FEB 11–12, 2026
CONTAINER days LONDON

Golden Paths for Async Workflows:

Dapr Meets OpenTelemetry

Mauricio "Salaboy" Salatino, Ecosystem Engineer at Diagrid.io
Kasper Borg Nissen, Principal Developer Advocate at Dash0

Diagrid dash0

Who?

Mauricio Salatino

Ecosystem Engineer &
Passionate about Open Source
Platform Engineering on
Kubernetes Author

Kasper Borg Nissen

Principal Developer Advocate at Dash0
Former KubeCon Co-Chair NA/EU
CNCF Ambassador
Golden Kubestronaut
CNCG Aarhus
Cloud Native Nordics/Denmark

dash0 Diagrid

TLTW (too long to watch)

If your applications and infrastructure grows in complexity you must have the right tools to understand what is going on at all times.

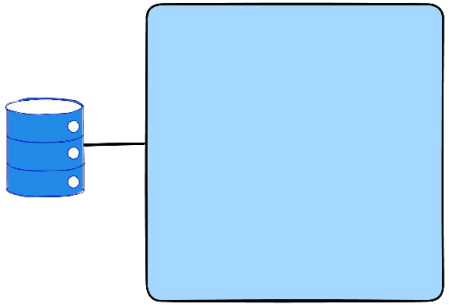
Diagrid dash0

Part 1

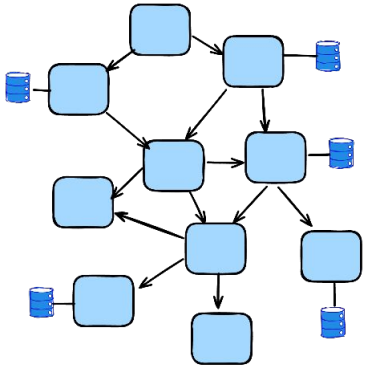
Why Async is Powerful

Diagrid dash0

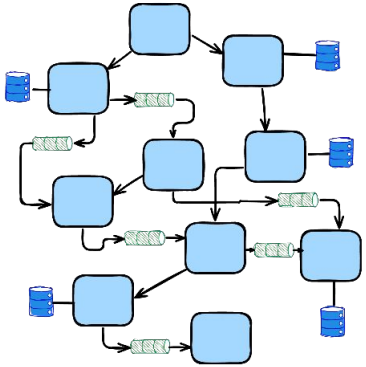
The Rise of Async Microservices



Monoliths

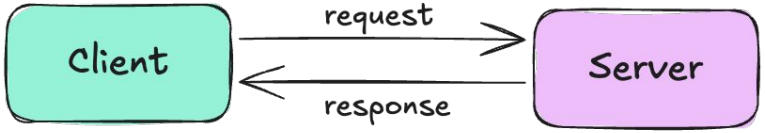


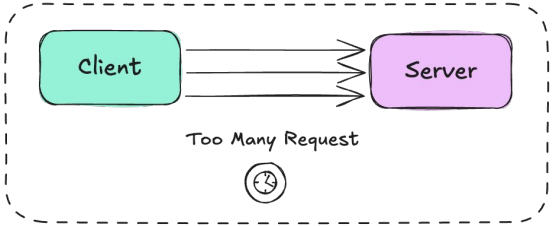
Microservices



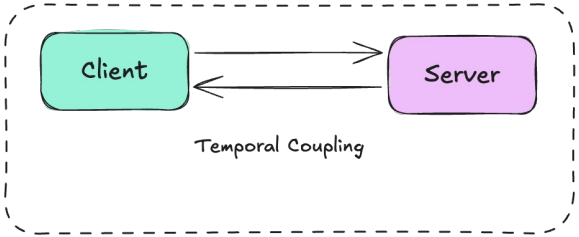
Event-driven Microservices

Limitations with synchronous communication

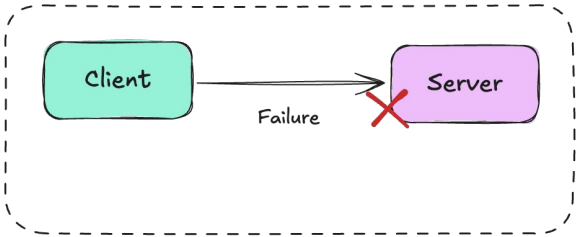




Increased Latency



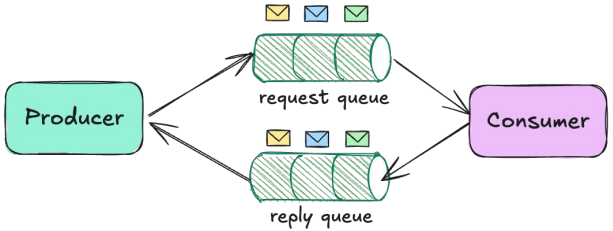
Tight Coupling



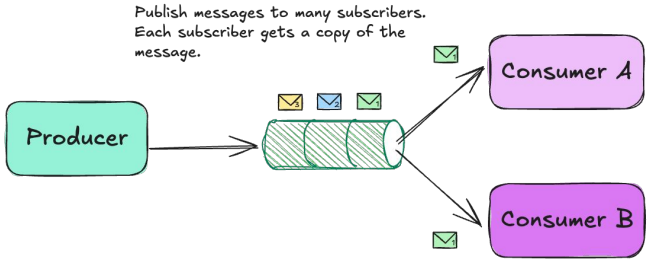
Downtime

Async?

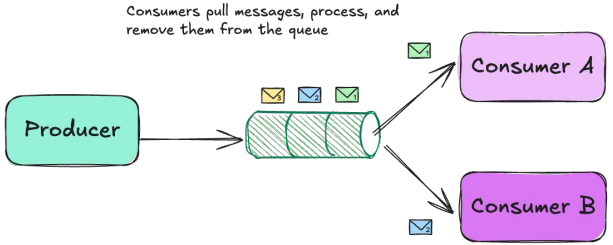
Request Reply Pattern



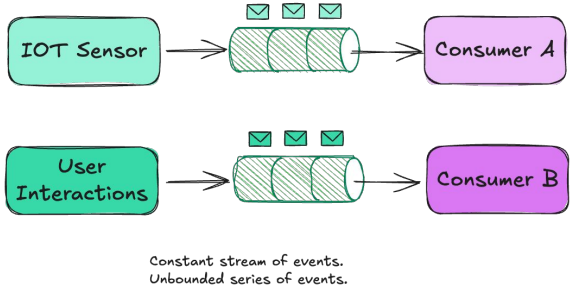
Publish/Subscribe



Queues



Streams



dash0 Diagrid

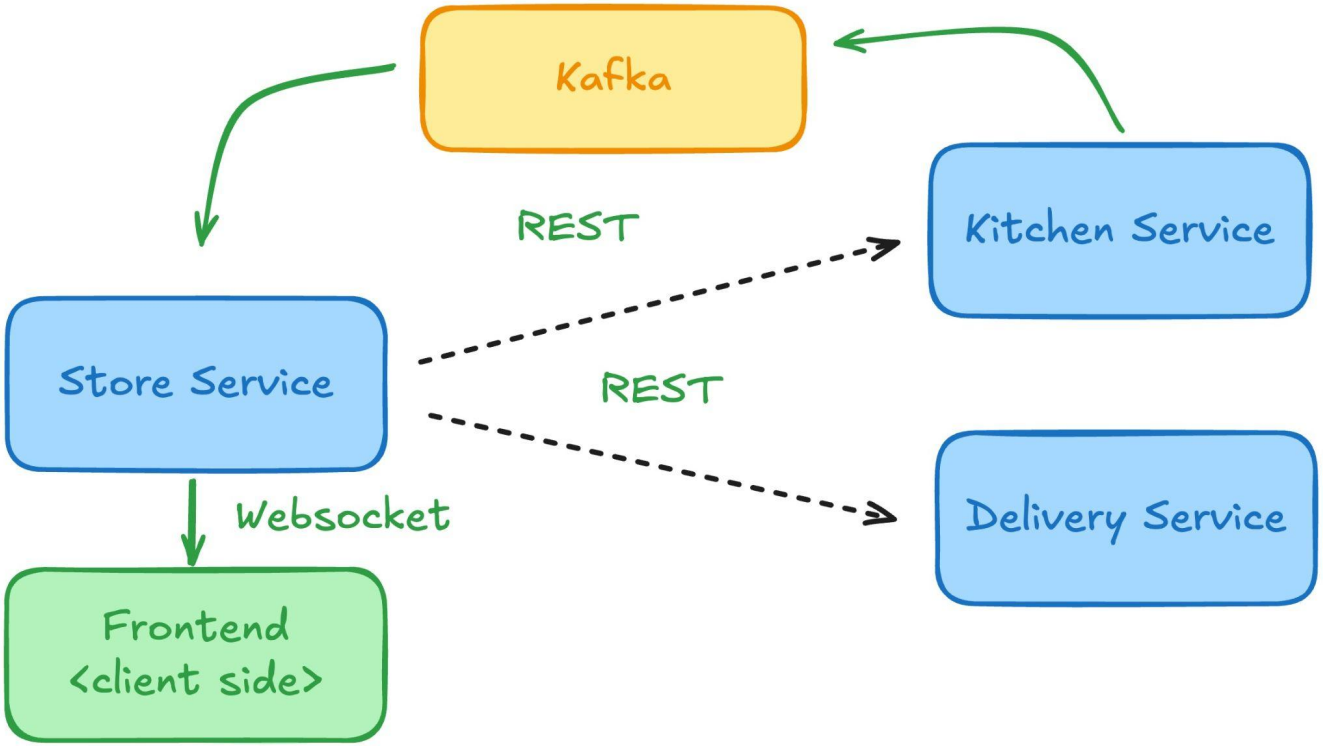
Synchronous communication is familiar, Asynchronous communication is powerful, and in real systems you need both to work seamlessly together.

Demo 1:

Pizza ordering app

Diagrid dash0

Recap



Part 2:

The Golden Path – Dapr + OpenTelemetry

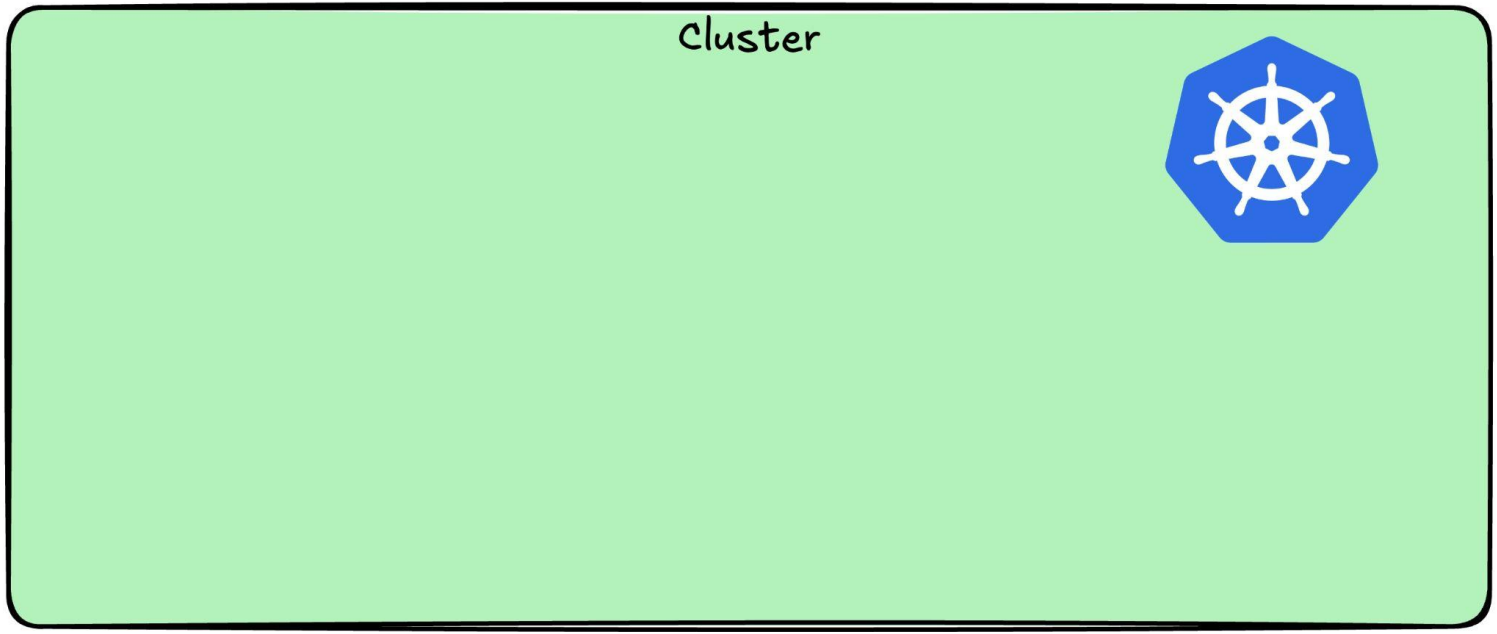
Diagrid dash0

The CNCF Opportunity: Shared Abstractions

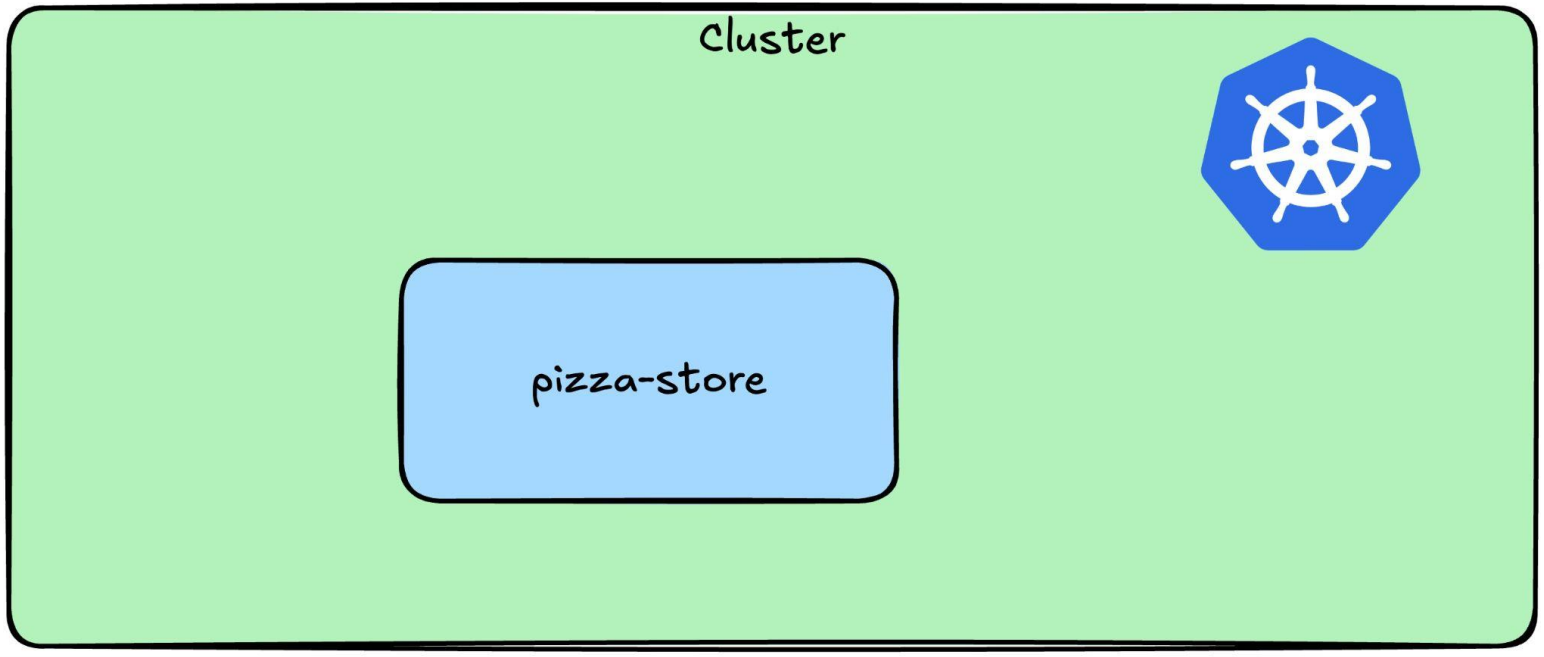
Dapr Building Blocks

- APIs to help developers build scalable and resilient distributed applications
- PubSub, Workflows, Secrets/Configs, Conversation (LLMs), etc
- All these APIs, behind the covers, implement cross-cutting concerns
  - Security
  - Resilience
  - ***Observability***

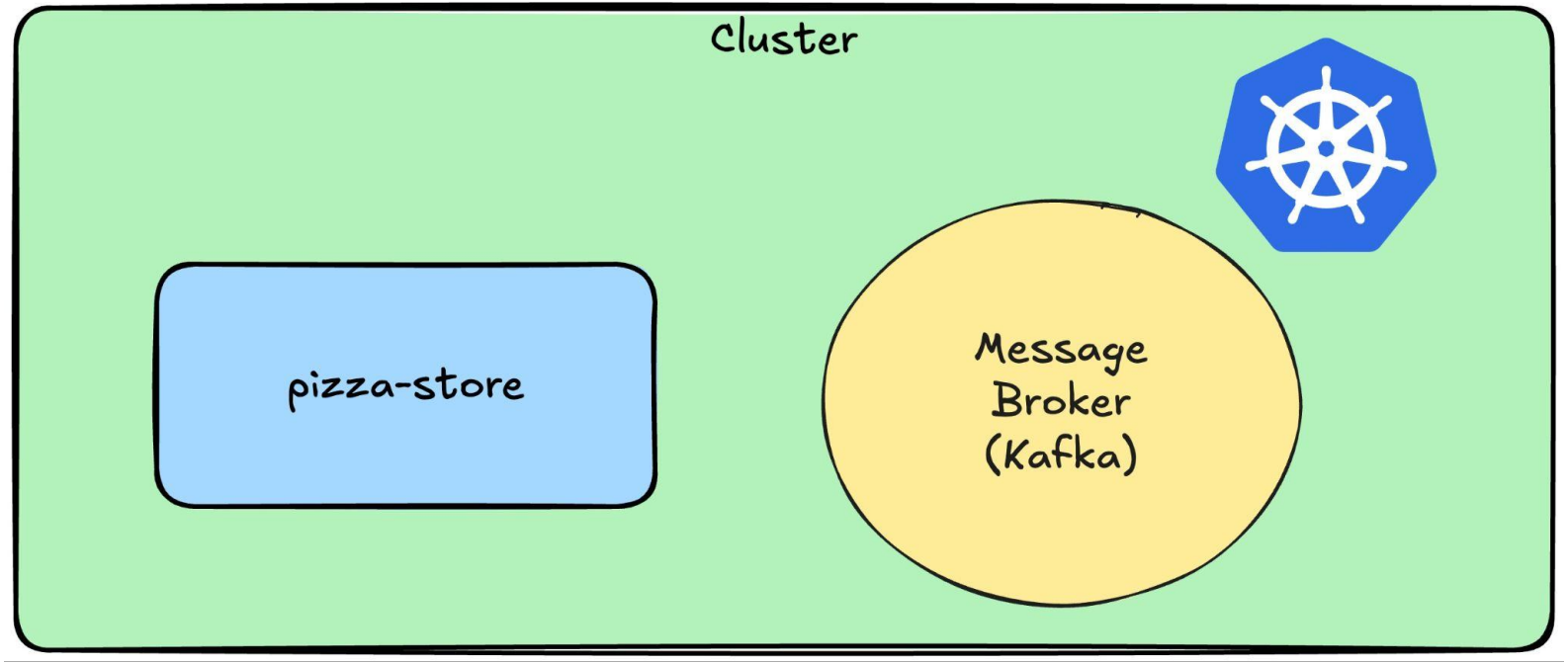
How does it work?



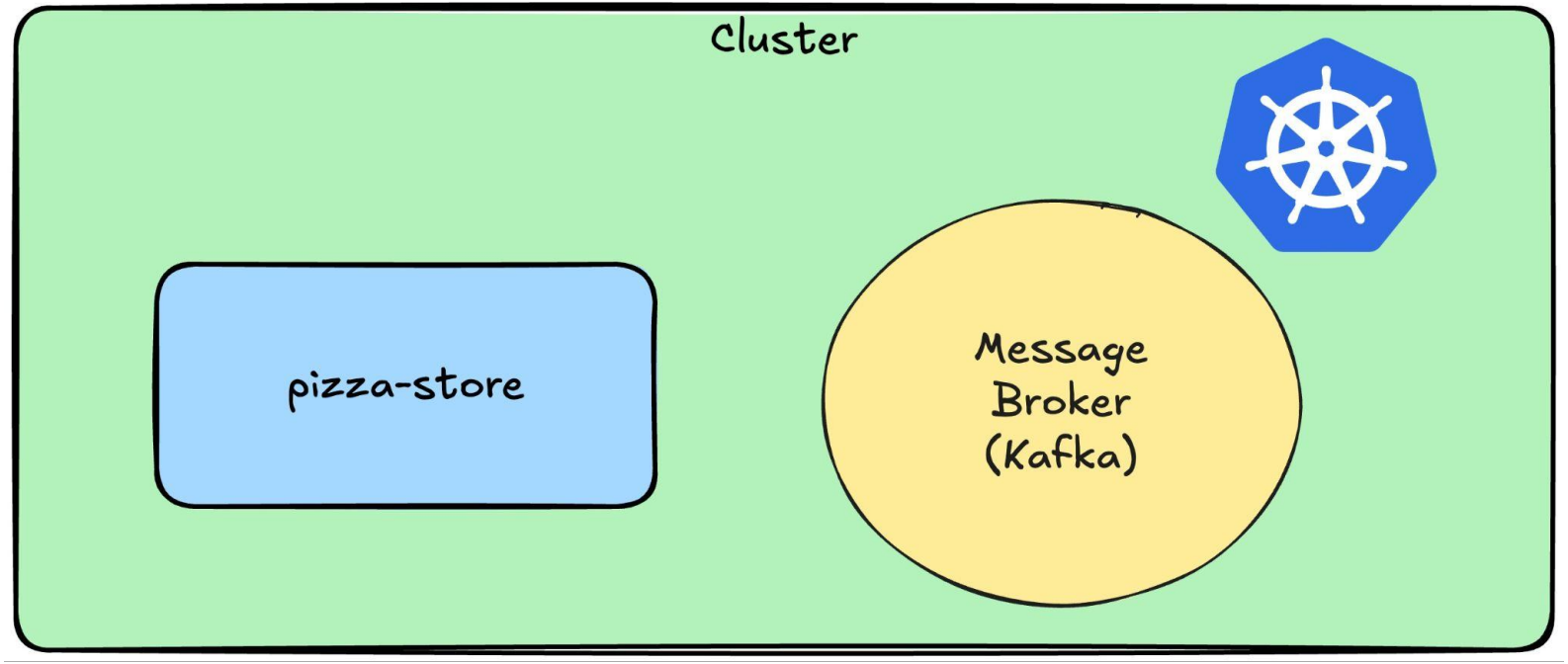
How does it work?



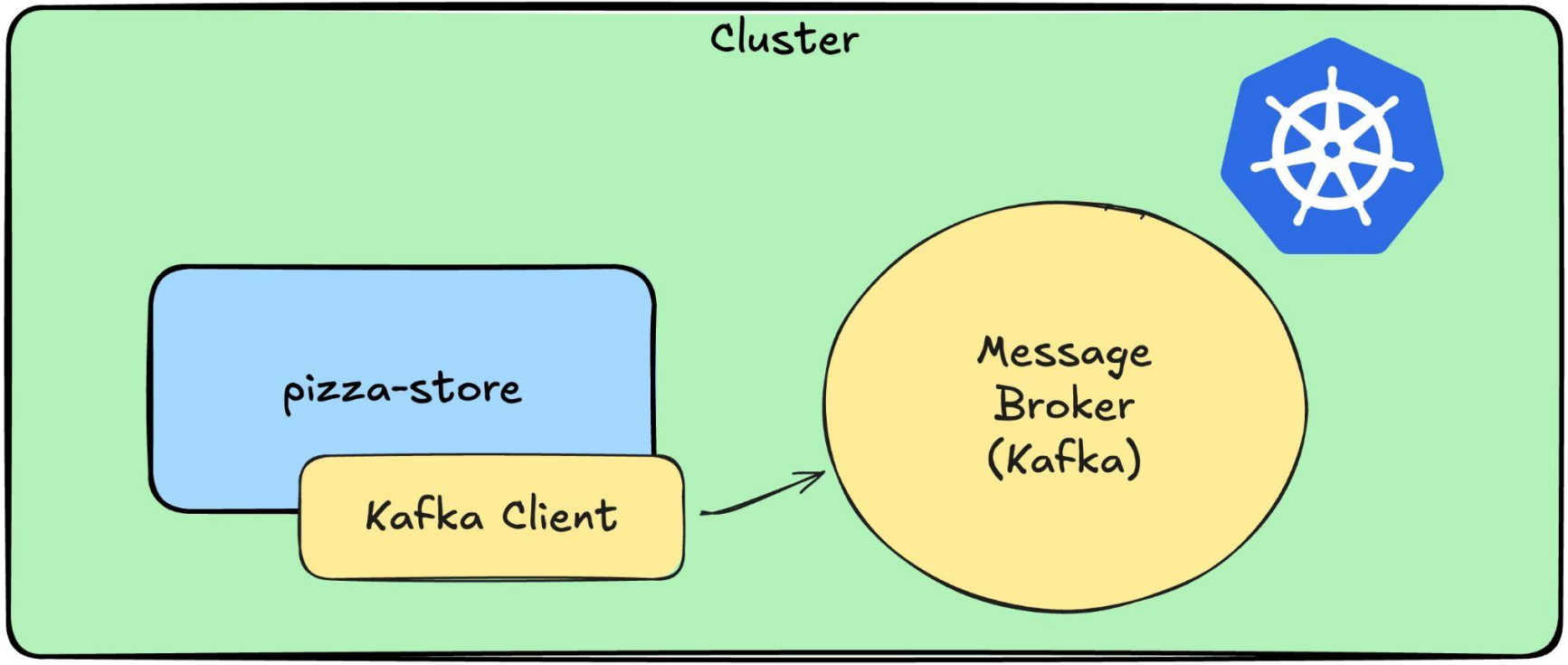
How does it work?



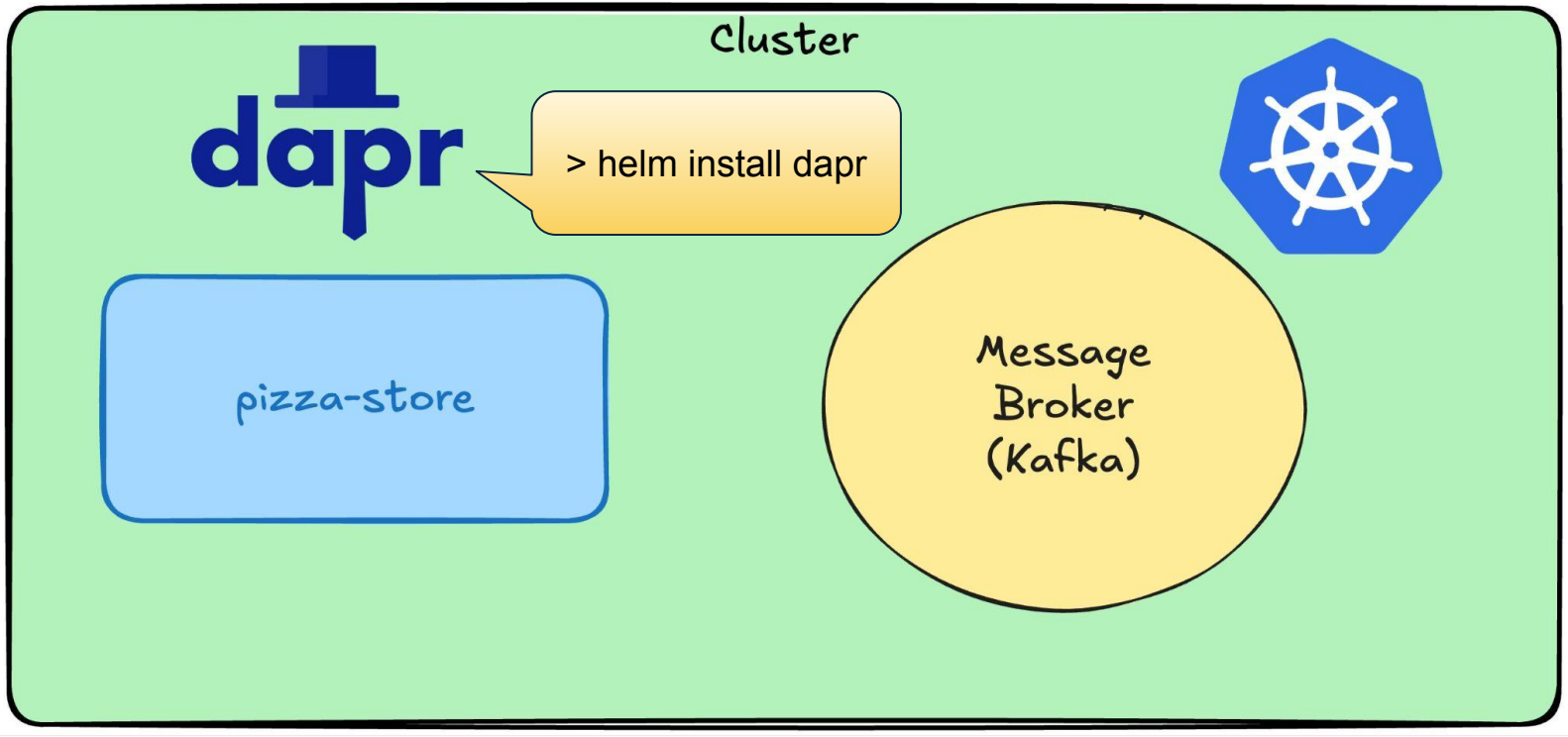
How does it work?



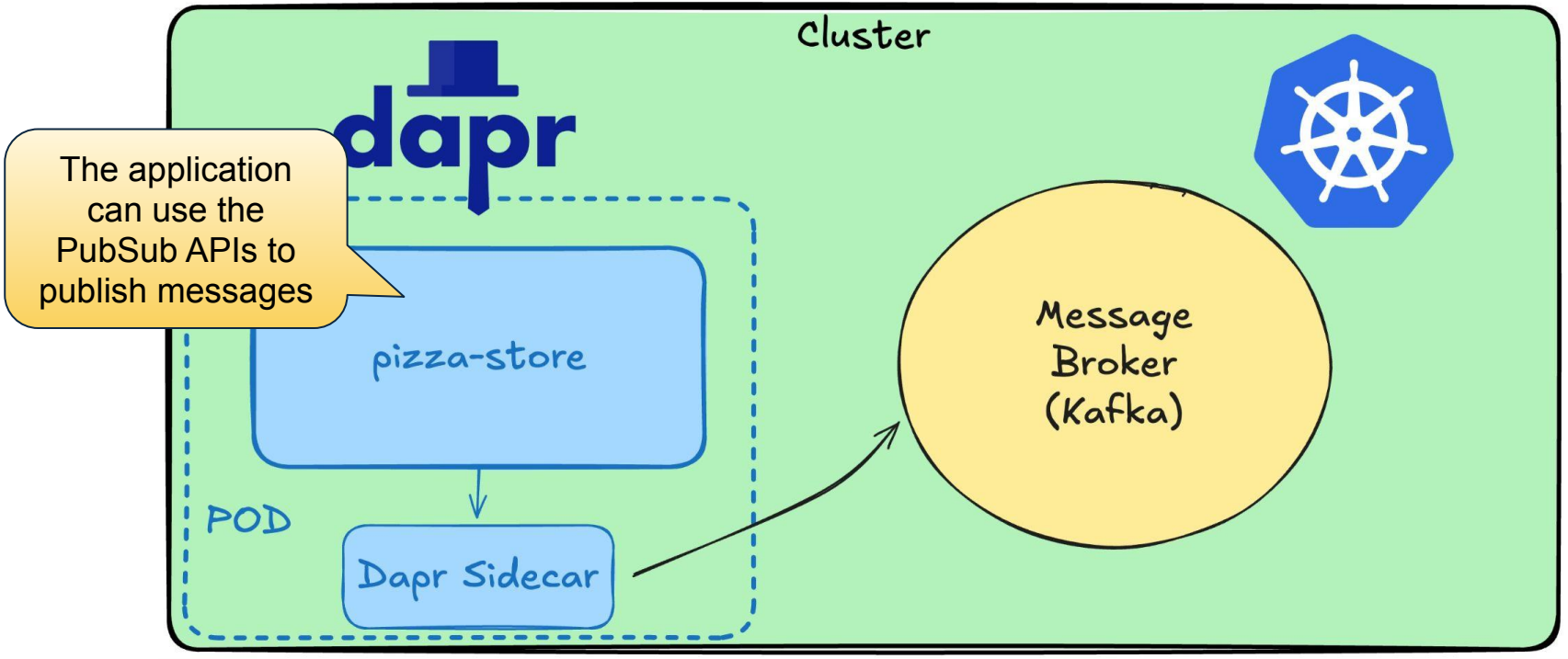
How does it work?



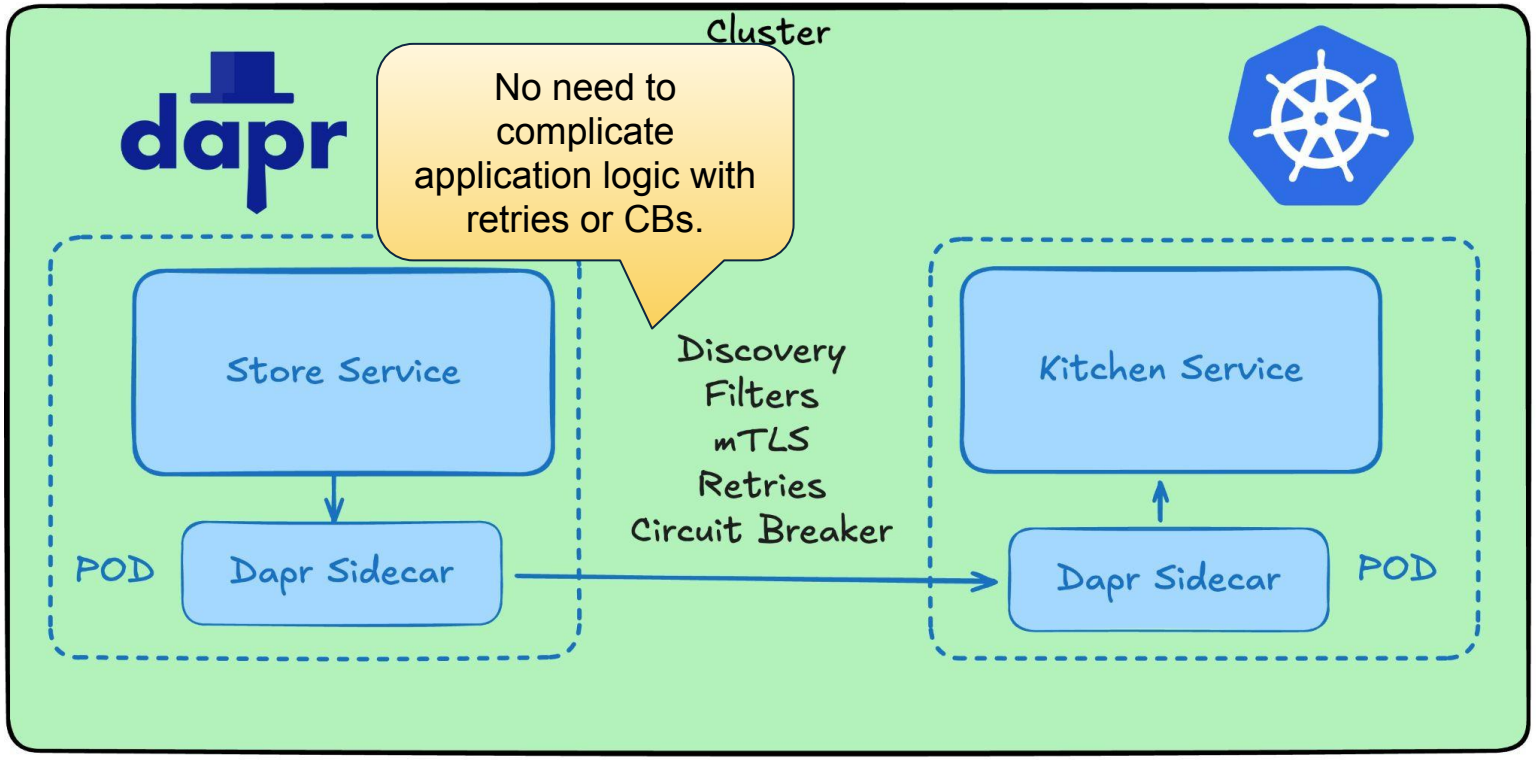
How does it work?



Sidecars to the rescue



Service To Service Invocation API



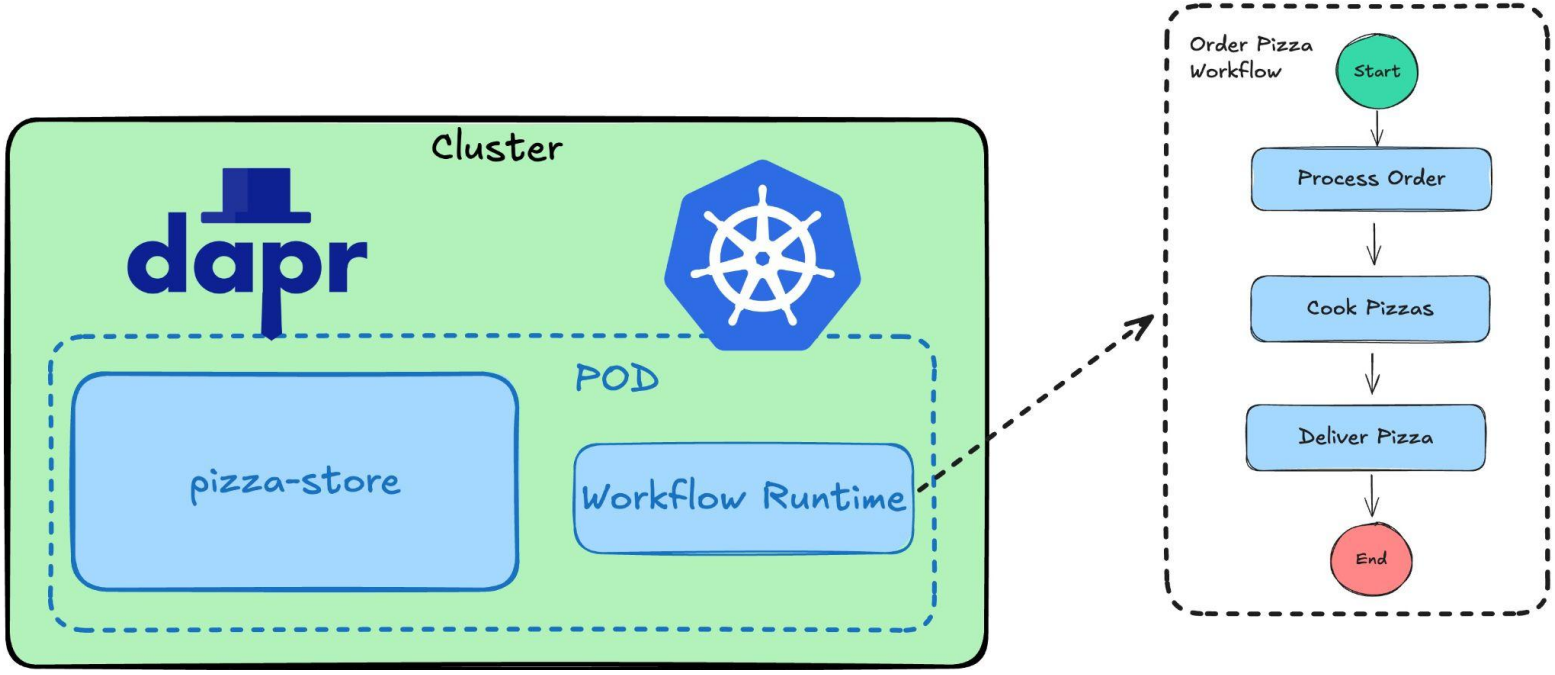
Ok, but what happen when things go wrong?

- What happens if the kitchen service is down and the retries are exhausted?
- What happens if Kafka is down?
- We cannot leave our pizza customers without their pizzas!

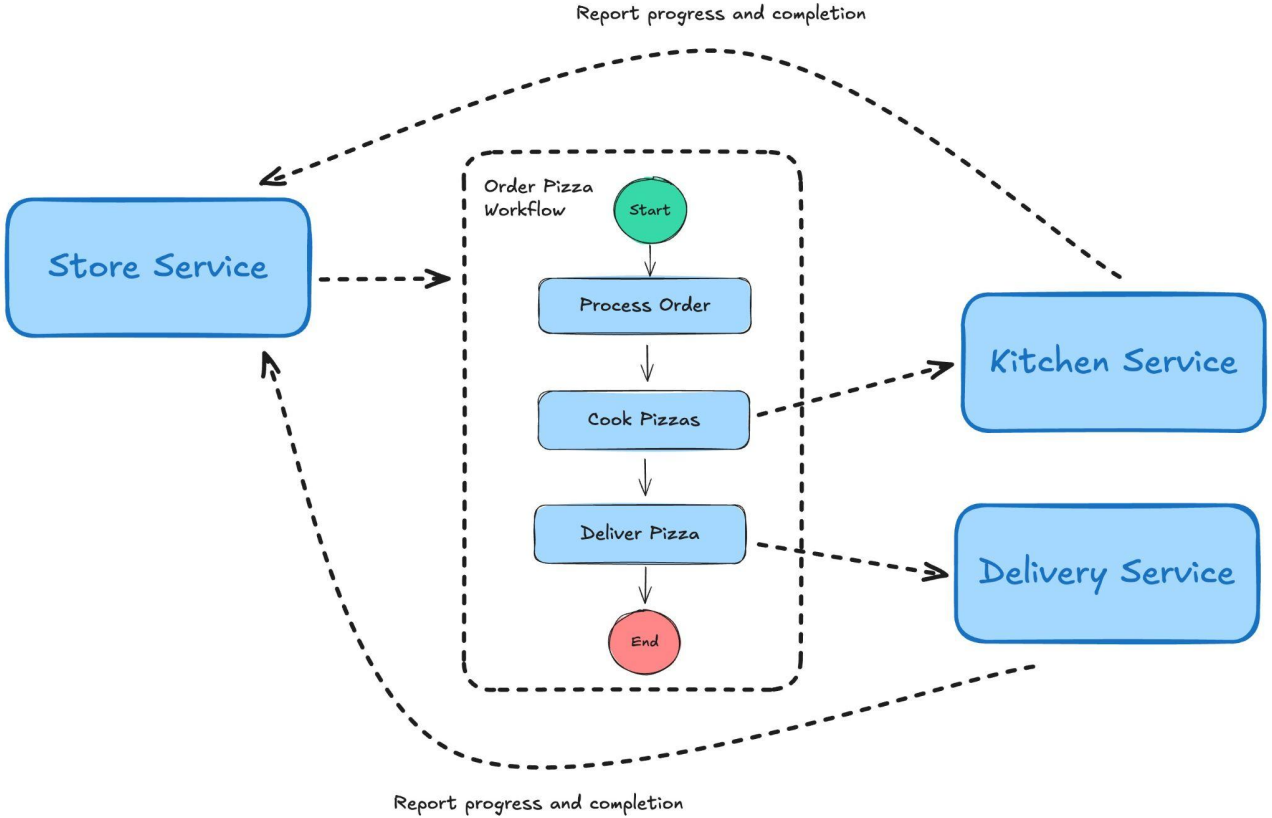
Dapr Workflows: Resilient orchestrations

- Workflows are defined in code, executed by the Dapr sidecar
- Durable, long-running state management
- Retries, timers, wait-for-events all included
- No single point of failure or SaaS service needed
- Workflows will keep trying no matter what goes down! (even the workflow runtime!!!)

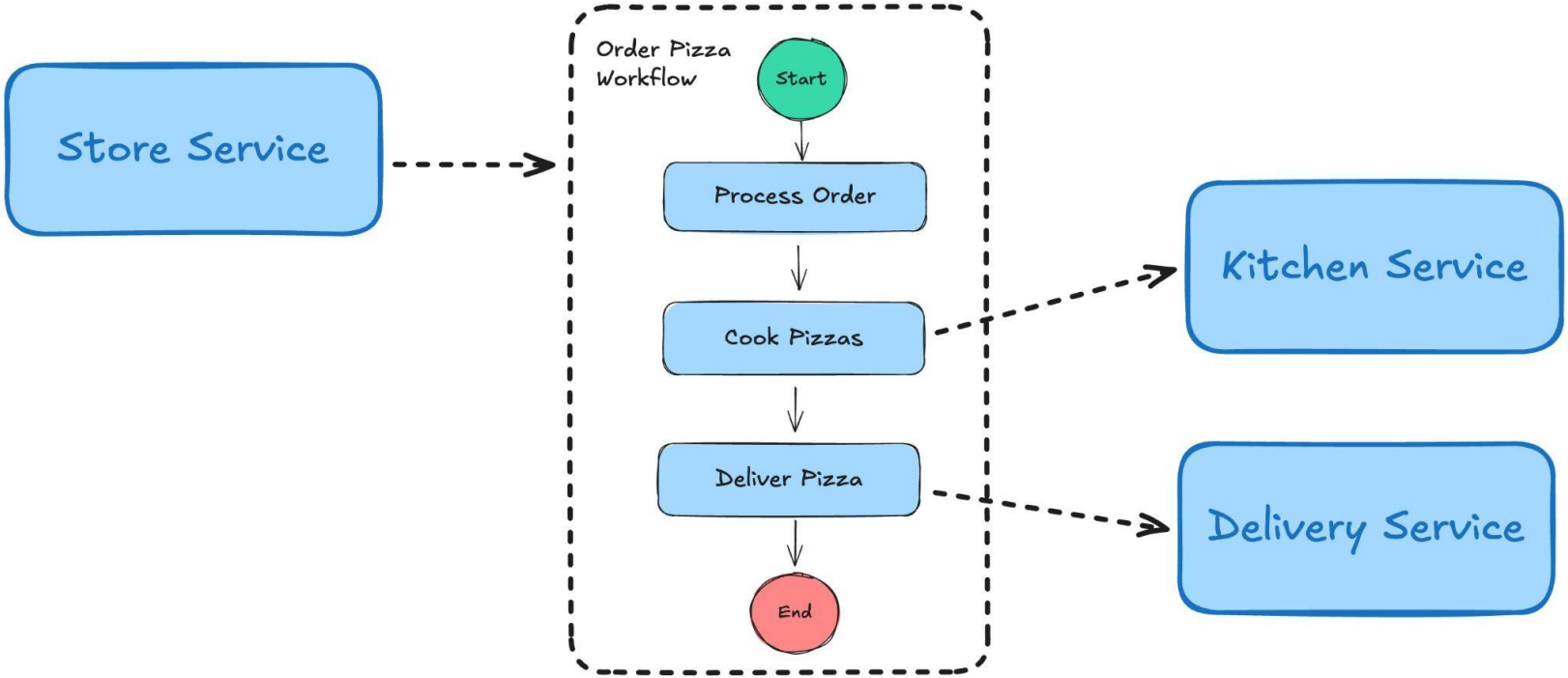
How does it work?



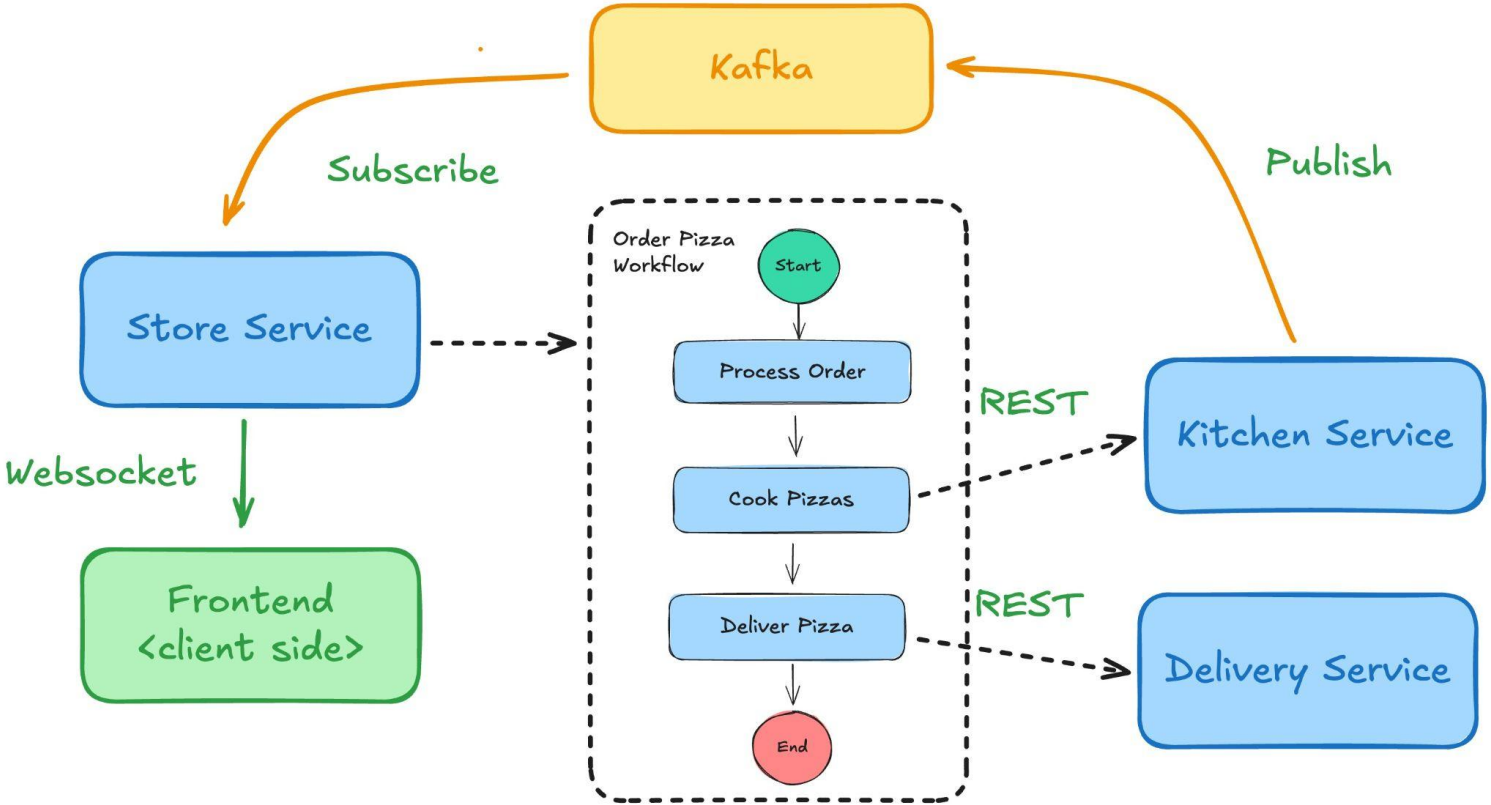
It is not that simple



Pizza Orchestration



It is not that simple ++



OpenTelemetry

dash0 Diagrid

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, RUM, and profiling (experimental)

Goals of the project

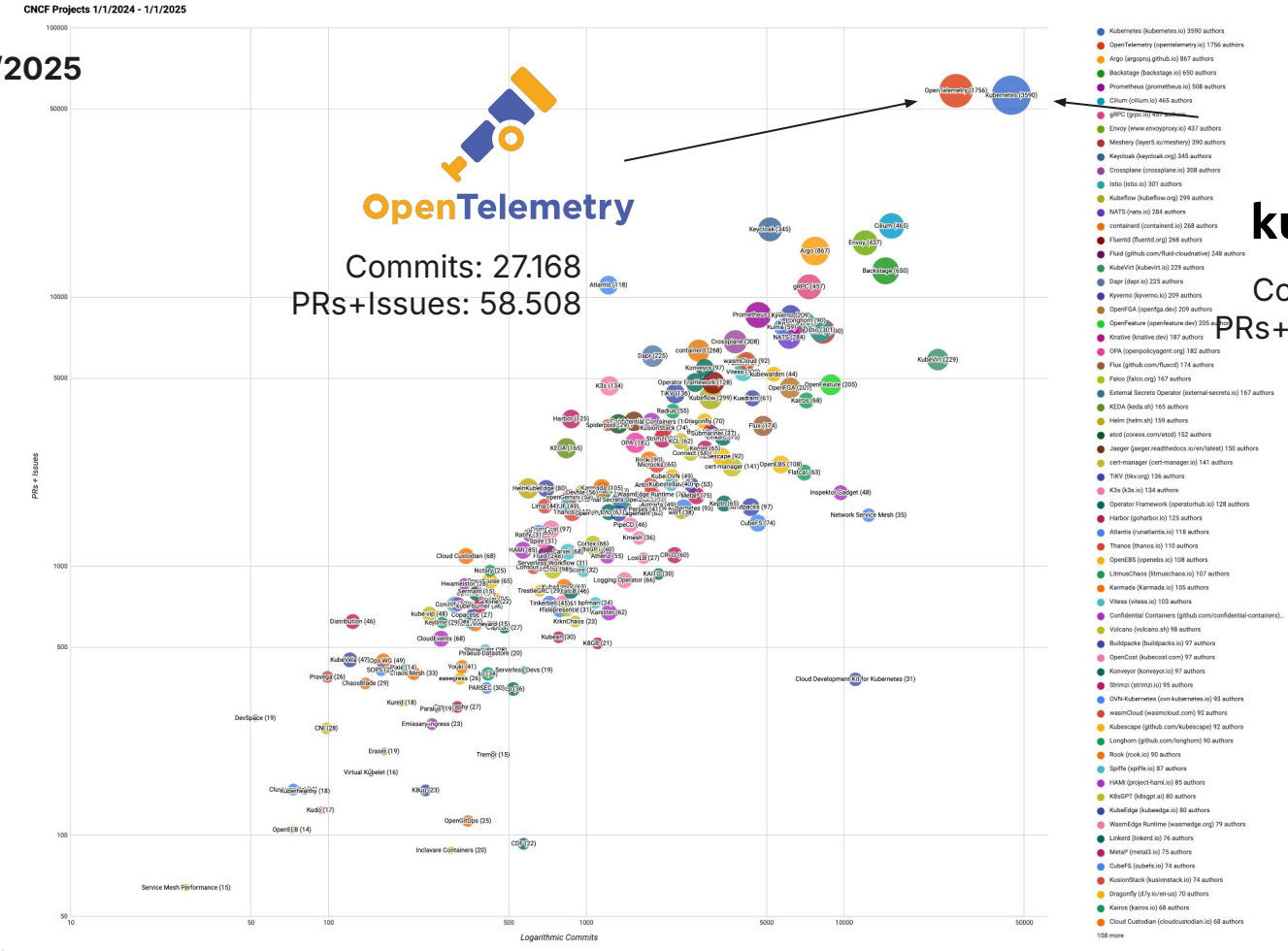
- Unified telemetry
- Vendor-neutrality
- Cross-platform

OpenTelemetry

CLOUD NATIVE
COMPUTING FOUNDATION

dash0 Diagrid

1/1/2024-1/1/2025



OpenTelemetry

Commits: 27.168
PRs+Issues: 58.508

kubernetes

Commits: 44.486
PRs+Issues: 56.299

Source: CNCF Velocity

dash0 Diagrid

49%

of respondents using OpenTelemetry in production.

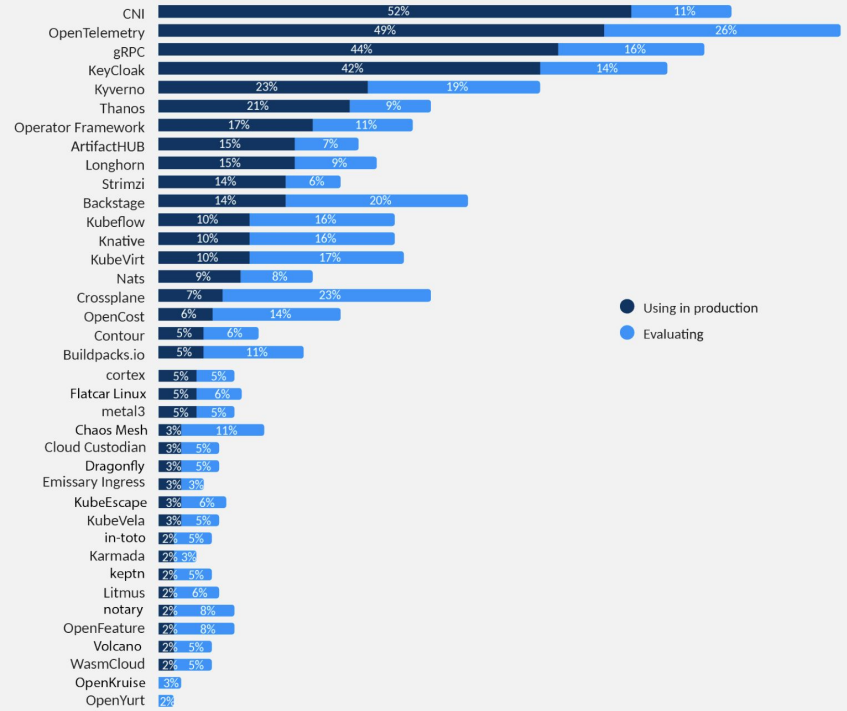
26%

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)

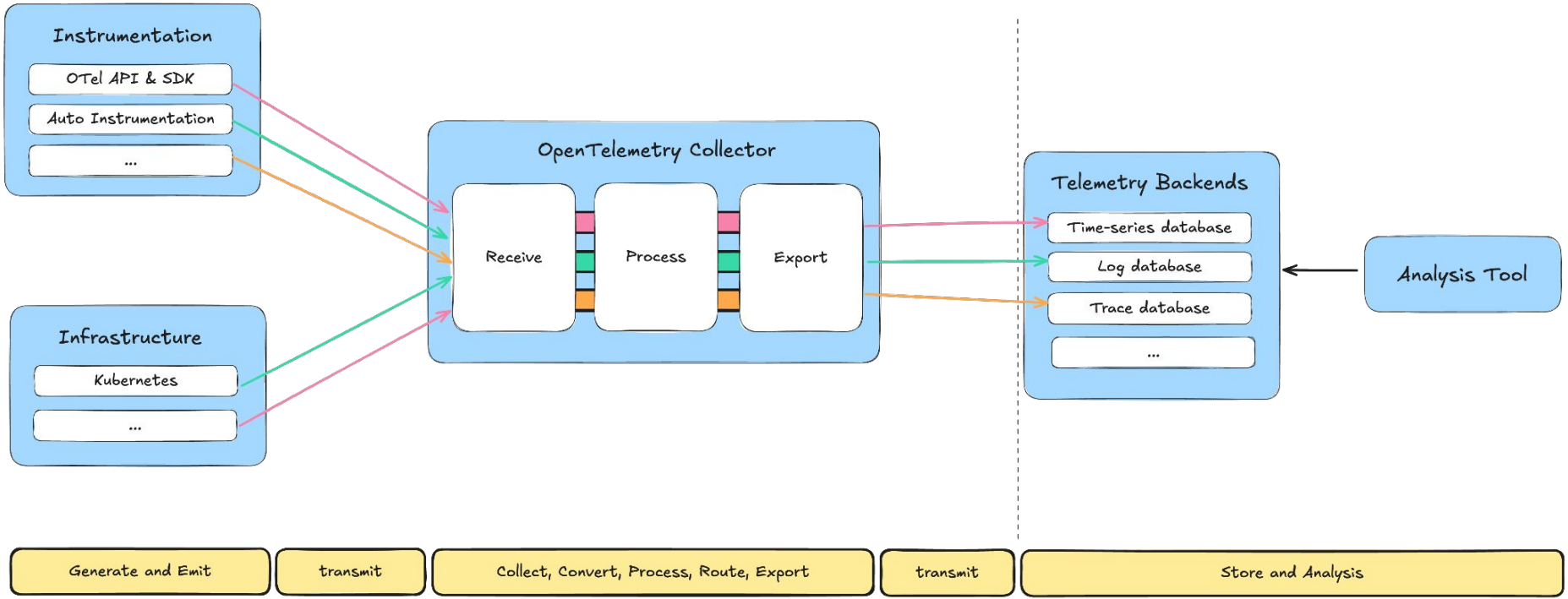


2025 CNCF Annual Survey, Q33, Sample size = 395-502, DKNS excluded

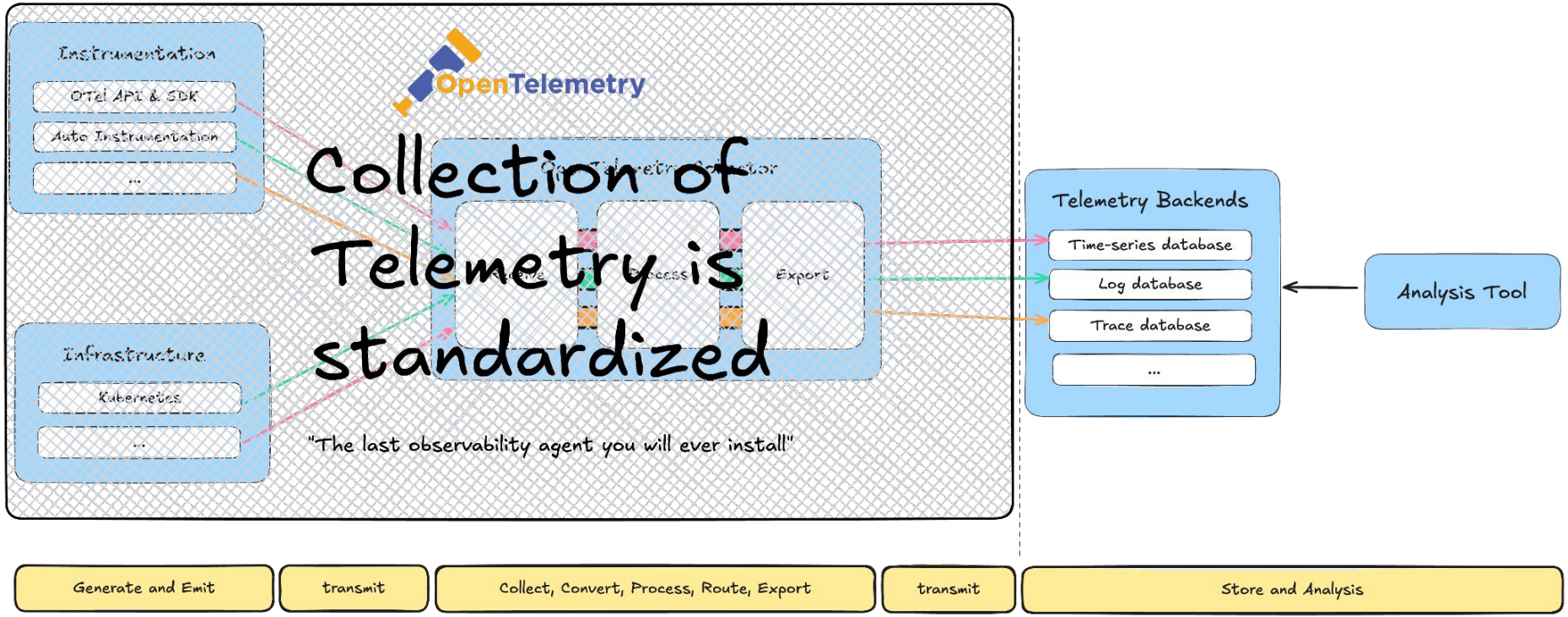
Source: https://www.cncf.io/wp-content/uploads/2026/01/CNCF_Annual_Survey_Report_final.pdf

dash0 Diagrid

OpenTelemetry Collector

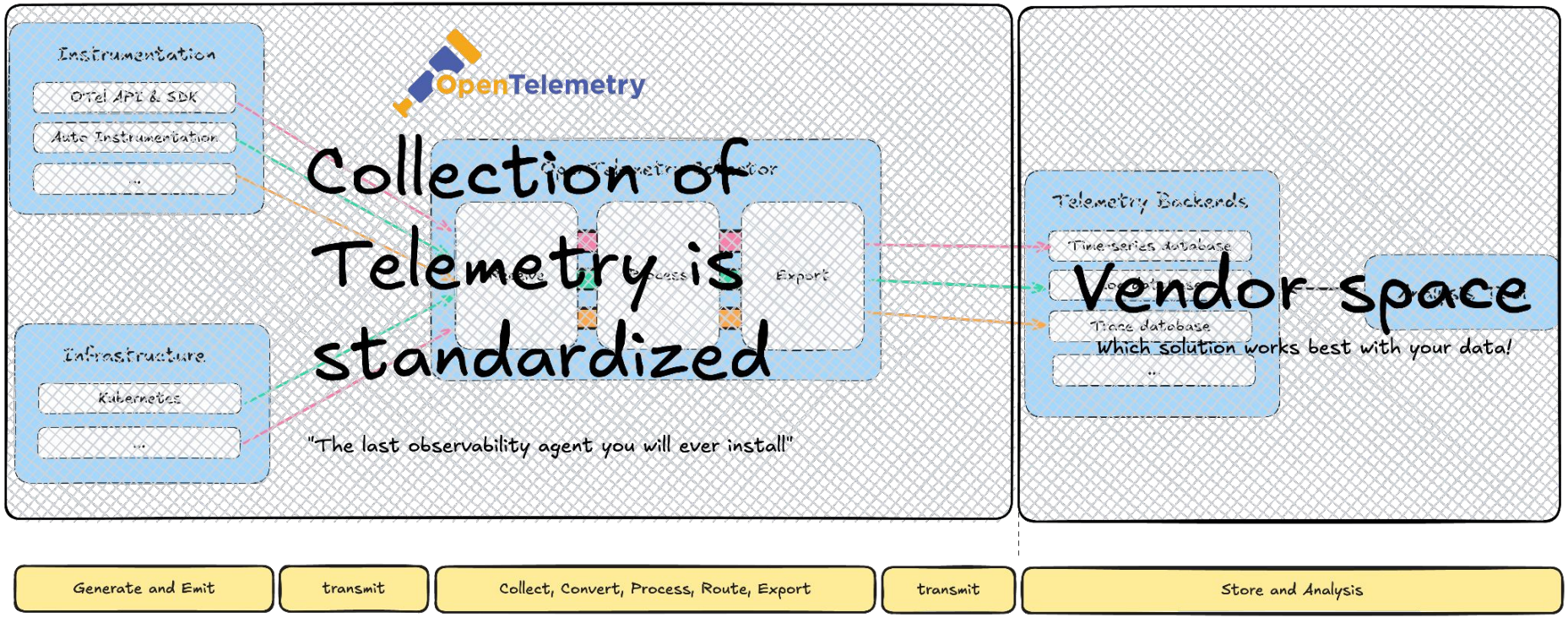


OpenTelemetry Collector



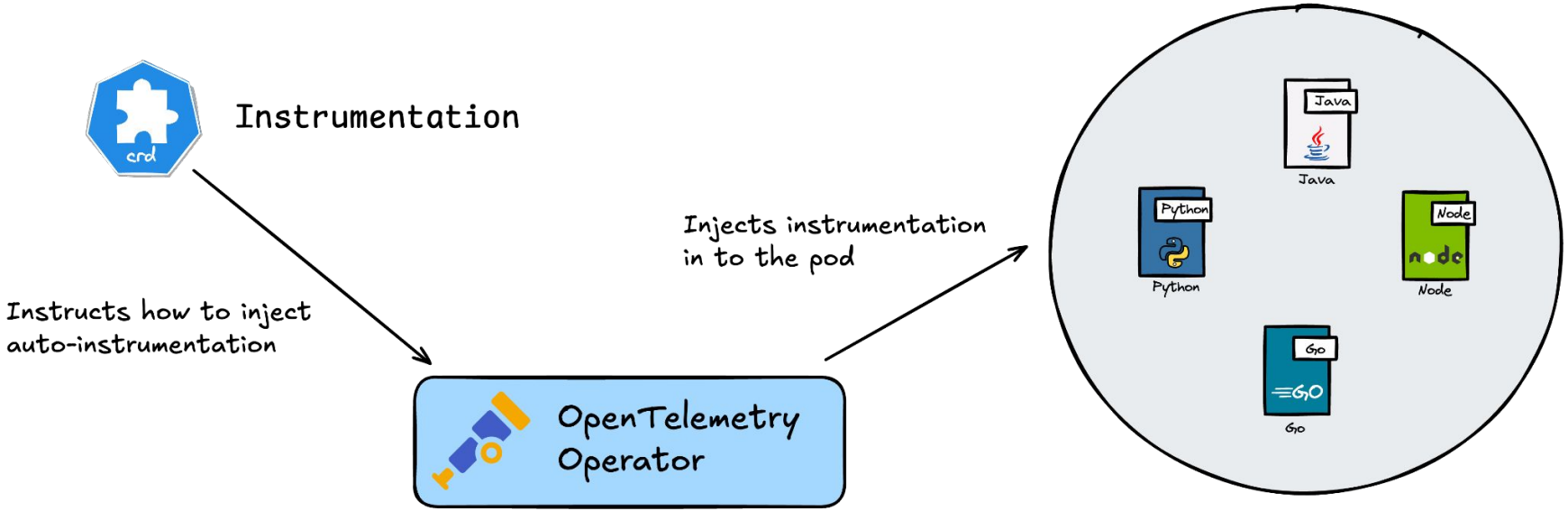
dash0 Diagrid

OpenTelemetry Collector



dash0 Diagrid

OpenTelemetry Operator

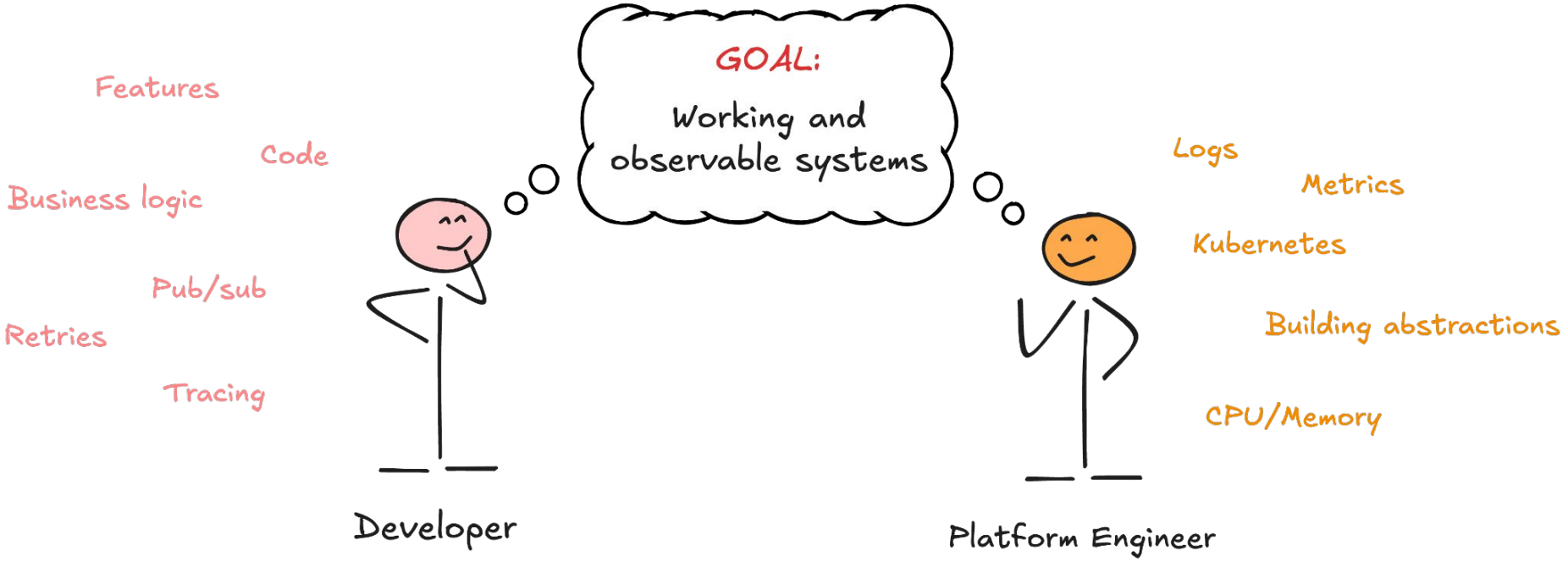


Part 3:

Making sense of all the complexity

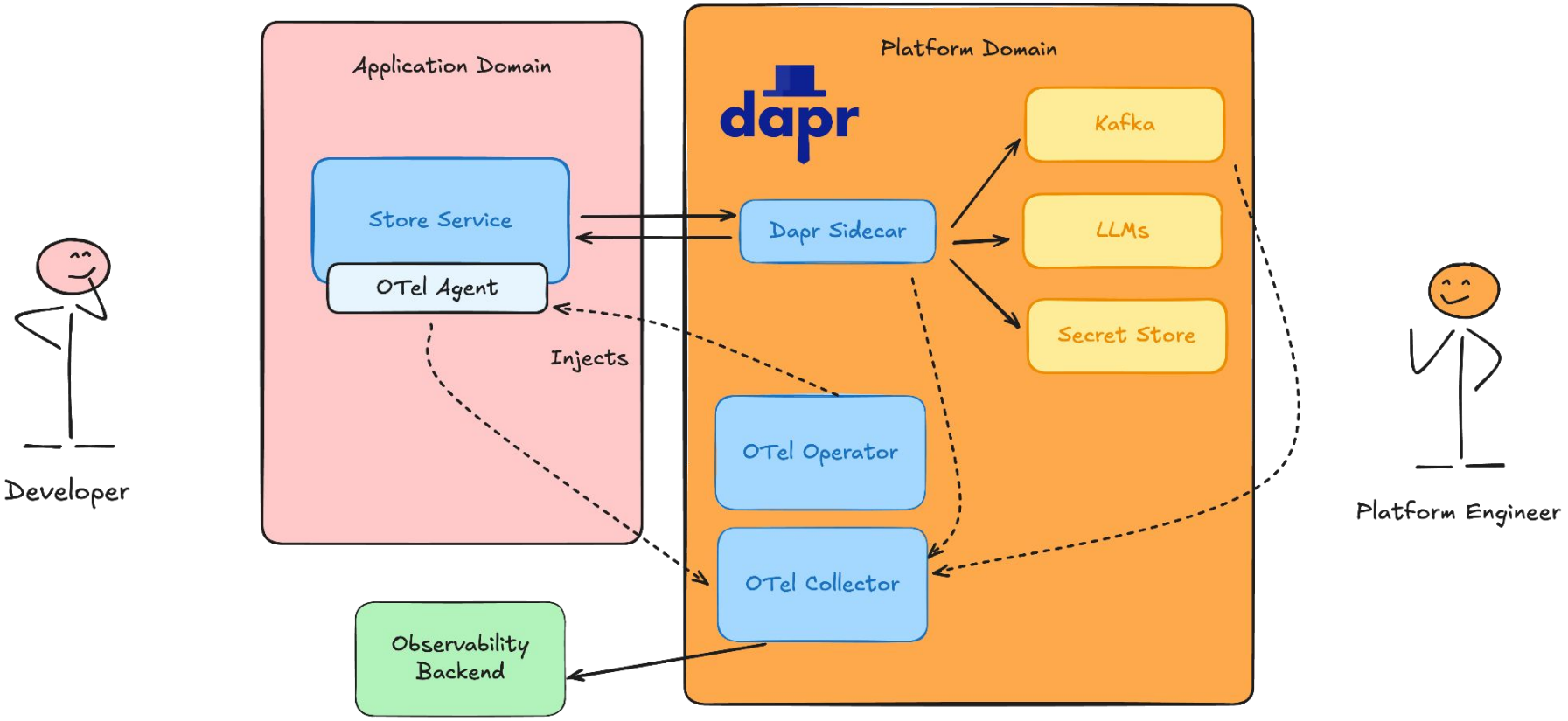
Diagrid dash0

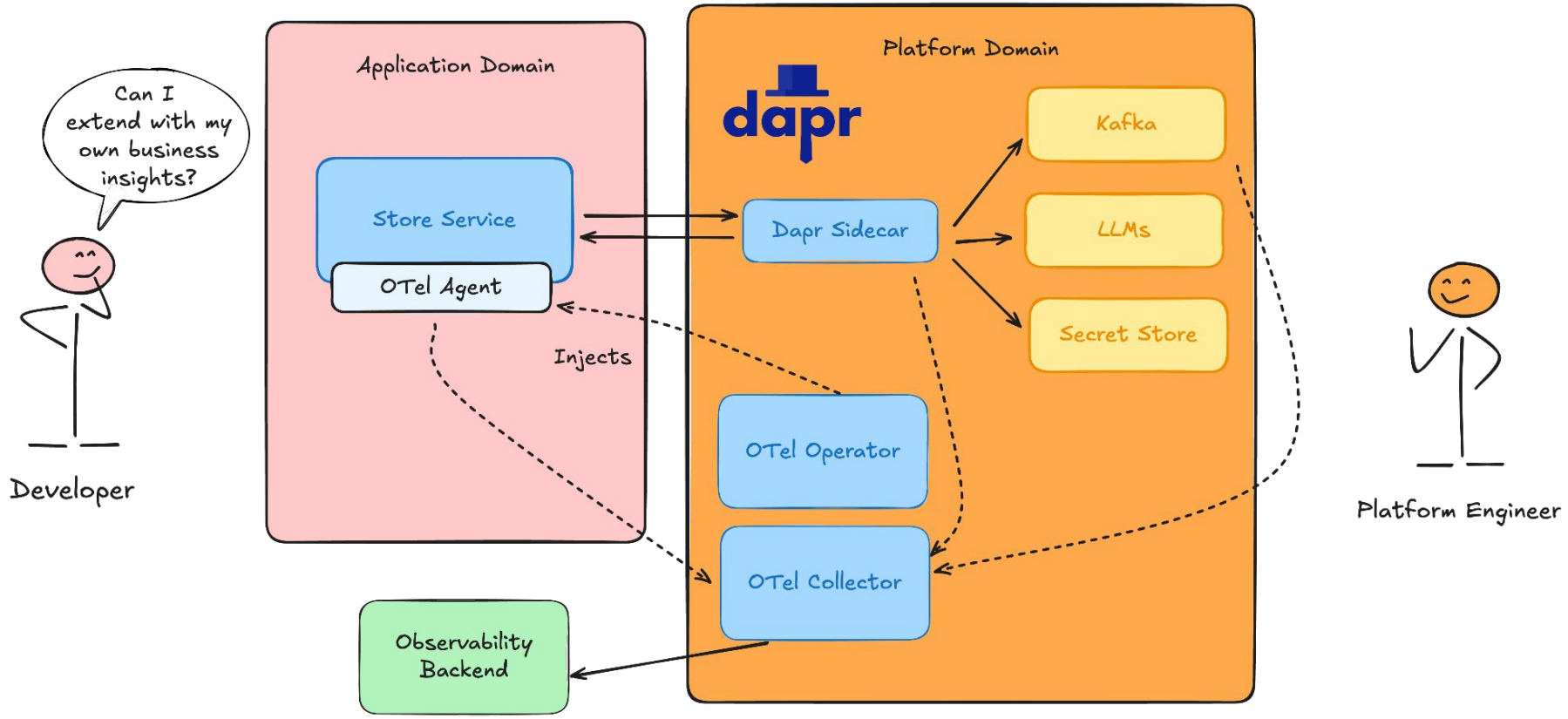
Two Perspectives, One Goal



Who Owns Tracing? A Hidden Conundrum

dash0 Diagrid

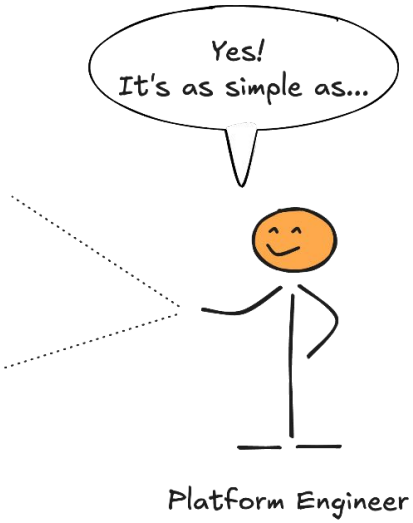

Application Domain
Store Service
OTel Agent
Injects
Platform Domain
dapr
Dapr Sidecar
Kafka
LLMs
Secret Store
OTel Operator
OTel Collector
Observability Backend
Developer
Platform Engineer



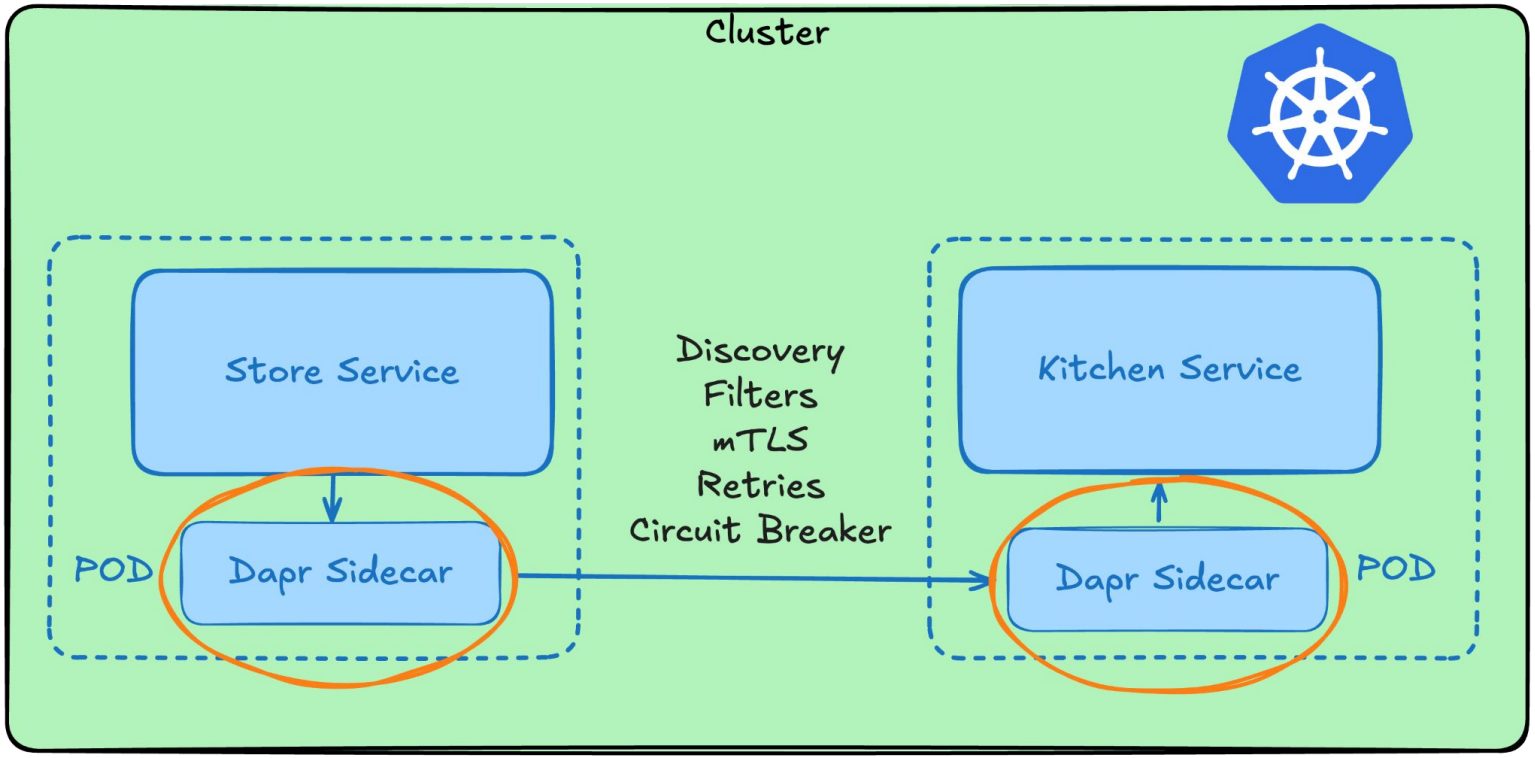
```
import io.opentelemetry.api.GlobalOpenTelemetry;
import io.opentelemetry.api.trace.Tracer;

Tracer tracer = GlobalOpenTelemetry.getTracer("application");

Span span = tracer.spanBuilder("doWork").startSpan();
...
span.end();
```



Why Observability is Critical with Dapr

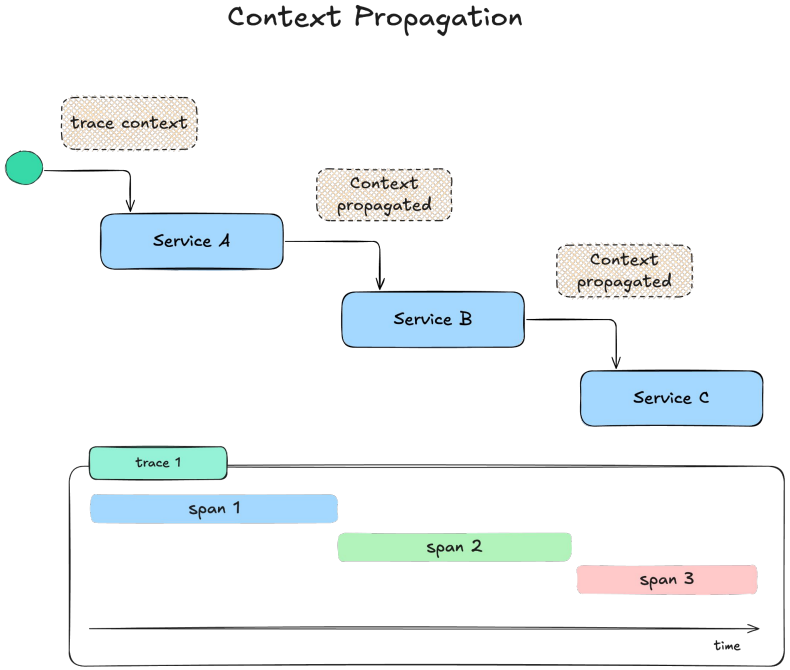


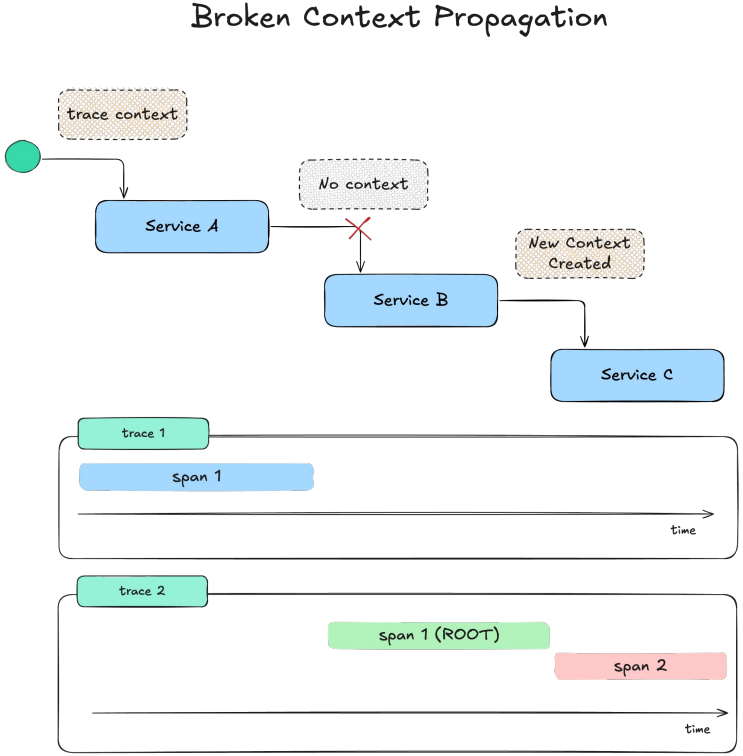
Why Async Is Hard to Observe

Traditional Tracing assumes synchronous request/response

Context often break between steps

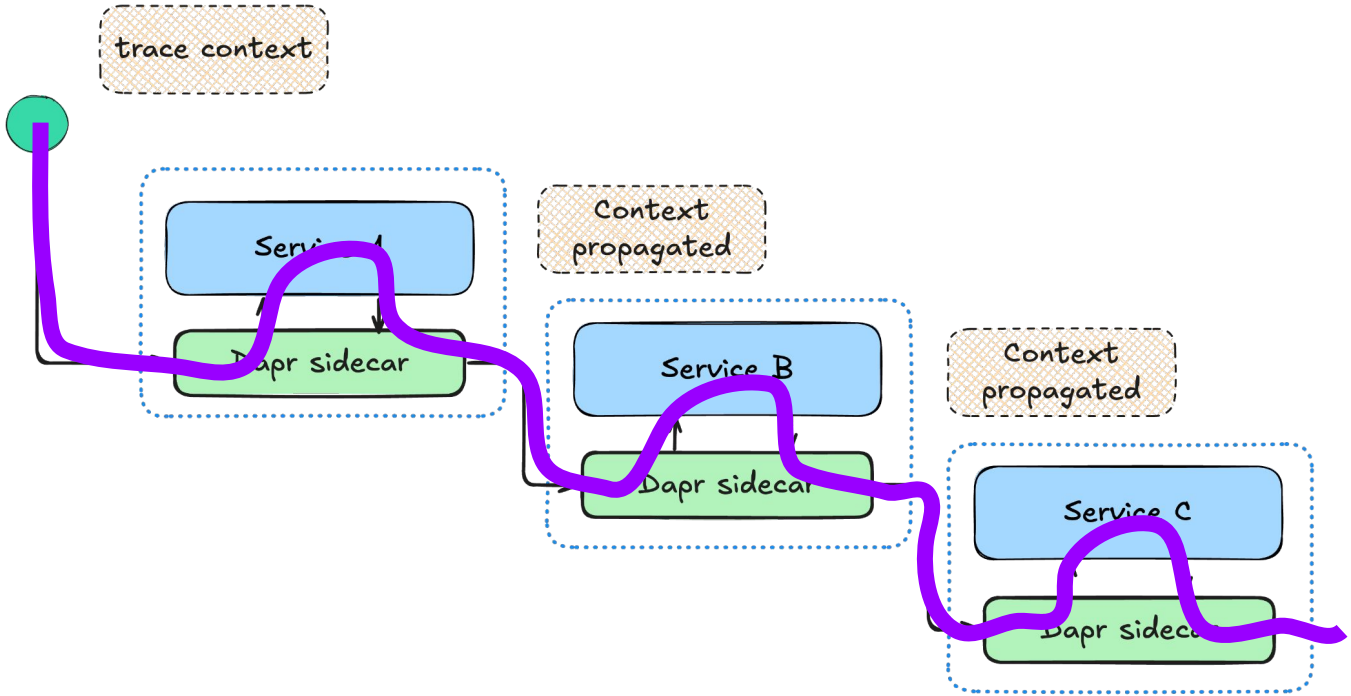
A Shared Pain: Context Gets Lost



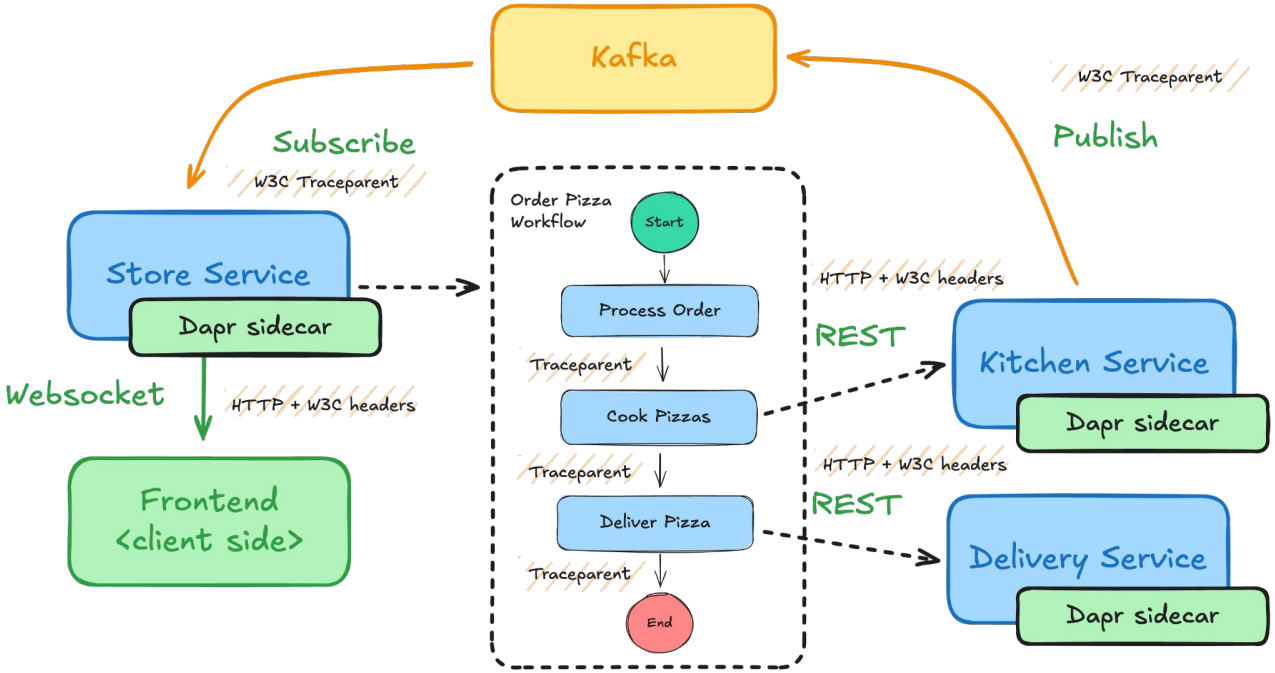


dash0 Diagrid

Trace propagation with Dapr



Context Propagation for Async Workflows



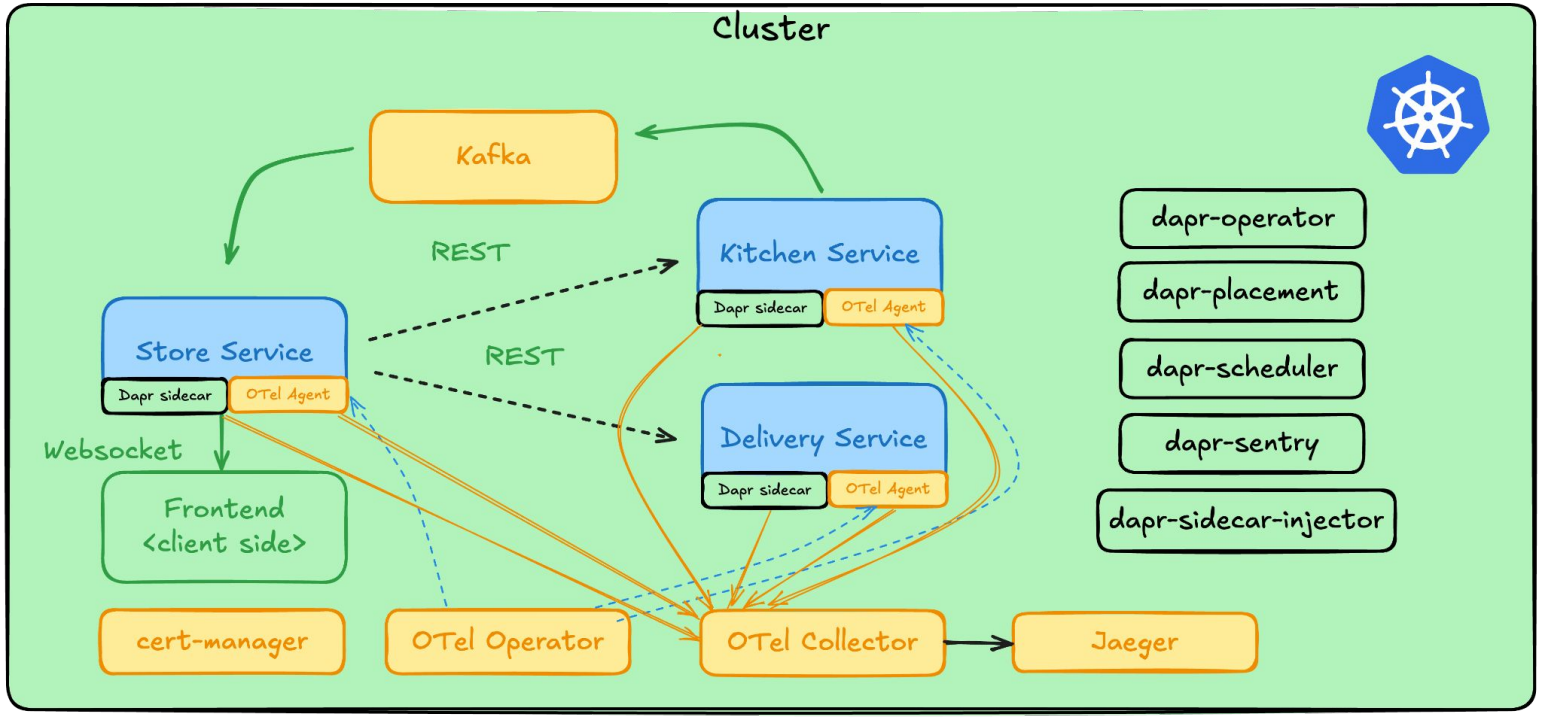
dash0 Diagrid

W3C Trace Context

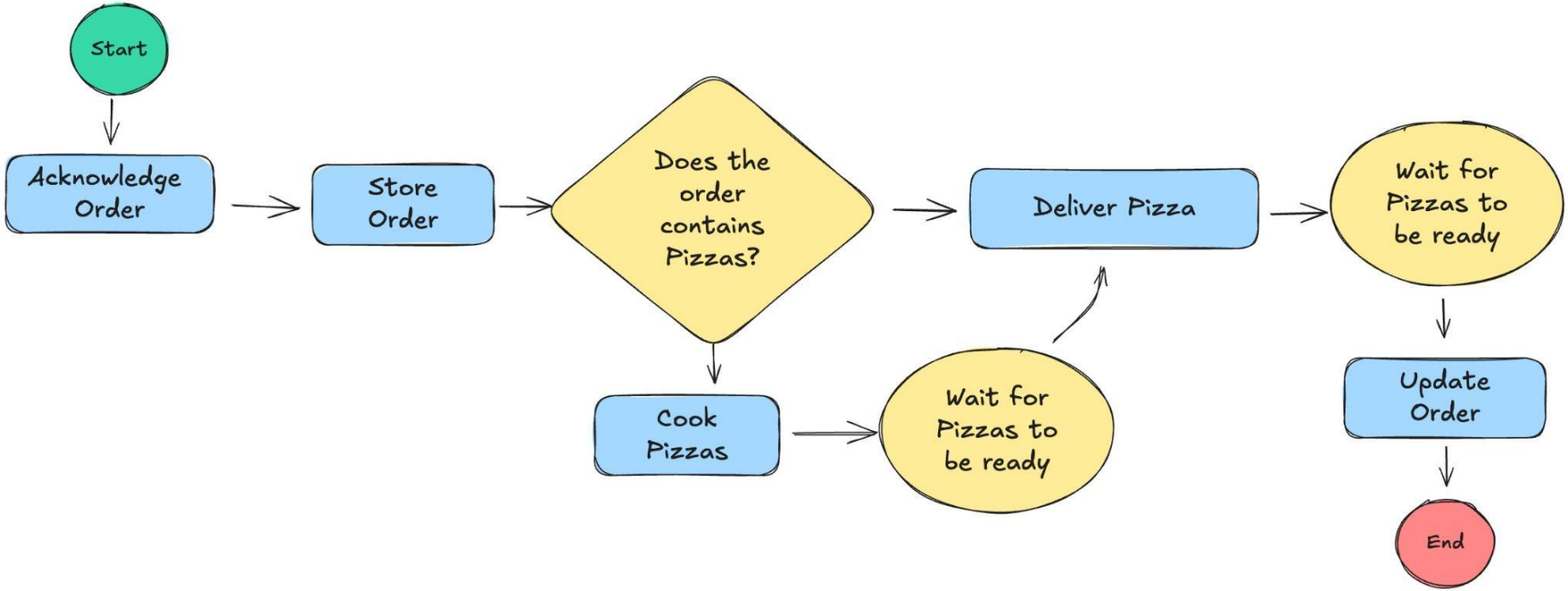
traceparent: <version>-<trace_id>-<parent_id>-<trace_flags>

tracestate: <vendor_name>=<value>,<vendor_name>=<value>

Demo setup


Cluster
Kafka
REST
Kitchen Service
Dapr sidecar
OTel Agent
REST
Store Service
Dapr sidecar
OTel Agent
Websocket
Frontend
<client side>
Delivery Service
Dapr sidecar
OTel Agent
dapr-operator
dapr-placement
dapr-scheduler
dapr-sentry
dapr-sidecar-injector
cert-manager
OTel Operator
OTel Collector
Jaeger

Workflow Details



Demo 2:

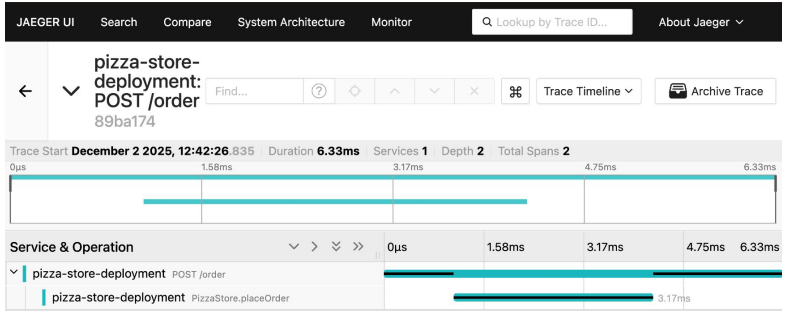
Putting it all together

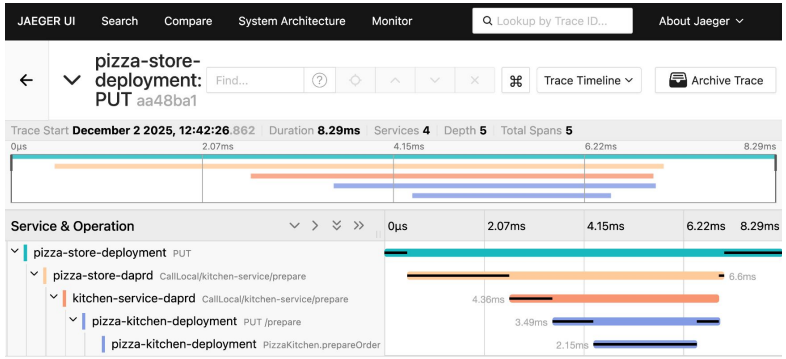
Diagrid dash0

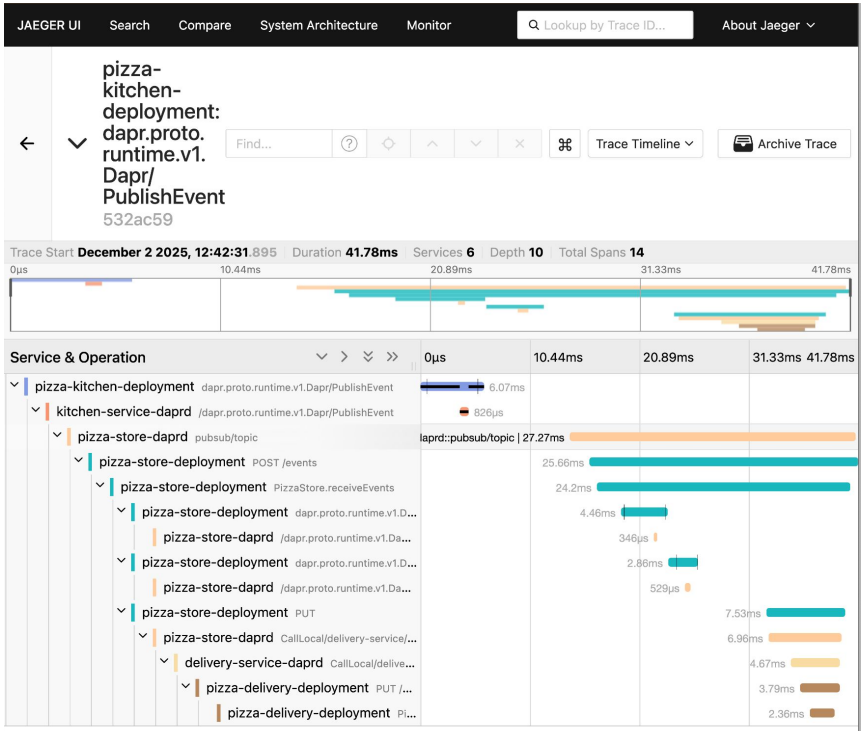
What Works Today

- Dapr supports OpenTelemetry out of the box
- Sidecar emits spans for pub/sub, service invocation, and workflows
- Consistent parent-child relationships
- OpenTelemetry Operator enables auto-instrumentation
- OpenTelemetry Collector handles ingestion, processing, export

What Works Today





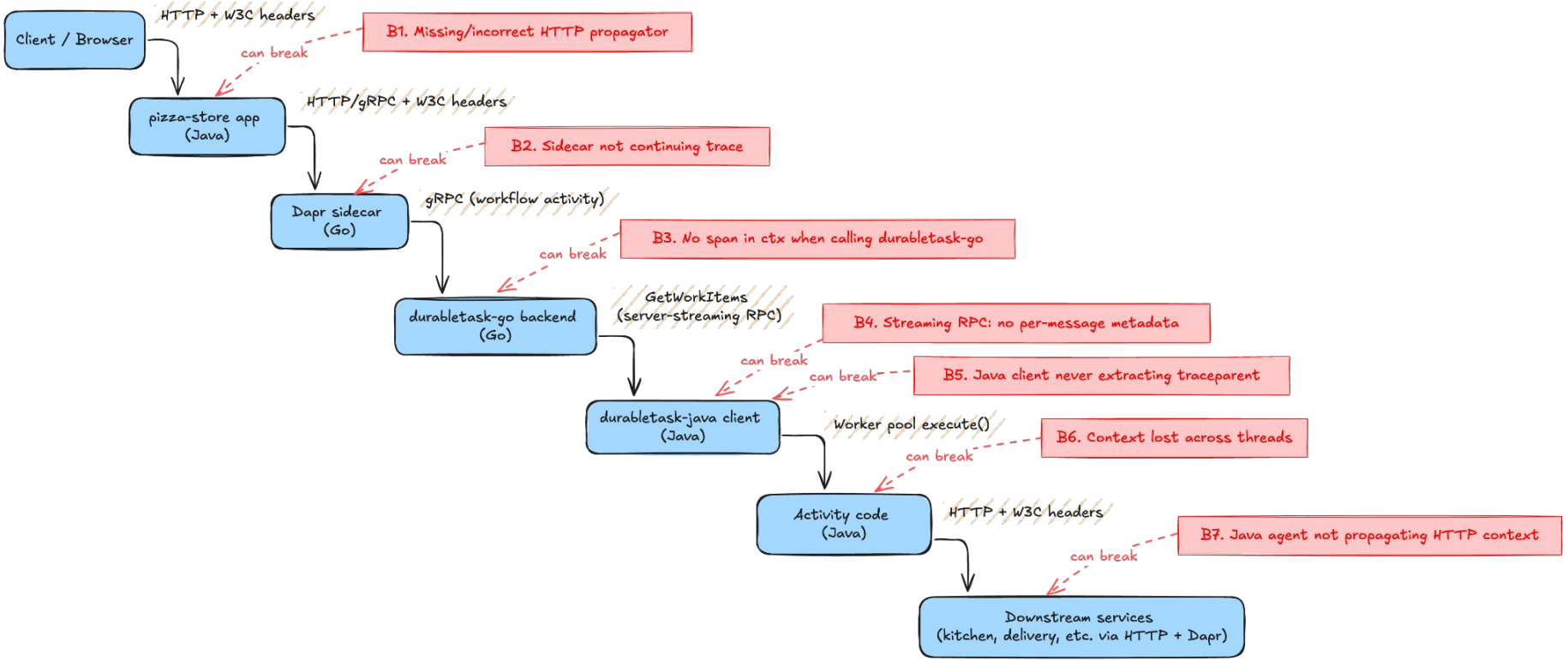


dash0 Diagrid

Challenges

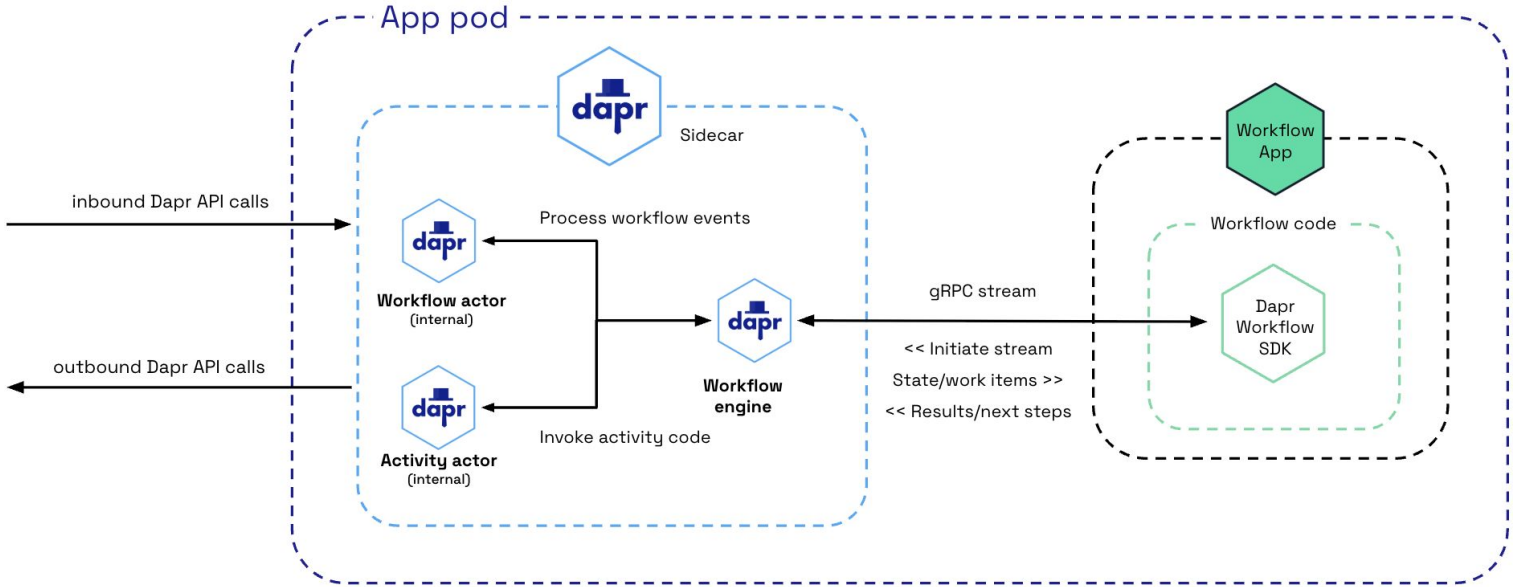
- Async boundaries break context
- Sidecars add additional hops
- Workflow engines introduce thread + process separation

Context Propagation for Async Workflows



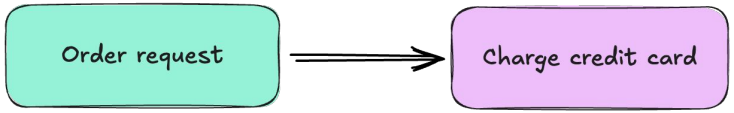
dash0 Diagrid

Challenges with gRPC streaming

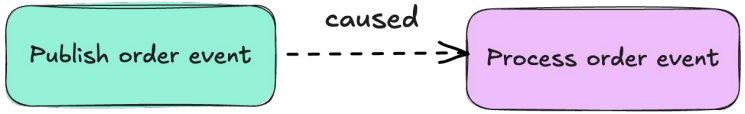


Trace Topology Is a Design Choice

- Parent–child implies temporal enclosure
- Links imply causal relationship
- Async systems force us to choose deliberately



Parent-Child



Span Link

Parent–Child vs Span Links

Parent-Child works when…

- Caller waits for completion
- Temporal enclosure is real
- Execution is synchronous or actively coordinated

Examples:

- HTTP / gRPC calls
- Service invocation
- Workflow orchestration (when the orchestrator is actively running)

Span links fit better when…

- Work is asynchronous
- No blocking relationship exists
- Causality does not imply execution order

Examples:

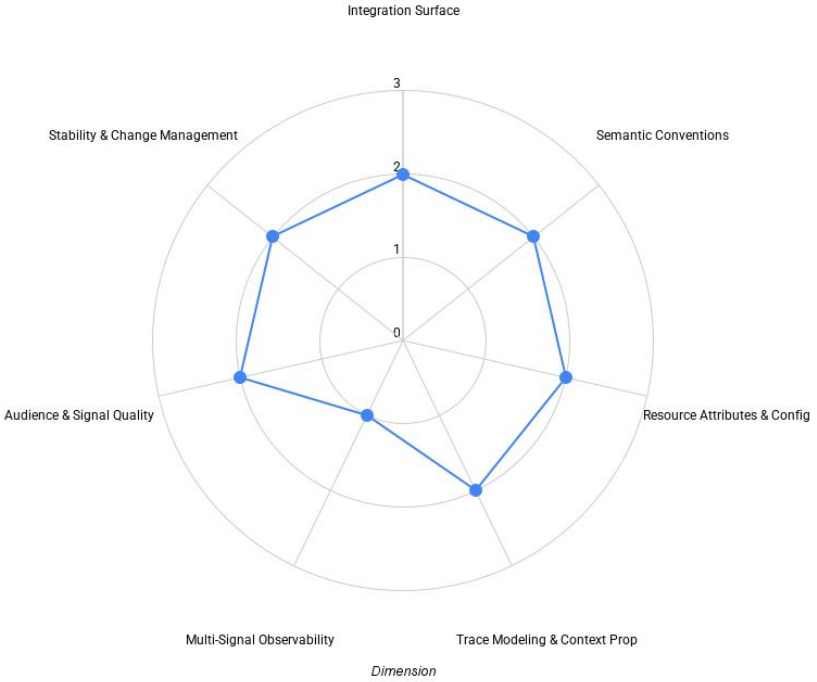
- Pub/Sub (producer → consumer)
- Fan-out / fan-in
- Fire-and-forget events

Why we started here

- Optimize for human understanding first
- Make the workflow readable as a single story
- Refine semantics once the mental model is clear

Proposing a maturity model for OpenTelemetry support

- A shared framework for evaluating OpenTelemetry support
- Inspired by the CNCF Platform Engineering maturity model
- Descriptive, not prescriptive
- Focused on evolution, not scoring



Github Issue: https://github.com/open-telemetry/community/issues/3247

dash0 Diagrid

OpenTelemetry maturity evolves through real fixes

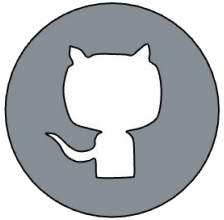
Level 0: Instrumented	Level 1: OpenTelemetry Aligned	Level 2: OpenTelemetry Native	Level 3: OpenTelemetry Optimized
Telemetry exists primarily to support internal debugging and development needs. OpenTelemetry is not yet a primary design concern.	OpenTelemetry is explicitly supported, often alongside legacy approaches. Telemetry works for common scenarios, but legacy assumptions still influence design.	OpenTelemetry is the primary integration surface. Telemetry is designed intentionally, with correlation and user experience in mind.	OpenTelemetry support is continuously refined based on real-world usage and feedback. Telemetry is treated as a long-lived product surface.
Context propagation exists and working	Parent–child relationships everywhere	Intentional trace design (span links at async boundaries)	Custom semantic conventions, refinement, and stewardship

Github Issue: https://github.com/open-telemetry/community/issues/3247

dapr

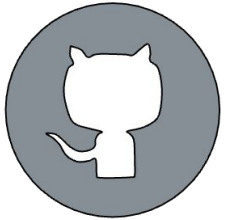
dash0 Diagrid

Gaps and Fixes in Dapr & OTel



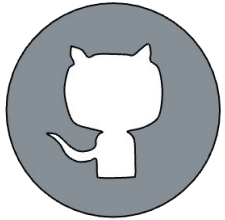
PR #57

Trace context, SemConv



PR #9213

Trace context, SemConv, Pub/Sub Span kind



PR #46

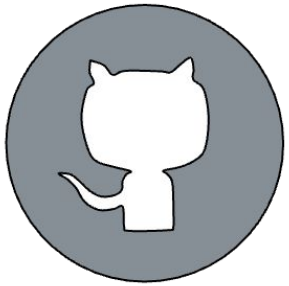
Propagating context to executors client side

If you want to learn more Check the Dapr University

https://www.diagrid.io/dapr-university#dapr-workflow

dash0 Diagrid

Get Involved



Github

github.com/open-telemetry
github.com/dapr

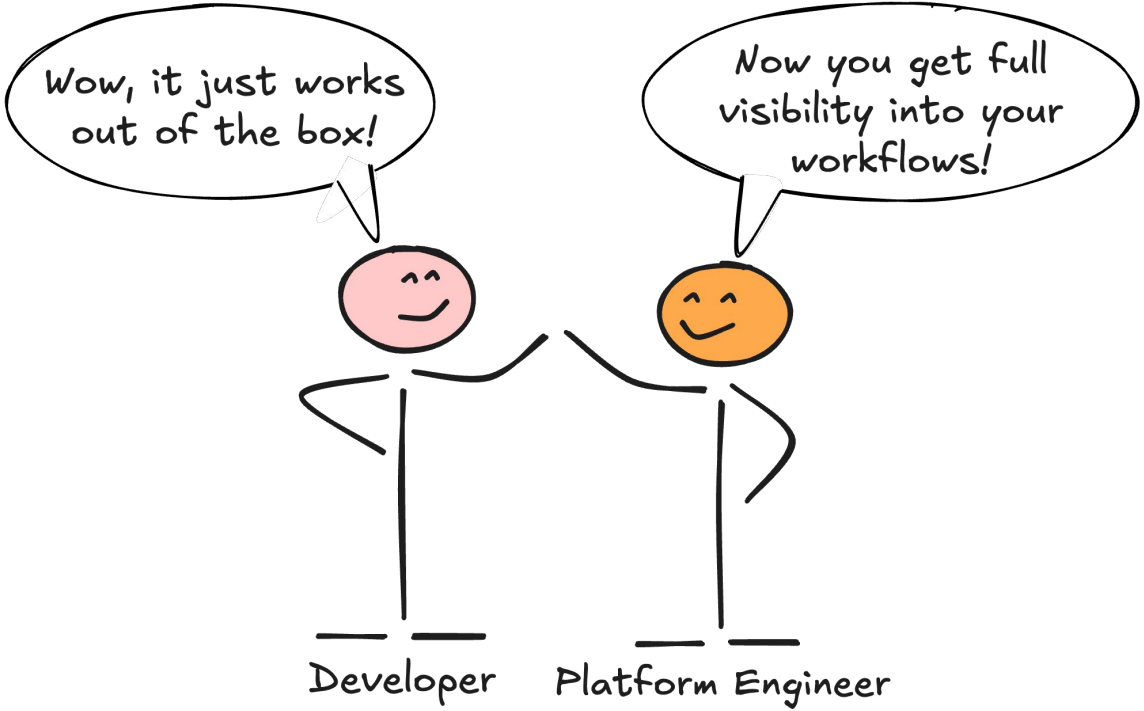
CNCF Slack

#dapr
#opentelemetry, otel-*

If you want to learn more Check the Dapr University

https://www.diagrid.io/dapr-university#dapr-workflow

Enabling the Golden Path...



dash0 Diagrid

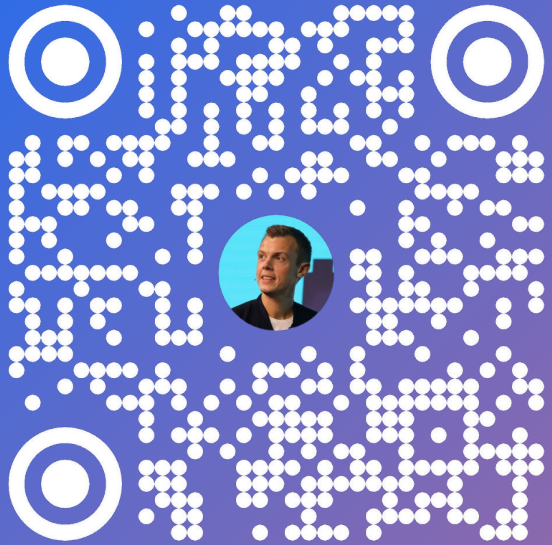
FEB 11–12, 2026
CONTAINER days LONDON

That’s all folks!

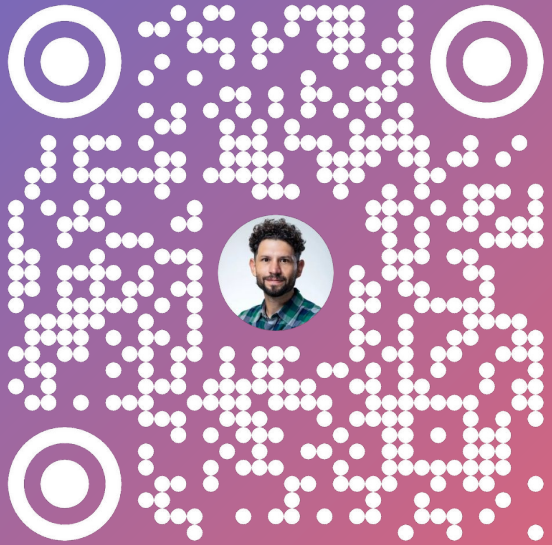
Thank you!

Diagrid dash0

Get in touch with us!



Kasper Borg Nissen



Mauricio Salatino

Diagrid dash0