Project Report: Complete Observability System (Metrics, Logs & Traces)

1. Introduction

Modern applications require continuous monitoring to ensure reliability, performance, and quick issue resolution. Observability plays a vital role in DevOps by providing deep visibility into systems through three core pillars: metrics, logs, and traces. This project implements a complete observability system using open-source tools like Prometheus, Grafana, Loki, and Jaeger to monitor a containerized Python web application.

2. Abstract

The goal of this project is to build an end-to-end observability stack that provides real-time metrics visualization, centralized logging, and distributed tracing. A sample Python application running in Docker emits data, which is collected and analyzed by observability tools. This setup helps in detecting bottlenecks, understanding system behavior, and troubleshooting effectively. All components are containerized and orchestrated using Docker Compose, ensuring easy deployment and scalability.

3. Tools Used

- Docker & Docker Compose: To containerize and orchestrate services.
- Prometheus: For collecting and querying application metrics.
- Grafana: To visualize metrics, logs, and traces via dashboards.
- Loki: For log aggregation and searching.
- Jaeger: For capturing distributed traces.
- Python (Flask): Sample web application emitting observability data.
- OpenTelemetry & Prometheus Client: For exporting trace and metric data from the app.

4. Steps Involved in Building the Project

1. Environment Setup:
   - Installed Docker Desktop for Windows.
   - Created project directories and required YAML/config files.

2. Python Application Development:
   - Developed a sample Flask app that logs requests, exposes Prometheus metrics, and sends trace data to Jaeger.

3. Docker Compose Setup:

- Defined services in docker-compose.yml for: app, prometheus, grafana, loki, jaeger.
- Set up network and volume configurations.

4. Configuration:
   - Configured Prometheus (prometheus.yml) to scrape the app.
   - Connected Grafana to Prometheus, Loki, and Jaeger as data sources.
   - Imported JSON dashboards into Grafana for visualization.

5. Execution:
   - Ran docker-compose up --build to launch all services.
   - Accessed Grafana on http://localhost:3000 to monitor system metrics, logs, and traces.

6. Testing:
   - Sent sample requests to the Python app.
   - Verified real-time logs, metrics, and traces in Grafana.

5. Conclusion

This project demonstrates how to implement a robust observability solution using open-source tools. By integrating Prometheus, Loki, Jaeger, and Grafana with a Python app, we achieved full-stack monitoring. This setup enables efficient debugging, performance tuning, and service health monitoring — essential for any production-grade DevOps environment. The entire system is modular, scalable, and easy to deploy via Docker, making it a valuable addition to any infrastructure.