

OTel/Prometheus Interoperability Survey

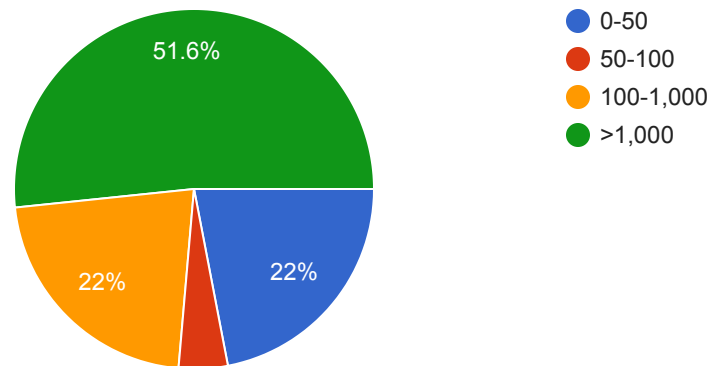
91 responses

[Publish analytics](#)

How large is your organization?

 [Copy](#)

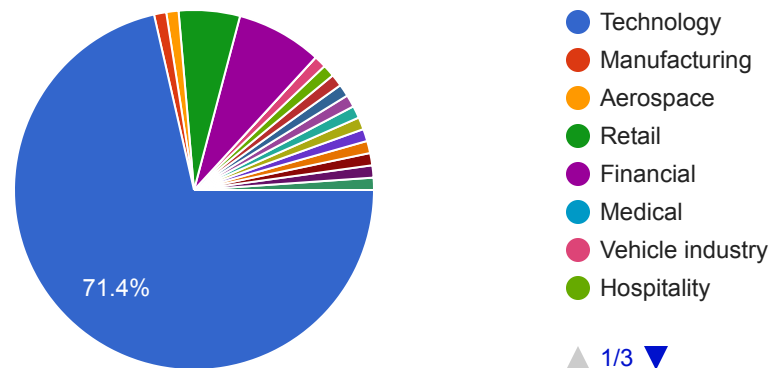
91 responses



What industry do you work in?

 [Copy](#)

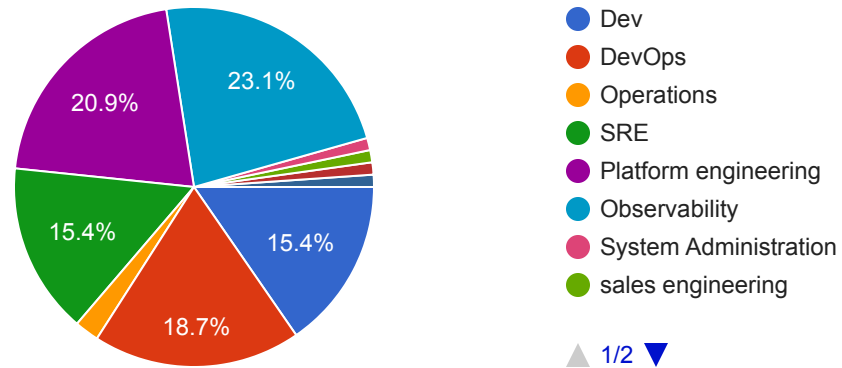
91 responses



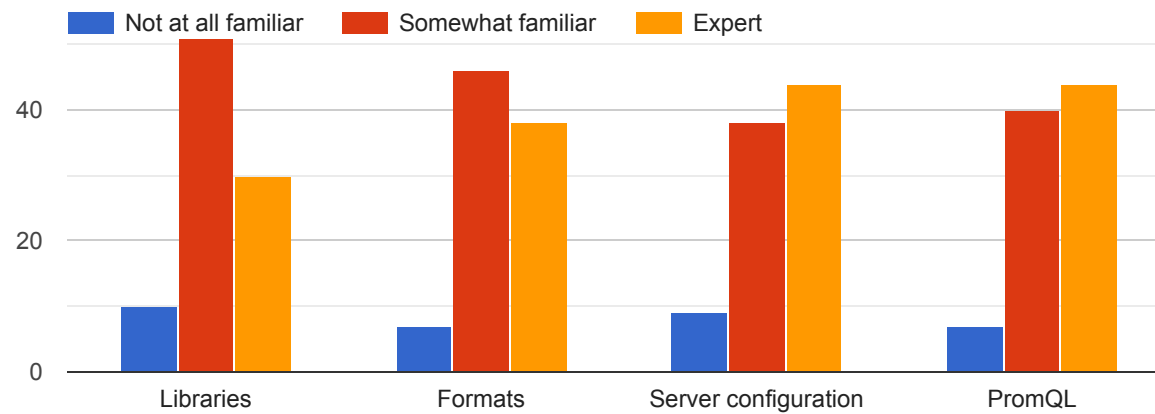
What type of team do to work on?

 Copy

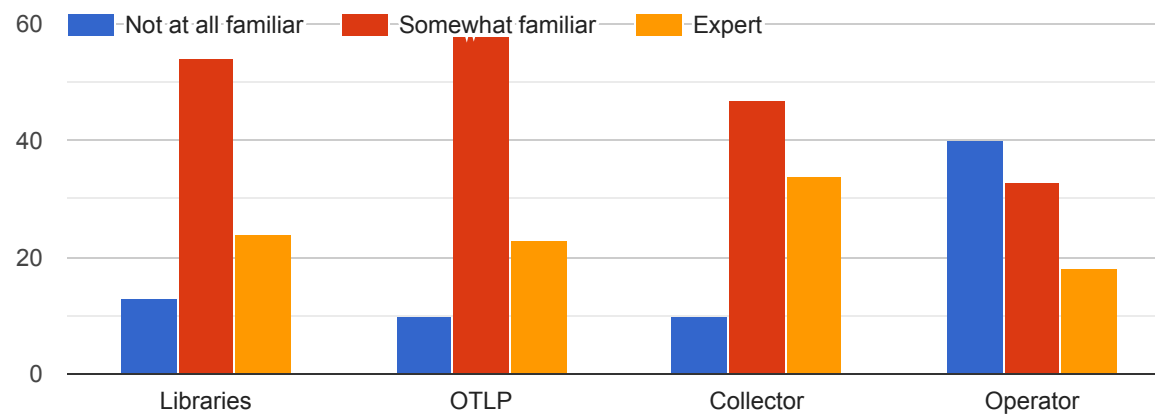
91 responses



How well do you understand Prometheus?

 Copy

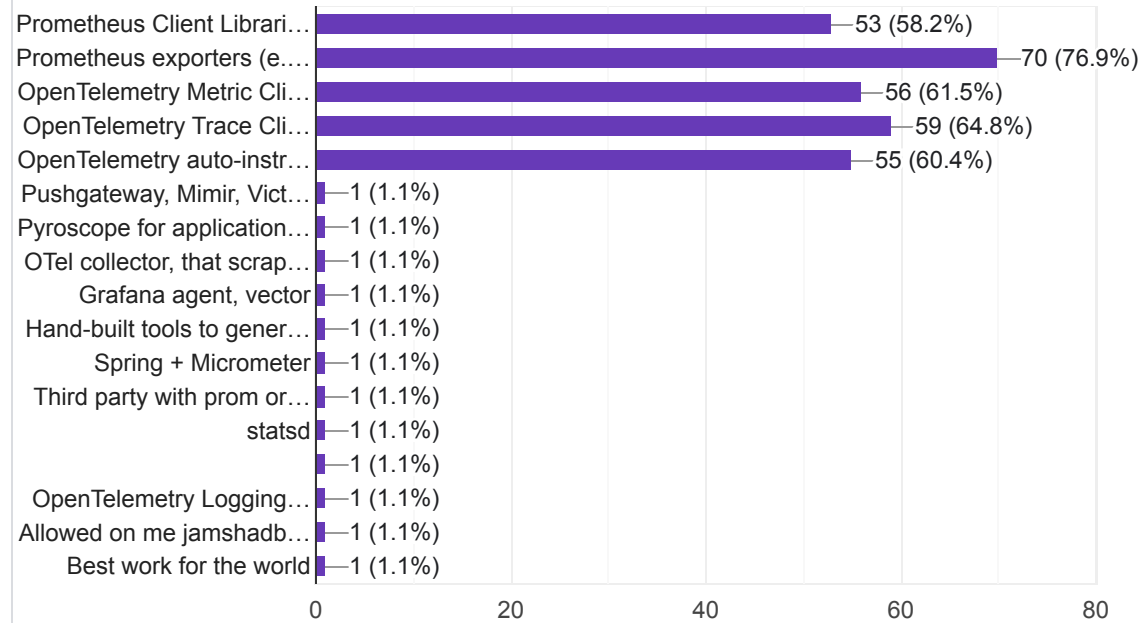
How familiar are you with OpenTelemetry?

 Copy

Which of the following instrumentation libraries or tools are you using
(select all that apply):



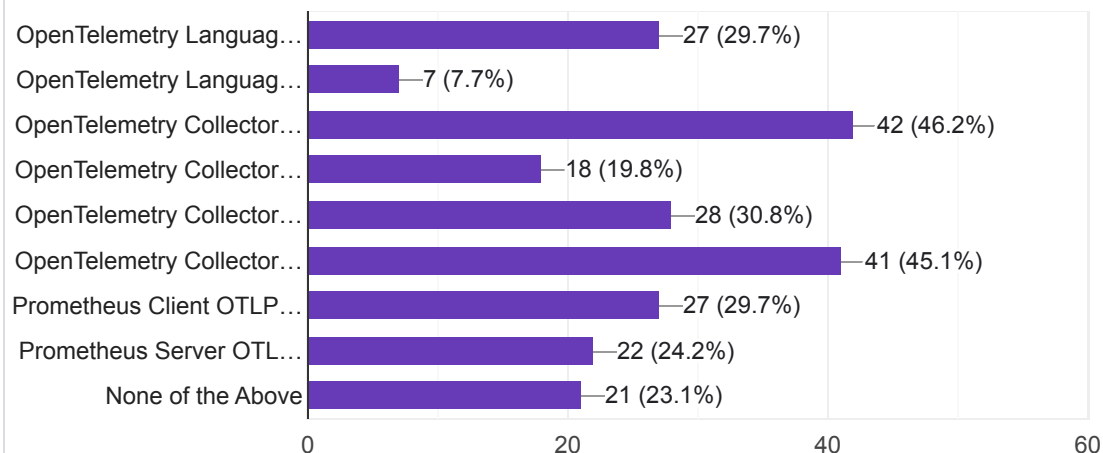
91 responses



Which of the following OpenTelemetry components are you using to deliver metrics to your backend (select all that apply):



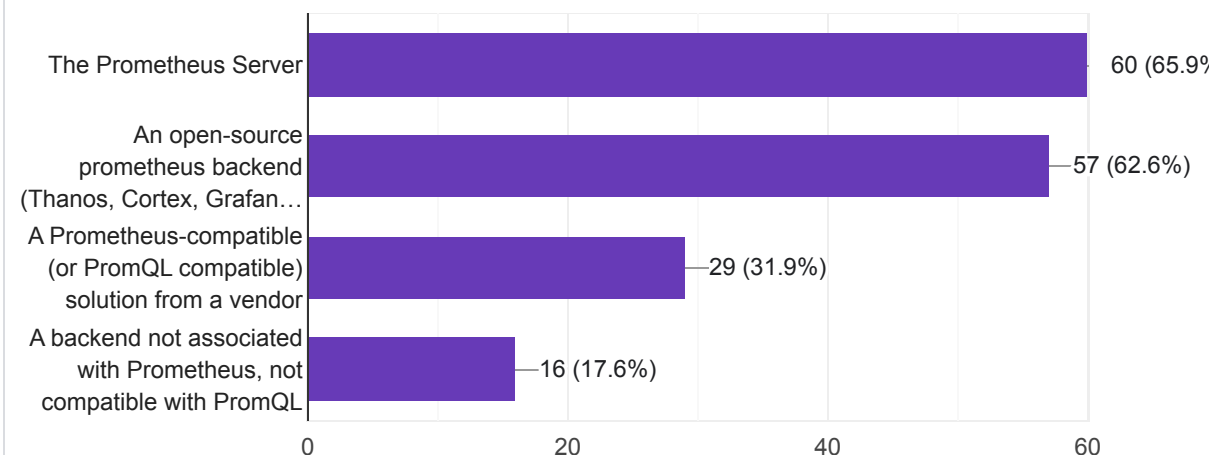
91 responses



Which of the following are you using to store metrics (select all that apply):



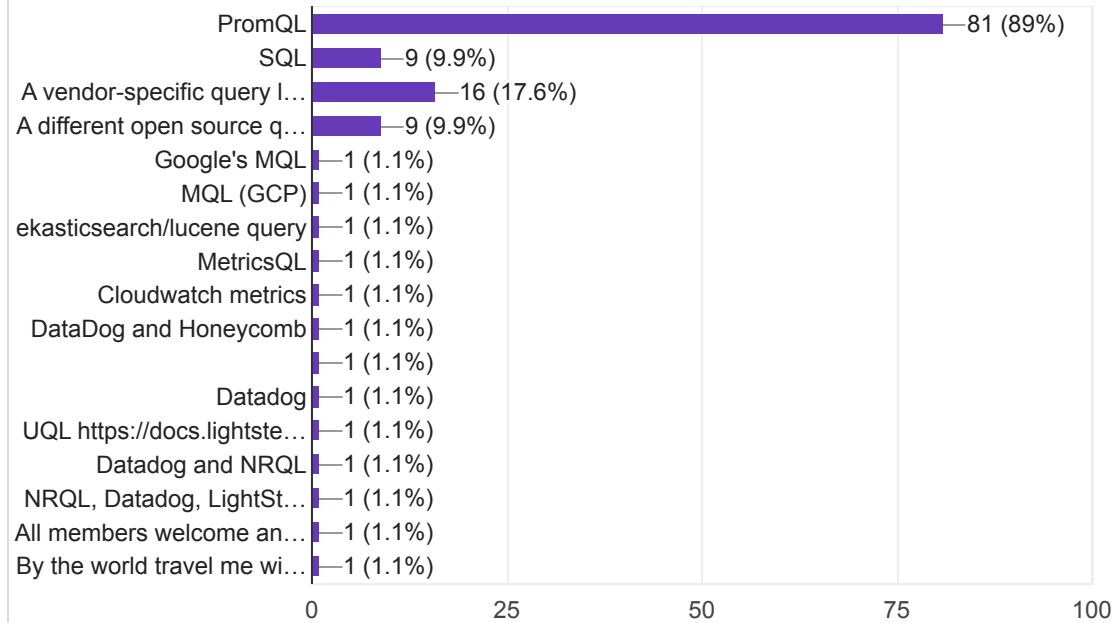
91 responses




Which query language are you using to query metrics (select all that apply):

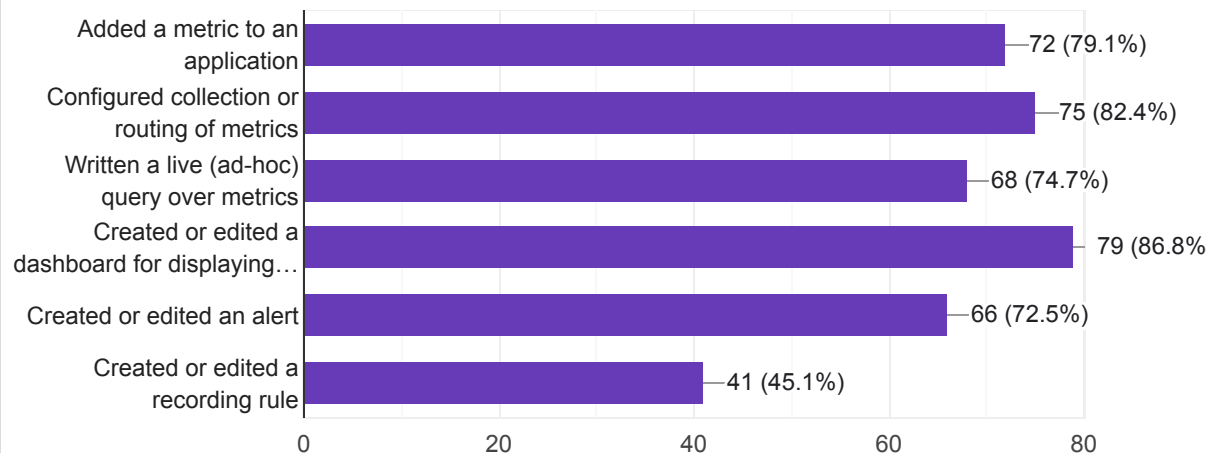


91 responses



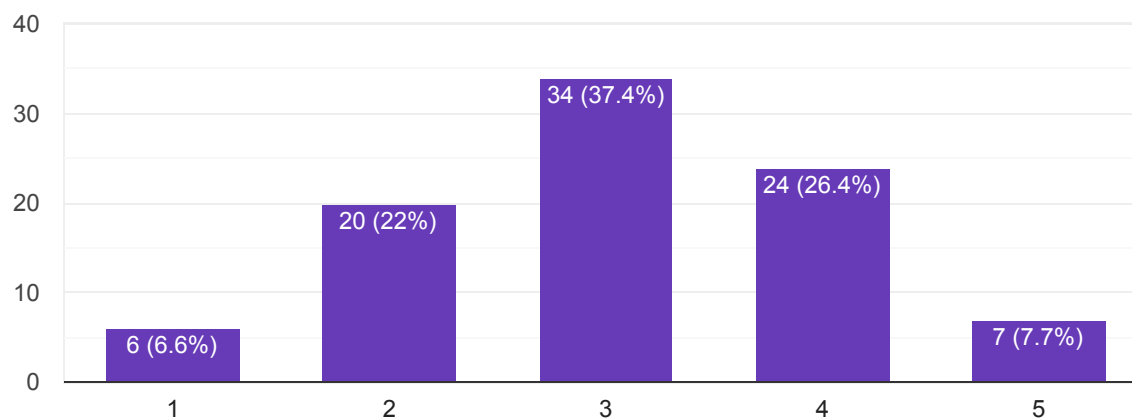
Which of these have you done in the past 6 months (select all that apply):  Copy

91 responses



How easy have you found it to use OpenTelemetry with Prometheus?  Copy

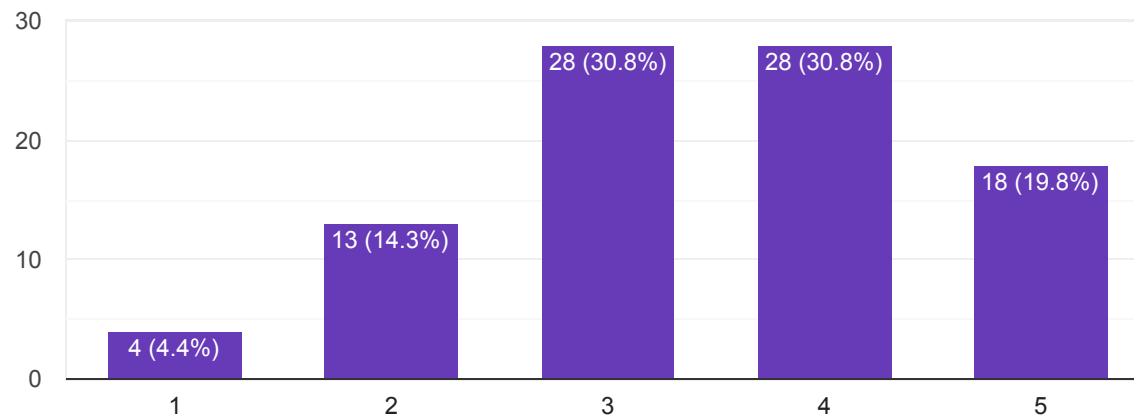
91 responses



How confusing have you found the translation between OpenTelemetry metrics and Prometheus metrics?



91 responses



What have been the biggest challenges you've faced using OpenTelemetry with Prometheus?

91 responses

One of the biggest challenges we've faced using OpenTelemetry with Prometheus is obtaining valid latency values from the OpenTelemetry endpoint, as the values reported in Prometheus do not match those in Jaeger, making validation difficult. Additionally, we have struggled to implement effective alerting to Microsoft Teams using Prometheus Alertmanager because the OpenTelemetry data lacks the necessary span IDs and other required labels and fields. These issues have hindered our ability to accurately monitor and alert on system performance.

N/A

None

We don't use OpenTelemetry to handle metrics. We don't see much benefit adding OTel Collector into the metrics handling pipeline.

Stability of the OpenTelemetry instrumentation libraries has been a major pain point with several breaking changes with some curious omissions (e.g. simple gauge support disappeared for a while)

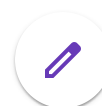
Metric names/labels changing

To understand difference

out of the box seamless integration

I don't have any need to use OTel with Prometheus

None.



Controlling the size of histogram buckets, moving from scraping -> pushing over OTLP

As of now it is working as expected

Metrics storage

resource attributes, time ordering with queue buffering, semconv

- Differences in metric semantics
- Lack of a notion of resource in Prometheus and the workarounds necessary.
- Not full UTF8 support for metric keys and labels.
- The necessity for suffixes on OTel metrics in Prometheus impedes usability

The dot vs underscore namespace syntax and different semconv

no

Dint find istio compatibility or ingesting service mesh data

The Otel instrumentation libraries are so crazy complicated, I just want to use Prometheus' instrumentation library.

different naming conventions

usage of Out of Order time window as opentelemetry data are not ordered

unexpected metric name modification

1. The default fields conversion from Prometheus metrics to OTLP (service.instance.id, etc.)
2. understanding Resource, Metrics, DataPoints and how prometheusreceiver converts it into

..



none

Performance of the OpenTelemetry Collector does not scale.

It's not possible to create automatic aggregation rules based on the type of instrument or by using annotations in the code.

Cardinality due to the excessive amount of metrics generated by most HTTP instrumentation (ASPNETCORE) as sparse/native histogram are still not a thing having one histogram per route per status code is insanely costly to store by default.

Metrics do strings and unit are still far from being well supported.

Filtering and processing metrics with otelcolcontrib before sending to DataDog and Honeycomb

The different metric types (instruments) are confusing and it is hard to tell which one to use when.

Last time I was faced an issue with the application tracing. I am not able to see the parameterized queries which I am using the application. But this issue is fixed.

Skills and knowledge

The different collectors all have different configurations

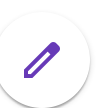
No Prometheus Remote Write Receiver in OpenTelemetry Collector

Na

Different label naming conventions; Mapping of OTEL attributes and resource attributes to Prometheus.

OTel does not support aggregations that are powerful enough.

Sometimes you do not want to send tons of data to OTel provider because it's super cost intense and all you need is an aggregated value, but IMO the idea of OTel is more to get a lot of money instead of providing value to the users.



Vendor buy in

Haven't used otel

The concept of meters and scope compared to the global namespace of Prometheus

Non-equivalence of names. How to handle OTel resource attributes in Prometheus. The confusion that OpenMetrics introduced some (partially weird) concepts that even Prometheus hasn't really embraced, and now OTel was in that weird situation what to support first class (e.g. OM says unit names MUST be in the metric name, Prometheus itself is more liberal, and now OTel is in an impossible situation what to follow).

I think the unit as part of the metric name is a good case study to understand the dissonance. Prometheus recommends both, putting the unit in the metric name _and_ using SI base units, to minimize misunderstandings. But that's partially coming from the lack of 1st class metadata support in Prometheus so that it is hard to attach unit information to a metric in a reliable fashion. Even with that metadata support, there might be arguments for including the unit name, but that was never enforced. OpenMetrics added the enforcement in the exposition format, but without getting consensus from Prometheus first. OTel, on the other hand, didn't even have a strong stance about using base units.

In my opinion, the best way out of that situation would be if all involved systems had strong metadata support, but still strongly recommended using base units, and let users make their own call if they want to have the unit in the metric name, but maybe use the unit in the metric name for commonly used metric names (e.g. in semantic conventions).

We don't have anything that wants to use OpenTelemetry instead of native Prometheus metrics.

Determining if my metrics producer (application) has stopped sending metrics

Instrumentation

integration

I've never tried. Also, I'm having a hard time understanding from OpenTelemetry's docs if they



can work together or how.

Getting the configuration in Otel right in a way it becomes transparent to the default configurations on Prometheus side.

"target info" translation is poor solution. Prometheus' restrictive naming formats and lack of data types is frustrating, permanent loss of fidelity when ingesting to Prometheus-compatible backends.

the conversion isn't obvious in the beginning, lack of control on client side such as limits on cardinality.

Prometheus not being a push based solution has caused several problems.

Ingest native histogram. Not having metadata in prometheus

Incompatible naming convention of fields, otel collector not really on HA once you scrape prom metrics etc.

Not

Supporting different temporalities (Delta/Cumulative) and converting from one to the other. The cost of high cardinality metrics on cumulative. Selecting histogram buckets

-

Naming conventions between 2 cncf project is messy how come

Metrics Translation from Otel to Prometheus and naming

The convention different between Prometheus and OTLP, which translates over into Loki. Namely, the dropping of resource attributes that aren't service.name and service.namespace (typing from memory). I'm so glad that the opentelemetry collector converts attributes to labels to solve this, however, if emitting over OTLP. Using the target_info metric to do joins



because Prometheus drops attributes when using OTLP is bonkers. Honestly, it should just support OTEL SemConv.

I'm satisfied with Prometheus, so I don't have any reason to use OpenTelemetry.

OTel standards requires more complexity than just using exporters with OpenMetrics format.

Understanding of architectural concepts and different components involved.

Conventions don't align very well. Otel's recent tendency to put namespaces everywhere, especially in attributes, makes it very clumsy in PromQL. But overall, the definition of metrics in Otel is much better than in Prometheus, because you don't have to put the type and unit of the metric in its name.

Dots, slashes and such in metric names

That OpenTelemetry chosen dots as separators, how over informative labels are and decisions, breaks with Prometheus model without good reasons or explanation (e.g. initial le vs ge for histograms), too complex OTLP format, tracing driven metric data model.

OTel created a whole new spec when Prometheus already dominated that space. Now everybody is requiring Otel and it just feels like unnecessary duplicated work. We're not solving new problems, we're just recreating old ones.

In recent otel-javaagent changes metric names changed and units in metric names are not consistent.

ليس الكثير

Educating people (tech and business) that we don't have a Prometheus instance anymore

Even if you use a agent, still need to manually associate the trace id.

Unstable status of the specs.



The collector prometheus receiver is very limited (support for crds, scalability, HA)

OpenTelemetry has concepts that don't exist in Prometheus such as resources. It was surprising to add an attribute to a resource but not have it propagate to the metric when exporting as prometheus

Pushing otel vs pulling prom

Lack of ability to easily query on Otel Resource attributes in PromQL (doing a left join on target_info{} is *not* an answer). Change in naming convention (e.g. always changing the "." to an "_". The inconsistency (in general - e.g. added suffixes for units/total) between names in Otel Semantic Conventions and Prometheus metrics. "job" vs "service.namespace/service.name". The need for "service.instance.id" to satisfy Prometheus' "instance" requirement.

Querying the metric data with promql for alerting in alert manager.

Not able to validate the latency values and struggling with alerting with MS Team

Prometheus Remote Write Setup

Not able to create alerts using PromQL queries on top of the OpenTelemetry-generated metrics. Ensure that the metric names and labels used in PromQL queries match exactly with those generated by OpenTelemetry.

Mapping concepts

Mainly updating services to export OTLP metrics and not just prometheus metrics and the shift in mentality for pushing metrics to the OTEL Collector and then to Prometheus.

The architectural complexities with the difficulty to find the documentation explaining it.

Easy



Understanding the difference between resources and labels in prometheus

Naming, conventions

I'm sure you are welcome anytime membership best friend request on All member login

I'm sure you are welcome anytime membership but I am now money option me account some follow me on All member

Initial implementation of the operator with target allocator

Documentation and API



What would you like to see changed or improved with OpenTelemetry's Prometheus support?

91 responses

N/A

Nothing

I would like to see improvements in the accuracy and consistency of latency values reported by the OpenTelemetry endpoint. Additionally, enhancing support for alerting integration with Prometheus Alertmanager, by ensuring the availability of necessary identifiers like span IDs and other required labels and fields in the OpenTelemetry data, would greatly improve our ability to monitor and alert on system performance effectively. We aiming to utilize OpenTelemetry to monitor our application endpoints and implement alerting functionality based on predefined latency thresholds. If any endpoint exceeds these defined latency values, we expect the system to trigger an alert to Microsoft Teams. This proactive approach will enable us to promptly identify and address issues related to specific application endpoints, ensuring optimal performance and user experience.

Nothing.

Stabilised API

Not sure

Improve docs to avoid confusion using push pull metrics receiver/exporters in collectors

I don't enjoy the idea of OTel forcing changes in PromQL and making it all complicated for non OTel users like me

Even more auto-instrumentation



Honestly not sure, sorry! After 7ish years in this space, I'm starting to feel like Prometheus itself is simply too user hostile to be the industry's de facto metrics tool.

Add RUM for frontend and mobile apps

resource attributes, time ordering with queue buffering, semconv

OTel's richer semantics are very much appreciated. Having to boils this down to fewer semantics feels wrong and adds usage complexity. I have no proposal that would address this shortcoming, but not forcing awareness of this on end-users would be great.

prometheus has to improve and support otel semconv

support Delta temporality

if it can have more interconnectivity like recording rules

Reverse support, eg. drop in support for the Prometheus library, and do everything the way Prometheus would but be able to take advantage of "the good parts" of otel, which is just to say a standardized wire protocol.

adopt the same naming convention

"Pairing" of metrics and traces to be able to debug issues

Prometheus to natively support opentelemetry semantic conventions

there should be an option in prometheusreceiver to not convert duplicate labels (e.g. job to exported_job, instance to exported_instance)

..

none



Keep the pull model it's good for discovery, maybe with a constantly opened connection though (reverse push) for performance and streaming updates, gRPC style.
Better support for sparse/native histogram, cheap with good buckets boundaries.
Make it easy to define how to aggregate the metrics from the language.
Propagate metrics doc strings and unit into some kind of useful available in Grafana.
(automatic panel units)
Enrich PromQL query language, it's very hard to explain to users that you can't do even basic transformations without having to rely on what looks like language hacks.

Stable JavaScript metrics library

I would like to see concepts close to what Prometheus has, rather than inventing new abstractions which have not been tested in practice.

Need more metrics related to the application performance and documented these metric related information in the OTEL docs. Please keep all metrics semantics in common format

Operations systems

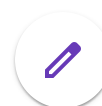
Easier query lang other than promQL

RemoteWriteReceiver and ability to have dot in metrics (service.name and service_name both end up being service_name in Prometheus)

Na

I think Prometheus should adapt to OpenTelemetry's metrics model. It should be very easy to export metrics with OTel Collector to Prometheus without additional configuration. It is important that it "just works".

Aggregations need to be built in to not pass tons of data you do not need to feed only the bill of your provider and heat up the world by unnecessary compute.
I hope you don't want to break again user experience like it was done by Opentracing to OTel migration....



Unsure

Documentation for the newcomer, not just those well versed in Otel.

Semantic conventions that aren't using characters that are invalid (or unusual) for Prometheus.

No opinion due to no use.

Delta temporality support :) It would make interoperability between systems much easier

explain otel and prometheus architecture more clearly

Is it supported? Where is the documentation for that? What does supported mean? Do prometheus metrics flow into the OTel system? Or do OTel metrics flow into Prometheus? Or both?

Converge of metrics naming conventions.

solution for target info, working with / pushing Prometheus for better compatibility

it would be great to have a unified semantic convention on both otel and prometheus without having to do conversion. Unit name configured in otel should be included by default in prometheus as part of measurement name.

Loadbalancing exporter for metrics.

Having SDKs exporters implement protobuf to ingest native histograms directly

All the things mentioned above

Delta support / better deltatocumulative support

-



Not sure I need otel for metrics

Support for dot in the metrics names in Prometheus

Prometheus moving to the OTEL standards.

A tool that can be set up easily.

I'm not sure, though I hope Prometheus could support OpenTelemetry without losing established Prometheus standards (like metrics format).

Simplified guide for a Prometheus -> to OTEL Collector -> to ANYTHING implementation

The translation of metrics from Otel to Prometheus works surprisingly well. It would be great if the prometheusreceiver could perform the same reverse translation.

I would prefer to see more alignment with what exists in CNCF. Do not reinvent the wheel.

OpenTelemetry reverted, revisited some decisions, and/or started experimenting and iterating more vs creating arbitrary decisions breaking everyone and/or explained their intention and goals clearly for each breaking change.

Some decisions e.g. Unit felt like without real motivation (or at least nothing noted).

I'd like to see OTel's metric spec to disappear and Prometheus to remain the standard.

Consistent units in metric names...

استخدام المواقع بصورتي

See operator's #1669 closed soon, separate target allocator CRD to have it VPA-scaled

Fully automated to collect observable data



Some HTTP / RPC metrics use seconds and some use milliseconds as the unit of choice (as per sem conv).

Better receiver for scrapping prometheus, an easy way to transform specific resource attributes to labels without having to use otel function (right now it is all or nothing with resource_to_telemetry_conversion property)

The ability to change attributes of metrics so that I can make them match the same conventions we use with Prometheus today.

Allow "." in Prometheus metric names. More examples of how to write recording rules in Prometheus receiver in Otel Collector documentation for people who are *not* Prometheus experts, but are OpenTelemetry experts.

Opentelemetry metric s should be match the Prometheus metric s.

More features to handle alerting and latency values of endpoints

Easier Ingestion into Prometheus

.Ensure that the metric names and labels used in PromQL queries match exactly with those generated by OpenTelemetry.

GO SDK

- * It would be great to have migration guides for Devs from prometheus client libraries over to opentelemetry
- * It would be great if Prometheus natively supported OTLP metrics using push based mechanisms without it being behind an experimental flag (using it in production so not a deal breaker)
- * It would be great if Prometheus could receive OTLP metrics over GRPC
- * It would also be great if Prometheus could store OTLP metrics using the OTLP naming conventions



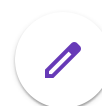
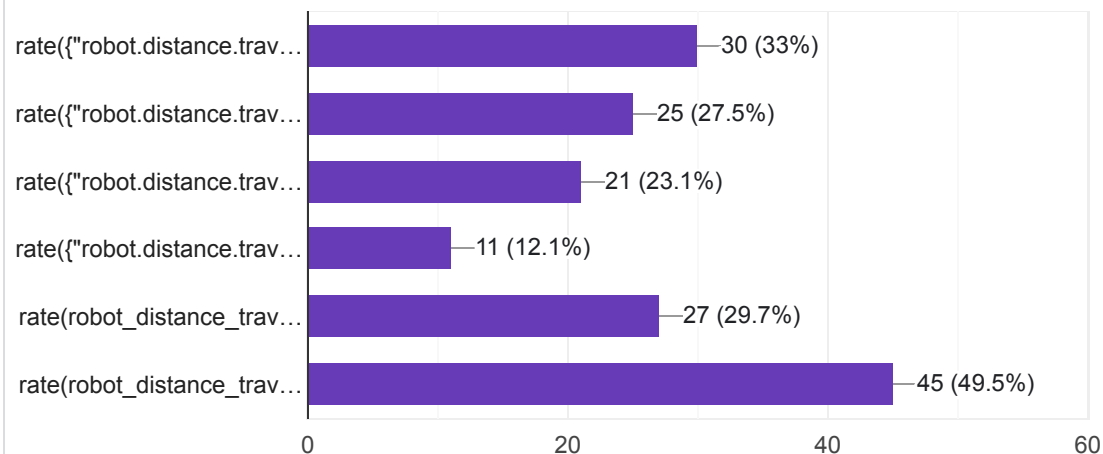


Suppose you've defined your own OpenTelemetry metric in Go to track how far your robot has traveled:

```
counter, _ := meter.Float64ObservableCounter("robot.distance.traveled",  
    metric.WithUnit("mm")  
)  
// Call counter.Add() each time the robot moves.
```

Which of the following would you expect to write (regardless of what is valid/correct today), (select all that apply):

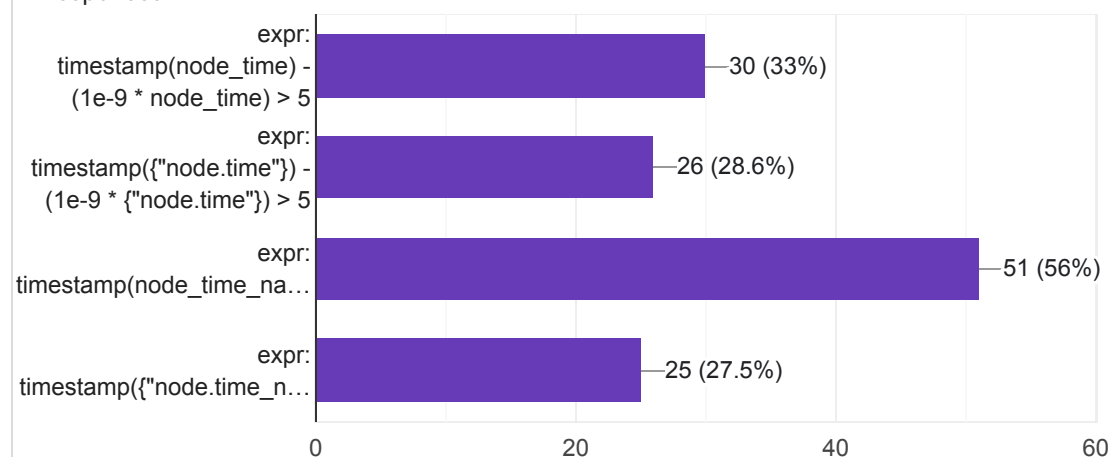
91 responses



Imagine you know that your node's OpenTelemetry instrumentation defines a `node.time` metric, but you don't remember its unit. Someone complains about a flaky alert, so you open a YAML file with your alert configuration. Which of the following you prefer the most (select all that apply):



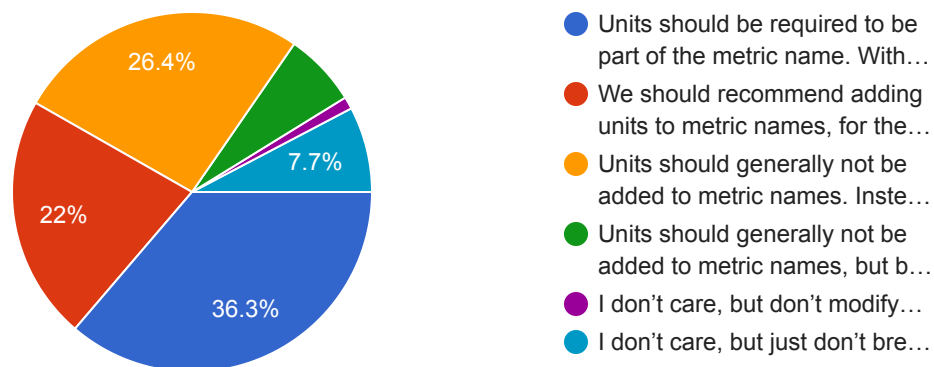
91 responses



Which of the following describes your opinion on units in metric names (select one):



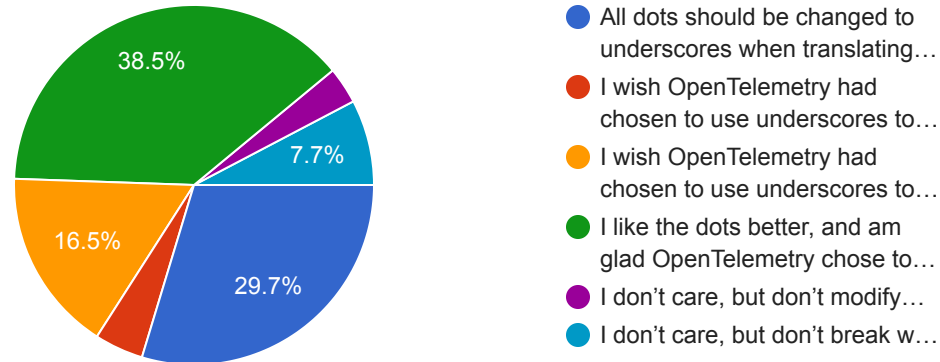
91 responses



Which of the following describes your opinion on dots vs underscores in metric names (select one):



91 responses



This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms



