Project Report: Log analysis with Flask, Prometheus, Loki, and Grafana

1. Requirements

- Real-time monitoring of a Flask application
- Logging and error tracking
- Visualization of metrics and logs
- Simulated log generation with latency and failure tracking
- Exportable metrics for observability
- Grafana dashboard for unified visualization
- API testing and interaction using Postman

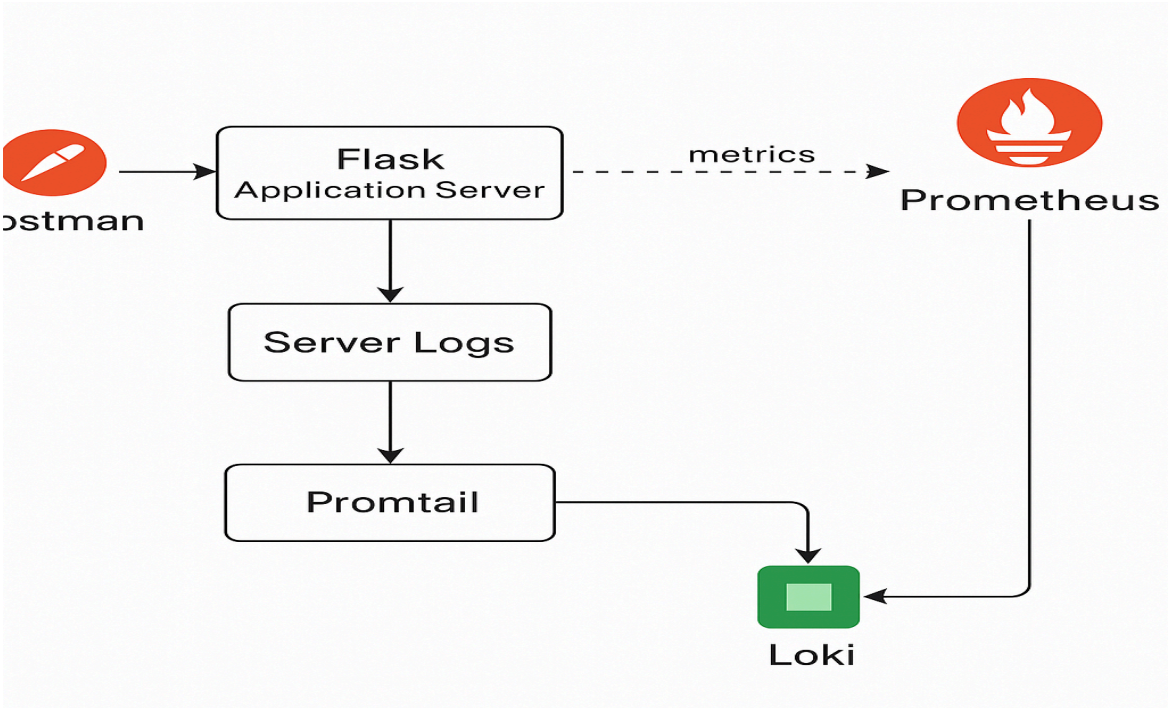
2. Technologies Used

- Flask: Python web framework for building the server.
- Prometheus: Time-series database for monitoring and metrics.
- Grafana: Visualization tool for metrics and logs.
- Loki: Log aggregation system by Grafana Labs.
- Promtail: Agent to collect logs and push them to Loki.
- Postman: API client used to test and trigger server endpoints.

3. Python Libraries Used

Library	One-liner Explanation
Flask	Lightweight web framework for Python
random	For generating randomized delays and IDs
time	To simulate delays and measure request latency
logging	Python standard logging library
prometheus_client	Exposes metrics to be scraped by Prometheus

4. Flow Diagram



5. WorkFlow

1. Flask Server

- Acts as the core application layer.
- Hosts endpoints like `/process` and `/generate_logs`.
- Each request triggers simulated delays, errors, and generates logs.
- Metrics such as request duration, error counts, and request totals are exposed via the `/metrics` endpoint.

2. Prometheus

- Continuously scrapes the Flask server's `/metrics` endpoint every second (as configured in `prometheus.yml`).
- Stores time-series data like request counts, durations, and failure rates.
- Acts as the primary monitoring tool for quantitative metrics.

- Enables alerting rules if desired (e.g., alert if failure count > threshold).

3. Grafana

- Visualizes the metrics collected by Prometheus.
- Connects to Prometheus and Loki as data sources.
- Dashboards show:
    - Real-time latency
    - Error trends
    - Traffic volume
    - Server health
- Provides unified visibility by combining metric graphs and logs.

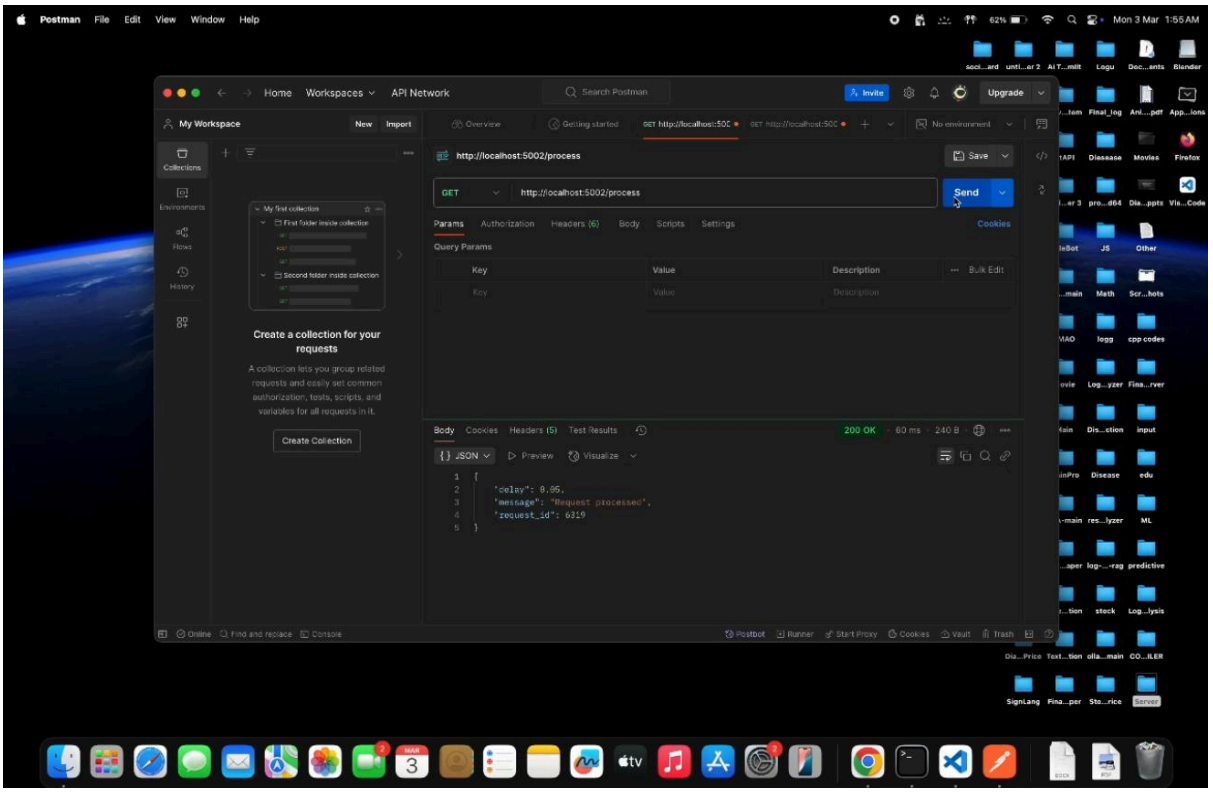
4. Logging with Promtail and Loki

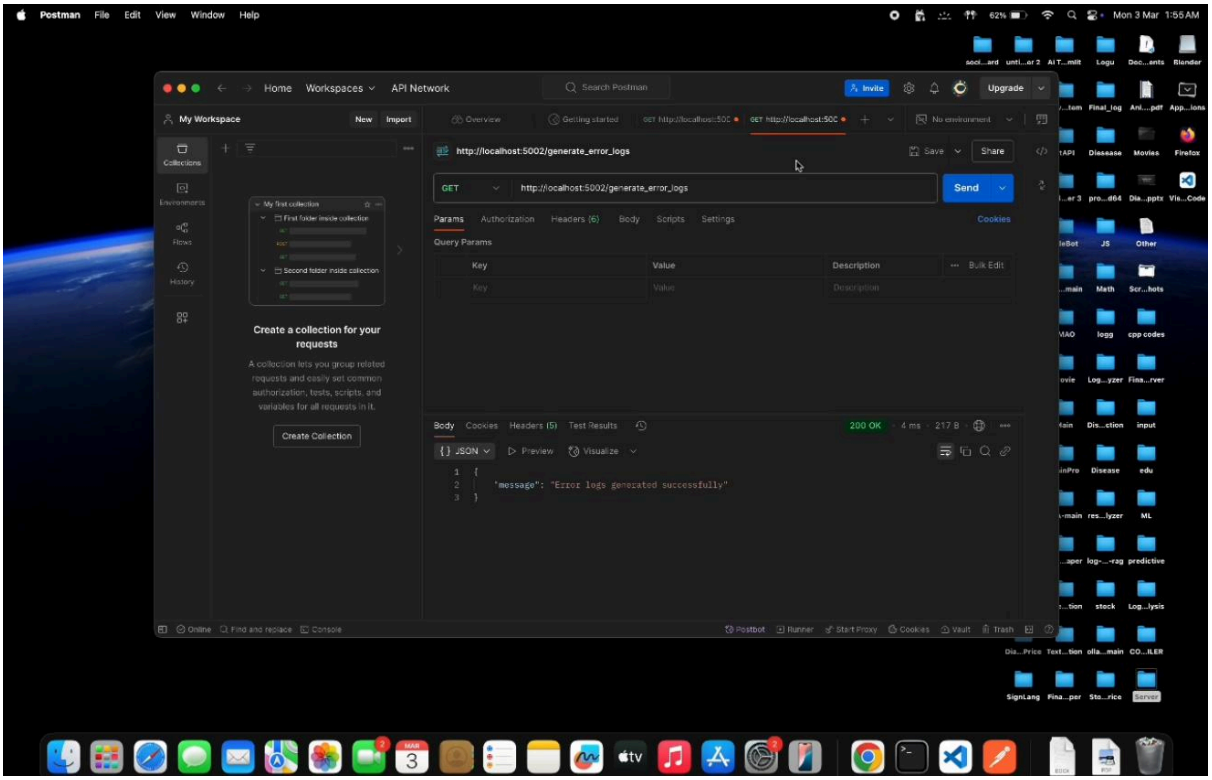
- Flask logs (e.g., in `server.log`) are continuously monitored by **Promtail**.
- Promtail tags and pushes these logs to **Loki**.
- Loki aggregates and indexes logs by job, path, and labels (like `job=service-logs`, etc.).
- Grafana connects to Loki and provides powerful log search and filtering.

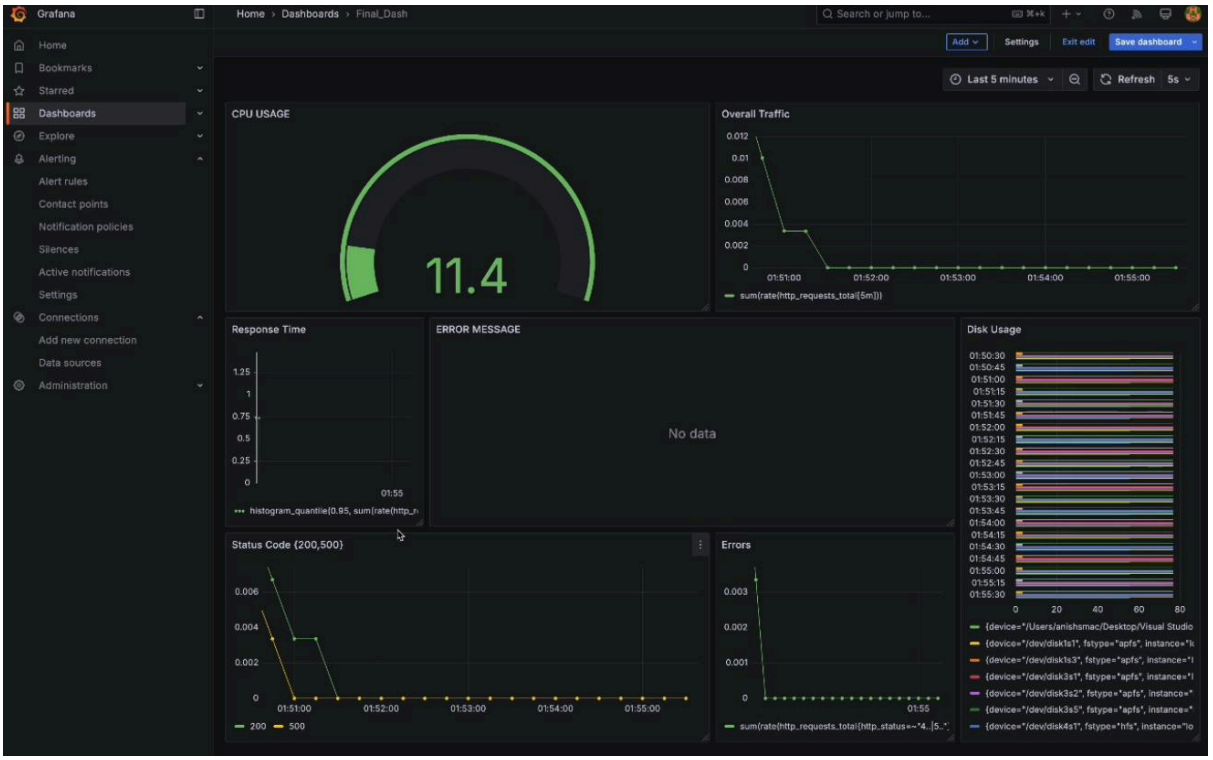
5. Postman

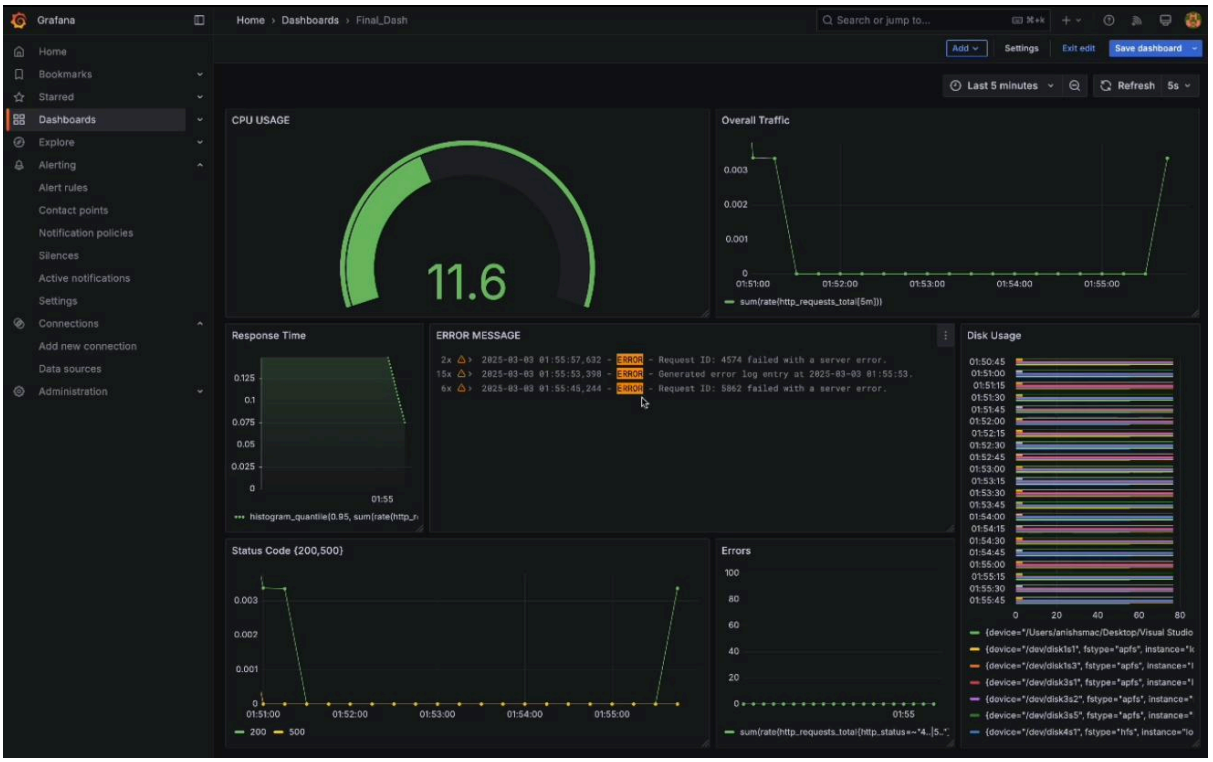
- Used as a testing tool to trigger various Flask routes:
    - `/process` simulates a request that may randomly succeed or fail.
    - `/generate_logs` creates simulated log entries for analysis.
- Helps verify that logs and metrics are generated properly.
- Useful during development and testing for endpoint validation and manual QA.

6. Screenshots






Grafana
Home › Dashboards › Final_Dash
Search or jump to...
Add
Settings
Exit edit
Save dashboard
Last 5 minutes
Refresh
5s
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Alert rules
Contact points
Notification policies
Silences
Active notifications
Settings
Connections
Add new connection
Data sources
Administration
CPU USAGE
11.4
Overall Traffic
sum(rate(http_requests_total[5m]))
Response Time
histogram_quantile(0.95, sum(rate(http_r
ERROR MESSAGE
No data
Disk Usage
Status Code {200,500}
200
500
Errors
sum(rate(http_requests_total{http_status=~"4..|5..",


Grafana
Home › Dashboards › Final_Dash
Search or jump to...
Add
Settings
Exit edit
Save dashboard
Last 5 minutes
Refresh
5s
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Alert rules
Contact points
Notification policies
Silences
Active notifications
Settings
Connections
Add new connection
Data sources
Administration
CPU USAGE
11.6
Overall Traffic
sum(rate(http_requests_total[5m]))
Response Time
histogram_quantile(0.95, sum(rate(http_r
ERROR MESSAGE
2x 2025-03-03 01:55:57,632 - ERROR - Request ID: 4574 failed with a server error.
15x 2025-03-03 01:55:53,398 - ERROR - Generated error log entry at 2025-03-03 01:55:53.
6x 2025-03-03 01:55:45,244 - ERROR - Request ID: 5862 failed with a server error.
Disk Usage
Status Code {200,500}
200
500
Errors
sum(rate(http_requests_total{http_status=~"4..|5..",