Short-Lived, Long Traced. Observability Deep Dive in Serverless

Maxime DAVID
CNCF User Group - Dublin - June 2025

Serverless

How to observe ephemeral executions, cold starts, and short-lived functions?

whoami >

Maxime David (maxday)
He/him

Senior Software Engineer @ AWS
AWS Lambda Runtimes team
CNCF OpenTelemetry Contributor
Speaker

Banana Bread lover

https://maxday.dev →

Why do we want to track?

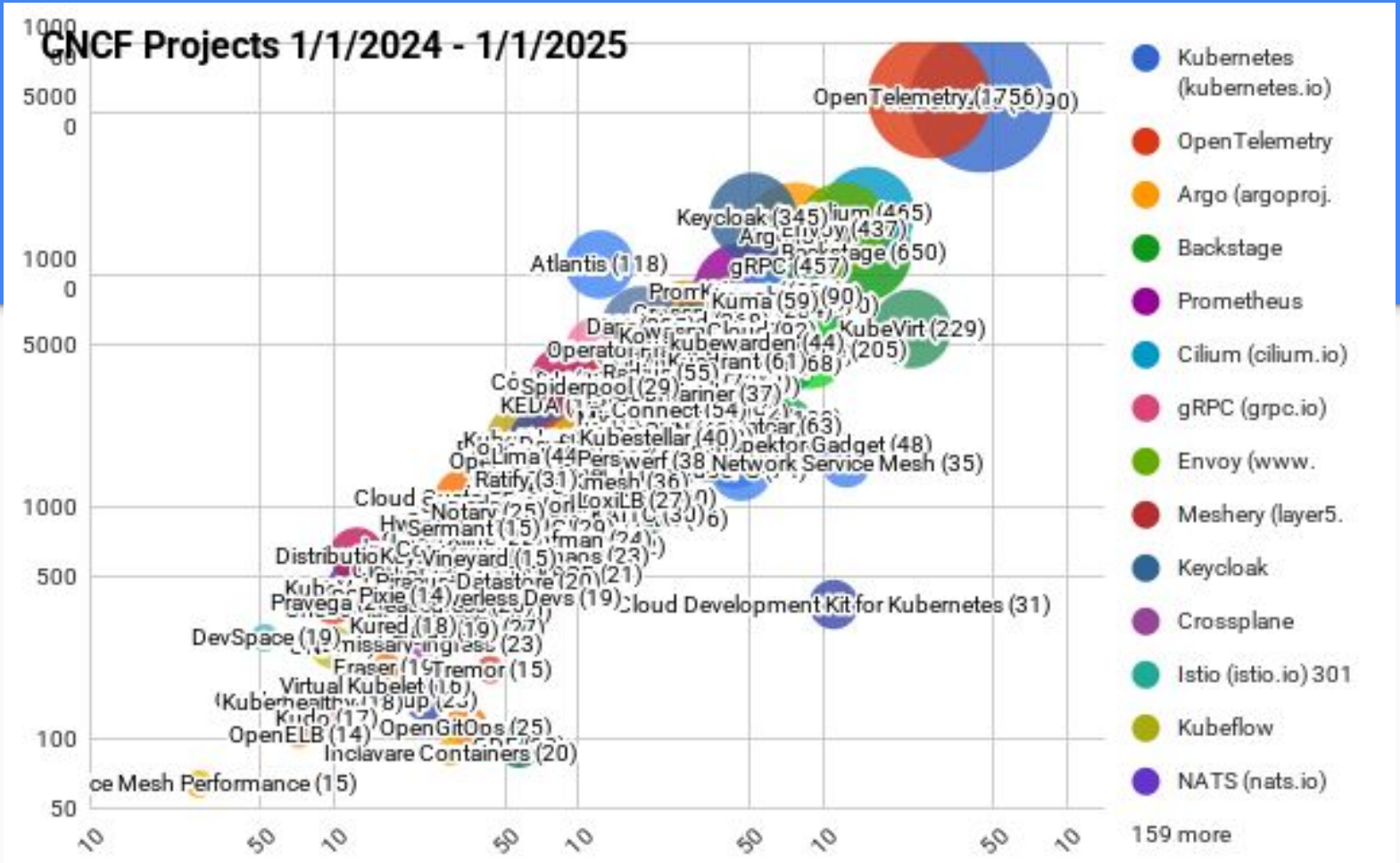
OpenTelemetry

Wait! Another tool?

OpenTelemetry was accepted to CNCF on May 7, 2019

and moved to the Incubating maturity level on August 26, 2021.

https://www.cncf.io/projects/opentelemetry/



Source: https://www.cncf.io/blog/2025/01/29/2024-year-in-review-of-cncf-and-top-30-open-source-project-velocity/

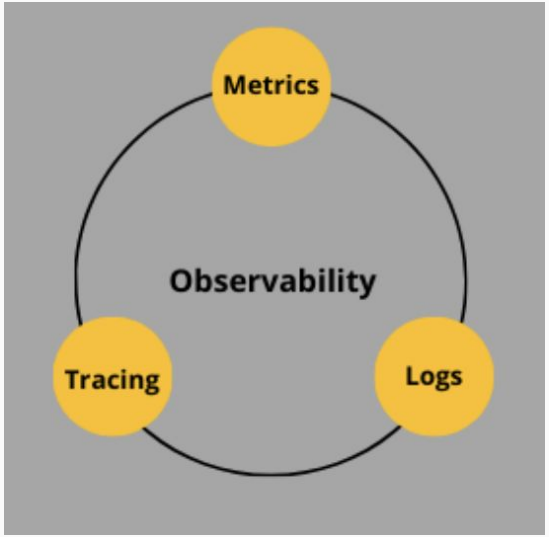
Observability != Monitoring

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.

Logs - Example

Plain text (apache access log)

```
10.1.2.3 - rehg [20/Jan/2023:19:22:12 -0000] "GET /hi-CNCF  HTTP/1.1" 200 3423
```

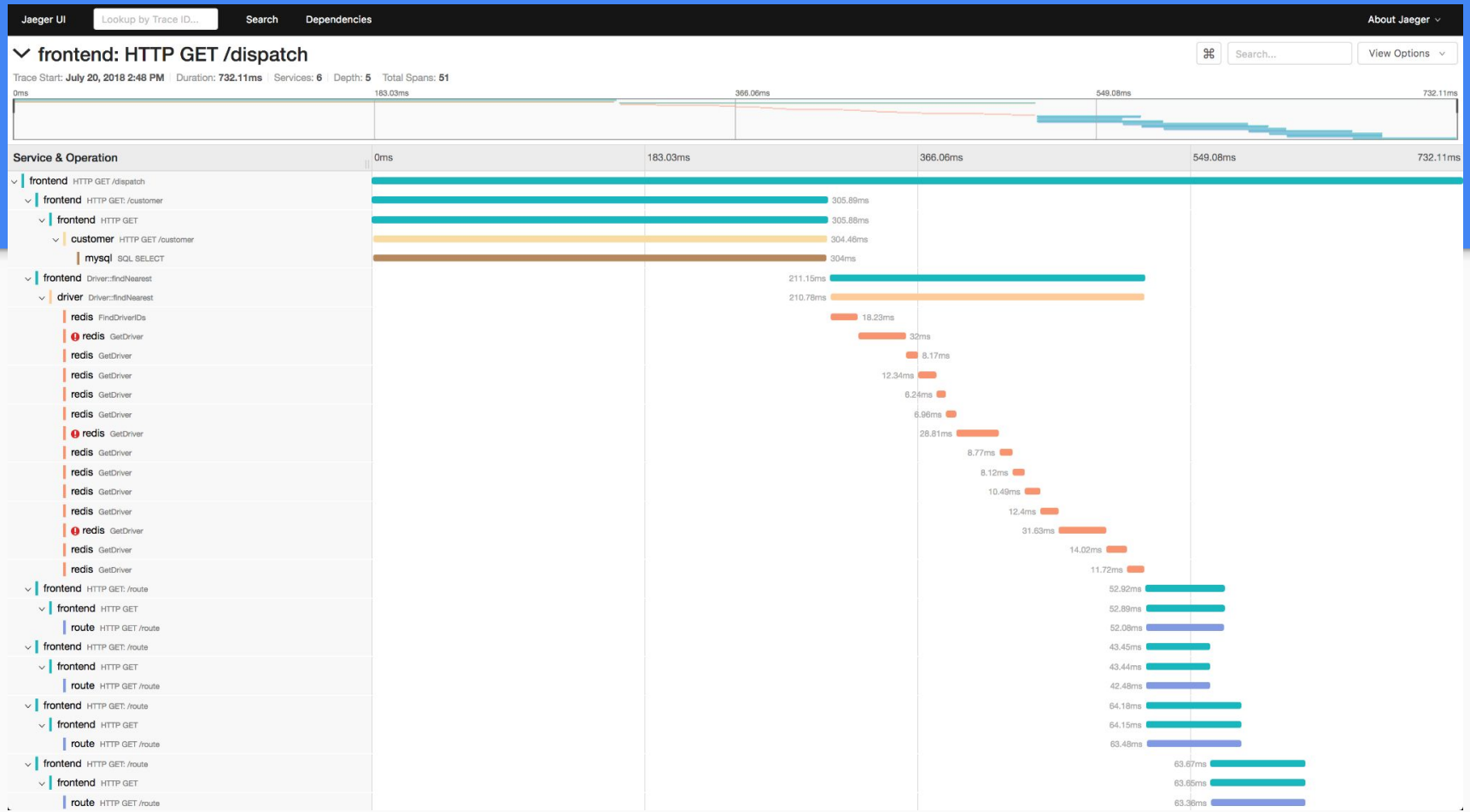
Logs - Example

JSON


```
{
    "time": "2025-06-12T11:42:17.939Z",
    "type": "platform.initStart",
    "record": {
        "initializationType": "on-demand",
        "phase": "init",
        "runtimeVersion": "nodejs:22.vXX",
        "runtimeVersionArn": "arn:aws:lambda:eu-west-1::runtime:fd2e05b3d9bd2e",
        "functionName": "nodejs-maxday",
        "functionVersion": "$LATEST",
        "instanceId": "2025/06/12/nodejs-maxday[$LATEST]f34b010aa5d5419a63dee6596f15a",
        "instanceMaxMemory": 134217728
    }
}
```

Metrics - Example

- System metrics
  - system.cpu.idle
  - redis.keys.evicted
- Business metrics
  - my.project.cart.item.added
  - my.project.password.forget.co



Source: https://www.jaegertracing.io/docs/1.49/

Focus on Tracing -> (more definitions 🙈)

Traces can be viewed as a **<u>directed acyclic graph</u>** 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

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
```

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

How does it work?

Instrumentation

opentelemetry-js-contrib / plugins / node /

↑ Top

instrumentation-amqplib	chore: release main (#2834)	last week
instrumentation-cucumber	chore: release main (#2834)	last week
instrumentation-dataloader	chore: release main (#2834)	last week
instrumentation-fs	chore: release main (#2834)	last week
instrumentation-kafkajs	chore: release main (#2834)	last week
instrumentation-lru-memoizer	chore: release main (#2834)	last week
instrumentation-mongoose	chore: release main (#2834)	last week
instrumentation-runtime-node	chore: release main (#2834)	last week
instrumentation-socket.io	chore: release main (#2834)	last week
instrumentation-tedious	chore: release main (#2834)	last week
instrumentation-typeorm	chore: release main (#2834)	last week
instrumentation-undici	fix(instr-undici): fix user agent extraction and handle of multiple v...	last week
opentelemetry-instrumentation-aws-lambda	chore: release main (#2834)	last week

Collector

github.com/open-telemetry/opentelemetry-collector

open-telemetry / opentelemetry-collector

Type / to search

Code | Issues 617 | Pull requests 41 | Discussions | Actions | Projects 1 | Security 1 | Insights

opentelemetry-collector Public

Watch 89 | Fork 1.6k | Star

main | 127 Branches | 3723 Tags | Go to file t | Add file | Code

jade-guiton-dd [service] Share log sampler core allocations with reflect ma... 4ddf2cc · yesterday 7,210 Commits

.chloggen	[service] Share log sampler core allocations with reflect ...	yesterday
.github	[semconv] remove deprecated package (#13071)	2 days ago
client	fix(deps): update module google.golang.org/grpc to v1.73...	2 days ago

About

OpenTelemetry Collector

opentelemetry.io

monitoring | metrics | telemetr
observability | opentelemetry
open-telemetry

Readme

Exporter

NOT to store **NOR** visualize data

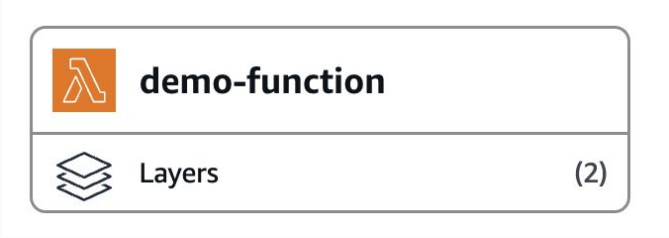
=> 3rd party

In Lambda?

index.mjs > handler

```javascript
import { S3Client, PutObjectCommand } from "@aws-sdk/client-s3";

const s3Client = new S3Client({});

export const handler = async (event) => {
    const bucketName = 'maxday-test-cncf';
    const fileName = `data-${new Date().toISOString()}.txt`;
    const content = 'Hello from CNCF Dublin!';

    const params = {
        Bucket: bucketName,
        Key: fileName,
        Body: content
    };

    try {
        const command = new PutObjectCommand(params);
        await s3Client.send(command);

        return {
            statusCode: 200,
            body: JSON.stringify(`Successfully wrote file ${fileName} to S3`)
        };
    } catch (error) {
        console.error('Error:', error);
        return {
            statusCode: 500,
            body: JSON.stringify(`Error: ${error.message}`)
        };
    }
};
```

In Lambda?



Layers Info

Merge order	Name
1	opentelemetry-collector-arm64-0_15_0
2	opentelemetry-nodejs-0_14_0

Environment variables (3)

The environment variables below are encrypted at rest with the default Lambda service key.

Find environment variables

Key	Value
AWS_LAMBDA_EXEC_WRAPPER	/opt/otel-handler
OTEL_TRACES_EXPORTER	console
OTEL_TRACES_SAMPLER	always_on

In Lambda?

Message


```
traceState: undefined
},
traceState: undefined,
name: 'S3.PutObject',
id: '4bf883f70a446c9a',
kind: 2,
timestamp: 1749733437484000,
duration: 32643.717,
attributes: {
'rpc.system': 'aws-api',
'rpc.method': 'PutObject',
'rpc.service': 'S3',
'aws.s3.bucket': 'maxday-test-cncf',
'aws.region': 'eu-west-1',
'aws.request.id': '538DH64KSY8Y21QY',
'http.status_code': 200,
'aws.request.extended_id': 'aAYj1WPQYOk0aBusLF0h6vr+MXBCu3QvyK7TzydFtwMs5Lys9H1Bd8IEb0EHDErSdCxCTJ6rOqQ='
},
status: { code: 0 },
events: [],
links: []
}
```

Thank you!

Maxime David (maxday)
He/him

Senior Software Engineer @ AWS
AWS Lambda Runtimes team
CNCF OpenTelemetry Contributor
Speaker

Banana Bread lover

https://maxday.dev →