Cloud-Native Real-Time Earthquake Monitoring Using USGS Data Feeds

José Raúl Luna Bermúdez
Universidad de Costa Rica
San José, Costa Rica
Email: raullunab12@gmail.com

***Abstract*—Increasing access to high-quality seismic data and the need for timely situational awareness have created opportunities to engineer cloud-native infrastructures for real-time earthquake monitoring [1]. This project deploys a Kubernetes-based architecture that ingests, processes, and visualizes USGS GeoJSON feed data in near real time [2]. The approach validates open-source, scalable, and reproducible infrastructure for global seismic awareness and creates the foundation for future distributed real-time analytics.**

***Index Terms*—Real-time earthquake monitoring, Kubernetes, OpenSearch, Grafana, Kafka, Cloud-native**

Contexto, motivación y justificación

El monitoreo sísmico es una actividad crítica para la gestión del riesgo y la comprensión del comportamiento geodinámico del planeta. La disponibilidad creciente de datos abiertos proporcionados por instituciones como el USGS permite a la comunidad académica desarrollar soluciones tecnológicas innovadoras sin depender de infraestructura física de detección. Al mismo tiempo, la industria ha adoptado arquitecturas cloud-native que facilitan la construcción de sistemas distribuidos, altamente escalables y resilientes, particularmente adecuados para el procesamiento de datos en tiempo real.

La motivación de este proyecto surge de la necesidad de explorar cómo estas tecnologías abiertas pueden aplicarse en un entorno educativo para diseñar y desplegar un flujo completo de monitoreo sísmico global utilizando únicamente componentes libres. Esto permite a estudiantes e investigadores adquirir competencias en áreas como microservicios, observabilidad, streaming de datos y visualización geoespacial, mientras se aprovecha un caso de uso real con alto valor social.

El proyecto se justifica como una oportunidad de aprendizaje aplicada, en la cual se combina sismología computacional con ingeniería de infraestructura moderna. Asimismo, sirve como base para investigaciones futuras que busquen optimizar detección de eventos, análisis automatizado o incluso alertamiento temprano conforme se adquieran conocimientos más avanzados.

I. Antecedentes

La evolución del monitoreo sísmico ha sido impulsada por flujos de datos en tiempo real, arquitecturas distribuidas y aplicaciones en la nube. Proyectos como *QuakeFlow* han demostrado que los modelos de aprendizaje automático en Kubernetes permiten detectar mayor cantidad de eventos con baja latencia [1].

El uso de *Distributed Acoustic Sensing* (DAS) ha permitido extender la cobertura sísmica usando fibra óptica existente [3]. En paralelo, el USGS ha desarrollado servicios públicos mediante APIs y formatos GeoJSON para facilitar la consulta y análisis de eventos sísmicos globales [4], [2], [5], [6].

Objetivos de investigación

El proyecto planteó como objetivo general diseñar e implementar una arquitectura distribuida capaz de ingerir, procesar y visualizar datos sísmicos globales en tiempo cercano al evento utilizando únicamente herramientas y servicios open-source desplegados en Kubernetes.

Los objetivos específicos establecidos fueron:

- Consumir datos sísmicos públicos del USGS mediante sus servicios GeoJSON en tiempo real.
- Implementar un pipeline cloud-native basado en microservicios y mensajería distribuida.
- Procesar datos sísmicos en flujo para su enriquecimiento y transformación.
- Almacenar eventos de manera estructurada para consultas analíticas.
- Visualizar la actividad sísmica global mediante dashboards interactivos.

Si bien se consiguió cumplir los objetivos relacionados con ingesta, almacenamiento inicial y visualización operacional, las metas vinculadas al procesamiento avanzado en flujo y análisis OLAP no pudieron completarse dentro del tiempo disponible debido a la complejidad técnica de herramientas como Apache Flink y ClickHouse. Esto evidencia una brecha de conocimiento inicial que afectó la alineación total con los objetivos propuestos, pero que al mismo tiempo fundamenta claramente el trabajo futuro recomendado.

II. Introducción

El incremento de datos sísmicos requiere sistemas resilientes, escalables y observables en tiempo real. Tecnologías como ClickHouse y Grafana permiten análisis dinámico y visualización operativa [7], [8].

OpenSearch Dashboards habilita filtros geoespaciales para identificar patrones de actividad sísmica [9]. La monitorización de Kafka y almacenamiento analítico se soporta con Prometheus y Grafana [10], [11].

El objetivo de este trabajo es diseñar y validar una arquitectura cloud-native para ingesta y visualización de datos sísmicos de USGS en tiempo cercano al evento.

III. Marco Teórico

El monitoreo sísmico en tiempo real requiere procesar grandes volúmenes de datos para mantener catálogos actualizados y confiables [1]. Para lograrlo, se emplean arquitecturas distribuidas que garanticen resiliencia y baja latencia en la comunicación entre servicios.

Kafka es una pieza fundamental en este tipo de pipelines, ya que permite la ingesta continua y tolerante a fallos de datos sísmicos transmitidos en flujo [10]. En paralelo, herramientas analíticas como ClickHouse han demostrado su capacidad para manejar escalas masivas de información con tiempos de consulta óptimos [12], integrándose a Grafana para visualización de métricas operativas en tiempo real [7].

A nivel de visualización geoespacial, OpenSearch Dashboards facilita la exploración de patrones de actividad sísmica mediante mapas interactivos y filtros dinámicos [9]. Para asegurar el funcionamiento continuo de la infraestructura, la observabilidad se implementa con Prometheus y Grafana, habilitando la recolección centralizada de métricas [11], [13]. Además, OpenTelemetry permite trazabilidad distribuida sobre microservicios modernos [14].

Finalmente, avances en redes DAS amplían el alcance del monitoreo al aprovechar infraestructura de fibra óptica existente [3], mientras que el USGS proporciona datos sísmicos abiertos y estandarizados mediante APIs y formatos GeoJSON [4], [2], [6]. Este ecosistema tecnológico habilita la creación de sistemas distribuidos para análisis sísmico con impacto educativo y social.

IV. Metodología

La arquitectura implementada se despliega sobre Kubernetes y consiste en:

- Microservicio Python que consume Feeds GeoJSON del USGS [6].
- Procesamiento ligero y publicación de datos en Kafka.
- Persistencia temporal en PostgreSQL.
- Observabilidad integrada con Prometheus y Grafana.
- Visualización geoespacial con OpenSearch Dashboards.

Por restricciones de tiempo no fue posible implementar Apache Flink ni ClickHouse, quedando planificados como mejoras futuras.

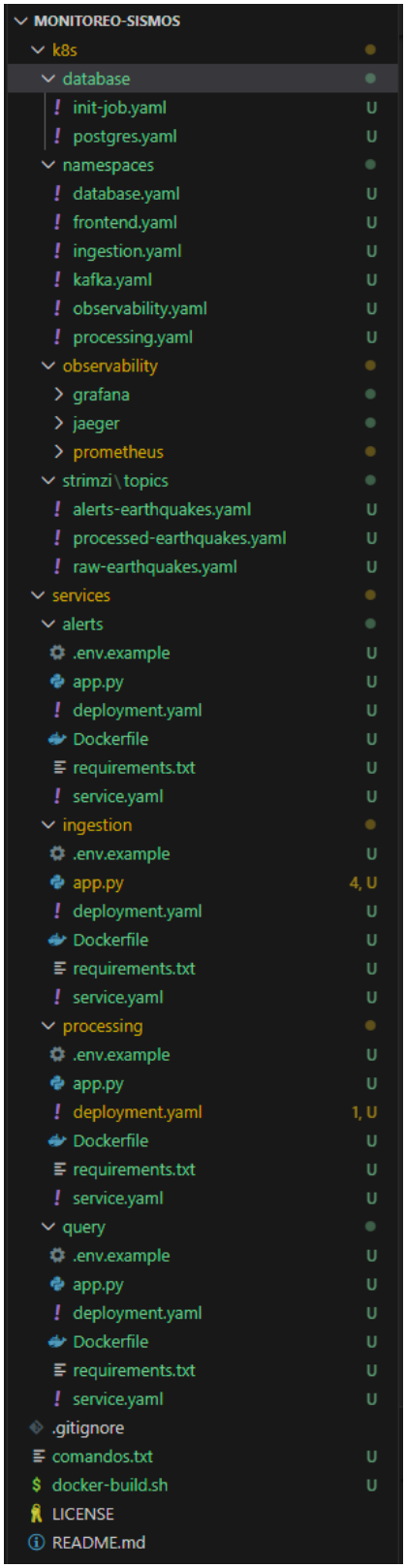

Fig. 1. Arquitectura cloud-native utilizada en el proyecto.

V. Resultados

La Fig. 2 presenta el tablero de monitoreo funcional, que muestra:

- Mapa de calor con densidad sísmica global,
- Histogramas de magnitudes,
- Últimos sismos con hora local/UTC y ubicación.

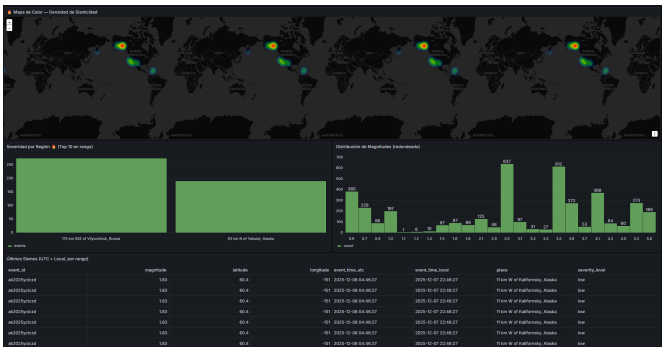

Fig. 2. Dashboard operacional mostrando sismicidad global en tiempo cercano al evento.

Durante las pruebas se mantuvo una ingesta continua con actualizaciones periódicas y baja latencia, demostrando la efectividad de Kubernetes para soportar pipelines sísmicos distribuidos.

VI. Conclusiones

El presente proyecto logró demostrar una arquitectura funcional para el monitoreo sísmico global en tiempo cercano al evento mediante tecnologías cloud-native y datos abiertos del USGS. La solución implementada validó la correcta ingesta, transformación inicial y visualización geoespacial de eventos sísmicos utilizando únicamente software libre.

Sin embargo, no fue posible completar el alcance técnico inicialmente definido debido a una brecha de conocimiento en herramientas avanzadas del ecosistema cloud-native, particularmente en lo referente al procesamiento distribuido (Apache Flink) y análisis OLAP masivo (ClickHouse). La curva de aprendizaje asociada a estas tecnologías resultó más elevada de lo estimado para el tiempo disponible.

Adicionalmente, Kubernetes presentó retos importantes en un entorno local, incluyendo consumo elevado de recursos, fallos de networking y necesidad de configuración detallada, lo cual ralentizó el avance operativo y aumentó el tiempo requerido para el despliegue estable de los servicios necesarios. Estos aspectos limitaron la capacidad de profundizar en análisis avanzados durante la fase de pruebas.

Como trabajo futuro, se recomienda:

- integrar Apache Flink para habilitar análisis en flujo con ventanas de tiempo,
- incorporar ClickHouse para mejorar el rendimiento y escalabilidad analítica,
- reforzar la trazabilidad distribuida mediante OpenTelemetry,
- migrar progresivamente a un entorno cloud administrado para reducir complejidad operativa.

A pesar de las limitaciones identificadas, los resultados obtenidos constituyen una base sólida y extensible para continuar con el desarrollo de sistemas educativos u operacionales de alerta sísmica, alineados con mejores prácticas modernas de ingeniería en la nube.

References

[1] W. Zhu, A. B. Hou, R. Yang, A. Datta, S. M. Mousavi, W. L. Ellsworth, and G. C. Beroza, "QuakeFlow: A Scalable Machine-learning-based Earthquake Monitoring Workflow with Cloud Computing," Aug. 2022, arXiv:2208.14564. [Online]. Available: http://arxiv.org/abs/2208.14564

[2] USGS, "GeoJSON Summary Format." [Online]. Available: https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php

[3] E. Biondi, G. Tepp, E. Yu, J. K. Saunders, V. Yartsev, M. Black, M. Watkins, A. Bhaskaran, R. Bhadha, Z. Zhan, and A. L. Husker, "Real-time processing of distributed acoustic sensing data for earthquake monitoring operations," May 2025, arXiv:2505.24077. [Online]. Available: http://arxiv.org/abs/2505.24077

[4] USGS, "API Documentation - Earthquake Catalog." [Online]. Available: https://earthquake.usgs.gov/fdsnws/event/1/

[5] ——, "GeoJSON Detail Format." [Online]. Available: https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson_detail.php

[6] ——, "Real-time Notifications, Feeds, and Web Services." [Online]. Available: https://earthquake.usgs.gov/earthquakes/feed/

[7] ClickHouse, "ClickHouse data source plugin for Grafana | ClickHouse Docs." [Online]. Available: https://clickhouse.com/docs/integrations/grafana

[8] Grafana, "ClickHouse integration | Grafana Cloud documentation." [Online]. Available: https://grafana.com/docs/grafana-cloud/monitor-infrastructure/integrations/integration-reference/integration-clickhouse/

[9] OpenSearch, "OpenSearch Dashboards," Sep. 2025. [Online]. Available: https://docs.opensearch.org/latest/dashboards/

[10] Confluent, "Monitor Apache Kafka Clusters with Prometheus, Grafana, and Confluent." [Online]. Available: https://www.confluent.io/blog/monitor-kafka-clusters-with-prometheus-grafana-and-confluent/

[11] R. Hodges, "Monitoring ClickHouse® on Kubernetes with Prometheus and Grafana," May 2020. [Online]. Available: https://altinity.com/blog/2020-5-5-monitoring-clickhouse-kubernetes-prometheus-grafana

[12] ClickHouse, "Scaling our Observability platform beyond 100 Petabytes by embracing wide events and replacing OTel." [Online]. Available: https://clickhouse.com/blog/scaling-observability-beyond-100pb-wide-events-replacing-otel

[13] Grafana, "How to monitor Clickhouse with Grafana Cloud." [Online]. Available: https://grafana.com/blog/2023/05/30/how-to-start-monitoring-your-clickhouse-instance-or-cluster-with-grafana-cloud/

[14] A. Yurchenko, "Building a Local Observability Stack: A Journey with OpenTelemetry, ClickHouse, and Grafana," May 2025. [Online]. Available: https://dev.to/alex_yurchenko_c2d664c0a/building-a-local-observability-stack-a-journey-with-opentelemetry-clickhouse-and-gra