[TAG Observability] Due Diligence Project Review: Open Metrics Incubation

Authors: Richard “RichiH” Hartmann, Ben Kochie, Brian Brazil, Rob Skillington
Reviewed by: Bartek Plotka (SIG Observability), Michael Hausenblas (SIG Observability) Steve Flanders (Splunk), Matt Young (SIG Observability), Kemal Akkoyun
TOC sponsor: Alena Prokharchyk, Cornelia Davis
Ticket: https://github.com/cncf/sig-observability/issues/30

Goals of this Document

This document provides a technical review of the OpenMetrics project in a form of Due Diligence described here. The main goal is to provide factual data that would help TOC to decide if OpenMetrics should be promoted to the Incubation stage.

Addendum for TOC, 2021-04

- YouTube recordings: https://www.youtube.com/playlist?list=PLoz-W_CUquUmYtI8L_eeqMwRkOvxAaFY1
- OpenMetrics ID: https://tools.ietf.org/html/draft-richih-opsawg-openmetrics

Status

- 2020-11-10: Document been shared with OpenMetrics team
- 2020-11-17: Filled out by OM team
- 2020-11-24: Shared with SIG Observability on cncf-sig-observability@lists.cncf.io
- 2020-11-24: All-hands Review during SIG Observability Meeting
- 2020-12-01: Question period from first review over, replies by OpenMetrics added as comments & suggestions
- 2020-12-08: Continued review by SIG, walkthrough of all questions, comments, and suggestions
- 2020-12-08: Approved in full by CNCF SIG Observability
- 2021-07-06: Added TOC questions & replies

2021-07 Questions TOC & replies

Updates over the last six months:

Recent focus is on extending our test suite with different test methods, in particular the Prometheus Conformance program which GB should approve this month and which will pull OM and its test suite in as a dependency. Other more janitorial work for now until we know where to go with high-resolution histograms

Work in the open after 1.0

All docs & meetings are now open. We are working mostly through PRs at the moment, and all the calls we had are public to join and uploaded to YouTube (on the Prometheus channel for now, until CNCF gets more YT accounts going which has cell phone number restrictions)

Make Markdown the primary format, not Internet Draft

The canonical document in Git was, is, and will remain in Markdown https://github.com/OpenObservability/OpenMetrics/blob/main/specification/OpenMetrics.md with minimal IETF syntactic sugar on it so it can be compiled into nroff and the other legacy formats we need for the pipeline. This is also the doc linked from our homepage, but if we can improve linking to it, we will happily do it

Meeting cadence

Until 1.0 we were on a bi-weekly call schedule, it's currently "last Friday of the month, if required" which we didn't need often as we're now working in PRs. Meetings are also announced on the mailing list.

Communication channels

We have #openmetrics on CNCF Slack, an IRC channel, and a mailing list

Update README with communication channels

We link to mailing list and Twitter, but will add links to all other communication channels as well.

Statement of CNCF Alignment to TOC Principles

1. Project is self-governing

Yes. See https://github.com/OpenObservability/OpenObservability/blob/main/GOVERNANCE.md

Comments:

- TODO: Request for more time to revisit
    - Extended by two weeks
- Where are the meetings are those opens?
    - Public meetings going forward

Fortnight call, intention to be public

- Is there application to IETF RFC discussion public?
    - Yes, IETF Ops WG list

There will be, but no meetings yet

- “It is copied from Prometheus”: https://prometheus.io/governance/ with the changes
    - Cleaner delineation of repositories within the GitHub org

    - IETF rough consensus instead of lazy consensus
    - Project lead

SIG O11y is happy with the section above

2. Is there a documented Code of Conduct that adheres to the CNCF guidelines?

Yes. https://github.com/cncf/foundation/blob/master/code-of-conduct.md

SIG O11y is happy with the section above

3. Does the project have production deployments that are high quality and high-velocity? (for incubation and graduated projects).

Yes. Two of the most used Prometheus client libraries support OpenMetrics. Python since October 2018 and Go since January 2020. Java, Ruby[1], Perl, and PHP client libraries are implementing support. Prometheus server and DataDog have been supporting OM since 2018, and Prometheus has been negotiating OM preferentially since 2018.

Exemplar support is widely used within Grafana Labs and Chronosphere.

SIG O11y is happy with the section above

4. Is the project committed to achieving the CNCF principles and do they have a committed roadmap to address any areas of concern raised by the community?

Yes, it is committed. There is no roadmap as such, but that's in the nature of a standard.

A sister project to specify a wire format for logs is next and will happen under the umbrella of the OpenObservability GitHub org. We are not aware of any concerns by the community.

Addressing concerns from last meeting:

- We commit to working with all CNCF projects in a constructive manner. In particular, quote Steve Flanders "helping address gaps, working to make it default format for Otel, etc."
    - The same is true for future work on a label-based logging wire format, but this is outside the scope of OpenMetrics
    - Seven members of Prometheus-team (three of them working on OpenMetrics) joined the last OTel Metrics WG call, but the meeting didn't happen
    - 2020-12-02 there's a call between OTel & OM
- Semantic conventions are self-contained within OpenMetrics.
- OpenMetrics 2.0 will most likely solve the problem of efficient high-resolution histograms

SIG O11y is happy with the section above

[1] https://github.com/prometheus/client_ruby/issues/189

5. Document that the project has a fundamentally sound design without obvious critical compromises that will inhibit potential widespread adoption.

As this project is formalizing what Prometheus does, there is substantial adoption as of today.
Comment:

- Steve: Open Telemetry would come up and if incompatibility is there would that not inhibit potential widespread adoption?
- Richi: Are there compatibility issues?
- Ben: I have already lib that has all signals
- Rob: This might Inhibit OpenTelemetry widespread adoption if OM are not supported
    - I cannot recommend using OT until it supports it? You cannot make this work with major Prometheus based vendors.
    - It can be resolved pretty easily: Make OT being compatible with OM
- Steve: Inhibit one project or another, still concern
- Richi: Fundamental goal or being compatible with other things. OM is key part for supporting Cortex, Thanos, Prometheus. Minor timestamp
- Steve: There is commitment to working with OpenTel and vice versa
- Richi: Agree

SIG O11y is happy with the section above

6. Document that the project is useful for cloud native deployments & degree that it's architected in a cloud native style.

OM is a truly cloud-native wire format for metrics. It came out number five in the CNCF Observability End User Survey and was put in the Adopt category.

SIG O11y is happy with the section above

7. Document that the project has an affinity for how CNCF operates and understand the expectation of being a CNCF project.

Three of the core OM maintainers are Prometheus maintainers, Prometheus being the second project to join the CNCF. One of them is a SIG o11y chair.

SIG O11y is happy with the section above

Review of graduation criteria and desired cloud native properties

From Sandbox Graduation (Exit Requirements)

1. Document that it is being used successfully in production by at least three independent end users which with focus on adequate quality and scope defined.

See CNCF End User Survey: https://www.cncf.io/blog/2020/09/11/cncf-end-user-technology-radar-observability-september-2020/ Examples include GitLab, SoundCloud, and EverQuote: https://github.com/OpenObservability/OpenMetrics/blob/master/ADOPTERS.md

SIG O11y is happy with the section above

2. Have a healthy number of committers. A committer is defined as someone with the commit bit; i.e., someone who can accept contributions to some or all of the project.

The core group of active maintainers consists of four people who have been participating in fortnightly calls for years now. This commitment remained even while working for a total of seven different companies.

- Ben Kochie, SoundCloud/GitLab, since 2015
- Brian Brazil, RobustPerception, since 2015
- RichiH, SpaceNet/Grafana Labs, since 2015
- Rob Skillington, Uber/Chronosphere, since 2017

SIG O11y is happy with the section above

3. Demonstrate a substantial ongoing flow of commits and merged contributions.

The spec was largely worked on within a Google Doc which are not caught by devstats. We make a hard commitment to keeping development in public going forward. Re-onboarding previous maintainers even after half a year proved next to impossible so we were forced to remove public exposure until the spec settled down.

Code contributions were largely done in the Prometheus org as that is where the implementations live. DataDog deserves special mention for contributing improvements to upstream through GitHub.

An example of our workflow can be seen at https://github.com/OpenObservability/OpenMetrics/issues/129#issuecomment-476800346 which is modelled upon Prometheus dev summit notetaking.

CNCF devstats: https://openmetrics.devstats.cncf.io/

SIG O11y is happy with the section above

Documentation of CNCF Alignment (if not addressed above):

*** name of project (must be unique within CNCF):** OpenMetrics

*** project description (what it does, why it is valuable, origin and history):** OpenMetrics creates an open standard for transmitting cloud-native metrics at scale, with support for both

text representation and Protocol Buffers. It is the de facto standard in cloud-native metrics transmission.

SIG O11y is happy with the section above

*** statement on alignment with CNCF charter mission:** We are 100% aligned.
*** sponsor from TOC (sponsor helps mentor projects): TBD/TODO**
*** license (charter dictates Apache 2 by default)**
Apache 2
*** source control (GitHub by default)**
https://github.com/OpenObservability/OpenMetrics
*** external dependencies (including licenses)**
This question does not make sense in the context of a spec. We do depend on several IETF RFCs and the Protobuf specification, all of which can be implemented freely.

SIG O11y is happy with the section above

*** release methodology and mechanics**
Being a standard, we are moving slowly and deliberately. Any changes are considered closely with Prometheus, Cortex, Thanos, and Kubernetes -- all of them CNCF projects.
Additionally, we follow the IETF RFC process, including BCP and errata.

SIG O11y is happy with the section above

*** community size and any existing sponsorship**

Hard to judge, but at the very least in the hundreds of thousands of users. Through client_golang, every Kubernetes user is also an OpenMetrics user. Also see https://www.cncf.io/blog/2020/09/11/cncf-end-user-technology-radar-observability-september-2020

SIG O11y is happy with the section above

* **An architectural, design and feature overview should be available.**

OpenMetrics supports transmitting metrics over both push and pull over webhooks (in particular over HTTP), with eight metric types that cover use cases that have been seen to arise over the past decades. It supports Exemplars to connect metrics to logs and traces, histograms for aggregatable quantiles, exposing enumerations, and metadata. Exposers must at the least support requesting metrics over HTTP, preferably over TLS.

Python reference implementation:
https://github.com/prometheus/client_python/tree/master/prometheus_client/openmetrics

SIG O11y is happy with the section above

- **What are the primary target cloud-native use cases? Which of those:**
    - **Can be accomplished now:** All cloud-native metrics use cases are covered
    - **Can be accomplished with reasonable additional effort (and are ideally already on the project roadmap):** IETF RFC release
    - **Are in-scope but beyond the current roadmap for the next six months:** More detailed and efficient histograms, once finalized by Prometheus-team
    - **Are out of scope:** Logging and tracing are out of scope for this metrics wire format, but collaboration across all pillars is in our interest. Also see https://github.com/OpenObservability/OpenMetrics/blob/master/specification/OpenMetrics.md#scope

SIG O11y is happy with the section above

What exactly are the failure modes? Are they well understood? Have they been tested? Do they form part of continuous integration testing? Are they appropriate given the intended usage (e.g. cluster-wide shared services need to fail gracefully etc)?
The main failure mode of a spec is when it is not followed. The specification is written following RFC 2119, BCP 14, and RFC 8174 in using MUST/MUST NOT, SHOULD/SHOULD NOT, and MAY. We have a test suite covering all features and RFC spec for independent and maintainer-led testing at https://github.com/OpenObservability/OpenMetrics/tree/master/tests

For extensions and improvements, please see the “Extensions and Improvements” section in the specification: https://github.com/OpenObservability/OpenMetrics/blob/master/specification/OpenMetrics.md#extensions-and-improvements

We are following semantic versioning in spirit where it makes sense for a wire format.

SIG O11y is happy with the section above

What trade-offs have been made regarding performance, scalability, complexity, reliability, security etc? Are these trade-offs explicit or implicit? Why? Are they appropriate given the intended usage? Are they user-tunable?
To ensure interoperability, we mandate that every implementation MUST support pull with the text format. They MAY also support push and/or protobuf to account for operational considerations, such as performance, relevant to their environment.

The wire format has seen massive testing through Prometheus, Cortex, Thanos, Chronosphere, DataDog, InfluxData, etc.

Trade-offs are explicit and explained in the Considerations section of spec.
We do believe that our trade-offs are reasonable for modern cloud-native deployments and as it's a wire format, users are free to adapt their metrics usage as they see fit.

SIG O11y is happy with the section above

What are the most important holes? No HA? No flow control? Inadequate integration points?

We are not aware of any important holes.

SIG O11y is happy with the section above

Code quality. Does it look good, bad or mediocre to you (based on a spot review). How thorough are the code reviews? Substance over form. Are there explicit coding guidelines for the project?

It does not strictly apply as it's spec. We do believe that in between our extensive operational experience, experience maintaining a wide variety of projects, and by following RFCs 2119, BCP 14, and RFC 8174 we have achieved a very high quality. Additionally, IETF will apply feedback through the Ops WG and dedicated, professional tech writer resources before publishing the RFC.

SIG O11y is happy with the section above

Dependencies. What external dependencies exist, do they seem justified?
See above.
Outside of RFCs and the protobuf specification, we do not have dependencies.

SIG O11y is happy with the section above

What is the release model? Versioning scheme? Evidence of stability or otherwise of past stable released versions?
See above, in particular SemVer, the CNCF End User Survey, the CNCF graduated and incubating projects, and Prometheus exposition format 0.0.4 and OpenMetrics 1.0.0.

SIG O11y is happy with the section above

What is the CI/CD status? Do explicit code coverage metrics exist? If not, what is the subjective adequacy of automated testing? Do different levels of tests exist (e.g. unit, integration, interface, end-to-end), or is there only partial coverage in this regard? Why?

All implementations we are currently aware of, reference or otherwise, have CI and integration testing as they are part of a graduated CNCF project which mandates this anyway.

In addition, there is a implementation compliance suite
https://github.com/OpenObservability/OpenMetrics/tree/master/tests

The Python reference implementation has 99% branch coverage and 100% line coverage.

SIG O11y is happy with the section above

What licensing restrictions apply? Again, CNCF staff will handle the full legal due diligence.
Apache 2

What are the recommended operational models? Specifically, how is it operated in a cloud-native environment, such as on Kubernetes?
Deploy any modern, cloud-native metrics ingestor and use properly instrumented code, exporters, or an agent.

Project

- **Do we believe this is a growing, thriving project with committed contributors?**
  Yes
- **Is it aligned with CNCF's values and mission?**
  Yes
- **Do we believe it could eventually meet the graduation criteria?**
  Yes
- **Should it start at the sandbox level or incubation level?**
  Incubation or graduated.
- **Does the project have a sound, documented process for source control, issue tracking, release management etc.**
  Yes.
- **Does it have a documented process for adding committers?**
  Yes, see governance.
- **Does it have a documented governance model of any kind?**
  Yes, see
https://github.com/OpenObservability/OpenObservability/blob/main/GOVERNANCE.md
- **Does it have committers from multiple organizations?**
  Yes.

- **Does it have a code of conduct?**
  Yes.

- **Does it have a license? Which one? Does it have a CLA or DCO? Are the licenses of its dependencies compatible with their usage and CNCF policies? CNCF staff will handle the full legal due diligence.**
  Yes.

- **What is the general quality of informal communication around the project (slack, github issues, PR reviews, technical blog posts, etc)?**
  Good.

- **How much time does the core team commit to the project?**
  Since 2015, we have invested thousands of people hours.

- **How big is the team? Who funds them? Why? How much? For how long?**
  Four core maintainers. Employed by four different companies, but substantial parts were worked on during our own free time.

- **Who are the clear leaders? Are there any areas lacking clear leadership? Testing? Release? Documentation? These roles sometimes go unfilled.**
  It's a group model, with SABDFL.

- **Besides the core team, how active is the surrounding community? Bug reports? Assistance to newcomers? Blog posts etc.**
  See CNCF End User Survey.

- **Do they make it easy to contribute to the project? If not, what are the main obstacles?**
  See above, for sanity, there was reduced publicity until release. Release has happened. Coordination with Prometheus maintainers, Kubernetes SIG Instrumentation has been continuous.

- **Are there any especially difficult personalities to deal with? How is this done? Is it a problem?**
  OM is modelled upon how Prometheus works

- **What is the rate of ongoing contributions to the project (typically in the form of merged commits).**
  See above. While it is impossible to get a complete overview due to the nature of a wire format, Google docs, etc. Taking the metric of accepted suggestions and discussion items, the total is thousands of "merges" over the years.

SIG O11y is happy with the section above

Users

- **Who uses the project? Get a few in-depth references from 2-4 of them who actually know and understand it.**
  This question is not 100% applicable to a specification. See CNCF End User Survey.

- **What do real users consider to be its strengths and weaknesses? Any concrete examples of these?**
  100% compatibility with Prometheus, Cortex, Thanos, and Kubernetes; plus various non-CNCF projects and products.

- **Perception vs Reality: Is there lots of buzz, but the software is flaky/untested/unused? Does it have a bad reputation for some flaw that has already been addressed?**
  There is substantial buzz even during the development phase, again see PromCorThanos etc for reliability. The recent release saw tens of thousands of engagements within days.

SIG O11y is happy with the section above

Context

- **What is the origin and history of the project?**
  Initially started by RichiH as a vehicle to force network vendors to support cloud-native metrics as part of a holistic approach to both infrastructure and services, it has gotten support from a wide variety of vendors and companies over the years.
  It has always been in close alignment with Prometheus, Cortex, Thanos, Kuberenetes, and non-CNCF projects and products.

- **Where does it fit in the market and technical ecosystem?**
  It is the lingua franca of cloud-native metrics.

- **Is it growing or shrinking in that space? Is that space growing or shrinking?**
  It's growing, and CNCF End User Survey put it in spot five.

- **How necessary is it? What do people who don't use this project do? Why exactly is that not adequate, and in what situations?**
  It's a hard requirement of cloud-native metrics monitoring within CNCF. If not using OpenMetrics, the only alternative is Prometheus exposition format 0.0.4, the predecessor of OpenMetrics. The creation of this project has been requested by Dan Kohn and Chris A. a few years back.

- **Clearly compare and contrast with peers in this space. A summary matrix often helps. Beware of comparisons that are too superficial to be useful, or might have been manipulated so as to favor some projects over others. Most balanced comparisons will include both strengths and weaknesses, require significant detailed research, and usually there is no hands-down winner. Be suspicious if there appears to be one.**

The only two alternatives in this space are OpenMetrics or Prometheus exposition format 0.0.4. As the Prometheus maintainers committed to deprecating 0.0.4 in favor of OpenMetrics, it is literally the only cloud-native metric wire format in existence.

OpenTelemetry intends to support OpenMetrics as a first class wire format.

Non-cloud-native metric monitoring wire format examples include:

SNMP: An older standard with some similarities (e.g. counters and gauges, tables are a form of label) with is near ubiquitous support in the networking space. It is high overhead to implement new metrics as they must be part of the global OID registry maintained by IANA. Out-of-band MIBs are used by SNMP, whereas OM uses in-band metadata. Out-of-band can be more efficient due to fewer bytes on the wire, whereas in-band is easier to work with as you don't have to seek out the MIBs. A hard requirement on ASN.1 causes substantial implementer pain and introduces non-obvious vulnerabilities.

Statsd: The statsd line protocol works primarily with events rather than metrics, to put it one way the statsd protocol would talk to something that would do temporal aggregation which could then expose OM. The statsd format can be simpler to get up and running, however it does not support key-value pairs, metadata, and support for non-integer values varies. Given that it is events rather than metrics, bandwidth used is directly proportional to traffic that a process receives which can lead to losing data under high load.

SIG O11y is happy with the section above