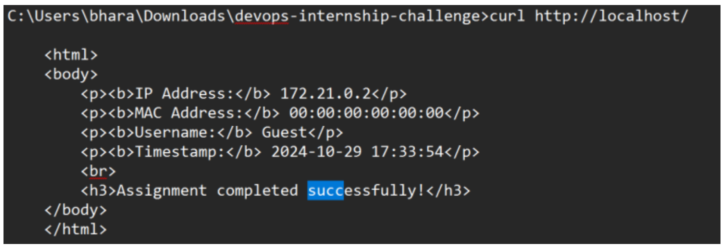
IP Address: 172.21.0.2

MAC Address: 00:00:00:00:00:00

Username: Guest

Timestamp: 2024-10-29 17:33:02

Assignment completed successfully!



```
C:\Users\bhara\Downloads\devops-internship-challenge>curl http://localhost/

    <html>
    <body>
        <p><b>IP Address:</b> 172.21.0.2</p>
        <p><b>MAC Address:</b> 00:00:00:00:00:00</p>
        <p><b>Username:</b> Guest</p>
        <p><b>Timestamp:</b> 2024-10-29 17:33:54</p>
        <br>
        <h3>Assignment completed successfully!</h3>
    </body>
    </html>
```

Summary

This assignment gave practical experience in finding and fixing common errors in Dockerized applications. Some of the main issues included fixing the docker-compose.yml file, dealing with database connections, and resolving permission problems. After fixing each issue, the application was able to run successfully.