

# Guide to Observability with OpenTelemetry



Evelien Schellekens  
Principal Solutions Architect



# Monolithic applications



# The Big Bang



# Microservices era



# Houston, we have problem

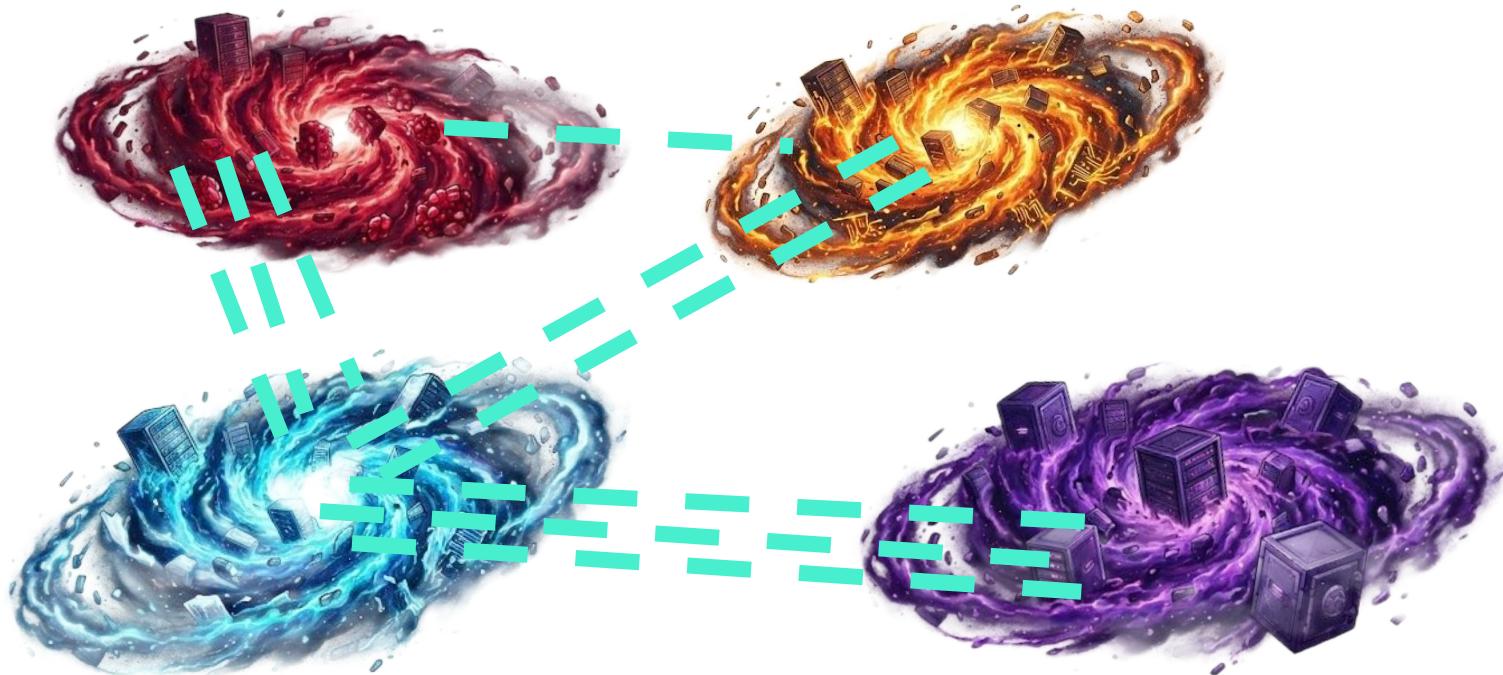
But we don't know where



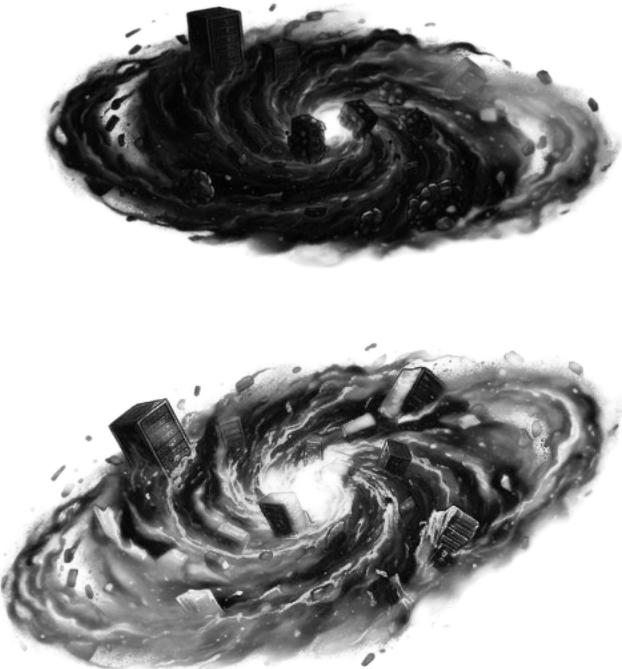
# Zooming out



# Zooming out



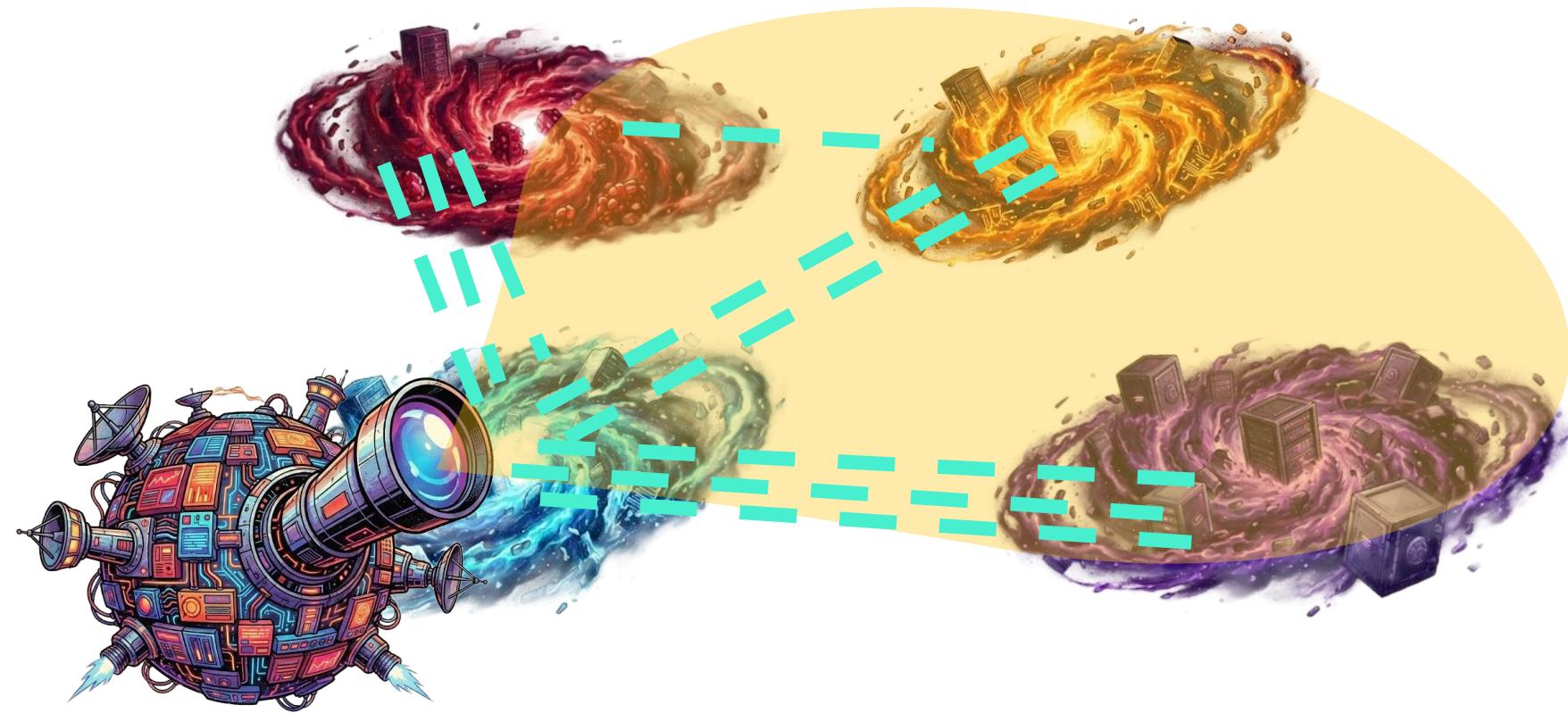
# Lost in space



The worst kind of alerts are

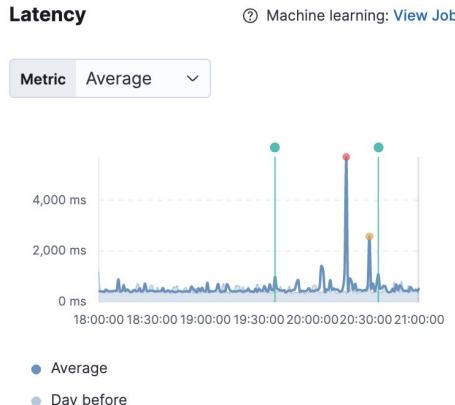
your customers

# Mission control

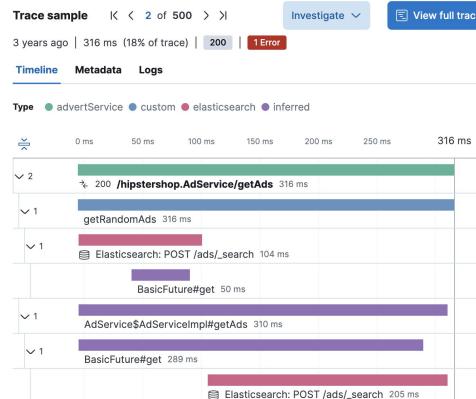


# Observability signals

metrics



traces



logs

Documents (5,605)	Field statistics
@timestamp	resource
Jun 1, 2022 @ 28:36:43.662	k8s_advertisement_advertise
	*** server shut down
Jun 1, 2022 @ 28:36:43.644	advertiservice
	*** shutting down gRPC ads server since JVM is shutting down
Jun 1, 2022 @ 28:36:43.454	advertiservice
	info ✓ Returning 1 ads
Jun 1, 2022 @ 28:36:43.453	advertiservice
	info ✓ Items 114024 now in cache
Jun 1, 2022 @ 28:36:43.452	advertiservice
	info ✓ Adding 1 items to cache
Jun 1, 2022 @ 28:36:43.179	advertiservice
	info ✓ Returning 2 ads
Jun 1, 2022 @ 28:36:43.179	advertiservice
	info ✓ Items 113521 now in cache
Jun 1, 2022 @ 28:36:43.178	advertiservice
	info ✓ Adding 2 items to cache
Jun 1, 2022 @ 28:36:43.179	advertiservice
	info ✓ Cache miss for category: Cookware
Jun 1, 2022 @ 28:36:43.443	advertiservice
	info ✓ Cache miss for category: Photography
Jun 1, 2022 @ 28:36:43.843	advertiservice
	info ✓ received ad request (context_words=[Cookware])
Jun 1, 2022 @ 28:36:42.773	advertiservice
	info ✓ Cache miss for category: Photography
Jun 1, 2022 @ 28:36:42.745	advertiservice
	info ✓ received ad request (context_words=[Photograph])
Jun 1, 2022 @ 28:36:48.598	advertiservice
	info ✓ Returning 1 ads
Jun 1, 2022 @ 28:36:49.597	advertiservice
	info ✓ Items 112521 now in cache
Jun 1, 2022 @ 28:36:49.596	advertiservice
	info ✓ Adding 1 items to cache
Jun 1, 2022 @ 28:36:49.388	advertiservice
	info ✓ Cache miss for category: Cookware
Jun 1, 2022 @ 28:36:49.388	advertiservice
	info ✓ received ad request (context_words=[Cookware])
Jun 1, 2022 @ 28:36:39.721	advertiservice
	info ✓ Cache miss for category: Photography
Jun 1, 2022 @ 28:36:39.721	advertiservice
	info ✓ Items 112020 now in cache
Jun 1, 2022 @ 28:36:39.720	advertiservice
	info ✓ Adding 2 items to cache
Jun 1, 2022 @ 28:36:39.312	advertiservice
	info ✓ Cache miss for category: Gardening



# Hello world

OpenTelemetry is an **open-source**  
**Observability** framework and toolkit  
designed to create and manage **telemetry**  
data such as traces, metrics, and logs.

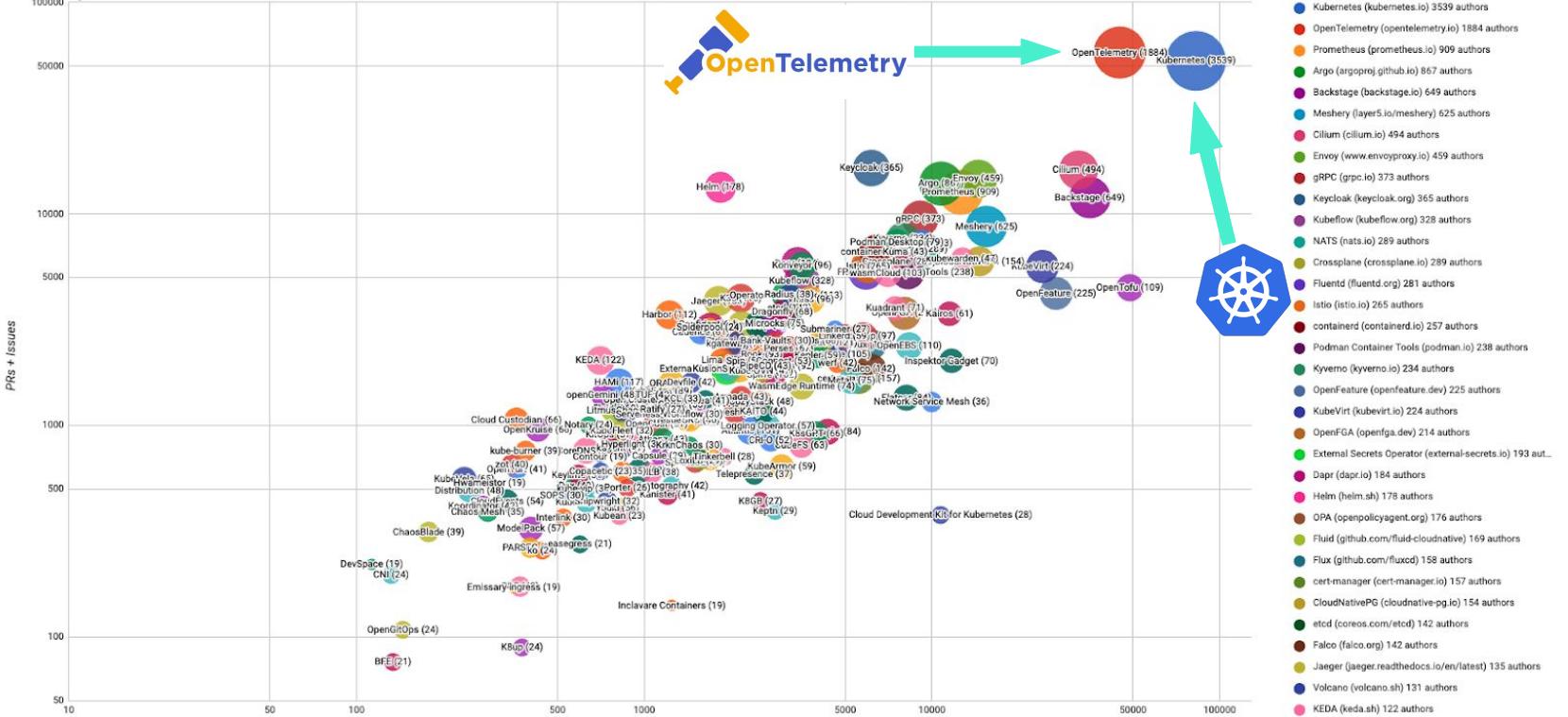
**Vendor-agnostic**, supported by a variety of  
observability backends.

OpenTelemetry was created in **2019** by  
merging OpenTracing and OpenCensus into a  
single standard.



# OpenTelemetry taking off

CNCF Projects 7/1/2024 - 7/1/2025



## source



# OTel me more

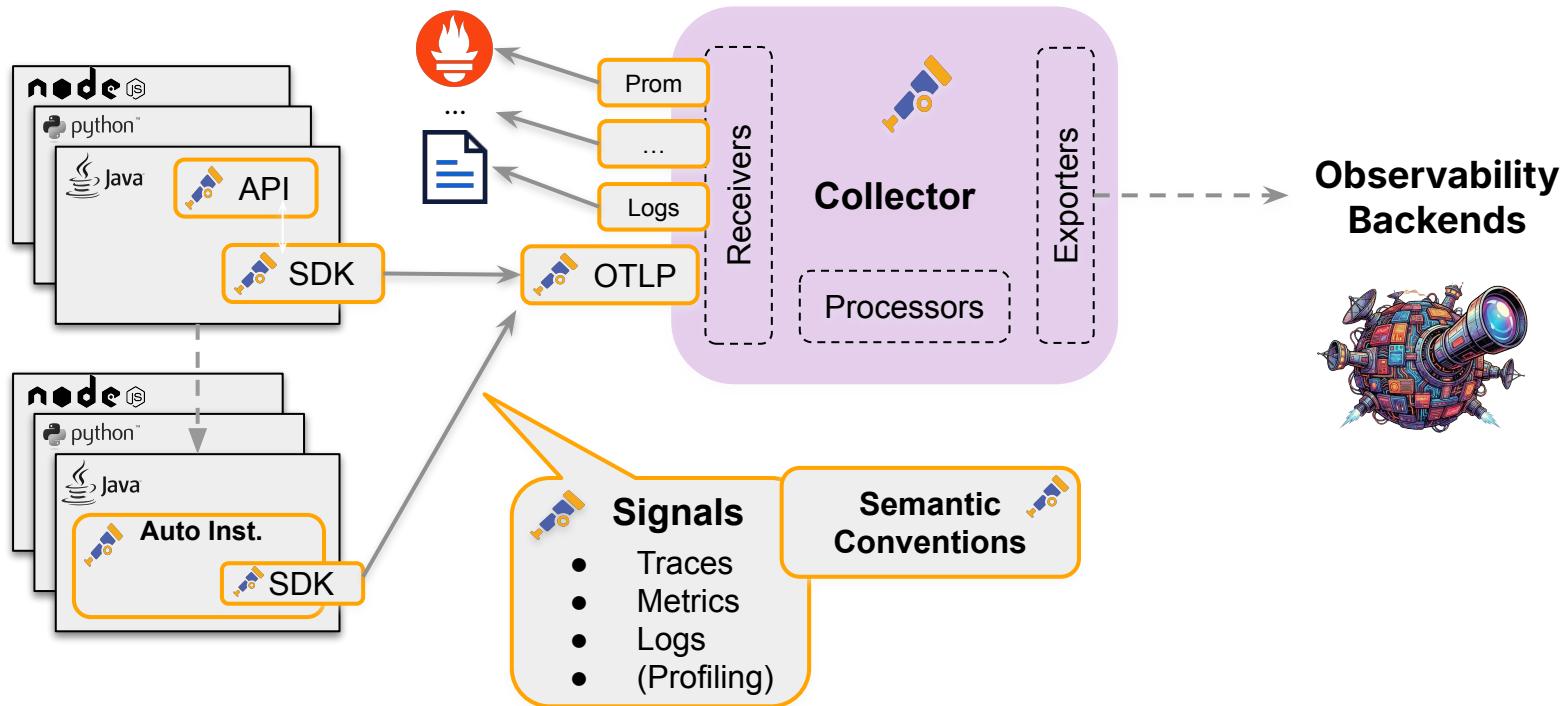
## standards

- API Specification
- OTel Protocol (OTLP) Signals
- Semantic Conventions
- Agent Management Protocol (OpAMP)

## tooling

- SDKs (Java, .NET, Python...)
- Auto-instrumentation
- OTel Collector
- K8s Operator
- RUM & Mobile

# OpenTelemetry Ecosystem



# Universal language

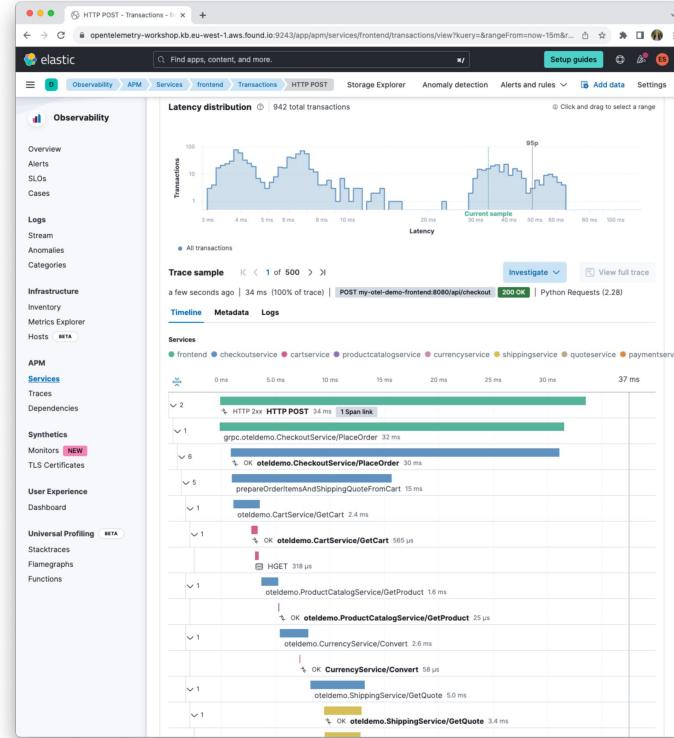
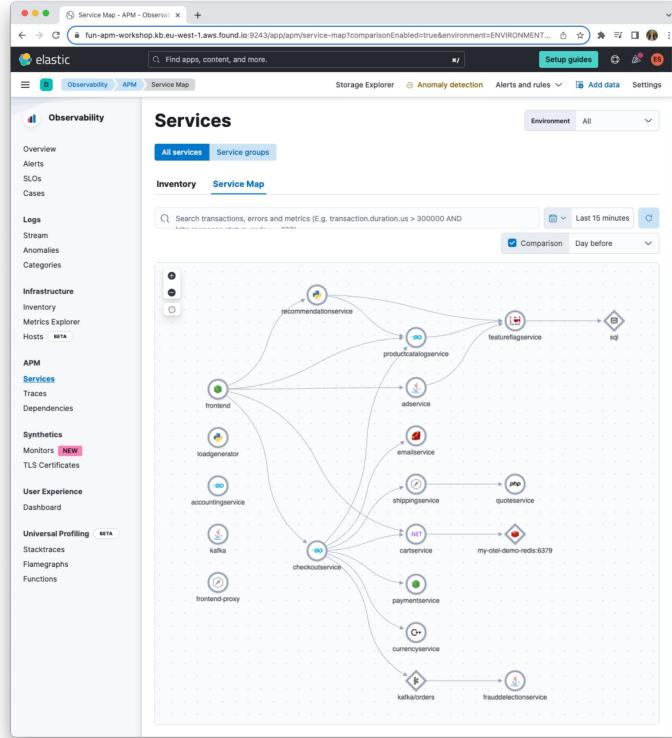
Searching ***without*** SemConv

```
host:my-host  
OR host_name:my-host  
OR hostname:my-host  
OR context.host.name:my-host  
OR system.name:my-host
```

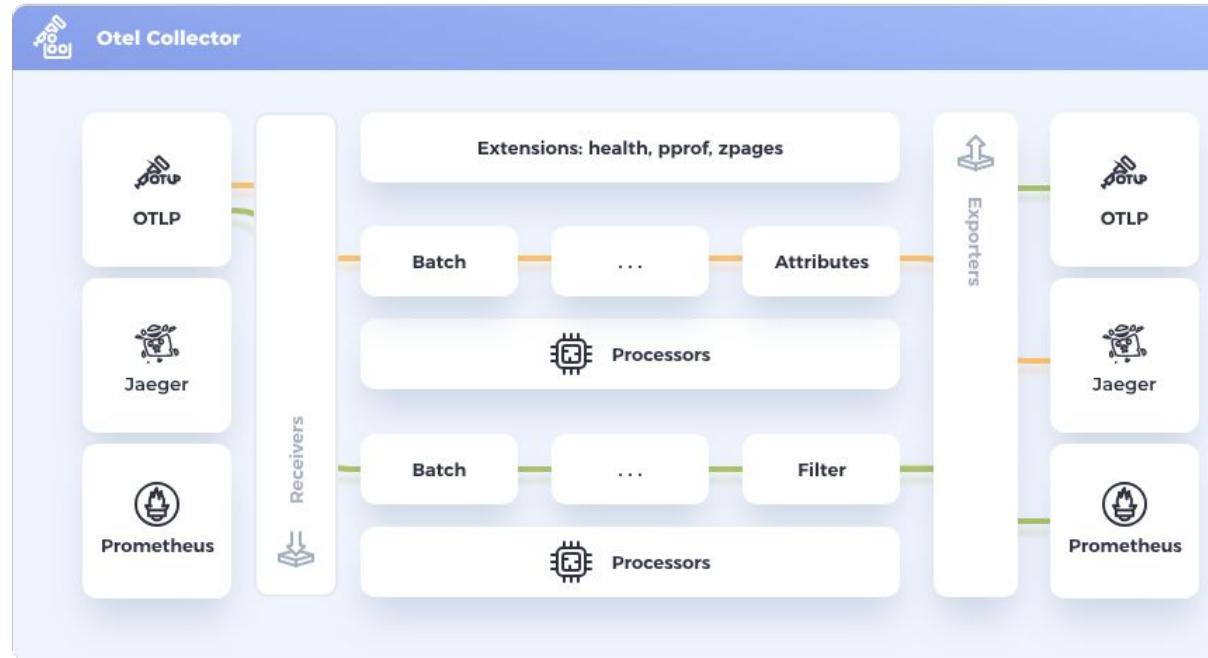
Searching ***with*** SemConv

```
host.name:my-host
```

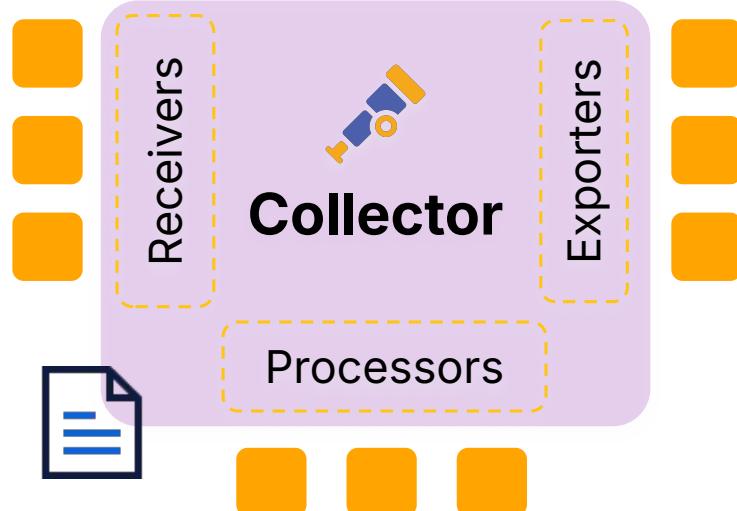
# Distributed Tracing



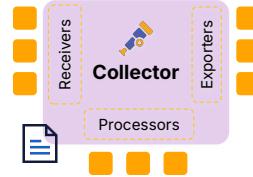
# OpenTelemetry Collector



# Distributions



# Distributions



## Collector Core Distro

- Only core components
- Low practical use / value



## Collector Contrib Distro

- All upstream components
- Covers most of the use cases



## Collector K8s Distro

- Selected components
- For typical K8s use cases



## Elastic Distro of OTel Collector

- Selected components (+ additional components)
- Pre-configured for better UX
- Technical support

# OpenTelemetry Operator



OTel operator manages:

OpenTelemetry Collectors

Auto-Instrumentation

# OpenTelemetry Operator

## OpenTelemetry Collectors

```
apiVersion: opentelemetry.io/v1beta1
kind: OpenTelemetryCollector
metadata:
  name: simplest
spec:
  config:
    ...

```

## *Collector run modes:*

```
apiVersion: opentelemetry.io/v1alpha1
kind: OpenTelemetryCollector
metadata:
  name: example
spec:
  mode: deployment (default) 1
  | daemonset 2
  | sidecar 3
  | statefulset 4
```

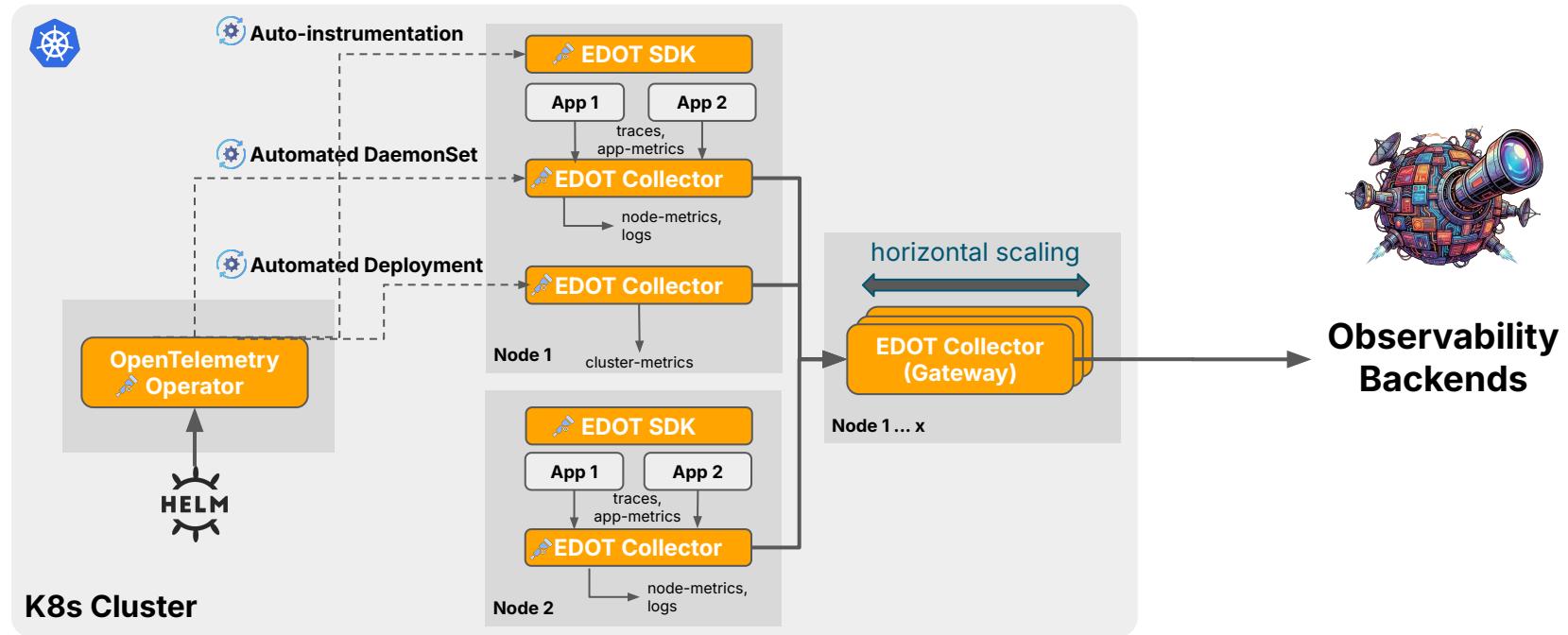
# OpenTelemetry Operator

## Auto-Instrumentation

```
apiVersion: opentelemetry.io/v1alpha1
kind: Instrumentation
metadata:
  name: demo-instrumentation
spec:
  exporter:
    endpoint: http://demo-collector:4318
  ...
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-app
  labels:
    app: python-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: python-app
  template:
    metadata:
      labels:
        app: python-app
    annotations:
      instrumentation.opentelemetry.io/inject-python:
        "demo-instrumentation"
  spec:
    containers:
      - name: python-app
        image: my-python-app:latest
        ports:
          - containerPort: 5000
```

# opentelemetry-kube-stack

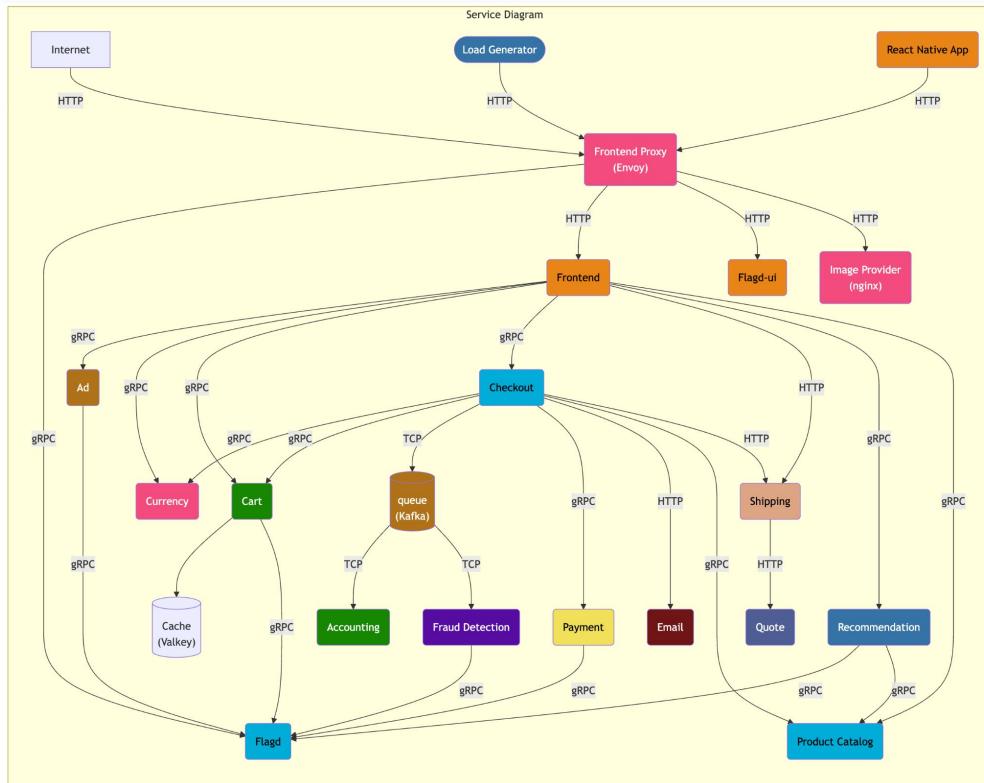


# OpenTelemetry demo application

The image displays three screenshots of a web application interface, illustrating the use of OpenTelemetry for monitoring and tracing.

- Home Page:** Shows a hero image of a person looking through a telescope. Text on the left reads: "The best telescopes to see the world closer". A "Go Shopping" button is present.
- Product Page:** Shows a product detail page for the "National Park Foundation Exploroscope". It includes a large image of the telescope, the product name, price (\$101.96), a quantity selector (set to 1), and an "Add To Cart" button. A detailed description below states: "The National Park Foundation's (NPF) Exploroscope 60AZ is a manual alt-azimuth, refractor telescope perfect for celestial viewing on the go. The NPF Exploroscope 60 can view the planets, moon, star clusters and brighter deep sky objects like the Orion Nebula and Andromeda Galaxy."
- Hot Products:** Shows a grid of telescope products. The first item is the "National Park Foundation Exploroscope" at \$101.96. The second item is the "Starsense Explorer Refractor Telescope" at \$349.95. The third item is the "Eclipsmart Travel Refractor Telescope" at \$129.95. Below the grid are images of telescope accessories: a lens cleaner, a penlight, and a cloth.

# OpenTelemetry demo application



# OpenTelemetry demo application

## Language Feature Reference

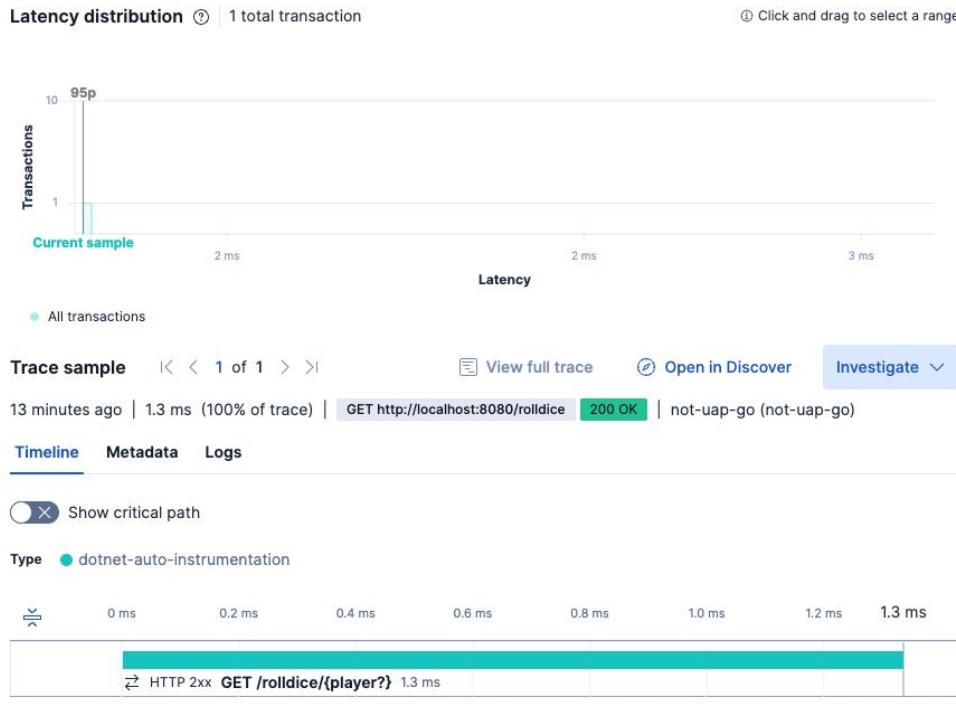
Want to understand how a particular language's instrumentation works? Start here.

Language	Automatic Instrumentation	Instrumentation Libraries	Manual Instrumentation
.NET	<a href="#">Accounting Service</a>	<a href="#">Cart Service</a>	<a href="#">Cart Service</a>
C++			<a href="#">Currency Service</a>
Go		<a href="#">Checkout Service</a> , <a href="#">Product Catalog Service</a>	<a href="#">Checkout Service</a> , <a href="#">Product Catalog Service</a>
Java	<a href="#">Ad Service</a>		<a href="#">Ad Service</a>
JavaScript			<a href="#">Payment Service</a>
TypeScript		<a href="#">Frontend</a> , <a href="#">React Native App</a>	<a href="#">Frontend</a>
Kotlin		<a href="#">Fraud Detection Service</a>	
PHP		<a href="#">Quote Service</a>	<a href="#">Quote Service</a>
Python	<a href="#">Recommendation Service</a>		<a href="#">Recommendation Service</a>
Ruby		<a href="#">Email Service</a>	<a href="#">Email Service</a>
Rust		<a href="#">Shipping Service</a>	<a href="#">Shipping Service</a>

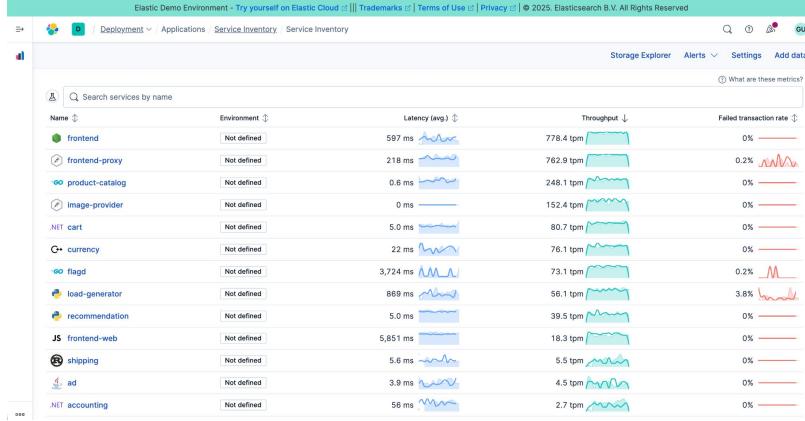
# Demo time!



# Houston, we have telemetry!



# Want to try yourself?



<https://otel.demo.elastic.co/>

The screenshot shows a GitHub repository page for "OpenTelemetry Demo with Elastic Observability". The page includes a README file, contributing guidelines, a license, and security information. The main content area features a title with icons for OpenTelemetry and Elastic Observability, followed by a description of how to set up the demo using Docker Compose or Kubernetes. It lists several changes made to the demo, such as replacing Java and .NET agents with their Elastic distributions, and replacing Node.js and Python agents with their Elastic distributions. Additionally, the OpenTelemetry Contrib collector has been changed to the Elastic OpenTelemetry Collector distribution.

The following guide describes how to setup the OpenTelemetry demo with Elastic Observability using [Docker compose](#) or [Kubernetes](#). This fork introduces several changes to the agents used in the demo:

- The Java agent within the [Ad](#), the [Fraud Detection](#) and the [Kafka](#) services have been replaced with the Elastic distribution of the OpenTelemetry Java Agent. You can find more information about the Elastic distribution in [this blog post](#).
- The .NET agent within the [Cart service](#) has been replaced with the Elastic distribution of the OpenTelemetry .NET Agent. You can find more information about the Elastic distribution in [this blog post](#).
- The Elastic distribution of the OpenTelemetry Node.js Agent has replaced the OpenTelemetry Node.js agent in the [Payment service](#). Additional details about the Elastic distribution are available in [this blog post](#).
- The Elastic distribution for OpenTelemetry Python has replaced the OpenTelemetry Python agent in the [Recommendation service](#). Additional details about the Elastic distribution are available in [this blog post](#).

Additionally, the OpenTelemetry Contrib collector has also been changed to the [Elastic OpenTelemetry Collector distribution](#). This ensures a more integrated and optimized experience with Elastic Observability.

<https://github.com/elastic/opentelemetry-demo>

# To infinity and beyond

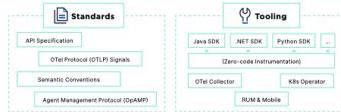
- Baggage
  - Metadata that travels with the request
- OTTL
  - OpenTelemetry Transformation Language
  - OTTL Playground: <https://ottl.run/>
- OpAmp
  - Open Agent Management Protocol
  - Remote management of large fleets of data collection Agents





## OpenTelemetry Cheatsheet

### The OpenTelemetry Ecosystem



### Tooling

- SDKs (Java, .NET, Python...): Language libraries for manual instrumentation
- Auto-instrumentation: Automatic tracing and metrics without code changes
- OTel Collector: Vendor-neutral pipeline to receive, process, and export telemetry
- K8s Operator: Automates Collector deployment and management in Kubernetes
- RUM & Mobile: Collects telemetry from browsers and mobile apps

### OpenTelemetry Collector



- Vendor-agnostic service that receives, processes, and exports telemetry data (traces, metrics, logs)
- Architecture built around pipelines, each made of **receivers** (ingest), **processors** (transform/filter/batch/sample), and **exporters** (send to back-ends)
- Supports multiple signal types in same deployment (logs, metrics, traces)
- Extensible: You can add custom receivers, processors, and exporters via contrib or build your own distribution
- Modes: Agent (sidecar or per-host) or gateway/central collector
- Handles batching, retries, filtering, and resource enrichment so application code stays lighter
- Installable via binaries and container images on Linux, Windows, macOS, and Kubernetes

### Collector distributions

- Core: Minimal, stable set of receivers, processors, and exporters. Maintained by the OTel project, focused on usability
- Contrib: Superset of core with many additional components. Faster moving, includes experimental or community-built features
- Vendor distros: Custom builds from vendors (e.g., Elastic Distributions of OTel EDOT, AWS Distro for OTel ADOT) that package selected core/contrib components plus vendor-specific exporters, defaults, and support
- Why it matters: Running the contrib build in production is not recommended. Use a custom build with only the required components or a vendor-supported distribution.

### Collector key commands

```
None
otelcol --config config.yaml      # Run with config
otelcol --config config.yaml --dry-run # Validate config syntax
otelcol components               # List available components
otelcol --version                # Show version
otelcol --help                   # Show help options
```

### Key environment variables

- General
  - OTEL\_RESOURCE\_ATTRIBUTES
  - OTEL\_LOG\_LEVEL
- OTLP Exporter
  - OTEL\_EXPORTER\_OTLP
  - OTEL\_EXPORTER
  - OTEL\_EXPORTER
  - OTEL\_EXPORTER
  - OTEL\_EXPORTER
- Sampling
  - OTEL\_TRACES\_SAMPLER
  - OTEL\_TRACES\_SAMPLER

### OpenTelemetry

- Language-specific and general te
  - Provide APIs for options to capture
  - Expose standard service/version
  - Support multiple (depending on language)
  - Pluggable exporter or through a Collector
  - Lightweight: Perfect for Collector for efficiency
  - Available for most Go, Node.js, Python

### SDK distributions

- Official SDKs: Node.js, Python, Java, .NET, and others
- Auto-instrumentation: Python (CLJ) that
- Vendor distros: Enhanced support for specific languages
- Custom Builds: Usages, experiments, and more

### Best practices

- Always set service.name
- Use semantic conventions
- Filter and transform in Collector, not in application
- Correlate logs/metrics/traces with IDs
- Use exporter sending\_queue=batch

### Debugging

- Env: OTEL\_LOG\_LEVEL=debug
- Use logging exporter
- Validate ports: 4317 (gRPC), 4318 (HTTP)

### Quick ports reference

- |                |                   |
|----------------|-------------------|
| SIGNAL         | Collector OTLP    |
| Collector gRPC | Collector HTTP    |
| Prometheus     | Prometheus scrape |

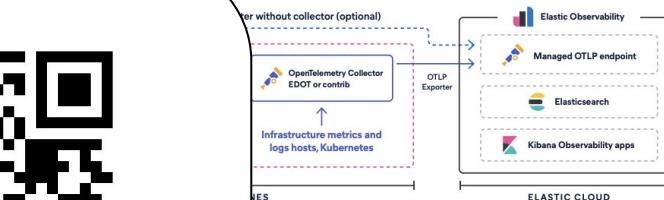
ela.st/  
otel-cheat-sheet

## OpenTelemetry + Elastic

### Exploring Elastic Observability with OTel

#### Option 1: Explore online

- Visit [otel.demo.elastic.co](https://otel.demo.elastic.co)
- Preconfigured Elastic Stack with telemetry from the OTel demo microservices app instrumented with EDOT
- Explore Kibana dashboards, APM UI, service maps, and Discover for data analysis



#### Step 3 | Instrument applications with EDOT SDKs

- Set up the corresponding language SDK <https://ela.st/edot-sdk>.
- Set resource attributes:

```
None
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317
.ENVIRONMENT_NAME=prod
```

- Configure your SDK to send data to the EDOT Collector OTLP endpoint or directly to Elastic depending on your architecture.

```
None
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317
export OTEL_EXPORTER_OTLP_HEADERS="Authorization:ApiKey <YourApiKey>"
```

#### Step 4 | Explore in Kibana

- APM UI: View traces, latency, and error rates per service.
- Service map: Visualize dependencies across services.
- Dashboards: Monitor key metrics out of the box.
- Discover: Query raw telemetry data (logs, traces, and metrics) with flexible search.

## RESOURCES FOR DEVELOPERS | BY DEVELOPERS LIKE YOU!



20 January 2026

### A train ride away from a million events per second with EDOT Cloud Forwarder

by Michalis Katsoulis, Andreas Gkizas, Miguel Luna

EDOT Cloud Forwarder for AWS from Elastic Observability is now Generally Available. Deploying EDOT Cloud Forwarder and reliably handling one million events per second with zero intervention, zero data loss, and zero idle cost.

OpenTelemetry AWS

[elastic.co/  
observability-labs](https://elastic.co/observability-labs)



# Conclusion

✨ Observability ✨

✨ OpenTelemetry ✨



✨ OTel Operator ✨

✨ OTel demo application ✨

# Questions?

Thank you!



Let's connect!



Evelien Schellekens

Principal Solutions Architect at Elastic

