SESIÓN 1 DE 2

Observabilidad Inteligente con Dynatrace

dynatrace

De los pilares fundamentales a la infraestructura, los servicios y los logs: cómo ver, entender y consultar todo lo que pasa en tu sistema.


Pilares


Infraestructura


Servicios


Logs + DQL

AGENDA DE LA SESIÓN

Cuatro bloques, de lo básico a Dynatrace

1. Pilares de la observabilidad

Qué es observar un sistema y las 3 señales que lo hacen posible.

2. Infraestructura: VMs y Kubernetes

De un servidor físico a un clúster dinámico de contenedores.

3. Servicios y trazas

Cómo Dynatrace descubre servicios y sigue una petición punta a punta.

4. Logs y DQL

El registro fino del sistema y el lenguaje para preguntarle cualquier cosa.

BLOQUE 1 · PILARES DE LA OBSERVABILIDAD

Monitorear no es lo mismo que observar



Monitorear

Vigilar métricas y umbrales que ya conoces de antemano: "avísame si el CPU pasa de 80%".



Observar

Poder hacer preguntas nuevas sobre un sistema sin haber previsto esa pregunta de antemano, usando los datos que ya genera.

En sistemas modernos —cientos de microservicios, contenedores que nacen y mueren— es imposible anticipar cada pregunta. Por eso se necesita observabilidad, no solo monitoreo.

BLOQUE 1 · PILARES DE LA OBSERVABILIDAD

Las 3 señales que hacen posible observar



Logs

Eventos discretos con todo el detalle: qué pasó exactamente y cuándo.

Ejemplo "Error 500 al conectar con la base de datos a las 14:32:07"



Métricas

Números agregados en el tiempo que describen la salud general.

Ejemplo CPU al 87% en el host web-03 durante los últimos 5 minutos



Trazas

El recorrido completo de una solicitud por varios servicios.

Ejemplo Un checkout pasa por 6 microservicios en 1.8s

BLOQUE 1 · PILARES DE LA OBSERVABILIDAD

En Dynatrace, los 3 pilares viven juntos

Grail es el lakehouse de datos de Dynatrace: logs, métricas, trazas y eventos de seguridad se guardan con el mismo contexto, listos para consultarse juntos con un solo lenguaje: DQL.



BLOQUE 2 · INFRAESTRUCTURA: VMS Y KUBERNETES

Todo empieza con un servidor

Antes de hablar de contenedores hay que entender los dos niveles de infraestructura sobre los que corre casi todo el software.



Servidor físico

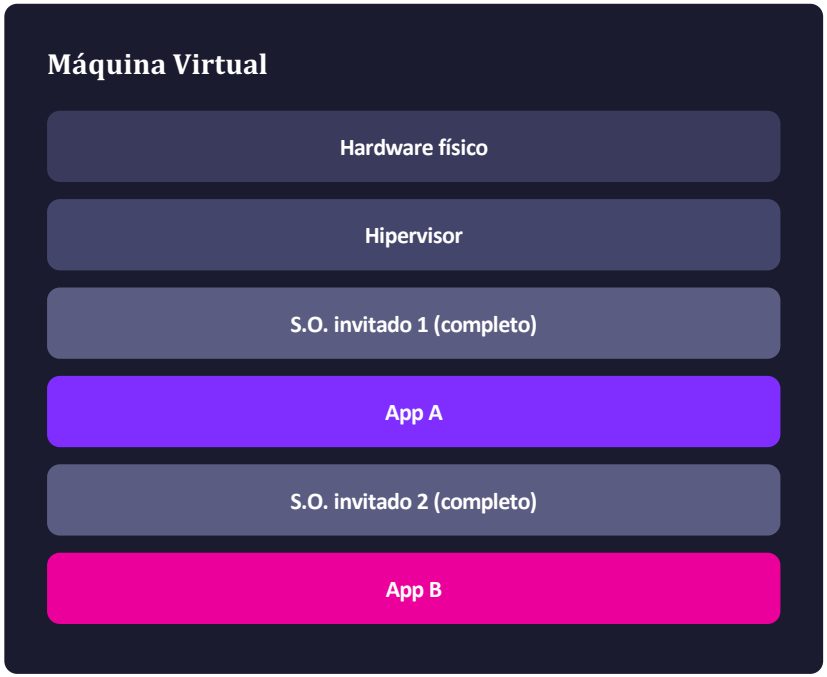
Una máquina real, con su propio CPU, memoria y disco. Cara, difícil de escalar rápido.



Máquina Virtual (VM)

Software que simula un servidor completo dentro de otro servidor físico, con su propio sistema operativo. Permite correr varias "máquinas" en un mismo hardware.

Contenedores: más ligeros que una VM





BLOQUE 2 · INFRAESTRUCTURA: VMS Y KUBERNETES

Kubernetes: quién ordena a los contenedores

Con cientos de contenedores en producción, alguien tiene que decidir dónde corre cada uno, reiniciarlos si fallan y escalarlos según demanda.

Ese "alguien" es Kubernetes.

Auto-reparación

Reemplaza contenedores caídos sin intervención humana.

Auto-escalado

Agrega o quita réplicas según CPU, memoria o tráfico.

Scheduling

Decide en qué nodo colocar cada contenedor.

Rolling updates

Despliega nuevas versiones de forma gradual, sin cortes.

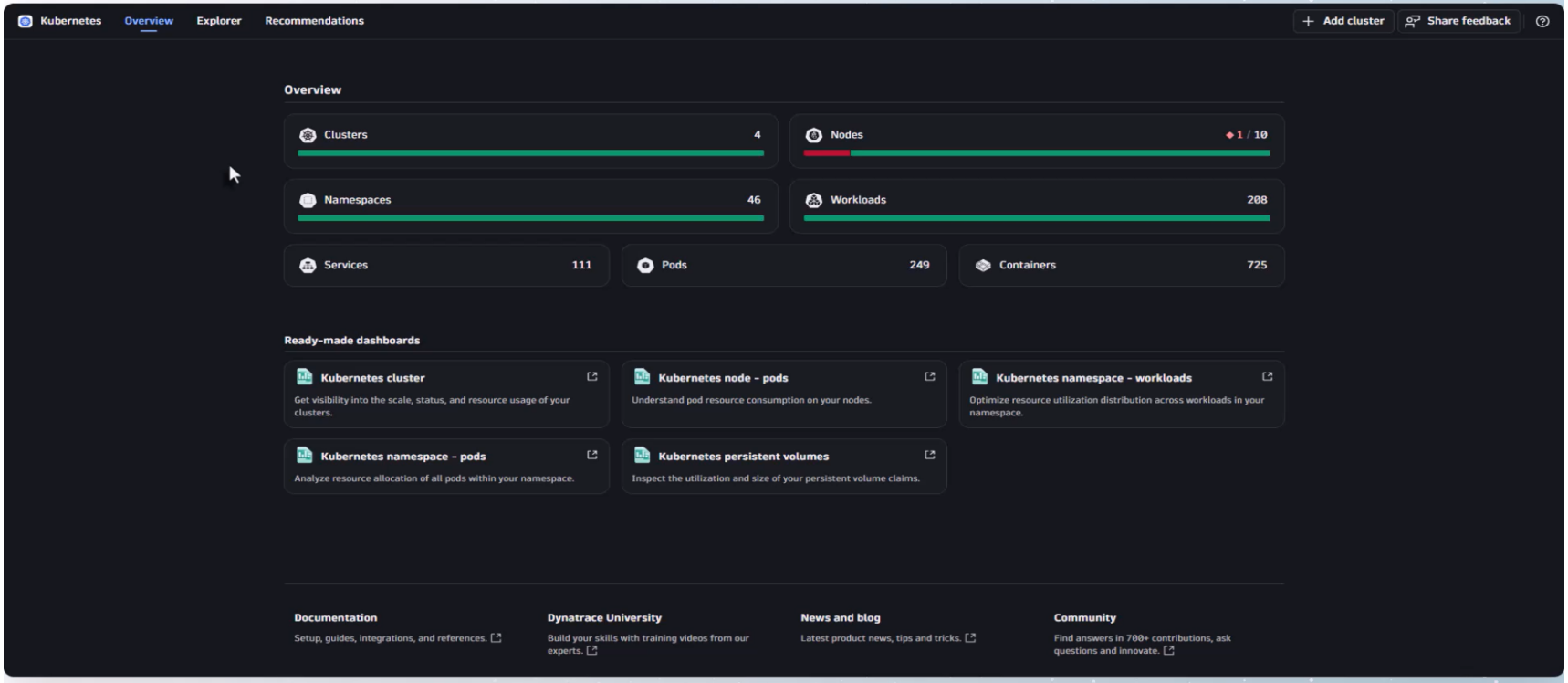
BLOQUE 2 · INFRAESTRUCTURA: VMS Y KUBERNETES

Tanta automatización... es difícil de vigilar a mano

Los Pods nacen y mueren, las IPs cambian, el clúster puede tener miles de contenedores en decenas de nodos y VMs. **¿Cómo sabes si un nodo se está quedando sin memoria ahora mismo?**



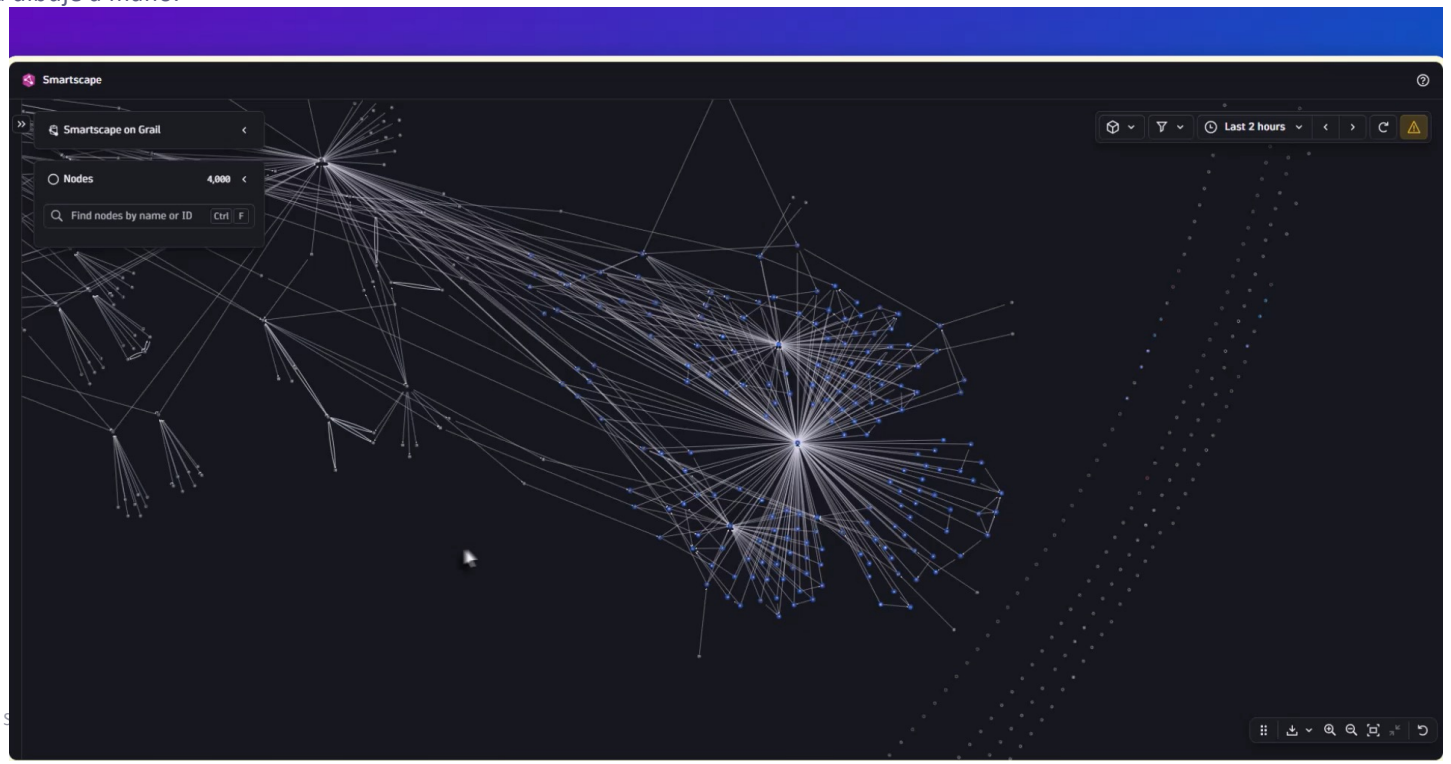
La app de Kubernetes: salud del clúster de un vistazo



BLOQUE 2 · INFRAESTRUCTURA: VMS Y KUBERNETES

Smartscape: el mapa vivo de tu infraestructura

Smartscape es la topología automática de Dynatrace: descubre y conecta en tiempo real hosts, VMs, contenedores, procesos y servicios, sin que nadie la dibuje a mano.

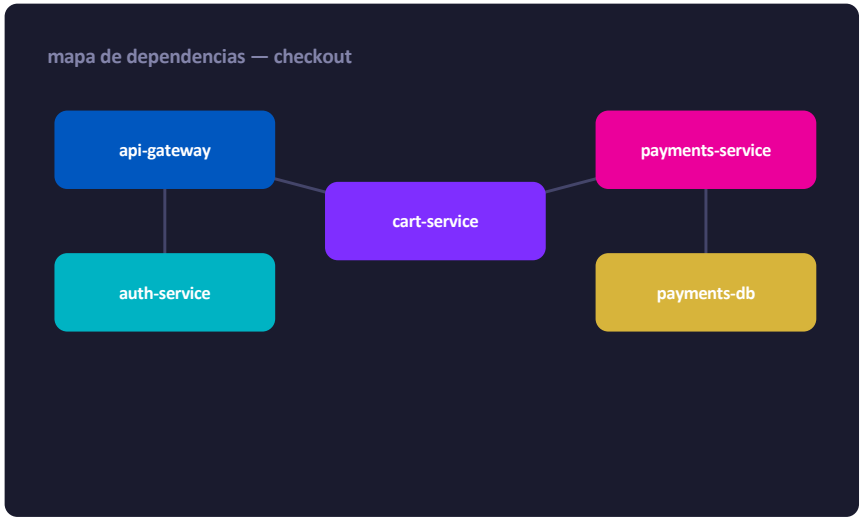


BLOQUE 3 · SERVICIOS Y TRAZAS

¿Qué es un "servicio" para Dynatrace?

Un servicio es cualquier proceso que atiende solicitudes: una API, un microservicio, una función. Dynatrace los detecta automáticamente al instalar OneAgent — sin configurar nada a mano.

- Se agrupan automáticamente por tecnología: Java, .NET, Node.js, Go…
- Cada servicio recibe una entidad propia con su historial de rendimiento.
- Se conectan entre sí solo, formando el mapa de dependencias real.



BLOQUE 3 · SERVICIOS Y TRAZAS

Las 4 métricas que definen la salud de un servicio

Tiempo de respuesta

Cuánto tarda el servicio en responder (p50, p90, p99).

Tasa de fallos

Porcentaje de solicitudes que terminan en error.

Throughput

Cuántas solicitudes procesa por minuto/segundo.

Saturación

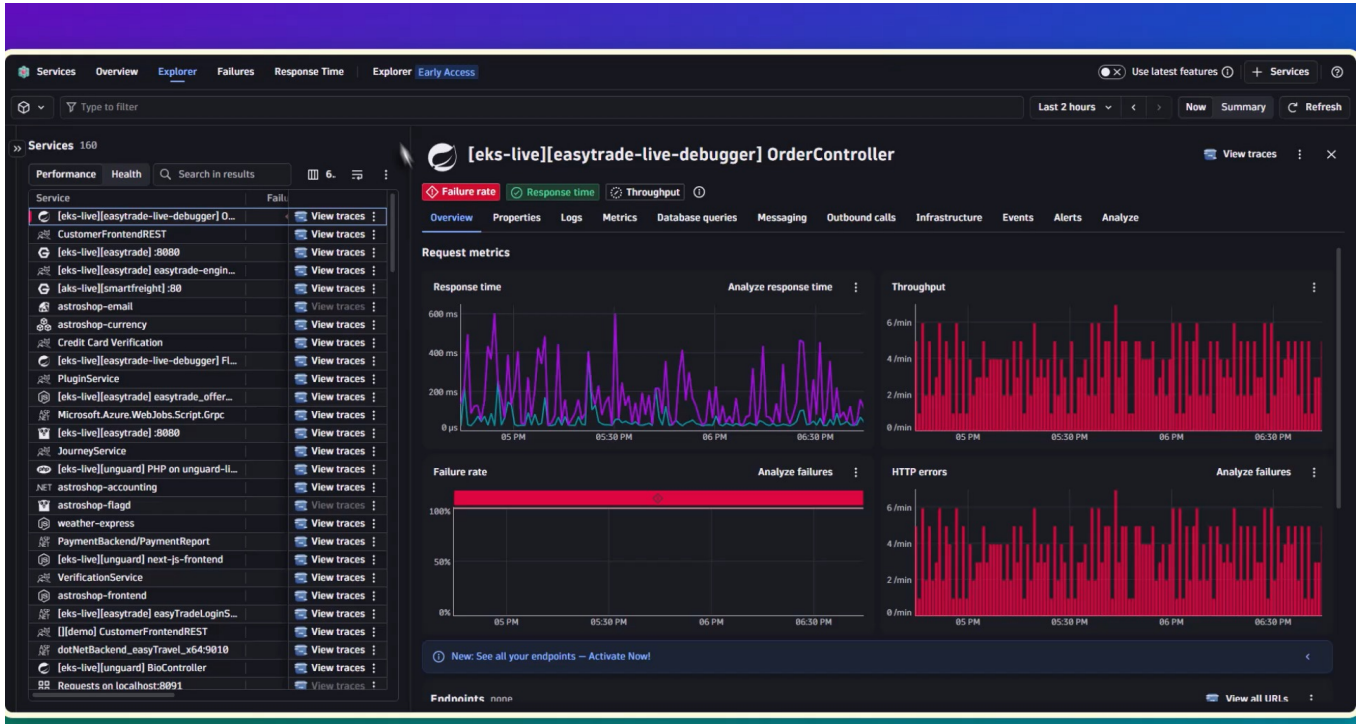
Qué tan cerca está de su capacidad máxima (CPU, memoria, hilos).

Dynatrace calcula estas 4 señales automáticamente para cada servicio detectado — son la base del Service flow y de cualquier alerta de rendimiento.

BLOQUE 3 · SERVICIOS Y TRAZAS

PurePath: la traza que sigue cada petición

PurePath es la tecnología de trazado de Dynatrace: captura cada solicitud individual de punta a punta, con precisión de código, sin muestreo — cada transacción, siempre.



BLOQUE 3 · SERVICIOS Y TRAZAS

Caso práctico: ¿quién ralentizó el checkout?



1. Alerta

El tiempo de respuesta de "checkout" sube de 400ms a 2.1s.



2. Service flow

Se abre el flujo y se ve que payments-service concentra el salto de latencia.



3. PurePath

Se inspecciona una traza puntual: la llamada a payments-db tarda 1.6s.



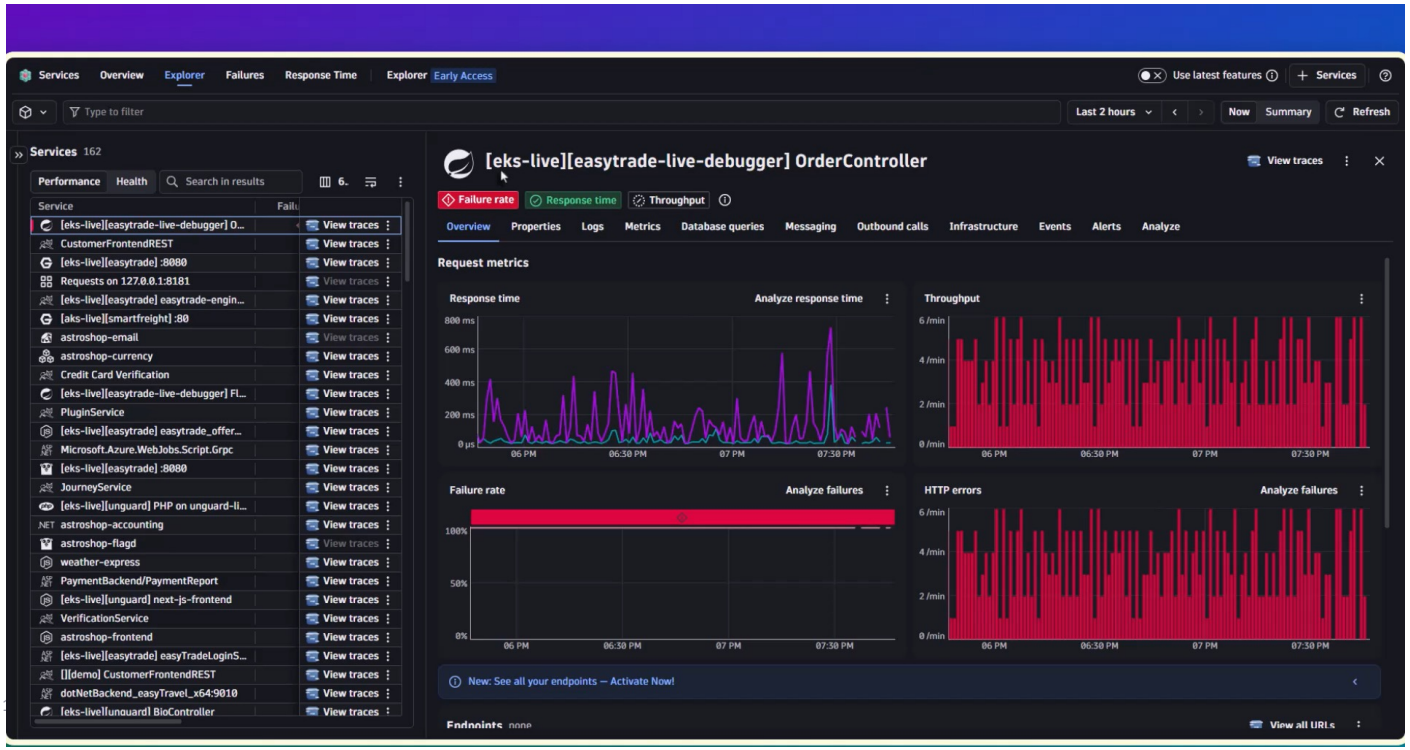
4. Causa raíz

Una consulta SQL sin índice — identificada sin adivinar, en minutos.

BLOQUE 4 · LOGS Y DQL

Logs: el detalle que las métricas no dan

Cuando una métrica te dice que algo está mal, el log te dice exactamente **qué** pasó.



BLOQUE 4 · LOGS Y DQL

DQL: una sola forma de preguntar por todo

DQL (Dynatrace Query Language) consulta logs, métricas, trazas y eventos de Grail encadenando comandos con el operador **pipe (|)**, igual que en la terminal de Linux.

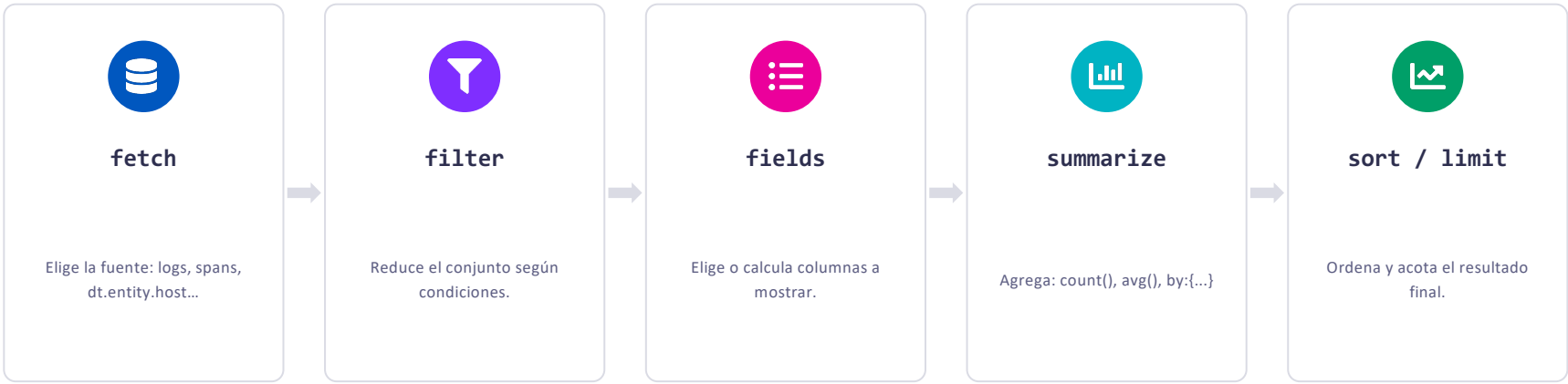
- No necesitas saber en qué tabla vive el dato.
- Funciona igual sobre logs, métricas, trazas o infraestructura.
- Se usa en Notebooks, Dashboards y Workflows.

dql · app de logs

```
fetch logs
| filter k8s.namespace.name ==
      "payments"
| filter loglevel == "ERROR"
| sort timestamp desc
| limit 50
```

BLOQUE 4 · LOGS Y DQL

La anatomía de una query DQL




```
fetch logs | filter k8s.namespace.name == "payments" | summarize errores = count(), by:{k8s.pod.name} | sort errores desc
```

BLOQUE 4 · LOGS Y DQL

Ejemplo: correlacionar un nodo con sus errores

ejemplo aplicado

```
fetch logs, from: -1h
| filter k8s.node.name ==
      "aks-prod-node-07"
| filter loglevel == "ERROR"
| summarize errores = count(),
      by:{k8s.pod.name}
| sort errores desc
```

De la infraestructura al detalle

Detectaste en el dashboard de Kubernetes que el nodo aks-prod-node-07 tiene memoria alta. Con esta query filtras solo los logs de error de los Pods que corren en ese nodo, agrupados para ver cuál concentra más fallos.

Así se cierra el círculo entre los 4 bloques de hoy

Infraestructura (dónde) → Servicio (qué proceso) → Traza (qué tan lento) → Log (por qué exactamente) — con DQL como el hilo que conecta las cuatro.

CIERRE DE LA SESIÓN 1

Lo que se llevan hoy

Observar > monitorear

Logs, métricas y trazas responden preguntas que no anticipaste.

K8s automatiza infraestructura

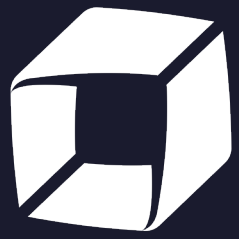
Pero la vuelve dinámica — y difícil de vigilar sin herramientas.

Los servicios se descubren solos

PurePath sigue cada transacción punta a punta, sin muestreo.

DQL conecta todo

Un lenguaje simple para preguntar sobre infraestructura, servicios y logs.



dynatrace

¡Gracias!

En la Sesión 2 veremos Application Security, User Experience, Business Observability y Dynatrace Intelligence.



