Real-Time E-Commerce Application Observability

Project Documentation

---

1. Project Overview

This project demonstrates the implementation of a **full observability stack** for a modern e-commerce application. It covers metrics collection, dashboard visualization, alerting, and centralized logging by integrating open-source tools including **Prometheus**, **Grafana**, **Alertmanager**, and the **ELK stack (Elasticsearch, Logstash, Kibana)**.

---

2. Goals

- Build a simple e-commerce web application with realistic user interactions.
- Instrument the app and infrastructure for monitoring.
- Create dashboards for real-time visualization of metrics.
- Configure alerts for operational issues.
- Centralize logs for error analysis and trend identification.
- Deploy the stack publicly with secure access.

---

3. Technology Stack

Layer	Tooling	Purpose
Application	Node.js	Backend API serving e-commerce features

Monitoring	Prometheus	Metrics collection
Visualization	Grafana	Dashboards & insights
Alerting	Alertmanager	Alert routing and notifications
Logging	ELK Stack (Elasticsearch, Logstash, Kibana)	Log ingestion, indexing, and visualization
Infrastructure	cAdvisor	Container and host-level metrics collection
Web Server	Nginx	Reverse proxy and HTTPS termination

---

4. Application Details

The e-commerce application exposes the following key endpoints:

- `/products`: Lists available products.
- `/cart`: Adds/removes products to/from cart.
- `/checkout`: Simulates purchase and payment.

Instrumentation:

- Integrated Prometheus client libraries to expose:
    - Request count per endpoint.
    - Request latency.
    - Error counts.
- Logs structured as JSON with levels (INFO, WARN, ERROR).

---

5. Observability Components

5.1 Metrics Collection

- Prometheus scrapes metrics from:
    - The e-commerce application (via `/metrics` endpoint).
    - Infrastructure metrics from Node Exporter and cAdvisor.

5.2 Visualization

- Grafana dashboards include:
    - Request volume and latency heatmaps.
    - Error rate graphs.
    - Host resource usage (CPU, memory).
    - Container health metrics.

5.3 Alerting

- Prometheus alert rules trigger for:
    - Error rate > 5% per endpoint.
    - Request latency exceeding 2 seconds.
    - CPU usage above 80%.
- Alertmanager routes notifications to Slack and email.
- Silencing configured during scheduled maintenance.

5.4 Logging

- Logstash parses application and container logs.
- Logs indexed and searchable via Elasticsearch.

- Kibana dashboards visualize:
    - Error frequency trends.
    - Keyword and exception searches.
    - Volume of logs by service.

---

6. Deployment & Security

- The entire stack and app are deployed on an AWS EC2 instance.
- Nginx reverse proxy serves:
    - Application APIs.
    - Grafana dashboards with anonymous read-only access.
- HTTPS secured using Let's Encrypt certificates.
- Firewall configured to allow only necessary ports.

---

7. Configuration and Automation

- Docker Compose orchestrates the stack for ease of deployment.
- Configuration files version-controlled in GitHub.
- Auto-restart policies enabled for critical services.
- Retention policies applied to Prometheus and Elasticsearch data.

---

8. How to Run the Project

Prerequisites:

- Docker and Docker Compose installed.
- AWS/DigitalOcean account (or local VM).

Steps:

Clone the repository:

```
 git clone https://github.com/yourusername/ecommerce-observability.git
cd ecommerce-observability
```

1.

Start the stack:

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

2.
3. Access:
    - E-commerce app: `https://your-domain.com/api`
    - Grafana dashboards: `https://your-domain.com/grafana`
    - Kibana logs: `https://your-domain.com/kibana`

---

9. Results & Outcomes

- Real-time monitoring of user activity and system health.
- Proactive alerting ensures quick response to performance degradation or errors.
- Centralized logs simplify troubleshooting and analysis.
- Public dashboards showcase your DevOps skills in observability.