dash0

PLATFORM CON 26

Observability as a Foundation for AI-Enabled Platforms

Kasper Borg Nissen, Director of Product Marketing & Developer Relations at Dash0

kaspernissen.xyz /in/kaspernissen kaspernissen

Who?

Director of Product Marketing & Developer Relations at Dash0

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

Author of OpenTelemetry for Dummies

CNCF Ambassador

Golden Kubestronaut

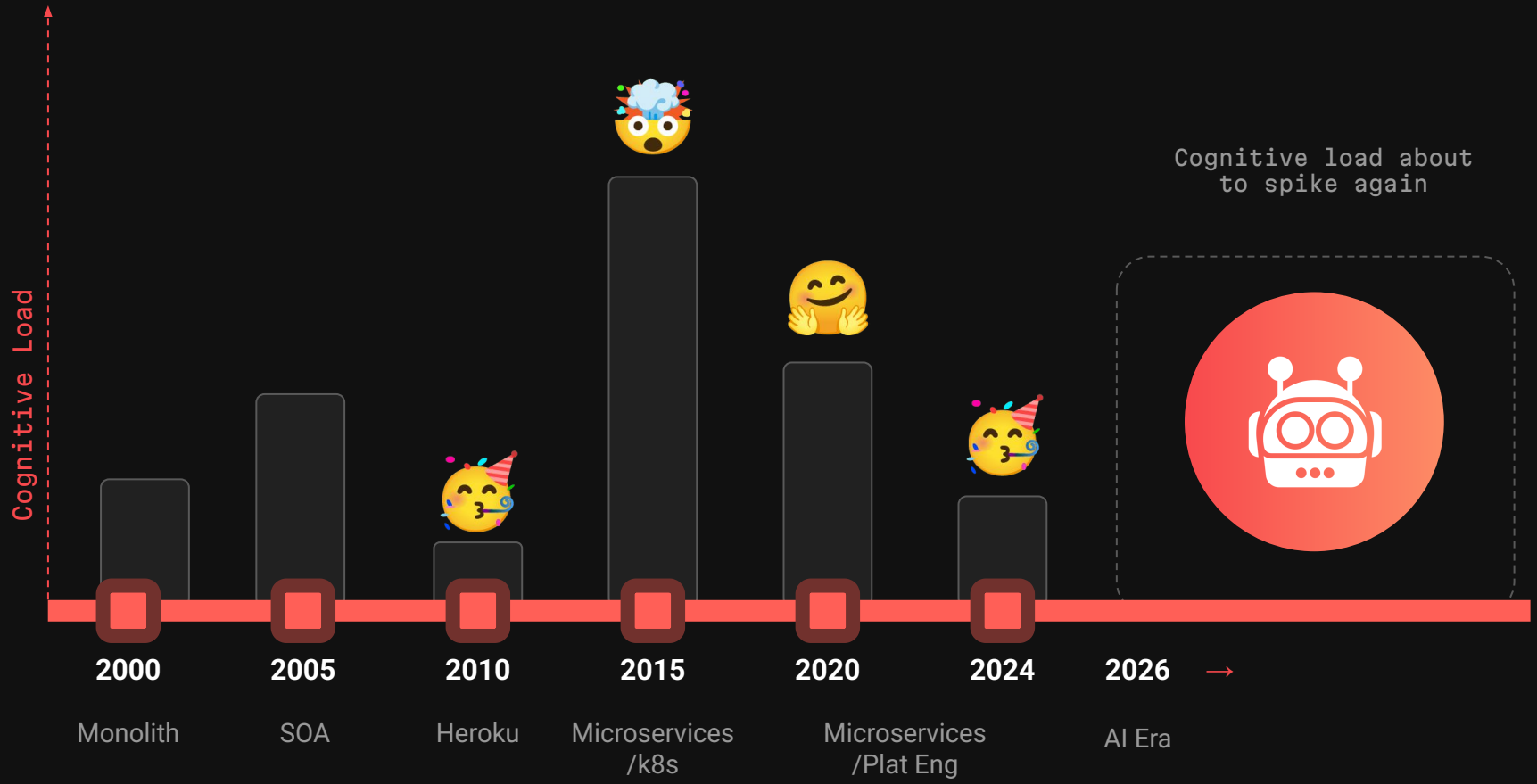
CNCG Aarhus, KCD Denmark Organizer

Co-founder & Community Lead Cloud Native Nordics

First things first…

OpenTelemetry

… NOW A CNCF GRADUATED PROJECT



dash0

Visualization Inspired by Daniel Bryant's talk

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

And, could you also…?

Build us an AI gateway

Multi-model routing, prompt logging, per-team rate limits.

Stand up inference

GPUs, llm-d or vLLM, autoscaling that doesn't melt the bill.

Govern what models see

Data boundaries, PII, model approvals, audit trails.

Attribute the cost back

Per-tenant token spend. Per-agent. Per request.

THE LAST ASK

Make it understandable when it breaks.

When the agent does something weird, and it will, somebody has to answer “why”.

AI workloads **aren’t like** anything we’ve operated

PROPERTY 1 – NON-DETERMINISM

Same input → Different output

Re-running the request does not reproduce the bug. The trace shape is unstable. Retries are not safe.

PROPERTY 2 – DYNAMIC TOOL USE

Calls invented at runtime.

The call graph is generated, not declared. Every tool just got a new caller - one that doesn't read your API docs.

PROPERTY 3 – VARIABLE LOAD

Token economics, not RPS.

Capacity is no longer requests per second. A single bad prompt can balloon spend 100×.

PROPERTY 4 – OPAQUE DECISIONS

Why did it do that?

There is no stack trace. The reasoning happened inside a model you don't own. You're interrogating, not debugging.

Three personas now share the same platform.

A third persona arrived on the platform…

Human operator

- Requests access
- Makes intentional changes
- Reviewable by another human

Services

- Declared behavior
- Configured access
- Emits telemetry

Agents

- Plans dynamically
- Chains tool calls
- Needs access to the platform's observability data

The platform's job is to absorb the complexity for all three.

The thing the AI conversation forgot to mention …

OpenTelemetry

The last observability agent you will ever install

49%	#2	+35%
IN PRODUCTION	CNCF PROJECT BY ACTIVITY	CONTRIBUTORS ‘25

AI doesn’t replace **platform thinking**.

It increases the need for it.

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**.

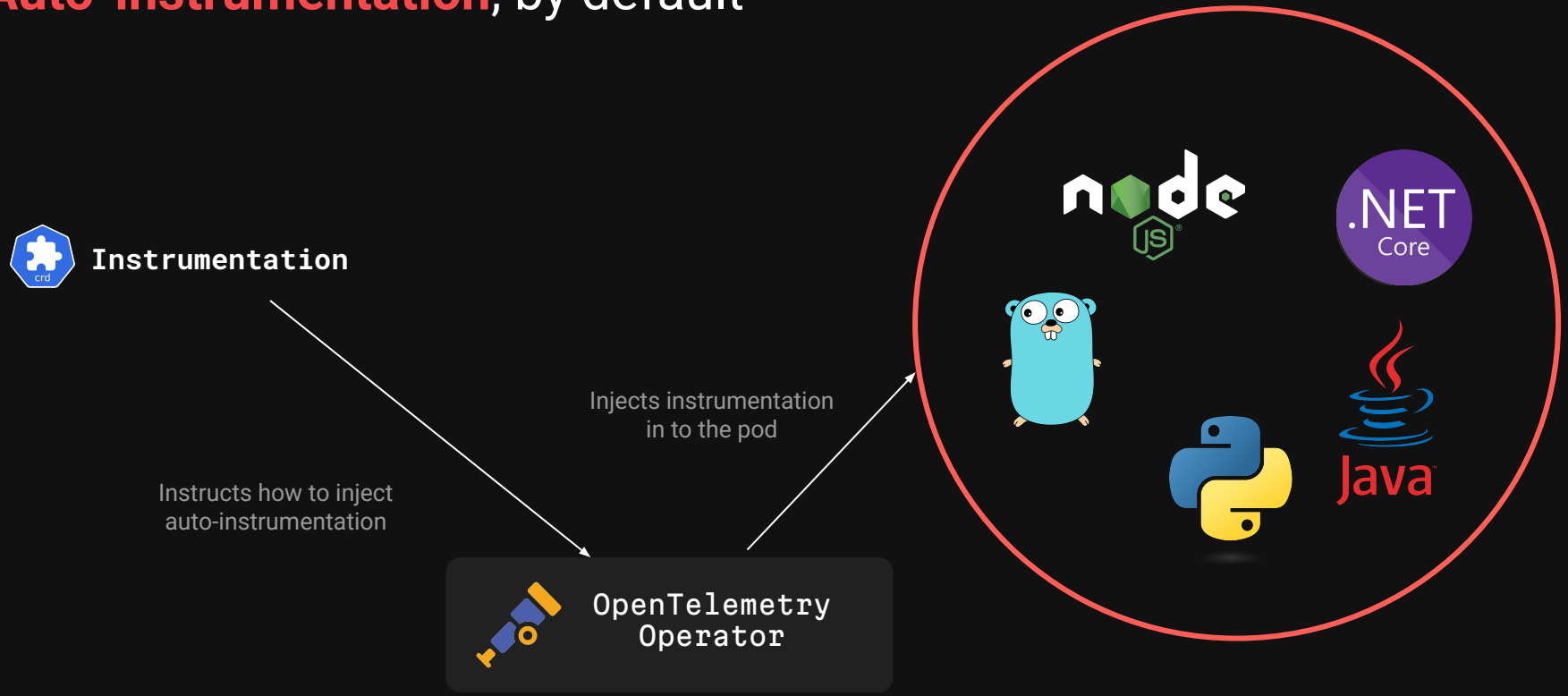
Creating a paved path for Observability

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

Auto-instrumentation, by default



What the agent is to your platform

`Workload. Tenant. Operator.`

One new persona. Three relationships with your platform - and a different job for each. Every one of them rests.

01 Workload
OBSERVE IT

Non-deterministic. Opaque. A call graph generated at runtime. You have to be able to reconstruct *why* it did what it did - after the fact.

AGENT OBSERVABILITY

02 Operator
TRUST IT

It reads your telemetry and acts on production. An AI SRE is only as good as the data it reads - and autonomy is *earned one rung at a time*.

AI SRE & THE AUTONOMY LADDER

03 Tenant
SERVE IT

Instrumentation, gatew identity, cost attrib The developer ad usiness logic. Everyth else is *inherited from th latform*, not rebuilt per team.

Topic for another talk

THE PAVED PATH

All three rest on the same foundation - OpenTelemetry

01

Workload

OBSERVE IT

How do we actually observe AI Workloads?

Every familiar signal has a **new equivalent**.

What we built for services	What we are building for agents
Stack traces	Reasoning chains
Status codes (2xx, 5xx)	Quality signals (hallucinations, confidence, evals)
RPS & Latency	Tokens, cost, blast radius
Static call graph	Dynamic tool invocation
Replayable requests	Non-deterministic runs

The old practice was right for it's time. It isn't enough now.

One vocabulary.

Shared vocabulary for GenAI workloads

Workload 01

```
# span.attributes
gen_ai.operation.name          = "chat"
gen_ai.system                  = "openai"
gen_ai.request.model           = "gpt-4o"
gen_ai.request.temperature     = 0.2
gen_ai.request.max_tokens      = 2048
gen_ai.usage.input_tokens      = 1283
gen_ai.usage.output_tokens     = 412
gen_ai.response.finish_reasons = ["stop"]

# tool call
gen_ai.tool.name               = "kubernetes.list_pods"
gen_ai.tool.call.id            = "call_84ax2"

# resource (process)
service.name                   = "agent-sre"
deployment.environment.name    = "production"
```

STABILIZED CONCEPTS

- Operation name & system
- Model + parameters
- Token usage
- Finish reasons
- Tool calls

One vocabulary? Not yet.

Workload 01

Five conventions for the same span

OTel GenAI SemConv

OPTIMIZES FOR VENDOR NEUTRALITY

`gen_ai.request.model`

OpenInference

OPTIMIZES FOR EVALUATION WORKFLOWS

7 span kinds - agent, tool, retriever, guardrail, evaluator…

OpenLLMetry

OPTIMIZES FOR DEVELOPER ERGONOMICS

`llm.model_name`

LangSmith

OPTIMIZES FOR THE LANGCHAIN ECOSYSTEM

Framework-native taxonomy

Langfuse

OPTIMIZES FOR IT'S OWN PLATFORM SCHEMA

Fifth name for the name

These are legitimate differences. Not naming preferences. They are not going away soon.

The platform **answer**.

Workload 01

Normalize at the edge.

OTel Collector processor

genainormalizer

Rewrites OpenInference and OpenLLMetry spans into GenAI semconv — in the pipeline, not in the apps.

OpenTelemetry

Spring AI

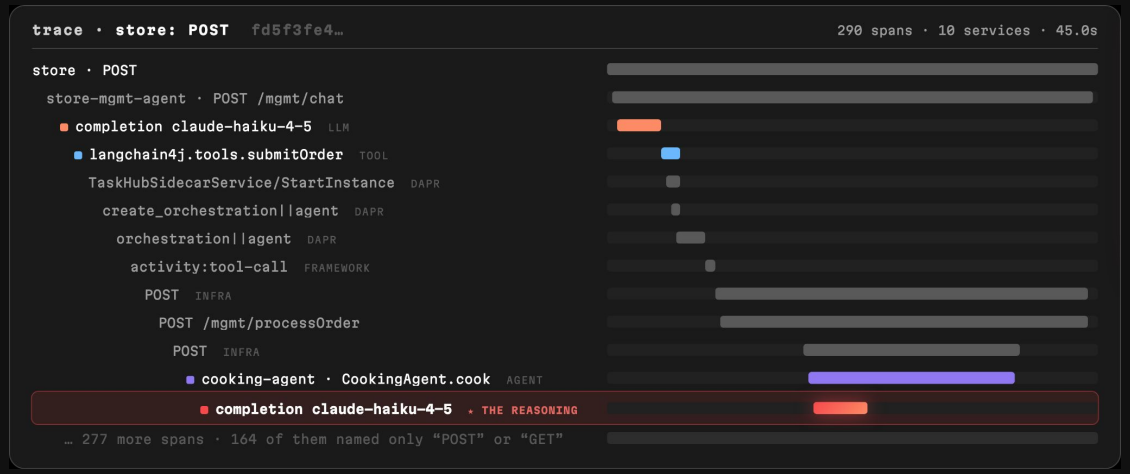
Arconia

Switch flavors without touching application code.

By Thomas Vitale

The answer is in the trace. **Reading it might be hard.**

Workload 01



THE ONE SPAN YOU ACTUALLY WANTED

`gen_ai.completion` =
"I'll cook one Pepperoni pizza for you. Let me follow the workflow step by step. STEP 1: Get inventory and acquire ingredients"

```
gen_ai.request.model = claude-haiku-4-5
gen_ai.usage.completion_tokens = 71
gen_ai.response.finish_reasons = TOOL_EXECUTION
```

More telemetry is not better telemetry. The signal is in the spans - extracting it is the frontier.

`gen_ai.completion is being deprecated in favor of: gen_ai.output.messages`

290	164	15
SPANS	SPANS	SPANS
For One order	Named only POST / GET	Carry the reasoning

dash0

02

Operator

TRUST IT

How to delegate work to our Agents?

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

Operator 02

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.

Six levels of AI autonomy

Operator 02

		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

L1 AI-Assisted (Spicy Auto-complete)

Operator 02

The AI serves as a reference tool or orphaned search, like a smarter Stack Overflow. The developer manually writes all the code, using AI for specific snippets or answers.

Source: https://www.danshapiro.com/blog/2026/01/the-five-levels-from-spicy-autocomplete-to-the-software-factory/

L2 AI-Generated,Human-Reviewed (Coding intern)

Operator 02

The AI writes unimportant or boilerplate code. The developer prompts for specific functions but immediately reviews and integrates the output.

dash0

Source: https://www.danshapiro.com/blog/2026/01/the-five-levels-from-spicy-autocomplete-to-the-software-factory/

L3 AI-Generated, Auto-Reviewed (Junior developer)

Operator 02

An interactive ‘pair programmer’ partnership. The developer and AI trade off control, with the human reviewing code as it’s generated in real-time.

dash0

Source: https://www.danshapiro.com/blog/2026/01/the-five-levels-from-spicy-autocomplete-to-the-software-factory/

L3.5 Selected Auto-Merge (Developer)

Operator 02

The AI generates the majority of the codebase. The developer reviews everything the AI does, acting as the bottleneck for verification before processing.

dash0

Source: https://www.danshapiro.com/blog/2026/01/the-five-levels-from-spicy-autocomplete-to-the-software-factory/

L4 Mostly Autonomous (Engineering team)

Operator 02

The AI runs unattended for long periods, handling complex tasks. The human trusts the system's self-checks, only checking the final results much later.

dash0

Source: https://www.danshapiro.com/blog/2026/01/the-five-levels-from-spicy-autocomplete-to-the-software-factory/

L5 Dark Factory (Dark software factory)

Operator 02

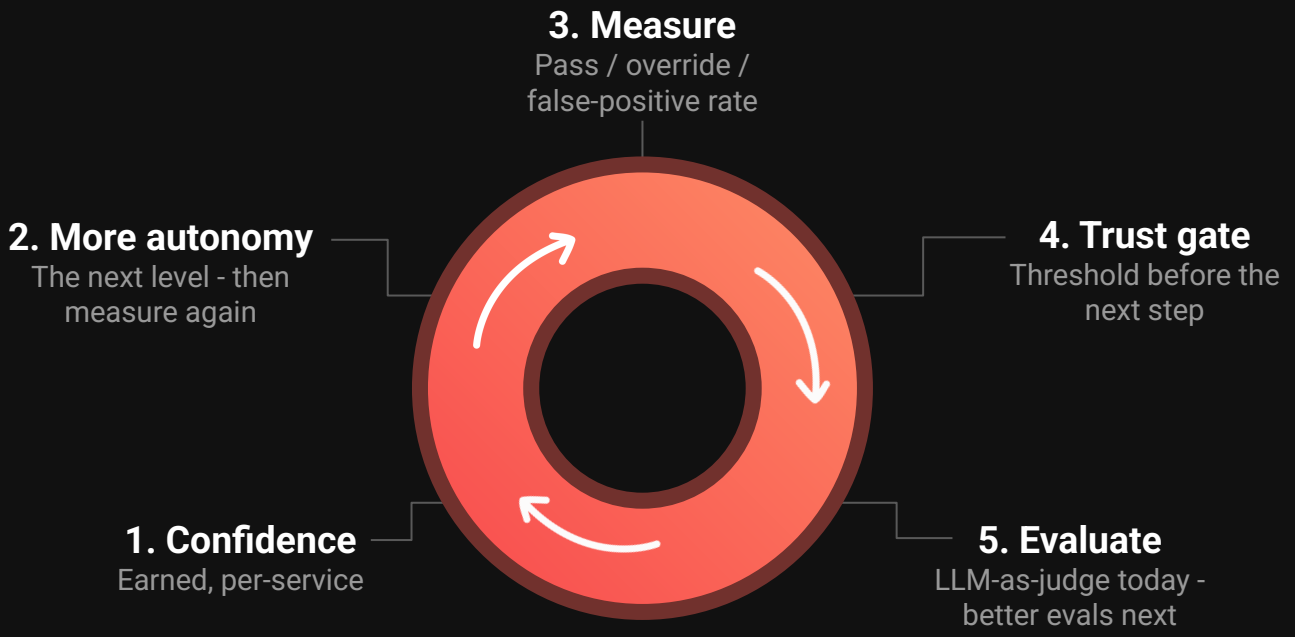
The engineer manages the goals and the system, not the code. They provide plain English descriptions. The AI defines implementation, writes code, tests, fixes bugs, and ships.

dash0

Source: https://www.danshapiro.com/blog/2026/01/the-five-levels-from-spicy-autocomplete-to-the-software-factory/

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.

Your platform climbs the ladder too.

Operator 02

L2	YOU'RE PROBABLY HERE	Code review CI/CD, shared infrastructure. You already have this.
L3	THE REVIEW STACK	Automated review gates. Eval frameworks. Audit logs. Spec templates.
L3.5	TRUST MEASUREMENT	Per-service pass rate, override rate, false-positive rate. Earned, not declared.
L4	CODIFIED KNOWLEDGE	Memory. Retrieval. Runbooks. Scoped sandboxes. Every recurring failure becomes durable substrate.
L5	GROUND-TRUTH SUBSTRATE	Specs validated against live telemetry. The factory tunes itself on real production data.

Observability is the load-bearing piece at every rung. Without telemetry, no measurement. Without measurement, no promotion

Reason. Act. **Observe**. Decide

Operator 02

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

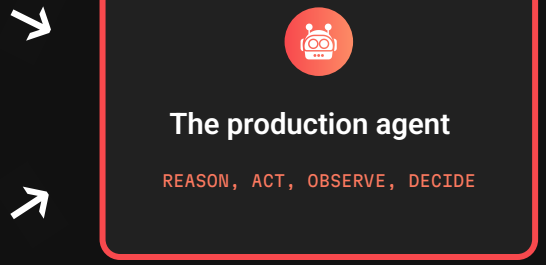
Facts - what it knows

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

DEPLOYMENTS FLOW RIGHT

PRE-PRODUCTION

Humans & Coding Agents

Specs become code. Code becomes pull requests. Pull requests become deployments.

Plan → Design → Code → Review → Test → Build → Deploy

The front half. Solved

PRODUCTION

Production Agents

Observability used to visualize this half. Agents now act on it - and ship the fix back.

Operate → Scan → Investigate → Fix · PR → Document → Improve

The back half. Now agentic.

THE FIX RETURNS AS A PR

runs underneath both halves.

Did you know: Observe your Coding Agents too…

The same OpenTelemetry pipeline — front half of the SDLC, not just production

Claude Code

Metrics + events, traces (beta), tokens, cost, tool calls, lines changed.

`gen_ai.* SemConv`

Gemini CLI

Logs + metrics, traces (opt-in), OTLP to any backend

`gen_ai.* (partial)`

OpenAI Codex CLI

Logs + Traces + Metrics
OTLP via config.toml

Cline

Metrics + logs
Built-in OTLP exporter

Goose

Traces + metrics + logs
Standard OTEL_* env

OpenCode

Traces + Logs
OTLP endpoint

Still locked in proprietary dashboards: Cursor, GitHub Copilot, Windsurf, zed, Continue

Extend Claude Code with: https://github.com/dash0hq/dash0-agent-plugin

The ecosystem is building on it.

CNCF SANDBOX

kagent

Agents as first-class workloads on Kubernetes. Agent lifecycle as a custom resource. Every step in the loop is a span.

CNCF SANDBOX

HolmesGPT

Agentic SRE troubleshooter. Reads OTel data directly. Calls observability tools via MCP. Escalates to humans.

CNCF SANDBOX

k8sgpt

Scans your cluster, identifies issues, explains them in plain language. The earliest "AI SRE" shape we had - still one of the cleanest.

OPEN SOURCE

agentgateway

Tool-call data plane. Sits between every agent and every tool. Natural enforcement point. Natural observability point.

Build it now. AI moves **faster** than you might think.

01

Instrument every layer

Agent runtime, gateway, MCP server, inference. None get a pass.

02

Adopt GenAI SemConv today

Even before they're finished. Waiting costs more than starting.

03

Govern AI like you govern services

Identity. Policy. Cost. Observability. Audit. Disciplines aren't new. Customers are.

The bottleneck is no longer the model. It's the telemetry feeding the evals.
The cleanest pipeline today buys the fastest autonomy ramp when the rest of the stack catches up.

The job has been the **same** all along

Kubernetes

PLATFORM ANSWER

Platform Engineering

Observability

PLATFORM ANSWER

OpenTelemetry

AI & Agents

PLATFORM ANSWER

Observability as the foundation

Don’t panic. You’ve **built this before**.

YOU ALREADY HAVE

GitOps & preview environments

Validate an agent’s PR in an ephemeral environment before a human ever sees it.

ArgoCD | Flux | vCluster

YOU ALREADY HAVE

Progressive delivery

Canary an agent’s change behind a trust gate before you let it merge on it’s own.

Argo Rollouts | Istio | Linkerd

YOU ALREADY HAVE

DORA metrics & pipeline telemetry

Measure the AI tuition tax - don’t adopt in the dark.

YOU ALREADY HAVE

OpenTelemetry & GenAI SemConv

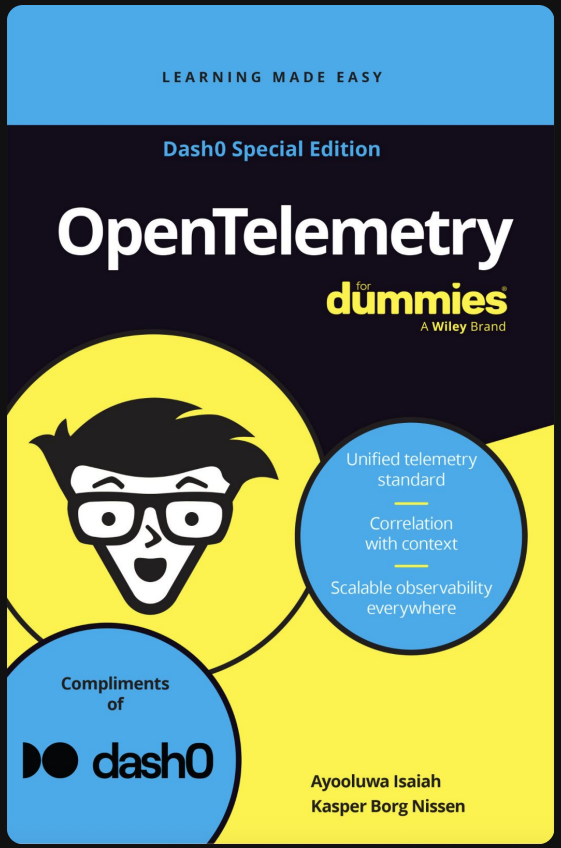
Observe the whole SDLC - CI, PRs, Tickets - not just prod. Interoperable across tools.

Expect a dip first - deploy frequency, failure rate up. That’s the tuition, not failure. Measure it and it pays back.

as the foundation for the

Agentic SDLC

Build the foundation.
Make it **boring**.
Let developers ship.



WANT TO GO DEEPER?

Free copy of OpenTelemetry for Dummies.

The Dash0 Special Edition - by Ayooluwa Isaiah and Kasper Nissen. Unified telemetry, correlation with context, and scalable observability, in plain language.

DOWNLOAD:

dash0.com/lp/opentelemetry-for-dummies

Giveaway!

Score the New Agent0 Kit!

Enter for a chance to win the limited-edition jersey shown here, plus other football surprises.

SCAN THE QR CODE FOR A CHANCE TO WIN!

Winners will be randomly drawn and contacted via email to coordinate shipping.
No fouls, just pure luck.

dash0

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

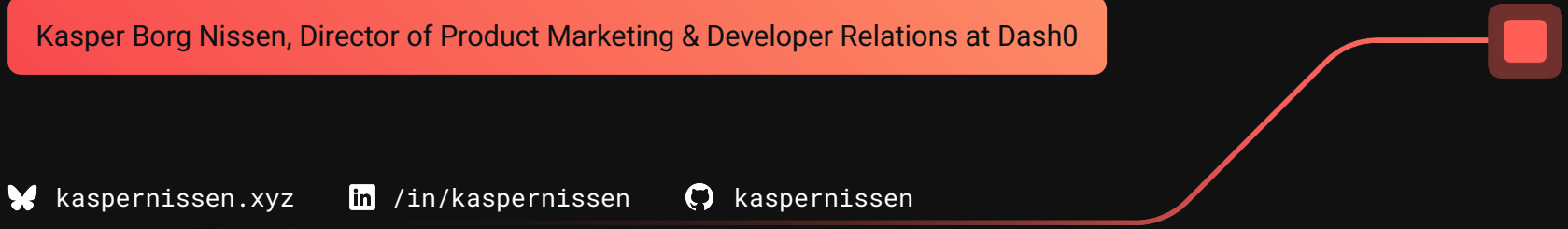
community.cncf.io/merge-forward

dash0

PLATFORM CON 26

Thank you!

Kasper Borg Nissen, Director of Product Marketing & Developer Relations at Dash0

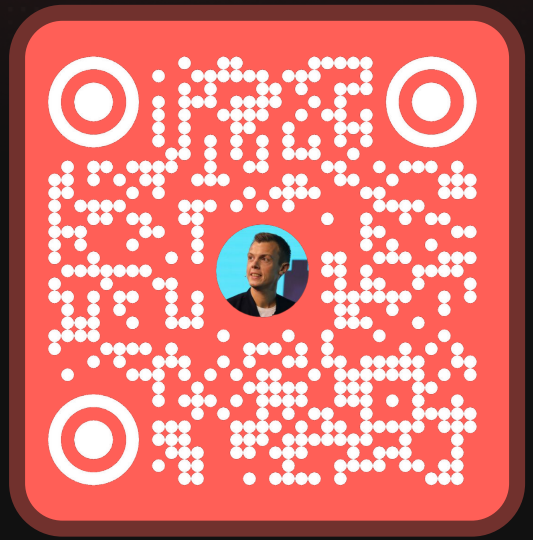


kaspernissen.xyz /in/kaspernissen kaspernissen

dash0

PLATFORM CON 26

Get in touch!

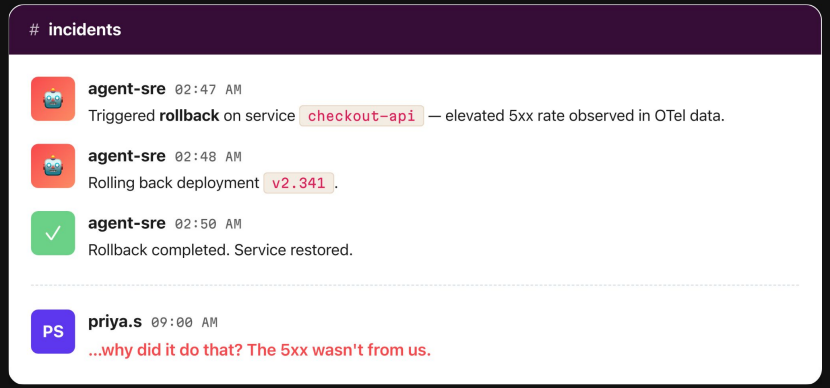


kaspernissen.xyz /in/kaspernissen kaspernissen

Stop looking. Start **asking**.

You're not jumping between dashboards anymore.
You're **asking** a system you can't replay - so it had better have captured the answer.

- What telemetry did the agent read?
- Which tools did it call?
- What did the model return?
- Did anything change after?



None of these are solved by a smarter model.

Tenant 02

Seven guarantees, three lenses…

WHO

01
Identity
Who is the agent? On whose behalf?

02
Authorization
What can it use, touch, or approve?

WHERE

03
Isolation
Where does code execute, what persists?

04
Egress
What is reachable, blocked, mediated?

HOW

05
Scheduling
Whose quota, what priority, what topology?

06
Governance
Tools, prompts, models - versioned.

07

Observability
Six guarantees are unverifiable without the seventh. OpenTelemetry is how the contract becomes legible

You can create a similar experience for your AI workloads

Paved Path for AI Workloads

Developer adds business logic.
Everything else is platform.

- GenAI auto-instrumentation
- OTel Collector
- Semantic Conventions
- Agent Gateway
- MCP servers
- Cost Attribution
- Identity
- Policy engine

AI expands the Platform Engineer's two hats

Observing the platform

- Cloud, cluster, CI/CD
- Shared databases
- Internal API's & messaging

And now...

- Agent infrastructure layer NEW
- Inference layer NEW

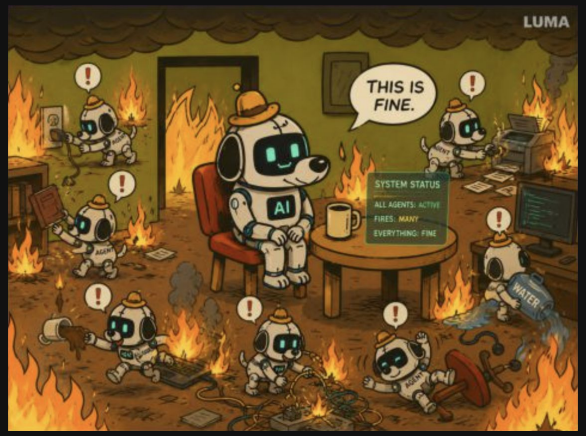
Enabling developers

- Traces, metrics, logs by default
- Profiling
- Dashboards-as-code

And now...

- Agent-readable conventions NEW
- AI-assisted workflows NEW

Same dual responsibility. Now extended one layer up.



"AI Increases the demand for platforms because it accelerates software creation faster than organizations can safely operationalize it."

Daniel Bryant, Head of Product Marketing at Syntasso

From the talk: Platform engineering for developers & architects & the rest of us (AI agents), LDX3 2026.

Source: https://speakerdeck.com/danielbryantuk/platform-engineering-for-developers-architects-and-the-rest-of-us-ai-agents?slide=7