Guide to Observability with OpenTelemetry

Evelien Schellekens

Principal Solutions Architect

elastic | The Search AI Company

Monolithic applications

The Big Bang

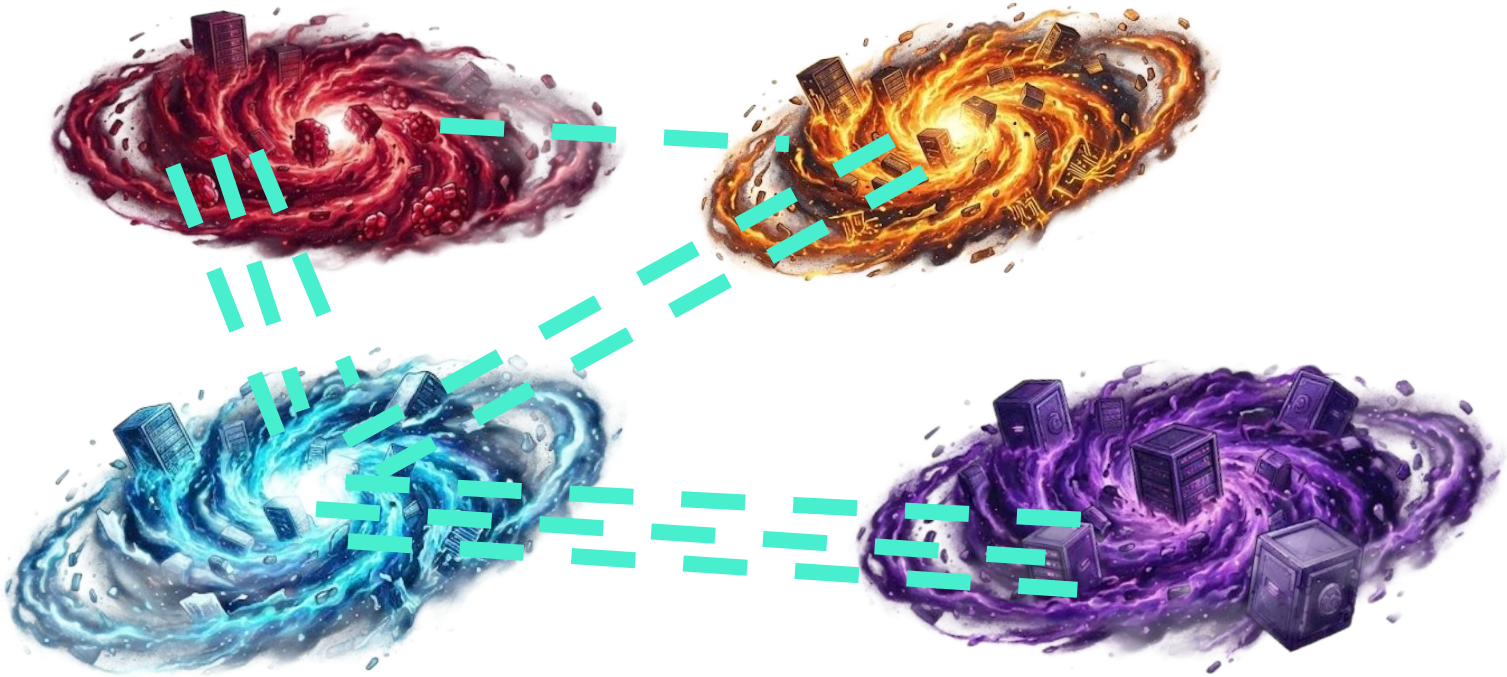
Microservices era

Houston, we have problem

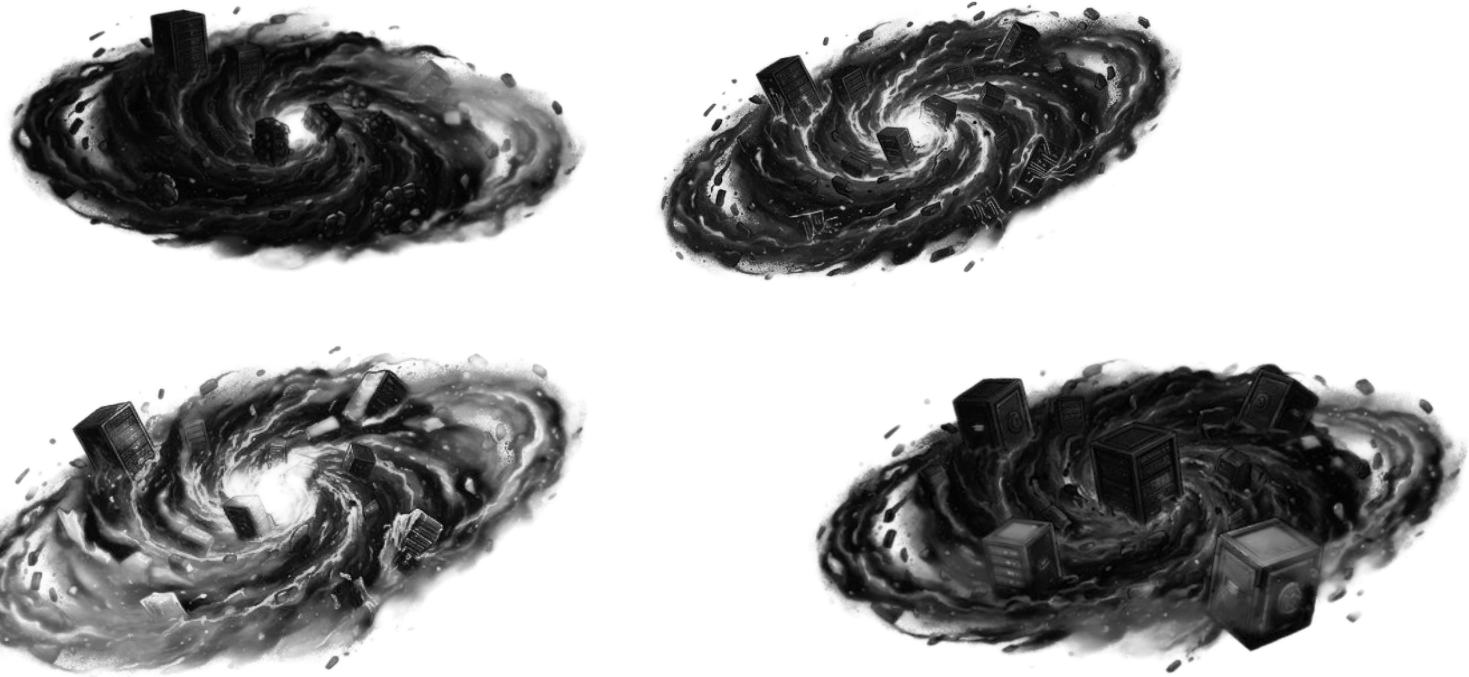
But we don’t know where

Zooming out

Zooming out



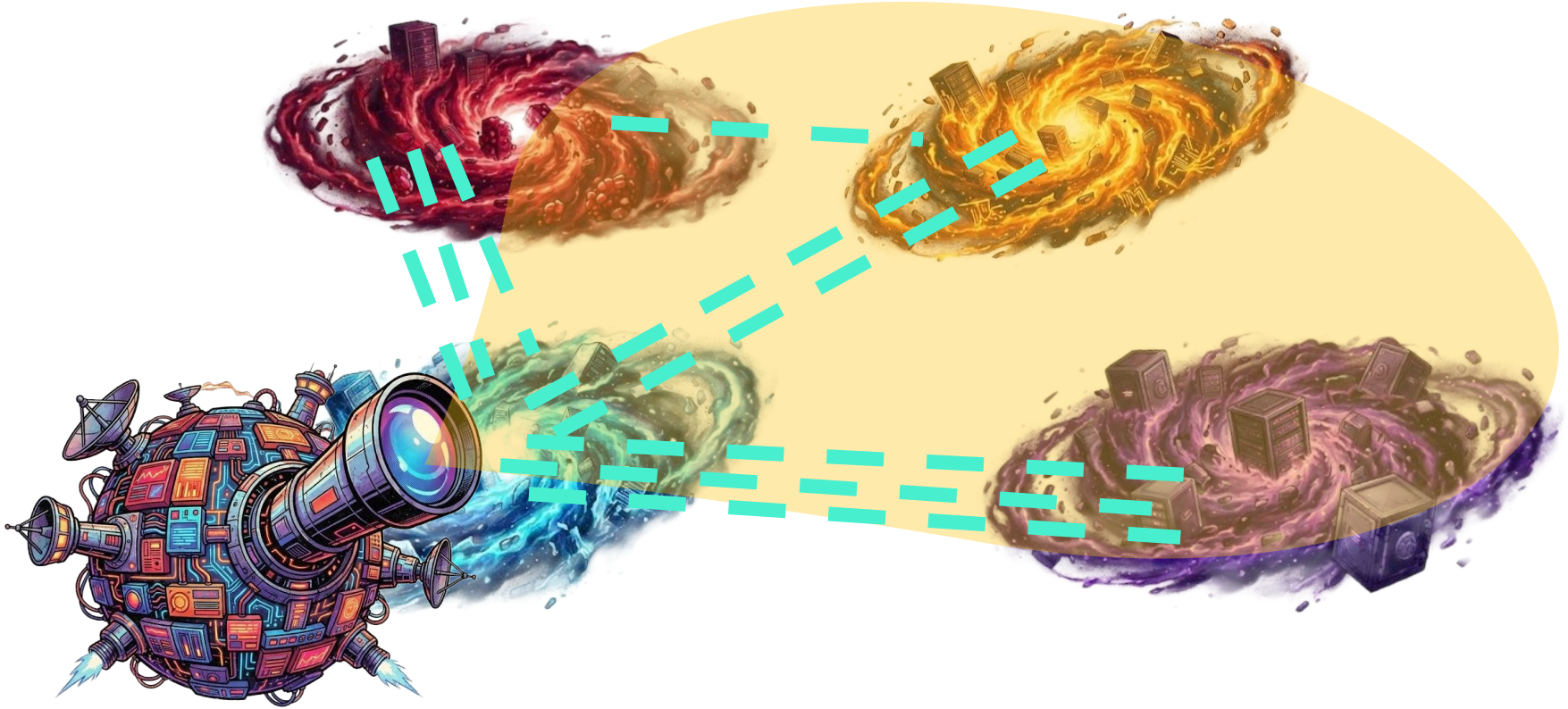
Lost in space



The worst kind of alerts are

your customers

Mission control

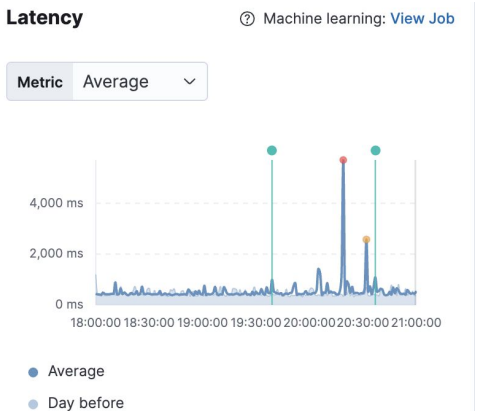


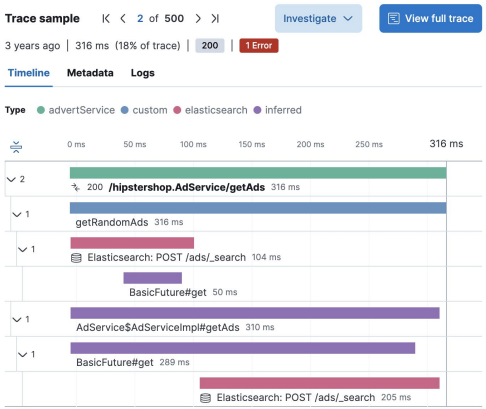
Observability signals

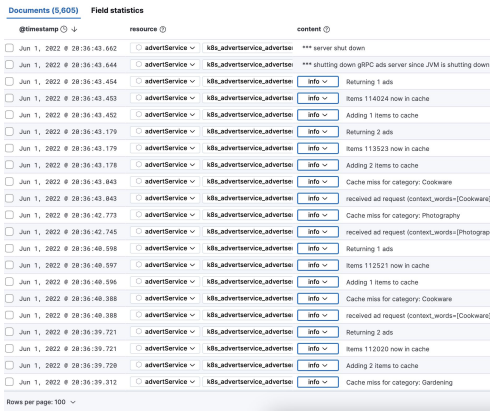
metrics

traces

logs







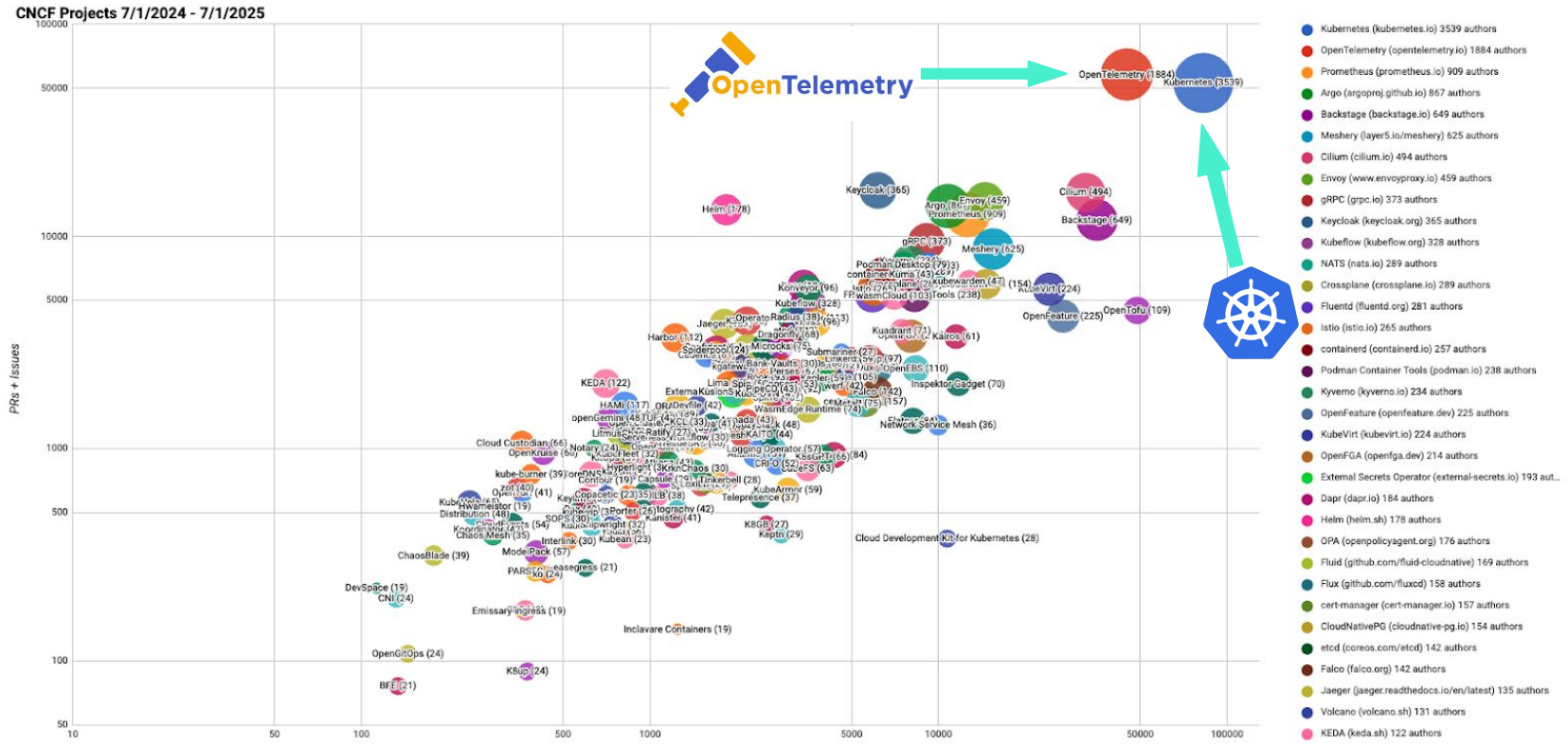
Hello world

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



[source]

elastic

OTel me more

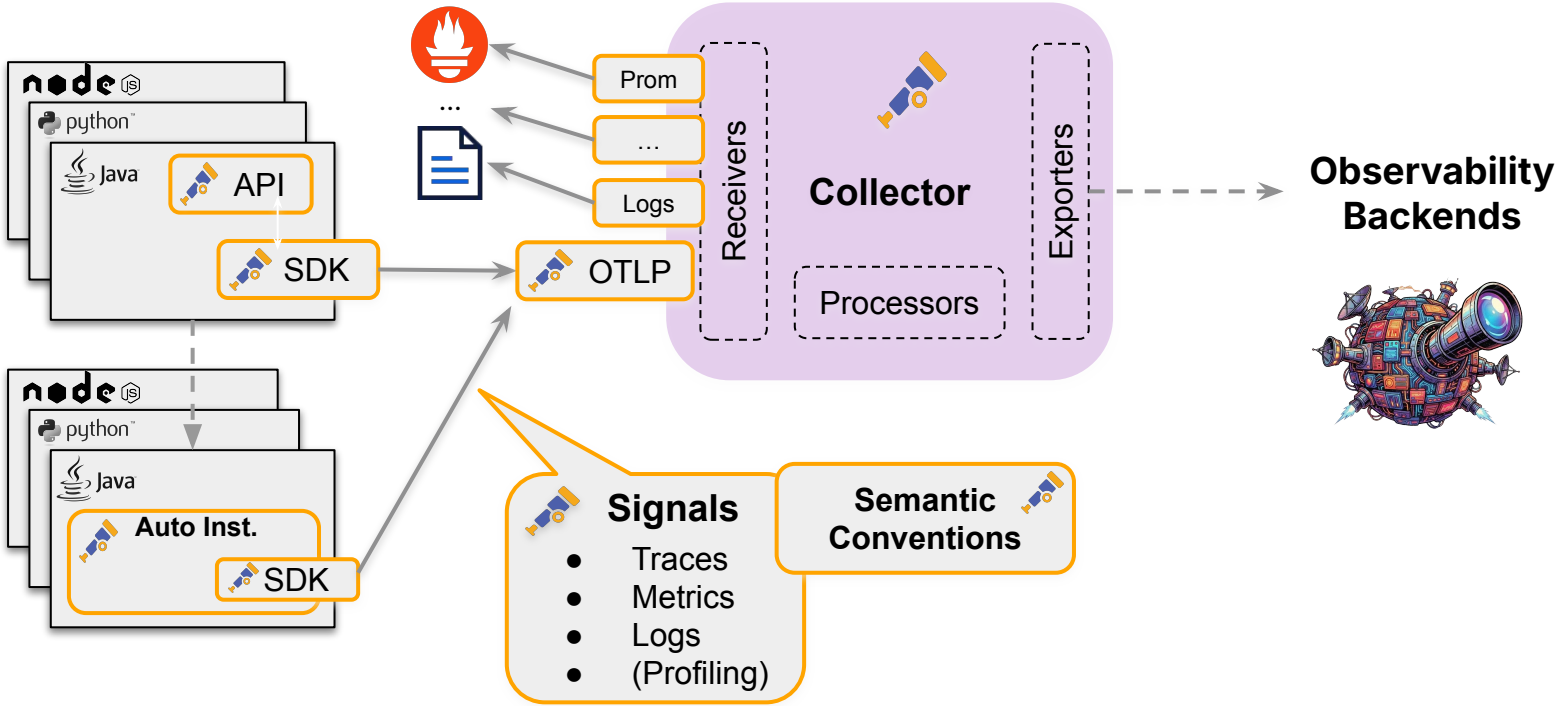
standards

- 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

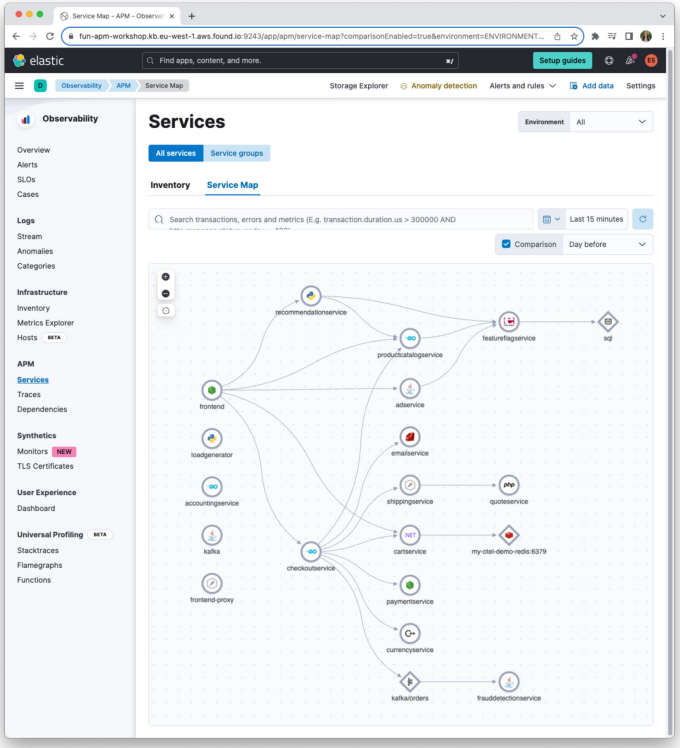
Searching ***without*** SemConv

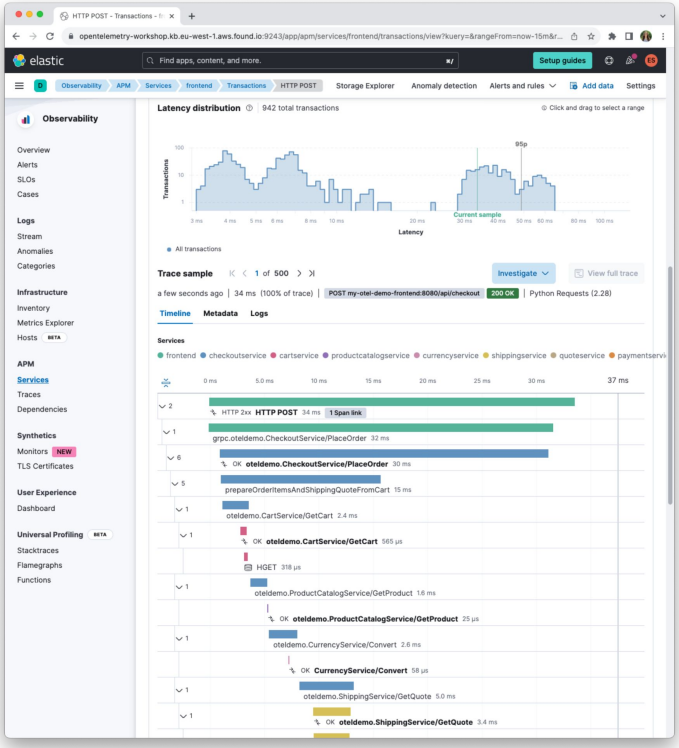
```
host:my-host
OR host_name:my-host
OR hostname:my-host
OR context.host.name:my-host
OR system.name:my-host
```

Searching ***with*** SemConv

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

Distributed Tracing



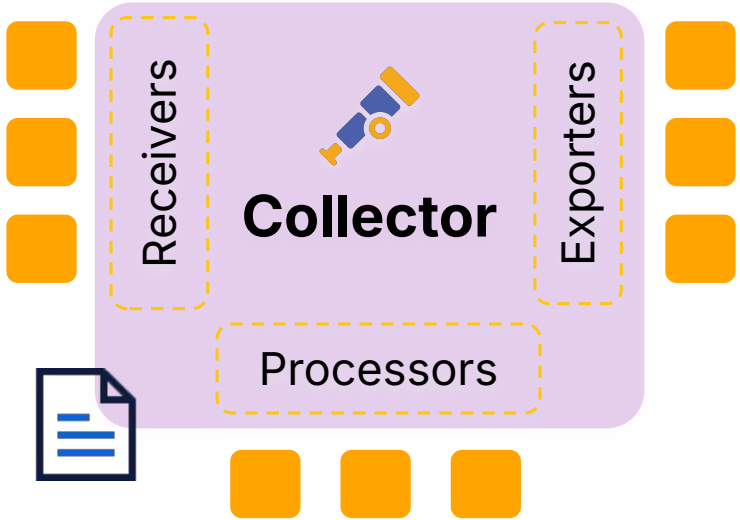


elastic

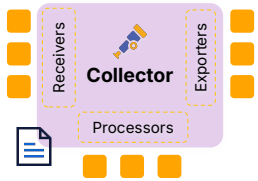
OpenTelemetry Collector

https://opentelemetry.io/docs/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

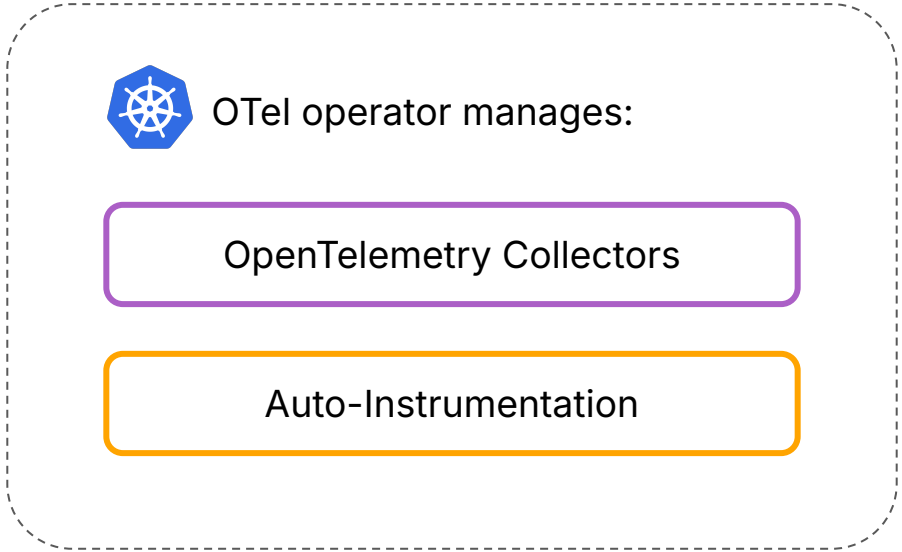
Collector K8s Distro

- 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



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
```

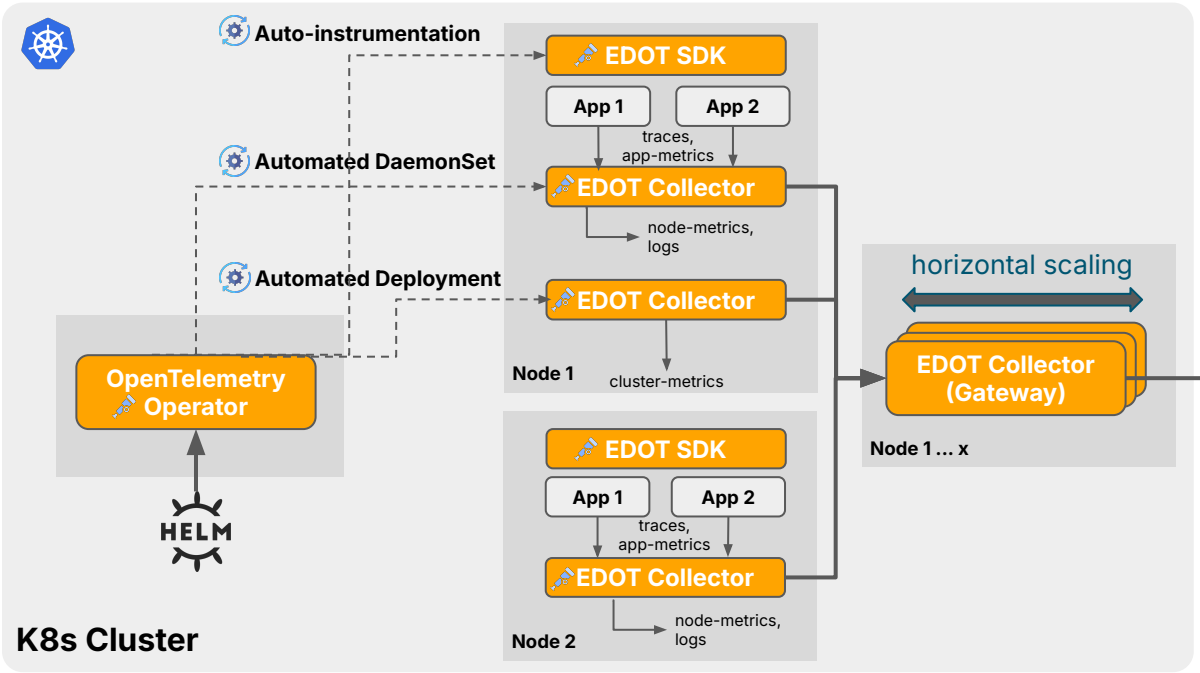
OpenTelemetry Operator

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
```

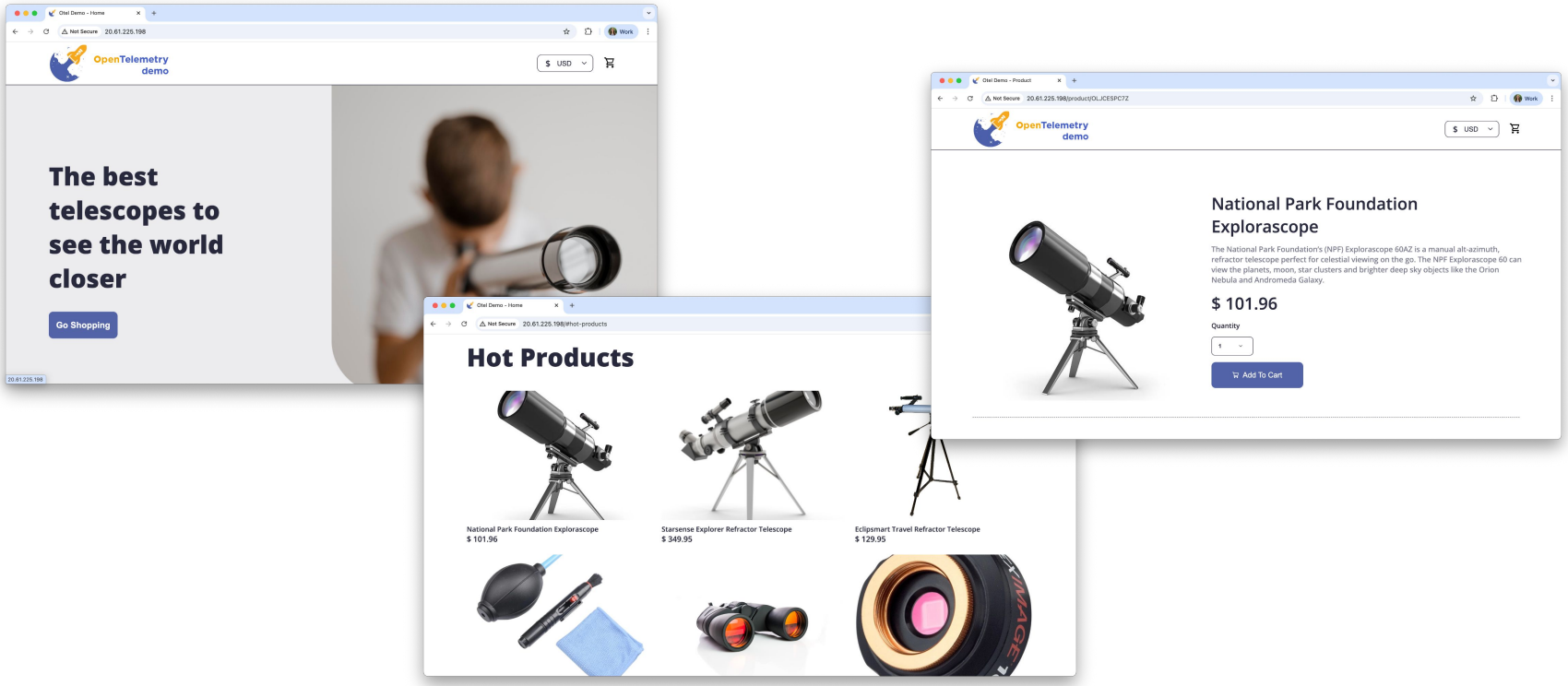
opentelemetry-kube-stack



Observability Backends

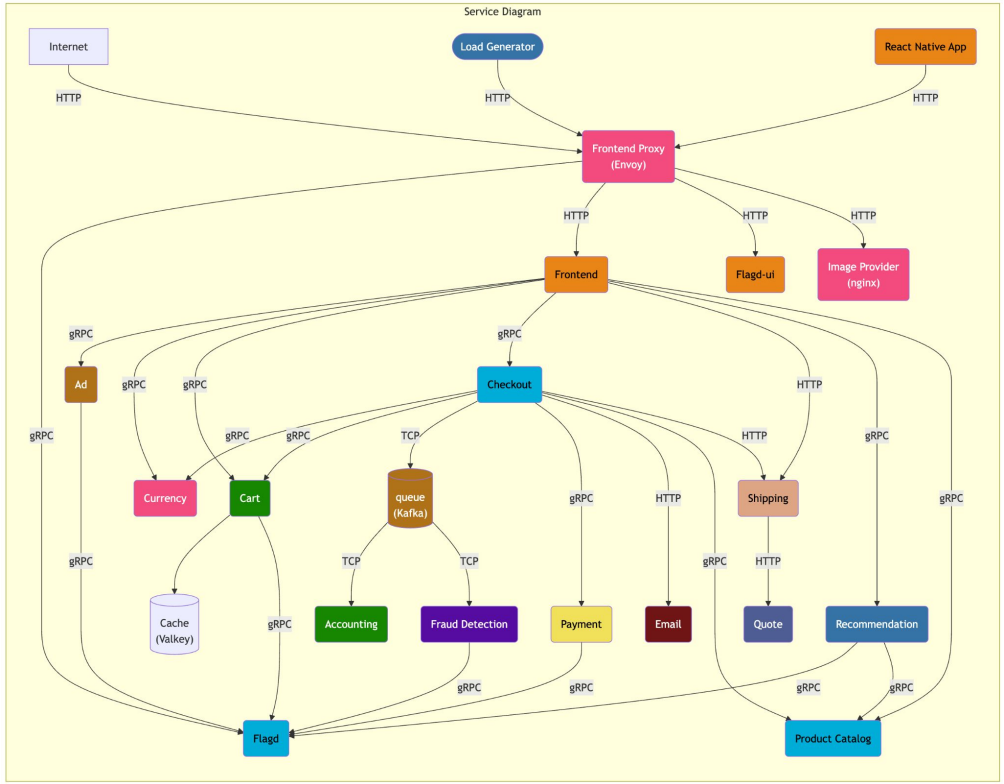
elastic

OpenTelemetry demo application



elastic

OpenTelemetry demo application



elastic

OpenTelemetry demo application

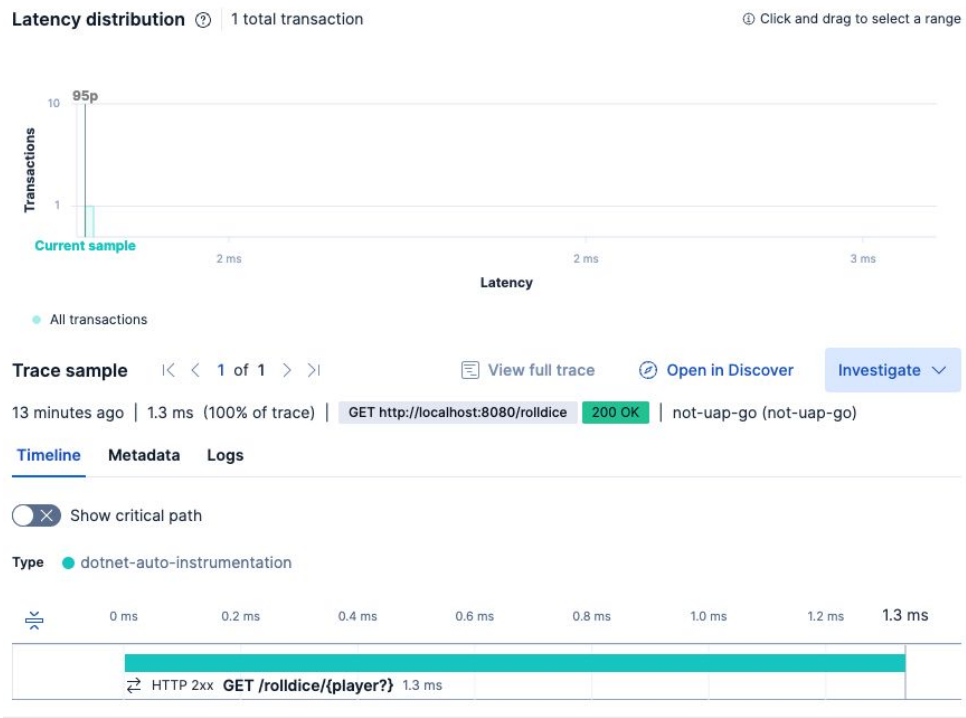
Language Feature Reference

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

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

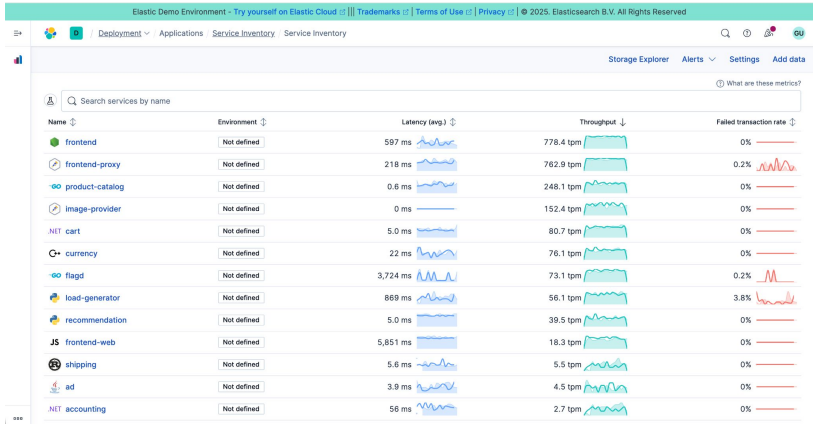
Demo time!

Houston, we have telemetry!

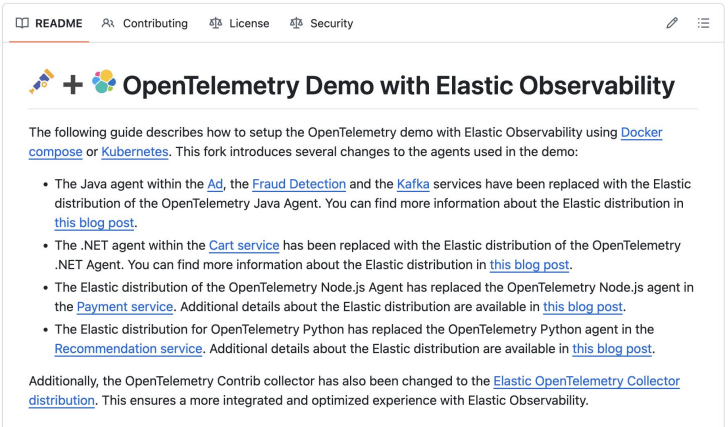


elastic

Want to try yourself?



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



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

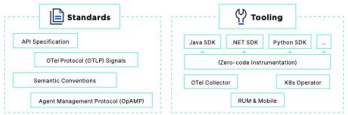
elastic

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



Standards

- **API Specification:** Defines cross-language tracing, metrics, and logs APIs
- **OTel Protocol:** Default protocol to transmit telemetry signals
- **Semantic Conventions:** Standard attribute keys for consistent data
- **OpAMP:** Protocol to manage and configure collectors and agents remotely

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 reliability
- **Contrib:** Superset of core with many additional components. Faster moving, includes experimental or community-contributed 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 variab

- **General**
  - OTEL_RESOURCE_AT
  - OTEL_LOG_LEVEL=d
- **OTLP Exporter**
  - OTEL_EXPORTER_
  - OTEL_EXPORTER_
- **Signal-specific en**
  - OTEL_EXPORTER_
  - OTEL_EXPORTER_
  - OTEL_EXPORTER_
- **Sampling**
  - OTEL_TRACES_SA
  - OTEL_TRACES_SA

OpenTelemetry

- Language-spec
  and generate te
- Provide APIs fo
  options to captu
- Expose standar
  service.version,
- Support multipl
  (depending on l
- Pluggable expo
  or through a Col
- Lightweight: Pu
  Collector for eff
- Available for ma
  Go, Node.js, PH

SDK distributions

- Official SDKs: M
- Auto-instrumen
  Python CLI) that
- Vendor distros:
  enhanced suppo
- Custom Builds:
  use cases, exper

Best practices

- Always set service.na
- Use semantic convention
- Filter and transform in Collector, r
- 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 gRPC
Collector OTLP HTTP
Prometheus scrape

ottl.run: Test OpenTelemetry Transformation Language expressions | otelbin.io: Build, run, and share OTel Collector configs in browser | Otel-tui A terminal OpenTelemetry viewer
Otel-desktop-viewer a CLI tool for receiving OpenTelemetry traces | Elastic OTel docs: elastic.co/docs/reference/opentelemetry

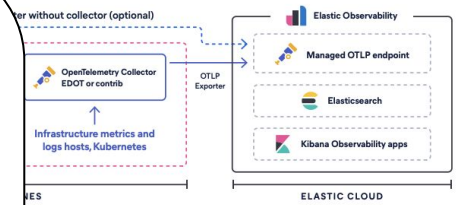
ela.st/
otel-cheat-sheet

OpenTelemetry + Elastic

Exploring Elastic Observability with OTel

Option 1: Explore online

- Visit 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_RESOURCE_ATTRIBUTES="service.name=checkout,deployment
.environment.name=prod"
```

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

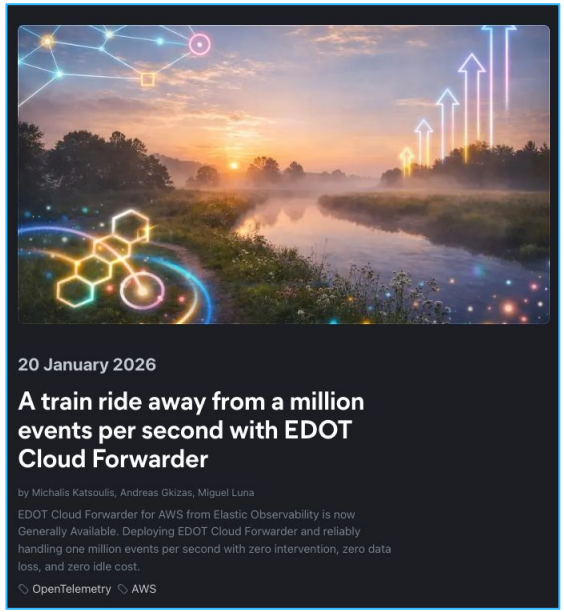
```
None
export OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4317
export OTEL_EXPORTER_OTLP_HEADERS="Authorization=ApiKey
<YourApiKey>"
```

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.

elastic

RESOURCES FOR DEVELOPERS | BY DEVELOPERS LIKE YOU!



elastic.co/
observability-labs

elastic

Conclusion

✨ Observability ✨

✨ OpenTelemetry ✨

✨ OTel Operator ✨

✨ OTel demo application ✨

Questions?

Thank you!

Let’s connect!

Evelien Schellekens

Principal Solutions Architect at Elastic

elastic