



ENTERPRISE SEARCH AT SCALE: TURNING COMPLEXITY INTO CAPABILITY

Abstract

Enterprise search has become a productivity and governance challenge. Business data is expanding across applications, documents, workflows and operational repositories. Users expect discovery to be fast, relevant and secure. However, many organizations still route search workloads through transactional systems, which are not designed for flexible retrieval at enterprise scale.

This whitepaper presents the view that enterprise search should be treated as a controlled search and retrieval layer, rather than as an afterthought embedded in application databases. It explains the problem, the architectural challenges, the role of Amazon OpenSearch Service as a managed search and retrieval layer, and a reusable adoption pattern that other organizations can apply.

A Point of View on Turning Search Bottlenecks into Governed Enterprise Retrieval Capability

A publication-ready whitepaper for leaders and architects modernizing complex enterprise search through a governed, reusable OpenSearch-backed retrieval pattern.

Attribute	Value
Primary Audience	CIOs, Enterprise Architects, Solution Architects, Application Leaders, Platform Engineering, DevOps and QA
Scope	Business context, decision framework, reference architecture, governance, case study, measurement, risks and adoption guidance
Source Basis	Official AWS/OpenSearch documentation and sanitized lessons from an enterprise search implementation

Executive Summary: Why This Matters

Executive point of view

The central argument of this paper is simple: search performance problems are often architecture problems. Amazon OpenSearch Service helps address the issue by separating search from transactional processing, converting complex business data into search-ready documents, protecting retrieval through governed APIs, and operating the index with monitoring, reconciliation, lifecycle and access controls.

Executive takeaways



Business issue: Slow, fragmented, and inconsistent search experiences reduce user productivity and erode trust in enterprise information systems.



Architectural response: Decouple search-heavy workloads from core transactional systems by introducing a governed read model optimized for retrieval performance.



Technology: Use Amazon OpenSearch Service as a managed retrieval layer for searchable documents, filtering and operational analytics.



Governance requirement: Keep source systems authoritative and expose search through controlled APIs with identity, authorization, audit and reconciliation controls.



Reusable value: Leverage a reusable, governed search architecture across multiple domains to optimize discovery, simplify retrieval workflows, strengthen monitoring and telemetry, and support enterprise-scale search operations.

Decision Area	Recommended Position
Adoption stance	Adopt for scalable search, filtering, analytics, observability and retrieval use cases.
Integration model	Expose via application/platform APIs, Step Functions or service orchestration. Do not expose indexes directly to end users.
Data ownership	Keep authoritative ownership in source systems; synchronize derived searchable documents to OpenSearch.
Operational model	Use batch, event-driven or managed ingestion with retry, throttling, reconciliation and alerting.

Problem Statement: Enterprise Search at Scale

Enterprise search becomes challenging when simple application-level search patterns are expected to support complex, high-volume retrieval needs. What starts as database queries, keyword filters and reporting screens can quickly become difficult to maintain as data grows across business objects, documents, workflows and access-controlled repositories. The key constraints are outlined below.

- Transactional databases are optimized for correctness and OLTP workloads, not always for complex relevance ranking, fuzzy matching, faceting and large search result exploration.
- Data is fragmented across applications, repositories and operational systems, causing users to search in multiple places and reconcile results manually.
- Dynamic business attributes and nested relationships can lead to expensive joins, slow queries and inconsistent user experiences when forced into relational search paths.
- Security is harder when search indexes aggregate data across domains, because broad retrieval can expose records unless access filtering is designed into the query path.
- Modern enterprise applications need trusted retrieval layers that provide current, governed and context-rich data without overloading transactional systems.

Challenges to Solve Before Modernizing Search

A search modernization program succeeds only when it addresses the full operating model, not just query speed. The architecture must define how data is shaped, secured, synchronized, observed, measured and reused. Without strong architecture and operating-model discipline, a faster index can still produce stale results, weak governance, rising cost or inconsistent user trust.

Search Evolution Area	Enterprise Relevance
Traditional application search	Keyword search, filters, facets and sorting inside business applications.
Knowledge discovery	Search across documents, metadata, policy repositories, articles and knowledge bases.
Operational analytics	Logs, events, audit records and observability data used to troubleshoot and detect patterns.
Advanced retrieval	Relevance tuning, synonyms, analyzers, scoring, ranking and metadata-aware retrieval for richer search experiences.

Solution Approach: OpenSearch as a Governed Retrieval Layer

Amazon OpenSearch Service is used in this whitepaper as a managed retrieval layer, not as a system of record. Source applications continue to own business truth, while selected data is transformed into search-ready documents and indexed for fast discovery, filtering, relevance and operational insight. The reusable

pattern is to move search complexity out of transactional paths and into a governed retrieval architecture where transformation services create search-ready documents, OpenSearch stores derived indexes for retrieval, and controlled APIs enforce identity, authorization, response shaping and auditability.

Decision Framework: Determining Where OpenSearch Delivers Enterprise Value

Scenario	Rating	Recommended Direction
Full-text application search	★★★★★	Use OpenSearch when relevance, tokenization, analyzers, faceting and filtering are required.
Dynamic attribute search	★★★★★	Use indexed key-value or explicit field mappings with multi-fields to support exact and analyzed behavior.
Transactional updates and referential integrity	★☆☆☆☆	Use the source database for OLTP. Synchronize a derived index into OpenSearch.
Large analytical joins	★★★☆☆	Pre-compute or de-normalize during ingestion. Avoid broad query-time joins.
Log analytics and observability	★★★★★	Use OpenSearch domains/serverless collections and managed ingestion where applicable.
Advanced relevance and hybrid retrieval	★★★☆☆	Use analyzers, synonyms, scoring, field boosts and metadata filters when user experience requires higher relevance and precision.
Enterprise reporting over historical data	★★★☆☆	Consider data warehouse/Lakehouse for deep BI; use OpenSearch for interactive search and operational exploration.

Target Architecture

The target reference architecture answers an executive question: How can the enterprise optimize retrieval performance and improve scalability without weakening governance or overloading core systems? The answer is an API-mediated retrieval layer backed by derived OpenSearch indexes. Source systems publish or synchronize data through controlled ingestion services, while users and applications retrieve results through governed APIs.

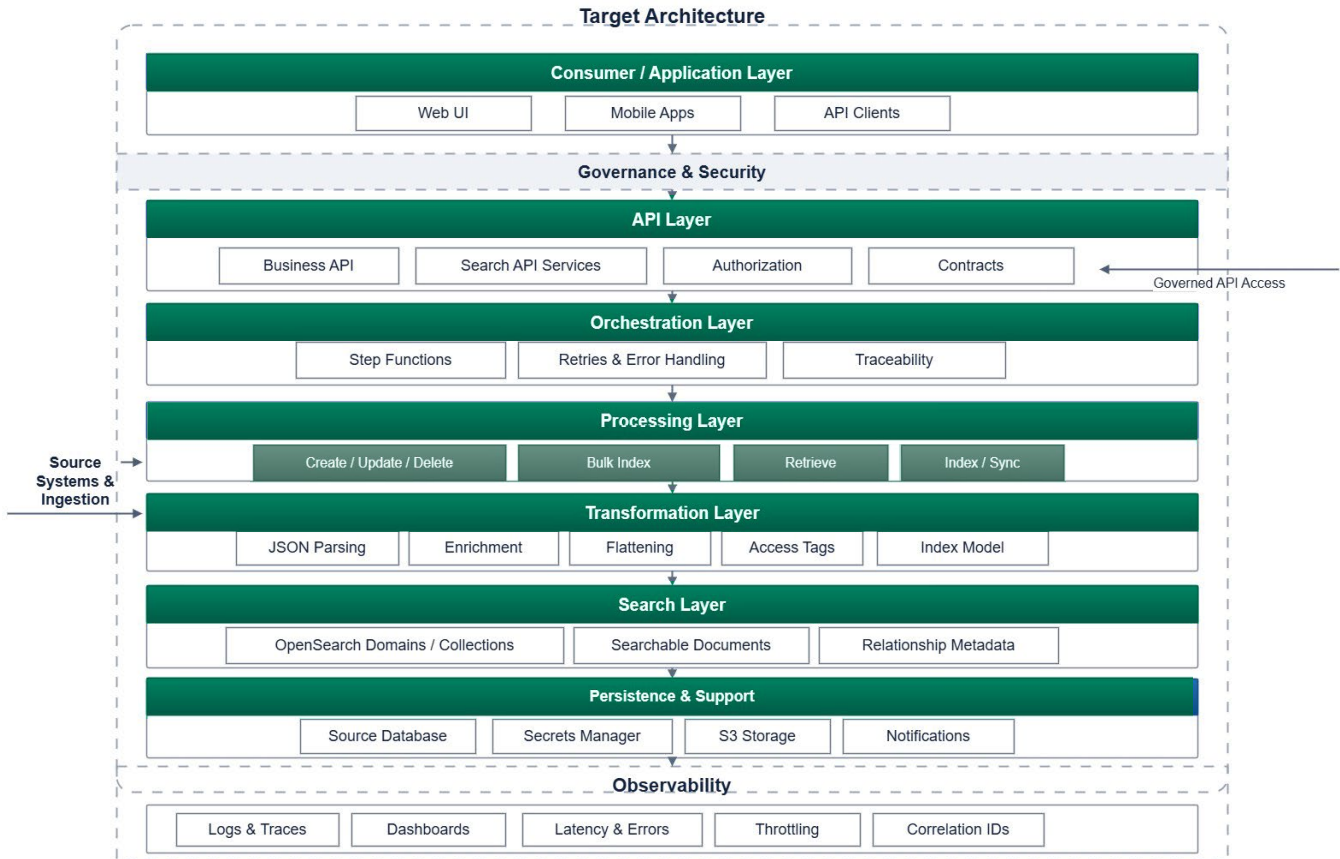


Figure 1. Target architecture: governed API-mediated search over derived OpenSearch indexes.

Layer	Components	Responsibilities
Consumer/Application Layer	Web UI, Mobile apps, API clients	Initiate search, template creation, retrieval and export requests.
API Layer	Business API and search API services	Validate authorization, route requests, standardize response payloads and enforce contracts.
Orchestration Layer	AWS Step Functions	Coordinate retries, defaulting, error handling and execution traceability.
Processing Layer	AWS Lambda or services	Create, update, delete, bulk index and retrieve search documents.
Transformation Layer	JSON parsing and enrichment	Flatten data, resolve reference values, derive access tags and build index models.
Search Layer	OpenSearch domains or collections	Store searchable documents, relationship metadata and table records.
Persistence and Support	Database, Secrets Manager, S3, notifications	Maintain source data, credentials, result files and user notifications.
Observability	CloudWatch, logs, traces, dashboards	Track latency, errors, throttling, failures and correlation IDs.

Architecture principle

OpenSearch should operate as a derived search index, not as the authoritative data store. Transactional systems must continue to own the source data. Index refresh, reconciliation, delete propagation, rollback and cutover should be designed as core platform capabilities.

Implementation Considerations

Implementation planning should translate the target architecture into clear execution choices. Before indexing begins, teams need to decide how data will be ingested, how source records will be transformed into search-ready documents, and how users will access results through governed APIs. The following points outline the practical considerations that should guide this implementation.

Use bulk indexing for initial loads, reindexing and scheduled synchronization.

Transform source records into search-ready documents with explicit mappings, normalized values, access tags and traceability metadata.

Avoid broad query-time paths; de-normalize or pre-compute relationships during ingestion where possible.

Expose search through APIs rather than direct user access to indexes.

Use asynchronous export for large result sets instead of returning large payloads through the UI.



Security, Governance and Operations

Because search indexes consolidate data from multiple systems, governance must be designed into the access path from the start. The baseline model should include API authorization, IAM and network controls, encryption, audit logs, least privilege, sensitive log masking, retention rules, reconciliation and operational monitoring. The table below translates these governance needs into practical control areas and concise implementation guidance.

Area	Concise Guidance
Access control	Validate identity at the API layer and apply access tags, IAM scope and negative authorization testing.
Data protection	Classify indexed fields, minimize sensitive attributes, encrypt data and mask confidential payloads in logs.
Performance	Validate batch size, query design, mappings, shard strategy, refresh behavior and representative query latency.
Operations	Monitor indexing failures, search latency, slow logs, execution history, source-to-index drift and retry/replay outcomes.
Cost	Control storage growth, shard count, replica strategy, retention, lifecycle policies and workload-specific deployment model.
Data privacy	Apply privacy-by-design controls by minimizing indexed personal or sensitive data, enforcing purpose-based access, masking or tokenizing confidential fields where required, and validating retention, consent, and regulatory obligations before indexing.



Enterprise Case Study: Anonymized Aerospace Organization

This anonymized case study, based on a large aerospace organization, highlights a common enterprise search pain point: users needed fast and reliable discovery across dynamic business objects, relationships, table data and workflow/search templates, but relational query paths resulted in slow performance, complex filtering and governance challenges. The issue was addressed by redesigning search as a governed, OpenSearch-backed retrieval capability. Source ownership remained with the application, search-ready documents were created through transformation services, access was enforced through APIs, and operational controls were added for synchronization, monitoring and reconciliation.

Case Study Area	Description
Business context	● Users needed fast discovery across dynamic business objects, relationships, table data and workflow/search templates.
Challenge	● Relational search across nested and dynamic attributes created performance, usability and governance challenges.
Solution	● The source application remained authoritative. ● Lambda-based transformation created search-optimized documents and indexed them into OpenSearch. ● APIs enforced token validation, access tags and response shaping.
Key design decisions	● Use OpenSearch as a derived index. ● Protect access behind APIs. ● Transform nested JSON into key-value searchable structures. ● Use bulk indexing for synchronization. ● Implement monitoring and reconciliation.
Operational controls	● CloudWatch logs. ● Step Functions history. ● Retry handling. ● Batch tuning. ● Timeout notifications. ● Sensitive log masking. ● Index drift checks.
Scale and performance outcome	● The implementation created three search indexes. ● One large index held approximately 10.6 million records. ● A representative full-text search previously took about five minutes when executed through a relational database search path that had to evaluate dynamic attributes, nested relationships and table data using database-side full-text search and complex filtering. ● After moving retrieval to OpenSearch-backed derived indexes, the same type of search was reduced to approximately 10–15 seconds. ● The improvement was driven by search-specific indexing, transformation of dynamic data into searchable structures, optimized query design, batch tuning and the operational controls described in this paper.
Lessons learned	● Model search documents for user questions, not source-table structures. ● Design access filtering, deletion, reconciliation and index lifecycle before production.

The broader lesson is that the outcome was driven by architecture, not indexing alone. The solution worked because the team clarified source ownership, modeled documents around user search questions, moved heavy retrieval away from transactional paths, enforced access through APIs and added operational controls for synchronization, monitoring and reconciliation.

Reusable Value for Other Organizations

Organizations facing similar constraints can reuse this pattern when search is slow, fragmented, expensive to change, difficult to govern or operationally opaque. The adoption model helps teams begin with a high-value search bottleneck and mature toward a reusable enterprise retrieval platform.

Stakeholder	Value Created
Business users	Faster discovery, better filtering and more consistent access to relevant information.
Application teams	Decoupled search-heavy workloads from transactional systems while preserving source ownership.
Architecture teams	A repeatable governed retrieval pattern that can be reused across business domains.
Security and compliance teams	Consistent authorization, audit, classification and retention controls for search experiences.
Operations teams	Better visibility into index freshness, failures, latency, drift and cost before search becomes a production risk.
Data privacy	Apply privacy-by-design controls by minimizing indexed personal or sensitive data, enforcing purpose-based access, masking or tokenizing confidential fields where required, and validating retention, consent, and regulatory obligations before indexing.

Business Outcomes and Measurement Framework

This measurement framework should be used to validate business impact before publication. It separates potential value areas from approved benefit claims and ensures that any quantified outcome is supported by auditable evidence.

Outcome Area	Metric	Evidence Source
Search latency	P50/P95 response time for common queries	API gateway, application logs and OpenSearch slow logs
Index freshness	Time between source update and searchable availability	Ingestion timestamps and reconciliation jobs
Index completeness	Document count match between source and index	Source counts and OpenSearch counts
User productivity	Reduction in manual lookup effort	User study, workflow telemetry or survey
Operational reliability	Failed syncs, retries and replay success	Step Functions, Lambda and queue telemetry



Risks and Mitigations

Risk	Impact	Mitigation
Index drift from source data	Stale or incomplete results	Scheduled reconciliation, replay and count comparison.
Overly broad access	Sensitive record exposure	Access tags, user identity checks and negative authorization tests.
Large batch timeout	Incomplete indexing	Tune batch size, use chunking and targeted replay.
Dynamic schema inconsistency	Missed matches or query failures	Mapping standards, type conversion rules and regression tests.
Sensitive logging	Token or payload exposure	Mask tokens and log correlation IDs instead of full payloads.
No lifecycle policy	Cost and performance degradation	Retention, rollover, snapshots and tiering.
Manual cutover errors	Wrong active index or downtime	Automate aliasing/cutover and rollback validation.

Best Practices and Reuse Checklist

Category	Checklist
Design	Define searchable entities, source ownership, document IDs, explicit mappings, multi-fields and refresh frequency.
API governance	Expose controlled APIs; standardize request and response contracts; avoid direct UI-to-index access.
Security	Validate tokens, scheduled-job identities, access tags, IAM scope and sensitive log masking.
Transformation	Flatten nested data, resolve reference values, maintain type-aware mappings and attach traceability metadata.
Performance	Benchmark batch sizes, concurrency, memory, timeout, query patterns and refresh settings.
Operations	Enable CloudWatch logs, execution history, dashboards, alerts and runbooks.
Testing	Cover invalid IDs, unauthorized users, no data, bulk sync, specific ID sync, update, delete and timeout notifications.
Release	Use load tests, reconciliation, rollback plan, deployment checklist and legal/IP review before external release.

References

Reference Area	URL	How It Was Used
Amazon OpenSearch Service overview	https://docs.aws.amazon.com/opensearch-service/latest/developerguide/what-is.html	Managed service positioning, deployment, scaling, security and operations context.
Amazon OpenSearch Service quotas	https://docs.aws.amazon.com/opensearch-service/latest/developerguide/limits.html	Service quota and regional planning guidance.
OpenSearch Serverless overview	https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-overview.html	Serverless option, collection types, scaling and limitations.
Amazon OpenSearch Ingestion	https://docs.aws.amazon.com/opensearch-service/latest/developerguide/ingestion.html	Managed ingestion, Data Prepper, transformation and CloudWatch monitoring.
Amazon Data Firehose integration	https://docs.aws.amazon.com/opensearch-service/latest/developerguide/integrations-fh.html	Streaming delivery, optional transformation and backup considerations.
Security in Amazon OpenSearch Service	https://docs.aws.amazon.com/opensearch-service/latest/developerguide/security.html	Security governance, IAM, data protection and auditability.
OpenSearch Query DSL	https://docs.opensearch.org/latest/query-dsl/	Search API, query patterns, filtering, scoring and aggregations.
OpenSearch SQL limitations	https://docs.opensearch.org/latest/sql-and-ppl/limitation/	Join, pagination and SQL/PPL limitations.

Conclusion: From Search Bottleneck to Reusable Retrieval Capability

The conclusion brings together the core message of this whitepaper: enterprise search modernization should be treated as an architecture and operating-model decision. The points below summarize the key principles and practical steps for moving from a search bottleneck to a reusable retrieval capability.

- Enterprise search modernization should begin with the business problem, not the technology choice.
- Search becomes a priority when it is slow, fragmented, difficult to govern or expensive to operate through transactional systems.
- Amazon OpenSearch Service can provide a managed and scalable retrieval layer.
- It should be designed as a derived, governed and observable search capability.
- Identify the highest-value search bottleneck first.
- Define source ownership and access rules.
- Transform source data into search-ready documents.
- Expose retrieval through governed APIs.
- Validate performance with representative workloads.
- Operationalize the platform with monitoring, reconciliation, lifecycle and cost controls.

Author



Dolphin Dua, Technology Architect, Infosys Ltd



For more information, contact askus@infosys.com



© 2026 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.