OpenTelemetry Project Roadmap Draft

NOW PUBLISHED HERE: https://github.com/open-telemetry/community/pull/1312

~~What We've Accomplished~~

~~OpenTelemetry has come a long way since its announcement in May of 2019!~~

~~In the past three years, OpenTelemetry's language agents and SDKs have established themselves as the standard way to propagate and capture distributed traces within backend services, the Collector has been widely adopted across the industry as an easy and powerful way to capture and perform basic processing of traces and system metrics, and OpenTelemetry's native metrics support has been released to the public and is becoming adopted.~~

~~Since late 2020, OpenTelemetry has been the second most active project in the Cloud Native Computing Foundation, and the number of companies and people contributing to and using the project continues to grow. At the same time, there's a need within various project SIGs for more permanent maintainers, and community members have also expressed a desire for more vision to direct future work.~~

~~With tracing and metrics support now generally available across much of the project, OpenTelemetry has fulfilled its original promise. This represents an opportune time for us to come together as a community, take stock of what's working well and what isn't, and chart our next steps.~~

~~Community-wide Projects~~

~~Logging (in-progress)~~

~~OpenTelemety established a logging SIG in mid-2020, with two goals:~~

1. Providing a performant path for capturing logs from existing sources (typically text files on disk), where all captured logs have OpenTelemetry's metadata consistently applied to them.

2. Providing a new, strongly-typed and extremely high-performance logging path for new applications, which allows logs to be authored and transmitted without being parsed from text, and which enforces the consistency of all metadata.

Much progress has been made on the first item, particularly through the donation of the Stanza logging agent into the OpenTelemetry Collector and the merging of Elastic Common Schema into the OpenTelemetry semantic conventions as a single schema project. While this functionality is not yet considered GA, it is being used in production at several vendors and end-users. OpenTelemetry's logging specification and protocol are now GA, meaning that no breaking changes will occur to them. While more work is required to take this functionality to GA, the scope of this work is known and a project plan has already been established.

Client Instrumentation (in-progress)

OpenTelemetry JS has technically supported capturing spans from web browsers since its first releases, however this behaviour was mostly unspecified, and there was no equivalent functionality for other types of client applications like those on Android, iOS, or Windows.

In late 2021, the Client Instrumentation SIG (often called the RUM SIG) was established, which seeks to specify client instrumentation behaviour so that there is consistency in the data captured from and in the developer-facing telemetry interfaces in different types of client applications. This SIG is currently completing its first round of spec work, which will need to be implemented by the JS, Swift, Java, and other SIGs once it is complete.

OpenTelemetry Control Plane (in-progress)

Since OpenTelemetry's initiation, end-users and vendors have expressed a desire to (a) understand what SDKs, language agents, and Collectors are deployed within their environment (along with their status), and to (b) be able to make changes to the configuration of these artifacts or possibly even update agent binaries.

Specification work is already underway to address both of these needs, and the agent management SIG has already produced a specification for OpAMP, the protocol that will drive these interactions. Over time, the SIGs that develop various OpenTelemetry artifacts will need to implement OpAMP to enable these scenarios.

Profiling

Distributed profiling has been a long-standing topic of discussion within OpenTelemetry, and contributors to other profiling projects have advocated for it to be added to OpenTelemetry as an additional signal type.

Sampled heap and CPU profiles would allow OpenTelemetry to extend end-users' visibility to the performance of their actual code. While other profiling solutions allow this kind of inspection today, few are able to properly correlate profiles with application and infrastructure resource metadata, and even fewer are able to correlate profiling telemetry with distributed traces or other signals. Adding this to OpenTelemetry would allow analysis solutions and end-users to find instances of poor performance between services and then immediately chase these down to their root cause within code.

Production Debugging (proposed)

While less common than production profiling solutions, production debugging (the creation of non-blocking breakpoints or log generation points in running services) could be a valuable addition to OpenTelemetry. However this seems less immediately useful than the other items listed above.

Capturing Environment Data

Can we get OTel components to capture environment variables or other environment data?

Stabilizing Semantic Conventions (in-progress)

- We can't release stable instrumentation because the semantic conventions aren't stabilized
- Currently being worked on for HTTP and messaging
- May also involve stabilizing resource conventions

Self-Contained Projects

Demo (in-progress)

OpenTelemetry launched a community demo SIG in May 2022, which will provide sample applications that demonstrate OpenTelemetry's capabilities to prospective end-users, and also allow the community to better perform automated testing of OpenTelemetry components. This SIG will accelerate their work by starting with existing cloud demo applications like Google's Hipstershop.

eBPF (in-progress?)

Could / should expand into a community-wide project.

More Instrumentation (in-progress within each SIG)

As a project, we will continually add to and update our cohort of instrumentation for each language and for the Collector.

Other Areas of Investment

Completing Existing Spec Implementations

Guidance for New Project Contributors

Many SIGs have great onboarding guides, but some new contributors just want to help out the project in the best way possible, and are willing to work on ~any SIG in any capacity (developers, doc writers, etc.). We currently don't have any guidance for them.

Getting More Maintainers

OpenTelemetry's SIG maintainers have all made a massive commitment to the project, and while they are recognized for this, we should find ways to reduce their burden.

One of the most apparent ways to do this is to promote more maintainers within the project, which would allow existing SIG maintainers to spread the load. To achieve this, we should (a) recruit more contributors into OpenTelemetry (as some of them will want to become maintainers), and (b) build a more clear and easy to follow path to maintainership.

Better Managing 'Contrib' Contributions

Related to the above, SIG maintainers have consistently labeled managing instrumentation and exporter contributions (usually stored in each SIG's 'contrib' repository) as a massive but necessary time-sink. While some work has already been done to alleviate this, more is still necessary.

One of the discussed proposals is to use folder-based permissions in GitHub to allow contributors who lack maintainer or approver permissions to maintain their own contributions. For example, a vendor could maintain the folder containing their exporter within the Collector-Contrib repository, meaning that the project maintainers would only have to interact with these projects if they break a build.

Promoting OpenTelemetry's APIs and OTLP to Telemetry Source Developers

OpenTelemetry provides significant value to thousands of organizations around the world. While much of this value comes from the features and functionality that the community has built, all of this would be useless without OpenTelemetry's components ability to integrate with various telemetry sources (web frameworks, client libraries, databases, etc.) out of the box.

Today, many of these integrations are provided by contributions to OpenTelemetry's SIG-specific contrib repositories that call out to telemetry source interfaces, capture data and convert it to an OpenTelemetry-compliant format, and then submit these to OpenTelemetry's APIs (either in-process or via OTLP). This instrumentation is functional, but incurs a large development and maintenance burden on the project.

The most ideal integrations occur when telemetry sources themselves use the OpenTelemery APIs or OTLP to generate telemetry. Some of these exist already, however we need to do work as a community to promote the usage of the OpenTelemetry APIs and OTLP amongst the maintainers of these telemetry sources. We haven't made a concerted effort for this thus far, but with the metrics APIs and OTLP declared GA, we can commence this immediately.

Promoting OTLP to Analysis System Developers

Improving Operationalization, Project Confidence, etc.

Prioritization

Votes at Kubecon	**Votes on maintainers call**	**Project**
xxxxxxxxxxxxxxxxxxxxx	xxxxx	Logging
xxxxxxx	xxx	Client instrumentation
xxxxxxxxx		Improving the maintainer experience
xxxxxxxxxxxxx	xxx	Control plane
xxxxxxxxxxxxxxxxxx	x	Profiling

~~xxxxx~~	~~x~~	~~Promoting APIs and OTLP~~
~~xxxxx~~		~~Capturing environment data~~
~~xxxxxxxxxx~~	~~x~~	~~Demo~~
~~x~~		~~eBPF~~
~~xxx~~		~~Promoting OTLP to analysis system providers~~
~~xxxxxxxxxxxxxxxxxx~~	~~xxxxxxxx~~	~~Improving operationalization, documentation, project confidence, etc.~~
	~~xxxxxxxxxxxx~~	~~Stabilizing Semantic Conventions~~