Modern Application Debugging: An Intro to OpenTelemetry

Josh Lee • Altinity • May 19 2025 • Red Hat Community Day

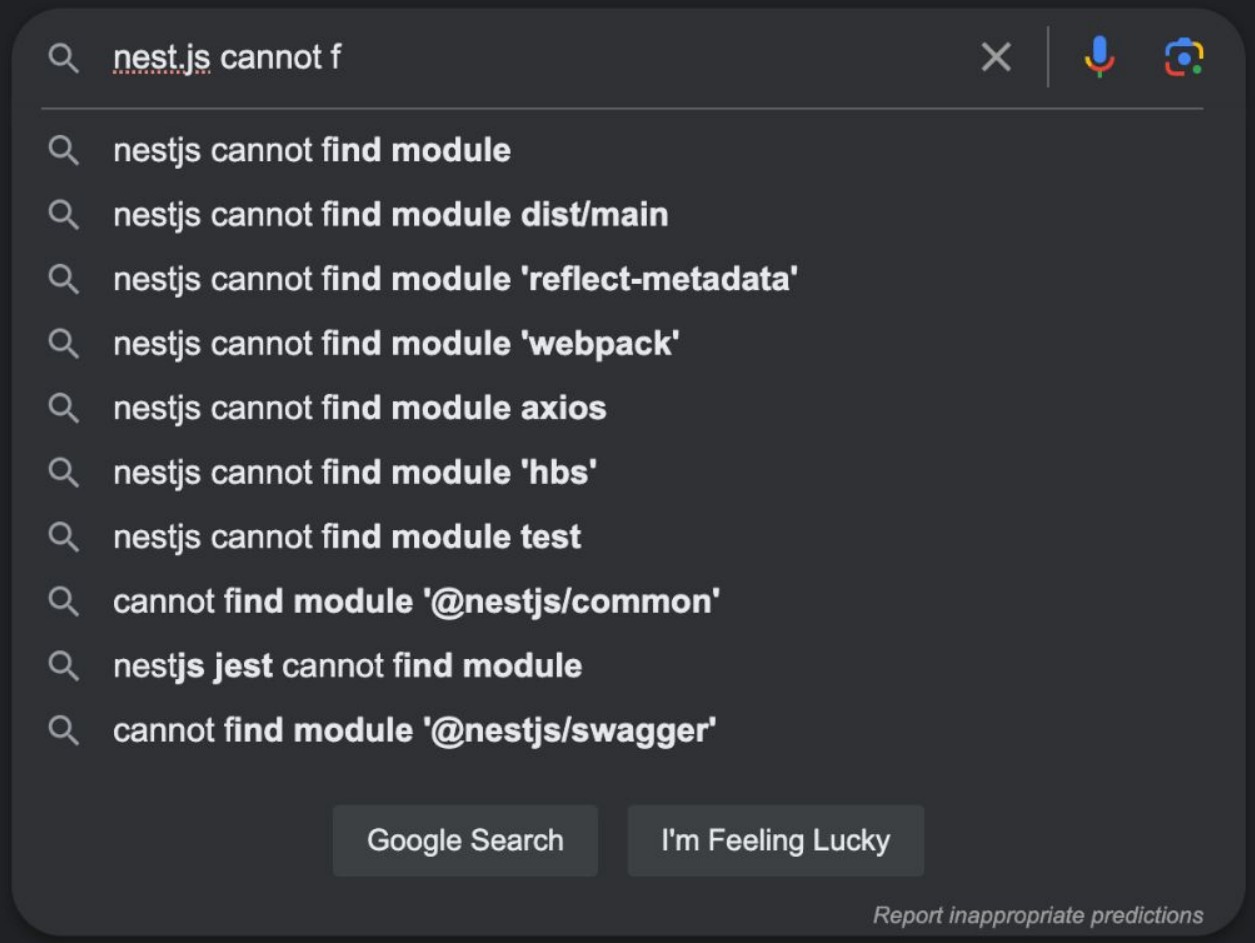
Josh Lee

Open Source Advocate

Altinity

We are but humble open source contributors

How do we debug our applications?


Google
nest.js cannot f
nestjs cannot find module
nestjs cannot find module dist/main
nestjs cannot find module 'reflect-metadata'
nestjs cannot find module 'webpack'
nestjs cannot find module axios
nestjs cannot find module 'hbs'
nestjs cannot find module test
cannot find module '@nestjs/common'
nestjs jest cannot find module
cannot find module '@nestjs/swagger'
Google Search
I'm Feeling Lucky
Report inappropriate predictions

Observability is our ability to understand a system from its outputs alone

“There are only two signals: metrics and (structured) logs”

— paraphrased from Charity Majors, Honeycomb

A Typical Request Log

```
2024-07-01 09:35:34 GET /home 200 ...
```

Adding Duration

```
2024-07-01 09:35:34 231ms GET /home 200
```

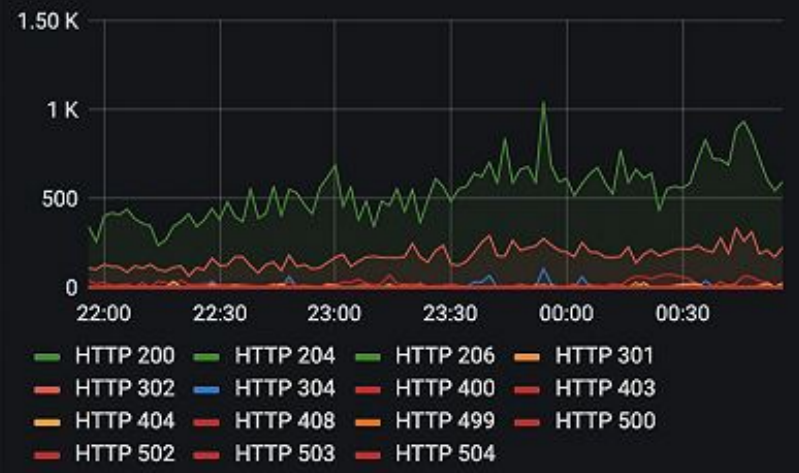
Controller Class: All | Namespace: live | Ingress: All | Ingress Pod: All | Config Reloads

Overview

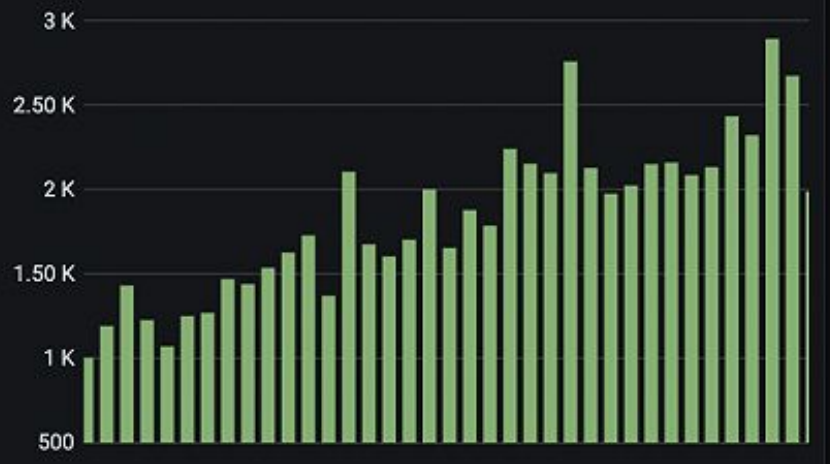
Requests (period)	% Succe...	Conns (...	Reqs (2...	% Success (2m)	HTTP 1/2xx (2m)	HTTP 3xx (2m)	HTTP 4xx (2m)	HTTP 5xx (2m)
67.3 K	97.0%	311	850	99.3%	592	228	30	0

HTTP Requests / Ingress

HTTP Status Codes



Total HTTP Requests

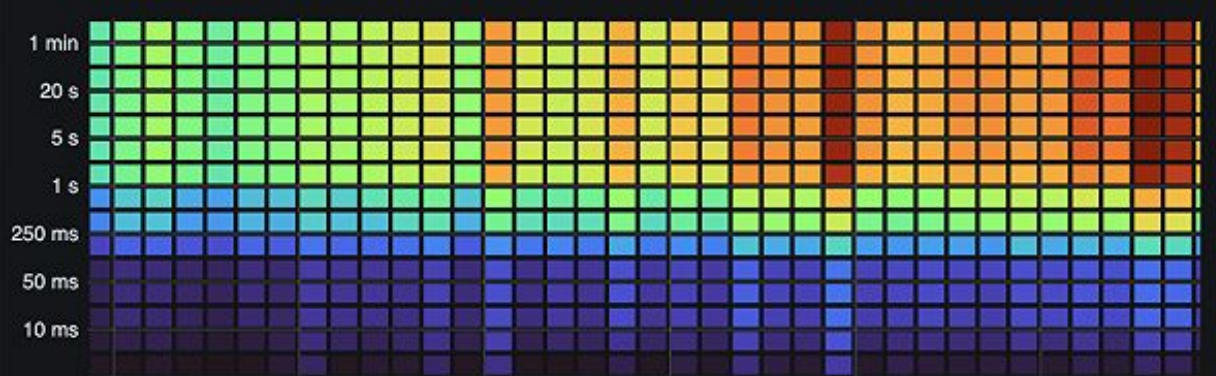


Latency

Latency (Average Percentiles)

	min	max	avg
0.99	4.3 s	42.3 s	6.5 s
0.95	2.4 s	4.5 s	2.8 s
0.9	2.1 s	2.7 s	2.3 s
0.5	357.2 ms	631.6 ms	443.7 ms
0.75	1.0 s	1.8 s	1.5 s
Average	3.2 s	5.1 s	4.1 s

Latency Heatmap



Back to our log…

```
2024-07-01 09:35:34 231ms GET /home 200
```

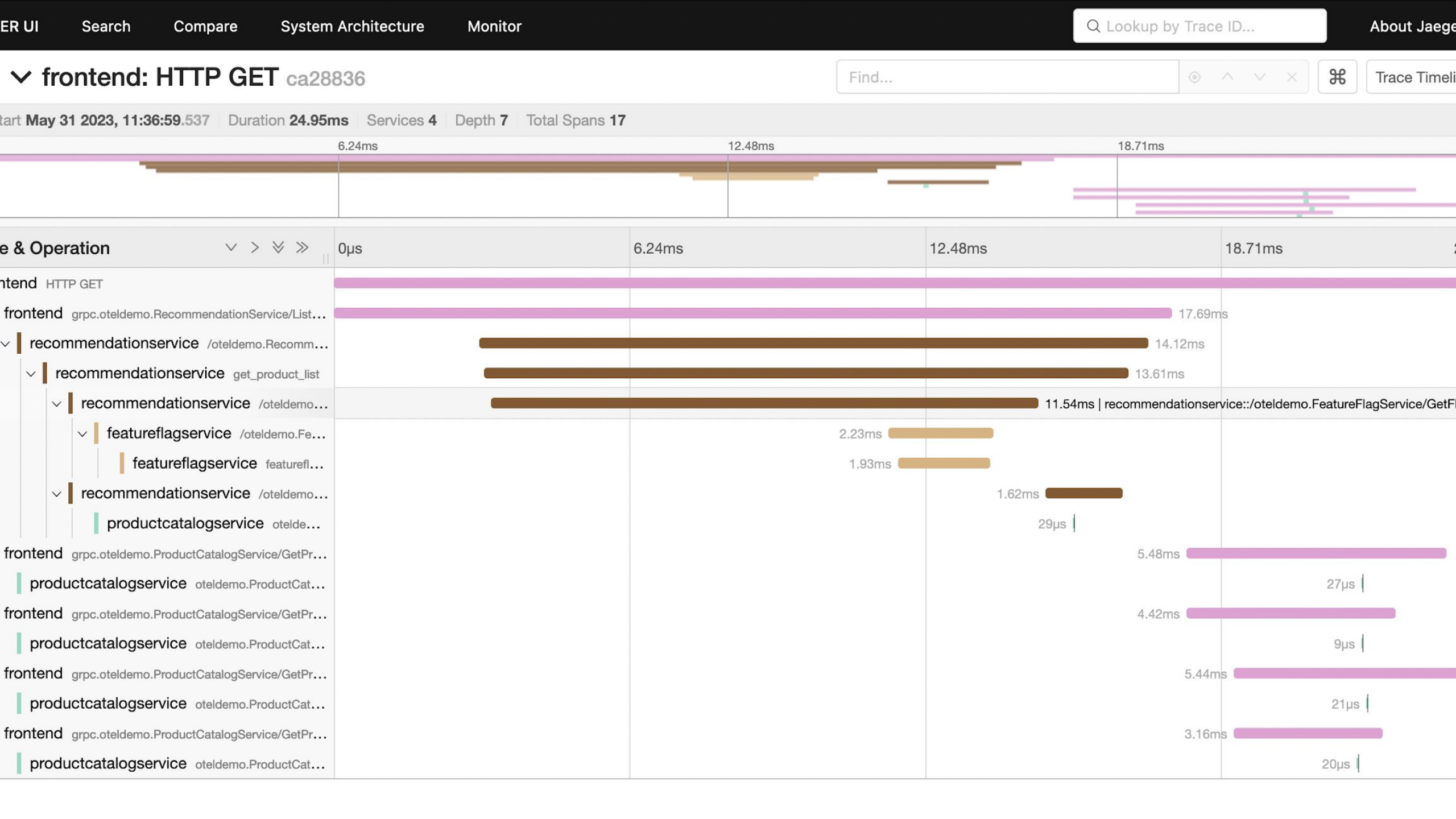
Back to our log…

```
Request:123 2024-07-01 09:35:34 231ms GET /home
200
```

Connecting the trace:

```
Trace:4ea3 Span:123 2024-07-01 09:35:34 231ms GET
/home 200

Trace:4ea3 Span:456 ParentSpan:123 2024-07-01
09:35:34 201ms GET /api/users 201
```


ER UI
Search
Compare
System Architecture
Monitor
Lookup by Trace ID...
About Jaege
frontend: HTTP GET ca28836
Find...
Trace Timeli
tart May 31 2023, 11:36:59.537
Duration 24.95ms
Services 4
Depth 7
Total Spans 17
6.24ms
12.48ms
18.71ms
e & Operation
0µs
6.24ms
12.48ms
18.71ms
ntend HTTP GET
frontend grpc.oteldemo.RecommendationService/List...
17.69ms
recommendationservice /oteldemo.Recomm...
14.12ms
recommendationservice get_product_list
13.61ms
recommendationservice /oteldemo...
11.54ms | recommendationservice::/oteldemo.FeatureFlagService/GetF
featureflagservice /oteldemo.Fe...
2.23ms
featureflagservice featurefl...
1.93ms
recommendationservice /oteldemo...
1.62ms
productcatalogservice otelde...
29µs
frontend grpc.oteldemo.ProductCatalogService/GetPr...
5.48ms
productcatalogservice oteldemo.ProductCat...
27µs
frontend grpc.oteldemo.ProductCatalogService/GetPr...
4.42ms
productcatalogservice oteldemo.ProductCat...
9µs
frontend grpc.oteldemo.ProductCatalogService/GetPr...
5.44ms
productcatalogservice oteldemo.ProductCat...
21µs
frontend grpc.oteldemo.ProductCatalogService/GetPr...
3.16ms
productcatalogservice oteldemo.ProductCat...
20µs

Observability is not any one signal…

Metrics

Aggregable

Is there a problem?

Traces

Request-Scoped

Where is the problem?

Logs

Verbose, time-stamped records

What is the problem?

Distributed Tracing is the *"Killer App"*

Understand complete request flows

Create a real-time map of system topology and dependencies

Derive metrics from the richness of trace metadata

Enrich logs and metrics with context

17

Introducing OpenTelemetry

OpenTelemetry Language-Specific SDKs

Metrics

Stable SDKs:
C++
.NET
Go
Java
JavaScript
Python
PHP

Development/Beta SDKs:
Erlang/Elixir
Ruby
Rust
Swift

Traces

Stable SDKs:
C++
.NET
Erlang / Elixir
Go
Java
JavaScript
Python
Ruby
PHP
Swift

Beta SDKs:
Rust

Logs

Stable SDKs:
C++
.NET
Java
PHP

Development/Beta SDKs:
Erlang / Elixir
Go
JavaScript
Python
Ruby
Rust
Swift

What is OpenTelemetry?

Specifications

- W3C Trace Context
- Language APIs
- OTLP
- Semantic Conventions

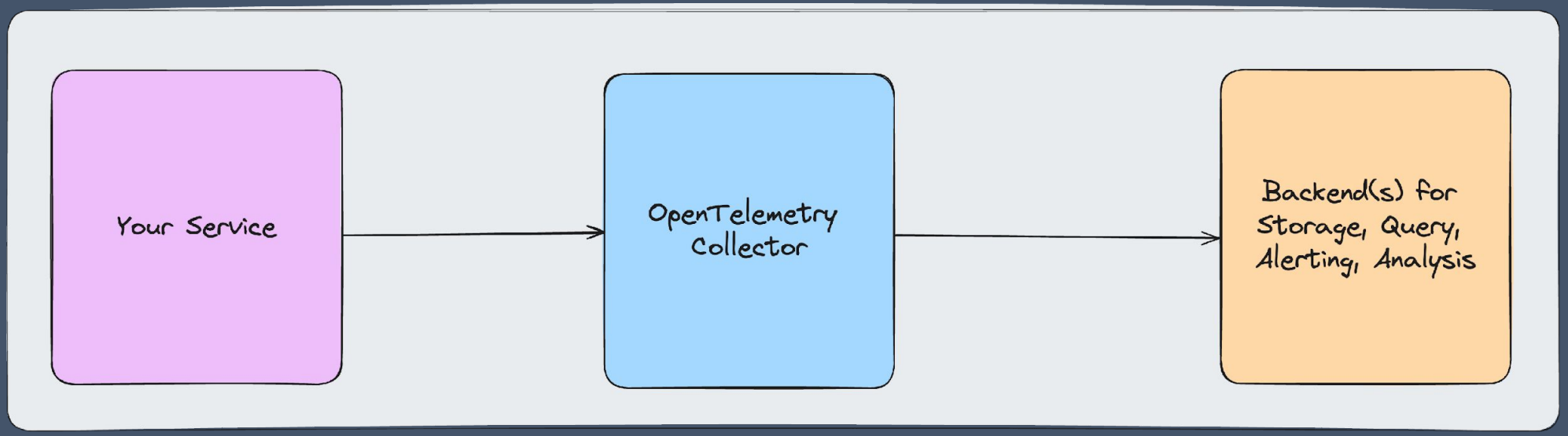
Libraries & Tools

- Language SDKs
- Instrumentation Libraries
- The Collector
- Kubernetes Operator

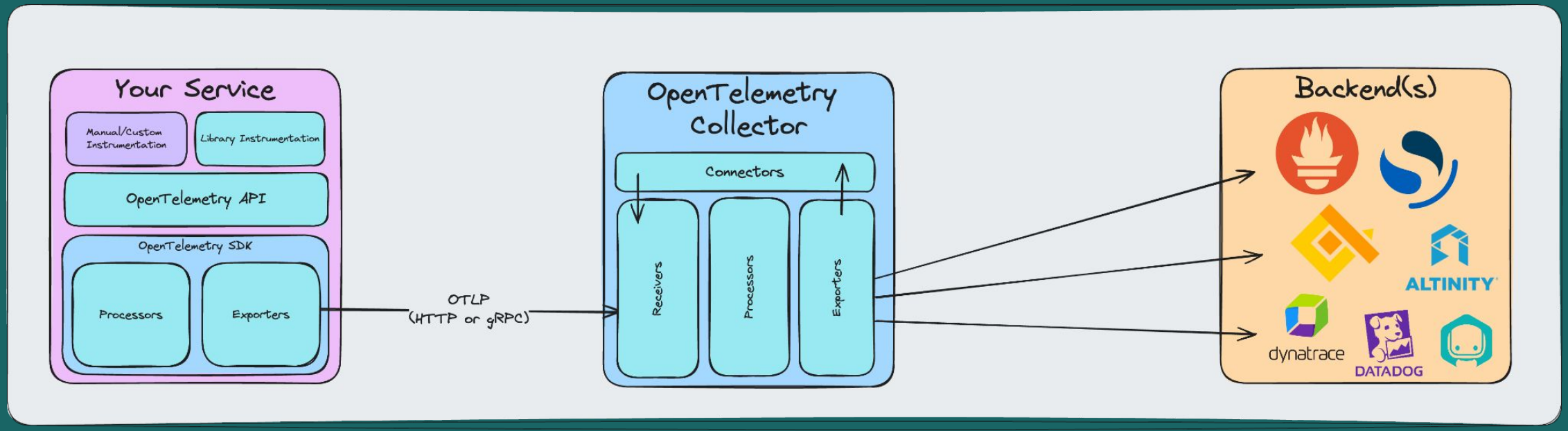
Community

- CNCF
- Events & Meetups
- OpenTelemetry End User WG (+ other WGs/SIGs)

OpenTelemetry Stack



OpenTelemetry Stack



OpenTelemetry Stack

In your code

- ➔ Language API & SDK
- ➔ Processors & exporters
- ➔ Instrumentation libraries
- ➔ Manual instrumentation

On your node*

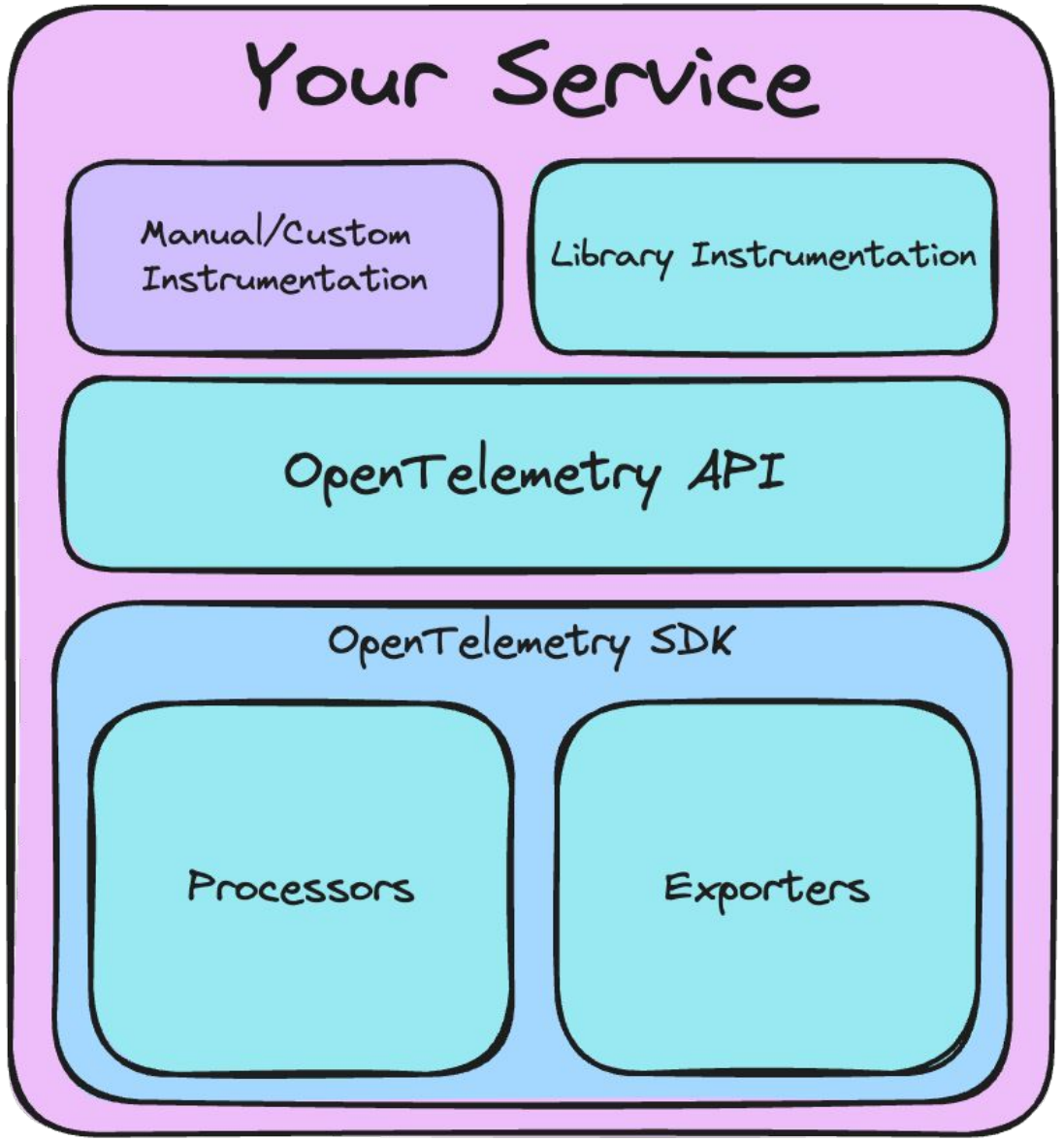
- ➔ The OpenTelemetry Collector
- ➔ Receivers, processors, & exporters
- ➔ Host metrics & logs
- ➔ Automatic instrumentation

Somewhere else…

Storage, querying, analysis, alerting
e.g. Prometheus, Grafana, Jaeger, OpenSearch, ClickHouse, DataDog…

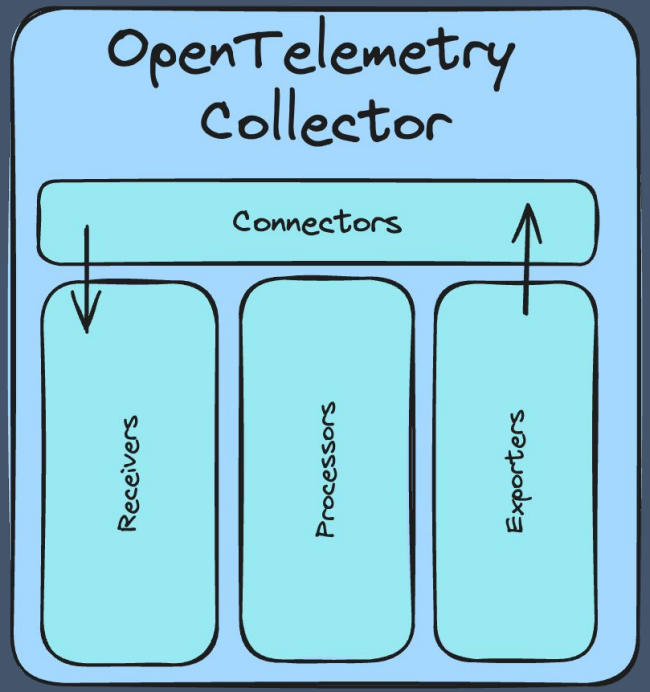
In your code

- Explicit API Calls / Manual Instrumentation
- Instrumentation Libraries
- OpenTelemetry API
- OpenTelemetry SDK



The OpenTelemetry Collector

The OpenTelemetry Collector



The OpenTelemetry Collector

Gather and forward telemetry

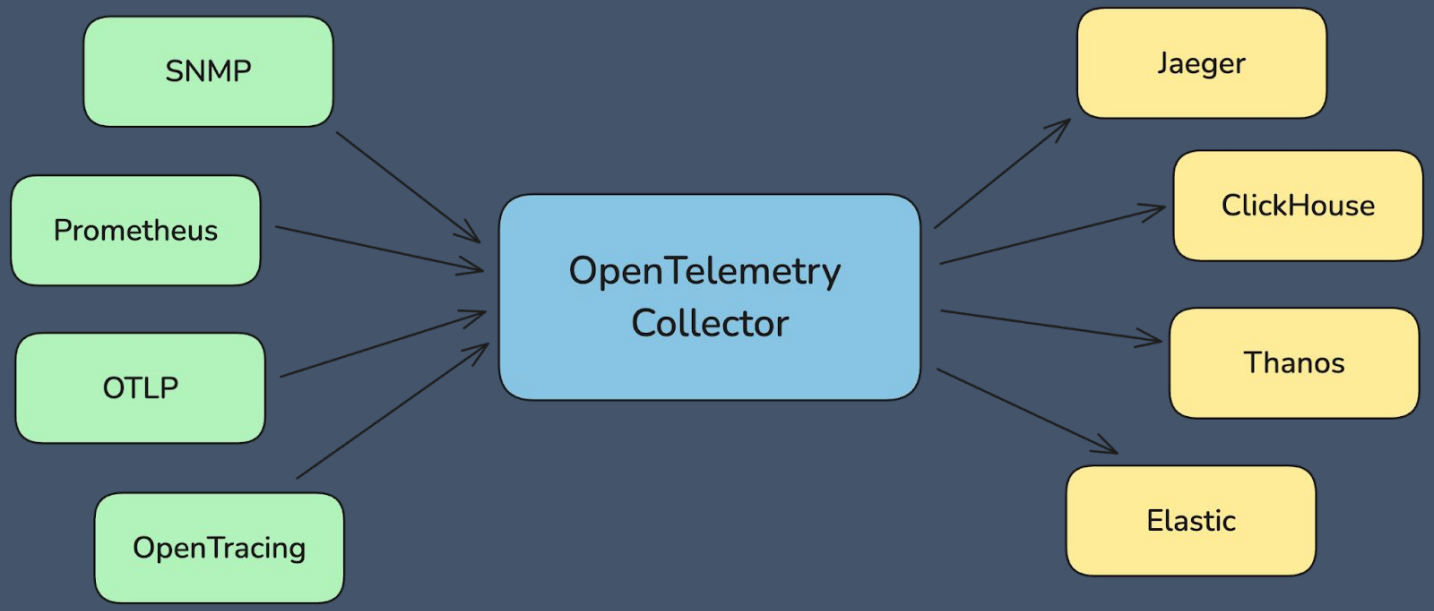
Apply filtering, sampling, and batching rules

Translate between any compatible sources and destinations

Gather node- and cluster-level telemetry

Deep dive article

The OpenTelemetry Collector



Using OpenTelemetry

1

Instrument your application using the OTel API, SDK, and instrumentation libraries for your language.

2

Add additional manual instrumentation and context.

3

Collect and process the exported data with the OpenTelemetry Collector.

4

Forward the data to your backend(s) of choice for storage and analysis.

1. Instrument your application using the OTel API, SDK, and instrumentation libraries for your language.

```
const opentelemetry = require('@opentelemetry/sdk-node');
const {getNodeAutoInstrumentations} =
   require('@opentelemetry/auto-instrumentations-node');
const {OTLPTraceExporter} = require('@opentelemetry/exporter-trace-otlp-grpc');
const {OTLPMetricExporter} = require('@opentelemetry/exporter-metrics-otlp-grpc');
const {PeriodicExportingMetricReader} = require('@opentelemetry/sdk-metrics');
```

2. Add additional manual instrumentation and context

```
1 const span = trace.getSpan(context.active()) as Span;
2 span.setAttribute("ProductId", productId);
```

3. Gather and process telemetry with the Collector

```
receivers:
    prometheus:
      config:
        scrape_configs:
          - job_name: k8s
            kubernetes_sd_configs:
            - role: pod
            metric_relabel_configs:
            - source_labels: [__name__]
              regex: "(request_duration_seconds.*|response_duration_seconds.*)"
              action: keep
```

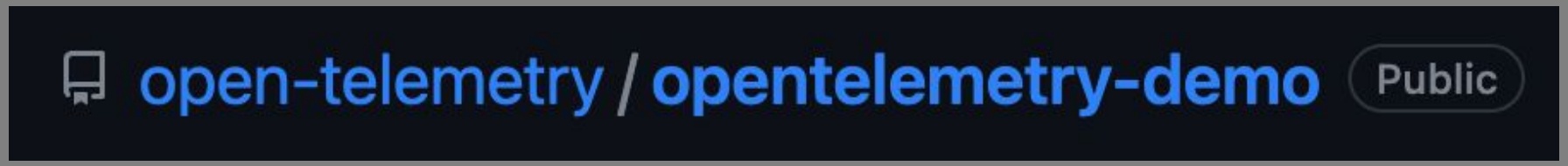
4. Export the telemetry to your backend(s) of choice

```yaml
exporters:
  clickhouse:
    endpoint: tcp://127.0.0.1:9000?dial_timeout=10s
    database: otel
    async_insert: true
```

EVERYTHING IS AWESOME!

Pitfalls & Warnings

- Many libraries and tools to stitch together
- Specifications are semi-stable
- SDKs are idiomatic to the source language
- Duplicate data can confuse tools
- Preference for high-cardinality, low-granularity time-series metrics
- ~~Lack of examples~~

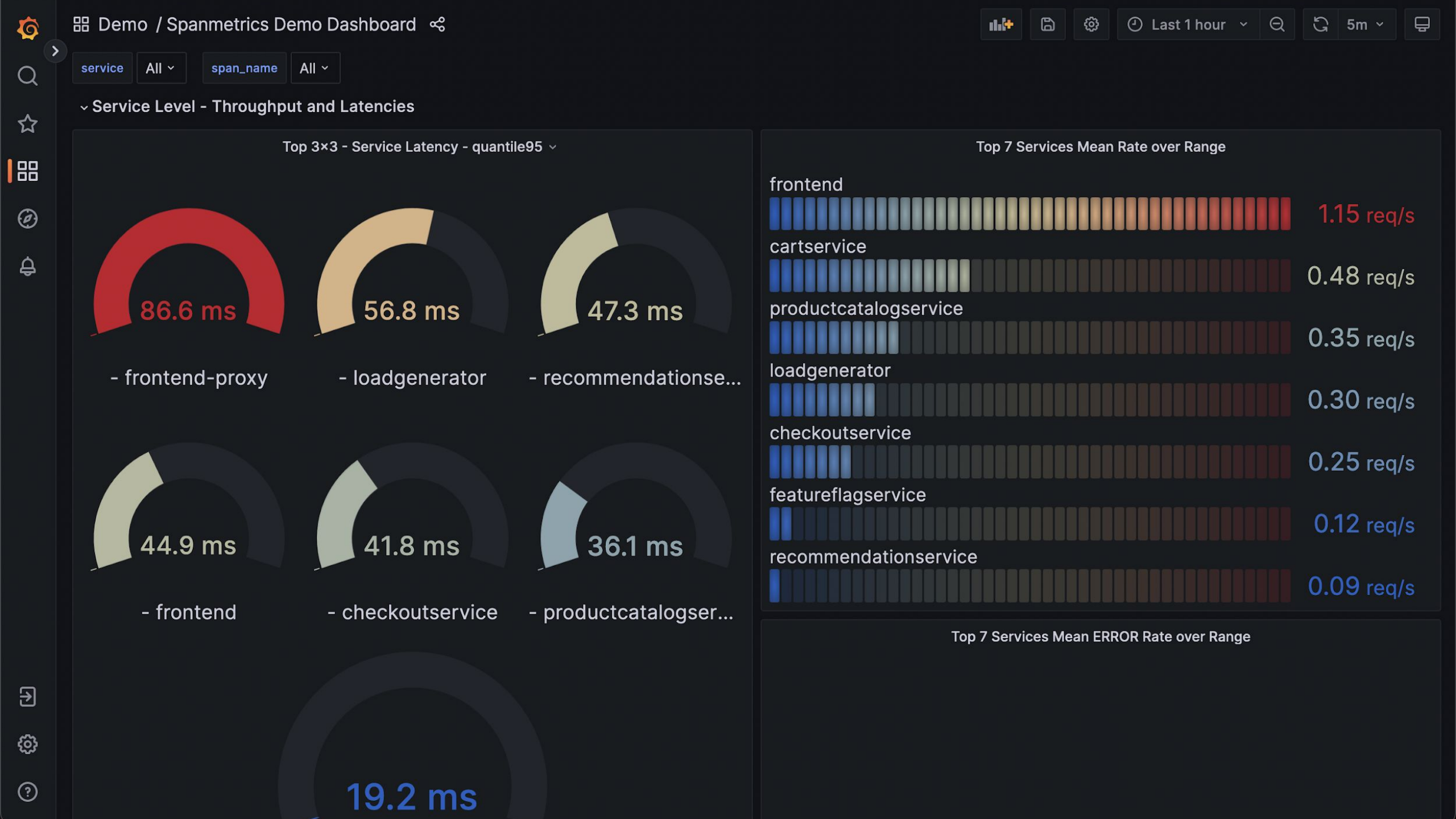

open-telemetry / opentelemetry-demo
Public

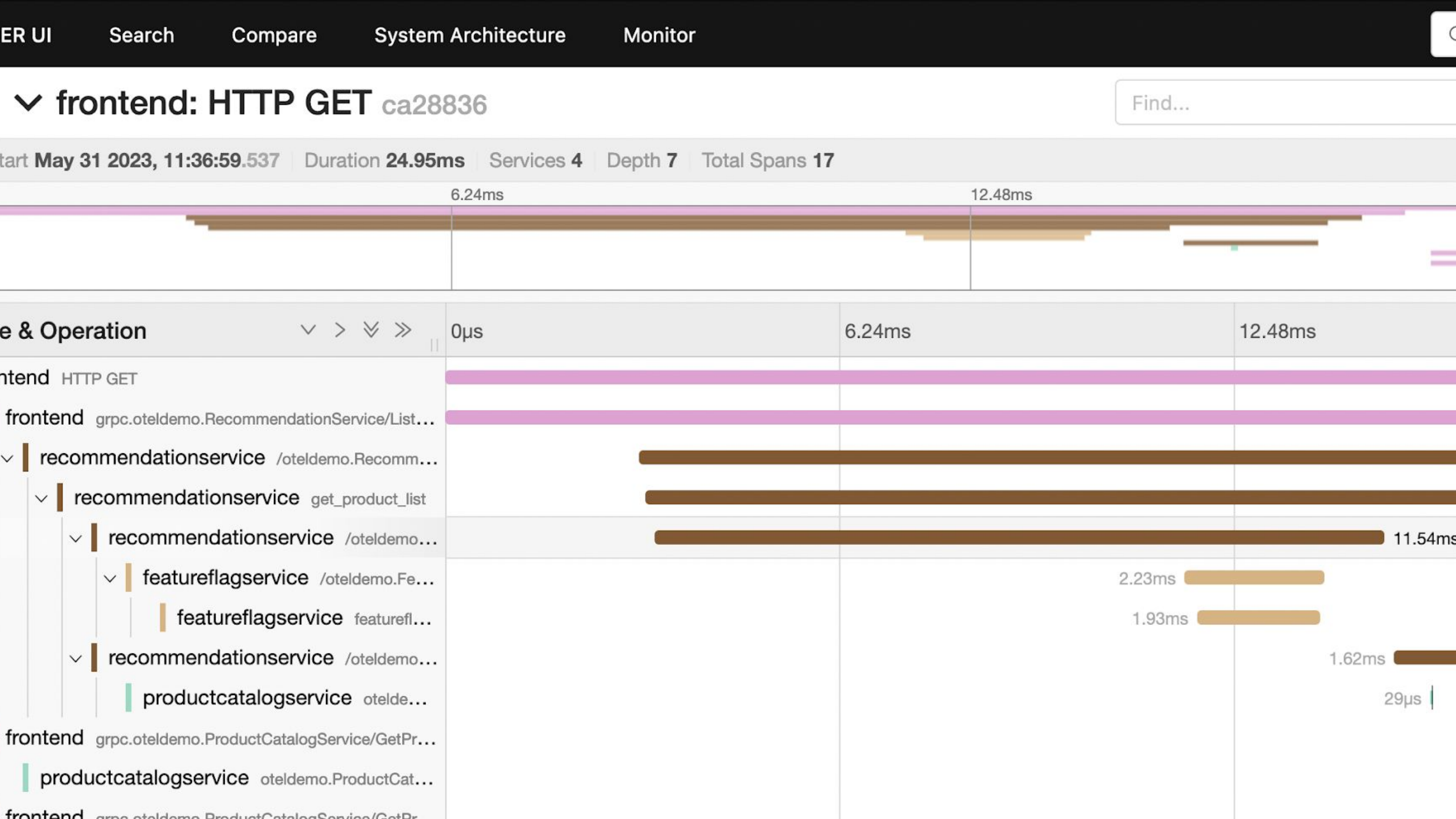
OpenTelemetry demo

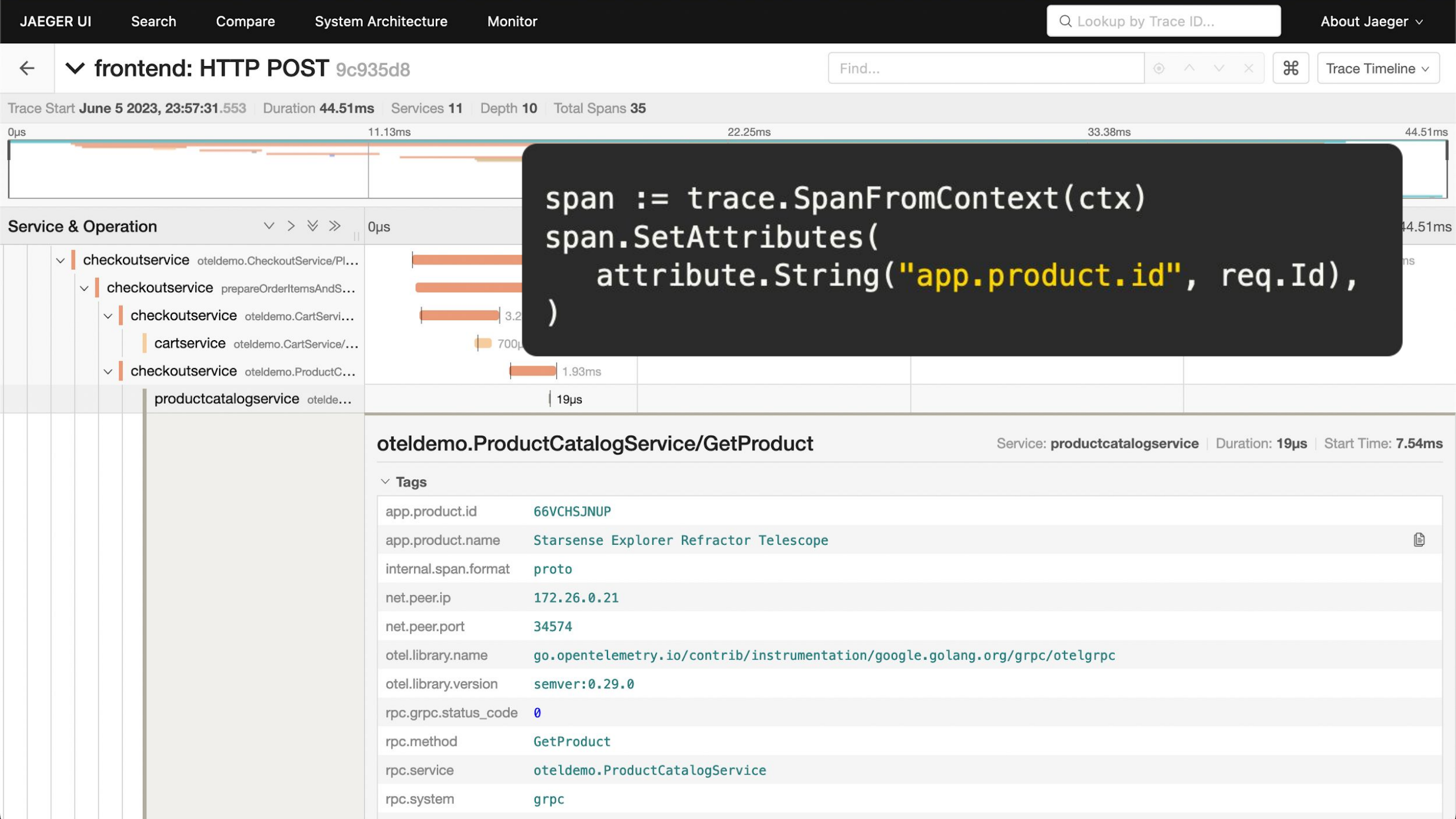
$ USD

The best telescopes to see the world closer

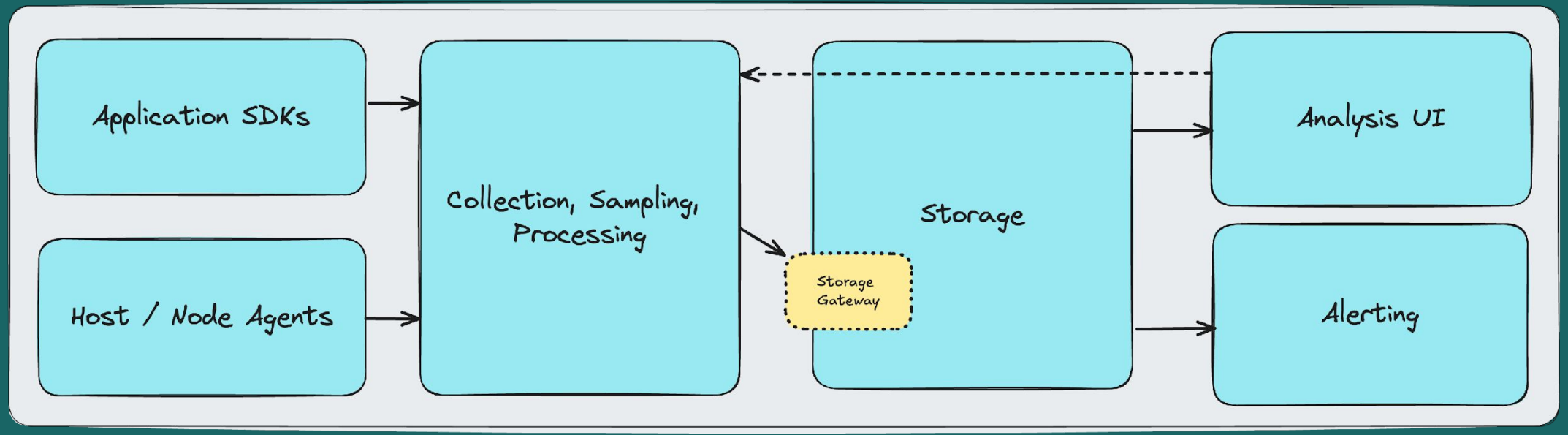
Go Shopping


Demo / Spanmetrics Demo Dashboard
Last 1 hour
5m
service
All
span_name
All
Service Level - Throughput and Latencies
Top 3×3 - Service Latency - quantile95
86.6 ms
- frontend-proxy
56.8 ms
- loadgenerator
47.3 ms
- recommendationse...
44.9 ms
- frontend
41.8 ms
- checkoutservice
36.1 ms
- productcatalogser...
19.2 ms
Top 7 Services Mean Rate over Range
frontend
1.15 req/s
cartservice
0.48 req/s
productcatalogservice
0.35 req/s
loadgenerator
0.30 req/s
checkoutservice
0.25 req/s
featureflagservice
0.12 req/s
recommendationservice
0.09 req/s
Top 7 Services Mean ERROR Rate over Range


ER UI
Search
Compare
System Architecture
Monitor
frontend: HTTP GET ca28836
Find...
tart May 31 2023, 11:36:59.537
Duration 24.95ms
Services 4
Depth 7
Total Spans 17
6.24ms
12.48ms
e & Operation
0µs
6.24ms
12.48ms
ntend HTTP GET
frontend grpc.oteldemo.RecommendationService/List...
recommendationservice /oteldemo.Recomm...
recommendationservice get_product_list
recommendationservice /oteldemo...
11.54ms
featureflagservice /oteldemo.Fe...
2.23ms
featureflagservice featurefl...
1.93ms
recommendationservice /oteldemo...
1.62ms
productcatalogservice otelde...
29µs
frontend grpc.oteldemo.ProductCatalogService/GetPr...
productcatalogservice oteldemo.ProductCat...


JAEGER UI
Search
Compare
System Architecture
Monitor
Lookup by Trace ID...
About Jaeger
frontend: HTTP POST 9c935d8
Find...
Trace Timeline
Trace Start June 5 2023, 23:57:31.553 | Duration 44.51ms | Services 11 | Depth 10 | Total Spans 35
0µs
11.13ms
22.25ms
33.38ms
44.51ms
span := trace.SpanFromContext(ctx)
span.SetAttributes(
attribute.String("app.product.id", req.Id),
)
Service & Operation
checkoutservice oteldemo.CheckoutService/Pl...
checkoutservice prepareOrderItemsAndS...
checkoutservice oteldemo.CartServi...
cartservice oteldemo.CartService/...
checkoutservice oteldemo.ProductC...
productcatalogservice oteldemo...
1.93ms
19µs
oteldemo.ProductCatalogService/GetProduct
Service: productcatalogservice | Duration: 19µs | Start Time: 7.54ms
Tags
app.product.id 66VCHSJNUP
app.product.name Starsense Explorer Refractor Telescope
internal.span.format proto
net.peer.ip 172.26.0.21
net.peer.port 34574
otel.library.name go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc
otel.library.version semver:0.29.0
rpc.grpc.status_code 0
rpc.method GetProduct
rpc.service oteldemo.ProductCatalogService
rpc.system grpc

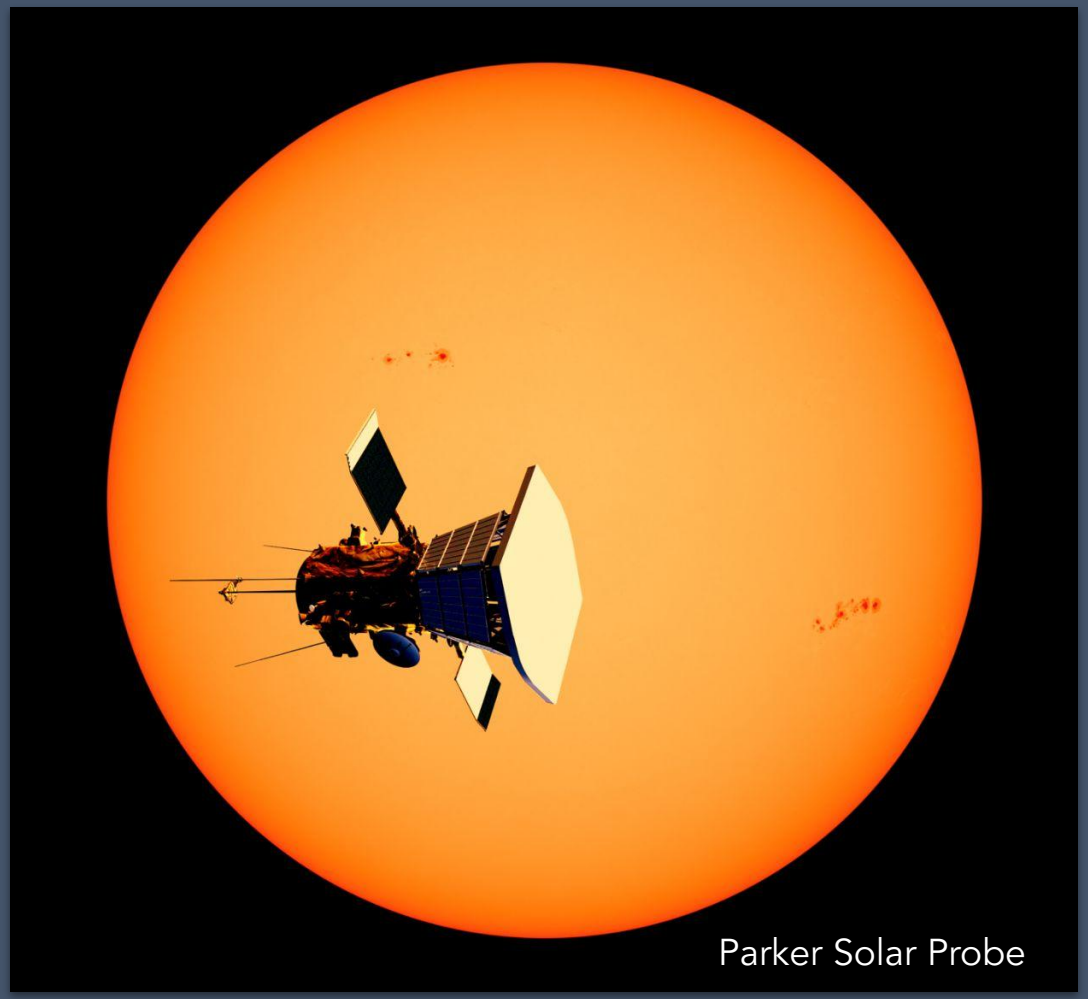
A complete observability solution



eBPF

Provides external observability into any syscalls made by a target process.

Allows network request mapping.



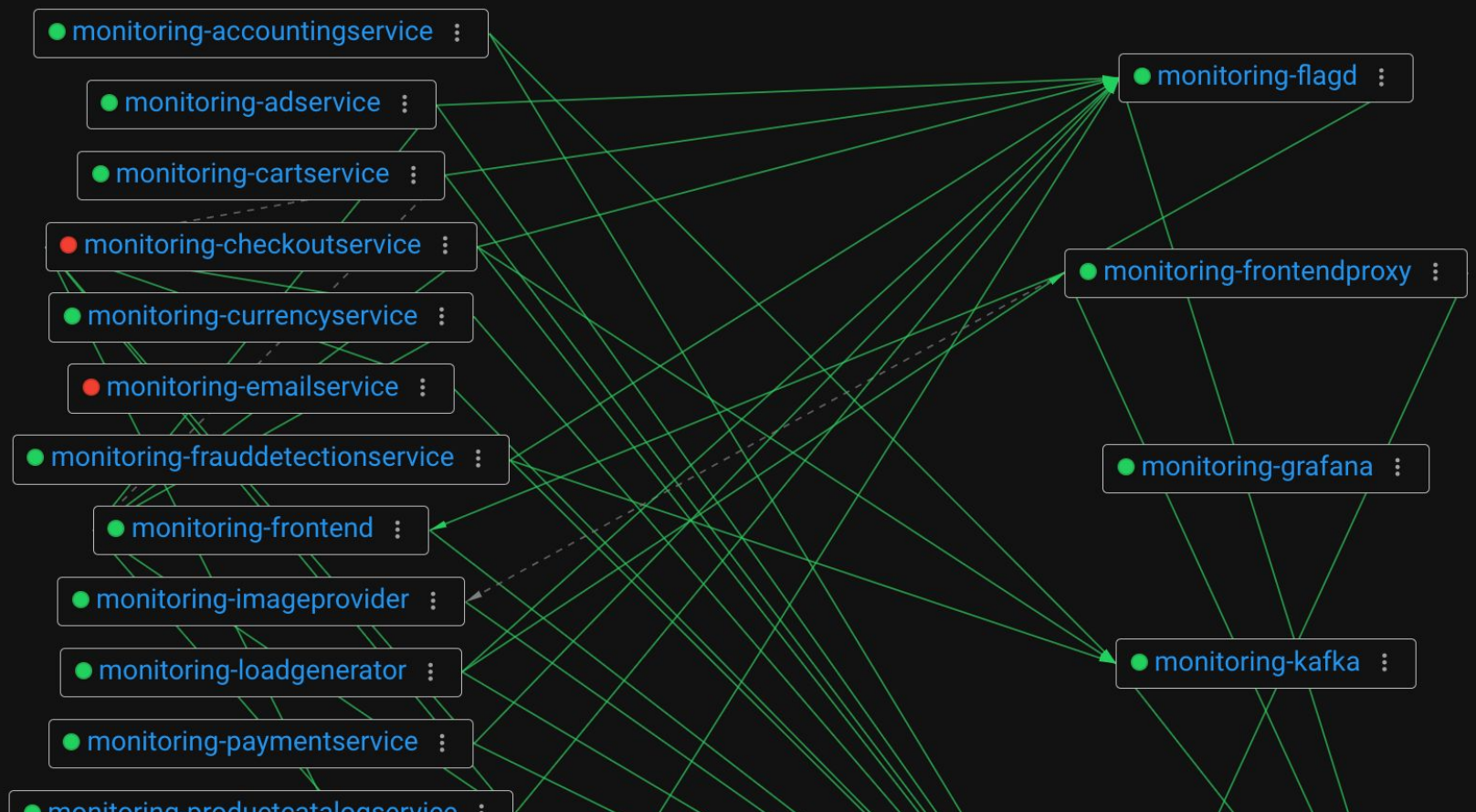
Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)

coroot :~# | default | search for apps and nodes | last hour

Overview

HEALTH | SERVICE MAP | TRACES | NODES | DEPLOYMENTS | COSTS

search | namespaces: monitoring | application | control-plane | monitoring | +



Closing Thoughts

Why Open Source Observability?

Vendor-neutral instrumentation

Portable telemetry formats

Interoperable toolchains

Learning & growing together

Q&A

Thank You Red Hat Community!

@joshleecreates.bsky.social

linkedin.com/in/joshuamlee

altinity.com/slack

osacon.io

Connect with me