

PERCONA

UNIVERSITY

MARCH 11 BANGKOK, THAILAND

In collaboration with:  solonica

 coroot :~#

 Altinity

 FOSS
asia

What Developers Should Know about Observability



Peter Zaitsev,
Co-Founder at Coroot
March 11, 2026



PERCONA
UNIVERSITY

MARCH 11 BANGKOK, THAILAND

In collaboration with:  solonica

 coroot :~#

 Altinity

 FOSS
asia

What is Observability?



You

what is observability



ChatGPT

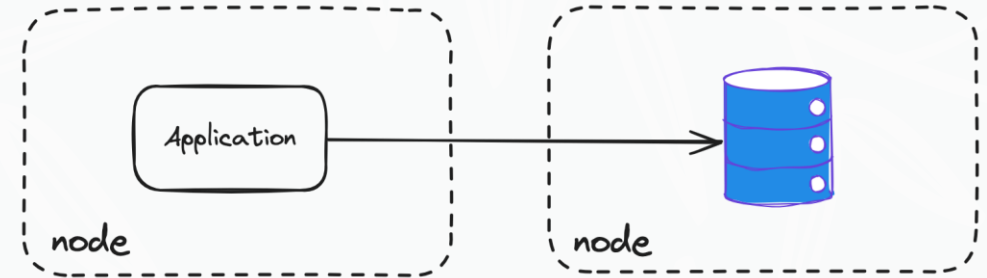
Observability is a concept in systems theory and engineering that refers to the ability to understand the internal state of a system based on its external outputs. In simpler terms, it's about how well you can observe and understand what's happening inside a system just by looking at its behavior from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

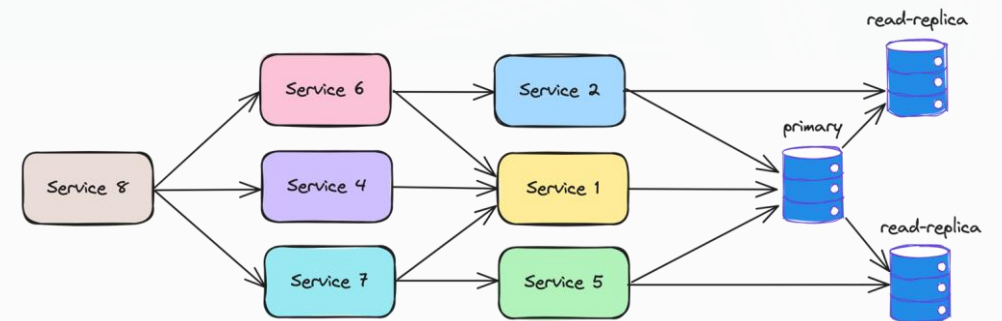
What Is Observability

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

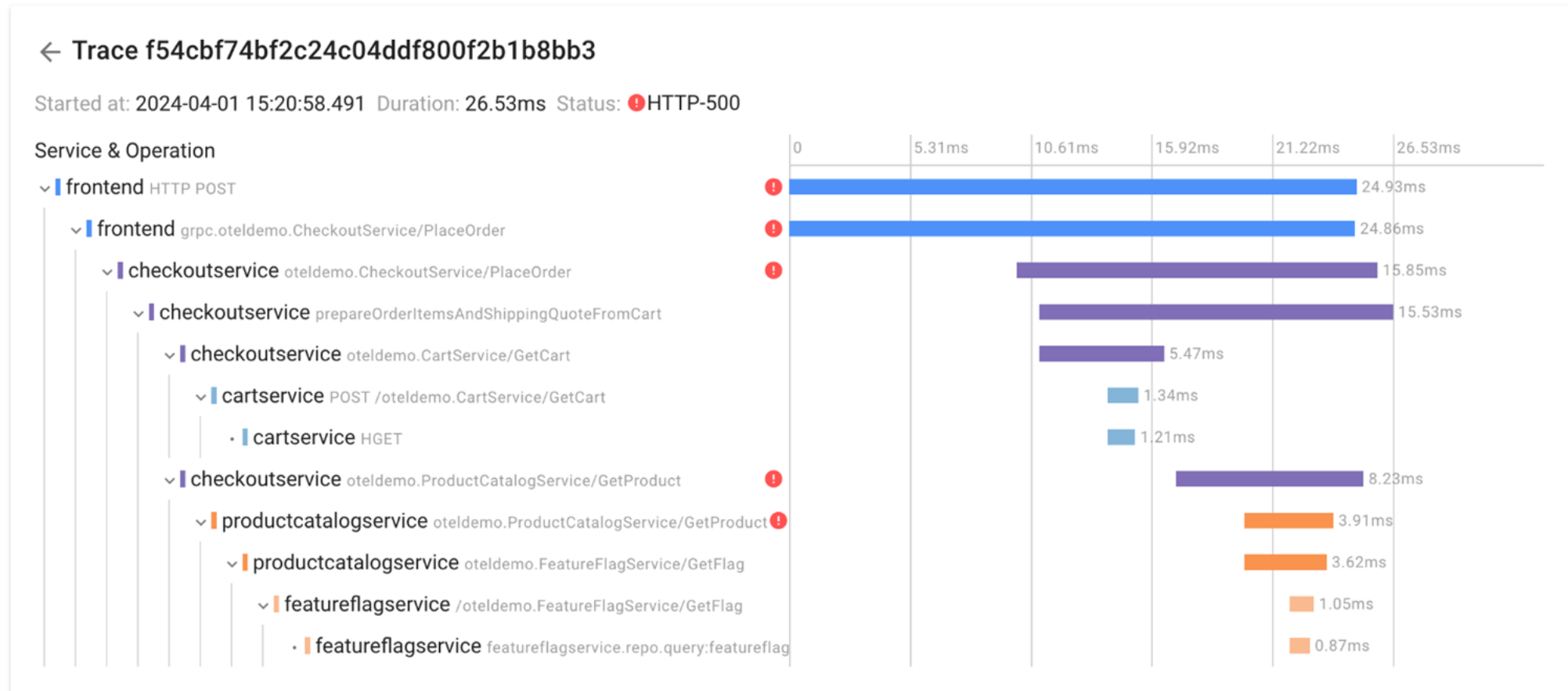
- Structured and Unstructured Format
- Have detailed information on what is happening
- Error logs – contain detailed information about cause of errors
- Expensive to Produce
- Expensive to Store and Analyze
- Sampling and Filtering is often used



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

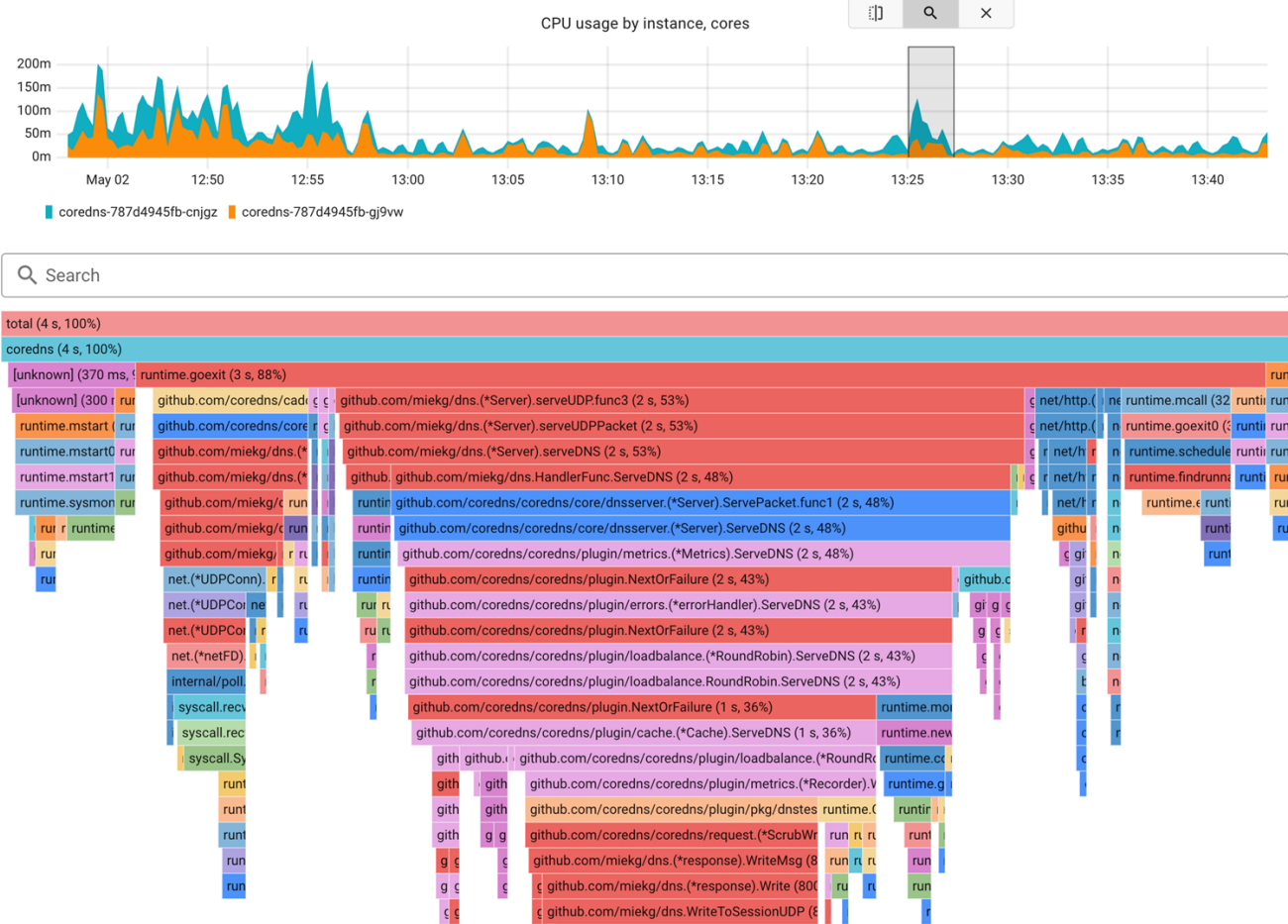




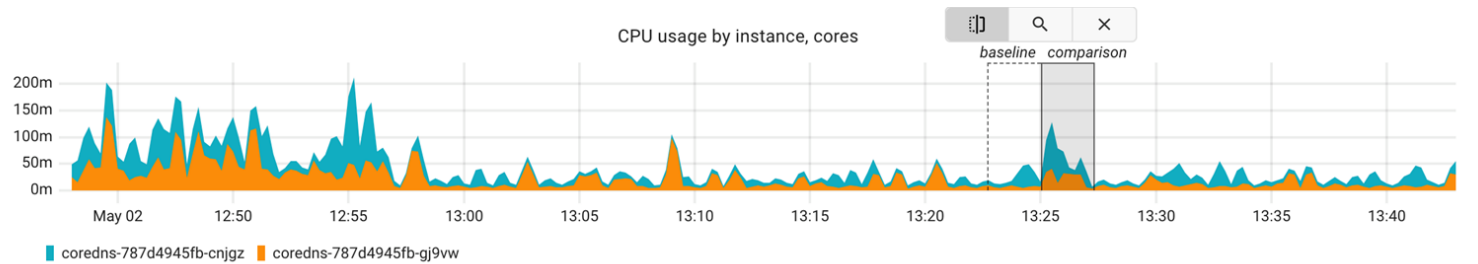
Profiling

- 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



Search

total	coredns	
[unknown] (+0.5)	runtime.goexit	
[unknown] (+0)	github.com/coredns/	github.com/miekg/dns.(*Server).serveUDP.func3 (+6.2%)
runtime.msta	github.com/coredns/	github.com/miekg/dns.(*Server).serveUDPpacket (+5.8%)
runtime.msta	github.com/miekg/dr	github.com/miekg/dns.(*Server).serveDNS (+6.2%)
runtime.msta	github.com/miekg/dr	github.com/miekg/dns.HandlerFunc.ServeDNS (+4.3%)
runtime.sysm	github.com/mi	github.com/coredns/coredns/core/dnsserver.(*Server).ServePacket.func1 (+4.3%)
run	github.com/mi	github.com/coredns/coredns/core/dnsserver.(*Server).ServeDNS (+4.7%)
run	github.com/mi	github.com/coredns/coredns/plugin/metrics.(*Metrics).ServeDNS (+5.0%)
run	net.(*UDPCon	github.com/coredns/coredns/plugin/NextOrFailure (+5.1%)
	net.(*UDPC	github.com/coredns/coredns/plugin/errors.(*ErrorHandler).ServeDNS (+5.1%)
	net.(*netFD	github.com/coredns/coredns/plugin/NextOrFailure (+5.1%)
	internal/pol	github.com/coredns/coredns/plugin/loadbalance.(*RoundRobin).ServeDNS (+5.1%)
	syscall.re	github.com/coredns/coredns/plugin/NextOrFailure (+1.2%)
	syscall.re	github.com/coredns/coredns/plugin/cache.(*Cache).ServeDNS (+0.81%)
	syscall.	github.com/coredns/coredns/plugin/loadbalance.(*RoundRo
	run	github.com/coredns/coredns/plugin/metrics.(*Recorder).W
	run	github.com/coredns/coredns/plugin/pkg/dnstest.(
	run	github.com/coredns/coredns/request.(*ScrubWrite
	run	github.com/miekg/dns.(*response).WriteMsg (+1.
	run	github.com/miekg/dns.(*response).Write (+0.87



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

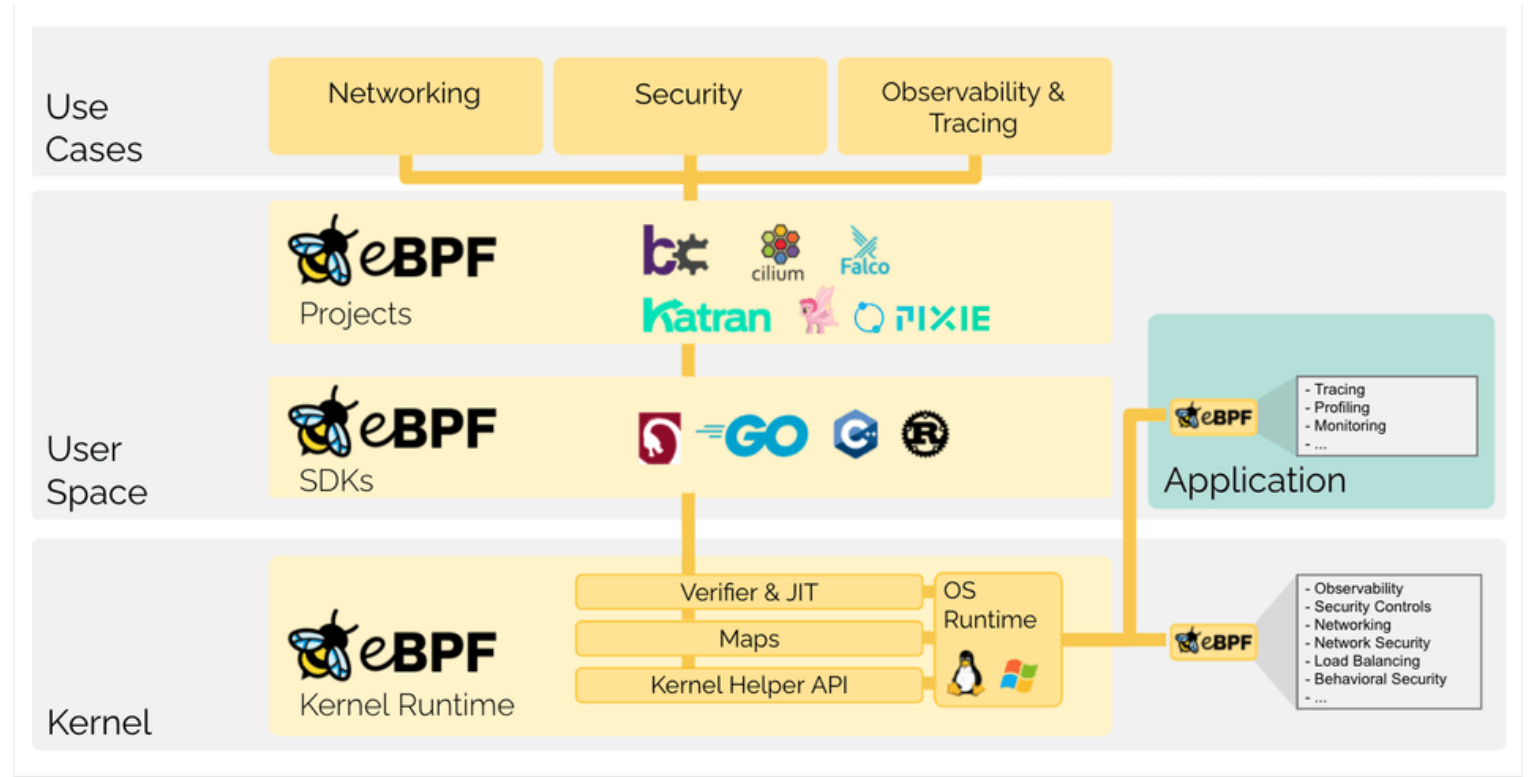
eBPF (Extended Berkeley Packet Filter)

- 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, though 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 Challenges





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

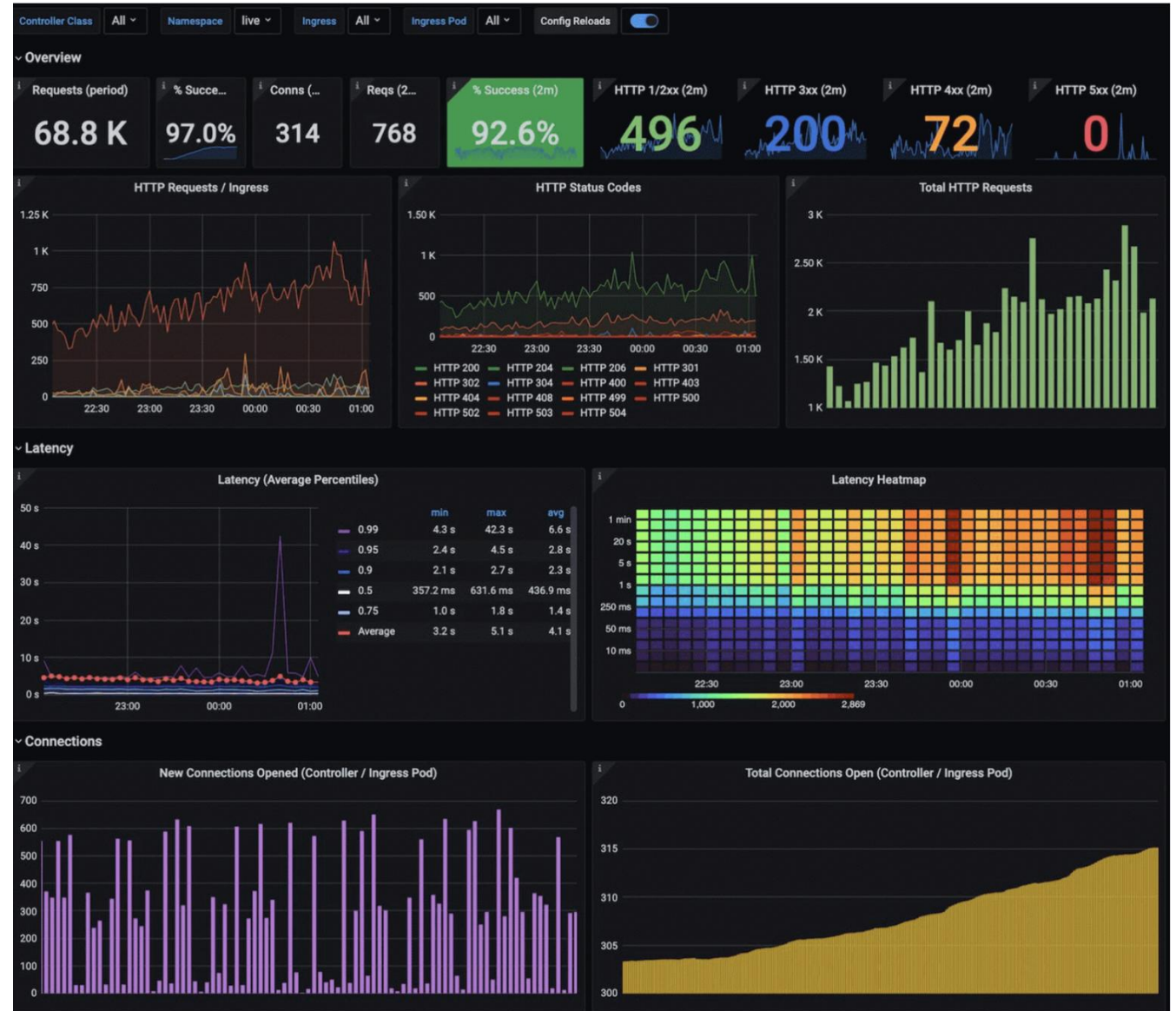
Swiss Cheese Observability

Full of Holes!



Hard to Use

Overwhelming amount of hard to understand data



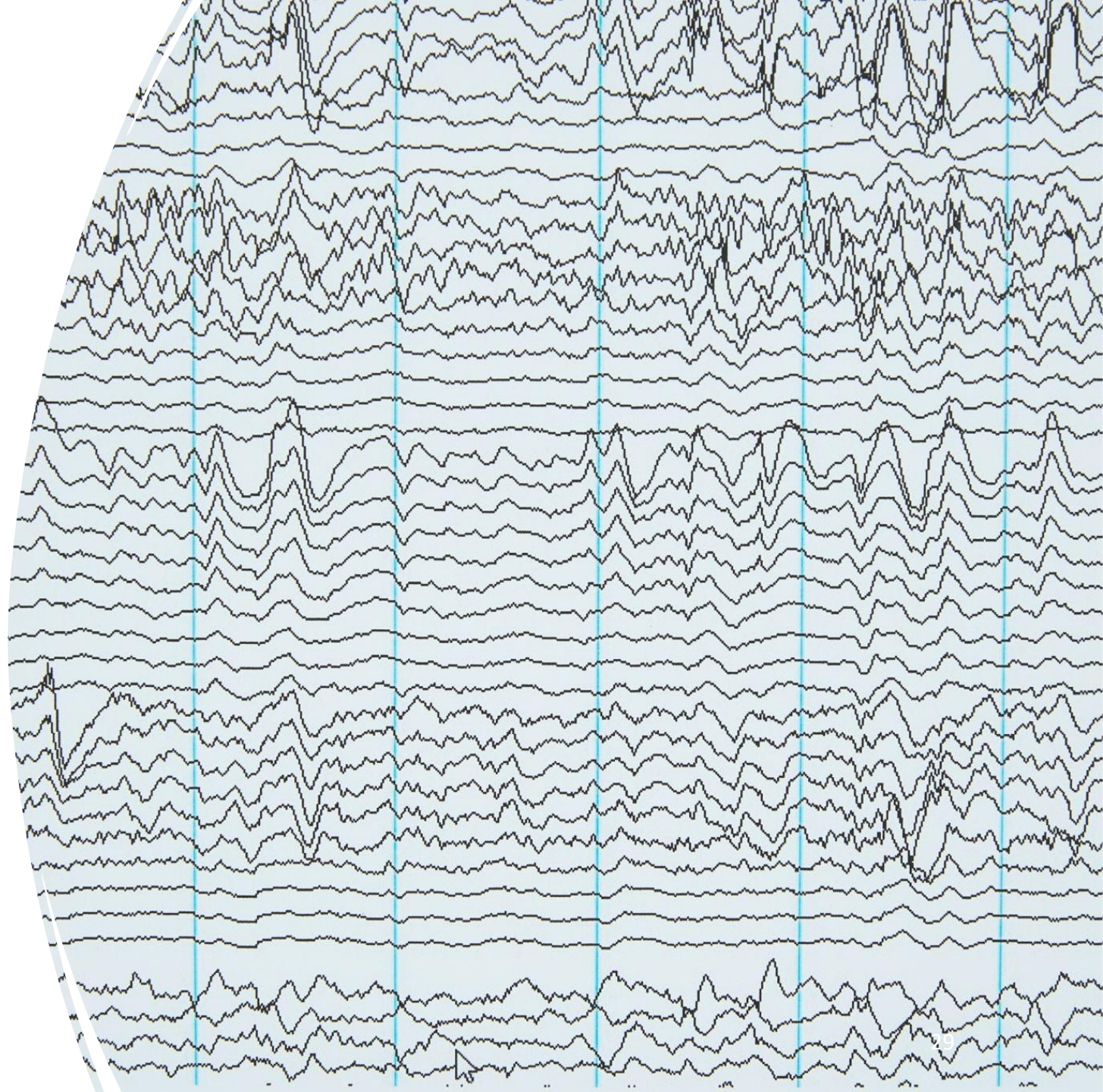
Silos

- 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 ▶ :~#



Full-stack observability in minutes

eBPF-powered and AI-guided — see every metric, log, and trace with **zero code changes**

100%

Coverage

80%

Faster MTTR

0

Code Changes

Try it free

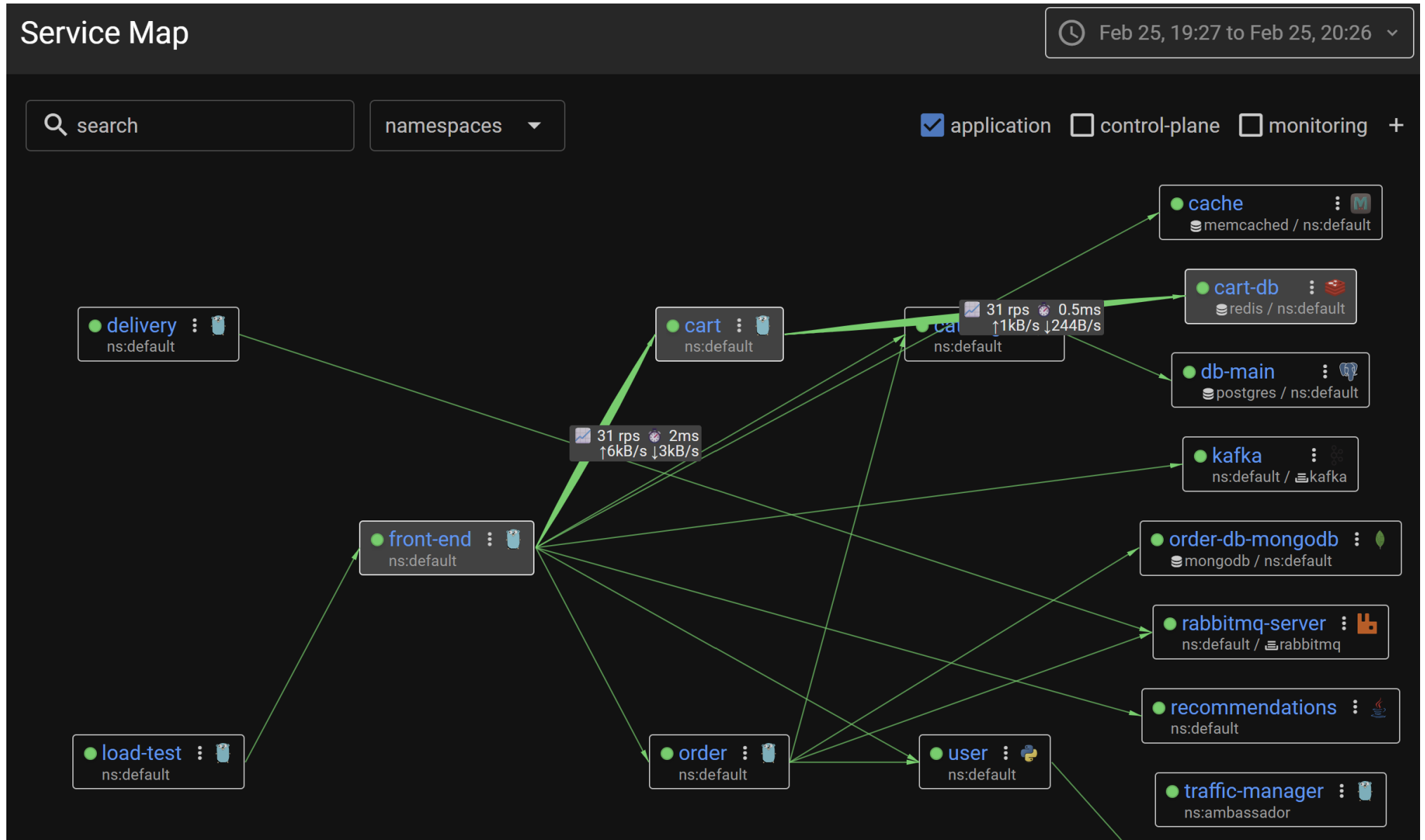
▶ Live Demo



eBPF Magic

- Automatic Instrumentation with eBPF
- Use 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 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

Highlight Key Risks

coroot Enterprise Edition

Risks

Feb 25, 19:27 to Feb 25, 20:26

Go to... ctrl+k

search namespaces

application control-plane monitoring +

3 Critical 35 Warning 0 Dismissed Show dismissed

Application	Application type	Risk category	Description
cart-db	redis	Availability	Redis isn't replicated - data loss possible
honeypot-mysql	mysql	Security	Publicly exposed database on 6 IPs 37.27.188.133 through the NodePort service <u>honeypot-mysql</u>
honeypot-mysql	mysql	Availability	Mysql isn't replicated - data loss possible
cache	memcached	Availability	Single instance - not resilient to node failure
cart	golang	Availability	All instances on one node - not resilient to node failure
cart-db	redis	Availability	Single instance - not resilient to node failure
delivery	golang	Availability	All instances on one node - not resilient to node failure
load-test	golang	Availability	Single instance - not resilient to node failure
rabbitmq-server	rabbitmq	Availability	Single instance - not resilient to node failure
traffic-manager	golang	Availability	Single instance - not resilient to node failure
user	python	Availability	All instances on one node - not resilient to node failure

Rows per page: 50 1-11 of 11

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



AI For Even Faster and Easier Troubleshooting

Evidence from Logs

Application logs confirm the resource starvation:

- Database connection failures: "failed to connect to db-main: operation was canceled"
- Context timeouts: "context canceled" and "context deadline exceeded"
- HTTP 500/502 errors due to upstream service failures

The Kubernetes events show the `analytics-updater` job was created and completed during the incident timeframe, correlating with the performance degradation.

Immediate Fixes

Monitor and limit resource usage of the `analytics-updater` job:

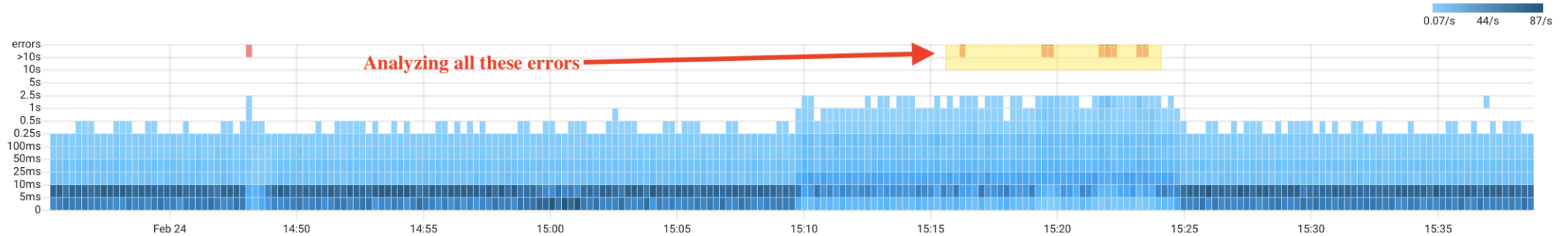
```
# Add resource limits to analytics-updater CronJob
resources:
  limits:
    cpu: "500m"
    memory: "512Mi"
  requests:
    cpu: "100m"
    memory: "256Mi"
```

Consider spreading workloads across nodes using node affinity rules to prevent resource contention on a single node.

Summary for traces: errors

Integrate OpenTelemetry

Latency & Errors heatmap, requests per second



OVERVIEW TRACES **ERROR CAUSES** LATENCY EXPLORER COMPARE ATTRIBUTES

Filters: +

Selection: time Feb 24, 15:15 – Feb 24, 15:24 where (trace duration 10s – Inf or trace status is Error) ×

Exclude auxiliary requests (from monitoring, control plane, etc)

Deepest failing span

Sample trace for cross-checking

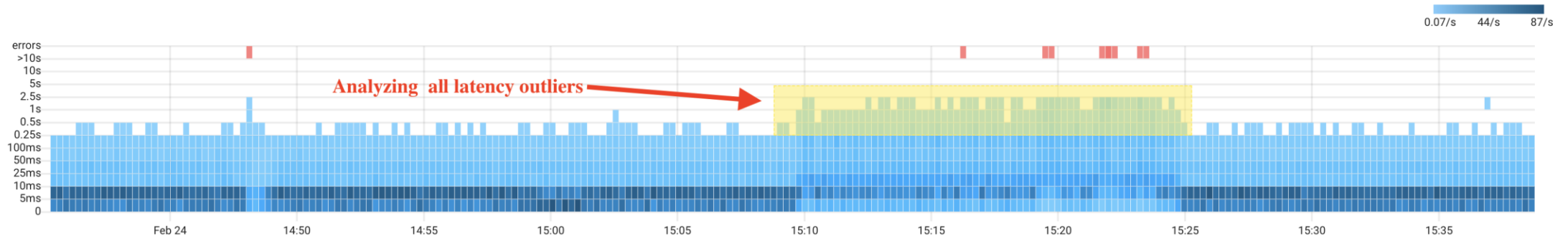
This section highlights the underlying reasons why traces within the selected range contain errors. It identifies the tracing spans where errors originated.

Service Name	Span	Error	Sample Trace	Percentage ↓
catalog	gorm.Query • db.system: postgresql	failed to connect to `host=db-main user=demo database=demo`: dial error (dial tcp 10.43.32.39:54...	30461830	98%
front-end	HTTP GET • net.peer.name: catalog	context canceled	d994f318	1.3%
front-end	HTTP POST • net.peer.name: order	context canceled	1bf3c919	<1%

Summary for traces: latency

Integrate OpenTelemetry

Latency & Errors heatmap, requests per second



OVERVIEW TRACES ERROR CAUSES **LATENCY EXPLORER** COMPARE ATTRIBUTES

Filters: +

Selection: time Feb 24, 15:08 – Feb 24, 15:25 where (trace duration 250ms – 5s) ×

Baseline: other events within the time window

Exclude auxiliary requests (from monitoring, control plane, etc)

View: FlameGraph Diff

The flamegraph shows most of the time is spent waiting for Postgres to respond.

Sample trace for cross-checking

This section shows the latency FlameGraph for the selected traces. A wider frame indicates greater time consumption by that tracing span.

Select instance

Search

total (7 min, 100%)

front-end: GET /catalog/ {net.host.name=front-end} (7 min, 95%)

front-end: HTTP GET {net.peer.name=catalog} (7 min, 92%)

catalog: GE catalog: GET /catalog/brands {net.host.name=catalog} (4 min, 56%)

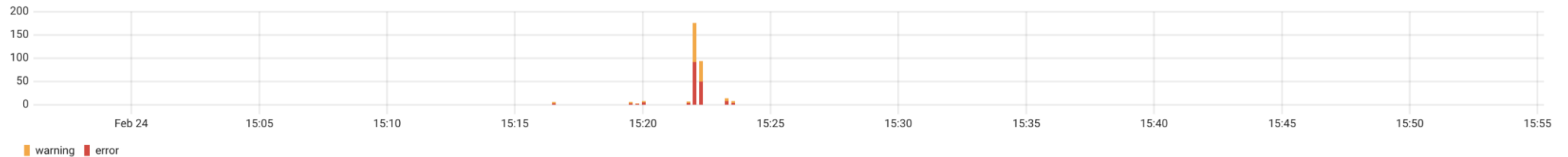
catalog: g catalog: gorm.Query {db.system=postgres} (4 min, 56%)

catalog: GET /catalog/product/{product} {net.host.name=catalog} (2 min, 30%)

catalog: GET /catalog/product/{product} {db.system=postgres} (2 min, 29%)

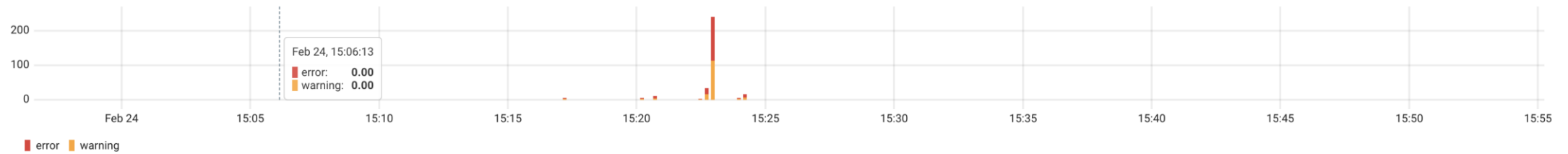
Zoom in
Open a sample trace

Raw logs: hard to see the big picture



Date	Message
Feb 24 15:23:43	E0224 18:23:42.386569 1 db.go:158] db is not available: failed to connect to `host=db-main user=demo database=demo`: dial error (dial tcp 10.43.32.39:5432: operation was canceled)
Feb 24 15:23:42	E0224 18:23:41.780141 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 2041ms
Feb 24 15:23:42	E0224 18:23:41.988125 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1872ms
Feb 24 15:23:41	E0224 18:23:40.917103 1 db.go:158] db is not available: failed to connect to `host=db-main user=demo database=demo`: dial error (dial tcp 10.43.32.39:5432: operation was canceled)
Feb 24 15:23:41	E0224 18:23:40.848388 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1984ms
Feb 24 15:23:40	E0224 18:23:39.695263 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1986ms
Feb 24 15:22:35	E0224 18:22:35.118808 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1953ms
Feb 24 15:22:35	E0224 18:22:34.588278 1 db.go:158] db is not available: timeout: context canceled
Feb 24 15:22:34	E0224 18:22:33.496789 1 middleware.go:152] GET /catalog/product/1751599 HTTP/1.1 500 1 1992ms
Feb 24 15:22:34	E0224 18:22:33.702132 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1997ms
Feb 24 15:22:34	E0224 18:22:33.555372 1 middleware.go:152] GET /catalog/brand/Omba HTTP/1.1 500 1 1997ms
Feb 24 15:22:33	E0224 18:22:33.156450 1 middleware.go:152] GET /catalog/brand/Blogtag HTTP/1.1 500 1 1984ms
Feb 24 15:22:33	E0224 18:22:32.920878 1 middleware.go:152] GET /catalog/product/1815059 HTTP/1.1 500 1 2005ms
Feb 24 15:22:32	E0224 18:22:32.803059 1 middleware.go:152] GET /catalog/product/1820359 HTTP/1.1 500 1 2071ms
Feb 24 15:22:32	E0224 18:22:31.917669 1 db.go:158] db is not available: failed to connect to `host=db-main user=demo database=demo`: dial error (dial tcp 10.43.32.39:5432: operation was canceled)
Feb 24 15:22:32	E0224 18:22:32.285094 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1997ms
Feb 24 15:22:32	E0224 18:22:31.589077 1 db.go:158] db is not available: failed to connect to `host=db-main user=demo database=demo`: dial error (dial tcp 10.43.32.39:5432: operation was canceled)
Feb 24 15:22:31	E0224 18:22:31.486831 1 middleware.go:152] GET /catalog/product/1831998 HTTP/1.1 500 1 1947ms
Feb 24 15:22:31	E0224 18:22:31.542018 1 middleware.go:152] GET /catalog/product/1777958 HTTP/1.1 500 1 2002ms
Feb 24 15:22:31	E0224 18:22:30.825780 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1991ms
Feb 24 15:22:31	E0224 18:22:31.072190 1 middleware.go:152] GET /catalog/brands HTTP/1.1 500 1 1987ms

Extracting repeated patterns



Log-based alerting

Alert `zvq9a100eewp` Suppressed ✕

Rule [Log errors](#)

Application `ssh`

Severity Warning

Opened at Feb 17, 16:35:13 (6d ago)

Duration 6d (ongoing)

Suppressed by AI

Summary new error in the logs (10 messages in the last 1h)

Description A new log pattern has been detected. Review the log messages to determine whether this indicates an error, a misconfiguration, or expected behavior. Suppress the alert if this is not a problem.

Sample `error: maximum authentication attempts exceeded for invalid user admin from 213.66.113.170 port 59990 ssh2 [preauth]` 📄

AI analysis This is a very common SSH brute-force login attempt from an external IP trying the "admin" username. These are constant background noise on any internet-facing SSH server and are not indicative of a real incident. The attempt failed (the user is "invalid"), meaning the username doesn't even exist on the system. This is expected behavior and is best handled by automated tools like fail2ban or firewall rules rather than manual investigation. No action is needed unless there's a sudden massive spike in volume suggesting a targeted attack.

Notifications -

Messages

Time	Messages
15:30	0
15:31	2
15:32	2
15:33	2
15:34	3
15:35	2
15:36	1
15:37	3
15:38	2
15:39	1
15:40	0
15:41	0
15:42	0
15:43	0
15:44	0
15:45	0

Show messages

↺ Reopen Close

Getting notified about critical issues without defining them in advance

Alert 16bozq7fa7px Firing ×

Rule [Log errors](#)

Application [k3s-agent](#)

Severity Warning

Opened at Feb 17, 16:31:46 (6d ago)

Duration 6d (ongoing)

Summary new error in the logs (140 messages in the last 1h)

Description A new log pattern has been detected. Review the log messages to determine whether this indicates an error, a misconfiguration, or expected behavior. Suppress the alert if this is not a problem.

Sample `E0206 18:06:47.656473 435805 pod_workers.go:1298] "Error syncing pod, skipping" err="failed to \"StartContainer\" for \"mysql\"`

AI analysis A MySQL secondary database container (user-db-mysql-secondary-0) is in CrashLoopBackOff with a 5-minute back-off interval, indicating it has been failing repeatedly. This is a database pod, which likely serves user-facing workloads, and a failed secondary could mean degraded redundancy or read capacity. An SRE should investigate the root cause (e.g., misconfiguration, resource limits, storage issues, or upstream primary problems) to restore the database replica before it impacts availability or data durability.

Notifications -





Power and Simplicity

- Ensure your observability is complete, with no blind spots
- 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 collaboration with:  solonica

 coroot :~#

 Altinity

 FOSS
asia