OTel Without Reservations

Ethan Kent

June 21, 2023

Overview

Introduction

What is OpenTelemetry?

According to the OpenTelemetry people,[1]

> *OpenTelemetry is a collection of tools, APIs, and SDKs. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.*[2]

[1] OpenTelemetry is a project of The Cloud Native Computing Foundation and is a merger of the OpenTracing and OpenCensus projects.

[2] The OpenTelemetry Authors. **OpenTelemetry.** 2023. URL: `https://opentelemetry.io/` (visited on 06/20/2023).

Why do I care?

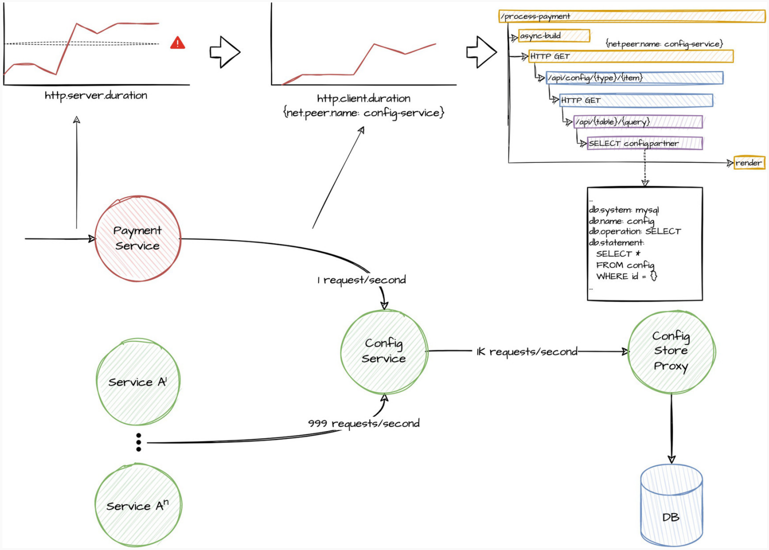
A distributed trace, more commonly known as a trace, records the paths taken by requests (made by an application or end-user) as they propagate through multi-service architectures, like microservice and serverless applications.

Without tracing, it is challenging to pinpoint the cause of performance problems in a distributed system.

It improves the visibility of our application or system's health and lets us debug behavior that is difficult to reproduce locally. Tracing is essential for distributed systems, which commonly have nondeterministic problems or are too complicated to reproduce locally.[3]

[3] The OpenTelemetry Authors. **Obervability Primer, OpenTelemetry.** 2023. URL: `https://opentelemetry.io/docs/concepts/observability-primer/` (visited on 06/20/2023).

Why do I care? (cont'd)



```
...
db.system: mysql
db.name: config
db.operation: SELECT
db.statement:
  SELECT *
  FROM config
  WHERE id = {}
...
```

Why do I care? (cont'd)

Observability OpenTelemetry (OTel) offers comprehensive observability across your systems. OTel provides a clear picture of what's happening within your services by bringing together traces, metrics, and logs.

Reduced Complexity OTel unifies multiple observability signals into a coherent framework. A single framework simplifies instrumentation and eliminates the need to learn and manage various systems.

Standards-Based OTel is open-source and vendor-neutral, providing standardized APIs and instrumentation that integrates with any backend.

Extensibility OTel's modular design allows you to use what you need, and its SDKs let you build custom integrations as required.

Cost and Resource Efficient OTel can reduce the overhead and costs of running multiple observability tools.

Why do I care? (cont’d)

Support for Modern Architectures OTel provides first-class support for Kubernetes, serverless functions, and service meshes.

Enhanced Debugging With better visibility, engineers can identify, understand, and resolve issues faster.

Performance Monitoring OTel helps you monitor system performance and user behavior in real-time, helping you make data-driven decisions to improve your services’ performance and user experience.

Future-Proof OTel’s wide adoption and large community likely mean that as new standards and best practices emerge, OTel will evolve with them.

Signals

Signals

"In OpenTelemetry, a signal refers to a category of telemetry."[4]

The four kinds of signals currently supported are

- Traces,
- Metrics,
- Logs, and
- Baggage.[5]

[4]The OpenTelemetry Authors. **Signals, OpenTelemetry.** 2023. URL: https://opentelemetry.io/docs/concepts/signals/ (visited on 06/20/2023).
[5]The OpenTelemetry Authors, *Signals, OpenTelemetry*, see n. 4.

Traces

Spans are the bread and butter of distributed tracing. A *Trace* has many *Spans*.

> *Traces give us the big picture of what happens when a request is made to an application. Whether your application is a monolith with a single database or a sophisticated mesh of services, traces are essential to understanding the full "path" a request takes in your application.*[6]

[6] The OpenTelemetry Authors. **Traces, OpenTelemetry.** 2023. URL: `https://opentelemetry.io/docs/concepts/signals/traces/` (visited on 06/20/2023).

Metrics

A metric is a measurement about a service, captured at runtime. Logically, the moment of capturing one of these measurements is known as a metric event which consists not only of the measurement itself, but the time that it was captured and associated metadata.[7]

[7] The OpenTelemetry Authors. **Metrics, OpenTelemetry.** 2023. URL: https://opentelemetry.io/docs/concepts/signals/metrics/ (visited on 06/20/2023).

Metrics (cont'd)

Use metrics when you want metrics.

> *Some telemetry backends allow to execute ad hoc queries on spans, [and it] is also possible to derive metrics from spans in OpenTelemetry Collectors.* ***This often results in an overuse of spans as a substitute for metrics when teams start to adopt tracing.*** *Trace sampling can affect any interpretations extracted directly from spans, . . . [and] the high volumes of data and cardinality processed by trace pipelines in high-throughput systems normally result in signals that are less stable than metrics originating directly from a service . . . .*[8]

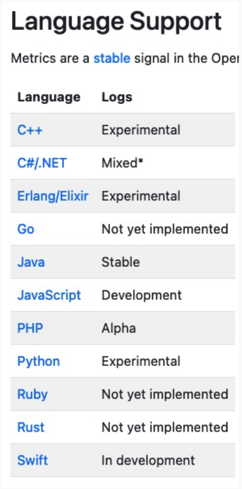
[8] Daniel Gomez Blanco. **Practical OpenTelemetry: Adopting Open Observability Standards Across Your Organization.** Berkeley, CA: Apress, 2023, ch. 6, emphasis added.

Logs

In OpenTelemetry, any data that is not part of a distributed trace or a metric is a log. For example, events are a specific type of log. Logs often contain detailed debugging/diagnostic info, such as inputs to an operation, the result of the operation, and any supporting metadata for that operation.[9]

[9]The OpenTelemetry Authors. **Logs, OpenTelemetry.** 2023. URL: https://opentelemetry.io/docs/concepts/signals/logs/ (visited on 06/20/2023).

Logs: not well-supported yet



Language Support

Metrics are a stable signal in the Ope

Language	Logs
C++	Experimental
C#/.NET	Mixed*
Erlang/Elixir	Experimental
Go	Not yet implemented
Java	Stable
JavaScript	Development
PHP	Alpha
Python	Experimental
Ruby	Not yet implemented
Rust	Not yet implemented
Swift	In development

Baggage

“In OpenTelemetry, Baggage is contextual information that’s passed between spans. It’s a key-value store that resides alongside span context in a trace, making values available to any span created within that trace.”[10]

It’s in the HTTP headers, so don’t store sensitive data.

“Common use cases include . . . Account Identification, User Ids, Product Ids, and origin IPs. . . . Passing these down your stack allows you to then add them to your Spans in downstream services to make it easier to filter when you’re searching in your Observability back-end.”[11]

[10] The OpenTelemetry Authors. **Baggage, OpenTelemetry.** 2023. URL: https://opentelemetry.io/docs/concepts/signals/baggage/ (visited on 06/20/2023).
[11] The OpenTelemetry Authors, *Baggage, OpenTelemetry*, see n. 10.

Context and Context Propagation

How does this distributed stuff work?

Distributed tracing is hard because it's tricky to reconstruct cause and effect. In distributed systems, many separate services—some internal and some external—may collaborate to produce a final result. The secret sauce for OpenTelemetry can be split into three pieces:

1. A trace ID, additional correlation IDs, and other metadata, collectively called "Context."
2. Standards governing how Context is transmitted within and among services, called "Context Propagation."
3. An ecosystem comprising open standards, libraries, SDKs, vendors, etc.

In short, OpenTelemetry is a well-thought-out system for assigning IDs in a tree-like structure, passing them across service boundaries, and allowing Humpty-Dumpty to be put back together again.

Context

A `Context` is a propagation mechanism which [sic] carries execution-scoped values across API boundaries and between logically associated execution units. Cross-cutting concerns access their data in-process using the same shared `Context` object.[12]

Execution Unit *An umbrella term for the smallest unit of sequential code execution, used in different concepts of multitasking. Examples are threads, coroutines or fibers.*[13]

[12]The OpenTelemetry Authors. **Context, OpenTelemetry.** 2023. URL: `https://opentelemetry.io/docs/specs/otel/context/` (visited on 06/21/2023).
[13]The OpenTelemetry Authors. **Glossary, OpenTelemetry.** 2023. URL: `https://opentelemetry.io/docs/specs/otel/glossary/` (visited on 06/21/2023).

Auto SDK

Manual

Collectors and Exporters

Backends