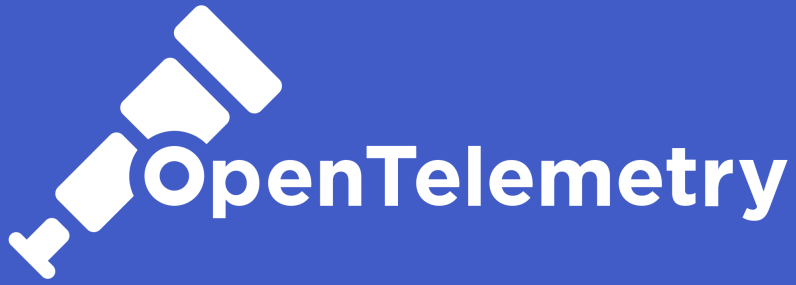


Overview and Backwards Compatibility of OpenTracing and OpenCensus

Steve Flanders (Omnition)

CloudNativeCon China, Shanghai 2019

Steve Flanders
Head of Product and Experience, Omnition
@smflanders | https://sflanders.net

OpenTelemetry

Background: OpenTracing and OpenCensus

Cloud-native telemetry

Telemetry "verticals"

Telemetry "layers"

	Tracing	Metrics	Logs, etc
Instrumentation APIs	`foreach(language)`		
Canonical implementations	`foreach(language)`		
Data infrastructure	collectors, sidecars, etc		
Interop formats	w3c trace-context, wire formats for trace data, metrics, logs, etc		

Overlapping but non-identical

- One “vertical” (Tracing)
- One “layer” (API)
- “Looser” coupling (small scope)
- Lots of languages (~12)
- Broad adoption
- (FYI: Already part of CNCF)

- Many “verticals” (Tracing, Metrics)
- Many “layers” (API, impl, infra)
- “Tighter” coupling (framework-y)
- Many languages (5 in beta)
- Broad adoption

OpenTelemetry

Overlapping but non-identical, and some issues

- One “vertical” (Tracing)
  Users want only one dependency
- One “layer” (API)
- “Looser” coupling (small scope)
- Lots of languages (~12)
- Broad adoption
- (FYI: Already part of CNCF)

- Many “verticals” (Tracing, Metrics)
- Many “layers” (API, impl, infra)
- “Tighter” coupling (framework-y)
  Users and vendors want flexibility
- Many languages (5 in beta)
- Broad adoption

Both widely adopted

- 370 contributors
- 5360 stars on GitHub
- 100s of supported integrations with OSS libraries and frameworks

- 390 contributors
- 3392 stars on GitHub
- Backed by Google, Microsoft, Omnition, Postmates, Dynatrace, Shopify (soon)

OpenTelemetry

So both projects are well-adopted! Great!

… kinda.

Choice: not always a good thing

No “clear winner” between OpenTracing and OpenCensus

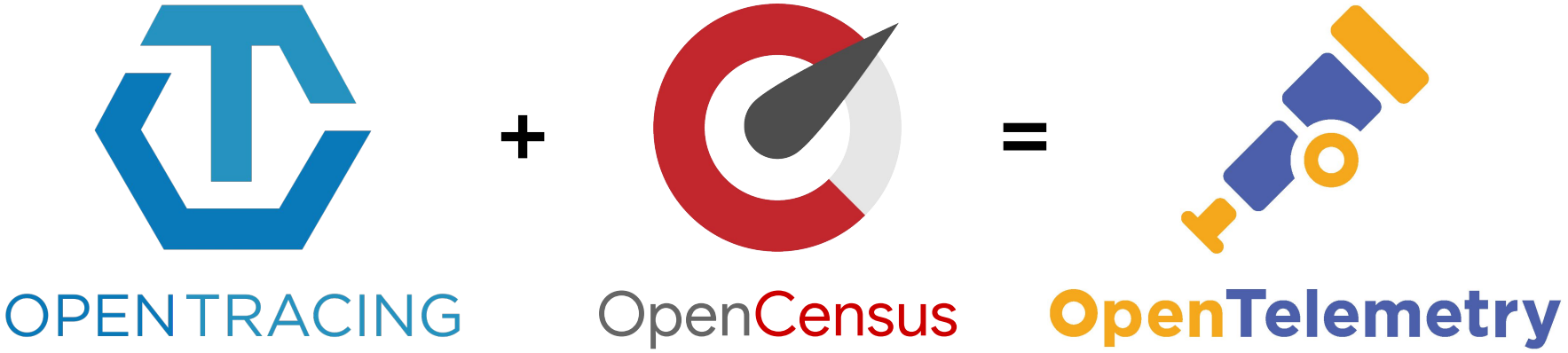
- “Lots of integration” vs “canonical implementation”
- Both projects have “escape velocity”

Ecosystem confused and visibly held back

- Both end-users *and* vendors
- E.g., this Hadoop ecosystem ticket where the choice was irresolvable

… and all of this made worse by general “negativity bias” on twitter/etc

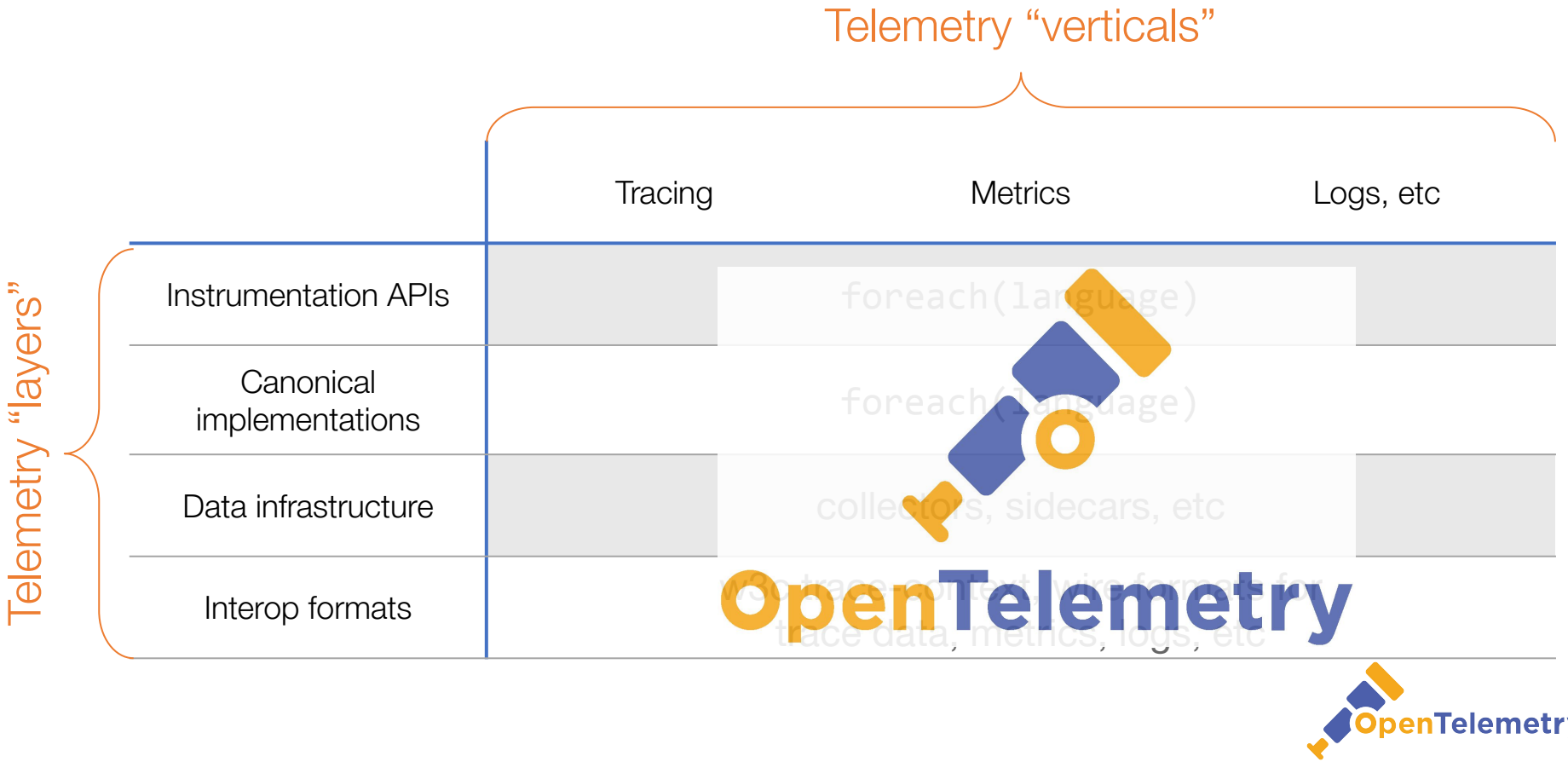
OpenTelemetry



OpenTelemetry: **the next major version**
of *both* OpenTracing and OpenCensus

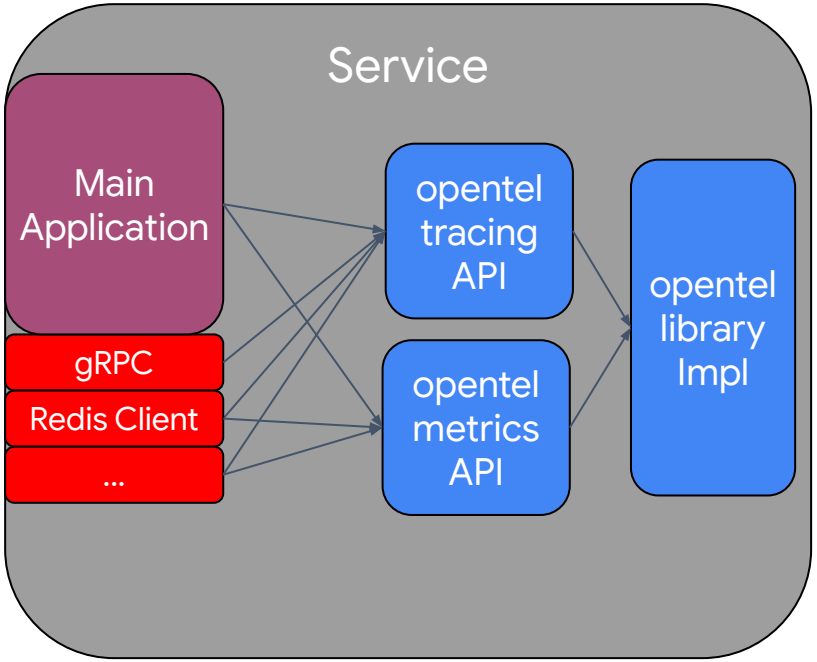
OpenTelemetry Overview

Cloud-native telemetry



OpenTelemetry

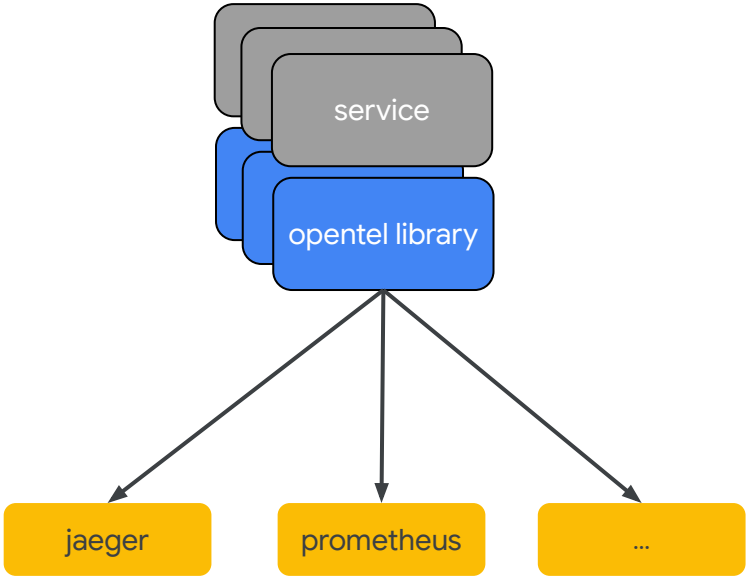
Instrumentation APIs



- Context propagation, tracing, metrics and correlation between them. Eventually logs
- Compatibility with existing OT/OC via bridge
- Will rely on and broaden the instrumentation base of OpenTracing/OpenCensus

OpenTelemetry

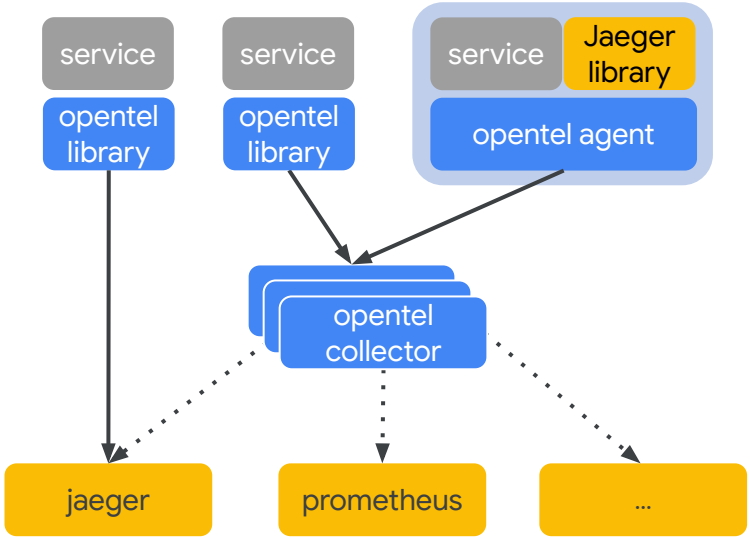
Canonical implementations



- One “reference implementation” for each language
- W3C trace-context based context propagation
- Open wire format for tracing, metrics, logs
-

OpenTelemetry

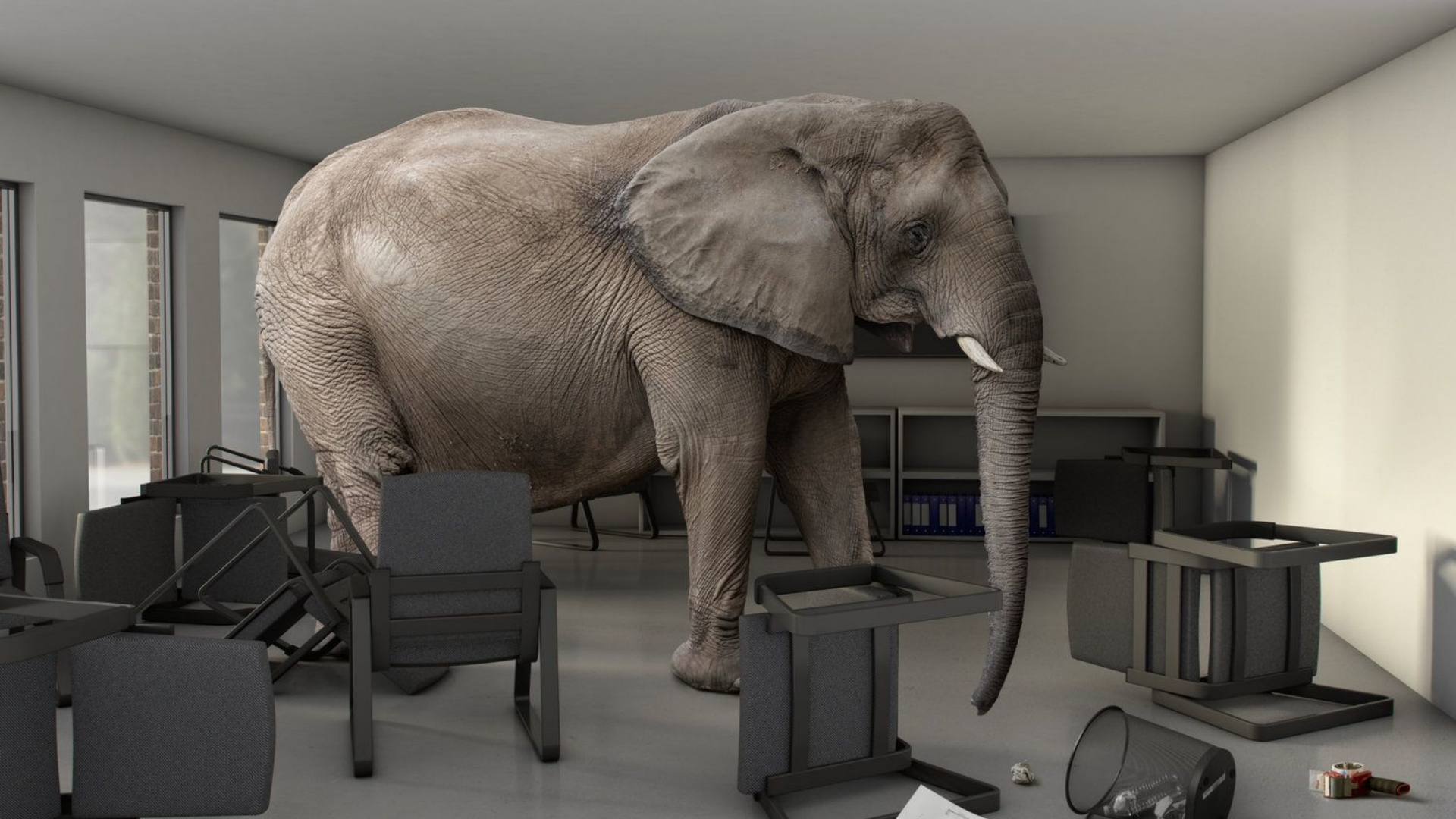
Data infrastructure (coming)

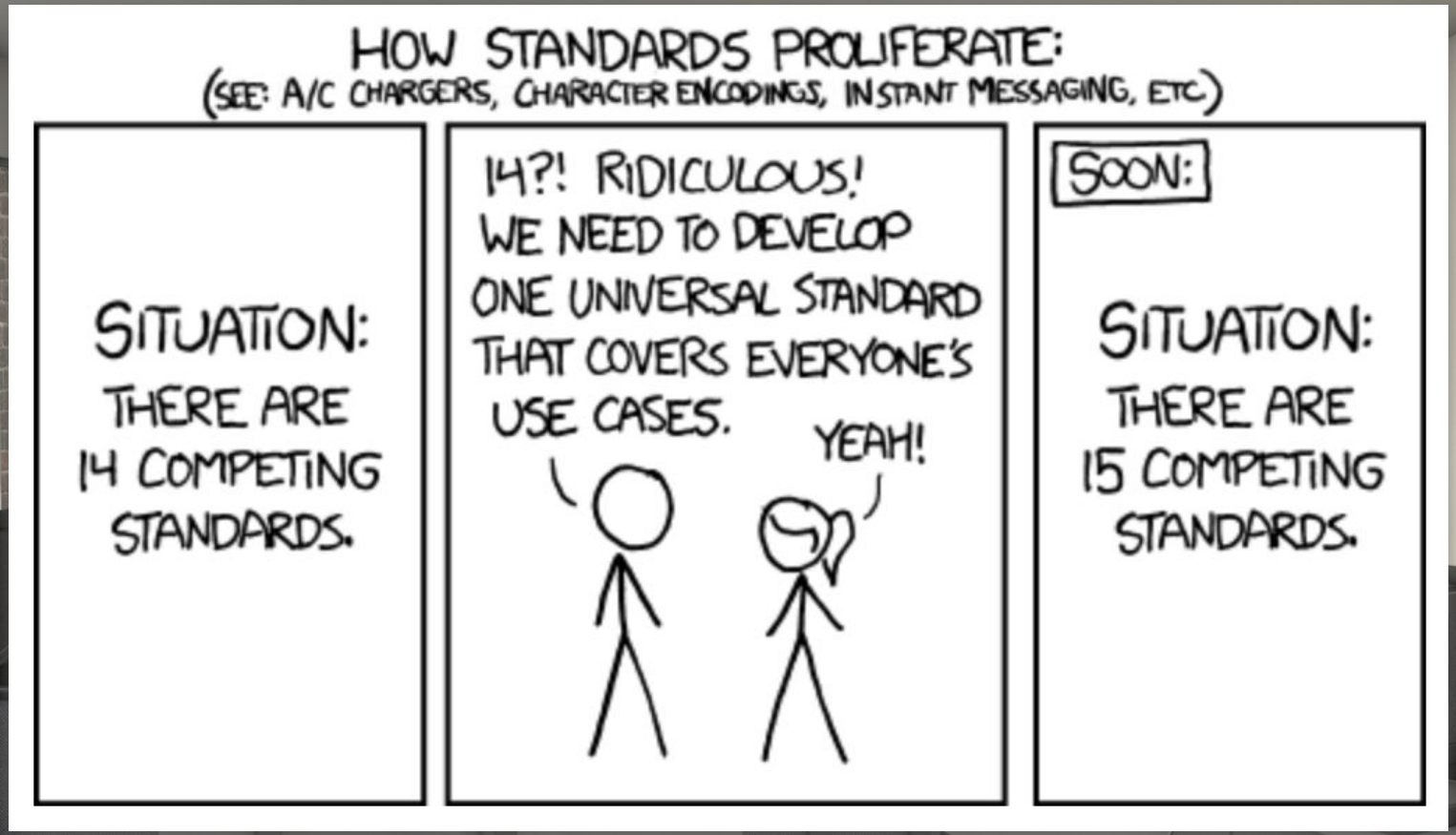


- Support for all popular OSS and commercial backends
- Application and infrastructure metric collection
- For standalone applications (e.g. Envoy or HBase) the backend can be configured in the agent without recompiling
- Tail-based sampling

OpenTelemetry

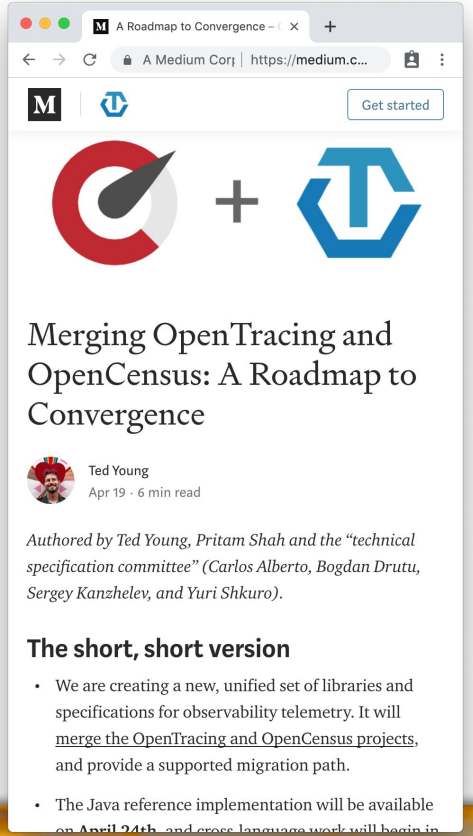
OpenTelemetry Goals




HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

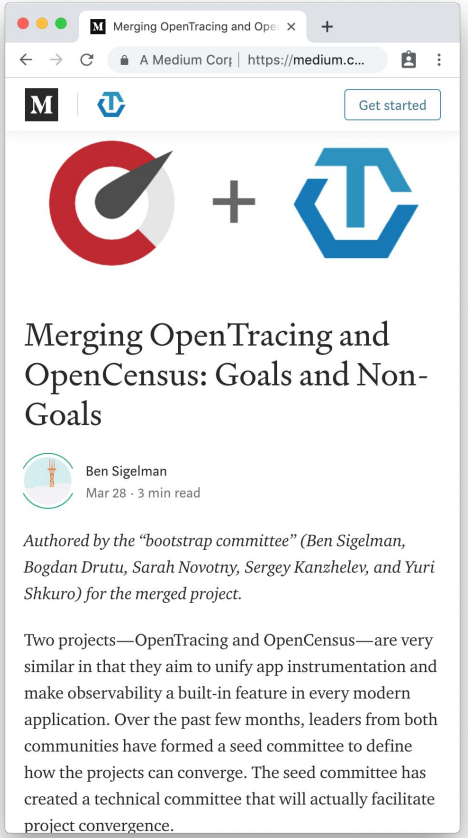
OpenTelemetry Goals: 2019

- Backwards-compatibility
- Backwards-compatibility
- Backwards-compatibility
- **Sept 2019:** “Time-to-parity” in major langs
- **Nov 2019:** OpenTracing and OpenCensus officially sunsetted (i.e., read-only)
  - Two-year compatibility guarantee



OpenTelemetry Goals: Long-term

- **One** project (not two, and def not three!)
- Broad surface area (tracing, metrics, APIs, reference impls, sidecars, data formats)
- Loose coupling (only take what you need)
- Open governance with representation from many orgs



OpenTelemetry

OpenTelemetry Governance

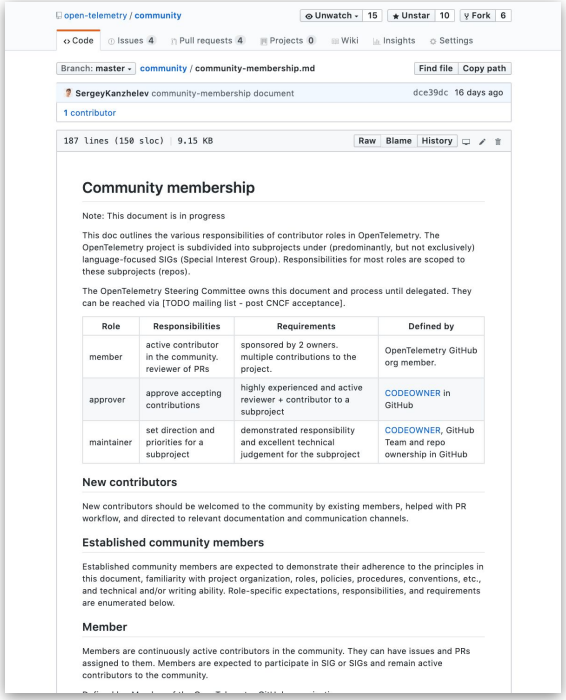
OpenTelemetry community membership

A SIG for the cross-language specification

SIGs for each language

Three levels of community membership:

1. Member: contributor, reviewer
2. Approver: experienced reviewer and approver
3. Maintainer: set direction and priorities



open-telemetry / community

Unwatch 15 | Unstar 10 | Fork 6

Code | Issues 4 | Pull requests 4 | Projects 0 | Wiki | Insights | Settings

Branch: master | community / community-membership.md | Find file | Copy path

SergeyKanzhelev community-membership document dce39dc 16 days ago

1 contributor

187 lines (150 sloc) | 9.15 KB | Raw | Blame | History

Community membership

Note: This document is in progress

This doc outlines the various responsibilities of contributor roles in OpenTelemetry. The OpenTelemetry project is subdivided into subprojects under (predominantly, but not exclusively) language-focused SIGs (Special Interest Group). Responsibilities for most roles are scoped to these subprojects (repos).

The OpenTelemetry Steering Committee owns this document and process until delegated. They can be reached via [TODO mailing list - post CNCF acceptance].

Role	Responsibilities	Requirements	Defined by
member	active contributor in the community. reviewer of PRs	sponsored by 2 owners. multiple contributions to the project.	OpenTelemetry GitHub org member.
approver	approve accepting contributions	highly experienced and active reviewer + contributor to a subproject	CODEOWNER in GitHub
maintainer	set direction and priorities for a subproject	demonstrated responsibility and excellent technical judgement for the subproject	CODEOWNER, GitHub Team and repo ownership in GitHub

New contributors

New contributors should be welcomed to the community by existing members, helped with PR workflow, and directed to relevant documentation and communication channels.

Established community members

Established community members are expected to demonstrate their adherence to the principles in this document, familiarity with project organization, roles, policies, procedures, conventions, etc., and technical and/or writing ability. Role-specific expectations, responsibilities, and requirements are enumerated below.

Member

Members are continuously active contributors in the community. They can have issues and PRs assigned to them. Members are expected to participate in SIG or SIGs and remain active contributors to the community.

OpenTelemetry

OpenTelemetry governance

TL;DR edition:

- Using CNCF Code of Conduct
- Intended to represent many organizations / companies
- Elections based on k8s: 9 seats with limits on overrepresentation
  - Active code contributors get to vote
- Maximum term limits

OpenTelemetry

Backwards Compatibility:
OpenTracing

Changes: SpanContext

- Generally, “highly non-abstract” (i.e., in Java it’s a `final` class)
- TraceIDs and SpanIDs much more prescriptive in OpenTelemetry
  - based on w3c standard
- OpenTracing Baggage via w3c’s “correlation-context” concept
- In-process context propagation in its own package
- Introduces a more formal notion of “sampling” (actual methods, not just a standard string key like OpenTracing)

OpenTelemetry

Changes: Span

- A few terminology adjustments, but mostly 1:1 mappings:
  - OpenTracing “tags” become OpenTelemetry “attributes”
  - OpenTracing “logs” become OpenTelemetry “events”
- OpenTracing “References” become OpenTelemetry “parent” Spans that may have varying “kinds” (e.g., Server, Client, Producer, Consumer, Internal)
- OpenTelemetry includes an OpenCensus-like “linked Span” concept: other elements in a batch, etc, etc
- User-provided timestamps for Span start/end not part of OpenTel’s **API**, though present in wire formats. Can be added if.f. needed.

OpenTelemetry

Changes: Tracer

- Both provide access to the current active Span
- OpenTelemetry introduces the concept of a “Resource” that can attach data to every Span created by the Tracer
- OpenTelemetry will probably not support Tracer.close(), though it probably will support Tracer.flush()

OpenTelemetry

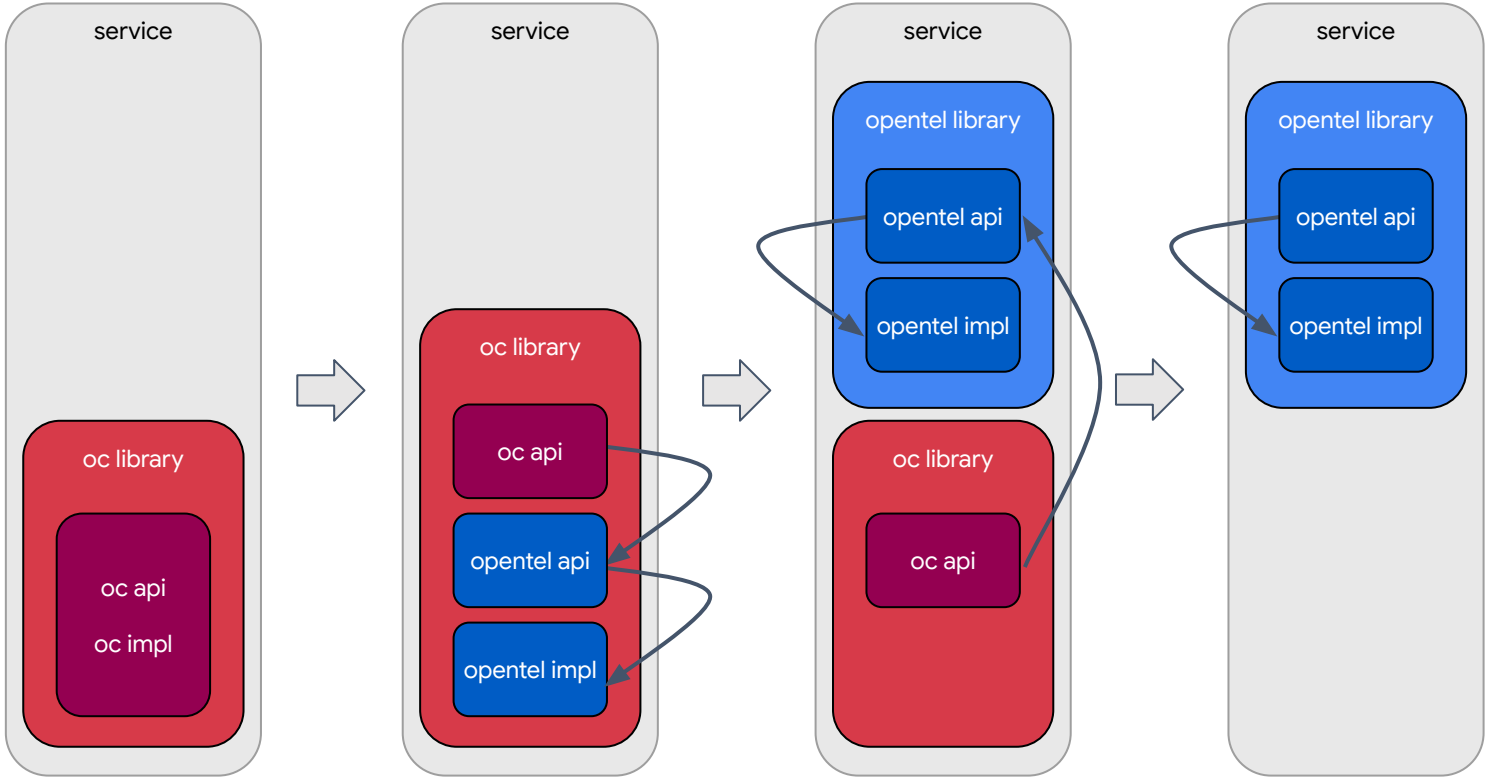
Backwards Compatibility:

OpenCensus

OpenTelemetry vs OpenCensus

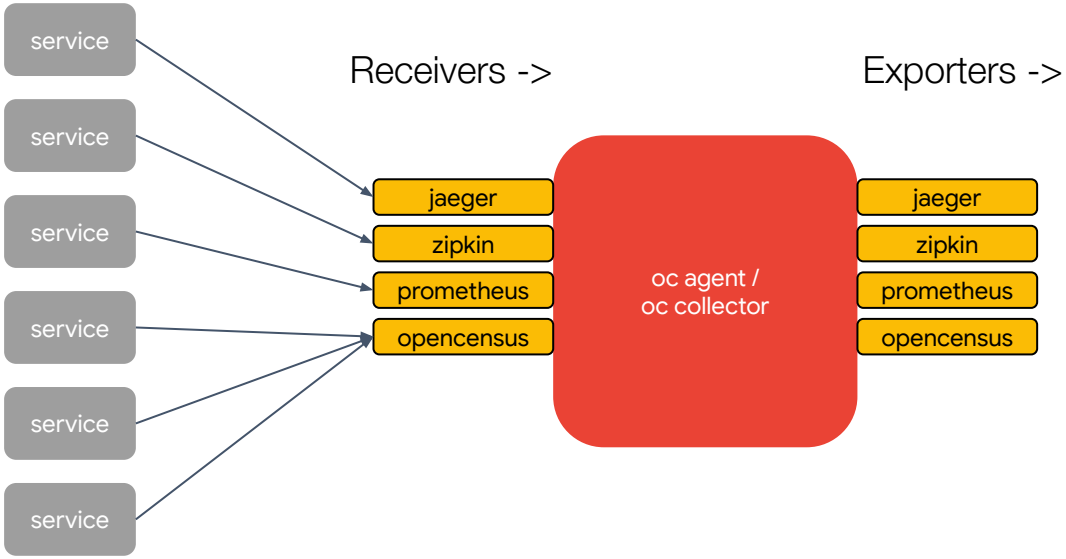
- Clear separation between the instrumentation API specification and the implementation (alternative implementations can be used)
- A few small changes to the instrumentation API
- OpenTelemetry inherits OpenCensus client library implementations with modifications to adhere to the new API spec
- OpenTelemetry inherits the collection infrastructure from OpenCensus

Transition plan for OpenCensus library



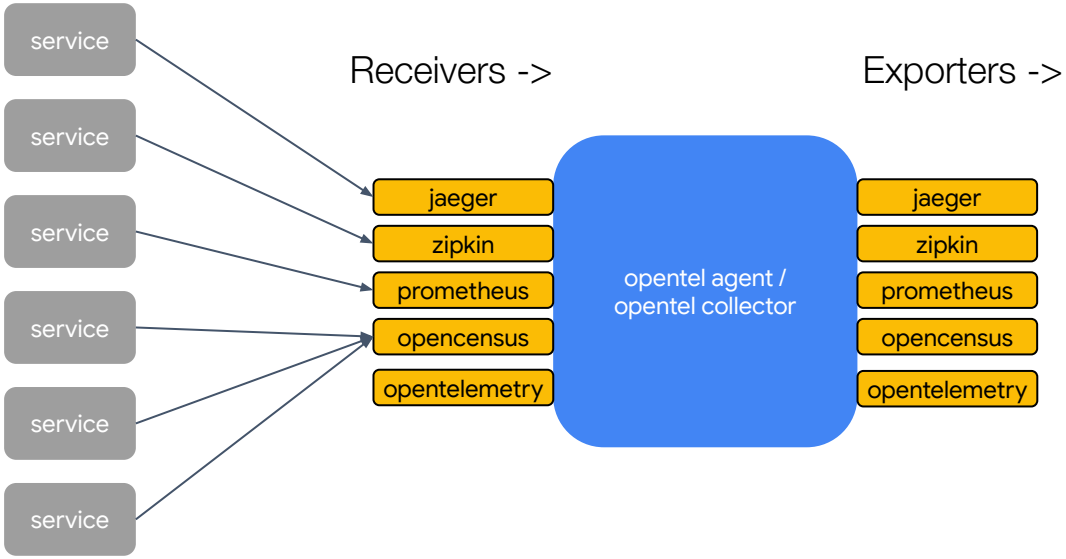
OpenTelemetry

Transition plan for OpenCensus Service



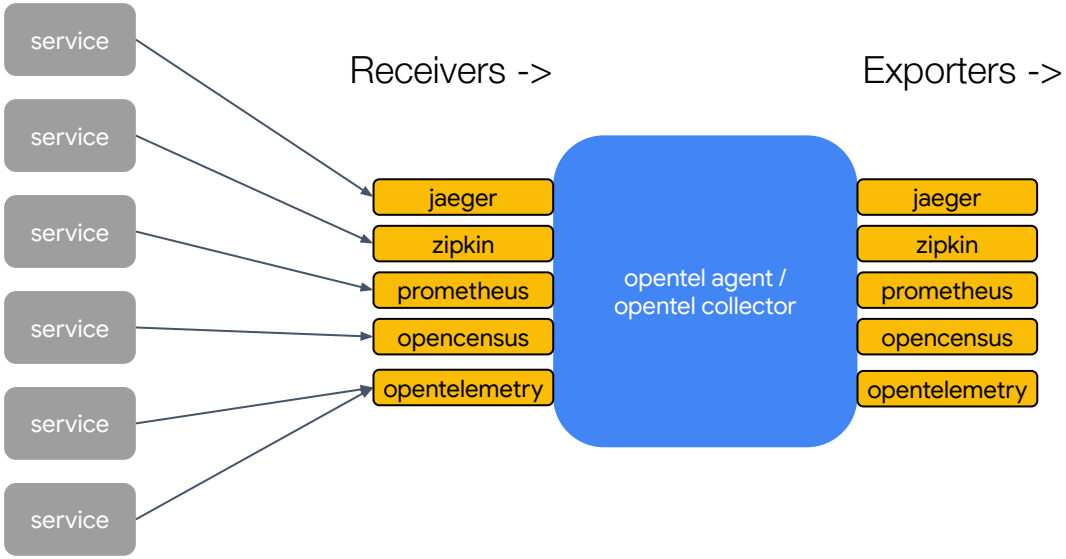
OpenTelemetry

Transition plan for OpenCensus Service



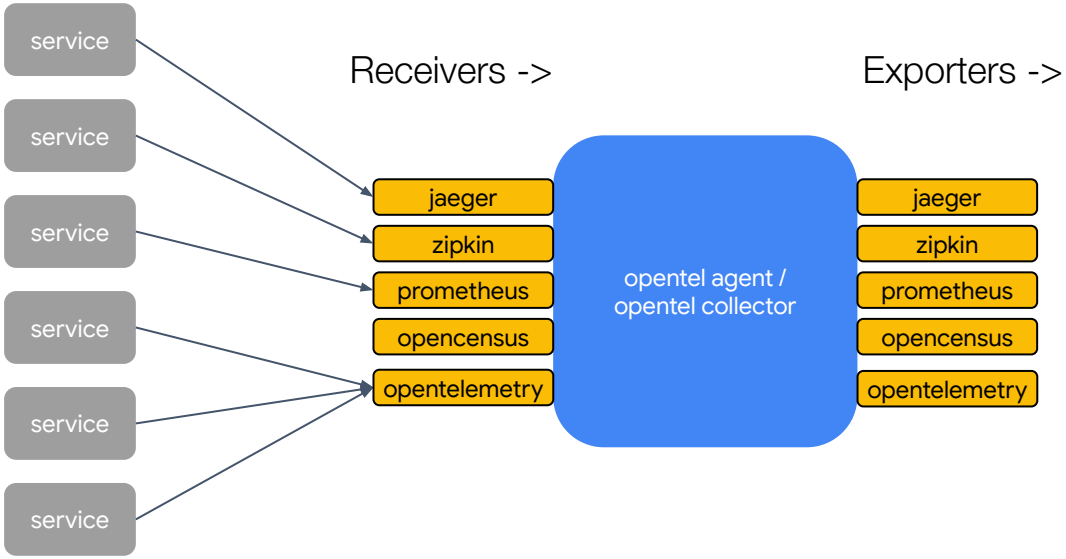
OpenTelemetry

Transition plan for OpenCensus Service



OpenTelemetry

Transition plan for OpenCensus Service



OpenTelemetry

Summary

What's Next

- First version of OpenTelemetry by end of 2019, support for 5 languages
- Bridge between OpenTelemetry and OpenTracing/OpenCensus
- OpenTracing and OpenCensus sunset (read-only) by end of 2019
- OpenTracing and OpenCensus backwards compatible through 2021
- OpenTracing terminology, context propagation, and sampling changes
- OpenCensus clear separation from API and implementation

OpenTelemetry: Q & A