O11y in One:

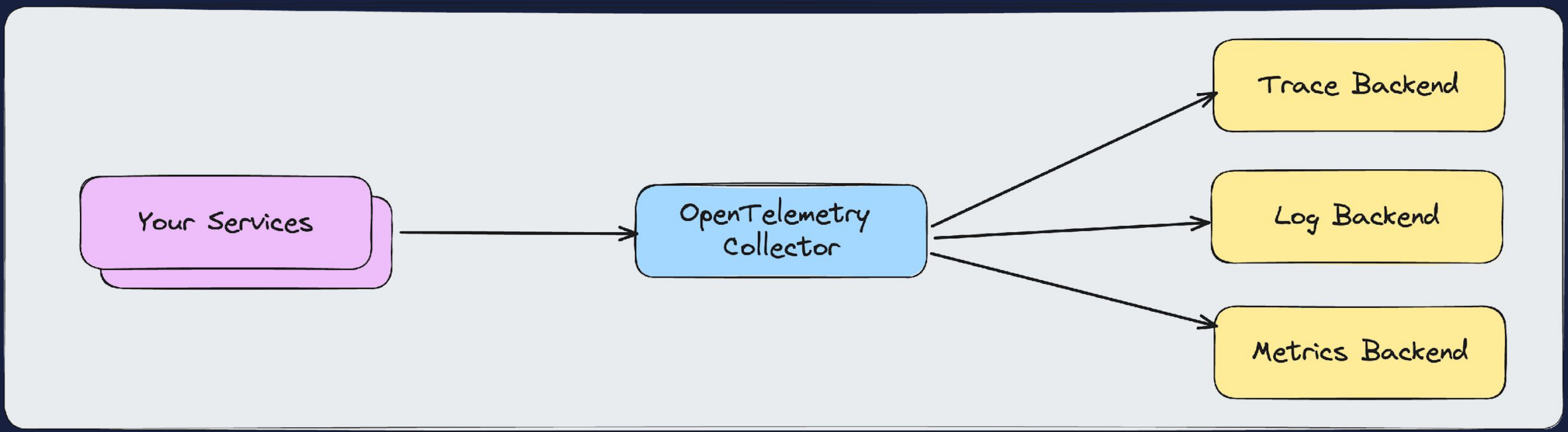
ClickHouse® as a unified telemetry database

Josh Lee

Open Source Advocate

Altinity

How I usually start...



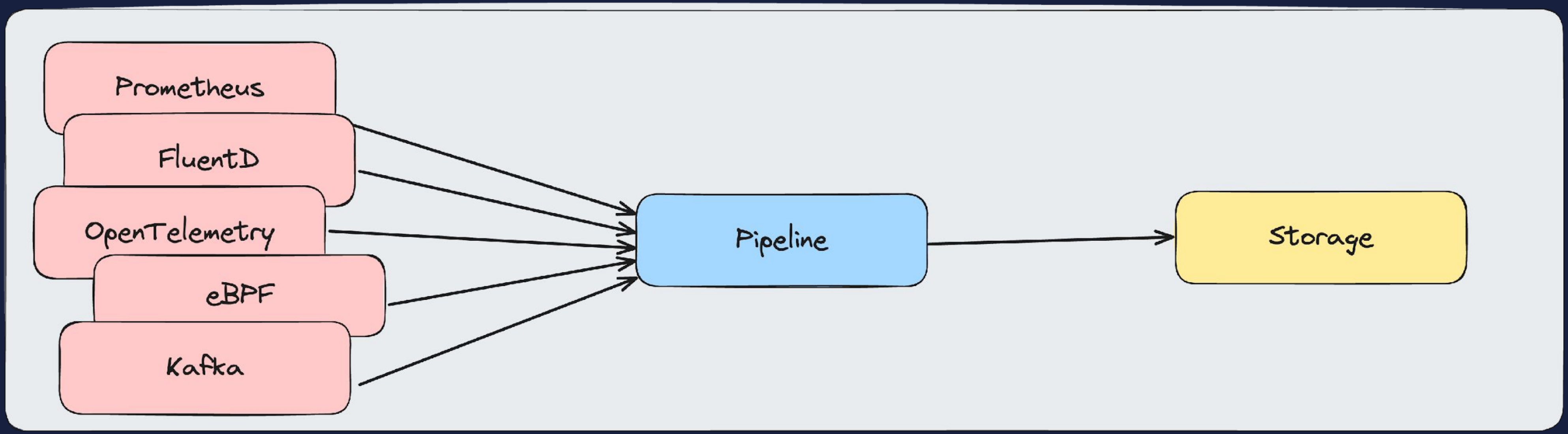
"The OpenTelemetry project does not include any kind of database or backend UI."

6

Minimum no. of o11y tools deployed by a typical organization

— Grafana State of Observability Report

What we really need...



Agenda

Challenges with Disparate Telemetry Systems

Josh Lee • Altinity • Database Frontiers • 15 Oct 2025

What are we storing?

Metrics, traces, logs, profiles, events

Resource metadata

Graphs & topologies

Snapshots & deltas

Configuration

What do we need for observability?

Fast streaming writes
Efficient compression & storage
Time-oriented management
“Real-time” analytics

"Anything you can do with a group by, that's what analytics is"

—Peter Marshall

More Requirements

Fast multi-row analytics
Full-text search
Tag/label search
Fast, frequent "last point" reads
Updates?

Is There a Silver Bullet?

No. Obviously.

… but ClickHouse comes pretty close.

Introducing ClickHouse

- SQL-compatible
- Massively scaleable
- Really, really fast

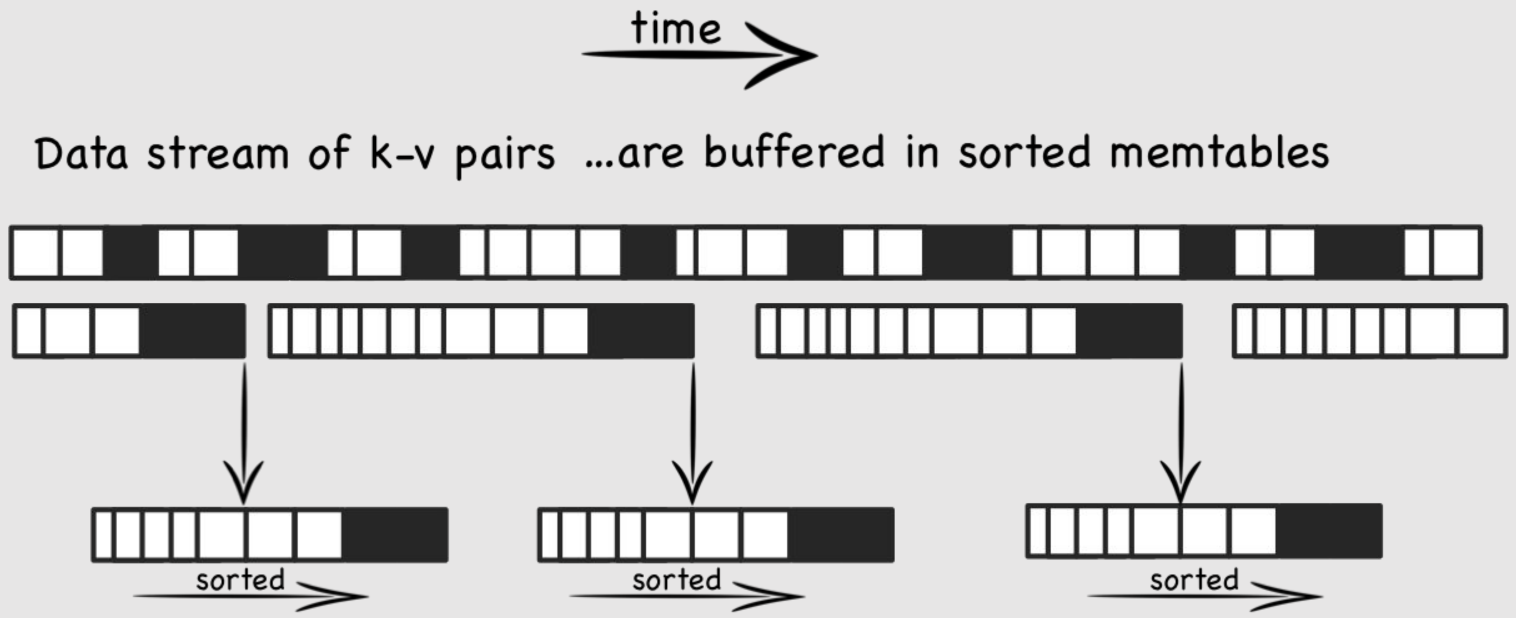
ClickHouse for Observability

Telemetry is WORM

Write-Once, Read-Many

B-Trees: Optimized for Reads

Log-Structured Merge Trees: Optimized for ingestion

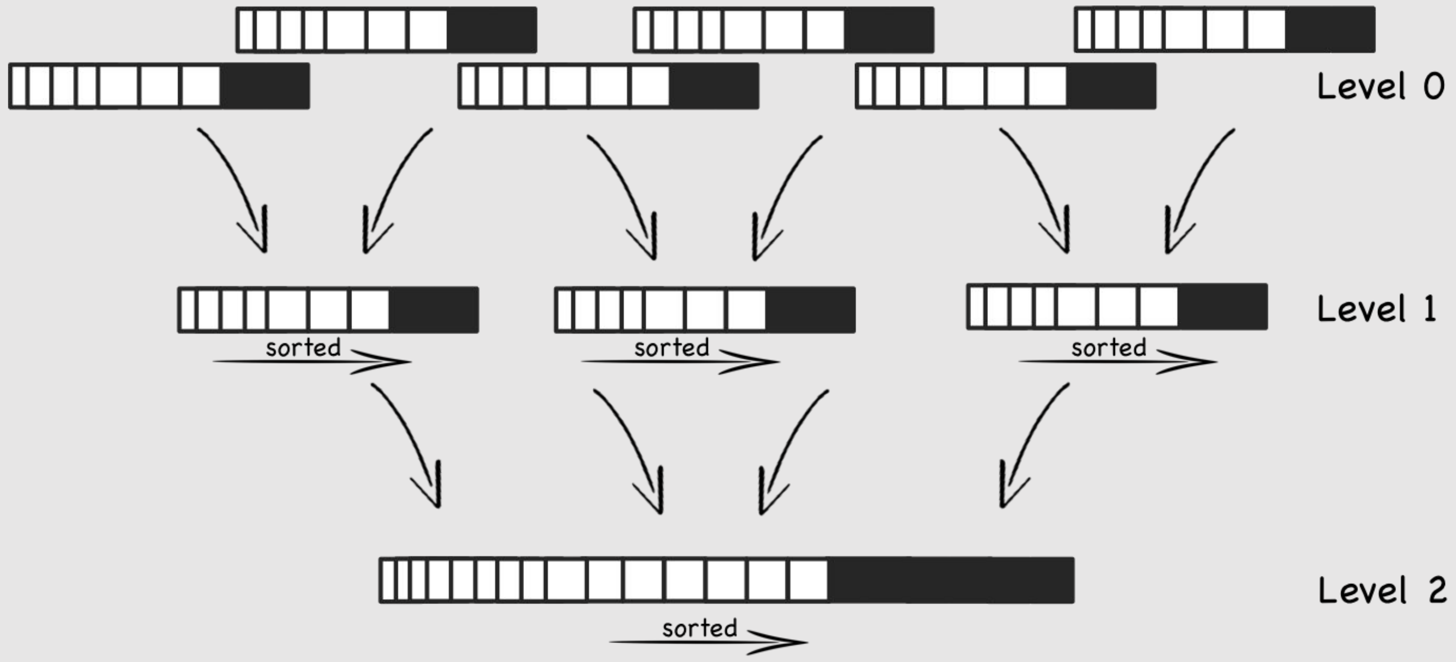


Data stream of k-v pairs ...are buffered in sorted memtables

and periodically flushed to disk...forming a set of small, sorted files.

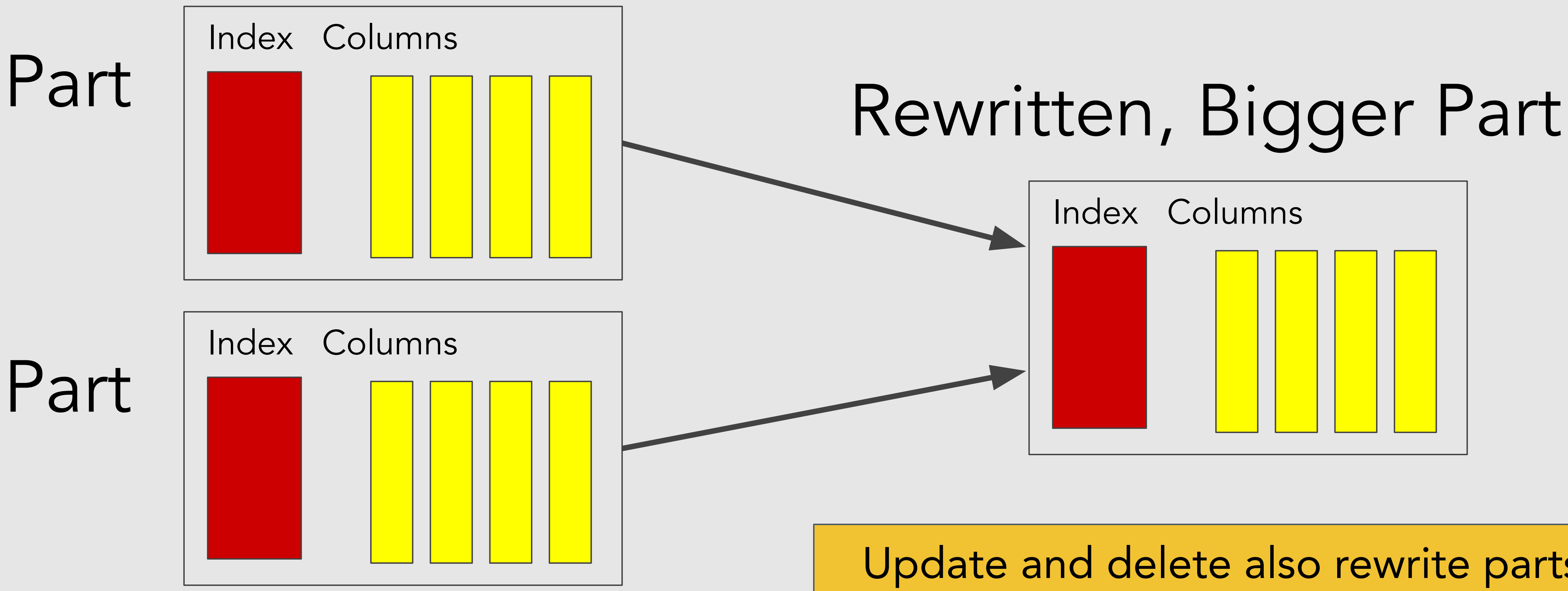
http://www.benstopford.com/2015/02/14/log-structured-merge-trees/

Log-Structured Merge Trees: Background compaction

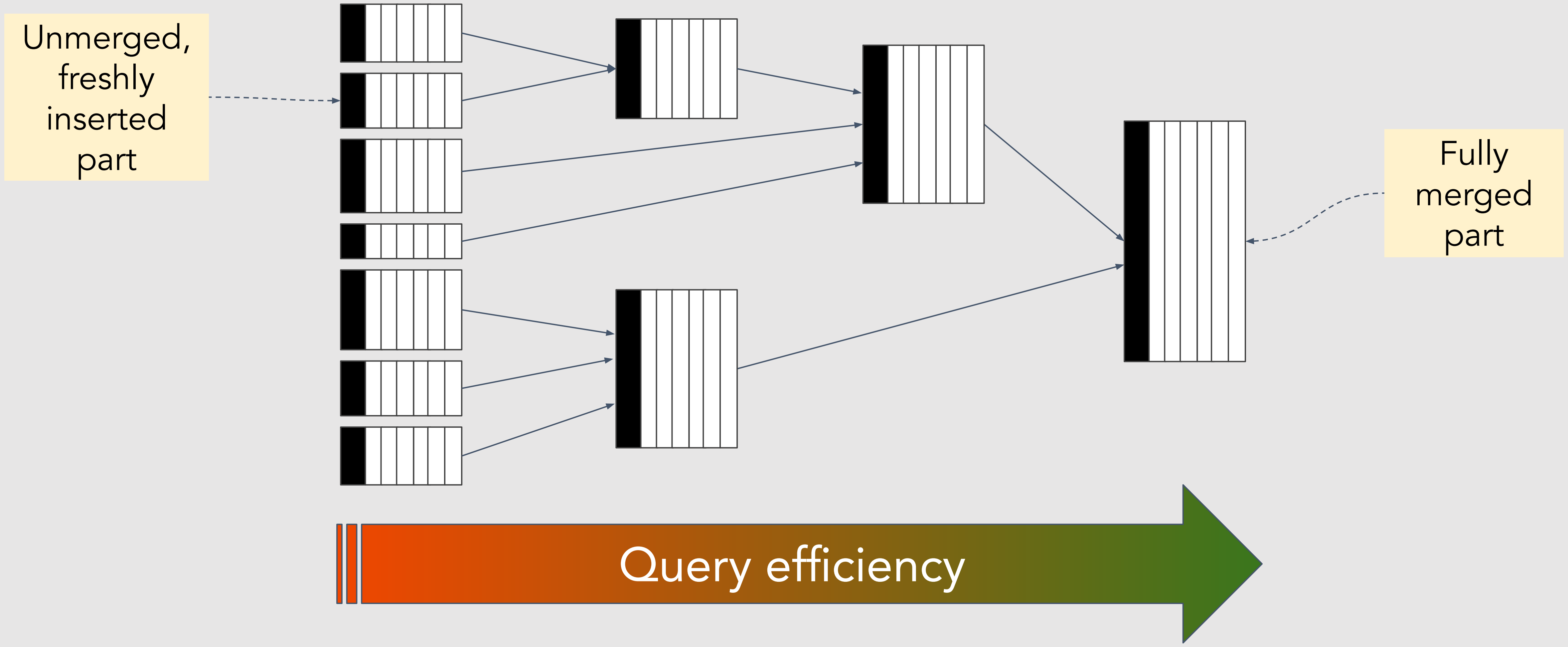


Compaction continues creating fewer, larger and larger files

http://www.benstopford.com/2015/02/14/log-structured-merge-trees/



Update and delete also rewrite parts


Unmerged, freshly inserted part
Fully merged part
Query efficiency

ClickHouse for Observability

How does this help?

- Fast writes
- Time-friendly
- Easy cleanup
- Cost-effective

Data Transformation & Management

- Materialized Views
- TTL
- Tiered storage
    - Even to S3! 👀

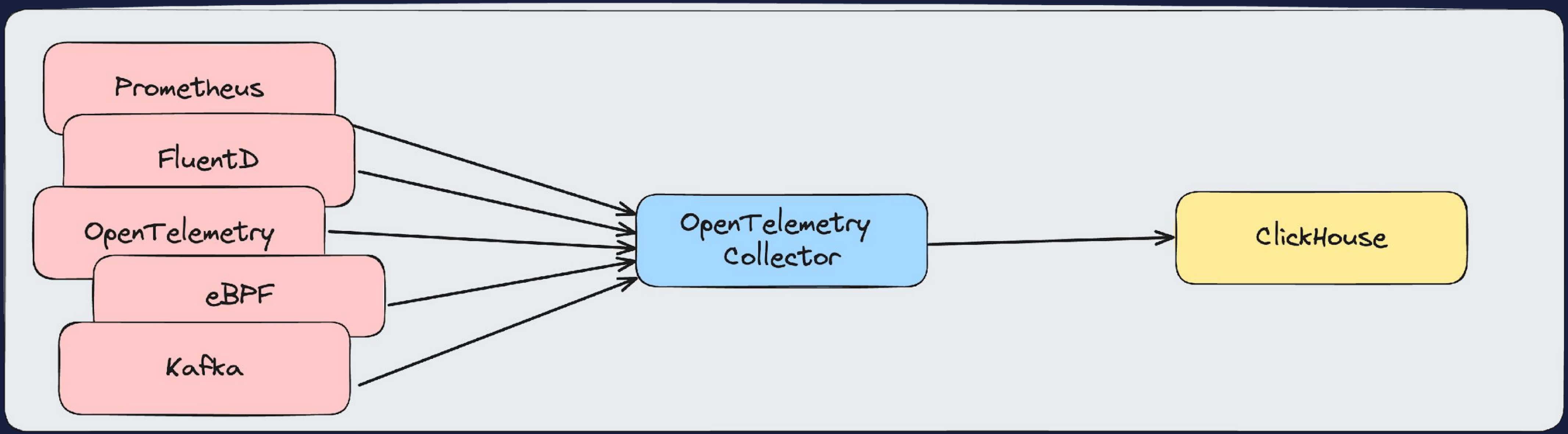
ClickHouse for Observability

Integrations

- Grafana
- Jaeger
- Loki, Tempo, Prometheus (via QRYN)
- Kafka
- OpenTelemetry!

ClickHouse for Observability

Integrations via OpenTelemetry



ClickHouse for Observability

More Benefits

- Excellent compression, even with variable schemas
- Practically unlimited cardinality
- Horizontally scalable ingestion & querying

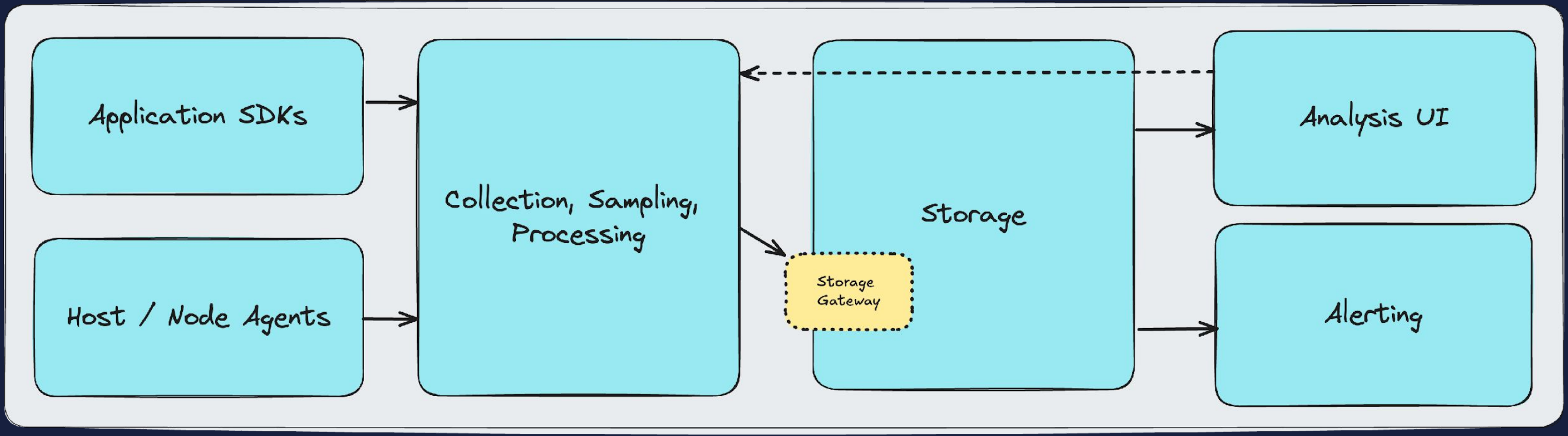
ClickHouse for Observability

Challenges

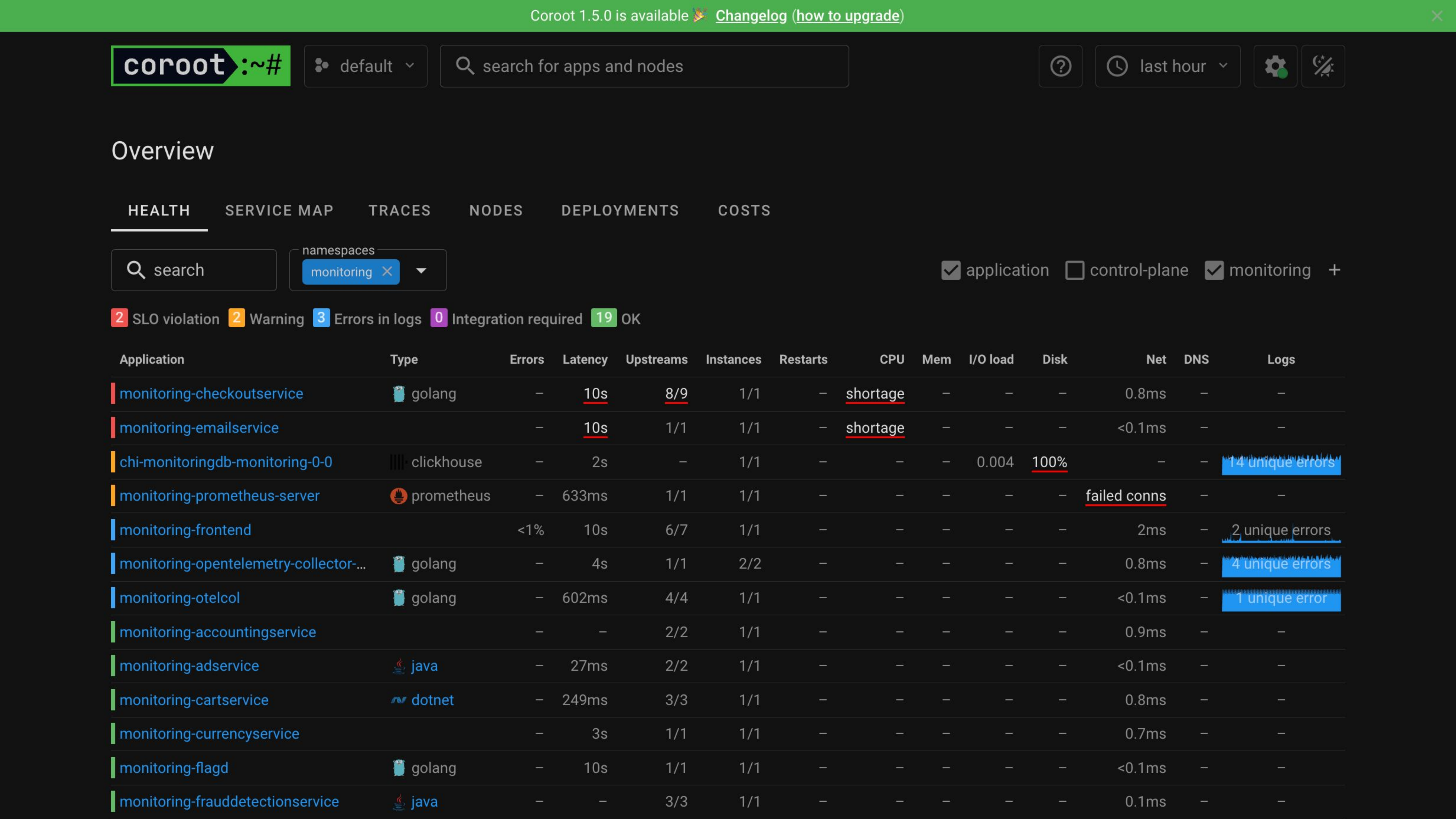
- SQL is not PromQL*
- Overly complex for small data volumes*
- Not a turn-key solution

"The OpenTelemetry project does not include any kind of database or *backend UI*."

We need a complete observability solution



SigNoz
Coroot
QRYN
HyperDX / ClickStack
DIY

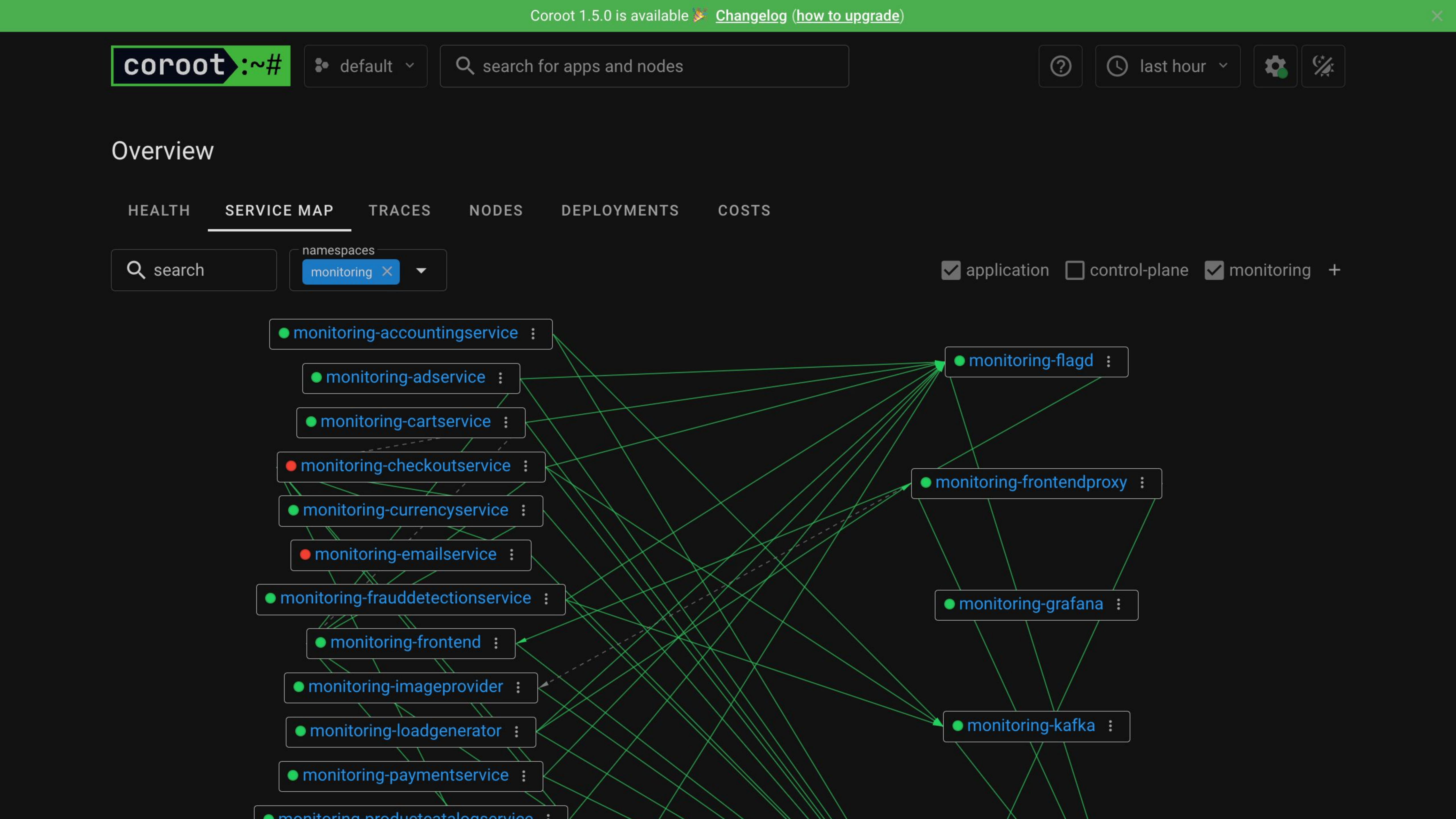

Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)
coroot :~#
default
search for apps and nodes
last hour

Overview

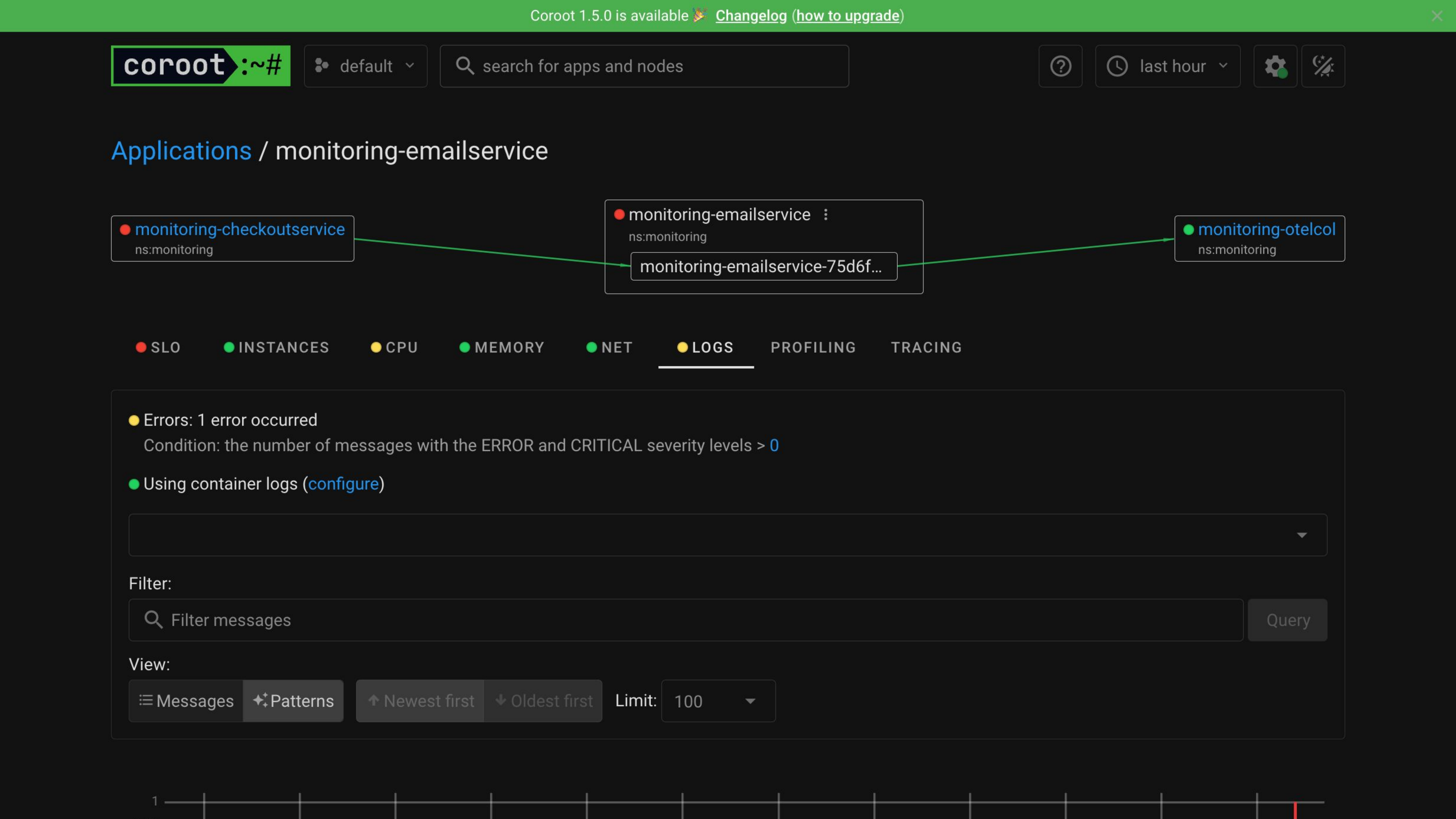
[✓ application] control-plane [✓ monitoring] +

2 SLO violation 2 Warning 3 Errors in logs 0 Integration required 19 OK

Application	Type	Errors	Latency	Upstreams	Instances	Restarts	CPU	Mem	I/O load	Disk	Net	DNS	Logs
monitoring-checkoutservice	golang	–	10s	8/9	1/1	–	shortage	–	–	–	0.8ms	–	–
monitoring-emailservice		–	10s	1/1	1/1	–	shortage	–	–	–	<0.1ms	–	–
chi-monitoringdb-monitoring-0-0	clickhouse	–	2s	–	1/1	–	–	–	0.004	100%	–	–	14 unique errors
monitoring-prometheus-server	prometheus	–	633ms	1/1	1/1	–	–	–	–	–	failed conns	–	–
monitoring-frontend		<1%	10s	6/7	1/1	–	–	–	–	–	2ms	–	2 unique errors
monitoring-opentelemetry-collector-...	golang	–	4s	1/1	2/2	–	–	–	–	–	0.8ms	–	4 unique errors
monitoring-otelcol	golang	–	602ms	4/4	1/1	–	–	–	–	–	<0.1ms	–	1 unique error
monitoring-accountingservice		–	–	2/2	1/1	–	–	–	–	–	0.9ms	–	–
monitoring-adservice	java	–	27ms	2/2	1/1	–	–	–	–	–	<0.1ms	–	–
monitoring-cartservice	dotnet	–	249ms	3/3	1/1	–	–	–	–	–	0.8ms	–	–
monitoring-currencyservice		–	3s	1/1	1/1	–	–	–	–	–	0.7ms	–	–
monitoring-flagd	golang	–	10s	1/1	1/1	–	–	–	–	–	<0.1ms	–	–
monitoring-frauddetectionservice	java	–	–	3/3	1/1	–	–	–	–	–	0.1ms	–	–


Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)

Overview


Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)

coroot:~#

default

search for apps and nodes

last hour

Applications / monitoring-emailservice

● Errors: 1 error occurred

Condition: the number of messages with the ERROR and CRITICAL severity levels > 0

● Using container logs (configure)

Filter:

View:

Limit: 100

Coroot

Batteries-included, no-code observability

SigNoz

Traditional APM Features, OTel-Native

QRYN

"Querying" — LogQL, PromQL, and TempoQL for OpenTelemetry sources, with ClickHouse storage

ClickStack

Bundled OTel + ClickHouse + HyperDX (visualization and query UI)

Coroot

- eBPF-based Node-Agent
- OTLP ingestion via Collector Gateway
- Uses (mostly) standard OpenTelemetry Exporter schema + new schema for profiles
- Prometheus for time-series 🫣

QRYN

- Uses on its own collector exporter / collector distribution
- Exposes Tempo, Loki, OTLP, and Prometheus APIs
- Projects into compatible formats using Materialized Views

ClickStack

- OTel Collector as Agent
- ClickHouse for storage
- HyperDX for visualization
- Built-in support for ClickHouse, OpenTelemetry
- Flexible support for arbitrary schema and data exploration
- Session replay!
- MIT licensed

SigNoz

- Feature-complete “Traditional APM”
- OpenTelemetry native
- OTel Collector as Agent
- Host + Application Monitoring
- Many integrations (Queues!)
- Alerting
- Migration path for many tools

Schema Considerations

Schema Considerations

- ZSTD Compression
- Delta encoding
- Bloom filter indexes for maps (resources) and logs
- MergeTree, partitioned on time
- 7-day TTL

OpenTelemetry Collector Exporter for ClickHouse

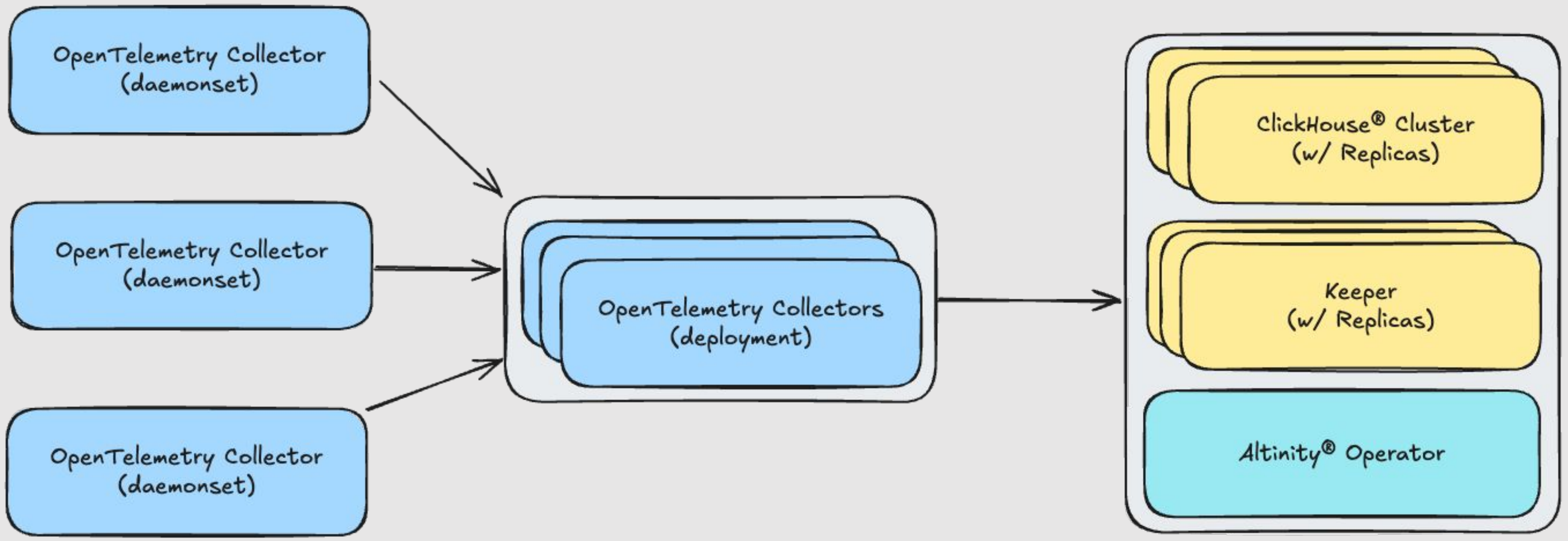
- Maps for metadata
- Efficient full-body text-search
- Materialized View for span durations

QRYN

- Fingerprints for unique time series
- Indexed labels (via Materialized Views)
- Allows for efficient updates (ReplacingMergeTree)
- Null Engine for raw ingest

Scaling for Production

Managing Multiple Collectors


OpenTelemetry Collector (daemonset)
OpenTelemetry Collector (daemonset)
OpenTelemetry Collector (daemonset)
OpenTelemetry Collectors (deployment)
ClickHouse® Cluster (w/ Replicas)
Keeper (w/ Replicas)
Altinity® Operator

The Altinity Operator

- PVC management
- Rolling upgrades
- Built-in monitoring

Alerting & Other Considerations

Conclusion

Why Unified Observability Storage?

- Simplified management
- Simplified scaling
- Cost management
- Standardization and normalization of metadata
- Post-hoc dependency mapping
- Cross-signal correlation around shared resource attributes

Thank You