
expoQA®26
MADRID 26th, 27th & 28th May
expoqa.eu

Rodrigo Martín Olivera
Test Architect

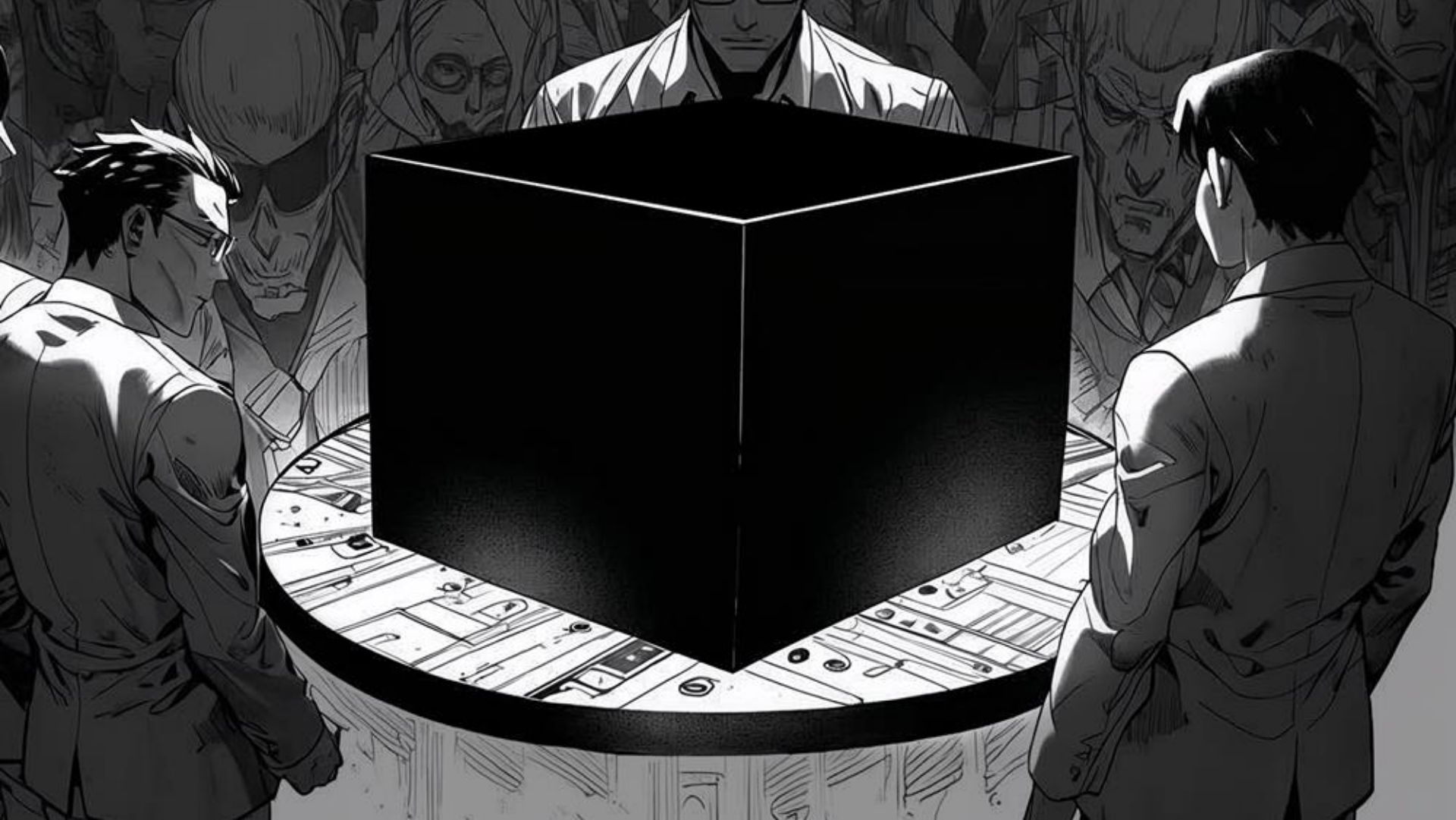
Observability in testing

Lessons from the real world

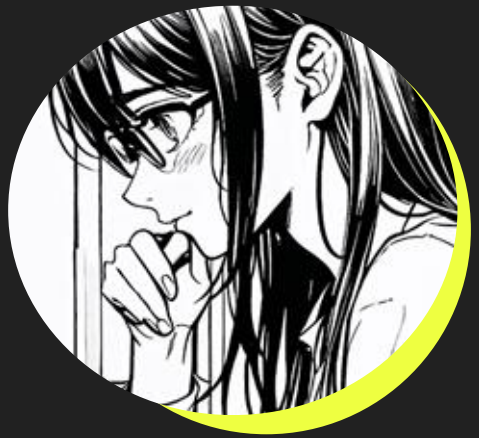
Hola 👋
I’m Rodrigo Martín Olivera.

Test Architect @ New Relic

- 🕵️ Tester since 2005
- 🎛️ Music producer
- 🤖 Sci Fi lover
- 🧉 From Argentina



Agenda



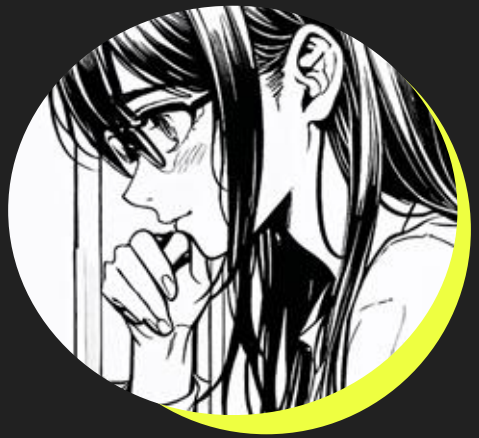
Observability in testing

What data to collect and how

Real world stories

Wrap up

Agenda



Observability in testing

What data to collect and how

Real world stories

Wrap up

Observability

What does it mean to you?

Observability is the ability to ask questions about the state of your system, by examining its outputs or external behavior.

Observability is the ability to ask questions about the state of your system, by examining its outputs or external behavior.

Observability is the ability to ask questions about the state of your system, by examining its outputs or external behavior.

Observability is the ability to ask questions about the state of your system, by examining its outputs or external behavior.

“All models are wrong,
but some are useful”

George E. P. Box

British statistician

MONITORING

Is my system working?

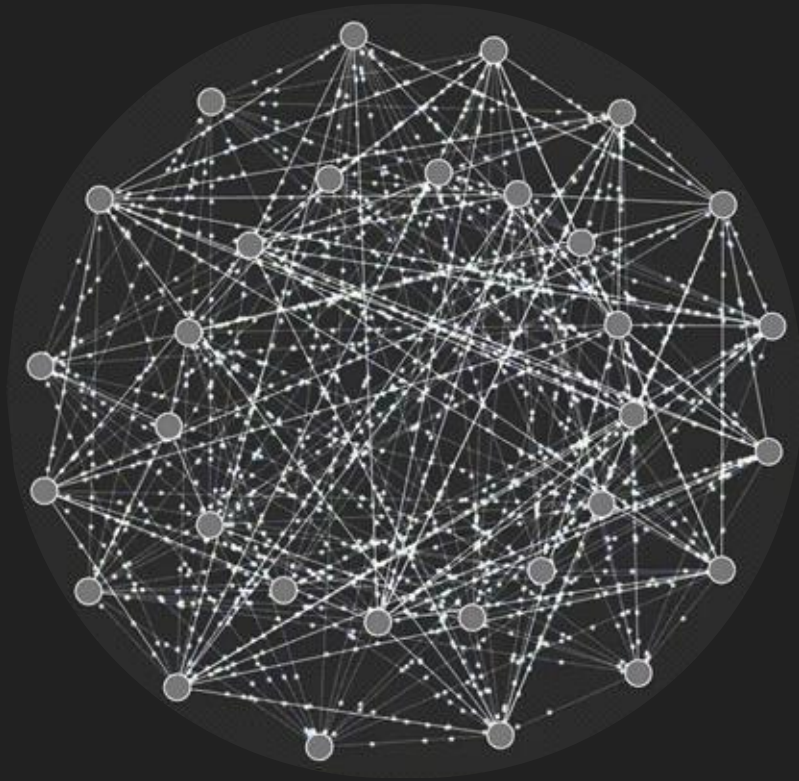
Reactive

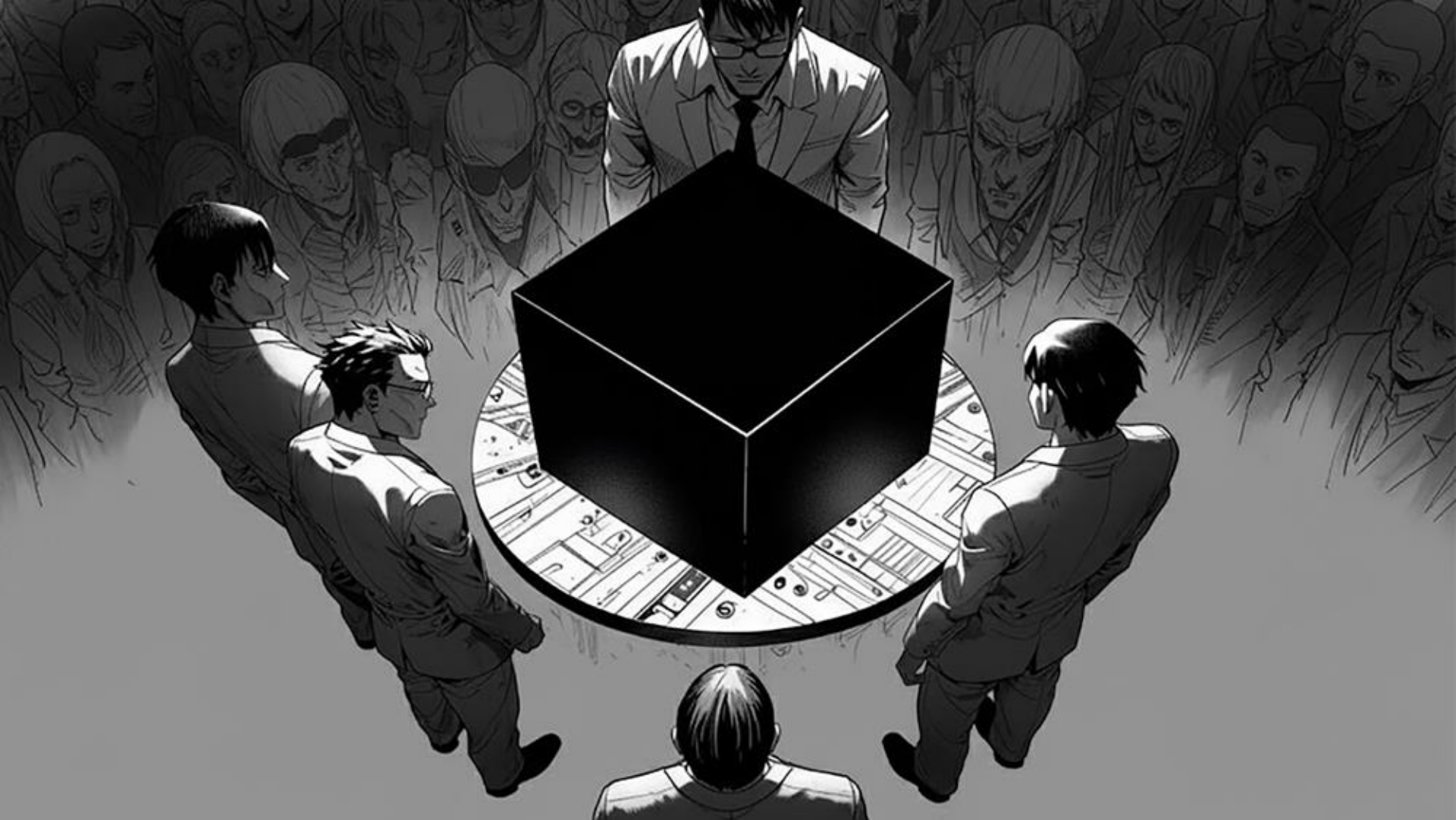
OBSERVABILITY

What is my system doing?

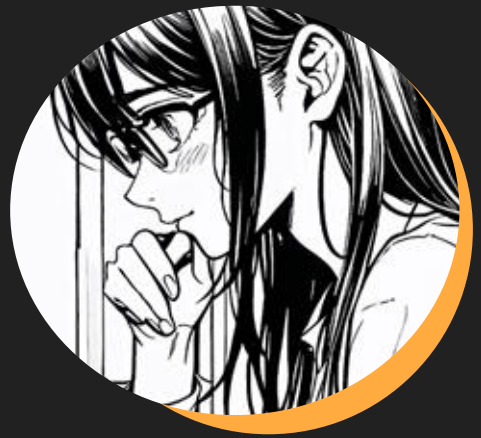
Proactive

Why observability in testing?





Agenda



Observability in testing

What data to collect and how

Real world stories

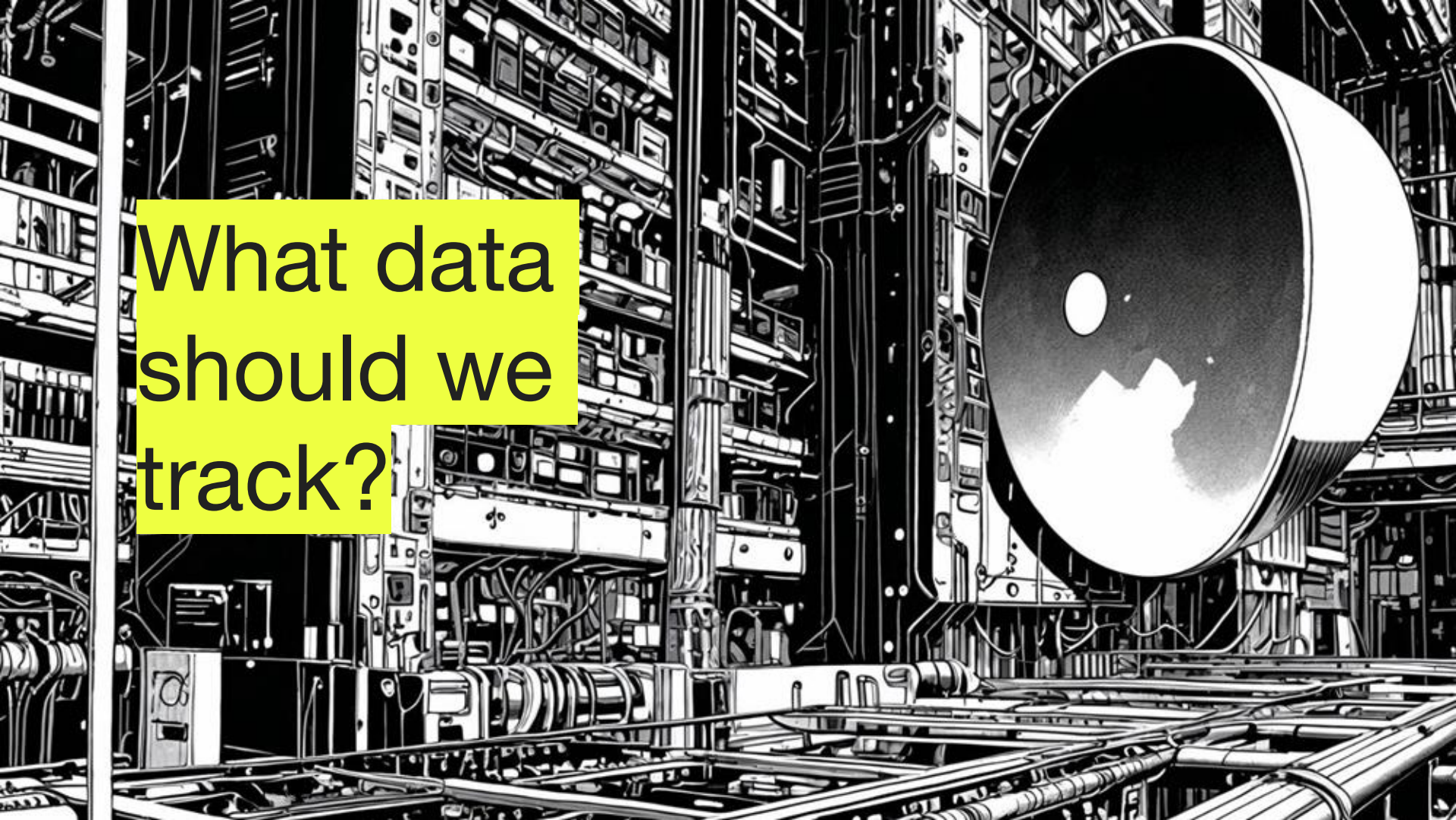
Wrap up

MELT

METRICS | **EVENTS** | **LOGS** | **TRACES**

MELT

METRICS | **EVENTS** | **LOGS** | **TRACES**


What data
should we
track?

Test execution flow

Test execution event metadata

Environment 🧪
Build Number 🛠️
Browser Name ⚙️
Base URL 🌐

Team 👶💻👦💻👦💻👱💻
Group 🏢
Annotations 🏷️

Result ✅ / ❌ / ⏭️
Outcome 🔴
Duration ⏳
Errors 🔥

Test execution event metadata

Environment 🧪
Build Number 🛠️
Browser Name ⚙️
Base URL 🌐

CONTEXT

Team 👨🏻‍💻 👨🏽‍💻 👨🏿‍💻 👱‍♂️💻
Group 🏢
Annotations 🏷️

Result ✅ / ❌ / ⏭️
Outcome 🟠
Duration ⏳
Errors 🔥

Test execution event metadata

Environment 🧪
Build Number 🛠️
Browser Name ⚙️
Base URL 🌐

Team 👦💻👦💻👦💻👱💻
Group 🏢
Annotations 🏷️

OWNER

Result ✅ / ❌ / ⏭️
Outcome 🟠
Duration ⏳
Errors 🔥

Test execution event metadata

Environment 🧪
Build Number 🛠️
Browser Name ⚙️
Base URL 🌐

Team 👦🏻💻👦🏽💻👦🏿💻👱🏼💻
Group 🏢
Annotations 🏷️

Result ✅ / ❌ / ⏭️
Outcome 🟠
Duration ⏳
Errors 🔥

FACTS

Code coverage flow

Code coverage event metadata

Coverage 📊
Covered ✅
Missed ❌

Release Tag 📦
Repository 📂
Team
👨🏻‍💻 👨🏽‍💻 👨🏿‍💻 👨🏼‍💻
Group 🏢

Code coverage event metadata

Coverage 📊
Covered ✅
Missed ❌

FACTS

Release Tag 📦
Repository 📂
Team
👨🏻‍💻 👨🏽‍💻 👨🏿‍💻 👱🏻‍💻
Group 🏢

Code coverage event metadata

Coverage 📊
Covered ✅
Missed ❌

Release Tag 📦
Repository 📂
Team
👨🏻‍💻 👨🏽‍💻 👨🏿‍💻 👱🏻‍💻
Group 🏢

CONTEXT


How to track this data?

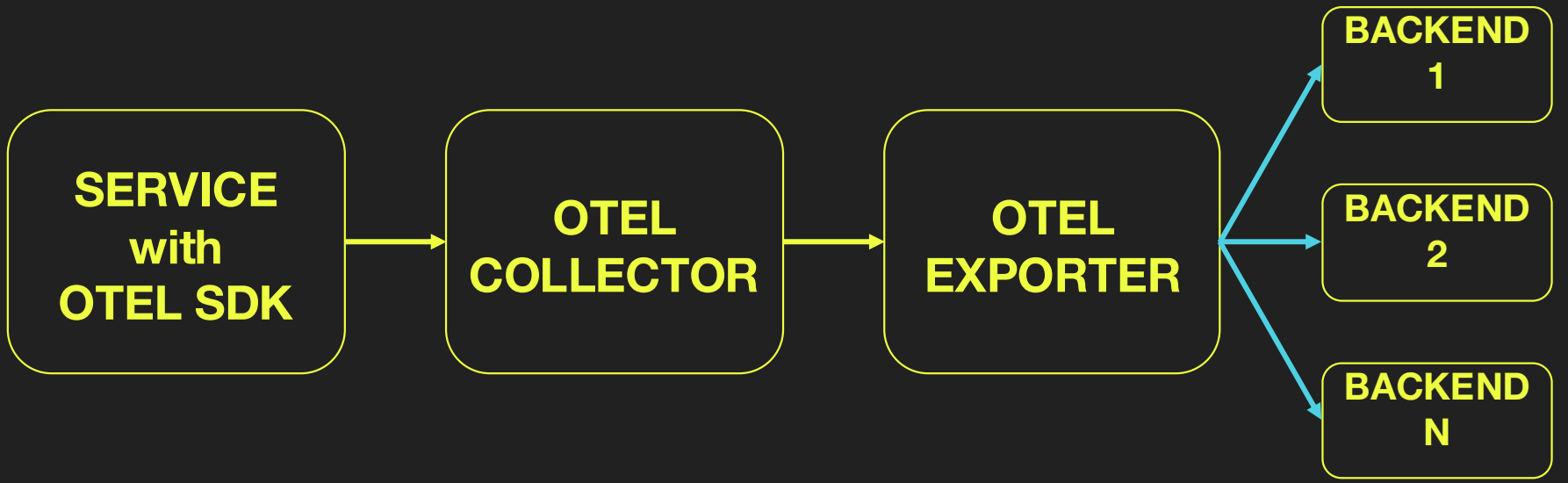
Two paths to observability

Vendor API Integration

Universal Data Collection

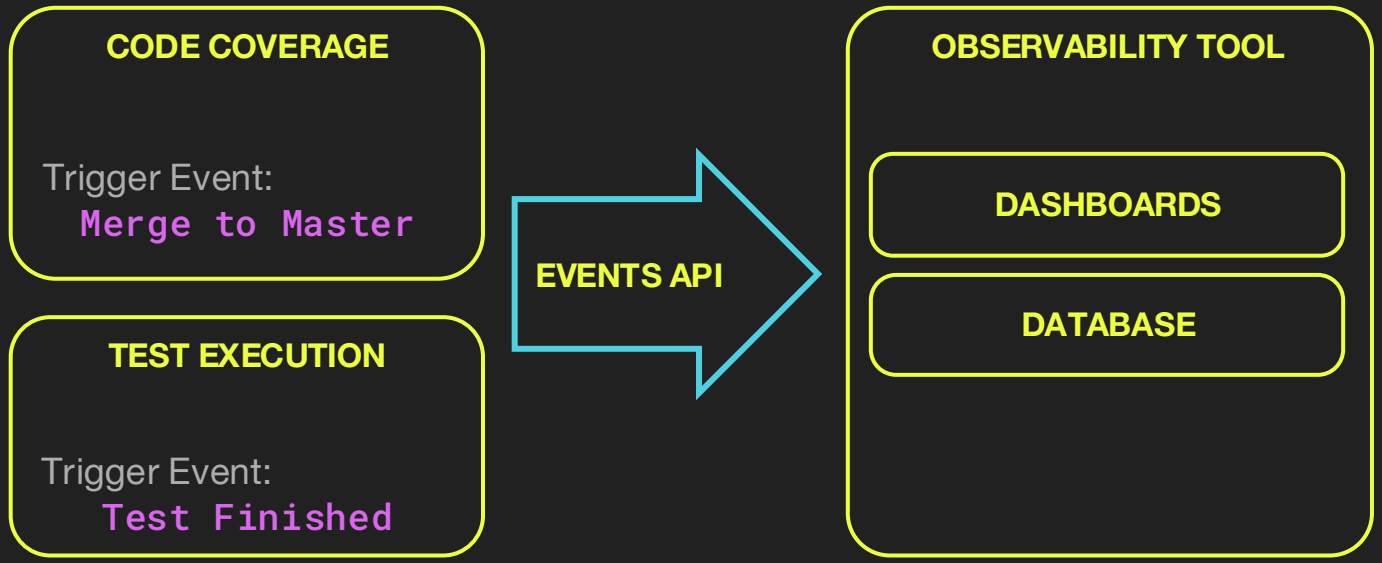
Vendor API integration

OpenTelemetry architecture

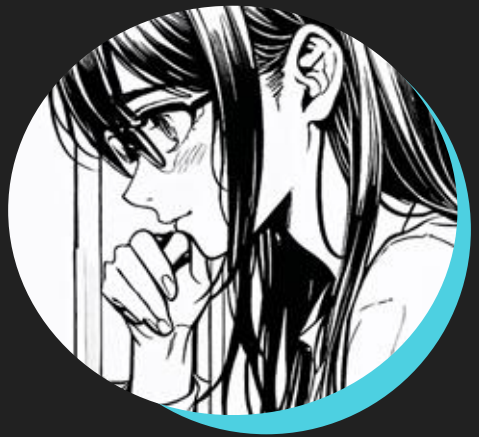


OpenTelemetry

Test Data Collection pipeline



Agenda



Observability in testing

What data to collect and how

Real world stories

Wrap up

My day to day job

- Coaching 
- Quality metrics 
- Proof of Concept 
- Test Strategy
- Reliability 
- Chaos Engineering 
- Feedback 
- Teams support 

Stories

1 Automation nightmare


FLAKY
TEST


ENGINEER


FLAKY
TEST
imgflip.com

Test execution event metadata

Environment 🧪
Build Number 🛠️
Browser Name ⚙️
Base URL 🌐

Team 👦💻 👦🏽💻 👦🏿💻 👱💻
Annotations 🏷️

Result ✅ / ❌ / ⏭️
Outcome 🟠
Duration ⏳
Errors 🔥

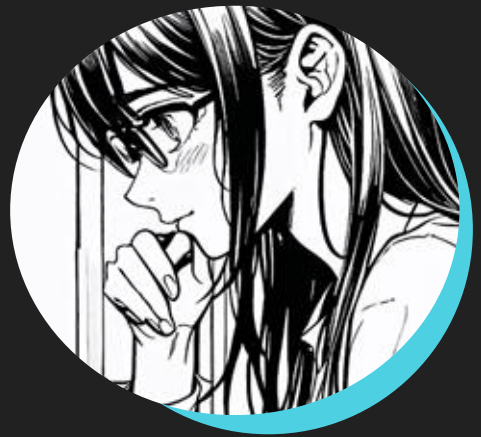
The outcome field

```
testCase.outcome():
"skipped"|"expected"|"unexpected"|"flaky"
```

The outcome field

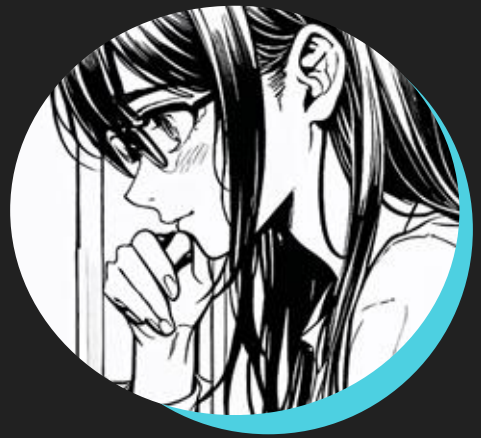
```
testCase.outcome():
"skipped"|"expected"|"unexpected"|"flaky"
```

The plan



1. Research possible root causes
2. Implement a fix
3. Measure success

The plan

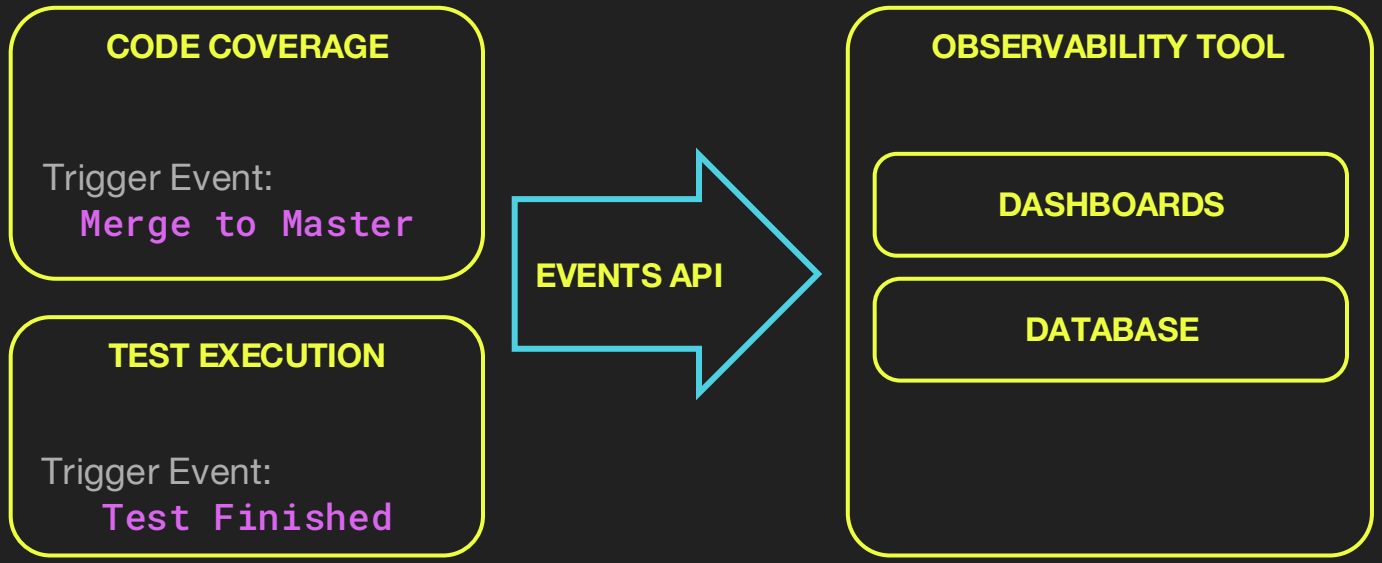


Research possible root causes

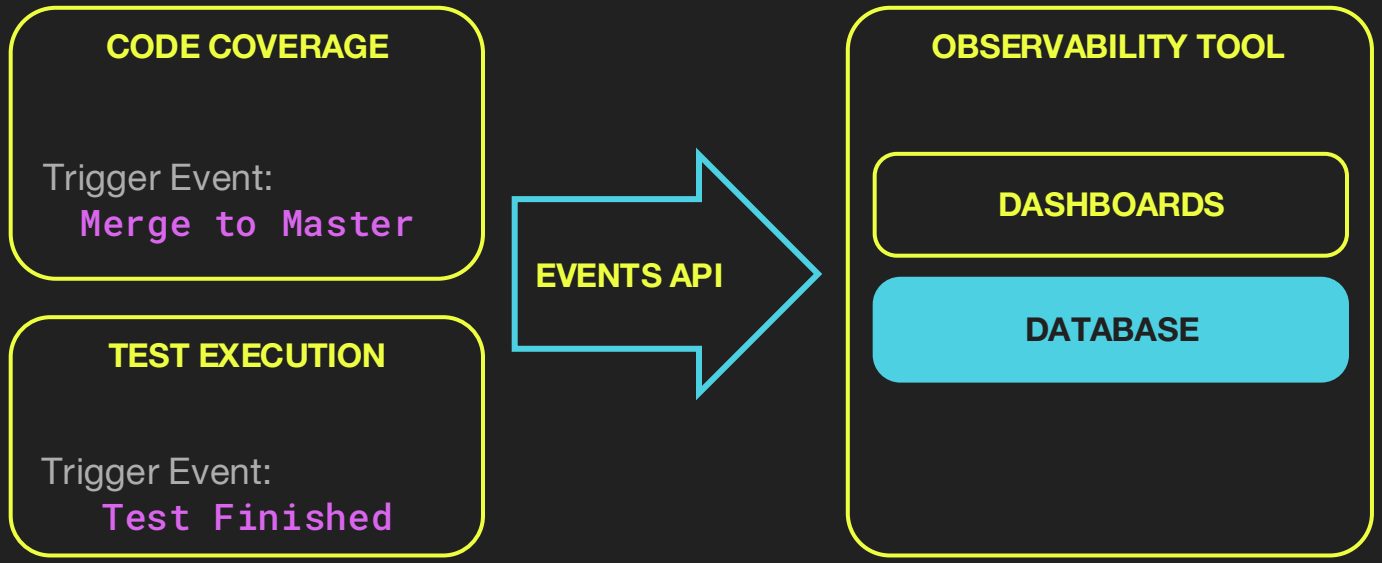
2. Implement a fix

Measure success

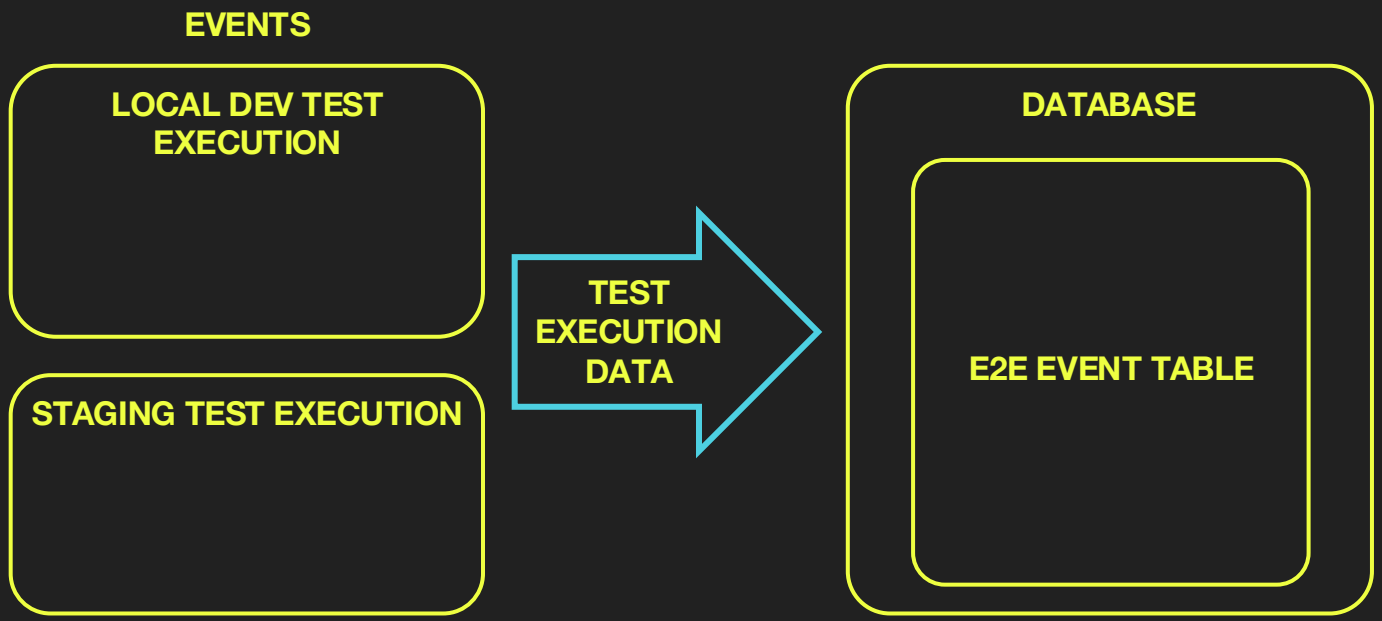
Finding the root cause



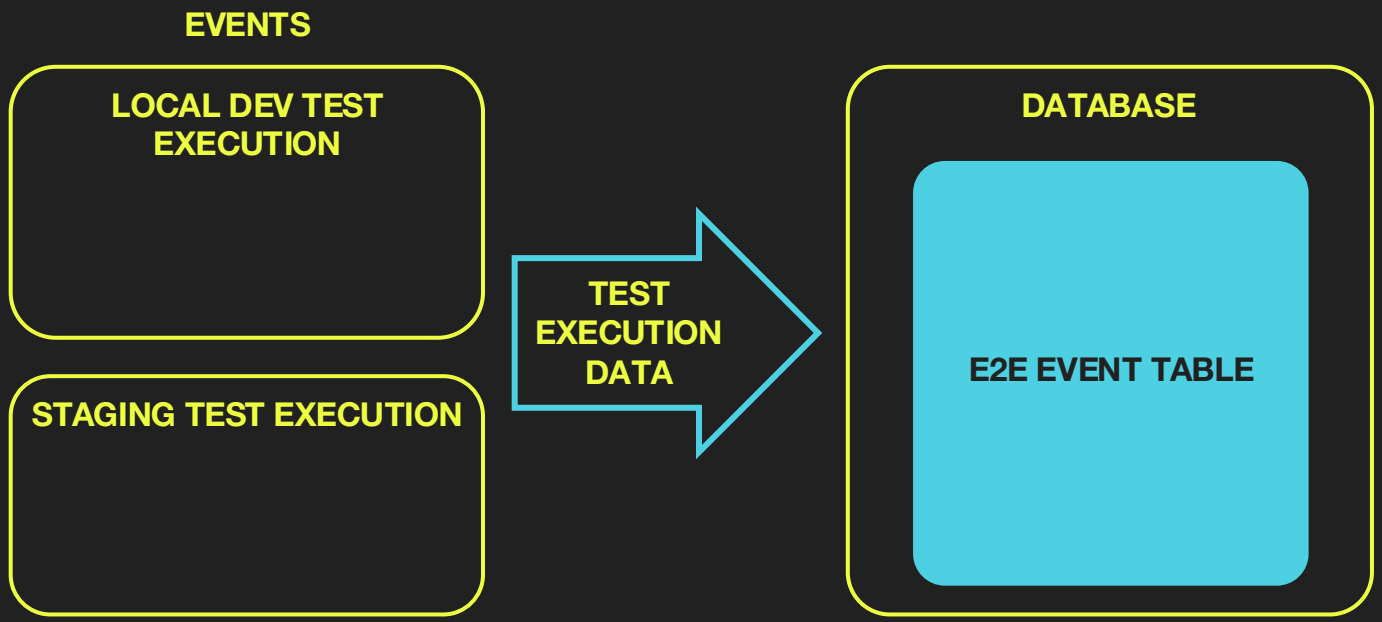
Finding the root cause



Test execution data



Test execution data

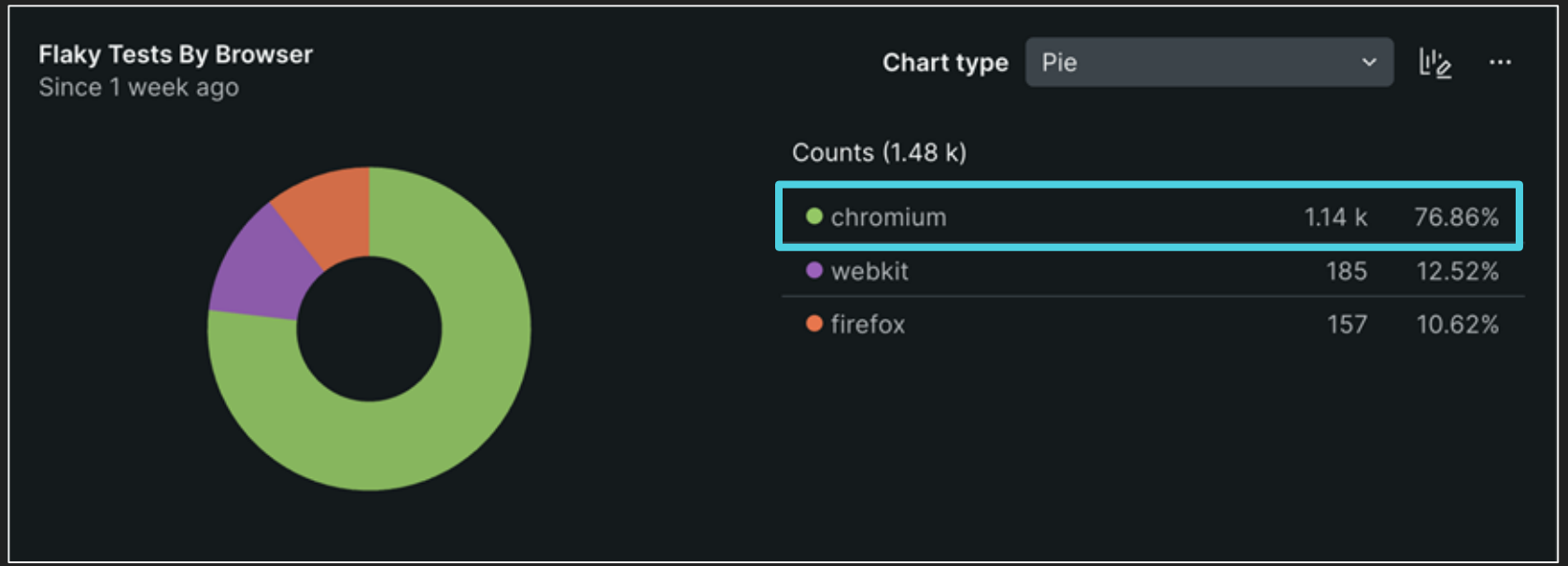


Query your data

NRQL (New Relic Query Language)

```
FROM E2E SELECT count(*) WHERE outcome =
'flaky' FACET browserName SINCE LAST WEEK
```

Flaky tests by browser



Query your data

NRQL (New Relic Query Language)

```
FROM E2E SELECT count(*) WHERE outcome =
'flaky' AND browserName = 'chromium'
FACET errorMessage SINCE LAST WEEK
```

Query your data

NRQL (New Relic Query Language)

```
FROM E2E SELECT count(*) WHERE outcome =
'flaky' AND browserName = 'chromium'
FACET errorMessage SINCE LAST WEEK
```

Top error messages

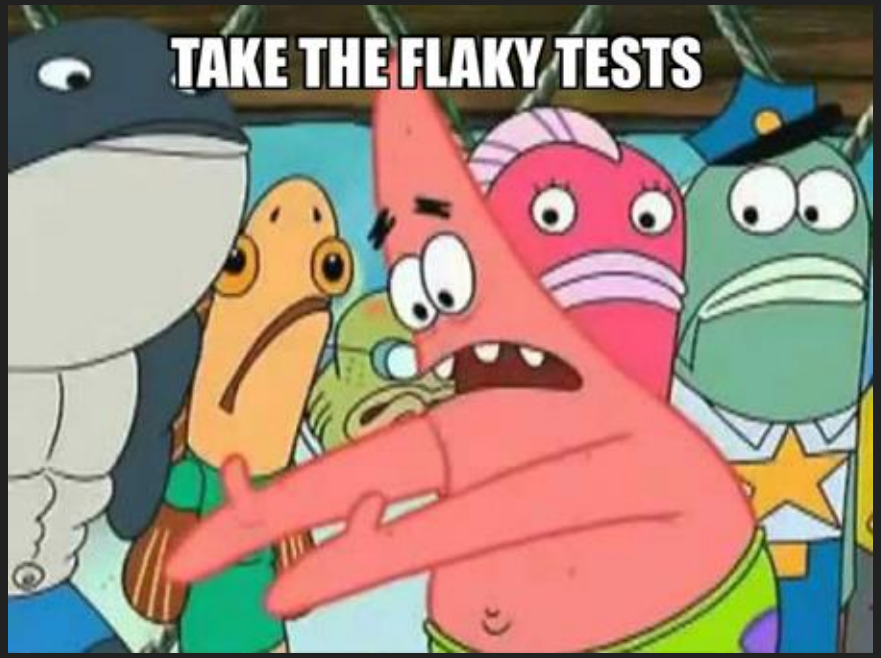
ERROR MESSAGE	COUNT
Timed out 5000ms waiting for expect(locator).toBeVisible()	588
locator.click: Test timeout of 90000ms exceeded.	369
expect.toHaveText: Error: strict mode violation: locator('.ace_line') resolved to 2 elements	112

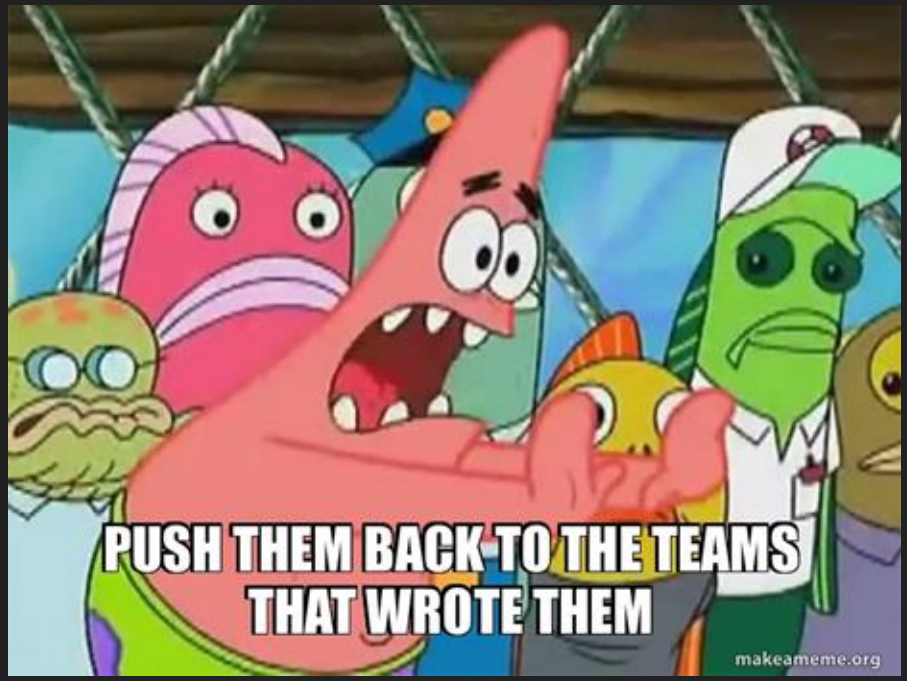
...

Top error messages

ERROR MESSAGE	COUNT
Timed out 5000ms waiting for expect(locator).toBeVisible()	588
locator.click: Test timeout of 90000ms exceeded.	369
expect.toHaveText: Error: strict mode violation: locator('.ace_line') resolved to 2 elements	112

...


TAKE THE FLAKY TESTS

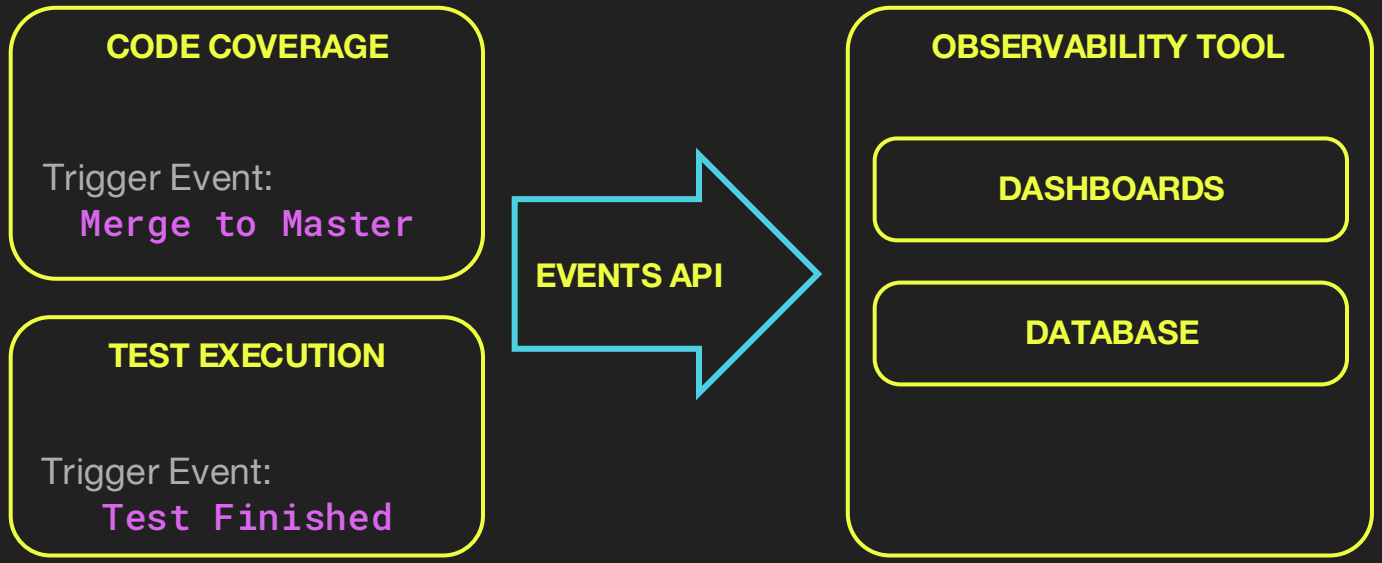

PUSH THEM BACK TO THE TEAMS
THAT WROTE THEM
makeameme.org


FIX
THE TEST.
IGNORE THE
FLAKY TEST.
DELETE
THE TEST.
FLAKINESS SOLVED.
imgflip.com

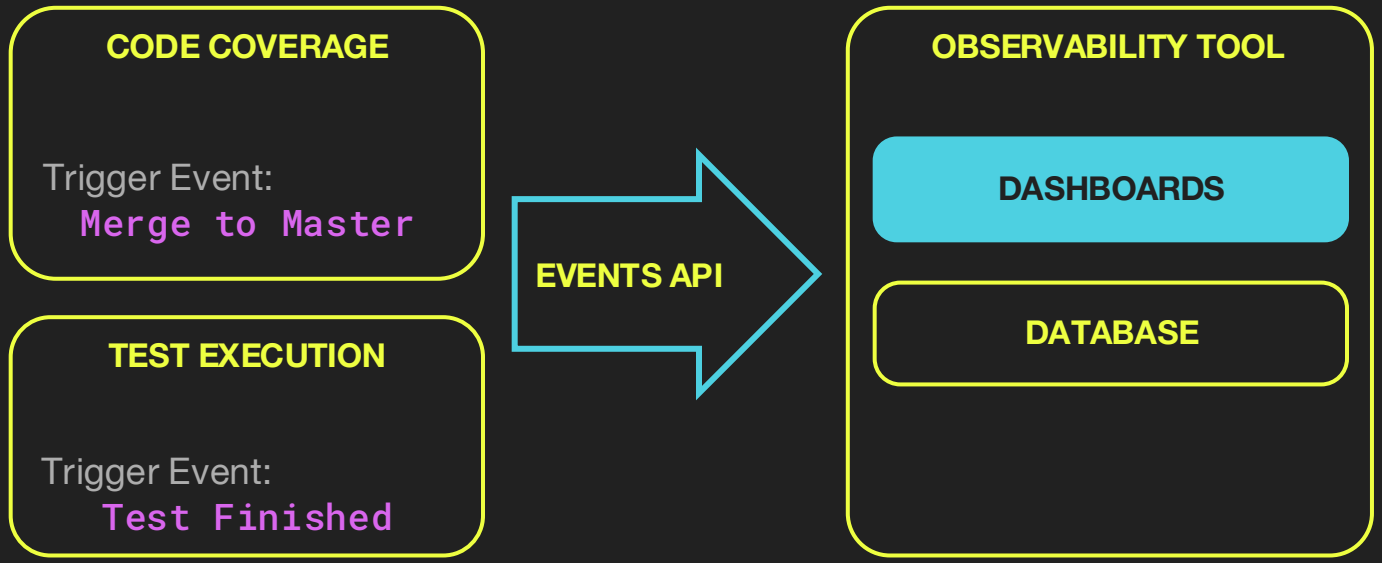
Flakiness mitigation strategy

1. Solid waiting strategies with Playwright
2. Page Objects for easy test maintenance
3. Pilot team led the way - others followed

Visibility for everyone



Visibility for everyone

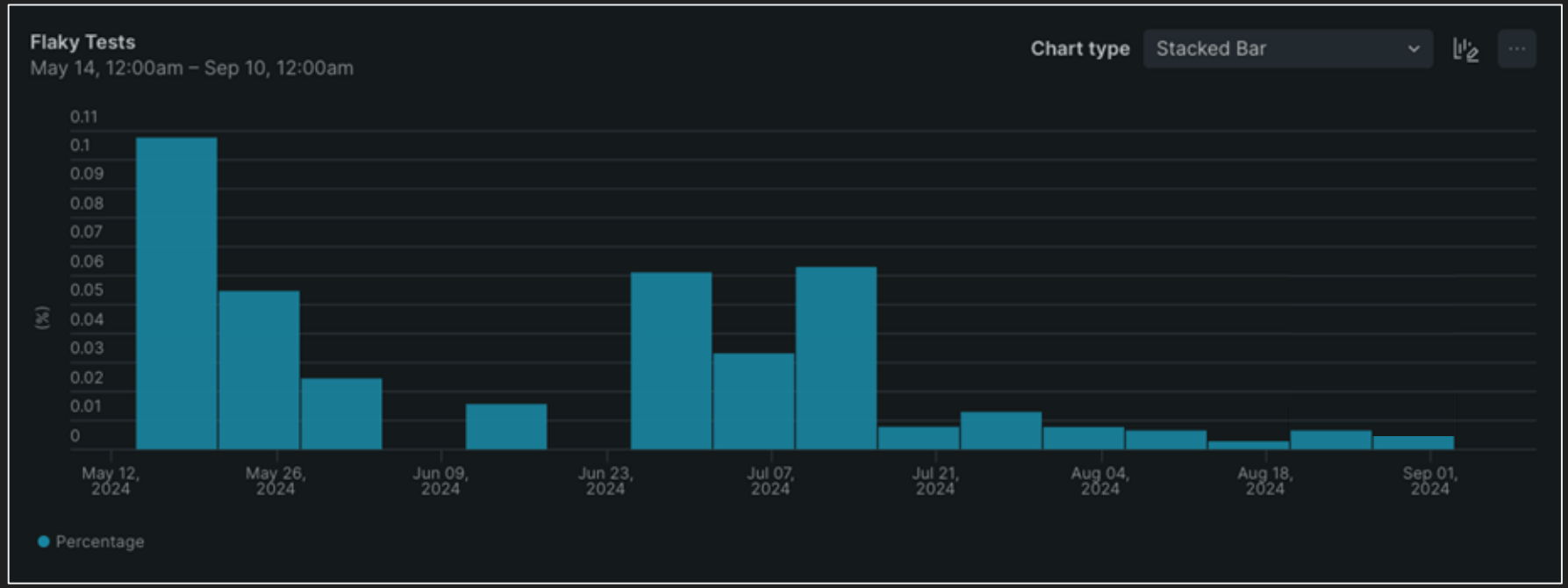


Dashboard for teams

Measuring success

```
SELECT percentage(count(*), WHERE outcome
= 'flaky') FROM E2E TIMESERIES SINCE
'2024-05-10' UNTIL '2024-09-10'
```

Measuring success



Results

5%

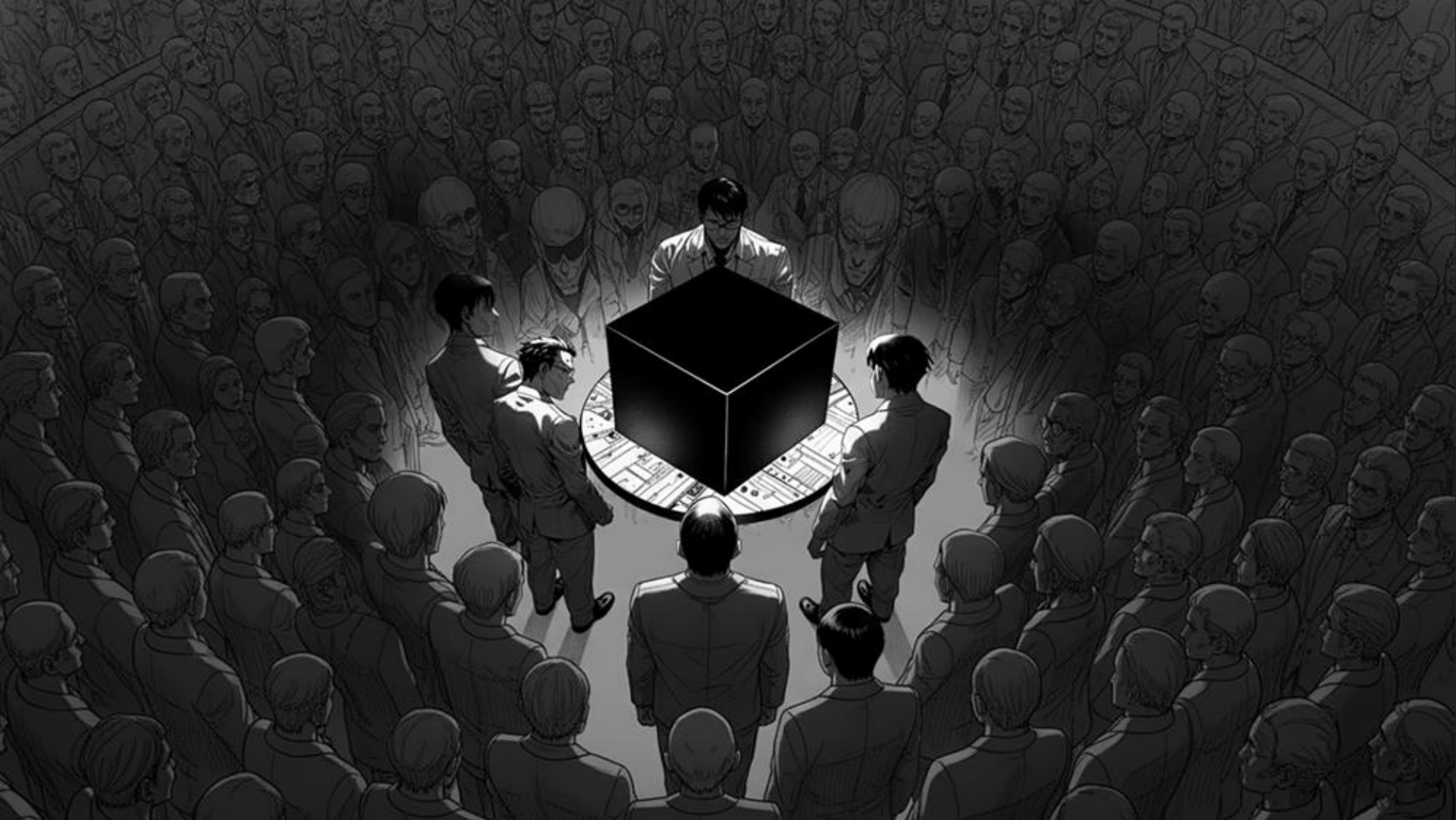
initial flakiness percentage

In May 2024

<1%

current flakiness percentage

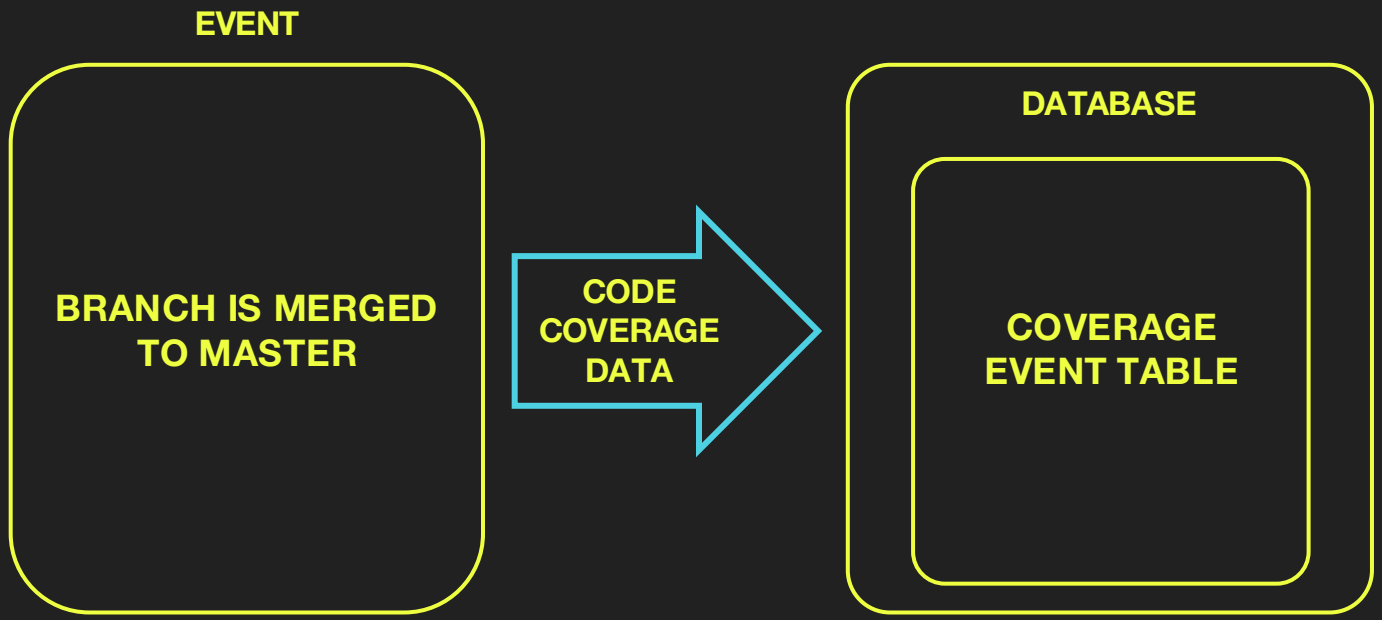
Since 1 year ago



Code Coverage 2

Stories

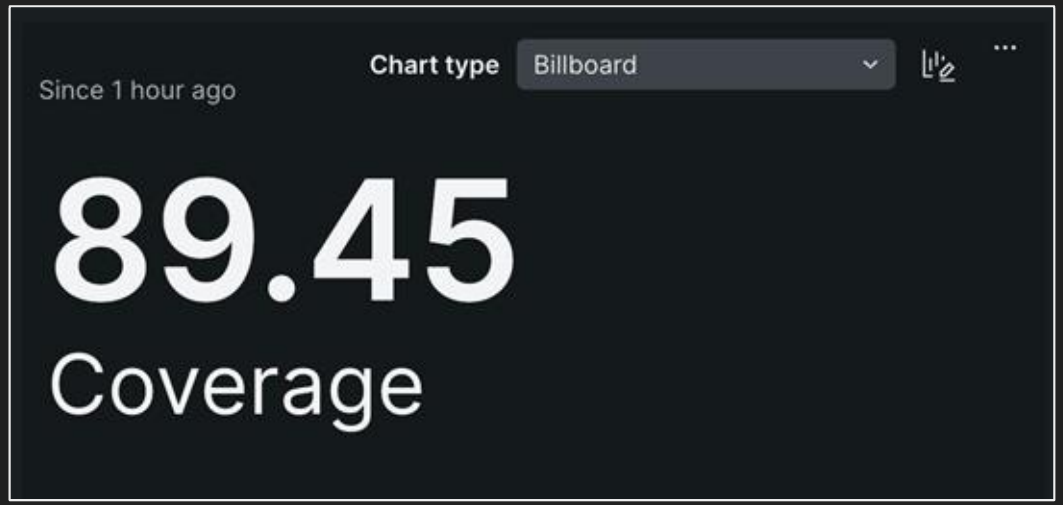
Code coverage data



Code coverage per team

```
FROM Coverage SELECT average(coverage)
where team = 'Scorecards'
```

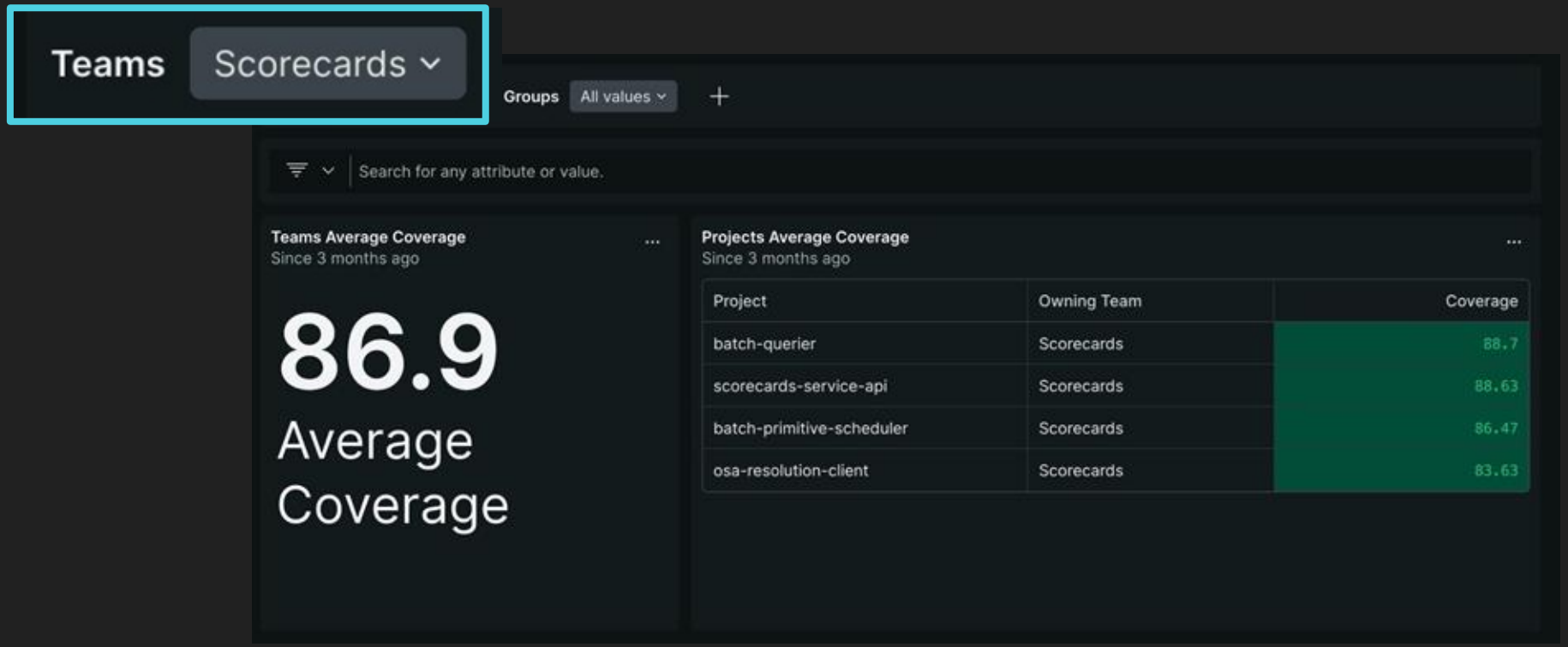
Code coverage per team



Code coverage per team and repository

```
FROM Coverage SELECT latest(coverage)
where team = 'Scorecards'AND
project = 'scorecards-api'
```

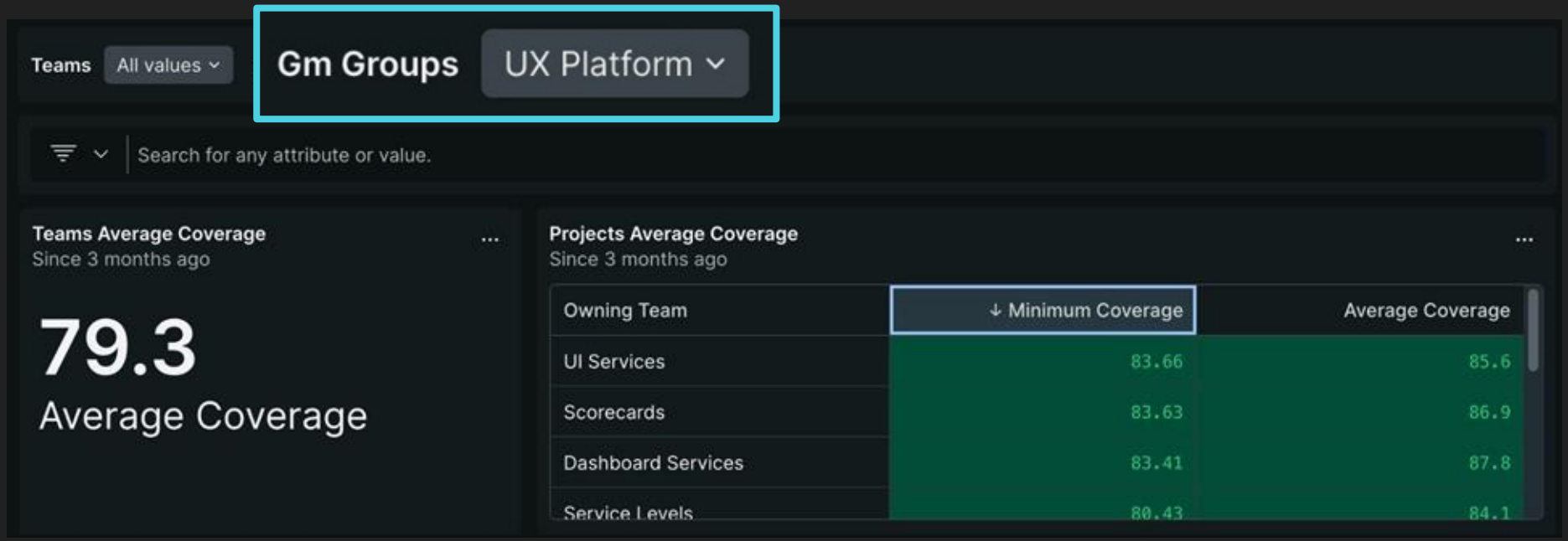
Code coverage dashboard



Code coverage per group

```
FROM Coverage SELECT average(coverage)
where engineering_group = 'UX Platform'
```

Code coverage dashboard




WHAT DO WE WANT?
CODE COVERAGE
HOW MUCH
DO WE WANT IT?
67% 34% 93%
imgflip.com

https://www.reddit.com/user/Shiroyasha_2308/

Now it’s your turn!

Define what data is important to you

Send it to your observability platform

Keep it simple

Get started and iterate!

Data enables collaboration

Still hungry? Learn more

Thank you!

Linkedin: martinrodrigo - **Github**: rodrigojmartin - **Soundcloud**: adids

expoqa®26

MADRID 26th, 27th & 28th May

Thank you for attending

expoqa.eu