Engineer

Observabilidad

entender un sistema desde fuera por los datos que emite

Qué es

La observabilidad es el grado en que un equipo puede entender desde fuera lo que hace un sistema en ejecución, utilizando solo los datos que este emite, sin abrirlo ni saber de antemano qué ha fallado. La introducción del proyecto OpenTelemetry lo plantea en términos de preguntas: un sistema observable permite hacer preguntas sobre él sin conocer su funcionamiento interno, incluidas preguntas que nadie se había planteado cuando se escribió el software. La fórmula breve de esa introducción para la pregunta que debería poder responder es «¿Por qué está ocurriendo esto?».

Eso hace de la observabilidad una propiedad del sistema, no un producto. Una herramienta puede almacenar y mostrar datos, pero antes esos datos tienen que salir de la aplicación. La documentación de OpenTelemetry es clara: para que un sistema sea observable, sus componentes deben estar instrumentados, es decir, su código debe emitir señales. Después fija un criterio práctico. Una aplicación se considera correctamente instrumentada cuando sus señales existentes permiten a los desarrolladores investigar un problema sin cambiar primero la instrumentación.

Esos datos se llaman telemetría, y OpenTelemetry los clasifica en señales:

- **Las trazas** registran el recorrido de una petición por el sistema. Una traza se compone de spans, cada uno una unidad de trabajo con un nombre, un tiempo de inicio y de fin y atributos como el método y la ruta HTTP. El primer span es la raíz y abarca la petición de principio a fin.
- **Las métricas** son mediciones recogidas durante la ejecución y agregadas a lo largo del tiempo: tasa de errores, tasa de peticiones o uso del procesador.
- **Los registros** son anotaciones de eventos con marca temporal. Muchos no están vinculados a una petición concreta.

- **Baggage** es información contextual que se transmite junto con una petición para que otras señales puedan utilizarla.

La página de señales también enumera los eventos, un tipo específico de registro, y los perfiles, registros del uso de recursos a nivel de código, como elementos en desarrollo o en fase de propuesta.

Lo que convierte tres montones de datos en algo que un equipo puede interrogar es la correlación. OpenTelemetry propaga contexto entre servicios, de modo que un span de un servicio sabe a qué traza pertenece, y puede incorporar los identificadores de la traza y del span activos a cada registro. Cada señal también incluye un recurso: atributos que identifican la entidad que la produce, como el nombre del servicio, el host o el contenedor. Con esas claves compartidas, un equipo puede pasar de una métrica a las peticiones que hay detrás y de una petición a las líneas de registro que produjo.

La observabilidad y la monitorización se solapan, pero no son lo mismo. La monitorización vigila condiciones que alguien decidió de antemano que eran relevantes y genera una alerta cuando aparece alguna. La observabilidad es aquello con lo que puede trabajar el equipo una vez que ha saltado la alerta y la causa es algo que nadie había previsto.

Utilice el concepto en una decisión

Aplique «¿Qué es la observabilidad?» 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

Los sistemas modernos fallan de formas difíciles de predecir. Una petición puede pasar por una puerta de enlace, varios servicios, una cola y una base de datos antes de recibir respuesta, y un fallo en cualquiera de ellos puede manifestarse como lentitud en otro lugar. La introducción considera esencial el rastreo distribuido para estos sistemas, porque sus problemas suelen ser no

deterministas o demasiado complejos para reproducirlos en el equipo de un desarrollador. Cuando un fallo no se puede reproducir, la telemetría registrada mientras ocurría es la única evidencia disponible.

Los paneles y las alertas responden a preguntas previstas. Eso es valioso y cubre los fallos que un equipo ya ha visto. Los incidentes difíciles son los nuevos, donde la pregunta no es si subieron las tasas de error, sino por qué empezaron a fallar peticiones de una región, a través de una versión, hacia un proveedor de pagos. Si responder exige escribir código nuevo, desplegarlo y esperar a que reaparezca el problema, la interrupción dura tanto como ese ciclo completo.

La observabilidad también mantiene al usuario en primer plano. La introducción define la fiabilidad como el hecho de que el servicio haga lo que esperan los usuarios y da un ejemplo: una tienda podría estar siempre disponible y seguir siendo poco fiable si al pulsar para añadir zapatos negros al carrito no se añaden siempre zapatos negros. Una métrica del host nunca mostraría eso. La telemetría a nivel de petición sí puede hacerlo y es la base de los indicadores de nivel de servicio: mediciones desde el punto de vista del usuario, que los objetivos de nivel de servicio vinculan después con las necesidades del negocio.

Por último, cambia el coste de un incidente. Un equipo que puede seguir una petición lenta hasta la llamada exacta que la retrasó dedica su tiempo a corregir en vez de hacer conjeturas, y el análisis posterior puede citar evidencias en lugar de recuerdos.

Cómo se aplica en la práctica

Considere una tienda en línea inventada. Desde las 14:00, una pequeña parte de las finalizaciones de compra tarda más de dos segundos. Nada está caído, no ha saltado ninguna alerta y no se creó ningún panel para este caso. Así lo investigaría un equipo con señales correlacionadas. Todos los nombres, identificadores y números que aparecen a continuación se han inventado para el ejemplo.

El primer paso es el histograma de latencia de la finalización de compra, medida en segundos. En el formato de texto OpenMetrics, un intervalo puede incluir un ejemplar después de `#` : una referencia a datos externos a la métrica, normalmente el identificador de una traza correspondiente a ese intervalo. Dos de los intervalos se muestran así:

```
checkout_bucket{le="1.0"} 95
checkout_bucket{le="2.5"} 97 # {trace_id="3f1c9a77e2b04d18a6c5e0b9d4127f60"} 2.3
```

El segundo paso sigue ese identificador de traza. La traza muestra en qué se emplearon los 2,3 segundos, span a span:

```
POST /checkout                      2.30 s   service checkout
  SELECT cart                       0.04 s   service checkout
  POST /authorise                   2.21 s   service payments
    event: provider timeout, retrying
  INSERT order                      0.03 s   service checkout
```

El tercer paso recupera las líneas de registro con el mismo identificador de traza. La especificación de OpenTelemetry para registros escritos en formatos distintos de su propio protocolo pide registrar los identificadores como campos de nivel superior llamados `trace_id` y `span_id` en JSON, en minúsculas y codificados en hexadecimal:

```
{
    "timestamp": 1793110051.402,
    "severity": "WARN",
    "body": "provider timeout after 2000 ms, retrying",
    "provider": "acquirer-b",
    "trace_id": "3f1c9a77e2b04d18a6c5e0b9d4127f60",
    "span_id": "a91c04e7d2f3b658"
}
```

Tres saltos, y la pregunta que nadie había previsto tiene respuesta: uno de los dos proveedores de pagos empezó a agotar el tiempo de espera, y cada compra que se le enviaba esperaba un reintento. Ahora se puede agrupar por el mismo atributo, `provider`, en todas las trazas recientes para confirmar el patrón antes de que nadie toque el código.

Nada de eso funciona si los servicios no se identifican. Los SDK de OpenTelemetry establecen el atributo de recurso `service.name` en `unknown_service` de forma predeterminada, y la documentación recomienda establecerlo explícitamente, en el código o mediante una variable de entorno. Se pueden añadir otros atributos de recurso de la misma manera:

```
env OTEL_SERVICE_NAME=checkout \
    OTEL_RESOURCE_ATTRIBUTES=deployment.environment.name=production \
    ./checkout
```

En este sitio, tres entradas del portafolio muestran cómo se fue construyendo la observabilidad por etapas. Una describe el primer panel de observabilidad de una empresa, que recopilaba métricas del sistema como CPU, memoria y disco de servidores Linux remotos por SSH; más adelante se creó una segunda versión sobre Grafana y Prometheus. El trabajo para reducir el riesgo operativo con Grafana y Prometheus instrumentó los servicios clave y eligió métricas significativas en lugar de cifras de vanidad. Una entrada posterior, monitorización de errores y trazado propios, añadió trazado OpenTelemetry sobre OTLP, de modo que una petición lenta pueda seguirse por el frontend, la API y la base de datos en lugar de recurrir a conjeturas, junto con un flujo de procesamiento de errores que sanea las cargas útiles antes de almacenarlas.

Dónde surgen los problemas

Comprarla en vez de incorporarla al sistema. Una herramienta nueva a la que no llega nada útil es una sala vacía. La observabilidad nace de la instrumentación del código y las bibliotecas, así que presupueste tiempo de ingeniería para añadir spans, atributos y campos de registro, no solo la licencia.

Tres señales, tres islas. Las métricas en una herramienta, los registros en otra y las trazas en una tercera, sin una clave compartida, dejan al equipo copiando marcas temporales de una pantalla a otra. Incluya los identificadores de traza y span en cada registro, adjunte ejemplares a los histogramas de latencia donde la pila los admita y dé a cada señal los mismos atributos de recurso.

Todos los servicios se llaman unknown_service. Es el valor predeterminado del SDK, y no se pueden distinguir los servicios de una flota que lo comparte. Establezca `OTEL_SERVICE_NAME` para cada servicio. El recurso se adjunta al crear los proveedores de trazas y métricas y no puede cambiarse después, así que debe ser correcto desde el arranque.

Solo se instrumentan los extremos. La instrumentación sin código es un buen comienzo y registra lo que ocurre en los extremos de una aplicación: en sus bibliotecas y en el entorno donde se ejecuta. Las preguntas de un incidente suelen referirse al negocio: qué nivel de cliente, qué proveedor o qué valor de

una bandera de funcionalidad. Para eso hacen falta atributos añadidos en el código. La documentación de OpenTelemetry presenta la instrumentación mediante código como complemento de la instrumentación sin código, y ambas pueden funcionar a la vez.

Instrumentar durante el incidente. Si la respuesta exigió una nueva línea de registro y un despliegue, el sistema no estaba correctamente instrumentado según el criterio de la propia introducción. Convierta «¿qué tuvimos que añadir?» en una pregunta fija de cada revisión de incidente y trate cada respuesta como una tarea de seguimiento.

Detalle sin límites en la señal equivocada. Los identificadores de usuario, las URL completas y los cuerpos de petición pertenecen a los spans y los registros, no a las etiquetas de métricas, donde cada valor nuevo crea otra serie temporal. El estándar OpenMetrics limita los nombres y valores de etiquetas de un ejemplar a 128 caracteres en total, precisamente para impedir que los ejemplares transporten datos de spans, y permite que un receptor descarte por completo los ejemplares. Compruebe que los enlaces sobreviven a todo el recorrido desde el exportador hasta la pantalla antes de confiar en ellos.

Telemetría que filtra información. Los spans y los registros recogen el contenido de las peticiones, que puede incluir direcciones, nombres y tokens. Decida qué atributos pueden registrarse, elimine el resto antes del almacenamiento y proteja la telemetría con el mismo cuidado que la base de datos que describe.

Lecturas adicionales

- Introducción a la observabilidad — definición del proyecto OpenTelemetry, con fiabilidad, SLI y SLO, spans y trazas distribuidas.
- Señales — trazas, métricas, registros y baggage, y las señales aún en desarrollo.
- Instrumentación — instrumentación mediante código y sin código, y cómo la propagación de contexto enlaza las señales.
- Recursos — atributos de recurso, el valor predeterminado `unknown_service` y las variables de entorno que los establecen.
- Contexto de traza en formatos de registro distintos de OTLP — especificación estable para registrar identificadores de traza y span en syslog, texto plano y registros JSON.

- OpenMetrics 1.0 — estándar del formato de exposición, incluidos los ejemplares y sus recomendaciones sobre cardinalidad.

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 la observabilidad?» 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/observability/