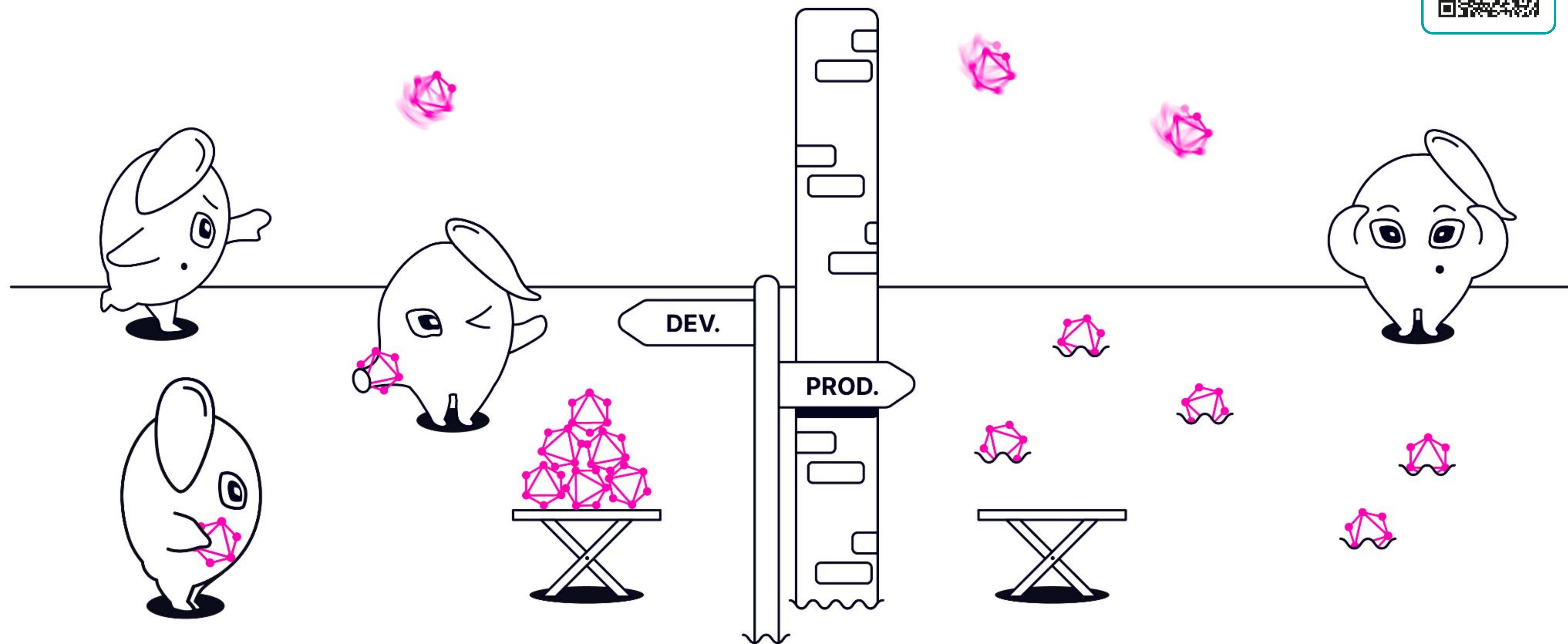




# What could go wrong with a **GraphQL query** and can **OpenTelemetry** help?

Sonja Chevre & Ahmet Soormally



# GraphQL 101

1

What kinds of problems does GraphQL solve?

2

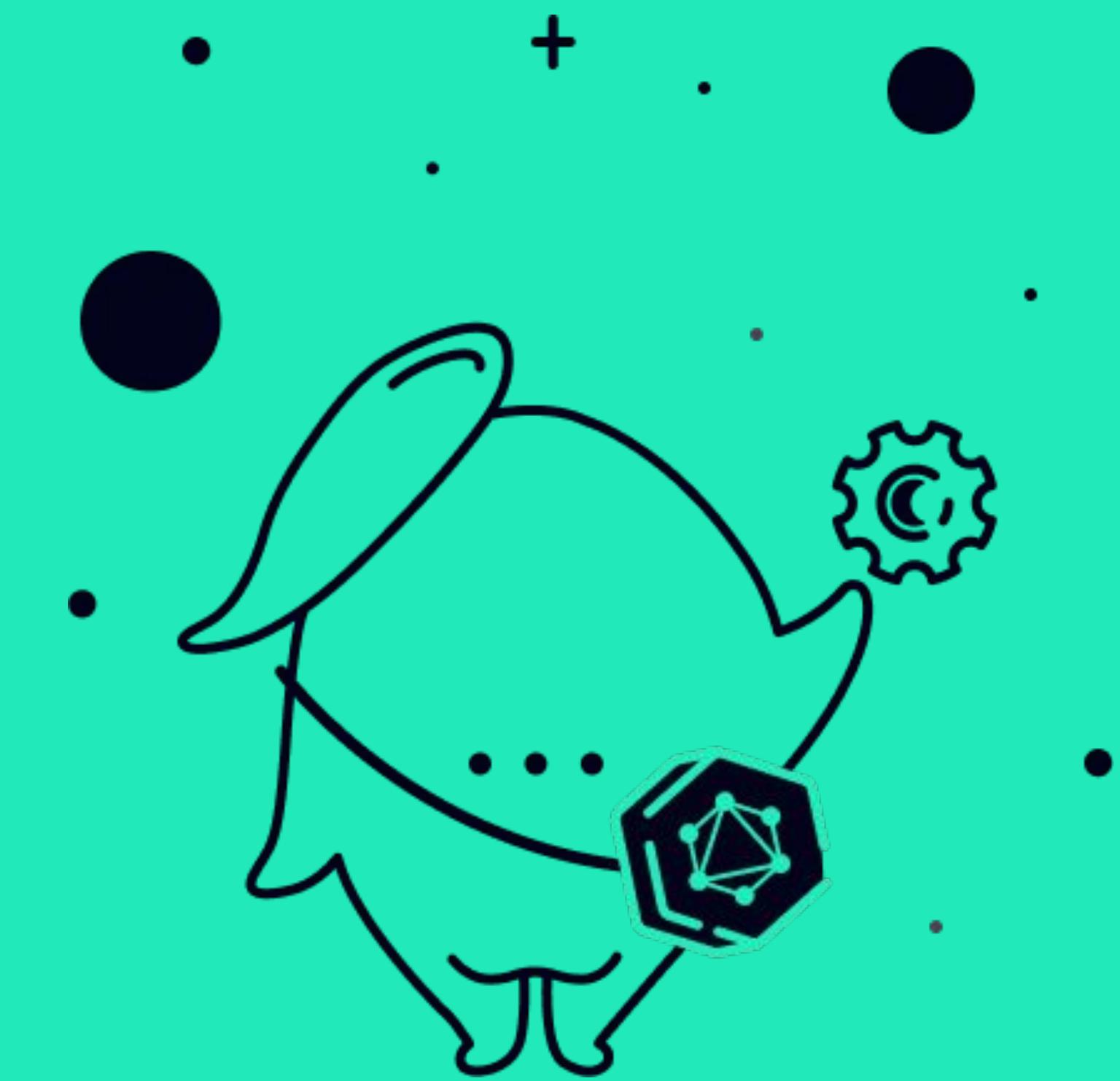
What are the general use-cases for GraphQL?

3

What does a query & schema look like?

4

How is GraphQL different from your typical REST API?





# GraphQL 101

## SQL

```
SELECT  
    authors.name,  
    posts.title  
FROM  
    posts  
LEFT JOIN authors  
    ON authors.id = posts.author_id  
WHERE posts.status = 'published';
```

## REST

```
GET /posts?status=published  
  
GET /authors/2  
GET /authors/7  
GET /authors/9  
...
```

## GRAPHQL

```
query {  
  postsByStatus(status: "published") {  
    title  
    author {  
      name  
    }  
  }  
}
```

# Overfetching



## SQL

```
SELECT
    authors.name,
    posts.title
FROM
    posts
LEFT JOIN authors
    ON authors.id = posts.author_id
WHERE posts.status = 'published';
```

## REST

**GET /posts?status=published**

GET /authors/2  
GET /authors/7  
GET /authors/9

## GRAPHQL

```
query {
  postsByStatus(status: "published") {
    title
    author {
      name
    }
  }
}
```

# Underfetching



## SQL

```
SELECT
    authors.name,
    posts.title
FROM
    posts
LEFT JOIN authors
    ON authors.id = posts.author_id
WHERE posts.status = 'published';
```

## REST

```
GET /posts?status=published
GET /authors/2
GET /authors/7
GET /authors/9
...
```

## GRAPHQL

```
query {
  postsByStatus(status: "published") {
    title
    author {
      name
    }
  }
}
```



# GraphQL 101

## Schema

```
type Query {  
  getPostsByStatus(status: String!): [Post]  
}
```

```
type User {  
  id: ID!  
  name: String!  
  email: String!  
  posts: [Post]  
  comments: [Comment]  
}
```

```
type Post {  
  title: String  
  content: String  
  status: String  
  author: User  
  comments: [Comment]  
}
```

```
type Comment {  
  id: ID!  
  author: User  
  comment: String  
}
```

## Operation

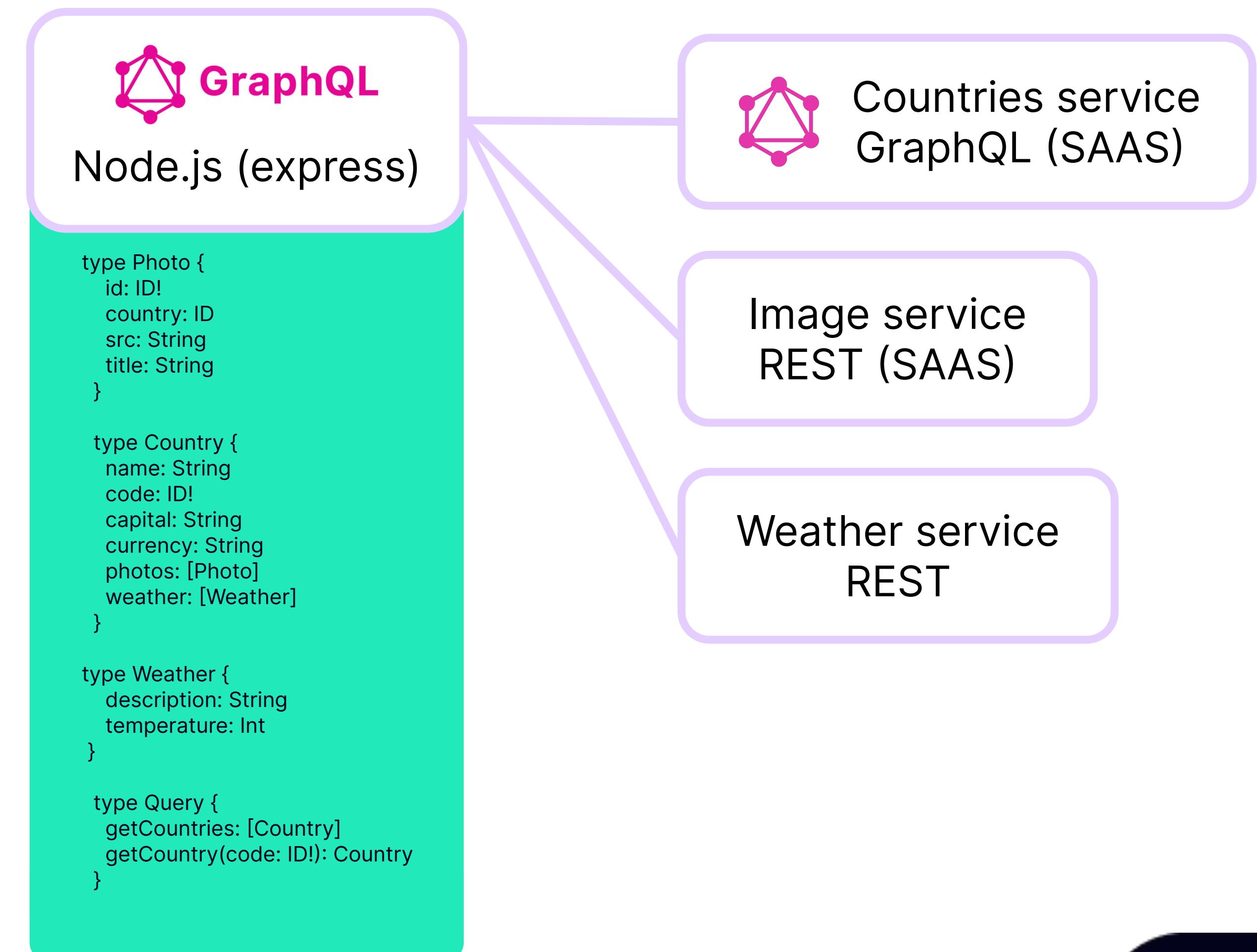
```
query {  
  getPostsByStatus(status: "published") {  
    title  
    author {  
      name  
    }  
  }  
}
```

## Response

```
{  
  "data": [  
    {  
      "title": "abc",  
      "author": {  
        "name": "Ahmet"  
      }  
    },  
    {  
      "title": "def",  
      "author": {  
        "name": "Sonja"  
      }  
    }  
  ]  
}
```

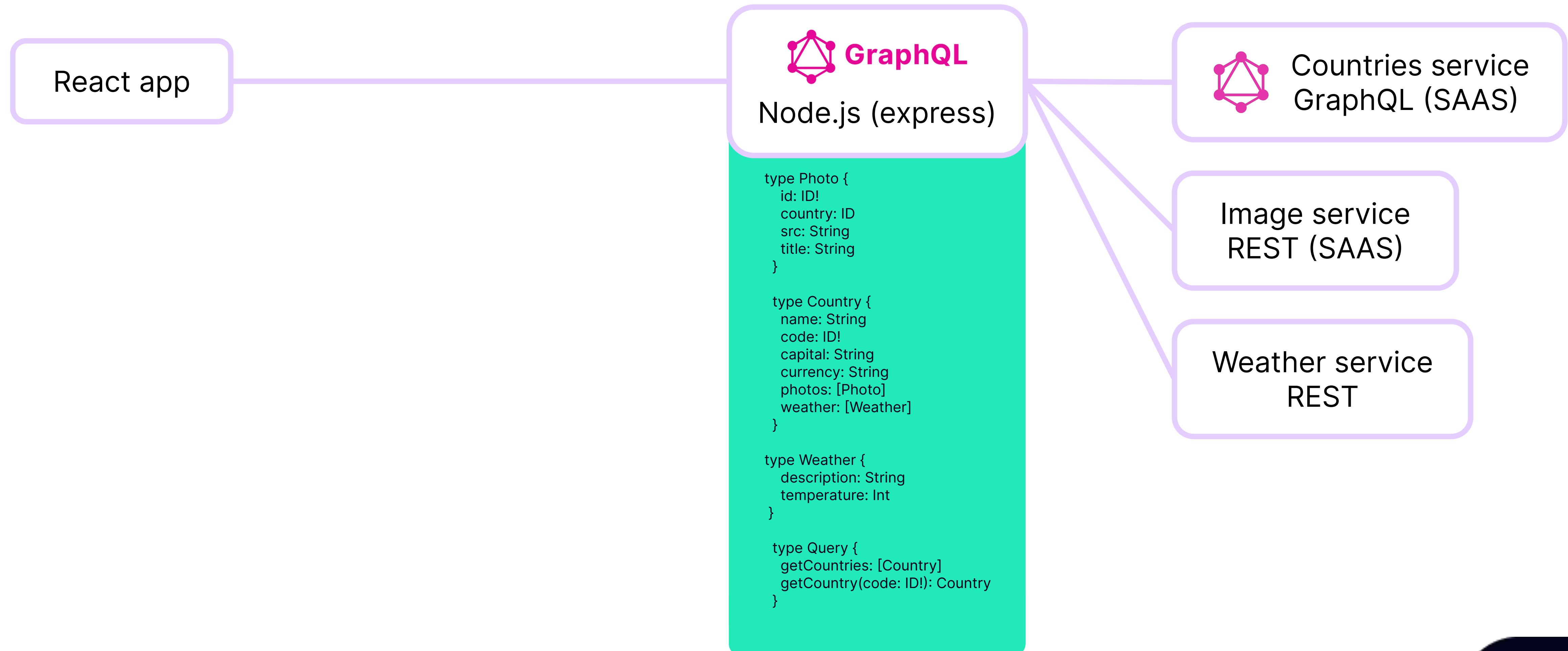


# Kube travel - Server side



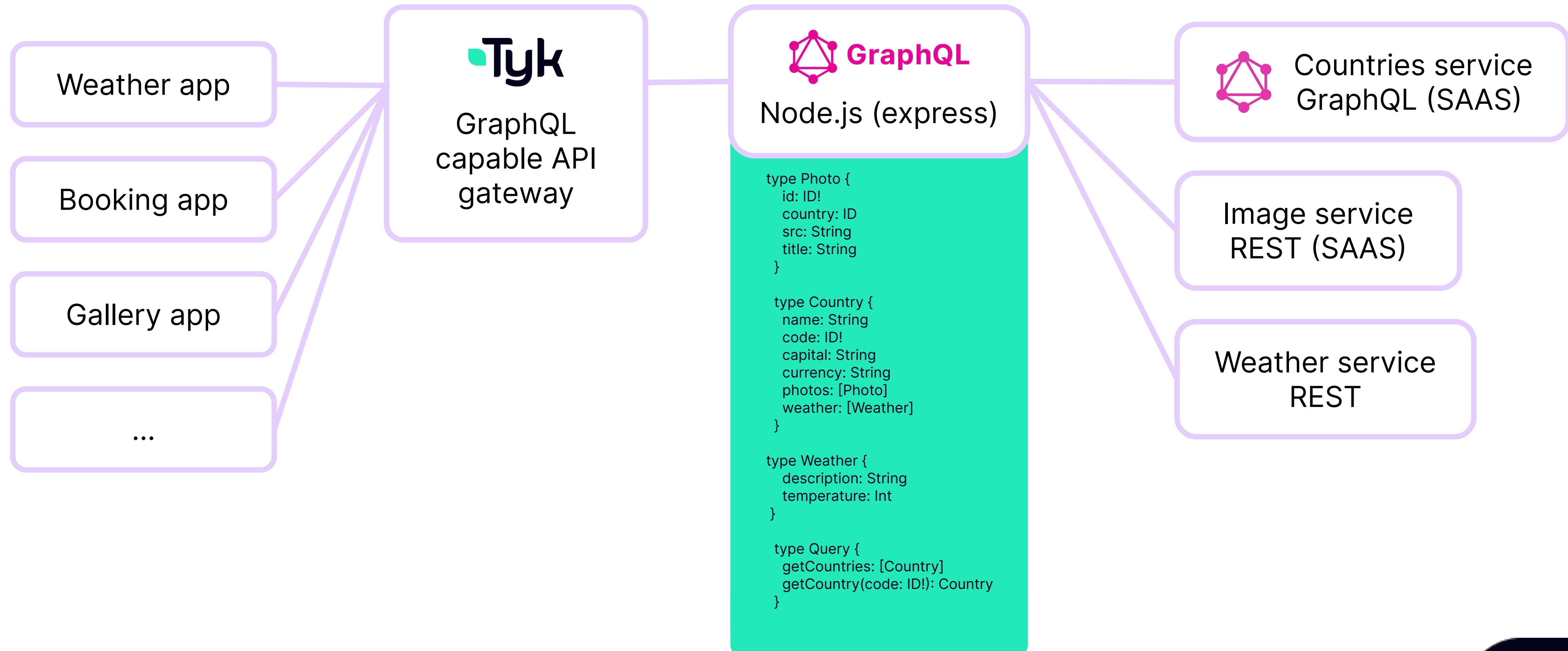


# Kube travel app





# Kube travel API product





# How do we monitor GraphQL in production?

Let's apply the **RED** method to GraphQL

**R**

Rate - the number of requests, per second, your services are serving.

**E**

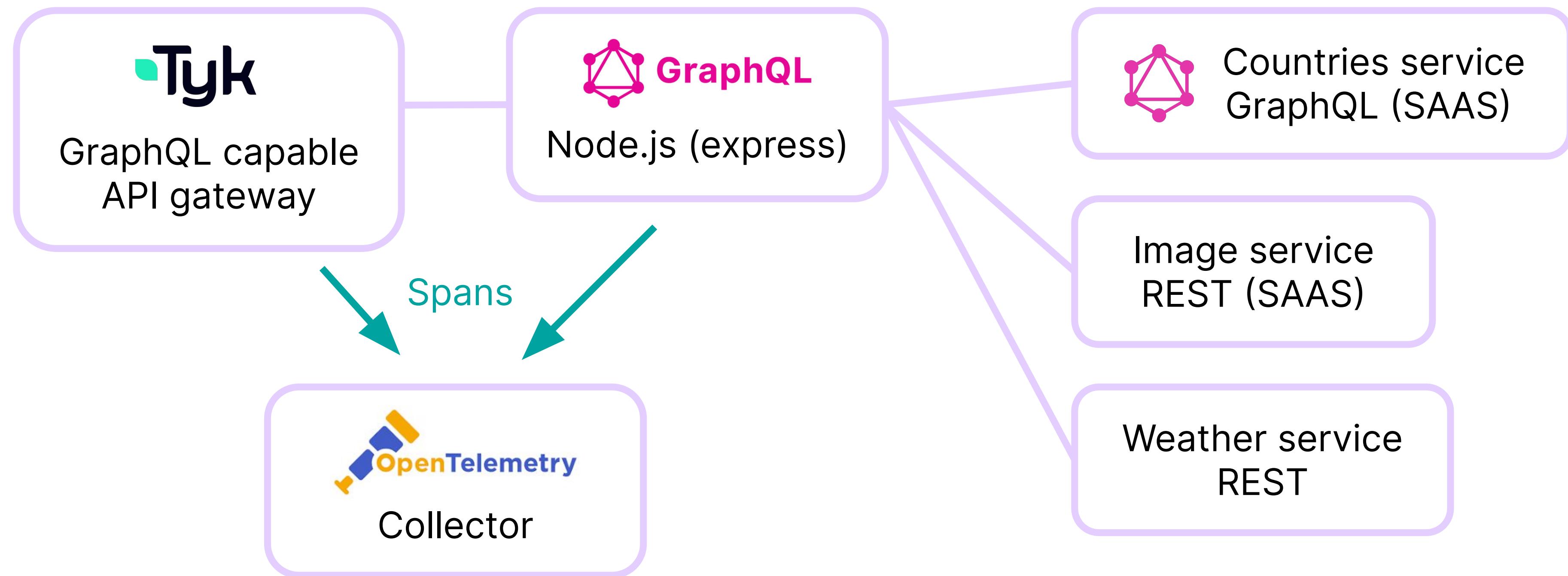
Errors - the number of failed requests per second.

**D**

Duration - distributions of the amount of time each request takes.



# Adding OpenTelemetry to Kube Travel

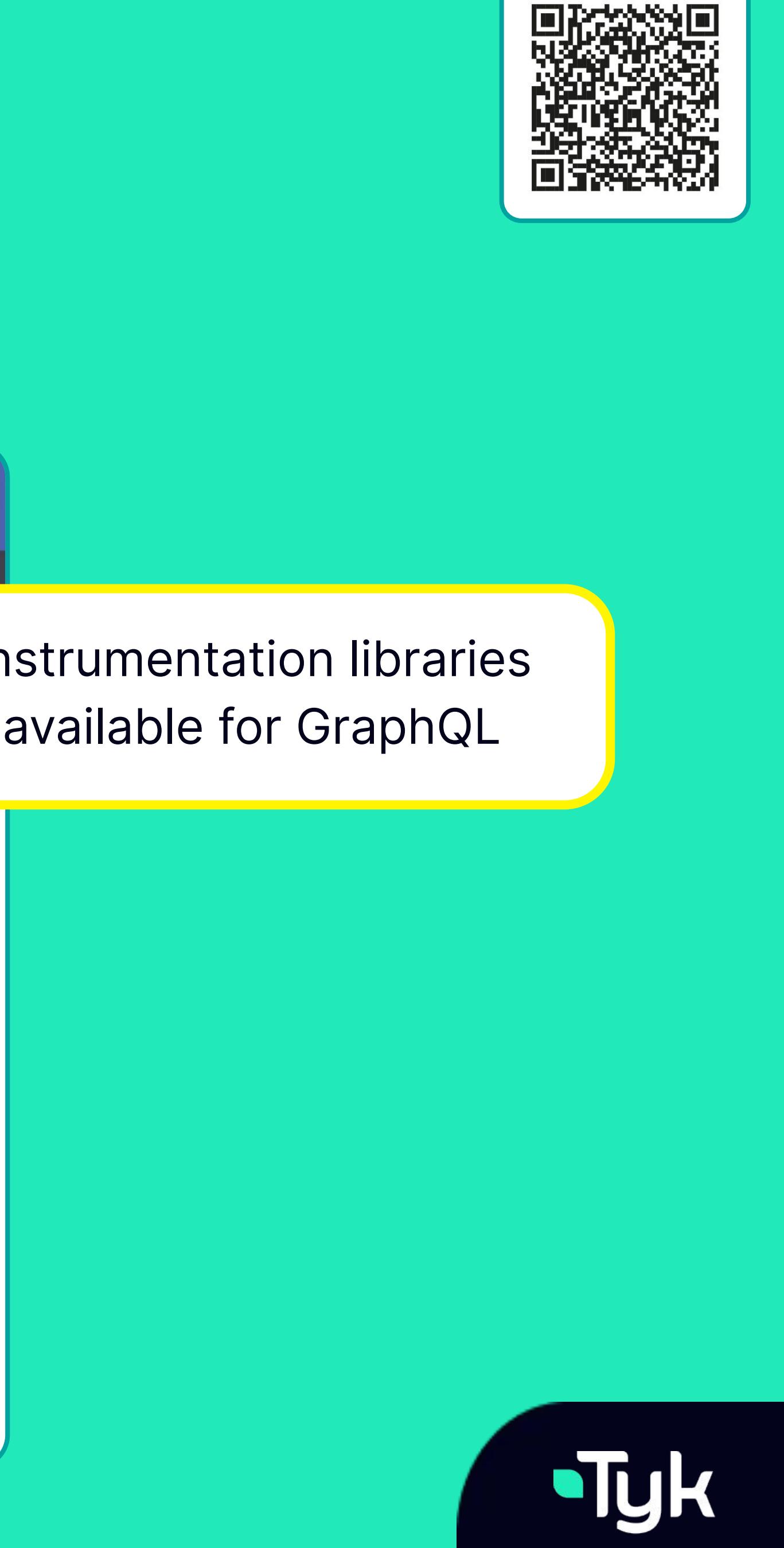




# Instrumenting GraphQL with OpenTelemetry

The screenshot shows the OpenTelemetry Ecosystem page. At the top, there's a navigation bar with links for Docs, Ecosystem, Status, Community, Blog, and a search bar. A yellow callout bubble points from the Ecosystem link to a box containing the text "Instrumentation libraries available for GraphQL". Below the navigation, there's a search bar with "graphql" typed in, and buttons for Submit, Reset, Language (set to "Type"), and Type (set to "Language"). The main content area lists several GraphQL instrumentation packages:

- splunkgraphql -- Instrumentation for github.com/graph-gophers/graphql-go**  
Instrumentation for the `github.com/graph-gophers/graphql-go` package.  
Tags: go, instrumentation
- GraphQL Java Instrumentation**  
This package provides an instrumentation library for GraphQL Java.  
Tags: java, instrumentation
- GraphQL Instrumentation**  
GraphQL instrumentation for Node.js.  
Tags: js, instrumentation
- GraphQL Instrumentation**  
GraphQL instrumentation for Ruby.  
Tags: ruby, instrumentation

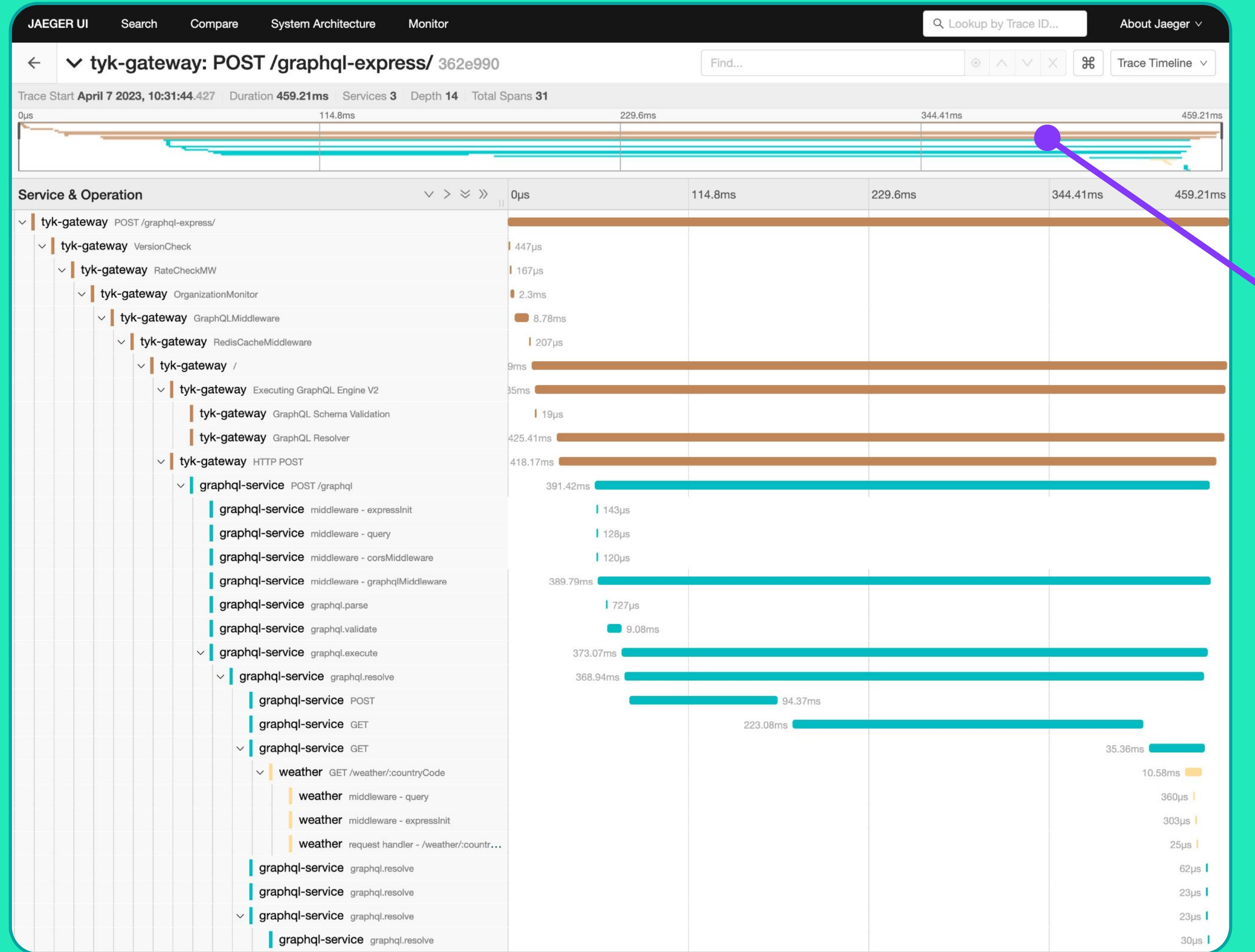




## GraphQL instrumentation

```
graphql-server > JS tracer.js > ...
1  /*tracing.js*/
2  const opentelemetry = require("@opentelemetry/sdk-node");
3  const { OTLPTraceExporter } = require("@opentelemetry/exporter-trace-otlp-http");
4  const { GraphQLInstrumentation } = require('@opentelemetry/instrumentation-graphql');
5  const { HttpInstrumentation } = require('@opentelemetry/instrumentation-http');
6  const { ExpressInstrumentation } = require('@opentelemetry/instrumentation-express');
7  const { Resource } = require('@opentelemetry/resources');
8  const { SemanticResourceAttributes } = require('@opentelemetry/semantic-conventions');

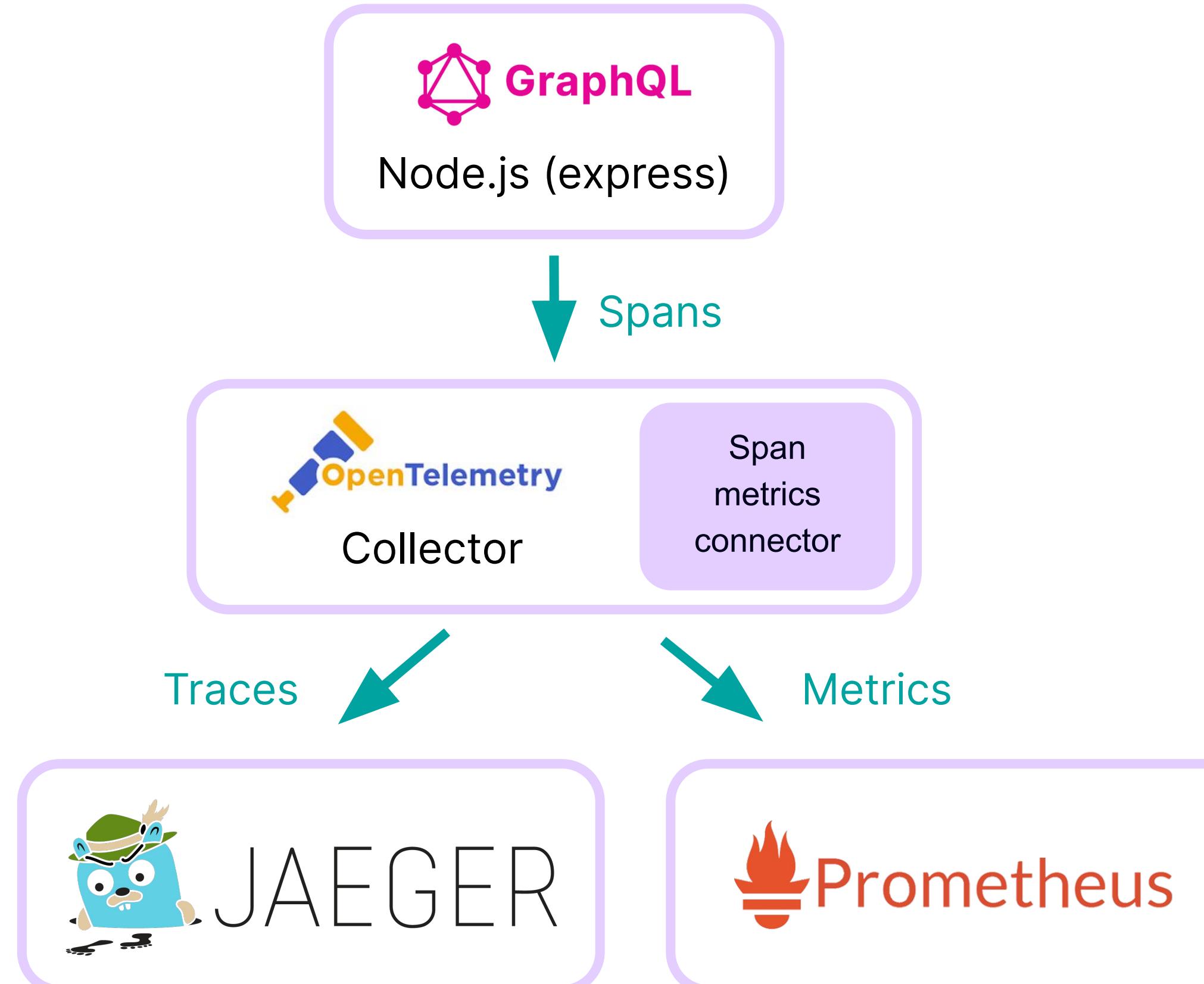
9
10
11 // Receiving service
12 const { context, propagation, trace } = require('@opentelemetry/api');
13
14 // Assume "input" is an object with 'traceparent' & 'tracestate' keys
15 const input = {};
16
17 // Extracts the 'traceparent' and 'tracestate' data into a context object.
18 //
19 // You can then treat this context as the active context for your
20 // traces.
21 let activeContext = propagation.extract(context.active(), input);
22
23
24 const sdk = new opentelemetry.NodeSDK({
25   traceExporter: new OTLPTraceExporter({
26     url: "http://localhost:4318/v1/traces",
27     // optional - collection of custom headers to be sent with each request, empty by default
28     headers: {},
29   }),
30   instrumentations: [
31     new GraphQLInstrumentation({
32       // allowAttributes: true,
33       // depth: 2,
34       // mergeItems: true,
35     }),
36     new HttpInstrumentation(),
37     new ExpressInstrumentation()
38   ],
39   resource: new Resource({
40     [SemanticResourceAttributes.SERVICE_NAME]: 'graphql-service',
41   }),
42 });
43 sdk.start();
```



# An end-to-end distributed trace in Jaeger



# RED metrics from traces with Jaeger



Span Metrics Connector creates two metrics based on spans:

`Calls_total`

Counts the total number of spans, including error spans. Call counts are differentiated from errors via the `status_code` label. Errors are identified as any time series with the label `status_code = "STATUS_CODE_ERROR"`.

`Latency`

`latency_count`: The total number of data points across all buckets in the histogram.

`latency_sum`: The sum of all data point values.

`latency_bucket`: A collection of `n` time series (where `n` is the number of latency buckets) for each latency bucket identified by an `le` (less than or equal to) label. The `latency_bucket` counter with lowest `le` and `le >= span_latency` will be incremented for each span.

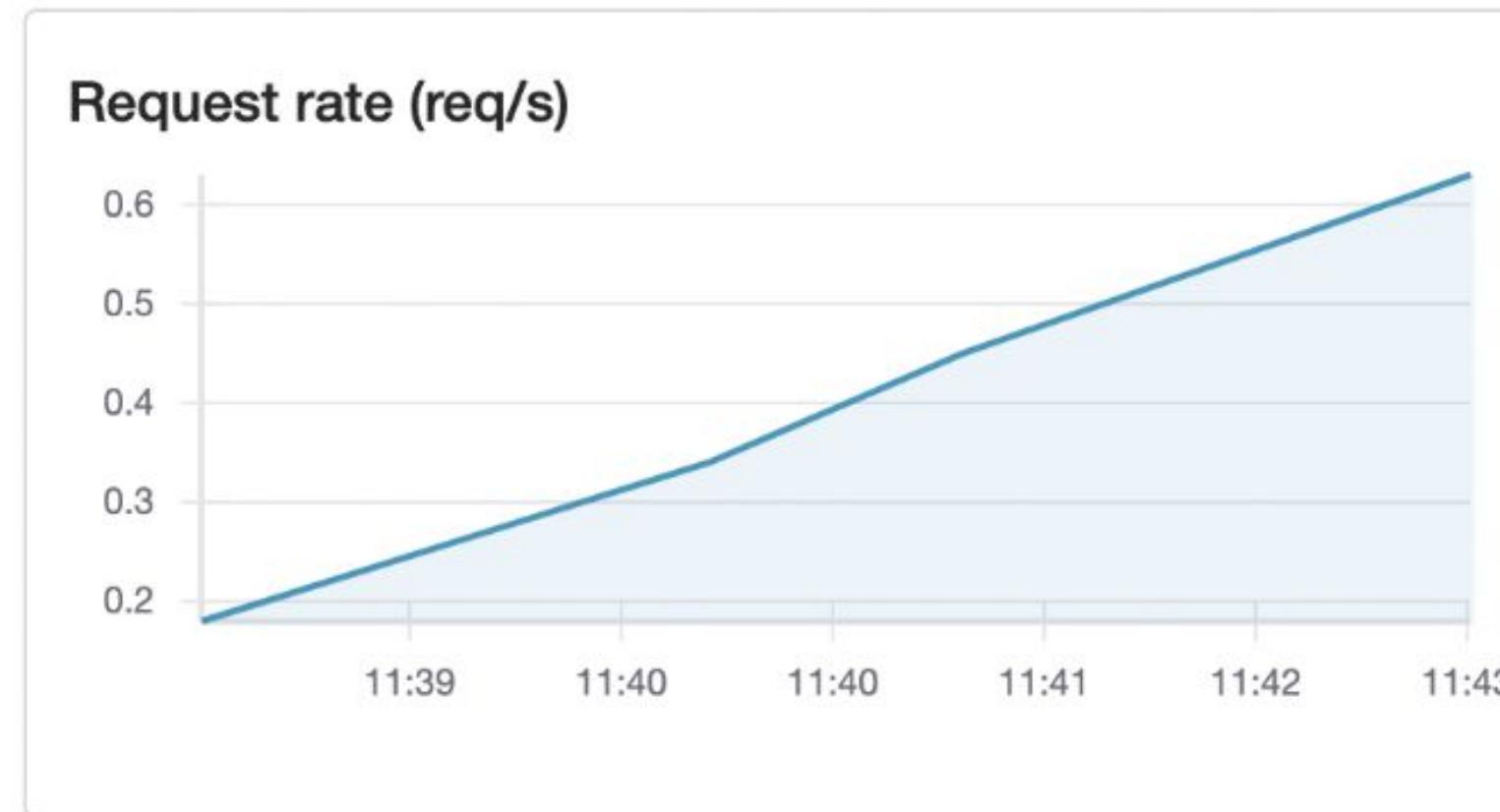
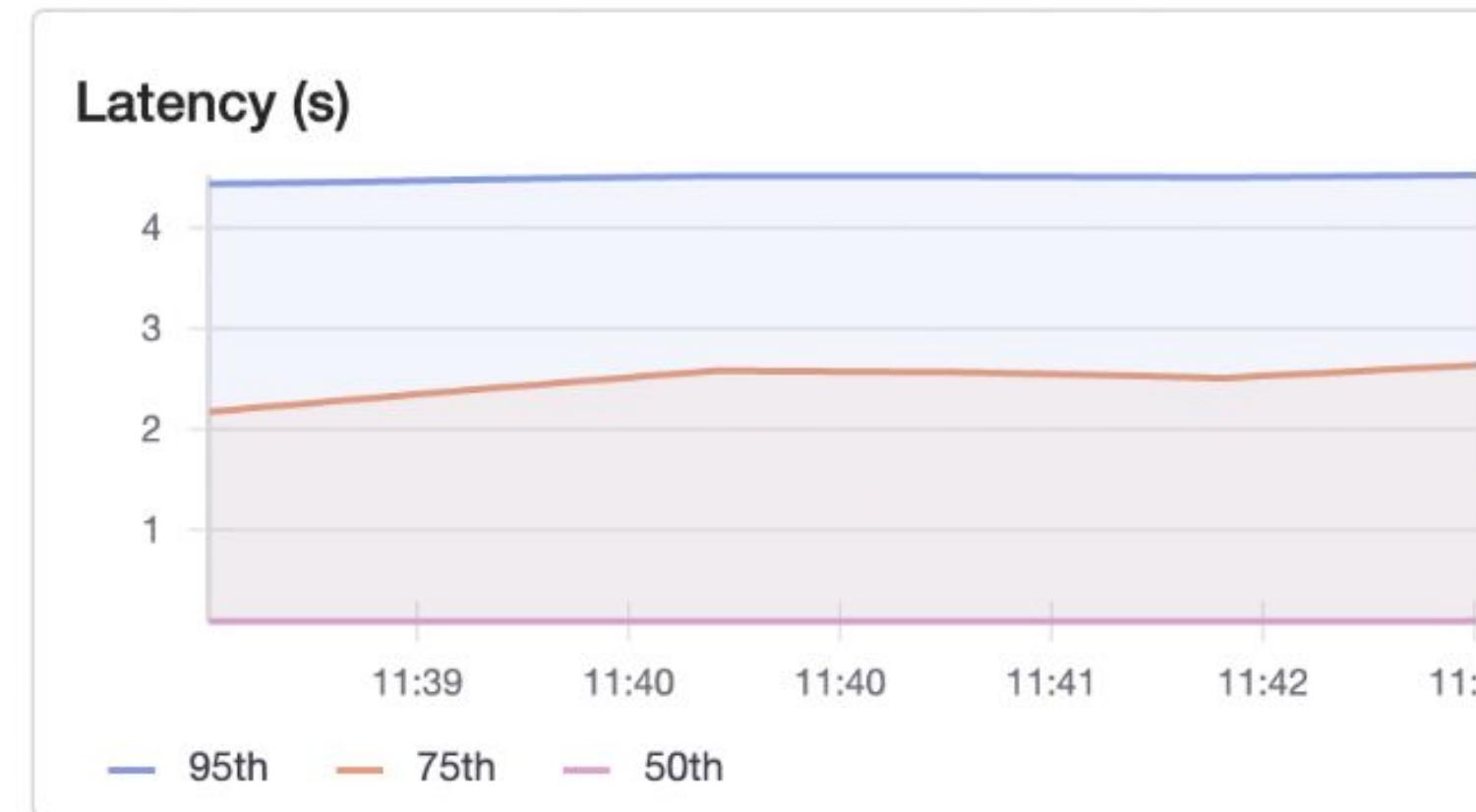


## Choose service

graphql-service

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

Last 5 minutes



## Operations metrics under graphql-service Over the last 5 minutes

Search operation

Name	P95 Latency	Request rate	Error rate	Impact
POST /graphql	4.49s	0.4 req/s	1%	High

Page: 1 of 1 | [View all operations](#)



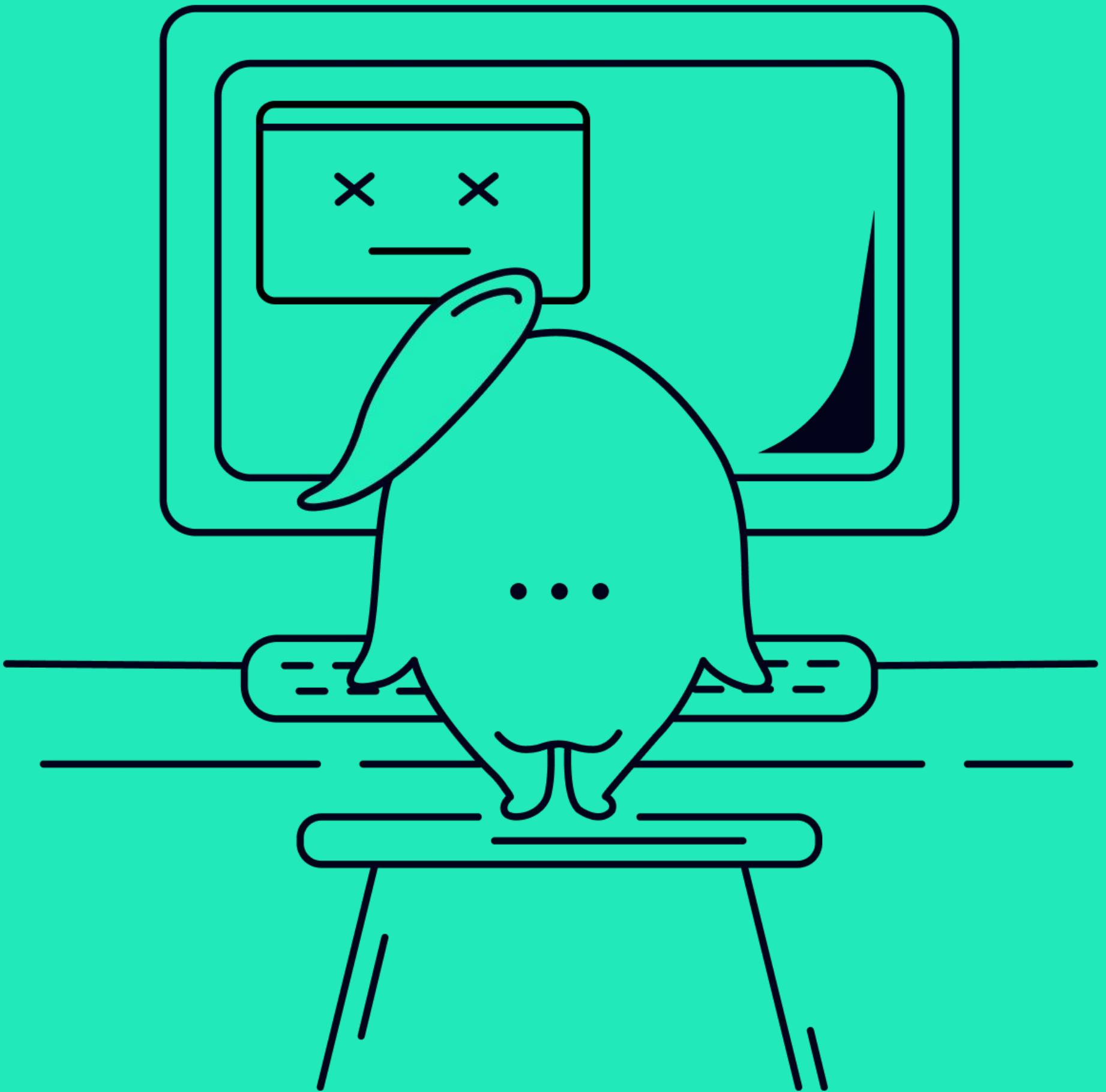
# Errors

1

Upstream errors

2

Resolver Errors





Tyk



Explorer

Prettify

```
1 { country (code: "IT")
2   {
3     code
4     name
5     capital
6     weather {
7       temperature
8     }
9   }
10 }
```



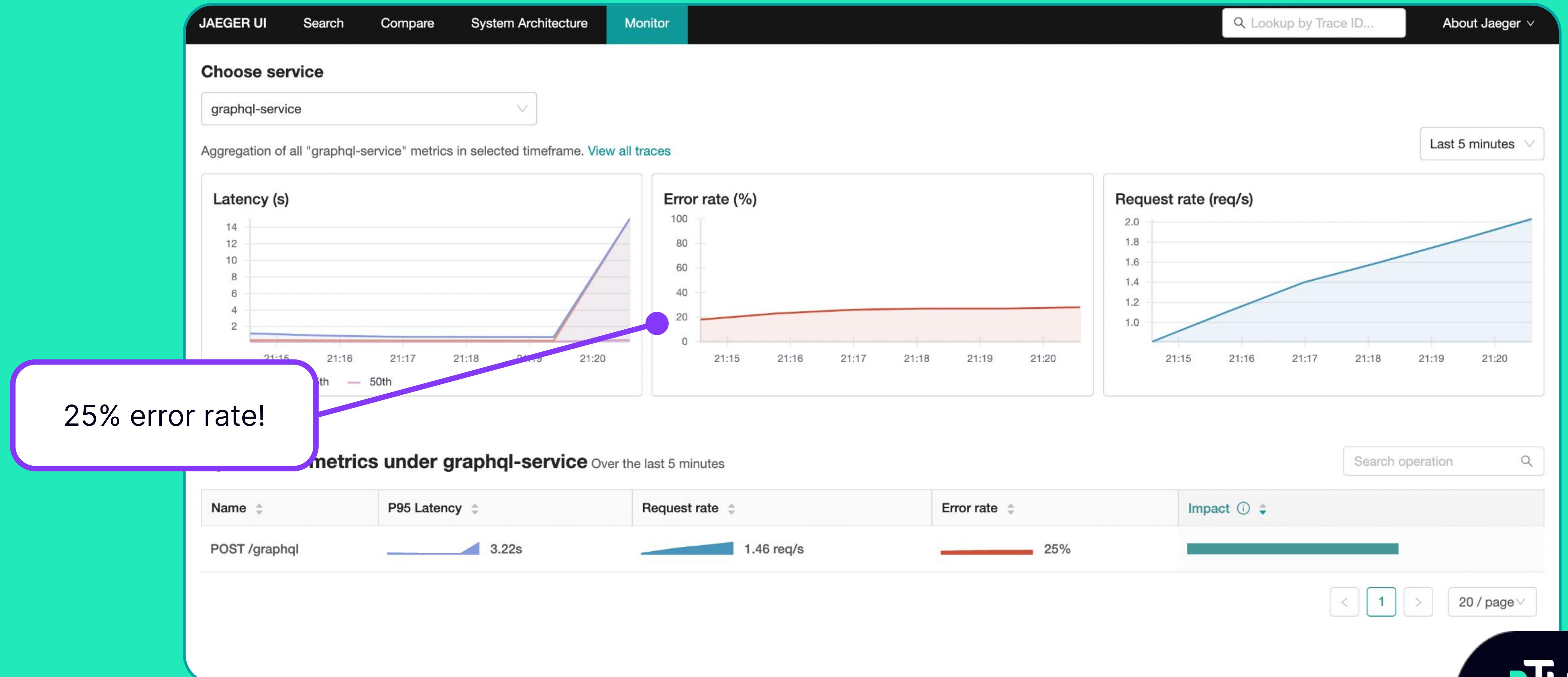
```
{
  "errors": [
    {
      "message": "Failed to fetch country data",
      "locations": [
        {
          "line": 1,
          "column": 16
        }
      ],
      "path": [
        "country"
      ]
    },
    {
      "message": "unable to resolve",
      "locations": [
        {
          "line": 1,
          "column": 3
        }
      ],
      "path": [
        "country"
      ]
    }
  ],
  "data": null
}
```

QUERY VARIABLES REQUEST HEADERS

Tyk



# Something is going on...

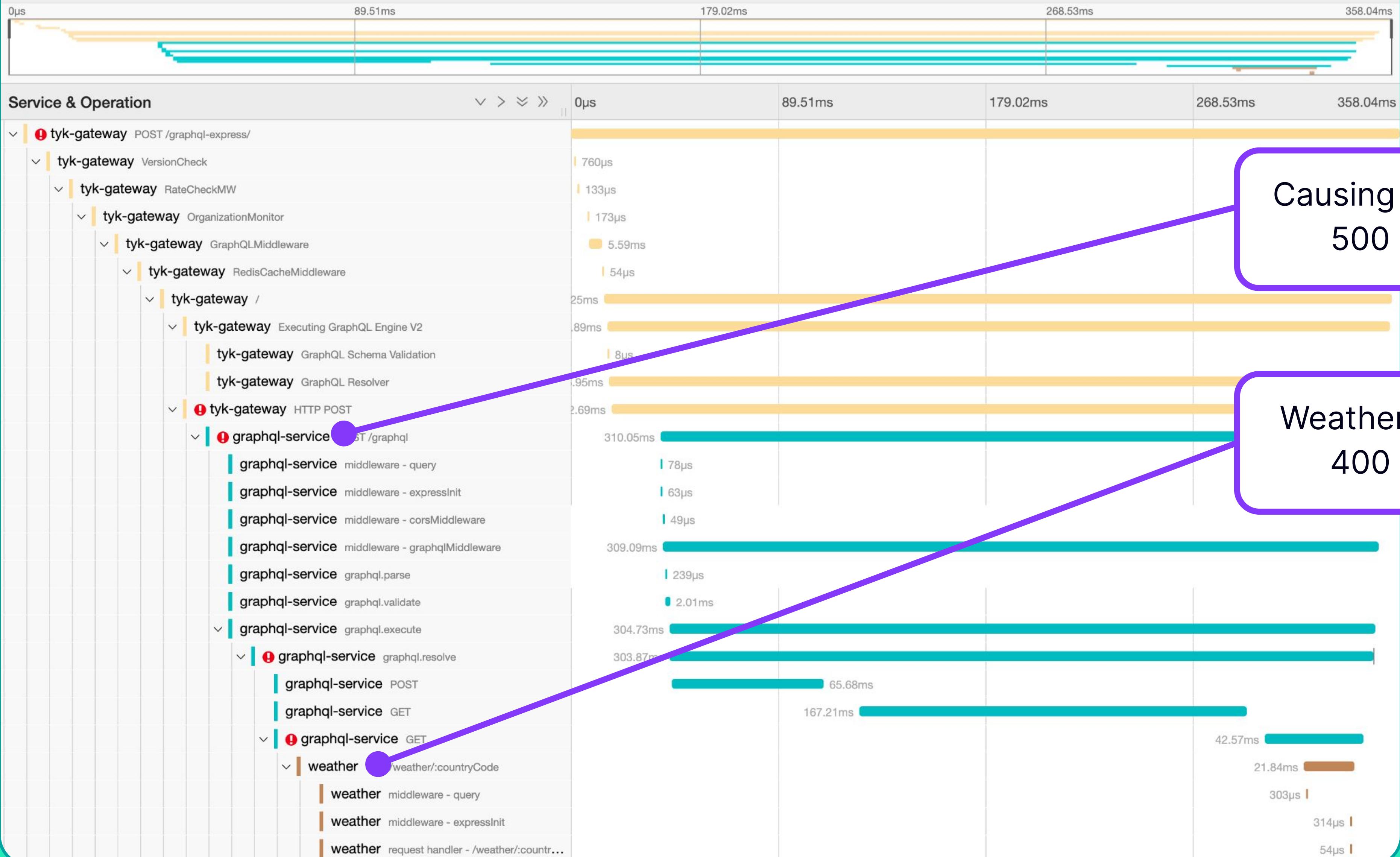




← ✓ tyk-gateway: POST /graphql-express/ 883c158

Find...

Trace Start **April 7 2023, 12:32:14.088** Duration **358.04ms** Services **3** Depth **14** Total Spans **27**



# Causing GraphQL to return a 500 http status code

# Weather service returning a 400 http status code



The GraphQL query that is causing the error

graphql-service graphql.execute      878.09ms

graphql.execute      Service: graphql-service | Duration: 878.09ms | Start Time: 88.54ms

Tags

graphql.operation.type	query
graphql.source	query(\$a: ID!){country(code: \$a){code name capital weather {temperature}}}
internal.span.format	proto
otel.library.name	@opentelemetry/instrumentation-graphql
otel.library.version	0.33.2
span.kind	internal

Process: telemetry.sdk.language = nodejs | telemetry.sdk.name = opentelemetry | telemetry.sdk.version = 1.10.1

SpanID: e6114fdeef5fad28



Tyk



Explorer

Prettify

```
1 { country (code: "NL")
2   {
3     code
4     name
5     weather {
6       temperature
7     }
8   }
9 }
```



```
{
  "errors": [
    {
      "message": "Cannot return null for non-nullable field Weather.temperature.",
      "locations": [
        {
          "line": 1,
          "column": 53
        }
      ],
      "path": [
        "country",
        "weather",
        "temperature"
      ]
    }
  ],
  "data": {
    "country": {
      "code": "NL",
      "name": "Netherlands",
      "weather": null
    }
  }
}
```

QUERY VARIABLES REQUEST HEADERS

Tyk



# Everything looks fine

JAEGER UI    Search    Compare    System Architecture    Monitor        About Jaeger ▾

**Choose service**

graphql-service ▾

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

Last 5 minutes ▾

**Latency (ms)**

95th   75th   50th

22:31 22:32 22:33 22:34 22:35 22:36

Time	95th (ms)	75th (ms)	50th (ms)
22:31	190	175	150
22:35	196	178	152

**Error rate (%)**

22:31 22:32 22:33 22:34 22:35 22:36

Time	Error Rate (%)
22:31	0
22:35	0

**Request rate (req/s)**

22:31 22:32 22:33 22:34 22:35 22:36

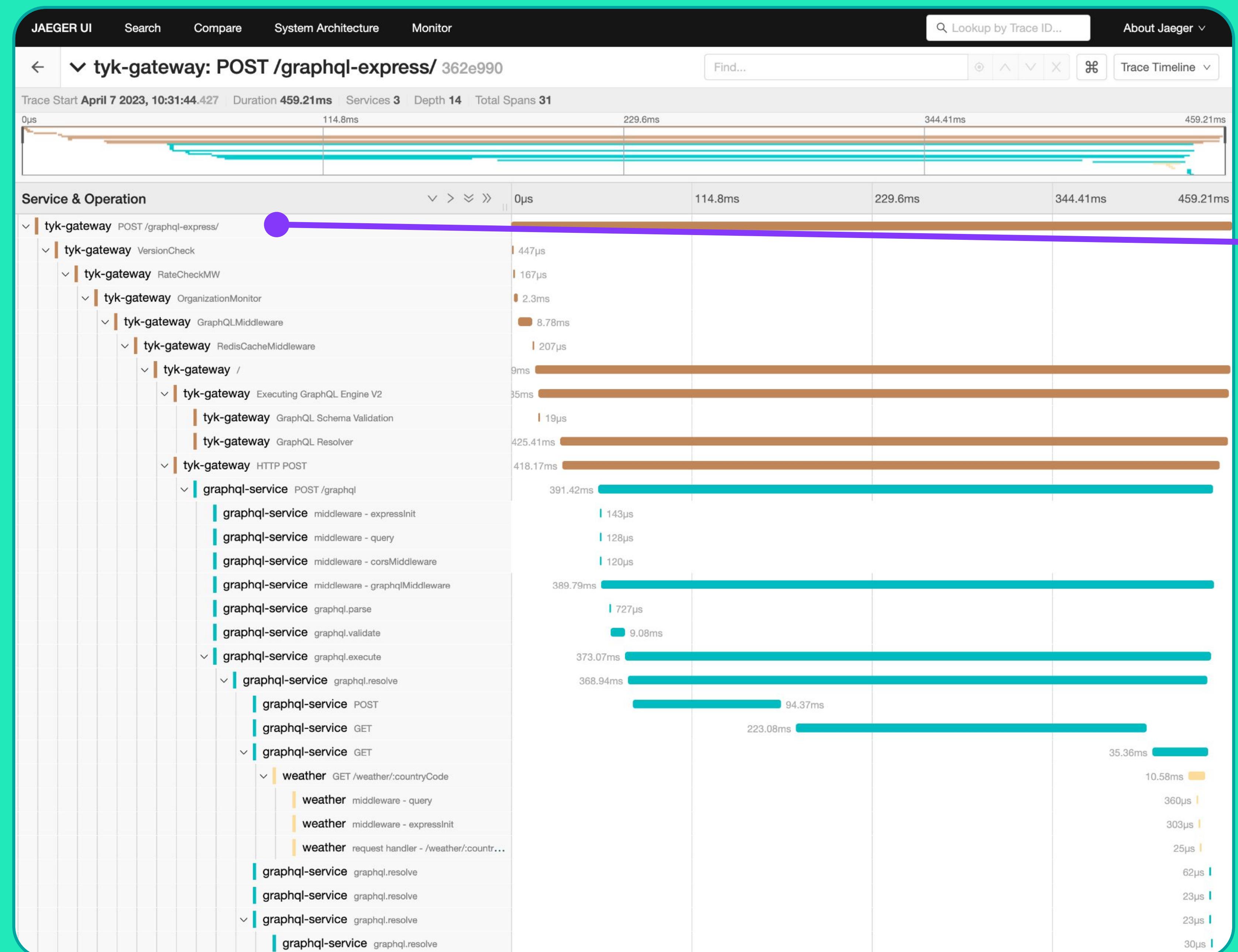
Time	Request Rate (req/s)
22:31	0.2
22:35	1.4

**Operations metrics under graphql-service** Over the last 5 minutes

Search operation ▾

Name	P95 Latency	Request rate	Error rate	Impact
POST /graphql	196.07ms	0.9 req/s	< 0.1%	High

20 / page ▾



All good!



# GraphQL errors

The screenshot shows the Tyk GraphQL playground. On the left, a code editor displays a GraphQL query:

```
1 { country (code: "NL")  
2   {  
3     code  
4     name  
5     weather {  
6       temperature  
7     }  
8   }  
9 }
```

On the right, the response pane shows the JSON response:

```
{  
  "errors": [  
    {  
      "message": "Cannot return null for non-nullable field Weather.temperature.",  
      "locations": [  
        {  
          "line": 1,  
          "column": 53  
        }  
      ],  
      "path": [  
        "country",  
        "weather",  
        "temperature"  
      ]  
    }  
  ],  
  "data": {  
    "country": {  
      "code": "NL",  
      "name": "Netherlands",  
      "weather": null  
    }  
  }  
}
```

A purple callout bubble points from the error message in the response to a purple exclamation mark icon inside a rounded rectangle on the left, containing the text:

⚠ The request returns HTTP status code 200 even though the response contains an error

Here is the error!



# Adding GraphQL to error spans

Span Status MUST be left unset if HTTP status code was in the 1xx, 2xx or 3xx ranges, unless there was another error (e.g., network error receiving the response body; or 3xx codes with max redirects exceeded), in which case status MUST be set to Error .

Attribute	Type	Examples
graphql.operation.name	string	findBookById
graphql.operation.type	string	query ; mutation ; subscription
graphql.document	string	query findBookById { bookById(id: ?) { name } }

Is there a semantic convention we can use?



Report the error to  
the active span  
with manual  
instrumentation

```
280 // Create an Express app
281 const app = express();
282 app.use(cors());
283
284 // Add a GraphQL API endpoint
285 app.use(
286   '/graphql',
287   graphqlHTTP({
288     schema: schema,
289     rootValue: root,
290     graphiql: true,
291     formatError: error => {
292       const params = {
293         message : error.message,
294         stack    : error.stack
295       };
296
297       tracer.startActiveSpan('graphql-tracer', (span) => {
298         span.setAttribute('graphql.error.message', error.message);
299         span.recordException(error.stack);
300         span.setStatus({ code: opentelemetry.SpanStatusCode.ERROR });
301         span.end();
302       });
303     }
304   })
305   return (params);
306 }
307 ),
308 );
309
310 // Start the server
311 app.listen(4000, () => {
312   console.log('GraphQL server running on http://localhost:4000/graphQL');
313 })
```



tyk-gateway: POST /graphql-express/ 3cec4a9

235.07ms

32 Spans | 1 Error

graphql-service (17) | tyk-gateway (11) | weather (4)

Today | 10:24:41 pm | 3 minutes ago

The error is now recorded in the span

Including error details

tyk-gateway HTTP POST

graphql-service POST /graphql

graphql-service middleware - graphqlMiddleware

graphql-service middleware - corsMiddleware

graphql-service middleware - query

graphql-service middleware - expressInit

graphql-service graphql.parse

graphql-service graphql.validate

graphql-service graphql.execute

graphql-tracer

Service: graphql-service | Duration: 23µs | Start Time: 227.66ms

Tags

error	true
graphql.error.message	Cannot return null for non-nullable field Weather.temperature.
internal.span.format	proto
otel.library.name	graphql-tracer
otel.status_code	ERROR
span.kind	internal

Process: telemetry.sdk.language = nodejs | telemetry.sdk.name = opentelemetry | telemetry.sdk.version = 1.10.1

Logs (1)

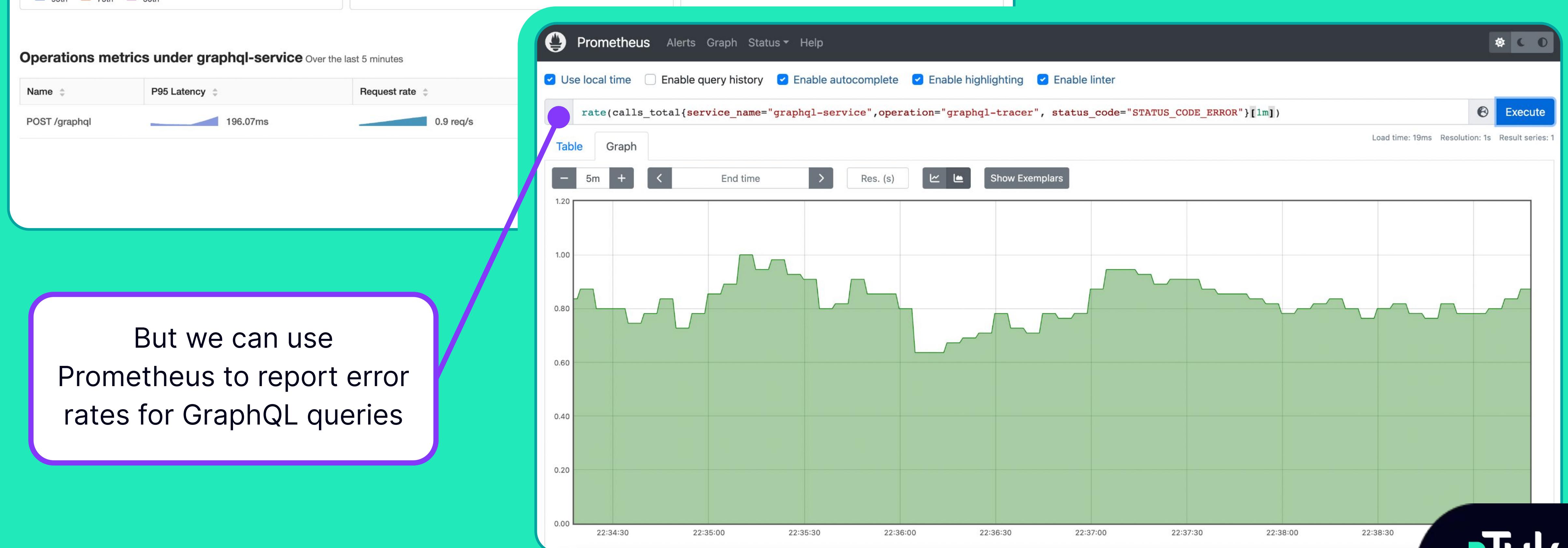
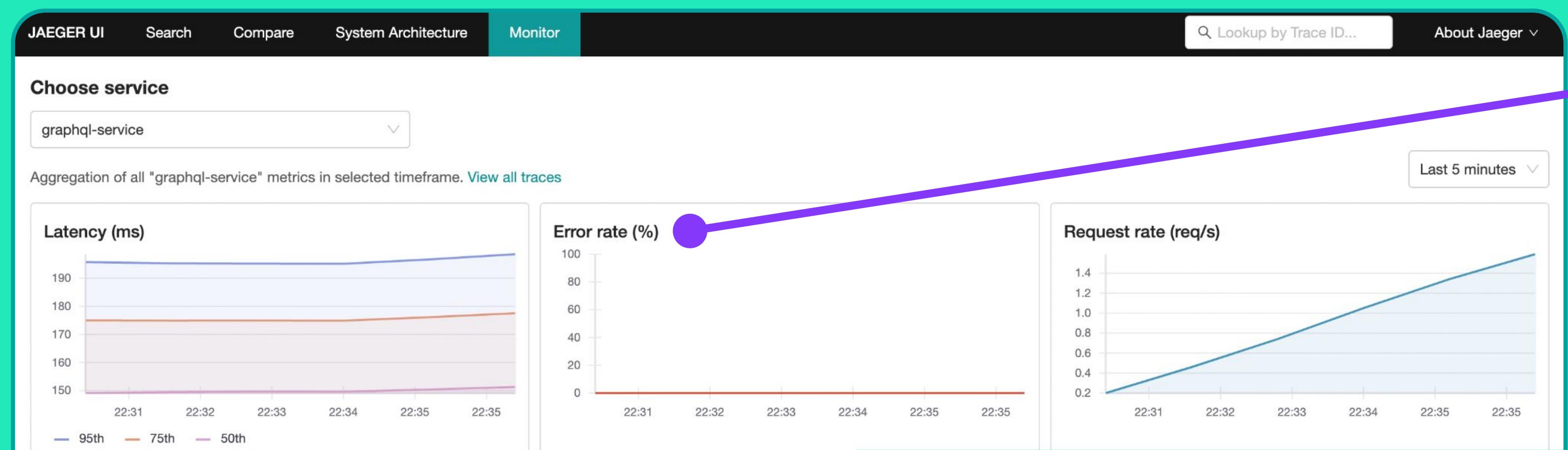
227.68ms

event exception

exception.message Error: Cannot return null for non-nullable field Weather.temperature.  
at completeValue (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:594:13)  
at executeField (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:489:19)  
at executeFields (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:413:20)  
at completeObjectValue (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:914:10)  
at completeValue (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:635:12)  
at executeField (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:489:19)  
at executeFields (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:413:20)  
at completeObjectValue (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:914:10)  
at completeValue (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:635:12)  
at completeValue (/Users/sonja/Documents/graphl-opentelemetry-demo/graphql-server/node\_modules/graphql/execution/execute.js:584:23)

Log timestamps are relative to the start time of the full trace.

SpanID: 7d416fa66ed568ca





# How can we detect performance issues?

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

**Choose service**

graphql-service

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

Last 5 minutes ▾

**Latency (s)**

11:39 11:40 11:41 11:42 11:43

— 95th — 75th — 50th

**Error rate (%)**

11:39 11:40 11:41 11:42 11:43

**Request rate (req/s)**

11:40 11:41 11:42 11:43

Is tracking the latency for a single graphql endpoint good enough?

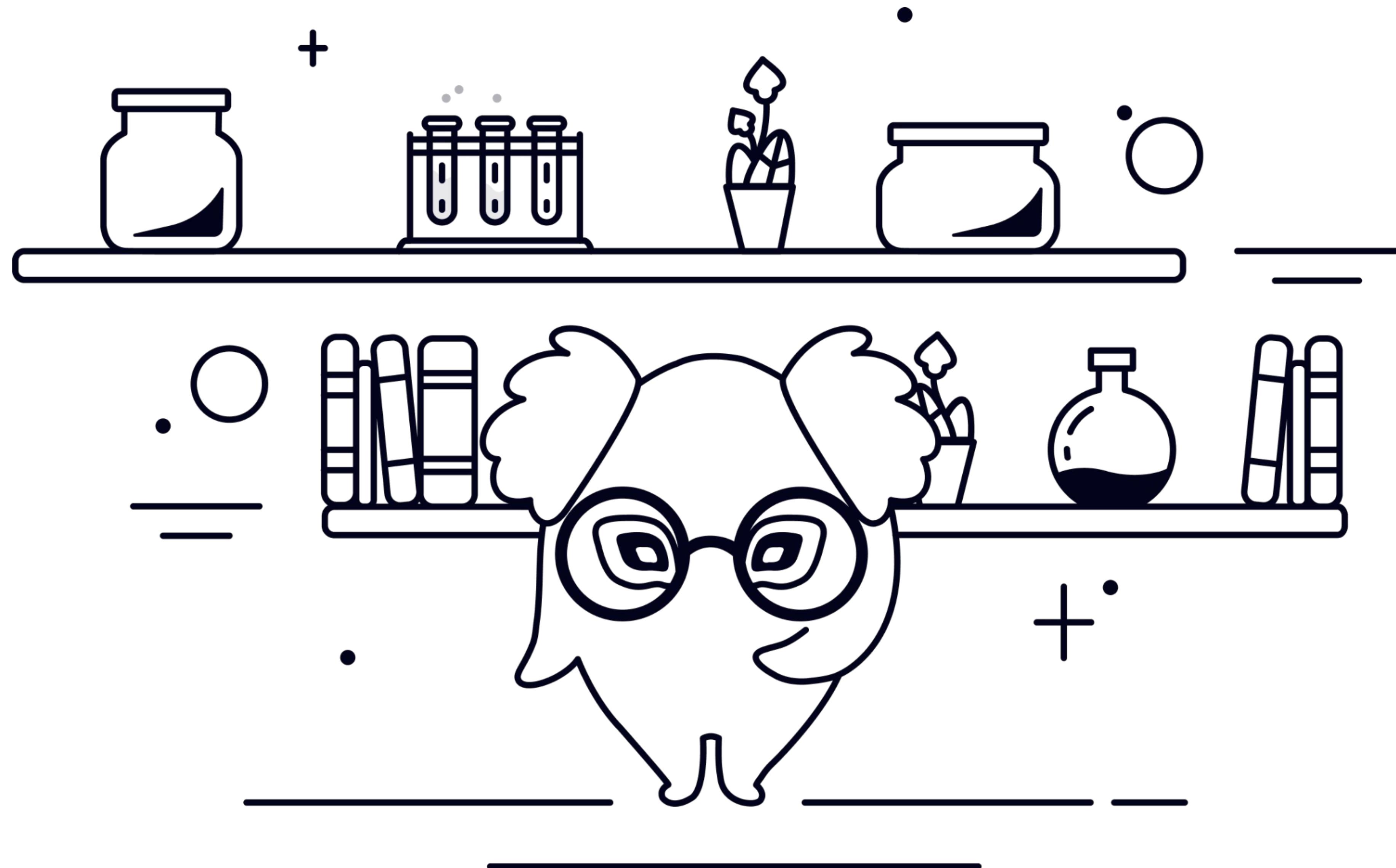
**Operations metrics under graphql-service** Over the last 5 minutes

Search operation

Name ▾ P95 Latency ▾ Request rate ▾ Error rate ▾ Impact ⓘ ▾

Name	P95 Latency	Request rate	Error rate	Impact
POST /graphql	4.49s	0.4 req/s	1%	Medium

< 1 > 20 / page ▾





# How can we detect performance issues?

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

**Choose service**

graphql-service

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

Last 5 minutes ▾

**Latency (s)**

— 95th   — 75th   — 50th

**Error rate (%)**

**Request rate (req/s)**

Is tracking the latency for a the single graphql endpoint good enough?

**Operations metrics under graphql-service** Over the last 5 minutes

Search operation

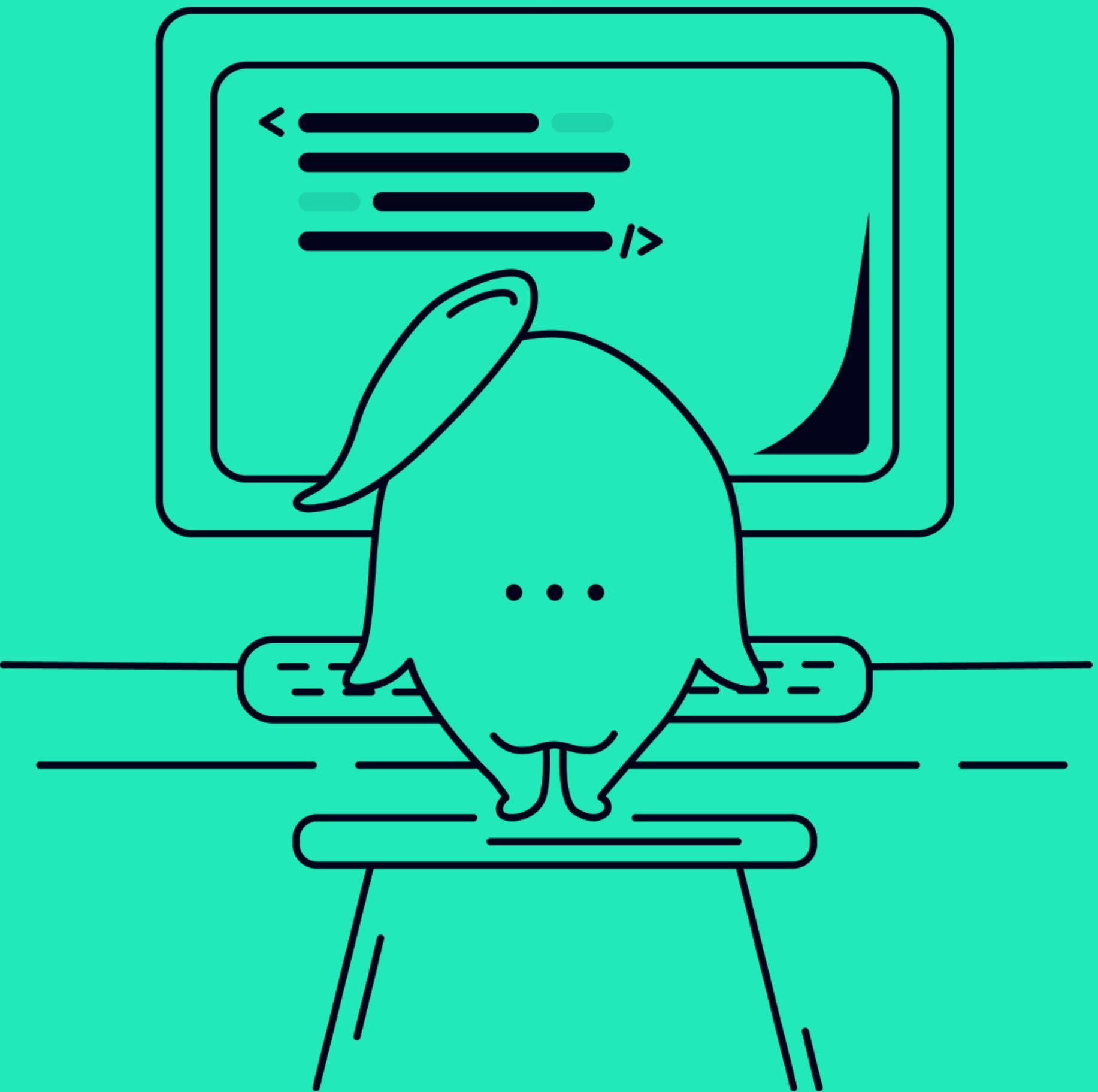
Name	P95 Latency	Request rate	Error rate	Impact
POST /graphql	4.49s	0.4 req/s	1%	Medium

< 1 > 20 / page ▾



# Performance issues

- 1 N+1
- 2 Cyclic queries
- 3 Query complexity
- 4 Query depth



# N+1 issue



Tyk       Explorer    Prettify    Docs

```
1 {  
2   continent (code: "OC") {  
3     countries {  
4       name  
5       weather {  
6         description  
7       }  
8     }  
9   }  
10 }  
11 }
```

The diagram illustrates the N+1 issue in GraphQL. On the left, a query is shown with a variable '1' highlighted in a purple box above the 'countries' field. A purple circle marks the 'countries' field, and a purple arrow points from it to the 'countries' field in the resulting JSON data on the right. On the left, a variable 'n' is highlighted in a teal box above the 'weather' field. A teal circle marks the 'weather' field, and four teal arrows point from it to the 'weather' fields in the resulting JSON data, representing multiple database queries for each country.

```
{  
  "data": {  
    "continent": {  
      "countries": [  
        {  
          "name": "American Samoa",  
          "weather": {  
            "description": "sunny"  
          }  
        },  
        {  
          "name": "Australia",  
          "weather": {  
            "description": "sunny"  
          }  
        },  
        {  
          "name": "Cook Islands",  
          "weather": {  
            "description": "snow"  
          }  
        },  
        {  
          "name": "Fiji",  
          "weather": {  
            "description": "rain"  
          }  
        },  
        {  
          "name": "Micronesia",  
          "weather": {  
            "description": "sunny"  
          }  
        },  
        {  
          "name": "Guam",  
          "weather": {  
            "description": "rain"  
          }  
        }  
      ]  
    }  
  }  
}
```

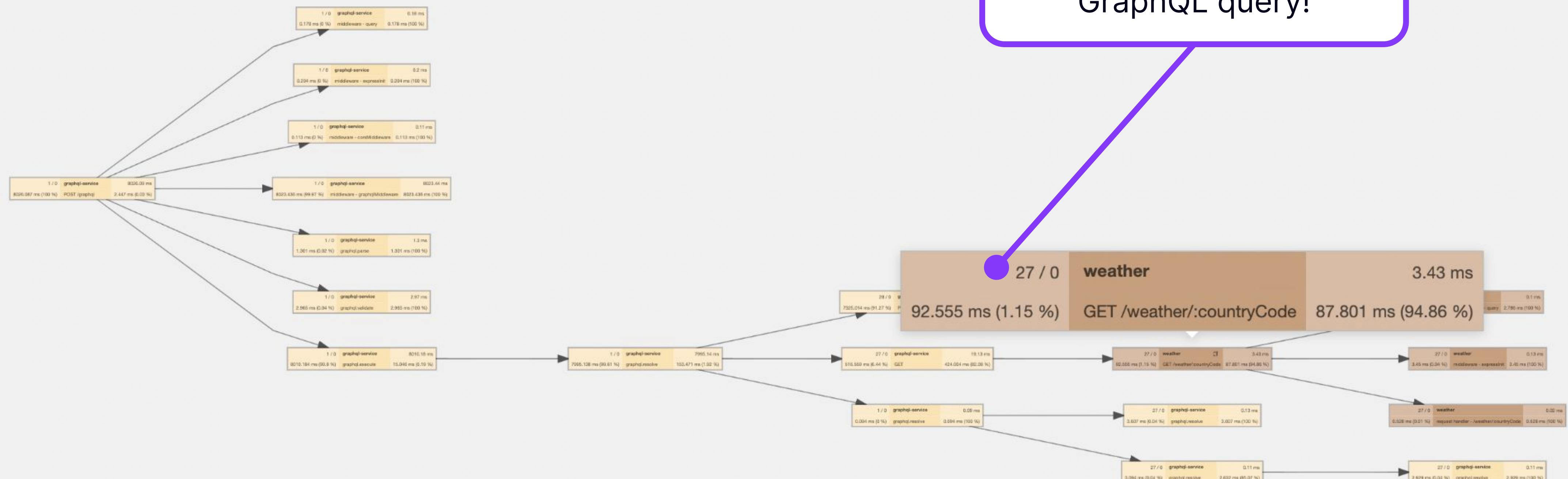


← ✓ graphql-service: POST /graphql 6a0e6d6

Find...

Trace Start **April 11 2023, 10:58:47.074** | Duration **8.03s** | Services **2** | Depth **6** | Total Spans **254**

## Experimental



27 HTTP GET calls to the weather service for this GraphQL query!

?

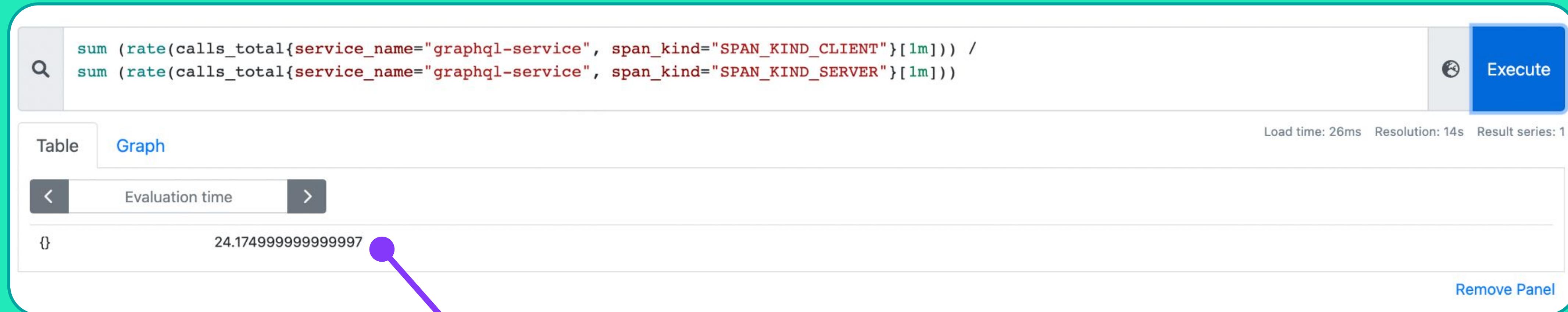
S

T

ST



# Detecting N+1 queries



On average, 1 GraphQL query makes 24 outgoing requests

# Cyclic queries



Tyk ▶ Explorer Prettify

```
1 {  
2   continents {  
3     countries {  
4       continent {  
5         countries {  
6           name  
7         }  
8       }  
9     }  
10    }  
11  }  
  
{  
  "data": {  
    "continents": [  
      {  
        "countries": [  
          {  
            "continent": {  
              "countries": [  
                {  
                  "name": "Angola"  
                },  
                {  
                  "name": "Burkina Faso"  
                },  
                {  
                  "name": "Burundi"  
                },  
                {  
                  "name": "Benin"  
                },  
                {  
                  "name": "Botswana"  
                },  
                {  
                  "name": "Democratic Republic of the Congo"  
                },  
                {  
                  "name": "Central African Republic"  
                },  
                {  
                  "name": "Republic of the Congo"  
                },  
                {  
                  "name": "Ivory Coast"  
                },  
                {  
                  "name": "Cameroon"  
                }  
              ]  
            }  
          ]  
        }  
      ]  
    }  
  }  
}
```

QUERY VARIABLES REQUEST HEADERS

Applica[REDACTED]

# Expensive Queries

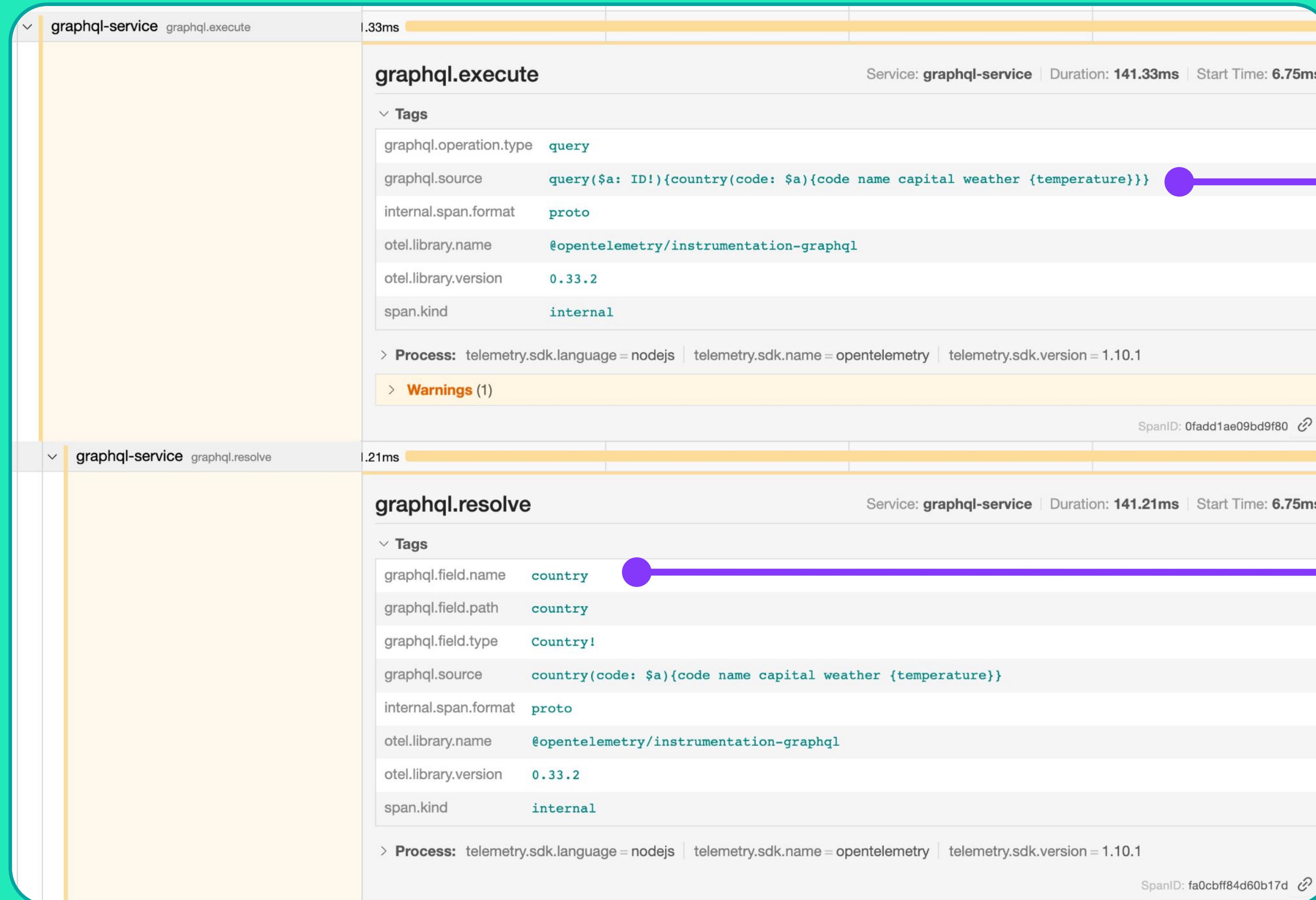
Depth 3

```
{  
  a {  
    b {  
      c  
    }  
  }  
}
```

Complexity 6

```
{  
  a b c {  
    d e {  
      f  
    }  
  }  
}
```

# What information do we have on our spans?



Full query (doesn't match the semantic conventions)

Field name (doesn't match the semantic conventions either)

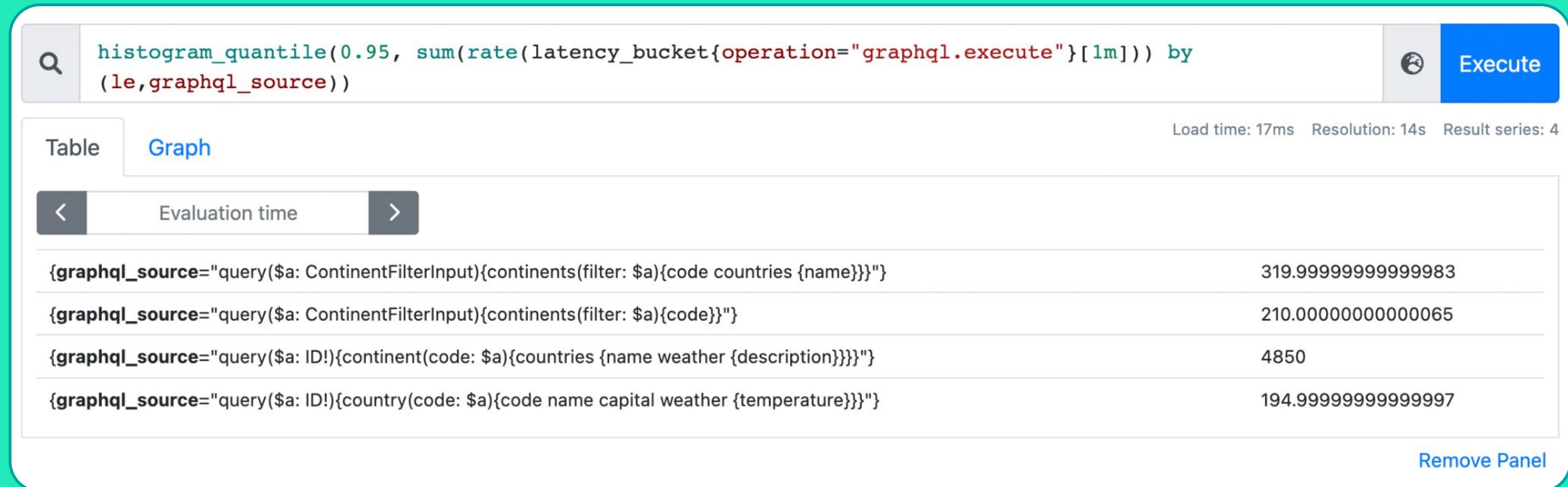
OpenTelemetry collector  
configuration to export  
additional span metrics

```
21 processors:  
22   batch:  
23     spanmetrics:  
24       dimensions:  
25         name: graphql.source  
26           - name: graphql.field.name  
27           - name: graphql.field.path  
28     metrics_exporter: prometheus  
29  
30   service:  
31     pipelines:  
32       traces:  
33         receivers: [otlp]  
34         processors: [spanmetrics, batch]  
35         exporters: [otlp]  
36       metrics:  
37         receivers: [otlp]  
38         processors: [batch]  
39         exporters: [prometheus]  
40
```





# Let try to group them by query





# Using operation type and name

Span Status MUST be left unset if HTTP status code was in the 1xx, 2xx or 3xx ranges, unless there was another error (e.g., network error receiving the response body; or 3xx codes with max redirects exceeded), in which case status MUST be set to Error.

Attribute	Type	Examples
graphql.operation.name	string	findBookById
graphql.operation.type	string	query ; mutation ; subscription
graphql.document	string	query findBookById { bookById(id: ?) { name } }

but not available in our instrumentation library

but not available in our instrumentation library



# Using operation type and name

query continents	300 ms
query continents	2400 ms
mutation updateCountry	500 ms



# Adding a client identification

query continents mobile	600 ms
query continents react	3500 ms
query continents weather-app	550 ms



# What have we learnt?

1

OpenTelemetry is helpful for monitoring and troubleshooting GraphQL queries, BUT:

2

Semantic conventions are not always respected by instrumentation libraries

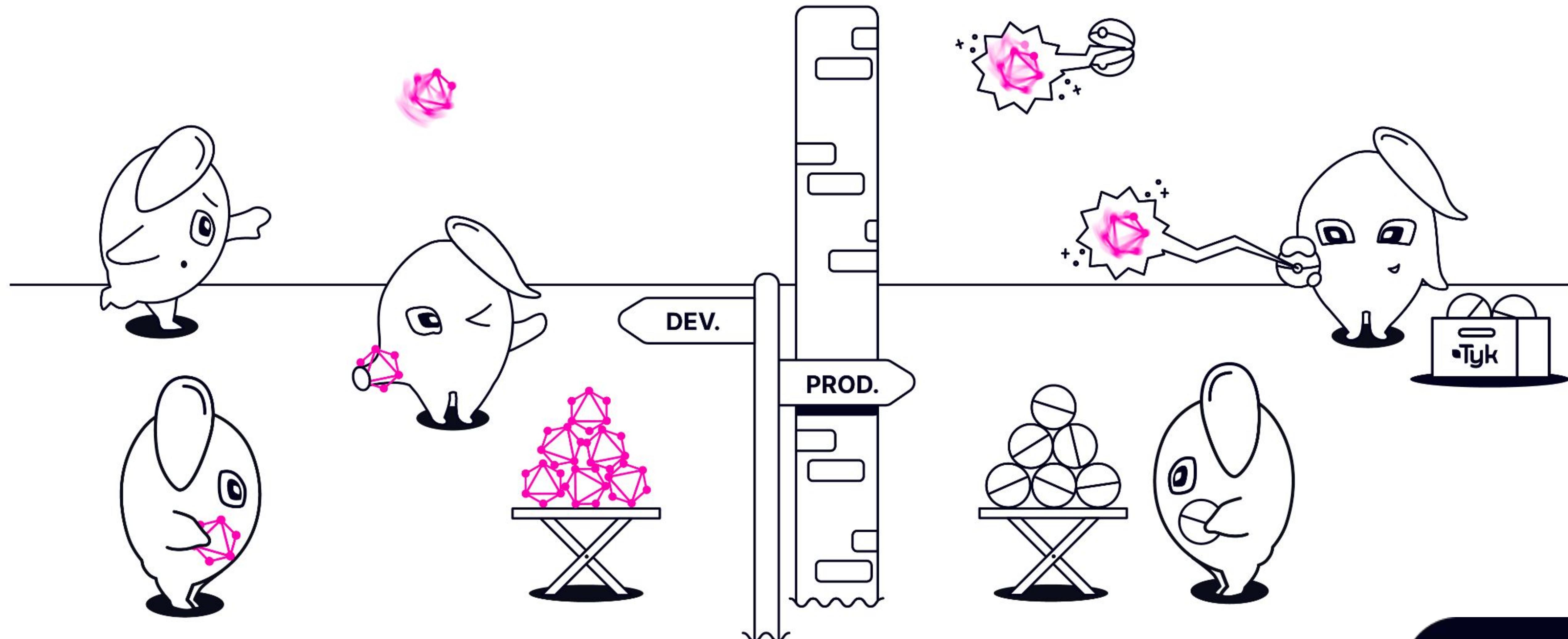
3

GraphQL errors are missing from semantic conventions and from instrumentation

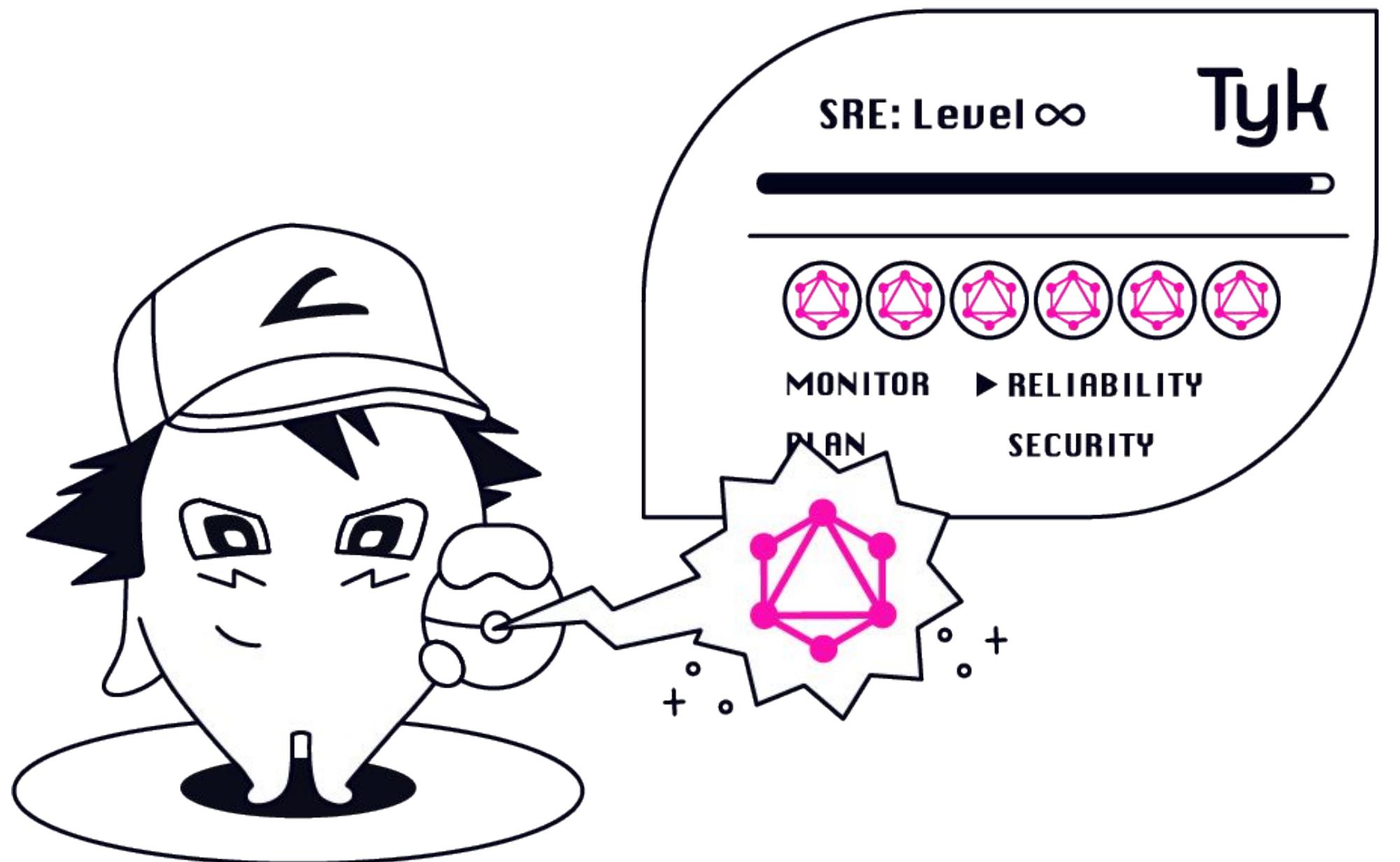
- needed to detect for errors
- and for error based sampling

4

RED metrics approach of monitoring needs to be extended to report GraphQL specifics  
(query type, operation name, query depth, query complexity)



•Tyk



# Thank you!

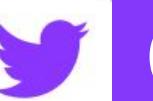
Come talk to us to continue the discussion  
or reach out:

■ **Sonja Chevre**

 @SonjaChevre

 [linkedin.com/in/sonjachevre](https://linkedin.com/in/sonjachevre)

■ **Ahmet Soormally**

 @SoormallyAhmet

 [linkedin.com/in/ahmet-soormally](https://linkedin.com/in/ahmet-soormally)