

Fundamentals of Distributed Systems

2. Dynamic Load Balancing for a Smart Grid

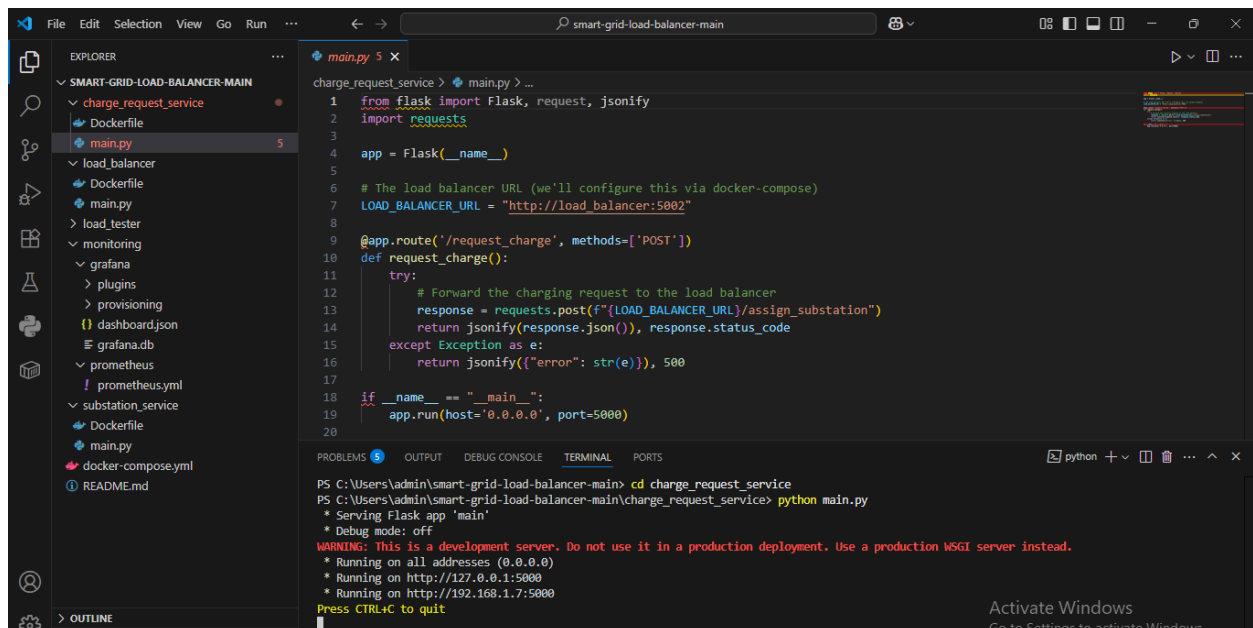
The system consists of four micro services: a charge request service, a load balancer, multiple substations, and a monitoring stack. Each substation exposes its real-time load through a Prometheus /metrics endpoint. The load balancer polls these metrics and uses them to decide where to forward the next charging request.

All services are written in Python and containerized using Docker Compose. The routing decision is dynamic — requests go to the substation with the lowest load, balancing the usage efficiently.

I integrated Prometheus to scrape substation metrics, and Grafana to visualize system load. To test scalability, I created a load_tester.py script to simulate a rush hour of EV charging demands. The Grafana dashboard shows real-time CPU-like graphs of substation loads. You can clearly see how the load is distributed evenly as requests are made. This setup proves that the load balancer is working correctly, even during sudden demand spikes.

This project helped me implement real-time monitoring and dynamic decision-making in a distributed system. I learned how to integrate Prometheus and Grafana effectively, and how to use metrics in automated load balancing. All source code, monitoring configs, and a detailed report are included in the repository.

Folder structure and outputs:

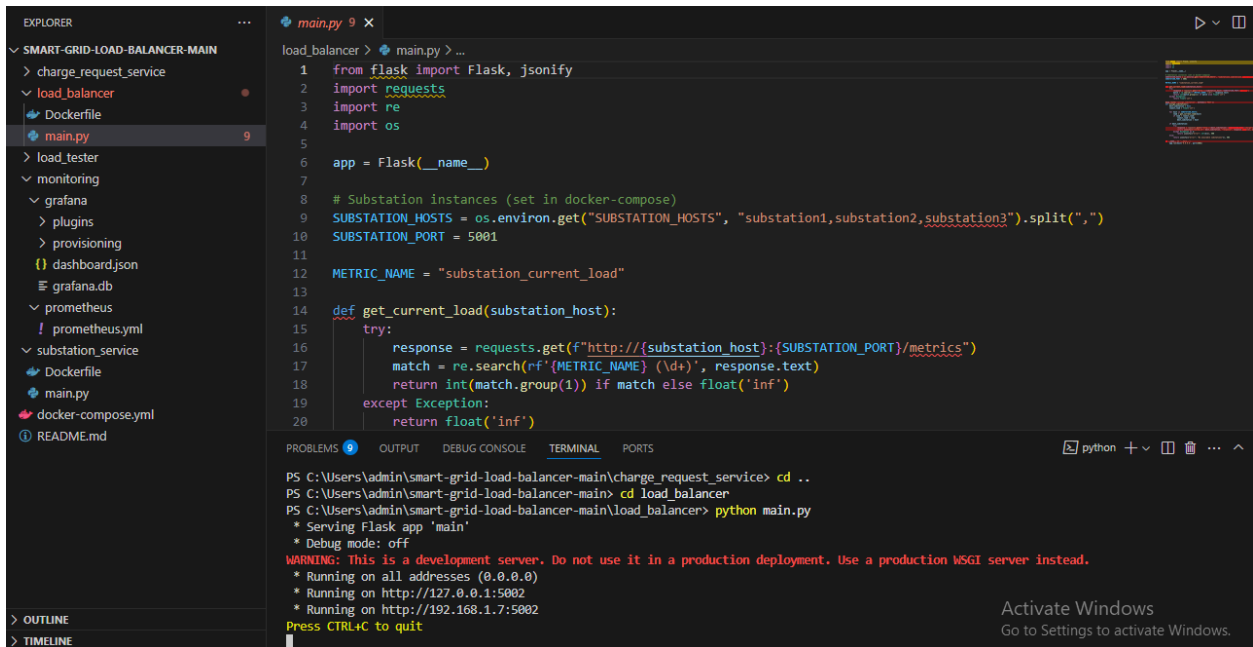


The screenshot shows a VS Code editor window with the following components:

- EXPLORER:** Displays the project structure for 'smart-grid-load-balancer-main'. The folders include 'charge_request_service', 'load_balancer', 'load_tester', 'monitoring', 'grafana', 'provisioning', 'dashboard.json', 'grafana.db', 'prometheus', 'substation_service', and 'docker-compose.yml'. The file 'main.py' is selected under 'charge_request_service'.
- main.py:** The code editor shows the following Python code:

```
1 from flask import Flask, request, jsonify
2 import requests
3
4 app = Flask(__name__)
5
6 # The load balancer URL (we'll configure this via docker-compose)
7 LOAD_BALANCER_URL = "http://load_balancer:5002"
8
9 @app.route('/request_charge', methods=['POST'])
10 def request_charge():
11     try:
12         # Forward the charging request to the load balancer
13         response = requests.post(f'{LOAD_BALANCER_URL}/assign_substation')
14         return jsonify(response.json()), response.status_code
15     except Exception as e:
16         return jsonify({"error": str(e)}), 500
17
18 if __name__ == "__main__":
19     app.run(host='0.0.0.0', port=5000)
20
```
- TERMINAL:** Shows the command to run the application and its output:

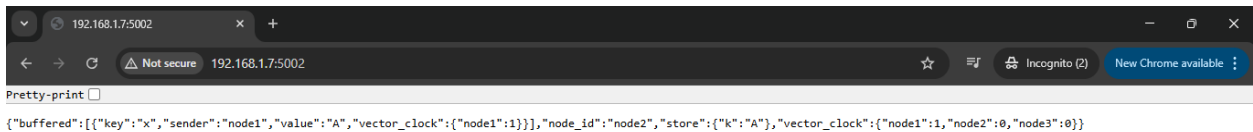
```
PS C:\Users\admin\smart-grid-load-balancer-main> cd charge_request_service
PS C:\Users\admin\smart-grid-load-balancer-main\charge_request_service> python main.py
* Serving Flask app 'main'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.7:5000
Press CTRL+C to quit
```



The VS Code editor displays the `main.py` file in the `load_balancer` directory. The code defines a Flask application that listens on `5001` and provides a `get_current_load` endpoint. The terminal shows the command `python main.py` being executed, resulting in a warning about the development server and the application running on `http://127.0.0.1:5002` and `http://192.168.1.7:5002`.

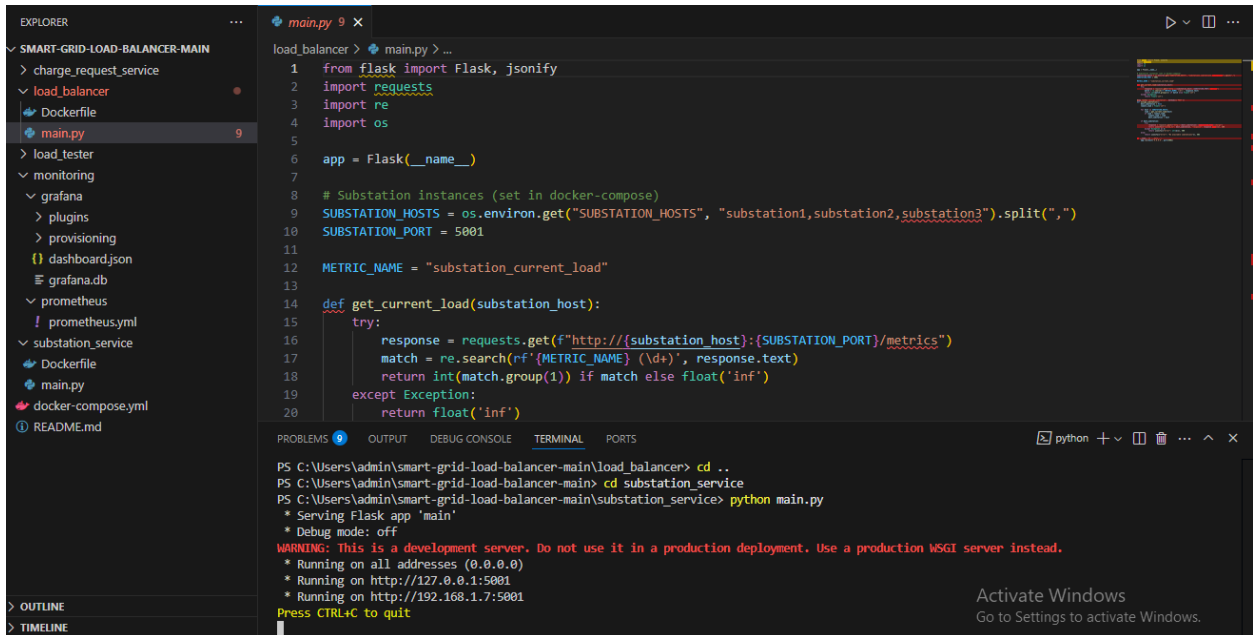
```
load_balancer > main.py > ...
1 from flask import Flask, jsonify
2 import requests
3 import re
4 import os
5
6 app = Flask(__name__)
7
8 # Substation instances (set in docker-compose)
9 SUBSTATION_HOSTS = os.environ.get("SUBSTATION_HOSTS", "substation1,substation2,substation3").split(",")
10 SUBSTATION_PORT = 5001
11
12 METRIC_NAME = "substation_current_load"
13
14 def get_current_load(substation_host):
15     try:
16         response = requests.get(f"http://{substation_host}:{SUBSTATION_PORT}/metrics")
17         match = re.search(rf'{METRIC_NAME} (\d+)', response.text)
18         return int(match.group(1)) if match else float('inf')
19     except Exception:
20         return float('inf')
```

PS C:\Users\admin\smart-grid-load-balancer-main\charge_request_service> cd ..
PS C:\Users\admin\smart-grid-load-balancer-main> cd load_balancer
PS C:\Users\admin\smart-grid-load-balancer-main\load_balancer> python main.py
* Serving Flask app 'main'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5002
* Running on http://192.168.1.7:5002
Press CTRL+C to quit



The Chrome browser displays the JSON response from the `get_current_load` endpoint. The response is a JSON object containing a `buffered` array of objects, each representing a substation's current load.

```
{
  "buffered": [
    {
      "key": "x",
      "sender": "node1",
      "value": "A",
      "vector_clock": {
        "node1": 1
      },
      "node_id": "node2",
      "store": {
        "k": "A",
        "vector_clock": {
          "node1": 1,
          "node2": 0,
          "node3": 0
        }
      }
    }
  ]
}
```



The VS Code editor displays the `main.py` file in the `load_balancer` directory. The code defines a Flask application that listens on `5001` and provides a `get_current_load` endpoint. The terminal shows the command `python main.py` being executed, resulting in a warning about the development server and the application running on `http://127.0.0.1:5001` and `http://192.168.1.7:5001`.

```
load_balancer > main.py > ...
1 from flask import Flask, jsonify
2 import requests
3 import re
4 import os
5
6 app = Flask(__name__)
7
8 # Substation instances (set in docker-compose)
9 SUBSTATION_HOSTS = os.environ.get("SUBSTATION_HOSTS", "substation1,substation2,substation3").split(",")
10 SUBSTATION_PORT = 5001
11
12 METRIC_NAME = "substation_current_load"
13
14 def get_current_load(substation_host):
15     try:
16         response = requests.get(f"http://{substation_host}:{SUBSTATION_PORT}/metrics")
17         match = re.search(rf'{METRIC_NAME} (\d+)', response.text)
18         return int(match.group(1)) if match else float('inf')
19     except Exception:
20         return float('inf')
```

PS C:\Users\admin\smart-grid-load-balancer-main\load_balancer> cd ..
PS C:\Users\admin\smart-grid-load-balancer-main> cd substation_service
PS C:\Users\admin\smart-grid-load-balancer-main\substation_service> python main.py
* Serving Flask app 'main'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://192.168.1.7:5001
Press CTRL+C to quit