Los 3 pilares de la observabilidad: métricas, logs y trazas

Durante muchos años la administración de sistemas se centró casi exclusivamente en **monitorizar servidores individuales**. El objetivo principal era comprobar si una máquina estaba encendida, si el disco tenía espacio o si la CPU estaba saturada. En arquitecturas tradicionales —monolíticas y alojadas en pocos servidores— ese enfoque solía ser suficiente: si algo fallaba, bastaba con revisar el servidor concreto y examinar algunos archivos de registro.

La situación cambió radicalmente con la adopción de **arquitecturas distribuidas**, especialmente con el auge de los **microservicios**, los contenedores y plataformas de orquestación como Kubernetes. En estos entornos una sola aplicación puede estar formada por decenas o incluso cientos de servicios que se comunican entre sí. Cuando surge un problema, el fallo puede encontrarse en cualquier punto de esa cadena: una API intermedia, un servicio de autenticación, una cola de mensajes o una base de datos. El modelo clásico de monitorización deja entonces de ser suficiente, porque ya no existe un único lugar donde buscar el error.

De esta necesidad surge el concepto moderno de **observabilidad**, que propone comprender el comportamiento interno de sistemas complejos a partir de los datos que generan. En la práctica, este enfoque se articula en torno a tres fuentes principales de información que permiten responder a preguntas distintas sobre lo que ocurre en el sistema.

El primer pilar son las **métricas**, que indican el *cuánto*. Herramientas como Prometheus recopilan valores numéricos agregados sobre el estado de la infraestructura y las

aplicaciones: consumo de CPU, número de peticiones, latencia media o tasas de error. Las métricas permiten detectar rápidamente anomalías globales —por ejemplo, un aumento repentino de errores HTTP 500 o una saturación de recursos— y actúan como un sistema de alerta temprana que indica que algo no está funcionando correctamente.

El segundo componente son los **registros o logs**, que responden al *qué*. Plataformas como Grafana Loki o Elastic Stack almacenan los mensajes generados por las aplicaciones durante su ejecución. En ellos aparece el detalle concreto de lo ocurrido: excepciones, mensajes de error, eventos internos o información contextual sobre las operaciones realizadas. Si las métricas indican que existe un problema, los logs permiten identificar el mensaje exacto que describe el fallo.

El tercer pilar corresponde a las **trazas distribuidas**, que explican el *dónde*. Herramientas como Jaeger o Grafana Tempo reconstruyen el recorrido completo de una petición a través de todos los servicios implicados. Gracias a ellas es posible observar, por ejemplo, que una solicitud de usuario pasó por un API Gateway, invocó varios microservicios y terminó esperando varios segundos en una consulta a la base de datos. Esta visibilidad resulta esencial para entender cuellos de botella en sistemas altamente distribuidos.

Lo interesante es que, aunque hoy se consideran conceptos fundamentales en ingeniería de sistemas, **hace una década apenas formaban parte del discurso habitual**. Los registros existían desde los primeros sistemas operativos, pero rara vez se trataban como una fuente estructurada de información para comprender sistemas complejos. La explosión de arquitecturas distribuidas obligó a replantear la forma de diagnosticar problemas y dio lugar a una disciplina específica —la observabilidad— que hoy se ha convertido en una práctica central en el diseño y operación de infraestructuras modernas.