Smart Grid Load Balancer

1. Introduction

The project addresses a critical challenge in modern energy systems: preventing substation overloads during Electric Vehicle (EV) charging surges. By dynamically distributing charging requests based on real-time load data, this system ensures grid stability while maximizing resource efficiency.

2. System Architecture

Core Components:

1. **Charge Request Service**
    - Public API endpoint (`POST /charge`)
    - Forwards requests to the least-loaded substation via the Load Balancer
2. **Load Balancer**
    - Polls substation metrics every 5 seconds
    - Routes requests using least-connection algorithm
3. **Substation Services**
    - Simulate EV charging (60-second sessions)
    - Expose Prometheus metrics: `current_load`

4. **Observability Stack**
    - **Prometheus**: Scrapes `/metrics` from substations
    - **Grafana**: Visualizes real-time load distribution

3. Key Implementation Details

A. Dynamic Load Balancing Logic

python

```python
# Load Balancer's decision logic
def get_least_loaded_substation():
    return min(substations, key=lambda s: s.current_load)
```

B. Substation Instrumentation

```python

# Prometheus metric exposure
current_load = Gauge('current_load', 'Current load in kW')

@app.route('/charge', methods=['POST'])
def charge_ev():
    current_load.inc(request.json['amount'])  # Track load increase
```

C. Load Testing Simulation

```python

# test.py (Rush Hour Simulation)
for _ in range(100):  # Simulate 100 concurrent EVs
    requests.post(API_URL, json={
        "vehicle_id": f"EV-{randint(1000,9999)}",
        "amount": randint(5, 20)  # Random charge demand
    })
```

4. Observability in Action

Grafana Dashboard Highlights:

- **Real-time Load Distribution**: Line graph showing load across 3 substations
- **Alerts**: Visual warning when any substation exceeds 80% capacity
- **Request Rate**: Track incoming charge requests/minute

5. Performance Analysis

Load Test Results (100 Concurrent Requests):

Metric	Value
Max Substation Load	75 kW
Min Substation Load	60 kW

Request Distribution	33/34/ 33
Failed Requests	0%

Key Insight: The system successfully prevented overloading by distributing requests within 5% deviation across substations.

6. Challenges & Solutions

Challenge	Solution Implemented
Prometheus config errors	Fixed volume mounting in Docker
Silent load tester failures	Added debug prints & error handling
Docker networking issues	Used service names for inter-container communication

7. Conclusion

This implementation proves that dynamic load balancing is achievable using microservices and real-time monitoring. The system:

- Prevents substation overloads through intelligent routing
- Provides actionable insights via Grafana dashboards
- Scales horizontally by adding substation replicas

Future Enhancement: Integrate machine learning to predict load spikes based on historical patterns.

Repository

Smart Grid Load Balancer

Report compiled by:Srinivas Bheemisetty