Report on Dockerizing and Monitoring FastAPI Application for MNIST Digit Prediction

Name - Siddhesh Gatkal
Roll - MM20B019
Link - https://github.com/Sid25072002/CS5830_Assignment7.git
Drive -
https://drive.google.com/drive/folders/1SJIQj8g85I9BWDvoh6kZscZP-dwnPNH_?usp=drive_link

Introduction

The objective of this project is to enhance an existing FastAPI application for MNIST digit prediction by adding health monitoring capabilities and containerizing it using Docker. The steps will involve integrating Prometheus for monitoring, visualizing metrics with Grafana, and deploying the application in a Dockerized environment. This report outlines the step-by-step process to achieve these goals.

Task 1: Monitoring with Prometheus and Grafana

1. Integrate Prometheus with FastAPI
- **Setup Prometheus:** Install and configure Prometheus on your machine. Prometheus will collect and store metrics data.
- **Add Monitoring Hooks:**
  - **Counters:** Track API usage from different client IP addresses.
  - **Gauges:** Monitor the processing time relative to the input size. Metrics to be captured include input length, total time taken, and processing time per character (T/L), along with client IP details.

2. Setup Grafana for Visualization
- **Install Grafana:** Install and configure Grafana to visualize the metrics collected by Prometheus.
- **Create Dashboards:** Configure Grafana dashboards to display:
  - API usage counters.
  - API run time and T/L time.
  - API memory and CPU utilization.
  - API network I/O bytes and rate.

3. Testing
- Use tools like Swagger UI, Postman, or curl from different machines to test the API and ensure metrics are collected accurately.

Task 2: Dockerizing the Application

1. Setup Docker Environment
- **Install Docker:** Ensure Docker is installed and running on your machine.
- **Start Docker Services:** Initialize Docker services to prepare for containerization.

2. Dockerize the FastAPI Application
- **Create Dockerfile:**
  - Define the base image (e.g., `python:3.9-slim`).
  - Copy the FastAPI application code into the image.
  - Install necessary dependencies.
  - Expose the required ports for FastAPI and Prometheus.
- **Build Docker Image:** Use the Dockerfile to build the Docker image of the FastAPI application.
- **Run Docker Container:** Launch the Docker container and verify that Task 1 functionalities work seamlessly.

3. Configure Resource Limits and Scaling
- **CPU Utilization Limit:** Set a limit of 1 CPU for the Docker container using command line options during container startup.
- **Scale Containers:** Deploy multiple instances of the Docker container, adjusting port mappings to avoid conflicts.
- **Cluster Monitoring:** Configure Prometheus and Grafana to monitor the cluster, ensuring all metrics are tracked across multiple instances.

Conclusion

By following this structured approach, we will successfully enhance the FastAPI application with robust monitoring capabilities and deploy it in a scalable, containerized environment. This project not only demonstrates technical proficiency in working with FastAPI, Docker, Prometheus, and Grafana but also emphasizes best practices in coding, testing, and project management.
