Engineer

Metrics

numbers a system reports about itself

What it is

Metrics are measurements of a running service, each taken at a moment and kept as a value with a timestamp and a few labels, so that the same quantity can be read again a minute, a day or a month later. One named quantity is a metric; stored as a sequence of values over time it becomes a time series; and OpenTelemetry counts metrics as one of its signals, beside traces and logs. Requests served, memory in use and jobs waiting in a queue are typical metrics.

What sets metrics apart from the other signals is aggregation. A log keeps one line per event, and a trace follows one request through the system. A metric folds many measurements into a few statistics over a window of time: a count, a sum, a current value, a spread across buckets. The OpenTelemetry documentation calls the result statistical information in aggregate, and that is both the strength and the limit: a metric says how many requests failed in the last minute, and a log says which ones.

OpenTelemetry and Prometheus agree on a small set of shapes, and choosing the right one is most of the design:

- A **counter** accumulates and only goes up, like the odometer in a car: requests served, bytes sent, errors seen. Prometheus allows one exception, a reset to zero when the process restarts.
- An **up-down counter**, in OpenTelemetry's terms, accumulates but can also go down, such as the length of a queue as work arrives and leaves.
- A **gauge** is a current value read at the moment it is read, like the fuel gauge in the same car: memory in use, a temperature, requests in flight.
- A **histogram** sorts measurements into buckets and keeps their count and sum, so it can answer how many requests took less than a second.

Prometheus also has a summary, which computes quantiles inside the program. The OpenTelemetry data model keeps it only as a legacy type, because summaries cannot always be merged meaningfully.

The text format that Prometheus scrapes shows the pieces together: a help line, a type line, then one line per label combination with the name, the labels, the value and an optional timestamp in milliseconds since the epoch.

```
# HELP http_requests_total The total number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{method="post",code="200"} 1027 1395066363000
http_requests_total{method="post",code="400"} 3 1395066363000
```

Use the concept in a decision

Put “What are metrics?” to work in a real decision. We can help relate the concept to the system you are building or buying. Monitoring and alerts that speak up while a fault is still small, and the figures that show how often it stays that way.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Built the company's first observability dashboard — the company's first view of its own systems

Built the company's first observability dashboard Get in touch →

Why it matters

Metrics are cheap enough to keep for everything. The memory they need grows with the number of distinct label combinations, not with traffic; the OpenTelemetry documentation says so in as many words, in contrast with logs. A counter incremented a billion times is still one number. That is what lets a team count every request rather than a sample of them, and keep the counts long enough to compare this month with the last.

They turn arguments into measurements. Whether a host is big enough, whether a release made answers slower, how soon a disk fills: each has an answer in a series recorded before anyone asked. Alerts are built on the same numbers, since an alert rule is a condition over a metric that someone decided was worth waking for.

They also outlive the code that emits them, so the shape is worth getting right. The Prometheus naming guide asks for one unit and one quantity per metric, in base units such as seconds and bytes rather than milliseconds or megabytes, with the unit as a plural suffix and `_total` on an accumulating count. Names such as `http_request_duration_seconds` and `process_cpu_seconds_total` then say on their face what they hold.

How totals travel matters as well. Prometheus reports cumulative totals, so gaps from an intermittent collection failure average out, at the price of memory in the sender in proportion to its label combinations. StatsD reports deltas, the change since the last report, which moves that cost out of the process.

What it looks like in practice

The company's own public-services host is a single-core machine with 512 MB of memory, of which about 464 MB is usable, and on it memory became the argument that settled decisions. Reporting there is a pull: seven commands that read the host and render Markdown, change nothing and run only when asked. When the question came up whether that one 512 MB host could also carry an onion service, the answer came from a recorded day of measured samples rather than an opinion: available memory never fell below about 310 MB of 464, swap sat at 2.6 per cent and the processor was 99.7 per cent idle.

Those reports later had to say what their numbers were worth. Six read-only collectors were made to state what they never read once it was clear that a blind collection and a healthy host rendered the same page: a mirror whose daemon changed its log format produced the same zero as a mirror nobody visited. Thresholds were fitted to the 116 samples the metrics report had already measured, with the observed ranges recorded beside the bounds, and the report tells three outcomes apart: a breach, no breach, and not judged because nothing measured it. Each collector reports its runtime and its blindness, such as files read against files unreadable, even when the answer is zero. The bounds are daily averages, which can hide a one-minute spike. The fifteen-minute switch checks available memory and free disk at each run, but a brief breach between its checks can still go unseen.

The most useful metrics often come from the problem rather than the host. The note on selling limited stock without overselling puts lock waits, expired holds and the distance between the gate's count and the database's on one dashboard,

and runs the invariant `available + held + sold = total` on a schedule, charted and paged on at any drift. Each of those numbers comes from knowing how that sale goes wrong.

Where it goes wrong

Metrics rarely break loudly. The graph keeps drawing, and the number on it quietly means something other than what its name says.

A value that falls is published as a counter. The Prometheus metric types page says not to use a counter for anything that can decrease, such as the number of running processes: that is a gauge. The query functions treat any fall in a counter as a restart and correct for it, so a value that falls on purpose shows up as increases that never happened.

Units drift between services. One service reports a duration in milliseconds, another in seconds, and a panel that compares them is off by a factor of a thousand. The naming guide says a metric must refer to a single unit and should use base units. Converting once, at the source, is cheaper than remembering the difference in every query.

A sum across labels means nothing. The naming guide's rule of thumb is that the sum or the average over all of a metric's labels should be meaningful. Its own example: the capacities of several queues belong together in one metric, while a queue's capacity and its current length do not.

Unmeasured reads as zero. A collector that has stopped reading its input leaves a zero, or a missing row, that looks exactly like a quiet system. The OpenTelemetry data model carries a flag for a point with no recorded value, its counterpart to the Prometheus staleness marker, because missing data and a value of zero are different facts. Report them differently, and give "not judged" its own outcome, as the collectors above do.

The window hides the spike. A gauge is a sample at a moment, and the OpenTelemetry data model reports only the last value in each interval, so a burst between two samples leaves no trace. A daily average smooths a minute at the ceiling away in the same way. Choose the window from the failure that has to be seen, and keep a fast check beside a slow one when both matter.

Overflow undercounts a filtered query. The OpenTelemetry SDK tracks at most 2,000 distinct attribute combinations per metric stream by default, and folds any further ones into a single data point marked `otel.metric.overflow=true`. The total stays correct, but overflow replaces the whole combination, not just its noisy part. In the documentation's own example, a request counter carries `url.path` and `success`; once it overflows, a query for `success=false` undercounts, and an error-rate alert built on it can stop firing. Keep attributes to small, known sets, and watch for the overflow marker.

Further reading

- OpenTelemetry: Metrics — instrument kinds, aggregation and the cardinality limit.
- OpenTelemetry: Metrics data model — gauges, sums, histograms, temporality and the no-recorded-value flag.
- Prometheus: Metric types — counter, gauge, histogram and summary.
- Prometheus: Metric and label naming — base units, unit suffixes and `_total`.
- Prometheus: Histograms and summaries — where quantiles are computed, and which ones can be combined.
- Prometheus: Exposition formats — the text format line by line.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What are metrics?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/metrics/