What You Need to Know About OpenMetrics

Richard “RichiH” Hartmann

Grafana Labs

Before Prometheus

- Historically, the closest to a standard is SNMP
- Many solutions based on ancient technology like ASN.1
- Often chatty and slow
- Many data formats are proprietary, hard to implement, or both
- Data models encourage per-vendor variations that follow the letter of the law, but not the spirit
- Hierarchical data models almost never fit a user's needs

After Prometheus

- Prometheus is the de-facto standard in cloud-native metric monitoring and beyond
  - Same is true for Prometheus exposition format
- Ease of exposing data has lead to an explosion in compatible metrics endpoints
  - Thousands of exporters and integrations
  - 100,000s of installations, 1,000,000s of user
- Standard exporters and libraries make integrating this easy
  - Exporter scaffold so you can focus on metrics, not HTTP endpoints
- Labelsets allow near-perfect access to your data

Politics

- Some other projects & vendors are torn about adopting something from a competing product or project
- Especially traditional vendors prefer to support official standards
- Re-use installed base of Prometheus
  - Ease of adoption
  - Reject kitchen sink approach, do one thing well, and remain focused and opinionated
- Many competing companies collaborated on OpenMetrics and helped shape it
- The result is an actually neutral standard

People & process

- Dozens of people and companies helped us get to where we are
- The marathon runners
  - Ben Kochie, GitLab
  - Brian Brazil, Robust Perception
  - Richard Hartmann, Grafana Labs
  - Rob Skillington, Chronosphere
  - Honorable mention: Sumeer Bhola

OpenMetrics & You

- The format is largely the same as the Prometheus format, with cleanups, and new features
- If you are using the Python client integration and Prometheus 2.5.0+, you have used OpenMetrics since late 2018
  - Other first-party libraries followed later
- Using our official client libraries migrates you transparently to OpenMetrics without you noticing
- By default things Just Work(™)

Breaking Changes

- Counters now require _total on the time series
  - Common convention, but now enforced
  - If your metric was `cpu_seconds`, our libraries will migrate you to `cpu_seconds_total`
- Timestamps are in seconds, for consistency
  - We use base units everywhere else
  - Used to be milliseconds
  - Exposing an explicit timestamp is possible, but usually an antipattern

Improvements & interoperability

- Cleaner and tighter specification, e.g. spacing, escaping
- Explicit EOF to detect incomplete scrapes
- Allowing for nanosecond resolution timestamps
- 64-bit integer values
- Unit as new metadata
- _created for metric creation & resets
- Considerations for both pull and push
- Text format still mandatory, reintroduce optional protobuf

What's New: Exemplars

Exemplars allow linking certain metrics to example traces:

```
# TYPE foo histogram
foo_bucket{le="0.01"} 0
foo_bucket{le="0.1"} 8 # {id="abc"} 0.043
foo_bucket{le="1"} 10 # {id="def"} 0.29
foo_bucket{le="10"} 17 # {id="ghi"} 7.73
foo_bucket{le="+Inf"} 18
foo_count 18
foo_sum 219.3
foo_created 1520430000.123
```

Already in Prometheus, Cortex, Thanos, Loki, Grafana

Current Status: Prometheus

- Prometheus Python client is the reference implementation, and uses the OpenMetrics data model internally
- Go & Java support
- Prometheus will negotiate preferentially for OpenMetrics
- Info & Enum are now first class features
  - No need to implement them by hand
  - Automatically exposed backwards-compatible if scraped via the Prometheus text format

Current Status: Other Implementations

- DataDog supports ingestion of OpenMetrics, and contributed performance improvements to the Python parser
- OpenTelemetry support OpenMetrics as a first-class wire format
  - We had some incompatibilities with the standard
  - OTel founded a WG Prometheus
  - Everything works together, now
- OpenMetrics is part of the Prometheus Conformance Program

Prometheus Conformance Program

- Officially launched on October 14th
- Several test suites for Prometheus interfaces
- “Prometheus Compatible” mark to designate compatible implementations
- Scrapers need 100% OpenMetrics compliance
- More details: Prometheus Conformance Program @ KubeCon

Spotting OpenMetrics

```
Prometheus exposition format 0.0.4:
# TYPE foo_seconds_total counter
foo_seconds_total 1.0

OpenMetrics 1.0 (including optional UNIT and _created):
# TYPE foo_seconds counter
# UNIT foo_seconds seconds
foo_seconds_total 1.0
foo_seconds_created 1572628096.0
# EOF
```

Transitioning to OpenMetrics by hand

- Add in _total now for counters, so you can control that change
  - Otherwise this should be a noop for those using existing client libraries.
- Client library & custom integration authors: ensure you're sending an appropriate Content-Type if you plan on continuing to expose Prometheus text format
- Scraper & ingester authors set Accept header to negotiate the Prometheus and/or OpenMetrics format as needed

Known issues in 1.0

- Few, but consistent reports of name collisions with _total
- _started doubles the cardinality of counters
- One line in specification clashes with the EBNF
  - EBNF authoritative as per specification, so no real impact, but annoying

-> These will be fixed in 1.1

OpenMetrics 2.0

- High resolution / native histograms
  - Which means complex data types
- Potentially move _created to be complex data types or metadata
- Fold into Prometheus?
  - 2017/2018, CNCF asked Prometheus team spin off OpenMetrics
  - OpenMetrics arguably achieved its goal: create a reference standard
  - With OpenTelemetry gaining adoption, there is another standards driver

Resources

- https://github.com/OpenObservability/OpenMetrics/
- https://github.com/prometheus/compliance
- https://github.com/OpenObservability/OpenMetrics/tree/main/tests

Thank You!

richih@richih.org
twitter.com/TwitchiH
github.com/RichiH/talks