Flask API Monitoring with Prometheus & Grafana

This project integrates **Prometheus** and **Grafana** with a Flask API to monitor HTTP requests, response times, and other key metrics.

Features

- Track total HTTP requests & response times
- Monitor API performance using Prometheus
- Visualize data in Grafana with thresholds & alerts
- Export metrics using `prometheus_flask_exporter`

🛠️ Installation & Setup

1️⃣ Install Dependencies

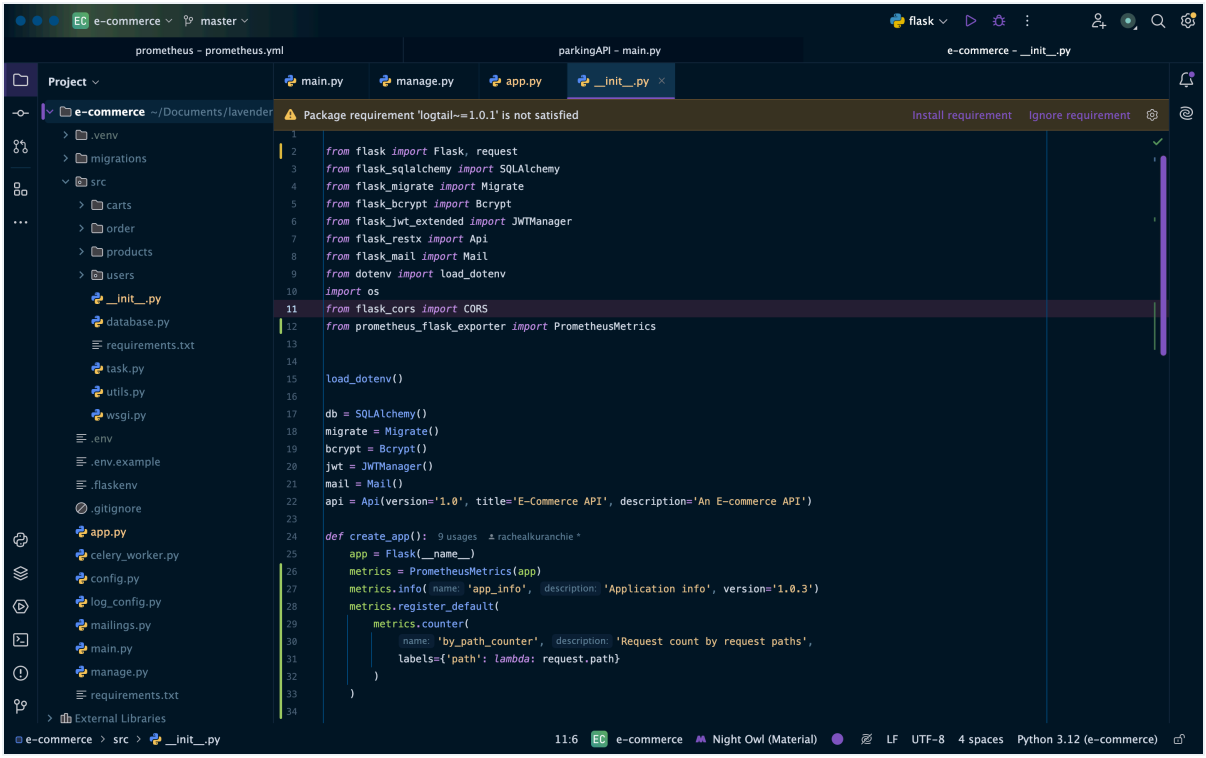
Ensure you have Python installed, then run:

```
pip install flask prometheus_flask_exporter
```

2️⃣ Create a Flask App with Prometheus Exporter

```
from flask import Flask
from prometheus_flask_exporter import PrometheusMetrics
app = Flask(__name__)
metrics = PrometheusMetrics(app)
@app.route("/")
def home():
    return "Welcome to Flask Monitoring!"
@app.route("/users/signup", methods=["POST"])
def signup():
    return "User Signed Up", 201
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

This will expose Prometheus metrics at `http://localhost:5000/metrics`.



3️⃣ Run Flask App

```
python app.py
```

📊 Setting Up Prometheus

1️⃣ Install Prometheus

Setup the docker file for Prometheus and Grafana. reference

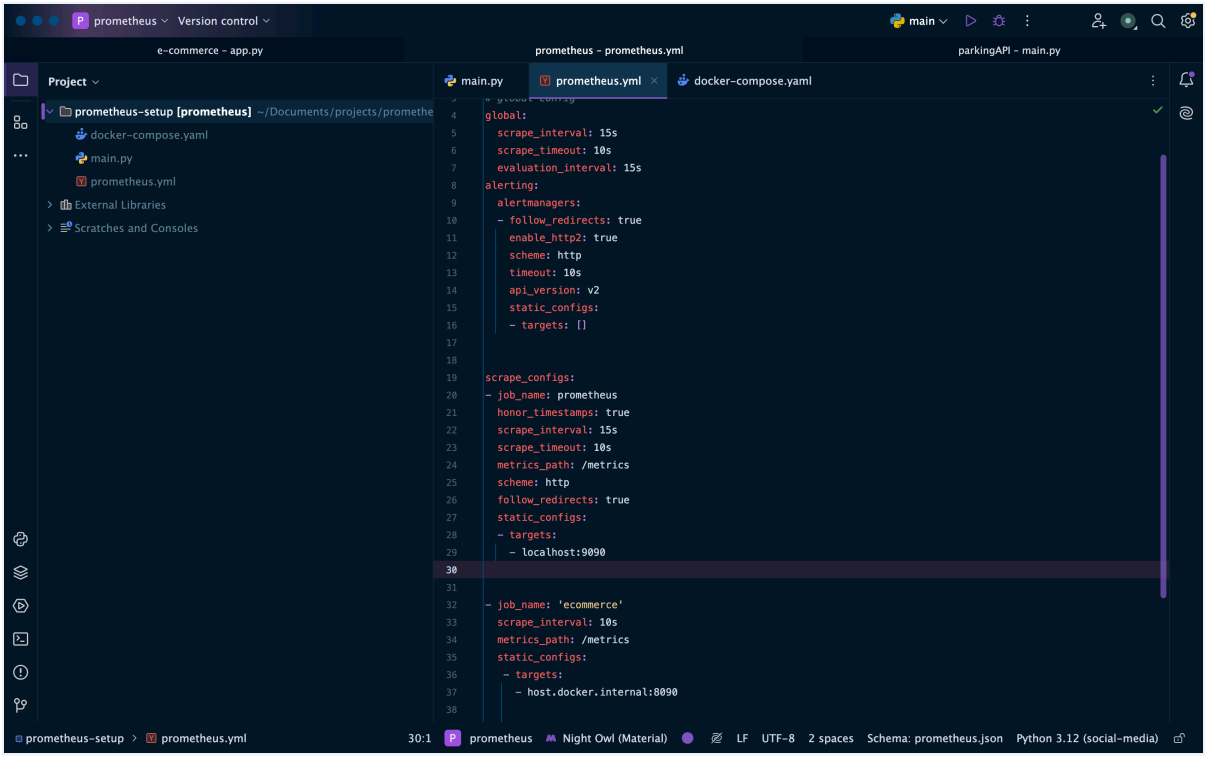
2️⃣ Configure `prometheus.yml`

Create a `prometheus.yml` file:

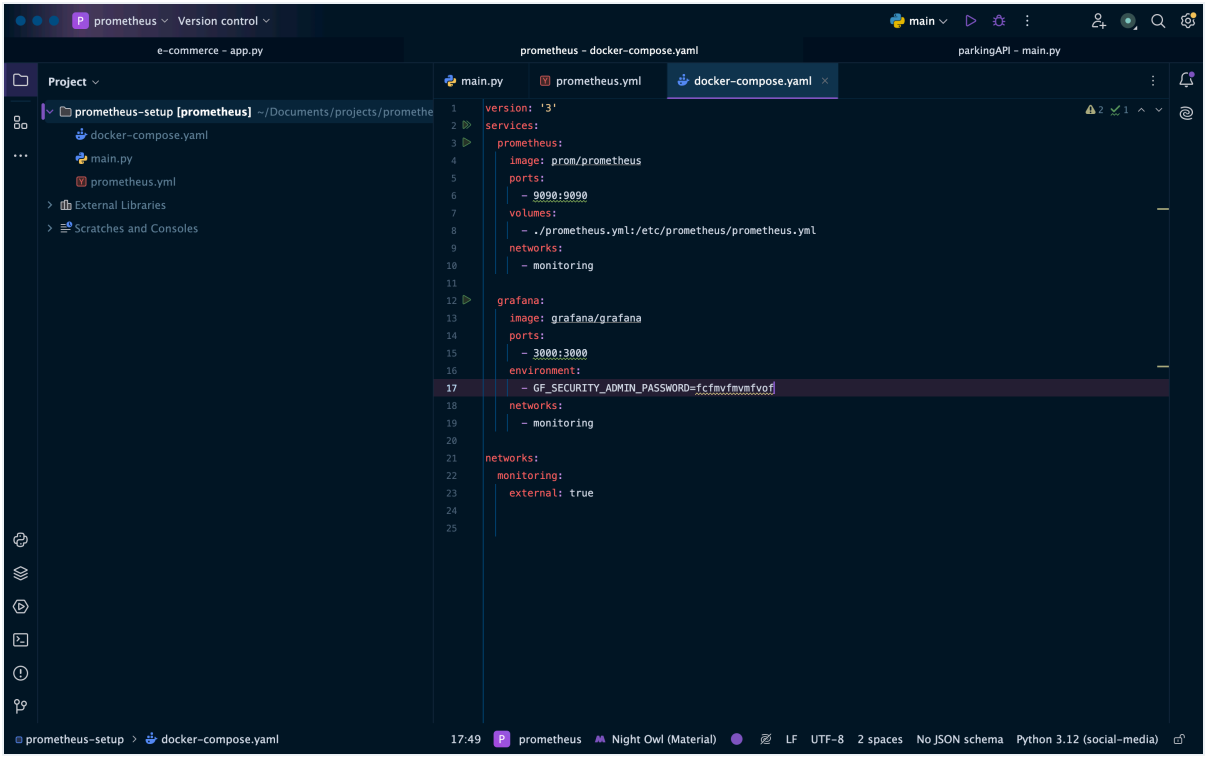
```
# global config
global:
  scrape_interval: 15s
  scrape_timeout: 10s
  evaluation_interval: 15s
alerting:
  alertmanagers:
  - follow_redirects: true
    enable_http2: true
    scheme: http
    timeout: 10s
    api_version: v2
    static_configs:
    - targets: []
scrape_configs:
- job_name: prometheus
  honor_timestamps: true
```

```
  scrape_interval: 15s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: http
  follow_redirects: true
  static_configs:
  - targets:
    - localhost:9090

scrape_configs:
 - job_name: 'ecommerce'
  scrape_interval: 10s
  metrics_path: /metrics
  static_configs:
   - targets:
     - host.docker.internal:8090
```



Create a docker-compose.yaml file and use the docker image of Prometheus and Grafana, setup your password; and make sure the networks for both grafana and prometheus are the same



3️⃣ Run the Docker compose

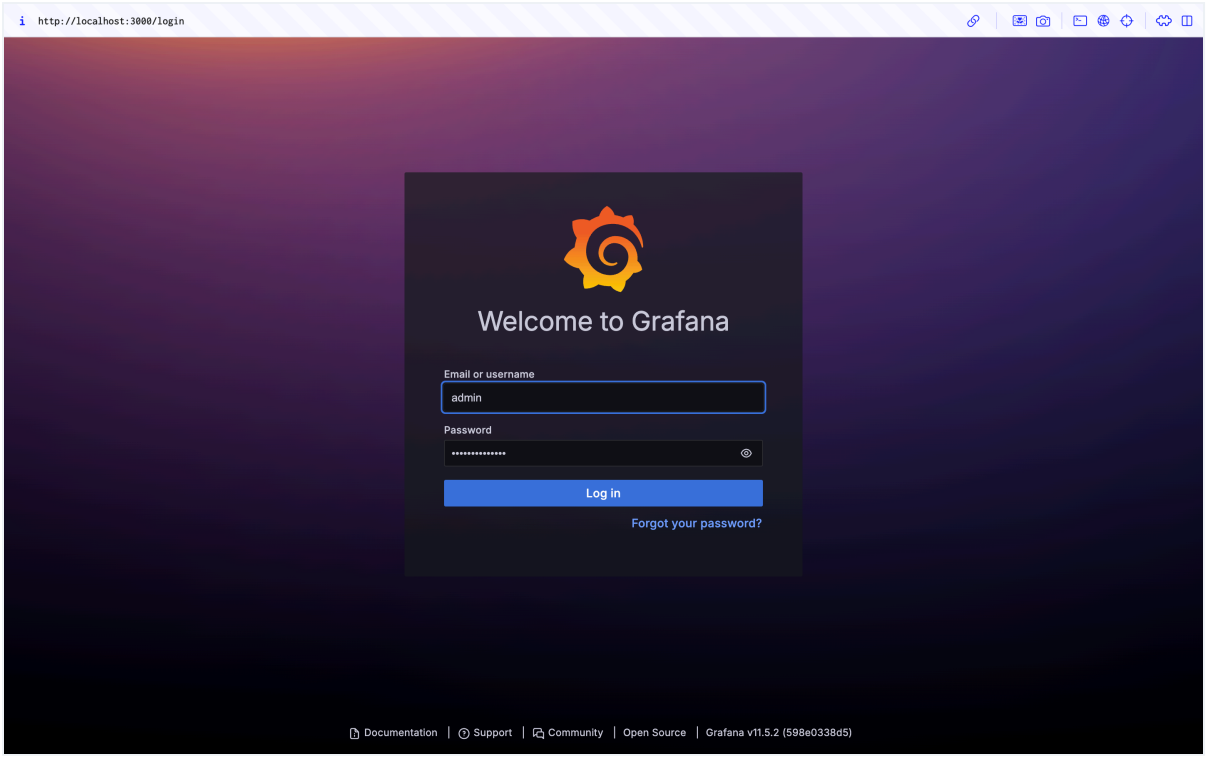
```
docker compose up -d
```

Visit `http://localhost:9090` to access the Prometheus UI.

📈 Setting Up Grafana

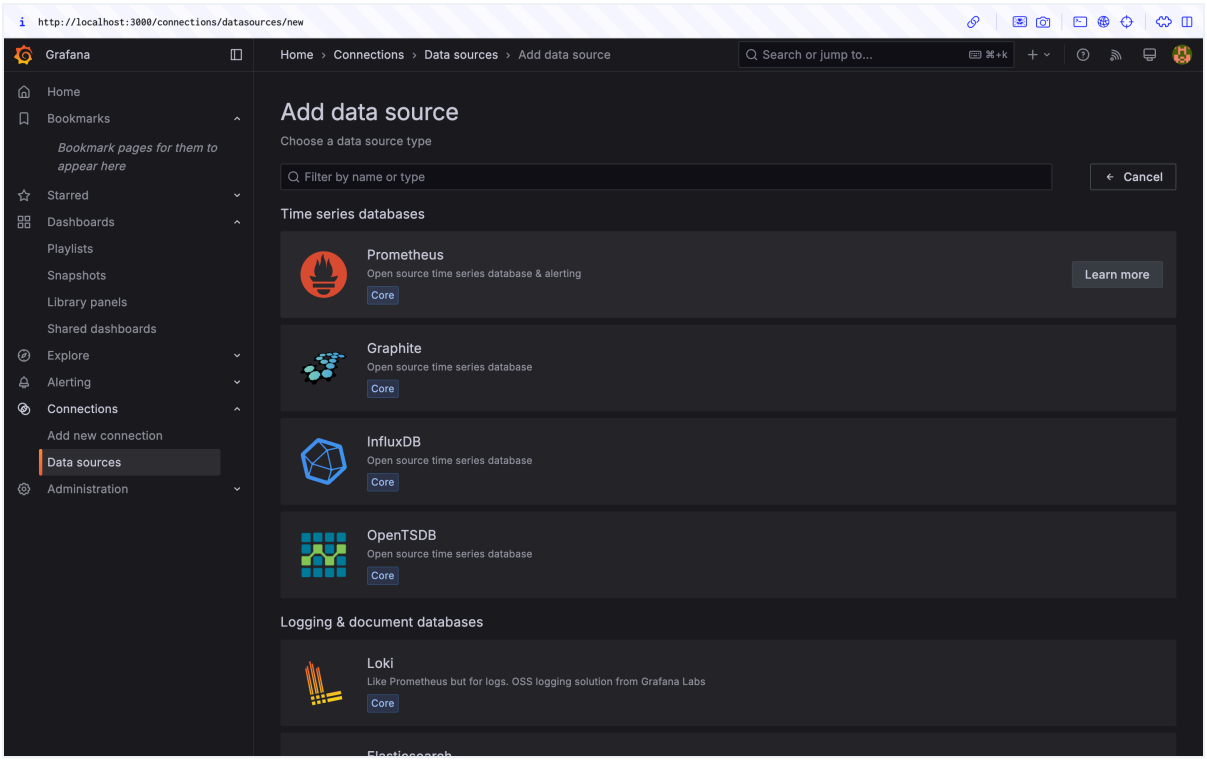
1️⃣ Open Grafana

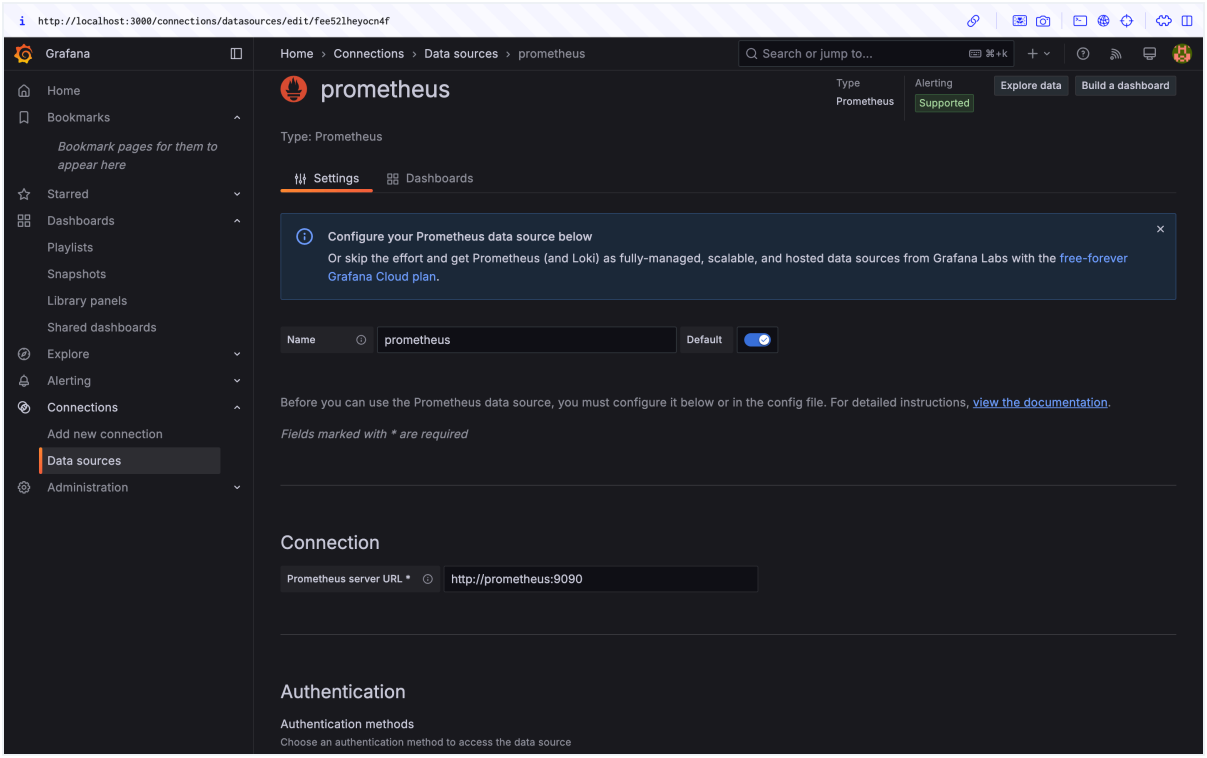
- Open Grafana on localhost 3000
- Use the password you placed in the docker compose on the grafana UI, so admin as the user and your password



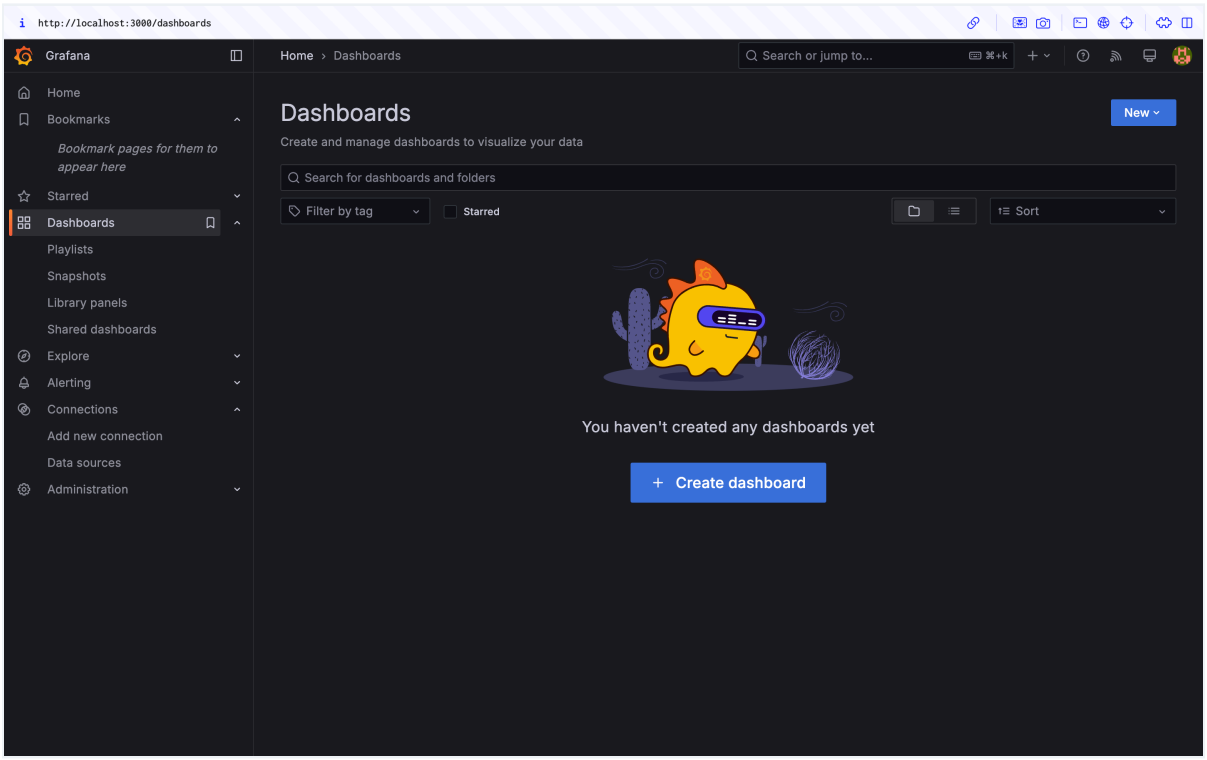
2️⃣ Add Prometheus as a Data Source

- Go to **Configuration → Data Sources**
- Select **Prometheus** and enter `http://localhost:9090`



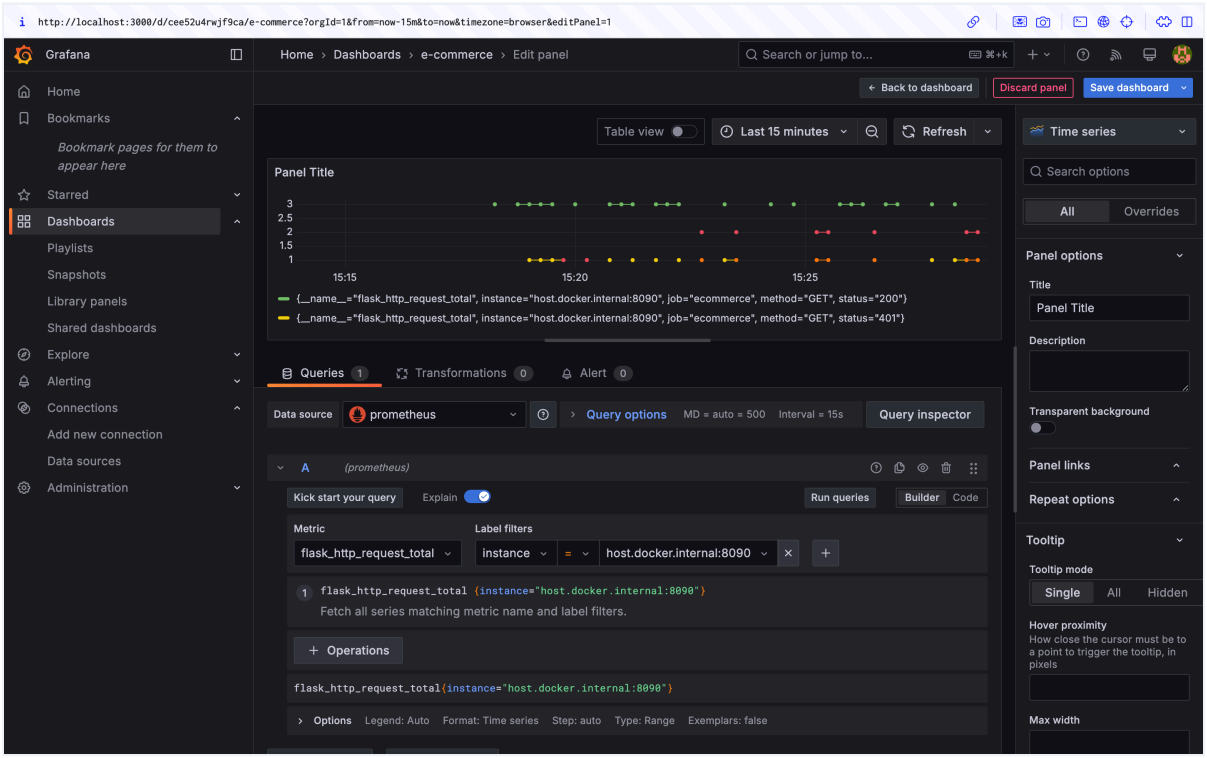


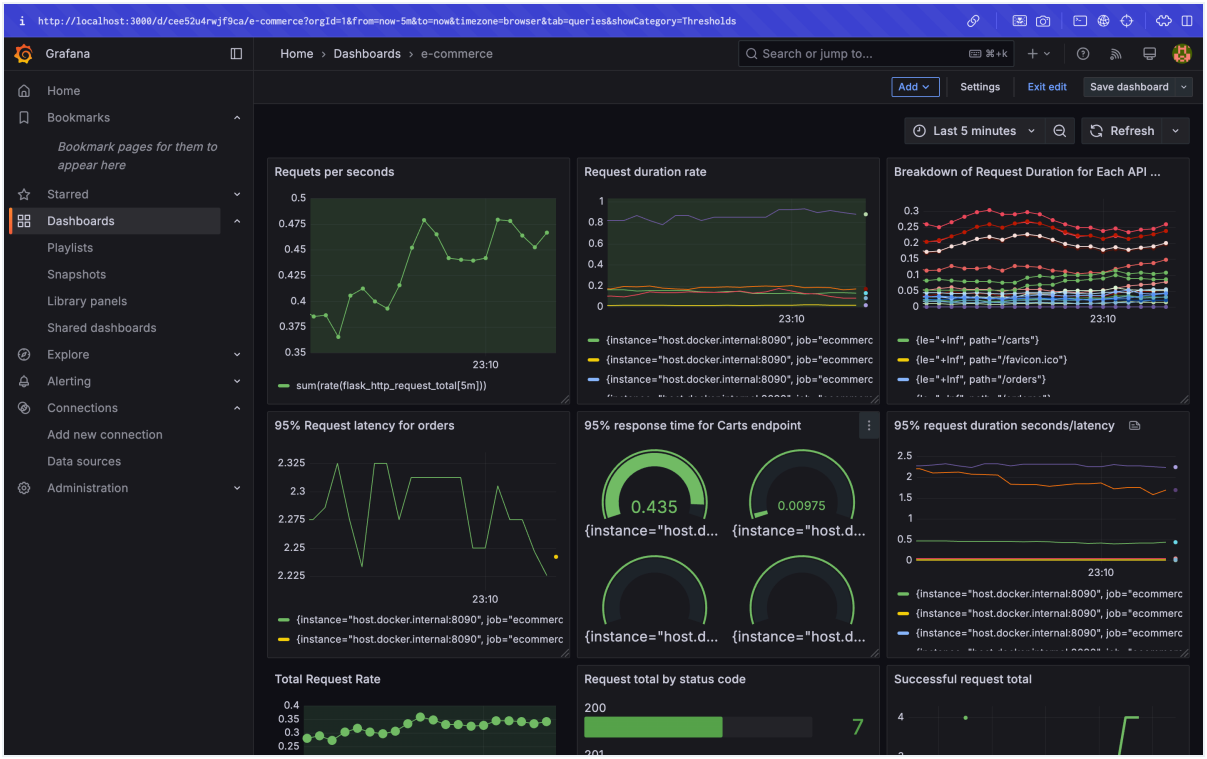
3 Create Dashboards & Panels

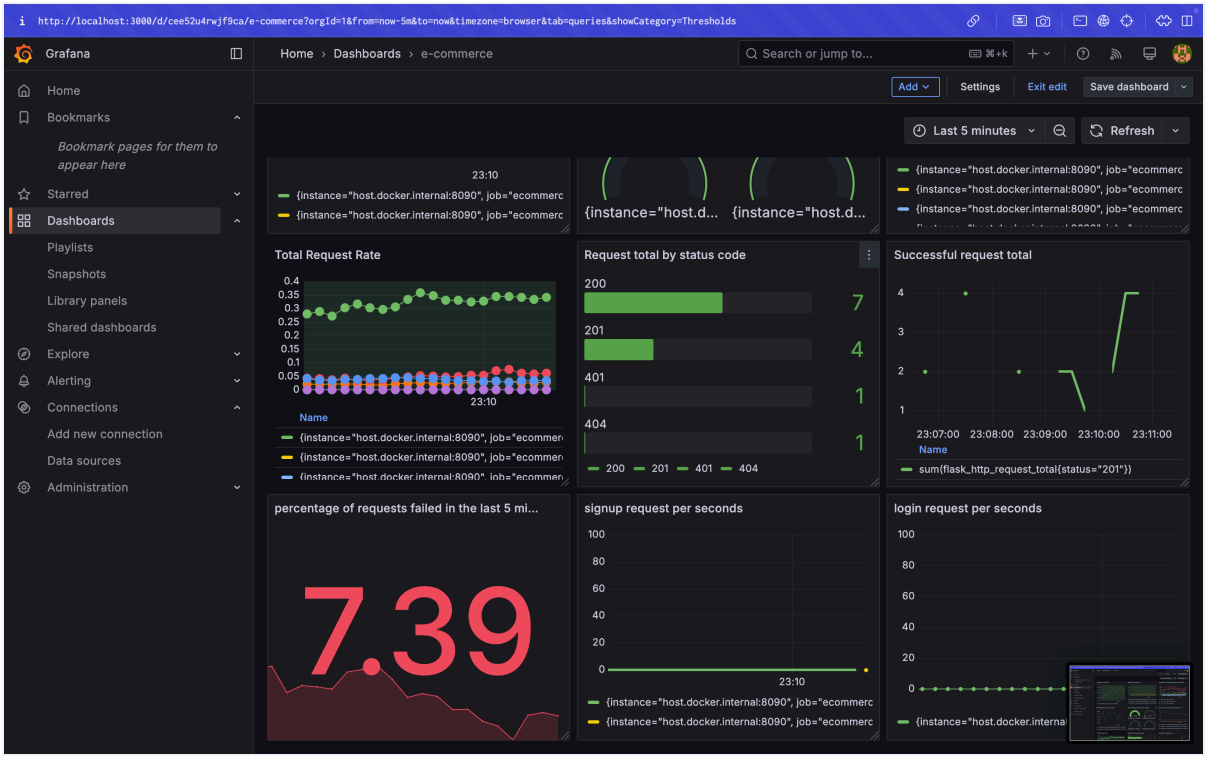


- **Add a panel and choose visualization, in the query, paste this and choose the appropriate visualization and save**
- **Query Example:**

```
sum(rate(flask_http_request_total[5m]))
```







Common Prometheus Queries

1 Total Requests

```
sum(flask_http_request_total)
```

2 Request Rate (Per Second)

```
sum(rate(flask_http_request_total[5m]))
```

3 95th Percentile Response Time

```
histogram_quantile(0.95, rate(flask_http_request_duration_seconds_bucket[5m]))
```

4 Average Response Time

```
sum(rate(flask_http_request_duration_seconds_sum[5m])) / sum(rate(flask_http_request_duration_seconds_count[5m]))
```

Troubleshooting

Issue: No Metrics Available in Prometheus

✓ Ensure Flask is running and `http://localhost:8090/metrics` is accessible. Check Prometheus `targets` at `http://localhost:9090/targets`. Restart Prometheus after updating `prometheus.yml`.

Issue: Grafana Panels Not Updating

Refresh dashboard manually. Check if Prometheus data source is connected. ✓ Adjust query time ranges.

Next Steps

Add more API endpoints & custom metrics. Set up Grafana alerts. Deploy to production using Docker.

📜 License