Project Report: Containerizing a Flask Application with Prometheus, Loki, Jaeger, and Grafana

1. Introduction

This project demonstrates the process of containerizing a **Flask** application and integrating it with monitoring, logging, and tracing tools: **Prometheus**, **Loki**, and **Jaeger**. The data gathered from these systems is visualized using **Grafana** dashboards, providing key insights into application performance, logs, and request traces.

2. System Architecture and Integration

- **Containerization**: The Flask app is containerized using **Docker**, enabling a consistent and isolated runtime environment. The app generates logs that aid in monitoring and troubleshooting.

- **Prometheus**: Metrics from the Flask application are scraped by **Prometheus**, providing valuable data on request counts, error rates, and latency.

- **Loki**: **Loki** collects and stores logs from the Flask app. The logs are filtered and structured to facilitate efficient querying.

- **Jaeger**: **Jaeger** traces the flow of HTTP requests, providing insights into the performance and interactions of services.

- **Grafana**: A **Grafana** dashboard aggregates data from Prometheus, Loki, and Jaeger, offering real-time visualizations of metrics, logs, and traces.

3. Docker-Compose Setup

A **docker-compose.yml** file is used to orchestrate all services involved in the project, including:

- **Flask Application**: The containerized backend service.

- **Prometheus**: The tool responsible for scraping and storing metrics.

- **Loki**: Collects and stores logs from the application.

- **Jaeger**: Traces HTTP requests made to the Flask app.

- **Grafana**: Displays the collected data through dynamic dashboards.

This configuration simplifies the management and deployment of the entire stack.

4. Prometheus Integration

Prometheus scrapes the `/metrics` endpoint exposed by the Flask application. It gathers essential metrics such as total request count, response latency, and error rates.

- **Prometheus Metrics Screenshots**:
  - Screenshots showcasing Prometheus metrics at different intervals (5-minute and 6-hour) to demonstrate performance trends over time.

5. Loki for Log Management

Logs generated by the Flask app are collected by **Loki**, which scrapes the logs and stores them for querying. Logs are formatted to allow for easy filtering, including the use of JSON for better data organization.

- **Log Screenshots**:
  - **Normal Logs**: A view of unfiltered logs from the Flask app.
  - **JSON-Filtered Logs**: A structured view of logs, optimized for efficient querying in Grafana.

6. Jaeger for Distributed Tracing

Jaeger is integrated to trace HTTP requests within the Flask application. Tracing allows for a detailed, end-to-end view of the requests, helping to identify potential bottlenecks or latency issues across the application.

- **Jaeger Trace Visualization**:
  - A screenshot visualizing traces of HTTP requests, showing how the application handles requests and how services interact.

7. Grafana Dashboards

Grafana is used to display the data collected from Prometheus, Loki, and Jaeger on interactive dashboards. These dashboards allow for easy monitoring and visualization of key metrics, logs, and traces.

The dashboards include:

- **Prometheus Metrics**: Graphs and visualizations of key application performance metrics.
- **Loki Logs**: A real-time view of the logs generated by the Flask app.
- **Jaeger Traces**: Visual traces showing how HTTP requests move through the system and the response times involved.

Grafana Dashboards JSON Export:

- The dashboards have been exported as JSON files, enabling easy import into any Grafana instance to replicate the setup.

8. Insights and Observations

- **Prometheus Metrics**:
    - Analysis of how the Flask application performs over time. Insights are derived from metrics like request counts, latencies, and error rates observed at both short (5 minutes) and long (6 hours) intervals.
- **Loki Logs**:
    - Logs provide valuable information about application behavior, including error patterns and performance anomalies.
- **Jaeger Traces**:
    - Tracing data provides insights into request flow and highlights any performance bottlenecks or areas of the application requiring optimization.

9. Conclusion

The project successfully integrates a **Flask** application with **Prometheus**, **Loki**, **Jaeger**, and **Grafana** to enable effective monitoring, logging, and tracing. The containerization of the app ensures consistency across environments, while the integration of Prometheus, Loki, and Jaeger provides in-depth visibility into the application's performance, logs, and request handling. The **Grafana** dashboards offer a unified view, allowing for easy analysis and troubleshooting.

10. Deliverables

- **docker-compose.yml**: The configuration file that defines the setup for all services.
- **Dashboards JSON**: The exported Grafana dashboards in JSON format.
- **Log Samples**: Screenshots of logs, both normal and JSON-filtered.
- **Trace Visualizations**: Screenshots from Jaeger displaying the traces of HTTP requests.
- **Report**: This document, detailing the setup and insights derived from the project.