Engineer

OpenTelemetry

💓 enviar telemetría sin depender de un proveedor

Qué es

OpenTelemetry es un estándar de código abierto, con bibliotecas y herramientas correspondientes, para producir trazas, métricas y registros del software en ejecución y entregarlos al sistema que los almacene y muestre. La mayoría de los ingenieros lo llaman OTel. Es un proyecto de la Cloud Native Computing Foundation, nacido al fusionarse dos iniciativas anteriores, OpenTracing y OpenCensus.

Lo más útil que conviene saber es lo que deja fuera. OpenTelemetry no es un backend de observabilidad: no conserva datos, no dibuja paneles ni envía alertas. Esa parte la hacen Jaeger, Prometheus y las plataformas comerciales. Por tanto, «¿OpenTelemetry o Prometheus?» compara un conducto con un destino, y la respuesta habitual es ambos: OpenTelemetry donde se producen los datos y un backend elegido por el equipo donde se guardan.

El proyecto consta de varias piezas que encajan entre sí:

- una **especificación** que define la API, el SDK y los datos para todos los lenguajes;
- **OTLP**, el protocolo que transporta telemetría por gRPC o HTTP, codificada con Protocol Buffers;
- **convenciones semánticas**, nombres compartidos para elementos compartidos, de modo que una ruta HTTP o una sentencia de base de datos tenga la misma etiqueta en Go, Java y Python;
- una **API y un SDK por lenguaje**, con bibliotecas de instrumentación e instrumentación automática para marcos populares;
- el **Collector**, un proceso separado que recibe telemetría, la procesa y la exporta hacia el siguiente destino.

De las señales que transporta, las trazas son las que más explicación necesitan. Una traza sigue una petición por todos los servicios que toca. Cada paso es un span: un nombre, una hora de inicio, una duración, un conjunto de atributos y la identidad del span que lo provocó. Cuando la petición pasa de un servicio al siguiente, esa identidad viaja con ella en una cabecera, de forma predeterminada la que define W3C Trace Context:

```
traceparent: 00-a0892f3577b34da6a3ce929d0e0e4736-f03067aa0ba902b7-01
```

Los cuatro campos son la versión del formato, el ID de traza compartido por todos los spans de la petición, el ID del span que hace la llamada y un conjunto de indicadores, donde un `01` final marca la traza como muestreada. Las métricas y los registros completan la imagen, y una cuarta señal, baggage, lleva pequeños datos de clave y valor junto con la petición para que un servicio situado tres saltos más allá aún pueda verlos.

Utilice el concepto en una decisión

Aplique «¿Qué es OpenTelemetry?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Monitorización y alertas que avisan cuando un fallo aún es pequeño, y cifras que muestran con qué frecuencia se mantiene así.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Primer panel de observabilidad de la empresa — la primera visión de la empresa sobre sus propios sistemas

Primer panel de observabilidad de la empresa Póngase en contacto →

Por qué importa

El almacenamiento es el coste visible de la observabilidad. El más persistente es la instrumentación: alguien tiene que añadir llamadas a cada servicio, en cada lenguaje, y mantenerlas al día conforme cambia el código. Cuando cada proveedor distribuía su propio agente y su propia API, cambiar de proveedor significaba repetir todo ese trabajo. Con OpenTelemetry, el código llama a una API, y la elección del backend pasa a la configuración, ya sea un exportador en el SDK o una ruta en el Collector. El proyecto enuncia directamente el principio: los datos que genera un sistema pertenecen a quien lo opera.

Esa propiedad tiene efectos prácticos además de contractuales. El Collector se sitúa entre las aplicaciones y el backend, lo que lo convierte en el lugar para filtrar, muestrear, renombrar o eliminar atributos antes de que salga nada y para

enviar el mismo flujo a dos backends mientras se realiza una migración. Pasar de un proveedor a otro se convierte en un cambio de configuración del Collector en vez de en cien servicios.

El volumen es el coste que hay que planificar. Cada petición puede producir decenas de spans, y con tráfico real la factura de conservarlos todos puede superar la del servicio observado. El muestreador predeterminado del SDK, `ParentBased(root=AlwaysOn)`, muestrea cada traza nueva y sigue la decisión de quien llama para el resto, por lo que, tal como viene configurado, se conserva todo. Cuánto conservar es una decisión que el equipo toma deliberadamente.

La madurez varía según la señal y el lenguaje. OTLP es estable para trazas, métricas y registros, mientras que el SDK de cada lenguaje avanza a su propio ritmo, y la página de estado del proyecto indica dónde está cada uno. Léala antes de decidirse por él para un servicio escrito en un lenguaje menos habitual.

Y no siempre compensa. Una sola aplicación con registros estructurados y unas pocas métricas gana poco con el trazado distribuido. OpenTelemetry justifica su coste cuando una petición cruza límites entre procesos, pasando por un frontend, una API, una cola y una base de datos, y responder a «¿en qué se fue el tiempo?» requiere una reunión.

Cómo se aplica en la práctica

En una plataforma, la monitorización se construyó en lugar de comprarse, y OpenTelemetry es su mitad de trazado. El trazado utiliza OTLP para seguir una petición lenta por el frontend, la API y la base de datos en lugar de hacer conjeturas. Junto a él funciona un flujo de procesamiento de errores propio, compuesto por un servicio errmon y otro de ingestión, que sanea las cargas útiles antes de almacenarlas, agrupa errores en problemas recurrentes, detecta picos y regresiones con un período de espera para que un despliegue defectuoso no envíe cuarenta avisos a la persona de guardia, resuelve los símbolos de las trazas de pila minificadas del frontend y ejecuta un latido sintético para demostrar que el propio flujo de procesamiento sigue vivo.

Los resultados llegan a la propia base de datos de la plataforma como eventos de error, grupos de errores, una bandeja de entrada, latencia de API y muestras de pilas. Los operadores trabajan con 11 vistas sobre esos datos, una de ellas dedicada a objetivos de nivel de servicio. La página es clara sobre el precio: construirlo requirió una inversión real de tiempo y añadió otro elemento que

mantener, mientras que una herramienta comprada de seguimiento de errores habría estado funcionando aquella misma tarde. A cambio, los datos sensibles permanecen en la plataforma. También las reglas de alerta se escribieron para esta plataforma en vez de para una genérica.

Dónde surgen los problemas

La traza se detiene en un límite. El contexto cruza fácilmente las llamadas HTTP porque las bibliotecas de instrumentación escriben y leen la cabecera en ambos extremos. Una cola de mensajes, una tarea programada o un proxy que elimine cabeceras desconocidas rompe la cadena salvo que alguien transporte el contexto a mano. La traza se divide en dos, y la mitad lenta parece no guardar relación con la petición que la causó. Incluya el contexto en los metadatos del mensaje y compruebe que todos los servicios ejecuten el mismo propagador.

Los atributos filtran información o se multiplican sin control. Es fácil añadir un atributo de span, y los valores más fáciles de añadir son los equivocados: correos electrónicos, URL completas con tokens en la cadena de consulta o SQL que aún contiene valores literales. En las métricas, el peligro es la cardinalidad. Un atributo con millones de valores distintos, como un ID de usuario, multiplica el número de series temporales y, con él, la factura. Siga las convenciones semánticas, mantenga los identificadores fuera de las métricas y oculte los datos sensibles en el Collector como segunda línea de defensa.

El muestreo ocurre por accidente. El muestreo al inicio decide al comienzo de una petición, antes de saber si será lenta o fallará, por lo que conservar el 1 % de las trazas también conserva aproximadamente el 1 % de las escasas trazas lentas. El muestreo al final decide después de ver juntos los spans de una traza y puede conservar todos los errores y todas las peticiones lentas, pero el componente que lo realiza debe mantener todos esos datos mientras espera. Es un servicio con estado cuya propia capacidad hay que planificar y vigilar.

Se confía ciegamente en el flujo de procesamiento. OTLP confirma la entrega salto a salto; la entrega de extremo a extremo queda fuera de su alcance. Un Collector que no dispone de memoria suficiente o se retrasa en su cola de exportación descarta telemetría, y las aplicaciones que se la envían continúan como si todo fuera bien. Vigile las propias métricas del Collector, ejecute más de uno y trate un hueco en las trazas como un fallo que explicar en lugar de como una hora tranquila.

Lecturas adicionales

- ¿Qué es OpenTelemetry? — definición del propio proyecto, sus componentes y lo que deliberadamente no es.
- Especificación OTLP — protocolo de transmisión: transportes, codificación, compresión y confirmación de recepción.
- Trace Context — recomendación del W3C en la que se basa la cabecera `traceparent`.
- Propagación de contexto — cómo sigue una traza una petición a través de límites entre procesos.
- Convenciones semánticas — nombres compartidos para atributos, métricas y recursos.
- Muestreo — muestreo al inicio y al final, y el coste de ejecutar cada uno.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es OpenTelemetry?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/opentelemetry/