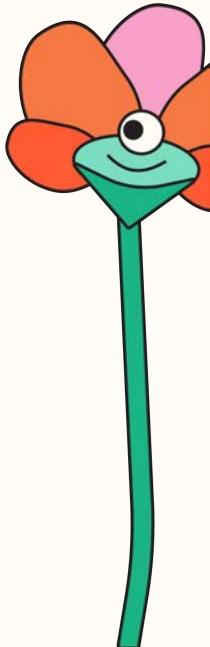
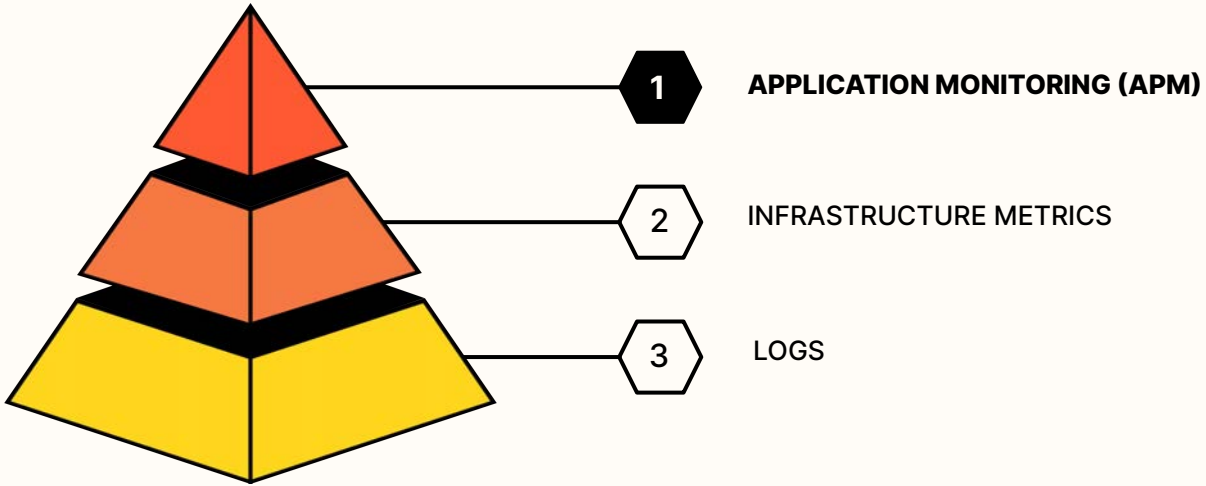
groundcover

Cloud-Native Observability

True Kubernetes Observability with eBPF

Conf42 Cloud Native March 2023

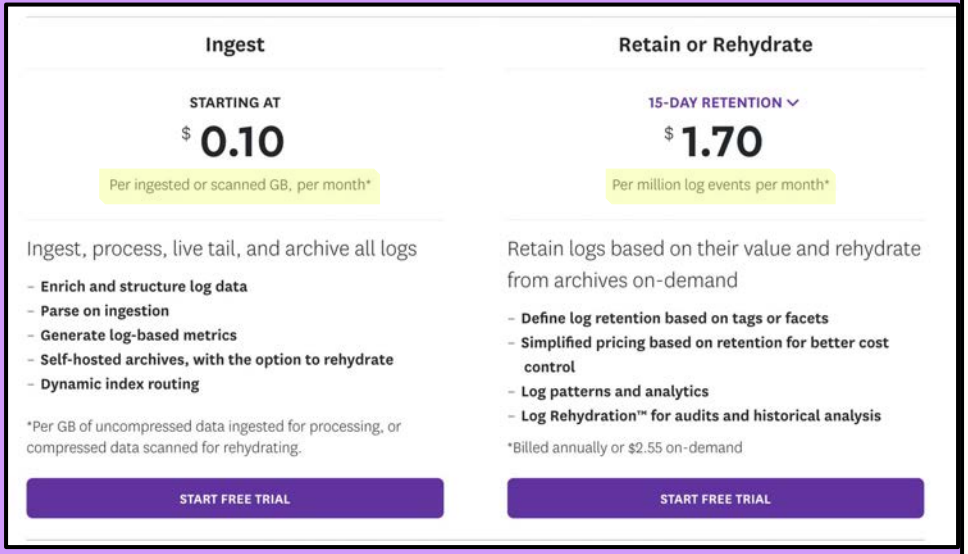


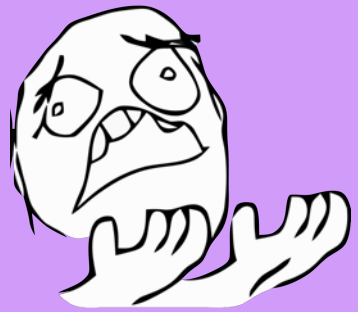


Observability is a Core Competency of any Team

O11y Spend are 20-30% of Infra Spend (!)

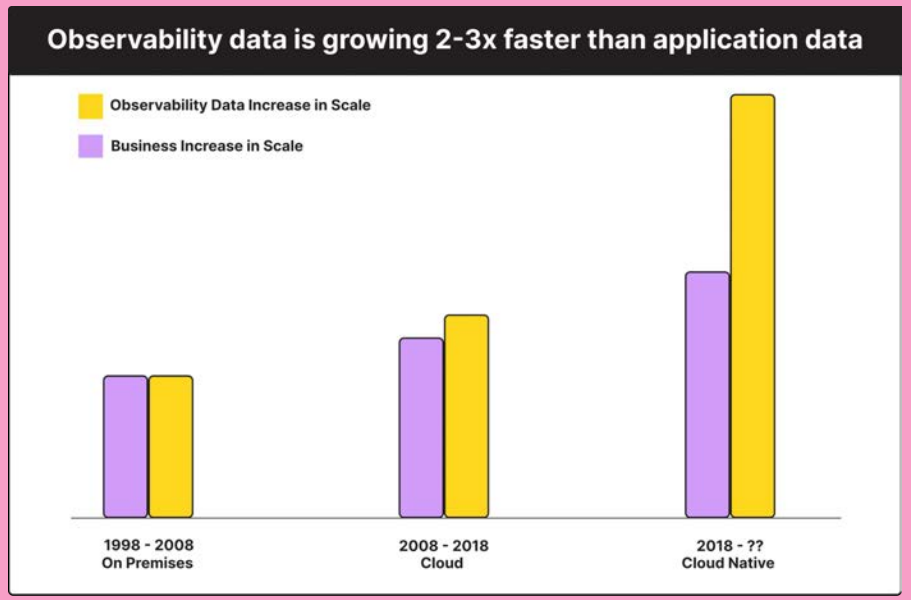
Charity Majors, Honeycomb.io





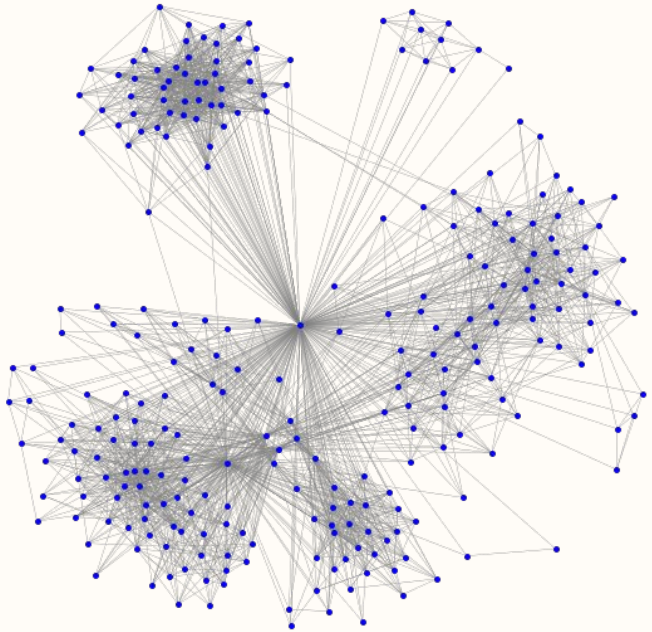
But mostly, Unpredictable!

Datadog Pricing, 2023



Cloud-native Made Things Worse

Cloud Native Monitoring, O'REILLY

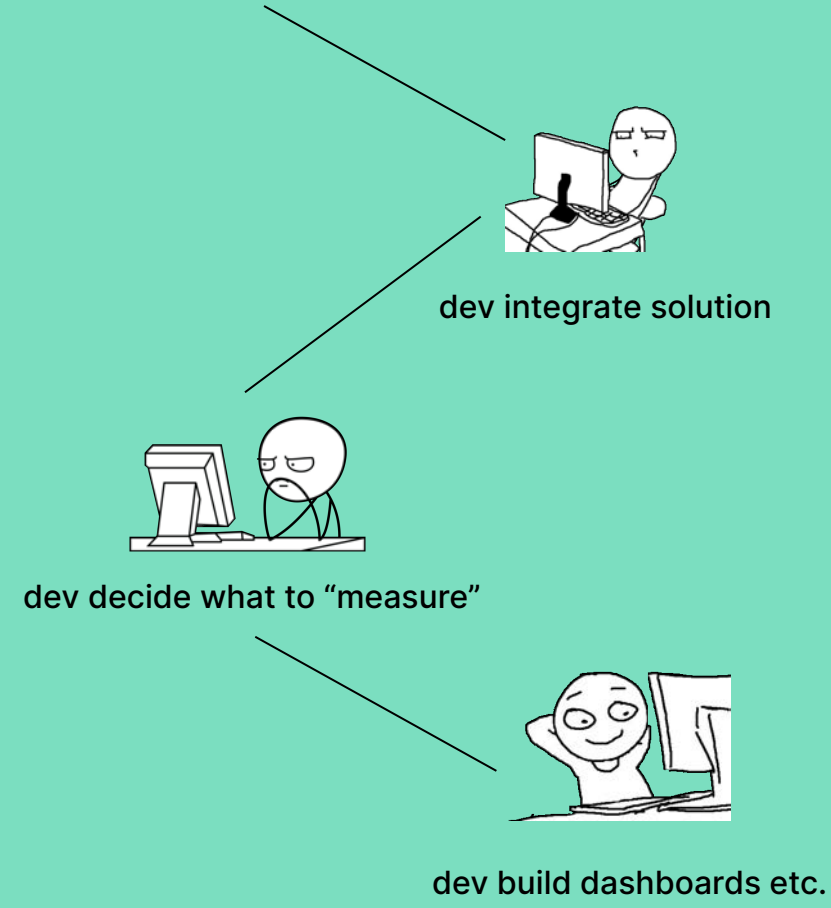


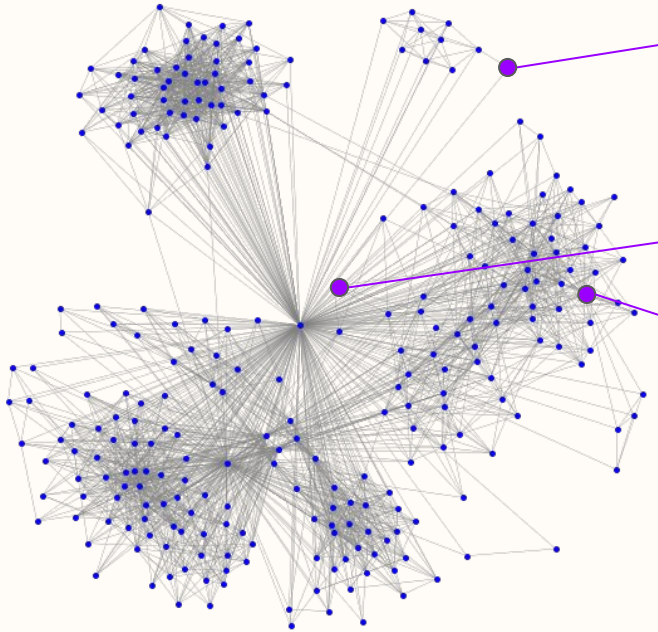
“Most companies have 184 microservices on average”

Survey by Kong, 2022

Observability was built

to be part of a dev cycle

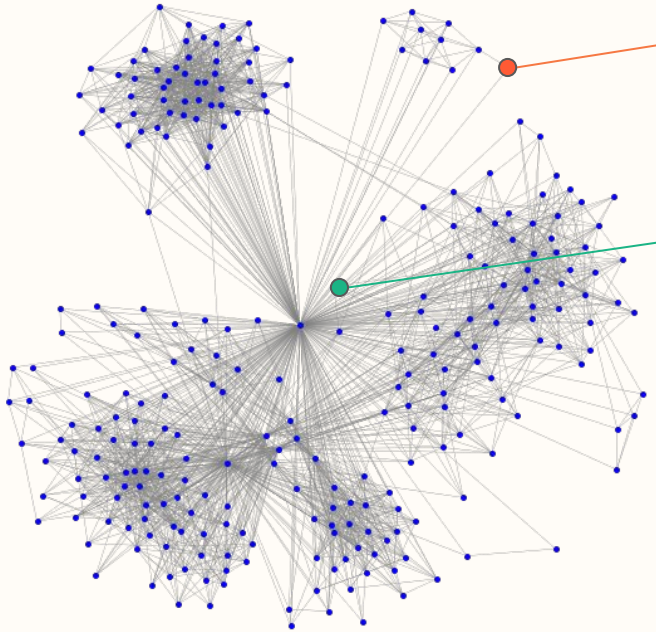




data science team
Python

web team
Node.js

backend team
Go

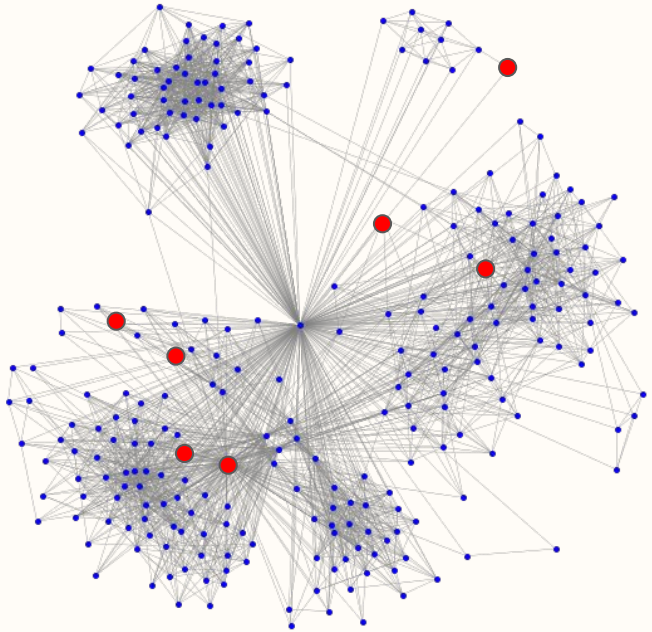


data science team
Python

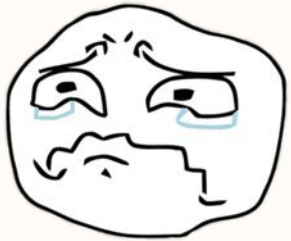
web team
Node.js

Who's asking the question

"Is all this data needed?"



Worried DevOps / SRE



Are we giving who's responsible

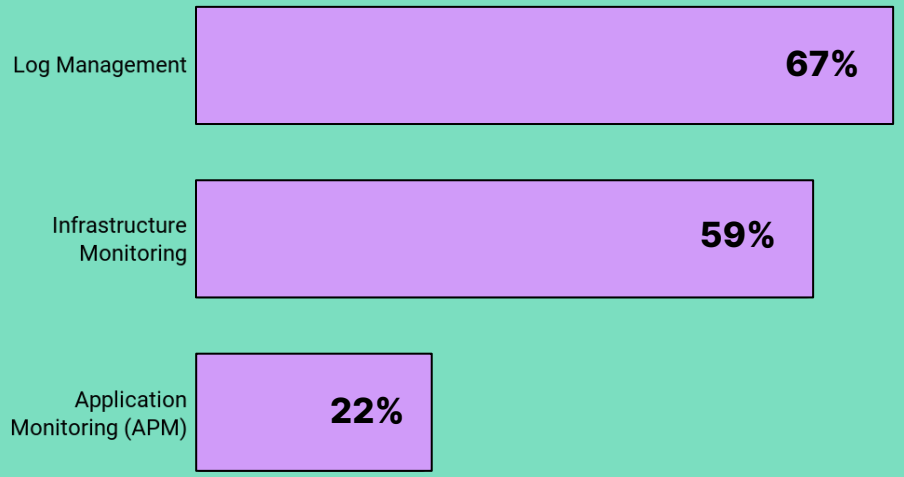
the Tools to Succeed?

→ get 100% coverage?

→ ensure cost - visibility trade-off

- Unscalable **Pricing** Models
- Harder Org **Alignment**
- Data **Privacy**

Legacy Observability for Cloud-Native



O11y is Under-Adopted

Devops Pulse, 2022

That's why we've built groundcover.

Data is collected
out-of-band with eBPF

Data is digested
distributedly where it lies

In-cloud architecture where
All data is kept private

O11y is hard to integrate.

Requiring lots of R&D effort around code instrumentations, and complex coordination between teams

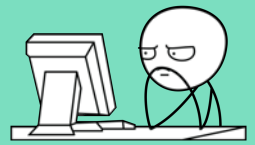
Auto-instrumentation...

```go
// Run starts polling users for Fibonacci number requests and writes results.
func (a *App) Run(ctx context.Context) error {
    for {
        // Each execution of the run loop, we should get a new "root" span and context.
        newCtx, span := otel.Tracer(name).Start(ctx, "Run")

        n, err := a.Poll(newCtx)
        if err != nil {
            span.End()
            return err
        }

        a.Write(newCtx, n)
        span.End()
    }
}
```

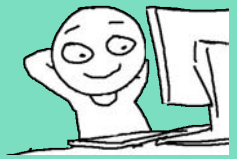
Back to the

the dev cycle

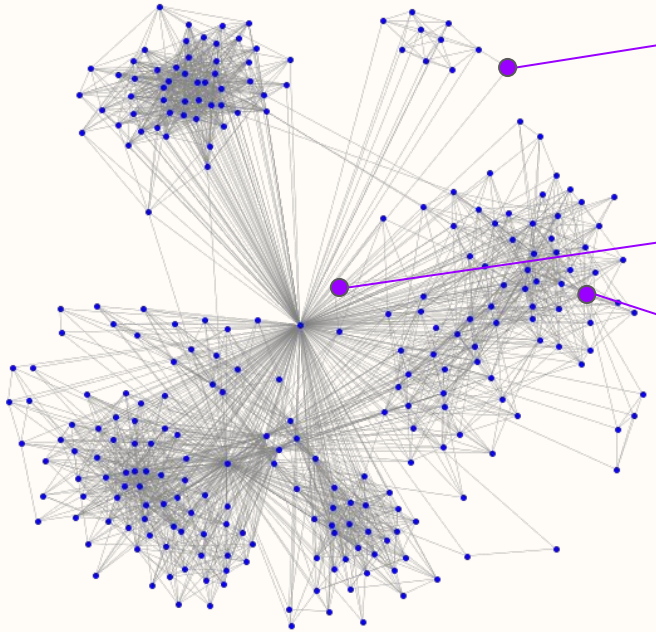
Instrument code...
Web team (Node.js)



Wait for version release...



Get value in production (!)

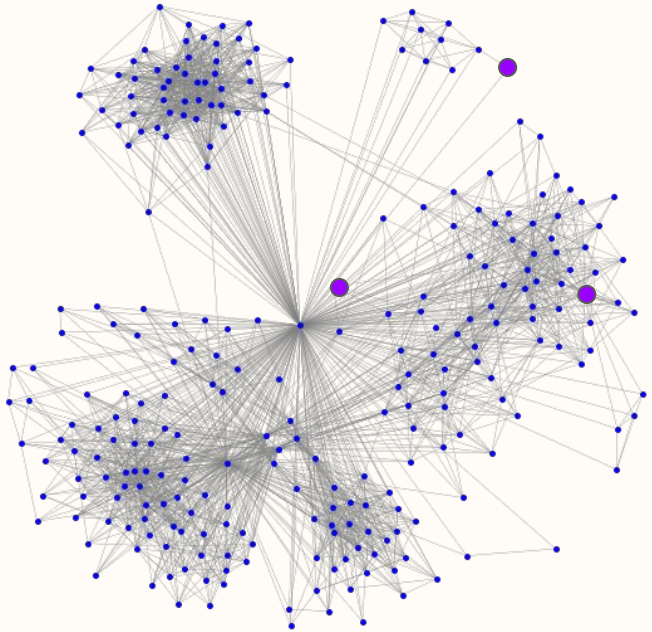


data science team
Python

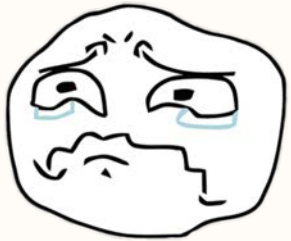
web team
Node.js

backend team
Go

Now try this in a **real company**



Worried DevOps / SRE



Are we giving who's responsible

for aligning all these teams?

→ to one uniform approach

→ to a high professional standard

> **eBPF does to Linux what JavaScript does to HTML…**with eBPF, instead of a fixed kernel, you can now write mini **programs…which are run in a safe virtual machine in the kernel.**”

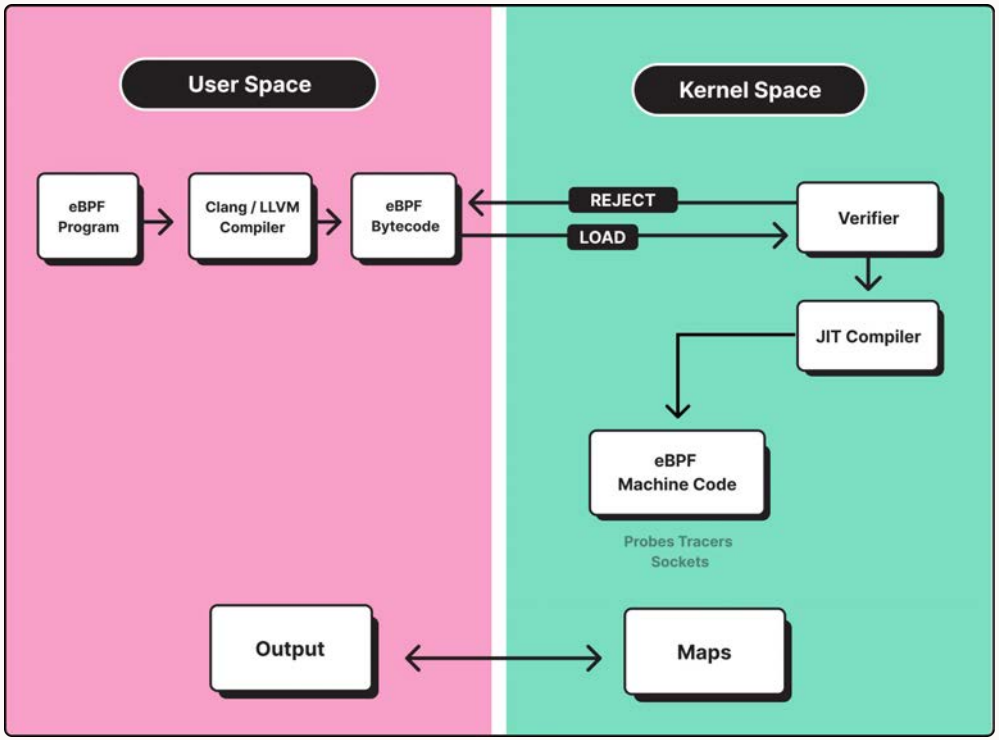
Brendan Gregg, production eng. @Netflix

eBPF, the next Linux superpower

eBPF in 60 secs

- BPF | 1993
- eBPF | 2014
- v4.14 | 2017

eBPF architecture



▲ Efficiency

Ever thought about your O11y SDK overhead?

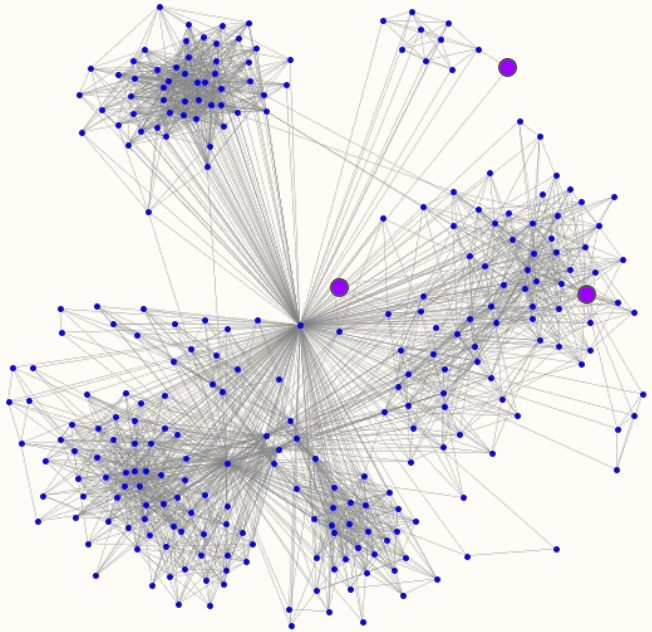
▲ Safety

You can’t crash the application.

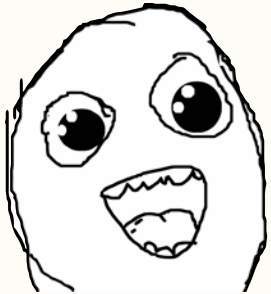
▲ 100% coverage

You can see all user-space apps, out-of-band

But why run things
in the kernel anyway?



Happy DevOps / SRE



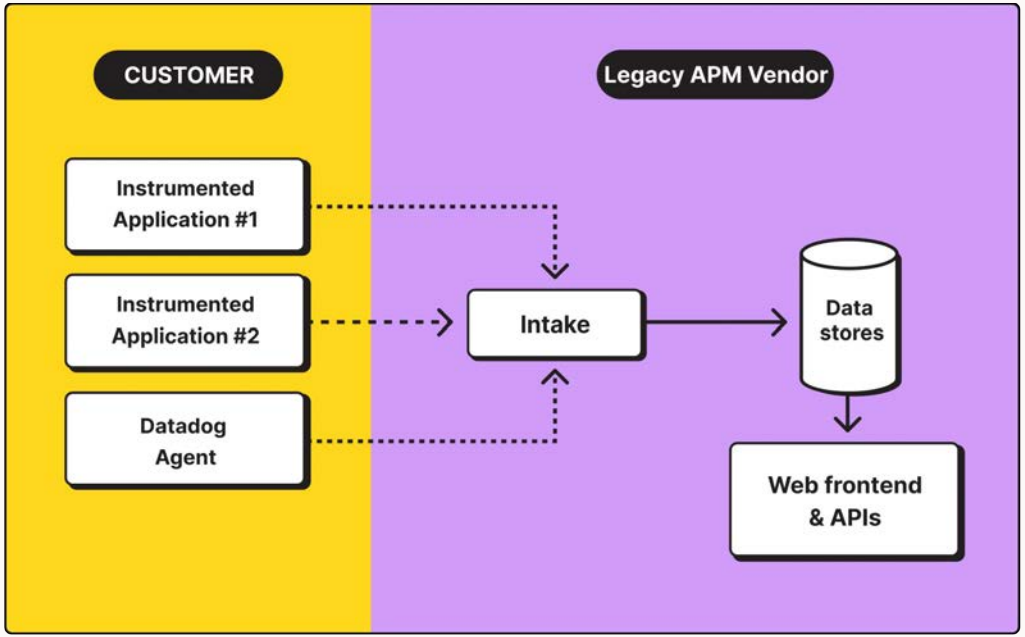
No more convincing **the R&D**

→ one uniform approach
→ super high professional standard

O11y doesn’t scale.

Storing huge amounts of irrelevant data for the little insight you need, making costs unbearable

Centralized architecture



▼ Random Sampling

”How can I control data volumes?
I have high-throughput APIs”

▼ Raw data is stored

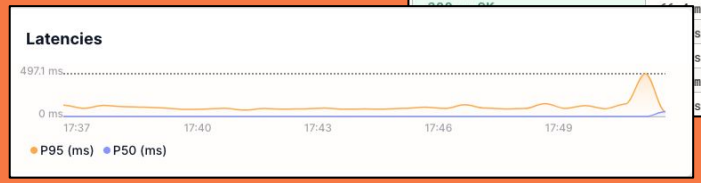
“How do I get a P50 latency metric over time?”
Span-based metric at the vendor’s backend

▼ Rigid data collection

“OK this is an interesting! give me everything!”
So you mean you want it all the time?

The debt of the
centralized architecture

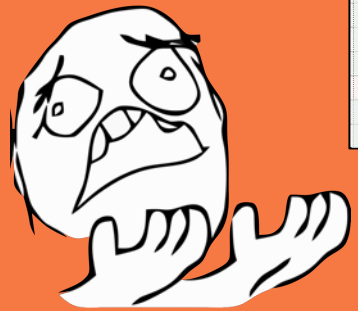
Status Code	Latency
200 - OK	29 ms
200 - OK	93.6 ms
200 - OK	7.8 ms
200 - OK	6.8 ms
200 - OK	228 µs
200 - OK	292 µs
200 - OK	103 µs
200 - OK	63.2 ms
200 - OK	189 µs
404 - Not Found	217 µs
404 - Not Found	217 µs
200 - OK	161 µs
200 - OK	98.2 µs
200 - OK	43.6 ms
200 - OK	669 µs
200 - OK	4.3 ms
200 - OK	98.5 ms
200 - OK	40.5 ms
200 - OK	75 µs



You store irrelevant data.

All the time.

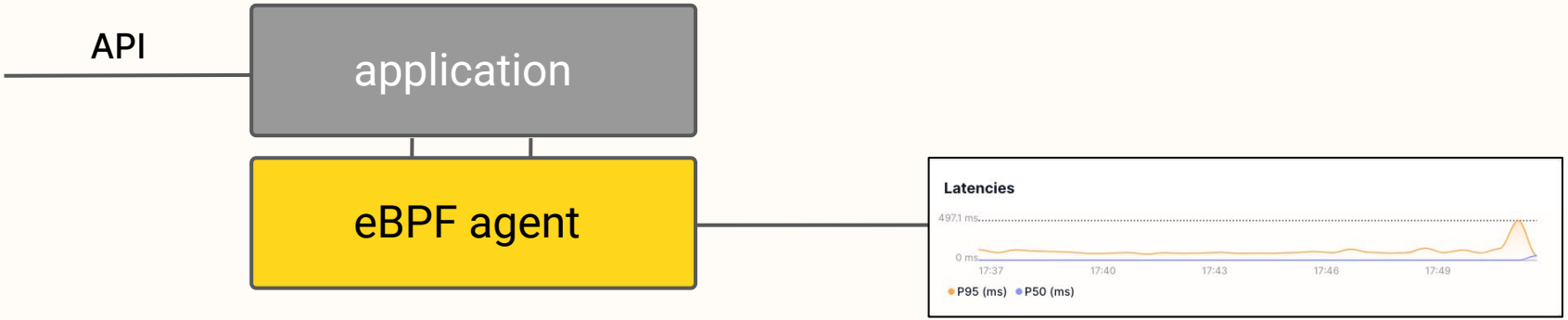
Status Code	Latency
200 - OK	29 ms
200 - OK	93.6 ms
200 - OK	7.8 ms
200 - OK	6.8 ms
200 - OK	228 µs
200 - OK	292 µs
200 - OK	103 µs
200 - OK	63.2 ms
200 - OK	189 µs
404 - Not Found	217 µs
404 - Not Found	217 µs
200 - OK	161 µs
200 - OK	98.2 µs
200 - OK	43.6 ms
200 - OK	669 µs
200 - OK	4.3 ms
200 - OK	98.5 ms
200 - OK	40.5 ms
200 - OK	75 µs
200 - OK	66.4 ms
200 - OK	9.6 ms
404 - Not Found	243 µs
200 - OK	25.9 ms
200 - OK	5.4 ms



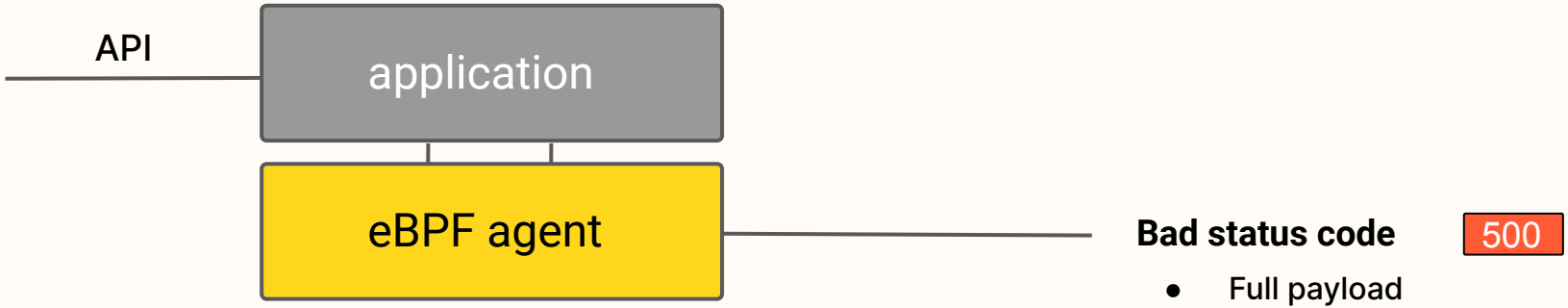
Limiting your cardinality

Where you do care.

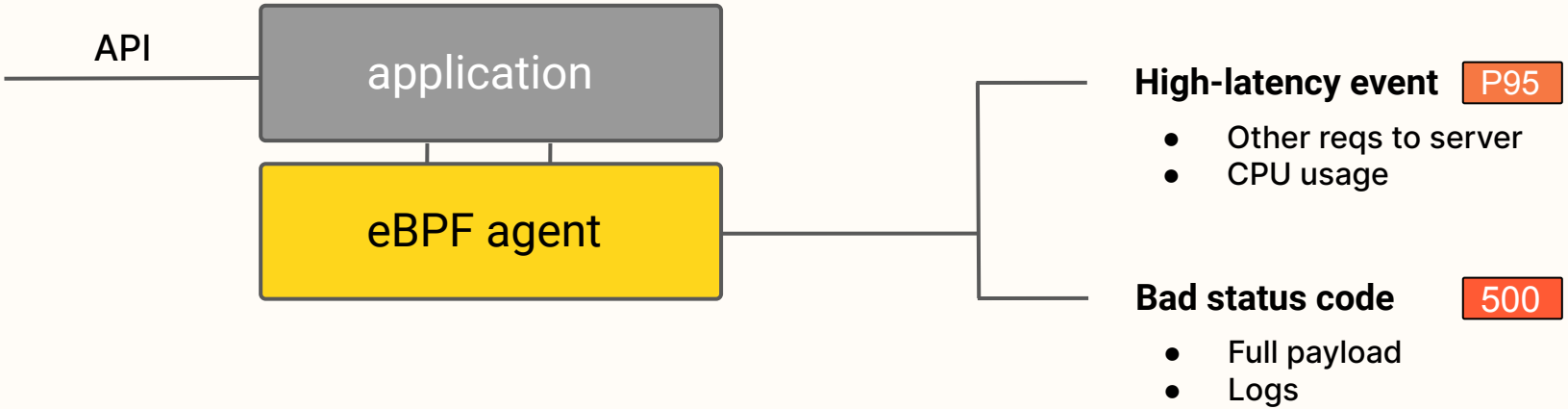
In-host span-based metrics



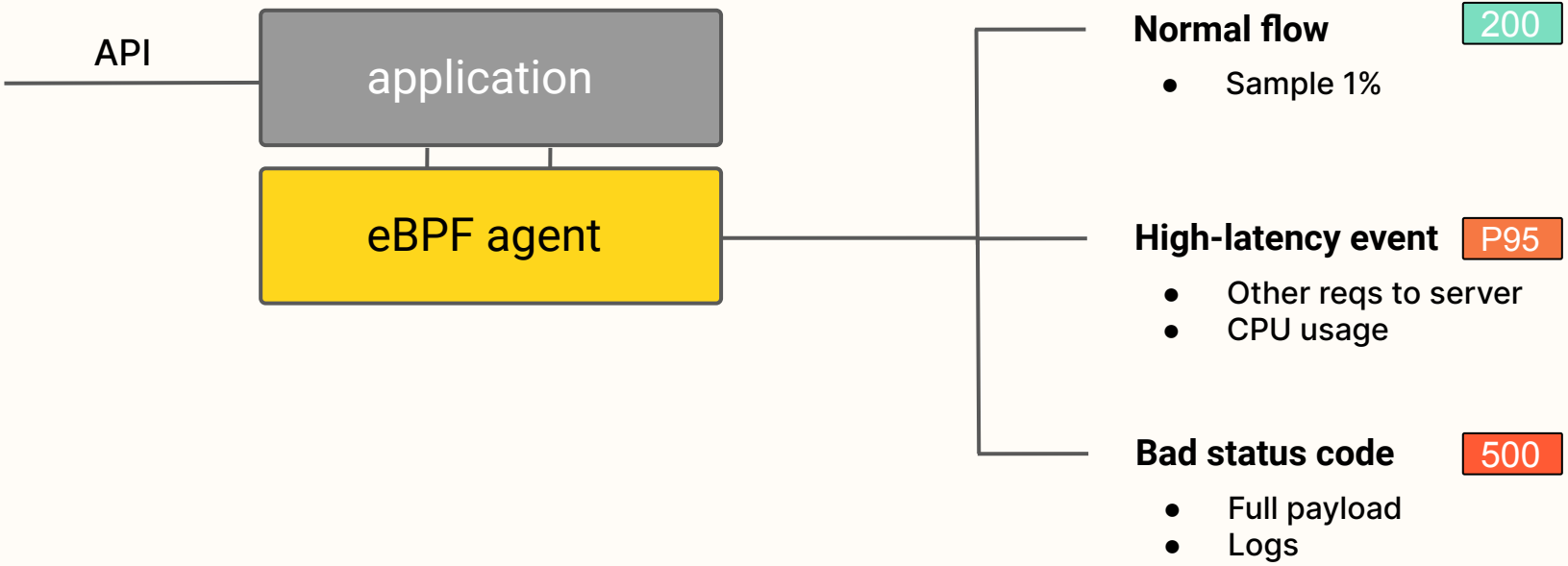
Variant-depth capturing



Variant-depth capturing

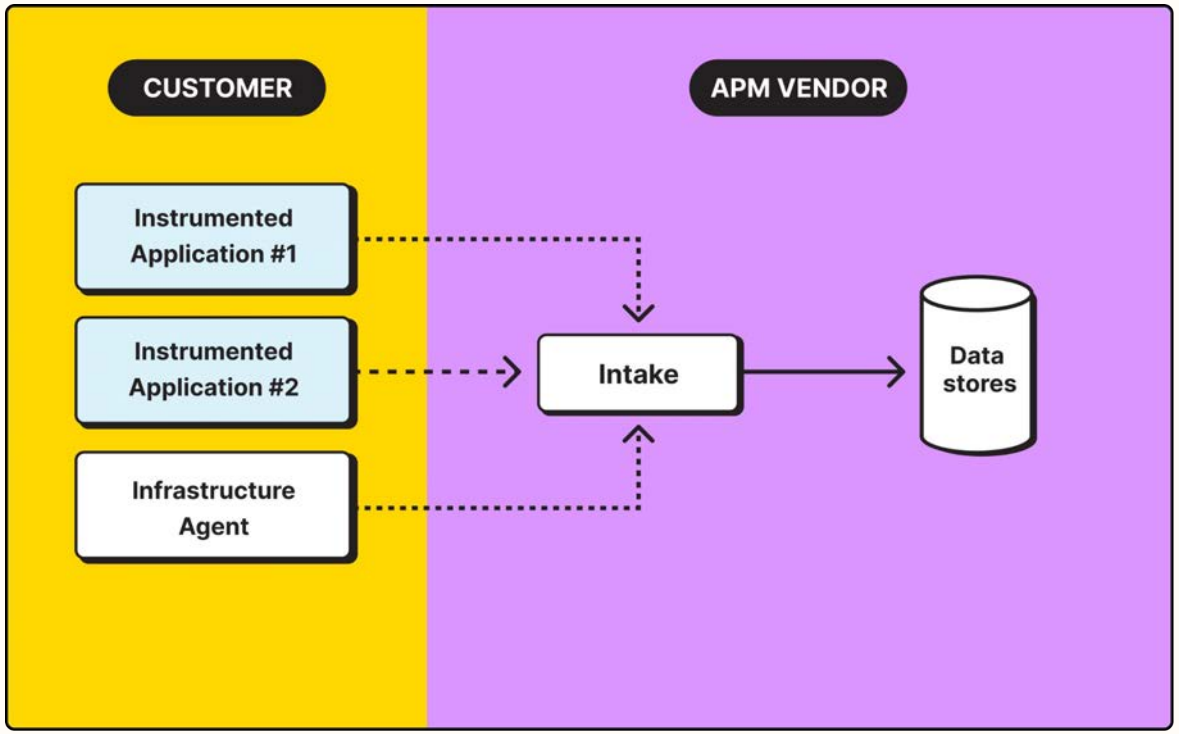


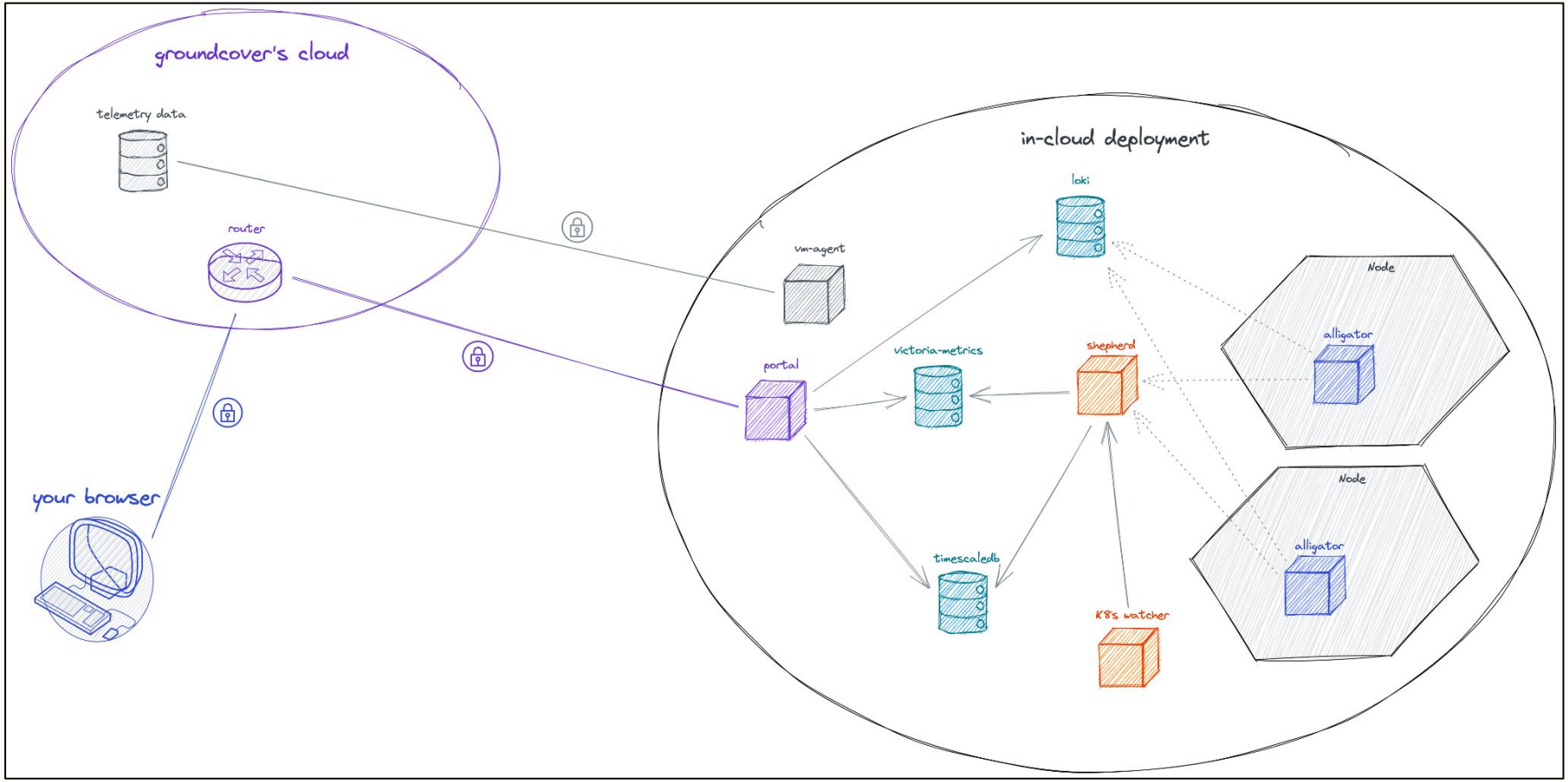
Variant-depth capturing



Your data is not kept private.

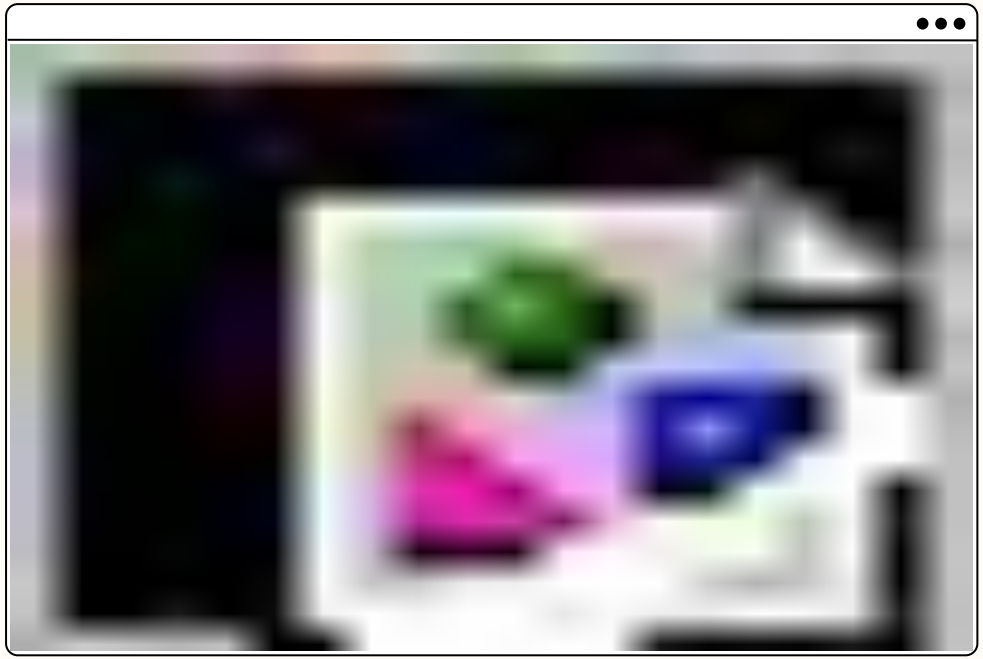
Storing sensitive information like logs and full traces at the vendor’s side


CUSTOMER
APM VENDOR
Instrumented Application #1
Instrumented Application #2
Infrastructure Agent
Intake
Data stores


groundcover's cloud
telemetry data
router
your browser
in-cloud deployment
loki
vm-agent
Node
alligator
portal
victoria-metrics
shepherd
Node
alligator
timescaledb
K8s watcher

This is our current reality.

- **eBPF instrumentation**
  - Immediate time-to-value
  - Out-of-band deployment
- **Edge-based observability**
  - Built for scale
  - Breaks all tradeoffs
- **In-cloud architecture**
  - Data is kept private
  - In your full control



groundcover

Start Free

www.groundcover.com