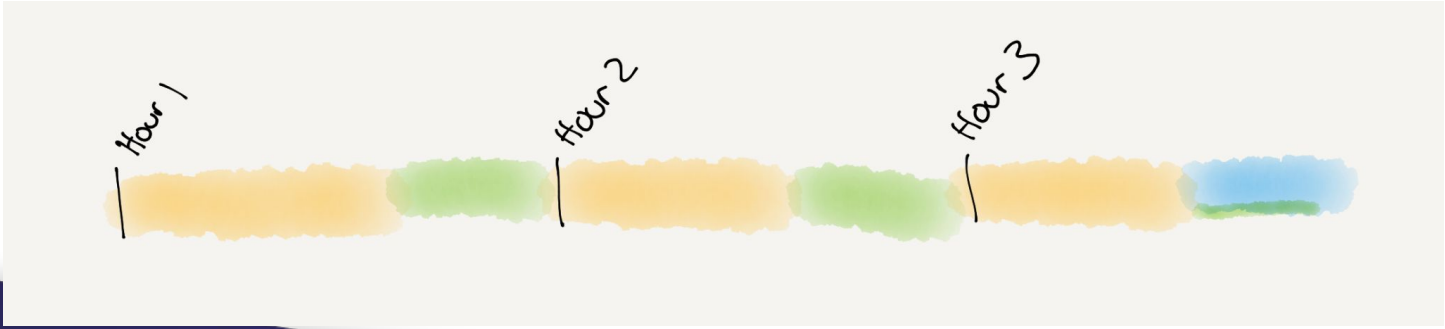
Introduction to Observability

15 June 2022

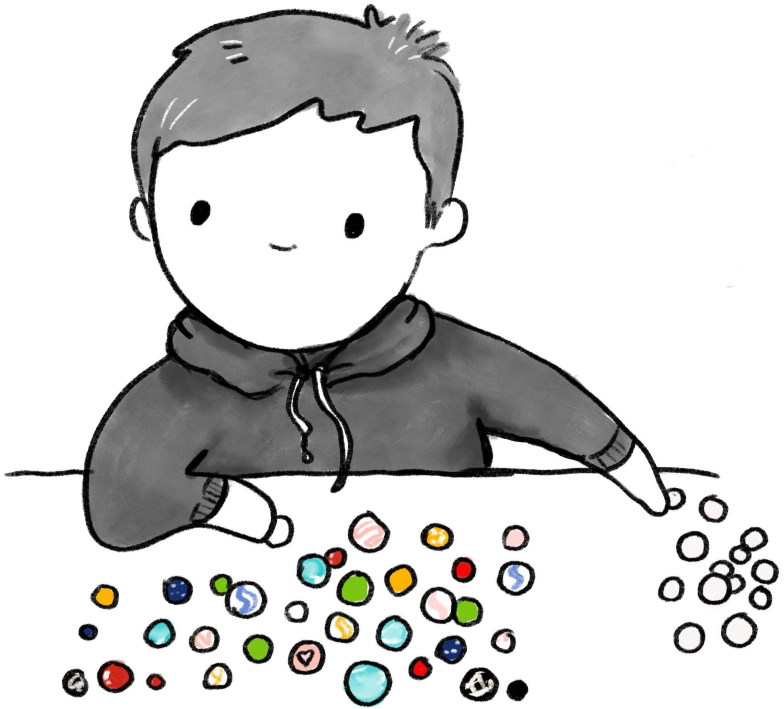
Today's agenda

- Housekeeping & Intros
- What is Observability?
- Get some data in!
- Answer questions with the data!
- What else might you do?
- Q&A



Some housekeeping items

- We are not recording this.
- Captions are available.
- In the main room: raise hands using reactji, or ask questions in zoom chat
- In breakout rooms, you are welcome to unmute video/audio.
- Also ask questions in Slack! They persist



Please join us in Pollinators Slack at #intro-to-o11y-workshop

Jessica Kerr

Principal Developer Advocate

David Marchante

Implementation Engineer

Martin Thwaites

Developer Advocate

Mike Terhar

Customer Architect

Phillip Carter

Senior PM, Telemetry

Why does Observability matter?

and how can it make your job easier?

Developers have a lot on our plates.

We want devs to:

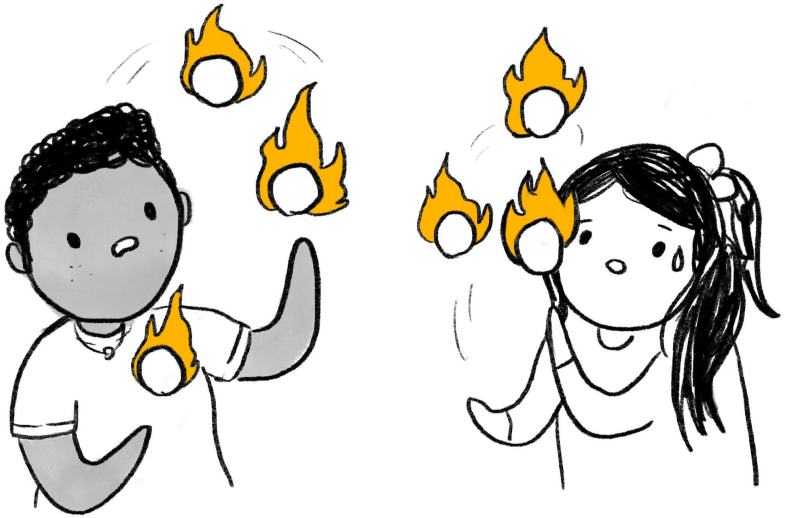
- Ship more often
- Produce better code
    - Spend less time debugging and more time creating value
- Decrease downtime

Working in production is challenging.

We're afraid we'll break things.

We've forgotten what we've built by the time it ships.

We can't figure out what's wrong and fix it.



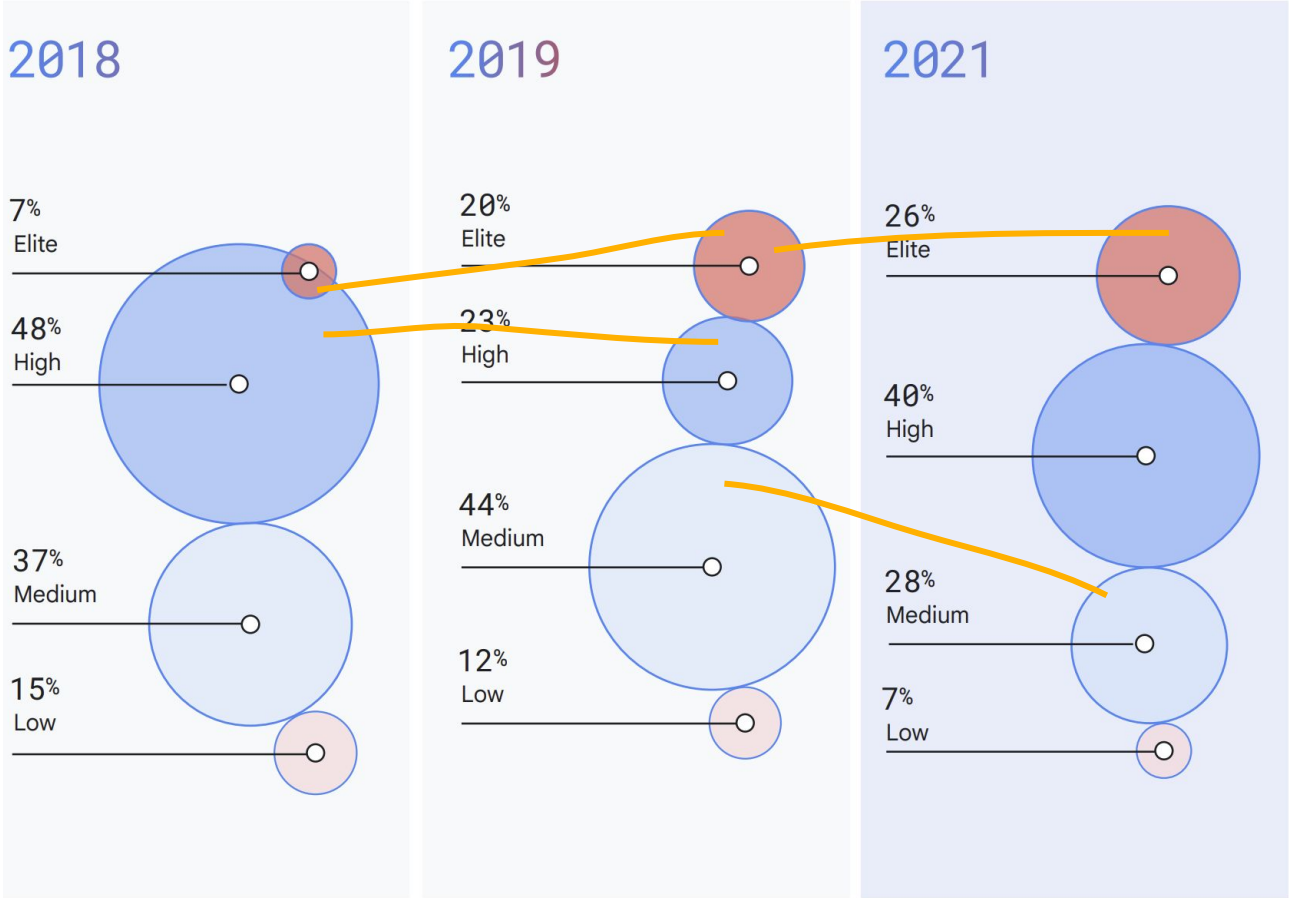
But some organizations have solved it!

DevOps & Progressive Delivery culture have the answers to some of these problems.

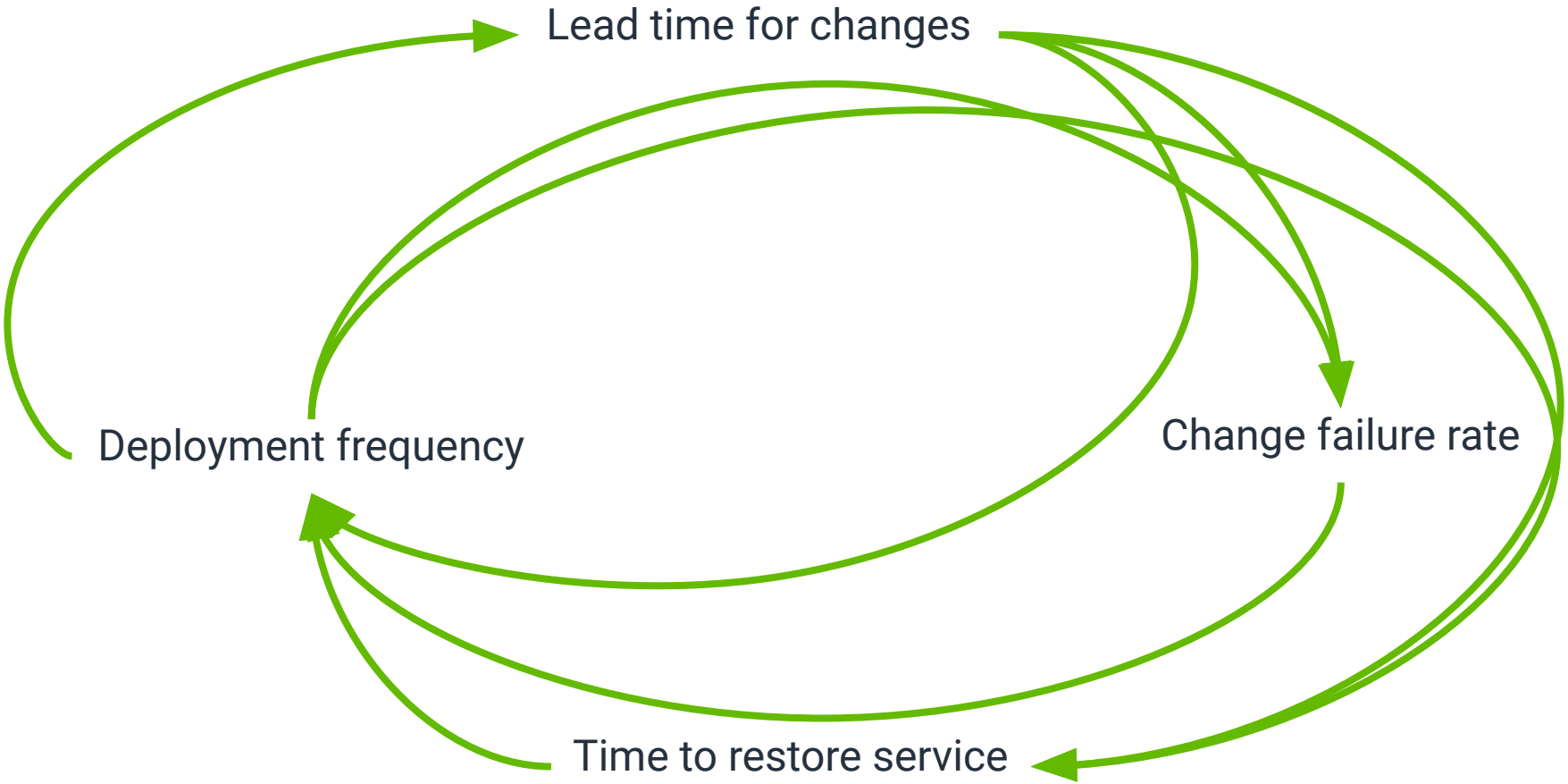
And a growing fraction of our industry is moving faster *and* safer!

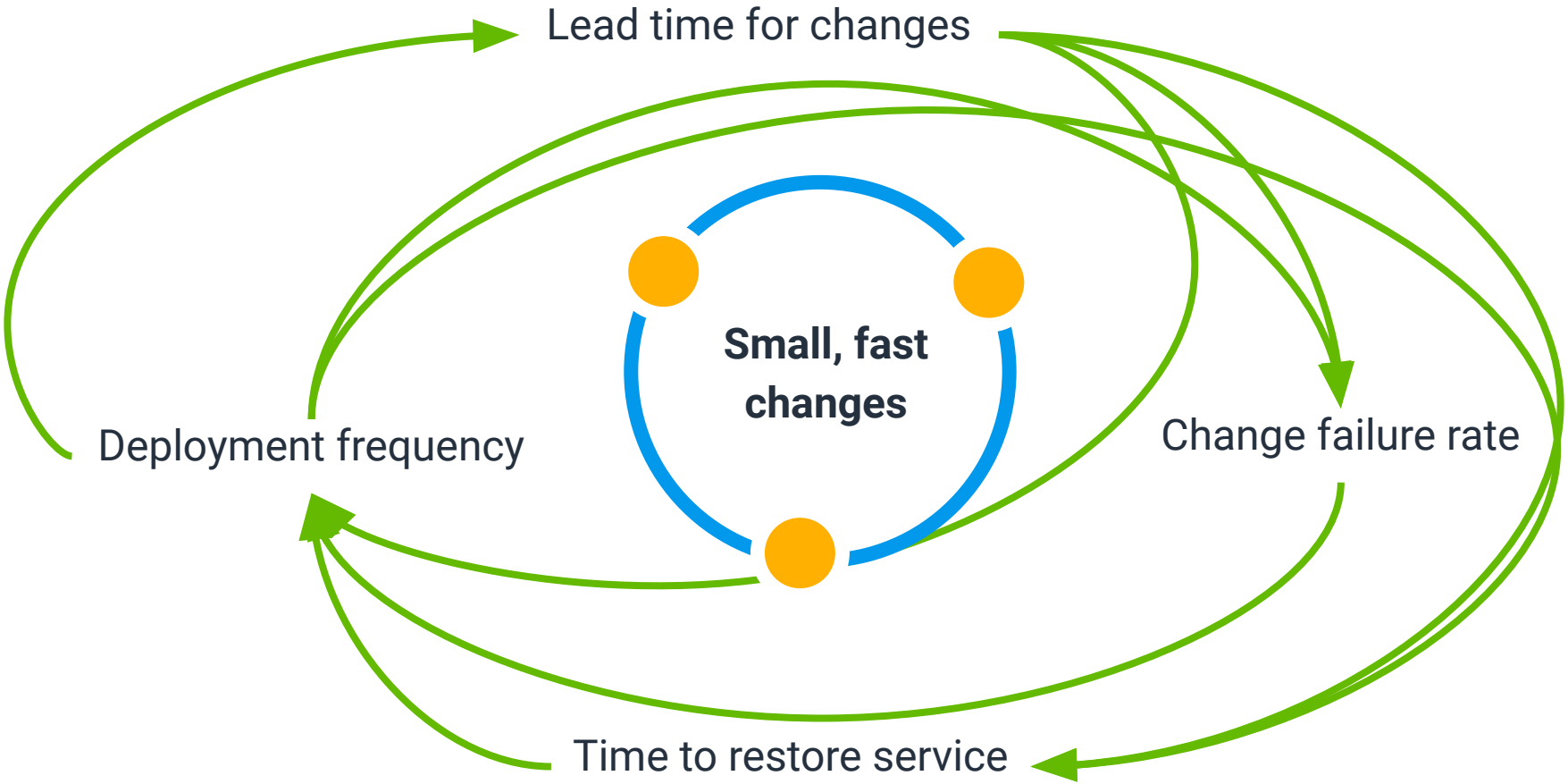
Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Accelerate: State of DevOps 2021, Smith et al, Google



Accelerate: State of DevOps 2021, Smith et al, Google





Monitoring and observability

As with previous years, we found that monitoring and observability practices support continuous delivery. Elite performers who successfully meet their reliability targets are 4.1 times more likely to have solutions that incorporate observability into overall system health. Observability practices give your teams a better understanding of your systems, which decreases the time it takes to identify and troubleshoot issues. Our research also indicates that teams with good observability practices spend more time coding. One possible explanation for this finding is that implementing observability practices helps shift developer time away from searching for causes of issues toward troubleshooting and eventually back to coding.

Accelerate: State of DevOps 2021, Google

Monitoring and observability

As with previous years, we found that monitoring and observability practices support continuous delivery. Elite performers who successfully meet their reliability targets are 4.1 times more likely to have solutions that incorporate observability into overall system health. Observability practices give your teams a better understanding of your systems, which decreases the time it takes to identify and troubleshoot issues. Our research also indicates that teams with good observability practices spend more time coding. One possible explanation for this finding is that implementing observability practices helps shift developer time away from searching for causes of issues toward troubleshooting and eventually back to coding.

Accelerate: State of DevOps 2021, Google

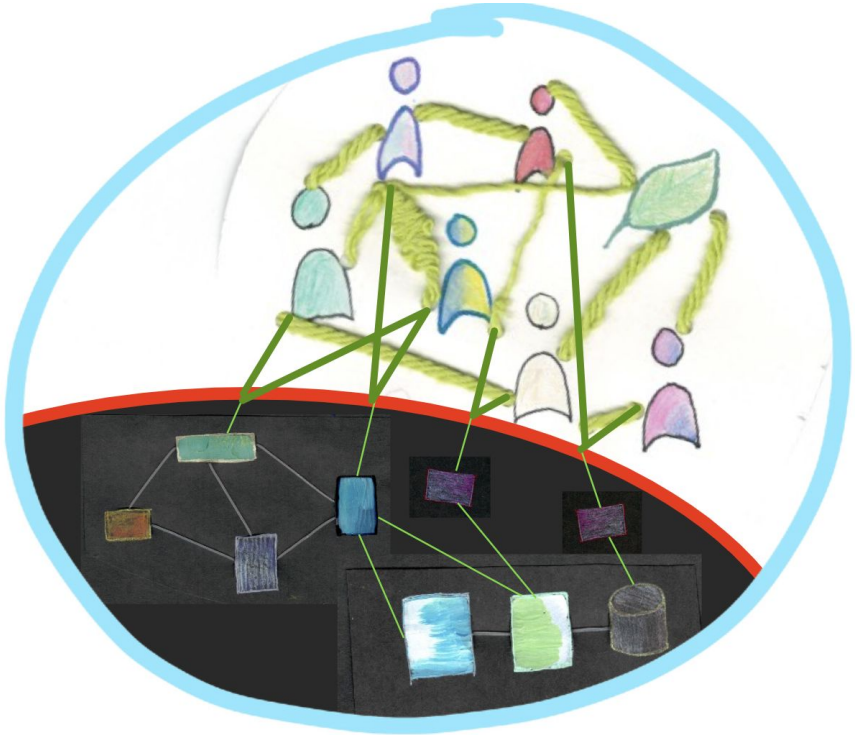
Your software and tools are on your team.

The running software, plus the tools, plus the people on our team form a system,

a *symmathesy*: a learning system made of learning parts.

Our software and tools learn from us, because we change them.

Make the software teach you about itself and the world!

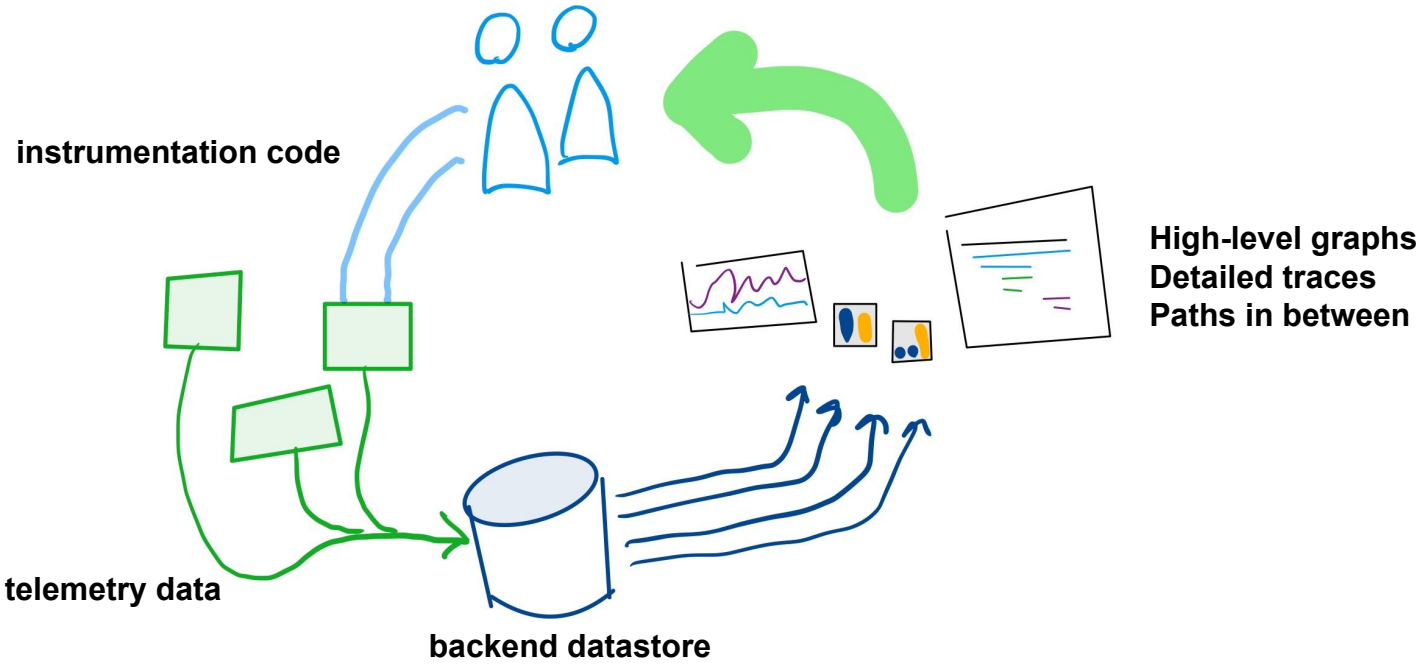


so adapt your culture, not just the tools.

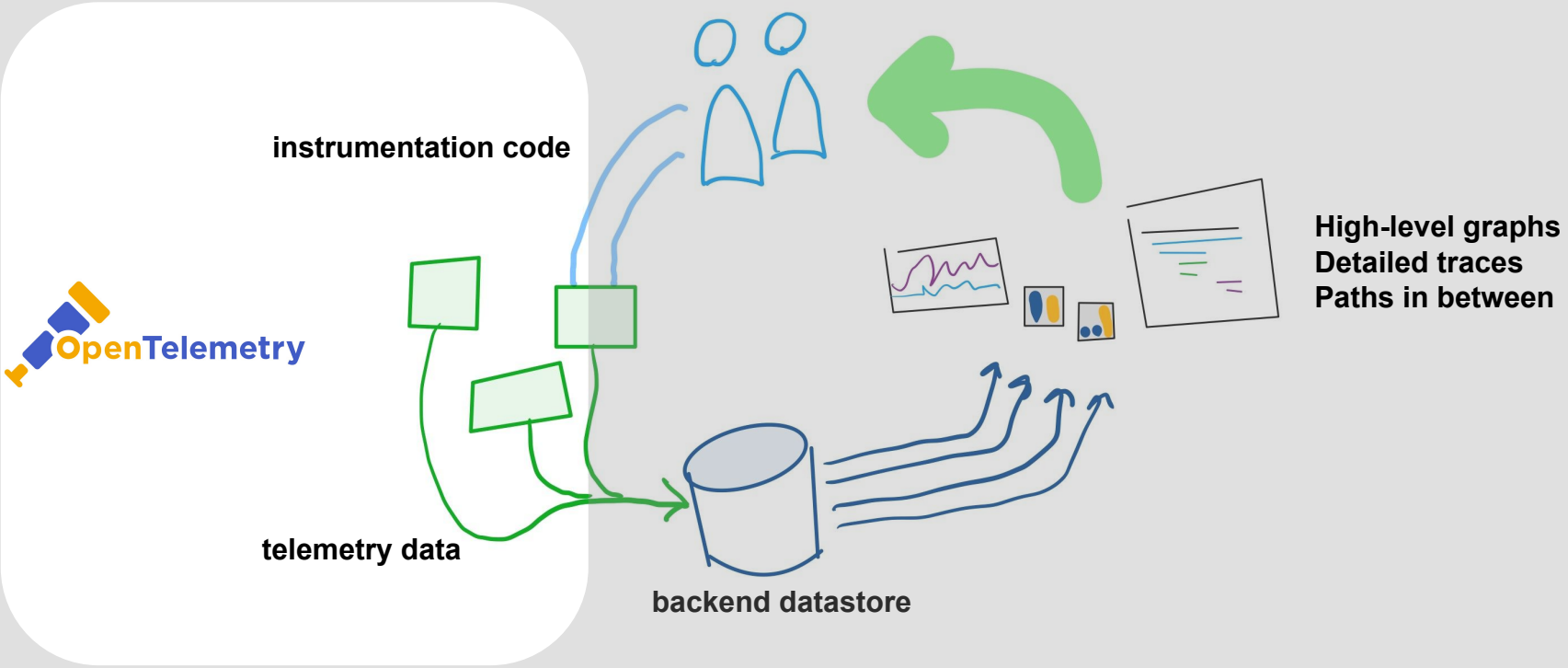
We need a culture of *curiosity*, not of fear.

We need to accelerate feedback cycles.

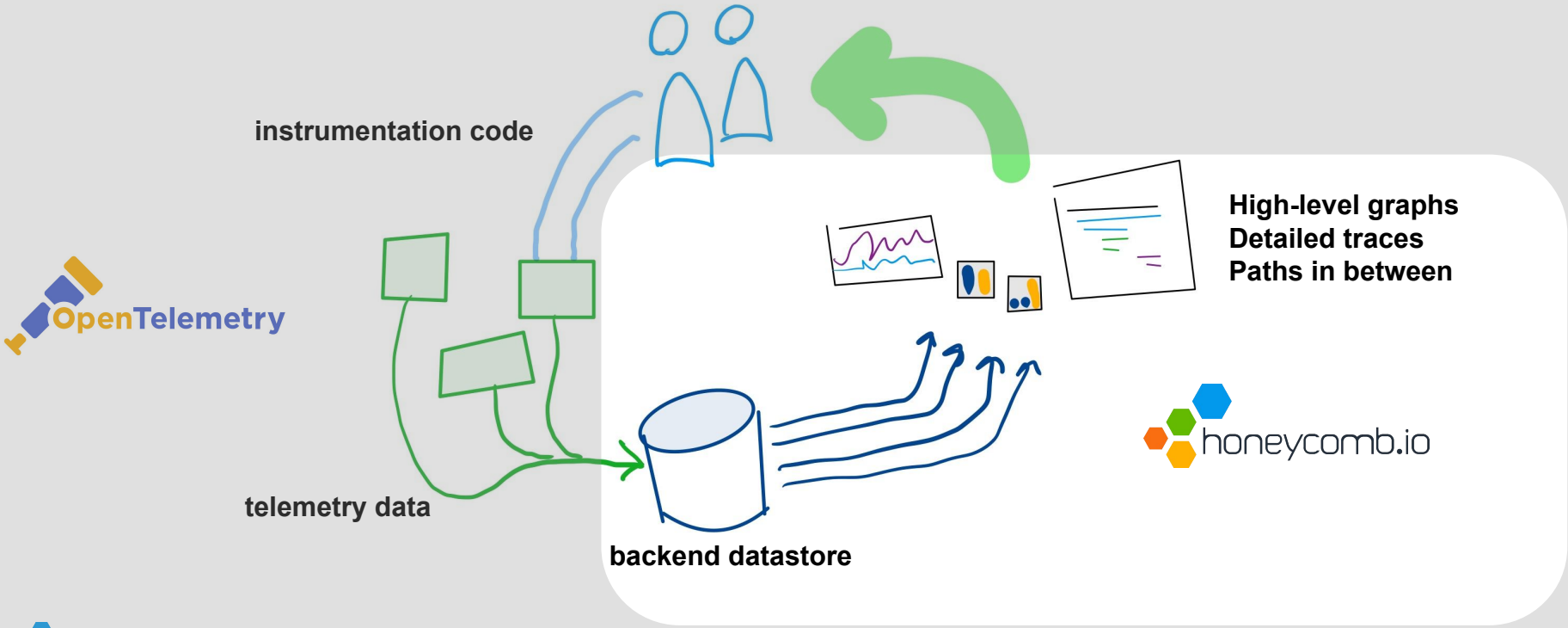
How does observability work?



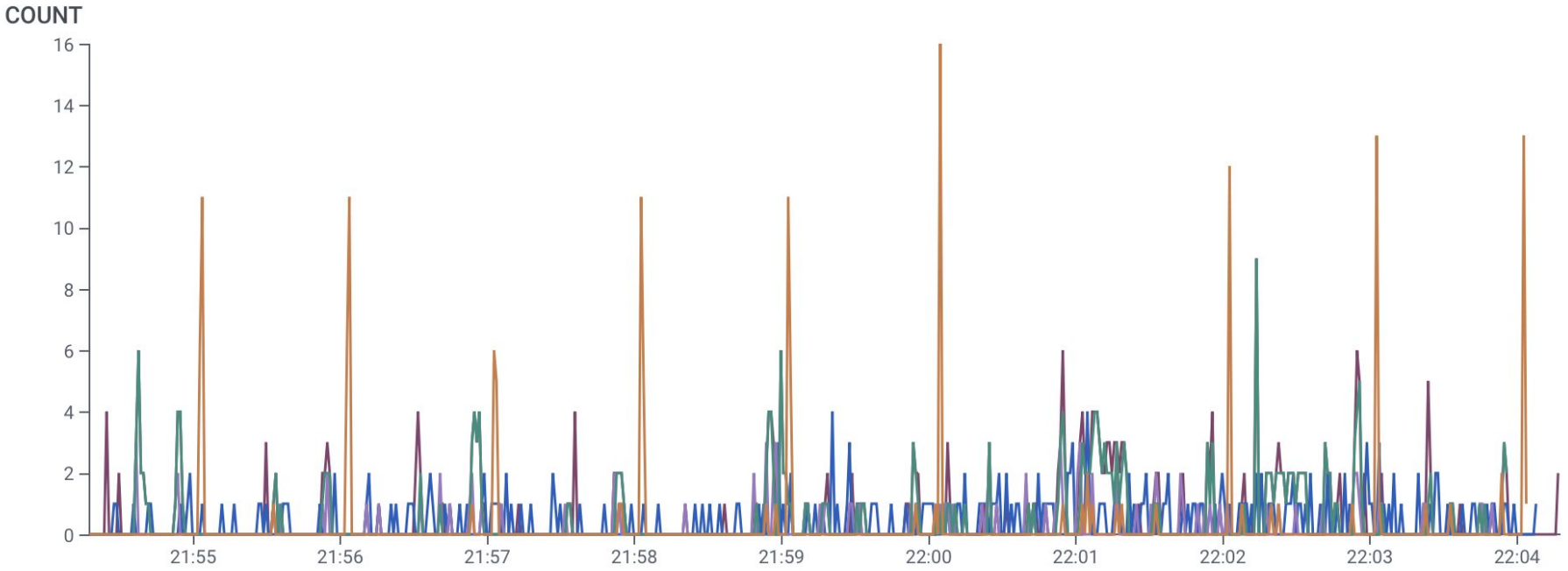
How does observability work?



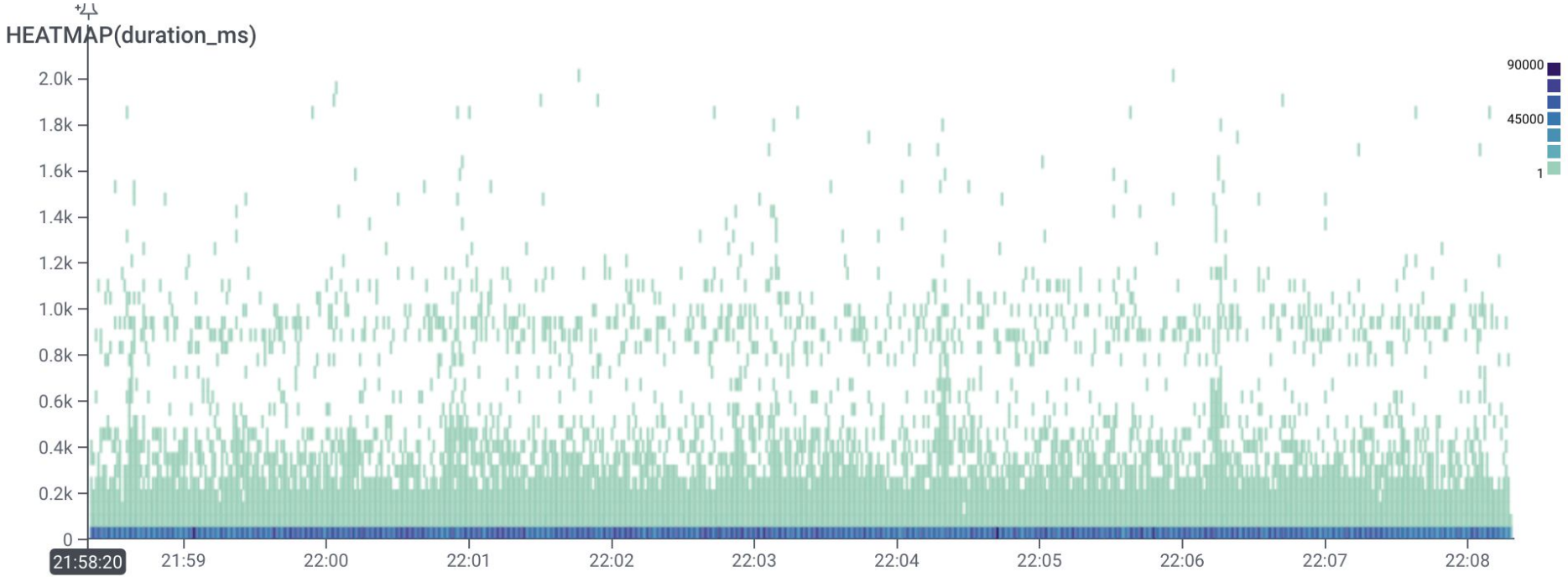
How does observability work?



Telemetry feeds graphs



Telemetry feeds heatmaps



Telemetry feeds traces

Trace 041f1cbd0b6095e0bdc73889ac20120c at 2021-10-29 10:42:44

Rerun

Search spans

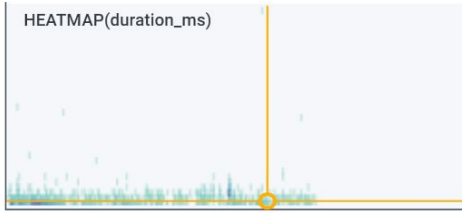
Fields

name	service_name	0s	0.005s	0.01s	0.015s	0.02136s
20 github.com/honey...ueryRunStatus-fm	poodle	21.36ms				
findByID	poodle	1.820ms				
launchdarkly.BoolVariations	poodle	0.2184ms				
launchdarkly.NumberVariation	poodle	23.9µs				
launchdarkly.JSONVariation	poodle	50.2µs				
launchdarkly.JSONVariation	poodle	36.0µs				
FindTeamBySlug	poodle	2.095ms				
FindWriteKeysByTeam	poodle	2.056ms				
CheckMembership	poodle	1.659ms				
GetRole	poodle	1.693ms				
ExternalAuthProvider	poodle	1.658ms				
launchdarkly.BoolVariations	poodle	0.1779ms				

poodle >

github.com/honeycombi
o...andler).QueryRunStat
fm

Distribution of span duration



Fields

trace.

str trace.span_id

df9fc50200e3ad1c

str trace.trace_id

041f1cbd0b6095e0bdc73889ac20120c

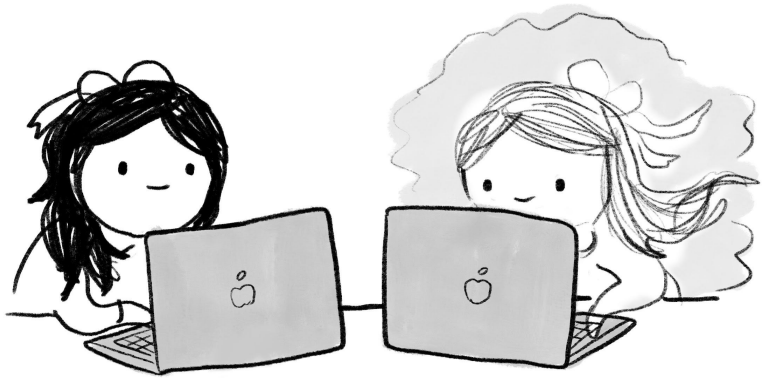
Observability requires telemetry.

Let's get some data in!

Section Overview

In this section, we will:

- Run a demo app instrumented with OpenTelemetry
- Get a Honeycomb Account
- Send telemetry from the demo app to Honeycomb



Get the Demo App

glitch.com/@honeycombio

- Node.js
- Python
- Go

https://github.com/orgs/honeycombio/repositories?q=intro

- Java
- .NET
- Node.js
- Python
- Go

(clone & run locally)

- Java
- .NET
- Node.js
- Python
- Go

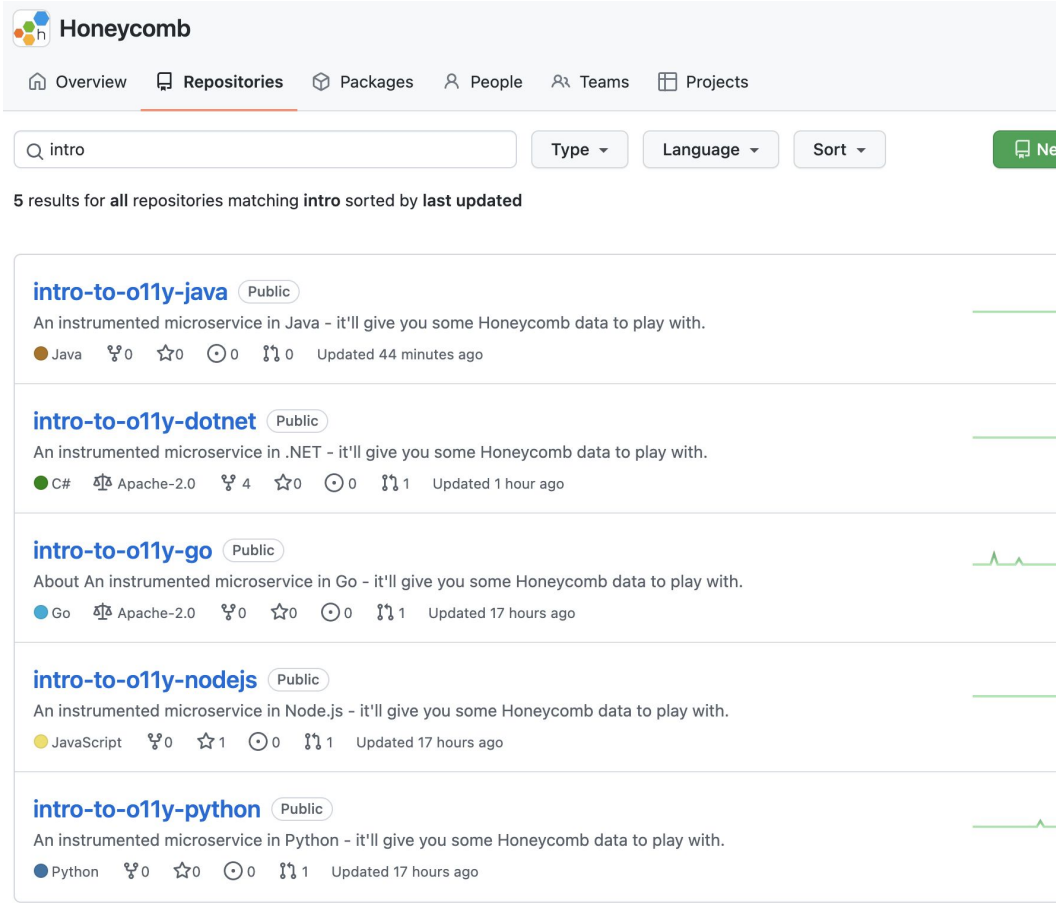
Choose the GitHub Repo for your language

Go to https://github.com/honeycombio

Find a repository…

“intro”

Choose the repository (repo) for your language.

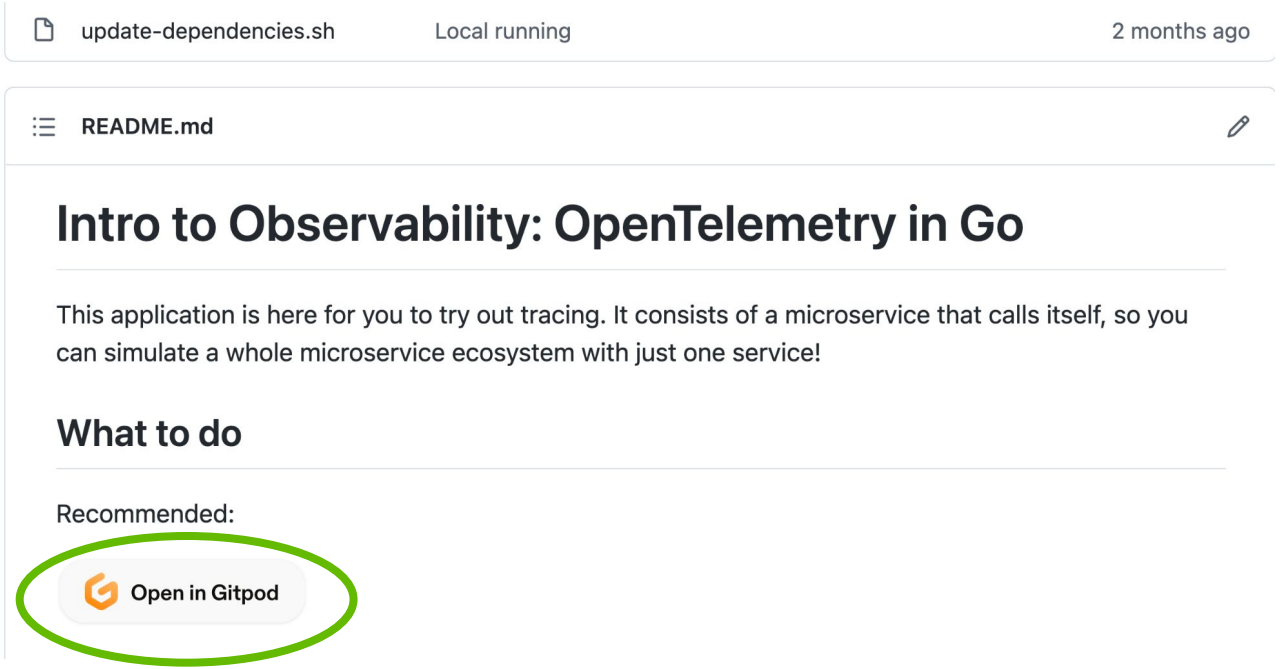


Choose the GitHub Repo for your language

Go to https://github.com/orgs/honeycombio/repositories?q=intro

Choose the repository (repo) for your language.

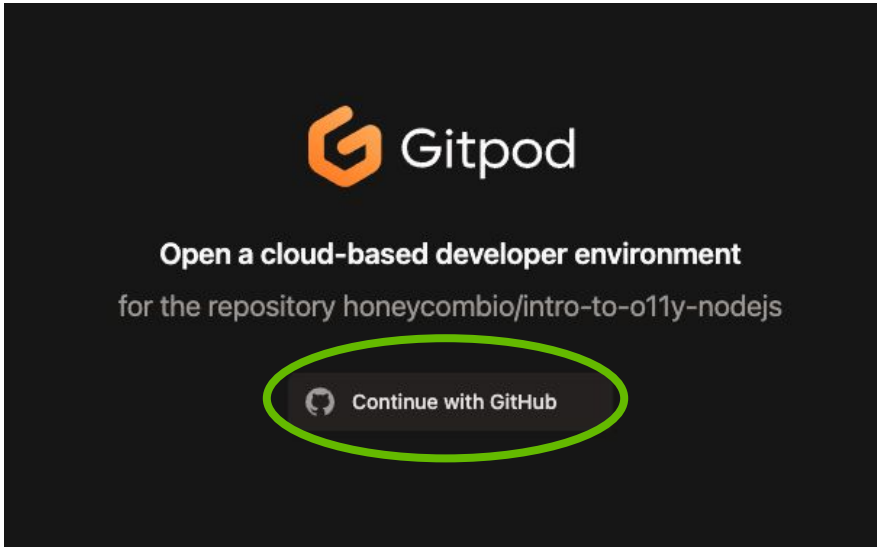
Scroll down and select the Gitpod button in the repo's README.



Log in

A screen will appear and prompt you to sign-in with GitHub.

Select **Continue with GitHub**



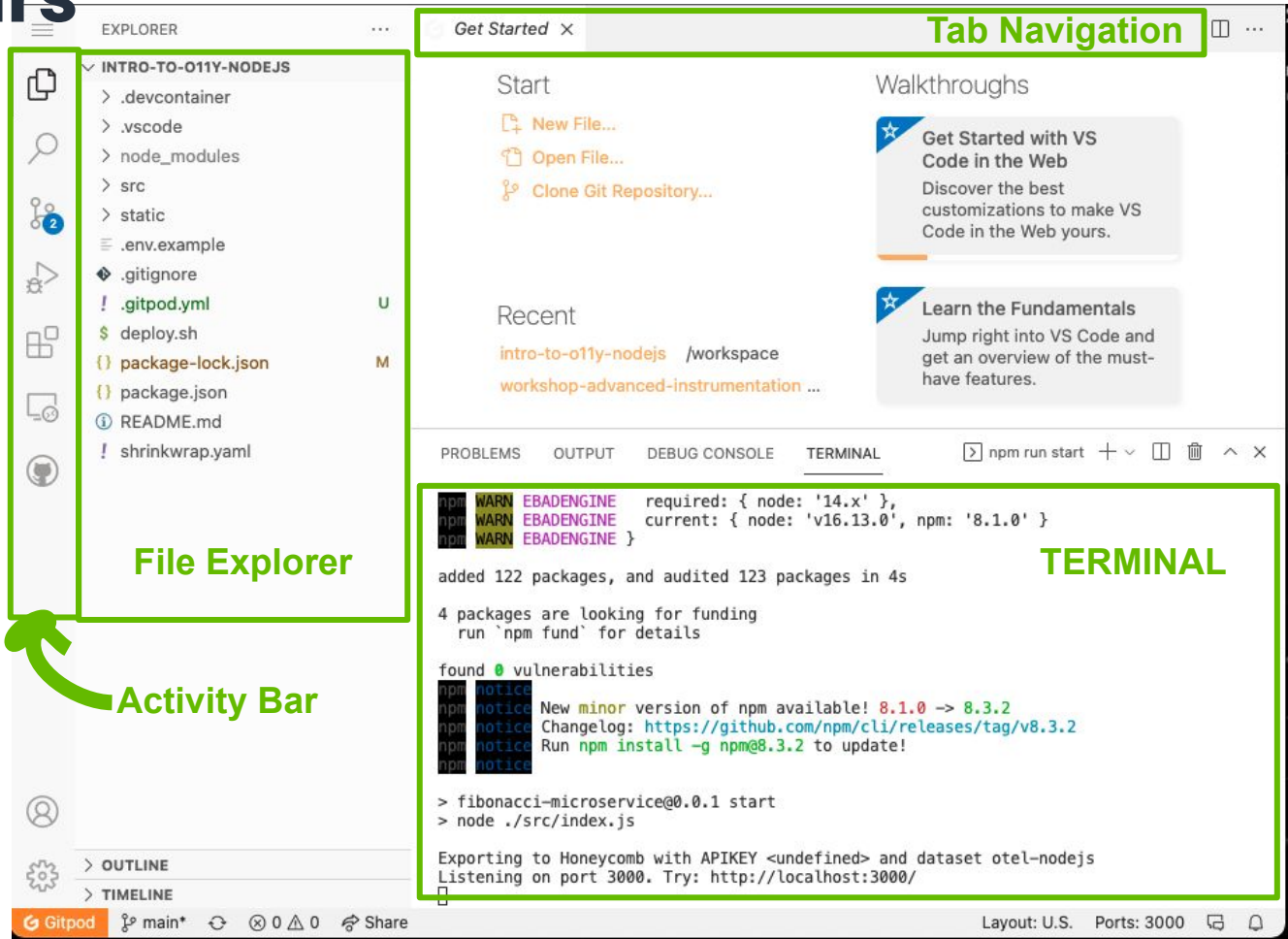
Gitpod Appears

Layout Reflects VSCode

Important areas include:

- Activity Bar
- File Explorer
- Tab Navigation
- Terminal

Select `README.md` in the File Explorer to see instructions.



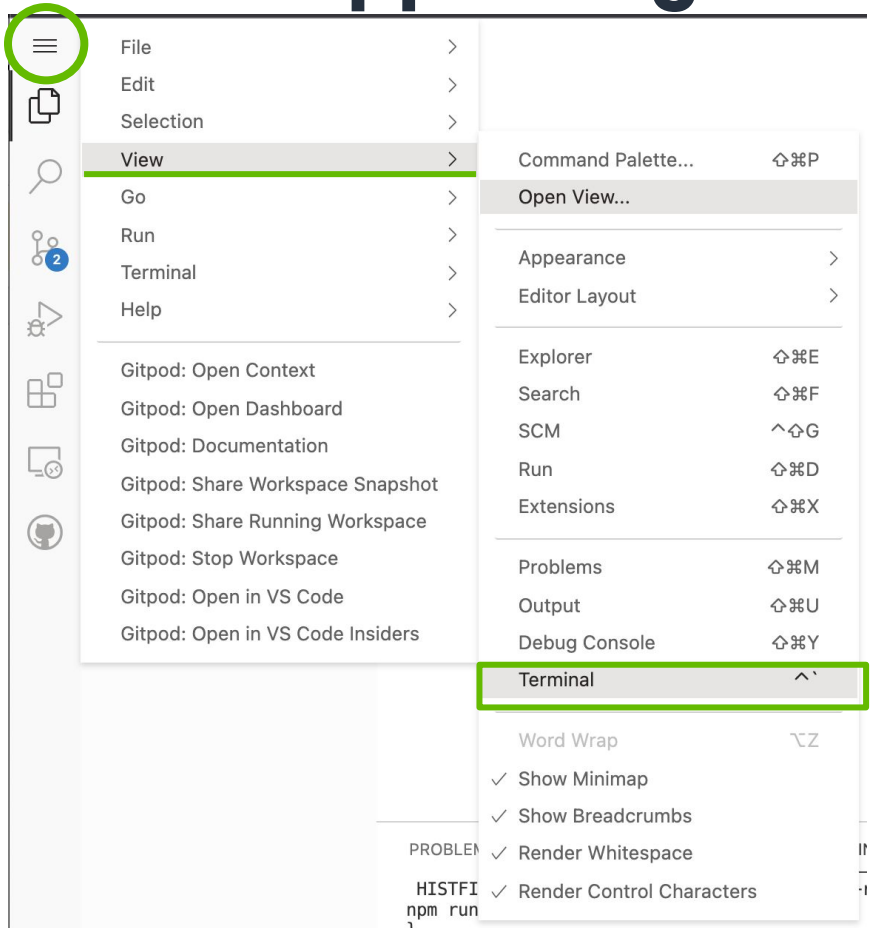
Troubleshooting: Terminal not appearing

Go to the upper left corner to the

Hamburger Menu > View > Terminal

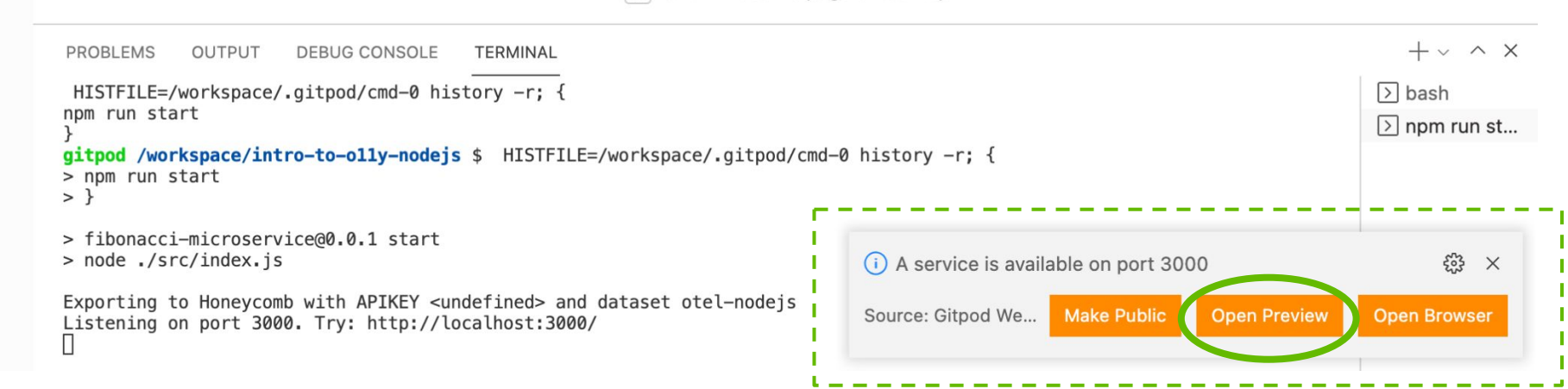
The terminal window will appear in the bottom half of the screen

Shortcut: Ctrl+J (Cmd+J on Mac)



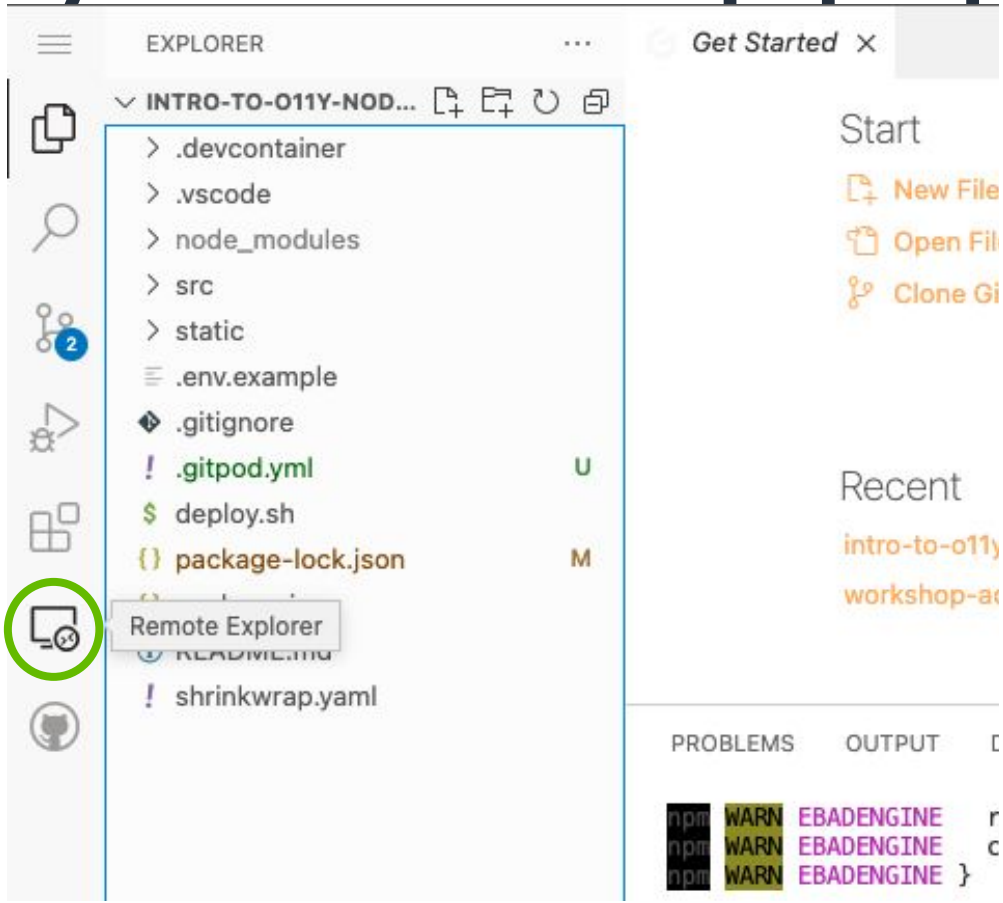
How to Open an App Preview

In the Terminal area, a "Service is available on port xxxx" pop-up may appear.
Select **Open Preview** to open the app display in a new tab.



Troubleshooting: If you don't see the pop-up

Go to **Remote Explorer**.

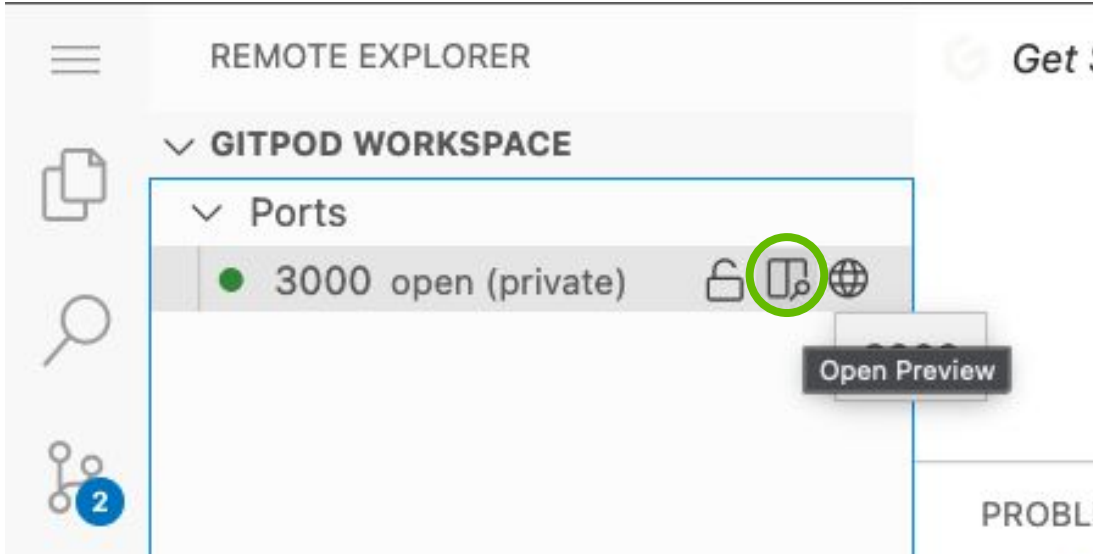


Troubleshooting: If you don't see the pop-up

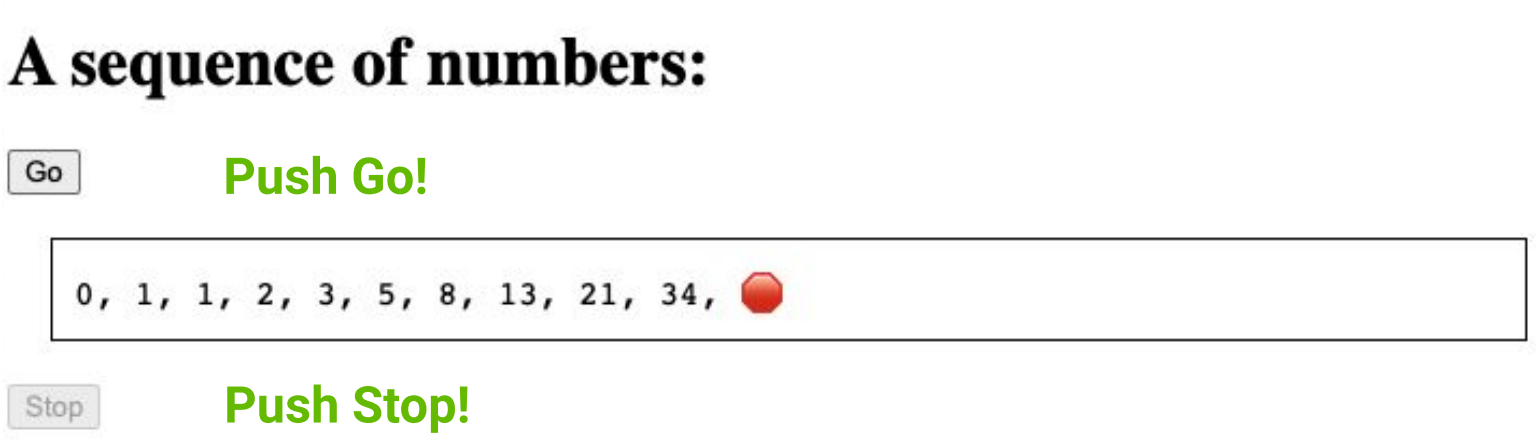
Go to Remote Explorer.

Expand *Ports* to see the listed port number. For example, *3000*

Choose the middle ***Open Preview*** icon for the app to appear in a new tab.



Try out the app

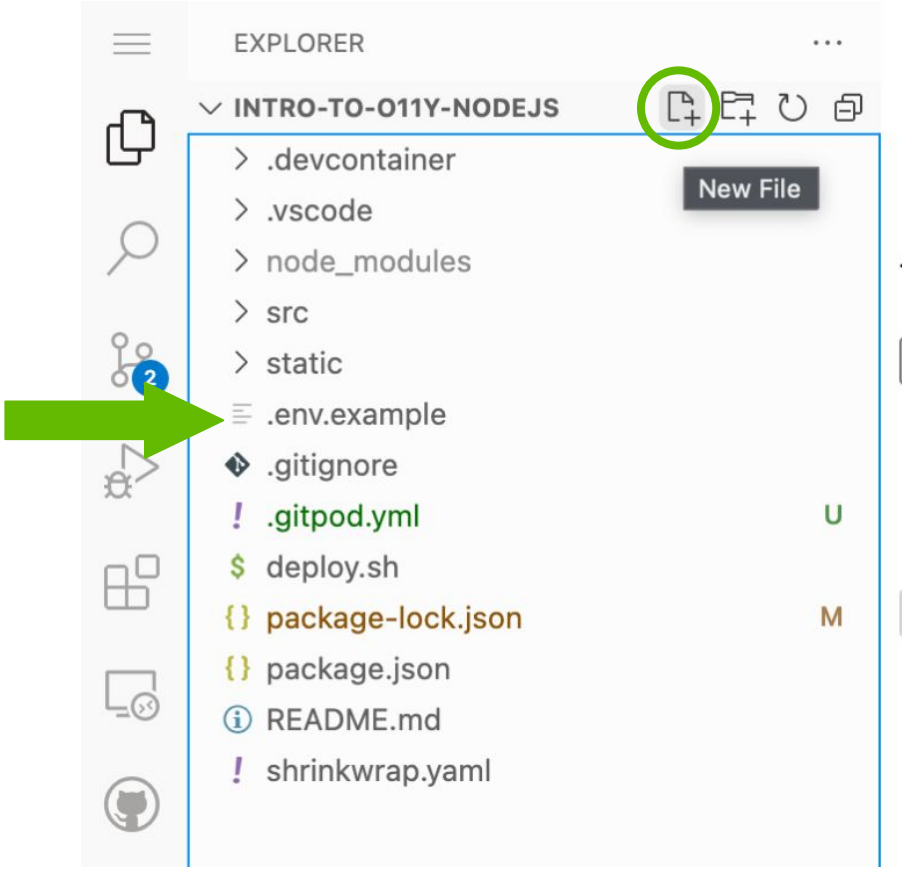


Add credentials to send to Honeycomb

Provide your credential variables in the **`.env`** file.

Using the File Explorer:

- Create a new file named **`.env`**
- Copy the contents of **`.env.example`** into the `.env` file
- Replace the credential variables in the `.env` file as needed



Add credentials to send to Honeycomb

Provide your credentials!

- `HONEYCOMB_API_KEY` identifies your team and environment.
  - Enter your API key from the Honeycomb UI

- `SERVICE_NAME` groups your data

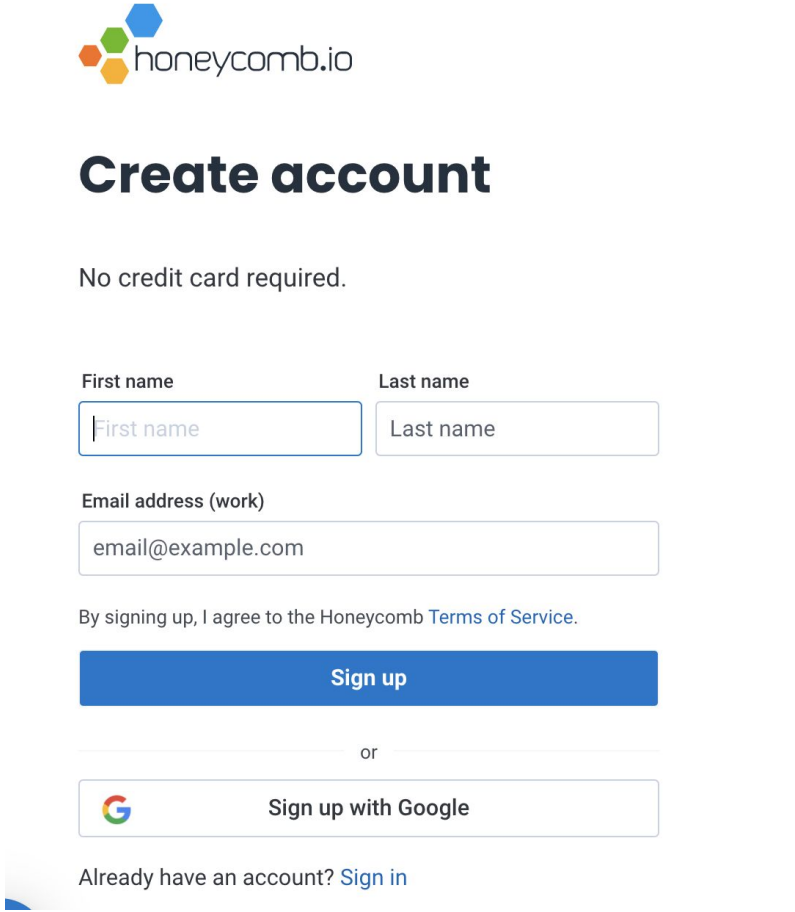
.env

```
1  # Honeycomb API key.
2  export HONEYCOMB_API_KEY=your-api-key-goes-here
3
4  # Service name (opentelemetry standard field). Can b
5  SERVICE_NAME=fib-microsvc
6
```

Get a Honeycomb account

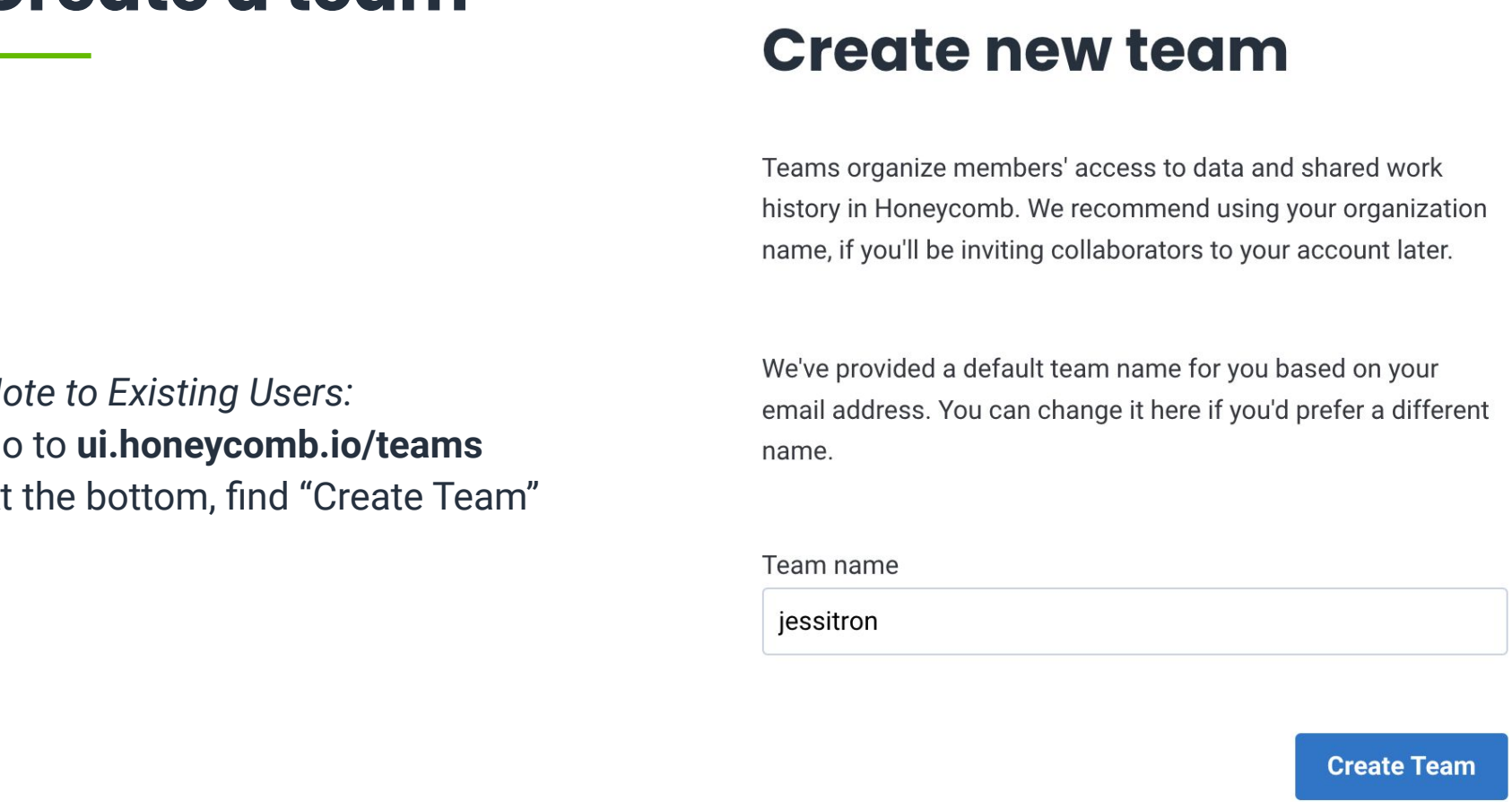
https://honeycomb.io/signup

Use the same email you used to register for this workshop.



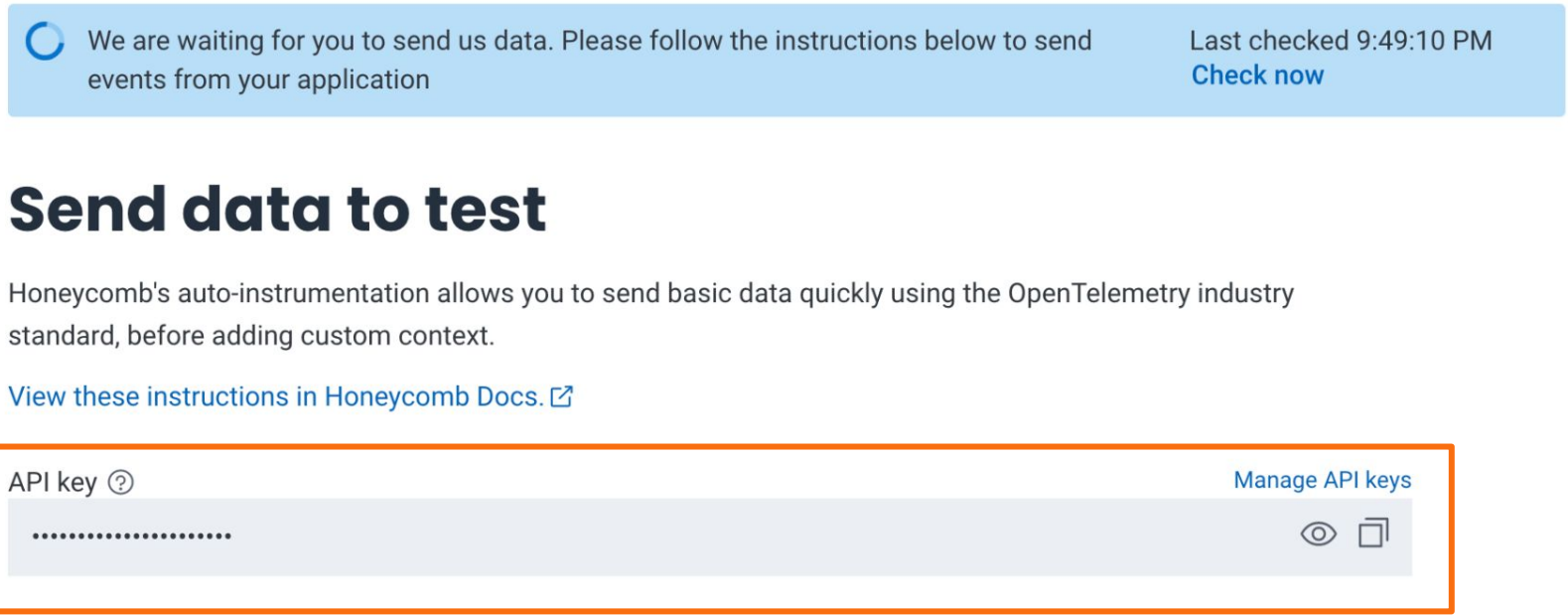
Create a team

Note to Existing Users:
Go to **ui.honeycomb.io/teams**
At the bottom, find “Create Team”



Get a Honeycomb API Key

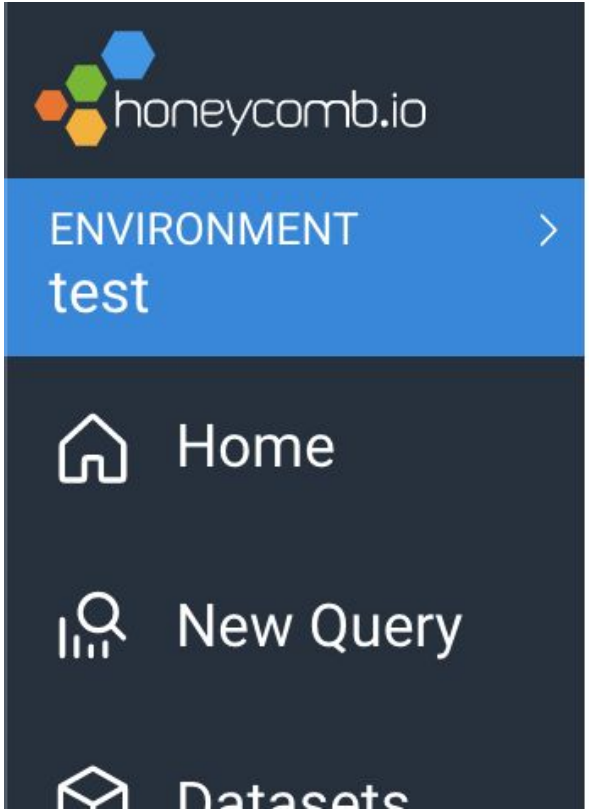
The first time you create a team as a new user, it takes you directly to your API Key.



Get a Honeycomb API Key

If you already have an account:

1. Click the environment selector at the top left
2. Choose **Manage environments** at the bottom.
3. View API Keys.



Add credentials to send events to Honeycomb

Provide your credentials!

- HONEYCOMB_API_KEY identifies your team and environment.
  - Enter your API key from the Honeycomb UI

- SERVICE_NAME groups your data

.env

```
1  # Honeycomb API key.
2  export HONEYCOMB_API_KEY=your-api-key-goes-here
3
4  # Service name (opentelemetry standard field). Can b
5  SERVICE_NAME=fib-microsvc
6
```

Re-run Your App

1)If your app is still running, stop it with `Ctrl+C`

2) Save your `.env` file.

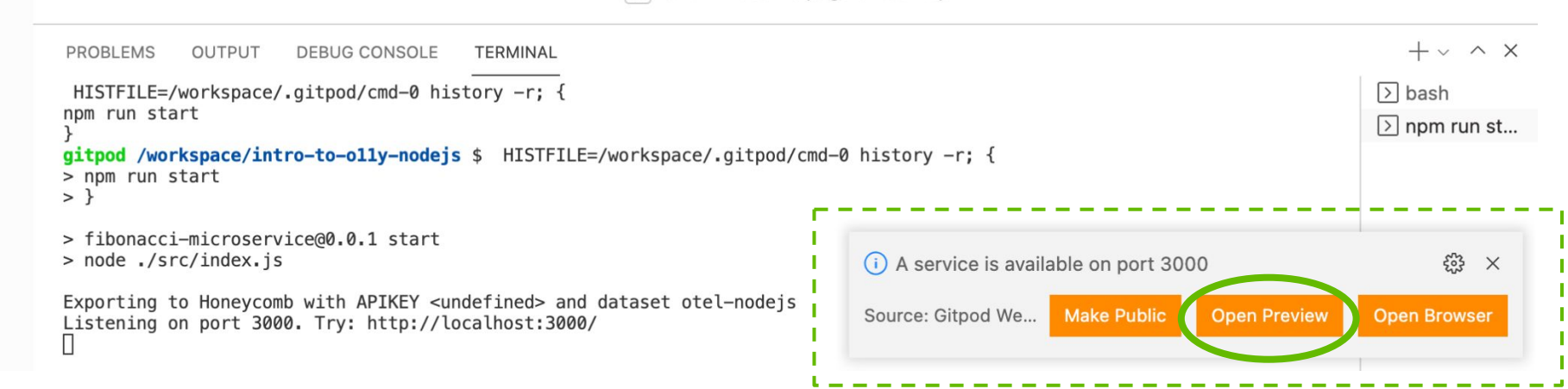
3) In the terminal, run the app with:

```
./run
```

Then, click “Go” and “Stop” in the app.

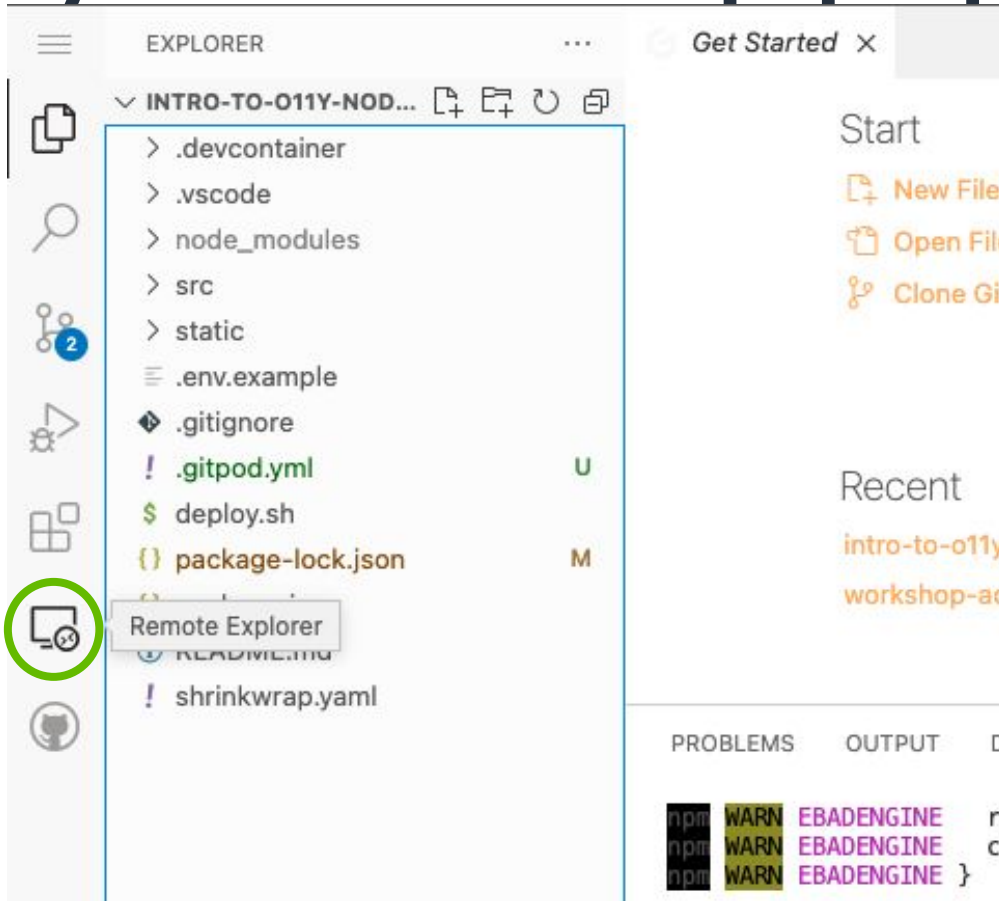
How to Open an App Preview

In the Terminal area, a "Service is available on port xxxx" pop-up may appear.
Select **Open Preview** to open the app display in a new tab.



Troubleshooting: If you don't see the pop-up

Go to **Remote Explorer**.

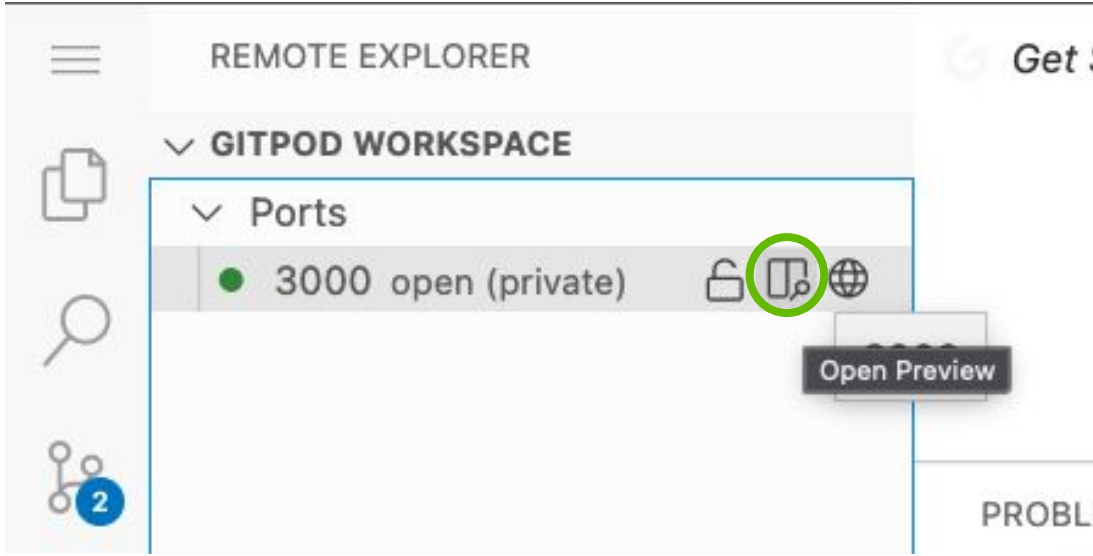


Troubleshooting: If you don't see the pop-up

Go to Remote Explorer.

Expand *Ports* to see the listed port number. For example, *3000*

Choose the middle ***Open Preview*** icon for the app to appear in a new tab.

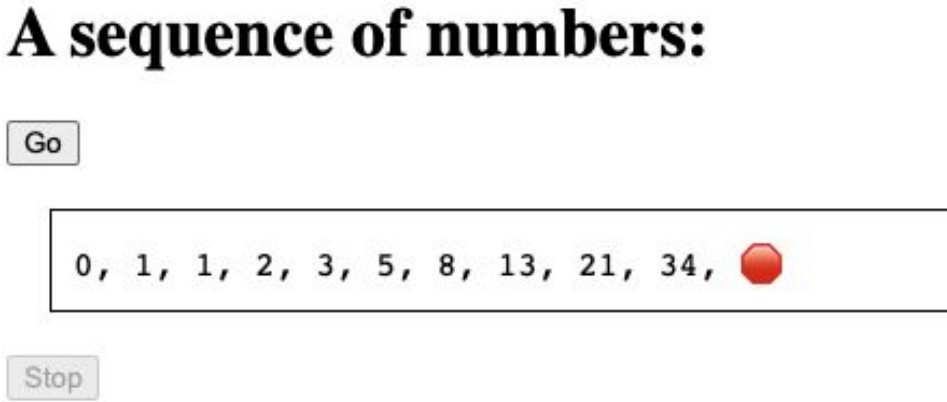


Use the Gitpod App again

Make the sequence of numbers appear.
Select **Go**!

… and then select **Stop**, after 5-7 numbers.

Repeat as needed to generate data.

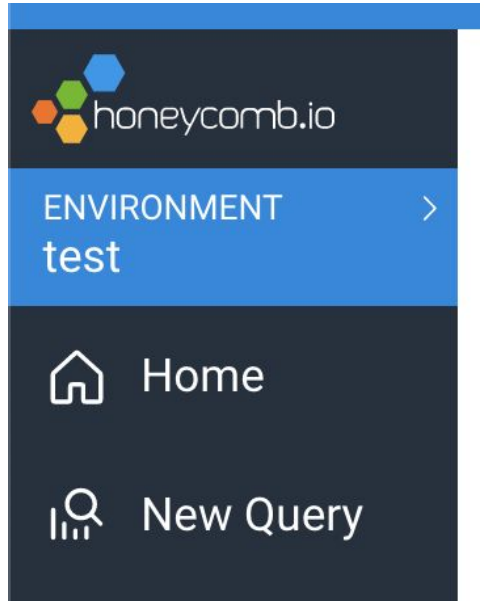


Data should start flowing!

The blue box turns green

✓ Data received! Events from your application are now available to explore in Honeycomb. Explore your data

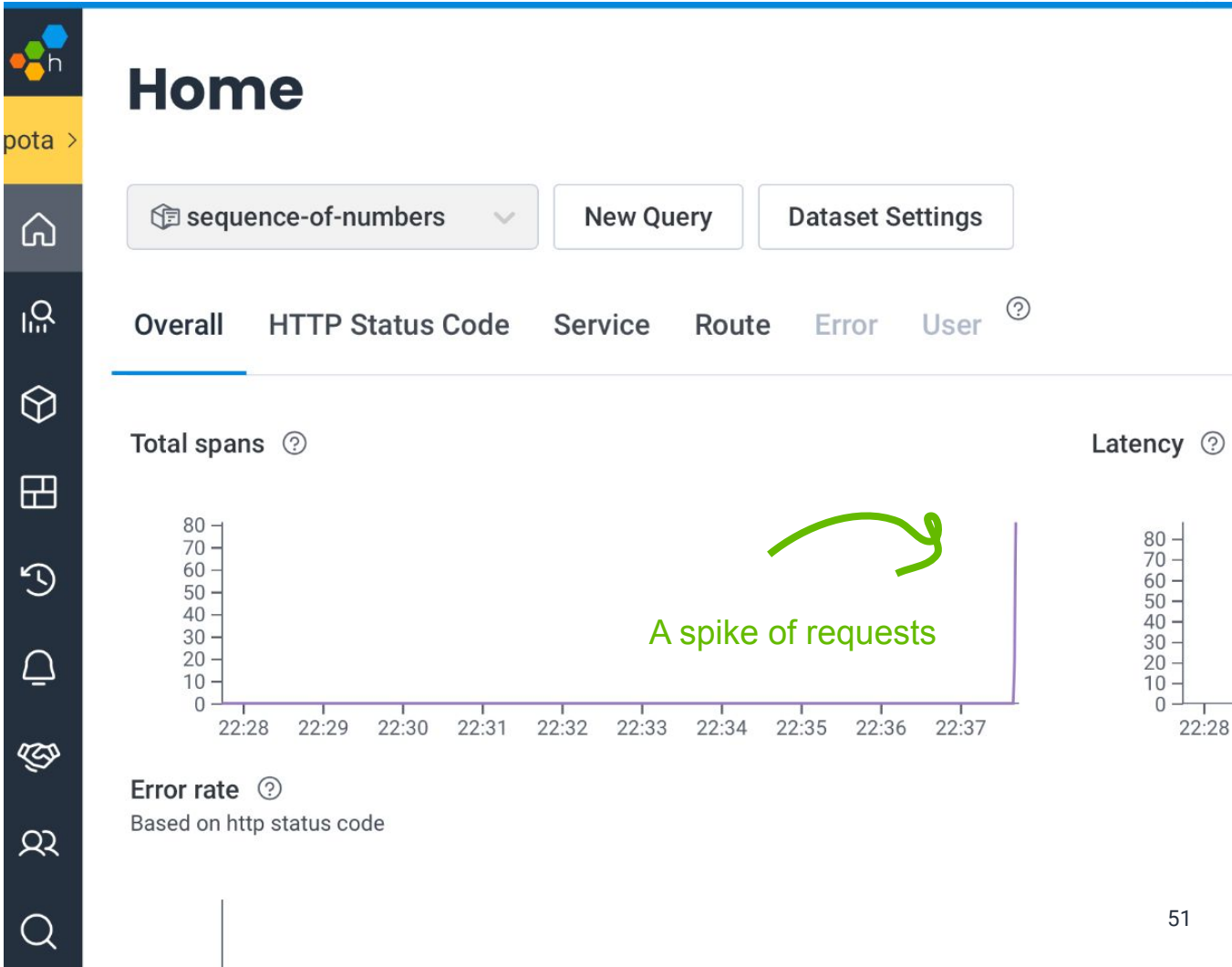
Click “Home”



Home

Got data?
Take a break.

Ask questions in break-out if you get stuck.



Summary

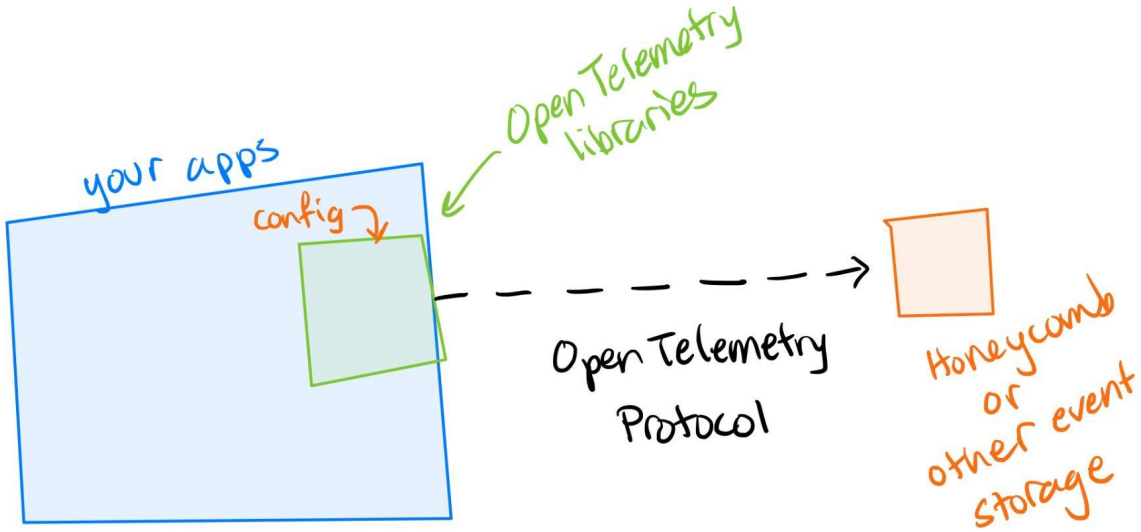
In this section, we:

- Ran a demo app that is instrumented with OpenTelemetry
- Got a Honeycomb Account
- Sent telemetry from the demo app to Honeycomb
- Confirmed data ingestion from the Honeycomb home screen

This is OpenTelemetry

OpenTelemetry is a vendor-neutral standard for instrumentation.

It has an instrumentation API, various SDKs, proxies, and wire formats pre-standardized.



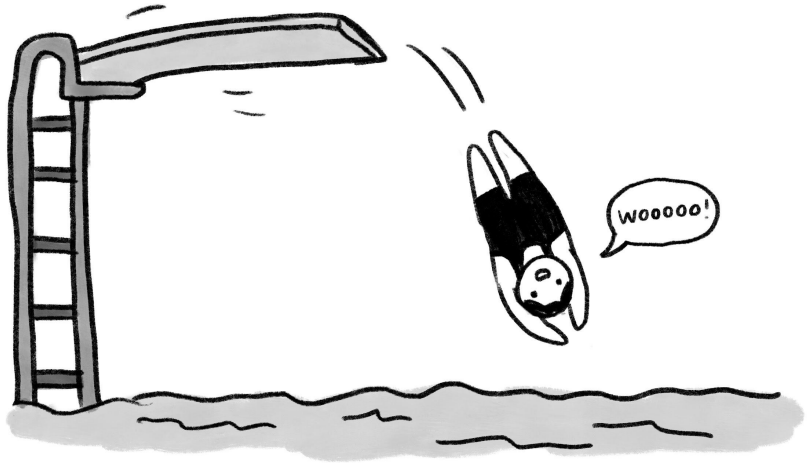
Using data to answer questions

Learn to use Honeycomb's product UI!

Section Overview

In this section, we will:

- Learn about Honeycomb UI and tools
- Find out what makes some calls to our Fibonacci app slow

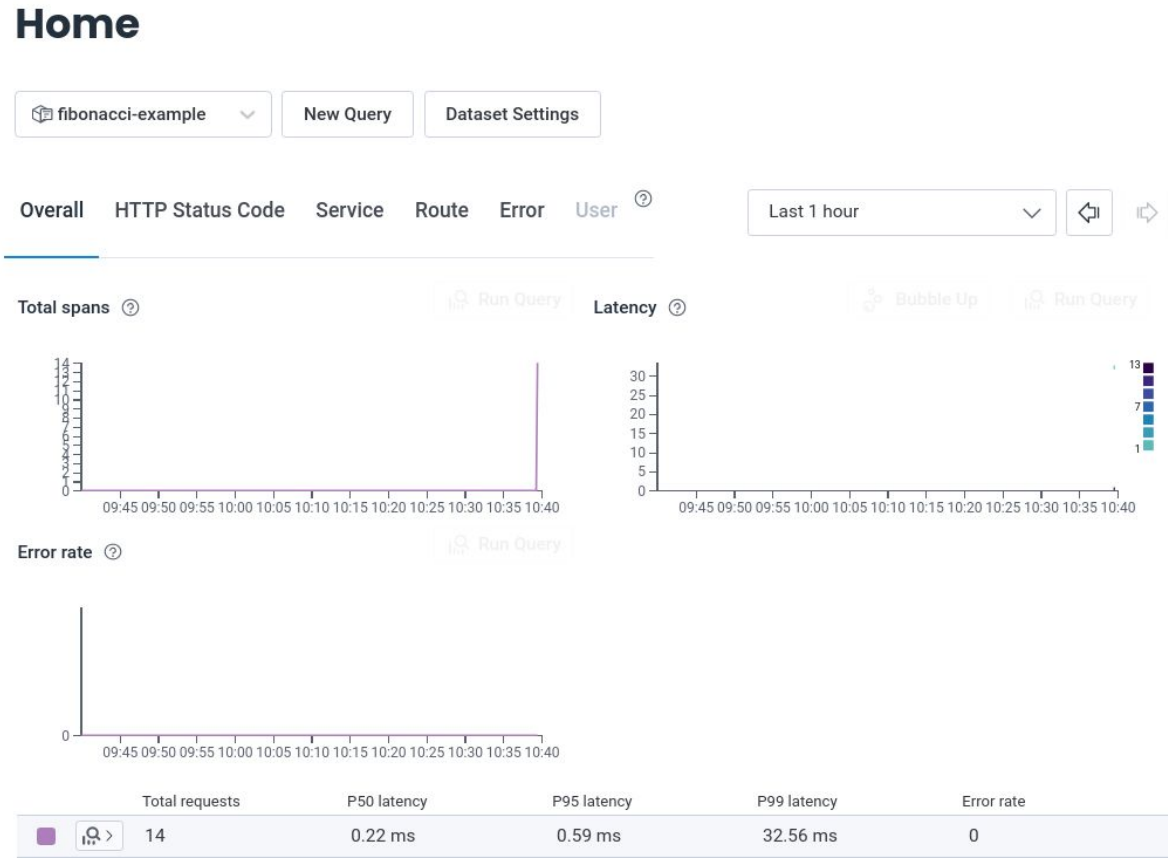


Home

The basics:

Rate, error, duration graphs

You can always come to Home via the Honeycomb logo at top left.

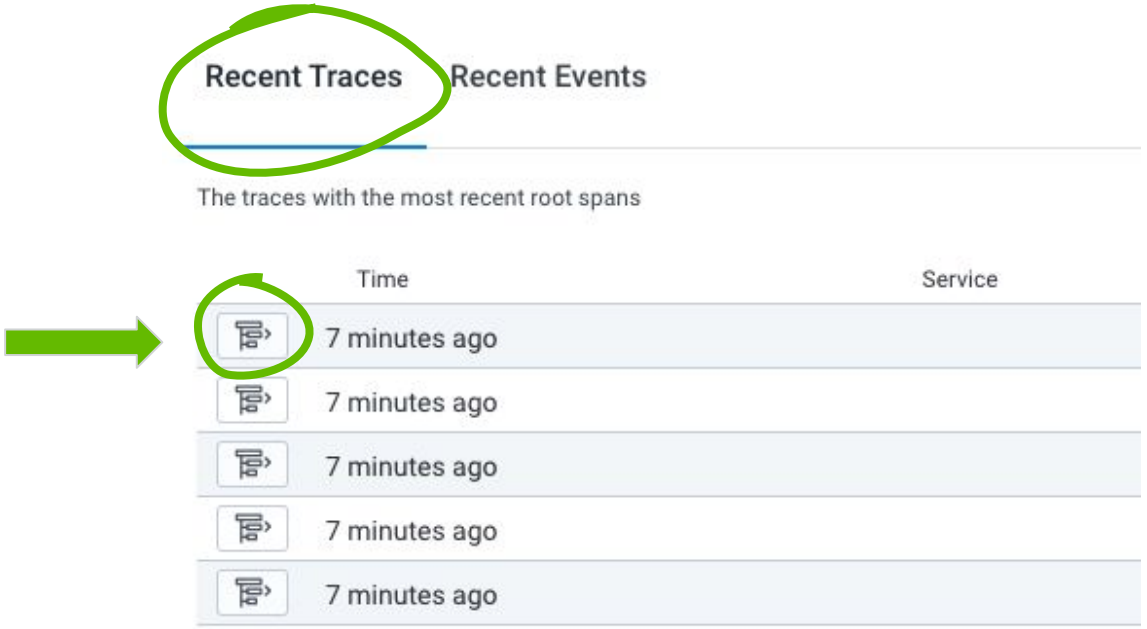


Let's find & examine a trace!

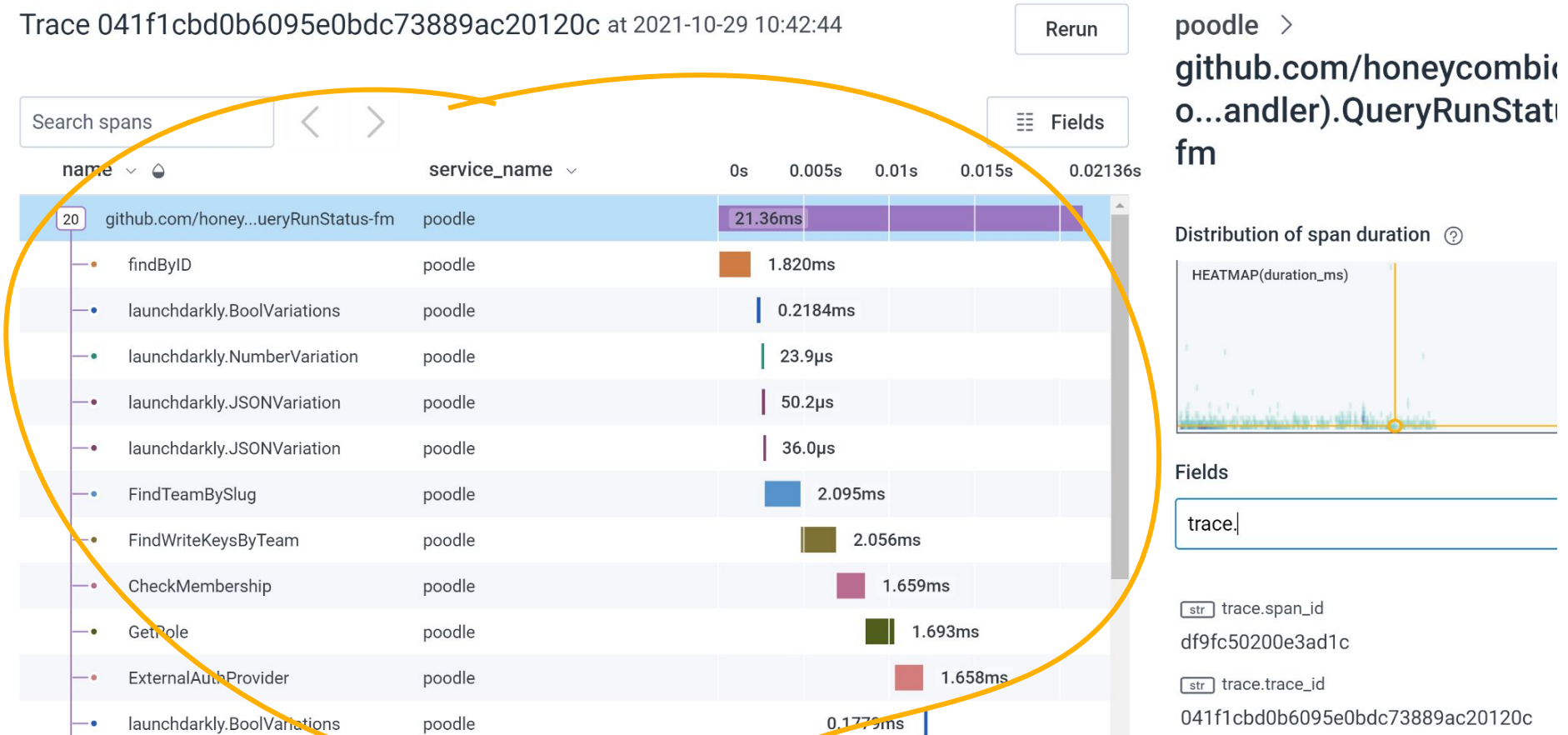
After using the app...

Look for the trace in Recent Traces tab at the bottom of the Home screen.

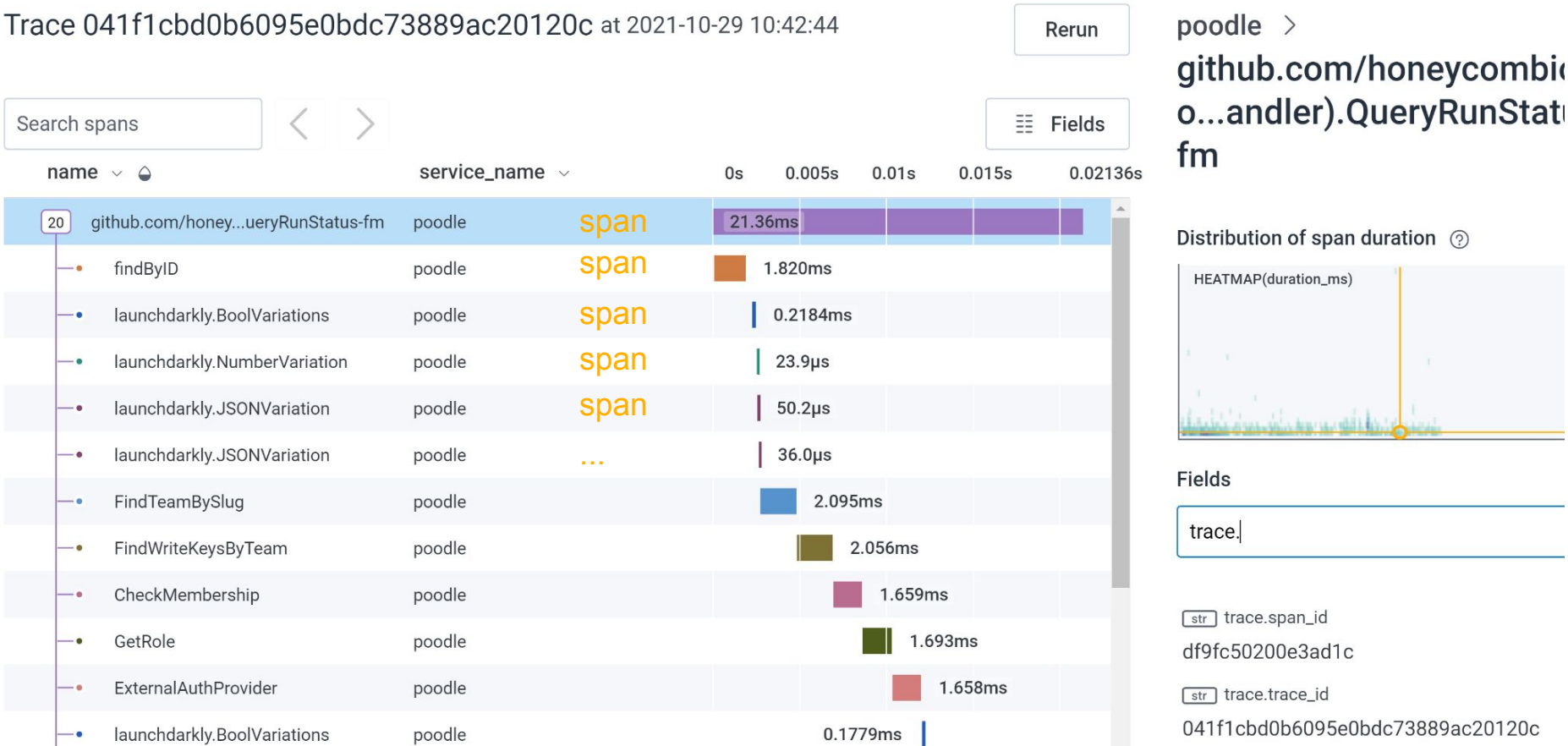
Use the **View Trace** button on the row you want to examine further.



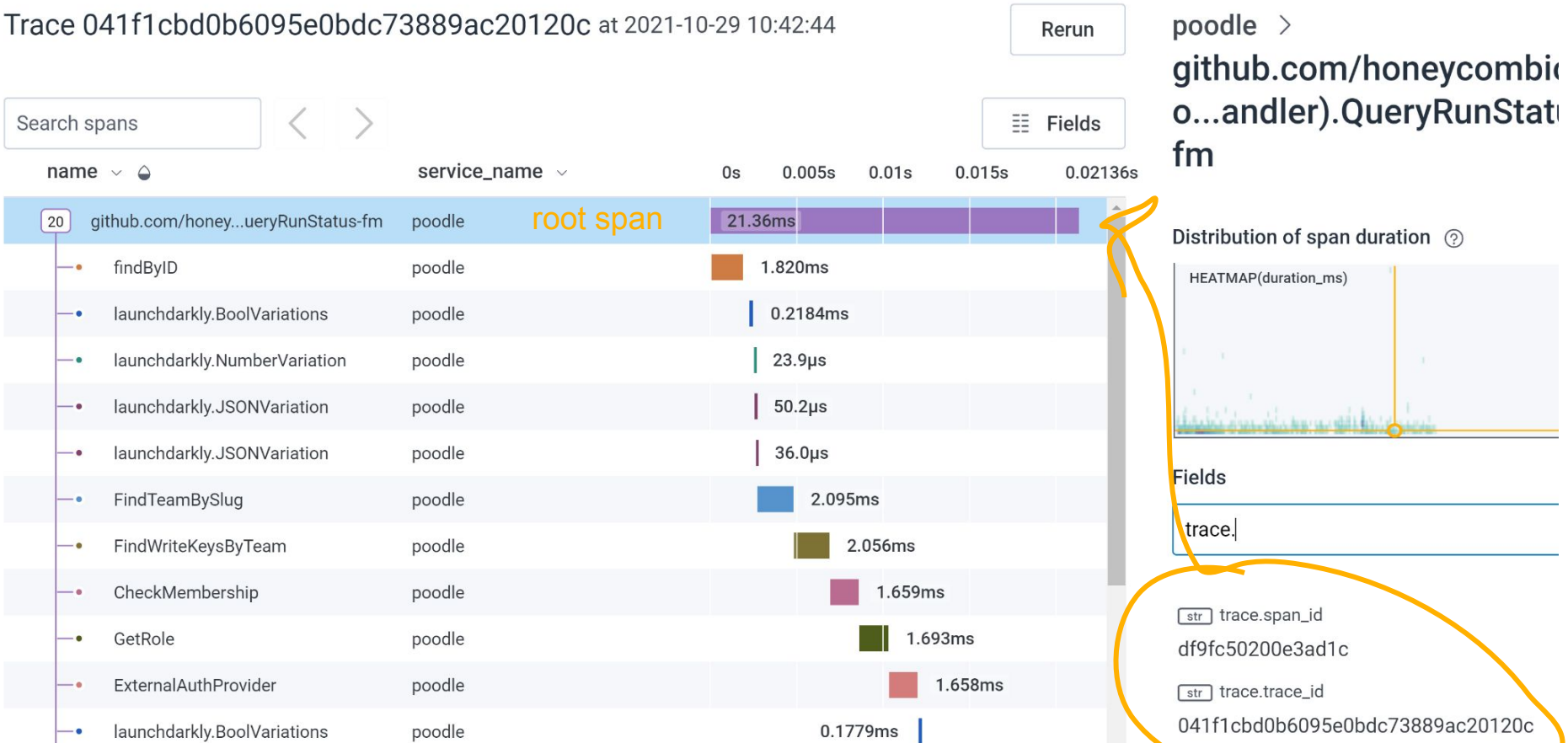
Parts of a trace



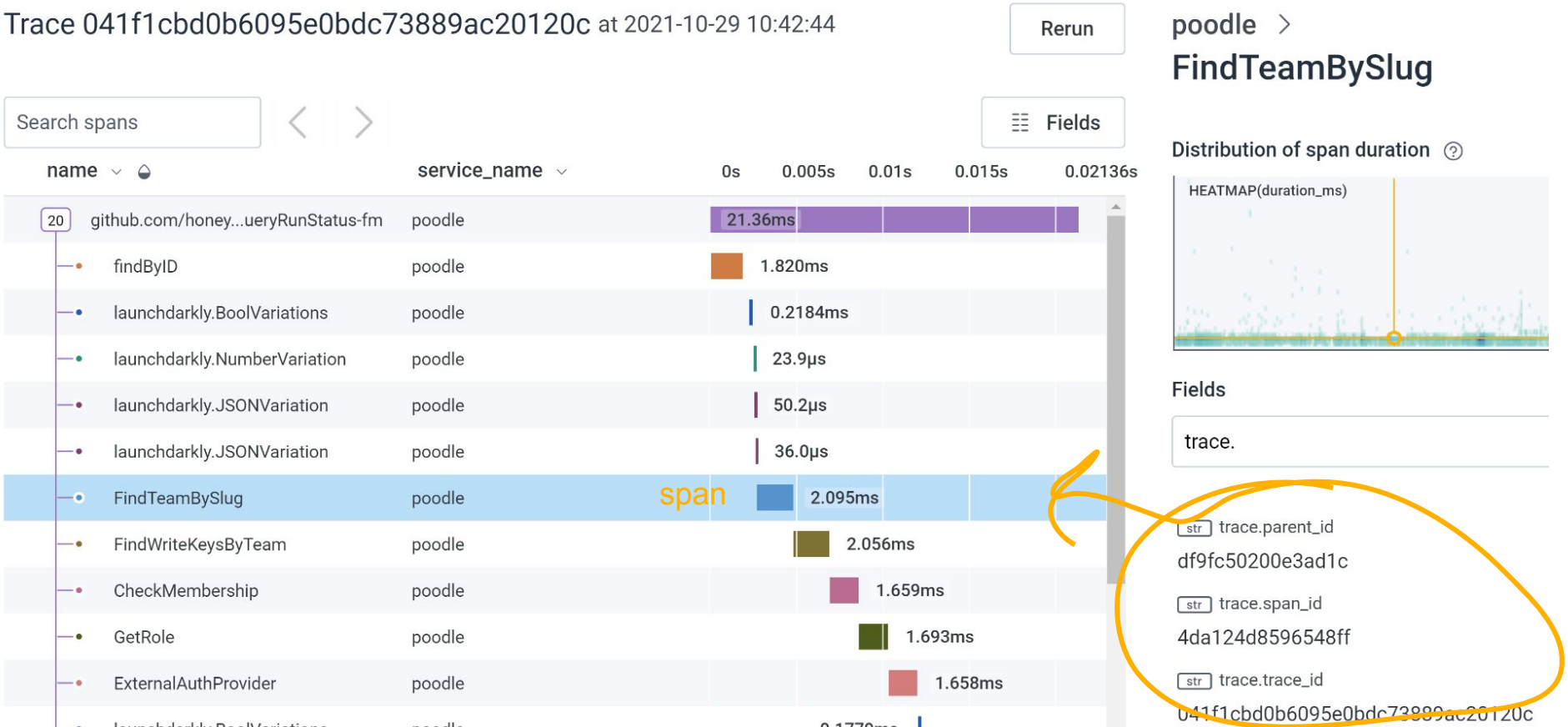
Parts of a trace



Parts of a trace

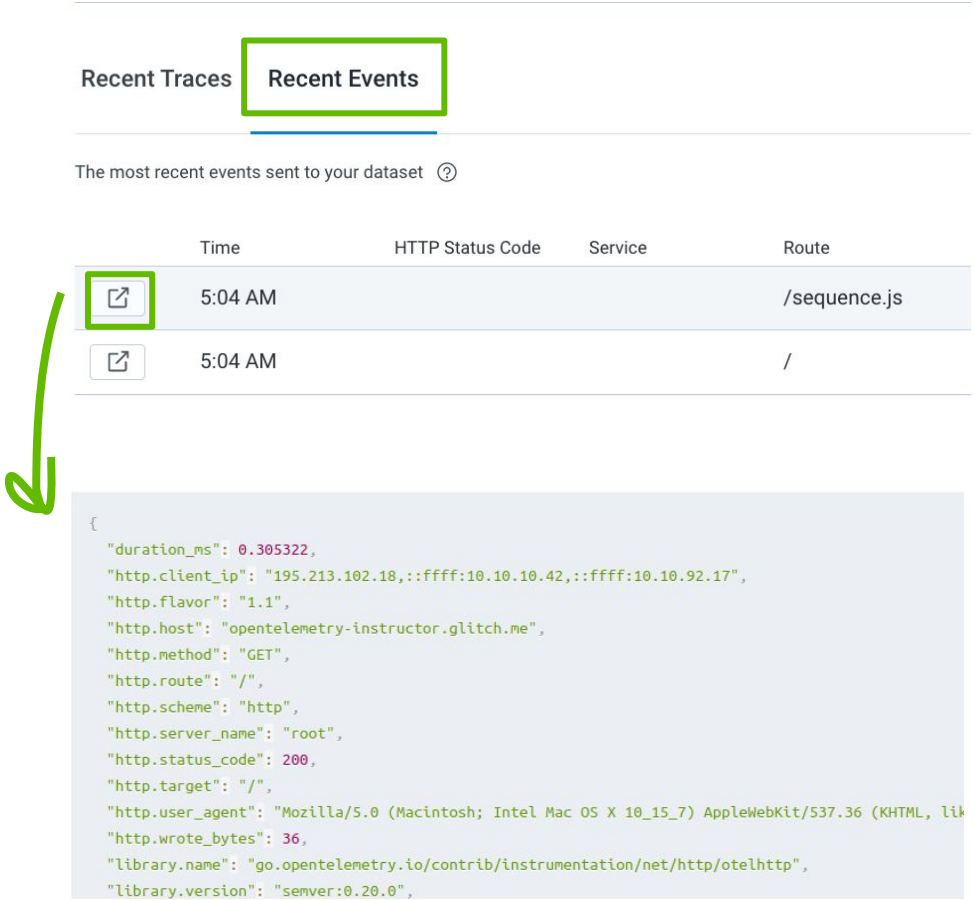


Parts of a trace



Raw data view

Also, the **Recent Events** tab at bottom of **Home** view shows raw JSON for events.

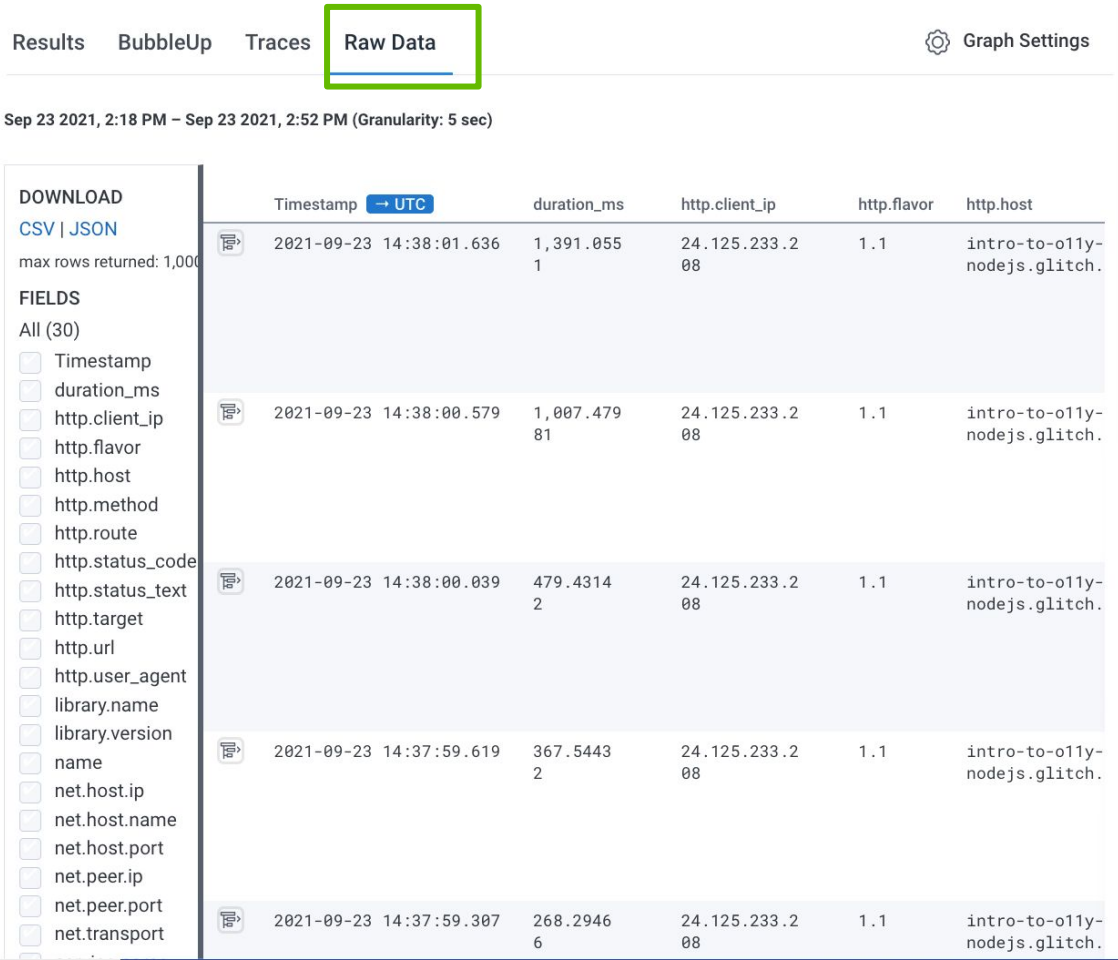


Raw data view

Sometimes you need to look at spans as wide events/structured logs!

Honeycomb supports showing a table of spans and fields on those spans.

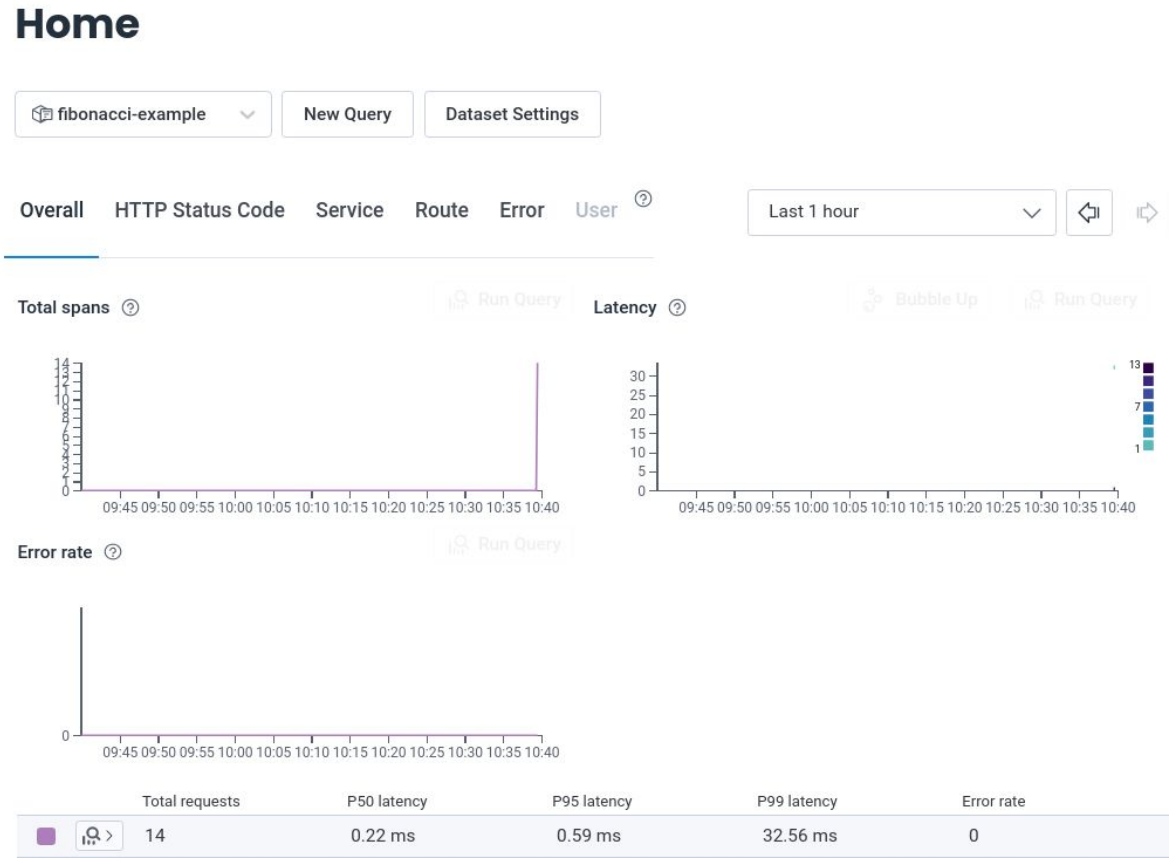
Go to the **Raw Data** tab.



Hello data

Click the house to get back to Home

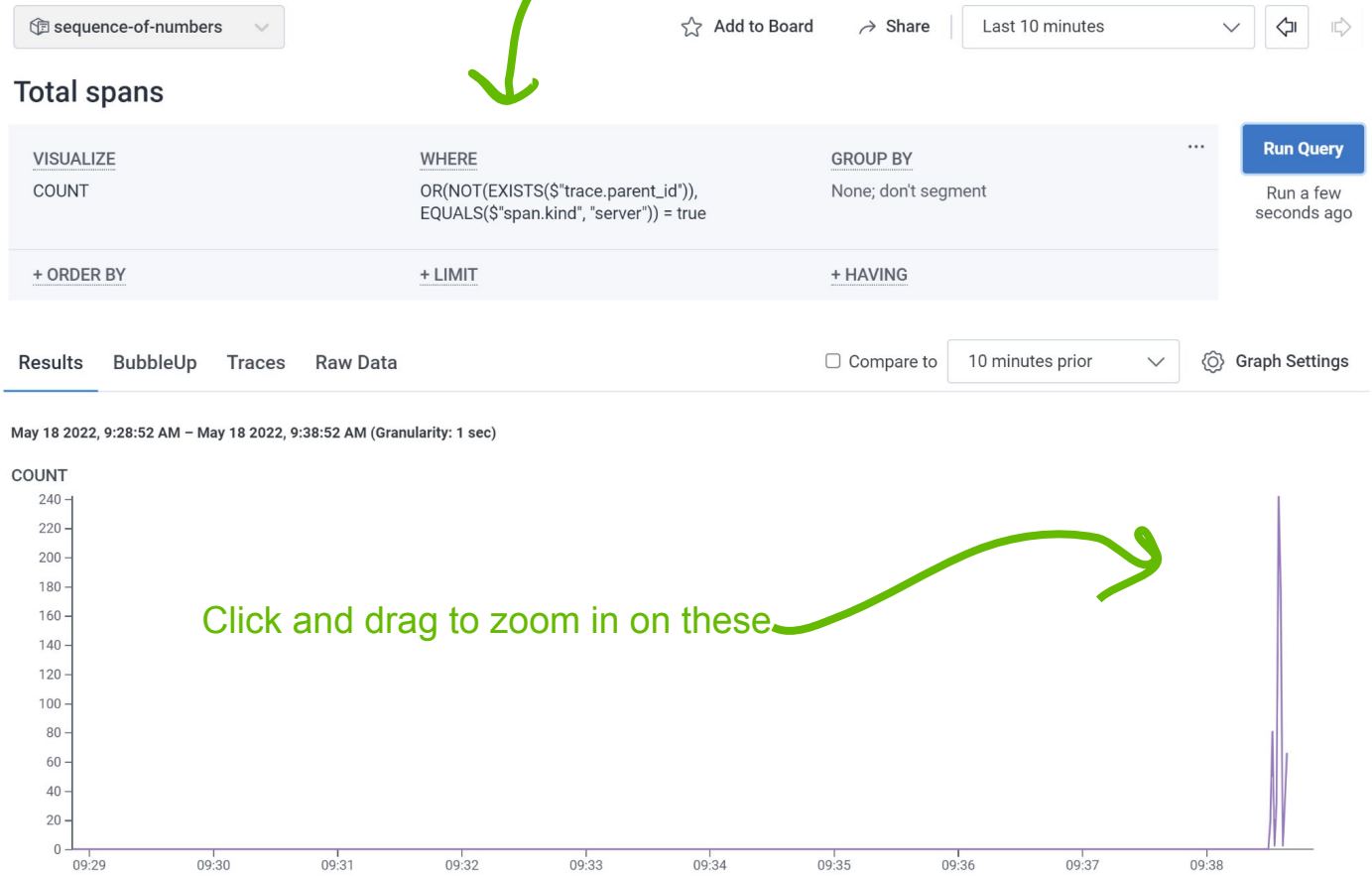
Click the first graph to zoom into it.



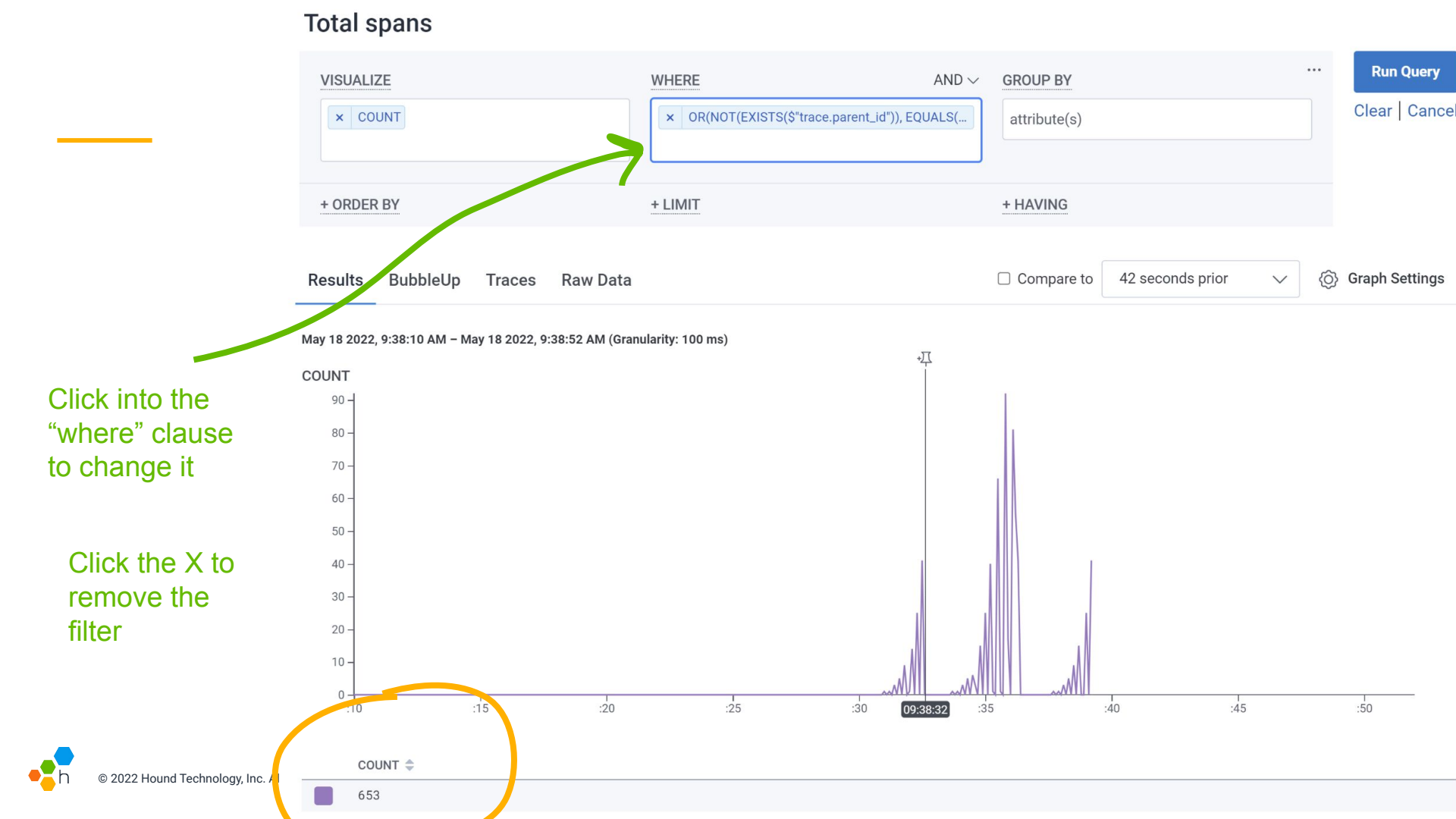
Interactive Querying

A guess at “top-level spans”

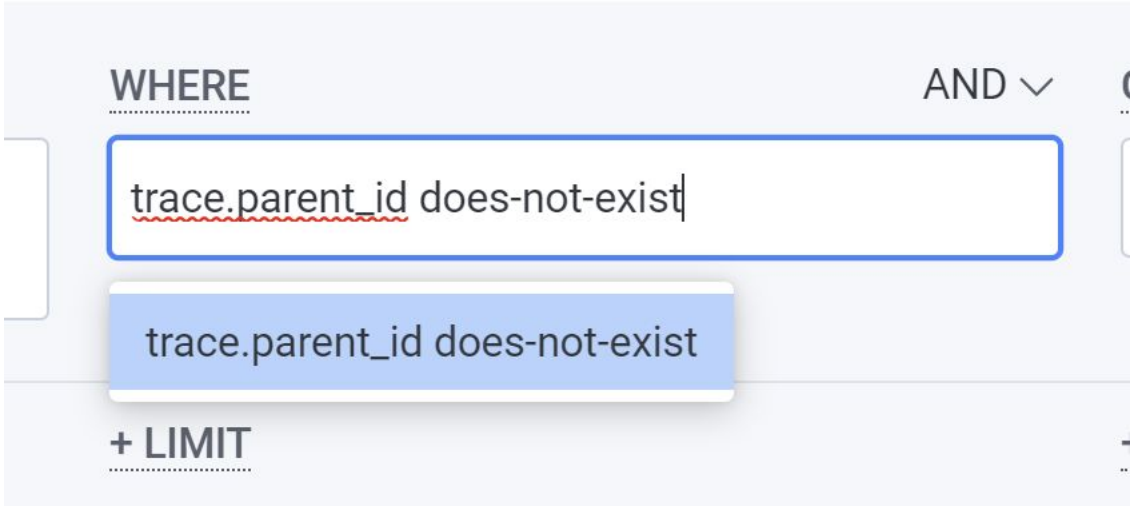
Your service



Click and drag to zoom in on these



Find only the root spans



VISUALIZE	WHERE	GROUP BY
COUNT	trace.parent_id does-not-exist	None; don't segment
+ ORDER BY	+ LIMIT	+ HAVING

Run Query

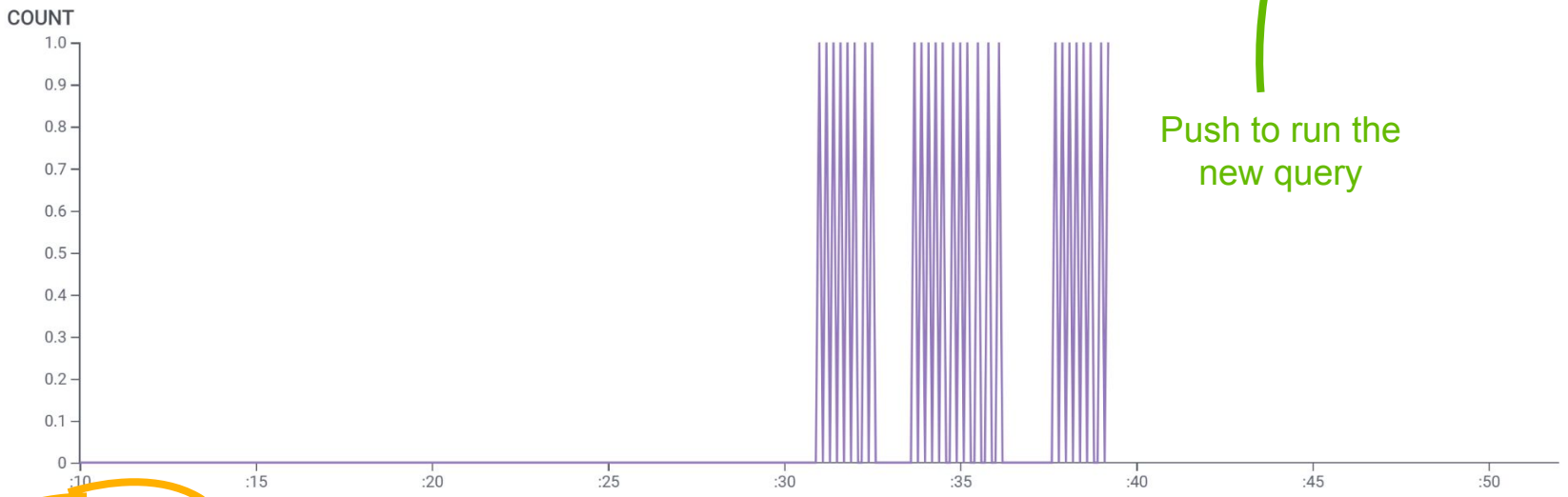
Run a few seconds ago

Results | BubbleUp | Traces | Raw Data

Compare to 42 seconds prior

Graph Settings

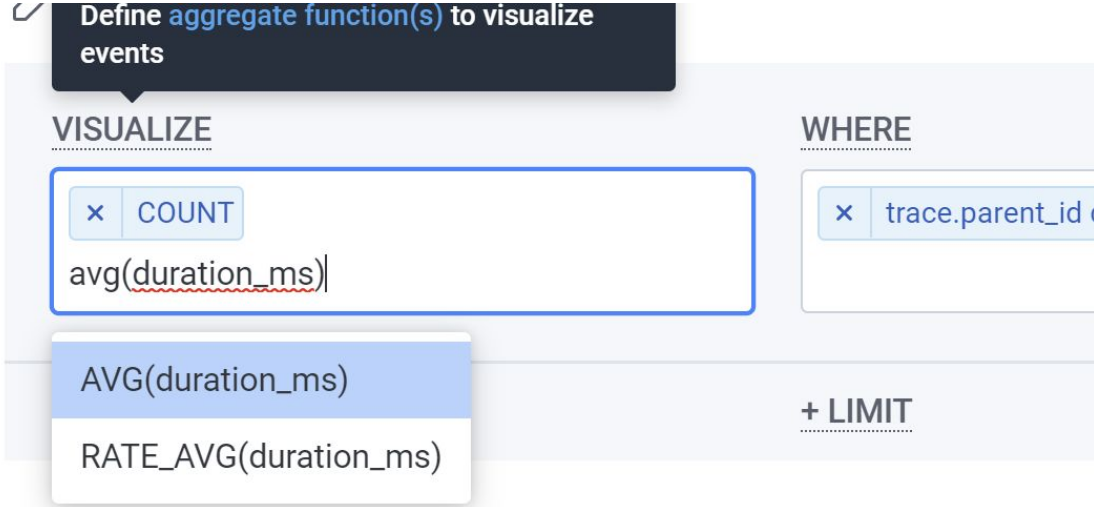
May 18 2022, 9:38:10 AM – May 18 2022, 9:38:52 AM (Granularity: 100 ms)



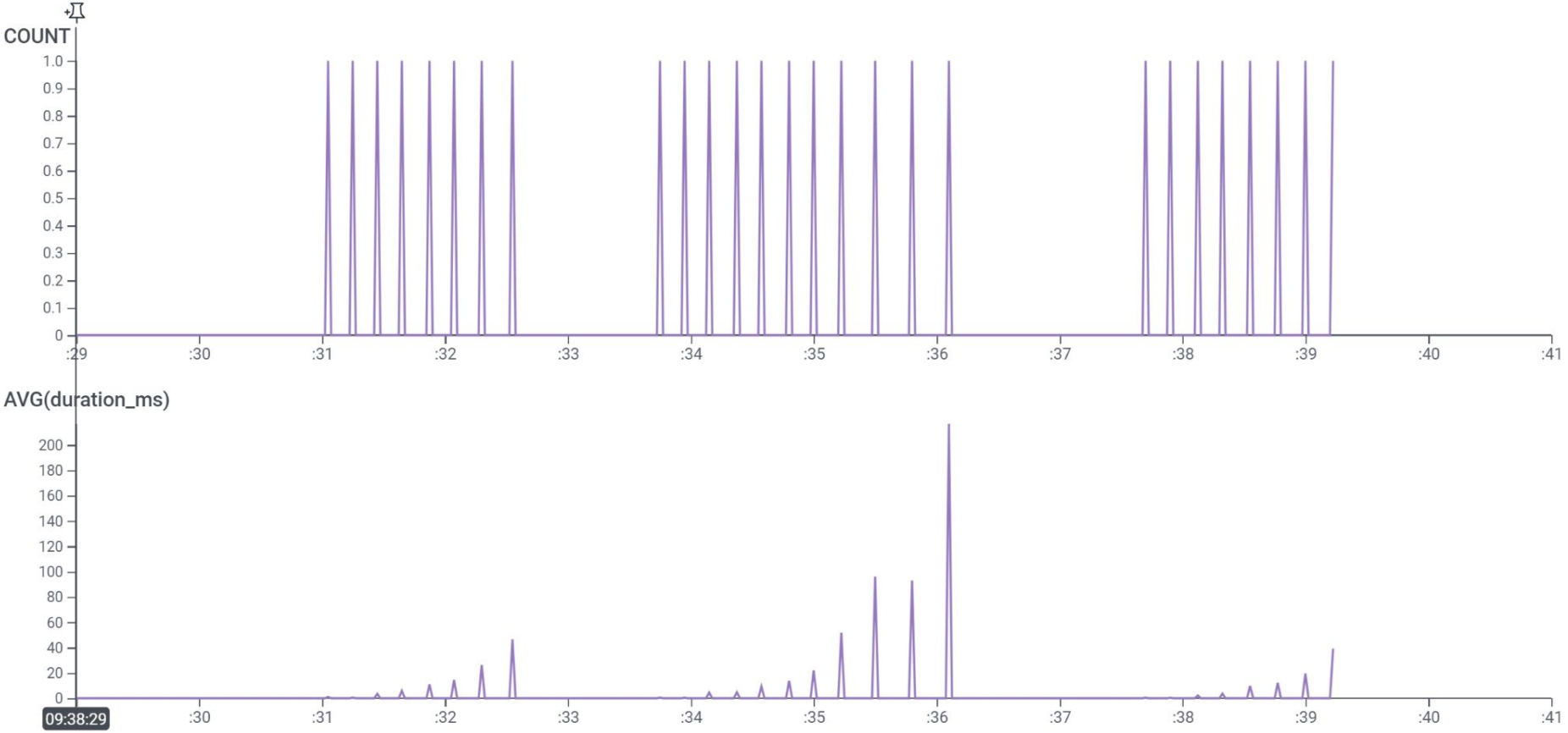
Push to run the new query

COUNT
27

Add more visualizations!



May 18 2022, 9:38:29 AM – May 18 2022, 9:38:41 AM (Granularity: 25 ms)



Now add a heatmap

Under VISUALIZE, add

HEATMAP(duration_ms)

then push Run Query.

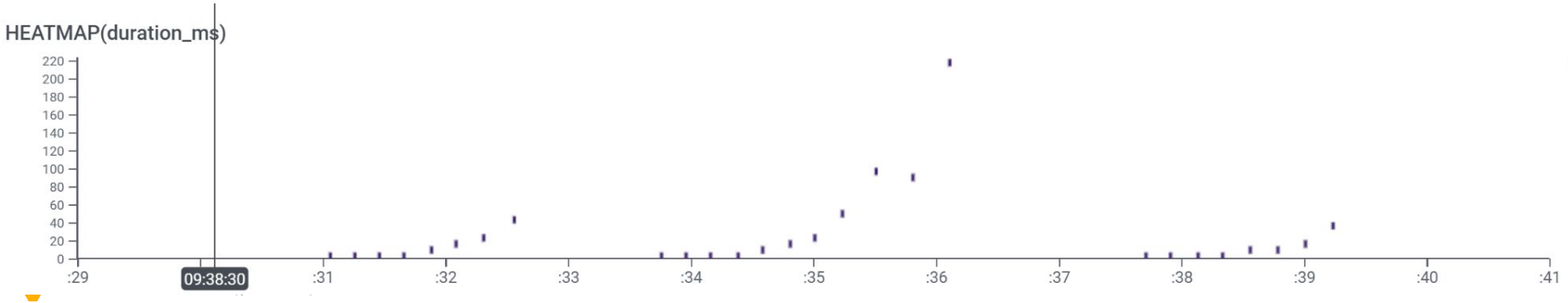
It doesn't look like much…

VISUALIZE

COUNT

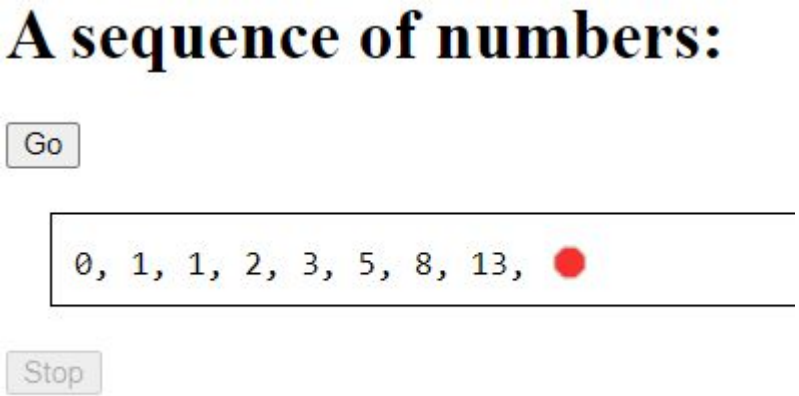
AVG(duration_ms)

HEATMAP(duration_ms)



Generating a little more data...

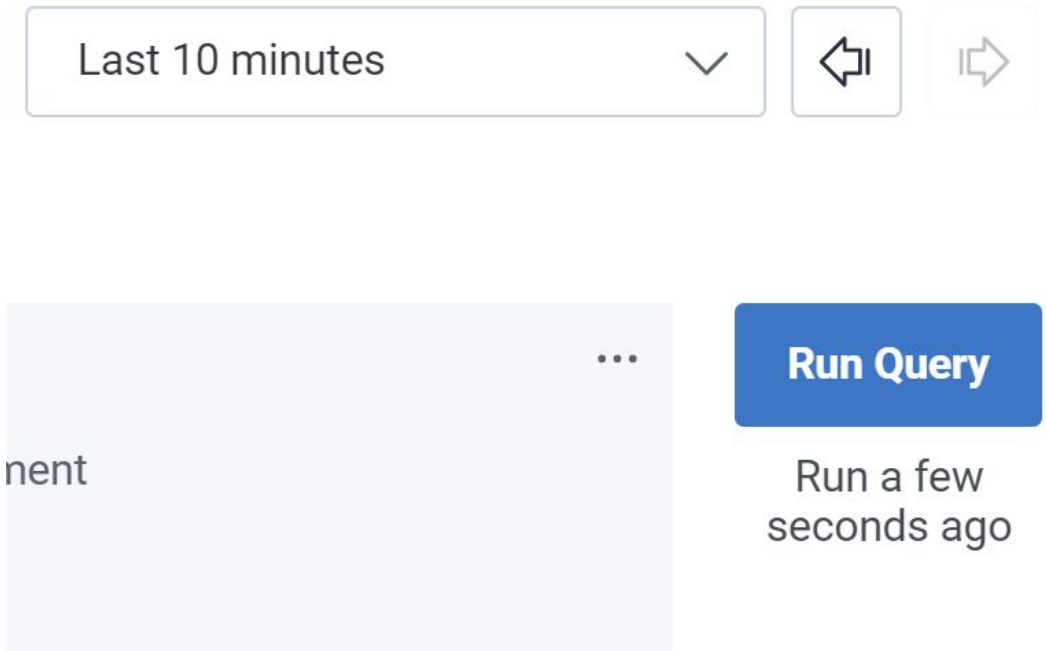
In your app, select **Go**! … and then make it **Stop** after 5-7 numbers several times.



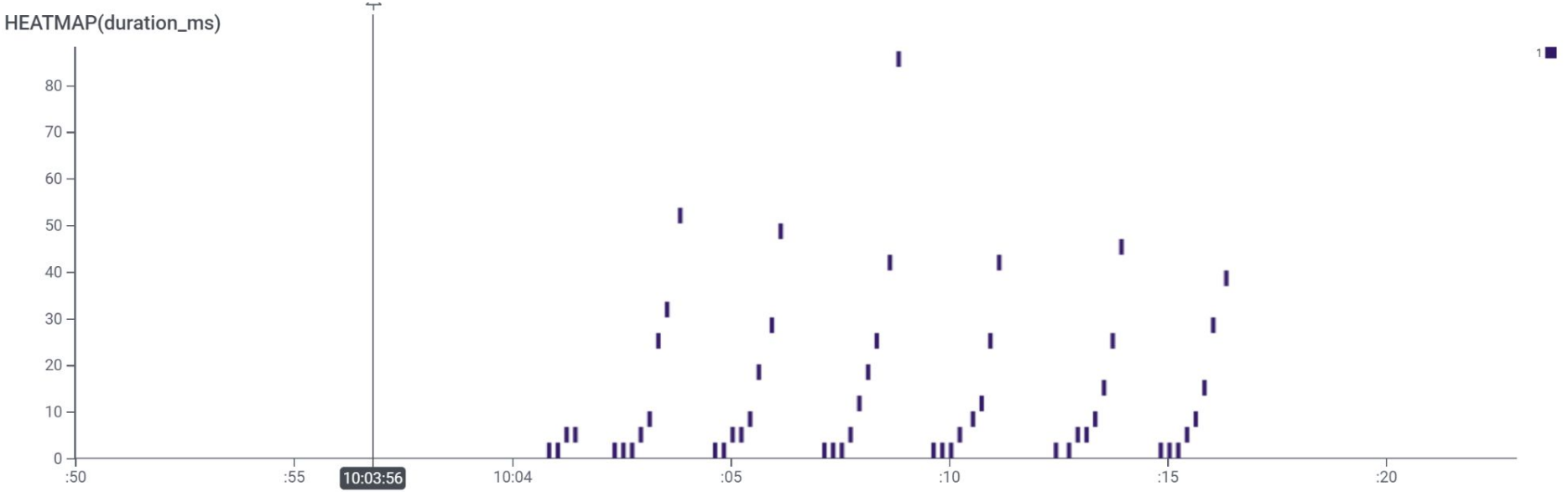
Run the query again

Change the timeframe back to “Last 10 minutes”

Then zoom in again on the part of the timeline with data.



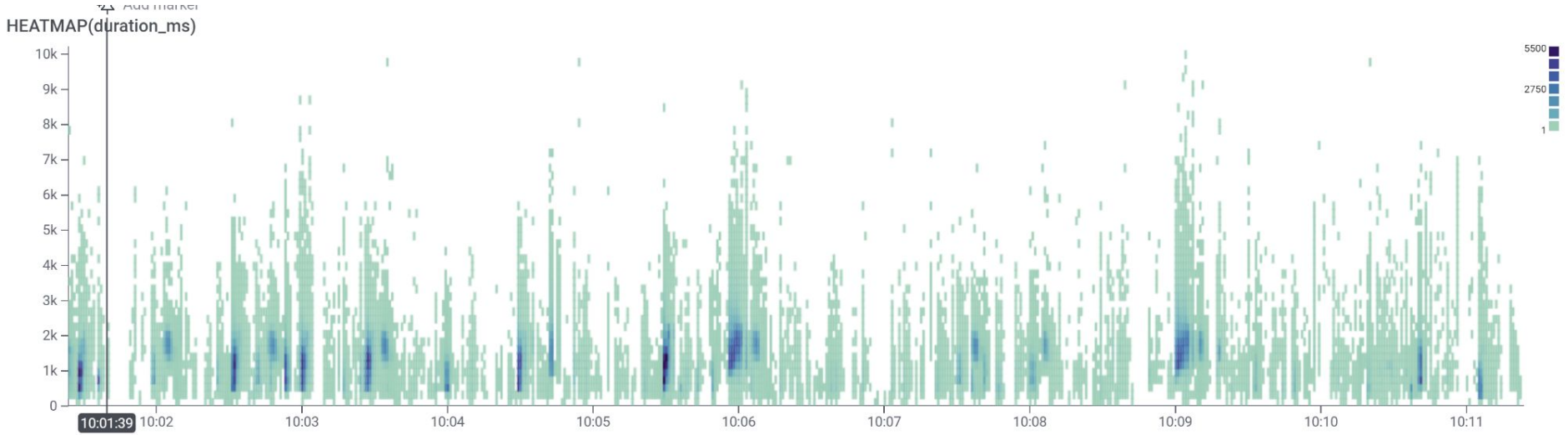
The higher the dot, the slower the request.



Heatmaps

Each area on the heatmap corresponds to a time of day, and a slowness of requests. (higher is slower)

The histogram's color shows the density in that area: how many requests took this long, at this time. (darker is more of them)



Accessing BubbleUp

Go to the **BubbleUp** tab and draw a box around a group of slow requests!

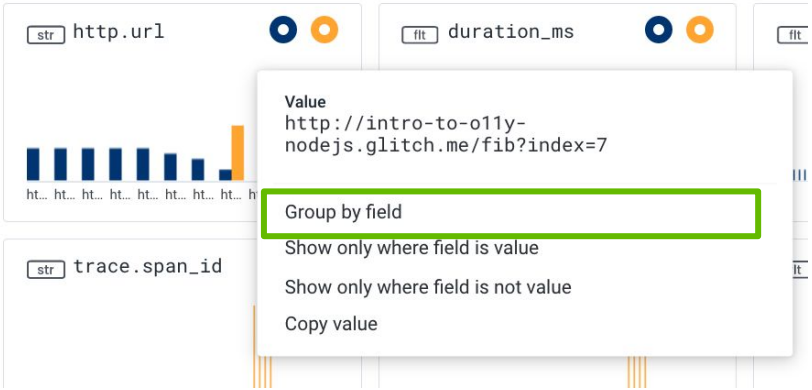
Honeycomb finds fields whose values are very different between **your selected events** and all the rest

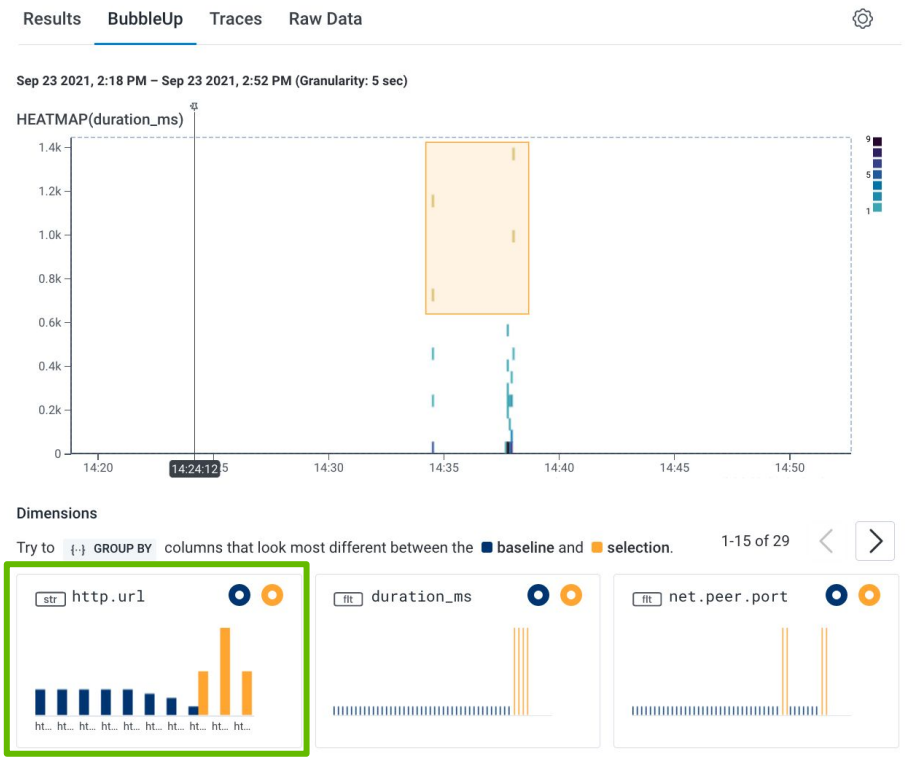
The yellow bars reflect fields and values disproportionately represented inside the area you drew.

The blue bars reflect the rest of the dataset.

Drilling down with BubbleUp

Click on the field that makes a difference (**http.url** or **http.target**).
Select **Group by field** in the menu.



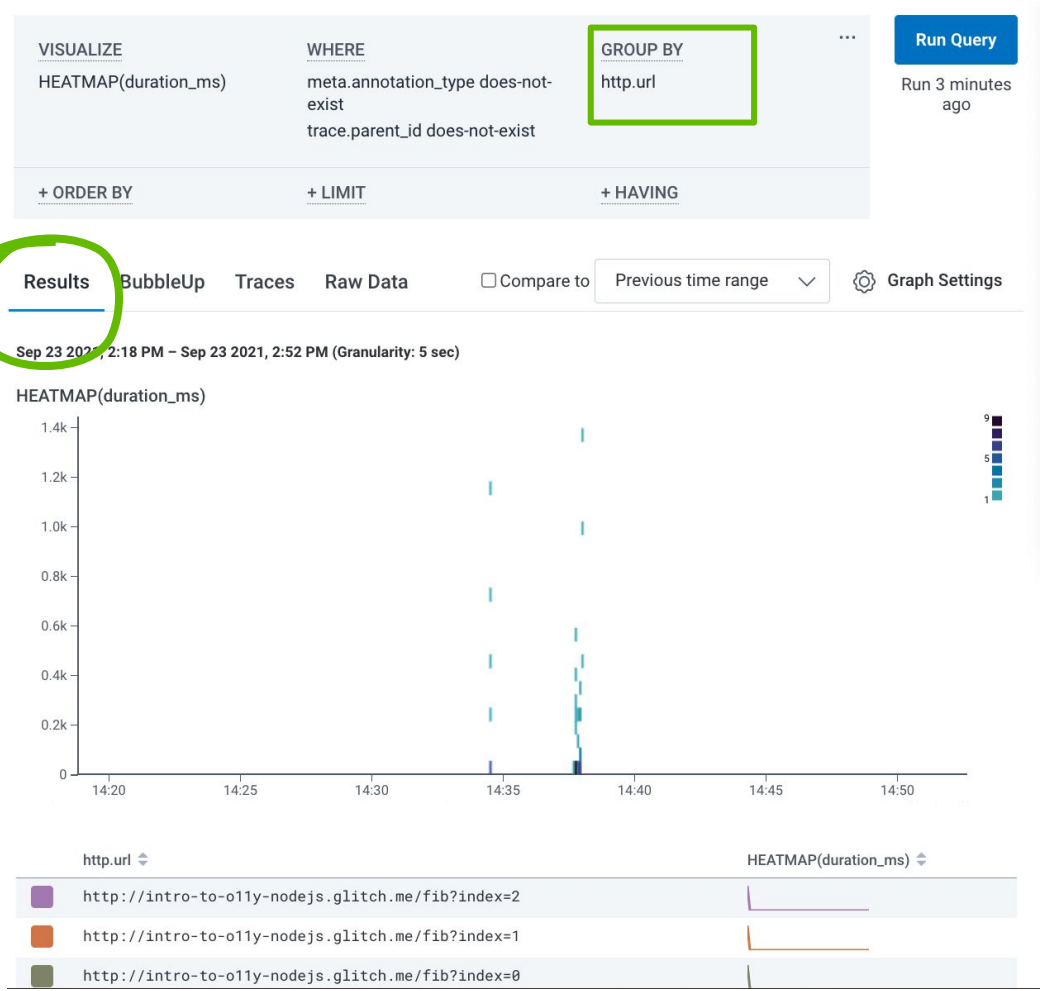


What did that do?

It doesn't look much different.

But the query has been customized…

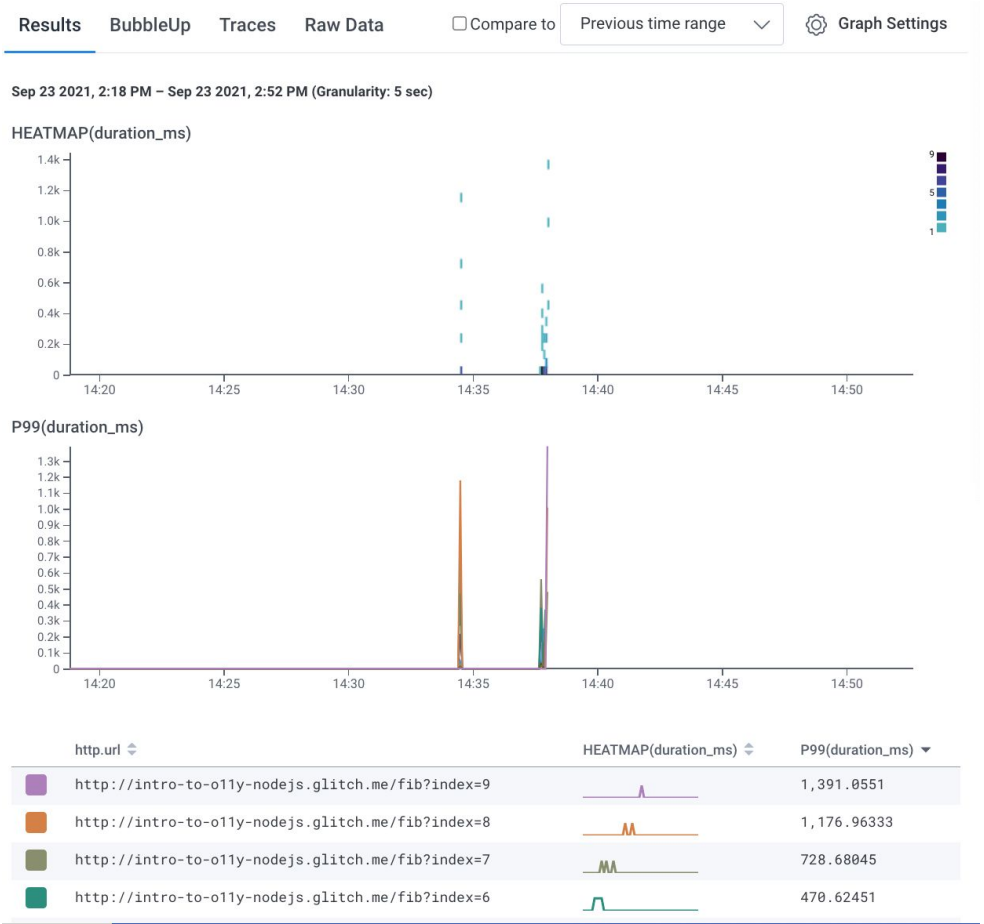
And if you select the **Results** tab, you will see the table below the graph.



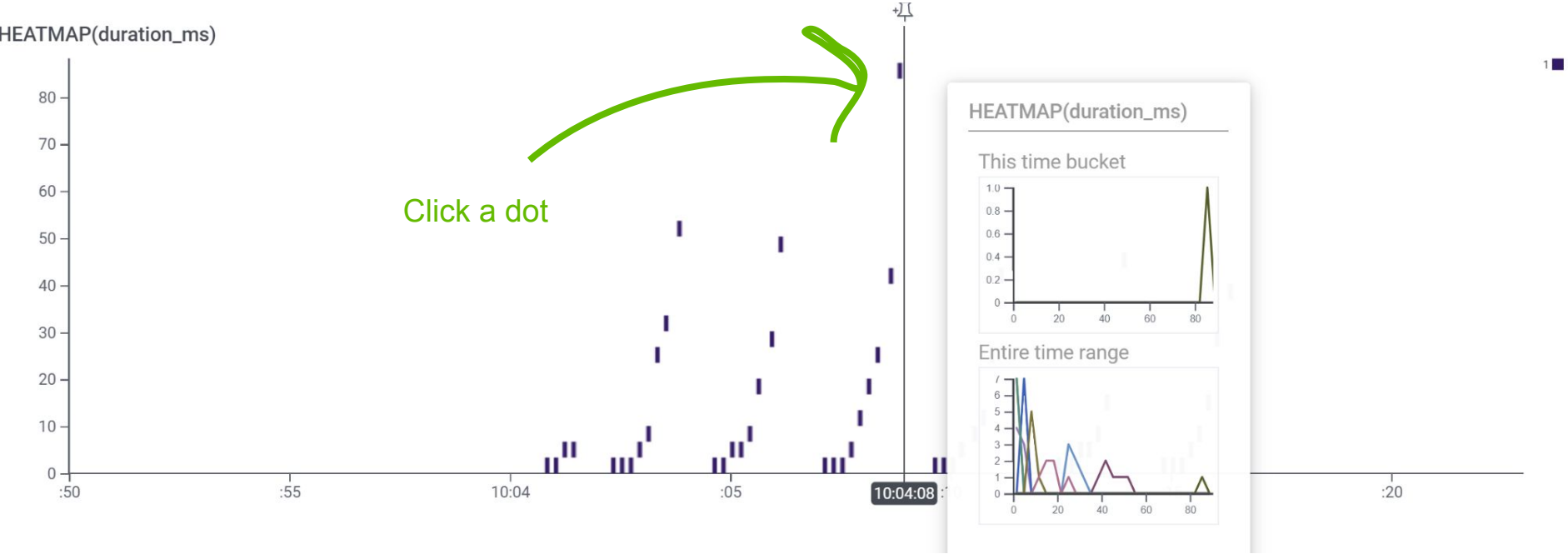
GROUP BY and HEATMAP

When in the **Results** tab...

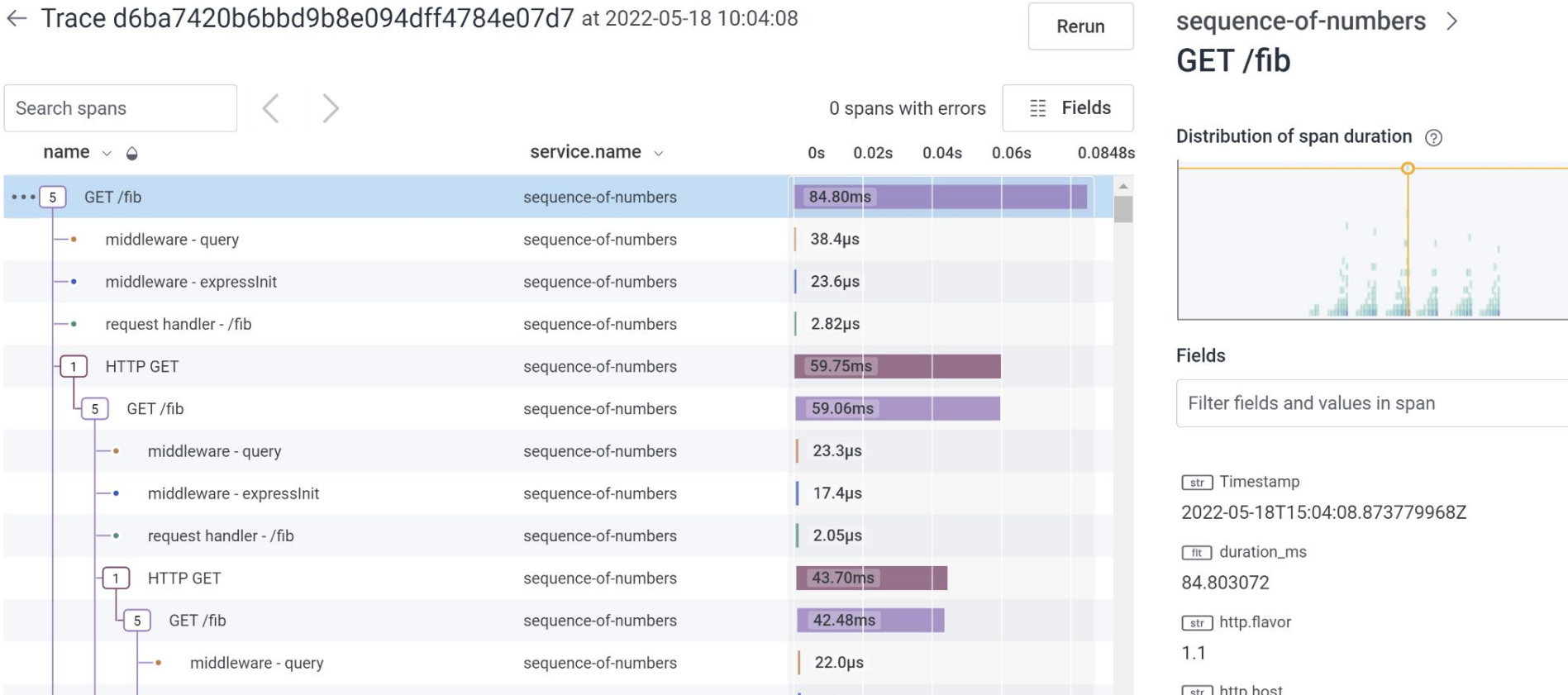
You can hover over a group of results to highlight just those entries in the heatmap.



Click on the heatmap to see a trace



See an example!

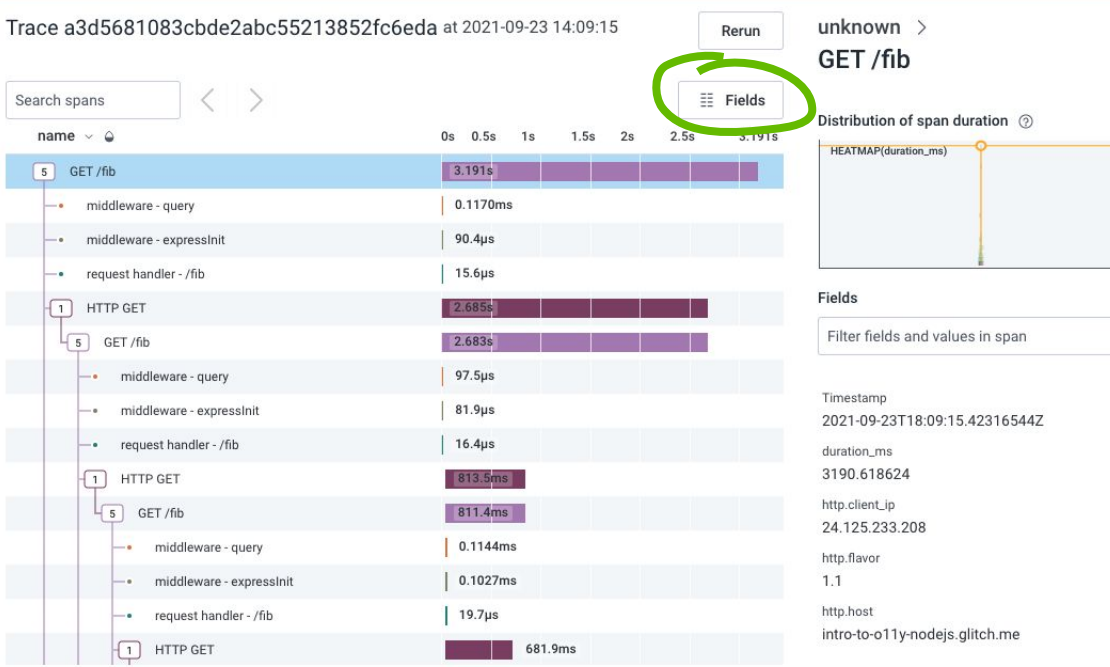


Examine a trace

To customize:

Select the **Fields** button in the upper right corner and enter **http.target**.

NOTE: For Python and Go implementations, use **http.target**
For others, use **http.url**

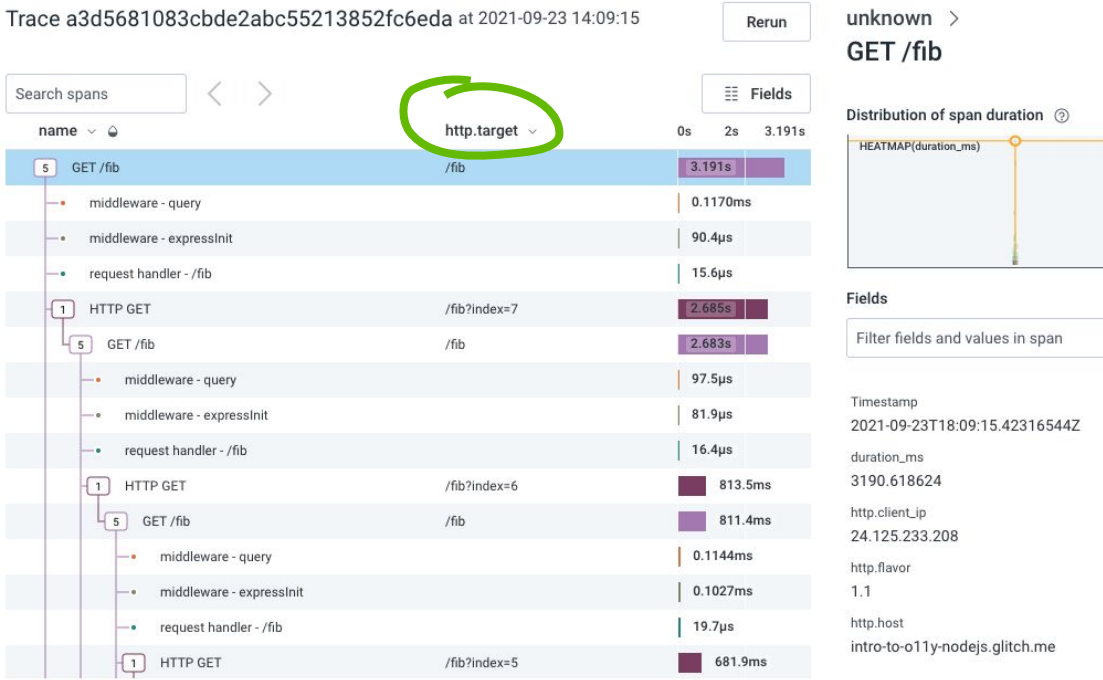


Examine a trace

To customize:

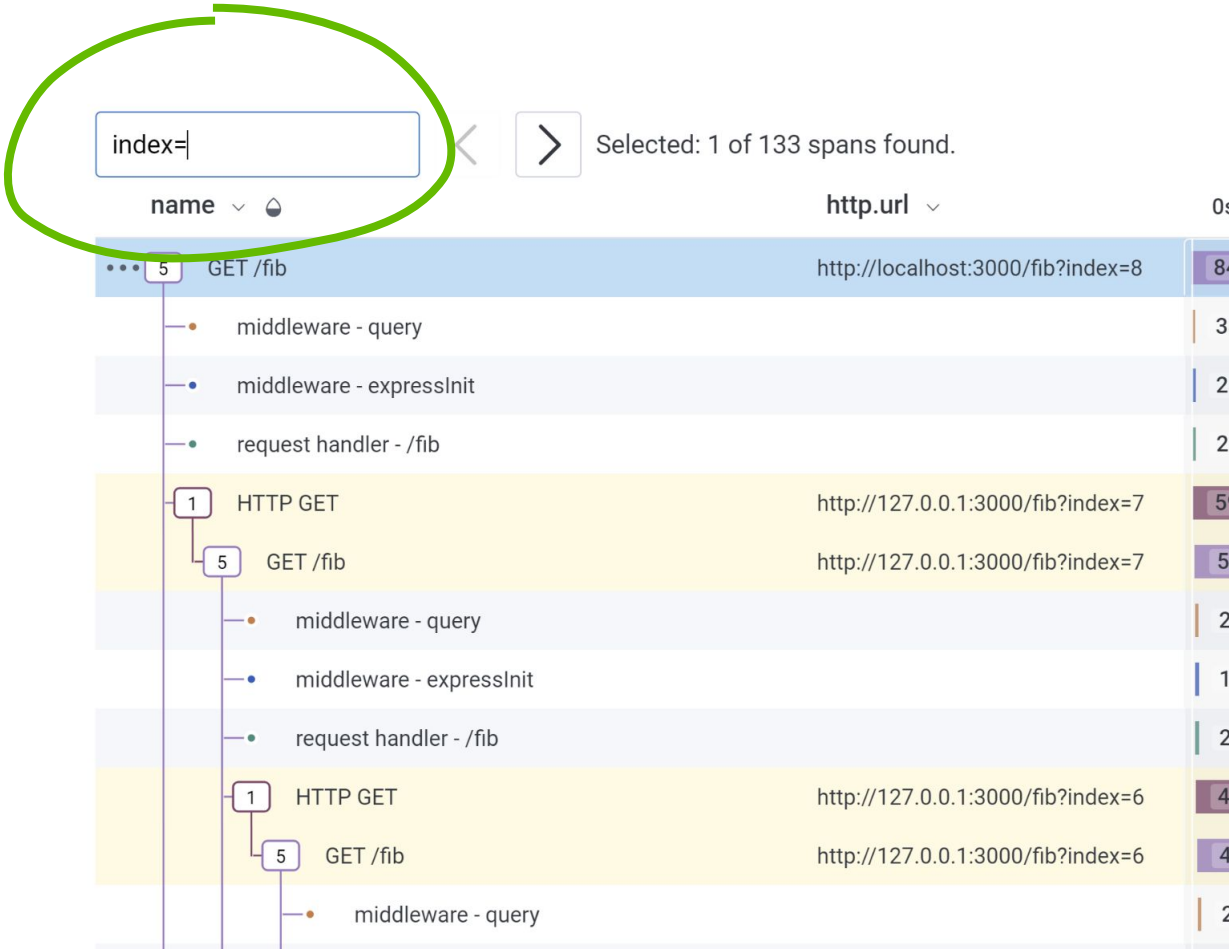
Select the **Fields** button in the upper right corner and enter **http.target** (for Python and Go)

The display will update to show a **http.target** column.



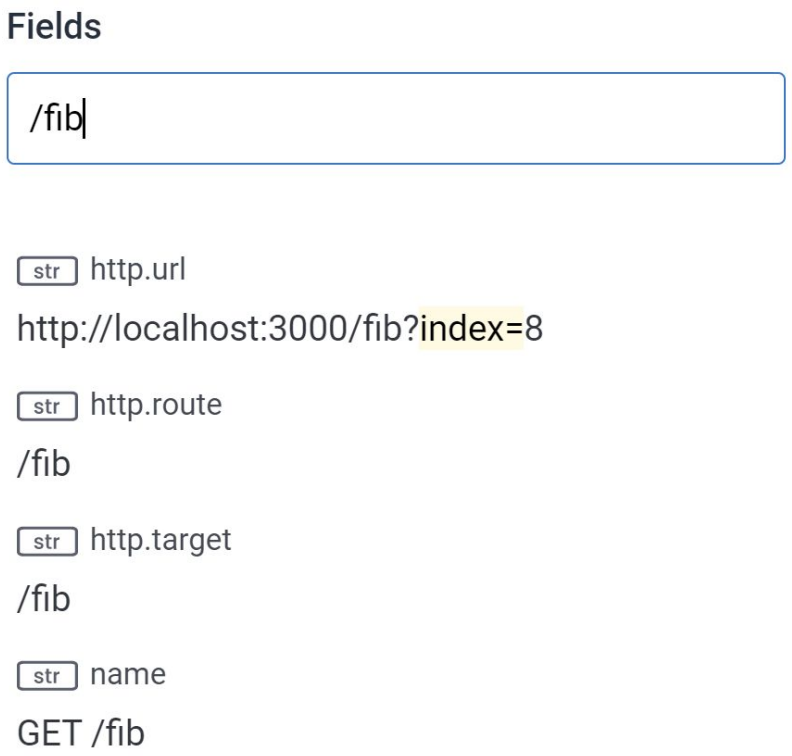
Search a trace

Search for spans that contain a phrase anywhere in their fields



Search a span

On the right, search all the fields in the selected span.

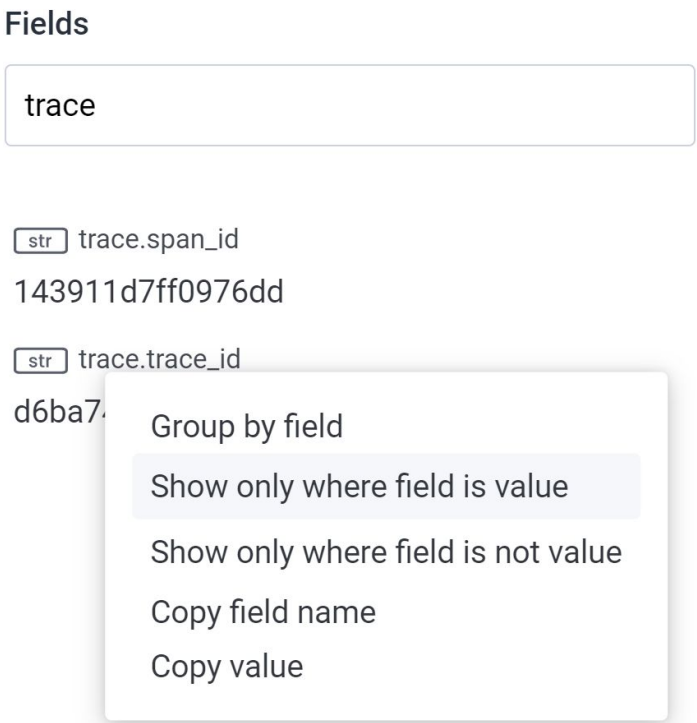


Query by a value

After searching for a field (say, trace.trace_id)

Click the three dots next to the value to get a handy menu

“Show only where field is value” gets you to a query



VISUALIZE	WHERE	GROUP BY
COUNT HEATMAP(duration_ms)	service.name = sequence-of-numbers trace.trace_id = d6ba7420b6bbd9b8e094dff4784e07d7	None; don't segment
ORDER BY	**LIMIT**	**HAVING**
COUNT desc	None	None; include all results

Run Q

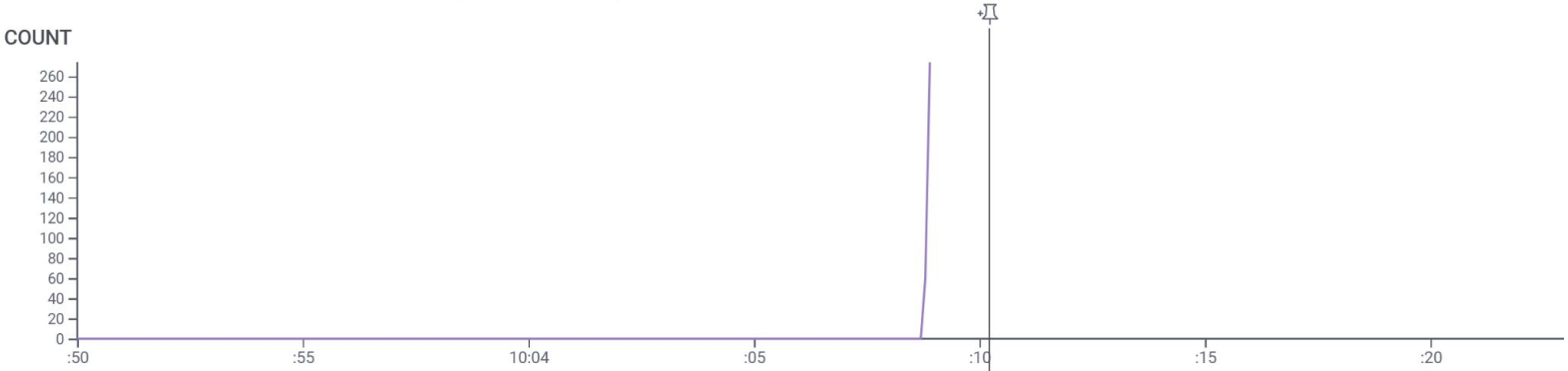
Run a second

Results | BubbleUp | Traces | Raw Data

Compare to 33 seconds prior

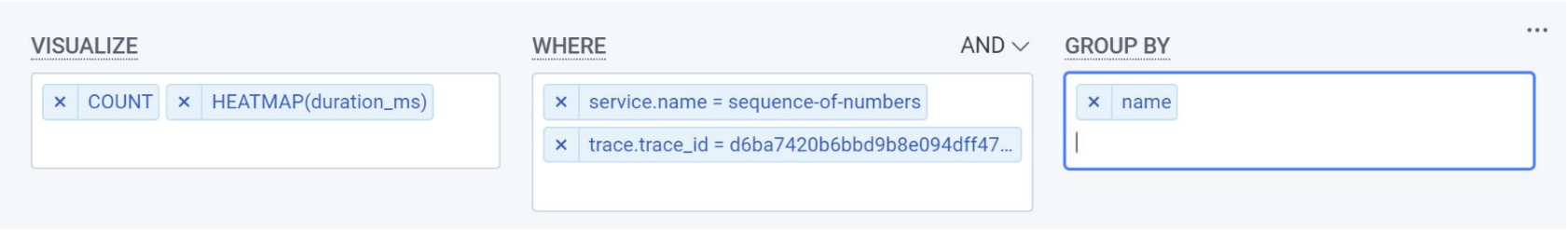
Graph Set

May 18 2022, 10:03:50 AM – May 18 2022, 10:04:23 AM (Granularity: 100 ms)



Try customizing with GROUP BY

Add “name” to the GROUP BY then Run Query



Then scroll down to the table of results

name	COUNT	HEATMAP(duration_ms)
middleware - expressInit	67	
middleware - query	67	
GET /fib	67	
request handler - /fib	67	
HTTP GET	66	

Add visualizations to see more results

In VISUALIZE, add:

P99(duration_ms)

Then add that to the ORDER BY too.

Now see the slowest spans in the result table!

name		
GET /fib	67	59.06406
HTTP GET	66	43.69894
middleware - query	67	0.05222
middleware - expressInit	67	0.04173
request handler - /fib	67	0.00333

elapsed query time: 38.444326ms # results: 5 rows examined: 12,942 nodes reporting: 100%

Putting it all together: custom queries

VISUALIZE

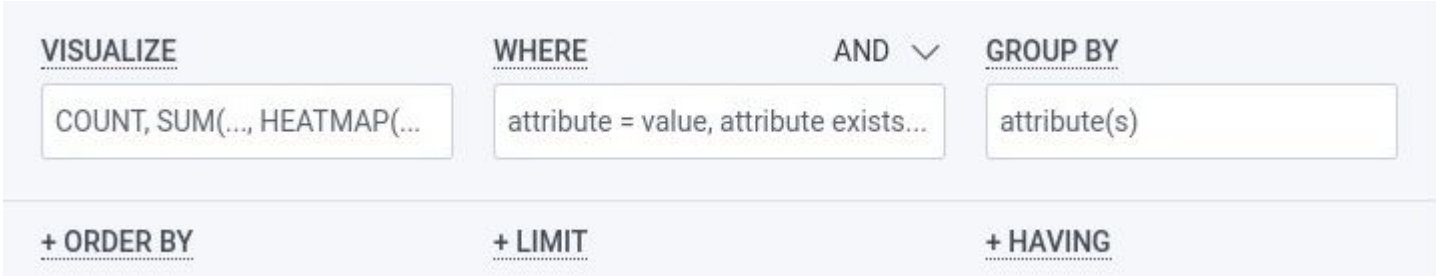
WHERE

GROUP BY

ORDER BY

LIMIT

HAVING

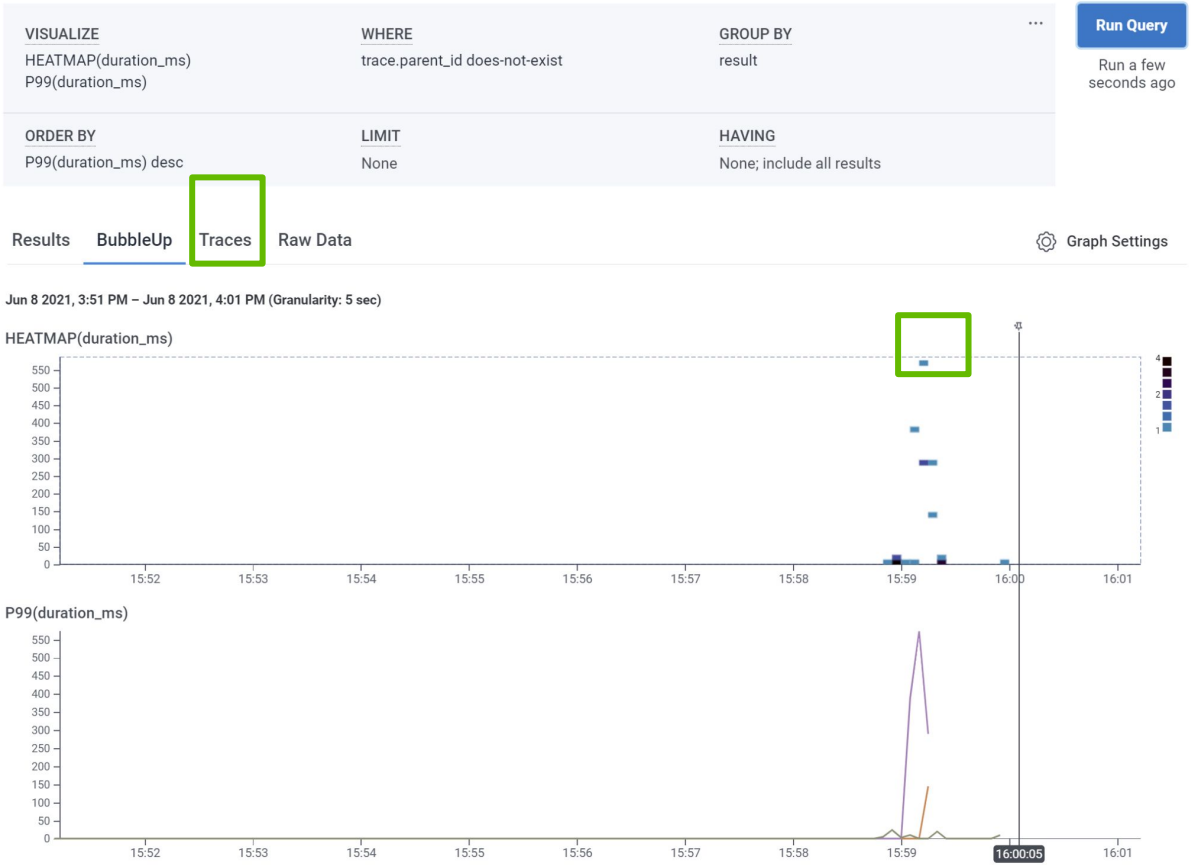


Find an especially slow trace

Go to the **Traces** tab now.

Click on the slowest trace in the heatmap.

The next screen will be your detailed Trace display.



Summary

In this section, we:

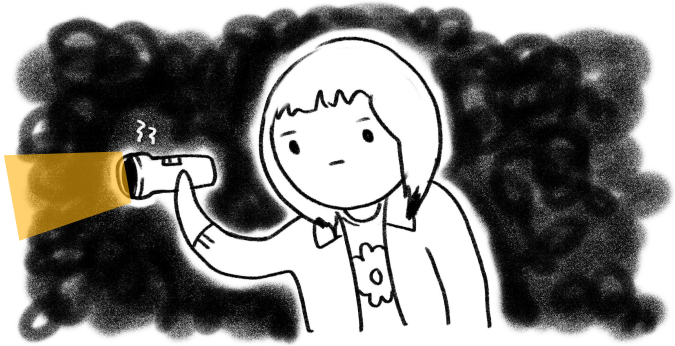
- Examined a trace for a specific request
- Identified outliers using Heatmaps and BubbleUp
- Customized our queries and used multiple visualizations
- Inspected our raw data and found the slowest traces

Puzzles for the class

How many times is /fib?index=1 *inside* a call to /fib?index=5?

How much longer (*P99*) does it take to evaluate /fib of 5 compared to of 4?

Why is this so slow as *index* increases?



Puzzles for the class

How many times is /fib?index=1 inside a call to /fib?index=5?

Hint: Find an example trace.
Use “WHERE http.url CONTAINS index=5”
Then, choose “Traces” tab.

How much longer (*P99*) does it take to evaluate /fib of 5 compared to of 4?

Hint: Use both
VISUALIZE P99(duration_ms)
GROUP BY http.url

Why is this so slow as index increases?

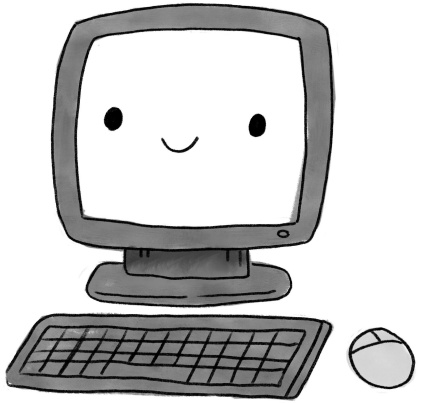
Don't forget to select the **Run Query** button!

There is **more than one method** to arrive at the answer.

Answered the puzzles? You're done for now!

Come back at top of the hour.

If you're stuck, speak up on Slack or join a help breakout room!



Puzzles for the class

How many times is /fib?index=1 inside a call to /fib?index=5?

Hint: Find an example trace.
Use “WHERE http.url CONTAINS index=5”
Then, choose “Traces” tab.

In Query Builder, start with:
VISUALIZE COUNT

A Solution:
WHERE trace.parent_id does-not-exist
GROUP BY http.url
[For Python and Go apps, use *http.target*]

Focus on */fib?index=5*.
Below the graph, **select** [···] button in *http.url* column in the */fib?index=5* row.
Choose "Show only where http.url = http://intro-to-o11y-nodejs.glitch.me/fib?index=5

Select any point on graph to see the Trace Detail View and **count** how many /fib?index=1 you see.

Puzzles for the class

How much longer (***P99***) does it take to evaluate /fib of 5 compared to of 4?

Hint - Use Query Builder & add to your query:
VISUALIZE P99(duration_ms)
GROUP BY http.url

[For Python and Go apps, use *http.target* instead.]

A Solution:
Return to your query builder display by selecting the arrow next to the trace name in the upper left corner.

Add to your query:
VISUALIZE P99(duration_ms)
GROUP BY http.url

Two graphs now appear.

In the table of results, sort and compare duration_ms for /fib of 5 vs. /fib of 4.

Puzzles for the class

Why is this so slow as index increases?

Examine a Trace in Trace Detail view.

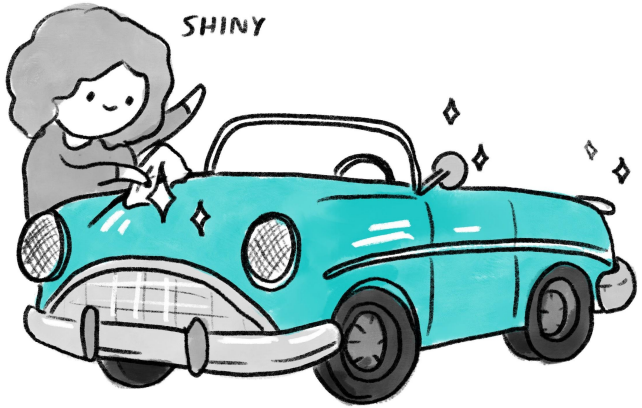
Notice the double recursion and
no caching present...

Where to go from here

Section Overview

In this section, we will:

- Add custom instrumentation
- Explore the dataset schema
- See how querying combined with collaboration tools help to quickly diagnose a problem



Add more fields and instrumentation

Edit or add at least one attribute name and one attribute value. (e.g. "$firstName was here")

Look for and change the key and value parameters to `SetAttributes()`

Check your fields in Honeycomb

Look on the right hand side for Dataset Details > Schema to see field names & types, and annotate field descriptions.

Go to Dataset Settings > Schema to coerce types.

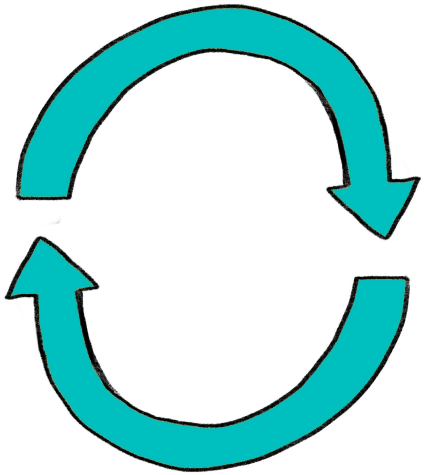
The core instrumentation loop

Start with auto-instrumentation.

Manually instrument key fields, expensive function calls, & mystery slowness.

Look at production to verify instrumentation correctness.

Repeat!

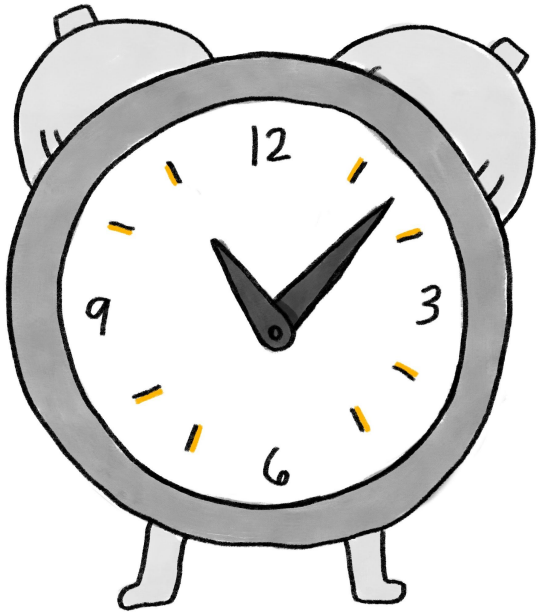


Using triggers

Set up a trigger to send you an email if someone triggers fib with a *parameter* of 5 or higher

(and then test it)

Set up a trigger on your account to send you an email if someone's /fib query takes longer than 10 seconds.



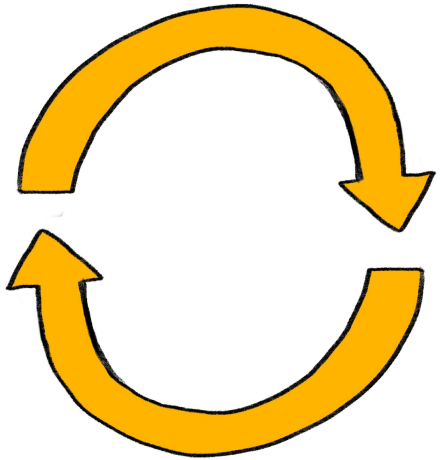
The core debugging loop

https://ui.honeycomb.io/quickstart/datasets/tracing-tour

Start with a heatmap. Run BubbleUp on outliers to find interesting groups.

Group by / filter by fields (especially instrumented fields) to find the culprit.

Look at an example trace or two.



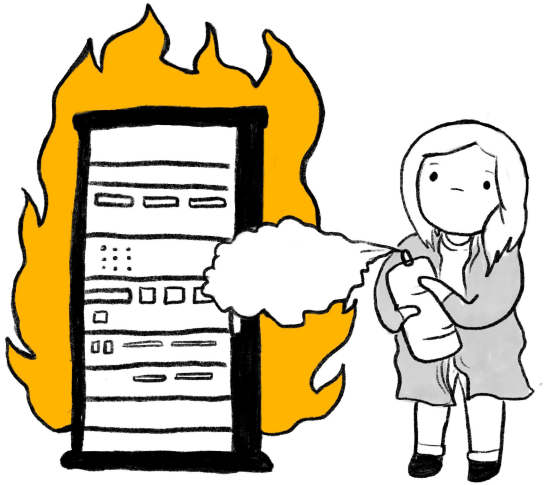
Don't panic, you have Honeycomb!

Remember remediation comes first.

Markers help you find what changed most recently.

Use bubbleup, group by, and traces to navigate to the problem.

(and Honeycomb SLOs can also help as you get more advanced!)



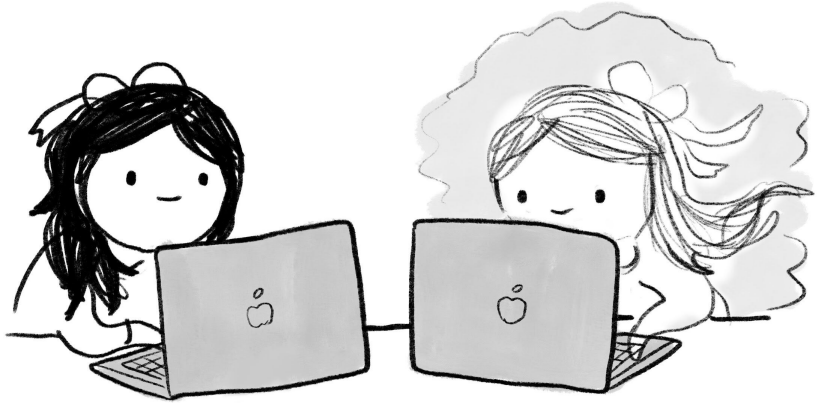
Using query history

Honeycomb is a virtual lab notebook!

Pop open query history on the right.

Pop open saved queries on the left.

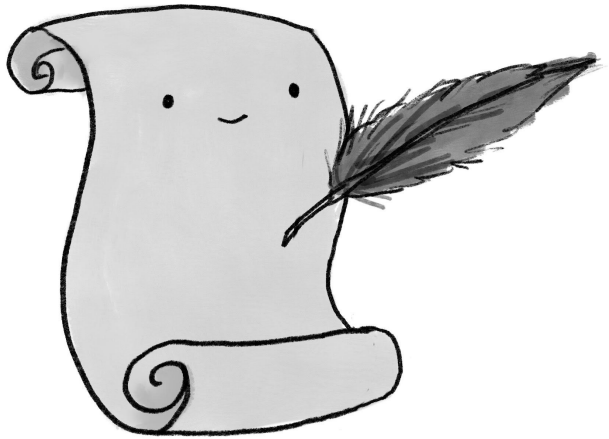
Shared with future teammates & colleagues! (and yourself, too)



Summary

In this section, we:

- Learned about adding custom instrumentation
- Explored Honeycomb features that keep everyone in the loop, like query history and triggers



Share with your colleagues

Show and tell

If you have a coworker here, you can share with them!

Otherwise, add a coworker to your team and show them!

Bee a hero, make others heroes!

Instrumenting with OTel is easy

Sending data to Honeycomb is free!

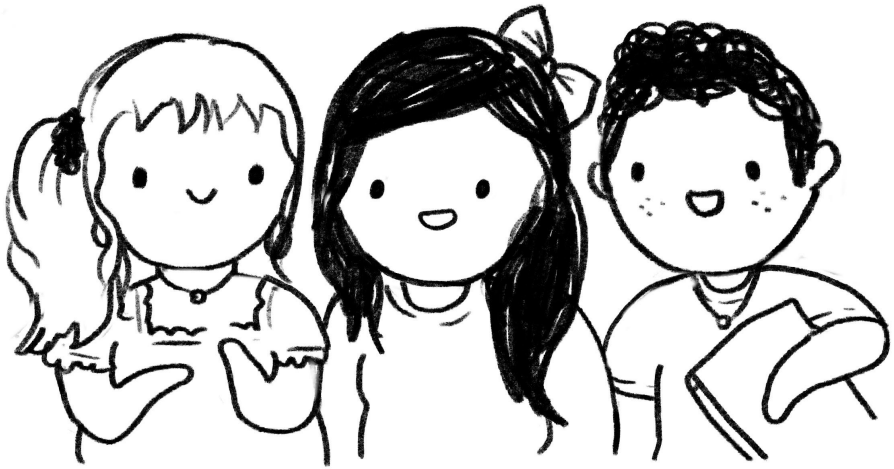
Querying reveals insights

We're here to help you!

Feel free to add OTel and Honeycomb to your own apps!

Here's what you learned today

- Why observability matters
- How to add instrumentation and get telemetry data
- How to use Honeycomb to answer questions and achieve observability
- How to add Honeycomb to your debugging workflow
- How to share these lessons with your team!



Event Survey + i test in prod t-shirt!

We value your feedback and read every comment!

All respondents will receive a t-shirt upon completion. The survey closes tomorrow, June 16 at 12 p.m. PT.

Secret word: opentelemetry

Please see the survey link in the Zoom chat window now!

Thank you

We'd love to hear your feedback!

Instructors will be around Slack, give us a 👋

www.honeycomb.io