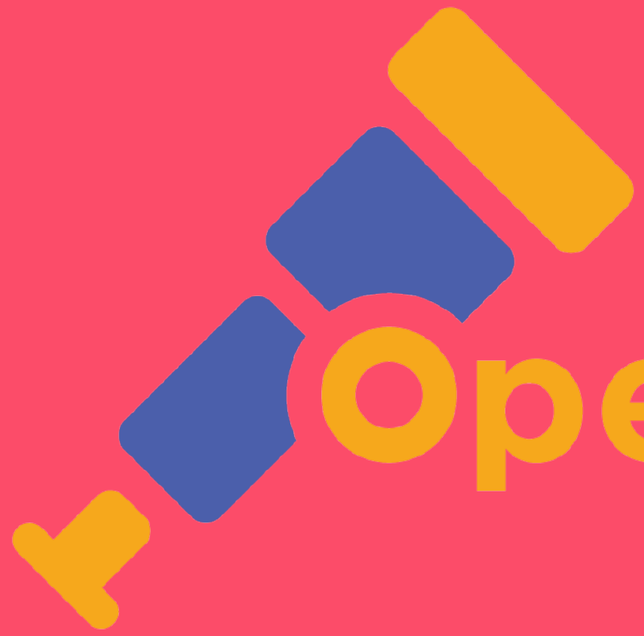
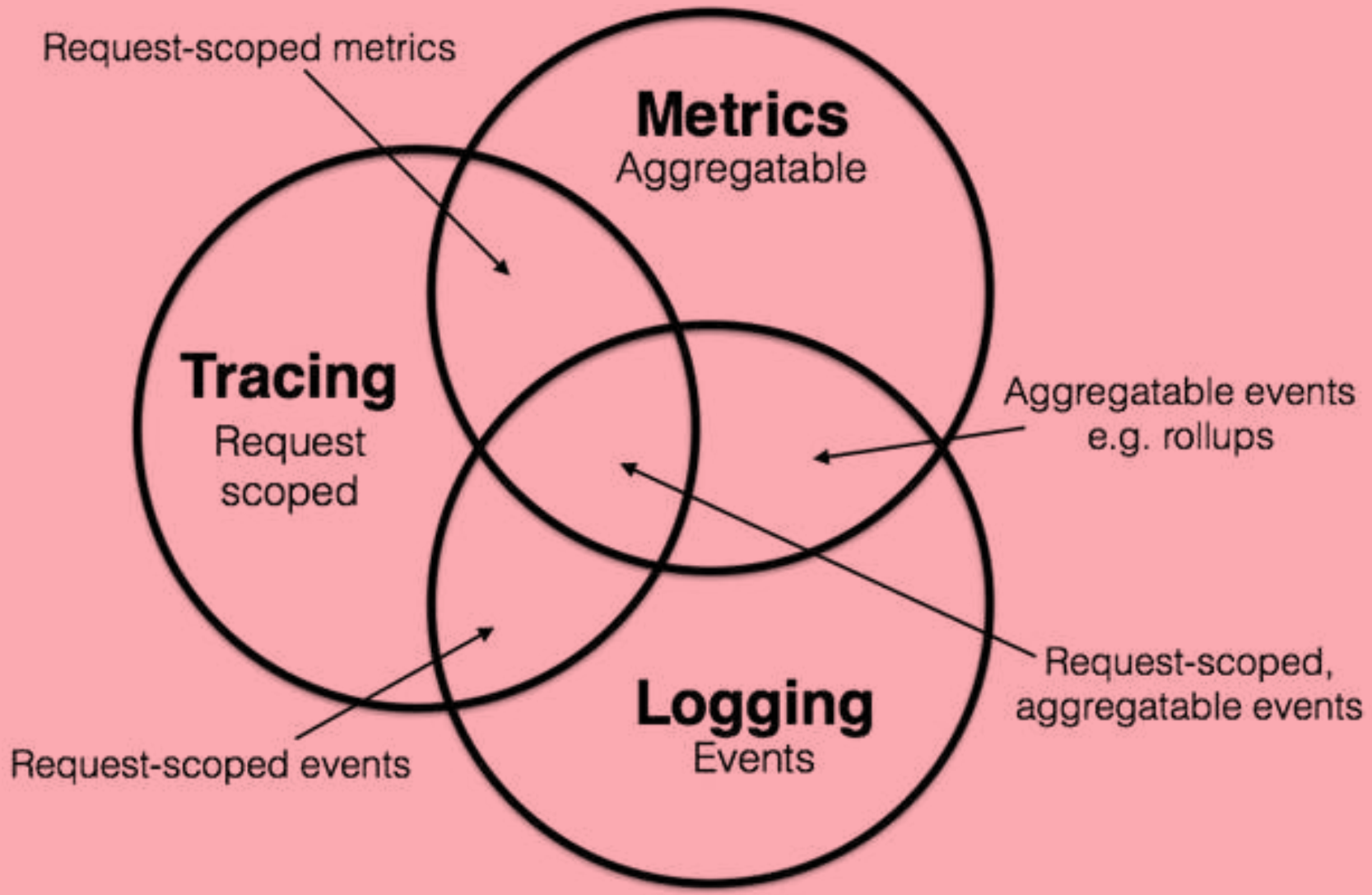
The State of



OpenTelemetry

Philipp Krenn @xeraa

Observability



https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html

[...] this seems like calling "gasoline, motor oil, and tires" the three pillars of F1 racing. It's not wrong, precisely, but ...

— https://twitter.com/fuzzychef/status/1186403652124069888

Observability

System attribute that you cannot buy*

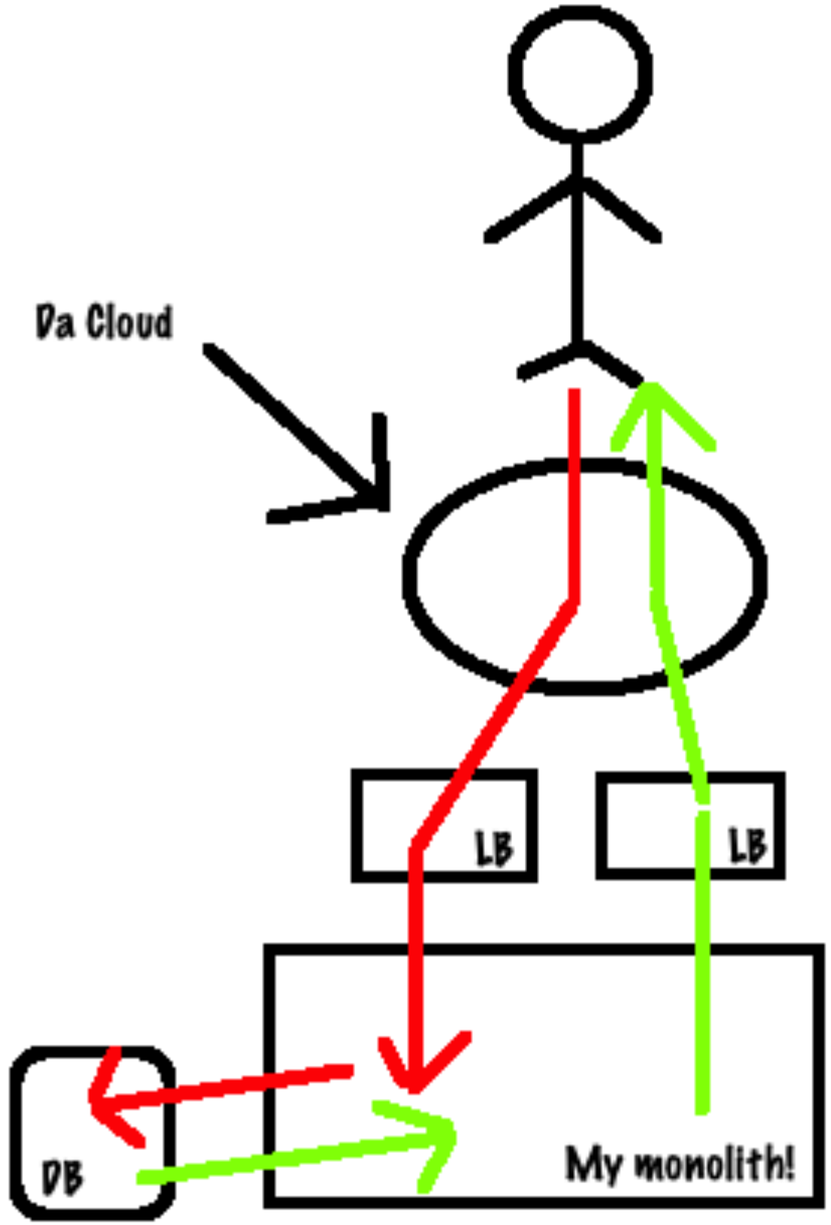
Monitoring is your bank telling you you're overdrawn.

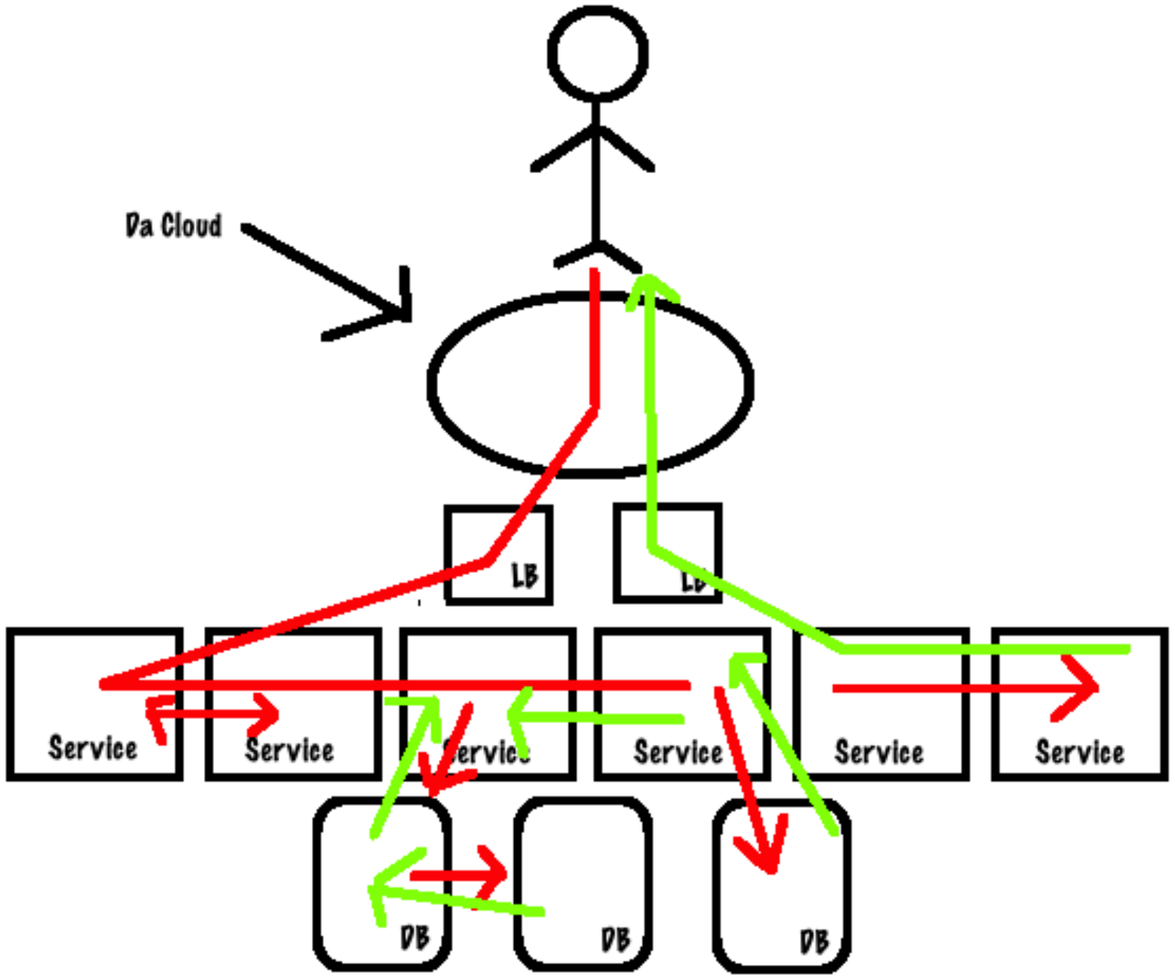
Observability is the ability to tell you're running out of money because you're spending too much money on chocolates, cakes and sweets because you've recorded data on what you spent your money on throughout the month.

— https://twitter.com/lizthegrey/status/1230979460708499456

Simpler Times

https://www.kartar.net/2019/07/intro-to-distributed-tracing/





Better Times for Vendors

https://www.kartar.net/2019/07/intro-to-distributed-tracing/

@xeraa

elastic

Developer 🥑

APM / Traces

Application Performance Monitoring

Distributed Tracing

elastic

@xeraa

Goals

Transaction context

Reconstruct flow

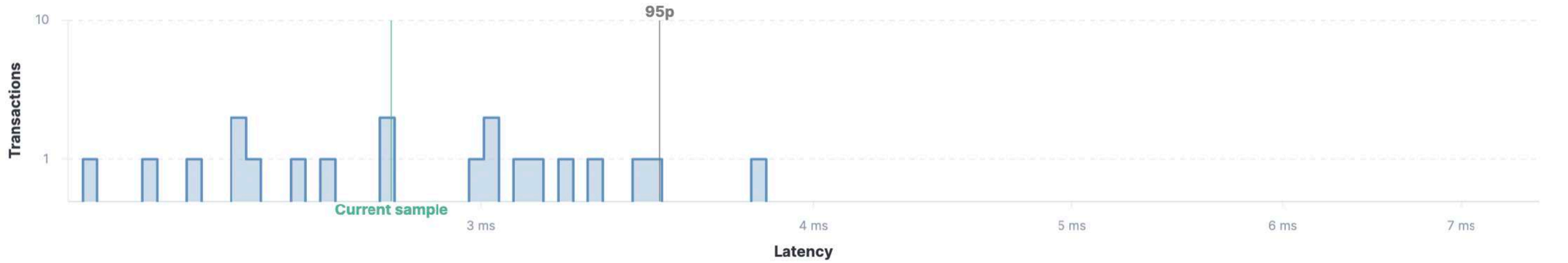
Query & visualize transactions

Trace samples | Latency correlations | Failed transaction correlations

Latency distribution

20 total transactions

Click and drag to select a range



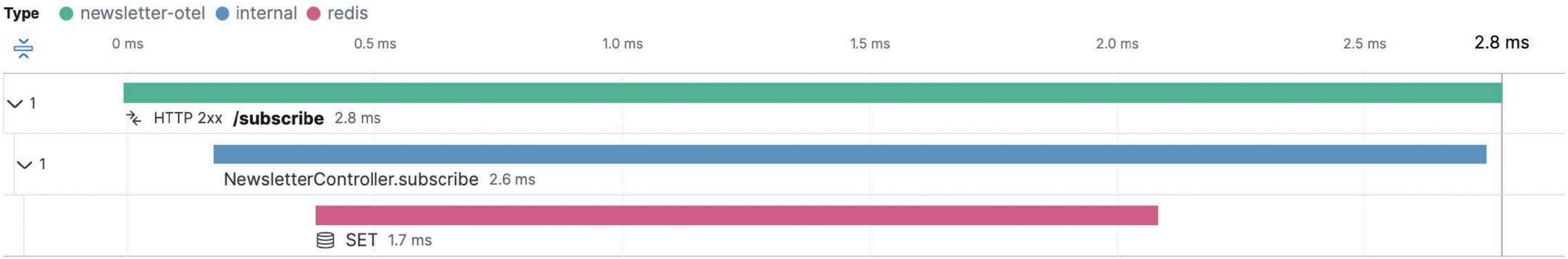
Trace sample

1 of 20

Investigate | View full trace

3 minutes ago | 2.8 ms (3.9% of trace) | PUT http://newsletter:8080/subscribe | 200 OK | Python Requests (2.21)

Timeline | Metadata | Logs



Agents

Language & framework specific

Detect start & end of request, capture errors

Agents

Wrap operations in standard & known 3rd party libraries

Extract additional information

Agents

Little to no overhead

Trace & hook, not profile

Distributed Tracing

elastic

@xeraa

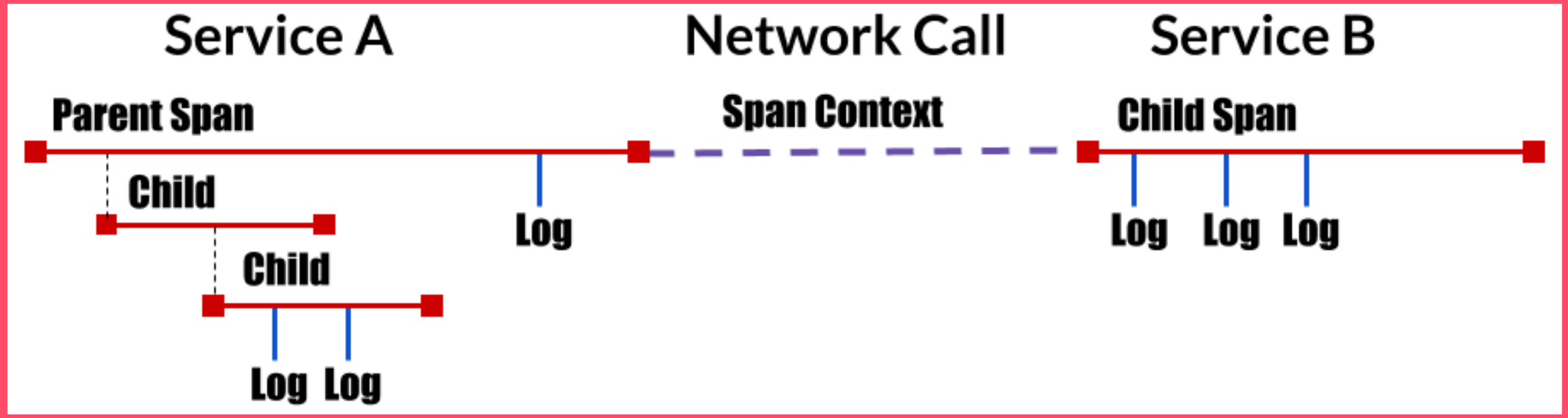
Terminology

Trace: record of a request, DAG

Span: named & timed single operation, optionally nested

Context Propagation: Attaching IDs

Trace



https://opentracing.io/docs/overview/

Problem

uber-trace-id: 118c6c15301b9b3b3:56e66177e6e55a91:18c6c15301b9b3b3:1

elastic-apm-traceparent: 00-f109f092a7d869fb4615784bacefcfd7-5bf936f4fcde3af0-01

OpenTracing

Vendor-neutral APIs for tracing

SpanContext

```
SpanContext:
- trace_id:"0123456789"
- span_id:"abcdef"
- Baggage Items:
  - special_id:"vsid0123"
```

API / Headers Only

No wire protocol
(in- or out-band)

W3C Trace Context

Pass trace context information across systems

Trace Context Propagation

```
traceparent:
00-                                 // Version
0123456789abcdef0123456789abcdef-   // Trace ID
abcdef0123456789-                   // Parent ID
01                                  // Sampling flags
```

CorrelationContext / baggage supported

OpenTelemetry

OSS, CNCF incubating

Traces, metrics, logs

Supersedes OpenTracing + OpenCensus

OpenTelemetry

Unification of instrumentation and data

Vendor neutral but wide support

Google, Splunk, Lightstep, Honeycomb, Elastic,...

Status

Tracing: stable

Metrics: stable / SDKs mixed

Logs: draft

https://opentelemetry.io/status/

Components

OpenTelemetry Protocol (OTLP)

Collector

API, SDK, auto instrumentation

OpenTelemetry Protocol (OTLP)

Protocol for OpenTelemetry data exchange

Supports gRPC and Protobuf or JSON over HTTP

OTLP Hello World

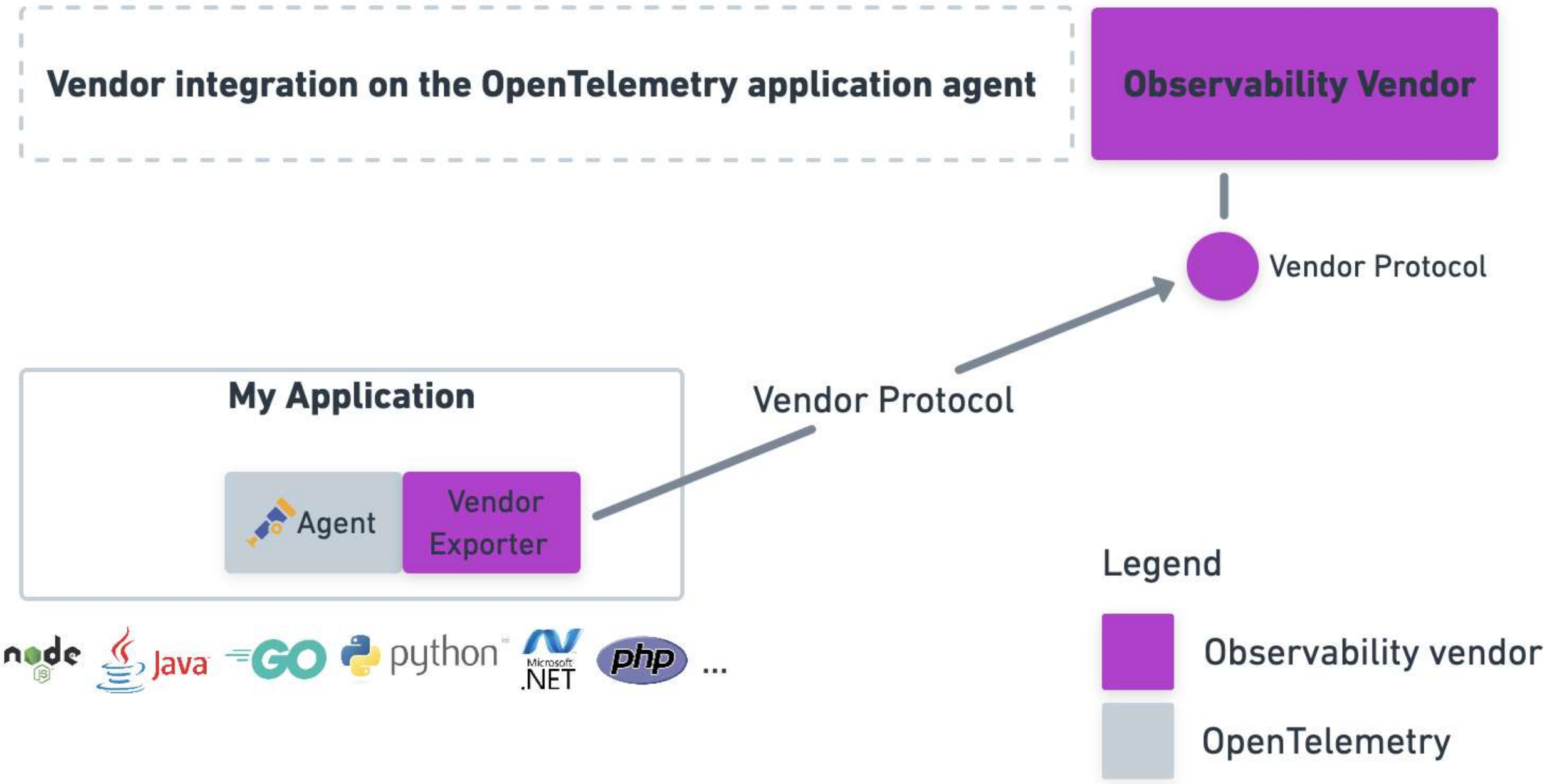
cURL ↦ OTel collector ↦ backend

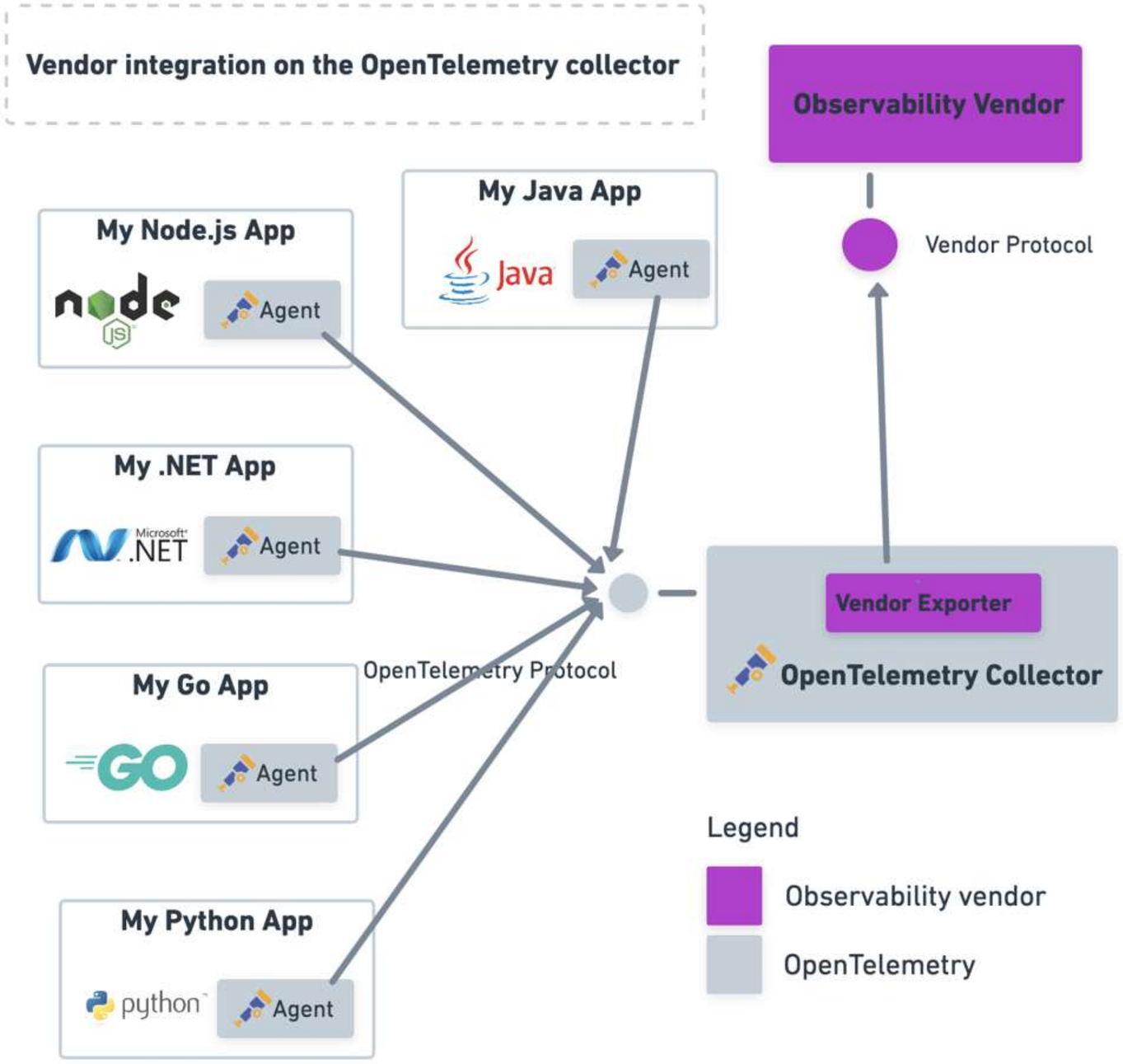
```
curl -i http://localhost:4318/v1/traces \
    -X POST -H "Content-Type: application/json" \
    -d @span.json
```

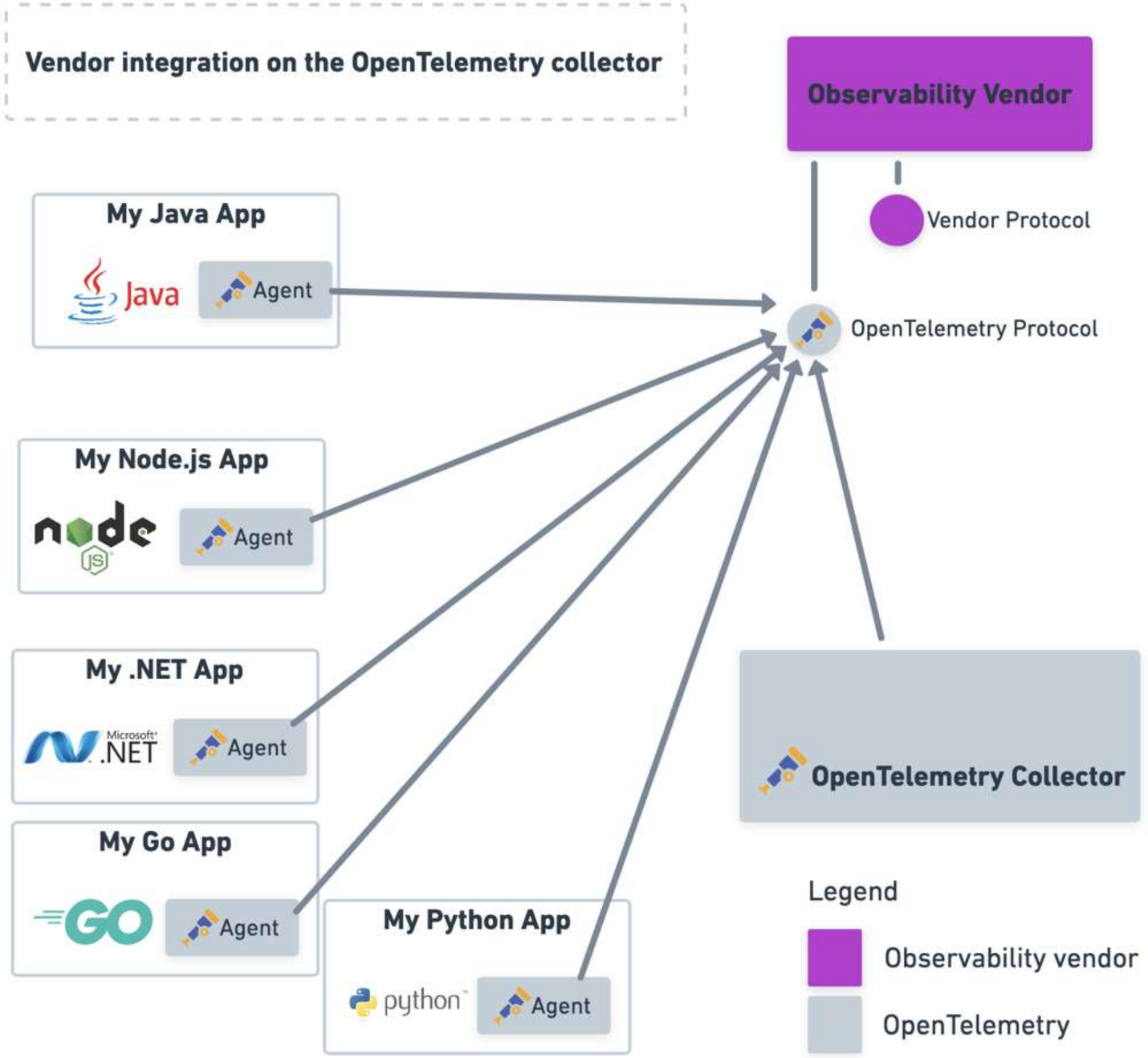
https://github.com/open-telemetry/opentelemetry-proto/blob/main/opentelemetry/proto/trace/v1/trace.proto

3 Ways to OpenTelemetry Integration

Vendors can implement one or more


Vendor integration on the OpenTelemetry application agent
Observability Vendor
Vendor Protocol
My Application
Agent
Vendor Exporter
Vendor Protocol
node
Java
GO
python
Microsoft .NET
php
...
Legend
Observability vendor
OpenTelemetry


Vendor integration on the OpenTelemetry collector
Observability Vendor
Vendor Protocol
My Java App
Java
Agent
My Node.js App
node
Agent
My .NET App
Microsoft .NET
Agent
My Go App
GO
Agent
My Python App
python
Agent
OpenTelemetry Protocol
Vendor Exporter
OpenTelemetry Collector
Legend
Observability vendor
OpenTelemetry


Vendor integration on the OpenTelemetry collector
Observability Vendor
Vendor Protocol
My Java App
Java
Agent
OpenTelemetry Protocol
My Node.js App
Agent
My .NET App
Microsoft .NET
Agent
My Go App
GO
Agent
My Python App
python
Agent
OpenTelemetry Collector
Legend
Observability vendor
OpenTelemetry

API & SDK

.NET, C++, Erlang / Elixir, Go, Java, JavaScript, PHP, Python, Ruby, Rust, Swift

OpenTelemetry for Java

github.com/open-telemetry/opentelemetry-java: standalone library for OTel support

github.com/open-telemetry/opentelemetry-java-instrumentation: auto-instrumentation agent, relying on opentelemetry-java

Java Agent Setup

```
export OTEL_RESOURCE_ATTRIBUTES=service.name=frontend,
                                service.version=1.0
                                deployment.environment=production
export OTEL_EXPORTER_OTLP_ENDPOINT=https://apm_server_url:8200
export OTEL_EXPORTER_OTLP_HEADERS="Authorization=Bearer <secret_token>"

java -javaagent:/path/to/opentelemetry-javaagent-all.jar \
     -classpath lib/*:classes/ \
     com.mycompany.checkout.CheckoutServiceServer
```

Manual Instrumentation

```
Tracer tracer =
    OpenTelemetry.getGlobalTracer("instrumentation-library-name", "1.0.0");

Span span = tracer.spanBuilder("my span").startSpan();
try (Scope scope = span.makeCurrent()) {
  // your use case
  ...
} catch (Throwable t) {
    span.setStatus(StatusCode.ERROR, "Change it to your error message");
} finally {
    span.end(); // closing the scope does not end the span; has to be done manually
}
```

Manual Instrumentation

```
@PostMapping
public ResponseEntity<Order> create(@RequestBody OrderForm form, HttpServletRequest request) {
    Span span = Span.current();

    List<OrderProductDto> formDtos = form.getProductOrders();

    String customerId = "customer-" + RANDOM.nextInt(100);
    span.setAttribute(OpenTelemetryAttributes.CUSTOMER_ID, customerId);

    double orderPrice = formDtos.stream().mapToDouble(po -> po.getQuantity() * po.getProduct()
            .getPrice()).sum();
    String shippingCountry = getCountryCode(request.getRemoteAddr());
    String shippingMethod = randomShippingMethod();
    String paymentMethod = randomPaymentMethod();
```

OTel for JavaScript

https://github.com/open-telemetry/opentelemetry-js-api: server & browser API

https://github.com/open-telemetry/opentelemetry-js: instrumentation SDK

https://github.com/open-telemetry/opentelemetry-js-contrib: auto instrumentation,...

JavaScript Agent Setup

```
npm install --save @opentelemetry/api
npm install --save @opentelemetry/sdk-node
npm install --save @opentelemetry/auto-instrumentations-node
```

```
const process = require('process');
const opentelemetry = require('@opentelemetry/sdk-node');
const { getNodeAutoInstrumentations } = require('@opentelemetry/auto-instrumentations-node');
const { ConsoleSpanExporter } = require('@opentelemetry/sdk-trace-base');
const { Resource } = require('@opentelemetry/resources');
const { SemanticResourceAttributes } = require('@opentelemetry/semantic-conventions');
```

Instrumentation

```
const traceExporter = new ConsoleSpanExporter();
const sdk = new opentelemetry.NodeSDK({
  resource: new Resource({
    [SemanticResourceAttributes.SERVICE_NAME]: 'my-service',
  }),
  traceExporter,
  instrumentations: [getNodeAutoInstrumentations()]
});

sdk.start()
  .then(() => console.log('Tracing initialized'))
  .catch((error) => console.log('Error initializing tracing', error));

process.on('SIGTERM', () => {
  sdk.shutdown()
    .then(() => console.log('Tracing terminated'))
    .catch((error) => console.log('Error terminating tracing', error))
    .finally(() => process.exit(0));
});
```

Demo Time

elastic

@xeraa

https://2354-2-139-188-114.eu.ngrok.io

Conclusion

Distributed Tracing & OpenTelemetry

Why & How

elastic

@xeraa

OpenTelemetry

Now: traces & metrics

Future: logs and more (profiling with eBPF, production debugging, control plane,...)

Differentation on the Backend

Store, search, visualize, manage

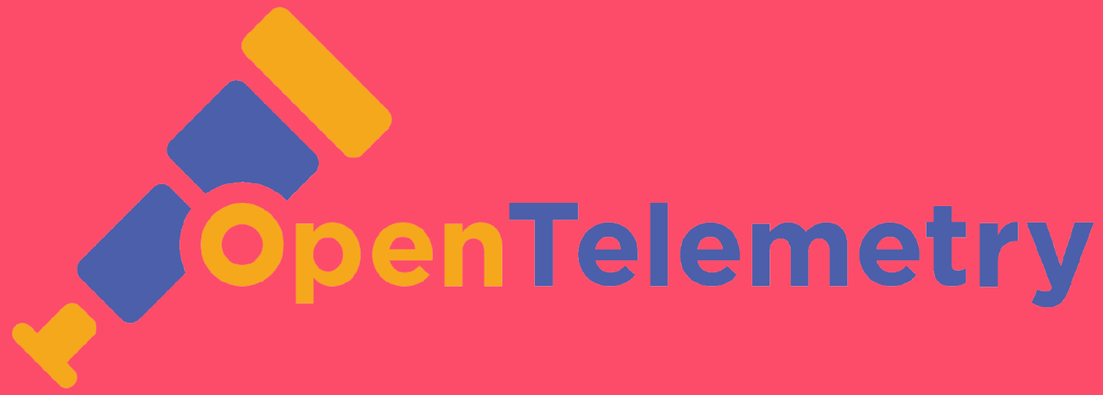
Standardization

Switching vendor or knowledge baseline?

Trying to learn how to keep your system healthy by only studying its failures is like trying to learn how to keep your marriage healthy by only studying divorces.

— https://twitter.com/relix42/status/1199871657696849921

The State of



Philipp Krenn @xeraa