Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Transitioning to Open-Source Observability: A Cost-Benefit Analysis and Migration Framework

Satyanarayana Gudimetla
Independent Researcher, USA
Corresponding Author: Satyanarayana Gudimetla, **E-mail**: satyagudimetla557@gmail.com

| ABSTRACT

Commercial observability platforms cost enterprises $1.5–2.5 million annually at 1TB daily telemetry, motivating evaluation of open-source alternatives. This paper assesses migrating to an open-source stack—Prometheus (metrics), Loki (logs), and Jaeger (traces)—through a case study of an 800GB/day cloud-native deployment. The migration achieved 68% cost reduction ($1.26M first-year savings) with 14–24% query latency increase. Break-even analysis places the cost-advantage threshold at 50–100GB daily ingestion. We present a nine-month phased migration framework covering infrastructure sizing, query translation, and parallel validation. Critical success factors include structured logging practices, dedicated platform engineering resources, and comprehensive training (45 hours per engineer). The framework suits Kubernetes-native organizations with mature DevOps capabilities. Limitations: Findings derive from a single organization with favorable preconditions; cost data is self-reported; alternative technology stacks are not evaluated. Organizations should assess applicability to their specific contexts.

| KEYWORDS

Observability, Open-source, Prometheus, Loki, Distributed Tracing, Cloud-native, Kubernetes, Cost Optimization

| ARTICLE INFORMATION

ACCEPTED: 20 December 2025 **PUBLISHED:** 29 December 2025 **DOI:** 10.32996/jcsts.2025.7.12.55

1. Introduction

When a production incident strikes at 2 AM, engineers need answers immediately. Which service failed? What changed? Where did the request spend its time? Observability platforms - which combine logs, metrics, and traces - provide the answers to these questions. But the platforms themselves have become a significant cost center.

Commercial observability vendors price their products by data volume. Datadog charges per GB of data ingested and per host monitored. Splunk licenses its service based on daily indexing volume, while New Relic bills per GB of telemetry. These models create an unsettling situation: better instrumentation means higher costs. Engineering teams face pressure to reduce logging verbosity, shorten retention windows, or limit who can access the platforms: all of which degrade observability quality.

The scale of spending is substantial. Industry surveys indicate that organizations processing 500GB–1TB of logs daily pay $750K–$2M annually for observability tooling [1]. Recent CNCF surveys indicate that observability ranks among the top three operational expenditures for cloud-native organizations [2]. For organizations employing extensive instrumentation—gathering intricate traces, exhaustive analytics, and detailed logging—expenses may surpass the infrastructure costs of the monitored services. Open-source alternatives have matured significantly. Prometheus (contributed to the Cloud Native Computing Foundation in 2016) is now being run in production by thousands of organizations, including GitLab, Digital Ocean, and Uber [3]. Grafana Loki, a log aggregation app that eliminates the storage overhead associated with full-text indexing, was launched in 2018. Jaeger is an Uber-based open-source tracing system that supports distributed tracing at scale. OpenTelemetry reached version v1.0 in 2021 and has seen rapid adoption, with implementation rates increasing by 340% from 2021 to 2023 [4]. Collectively, these technologies offer a viable alternative to proprietary technologies.

Copyright: © 2025 the Author(s). This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) 4.0 license (https://creativecommons.org/licenses/by/4.0/). Published by Al-Kindi Centre for Research and Development, London, United Kingdom.

However, migration is a process that carries risk. Commercial platforms provide professional experiences involving amalgamation, automatic anomaly detection, and upended support from the vendors. Open-source solutions require assembly, configuration, and continued operational investment. Therefore, the potential cost savings should surpass the added complexity.

This article addresses four questions:

1. "What are the distinctions in actual expenditures?" - The analytical approach reveals the total cost of ownership (TCO) for data quantities between 100 GB/day and 5 TB/day, incorporating infrastructural and engineering expenses.
2. "What capabilities differ?" - Commercial platforms possess characteristics that are either absent in open-source tools or implemented differently; these differences have been systematically described.
3. "How can organizations do smooth migrations?" - A phased implementation framework is presented, which reduces operational risk across the transition.
4. "Who receives the most benefits?"- The study outlines organizational characteristics that are a prognosis for successful adoption.

This article focuses on a case study in A SaaS provider (hereafter called "TechCo") that has a throughput of 800 GB per day running through 450 microservices. TechCo completed a nine-month migration and reduced annual observability expenditure from $1.85 million to $590K. The article documents the approach, challenges, and results. This article possesses methodological limitations that impact generalizability. The analysis depends on a single firm possessing advantageous initial conditions, including established Kubernetes infrastructure, advanced DevOps methods, and pre-existing organized logs. Cost data is derived from TechCo's internal financial monitoring without an external audit. While commercial pricing research is available in the market, specific dollar amounts should also be treated as indicative rather than universal. As Organizations considering migration face different conditions, they must consider modifying the architecture outlined here as per the local context.

The article proceeds as follows. Section 2 reviews related work on observability architectures and cost modeling. Section 3 presents the economic analysis framework. Section 4 details the technical architecture of open-source observability stacks. The migration case study is discussed in Section 5. In Section 6, a decision framework is developed by synthesising the findings. Limitations are discussed in Section 7, while the summary is presented in the Conclusion section.

2. Background and Related Work

2.1 The Three Pillars of Observability

Observability is the ability to understand a system's internal state by looking at its outputs. It relies on three complementary data types, also referred to as its three pillars [5].

Logs document distinct occurrences: errors, state transitions, and audit entries. Every log entry records a specific instant together with contextual information. In distributed systems, logs from numerous services must be centrally aggregated; when a container terminates, its local logs vanish unless they have been previously transmitted elsewhere.

Metrics quantify numerical values across time: request frequencies, error rates, delay percentiles, and CPU utilization. Time-series databases conveniently store measurements, facilitating queries such as "What was the 99th percentile latency for the checkout service yesterday?" Prometheus popularized a pull-based collecting strategy in which a central server periodically scrapes metrics endpoints. Traces track specific requests across service boundaries. A trace consists of multiple spans: each denoting the tasks performed by a specific service. When a user's checkout request interacts with the API gateway, inventory service, payment processor, and notification system, a trace links these events, elucidating time distribution and pinpointing any errors that occurred. Google initiated distributed tracing with Dapper [6], and OpenTelemetry [5] has subsequently standardized it.

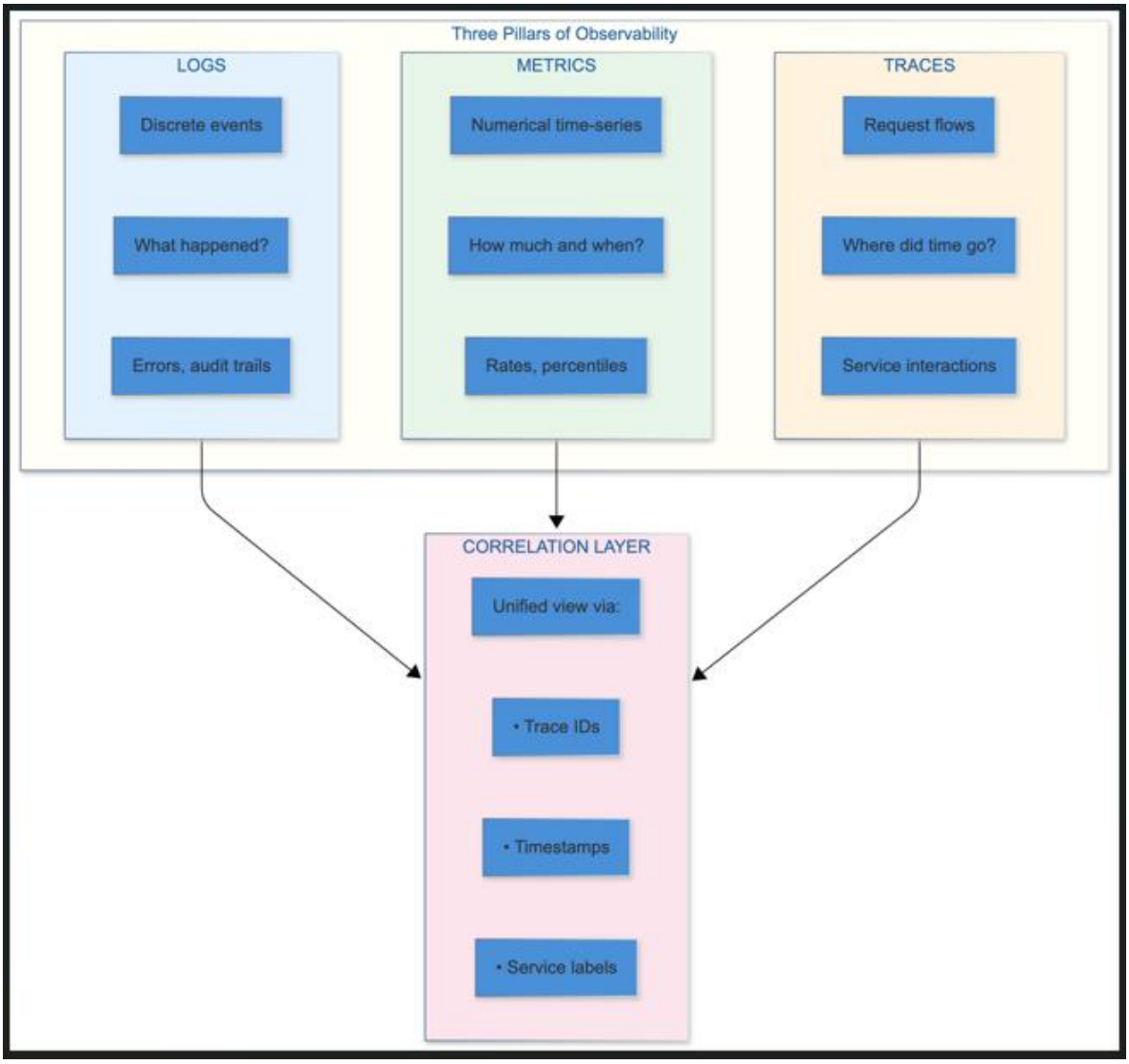

Figure 1: The three pillars of observability and their correlation

The three pillars of observability become powerful when correlated. A latency spike visible in metrics leads to traces showing slow database queries, which leads to logs revealing lock contention. For this investigative flow to work, logs, metrics, and traces must all have the same identifiers - the usual identifiers being trace IDs and service labels.

2.2 Commercial Platform Landscape

The commercial observability market consolidated significantly between 2018 and 2023. Splunk acquired SignalFx (metrics) and Omnition (tracing). Datadog acquired capabilities across all three pillars, New Relic pivoted from APM to full-stack observability, while Dynatrace expanded beyond its Java monitoring roots. These platforms compete based on the quality of their integration, specifically how seamlessly logs connect to traces and metrics. They also differentiate through machine learning features: automatic baseline detection, anomaly alerting, and root cause suggestions - which require substantial training data and engineering investment to replicate.

Pricing models differ, but possess the following common elements [1]:

- Ingestion-based: charges per GB of data received
- Host-based: charges per server, container, or function monitored
- User-based: charges per seat accessing the platform
- Retention-based: longer retention windows cost more

Most vendors combine these elements, creating complex pricing that scales with both infrastructure size and data volume.

2.3 Open-Source Observability Stack

The Cloud Native Computing Foundation (CNCF) oversees principal open-source observability initiatives.

Prometheus (which became a graduated CNCF project in 2018) offers solutions for metrics gathering and storage. The entire industry has been impacted by its labeled time series data technique. PromQL, its query language, has established itself as a de facto standard. Prometheus operates efficiently at a moderate scale; for large-scale deployments, services like Thanos [8] and Cortex offer horizontal scaling and long-term storage solutions.

Jaeger (graduated 2019) implements distributed tracing with support for multiple storage backends (Cassandra, Elasticsearch, Kafka). It originated at Uber, where it processes billions of spans daily [9].OpenTelemetry (incubating) provides vendor-neutral instrumentation libraries and a collector for routing telemetry data. It emerged from the merger of OpenTracing and OpenCensus, ending years of fragmentation in instrumentation standards [5].

Grafana (outside CNCF but widely used) provides visualization and dashboarding. Grafana Labs also develops Loki (log aggregation) and Tempo (trace storage), both designed to complement Prometheus.

2.4 Prior Migration Studies

Documented migration experiences are scarce. The majority of case studies are found in vendor blogs or conference presentations, rather than in peer-reviewed publications. Current scholarly research emphasizes observability designs [11] and performance attributes [12] instead of migration economics. A 2023 comparative analysis investigated open-source tracing technologies but omitted migration methods and associated expenses [13].

The FinOps Foundation published a report in 2023 in which it crafted observability as a target for cost optimization, but the advice on open-source alternatives was sparse [14].

Research on the adoption of Kubernetes does mention wider infrastructure transitions, but scarcely mentions changes in observability tools, which is also a potential aspect [15]. This disparity motivated the investigation carried out by this article's team. Organizations considering migration lack empirical data on costs, timelines, and success factors. This article provides the data through a detailed case study analysis.

3. Economic Analysis Framework

3.1 Cost Model Structure

Total cost of ownership for observability platforms comprises several components:

Direct costs:

- Licensing fees (commercial) or infrastructure costs (open-source)
- Storage for retained telemetry data
- Network transfer for data ingestion

Operational expenses:

- Engineering time for platform management
- Training and skill development
- Incident response related to observability infrastructure

Opportunity costs:

- Features unavailable in the chosen platform
- Productivity impact from capability gaps

This article presents a model that quantifies direct and operational expenses. Opportunity costs remain qualitative due to their context-dependent nature.

3.2 Commercial Platform Cost Estimation

Pricing data was collected through three channels: published list prices from vendor websites (Datadog, Splunk, New Relic, Dynatrace), informal discussions with procurement teams at six organizations, and a review of anonymized contract data shared by two SaaS companies. The surveyed firms varied in size from 150 to 2,500 employees, all within the technology sector, processing

between 100GB and 1.5TB daily. Pricing negotiations fluctuate considerably depending on contract duration, data volume obligations, and vendor rivalry. The ranges in Table 1 (placed after Figure 2) illustrate this variability: lower bounds denote aggressive multi-year contracts with volume discounts, while upper bounds indicate month-to-month or short-term agreements. Organizations should anticipate negotiating towards the bottom spectrum of these ranges. Commercial expenses primarily increase with the volume of consumption.

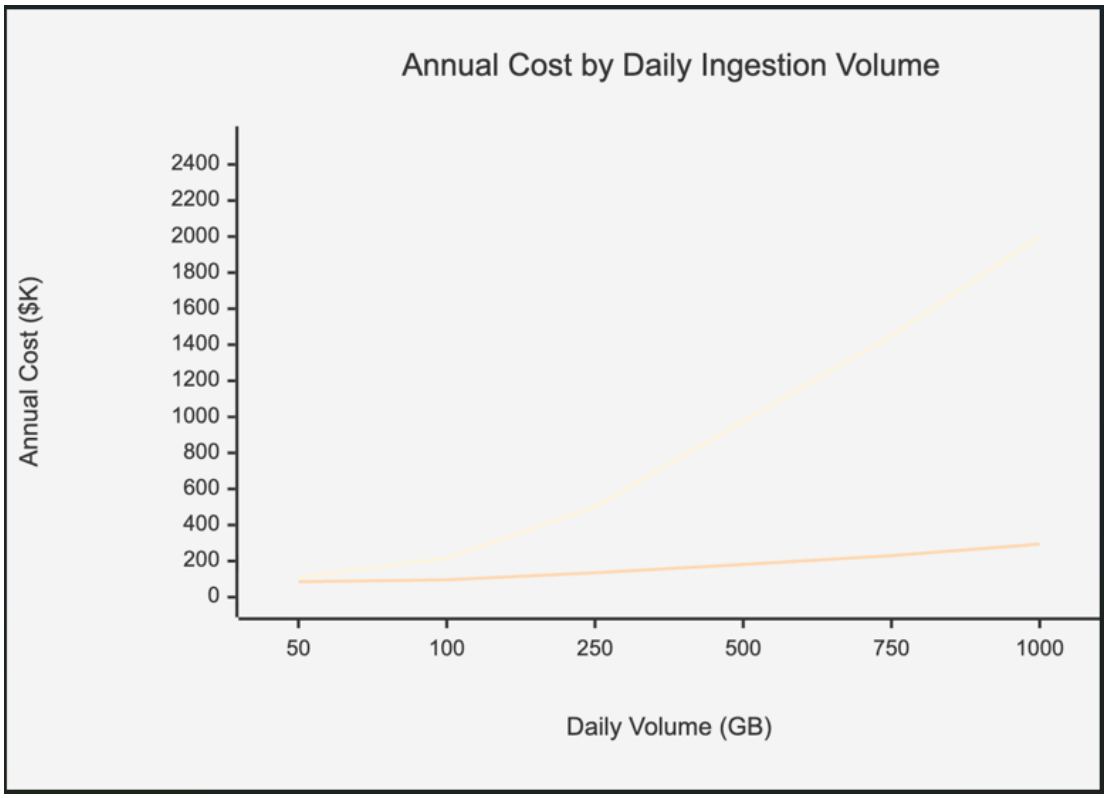

Figure 2: Commercial observability costs scale linearly with volume

Table 1 below summarizes these findings:

Daily Volume	Annual Cost Range	Median	Median Uncertainty	Per-GB Cost	Sample Size
100 GB	$180K - $250K	$215K	± $35K	$5.90	4
250 GB	$400K - $600K	$500K	± $85K	$5.48	3
500 GB	$750K - $1.2M	$975K	± $190K	$5.34	6
1 TB	$1.5M - $2.5M	$2.0M	± $380K	$5.48	2

Table 1: Commercial observability platform costs by ingestion volume (2023-2024 data)

Per-GB expenses exhibit relative constancy over varying scales, signifying linear pricing. While certain bulk reductions are available, they do not fundamentally alter the economics.

3.3 Open-Source Cost Estimation

Open-source costs comprise infrastructure and engineering:

Infrastructure costs depend on:

- Compute for ingestion, processing, and queries
- Storage for time-series data, logs, and traces
- Network for internal communication and user access

Engineering costs depend on:

- Initial migration effort (one-time)
- Ongoing operations and maintenance
- Training and knowledge development

Infrastructure modeling was conducted utilizing AWS pricing (on-demand rates, us-east-1):

Daily Volume	Compute	Storage	Network	Monthly Total
100 GB	$800.00	$400.00	$150.00	$1,350.00
500 GB	$2,800.00	$1,800.00	$450.00	$5,050.00
1 TB	$5,200.00	$3,500.00	$800.00	$9,500.00

Table 2: Open-source infrastructure costs (monthly, 30-day retention)

Engineering overhead varies by an organization's maturity. Interviews with platform teams at six organizations yielded the following estimates:

- Mature platform teams (existing Kubernetes expertise): 0.25–0.5 FTE ongoing
- Developing teams (some cloud-native experience): 0.5-1.0 FTE ongoing
- New teams (limited prior experience): 1.0-1.5 FTE ongoing

With a fully-loaded cost of approximately $150K per engineer, this equates to an annual range of approximately $37.5K to $225K.

3.4 Break-Even Analysis

The TCO is calculated by combining the infrastructure and engineering costs.

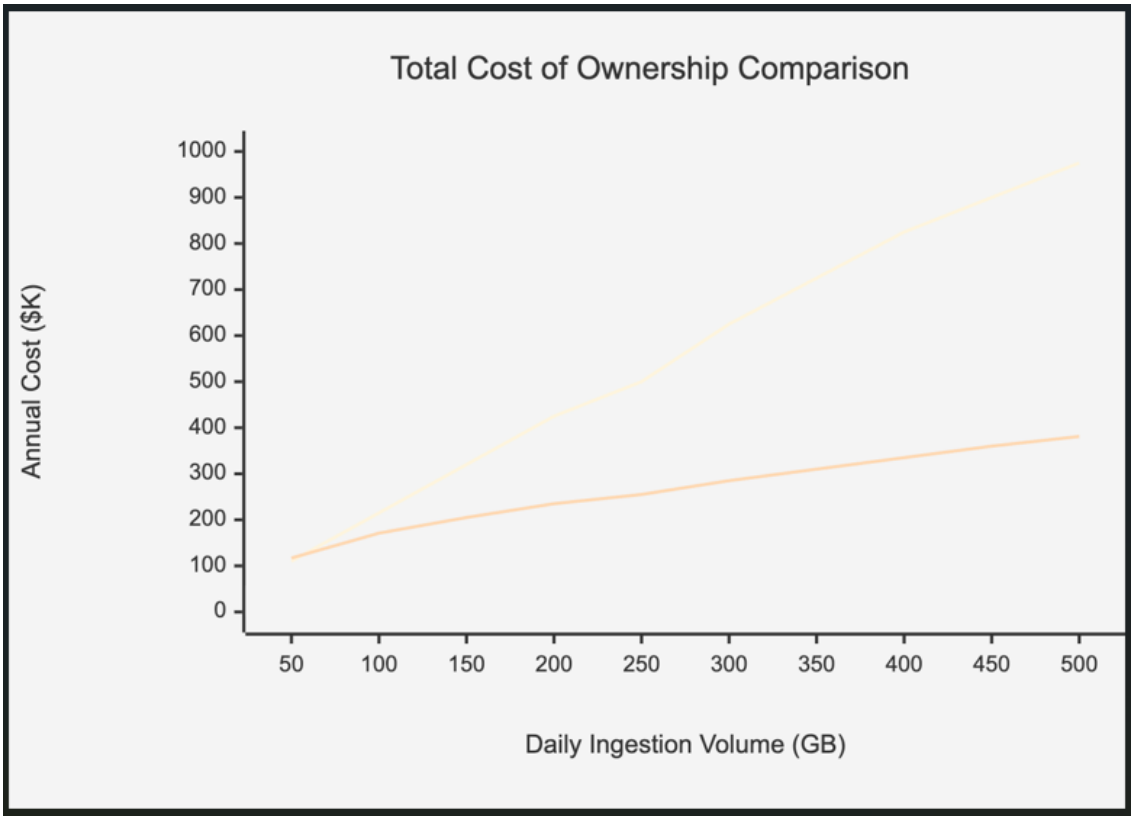

Figure 3: TCO comparison showing break-even at 50-100GB daily

Key finding: The break-even threshold lies between 50GB and 100GB of daily data ingestion. For data under 50 GB, the ease of commercial platforms generally surpasses the associated cost premium. For data exceeding 100 GB, open-source platforms provide evident cost benefits irrespective of engineering cost assumptions.

Table 3 shows projected savings at scale, as follows:

Daily Volume	Commercial Cost	Commercial Uncertainty	Open-Source TCO	Open-Source Uncertainty	Annual Savings	Savings Percentage
100 GB	$215K	± $35K	$96K	± $25K	$119K	55%
500 GB	$975K	± $190K	$181K	± $40K	$794K	81%
1 TB	$2.0M	± $380K	$294K	± $65K	$1.7M	85%

Table 3: Projected annual savings from open-source migration

The open-source total cost of ownership encompasses infrastructure (annualized in Table 2) together with around $75,000 in engineering overhead (0.5 full-time equivalent). Actual savings are contingent upon negotiated commercial pricing and engineering efficiency.

4. Technical Architecture

4.1 Reference Architecture

An operational open-source observability stack requires the integration of several key components, as shown in Figure 4:

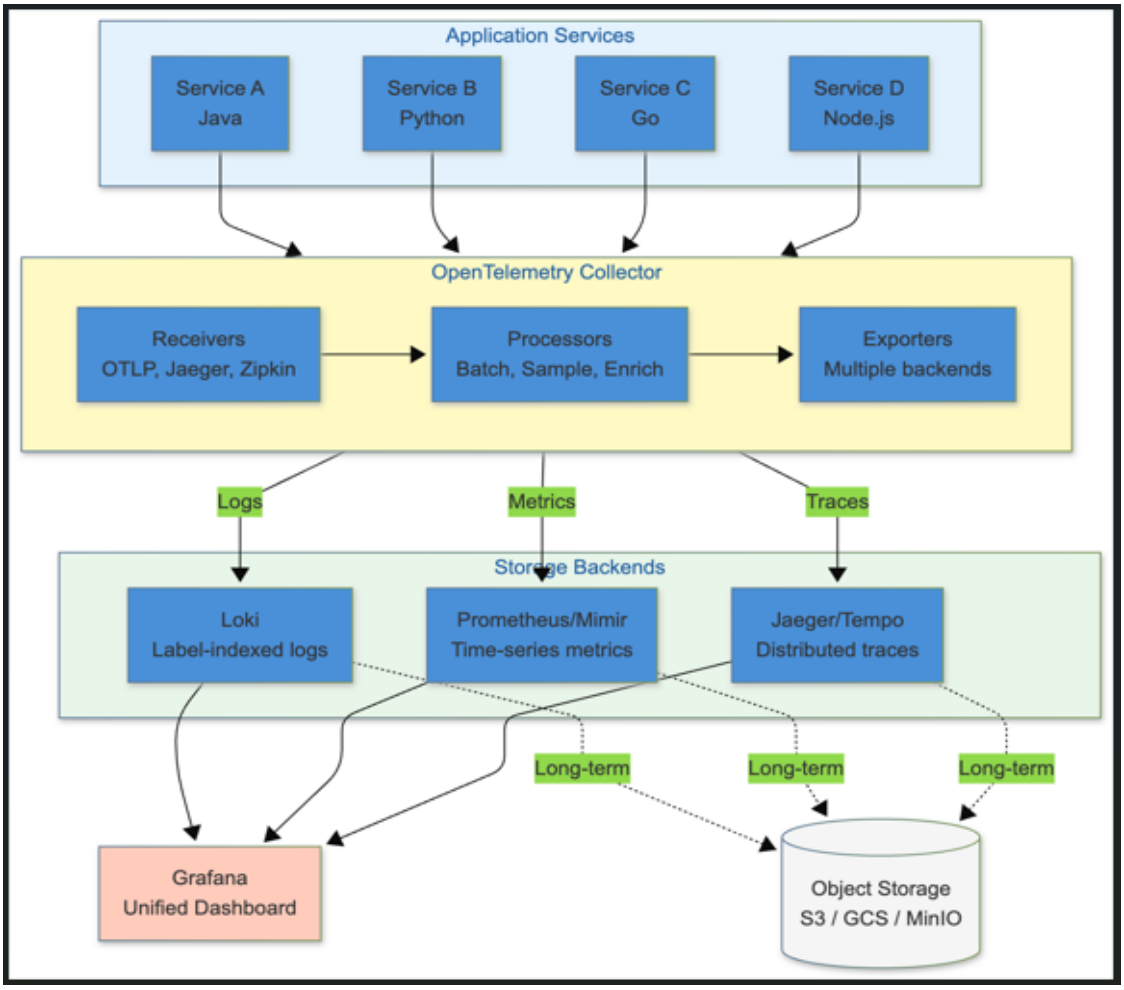

Figure 4: Reference architecture for open-source observability stack

- The **OpenTelemetry Collector** serves as the primary routing layer. It acquires telemetry from apps using OTLP (OpenTelemetry Protocol), analyzes it (sampling, enrichment, batching), and outputs it to the relevant backends. This decoupling facilitates the alteration of backends without necessitating the re-instrumentation of apps.
- Metrics are stored in **Prometheus/Mimir**. Prometheus manages data gathering and short-term storage, while tools like Mimir, Thanos, and Cortex provide horizontal scaling and long-term retention for object storage.
- **Loki** uses label-based indexing for log storage, and unlike Elasticsearch, it does not index log content; rather exclusively indexes metadata labels. This substantially reduces storage costs while simultaneously limiting query flexibility.
- **Jaeger** archives dispersed traces and accommodates various back-ends. Kafka-backed ingestion combined with Elasticsearch or Cassandra storage is effective at production scale.

.

Grafana provides integrated visualization for all three data sources, including dashboards, alerting, and exploration interfaces.

4.2 Loki: Label-Based Log Aggregation

Loki's architecture fundamentally contrasts with conventional log platforms, as shown in Figure 5:

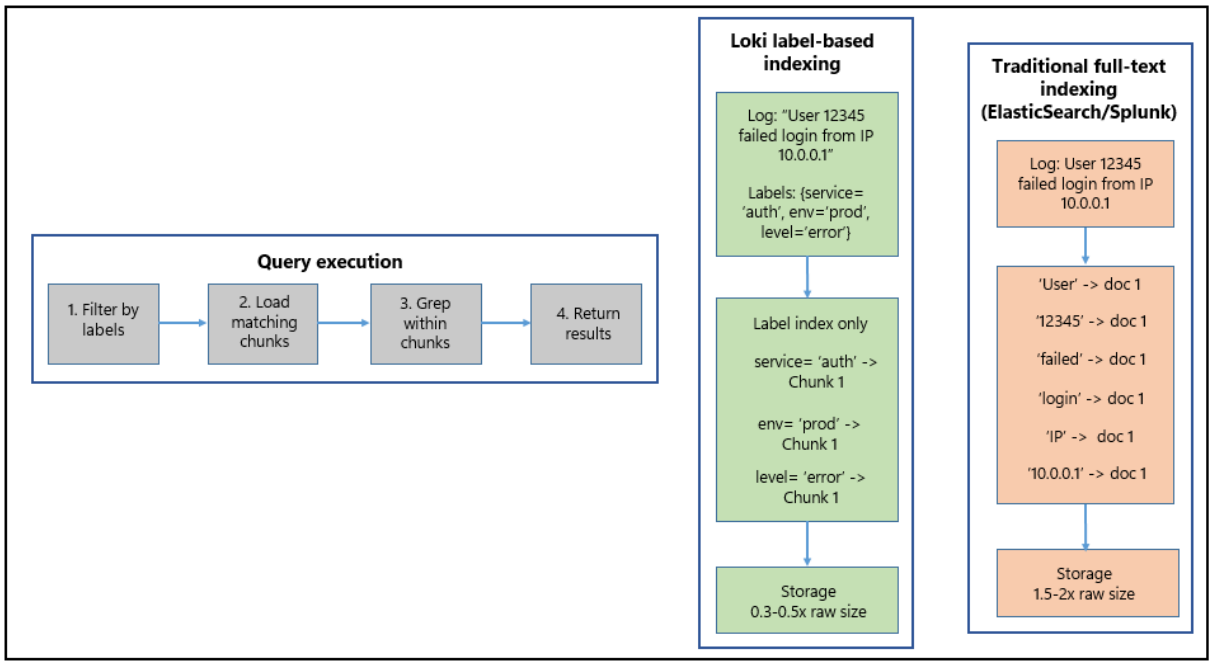

Figure 5: Loki's label-based indexing vs traditional full-text indexing

Advantages:

- 50-80% storage reduction compared to full-text indexing
- Simpler operations (no index management, compaction tuning)
- Inherent compatibility with Kubernetes labels (pod, namespace, container)

Limitations:

- Queries must specify labels; can't grep across entire corpus
- Regex searches on content scan all matching chunks (slower)
- No automatic field extraction or parsing

Loki is optimized for structured logs. Logs structured in JSON with uniform field names facilitate quick querying. Unstructured text logs lead to the forfeiture of major advantages.

4.3 Prometheus Scaling Patterns

Single instances of Prometheus can handle millions of time series with weeks of retention. Beyond this point, horizontal scaling is imperative, as shown in Figure 6.

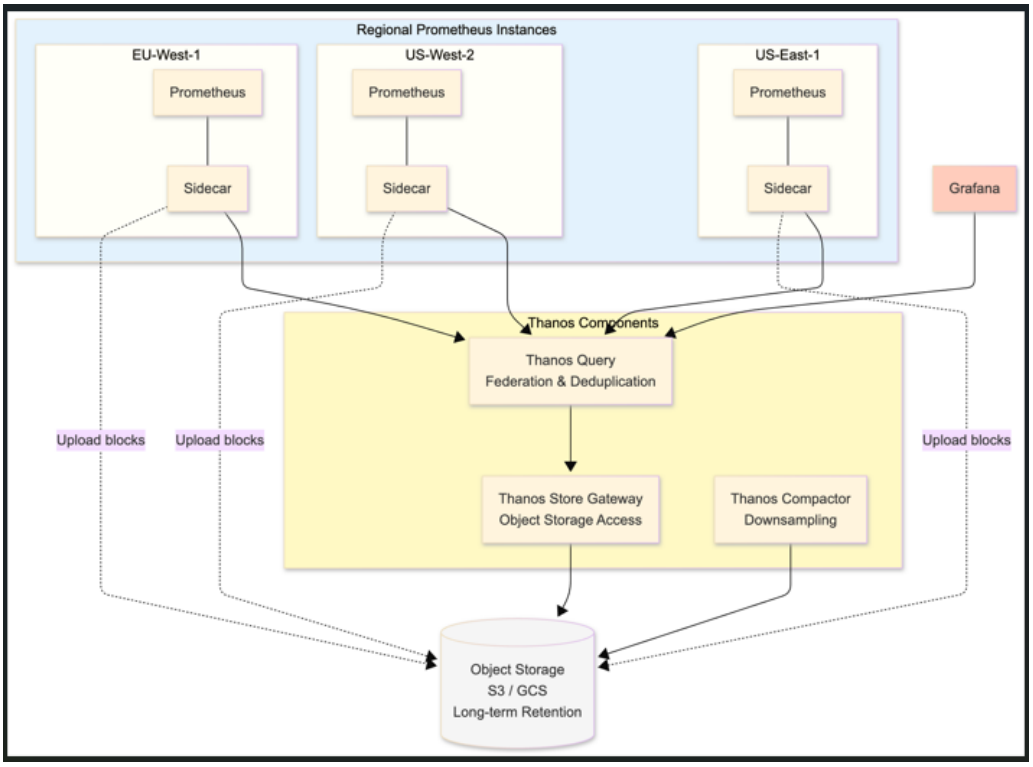

Figure 6: Prometheus horizontal scaling with Thanos

Mimir (Grafana's fork of Cortex) provides an alternative - featuring inherent multi-tenancy and streamlined operations. Both achieve:

- Billions of active time series
- Sub-second queries for 30-day windows
- 40-60% storage efficiency gains through compression.

4.4 Feature Comparison

Table 4 compares commercial and open-source observability solution capabilities:

Capability	Commercial Platforms	Open-Source Stack
Log full-text search	Native, fast	Label-filter required first
Automatic anomaly detection	Built-in ML	Requires custom development
Distributed tracing	Integrated	Jaeger/Tempo + integration work
Single-pane dashboards	Native	Grafana (excellent)
Alert management	Built-in	Alertmanager + configuration
User management	Native RBAC	Grafana + identity provider
Mobile access	Native apps	Grafana mobile (limited)
Compliance certifications	SOC2, HIPAA, etc.	Self-managed compliance

Table 4: Feature comparison between commercial and open-source observability

The differences matter the most for organizations requiring:

- Full-text search across months of unstructured logs
- ML-based anomaly detection without engineering investment
- Vendor-certified compliance (SOC 2, HIPAA, FedRAMP)

5. Migration Case Study

5.1 Organization Profile

TechCo (anonymized) manages a B2B SaaS platform with 2.3 million active users with heir infrastructure operating on AWS EKS, comprising 450 microservices developed using Java, Python, Go, and Node.js.

Pre-migration observability:

- Logs: Commercial platform A, 650 GB per day, approximately $1.15 million per year
- Metrics: Commercial platform B, 120 GB/day (8 million time series), approximately $380,000 annually
- APM/Traces: Commercial platform C, 30 GB/day (45 million spans), approximately $320,000 annually
- Total: around $1.85 million per year

Team composition:

- 85 software engineers
- 12 site reliability engineers
- No dedicated observability team

Motivation: Annual license renewal included an 18% price increase. The leadership instructed the platform team to assess options.

5.2 Migration Approach

The migration occurred in four phases over a duration of nine months:

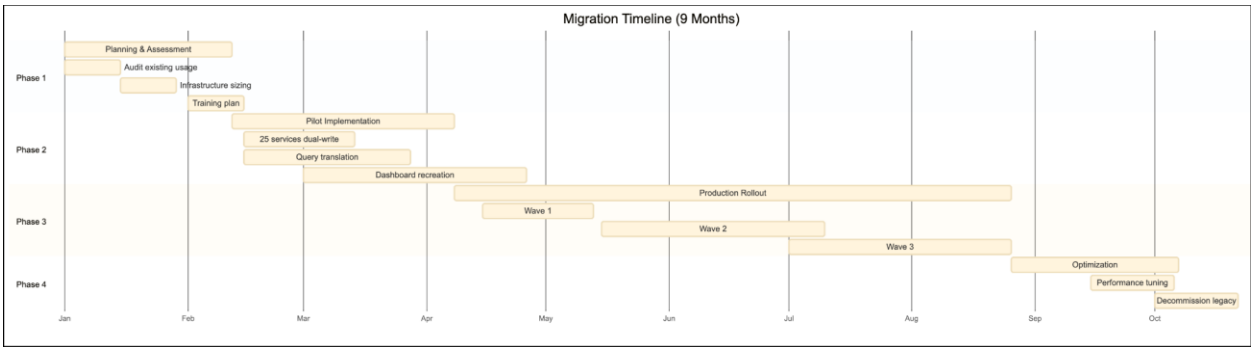

Figure 7: Nine-month migration timeline

Phase 1: Planning (Weeks 1–6)

The team audited existing usage patterns:

- Which dashboards were accessed on a weekly basis, and which ones were accessed on an annual basis?
- Which alarms successfully notified engineers, and which ones were disregarded?
- What queries appeared in incident postmortems?

This indicated considerable inefficiency: 40% of dashboards had not been accessed in six months. A multitude of alarms had been muted and disregarded. The audit lowered the migration scope and emphasized essential observability requirements. Infrastructure sizing utilized current ingestion rates with an additional 50% margin:

- Loki: 8 ingesters, 4 queriers, S3 backend
- Mimir: 12 ingesters, 6 queriers, S3 backend
- Jaeger: Kafka + Elasticsearch cluster
- Grafana: 3-node HA deployment

Phase 2: Pilot (Weeks 7–14)

Twenty-five services, selected to exemplify various technology stacks and traffic patterns, commenced dual-writing telemetry to both legacy and contemporary systems. Dual-write served multiple purposes:

- Validation: Did the new system capture all events?
- Comparison: Did equivalent queries return equivalent results?
- Training: Engineers learned new query languages without production pressure

The Pilot phase demonstrated that query translation required more effort than anticipated: LogQL (Loki) and PromQL (Prometheus) display significant disparities from the query languages of commercial platforms. The team allocated 120 technical hours to translate 280 stored queries, averaging around 25 minutes each query. Complex queries with many aggregations needed over 2 hours apiece, and the recreation of the dashboard required 8 weeks, as against the anticipated 4 weeks. The production dashboards (over 150) could not be exported and imported directly; each necessitated painstaking reconstruction in Grafana with modified queries.

Phase 3: Production Rollout (Weeks 15–34)

Migration proceeded in waves:

- **Wave 1 (Weeks 15–18):** 50 internal services that were not critical. Operations processes (runbooks, incident response plans, escalation paths) were put in place before going after customer-facing services.
- **Wave 2 (Weeks 19–26):** 200 moderate-criticality services. These had to be integrated within systems, including SSO for authentication, PagerDuty for alerts, and permissions for dashboard sharing.
- **Wave 3** involved **Weeks 27-34** to migrate the remaining services and historical data. The team migrated 18 months of archived data, including logs, metrics, and traces, to provide historical comparisons and trend analysis.

Throughout the production rollout, both legacy and contemporary systems operated concurrently. This increased short-term expenses but also established a safety net: when engineers encountered difficulties in the new system, they reverted to the old one during the learning process.

Phase 4: Optimization (Weeks 35–40)

Following migration completion, the focus switched to efficiency:

- Right-sizing container resources based on actual usage
- Tuning retention policies (7 days for verbose debug logs, 90 days for errors)
- Consolidating duplicate dashboards
- Reducing alert noise through threshold refinement

The decommissioning of the commercial platform commenced after four weeks of exclusive operation of the new system, thereby demonstrating its stability.

Why did TechCo succeed?

Not all migrations succeed. The platform lead at TechCo talked about this case study in a regional meetup, which is how it came to light. Success here reflects several favorable preconditions:

- Kubernetes infrastructure already in place (3+ years operational experience)
- Structured JSON logging is already standardized across services
- Platform team with two dedicated SREs and executive support
- No compliance requirements mandating vendor certifications
- Financial pressure from the 18% license increase is creating organizational urgency

Organizations without these conditions encounter more challenging trajectories. An infrastructure consultancy, with which the author conversed, reported collaborating with a customer who discontinued migration after three months upon realizing that log formats were inconsistent across over 200 services - the standardization effort would have required more time than the transfer itself.

5.3 Challenges and Solutions

5.3.1 Challenge 1: Query language learning curve

PromQL and LogQL possess distinct conceptual frameworks compared to commercial systems. Engineers encountered difficulties initially.

Solution: A translation reference cheat-sheet, correlating prevalent commercial inquiries to open-source alternatives, was created. Platform team members assisted engineers in interpreting requests during designated weekly office hours.

5.3.2 Challenge 2: Missing full-text search

Numerous workflows relied on querying random strings throughout all logs, especially for security investigations.

Solution: A hybrid strategy was implemented. High-value logs (security events, audit trails) continued to be transmitted to a designated commercial platform instance with limited licensing. Operational logs were transferred to Loki. Total commercial expenditure decreased from approximately $1.85 million to approximately $180,000.

5.3.3 Challenge 3: Alert tuning

Commercial systems' machine learning-based alerting has assimilated normative patterns over the years. Open-source alerting commenced from scratch, resulting in false positives.

Solution: A six-week calibration phase, with increased alarm noise, was agreed to. Engineers recorded threshold modifications, cultivating institutional knowledge regarding normal variance.

5.3.4 Challenge 4: Inaccurate estimated training time

Initial estimate: 24 hours per engineer. Actual observation: 45 hours per engineer.

Solution: The training schedule was prolonged. Recorded sessions were produced for asynchronous education. Junior engineers were assigned to collaborate with platform team members during on-call shifts.

5.4 Results

The following table illustrates the cost comparison:

Category	Before	After	Change
Platform licensing	$1,850K	$180K*	-90%
Infrastructure	–	$280K	–
Engineering (ongoing)	–	$75K	–
Training (one-time)	–	$45K	–
Migration (one-time)	–	$120K	–
Total Year 1	$1,850K	$700K	-62%
Total Year 2+	$2,035K†	$535K	-74%
Notes:			
***Retained commercial instance for security/audit logs**			
†Assumes 10% annual price increase			

Table 5: Cost comparison pre- and post-migration

First-year savings: approximately $1.15 million (62%); **Ongoing annual savings:** approximately $1.5 million (74%); **Payback duration:** 7.3 months

The following table illustrates the performance comparison:

Metric	Before	After	Change
Log query p50	280 ms	320 ms	14%
Log query p95	650 ms	780 ms	20%
Metric query p50	180 ms	210 ms	17%
Metric query p95	420 ms	480 ms	14%
Trace query p50	450 ms	520 ms	16%
System availability	99.95%	99.91%	-0.04%

Table 6: Query performance before and after migration

Latency increased by 14–20% across various query types. However, no significant resultant effect on incident response was reported by engineers- as differences in response times such as 280 ms and 320 ms. are practically imperceptible while troubleshooting. Availability of the system dropped from 99.95% to 99.91% with an extra 21 minutes of downtime over the course of a year. This showed a learning curve that availability enhanced as the team acquired expertise. Operational metrics are depicted below:

Metric	Before	After
Mean time to detection	2.3 min	2.7 min
Mean time to resolution	18 min	21 min
False alert rate	3.2%	3.8%
Engineers with platform access	25	97

Table 7: Operational metrics comparison

Though MTTD and MTTR had risen a bit during the stabilization period, the team expected things to get back to normal after 6-12 months once people got used to the new system. The number of users with platform access was significantly expanded from 25 licensed users to all 97 engineers, enabling them to debug issues directly. This result provided a takeaway: engineers could directly investigate issues instead of awaiting assistance from individuals with platform access.

5.5 External Validation: Survey Data and Contrasting Cases

5.5.1 Data resulting from the survey

TechCo's financial constraints and relocation goals align with prevailing industry trends. The CNCF Annual Survey 2023 indicates that 73% of cloud-native enterprises utilize Prometheus for metrics, while Grafana (64%) and OpenTelemetry (41%) exhibit swift adoption rates. The FinOps Foundation's 2023 research identifies observability as one of the three primary objectives for cloud cost optimization, with participants reporting that the growth rates of observability expenditures exceed those of infrastructure by

20-40%. The survey indicates that TechCo's encounter with financial difficulties, leading to the evaluation of open-source alternatives, reflects a common trend among other firms.

5.5.2 Contrasting case: smaller-scale partial migration

Not all migrations adhere to TechCo's path. Leveraging industry connections, the team of this paper documented a distinct instance: a fintech company ("FinStart," consisting of 35 engineers and 120 microservices, handling 150GB daily) endeavored a comprehensive transfer in 2023. After four months, they terminated the initiative and returned to their earlier commercial platform.

Key differences from TechCo:

- **Scale**: With a daily rate of 150GB, the projected annual savings of around $180,000 scarcely warranted the engineering costs (0.5 FTE = ~$75K).
- **Expertise gap:** The team did not have appropriate operational experience in Kubernetes, resulting in the learning curve surpassing earlier forecasts.
- Compliance requirements: An SOC 2 audit necessarily requires displaying vendor security certifications, which open-source solutions can't comply with, unless substantial additional effort is invested.

FinStart's experience demonstrates a boundary condition: firms lacking established platform engineering capabilities encounter marginal economics and increased execution risk when operating below 200 GB/day. The results reinforce the fact that migration does not necessarily benefit every organization.

6. Decision Framework

6.1 Criteria in Favor of Migration

Based on the analysis and case study performed by this article's team, migration is successful when:

Economic factors:

- The daily telemetry volume exceeds 100 GB
- Observability costs exceed 10% of the infrastructure budget
- Commercial contract renewal includes significant price increases
- Multi-year commitment to the commercial platform is ending

Technical factors:

- Kubernetes-based container orchestration is being used
- Structured logging practices are already established
- The team has experience in cloud-native operations
- Existing investment in Prometheus or Grafana

Organizational factors:

- The platform engineering team is either currently established or can be formed
- The leadership approves a migration schedule spanning several months
- Risk tolerance permits transient capability deficiencies

6.2 Indicators Against Migration

Migration may be inappropriate in the following scenarios:

Capability requirements:

- Heavy dependence on full-text log search over long retention periods
- Reliance on machine learning-based anomaly detection
- Requirement for vendor compliance certifications (SOC 2, HIPAA)

Resource constraints:

- Lack of specialized platform engineering resources
- The team possesses insufficient operational experience with Kubernetes
- Restrictive timelines inhibit phased migration

Economic factors:

- Daily throughput being under 50 GB
- Advantageous current contractual provisions
- Significant switching costs due to extensive commercial platform integration

6.3 Hybrid Approaches

The following hybrid approaches can be considered as alternatives to complete migration (which isn't always necessary):

- Log-only migration: Logs generally account for 60–70% of the observability expenses. Retaining commercial APM and transferring logs to Loki maximizes savings with lesser risk.
- **Environment-based split:** Development and staging environments are migrated to open-source platforms, while the production environment continues on commercial platforms. This lowers expenses while preserving production dependability.
- **Volume-based split:** High-volume services (accounting for over 80% of telemetry) transition to open-source platforms, while low-volume services persist on commercial platforms. This arrangement optimizes savings despite the limited scope of migration.

6.4 The following figure illustrates a decision tree:

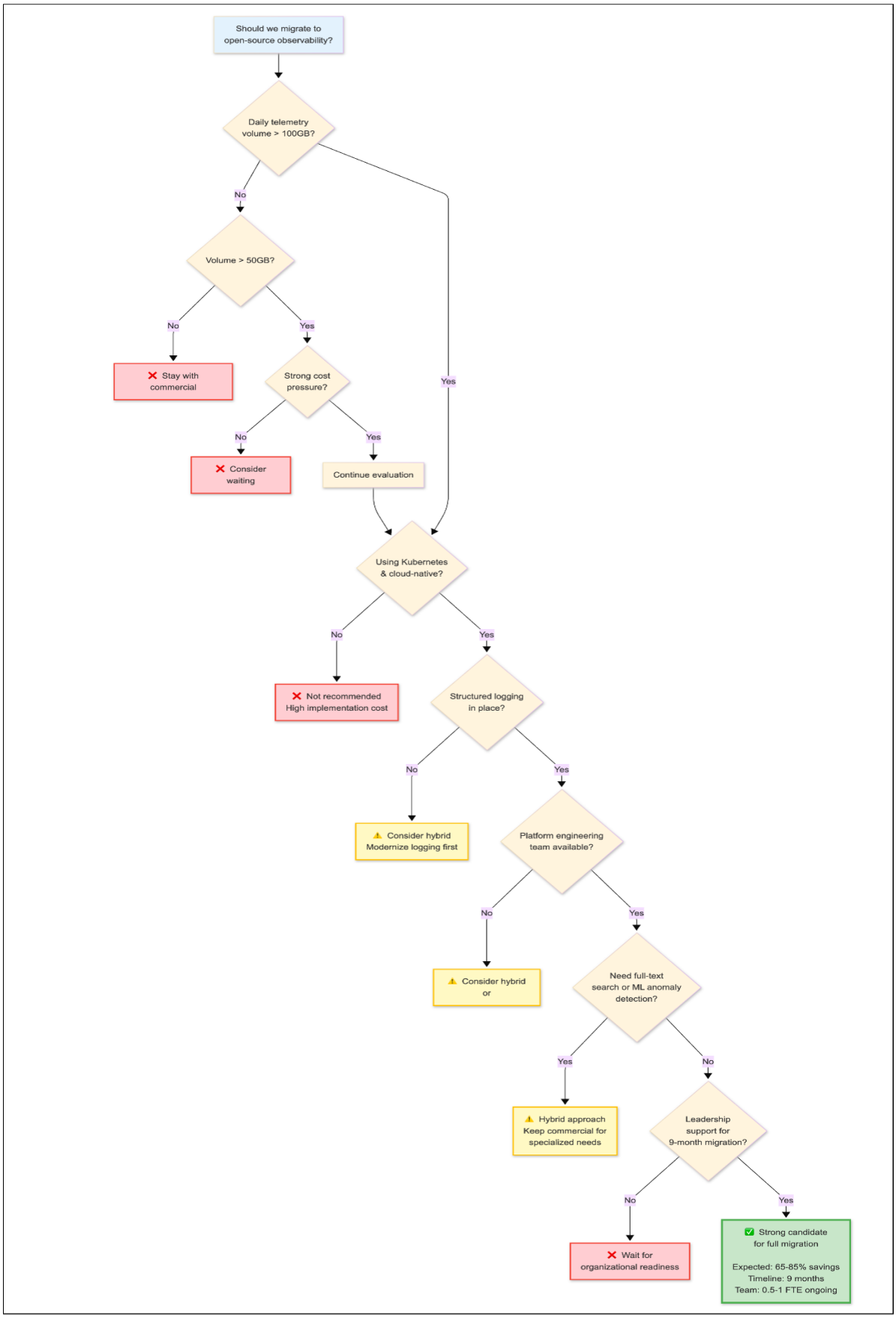

Figure 8: Decision tree

7. Limitations

This study has several limitations affecting generalizability:

- **Case study scope:** The analysis documented in this article is based on a singular successful migration (TechCo). Although the findings are augmented with survey data and a contrasting partial-migration instance (FinStart), the richness of quantitative data is predominantly confined to a single company. TechCo possessed already-existing Kubernetes proficiency, established DevOps techniques and implemented structured logging. Organizations not having these advantageous circumstances may face more challenging migrations.
- **Technology-specific scope:** This team primarily concentrated on Prometheus, Loki, Jaeger, and Grafana. Alternative stacks, such as ClickHouse for logs and VictoriaMetrics for metrics, may have varying trade-offs.
- **Temporal snapshot:** The capabilities outlined above may already be obsolete since open-source technologies are undergoing rapid evolution. Commercial platforms are also advancing, thereby reducing cost disparities.
- **Self-reported data:** The cost and performance metrics are derived from TechCo's internal monitoring.The article's team was unable to independently substantiate all these.
- **Limited observation period:** The results were tracked for 12 months post-migration.
- **Long-term expenses,** including major version upgrades, scaling difficulties, and employee attrition, remain unknown.

Future works should encompass:

- Multi-case studies across industries and scales
- Longitudinal analysis extending beyond twelve months
- Controlled performance benchmarking
- Assessment of alternative technology stacks

Conclusion

Open-source observability tools have sufficiently advanced to supplant commercial platforms for numerous organizations. The economic argument is compelling: organizations handling over 500 GB daily can realize a cost reduction of 70–85% with tolerable performance compromises. The results articulated in this article illustrate practical viability. TechCo accomplished the migration in nine months, yielding approximately $1.15 million in first-year savings. Query latency rose by 14-20%, evident in benchmarks yet undetectable during incident response. System availability exceeded 99.9%. Data from cross-organizational surveys and a contrasting unsuccessful migration scenario (FinStart) delineate boundary conditions: success necessitates adequate scale, platform engineering capability, and organizational preparedness. Migration is a challenging process; succeeding in it necessitates systematic recording methodologies, committed engineering resources, and organizational perseverance during an extended transition period. Not all organizations derive benefits: those operating below the 100–200 GB/day threshold, those reliant on machine learning-based functionalities, or those deficient in platform engineering capabilities should meticulously assess alternatives. For organizations having either Kubernetes-native architectures, established operational practices, tools suitable for production deployment, or facing cost constraints, open-source observability is indeed valuable. The organization's competency in facing challenges and successfully migrating is what ultimately matters.

Acknowledgments

The author thanks the engineering team at TechCo for sharing detailed migration data. The author also thanks [reviewers] for constructive feedback on earlier drafts.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
ORCID: https://orcid.org/0009-0009-9149-4811
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Gartner, "Market Guide for Application Performance Monitoring and Observability," Publication ID G00793855, January 2024. Available through Gartner subscription services.

[2] Cloud Native Computing Foundation, "CNCF Annual Survey 2023". [Online]. Available: https://www.cncf.io/reports/cncf-annual-survey-2023/

[3] J. Volz and B. Rabenstein, "Prometheus: A Next-Generation Monitoring System (Talk)", Usenix - SREcon15 Europe, 2015. [Online]. Available: https://www.usenix.org/conference/srecon15europe/program/presentation/rabenstein

[4] Cloud Native Computing Foundation, "OpenTelemetry Adoption and Implementation Report 2023," CNCF, June 2023. [Online]. Available: https://www.cncf.io/blog/2023/06/15/opentelemetry-momentum-2023/

[5] C. Sridharan, Distributed Systems Observability, O'Reilly Media - Humio, 2018. [Online]. Available: https://unlimited.humio.com/rs/756-LMY-106/images/Distributed-Systems-Observability-eBook.pdf

[6] B. H. Sigelman et al., "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure" Google Technical Report, 2010. [Online]. Available:

https://static.googleusercontent.com/media/research.google.com/en//archive/papers/dapper-2010-1.pdf

[7] OpenTelemetry Authors, "OpenTelemetry Specification". [Online]. Available: https://opentelemetry.io/docs/specs/otel/

[8] B. Płotka et al., "Thanos: Highly Available Prometheus Setup with Long Term Storage Capabilities," 2018. [Online]. Available: https://thanos.io/

[9] Y. Shkuro, Mastering Distributed Tracing, Packt Publishing, 2019.

[10] Cloud Native Computing Foundation, "Cloud Native Observability Microsurvey," CNCF, November 2023. [Online]. Available: https://www.cncf.io/blog/2023/11/20/observability-microsurvey/

[11] J. Mace et al., "Pivot Tracing: Dynamic Causal Monitoring for Distributed Systems", ACM Transactions on Computer Systems, 2018. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3208104

[12] A. Janes et al., "Open tracing tools: Overview and critical comparison", ScienceDirect - Journal of Systems and Software, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223001887

[13] R. K. Mehta and S. Agarwal, "Performance Evaluation of Distributed Tracing Systems in Microservice Architectures," in Proc. IEEE International Conference on Cloud Computing, 2023, pp. 156-163.

[14] FinOps Foundation, "State Of FinOps 2025 Report". [Online]. Available: https://data.finops.org/

[15] B. Burns et al., "Borg, Omega, and Kubernetes," Communications of the ACM, 2016. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2890784

[16] P. Pezaris, "The business value of observability: Insights from the 2023 Observability Forecast", New Relic, 2023. [Online]. Available: https://newrelic.com/blog/nerdlog/insights-2023-observability-forecast

Conflict of Interest Statement

This article received no external funding. The author has no financial relationships with any observability platform vendors, commercial or open-source.

AI Tools Disclosure

This manuscript was prepared with assistance from AI writing tools for grammar checking and formatting consistency. All technical content, analysis, data interpretation, and conclusions represent the author's original work. The case study data was collected through direct research and documentation review, not generated or synthesized by AI systems.