BMS COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

LAB CYCLE – AY2024-2025 [ODD]

Assignment Assessment -3

Posted Date: 21st Dec 2024, 6 PM

Submit Date: 28th Dec 2024, 11 PM

COURSE: DevOps	**SECTION: A, B, C & D**
COURSE CODE: 21IS7PEDVR	**TOTAL CREDITS: 10 Marks**

Demonstrate a working solution on your laptop or on a lab system and share screenshots. Use any internet resources available (Google, GPT)

Contents

Problem Statement

Imagine **Visual Craft** is an innovative AI-powered art platform aimed at transforming digital images into artistic images using an AI artistic style transfer technique.

As a DevOps Engineer at **Visual Craft** your role is to apply DevOps principles to deploy, scale and manage the AI Artistic Style Service in a production-ready environment. This project challenges you to containerize, orchestrate and monitor the application, ensuring reliability and performance.

Use Case

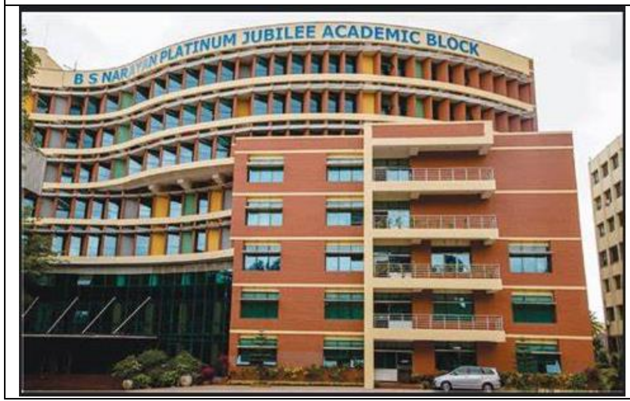
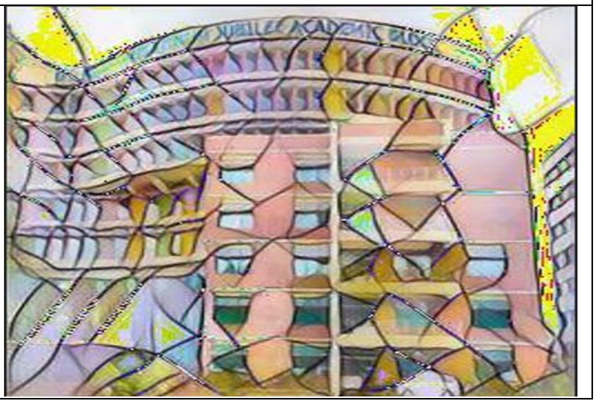
The AI Artistic Style Service enables users to transform any given image to an artistic image. Here are steps involved:
1. Pull "AI Artistic Style" docker Image from Docker hub:
https://hub.docker.com/r/urmsandeep/ai-artistic-style-service

2. Run the docker image by providing an input image (JPG) of your choice

3. The AI model generates an output image transformed into an artistic system.

Example

Here's an example where an input image (bmsce.jpg) is submitted to an AI Artistic Style Service (running in a docker) and the Output image (output.jpg) is generated by the service.

Input Image **to AI Artistic Style Service** **(bmsce.jpg)**	**Output Image** **from to AI Artistic Style Service** **(output.jpg)**
	

Task

Your task is to integrate DevOps tools and practices to:
- Automate deployments.
- Ensure high availability.
- Monitor and optimize performance.
- Implement secure operations.

You will use the following Docker Hub image for the AI Artistic Style Service:
`urmsandeep/ai-artistic-style-service`

Assignment Requirements

1. Dockerized AI Artistic Style Service:
    a. Use the provided Docker image to deploy the AI service running on port 5001
    b. Expose the service on port 5001 for API calls.
2. API Testing:
    a. Test the /styleTransfer endpoint using curl or Postman.
    b. Input: A sample JPEG image.
    c. Output: A stylized version of the image.
3. Container Orchestration:
    a. Use Docker Compose (**docker-compose.yml**) to deploy the AI service docker.
    b. Configure the container to restart automatically on failures.
4. Monitoring:
    a. Integrate Prometheus and Grafana to monitor API usage and resource metrics.
    b. **Number of Requests**:
        i. Monitor the total number of API requests made to the `/styleTransfer` endpoint.
        ii. Group requests by status code (`200`, `400`, `500`) to analyze success and error rates.
    c. **Request Rate**:
        i. Measure the rate of incoming requests per second or minute.
        ii. This helps identify traffic patterns and detect sudden spikes.
    d. **Response Time (Latency)**:
        i. Track the time taken by the API to respond to requests.
5. CI/CD Pipeline:
    - Automate the deployment of the AI service using a CI/CD tool like GitHub Actions or Jenkins.
    - Ensure tests run before deployment to validate service functionality.

Deliverables and Deployment

- **Deployment**
    - Pull the Docker image using **docker pull urmsandeep/ai-artistic-style-service**
    - Create a docker-compose.yml file to run the service on port 5001.
    - Test the /styleTransfer API endpoint with a sample image. Refer to Section below on how to run the AI Artistic Image Service {**How** to Run the AI Artistic Style Service docker image]
    - Expected Outcome: The service processes the image and returns a stylized version,
- **API Monitoring**
    - Integrate Prometheus to collect metrics.
    - Visualize metrics using Grafana dashboards.
    - Track the number of API calls and response times
- **Resilience Testing**
    - Objective: Verify service availability during container failures:
    - Simulate container failures using Docker Compose.
    - Verify that the service restarts automatically
    - Expected Outcome: The service remains functional and self-recovers
- **CI/CD Pipeline Automation**
    - Objective: Automate the deployment of updates
    - Steps:
        - Set up a CI/CD pipeline using GitHub Actions or Jenkins.
        - Push changes to a repository to trigger the pipeline.
        - Verify that the service is redeployed automatically.
    - Expected Outcome: Automated testing and deployment of the service.

Submission Requirements

- Code and Configuration Files:
    - docker-compose.yml
    - CI/CD pipeline configuration (e.g. Jenkinsfile).
- API Testing Results:
    - Screenshots of API requests and responses.
    - Screenshots of Input Image and a Stylized output image. (choose your own input image)
- Monitoring Dashboards:
    - Grafana screenshots displaying API metrics. Suggested Grafana Dashboards:
      Dashboard 1: API Performance
        - Panel 1: Total API Requests
            - Visualize the total number of requests received by the /styleTransfer endpoint.

- Panel 2: CPU Usage
  - Show CPU usage percentage over time.
- Panel 3: Memory Usage
  - Show memory consumption trends.

How to Run the AI Artistic Style Service docker image

Here's step on how to run the AI Artistic Style Service. In this example, I have an image called **bmsce.jpg**. Steps:

1. Pull the docker image **`urmsandeep/ai-artistic-style-service`**
2. Run docker image **`urmsandeep/ai-artistic-style-service`**
3. Verify if docker image is running successfully.
4. Use curl to invoke the service on port 5001, giving path to input image "bmsce.jpg'
5. Verify if the output.jpg is created, which is an transformed artistic image.

```
docker pull urmsandeep/ai-artistic-style-service

Using default tag: latest
latest: Pulling from urmsandeep/ai-artistic-style-service
bc0965b23a04: Pull complete
9b871d410cbf: Pull complete
8bfa778b5b23: Pull complete
258b25b92655: Pull complete
8f839e8d10a8: Pull complete
76c3a07be88c: Pull complete
ae9ac8307df2: Pull complete
Digest: sha256:b9c40b3bda1d4eea99ec4adb195d73d40ea2ddfd6b9ee662b69438a1b7da4756
Status: Downloaded newer image for urmsandeep/ai-artistic-style-service:latest
docker.io/urmsandeep/ai-artistic-style-service:latest
```

```
docker run -d -p 5001:5001 urmsandeep/ai-artistic-style-service

c0d78eafb9f813df76e7c03d1a40f59646ab5b65987bcb584770b04f7e110b9c
```

```
docker ps -a | grep ai-artistic-style-service

c0d78eafb9f8   urmsandeep/ai-artistic-style-service   "python ai_artistic_…"   2 minutes ago
Up 2 minutes
    0.0.0.0:5001->5001/tcp, :::5001->5001/tcp
              unruffled_agnesi
```

```
curl -X POST http://127.0.0.1:5001/styleTransfer -F
"image=@bmsce.jpeg" --output styled_image.jpg
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 59829  100 25217  100 34612  89548   120k --:--:-- --:--:-- --:--:--  207k
```

ls *.jpg*

```
bmsce.jpg   # This is an example, pick you own image as Input Image
styled_image.jpg # Output Image
```

Partial Solution and Hints

docker-compose.yml

```
version: '3.8'
services:
  ai-artistic-style-service:
    image: urmsandeep/ai-artistic-style-service:latest
    container_name: ai-artistic-style
    ports:
      - "5001:5001"  # Map host port 5001 to container port 5001
    restart: always  # Ensure the container restarts automatically on
failure
```

Start service: `docker-compose up -d`
Verify service: `docker ps`
Stop service: `docker-compose down`
Check logs: `docker-compose logs`

Jenkinsfile

```
pipeline {
    agent any

    stages {
        stage('Pull Docker Image') {
            steps {
                // Pull the pre-built Docker image from Docker Hub
                script {
                    sh 'docker pull urmsandeep/ai-artistic-style-
service:latest'
                }
            }
        }

        stage('Run Tests') {
            steps {
                // Run tests to ensure the container works correctly
                script {
                    sh '''
                    docker run --rm -p 5001:5001 urmsandeep/ai-artistic-
style-service:latest pytest tests/
                    '''
                }
            }
        }

        stage('Deploy Service') {
            steps {
                // Deploy the service using Docker Compose
                script {
                    sh '''
                    docker-compose down
                    docker-compose up -d
                    '''
                }
            }
        }

        stage('Verify Deployment') {
            steps {
                // Check if the service is running and responding
                script {
                    sh '''
                    sleep 5
                    curl -X POST http://127.0.0.1:5001/styleTransfer -F
"image=@test.jpg" --output styled_output.jpg
                    '''
                }
            }
        }
    }

    post {
        always {
            // Clean up dangling Docker containers and images
            sh 'docker system prune -f'
        }
        success {
            echo 'Pipeline executed successfully. The service is running
and functional!'
        }
        failure {
            echo 'Pipeline failed. Check logs for errors.'
        }
    }
}
```

Prometheus Integration

- **Prometheus Configuration**: Add the AI service as a target in prometheus.yml:

```
scrape_configs:

  - job_name: "ai-artistic-style-service"

    static_configs:

      - targets: ["ai-artistic-style:5001"]
```

Use Exporters:

  - Use **Node Exporter** to gather system-level metrics.
  - Use **cAdvisor** to monitor container resource usage.

Grafana Setup

- Import pre-built dashboards from the Grafana Dashboard Library for Prometheus.
- Create custom dashboards based on the API and container metrics mentioned above.