{desafío} latam_

CURSO

DEVOPS SENIOR

Objetivo General del Curso

DISEÑAR ENTORNOS CLOUD NATIVE INTEGRANDO PRÁCTICAS DE KUBERNETES Y GITOPS, DE ACUERDO CON ESTÁNDARES DE SEGURIDAD Y OBSERVABILIDAD.

Objetivo específico del Módulo

DISTINGUIR CONCEPTOS DE OBSERVABILIDAD INTEGRAL Y TRACING, DE ACUERDO A LAS PRACTICAS AVANZADAS DE GITOPS, DEVSECOPS, KUBERNETES, OBSERVABILIDAD, IAC, FINOPS Y AIOPS.

Contenidos

Módulo 4: OBSERVABILIDAD AVANZADA.

Capítulo 1: Prometheus, Grafana, Open Telemetry, Distributed Tracing (Jeager, Tempo).

En entornos distribuidos modernos, la gestión de errores, optimización de desempeño y detección de fallos no puede depender de procesos manuales ni de herramientas aisladas. La observabilidad surge como una disciplina avanzada, mucho más allá del monitoreo tradicional, permitiendo entender no solo qué está fallando, sino por qué y dónde.

Esta capacidad es crítica en arquitecturas basadas en microservicios, Kubernetes, pipelines GitOps y CI/CD.

Fundamentos de la Observabilidad Completa

La observabilidad moderna en DevOps se estructura sobre tres pilares complementarios:

1. **Métricas:** valores cuantificables recolectados en intervalos (CPU, memoria, QPS, tasa de error).

2. **Logs estructurados:** registros detallados de eventos, errores, advertencias y flujos internos.

3. **Trazas distribuidas (distributed traces):** representaciones gráficas de una transacción que atraviesa múltiples servicios, con detalle por segmento.

Cada uno ofrece una vista parcial. Solo la integración de los tres permite diagnósticos de alta precisión en sistemas complejos.

1. Prometheus: Recolección de métricas de series temporales

¿Qué es Prometheus?

Prometheus es un sistema de monitoreo open source creado por SoundCloud y actualmente parte de la CNCF. Funciona mediante una arquitectura pull-based, extrayendo métricas desde endpoints definidos, en formato Prometheus exposition format.

Arquitectura

- **Prometheus Server:** motor central de scraping y almacenamiento.
- **Exporters:** exponen métricas de aplicaciones, sistemas, bases de datos, etc.
- **Alertmanager:** canaliza y agrupa alertas configuradas en PromQL.
- **TSDB (Time Series Database):** motor interno optimizado para datos temporales.

Ejemplo de métrica:

http_requests_total{method="GET",status="200",handler="/login"}

Flujo operativo:

1. El servicio expone /metrics con las métricas.
2. Prometheus scrappea el endpoint periódicamente.
3. Se aplican reglas de alerta.
4. Los datos se consultan mediante PromQL o se visualizan vía Grafana.

Usos avanzados:

- Monitoreo granular por pod, container o namespace en Kubernetes.
- Análisis de capacidad predictiva (basado en evolución de uso de recursos).
- Tiempos de build/deploy por pipeline.
- Detección automática de targets vía service discovery.

2. Grafana: Visualización avanzada y alerta multicanal

¿Qué es Grafana?

Grafana es una plataforma de observabilidad y visualización multiplataforma, altamente extensible mediante plugins y dashboards personalizados. Es el estándar de visualización en la mayoría de las empresas DevOps.

Características destacadas:

- Soporte nativo para más de 50 fuentes de datos: Prometheus, Tempo, Loki, ElasticSearch, PostgreSQL, etc.
- Dashboards temáticos y componibles.
- Alertas visuales y por umbral con lógica compuesta.
- Control de acceso granular y auditoría de usuarios.

Integraciones clave:

- **Kubernetes Dashboard:** panel de nodos, pods, deployments, uso de CPU/RAM, errores.

- Alertas vía Slack, Teams, PagerDuty, correo, Webhooks.

- Análisis de series temporales + correlación con eventos externos (por ejemplo, cambios de código o despliegues).

3. OpenTelemetry: Instrumentación universal

¿Qué es OpenTelemetry (OTel)?

OpenTelemetry es un estándar abierto y modular respaldado por CNCF para la instrumentación de aplicaciones, capaz de capturar métricas, trazas y logs en una arquitectura distribuida.

Componentes principales:

- SDKs por lenguaje (Go, Java, Python, JavaScript, .NET).
- **Collector:** componente central que recibe, procesa y exporta los datos a múltiples backends.
- **Instrumentación automática o manual:** según lenguaje y framework.

Exportación hacia:

- Jaeger, Tempo, Zipkin (tracing)
- Prometheus, New Relic, Datadog (metrics)
- Elastic, Splunk, Loki (logs)

Ejemplo: Instrumentación de microservicio

- Cada servicio incluye el SDK de OpenTelemetry.
- Las trazas se agrupan en spans por función, endpoint o llamada externa.
- Los datos son recolectados por el collector y enviados al backend.

4. Distributed Tracing: Jaeger y Tempo

¿Qué es el tracing distribuido?

Permite seguir una única solicitud (trace ID) a través de múltiples servicios, midiendo latencias, errores, cuellos de botella y dependencia entre componentes.

Ejemplo:

Una petición GET a /checkout podría activar:

- Frontend → API Gateway → AuthService → CartService → PaymentService.

Cada uno genera un span, y todos conforman el trace completo.

Jaeger

- Herramienta de trazabilidad distribuida desarrollada por Uber.
- Visualiza cada trace y permite filtrar por duración, errores, usuario, operación.
- Soporta múltiples formatos: OpenTracing, OpenTelemetry, Zipkin.
- Integrable vía sidecar, SDK o proxy.

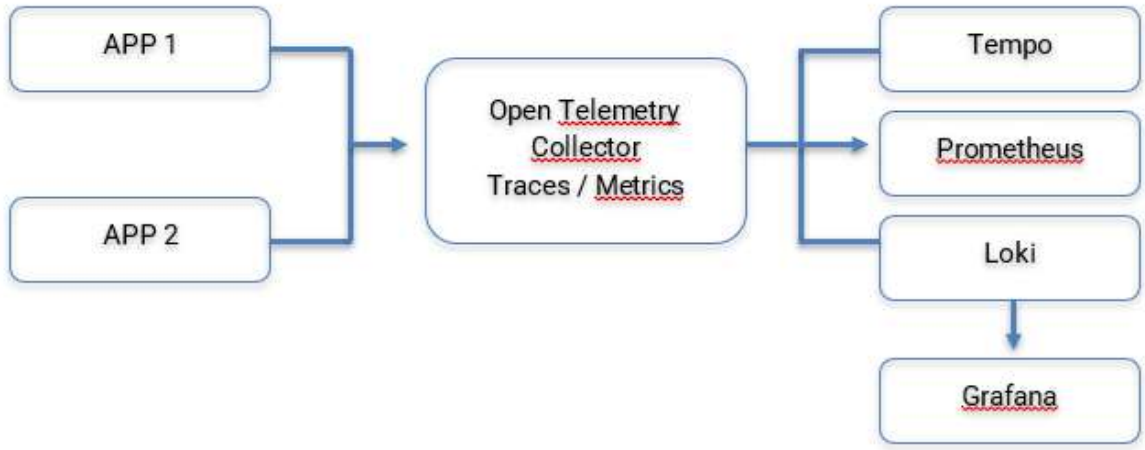
Tempo

- Solución de tracing de Grafana Labs.
- No requiere base de datos (usa almacenamiento de objetos: S3, GCS, etc.).
- Integración perfecta con Grafana y Loki (logs).
- Alta escalabilidad y bajo costo operativo.

Ejemplos de uso:

- Rastrear demoras en servicios críticos (latencia > p95).
- Detectar fallas esporádicas asociadas a componentes específicos.
- Correlacionar trazas con alertas de Prometheus y logs de Loki.

Arquitectura de Observabiliad Unificada



Rol del DevOps Senior

El DevOps senior debe ser capaz de:

Diseñar arquitecturas de observabilidad que cubran todo el ciclo de vida del software.

Aplicar instrumentación avanzada con OpenTelemetry en microservicios y funciones serverless.

Establecer alertas basadas en SLIs/SLOs reales, no solo en métricas crudas.

Correlacionar métricas, logs y trazas para análisis forense o postmortem.

Entrenar a los equipos para responder con base en evidencia observable, no en suposiciones.