O11Y_DAY :SFO

Honeycomb Observability Day

Welcome

O11Y_DAY:SFO

Charity Majors

CTO and Co-founder | Honeycomb

KEYNOTE

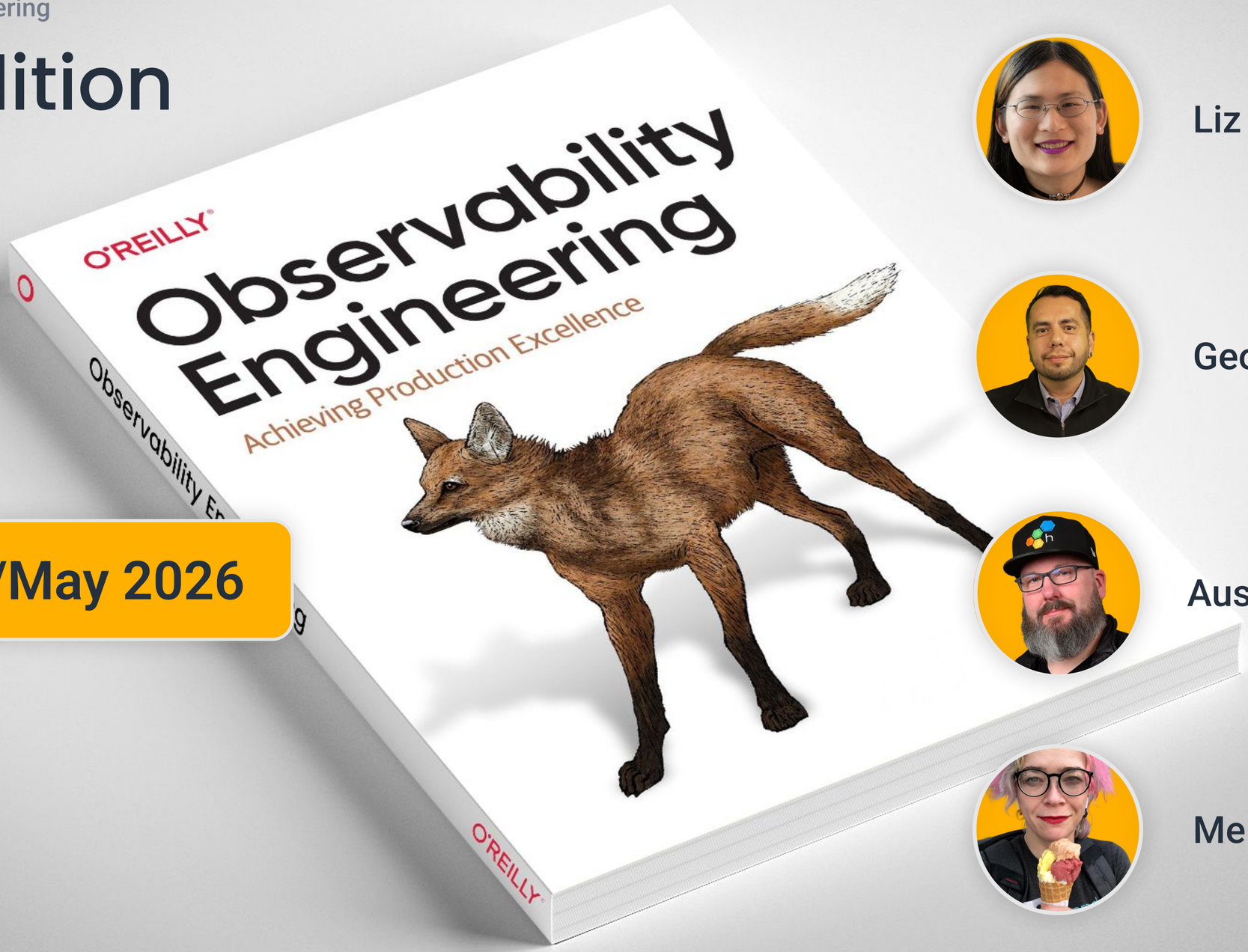
Observability Engineering

Scenes on Team, from the 2nd Edition

Charity Majors
CTO and Co-founder, Honeycomb

Observability Engineering

2nd Edition



April/May 2026

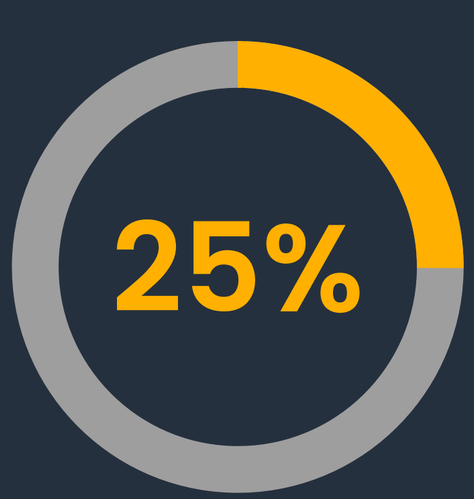
Liz Fong-Jones

George Miranda

Austin Parker

Me

Observability Teams



Cloud Spend on Observability

> There is no other team in the engineering org that can compete with observability, in terms of impact to your business and the bottom line.
>
> **Not even close.**

— Charity Majors | CTO and Co-founder

Not well understood

Engineering leadership, ICs, Managers, VPs, CTOs still **struggle with observability**

I get it now!

I didn’t put it all together until this edition of the book

Few teams

Operate at or near this level of peak potential

Some teams do!

We’ll get back to this point later

Prediction

By the 3rd edition, this will be common knowledge—AI is making this inevitable

What do execs think about observability?

It costs too much!!!

They're right.

In 2015

Everyone was paying **monitoring prices** for **monitoring results**

In 2025

Most are paying **observability prices** for **monitoring results**

Observability costs more.

2009

$50k

per year

2024

$24M

per year

Monitoring

- Operational tool
- Operational outcomes

Observability

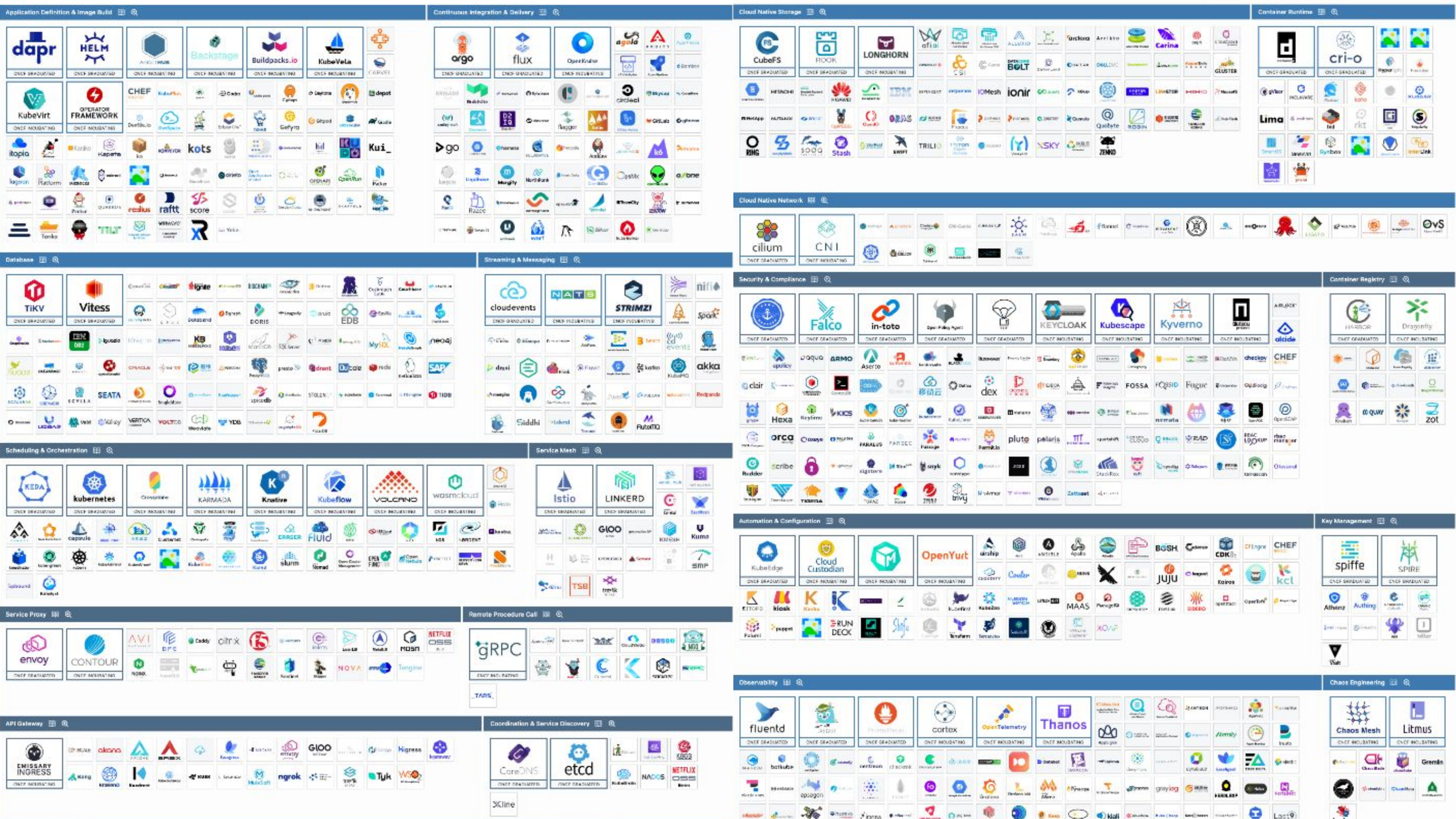
- The sense-making functionality of complex sociotechnical systems

Observability does more. Or it should.

2009

- Linux
- Apache
- MySQL
- Php_perl_python

2024


Application Definition & Image Build
dapr
HELM
Backstage
Buildpacks.io
KubeVela
KubeVirt
OPERATOR FRAMEWORK
CHEF
kots
Kui_
radius
rafttt
score
Continuous Integration & Delivery
argo
flux
OpenKruise
go
Cloud Native Storage
CubeFS
HOOK
LONGHORN
Container Runtime
cri-o
Lima
Cloud Native Network
cilium
CNI
Database
TiKV
Vitess
Streaming & Messaging
cloudevents
NATS
STRIMZI
Security & Compliance
Falco
in-toto
KEYCLOAK
Kubescape
Kyverno
Container Registry
HARBOR
Dragonfly
Scheduling & Orchestration
KEDA
kubernetes
KARMADA
Knative
Kubeflow
VOLCANO
Service Mesh
Istio
LINKERD
Automation & Configuration
KubeEdge
Cloud Custodian
OpenYurt
Key Management
spiffe
SPIRE
Service Proxy
envoy
CONTOUR
Remote Procedure Call
gRPC
Observability
fluentd
Thanos
cortex
OpenTelemetry
Chaos Engineering
Chaos Mesh
Litmus
API Gateway
EMISSARY INGRESS
Coordination & Service Discovery
CoreDNS
etcd

2016 definition

Monitoring / Logging

- Monitors
- Logs

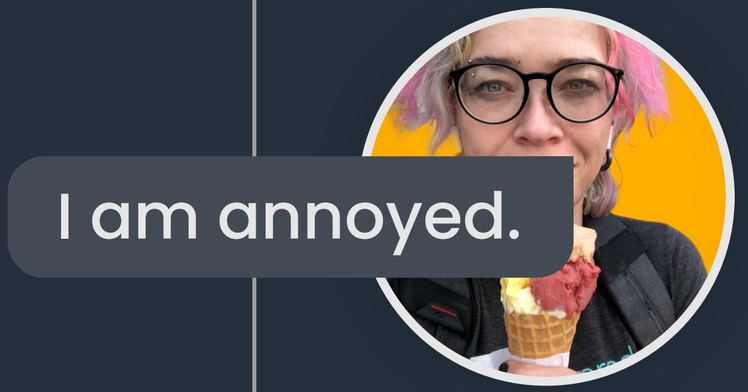
Observability

We *tried* to give it a specific technical definition (We failed)

2020

Observability

“**Anything** that has anything to do with **software** and **telemetry**”



The Reality

Monitoring / Logging

- Ops tool
- Operational outcomes (up, down, slow)

Observability

- The sense-making functionality of sociotechnical systems.

The Reality

Execs would not be complaining about the price tag if they were getting results to match.

Cloud Spend on Observability

10x

Faster Delivery

Smaller Team Size

10x

Customer Satisfaction

Observability engineering teams are the single greatest point of leverage in any engineering organization.

Or they can be.

They should be.

Systems thinking shows us **why.**

The core of every high-performing engineering org is **fast feedback loops.**

And observability is the feedback loop of feedback loops, the sense-making apparatus for complex systems.

Finance metaphor

Observability engineering teams are the single greatest point of leverage in any engineering organization.

Or they can be. **They should be.**

The path to getting there is **not complicated**.

(It is not *easy*. But it's not complicated.)

The 2010s

Technologies managed by Ops

DNS	Web Servers	File Storage
SMTP/IMAP	LDAP	Databases
Monitoring	OSes	Caches
Kerberos	File systems	Proxies

The 2020s

Monitoring

Observability

A systems approach to observability

"Give me a lever long enough and a fulcrum on which to place it, and I shall move the world"

— Archimedes

First

Run observability like a platform engineering team

Not like infrastructure

Platform principles

a. Build like a **product**, own as little code as possible
b. Focus on developer experience and **fast feedback loops**
c. Applying **design principles** and **user research**
d. Make it simple and easy to do the right thing on **autopilot**
e. Oriented around the **needs of the business**
f. Staffed by engineers who understand **devex**

Most of all:

Limit the cognitive bandwidth demands on developers

Second

Manage your observability like an investment

Not a cost center

You can’t make more money by spending more on infrastructure.

Observability is different.

a. Spending more on your observability team & tools can yield real, sizable **returns on investment**. It CAN be one of the most efficient, high performing investments you can possibly make.
b. The key here is understanding the **dual mandate of observability**, and mapping it back to your company’s unique **goals, products and objectives**

The dual mandate of observability

External

Maps to customer happiness

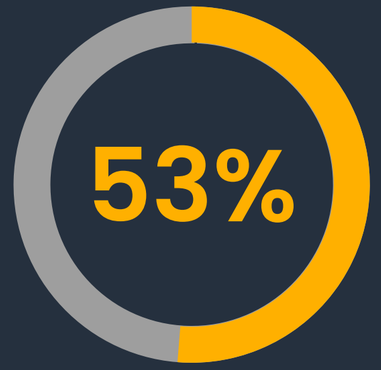
Internal

Maps to developer experience and your ability to move swiftly, with confidence

External

Customer happiness → Revenue

a. Observability plays a critical role in every single customer experience.
b. Responsive UX, fast queries, end to end performance tuning
c. Find problems before your users do



Site visits abandoned >3s load

Users find slow sites more annoying than down sites

100ms = 1%

Page speed increase boosts revenue by 1%

All data pulled from https://www.webuters.com/the-high-cost-of-slow-load-times-in-e-commerce

External

Customer happiness → Revenue

a. Run more experiments
b. Build better products
c. Try more things
d. How are your users **actually using** what you built?
e. Be an opportunist

Internal

Developer experience → Business value

It’s about your ability to move swiftly, with confidence,
as a business.

Internal

Developer experience → Business value

a. Shipping is your company's heartbeat
b. Too many are in "software engineering death spiral"

1. How fast are your deploys?
2. How long to run tests?
3. How long to ship a single line of code?
4. How long to understand its impact on a single random user?

Internal

Developer experience → Business value

a. The ceiling has gone WAY UP over the past decade in terms of how fast you should be able to develop and ship code
b. In 2015, you didn't have access to a high quality developer tool chain unless you were behind the walled gardens of big tech. Now you do.
c. But these tools are more effective as a one-two punch:
    1. Modern observability AND feature flags
    2. Modern observability AND progressive deployments
    3. Modern observability AND canarying

Honestly, that's **basically** it

- Run your observability teams like **platform engineering**
- Manage your observability tools like **an investment**

a. Roll up to software engineering, not infra or IT
b. Mind the dual mandate
c. Build for fast feedback loops and minimize cognitive overhead

Change is hard.

But all our AI investments are riding on this…

a. “I think developer experience is not an outcome of using AI tools really well. It's a prerequisite to use them well. **What's good for an individual human developer is also good for an agent**. There's extra stuff that you might want to do for agents, but the physics of what makes software or codebases easy to contribute to, easy to change...that's important for AI agents and for humans.

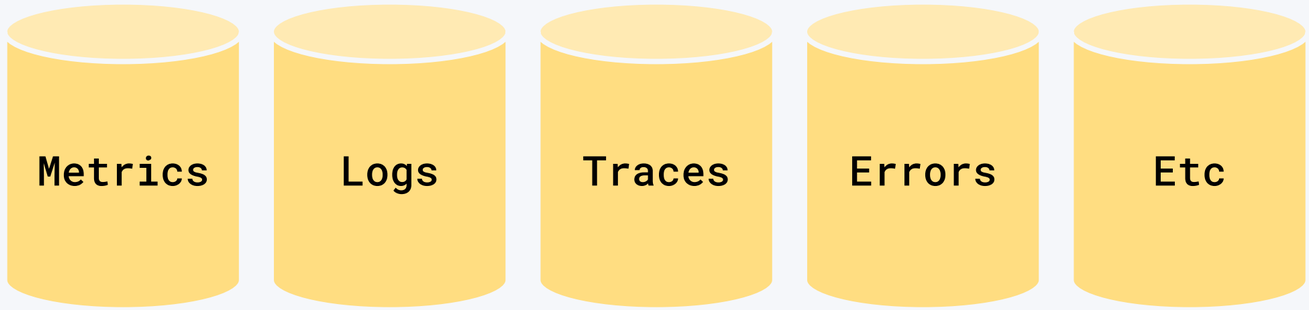
b. “**AI is an amplifier.** If you have parts of your system that are a bit garbage, they're going to be amplified garbage now. And if you have really solid engineering practices that have stayed ahead of the industry curve, then that's going to be amplified and you're going to get even better results.”

Laura Tacho
https://www.heavybit.com/library/article/ai-productivity-for-engineering-teams

Multiple Pillars Doom Engineering Orgs

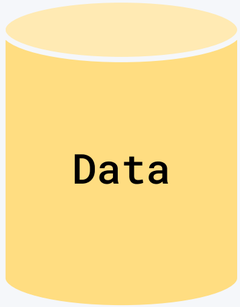
Multiple pillars model (aka “three pillars”, aka “observability 1.0”)

a. Metrics -> time series db
b. Logs -> log aggregator
c. Traces -> tracing tool
d. Exceptions, errors -> exception tracker
e. (etc)



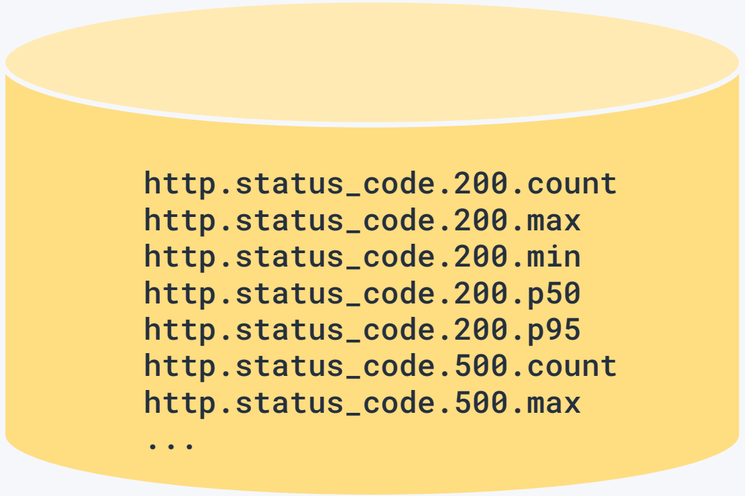
Unified storage model (aka “observability 2.0”)

a. Takes structured data
b. Stores it once
c. No dead ends



Three Pillars Cognitive Overload 🤯

Each metric is its own cardinality factory



"What is the type definition?"
"Is it a gauge? A rate? A count? Histogram?"
"When does it reset?"
"How do you relate them together?"
"How much are these going to cost us?"

Logs = gobs of semi-related data may be structured

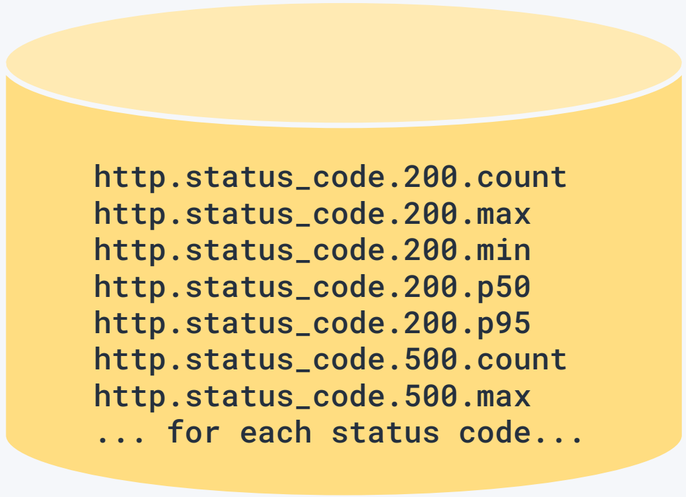
```
2025-08-09T12:45 INFO Customer Created
2025-08-09T12:50 TRACE { "tx_id": 123, "op": "debit",
                         "status", "success" }
2025-08-09T12:55 TRACE { "trans_id": 2343, "operation": "credit" }
2025-08-09T12:50 TRACE { "tx_id": 123, "op": "debit",
                         "status", "success" }
2025-08-09T12:55 TRACE { "trans_id": 2343, "operation": "credit" }
2025-08-09-12:56 ERROR Transaction rollback, Stack Trace...
```

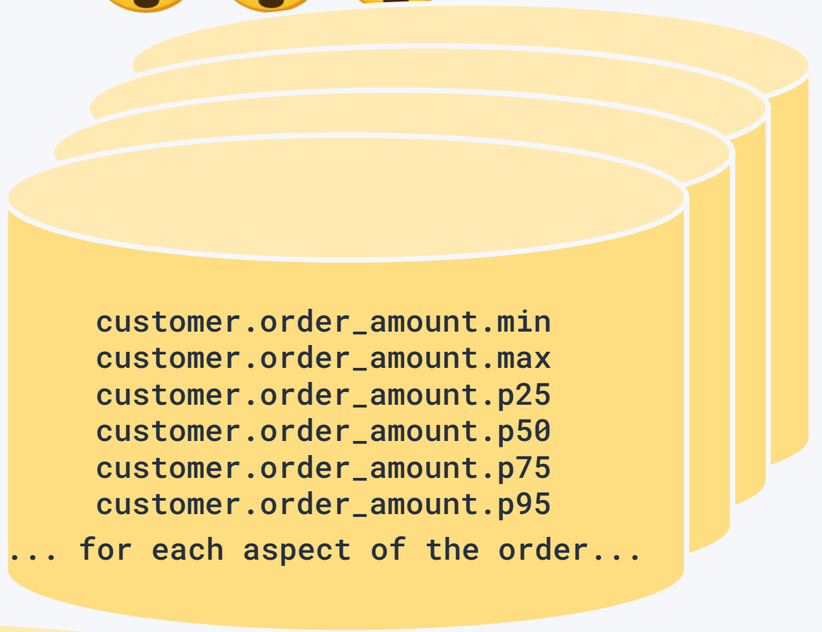
"Did everyone structure the logs the same way?"
"How do changes to log records impact analysis?"
"Do the attributes match across log records?"
"How would I correlate things together in a systematic way?"

... and how do I relate these pillars of data?

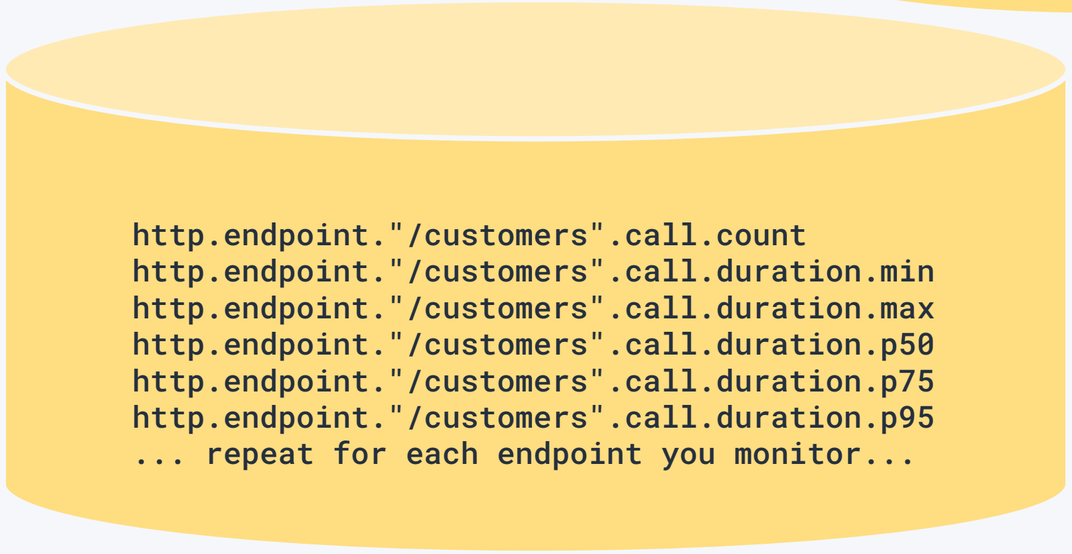
Three Pillars Cognitive Overload 🤯🤯🤯

Lots of metrics = Cardinality Explosion!





Each metric requires its own set of discrete keys to represent the measurements for each individual represented value.



This could be one structured data blob

```
{
   "name": "customer-order",
   "http.endpoint": "/customers/234/order",
   "http.method": "POST",
   "http.status_code": 200,
   "duration_ms": 3000,
   "customer.order_amount": 3040.20,
   "db.transaction.id": 2342345234,
   "db.operation": "credit",
   "db.query": "INSERT INTO ORDERS(customer_id, cart_id...) VALUES (?, ?, ?)"
}
```

More attributes? Add them!

This could be a trace span or just a plain old log, it's just an EVENT!

See the actual transactions!

Aggregate over the events to see metric-like trends (avg, p95, etc)!

Multiple Pillars Doom Engineering Orgs

Multiple pillars model (aka “three pillars”, aka “observability 1.0”)

a. Metrics -> time series db
b. Logs -> log aggregator
c. Traces -> tracing tool
d. Exceptions, errors -> exception tracker

- Gartner says most customers use 10-20 tools
- Therefore, cost multiplier is 10-20x at baseline
- Cardinality a constant battle
- Takes deep expertise and intuition to leap between datasets, since the relationships are not stored
- Cognitive overhead is massive
- 🔥🔥🔥 **Not a feedback loop** 🔥🔥🔥

Multiple Pillars Doom Engineering Orgs

Unified storage model (aka “observability 2.0”)

a. Takes structured data
b. Stores it once
c. No dead ends

- Cost multiplier is 1x
- High cardinality baked in (translation: “infinite custom metrics for free”)
- Every software engineer knows how to handle structured data

The Multiple Pillars are Doomed

a. The last observability company to be founded on the “three pillars” model was Chronosphere, in 2019
b. Every observability startup founded since then has used the singular storage (“o11y 2.0”) model we sketched out in 2016
c. I believe that on a ~5 year timeline, the industry will be moving towards a data lakehouse model (so do analysts, it seems)

Main blocker is chicken/egg

Everyone’s using the same terms to describe their product.
If you haven’t seen the difference, it’s hard to see.

We have to **show people**.

If all you need is an ops tool for operational outcomes, do monitoring.

In 2015

Everyone was paying **monitoring prices** for **monitoring results**

In 2025

People are paying **observability prices** for **monitoring results**

Success in observability?

Investment

Treat observability like an investment rather than a cost center

Platform team

Build your observability practice like a product team

Tools + enablement

Give your teams what they need to achieve their goals

and...

Spark Joy!!!

O11Y_DAY:SFO

Charity Majors

linkedin.com/in/charity-majors/

The dual mandate of observability

External: customer happiness -> revenue

a. Run more experiments
b. Build better products
c. Try more things
d. How are your users **actually using** what you built?
e. Be an opportunist

The dual mandate of observability

Internal -> developer experience

Developer experience isn’t (just) about developers

It’s about your ability to move swiftly, with confidence, as a business.

The dual mandate of observability

Internal -> developer experience

a. Shipping is your company's heartbeat
b. Too many are in "software engineering death spiral"

1. How fast are your deploys?
2. How long to run tests?
3. How long to ship a single line of code?
4. How long to understand its impact on a single random user?

The dual mandate of observability

Internal -> developer experience

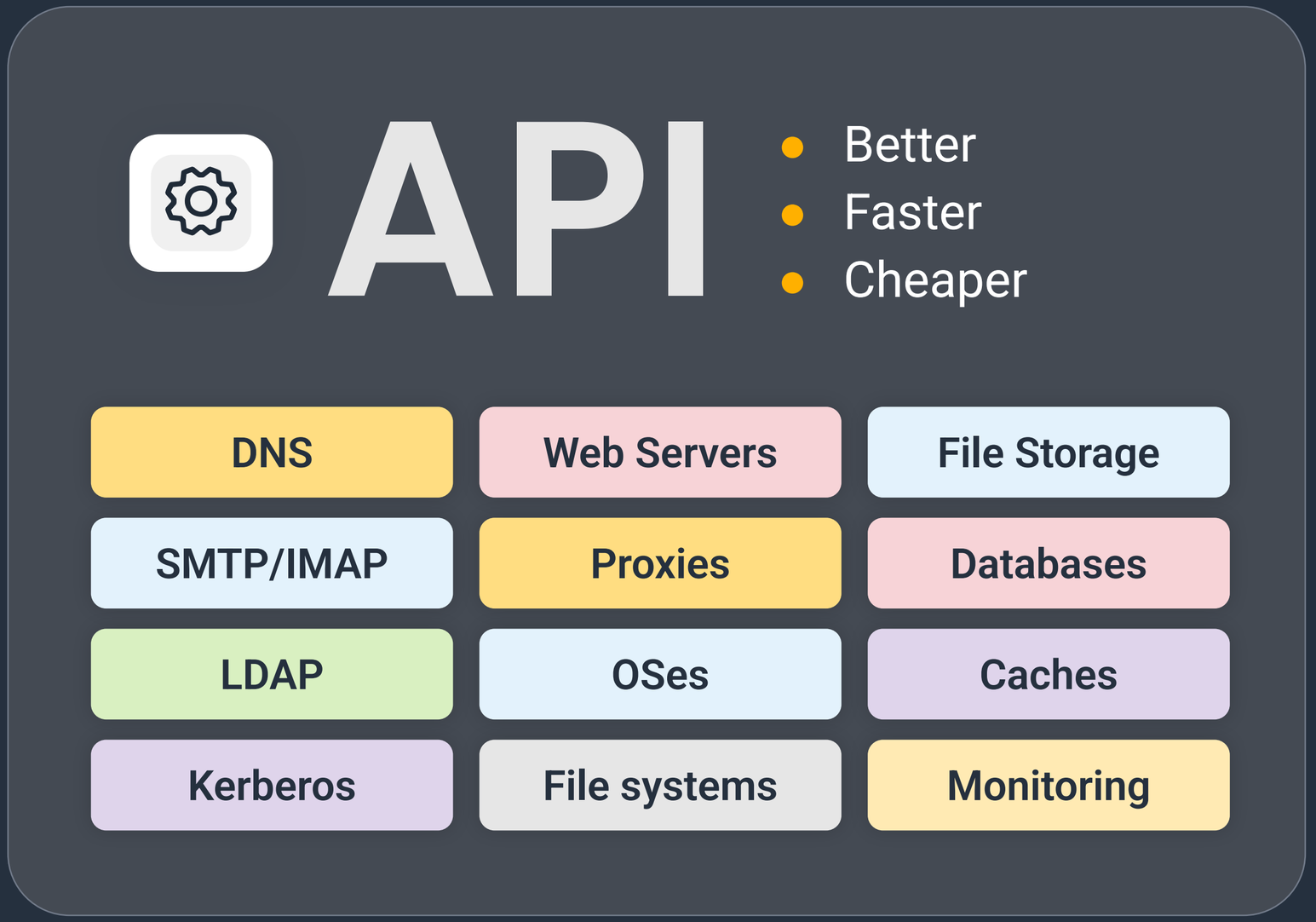
a. The ceiling has gone WAY UP over the past decade in terms of how fast you should be able to develop and ship code
b. In 2015, you didn't have access to a high quality developer tool chain unless you were behind the walled gardens of big tech. Now you do.
c. But these tools are more effective as a one-two punch:
    a. Modern observability AND feature flags
    b. Modern observability AND progressive deployments
    c. Modern observability AND canarying

The 2020s

Observability Team

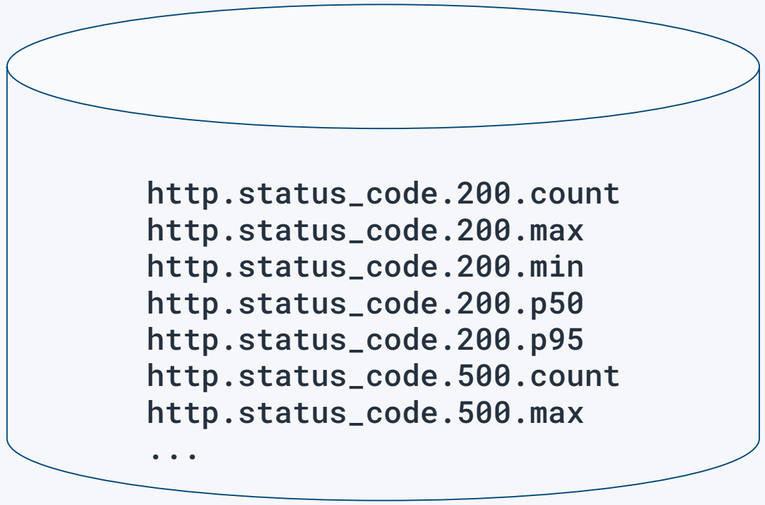
Some just renamed teams

- Ops > DevOps > Platform
- Monitoring / Logging > Observability
- Still using 10yr old tools
- Same old skill sets
- This won't yield results you seek…



Three Pillars Cognitive Overload!

Each Metric is its own cardinality factory



"What is the type definition?"
"Is it a gauge? A rate? A count? Histogram?"
"When does it reset?"
"How do you relate them together?"
"How much are these going to cost us?"

Logs = gobs of semi-related data, may be structured

```
2025-08-09T12:45 INFO Customer Created
2025-08-09T12:50 TRACE { "tx_id": 123, "op": "debit",
                         "status", "success" }
2025-08-09T12:55 TRACE { "trans_id": 2343, "operation": "credit" }
2025-08-09T12:50 TRACE { "tx_id": 123, "op": "debit",
                         "status", "success" }
2025-08-09T12:55 TRACE { "trans_id": 2343, "operation": "credit" }
2025-08-09-12:56 ERROR Transaction rollback, Stack Trace...
```

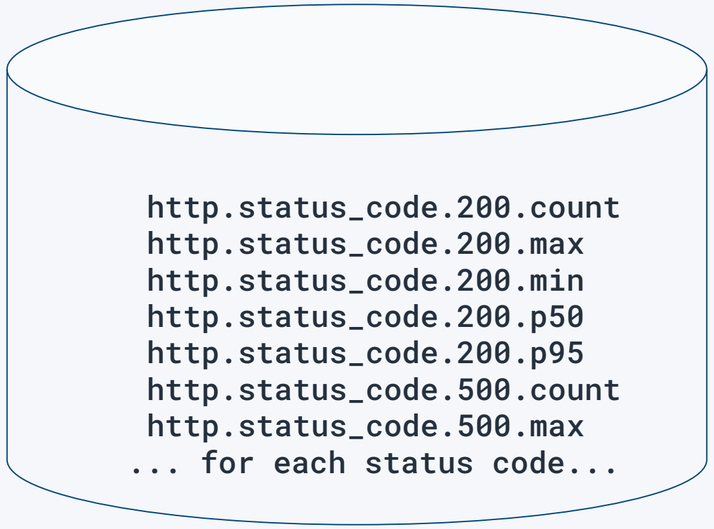
"Did everyone structure the logs the same way?"
"How do changes to log records impact analysis?"
"Do the attributes match across log records?"
"How would I correlate things together in a systematic way?"

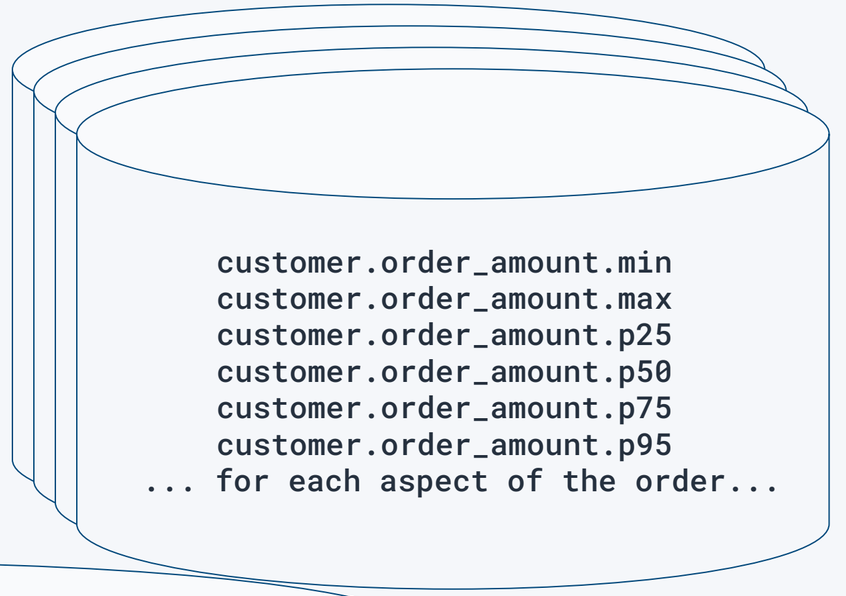
... and how do I relate these pillars of data?

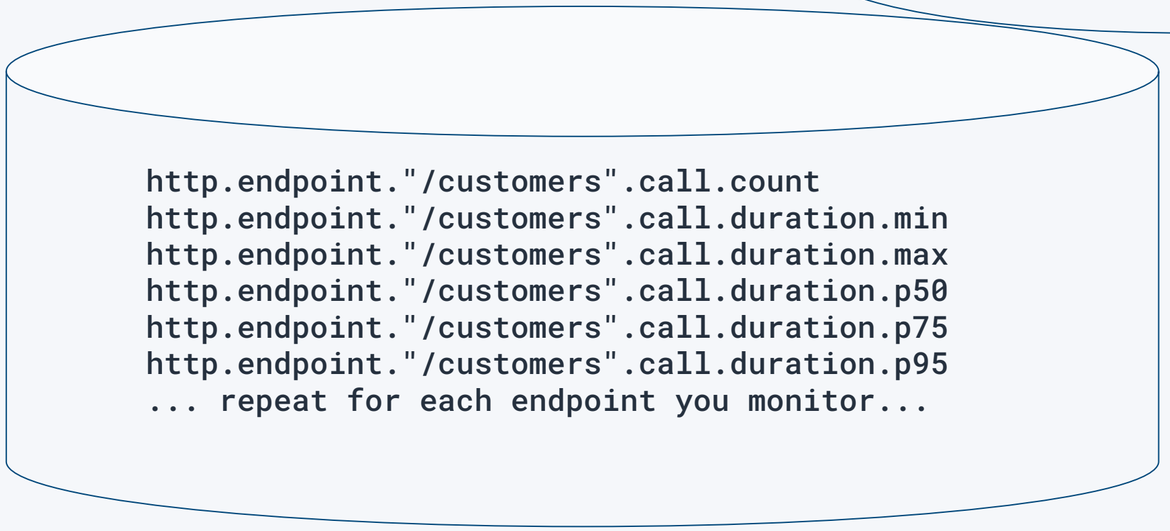
Three Pillars Cognitive Overload!

Lots of metrics =

Cardinality Explosion!







Each metric requires its own set of discrete keys to represent the measurements for each individual represented value.