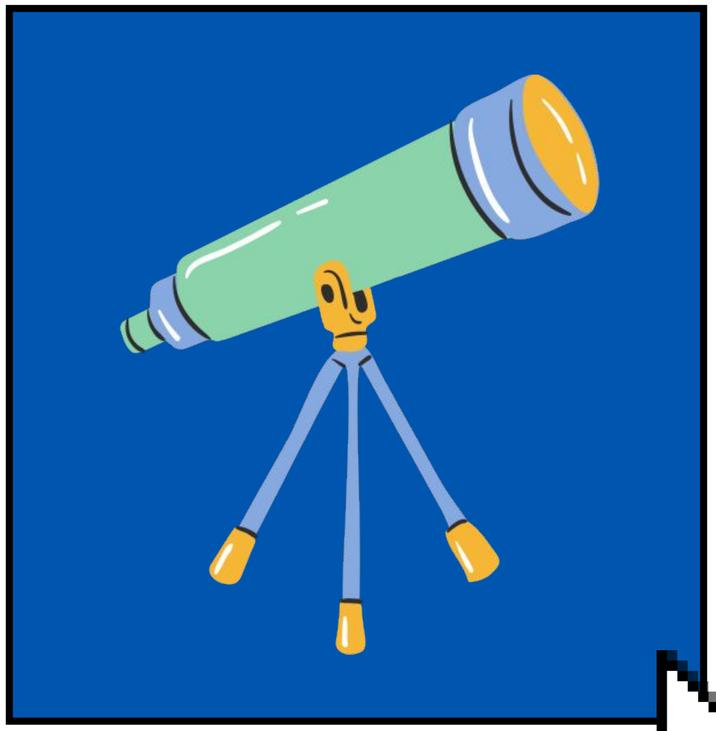
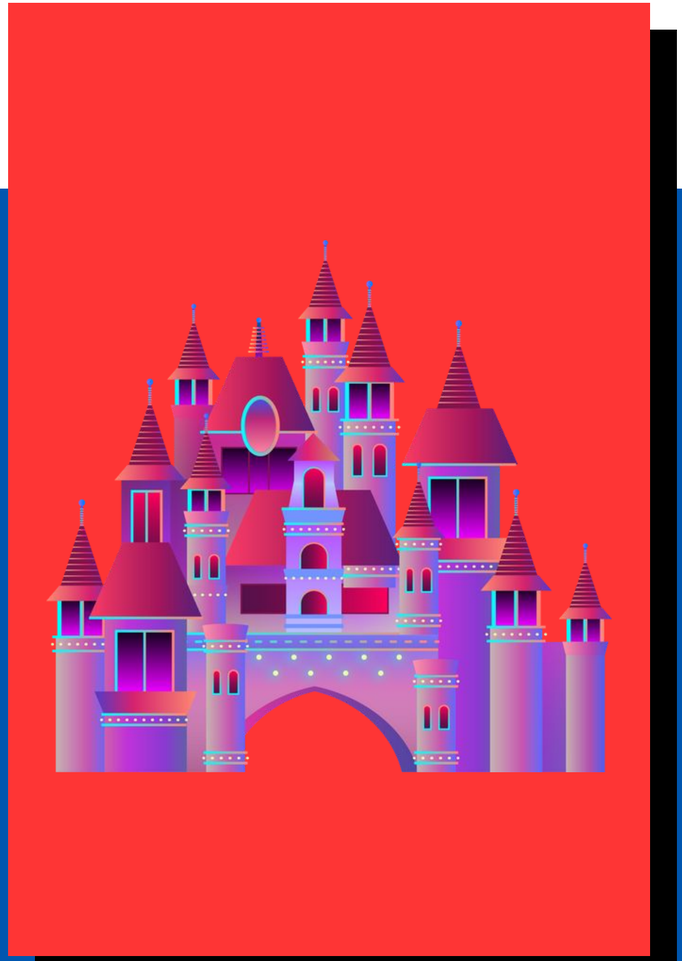


TELEMETRY WITHOUT THE TOOL TAX

Ben Greenberg

@rabbigreenberg

Once upon a time...

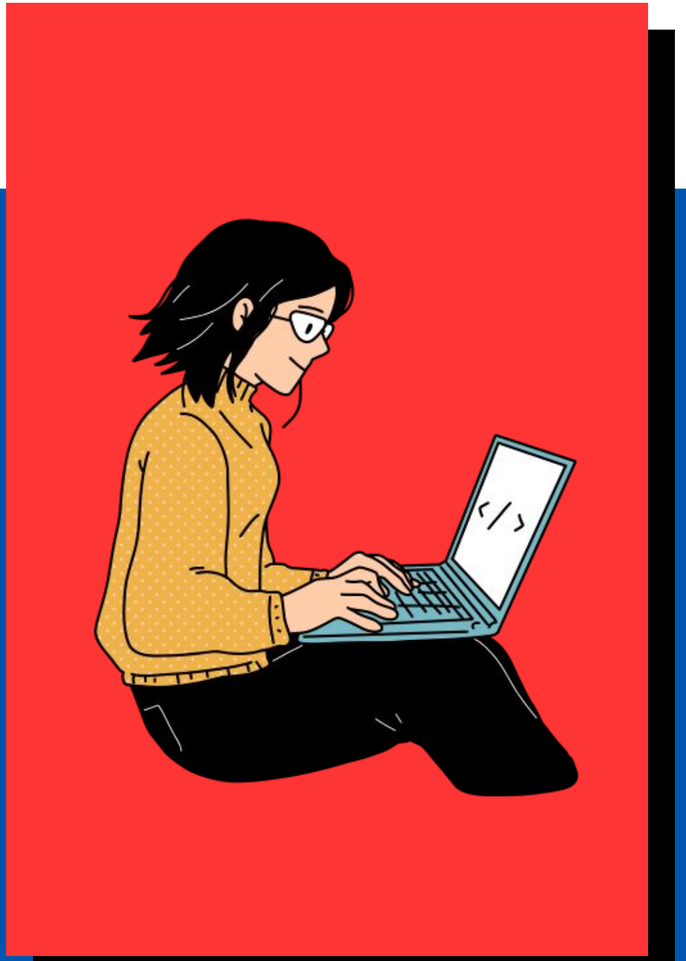


@rabbigreenberg

```php
<html>
    <head>
    </head>
    <body>
        <ul>
            <?php for($i=1;$i<5;$i++){ ?>
                <li>
                    My Brilliant Idea Number: <?php echo $i; ?>
                </li>
            <?php } ?>
        </ul>
    </body>
</html>
```

@rabbigreenberg

Web development was simpler

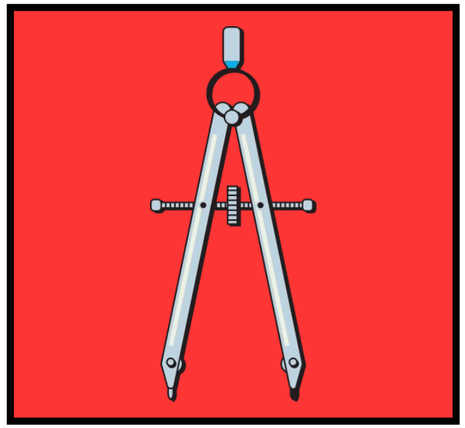


@rabbigreenberg

What happened?

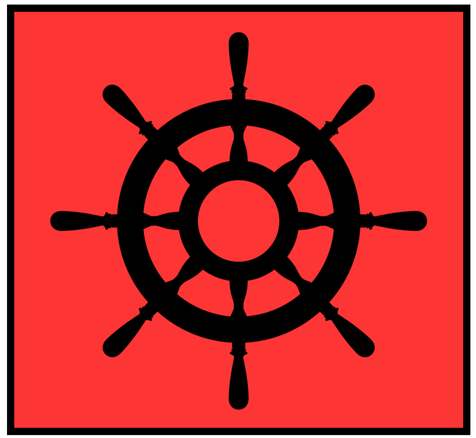
@rabbigreenberg

Distributed Architecture



Containerization

Kubernetes



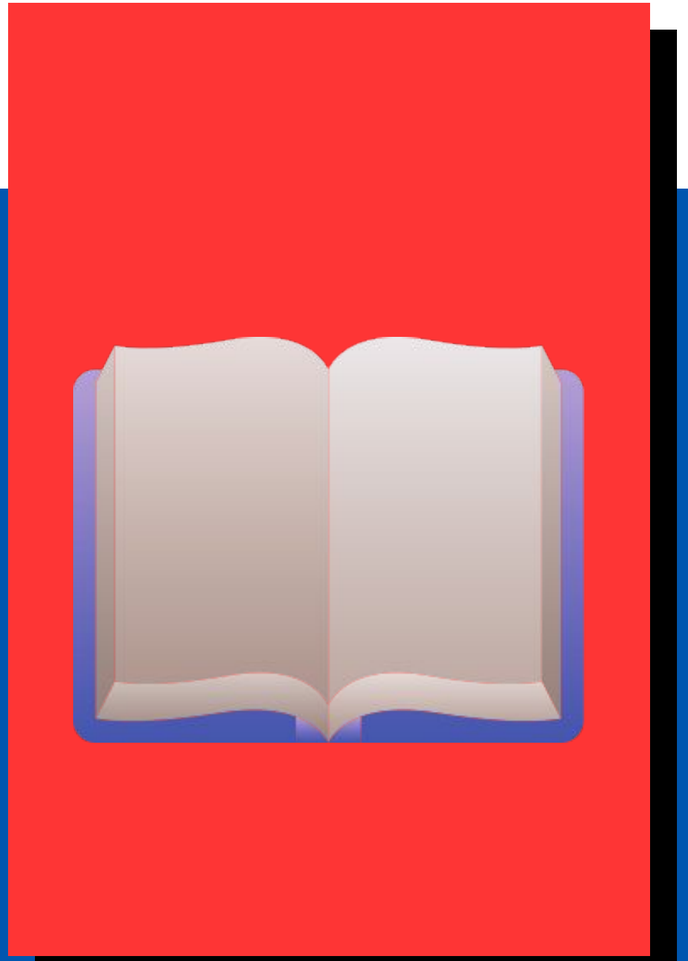
Five Nines

@rabbigreenberg

As complexity grows, so does the ecosystem to support it

@rabbigreenberg

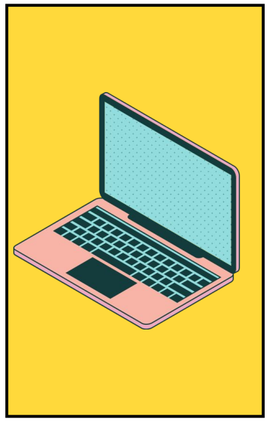
Maybe you have a similar story to my own?



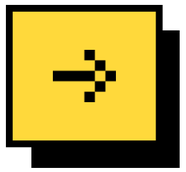
@rabbigreenberg

Hi, my name is Ben

- Software Developer

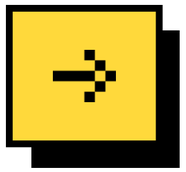


@rabbigreenberg



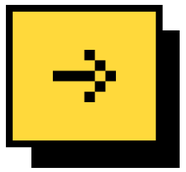
I write code

@rabbigreenberg



Sometimes,
I even write tests

@rabbigreenberg



Then one day, something happened...

@rabbigreenberg

I discovered Observability was useful

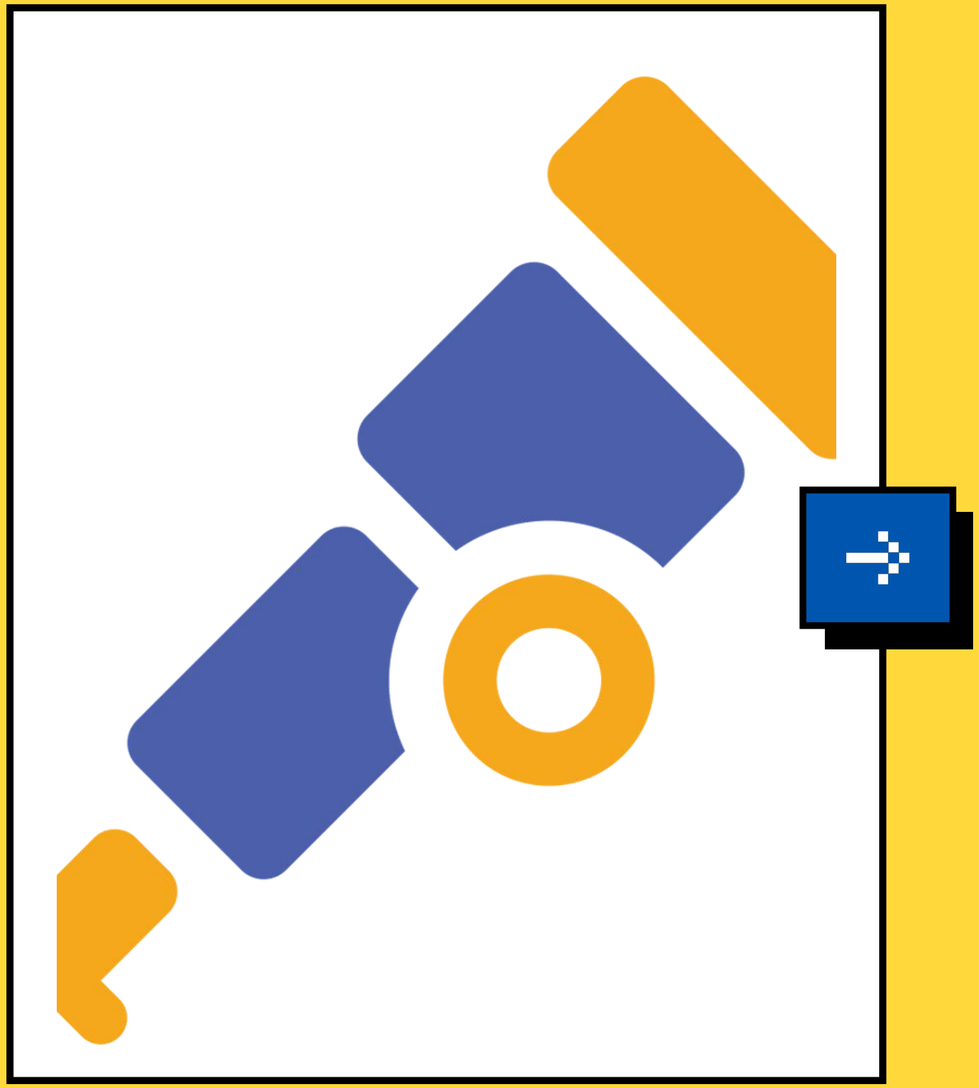
@rabbigreenberg

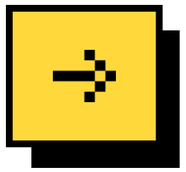
But, there's a lot of tooling out there

How do you choose?

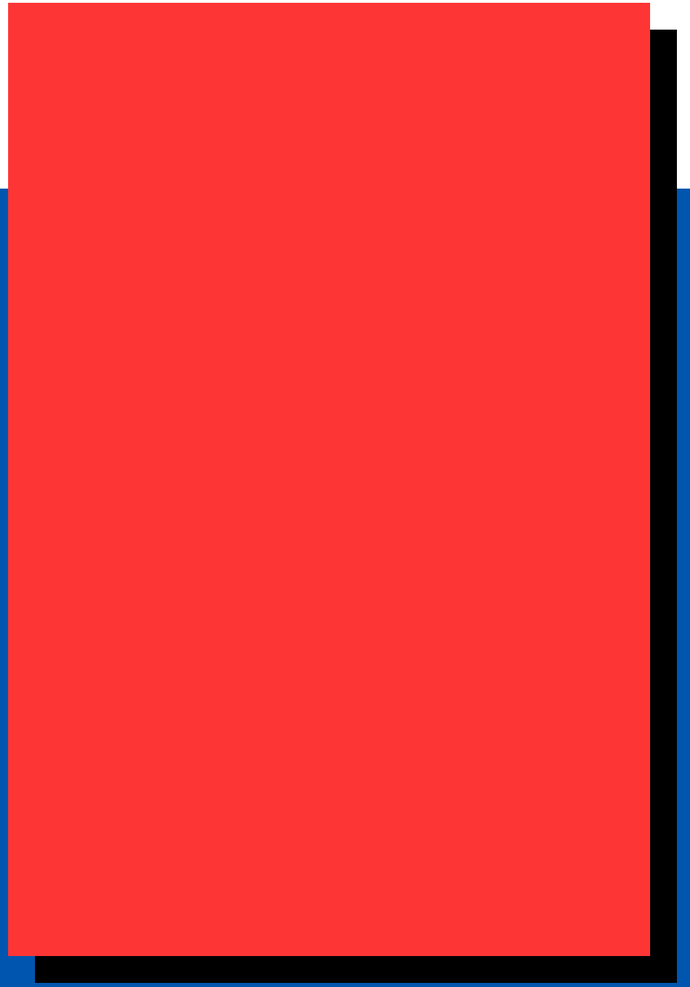
OpenTelemetry

is a response to tooling overload and vendor lock-in





Gain critical visibility into your app with one dependency



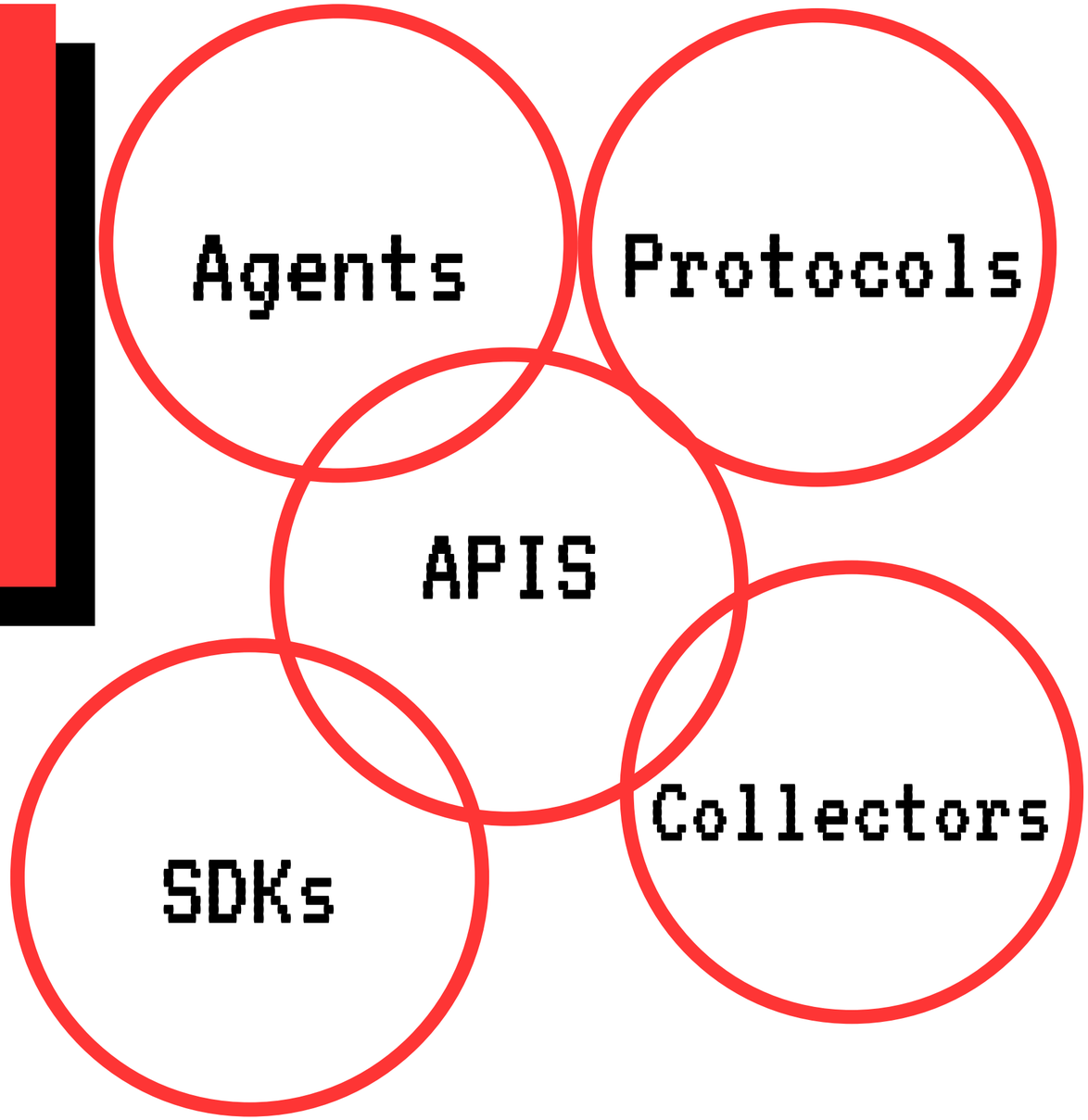
@rabbigreenberg

HOW DID IT ALL GET STARTED?

The origin story of OTel

The Telemetry Reality:

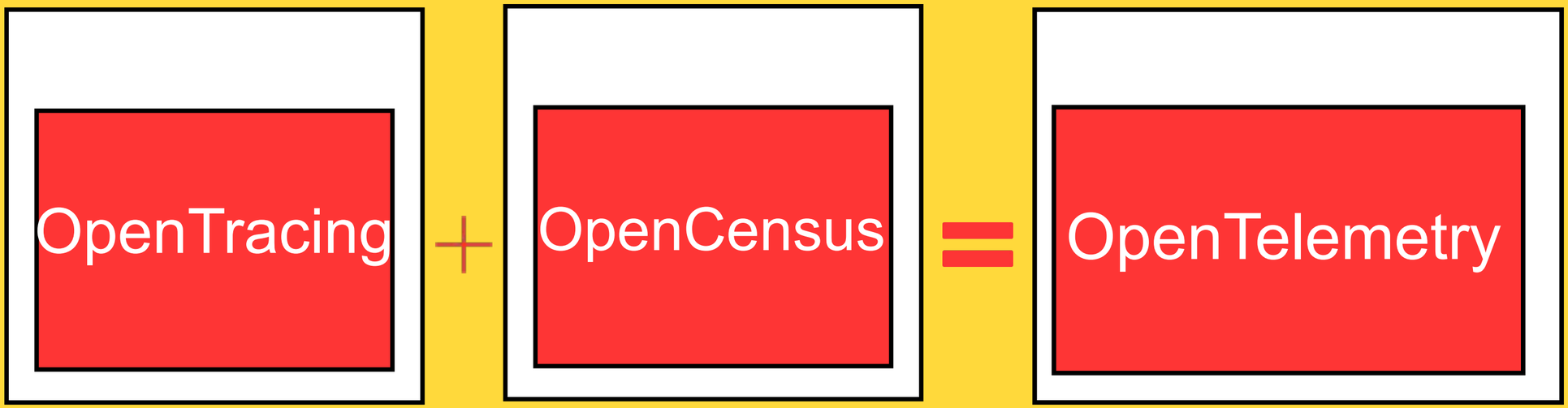
Divided instrumentation



The evolution of the project

...

CLOUD NATIVE
COMPUTING FOUNDATION



2015

A vendor neutral standard and set of APIs to trace across microservices

OpenCensus

2017

Collect metrics and traces and send to any backend of your choice

@rabbigreenberg

OpenTelemetry

2019

A unified point to integrate metrics and traces across all surfaces

@rabbigreenberg

OpenTelemetry

2021

...

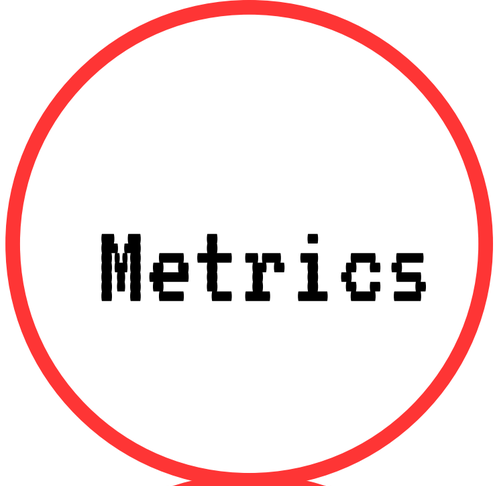
v1.0.0 released of the specification

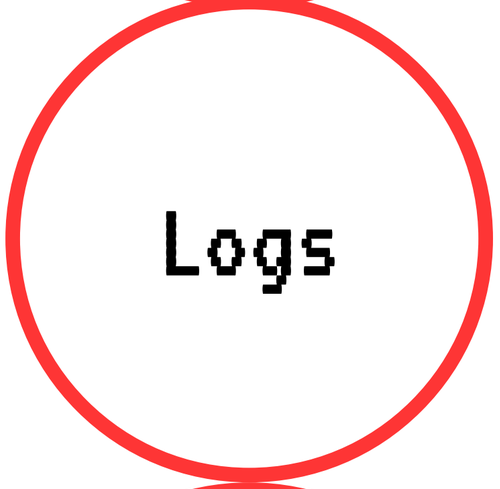
@rabbigreenberg

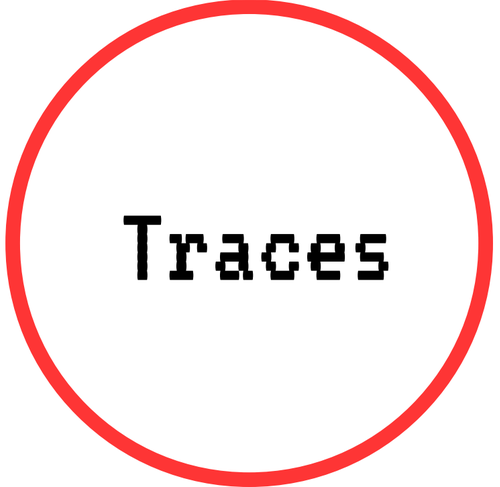
The OTel Perspective:

United Instrumentation

@rabbigreenberg







Open standards decrease complexity

@rabbigreenberg

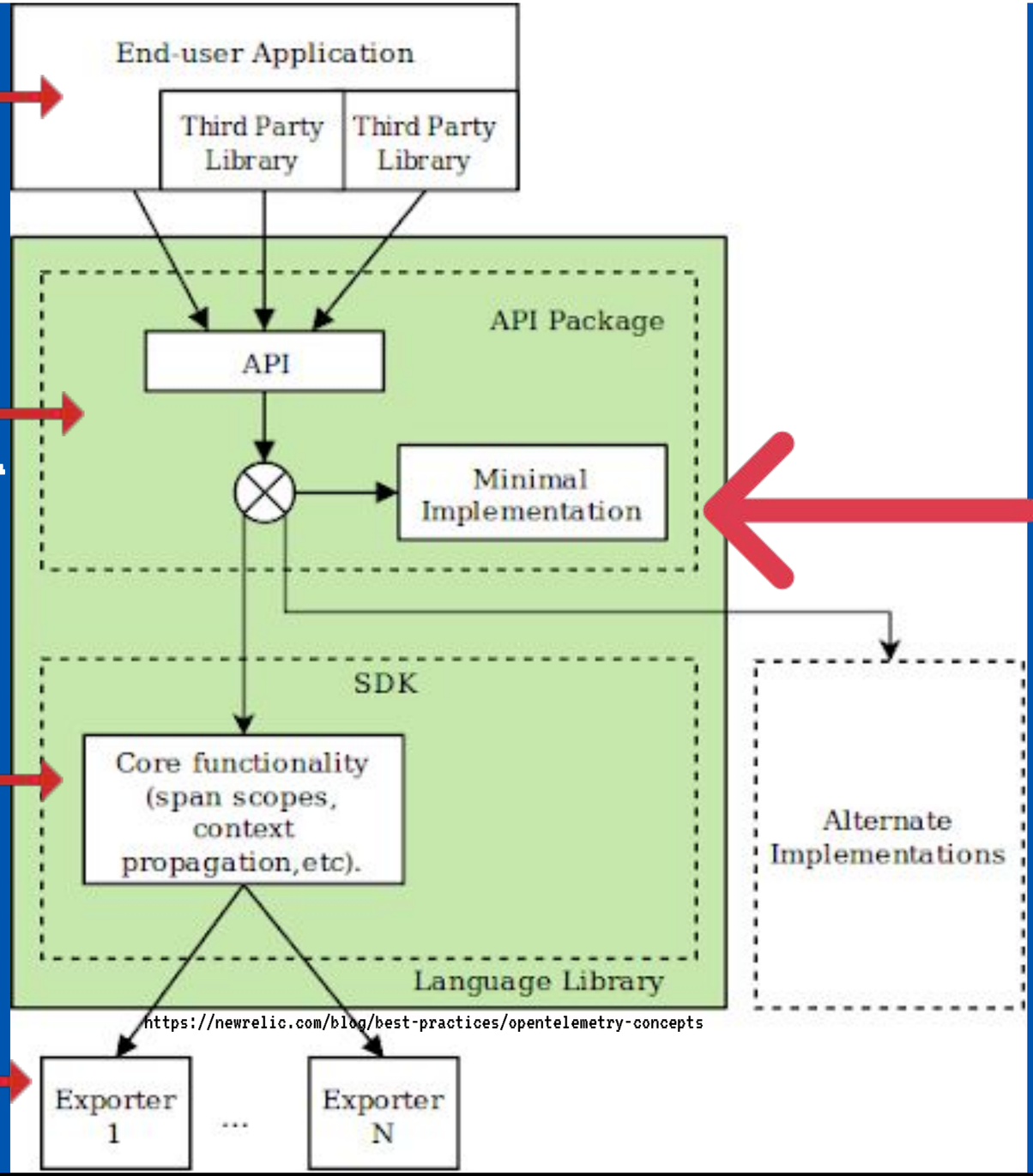
HOW DOES IT WORK?

This is your application

This is the OTel API layer

The instrumentation layer by language

Processor and exporter of the telemetry data



Construct metric data without concern if it gets consumed by an OTel agent or not

@rabbigreenberg

Breaking down the OTel Collector

Receiver

Accepts telemetry data

Popular library -

OTLP (OTel Protocol) Receiver

Processor

Pipeline to process the data

(i.e. filter, sample and enrich the data)

Exporter

Send the data to a backend of your choice

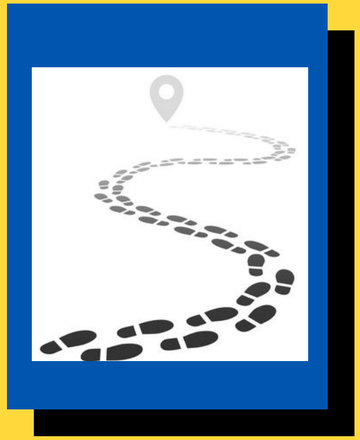
Formats the telemetry data for the requirements of the chosen backend

TRACES AND METRICS

What's the difference?

Traces

Capturing a Single Request



01 **Trace**

is a single request in the system

02 **Span**

is a single component of a trace

03 **Traces**

are trees of spans

@rabbigreenberg

Metrics

Capturing data over time

01 **Metrics**

are aggregated measurements

02 **Metric**

data captures a moment in time

03 **Metrics**

are less detailed than traces

@rabbigreenberg

Example of Trace API Implementation

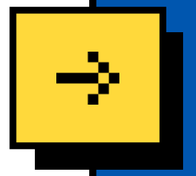
```
Span span = myTracer.startspan(/*...*/);

try {
  // Code that does the actual work which the span represents
} catch (Throwable e) {
  span.recordException(e, Attributes.of("exception.escaped", true));
  throw e;
} finally {
  span.end();
}
```

https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/trace/semantic_conventions/exceptions.md

Example of Metrics API Implementation

```
meter
    .gaugeBuilder("cpu_usage")
    .setDescription("CPU Usage")
    .setUnit("ms")
    .buildWithCallback(measurement -> {
        measurement.record(getCpuUsage(), Attributes.of(stringKey("Key"), "SomeWork"));
    })
```



https://opentelemetry.io/docs/instrumentation/java/manual/

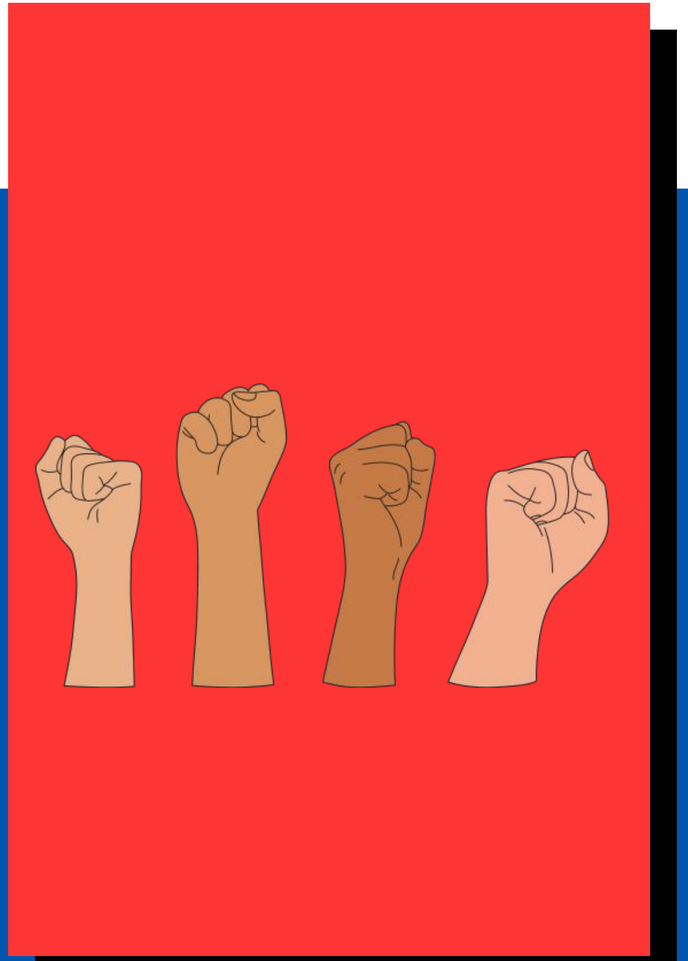
@rabbigreenberg

WHAT TO DO WITH YOUR DATA

What happens after you collect?

The new golden rule:

Your data is yours

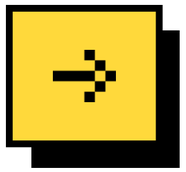


@rabbigreenberg

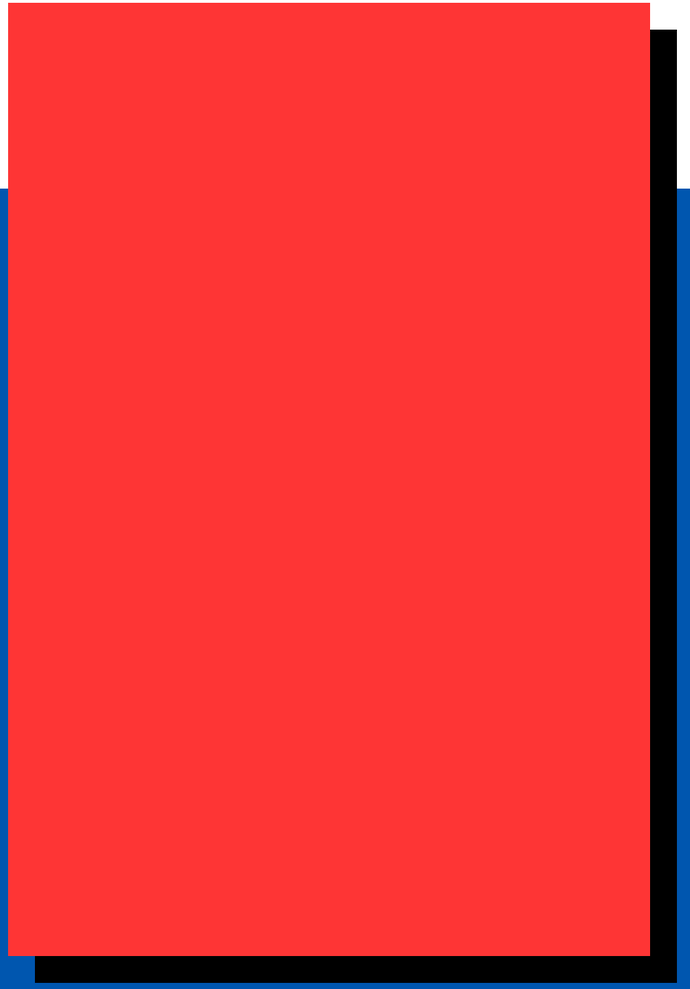
Build Your Own

Observability Backend

@rabbigreenberg



Uniform specs make building your own backend possible



@rabbigreenberg

Bring your data to the vendor of your choice

@rabbigreenberg

Vendors that support OpenTelemetry

APPDYNAMICS

logz.io

Azure

dynatrace

DATADOG

Grafana

new relic

SENTRY

@rabbigreenberg

WHY DOES THIS MATTER?

Aleksey Vorona

InformationWeek

"Tool proliferation remains of the challenging **double-edged swords** facing DevOps-minded teams."

@rabbigreenberg

Nana Janashia

Techworld with nana

"... the DevOps space is developing really fast and we see new technologies, new concepts emerging all the time, so many engineers feel pressure to learn new tools to use in their projects."

@rabbigreenberg

OPENTELEMETRY PROVIDES YOU WITH THE DATA YOU NEED...

Without The Tool Tax

@rabbigreenberg

THANK YOU!

@rabbigreenberg