Unified Enterprise Logging Strategy

This document summarizes the architectural discussion and recommendations for implementing a unified, scalable, and industry-standard logging strategy across Java and C# applications.

1. Current Challenges

The organization operates a mixed technology stack. Java applications run on Kubernetes and rely on kubectl-based log access, while C# applications write logs directly into SQL Server tables. This fragmented approach makes it difficult for developers, business analysts, and managers to investigate production issues efficiently.

2. Key Requirements for Centralized Logging

- Centralized access to logs across all applications and platforms
- Structured, JSON-based logs for consistency and searchability
- High performance and scalability under heavy load
- Minimal impact on application performance
- Usable dashboards and queries for both technical and non-technical users

3. Industry-Standard Logging Solutions

The industry strongly favors adopting proven open-source or commercial logging platforms rather than building custom logging systems.

Commonly used solutions include:

- OpenSearch / ELK Stack (Elasticsearch + Kibana + Fluent Bit)
- Grafana Loki (often paired with Prometheus and Grafana)
- Splunk (commercial, enterprise-grade)
- Datadog or New Relic (SaaS observability platforms)

4. Evaluation of a Custom Logging Microservice

Building a custom asynchronous logging microservice is generally not recommended for centralized enterprise logging. Logging platforms are complex infrastructure systems that must handle bursty traffic, backpressure, indexing, retention, security, and near-real-time search.

Developing and operating such a system in-house introduces high risk, long-term maintenance cost, and operational burden, with little competitive advantage compared to mature open-source or commercial solutions.

5. Suitability of Snowflake for Logging

Snowflake is an excellent platform for large-scale analytics and historical data processing, but it is not designed to be a primary log ingestion or real-time troubleshooting system.

- High-frequency log ingestion is inefficient and costly
- Near-real-time search and debugging are not Snowflake’s strengths
- Free-text and ad-hoc log search is limited compared to log-native platforms

6. Recommended Architecture

A best-practice approach is to use a dedicated logging platform as the system of record, with optional downstream integration to Snowflake for long-term analytics and reporting.

Typical flow:

- Applications emit structured JSON logs to stdout or local files
- Fluent Bit or Filebeat collects and enriches logs
- Logs are sent to OpenSearch / ELK / Loki for real-time search and dashboards
- Optionally, logs are streamed to Snowflake for historical analytics

7. Final Recommendation

Adopt a proven, industry-standard logging platform for centralized logging. Avoid building a custom logging microservice and avoid using Snowflake as the primary log store. This approach provides flexibility, scalability, performance, and faster incident resolution while minimizing operational risk.