OpenTelemetry Is expanding into CI/CD observability

Posted on November 4, 2024 by Dotan Horovits + Adriel Perkins

CNCF projects highlighte

OpenTelemetry

Open Telemetry

Is Eiffel really the future?

*SIG post by **Dotan Horovits** and **Adriel Perkins**, Project Leads, SIG CI/CD Observability, OpenTelemetry*

We've been talking about the need for a common "language" for reporting and observing CI/CD pipelines for years, and finally, we see the first "words" of this language entering the "dictionary" of observability – the OpenTelemetry open specification. With the recent release of OpenTelemetry's Semantic Conventions, v1.27.0, you can find **designated attributes for reporting CI/CD pipelines**.

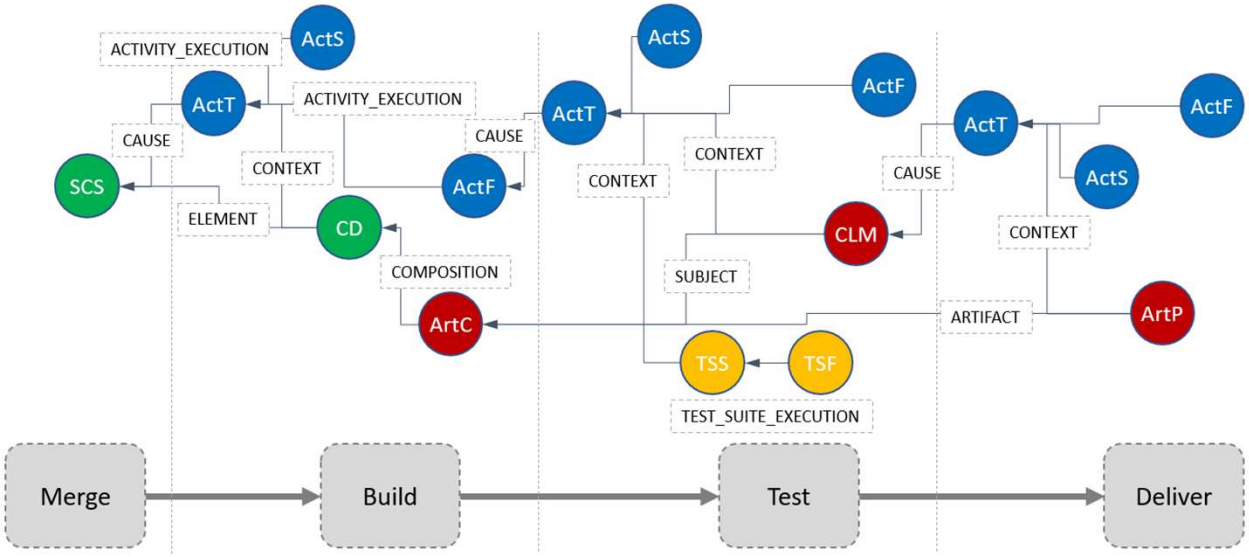
This is the result of the hard work of the **CI/CD Observability Special Interest Group (SIG) within OpenTelemetry**. As we accomplish the core milestone for the first phase, we thought it'd be a good time to share it with the world.

Is Eiffel Really the future?

Short introduction to OTEL traces

Open discussion

Eiffel



- Our trusted Eiffel we use for:
  - Observability in CI/CD chains
  - Traceability
  - Interoperability
  - Confidence levels
- Community of mostly Nordic countries

OTEL - **OpenTelemetry**

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

OpenTelemetry is a Cloud Native Computing Foundation (CNCF) project that is the result of a merger between two prior projects, OpenTracing and OpenCensus.

- An **observability framework and toolkit** designed to facilitate the
  - Generation
  - Export
  - Collection

of telemetry data such as traces, metrics, and logs.

OpenTelemetry made its **first commit to GitHub** in April 2019. Since **joining CNCF** on May 17, 2019, OpenTelemetry has added:

9,160+
Contributors

55,640+
Code commits

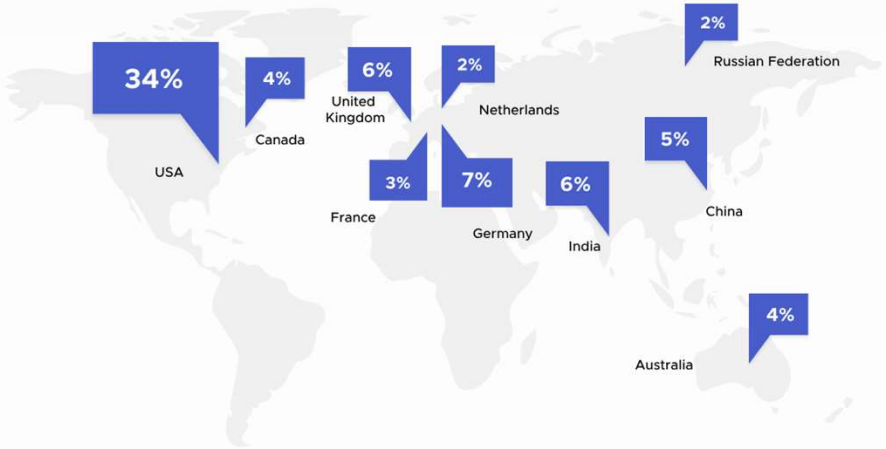
67,250+
Pull requests

466,000+
Contributions

1,100+
Contributing companies

GEOGRAPHIC DIVERSITY OF CONTRIBUTORS

TOP CONTRIBUTING COUNTRIES



By the numbers:

1,106 ↑
COMPANIES CONTRIBUTING CODE

13,650%
increase since project inception (from 8 to more than 1,106)

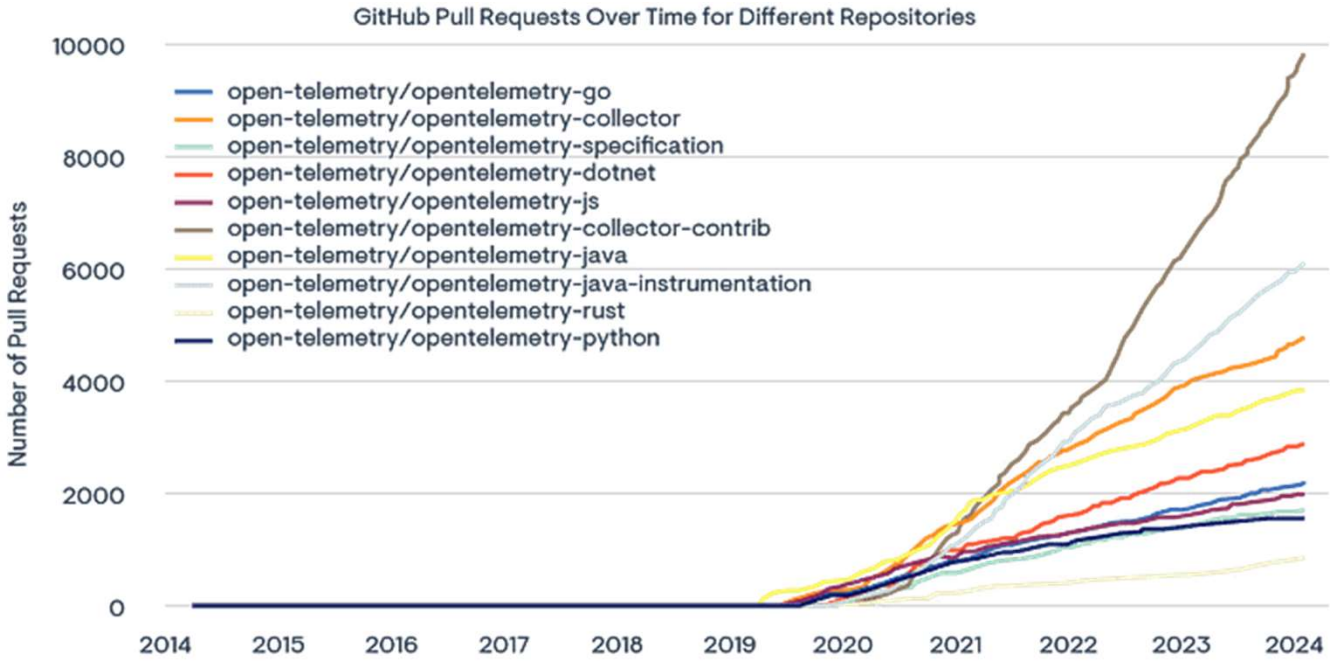
TOP 3 ↑
CONTRIBUTING ORGANIZATIONS OVERALL

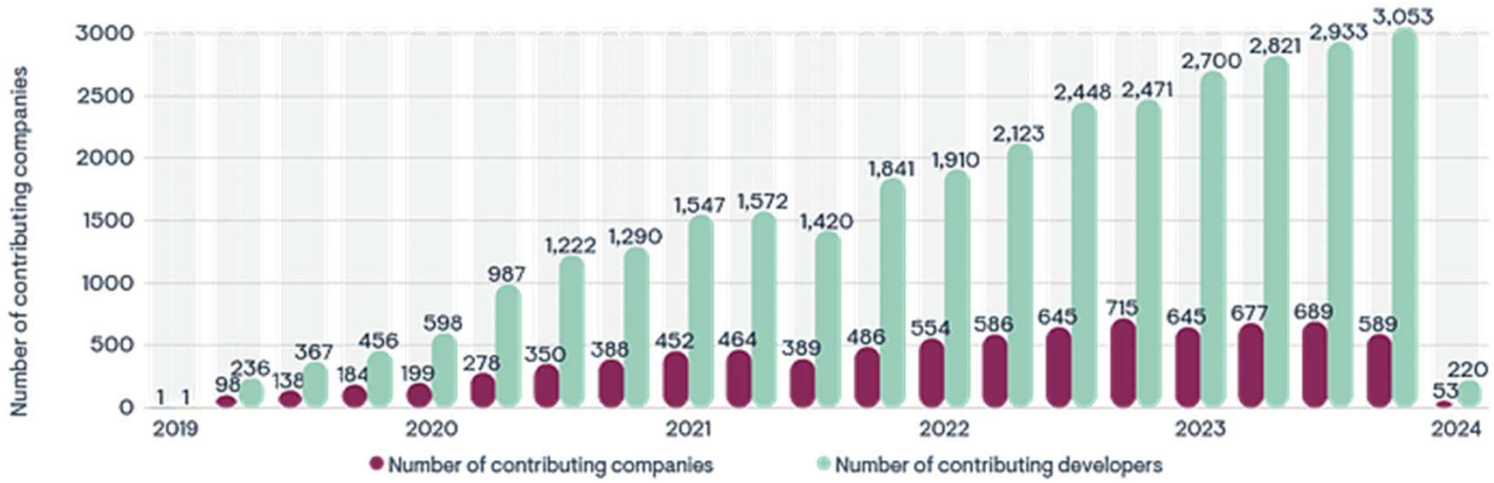
27% SPLUNK
17% MICROSOFT
8% LIGHTSTEP

9,168 ↑
INDIVIDUALS CONTRIBUTING CODE

53,829%
increase since project inception (from 17 to 9,168)

https://www.cncf.io/reports/opentelemetry-project-journey-report/


GitHub Pull Requests Over Time for Different Repositories
Number of Pull Requests
10000
8000
6000
4000
2000
0
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
open-telemetry/opentelemetry-go
open-telemetry/opentelemetry-collector
open-telemetry/opentelemetry-specification
open-telemetry/opentelemetry-dotnet
open-telemetry/opentelemetry-js
open-telemetry/opentelemetry-collector-contrib
open-telemetry/opentelemetry-java
open-telemetry/opentelemetry-java-instrumentation
open-telemetry/opentelemetry-rust
open-telemetry/opentelemetry-python


Number of contributing companies
3000
2500
2000
1500
1000
500
0
2019
2020
2021
2022
2023
2024
1 1
98 236
138 367
184 456
199 598
278 987
350 1,222
388 1,290
452 1,547
464 1,572
389 1,420
486 1,841
554 1,910
586 2,123
645 2,448
715 2,471
645 2,700
677 2,821
689 2,933
589 3,053
53 220
Number of contributing companies
Number of contributing developers

OpenTelemetry is set to become the backbone of the open source monitoring ecosystem. Because these are open standards, and open source SDKs and tools, OpenTelemetry enables vendor-neutral solutions. These solutions can be composed from the best tools in the industry, such as Prometheus. OpenTelemetry is a great example of a successful, community-driven open source project.

Myrle Krantz
Department Head, Application Observability, Grafana Labs

OpenTelemetry is establishing itself as the industry standard to provide our customers the means to collect, process, and ingest traces, metrics, and (soon) logs in a vendor-neutral fashion. We see a lot of interest in OpenTelemetry and work upstream as well as in the service team to overcome the adoption hurdles.

Michael Hausenblas
Solution Engineering Lead at AWS, OpenTelemetry Maintainer

https://www.cncf.io/reports/opentelemetry-project-journey-report/

What is new?

OpenTelemetry Is expanding into CI/CD observability

Posted on November 4, 2024 by Dotan Horovits + Adriel Perkins

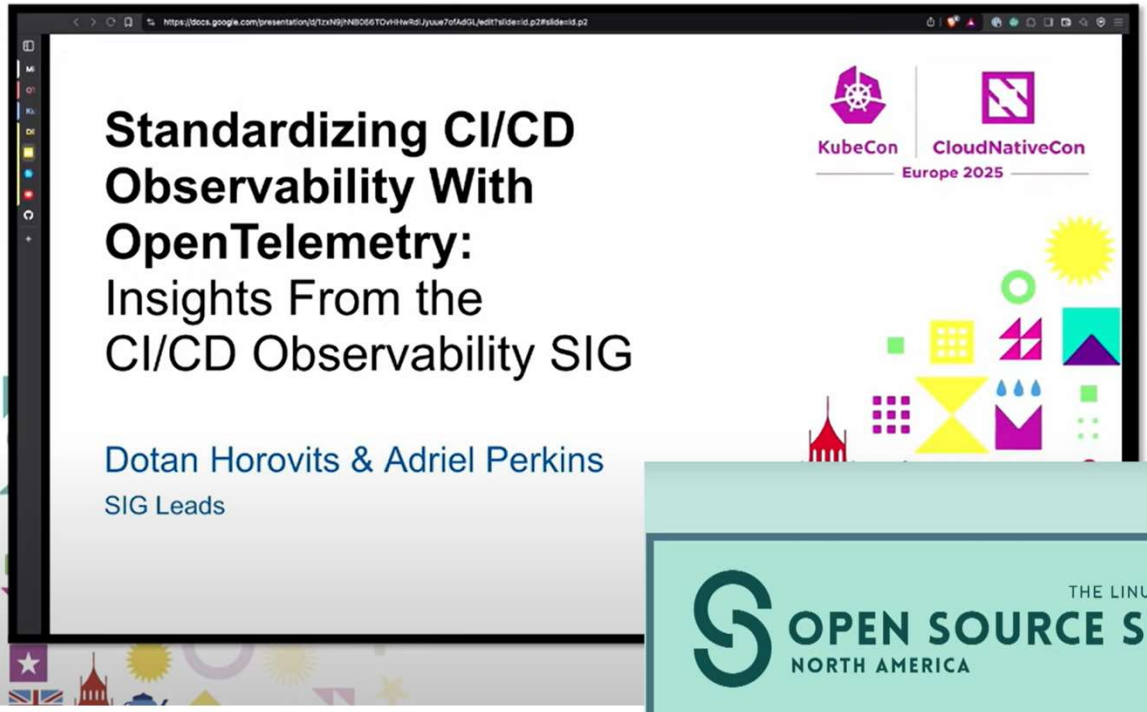
CNCF projects highlighted in this post

*SIG post by **Dotan Horovits** and **Adriel Perkins**, Project Leads, SIG CI/CD Observability, OpenTelemetry*

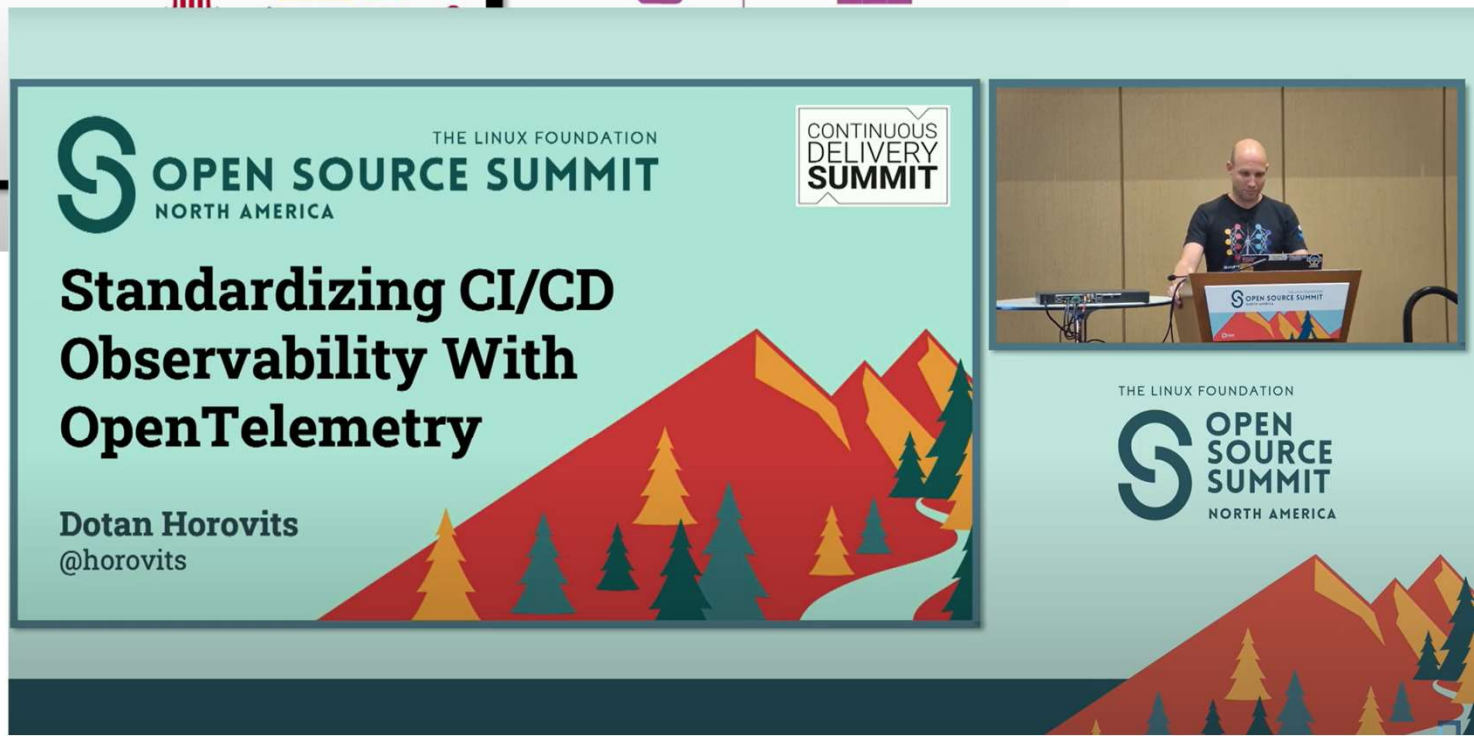
We've been talking about the need for a common "language" for reporting and observing CI/CD pipelines for years, and finally, we see the first "words" of this language entering the "dictionary" of observability – the OpenTelemetry open specification. With the recent release of OpenTelemetry's Semantic Conventions, v1.27.0, you can find **designated attributes for reporting CI/CD pipelines**.

This is the result of the hard work of the **CI/CD Observability Special Interest Group (SIG) within OpenTelemetry**. As we accomplish the core milestone for the first phase, we thought it'd be a good time to share it with the world.

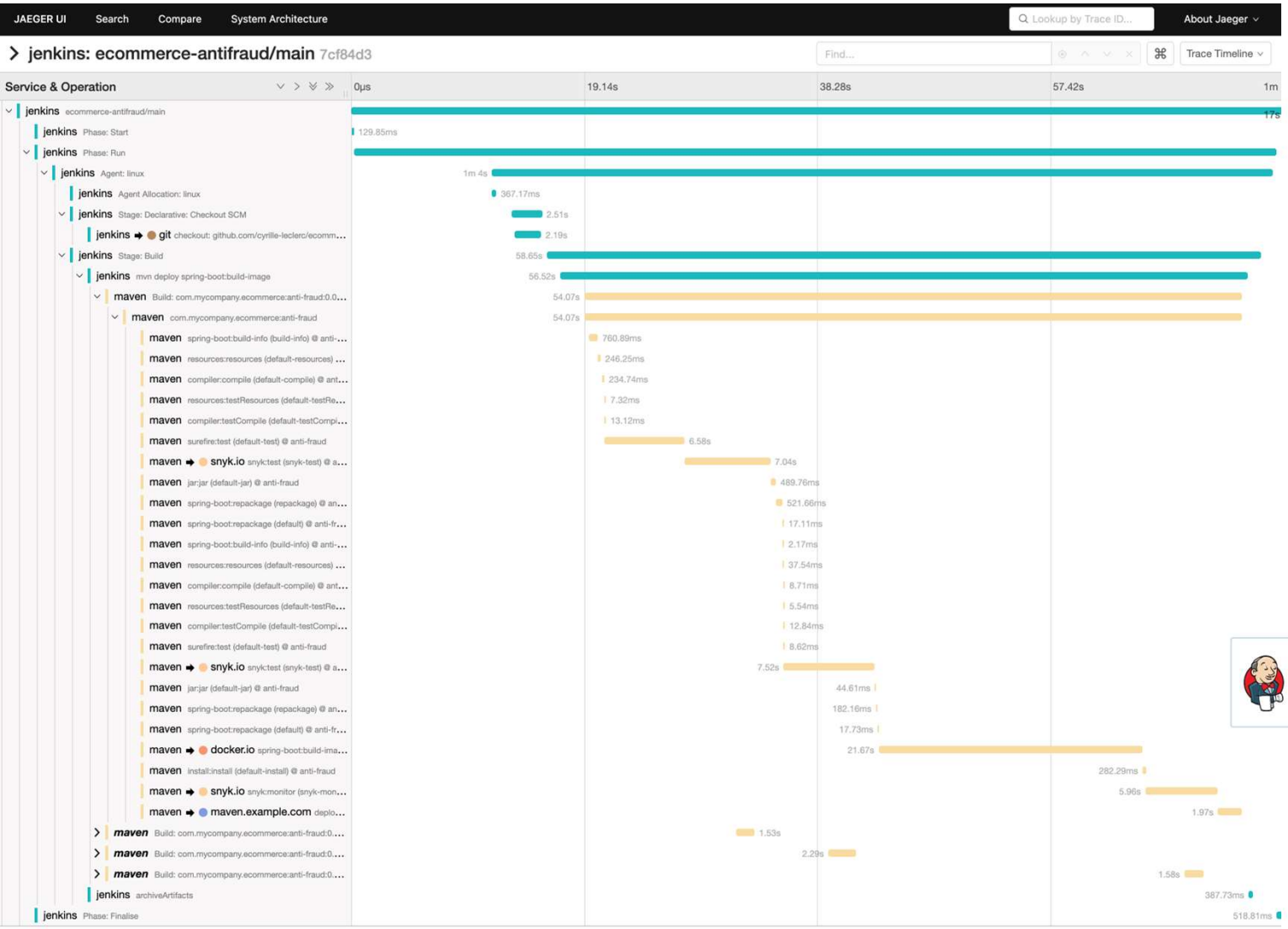
https://www.cncf.io/blog/2024/11/04/opentelemetry-is-expanding-into-ci-cd-observability/

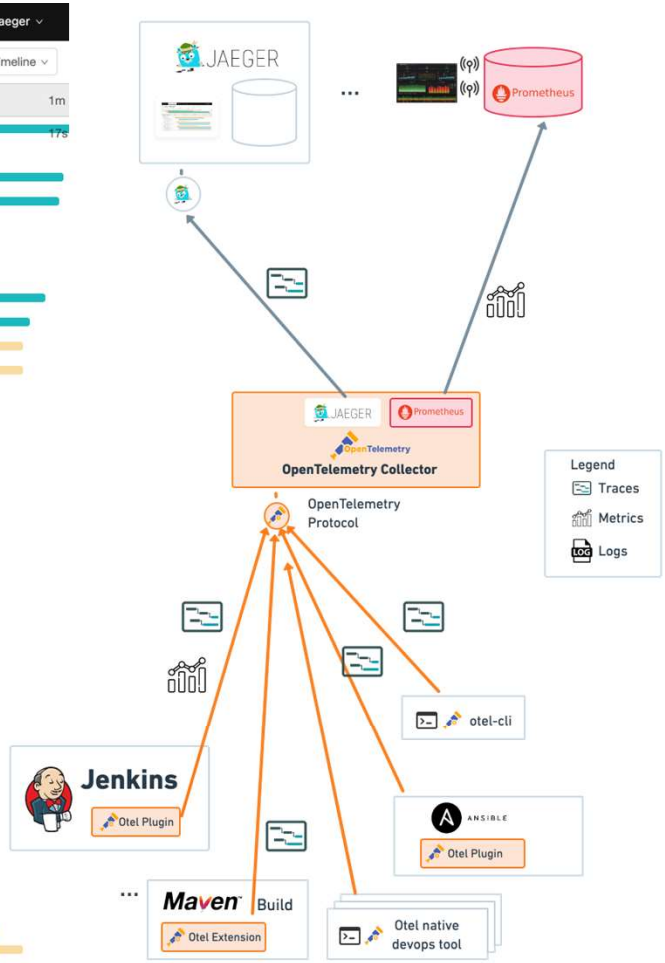


https://www.youtube.com/watch?v=IvIgsHS5MDk



https://www.youtube.com/watch?v=GsqxkoYa9Qc





https://plugins.jenkins.io/opentelemetry/

Distributed traces > d6ae0fb7c9ef91d9062d205a62c1df7b

'BUILD Dynatrace' Trace

Start time: 2024 August 2 11:55:00

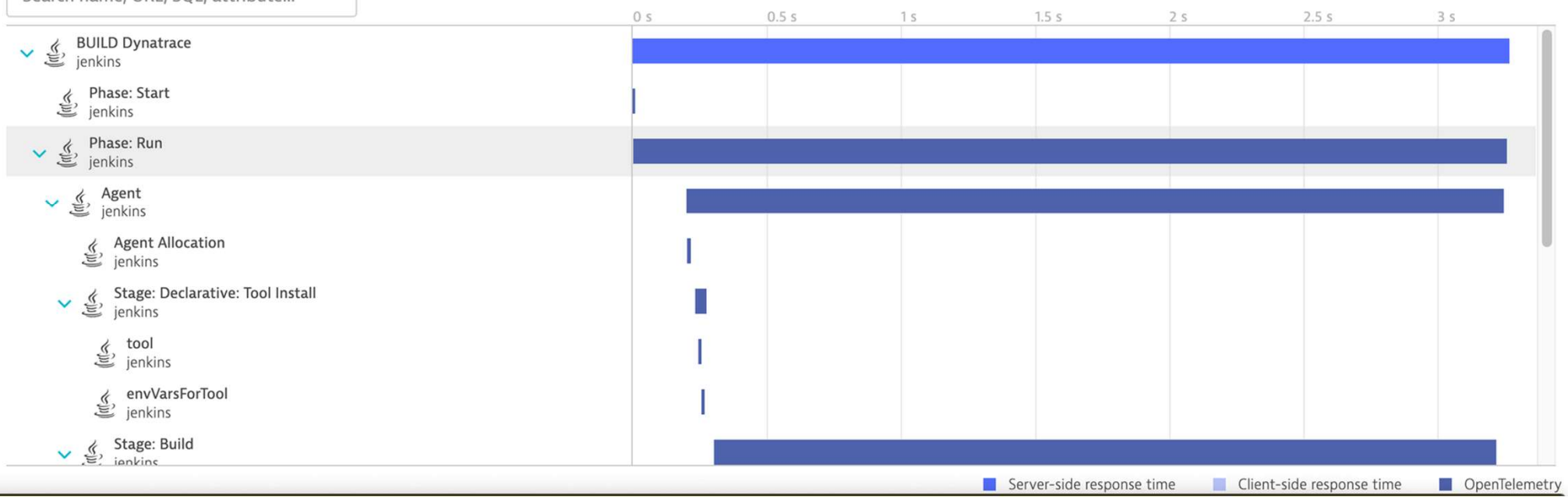
View logs for this trace | Refine analysis | ···

3.27s Response time

Attributes

ci.pipeline.id: Dynatr... | ci.pipeline.run.committe | ci.pipeline.run.cause: U. | ci.pipeline.run.complete | ci.pipeline.run.url: htt... | 6 more...

Search name, URL, SQL, attribute...



https://community.dynatrace.com/t5/Troubleshooting/Jenkins-Pipeline-Analytics-using-OpenTelemetry-and-Dynatrace/ta-p/252223

GitLab instrumentation for OpenTelemetry

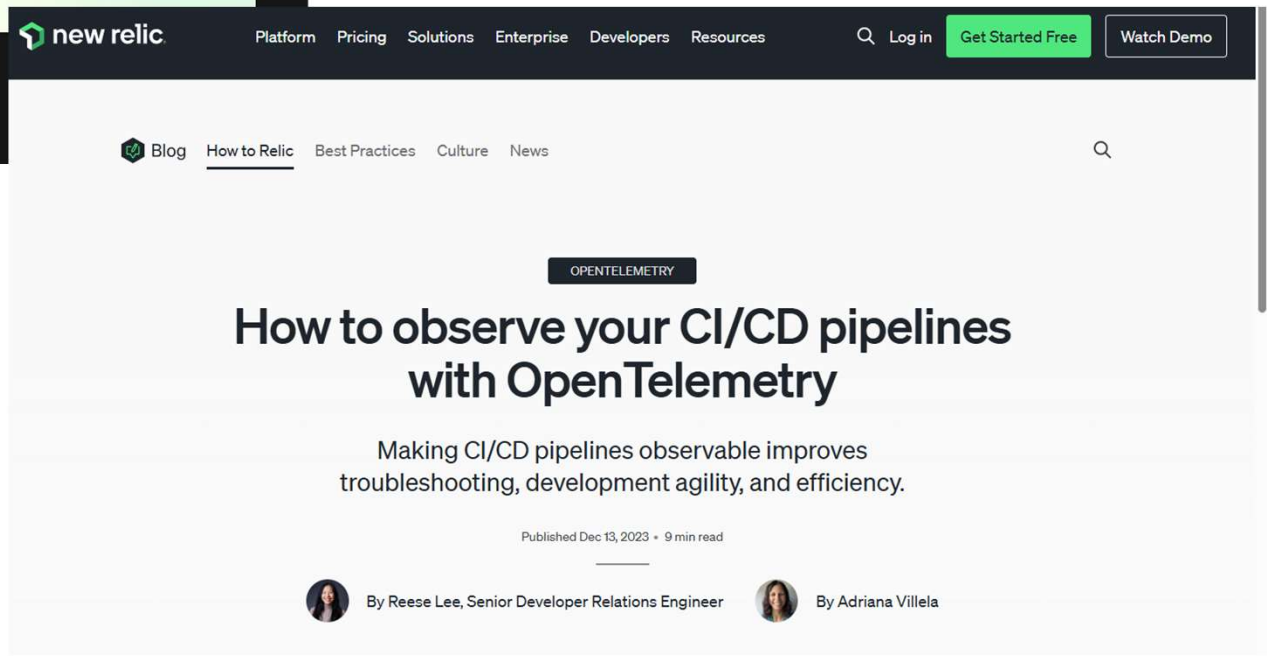
Contribute

Enable OpenTelemetry tracing, metrics, and logs in GDK development

Currently the default GDK environment is not set up by default to properly collect and display OpenTelemetry data. Therefore, you should point the `OTEL_EXPORTER_*_ENDPOINT` ENV vars to a GitLab project:

https://docs.gitlab.com/development/stage_group_observability/gitlab_instrumentation_for_opentelemetry/

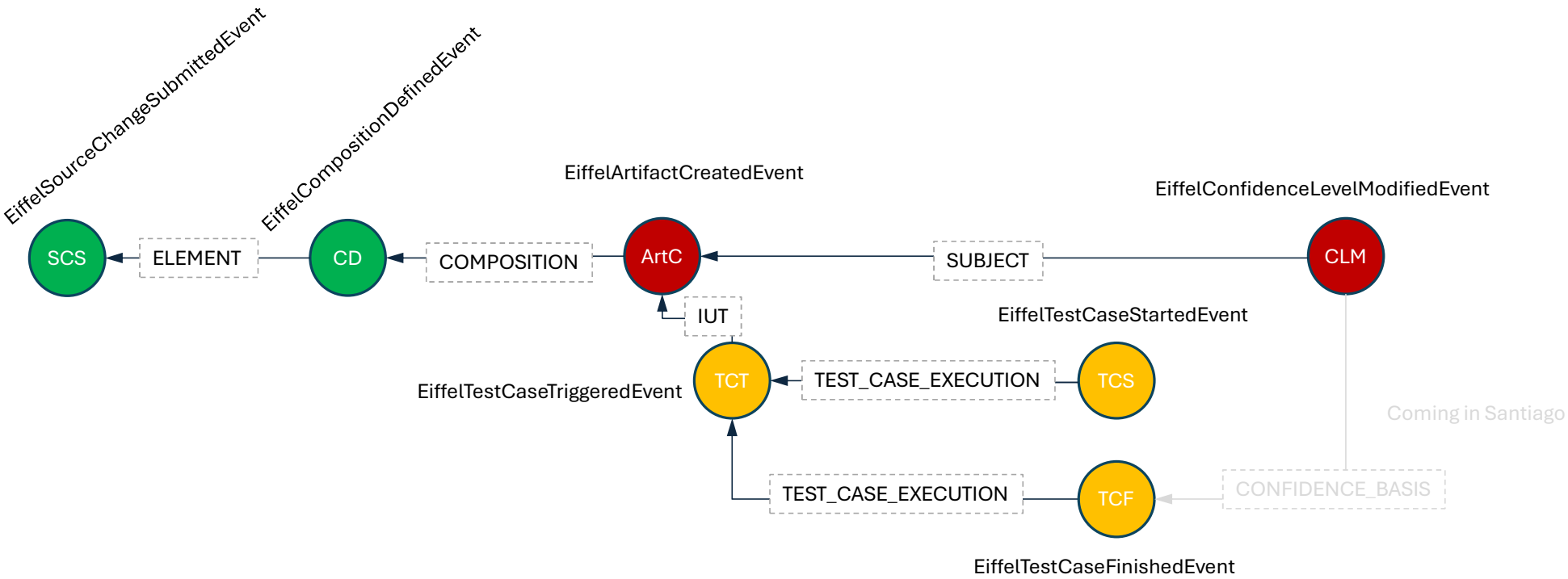
https://www.cloudraft.io/blog/cicd-observability-using-opentelemetry



https://newrelic.com/blog/how-to-relic/how-to-observe-your-cicd-pipelines-with-opentelemetry

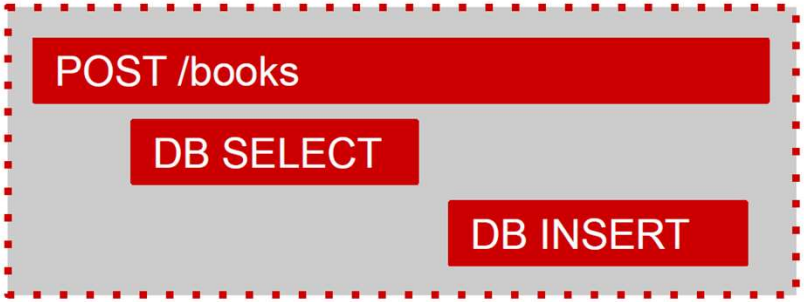
Let us take a closer look

Deep dive - Eiffel



A primer on traces and spans

A trace is a collection of spans.



Eiffel

From Magnus Bäck's presentation:
https://github.com/eiffel-community/community/blob/master/presentations/Eiffel%20and%20OpenTelemetry.pdf

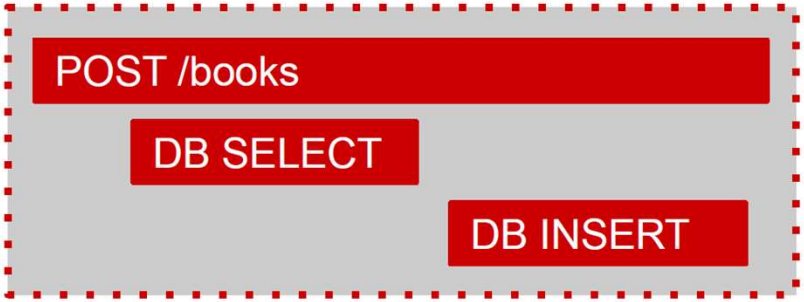
A primer on traces and spans

A trace is a collection of spans.

A span describes what happens, when, and for how long.

A span can declare relationships to other spans, most notably its parent.

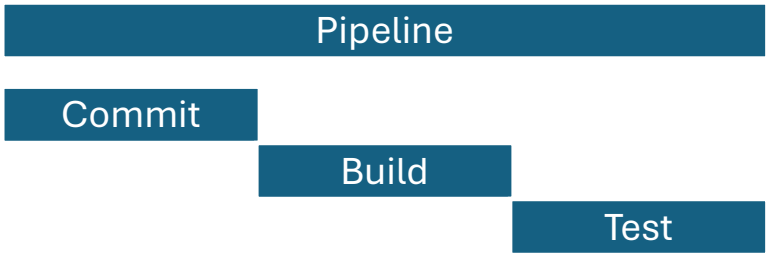
A span has attributes that provide additional context and enables searching and aggregating spans.



Eiffel

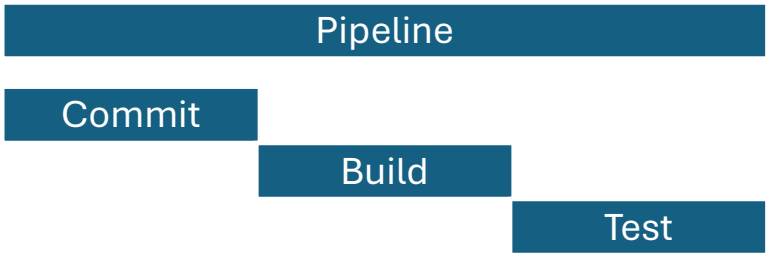
From Magnus Bäck's presentation:
https://github.com/eiffel-community/community/blob/master/presentations/Eiffel%20and%20OpenTelemetry.pdf

OTEL version (simplified)




```
{
  "traceId":
"4bf92f3577b34da6a3ce929d0e0e4736",
  "spans": [
    {
      "...": "...",
      "vcs.change.id": "..",
      "vcs.change.state": "..",
      "vcs.change.title": ".."
    },
    {
      "...": "...",
      "cicd.pipeline.name": "..",
      "cicd.pipeline.action.name": "..",
      "cicd.pipeline.run.id": ".."
    },
    {
      "...": "...",
      "test.case.name": "..",
      "test.case.result.status": ".."
    }
  ]
}
```

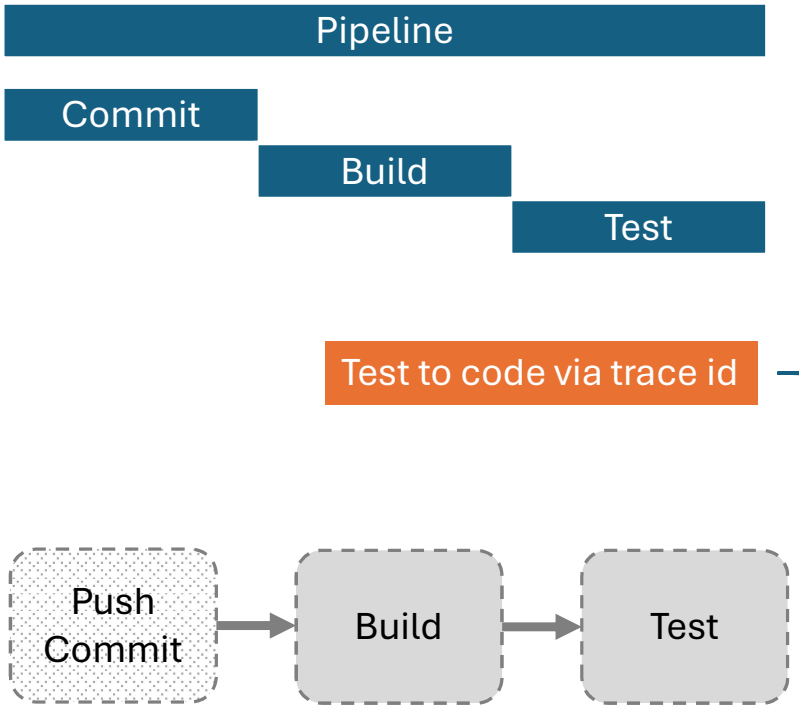
OTEL version




```
{
  "traceId":
"4bf92f3577b34da6a3ce929d0e0e4736",
  "spans": [
    {
      "...": "...",
      "vcs.change.id": "..",
      "vcs.change.state": "..",
      "vcs.change.title": ".."
    },
    {
      "...": "...",
      "cicd.pipeline.name": "..",
      "cicd.pipeline.action.name": "..",
      "cicd.pipeline.run.id": ".."
    },
    {
      "...": "...",
      "test.case.name": "..",
      "test.case.result.status": ".."
    }
  ]
}
```


Do you see any limitations with this?

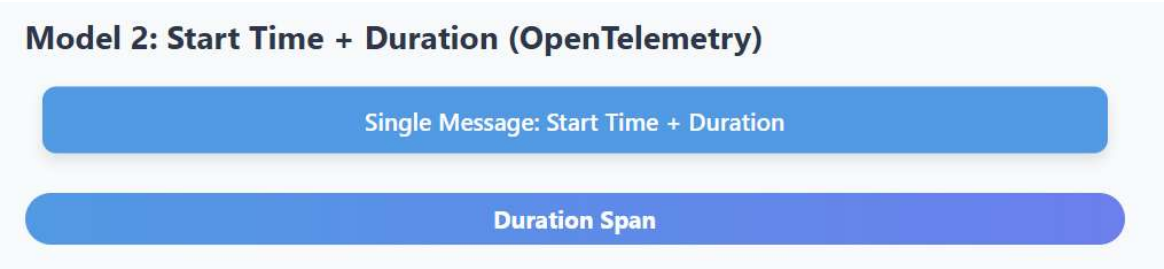
OTEL version

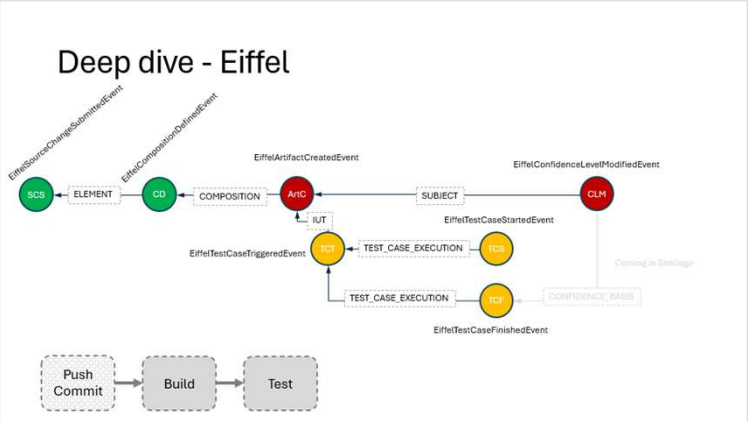


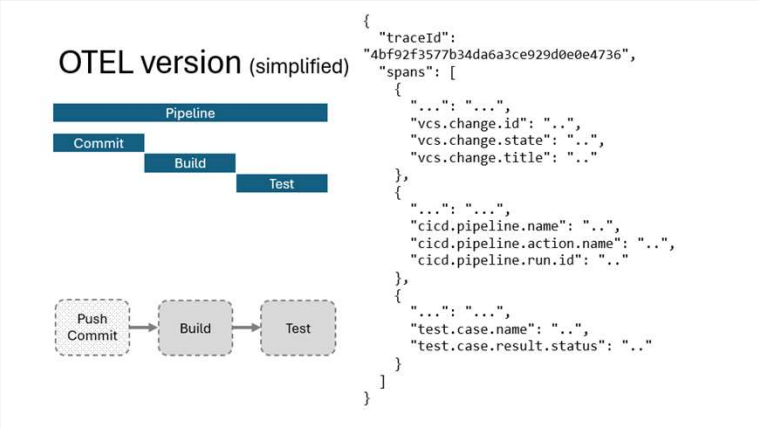

```
{
  "traceId":
"4bf92f3577b34da6a3ce929d0e0e4736",
  "spans": [
    {
      "...": "...",
      "vcs.change.id": "..",
      "vcs.change.state": "..",
      "vcs.change.title": ".."
    },
    {
      "...": "...",
      "cicd.pipeline.name": "..",
      "cicd.pipeline.action.name": "..",
      "cicd.pipeline.run.id": ".."
    },
    {
      "...": "...",
      "test.case.name": "..",
      "test.case.result.status": ".."
    }
  ]
}
```


No artifact concept

Thinking of Tracing

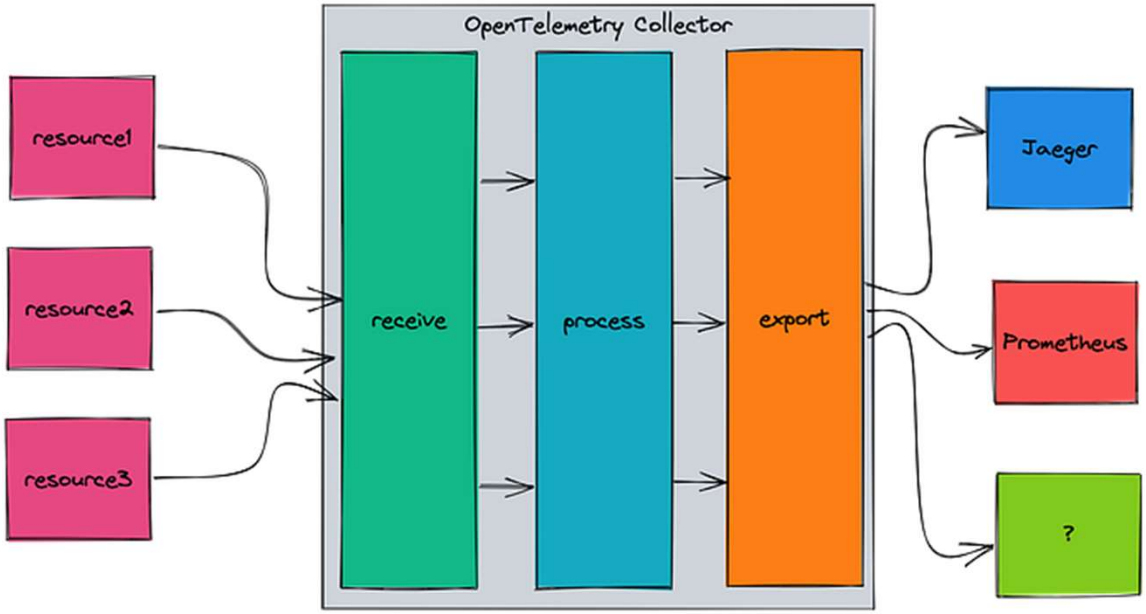






Some notes and observations...

Focus on Observability not interoperability



No defined Triggers

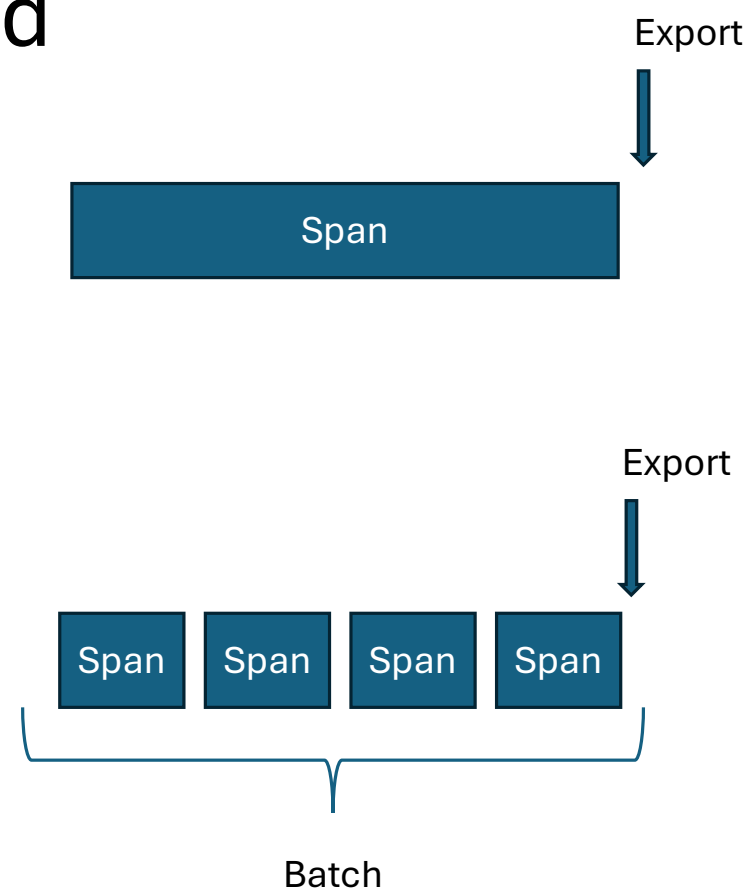
- Eiffel – I want to listen to this
- OTEL – I want you to hear this, (a slack user will not config a new listen from slack)

https://medium.com/ekino-france/next-js-observability-the-opentelemetry-case-study-92a687a69a3d

Spans Exported at the end

- Spans are exported after they end

- By default, the Node SDK uses the BatchSpanProcessor, and this span processor is also chosen in the Web SDK example. The BatchSpanProcessor processes spans in batches before they are exported. This is usually the right processor to use for an application.



OTEL have started to look at long running tasks

Visibility Challenge - Long-Running Processes and Ungraceful Shutdowns #4646

mladjan-gadzic started this conversation in **Ideas -> please open issues instead for feature requests!**

mladjan-gadzic last week edited

TL;DR:

`Partial Trace Connector` solution proposal fixes the problem of losing trace data when processes `crash` or are `terminated` unexpectedly. The connector would periodically export incomplete spans from `long-running` operations, ensuring better observability and debugging capabilities even when processes don't finish properly. Key benefits include real-time monitoring and complete traces despite system failures, though questions remain about implementation details like timing intervals and completion detection methods.

Overview

This proposal introduces a new component called a "partial trace connector" to handle incomplete spans from processes that may be long-running, crash or terminate unexpectedly, ensuring trace completeness in distributed systems.

Current Problem

When a process crashes or terminates unexpectedly, spans that were in progress are lost, leading to incomplete traces and making it difficult to debug issues in distributed systems.

Visibility Challenges for Long-Running Processes

Long-running processes with active spans present significant visibility challenges in observability systems:

Lack of Real-Time Monitoring:

- Operations that run for minutes, hours, or days provide no observability data until completion
- System operators cannot see current progress, resource utilization, or performance metrics
- Debugging active issues becomes impossible without visibility into ongoing operations

Category

Ideas -> please open issues instead for feature requests!

Labels

None yet

4 participants

Notifications

Subscribe

You're not receiving notifications from this thread.

Events in OTEL

Docs / Specs / Semantic conventions 1.37.0 / General / Events

Semantic conventions for events

Status: Development

This document describes the characteristics of standalone Events that are represented in the data model by `LogRecord` s.

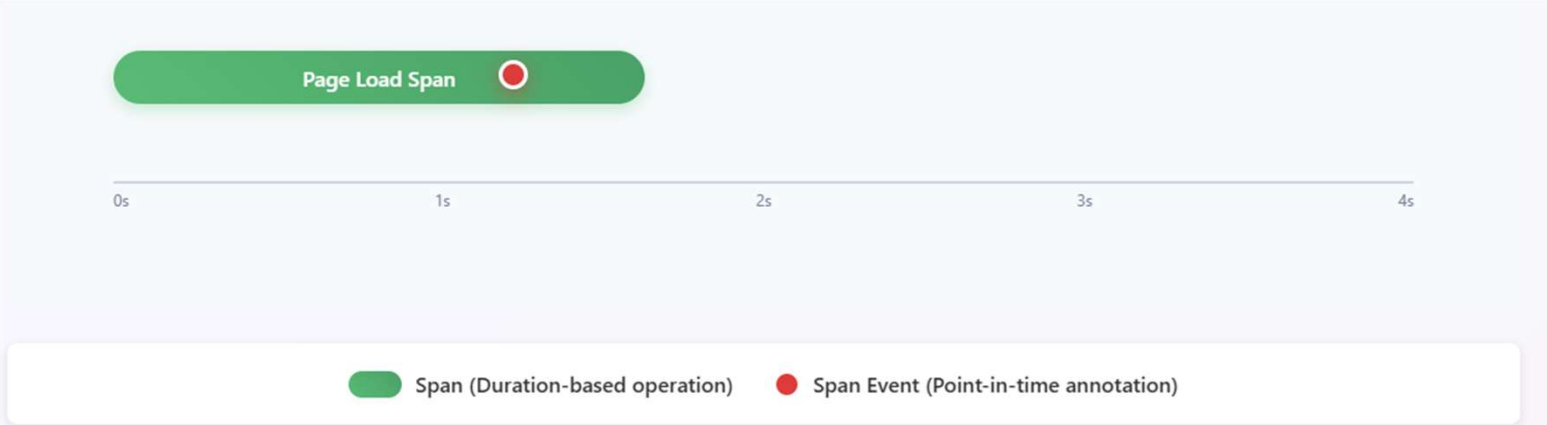
Semantically, an Event is a named occurrence at an instant in time. It signals that "this thing happened at this time" and provides additional specifics about the occurrence. Examples of Events might include things like button clicks, user logout, network connection severed, etc.

In OpenTelemetry, Events are implemented as a specific type of `LogRecord` that conforms to the conventions included here.

OpenTelemetry Semantic Conventions that define events SHOULD document the event name along with attributes and the type of the body if any.

https://opentelemetry.io/docs/specs/semconv/general/events/

Span and Events



Page Load Scenario

The **Span** represents the entire page loading operation, which has a clear start (navigation begins) and end (page fully loaded). This duration-based operation is perfect for a Span because it tracks the complete lifecycle of an operation.

The **Span Event** marks the exact moment when the page becomes interactive - a meaningful point in time that doesn't have a duration. This singular moment is ideal for a Span Event because it annotates a specific instant within the broader page load operation.

Span Events

A Span Event can be thought of as a structured log message (or annotation) on a Span, typically used to denote a meaningful, singular point in time during the Span's duration.

For example, consider two scenarios in a web browser:

1. Tracking a page load
2. Denoting when a page becomes interactive

A Span is best used to the first scenario because it's an operation with a start and an end.

A Span Event is best used to track the second scenario because it represents a meaningful, singular point in time.

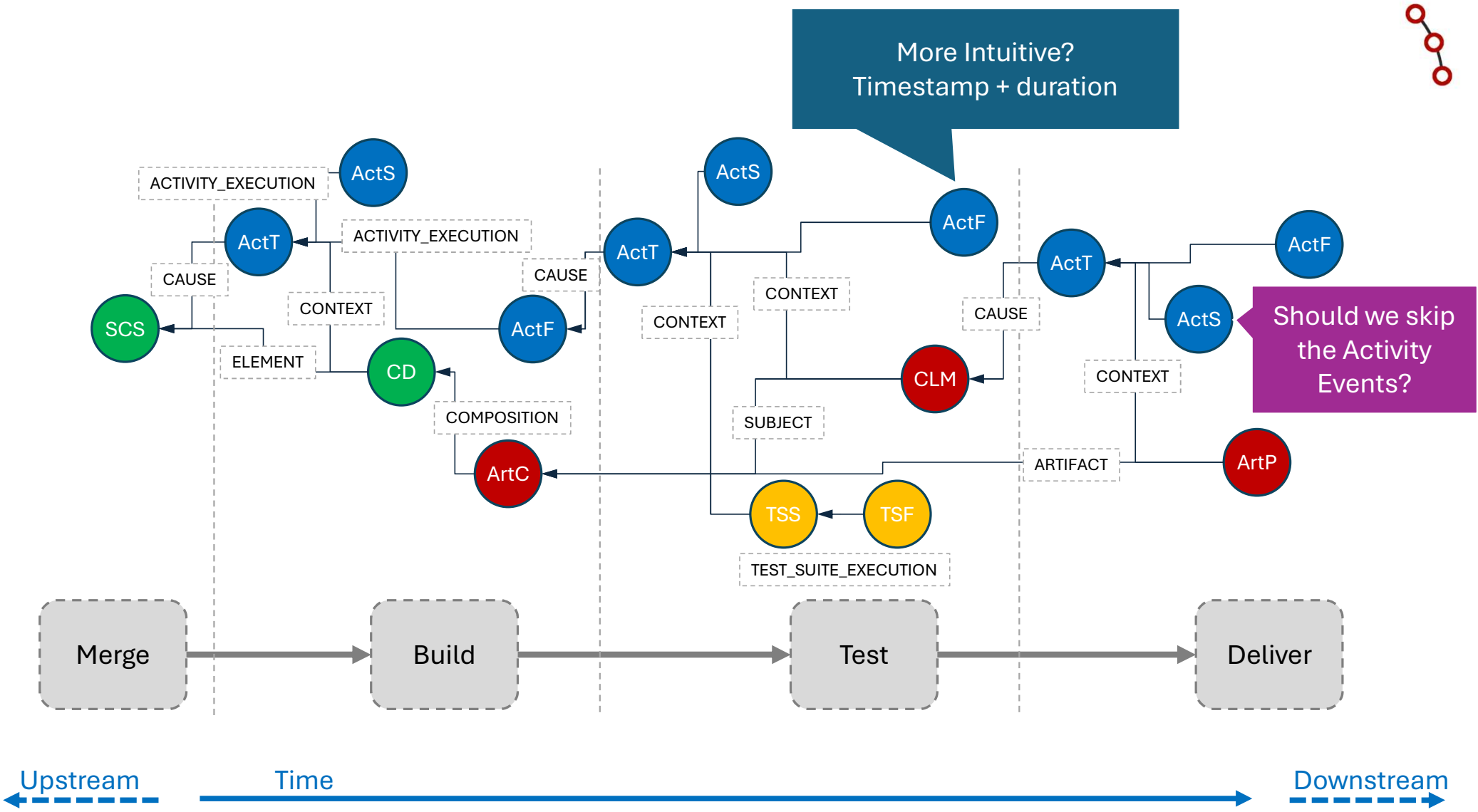
https://opentelemetry.io/docs/concepts/signals/traces/#span-events


```
{
  "name": "/v1/sys/health",
  "context": {
    "trace_id": "7bba9f33312b3dbb8b2c2c62bb7abe2d",
    "span_id": "086e83747d0e381e"
  },
  "parent_id": "",
  "start_time": "2021-10-22 16:04:01.209458162 +0000 UTC",
  "end_time": "2021-10-22 16:04:01.209514132 +0000 UTC",
  "status_code": "STATUS_CODE_OK",
  "status_message": "",
  "attributes": {
    "net.transport": "IP.TCP",
    "net.peer.ip": "172.17.0.1",
    "net.peer.port": "51820",
    "net.host.ip": "10.177.2.152",
    "net.host.port": "26040",
    "http.method": "GET",
    "http.target": "/v1/sys/health",
    "http.server_name": "mortar-gateway",
    "http.route": "/v1/sys/health",
    "http.user_agent": "Consul Health Check",
    "http.scheme": "http",
    "http.host": "10.177.2.152:26040",
    "http.flavor": "1.1"
  },
  "events": [
    {
      "name": "",
      "message": "OK",
      "timestamp": "2021-10-22 16:04:01.209512872 +0000 UTC"
    }
  ]
}
```

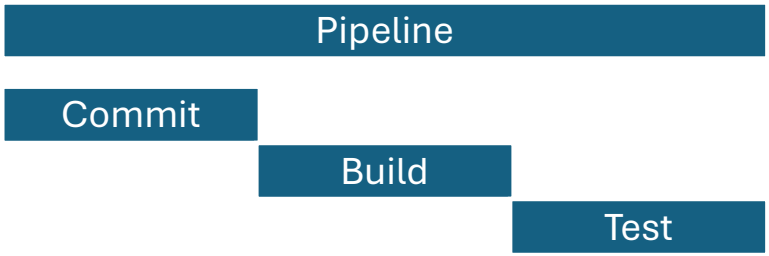
Discussion time

FIRST REACTION?

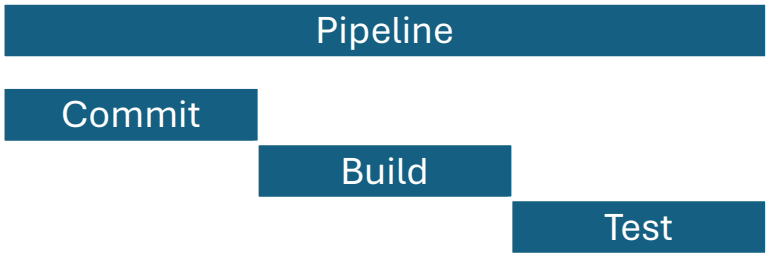
HOW MUCH OTEL ARE YOU USING
IN YOUR PIPELINES TODAY?


More Intuitive?
Timestamp + duration
ACTIVITY_EXECUTION
ActS
ActT
CAUSE
SCS
CONTEXT
ELEMENT
CD
ACTIVITY_EXECUTION
ActF
COMPOSITION
ArtC
CAUSE
ActS
ActT
CONTEXT
CONTEXT
ActF
CLM
SUBJECT
TSS
TSF
TEST_SUITE_EXECUTION
ARTIFACT
CAUSE
ActT
ActF
ActS
CONTEXT
ArtP
Should we skip the Activity Events?
Merge
Build
Test
Deliver
Upstream
Time
Downstream

Enough?

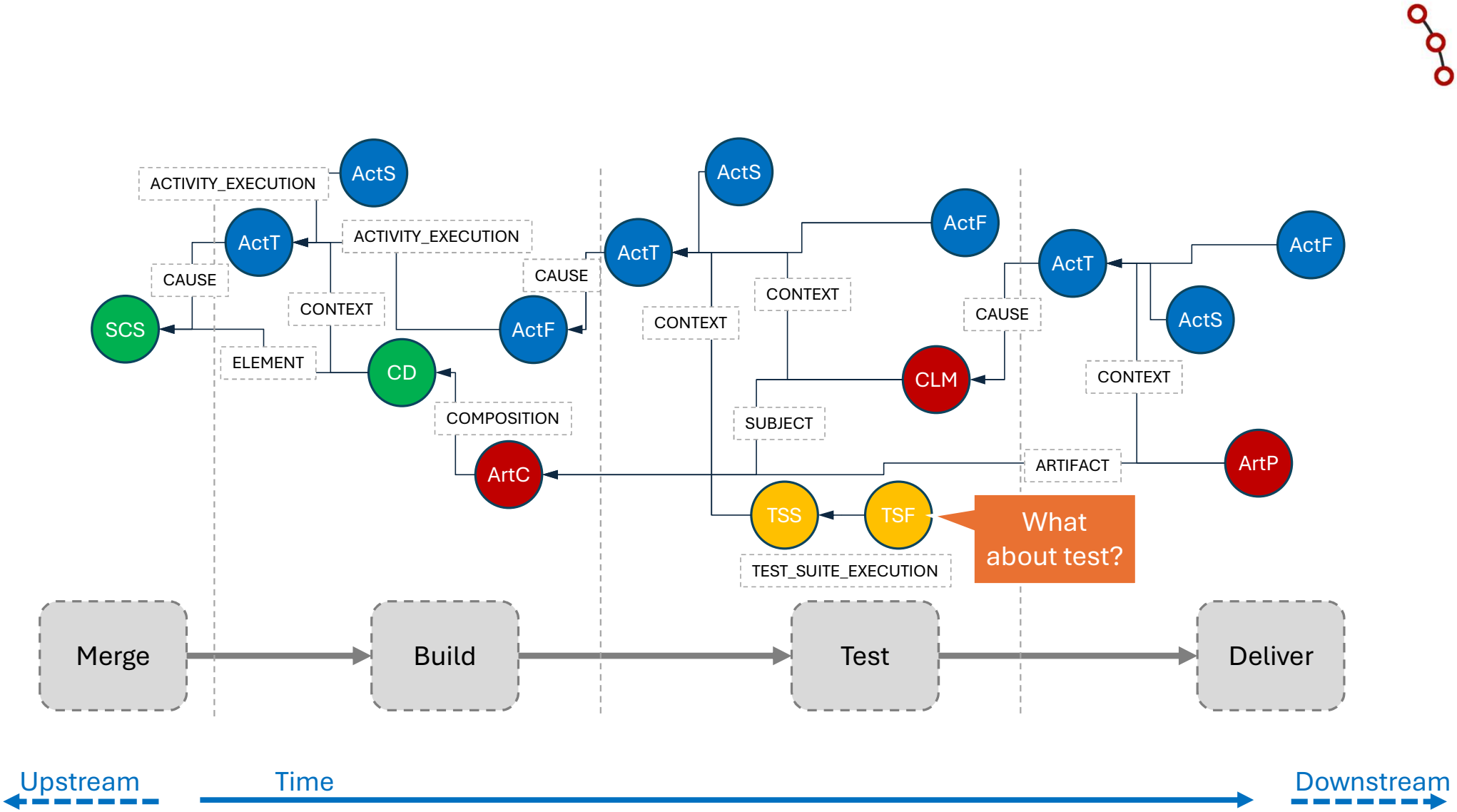


OTEL version

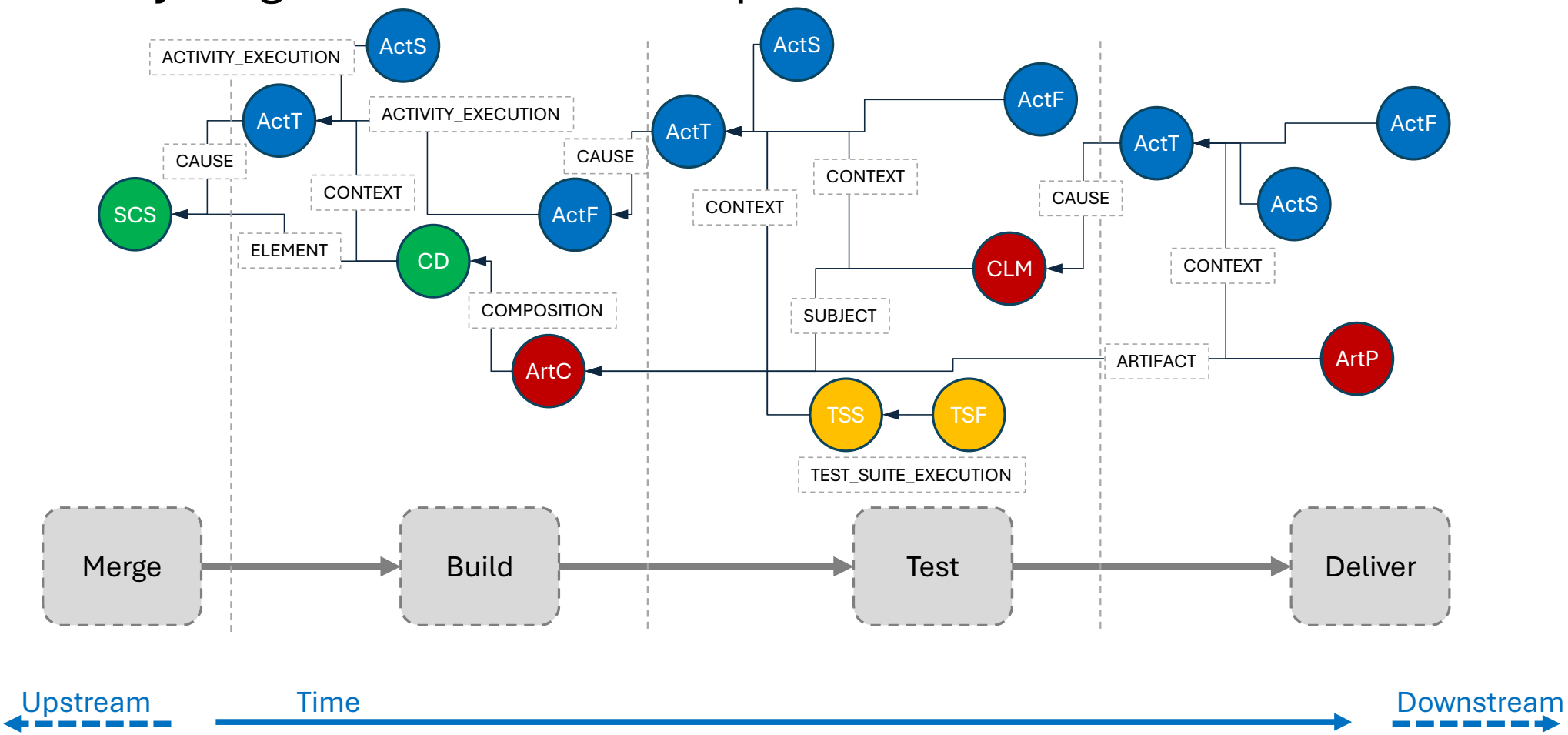



```
{
  "traceId":
"4bf92f3577b34da6a3ce929d0e0e4736",
  "spans": [
    {
      "...": "...",
      "vcs.change.id": "..",
      "vcs.change.state": "..",
      "vcs.change.title": ".."
    },
    {
      "...": "...",
      "cicd.pipeline.name": "..",
      "cicd.pipeline.action.name": "..",
      "cicd.pipeline.run.id": ".."
    },
    {
      "...": "...",
      "test.case.name": "..",
      "test.case.result.status": ".."
    }
  ]
}
```


Do you see any limitations with this?

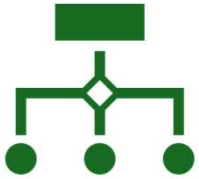

ACTIVITY_EXECUTION
ActS
ActT
ACTIVITY_EXECUTION
CAUSE
SCS
CONTEXT
ELEMENT
CD
ActF
COMPOSITION
ArtC
ActS
ActT
CAUSE
CONTEXT
ActF
CONTEXT
CLM
SUBJECT
TSS
TSF
TEST_SUITE_EXECUTION
What about test?
ActT
CAUSE
ActF
ActS
CONTEXT
ARTIFACT
ArtP
Merge
Build
Test
Deliver
Upstream
Time
Downstream

Anything else that could be replaced



Your role in the future

Should we work on connecting the two?



Should Eiffel just “let go” of certain parts?

How do we ensure Eiffel's voice is heard in the OpenTelemetry CI/CD conversations?

Getting involved

- CI/CD Observability Semantic Conventions SIG
- Issues
  - Visibility Challenge - Long-Running Processes and Ungraceful Shutdowns #4646
  - CI/CD: producing long running traces #1648
- Reading
  - https://www.cncf.io/blog/2024/11/04/opentelemetry-is-expanding-into-ci-cd-observability/

https://wallpapers.com/wallpapers/looney-tunes-that-s-all-folks-qslztlo8w335ubvc.html