(Senior Academy - IT training center)

The Place You Can Be A Senior

www.seniorsteps.net
https://www.facebook.com/seniorsteps.it
contact us: 0224153419 - 01090873748

فرع مدينة نصر 1 : عمارة 4 – شارع محمد توفيق دياب – عباس العقاد - مدينة نصر – الدورال 1

Senior Academy - IT training center
www.seniorsteps.net
contact us: 0224153419 - 01090873748
عمارة 4 – شارع محمد توفيق دياب – عباس العقاد - مدينة نصر – الدورال 1

DevOps Engineer Diploma

DevOps Engineer Diploma

Containers With Docker

Lab 02

Dockerizing a Flask Microservice and Local Development Environment Setup

Lab Objectives

- **Flask Microservice Dockerization**
- **Service Connectivity and Isolation**
- **Data Persistence**
- **NGINX Frontend Exposure**

Task 1

❖ Dockerizing a simple Python Flask microservice:

your team is migrating a monolithic application to a Microservices Architecture. You are tasked with preparing a small, lightweight Flask application—which serves as a basic health endpoint (/) — for production deployment. This container must be secure, efficient, and ready to be consumed by orchestration tools like Kubernetes.

Source code : https://github.com/abdelrahmanonline4/EFE-Labs-/tree/Master/Lab2

 Required Deliverables (Submission Files)

1 .app.py: The core Python Flask code.

2 requirements.txt: The list of Python dependencies.

3 Dockerfile: The instructions for building the Docker image.

A. Code (app.py & requirements.txt)

 The Flask application must run on port 5000.

 The application must be configured to listen on 0.0.0.0 to be accessible from outside the container (critical for Docker networking).

 The requirements.txt must clearly define the necessary Flask dependency.

The Dockerfile must meet these optimization and clarity standards:

1. Base Image: Use an official, minimized base image (e.g., python:3.11-slim).

2. Caching Strategy: Utilize the Docker Build Cache efficiently by copying requirements.txt and installing dependencies before copying the main application code.

3. Working Directory /app

4. Port 5000

5. Execution: Use the CMD instruction to define the final command that starts the Flask application.

✓ Verification Steps (What to Submit)

Students must provide the commands and output for the following steps to prove the container is operational:

1. Build Command: The command used to successfully build the Docker image (e.g.,

docker build...).

2. Run Command: The command used to run the container, ensuring Port 8080 on the host is mapped to Port 5000 inside the container.

3. Verification Output: The output of running a curl command against the mapped host port (e.g., curl http://localhost:8080) showing the successful response from the Flask application.

Task 2

You are setting up a local development environment for a new web application that uses NGINX as the public-facing server and MySQL as its backend database. Your critical tasks are:

1 : Service Connectivity: Ensure the two services (nginx and mysql_db) are running on a private, isolated network (app-net) and can communicate using their service names.

2:Data Persistence: Ensure all data generated by the MySQL database is stored permanently on the host machine, independent of the database container's lifecycle.

3: Frontend Access: Expose the NGINX service to the outside world so developers can access it locally.

Execution and Verification Steps (What to do)

1. Deploy the Stack: Run the Docker Compose file to start the services.

2. Check if NGINX is serving its default page.

3. Ping the database service name from the NGINX container to prove network connectivity inside the app-net bridge.