
**IN PARTNERSHIP WITH:**

ALEF LAB

Red Hat

FerretDB

# What Developers Should Know about Observability


Peter Zaitsev,
Co-Founder at Coroot
March 4, 2025

# Let’s Get to Know You

# What is Observability?

**You**

what is observability

**ChatGPT**

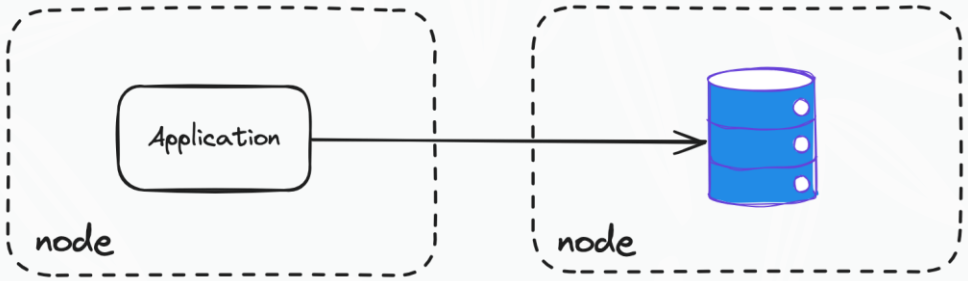
Observability is a concept in systems theory and engineering that refers to the ability to understand the internal state of a system based on its external outputs. In simpler terms, it's about how well you can observe and understand what's happening inside a system just by looking at its behavior from the outside.

In software engineering and computer science, observability is particularly important in the context of complex distributed systems, where traditional debugging techniques might not be sufficient. It involves collecting and analyzing data from various sources such as logs, metrics, traces, and events to gain insights into the performance, health, and behavior of the system.

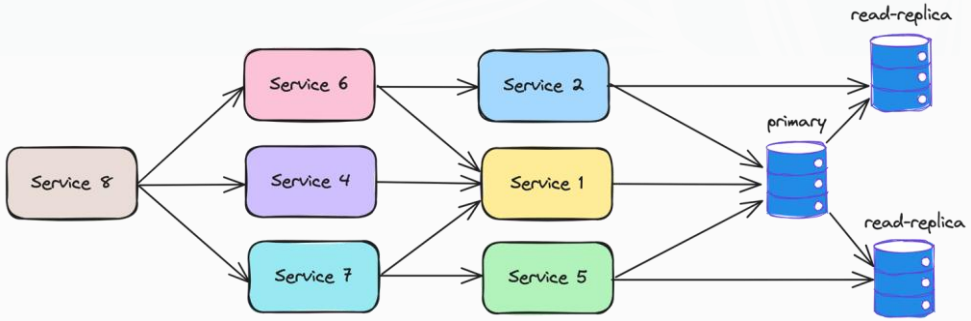

# What Is Observability

# Evolution of the Applications

2000s




2020s



# Why Observability?

- Availability
- Performance
- Cost Management
- Security

# Tasks Observability Helps with

- Reactive
  - Troubleshooting and Optimization
- Proactive
  - Finding Problems before they are Big Problems

# Alerting

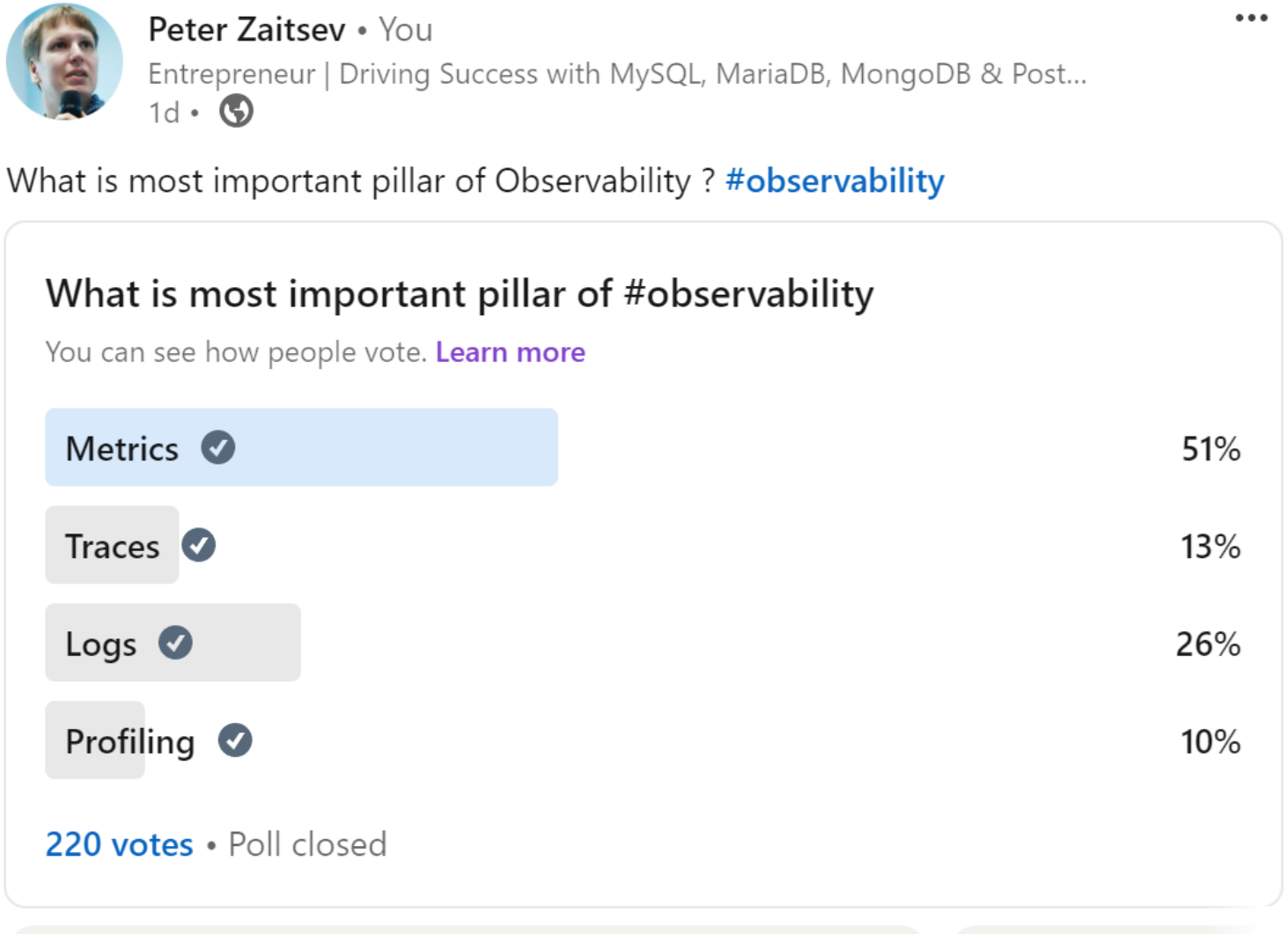
Notifies when Problem happens

Ensures the Right Escalation happens and the problem is resolved

# 4 Pillars of Observability

- Metrics
- Logs
- Tracing
- Profiling

# What is the Most Useful?

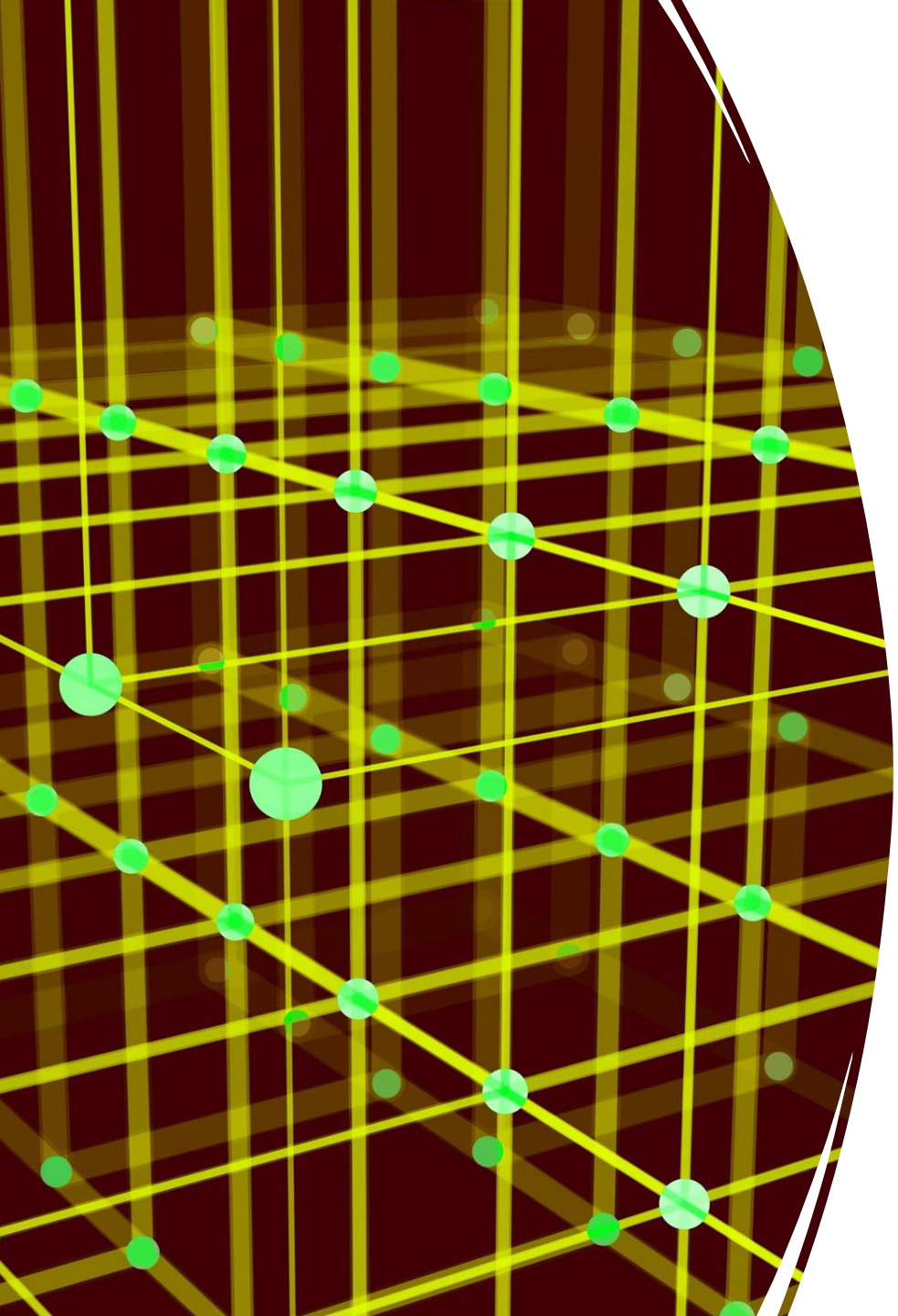


# Metrics

- High Level overview
- How many requests/sec there are happening?
- How many errors?
- Is the Host Down?
- 1000s of metrics may be collected every second
- Displayed on hundreds of graphs

# Logs

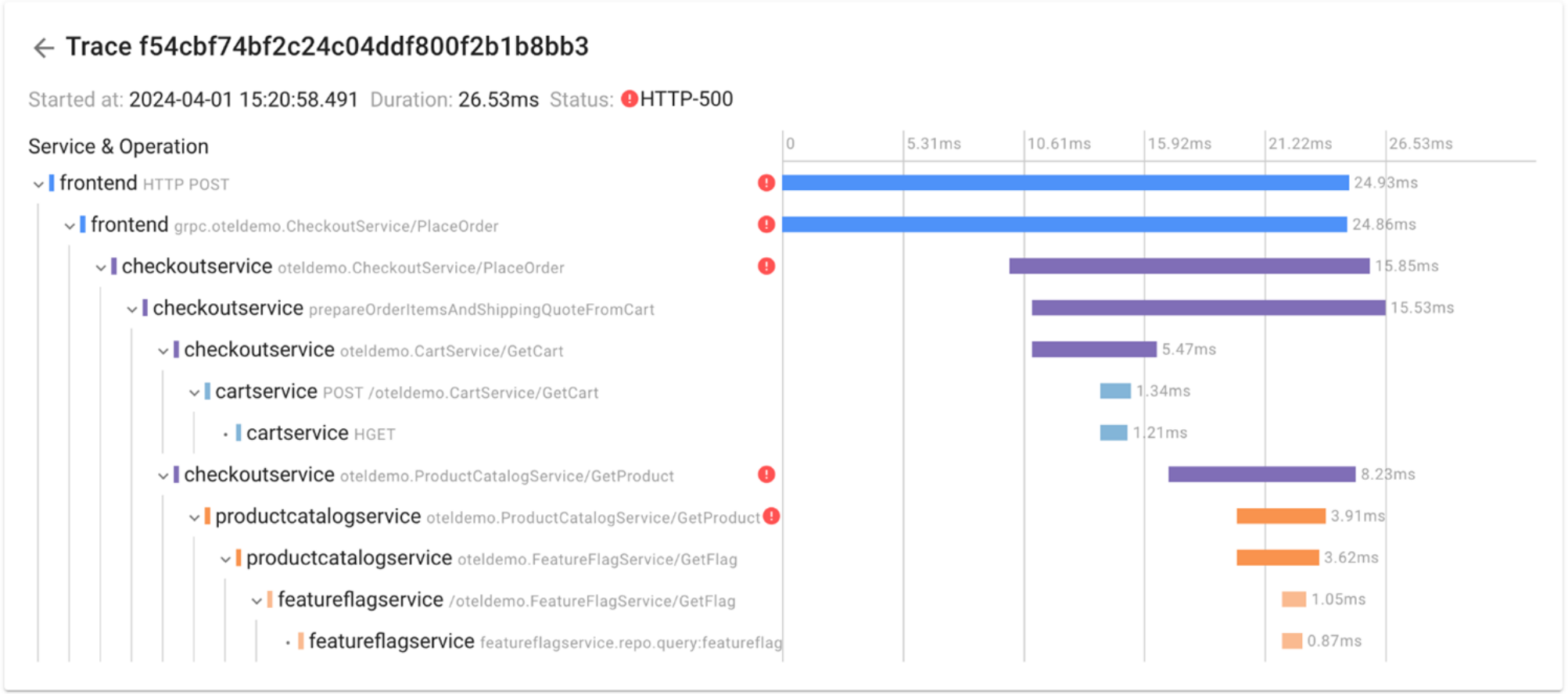
- Structured and Unstructured Format
- Have detailed information on what is happening
- Error logs – contain detailed information about cause of errors
- Expensive to Produce
- Expensive to Store and Analyze
- Sampling and Filtering is often used



# Distributed Tracing

- Tracks Application Requests as they Pass through the system
- Tricky as we need to pass some Trace_ID between different services
- Span – Named, Timed Operation which represents part of Workflow
- Great for Root Cause Analyses
- Often Sampled
- Expensive to produce and store

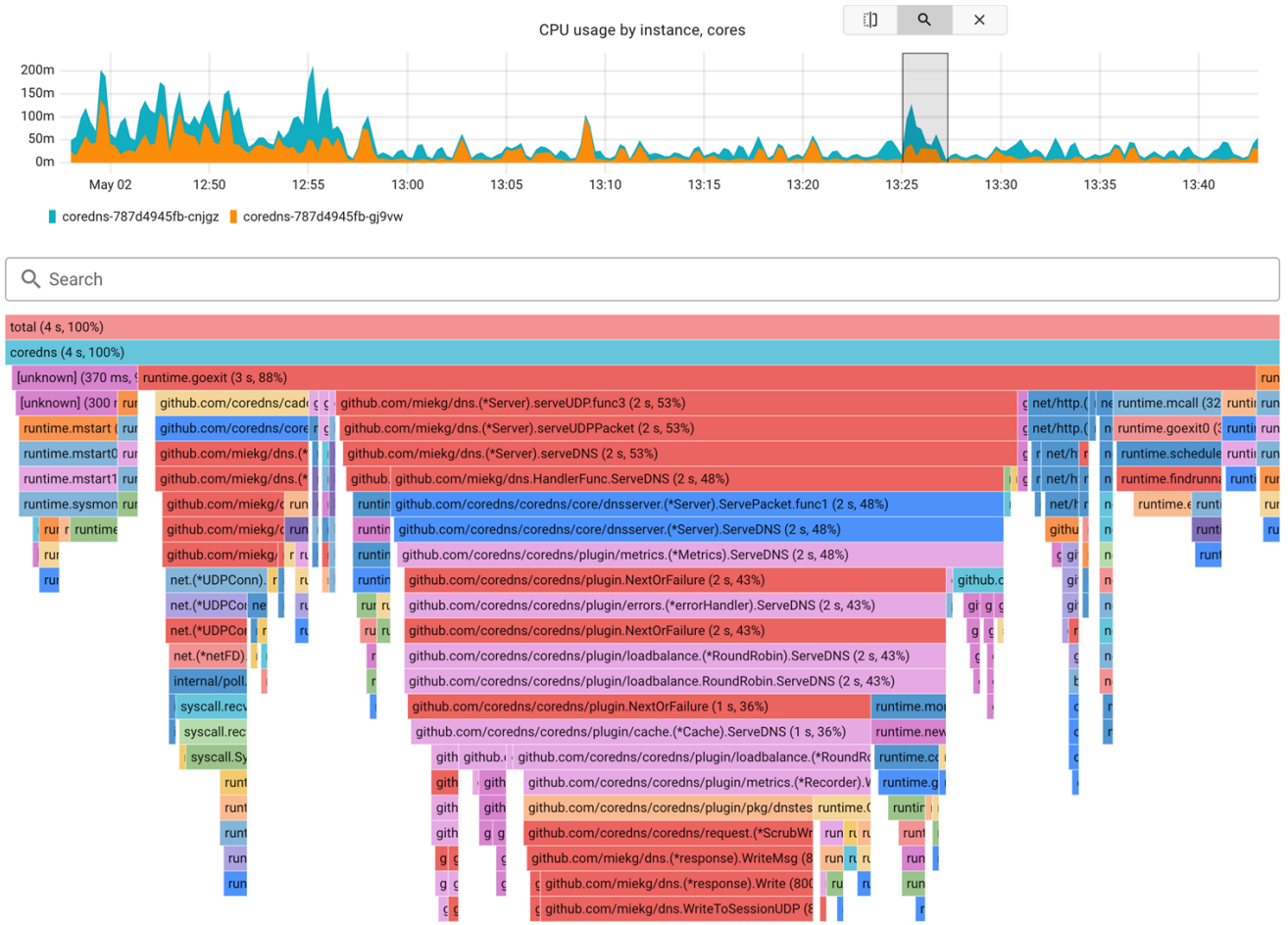
# Distributed Tracing Example



# Profiling

- Where CPU Time or Wall Clock time is Spent
- Single Service or Distributed
- Language Developer can Understand
- Comparisons are very helpful
- Programming language specific support needed

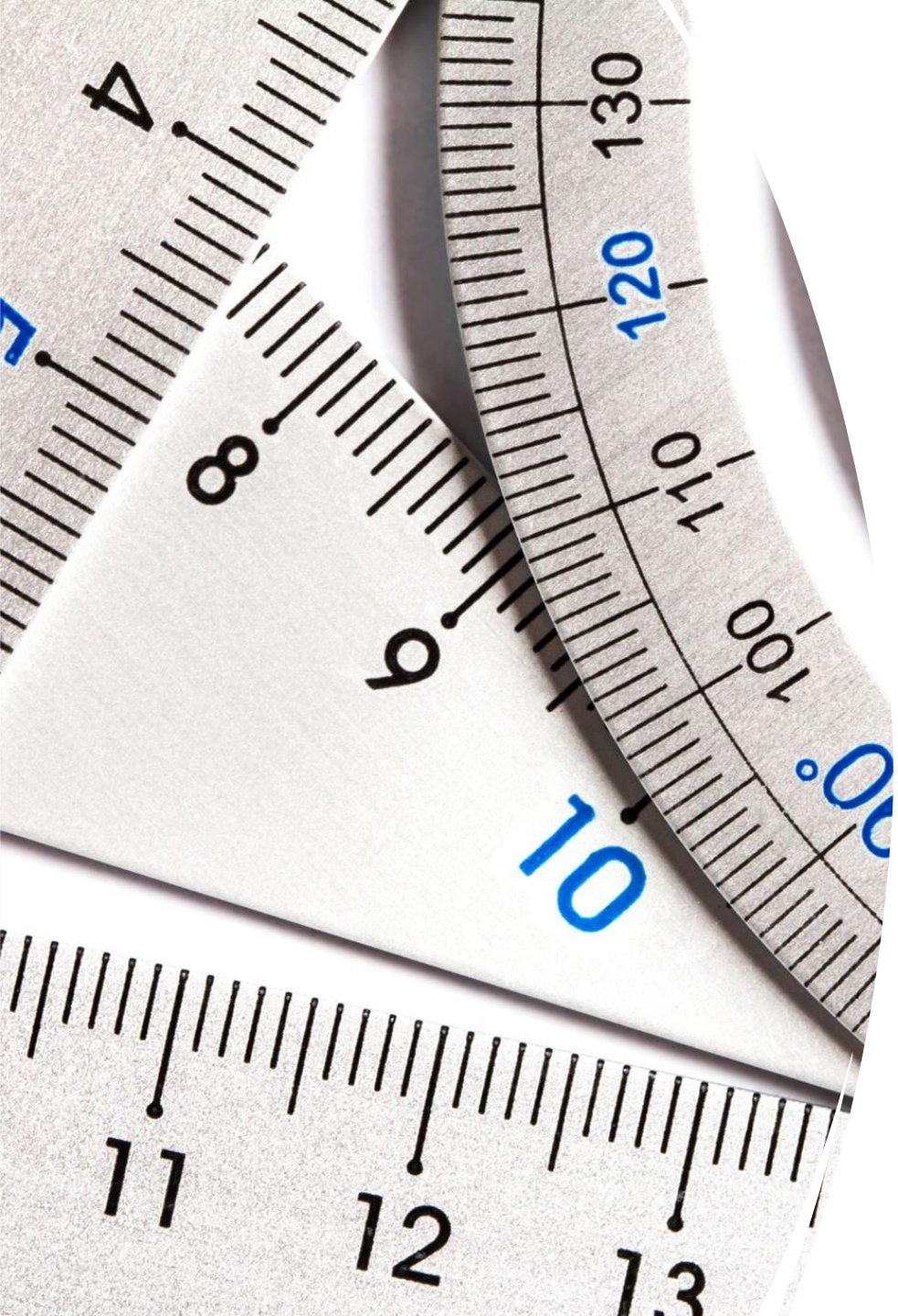
# eBPF Based CPU Profiling in Coroot



# Profiling: Comparison Mode

# Instrumentation

## How do we get all that Observability?



# Types of Instrumentation

- Static Instrumentation
  - Specific Places in the Code can Produce Metrics, Emit Logs, Traces
  - Linux ProcFS
- Dynamic Instrumentation
  - Allow Instrumentation “anything” dynamically
  - dTrace,  eBPF


Key Observability
Technologies

# Prometheus

- Metrics Capture and Processing
- OpenMetrics merged here
- Client Libraries for many programming languages
- Easy way to expose Metrics from your Application
- Coding Required

# Open Telemetry (OTEL)

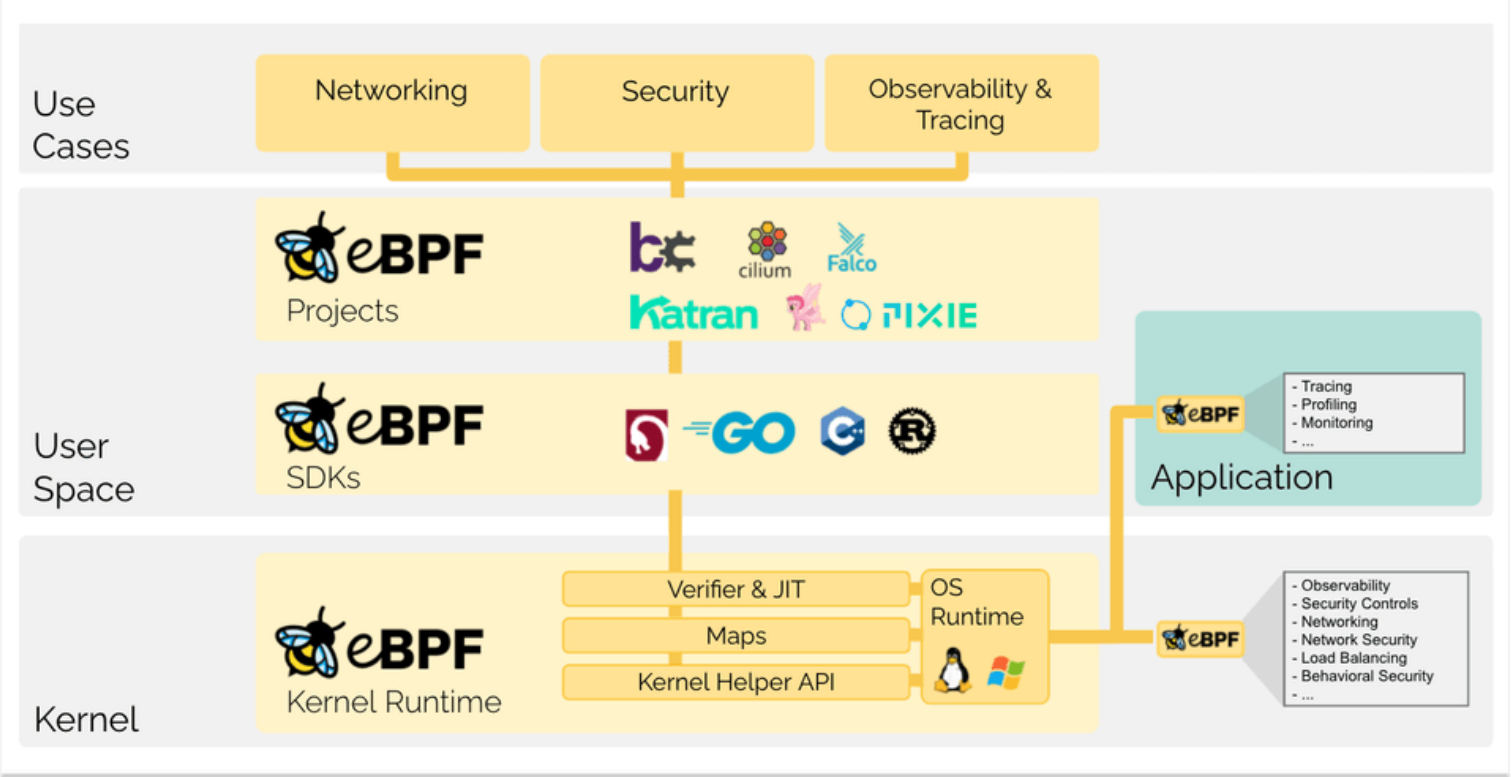
- Collection of APIs, SDKs and Tools
- To Instrument, Generate, Collect and Export Telemetry Data
- Covers Metrics, Logs, Traces
- Formed through merger of OpenTracing and OpenCensus projects
- Coding Required

# eBPF（Extended Berkley Packet Filter)

- Not Just Observability
- Dynamic Instrumentation
- User Space and Kernel Space
- Efficient
- Safe
- No Coding Required

# eBPF Illustrated

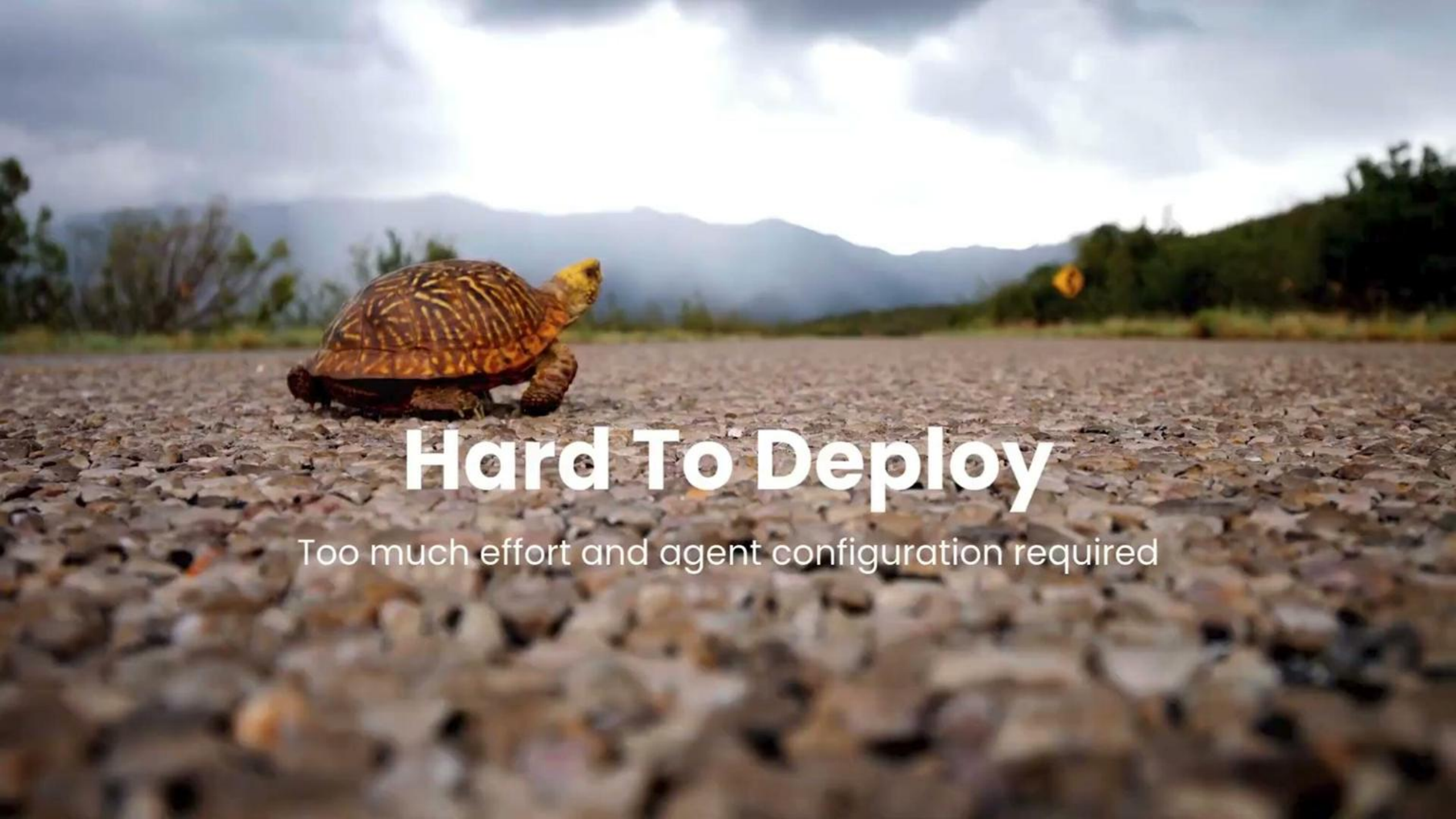
https://ebpf.io/what-is-ebpf/



# Typical SLIs (Service Level Indicators)

- Rate of Requests
  - Hard to define what is norm, through anomaly detection can help
- Availability
  - Error Rate;  High Latency Becomes Error at certain point
- Request Latency
  - Looking at 99% or more; Often with breakdown

# Observability Problems


Hard To Deploy
Too much effort and agent configuration required

# Hard to cover all of Infrastructure

Many solutions support only some kinds of infrastructure well

# Swiss Cheese Observability

Full of Holes!

# Hard to Use

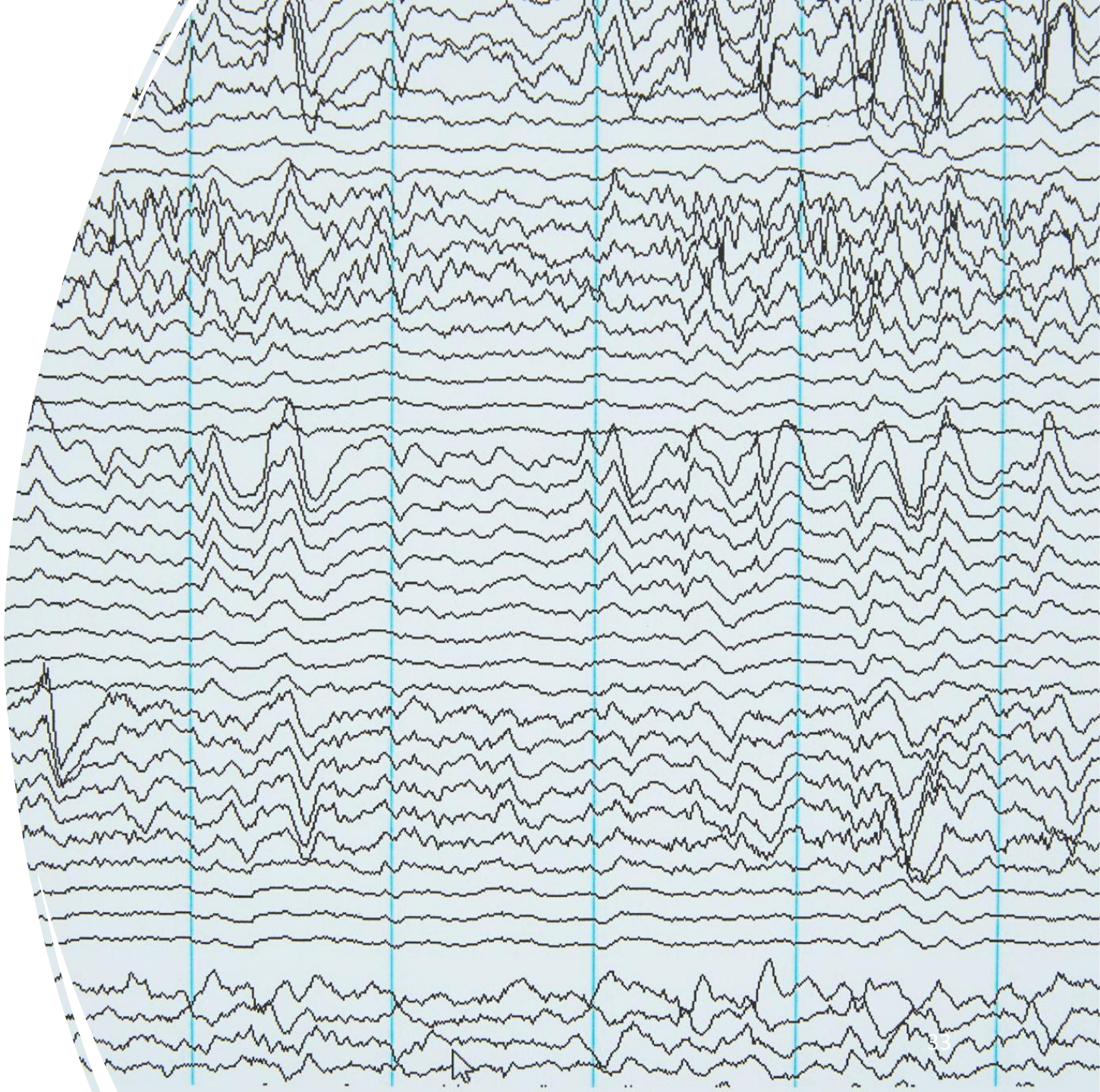
Overwhelming amount of hard to understand data

# Silos

- Large organizations have many groups with different responsibilities
- Multiple Vendors responsible for different parts of infrastructure
- Often using different tools
- Blame game and responsibility avoidance is common

# Noise

- Too much Noise in Alerting
- Team Fatigue and Burnout
- Alerts being Ignored or Mishandled
- Increased Downtime
- Poor User Experience



# Overhead


Instrumentation Slow things down

Observability data hard to store and process

Inadequate Level of Instrumentation

# Cost

- Proprietary and Cloud Observability solutions can get super expensive
- Why do you think Cisco bought Splunk ?
- High Costs drive Observability Choices
- Need Open Source, Efficient Solutions

# Coroot – Looking to Solve Some of Those

coroot :~#

Product Solutions Pricing Resources Live Demo

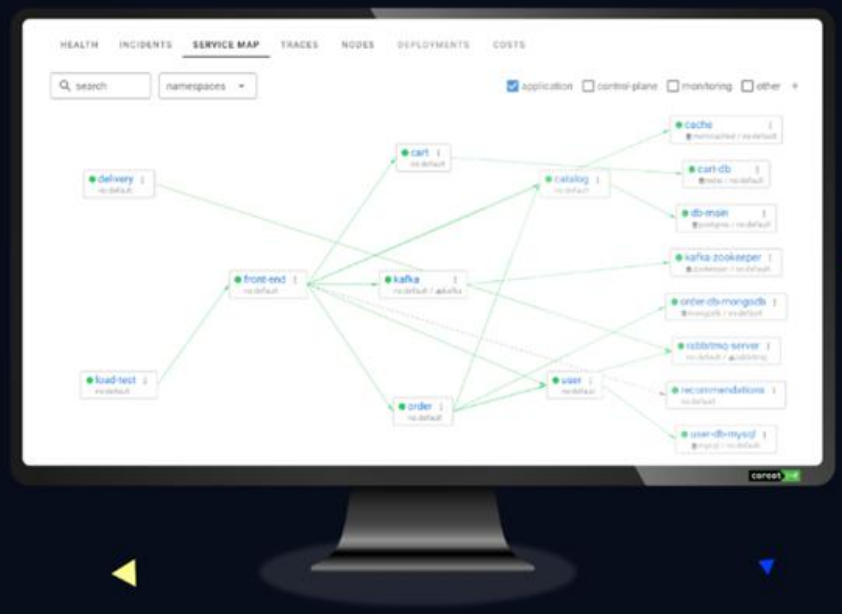
Star 5,285

## Enable system observability in minutes, no code changes required

Metrics, logs, traces, continuous profiling, and SLO-based alerting, supercharged with predefined dashboards and inspections.

Start free trial Live demo

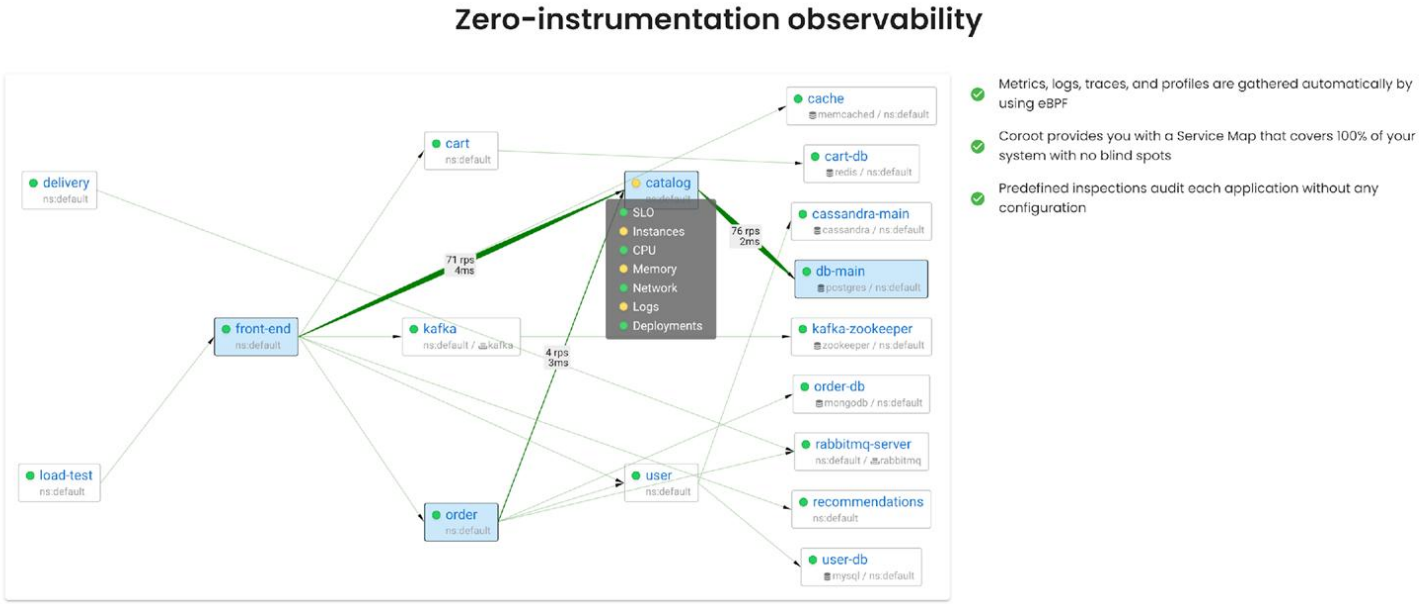
or install Coroot Community Edition (Open Source) →



# Coroot Editions

Open Source

Enterprise

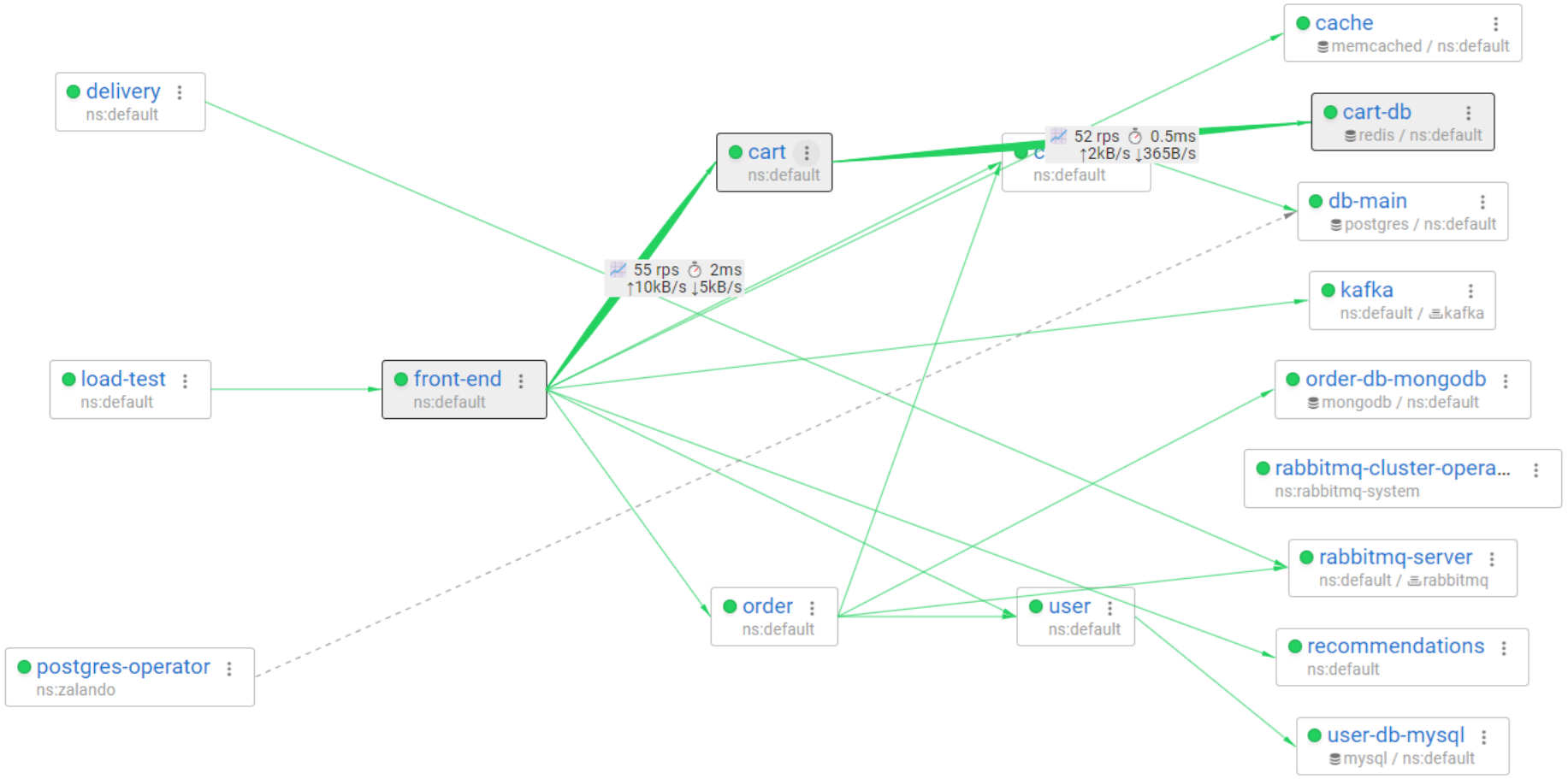



https://coroot.com/

# eBPF Magic

- Automatic Instrumentation with eBPF
- Use to together with conventionally exposed Linux Data
- Instrument SSL Calls
- L7 Protocol Decoding

# eBPF-based metrics



# Node Types & Costs for major clouds

HEALTH | INCIDENTS | SERVICE MAP | TRACES | NODES | DEPLOYMENTS | COSTS | ANOMALIES

## Nodes

| Node | CPU | Memory | Price | Idle cost | Cross-AZ traffic | Internet egress traffic |
|---|---|---|---|---|---|---|
| node5<br>c6a.xlarge / 4 vCPU / 8GB | usage:<br>request: | | $121.18/mo<br>on-demand | $39.09/mo | $0.00/mo | $0.00/mo |
| node1<br>c6a.xlarge / 4 vCPU / 8GB | usage:<br>request: | | $121.18/mo<br>on-demand | $34.62/mo | $0.00/mo | $0.00/mo |
| node3<br>c6a.xlarge / 4 vCPU / 8GB | usage:<br>request: | | $121.18/mo<br>on-demand | $32.68/mo | $0.00/mo | $0.00/mo |
| node2<br>c6a.xlarge / 4 vCPU / 8GB | usage:<br>request: | | $121.18/mo<br>on-demand | $27.39/mo | $0.00/mo | $0.00/mo |
| node4<br>c6a.xlarge / 4 vCPU / 8GB | usage:<br>request: | | $121.18/mo<br>on-demand | $23.90/mo | $0.00/mo | $0.00/mo |
| **TOTAL** | | | **$605.88**/mo | **$157.69**/mo | **$0.00**/mo | **$0.00**/mo |

Rows per page: 10 1-5 of 5

## Applications

| Category | Usage costs | Allocation costs | Overprovisioning costs | Cross-AZ traffic | Internet egress traffic |
|---|---|---|---|---|---|
| application | $108.47/mo | $174.39/mo | $92.78/mo | – | – |
| monitoring | $80.14/mo | $1.70/mo | $1.29/mo | – | $0.00/mo |
| control-plane | $51.47/mo | $17.20/mo | $11.98/mo | – | – |
| **TOTAL** | **$240.07**/mo | **$193.29**/mo | **$106.05**/mo | – | **$0.00**/mo |

# Handling Observability Problems

# Hard to Deploy

No Code, No Configuration to get 90% of the Value


Hard to cover all the Infrastructure
Cover Kubernetes, Containers, Cloud, VM, Bare Metal

# Hard to Use

Less Raw Data, More Actionable Insights

# Silos

- Complete Infrastructure and App Coverage
- Evidence based Root Cause Analyses

# Noise

Focus on Objective Measures rather than Vanity Ratios

# Overhead

- eBPF – Modern Technology designed for speed
- Heavily Optimized Open Source Agents

# Cost

- Roll your own
- Flexible data retention options
- State of Art Data Storage Technologies

# Words of Advice

- Ensure your observability is complete, with no blind spots
- Ensure you have tools to reliably identify the components experiencing issues
- Use skill-appropriate tools, less can be more in the time crunch
- Have evidence for escalation to another team or vendor

# Thank you!
# Let's Connect!

https://www.linkedin.com/in/peterzaitsev/

https://twitter.com/PeterZaitsev

http://www.peterzaitsev.com

**IN PARTNERSHIP WITH:**

ALEF LAB

Red Hat

FerretDB