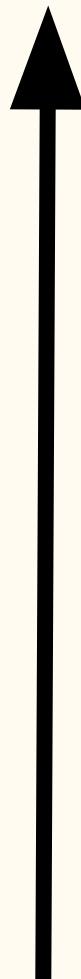
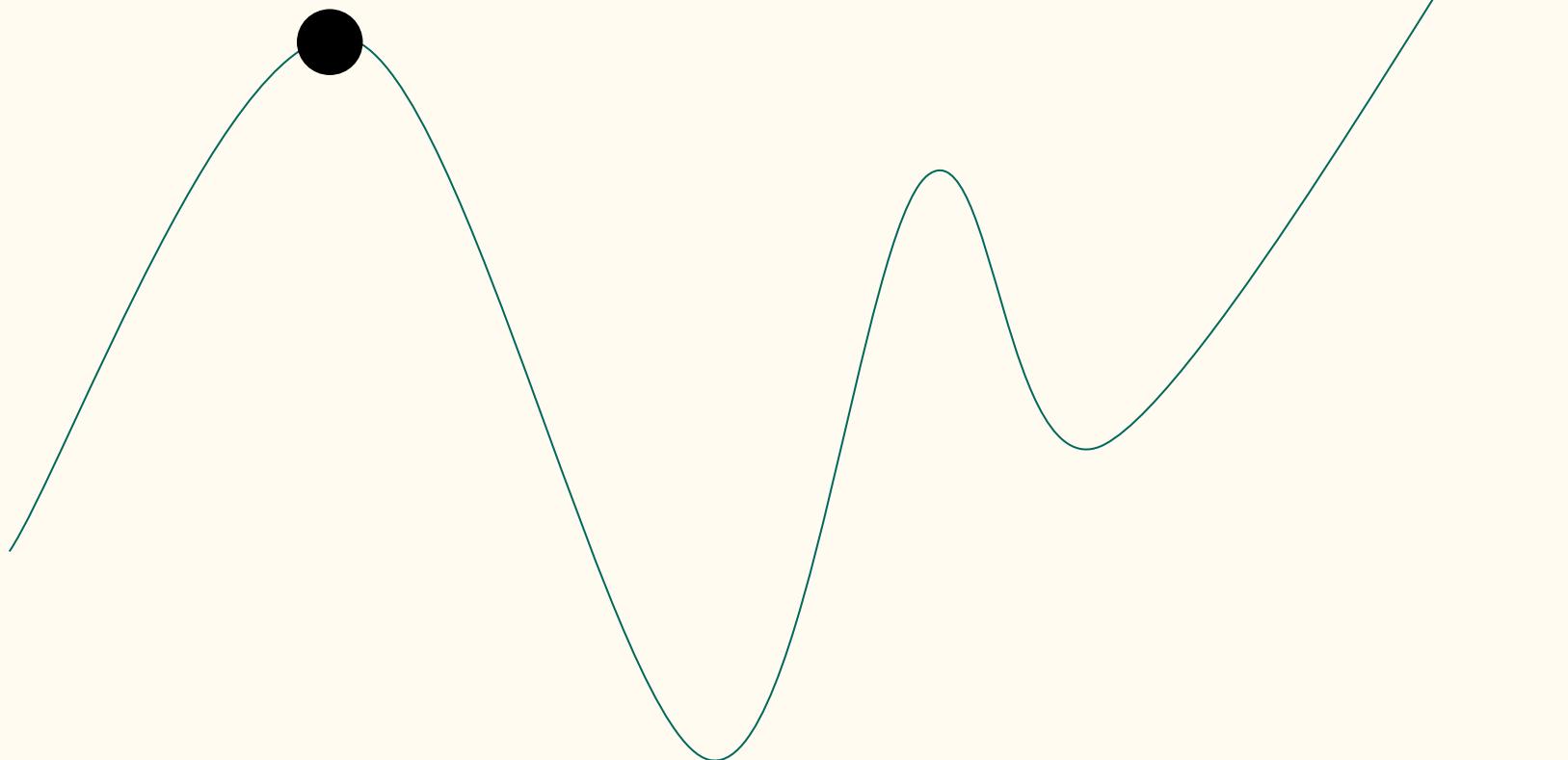


A culture of observing

How to get better understanding
through observability

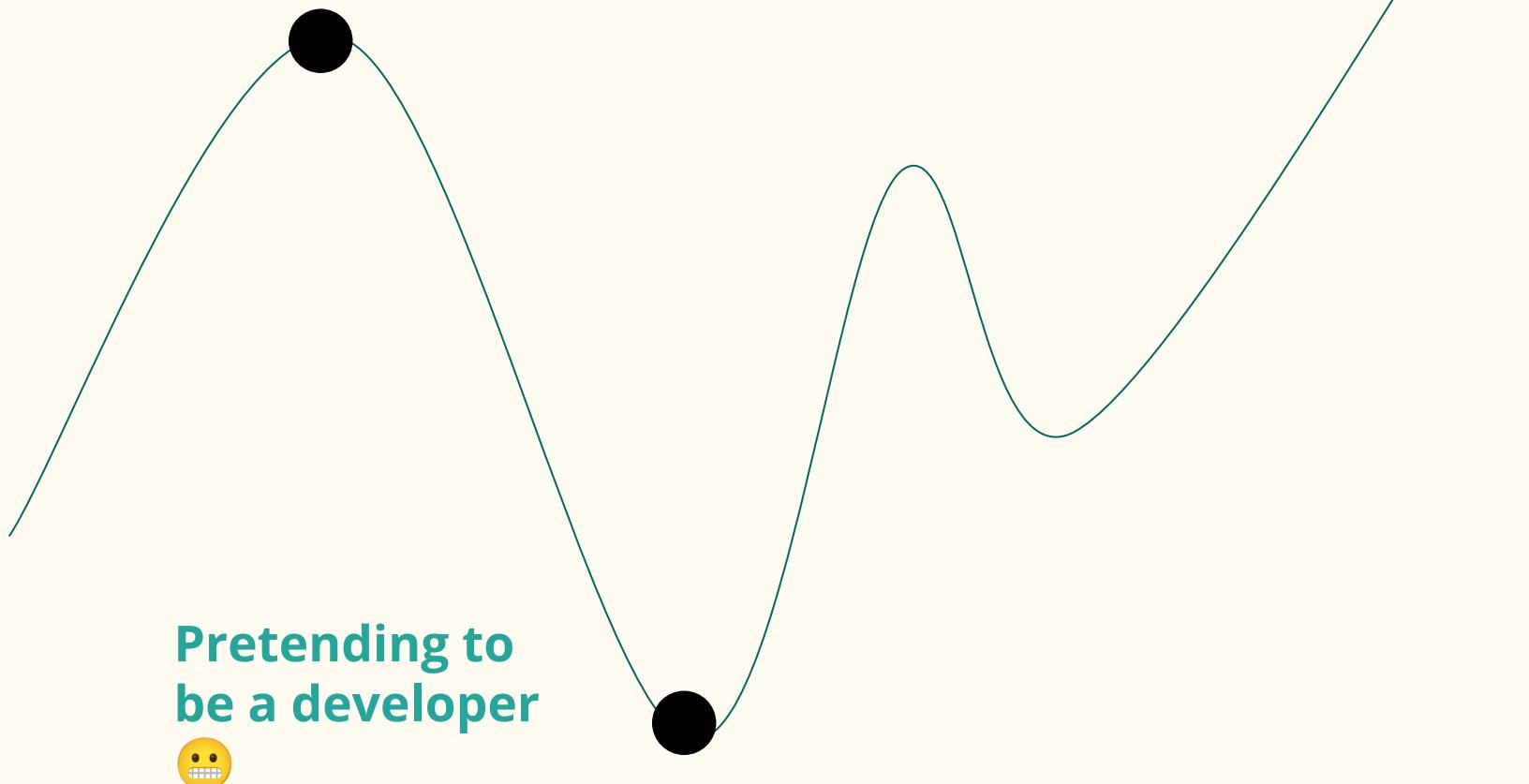


Don't care how the
world works 😎





Don't care how the
world works 😎





Don't care how the
world works 😎



I've got this.
It's easy 💪



Pretending to be
a developer 😷





Don't care how the
world works 😎



I've got this.
It's easy 💪



Pretending to be
a developer 😷



Oops,
production
breaks 😰





Don't care how the world works 😎



I've got this.
It's easy 💪



Observability



Pretending to be a developer 😊



Oops,
production
breaks 😢



observability = o11y

observabilityy = o**11**y



A numeronym



“What about observability?”

“How do you know it will work in production?”

“Have you seen this spike on Datadog?”



“What about observability?”



That's how you get the nickname
To11y

Toli + o11y 

To11y

Apostolis (Toli) Apostolidis

@apostolis09



$\frac{1}{2}$  $\frac{1}{2}$ 



Why am I here?

The more **observable** our systems,
the more democratised our
understanding

The more **observable** our systems,
the more democratised our
understanding*

***of our software systems are across the org**

This talk is about
observability.

If you can understand any novel state
without needing to ship new code,
you have **observability**.

Charity Majors, Liz Fong-Jones, George Miranda.
Observability Engineering



It's a **property** of the system



Explain any bizarre or novel state



Without shipping **new code**

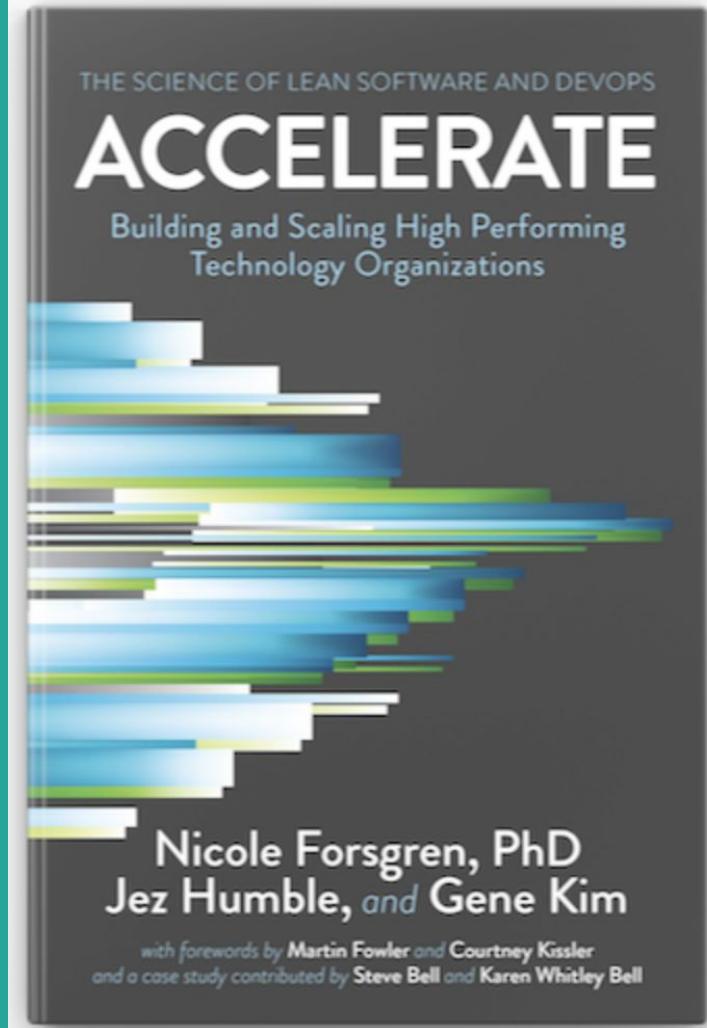


Without **predicting** any state

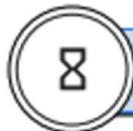
It's **not** yet
another fad.



The cover features the Google Cloud logo, the DORA logo, and the text "Community of Practice". Below this is a cartoon illustration of four small characters (two yellow, one green, one pink) around a house-like icon with the word "DORA" on it. The text "2022 Accelerate State of DevOps" is prominently displayed. A young man with curly hair, wearing a yellow sweater, is shown smiling and pointing at a laptop screen.



SOFTWARE DELIVERY PERFORMANCE



lead time for changes



time to restore service



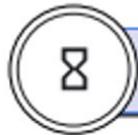
deployment frequency



change failure rate

Delivery **predicts** company success

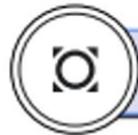
SOFTWARE DELIVERY PERFORMANCE



lead time for changes



time to restore service



deployment frequency



change failure rate

OPERATIONAL PERFORMANCE

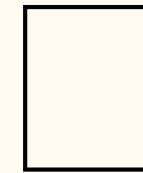
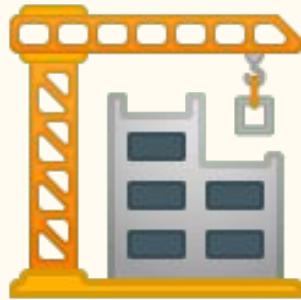


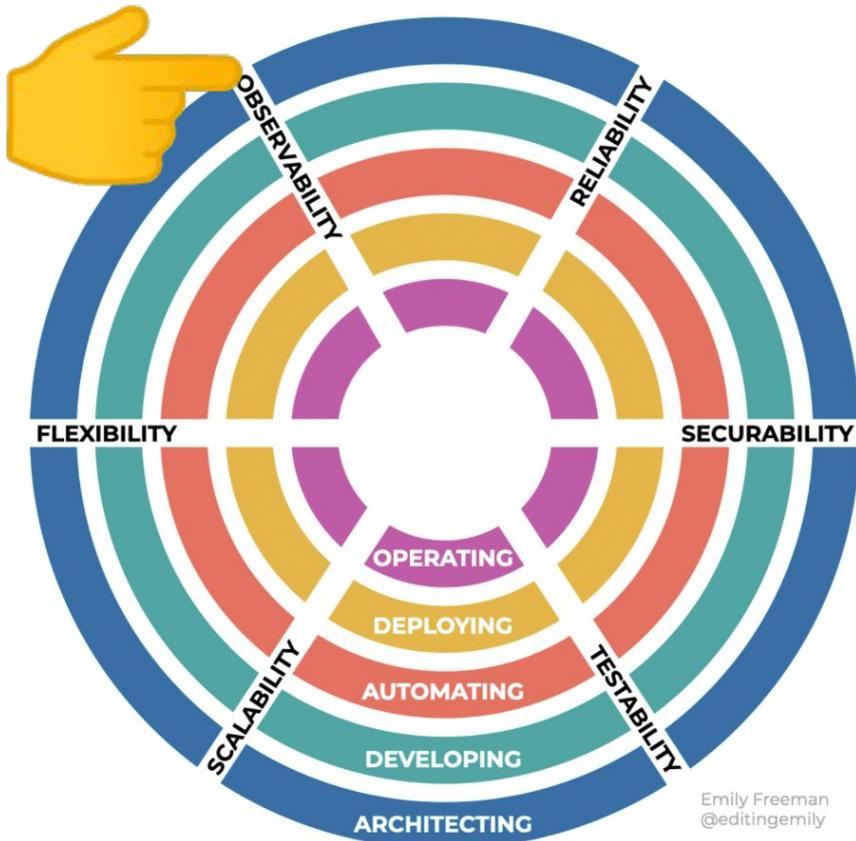
reliability



It's not enough to **deliver** well, you need to **operate reliably** too.

Build, ship, support

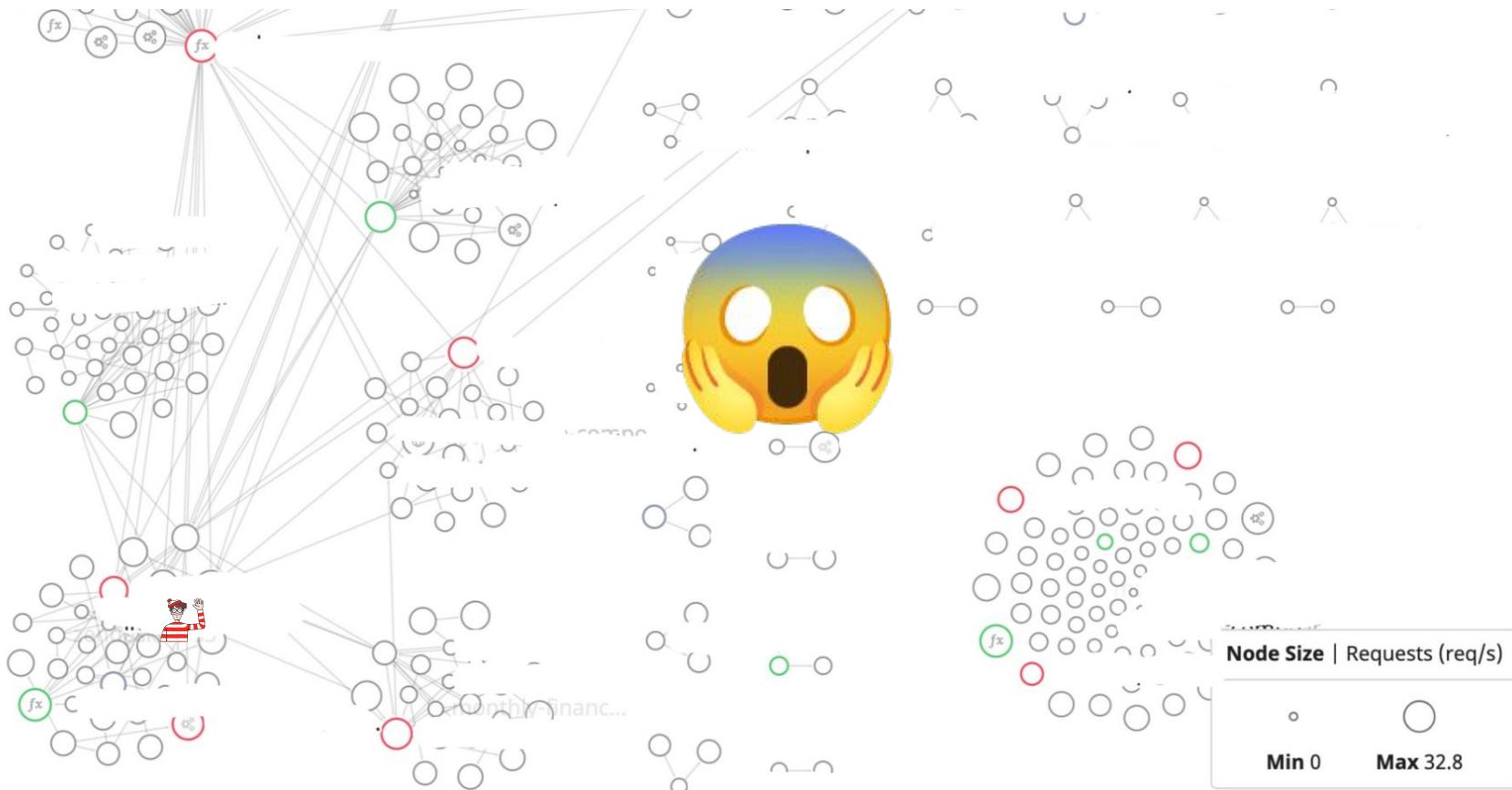




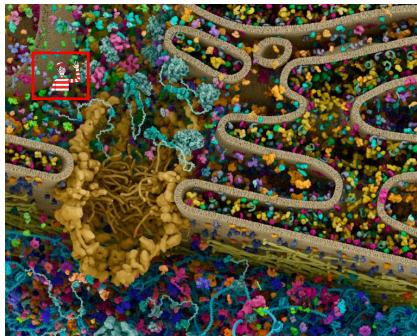
Revolution Model, *Emily Freeman*



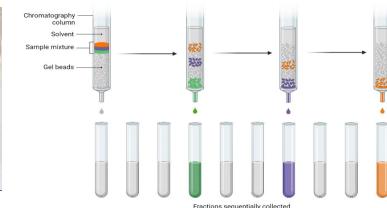
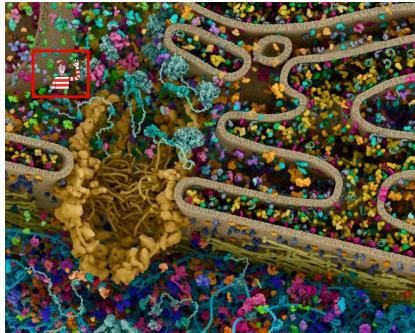
Systems are
complex.



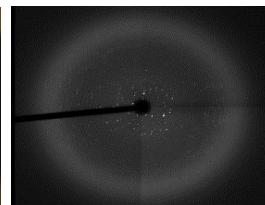
Inside a cell



Inside a cell

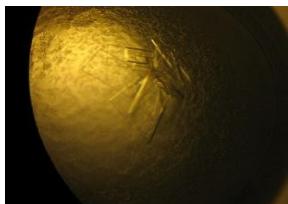
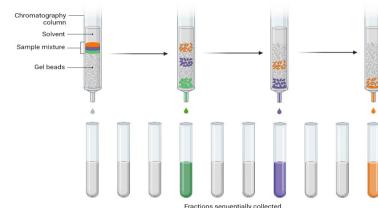
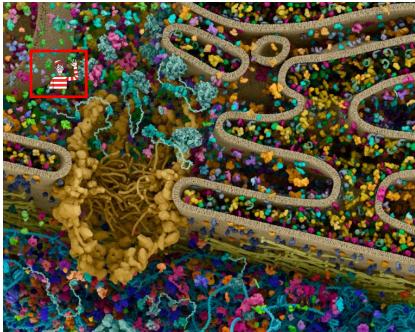


Protein crystals

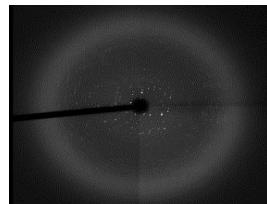


XRay diffraction

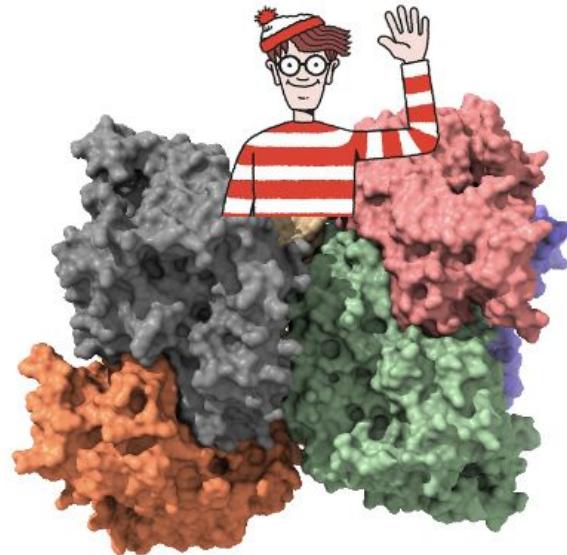
Inside a cell



Protein crystals

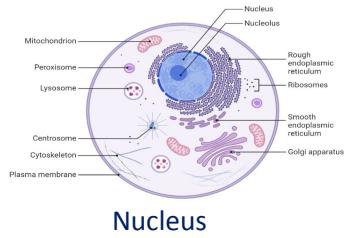


XRay diffraction

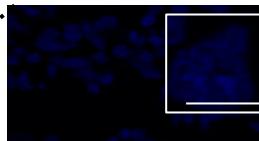


2.5 Å

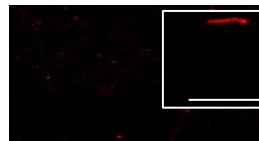
2-7 years



Fluorescent laser
microscopy
Live cell imaging



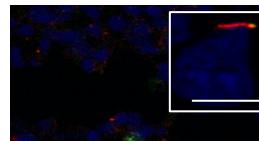
10 µm



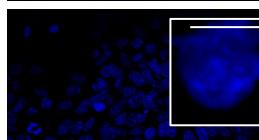
10 µm



10 µm



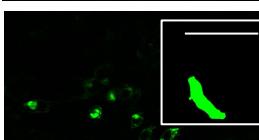
10 µm



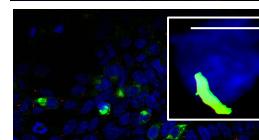
10 µm



10 µm



10 µm

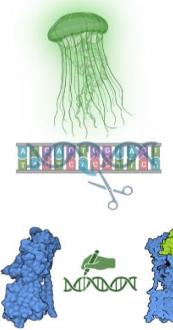


10 µm

Cell signalling



Prim. Cilium



Channels
MERGED



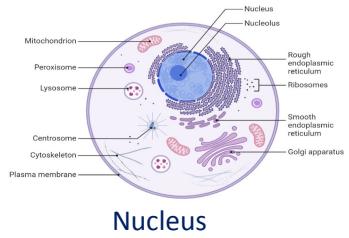
Control



Stimulus



CRISPR/CAS9 Genetic
Reprogramming



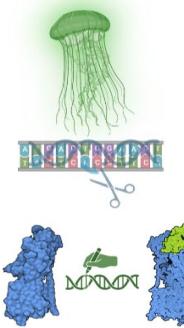
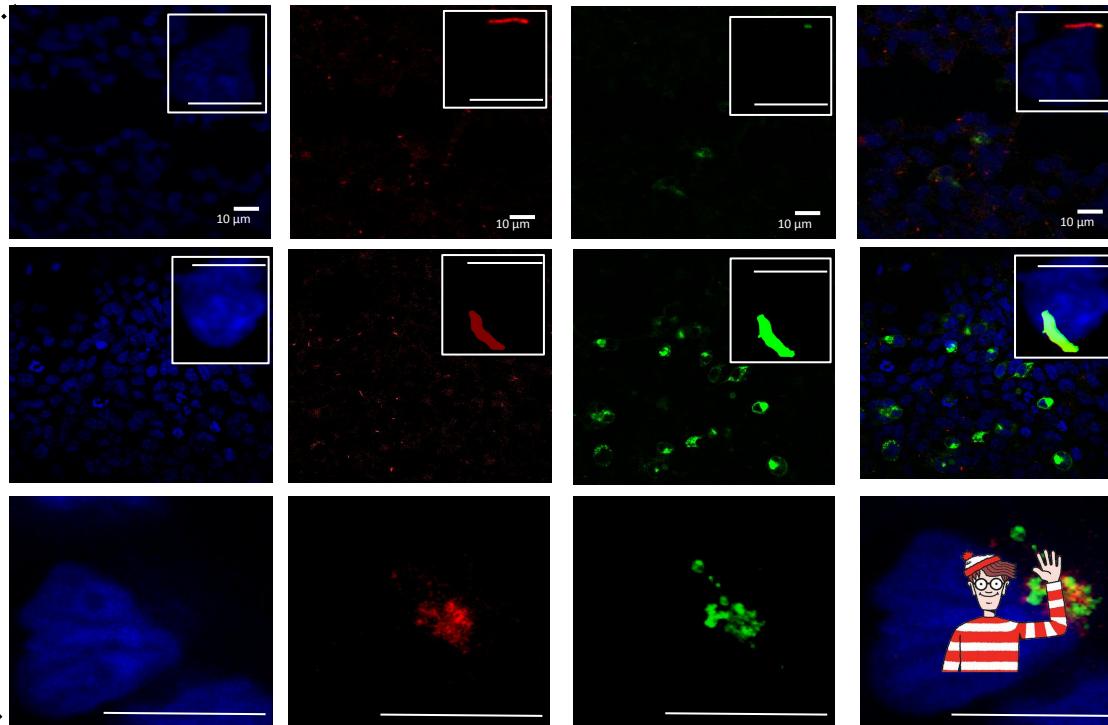
Cell signalling



Prim. Cilium



Fluorescent laser
microscopy
Live cell imaging



Channels
MERGED



Control



Stimulus



CRISPR/CAS9 Genetic
Reprogramming



It's a **property** of the system



Explain any bizarre or novel state



Without shipping **new code**



Without **predicting** any state

So, we need it.

We established a
culture of observability
at cinch.

It took us **3 years.**

How can **you**
establish a
culture of
Observability?

1. Choose an observability tool

1. Choose an observability tool

2. “Enable” telemetry data

1. Choose an observability tool
2. “Enable” telemetry data
3. Auto-generate the dashboard

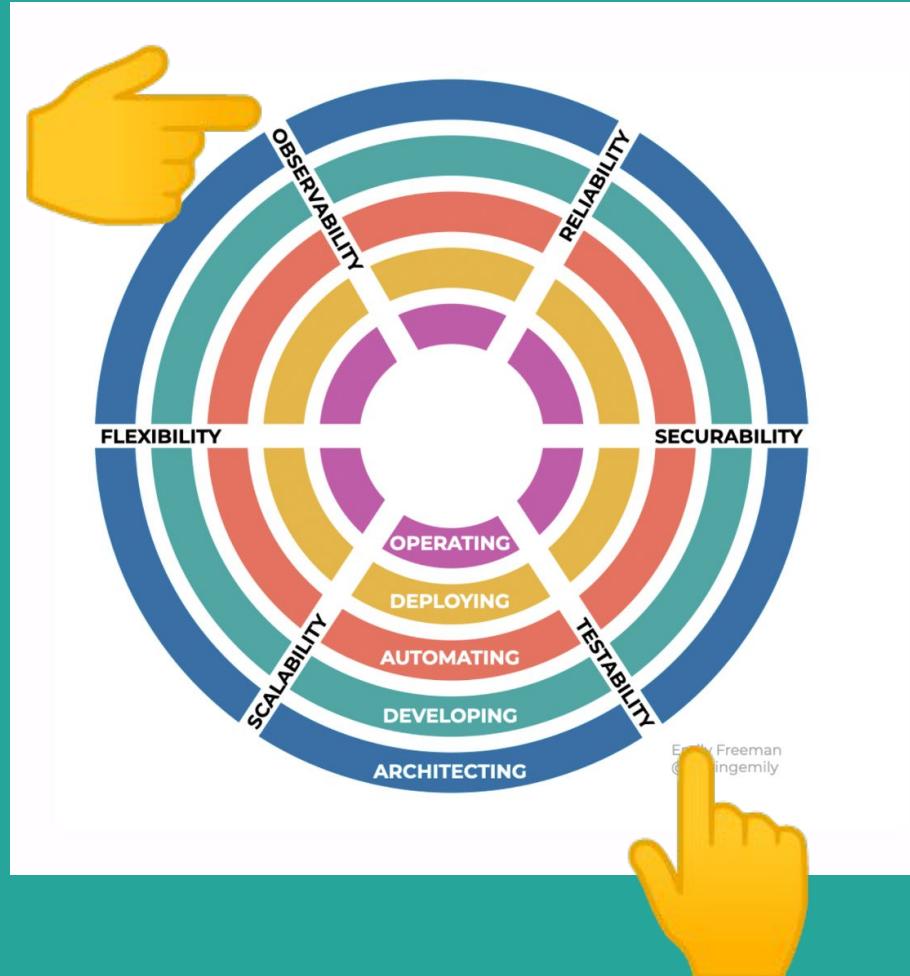
1. Choose an observability tool
2. “Enable” telemetry
3. Auto-generate dashboards





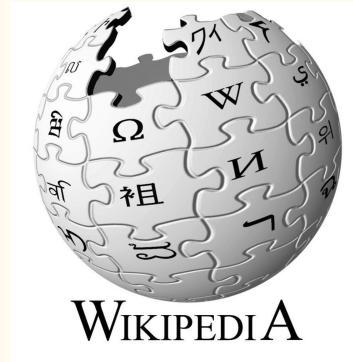
MOVIECLIPS.COM

Observability like Testability

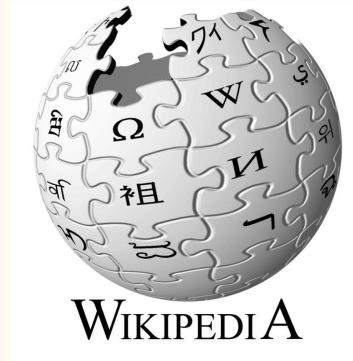


Observability is the **testability** of
the 2020s, but
better.

...the degree to which a software artifact **supports testing** in a given test context.



If **testability** is high, then **finding faults** is **easier**.



Observability is **not as**
mainstream as testability

Tests tell you what's broken

Tests tell you what's broken

Observability tells you what's happened

Observability > Testability

Testability



Observability

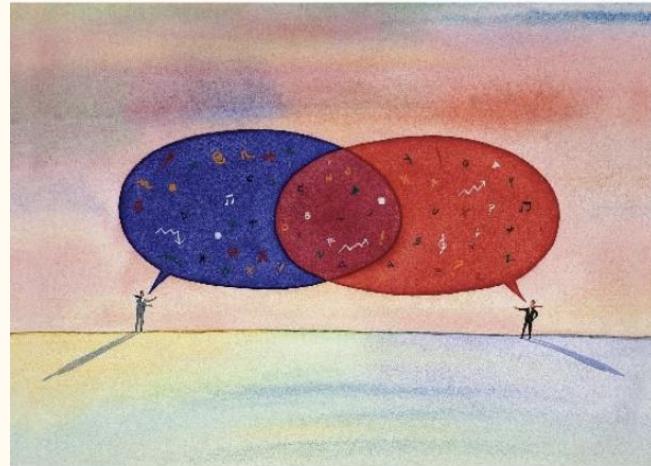


O'REILLY®

"Truly, a blueprint for the systems of tomorrow."
—Mike Dvorkin, Distinguished Engineer at Cisco

THINKING IN PROMISES

DESIGNING SYSTEMS FOR COOPERATION



What **promise** are
we keeping with
testing?

MARK BURGESS

I **promise** that I will change the software, and it **won't get worse.**

I **promise** that I will change the software, and it **won't get worse**.

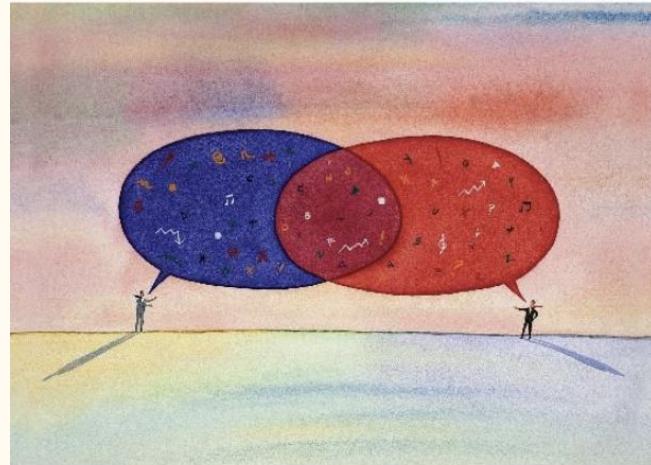
If anything, it will **get better**.

O'REILLY®

"Truly, a blueprint for the systems of tomorrow."
—Mike Dvorkin, Distinguished Engineer at Cisco

THINKING IN PROMISES

DESIGNING SYSTEMS FOR COOPERATION

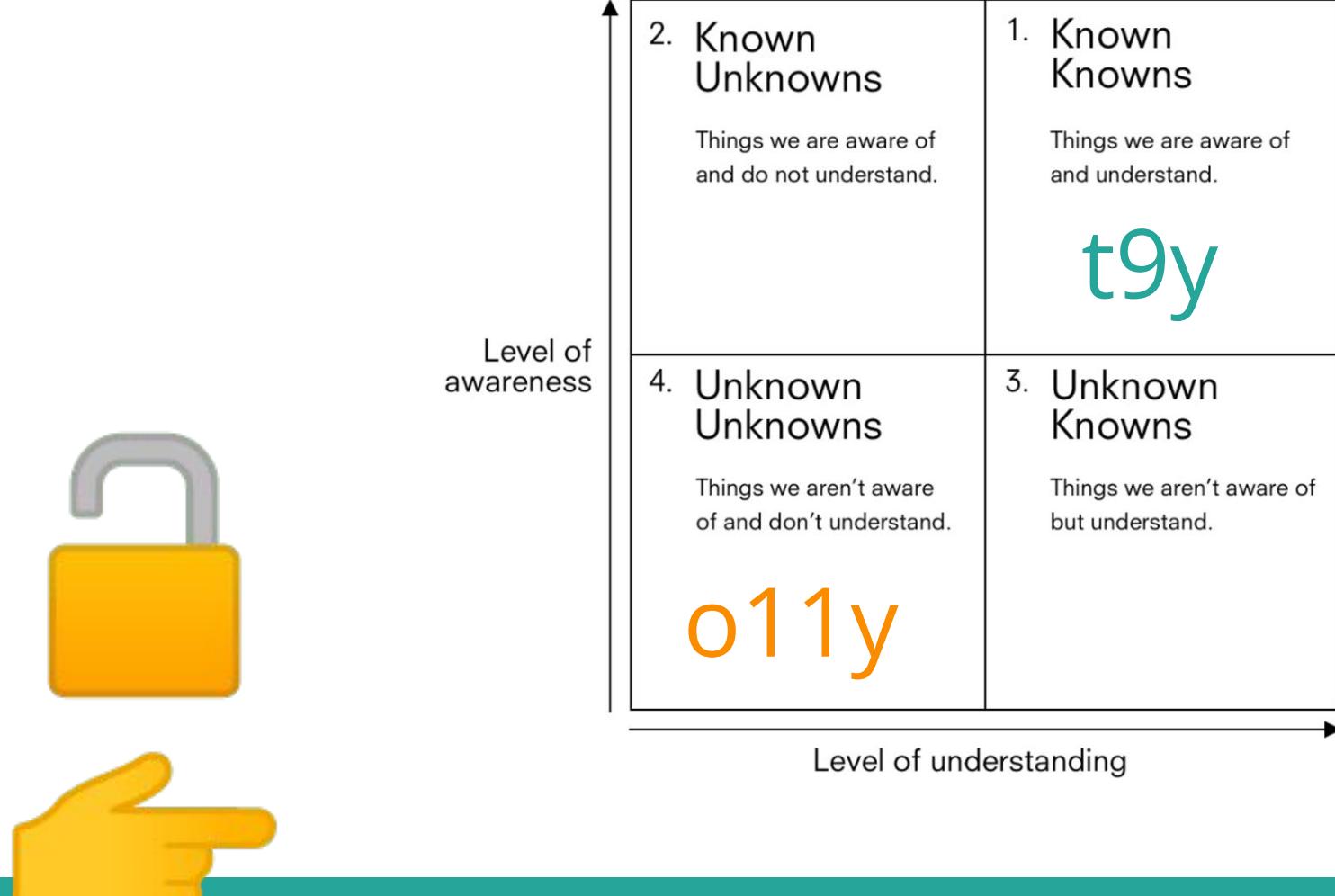


What **promise** are we keeping with **observability**?

MARK BURGESS

I **promise** that you will **have a**
service,
and you can **rely** on me.

If I make a change, I **promise** that
your service will **get better**, and I
will know if it doesn't.



OK, what do I do?

Improve how we
create & curate
our telemetry data to
enable observability.

Where do I
start?

 Pick **o11y tool** & let everyone have it **Instrument** code Support & enable **learning**



Pick o11y tool & let everyone have it



- Instrument code



- Support & enable learning

Traditional



Next-gen



Serverless



“3 pillars”



**“Everything
is an event”**



**“Betting on
serverless”**



How do I
choose?



Can **everyone** use the service with no extra cost?



Can **everyone** use the service with no extra cost?



Does the vendor have excellent **DX tooling**?



Can **everyone** use the service with no extra cost?



Does the vendor have excellent **DX tooling**?



How far do you get **without checking production**?

 Can **everyone** use the service with no extra cost?

 Does the vendor have excellent **DX tooling?**

 How far do you get **without checking production?**

 Can you manage **billing** effectively?



Pick o11y tool & let everyone have it



Instrument code



Support & enable learning



Pick o11y tool & let



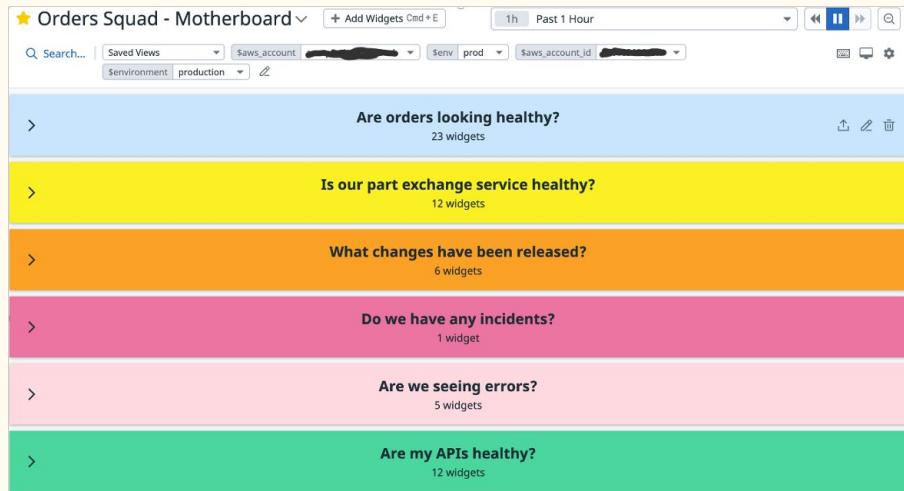
Instrument code



Support & enable



```
orderPlaced(orderEvent) {  
    tag(orderId, "X")  
    tag(userId, "Y")  
    <business logic>  
    tag(orderPlaced, true)  
}
```



```
orderPlaced(orderEvent) {  
    tag(orderId, "X")  
    tag(userId, "Y")  
    <business logic>  
    tag(orderPlaced, true)  
}
```

Screenshot of a monitoring dashboard titled "Orders Squad - Motherboard". The dashboard shows six colored cards with status questions and widget counts:

- Blue card: "Are orders looking healthy?" (23 widgets)
- Yellow card: "Is our part exchange service healthy?" (12 widgets)
- Orange card: "What changes have been released?" (6 widgets)
- Pink card: "Do we have any incidents?" (1 widget)
- Pink card: "Are we seeing errors?" (5 widgets)
- Green card: "Are my APIs healthy?" (12 widgets)



Learning **how** to
instrument is
important

```
log("An order was placed  
with order id = XYZ")
```

log("An order was placed
with order id = XYZ")

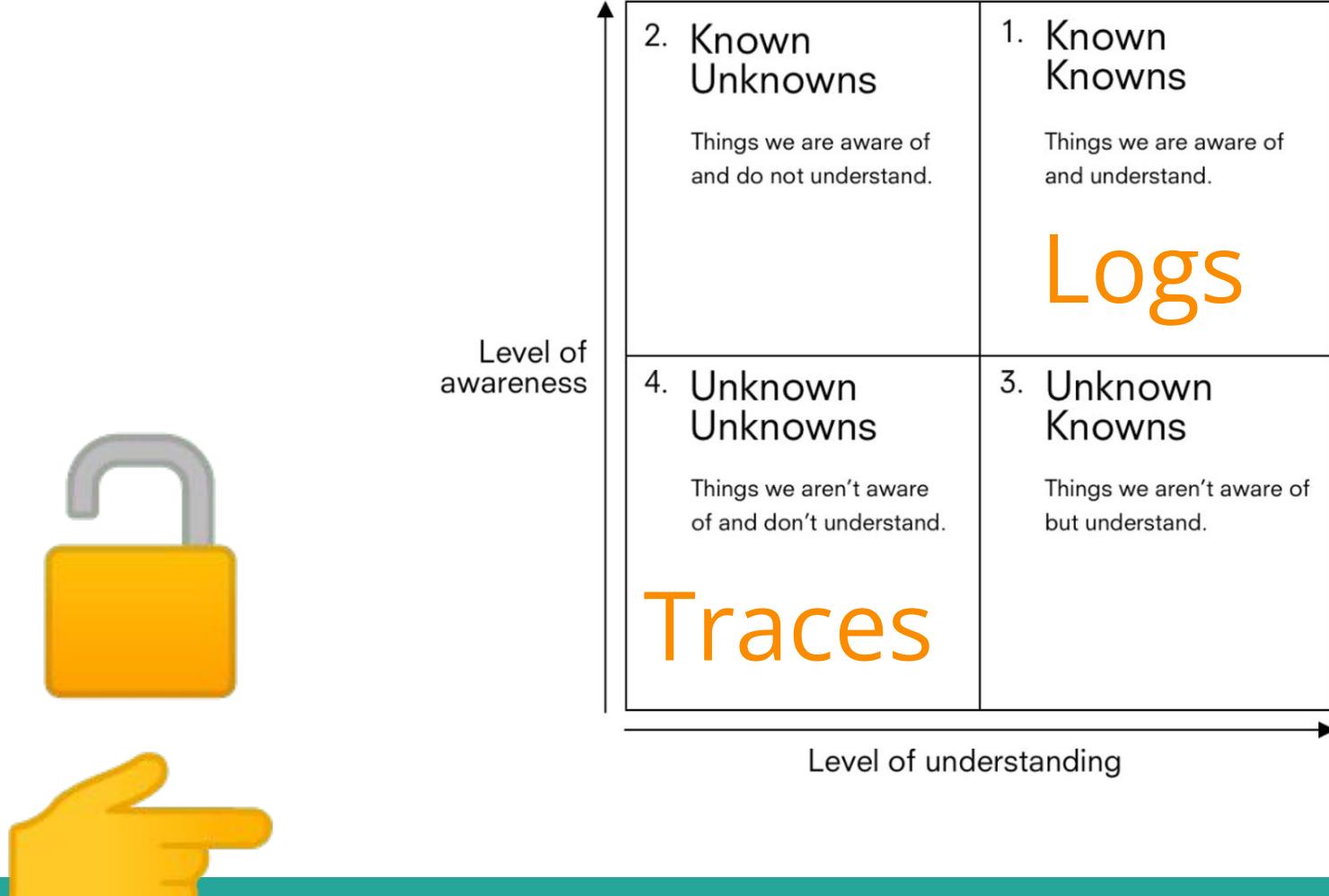


log("An order was placed
with order id = XYZ")



addTag("orderId", XYZ)
addTag("userId", ABC)
addTag("orderPlaced", true)

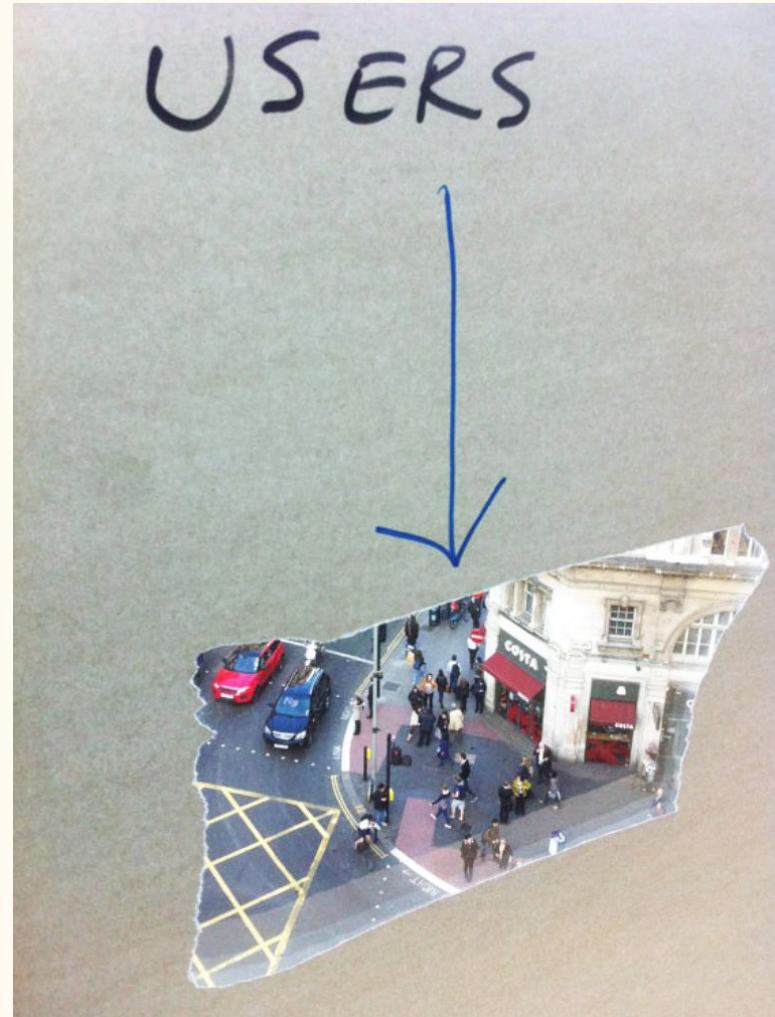




Think of **every** input, output or parameter in your code as an **opportunity** to **learn**.

```
orderPlaced(orderEvent) {  
    tag(orderId, "X")  
    tag(userId, "Y")  
    <business logic>  
    tag(orderPlaced, true)  
}
```

```
orderPlaced(orderEvent) {  
    tag(orderId, "X")  
    tag(userId, "Y")  
    <business logic>  
    tag(orderPlaced, true)  
}
```



select * from orderEvent

```
select * from orderEvent  
where userId=X
```

You can ask
any question.

1 Code

2 Test

3 Instrument

1 Test

2 Code

3 Instrument

1 Test

2 Instrument

3 Code

1 Instrument

2 Code

3 Test



Pick o11y tool & let everyone have it



Instrument code



Support & enable learning

Instrumenting
won't quite cut
it

...you can **construct the behaviour of a system** from its externally visible components.

*Adrian Cockcroft,
Hacking The Org podcast*



=



How can teams
make **sense** of
it?



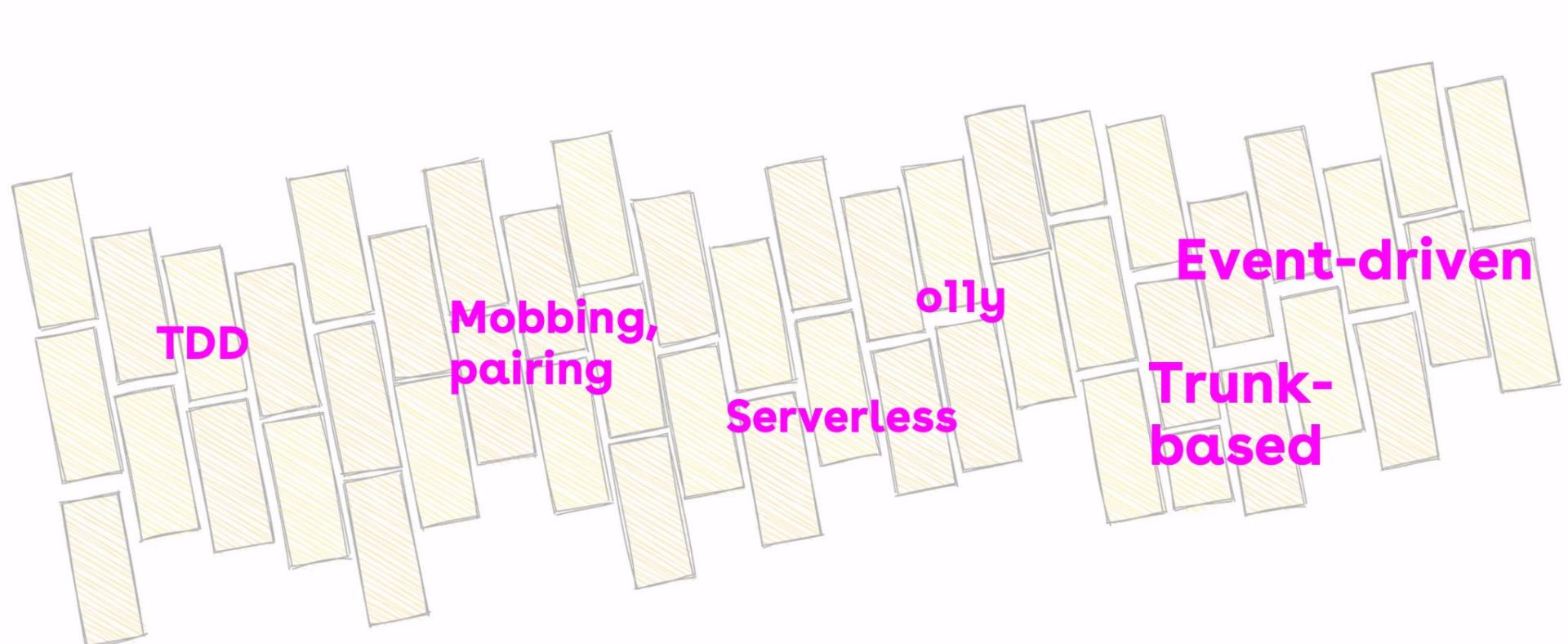
Support



Learn & Improve



Practice



TDD

Mobbing,
pairing

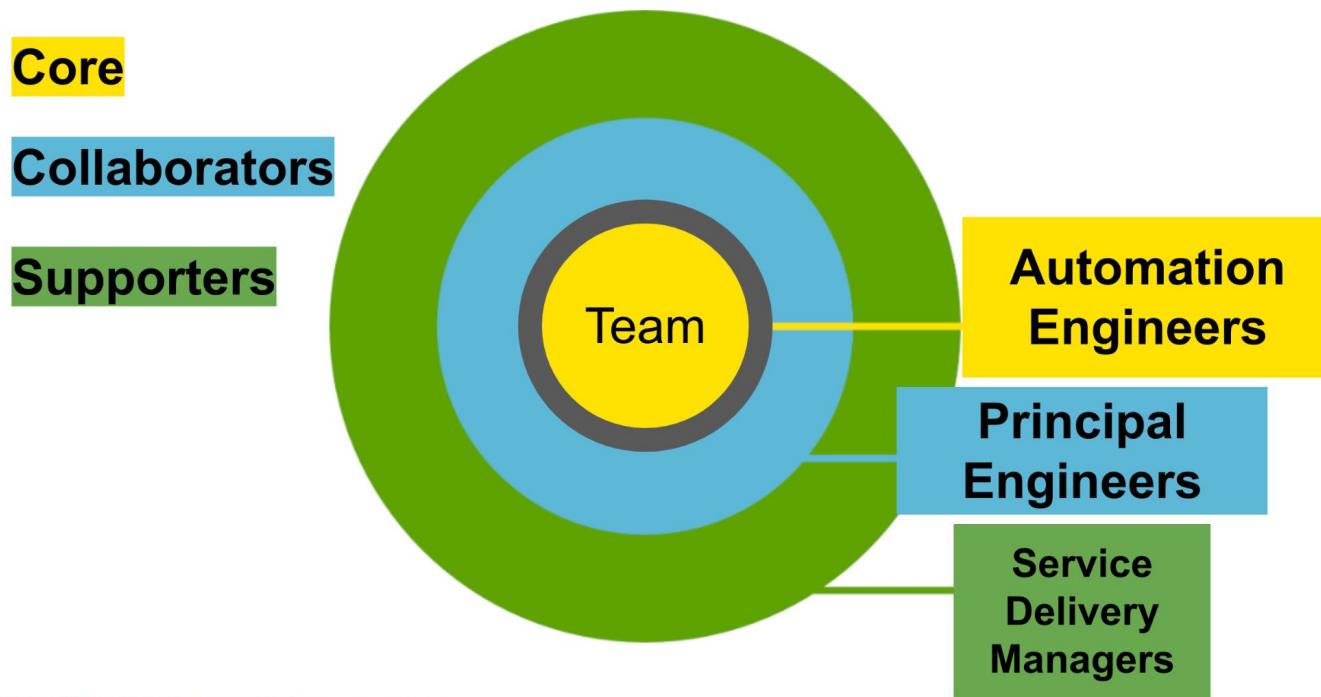
Serverless

o1ly

Event-driven
Trunk-
based



Add **o1ly** to your golden path



<https://teamonion.works/>, Emily Webber



Plant **enablers** into teams



Community of Interest

★ RUM Datadog Workshop - LI



Created by Sarika Antony

Last updated 12 days ago

RUM Datadog Workshop ...

- [Introduction 🎬](#)
- Warm up - Personal...
 -
- RUM 📈
 - Quick Intro
 - UX Monitoring - Re...
- Instrumenting Cust...
 - Browsing the Docu...
 - Frontend Telemetry
- Feedback time 🕒

Introduction 🎬

In this session, you will learn to navigate yourself around the Datadog UI and understand the various Datadog views and products that underpin cinch's observability and monitoring platform.

To start with you can have a quick glance at:

- [DevOps Useful Resources](#)
- [Datadog Glossary reference](#)
- [Miro visual reference of products](#)
- [Miro visual reference of Data Sources](#)
- [Miro visual reference of Data Visualisation](#)



O11y workshops

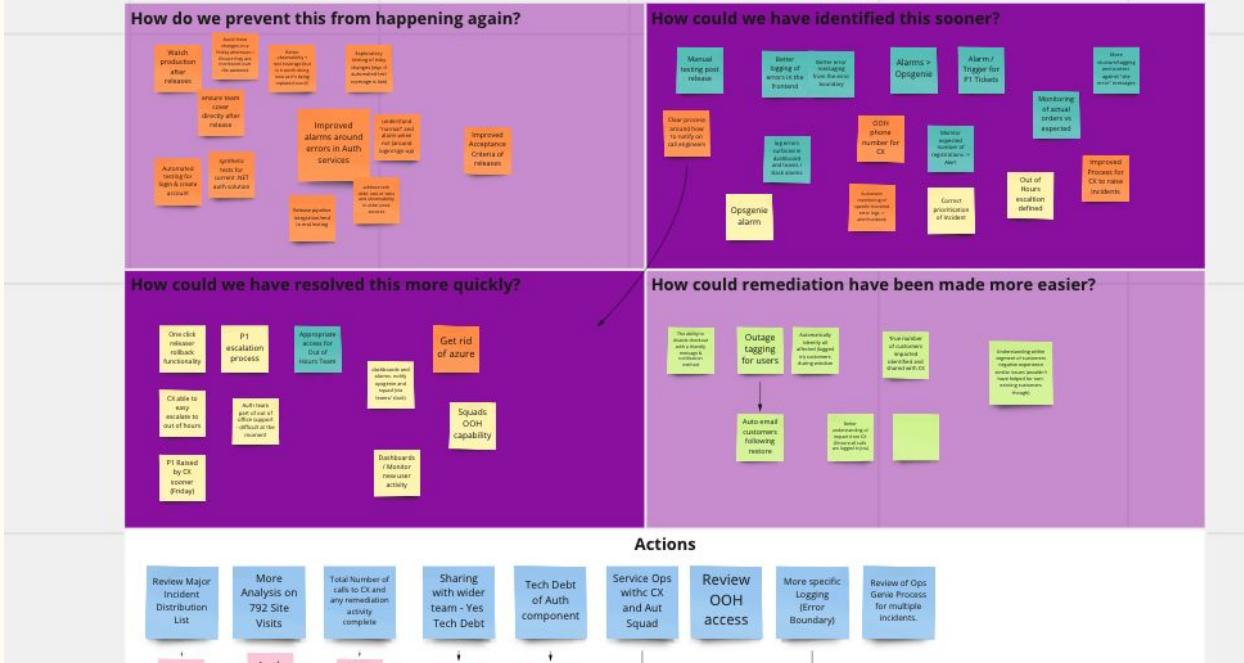
Dr. Ron Westrum's 'Three Cultures Model'

PATHOLOGICAL	BUREAUCRATIC	GENERATIVE
Power-oriented	Rule-oriented	Performance-oriented
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure = scapegoating	Failure = justice	Failure = inquiry
Novelty crushed	Novelty = problems	Novelty implemented



Psychological **safety**

10/11/12 - ST-1050 - Unable to create account



Incident reviews

Frame 1



Regular **Chaos Days**

Working Group API: O11y Blueprints



Created by Greg Farrow, with a template

Last updated: Jun 23, 2022 by Apostolis Apostolidis • 3 min read • 49 people viewed

[🕒 Recently updated](#) | [ℹ️ Overview](#) | [🤔 What is the problem?](#) | [🎯 Goals](#) | [📝 Kill Criteria](#) | [🚫 What is out of scope?](#) | [💬 How are we collaborating?](#) | [📣 Team news](#) | [📝 Relevant Documents](#) | [✖️ Limitations & Concerns](#) | [🔮 Future Work](#)

🕒 Recently updated

- [2022-03-10 | O11y Club | Day 17, Hunting for spans](#)
Mar 16, 2022 • contributed by Paul Richards
- [2022-03-04 | O11y Club | Day 16, The Chain](#)
Mar 07, 2022 • contributed by Greg Farrow
- [2022-02-03 | O11y Club | Day 10, 11 & 12 – First an apology](#)
Feb 03, 2022 • contributed by Greg Farrow
- [2022-01-14 | O11y Club | Day 9 – now streaming on cinch-flix: "RUM and the Redaction of the Personal Identifiable Information"](#)
Jan 19, 2022 • contributed by Paul Richards
- [2022-01-06 | O11y Club | Day 8 – Jest, Test and Seeding DynamoDB](#)
Jan 13, 2022 • contributed by Greg Farrow

[Show More](#)

	Name	O11y Blueprints
	Status	COMPLETE



Working Groups

☰ README.md

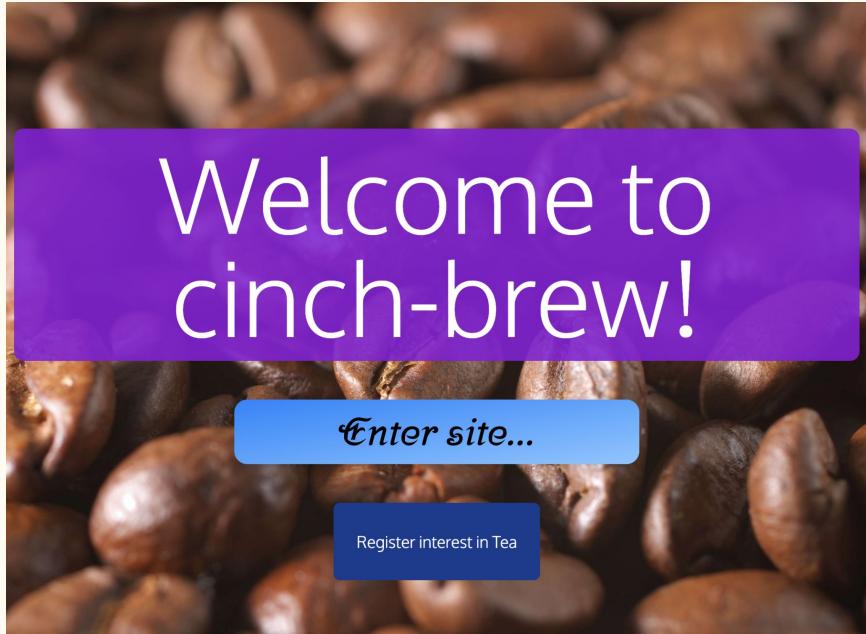
cinch-brew

Cinch brew is a fictional software system demonstrating good observability.

The system is designed with common cinch architectural patterns in mind - including backend and frontend.

An observability blueprint

An observability blueprint serves as the educational center for good observability patterns and practices used at cinch. Its purpose is to offer guidance by way of working examples of common scenarios you will come across whilst developing components.



Code **blueprints**

 Pick o11y tool & let everyone have it Instrument code Support & enable learning

What can
possibly go
wrong?



One person knows everything



One person knows everything



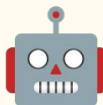
Nobody understands the dashboards



One person knows everything



Nobody understands the dashboards



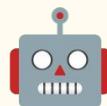
Overusing infrastructure-level metrics



One person knows everything



Nobody understands the dashboards



Overusing infrastructure-level metrics



Know the “what” but not the “why”

When do you
know you've
made it?

Team: o11y is a team responsibility

Team: o11y is a team responsibility

Individual: o11y practices become core skill

Team: o11y is a team responsibility

Individual: o11y practices become core skill

Org: all teams practice ODD*

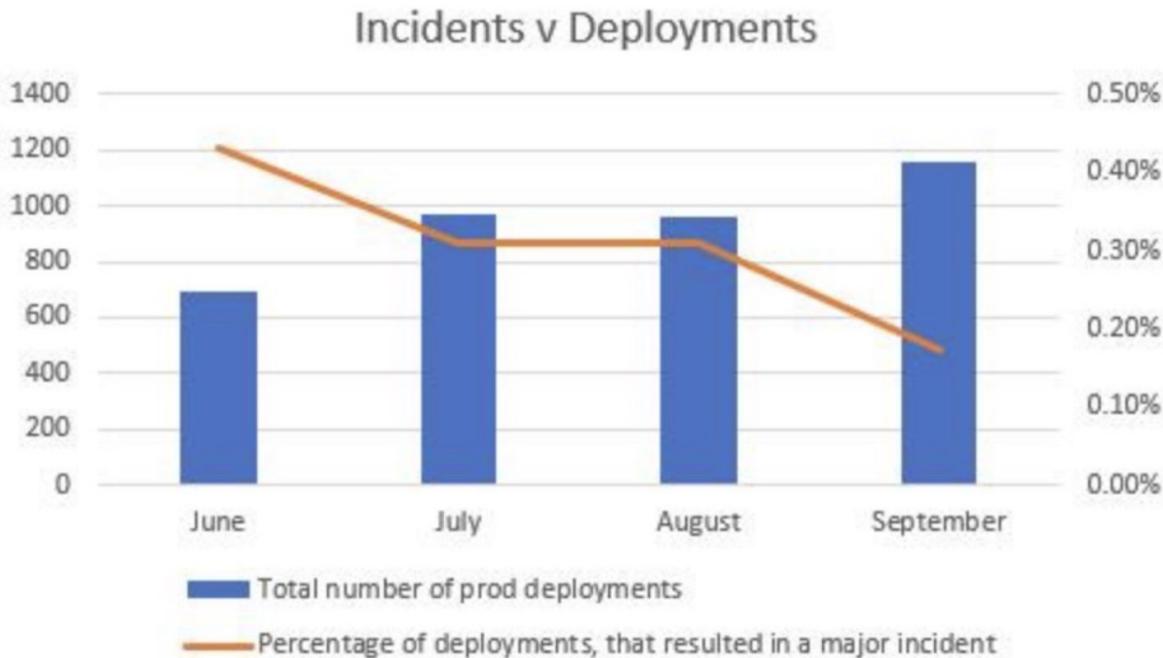
***Observability Driven Development**

So **what?**



Observability is **expensive**.

Incidents are **rare**.



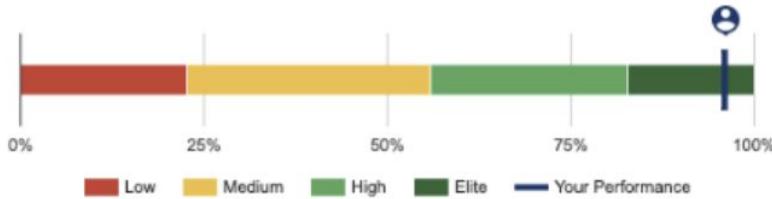
Mean time to recovery
is not an issue.

Your software delivery performance

Your performance:

Elite

You're performing better than 96% of [State of DevOps Survey](#) respondents. ⓘ



[Find a car](#)[Car finance](#)[Part exchange](#)[How cinch works](#)[News & reviews](#)[My profile](#)

Electric Deal!

Get up to **£2000** off selected electric cars plus **£500 charging**
on us with cinchCharge. T&Cs Apply.

[Shop cars on offer](#)

Select make

Select model

[Search all 6,501 cars](#)

Not sure where to start? Try our [Help Me Choose tool](#)



Excellent  Trustpilot

But this will **not** happen overnight.

Year 1

Instrument their code by default

Take ownership of their **alarms**

Have a good view of **business transactions**

Year 1

Own and evolve dashboards

Review dashboards during **stand-ups**

Review dashboards at the **end of the day**

Year 1

🔍 Search...

Saved Views

\$aws_account

██████████

\$env

prod

\$aws_account_id

██████████



\$environment

production



Are orders looking healthy?

23 widgets



Is our part exchange service healthy?

12 widgets



What changes have been released?

6 widgets



Do we have any incidents?

1 widget



Are we seeing errors?

5 widgets



Are my APIs healthy?

12 widgets

Have a **rhythm of o11y improvement** katas

Improve o11y via **incident review feedback**

Enable **non-tech roles** to use dashboards

Year 1

Year 2

Traces are used as the telemetry unit

Logs are used in the context of traces

Using **RUM** linked to back-end **traces**

Year 2

Other parts of the system are instrumented

Teams start creating and owning **SLOs**

An **o11y community** starts to emerge

Year 2

Year 3

o11y blueprints emerge

Self-sustaining o11y Community flourishes

Subject matter experts in o11y grow

Year 3

Engineers are **skilled o11y practitioners**

Non-techies know about o11y

All teams consider **o11y as a given**

Year 3

How is o11y
democratised?

The more **observable** our systems,
the more democratised our
understanding*

***of our software systems are across the org**

o11y

Business Analytics & Data Intelligence

Observability is **blurring** the lines.

Is o11y a **dev** thing?

BA friendly intro to Observability



I didn't realise this was a technical hack day

BAs can join in too. Sit down



Observability: A Lizard's way of thinking



Imagine this guy turns up to his first working day at cinch.

He's so drunk, he can't speak. We have a problem here. How did this happen?



Let's check the dashboard and the traces



We can identify via the contextual information in the receive text lambda that a text from "our kid" was received at 8pm last night. This text has subject "pub, just for one?"



The "leave house" lambda was invoked last night at 8:30 pm



We can see pub in the logs, which lasted 4 hours



The "Mojo" lambda was triggered at 1am. The alert for Jagerbombs went off at 1:05am.

YOU SEE

**OBSERVABILITY ISN'T
JUST FOR TECHNICAL PEOPLE**

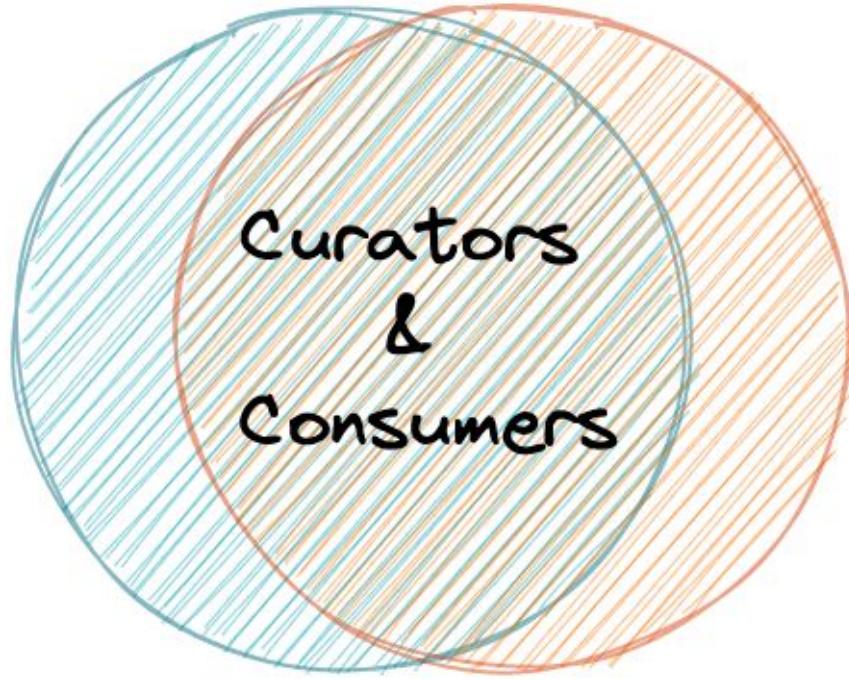
Thanks for listening!

Understanding the software is
only as good
as your telemetry data.



Software Engineers

Business Analysts



Software Engineers

Business Analysts

Just like **DevOps**.

“Devs” have taken **ownership** of the **quality** of the telemetry data and they **support** it.

Multiple Cards

Multiple Cards Selection

16.12%

Users selecting multiple cards

208

Select Multiple Cards => Verify Cards => Confirm & Pay

Below are the total orders that enter each stage of the flow.

Single Card

Single Card Selection

83.88%

Users selecting single card

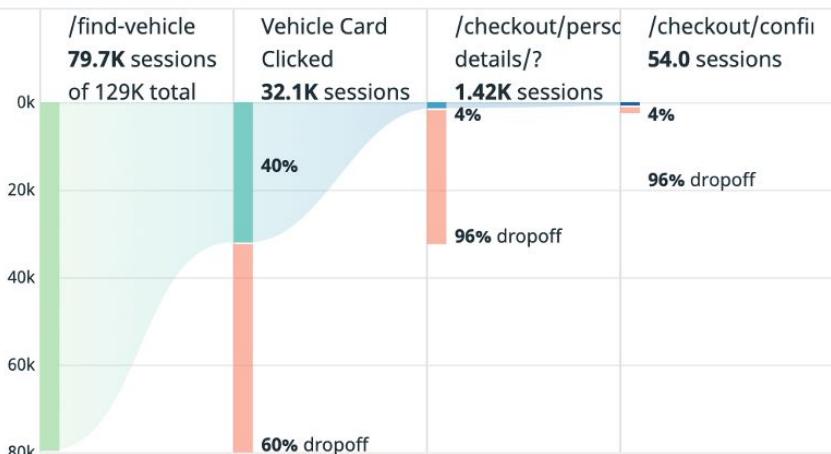
1.08k

Select Single Cards => Confirm & Pay

Below are the total orders that enter each stage of the flow.

Traffic At A Glance

Funnel Conversion



Model Searches

7.87k	Fiesta
7.32k	Golf
6.49k	A-Class
6.00k	Focus
5.23k	A3
4.83k	1 Series
4.11k	3 Series
3.98k	A1
3.85k	Polo

Search devices

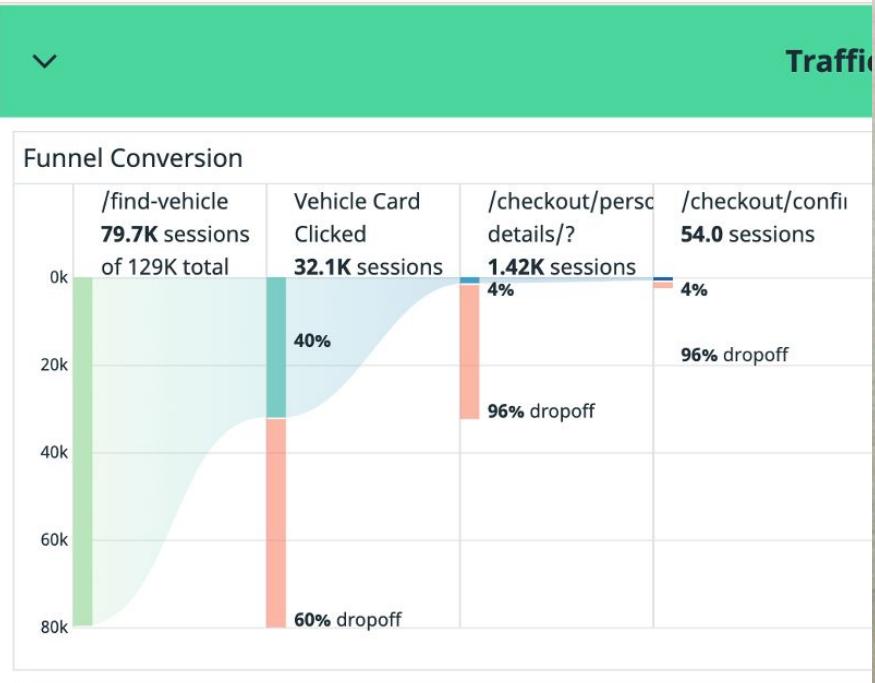
45.33k	Mobile
19.21k	Desktop
3.65k	Tablet
51.00	Bot
3.00	Appliance

Make Searches

33.88k	BMW
27.16k	Ford
25.99k	Audi
25.43k	Mercedes-Benz
22.70k	Volkswagen
12.08k	Land Rover
11.57k	KIA
11.57k	Vauxhall
9.48k	POD

Search Browsers

22.61k	Mobile Safari
16.30k	Chrome Mobile
10.28k	Chrome
4.66k	Safari
3.97k	Samsung Internet
3.91k	Edge
3.41k	Google
2.03k	Chrome Mobile iOS
1.87k	Firefox



Wrapping up



Lack of o11y is our
collective tech debt.

We're now fixing it.

“How do you
know your code
is **working in**
production?”



= 20+
teams
practicing
observability

Observability > Testability



Pick o11y tool & let **everyone** have it

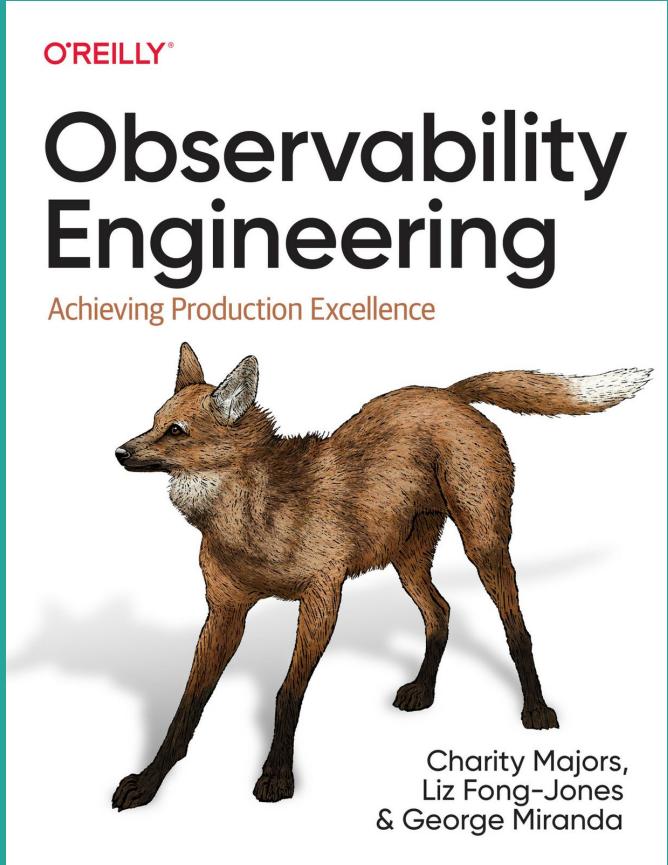


Instrument code



Support & enable **learning**

Read this book





← **Charity Majors**

59.3K Tweets

... Following

Charity Majors

@mipsytipsy

cofounder/CTO @honeycombi, co-author
Database Reliability Engineering. I test in pi

② San Francisco charity.wtf Joined

529 Following 77.8K Followers



CHARITY.WTF

OBSERVABILITY IS A MANY-SPLENDORED DEFINITION

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

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

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

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

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



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



olly
cast

There is **no singular and correct order or prescriptive way** of doing these things...

Charity Majors, Liz Fong-Jones, George Miranda.
Observability Engineering

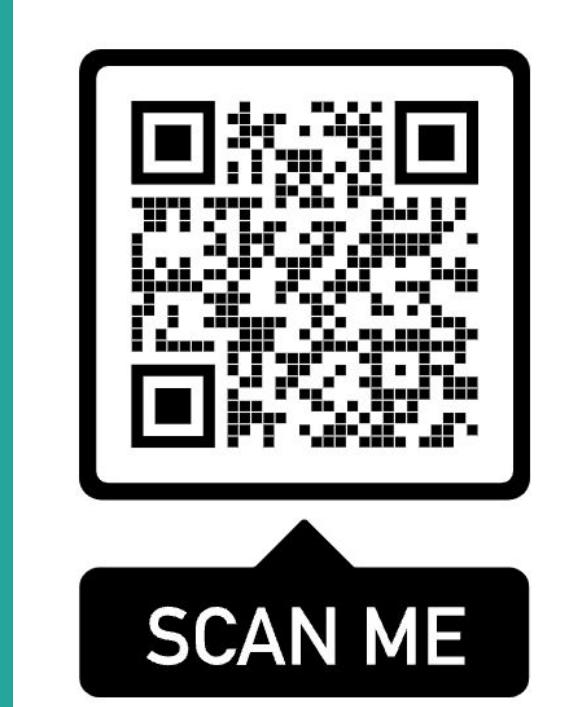
...at each step, **focus on what you're hoping to achieve.**

*Charity Majors, Liz Fong-Jones, George Miranda.
Observability Engineering*

Take observability **seriously**, make it work for your use case.

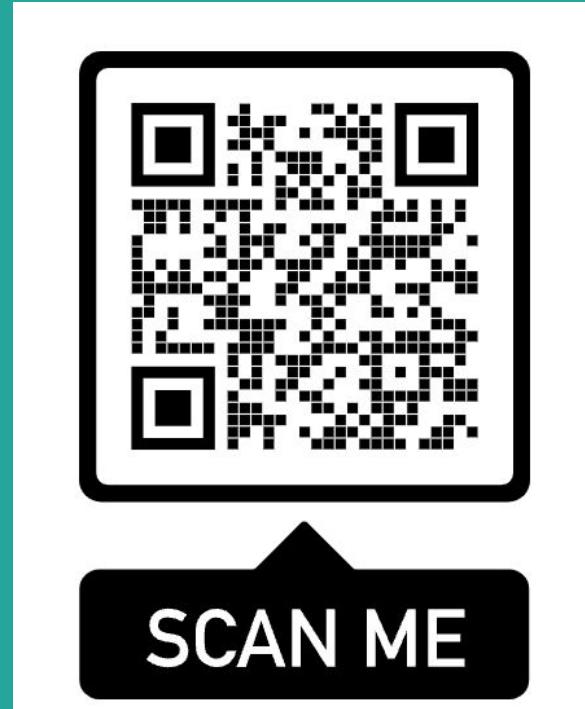
You will **amplify** and **democratise**
the **understanding** of the **behaviour**
of your software system

Thank you



@apostolis09

Questions?



@apostolis09