

# 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

### 1 Install Dependencies

Ensure you have Python installed, then run:

```
1 pip install flask prometheus_flask_exporter
```

### 2 Create a Flask App with Prometheus Exporter

```
1 from flask import Flask
2 from prometheus_flask_exporter import PrometheusMetrics
3 app = Flask(__name__)
4 metrics = PrometheusMetrics(app)
5 @app.route("/")
6 def home():
7     return "Welcome to Flask Monitoring!"
8 @app.route("/users/signup", methods=["POST"])
9 def signup():
10    return "User Signed Up", 201
11 if __name__ == "__main__":
12    app.run(host="0.0.0.0", port=5000)
```

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

The screenshot shows the VS Code interface with the following details:

- Project Explorer:** Shows the project structure under "e-commerce".
- Code Editor:** Displays the file `_init_.py` from the `src` directory. A yellow warning icon is present at the top left of the editor area.
- Bottom Status Bar:** Shows the file path `e-commerce / src / _init_.py`, the status bar text "11:6 EC e-commerce Night Owl (Material)", and system information like "LF UTF-8 4 spaces Python 3.12 (e-commerce)".

### 3 Run Flask App

```
1 python app.py
```

## Setting Up Prometheus

### 1 Install Prometheus

Setup the docker file for Prometheus and Grafana. [reference](#)

### 2 Configure prometheus.yml

Create a `prometheus.yml` file:

```
1 # global config
2 global:
3   scrape_interval: 15s
4   scrape_timeout: 10s
5   evaluation_interval: 15s
6   alerting:
7     alertmanagers:
8       - follow_redirects: true
9       enable_http2: true
10      scheme: http
11      timeout: 10s
12      api_version: v2
13      static_configs:
14        - targets: []
15   scrape_configs:
16     - job_name: prometheus
17       honor_timestamps: true
```

```

18   scrape_interval: 15s
19   scrape_timeout: 10s
20   metrics_path: /metrics
21   scheme: http
22   follow_redirects: true
23   static_configs:
24     - targets:
25       - localhost:9090
26
27 scrape_configs:
28   - job_name: 'ecommerce'
29     scrape_interval: 10s
30     metrics_path: /metrics
31     static_configs:
32       - targets:
33         - host.docker.internal:8090

```

```

global:
  scrape_interval: 15s
  scrape_timeout: 10s
  evaluation_interval: 15s
  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
  - 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

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar labeled 'Project' with a tree view containing 'prometheus-setup [prometheus]' and files like 'docker-compose.yaml', 'main.py', and 'prometheus.yml'. The main area has tabs for 'main.py', 'prometheus.yml', and 'docker-compose.yaml'. The 'docker-compose.yaml' tab is active, displaying the following YAML configuration:

```

version: '3'
services:
  prometheus:
    image: prom/prometheus
    ports:
      - 9090:9090
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    networks:
      - monitoring
  grafana:
    image: grafana/grafana
    ports:
      - 3000:3000
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=fcmvfmavmfot
    networks:
      - monitoring
networks:
  monitoring:
    external: true

```

At the bottom of the editor, there are status indicators: '17:49', 'P prometheus', 'Night Owl (Material)', 'LF', 'UTF-8', '2 spaces', 'No JSON schema', 'Python 3.12 (social-media)', and a refresh icon.

### 3 Run the Docker compose

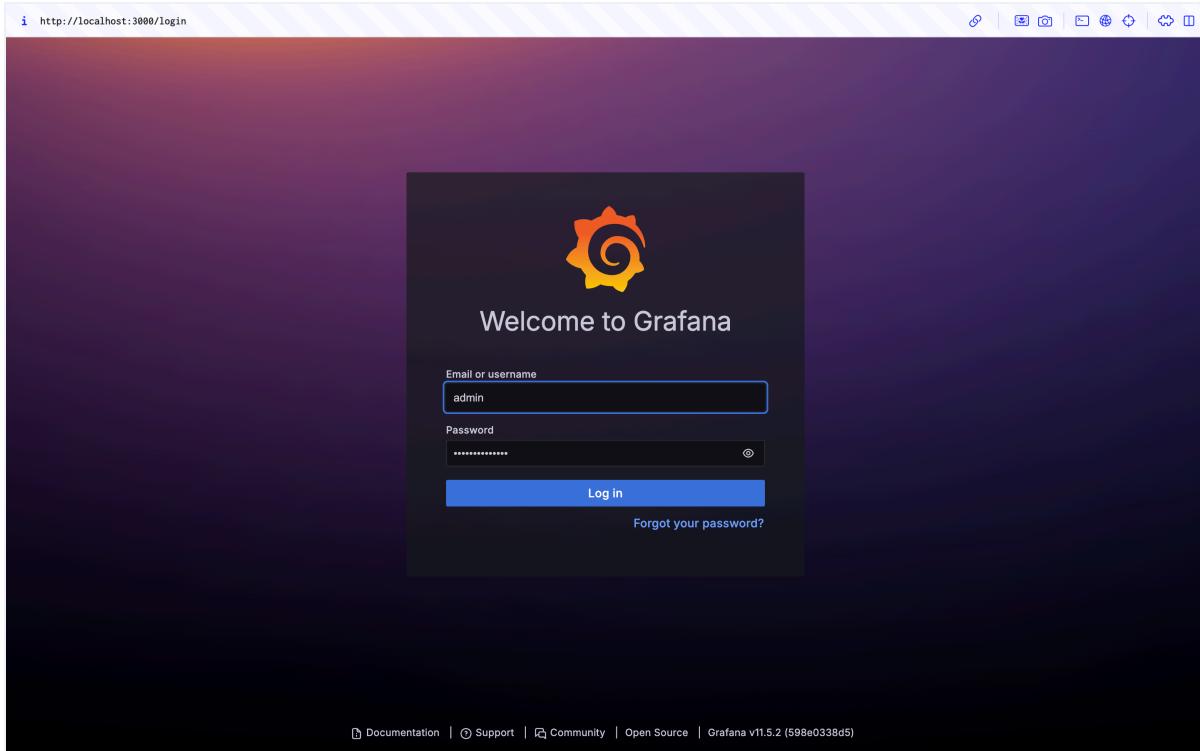
```
1 docker compose up -d
```

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

## ✓ Setting Up Grafana

### 1 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



## 2 Add Prometheus as a Data Source

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

A screenshot of the Grafana 'Add data source' configuration page. The URL in the address bar is http://localhost:3000/connections/datasources/new. The left sidebar shows a navigation menu with 'Data sources' selected. The main content area has a heading 'Add data source' and a sub-heading 'Choose a data source type'. Below this is a search bar labeled 'Filter by name or type' and a 'Cancel' button. The page is divided into sections: 'Time series databases' and 'Logging &amp; document databases'. Under 'Time series databases', there are four entries: 'Prometheus' (selected), 'Graphite', 'InfluxDB', and 'OpenTSDB'. Each entry includes a small icon, a name, a description, and a 'Core' tag. Under 'Logging &amp; document databases', there are two entries: 'Loki' and 'Elasticsearch'. The 'Loki' entry includes a description: 'Like Prometheus but for logs. OSS logging solution from Grafana Labs'.

The screenshot shows the Grafana interface for configuring a data source. The left sidebar is open, showing the 'Data sources' section. The main area is titled 'prometheus' and shows the configuration for a Prometheus data source. It includes fields for 'Name' (set to 'prometheus'), 'Prometheus server URL' (set to 'http://prometheus:9090'), and a note about configuring it below or in the config file. Below this is the 'Connection' section with the same URL. The 'Authentication' section is present but empty. A modal window at the top right says 'Configure your Prometheus data source below'.

### 3 Create Dashboards & Panels

The screenshot shows the Grafana interface for creating dashboards. The left sidebar is open, showing the 'Dashboards' section. The main area is titled 'Dashboards' and shows a message 'You haven't created any dashboards yet'. It features a large cartoon sun icon and a 'Create dashboard' button. There are search and filter options at the top.

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

1 sum(rate(flask\_http\_request\_total[5m]))

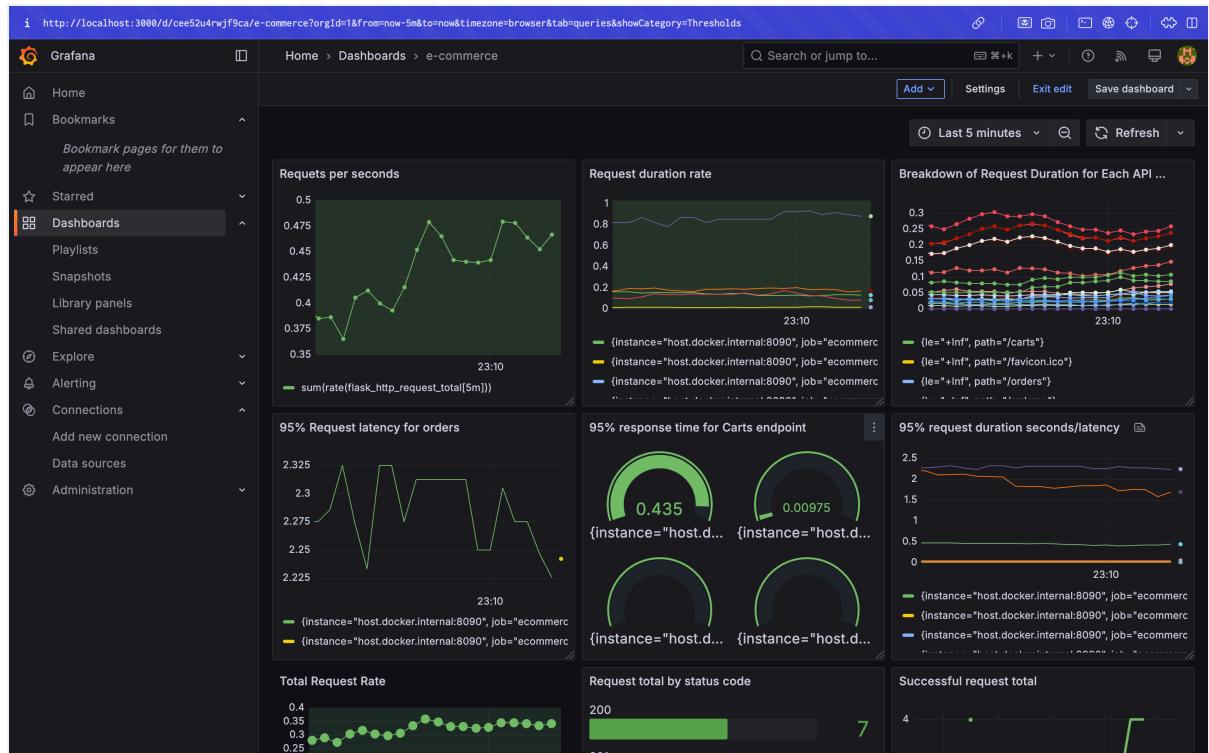
**Panel Title:** Panel Title

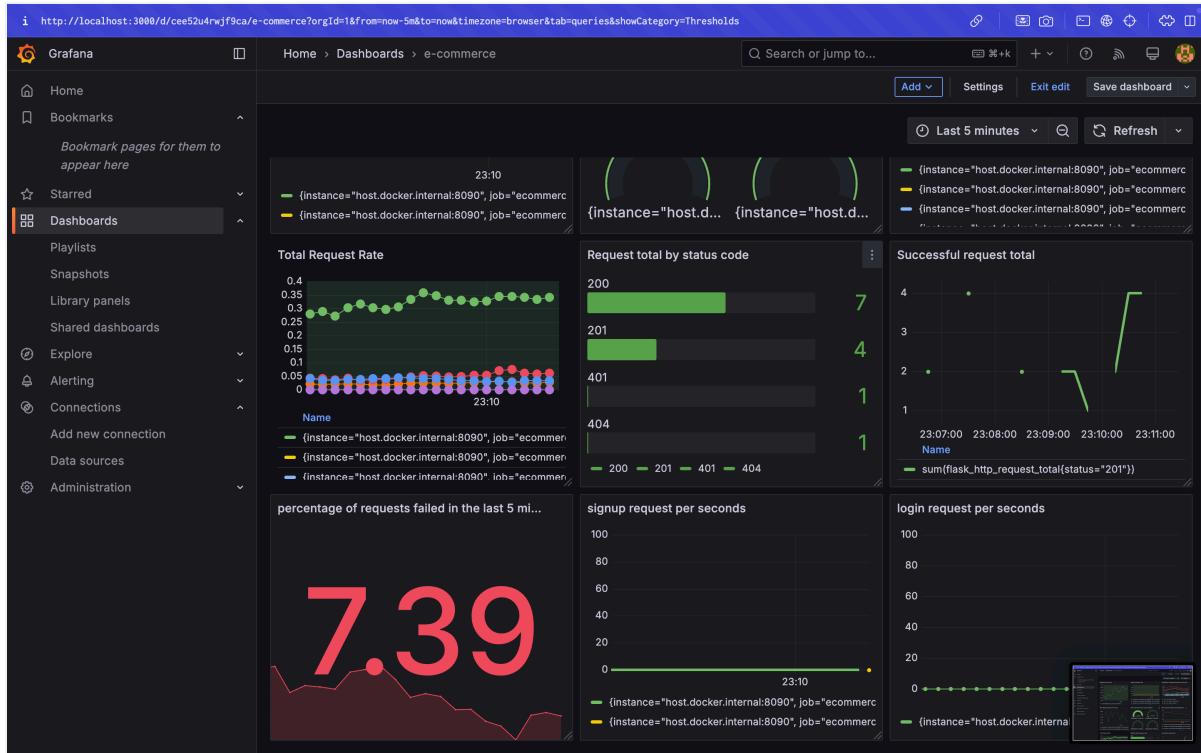
**Queries (1):**

- Data source:** prometheus
- Metric:** flask\_http\_request\_total
- Label filters:** instance = host.docker.internal:8090
- Query:** 1 flask\_http\_request\_total {instance="host.docker.internal:8090"}  
Fetch all series matching metric name and label filters.

**Panel Options:**

- Time series
- Search options
- All Overrides
- Title: Panel Title
- Description
- Transparent background
- Panel links
- Repeat options
- Tooltip
- Tooltip mode: Single All Hidden
- Hover proximity: How close the cursor must be to a point to trigger the tooltip, in pixels
- Max width





## Common Prometheus Queries

### 1 Total Requests

```
1 sum(flask_http_request_total)
```

### 2 Request Rate (Per Second)

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

### 3 95th Percentile Response Time

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

### 4 Average Response Time

```
1 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

This project is created by Racheal Kuranchie