

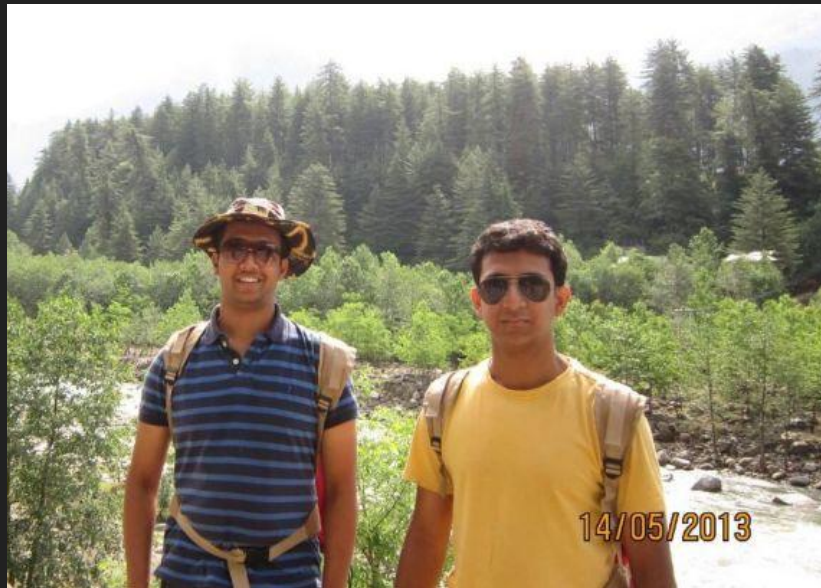
Is OpenTelemetry too Complicated to get started?

Pranay Prateek

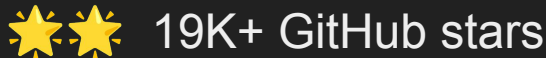
Maintainer & Co-Founder, SigNoz


Who am I?

- Co-Founder & Maintainer at SigNoz
- Used to run OpenTelemetry APAC End User group
- Love Reading & Trekking



OpenTelemetry-Native Traces, Metrics & Logs in a single pane



 150+ contributors



r/devops • 3 yr. ago
twocolor



OpenTelemetry is great but why is it so bloody complicated?

I've been reading the documentation over the last couple of days and using the libraries to instrument a node.js GraphQL API.

I love the idea of open standards and interoperability. For example, I'd even go to argue that in large Kubernetes achieved those goals to a significant degree in the sense that you can pretty easily move workload/ from one cloud to another. And I can see a similar thing happening with OpenTelemetry.

But OpenTelemetry feels like it's a level up on getting to a productive threshold. In other words, the oppose of the pit of success for such an important and useful technology — especially developers building data intensive applications.

Here are my current gripes:

- difficult naming. Propagator, exporter, manager, provider OTel, OTLP. It seems that interchangeable terms are used for the similar things.
- the number of dependencies needed to instrument an app is insane. Let alone figuring out the correct versions
- Documentation is playing catch up with the implementation
- resources is either very specific or very general. I haven't been able to find any good resources that pick the right balance of breadth and depth.

▲ **OpenTelemetry in 2023** (kevinslin.com)

341 points by kevinslin on Aug 28, 2023 | hide | past | favorite | 244 comments

▲ paulddraper on Aug 28, 2023 | next [-]

Two problems with OpenTelemetry:

1. It doesn't know what the hell it is. Is it a semantic standard? Is a protocol? It is a facade? It is a library? What layer of abstraction does it provide? Answer: All of the above! All the things! All the layers!
2. No one from OpenTelemetry has actually tried instrumenting a library. And if they have, they haven't the first suggestion on how instrumenters should actually use metrics, traces, and logs. Do you write to all three? To one? I asked this question two years ago, zero answers :([1]

[1] [https://github.com/open-telemetry/opentelemetry-specification...](https://github.com/open-telemetry/opentelemetry-specification)

▲ withinrafael on Aug 28, 2023 | parent | next [-]

1. Agreed. It's the sink and the house attached to it, and the docs are thin and confusing as a result.
2. I had a similar experience to you. I wanted to implement a simple heartbeat in our desktop app to get an idea of usage numbers. This is surprisingly not possible, which greatly confuses me given the name of the project. The low engagement on my question put me off and I abandoned my OpenTelemetry planning completely [1][2].

[1] <https://github.com/open-telemetry/community/discussions/1598>

[2] [https://github.com/open-telemetry/semantic-conventions/issue...](https://github.com/open-telemetry/semantic-conventions/issues...)

What we will discuss

- What issues we hear from users looking to implement OpenTelemetry
- Potential reasons for it and how we as a community can improve it
- OpenTelemetry adoption in organizations & issues faced

What do we hear from users?

Complex Learning Curve

- Too many concepts required for basic instrumentation
- Complicated architecture with processors and exporters



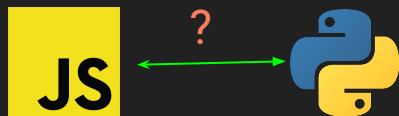
MeterProvider.. Meter.. Instrument.. Oh my!

```
+-- MeterProvider(default)
|
+-- Meter(name='io.opentelemetry.runtime', version='1.0.0')
|
|   +-- Instrument<Asynchronous Gauge, int>(name='cpython.gc', attributes=['generat
|   |
|   +-- instruments...
|
+-- Meter(name='io.opentelemetry.contrib.mongodb.client', version='2.3.0')
|
|   +-- Instrument<Counter, int>(name='client.exception', attributes=['type'], unit
|   |
|   +-- Instrument<Histogram, double>(name='client.duration', attributes=['server.a
|   |
|   +-- instruments...

+-- MeterProvider(custom)
|
+-- Meter(name='bank.payment', version='23.3.5')
|
+-- instruments...
```

Cross Language inconsistencies

- Behavior varies significantly across language SDKs (JavaScript, Python, Go)
- Recent standardization efforts have only partially addressed these differences ^[1]



[1] <https://github.com/open-telemetry/opentelemetry-configuration>

Relax validation rules on header values passed by environment variables #3189



Open

cpnat opened this issue on Feb 19, 2023 · 5 comments



cpnat commented on Feb 19, 2023 · edited

...

Is your feature request related to a problem? Please describe.

Headers can set by environment variables, as defined [here](#).

However the values must be URL encoded strings, and pass through validation [here](#).

Assignees

No one assigned

Labels

blocked by spec



moxious commented on Oct 18, 2023 · edited

...

+1 this, but also dropped by to add a note here that the way the python SDK works disagrees with the JS SDK.

In the JS SDK, this will work without any warning, even when logging set to verbose:

```
export OTEL_EXPORTER_OTLP_HEADERS="Authorization=Basic $(echo -n $USER:$PASSWORD | base64)"
```



The same fails with the Python SDK with the error:

```
Header format invalid! Header values in environment variables must be URL encoded per the OpenTelemetry
```



Unexpected API Differences

- Common operations differ from other ecosystems and are hard to follow.
- Example: Histogram bucket configuration varies significantly from Prometheus conventions

Silent Failure Modes

- Issues occur without clear error messages or debugging information
- Makes initial setup and troubleshooting difficult
e.g. issues with pre-fork in python
- Non verbose logs - esp. version compatibility issues are difficult to detect

Is Hypercorn pre-fork or post-fork? How can I integrate it with opentelemetry? #215



Open

ar-qun opened this issue on Apr 18 · 5 comments



ar-qun commented on Apr 18



Hello everybody,

I don't seem to manage to make open telemetry work with Hypercorn.

Could someone give me any suggestions related to the process model Hypercorn is using? Here is for uWSGI and Gunicorn <https://opentelemetry-python.readthedocs.io/en/latest/examples/fork-process-model/README.html>

Assignees

No one assigned

Labels

None yet

Projects

Potential reasons and areas to focus on

Extensibility as a design choice - has made getting started with learning curve

- This is also because people are coming to OpenTelemetry from other ecosystems
- This will change as OpenTelemetry becomes the default protocol with which people start their o11y journey

OTel is still the most advanced implementation doing all 3 signal telemetry

- Other implementation simpler as they had narrow scope and extensibility
- More signals will be added like profiling etc.

We still need to do work to iron out the issues around

- Better declarative config
- Verbose errors to let users debug themselves

For an org, does investing in OTel justify the RoI?

Will investing in OpenTelemetry be worth the investment for my team?

What's the best practices to deploy in production?

Need more user stories on how orgs have deployed opentelemetry at scale

- We at SigNoz are publishing case studies and will share more such otel spec case studies in future
- Guides on different OTel collector deployment patterns

Summary

- OTel still has steep learning curve esp. as people are coming from other ecosystems
- One of the reason is extensibility as a design choice which would be beneficial in the long run
- As a community we need to work together to remove cross language inconsistencies and provide more effective error messages
- OTel is **still** the most advanced implementation doing all 3 signal telemetry



**Want to discuss more around
implementing OpenTelemetry?**

Join us for Happy Hours!

Studio at Soundwell

149 W 200 S

Chat more in our slack
community signoz.io/slack



@pranay01

Thank You!