Journal for Docker Challenges 3 and 4

Student Name: Prit Patel

Date: 24-04-2024

Course: Winter 2024 Operating Systems (CPSY-300-E)

Introduction

This journal documents the technical journey and learnings from participating in the second part of a Docker challenge series. This series is designed to enhance students' skills in using Docker, particularly how it applies to operating systems and cloud computing disciplines.

Challenge 3: Full Stack Application

Objective

To integrate a three-tier application using Docker Compose, consisting of a web server (nginx), an application server (Node.js), and a database (MariaDB).

Environment Setup

- **Tools Installed:** Git, Docker, and a GitHub account.
- **Initial Setup:** Forked and cloned the required repository, ensuring all initial files were included in the **challenge3** folder.

Configuration

- **Environment Variables:**
    - Created a **.env** file to hold sensitive configuration settings such as database credentials.
    - Variables included: **DB_ROOT_PASSWORD**, **DB_DATABASE**, **DB_USERNAME**, **DB_PASSWORD**, and **DB_HOST**.
- **Docker Compose Setup:**
    - Configured **docker-compose.yml** to run three services:
        1. **nginx**: Configured to route traffic to the Node.js application.

2. **node-service**: Set up to handle backend logic and communicate with the database.
3. **db**: A MariaDB database initialized with schemas and data essential for the application.

Execution and Verification

- Ran **docker-compose up** to start all services.
- Accessed **http://localhost:8080/api/books** to fetch a JSON message with all books, verifying the application's functionality.
- Accessed **http://localhost:8080/api/books/1** to fetch a JSON message with a single book, ensuring detailed data retrieval was functional.

Troubleshooting

- Encountered issues with database connections initially due to misconfigured environment variables. Adjustments were made in the **.env** file, followed by restarting the Docker services.

Challenge 4: Scaling up an Application

Objective

To learn about and implement scaling of the Node.js service using Docker Compose, increasing the instance count from 1 to 3.

Scaling Implementation

- Utilized the existing configuration from Challenge 3.
- Executed the command:

bash

Copy code

docker-compose up --scale node-service=3

- This command increased the number of node-service instances to three, demonstrating the ability to handle increased load.

Observations

- **Pre-Scaling:** Initially, repeated requests to **http://localhost:8080/api/stats** returned the same hostname.
- **Post-Scaling:** After scaling, different hostnames were returned upon repeated requests, indicating requests were distributed across the three instances.

Documentation and Submission

- Captured screenshots of the Docker Compose terminal output and the browser displaying different hostnames.
- Documented all steps, configurations, and results in this journal.
- Submitted the journal PDF and repository URL through the D2L platform.

Conclusion

These challenges provided practical insights into Docker's capabilities for developing, deploying, and scaling applications. They emphasized the importance of understanding inter-service connectivity and the operational benefits of scaling services within Dockerized environments.

References

- Docker Official Documentation
- YouTube tutorials on Docker and Docker Compose
- Apps Developer Blog on Docker scaling