Project Report: Integrated Monitoring System for Containerized Applications

Objective

To build an **integrated observability stack** using open-source tools — **Prometheus**, **Grafana**, **Loki**, and **Jaeger** — for comprehensive monitoring of a containerized application. The system tracks:

- **Performance metrics** with Prometheus
- **Centralized logs** with Loki
- **Request tracing** with Jaeger
- **Dashboards and visualizations** via Grafana

Tools & Technologies

Tools	Purpose
Flask(Python)	Sample web application
Grafana	Dashboard and visualization layer
Prometheus	Metrics collection and alerting
Loki	Centralized logging system
Promtail	Scrape logs from app.log
Jaeger	Distributed tracing
Docker Compose	Container orchestration

Architecture Overview

Each component is deployed as a Docker container. The Flask app exposes Prometheus metrics and structured logs. Prometheus scrapes metrics, Loki collects logs via a log driver or Promtail, and Jaeger traces requests through instrumentation. Grafana unifies all observability signals in a single UI.

Key Features:

- Multi-source data integration
- Modular Docker Compose setup
- Pre-provisioned dashboards and data sources
- Real-time insights into request rate, latency, error rate, and logs

Project Structure

```
Project
├── Dockerfile
├── app-logs
│   └── app.log
├── app.py
├── docker-compose.yml
├── grafana
│   └── provisioning
│       ├── dashboards
│       │   ├── dashboards.yaml
│       │   ├── loki-dashboard.json
│       │   └── sample-dashboard.json
│       └── datasources
│           └── datasources.yaml
├── images
│   └── file.txt
├── local-config.yaml
├── loki-data
│   ├── 00000036
│   ├── 00000037
│   ├── checkpoint.000035
│   │   └── 00000000
│   └── checkpoint.000036.tmp
│       └── 00000000
├── prometheus.yml
└── promtail-config.yml
```

Implementation Steps

1. **Built a sample Flask application** instrumented with:
   - Prometheus metrics (http_requests_total, http_request_duration_seconds)
   - OpenTelemetry tracing for Jaeger
   - Structured logs for Loki
2. **Configured Docker Compose stack** with the following services:
   - app: Flask app container
   - prometheus: Scrapes metrics from app
   - grafana: Displays dashboards with all integrations
   - loki: Ingests logs from app containers
   - jaeger: Receives and displays distributed traces
3. **Provisioned Grafana** with:
   - Preloaded dashboards (sample-dashboard.json, loki-dashboard.json)
   - Preconfigured data sources (Prometheus, Loki, Jaeger)

Observability Highlights

Metrics (Prometheus)

- **Request Rate**: rate(http_requests_total[1m])
- **Latency Distribution**: histogram_quantile(0.95, rate(http_request_duration_seconds_bucket[5m]))
- **Error Rate**: rate(http_requests_total{status=~"5.."}[1m])

Logs (Loki)

- Filter logs by container, level, or keyword:
- {container="app"} |= "ERROR"

Traces (Jaeger)

- Visualize span timings per HTTP request
- Correlate slow requests with metrics and logs

Troubleshooting:

Problem Summary

After deploying the stack using docker-compose, **Grafana Logs UI was not showing any logs or labels** from the Loki data source. Promtail and Loki appeared to be running, but no log data was visible in Grafana.

Investigation & Root Cause

1. Loki container was **exiting with error**:

   ```
   error running loki: creating WAL folder at "/wal": permission denied
   ```

   Solution: Added volume mount for Loki.

2. Even after fixing Loki, **no logs appeared** in Grafana:
   - Promtail was running
   - No log labels showed up
   - `app-logs/` folder was empty
3. After inspecting the app container:

   `/app/logs/app.log` did not exist

   Root cause: Application was **not writing logs to file**, so Promtail had nothing to collect.

Fixes Applied

Application Changes

- Modified app to **write logs to `/app/logs/app.log`**
- Ensured logs are written using Python's `logging` module:

  ```
  logging.basicConfig(filename='/app/logs/app.log', level=logging.INFO)
  ```

Docker Compose Adjustments

- Mounted `./app-logs` directory from host:

  ```
  volumes:

    - ./app-logs:/app/logs
  ```

- Mounted same directory in Promtail:

  ```
  volumes:
    - ./app-logs:/var/log/app:ro
  ```

Promtail Configuration

Updated `promtail-config.yml`:

```
__path__: /var/log/app/*.log
labels:
  job: app
```

Instance Recommendation

For development:

- t3.medium (2 vCPU, 4 GB RAM) is suitable

For light production:

- t3.large or t3.xlarge if you expect more traffic or logs

Project Outcomes

- Deployed a unified monitoring stack with zero manual dashboard setup
- Enabled proactive observability for application and infrastructure
- Demonstrated how to correlate **metrics**, **logs**, and **traces** in one interface

Future Improvements

- Add alerting rules in Prometheus
- Use Promtail for broader log collection
- Integrate Slack or email alerting in Grafana
- Auto-scale infrastructure with Terraform + AWS