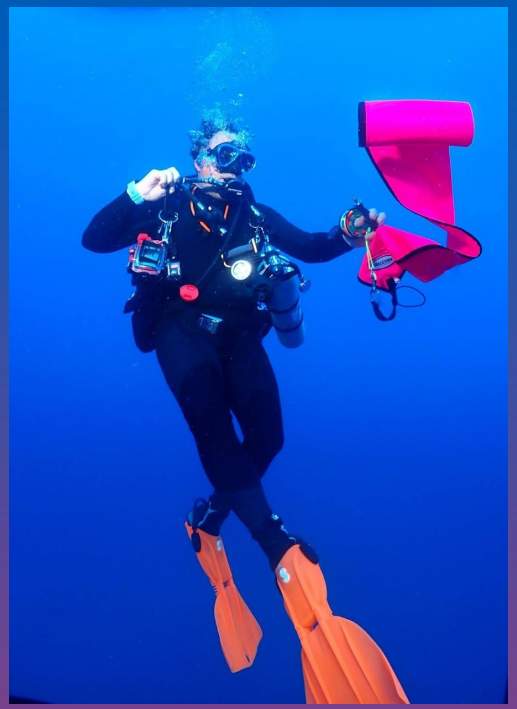
Implementing OpenTelemetry and Jaeger with OpenSearch

Jonah Kowall [@jkowall]

OpenSearchCon
NORTH AMERICA | San Francisco 2024

Jonah Kowall
@jkowall

- 15+ years Ops, Security, Performance for Enterprises and Startups
  - Security - CISSP, CISA, PCI
  - Thomson Reuters, MFG.com (Bezos Expeditions)
- Research VP Gartner (Infrastructure & Operations) 4 years
- VP Product+Strategy AppDynamics→Cisco 4 years
- Kentik CTO 1 year
- Logz.io CTO 3 years
- Aiven VP Product 2 years
- **Next roll to be announced soon!**
- OpenSearch Technical Steering Committee Member
- Maintainer : Jaeger
- Contributor: OpenSearch, OpenTelemetry



Observability 3 most popular Signals

Why Trace

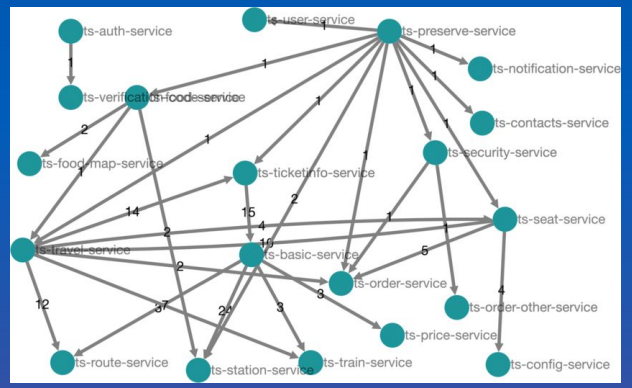
Microservices Architectures

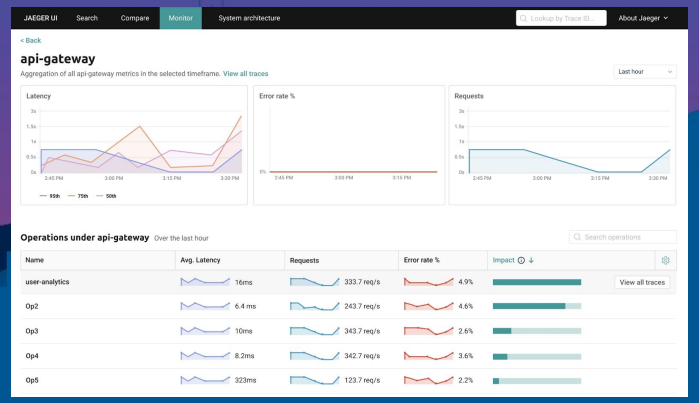
63% of enterprises are using Microservices most often managed by different teams.

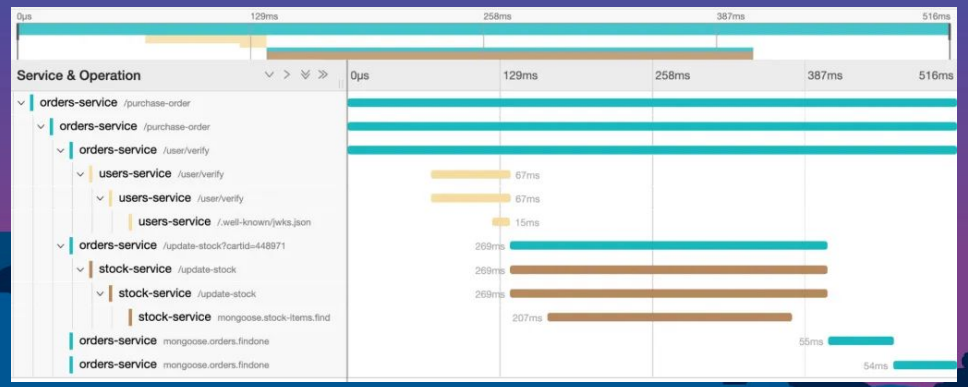
When things break or are slow, how do you find the right team to help resolve the problem?

Benefits of Distributed Tracing

- Root cause analysis
- Build a service map of relationships and dependencies
- Measure and drill into specific user actions
- Collaborate across teams
- Monitor and maintain SLAs

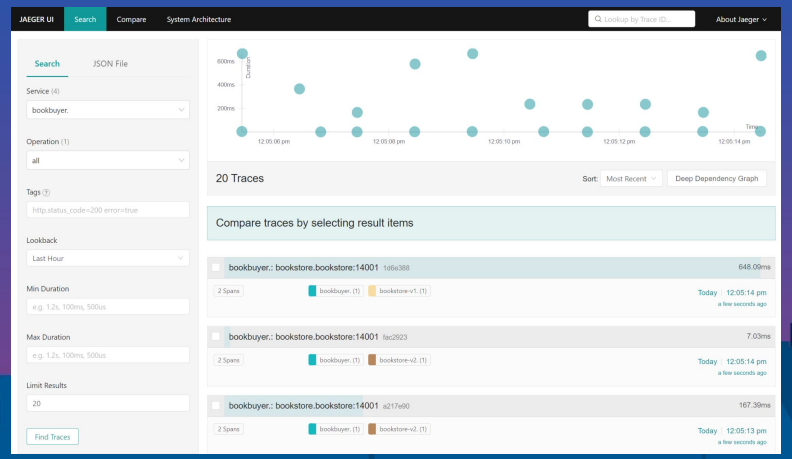






Needed for tracing

- Instrumentation (OpenTelemetry)
- Data Collection (OpenTelemetry)
- Storage & Analysis & Visualization (Jaeger/OpenSearch)



Tracing terminology

- **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

It's all about the context

```
Causal relationships between Spans in a single Trace


       [Span A]  ←←←(the root span)
           |
    +------+------+
    |             |
 [Span B]      [Span C] ←←←(Span C is a `ChildOf` Span A)
    |             |
 [Span D]      +---+-------+
               |           |
          [Span E]    [Span F] >>> [Span G] >>> [Span H]
                                        ↑
                                        ↑
                                        ↑
                       (Span G `FollowsFrom` Span F)
```

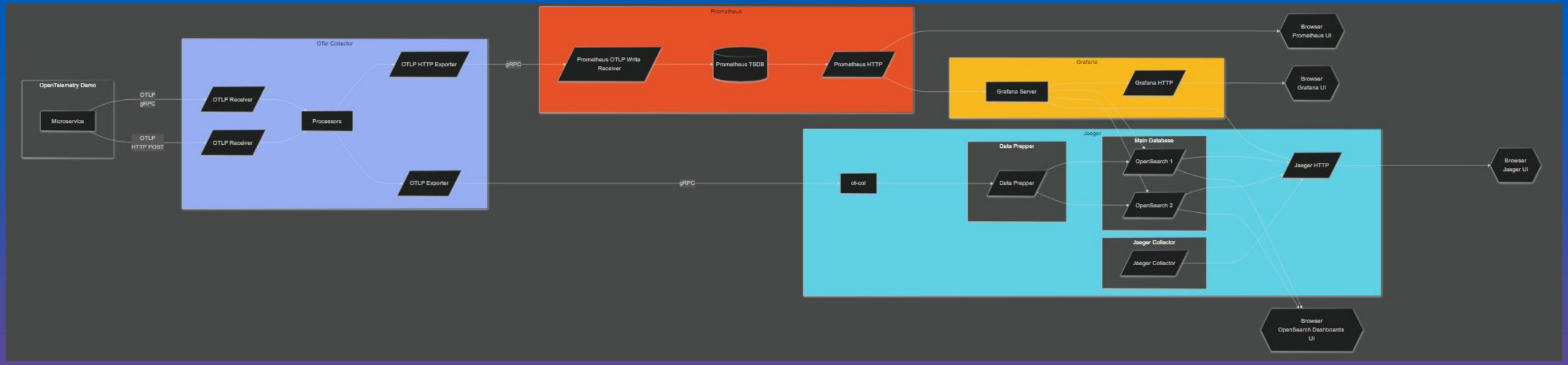
```
Temporal relationships between Spans in a single Trace


--|-------|-------|-------|-------|-------|-------|-------|-> time

 [Span A·······················································]
   [Span B·················································]
      [Span D·····································]
    [Span C·································]
         [Span E·····]        [Span F··] [Span G··] [Span H··]
```

Measure errors, latency, and other indicators across each span

Architecture of Observability Stack

Thanks to @YANG-DB :

opentelemetry-demo Public
forked from open-telemetry/opentelemetry-demo

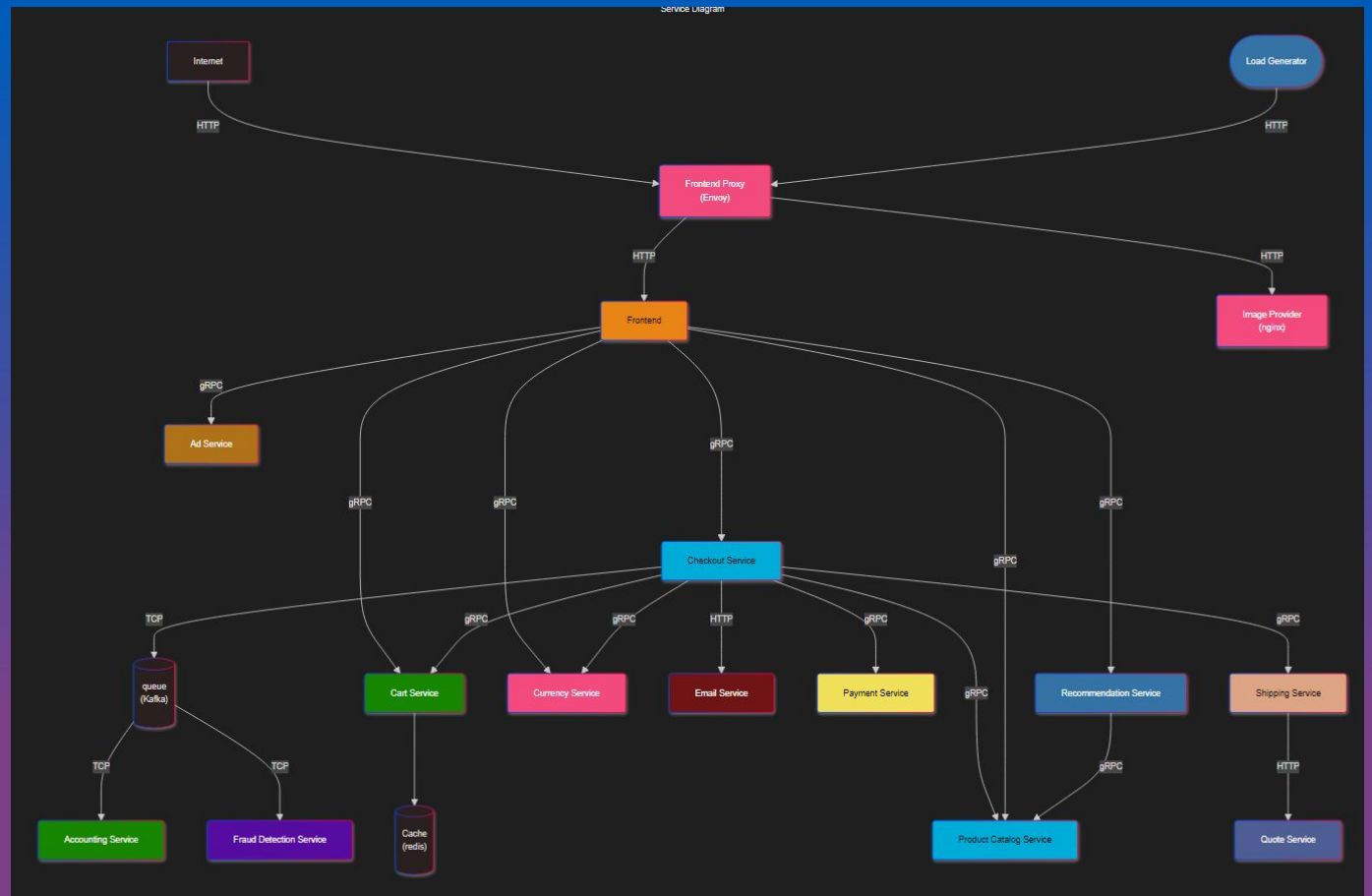


To follow along or see the modified demo for this talk :
https://github.com/jkowall/opensearch-talk

Architecture of the Application

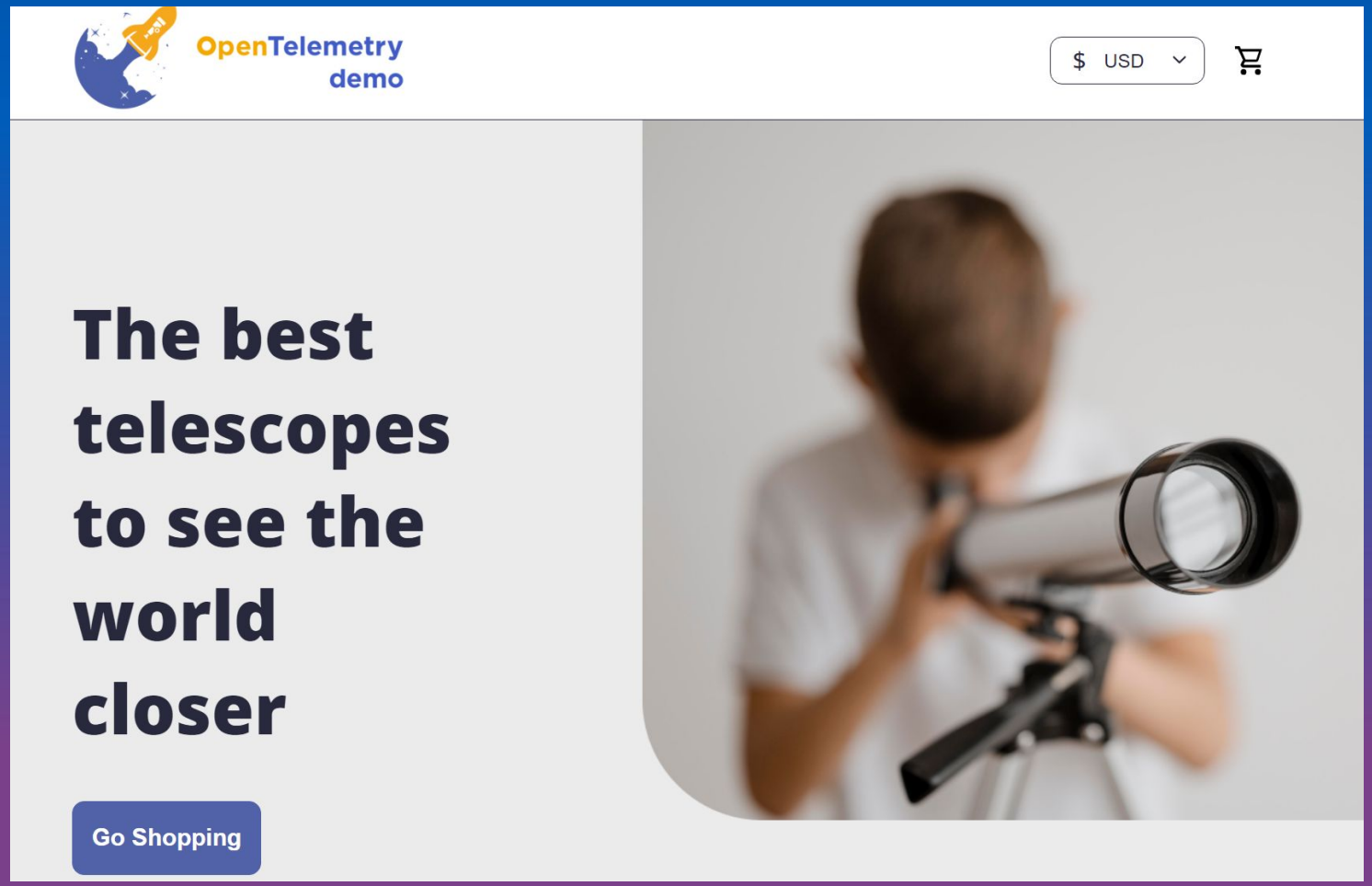
Thanks to OpenTelemetry :

https://opentelemetry.io/docs/demo/architecture/



Frontend Application

E-commerce application - http://frontend:8080/



Jaeger

http://localhost:16686

JAEGER UI | Search | Compare | System Architecture | Monitor | Lookup by Trace ID... | About Jaeger

Search | Upload

Service (15)
frontend

Operation (19)
all

Tags
http.status_code=200 error=true

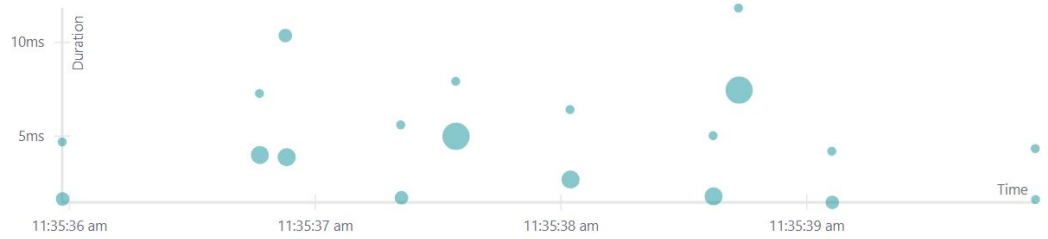
Lookback
Last Hour

Max Duration
e.g. 1.2s, 100...

Min Duration
e.g. 1.2s, 100...

Limit Results
20

Find Traces



20 Traces | Sort: Most Recent | Download Results | Deep Dependency Graph

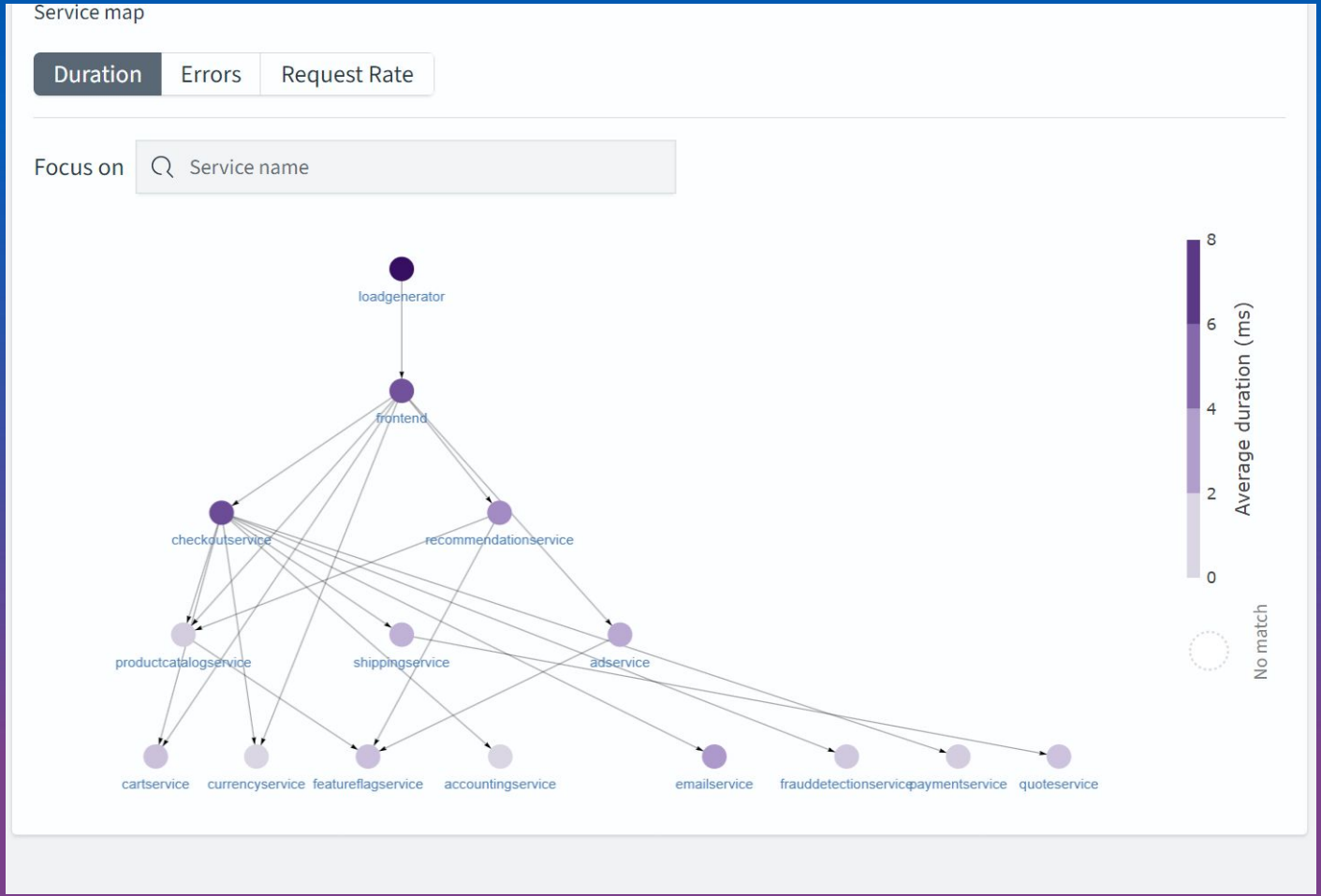
Compare traces by selecting result items

frontend: HTTP GET fad94d2 — 1.62ms
2 Spans | frontend (2) | Today 11:35:39 am | a few seconds ago

loadgenerator: GET 46b2095 — 4.33ms
2 Spans | frontend (1) | loadgenerator (1) | Today 11:35:39 am | a few seconds ago

OpenSearch Dashboards

http://dashboards:5601/app/observability-traces#/traces



Grafana

http://localhost:3000/grafana/explore

Search or jump to... ctrl+k
Sign in
Home › Explore
Add to dashboard

A (Jaeger)
Query type: Search | TraceID
Import trace
Service Name: frontend
Operation Name: All
Tags: http.status_code=200 error=true
Min Duration: e.g. 1.2s, 100ms, 500us
Max Duration: e.g. 1.2s, 100ms, 500us
Limit:
+ Add query
Query inspector

Table

Trace ID	Trace name	Start time	Du
7882d31deeea54...	frontend: HTTP GET	2024-08-24 11:41:...	1
146ed7a33e974e...	frontend: GET	2024-08-24 11:41:...	1
3f386436f024afe...	frontend: HTTP GET	2024-08-24 11:41:...	4
e26e29dc3579e6...	loadgenerator: GET	2024-08-24 11:41:...	8
805debb1483fbb...	frontend: HTTP GET	2024-08-24 11:41:...	2
74142d4ab01a79...	loadgenerator: GET	2024-08-24 11:41:...	4
70a8cb97eaf8231...	frontend: HTTP GET	2024-08-24 11:41:...	3

A (Jaeger)
Query type: Search | TraceID
Import trace
Trace ID: 3f386436f024afec977838b070996e2f
+ Add query
Query inspector

Trace
frontend: HTTP GET 4.33ms
Give feedback | Trace ID | Export
2024-08-24 11:41:41.809 GET 200 /api/data
Span Filters
5 spans
Prev | Next
0µs | 1.08ms | 2.17ms | 3.25ms | 4.33ms
Service ...
frontend HTTP GET (4.33ms)
grpc.oteldemo.AdService/GetAd
adservice oteldemo.AdServic
2ms
getAdsByCategory (61µs)
61µs
getRandomAds (81µs)
81µs

Things we did not cover

The focus is on tracing as a signal

1. Use of metrics from traces to monitor and measure application performance

2. How to debug something slow with traces

OpenSearchCon
NORTH AMERICA | San Francisco 2024

" Thank You "

If you want to learn more about tracing join the CNCF Slack :
#Jaeger or #OpenTelemetry

jkowall

@jkowall