The road to observable systems

January 2022

Hi, I'm Toli.

Stumbled into software engineering

Worked in the software industry for 10+ years

Experience across a few industries (Logistics, Energy, Automotive)

Touched all traditional tech stacks

cinch

My current role

Principal Practice Engineer at cinch

Advocating for good DevOps and o11y patterns & practices

cinch

Cinch offers the UK's biggest range of cars to buy entirely online

We remove the faff from finding, buying and owning a car

We don't just let people nail it when they buy a new car

cinch

We make sure they absolutely cinch it.

Hence the name*

cinch[1] [sinch] SHOW IPA

See synonyms for: cinch / cinched / cinching on Thesaurus.com

High School Level

noun

1 a strong girth used on stock saddles, having a ring at each end to which a strap running from the saddle is secured.

2 a firm hold or tight grip.

3 *Informal.*

a something sure or easy:

This problem is a cinch.

b a person or thing certain to fulfill an expectation, especially a team or contestant certain to win a sporting event:

The Giants are a cinch to win Sunday's game.

*https://www.dictionary.com/browse/cinch

cinch

The road to observable systems

Why did we focus so much on observability?

What impact did this have on our organization and engineering practices?

What lessons did we learn along the way?

cinch

Tech stack

TypeScript

cinch

Engineering principles

Serverless, Event Driven Architecture	Build serverless, asynchronous, resilient systems to serve team autonomy
Test Driven Development	Automate as much testing as possible
Flow	Fast flow, small batches, frequent changes
Feedback	Build observable systems for optimal feedback

cinch

Engineering principles

Serverless, Event Driven Architecture	Build asynchronous, resilient systems to serve team autonomy
Test Driven Development	Automate as much testing as possible
Flow	Fast flow, small batches, frequent changes
Feedback	Build observable systems for optimal feedback

cinch

Engineering principles

Serverless, Event Driven Architecture	Build asynchronous, resilient systems to serve team autonomy
Test Driven Development	Automate as much testing as possible
Flow	Fast flow, small batches, frequent changes
Feedback	Build observable systems for optimal feedback

cinch

Engineering principles

Serverless, Event Driven Architecture	Build asynchronous, resilient systems to serve team autonomy
Test Driven Development	Automate as much testing as possible
Flow	Fast flow, small batches, frequent changes
Feedback	Build observable systems for optimal feedback

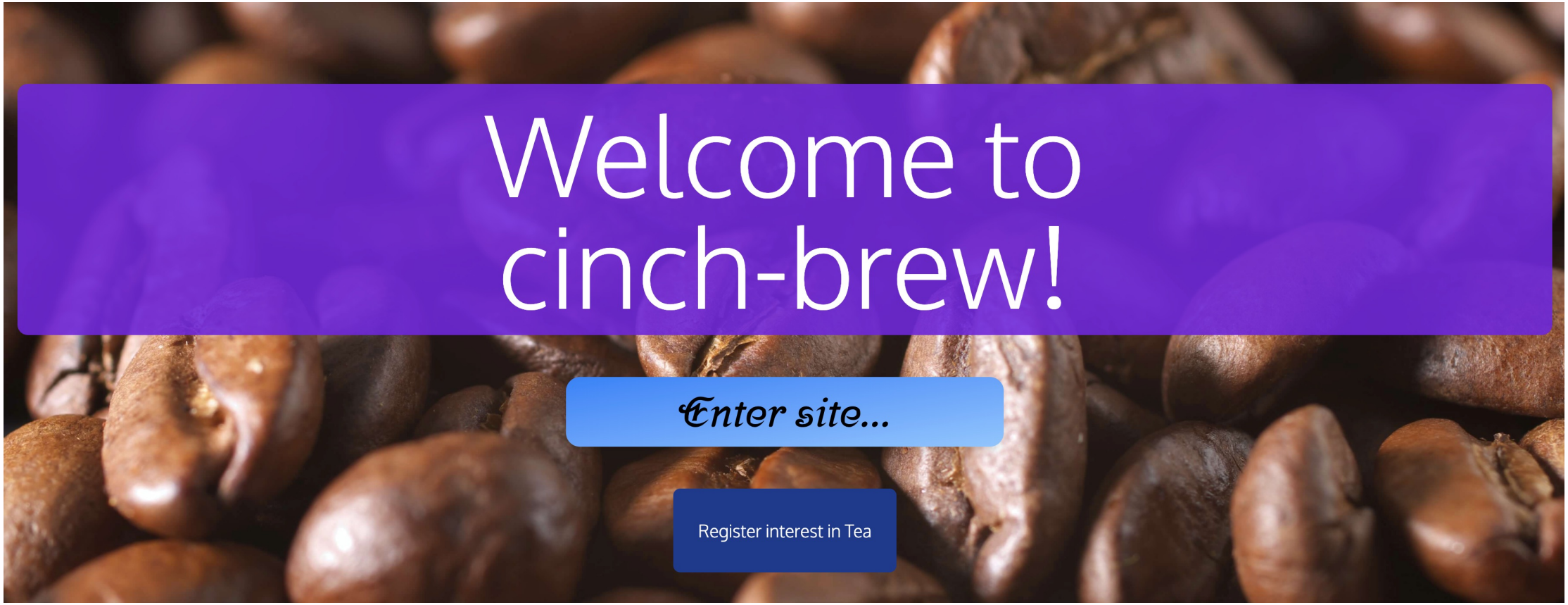
cinch

Serverless scales beautifully 👏

What does serverless look like?

cinch

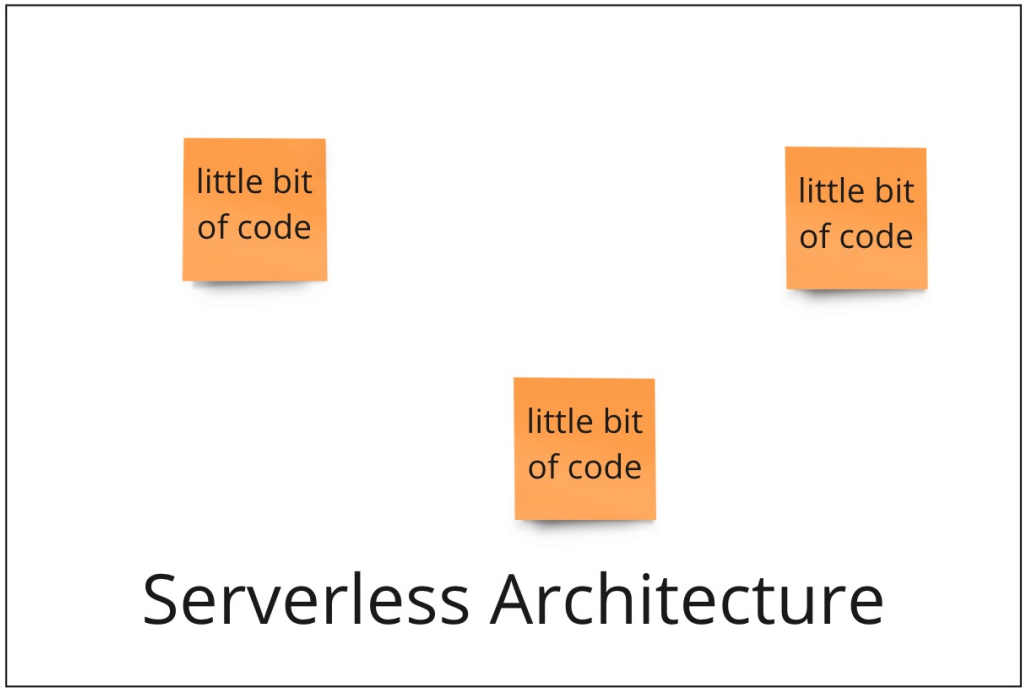
This is not real, don't get excited!*



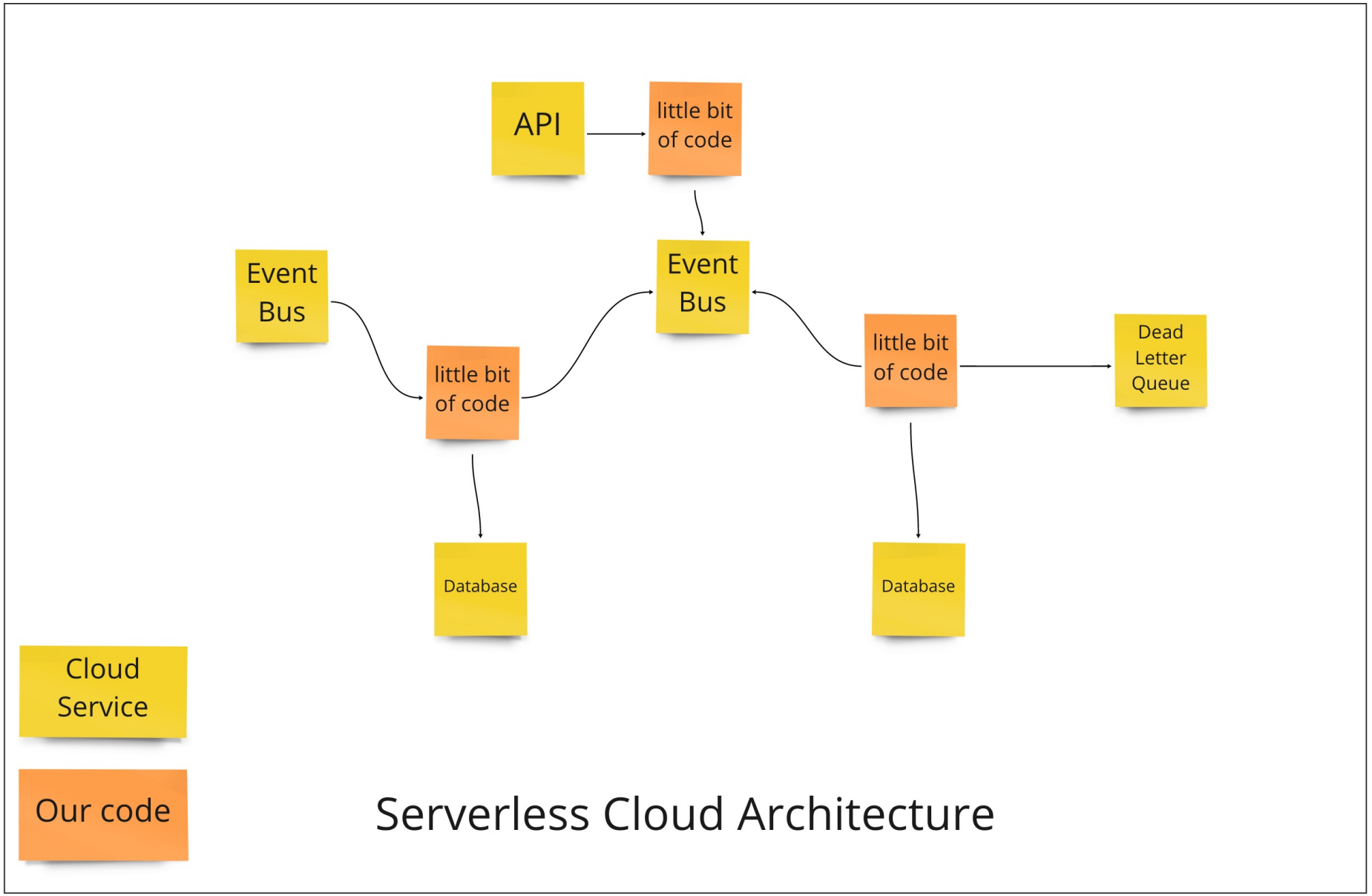
*We're going to use a fictional software system to take you on our serverless observability journey – it's called 'cinch brew'

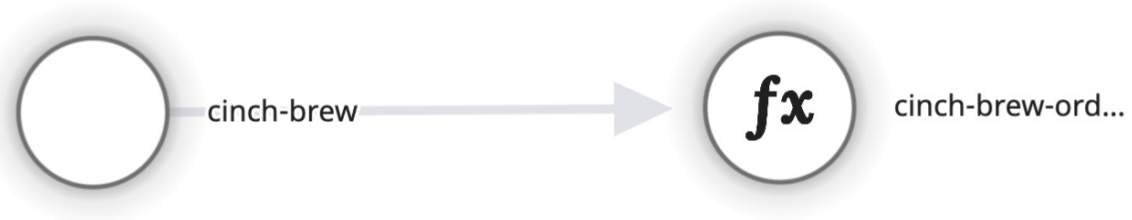
cinch

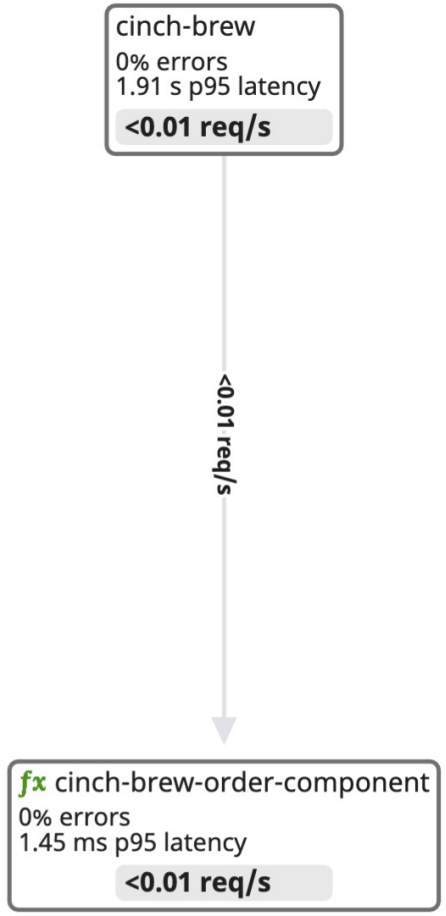
Little bits of code here & there

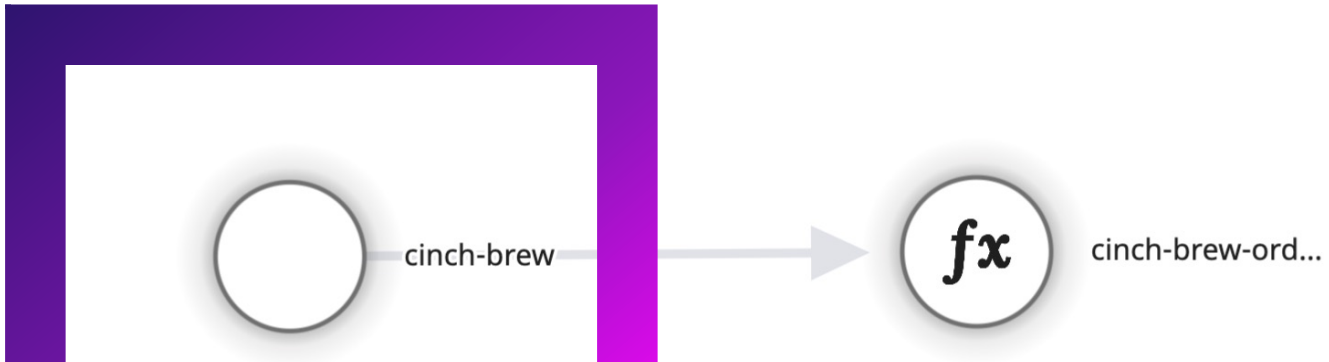


cinch

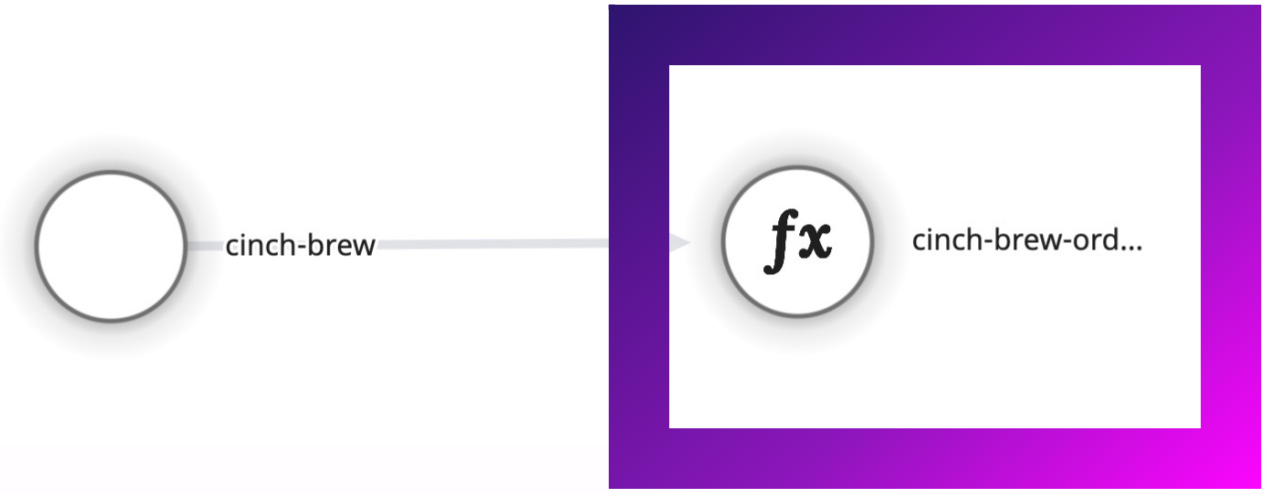

API
little bit of code
Event Bus
Event Bus
little bit of code
little bit of code
Dead Letter Queue
Database
Database
Cloud Service
Our code
Serverless Cloud Architecture


cinch-brew
cinch-brew-ord...

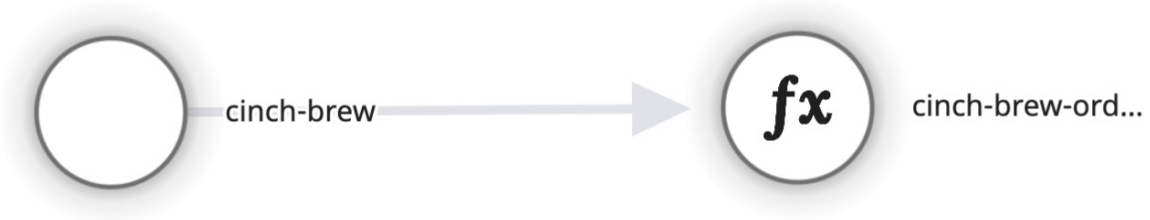

cinch-brew
0% errors
1.91 s p95 latency
<0.01 req/s
<0.01 req/s
cinch-brew-order-component
0% errors
1.45 ms p95 latency
<0.01 req/s



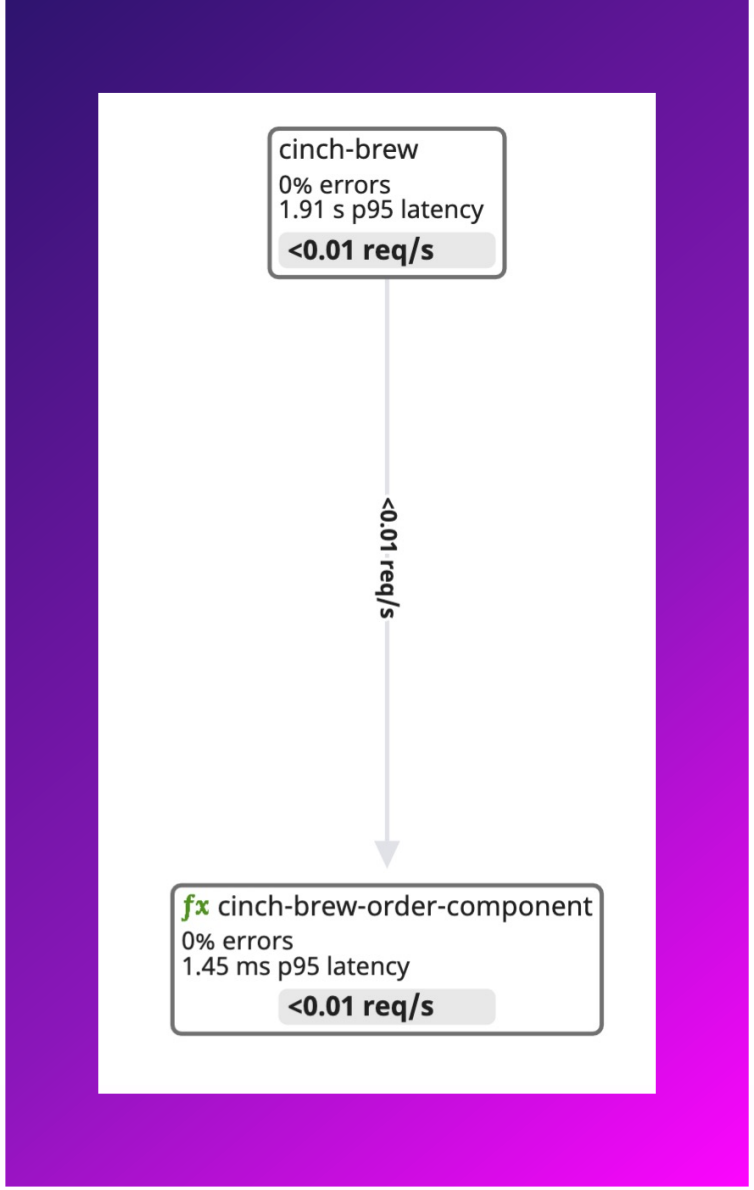
Frontend

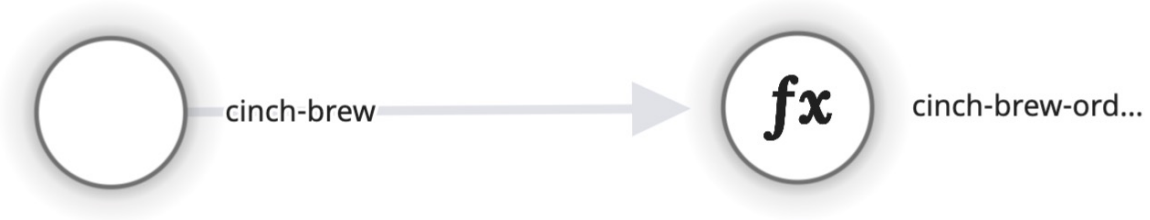


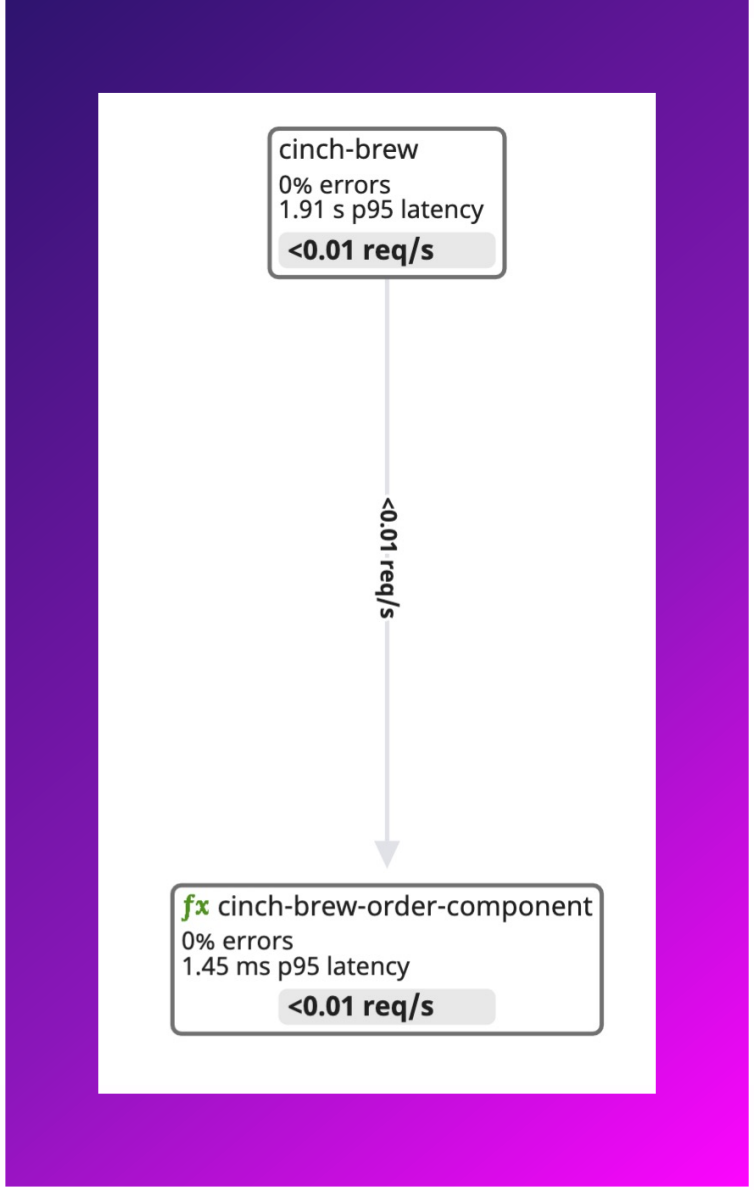
Backend



Requests, Errors, Duration

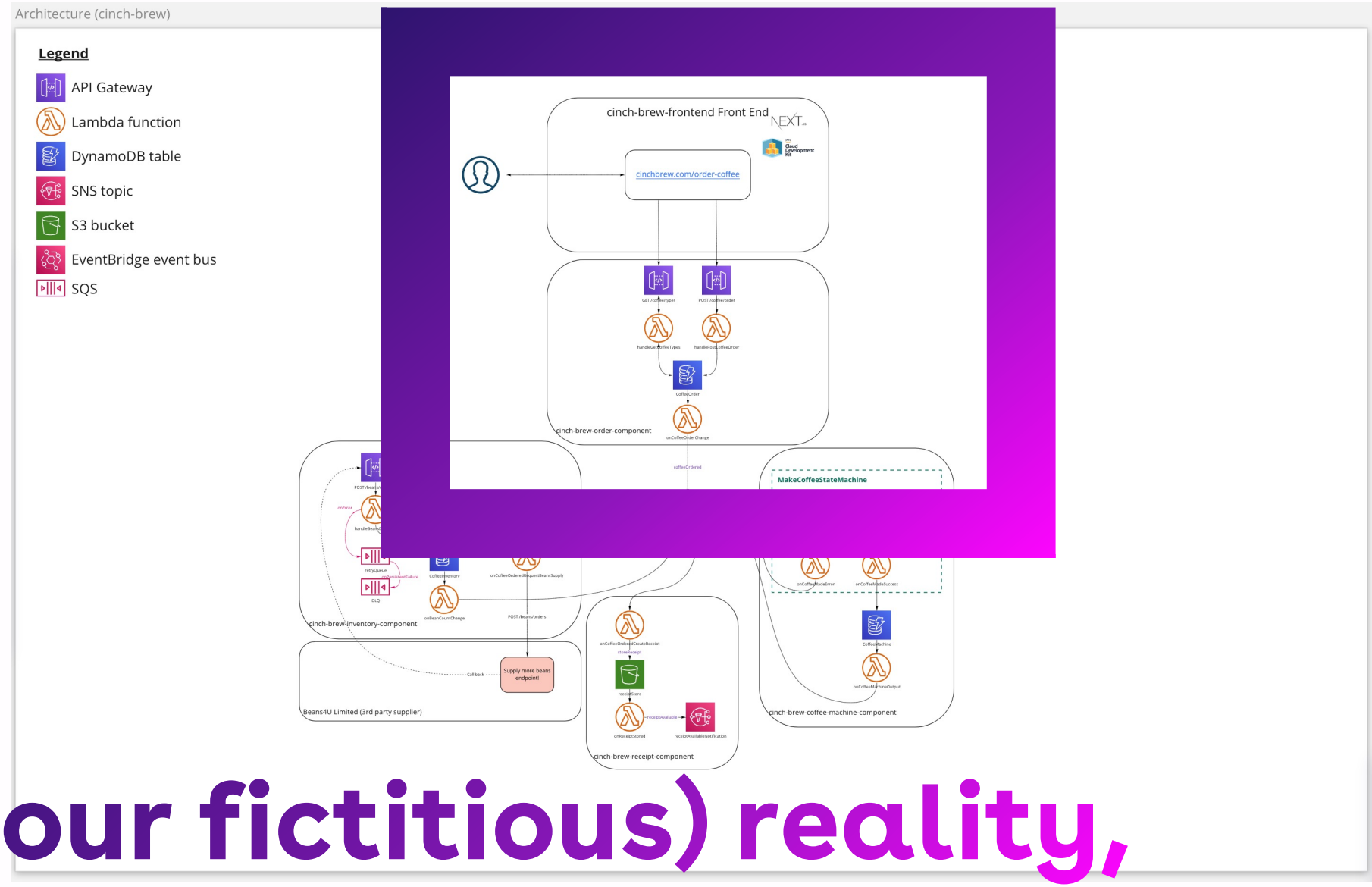






Serverless observability is simple, right?

cinch



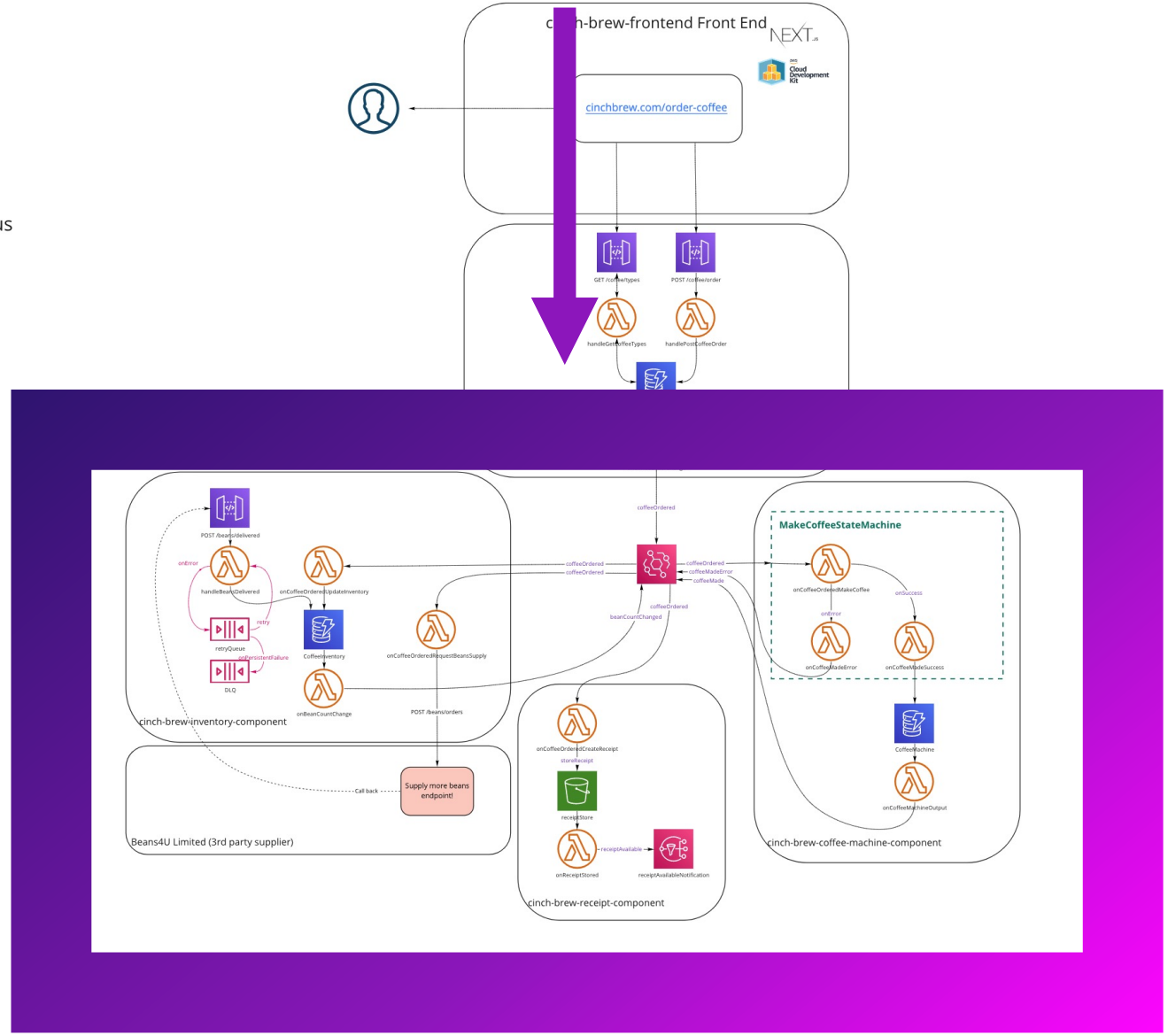
In (our fictitious) reality, systems are more complex

Architecture (cinch-brew)

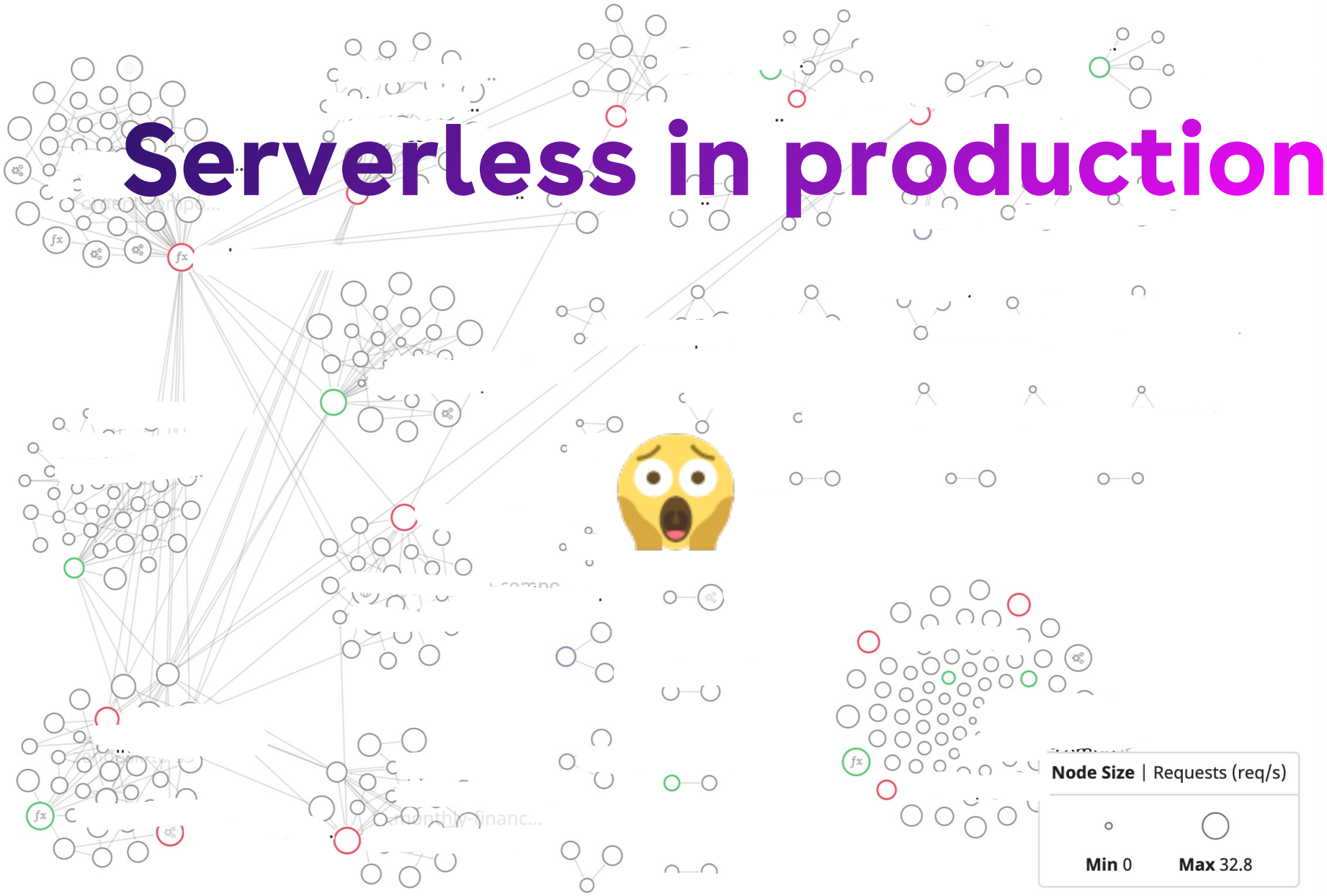
There is this bit too

Legend

- API Gateway
- Lambda function
- DynamoDB table
- SNS topic
- S3 bucket
- EventBridge event bus
- SQS



Serverless in production



cinch

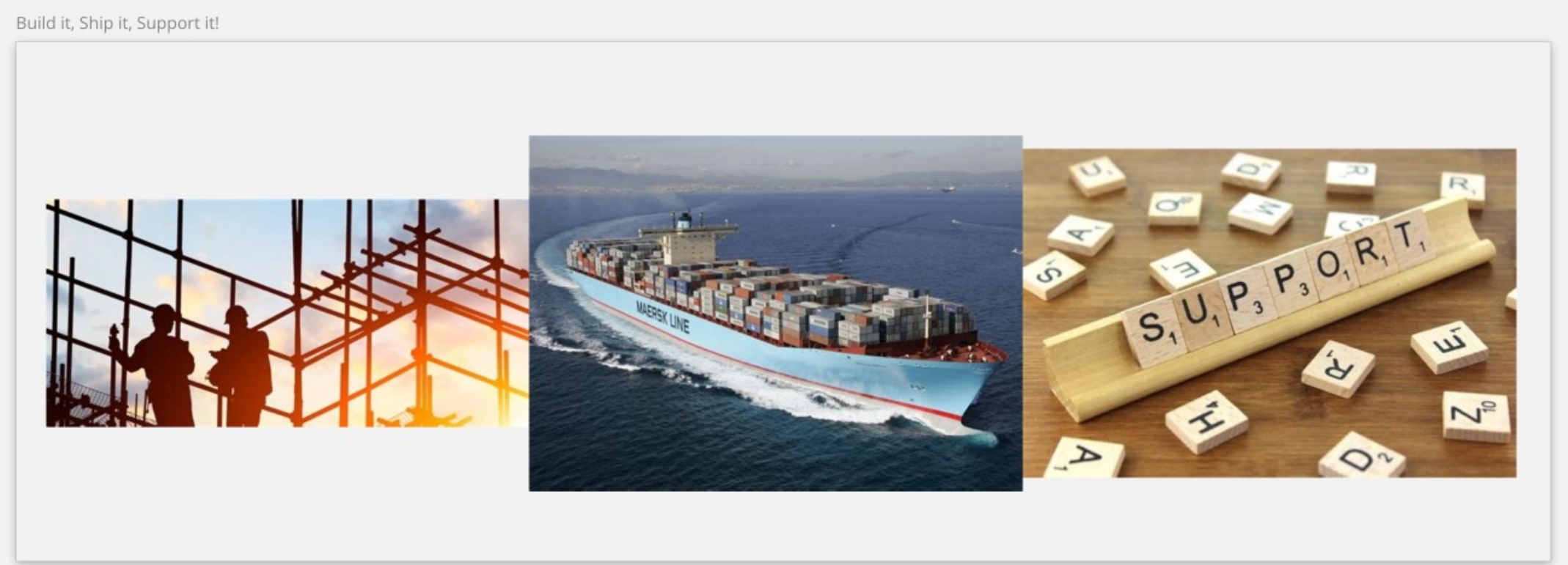
Can you see faff creeping in?

We don’t do faff. 

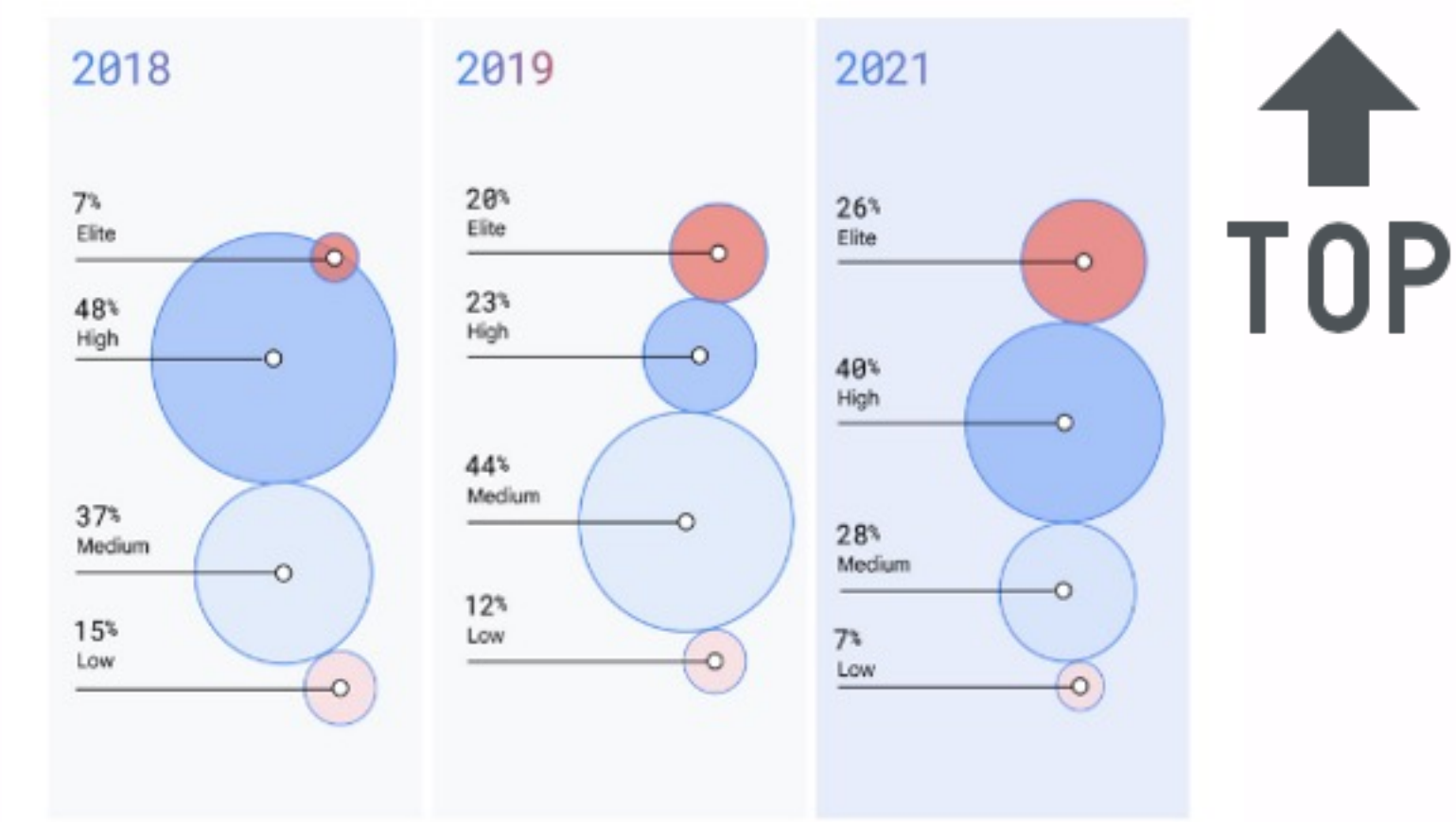
cinch

How do we know our services are keeping their promises?

Our model:
Build it, Ship it, Support it



The industry has advanced



https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

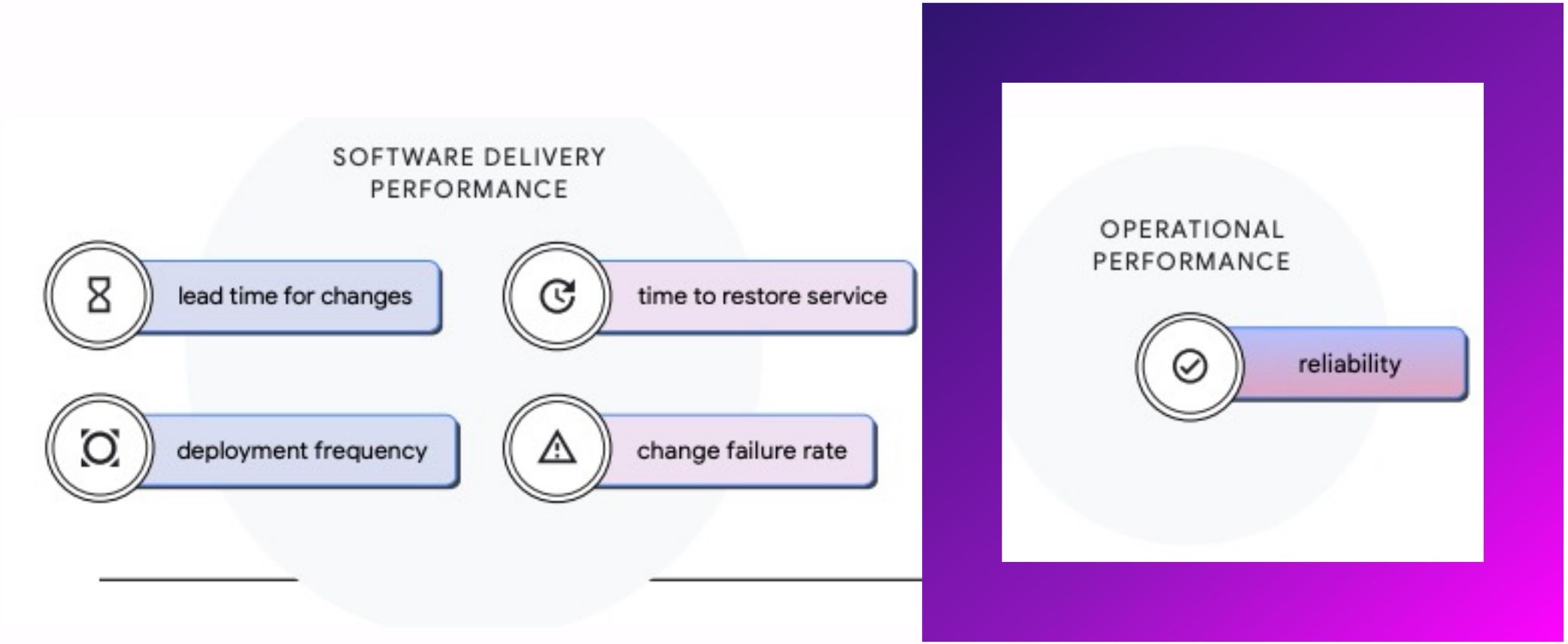
cinch

Serverless means we are elite, right?

Build it, Ship it, Support it!

There is a fifth Accelerate metric

The fifth metric: Reliability*



* Reliability does not equal observability. But serverless helps. And observability helps too.

cinch

Reliability = serverless + o11y*

*observability

cinch

Glossary

Term	Definition
Observability	“Observability” is being able to fully understand our systems. In control theory, observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. Honeycomb coined the term in software from control theory and has popularised it across the industry.

Charity Majors, Honeycomb CEO

CHARITY.WTF

OBSERVABILITY IS A MANY-SPLENDORED DEFINITION

March 3, 2020

mipsytipsy

apm, fight fight fight, instrumentation, logs, metrics, monitoring, traces

Last weekend, @swyx posted a great little primer to instrumentation titled "Observability Tools in JavaScript". A friend sent me the link and suggested that I might want to respond and clarify some things about observability, so I did, and we had a great conversation! Here is a lightly edited transcript of my reply tweet storm.

First of all, confusion over terminology is understandable, because there are some big players out there actively trying to confuse you! Big Monitoring is indeed actively trying to define observability down to "metrics, logs and traces". I guess they have been paying attention to the interest heating up around observability, and well... they have metrics, logs, and tracing tools to sell? So they have hopped on the bandwagon with some undeniable zeal.

But metrics, logs and traces are just data types. Which actually has nothing to do with observability. Let me explain the difference, and why I think you should care about this.

"OBSERVABILITY? I DO NOT THINK IT MEANS WHAT YOU THINK IT MEANS."

Observability is a borrowed term from mechanical engineering/control theory. It means, paraphrasing: "can you understand what is happening inside the system — can you understand ANY internal state the system may get itself into, simply by asking questions from the outside?" We can apply this concept to software in interesting ways, and we may end up using some data types, but that's putting the cart before the horse.

It's a bit like saying that "database replication means structs, longints and elegantly diagrammed English sentences." Er, no.. yes.. missing the point much?

https://charity.wtf/2020/03/03/observability-is-a-many-splendored-thing/

cinch

Observable systems are built by teams

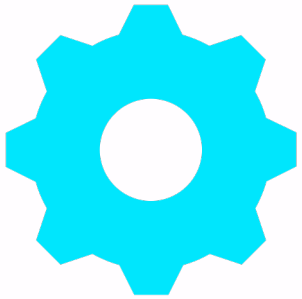
cinch

Instrument, don’t log

Instrumentation	"Instrumentation" is the practice of systematically configuring our codebase to emit telemetry data that provide insights into how our code is exercised by data in production.

cinch

You can auto-instrument, be more ambitious



Auto-instrumentation

Custom instrumentation

Sadly, there are many telemetry data types

Traces

Metrics

Logs

RUM Events

We use logs, but minimally

Traces

high context & traceability

Metrics

high value signals with limited context

Logs

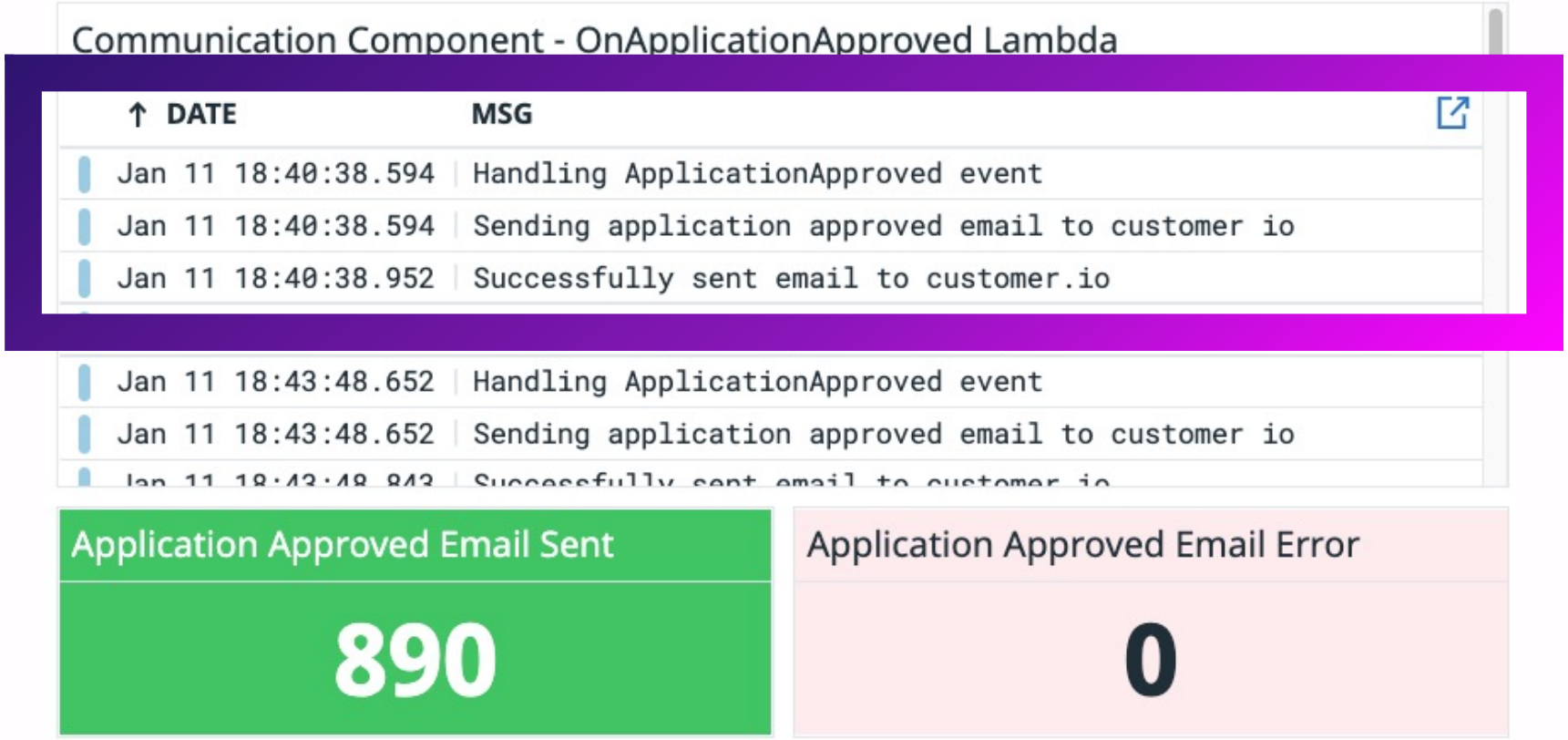
high context with free text included & links to traces

RUM Events

high context & links to traces

cinch

Logs are useful for non techies



They can be turned into numbers too

Communication Component - OnApplicationApproved Lambda

↑ DATE	MSG
Jan 11 18:40:38.594	Handling ApplicationApproved event
Jan 11 18:40:38.594	Sending application approved email to customer io
Jan 11 18:40:38.952	Successfully sent email to customer.io
Jan 11 18:40:38.953	Successfully sent application approved email to customer io
Jan 11 18:43:48.652	Handling ApplicationApproved event
Jan 11 18:43:48.652	Sending application approved email to customer io

Application Approved Email Sent

890

pplication Approved Email Error

0

Our favourite type: traces (and spans)

Traces

high context & traceability

Metrics

high value signals with limited context

Logs

high context with free text included & links to traces

RUM Events

high context & links to traces

cinch

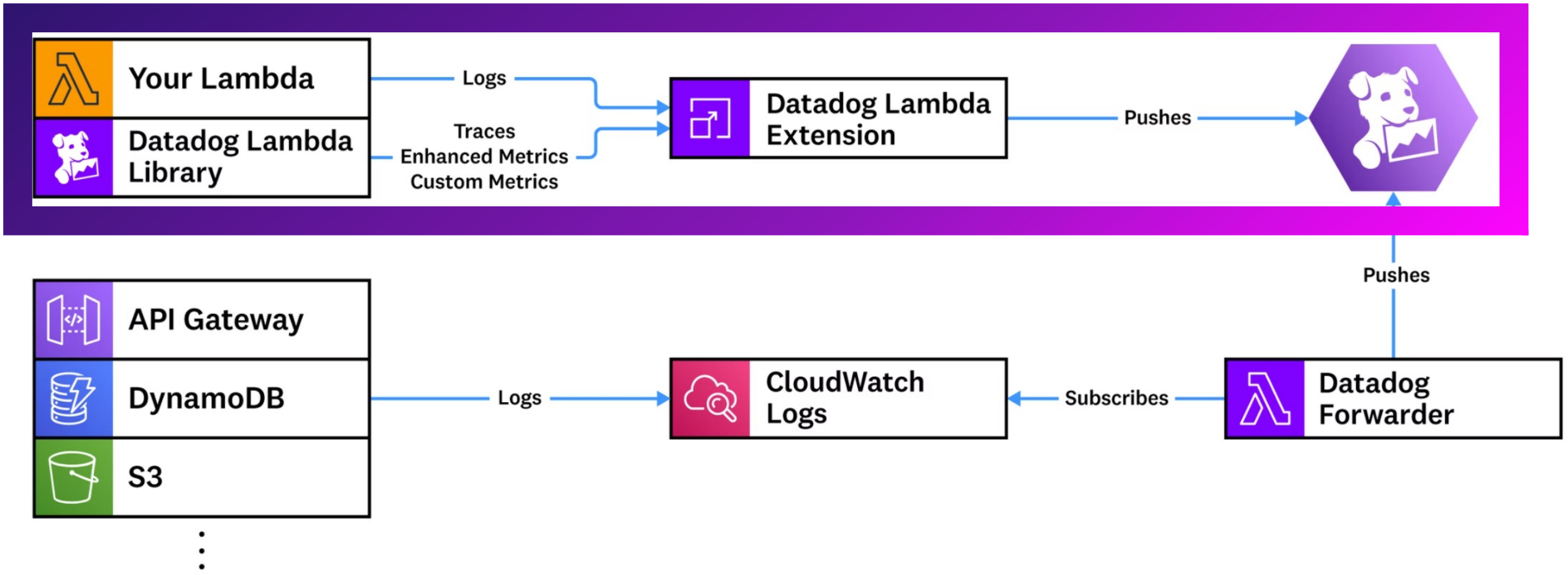
Auto-instrumentation

```
64          const datadog = new Datadog(this, 'Datadog', {
65              addLayers: true,
66              apiKey: props.datadogApiKey,
67              extensionLayerVersion: 12,
68              site: 'datadoghq.eu',
69              nodeLayerVersion: 64,
70          });
```

```
153         datadog.addLambdaFunctions([listCoffeeBeanTypes]);
```

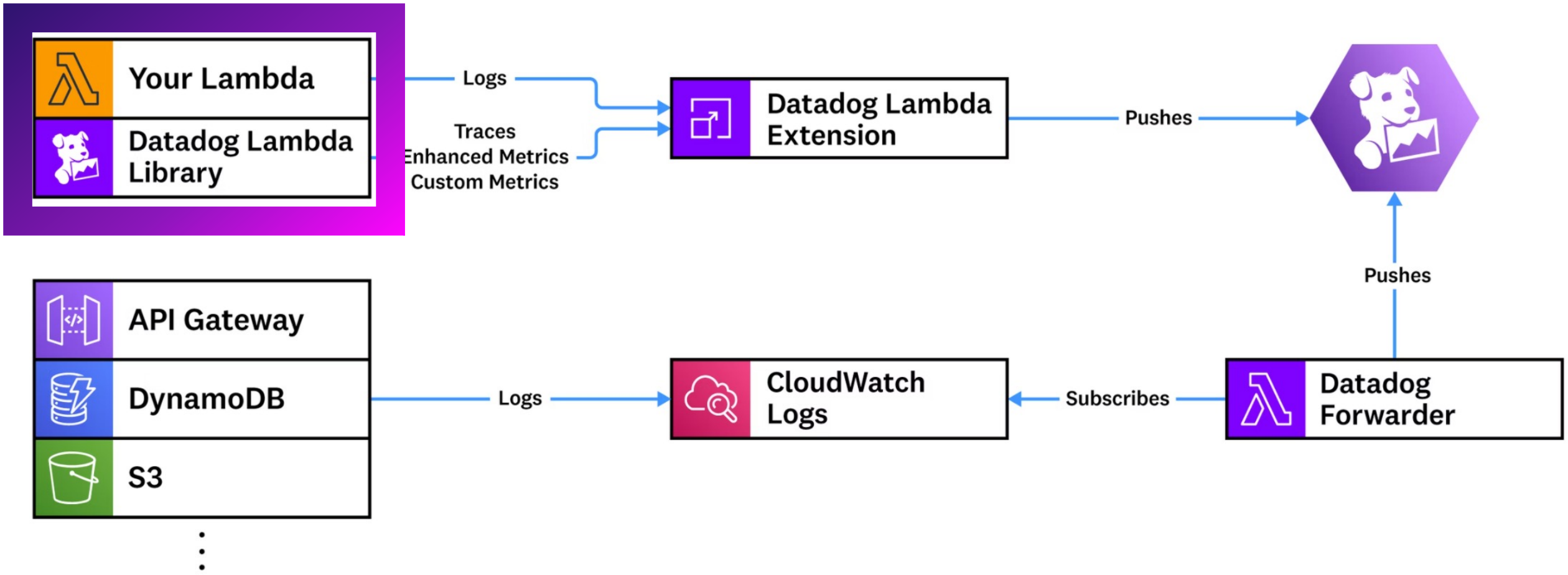
cinch

Hook into lambda runtime



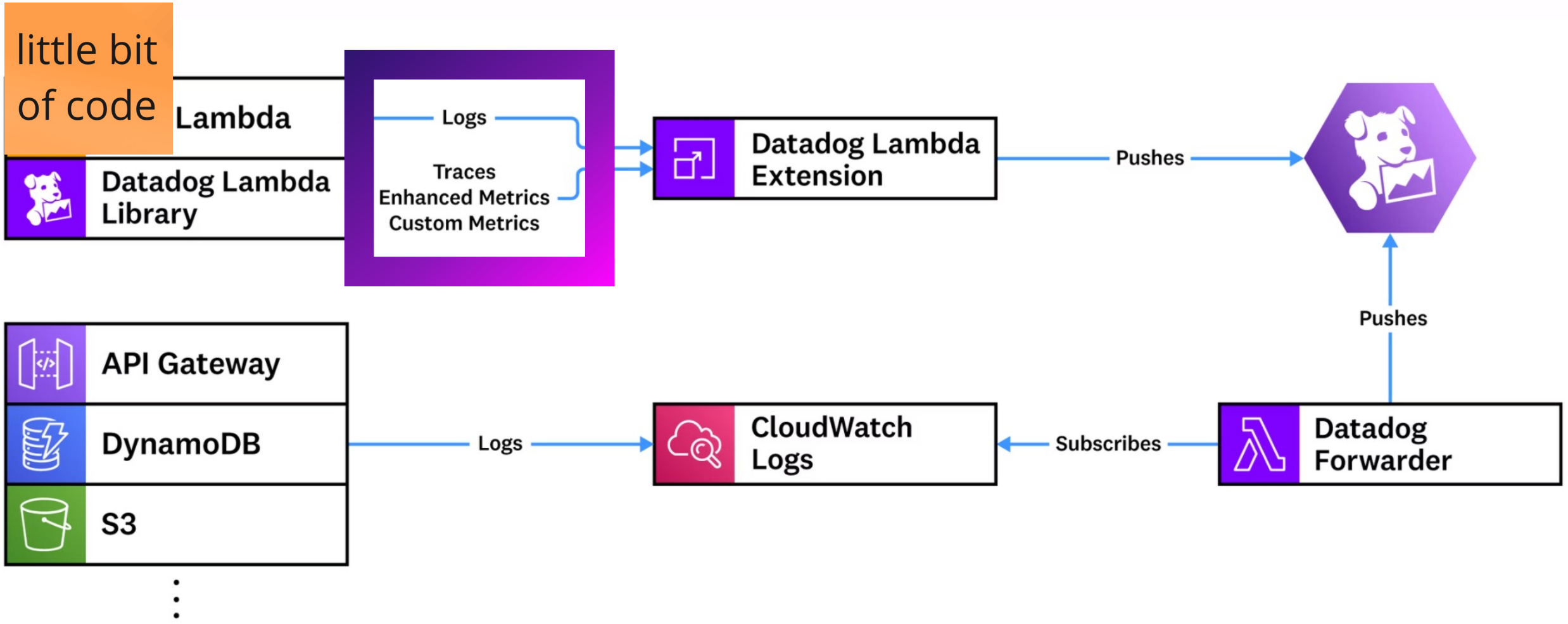
cinch

We extend our lambdas

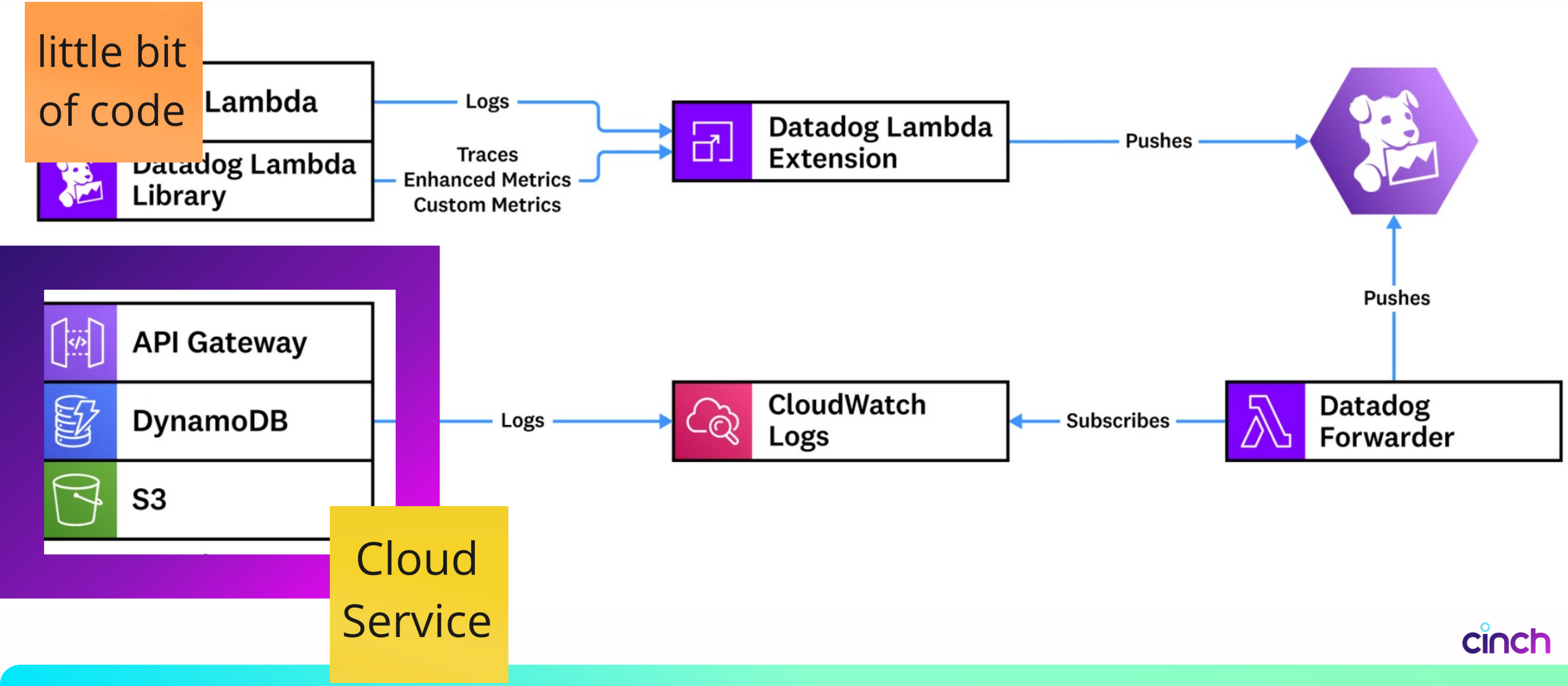


cinch

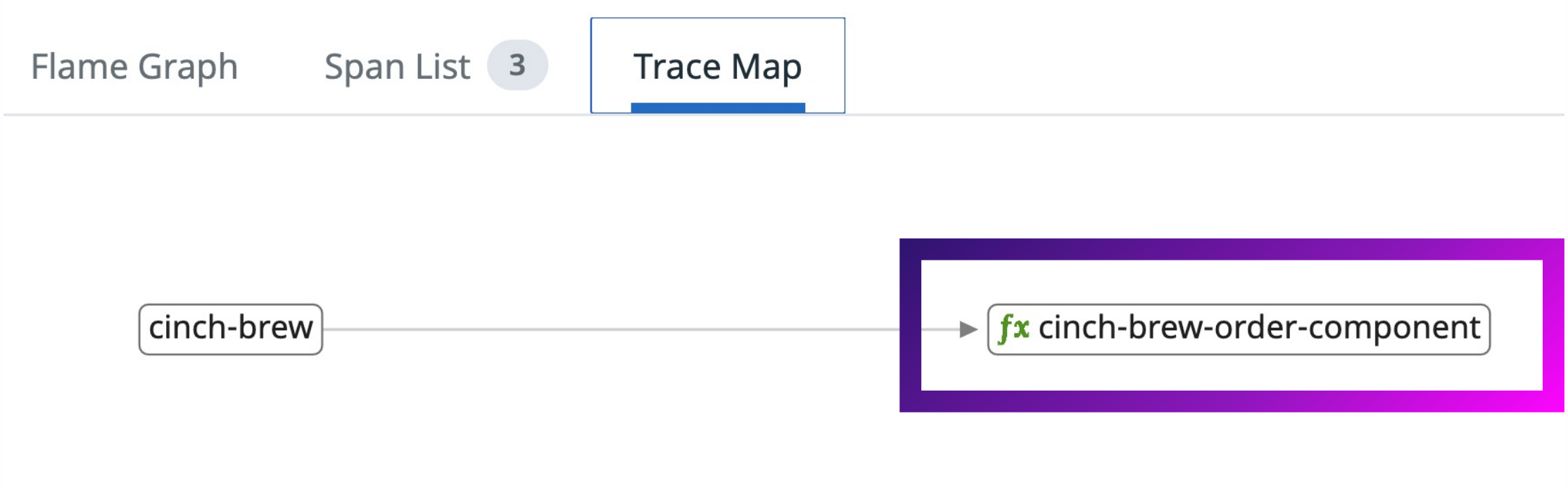
Who needs CloudWatch?



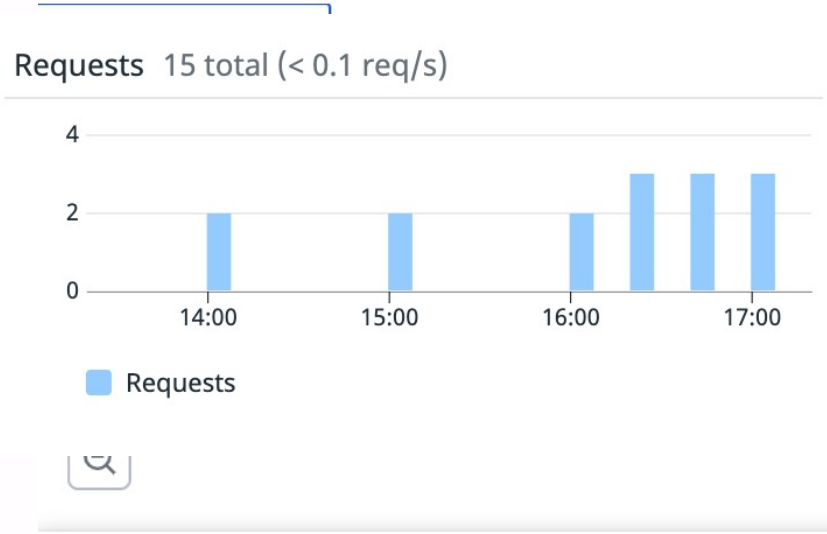
We pull logs for Cloud services

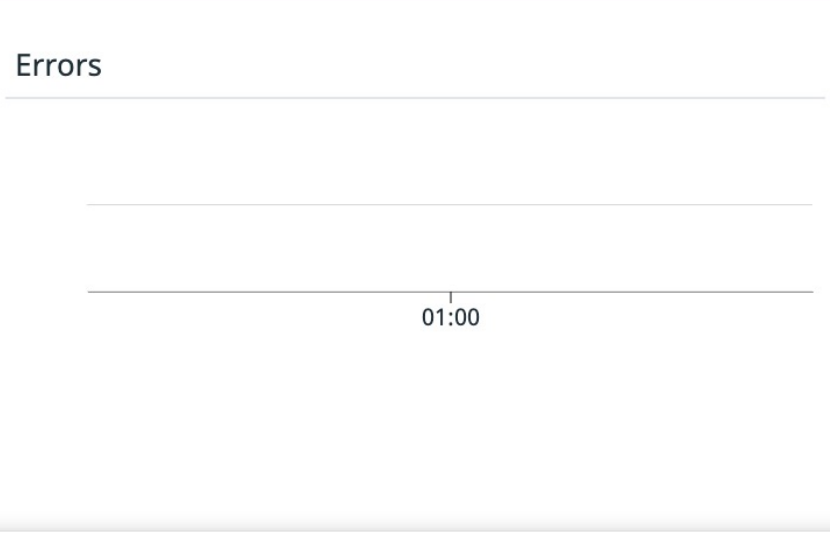


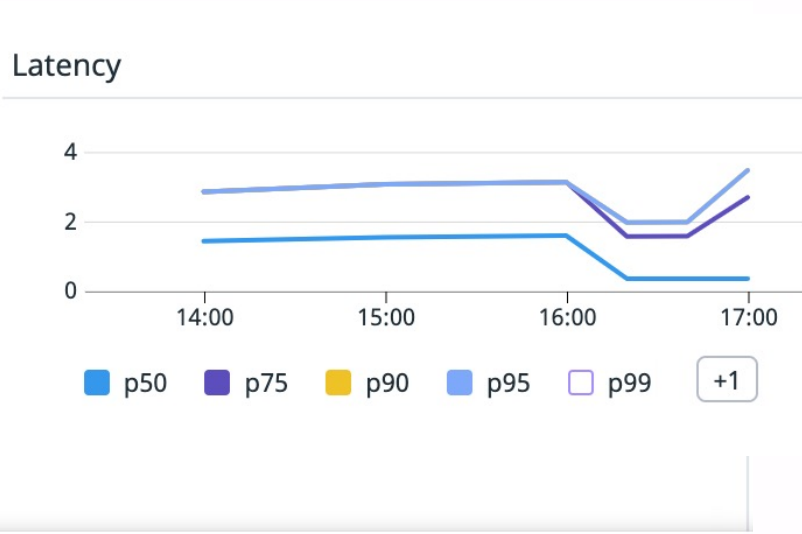
Out of the box trace map



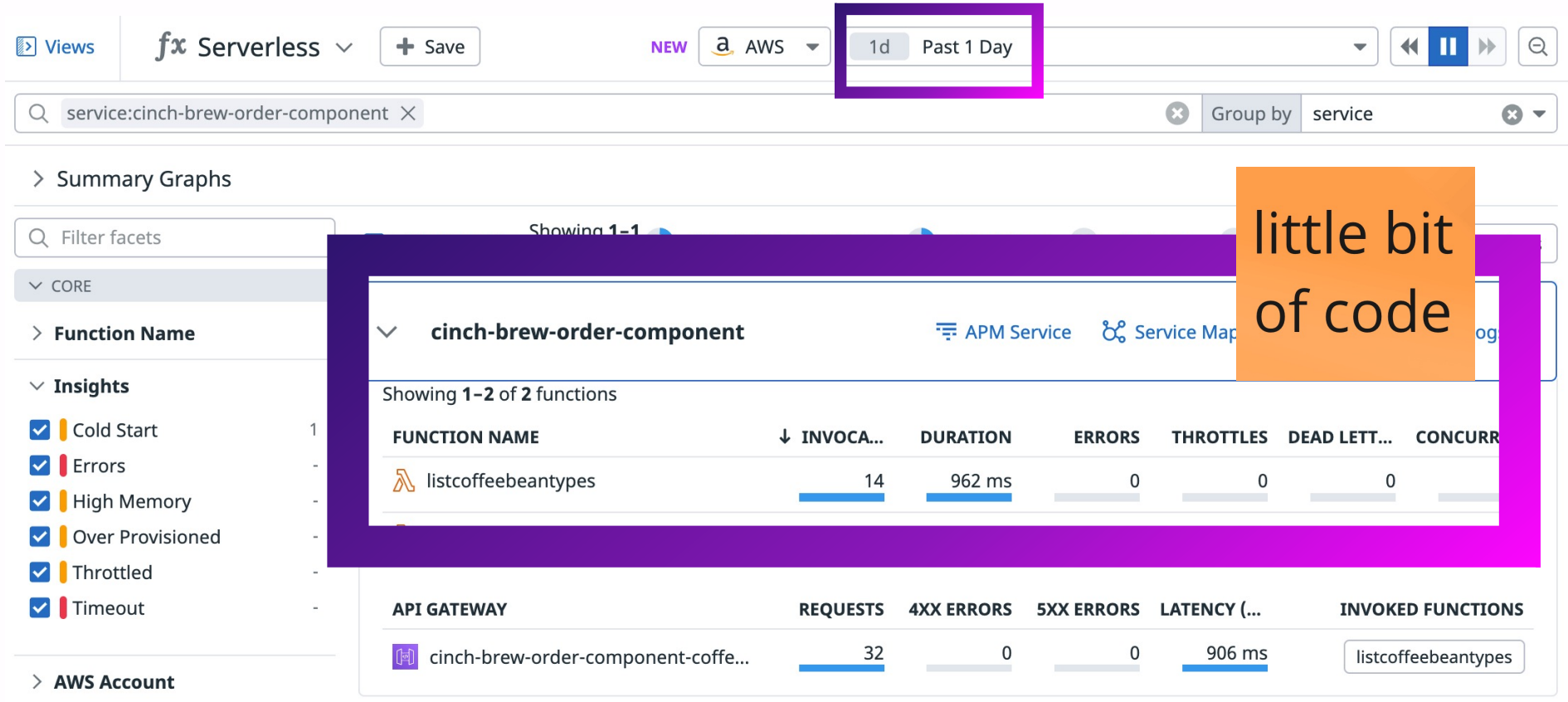
Out of the box top level metrics



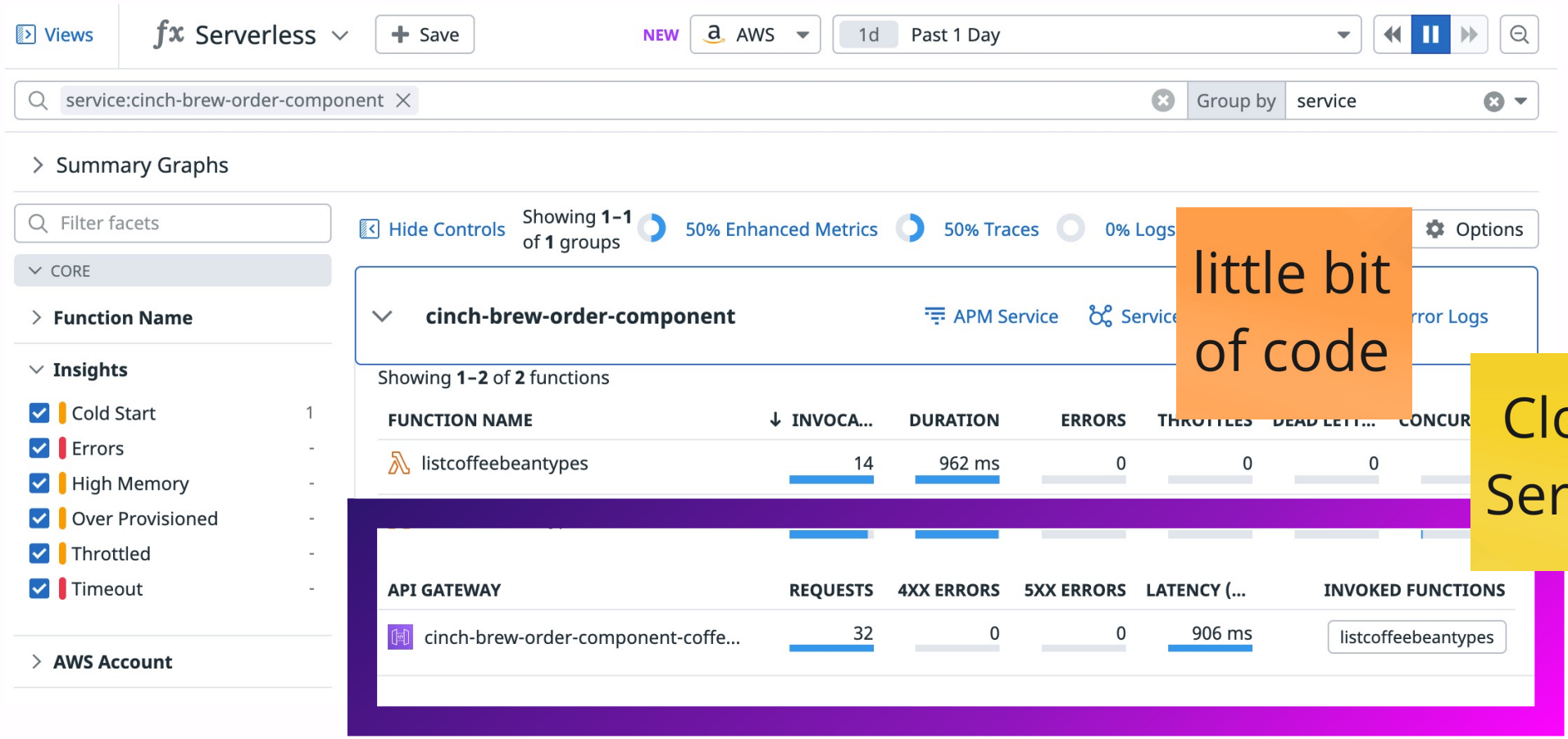




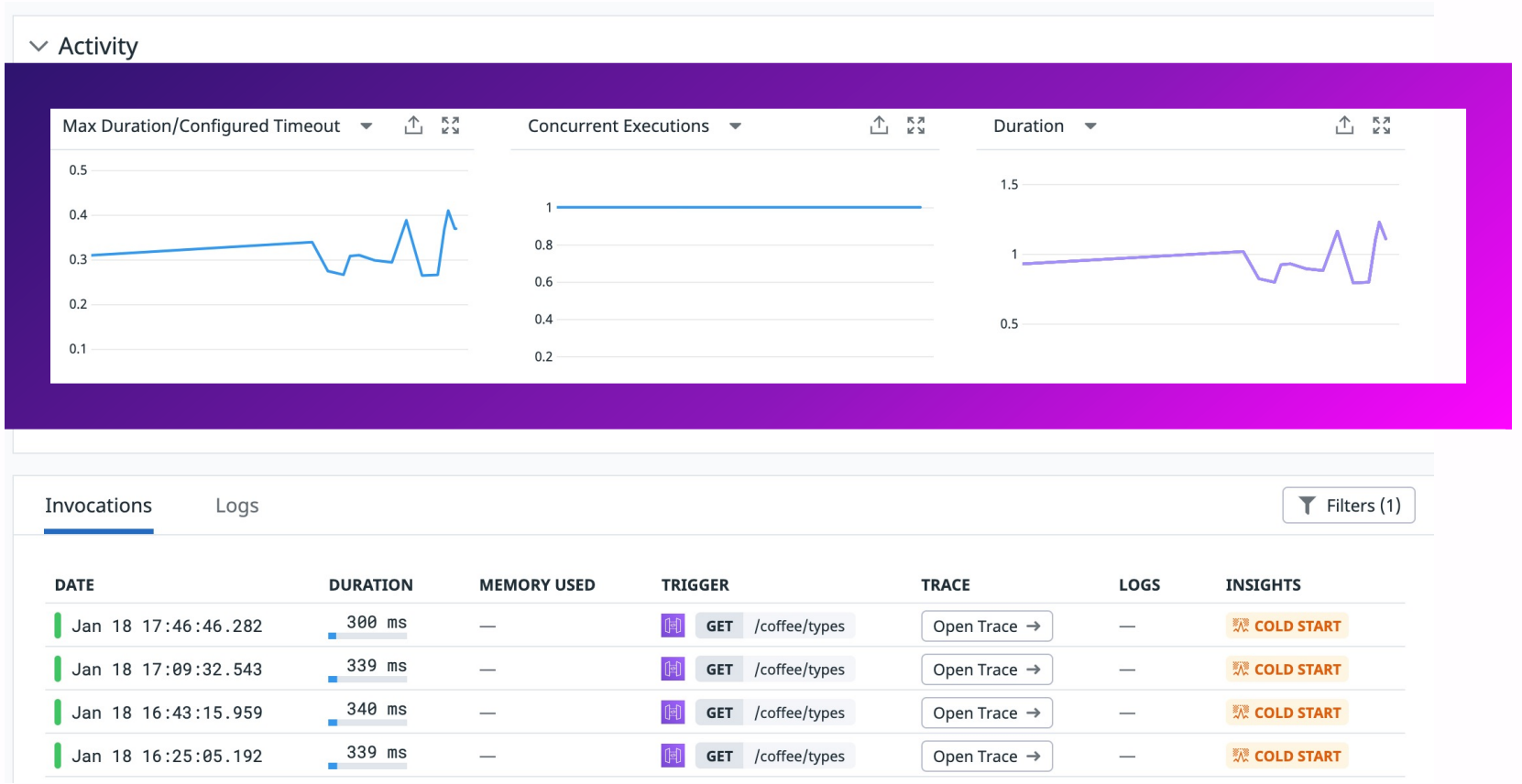
Out of the box stack overview



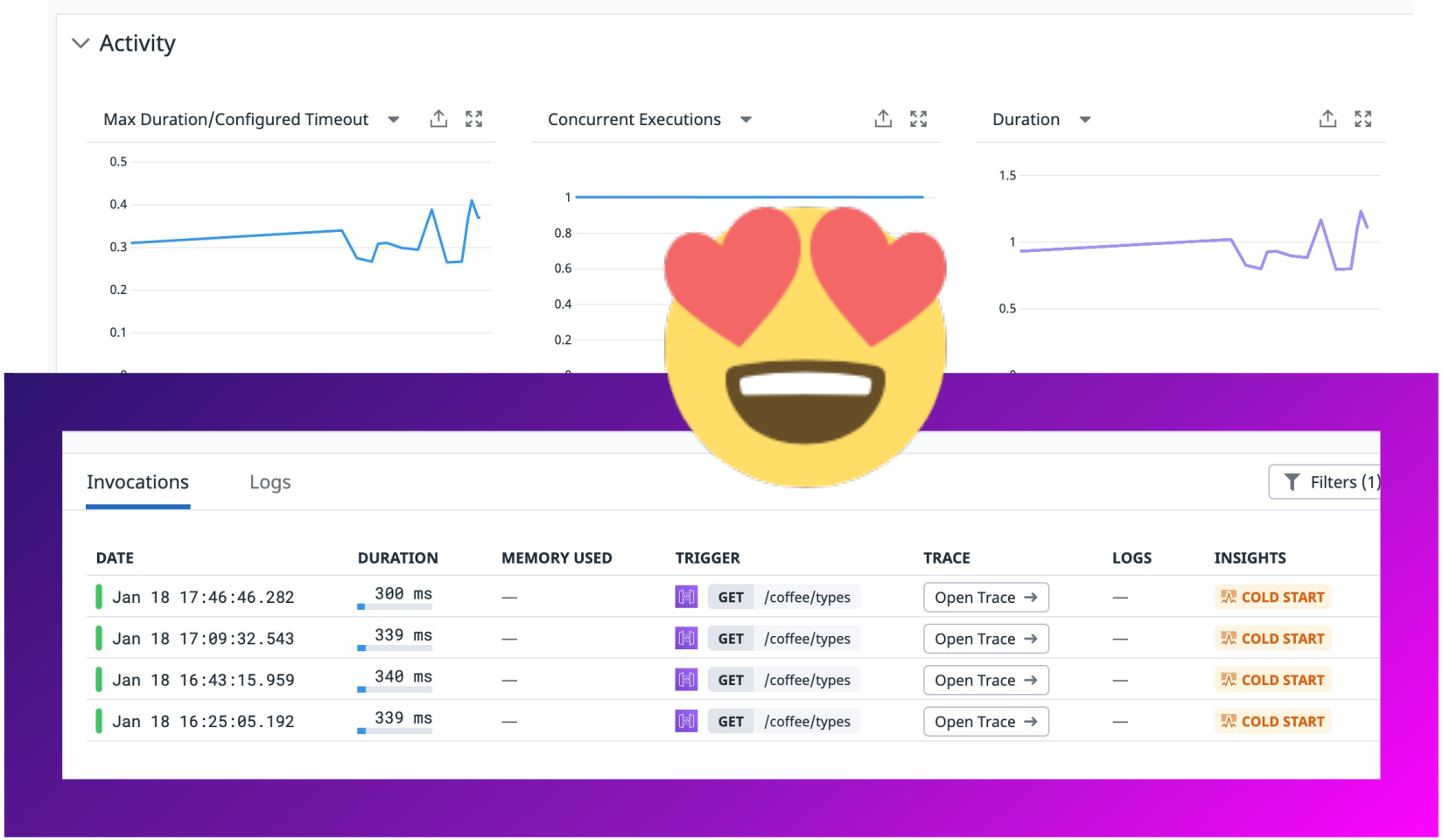
Cloud service too!



Serverless metrics



All your invocations



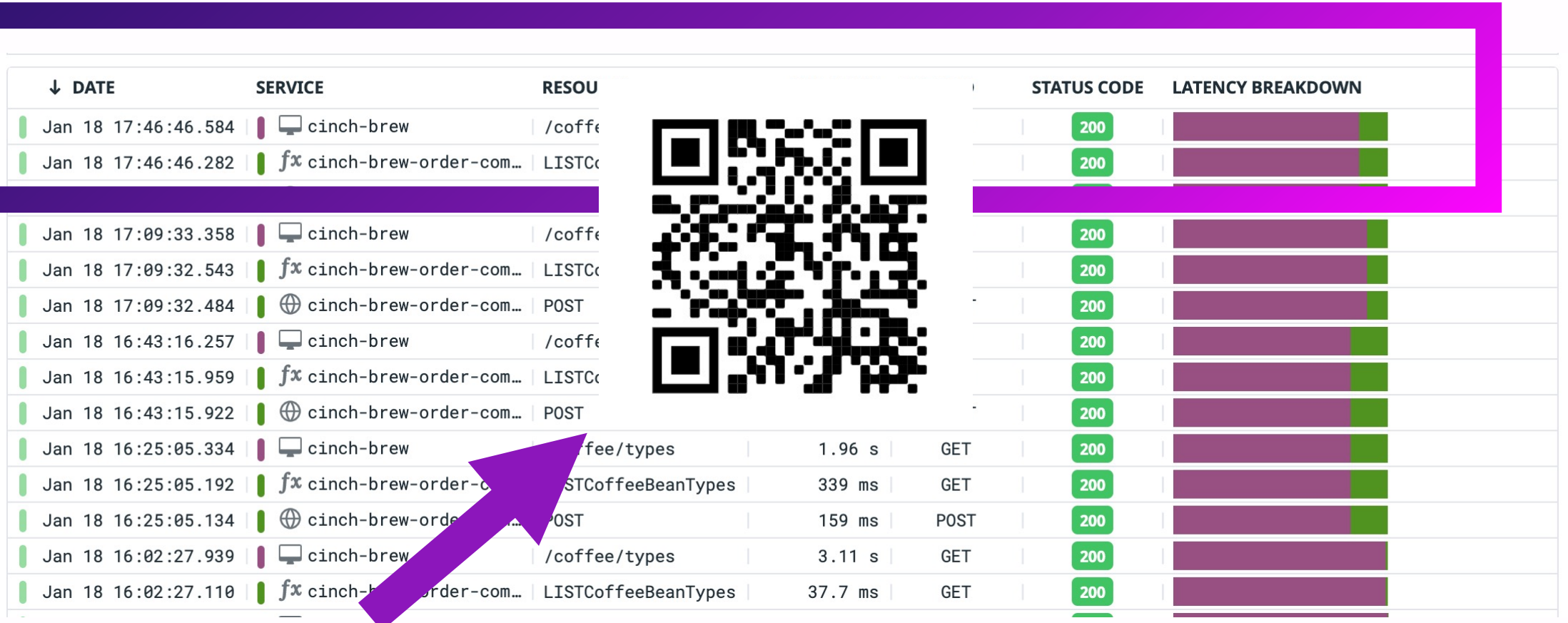
All our spans* too!

↓ DATE	SERVICE	RESOURCE	DURATION	METHOD	STATUS CODE	LATENCY BREAKDOWN
Jan 18 17:46:46.584	cinch-brew	/coffee/types	2.23 s	GET	200	
Jan 18 17:46:46.282	cinch-brew-order-com…	LISTCoffeeBeanTypes	300 ms	GET	200	
Jan 18 17:09:33.358	cinch-brew	/coffee/types	3.46 s	GET	200	
Jan 18 17:09:32.543	cinch-brew-order-com…	LISTCoffeeBeanTypes	339 ms	GET	200	
Jan 18 17:09:32.484	cinch-brew-order-com…	POST	139 ms	POST	200	
Jan 18 16:43:16.257	cinch-brew	/coffee/types	1.96 s	GET	200	
Jan 18 16:43:15.959	cinch-brew-order-com…	LISTCoffeeBeanTypes	340 ms	GET	200	
Jan 18 16:43:15.922	cinch-brew-order-com…	POST	141 ms	POST	200	
Jan 18 16:25:05.334	cinch-brew	/coffee/types	1.96 s	GET	200	
Jan 18 16:25:05.192	cinch-brew-order-com…	LISTCoffeeBeanTypes	339 ms	GET	200	
Jan 18 16:25:05.134	cinch-brew-order-com…	POST	159 ms	POST	200	
Jan 18 16:02:27.939	cinch-brew	/coffee/types	3.11 s	GET	200	
Jan 18 16:02:27.110	cinch-brew-order-com…	LISTCoffeeBeanTypes	37.7 ms	GET	200	

*I'm not going to get into spans, look them up

cinch

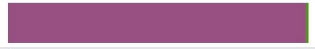
All our spans* too!



↓ DATE	SERVICE	RESOU			STATUS CODE	LATENCY BREAKDOWN
Jan 18 17:46:46.584	cinch-brew	/coffe			200	
Jan 18 17:46:46.282	cinch-brew-order-com…	LISTCo			200	
Jan 18 17:09:33.358	cinch-brew	/coffe			200	
Jan 18 17:09:32.543	cinch-brew-order-com…	LISTCo			200	
Jan 18 17:09:32.484	cinch-brew-order-com…	POST			200	
Jan 18 16:43:16.257	cinch-brew	/coffe			200	
Jan 18 16:43:15.959	cinch-brew-order-com…	LISTCo			200	
Jan 18 16:43:15.922	cinch-brew-order-com…	POST			200	
Jan 18 16:25:05.334	cinch-brew	ffee/types	1.96 s	GET	200	
Jan 18 16:25:05.192	cinch-brew-order-c	STCoffeeBeanTypes	339 ms	GET	200	
Jan 18 16:25:05.134	cinch-brew-orde	POST	159 ms	POST	200	
Jan 18 16:02:27.939	cinch-brew	/coffee/types	3.11 s	GET	200	
Jan 18 16:02:27.110	cinch-b…rder-com…	LISTCoffeeBeanTypes	37.7 ms	GET	200	

*Here's a useful link

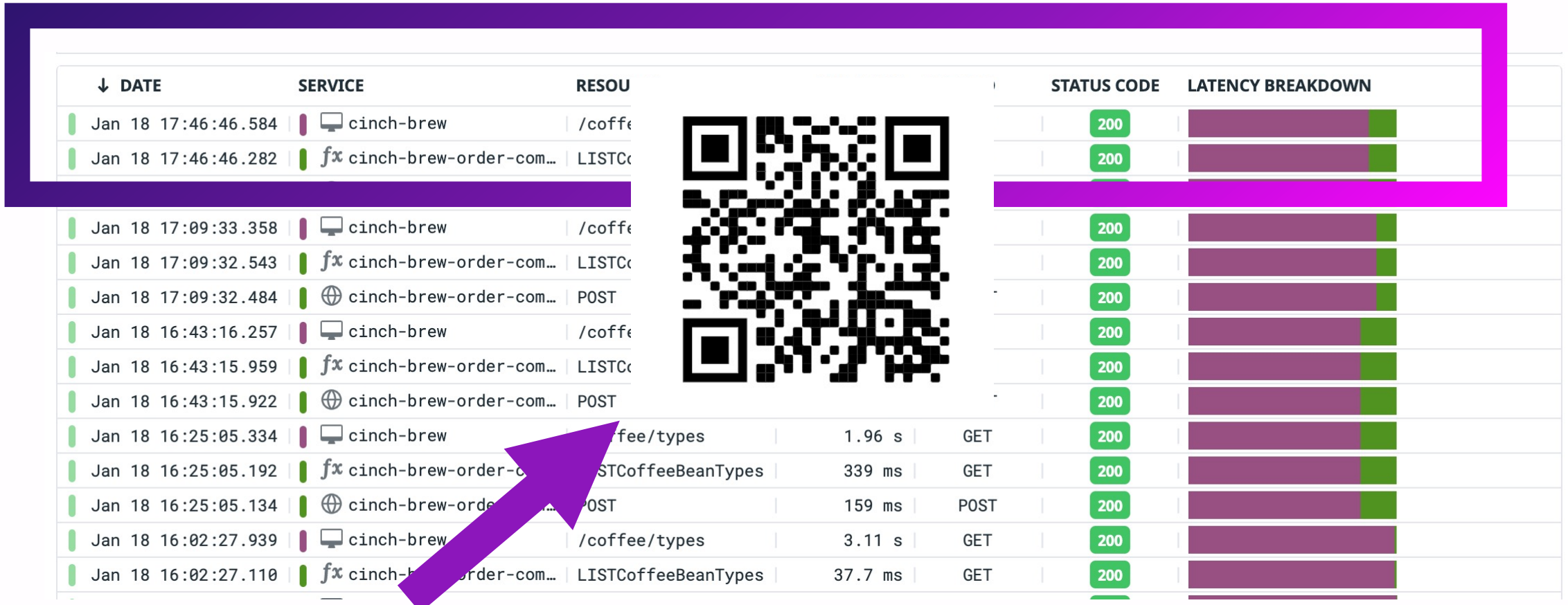
All our spans* too!

↓ DATE	SERVICE	RESOURCE	DURATION	METHOD	STATUS CODE	LATENCY BREAKDOWN
Jan 18 17:46:46.584	cinch-brew	/coffee/types	2.23 s	GET	200	
Jan 18 17:46:46.282	cinch-brew-order-com…	LISTCoffeeBeanTypes	300 ms	GET	200	
Jan 18 17:09:33.358	cinch-brew	/coffee/types	3.46 s	GET	200	
Jan 18 17:09:32.543	cinch-brew-order-com…	LISTCoffeeBeanTypes	339 ms	GET	200	
Jan 18 17:09:32.484	cinch-brew-order-com…	POST	139 ms	POST	200	
Jan 18 16:43:16.257	cinch-brew	/coffee/types	1.96 s	GET	200	
Jan 18 16:43:15.959	cinch-brew-order-com…	LISTCoffeeBeanTypes	340 ms	GET	200	
Jan 18 16:43:15.922	cinch-brew-order-com…	POST	141 ms	POST	200	
Jan 18 16:25:05.334	cinch-brew	/coffee/types	1.96 s	GET	200	
Jan 18 16:25:05.192	cinch-brew-order-com…	LISTCoffeeBeanTypes	339 ms	GET	200	
Jan 18 16:25:05.134	cinch-brew-order-com…	POST	159 ms	POST	200	
Jan 18 16:02:27.939	cinch-brew	/coffee/types	3.11 s	GET	200	
Jan 18 16:02:27.110	cinch-brew-order-com…	LISTCoffeeBeanTypes	37.7 ms	GET	200	

*Seriously, next slide can wait.

cinch

All our spans* too!

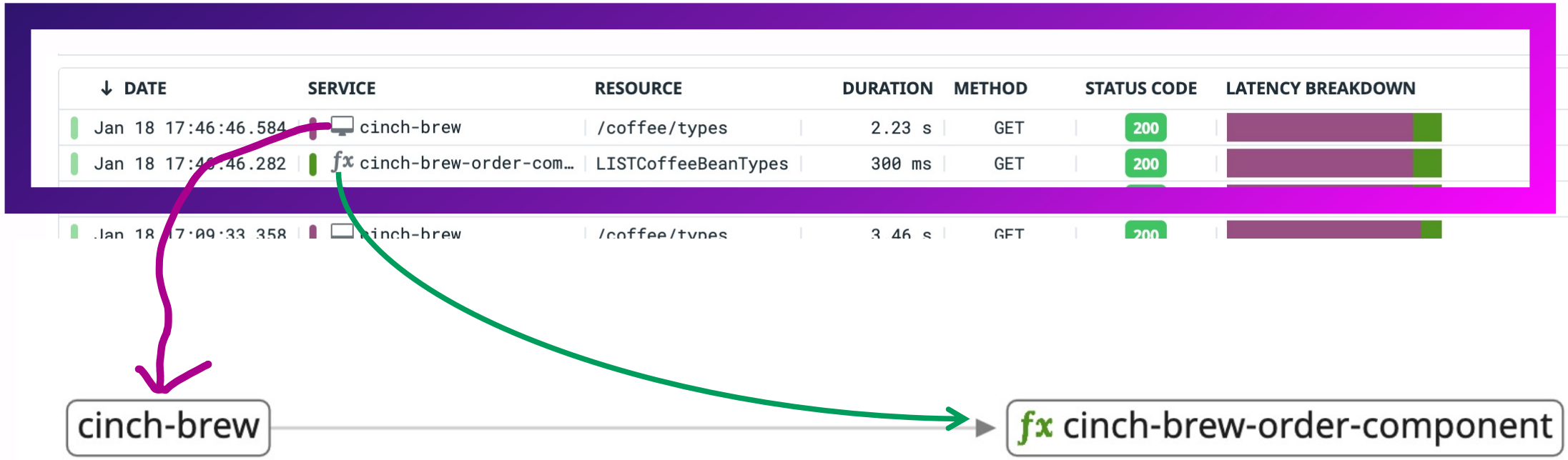


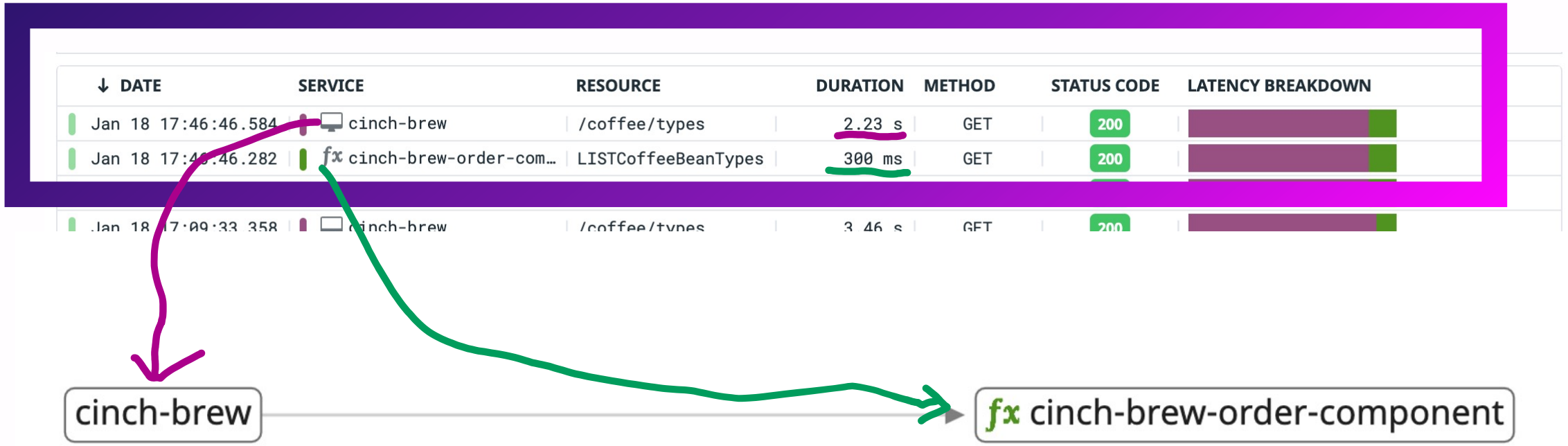
↓ DATE	SERVICE	RESOU			STATUS CODE	LATENCY BREAKDOWN
Jan 18 17:46:46.584	cinch-brew	/coffe			200	
Jan 18 17:46:46.282	cinch-brew-order-com…	LISTCo			200	
Jan 18 17:09:33.358	cinch-brew	/coffe			200	
Jan 18 17:09:32.543	cinch-brew-order-com…	LISTCo			200	
Jan 18 17:09:32.484	cinch-brew-order-com…	POST			200	
Jan 18 16:43:16.257	cinch-brew	/coffe			200	
Jan 18 16:43:15.959	cinch-brew-order-com…	LISTCo			200	
Jan 18 16:43:15.922	cinch-brew-order-com…	POST			200	
Jan 18 16:25:05.334	cinch-brew	ffee/types	1.96 s	GET	200	
Jan 18 16:25:05.192	cinch-brew-order-c	STCoffeeBeanTypes	339 ms	GET	200	
Jan 18 16:25:05.134	cinch-brew-orde	OST	159 ms	POST	200	
Jan 18 16:02:27.939	cinch-brew	/coffee/types	3.11 s	GET	200	
Jan 18 16:02:27.110	cinch-b…order-com…	LISTCoffeeBeanTypes	37.7 ms	GET	200	

*https://opentelemetry.lightstep.com/spans/

cinch

Anyway, these are two spans

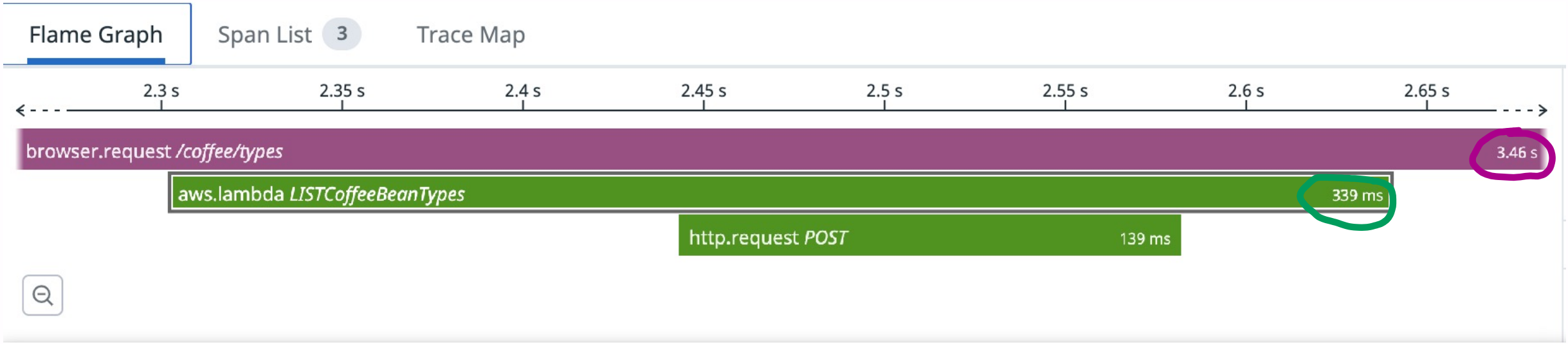




Frontend span

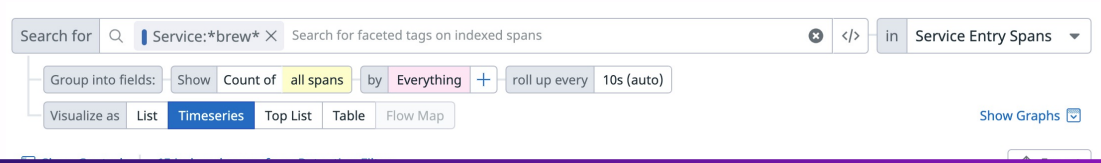
Backend span

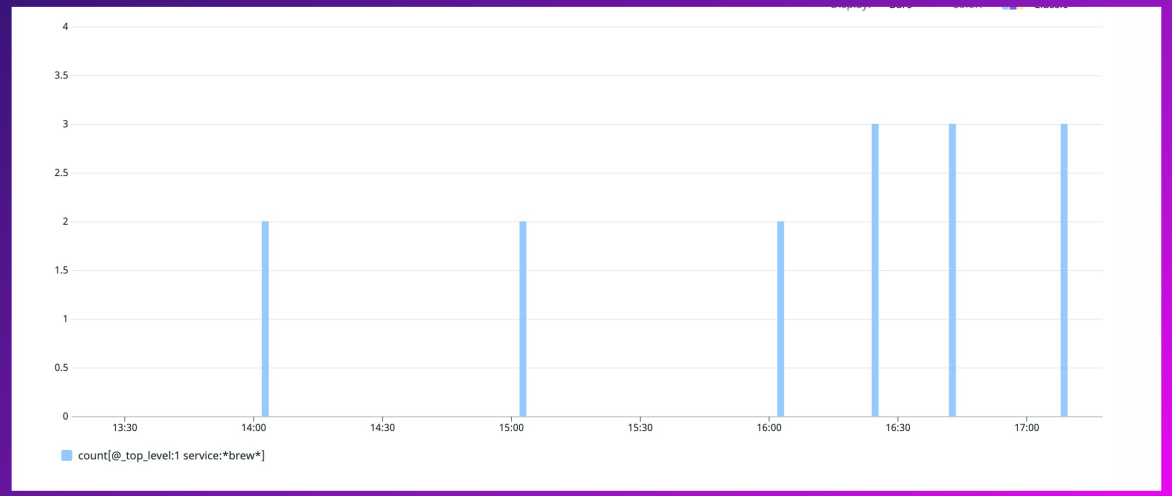
Flame graph: a better way of visualising spans

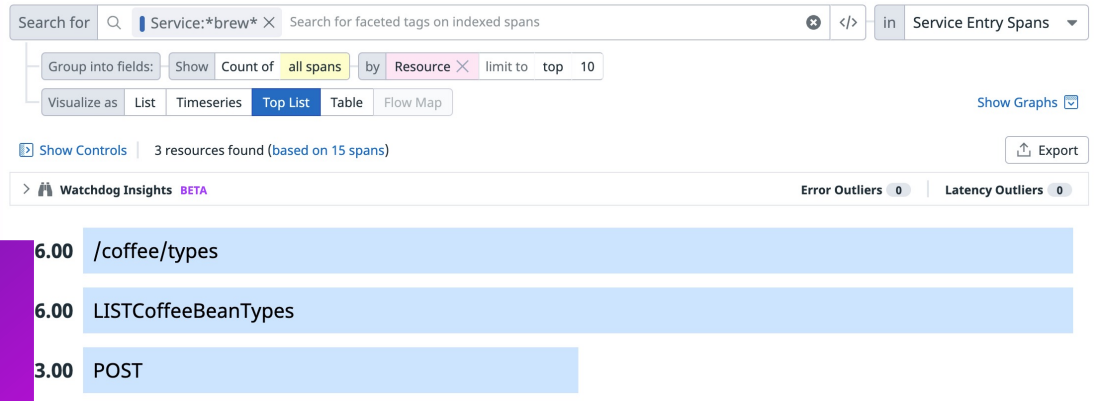


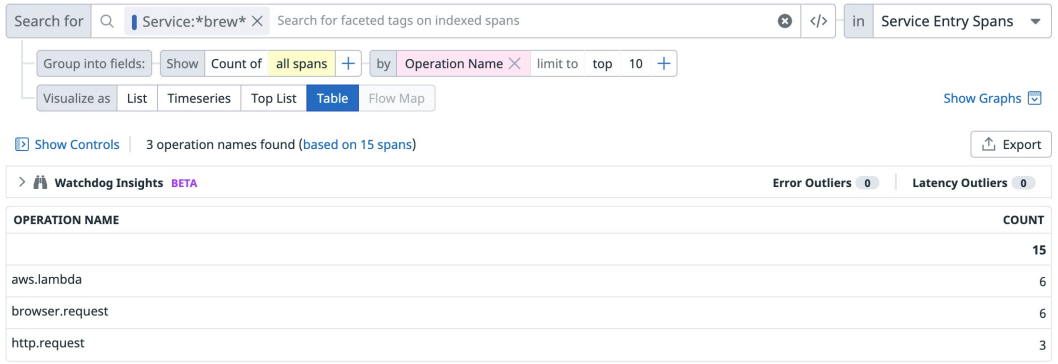
cinch

Spans as time series

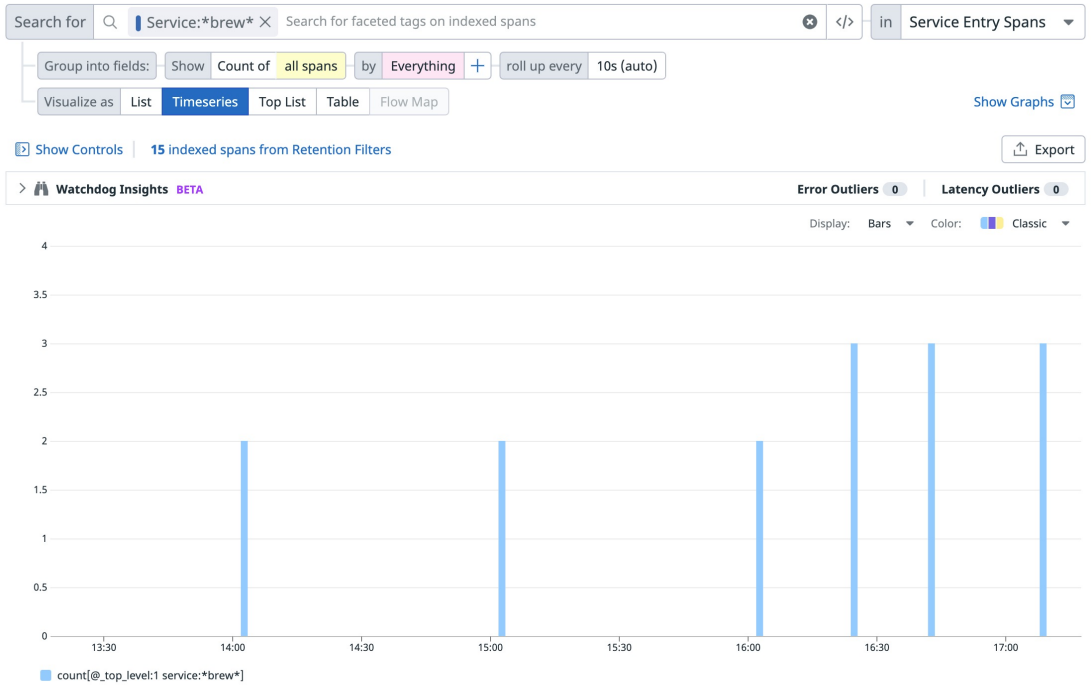


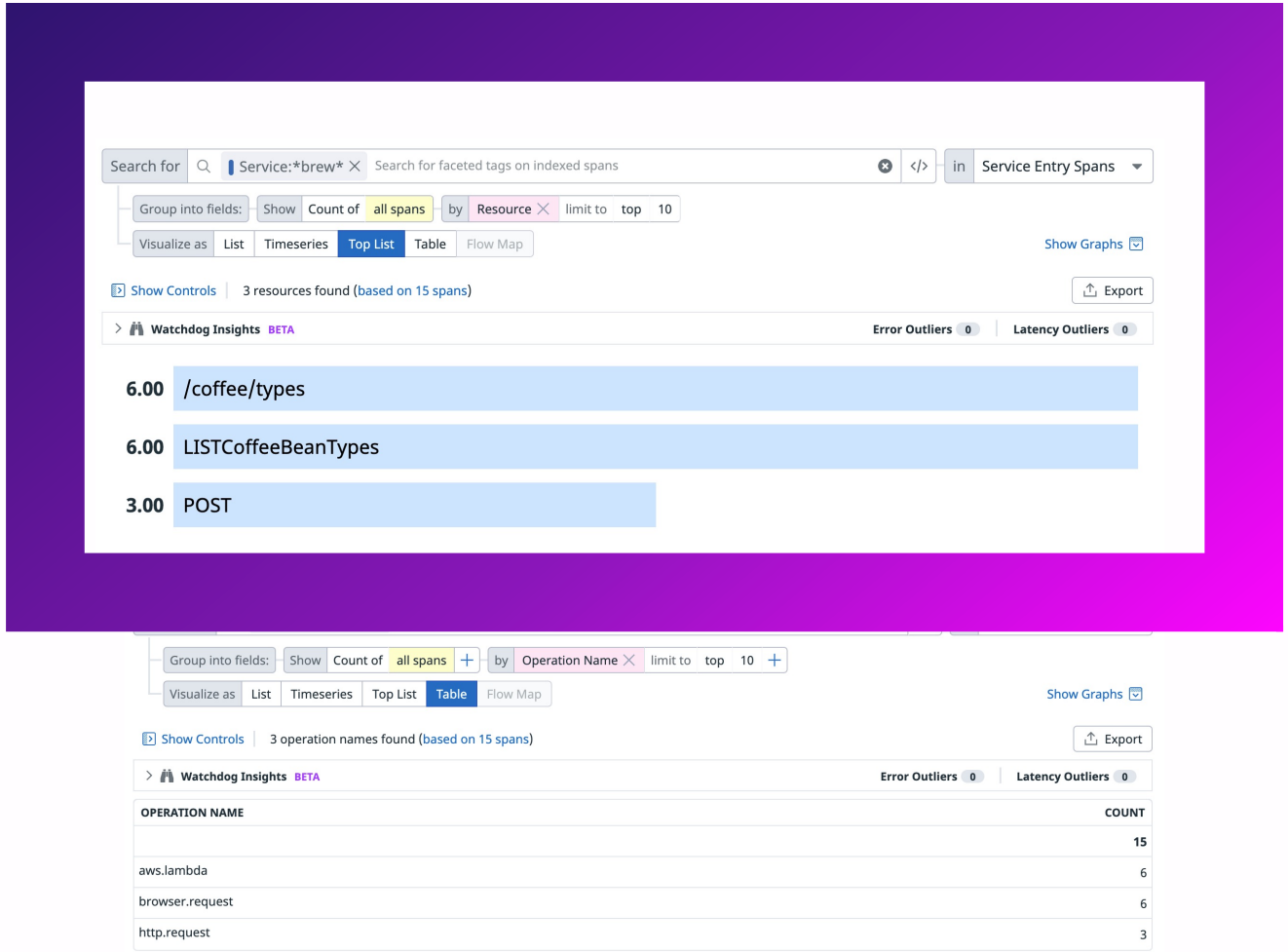




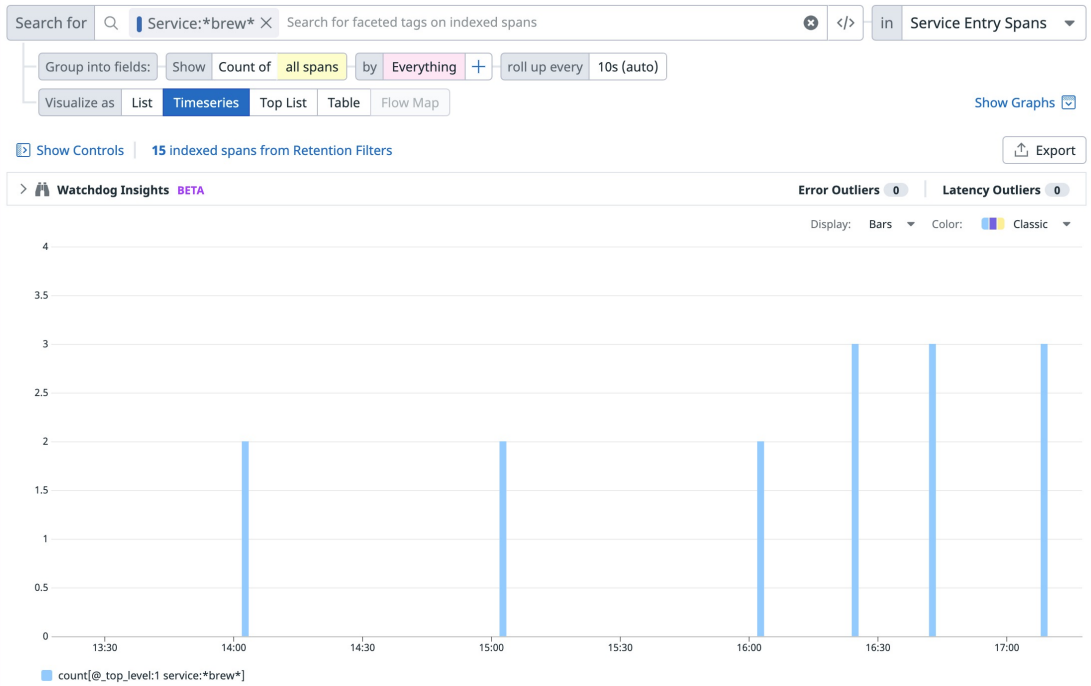


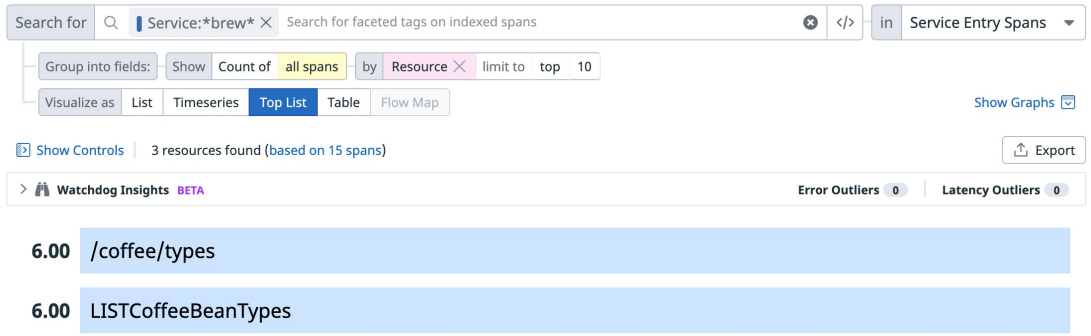
Spans as top list

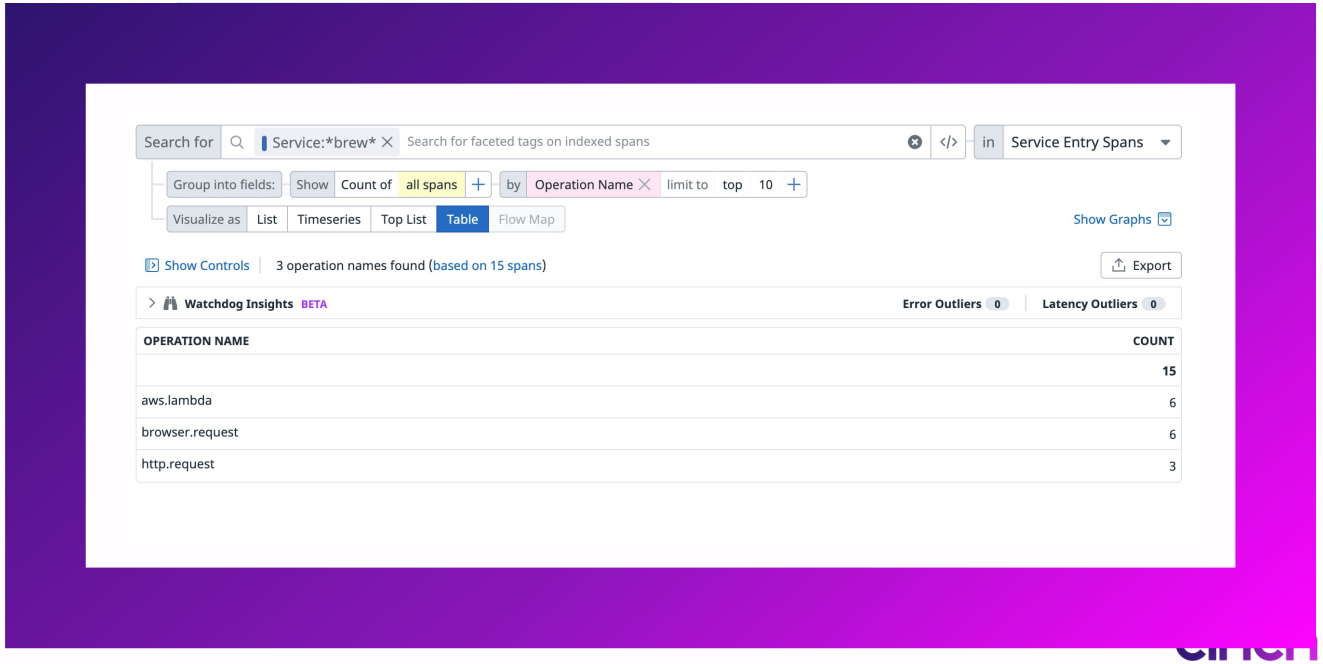




Even spans as tables







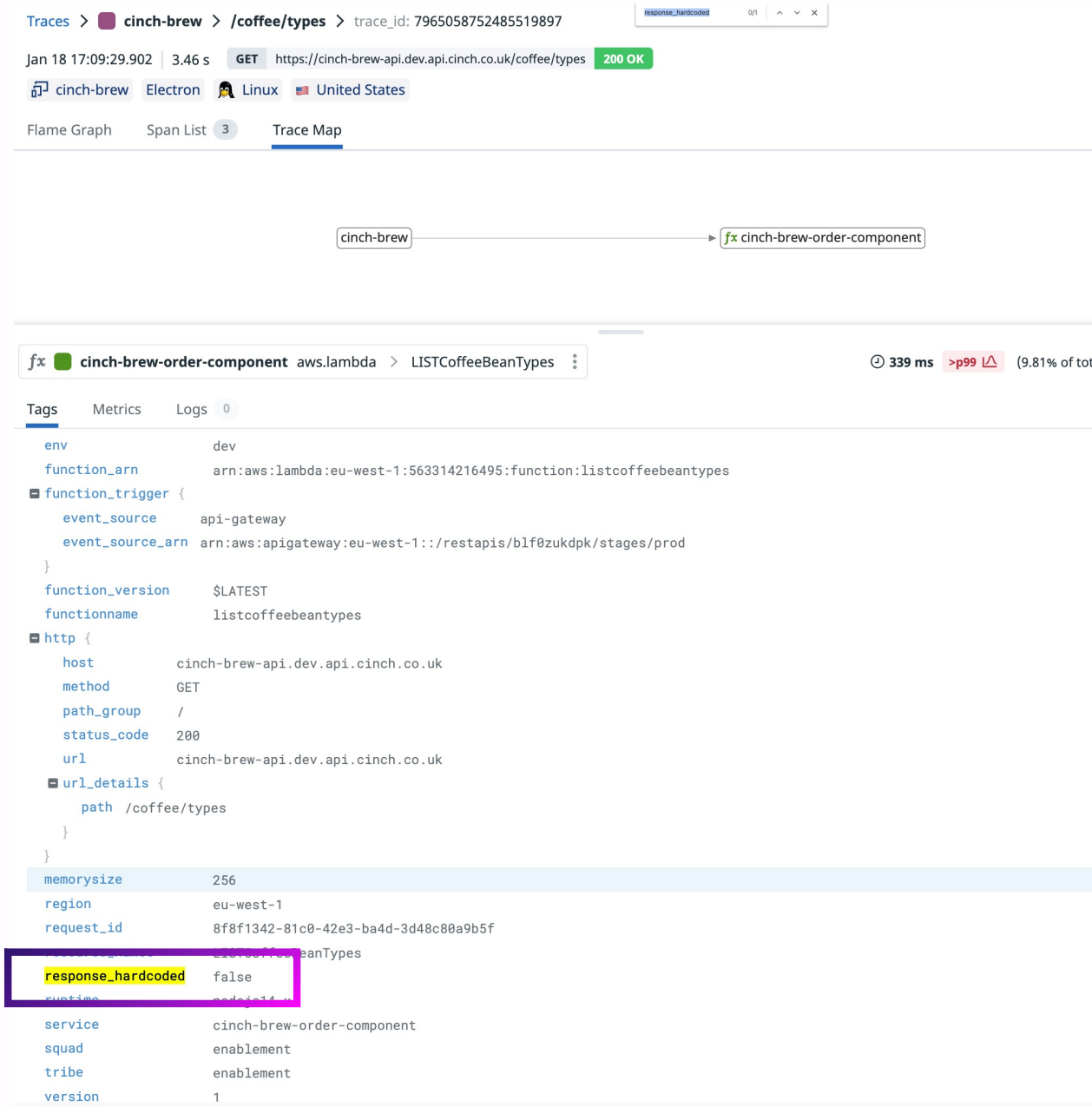
Custom instrumentation makes the difference.

cinch

Custom instrumentation: set tags

```
4  import { setTag, sendMetric } from '../../../helpers/observability';
5  import { GetCommand } from '@aws-sdk/lib-dynamodb';
6  import { dynamodbDocClient } from '../../../helpers/aws-clients';
7
8  export const handler: Handler<APIGatewayProxyEventV2, APIGatewayProxyStructuredResultV2> = async () => {
9      setTag('response_hardcoded', false);
10     const params = {
11         TableName: 'cinch-brew-order-component-store',
12         Key: { id: 'TYPES', sk: 'COFFEE_BEANS' },
13     };
14     const coffeeBeanTypes = await dynamodbDocClient.send(new GetCommand(params)).then((res) => res.Item?.options || []);
15
16     sendMetric('available_coffee_bean_types', coffeeBeanTypes.length, 'metricType:count');
17
18     if (coffeeBeanTypes.length > 0) {
19         return apiGatewayResult({ body: { types: coffeeBeanTypes } });
20     } else {
21         return apiGatewayResult({ status: 404 });
22     }
23 };
```

Our custom tag on the span



We use metrics heavily*

Traces

high context & traceability

Metrics

high value signals with limited context

Logs

high context with free text included & links to traces

RUM Events

high context & links to traces

* Although we do prefer traces and spans

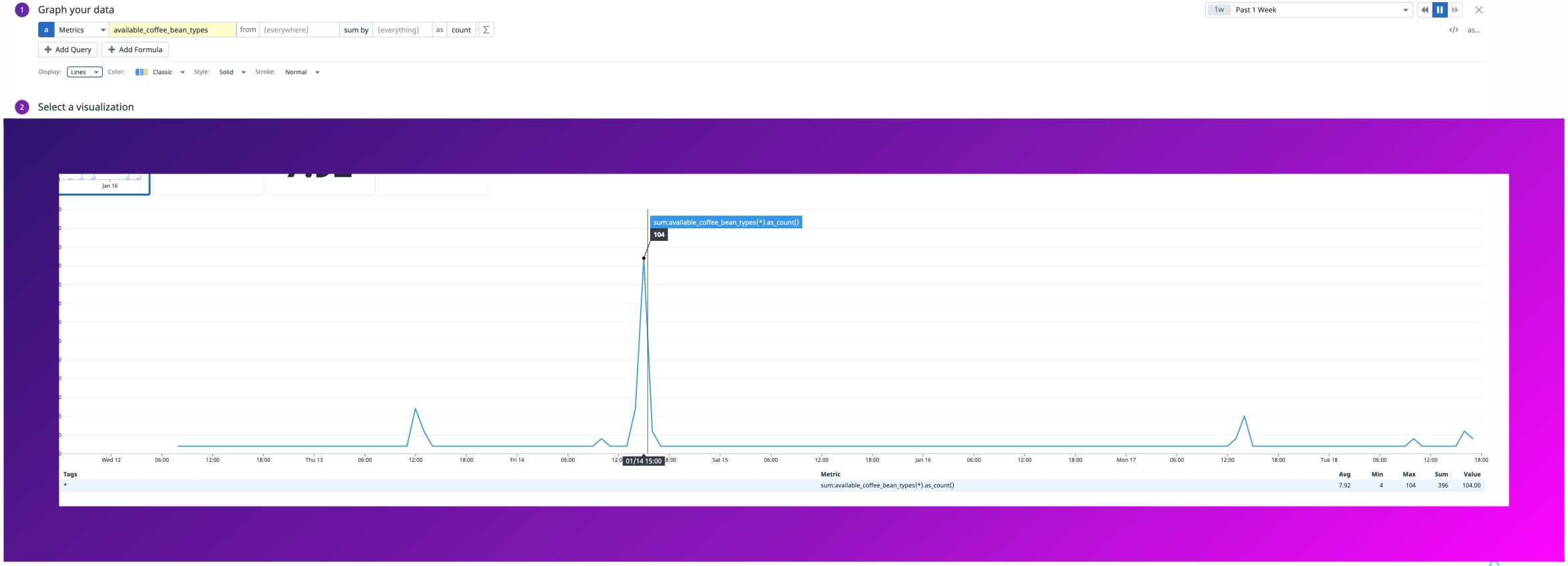
cinch

We send metrics directly from code

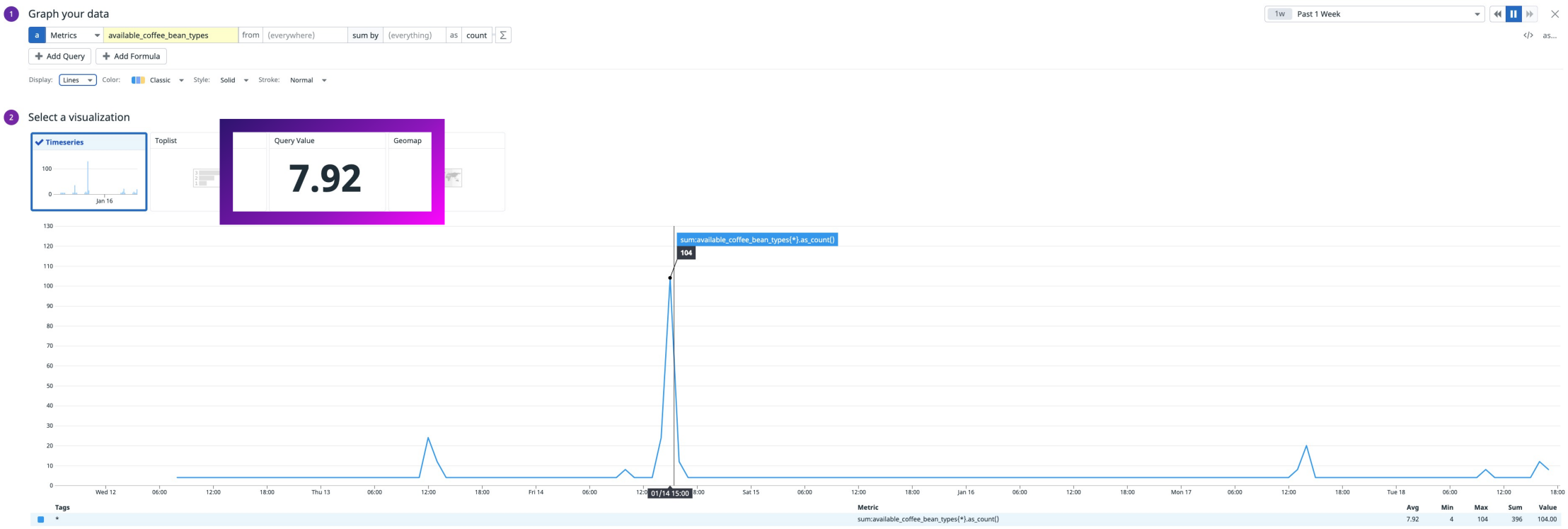
```
4  import { setTag, sendMetric } from '../../../helpers/observability';
5  import { GetCommand } from '@aws-sdk/lib-dynamodb';
6  import { dynamodbDocClient } from '../../../helpers/aws-clients';
7
8  export const handler: Handler<APIGatewayProxyEventV2, APIGatewayProxyStructuredResultV2> = async () => {
9      setTag('response_hardcoded', false);
10     const params = {
11         TableName: 'cinch-brew-order-component-store',
12         Key: { id: 'TYPES', sk: 'COFFEE_BEANS' },
13     };
14     const coffeeBeanTypes = await dynamodbDocClient.send(new GetCommand(params)).then((res) => res.Item?.options || []);
15
16     sendMetric('available_coffee_bean_types', coffeeBeanTypes.length, 'metricType:count');
17
18     if (coffeeBeanTypes.length > 0) {
19         return apiGatewayResult({ body: { types: coffeeBeanTypes } });
20     } else {
21         return apiGatewayResult({ status: 404 });
22     }
23 };
```

cinch

We can then graph metric values over time



Or create a single number



RUM* events: the tracing of the frontend

Traces

high context & traceability

Metrics

high value signals with limited context

Logs

high context with free text included & links to traces

RUM Events

high context & links to traces

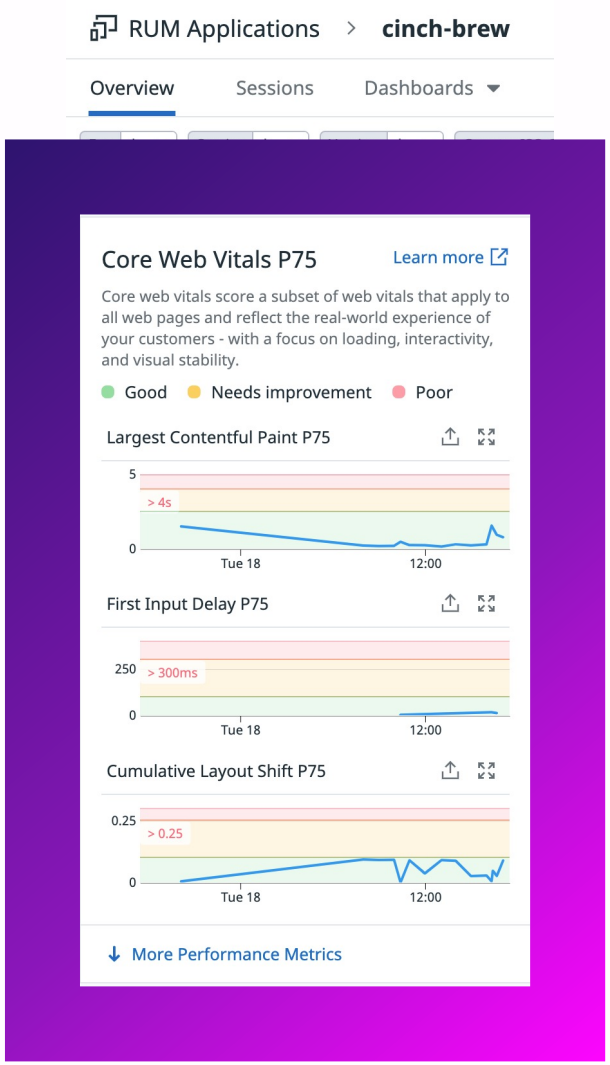
* RUM stands for Real User Monitoring. I think it deserves to be abbreviated.

cinch

Auto-instrumentation

```
import { datadogRum } from "@datadog/browser-rum";

datadogRum.init({
  applicationId: "[redacted]",
  clientToken: "[redacted]",

  service: "cinch-brew",
  env: "dev",
  version: process.env.NEXT_PUBLIC_BUILD_ID,

  replaySampleRate: 100,
  trackInteractions: true,
  defaultPrivacyLevel: "mask-user-input",
  allowedTracingOrigins: [
    "https://cinch-brew[redacted]cinch.co.uk",
    /https:\/\/.*\.cinch\.co\.uk/,
  ],
});

datadogRum.startSessionReplayRecording();
```

Out of the box analytics



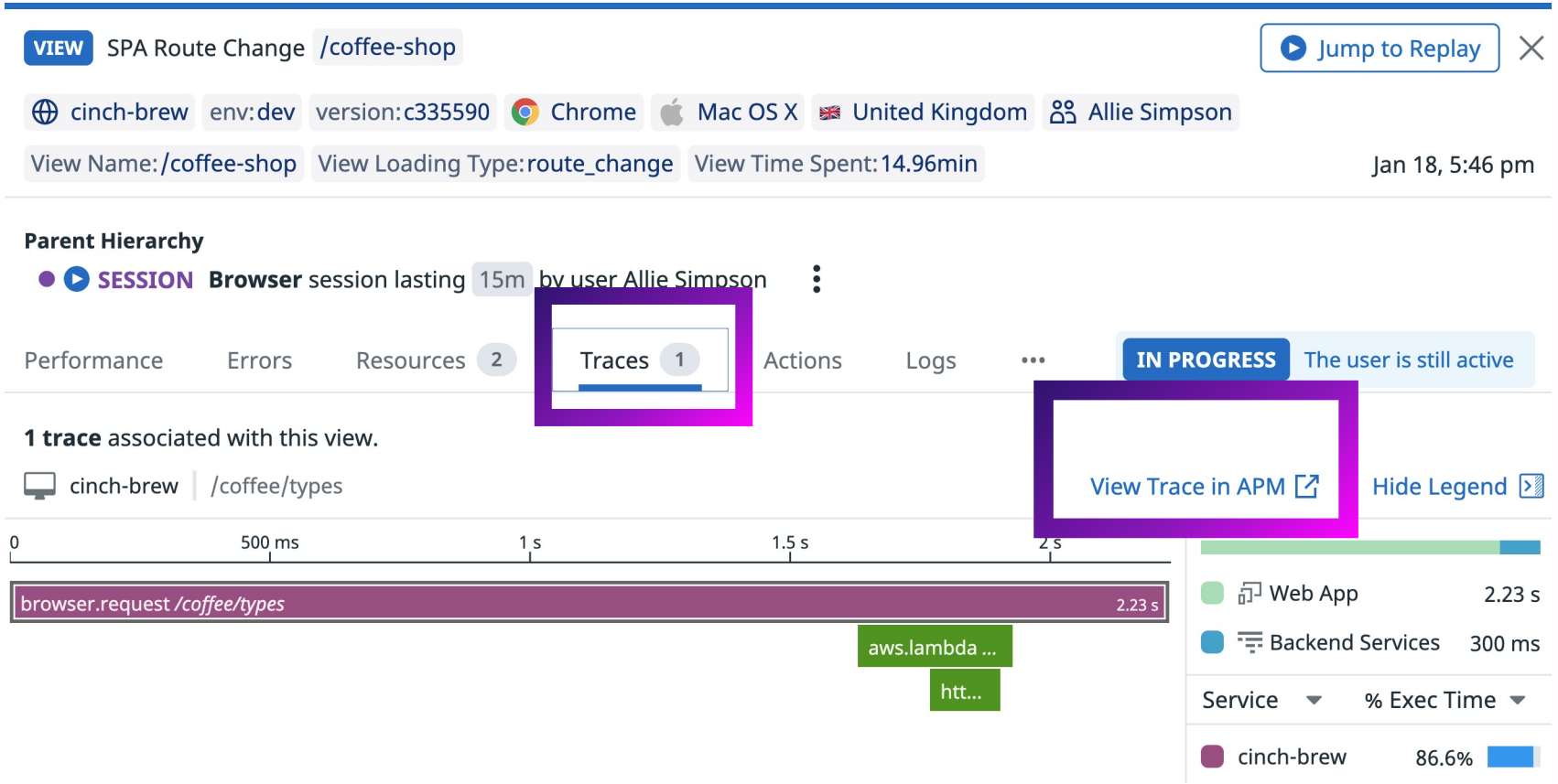
Out of the box performance (and many more)

Most viewed pages

Performance overview of most popular pages

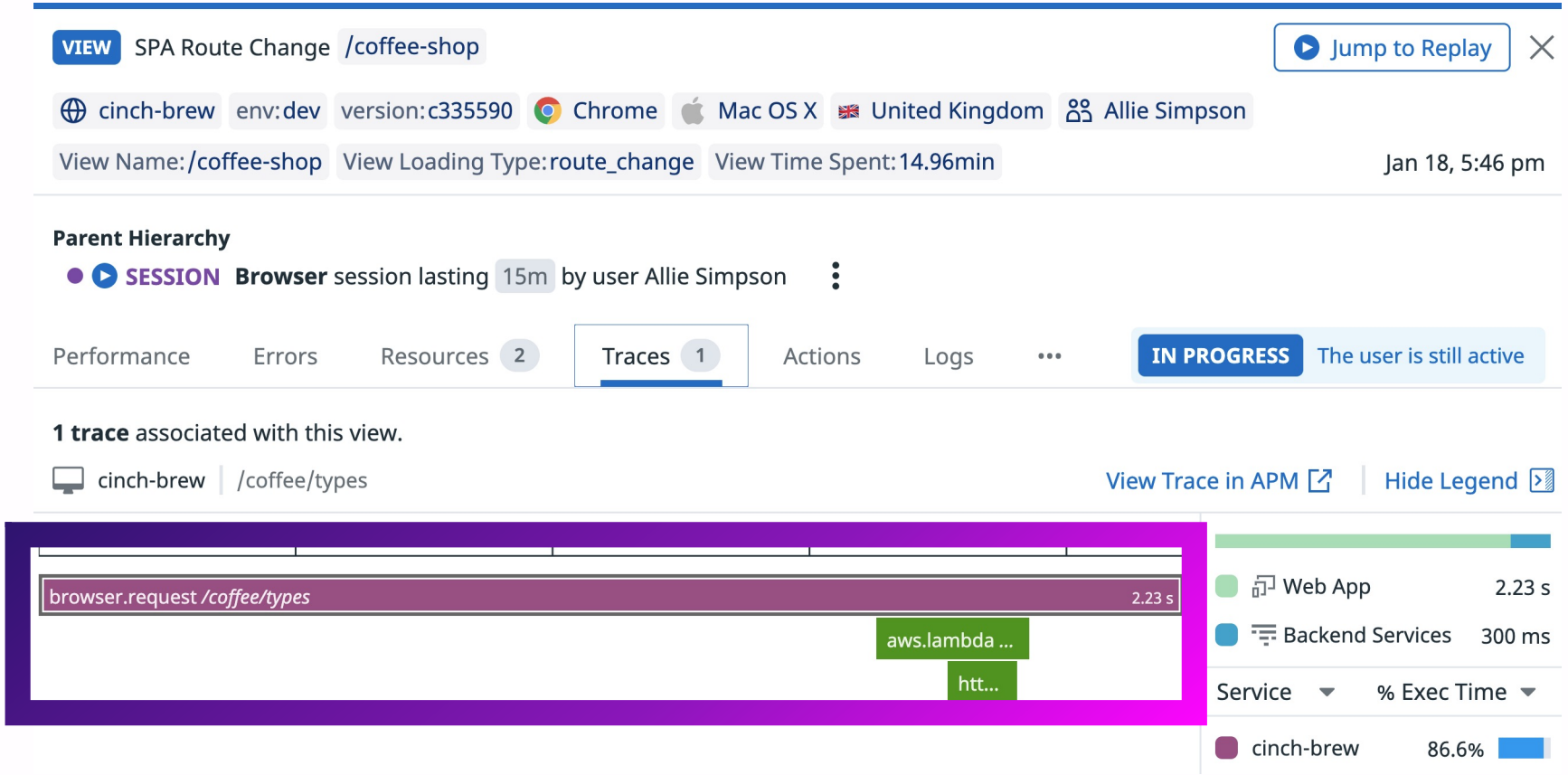
VIEW NAME	↓ COUNT	PC75:LOADING TIME	PC75:LARGEST CONTENTFUL PAINT	PC75:FIRST INPUT DELAY	PC75:CUMULATIVE LAYOUT SHIFT
/	16	1.55s	635.85ms	13.33ms	0.024
/login	14	1.26s	N/A	N/A	0.036

Tracing frontend to backend*

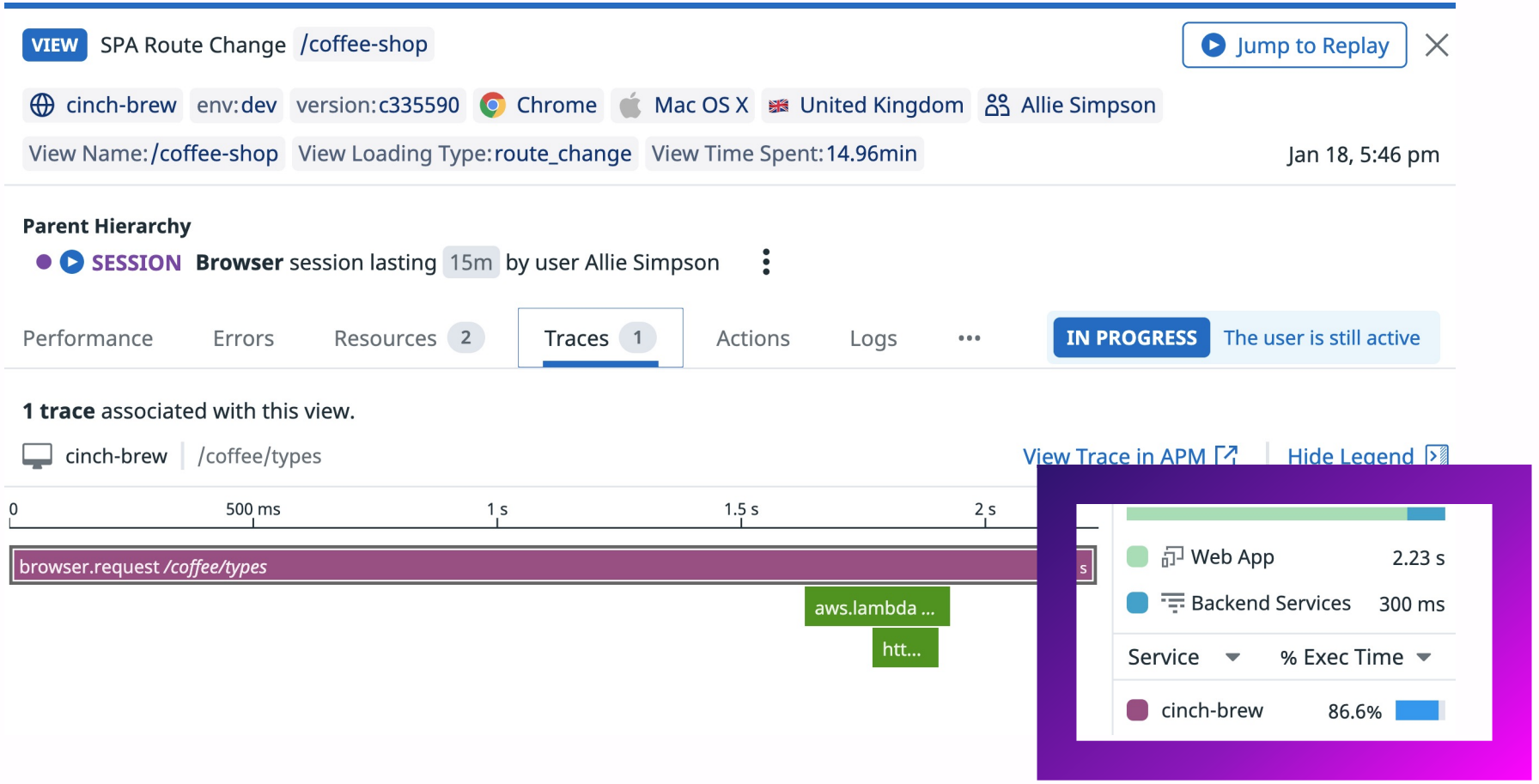


* I have hinted it before. I'm making it explicit here

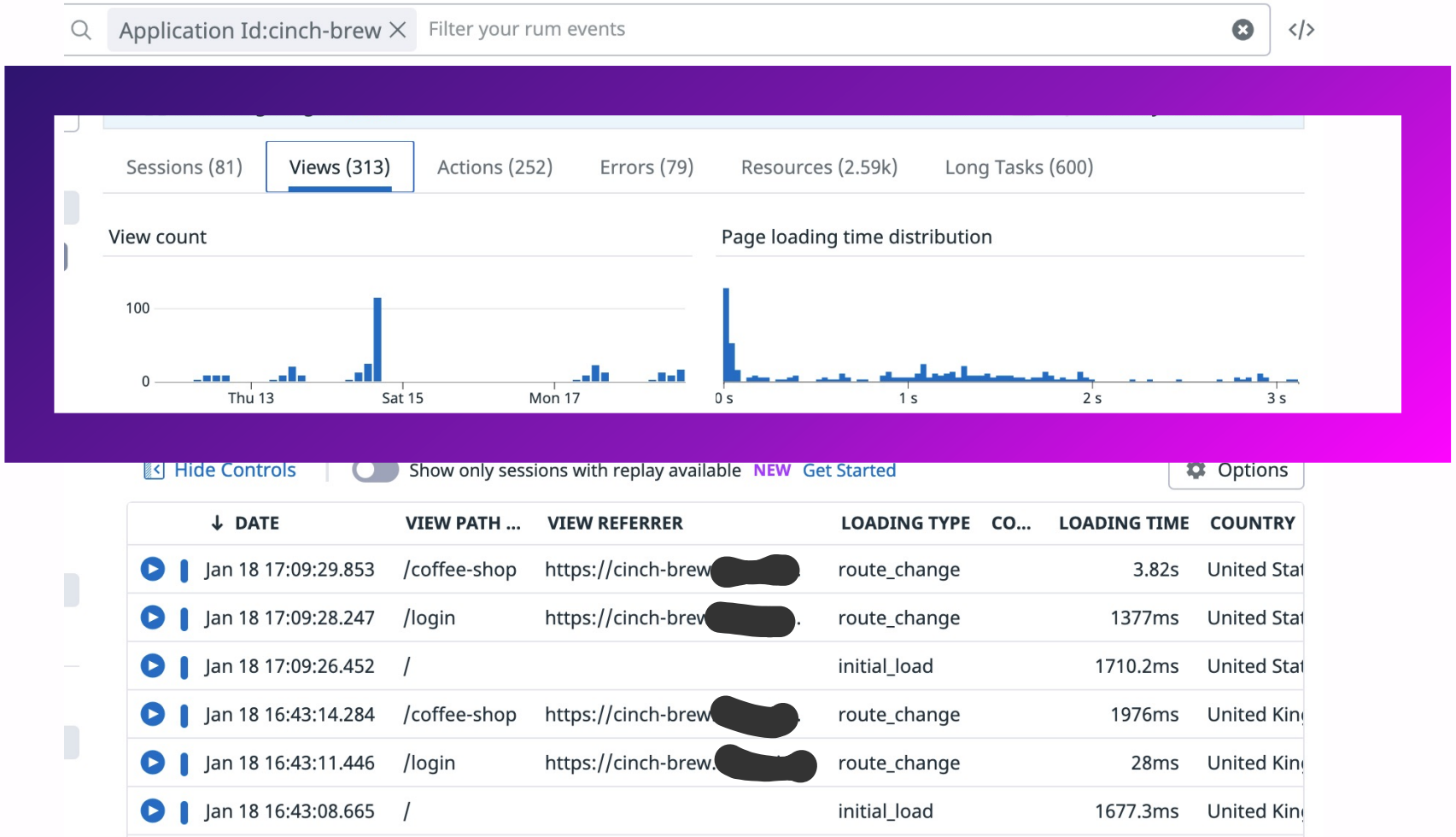
Spans with both frontend and backend



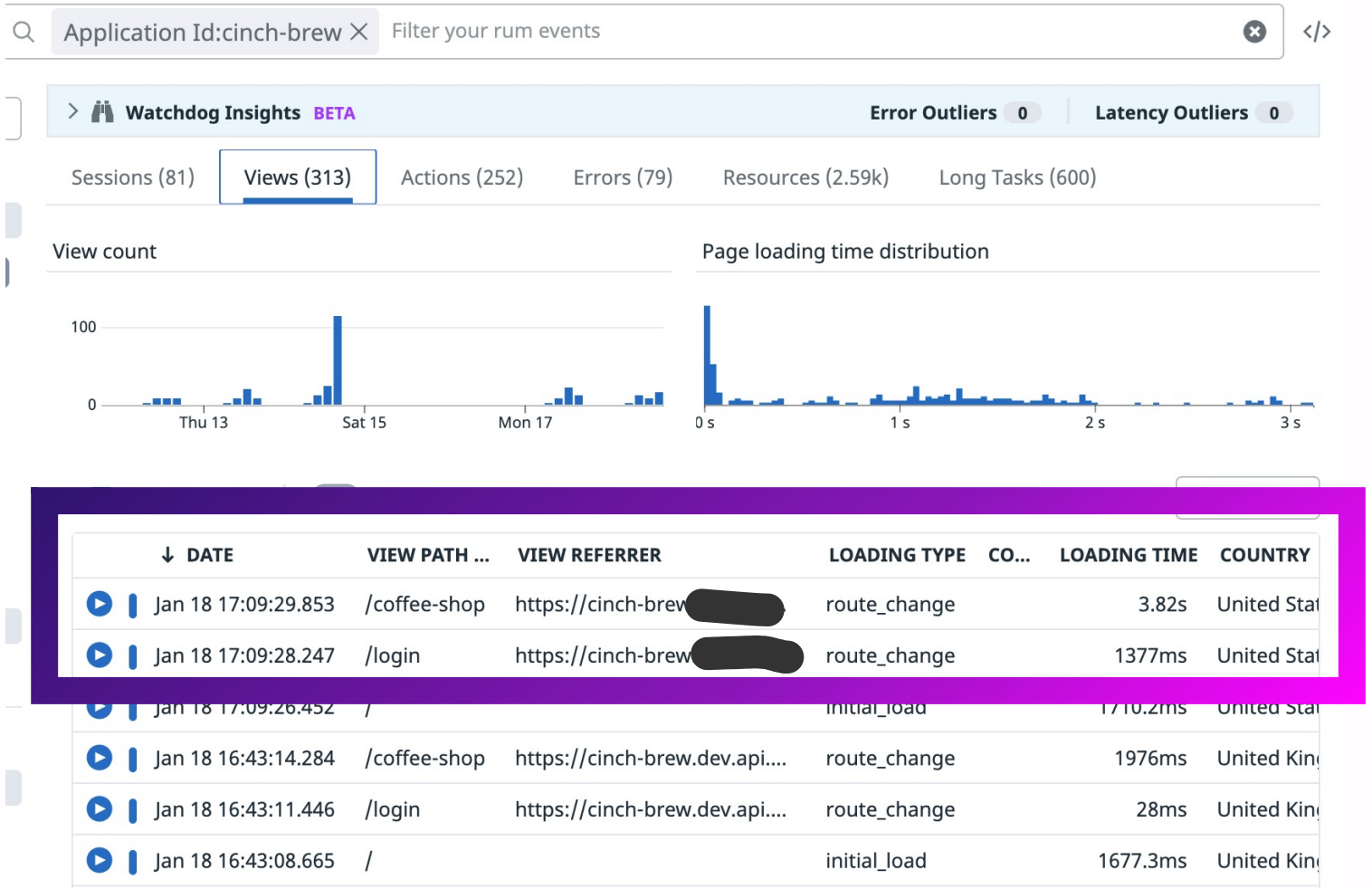
86% taken on the frontend



Like spans, you can list views



Each row is an individual view



Again, custom instrumentation makes the difference.

cinch

Add a custom action*

```
7   useTimeout(() => {
8     // create a custom action to datadog rum to notify that the user has not done much
9     datadogRum.addAction("userSleeping", { inactivitySeconds: 5 });
10  }, 5_000);
```

* I don't have an example of this in action right now.

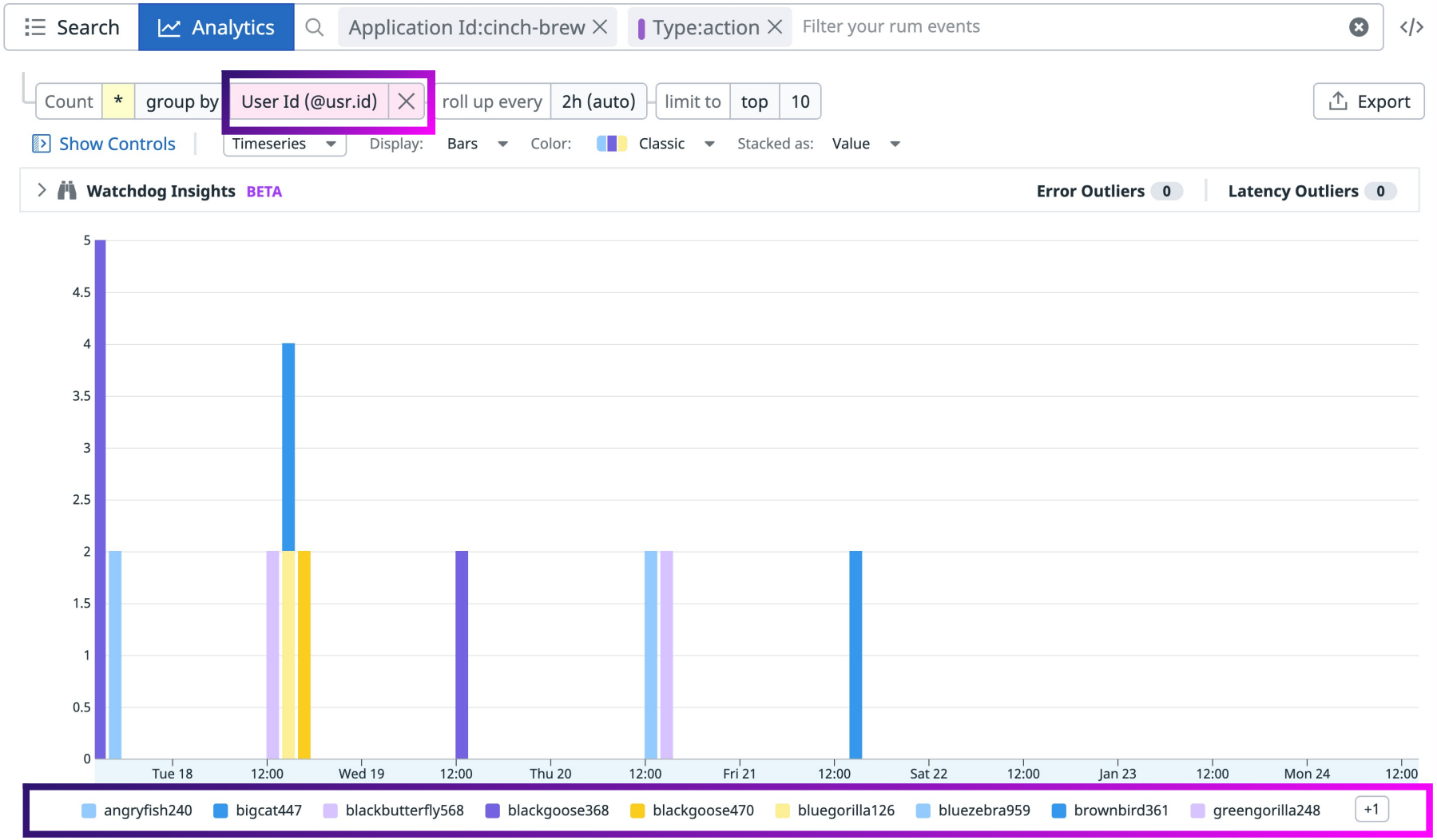
cinch

Or set a user

```
26        if (user) {
27          localStorage.setItem("user", JSON.stringify(user));
28          datadogRum.setUser({
29            id: user.username,
```

cinch

Graph all views by user*



* We use ids by default. We mask sensitive data by default. We don't set PII.

Instrumentation / tips

Use traces & RUM events as a rich dataset of events

Adopt custom instrumentation practices

Get your service boundaries right

Follow the o11y tool's principles

cinch

Instrumentation / challenges

TOO MANY DATA TYPES ON THE BACKEND IS CONFUSING

ENGINEERS ARE VERY USED TO LOGS

ENGINEERS NEED TO ‘SEE’ TRACES & RUM EVENTS WORKING TO BE CONVINCED

cinch

Instrumentation / challenges

01

Native SDKs very divergent from o11y tool SDKs

02

Overuse of metrics loses context

03

Getting everything right at once is hard

Milestone #2: Dashboards

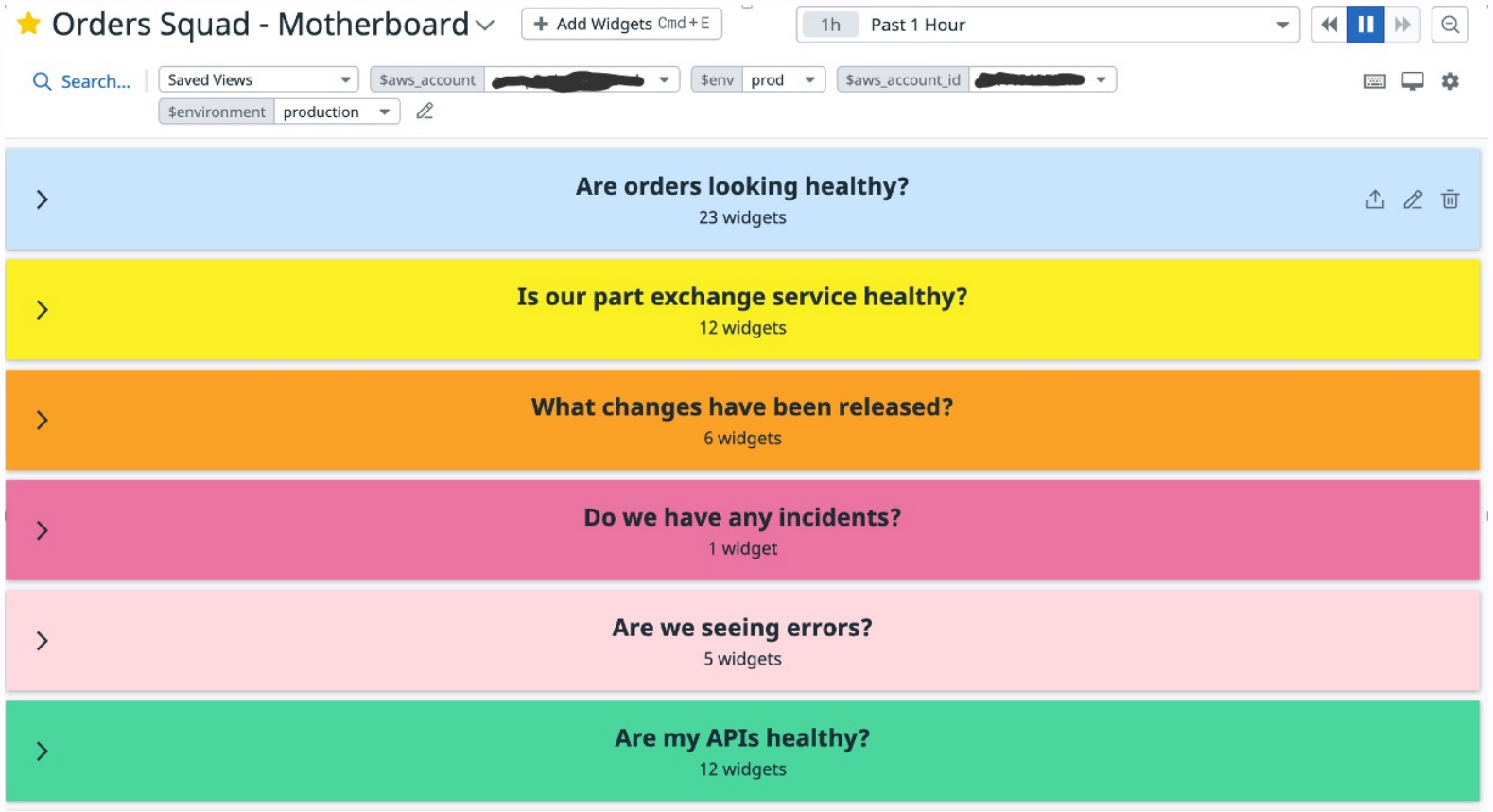
SQUADS OWN & EVOLVE THEIR DASHBOARDS

A VEHICLE TO GET SIGNALS FROM PRODUCTION

FOCAL POINTS AT STAND UPS WITHIN A SQUAD

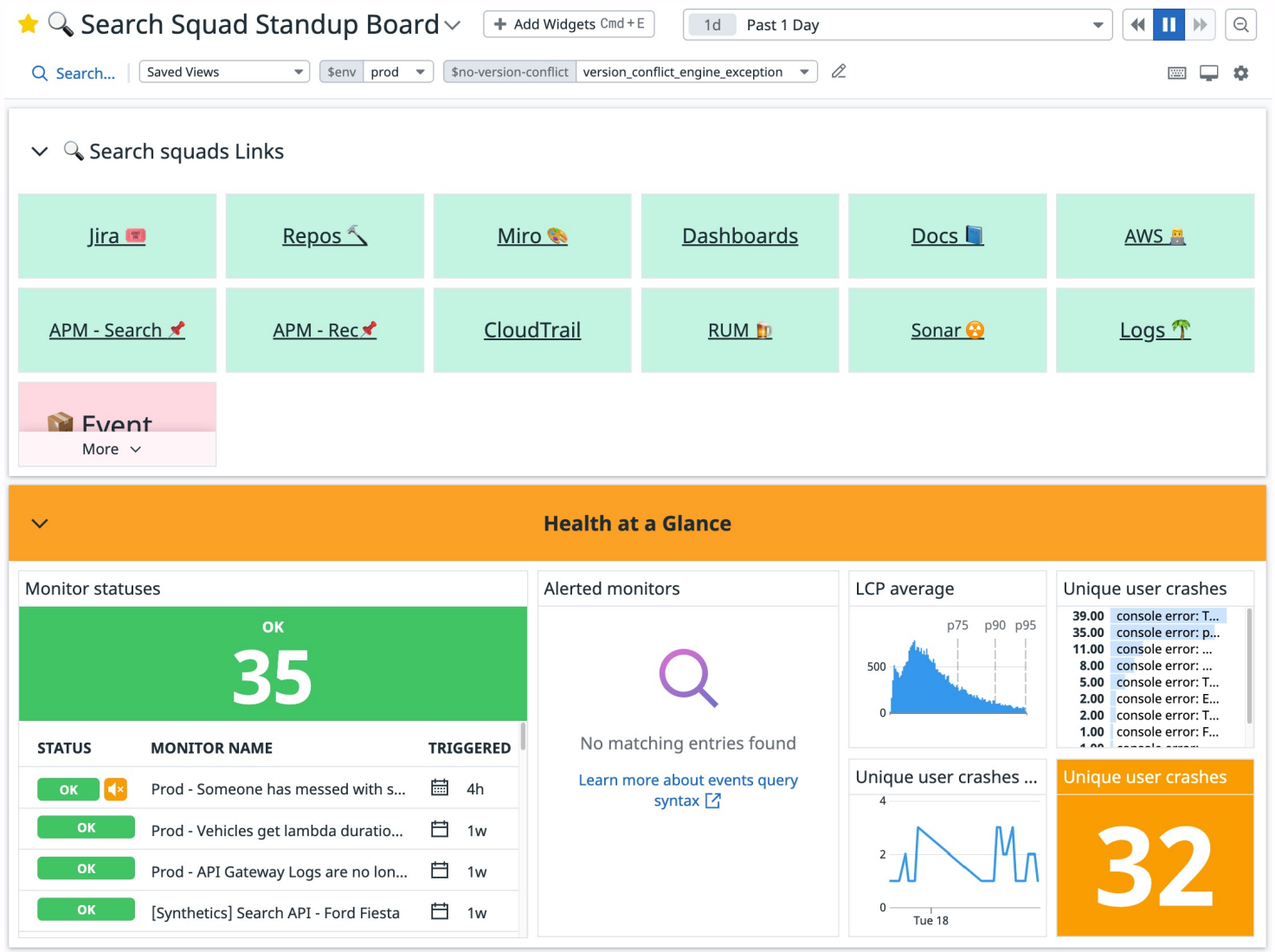
INFORMATION RADIATORS FOR EXTERNAL TO THE SQUAD AUDIENCE

Inclusive

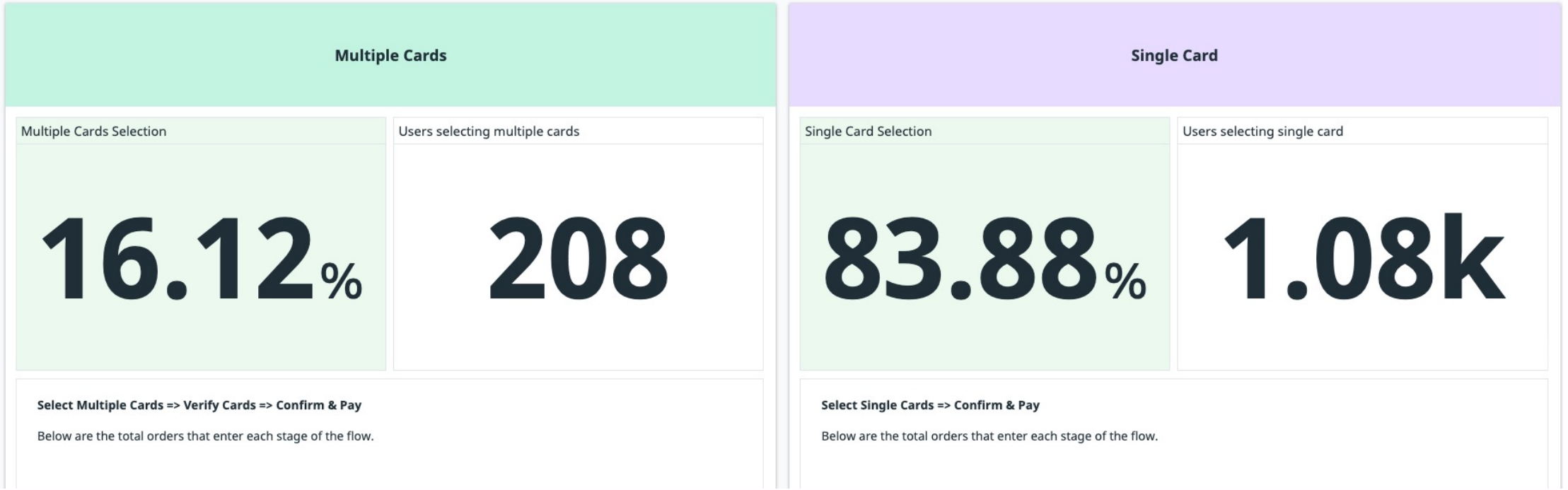


cinch

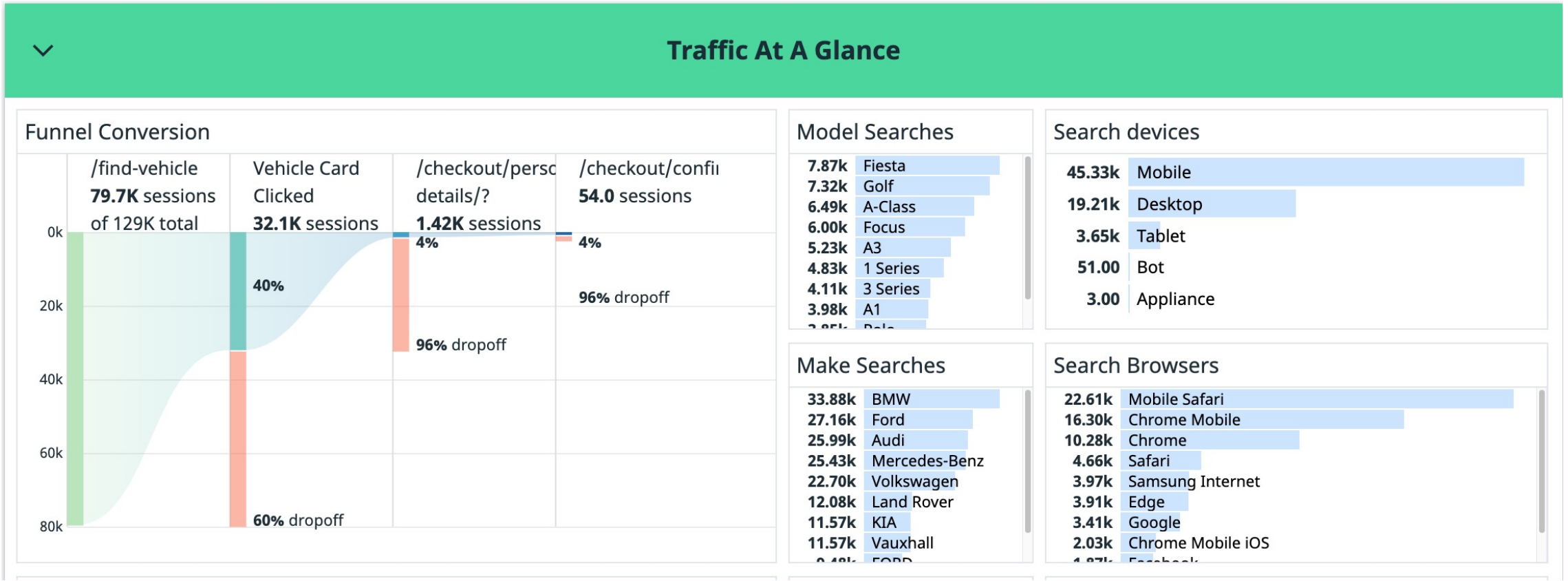
Comprehensive



Customer centric

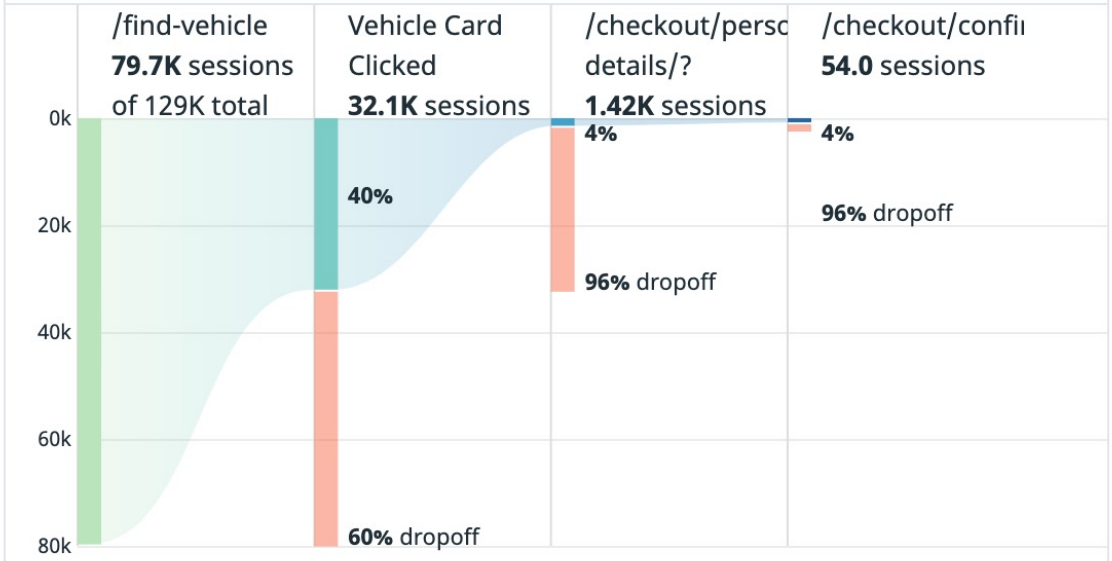


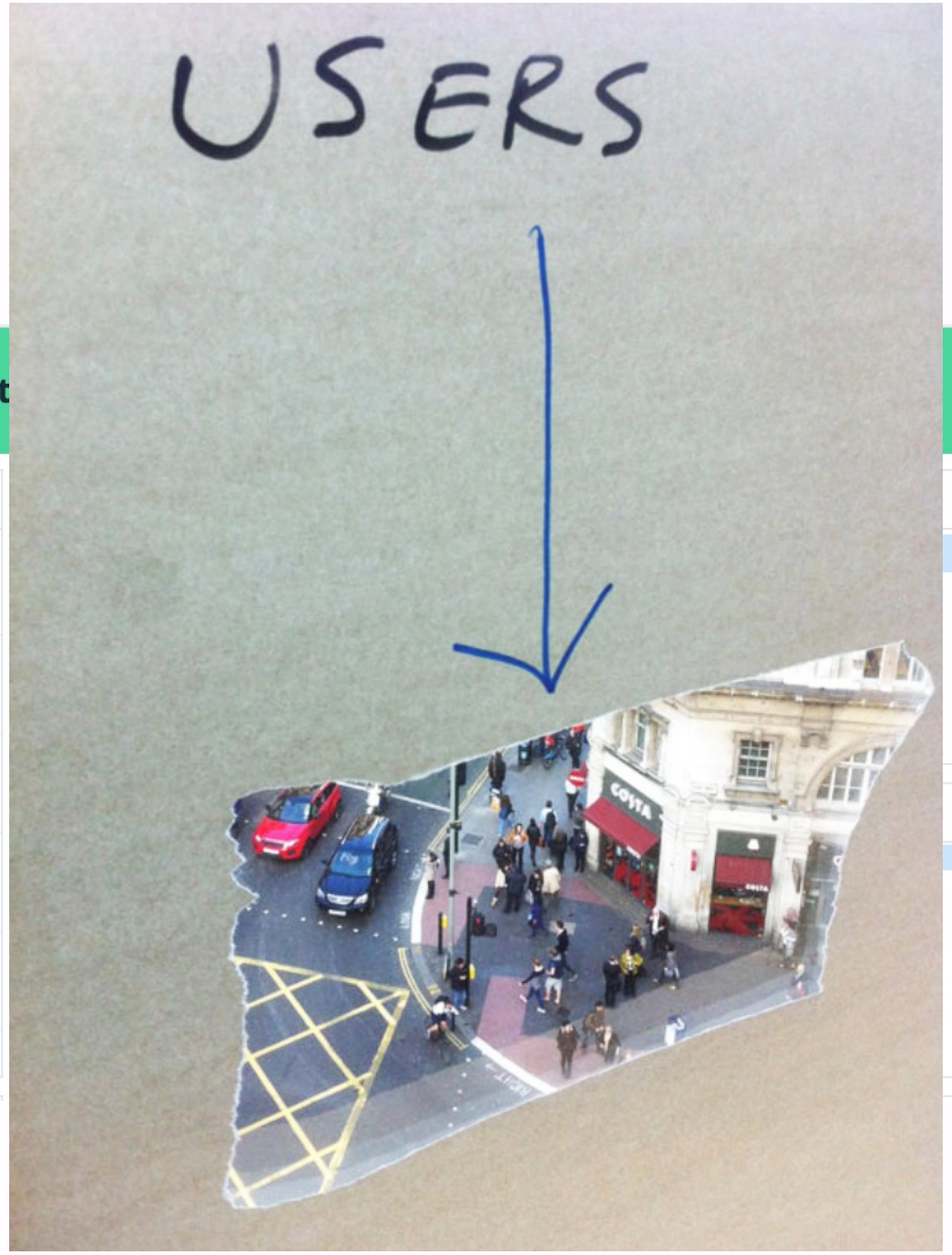
Teams care about funnels



Traffic At

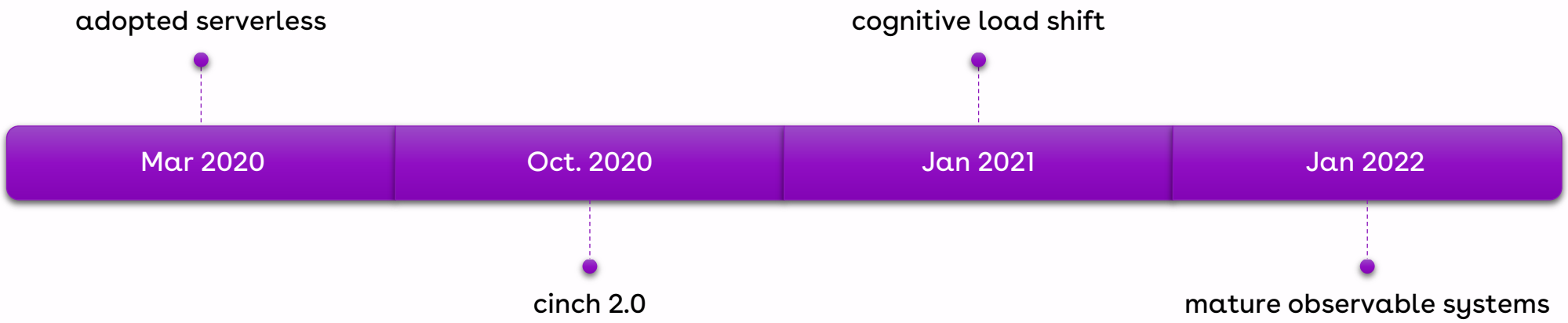
Funnel Conversion





A collective cognitive load shift.

Mature Observability



Serverless...

Reduces

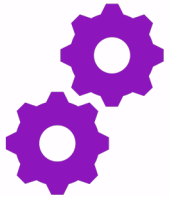
Caring about target infrastructure

Cognitive load

Requires

New engineering practices

New tooling

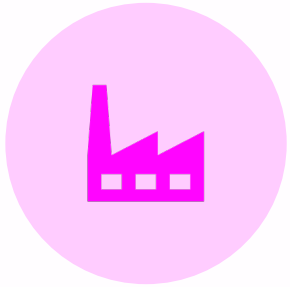


Upgrades

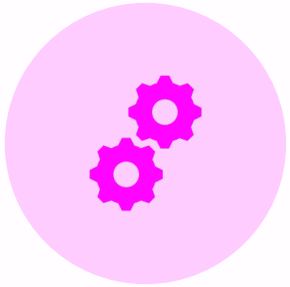
DevOps patterns and practices

Observability of systems

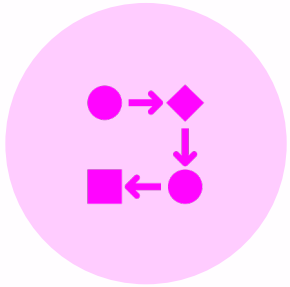
Organizational wins



Focused on production, rather than platform



Squads own the reliability of their production systems



Fast flow, instant, short feedback loops

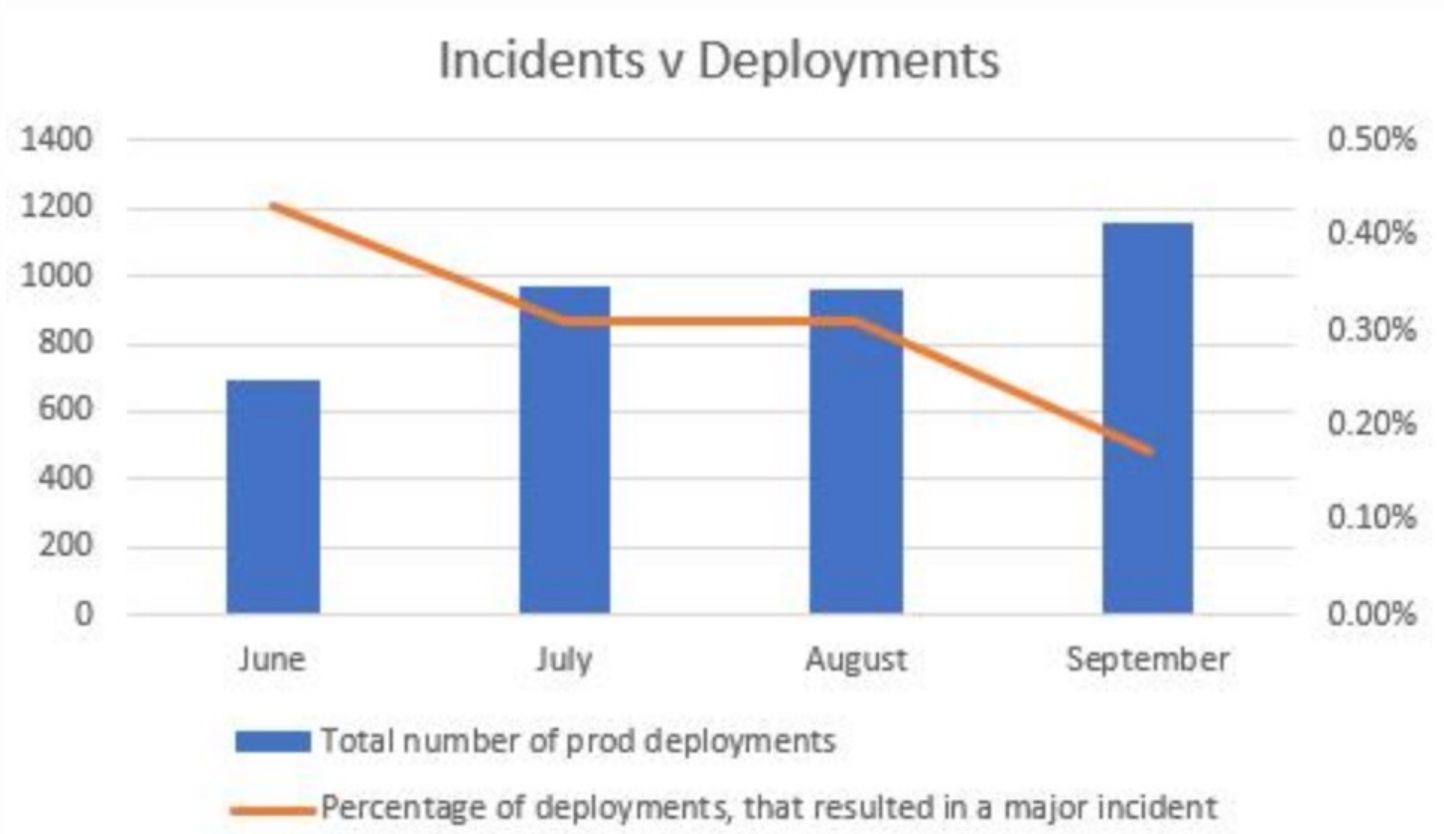
Experimentation led

Emergent practices

- **Review** dashboards after/during stand ups & stand downs
- **Review** monitoring and observability weekly
- **In-hours,** on-call rotas
- **Observability** workshops
- **Surface** various CI/CD quality metrics

cinch

Incidents are very rare



What next?

What next?

DevOps Community of Practice

Principal Engineers to support squads

What next?

Rely more on SLOs

Establish observability blueprints (cinch brew)

Optimize incident management

Deployment & Test observability

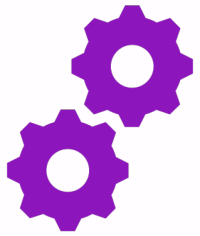
Security cognitive load shift

Public dashboards

cinch

My final 2 cents.

Choose tech stack that allows for a cognitive load shift towards observability



Adopt engineering practices & patterns that enable teams to truly have a DevOps mindset

Trust your teams, be patient

cinch

Thank you!

Get in touch

dev@toli.io (personal)

toli@cinch.co.uk (work)

@apostolis09 (twitter)

cinch