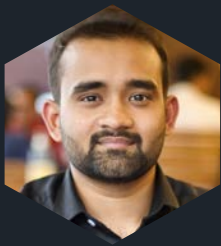




OpenTelemetry 101



Zameer Fouzan

Senior Developer Relations Engineer



Agenda

- 01 OpenTelemetry Background

- 02 Core Concepts of OpenTelemetry

- 03 Instrumentation - Hands-on

- 04 Open Telemetry Collector

A meme image featuring a young girl with brown hair in the foreground, looking slightly to the side with a neutral expression. In the background, a house is on fire, with bright orange flames and smoke. A yellow fire hose is visible on the ground, and a firefighter in a white helmet and dark uniform is standing near the house. The text "WORKED FINE IN DEV" is overlaid in white, bold, sans-serif font across the top half of the image.

**WORKED FINE
IN DEV**

OPS PROBLEM NOW

OpenTelemetry

Background

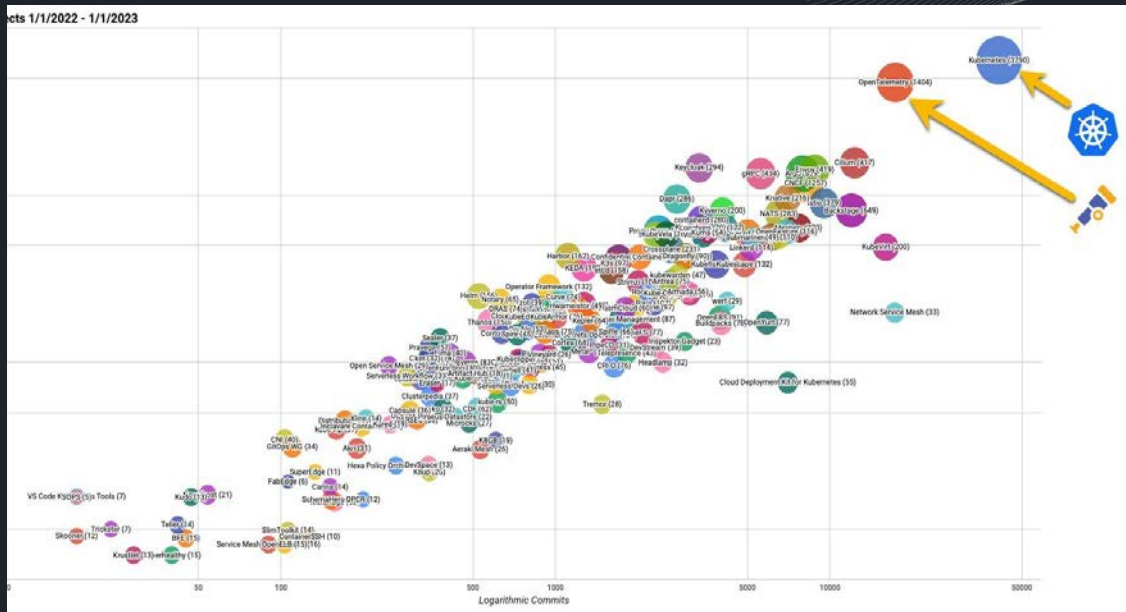
OpenTelemetry

Key Facts on OpenTelemetry

- OpenTelemetry is an **Incubating** project of CNCF.
- Formed through a merger of the **OpenTracing** and **OpenCensus** projects.
- **Vendor agnostic** - set of APIs, libraries, integrations, and a collector service for telemetry.
- **Standardizes** how you collect telemetry data from your applications and services.
- Send it to an **Observability platform** of your choice.



The Rise of OpenTelemetry



OpenTelemetry is second in CNCF - strong interest in modern **Observability**.

Enhanced support for **Open Standards**, including OpenTelemetry, eBPF, and Grafana - Gartner MQ 2022

The Rise of OpenTelemetry



Ubiquity

Promotes better coverage
for instrumentation



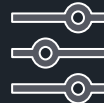
Vendor Neutral

Provides flexibility to
change backend



Interoperable

End-to-end visibility with
standard instrumentation



Configurable

Pick and choose from the
pieces what is needed

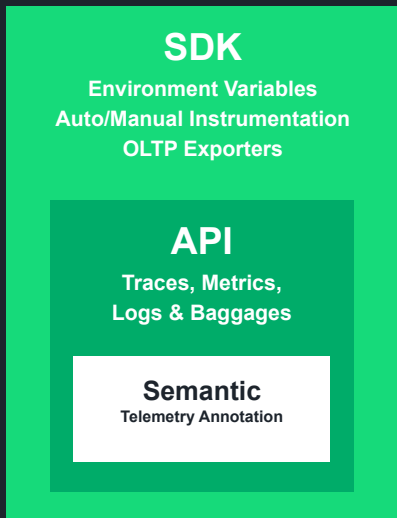
By 2025, **70%** of new cloud-native application monitoring will use open source instrumentation

⁽¹⁾ Source: Gartner Magic Quadrant 2021 for Application Performance Monitoring - [link](#)

Core Concepts

The building blocks

OpenTelemetry - Building blocks



Core Concepts on Instrumentation

- **Semantic Conventions** - annotate telemetry with attributes specific to the represented operation, such as HTTP calls.
- **API** - data types for tracing, metrics, and logging data.
- **SDK** - language-specific implementation of the API.
- SDKs incorporate automatic instrumentation for common libraries and frameworks for your application.
- **OpenTelemetry Protocol (OTLP)** - used to send data to your backend Observability platform of choice.

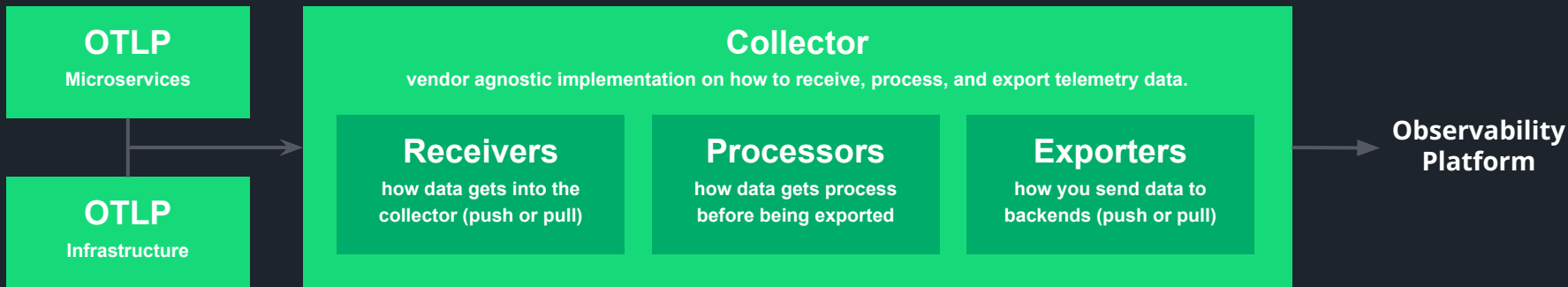
Application Instrumentation

Hands-On+Demo

Infrastructure

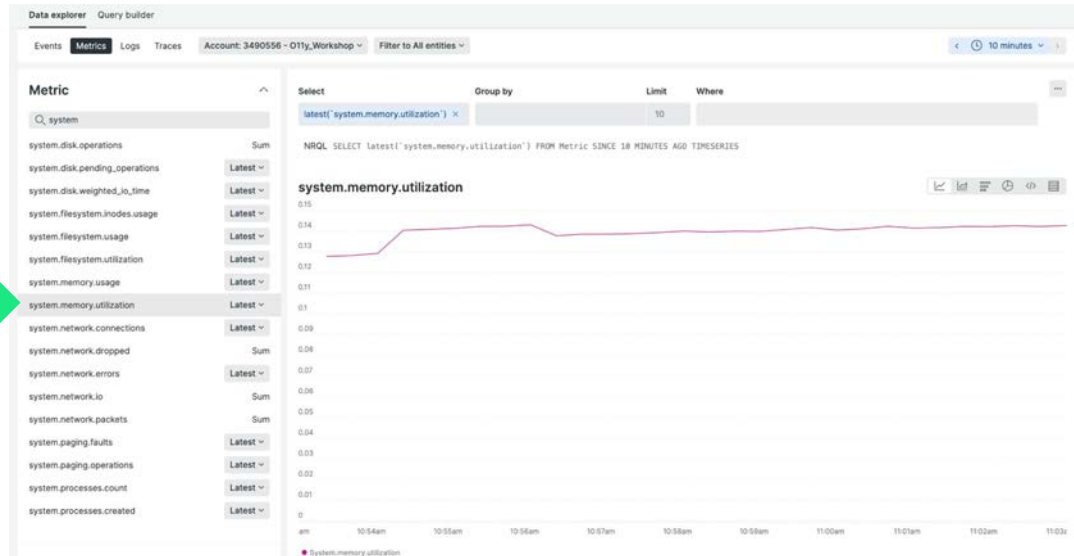
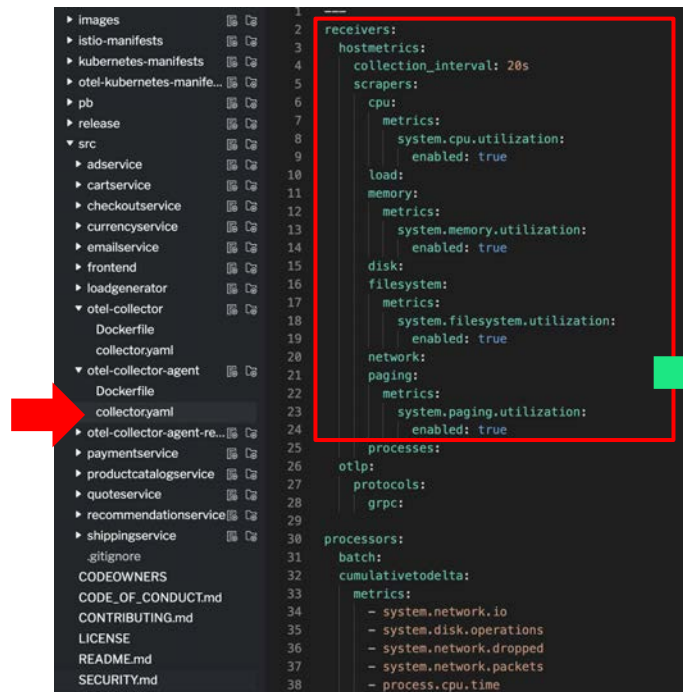
The Collector

OpenTelemetry - Collector



a **proxy** that process multiple telemetry formats, via an **agent** or a **gateway** (e.g., OTLP, Jaeger, Prometheus, as well as many commercial/proprietary tools)

OpenTelemetry - Collector



Collect additional infrastructure metrics
for OpenTelemetry

OpenTelemetry - Collector

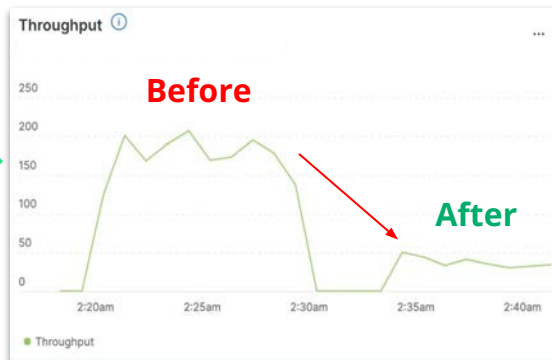
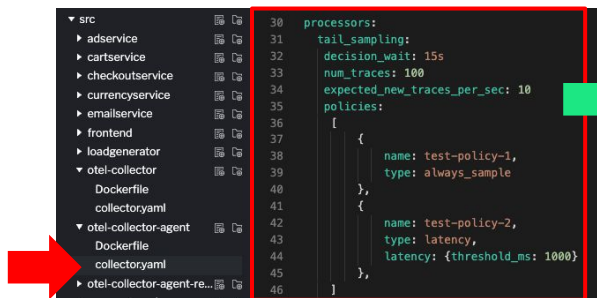
Storing all tracing data is **costly**. Most of the time, you **ONLY NEED THE RIGHT DATA**, when outages are occurring, or patterns are emerging.



Head based sampling works well for an overall statistical sampling of requests through a distributed system.

Tail based sampling is best to decide what to keep, based on isolated, independent portions of the trace data.

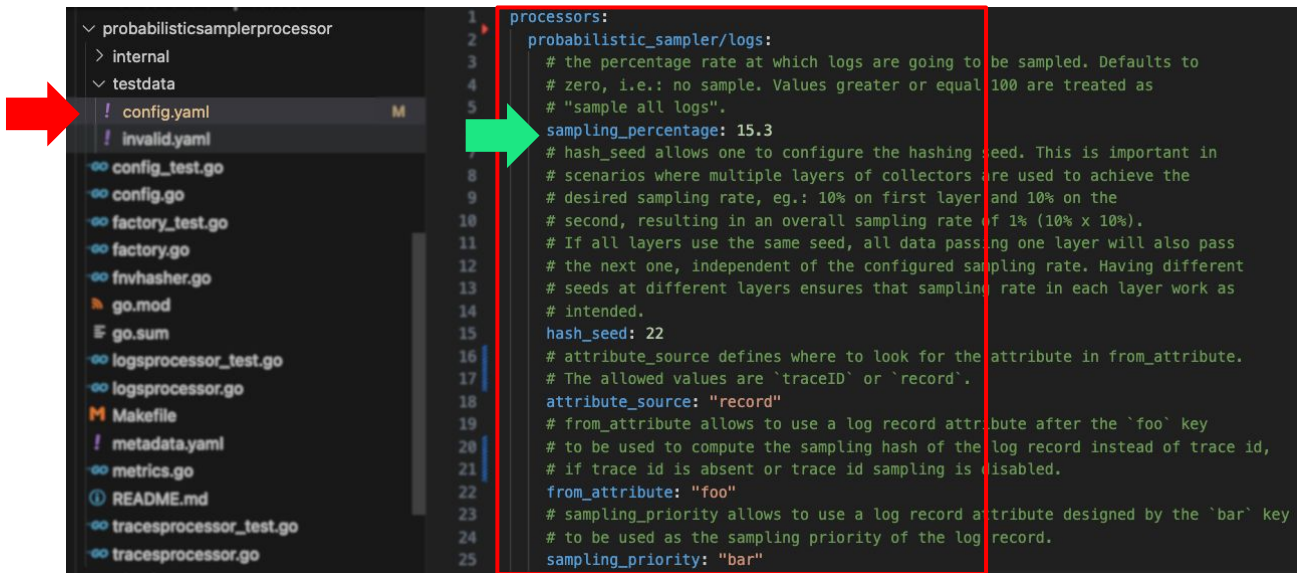
OpenTelemetry - Collector



Effects from Tail Based Sampling

Apply processing decision in the Collector, before shipping data out

OpenTelemetry - Collector



Probabilistic Based Sampling

Severless

AWS Lambda

Lambda Instrumentation

```

  "opentelemetry/instrumentation-aws-lambda": {
    enabled: true,
    disableAwsContextPropagation: true,
    requestHook: (span, { event, context }) => {
      span.setAttribute("faas.name", context.functionName);

      if (event.requestContext && event.requestContext.http) {
        span.setAttribute(
          "faas.http.method",
          event.requestContext.http.method
        );
        span.setAttribute(
          "faas.http.target",
          event.requestContext.http.path
        );
      }
    },
    responseHook: (span, { err, res }) => {
      if (err instanceof Error)
        span.setAttribute("faas.error", err.message);
      if (res) {
        span.setAttribute("faas.http.status_code", res.statusCode);
      }
    },
  },
},

```

- Official Lambda library
- Request Hooks
- Response Hooks
- Custom Attributes

```
NODE_OPTIONS: "--require ./otel-wrapper"
```

Lambda Instrumentation

The screenshot displays the New Relic One interface for a function named **otel-sls-sdk-dev-api**. The interface is divided into several sections:

- Left Sidebar:** Contains navigation links for various New Relic products and services.
- Top Section:** Shows the function name, environment (Node-Lambda-Otel-SDK), date and time (November 19 at 5:31pm), trace ID (0b93397ddcaccf11a8ab04ff24deee1), and spans (22).
- Trace Details:** A table listing the spans of the trace. A green arrow points to the **GET /api/weather** span, which has a duration of 20.15 ms. Below this, a detailed view of the span is shown, including a timeline of operations and their durations.
- Attributes:** A table listing the attributes of the span. Two attributes are highlighted with red boxes and green arrows: **http.route** (value: /api/weather) and **http.target** (value: /api/weather?location=bangalore).

Operation	Duration
otel-sls-sdk-dev-api	59.74 ms
GET /api/weather	20.15 ms
middleware - urlencodedParser	0.02 ms
middleware - <anonymous>	0.03 ms
router - /api/weather	0.01 ms
request handler - /	0.00 ms
middleware - query	0.05 ms
middleware - expressInit	0.03 ms
middleware - jsonParser	0.03 ms
GET	18.28 ms
Uninstrumented time	8.58 ms

Attribute	Value
http.host	3.230.230.121
http.method	GET
http.route	/api/weather
http.scheme	http
http.status_code	200
http.status_text	OK
http.target	/api/weather?location=bangalore
http.url	http://3.230.230.121/api/weather?location=bangalore
http.user_agent	axios/1.6.1
id	304ba16ee5677b62
instrumentation.provider	opentelemetry

Recap and Highlights

Exciting times for Open Source Observability!

OpenTelemetry is growing and being adopted at a rapid pace.

Be mindful with your maturity, and plan ahead on the adoption of OpenTelemetry.

Just having telemetry is **NOT** observability. Instrumentation should include contextual **traces**, **logs**, or **metrics** to improve observations.

Deployment of the OpenTelemetry collector is easier to gather Telemetry.

Gather telemetry from your pipelines to measure your MTTI, MTTD, MTTR

Actively investing in OpenTelemetry, and help engineers work based on **"data, not opinions"**.

Resources

Further Reading

Open Telemetry Resources

- [Application Instrumentation](#)
- [Open Telemetry on Serverless](#)
- [Open Telemetry in CI/CD](#)
- [Collector](#)
 - [Sampling Strategies](#)
- Collector on Kubernetes
 - Configuration - <https://opentelemetry.io/docs/collector/configuration/>
 - Helm - <https://opentelemetry.io/docs/kubernetes/helm/collector/>
 - Operator - <https://opentelemetry.io/docs/kubernetes/operator/>
 - Automatic instrumentation - <https://opentelemetry.io/docs/kubernetes/operator/automatic/>

\$whoami



Zameer Fouzan

Senior Developer Relations Engineer

