Dutch Cloud Native & AI Community - December 2025

Breaking Free with Open Standards:

OpenTelemetry and Perses for Observability

Kasper Borg Nissen, Principal Developer Advocate at dash0

@phennex kaspernissen.xyz

dash0

Who?

Principal Developer Advocate at Dash0

KubeCon+CloudNativeCon EU/NA 24/25 Co-Chair (former)

CNCF Ambassador

Golden Kubestronaut

CNCG Aarhus, KCD Denmark Organizer

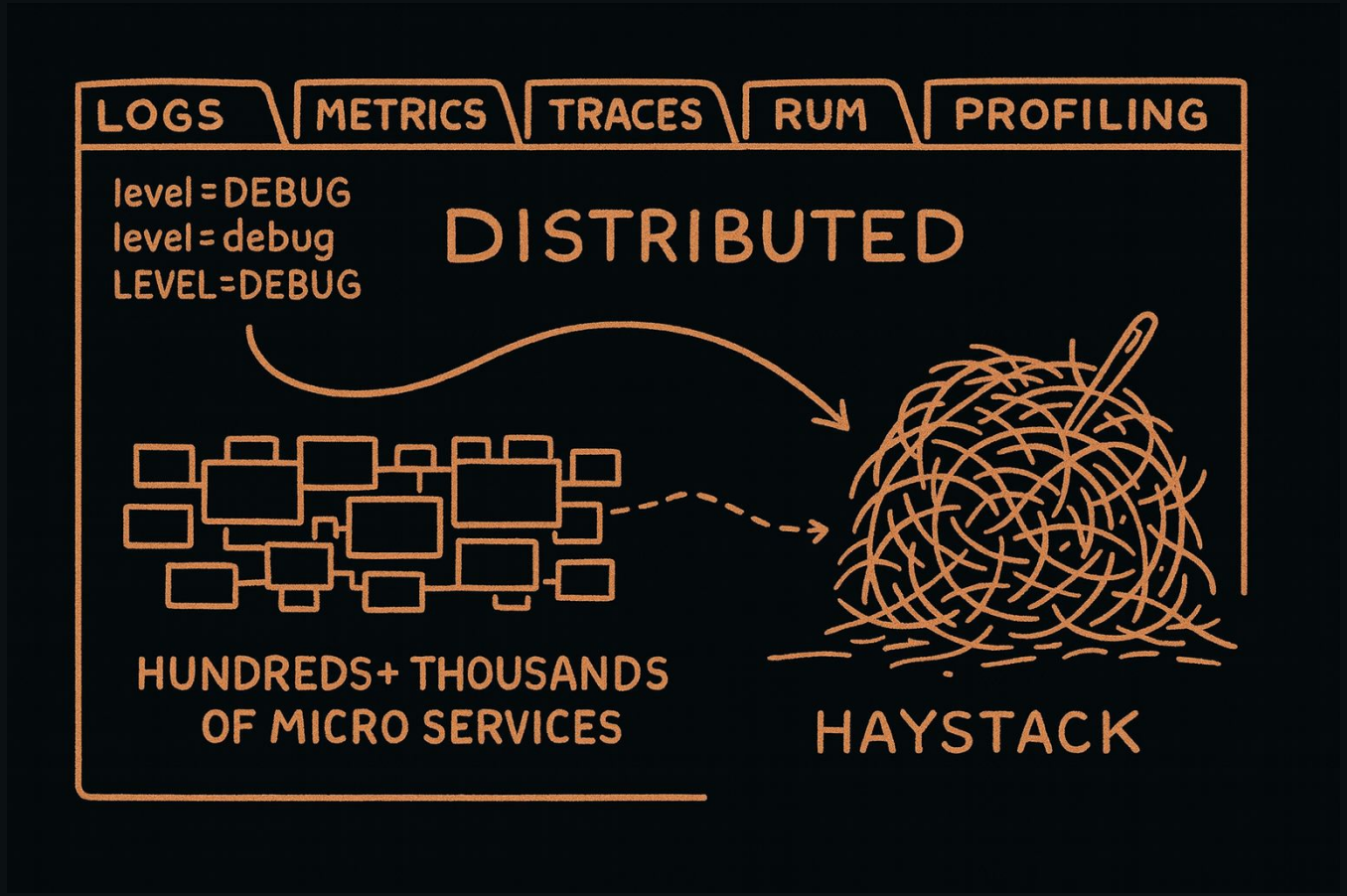
Co-founder & Community Lead Cloud Native Nordics

dash0

tl;dr

- OpenTelemetry is standardizing telemetry collection.
- Perses is standardizing dashboarding.
- Applying Platform Engineering principles transforms observability into a seamless, scalable, and developer-friendly experience.
- Building on Open Standards allows you to freely move between vendors, ensuring they stay on their toes and provide you the best possible experience.

Today...


LOGS
METRICS
TRACES
RUM
PROFILING
level=DEBUG
level=debug
LEVEL=DEBUG
DISTRIBUTED
HUNDREDS+ THOUSANDS
OF MICRO SERVICES
HAYSTACK

This fragmentation, leads to

Image by pngtree.com

This fragmentation, leads to

Complex Query Languages

Image by pngtree.com

This fragmentation, leads to

Complex Query Languages

Vendor lock-in

This fragmentation, leads to

Complex Query Languages

Vendor lock-in

Metadata Inconsistency

This fragmentation, leads to

Complex Query Languages

Vendor lock-in

Metadata Inconsistency

No instrumentation due to high complexity

This fragmentation, leads to

Complex Query Languages

Vendor lock-in

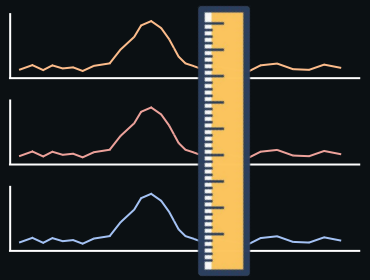
Metadata Inconsistency

No instrumentation due to high complexity

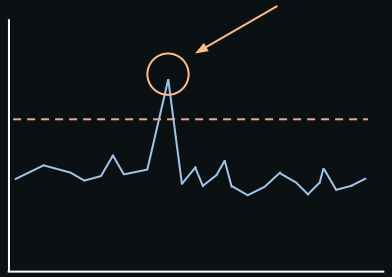
Lack of unified insights

A shift is happening.

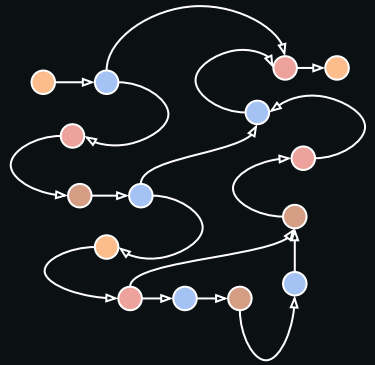
A shift toward correlation



Find related information



Jump between signals



Reconstruct chain of events

A shift toward...

OpenTelemetry

OpenTelemetry (OTel) is an open source project designed to provide standardized tools and APIs for generating, collecting, and exporting telemetry data such as traces, metrics, and logs

The de-facto standard for distributed tracing, also supports metrics and logs (soon profiling)

The main goals of the project are:

- Unified telemetry
- Vendor-neutrality
- Cross-platform

OpenTelemetry in a nutshell

What it is

2nd largest CNCF project by contributor count

A set of various things focused on letting you collect telemetry about systems:

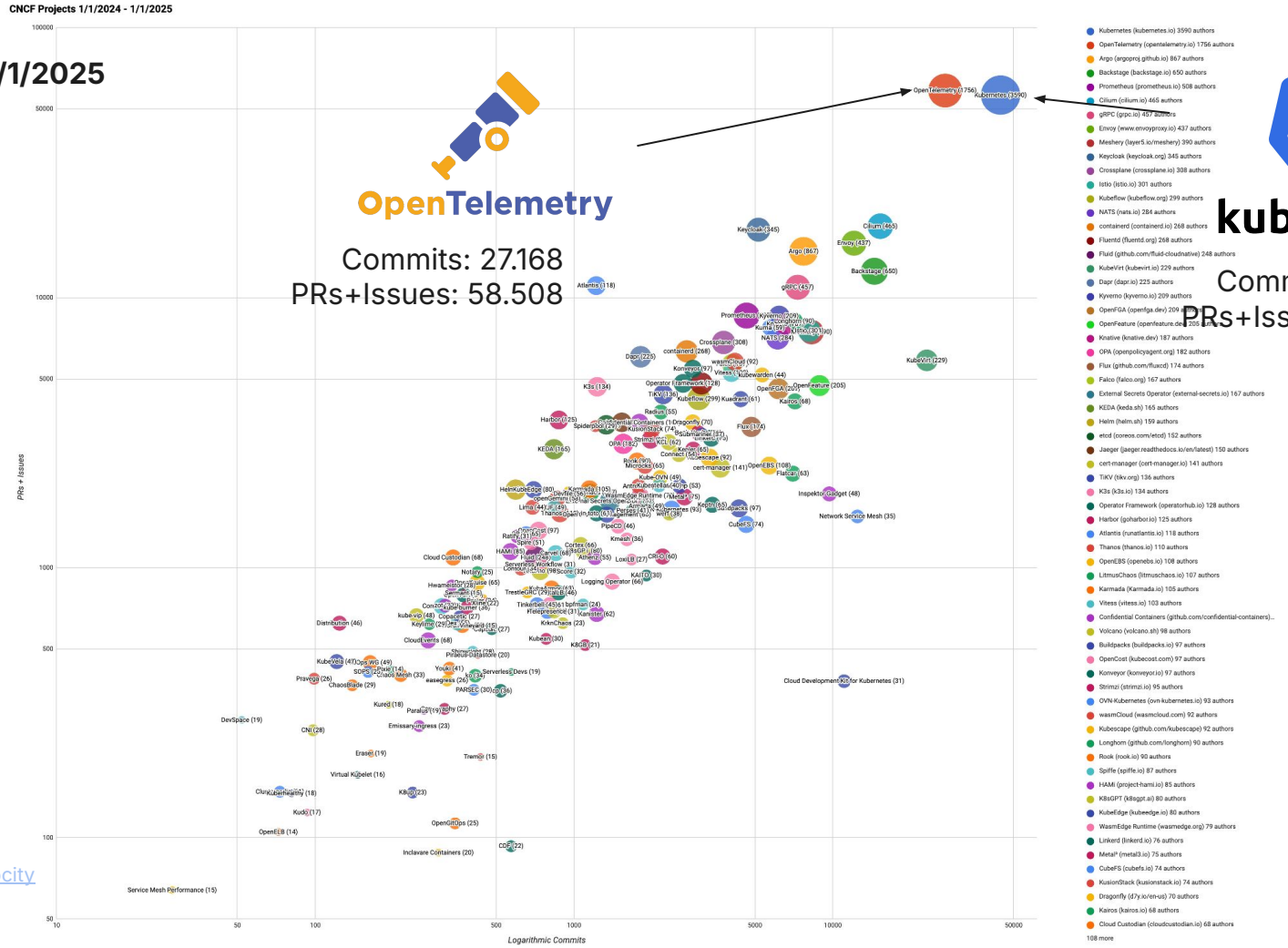
- Data models
- API specifications
- Semantic conventions
- Library implementations in many languages
- Utilities
- and much more

OpenTelemetry in a nutshell

What it is NOT

- Proprietary
- An all-in-one observability tool
- A data storage or dashboarding solution
- A query language
- A Performance Optimizer
- Feature complete

1/1/2024-1/1/2025



OpenTelemetry

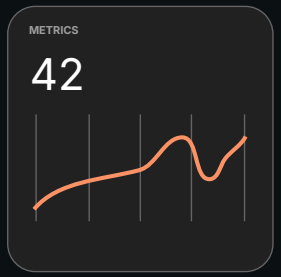
Commits: 27.168
PRs+Issues: 58.508

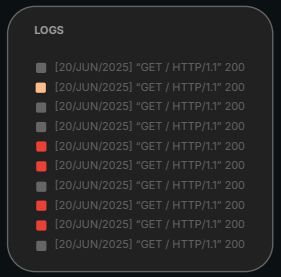
kubernetes

Commits: 44.486
PRs+Issues: 56.299

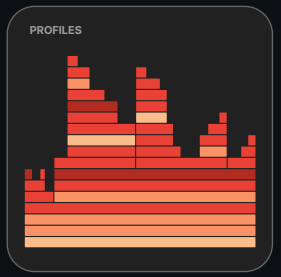
Source: [CNCF Velocity]

Signals


METRICS
42


LOGS
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200
[20/JUN/2025] "GET / HTTP/1.1" 200


TRACES


PROFILES


RUM

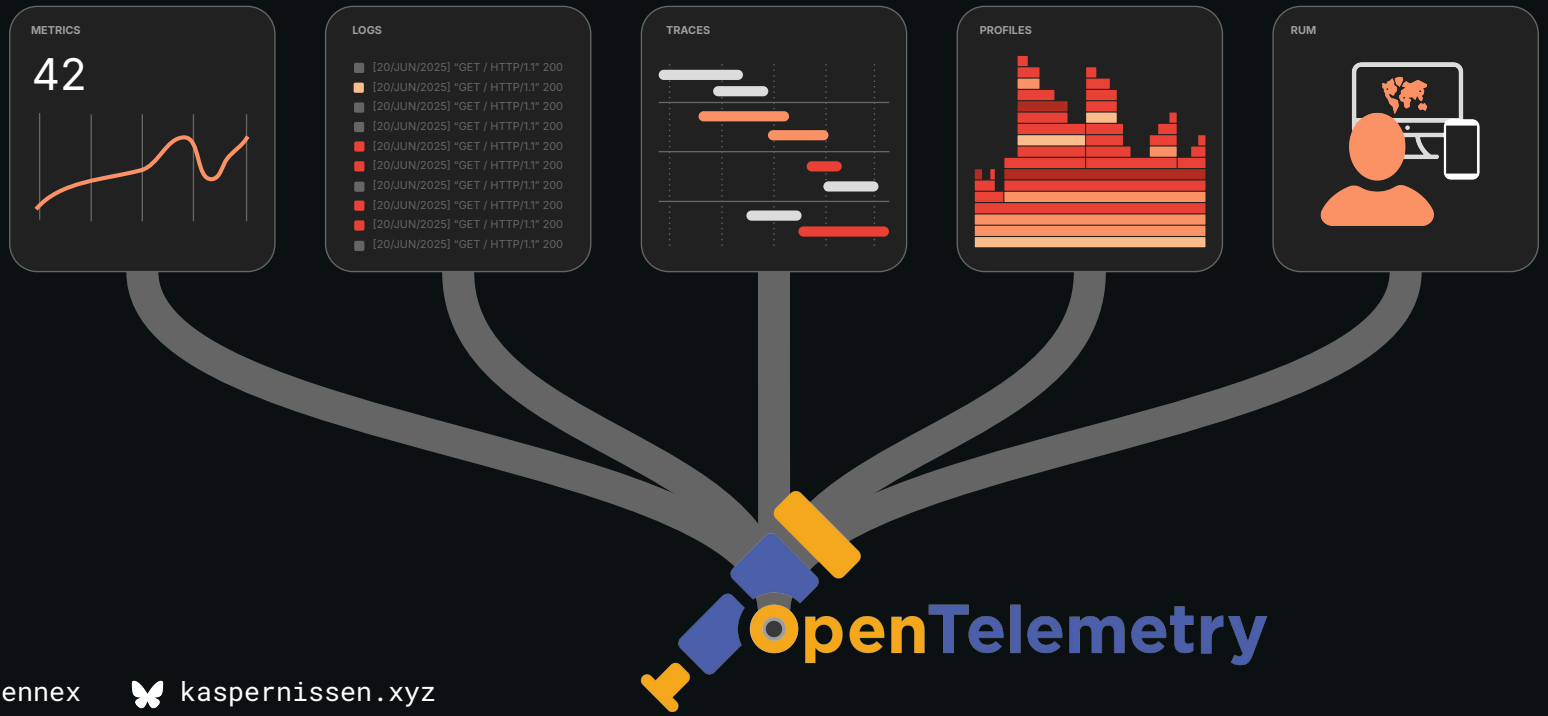
So let’s stop talking about pillars ...

So let’s stop talking about pillars ...

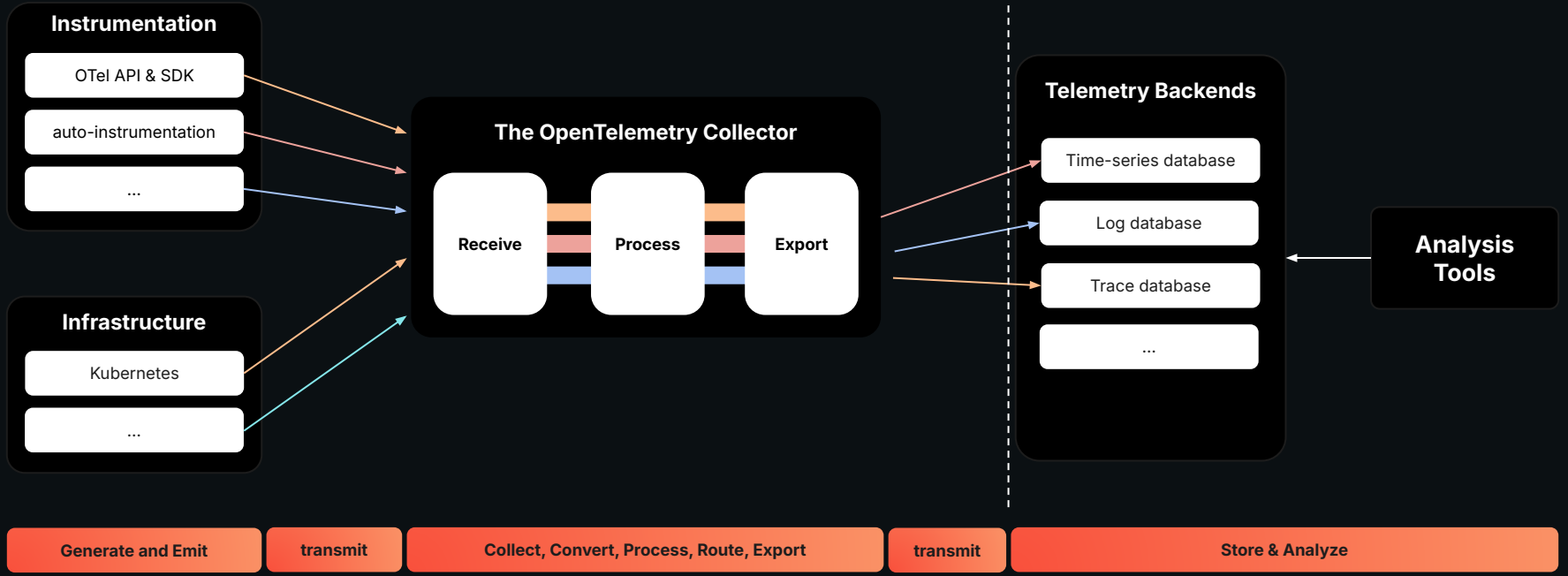
**We don’t have a *metrics* problem,
or a *tracing* problem.
We have *systems* problems.**

Metrics Logs Traces

Correlation is the superpower



OpenTelemetry: A 1000 miles view



OpenTelemetry: A 1000 miles view

Collection of Telemetry is standardized

OpenTelemetry

"The last observability agent you will ever install"

Vendor space

dash0

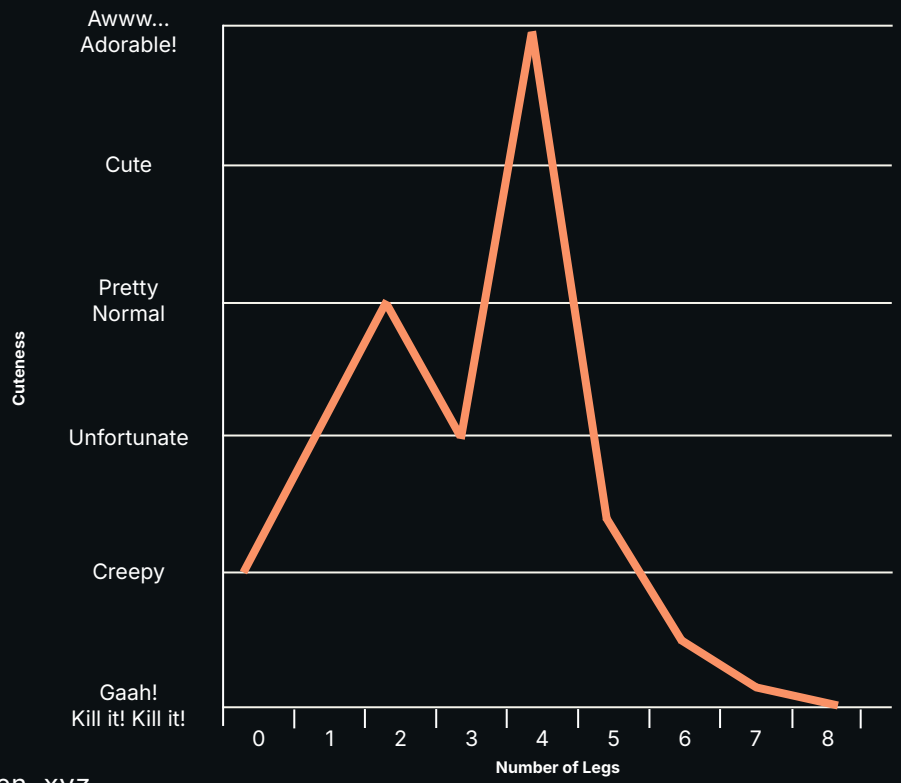
... and many more.

Generate and Emit	transmit	Collect, Convert, Process, Route, Export	transmit	Store & Analyze

> Telemetry without context is just data

What are we looking at?

What are we looking at?



Reddit /r/funny, "Cuteness Vs Number of legs" (circa 2010)

How we talk about system context

Organization (By whom)
1
Which team owns it?
“Who you gonna call?”
...

Architecture (What / Why)
Which **service** / system component is this?
2

Compute (How/2)
3
Which container?
Which process? Pid? Startup args?
Which runtime is it? Node.js? JVM?
.NET? Which build? Which version?
...

Platform (How)
Kubernetes? Which cluster / namespace / deployment / cronjob / job / pod?
AWS ECS? Which cluster / service / task?
...
4

Infrastructure (Where)
5
Which datacenter / Cloud region / availability zone / account does it run in?
...

How to set resource attributes?

- Resource detectors & manual “hard-coding”.
- `OTEL_RESOURCE_ATTRIBUTES` env var
- Added to telemetry “in transit” using the OpenTelemetry Collector.

```
import { NodeSDK } from '@opentelemetry/sdk-node';
import { ConsoleSpanExporter } from '@opentelemetry/sdk-trace-node';
import { envDetector, processDetector, Resource} from '@opentelemetry/resources';
import { awsEcsDetector } from '@opentelemetry/resource-detector-aws';

const sdk = new NodeSDK({
 traceExporter: new ConsoleSpanExporter(),
 // Skip metric exporter, auto-instrumentations and more. See
 // https://opentelemetry.io/docs/languages/js/getting-started/nodejs/
 instrumentations: [getNodeAutoInstrumentations()],
 // Specify which resource detectors to use
 resourceDetectors: [envDetector, processDetector, awsEcsDetector],
 // Hard-coded resource
 resources: [new Resource({
   team: 'awesome',
 })],
});

sdk.start();
```

Sample initialization of the OpenTelemetry JS Distro in a Node.js application

OpenTelemetry without ~~context~~ semantic conventions is just data

@phennex kaspernissen.xyz

dash0

Semantic Conventions

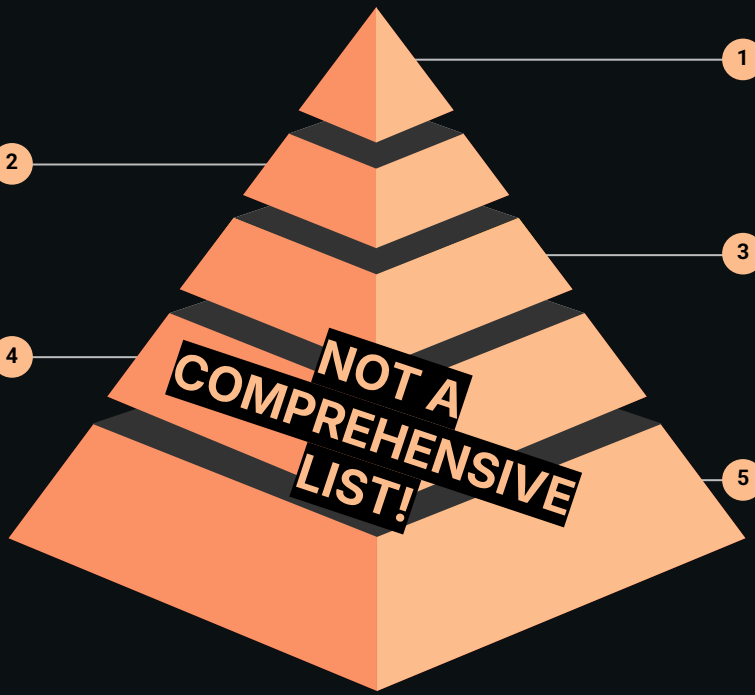
Semantic Conventions define a common set of (semantic) attributes which provide meaning to data when collecting, producing and consuming it.

https://github.com/open-telemetry/semantic-conventions

Semantic Conventions by signals:

- Events: Semantic Conventions for event data.
- Logs: Semantic Conventions for logs data.
- Metrics: Semantic Conventions for metrics.
- Resource: Semantic Conventions for resources.
- Trace: Semantic Conventions for traces and spans.

OpenTelemetry semantic conventions to context layers



1 **Organization**
😢

2 **Architecture**
Service (stable) and (experimental)
Deployment Environment

3 **Compute**
Telemetry SDK (stable) and (experimental)
Compute Unit and Instance
Operating System
Process & Process Runtimes
Device, Browser, Webengine, …

…

4 **Platform**
Kubernetes
Cloud (`cloud.platform` specifically)
Cloud-provider specific

5 **Infrastructure**
Cloud (general stuff)

So, why OpenTelemetry?

Instrument once, use everywhere

Separate telemetry generation from analysis

Make software observable by default

Improve how we use telemetry

That’s all great, but how do I make it easily accessible for my developers?

The dual role of Platform Engineers in Observability

@phennex kaspernissen.xyz

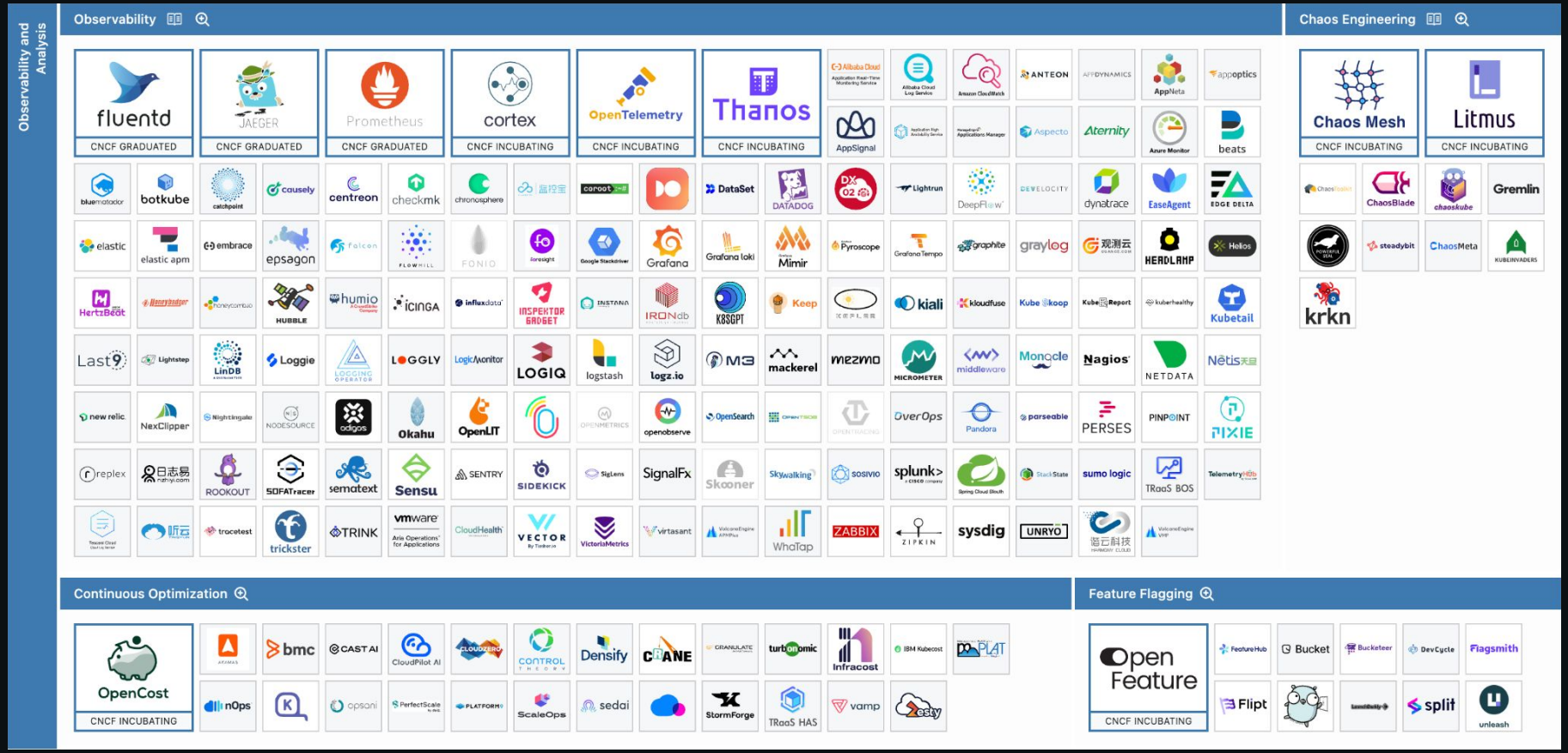
dash0

Observe the platform

(cloud, cluster, CI/CD, shared DBs, etc.)

Enable developers

(traces, metrics, logs, profiling)

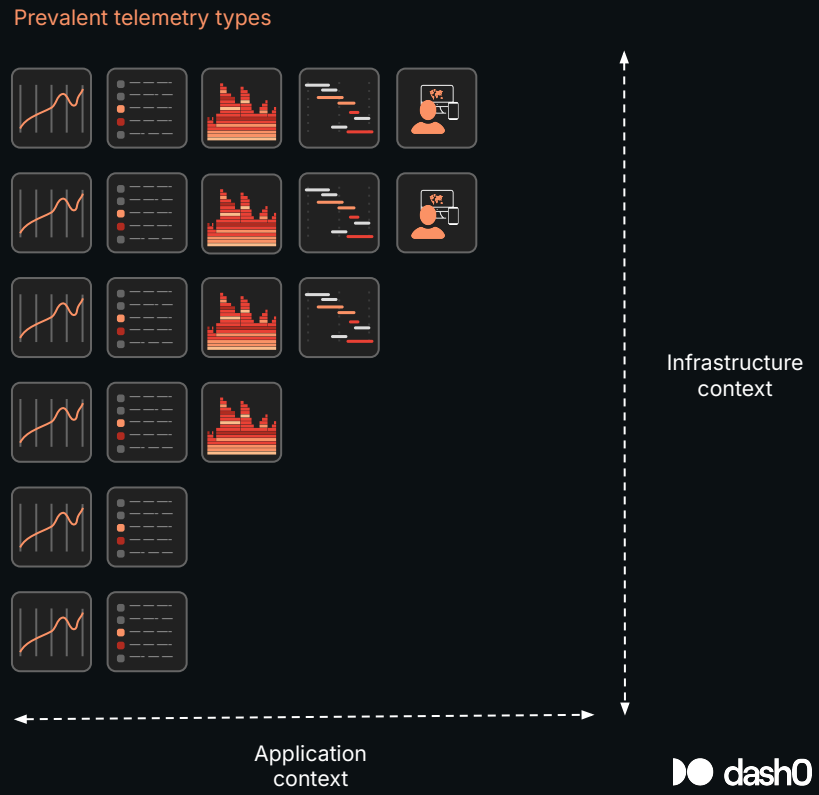


@phennex kaspernissen.xyz

dash0

What types of Telemetry do I need?

- End-user devices and IoT
- Runtimes, applications and services
- Cloud, FaaS, Container orchestration
- Operating system
- Virtualisation
- Bare metal



Based on: "What is observability?" by ubuntu.com

dash0

Platform Engineering for Observability

Self-Service Experience

Explicit and Consistent APIs

Golden Paths

Modularity

Platform as a Product

Core Requirements

Platform Engineering for Observability

Self-Service Experience
Auto-Instrumentation

Explicit and Consistent APIs
Semantic Conventions

Golden Paths
Observability built-in

Modularity
Collector Pipelines

Platform as a Product
Documentation + Support

Core Requirements
Cross-signal correlation

That’s all great, but I ask again, how do I make it easily accessible for my developers?

The answer:
Auto-instrumentation + Operators

=

No-touch Instrumentation

OpenTelemetry Operator

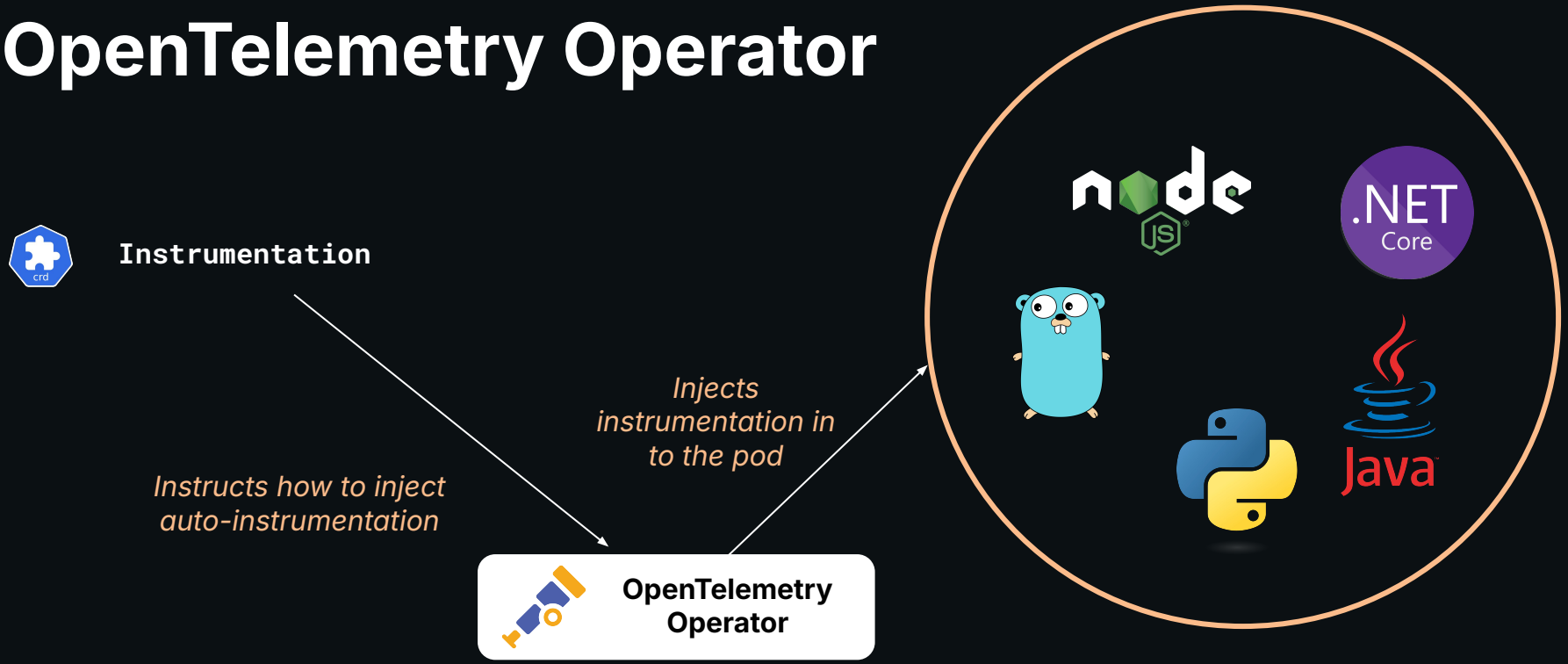
`Instrumentation`

`OpenTelemetryCollector`

`OpAMPBridge`

`TargetAllocator`

Auto-Instrumentation with the OpenTelemetry Operator



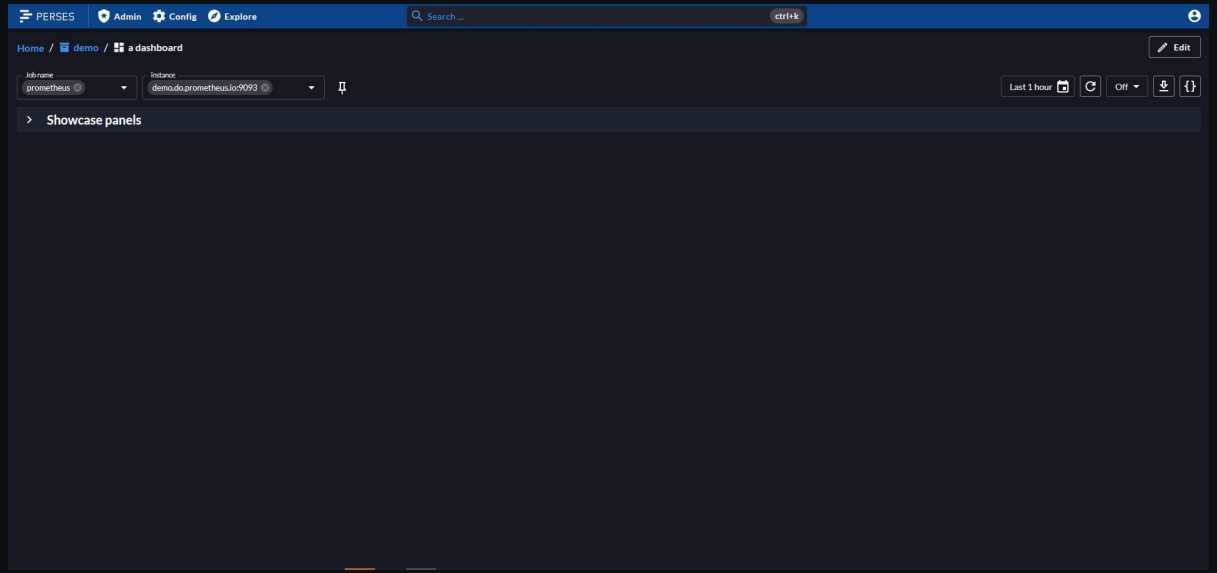
Observability doesn’t stop at instrumentation.

Perses

PERSES

An open specification for dashboards.

CNCF Sandbox project



CLOUD NATIVE COMPUTING FOUNDATION

Dashboards as Code

PERSES

`Perses`

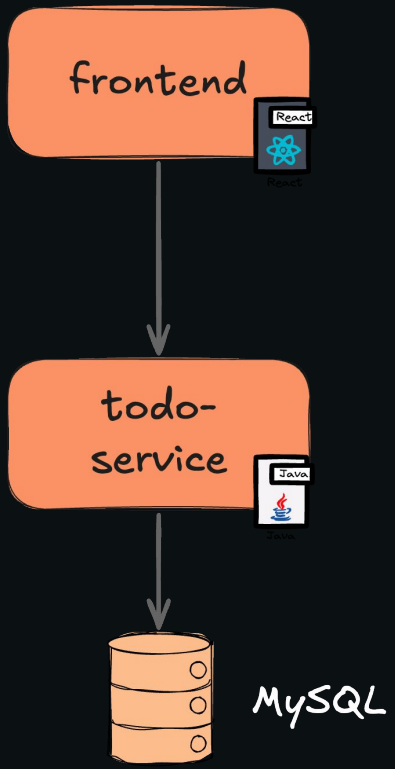
`PersesDatasource`

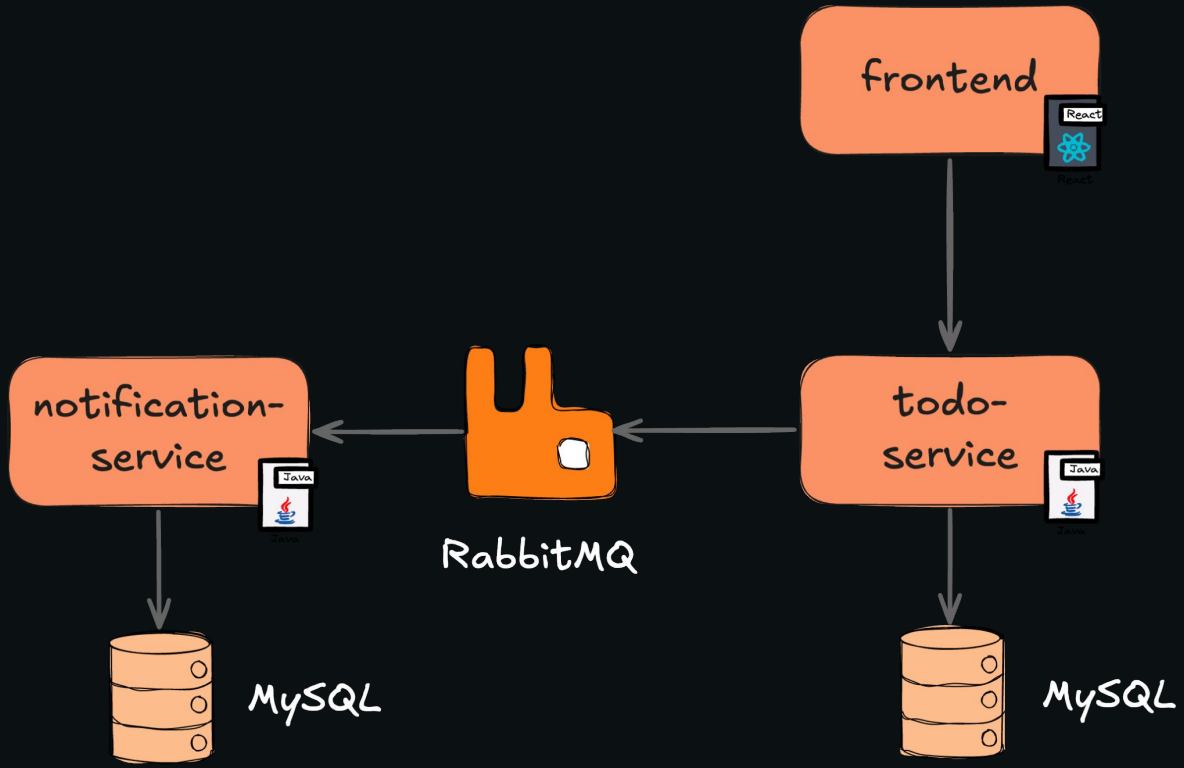
`PersesDashboard`

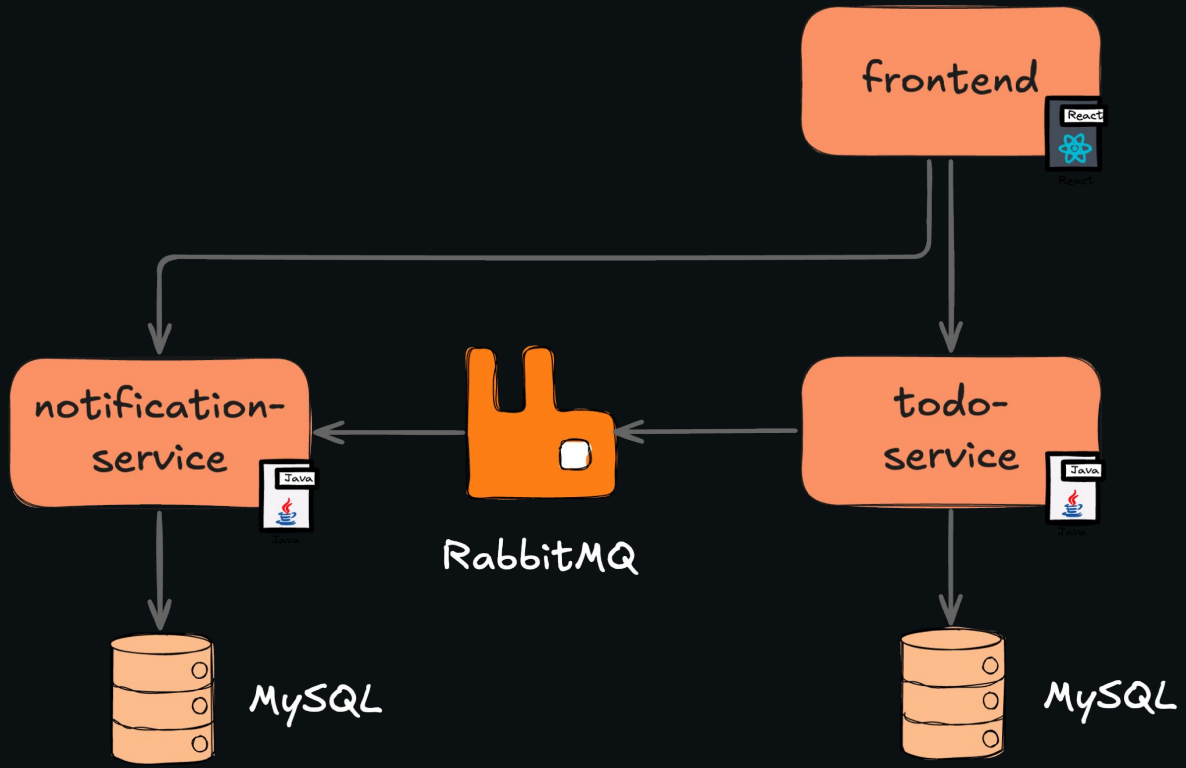
perses-operator

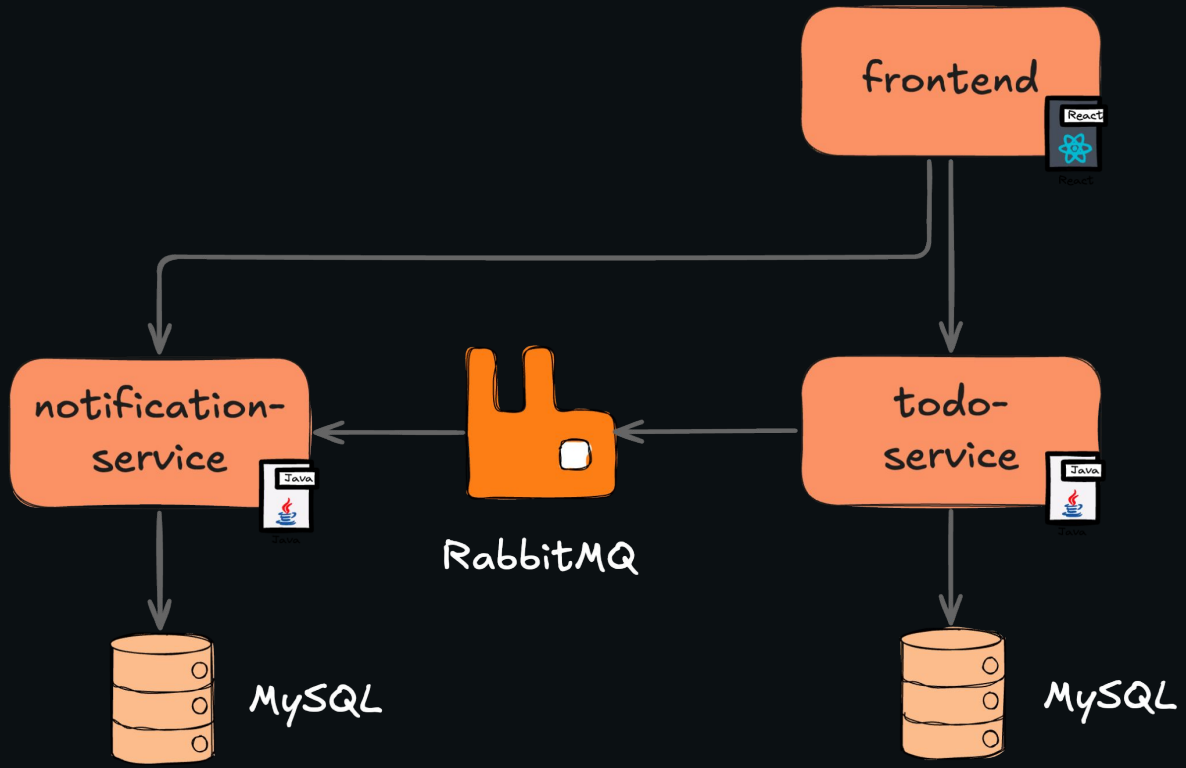
Demo


frontend
React
React


frontend
React
todo-
service
Java
MySQL

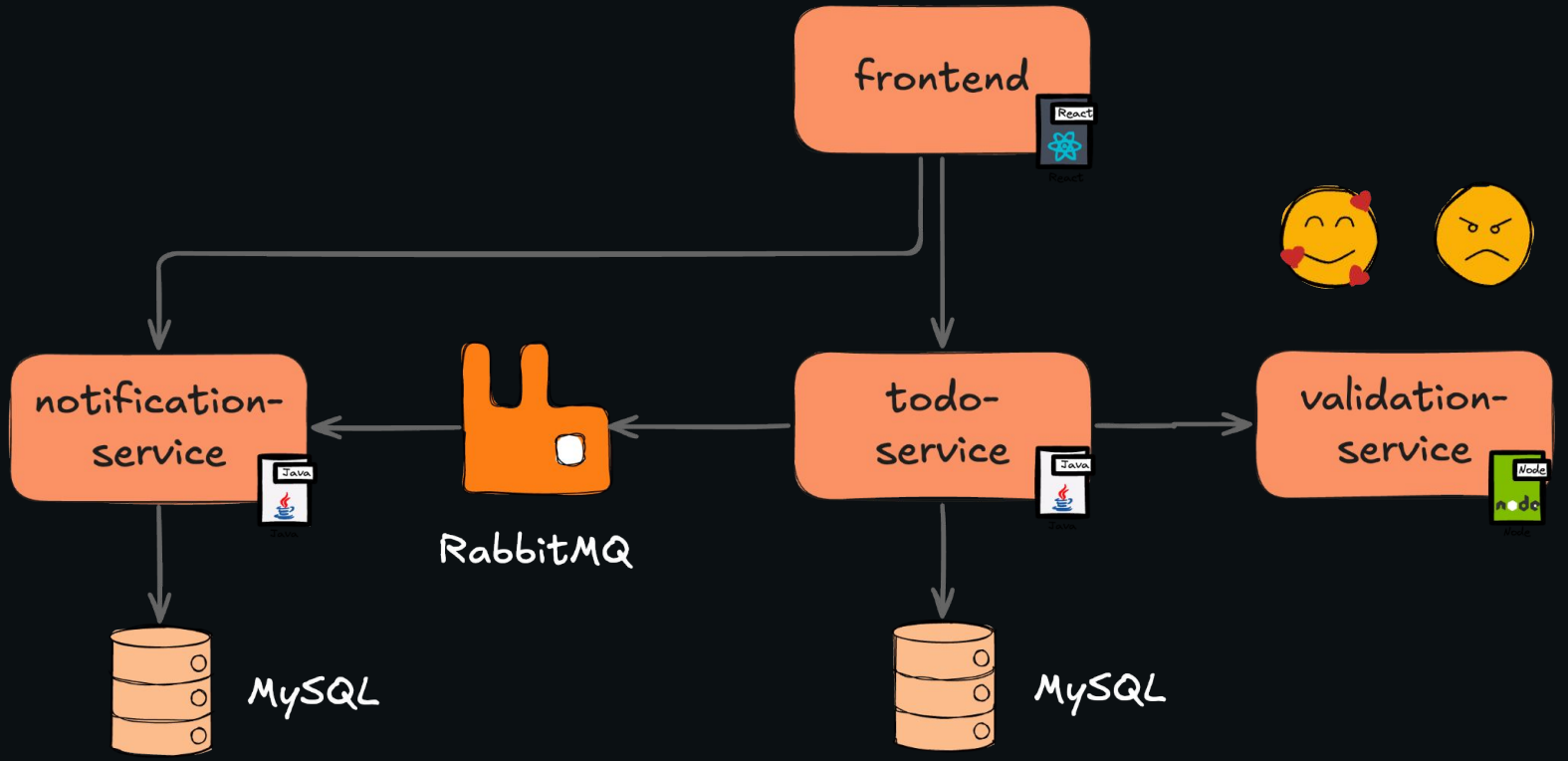

frontend
React
notification-service
Java
RabbitMQ
todo-service
Java
MySQL
MySQL

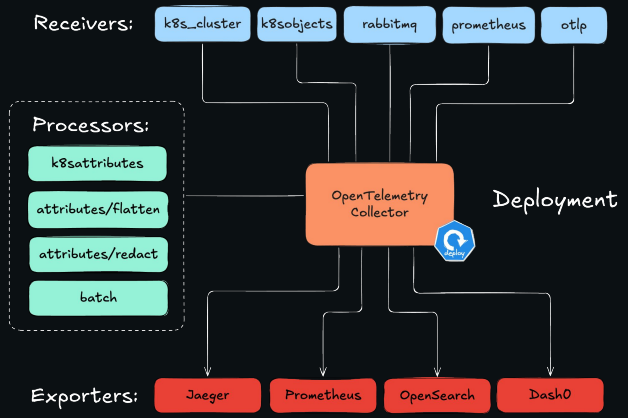

frontend
React
notification-service
Java
RabbitMQ
todo-service
Java
MySQL
MySQL



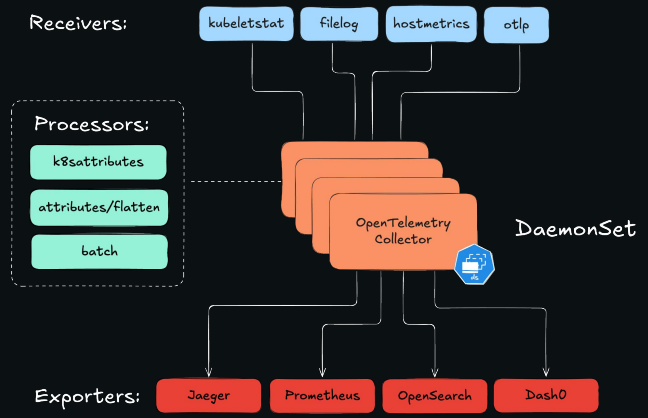
TASK:

Implement a new service that validates the todos and checks them for bad words.


frontend
React
notification-service
Java
RabbitMQ
todo-service
Java
validation-service
Node
MySQL
MySQL

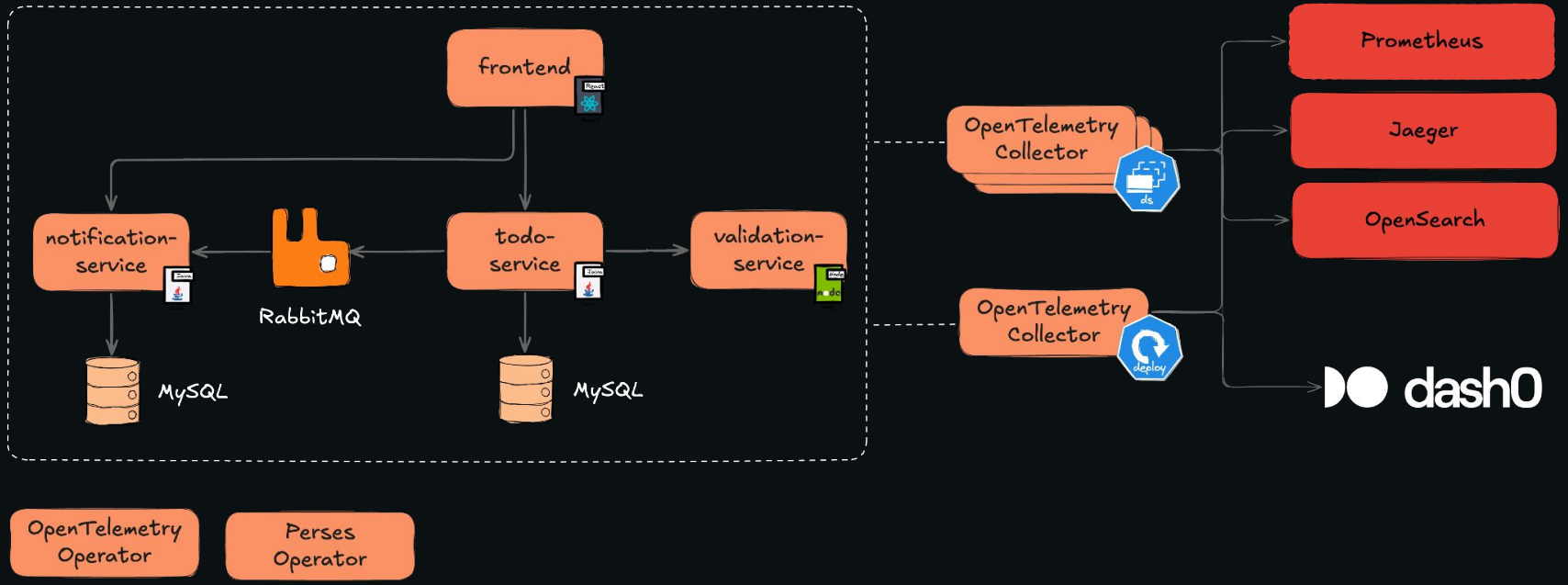


Deployment



DaemonSet

Putting it all together...



Golden defaults for developers

Observability is evolving - fast.

OpenTelemetry is standardizing telemetry collection.

Perses is standardizing dashboarding.

Applying Platform Engineering principles can transform observability from an afterthought into a seamless, scalable, and developer-friendly experience.

Observability is a systems problem
- not a tracing, logging, or metrics problem.

When we connect signals together,
we empower developers to solve problems faster.

And last but not least,

Building on Open Standards allows you to freely move between vendors, ensuring they stay on their toes and provide you the best possible experience.

Observability course for Platform Engineering

Gain a solid foundation of modern observability in platform engineering, from essential concepts to practical tooling.

Platform Engineering

dash0

Sign up for free

https://university.platformengineering.org/observability-for-platform-engineering

Michele Mancioppi

Head of Product @ Dash0

Kasper Borg Nissen

Developer Relations @ Dash0

WHITEPAPER

Observability for Platform Engineering

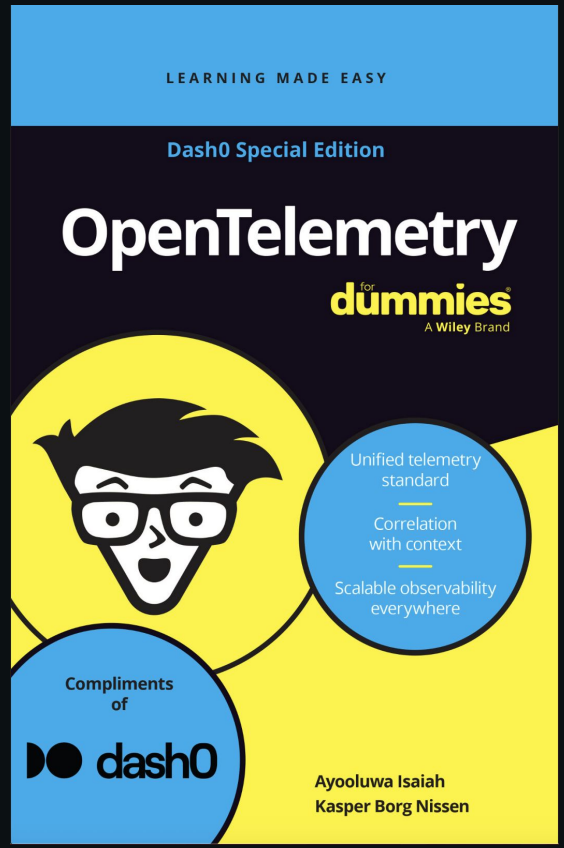
Platform Engineering

dash0

https://platformengineering.org/reports/observability-for-platform-engineers

New book

Get a free copy



dash0

Remember to join the raffle...

Win a Dash0 backpack

dash0

Thank you!

Get in touch!

Kasper Borg Nissen, Principal Developer Advocate at dash0

Demo can be found here!
https://github.com/dash0hq/otel-platform-demo

@phennex kaspernissen.xyz

dash0