Source: Andres Siimon on Unsplash

HOLISTIC OBSERVABILITY WITH SYNTHETIC MONITORING AND OpenTelemetry SIGNALS

CARLY RICHMOND

elastic

carlyrichmond.bsky.social

ABOUT ME

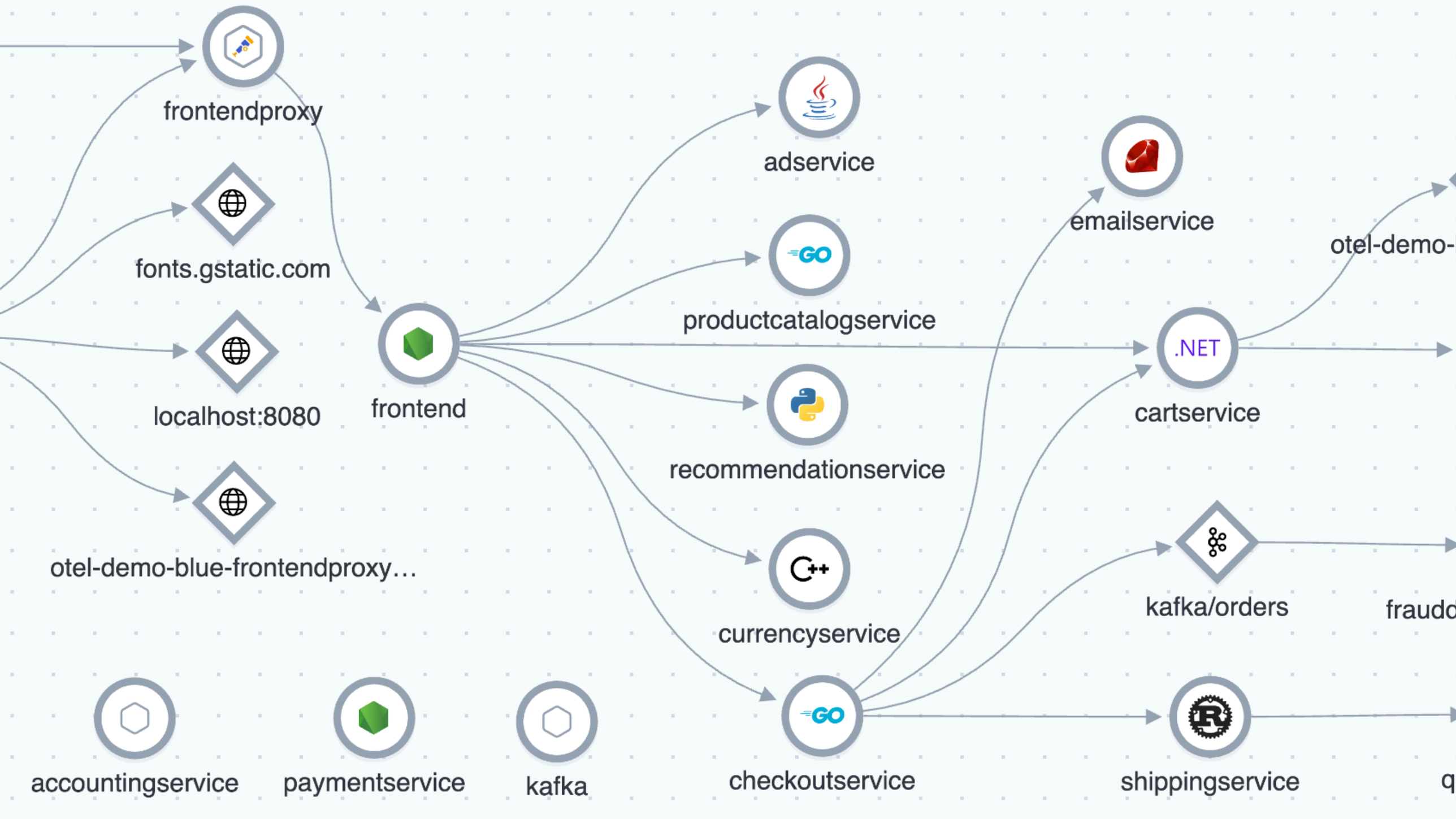
- Developer Advocate & Manager @ elastic
- Frontend Engineer, Speaker & Blogger

SCAN ME

carlyrichmond.bsky.social

"**Holistic**: relating to the whole of something or to the total system instead of just to its parts"

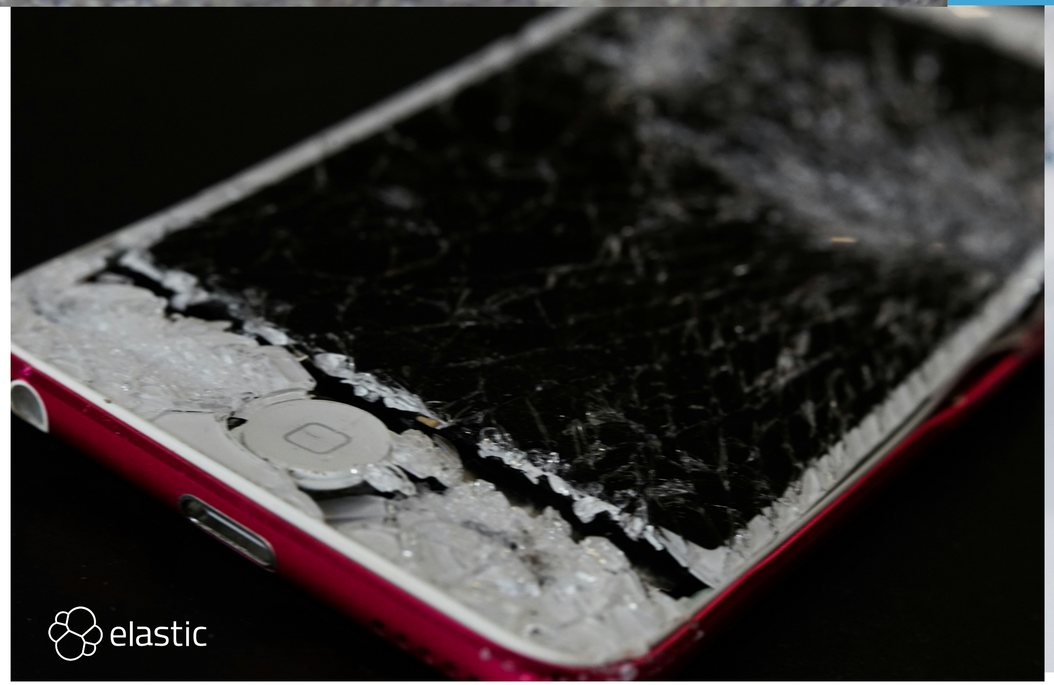
HOLISTIC, CAMBRIDGE ENGLISH DICTIONARY


frontendproxy
fonts.gstatic.com
localhost:8080
otel-demo-blue-frontendproxy…
frontend
adservice
productcatalogservice
recommendationservice
currencyservice
checkoutservice
emailservice
cartservice
kafka/orders
shippingservice
otel-demo-
fraudo
q
accountingservice
paymentservice
kafka

log — -zsh — 104×33

```
2025-04-23T20:47:41.233Z INFO 1 --- [records-service] [r  080-exec-5] c.e.o.r.s.r.s.record.RecordCont  ng all records

2025-04-23T20:47:41.233Z INFO 1 --- [records-ser  -exec-5] c.e.o.r.s.r.s.record.Record  records in catalog

[otel.javaagent 2025-04-24 08:30:20:767 +0000]  dcd88eee64c8a9f4cd27bcba2aa14  .io/...] ERROR
io.opentelemetry.exporter.internal.http.HttpExp  port metrics. The request  error message: timeout

java.io.InterruptedIOException: timeout

at okhttp3.internal.connection.RealCall.timeoutEx

at okhttp3.internal.connection.RealCall.callDone(R

at okhttp3.internal.connection.RealCall.noMoreExchan  5)

at okhttp3.internal.connection.RealCall.getResponseWit

at okhttp3.internal.connection.RealCall$AsyncCall.run(R

at java.base/java.util.concurrent.ThreadPoolExecutor.runW

at java.base/java.util.concurrent.ThreadPoolExecutor$Worker

at java.base/java.lang.Thread.run(Unknown Source)

Caused by: java.io.IOException: Canceled

at okhttp3.internal.http.RetryAndFollowUpInterceptor.int  2)

at okhttp3.internal.http.RealInterceptorChain.proceed

at io.opentelemetry.exporter.sender.okhttp.interna  91)

at okhttp3.internal.http.RealInterceptorChain.r

at okhttp3.internal.connection.RealCall.get  ok

... 4 more

2025-04-24T08:43:18.094Z INFO 1 --- [  c-8] c.e.o.r.  : Waiting...

2025-04-24T08:43:18.103Z INFO 1 --- [re  exec-8] c.e.o.r.s.  ier : Getting all records

2025-04-24T08:43:18.104Z INFO 1 --- [re  80-exec-8] c.e.o.r.s.r.s  sitory : Finding all records in catalog

2025-04-24T08:43:18.468Z INFO 1 --- [re  -8080-exec-3] c.e.o.r.s.r.s.re  RecordController : Waiting...

2025-04-24T08:43:18.468Z INFO 1 --- [reco  io-8080-exec-3] c.e.o.r.s.r.s.record.RecordController : Getting featured records

2025-04-24T08:43:19.037Z INFO 1 --- [records-se  [io-8080-exec-10] c.e.o.r.s.r.s.record.RecordController : Waiting...
```

Source: Various on Unsplash



elastic

carlyrichmond.bsky.social

Logs

Traces

Source: Ruben. on Unsplash

Logs

Metrics

Traces

Source: Ruben. on Unsplash

Logs

Metrics

Traces

DevTools - arstechnica.com/

Elements | Console | Sources | Network | Performance | » | 1 | 3

top | Filter | All levels

```
[DOM] Input elements should have autocomplete attributes (suggested: "current-password"): (More info: https://goo.gl/9p2vKq)  (index):1
  <input type="password" name="password" id="password" placeholder="Password" aria-label="Password">
6  [Violation] 'setTimeout' handler took <N>ms
▶Network quality values are overridden using a holdback experiment, and so may be inaccurate  gpt.js:1
▶CNE Player: no interlude for "" ("Availability: unsupported template") Error: Availability: unsupported template  arstechnica.js:2
    at new t (arstechnica.js:2)
    at H (arstechnica.js:2)
[Violation] 'setInterval' handler took 55ms  services.min.js?1555014600:1
▶GET https://mid.rkdms.com/ids?ptk=17c1789b-e660-493b-aa74-3c8fb990dc5f&pubid=CONDENAST 404  htw-condenast.js:1
Evidon -- evidon-notice-link not found on page, cant display the consent link.  evidon-sitenotice-tag.js:1
18  [Violation] Avoid using document.write(). <URL>
```

Console

Source: Ruben. on Unsplash

Logs

Metrics

Traces

record-store-server-java

Tell us what you think!

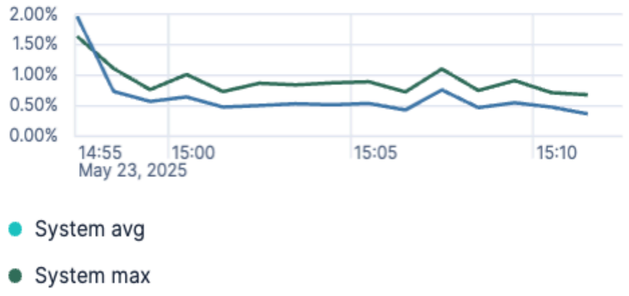
Environment All

Overview | Transactions | Dependencies | Errors | Metrics | Infrastructure β | Service Map | Logs | Alerts | Dashboards

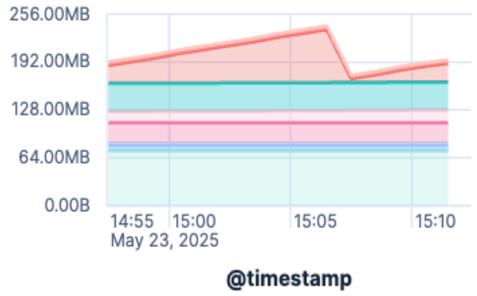
Search metrics (E.g. process.pid = "1234")

Last 15 minutes | Refresh

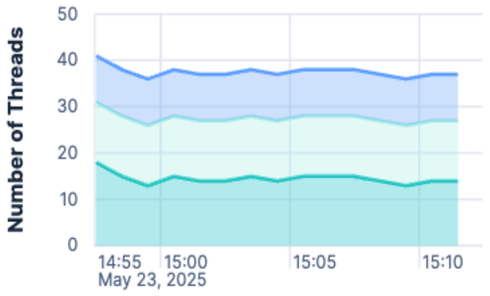
CPU Usage



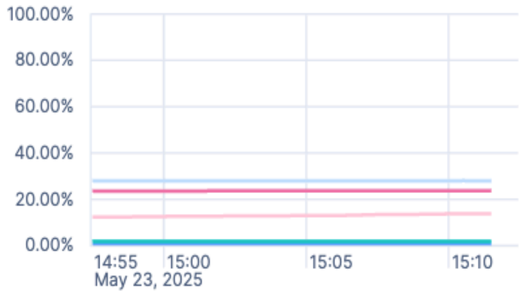
Absolute Memory Usage



Thread Count



Relative Memory Usage



Heap Memory Region Statistics

Heap Region	Limit	Used avg	Used avg [%]	Max used after GC	Max used after GC [%]	Committed avg
G1 Old Gen	1.91GB	37.73MB	1.93%	37.73MB	1.93%	48.00MB
G1 Eden Spac	-	36.80MB	-	0.00B	-	77.00MB
G1 Survivor Sp	-	4.58MB	-	4.69MB	-	5.00MB

Non-Heap Memory Region Statistics

Non-Heap Region	Limit	Used avg	Used avg [%]	Used max	Used max [%]	Comm
Metaspace	-	73.07MB	-	73.07MB	-	74.
Compressed Class Space	1.00GB	8.90MB	0.87%	8.90MB	0.87%	9.
CodeHeap 'non-nmethods'	5.57MB	1.56MB	28.11%	1.57MB	28.11%	2.

Source: Unsplash

elastic

carlyrichmond.bsky.social

LCP

LARGEST CONTENTFUL PAINT

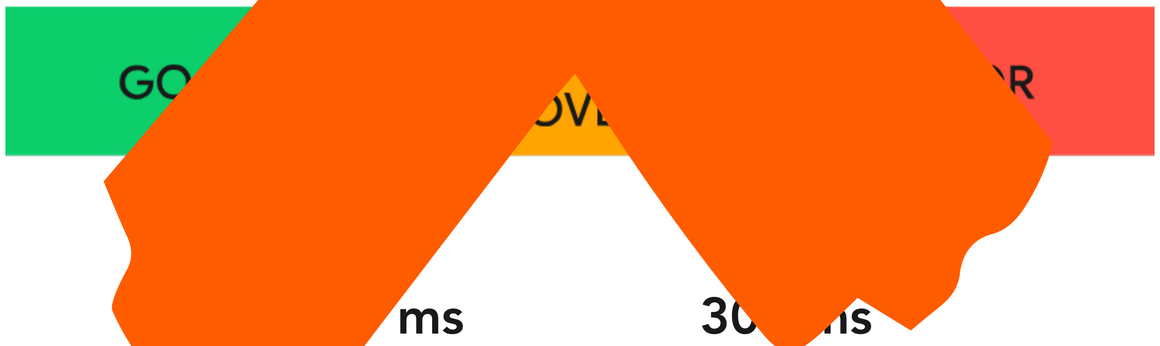
The render time of the largest image or text block visible within the viewport

https://web.dev/explore/learn-core-web-vitals

FID

FIRST INPUT LAY

The time from when a use on or command and the page executes it



https://web.dev/explore/learn-core-web-vitals

INP

INTERACTION TO NEXT PAINT

The longest duration of latency of all click, tap, and keyboard interactions with a page throughout its lifespan, excluding outliers

https://web.dev/articles/inp#inp-vs-fid

CLS

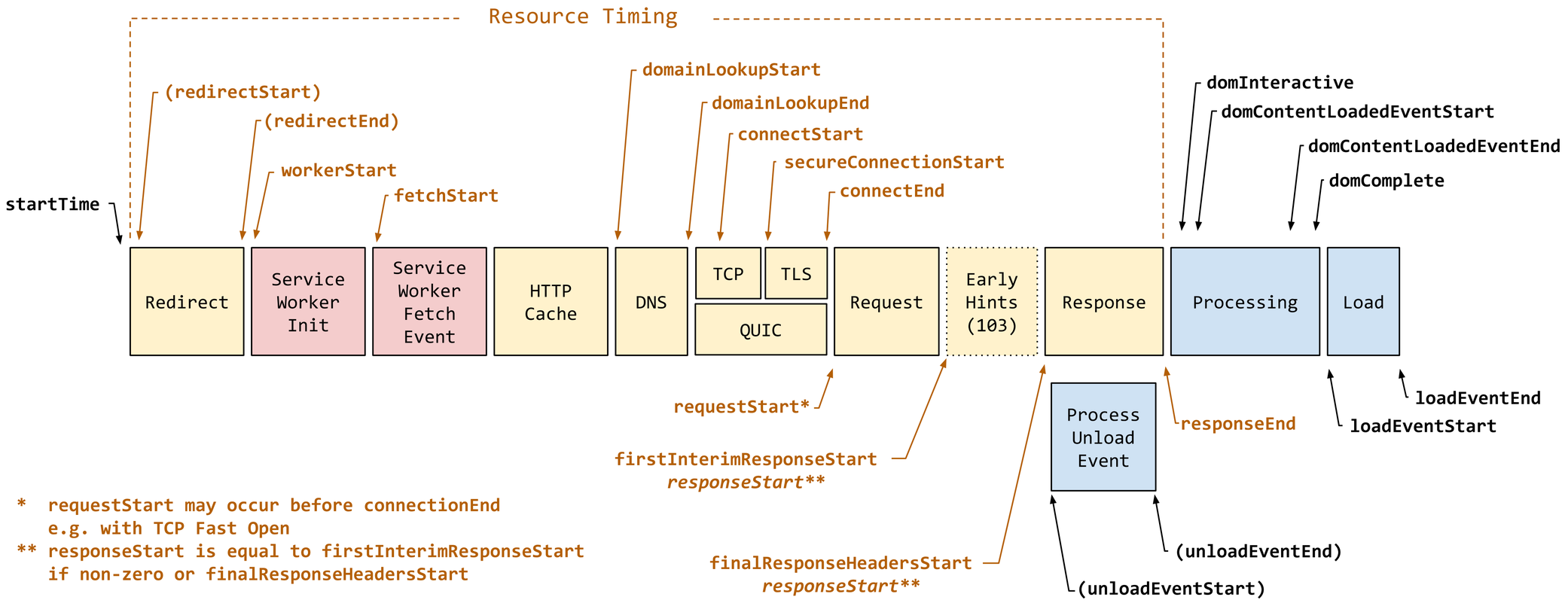
CUMULATIVE LAYOUT SHIFT

The *largest burst* of *layout* shift scores for every unexpected layout shift that occurs during the entire lifespan of a page

https://web.dev/explore/learn-core-web-vitals

OTHER VITALS

PAGE LOAD



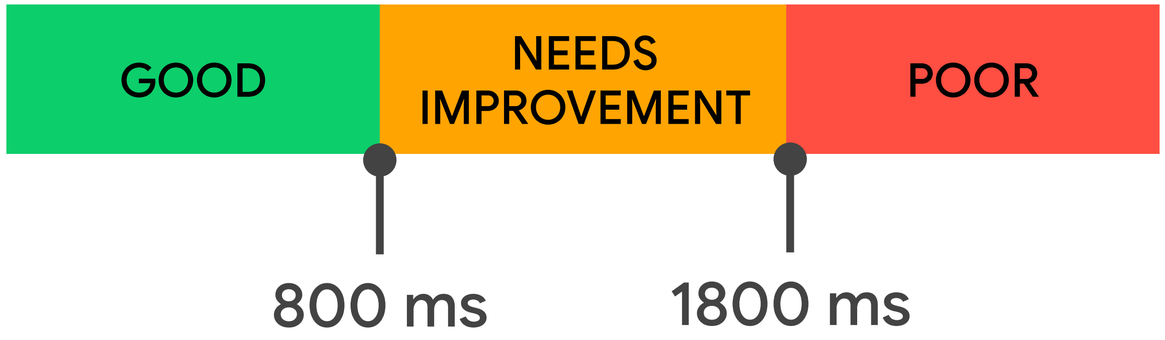
* requestStart may occur before connectionEnd
e.g. with TCP Fast Open
** responseStart is equal to firstInterimResponseStart
if non-zero or finalResponseHeadersStart

https://web.dev/articles/ttfb

TTFB

TIME TO FIRST BYTE

The time between the request for a resource and when the first byte of a response begins to arrive.



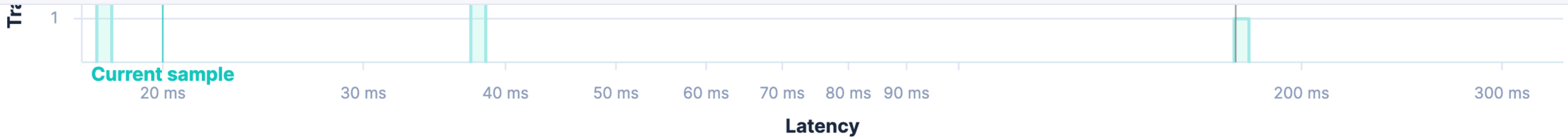
https://web.dev/articles/ttfb

Source: Ruben. on Unsplash

Logs

Metrics

Traces



All transactions

Trace sample

1 of 8

Investigate

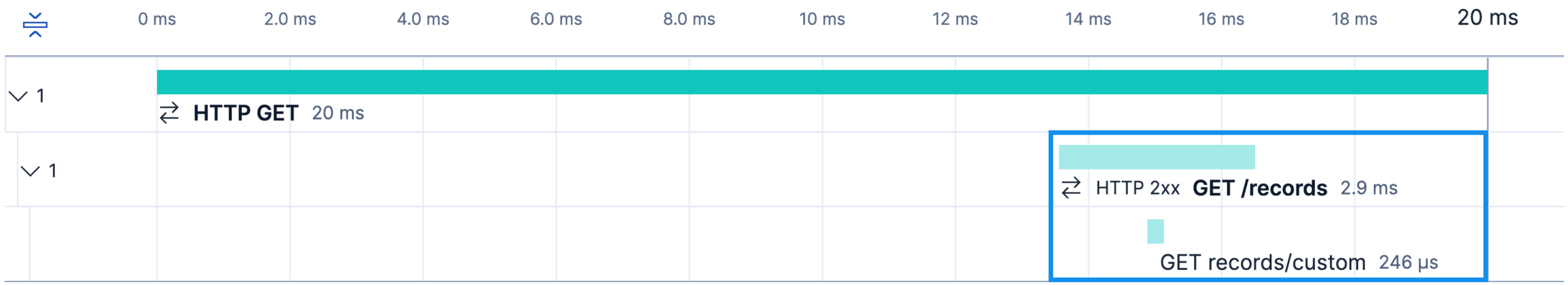
View full trace

5 minutes ago | 20 ms (100% of trace)

Timeline Metadata Logs

Show critical path

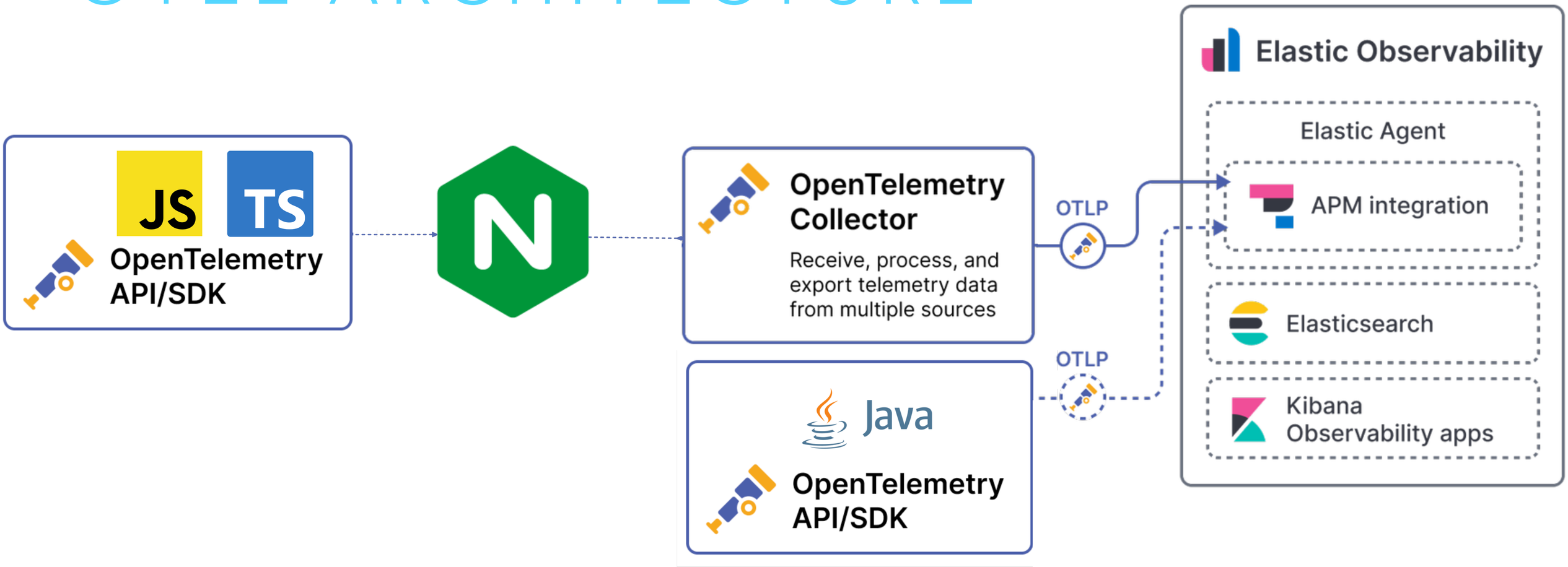
Services records-ui-web record-store-server-java



Source: Thomas Kinto on Unsplash

OpenTelemetry

OTEL ARCHITECTURE



"**Holistic**: relating to the whole of something or to the **total system** instead of just to its **parts**"

HOLISTIC, CAMBRIDGE ENGLISH DICTIONARY

```
97      resource: resource,
98      spanProcessors: [
99          // Send each completed span through the exporter
100         new SimpleSpanProcessor(exporter),
101         new SimpleSpanProcessor(new ConsoleSpanExporter())
102     ]
103 });
104
105 // Register the provider with propagation and set up the async context manager for spans
106 provider.register({
107     contextManager: new ZoneContextManager(),
108     propagator: new CompositePropagator({
109         propagators: [new W3CBaggagePropagator(), new W3CTraceContextPropagator()]
110     })
111 });
112
113 export class ClientTelemetry {
114     private static instance: ClientTelemetry;
115     private initialized = false;
116
117     private constructor() {}
118
119     // Obtain singleton instance of provider
120     public static getInstance(): ClientTelemetry {
121         if (!ClientTelemetry.instance) {
```

"**Observability** is the ability to collect data about **programs' execution**, **modules' internal states**, and the **communication among components**."

–OBSERVABILITY (SOFTWARE), WIKIPEDIA

JavaScript | OpenTelemetry

https://opentelemetry.io/docs/languages/js/

New Chrome available

OpenTelemetry

Docs Ecosystem Status Community Blog English Search this site...

Docs

- What is OpenTelemetry?
- Getting Started
- Concepts
- Demo
- Language APIs & SDKs
  - SDK Config
  - C++
  - .NET
  - Erlang/Elixir
  - Go
  - Java
  - JavaScript
    - Getting Started
    - Instrumentation
    - Libraries
    - Exporters
    - Context
    - Propagation
    - Resources
    - Sampling
    - Serverless

JavaScript

A language-specific implementation of OpenTelemetry in JavaScript (for Node.js & the browser).

This is the OpenTelemetry JavaScript documentation. OpenTelemetry is an observability framework – an API, SDK, and tools that are designed to aid in the generation and collection of application telemetry data such as metrics, logs, and traces. This documentation is designed to help you understand how to get started using OpenTelemetry JavaScript.

Status and Releases

The current status of the major functional components for OpenTelemetry JavaScript is as follows:

Traces	Metrics	Logs
Stable	Stable	Development

For releases, including the latest release, see Releases.

Warning

Client instrumentation for the browser is **experimental** and mostly **unspecified**. If you are interested in helping out, get in touch with the Client Instrumentation SIG.

View page source
Edit this page
Create documentation issue

Status and Releases
Version Support
Repositories
Help or Feedback

Backlog · Spec: Client SDK ar

github.com/orgs/open-telemetry/projects/19/views/1

open-telemetry / Projects / **Spec: Client SDK and Instrumentation**

Type / to search

Spec: Client SDK and Instrumentation

Off track

Backlog

Filter by keyword or by field

Discard

No Status 1

semantic-conventions #1903

Clarify `OS terminology` and `OS identifiers` in device.app.lifecycle event

Icebox 0

Backlog 4

opentelemetry-js-contrib #2151

Implement browser Page Navigation Timing event instrumentation

opentelemetry-js-contrib #2150

Implement event-based instrumentation for web errors

opentelemetry-js-contrib #2140

Implement UserAction event instrumentation

opentelemetry-specification #3811

Standardize Server-Timing: traceparent "propagator" across vendors

In Progress 4

opentelemetry-js-contrib #2372

Implement PageView event instrumentation

opentelemetry-js-contrib #2358

Session handling implementation

opentelemetry-js-contrib #2152

Implement ResourceTiming event instrumentation for browser

opentelemetry-js-contrib #1461

Core Web Vitals Plugin

web-vitals - npm

https://www.npmjs.com/package/web-vitals#overview

Pro Teams Pricing Documentation

npm

Search packages

Search

Sign Up

Sign In

web-vitals TS

4.2.3 • Public • Published 2 months ago

Readme | Code Beta | 0 Dependencies | 18,887 Dependents | 76 Versions

web-vitals

- Overview
- Install and load the library
  - From npm
  - From a CDN
- Usage
  - Basic usage
  - Report the value on every change
  - Report only the delta of changes
  - Send the results to an analytics endpoint
  - Send the results to Google Analytics
  - Send the results to Google Tag Manager
  - Send attribution data
  - Batch multiple reports together

Install

```
npm i web-vitals
```

Repository

github.com/GoogleChrome/web-vitals

Homepage

github.com/GoogleChrome/web-vitals#...

Weekly Downloads

4,090,498

Version

4.2.3

License

Apache-2.0

Unpacked Size

Total Files

OpenTelemetry

Source: Andres Siimon on Unsplash

DEMO

elastic

carlyrichmond.bsky.social

frontend.tracer.ts

```
1 /* OpenTelemetry Frontend packages */
2
3 // Import the WebTracerProvider, which is the core provider for browser-based tracing
4 import { WebTracerProvider } from '@opentelemetry/sdk-trace-web';
5
6 // Used to auto-register built-in instrumentations like page load and user interaction
7 import { registerInstrumentations } from '@opentelemetry/instrumentation';
8
9 // Document Load Instrumentation automatically creates spans for document load events
10 import { DocumentLoadInstrumentation } from '@opentelemetry/instrumentation-document-load';
11
12 // Automatically creates spans for user interactions like clicks
13 import { UserInteractionInstrumentation } from '@opentelemetry/instrumentation-user-
   interaction';
14
15 // Import the auto-instrumentations for web, which includes common libraries, frameworks and
   document load
16 import { getWebAutoInstrumentations } from '@opentelemetry/auto-instrumentations-web';
17
18 // This context manager ensures span context is maintained across async boundaries in the
   browser
19 import { ZoneContextManager } from '@opentelemetry/context-zone';
20
21 /* Packages for exporting traces */
22 // SimpleSpanProcessor immediately forwards completed spans to the exporter
23 import { ConsoleSpanExporter, SimpleSpanProcessor } from '@opentelemetry/sdk-trace-base';
24
25 // Import the OTLP HTTP exporter for sending traces to the collector over HTTP
26 import { OTLPTraceExporter } from '@opentelemetry/exporter-trace-otlp-http';
27
28 /* Packages for handling logs */
29 import { logs, SeverityNumber } from '@opentelemetry/api-logs';
30 import { BatchLogRecordProcessor, LoggerProvider } from '@opentelemetry/sdk-logs';
31 import { OTLPLogExporter } from '@opentelemetry/exporter-logs-otlp-http';
32
33 // These help with logging, diagnostics, and traces
34 import { diag, DiagConsoleLogger, DiagLogLevel } from '@opentelemetry/api';
35
36 // Defines a Resource to include metadata like service.name, required by Elastic
37 import { resourceFromAttributes, detectResources } from '@opentelemetry/resources';
38
39 // Experimental detector for browser environment
40 import { browserDetector } from '@opentelemetry/opentelemetry-browser-detector';
41
42 // Provides standard semantic keys for attributes, like service.name
43 import { ATTR_SERVICE_NAME } from '@opentelemetry/semantic-conventions';
44
45 // Context Propagation across signals
46 import {
47     CompositePropagator,
48     W3CBaggagePropagator,
49     W3CTraceContextPropagator
50 } from '@opentelemetry/core';
51
52 /* Custom dependencies */
53
54 import { LOGS_URL, TRACE_URL } from './constants';
55 import { WebVitalsInstrumentation } from './web-vitals.instrumentation';
56
57 // Enable OpenTelemetry debug logging to the console
58 diag.setLogger(new DiagConsoleLogger(), DiagLogLevel.DEBUG);
59
60 // Define resource metadata for the service, used by exporters (Elastic requires service.name)
61 const SERVICE_NAME = 'records-ui-web';
62
63 const detectedResources = detectResources({ detectors: [browserDetector] });
64 let resource = resourceFromAttributes({
65     [ATTR_SERVICE_NAME]: SERVICE_NAME,
66     'service.version': 1,
67     'deployment.environment': 'dev'
68 });
69 resource = resource.merge(detectedResources);
70
71 // Configure logging to send to the collector via nginx
72 const logExporter = new OTLPLogExporter({
73     url: LOGS_URL // nginx proxy
74 });
75
76 const loggerProvider = new LoggerProvider({
77     resource: resource,
78     processors: [new BatchLogRecordProcessor(logExporter)]
79 });
80
81 logs.setGlobalLoggerProvider(loggerProvider);
82
83 const logger = logs.getLogger('default', '1.0.0');
84 logger.emit({
85     severityNumber: SeverityNumber.INFO,
86     severityText: 'INFO',
87     body: 'Logger initialized'
88 });
89
90 // Configure the OTLP exporter to talk to the collector via nginx
91 const exporter = new OTLPTraceExporter({
92     url: TRACE_URL // nginx proxy
93 });
94
95 // Instantiate the trace provider and inject the resource
96 const provider = new WebTracerProvider({
97     resource: resource,
98     spanProcessors: [
99         // Send each completed span through the exporter
100         new SimpleSpanProcessor(exporter),
101         new SimpleSpanProcessor(new ConsoleSpanExporter())
102     ]
103 });
104
105 // Register the provider with propagation and set up the async context manager for spans
106 provider.register({
107     contextManager: new ZoneContextManager(),
108     propagator: new CompositePropagator({
109         propagators: [new W3CBaggagePropagator(), new W3CTraceContextPropagator()]
110     })
111 });
112
113 export class ClientTelemetry {
114     private static instance: ClientTelemetry;
115     private initialized = false;
116
117     private constructor() {}
118
119     // Obtain singleton instance of provider
120     public static getInstance(): ClientTelemetry {
121         if (!ClientTelemetry.instance) {
122             ClientTelemetry.instance = new ClientTelemetry();
123             ClientTelemetry.instance.start();
124         }
125         return ClientTelemetry.instance;
126     }
127
128     // Initializer
129     public start() {
130         if (!this.initialized) {
131             // Enable automatic span generation for document load and user click interactions
132             registerInstrumentations({
133                 instrumentations: [
134                     // Automatically tracks when the document loads
135                     new DocumentLoadInstrumentation(),
136                     getWebAutoInstrumentations({
137                         '@opentelemetry/instrumentation-fetch': {
138                             propagateTraceHeaderCorsUrls: /.*/,
139                             clearTimingResources: true
140                         }
141                     }),
142                     // User events
143                     new UserInteractionInstrumentation({
144                         eventNames: ['click'] // instrument click events only
145                     }),
146                     // Custom Web Vitals instrumentation
147                     new WebVitalsInstrumentation()
148                 ]
149             });
150
151             logger.emit({
152                 severityNumber: SeverityNumber.INFO,
153                 severityText: 'INFO',
154                 body: 'Client Telemetry Initialised'
155             });
156             this.initialized = true;
157         }
158     }
159 }
```

elastic

carlyrichmond.bsky.social

```
// Enable OpenTelemetry debug logging to the console
diag.setLogger(new DiagConsoleLogger(), DiagLogLevel.DEBUG);

// Define resource metadata for the service, used by exporters (Elastic requires service.name)
const SERVICE_NAME = 'records-ui-web';

const detectedResources = detectResources({ detectors: [browserDetector] });
let resource = resourceFromAttributes({
    [ATTR_SERVICE_NAME]: SERVICE_NAME,
    'service.version': 1,
    'deployment.environment': 'dev'
});
resource = resource.merge(detectedResources);

// Configure logging to send to the collector via nginx
const logExporter = new OTLPLogExporter({
    url: LOGS_URL // nginx proxy
});

const loggerProvider = new LoggerProvider({
    resource: resource,
    processors: [new BatchLogRecordProcessor(logExporter)]
});

logs.setGlobalLoggerProvider(loggerProvider);

```

```
90 // Configure the OTLP exporter to talk to the collector via nginx
91 const exporter = new OTLPTraceExporter({
92     url: TRACE_URL // nginx proxy
93 });
94
95 // Instantiate the trace provider and inject the resource
96 const provider = new WebTracerProvider({
97     resource: resource,
98     spanProcessors: [
99         // Send each completed span through the exporter
100        new SimpleSpanProcessor(exporter),
101        new SimpleSpanProcessor(new ConsoleSpanExporter())
102    ]
103 });
104
105 // Register the provider with propagation and set up the async context manager for spans
106 provider.register({
107     contextManager: new ZoneContextManager(),
108     propagator: new CompositePropagator({
109         propagators: [new W3CBaggagePropagator(), new W3CTraceContextPropagator()]
110     })
111 });
112
113 export class ClientTelemetry {
114     private static instance: ClientTelemetry;
115     private initialized = false;
```

```
127
128     // Initializer
129     public start() {
130         if (!this.initialized) {
131             // Enable automatic span generation for document load and user click interactions
132             registerInstrumentations({
133                 instrumentations: [
134                     // Automatically tracks when the document loads
135                     new DocumentLoadInstrumentation(),
136                     getWebAutoInstrumentations({
137                         '@opentelemetry/instrumentation-fetch': {
138                             propagateTraceHeaderCorsUrls: /.*/,
139                             clearTimingResources: true
140                         }
141                     }),
142                     // User events
143                     new UserInteractionInstrumentation({
144                         eventNames: ['click'] // instrument click events only
145                     }),
146                     // Custom Web Vitals instrumentation
147                     new WebVitalsInstrumentation()
148                 ]
149             });
150
151             logger.emit({
152                 severityNumber: SeverityNumber.INFO,
```

frontend.tracer.ts

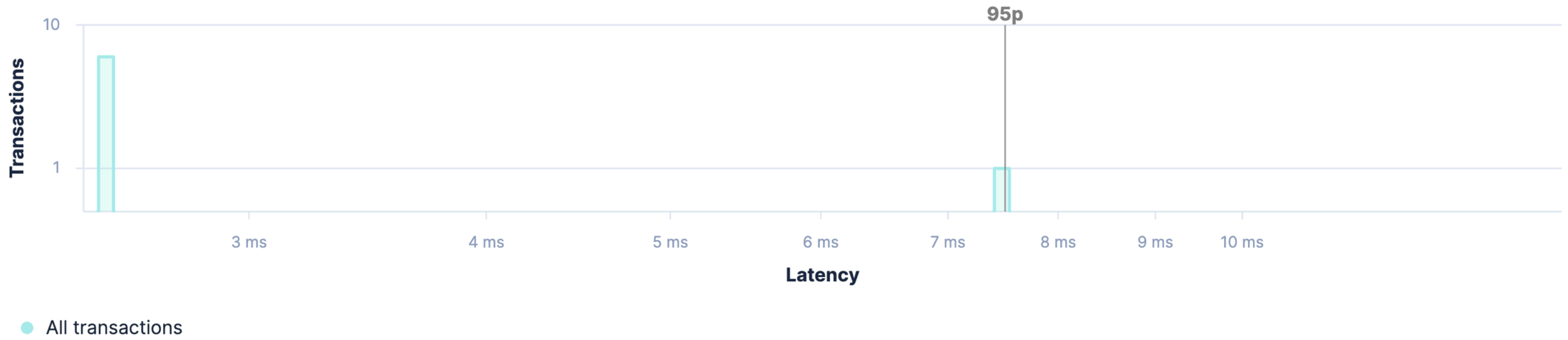
```
1 /* OpenTelemetry Frontend packages */
2
3 // Import the WebTracerProvider, which is the core provider for browser-based tracing
4 import { WebTracerProvider } from '@opentelemetry/sdk-trace-web';
5
6 // Used to auto-register built-in instrumentations like page load and user interaction
7 import { registerInstrumentations } from '@opentelemetry/instrumentation';
8
9 // Document Load Instrumentation automatically creates spans for document load events
10 import { DocumentLoadInstrumentation } from '@opentelemetry/instrumentation-document-load';
11
12 // Automatically creates spans for user interactions like clicks
13 import { UserInteractionInstrumentation } from '@opentelemetry/instrumentation-user-
   interaction';
14
15 // Import the auto-instrumentations for web, which includes common libraries, frameworks and
   document load
16 import { getWebAutoInstrumentations } from '@opentelemetry/auto-instrumentations-web';
17
18 // This context manager ensures span context is maintained across async boundaries in the
   browser
19 import { ZoneContextManager } from '@opentelemetry/context-zone';
20
21 /* Packages for exporting traces */
22 // SimpleSpanProcessor immediately forwards completed spans to the exporter
23 import { ConsoleSpanExporter, SimpleSpanProcessor } from '@opentelemetry/sdk-trace-base';
24
25 // Import the OTLP HTTP exporter for sending traces to the collector over HTTP
26 import { OTLPTraceExporter } from '@opentelemetry/exporter-trace-otlp-http';
27
28 /* Packages for handling logs */
29 import { logs, SeverityNumber } from '@opentelemetry/api-logs';
30 import { BatchLogRecordProcessor, LoggerProvider } from '@opentelemetry/sdk-logs';
31 import { OTLPLogExporter } from '@opentelemetry/exporter-logs-otlp-http';
32
33 // These help with logging, diagnostics, and traces
34 import { diag, DiagConsoleLogger, DiagLogLevel } from '@opentelemetry/api';
35
36 // Defines a Resource to include metadata like service.name, required by Elastic
37 import { resourceFromAttributes, detectResources } from '@opentelemetry/resources';
38
39 // Experimental detector for browser environment
40 import { browserDetector } from '@opentelemetry/opentelemetry-browser-detector';
41
42 // Provides standard semantic keys for attributes, like service.name
43 import { ATTR_SERVICE_NAME } from '@opentelemetry/semantic-conventions';
44
45 // Context Propagation across signals
46 import {
47     CompositePropagator,
48     W3CBaggagePropagator,
49     W3CTraceContextPropagator
50 } from '@opentelemetry/core';
51
52 /* Custom dependencies */
53
54 import { LOGS_URL, TRACE_URL } from './constants';
55 import { WebVitalsInstrumentation } from './web-vitals.instrumentation';
56
57 // Enable OpenTelemetry debug logging to the console
58 diag.setLogger(new DiagConsoleLogger(), DiagLogLevel.DEBUG);
59
60 // Define resource metadata for the service, used by exporters (Elastic requires service.name)
61 const SERVICE_NAME = 'records-ui-web';
62
63 const detectedResources = detectResources({ detectors: [browserDetector] });
64 let resource = resourceFromAttributes({
65     [ATTR_SERVICE_NAME]: SERVICE_NAME,
66     'service.version': 1,
67     'deployment.environment': 'dev'
68 });
69 resource = resource.merge(detectedResources);
70
71 // Configure logging to send to the collector via nginx
72 const logExporter = new OTLPLogExporter({
73     url: LOGS_URL // nginx proxy
74 });
75
76 const loggerProvider = new LoggerProvider({
77     resource: resource,
78     processors: [new BatchLogRecordProcessor(logExporter)]
79 });
80
81 logs.setGlobalLoggerProvider(loggerProvider);
82
83 const logger = logs.getLogger('default', '1.0.0');
84 logger.emit({
85     severityNumber: SeverityNumber.INFO,
86     severityText: 'INFO',
87     body: 'Logger initialized'
88 });
89
90 // Configure the OTLP exporter to talk to the collector via nginx
91 const exporter = new OTLPTraceExporter({
92     url: TRACE_URL // nginx proxy
93 });
94
95 // Instantiate the trace provider and inject the resource
96 const provider = new WebTracerProvider({
97     resource: resource,
98     spanProcessors: [
99         // Send each completed span through the exporter
100         new SimpleSpanProcessor(exporter),
101         new SimpleSpanProcessor(new ConsoleSpanExporter())
102     ]
103 });
104
105 // Register the provider with propagation and set up the async context manager for spans
106 provider.register({
107     contextManager: new ZoneContextManager(),
108     propagator: new CompositePropagator({
109         propagators: [new W3CBaggagePropagator(), new W3CTraceContextPropagator()]
110     })
111 });
112
113 export class ClientTelemetry {
114     private static instance: ClientTelemetry;
115     private initialized = false;
116
117     private constructor() {}
118
119     // Obtain singleton instance of provider
120     public static getInstance(): ClientTelemetry {
121         if (!ClientTelemetry.instance) {
122             ClientTelemetry.instance = new ClientTelemetry();
123             ClientTelemetry.instance.start();
124         }
125         return ClientTelemetry.instance;
126     }
127
128     // Initializer
129     public start() {
130         if (!this.initialized) {
131             // Enable automatic span generation for document load and user click interactions
132             registerInstrumentations({
133                 instrumentations: [
134                     // Automatically tracks when the document loads
135                     new DocumentLoadInstrumentation(),
136                     getWebAutoInstrumentations({
137                         '@opentelemetry/instrumentation-fetch': {
138                             propagateTraceHeaderCorsUrls: /.*/,
139                             clearTimingResources: true
140                         }
141                     }),
142                     // User events
143                     new UserInteractionInstrumentation({
144                         eventNames: ['click'] // instrument click events only
145                     }),
146                     // Custom Web Vitals instrumentation
147                     new WebVitalsInstrumentation()
148                 ]
149             });
150
151             logger.emit({
152                 severityNumber: SeverityNumber.INFO,
153                 severityText: 'INFO',
154                 body: 'Client Telemetry Initialised'
155             });
156             this.initialized = true;
157         }
158     }
159 }
```

elastic

carlyrichmond.bsky.social

Type records-ui-web http

	0 s	50 s	100 s	150 s	200 s	250 s	300 s	350 s	402 s
10	Success **documentLoad** 171 ms								
	documentFetch 120 ms								
							resourceFetch 14 ms		
							resourceFetch 15 ms		
							resourceFetch 13 ms		
							resourceFetch 41 ms		
							resourceFetch 41 ms		
								resourceFetch 52 ms	
								resourceFetch 14 ms	



Trace sample

2 of 7

Investigate

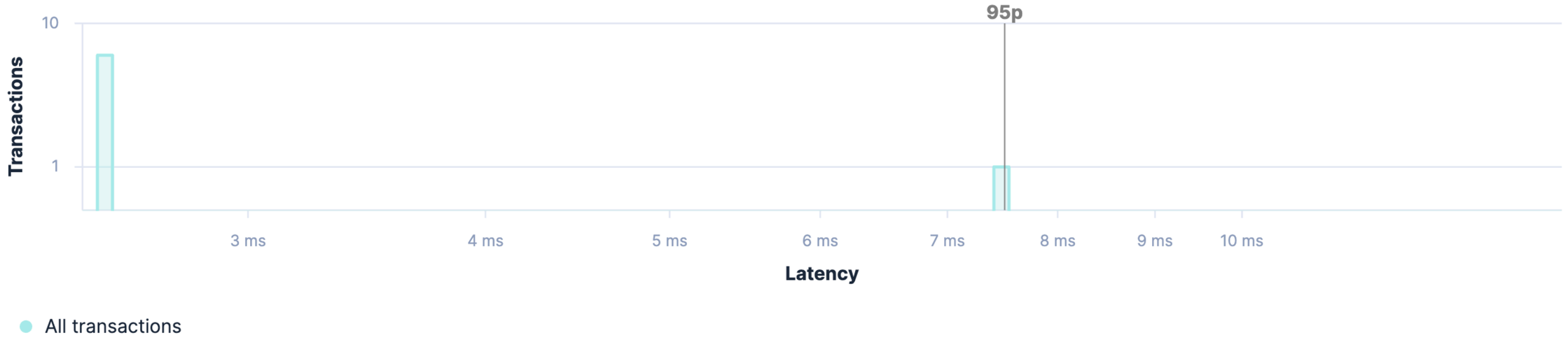
View full trace

3 minutes ago | 100 µs (100% of trace)

Timeline Metadata Logs

Show critical path

Type records-ui-web



Trace sample

2 of 7

Investigate

View full trace

4 minutes ago | 100 µs (100% of trace)

Timeline | Metadata | Logs

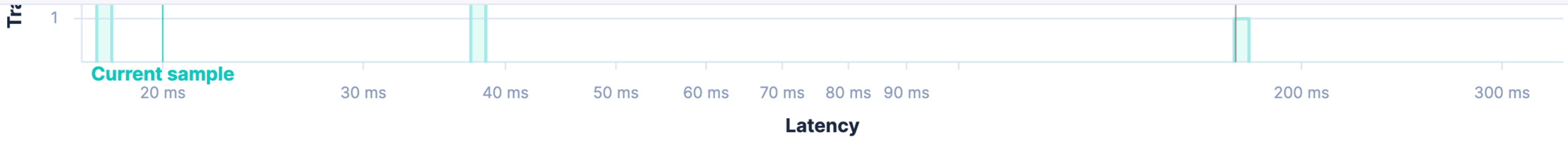
xp

How to add labels and other data

attributes

attributes.target_xpath	//html/body/div/section/div/div/input

target_xpath



All transactions

Trace sample

1 of 8

Investigate

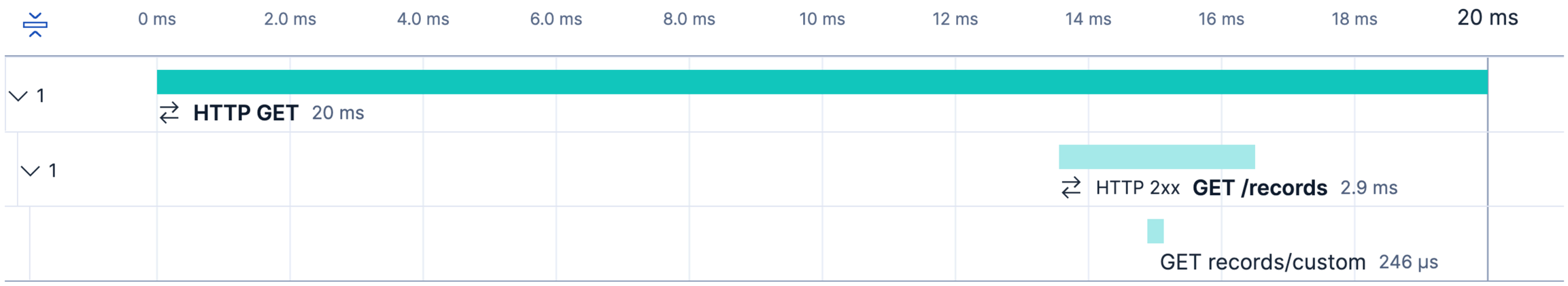
View full trace

5 minutes ago | 20 ms (100% of trace)

Timeline Metadata Logs

Show critical path

Services records-ui-web record-store-server-java



web-vitals.instrumentation.ts

```typescript
/* Web Vitals Frontend package
 example based on https://www.honeycomb.io/blog/core-web-vitals-opentelemetry */

import { onLCP, onINP, onCLS, type CLSMetric, type LCPMetric,
  type INPMetric } from 'web-vitals';

/* OpenTelemetry JS packages */

// Instrumentation base to create a custom Instrumentation for our provider
import { InstrumentationBase, type InstrumentationModuleDefinition } from
'@opentelemetry/instrumentation';

// Tracing API
import { trace, context, type Context } from '@opentelemetry/api';

// Time calculator via performance component
import { hrTime } from '@opentelemetry/core';

export class WebVitalsInstrumentation extends InstrumentationBase {

    constructor(config = {}) {
        super('WebVitalsInstrumentation', '1.0', config)
    }

    protected init(): InstrumentationModuleDefinition | InstrumentationModuleDefinition[] | void {}

    onReport(metric: LCPMetric | CLSMetric | INPMetric, parentSpanContext: Context | undefined) {
        const now = hrTime();

        // Start the span
        const webVitalsSpan = trace
            .getTracer('web-vitals-instrumentation')
            .startSpan(metric.name, { startTime: now }, parentSpanContext);

        // Add Core Web Vital attributes
        webVitalsSpan.setAttributes({
            [`web_vital.name`]: metric.name,
            [`web_vital.id`]: metric.id,
            [`web_vital.navigationType`]: metric.navigationType,
            [`web_vital.delta`]: metric.delta,
            [`web_vital.rating`]: metric.rating,
            [`web_vital.value`]: metric.value,
            // metric specific attributes
            [`web_vital.entries`]: JSON.stringify(metric.entries)
        });

        // End the span
        webVitalsSpan.end();
    }

    enable() {

        // Create parent span
        const parentSpan = trace.getTracer('web-vitals-instrumentation').startSpan('web-vitals');
        const ctx = trace.setSpan(context.active(), parentSpan);
        parentSpan.end();

        // Capture Interaction to Next Paint
        onINP((metric) => {
            this.onReport(metric, ctx);
        },
        { reportAllChanges: true });

        // Capture Cumulative Layout Shift
        onCLS((metric) => {
            this.onReport(metric, ctx);
        },
        { reportAllChanges: true });

        // Capture Largest Contentful Paint
        onLCP((metric) => {
            this.onReport(metric, ctx);
        },
        { reportAllChanges: true });
    }
}
```

```
18 export class WebVitalsInstrumentation extends InstrumentationBase {
19
20     constructor(config = {}) {
21         super('WebVitalsInstrumentation', '1.0', config)
22     }
23
24     protected init(): InstrumentationModuleDefinition | InstrumentationModuleDefinition[] | void {}
25
26     onReport(metric: LCPMetric | CLSMetric | INPMetric, parentSpanContext: Context | undefined) {
27         const now = hrTime();
28
29         // Start the span
30         const webVitalsSpan = trace
31             .getTracer('web-vitals-instrumentation')
32             .startSpan(metric.name, { startTime: now }, parentSpanContext);
33
34         // Add Core Web Vital attributes
35         webVitalsSpan.setAttributes({
36             [`web_vital.name`]: metric.name,
37             [`web_vital.id`]: metric.id,
38             [`web_vital.navigationType`]: metric.navigationType,
39             [`web_vital.delta`]: metric.delta,
40             [`web_vital.rating`]: metric.rating,
41             [`web_vital.value`]: metric.value,
42             // metric specific attributes
43             [`web_vital.entries`]: JSON.stringify(metric.entries)
44         });
```

```
47          webVitalsSpan.end();
48      }
49
50      enable() {
51
52          // Create parent span
53          const parentSpan = trace.getTracer('web-vitals-instrumentation').startSpan('web-vitals');
54          const ctx = trace.setSpan(context.active(), parentSpan);
55          parentSpan.end();
56
57          // Capture Interaction to Next Paint
58          onINP((metric) => {
59              this.onReport(metric, ctx);
60          },
61          { reportAllChanges: true });
62
63          // Capture Cumulative Layout Shift
64          onCLS((metric) => {
65              this.onReport(metric, ctx);
66          },
67          { reportAllChanges: true });
68
69          // Capture Largest Contentful Paint
70          onLCP((metric) => {
71              this.onReport(metric, ctx);
72          },
73          { reportAllChanges: true });
```

web-vitals.instrumentation.ts

```typescript
/* Web Vitals Frontend package
 example based on https://www.honeycomb.io/blog/core-web-vitals-opentelemetry */

import { onLCP, onINP, onCLS, type CLSMetric, type LCPMetric,
  type INPMetric } from 'web-vitals';

/* OpenTelemetry JS packages */

// Instrumentation base to create a custom Instrumentation for our provider
import { InstrumentationBase, type InstrumentationModuleDefinition } from
'@opentelemetry/instrumentation';

// Tracing API
import { trace, context, type Context } from '@opentelemetry/api';

// Time calculator via performance component
import { hrTime } from '@opentelemetry/core';

export class WebVitalsInstrumentation extends InstrumentationBase {

    constructor(config = {}) {
        super('WebVitalsInstrumentation', '1.0', config)
    }

    protected init(): InstrumentationModuleDefinition | InstrumentationModuleDefinition[] | void {}

    onReport(metric: LCPMetric | CLSMetric | INPMetric, parentSpanContext: Context | undefined) {
        const now = hrTime();

        // Start the span
        const webVitalsSpan = trace
            .getTracer('web-vitals-instrumentation')
            .startSpan(metric.name, { startTime: now }, parentSpanContext);

        // Add Core Web Vital attributes
        webVitalsSpan.setAttributes({
            [`web_vital.name`]: metric.name,
            [`web_vital.id`]: metric.id,
            [`web_vital.navigationType`]: metric.navigationType,
            [`web_vital.delta`]: metric.delta,
            [`web_vital.rating`]: metric.rating,
            [`web_vital.value`]: metric.value,
            // metric specific attributes
            [`web_vital.entries`]: JSON.stringify(metric.entries)
        });

        // End the span
        webVitalsSpan.end();
    }

    enable() {

        // Create parent span
        const parentSpan = trace.getTracer('web-vitals-instrumentation').startSpan('web-vitals');
        const ctx = trace.setSpan(context.active(), parentSpan);
        parentSpan.end();

        // Capture Interaction to Next Paint
        onINP((metric) => {
            this.onReport(metric, ctx);
        },
        { reportAllChanges: true });

        // Capture Cumulative Layout Shift
        onCLS((metric) => {
            this.onReport(metric, ctx);
        },
        { reportAllChanges: true });

        // Capture Largest Contentful Paint
        onLCP((metric) => {
            this.onReport(metric, ctx);
        },
        { reportAllChanges: true });
    }
}
```

OTel Core Web Vitals Summary

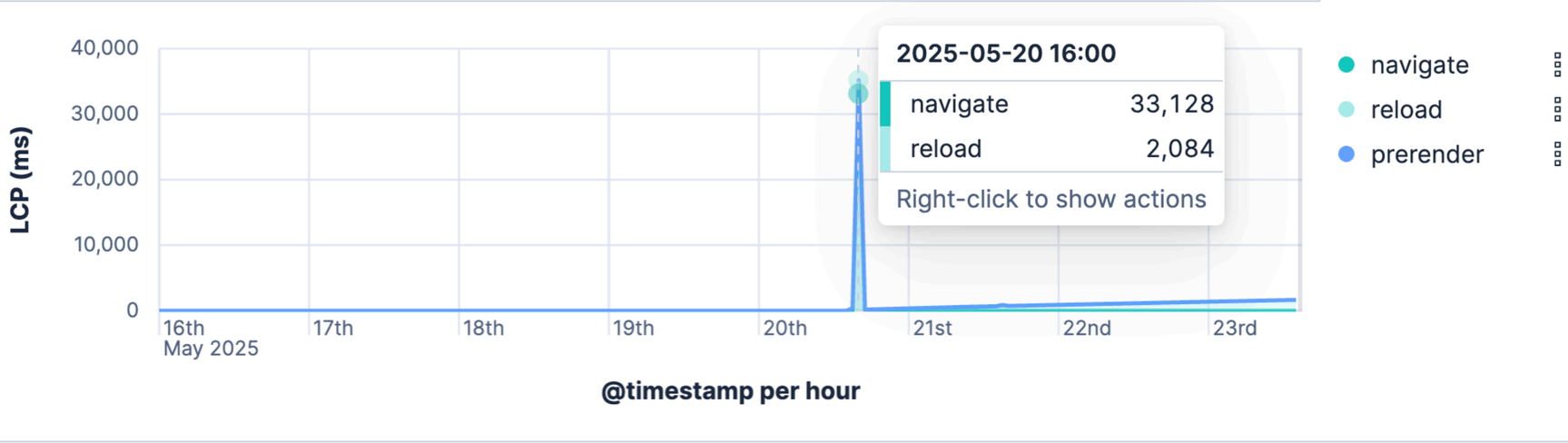
This dashboard shows the results of Core Web Vitals captured from the OTel record store application using the web-vitals JS library and OpenTelemetry JavaScript SDK.

Full instrumentation code available here.

Largest Contentful Paint (LCP)

LCP rating good

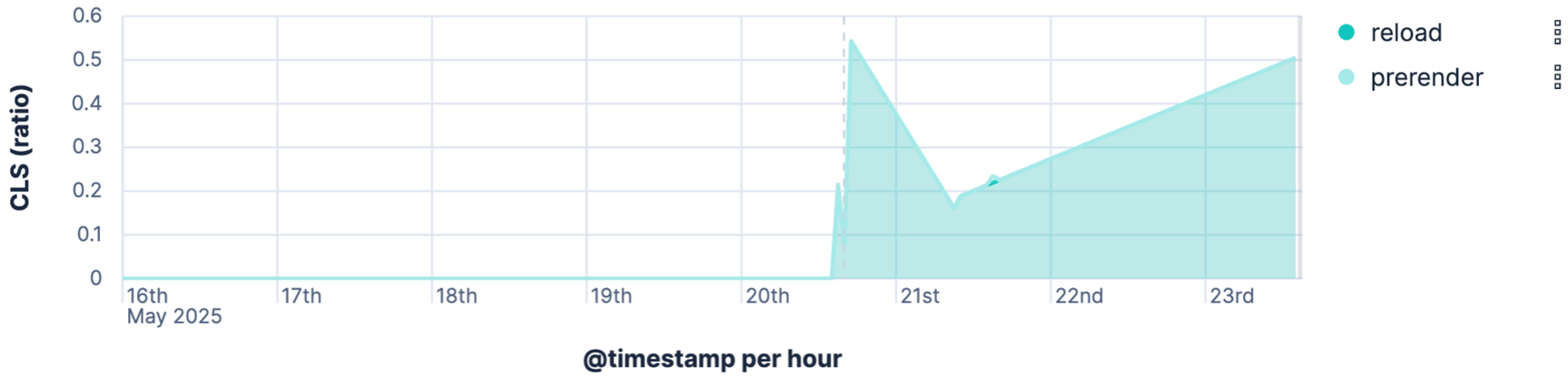
1,660



Cumulative Layout Shift (CLS)

CLS rating poor

0.506



Exit full screen

Source: Peter Muniz on Unsplash

Source: James Lewis on Unsplash

AUTOMATIC VERSUS MANUAL INSTRUMENTATION

OpenTelemetry

Dockerfile

```dockerfile
FROM eclipse-temurin:21-jdk-jammy as deps

# Other build steps omitted

# Java OTel ENV
ENV OTEL_RESOURCE_ATTRIBUTES=service.version=1,deployment.environment=dev
ENV OTEL_SERVICE_NAME=record-store-server-java
ENV OTEL_EXPORTER_OTLP_ENDPOINT=$ELASTIC_ENDPOINT
ENV OTEL_EXPORTER_OTLP_HEADERS="Authorization=ApiKey ${ELASTIC_API_KEY}"
ENV OTEL_TRACES_EXPORTER=otlp
ENV OTEL_METRICS_EXPORTER=otlp
ENV OTEL_LOGS_EXPORTER=otlp

COPY src/main/resources/otel/opentelemetry-javaagent.jar ./
ENV JAVA_TOOL_OPTIONS="-javaagent:./opentelemetry-javaagent.jar"

# Copy the executable from the "package" stage.
COPY --from=extract build/target/extracted/dependencies/ ./
COPY --from=extract build/target/extracted/spring-boot-loader/ ./
COPY --from=extract build/target/extracted/snapshot-dependencies/ ./
COPY --from=extract build/target/extracted/application/ ./

EXPOSE 8080
ENTRYPOINT [ "java", "org.springframework.boot.loader.launch.JarLauncher" ]
```

RecordController.java

```java
package com.rum.records.store.server.rumrecordsstore.record;

import io.opentelemetry.api.OpenTelemetry;
import io.opentelemetry.api.trace.SpanKind;
import io.opentelemetry.api.trace.Tracer;
import io.opentelemetry.api.trace.Span;
import io.opentelemetry.context.Scope;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import reactor.core.publisher.Flux;

@RestController
@CrossOrigin(origins = {"http://localhost:4205", "http://localhost:3000"})
@RequestMapping("/records")
public class RecordController {

  private static Logger logger = LogManager.getLogger(RecordController.class);
  private final Tracer tracer;
  private final String SERVICE_NAME = System.getenv("OTEL_SERVICE_NAME");
  private final RecordRepository recordRepository;

  public RecordController(OpenTelemetry openTelemetry, RecordRepository recordRepository) {
    this.recordRepository = recordRepository;
    this.tracer = openTelemetry.getTracer(SERVICE_NAME, "1.0.0");
  }

  @GetMapping
  private Flux<MusicRecord> getAllRecords() throws InterruptedException {
    Span span = tracer.spanBuilder("GET records/custom").setSpanKind(SpanKind.SERVER).startSpan();
    span.setAttribute("http.method", "GET");
    span.setAttribute("http.url", "/records/custom");
    span.setAttribute("http.response.status_code", "200");


    try (Scope scope = span.makeCurrent()) {
      logger.info("Waiting...");
      Thread.sleep(10000);

      logger.info("Getting all records");
      return recordRepository.findAllRecords();
    } catch(Throwable t) {
      span.recordException(t);
      throw t;
    } finally {
      span.end();
    }
  }

  @GetMapping("/{query}")
  private Flux<MusicRecord> getRecordByQuery(@PathVariable String query) throws
InterruptedException {
    logger.info("Waiting...");
    Thread.sleep(10000);
    logger.info(String.format("Getting records that match query: %s", query));
    return recordRepository.findRecordsByQuery(query);
  }
}
```

elastic

carlyrichmond.bsky.social

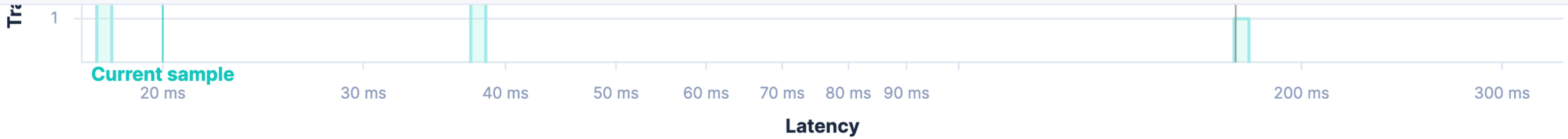
```
28
29  public RecordController(OpenTelemetry openTelemetry, RecordRepository recordRepository) {
30    this.recordRepository = recordRepository;
31    this.tracer = openTelemetry.getTracer(SERVICE_NAME, "1.0.0");
32  }
33
34  @GetMapping
35  private Flux<MusicRecord> getAllRecords() throws InterruptedException {
36    Span span = tracer.spanBuilder("GET records/custom").setSpanKind(SpanKind.SERVER).startSpan();
37    span.setAttribute("http.method", "GET");
38    span.setAttribute("http.url", "/records/custom");
39    span.setAttribute("http.response.status_code", "200");
40
41
42    try (Scope scope = span.makeCurrent()) {
43      logger.info("Waiting...");
44      Thread.sleep(10000);
45
46      logger.info("Getting all records");
47      return recordRepository.findAllRecords();
48    } catch(Throwable t) {
49      span.recordException(t);
50      throw t;
51    } finally {
52      span.end();
53    }
54  }
```

RecordController.java

```java
package com.rum.records.store.server.rumrecordsstore.record;

import io.opentelemetry.api.OpenTelemetry;
import io.opentelemetry.api.trace.SpanKind;
import io.opentelemetry.api.trace.Tracer;
import io.opentelemetry.api.trace.Span;
import io.opentelemetry.context.Scope;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import reactor.core.publisher.Flux;

@RestController
@CrossOrigin(origins = {"http://localhost:4205", "http://localhost:3000"})
@RequestMapping("/records")
public class RecordController {

  private static Logger logger = LogManager.getLogger(RecordController.class);
  private final Tracer tracer;
  private final String SERVICE_NAME = System.getenv("OTEL_SERVICE_NAME");
  private final RecordRepository recordRepository;

  public RecordController(OpenTelemetry openTelemetry, RecordRepository recordRepository) {
    this.recordRepository = recordRepository;
    this.tracer = openTelemetry.getTracer(SERVICE_NAME, "1.0.0");
  }

  @GetMapping
  private Flux<MusicRecord> getAllRecords() throws InterruptedException {
    Span span = tracer.spanBuilder("GET records/custom").setSpanKind(SpanKind.SERVER).startSpan();
    span.setAttribute("http.method", "GET");
    span.setAttribute("http.url", "/records/custom");
    span.setAttribute("http.response.status_code", "200");


    try (Scope scope = span.makeCurrent()) {
      logger.info("Waiting...");
      Thread.sleep(10000);

      logger.info("Getting all records");
      return recordRepository.findAllRecords();
    } catch(Throwable t) {
      span.recordException(t);
      throw t;
    } finally {
      span.end();
    }
  }

  @GetMapping("/{query}")
  private Flux<MusicRecord> getRecordByQuery(@PathVariable String query) throws
InterruptedException {
    logger.info("Waiting...");
    Thread.sleep(10000);
    logger.info(String.format("Getting records that match query: %s", query));
    return recordRepository.findRecordsByQuery(query);
  }
}
```

elastic

carlyrichmond.bsky.social



All transactions

Trace sample

1 of 8

Investigate

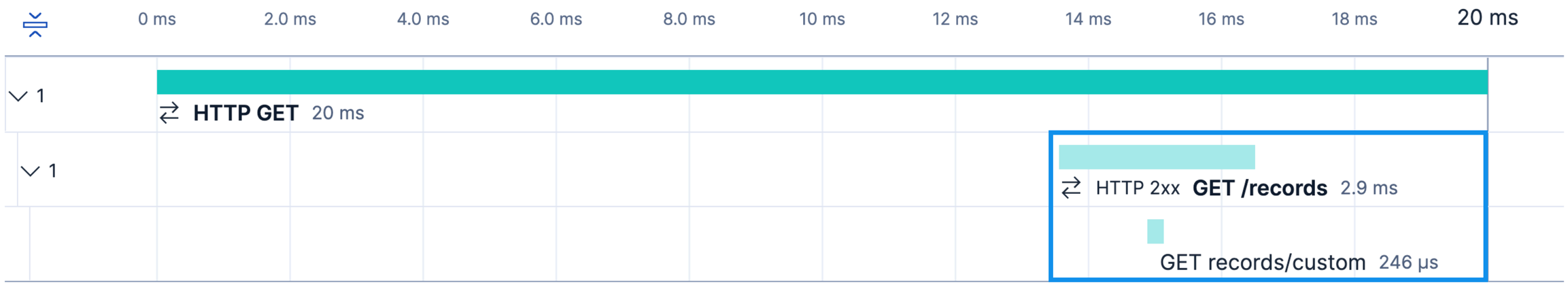
View full trace

5 minutes ago | 20 ms (100% of trace)

Timeline Metadata Logs

Show critical path

Services records-ui-web record-store-server-java



Current samp
20 ms

All transactions

Trace sample

9 minutes ago | 20 m

Timeline Metadat

Show critical

Services records-

0 ms

1

HT

1

Transaction details

Investigate

Service
record-store-server-java

Transaction
GET /records

9 minutes ago | 2.9 ms (15% of trace) | GET http://localhost:8080/records 200 OK

Metadata

env

How to add labels and other data

deployment

deployment.environment	dev

process

process.runtime.name	OpenJDK Runtime Environment

resource

resource.attributes.deployment.environment	dev

ManualInstrumentation.java

```java
package otel;

import io.opentelemetry.api.OpenTelemetry;
import io.opentelemetry.api.trace.propagation.W3CTraceContextPropagator;
import io.opentelemetry.context.propagation.ContextPropagators;
import io.opentelemetry.exporter.otlp.logs.OtlpGrpcLogRecordExporter;
import io.opentelemetry.exporter.otlp.metrics.OtlpGrpcMetricExporter;
import io.opentelemetry.exporter.otlp.trace.OtlpGrpcSpanExporter;
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.logs.SdkLoggerProvider;
import io.opentelemetry.sdk.logs.export.BatchLogRecordProcessor;
import io.opentelemetry.sdk.metrics.SdkMeterProvider;
import io.opentelemetry.sdk.metrics.export.PeriodicMetricReader;
import io.opentelemetry.sdk.resources.Resource;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;
import io.opentelemetry.semconv.ServiceAttributes;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class RecordsServiceApplication {
  public static void main(String[] args) {
    SpringApplication.run(RecordsServiceApplication.class, args);
  }

  @Bean
  public OpenTelemetry openTelemetry() {
    Resource resource =
        Resource.getDefault().toBuilder()
            .put(ServiceAttributes.SERVICE_NAME, "record-store-server-java")
            .put(ServiceAttributes.SERVICE_VERSION, "0.1.0")
            .build();

    SdkTracerProvider sdkTracerProvider =
        SdkTracerProvider.builder()
            .addSpanProcessor(
                BatchSpanProcessor.builder(OtlpGrpcSpanExporter.builder().build()).build())
            .setResource(resource)
            .build();

    SdkMeterProvider sdkMeterProvider =
        SdkMeterProvider.builder()
            .registerMetricReader(
                PeriodicMetricReader.builder(OtlpGrpcMetricExporter.builder().build()).build())
            .setResource(resource)
            .build();

    SdkLoggerProvider sdkLoggerProvider =
        SdkLoggerProvider.builder()
            .addLogRecordProcessor(
                BatchLogRecordProcessor.builder(OtlpGrpcLogRecordExporter.builder().build())
                    .build())
            .setResource(resource)
            .build();

    OpenTelemetry openTelemetry =
        OpenTelemetrySdk.builder()
            .setTracerProvider(sdkTracerProvider)
            .setMeterProvider(sdkMeterProvider)
            .setLoggerProvider(sdkLoggerProvider)
            .setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInstance()))
            .buildAndRegisterGlobal();

    return openTelemetry;
  }
}
```

ManualInstrumentation.java

```java
package otel;

import io.opentelemetry.api.OpenTelemetry;
import io.opentelemetry.api.trace.propagation.W3CTraceContextPropagator;
import io.opentelemetry.context.propagation.ContextPropagators;
import io.opentelemetry.exporter.otlp.logs.OtlpGrpcLogRecordExporter;
import io.opentelemetry.exporter.otlp.metrics.OtlpGrpcMetricExporter;
import io.opentelemetry.exporter.otlp.trace.OtlpGrpcSpanExporter;
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.logs.SdkLoggerProvider;
import io.opentelemetry.sdk.logs.export.BatchLogRecordProcessor;
import io.opentelemetry.sdk.metrics.SdkMeterProvider;
import io.opentelemetry.sdk.metrics.export.PeriodicMetricReader;
import io.opentelemetry.sdk.resources.Resource;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;
import io.opentelemetry.semconv.ServiceAttributes;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class RecordsServiceApplication {
  public static void main(String[] args) {
    SpringApplication.run(RecordsServiceApplication.class, args);
  }

  @Bean
  public OpenTelemetry openTelemetry() {
    Resource resource =
        Resource.getDefault().toBuilder()
            .put(ServiceAttributes.SERVICE_NAME, "record-store-server-java")
            .put(ServiceAttributes.SERVICE_VERSION, "0.1.0")
            .build();

```

```
32        Resource.getDefault().toBuilder()
33            .put(ServiceAttributes.SERVICE_NAME, "record-store-server-java")
34            .put(ServiceAttributes.SERVICE_VERSION, "0.1.0")
35            .build();
36
37    SdkTracerProvider sdkTracerProvider =
38        SdkTracerProvider.builder()
39            .addSpanProcessor(
40                BatchSpanProcessor.builder(OtlpGrpcSpanExporter.builder().build()).build())
41            .setResource(resource)
42            .build();
43
44    SdkMeterProvider sdkMeterProvider =
45        SdkMeterProvider.builder()
46            .registerMetricReader(
47                PeriodicMetricReader.builder(OtlpGrpcMetricExporter.builder().build()).build())
48            .setResource(resource)
49            .build();
50
51    SdkLoggerProvider sdkLoggerProvider =
52        SdkLoggerProvider.builder()
53            .addLogRecordProcessor(
54                BatchLogRecordProcessor.builder(OtlpGrpcLogRecordExporter.builder().build())
55                    .build())
56            .setResource(resource)
57            .build();
58
59    OpenTelemetry openTelemetry =
60        OpenTelemetrySdk.builder()
61            .setTracerProvider(sdkTracerProvider)
62            .setMeterProvider(sdkMeterProvider)
63            .setLoggerProvider(sdkLoggerProvider)
64            .setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInstance()))
65            .buildAndRegisterGlobal();
66
67    return openTelemetry;
68  }
69 }
```

ManualInstrumentation.java

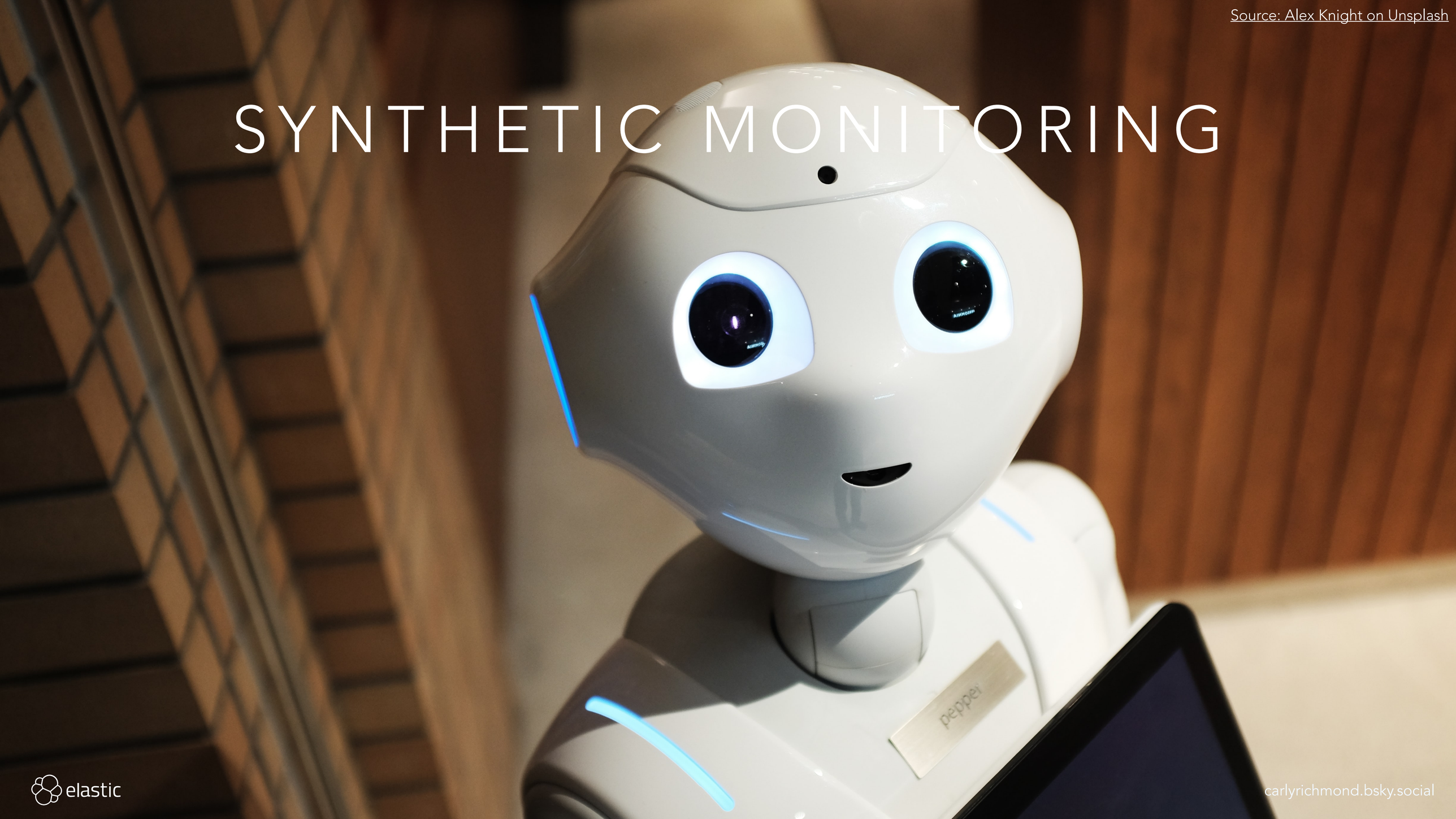
```java
package otel;

import io.opentelemetry.api.OpenTelemetry;
import io.opentelemetry.api.trace.propagation.W3CTraceContextPropagator;
import io.opentelemetry.context.propagation.ContextPropagators;
import io.opentelemetry.exporter.otlp.logs.OtlpGrpcLogRecordExporter;
import io.opentelemetry.exporter.otlp.metrics.OtlpGrpcMetricExporter;
import io.opentelemetry.exporter.otlp.trace.OtlpGrpcSpanExporter;
import io.opentelemetry.sdk.OpenTelemetrySdk;
import io.opentelemetry.sdk.logs.SdkLoggerProvider;
import io.opentelemetry.sdk.logs.export.BatchLogRecordProcessor;
import io.opentelemetry.sdk.metrics.SdkMeterProvider;
import io.opentelemetry.sdk.metrics.export.PeriodicMetricReader;
import io.opentelemetry.sdk.resources.Resource;
import io.opentelemetry.sdk.trace.SdkTracerProvider;
import io.opentelemetry.sdk.trace.export.BatchSpanProcessor;
import io.opentelemetry.semconv.ServiceAttributes;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class RecordsServiceApplication {
  public static void main(String[] args) {
    SpringApplication.run(RecordsServiceApplication.class, args);
  }

  @Bean
  public OpenTelemetry openTelemetry() {
    Resource resource =
        Resource.getDefault().toBuilder()
            .put(ServiceAttributes.SERVICE_NAME, "record-store-server-java")
            .put(ServiceAttributes.SERVICE_VERSION, "0.1.0")
            .build();

    SdkTracerProvider sdkTracerProvider =
        SdkTracerProvider.builder()
            .addSpanProcessor(
                BatchSpanProcessor.builder(OtlpGrpcSpanExporter.builder().build()).build())
            .setResource(resource)
            .build();

    SdkMeterProvider sdkMeterProvider =
        SdkMeterProvider.builder()
            .registerMetricReader(
                PeriodicMetricReader.builder(OtlpGrpcMetricExporter.builder().build()).build())
            .setResource(resource)
            .build();

    SdkLoggerProvider sdkLoggerProvider =
        SdkLoggerProvider.builder()
            .addLogRecordProcessor(
                BatchLogRecordProcessor.builder(OtlpGrpcLogRecordExporter.builder().build())
                    .build())
            .setResource(resource)
            .build();

    OpenTelemetry openTelemetry =
        OpenTelemetrySdk.builder()
            .setTracerProvider(sdkTracerProvider)
            .setMeterProvider(sdkMeterProvider)
            .setLoggerProvider(sdkLoggerProvider)
            .setPropagators(ContextPropagators.create(W3CTraceContextPropagator.getInstance()))
            .buildAndRegisterGlobal();

    return openTelemetry;
  }
}
```

"**Holistic**: relating to the whole of something or to the total system instead of just to its parts"

HOLISTIC, CAMBRIDGE ENGLISH DICTIONARY

Source: Alex Knight on Unsplash

SYNTHETIC MONITORING



elastic

carlyrichmond.bsky.social

heartbeat.yml

```yaml
heartbeat.monitors:
- type: http
  name: Replicator HTTP ping
  id: synthetics-replicator-monitor-http
  enabled: true
  urls: "https://synthetics-replicator.netlify.app/"
  schedule: '@every 3m'
  timeout: 16s
```

Monitors

Create Monitor Refresh

Overview Management

Search by name, URL, host, tag, project or locatior

Up Down Disabled

Type 2 Location 1 Tags 0 Frequency 3 Project 1

Monitors status

3 Up

1 Down

1 Disabled

Last 6 hours

1 Errors

Last 12 hours

1 Alerts

Showing **5 Monitors**

Spaces Default Add to dashboard Sort by Status Group by None

Replicator Order Journey

Europe - United Kingdom

Journey

Duration

690 ms

Recorded Order journey

Europe - United Kingdom

Journey

Duration

1 s

Replicator HTTP ping

Europe - United Kingdom

HTTP

Duration

58 ms

Replicator Login Journey

Europe - United Kingdom

Journey

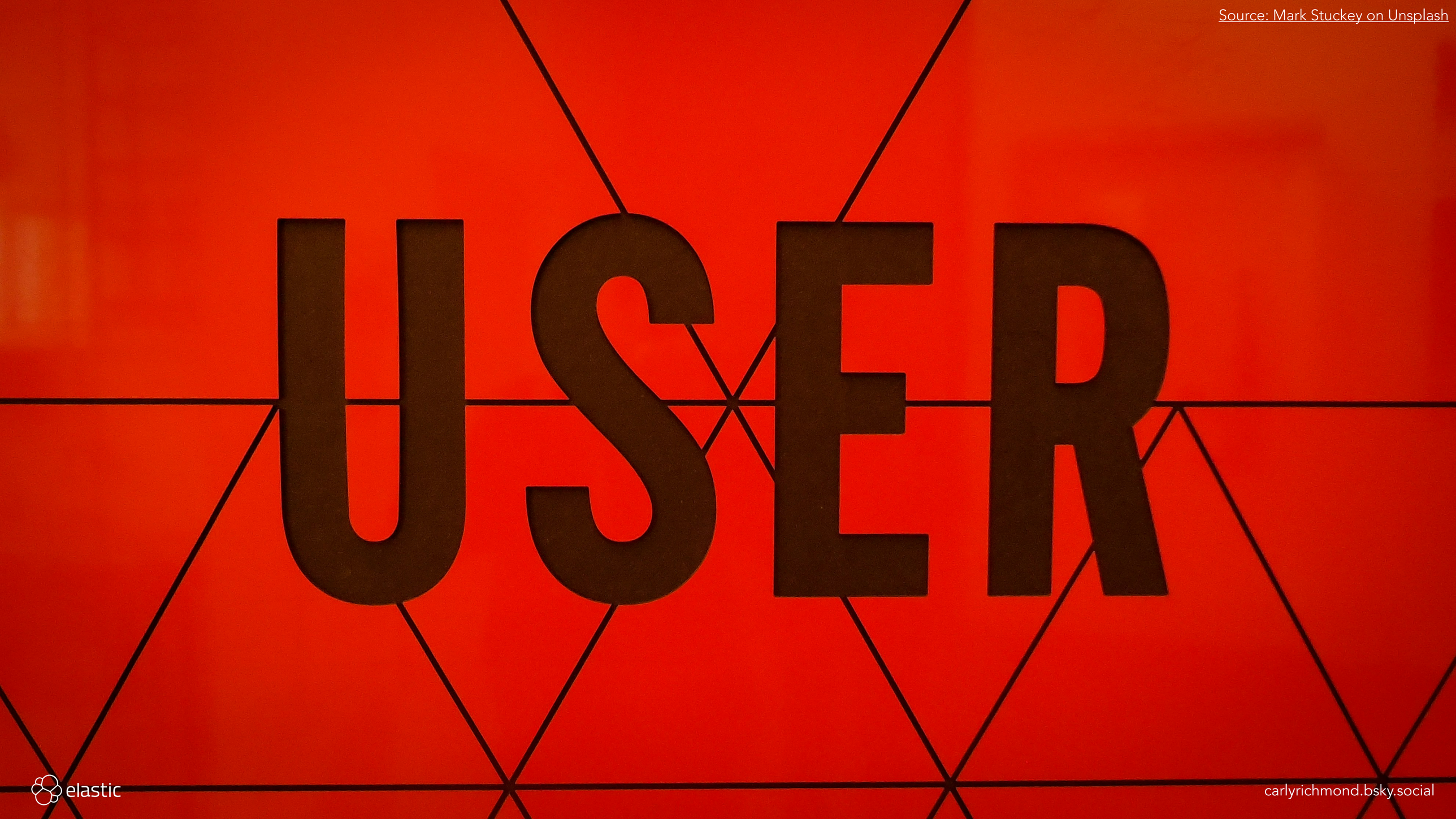
Duration

1 s

Local Replicator Order Journey

"**Holistic**: relating to the whole of something or to the total system instead of just to its **parts**"

HOLISTIC, CAMBRIDGE ENGLISH DICTIONARY



Source: Mark Stuckey on Unsplash

Source: Lucas Alexander on Unsplash

Source: AltumCode on Unsplash

MONITORING AS CODE

elastic

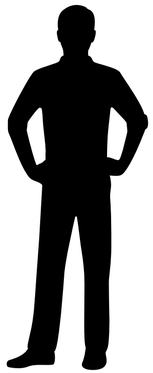
carlyrichmond.bsky.social

"**Holistic**: relating to the whole of something or to the **total system** instead of just to its parts"

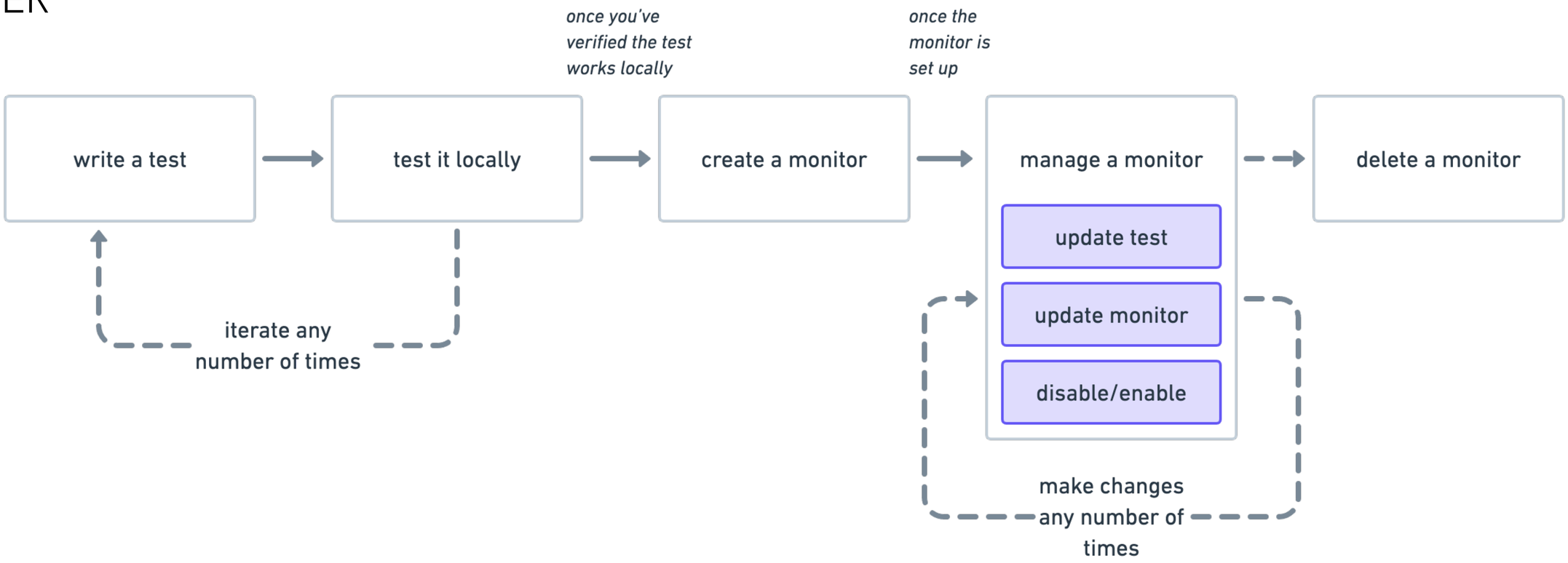
HOLISTIC, CAMBRIDGE ENGLISH DICTIONARY

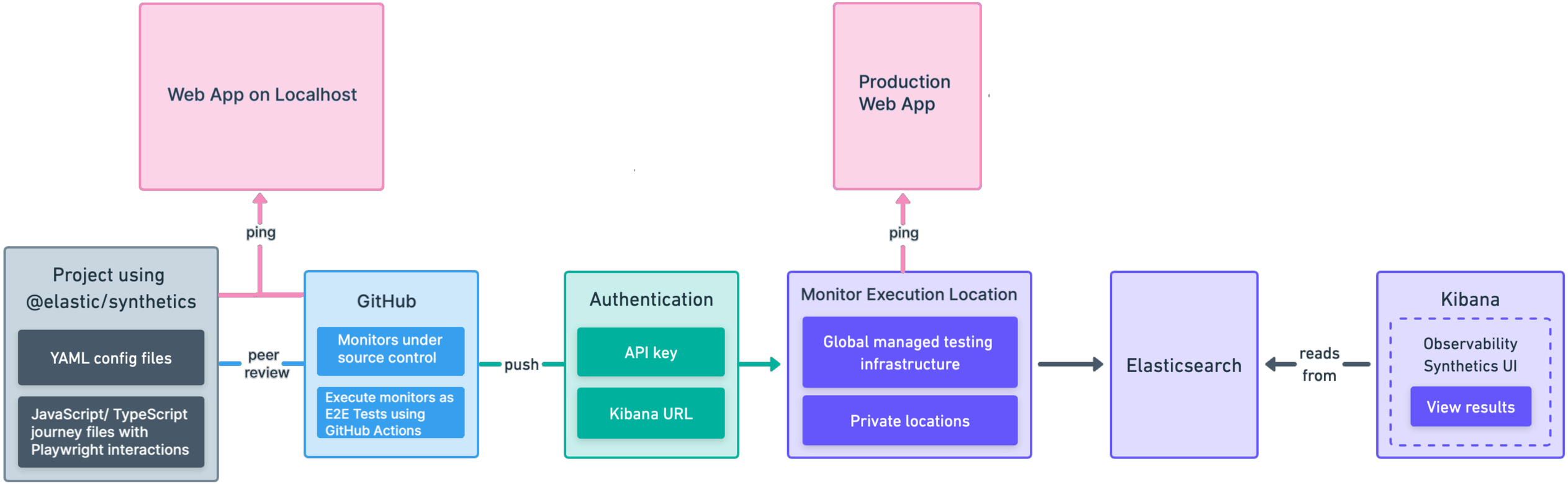
SCAN ME
carlyrichmond.bsky.social

DEVELOPER



SUPPORT/ SRE




Web App on Localhost
Production Web App
ping
ping
Project using @elastic/synthetics
YAML config files
JavaScript/ TypeScript journey files with Playwright interactions
peer review
GitHub
Monitors under source control
Execute monitors as E2E Tests using GitHub Actions
push
Authentication
API key
Kibana URL
Monitor Execution Location
Global managed testing infrastructure
Private locations
Elasticsearch
reads from
Kibana
Observability Synthetics UI
View results

Svelte + TS + Elastic Synthet

Synthetics Overview | Synthe

synthetics-replicator.netlify.app

REPLICATR

Login

To seek out new food and new cuisines.

To boldly eat what no one has eaten before!

Order Now!

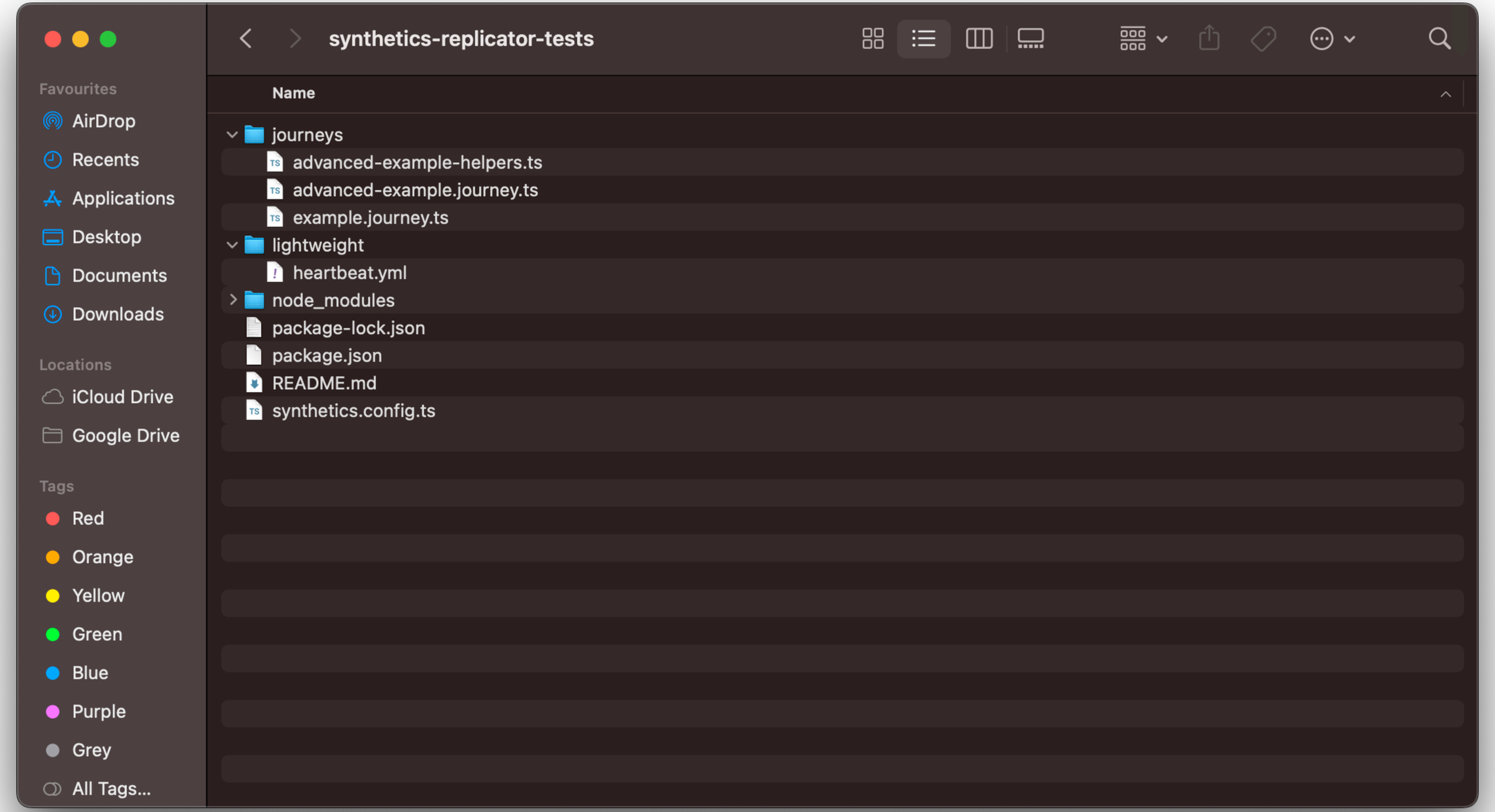
synthetics-replicator - -zsh

```
> npm install -g @elastic/synthetics
> npx @elastic/synthetics -h
> npx @elastic/synthetics init synthetics-replicator-test
```

Source: Andres Siimon on Unsplash

elastic

carlyrichmond.bsky.social


synthetics-replicator-tests
Favourites
AirDrop
Recents
Applications
Desktop
Documents
Downloads
Locations
iCloud Drive
Google Drive
Tags
Red
Orange
Yellow
Green
Blue
Purple
Grey
All Tags...
Name
journeys
advanced-example-helpers.ts
advanced-example.journey.ts
example.journey.ts
lightweight
heartbeat.yml
node_modules
package-lock.json
package.json
README.md
synthetics.config.ts

synthetics.config.ts

```typescript
import { SyntheticsConfig } from '@elastic/synthetics';
import { devices } from "playwright-chromium";

export default env => {
  const config: SyntheticsConfig = {
    params: {
      url: 'http://localhost:5173',
      username: process.env.REPLICATR_USERNAME,
      password: process.env.REPLICATR_PASSWORD,
      isLocal: env !== 'production'
    },
    playwrightOptions: {
      ignoreHTTPSErrors: false,
      ...devices['Desktop Chrome']
    },
    /**
     * Configure global monitor settings
     */
    monitor: {
      schedule: 10,
      locations: ['united_kingdom'],
      privateLocations: []
    },
    /**
     * Project monitors settings
     */
    project: {
      id: 'synthetics-replicator-tests',
      url: process.env.ELASTIC_DEPLOYMENT || 'http://localhost:443',
      space: 'default',
    },
  };
  if (env === 'production') {
    config.params!.url = 'https://synthetics-replicator.netlify.app/';
  }
  return config;
};
```

login.journey.ts

```typescript
import { journey, step, monitor, expect, before } from '@elastic/synthetics';

journey('Replicator Login Journey', ({ page, params }) => {
  // Only relevant for the push command to create
  // monitors in Kibana
  monitor.use({
    id: 'synthetics-replicator-monitor-login',
    schedule: 30,
    enabled: !params.isLocal
  });

  before(async () => {
    await page.goto(params.url);
  });

  step('assert home page loads', async () => {
    const header = await page.locator('h1');
    expect(await header.textContent()).toBe('Replicatr');
  });

  step('assert go to login page', async () => {
    const loginButton = await page.getByTestId('account-button');
    expect(loginButton).toHaveText('Login');
    await loginButton.click();

    const url = page.url();
    expect(url).toContain('/login');

    const loginForm = await page.getByTestId('login-form');
    expect(loginForm).toBeDefined();
  });

  step('assert manual login successful', async () => {
    const loginForm = await page.getByTestId('login-form');
    expect(loginForm).toBeDefined();

    // Check submit button is disabled before entry
    const submitButton = await page.getByTestId('login-button');
    expect(submitButton).toBeDisabled();

    // Add credentials
    const username = params.username;
    const usernameInput = page.getByTestId('username-input');
    usernameInput.fill(username);
    await expect(usernameInput).toHaveValue(username);

    const password = params.password;
    const passwordInput = page.getByTestId('password-input');
    passwordInput.fill(password);
    await expect(passwordInput).toHaveValue(password);

    // Check login button is now enabled
    expect(submitButton).toBeEnabled();
    await submitButton.click();

    // Wait for login
    await page.waitForURL('**/order');

    // Validate we have logged in successfully
    const loginButton = await page.getByTestId('account-button');
    expect(loginButton).toContainText('Welcome');

    // Ensure we have navigated to menu
    const menuTiles = await page.getByTestId('menu-item-card');
    expect(await menuTiles.count()).toBeGreaterThan(0);
  });

  step('assert unable to go to login page when signed in', async () => {
    const loginButton = await page.getByTestId('account-button');
    expect(loginButton).toContainText('Welcome');
    await loginButton.click();

    const url = page.url();
    expect(url).toContain('/order');
  });
});
```

login.journey.ts

```
1  import { journey, step, monitor, expect, before } from '@elastic/synthetics';
2
3  journey('Replicator Login Journey', ({ page, params }) => {
4    // Only relevant for the push command to create
5    // monitors in Kibana
6    monitor.use({
7      id: 'synthetics-replicator-monitor-login',
8      schedule: 30,
9      enabled: !params.isLocal
10   });
11
12   before(async () => {
13     await page.goto(params.url);
14   });
15
16   step('assert home page loads', async () => {
17     const header = await page.locator('h1');
18     expect(await header.textContent()).toBe('Replicatr');
19   });
20
21   step('assert go to login page', async () => {
```

```
21  step('assert go to login page', async () => {
22    const loginButton = await page.getByTestId('account-button');
23    expect(loginButton).toHaveText('Login');
24    await loginButton.click();
25
26    const url = page.url();
27    expect(url).toContain('/login');
28
29    const loginForm = await page.getByTestId('login-form');
30    expect(loginForm).toBeDefined();
31  });
32
33  step('assert manual login successful', async () => {
34    const loginForm = await page.getByTestId('login-form');
35    expect(loginForm).toBeDefined();
36
37    // Check submit button is disabled before entry
38    const submitButton = await page.getByTestId('login-button');
39    expect(submitButton).toBeDisabled();
40
41    // Add credentials
42    const username = params.username;
43    const usernameInput = page.getByTestId('username-input');
44    usernameInput.fill(username);
```

```
47      const password = params.password;
48      const passwordInput = page.getByTestId('password-input');
49      passwordInput.fill(password);
50      await expect(passwordInput).toHaveValue(password);
51
52      // Check login button is now enabled
53      expect(submitButton).toBeEnabled();
54      await submitButton.click();
55
56      // Wait for login
57      await page.waitForURL('**/order');
58
59      // Validate we have logged in successfully
60      const loginButton = await page.getByTestId('account-button');
61      expect(loginButton).toContainText('Welcome');
62
63      // Ensure we have navigated to menu
64      const menuTiles = await page.getByTestId('menu-item-card');
65      expect(await menuTiles.count()).toBeGreaterThan(0);
66    });
67
68    step('assert unable to go to login page when signed in', async () => {
69      const loginButton = await page.getByTestId('account-button');
70      expect(loginButton).toContainText('Welcome');
```

```
54      await submitButton.click();
55
56      // Wait for login
57      await page.waitForURL('**/order');
58
59      // Validate we have logged in successfully
60      const loginButton = await page.getByTestId('account-button');
61      expect(loginButton).toContainText('Welcome');
62
63      // Ensure we have navigated to menu
64      const menuTiles = await page.getByTestId('menu-item-card');
65      expect(await menuTiles.count()).toBeGreaterThan(0);
66    });
67
68    step('assert unable to go to login page when signed in', async () => {
69      const loginButton = await page.getByTestId('account-button');
70      expect(loginButton).toContainText('Welcome');
71      await loginButton.click();
72
73      const url = page.url();
74      expect(url).toContain('/order');
75    });
76  });
```

login.journey.ts

```typescript
import { journey, step, monitor, expect, before } from '@elastic/synthetics';

journey('Replicator Login Journey', ({ page, params }) => {
  // Only relevant for the push command to create
  // monitors in Kibana
  monitor.use({
    id: 'synthetics-replicator-monitor-login',
    schedule: 30,
    enabled: !params.isLocal
  });

  before(async () => {
    await page.goto(params.url);
  });

  step('assert home page loads', async () => {
    const header = await page.locator('h1');
    expect(await header.textContent()).toBe('Replicatr');
  });

  step('assert go to login page', async () => {
    const loginButton = await page.getByTestId('account-button');
    expect(loginButton).toHaveText('Login');
    await loginButton.click();

    const url = page.url();
    expect(url).toContain('/login');

    const loginForm = await page.getByTestId('login-form');
    expect(loginForm).toBeDefined();
  });

  step('assert manual login successful', async () => {
    const loginForm = await page.getByTestId('login-form');
    expect(loginForm).toBeDefined();

    // Check submit button is disabled before entry
    const submitButton = await page.getByTestId('login-button');
    expect(submitButton).toBeDisabled();

    // Add credentials
    const username = params.username;
    const usernameInput = page.getByTestId('username-input');
    usernameInput.fill(username);
    await expect(usernameInput).toHaveValue(username);

    const password = params.password;
    const passwordInput = page.getByTestId('password-input');
    passwordInput.fill(password);
    await expect(passwordInput).toHaveValue(password);

    // Check login button is now enabled
    expect(submitButton).toBeEnabled();
    await submitButton.click();

    // Wait for login
    await page.waitForURL('**/order');

    // Validate we have logged in successfully
    const loginButton = await page.getByTestId('account-button');
    expect(loginButton).toContainText('Welcome');

    // Ensure we have navigated to menu
    const menuTiles = await page.getByTestId('menu-item-card');
    expect(await menuTiles.count()).toBeGreaterThan(0);
  });

  step('assert unable to go to login page when signed in', async () => {
    const loginButton = await page.getByTestId('account-button');
    expect(loginButton).toContainText('Welcome');
    await loginButton.click();

    const url = page.url();
    expect(url).toContain('/order');
  });
});
```

synthetics-replicator — -zsh — 100×30

```
carly.richmond@Carlys-MBP synthetics-replicator % npm run test

> synthetics-replicator@0.0.0 test
> cd apps/synthetics-replicator-tests && npm run test


> synthetics-replicator-tests@1.0.0 test
> npx @elastic/synthetics journeys


Journey: Replicator Login Journey
   ✓  Step: 'assert home page loads' succeeded (61 ms)
   ✓  Step: 'assert go to login page' succeeded (96 ms)
   ✓  Step: 'assert manual login successful' succeeded (386 ms)
   ✓  Step: 'assert unable to go to login page when signed in' succeeded (67 ms)

Journey: Recorded Order journey
   ✓  Step: 'Go to order items page' succeeded (295 ms)
   ✓  Step: 'Add item to cart successfully' succeeded (84 ms)
   ✓  Step: 'Add 2nd item to cart successfully' succeeded (67 ms)
   ✓  Step: 'Add 3rd item to cart successfully' succeeded (67 ms)

Journey: Replicator Order Journey
   ✓  Step: 'assert home page loads' succeeded (55 ms)
   ✓  Step: 'assert move to order page' succeeded (219 ms)
   ✓  Step: 'assert adding to order' succeeded (252 ms)

 11 passed (2451 ms)

carly.richmond@Carlys-MBP synthetics-replicator %
```

workflow-synthetics-replicator.yml

```yaml
name: Push Workflow- Elastic Synthetics With Svelte Replicator
on:
  push:
    branches: [main, master]
jobs:
  test:
    env:
      NODE_ENV: development
      REPLICATR_USERNAME: ${{ secrets.REPLICATR_USERNAME }}
      REPLICATR_PASSWORD: ${{ secrets.REPLICATR_PASSWORD }}
      SYNTHETICS_JUNIT_FILE: junit-synthetics.xml
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm ci
      - run: npm start &
      - run: "npm install @elastic/synthetics && npx @elastic/synthetics . --reporter=junit"
        working-directory: ./apps/synthetics-replicator-tests/journeys
      - name: Publish Unit Test Results
        uses: EnricoMi/publish-unit-test-result-action@v2
        if: always()
        with:
          junit_files: '**/junit-*.xml'
          check_name: Elastic Synthetics Tests
  push:
    env:
      NODE_ENV: production
      REPLICATR_USERNAME: ${{ secrets.REPLICATR_USERNAME }}
      REPLICATR_PASSWORD: ${{ secrets.REPLICATR_PASSWORD }}
      ELASTIC_DEPLOYMENT: ${{ secrets.ELASTIC_DEPLOYMENT }}
      SYNTHETICS_API_KEY: ${{ secrets.SYNTHETICS_API_KEY }}
    needs: test
    defaults:
      run:
        working-directory: ./apps/synthetics-replicator-tests
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm ci
      - run: npm run push
```

```yaml
  push:
    branches: [main, master]
jobs:
  test:
    env:
      NODE_ENV: development
      REPLICATR_USERNAME: ${{ secrets.REPLICATR_USERNAME }}
      REPLICATR_PASSWORD: ${{ secrets.REPLICATR_PASSWORD }}
      SYNTHETICS_JUNIT_FILE: junit-synthetics.xml
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm ci
      - run: npm start &
      - run: "npm install @elastic/synthetics && npx @elastic/synthetics . --reporter=junit"
        working-directory: ./apps/synthetics-replicator-tests/journeys
      - name: Publish Unit Test Results
        uses: EnricoMi/publish-unit-test-result-action@v2
        if: always()
        with:
          junit_files: '**/junit-*.xml'
          check_name: Elastic Synthetics Tests
```

```yaml
      - name: Publish Unit Test Results
        uses: EnricoMi/publish-unit-test-result-action@v2
        if: always()
        with:
          junit_files: '**/junit-*.xml'
          check_name: Elastic Synthetics Tests
  push:
    env:
      NODE_ENV: production
      REPLICATR_USERNAME: ${{ secrets.REPLICATR_USERNAME }}
      REPLICATR_PASSWORD: ${{ secrets.REPLICATR_PASSWORD }}
      ELASTIC_DEPLOYMENT: ${{ secrets.ELASTIC_DEPLOYMENT }}
      SYNTHETICS_API_KEY: ${{ secrets.SYNTHETICS_API_KEY }}
    needs: test
    defaults:
      run:
        working-directory: ./apps/synthetics-replicator-tests
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm ci
      - run: npm run push
```

workflow-synthetics-replicator.yml

```yaml
name: Push Workflow- Elastic Synthetics With Svelte Replicator
on:
  push:
    branches: [main, master]
jobs:
  test:
    env:
      NODE_ENV: development
      REPLICATR_USERNAME: ${{ secrets.REPLICATR_USERNAME }}
      REPLICATR_PASSWORD: ${{ secrets.REPLICATR_PASSWORD }}
      SYNTHETICS_JUNIT_FILE: junit-synthetics.xml
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm ci
      - run: npm start &
      - run: "npm install @elastic/synthetics && npx @elastic/synthetics . --reporter=junit"
        working-directory: ./apps/synthetics-replicator-tests/journeys
      - name: Publish Unit Test Results
        uses: EnricoMi/publish-unit-test-result-action@v2
        if: always()
        with:
          junit_files: '**/junit-*.xml'
          check_name: Elastic Synthetics Tests
  push:
    env:
      NODE_ENV: production
      REPLICATR_USERNAME: ${{ secrets.REPLICATR_USERNAME }}
      REPLICATR_PASSWORD: ${{ secrets.REPLICATR_PASSWORD }}
      ELASTIC_DEPLOYMENT: ${{ secrets.ELASTIC_DEPLOYMENT }}
      SYNTHETICS_API_KEY: ${{ secrets.SYNTHETICS_API_KEY }}
    needs: test
    defaults:
      run:
        working-directory: ./apps/synthetics-replicator-tests
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
      - run: npm ci
      - run: npm run push
```

Monitors

Create Monitor Refresh

Overview Management

Search by name, URL, host, tag, project or locatior

Up Down Disabled

Type 2 Location 1 Tags 0 Frequency 3 Project 1

Monitors status

3 Up

1 Down

1 Disabled

Last 6 hours

1 Errors

Last 12 hours

1 Alerts

Showing **5 Monitors**

Spaces Default Add to dashboard Sort by Status Group by None

Replicator Order Journey

Europe - United Kingdom

Journey

Duration

690 ms

Recorded Order journey

Europe - United Kingdom

Journey

Duration

1 s

Replicator HTTP ping

Europe - United Kingdom

HTTP

Duration

58 ms

Replicator Login Journey

Europe - United Kingdom

Journey

Duration

1 s

Local Replicator Order Journey

Replicator Order Journey

Location
Europe - United Kingdom

Status
Failed

Last run
3 minutes ago

Run test manually

Edit monitor

Refresh

Overview | History | Errors | 1 Alerts

Summary To date

5.060%
Availability

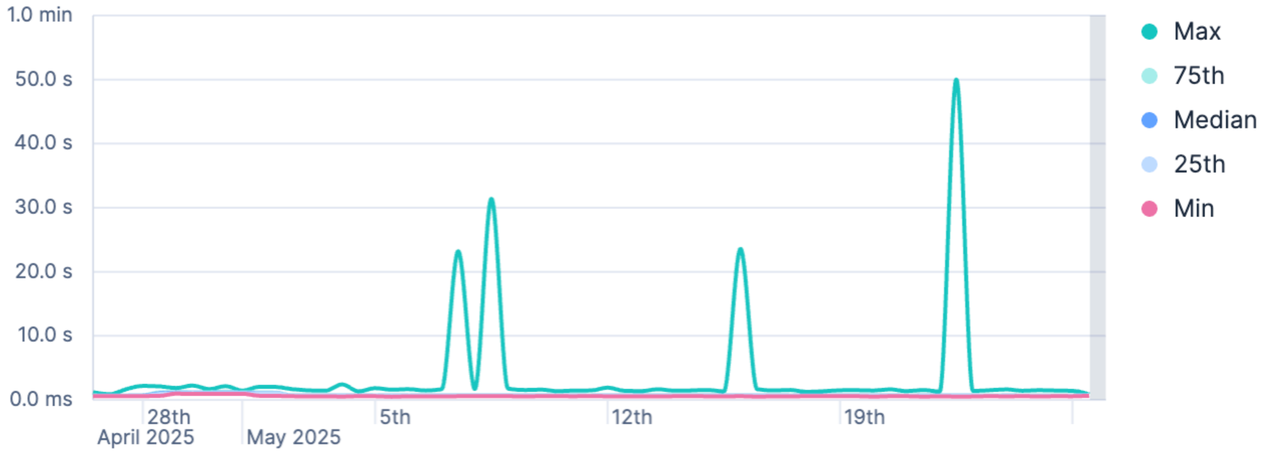
671.4 ms
Median duration

3
Errors

Monitor details

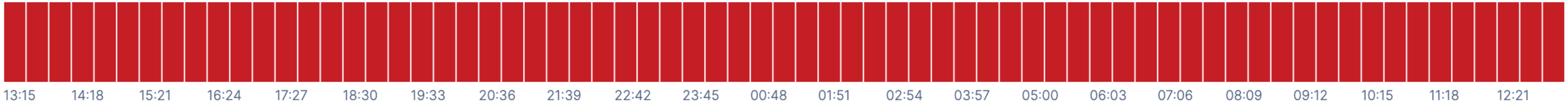
Enabled (all locations)	
URL	https://synthetics-replicator.netlify.app/
Last run	May 26, 2025 @ 12:53:05 PM
Last modified	May 2, 2025 @ 11:50:30 AM
Project ID	synthetics-replicator-tests
Monitor ID	synthetics-replicator-monitor-order-synthetics-replicator-tests-default
Monitor type	Journey
Frequency	Every 3 minutes
Locations	Europe - United Kingdom

Duration trends To date



Status Last 24 hours

View History



Complete Failed Skipped

Last test run

Failed May 26, 2025 @ 12:53:05 PM View test run

error executing step: expect(received).toBeGreaterThan(expected)

View error details

#	Screenshot	Step name	Result	Last successful	
1		assert home page loads	Complete after 138 ms	0 ms	
2		assert move to order page	Failed after 351 ms	0 ms	
3		assert adding to order	Skipped after 0 ms	0 ms	

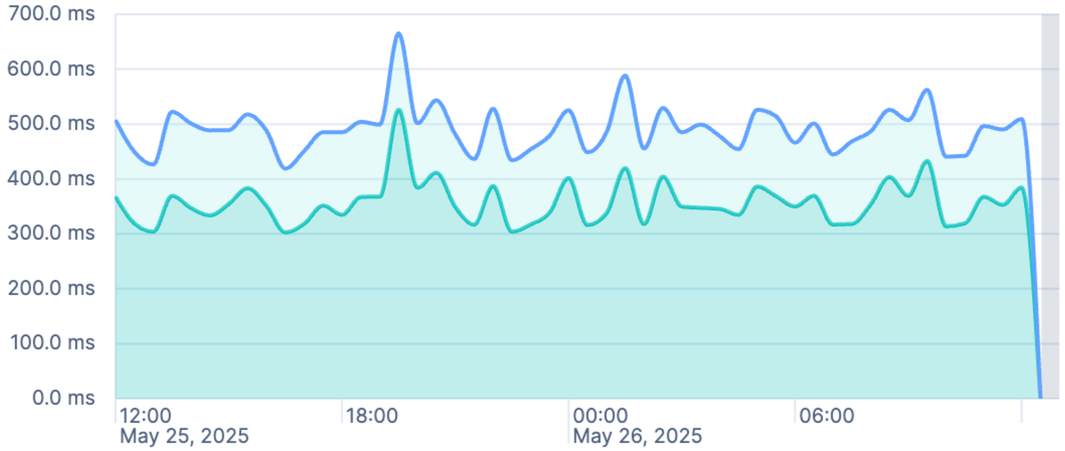
Alerts (4) To date

Actions

2 Active

2 Recovered

Duration by step Last 24 hours



assert move to order page
assert home page loads
assert adding to order

Last 10 Test Runs

All Up Down

View History

Screenshot	@timestamp	Result	Message	Duration
	May 26, 2025 @ 12:53:05 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	701 ms
	May 26, 2025 @ 12:50:05 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	677 ms
	May 26, 2025 @ 12:47:05 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	695 ms
	May 26, 2025 @ 12:44:05 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	743 ms
	May 26, 2025 @ 12:41:05 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	659 ms
	May 26, 2025 @ 12:38:06 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	1 sec
	May 26, 2025 @ 12:35:05 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	615 ms
	May 26, 2025 @ 12:32:06 PM	Failed	error executing step: expect(received).toBeGreaterThan(expected)	704 ms

Replicator Order Journey

2. assert move to order page

Step
Step 2 of 3

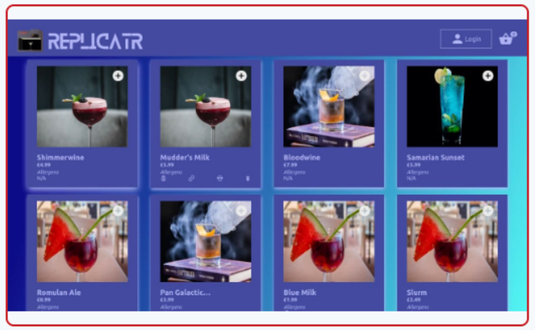
Status
Failed

Location
Europe - United Kingdom

Date
May 26, 2025 @ 12:53:05 PM

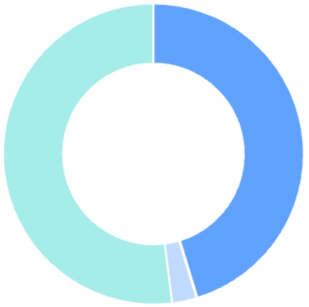
⚠ error executing step: expect(received).toBeGreaterThan(expected)

Screenshot



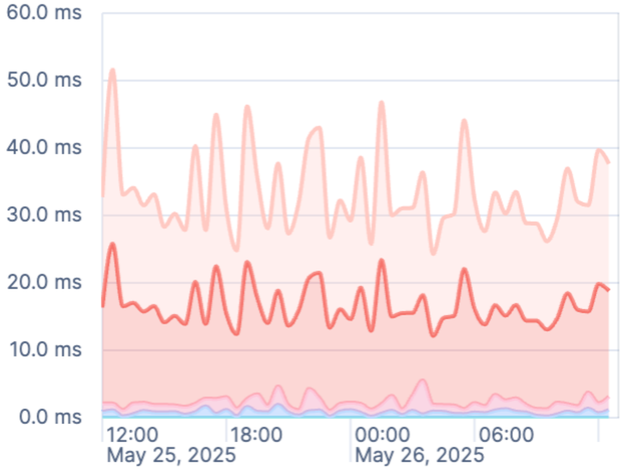
Received | Expected

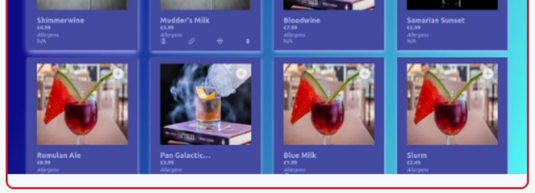
Timing breakdown



Blocked	39 ms ↑
DNS	-- ↑
Connect	-- ↑
TLS	-- –
Wait	743 ms ↑
Receive	647 ms ↑
Send	0 ms –

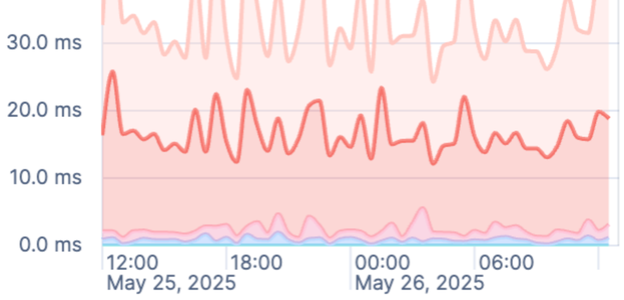
Last 24 hours





Received | Expected

Connect	--	↑
TLS	--	—
Wait	743 ms	↑
Receive	647 ms	↑
Send	0 ms	—



Metrics

Definitions

351 ms — Step duration	0 ms — LCP	0 ms — FCP
0 — CLS	0 ms — DCL	2 MB ↑ Transfer size

Object weight

Total size: **2 MB**

HTML	0 Bytes	—
CSS	0 Bytes	—
Font	0 Bytes	—
JS	0 Bytes	—
Image	0 Bytes	—
Media	0 Bytes	—
XHR	7 KB	—
Other	2 MB	↑

Object count

Total: **18**

HTML	0	—
CSS	0	—
Font	0	—
JS	0	—
Image	0	—
Media	0	—
XHR	5	—
Other	13	↑

Network Requests

Search

Showing **18** of 18 **network requests**

Select an item to apply filter

HTML CSS Font JS Image Media XHR Other

Custom marks

Other 2 MB ↑ Other 13 ↑

Network Requests

Search

Select an item to apply filter

HTML CSS Font JS Image Media XHR Other

Custom marks

Showing **18** of 18 **network requests**

0 ms 20 ms 40 ms 60 ms 80 ms 100 ms 120 ms

1	synthetics-replicator.ne... /drink2.avif
2	synthetics-replicator.net.../drink1.avif
3	synthetics-replicator.ne... /drink3.avif
4	synthetics-replicator.ne... /drink4.avif
5	synthetics-replicator.ne... /coffee.avif
6	synthetics-replicator.n... snacks2.avif
7	synthetics-replicator.n... snacks1.avif
8	synthetics-replicator.n... snacks3.avif
9	synthetics-replicator.n... es/soup.avif
10	synthetics-replicator.n... s/main2.avif
11	synthetics-replicator.n... s/main3.avif
12	synthetics-replicator.n... essert2.avif
13	synthetics-replicator.ne... essert1.avif
14	api.iconify.design/mat... rcle-rounded
15	api.iconify.design/ta... =eggs%2Cmilk
16	api.iconify.design/flu... er%2Cpeanuts
17	api.iconify.design/luci... on?icons=nut
18	api.iconify.design/mdi....icons=gluten

elastic

Find apps, content, and more. ⌘/

Setup guide

CR

D Observability > Alerts > Rules

Add data

Observability

Overview
Alerts
SLOs
Cases
AI Assistant

Logs
Explorer BETA
Stream
Anomalies
Categories

Infrastructure
Inventory
Metrics Explorer
Hosts BETA

APM
Services
Traces
Dependencies

Rules

Rules | Logs

Search

Rule s

● Succeeded: 4 ● Failed: 0 ● Warning: 0

4 rules

Name	Last run	Noti
monitors-alive-check Uptime monitor status	Oct 18, 2024 16:09:06pm 4 minutes ago	
Replicator Order Journey(Simple status alert) Uptime monitor status	Oct 18, 2024 16:12:18pm 29 seconds ago	
Synthetics internal alert Synthetics monitor status	Oct 18, 2024 16:12:18pm 29 seconds ago	
Synthetics internal TLS alert Synthetics TLS certificate	Oct 18, 2024 16:12:18pm 29 seconds ago	

Rows per page: 10

Create rule

Name

Synthetic Monitor Alert

Tags Optional

Synthetics monitor status

Alert when a monitor is down. Learn more

Manage synthetics monitor status rule actions.

Check every 5 minutes

Advanced options

Alert after 5 consecutive matches

Cancel | Show API request | Save

Search your SLOs ...

Status All

Tags All

SLOs

Tell us what you think!

Create SLO

Overview

2	1	0	0	0
Healthy	Violated	No data	Degrading	Stale

Burn rate Last 24h

1	0	1
Active alerts	Recovered alerts	Rules

Showing **1-3** of 3 **SLOs**

Sort by SLO status

Group by None

Replicatr Availability

monitor.name: Replicator Order Journey

1 +2 instances

Target 99%

0%

Replicatr Availability

monitor.name: Replicator HTTP ping

+2 instances

Target 99%

100%

Replicatr Availability

monitor.name: Replicator Login Journey

+2 instances

Target 99%

100%

Single Metric Viewer

Apr 25, 2025 @ 1... → May 26, 2025 @ 1...

Refresh

replicator

Edit job selection

Detector

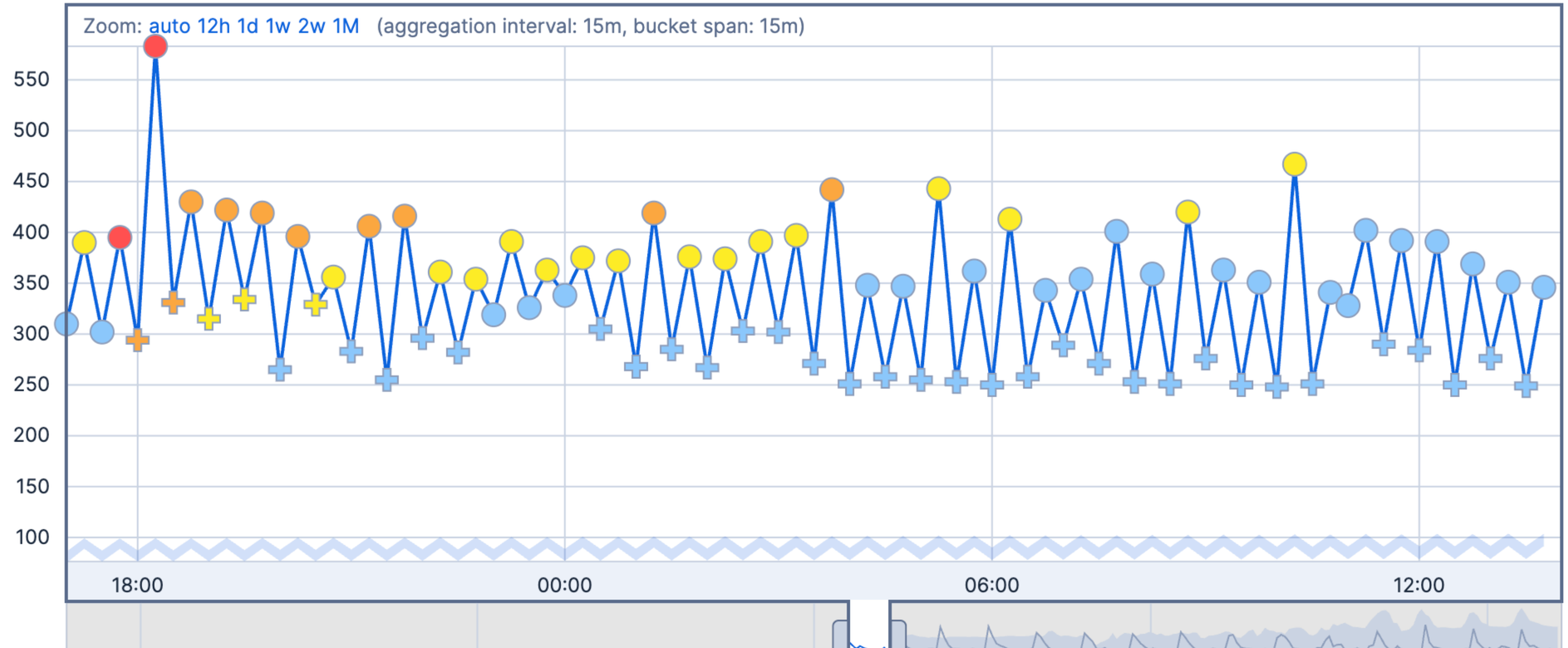
count

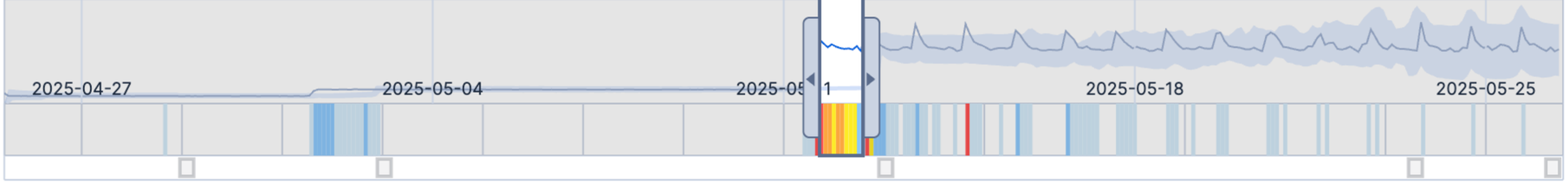
show model bounds

annotations

Forecast

Single time series analysis of count





Anomalies

Severity warning

Interval Auto

Time	Severity	Detector	Actual	Typical	Description	Actions
May 11th 2025, 18:00	83	count	583	93	↑ 6x higher	
May 11th 2025, 17:00	77	count	395	92.7	↑ 4x higher	
May 11th 2025, 19:00	70	count	422	93.2	↑ 5x higher	
May 11th 2025, 20:00	59	count	396	93.3	↑ 4x higher	
May 11th 2025, 21:00	57	count	416	93.4	↑ 4x higher	
May 12th 2025, 03:00	51	count	442	94.3	↑ 5x higher	
May 12th 2025, 01:00	50	count	419	94.1	↑ 4x higher	
May 12th 2025, 05:00	46	count	443	94.7	↑ 5x higher	
May 11th 2025, 23:00	42	count	391	93.8	↑ 4x higher	
May 12th 2025, 10:00	41	count	467	95.5	↑ 5x higher	
May 11th 2025, 22:00	38	count	361	93.6	↑ 4x higher	
May 12th 2025, 00:00	37	count	375	93.9	↑ 4x higher	

Logs

Traces

Source: Ruben. on Unsplash

Logs

Metrics

Traces

Source: Alex Knight on Unsplash

SYNTHETIC MONITORING

elastic

Source: Thomas Kinto on Unsplash

OpenTelemetry

carlyrichmond.bsky.social

DANKE!
THANK YOU!
MERCI!
GRAZIE!
GRACIAS!
DANK JE WEL!

. . . . . .

SCAN ME

carlyrichmond.bsky.social

elastic