Hanoi, Vietnam 2024

PERCONA UNIVERSITY

Open Source Databases Meetup

In partnership with BNH

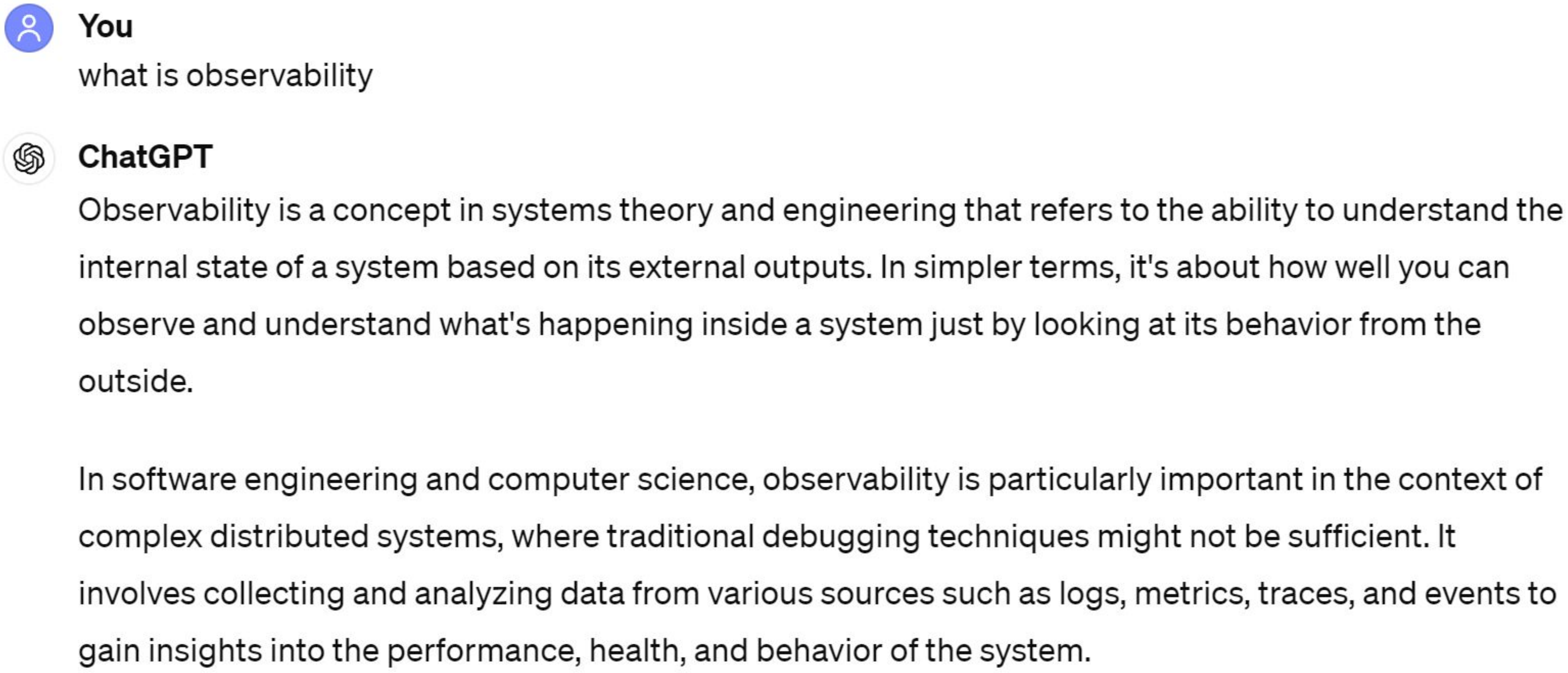
Better and Higher

What Developers Should Know about Observability

Peter Zaitsev,
Co-Founder at Coroot
November 2, 2024

Let’s Get to Know You

What is Observability?



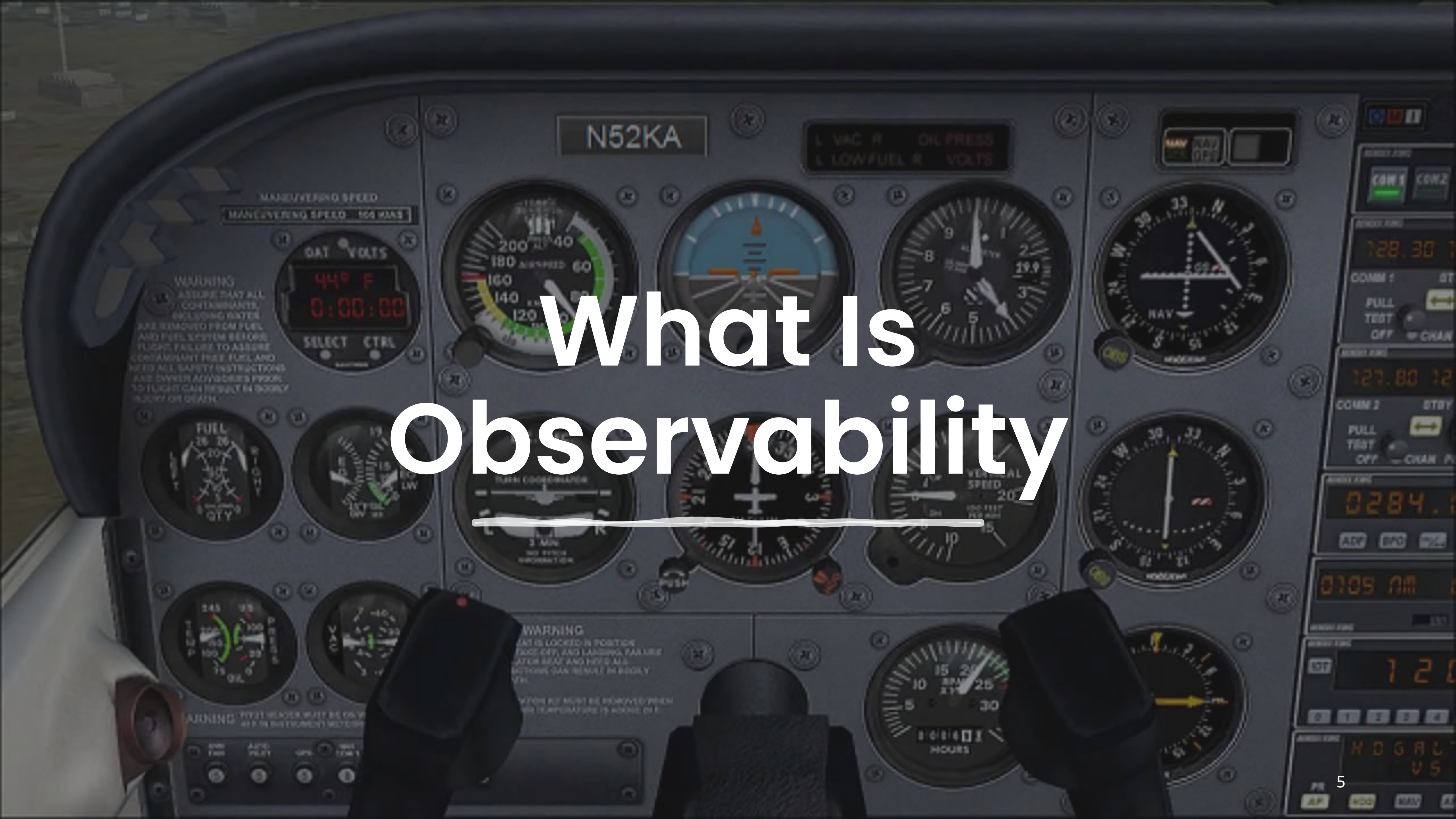
You

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.

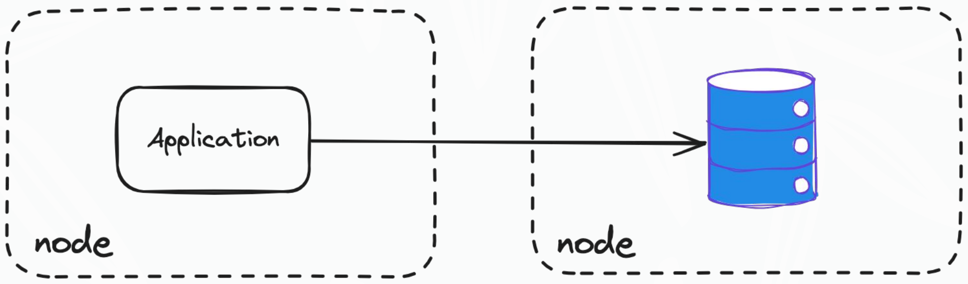
In software engineering and computer science, observability is particularly important in the context of complex distributed systems, where traditional debugging techniques might not be sufficient. It involves collecting and analyzing data from various sources such as logs, metrics, traces, and events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



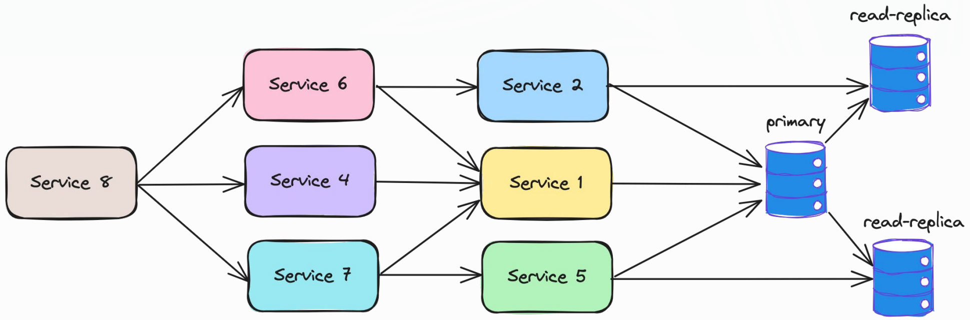
What Is Observability

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

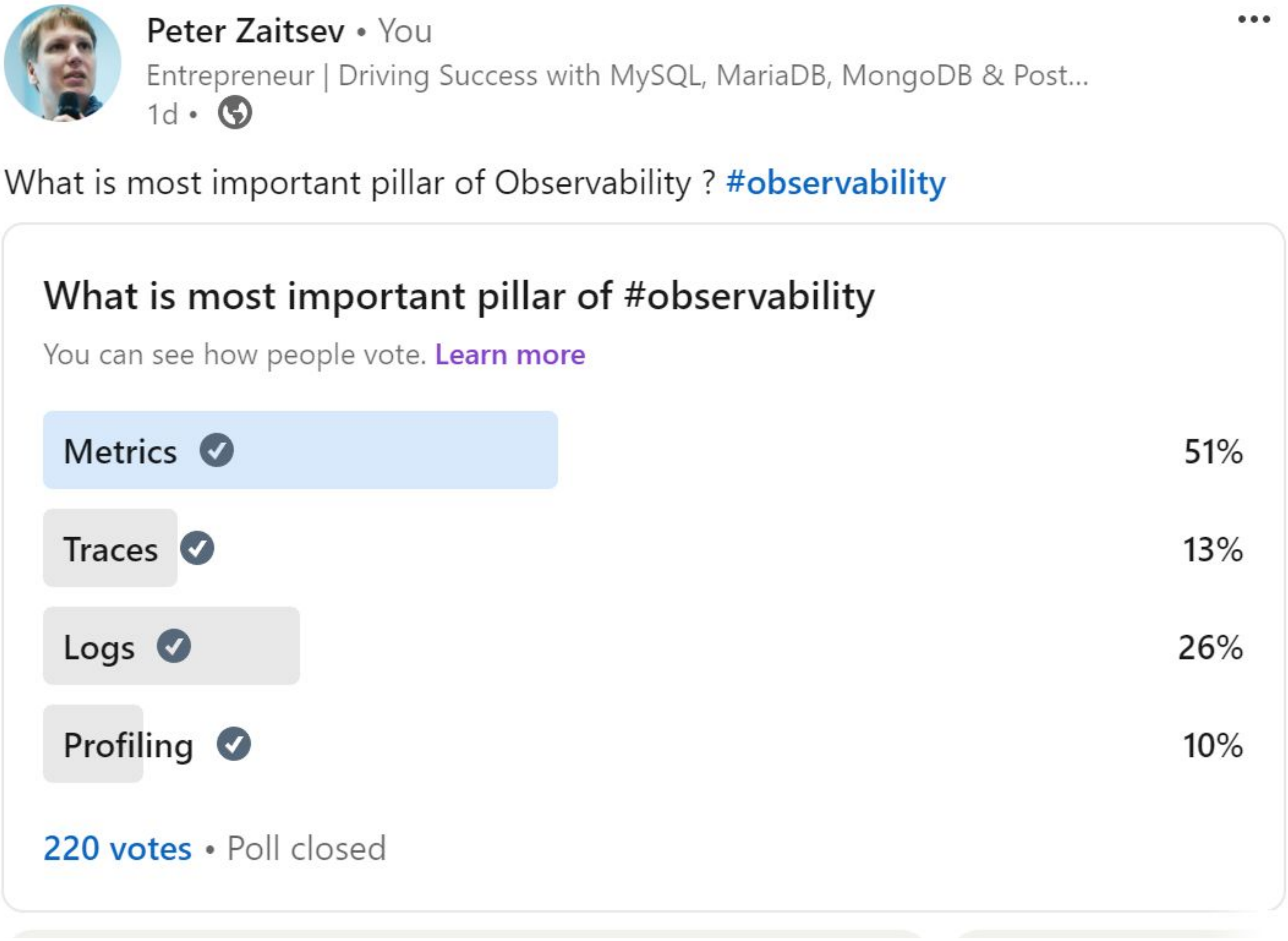
Notifies when Problem happens

Ensures the Right Escalation happens and the problem is resolved

4 Pillars of Observability

- Metrics
- Logs
- Tracing
- Profiling

What is the Most Useful?



Peter Zaitsev • You
Entrepreneur | Driving Success with MySQL, MariaDB, MongoDB & Post...
1d •

What is most important pillar of Observability ? #observability

What is most important pillar of #observability

You can see how people vote. Learn more

Metrics ✓	51%
Traces ✓	13%
Logs ✓	26%
Profiling ✓	10%

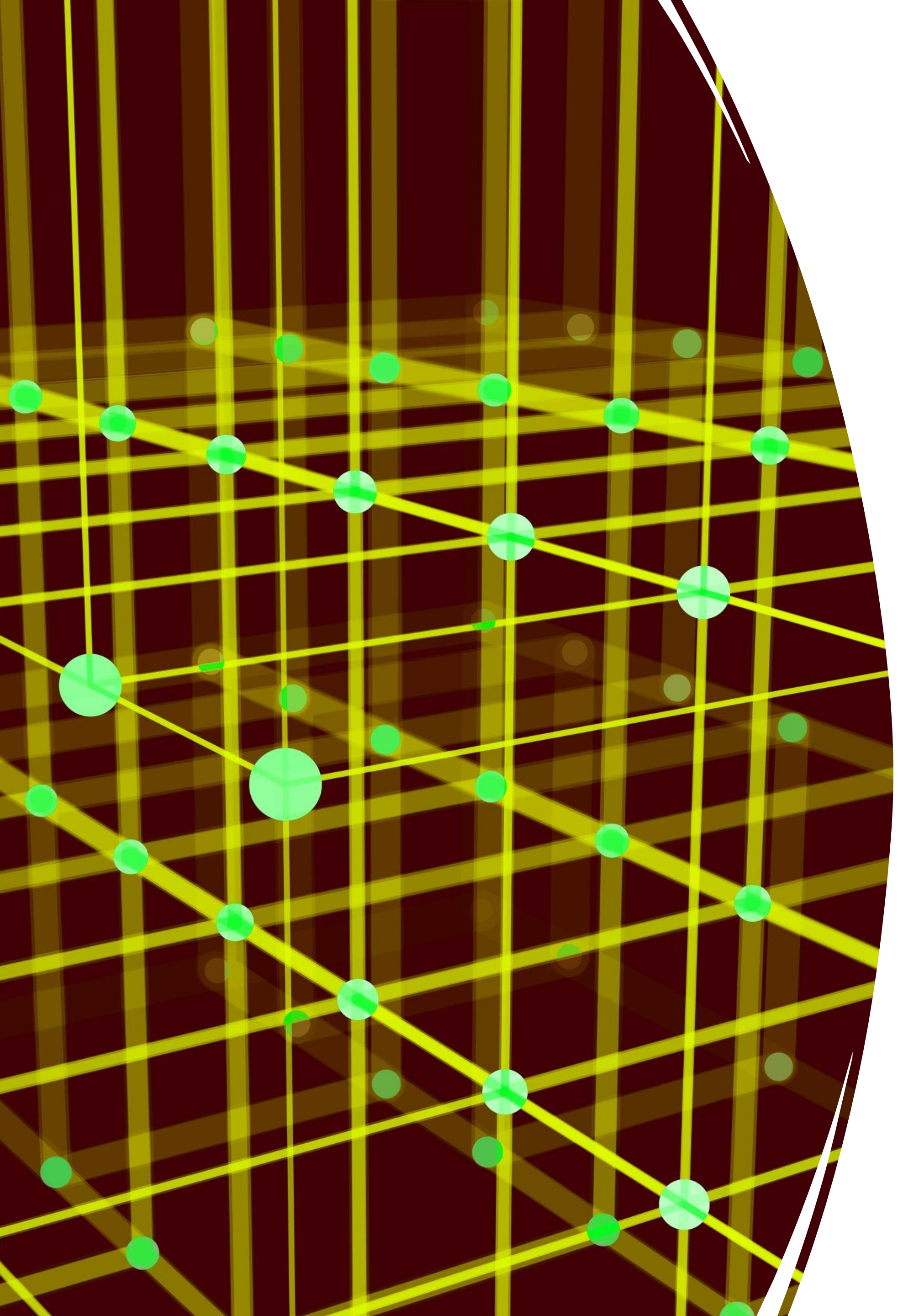
220 votes • Poll closed

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

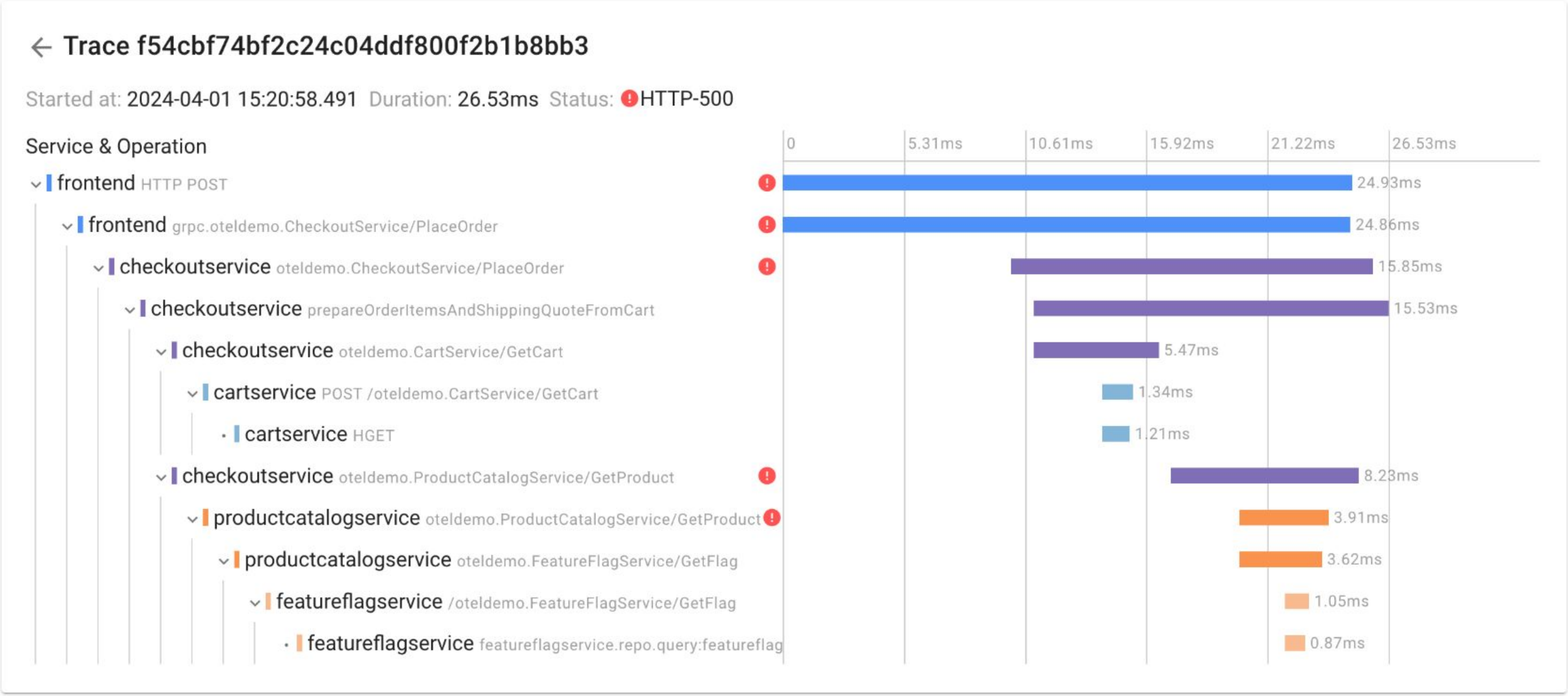
- Structured and Unstructured Format
- Have detailed information on what is happening
- Error logs – contain detailed information about cause of errors
- Expensive to Produce
- Expensive to Store and Analyze
- Sampling and Filtering is often used



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



← Trace f54cbf74bf2c24c04ddf800f2b1b8bb3

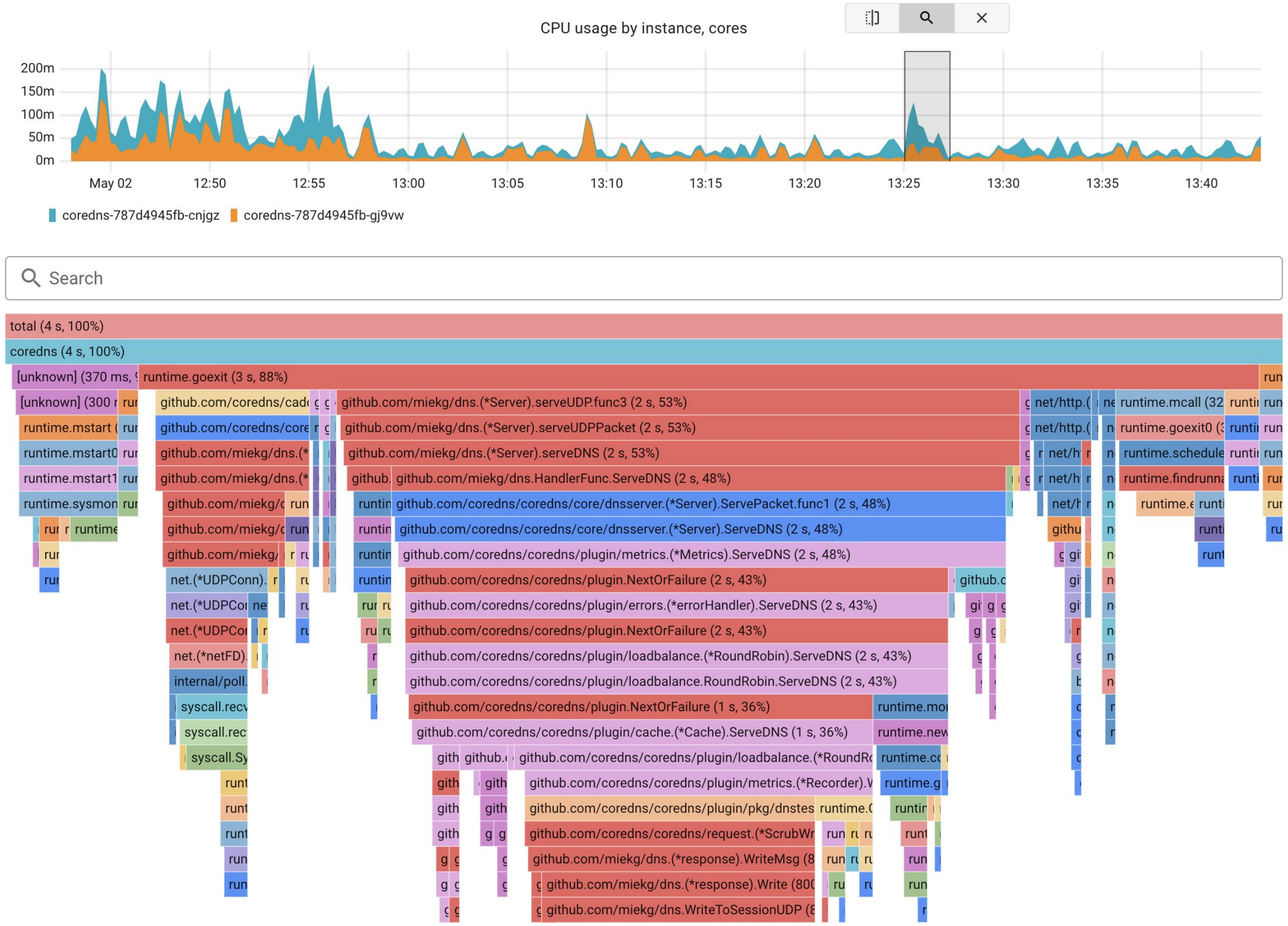
Started at: 2024-04-01 15:20:58.491 Duration: 26.53ms Status: ❗HTTP-500

Service & Operation		
frontend HTTP POST	❗	24.93ms
frontend grpc.oteldemo.CheckoutService/PlaceOrder	❗	24.86ms
checkoutservice oteldemo.CheckoutService/PlaceOrder	❗	15.85ms
checkoutservice prepareOrderItemsAndShippingQuoteFromCart		15.53ms
checkoutservice oteldemo.CartService/GetCart		5.47ms
cartservice POST /oteldemo.CartService/GetCart		1.34ms
cartservice HGET		1.21ms
checkoutservice oteldemo.ProductCatalogService/GetProduct	❗	8.23ms
productcatalogservice oteldemo.ProductCatalogService/GetProduct	❗	3.91ms
productcatalogservice oteldemo.FeatureFlagService/GetFlag		3.62ms
featureflagservice /oteldemo.FeatureFlagService/GetFlag		1.05ms
featureflagservice featureflagservice.repo.query:featureflag		0.87ms

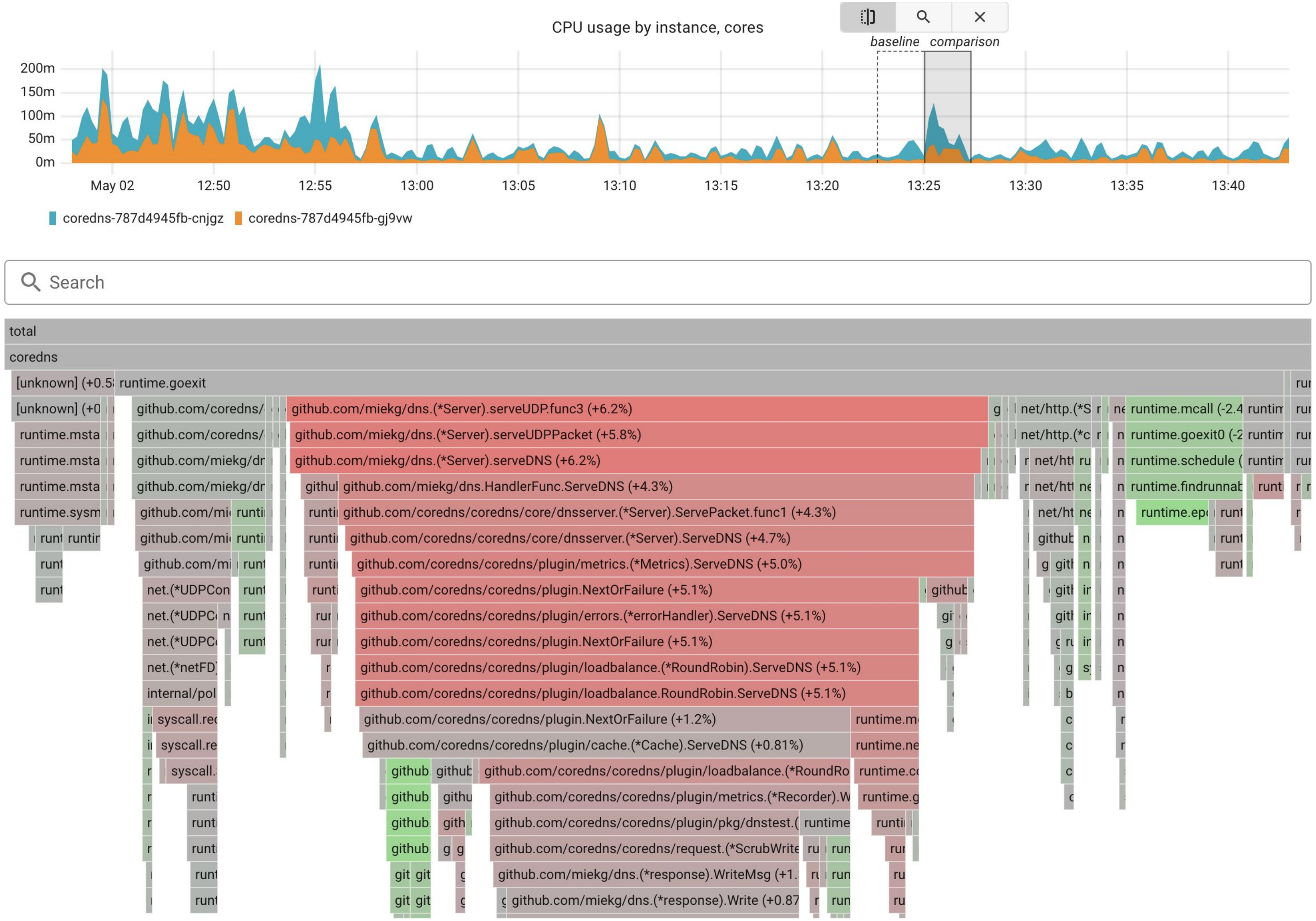
Profiling

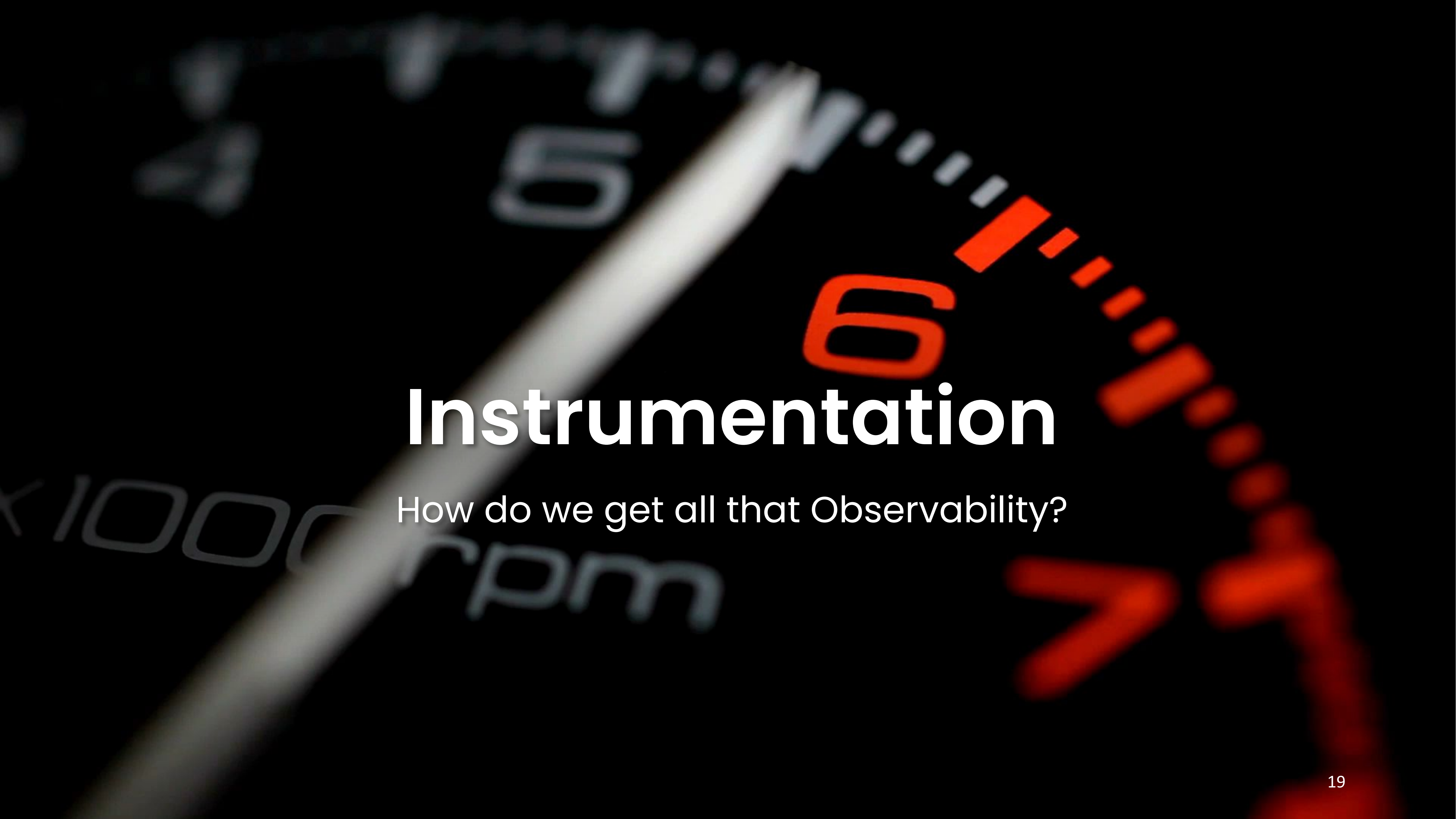
- Where CPU Time or Wall Clock time is Spent
- Single Service or Distributed
- Language Developer can Understand
- Comparisons are very helpful
- Programming language specific support needed

eBPF Based CPU Profiling in Coroot


CPU usage by instance, cores
200m
150m
100m
50m
0m
May 02
12:50
12:55
13:00
13:05
13:10
13:15
13:20
13:25
13:30
13:35
13:40
coredns-787d4945fb-cnjgz
coredns-787d4945fb-gj9vw
Search
total (4 s, 100%)
coredns (4 s, 100%)
runtime.goexit (3 s, 88%)
github.com/miekg/dns.(*Server).serveUDP.func3 (2 s, 53%)
github.com/miekg/dns.(*Server).serveUDPPacket (2 s, 53%)
github.com/miekg/dns.(*Server).serveDNS (2 s, 53%)
github.com/miekg/dns.HandlerFunc.ServeDNS (2 s, 48%)
github.com/coredns/coredns/core/dnsserver.(*Server).ServePacket.func1 (2 s, 48%)
github.com/coredns/coredns/core/dnsserver.(*Server).ServeDNS (2 s, 48%)
github.com/coredns/coredns/plugin/metrics.(*Metrics).ServeDNS (2 s, 48%)
github.com/coredns/coredns/plugin.NextOrFailure (2 s, 43%)
github.com/coredns/coredns/plugin/errors.(*errorHandler).ServeDNS (2 s, 43%)
github.com/coredns/coredns/plugin.NextOrFailure (2 s, 43%)
github.com/coredns/coredns/plugin/loadbalance.(*RoundRobin).ServeDNS (2 s, 43%)
github.com/coredns/coredns/plugin/loadbalance.RoundRobin.ServeDNS (2 s, 43%)
github.com/coredns/coredns/plugin.NextOrFailure (1 s, 36%)
github.com/coredns/coredns/plugin/cache.(*Cache).ServeDNS (1 s, 36%)
runtime.mstart
runtime.mstart0
runtime.mstart1
runtime.sysmon
runtime.mcall
runtime.schedule
net.(*UDPConn).
net.(*netFD).
syscall.recv

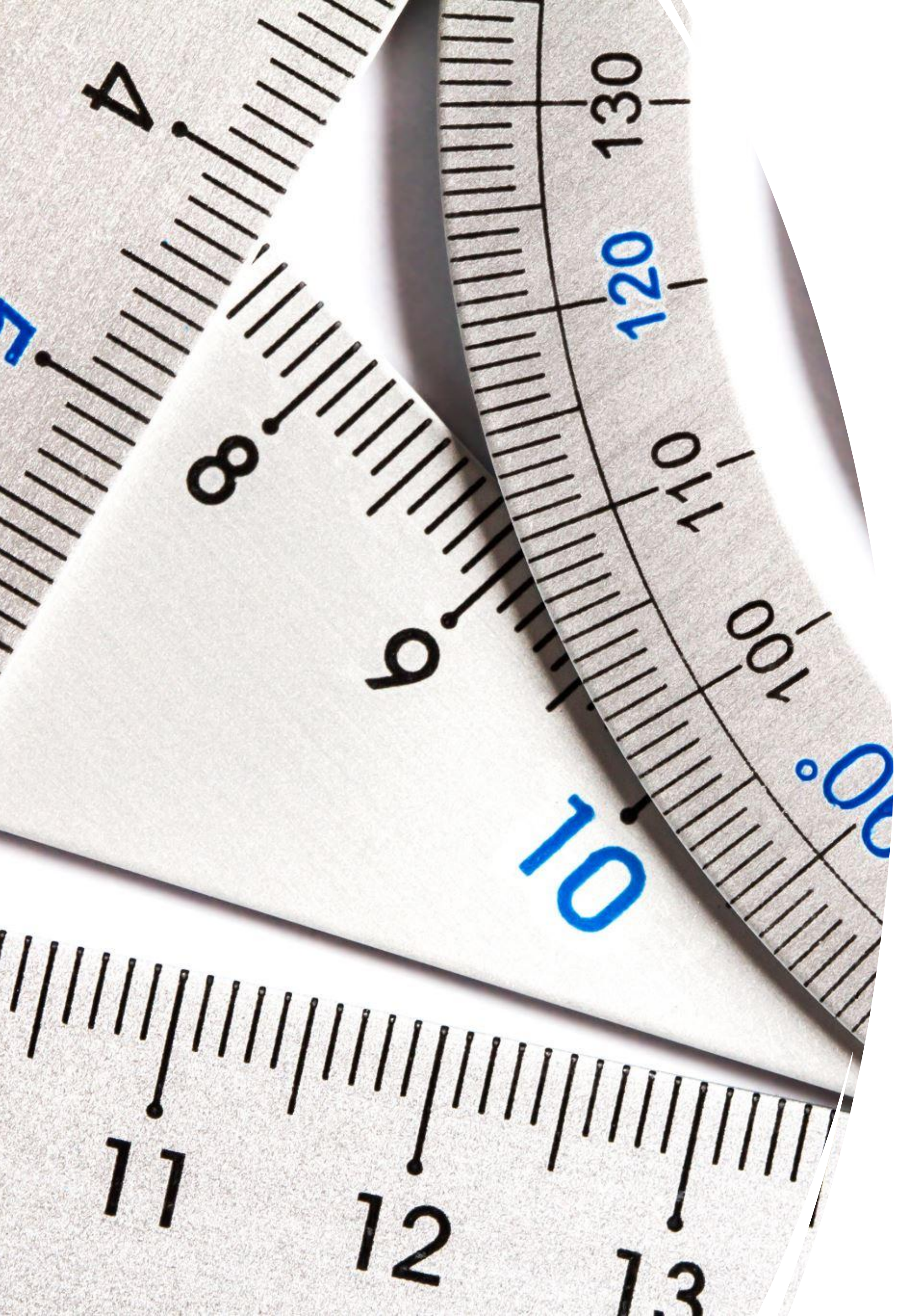
Profiling: Comparison Mode


CPU usage by instance, cores
baseline
comparison
200m
150m
100m
50m
0m
May 02
12:50
12:55
13:00
13:05
13:10
13:15
13:20
13:25
13:30
13:35
13:40
coredns-787d4945fb-cnjgz
coredns-787d4945fb-gj9vw
Search
total
coredns
runtime.goexit
github.com/miekg/dns.(*Server).serveUDP.func3 (+6.2%)
github.com/miekg/dns.(*Server).serveUDPPacket (+5.8%)
github.com/miekg/dns.(*Server).serveDNS (+6.2%)
github.com/miekg/dns.HandlerFunc.ServeDNS (+4.3%)
github.com/coredns/coredns/core/dnsserver.(*Server).ServePacket.func1 (+4.3%)
github.com/coredns/coredns/core/dnsserver.(*Server).ServeDNS (+4.7%)
github.com/coredns/coredns/plugin/metrics.(*Metrics).ServeDNS (+5.0%)
github.com/coredns/coredns/plugin.NextOrFailure (+5.1%)
github.com/coredns/coredns/plugin/errors.(*errorHandler).ServeDNS (+5.1%)
github.com/coredns/coredns/plugin.NextOrFailure (+5.1%)
github.com/coredns/coredns/plugin/loadbalance.(*RoundRobin).ServeDNS (+5.1%)
github.com/coredns/coredns/plugin/loadbalance.RoundRobin.ServeDNS (+5.1%)
github.com/coredns/coredns/plugin.NextOrFailure (+1.2%)
github.com/coredns/coredns/plugin/cache.(*Cache).ServeDNS (+0.81%)
github.com/coredns/coredns/plugin/loadbalance.(*RoundRo
github.com/coredns/coredns/plugin/metrics.(*Recorder).W
github.com/coredns/coredns/plugin/pkg/dnstest.(
github.com/coredns/coredns/request.(*ScrubWrite
github.com/miekg/dns.(*response).WriteMsg (+1.
github.com/miekg/dns.(*response).Write (+0.87
runtime.mcall (-2.4
runtime.goexit0 (-2
runtime.schedule (
runtime.findrunnab
runtime.ep
net.(*UDPCon
syscall.rec
runtime.msta
runtime.sysm
internal/pol



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)

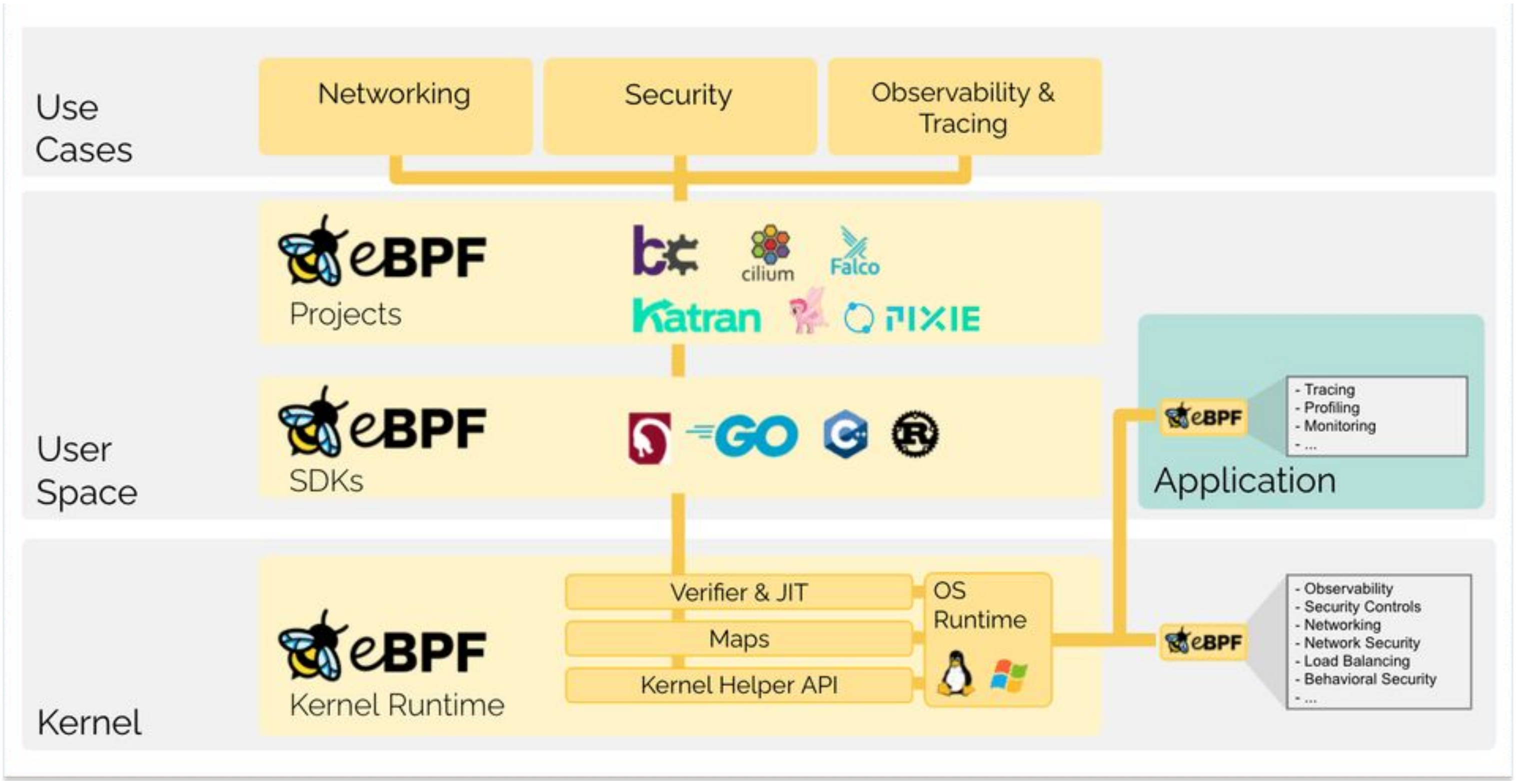
- Collection of APIs, SDKs and Tools
- To Instrument, Generate, Collect and Export Telemetry Data
- Covers Metrics, Logs, Traces
- Formed through merger of OpenTracing and OpenCensus projects
- Coding Required

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

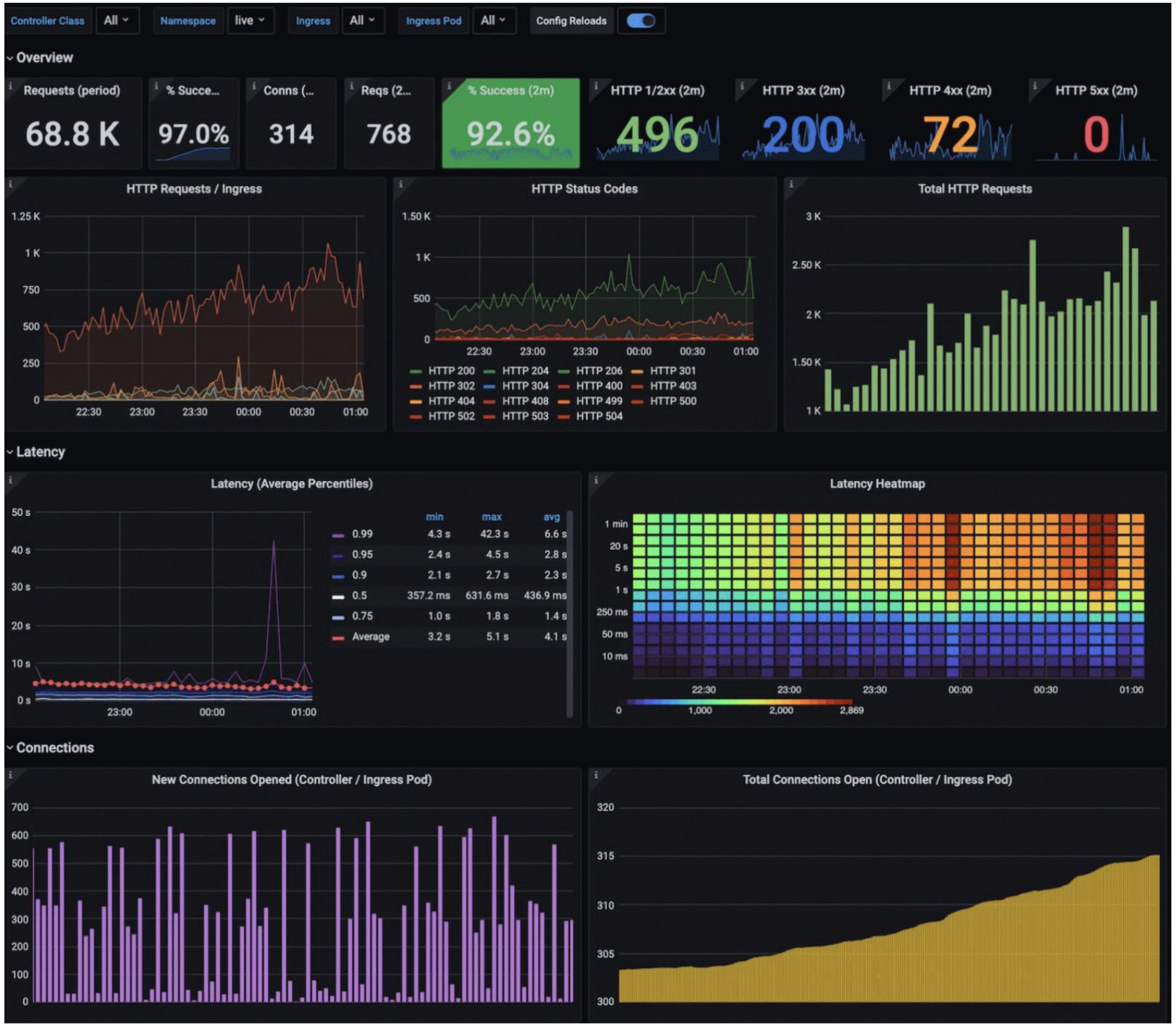
Many solutions support only some kinds of infrastructure well

Swiss Cheese Observability

Full of Holes!

Hard to Use

Overwhelming amount of hard to understand data



Controller Class: All
Namespace: live
Ingress: All
Ingress Pod: All
Config Reloads

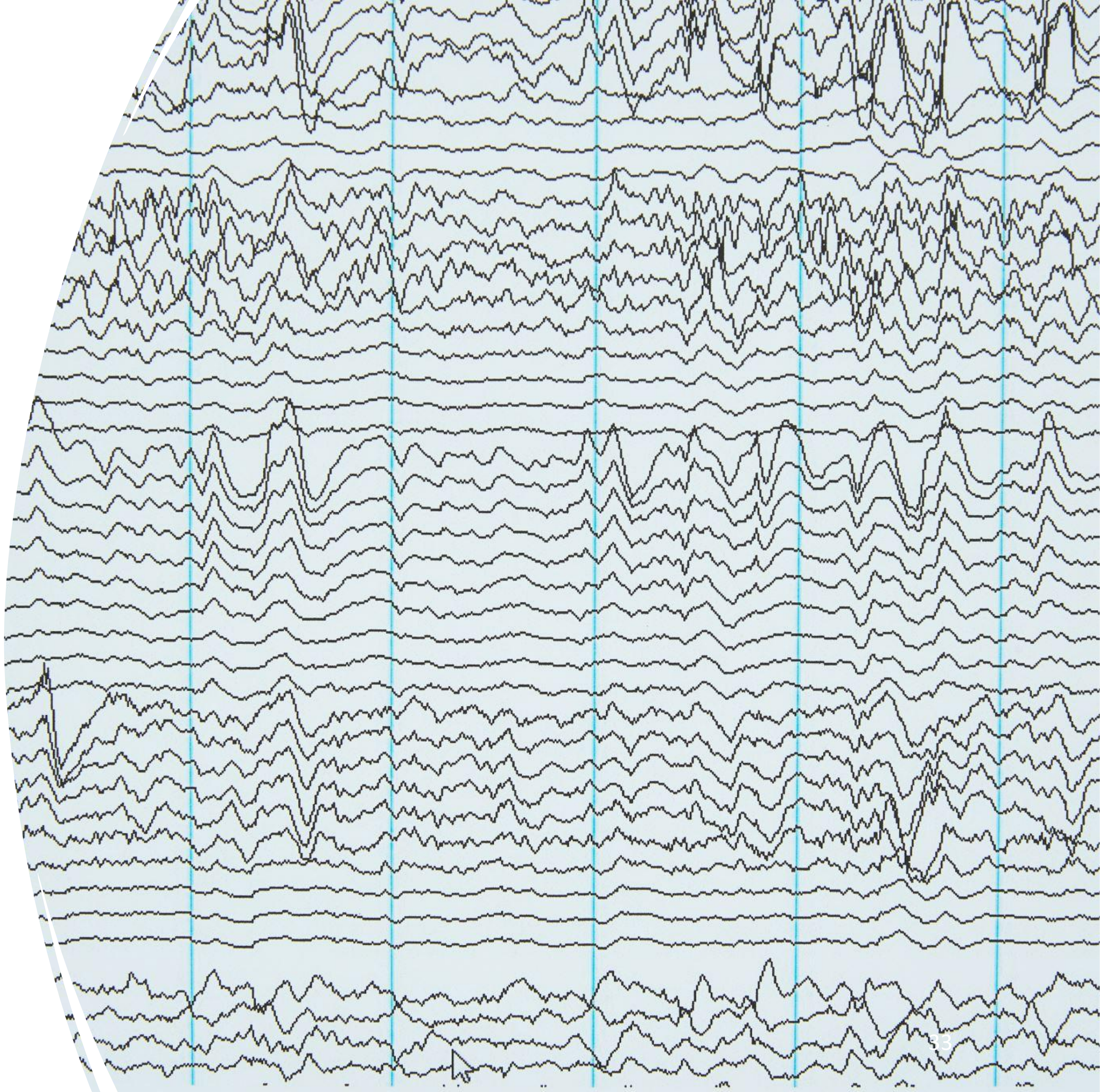
	min	max	avg
0.99	4.3 s	42.3 s	6.6 s
0.95	2.4 s	4.5 s	2.8 s
0.9	2.1 s	2.7 s	2.3 s
0.5	357.2 ms	631.6 ms	436.9 ms
0.75	1.0 s	1.8 s	1.4 s
Average	3.2 s	5.1 s	4.1 s

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

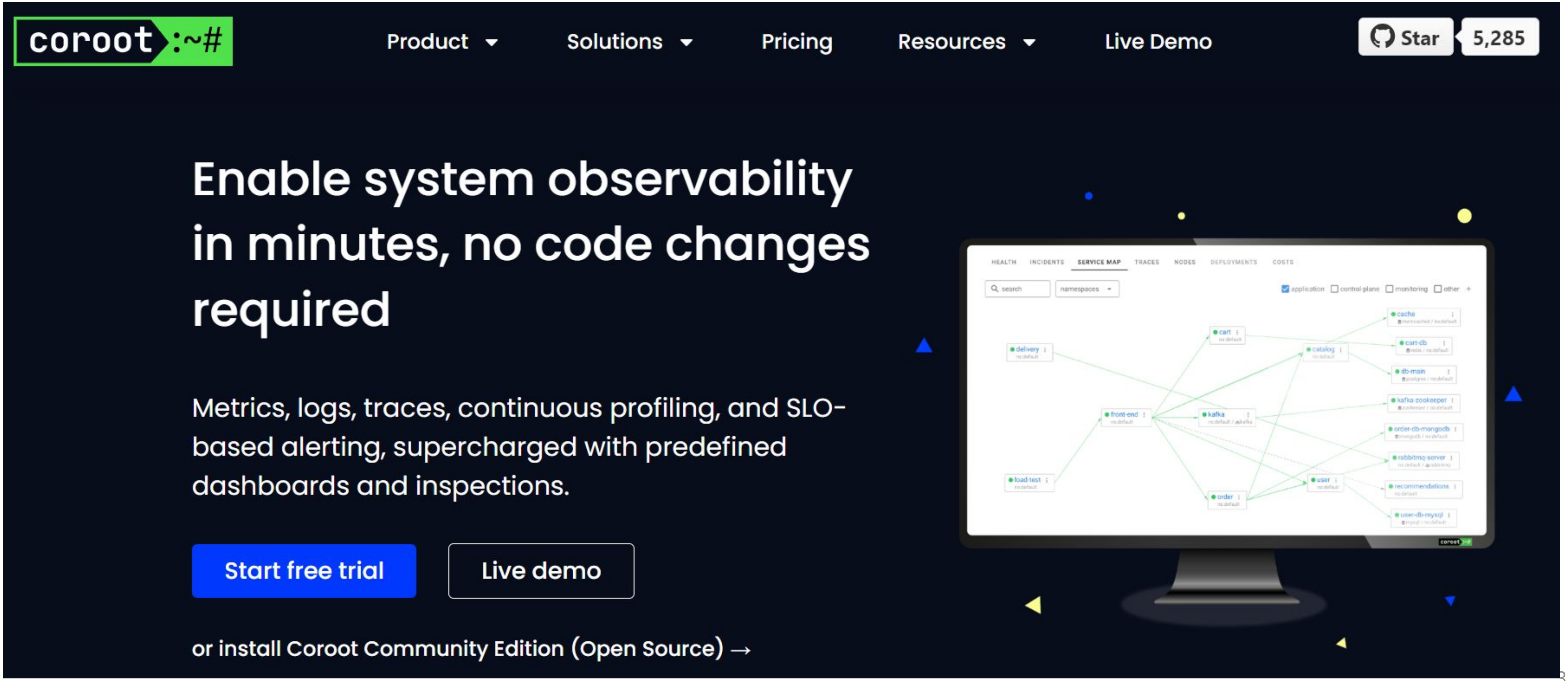
Observability data hard to store and process

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



Enable system observability in minutes, no code changes required

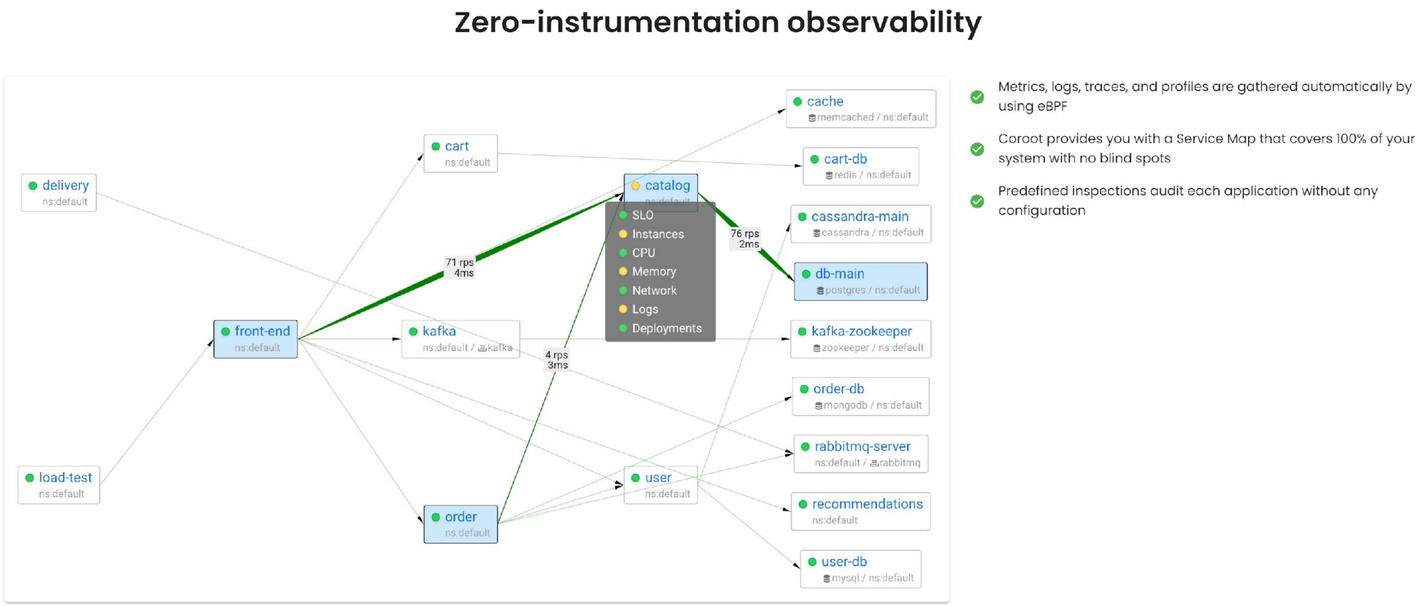
Metrics, logs, traces, continuous profiling, and SLO-based alerting, supercharged with predefined dashboards and inspections.

or install Coroot Community Edition (Open Source) →

Coroot Editions

Open Source

Enterprise



Zero-instrumentation observability

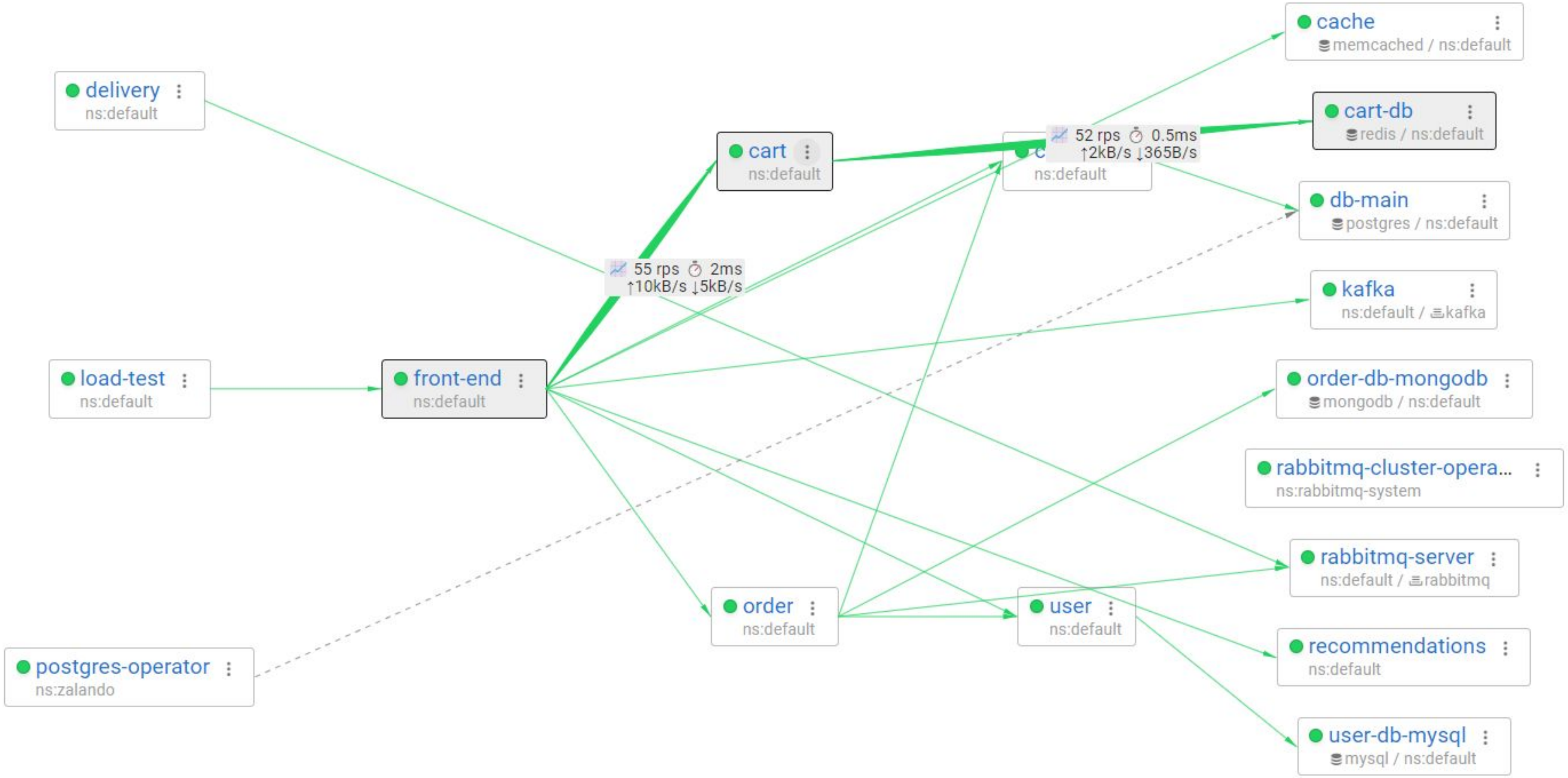
- Metrics, logs, traces, and profiles are gathered automatically by using eBPF
- Coroot provides you with a Service Map that covers 100% of your system with no blind spots
- Predefined inspections audit each application without any configuration

https://coroot.com/

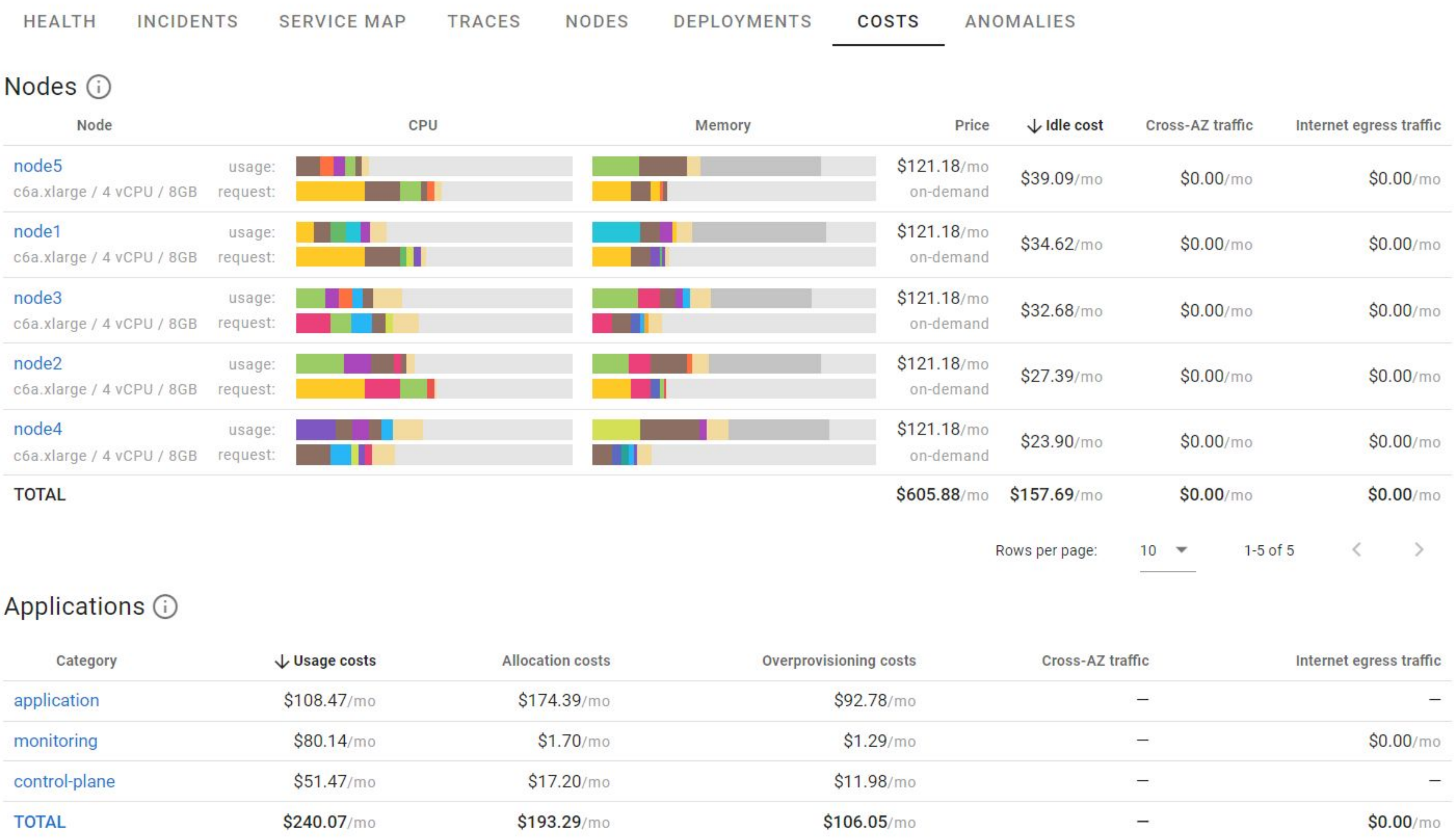
eBPF Magic

- 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



Nodes ⓘ

Node		CPU	Memory	Price	↓ Idle cost	Cross-AZ traffic	Internet egress traffic
node5 c6a.xlarge / 4 vCPU / 8GB	usage: request:			$121.18/mo on-demand	$39.09/mo	$0.00/mo	$0.00/mo
node1 c6a.xlarge / 4 vCPU / 8GB	usage: request:			$121.18/mo on-demand	$34.62/mo	$0.00/mo	$0.00/mo
node3 c6a.xlarge / 4 vCPU / 8GB	usage: request:			$121.18/mo on-demand	$32.68/mo	$0.00/mo	$0.00/mo
node2 c6a.xlarge / 4 vCPU / 8GB	usage: request:			$121.18/mo on-demand	$27.39/mo	$0.00/mo	$0.00/mo
node4 c6a.xlarge / 4 vCPU / 8GB	usage: request:			$121.18/mo 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 BNH Better and Higher