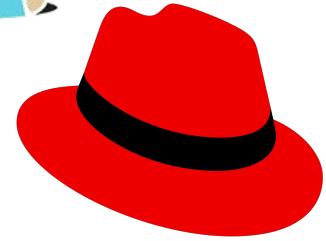

Jaeger: The Future with OpenTelemetry and Metrics

Maintainer Talk - Kubecon EU 2023

Pavol Loffay [[@ploffay](#)]
Jonah Kowall [[@jkowall](#)]



Pavol Loffay?



Jonah Kowall?



Agenda

1. Why Distributed Tracing? (Jonah)
2. Intro to Distributed Tracing and Jaeger (Jonah)
3. OpenTelemetry auto instrumentation (Pavol)
4. Pipelines with Jaeger and OpenTelemetry (Pavol)
5. Jaeger operator (Pavol)
6. New Monitoring tab and Prometheus support in Jaeger (Jonah)
7. New Key Features + Roadmap for Jaeger (Jonah)
8. Q&A from the audience (Jonah + Pavol)

Why Distributed Tracing?

Microservices Architectures

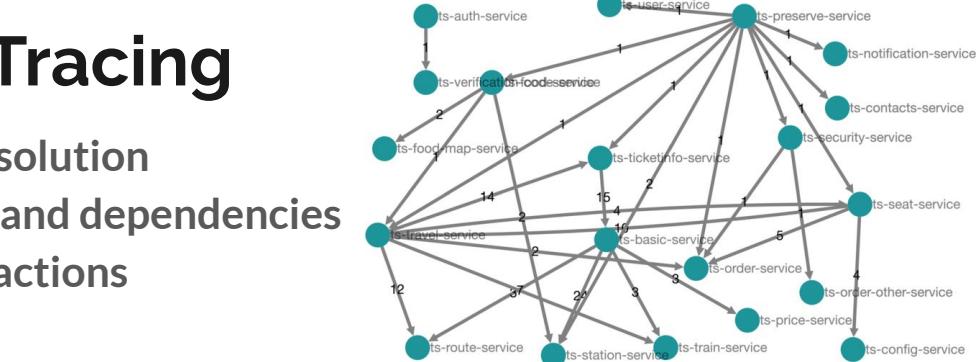
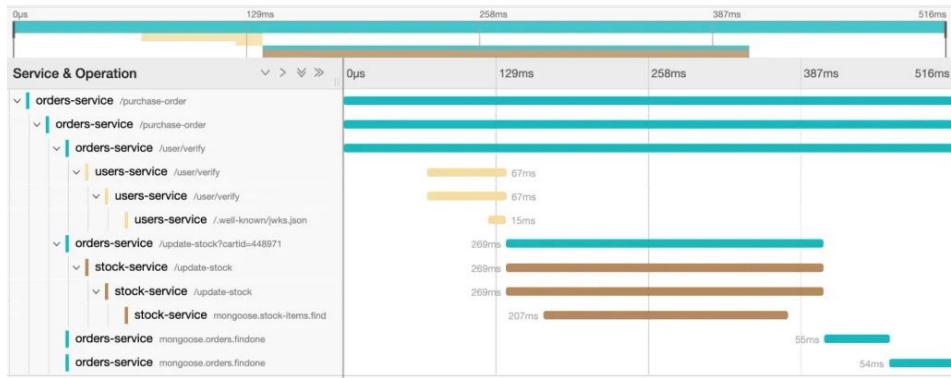
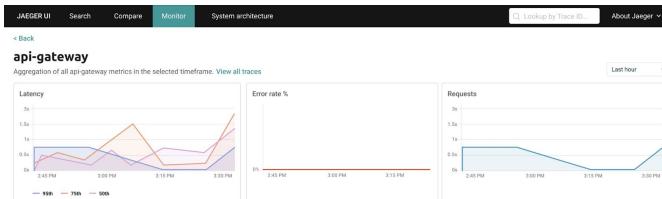
63% of enterprises are using Microservices most often managed by different teams.

When things break or are slow, how do you find the right team to help resolve the problem?



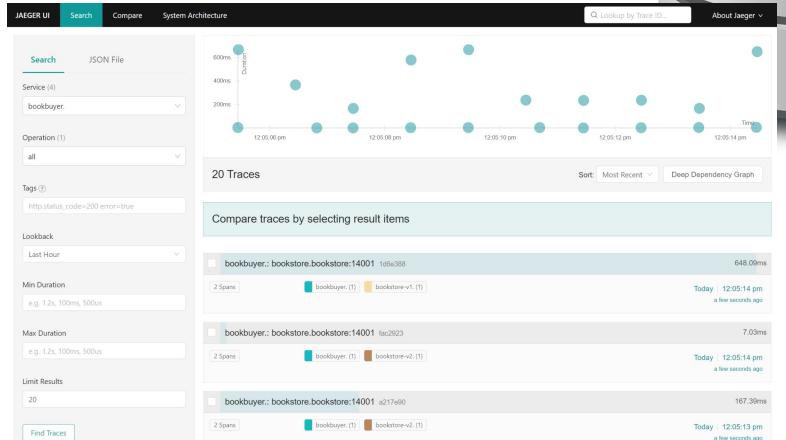
Benefits of Distributed Tracing

- Problem Detection and Problem Resolution
- Build a service map of relationships and dependencies
- Measure and drill into specific user actions
- Collaborate across teams
- Monitor and maintain SLAs



Tracing Tools

- Instrumentation
- Data Collection
- Analysis and visualization



Intro to Distributed Tracing and Jaeger



OpenTelemetry Semantics

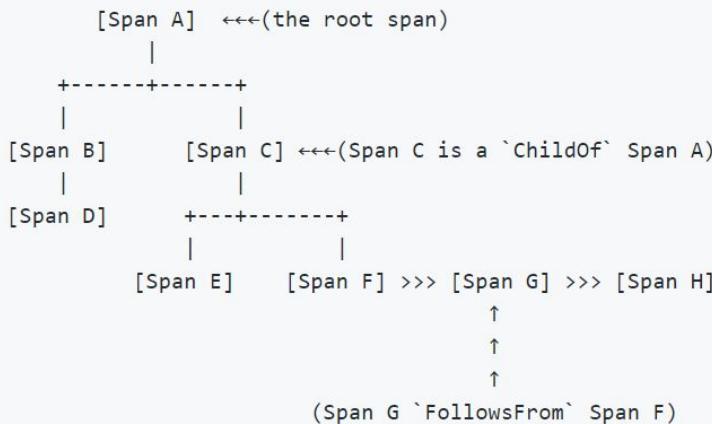
Trace represents an end-to-end request (and response); made up of single or multiple Spans

Span represents work done by a single-service or component with time intervals and associated metadata such as **Tags**

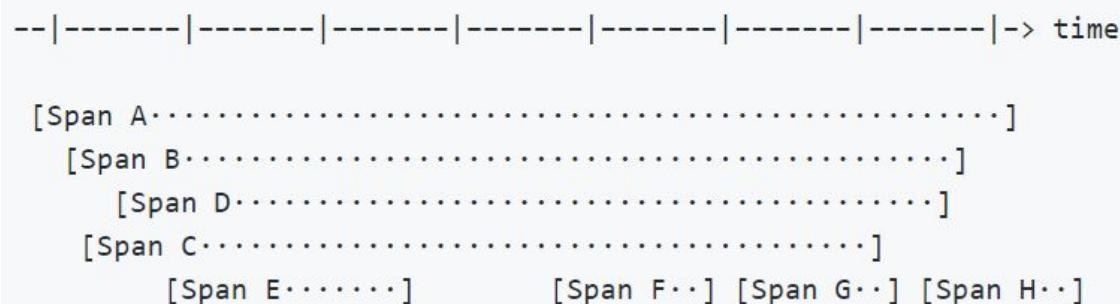
Tags contain metadata to help contextualize a span

Relationships in tracing

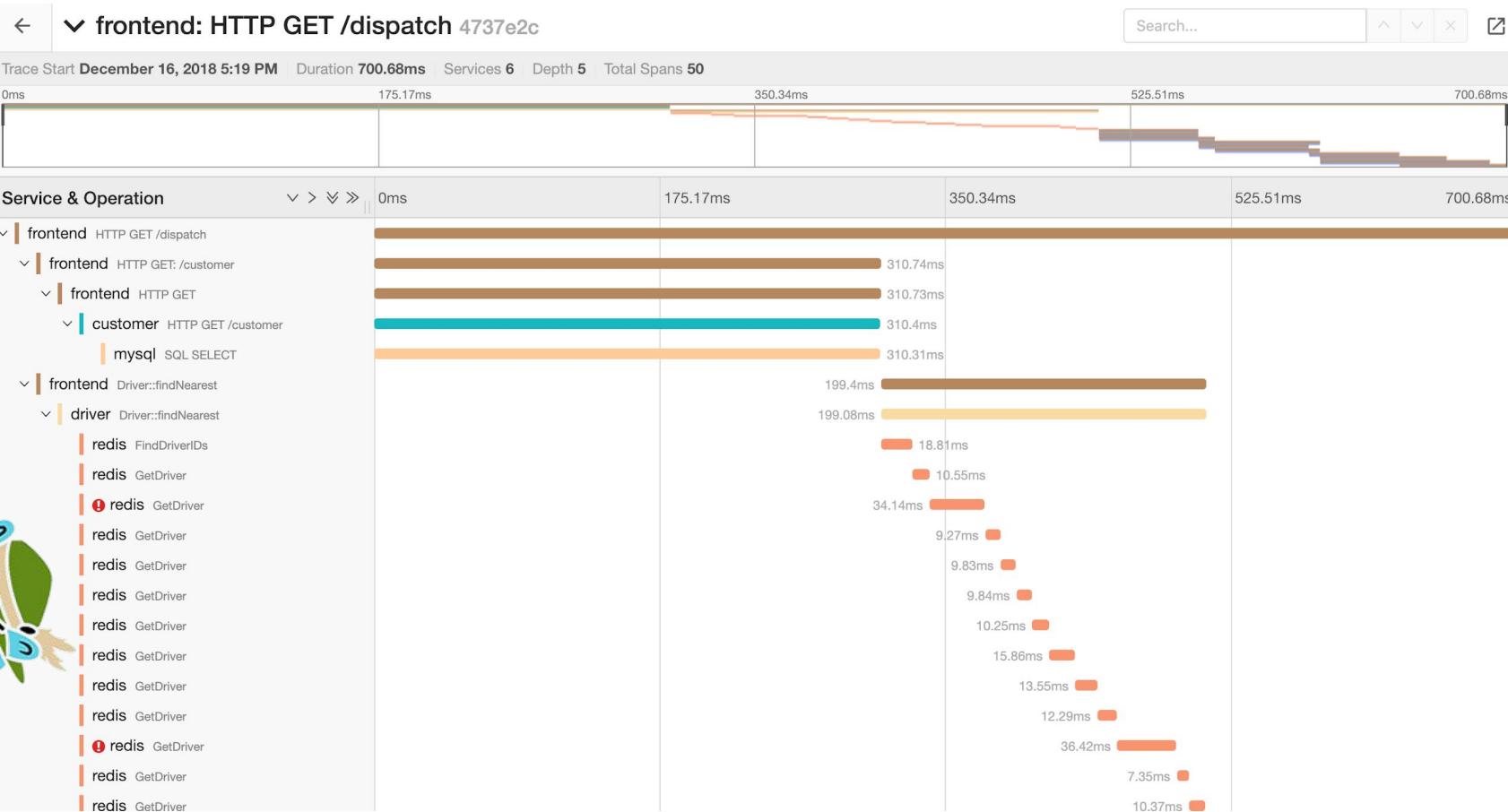
Causal relationships between Spans in a single Trace



Temporal relationships between Spans in a single Trace



Measure errors, latency, and other indicators across each span



OpenTelemetry (auto) instrumentation

Deprecated Clients

Sunsetting Jaeger clients 🏁 #3362

Open

7 of 9 tasks

yurishkuro opened this issue on Oct 31, 2021 · 17 comments



yurishkuro commented on Oct 31, 2021 • edited

The Jaeger clients have faithfully served our community for several years, providing reliable sampling and per-operation / adaptive sampling which

Version 1.37

Latest



This library is DEPRECATED!

No new pull requests are accepted except for security fixes.

We urge all users to migrate to [OpenTelemetry](#). Please refer to the [notice in the documentation](#) for details.

Jaeger Bindings for Go OpenTracing API

See also:

- [Client Features](#)



Jaeger clients are being retired.

Wide Language Support (OpenTracing Shim)



<https://opentelemetry.io/docs/migration/opentracing/>

Environment Variable Based Config

JAEGER_SERVICE_NAME
JAEGER_TAGS

JAEGER_AGENT_HOST



OTEL_SERVICE_NAME
OTEL_RESOURCE_ATTRIBUTES

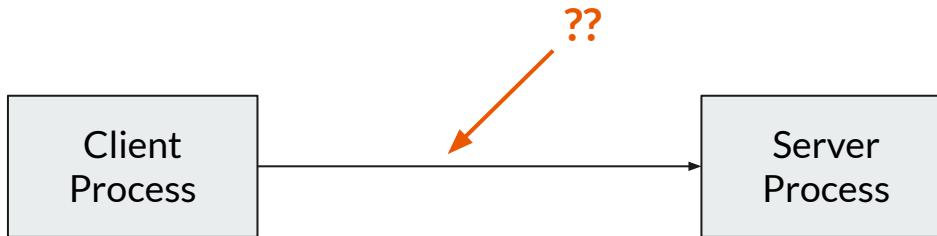
OTEL_TRACES_EXPORTER
OTEL_EXPORTER_JAEGER_PROTOCOL
OTEL_EXPORTER_JAEGER_AGENT_HOST

JAEGER_SAMPLER_TYPE
JAEGER_SAMPLER_PARAM
JAEGER_PROPAGATION

OTEL_TRACES_SAMPLER
OTEL_TRACES_SAMPLER_ARG
OTEL_PROPAGATORS

<https://opentelemetry.io/docs/reference/specification/sdk-environment-variables/>

Trace Propagation



OTEL_PROPAGATORS

default: tracecontext,baggage (w3c)

supported: jaeger, b3, b3multi

<https://opentelemetry.io/docs/reference/specification/sdk-environment-variables/>

(Auto) Instrumentation

Auto!

- Java
- Python
- .NET
- Node.JS

Less Auto!

- Go
eBPF auto-inst coming soon
- Ruby
minor global code change
- C++, PHP, Rust...

<https://opentelemetry.io/docs/instrumentation/>

Java/Spring

The screenshot displays a trace visualization for a Java/Spring application. On the left, a timeline shows the execution flow from 2022-09-16 13:11:23.594 to 13:11:23.594, with a total duration of 57.96ms. The timeline is color-coded by service, with most time spent in the application layer (orange) and some in the database layer (red). Below the timeline, a tree view shows the sequence of operations:

- Service & Operation: demo /TianMiao/api/users (57.96ms)
 - demo UserController.createUser (55.06ms)
 - demo UserRepository.save (37.81ms)
 - Session.persist com.example.TianMiao.model.User
 - demo INSERT tianmiao.user (1.13ms)
 - demo Transaction.commit (8.12ms)

On the right, two detailed attribute tables are shown for the database operation:

Attributes	
http.flavor	"1.1"
http.host	"localhost:8080"
http.method	"POST"
http.request_content_length	19
http.route	"/TianMiao/api/users"
http.scheme	"http"
http.status_code	200
http.target	"/TianMiao/api/users"
http.user_agent	"curl/7.58.0"
net.sock.peer.addr	"172.21.0.1"
net.sock.peer.port	39036
net.transport	"ip_tcp"
otel.library.name	"io.opentelemetry.tomcat-7.0.0-alpha"
otel.library.version	"1.17.0-alpha"
span.kind	"server"

Attributes	
db.connection_string	"mysql://mysql:3306"
db.name	"tianmiao"
db.operation	"INSERT"
db.sql.table	"user"
db.system	"mysql"
db.user	"root"
net.peer.name	"mysql"
net.peer.port	3306
otel.library.name	"io.opentelemetry.jdbc"
otel.library.version	"1.17.0-alpha"
span.kind	"client"
status.code	0
thread.id	37
thread.name	"http-nio-8080-exec-2"

<https://github.com/joe-elliott/tempo-springboot-example>

Pipelines with Jaeger and OpenTelemetry

OTel Collector + OTel Clients + Jaeger

- OTel Supports most Jaeger protocols (client and collector*)
 - Agent
 - thrift compact (6831)
 - thrift binary (6832)
 - Collector
 - protobuf/gRPC (14250)
 - thrift over HTTP (14268)
 - Jaeger remote sampler extension (HTTP 5778, 14250 gRPC)
- Can also write to Kafka in supported Jaeger formats

*<https://opentelemetry.io/docs/collector/distributions>

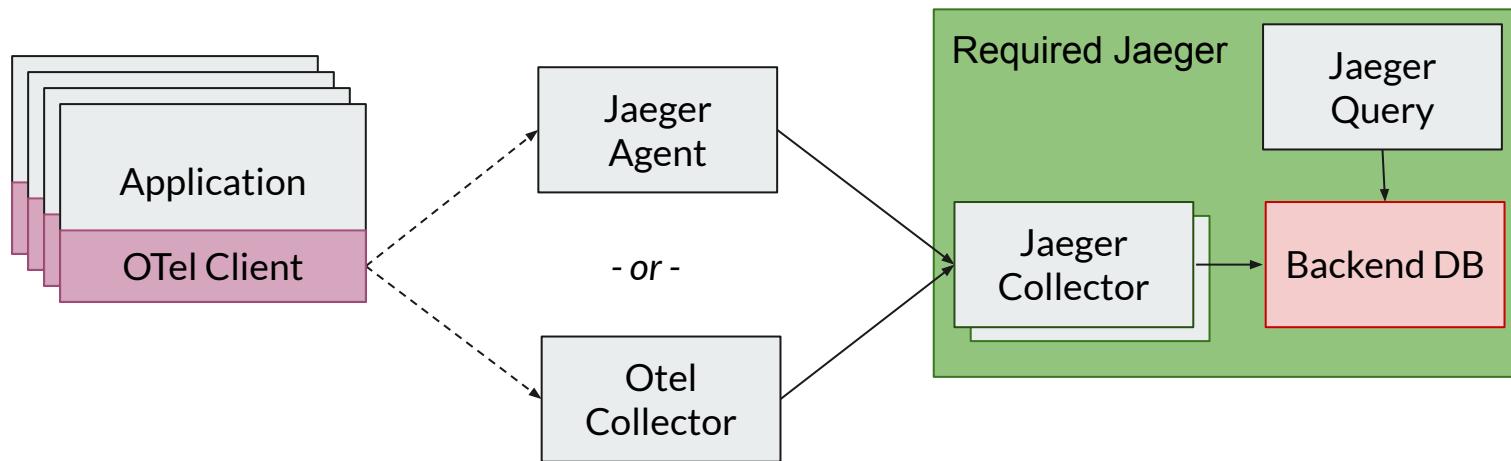
OTel Processors

```
processors:  
  redaction:  
    # allow_all_keys is a flag which when set to true, which can disable  
    # allowed_keys list. The list of blocked_values is applied regardless  
    # you just want to block values, set this to true.  
    allow_all_keys: false  
    # allowed_keys is a list of span attribute keys that are allowed to  
    # through. The list is designed to fail closed. If allowed_keys is  
    # no span attributes are allowed and all span attributes are removed  
    # allow all keys, set allow_all_keys to true. To allow the span attributes  
    # you know are good, add them to the list.  
    allowed_keys:  
      - description  
      - group  
      - id  
      - name  
    # blocked_values is a list of regular expressions for blocking values  
    # allowed span attributes. Values that match are masked  
    blocked_values:  
      - "4[0-9]{12}(?:[0-9]{3})?" ## Visa credit card number  
      - "(5[1-5][0-9]{14})"      ## MasterCard number
```

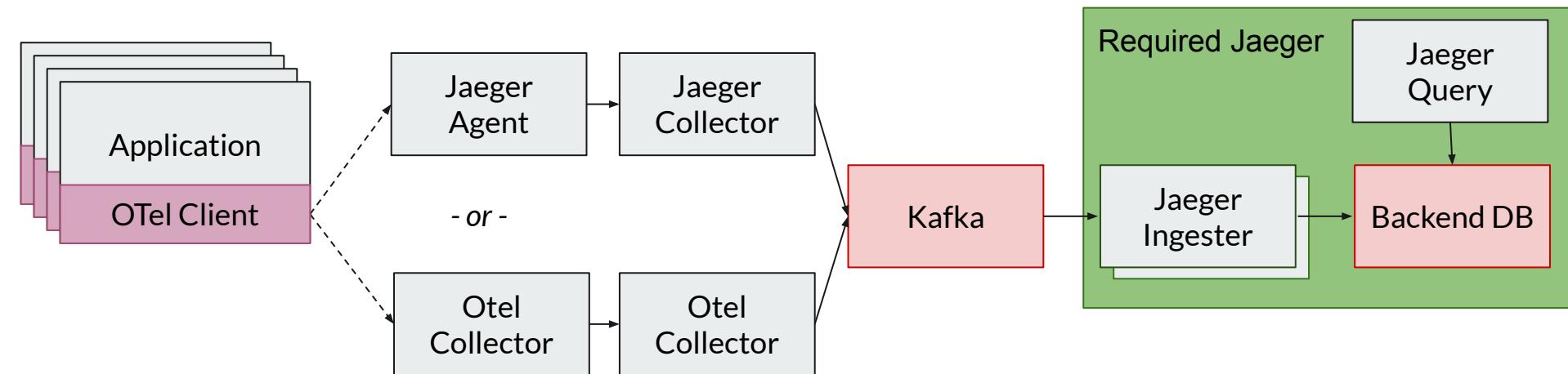
attributesprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
cumulativetodeltaprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
deltatorateprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
filterprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
groupbyattrprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
groupbytraceprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
k8sattributesprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
logtransformprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
metricsgenerationprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
metricstransformprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
probabilisticsamplerprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
redactionprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
resourcedetectionprocessor	[chore] dependabot updates Mon Oct 24 22:43:13 UTC 2022 (#15650)
resourceprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
routingprocessor	[chore] dependabot updates Mon Oct 24 22:43:13 UTC 2022 (#15650)
schemaprocessor	[chore] dependabot updates Mon Oct 24 22:43:13 UTC 2022 (#15650)
servicegraphprocessor	[chore] dependabot updates Mon Oct 24 22:43:13 UTC 2022 (#15650)
spanmetricsprocessor	[chore] dependabot updates Mon Oct 24 22:43:13 UTC 2022 (#15650)
spanprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
tailsamplingprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)
transformprocessor	[chore] dependabot updates Mon Oct 24 18:28:10 UTC 2022 (#15536)

<https://github.com/open-telemetry/opentelemetry-collector-contrib/tree/main/processor>

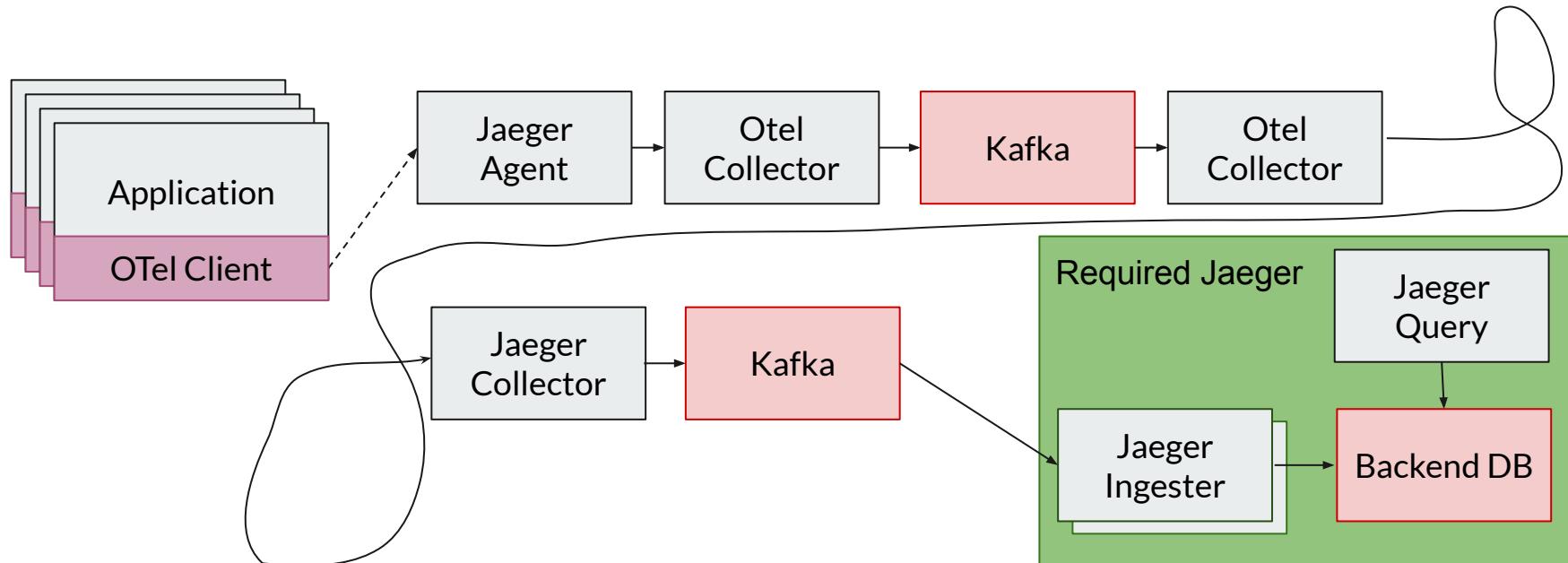
Without Queueing



With Queueing



Ad infinitum



Jaeger operator

“operates” Jaeger on Kubernetes

Getting Started

- jaegertracing.io/docs/latest/operator
- Storage schema creation
- Jaeger upgrades
- Auto recognizes available APIs - OpenShift/Kubernetes, ES/Stimzi
- Can generate plain Kubernetes manifest files



Jaeger CRD

- CRD - Custom Resource Definition

```
apiVersion: jaegertracing.io/v1
kind: Jaeger
metadata:
  name: jaeger-cluster
spec:
  strategy: allInOne | production | streaming
  storage:
    type: memory|elasticsearch|cassandra|badger|grpc-plugin|kafka
```

- kubectl get jaegers

Jager agent sidecar injection

- Use OpenTelemetry operator!
 - it exposes the same ports



OpenTelemetry Kubernetes tutorial

- Exploring the power of OpenTelemetry on Kubernetes
- Friday 4 pm - 5.30 pm
- [pavolloffay/kubecon-eu-2023-opentelemetry-kubernetes-tutorial](https://github.com/pavolloffay/kubecon-eu-2023-opentelemetry-kubernetes-tutorial)



New Monitoring tab and Prometheus support in Jaeger

Tracing and Monitoring

- Moving Jaeger from “distributed tracing” towards APM
 - Traces / Events
 - Metrics
- Additional use cases
 - Operational Monitoring
 - Operational Alerting
 - Change Planning



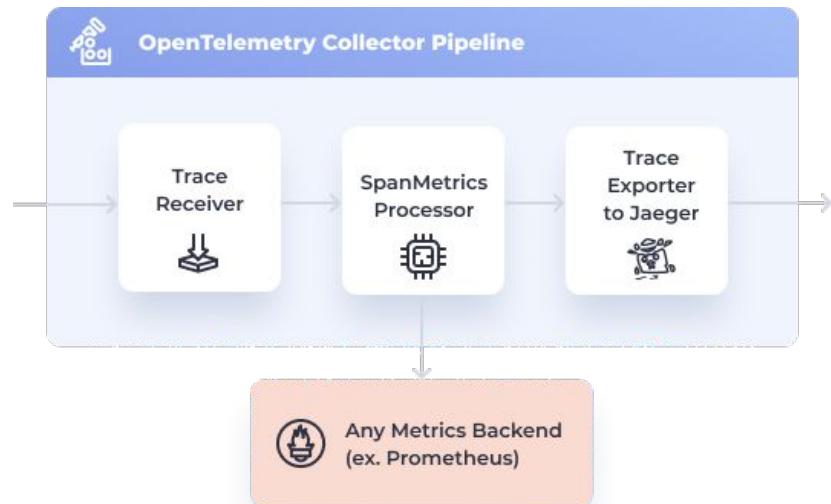
SpanMetrics Processor Flow

Prometheus handles all metrics use cases (monitor, alert, plan)
So can many other metrics platforms...



OpenTelemetry to derive
aggregated metrics from traces

SpanMetrics Processor



SpanMetrics Processor in OpenTelemetry

```
processors:  
  batch:  
    spanmetrics:  
      metrics_exporter: otlp/spanmetrics  
      latency_histogram_buckets: [100us, 1ms, 2ms, 6ms, 10ms, 100ms, 250ms]  
      dimensions:  
        - name: http.method  
          default: GET  
        - name: http.status_code  
  
service:  
  pipelines:  
    traces:  
      receivers: [jaeger]  
      processors: [spanmetrics, batch]  
      exporters: [jaeger]  
  
    # The exporter name must match the metrics_exporter name.  
    # The receiver is just a dummy and never used; added to pass validation requiring at least one receiver in a pipeline.  
    metrics/spanmetrics:  
      receivers: [otlp/spanmetrics]  
      exporters: [otlp/spanmetrics]  
  
    metrics:  
      receivers: [otlp]  
      exporters: [prometheus]
```

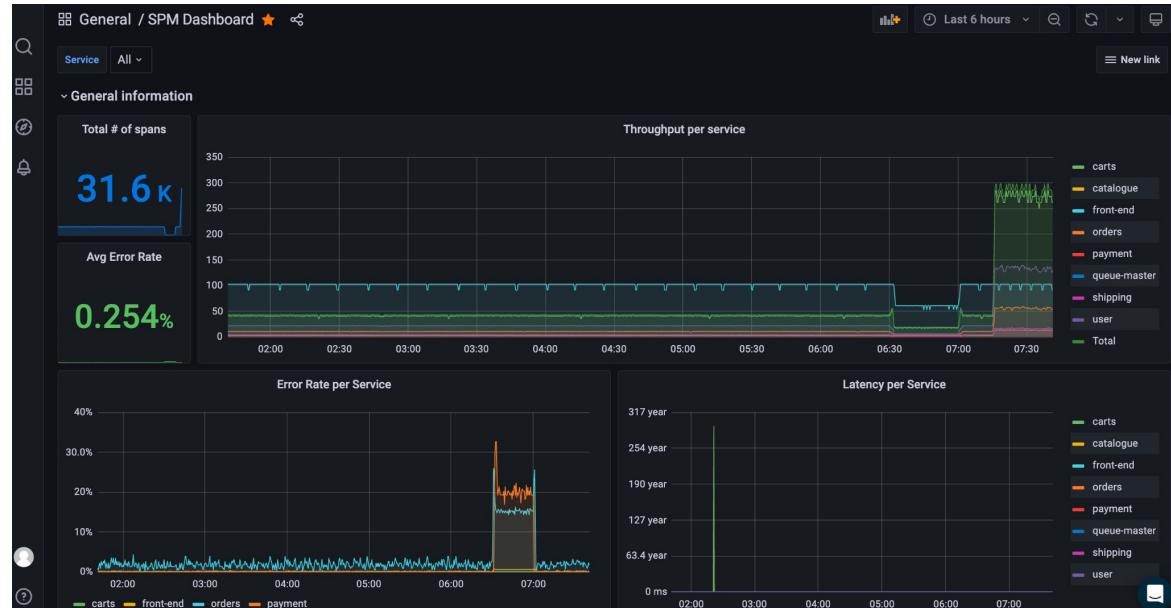
Define how many metrics (in this case method type and status code) and the buckets.

Result: Generate a metric per bucket per status code

Using Prometheus Metrics in Grafana

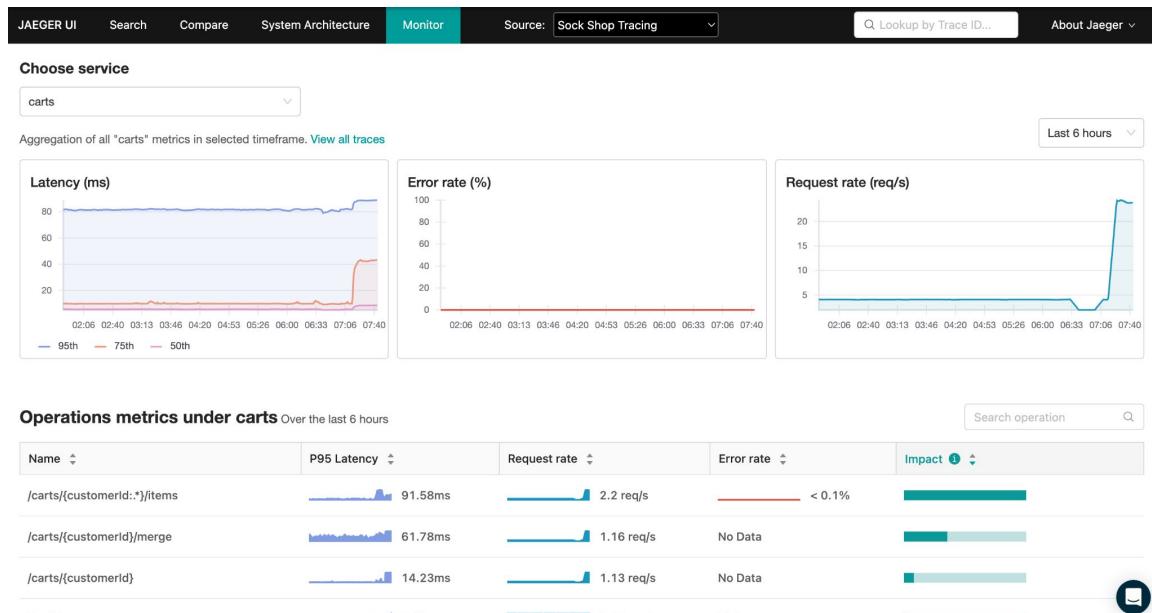
MetricQuery service in Jaeger to query metric backends.

- Any promql compatible backend (ex: Prometheus, Cortex, Thanos, M3DB, and Mimir)
- Community can add other metric systems as needed



Using Prometheus Metrics inside Jaeger UI

- New “monitor” homepage in Jaeger to provide status and health of transactions



New Key Features Since Kubecon Europe + Roadmap



Other New Features

- Native OTLP support in Jaeger collector
- Adaptive Sampling - Jaeger backend can be configured to perform fully automated and dynamic control of sampling rates based on predefined targets
- Flame graph views added versus table view of trace
- **NOTE:** Jaeger native SDKs are no longer supported, OpenTelemetry SDKs and Agents are the future path
 - Hotrod demo application and tracegen are now using OpenTelemetry



Roadmap

- Support for ClickHouse as a native datastore for Jaeger
- Updates to dependency graphs
 - Normalize the three types of graphs in Jaeger
 - Overlay service performance metrics on graph
 - Potentially move calculations from Spark/Kafka streams to OpenTelemetry collector service graph processor
- Move towards OpenTelemetry collector
 - Remove the need for Jaeger collector and normalize on a distribution of the collector for writes to Jaeger data stores

And more interesting capabilities coming in the future

**Q&A from audience in room and
online**

Resources



jaegertracing.io/docs



[monthly community call and Notes](#)

[CNCF Slack #jaeger : https://slack.cncf.io](https://slack.cncf.io)



[@jaegertracing](https://twitter.com/jaegertracing)



medium.com/jaegertracing

