dash0

# Rethinking Observability as a Platform Product

Slides

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, Cloud Native Denmark Organizer (former KCD)

Co-founder & Community Lead Cloud Native Nordics



AMBASSADOR

AAIF AMBASSADOR

GOLDEN Kubestronaut

MERGE FORWARD

01

PART ONE

# Something you already know.



> “Kubernetes is a vehicle taking us to our destination, and not the destination itself... Too many times we focus on Kubernetes as a product. **It is not a product. It is a car.** „



Bryan Liles
KubeCon + CloudNativeCon NA, San Diego 2019


OpenTelemetry

> “ OpenTelemetry is a vehicle taking us to our destination, and not the destination itself... Too many times we focus on OpenTelemetry as a product. **It is not a product. It is a car with a full tank.** „



Kasper Borg Nissen
Today

# Kubernetes needs to find it’s Rails moment...



# “Convention over configuration” is three promises, not one slogan.

01

Someone opinionated picks the defaults

02

The defaults are easy to override

03

Overriding one thing does not break the rest

> How do we capture the essence of Rails without diluting the power of Kubernetes?



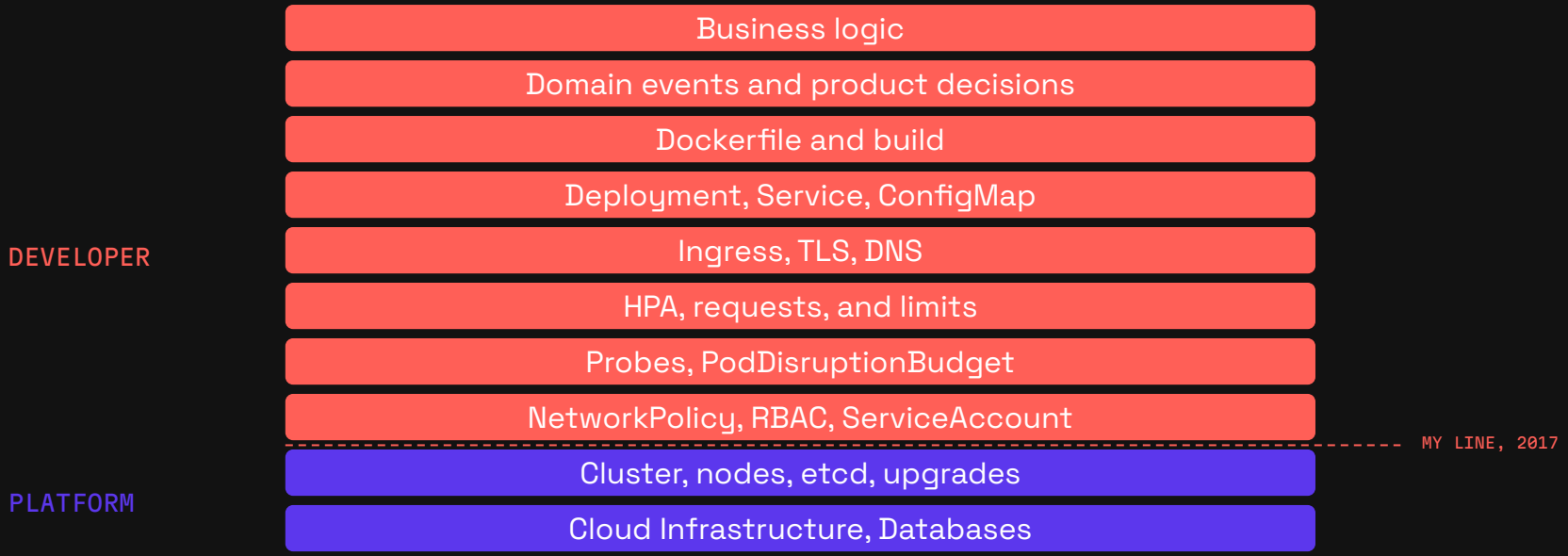
Bryan Liles
KubeCon + CloudNativeCon NA, San Diego 2019

02

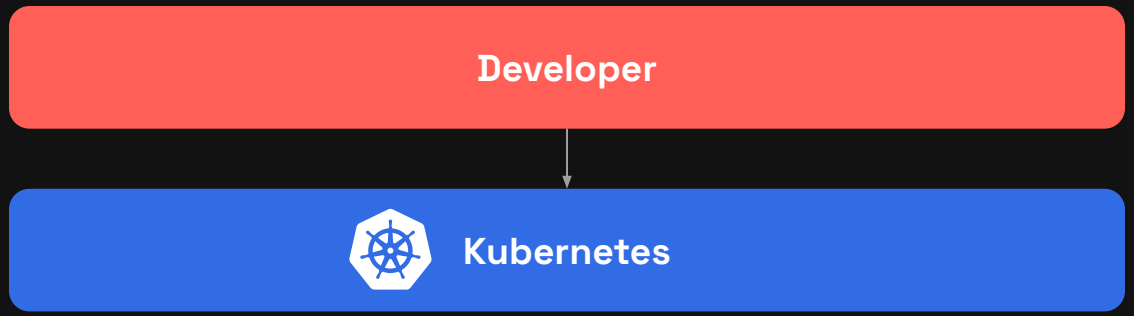
PART TWO

# We drew the line in the wrong place.

# I stood on stage in 2017



Is it kube-c-t-l kube-cuddle, kube-control? I don't care!



Developer

?



Kubernetes



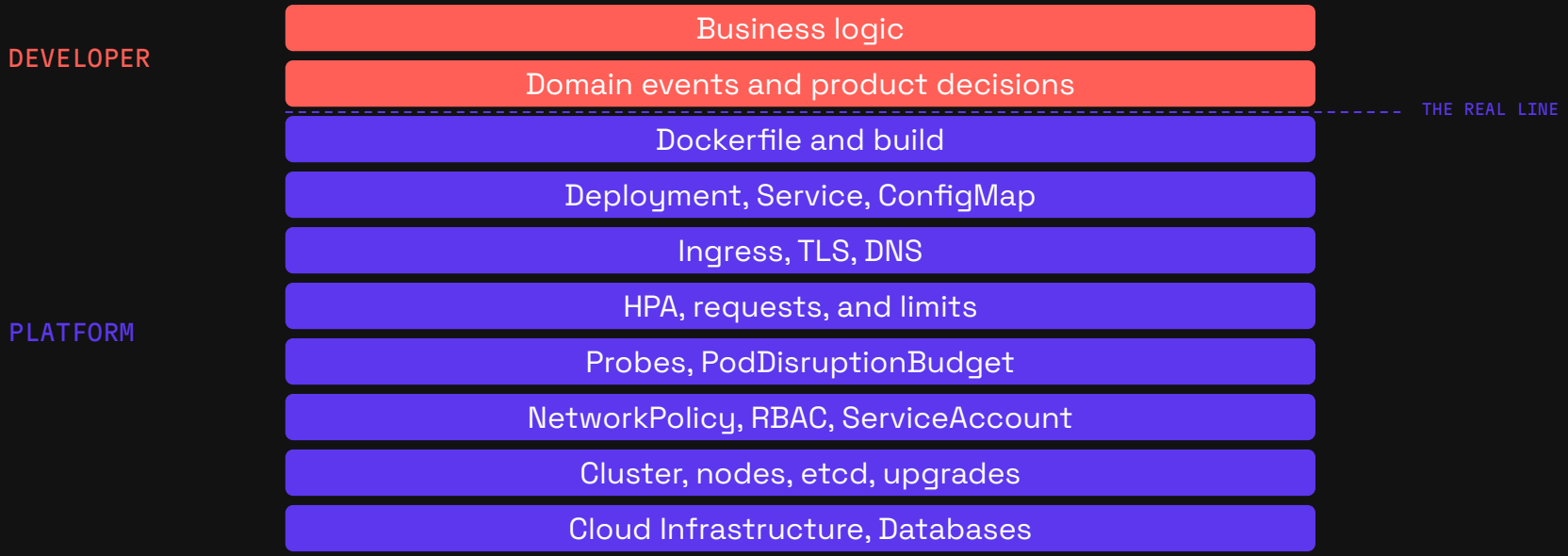
Developer

Platform

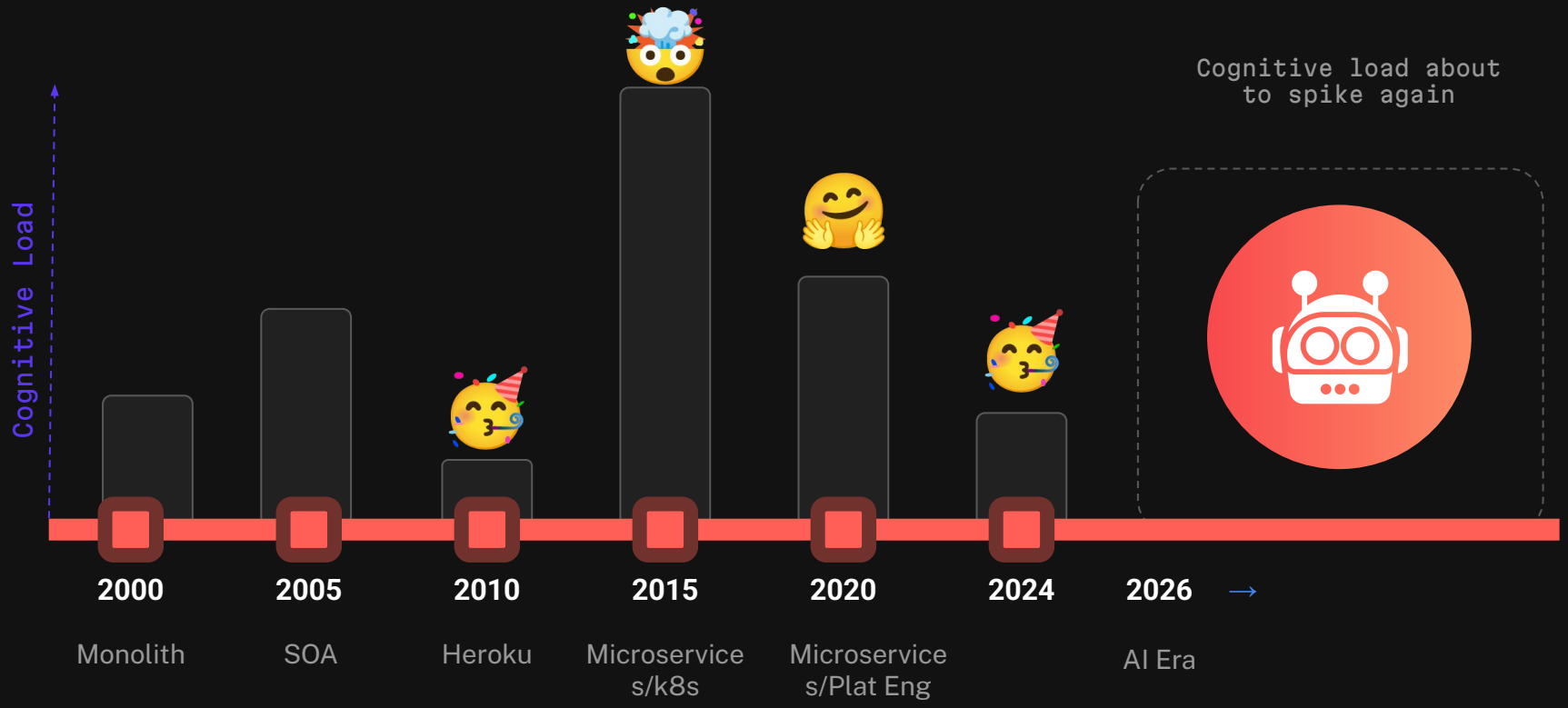


Kubernetes

# That line was wrong!



# Every shift traded one kind of load for another




Visualization Inspired by Daniel Bryant's talk

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

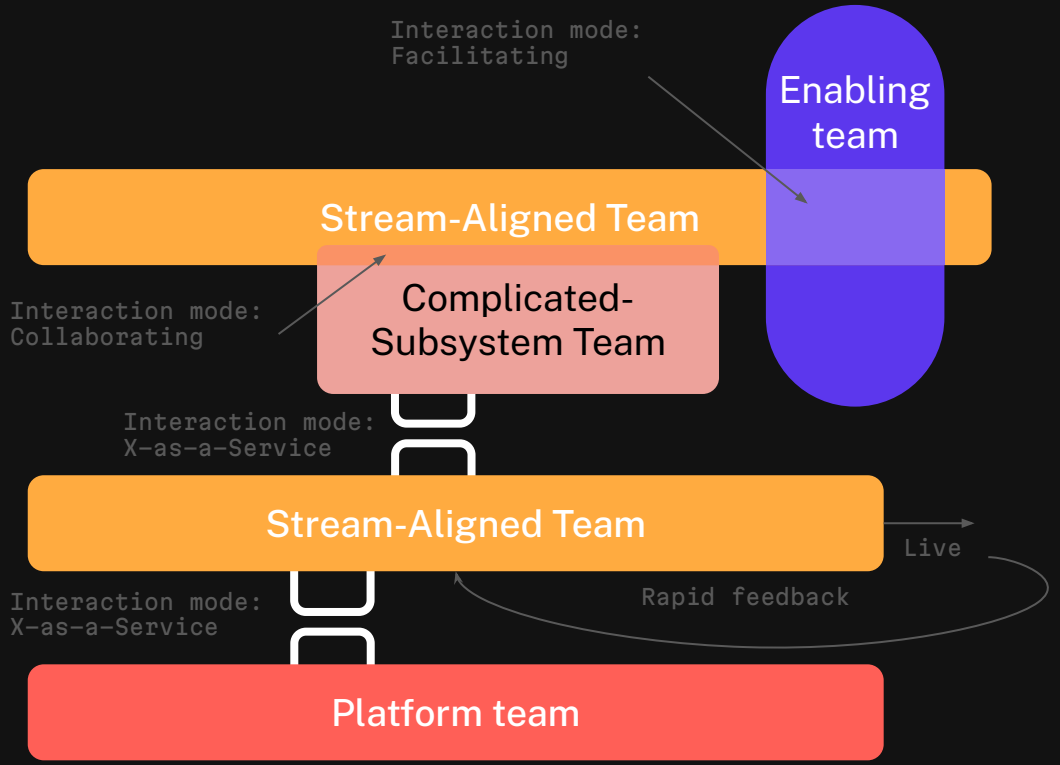


Evan Bottcher

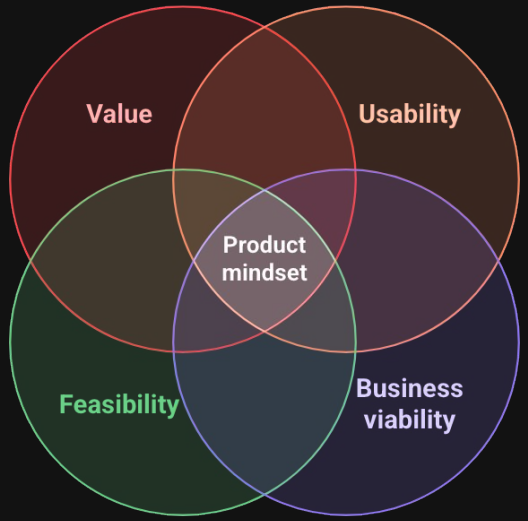
martinfowler.com/articles/talk-about-platforms.html

# Team Topologies

**Platform team:**
a grouping of other team types that provides 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.

# Shift left moved the work

BEFORE – EVERYTHING IS THE DEVELOPER'S

business logic | SDK wiring | sampling config | dashboards | alert rules | attribute naming | collector config | SDK upgrades

↓ SHIFT DOWN ↓

AFTER – THE DEVELOPER

business logic | which business events matter

AFTER – THE PLATFORM, ONCE, FOR EVERYONE

SDK wiring | Sampling config | dashboards | alert rules | attribute naming | collector config | SDK upgrades

Shift left never made this this shorter. It just moved things around.

# Seven years later...

| | | | |
|---|---|---|---|
| **Kubernetes Manifests**<br>“developers write their own” | → | Helm charts, Kustomize bases, operators, CRDs.<br>Nobody types the nginx manifest from scratch. | MOVED DOWN |
| **Dockerfiles**<br>“developers own their build” | → | Buildpacks, ko, Jib, a base image maintained once.<br>The pipeline builds it. Most teams never open one. | MOVED DOWN |
| **Security Policy**<br>“they should just work” | → | Admission control, Pod Security Standards, OPA,<br>Kyverno. Enforced centrally. Nobody hand-writes it. | MOVED DOWN |
| **Network Policy**<br>“you know what they are” | → | A first-class resource, enforced by the CNI.<br>No Java developer has authored one this decade. | MOVED DOWN |
| **Observability**<br>“wiring it up” | ← | More SDKs. More sampling decisions. Own your<br>dashboards. Write your own SLOs. Per team. | MOVED UP |

03

PART THREE

# And then there is observability.

# 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.

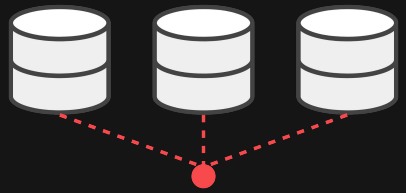
Every signal its own tab. Every tab its own query language or filter. And one engineer trying to correlate it all - in their head.




COGNITIVE OVERLOAD



# What you actually bought



Three stores.
Three query languages.




A bill with
its own slope.



An exit priced in
re-instrumentation.

# 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

# What that costs, measured

**57%**

say their setup produces **too much noise**, or does not help them find root cause

**58%**

name **instrumentation skill gaps** as the primary blocker

**39%**

say adding metrics or traces adds **more than 24 hours** to feature-to-production

Weave Intelligence, Market Guide for Observability Trends in Platform Engineering, 2026 · n = 105

# What that costs, measured

> “The platform absorbs the instrumentation burden; developers receive telemetry as a default capability rather than a responsibility. ”

Weave Intelligence, Market Guide for Observability Trends in Platform Engineering, 2026 (p.10)

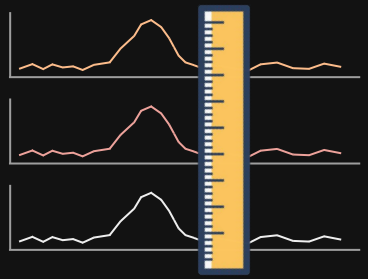


04

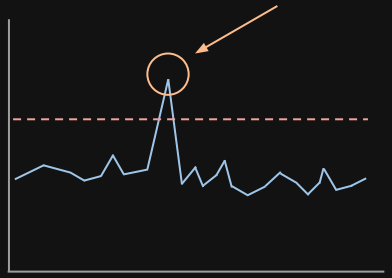
PART FOUR

# A shift is happening.

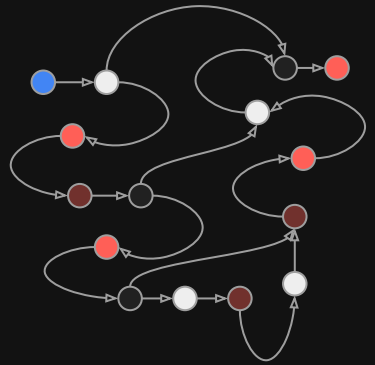
# A shift toward correlation



Find related information



Jump between signals

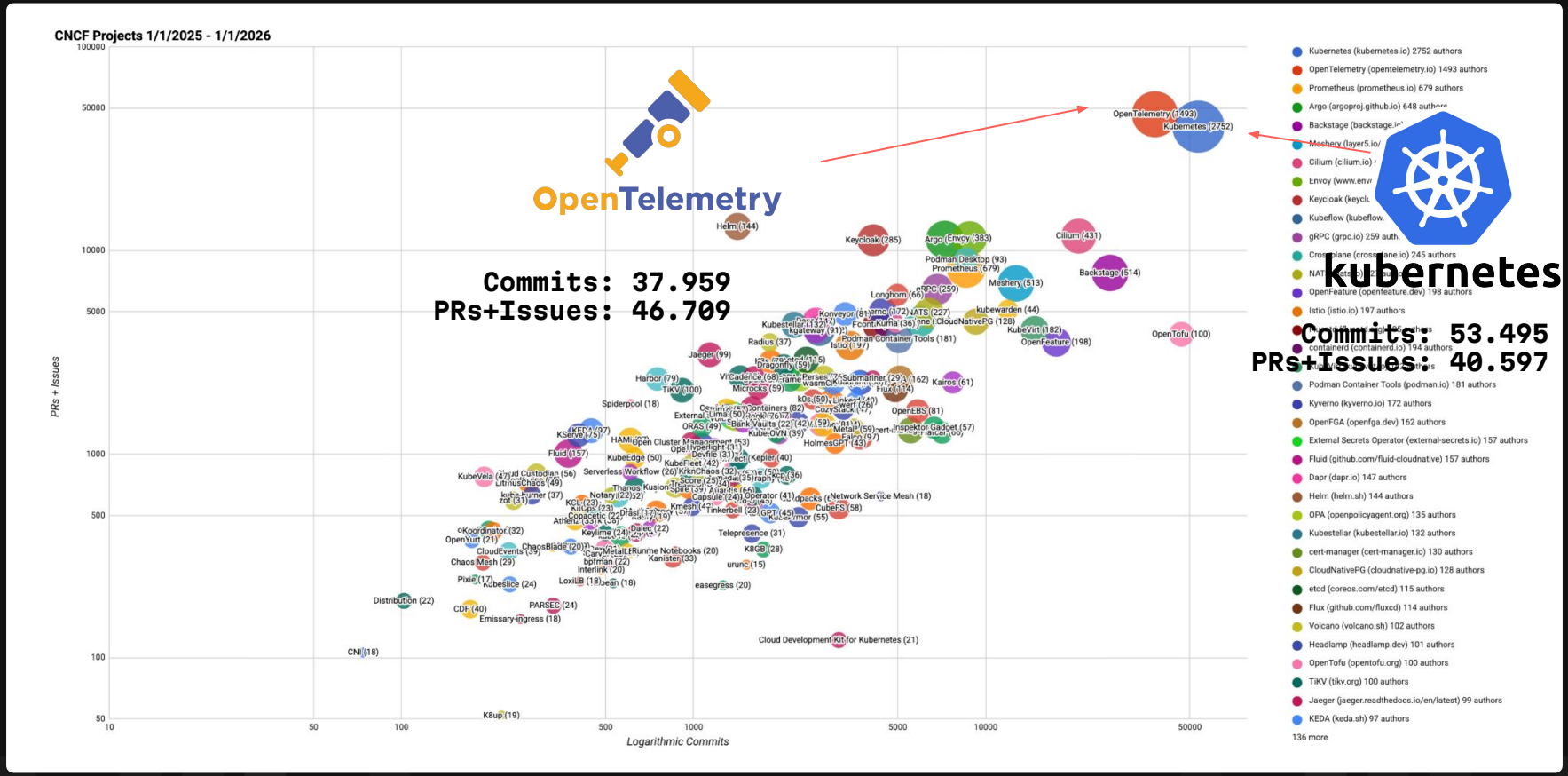


Reconstruct chain of events

# A shift toward...



1/1/2025-1/1/2026



# 49%

of respondents using OpenTelemetry in production.

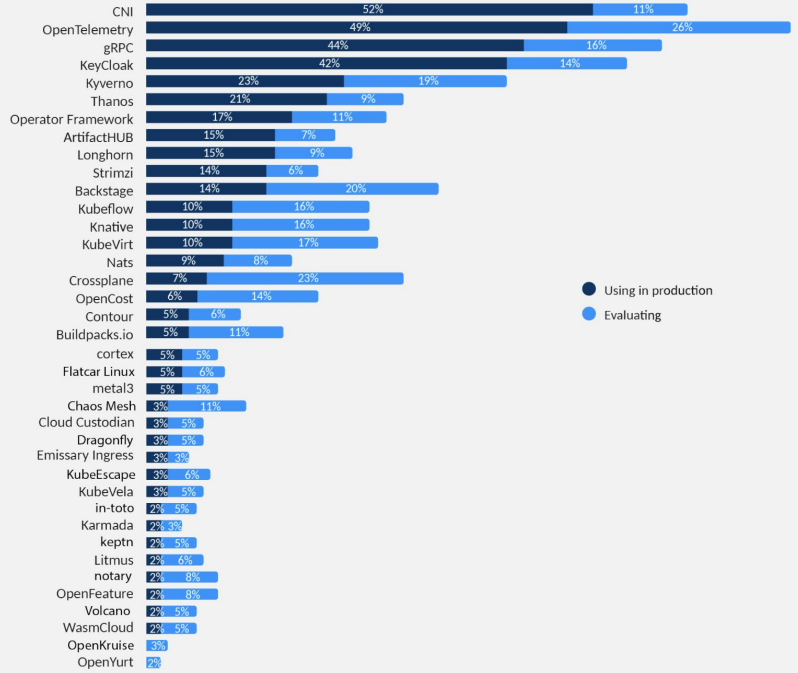
# 26%

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

# OpenTelemetry in a nutshell

WHAT IT IS

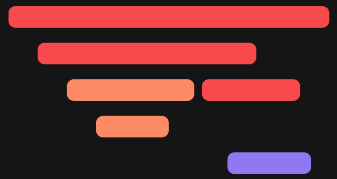
- Data models
- API specifications
- Semantic conventions
- Library implementations in many languages
- Utilities
- and much more

WHAT ITS NOT

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

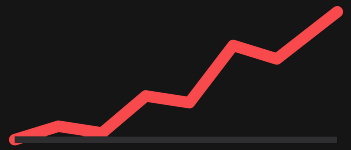
# Four signals. One vocabulary.

## TRACES · STABLE



Where did the time go?

## METRICS · STABLE



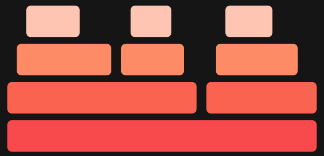
Is it getting worse?

## LOGS · STABLE

```
20/JUN "GET / HTTP/1.1" 200
20/JUN "GET / HTTP/1.1" 200
20/JUN "GET / HTTP/1.1" 200
20/JUN "GET / HTTP/1.1" 500
20/JUN "GET / HTTP/1.1" 200
20/JUN "GET / HTTP/1.1" 200
```

What exactly happened?

## PROFILES · ALPHA



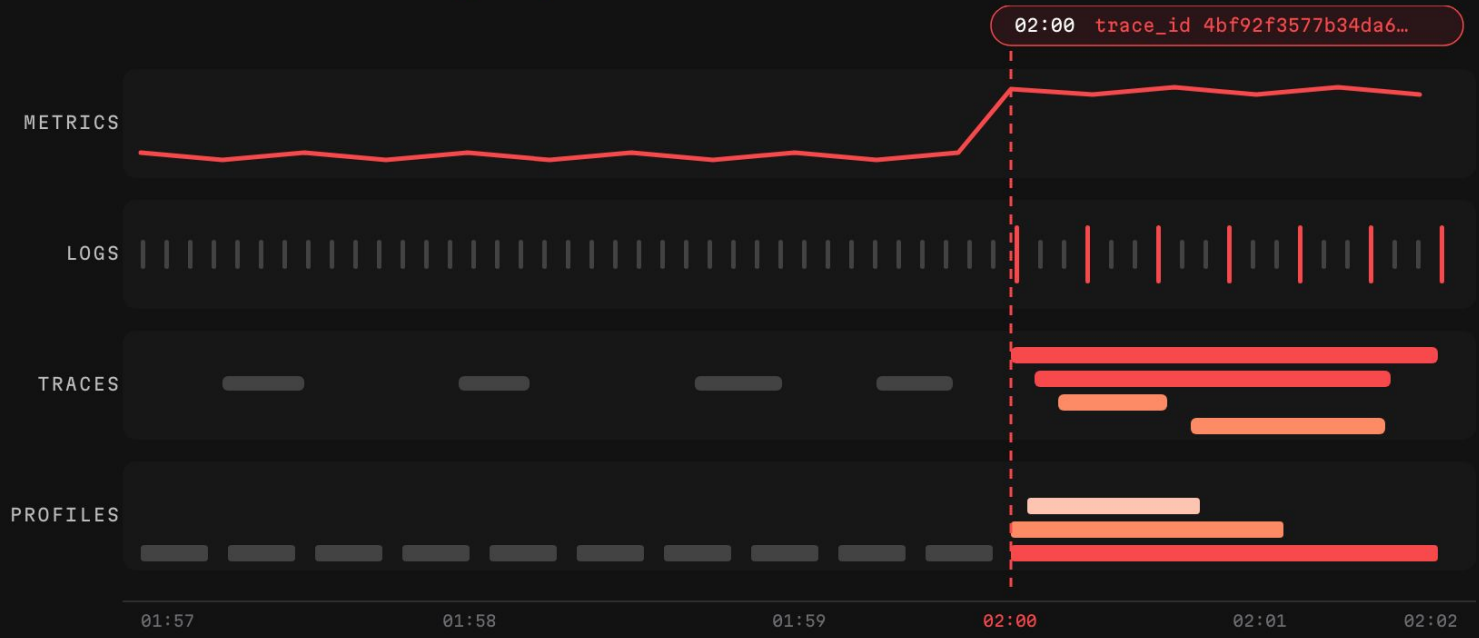
Which code burned CPU?

BUILT ON TOP

RUM · Mobile · Synthetic checks · Business KPIs

Use cases, not signals.
The same four, emitted from somewhere else.

# Correlation is the superpower.




One moment. One trace ID. Four partial views.

# You need the full picture...

# What makes four signals one story




THE SAME RESOURCE ATTRIBUTES ON EVERY SIGNAL

`service.name` · `service.version` · `deployment.environment.name` · `k8s.pod.name` · `cloud.region`

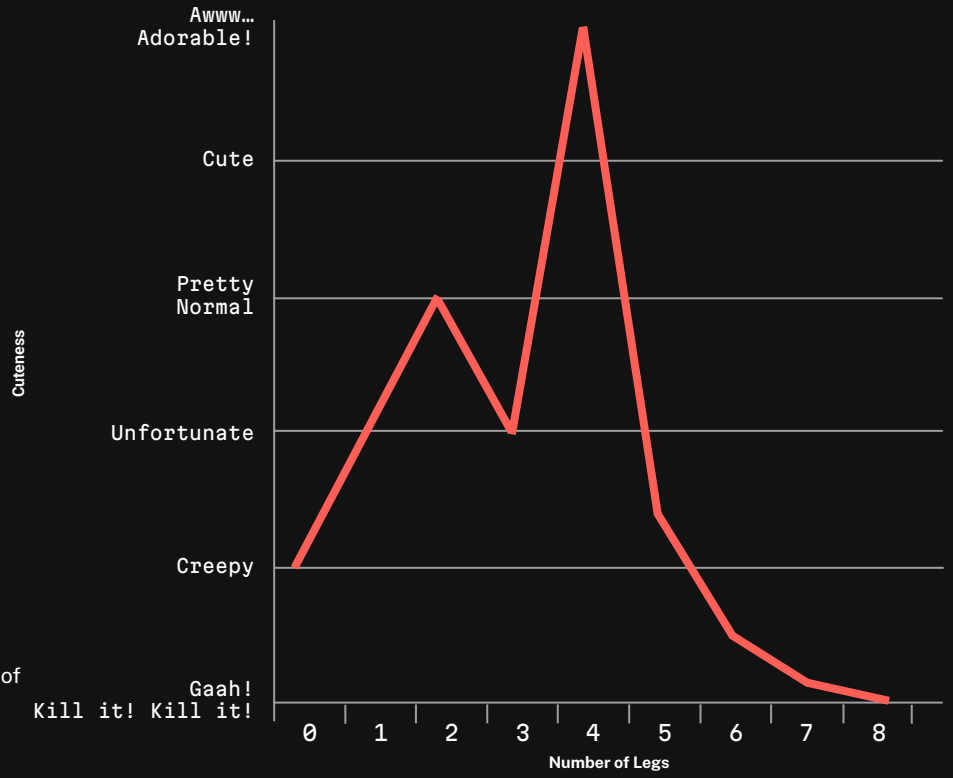
Context that travels, and names that match.

> “Telemetry without context is just data.”



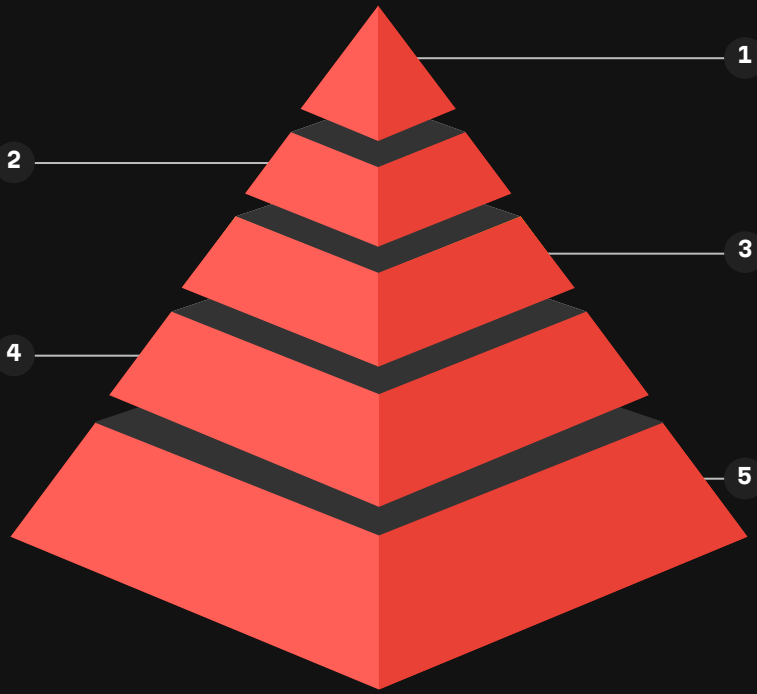
**Michele Mancioppi**
Chief Architect, Head of OSS & Community at Dash0






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

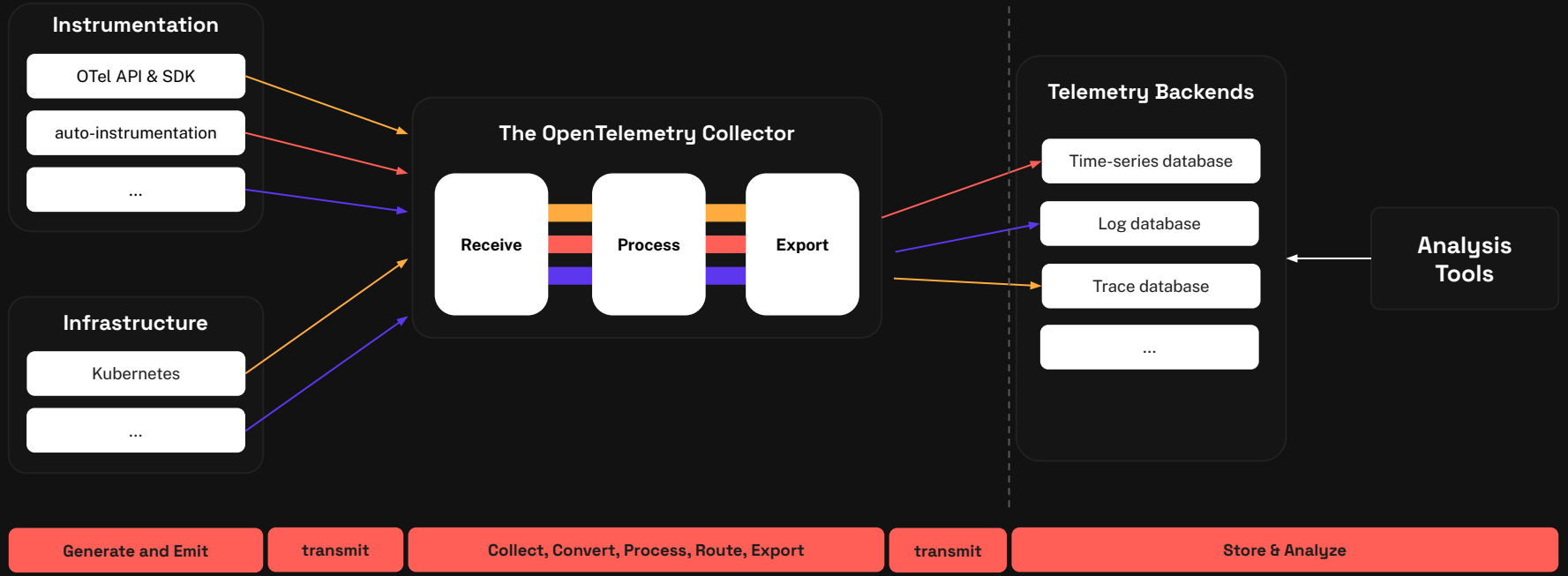
3 **Compute**
Telemetry SDK (stable) and (experimental)
Compute Unit and Instance
Operating System
Process & Process Runtimes
Device, Browser, Webengine, ...

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

...

5 **Infrastructure**
Cloud (general stuff)

# OpenTelemetry: a 1000 miles view



# 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

05

PART FIVE

# Observability as a platform product.

# Three phrases people mix up

NOT THIS



**An observability platform**

Something you can buy, and often worth buying. That is a purchase, not an operating model.

NOT THIS EITHER



**Platform observability**

Watching your own clusters.

THIS



**Observability as a platform product**

Part of what you expose to your developers, not something you buy and hand over. Telemetry arrives correct by default.

# So here is the contract I would sign...

| THE PLATFORM GUARANTEES | YOU DECIDE |
| --- | --- |
| Telemetry from every workload, no code change | Which business events matter enough to emit |
| Resource attributes correct: service, pod, cluster, environment, region, owner | Span attributes that describe the domain: cart value, tenant tier, feature flag state |
| Semantic conventions enforced before storage | Naming inside your own namespace |
| Signals correlated by construction | What “healthy” means. The SLO and its error budget |
| One auditable policy for retention, sampling, redaction, routing | The dashboards and alerts that encode what you know about this service |
| SDK, agent and Collector patched as a fleet | Being on call. Fixing the thing. |

> “Some people worry that by building things into the platform we’re removing autonomy and ownership. That’s not the case. What we’re doing is giving them focus… It’s the same thing cloud providers gave us by removing the need to worry about **power and physical security and cooling.** „



Abby Bangser
KubeCon+CloudNativeCon Keynote 2025

# Observability as a platform product

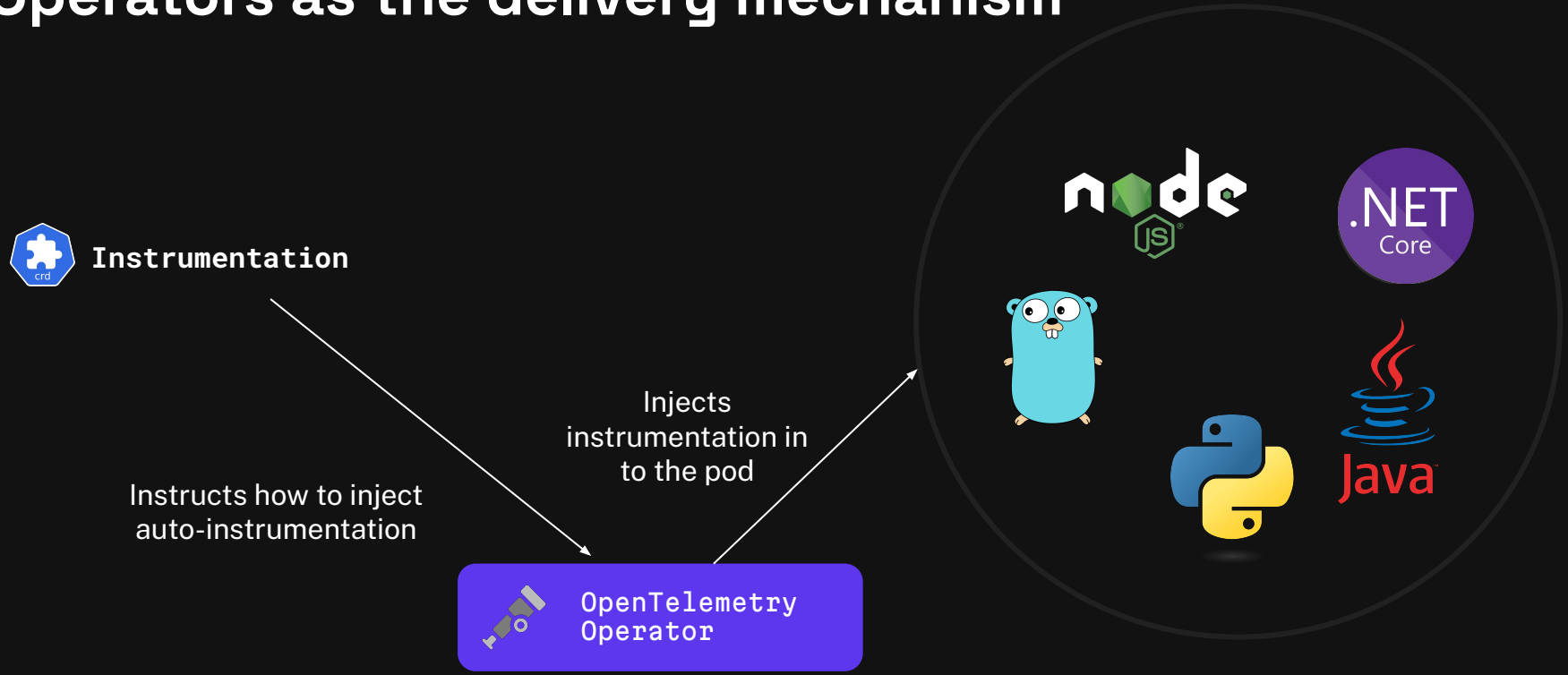


Paved Path for Observability

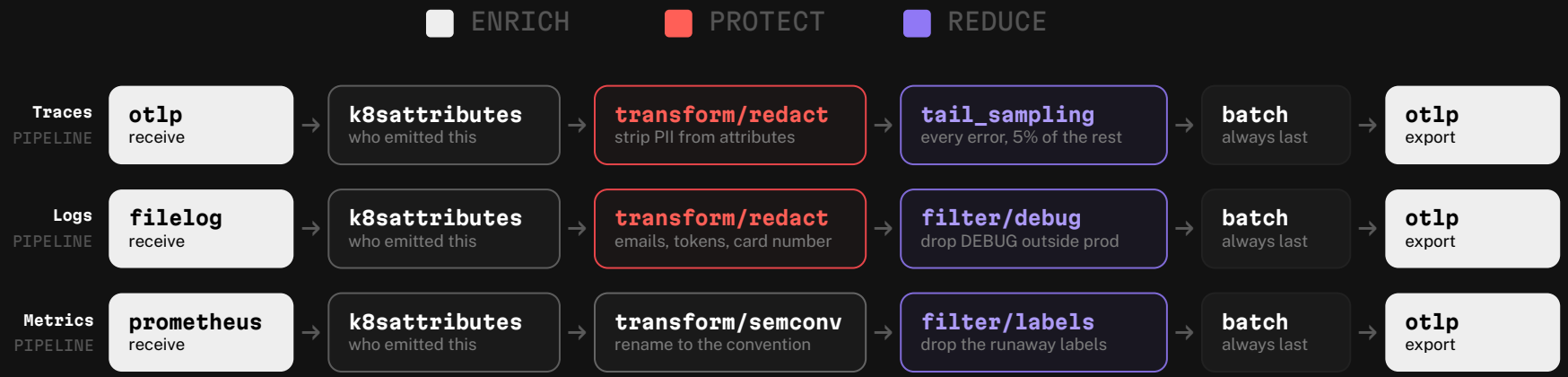
Developer adds business logic.
Everything else is platform.

- 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

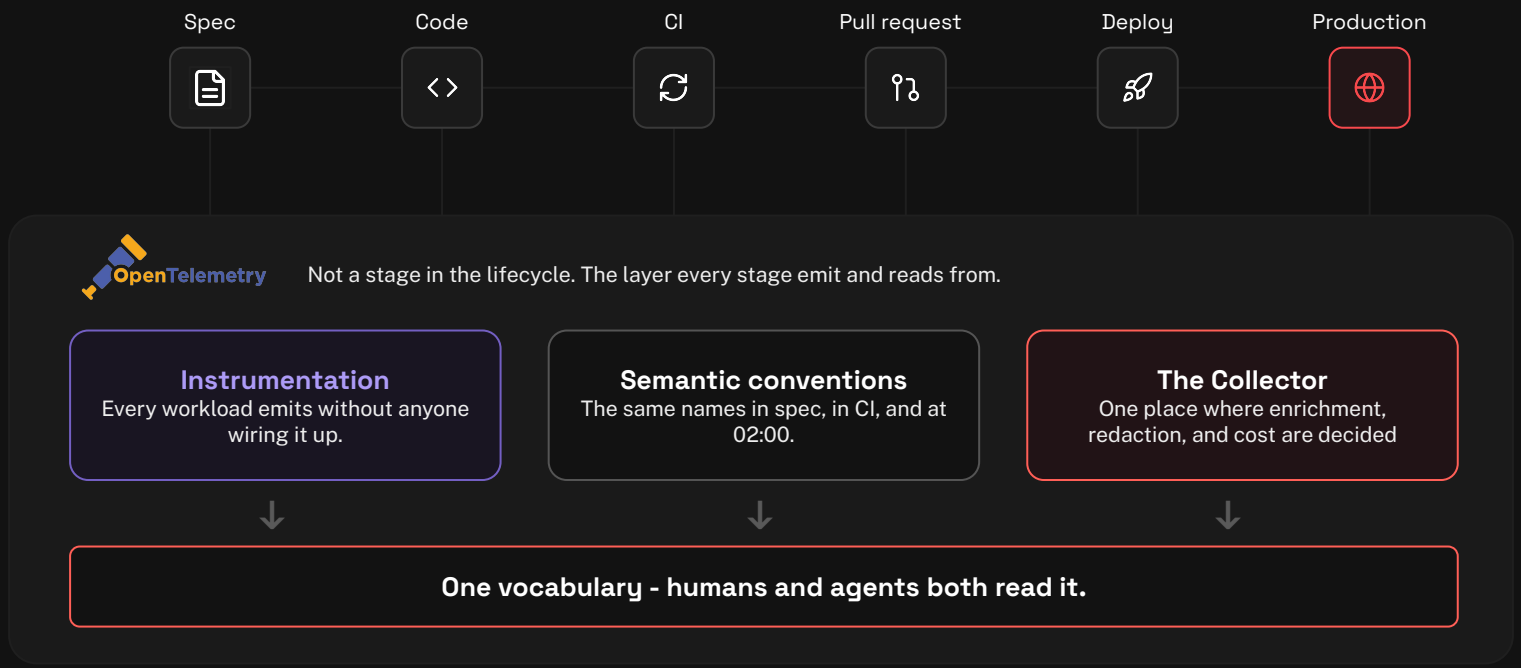


# And one place where all of this becomes true




Same enrichment everywhere. A different cost policy per signal.
**Written once, inherited by everyone**

# And it is the floor the whole lifecycle stands on



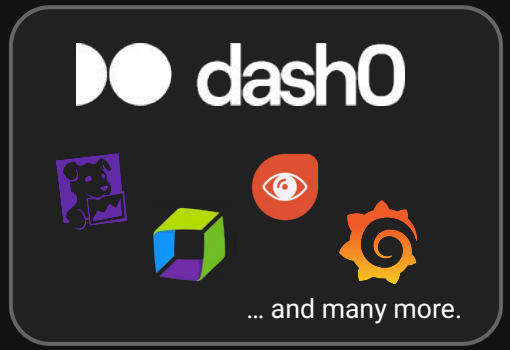
# Observability doesn't stop at instrumentation




**Your environment**
(k8s, cloud, etc)

How telemetry is produced and collected

**Vendors**






**OSS**

Where telemetry lives and how it's accessed

How humans and agents understand the system

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

# Demo!!!



https://github.com/dash0hq/otel-platform-demo

06

PART SIX

# What that unlocks.

# Why AI needs a platform first



GARBAGE IN

Inconsistent, fragmented, uncorrelated telemetry

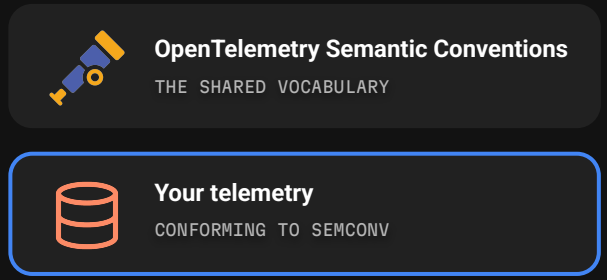
AI/LLM

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

GARBAGE OUT

Automated confusion. Faster. At scale.

# SemConv turn telemetry into something an LLM understands




YOUR DATA

Conform to them, and the magic is real.



AI/LLM

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



UNICORNS AND RAINBOWS

Correlation. Root cause. Answers you can trust.

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

# An AI SRE is a harness

THE MODEL

Rented.
Swapped quarterly.

Gets better without you.
Not your differentiator.

THE HARNESS

CONTEXT

Telemetry it can actually read

semconv | one service.name | correlated

TOOLS

Capabilities, not screens

query | MCP | deploy | roll back

MEMORY

What happened last time

runbooks | past incidents | docs

GUARDRAILS

What it may do unattended

blast radius | review gates | policy

OBSERVE · REASON · DECIDE · ACT

PRODUCES

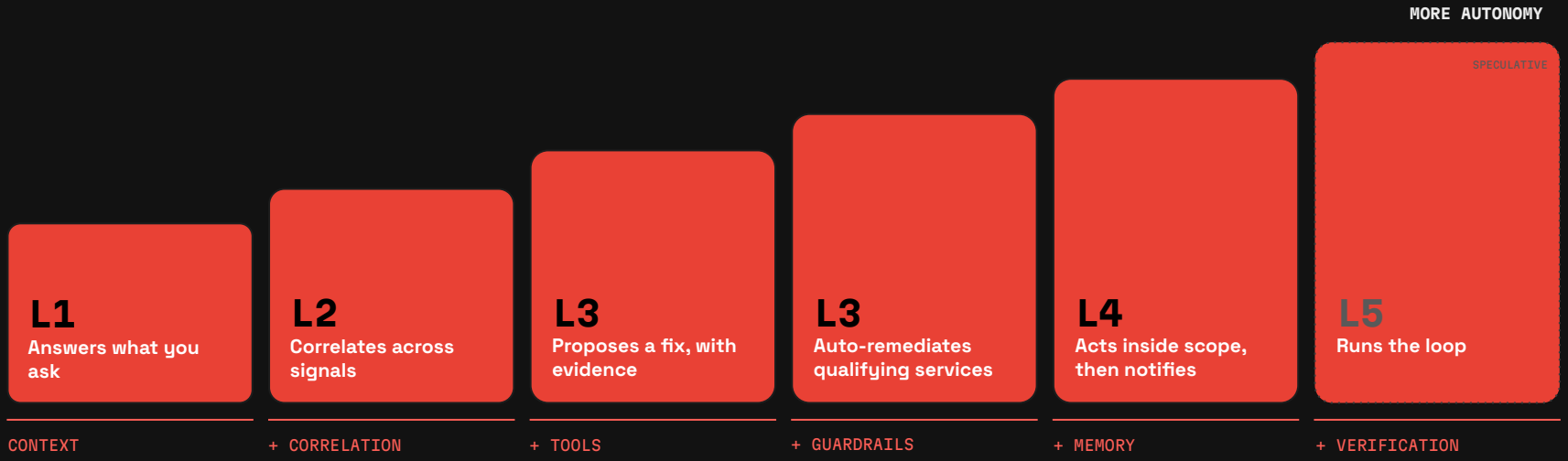
- Dashboards
- Alerts
- Runbooks
- Pull Requests
- … anything over MCP

OpenTelemetry

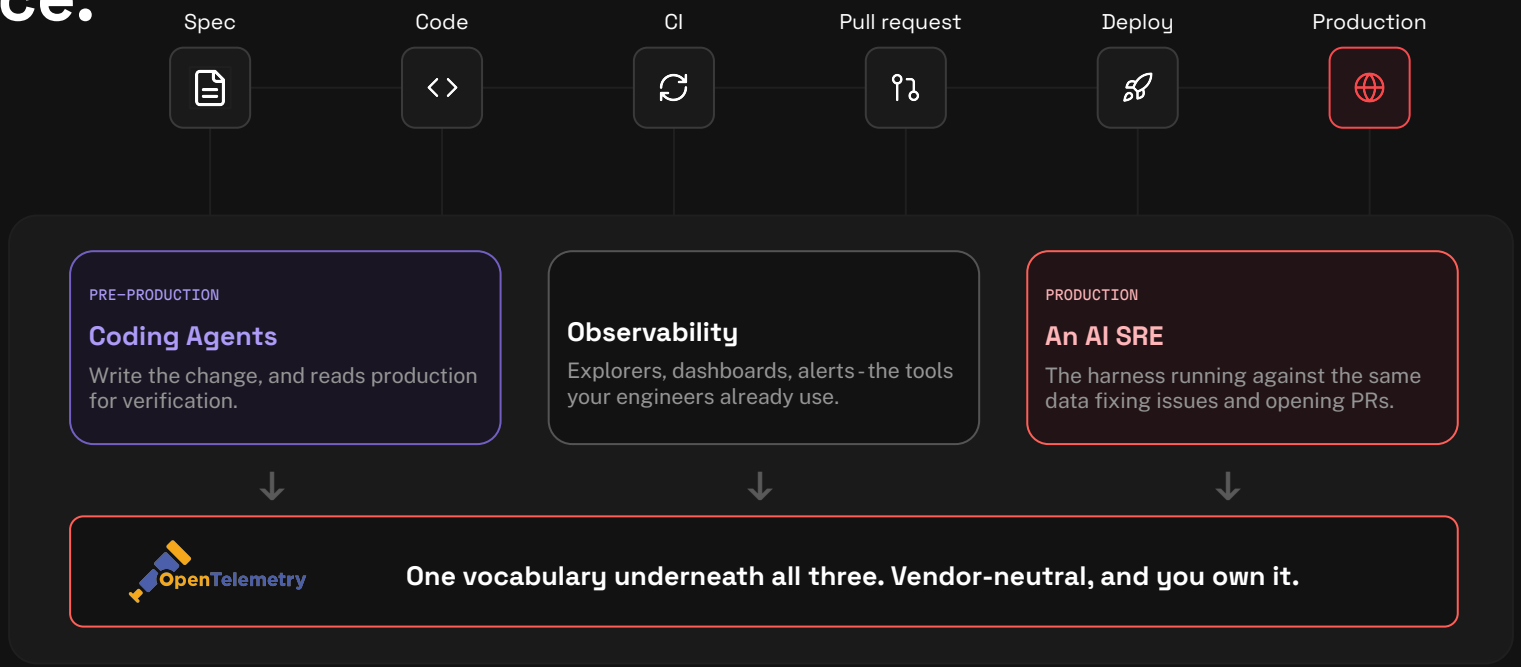
Semantic Conventions - The Collector - An OpenTelemetry-native backend

The context layer

# How much autonomy?



# Three things stand on it. You only build the floor once.



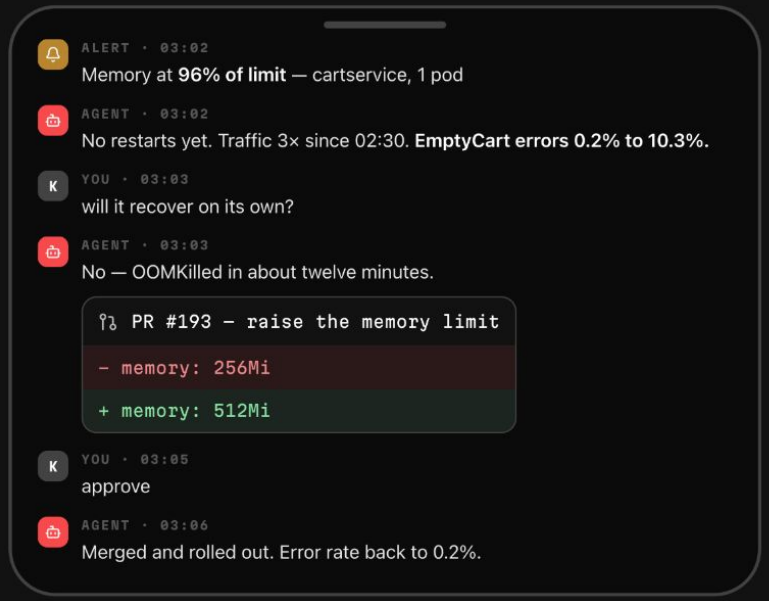
# 03:02 - and nobody gets out of bed

### IT REACHES YOU

In the thread you are already in.
Not a fifth tab.

### YOU ASK IN WORDS

No query language at three in the morning.




### IT PROPOSES, YOU DECIDE

A real pull request, into your existing review gates.

### FOUR MINUTES

No dashboard. No laptop.
Back to sleep.

07

PART SEVEN

# Wrapping up.

# In summary

01 **Convention over configuration already won. Twice.**
Rails, then Kubernetes. The right thing became the default thing.

02 **We keep drawing the line in the wrong place.**
Every team solving instrumentation once each is not autonomy. It is a tax.

03 **You did not buy observability. You bought an exit fee.**
The lock-in was never the contract. It was the agent inside your process.

04 **OpenTelemetry is the foundation, not the product.**
One vocabulary, vendor-neutral, and you own it. Getting data out is the easy half.

05 **Context is the thing you are actually building.**
Semantic conventions separate telemetry you store from telemetry anyone can reason about.

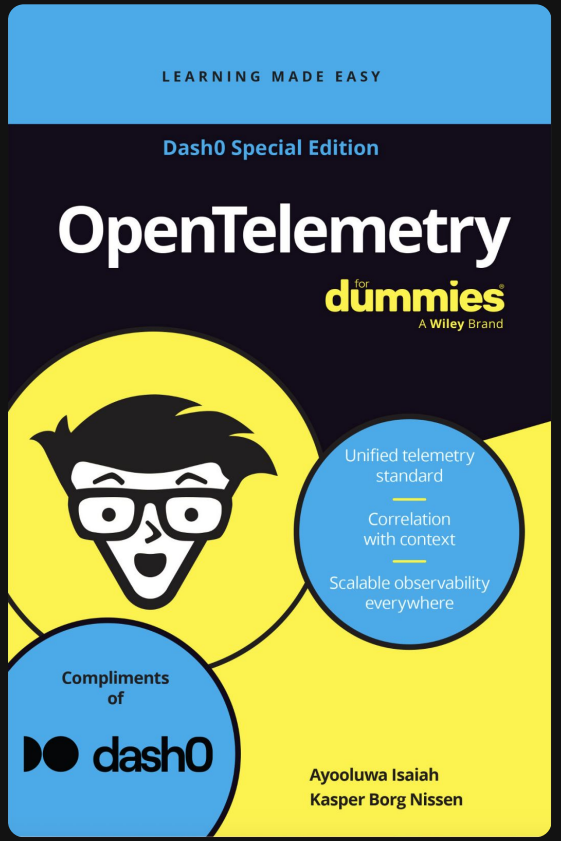
06 **Observability is a platform product.**
It has users, defaults, paved paths and an owner. If nobody owns it, everybody re-solves it.

07 **AI does not lower the bar. It raises it.**
An AI SRE is a harness, and every slot in it is something your platform either provides or does not.

# Dash0 Special Edition: OpenTelemetry For Dummies

This book shows you how to bring order and clarity to observability with OpenTelemetry. It explains its core components, practical setup steps, and the real-world impact of adopting an open observability standard.

Whether you're just exploring observability or ready to standardize across your organization, this book will demonstrate how to do it with confidence.



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

# Thank you. Questions?

Kasper Borg Nissen - Director of Developer Relations at Dash0

kaspernissen.xyz /in/kaspernissen kaspernissen

