Jaeger: The future with OpenTelemetry and Metrics

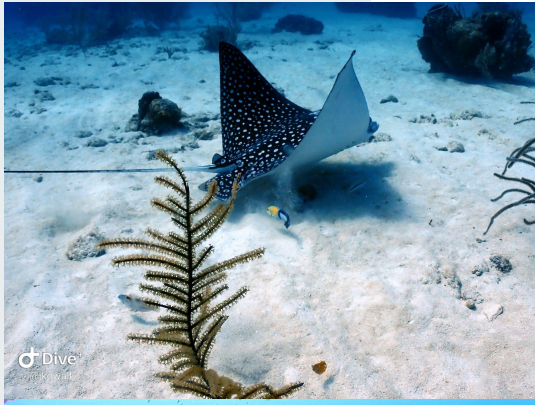
Maintainer Talk - Kubecon NA 2022

Joe Elliott [@actually_chores]
Jonah Kowall [@jkowall]

Joe Elliott?

actually_chores

Jonah Kowall?



logz.io

jkowall

Agenda

1. Intro to Distributed Tracing and Jaeger (Jonah)
2. OpenTelemetry auto instrumentation (Joe)
3. Pipelines with Jaeger and OpenTelemetry (Joe)
4. New Monitoring tab and Prometheus support in Jaeger (Jonah)
5. New Key Features + Roadmap for Jaeger (Jonah)
6. Q&A from audience in room and online (Jonah + Joe)

Intro to Distributed Tracing and Jaeger

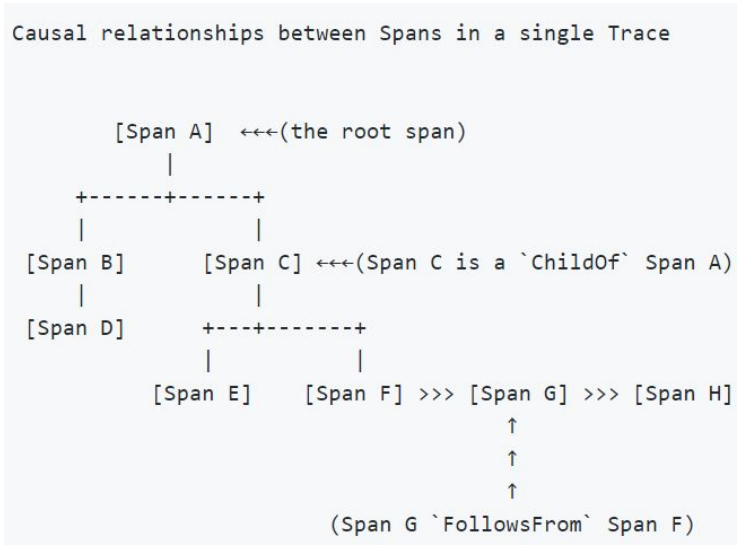
OpenTelemetry Semantics

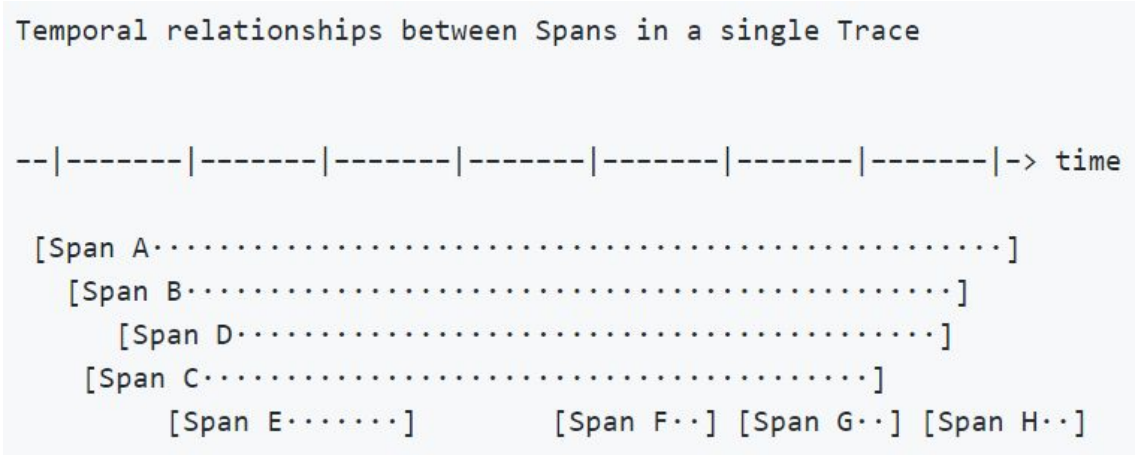
Trace represents an end-to-end request (and response); made up of single or multiple **Spans**

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

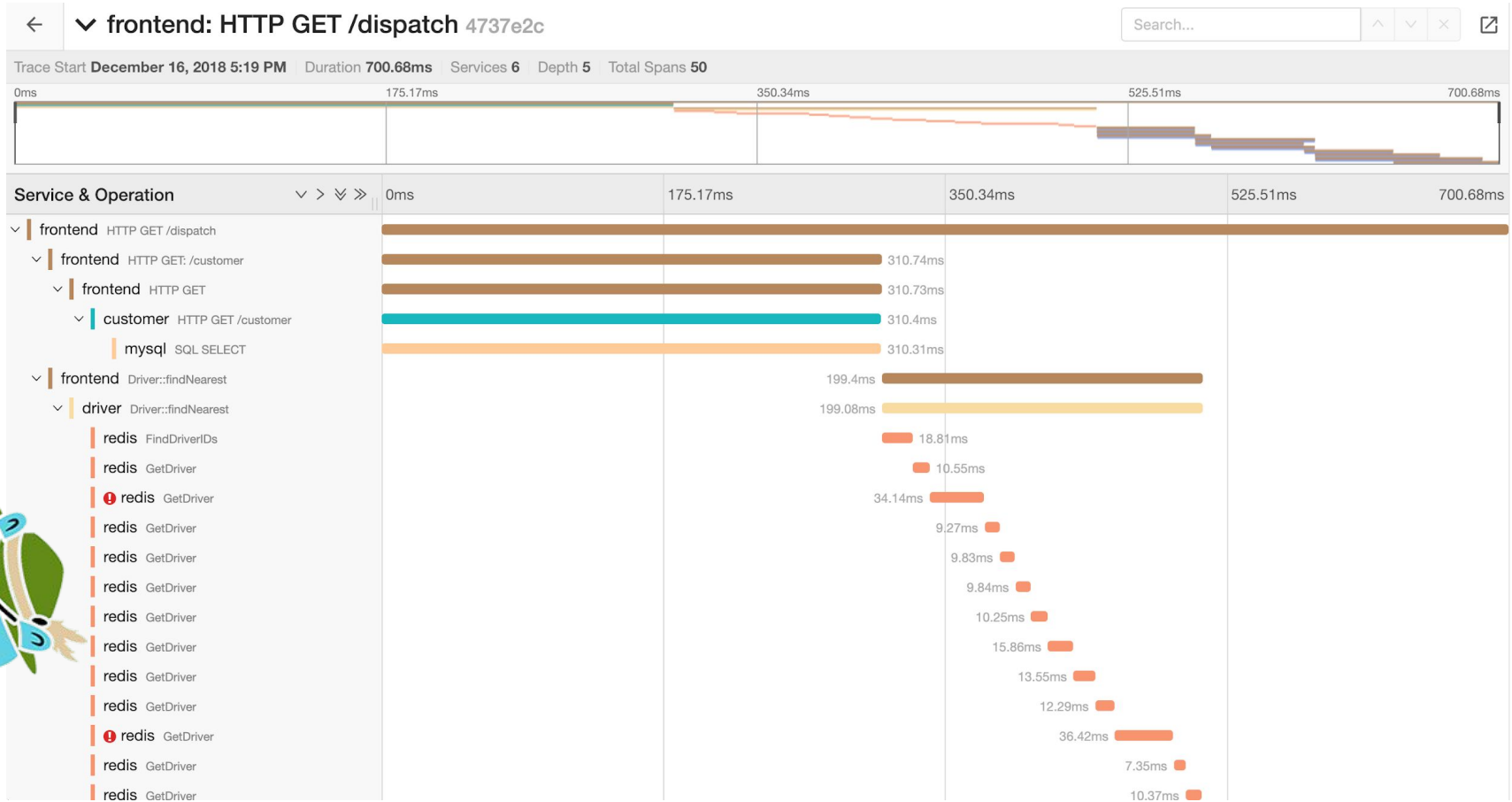
Tags contain metadata to help contextualize a span

Relationships in tracing



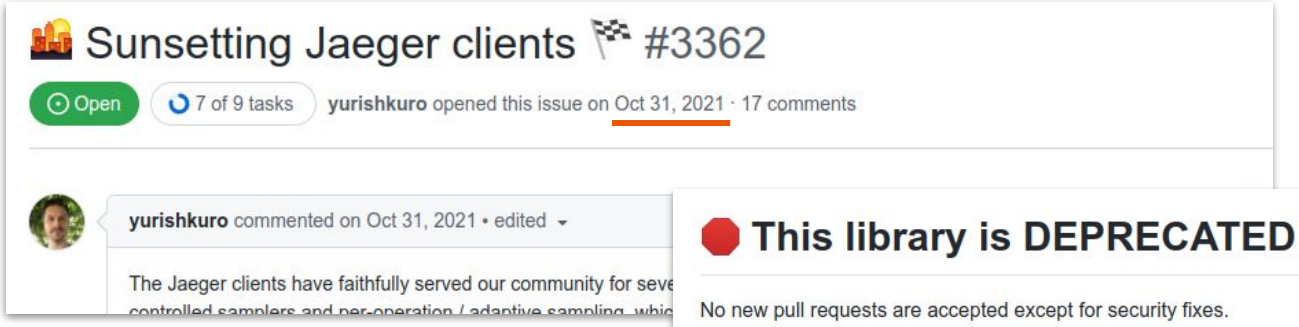


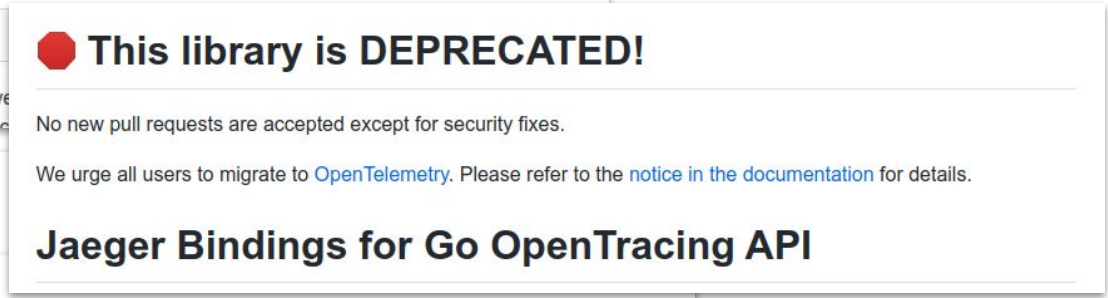
Measure errors, latency, and other indicators across each span

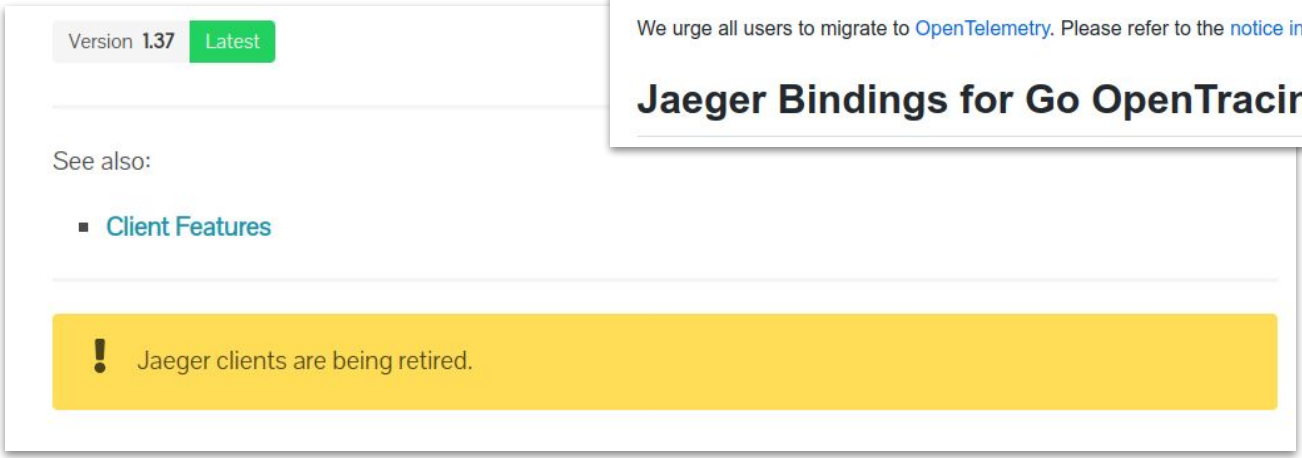

frontend: HTTP GET /dispatch 4737e2c
Search...
Trace Start December 16, 2018 5:19 PM
Duration 700.68ms
Services 6
Depth 5
Total Spans 50
0ms
175.17ms
350.34ms
525.51ms
700.68ms
Service & Operation
frontend HTTP GET /dispatch
frontend HTTP GET: /customer
frontend HTTP GET
customer HTTP GET /customer
mysql SQL SELECT
frontend Driver::findNearest
driver Driver::findNearest
redis FindDriverIDs
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
310.74ms
310.73ms
310.4ms
310.31ms
199.4ms
199.08ms
18.81ms
10.55ms
34.14ms
9.27ms
9.83ms
9.84ms
10.25ms
15.86ms
13.55ms
12.29ms
36.42ms
7.35ms
10.37ms

OpenTelemetry (auto) instrumentation

Deprecated Clients







Wide Language Support

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

Environment Variable Based Config

JAEGER_SERVICE_NAME		OTEL_SERVICE_NAME
JAEGER_TAGS		OTEL_RESOURCE_ATTRIBUTES
JAEGER_AGENT_HOST	→	OTEL_TRACES_EXPORTER OTEL_EXPORTER_JAEGER_PROTOCOL OTEL_EXPORTER_JAEGER_AGENT_HOST
JAEGER_SAMPLER_TYPE		OTEL_TRACES_SAMPLER
JAEGER_SAMPLER_PARAM		OTEL_TRACES_SAMPLER_ARG
JAEGER_PROPAGATION		OTEL_PROPAGATORS

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

(Auto) Instrumentation

Auto!

Java
specify -javaagent

.NET
env vars point to an assembly

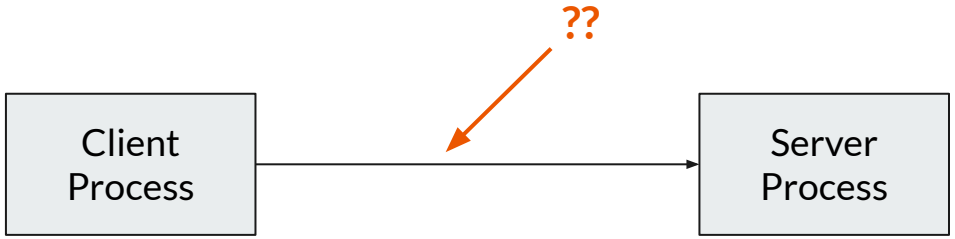
Less Auto!

Go
wrappers for client/server

Ruby
minor global code change

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

Trace Propagation



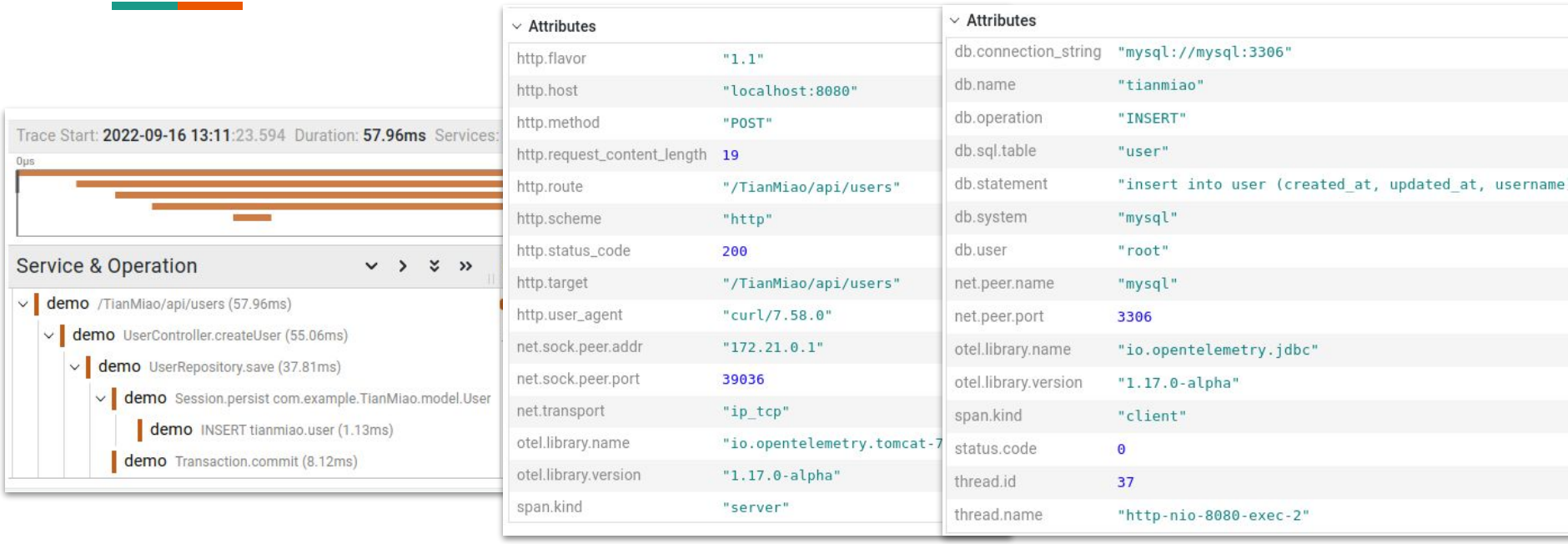
OTEL_PROPAGATORS

default: tracecontext,baggage (w3c)

supported: b3, b3multi, jaeger

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

Java/Spring



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

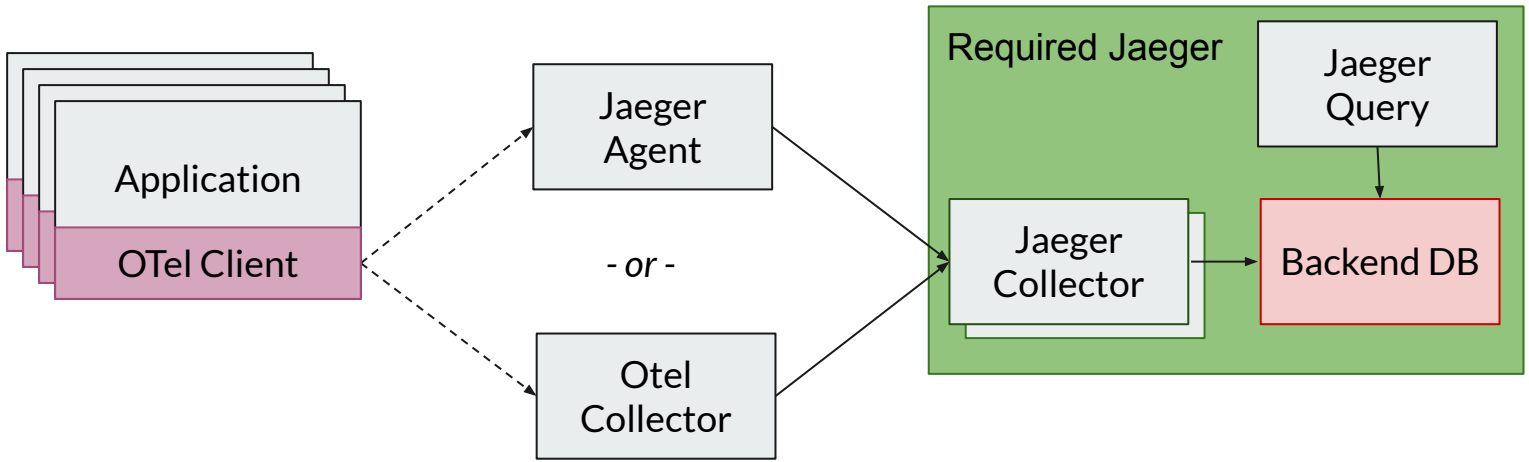
Pipelines with Jaeger and OpenTelemetry

OTel Collector + OTel Clients + Jaeger

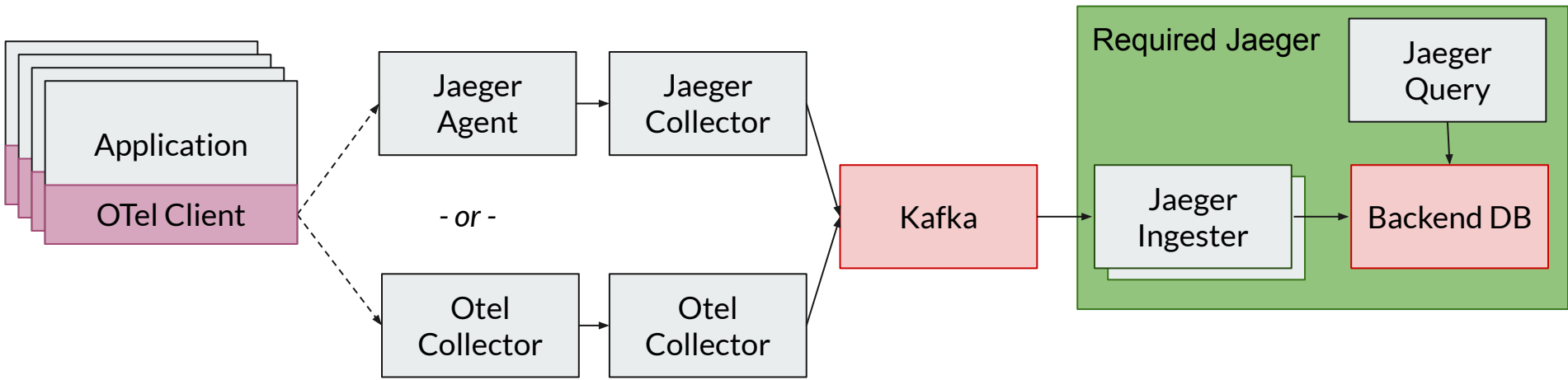
- OTel Supports most Jaeger protocols (client and collector*)
  - Agent
    - thrift compact (6831)
    - thrift binary (6832)
  - Collector
    - protobuf/gRPC (14250)
    - thrift over HTTP (14268)
- Can also write to Kafka in supported Jaeger formats

*https://hub.docker.com/r/otel/opentelemetry-collector-contrib

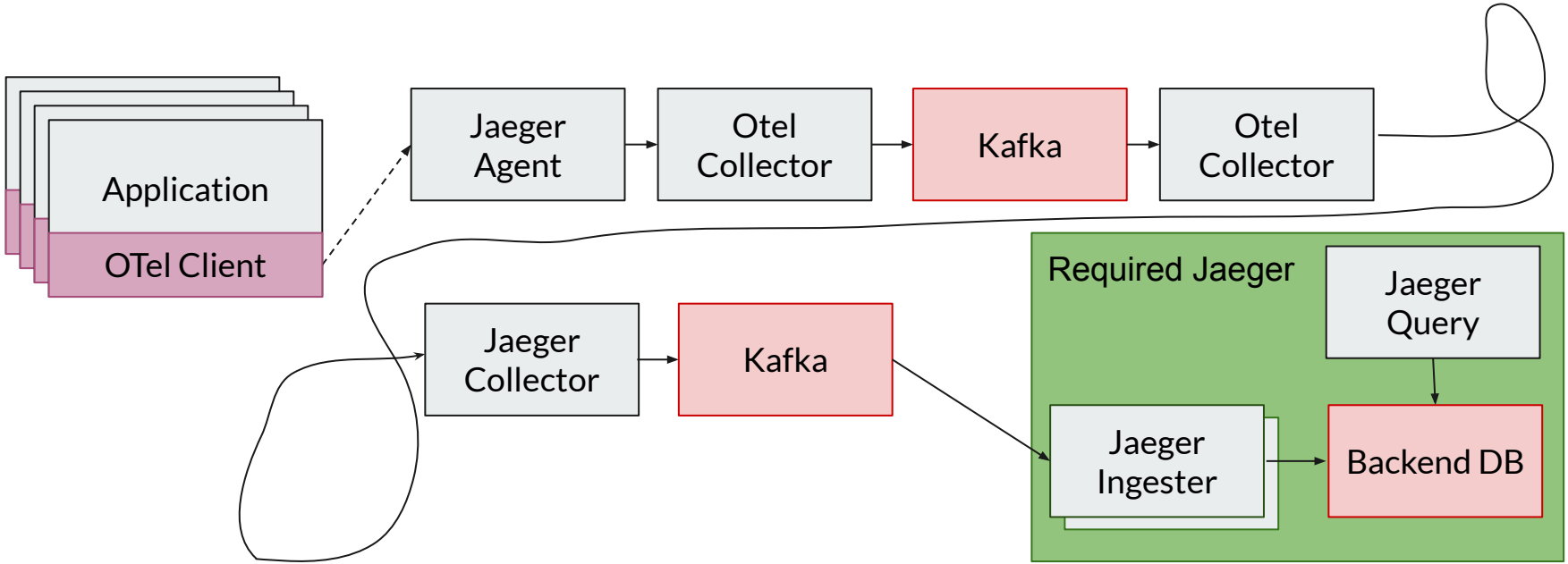
Without Queueing



With Queueing

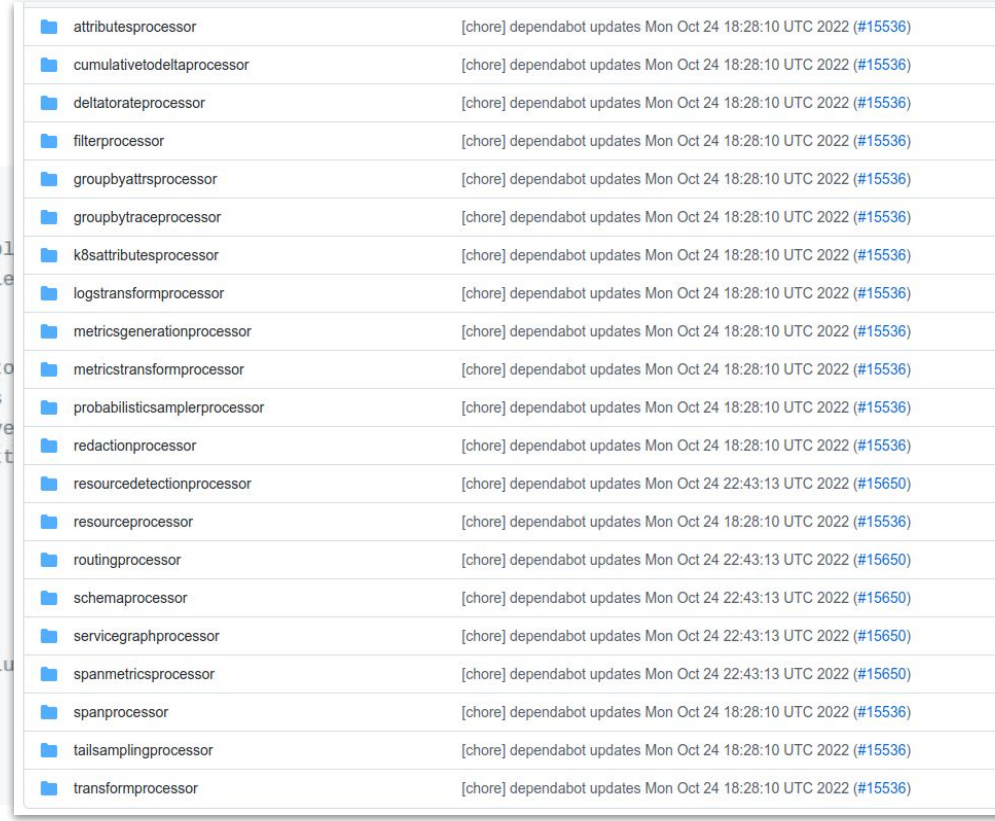


Ad infinitum



OTel Processors

```yaml
processors:
  redaction:
    # allow_all_keys is a flag which when set to true, which can disabl
    # allowed_keys list. The list of blocked_values is applied regardle
    # 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 remove
    # allow all keys, set allow_all_keys to true. To allow the span att
    # 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 valu
    # 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)
groupbyattrsprocessor	[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)
logstransformprocessor	[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

New Monitoring tab and Prometheus support in Jaeger

Tracing and Monitoring

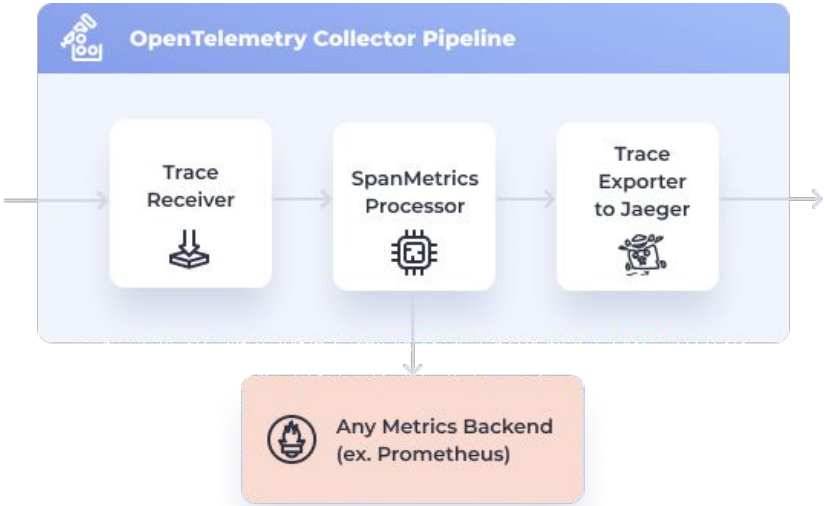
- Moving Jaeger from “distributed tracing” towards APM
  - Traces / Events
  - Metrics
- Additional use cases
  - Operational Monitoring
  - Operational Alerting
  - Change Planning

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

```yaml
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

opentelemetry-collector-contrib/processor/spanmetricsprocessor at main · open-telemetry/opentelemetry-collector-contrib (github.com)

Using Prometheus Metrics in Grafana

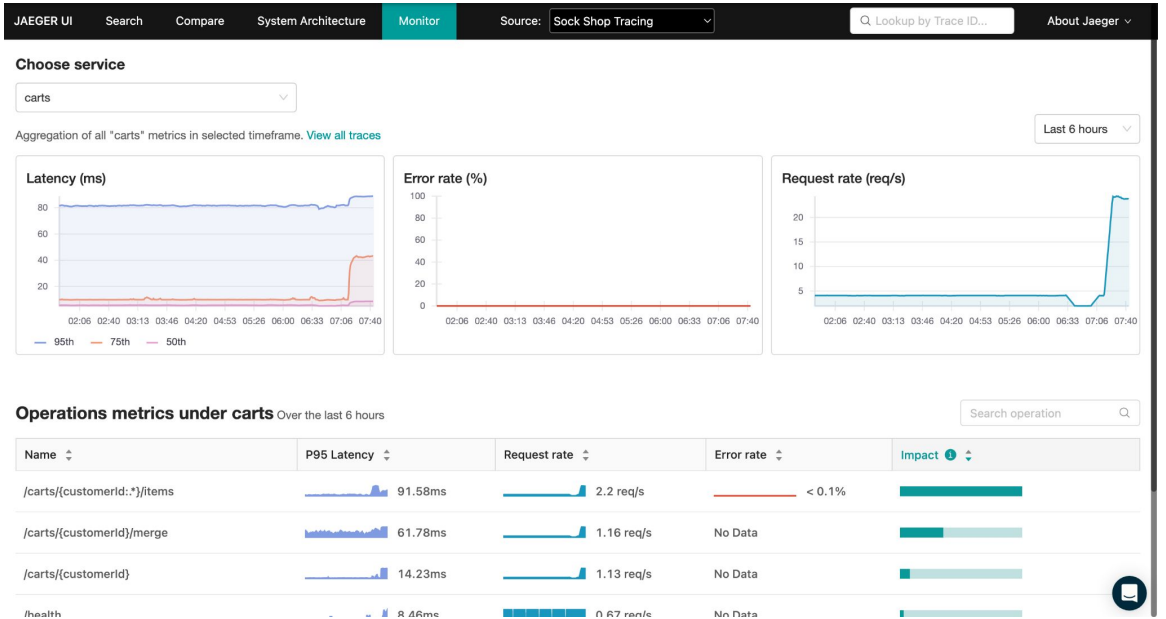
MetricQuery service in Jaeger to query metric backends.

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

DEMO : SPM in Grafana of both dashboards

Using Prometheus Metrics inside Jaeger UI

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



DEMO : Monitor tab in Jaeger

New Key Features + 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, OpenTelemtry SDKs and Agents are the future path

Roadmap

- 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

@jaegertracing

medium.com/jaegertracing

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Please scan the QR Code above
to leave feedback on this session