dash0

Tech Hub Aarhus Day

Rethinking Observability as a Platform Capability

Kasper Borg Nissen, Principal Developer Advocate at Dash0


kaspernissen.xyz


/in/kaspernissen

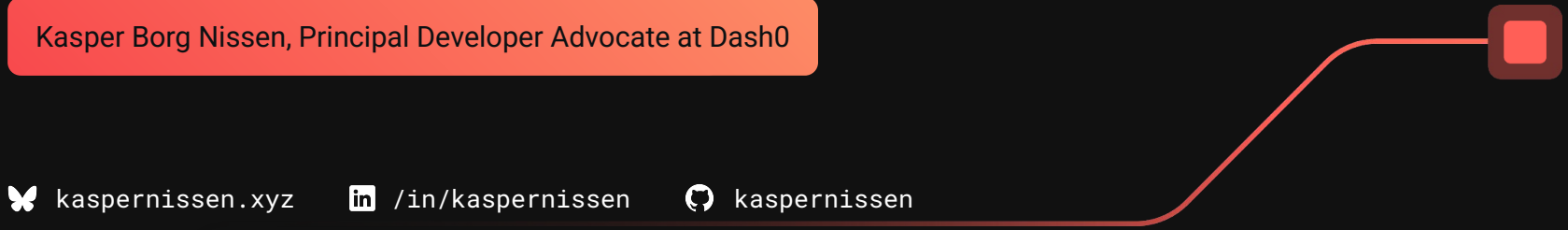

kaspernissen

dash0

Tech Hub Aarhus Day

Rethinking Observability as a Platform ~~Capability~~ Product

Kasper Borg Nissen, Principal Developer Advocate at Dash0



kaspernissen.xyz /in/kaspernissen kaspernissen

Who?

Principal Developer Advocate at Dash0 (previously Lunar)

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

Author of OpenTelemetry for Dummies

CNCF Ambassador

Golden Kubestronaut

CNCG Aarhus, KCD Denmark Organizer

Co-founder & Community Lead Cloud Native Nordics

Let’s talk about Kubernetes

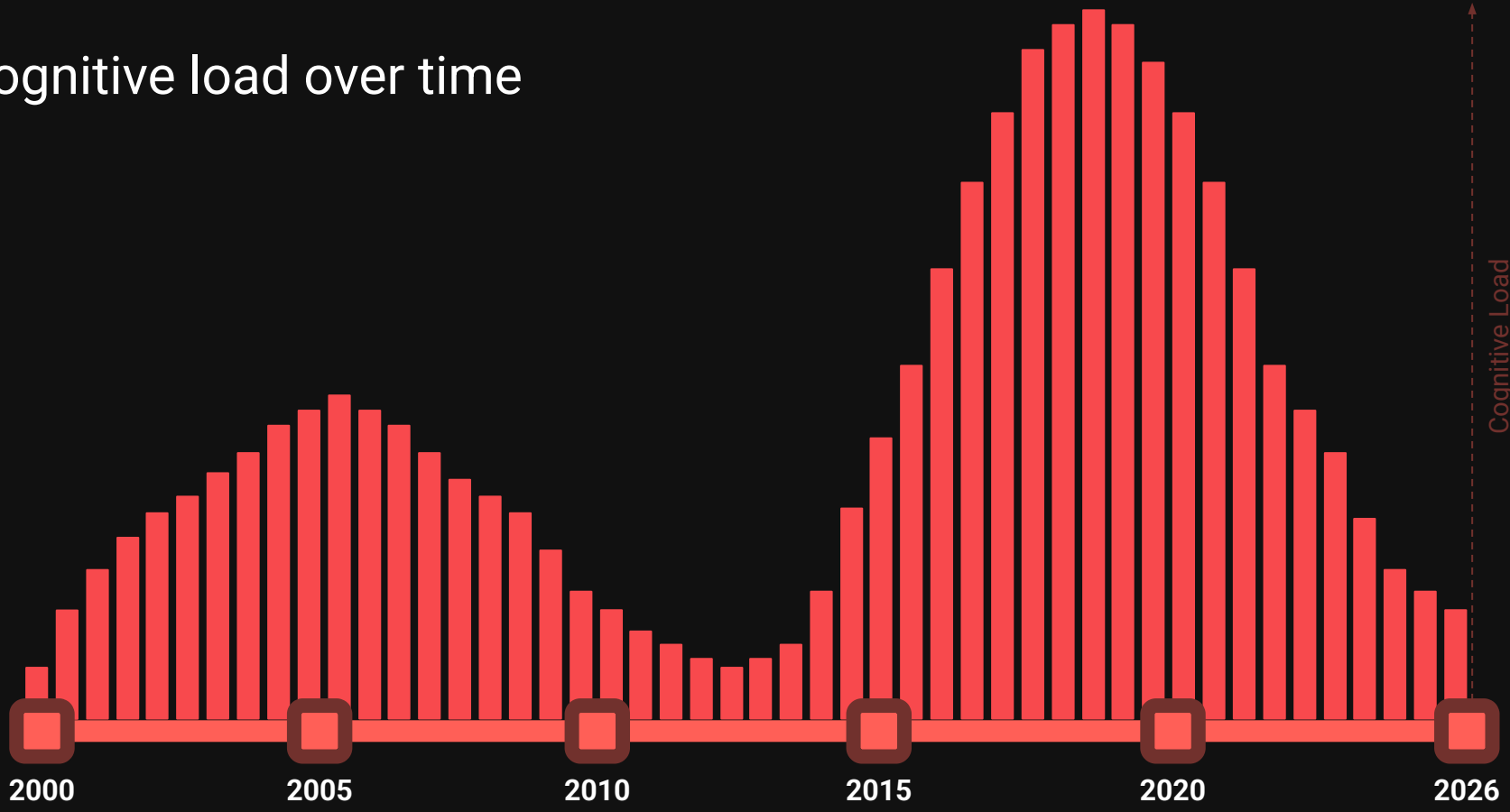
Kubernetes is a platform for building platforms

Bryan Liles

KubeCon+CloudNativeCon in San Diego 2019

The Rails Moment

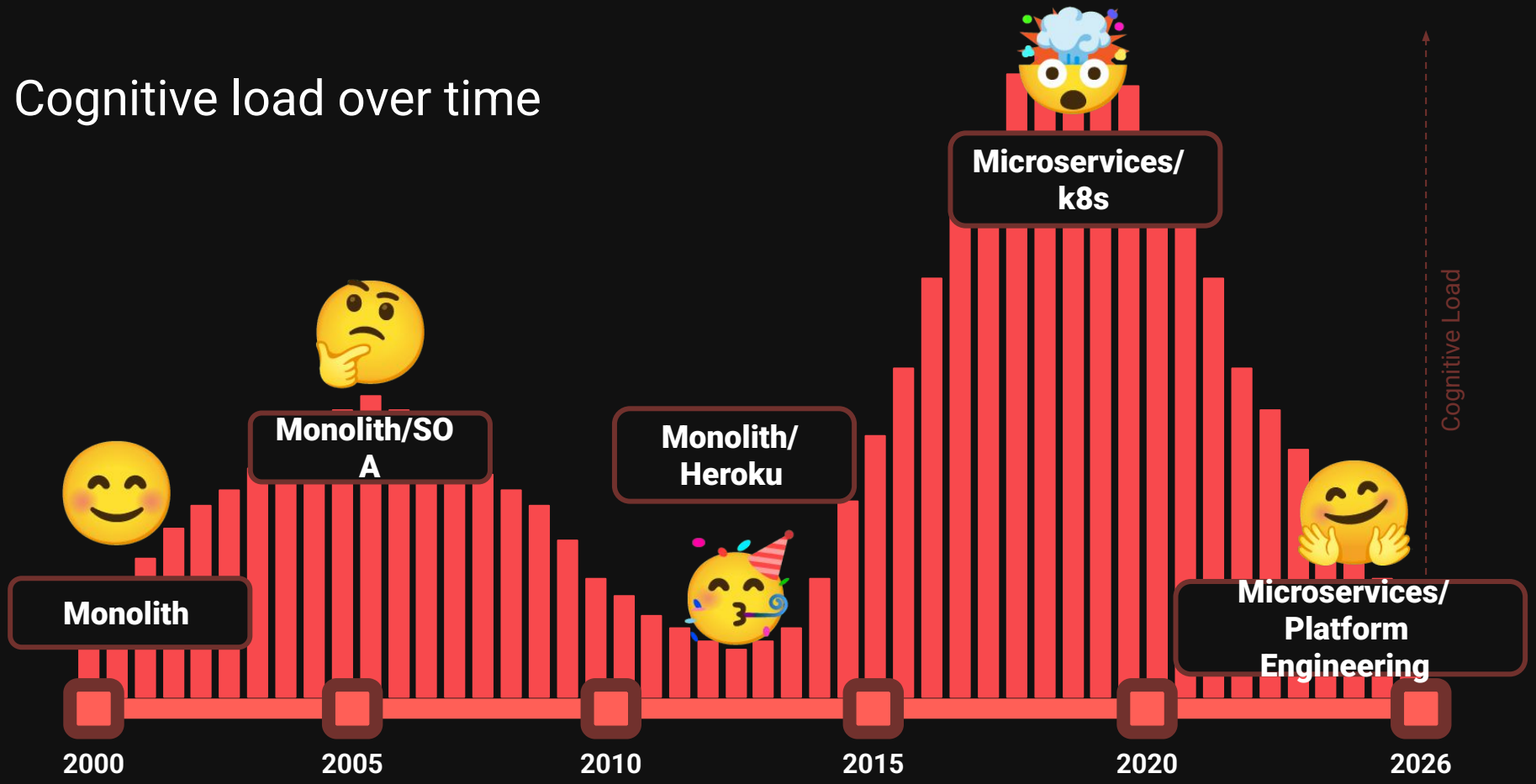
Cognitive load over time



dash0

Visualization Inspired by Daniel Bryant's talk

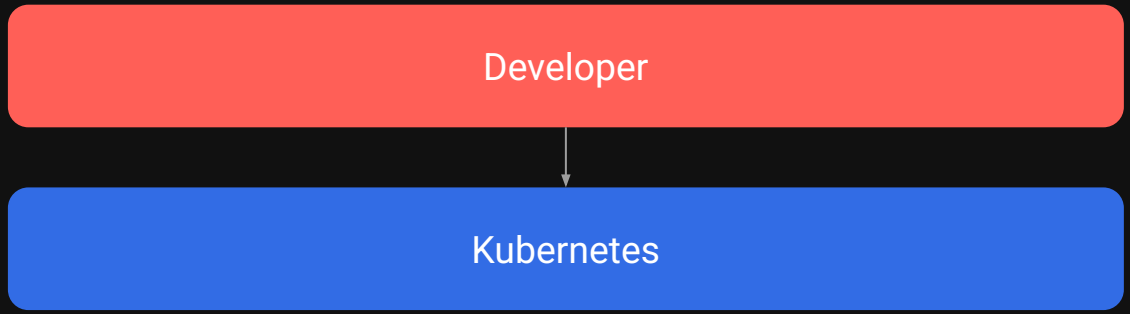
Cognitive load over time



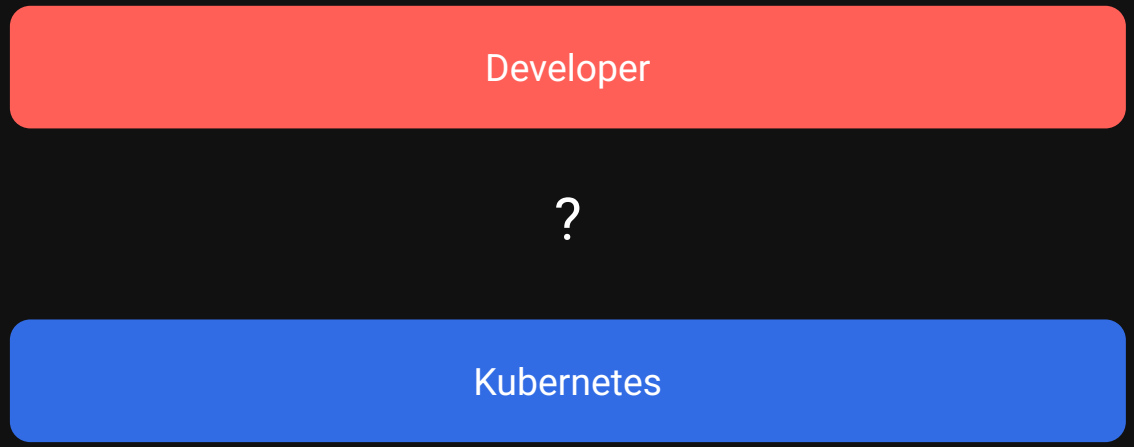
dash0

Visualization Inspired by Daniel Bryant's talk

Why Platform Engineering Emerged



Why Platform Engineering Emerged



Why Platform Engineering Emerged

Developer

Platform

Kubernetes

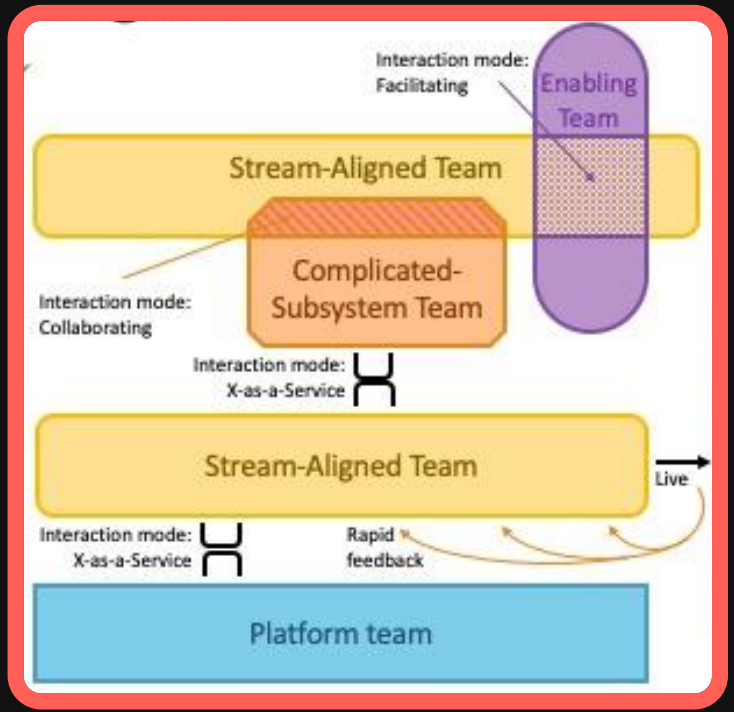
A digital platform is a foundation of self-service APIs, tools, services, knowledge and support arranged as a compelling internal product.

Evan Bottcher

https://martinfowler.com/articles/talk-about-platforms.html

Team Topologies

- Platform Team: a grouping of other team types that provide a compelling internal product to accelerate delivery by Stream-aligned teams



Platform as a Product

- Platform as a Product (PaaP) is an approach where internal platforms are treated as evolving products with defined users (developers, operations teams) and lifecycles.

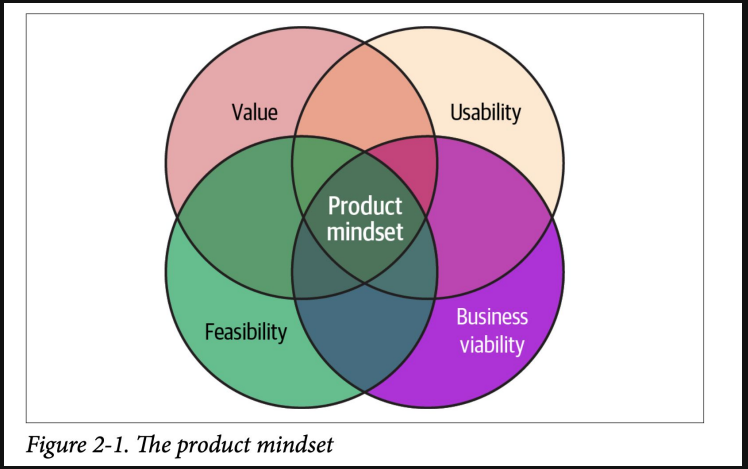
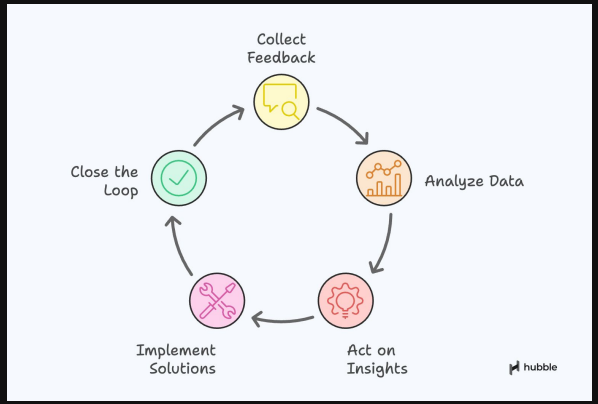
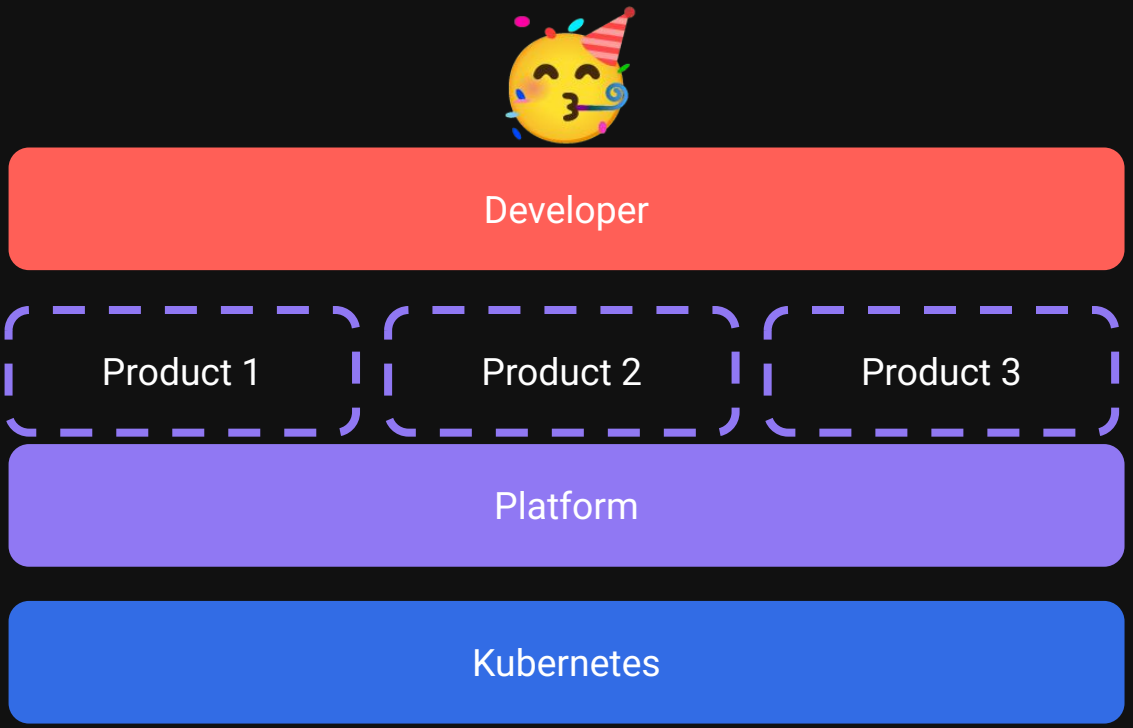


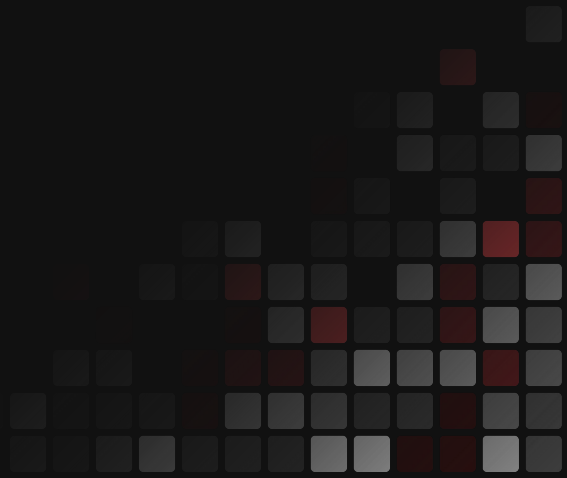
Figure 2-1. The product mindset



The missing abstraction



Observability Today… Same mistakes, different domain



dash0

Observability promised a lot

Faster root cause analysis

Understanding system behavior

Lower MTTR

Reduced guesswork

Expectation

Where do I start?

Which tool is right?

Why is this metric spiking?

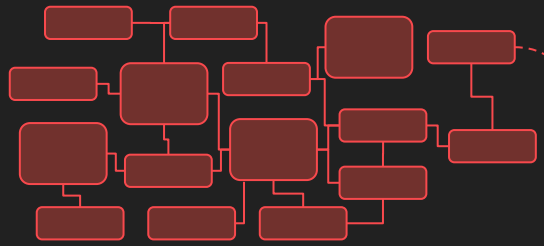
Human correlation

Reality

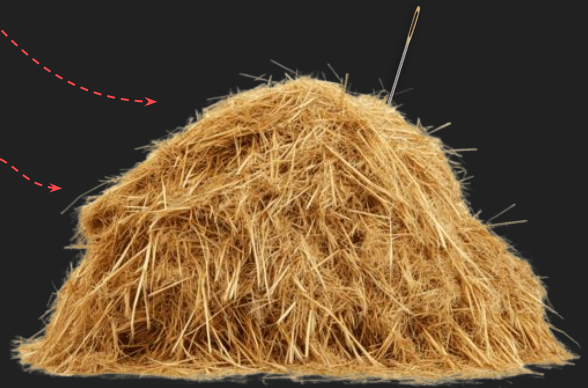
LOGS | METRICS | TRACES | RUM | PROFILING

```
level=DEBUG
level=debug
LEVEL=DEBUG
```

DISTRIBUTED SYSTEMS



100s/1000s of Microservices



Finding the needle in the haystack

dash0

The current reality: fragmentation

The current reality: fragmentation

Complex Query Languages

The current reality: fragmentation

Complex Query Languages

Vendor lock-in

The current reality: fragmentation

Complex Query Languages

Vendor lock-in

Metadata Inconsistency

The current reality: fragmentation

Complex Query Languages

Vendor lock-in

Metadata Inconsistency

No instrumentation due to high complexity

The current reality: fragmentation

Complex Query Languages

Vendor lock-in

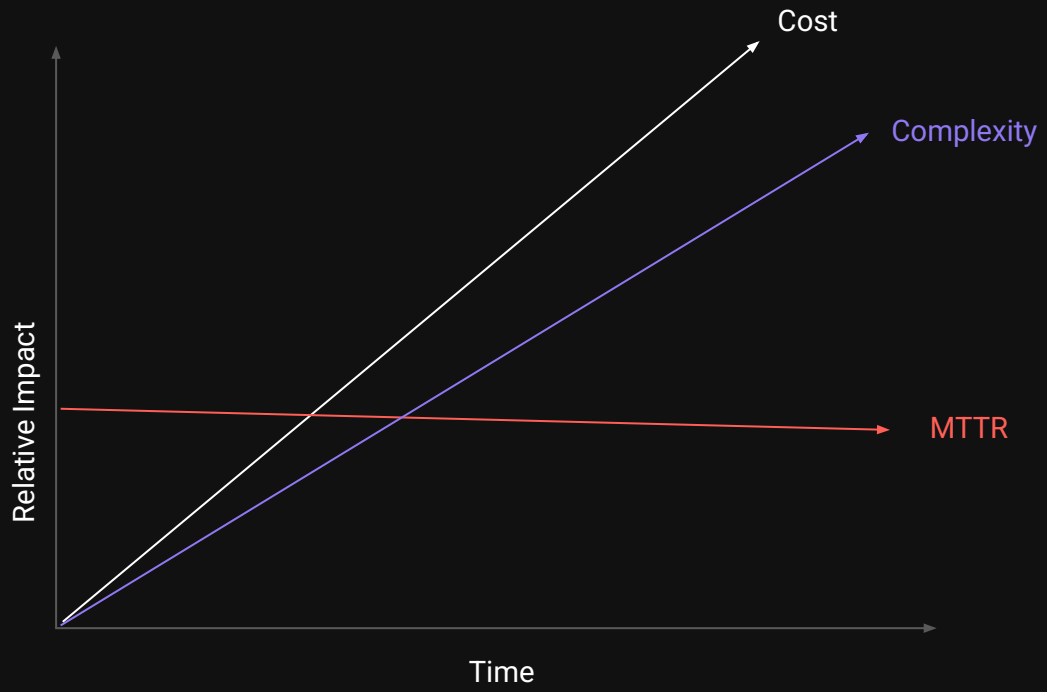
Metadata Inconsistency

No instrumentation due to high complexity

Lack of unified insights

The cost & complexity paradox

More telemetry, more tooling - same time to recovery



Up to 84% of current observability users struggle with the costs and complexity of their daily monitoring responsibilities.

Gartner Hype Cycle Report, 2025

Source: https://www.gartner.com/en/documents/6755734

This isn’t necessarily a tooling problem

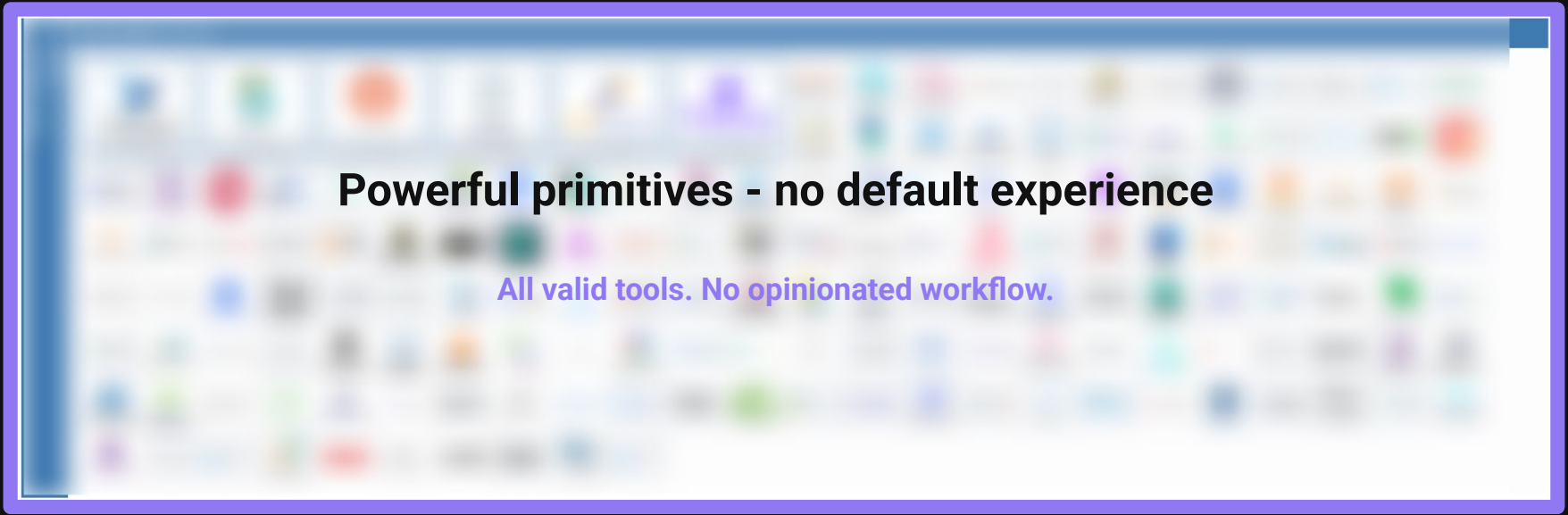
**We don’t have a *metrics* problem,
or a *tracing* problem.
We have *systems* problems.**

Metrics Logs Traces

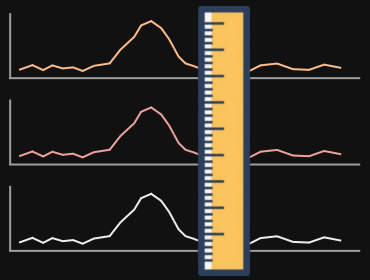
Let’s stop talking about the three pillars of observability …

Metrics Logs Traces

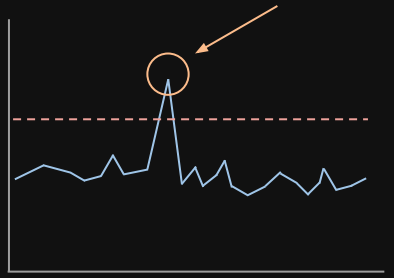
Same mistake as early Kubernetes



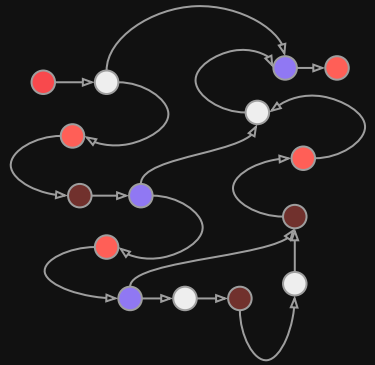
A shift toward correlation



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, supports metrics, logs, profiling & RUM

The main goals of the project are:

- Unified telemetry
- Vendor neutrality
- Cross platform

OpenTelemetry in a nutshell

OpenTelemetry is a toolkit and a specification.

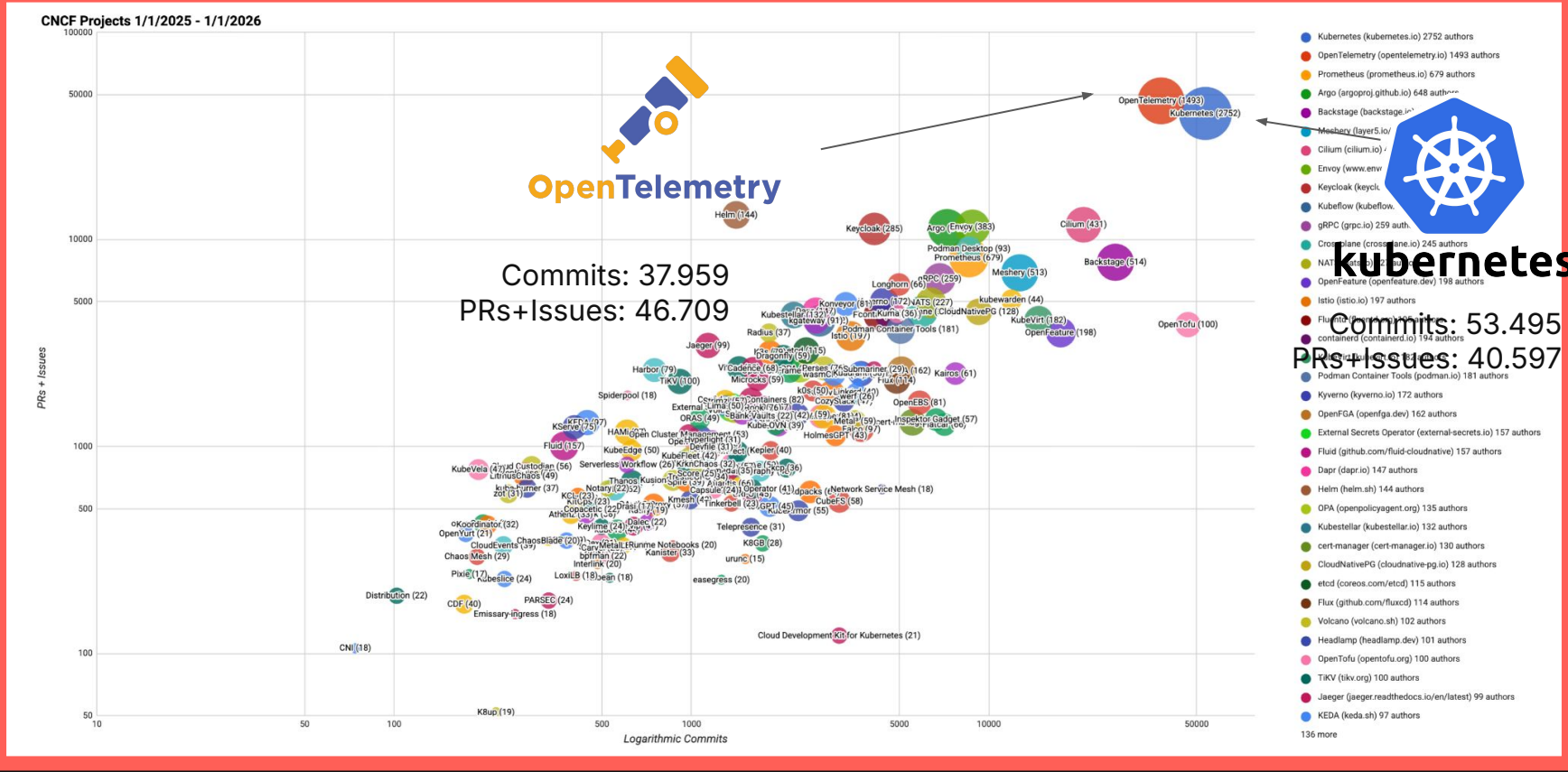
What it is

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

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

1/1/2025-1/1/2026



dash0

Source: CNCF Velocity Report

49%

of respondents using OpenTelemetry in production.

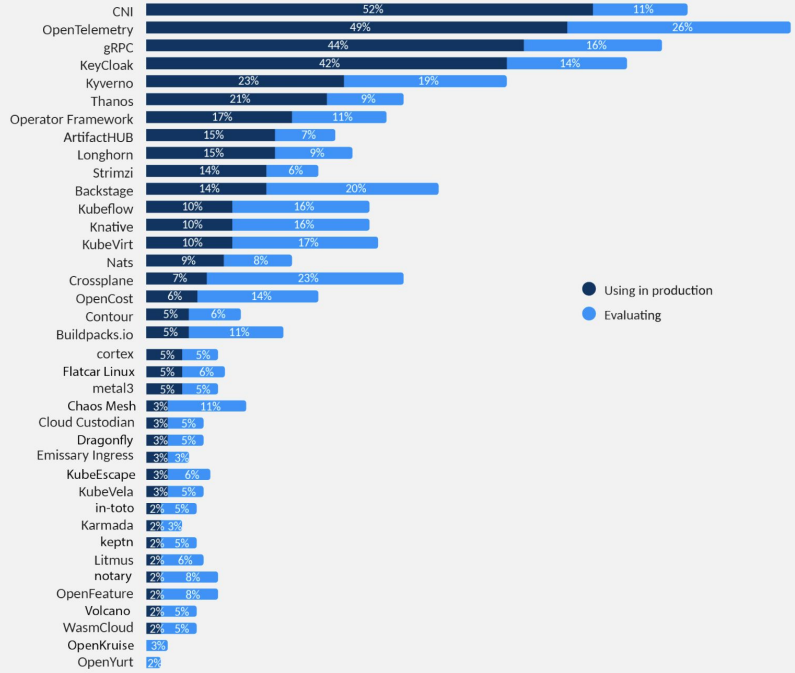
26%

of respondents evaluating OpenTelemetry.

FIGURE 20

CNCF INCUBATING PROJECTS

Which of these incubating CNCF projects is your organization using in production or evaluating? (select one)

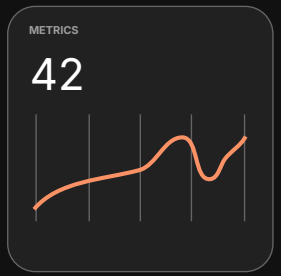


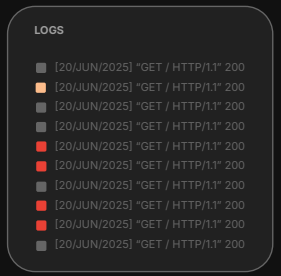
2025 CNCF Annual Survey, Q33, Sample size = 395-502, DKNS excluded

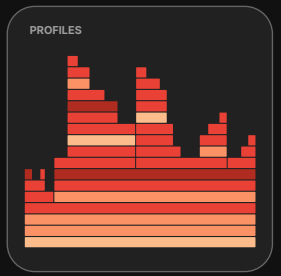
dash0

Source: https://www.cncf.io/wp-content/uploads/2026/01/CNCF_Annual_Survey_Report_final.pdf

Signals

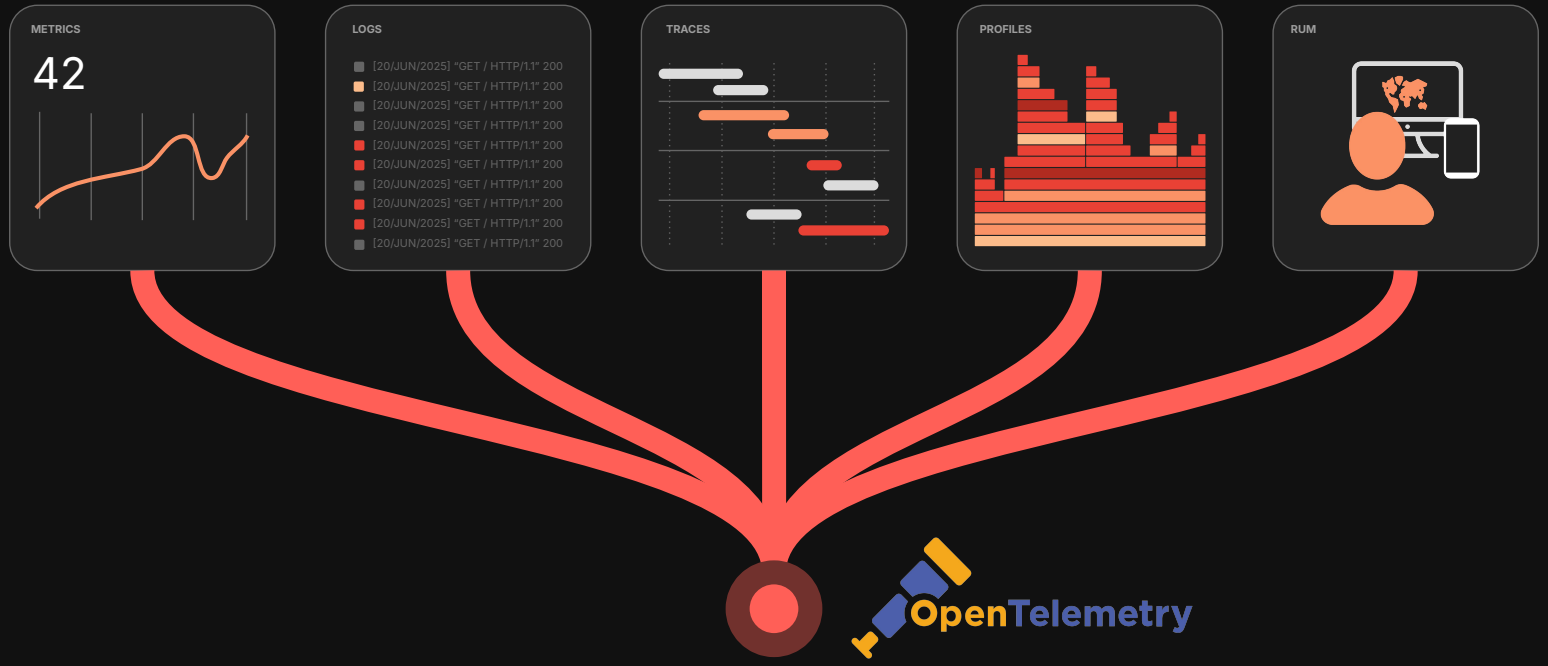




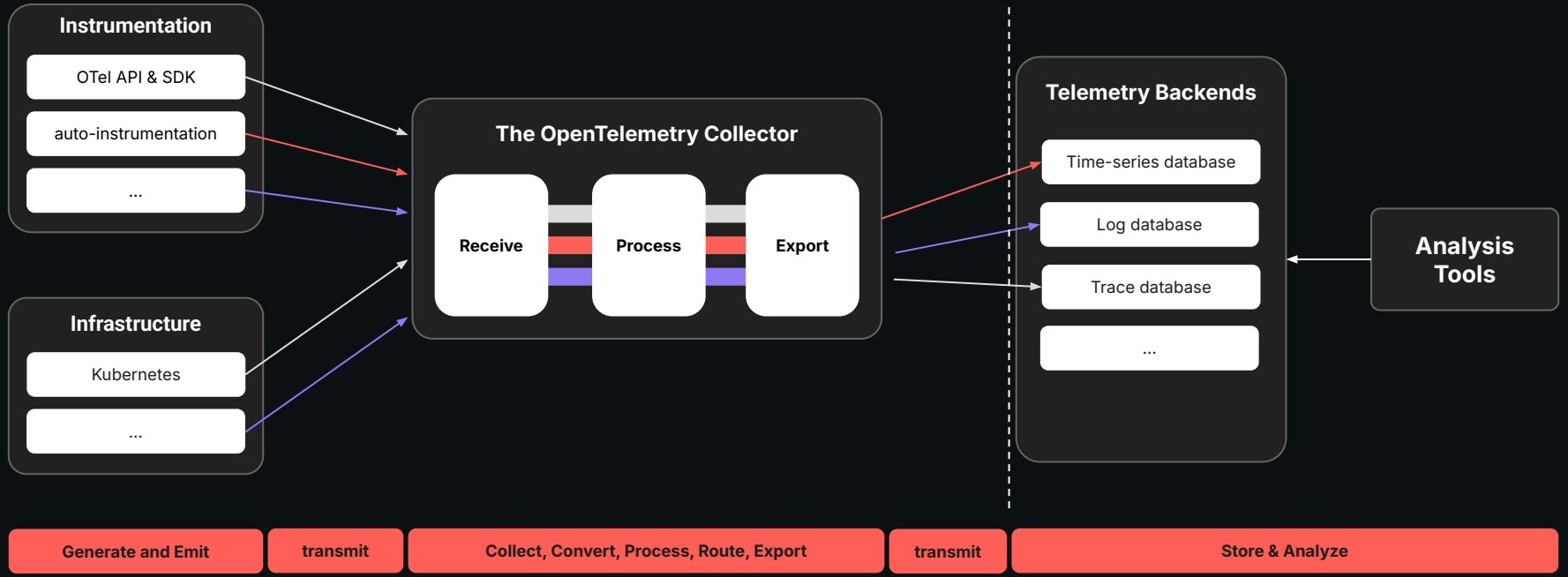


dash0

Correlation is the superpower



OpenTelemetry: A 1000 miles view



dash0

Inspired by visualizations from LFS148

OpenTelemetry: A 1000 miles view

Collection of Telemetry is standardized

"The last observability agent you will ever install"

Vendor space

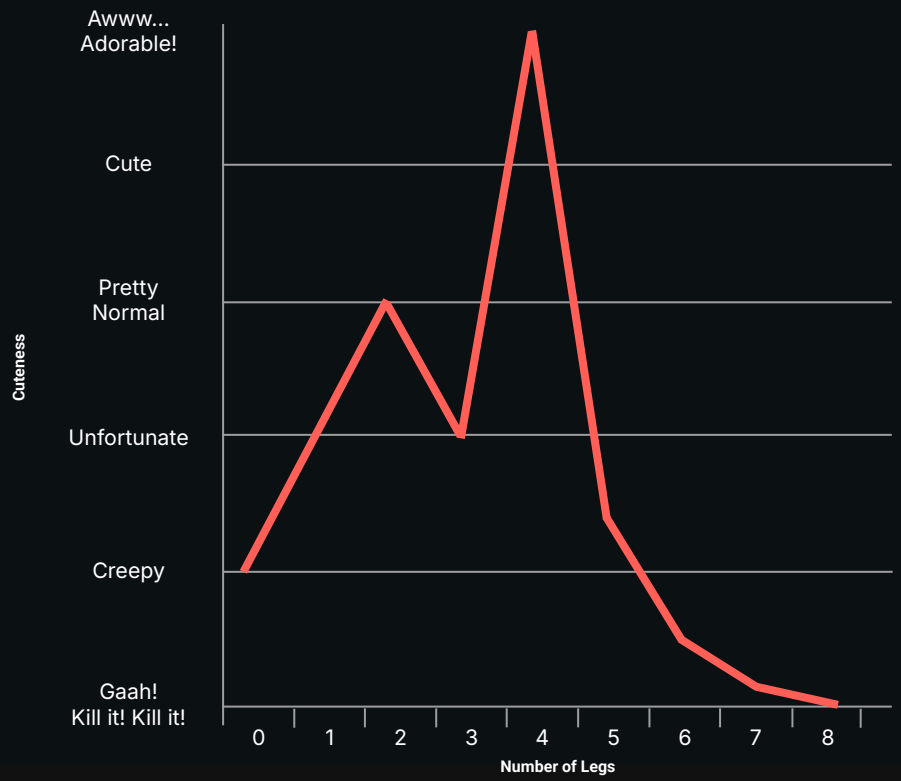
... and many more.

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)
Which team owns it?
“Who you gonna call?”

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

Compute (How/2)
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? …

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

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)

The Platform Team's Role

Observe the platform

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

Enable developers

traces, metrics, logs, profiling

What does “Observability as a Product” mean?

The platform absorbs complexity so teams can focus on understanding systems.

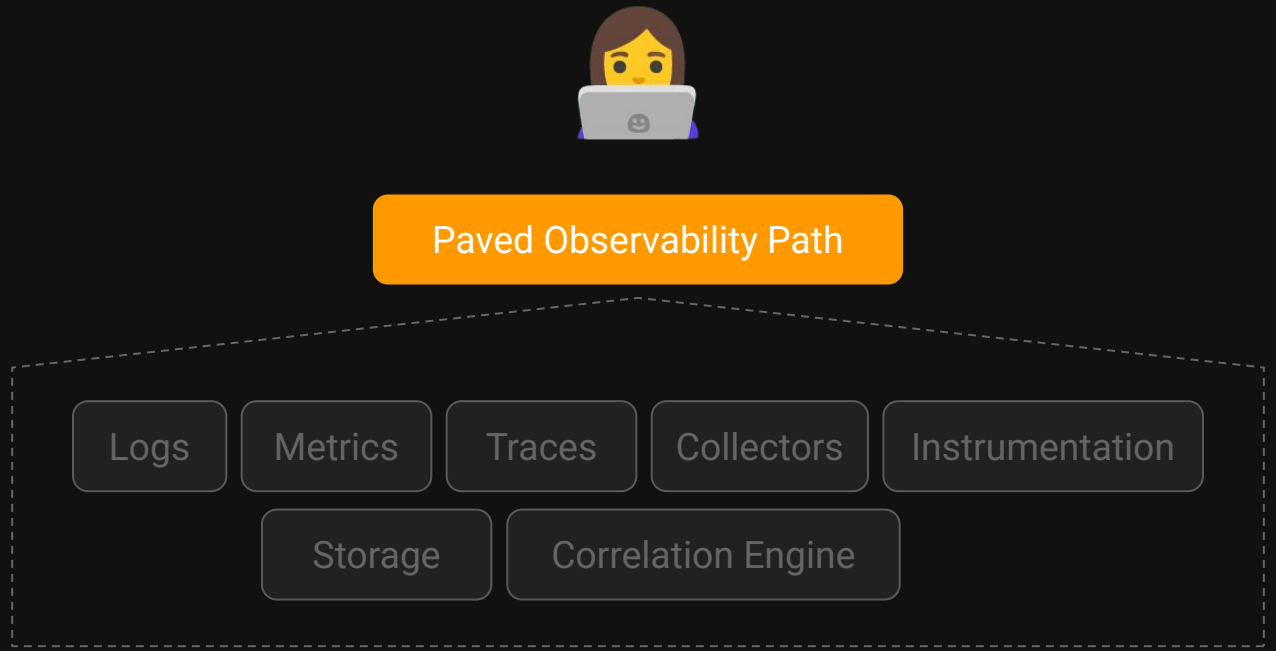
Observability as tooling

- Terminal
- YAML
- Dashboards with empty states
- Many knobs

Observability as a Product

- Clean UI
- Pre-populated dashboards
- Correlated views
- “It just works”

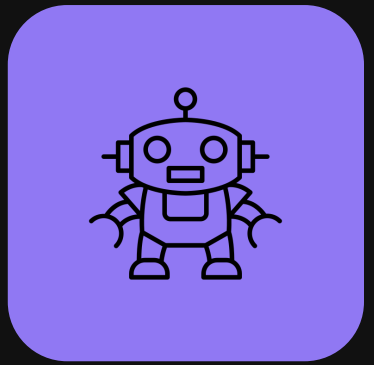
Paved Paths for Observability



Auto-instrumentation changes the game

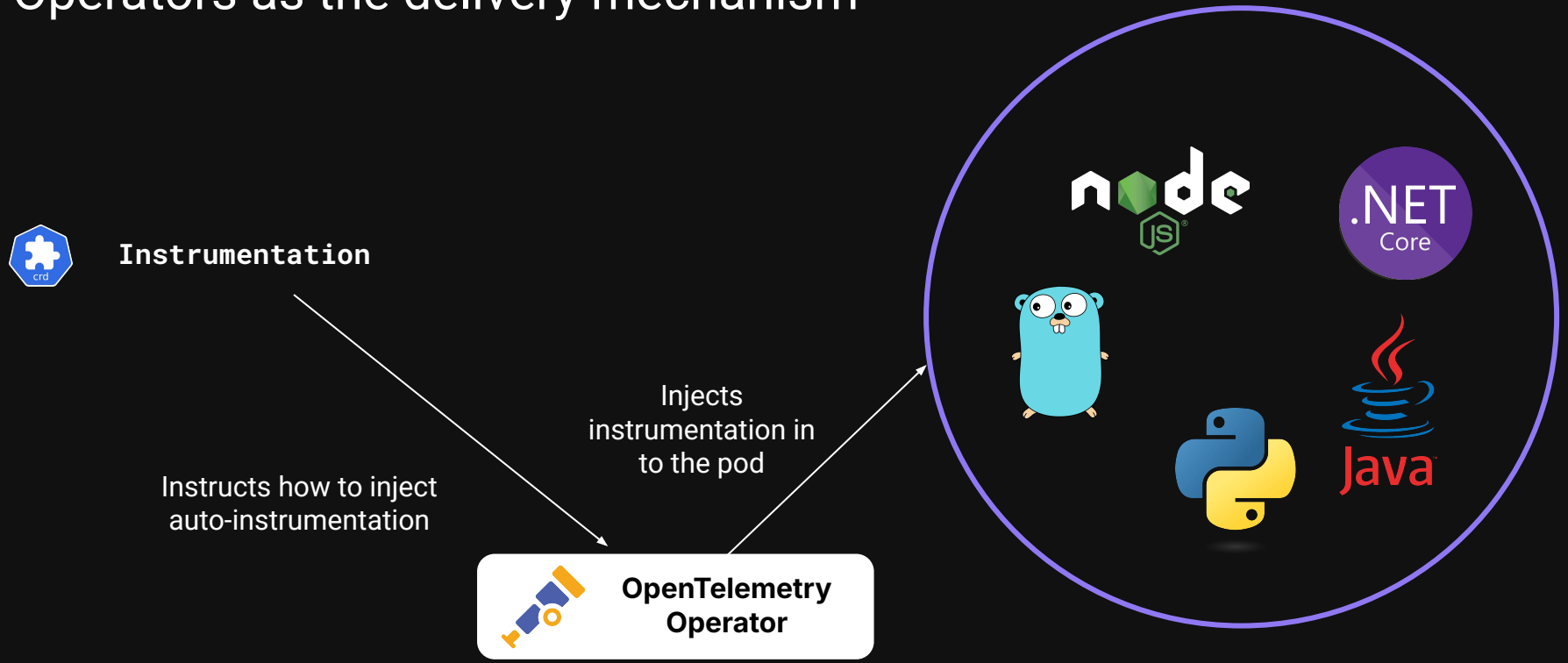
Manual Instrumentation
(fully code-based)

Automatic Instrumentation
(agent, binaries)



No-touch Instrumentation
(or zero-code)

Operators as the delivery mechanism



Observability doesn’t stop at instrumentation

How humans and agents understand the system

- Explorers
- Dashboards
- Alerting
- Synthetic Checks
- Service Maps
- Agents
- Cost Insights

Vendors

dash0

… and many more.

OSS

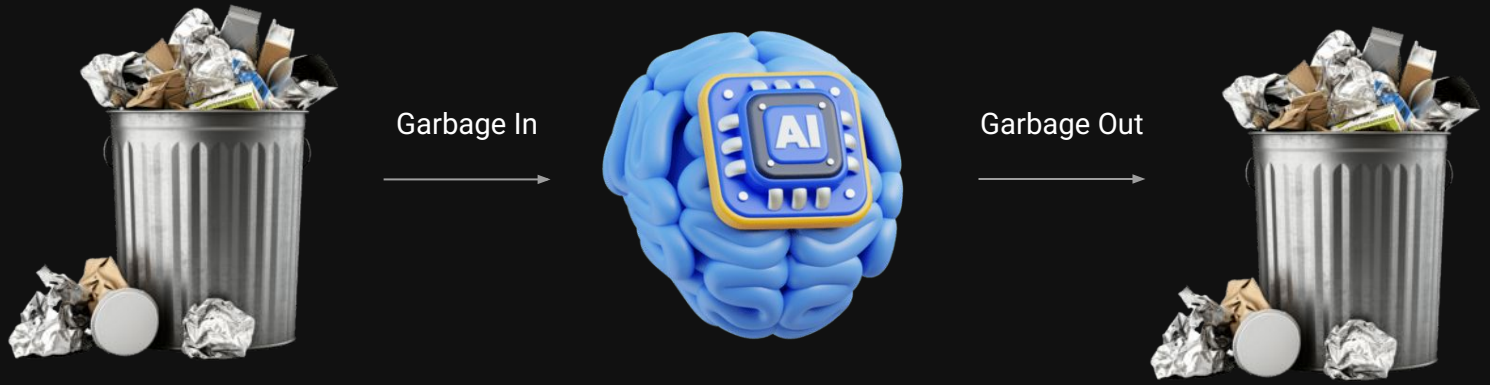
OpenTelemetry

Your environment
(k8s, cloud, etc)

How telemetry is produced and collected

Where telemetry lives and how it’s accessed

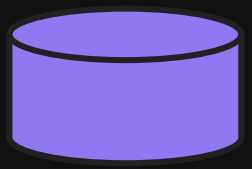
Why AI needs a platform first



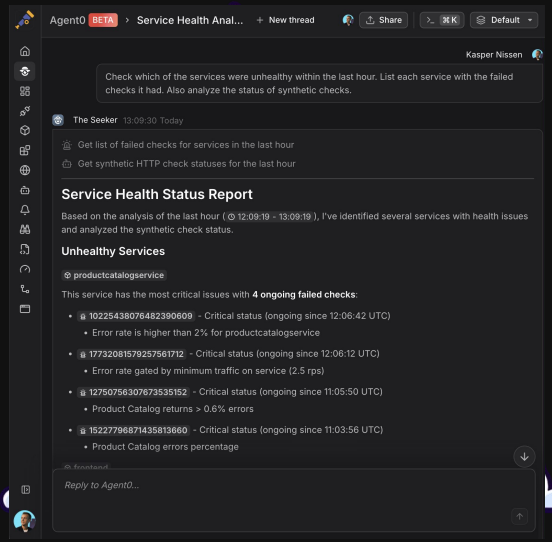
Why AI needs a platform first

OpenTelemetry

Specifications and
Semantic Conventions

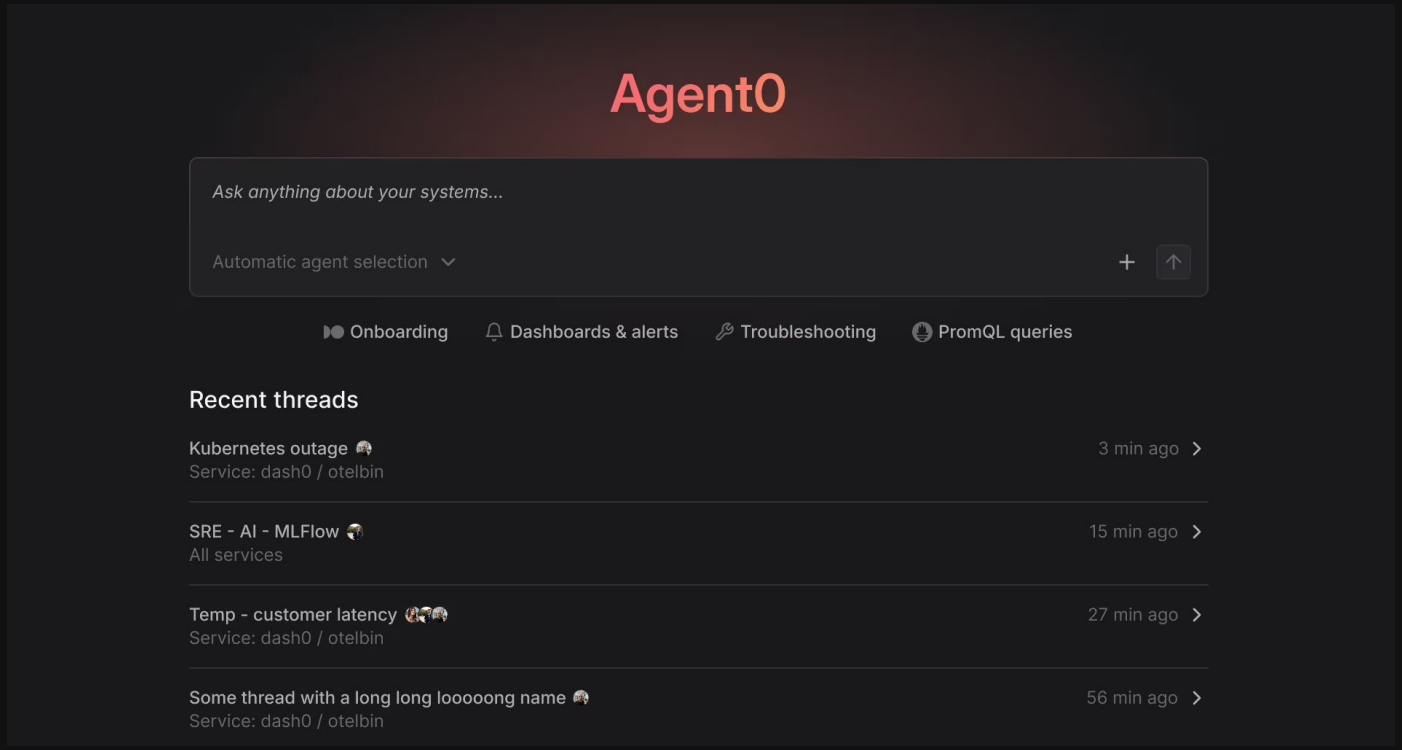


Your Telemetry



dash0

The future interaction model



Demo

So why, OpenTelemetry?

Instrument once, use everywhere

Separate telemetry generation from analysis

Make software observable by default

Improve how we use telemetry

Why Observability as Platform Product?

- Reduce cognitive load
- Correlation by default
- Structure, standardized telemetry
- Foundation for AI-driven workflows

Developers can focus on delivering business value, with observability as a built-in safety net.

dash0

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

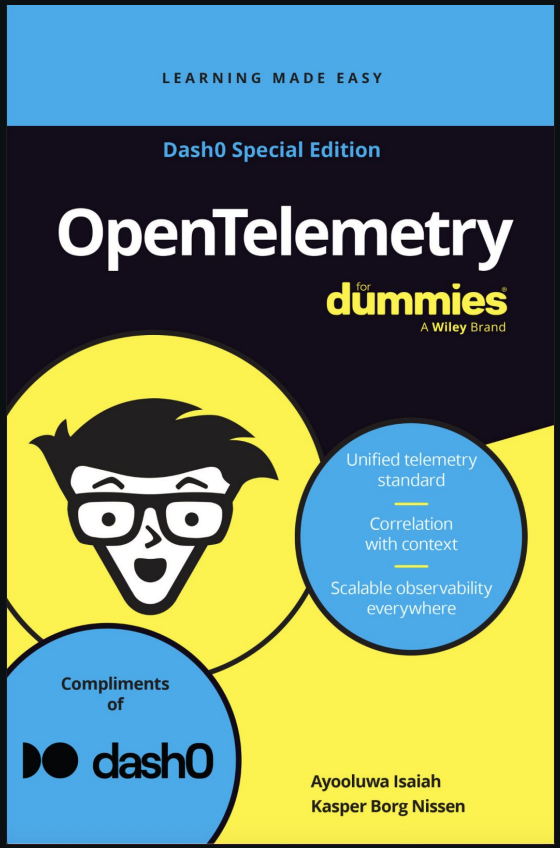
Head of Product @ Dash0

Kasper Borg Nissen

Developer Relations @ Dash0

Book

Get a free copy



Thank you!

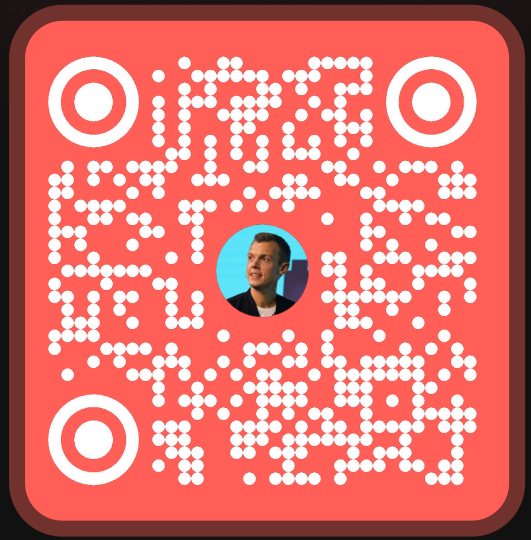
Kasper Borg Nissen, Principal Developer Advocate at Dash0

kaspernissen.xyz /in/kaspernissen kaspernissen

dash0

Tech Hub Aarhus Day

Get in touch!



kaspernissen.xyz /in/kaspernissen kaspernissen