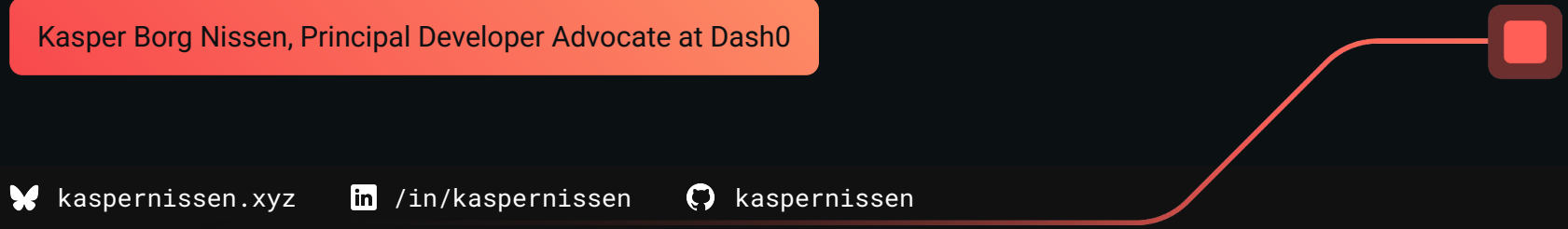
dash0

Cloud Native Computing Rheinland

Breaking Free with Open Standards:

OpenTelemetry and Perses for Observability

Kasper Borg Nissen, Principal Developer Advocate at Dash0



kaspernissen.xyz /in/kaspernissen kaspernissen

Who?

Principal Developer Advocate 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

tl;dr;

OpenTelemetry is standardizing telemetry collection.

Perses is standardizing dashboarding.

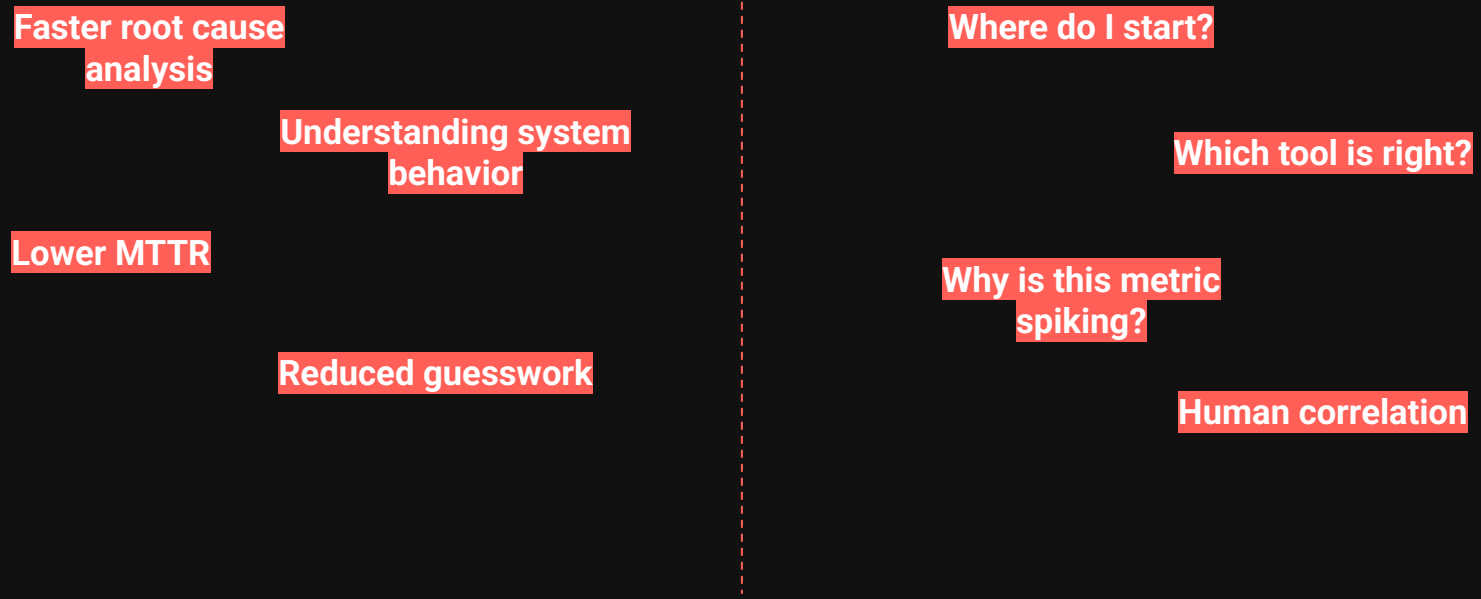
Applying Platform Engineering principles transforms observability into a seamless, scalable, and developer-friendly experience.

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

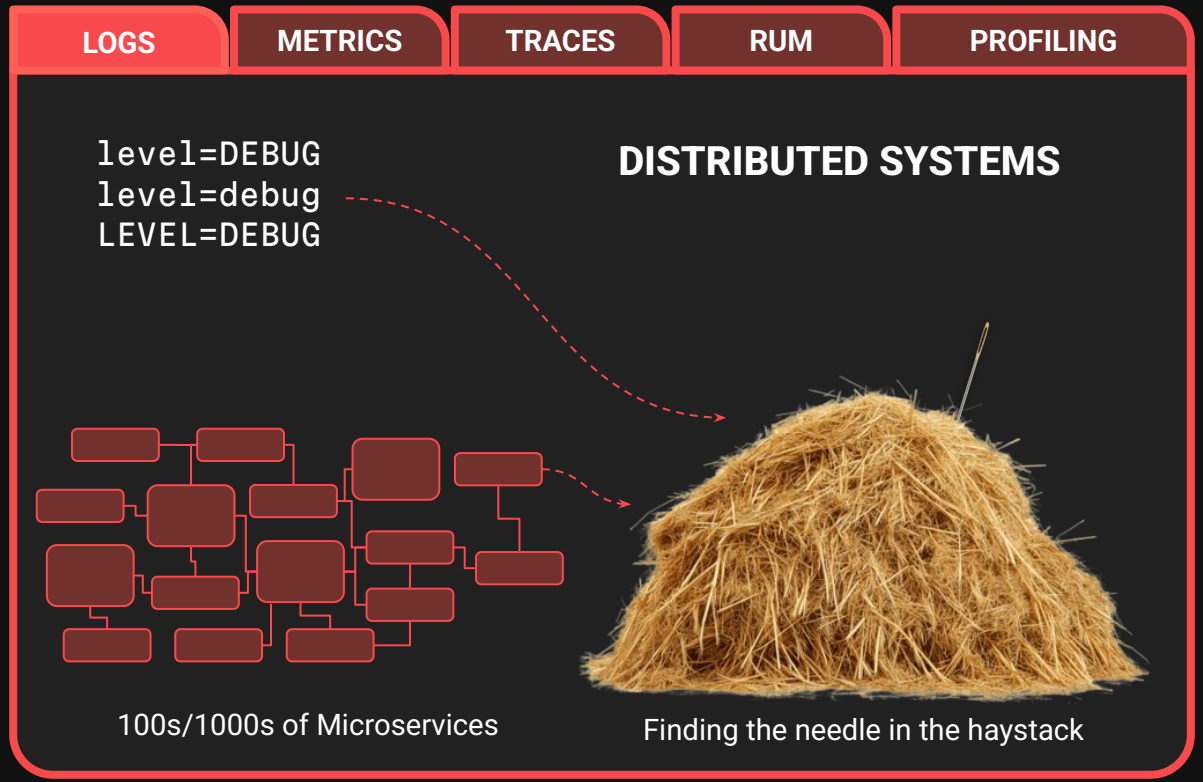
Today…

… observability promised a lot.

Observability promised a lot



The current reality: The browser tabs of Observability



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

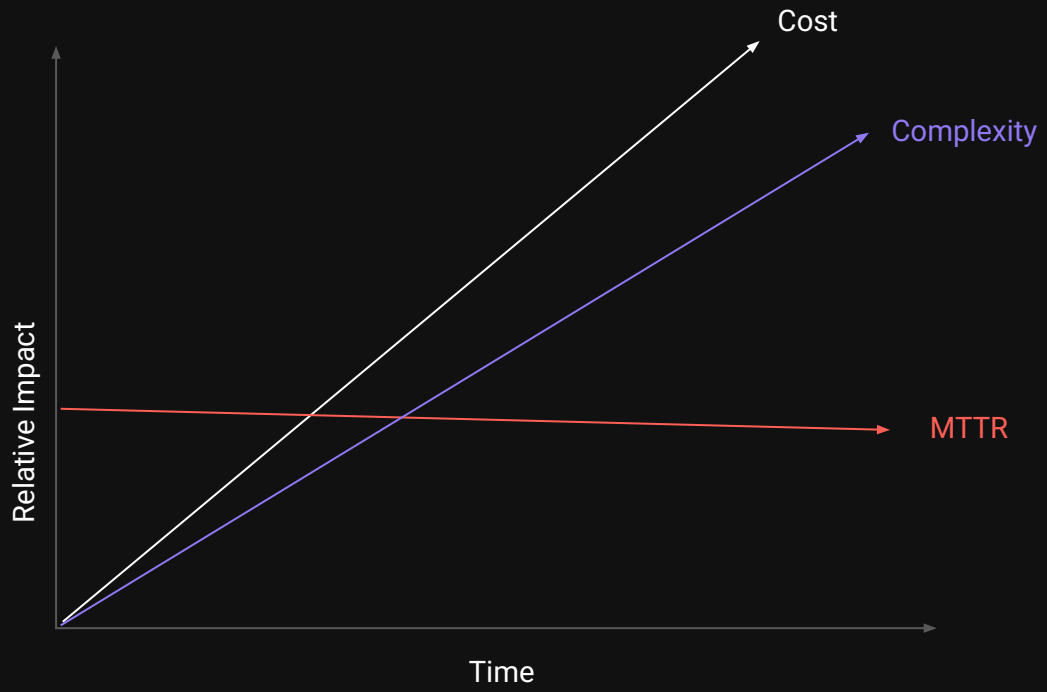
Metadata Inconsistency

No instrumentation due to high complexity

Lack of unified insights

The cost & complexity paradox

More telemetry, more tooling - same time to recovery



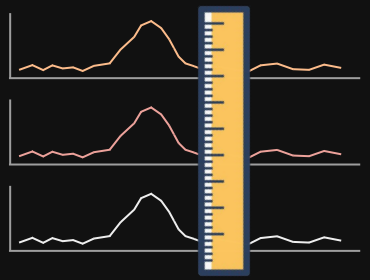
Up to 84% of current observability users struggle with the costs and complexity of their daily monitoring responsibilities.

Gartner Hype Cycle Report, 2025

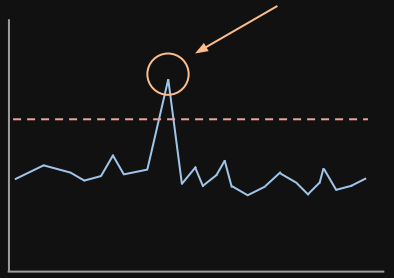
Source: https://www.gartner.com/en/documents/6755734

A shift is happening.

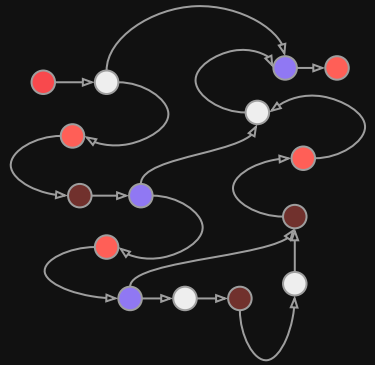
A shift toward correlation



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, metrics, logs, profiling & RUM (experimental)

The main goals of the project are:

- Unified telemetry
- Vendor neutrality
- Cross platform

OpenTelemetry in a nutshell

OpenTelemetry is a toolkit and a specification.

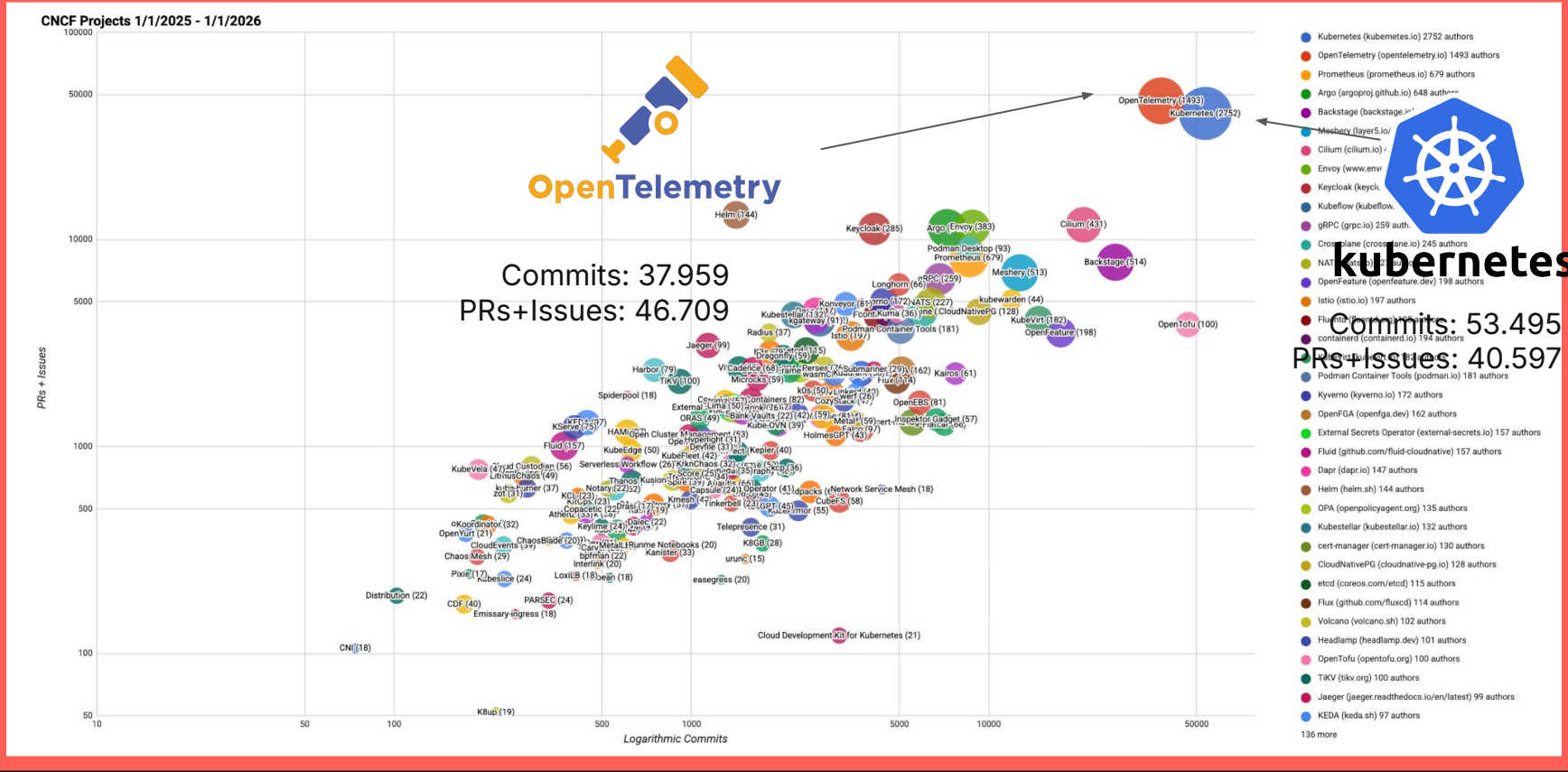
What it is

- 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.

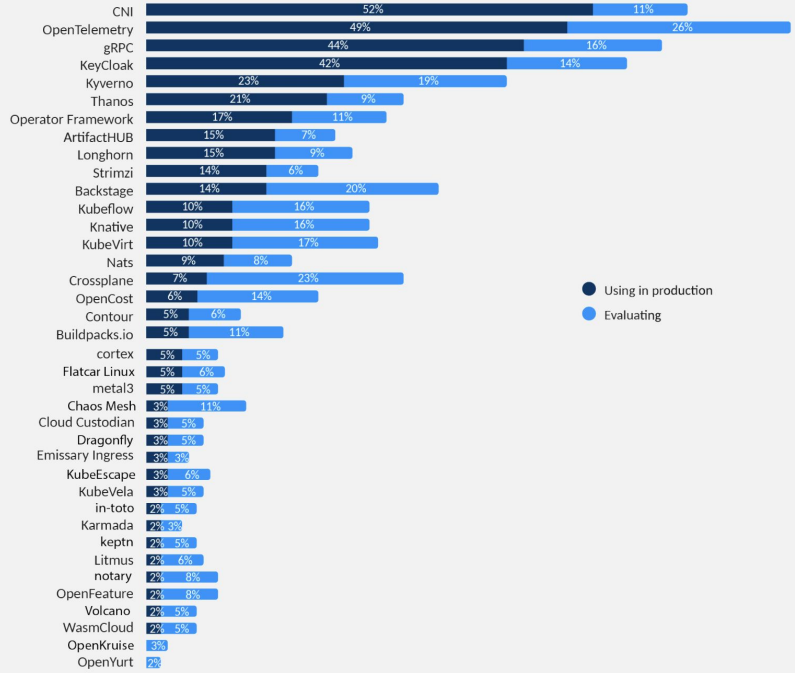
26%

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)

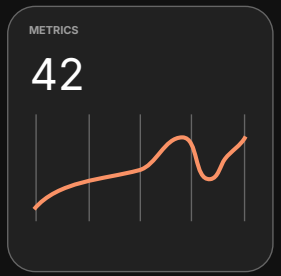


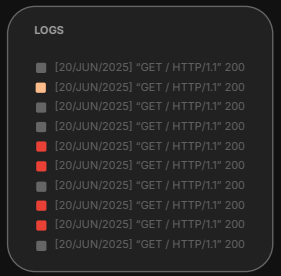
2025 CNCF Annual Survey, Q33, Sample size = 395-502, DKNS excluded

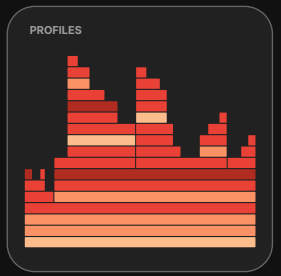
dash0

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

Signals







dash0

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

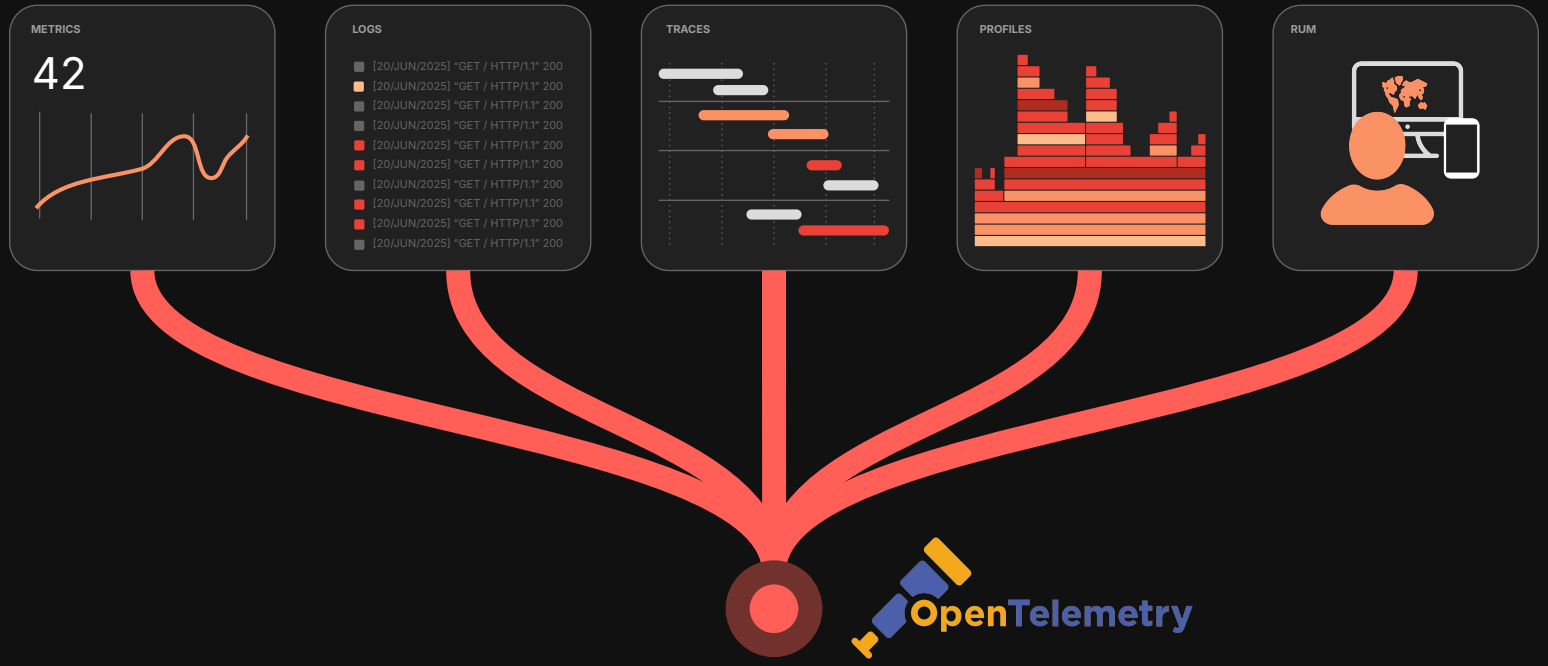
Metrics Logs Traces

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

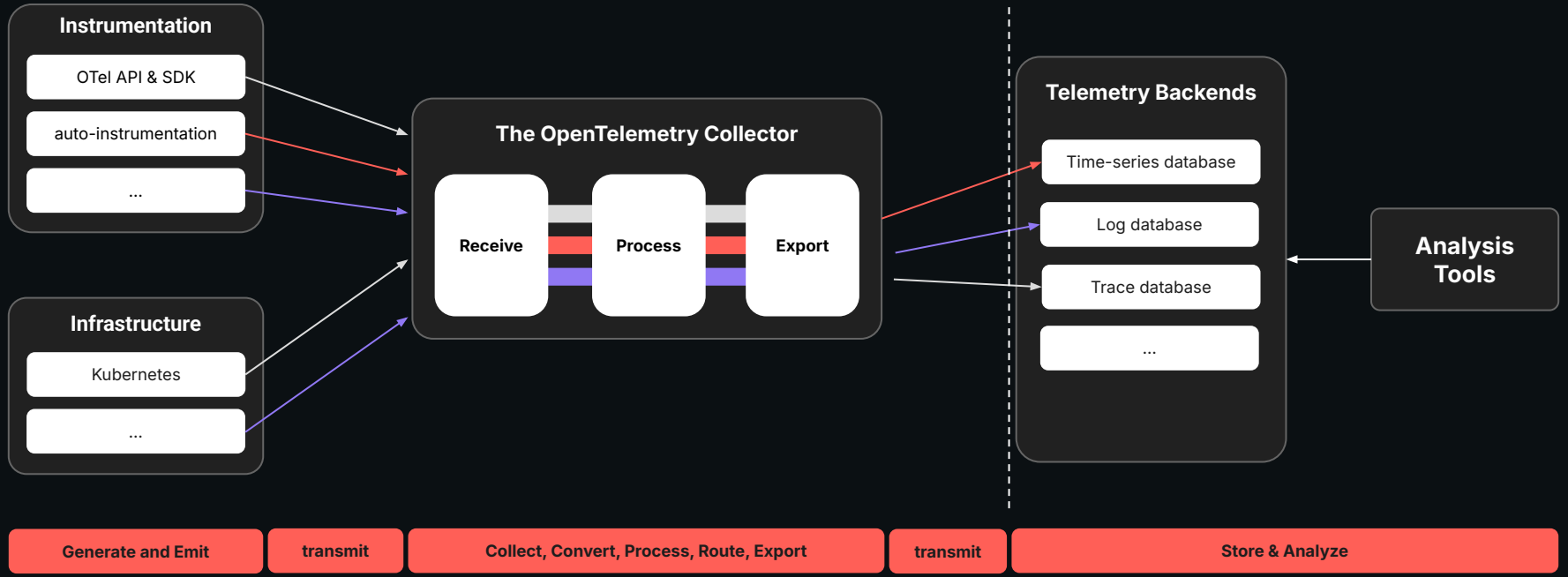
We don’t have a *metrics* problem,
or a *tracing* problem.
We have *systems* problems.

Metrics Logs Traces

Correlation is the superpower



OpenTelemetry: A 1000 miles view



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

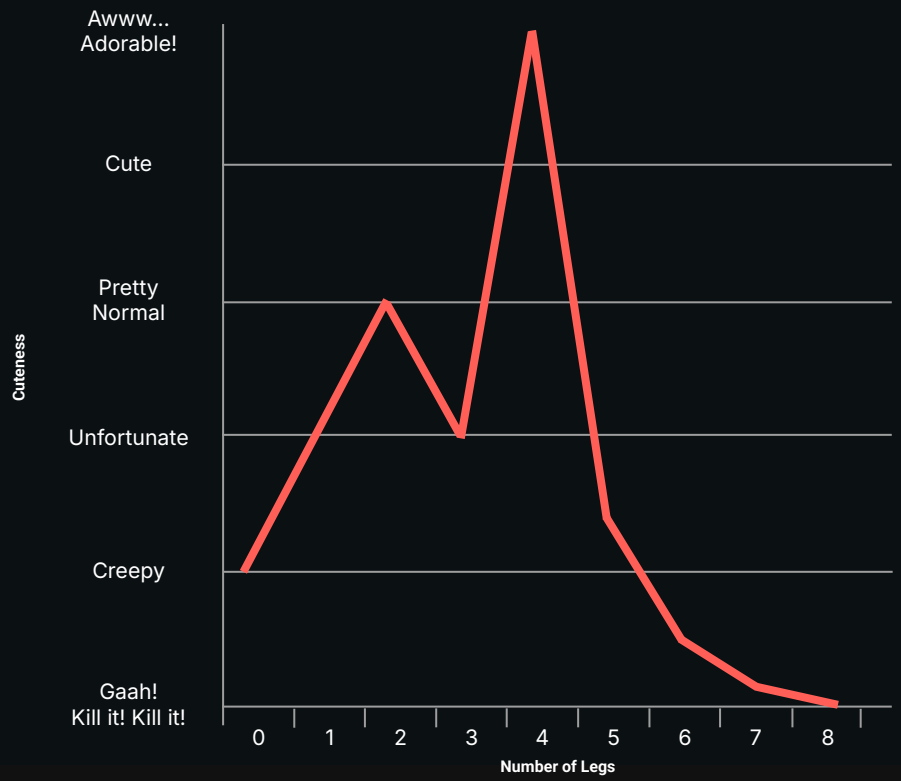
... and many more.

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? …

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**

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

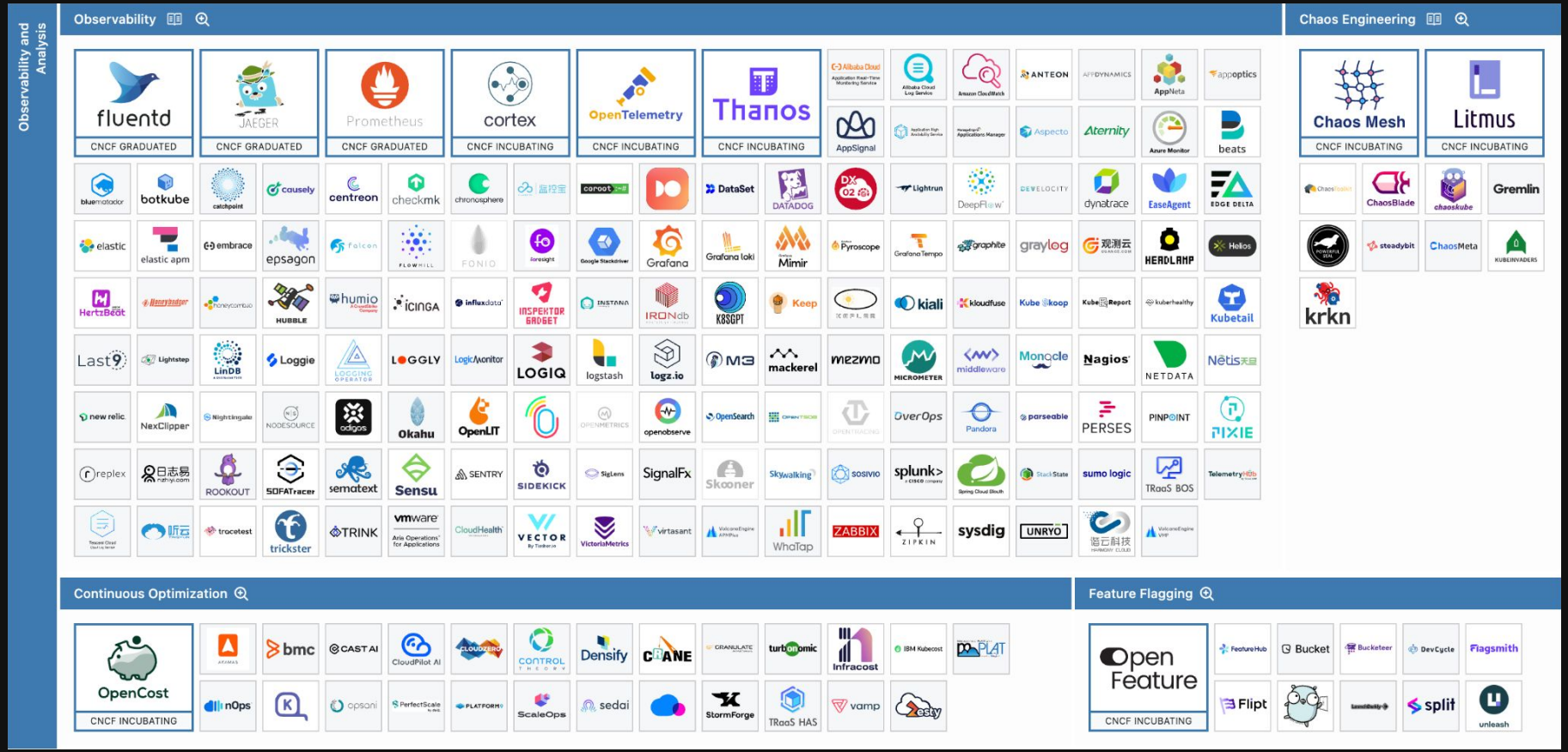
dash0

Observe the platform

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

Enable developers

(traces, metrics, logs, profiling)


Observability and Analysis
Observability
fluentd
CNCF GRADUATED
JAEGER
CNCF GRADUATED
Prometheus
CNCF GRADUATED
cortex
CNCF INCUBATING
OpenTelemetry
CNCF INCUBATING
Thanos
CNCF INCUBATING
bluematador
botkube
catchpoint
causely
centreon
checkmk
chronosphere
coroot
DataSet
DATADOG
Lightrun
DeepFlow
DEVELOCITY
dynatrace
EaseAgent
EDGE DELTA
elastic
elastic apm
embrace
epsagon
Falcon
FLOWMILL
FONIO
Foresight
Google Stackdriver
Grafana
Grafana loki
Mimir
Pyroscope
Grafana Tempo
graphite
graylog
HEADLAMP
Helios
HertzBeat
HoneyBadger
honeycomb.io
HUBBLE
humio
ICINGA
influxdata
INSPEKTOR GADGET
INSTANA
IRONdb
K8SGPT
Keep
kiali
kloudfuse
Kube Skoop
KubeReport
kuberhealthy
Kubetail
Last9
Lightstep
LinDB
Loggie
LOGGLY
LogicMonitor
LOGIQ
logstash
logz.io
M3
mackerel
mezmo
MICROMETER
middleware
Monocle
Nagios
NETDATA
Netis
new relic
NexClipper
Nightingale
NODESOURCE
odigos
Okahu
OpenLIT
OPENMETRICS
openobserve
OpenSearch
OverOps
Pandora
parseable
PERSES
PINPOINT
PIXIE
replex
ROOKOUT
SDFATracer
sematext
Sensu
SENTRY
SIDEKICK
SigLens
SignalFx
Skooner
SkyWalking
SOSIVIO
splunk>
Spring Cloud Sleuth
StackState
sumo logic
TRaaS BOS
trocetest
TRINK
trickster
vmware
Aria Operations for Applications
CloudHealth
VECTOR
VictoriaMetrics
virtasant
WhaTap
ZABBIX
ZIPKIN
sysdig
UNRYO
AppSignal
Aspecto
Aternity
Anure Monitor
beats
Chaos Engineering
Chaos Mesh
CNCF INCUBATING
Litmus
CNCF INCUBATING
ChaosBlade
chaoskube
Gremlin
steadybit
ChaosMeta
KUBEINVADERS
krkn
Continuous Optimization
OpenCost
CNCF INCUBATING
bmc
CAST AI
CloudPilot AI
CLOUDZERO
CONTROL THEORY
Densify
CRANE
turbonomic
Infracost
IBM Kubecost
nOps
opsani
PerfectScale
PLATFORM9
ScaleOps
sedai
StormForge
TRaaS HAS
vamp
Feature Flagging
Open Feature
CNCF INCUBATING
FeatureHub
Bucket
Bucketeer
DevCycle
Flagsmith
Flipt
split
unleash

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

dash0

Based on: “What is observability?” by ubuntu.com

Platform Engineering for Observability

Self-Service Experience

Explicit and Consistent APIs

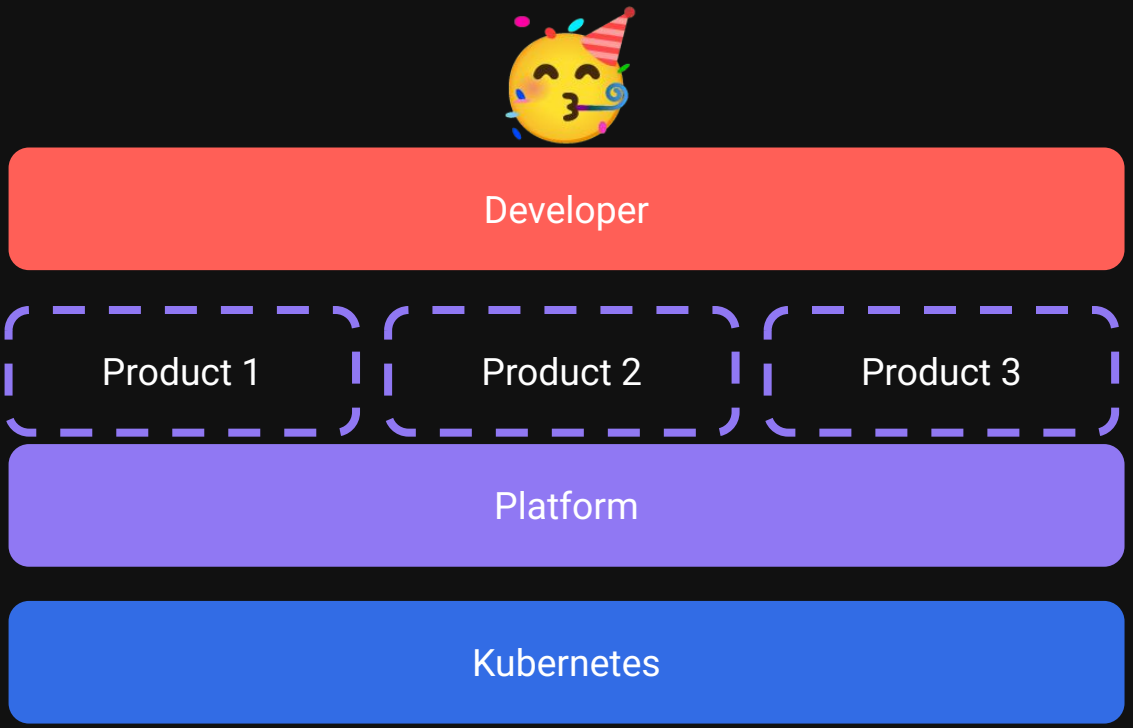
Golden Paths

Modularity

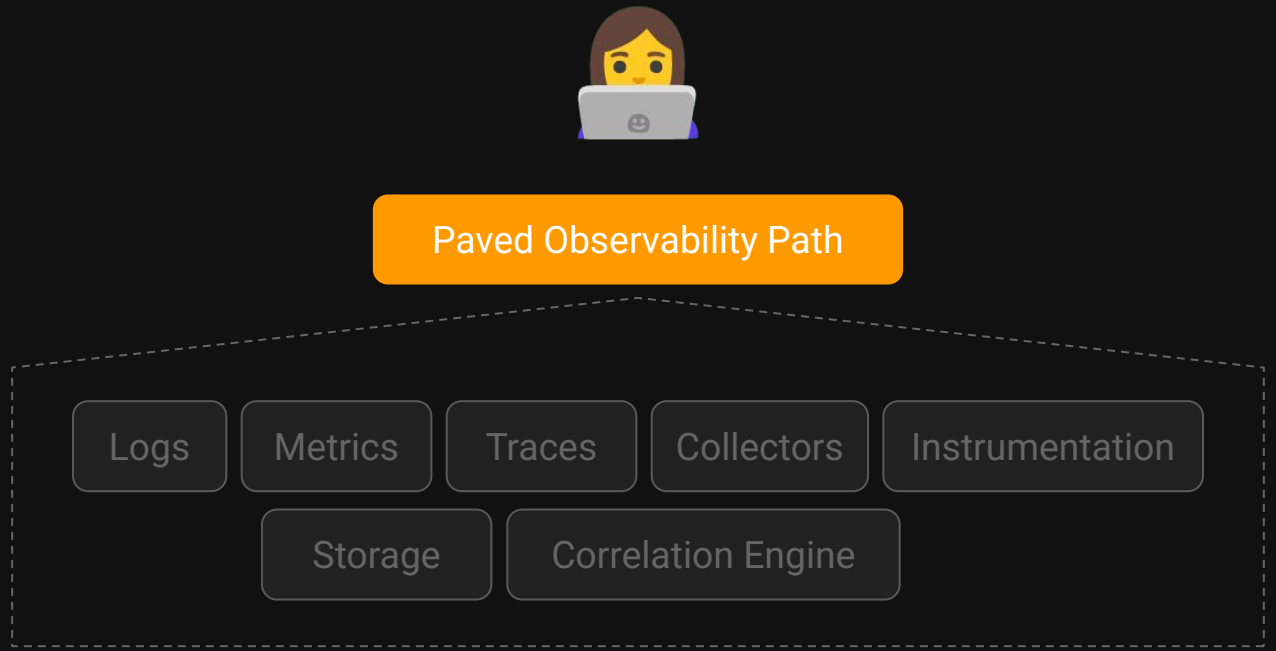
Platform as a Product

Core Requirements

Platform as a Product



Paved Paths for Observability



That’s all great, but I ask again,
how do I do that?

dash0

The answer:

Auto-instrumentation + Operators

=

No-touch Instrumentation

OpenTelemetry Operator

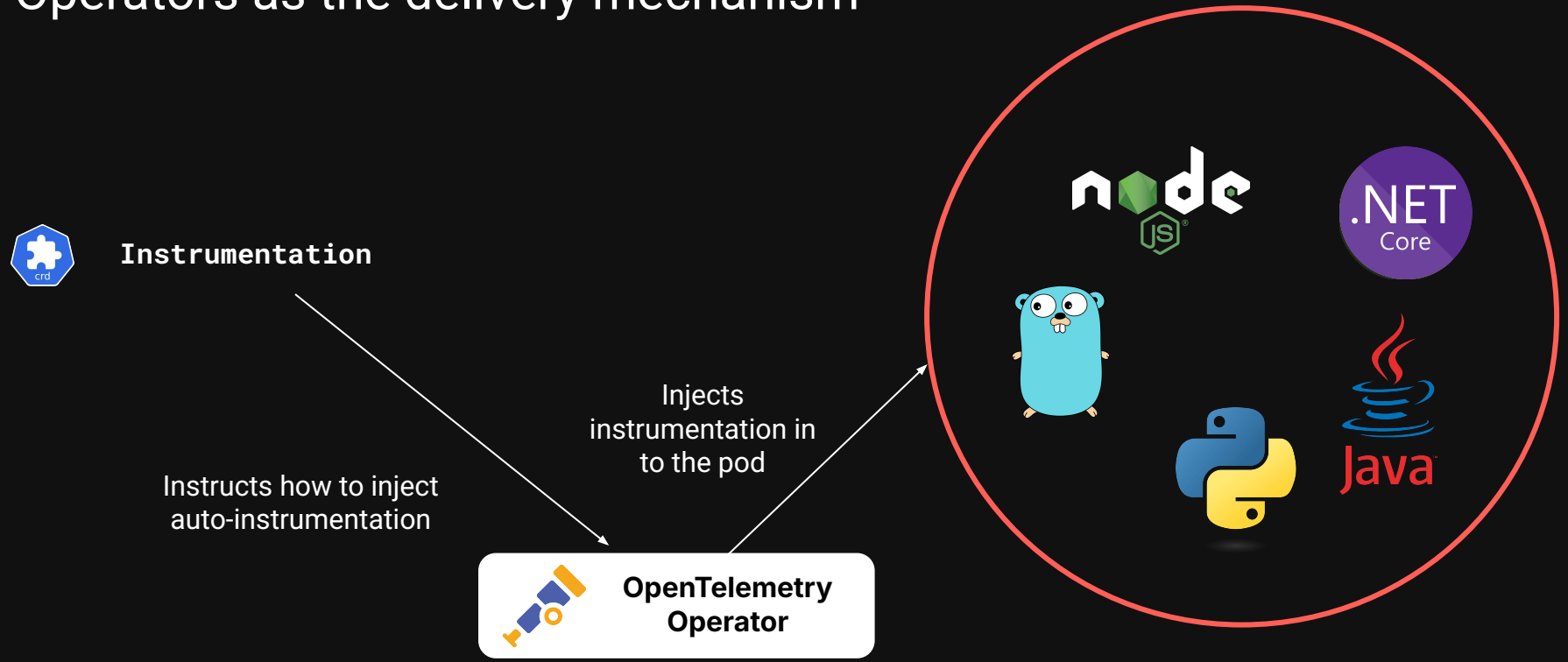
`Instrumentation`

`OpenTelemetryCollector`

`OpAMPBridge`

`TargetAllocator`

Operators as the delivery mechanism



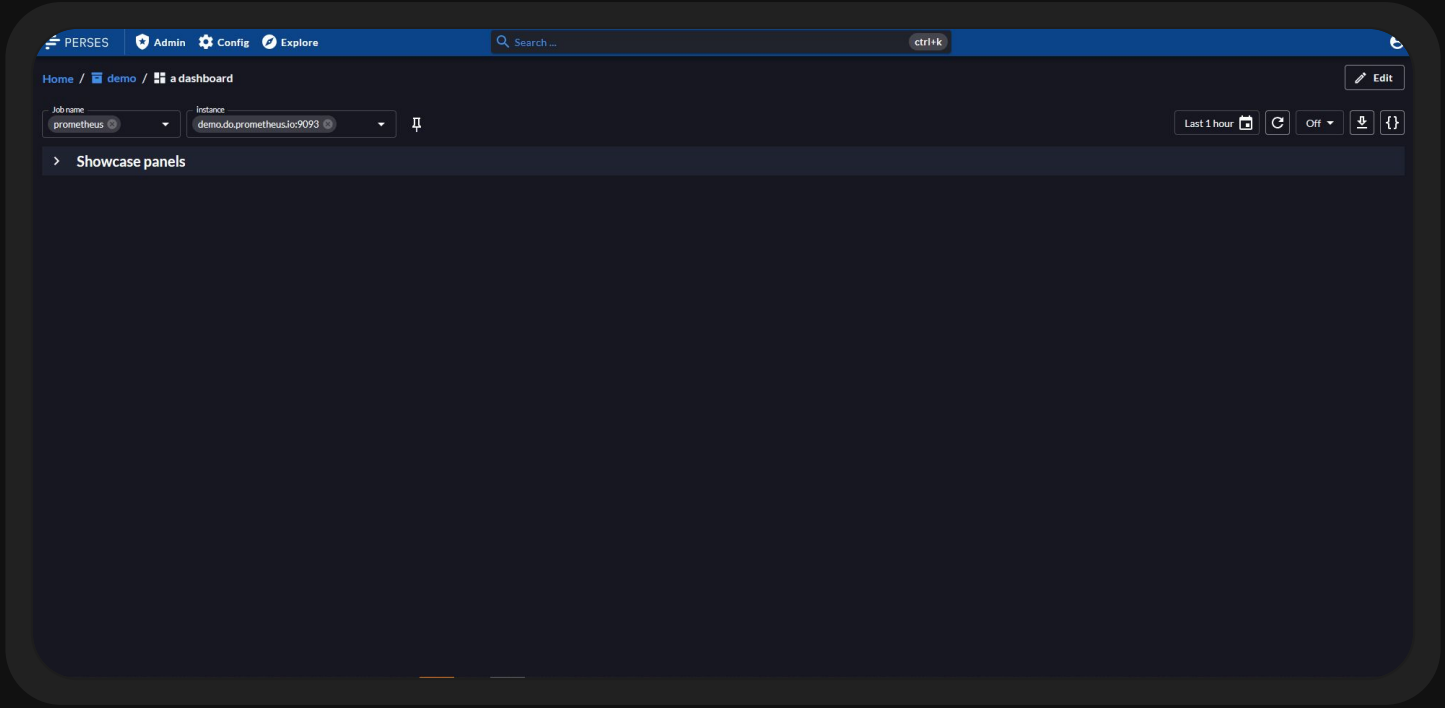
Observability doesn’t stop at instrumentation.

Perses

An open specification for dashboards.

CLOUD NATIVE COMPUTING FOUNDATION

PERSES



dash0

Dashboards as code

PERSES

`Perses`

`PersesDatasource`

`PersesDashboard`

perses-operator

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

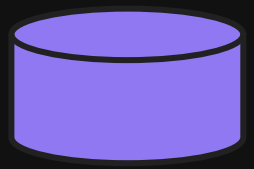
Where telemetry lives and how it’s accessed

A quick word about AI

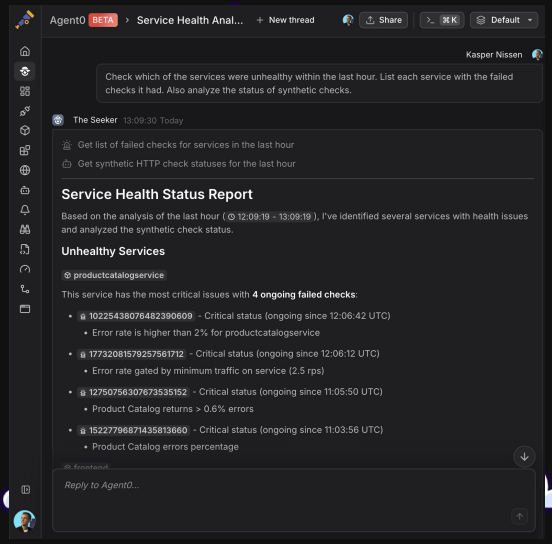
A quick word about AI

OpenTelemetry

Specifications and
Semantic Conventions



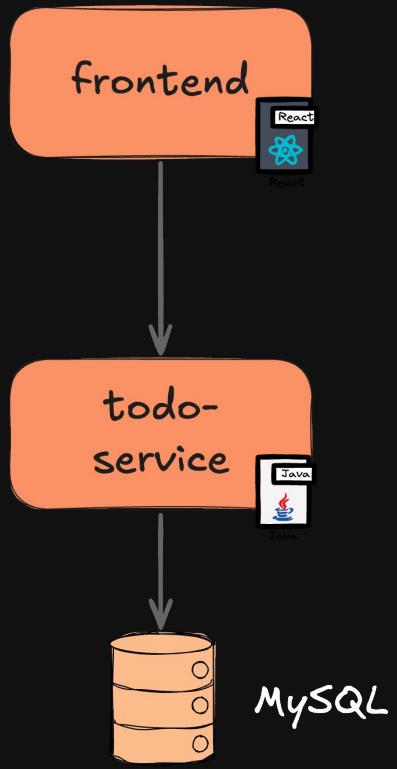
Your Telemetry

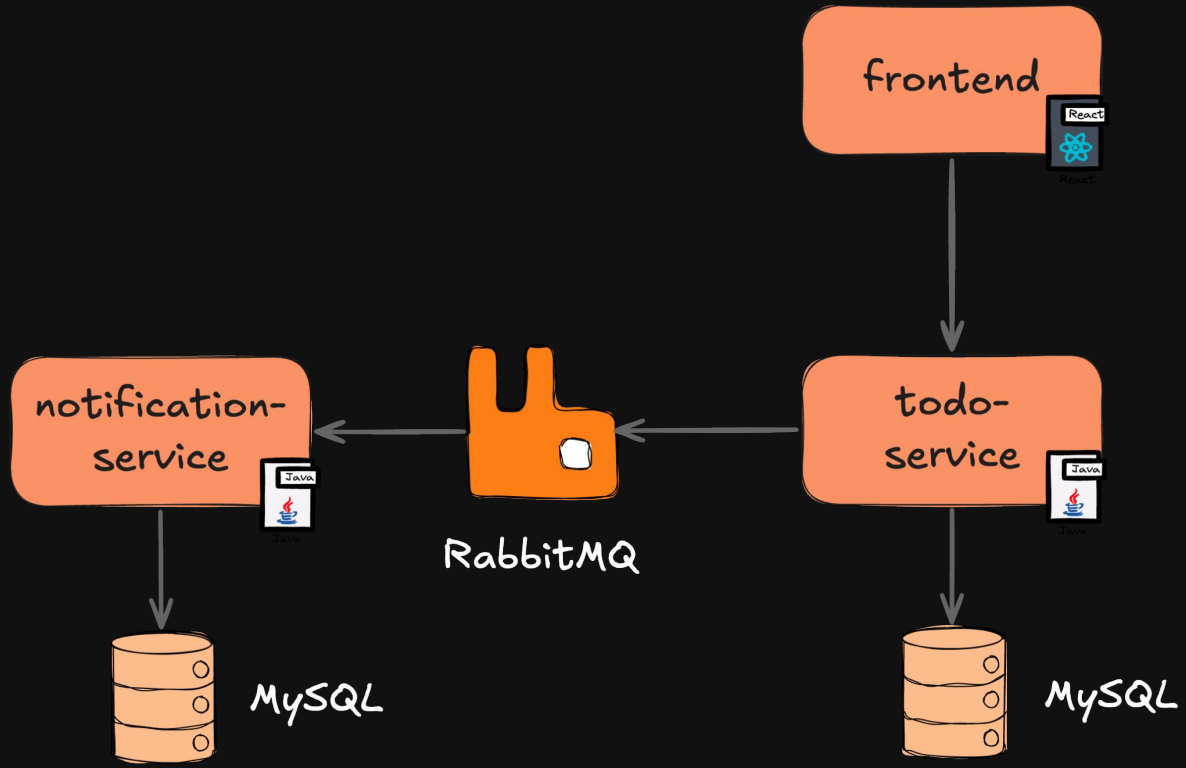


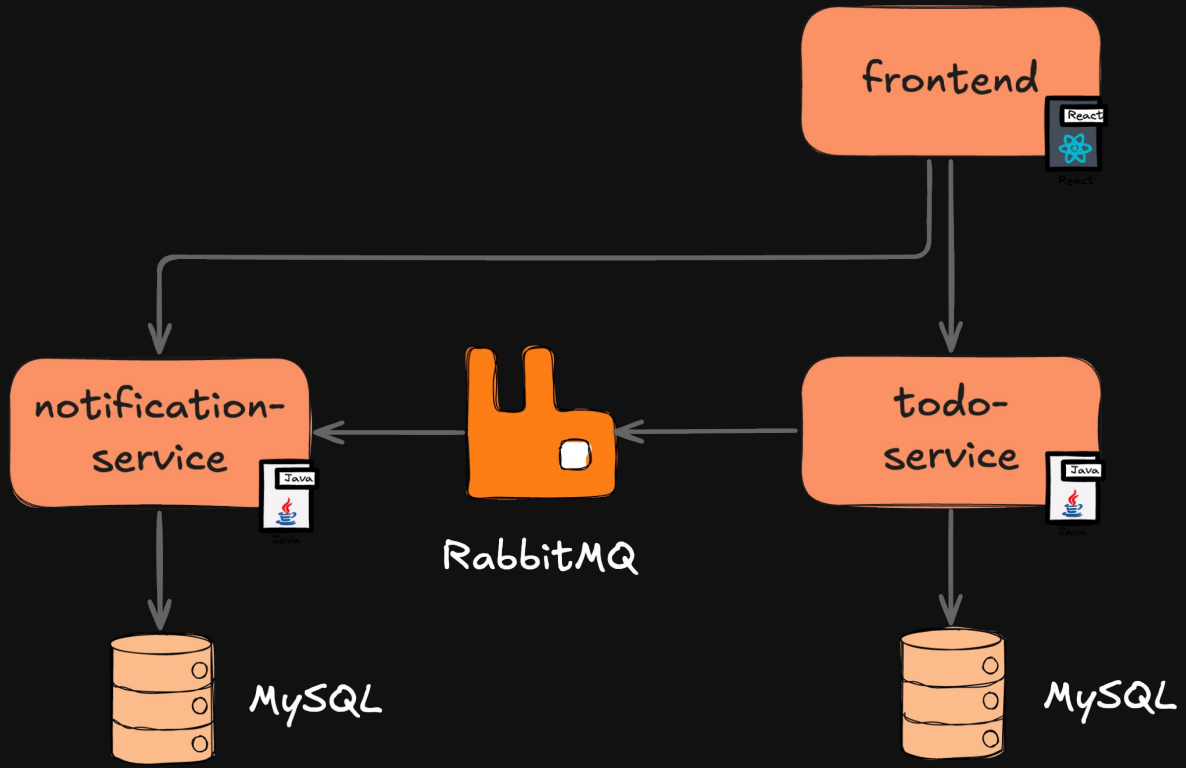
Demo

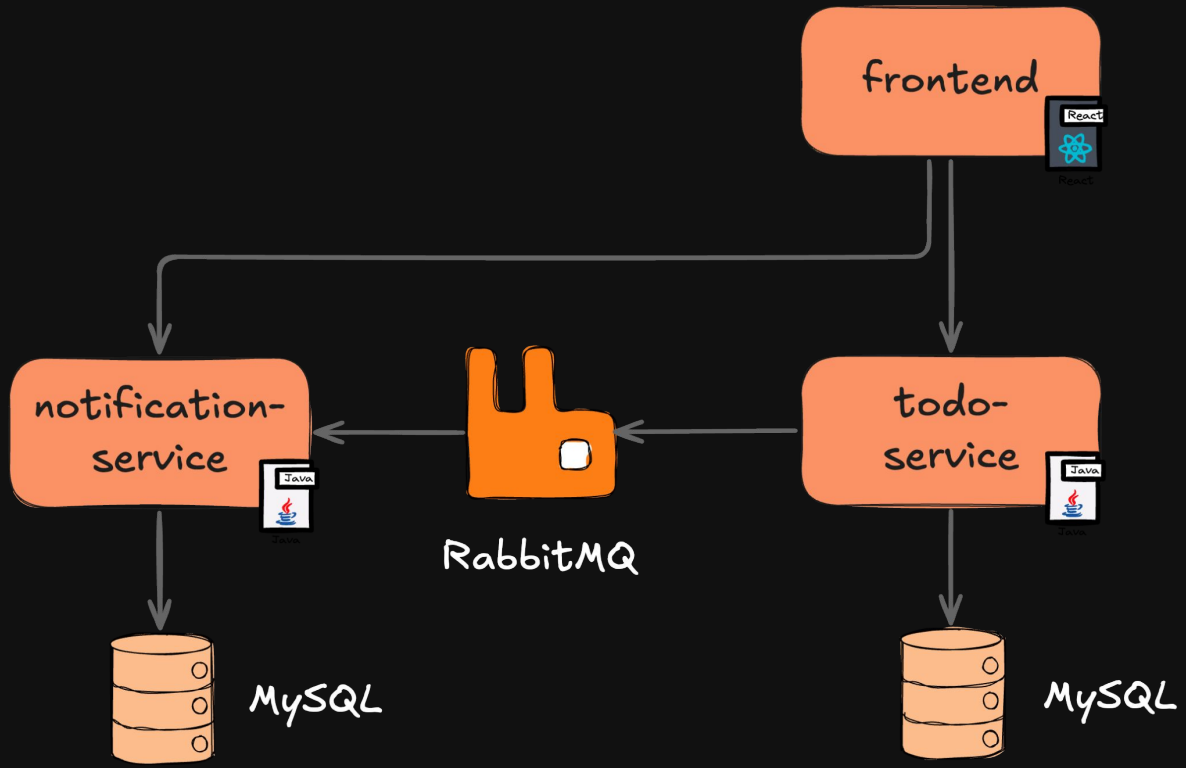
dash0


frontend
React
React


frontend
React
todo-
service
Java
MySQL

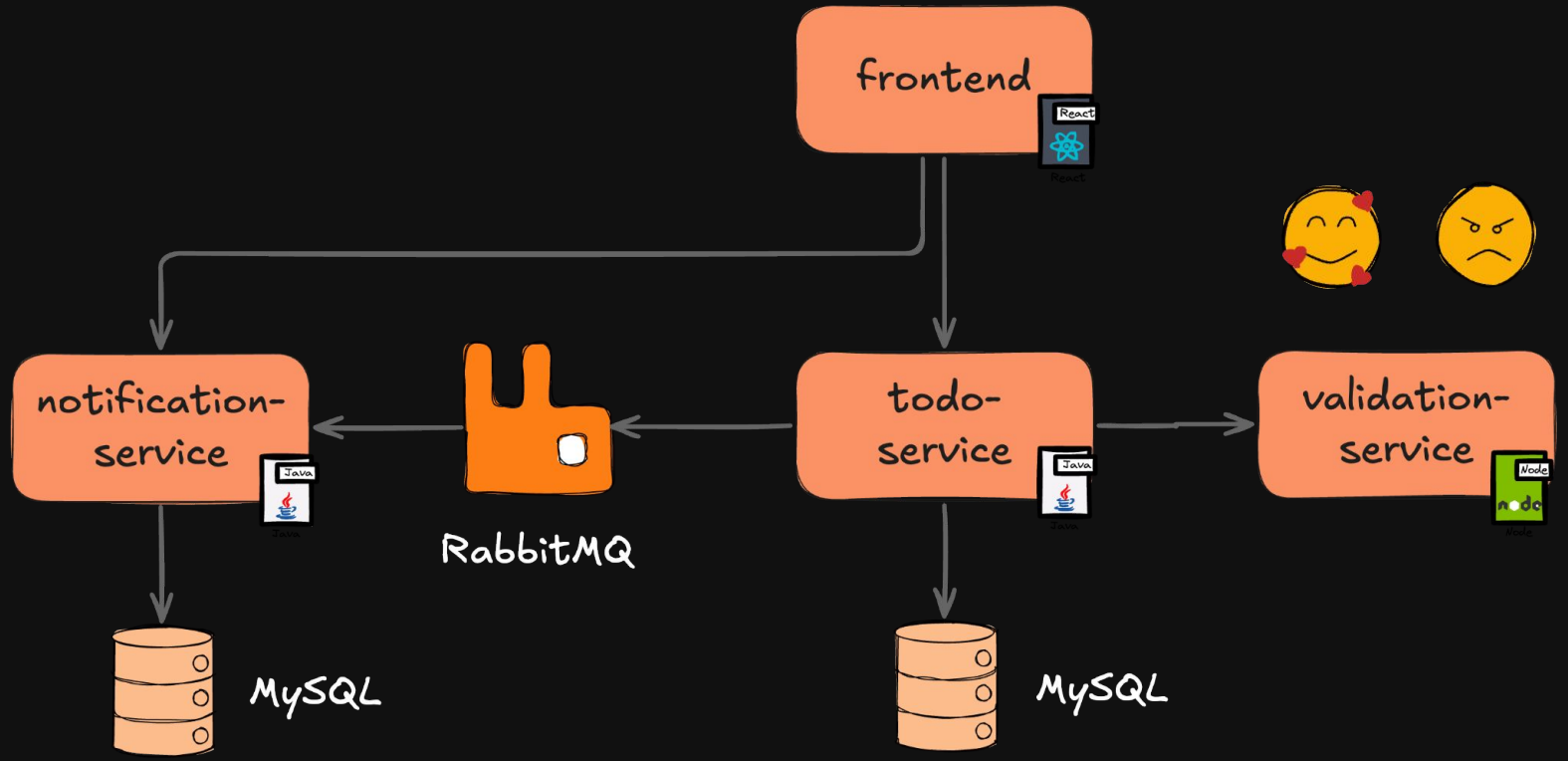

frontend
React
notification-service
Java
RabbitMQ
todo-service
Java
MySQL
MySQL

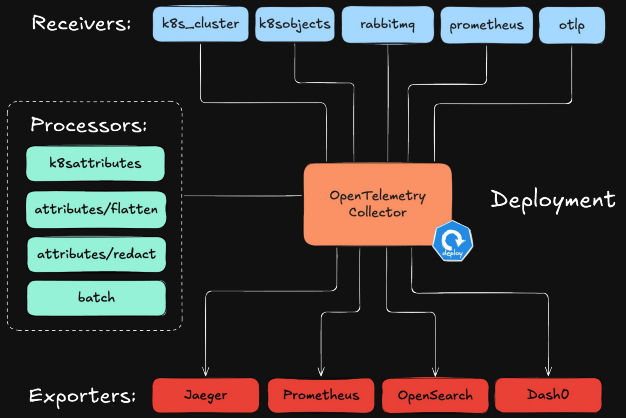

frontend
React
notification-service
Java
RabbitMQ
todo-service
Java
MySQL
MySQL



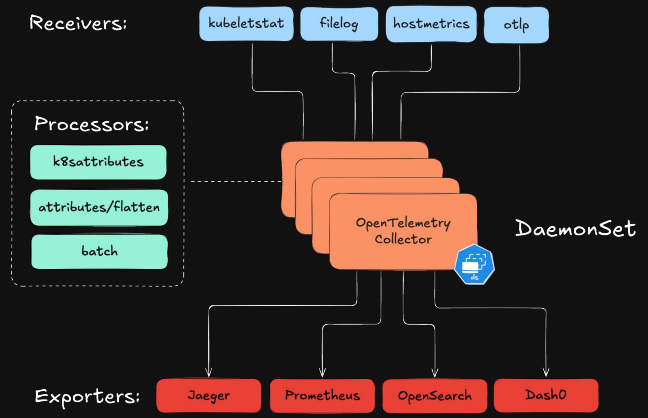
TASK:

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

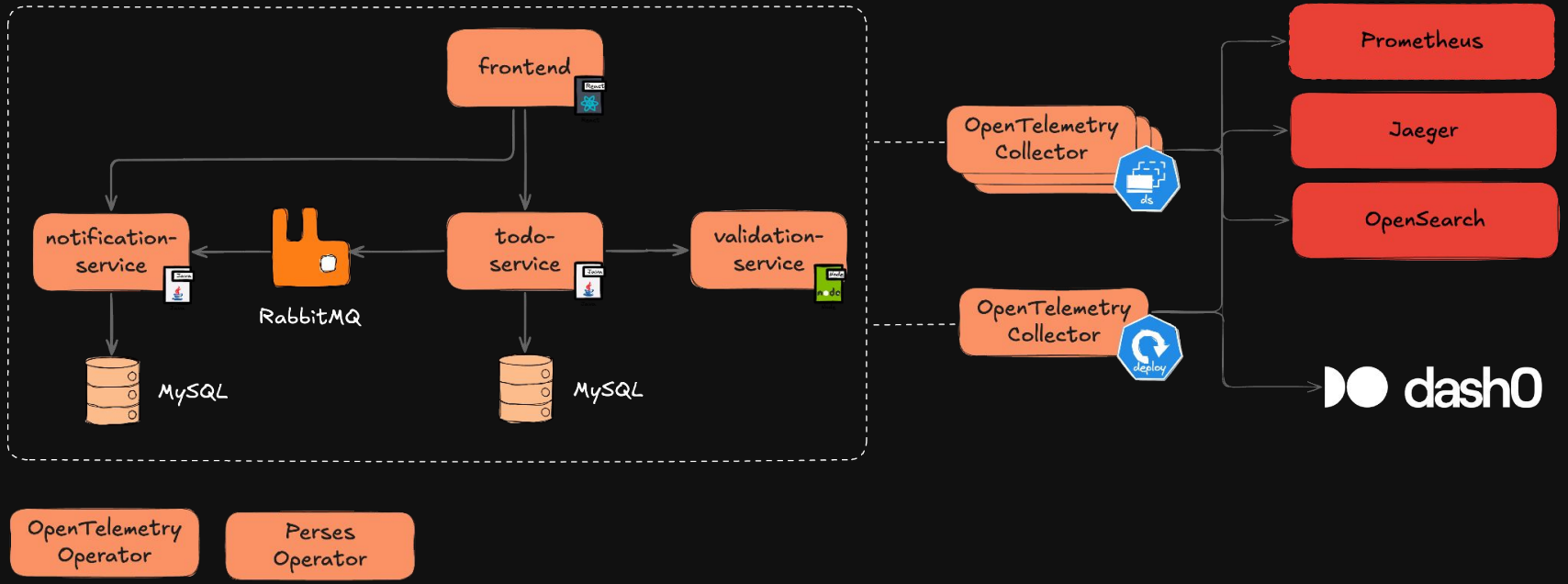


Deployment



DaemonSet

Putting it all together...



Golden defaults for developers

Observability is evolving - fast.

dash0

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.

dash0

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

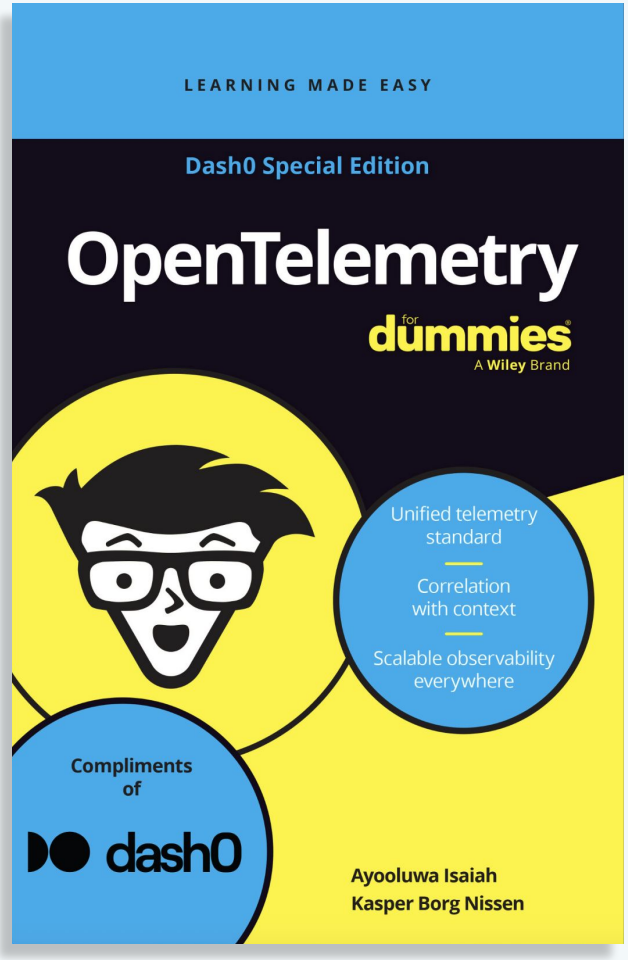
https://university.platformengineering.org/observability-for-platform-engineering

Michele Mancioppi

Head of Product @ Dash0

Kasper Borg Nissen

Developer Relations @ Dash0



Get a free copy!

dash0

Win 1 of 5 Premium Dash0 Backpacks

Thanks for joining us tonight! We will select 5 lucky attendees to win a limited-edition, premium Dash0 backpack. To be in with the chance of winning, all you need to do is complete the form to enter the draw.

First Name*

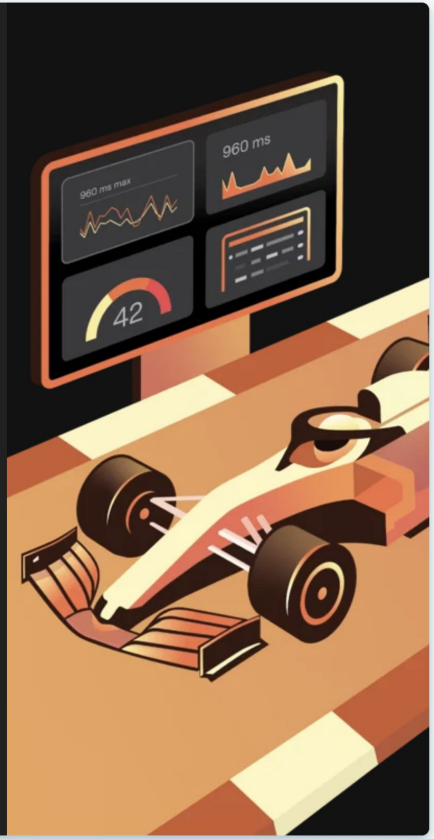
Last Name*

Email*

Company Name*

Job Title

Submit



MERGE FORWARD

CLOUD NATIVE COMPUTING FOUNDATION

Follow us on LinkedIn

Merge Forward (CNCF)

Building a stronger open source future together!

Learn more

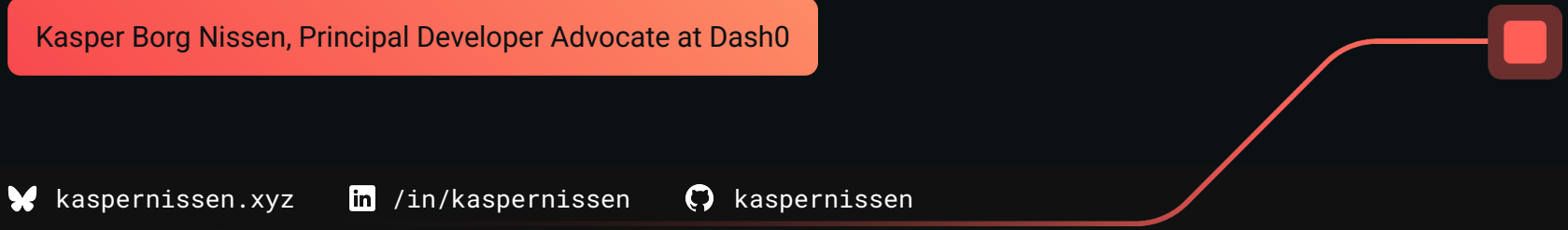
#merge-forward on Slack!

community.cncf.io/merge-forward

dash0

Thank you!

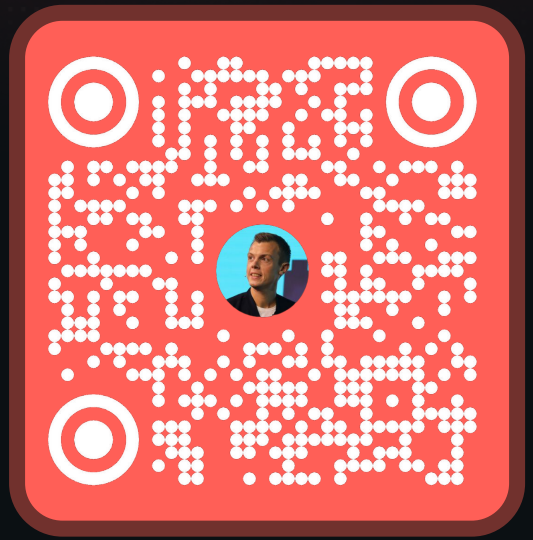
Kasper Borg Nissen, Principal Developer Advocate at Dash0



kaspernissen.xyz /in/kaspernissen kaspernissen

dash0

Get in touch!



kaspernissen.xyz /in/kaspernissen kaspernissen