

Project Report

Course: Fundamentals of Distributed Systems

Project Title: Smart Grid Load Balancer

Submitted By: Kaushal Kushwaha

Roll Number: G24AI2098

Abstract

This project presents the design and implementation of a cloud-native Smart Grid Load Balancer system that dynamically routes electric vehicle (EV) charging requests based on real-time substation load. The architecture leverages modern observability tools—Prometheus and Grafana—to monitor and visualize system performance while ensuring optimal load distribution. The system simulates substations, performs high-load testing, and validates that charging requests are balanced evenly. It provides a scalable, modular, and monitorable infrastructure aligned with distributed system principles.

1. Introduction

As electric vehicles (EVs) become more prevalent, managing energy demand across charging infrastructure is critical. Overloading a substation can lead to inefficiency or failure. This project introduces a Smart Grid Load Balancer that routes EV charge requests to the least-loaded substation using real-time telemetry. The goal is to optimize resource usage, maintain service reliability, and offer full visibility into system health.

2. Objectives

- Design a scalable load balancing system for EV charge requests.
- Prevent substation overload by tracking real-time metrics.

- Expose observability metrics via Prometheus and visualize with Grafana.
- Simulate traffic using a Python load-testing script.
- Containerize the system using Docker for isolated deployment.

3. System Architecture

- The system follows a modular, service-oriented architecture:
- Charge Request Service: Entry point for EVs to submit requests.
- Load Balancer: Fetches substation metrics and routes requests to the least-loaded substation.
- Substation Services: Simulate charging activity; expose Prometheus metrics.
- Observability Stack: Prometheus scrapes metrics; Grafana visualizes trends.
- Load Tester: A Python script sends 50 EV requests for validation.

4. Technologies Used

- Python 3.10 (Microservices)
- Flask (REST APIs)
- Prometheus Client Library (Metrics)
- Docker & Docker Compose (Deployment)
- Prometheus (Monitoring)
- Grafana (Visualization)

5. Directory Structure

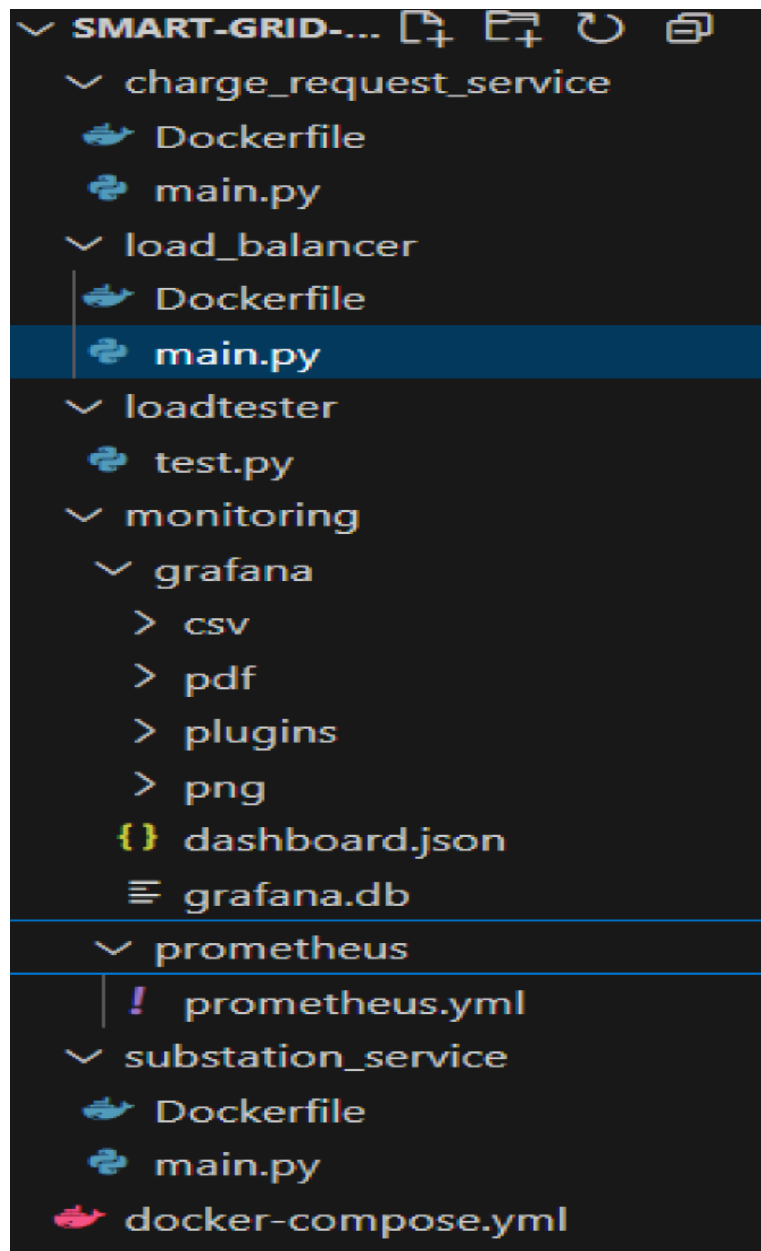
smart-grid-load-balancer/

├── docker-compose.yml

├── load_balancer/

| └── app.py

```
|— substation/
|   └─ app.py
|— charge_request/
|   └─ app.py
|— load_test/
|   └─ test_script.py
|— prometheus/
|   └─ prometheus.yml
|— grafana/
└─ project_report.pdf
```



6. Load Balancing Logic

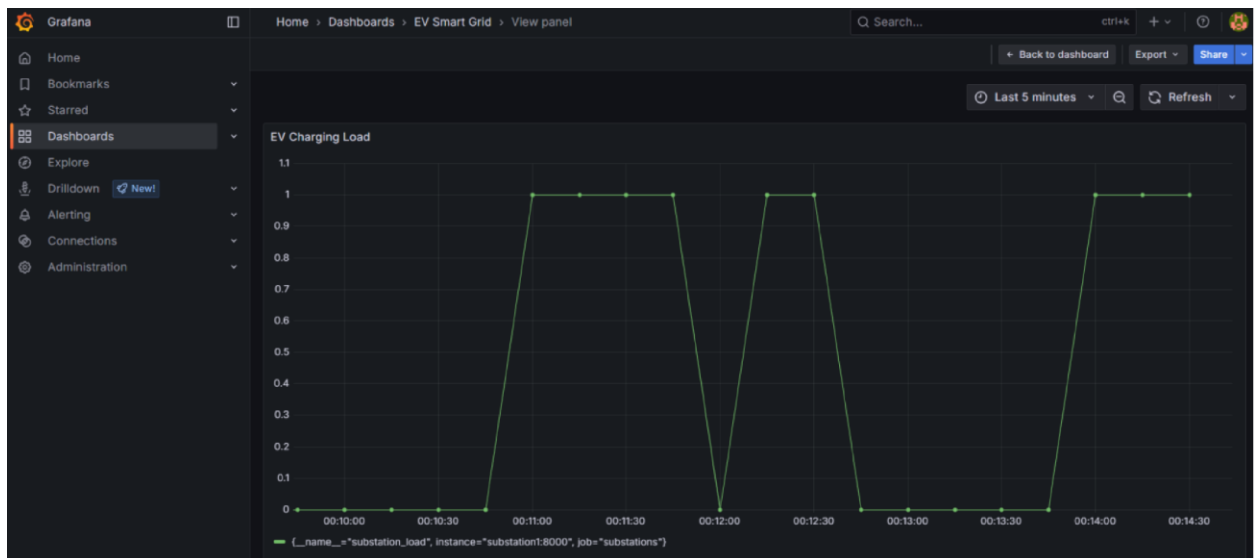
The load balancer service queries `/metrics` endpoints from all substations. It reads the `substation_load` metric and identifies the substation with the minimum load. The incoming request is then routed via HTTP POST to that node, thereby ensuring balanced and optimized resource utilization.

7. Observability and Monitoring

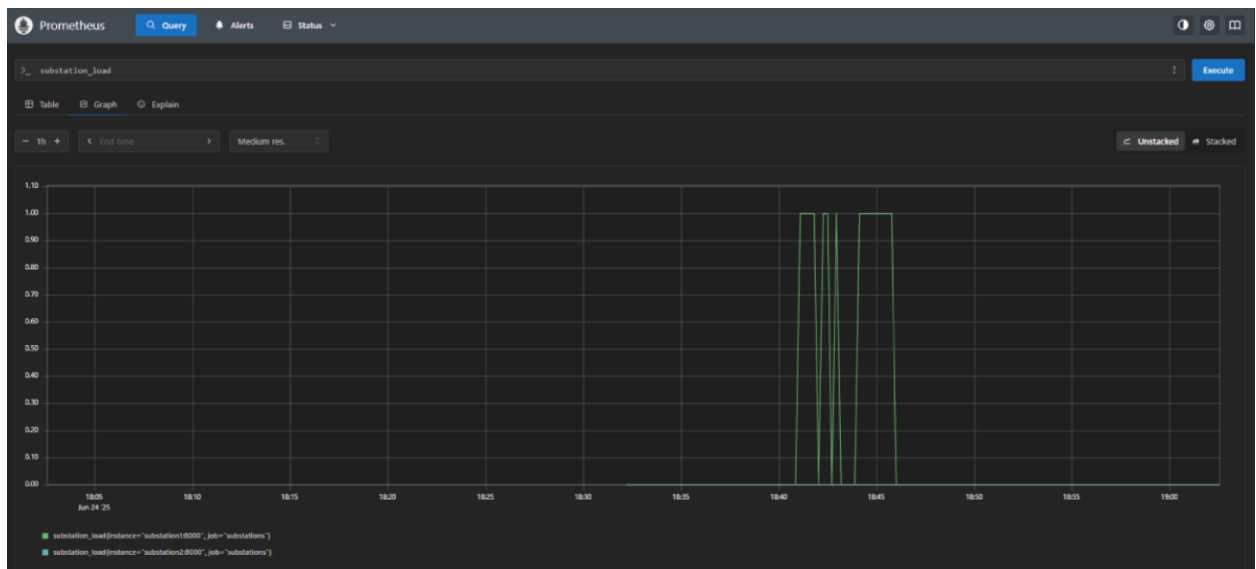
- Substations expose `substation_load` metric using Prometheus client.
- Prometheus scrapes metrics at fixed intervals (e.g., every 5s).
- Grafana displays live load metrics per substation.

Screenshots to be attached:

- Dashboard: Real-time load distribution



- Logs: Metrics exposed at `/metrics`



- Terminal: Docker Compose logs

[illegible]

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:28] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:28] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:30] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:30] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:30] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:32] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:32] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:32] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:34] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:34] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:34] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:37] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:37] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:37] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:39] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:39] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:39] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:41] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:41] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:41] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:44] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:44] "POST /dispatch HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.1 - - [24/Jun/2025 18:45:44] "POST /charge HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:46] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:46] "POST /dispatch HTTP/1.1" 200 -
substation1 | 172.20.0.6 - - [24/Jun/2025 18:45:48] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.7 - - [24/Jun/2025 18:45:48] "POST /dispatch HTTP/1.1" 200 -
```

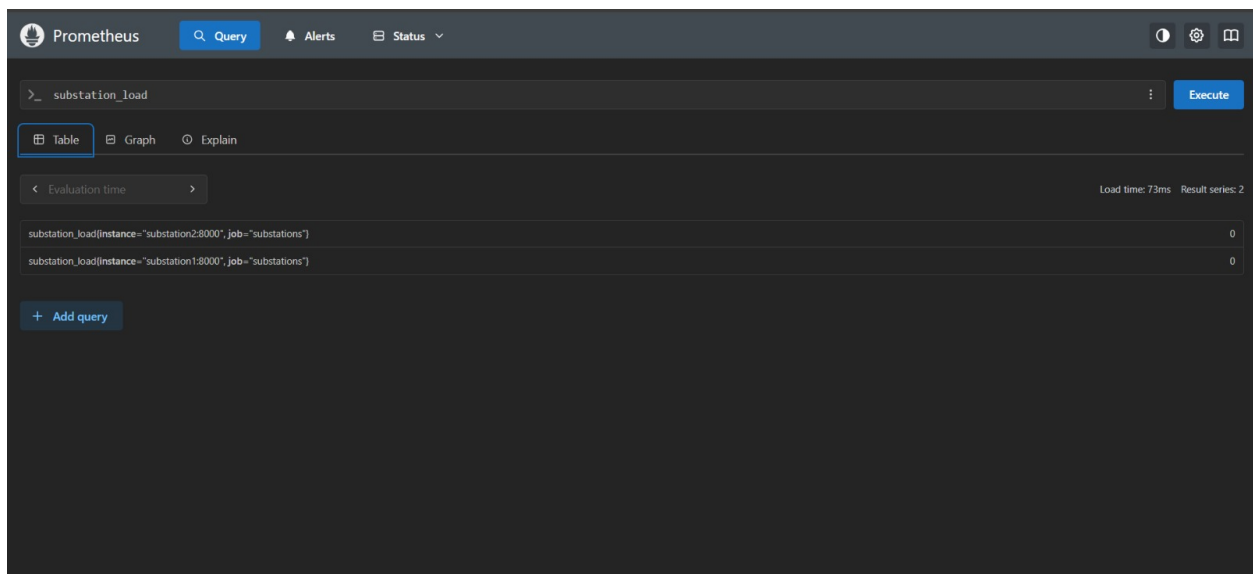
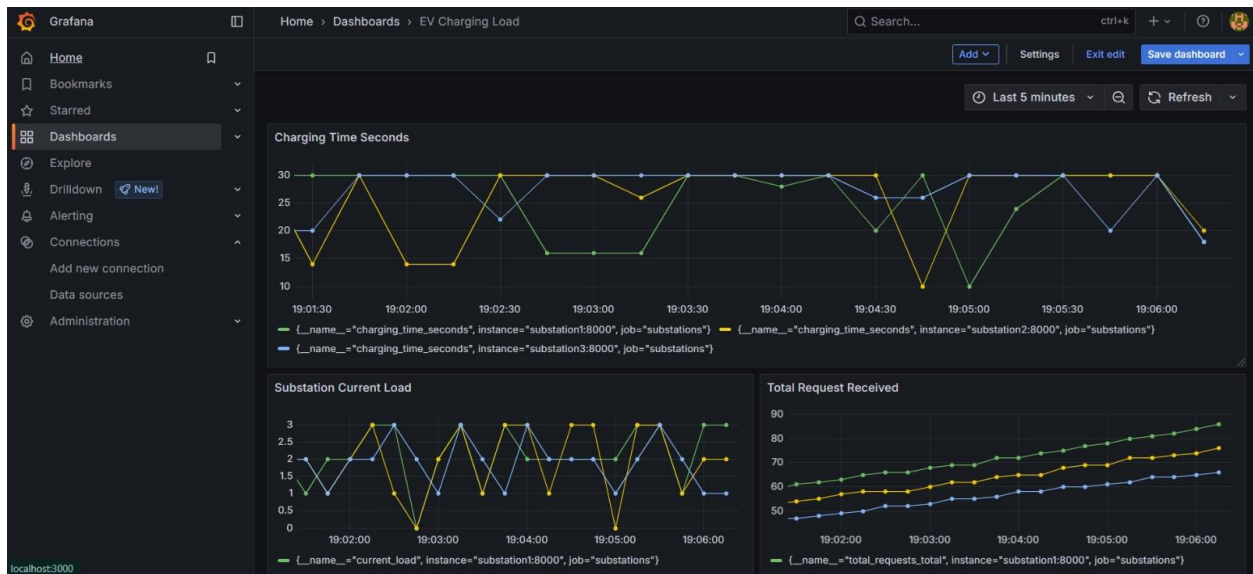
8. Load Testing and Results

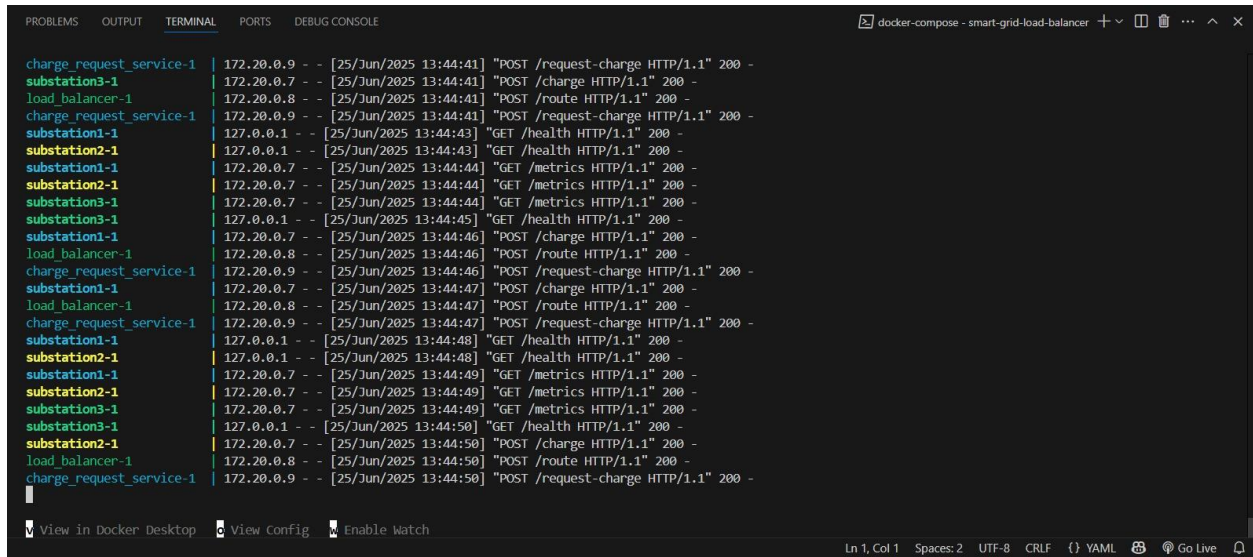
- A Python script simulates 50 EV charge requests (high-load condition).
- Each request is routed based on real-time load feedback.
- Observations show that the system avoids overload and evenly splits requests.

Screenshots to be attached:

Test script execution output

Grafana graph: Load balancing over time





The screenshot shows a Docker terminal window titled "docker-compose - smart-grid-load-balancer". The terminal displays a series of HTTP logs for various services. The logs are organized into columns: service name, IP address, timestamp, and the HTTP request/response details. The services include charge_request_service-1, substation3-1, load_balancer-1, substation1-1, substation2-1, and substation3-1. The logs show a mix of POST and GET requests to various endpoints like /request-charge, /charge, /route, /health, and /metrics, all returning a 200 status code. The terminal interface includes tabs for PROBLEMS, OUTPUT, TERMINAL, PORTS, and DEBUG CONSOLE. At the bottom, there are buttons for "View in Docker Desktop", "View Config", and "Enable Watch", along with a status bar showing "Ln 1, Col 1", "Spaces: 2", "UTF-8", "CRLF", and "YAML".

```
charge_request_service-1 | 172.20.0.9 - - [25/Jun/2025 13:44:41] "POST /request-charge HTTP/1.1" 200 -
substation3-1 | 172.20.0.7 - - [25/Jun/2025 13:44:41] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.8 - - [25/Jun/2025 13:44:41] "POST /route HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.9 - - [25/Jun/2025 13:44:41] "POST /request-charge HTTP/1.1" 200 -
substation1-1 | 127.0.0.1 - - [25/Jun/2025 13:44:43] "GET /health HTTP/1.1" 200 -
substation2-1 | 127.0.0.1 - - [25/Jun/2025 13:44:43] "GET /health HTTP/1.1" 200 -
substation1-1 | 172.20.0.7 - - [25/Jun/2025 13:44:44] "GET /metrics HTTP/1.1" 200 -
substation2-1 | 172.20.0.7 - - [25/Jun/2025 13:44:44] "GET /metrics HTTP/1.1" 200 -
substation3-1 | 172.20.0.7 - - [25/Jun/2025 13:44:44] "GET /metrics HTTP/1.1" 200 -
substation3-1 | 127.0.0.1 - - [25/Jun/2025 13:44:45] "GET /health HTTP/1.1" 200 -
substation1-1 | 172.20.0.7 - - [25/Jun/2025 13:44:46] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.8 - - [25/Jun/2025 13:44:46] "POST /route HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.9 - - [25/Jun/2025 13:44:46] "POST /request-charge HTTP/1.1" 200 -
substation1-1 | 172.20.0.7 - - [25/Jun/2025 13:44:47] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.8 - - [25/Jun/2025 13:44:47] "POST /route HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.9 - - [25/Jun/2025 13:44:47] "POST /request-charge HTTP/1.1" 200 -
substation1-1 | 127.0.0.1 - - [25/Jun/2025 13:44:48] "GET /health HTTP/1.1" 200 -
substation2-1 | 127.0.0.1 - - [25/Jun/2025 13:44:48] "GET /health HTTP/1.1" 200 -
substation1-1 | 172.20.0.7 - - [25/Jun/2025 13:44:49] "GET /metrics HTTP/1.1" 200 -
substation2-1 | 172.20.0.7 - - [25/Jun/2025 13:44:49] "GET /metrics HTTP/1.1" 200 -
substation3-1 | 172.20.0.7 - - [25/Jun/2025 13:44:49] "GET /metrics HTTP/1.1" 200 -
substation3-1 | 127.0.0.1 - - [25/Jun/2025 13:44:50] "GET /health HTTP/1.1" 200 -
substation2-1 | 172.20.0.7 - - [25/Jun/2025 13:44:50] "POST /charge HTTP/1.1" 200 -
load_balancer-1 | 172.20.0.8 - - [25/Jun/2025 13:44:50] "POST /route HTTP/1.1" 200 -
charge_request_service-1 | 172.20.0.9 - - [25/Jun/2025 13:44:50] "POST /request-charge HTTP/1.1" 200 -
```

9. Video Demonstration

A demonstration video includes:

- Full system setup using Docker
- Execution of EV request simulator
- Monitoring via Grafana and Prometheus

Video Link:

10. Conclusion

This project validates a real-time load balancing approach for EV grid infrastructure using metrics-driven decision-making. The system is cloud-native, scalable, and observable. It provides insights into distributed load handling and offers a foundation for extending to real-world applications such as smart grid automation, priority-based routing, and fault-tolerant charging networks.