

The background features a vibrant, multi-colored abstract design. On the left, there are overlapping, wavy bands of color in shades of orange, red, and yellow. On the right, a bright white light source emits a series of colorful rays in shades of blue, green, and yellow, creating a sunburst effect. The overall composition is dynamic and energetic.

cisco *Live!*

Let's go

#CiscoLive



The bridge to possible

# Do Tell About OTel:

An Introduction to OpenTelemetry and How  
AppDynamics is Embracing It

Wayne Brown, AppDynamics SE Manager

@thewaynebrown

BRKAPP-1154



#CiscoLive

# Cisco Webex App

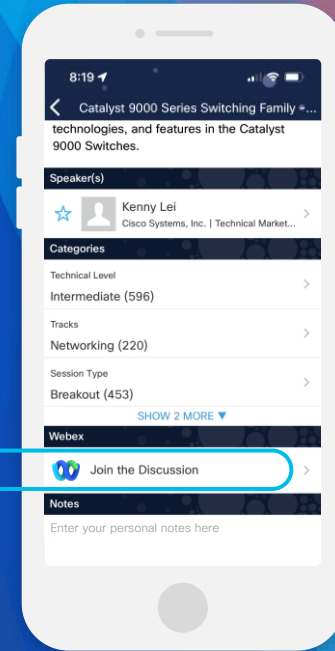
## Questions?

Use Cisco Webex App to chat with the speaker after the session

## How

- 1 Find this session in the Cisco Live Mobile App
- 2 Click “Join the Discussion”
- 3 Install the Webex App or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 9, 2023.



<https://ciscolive.ciscoevents.com/ciscolivebot/#BRKAPP-1154>

# Agenda

- Introduction / Overview
- Core Concepts
  - Signals
  - Data Collection
- Instrumentation
  - Examples
- Embracing OpenTelemetry @ Cisco & Cisco AppDynamics
- Demos
- Conclusion

# What We Will Not Cover

- Deep Dive into Specifications
- Specific Implementations
- Building Custom Artifacts

# Introduction / overview

# What is OpenTelemetry?



# Let's Ask ChatGPT!





WA

Can you explain the concept of OpenTelemetry like I am 5 years old?



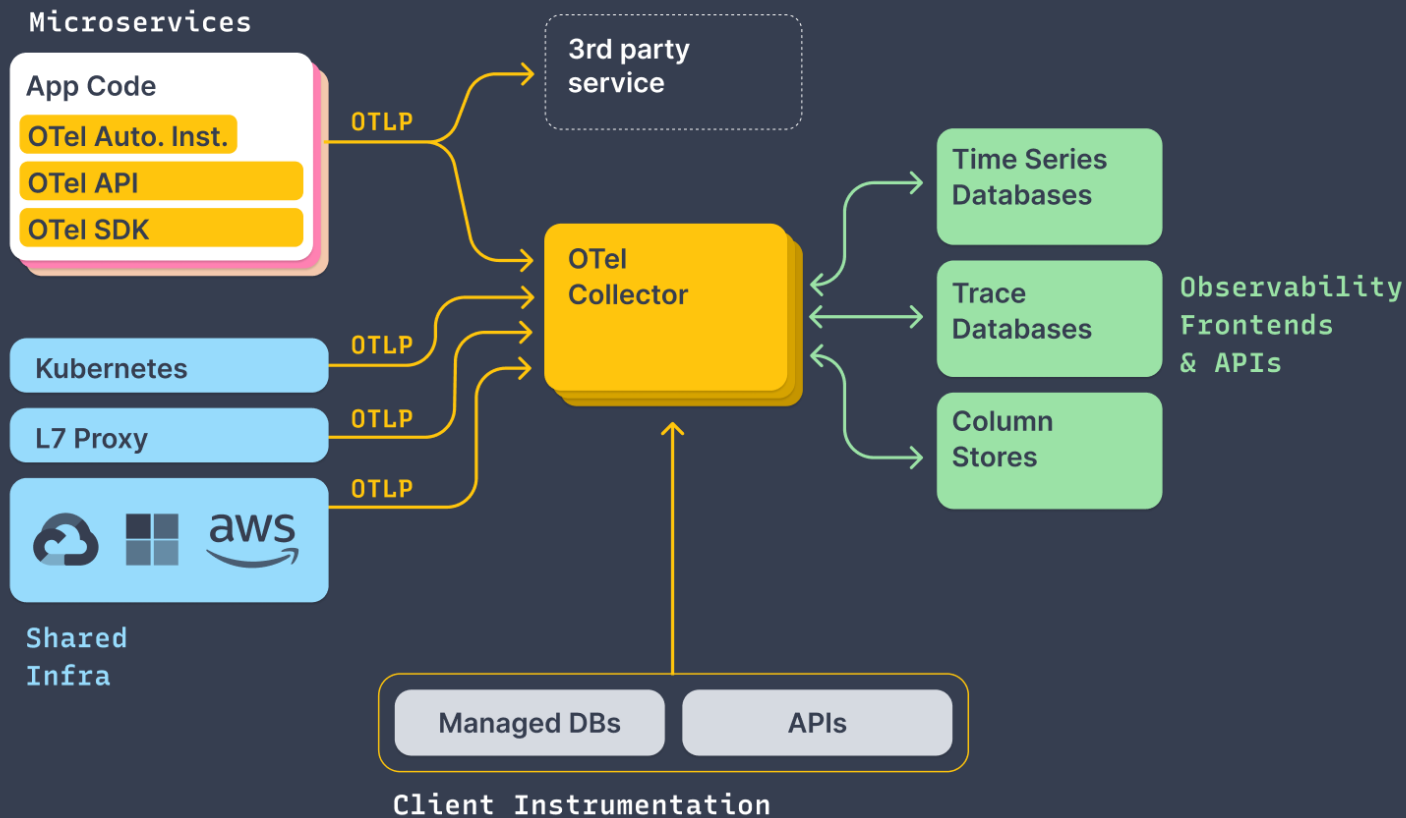
Sure! Imagine you are playing with a bunch of toys in your room. Your mom wants to know which toys you like the most and how you play with them. But she's not always in your room to see you play.

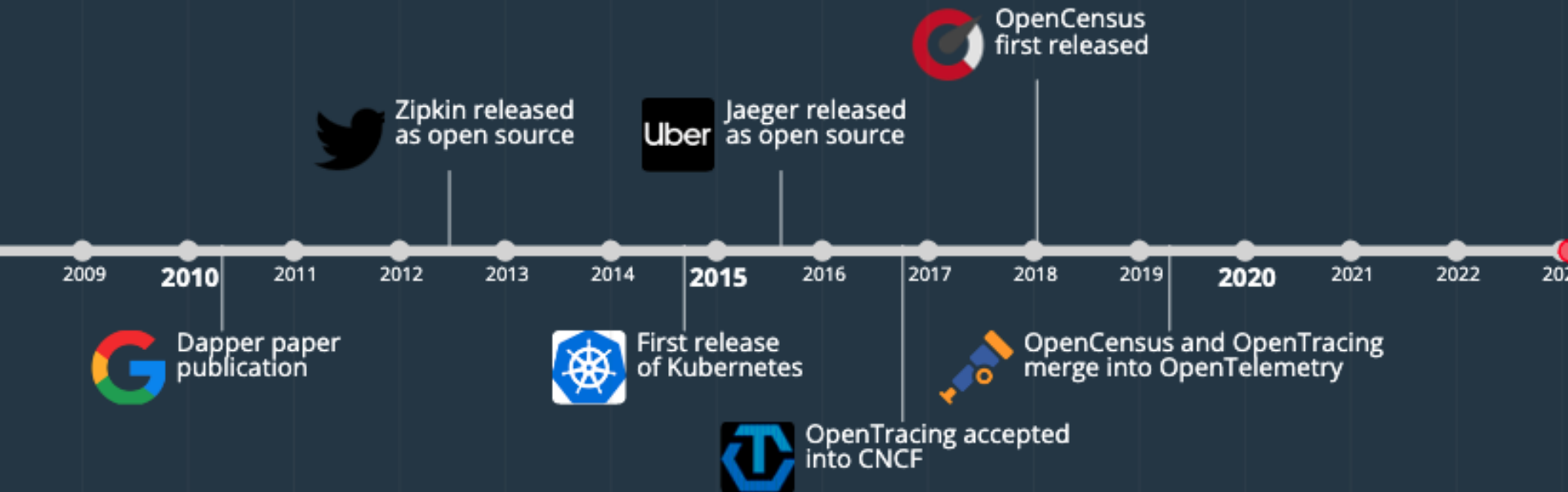


So she asks you to tell her every time you start playing with a toy and when you stop playing with it. This way, she can keep track of which toys you like the most and how you play with them, even when she's not in the room.

OpenTelemetry is like that for computer programs. It helps developers keep track of what their programs are doing, even when they're not watching. It does this by automatically recording information about what the program is doing and sending it to a place where developers can see it.

This helps developers figure out how their programs are working, which parts are slow, and how they can make them better. It's like having a mom who always knows what you're doing with your toys, even when she's not in the room!





# Core concepts



# What is Observability?

*“Observability lets us understand a system from the outside, by letting us ask questions about that system without knowing its inner workings.”*

# *Observability $\neq$ Monitoring*

# Core concepts: signals



# Types of Telemetry Supported by OpenTelemetry



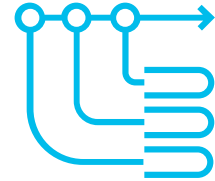
Metrics



Logs



Traces

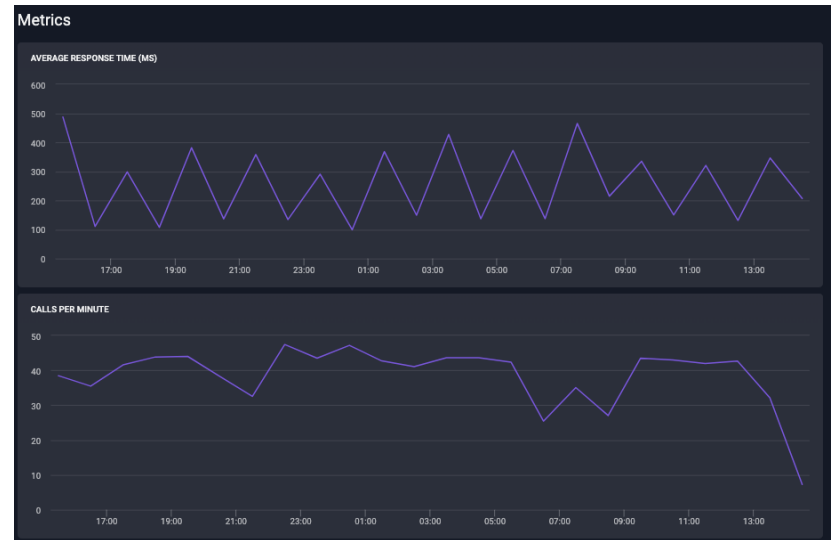


Baggage



# Signals: Metrics

- Measurement about a service captured at runtime
- Three classifications of metrics\*:
  - Counter
  - Observer
  - Measure
- Status: Mostly Stable

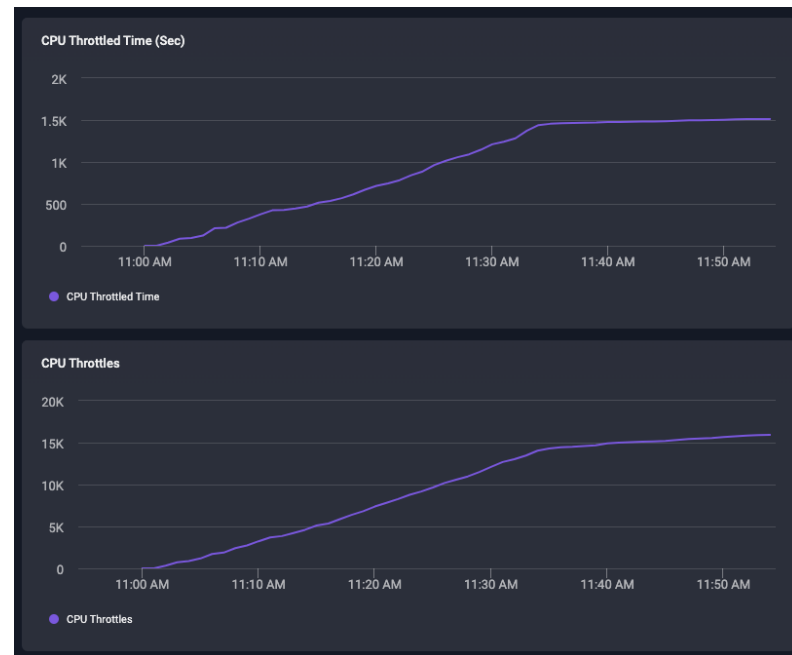




# Signals: Metrics (contd.)

## Counter

- Value that is incremental over time
- Examples:
  - Automobile odometer
  - Service instance uptime
  - CPU Throttled Time





# Signals: Metrics (contd.)

## Observer

- Value / set of values captured at a particular point in time
- Examples:
  - Automobile fuel gauge
  - Memory Usage
  - CPU Utilization

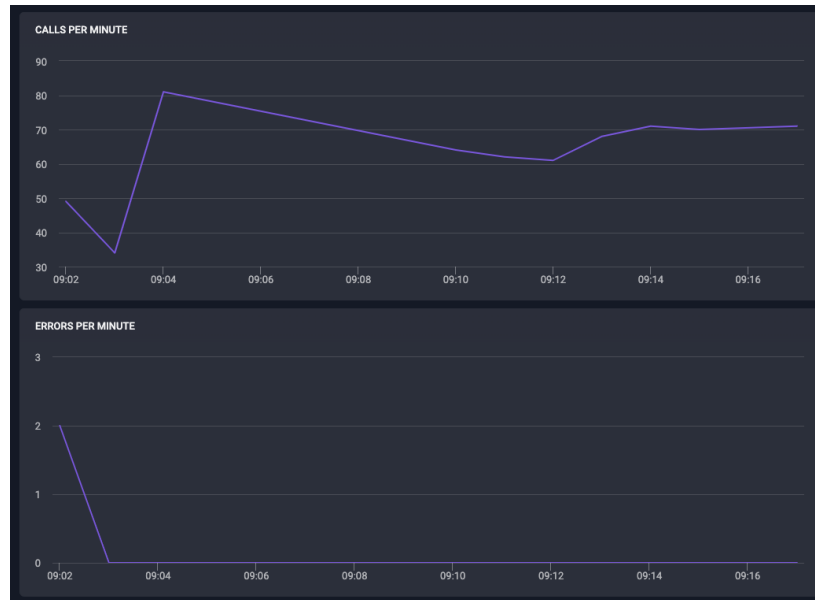




# Signals: Metrics (contd.)

## Measure

- Value that is aggregated over time
- Examples:
  - Automobile trip odometer
  - Average Response Time
  - Error Rate





# Signals: Logs

- Timestamped text record with metadata
- Data that is not part of a distributed trace or a metric is a log.
  - Events – type of log with specific semantic conventions
- Status: Mostly Experimental

Severity	Timestamp ↓	Message
UNKNOWN	1/9/2023, 03:28:40.000	2023-01-09 10:28:40 - c.d.e.c.FulfillmentController - [ /supply [MainSupplierThread] async message sent to getApMainSupplierBaseurl trace_id= span_id= trace_flags=
UNKNOWN	1/9/2023, 03:28:28.000	2023-01-09 10:28:28 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=f7de6c95c1f03860da4
UNKNOWN	1/9/2023, 03:28:17.000	2023-01-09 10:28:17 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=82670be5c777064a93
UNKNOWN	1/9/2023, 03:28:08.000	2023-01-09 10:28:08 - c.d.e.c.FulfillmentController - [ /supply [MainSupplierThread] async message sent to getApMainSupplierBaseurl trace_id= span_id= trace_flags=
UNKNOWN	1/9/2023, 03:27:47.000	2023-01-09 10:27:47 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=13c59a83c03a56f00
UNKNOWN	1/9/2023, 03:27:35.000	2023-01-09 10:27:35 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=88d9f016e7ecd7205
UNKNOWN	1/9/2023, 03:26:58.000	2023-01-09 10:26:58 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=6d848244c991a6c939
UNKNOWN	1/9/2023, 03:26:47.000	2023-01-09 10:26:47 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=28165ac370f5e4bfe79
UNKNOWN	1/9/2023, 03:26:41.000	2023-01-09 10:26:41 - o.s.web.servlet.DispatcherServlet - [ GET "/fulfillment/supply/order/uid-blue&desc=blue", parameters=(masked) trace_id=0d10a01adb0e6c9b45



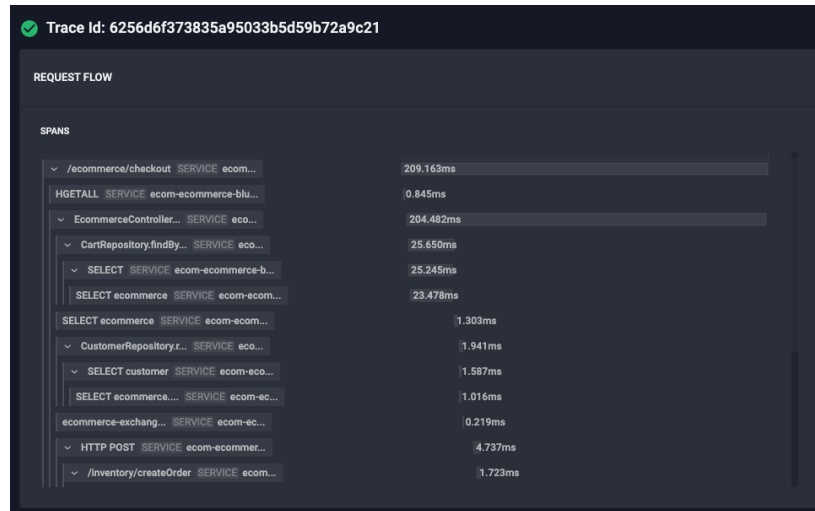
# Log Record Definition

Field Name	Description	Data Type	Mandatory?
Timestamp	Time when the event occurred at the source	uint64 (ns since epoch)	N
ObservedTimestamp	Time when the event was observed	uint64 (ns since epoch)	Y*
TraceId	Request trace id	byte array	N
SpanId	Request span id	byte array	N
TraceFlags	W3C trace flag	byte	N
SeverityText	The severity text (log level)	string	N
SeverityNumber	Numerical value of severity (1-24)	number	N
Body	Body of the log record	any	N
Resource	Describes the source of the log	map<string,any>	N
InstrumentationScope	Describes the scope that emitted the log	(Name,Version) tuple	N*
Attributes	Additional information about the log / event	map<string,any>	N



# Signals: Traces

- Record of paths taken by requests that navigate multi-service architectures
- Helps identify where bottlenecks are occurring.
- Status: Mostly Stable







# Signals: Traces (contd.)

## Fundamental Components of Instrumentation

- Tracer Provider
- Tracer
- Trace Exporter
- Trace Context

```
1  {
2    "resourceSpans": [
3      {
4        "resource": {
5          "attributes": [
6            {
7              "key": "service.name",
8              "value": {
9                "stringValue": "manual-test-trace"
10             }
11           ]
12         },
13       },
14     ],
15     "scopeSpans": [
16       {
17         "scope": {
18           "name": "manual-test"
19         },
20         "spans": [
21           {
22             "traceId": "71699b6fe85982c7c8995ea3d9c95df2",
23             "spanId": "3c191d03fa8be065",
24             "name": "span1",
25             "kind": 2,
26             "droppedAttributesCount": 0,
27             "events": [],
28             "droppedEventsCount": 0,
29             "status": {
30               "code": 1
31             }
32           }
33         ]
34       }
35     ]
36   }
37 }
```



# Signals: Traces (contd.)

## Spans

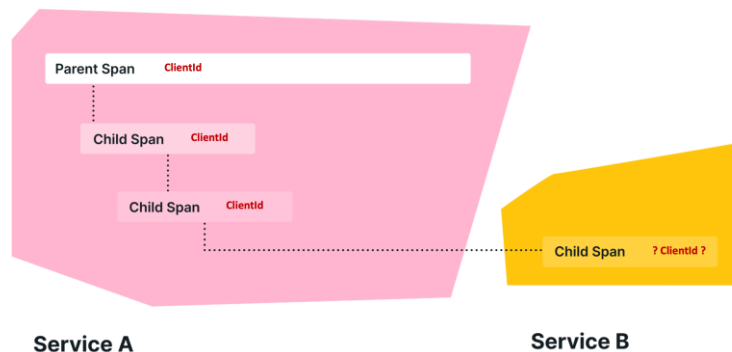
- Individual logical unit of work
- Every trace has one root span
- Spans have 0 or more child spans
- Examples:
  - API call from a microservice
  - Segment in network path

```
1  {
2    "resourceSpans": [
3      {
4        "resource": {
5          "attributes": [
6            {
7              "key": "service.name",
8              "value": {
9                "stringValue": "manual-test-trace"
10             }
11           }
12         ]
13       },
14       "scopeSpans": [
15         {
16           "scope": {
17             "name": "manual-test"
18           },
19           "spans": [
20             {
21               "traceId": "71699b6fe85982c7c8995ea3d9c95df2",
22               "spanId": "3c191d03fa8be065",
23               "name": "span1",
24               "kind": 2,
25               "droppedAttributesCount": 0,
26               "events": [],
27               "droppedEventsCount": 0,
28               "status": {
29                 "code": 1
30               }
31             }
32           ]
33         }
34       ]
35     }
36   ]
37 }
```



# Signals: Baggage

- Contextual information passed between spans
- Key/value pairs stored in HTTP headers
- Status: Stable



Monitor  
(Detect)

Metrics  
“What is Happening?”

Explore  
(Troubleshoot)

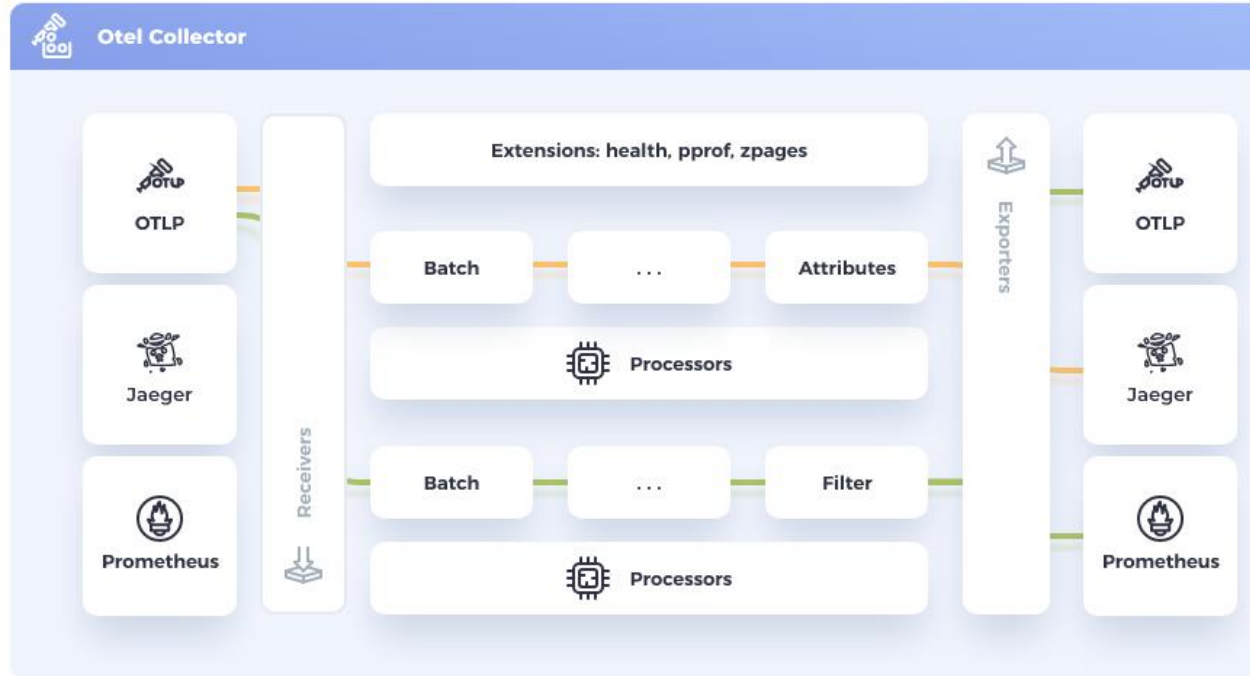
Traces & Events  
“Where is it  
Happening?”

Discover  
(Root Cause)

Logs  
“Why is it Happening?”

# Core concepts: Data collection

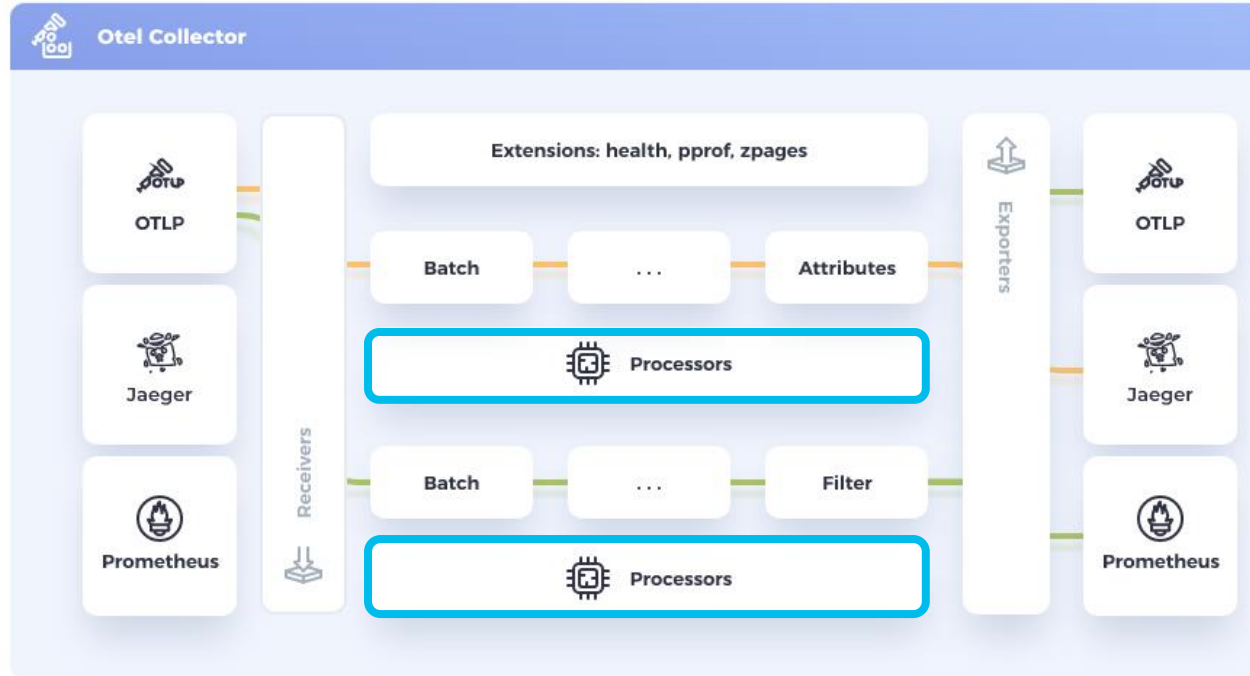
# OpenTelemetry Collector Architecture



# OpenTelemetry Collector Architecture

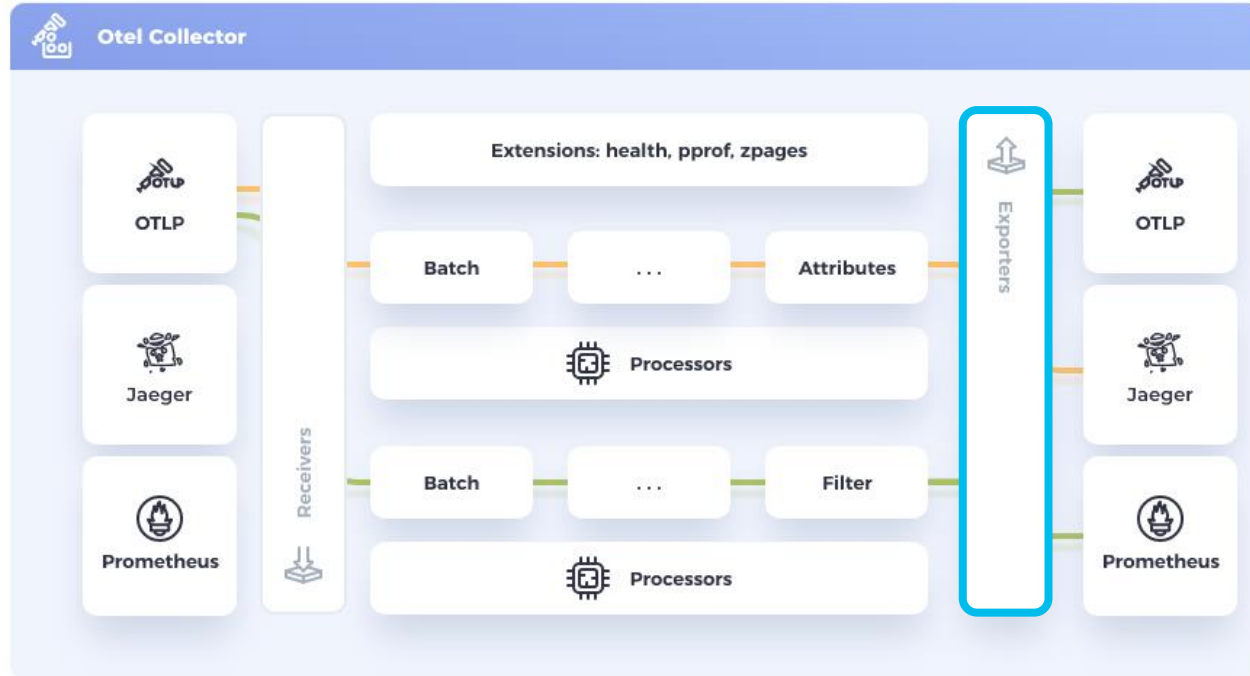


# OpenTelemetry Collector Architecture

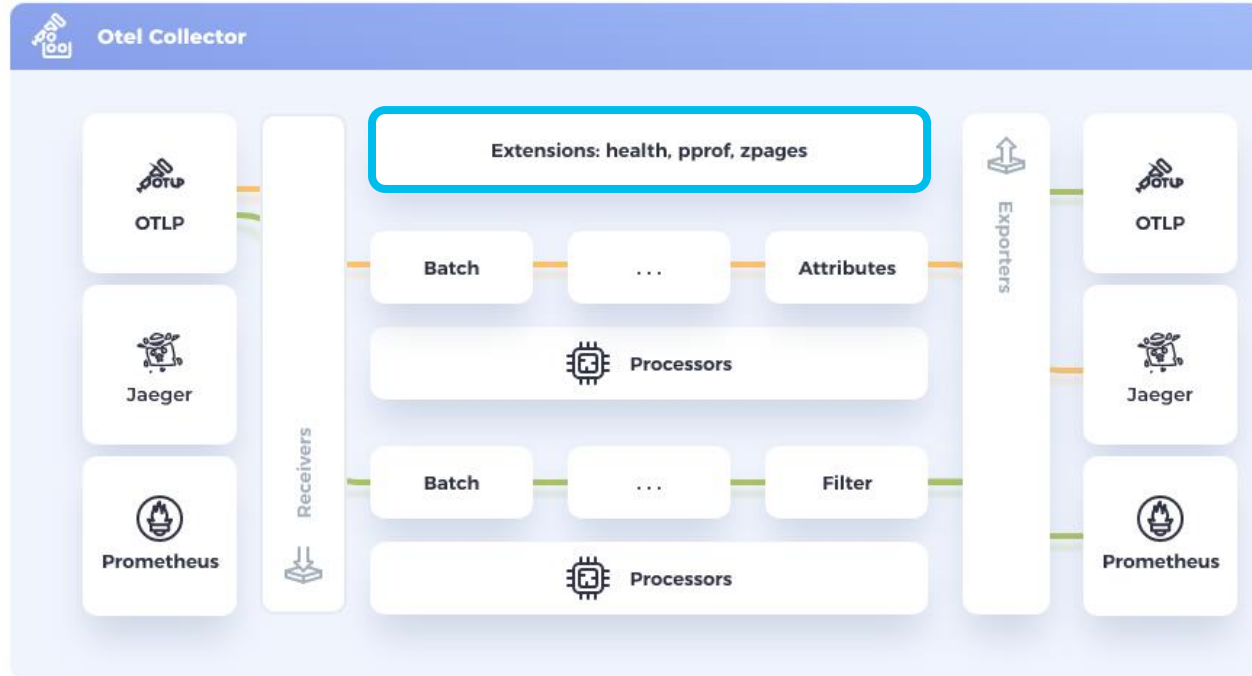




# OpenTelemetry Collector Architecture



# OpenTelemetry Collector Architecture



# OpenTelemetry Collector Configuration

- Component configuration defined in YAML sections
- Service section defines pipelines for telemetry data
- Order matters
- Authentication available as extensions

```
receivers:
  otlp:
    protocols:
      grpc:
      http:
    otlp/2:
      protocols:
        grpc:
          endpoint: 0.0.0.0:55690

processors:
  batch:
  batch/test:

exporters:
  otlp:
    endpoint: otelcol:4317
  otlp/2:
    endpoint: otelcol2:4317

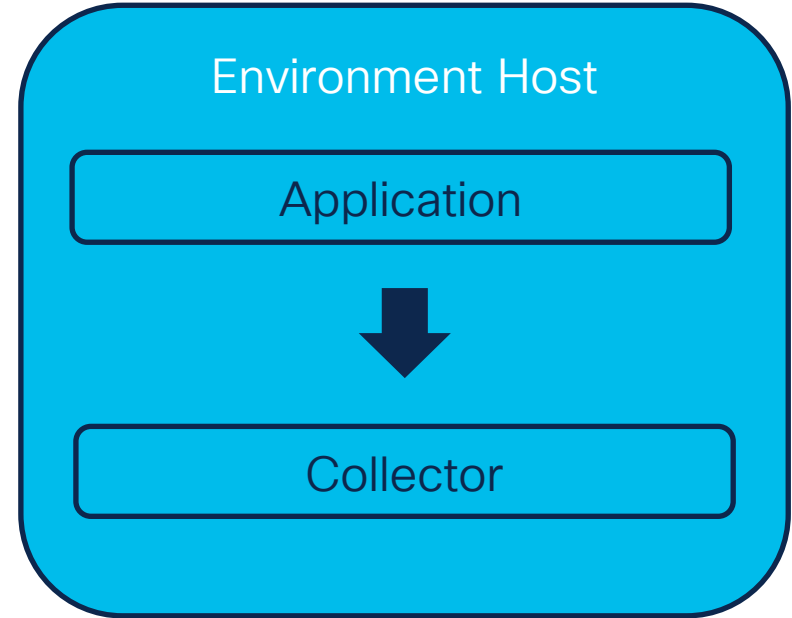
extensions:
  health_check:
  pprof:
  zpages:

service:
  extensions: [health_check, pprof, zpages]
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp]
    traces/2:
      receivers: [otlp/2]
      processors: [batch/test]
      exporters: [otlp/2]
    metrics:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp]
    logs:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp]
```

# Collector Deployment Models

## Agent-Based

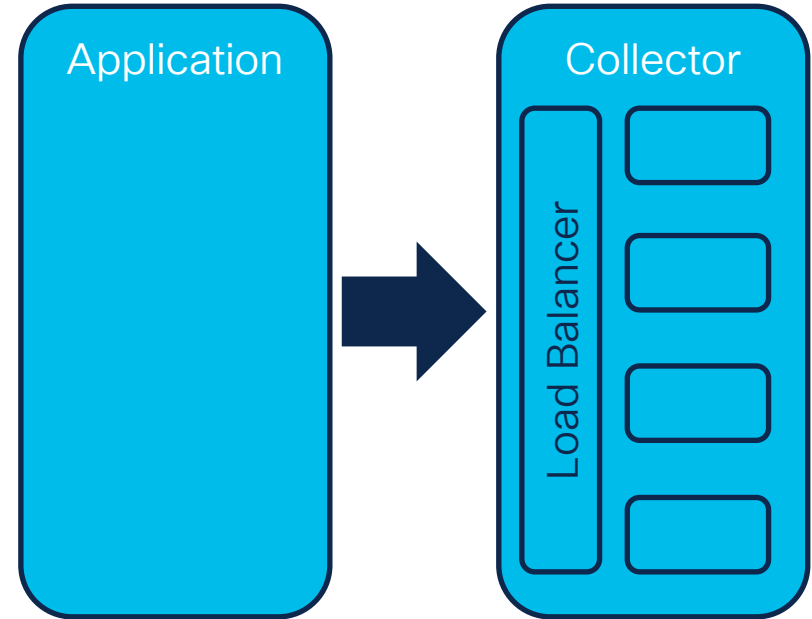
- Runs within the application or on the same host as the application
  - Examples: sidecar, daemonset, AWS Lambda Layer
- Offloads responsibilities from client instrumentation



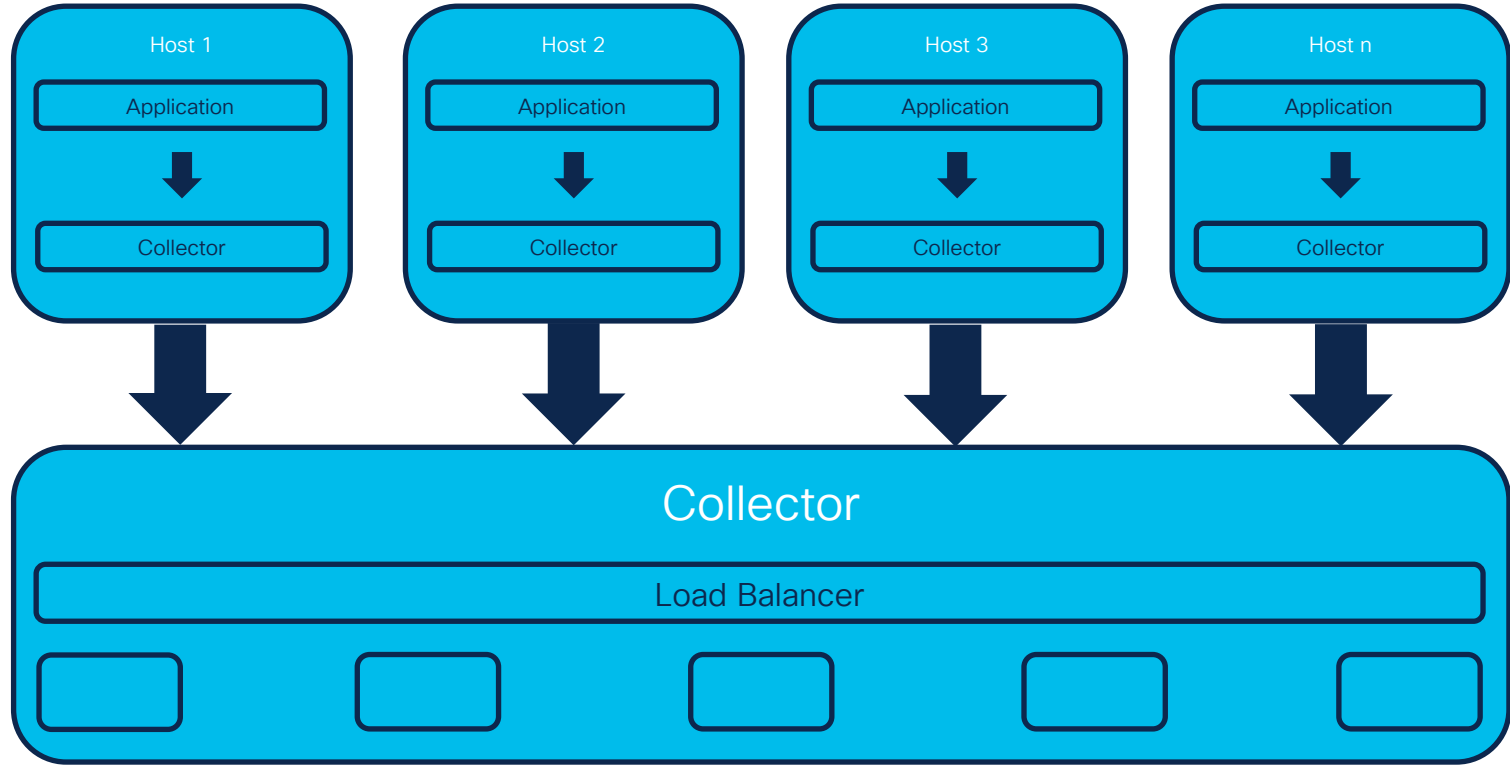
# Collector Deployment Models

## Gateway-Based

- Composed of 1+ instances
- Runs as a standalone service
- Offers advantages / advanced capabilities
  - Limit egress points
  - API token management



# Sample Pipeline Architecture



# Instrumentation

# Instrumentation

- How we get applications / services to emit traces / logs / metrics
- Automatic or manual
- Support for most popular languages





# Instrumentation Status Grid

✓ Stable

⚠ Experimental

☹ Mixed

α Alpha

⊘ Not yet implemented

⚙ In Development

Language	Traces	Metrics	Logs
C++	✓	✓	⚠
C# / .NET	✓	✓	☹
Erlang / Elixir	✓	⚠	⚠
Go	✓	α	⊘
Java	✓	✓	⚠
JavaScript	✓	✓	⚙

Source: <https://opentelemetry.io/docs/instrumentation/>

# Instrumentation Support Grid (contd.)

✓ Stable

⚠ Experimental

β Beta

α Alpha

⊘ Not yet implemented

⚙ In Development

Language	Traces	Metrics	Logs
PHP	β	β	⊘
Python	✓	✓	⚠
Ruby	✓	⊘	⊘
Rust	✓	α	⊘
Swift	✓	⚠	⚙

Source: <https://opentelemetry.io/docs/instrumentation/>

# Why Use Manual Instrumentation?

- Language doesn't support automatic instrumentation
- More options available for customization
  - Adding custom attributes / data
  - Customized span creation



# Why Use Automatic Instrumentation?

- Does not require code changes
- Quicker time to implementation
- Also allows for manual instrumentation when needed



# Instrumentation: Examples

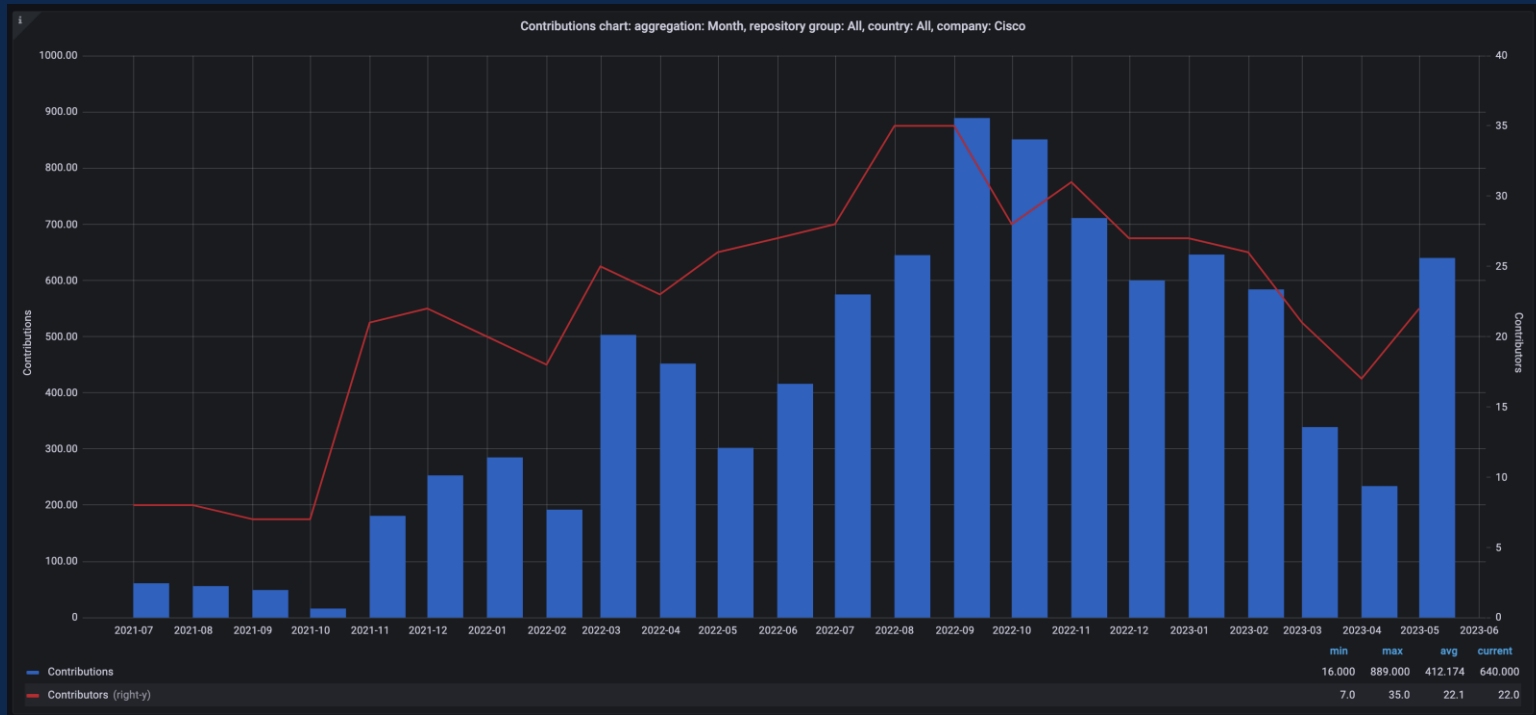
# Official OpenTelemetry Demo

- URL: <https://cs.co/otel-demo>



# Embracing OpenTelemetry @ Cisco and Cisco AppDynamics

# Our Contribution to OpenTelemetry



Source: <https://opentelemetry.devstats.cncf.io/d/74/contributions-chart>

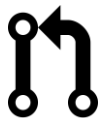


# Our Contribution to OpenTelemetry (contd.)

## All-Time Rankings



Overall Contributions: #11



Pull Requests: #9



Contributors: #4

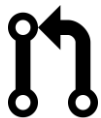
\* Rankings as of May 31, 2023. Source: <https://opentelemetry.devstats.cncf.io/d/5/companies-table?orgId=1>

# Our Contribution to OpenTelemetry (contd.)

Rankings Last 12 mo.



Overall Contributions: #8



Pull Requests: #5



Contributors: #3

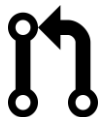
\* Rankings as of May 31, 2023. Source: <https://opentelemetry.devstats.cncf.io/d/5/companies-table?orgId=1>

# Our Contribution to OpenTelemetry (contd.)

Rankings Last 3 mo.



Overall Contributions: #9



Pull Requests: #7



Contributors: #1

\* Rankings as of May 31, 2023. Source: <https://opentelemetry.devstats.cncf.io/d/5/companies-table?orgId=1>

# Our Contribution to OpenTelemetry (contd.)

## Presence / Contributions



16 Established Community Members



8 Approvers



1 Maintainer

## • Project Contributions

- Authenticator for OAuth2
- Apache httpd instrumentation
- Nginx httpd instrumentation
- JMX Metric Insight
- Container resource detection for Java / JavaScript / .NET / Go
- HTTP data collectors for Python
- SQL instrumentation for Go
- Auto Instrumentation for Node JS

# Demos

# Conclusion

# Additional Resources

- OpenTelemetry Documentation – <https://docs.opentelemetry.io>
- o11y news – <https://o11y.news/>
- [Advanced] Building Your Own Collector --  
<https://github.com/rquedas/otel4devs/tree/main/collector>
- OpenTelemetry Demo Application – <https://cs.co/otel-demo>

# Fill out your session surveys!



Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!



Attendees will also earn 100 points in the **Cisco Live Game** for every survey completed.



**These points** help you get on the leaderboard and increase your chances of winning daily and grand prizes



# Continue your education



- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at [www.CiscoLive.com/on-demand](https://www.CiscoLive.com/on-demand)

# Related Sessions

Monday

5

Time: 9:30 AM

Title: Do Tell About OTel: An Introduction to OpenTelemetry and How AppDynamics is Embracing It  
Session Code: BRKAPP-1154

1

Time: 11:00 AM

Title: Monitor and Troubleshoot Applications in AWS and Azure with Cloud Native Application Observability  
Session Code: BRKAPP-1008

Time: 3:30 PM

Title: New AppDynamics Cloud and Security Innovations  
Session Code: PSOAPP-1785

Tuesday

6

Time: 10:30 AM

Title: Interactive Guide to Unlocking Observability with OpenTelemetry  
Session Code: IBOAPP-2980

2

Time: 1:00 PM

Title: Open Source Grafana monitoring for AppDynamics Cloud  
Session Code: BRKAPP-1510

Time: 2:30 PM

Title: Empower a new observability ecosystem with an open and extensible Cisco FSO Platform  
Session Code: BRKAPP-1013

Wednesday

7

Time: 10:30 AM

Title: Cross MELT troubleshooting with AppDynamics Cloud and FSO Platform  
Session Code: BRKAPP-2915

3

Time: 1:00 PM

Title: Cisco FSO Platform Ecosystem for Developers  
Session Code: BRKAPP-2006

Time: 2:30 PM

Title: Royal Caribbean elevates customer experience with NTT 360 Observability service powered by Cisco FSO  
Session Code: BRKAPP-2010

Thursday

8

Time: 8:00 AM

Title: New AppDynamics Cloud Efficiency and Cost Optimization Capability for Kubernetes Workloads  
Session Code: BRKAPP-1916

Time: 10:30 AM

Title: The Journey of migrating Cisco IT's AIOps solution to the Cisco FSO platform  
Session Code: BRKAPP-2104

Time: 1:00 PM

Title: From Monolith to Microservices: How to plan application migration with observability in mind  
Session Code: BRKAPP-2499

4

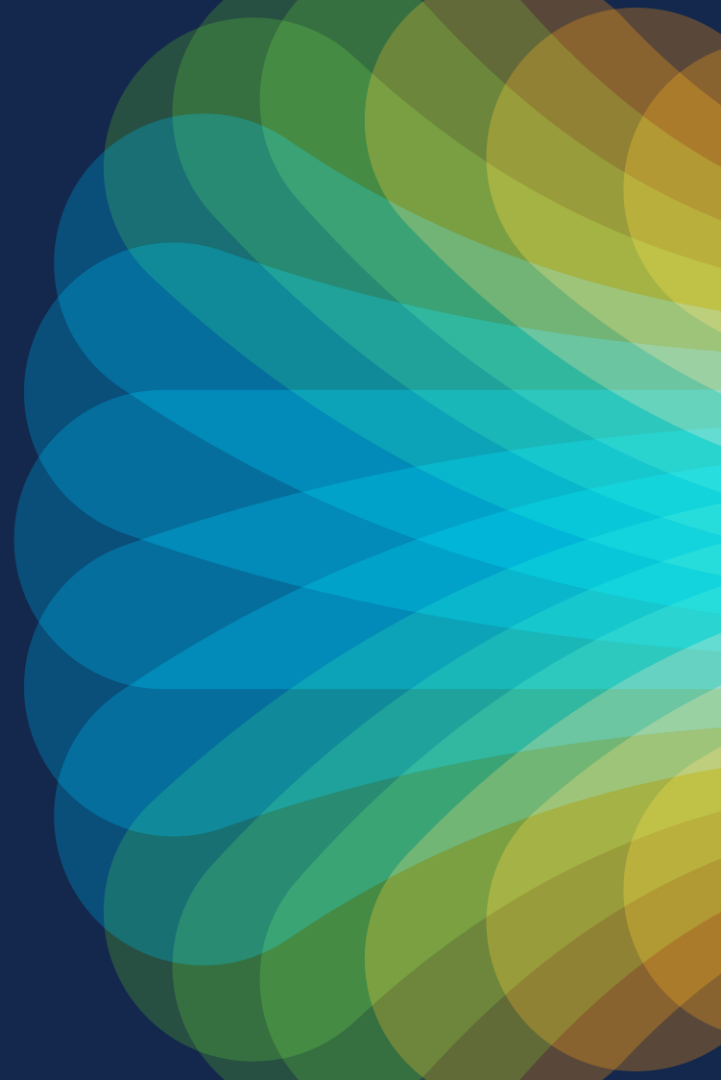


The bridge to possible

# Thank you



#CiscoLive

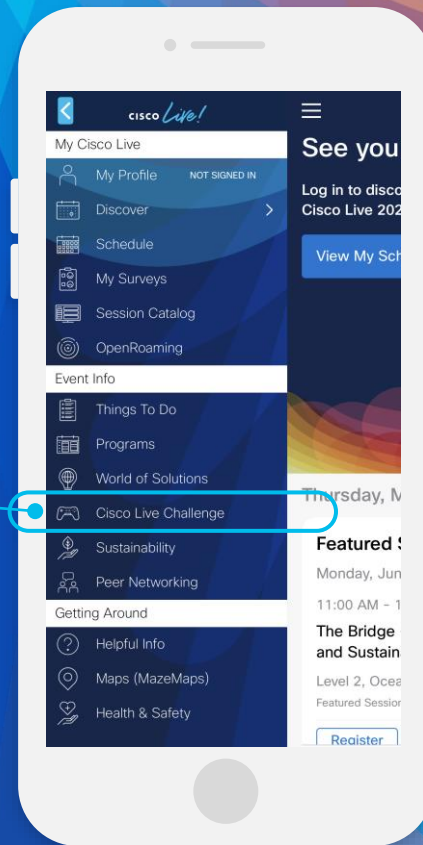


# Cisco Live Challenge

Gamify your Cisco Live experience!  
Get points for attending this session!

## How:

- 1 Open the Cisco Events App.
- 2 Click on 'Cisco Live Challenge' in the side menu.
- 3 Click on View Your Badges at the top.
- 4 Click the + at the bottom of the screen and scan the QR code:



The background is a vibrant, abstract graphic. It features a central bright white light source from which numerous colorful rays emanate, creating a sunburst or starburst effect. The rays transition through a spectrum of colors including yellow, orange, red, and various shades of blue and green. Overlaid on this are several large, semi-transparent, wavy shapes in similar color tones, giving the overall image a sense of motion and energy.

cisco *Live!*

Let's go

#CiscoLive