Introduction to distributed tracing

Maxime DAVID
CNCF User Group - November 2023

whoami >

Maxime David
AWS Community Builder
Serverless @Datadog

Youtuber (rust)

https://maxday.dev →

Agenda

1. Overview
2. Definitions, concepts and use cases
3. Demo
4. Going further
5. Questions!

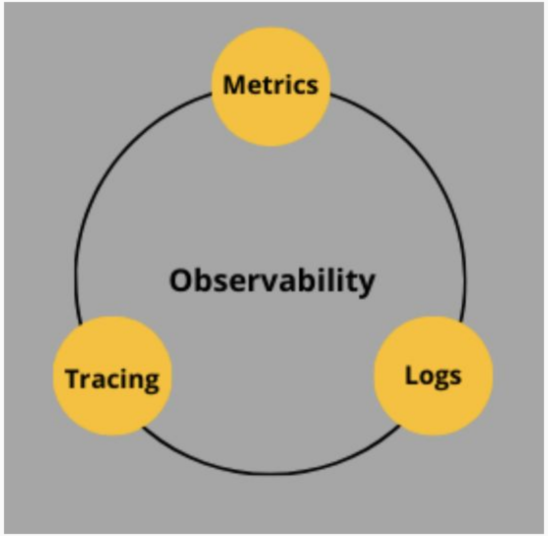
Observability != Monitoring

Observability refers to the ability to understand the internal state of a system by examining its outputs, such as **logs, metrics and traces**. It allows for the diagnosis of issues by providing insight into the system's behavior over time.

Monitoring refers to the continuous collection of data from a system to **check for any abnormal behavior or performance issues**

Without observability -> no monitoring!

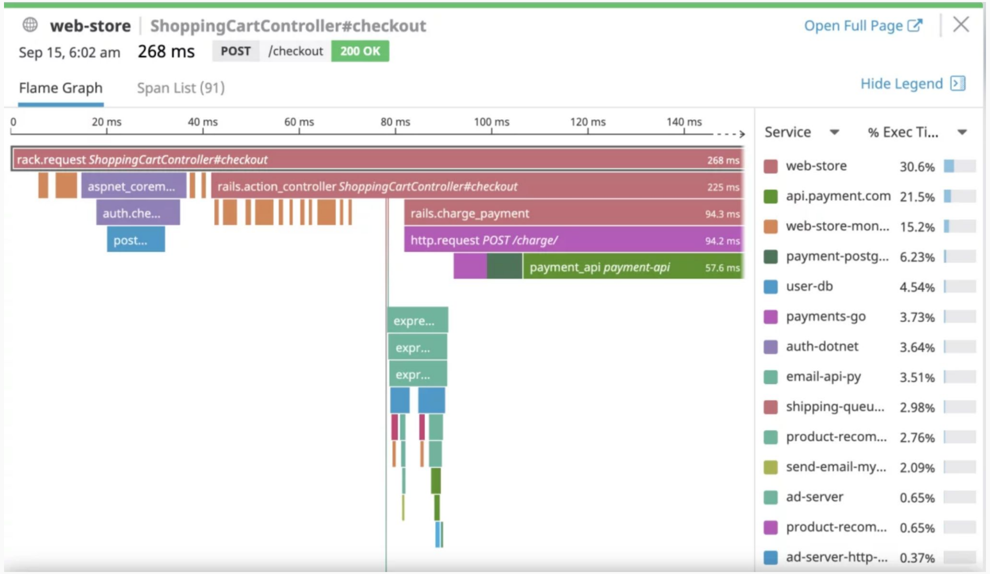
Three pillars of observability



Source: https://iamondemand.com/blog/the-3-pillars-of-system-observability-logs-metrics-and-tracing/

- **Logs** provide a record of events that occur within a system
- **Metrics** provide measurable values that can be used to track the performance and health of a system.
- **Traces** provide a detailed record of the steps taken by a request or process as it flows through a distributed system, and can be used for debugging and performance analysis.

Traces - Example



web-store | ShoppingCartController#checkout

Sep 15, 6:02 am 268 ms POST /checkout 200 OK

Service	% Exec Ti...
🟥 web-store	30.6%
🟩 api.payment.com	21.5%
🟧 web-store-mon...	15.2%
🟩 payment-postg...	6.23%
🟦 user-db	4.54%
🟪 payments-go	3.73%
🟪 auth-dotnet	3.64%
🟩 email-api-py	3.51%
🟥 shipping-queu...	2.98%
🟩 product-recom...	2.76%
🟨 send-email-my...	2.09%
🟩 ad-server	0.65%
🟪 product-recom...	0.65%
🟦 ad-server-http-...	0.37%

Source: https://docs.datadoghq.com/fr/video-categories/flamegraph/

OpenTelemetry

OpenTelemetry is a collection of tools, APIs, and SDKs.

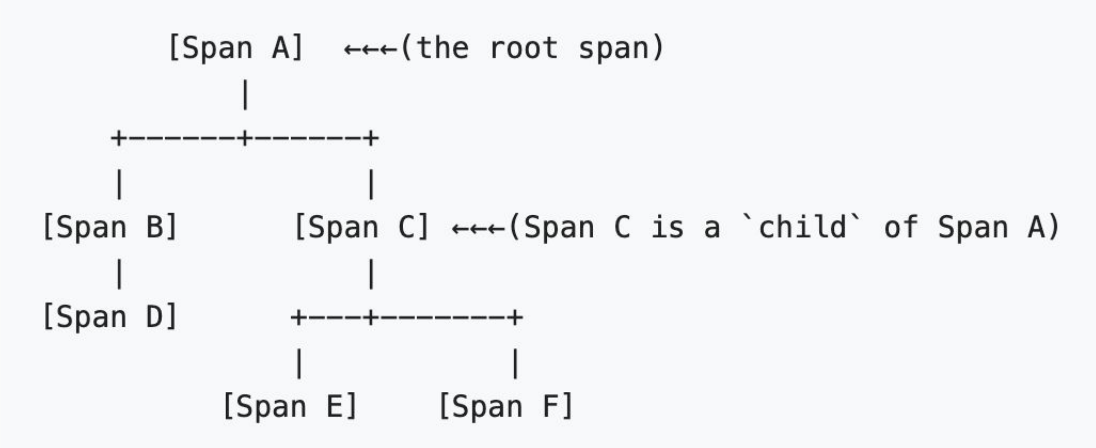
Use it to **instrument, generate, collect, and export** telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.

Source: https://opentelemetry.io/

NOT to store **NOR** visualize data

Focus on Tracing -> (more definitions 🙈)

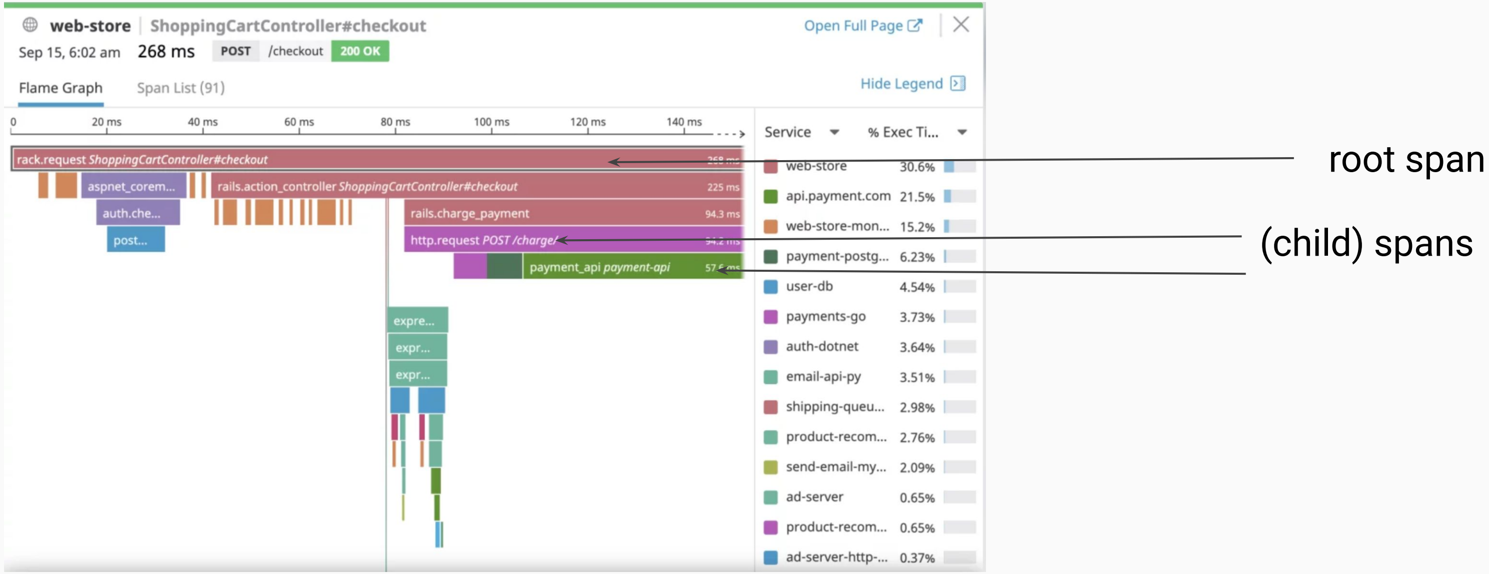
Traces can be viewed as a **directed acyclic graph** of **Spans**



```
        [Span A]  ←←←(the root span)
            |
     +------+------+
     |             |
 [Span B]      [Span C] ←←←(Span C is a `child` of Span A)
     |             |
 [Span D]      +---+-------+
               |           |
           [Span E]    [Span F]
```

Source: https://opentelemetry.io/docs/reference/specification/overview

Focus on Tracing



web-store | ShoppingCartController#checkout

Sep 15, 6:02 am 268 ms POST /checkout 200 OK

Service	% Exec Ti...
🟥 web-store	30.6%
🟩 api.payment.com	21.5%
🟧 web-store-mon...	15.2%
🟩 payment-postg...	6.23%
🟦 user-db	4.54%
🟪 payments-go	3.73%
🟪 auth-dotnet	3.64%
🟩 email-api-py	3.51%
🟥 shipping-queu...	2.98%
🟩 product-recom...	2.76%
🟩 send-email-my...	2.09%
🟩 ad-server	0.65%
🟪 product-recom...	0.65%
🟦 ad-server-http-...	0.37%

root span

(child) spans

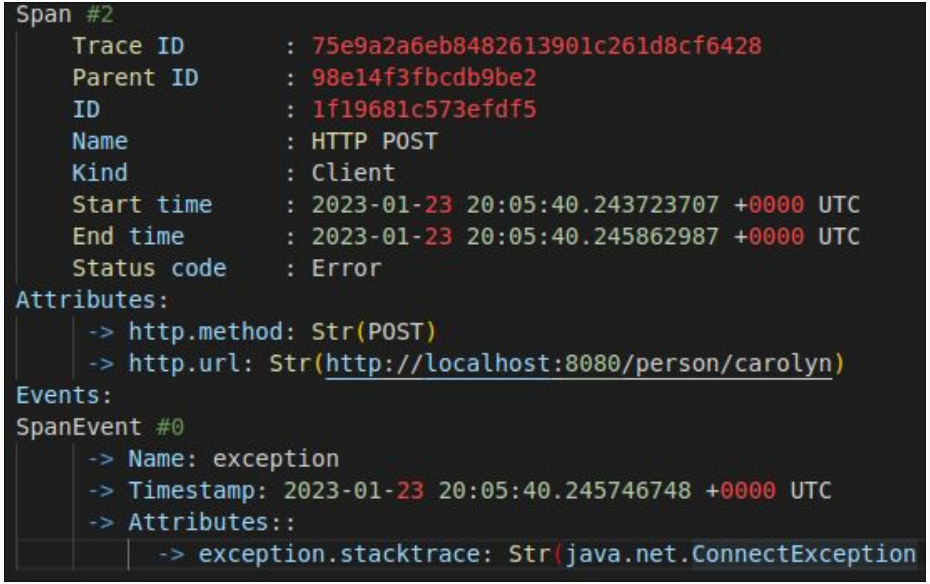
Source: https://docs.datadoghq.com/fr/video-categories/flamegraph/

Spans

Span = operation within a transaction.

Span contains

- Parent’s Span identifier (remember the DAG)
- An operation name
- A start and finish timestamp
- Attributes -> key-value pairs.
- A set of zero or more Events, each of which is itself a tuple (timestamp, name, attributes)



```
Span #2
    Trace ID       : 75e9a2a6eb8482613901c261d8cf6428
    Parent ID      : 98e14f3fbcdb9be2
    ID             : 1f19681c573efdf5
    Name           : HTTP POST
    Kind           : Client
    Start time     : 2023-01-23 20:05:40.243723707 +0000 UTC
    End time       : 2023-01-23 20:05:40.245862987 +0000 UTC
    Status code    : Error
Attributes:
     -> http.method: Str(POST)
     -> http.url: Str(http://localhost:8080/person/carolyn)
Events:
SpanEvent #0
     -> Name: exception
     -> Timestamp: 2023-01-23 20:05:40.245746748 +0000 UTC
     -> Attributes::
          -> exception.stacktrace: Str(java.net.ConnectException
```

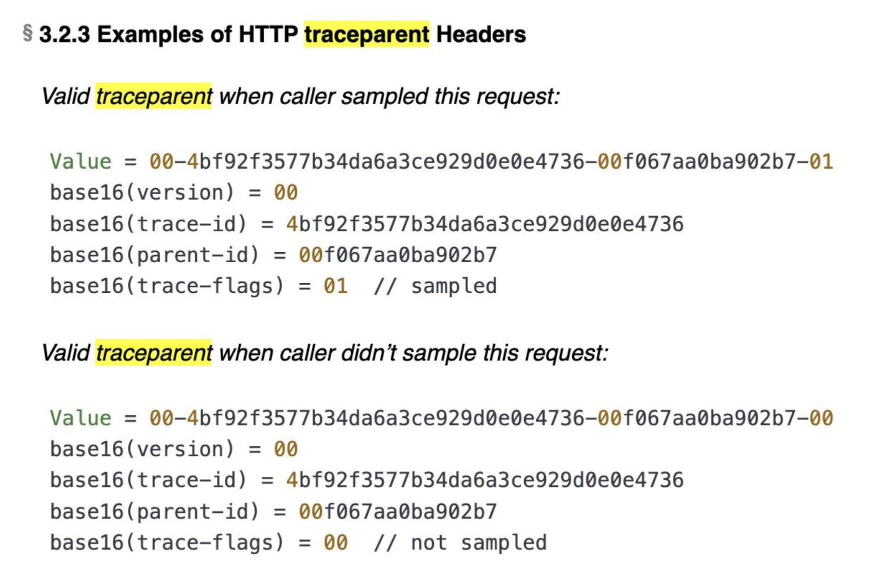
Trace Propagation

When possible:

- via http headers

Otherwise:

- custom solutions :)



§ **3.2.3 Examples of HTTP traceparent Headers**

Valid traceparent when caller sampled this request:

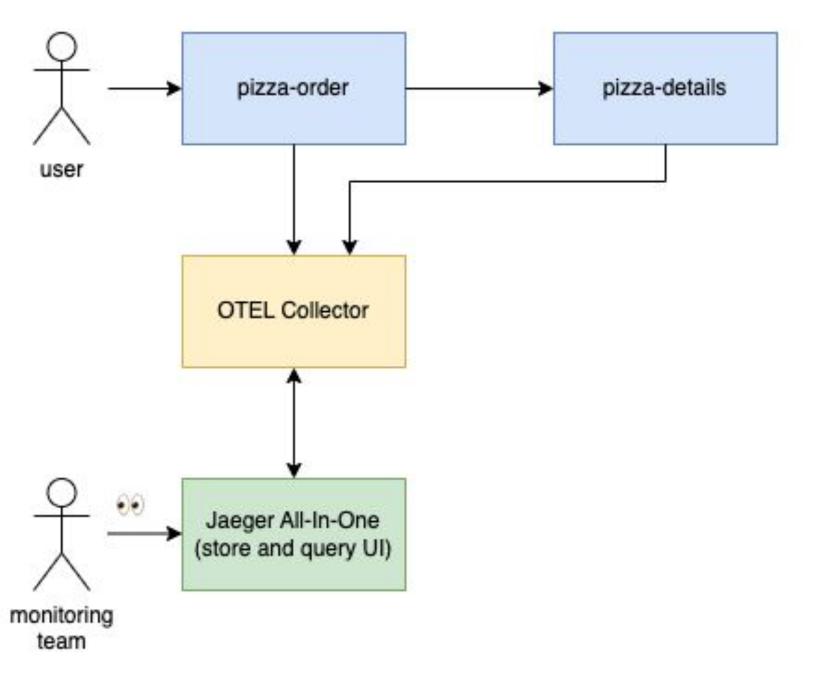
```
Value = 00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01
base16(version) = 00
base16(trace-id) = 4bf92f3577b34da6a3ce929d0e0e4736
base16(parent-id) = 00f067aa0ba902b7
base16(trace-flags) = 01  // sampled
```

Valid traceparent when caller didn't sample this request:

```
Value = 00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-00
base16(version) = 00
base16(trace-id) = 4bf92f3577b34da6a3ce929d0e0e4736
base16(parent-id) = 00f067aa0ba902b7
base16(trace-flags) = 00  // not sampled
```

Source: https://www.w3.org/TR/trace-context/#traceparent-header-field-values

Demo 🍕


user
pizza-order
pizza-details
OTEL Collector
👀
Jaeger All-In-One
(store and query UI)
monitoring
team

DEMO 🔥

Going further

Correlation between pillars

- Trace with a full context (metrics + logs)

More than 3 pillars

- Profiling
- Code integration

Visualization

- You likely need a tool to visualize

Thanks!

Maxime David
AWS Community Builder
Serverless @Datadog

Youtuber (rust)

https://maxday.dev →