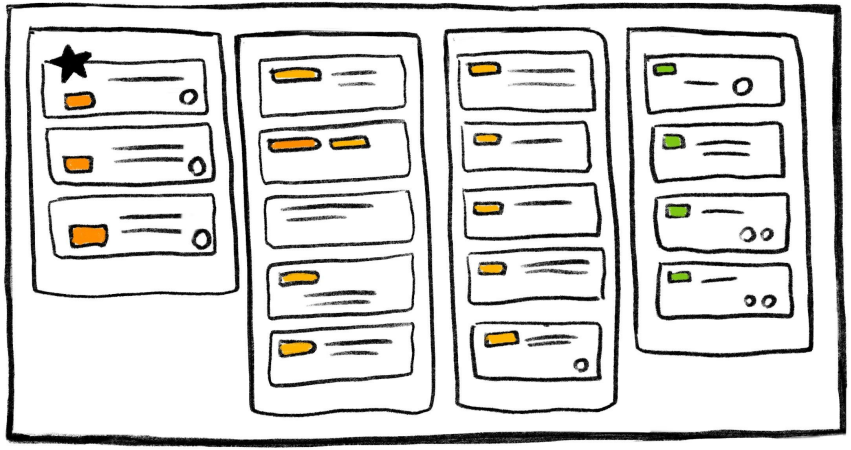
Introduction to Observability

Jan 26, 2022

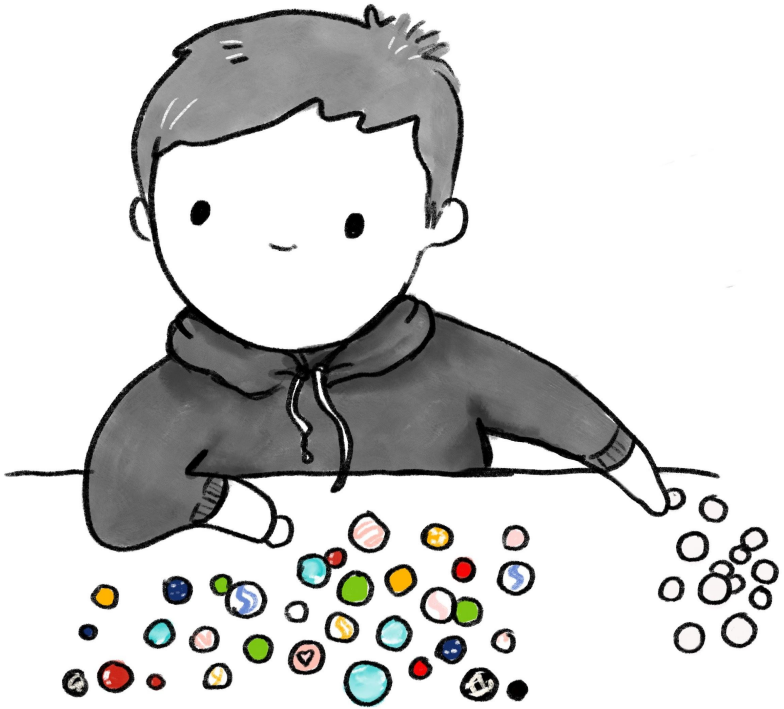
Today's agenda

- Meet your instructors
- What is Observability? Why does it matter?
- Get started with Honeycomb & OpenTelemetry
- Answer questions with Honeycomb
- Where to go from here



Some housekeeping items

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



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

Jessica Kerr

Principal Developer Advocate
@jessitron

Mary Jinglewski

Instructional Designer

Liz Fong-Jones

Principal Developer Advocate
@lizthegrey

Why does Observability matter?

and how can it make your job easier?

Developers have a lot on our plates.

We want devs to:

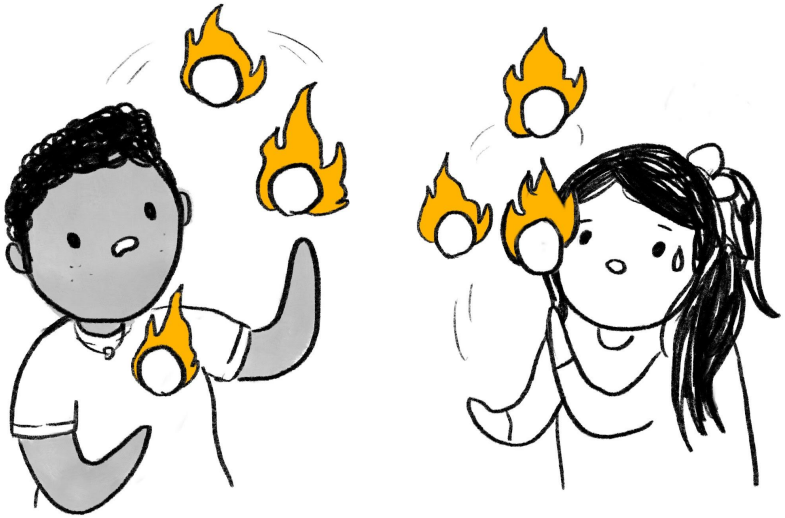
- Ship more often
- Produce better code
    - Spend less time debugging and more time creating value
- Decrease downtime

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.



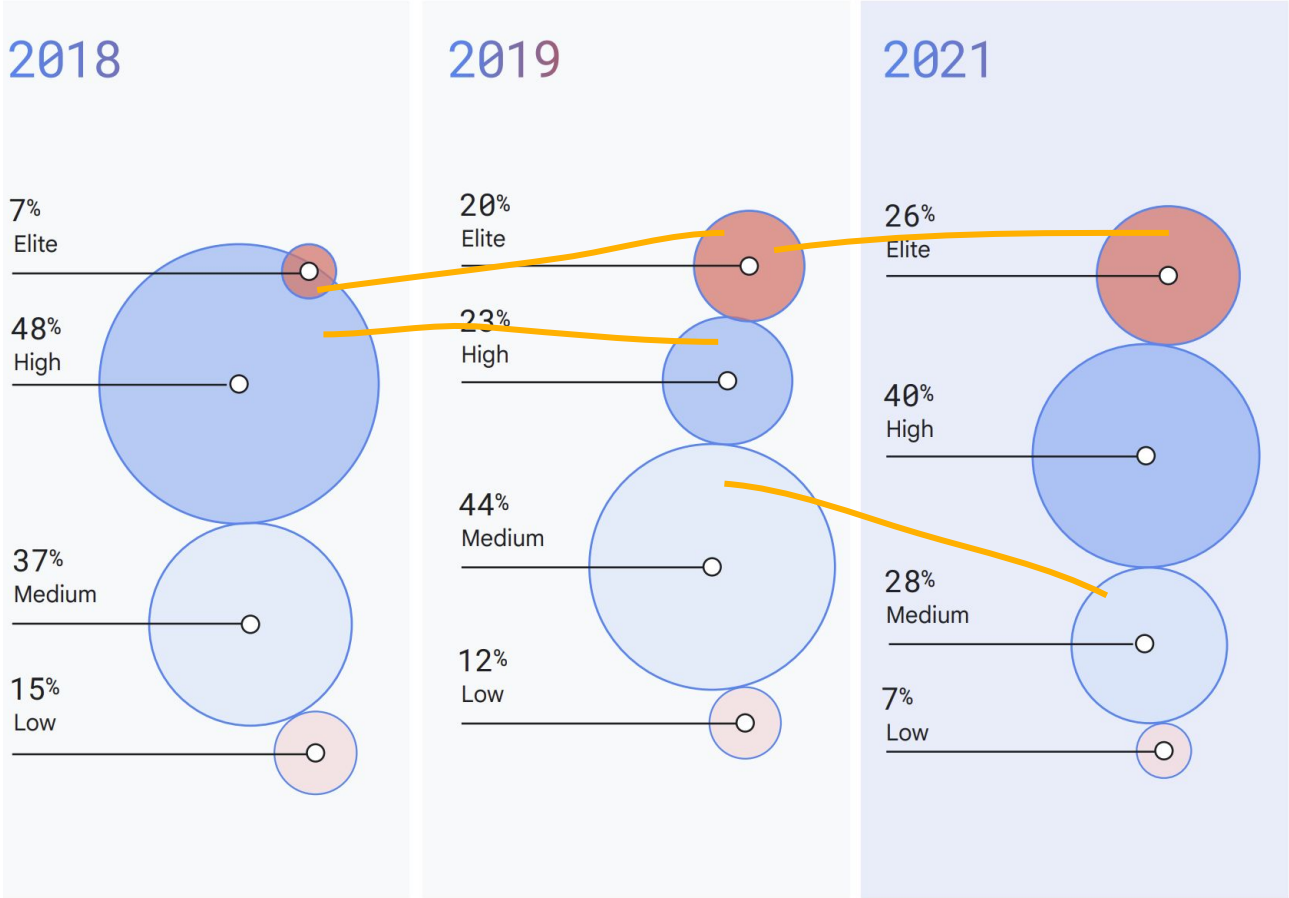
But some organizations have solved it!

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

Production excellence addresses rework

- Happier developers
  - Leverage collective intelligence and work as a unified team
  - Extend the knowledge and productivity gains to other teams
- Greater customer satisfaction
  - Faster time to recover from outage
  - Service Level Objectives met

Service Level Objectives measure success.

Aiming for 100% slows us down and makes us afraid.

Not every service has the same reliability requirement.

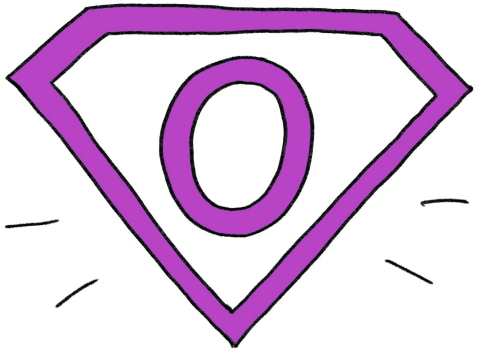
Measure good events, not good minutes.

Observability helps us debug outages.

Distributed systems experience *emergent* failure modes.

but SLOs don't tell us where it broke!

We can’t think of every possible error in advance. How can we ensure we’ll be able to debug some new failure?



> **Observability is about the unknown-unknowns. A system is observable when you can ask any arbitrary question about it and dive deep, explore, follow bread crumbs. Observability is about being able to trace the inner workings, just by observing the outside.**
>
> Charity Majors

We must be able to solve novel problems.

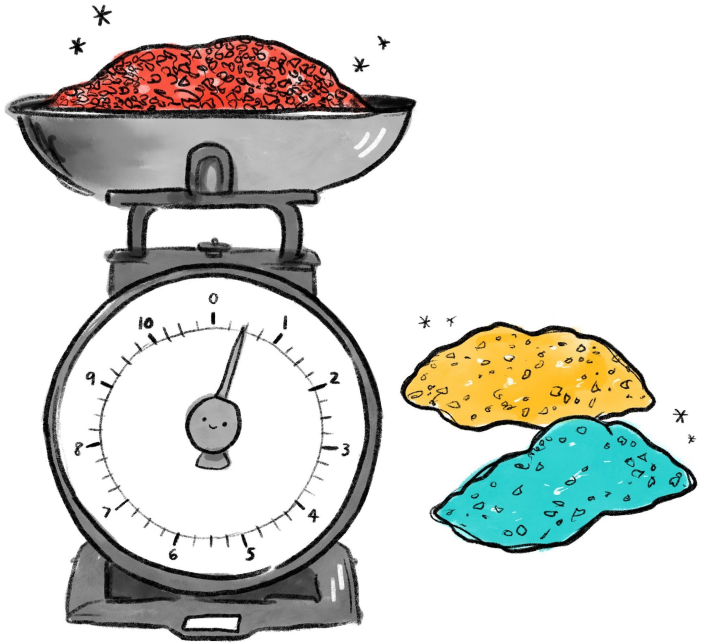
Every complex problem in production is *new*.

Answering known unknowns isn't enough.

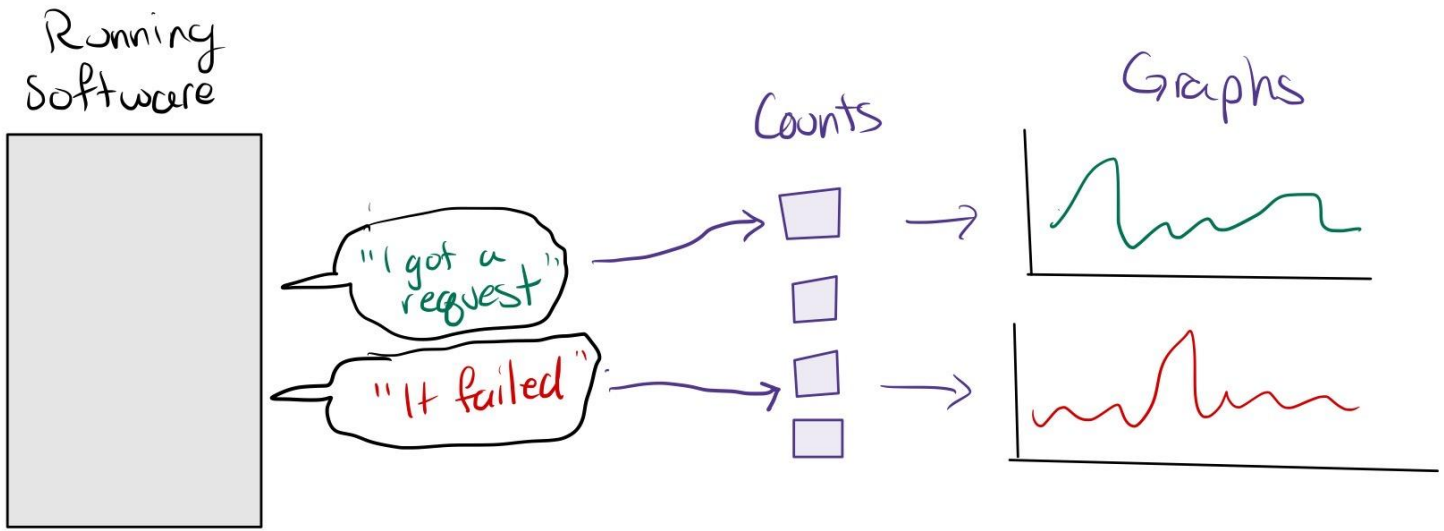
We must be able to synthesize data in new ways.

How is o11y different from monitoring?

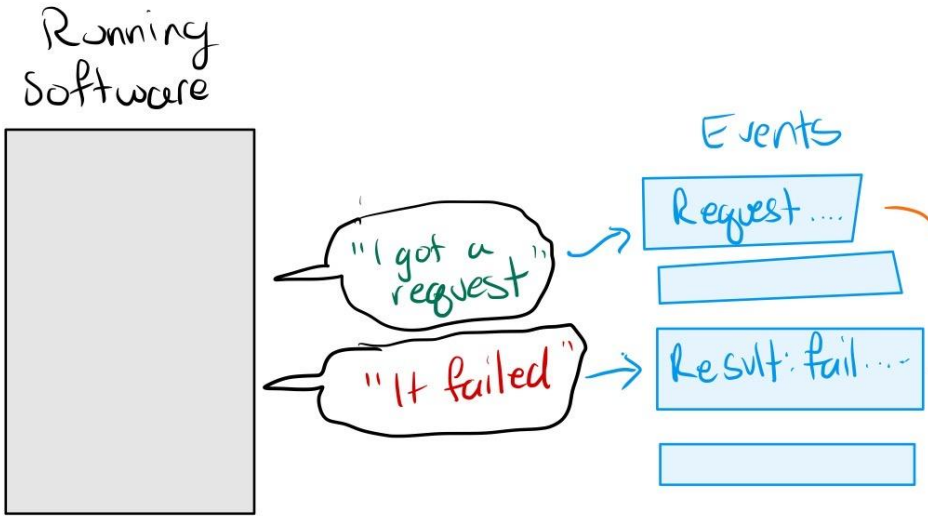
- Metrics are *reactive*.
- You have to decide ahead of time what metrics to aggregate.



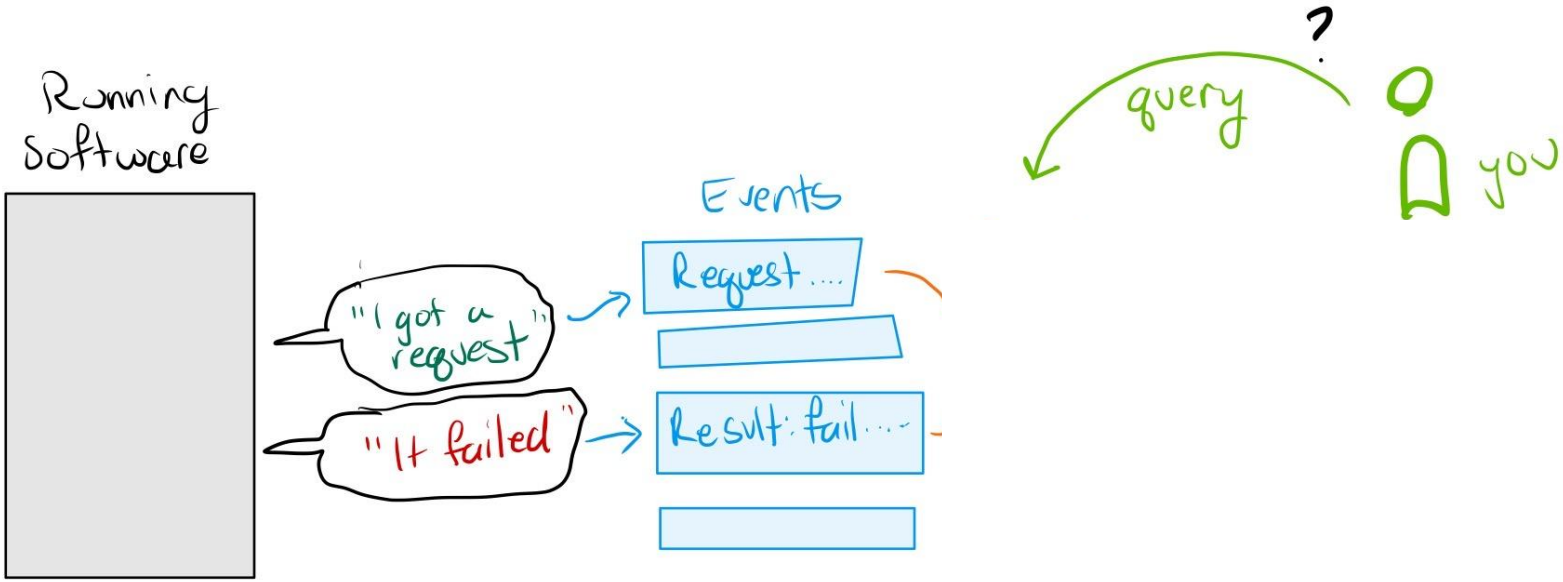
Monitoring: graph what you counted



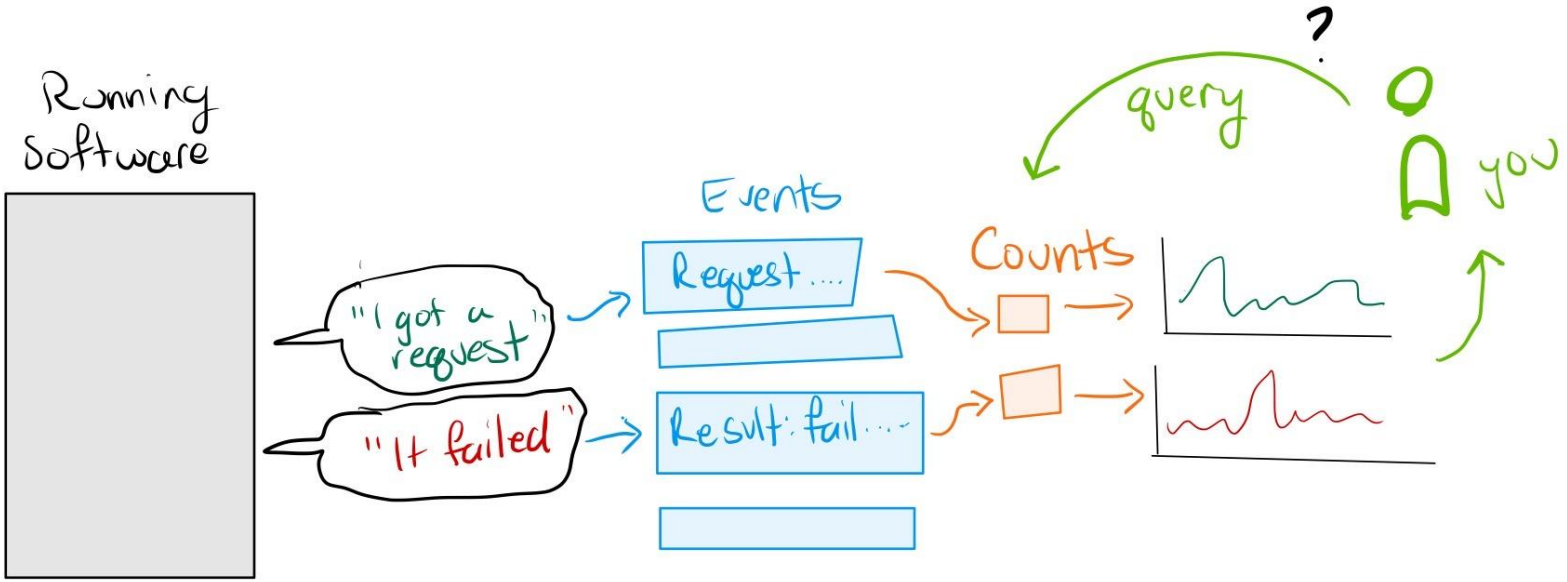
Observability: graph what you want to count



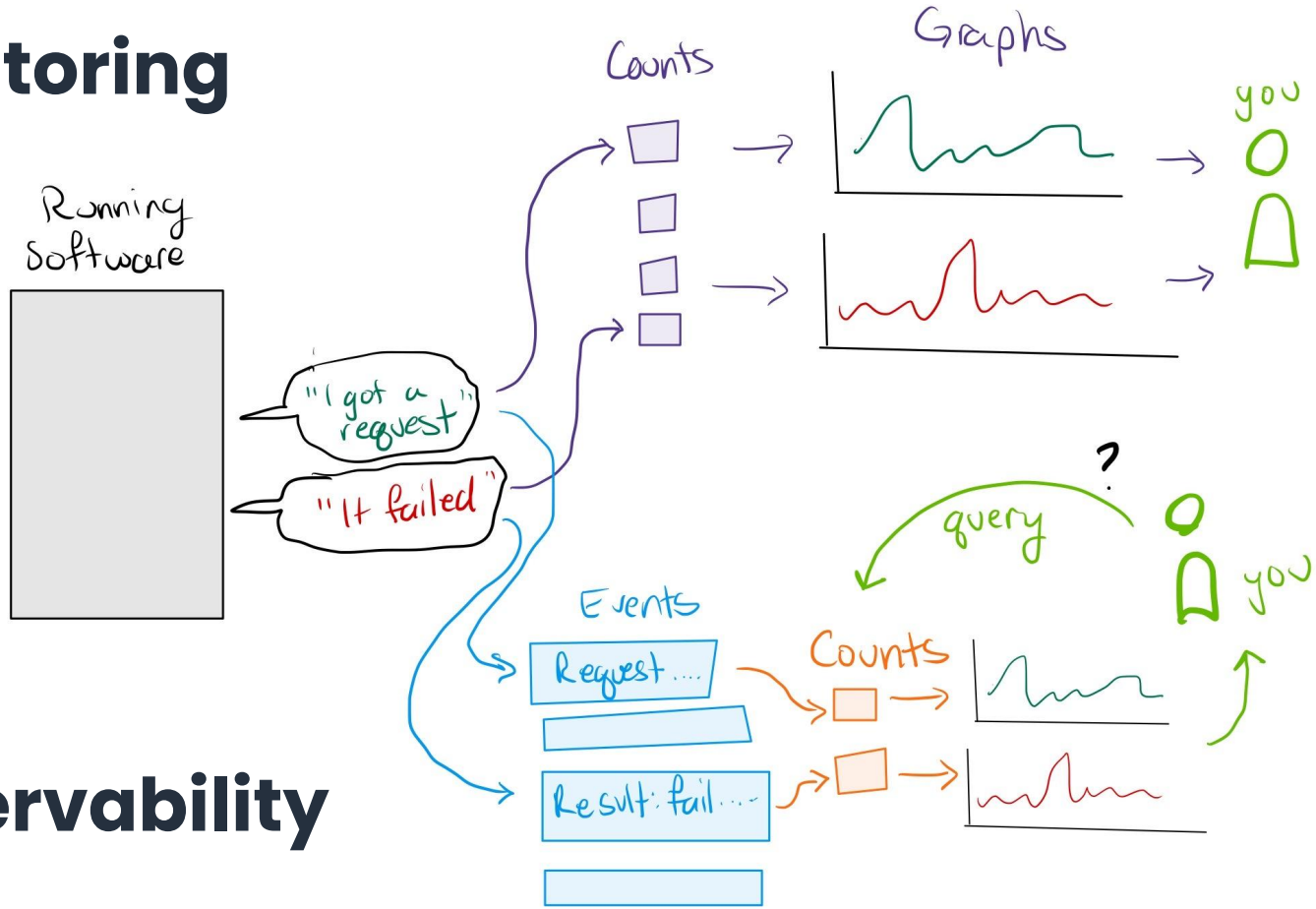
Observability: graph what you want to count



Observability: graph what you want to count



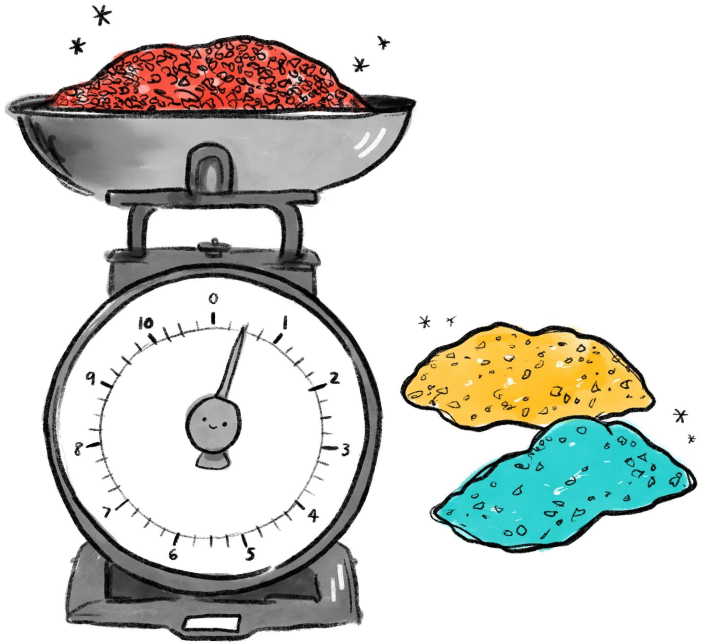
Monitoring



Observability

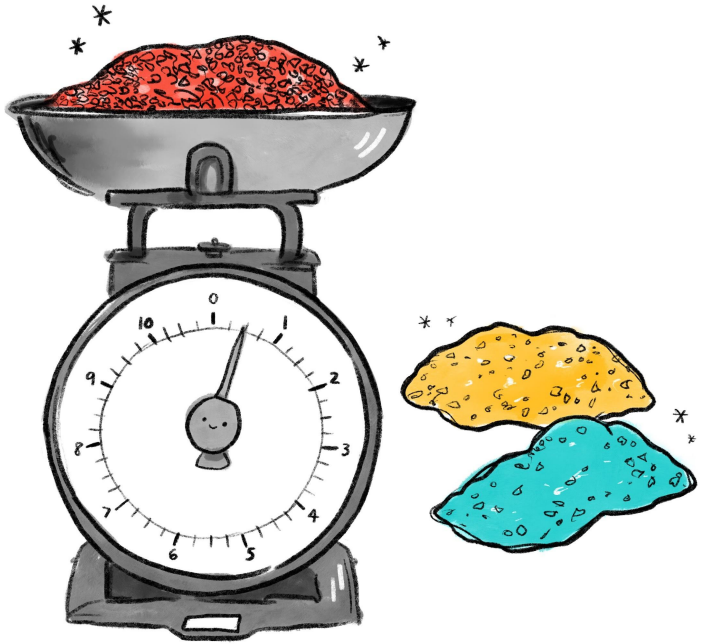
How is o11y different from monitoring?

- Metrics are *reactive*.
- You have to decide ahead of time what metrics to aggregate.
- Observability lets you graph what you need *now*.
    - by fields as granular as customer ID
    - by new combinations of fields

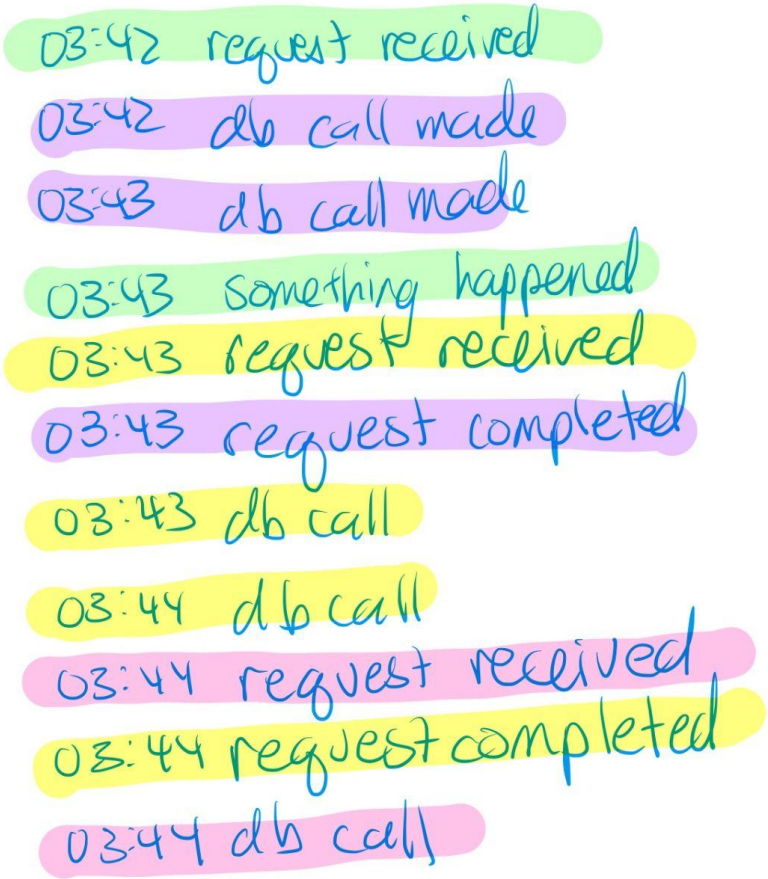


How is observability different from logs?

- Log statements lack *context*.



03:42 request received
03:42 db call made
03:43 db call made
03:43 something happened
03:43 request received
03:43 request completed
03:43 db call
03:44 db call
03:44 request received
03:44 request completed
03:44 db call


03:42 request received
03:42 db call made
03:43 db call made
03:43 something happened
03:43 request received
03:43 request completed
03:43 db call
03:44 db call
03:44 request received
03:44 request completed
03:44 db call

03:42 request received
03:42 db call made
03:43 db call made
03:43 something happened
03:43 request received
03:43 request completed, duration = 860 ms
03:43 db call
03:44 db call
03:44 request received
03:44 request completed, duration = 2604 ms
03:44 db call

03:43 request received

03:43 db call

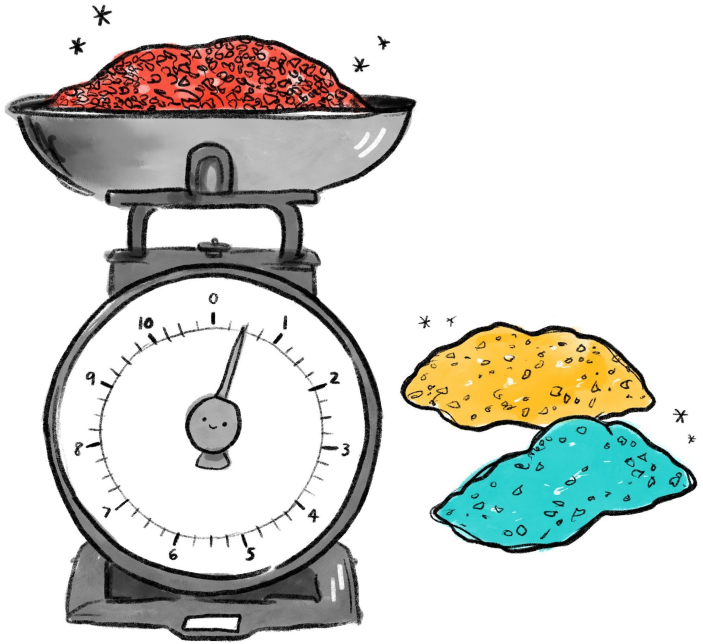
03:44 db call

03:44 request completed, duration = 2604 ms


03:43 request received & completed , duration = 2604 ms
03:43 db call, duration 60 ms
03:44 db call, duration 1842 ms

How is observability different from logs?

- Log statements lack *context*.
- Traces knit events together.
  - See where a call came from
  - See who was affected
  - See how long each bit took
- Find out which traces to look at: use custom graphs and queries.



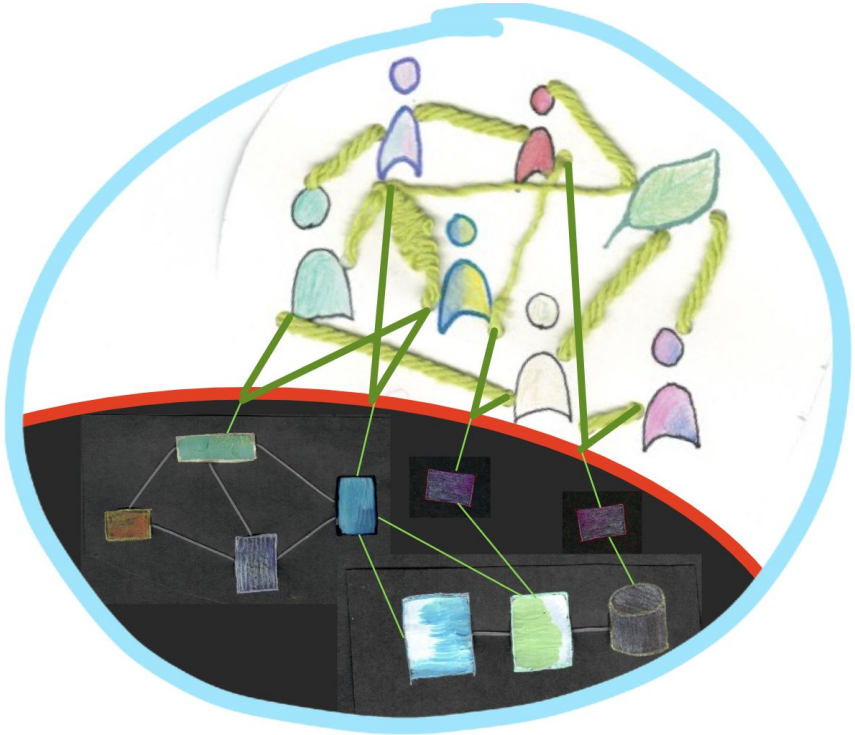
Your software and tools are on your team.

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!



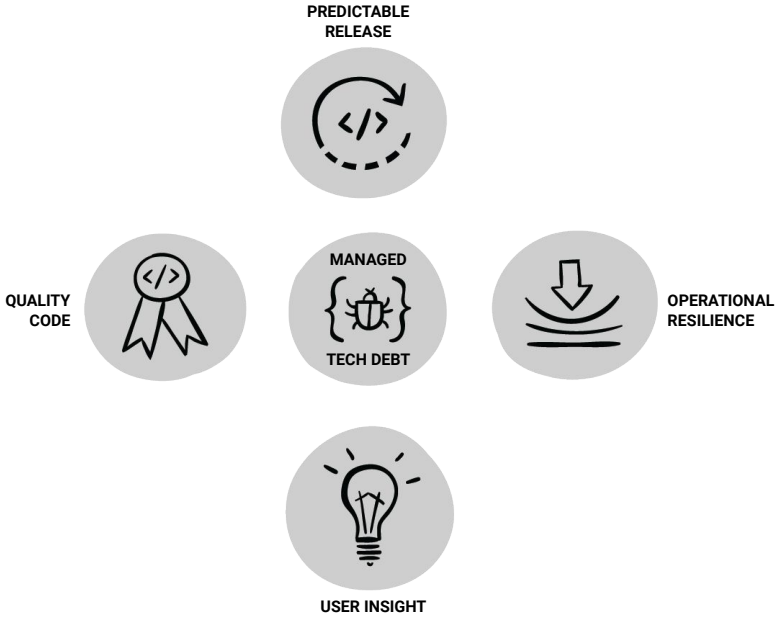
so adapt your culture, not just the tools.

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

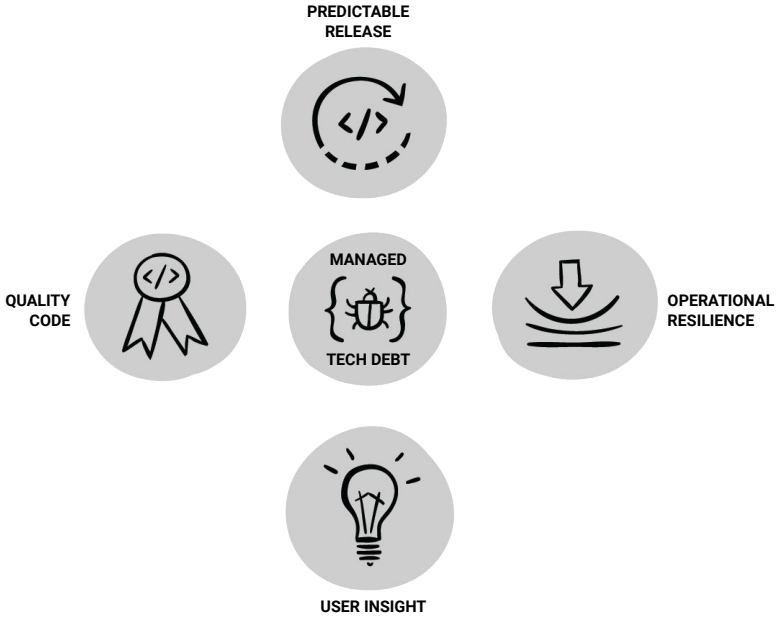
We need to accelerate feedback cycles.

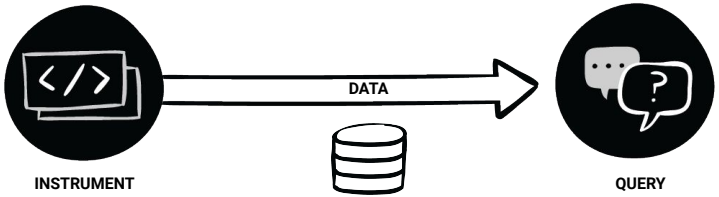
And yes, sometimes tooling can help us. But only once we've thought about our goals.

Observability use cases



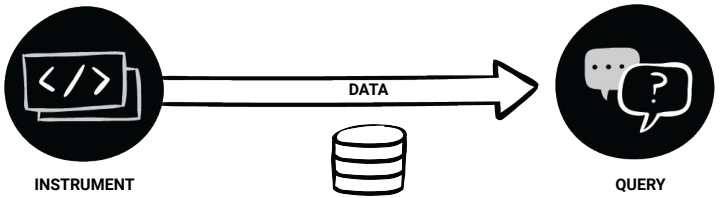
Observability use cases / not the telemetry



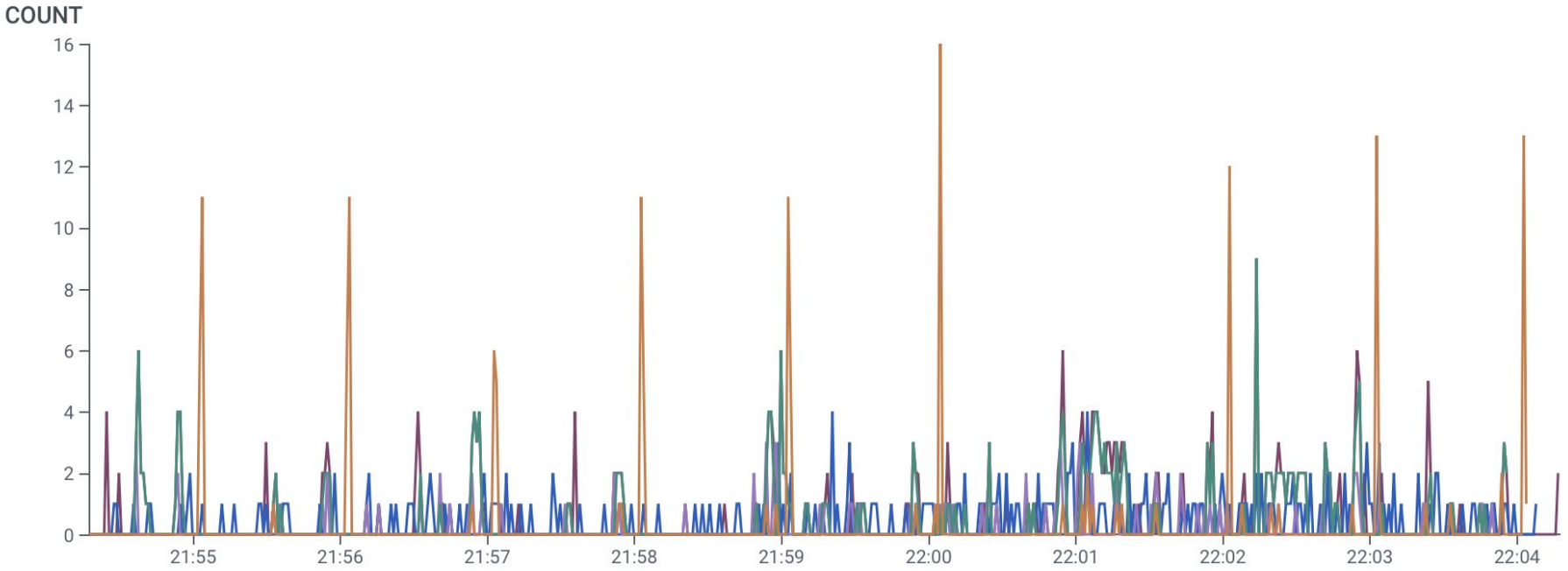


Telemetry enables observability

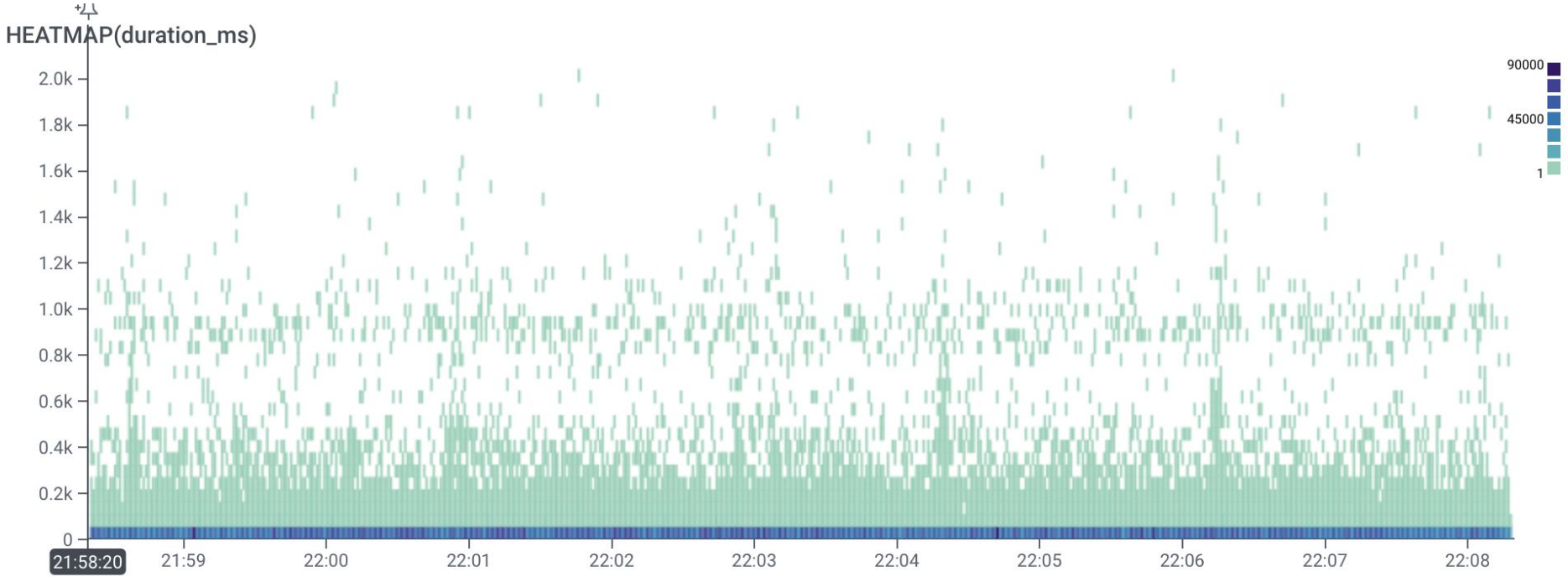
Telemetry is data that an app emits to make visible its internal state.



Telemetry feeds graphs



Telemetry feeds heatmaps



Traces are Telemetry

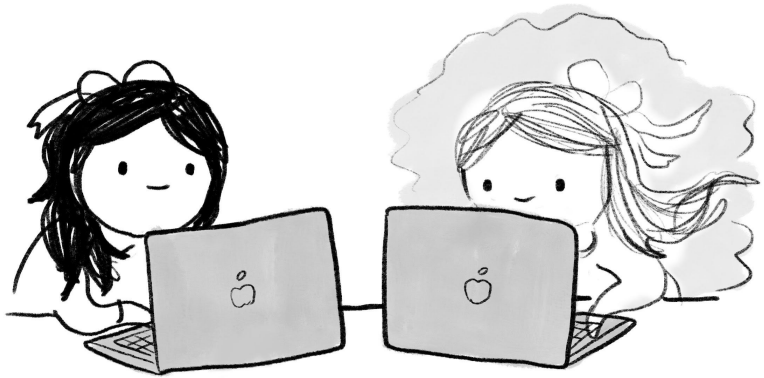
Observability requires telemetry.

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

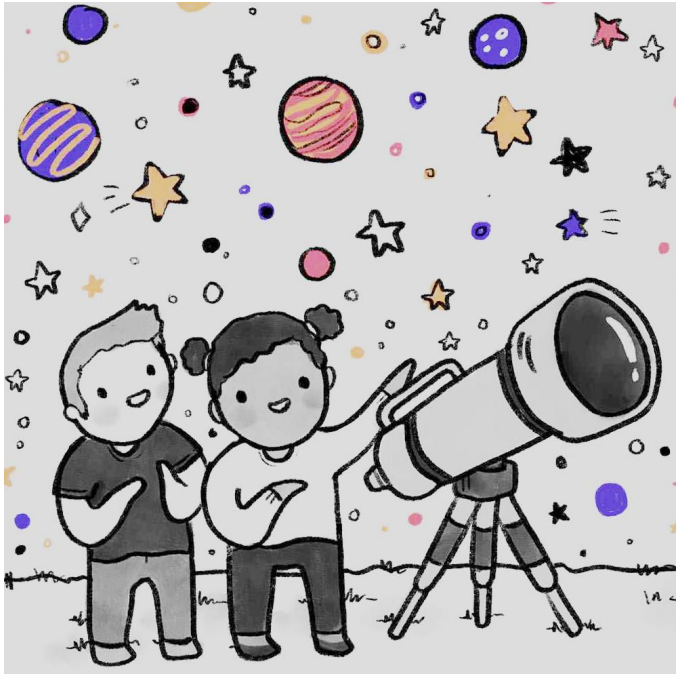


This is OpenTelemetry

In case you missed it:

OpenTelemetry is a vendor-neutral standard for instrumentation.

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



Let's put our principles into action.

Start with the fibonacci demo app, already instrumented!

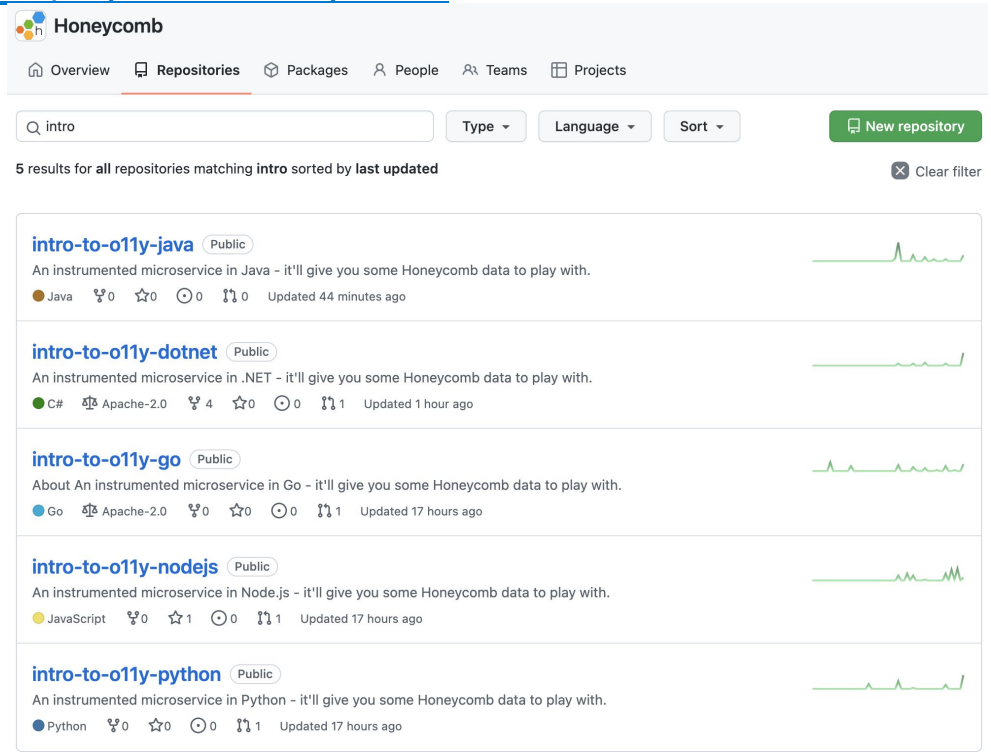
Observability requires us to have the right data to start with.

We'll be sending telemetry from the demo app to Honeycomb and analyzing it.

Choose the GitHub Repo for your language

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

Choose the repository (repo) for your language.

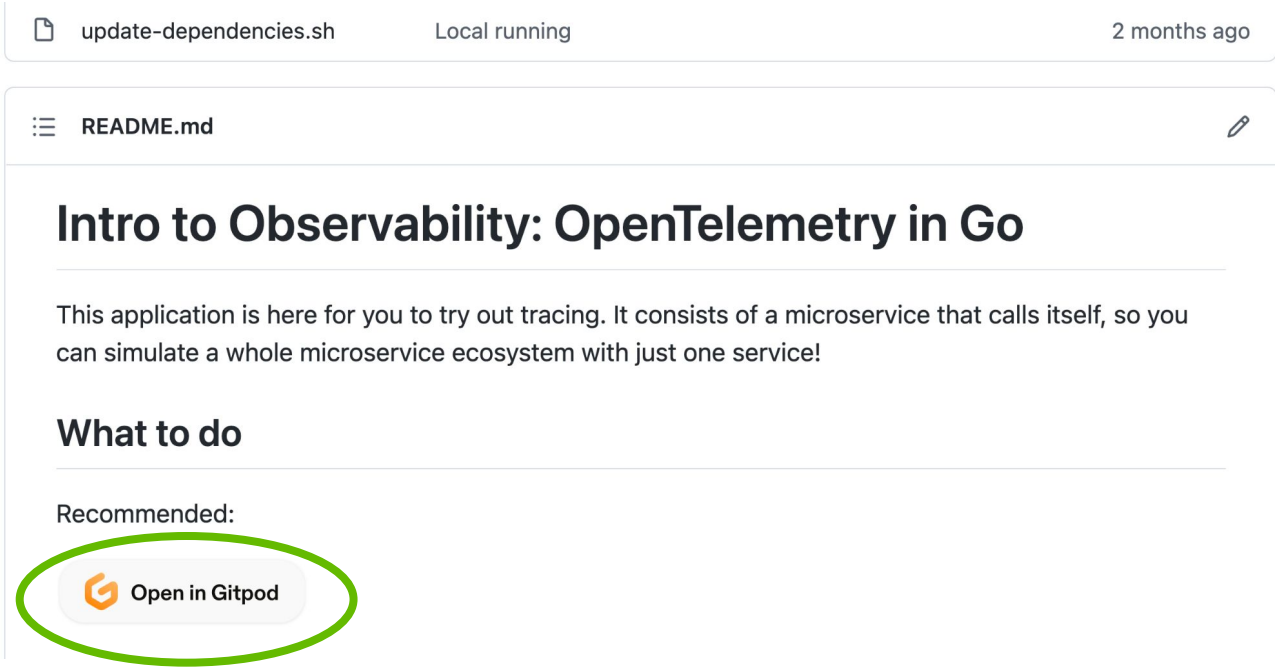


Choose the GitHub Repo for your language

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

Choose the repository (repo) for your language.

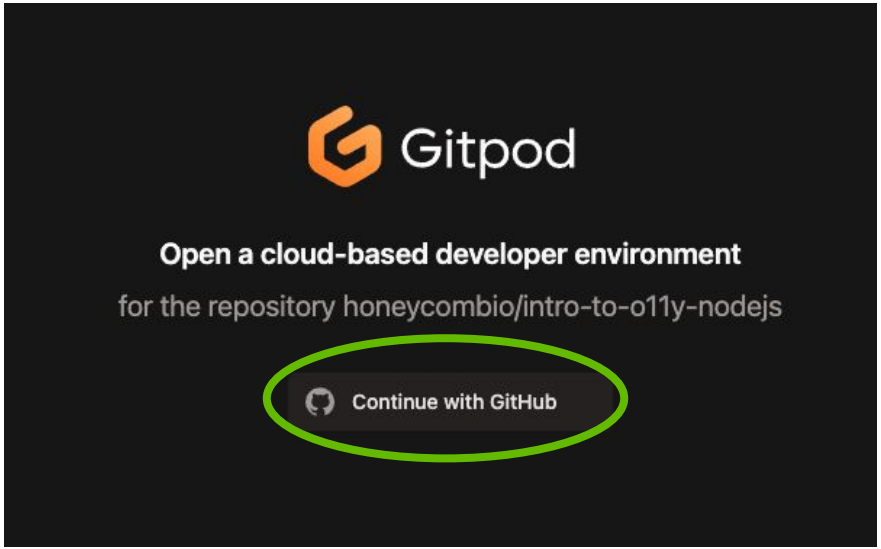
Scroll down and select the Gitpod button in the repo's README.



Log-in

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

Select **Continue with GitHub**



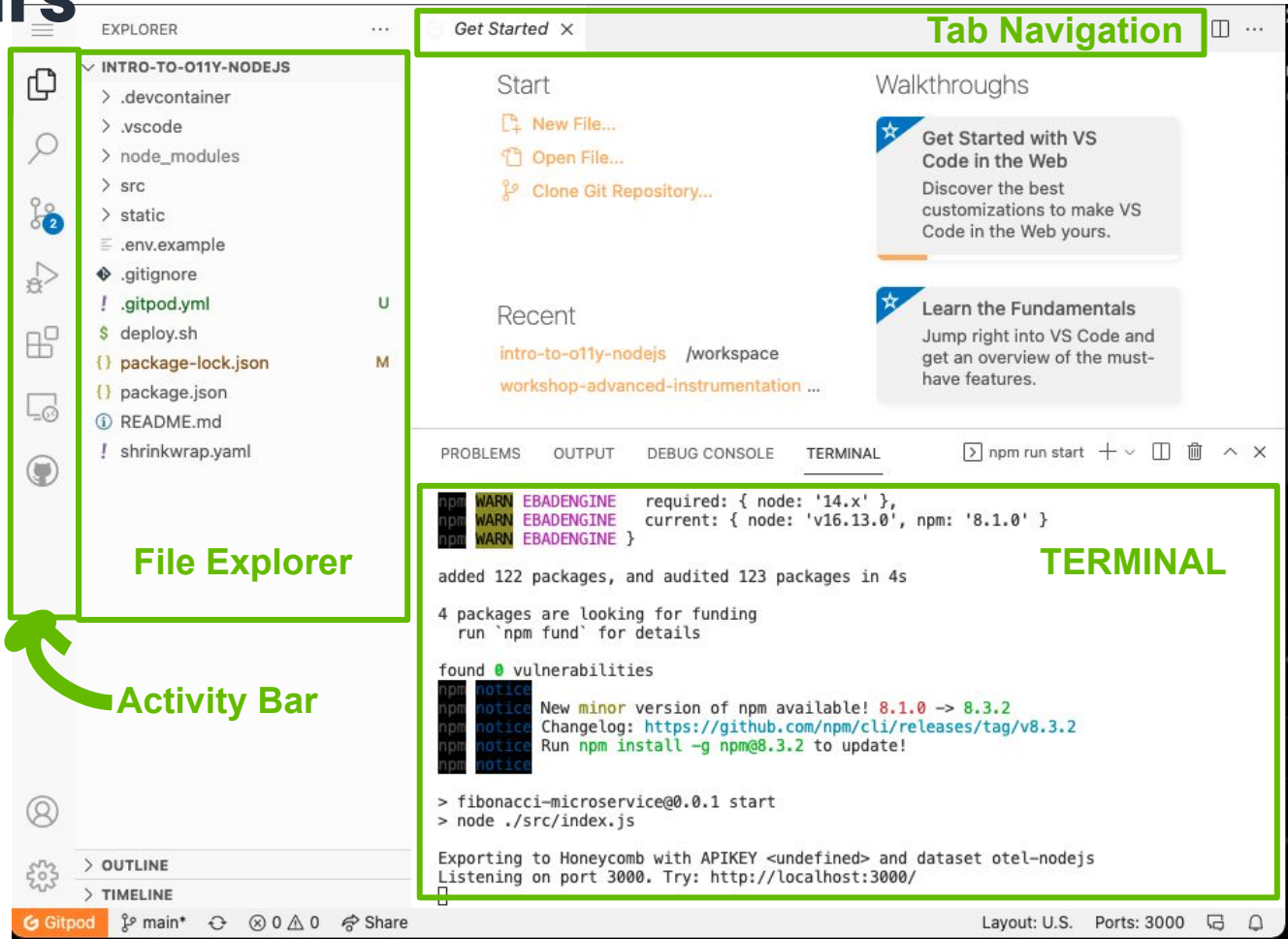
Gitpod Appears

Layout Reflects VSCode

Important areas include:

- Activity Bar
- File Explorer
- Tab Navigation
- Terminal

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

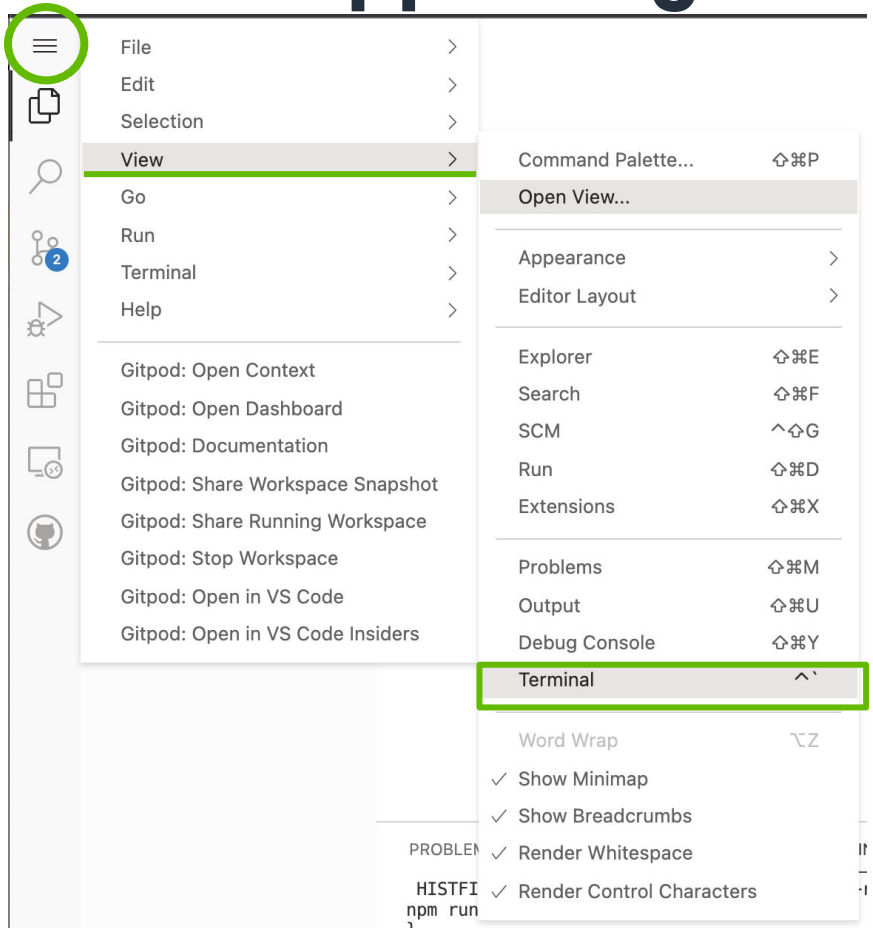


Troubleshooting: Terminal not appearing

Go to the upper left corner to the

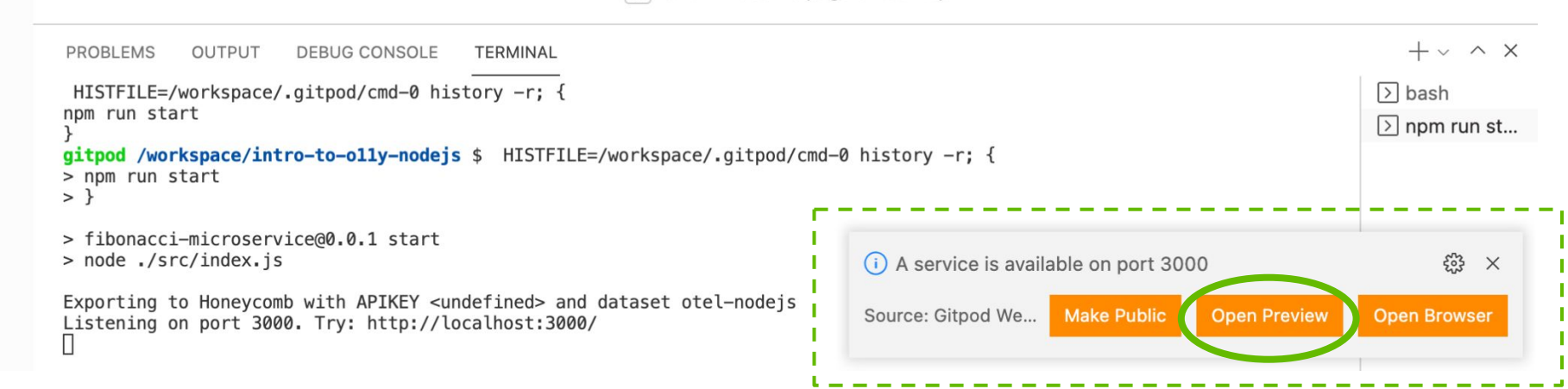
Hamburger Menu > View > Terminal

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



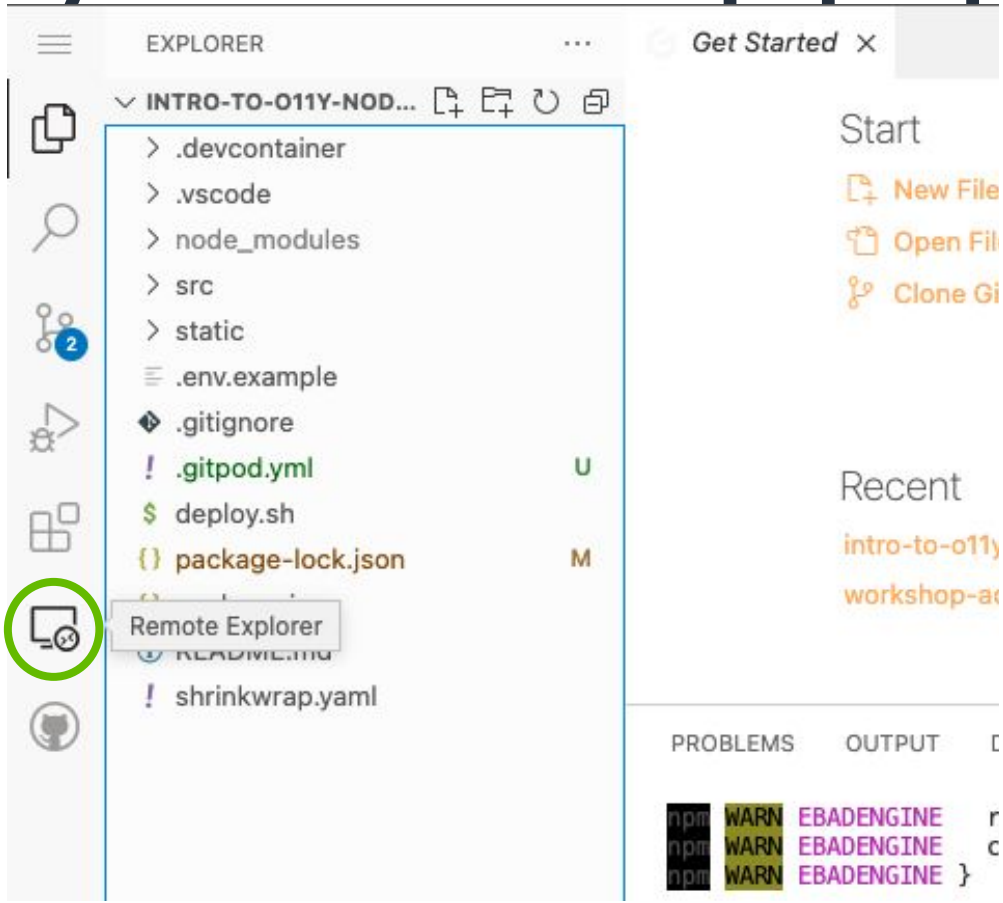
How to Open an App Preview

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**.

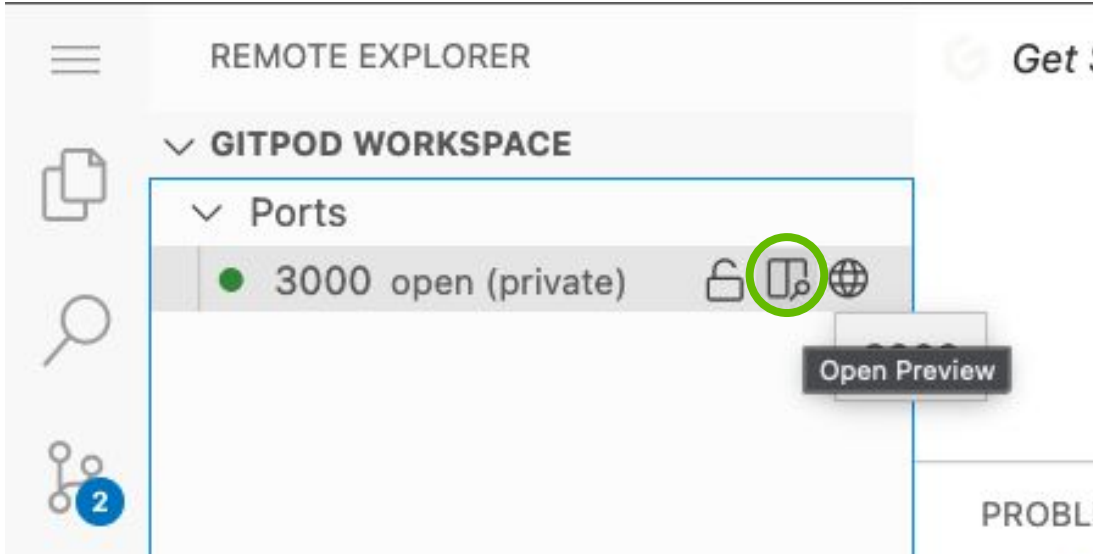


Troubleshooting: If you don't see the pop-up

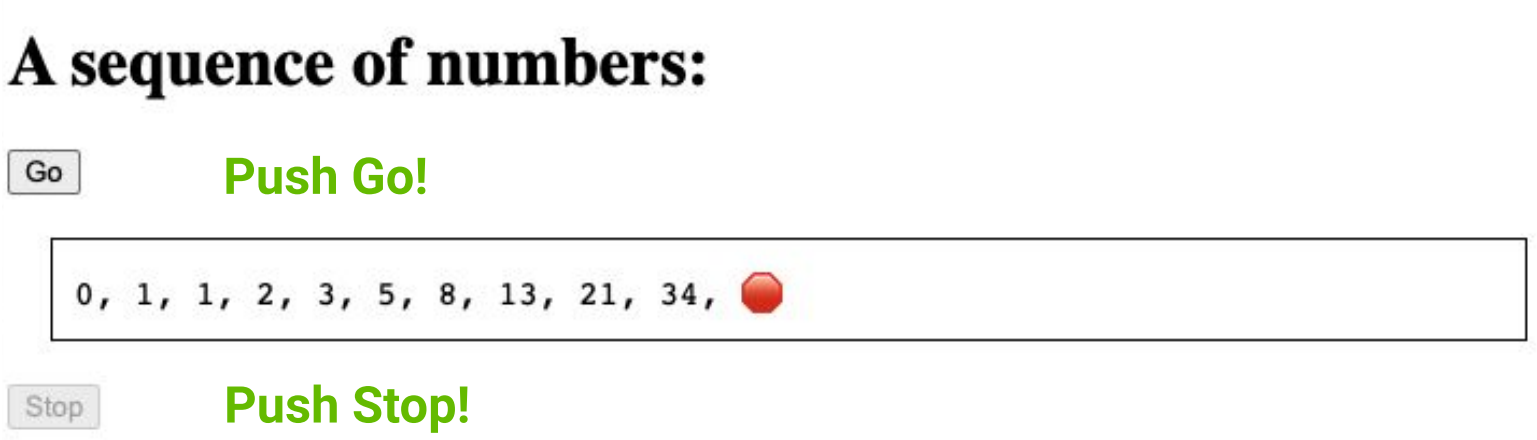
Go to Remote Explorer.

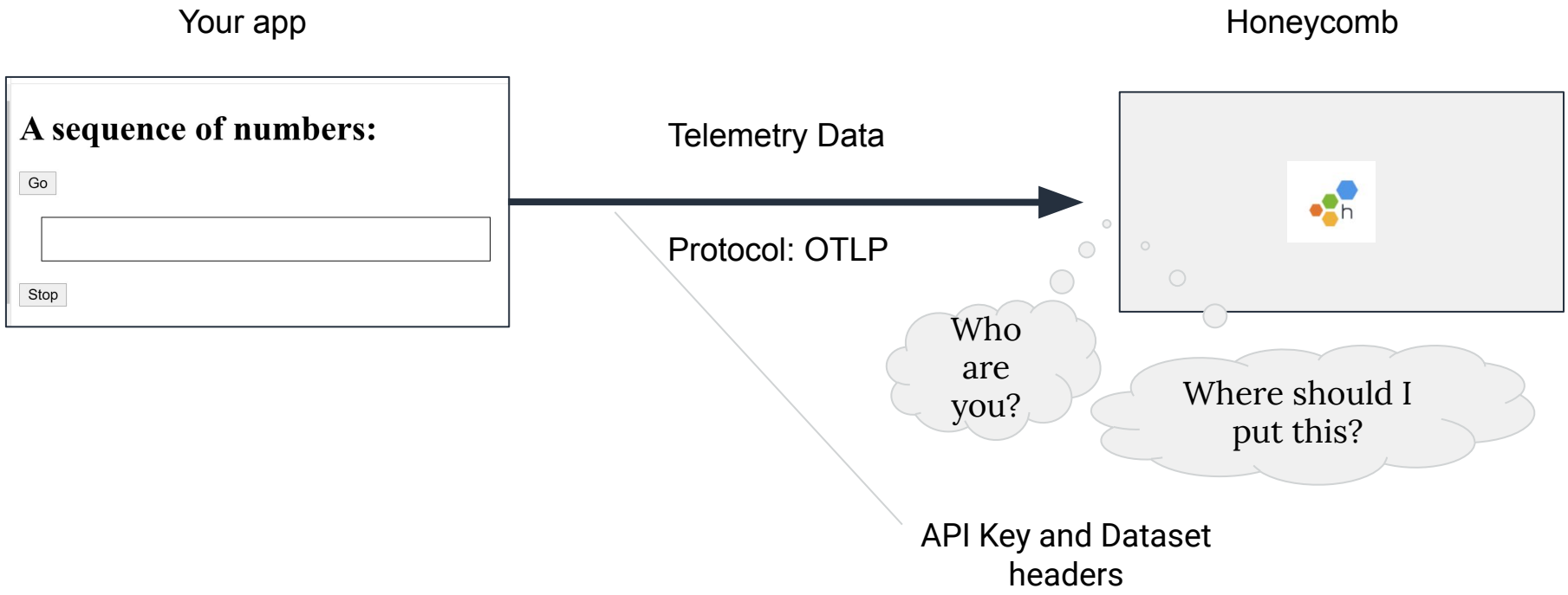
Expand *Ports* to see the listed port number. For example, *3000*

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



Try out the app



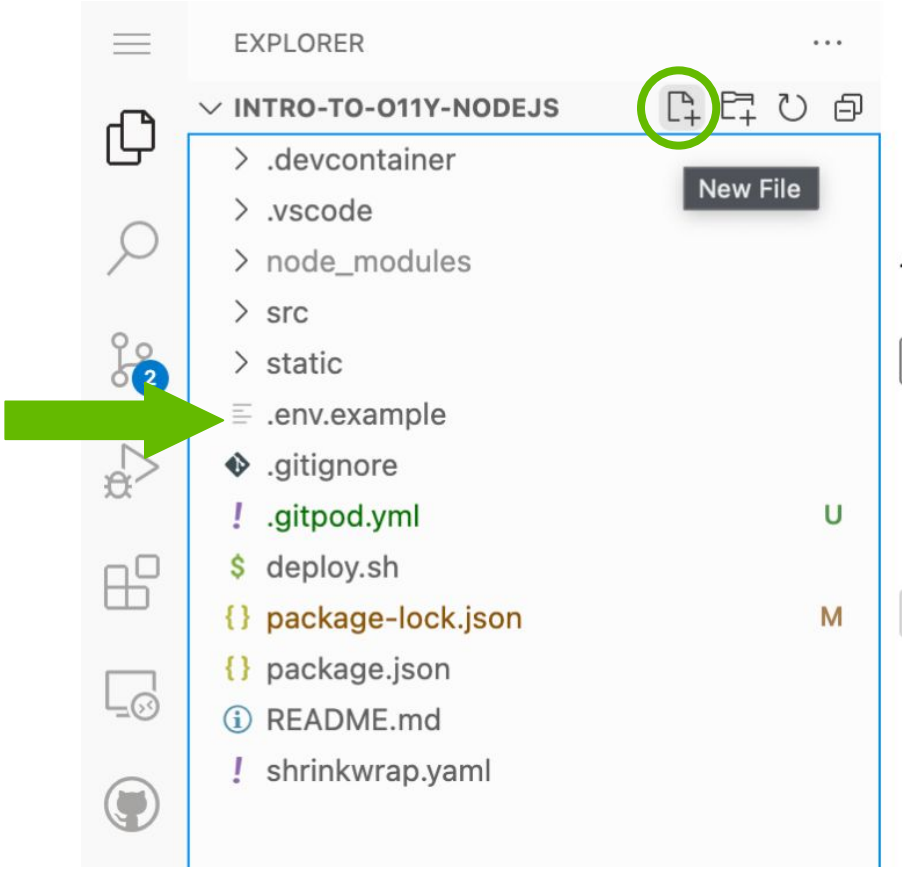


Add credentials to send to Honeycomb

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

Using the File Explorer:

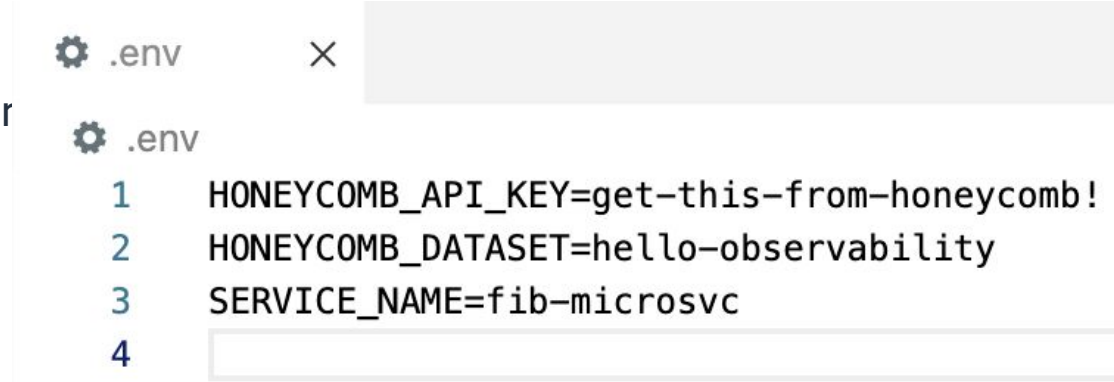
- Create a new file named **`.env`**
- Copy the contents of **`.env.example`** into the `.env` file
- Replace the credential variables in the `.env` file as needed



Add credentials to send OTLP to Honeycomb

Provide your credential variables!

- `HONEYCOMB_API_KEY` identifies your team - **REQUIRED**
  - Enter your API key from the Honeycomb UI
- `HONEYCOMB_DATASET` groups your data
  - Name your dataset here
  - Example names: “Production” or “Staging”
  - Honeycomb will automatically create your dataset once data flows.

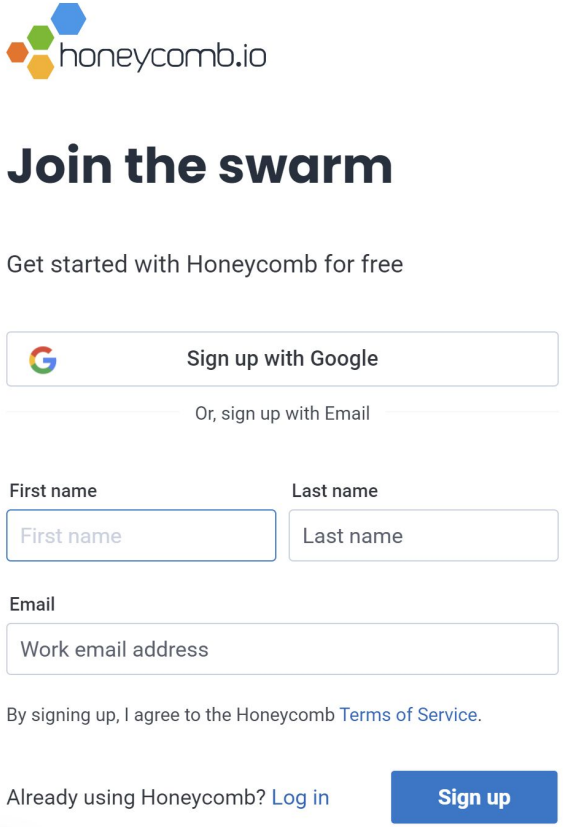


Note: `SERVICE_NAME` is optional, may not exist depending on language.

Get a Honeycomb account

https://honeycomb.io/signup

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

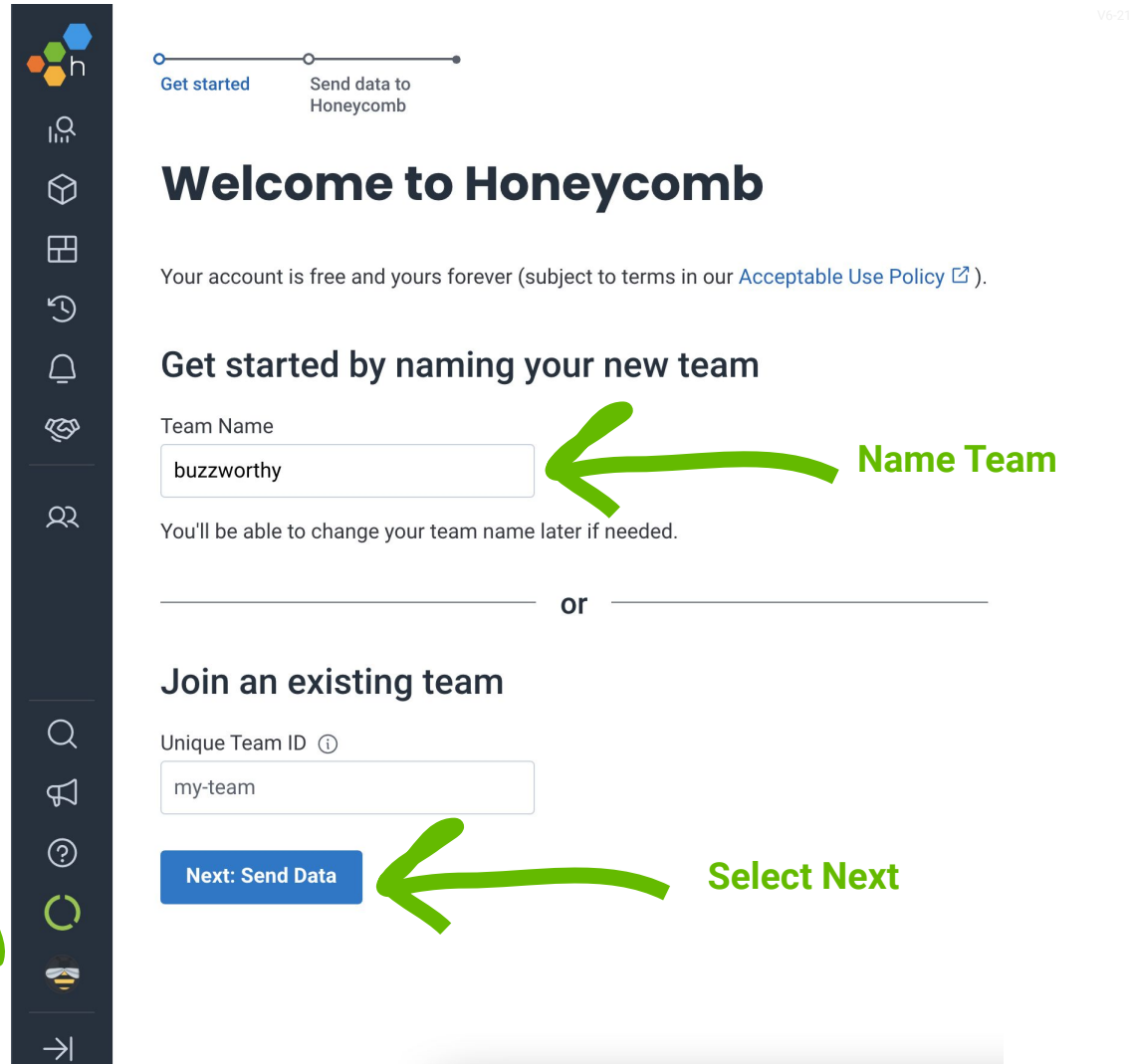


Create a team

When you first sign in, you can join a team or create a team.

Create a new team!

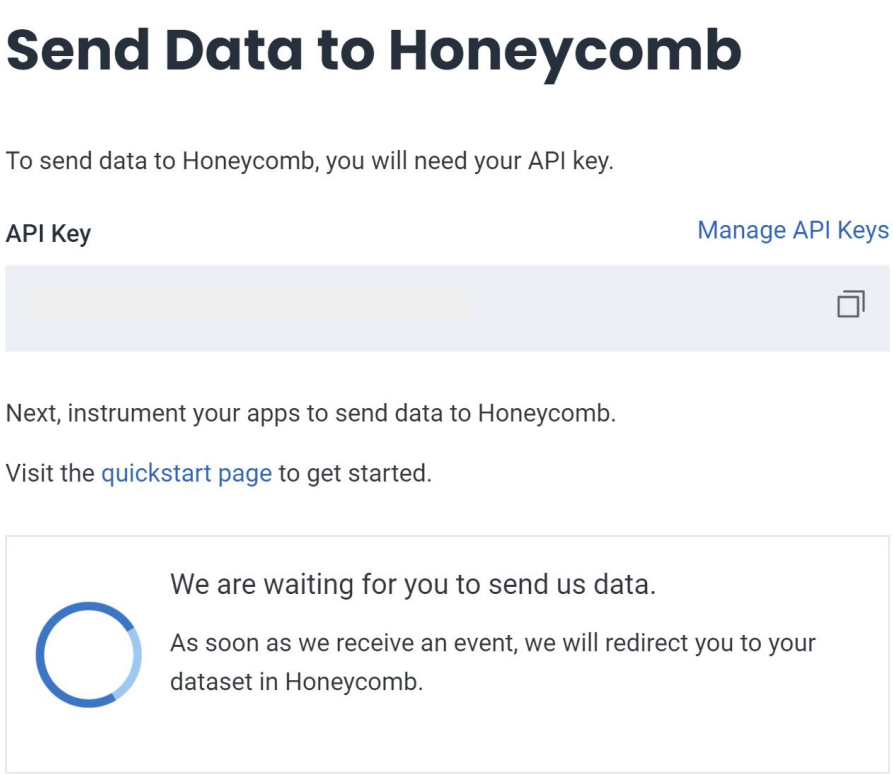
Note to Existing Users:
Select your profile picture in the lower left corner, go to **Team settings**, and find “Create New Team"



Get a Honeycomb API Key

Get started — Send data to Honeycomb

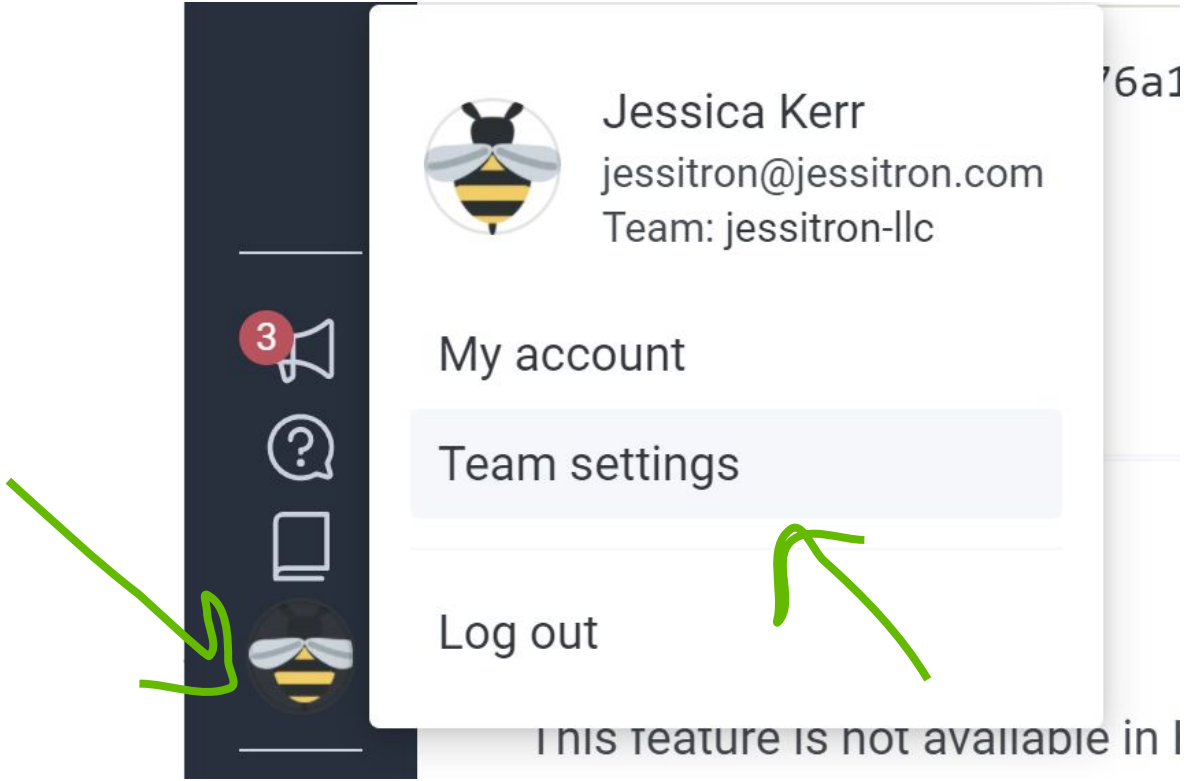
The first time you create a team as a new user, it takes you directly to your API Key.



Get a Honeycomb API Key

If you already have an account:

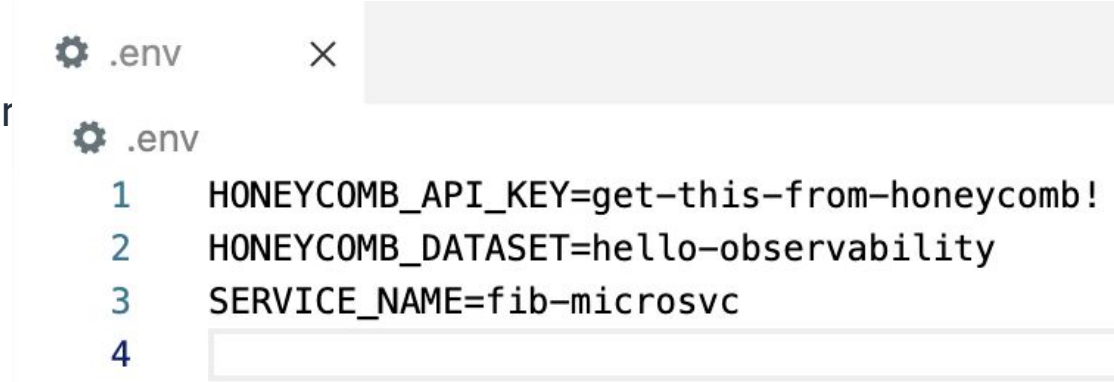
1. Select your profile picture in the lower left corner
2. Choose **Team settings**



Add credentials to send OTLP to Honeycomb

Provide your credential variables!

- `HONEYCOMB_API_KEY` identifies your team - **REQUIRED**
  - Enter your API key from the Honeycomb UI
- `HONEYCOMB_DATASET` groups your data
  - Name your dataset here
  - Example names: “Production” or “Staging”
  - Honeycomb will automatically create your dataset once data flows.



Note: `SERVICE_NAME` is optional, may not exist depending on language.

Re-run Your App

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

2) Save your `.env` file.

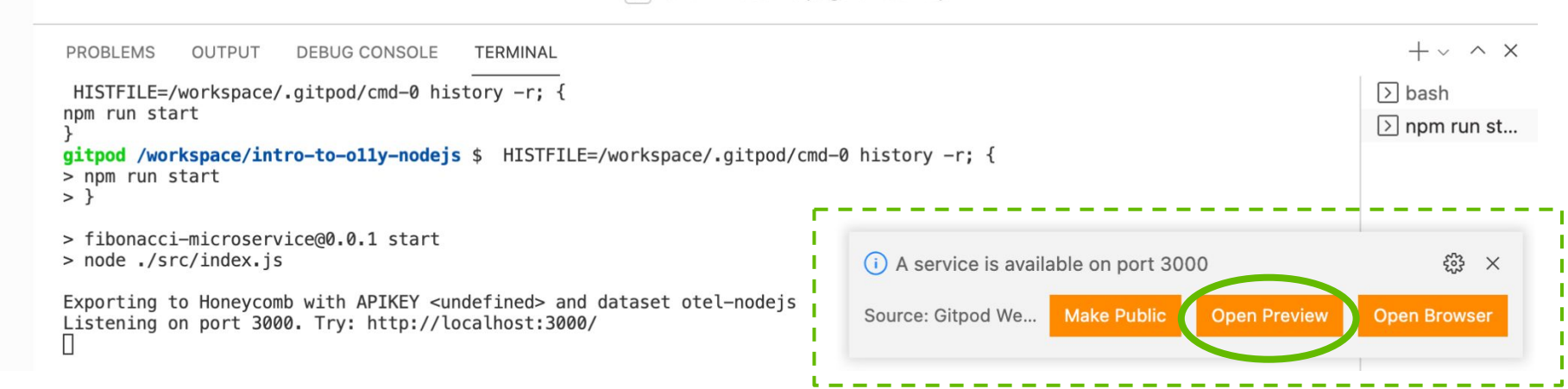
3) In the terminal, run the app using the cmd:

```
./run
```

Then, use the Gitpod app again.

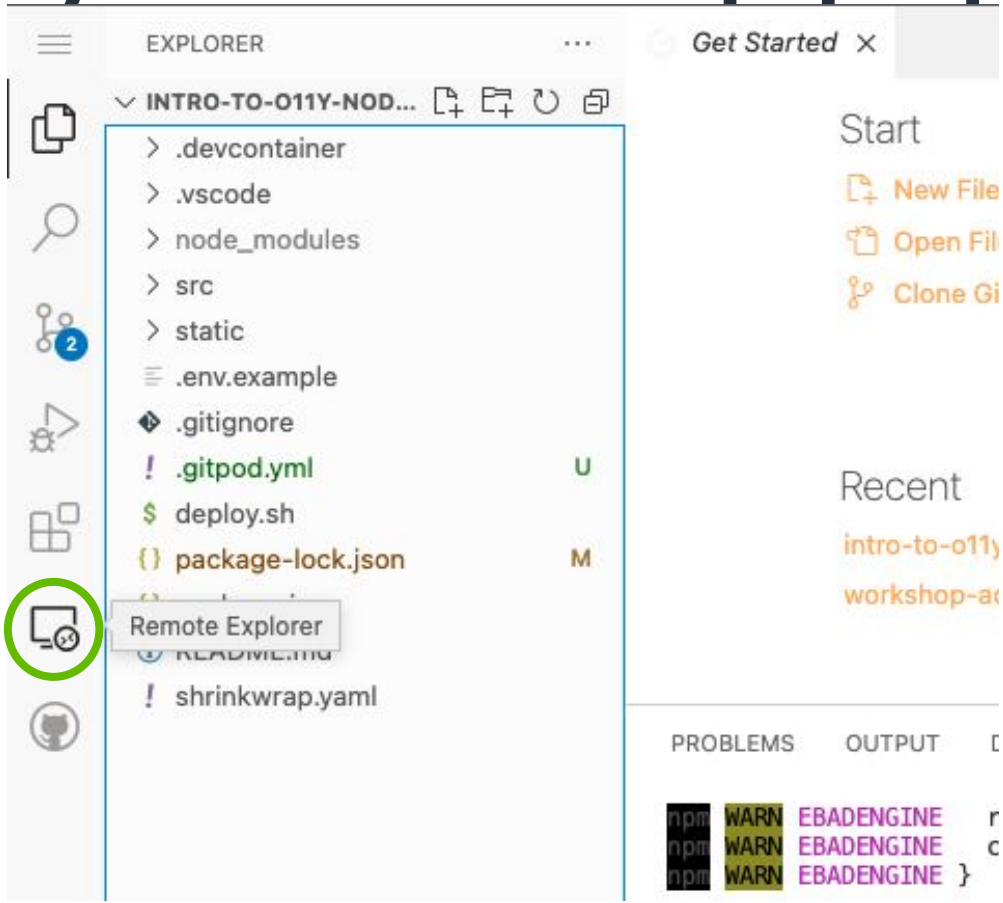
How to Open an App Preview

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**.

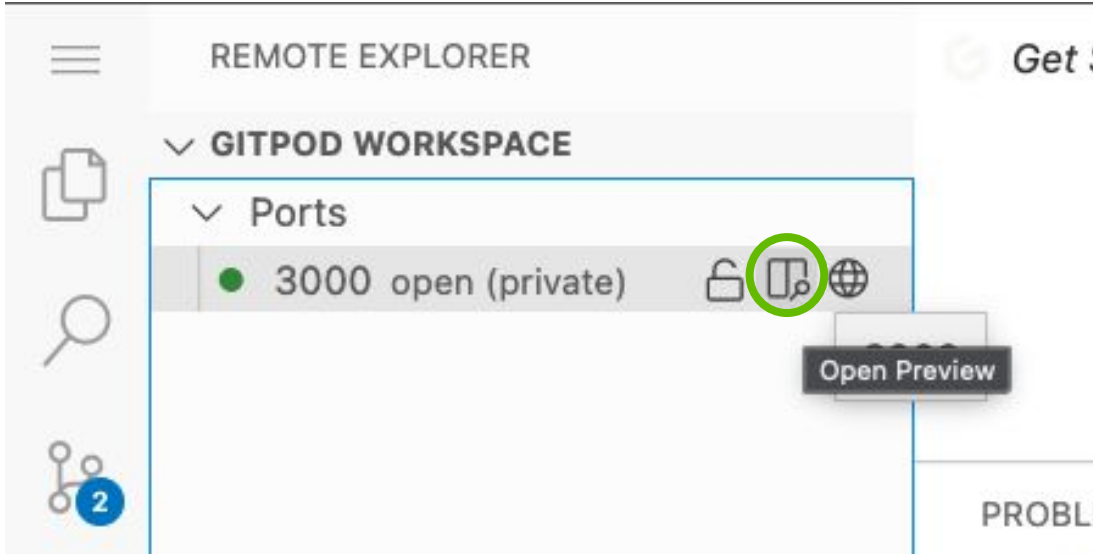


Troubleshooting: If you don't see the pop-up

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.

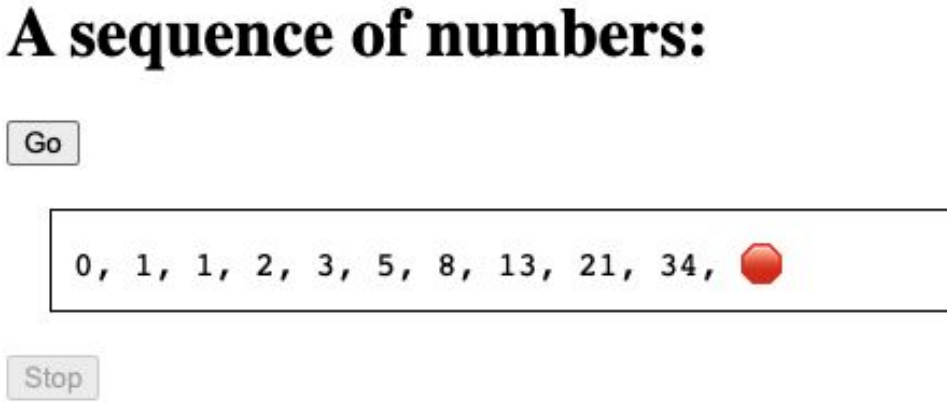


Use the Gitpod App again

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!

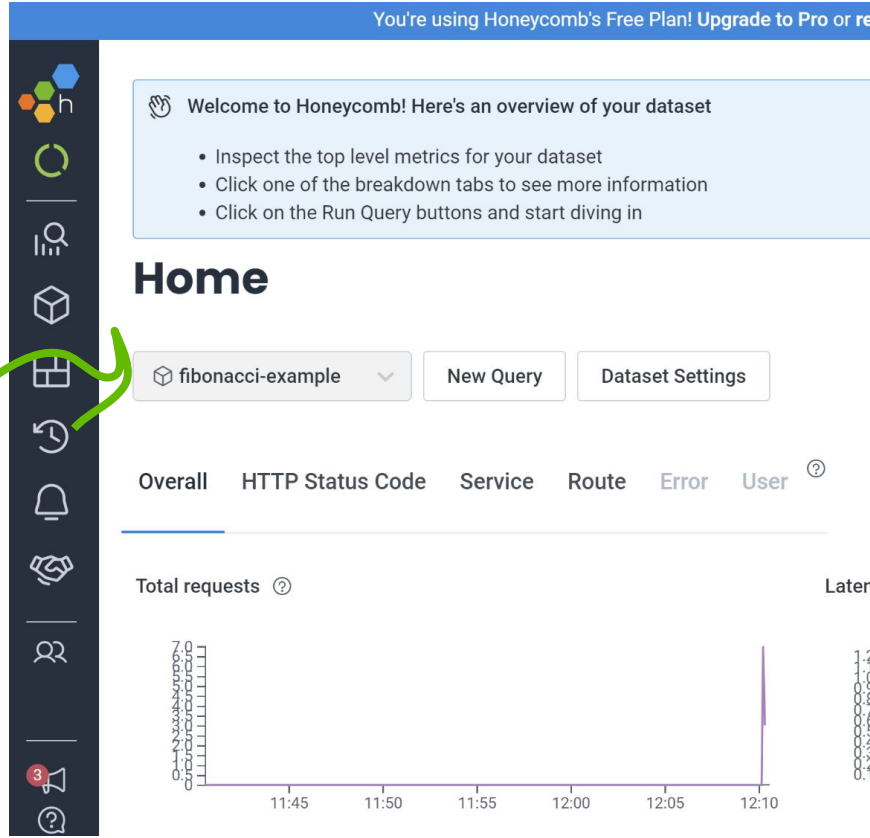
In Honeycomb, click the Honeycomb logo at the top left to go to your Home screen.

See a graph appear.

If you have used Honeycomb before, you may need to select your new dataset by name.

If you are having trouble, share your glitch app link with a TA via Slack.

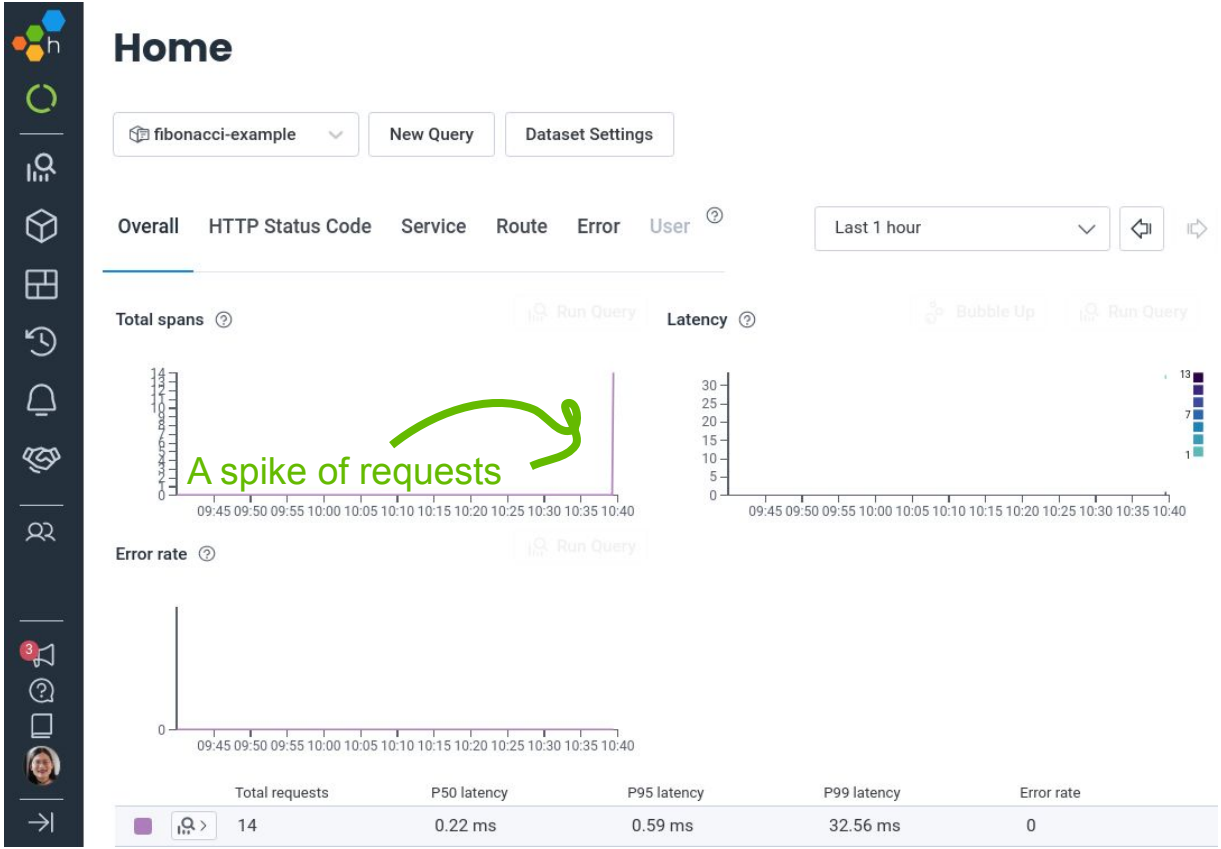
If all else fails, ask a TA for access to our "opentelemetry-workshop" team.



If you got this screen, you can go take a break!

See you at
the top of hour

Ask questions in
break-out if you
get stuck.



Summary

In this section, we:

- Instrumented a demo app with OpenTelemetry
- Got a Honeycomb Account
- Sent telemetry from the demo app to Honeycomb
- Confirmed data ingestion from the Honeycomb home screen

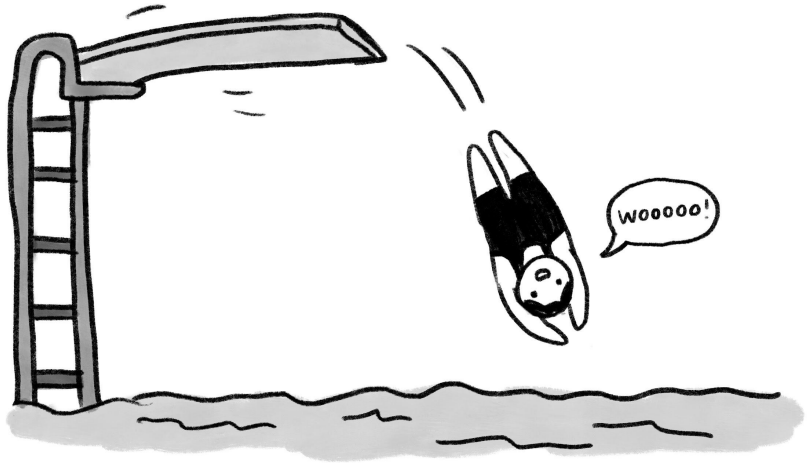
Using data to answer questions

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



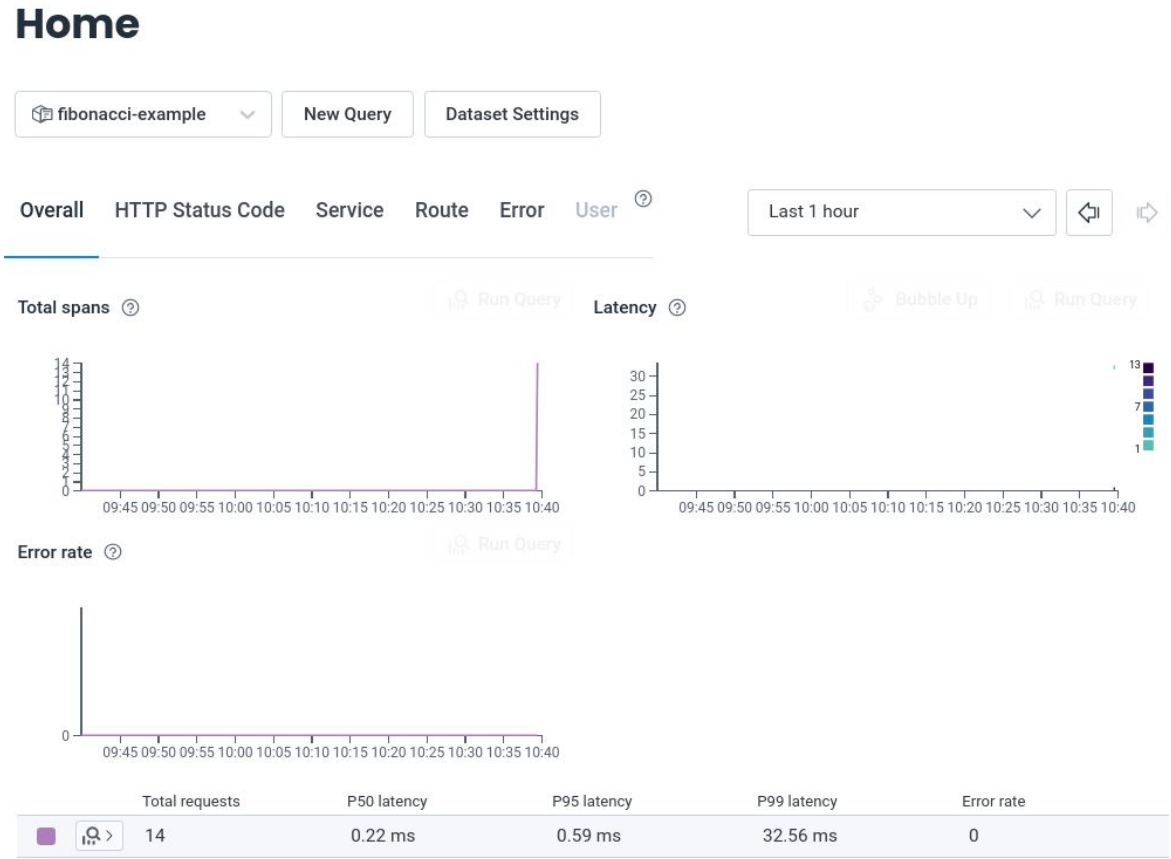
Orienting yourself in your data

Welcome "Home" to APM home!

Rate, error, duration graphs

Recent event and trace view

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

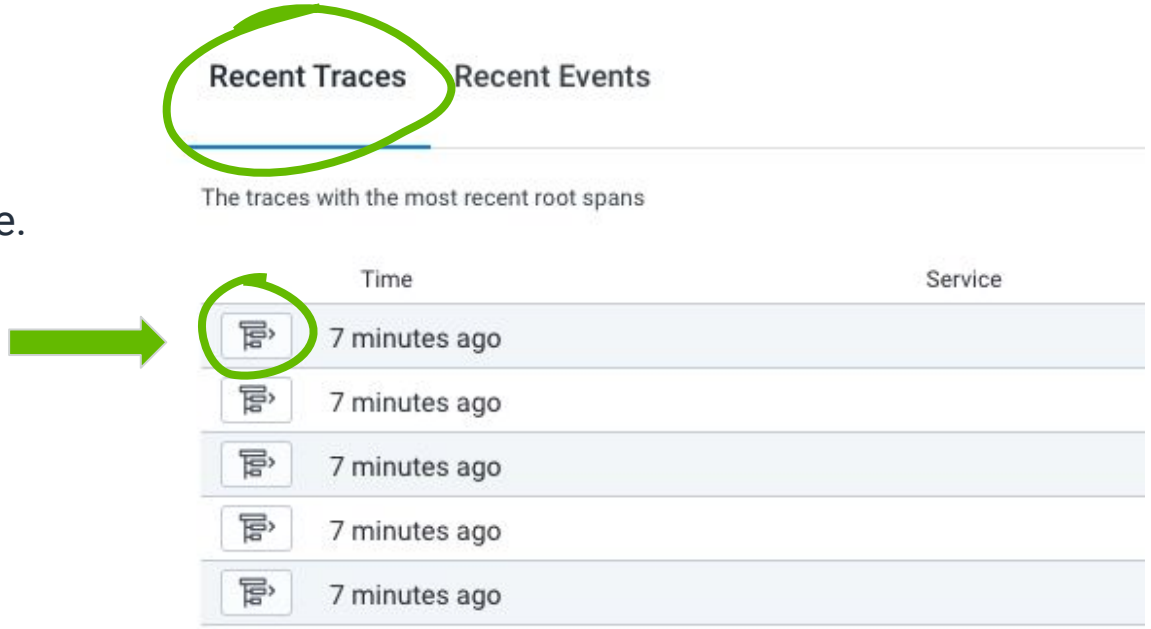


Let's generate & examine a trace!

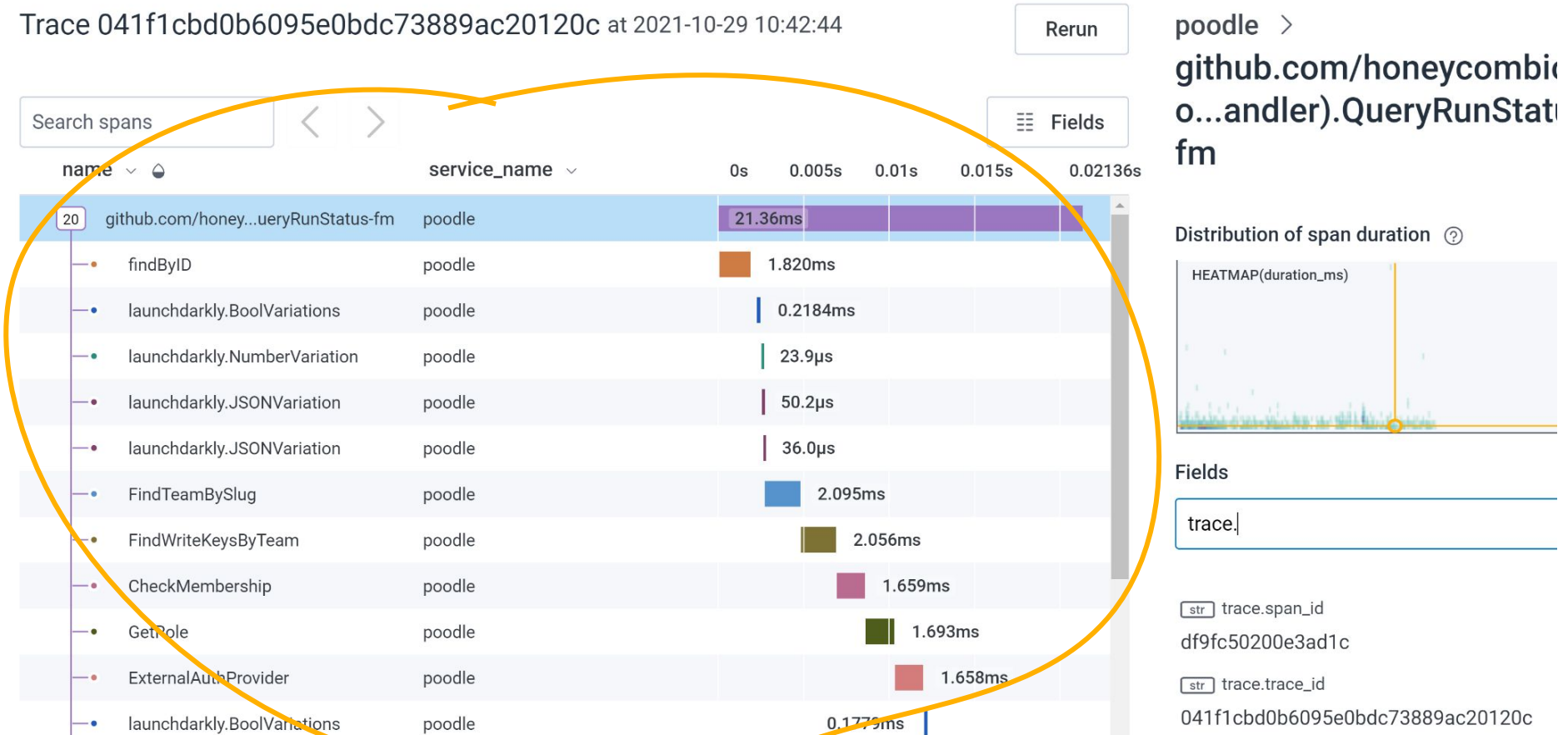
After using the app...

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

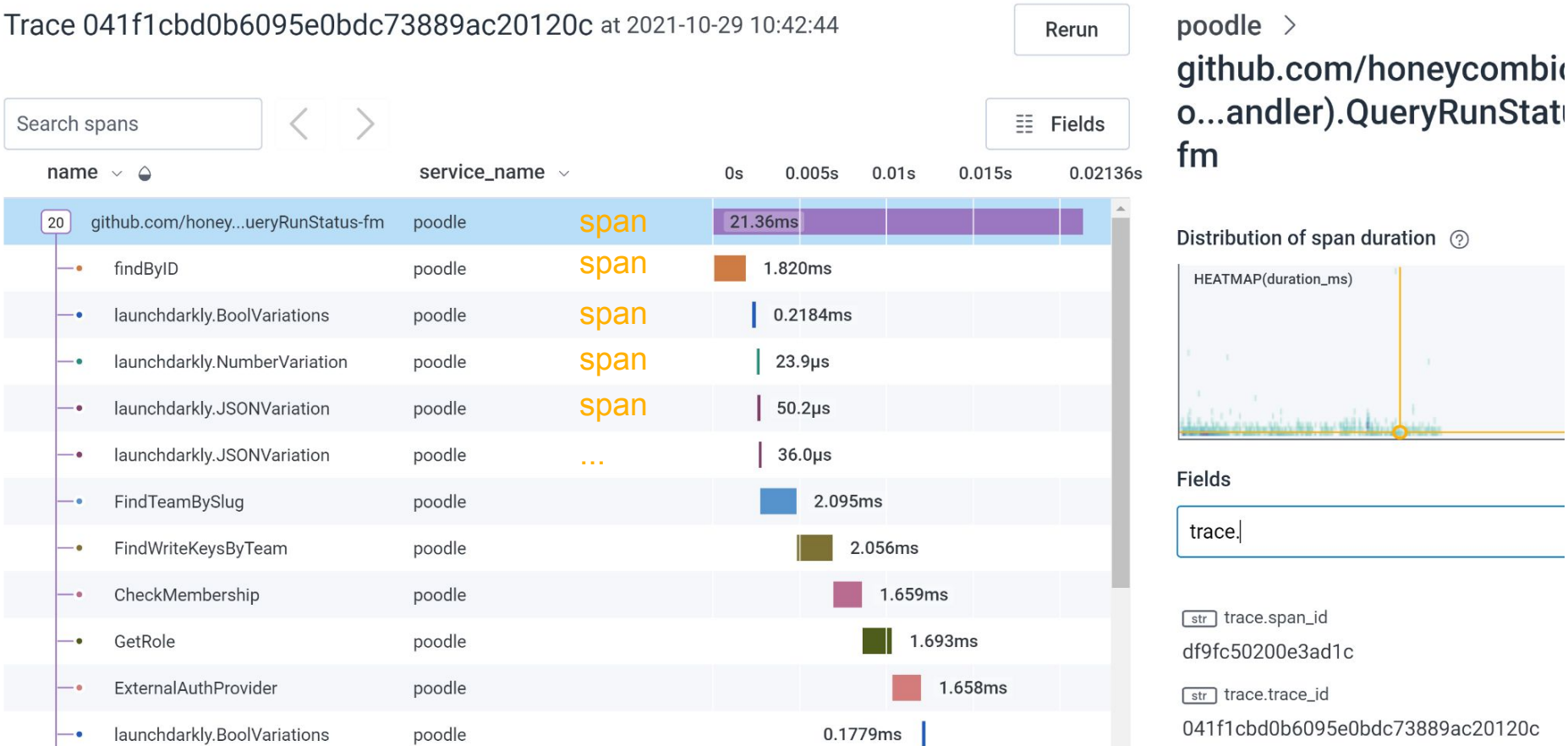
Use the **View Trace** button on the row you want to examine further.



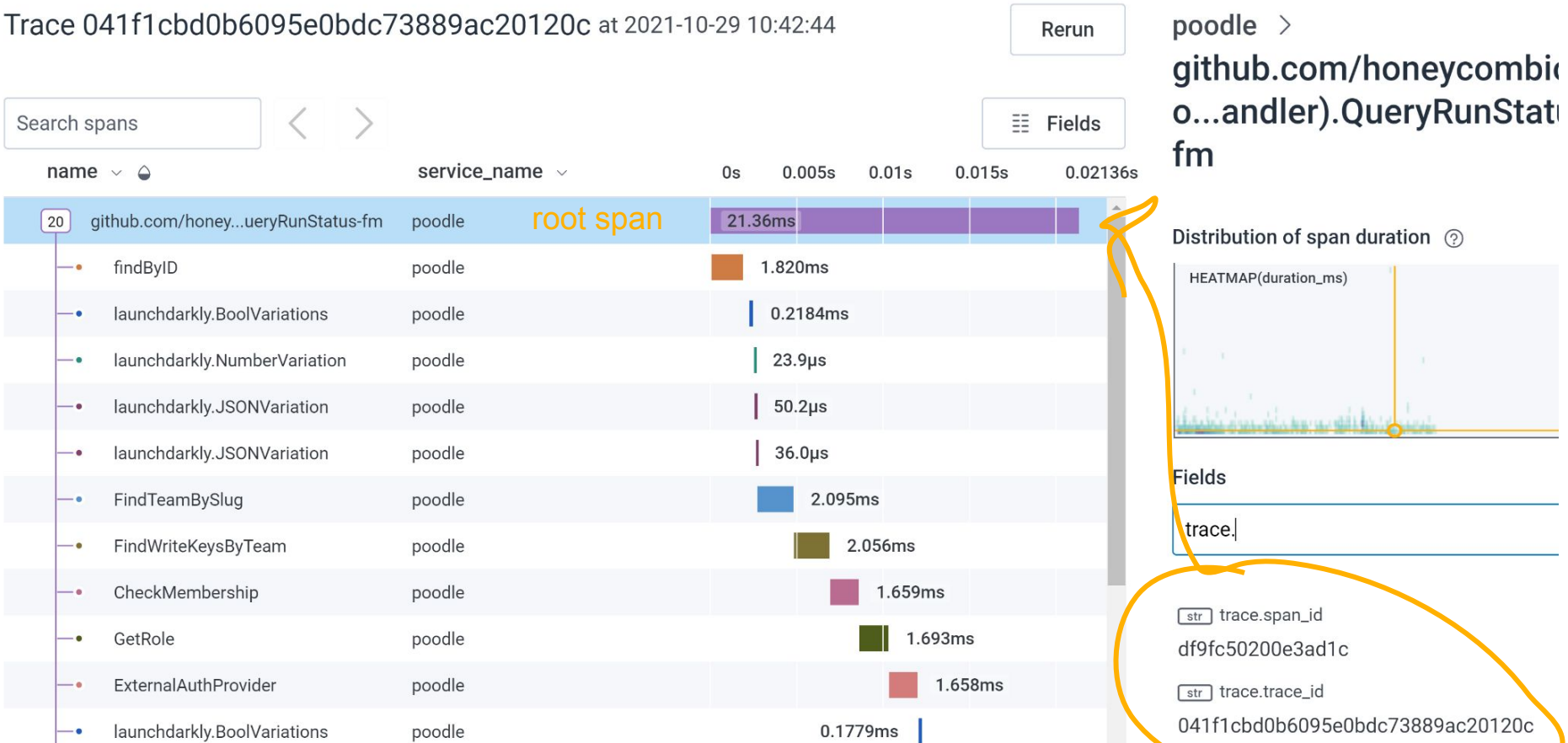
Parts of a trace



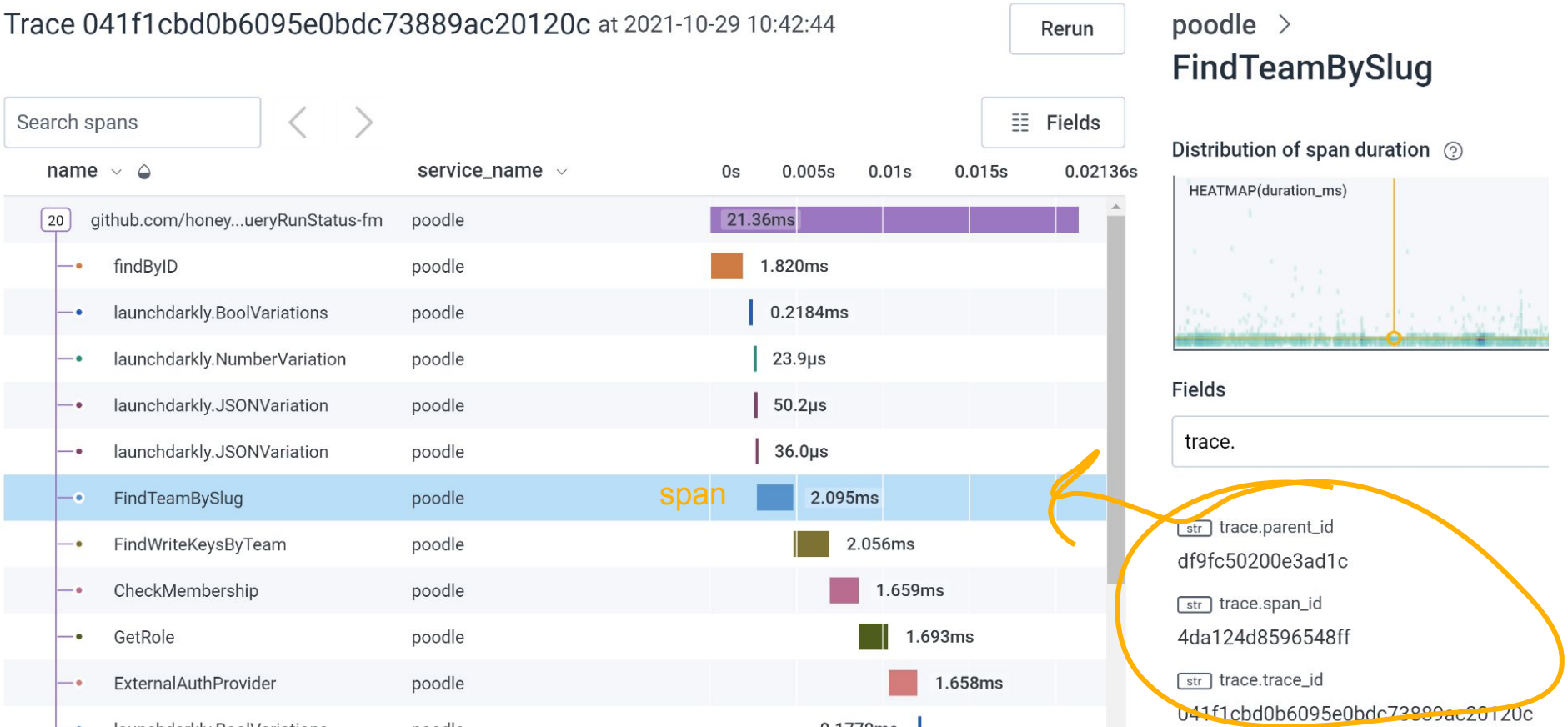
Parts of a trace



Parts of a trace



Parts of a trace

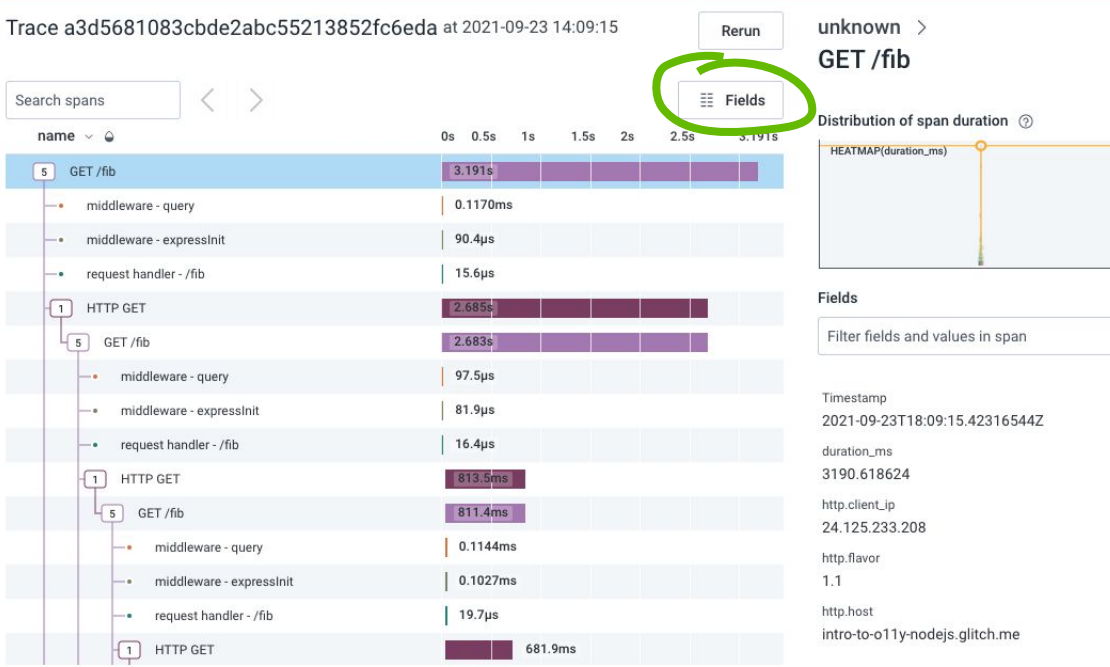


Let's generate & examine a trace!

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**

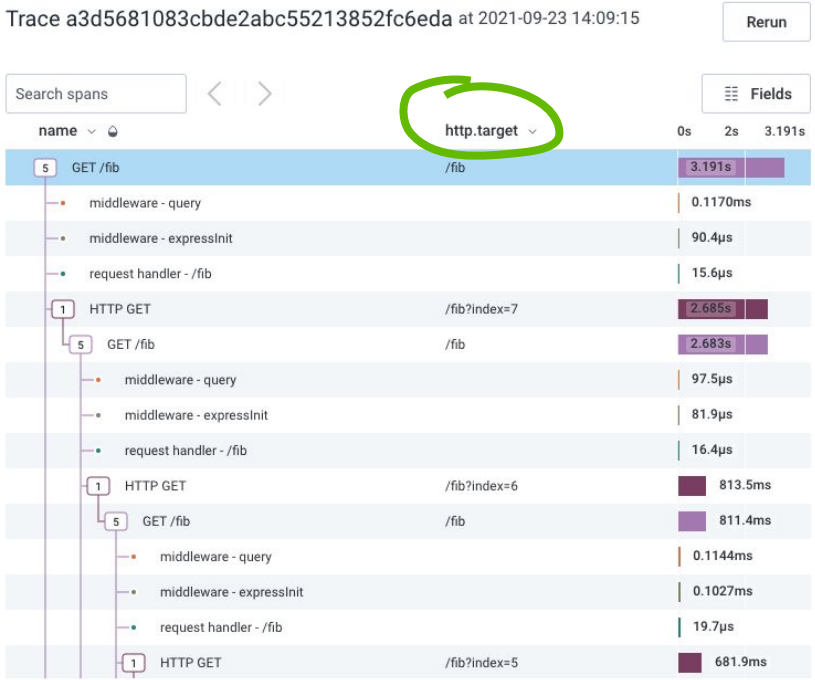


Let's generate & examine a trace!

To customize:

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

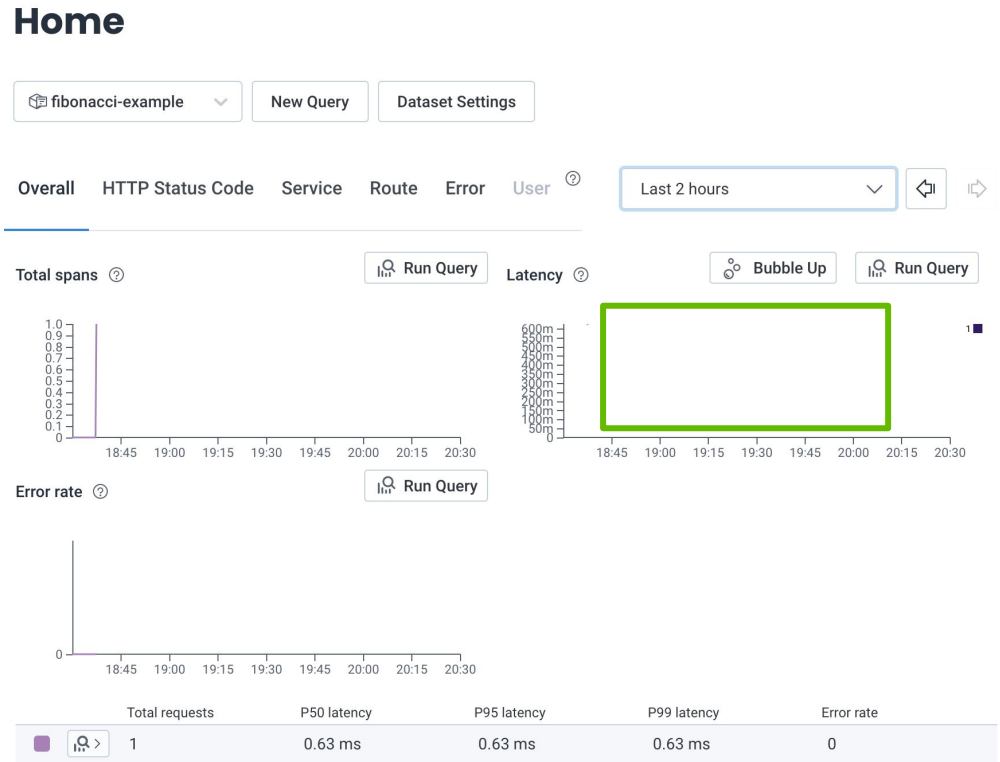
The display will update to show a **http.target** column.



Generating a little more data...

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

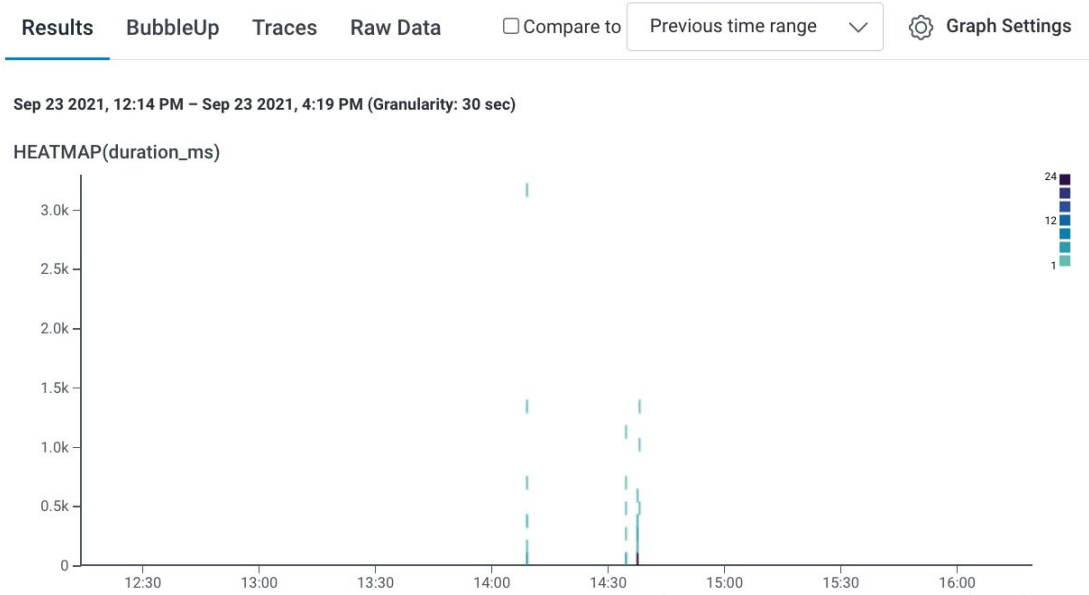
Then, click anywhere on the Latency graph from APM Home view.



Accessing heatmaps

Each area on the heatmap corresponds to a time of day, and a slowness of requests.

The histogram's color shows the density in that area: how many requests took this long, at this time.



Accessing BubbleUp

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

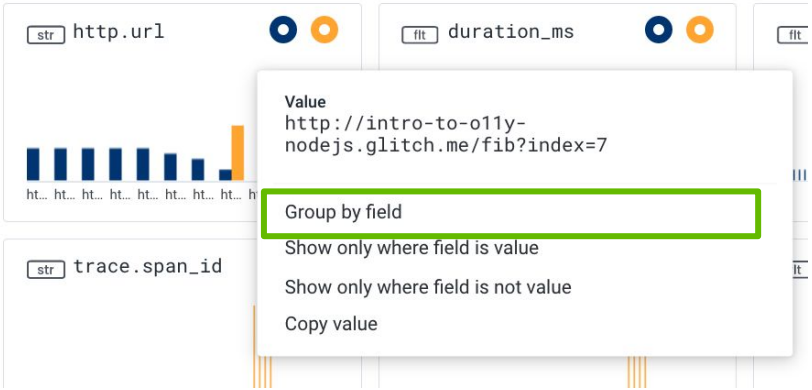
Honeycomb finds fields whose values are very different between **your selected events** and all the rest

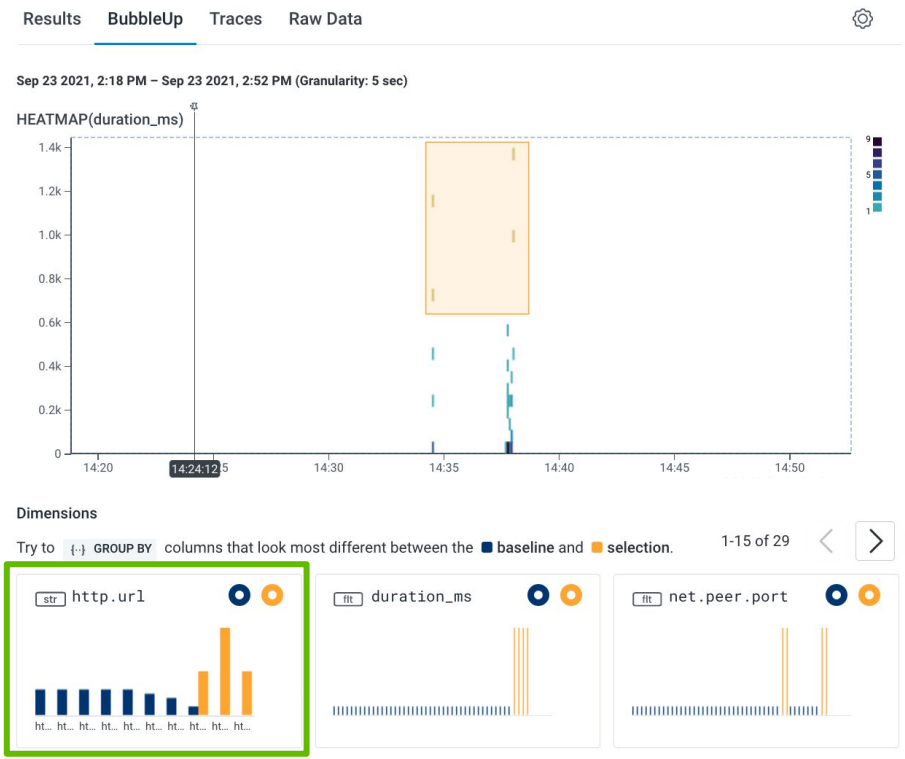
The yellow bars reflect fields and values disproportionately represented inside the area you drew.

The blue bars reflect the rest of the dataset.

Drilling down with BubbleUp

Click on your parameter (or **http.url** in this case of a NodeJS app).
Select **Group by field** in the menu.



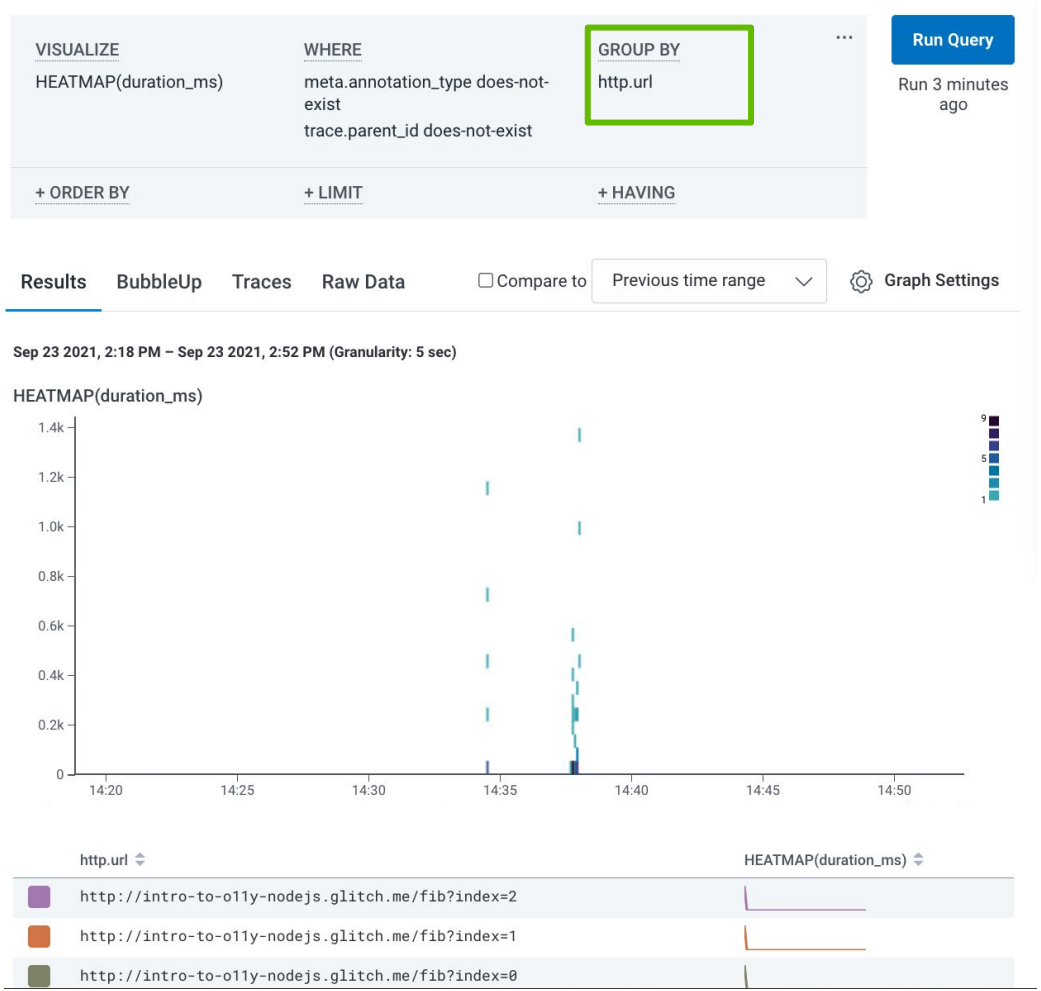


What did that do?

It doesn't look much different.

But the query has been customized…

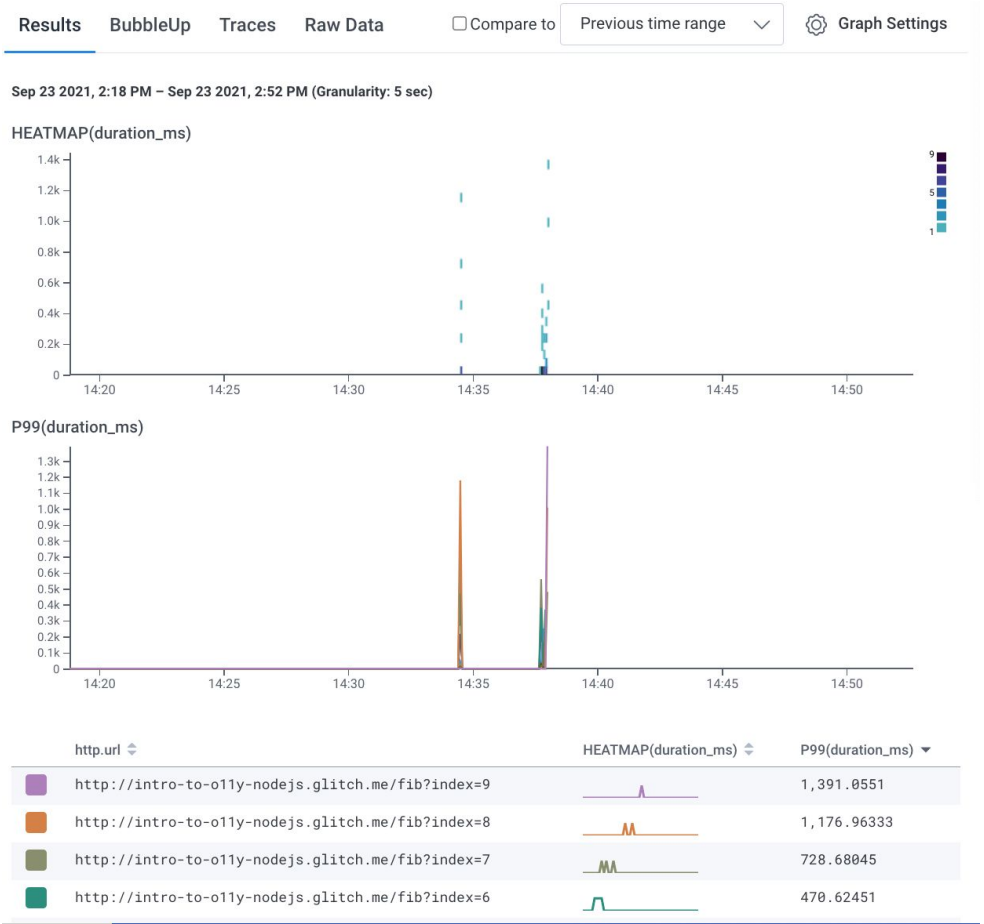
And if you select the **Results** tab, you will see the Group by display below the graph.



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.



Putting it all together: custom queries

VISUALIZE

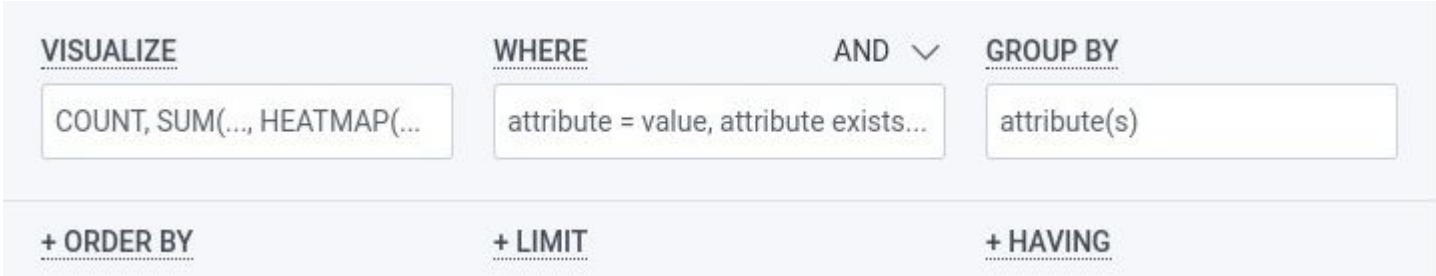
WHERE

GROUP BY

ORDER BY

LIMIT

HAVING

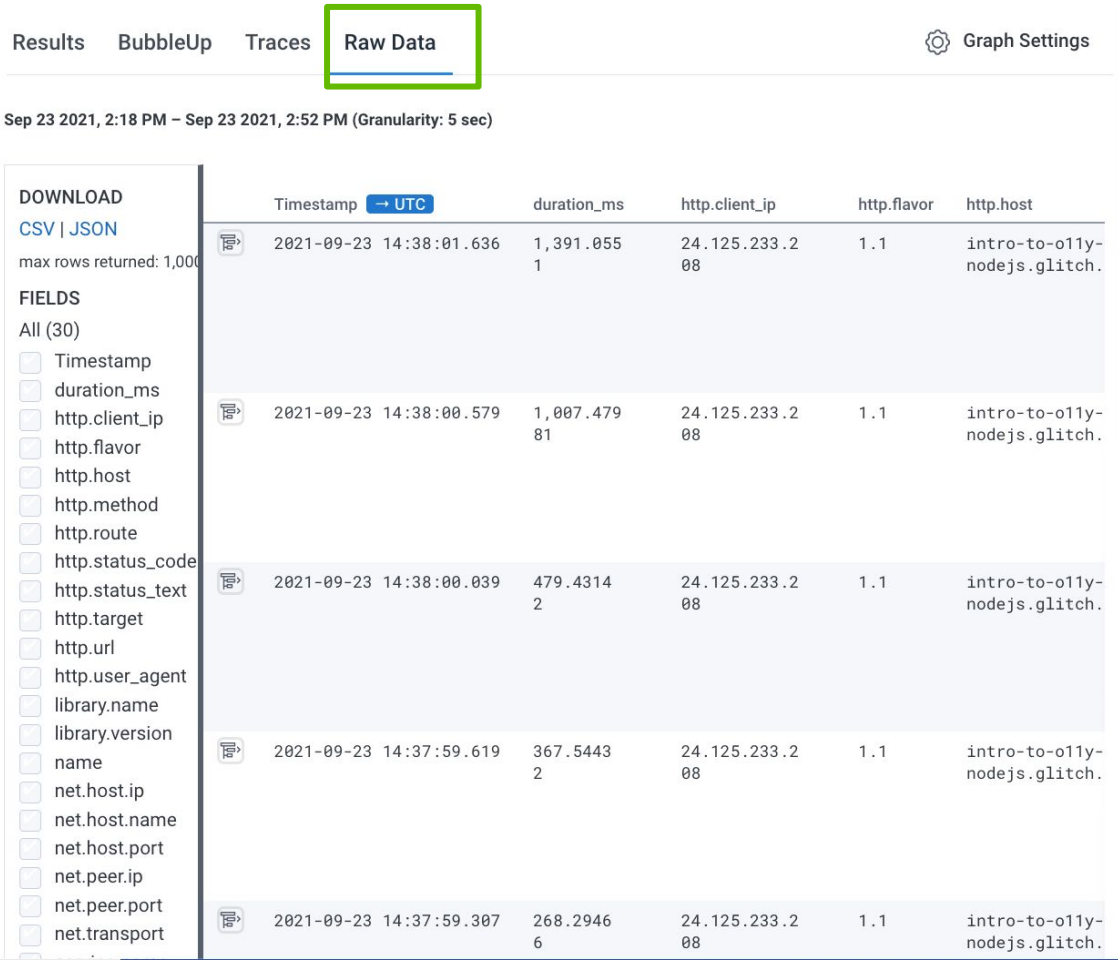


Raw data view

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

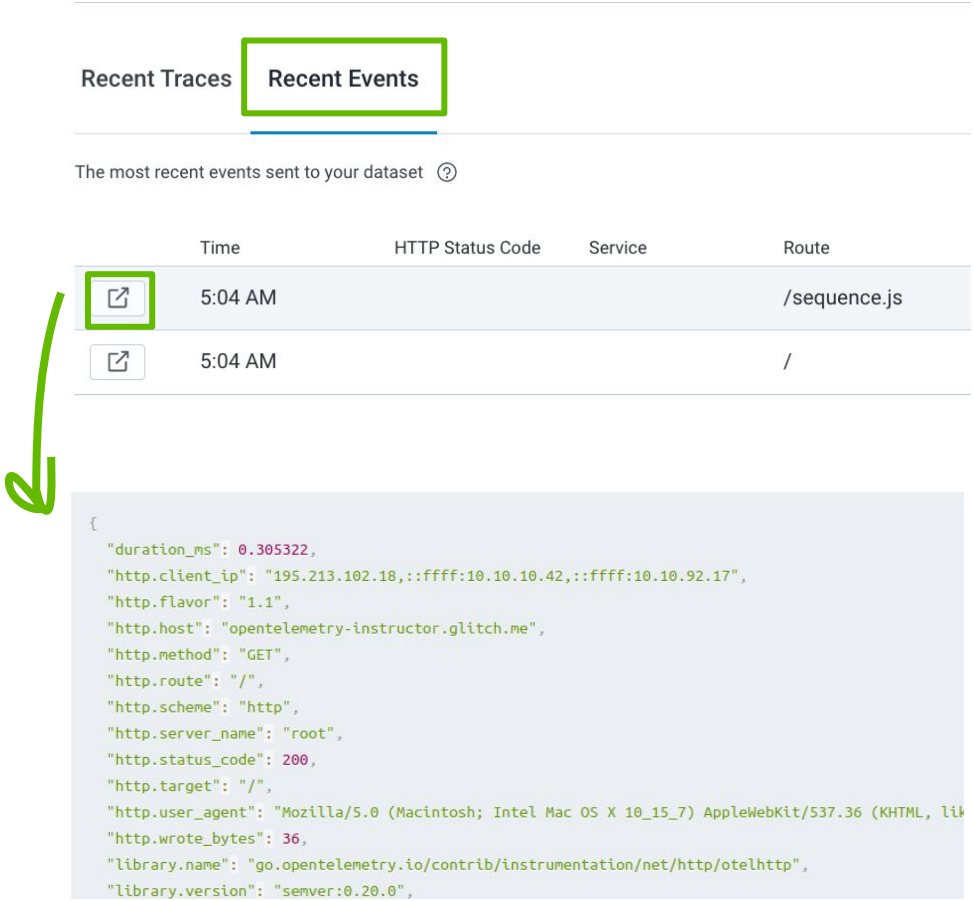
Honeycomb supports showing a table of spans and fields on those spans.

Go to the **Raw Data** tab.



Raw data view

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

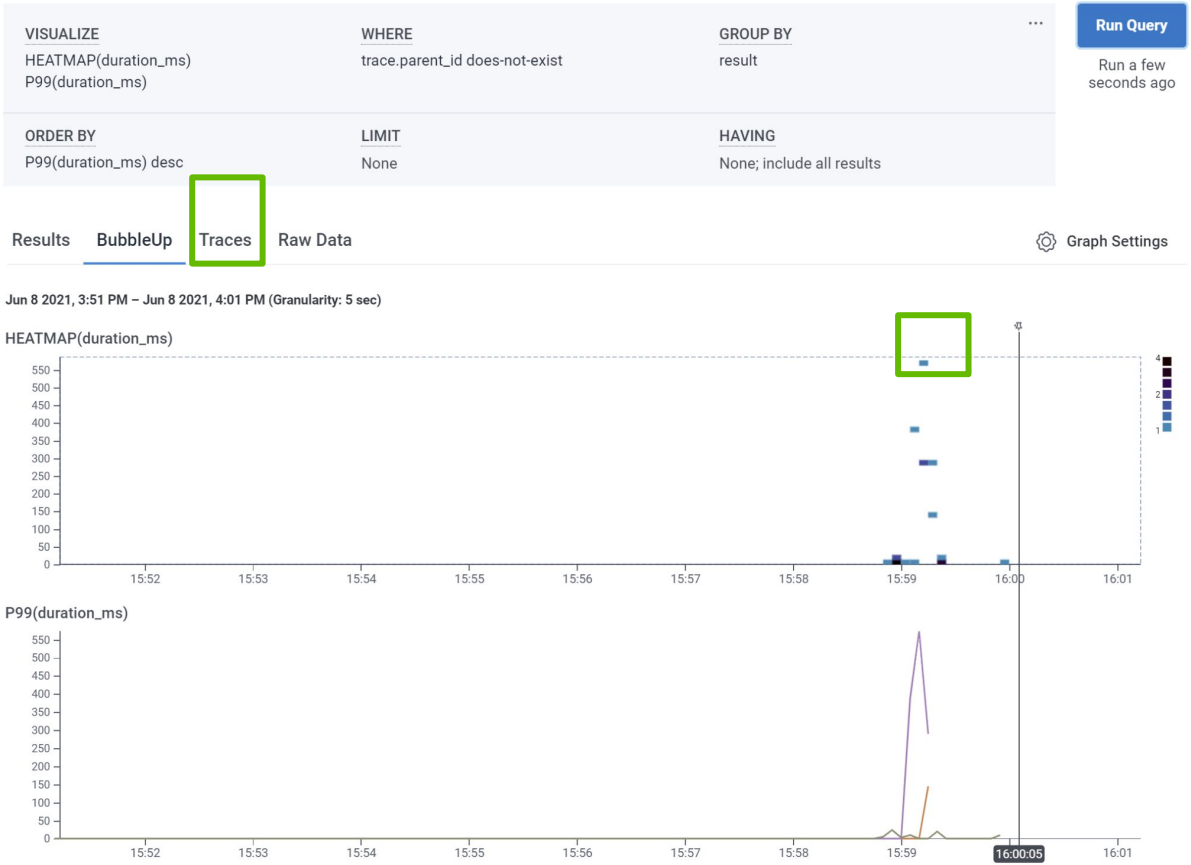


Find an especially slow trace

Go to the **Traces** tab now.

Click on the slowest trace in the heatmap.

The next screen will be your detailed Trace display.

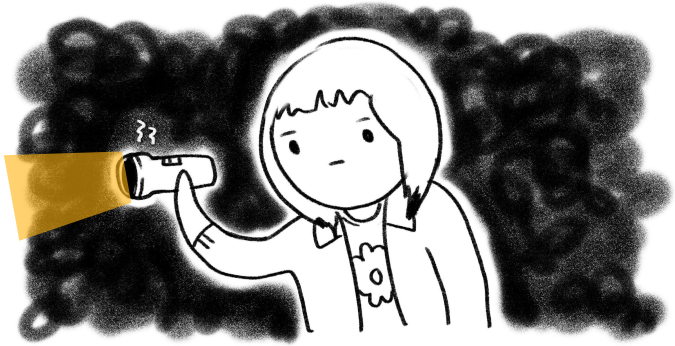


Puzzles for the class

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 i increases?

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

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

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 i increases?

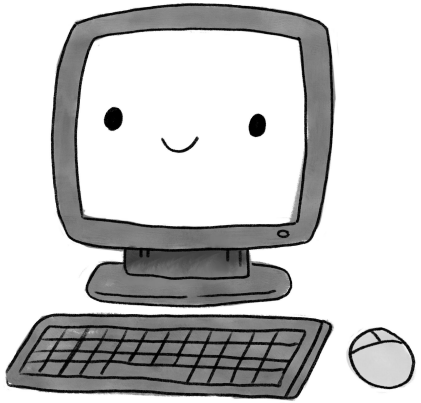
Examine a Trace in Trace Detail view.

Notice the double recursion and
no caching present...

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!



Summary

In this section, we:

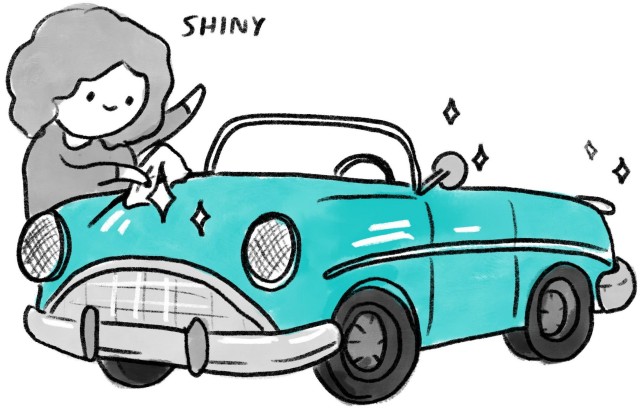
- Examined a trace for a specific request
- Identified outliers using Heatmaps and BubbleUp
- Customized our queries and used multiple visualizations
- Inspected our raw data and found the slowest traces

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.

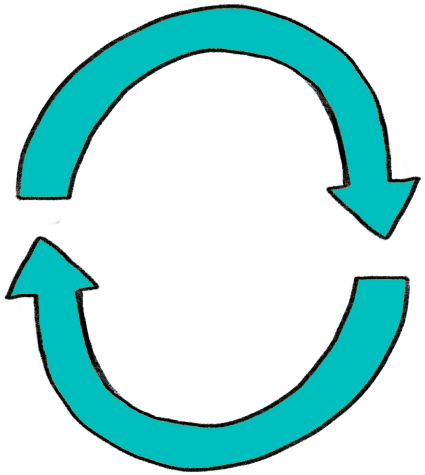
The core instrumentation loop

Start with auto-instrumentation.

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

Look at production to verify instrumentation correctness.

Repeat!

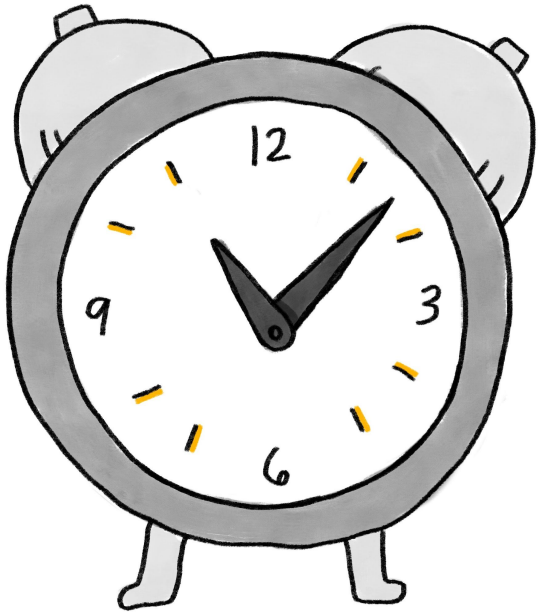


Using triggers

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.



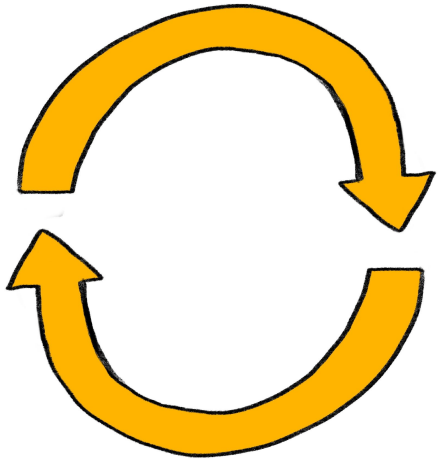
The core debugging loop

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.



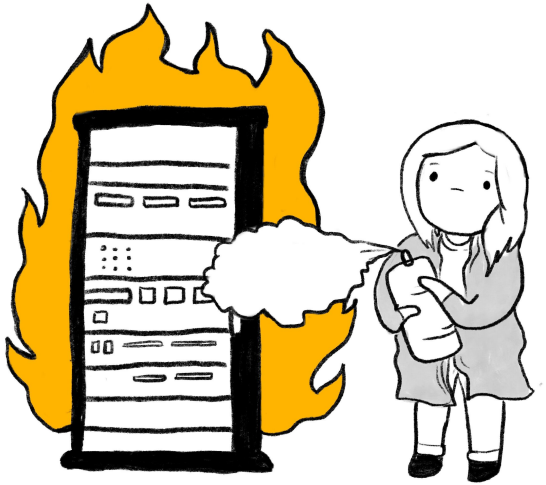
Don't panic, you have Honeycomb!

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!)



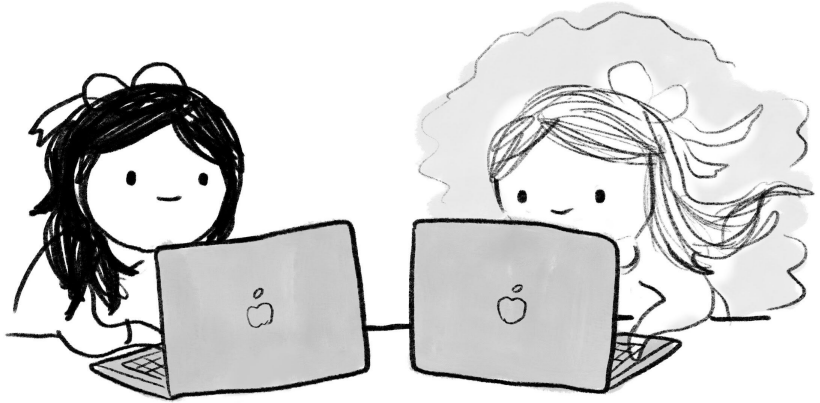
Using query history

Honeycomb is a virtual lab notebook!

Pop open query history on the right.

Pop open saved queries on the left.

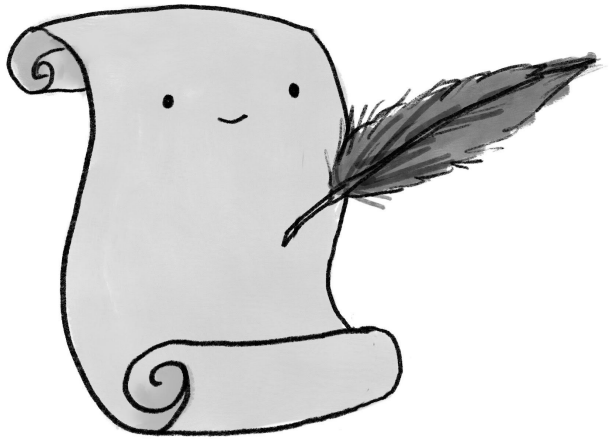
Shared with future teammates & colleagues! (and yourself, too)



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!

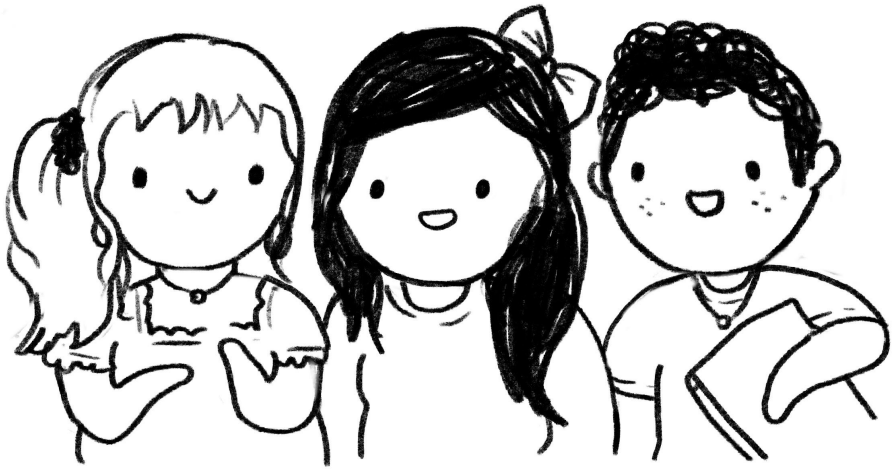
Querying reveals insights

We're here to help you!

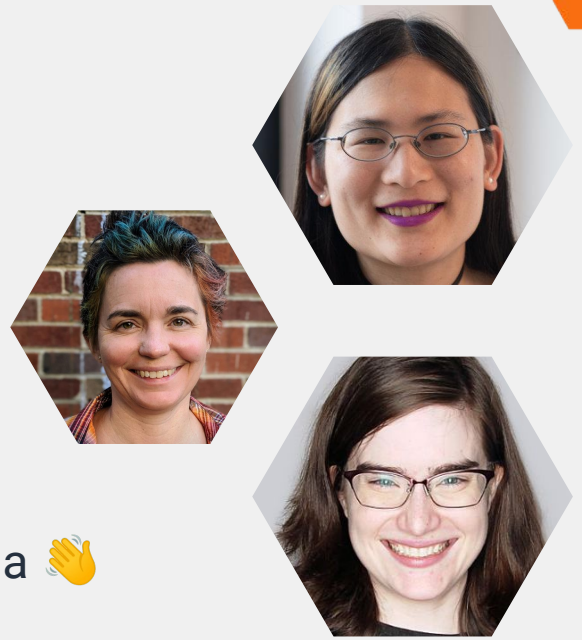
Feel free to add OTel and Honeycomb to your own apps!

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!



Thank you



We'd love to hear your feedback!

Instructors will be around Slack, give us a 👋