

PULPØ
CØN 23



Observabilidad,
Open Telemetry
¿Whaaaat?





Hello! I'm *Maria*

Soy Senior SRE en Flywire, donde estoy desde hace un par de años haciendo cosas raras e intentando que desarrolladores no nos maten mucho.

Llevo en este mundillo algo más de 5 años 😊 gracias a empezar la carrera de Telecomunicaciones y ver que eso no era para mi



Qué vamos a ver hoy

01 Definición

¿Qué es observabilidad?

04 Otros conceptos

SLO/SLI, feature flags,...

Monitorización vs 011y

Diferencias entre ambas y utilidades

05 Open Telemetry

Cómo se puede implementar esto de la observabilidad

03 Pilares

En qué se basa y por qué

Tentáculos a la obra

Debate, escenarios, ...



01

Definición

Habilidad para poder responder cualquier pregunta de nuestros sistemas en cualquier momento sin necesidad de acceder a ellos de ninguna forma.

En profundidad...

- Cuando tengamos errores desconocidos (unkonwn unkowns) seremos capaces de encontrar el por qué.
- Podremos ver fácilmente si hay degradación en cualquiera de nuestros procesos.
- Podremos alertar de forma automática de cualquier anomalía y encontrar la causa sin morir (mucho) en el intento

CONOCIMIENTO	DATOS DISPONIBLES	CONOCIMIENTO	DATOS DISPONIBLES
CONOCIDOS DESCONOCIDOS Datos que entendemos pero no sabemos su utilidad	Al llegar al 85 % de CPU ¿que pasa?	CONOCIDOS DESCONOCIDOS Datos que tenemos y entendemos	CPU al 85 %
DESCONOCIDOS CONOCIDOS Datos que ni tenemos ni conocemos	El servidor no responde	DESCONOCIDOS CONOCIDOS Tenemos los datos pero no se entienden	CPU al 95 % pero no sabemos la causa



02

Monitorización vs observabilidad

Ya sabemos en qué nos ayuda pero, ¿en qué se diferencia de la monitorización?

Monitorización Observabilidad

- Se basa en datos predefinidos - forma reactiva.
- Se buscan respuestas en base a paneles.
- Útil en entornos estáticos y monolitos.
- Responde a: ¿Funciona mi sistema?
- Trabaja con datos de disponibilidad, capacidad y rendimiento.
- Nos ayudan en caso de errores conocidos.

- Se basa en patrones y propiedades - forma proactiva.
- Se buscan respuestas en base a hipótesis.
- Útil en entornos dinámicos y distribuidos.
- Responde a ¿Cómo está mi sistema?
- Trabaja con logs, trazas y métricas que incluyen los datos de monitorización.
- Proporciona información para trabajar..... con errores desconocidos.

Podríamos decir que la monitorización es un componente de la observabilidad.



03

Pilares

Conceptos básicos para entender cómo funcionan nuestros sistemas

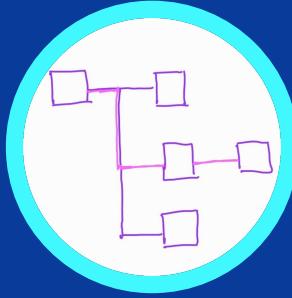
Los 3 pilares



Métricas

P: ¿Tengo un problema?

Datos numéricos que cuantifican aspectos del rendimiento del sistema como tiempo de respuesta, ratios de error, caudal y uso de recursos. Suelen ser recogidas a intervalos y nos dan una visión cuantitativa del sistema



Trazas

P: ¿Dónde está?

Nos proporcionan una imagen detallada del flujo de peticiones dentro de nuestras aplicaciones y sus componentes. Son útiles para identificar cuellos de botella, problemas de latencia y para poder comprender mejor cómo se interconectan las distintas partes del sistema (microservicios)



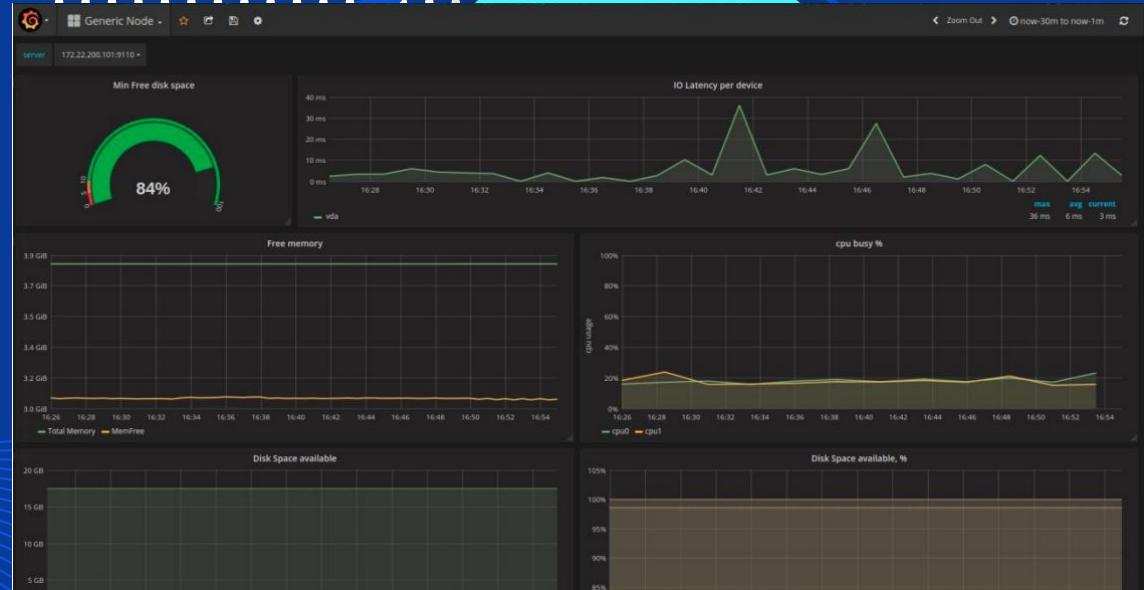
Logs

P: ¿Qué lo está causando?

Registros detallados de eventos en el sistema. Nos proporcionan información sobre interacciones de usuarios, mensajes de error, eventos del sistema y otros datos relevantes. Son necesarios para desarrollar, resolver incidentes, audits...

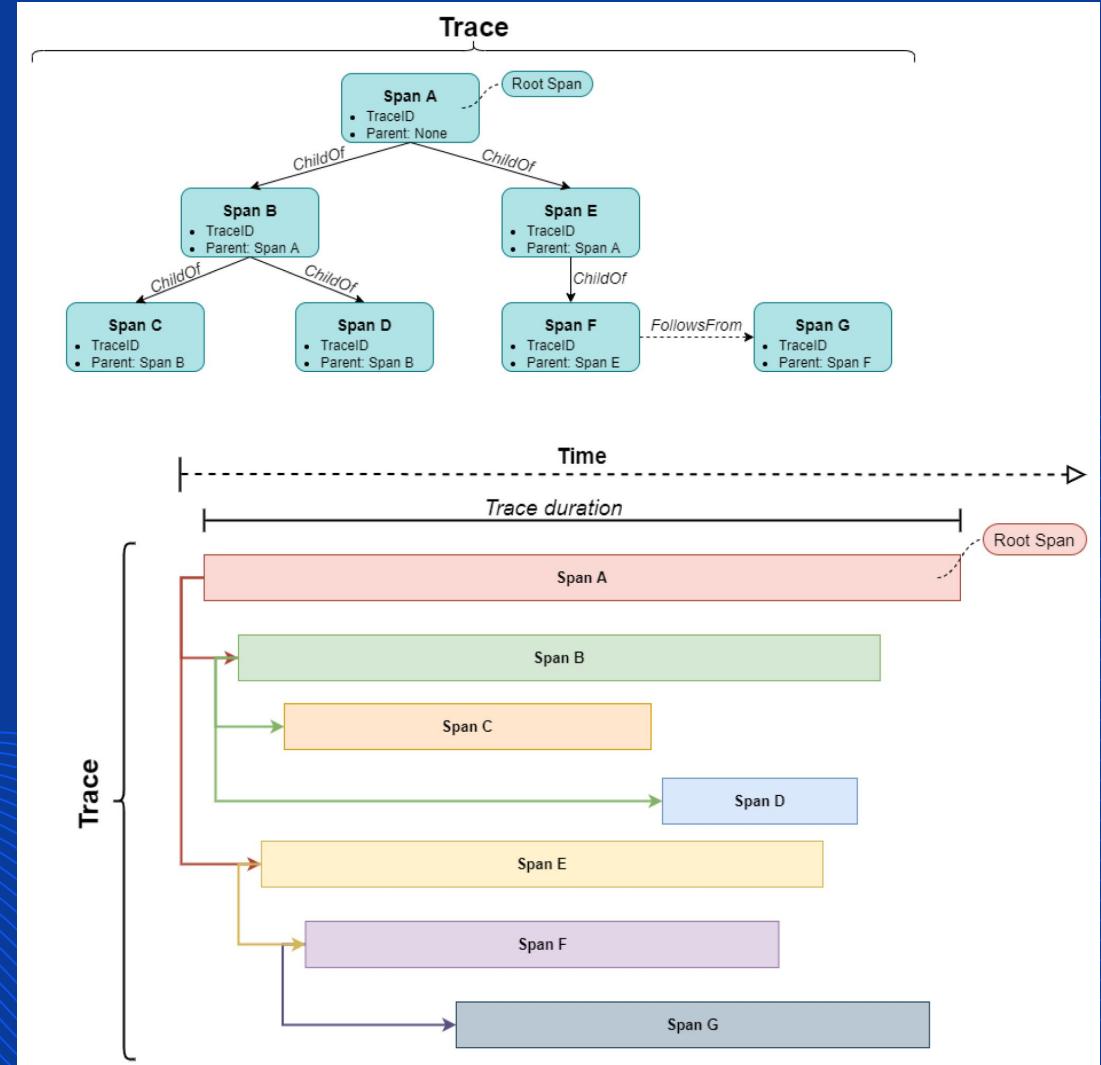
Métricas

- Nos van a proporcionar información de una forma más visual en la que detectar patrones, ver actividad inusual y ayudarnos a navegar nuestros errores.
- Útiles como primera aproximación en caso de error.
- Hasta ahora han sido nuestras aliadas en caso de fallo.



Trazas - distributed tracing

- Está compuesta por spans que tienen asociado un traceID para poder ser identificados.
- Span: operaciones que componen la traza que siguen un orden.
- TraceID: identificador de una traza que será siempre igual en todos los spans que la compongan. Muy útil para encontrar el inicio de traza de la que viene un error por ejemplo.



Logs

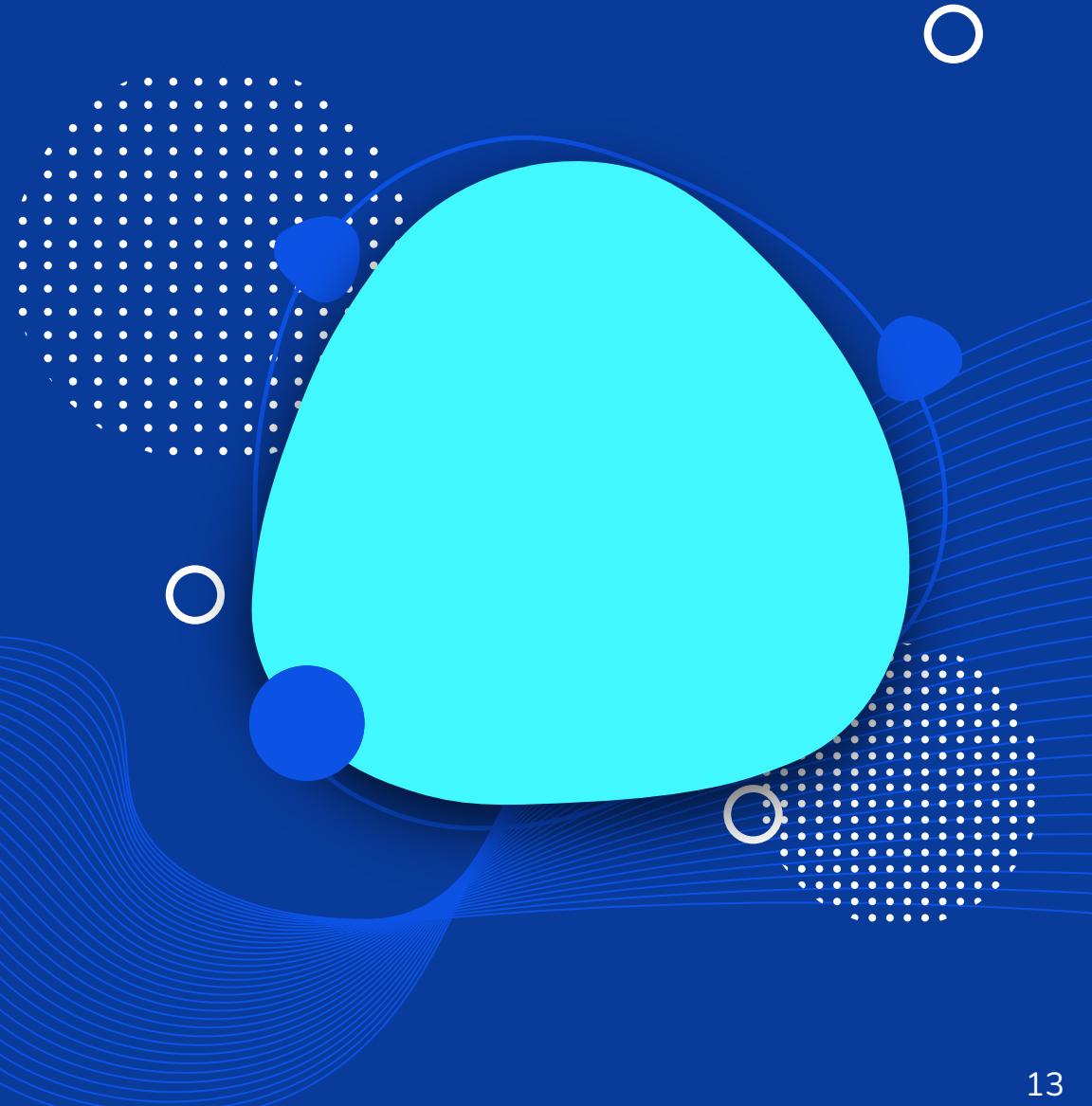
- Log aggregation: proceso de recolectar, estandarizar y consolidar los logs en nuestras aplicaciones para poder mandarlos a nuestro sistema de monitorización/observabilidad.
- Para que los logs sean útiles, es necesario que sigan cierto estándar.
- nos facilitará el poder buscar, filtrar y agrupar en caso de que necesitemos algún tipo de información.

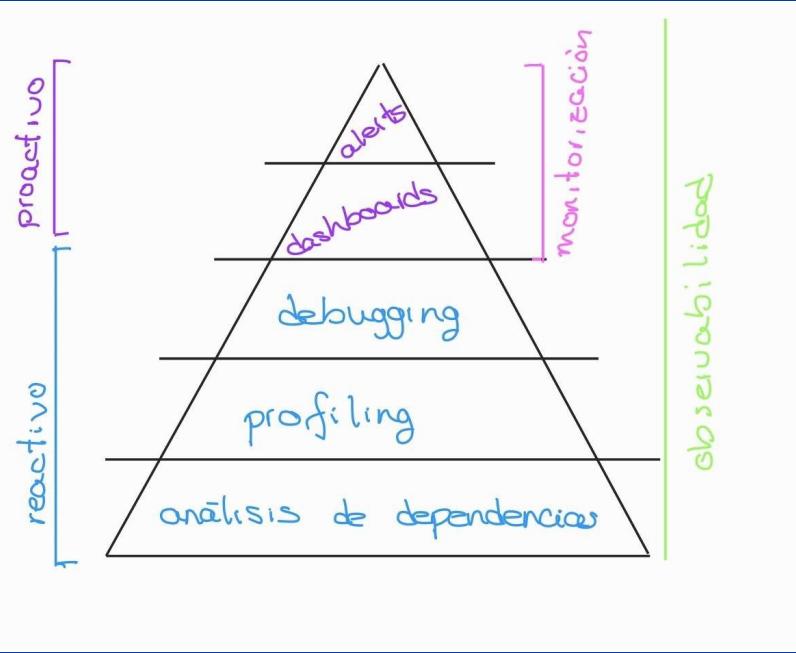
To sample or not to sample

Llegado cierto volumen de logs, podemos decidir si nos puede resultar útil coger muestras en vez de todo el volumen. No todas nuestras aplicaciones van a poder aplicar esto ya que habrá algunas de las que necesitemos el 100% de los datos por su naturaleza (ejemplo: la que controla los estados de los pagos), sin embargo, habrá otras que nos podemos permitir perder ciertos logs sabiendo que, si hubiera algún problema, vamos a poder encontrar información relevante en los que si hemos guardado y mandado a nuestra herramienta de logging.

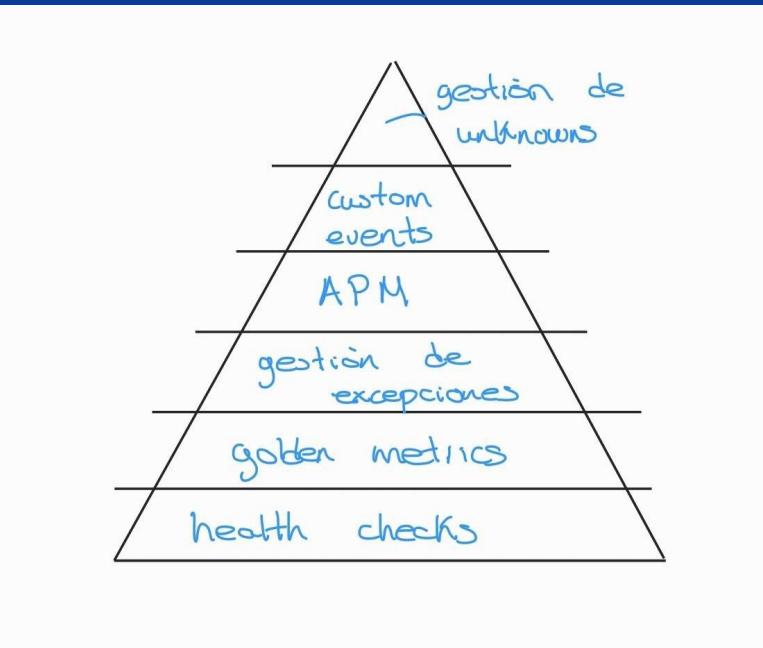
Pirámide de observabilidad

- Al estilo de la pirámide de Maslow, podemos crear una pirámide de observabilidad.
- Con ella se puede medir la madurez de un sistema en cuanto a observabilidad y ver qué necesidades debemos cubrir.
- Según la posición en la que esté la aplicación se decide cuánto hay que invertir en mejorar la situación, si fuera necesario y permite priorizar el trabajo.
- Cada paso será más complejo que el anterior, hasta llegar a la 'perfección'.





- APM: (application performance monitoring) Software que nos permite trazar y diagnosticar problemas a nivel de aplicación, ver la experiencia de usuario, service maps, ...
- Golden metrics: métricas relevantes para un primer vistazo al estado de la aplicación. Incluye latencia, tráfico, errores, saturación.
- Profiling: observar métricas (memoria, CPU, ...) del servidor en el que está corriendo la aplicación.





04

Otros conceptos

Otros conceptos relacionados y que conviene tener en cuenta



01

Alerta

Notificaciones automáticas que se envían cuando se cruza cierto límite impuesto. Suele estar basada en conocimiento previo.



02

Dashboard

Representaciones visuales de la información que nos proporcionan los pilares, alarmas, etc. Nos dan una forma sencilla de ver de forma rápida el estado actual del sistema.



03

Service level

Cómo rinde un servicio con respecto a las expectativas del usuario. Es un concepto un tanto etéreo ya que depende de qué haga el servicio y lo que entienda el usuario

Ej: El usuario espera que sea rápido y fiable.



04

SLI

Service level indicator.

Medida cuantitativa de las expectativas del usuario del rendimiento de un servicio (service level).

Número de peticiones OK dividido por el número total de peticiones válidas

05

SLO

Service level objective.

Objetivo esperado de un servicio con respecto del SLI.

95% de peticiones tienen que ser respondidas en < 400 ms

06

SLA

Service level agreement.

Qué ocurre si no se cumple el SLO. Suele ser un acuerdo legal que se ignora.

Necesitamos disponibilidad mensual del 99.99% (4 nubes)



07

Error budget

Define el máximo tiempo de downtime o con errores aceptable. Está muy relacionado con el SLO. Nos dan una guía para poder hacer mejoras/mantenimientos en las aplicaciones.



08

Feature flag

Técnica que permite activar/desactivar funcionalidades en el código sin necesidad de modificarlo. Puede ser una forma relativamente segura de ver el comportamiento de un cambio directamente en producción con un rollback rápido.



05

Open Telemetry

Segunda caja de pandora de la mañana

¿Qué es Open Telemetry?

- Herramienta para manejar los datos de telemetría, independiente del backend de visualización, con un protocolo estándar para dichos datos.
-

Definición oficial:

OpenTelemetry is an Observability framework and toolkit designed to create and manage **telemetry data** such as traces, metrics, and logs. Crucially, OpenTelemetry is vendor- and tool-agnostic, meaning that it can be used with a broad variety of Observability backends, including open source tools like Jaeger and Prometheus, as well as commercial offerings. OpenTelemetry is a CNCF project.

Conceptos a tener en cuenta



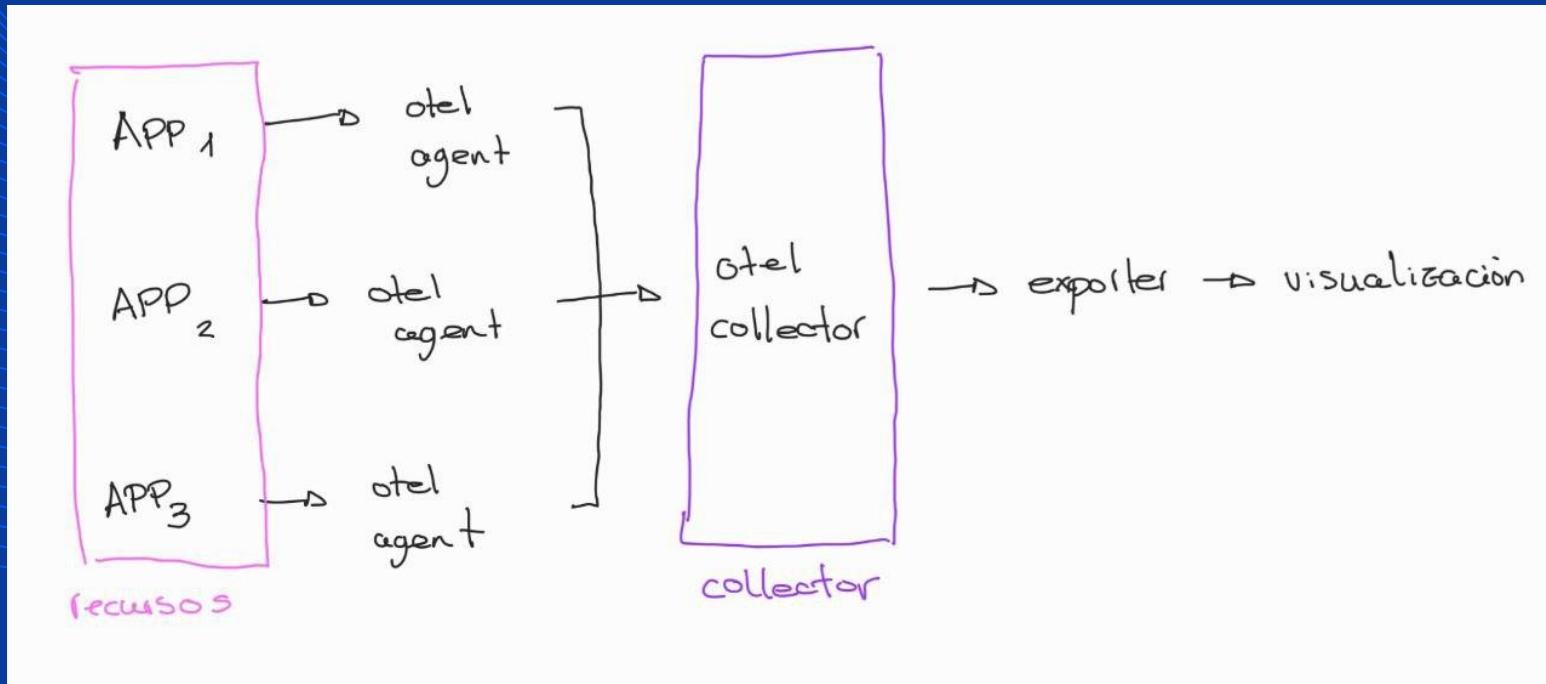
Elemento que recibe, procesa y exporta los datos.



Servicio que ha generado la señal de telemetría.



Cuando un servicio llama a otro se manda en la petición metadatos de forma que se asocia a la traza de origen. Podríamos traducirlo como herencia de contexto o traceld.



Jaeger

- Software para supervisar y solucionar problemas en un sistema distribuido realizado por Uber y open source.
- Útil para seguir una traza y aislar cualquier tipo de problema.
- Es en sí un sistema distribuido con diferentes componentes: cliente, agente, collector, consola, ...
- Se recomienda que los datos que recibe estén instrumentados con Open Telemetry.





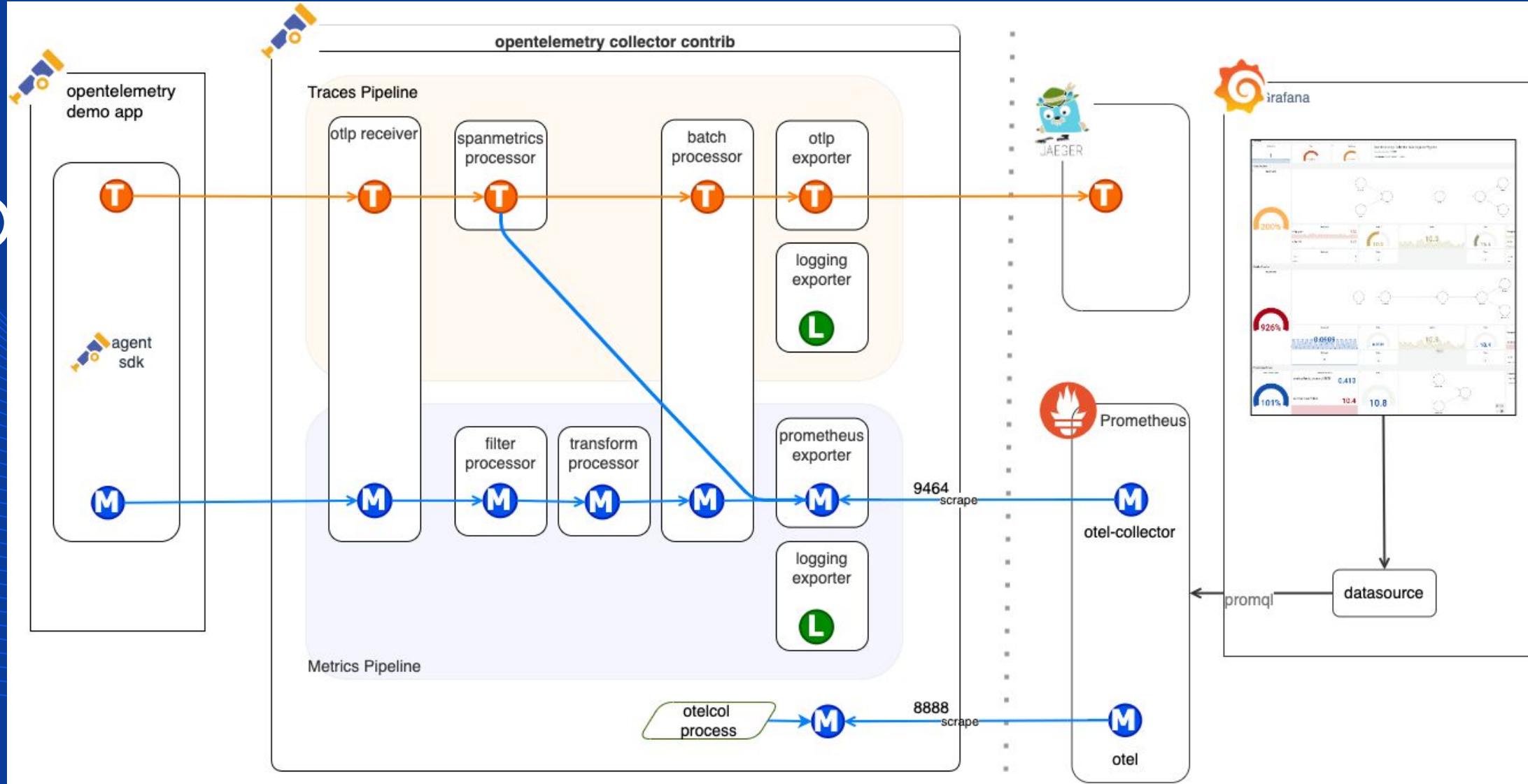
06

!Tentáculos a la obra!

Ya he hablado mucho, es hora de ponerse manos a la obra

- Web store: <http://localhost:8080/>
- Grafana: <http://localhost:8080/grafana/>
- Feature Flags UI: <http://localhost:8080/feature/>
- Load Generator UI:
<http://localhost:8080/loadgen/>
- Jaeger UI: <http://localhost:8080/jaeger/ui/>

Flujo de datos



Cosas interesantes

1. ¿Dónde podemos ver las métricas?
2. ¿Dónde podemos ver las trazas?
3. ¿Dónde podemos ver los logs?
4. ¿Dónde podemos ver la arquitectura del sistema?
5. ¿Qué feature flags tenemos?

Escenario 1 - vamos a situarnos

Coged una traza, la primera que veais y seguid hasta llegar a la raíz.

- ¿Hay algo raro en esa traza?
- ¿Cómo son los tiempos?
- ¿Cumpliría la condición de que las peticiones tienen que tardar menos de 40ms?
- Con un SLO en el que el 95% de las peticiones tienen que tardar menos de 40ms, ¿nuestra aplicación lo cumpliría?

Escenario 2 - peticiones con errores

- Para este escenario vamos a necesitar activar la feature flag productCatalogFailure
- Esto va a generar peticiones con errores para un cierto productId, ¿cuál es?

Solucion:

Escenario 2 - peticiones con errores

- Para este escenario vamos a necesitar activar la feature flag productCatalogFailure
- Esto va a generar peticiones con errores para un cierto productId, ¿cuál es?

Solución: OLJCESPC7Z

Escenario 3 - memory leak

Para este escenario, necesitamos dejar corriendo la aplicación unos 10 minutos para tener datos y luego activar la FF **recommendationCache** y dejar la aplicación corriendo unos 10 minutos.

Escenario 3 - memory leak

Para este escenario, necesitamos dejar corriendo la aplicación unos 10 minutos para tener datos y luego activar la FF **recommendationCache** y dejar la aplicación corriendo unos 10 minutos.

Siguiendo la slide de los pilares:

1. ¿Tengo un problema? - Metricas
 - a. En el dashboard de los servicios se ve como hay picos en CPU, latencia y ratio de errores.
Con estos datos ya sabemos que algo pasa en el recommendation service.
2. ¿Dónde está el problema? - Trazas
 - a. Sabemos que la latencia ha aumentado por lo que puede que mirando las trazas del recommendation service veamos que hay algunas con tiempos de carga fuera de lo normal.
 - b. Si cogemos alguna de las trazas lentas podemos ver algunos atributos útiles (app.cache_hit = true)
 - c. Con esto volvemos al buscador y comparamos las trazas con app.cache_hit a false y a true. ¿Qué ocurre?

Escenario 4 - ¿Dónde está la configuración?

A priori, esta demo está pensada para poder jugar y ver conceptos teóricos en práctica pero, ¿y si quiero usar mi propio backend?

Escenario 4 - ¿Dónde está la configuración?

A priori, esta demo está pensada para poder jugar y ver conceptos teóricos en práctica pero, ¿y si quiero usar mi propio backend?

Tenemos 2 ficheros de configuración

- src/otelcollector/otelcol-config.yml
- src/otelcollector/otelcol-config-extras.yml

En este segundo fichero podríamos añadir nuestro exporter y el pipeline (activación del exporter añadido antes) con el nuevo exporter. En este caso nuestro backed soporta OTLP sobre HTTP

```
1 exporters:  
2   otlphttp/example:  
3     endpoint: <your-endpoint-url>  
4  
5 service:  
6   pipelines:  
7     traces:  
8       receivers: [otlp]  
9       processors: [batch]  
10      exporters: [otlphttp/example]
```

Ring - monitorización vs observabilidad

Por suerte, hoy nos toca turno de oncall y nos ha llegado una alerta mientras peleábamos con un mueble de IKEA. Nos dice que en nuestra tienda online han dejado de mostrarse algunos productos.

- ¿Cómo lo haríamos con herramientas de monitorización?
- ¿Y si tuviéramos implementada alguna solución de o11y como la del ejemplo?

Discusión

- ¿Alguien tiene pactados SLO/SLI? ¿Son realistas?
- ¿Quien tomó las decisiones? ¿Había perfiles técnicos?
- ¿En qué nivel estáis y en cual os gustaría estar?
- ¿Como se podria dar valor y fuerza a una iniciativa de observabilidad?
- De las preguntas clásicas de observabilidad, ¿cuáles aplicáis ya?
 - ¿Puedo saber el estado del servicio sin entrar en el servidor o ejecutar algo de forma remota?
 - ¿Cómo veo los logs?
 - ¿Tengo algún sistema de alertas o los avisos son con intervención humana?
 - ¿El equipo de oncall es capaz de resolver un incidente sin sensación de estar en un escape room?



Thank you!

Credits.

Presentation Template: [SlidesMania](#)

Sample Images: [Unsplash](#)

Fonts used in this presentation: Nunito, **Barlow Semi Condensed Bold** and *Pacifico*.