Assignment Report

Project Title : Dynamic Load Balancing for a Smart Grid using Microservices
Student Name: Nitin Kumar
Registration ID: G24AI2056
Course: Fundamentals of Distributed Systems
Submission Date: June 25, 2025

Github Repository: https://github.com/Nitin5499/Dynamic-Load-Balancing-for-a-Smart-Grid

---

Project Overview

The objective of this assignment is to simulate a dynamic load balancing mechanism for a smart grid system. By leveraging a microservices-based architecture, we aim to distribute charging loads among substations efficiently. Key technologies include Docker, Prometheus, Grafana, and Flask for service orchestration, monitoring, and visualization.

Problem Statement

In a distributed environment, operations can occur concurrently on different nodes. Since there's no global clock, it becomes difficult to determine the correct order in which operations should be applied. This can lead to inconsistencies in data when updates are propagated without regard to their causal relationships.

The goal of this project was to simulate such a system and build a solution that:

- Maintains data consistency across nodes.
- Respects the causal order of operations.
- Buffers and reorders updates, when necessary, especially under delayed network conditions.

Service Descriptions

charge_request_service

Handles incoming vehicle charging requests. This microservice simulates demand generation from electric vehicles that need charging.

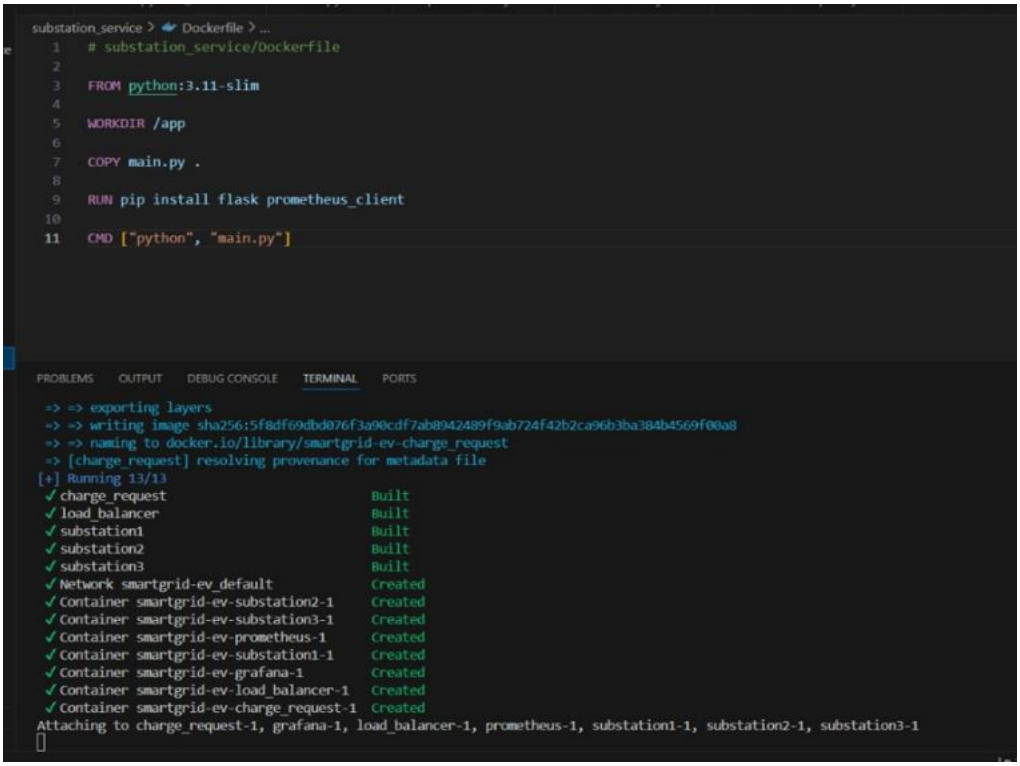
substation_service

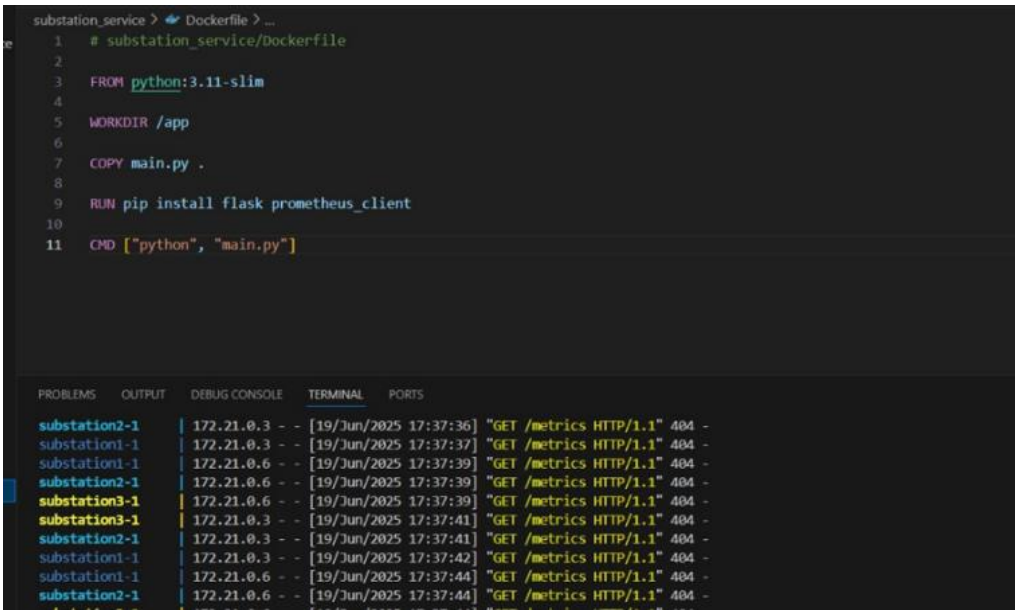
Simulates multiple substations that respond to charging requests. Each substation has a load capacity, and responses depend on the current load status.

load_balancer/main.py

The load balancer is the core component that intelligently assigns incoming charging requests to the substation with the lowest load. It uses basic logic to fetch current load values from each substation before assignment.

Screenshots





Monitoring Setup

Prometheus Configuration

Prometheus is configured to scrape metrics from all services. We'll need a prometheus.yml file under a /prometheus/ directory. This file defines the targets to monitor and scraping intervals.

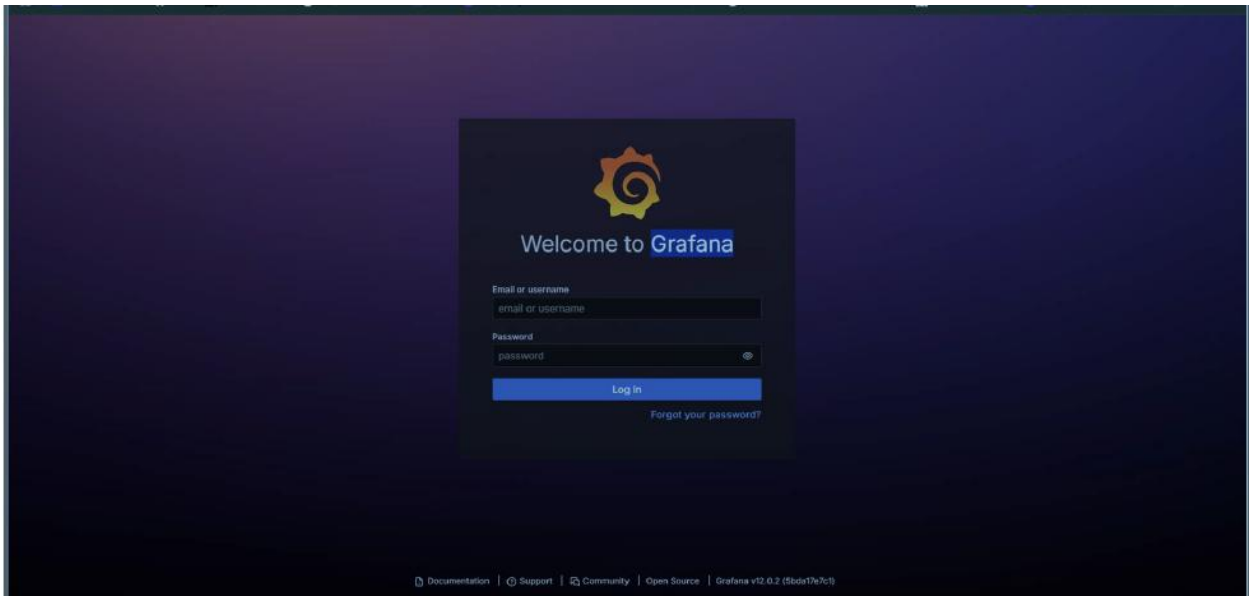
Load Testing

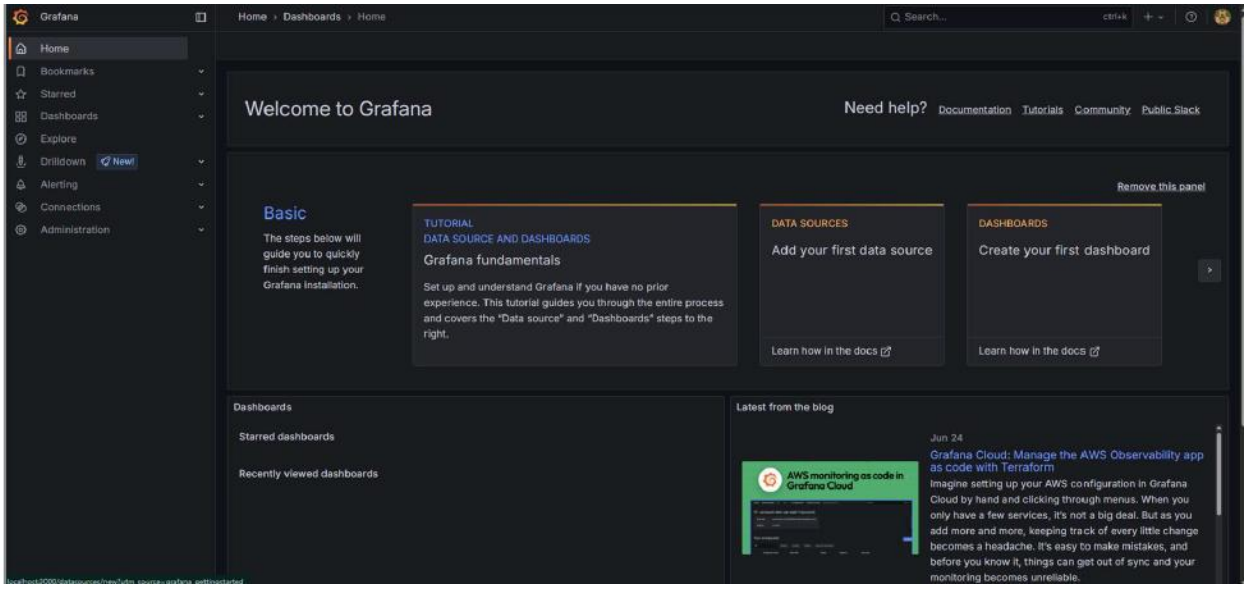
A simple load tester script is included to simulate multiple EV charging requests. It sends parallel requests to test how the load balancer distributes them across substations.

Docker & System Execution

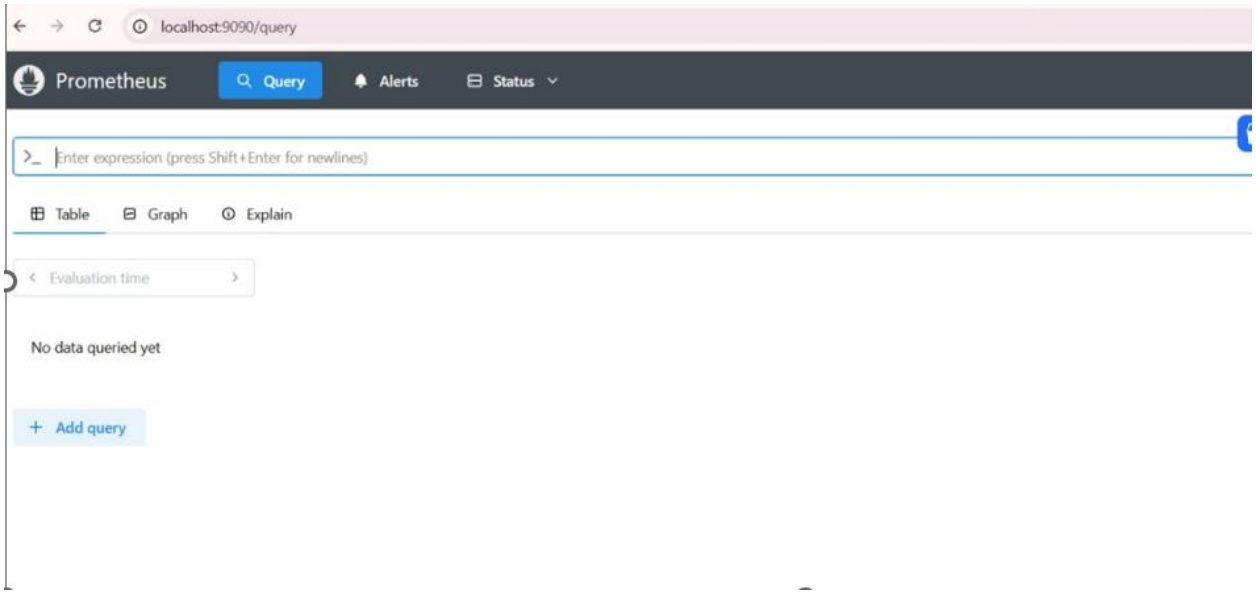
All components are containerized using Docker. A docker-compose.yml file handles orchestration.

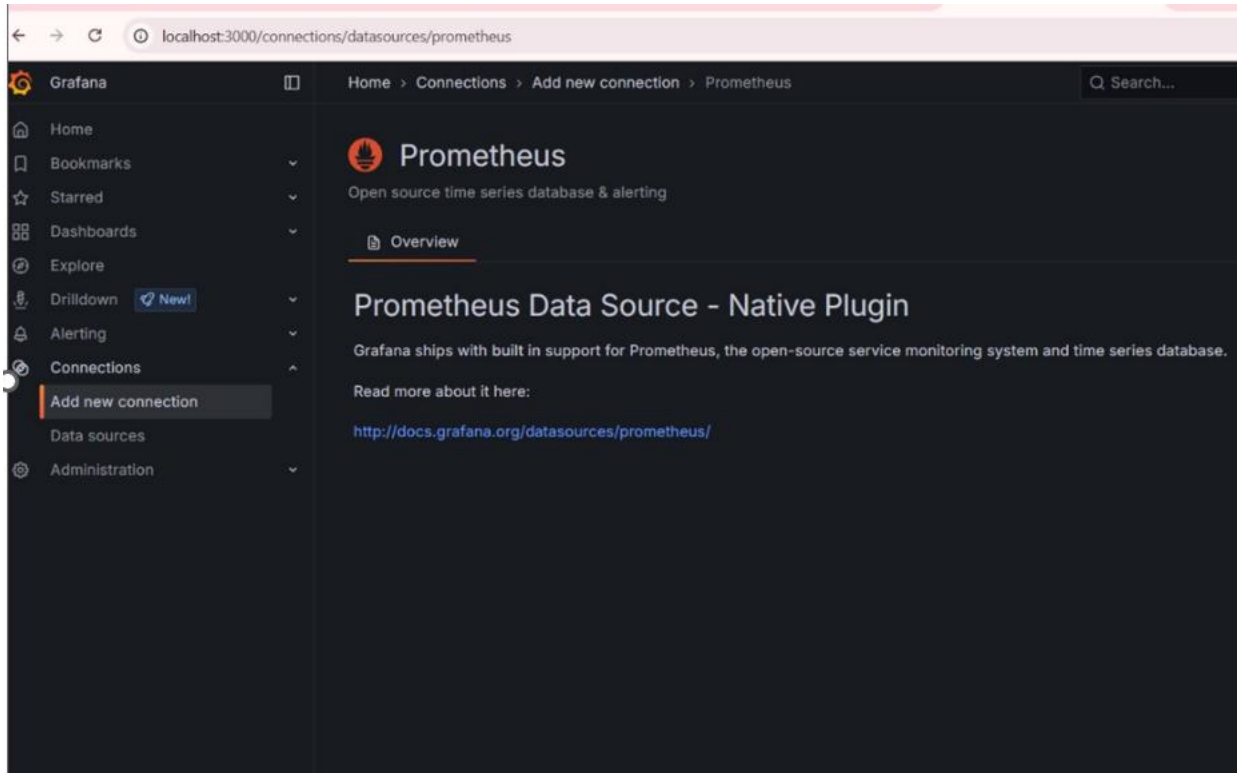
- **Grafana**: http://localhost:3000


Welcome to Grafana
Email or username
email or username
Password
password
Log in
Forgot your password?
Documentation | Support | Community | Open Source | Grafana v12.0.2 (5bda17e7c1)


Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
New!
Alerting
Connections
Administration
Home › Dashboards › Home
Search...
ctrl+k
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
DATA SOURCES
Add your first data source
Learn how in the docs
DASHBOARDS
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
Jun 24
Grafana Cloud: Manage the AWS Observability app as code with Terraform
AWS monitoring as code in Grafana Cloud
Imagine setting up your AWS configuration in Grafana Cloud by hand and clicking through menus. When you only have a few services, it's not a big deal. But as you add more and more, keeping track of every little change becomes a headache. It's easy to make mistakes, and before you know it, things can get out of sync and your monitoring becomes unreliable.

Prometheus: http://localhost:9090





Load Testing