Fundamentals of Distributed Systems

Assignment – Dynamic Load Balancing for a Smart Grid

Name – Arpit Tomar (G24AI2001)

Date – 25th June 2025

Github - https://github.com/arpittomar246/smart-grid-load-balancer-G24AI2001

Objective

The goal of this project is to design and implement a scalable system for a **Smart Grid** that dynamically balances Electric Vehicle (EV) charging requests across multiple substations based on their **real-time load**. The system aims to prevent overload on any individual substation while ensuring efficient energy usage and grid stability. An observability stack is integrated to monitor and visualize the performance of the system.

Technologies Used

- **Python (Flask)**: For all service implementations.
- **Docker & Docker Compose**: To containerize and orchestrate the microservices.
- **Prometheus**: For scraping metrics from substations.
- **Grafana**: For real-time dashboard visualization.

System Architecture

The system consists of the following main components:

1. Charge Request Service

- Acts as the **public entry point** for all incoming EV charging requests.
- Forwards requests to the **Load Balancer** service for intelligent routing.

2. Load Balancer Service

- Core logic of the grid system.
- Periodically polls the /metrics endpoint from each **Substation Service** to gather real-time load data.
- Routes incoming requests to the **least loaded** substation based on Prometheus-style metrics.

3. Substation Service

- Simulates EV charging behavior.
- Maintains and exposes its **current load** via a /metrics endpoint (compatible with Prometheus).

4. Load Tester

- Python script that simulates **rush hour traffic** of EVs.
- Sends numerous charging requests in quick succession to emulate peak load scenarios.

5. Observability Stack

- **Prometheus**: Scrapes real-time load metrics from substations.
- **Grafana**: Displays a live dashboard showing substation loads and system behavior during testing.

Execution Steps

1. **Navigate to the project directory**:

$ cd smart-grid-load-balancer

2. **Build and launch the entire system**:

$ docker-compose up --build

3. **Execute load tester**:

$ python3 load_tester/test.py

4. **Monitor the system**:

- Access **Grafana Dashboard** at: http://localhost:3000
- Access **Prometheus UI** at: http://localhost:9090

5. **Shutdown the system**:

$ docker-compose down

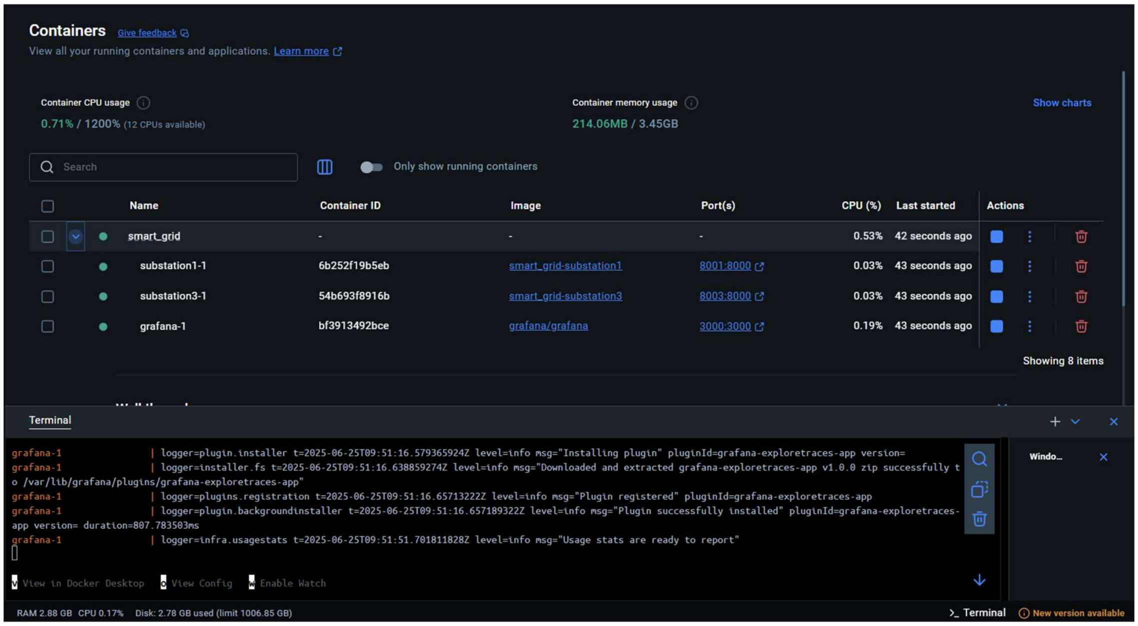
Performance Analysis

During the simulated rush hour using the load tester, Grafana visualizations showed:

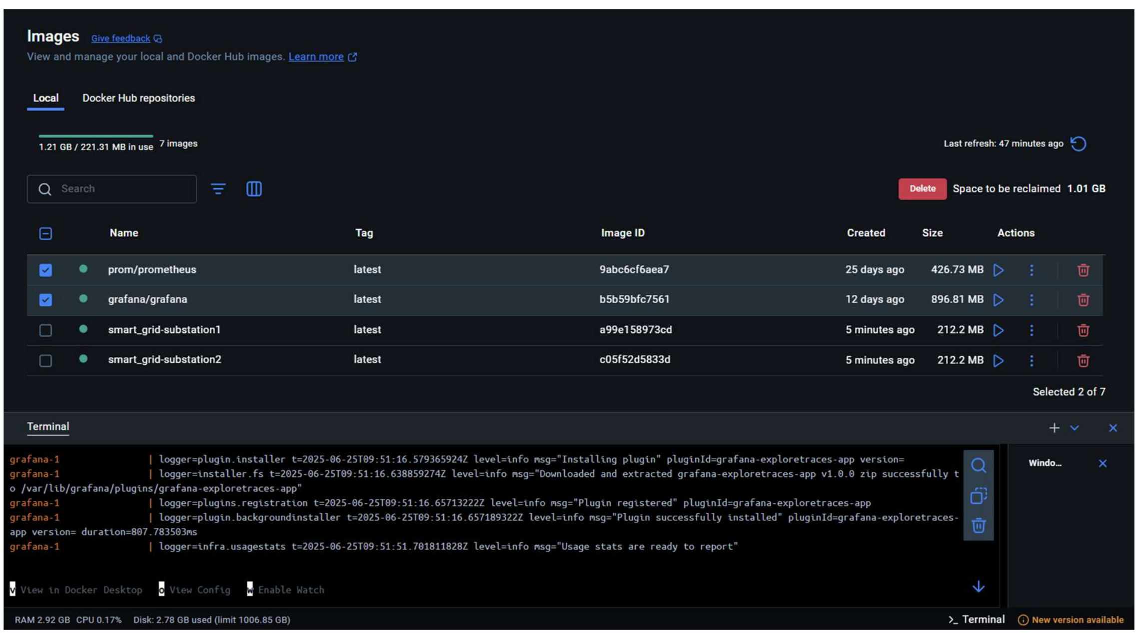
- Load balancing decisions based on real-time metrics.
- Uniform distribution of charging requests among substations.
- No individual substation was overloaded.
- Live updates of metrics and substation response times.
- Dashboard showing EV request spikes
- Load comparison graphs across substations

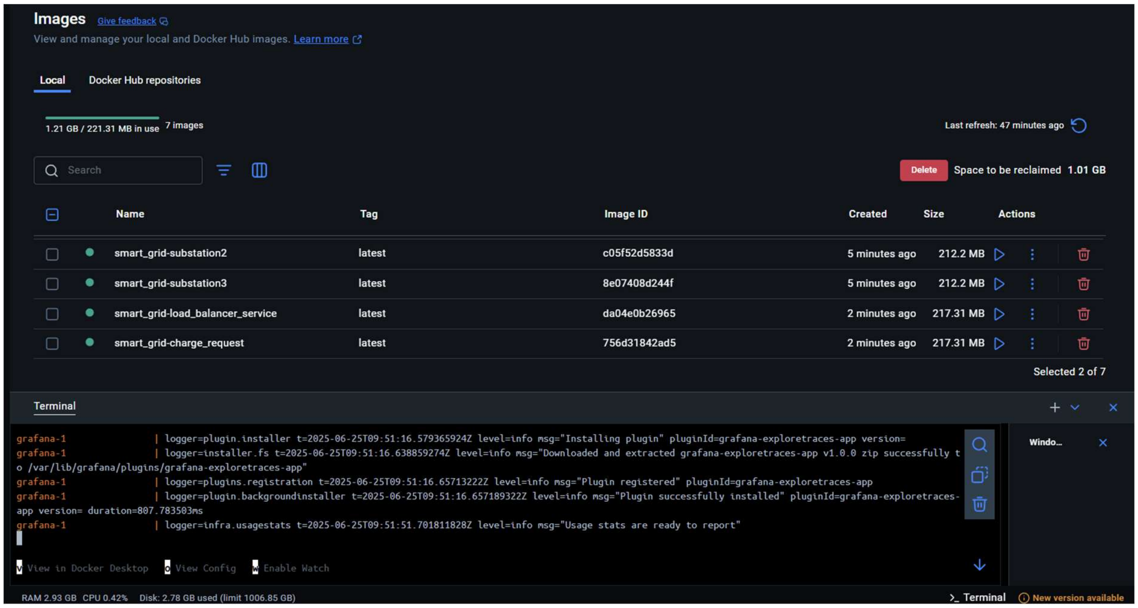
Screenshots –

Containers -

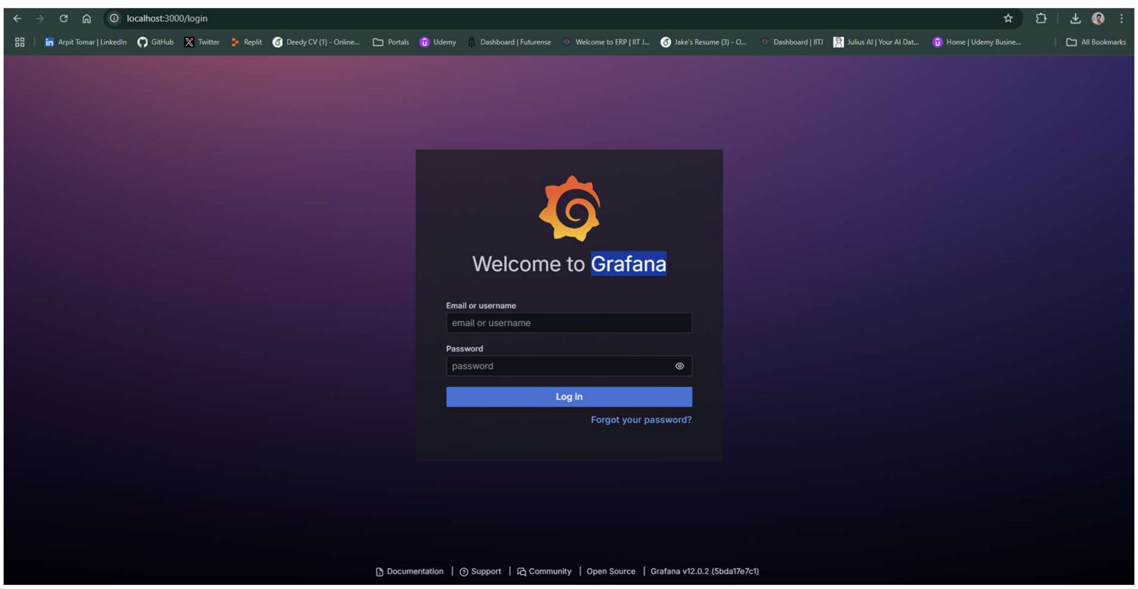


Images -

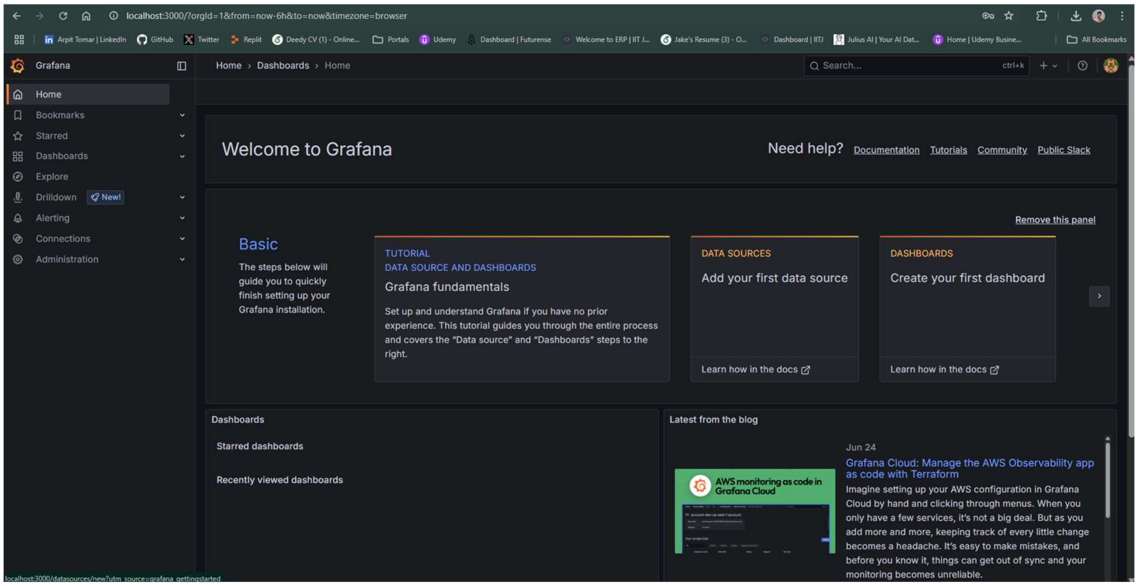




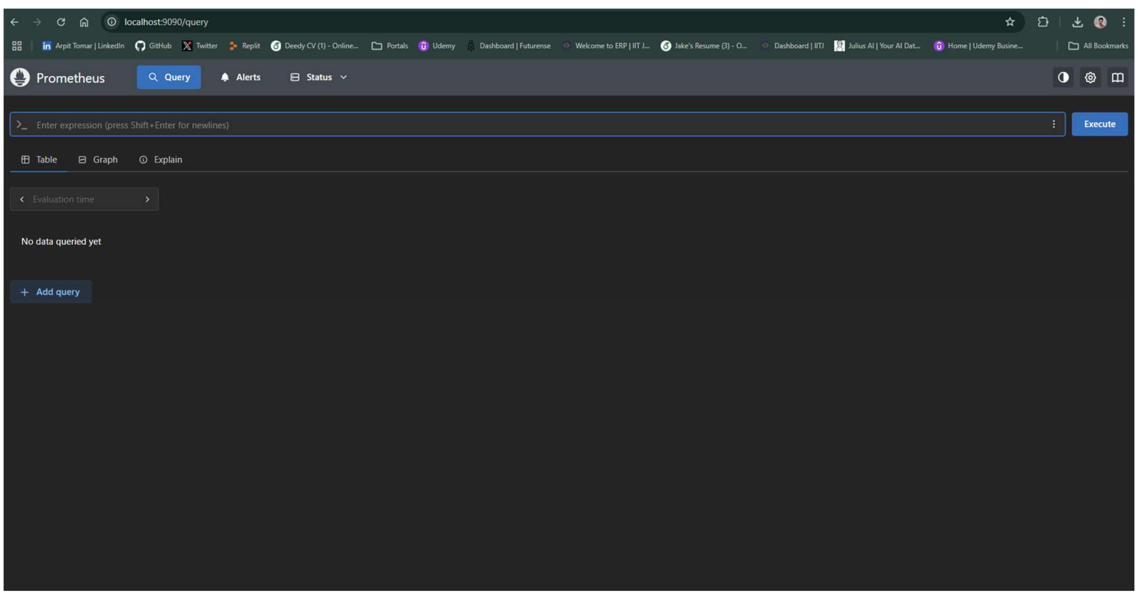
http://localhost:3000



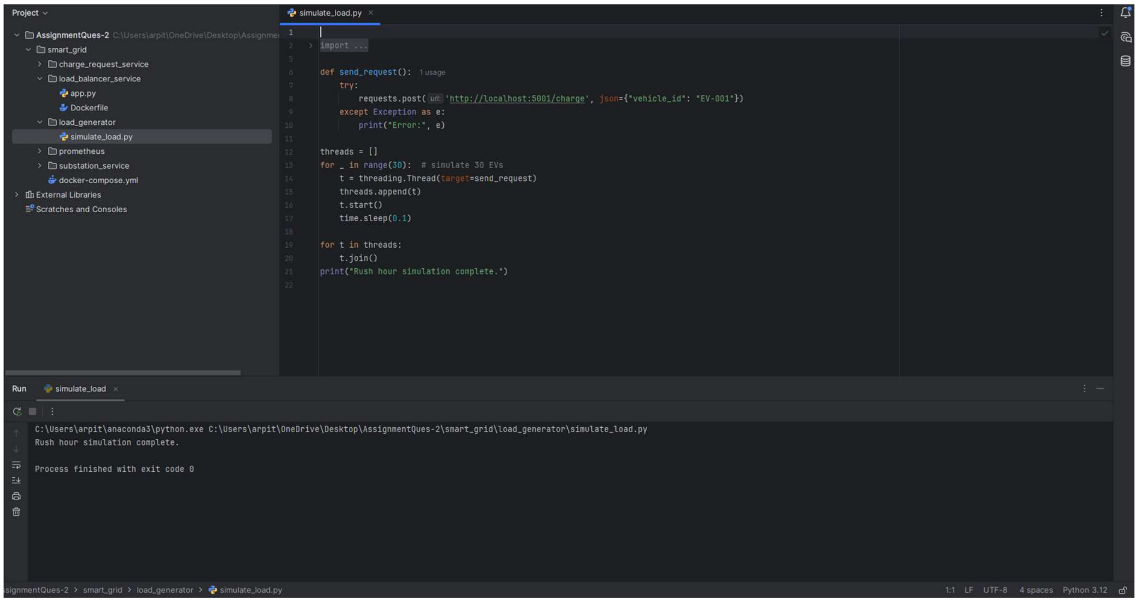
After password reset -



http://localhost:9090



Pycharm Screenshots -



Conclusion

This project successfully demonstrates a scalable approach to **dynamic load balancing** for EV charging in a smart grid context. The custom load balancer, integrated with Prometheus and Grafana, intelligently routes requests to avoid overload while providing full observability into system performance. This framework could be extended further to support more advanced load prediction, prioritization, or integration with real-world energy management systems.