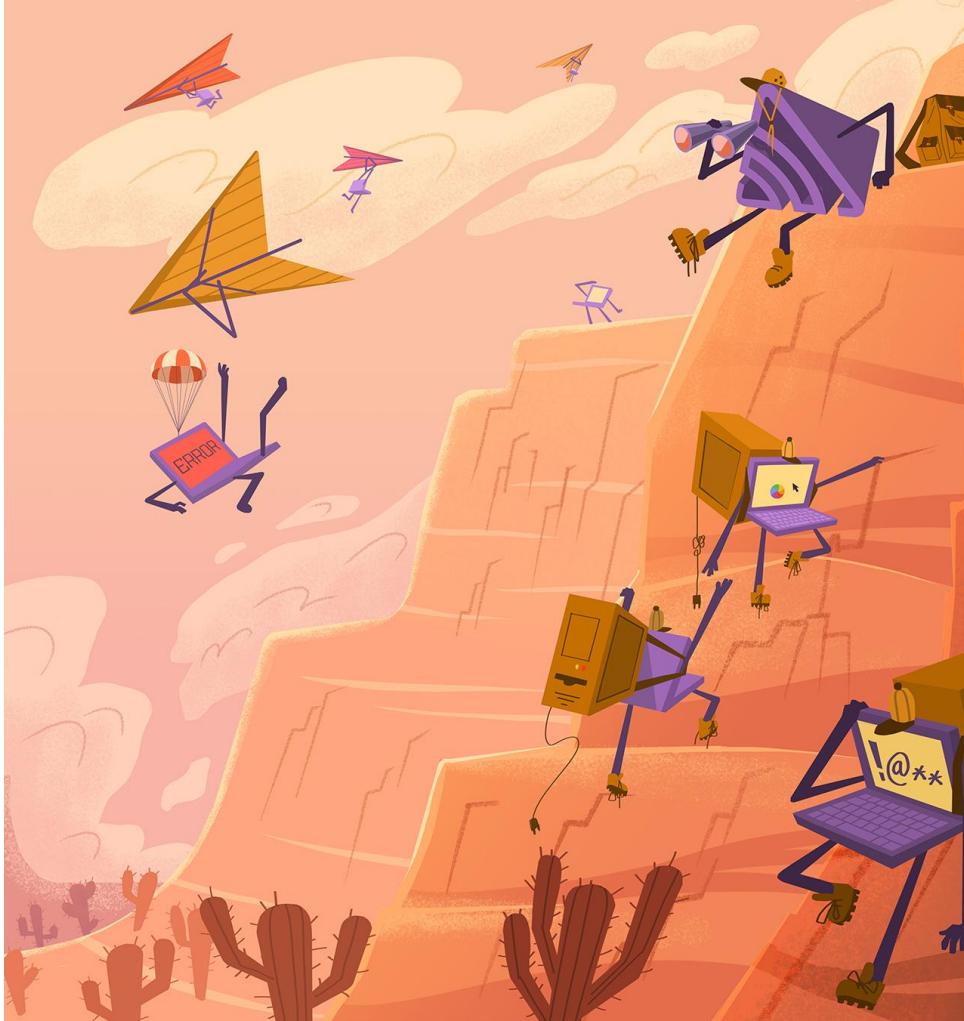


A pragmatic primer to

Distributed Tracing & OpenTelemetry

Daniel Khan
@dkhan





About me

Mühlviertel, 4 Kids, 1 puppy (since last Friday)



2015 - 2021: Director Technology Strategy



since 2016: Course Author



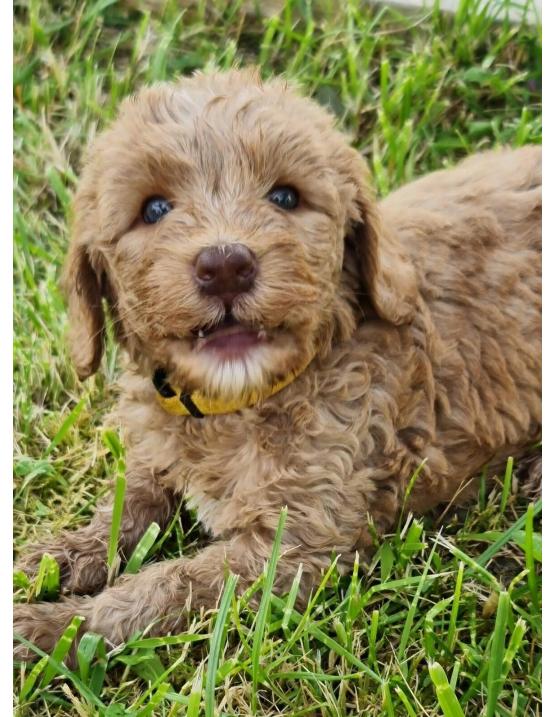
2019 - 2021: Co-Chair of W3C Distributed Tracing



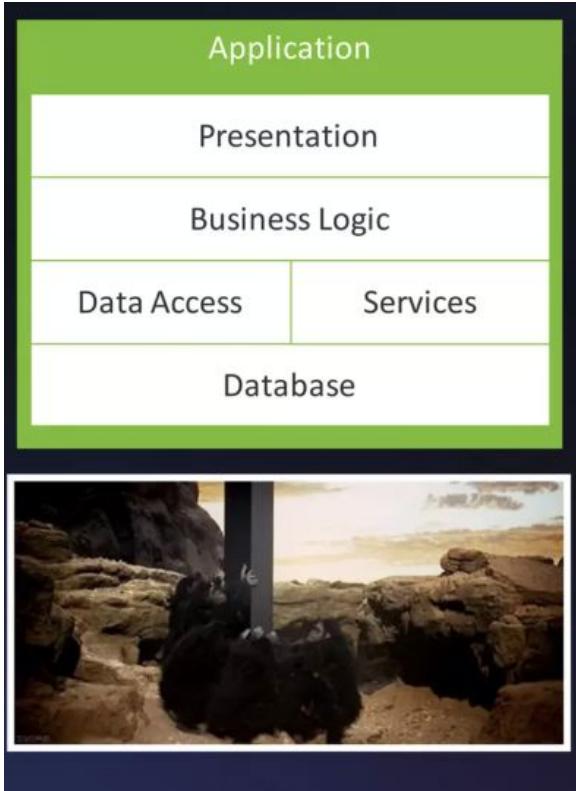
2021-2022: Director Product Management, Unified Observability



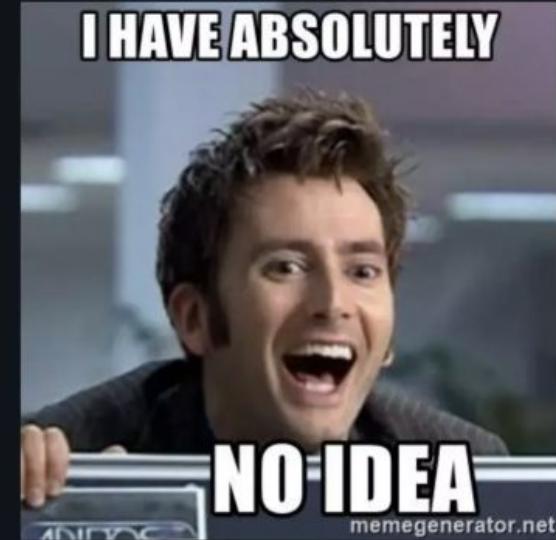
since 2022: Director Product Management, Telemetry



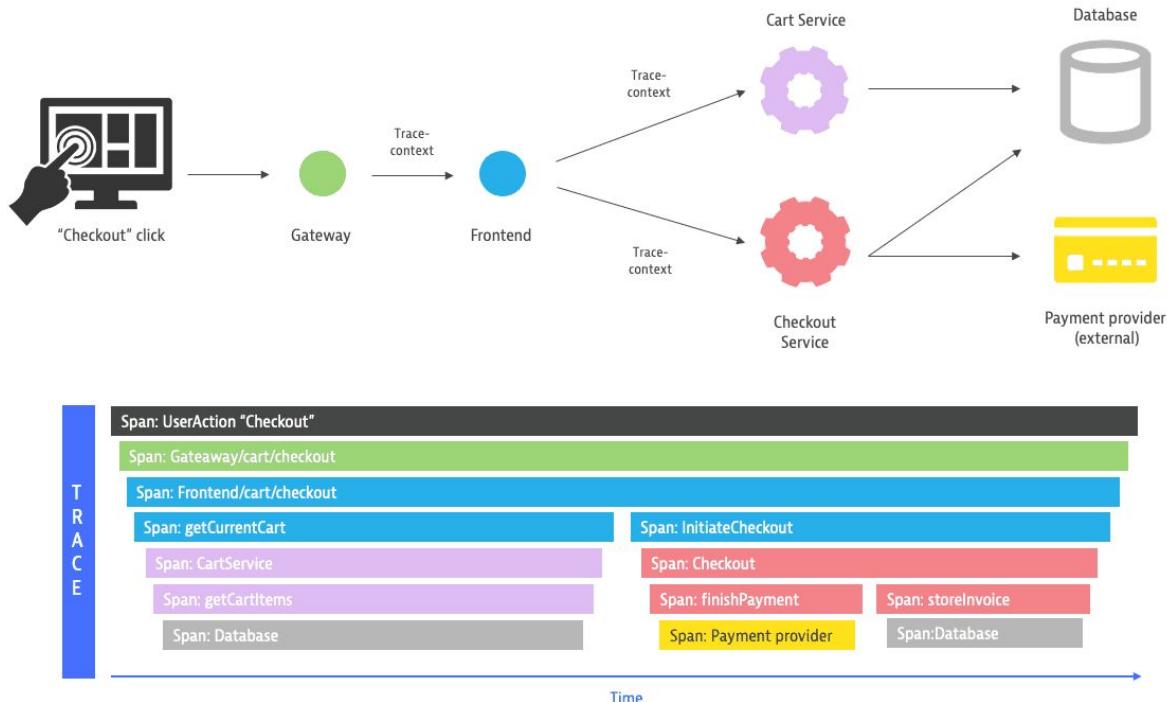
A brief history of (distributed) tracing



Cart Service

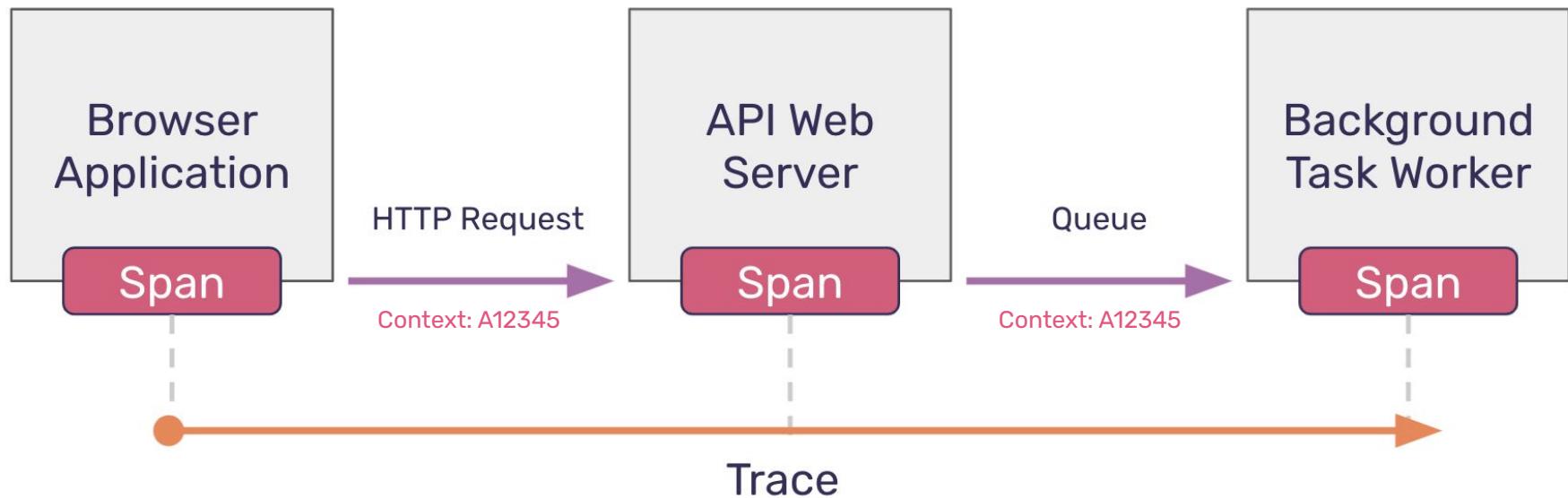


Understanding distributed execution

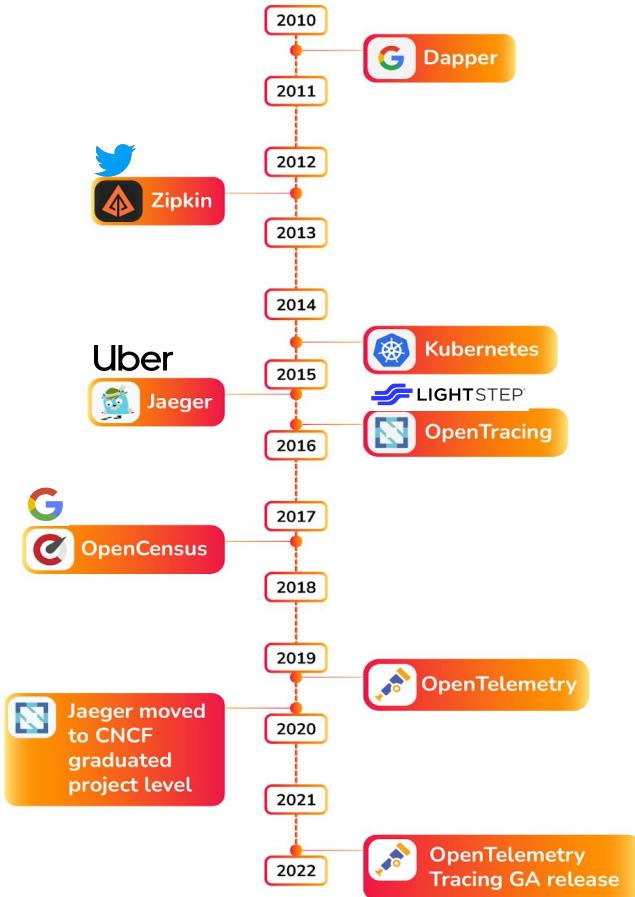


Source: <https://www.dynatrace.com/support/help/observe-and-explore/purepath-distributed-traces>

Context Propagation

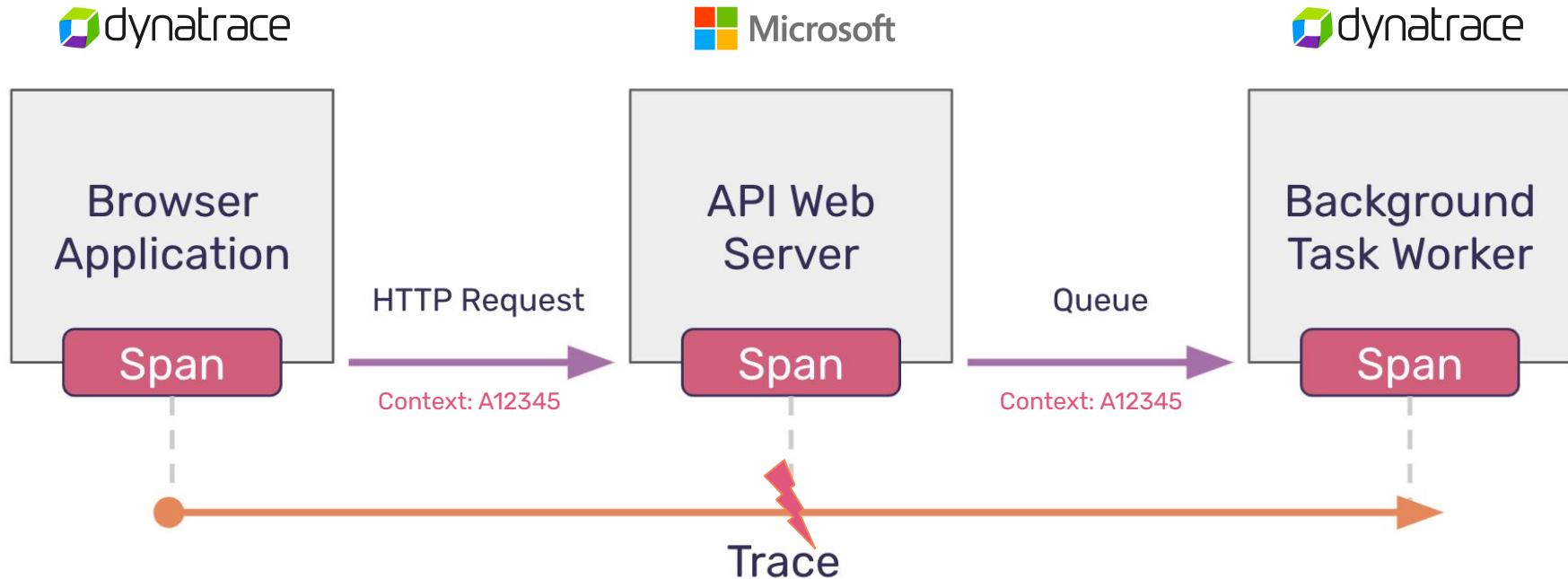


History



Source: <https://tracetest.io/blog/tracing-the-history-of-distributed-tracing-opentelemetry>

Multi Vendor Scenarios



W3C Distributed Tracing



Google Cloud



AppDynamics



New Relic



**Abstract****Status of This Document**

1.	Conformance
2.	Overview
2.1	Problem Statement
2.2	Solution
2.3	Design Overview
3.	Trace Context HTTP Headers Format
3.1	Relationship Between the Headers
3.2	Traceparent Header
3.2.1	Header Name
3.2.2	traceparent Header Field Values
3.2.2.1	version
3.2.2.2	version-format
3.2.2.3	trace-id
3.2.2.4	parent-id
3.2.2.5	trace-flags
3.2.2.5.1	Sampled flag
3.2.2.5.2	Other Flags
3.2.3	Examples of HTTP traceparent Headers
3.2.4	Versioning of traceparent
3.3	Tracestate Header
3.3.1	Header Name
3.3.1.1	tracestate Header Field Values
3.3.1.2	list
3.3.1.3	list-members
3.3.1.3.1	Key
3.3.1.3.2	Value
3.3.1.4	Combined Header Value
3.3.1.5	tracestate Limits:
3.3.2	Examples of tracestate HTTP Headers
3.3.3	Versioning of tracestate
3.4	Mutating the traceparent Field
3.5	Mutating the tracestate Field

Trace Context

W3C Recommendation 23 November 2021

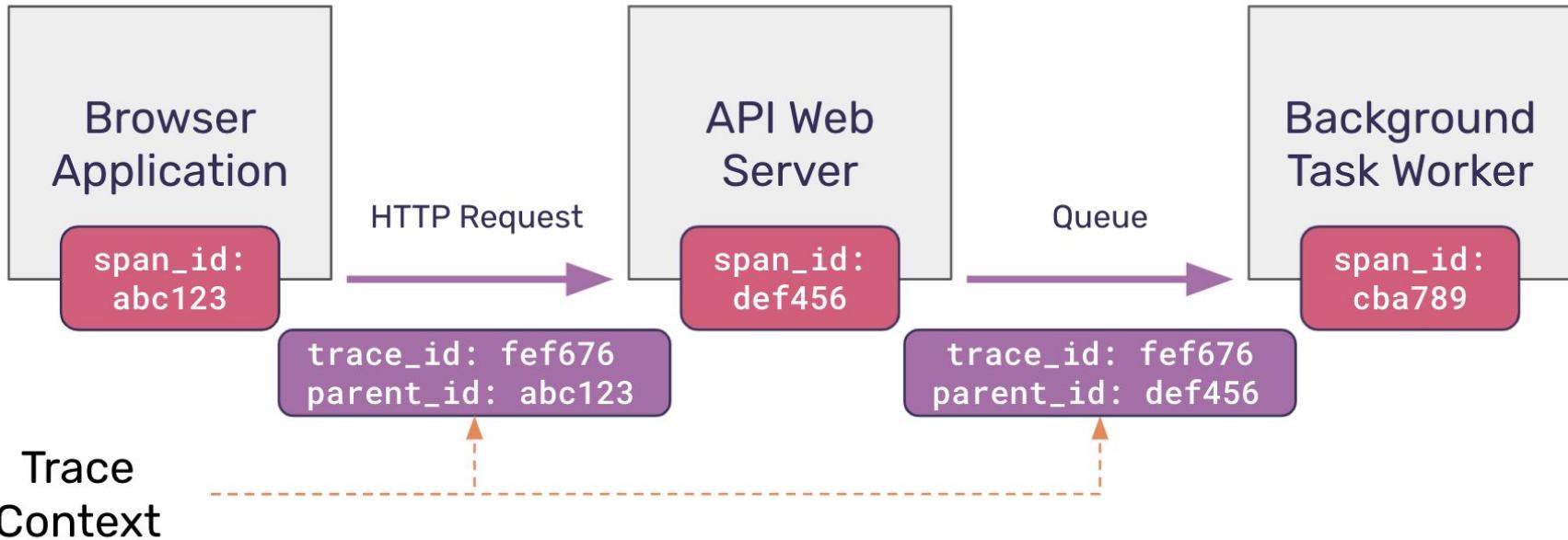
**▼ More details about this document****This version:**<https://www.w3.org/TR/2021/REC-trace-context-1-20211123/>**Latest published version:**<https://www.w3.org/TR/trace-context-1/>**Latest editor's draft:**<https://w3c.github.io/trace-context/>**History:**<https://www.w3.org/standards/history/trace-context-1>[Commit history](#)**Implementation report:**<https://github.com/w3c/trace-context/#reference-implementations>**Editors:**Sergey Kanzhelev ([Microsoft](#))Morgan McLean ([Google](#))Alois Reitbauer ([Dynatrace](#))Bogdan Drutu ([Google](#))Nik Molnar ([Microsoft](#))

Yuri Shkuro (Invited Expert)

Feedback:[GitHub w3c/trace-context](#) (pull requests, new issue, open issues)public-trace-context@w3.org with subject line trace-context ([archives](#))**Errata:**[Errata exists.](#)**Discussions**[We are on Gitter.](#)**See also** [translations](#).Copyright © 2021 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C liability, trademark and permissive document license rules apply.

Abstract

W3C Trace Context



GET /api/recommendations

Event ID	Event Duration
39d50c0b	4.92min

Browser ?
Python Requests
2.28

Trace Navigator

This Event — 6 Children — 1 Descendant

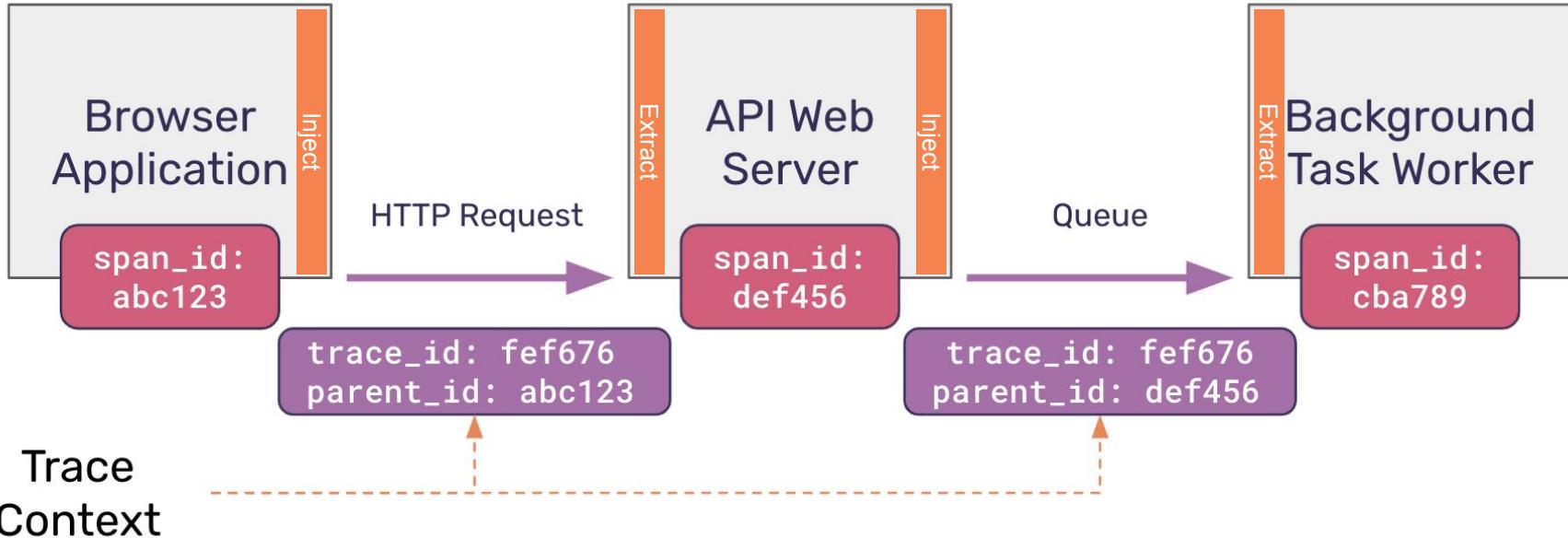
≡ Filter ▾

The screenshot shows a distributed tracing visualization with a timeline at the top and a call graph below it. The timeline spans from 0.00ms to 295,174.19ms. A single orange bar at the top represents an 'rpc' call that took 295,168.25ms and accounted for 100% of the time.

The call graph details the breakdown of this time across different service components:

- http.server - 27a7f810f71552c7**: Total duration 295,174.19ms.
- rpc - grpc.hipstershop.RecommendationService/ListRecommendations**: Total duration 295,162.20ms.
- /hipstershop.RecommendationService/ListRecommendations**: Total duration 163,629.81ms.
 - get_product_list - get_product_list**: Duration 24.31ms.
 - /hipstershop.FeatureFlagService/GetFlag**: Duration 11.85ms.
 - /hipstershop.ProductCatalogService/ListProducts**: Duration 3.86ms.
- default - e8c585446e83488f**: Duration 26.13ms.
- rpc - grpc.hipstershop.ProductCatalogService/GetProduct**: Duration 6.05ms.
 - rpc - 74dfd988f0aaf31**: Duration 0.78ms.
- rpc - grpc.hipstershop.ProductCatalogService/GetProduct**: Duration 5.00ms.
 - rpc - ee3c8a9182d83ae3**: Duration 0.51ms.
- rpc - grpc.hipstershop.ProductCatalogService/GetProduct**: Duration 4.22ms.
 - rpc - 48e11df90cbb964c**: Duration 0.25ms.
- rpc - grpc.hipstershop.ProductCatalogService/GetProduct**: Duration 3.46ms.
 - rpc - 19eb8d76e9545246**: Duration 0.16ms.

Tag Details	
browser	Python Requests 2.28
browser.name	Python Requests
environment	production-dev
level	info
os	Alpine Linux 3.17.3
os.name	Alpine Linux
release	6NRrBDwJzzLQFy... ...
runtime	node v18.16.0
runtime.name	node
server_name	otel-demo-frontend...
transaction	GET /api/recommen...
url	http://otel-demo... 



2018/2019: OpenTelemetry enters the stage

Google



OpenCensus
(Metrics, Traces)



Morgan McLean

LIGHTSTEP



OpenTracing
(Traces)



Ted Young



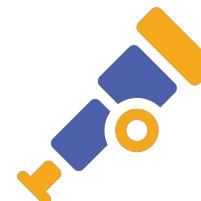
CLOUD NATIVE
COMPUTING FOUNDATION



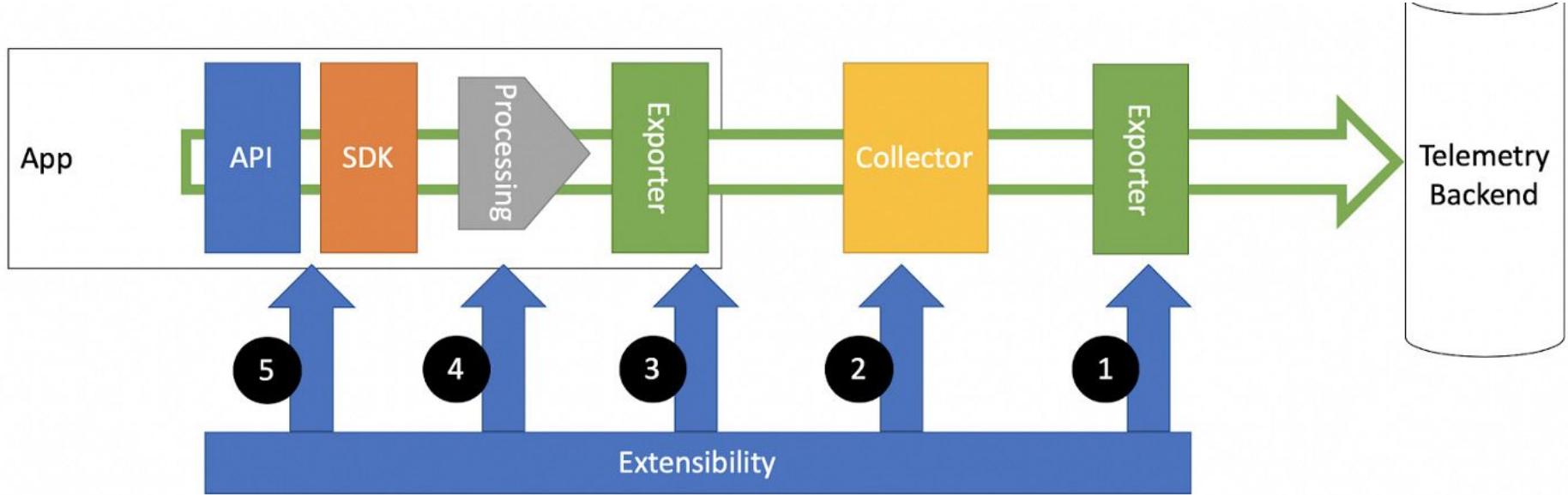
Jaeger
(Tracing UI)



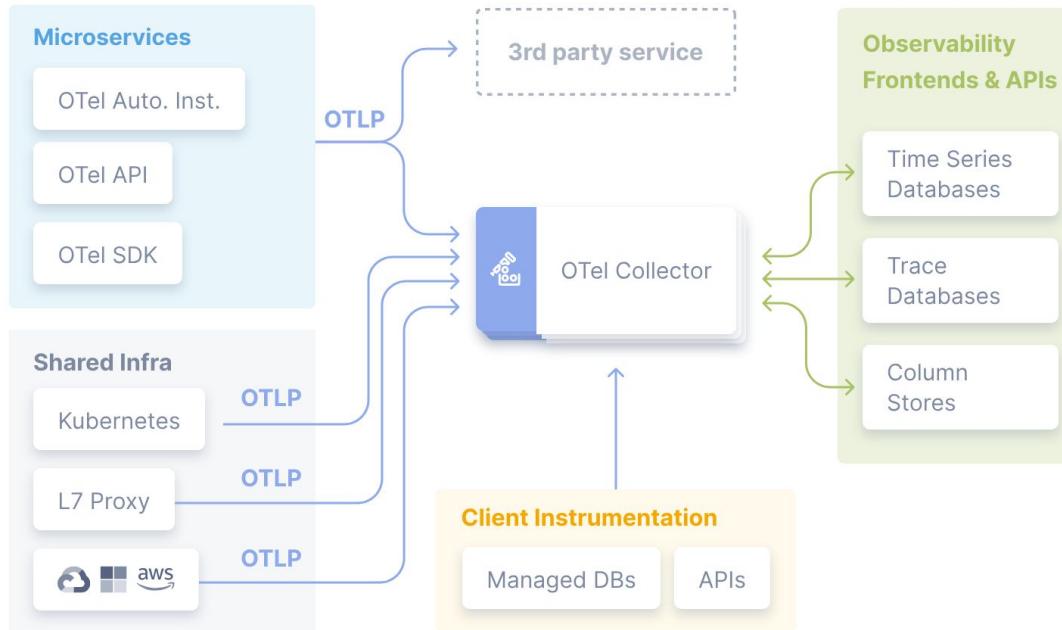
Yuri Shkuro

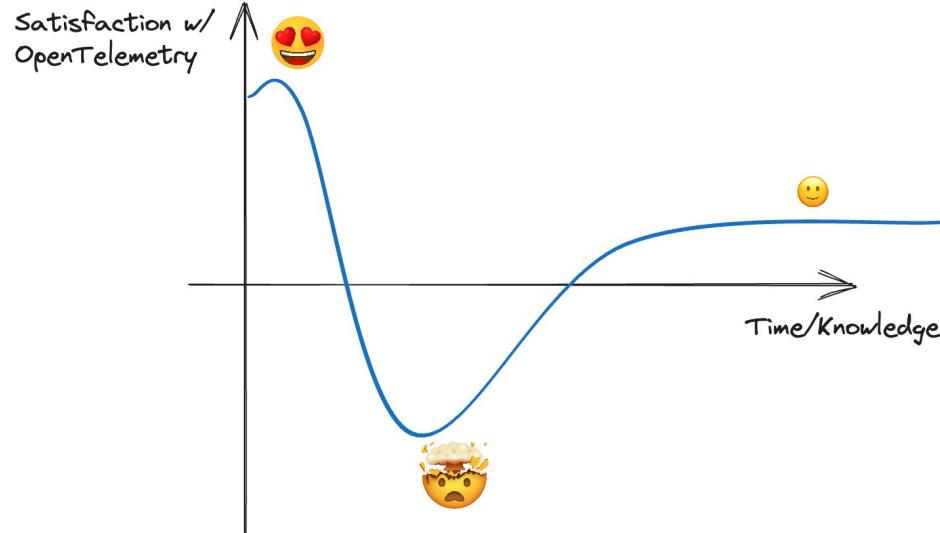


OpenTelemetry
(Metrics, Logs,
Traces)



OTel as open, extendable standard







KubeCon



CloudNativeCon

North America 2019

Keynote: (Open)Telemetry Makes Observability Simple

Sarah Novotny, Open Source Wonk, Azure OCTO, Microsoft & Liz Fong-Jones, Principal Developer Advocate, Honeycomb.io

Play (k)

▶ ▶ | 0:01 / 19:54

▶ CC HD □ □ □ □



Source: https://www.youtube.com/watch?v=W_8MHdtrgZE

- Finally the problem of distributed tracing has been solved.
Despite the problem has been solved more than a decade ago.
- OpenTelemetry is the solution to this.
Despite having very limited features compared to vendors.
- (We built it.)
Despite being a community effort.

We need new tooling and we need the ability to observe and to look inside our systems so that we know what's happening under the hood.

This sounds obvious but if it's so obvious, why haven't we done it already?

Because it's hard.

```
otel-kubcon-demo ✘ Show ✘ src/rain.go
○ 1  "http/httputil"
○ 2  "os"
○ 3  "strconv"
● 4  "sync"
○ 5  "go.opentelemetry.io/otel/plugin/httputp"
○ 6  "sdkttrace" "go.opentelemetry.io/otel/sdk/trace"
○ 7  "go.opentelemetry.io/otel/global"
○ 8  "go.opentelemetry.io/otel/exporter/trace/jaeger"
○ 9  "go.opentelemetry.io/otel/exporter/trace/stackdriver"
○ 10 "go.opentelemetry.io/otel/api/kaf"
○ 11 "go.opentelemetry.io/otel/api/trace"
○ 12 "go.opentelemetry.io/otel/plugin/httputp"
○ 13 "github.com/CloudNativeCon/lightstep/opentelemetry-exporter-go/..."
○ 14 )

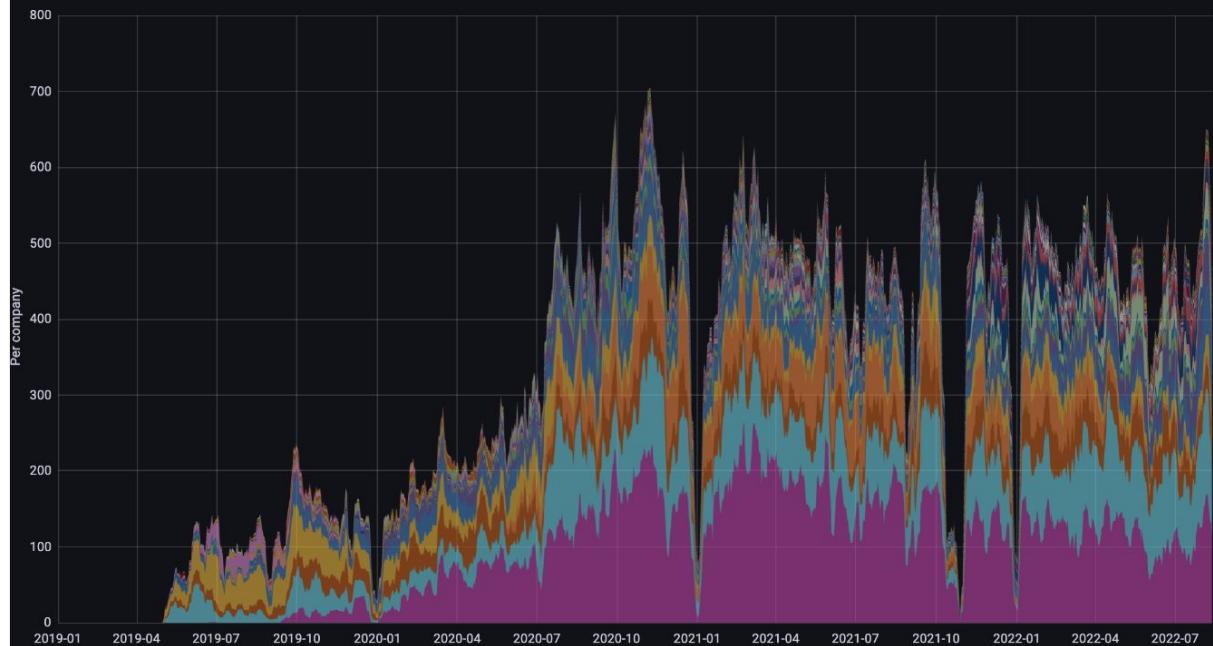
func main() {
    serviceName, _ := os.LookupEnv("PROJECT_NAME")
    jExporter, _ := jaeger.NewExporter(
        jaeger.Configuration{
            ServiceName: serviceName,
        },
        jaeger.Reporter(jaeger.StdReporter{}),
    )
    jExporter.Start()
}
```

Logs

```
[rain.go:16:2] imported and not used: "go.opentelemetry.io/otel/api/trace"
[rain.go:16:29] undefined: context
[command-line-arguments]
[rain.go:77:39] undefined: key
[rain.go:77:29] undefined: context
[command-line-arguments]
[rain.go:77:39] undefined: key
[rain.go:77:29] undefined: context
[command-line-arguments]
[rain.go:77:39] undefined: key
[rain.go:77:29] undefined: context
```

And then they go on trying to instrument a go app and it doesn't look easy and doesn't work either.

Companies Contributions metric (All, 7 Days MA)



	min	max	avg	current	total
Splunk Inc.	0	265.57	101.51	109.14	122.01 K
Microsoft Corporation	0	172.14	62.74	79.29	75.41 K
LightStep Inc.	0	74.43	29.99	21.29	36.05 K
Amazon	0	112.71	28.60	10.29	34.37 K
Google LLC	0	76.86	25.30	27.29	30.41 K
Dynatrace LLC	0	73.00	23.47	40.86	28.21 K
New Relic Inc.	0	69.00	12.29	7.00	14.78 K
Independent	0	32.00	7.91	7.43	9.50 K
Red Hat Inc.	0	41.86	7.08	7.00	8.51 K
observIQ	0	51.57	6.72	37.00	8.07 K
Shopify Inc.	0	28.71	4.96	6.14	5.96 K
Raintank Inc. – Grafana Labs	0	52.43	4.88	15.29	5.86 K
Datadog Inc	0	24.29	4.41	10.29	5.30 K
Toptal LLC	0	20.00	4.27	0	5.14 K
Cisco	0	25.43	3.43	10.86	4.13 K
Sumo Logic Inc.	0	14.86	2.24	1.14	2.70 K
MailChimp	0	11.29	1.99	1.57	2.40 K
Postmates Inc	0	15.00	1.77	0.86	2.12 K
Atlassian	0	34.14	1.74	0.86	2.09 K
Aspecto	0	15.86	1.70	2.14	2.04 K
Hound Technology Inc. dba Honeycomb	0	14.00	1.52	5.29	1.83 K
Omnition	0	38.57	1.48	0	1.78 K

Industry Reaction

Since 2019:

Many new vendors enter the market.

September 2019:

Splunk acquires **SignalFX** and **Omnition**.

May 2021:

ServiceNow acquires **LightStep**.

Figure 1: Magic Quadrant for Application Performance Monitoring Suites



Figure 1: Magic Quadrant for Application Performance Monitoring and Observability



Source: Gartner (March 2018)

Source: Gartner (June 2022)

	min	max	avg	current	total
Splunk Inc.	55.43	169.14	105.21	137.86	9.47 K
Microsoft Corporation	57.29	142.71	101.50	131.29	9.13 K
LightStep Inc.	21.71	61.14	39.50	29.43	3.55 K
New Relic Inc.	21.00	69.00	37.80	54.86	3.40 K
Dynatrace LLC	13.57	71.43	28.93	57.00	2.60 K
observIQ	5.29	51.57	27.54	30.86	2.48 K
Google LLC	6.00	36.43	17.50	24.71	1.58 K
Cisco	3.43	25.43	14.15	8.57	1.27 K
Independent	5.57	21.71	12.94	5.86	1.16 K
Datadog Inc	1.00	24.29	12.11	11.86	1.09 K
Amazon	3.29	16.71	9.64	11.00	867.77
Red Hat Inc.	0.71	17.29	7.69	10.14	692.20
Shopify Inc.	0.57	15.43	6.82	8.43	614.20
Raintank Inc. – Grafana Labs	0	27.57	6.47	10.86	582.64
Sumo Logic Inc.	0.57	11.71	4.54	0.71	408.69
Atlassian	0	16.57	3.66	0.29	329.60
Tencent	0.29	7.43	2.91	1.00	261.86
Aspecto	0	9.29	2.86	0	257.71
Sentry	0	8.43	2.49	2.29	224.00
DaoCloud Network Technology Co. Ltd.	0	6.57	2.20	0.71	197.56
MailChimp	0	5.00	1.91	0	172.12
Salesforce.com Inc.	0	6.71	1.42	0	127.56

THE STATE OF OPENTELEMETRY PER SIGNAL



Generally Available

- API, SDK, & Protocol are stable
- Client libraries v1.0+ exist for Java, Go, .Net, Python, C++, JavaScript, Ruby, Erlang, Swift (and maybe more)
- Long-term support, backwards compatibility, and dependency isolation guarantees

Release Candidate

- RC announced May '22
- GA expected soon
- API, SDK, & Protocol are stable
- Client libraries in Java, .NET, and Python exist in RC (and maybe more)
- Prometheus support

Experimental

- Protocol is stable, API & SDK in draft
- Focusing first on integration with existing logging systems
- Log appenders are under development in many languages
- Next focus: a new strongly-typed and machine-readable format

Source: <https://logz.io/learn/opentelemetry-guide/#current>

OpenTelemetry **does not** provide

- A backend to store the data
- Analytics on top of this data
- A user interface
- Proper frontend/mobile monitoring (as of today)

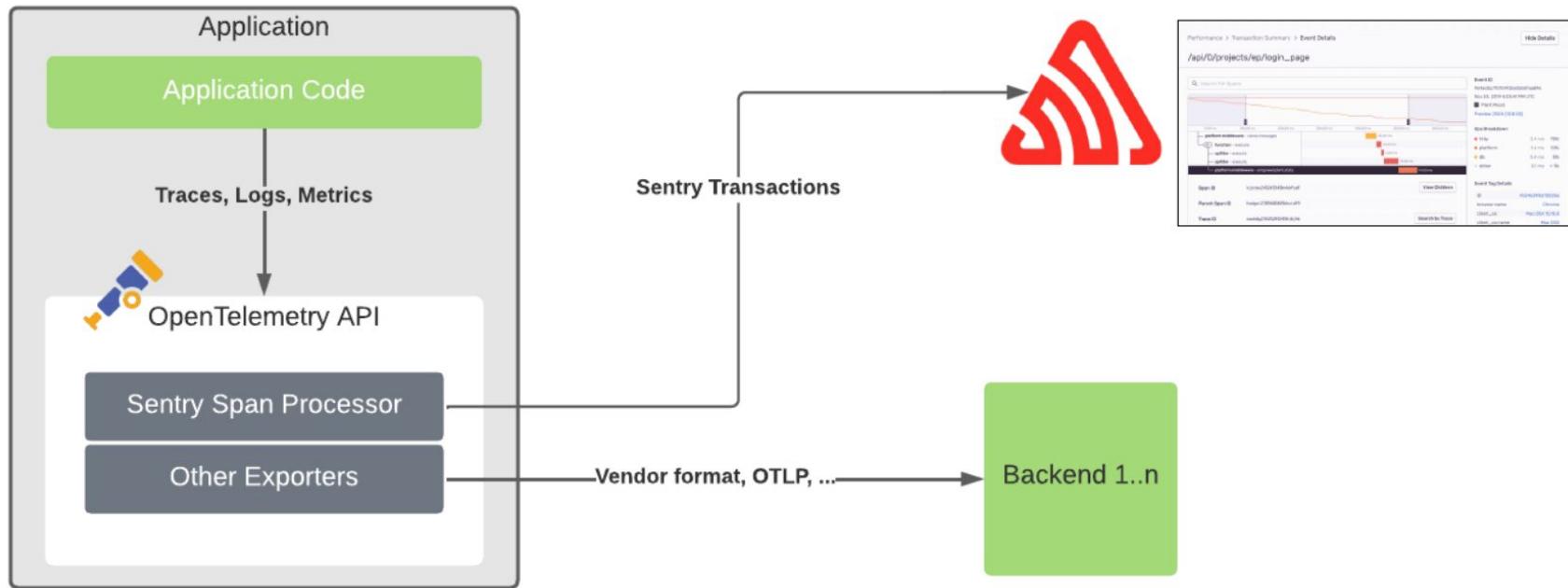
OpenTelemetry provides

- A unified way for libraries and components to add observability
- Auto-Instrumentation for many platforms and frameworks
- A SDK for vendor neutral manual instrumentation
- A data format (OTLP) to exchange telemetry data
- A way to send data to multiple observability backends

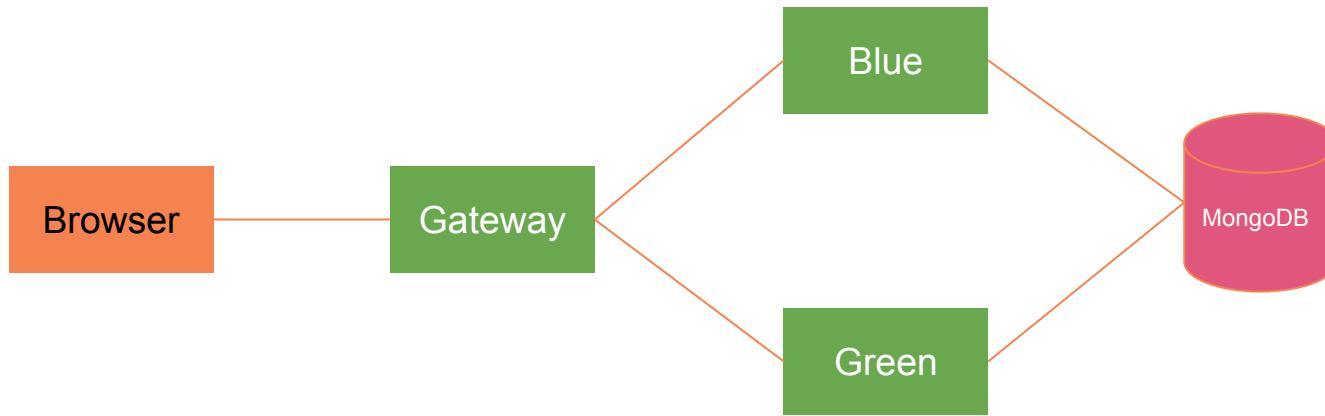
**“We want every platform and library to
be pre-instrumented with
OpenTelemetry and we’re committed to
making this as easy as possible.”**

Sergey Kanzhelev (Google)

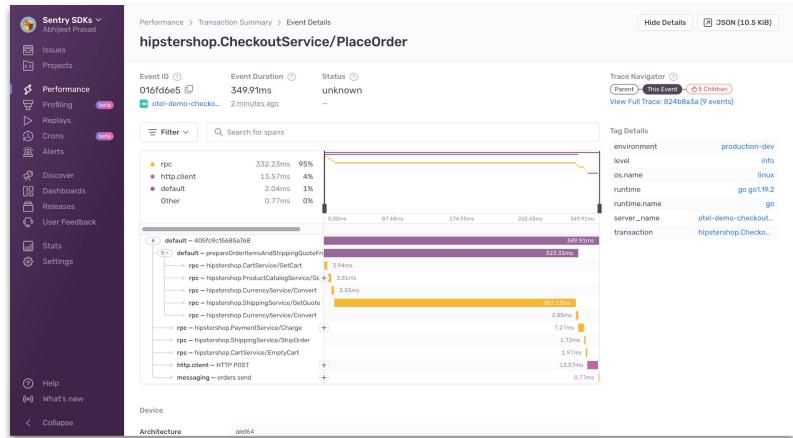
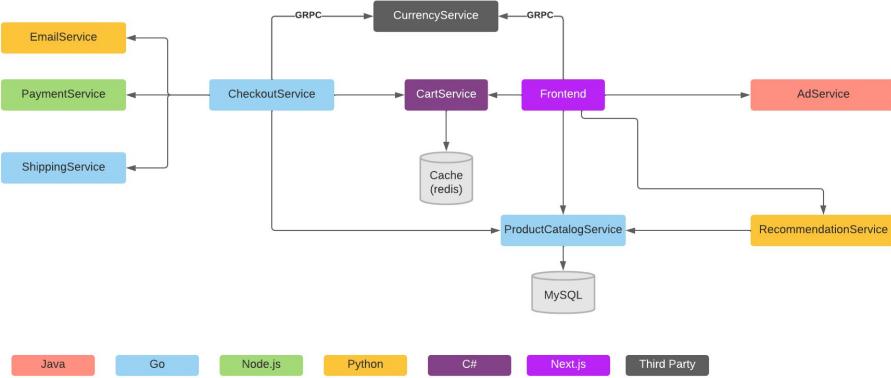
OpenTelemetry at Sentry



Demo



Official OTel Microservices Demo



<https://github.com/getsentry/opentelemetry-demo>

Thank you!
Questions?