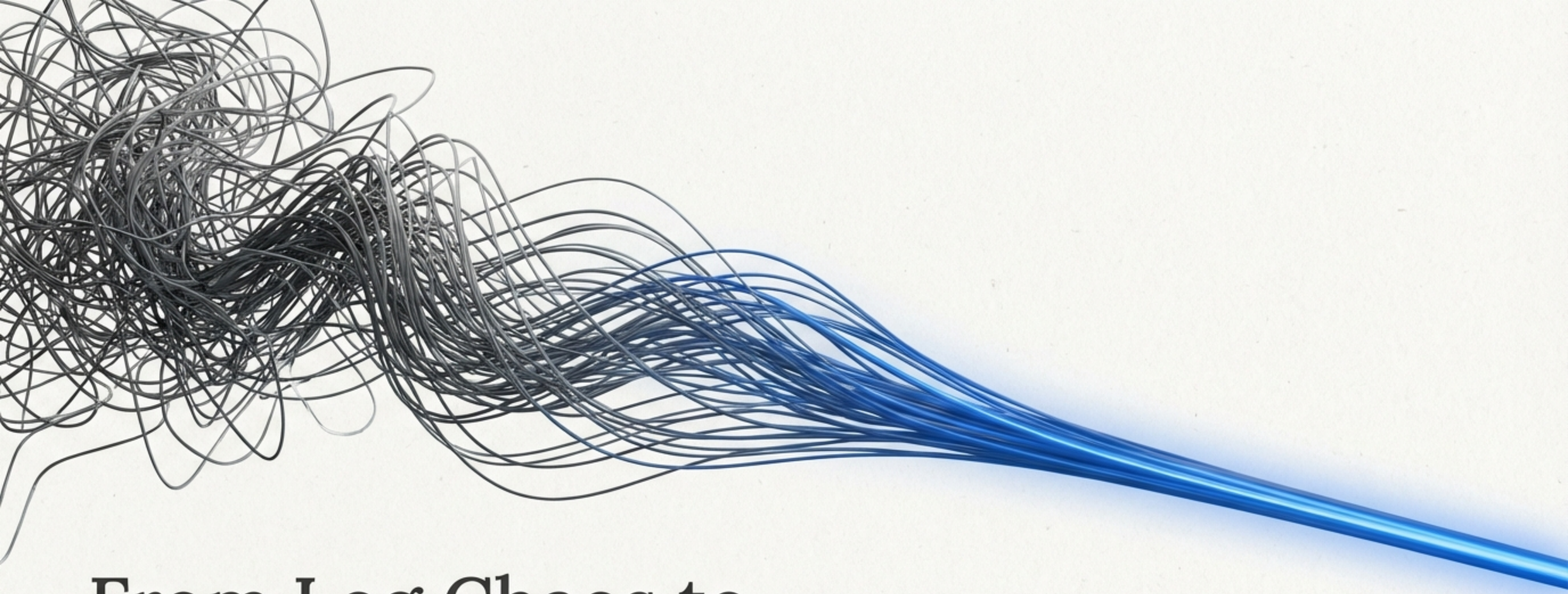


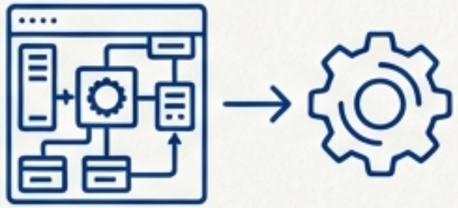
From Log Chaos to Strategic Insight

A Unified Enterprise Logging Strategy

The Vision: What Does Success Look Like?

Faster Issue Investigation

Drastically reduce mean time to resolution (MTTR) by allowing developers to trace issues across all services from a single interface.



Reduced Operational Overhead

Eliminate the need to maintain multiple, disparate logging solutions and remove direct logging to production databases.

Data for All Stakeholders

Provide a single source of truth for logs that serves Developers, Business Analysts, and Managers with tailored dashboards and alerts.

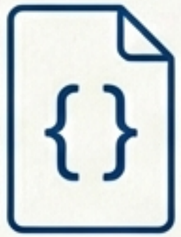
Future-Proof Architecture

Implement a scalable, cloud-native architecture that supports our current stack (Java, C#, Kubernetes) and is flexible enough for future platforms.

Our Guiding Principles

Centralized

All application logs flow into a single, unified logging platform.



Structured

Logs use a consistent JSON format instead of free-form text to enable reliable parsing and querying.

Searchable

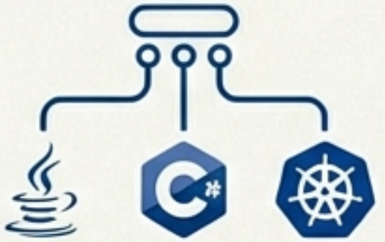
The system must provide powerful, fast querying for developers, BAs, and managers alike.

Searchable

The system must provide powerful, fast querying for developers, BAs, and managers alike.

Scalable

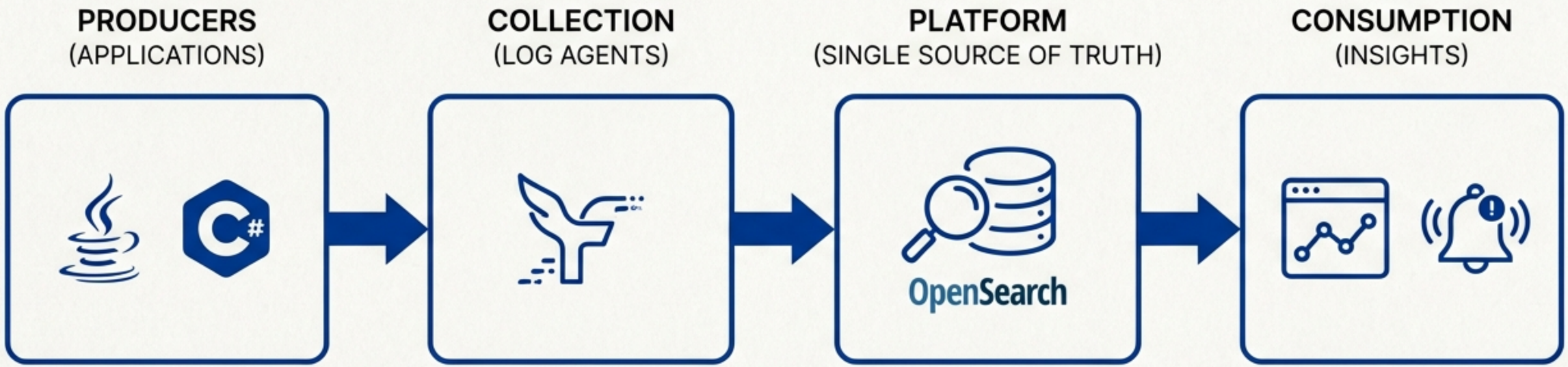
The platform must handle high log volume with minimal performance overhead for our applications.



Flexible

The solution must work seamlessly across our entire technology stack, including Java, C#, and Kubernetes.

The Recommended Architecture: Our Blueprint for Success



Applications log structured JSON to stdout/stderr

Lightweight agents collect, parse, and enrich logs

A centralized platform for ingestion, storage, and search

Dashboards, alerts, and powerful query interfaces for all users

The Cornerstone: A Common Log Schema

A consistent, structured log format is the shared language that unlocks the full power of centralized logging.

```
{
  "timestamp":    "...",
  "logLevel":     "INFO",
  "serviceName":  "...",
  "environment":  "...",
  "traceId":      "...",
  "userId":       "...",
  "message":      "...",
  "errorStack":   "..."
}
```

This schema ensures every log entry is searchable, filterable, and can be correlated across the entire request lifecycle.

The Crossroads: Should We Build a Custom Service or Adopt an Industry Standard?

Build a Custom Solution

‘Flexible, customizable, and under our control.’

Adopt a Proven Platform

‘Battle-tested, community-supported, and feature-rich.’

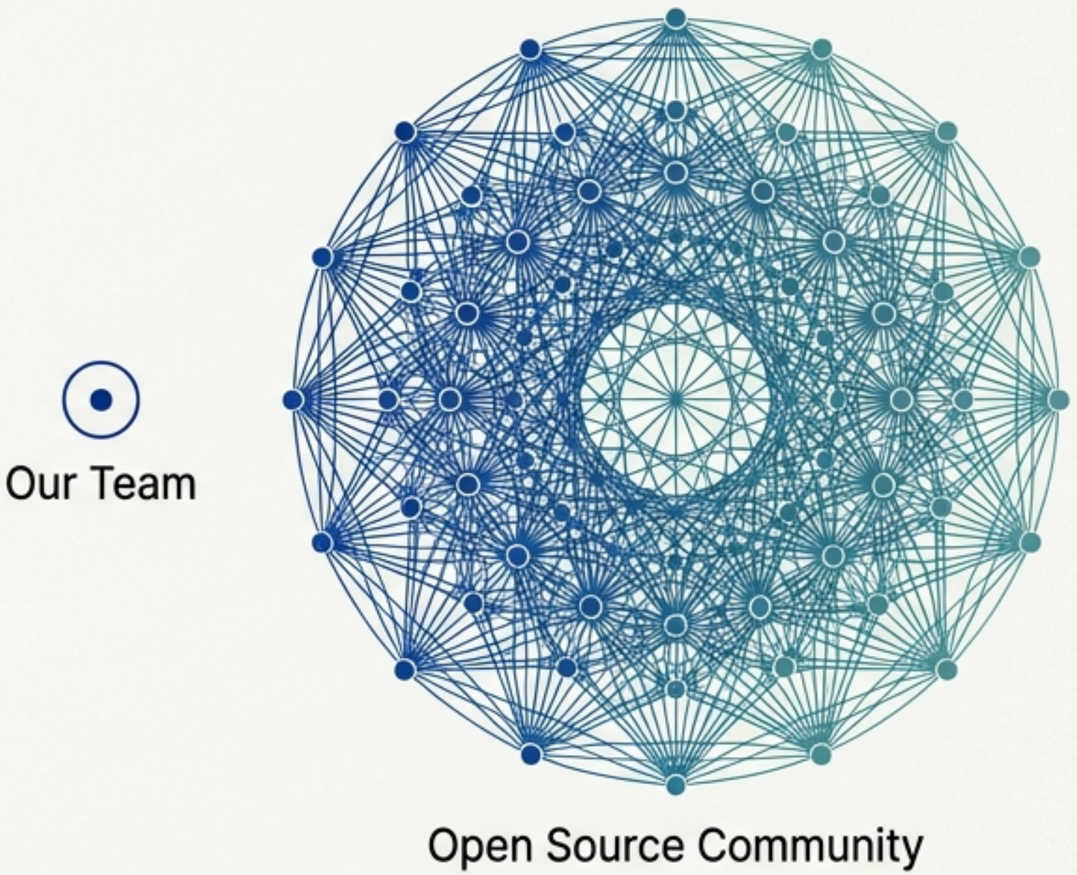
Let's examine the trade-offs of reinventing a solved problem.

The “Build” Trap: Reinventing a Highly Complex System

Building a custom logging service is a classic architectural trap. What seems simple on the surface hides immense complexity.

A production-grade system must handle:

- Millions of events per second
- Bursty traffic & log storms
- Backpressure to protect applications
- Schema evolution and management
- Efficient indexing & sub-second search
- Retention policies & tiered storage
- Role-Based Access Control (RBAC)
- User-friendly dashboards & alerting

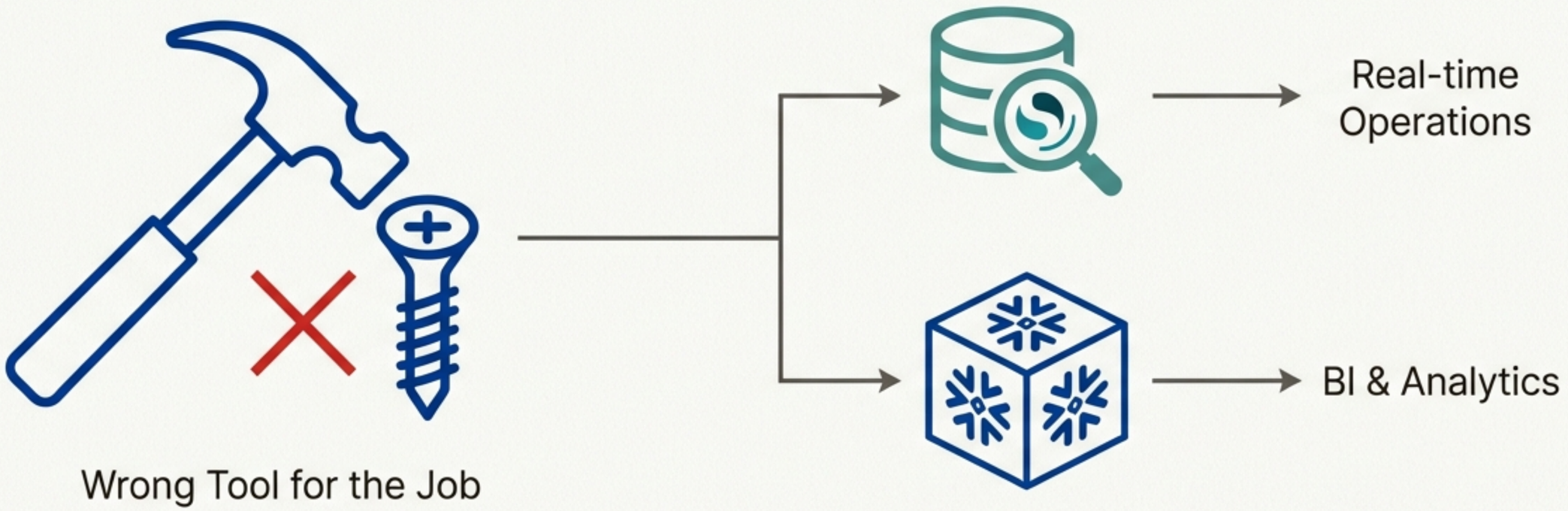


Open-source tools took hundreds of engineers years to get this right. Our focus should be on building business value, not foundational infrastructure.

The “Snowflake” Trap: The Right Tool for the Wrong Job

Snowflake is an exceptional platform for data analytics, business intelligence, and long-term data warehousing.

However, is it suited for the high-frequency writes and near-real-time search required for operational logging and incident response?



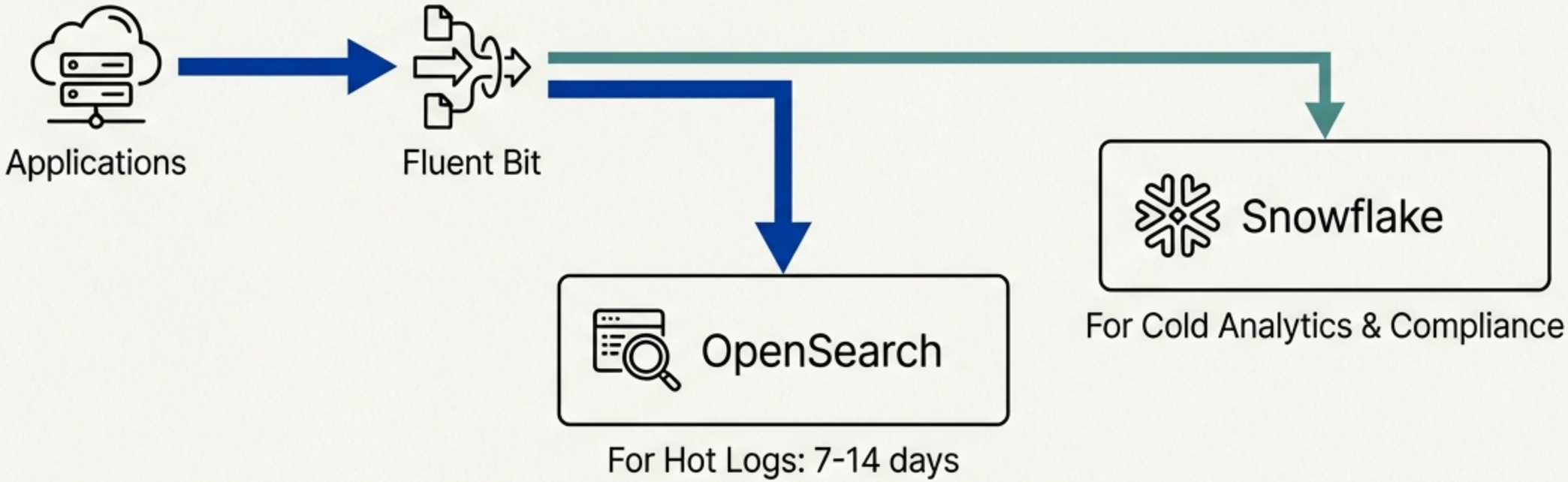
Wrong Tool for the Job

Real-time Operations vs. Historical Analytics: A Capabilities Mismatch

Requirement	Logging System (ELK/OpenSearch)	Snowflake
High-Frequency Writes	✓ Optimized	✗ Not Ideal
Near-Real-Time Search	✓ Seconds / Milliseconds	✗ Minutes
Free-Text Search	✓ Native	✗ Limited
Complex JSON Indexing	✓ Native	⚠ Possible, but Costly
Backpressure Handling	✓ Built-in	✗ Requires External Logic
Real-Time Dashboards	✓ Native	⚠ BI-Oriented
Real-Time Alerting	✓ Native	✗ Not Natively Real-time

The Best of Both Worlds: A Pragmatic Hybrid Architecture

The industry-standard pattern is to use the right tool for each job. A dedicated logging platform handles real-time needs, while Snowflake can be used for long-term analytics and compliance.



Hot Tier (OpenSearch)

Fast, interactive debugging and real-time dashboards for incident response.

Cold Tier (Snowflake)

Cost-effective long-term storage, deep historical analysis, and BI reporting.

Unlocking Value for Every Stakeholder

For Developers

- Trace requests across all microservices with a single query.
- View full, formatted stack traces instantly.
- Stop writing and maintaining custom logging code for SQL tables.

For Business Analysts

- Access pre-built dashboards for key business flows (e.g., failed transactions).
- Track volume trends and processing times without needing developer assistance.
- Query on human-readable business fields like `orderId` and `customerId`.

For Managers

- Monitor high-level KPIs like system health and error rates in real time.
- Receive automated alerts on SLA/SLO breaches.
- Gain insight into system performance and stability trends.

Our Practical 6-Step Migration Plan

1. **Define and Ratify the Common Log Schema**
   Agree on the universal JSON standard for all teams.
2. **Instrument Applications**
   Update Java (Logback/Log4j2) and C# (Serilog/NLog) apps to produce JSON logs to stdout.
3. **Deploy Log Agents**
   Roll out Fluent Bit/Filebeat as a DaemonSet in Kubernetes and as services on servers.
4. **Stand Up Central Platform**
   Provision and configure the chosen OpenSearch/ELK cluster.
5. **Migrate & Decommission**
   Gradually shift services to the new platform and decommission the legacy SQL Server-based logging.
6. **Build Shared Dashboards & Alerts**
   Create the initial set of high-value dashboards for all key stakeholders.

Our Recommendation: Clarity and Conviction

DO THIS

- ✓ Use OpenSearch / ELK or Loki for centralized logging.
- ✓ Standardize on a common JSON log schema.
- ✓ Deploy Fluent Bit / Filebeat for universal log collection.
- ✓ Keep logging infrastructure boring, proven, and reliable.

AVOID THIS

- ✗ Writing logs directly to Snowflake for real-time operations.
- ✗ Building custom logging microservices.
- ✗ Using SQL databases as primary log stores.

Logging is foundational infrastructure — build business value *on top of* it, not from scratch.

Next Steps & Discussion

Formalize the Decision: Draft an Architecture Decision Record (ADR) to document this strategy.

Design the Hybrid Architecture: Detail the integration points between OpenSearch and Snowflake.

Establish a Governance Team: Form a small group to oversee the schema definition and rollout plan.

Open Discussion: Address any remaining questions and concerns.