



Jaeger V2: The Maintainers' Guide To OpenTelemetry-Native Tracing

Maintainer Talk - Kubecon EU 2026

Pavol Loffay

Principal Software Engineer - Red Hat





Pavol Loffay?

 ploffay





Agenda

1. Why Distributed Tracing?
2. Intro to Jaeger + Demo
3. Monitoring with Prometheus + Jaeger
4. Jaeger V2
5. New Key Features + Roadmap for Jaeger
6. Q&A from the audience

Why Distributed Tracing?

Microservices Architectures

85% of enterprises are using Microservices

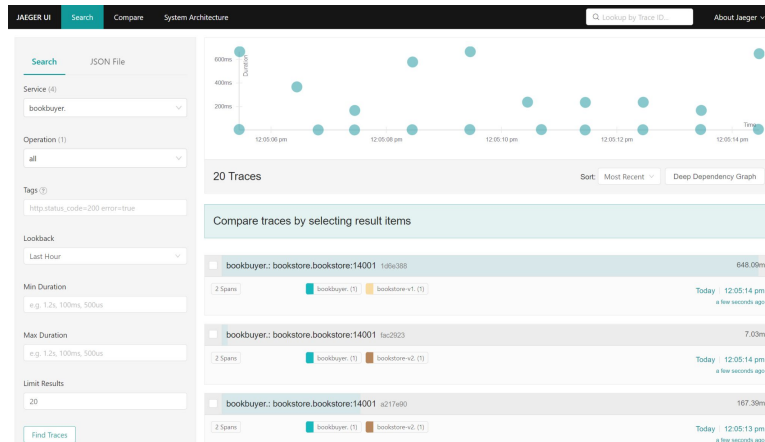
most often managed by different teams.

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



Tracing Tools

1. Instrumentation
2. Data Collection
3. Storage & Analysis & Visualization



Intro to Jaeger



OpenTelemetry Semantics

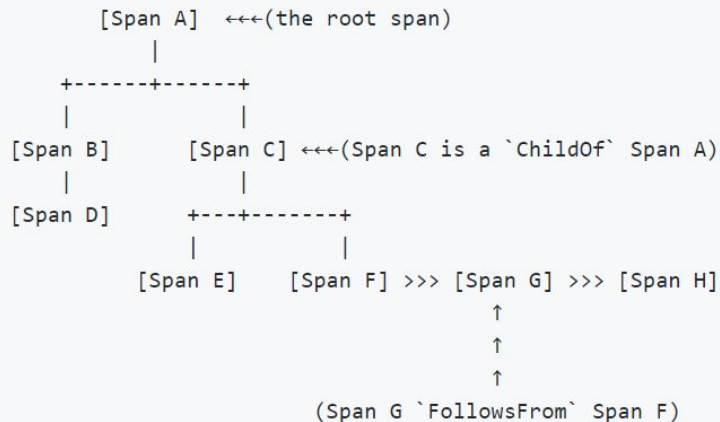
Trace represents an end-to-end request (and response); made up of single or multiple **Spans**

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

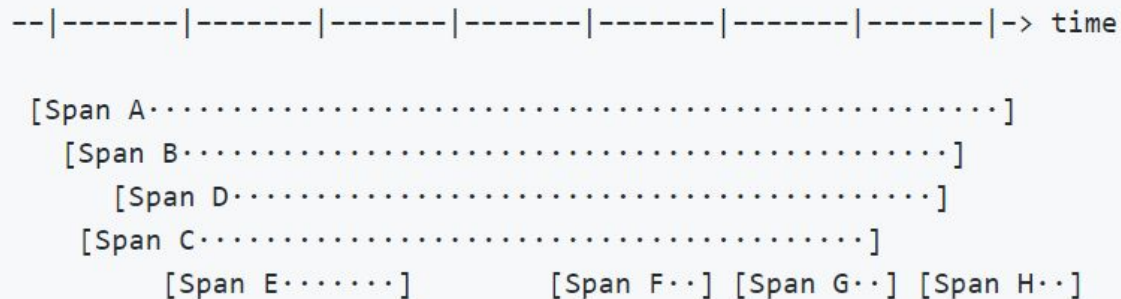
Tags/Attributes contain metadata to help contextualize a span

Relationships in tracing

Causal relationships between Spans in a single Trace



Temporal relationships between Spans in a single Trace



Metadata includes errors, latency, and other attributes of each span

Jaeger Demo



Search

Upload

Service (9)

frontend

Operation (2)

all

Tags

http.status_code=200 error=true

Lookback

Last Hour

Max Duration

e.g. 1.2s, 100ms, 500us

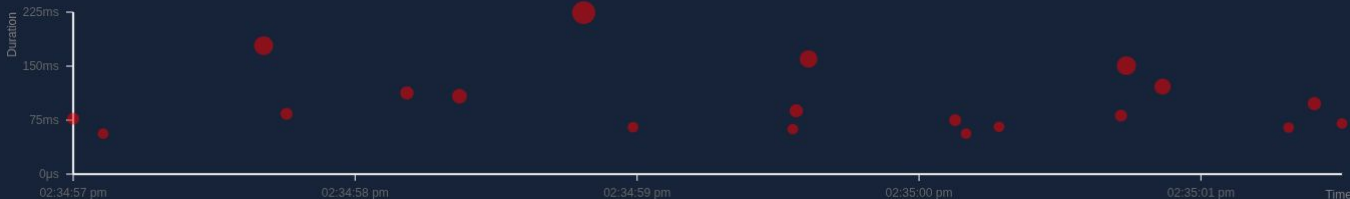
Min Duration

e.g. 1.2s, 100ms, 500us

Limit Results

20

Find Traces



20 Traces

Sort: Most Recent

Download Results

Deep Dependency Graph

Compare traces by selecting result items



test-executor: runTest b1c04e8

70.4ms

15 Spans

9 Errors

customer (2)

driver (3)

frontend (3)

mysql (1)

redis (2)

test-executor (2)

ui (2)

Today | 2:35:01 pm
a few seconds ago

test-executor: runTest 960530d

97.98ms

19 Spans

9 Errors

customer (2)

driver (5)

frontend (3)

mysql (1)

redis (4)

test-executor (2)

ui (2)

Today | 2:35:01 pm
a few seconds ago

test-executor: runTest c290c55

64.67ms

15 Spans

9 Errors

customer (2)

driver (3)

frontend (3)

mysql (1)

redis (2)

test-executor (2)

ui (2)

Today | 2:35:01 pm
a few seconds ago

test-executor: runTest 4abc869

121.56ms

23 Spans

9 Errors

customer (2)

driver (7)

frontend (3)

mysql (1)

redis (6)

test-executor (2)

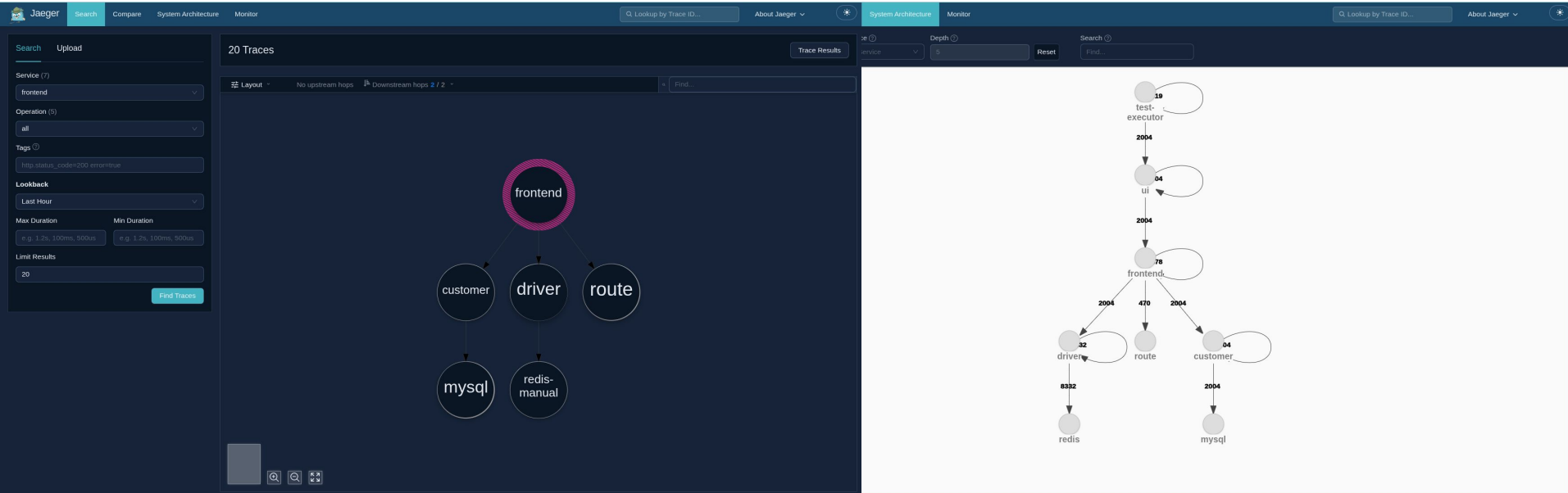
ui (2)

Today | 2:35:00 pm
a few seconds ago

test-executor: runTest 4c5571d

150.65ms

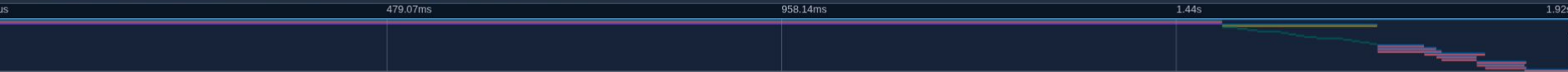
Relationships between services & components



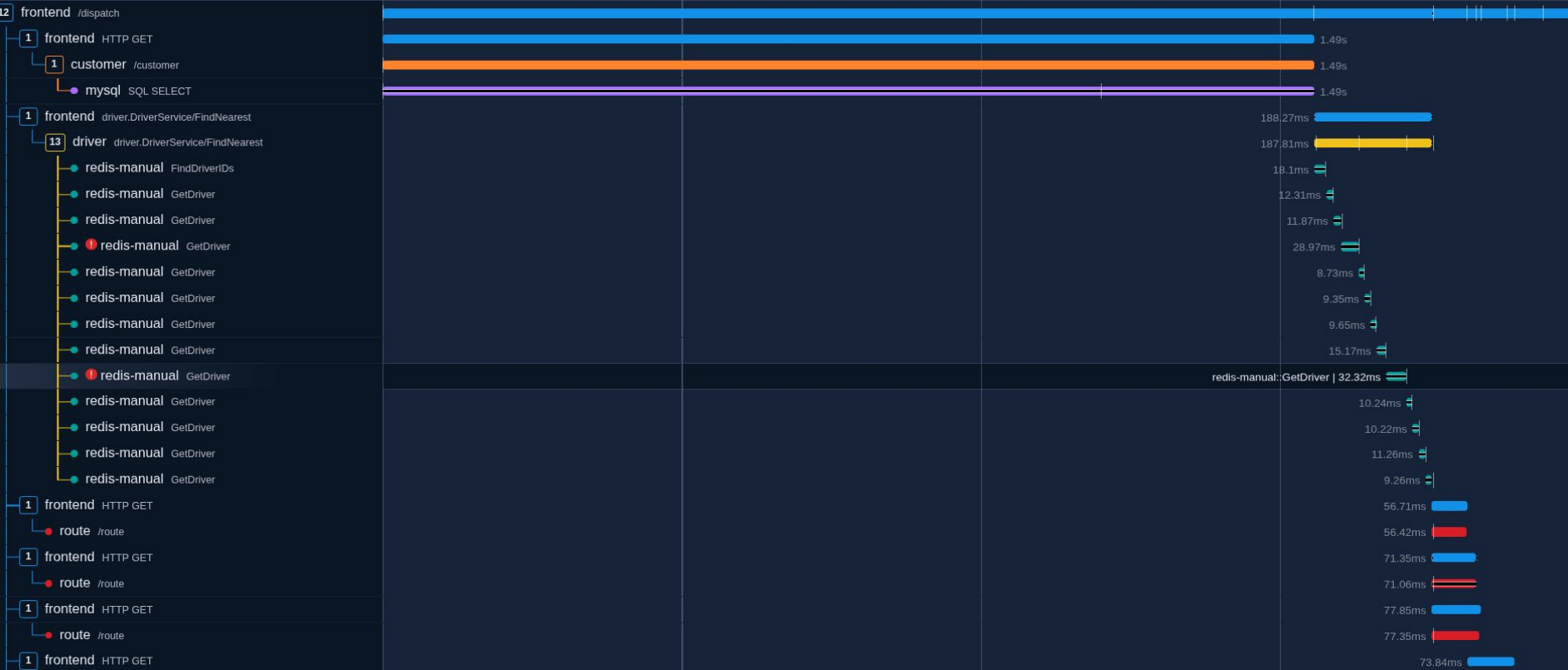
frontend: /dispatch 5b330c7

Find... ?

Trace Start March 16 2026, 14:44:55.585 Duration 1.92s Services 6 Depth 4 Total Spans 39



Service & Operation 0µs 479.07ms 958.14ms 1.44s 1.92s



Trace Start March 16 2026, 14:35:01.556 | Duration 70.4ms | Services 7 | Depth 9 | Total Spans 15

Search...

Head first

Reset View

Collapse nodes above

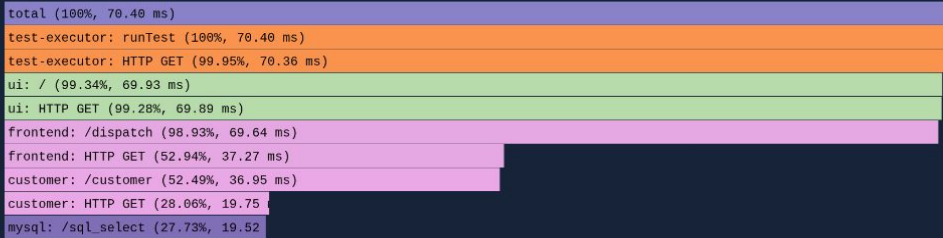
Table

Both

Flamegraph

Location	Self	Total
mysql: /sql_select	19.52 ms	19.52 ms
customer: /customer	17.20 ms	36.95 ms
test-executor: HTTP GET	0.43 ms	70.36 ms
frontend: HTTP GET	0.32 ms	37.27 ms
ui: HTTP GET	0.25 ms	69.89 ms
customer: HTTP GET	0.23 ms	19.75 ms
frontend: /dispatch	0.11 ms	69.64 ms
ui: /	0.04 ms	69.93 ms
test-executor: runTest	0.04 ms	70.40 ms
total	0.00 ms	70.40 ms

Frame width represents aggregated span duration



Trace Start **March 16 2026, 14:35:01.556** | Duration **70.4ms** | Services **7** | Depth **9** | Total Spans **15**

Trace Statistics

Group By: Service Name

Sub-Group: No item selected

☐ Color by: Count

Group	Count	Total	Avg	Min	Max	ST Total	ST Avg	ST Min	ST Max	ST in Duration
mysql	1	19.52ms	19.52ms	19.52ms	19.52ms	19.52ms	19.52ms	19.52ms	19.52ms	27.73%
test-executor	2	140.76ms	70.38ms	70.36ms	70.4ms	0.47ms	0.23ms	0.04ms	0.43ms	0.66%
ui	2	139.82ms	69.91ms	69.89ms	69.93ms	0.29ms	0.14ms	0.04ms	0.25ms	0.41%
customer	2	56.71ms	28.35ms	19.75ms	36.96ms	17.43ms	8.72ms	0.23ms	17.2ms	24.76%
redis	2	31.36ms	15.68ms	15.17ms	16.19ms	31.36ms	15.68ms	15.17ms	16.19ms	44.54%
frontend	3	139.18ms	46.39ms	32.26ms	69.64ms	0.74ms	0.25ms	0.11ms	0.32ms	1.05%
driver	3	63.83ms	21.28ms	15.36ms	31.95ms	0.59ms	0.2ms	0.08ms	0.33ms	0.84%

A test-executor: runTest b1c04e8

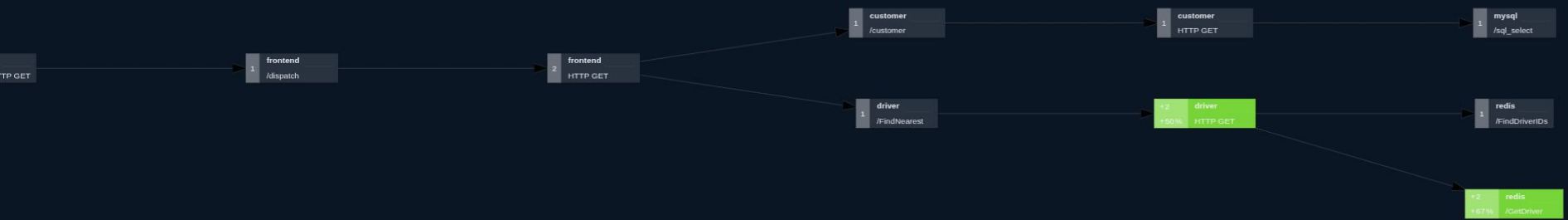
Today, 2:35:01 pm Duration: 70.4ms Spans: 15

vs

B

test-executor: runTest 960530d

Today, 2:35:01 pm Duration: 97.98ms Spans: 19



A

test-executor: runTest b1c04e8 [↗](#)

Today, 2:35:01 pm Duration: 70.4ms Spans: 15

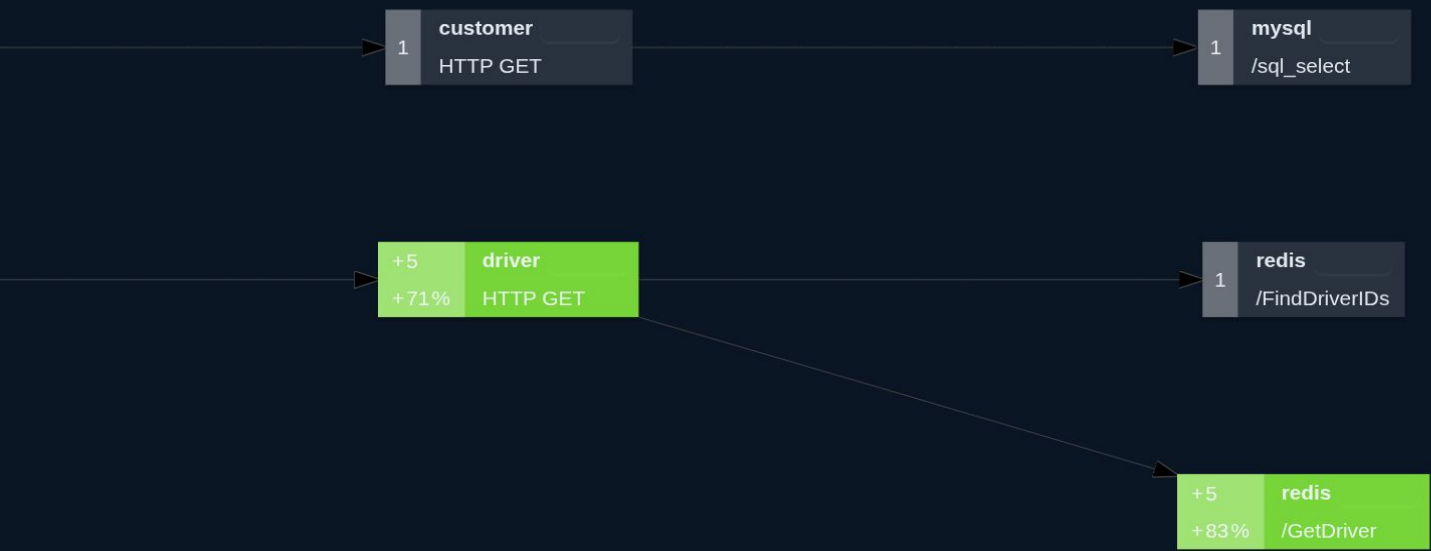


vs

B

test-executor: runTest 0ca903a [↗](#)

Today, 2:34:59 pm Duration: 159.92ms Spans: 25



Service

Span Kind

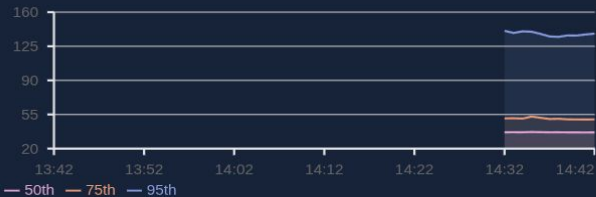
frontend

Client

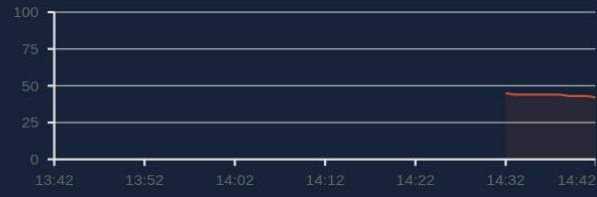
Aggregation of all "frontend" metrics in selected timeframe.
 [View all traces](#)

Last Hour

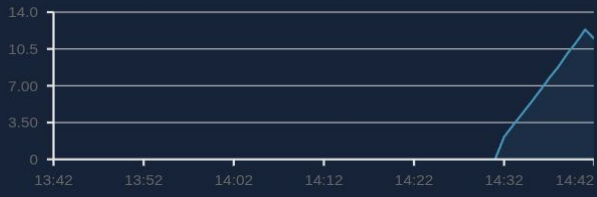
Latency (ms)



Error rate (%)



Request rate (req/s)



Operations metrics under frontend
 Over the last hour

Search operation

Name	P95 Latency	Request rate	Error rate	Impact ⓘ
HTTP GET	 137.68ms	 6.93 req/s	 44%	

Monitoring with Prometheus + Jaeger

Tracing and Monitoring



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



Jaeger Natively emits Prometheus Metrics

Remote write or scrape Prometheus metrics from Jaeger directly to your Prometheus compatible metrics backend.

Application level metrics to measure and alert on performance and usage:

- Rate
- Errors
- Duration

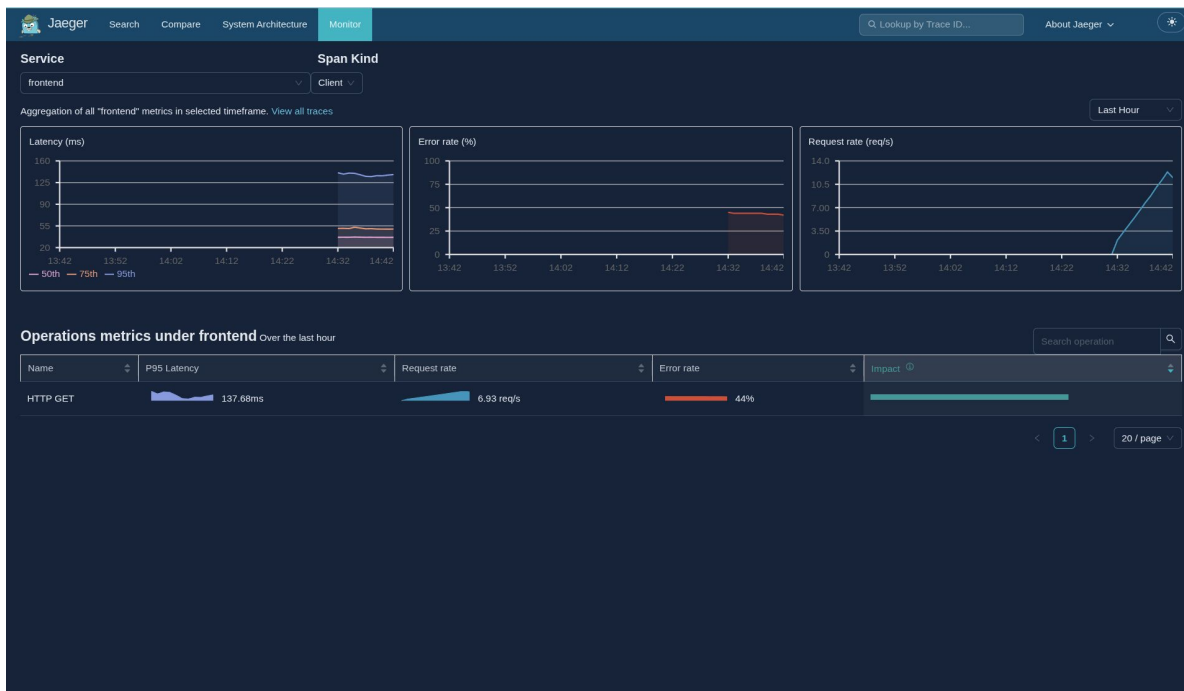
SpanMetrics Connector built into Jaeger v2

```
service:
  extensions: [jaeger_storage, jaeger_query]
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [jaeger_storage_exporter, spanmetrics]
    metrics/spanmetrics:
      receivers: [spanmetrics]
      exporters: [prometheus]
  extensions:
    jaeger_query:
      storage:
        traces: some_storage
        metrics: some_metrics_storage
    jaeger_storage:
      backends:
        some_storage:
          memory:
            max_traces: 100000
      metric_backends:
        some_metrics_storage:
          prometheus:
            endpoint: http://prometheus:9090
```

Visualize app performance R.E.D metrics

The “monitor” page in Jaeger to provides the status and health of transactions and application components

Deeper analysis can be done with Perces or Grafana.



Jaeger v2

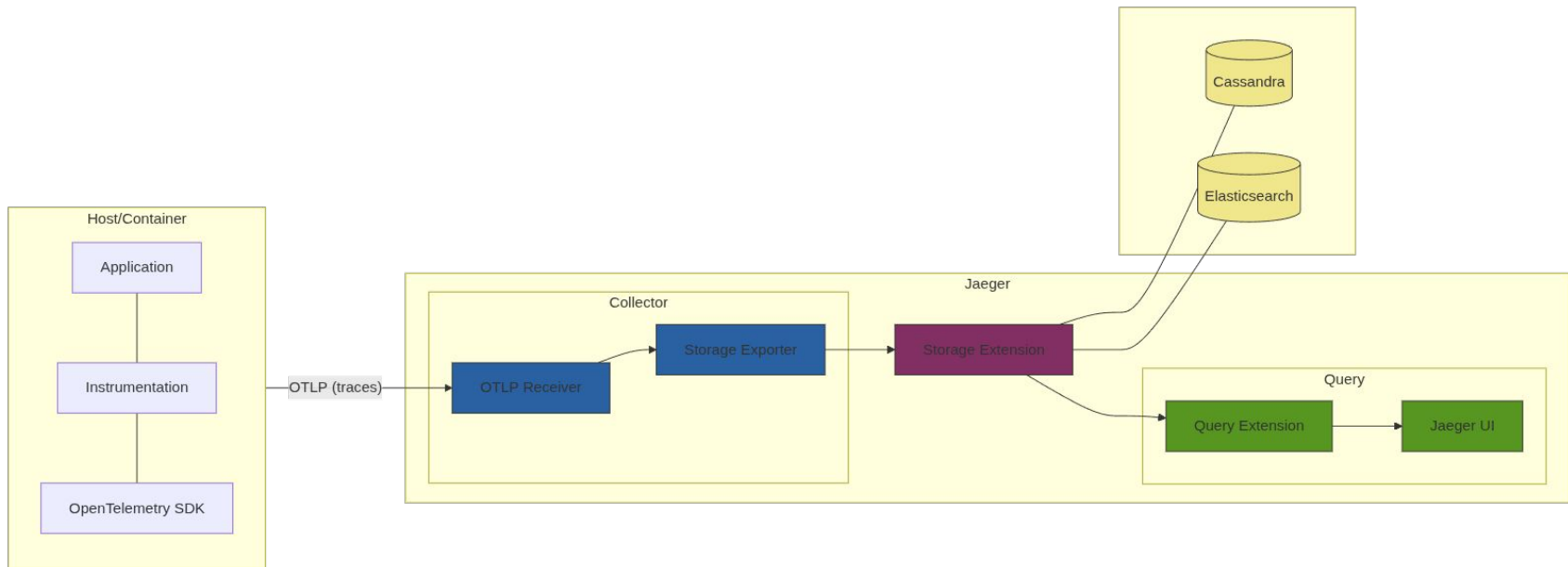
based on OpenTelemetry collector



Jaeger V2

- Jaeger functionality implemented as OpenTelemetry components
 - New storage APIs for Jaeger 2.x
 - Allow for backwards compatibility to Jaeger 1.x storage
 - Native OTLP support for storage
-
- Single binary for all components
 - Adopts OpenTelemetry configuration style (no CLI flags)

Jaeger v2 Binary Architecture



V2 Configuration



```
service:
  extensions: [jaeger_storage, jaeger_query, jaeger_mcp, remote_sampling]
  pipelines:
    traces:
      receivers: [otlp, jaeger, zipkin]
      processors: [batch, adaptive_sampling]
      exporters: [jaeger_storage_exporter]

extensions:
  jaeger_query:
    storage:
      traces: some_store
      traces_archive: another_store
  ui:
    config_file: ./cmd/jaeger/config-ui.json
  jaeger_storage:
    backends:
      some_store:
        memory:
          max_traces: 100000
      another_store:
        memory:
          max_traces: 100000
```

New Key Features Since Kubecon NA + Roadmap

Jaeger Model Context Protocol Server (MCP)

MCP extension



Progressive Disclosure Architecture

"The key insight driving this design is progressive disclosure: rather than dumping entire traces into an LLM context, we provide tools that allow the LLM to follow a guided "drill-down" workflow":

- Search → Find candidate traces matching specific criteria (service, time range, attributes, duration)
- Map → Visualize trace structure (topology) without loading attribute data
- Diagnose → Identify the critical execution path that contributed to latency
- Inspect → Load full details only for specific, suspicious spans

MCP extension - tools



- `get_services` -> service names
- `search_traces` -> trace summaries
- `get_trace_topology` -> tree structure with metadata
- `get_critical_path` -> spans on the critical path
- `get_trace_errors` -> all error spans details
- `get_span_details` -> span details



New Features (Backend)

- Removed V1 in 2.14.0 (2026-01-01)
- Added direct Clickhouse support
- Backends improvements
 - Cassandra v5, Elasticsearch v9, OpenSearch v3
- MCP Server
- Added Sigv4 auth: OpenSearch/Elasticsearch, Prometheus client
- Helm Chart for Jaeger v2
- Live demo <https://www.jaegertracing.io/demo/>

New Features (UI)



- Dark mode
- Trace timeline view improvements
 - Navigation tree
 - Span details in the side panel
- OpenTelemetry naming alignment
- Incomplete trace detection



Roadmap

- AI assistant for trace analysis
- Jaeger for Agentic Observability
- Refactoring and improving visualizations for topology

And more useful and interesting capabilities coming in the future

Q&A

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