tim & koko

OpenTelemetry & Grafana Tempo

Distributed Tracing Made Easy

Thomas Philipona

Bern, 21.08.2025

tim&
koko

Agenda

1. Why
2. OpenTelemetry Introduction
3. Grafana Tempo
4. Demo
5. Distributed Tracing Stack in Practice

Observability – See, Understand, Improve

Know what’s happening

It allows you to understand a system from the outside.
It is the foundation to detect issues quickly and understand what is happening.

Understand why

Diagnose root causes with context, which is super important in increasingly complex environments.

Improve faster

The what and why helps to reduce the time to fix issues (Shorter Mean Time to Repair).
Creates business value and happy customers!

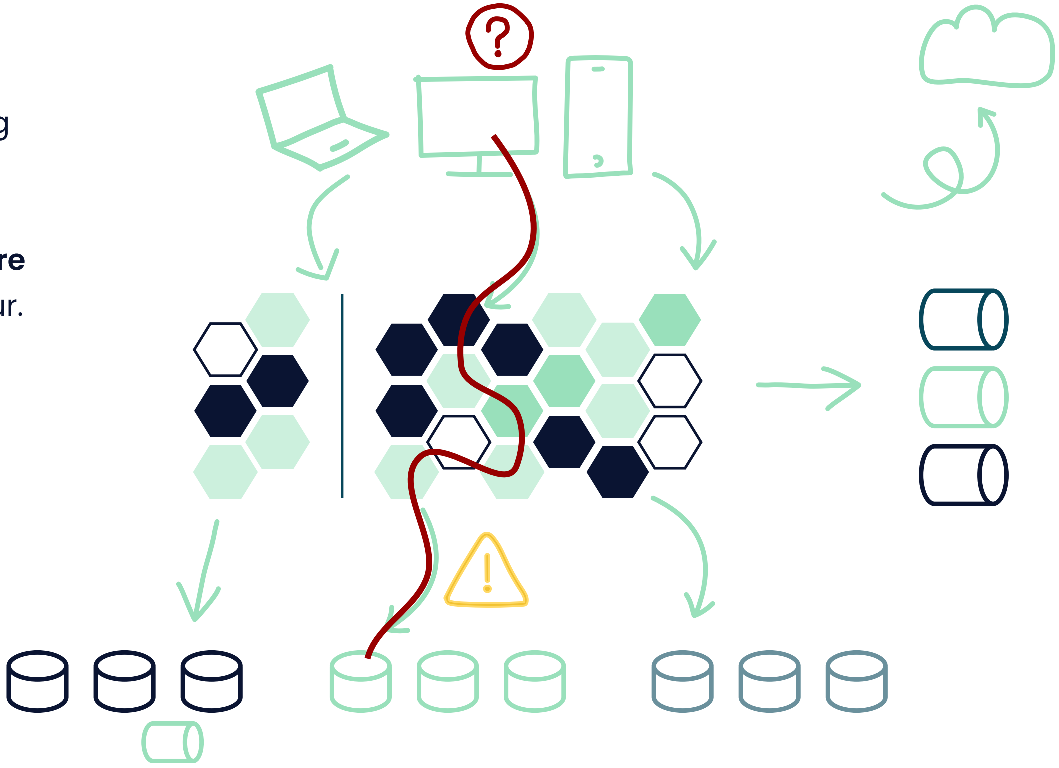
tim&
koko

Why Distributed Tracing matters?

Distributed tracing is a method for tracking requests in **distributed systems** (e.g., microservices architectures) from **start** to **finish** in order to understand **how** and **where** they spend time and where **problems** occur.

Three components complete the picture:

- Logs: What happened?
- Metrics: How is it going?
- Traces: Where is it stuck?



Distributed Tracing

OpenTelemetry

OpenTelemetry

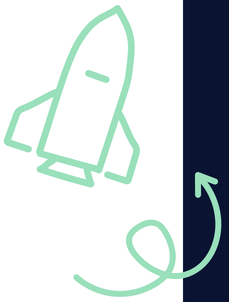
OpenTelemetry
What is OTel?

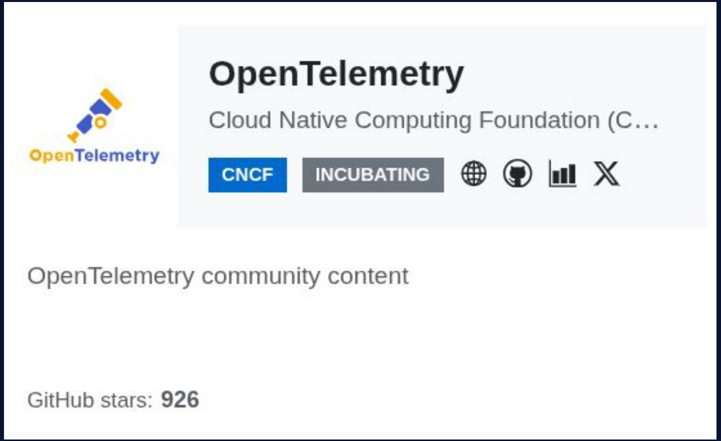
An observability framework and toolkit designed to facilitate the

- Generation
- Export
- Collection

of telemetry data such as **traces**, **metrics**, and **logs**.

https://opentelemetry.io





OpenTelemetry

Cloud Native Computing Foundation (C…

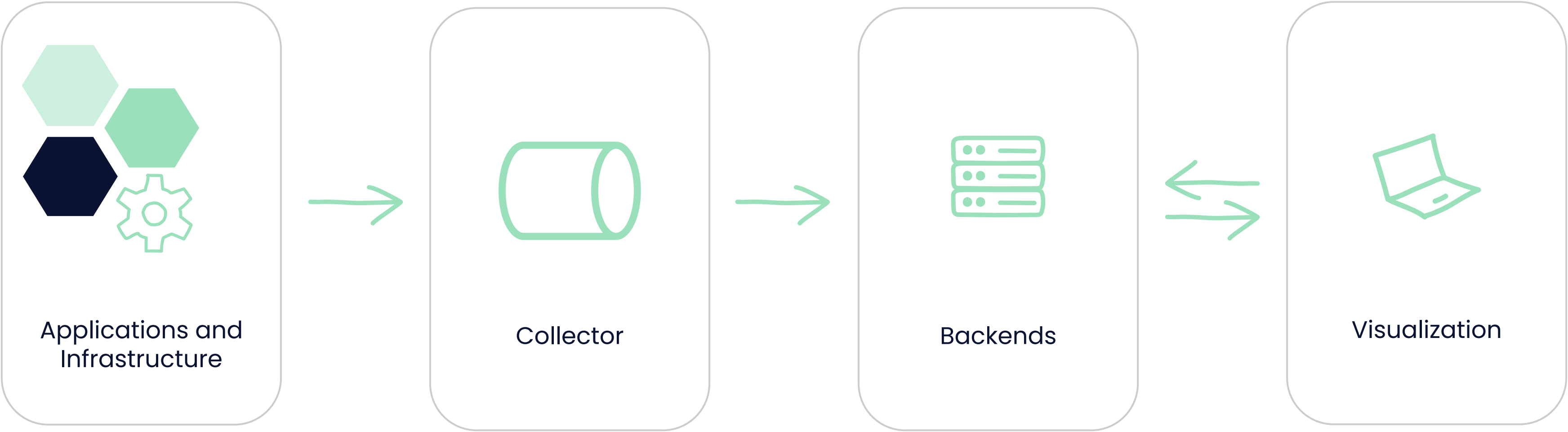
CNCF INCUBATING

OpenTelemetry community content

GitHub stars: **926**

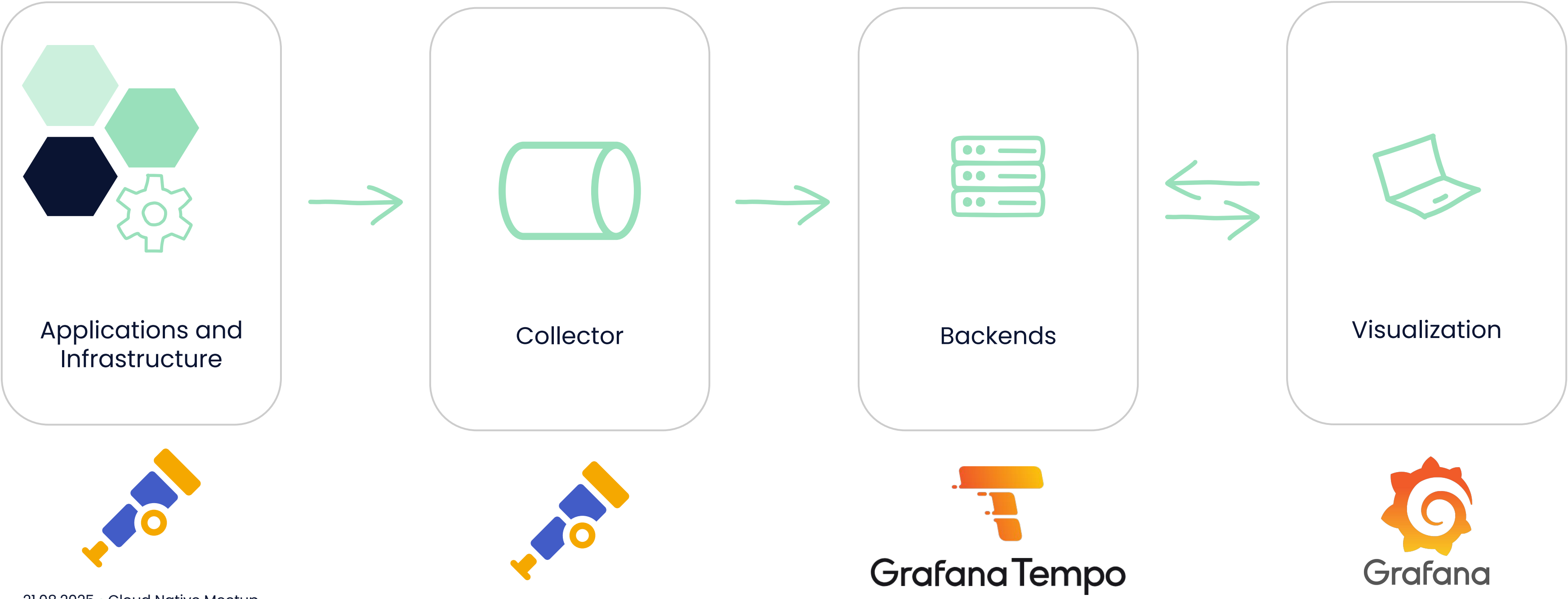
OpenTelemetry

OpenTelemetry
Components / Architecture



OpenTelemetry

Components / Architecture for Traces



tim&
koko

OpenTelemetry
Instrumentation

Zero Code

Great for getting started! Packaged SDK, libraries and exporters are injected automatically into the application. You will get your code auto instrumented. Such as HTTP APIs, Backend Calls, DB Calls, Messageques and so on.

.NET, Go, Java, JavaScript, PHP, Python

Code-based

Import OTel API and SDKs which provide access to the OpenTelemetry functionality.
Ideally for custom instrumentation, to implement custom traces and metrics.

.NET, Go, Java, JavaScript, PHP, Python, C++, C, Rust, Ruby, Swift, Erlang

Zero Code Instrumentation
Java Spring Boot

- Add dependency to your application
- configure an endpoint and export signals
- optionally configure the instrumentation
- Similarly done for other languages (.NET, Go, JavaScript, PHP, Python)

gradle.properties

```
import org.springframework.boot.gradle.plugin.SpringBootPlugin

plugins {
  id("java")
  id("org.springframework.boot") version "3.2.0"
}

dependencies {
  implementation(platform(SpringBootPlugin.BOM_COORDINATES))

implementation(platform("io.opentelemetry.instrumentation:open
-instrumentation-bom:2.19.0"))
}
```

Auto Instrumentation
Kubernetes Operator

```yaml
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  propagators:
    - tracecontext
    - baggage
  sampler:
    type: parentbased_traceidratio
    argument: '1'
  java:
    env:
      - name: OTEL_INSTRUMENTATION_KAFKA_ENABLED
        value: false
      - name: OTEL_INSTRUMENTATION_REDISCALA_ENABLED
        value: false
```

Annotation on Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
  [..]
spec:
 template:
  metadata:
   annotations:
    instrumentation.opentelemetry.io/inject-java: "true"
    spec:
      containers:
        [..]
```

OpenTelemetry

Telemetry Data /Signals

Traces

Path of a request through the entire system

Metrics

A measurement captured at runtime. eg.

http.requests.total 42
db.queries.count 501

Logs

A timestamped recording of an event.

Can be structured or unstructured.

Baggage

Contextual information that can be passed between signals.

OpenTelemetry

Telemetry Data /Signals

Traces

Path of a request through the entire system

Metrics

A measurement captured at runtime. eg.

http.requests.total 42
db.queries.count 501

Logs

A timestamped recording of an event.

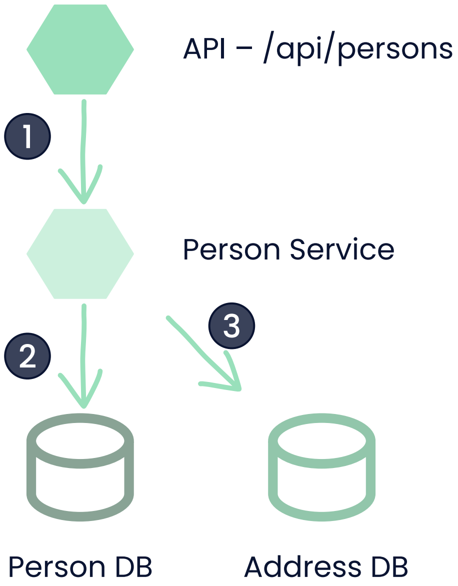
Can be structured or unstructured.

Baggage

Contextual information that can be passed between signals.

OpenTelemetry

What is a Trace?

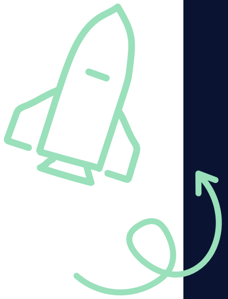


API – /api/persons

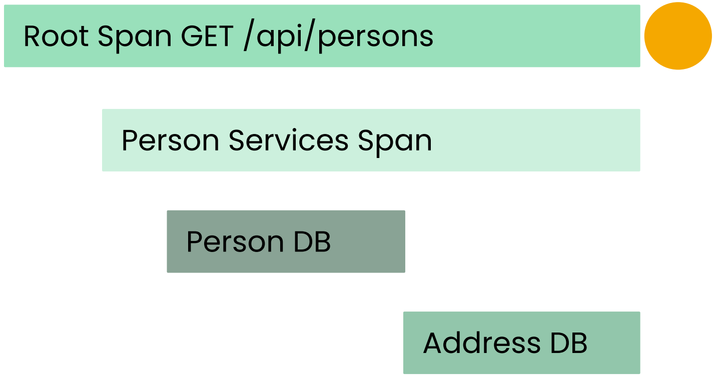
Person Service

Person DB

Address DB



OpenTelemetry
What is a Trace?

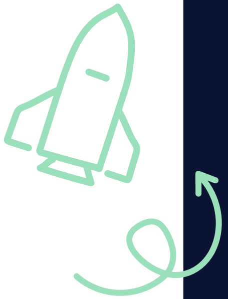

Root Span GET /api/persons

Person Services Span

Person DB

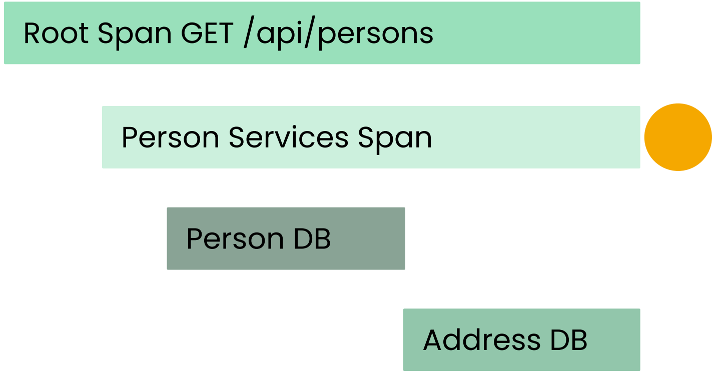
Address DB

```
{
  "name": "GET /api/persons",
  "context": {
    "trace_id": "5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2025-04-29T18:52:58.114201Z",
  "end_time": "2025-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "/api/persons",
    ...
  },
  "events": [
    {
      "name": "request.received",
      "timestamp": "2025-04-29T18:52:58.114561Z",
      "attributes": {
        "event_attributes": 1
      }
    }
  ]
}
```



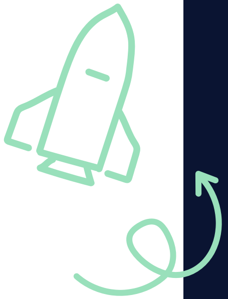
OpenTelemetry

What is a Trace?



```
{
  "name": "Call Person DB",
  "context": {
    "trace_id": "5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "5fb397be34d26b51"
  },
  "parent_id": "051581bf3cb55c13",
  "start_time": "2025-04-29T18:52:58.114304Z",
  "end_time": "2025-04-29T22:52:58.114561Z",
  "attributes": {
    "db.system": "mysql",
    "db.statement": "SELECT * FROM person WHERE id=?",

  }
}
```



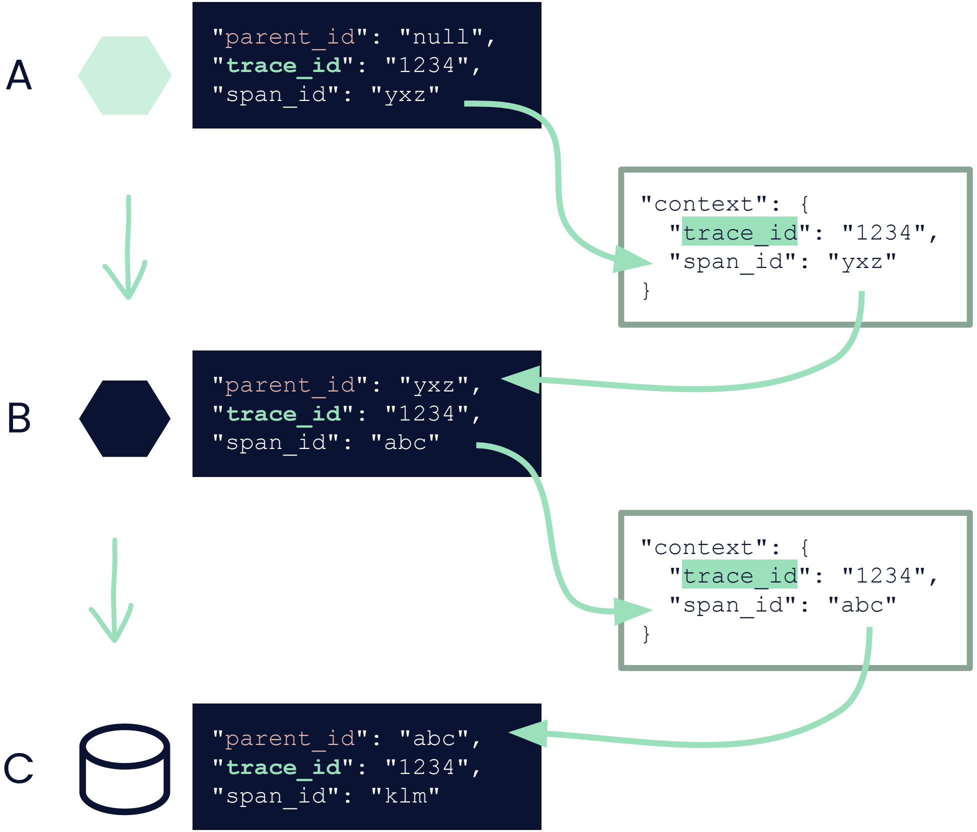
OpenTelemetry

Context Propagation

With context propagation, signals can be correlated with each other, regardless of where they are generated.

- Service A calls Service B includes a trace ID and a span ID
- Service B uses these values to create a new span that belongs to the same trace, setting the span from Service A as its parent.

Default propagator W3C TraceContext



```
"parent_id": "null",
"trace_id": "1234",
"span_id": "yxz"
```

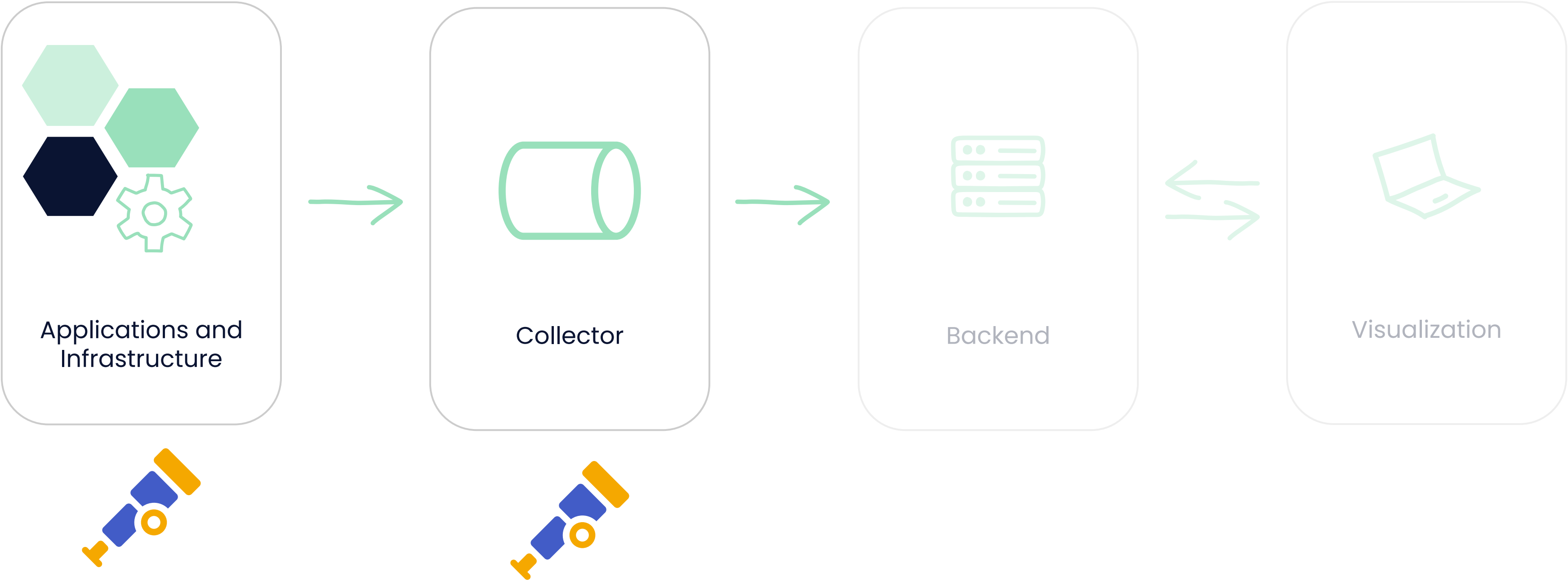
```
"context": {
  "trace_id": "1234",
  "span_id": "yxz"
}
```

```
"parent_id": "yxz",
"trace_id": "1234",
"span_id": "abc"
```

```
"context": {
  "trace_id": "1234",
  "span_id": "abc"
}
```

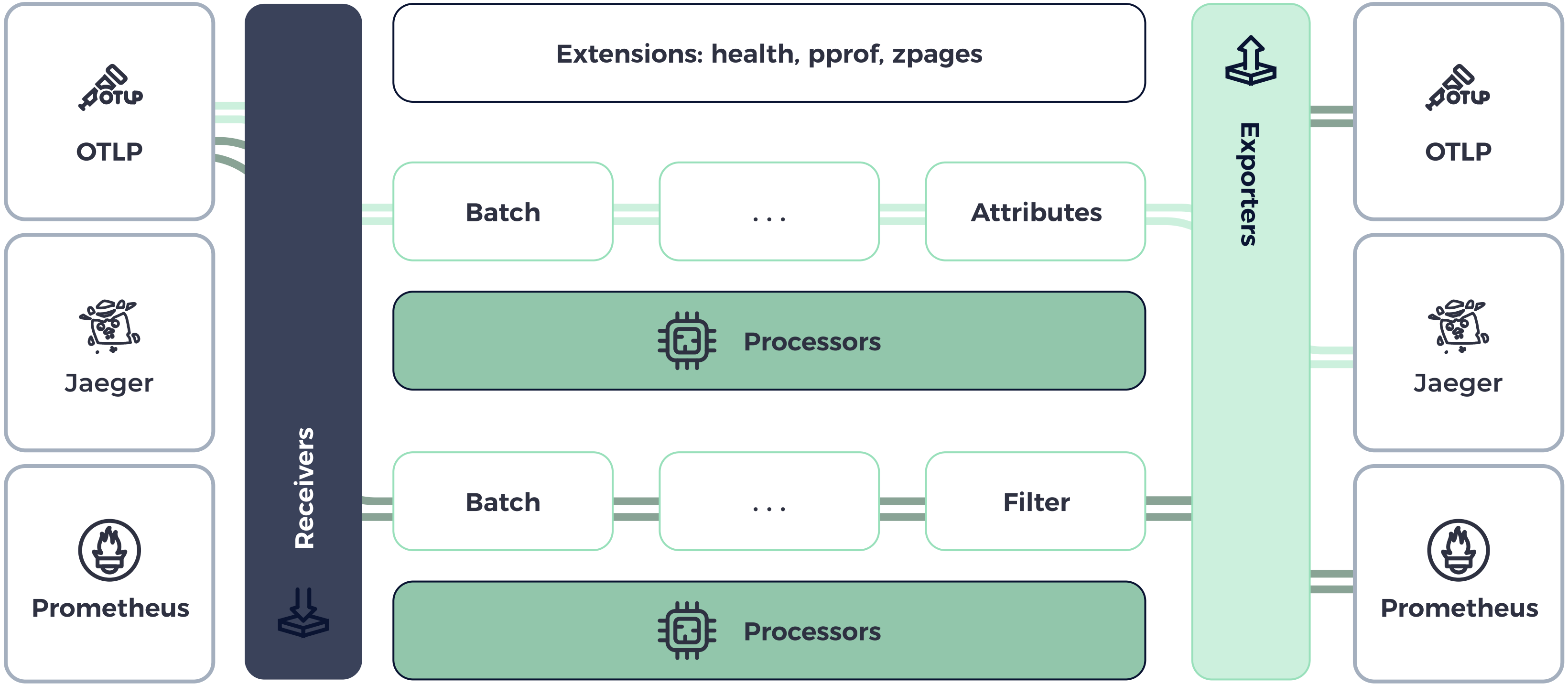
```
"parent_id": "abc",
"trace_id": "1234",
"span_id": "klm"
```

OpenTelemetry
Collector



tim&
koko

OpenTelemetry
Collector



OpenTelemetry
Collector Configuration

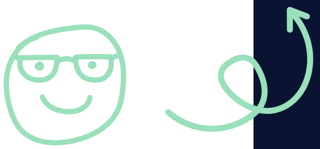
- Configure the collector setup and chain is straight forward.
- It contains of reasonable defaults by default.
- Decoupling and offering of open source observability data formats
- It is not necessary to send telemetry data through a collector, it is recommended.

```yaml
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 0.0.0.0:4317
      http:
        endpoint: 0.0.0.0:4318
processors:
  batch:

exporters:
  otlp:
    endpoint: otelcol:4317

extensions:
  health_check:
    endpoint: 0.0.0.0:13133
  pprof:
    endpoint: 0.0.0.0:1777
  zpages:
    endpoint: 0.0.0.0:55679

service:
  extensions: [health_check, pprof, zpages]
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp]
    metrics:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp]
    logs:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp]
```



tim&
koko

OpenTelemetry

Additional Components and Concepts

Platforms

- Client-side Apps
- FaaS
- Kubernetes, Helm Charts and Operator

OTel Specification

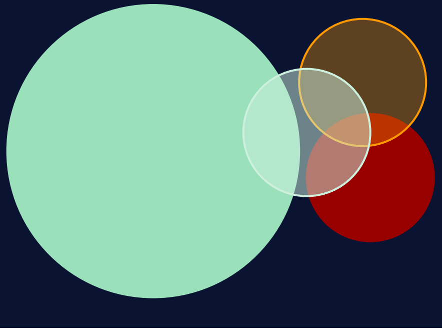
- OTel
- OTelP

Sampling

Most of the requests are successful and finish with acceptable latency and no errors.

Those might not be worth the cost.

Sampling is a way to reduce the amount of traces collected.

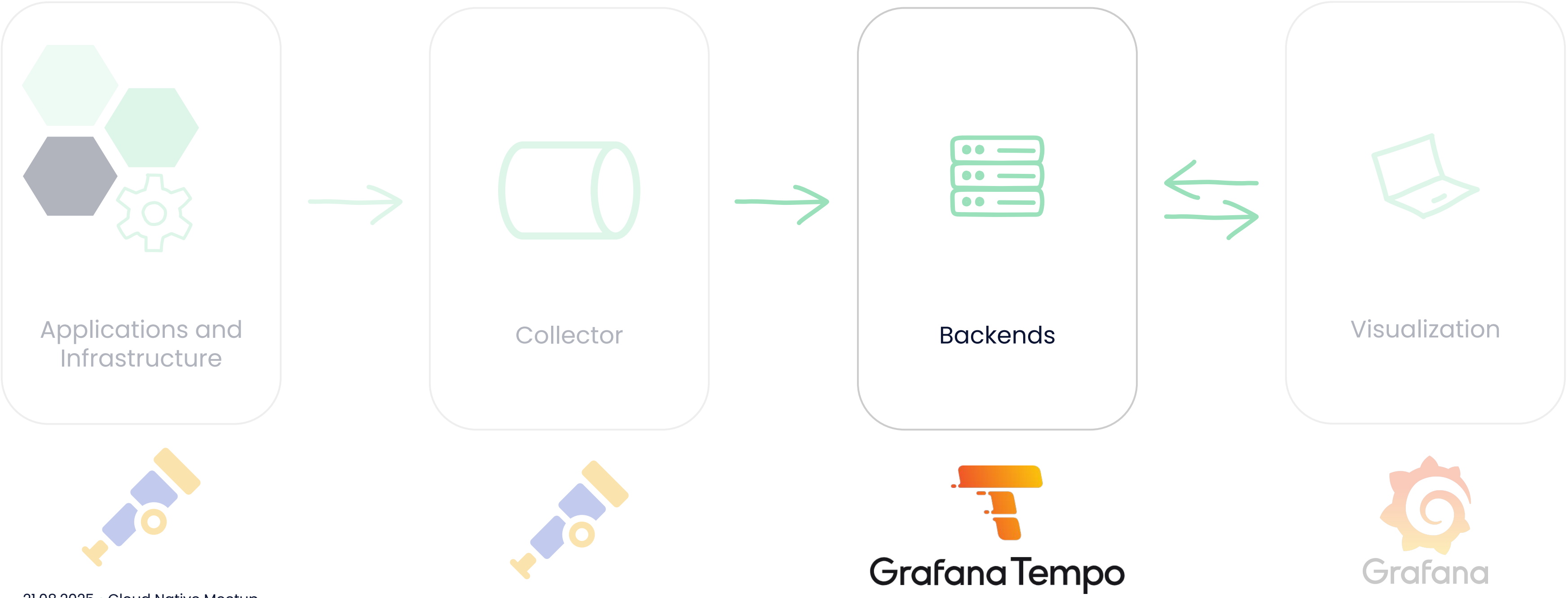


Distributed Tracing

Grafana Tempo

Grafana Tempo

Backend for Traces

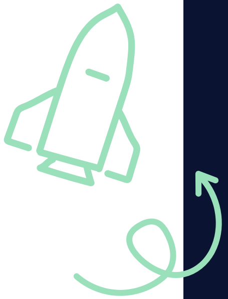


Grafana Tempo
Introduction

Tempo is cost-efficient, requiring only object storage to operate, and is deeply integrated with Grafana, Prometheus, and Loki. Tempo can ingest common open source tracing protocols, including Jaeger, Zipkin, and **OpenTelemetry**.

- Announced 2020
- GA 2021
- AGPLv3

https://grafana.com/oss/tempo/



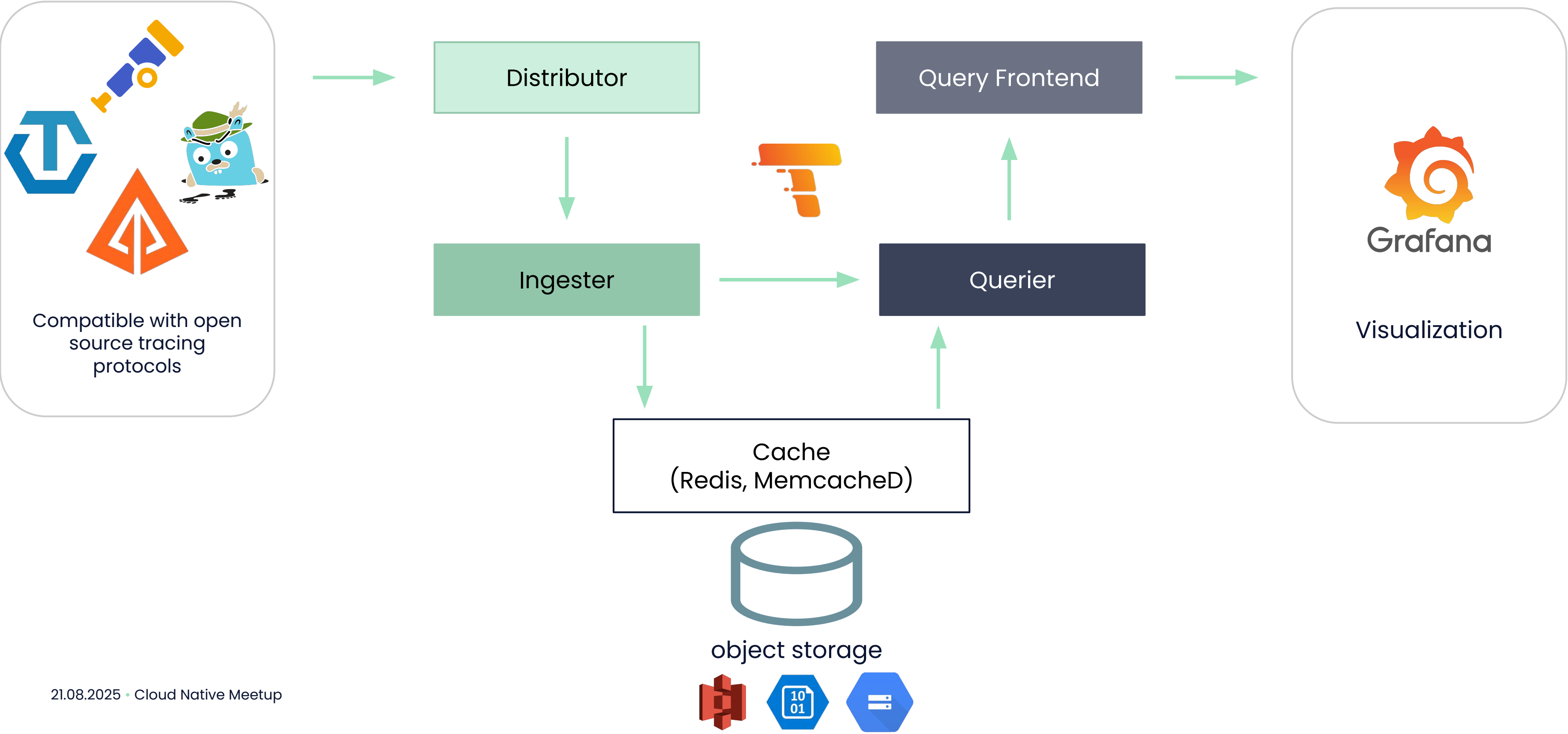
«Grafana Tempo is an open source, easy-to-use, and high-scale distributed tracing backend.»

- Built for massing scale: affordable long term storage
- Cost-effective: Traces are not indexed

Grafana Tempo

Grafana Tempo

Components / Architecture



Compatible with open source tracing protocols

Distributor

Ingester

Query Frontend

Querier

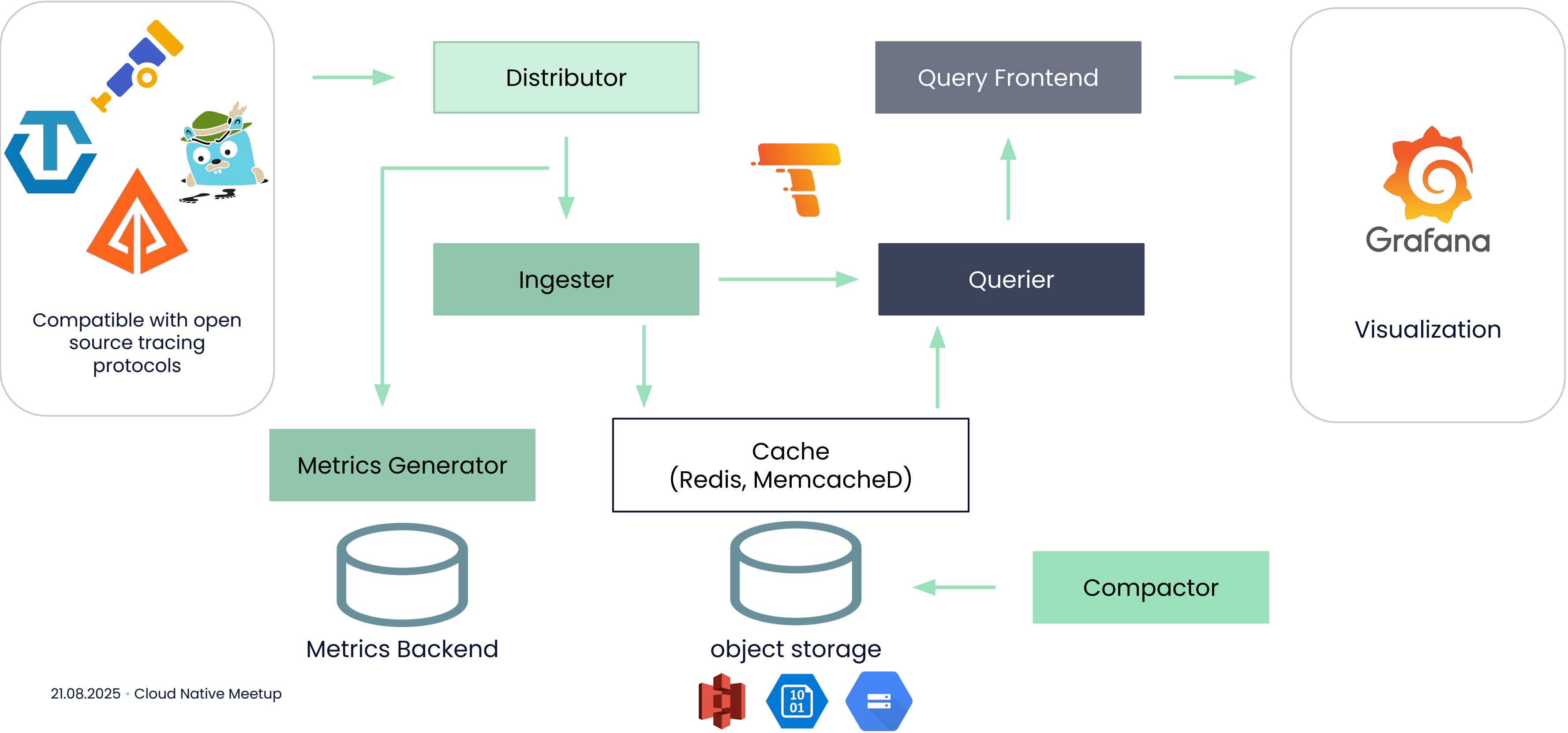
Cache
(Redis, MemcacheD)

object storage

Visualization

Grafana Tempo

Components / Architecture



Compatible with open source tracing protocols

Distributor

Ingester

Query Frontend

Querier

Metrics Generator

Cache
(Redis, MemcacheD)

Metrics Backend

object storage

Compactor

Visualization

Grafana Tempo – Deployment Variants

Kubernetes

Helm, Operator or Tanka (Jsonnet) based deployments on your kubernetes clusters.

Virtual Machine

Run a monolithic installation based on docker-compose on a linux virtual machine.

Grafana Cloud Traces

Grafana Tempo is part of the Grafana Cloud Services.

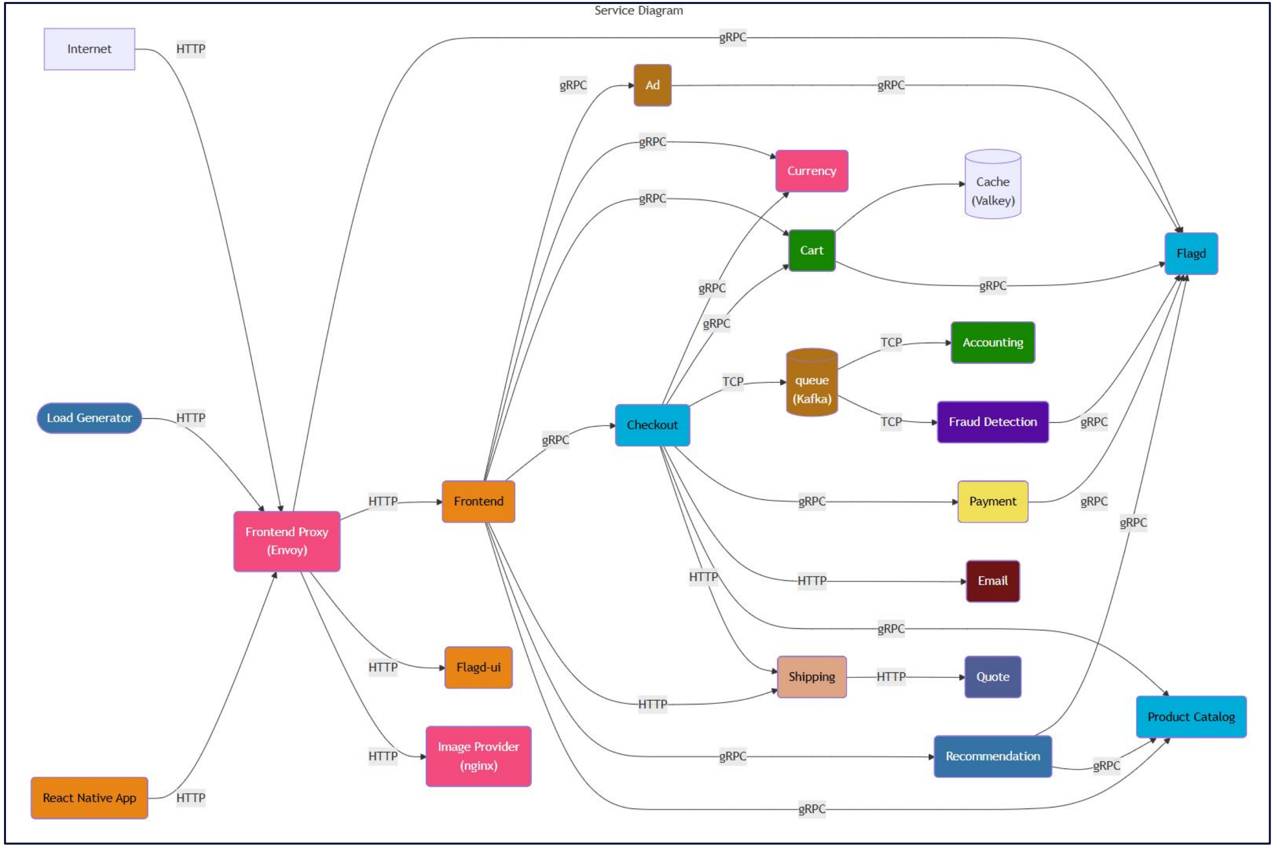
Pay as you go.

Distributed Tracing

Demo

https://opentelemetry.io/docs/demo/
https://github.com/tim-koko/opentelemetry-demo

Demo Application



.NET

C++

Go

Java

JavaScript

Kotlin

PHP

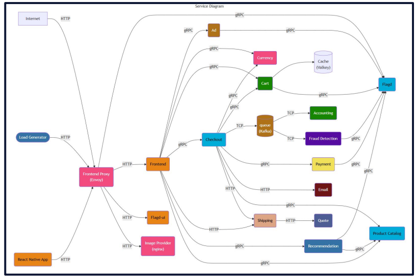
Python

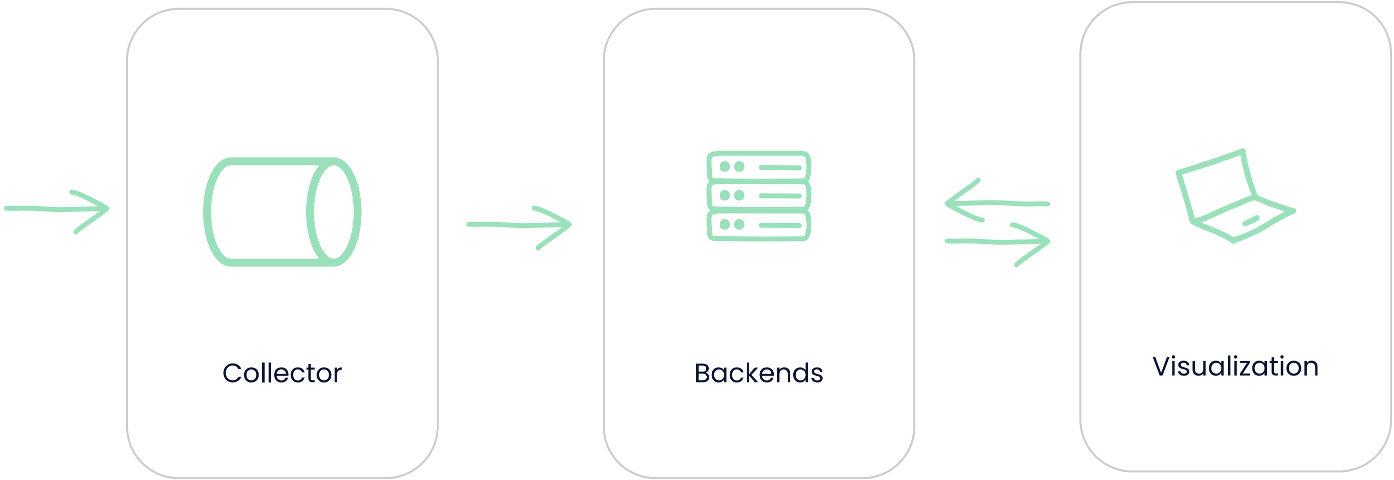
Ruby

Rust

TypeScript

Demo



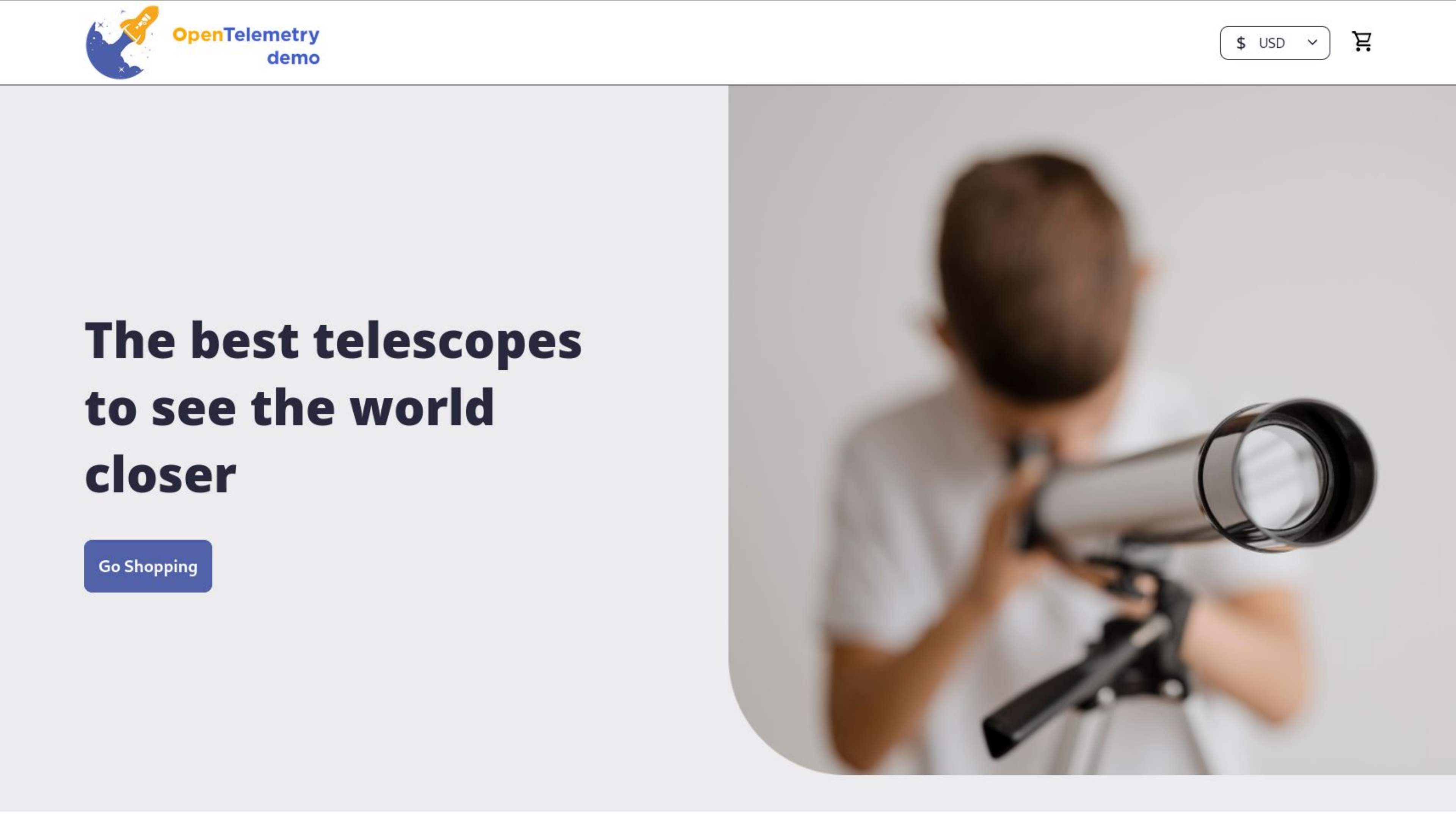

Collector

Backends

Visualization

Grafana Tempo

Grafana



The best telescopes to see the world closer

Hot Products

National Park Foundation Explorascope
$ 101.96

Starsense Explorer Refractor Telescope
$ 349.95

Eclipsmart Travel Refractor Telescope
$ 129.95

Lens Cleaning Kit
$ 21.95

Roof Binoculars
$ 209.95

Solar System Color Imager
$ 175.00

OpenTelemetry demo

$ USD

Optical Tube Assembly

Capturing impressive deep-sky astroimages is easier than ever with Rowe-Ackermann Schmidt Astrograph (RASA) V2, the perfect companion to today's top DSLR or astronomical CCD cameras. This fast, wide-field f/2.2 system allows for shorter exposure times compared to traditional f/10 astroimaging, without sacrificing resolution. Because shorter sub-exposure times are possible, your equatorial mount won't need to accurately track over extended periods. The short focal length also lessens equatorial tracking demands. In many cases, autoguiding will not be required.

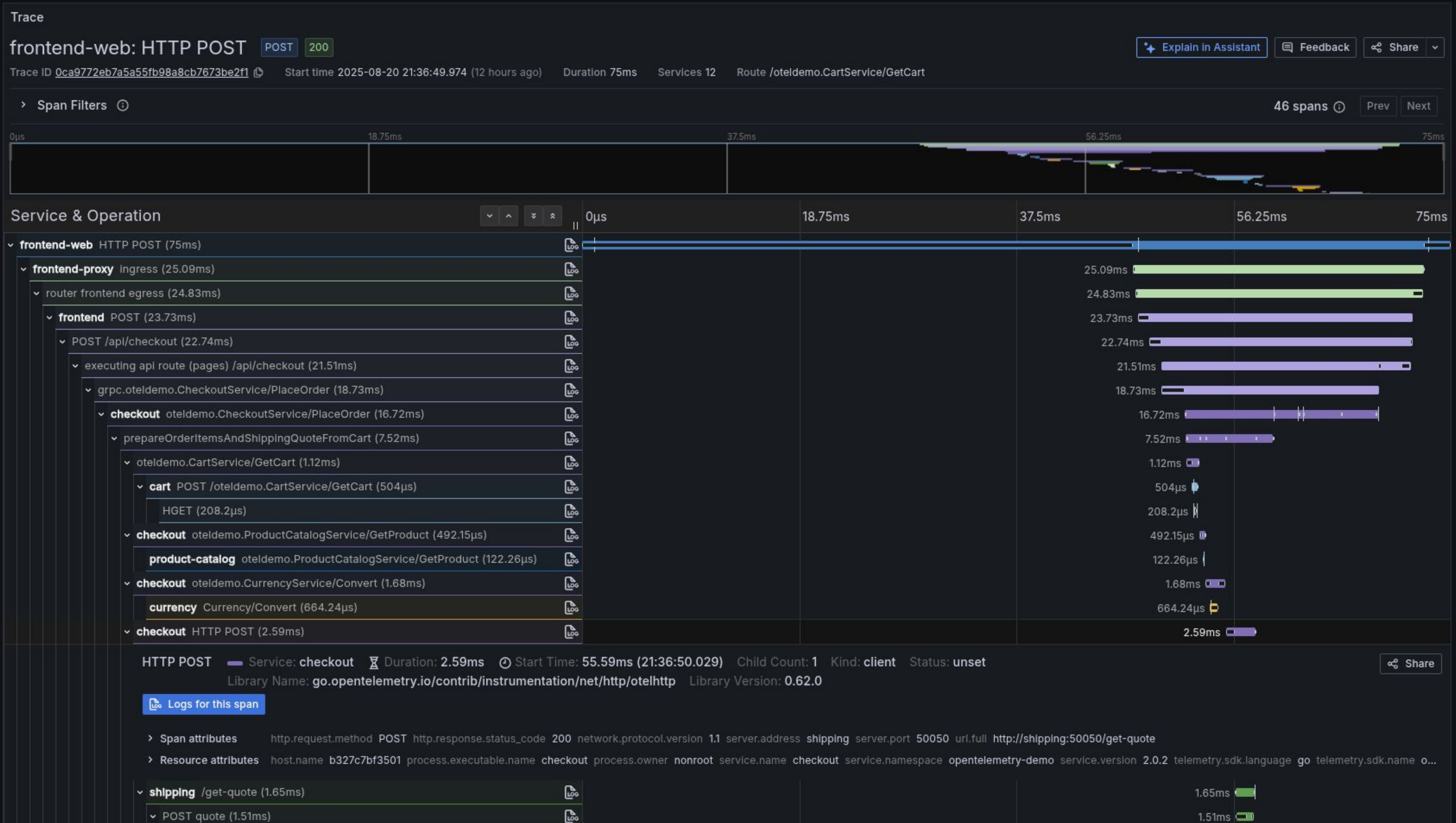
$ 3599.00

Quantity

1

Add To Cart

You May Also Like



Trace

frontend-web: HTTP POST POST 200

Trace ID 0ca9772eb7a5a55fb98a8cb7673be2f1 Start time 2025-08-20 21:36:49.974 (12 hours ago) Duration 75ms Services 12 Route /oteldemo.CartService/GetCart

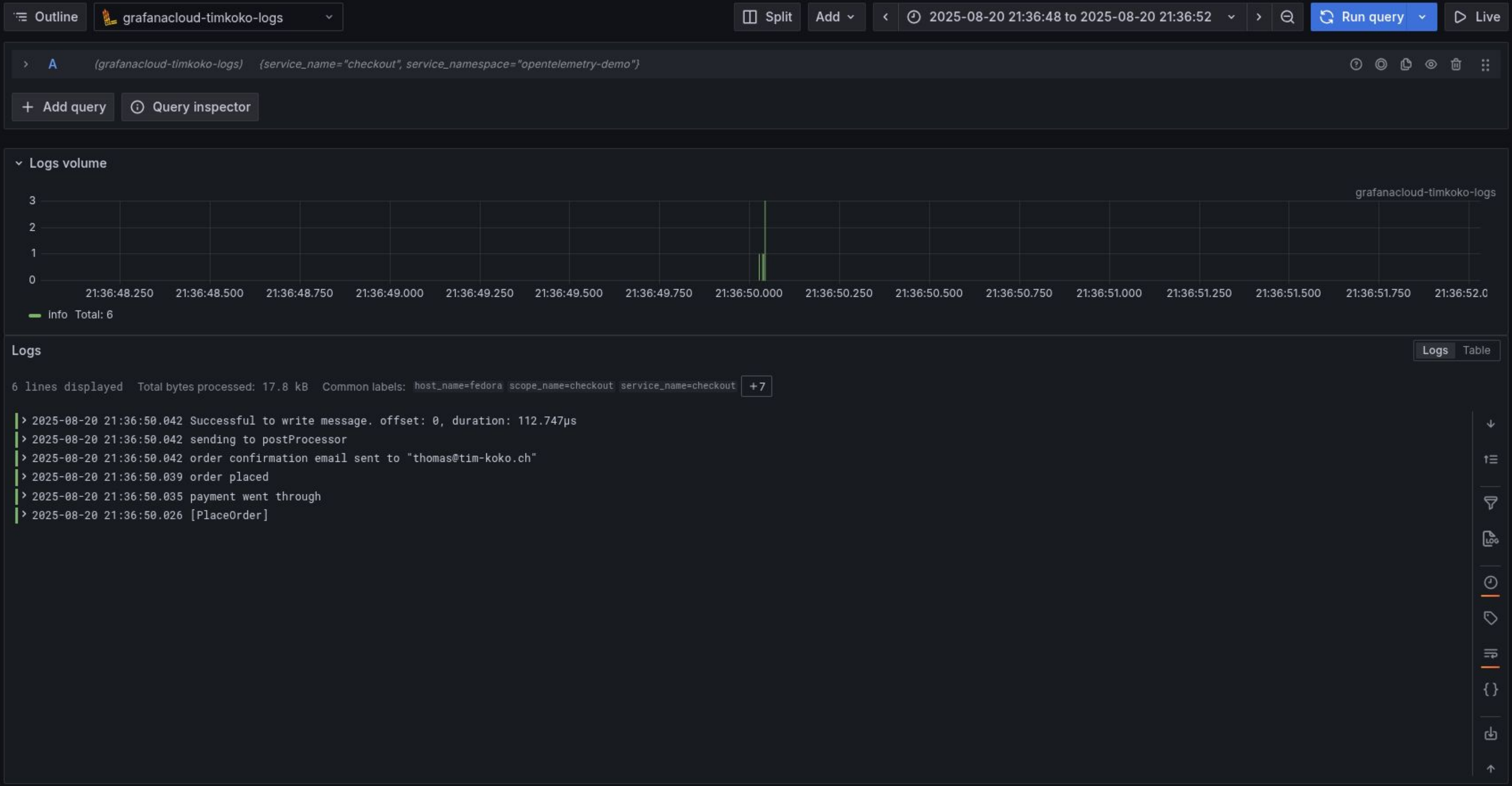
```
frontend-web HTTP POST (75ms)
  frontend-proxy ingress (25.09ms)                                          25.09ms
    router frontend egress (24.83ms)                                        24.83ms
      frontend POST (23.73ms)                                               23.73ms
        POST /api/checkout (22.74ms)                                        22.74ms
          executing api route (pages) /api/checkout (21.51ms)               21.51ms
            grpc.oteldemo.CheckoutService/PlaceOrder (18.73ms)              18.73ms
              checkout oteldemo.CheckoutService/PlaceOrder (16.72ms)        16.72ms
                prepareOrderItemsAndShippingQuoteFromCart (7.52ms)          7.52ms
                  oteldemo.CartService/GetCart (1.12ms)                     1.12ms
                    cart POST /oteldemo.CartService/GetCart (504µs)         504µs
                      HGET (208.2µs)                                        208.2µs
                  checkout oteldemo.ProductCatalogService/GetProduct (492.15µs)  492.15µs
                    product-catalog oteldemo.ProductCatalogService/GetProduct (122.26µs)  122.26µs
                  checkout oteldemo.CurrencyService/Convert (1.68ms)        1.68ms
                    currency Currency/Convert (664.24µs)                    664.24µs
                  checkout HTTP POST (2.59ms)                               2.59ms
```

HTTP POST Service: checkout Duration: 2.59ms Start Time: 55.59ms (21:36:50.029) Child Count: 1 Kind: client Status: unset
Library Name: go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp Library Version: 0.62.0

Span attributes http.request.method POST http.response.status_code 200 network.protocol.version 1.1 server.address shipping server.port 50050 url.full http://shipping:50050/get-quote

Resource attributes host.name b327c7bf3501 process.executable.name checkout process.owner nonroot service.name checkout service.namespace opentelemetry-demo service.version 2.0.2 telemetry.sdk.language go telemetry.sdk.name o...

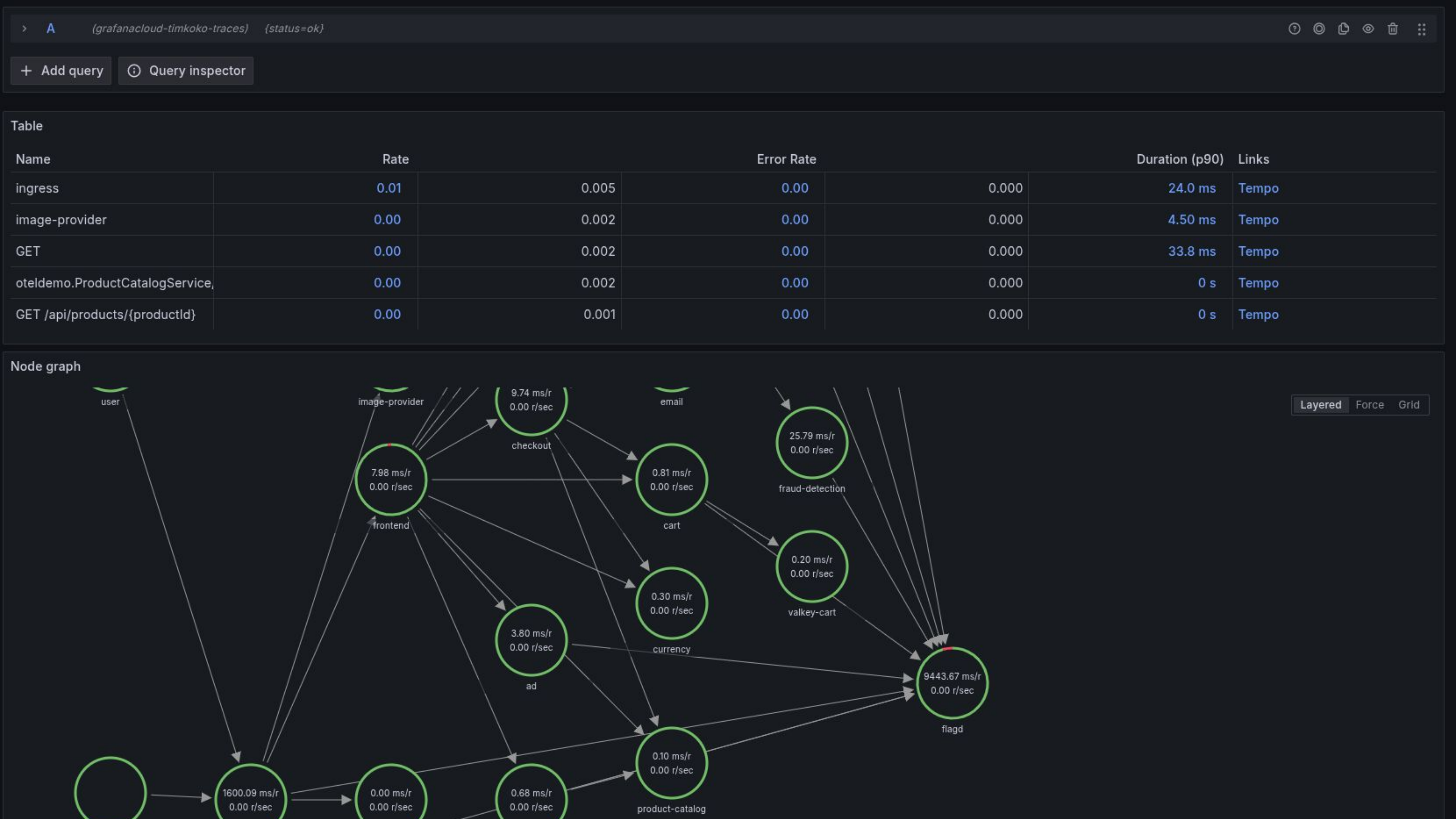
```
                    shipping /get-quote (1.65ms)                            1.65ms
                      POST quote (1.51ms)                                   1.51ms
```



```
(grafanacloud-timkoko-logs)    {service_name="checkout", service_namespace="opentelemetry-demo"}
```

6 lines displayed   Total bytes processed: 17.8 kB   Common labels: host_name=fedora scope_name=checkout service_name=checkout +7

```
> 2025-08-20 21:36:50.042 Successful to write message. offset: 0, duration: 112.747µs
> 2025-08-20 21:36:50.042 sending to postProcessor
> 2025-08-20 21:36:50.042 order confirmation email sent to "thomas@tim-koko.ch"
> 2025-08-20 21:36:50.039 order placed
> 2025-08-20 21:36:50.035 payment went through
> 2025-08-20 21:36:50.026 [PlaceOrder]
```



A (grafanacloud-timkoko-traces) {status=ok}

+ Add query

ⓘ Query inspector

Table

Name	Rate		Error Rate		Duration (p90)	Links
ingress	0.01	0.005	0.00	0.000	24.0 ms	Tempo
image-provider	0.00	0.002	0.00	0.000	4.50 ms	Tempo
GET	0.00	0.002	0.00	0.000	33.8 ms	Tempo
oteldemo.ProductCatalogService,	0.00	0.002	0.00	0.000	0 s	Tempo
GET /api/products/{productId}	0.00	0.001	0.00	0.000	0 s	Tempo

Node graph

Distributed Tracing

Distributed Tracing Stack in Practice

Tracing Platform PoC

based on OpenTelemetry and Grafana Tempo

Baloise faced the challenge that their existing solution was costly, had low adoption, and lacked self-service capabilities. Their objective was to assess a cloud-native alternative to the proprietary system in place.

- PoC was a success
- Familiar user experience
- Full ownership
- Open standards and vendor independence

https://tim-koko.ch/en/references/baloise-distributed-tracing-opentelemetry/

Distributed Tracing Made Easy

Learnings

Observability is key

Observability is key in distributed systems to understand what is going on. Distributed systems are complex, so is distributed tracing. It is a platform topic, but is much closer to the applications than you might guess.

Correlate Signals and Levels

The correlation of observability signals is extremely valuable and can be achieved on several levels. Signals have different goals and requirements, which support the approach of different backends.

Stick to standards

Decoupling your business from vendors using open standards and open source.
Building your own open tracing and observability platform come with costs, though...

About us

tim&koko

We are hiring!

tim-koko.ch/jobs

Thomas Philipona
thomas@tim-koko.ch
+41 79 325 55 83

Thank you for your attention

Besuche uns auf

tim-koko.ch