Observability Data Centralization and Tracing Implementation

CERN openlab Summer Student Programme Lightning talks

Presenter:
Rishab Kumar Jha

Supervisor(s):
Daniel Gonçalves Portovedo, Antonio Nappi

The problem

Lack of centralized observability and missing support for tracing.

Logs ✓

Already centralized. I wired them into the shared dashboard.

`Fluent Bit → OpenSearch`

Metrics ✓

Already centralized. Correlated with logs in one RED dashboard.

`Prometheus → Mimir`

Traces ✗

Missing. No way to follow one request across services and clusters.

`most of my project`

✗ Centralized observability

Missing. Observability data live in separate tools. Very Hard to correlate them.

Getting traces out of an app

Auto Instrumentation and OpenTelemetry SDK

INFRA SIDE · SET IT UP ONCE, CENTRALLY

Auto-instrumentation

The operator injects basic tracing code from one pod annotation, with no code change.

Cannot name business steps. Generates more data than we need.

HANDOVER

DEVELOPER SIDE · OPT IN

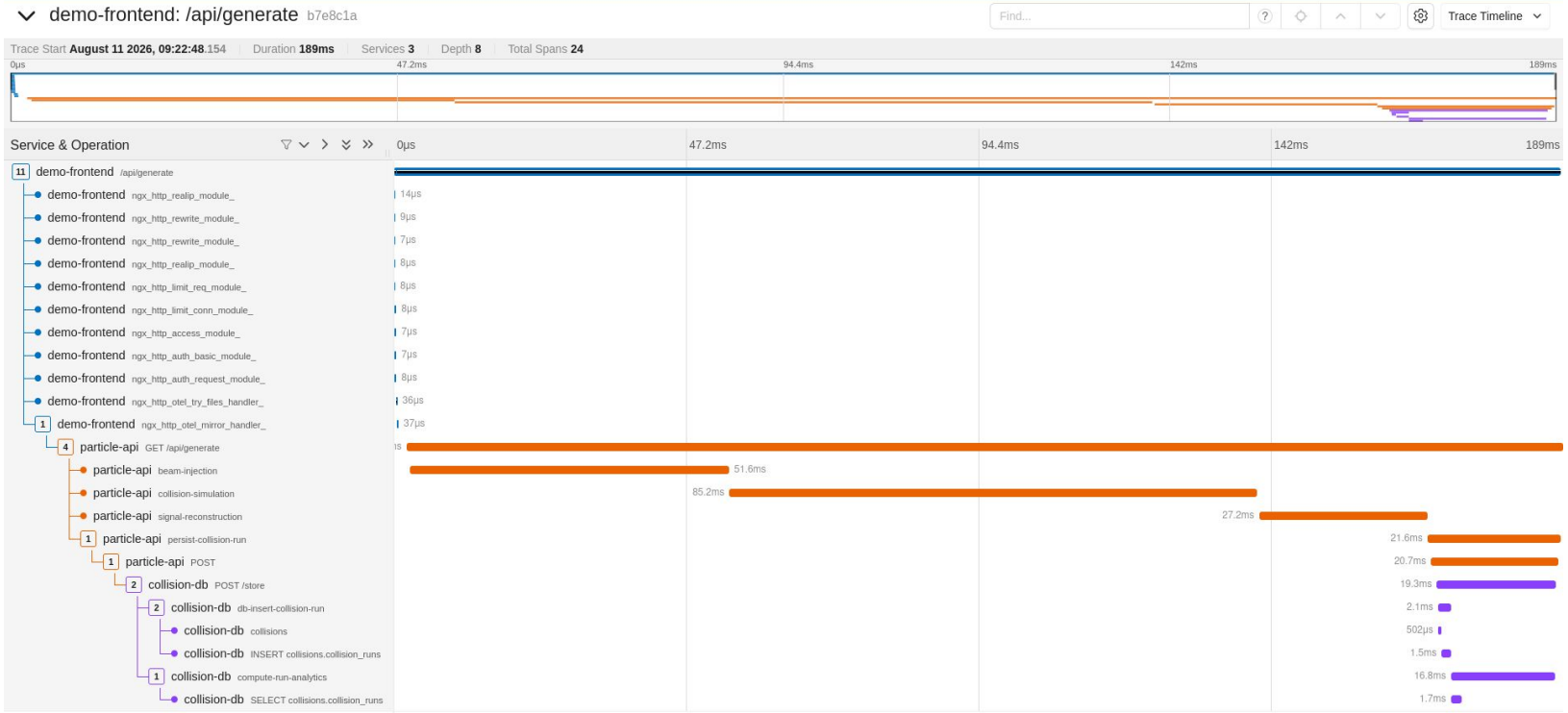
OpenTelemetry SDK

The team writes tracing logic in its own code, so traces carry real business context.

Maintenance burden.

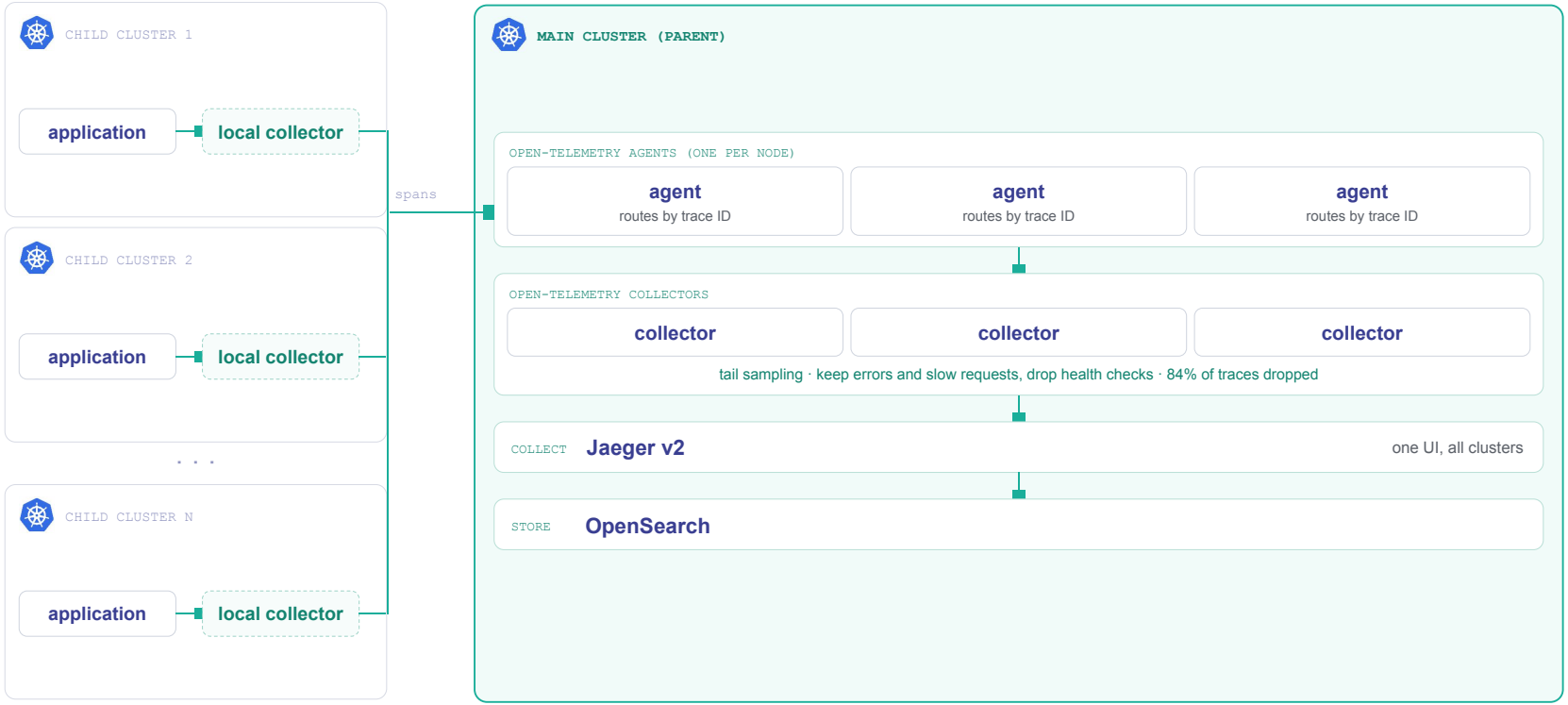
Tracing Demo

One trace ID. Three Clusters. Twenty Four Spans



The Architecture

Every cluster forwards to one place. Spans are routed by trace ID so sampling works at scale.



Evaluation and production readiness

Comparing the tools and testing whether it holds up in prod

Resource Usage

- Memory ~11 MB per app
- CPU <0.1 core per app
- Storage 45–150 MB/day

Reliability

- Async on every hop
- Best-effort, buffered
- App never blocks

Self-monitoring

- Collects its own metrics
- Grafana dashboard
- Alerts on drops, queues

Security path

- mTLS on every hop
- SSO Proxy in front of Jaeger
- Sensitive data stripped centrally

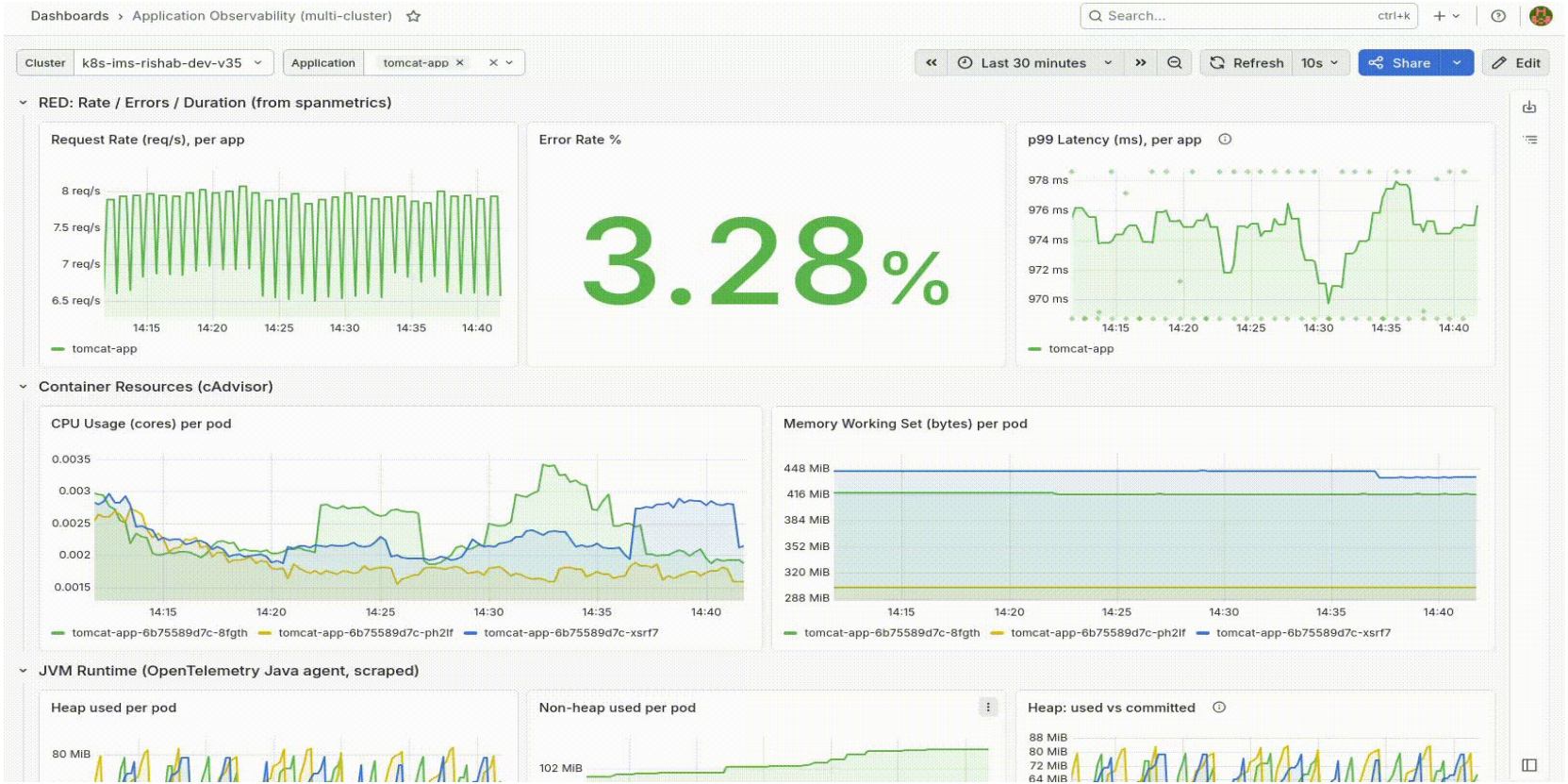
Tools comparisons

- Logs: Fluent Bit > OTel
- Metrics: Prometheus > OTel
- Traces: OTel wins

HAProxy 3.4

- Native OTel filter compiled
- Front door joins the trace
- Dynamically turn tracing on/off

Everything in one dashboard



Questions?

kumarjharishab@gmail.com