Case Management Search Architecture

Technical Analysis Report

Comprehensive Analysis of Search Features, Elasticsearch Integration, and Analytics Module in the Case Management Domain

Generated on: 2025-09-19 10:38:27

Datadog Case Management Team

Key Statistics

- 393+ Elasticsearch fields
- 7 specialized query handlers
- Nested custom attributes support
- Multi-environment deployment (US1, EU1, AP1+)

Executive Summary

The Case Management domain implements a sophisticated search architecture built on Elasticsearch, providing comprehensive search capabilities for case data across multiple Datadog environments.

KEY FINDINGS:

 SEARCH ARCHITECTURE
• Multi-layered architecture: API → Handlers → Parser → Elasticsearch
• 7 specialized query handlers for different search use cases
• ANTLR-based query parsing with custom grammar
• Project-based security filtering integrated at query level

 ELASTICSEARCH INTEGRATION
• Primary index: "cases" with 393+ mapped fields
• Custom attributes stored as nested objects (not flattened)
• Dynamic runtime field support for complex calculations
• Multi-environment deployment across US, EU, and AP regions

 ANALYTICS MODULE
• Real-time time-series aggregations with configurable intervals
• Support for 20+ groupBy fields plus custom attributes
• Complex nested aggregations for custom attribute analysis
• Metrics: count, sum, avg, max, min, percentiles (pc50, pc95, pc99)

 ADVANCED FEATURES
• Faceted search with autocomplete
• Custom attribute querying via nested Elasticsearch structures
• User/project/case-type enrichment with UUID resolution
• Pagination, sorting, and filtering capabilities

 PERFORMANCE OPTIMIZATIONS
• Query builder with project restriction filtering
• Aggregation bucket limits (max 1000 buckets)
• Caching and timeout configurations
• Multiple search indexer applications for real-time updates

The system demonstrates enterprise-grade search capabilities with strong separation of concerns, robust error handling, and comprehensive logging for debugging and monitoring.

Search Architecture Overview

COMPONENT BREAKDOWN:

1. CLIENT LAYER
   • Web UI search interfaces
   • API clients (REST/gRPC)
   • External integrations

2. API GATEWAY
   • Case Rapid API (/search endpoints)
   • Request validation and routing
   • Authentication and authorization

3. QUERY HANDLERS (7 specialized handlers)
   • SearchHandler: Basic search with pagination/sorting
   • AnalyticHandler: Time-series aggregations and metrics
   • SearchFacetValuesHandler: Faceted search with partial matching
   • SearchAutocompleteHandler: Search suggestions
   • GetFacetsHandler: Available facets retrieval
   • GetAttributeValuesHandler: Attribute value enumeration
   • GetAttributesKeyHandler: Attribute key discovery

4. QUERY PROCESSING
   • ANTLR parser with custom grammar (grammar.peg)
   • ESQueryBuilder converts parsed queries to Elasticsearch DSL
   • Support for complex nested queries and custom attributes
   • Project-based access control filtering

5. ELASTICSEARCH CLUSTER
   • Primary index: "cases" (393+ fields)
   • Nested custom_attributes structure
   • Analytics fields with date_range spans
   • Multi-environment deployment

6. ENRICHMENT SERVICES
   • UserService (OUI integration) - UUID to email resolution
   • ProjectService - Project ID to name mapping
   • CaseTypeService - Case type enrichment
   • Real-time data enrichment post-query

Elasticsearch Index Structure

INDEX: "cases" (393+ Fields)

CORE FIELDS:
• internal_id, case_id, public_id (keyword) - Unique identifiers
• assignee_id, created_by, project_id, org_id (keyword/long) - References
• status, priority, type_id (long) - Enumerated values
• created_at, modified_at, closed_at (date) - Timestamps
• title, description (text) - Full-text searchable content

CUSTOM ATTRIBUTES (NESTED):
• Type: "nested" (enables complex nested queries)
• Structure:
  - key (keyword) - Attribute name
  - value_text (text) - String values
  - value_number (text + as_double field) - Numeric values
• Query Pattern: nested → filter → terms → reverse_nested

ANALYTICS FIELDS:
• analytics.status_* (object) - Status duration tracking
• spans (date_range) - Time period definitions
• total (long) - Calculated duration metrics
• Supports time-series aggregations

ADDITIONAL PROPERTIES:
• event_management.* - Incident management data
• on_call.* - Responder and escalation information
• change_request.* - Change request metadata
• campaign.* - Campaign tracking data
• Flattened structure for direct querying

INTEGRATION FIELDS:
• jira_issue.* - Jira integration metadata
• servicenow_ticket.* - ServiceNow synchronization
• insights.* - ML/AI insight references
• notification_handles.* - Alert routing information

PERFORMANCE CONSIDERATIONS:
• Dynamic mapping: false (explicit schema control)
• Text fields: analyzed for full-text search
• Keyword fields: not_analyzed for exact matching
• Date fields: optimized for range queries
• Nested fields: isolated document storage for complex queries

Analytics Module Integration

ANALYTICS HANDLER (analytic_handler.go:1520 lines)

SUPPORTED METRICS:
• count - Document counting
• sum, avg, max, min - Statistical aggregations
• pc50, pc95, pc99 - Percentile calculations
• Custom metric definitions via metrics framework

GROUP BY FIELDS (20+ supported):
• Standard: assignee, creator, status, priority, project, service, team
• Time-based: created_at, modified_at, closed_at
• Custom: custom_attributes.* (any custom attribute key)
• Advanced: change_request.risk, event_management.event_aggregation_keys

TIME BUCKETING:
• Configurable intervals: 1s to 1year
• Default intervals: [1s, 2s, 5s, 10s, 1m, 2m, 5m, 10m, 1h, 1d, 7d, 30d, 365d]
• Max bucket limit: 1000 buckets (configurable)
• Automatic interval adjustment to stay under limit

AGGREGATION STRUCTURE:
1. Group By Aggregation (if specified)
   • Terms/MultiTerms for standard fields
   • Nested aggregation for custom attributes
2. Time Bucket Aggregation
   • DateRange aggregation with calculated intervals
3. Metric Aggregation (if specified)
   • Applied within each time bucket

CUSTOM ATTRIBUTE SUPPORT:
• Complex nested aggregation chain:
  nested → filter(key match) → terms(values) → reverse_nested → date_range → metric
• Supports both text and numeric custom attribute values
• text_values: uses .keyword field for exact matching
• number_values: uses .as_double field for numeric operations

ENRICHMENT PROCESS:
• UUID to human-readable name conversion
• User resolution via OUI service (UserService)
• Project name resolution (ProjectService)
• Case type name resolution (CaseTypeService)
• Applied post-aggregation for performance

RESPONSE FORMAT:
• MetricsBuffer with LZ4 compression
• Time-series data optimized for frontend consumption
• Groups, epochs, and values arrays
• Supports Datadog's metrics infrastructure

Query Processing Flow

QUERY PROCESSING PIPELINE:

1. INPUT PARSING (ANTLR Parser)
   • Grammar-based parsing (grammar.peg)
   • Supports complex boolean logic: AND, OR, NOT
   • Field-specific queries: status:open, priority:high
   • Range queries: created_at:[2023-01-01 TO 2023-12-31]
   • Custom attribute queries: custom_attributes.environment:prod

2. QUERY BUILDING (ESQueryBuilder)
   • Converts parsed tree to Elasticsearch Query DSL
   • Handles different query types:
     - Match queries for full-text search
     - Term queries for exact matching
     - Range queries for numeric/date ranges
     - Nested queries for custom attributes
     - Boolean queries for complex logic

3. SECURITY FILTERING
   • Project-based access control
   • Automatic org_id filtering
   • Restricted project ID filtering via ProjectService
   • Work type filtering (CASE, etc.)

4. ELASTICSEARCH EXECUTION
   • Query optimization and caching
   • Timeout configuration (5s for analytics)
   • Index selection based on configuration
   • Result aggregation and processing

5. RESULT PROCESSING
   • Hit extraction from Elasticsearch response
   • Document deserialization (JSON → protobuf)
   • Pagination and sorting application
   • Total count and page count calculation

6. ENRICHMENT
   • UUID resolution to human names
   • Project name enrichment
   • User email/handle resolution
   • Case type name mapping

7. RESPONSE FORMATTING
   • Protobuf serialization
   • Metadata addition (totals, pagination info)
   • Error handling and logging

SUPPORTED QUERY PATTERNS:
• Simple: "status:open"
• Boolean: "status:open AND priority:high"
• Range: "created_at:[now-7d TO now]"
• Custom attributes: "custom_attributes.environment:production"
• Complex: "(status:open OR status:in_progress) AND assignee:user123"

Deployment Architecture

MULTI-ENVIRONMENT DEPLOYMENT:

PRODUCTION ENVIRONMENTS:
• US1 (us-east-1) - Primary US production
• US3 (us-west-2) - Secondary US production
• US5 (us-west-1) - Tertiary US production
• EU1 (eu-west-1) - European production
• AP1 (ap-southeast-1) - Asia Pacific production
• AP2 (ap-northeast-1) - Secondary AP production

STAGING ENVIRONMENT:
• US1-STAGING - Full staging environment with production data subset

SEARCH INDEXER APPLICATIONS:
Multiple case-event-handler deployments per environment:
• case-event-handler-search-indexer-es-v8-us1.yaml
• case-event-handler-search-indexer-es-v8-eu1.yaml
• case-event-handler-search-indexer-es-v8-ap1.yaml
• case-event-handler-analytic-*.yaml (analytics configuration)

ELASTICSEARCH CLUSTERS:
• Environment-specific ES clusters
• Index per environment isolation
• Cluster-level security and access control
• Regional data sovereignty compliance

CONFIGURATION MANAGEMENT:
• Consul-based configuration (consulconfig)
• Environment-specific overrides
• Feature flags and runtime parameters
• Monitoring and alerting integration

KEY APPLICATIONS:

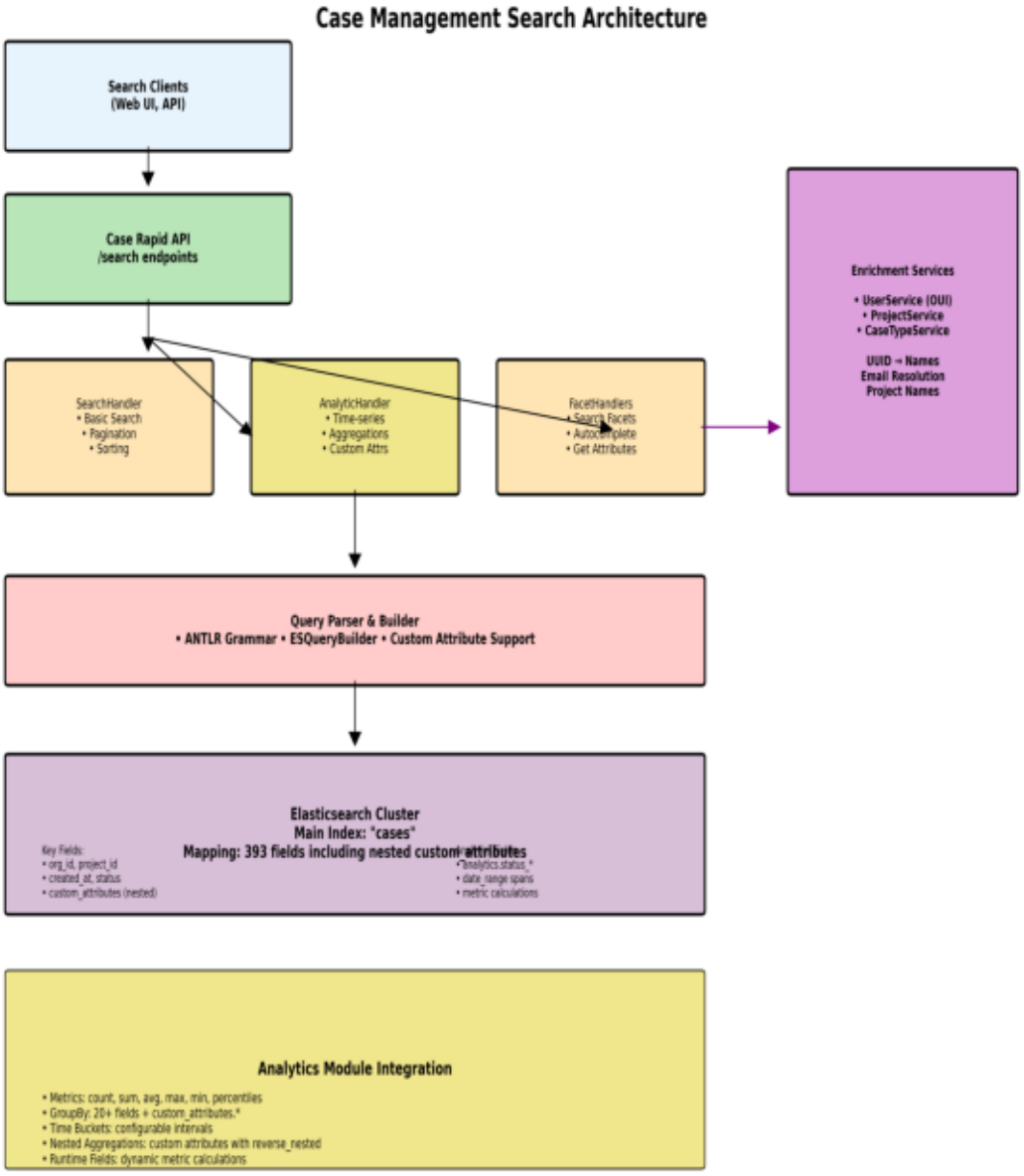
1. CASE-RAPID-API
   • REST API endpoints for search
   • gRPC service interfaces
   • Authentication and rate limiting
   • Request/response transformation

2. CASE-EVENT-HANDLER
   • Real-time index updates
   • Event sourcing integration
   • Search index synchronization
   • Analytics data computation

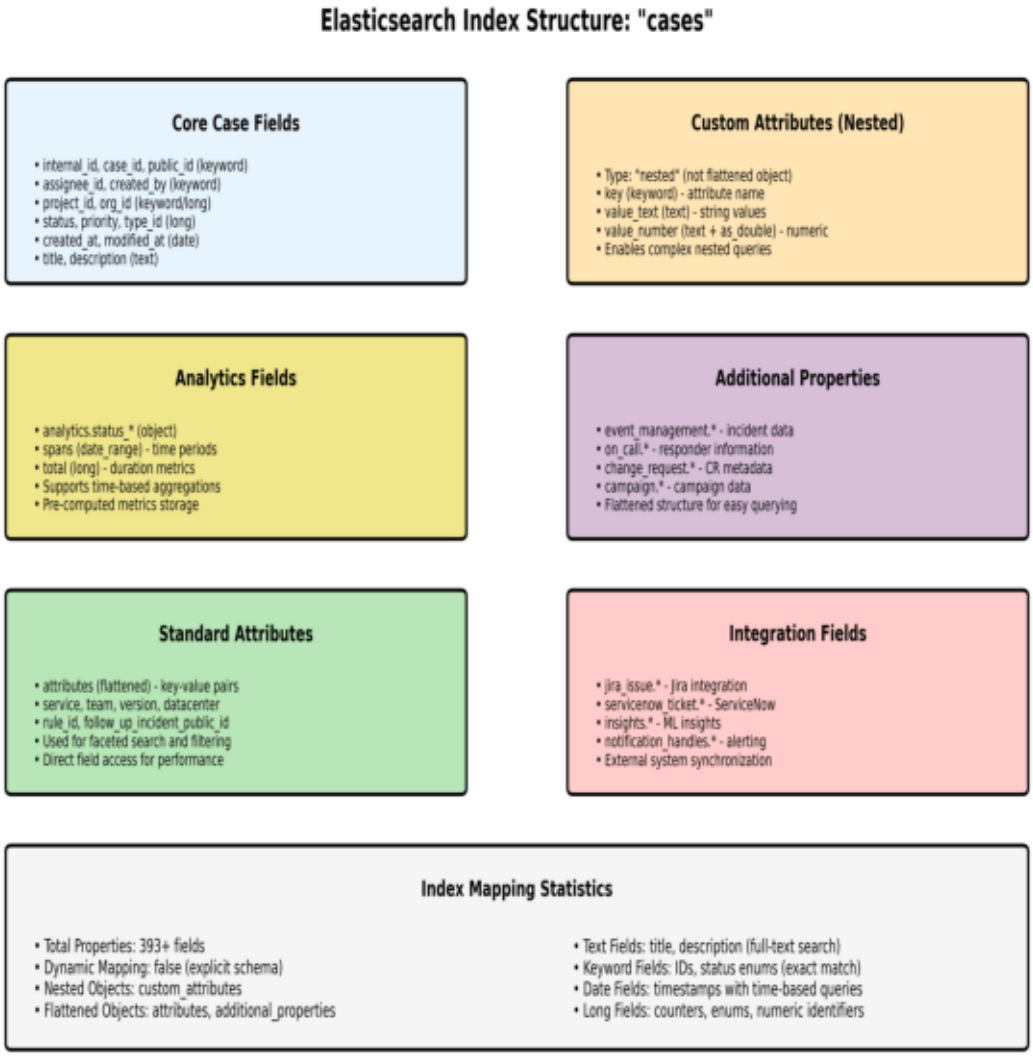
3. SEARCH INDEXER
   • Elasticsearch document management
   • Index lifecycle management
   • Mapping updates and migrations
   • Performance optimization

MONITORING & OBSERVABILITY:
• Structured logging with contextual information
• Metrics collection and dashboards
• Error tracking and alerting
• Performance monitoring and profiling

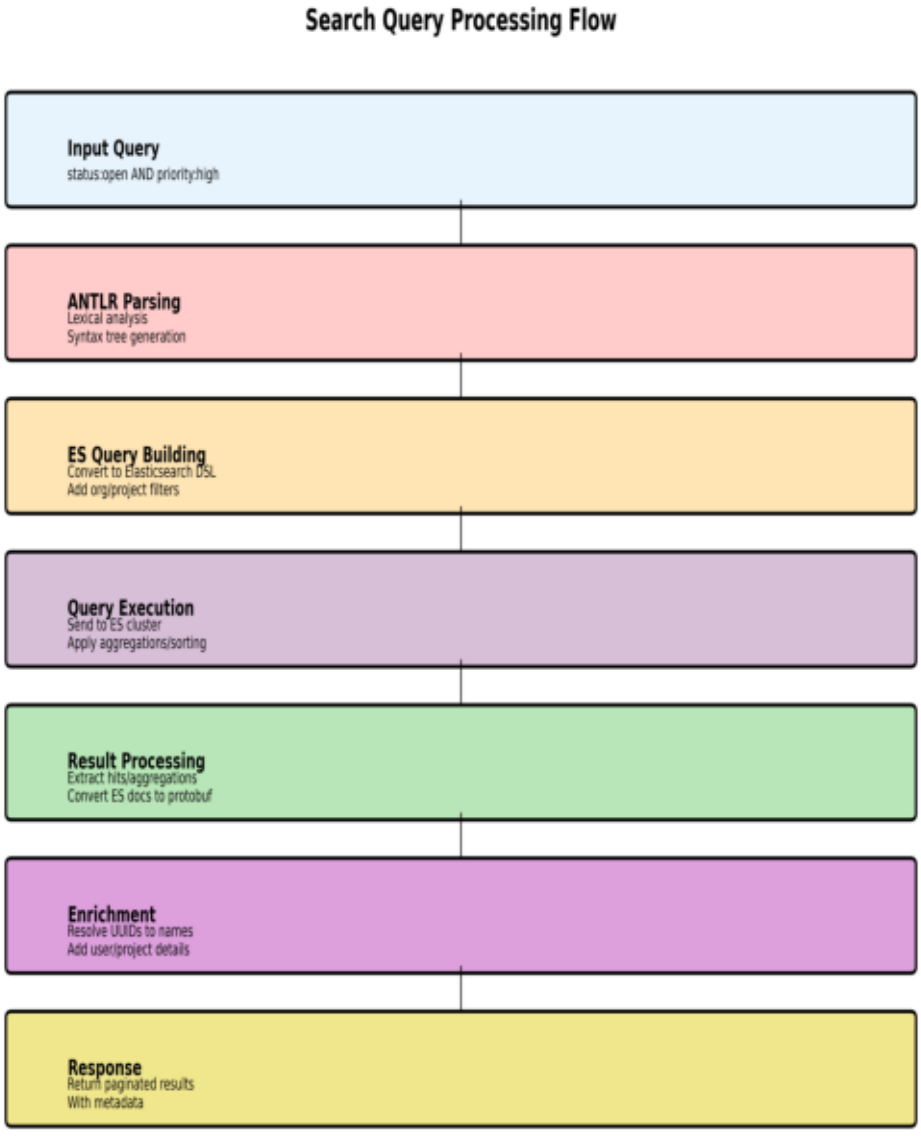
Search Architecture Diagram



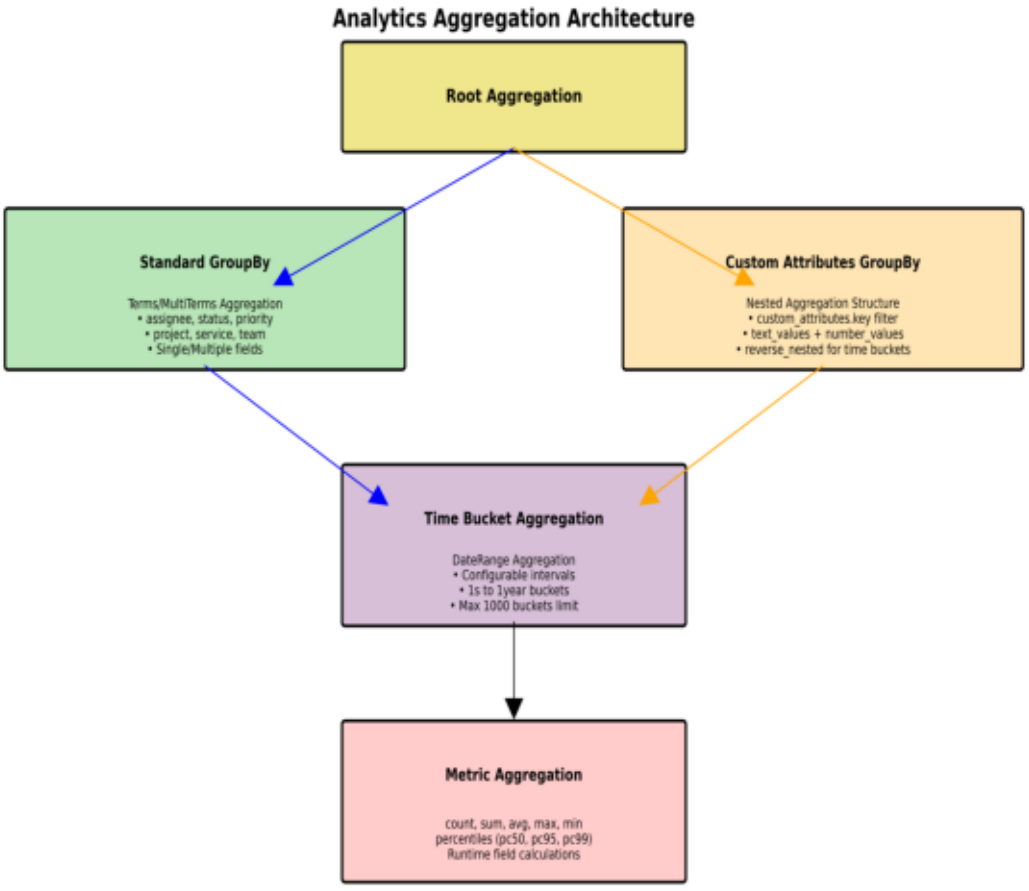
Elasticsearch Index Mapping



Query Processing Flow



Analytics Aggregation Structure



Custom Attribute Aggregation Details

Structure: nested → filter → terms → reverse_nested → date_range → metric

1. Nested: Enter custom_attributes array
2. Filter: Match specific attribute key (e.g., "environment")
3. Terms: Group by text_values.keyword or number_values.as_double
4. Reverse_nested: Exit to parent document for time calculations
5. Date_range: Create time buckets
6. Metric: Apply aggregation method (count, avg, etc.)

Conclusions & Recommendations

STRENGTHS:

 ROBUST ARCHITECTURE
• Clean separation of concerns with specialized handlers
• Comprehensive error handling and logging
• Scalable multi-environment deployment
• Strong typing with protobuf integration

 ADVANCED SEARCH CAPABILITIES
• Complex query parsing with ANTLR grammar
• Nested custom attribute support
• Real-time analytics with flexible aggregations
• Comprehensive faceted search functionality

 PERFORMANCE OPTIMIZATIONS
• Project-based security filtering
• Query builder optimizations
• Configurable timeouts and limits
• LZ4 compression for analytics responses

RECOMMENDATIONS FOR IMPROVEMENT:

 PERFORMANCE ENHANCEMENTS
• Implement query result caching for frequently accessed data
• Add query performance monitoring and slow query alerts
• Consider read replicas for analytics-heavy workloads
• Optimize aggregation queries for large datasets

 SCALABILITY IMPROVEMENTS
• Implement horizontal scaling for search handlers
• Add connection pooling for Elasticsearch clients
• Consider implementing circuit breakers for external services
• Add request queuing and rate limiting

 MONITORING & OBSERVABILITY
• Enhanced metrics collection for query performance
• Distributed tracing for complex query flows
• Custom dashboards for search analytics
• Automated alerting for search failures

 DOCUMENTATION & TOOLING
• Interactive query builder UI for testing
• Comprehensive API documentation with examples
• Performance testing framework
• Schema migration tools for Elasticsearch mappings

FUTURE CONSIDERATIONS:
• Machine learning integration for search ranking
• Support for vector/semantic search capabilities
• Advanced caching strategies (Redis integration)
• Multi-tenancy improvements for better isolation