dash0

Cloud Native Dublin

Rethinking Observability as a Platform Product

Kasper Borg Nissen, Director of Developer Relations at Dash0


kaspernissen.xyz


/in/kaspernissen


kaspernissen

Who?

Director of Developer Relations at Dash0

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

Author of OpenTelemetry for Dummies

CNCF, AAIF, and MergeForward Ambassador

Golden Kubestronaut

CNCG Aarhus, KCD Denmark Organizer

Co-founder & Community Lead Cloud Native Nordics

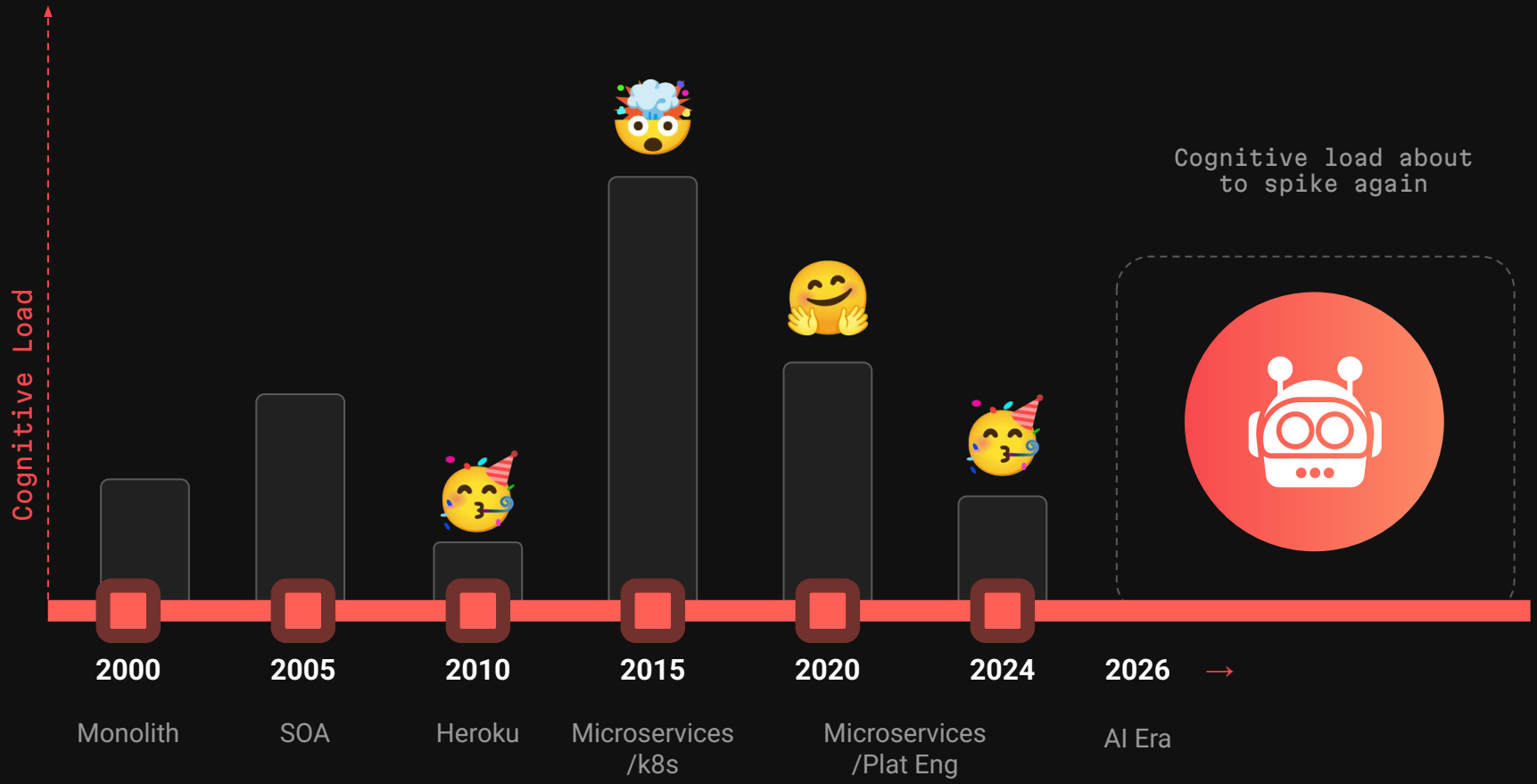
Let’s talk about Kubernetes

Kubernetes is a platform for building platforms

Bryan Liles

KubeCon+CloudNativeCon in San Diego 2019

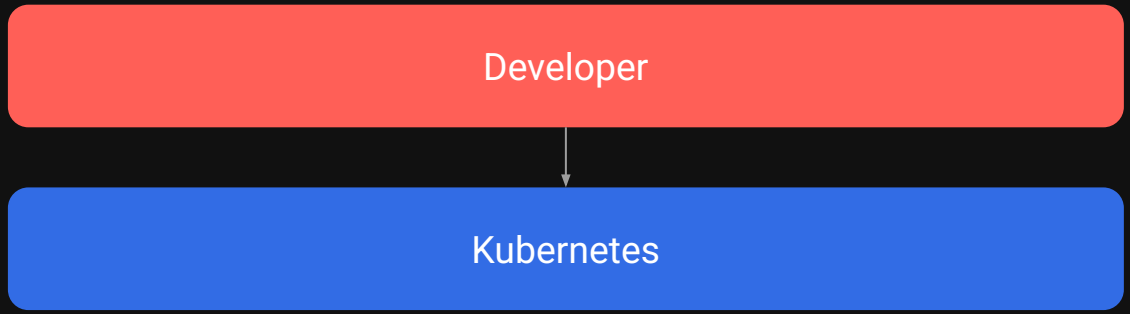
The Rails Moment



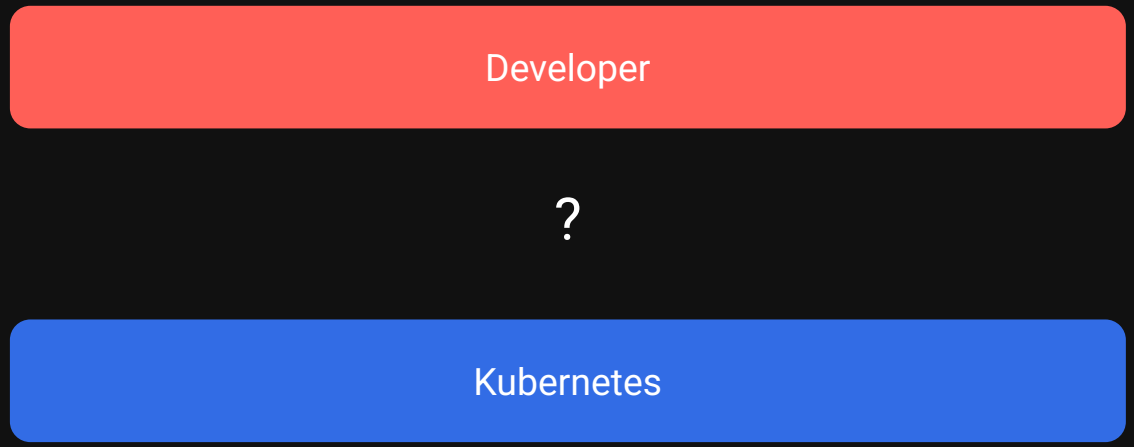
dash0

Visualization Inspired by Daniel Bryant's talk

Why Platform Engineering Emerged



Why Platform Engineering Emerged



Why Platform Engineering Emerged

Developer

Platform

Kubernetes

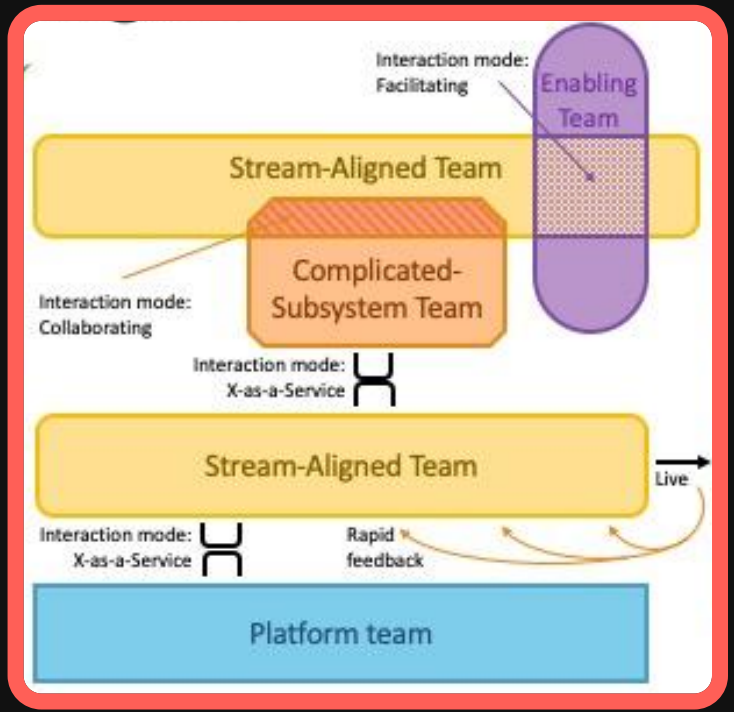
A digital platform is a foundation of self-service APIs, tools, services, knowledge and support arranged as a compelling internal product.

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.

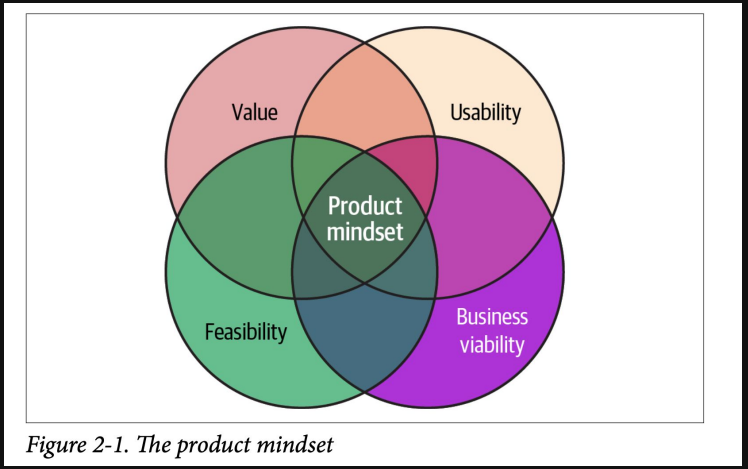
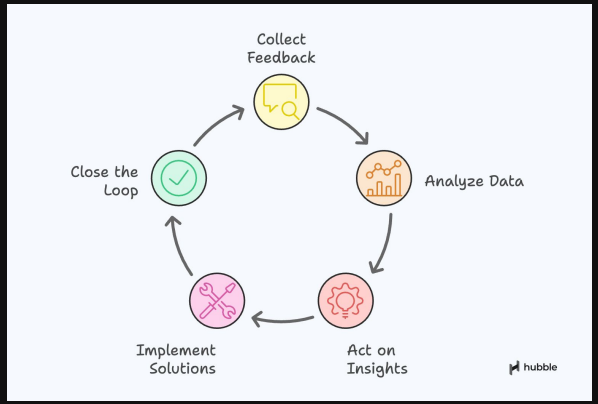
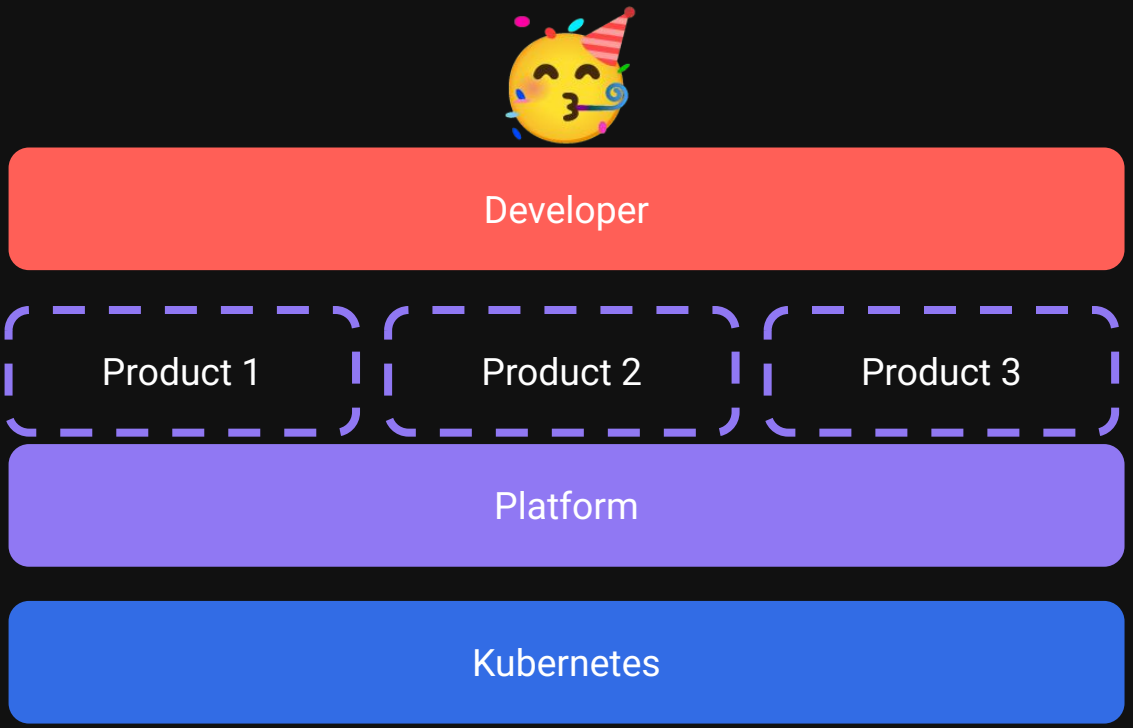


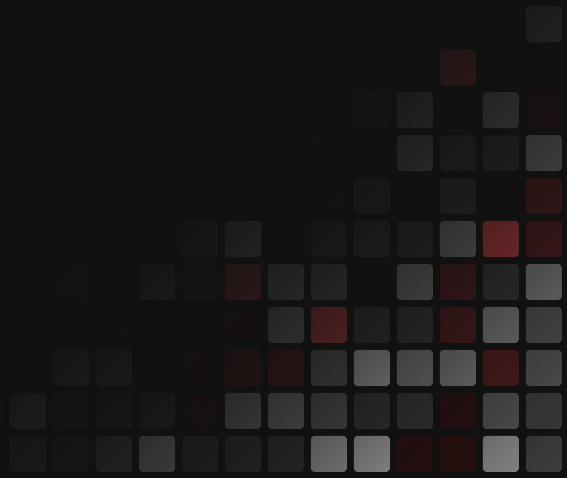
Figure 2-1. The product mindset



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?

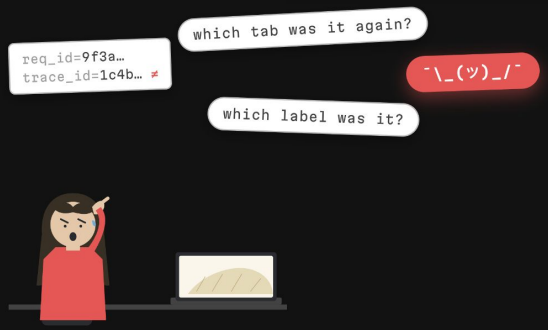
Why is this metric spiking?

Human correlation

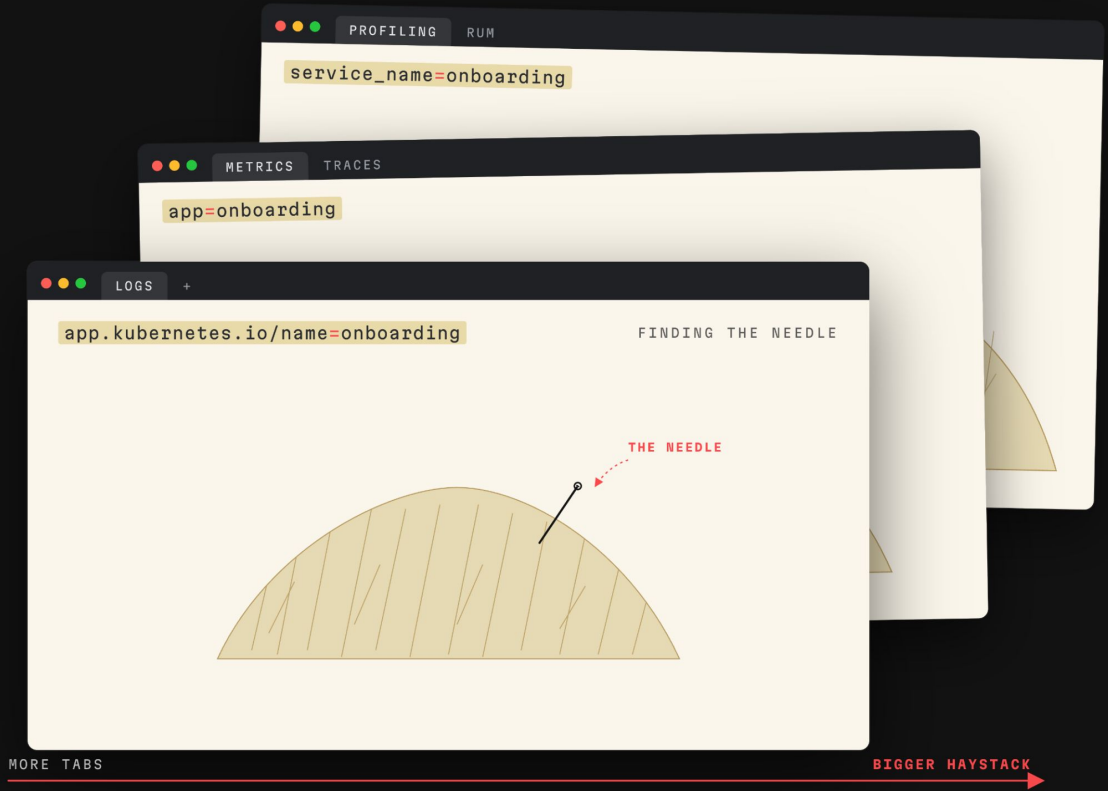
Reality

Scattered signals. Growing haystacks.

Every signal its own tab. Every tab its own query language or filter. And one engineer trying to correlate it all - in their head.



COGNITIVE OVERLOAD



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

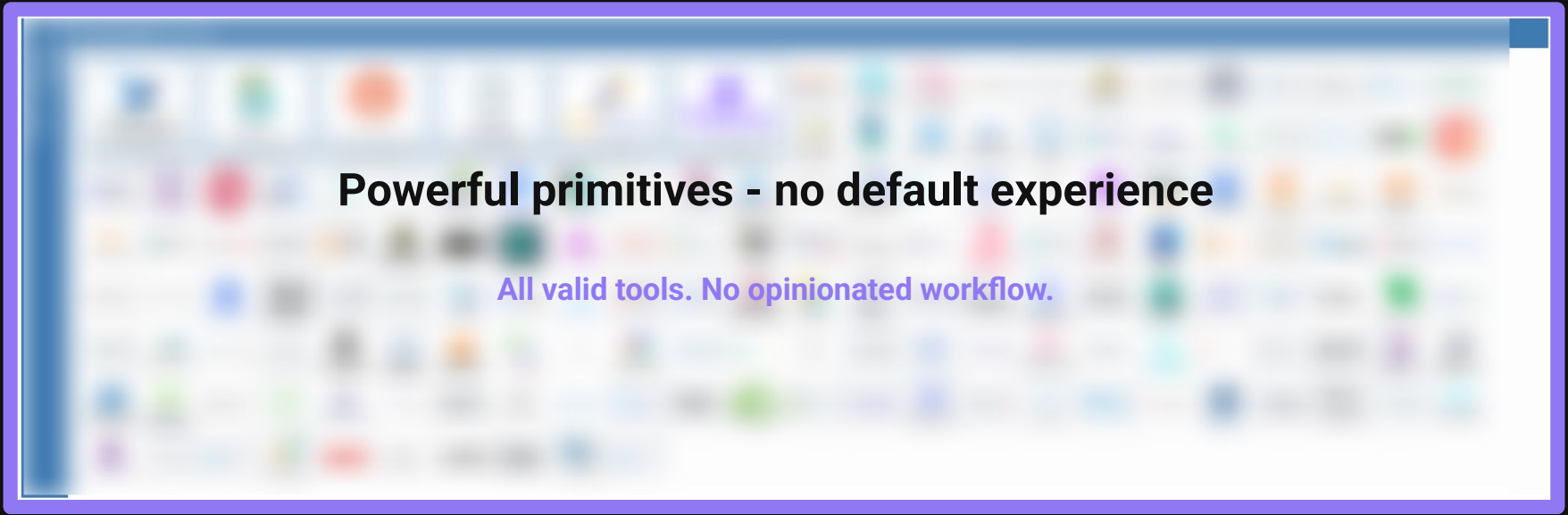
Lack of unified insights

This isn’t necessarily a tooling problem

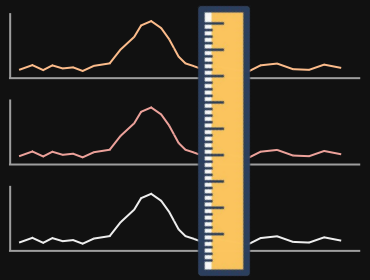
**We don’t have a *metrics* problem,
or a *tracing* problem.
We have *systems* problems.**

Metrics Logs Traces

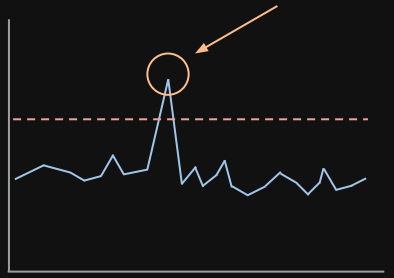
Same mistake as early Kubernetes



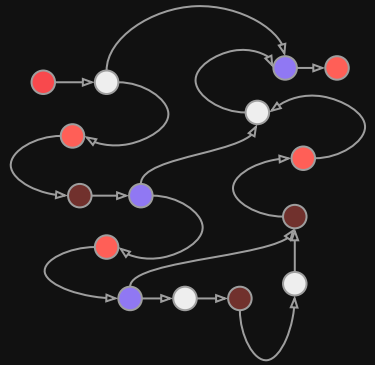
A shift toward correlation



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

OpenTelemetry

CLOUD NATIVE
COMPUTING FOUNDATION

OpenTelemetry in a nutshell

OpenTelemetry is a toolkit and a specification.

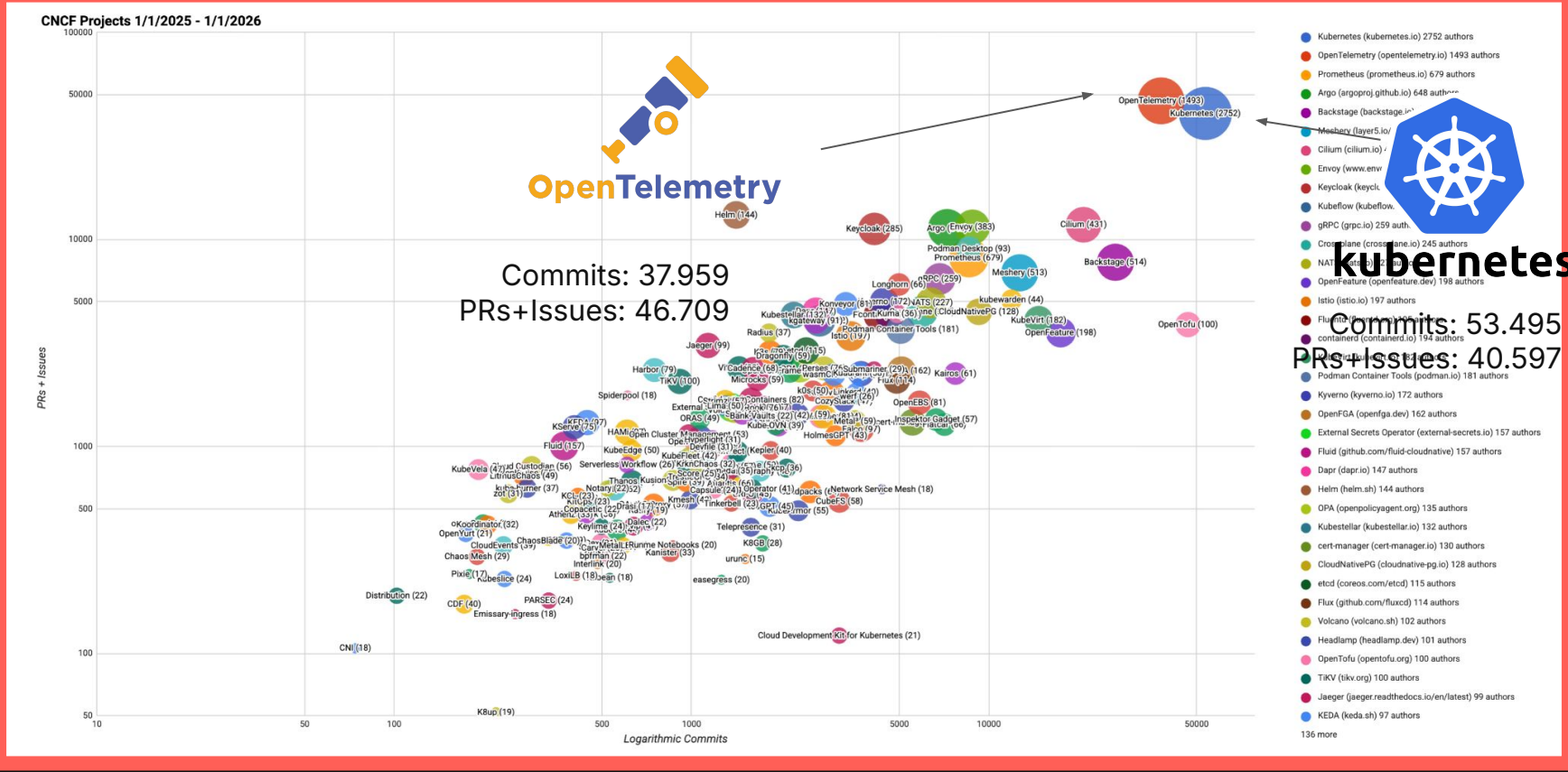
What it is

- 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.

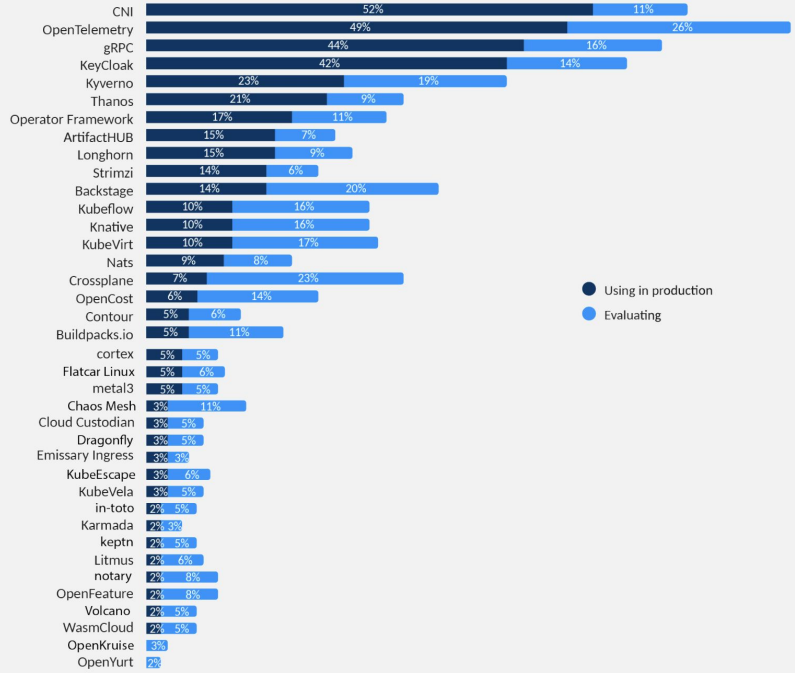
26%

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)

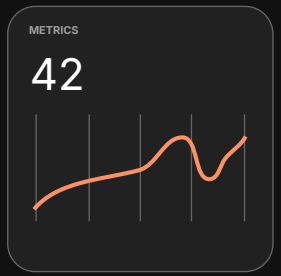


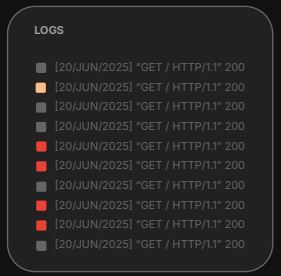
2025 CNCF Annual Survey, Q33, Sample size = 395-502, DKNS excluded

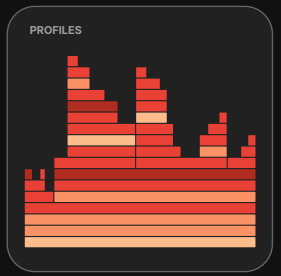
dash0

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

Signals

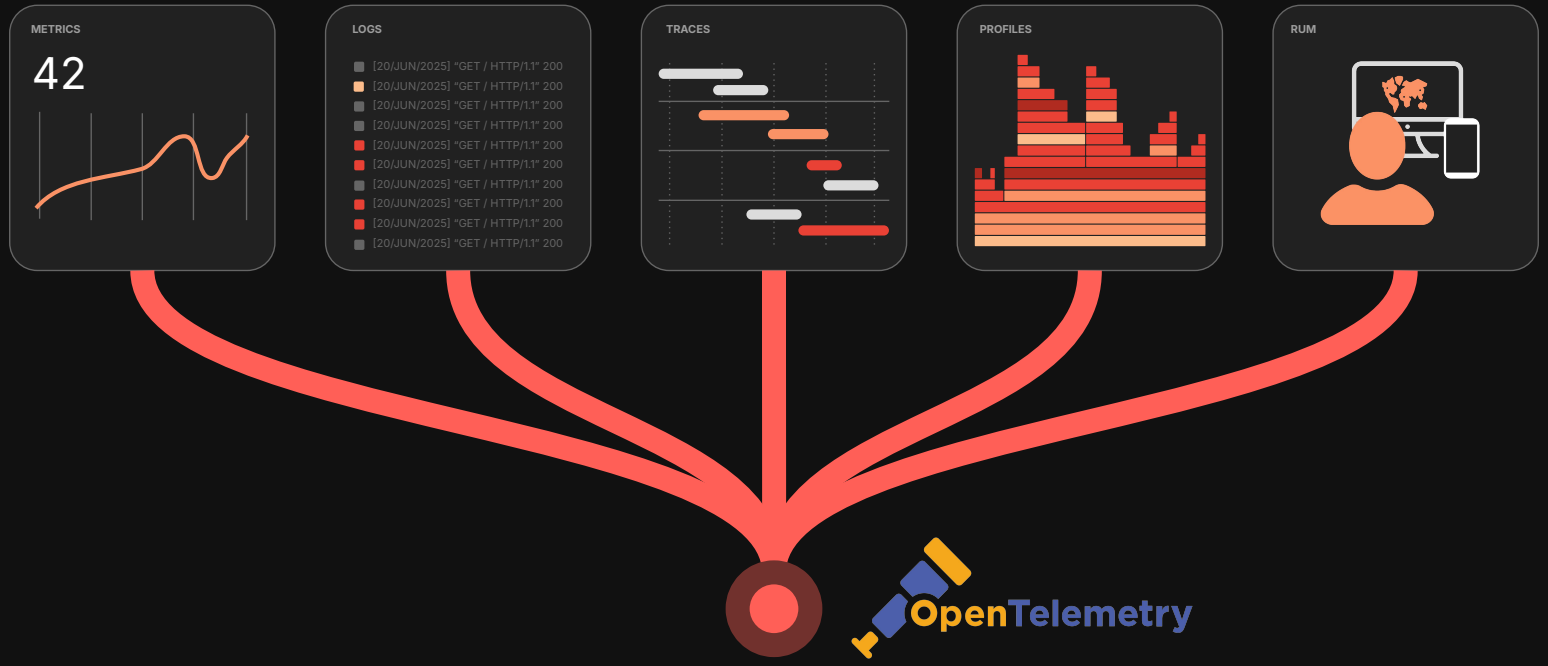




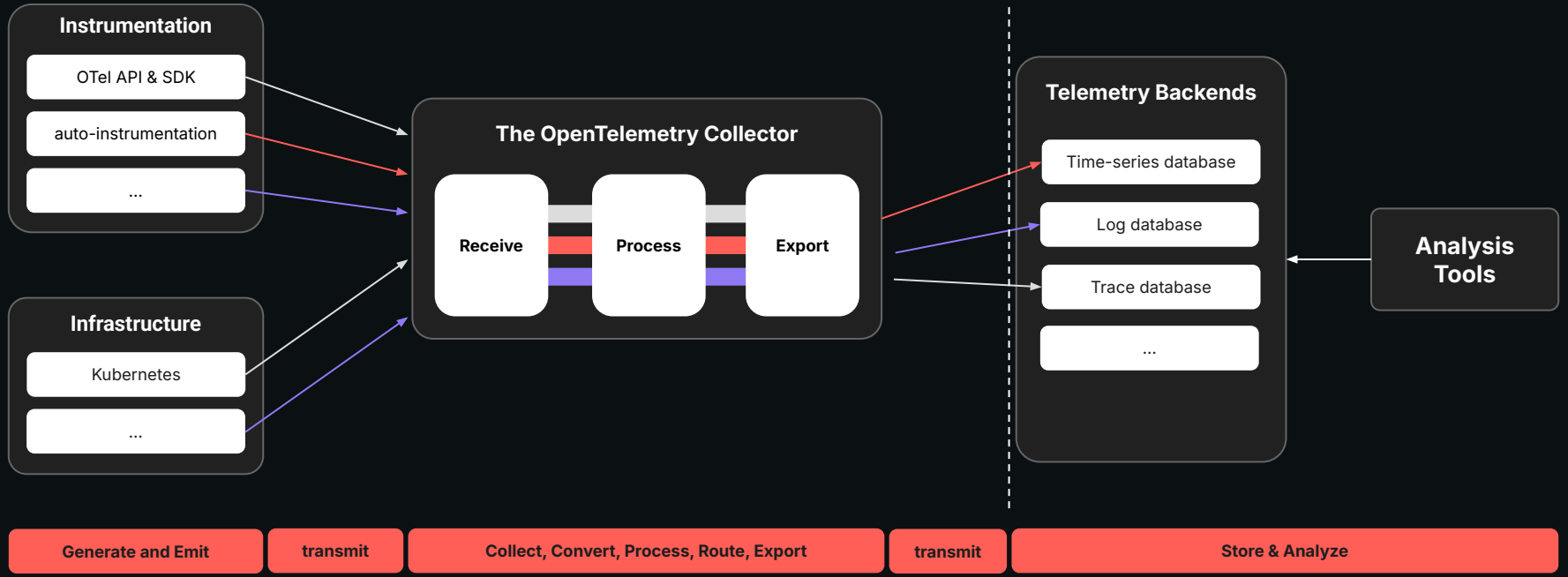


dash0

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

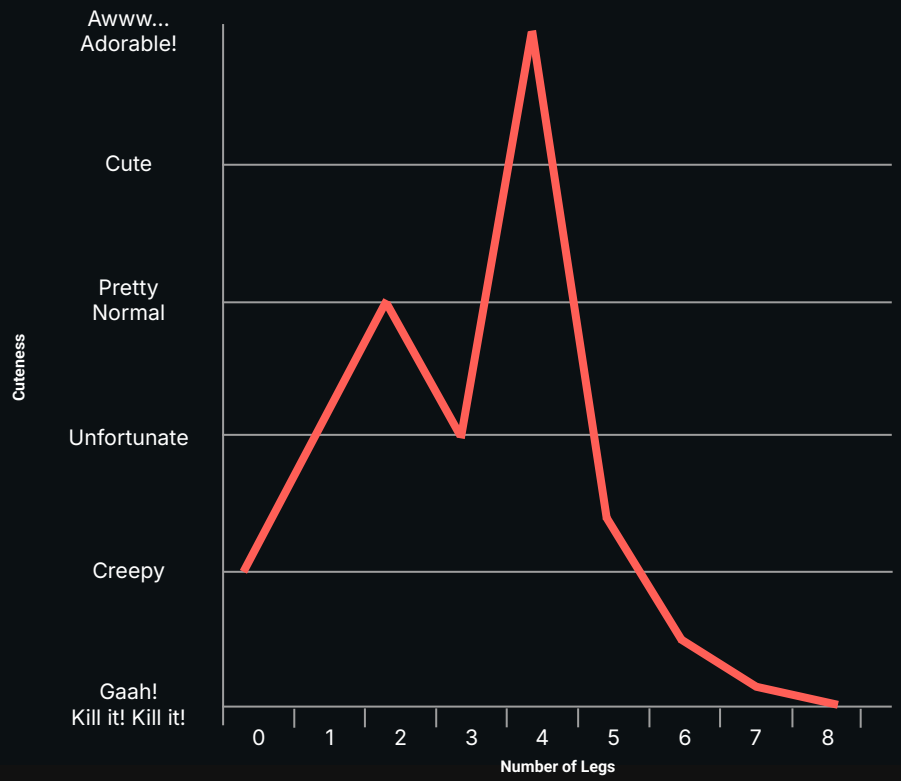
... and many more.

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

Paved Paths for Observability

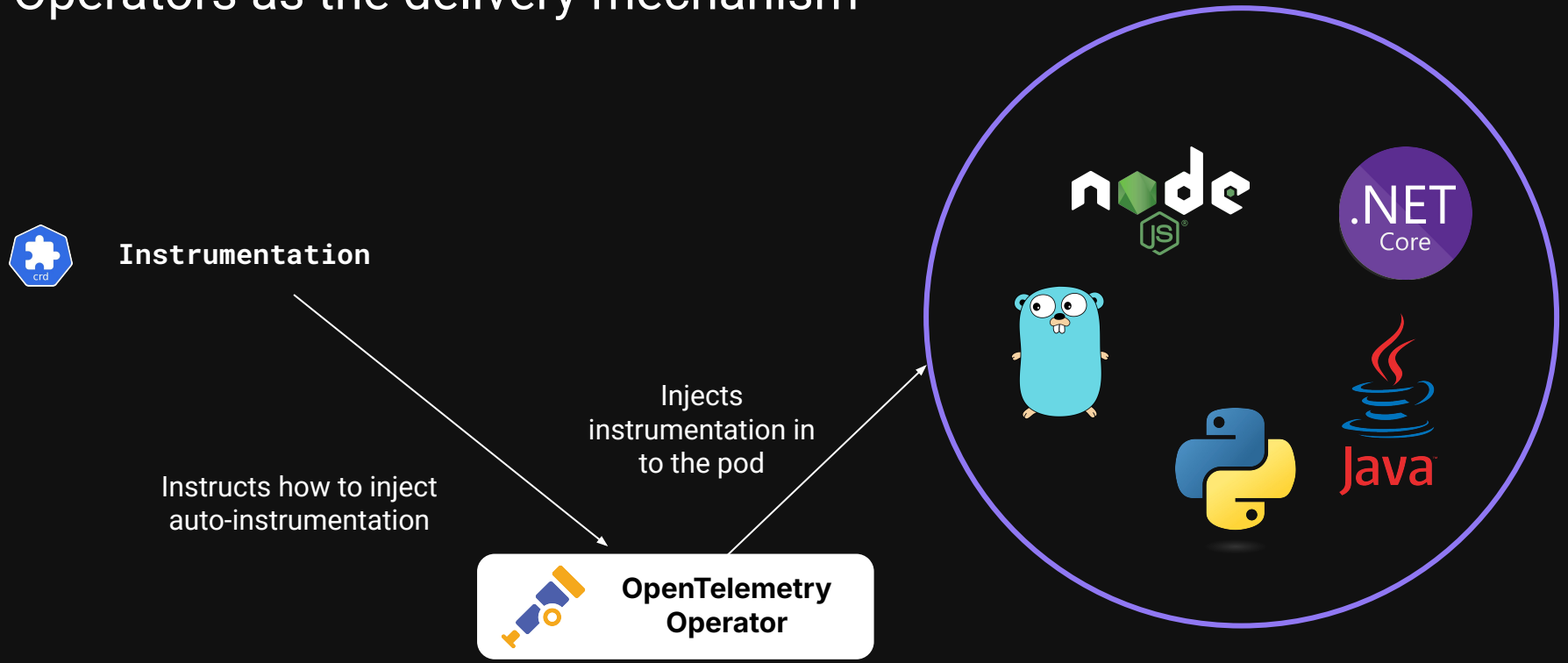
Observability as a Platform Product

Paved Path for Observability

Developer adds business logic.
Everything else is platform.

- Auto-instrumentation
- OpenTelemetry Collector
- Semantic Conventions
- Logs, Metrics, Traces
- Continuous Profiling
- Dashboards-as-code
- Alerting & SLO's
- Correlation Engine
- Cost & Cardinality controls

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

AI increases the need for platform thinking

GARBAGE IN

Inconsistent, fragmented, uncorrelated telemetry

AI/LLM

Doesn't invent clarity. Doesn't reason. Just summarizes.

GARBAGE OUT

Automated confusion. Faster. At scale.

Structure it to the **Semantic Conventions**.

Conventions are what turn telemetry into something an LLM understands.

OpenTelemetry Semantic Conventions
THE SHARED VOCABULARY

Your telemetry
CONFORMING TO SEMCONV

YOUR DATA

Conform to them, and the magic is real.

AI/LLM

Now it has structure to reason over.

UNICORNS AND RAINBOWS

Correlation. Root cause. Answers you can trust.

Without conventions, correlation fails.
Without correlation, AI guesses.
With structure and context, AI reasons.

dash0

AI SRE - an agent that **uses the tooling you already run**.

It doesn't get a private stack. It reads your telemetry, asks the questions, proposes the fix.

READS

Your telemetry

Spans, metrics, logs - conforming to SemConv.

→

ASKS

"Why did it happen?"

→

PROPOSES

The fix

A pull request, into your existing review gates

So how much do you trust it? That depends on the level you're on.

It allows you start asking questions about your system

```
Any issues with my system today?
```

```
Why is service X failing?
```

```
Help me understand this trace with 1500 spans
```

But this is just the beginning....

Six levels of AI autonomy

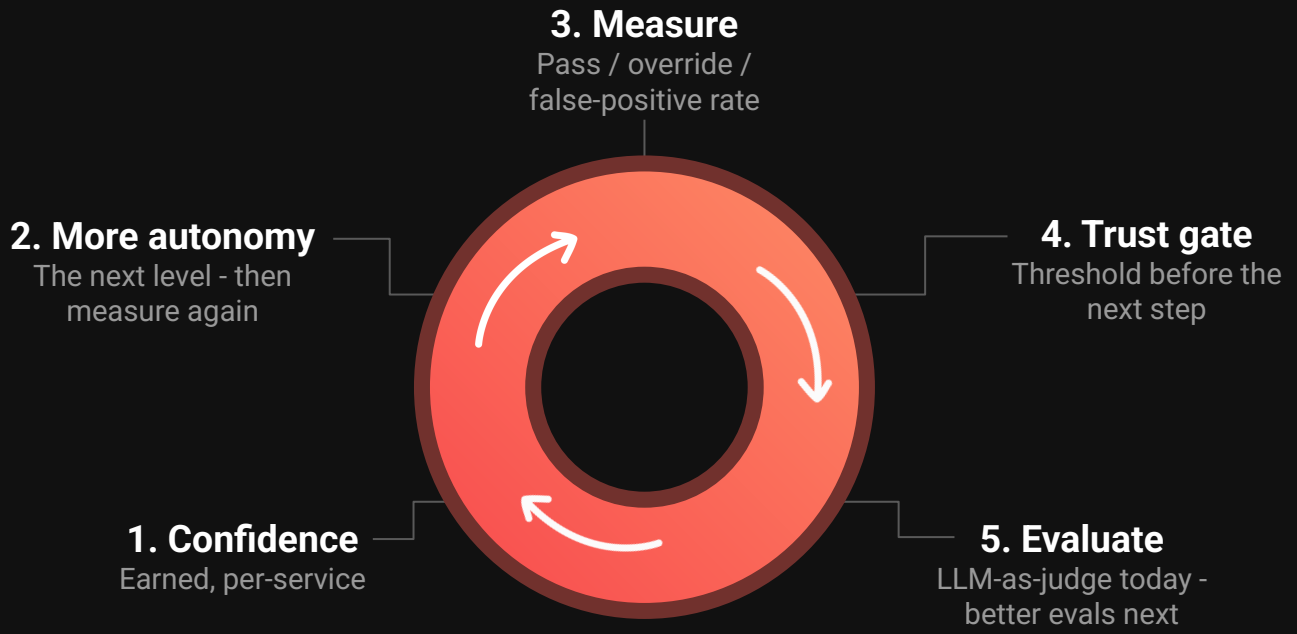
		What AI do	What humans do
L1	AI-Assisted	Autocompletes lines, suggests snippets.	Write the code. Decide on each suggestion as it appears.
L2	AI-Generated, Human-Reviewed	Writes whole functions, files, or PRs.	Drive the work. Review and approve every PR before merge.
L3	AI-Generated, Auto-Reviewed	Generates code from a spec. Runs tests, linters, security scans, holdout scenarios automatically.	Approve merges. Look at intent and proof (satisfaction reports, screenshots, behavior evidence), not diffs.
L3.5	Selected Auto-Merge	Generates code. Routine PRs in qualifying services auto-merge.	Veto rights on auto-merged PRs. Active review only on high-risk services and out-of-policy changes.
L4	Mostly Autonomous	Handles the full loop within a defined scope. Notifies humans, does not ask them.	Set goals and scope. Handle escalations on boundary violations.
L5	Dark Factory	End-to-end SDLC: spec, code, test, review, deploy, maintain.	Write specs. Define system boundaries and quality thresholds. Improve the factory when novel failures emerge.

dash0

Source: https://www.dash0.com/knowledge/the-six-levels-of-agentic-software-engineering

You don’t jump levels. **You earn them**.

How autonomy is actually earned.



Expect a dip first - the “AI tuition tax.” Measure it with DORA, and it pays back.

dash0

Reason. Act. **Observe**. Decide

REACH IT FROM ANYWHERE

Chat | CLI | MCP | Slack | Issue tracker

Every surface calls the same tools. UI-only features are invisible to the agent.

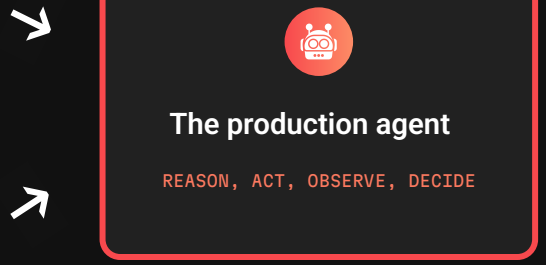
REASONS OVER YOUR CONTEXT

Facts - what it knows

Telemetry | Docs | Memory | Code

Behaviors - how it acts

Skills | Runbooks

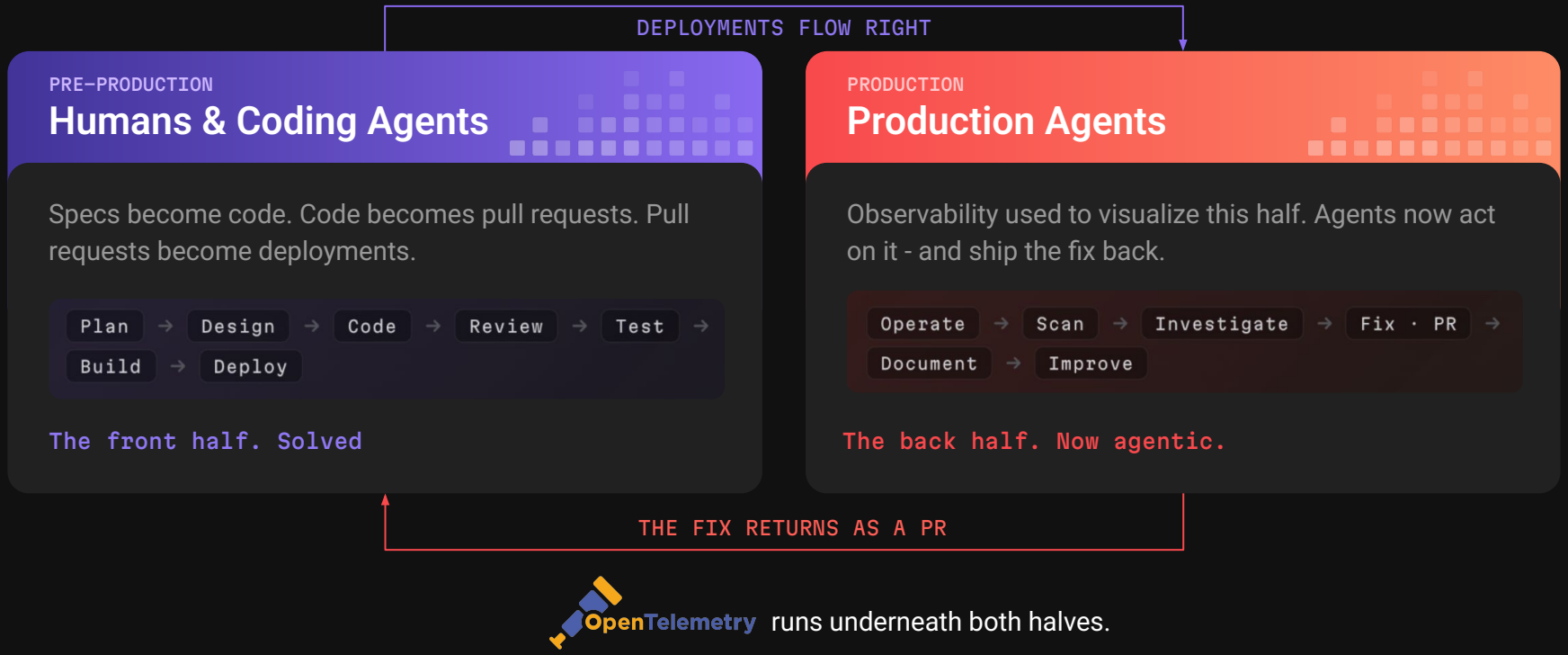


The model is the reasoning engine.
The harness around it is the engineering

PRODUCES

- Dashboards
- Alerts
- Runbooks
- Pull Requests
- … anything over MCP

Agentic SDLC



OpenTelemetry runs underneath both halves.

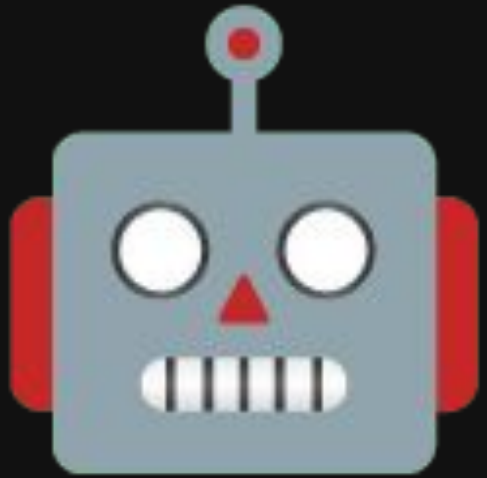
Why Observability as Platform Product?

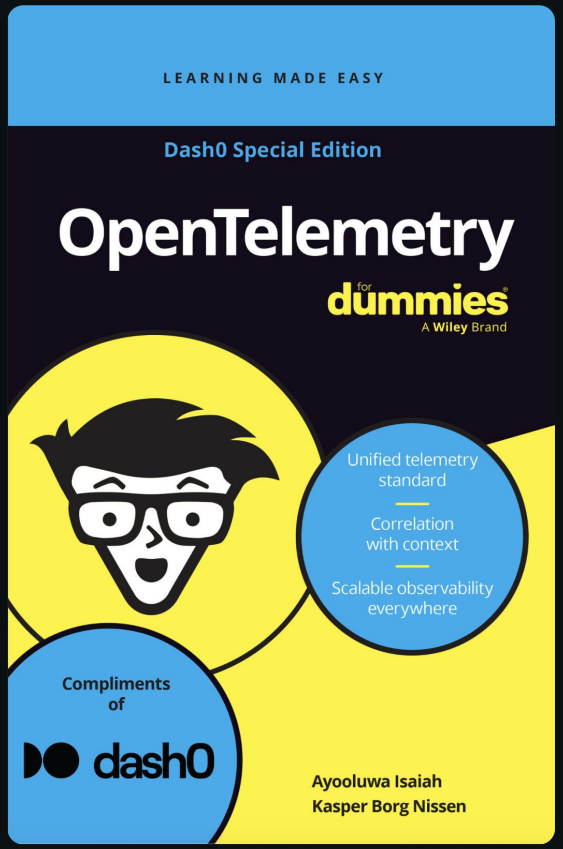
- Reduce cognitive load
- Correlation by default
- Structure, standardized telemetry
- Foundation for AI-driven workflows

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

Start building this into your platform now.

AI moves at an incredible speed, if you don’t have the foundation, you will fall behind.




LEARNING MADE EASY
Dash0 Special Edition
OpenTelemetry
for dummies
A Wiley Brand
Unified telemetry standard
Correlation with context
Scalable observability everywhere
Compliments of
dash0
Ayooluwa Isaiah
Kasper Borg Nissen

MERGE FORWARD

CLOUD NATIVE COMPUTING FOUNDATION

Follow us on LinkedIn

Merge Forward (CNCF)

Join Merge Forward!

Building a stronger open source future together!

#merge-forward on Slack!

Learn more

community.cncf.io/merge-forward

dash0

Cloud Native Dublin

Thank you!

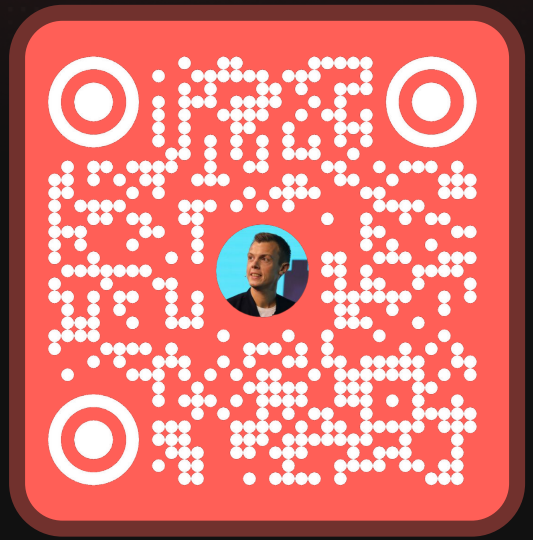
Kasper Borg Nissen, Director of Developer Relations at Dash0

kaspernissen.xyz /in/kaspernissen kaspernissen

dash0

Cloud Native Dublin

Get in touch!



kaspernissen.xyz /in/kaspernissen kaspernissen