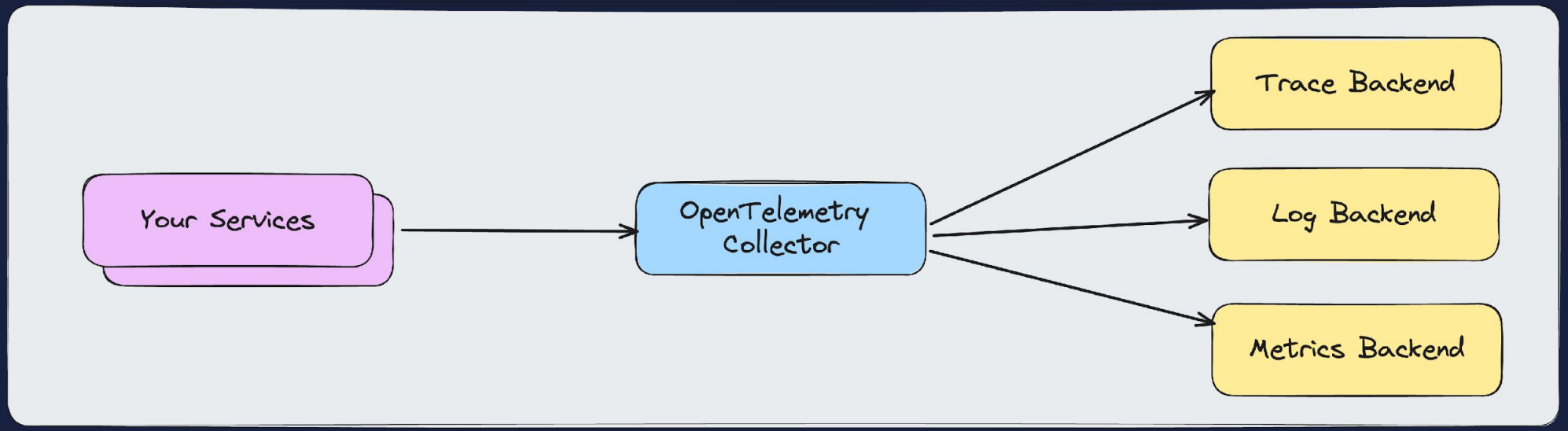
O11y in One:

ClickHouse® as a unified telemetry database

Josh Lee
@joshleecreates@hachyderm.io

How I usually start...



Josh Lee
@joshleecreates@hachyderm.io

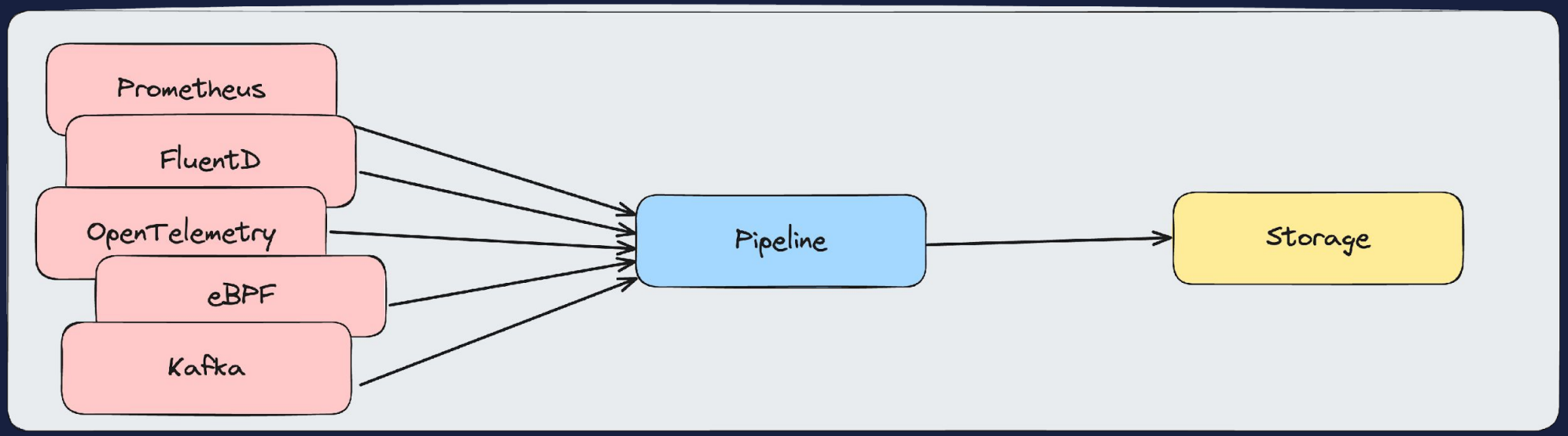
"The OpenTelemetry project does not include any kind of database or backend UI."

6

Minimum no. of o11y tools deployed by a typical organization

— Grafana State of Observability Report

What we really need...



Challenges with Disparate Telemetry Systems

Josh Lee
@joshleecreates@hachyderm.io

What are we storing?

Metrics, traces, logs, profiles, events

Resource metadata

Graphs & topologies

Snapshots & deltas

Configuration

Is There a Silver Bullet?

Full-text search

Efficient compression

Real-time analytics

Relational

Petabyte-scale

No. Obviously.

... but ClickHouse comes pretty close.

Introducing ClickHouse

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

ClickHouse for Observability

Josh Lee
@joshleecreates@hachyderm.io

Telemetry is WORM

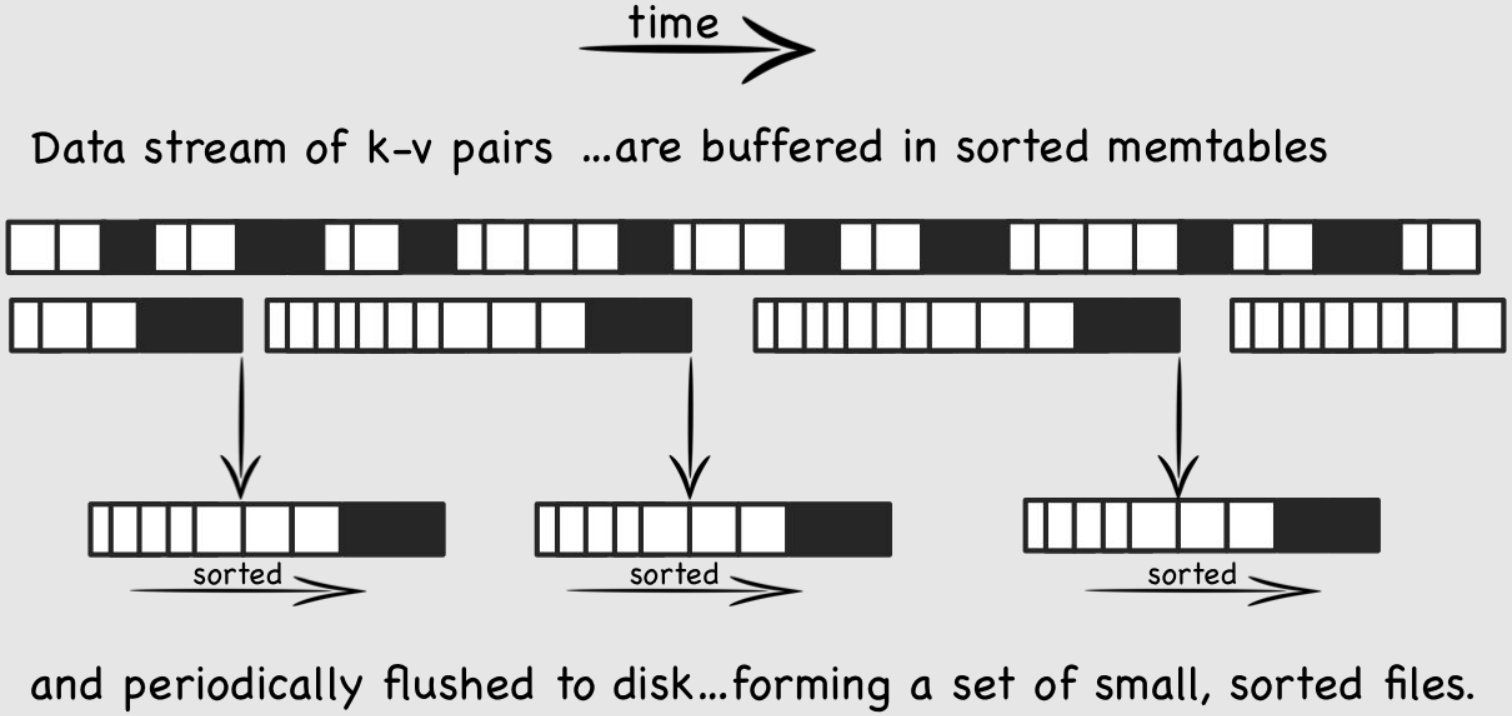
Write-Once, Read-Many

Josh Lee
@joshleecreates@hachyderm.io

B-Trees: Optimized for Reads

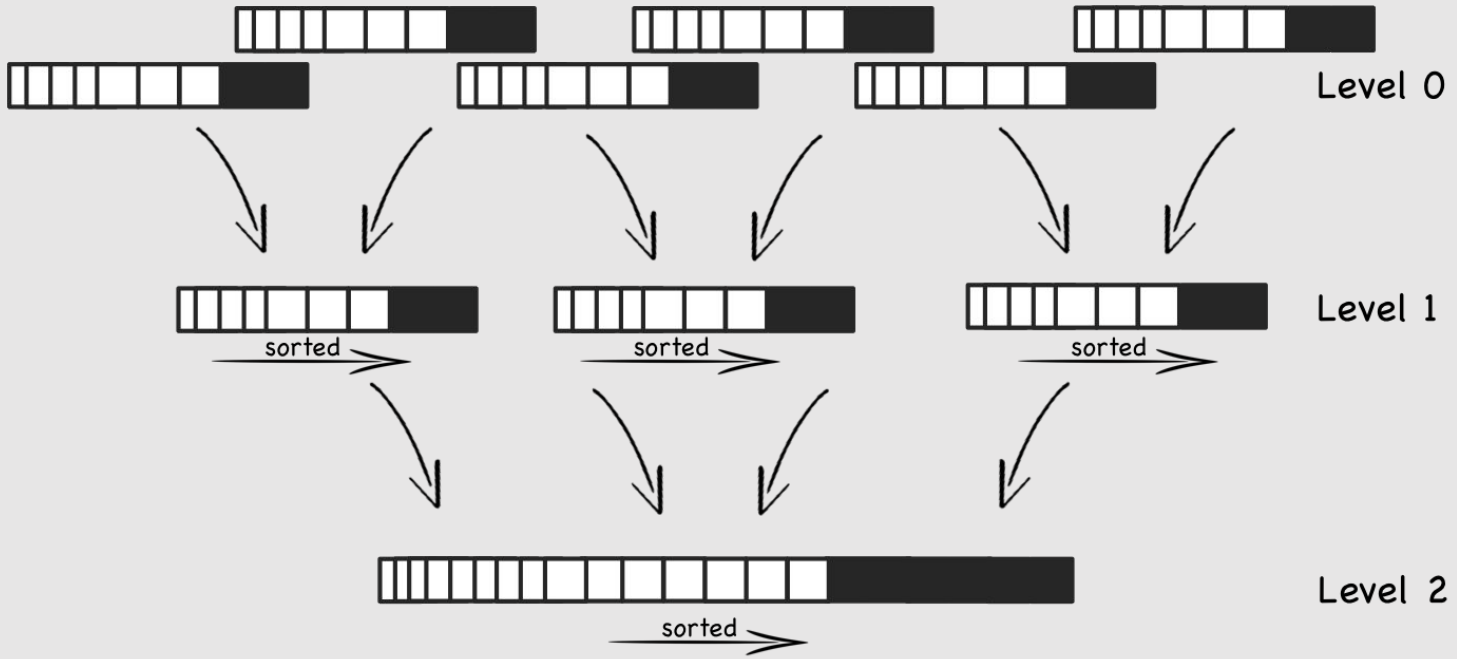
Josh Lee
@joshleecreates@hachyderm.io

Log-Structured Merge Trees: Optimized for ingestion



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

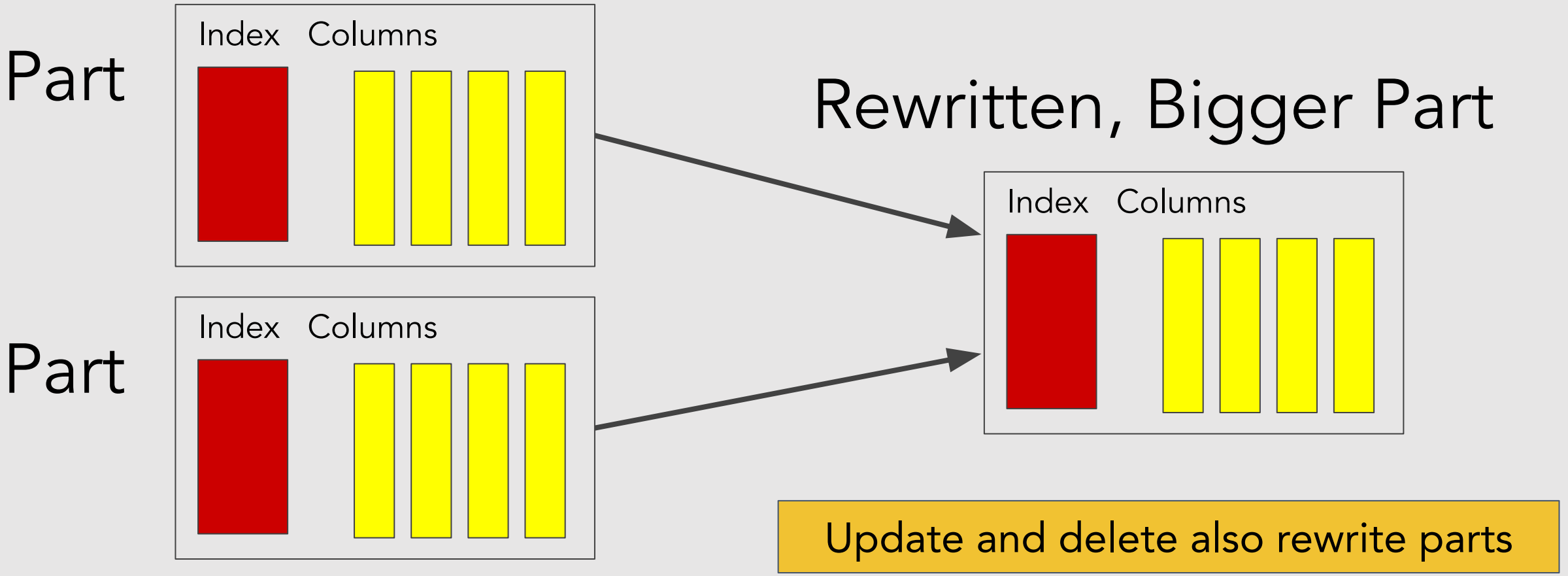
Log-Structured Merge Trees: Background compaction

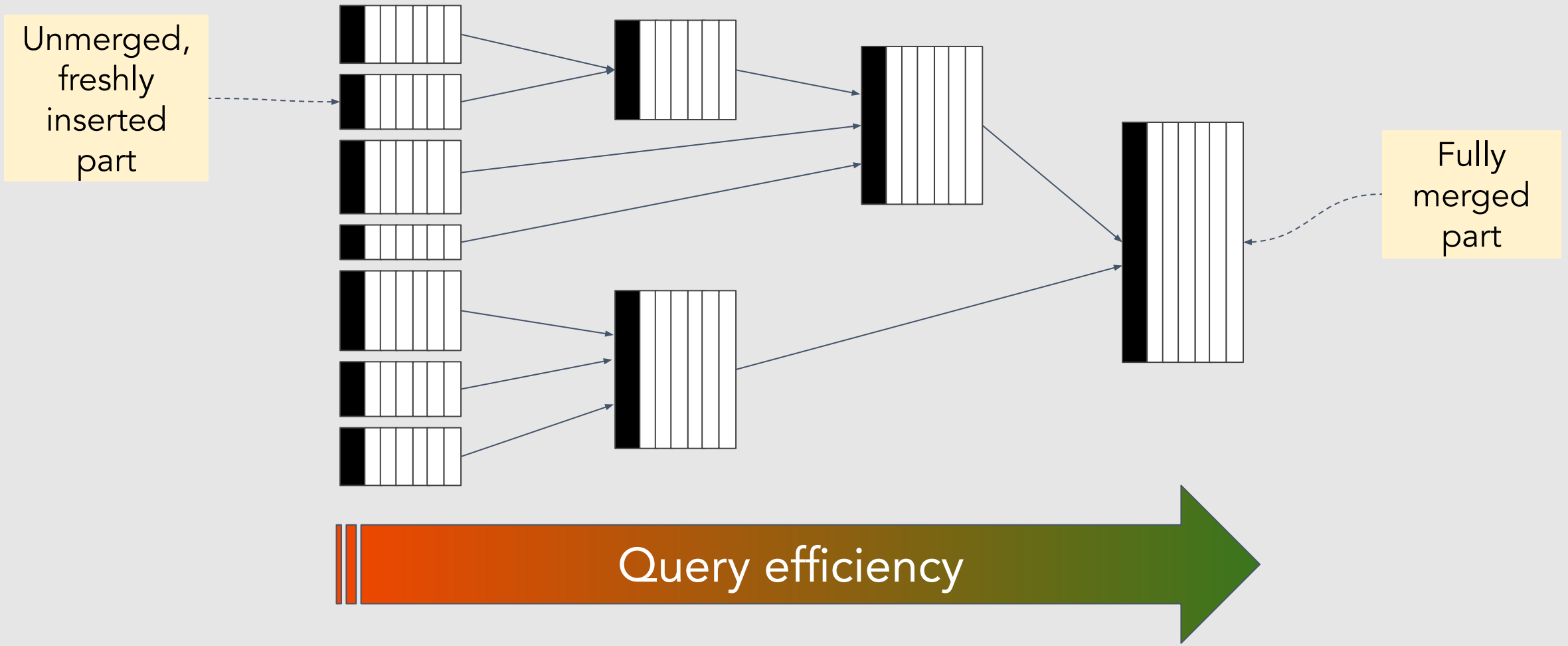


Compaction continues creating fewer, larger and larger files

Josh Lee
@joshleecreates@hachyderm.io

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


Part
Index
Columns
Part
Index
Columns
Rewritten, Bigger Part
Index
Columns
Update and delete also rewrite parts


Unmerged, freshly inserted part
Fully merged part
Query efficiency

How does this help?

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

Josh Lee
@joshleecreates@hachyderm.io

Data Transformation & Management

- Materialized Views
- TTL
- Tiered storage

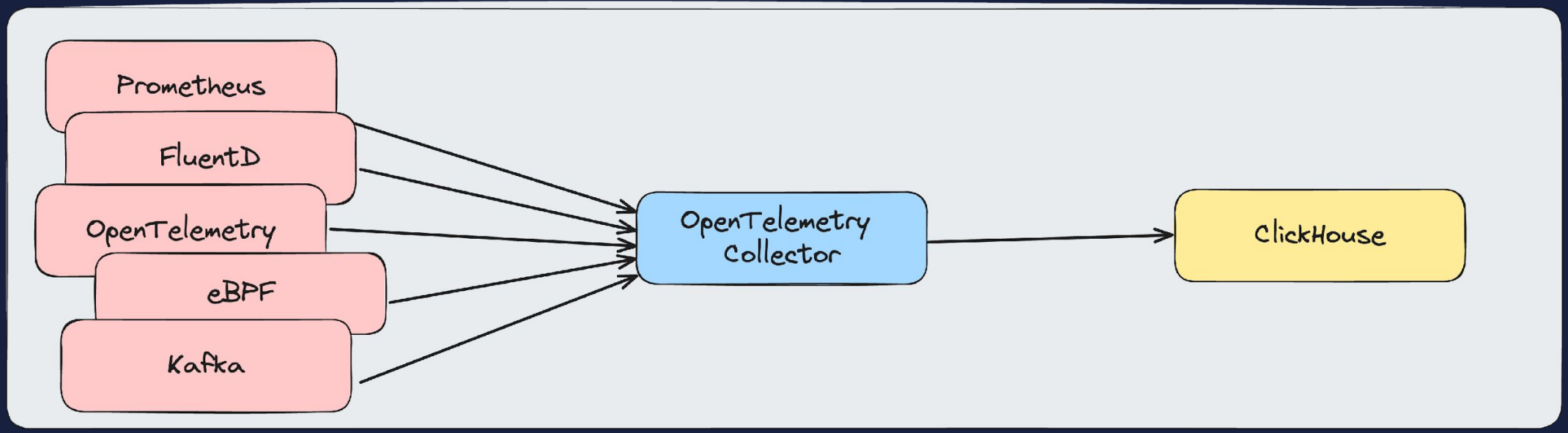
Integrations

- Grafana Datasource Plugin
- Jaeger w/ ClickHouse backend
- cLoki
- Kafka table engine

Josh Lee
@joshleecreates@hachyderm.io

ClickHouse for Observability

Integrations via OpenTelemetry



Josh Lee
@joshleecreates@hachyderm.io

ClickHouse for Observability

More Benefits

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

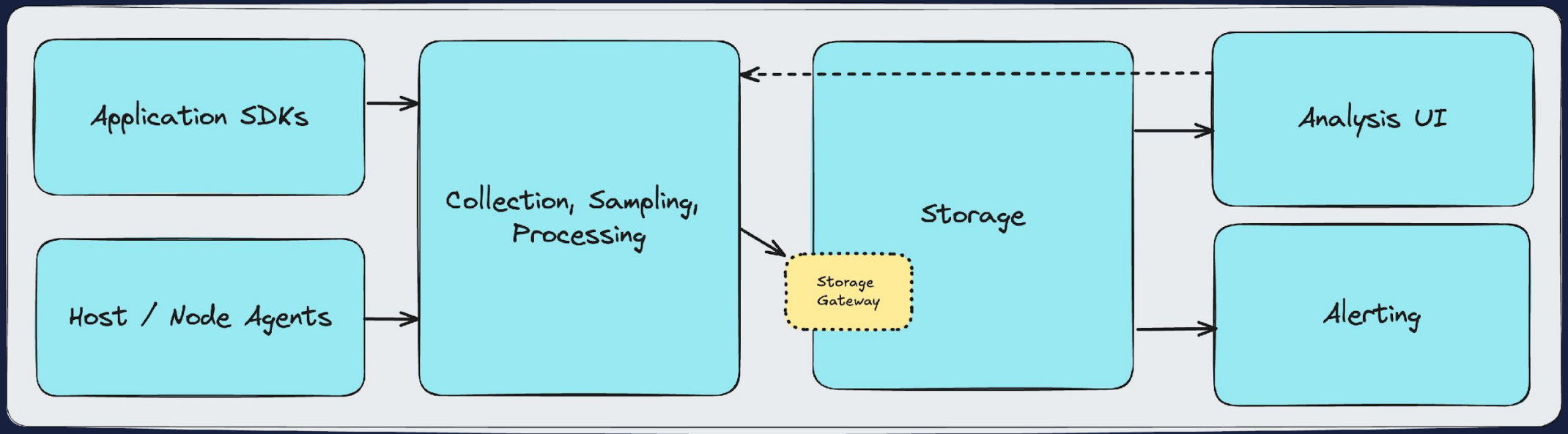
ClickHouse for Observability

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



Josh Lee
@joshleecreates@hachyderm.io

SigNoz

coroot

qryn

HyperDX

DIY

Josh Lee
@joshleecreates@hachyderm.io

SigNoz

coroot

qryn

HyperDX

DIY

Josh Lee
@joshleecreates@hachyderm.io

Coroot

Batteries-included, no-code observability

Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)

coroot :~# | default | search for apps and nodes | last hour

Overview

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

search | namespaces: monitoring | 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	–	–

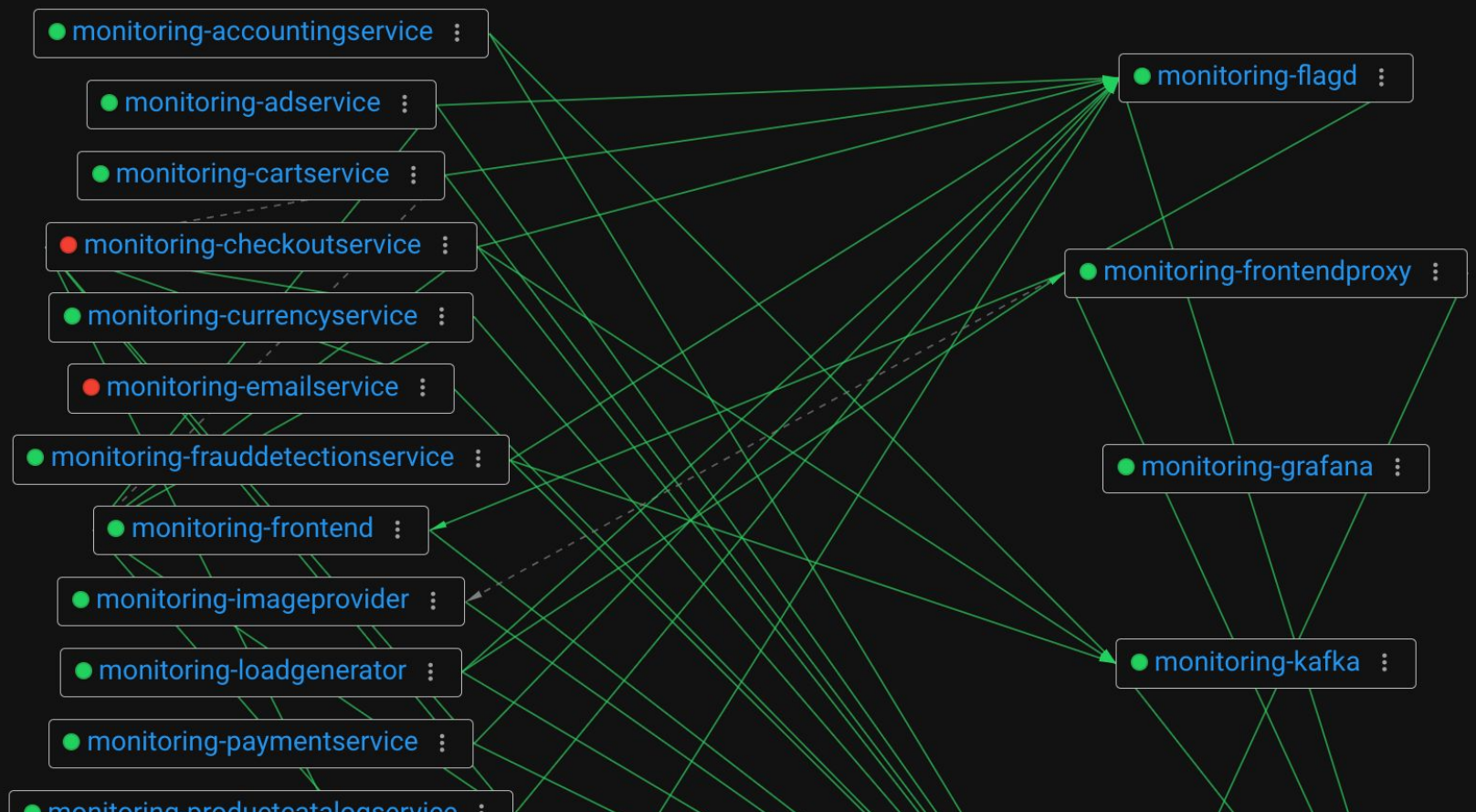
Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)

coroot :~# | default | search for apps and nodes | last hour

Overview

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

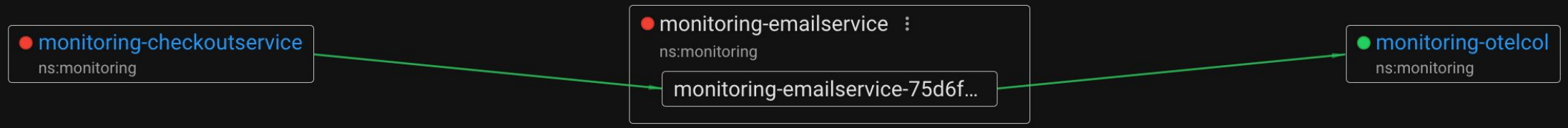
search | namespaces: monitoring | application | control-plane | monitoring | +



Coroot 1.5.0 is available 🎉 Changelog (how to upgrade)

coroot :~# | default | search for apps and nodes | last hour

Applications / monitoring-emailservice



SLO | INSTANCES | CPU | MEMORY | NET | LOGS | PROFILING | TRACING

Errors: 1 error occurred

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

Using container logs (configure)

Filter:

Filter messages | Query

View:

Messages | Patterns | Newest first | Oldest first | Limit: 100

qryn

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

Josh Lee
@joshleecreates@hachyderm.io

Data Ingestion

Ingesting data with **qryn** is easy and painless thanks to our **polyglot** design.

Use any **Agent** or **Library** compatible with *Opentelemetry, Loki, Datadog, Elastic, Prometheus, Tempo, Graphite, Pyroscope & more*

Logs | Metrics | Telemetry | Profiling | Custom

Get started using the Log Ingestion section

API Support

coroot

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

qryn

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

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

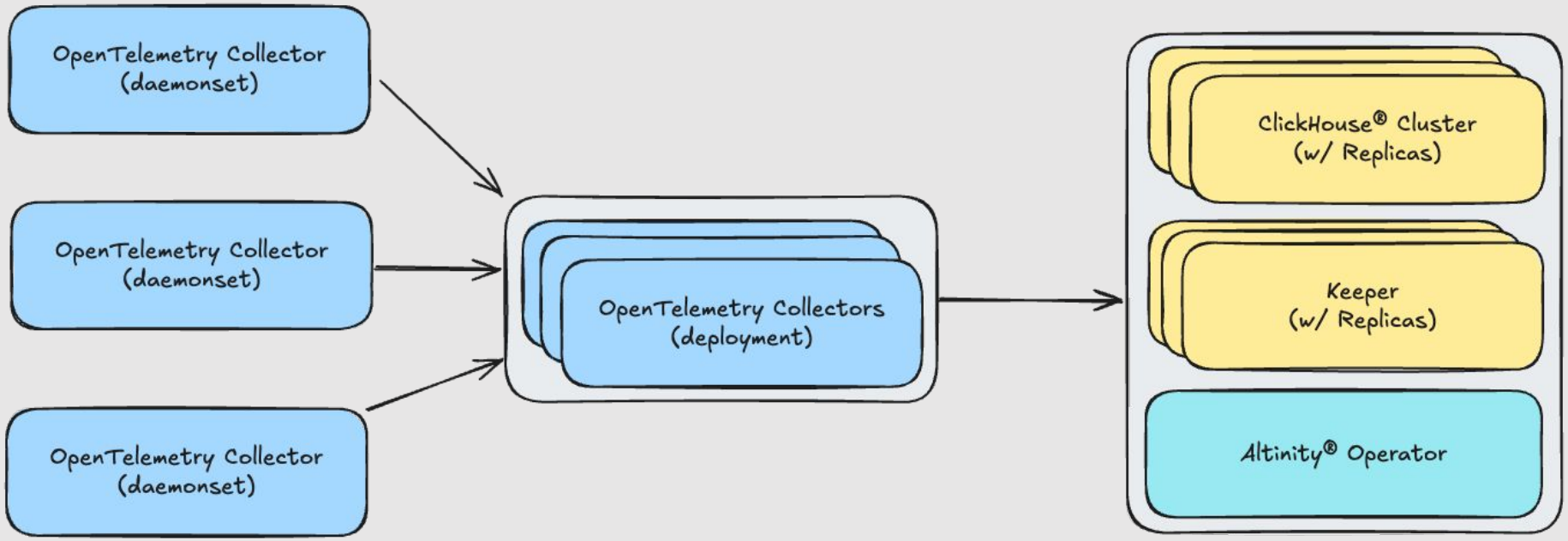
Josh Lee
@joshleecreates@hachyderm.io

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



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

Josh Lee
@joshleecreates@hachyderm.io

Thank You

Josh Lee
@joshleecreates@hachyderm.io