Elasticsearch Vector Database Design Document

Document Information

Project: Elasticsearch Vector Database on OpenShift

Version: 1.0

Date: September 2025

Owner: Platform Engineering Team

Status: Design Review

1. Executive Summary

This document outlines the design for deploying Elasticsearch as a vector database on OpenShift cluster SILO-06. The system will support hybrid search capabilities combining traditional text search (BM25) with semantic vector search using dense embeddings.

Key Objectives

Enable semantic search across enterprise documents and logs

Provide scalable vector storage and retrieval capabilities

Integrate with existing data sources (NetApp S3, NAS, databases)

Support real-time and batch data ingestion workflows

Maintain high availability and disaster recovery capabilities

2. System Overview

Architecture Principles

Cloud-native: Kubernetes-native deployment using operators

Scalable: Horizontal scaling of compute and storage

Resilient: Multi-node clustering with automated failover

Observable: Comprehensive monitoring and logging

Secure: RBAC, network policies, and encryption at rest/transit

High-Level Components

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ Data Sources │───▶│ ETL Pipeline │───▶│ Elasticsearch │

│ │ │ │ │ Vector DB │

│ • NetApp S3 │ │ • JupyterHub │ │ │

│ • NAS/ICMP │ │ • Batch Jobs │ │ • Text Search │

│ • DB Exports │ │ • Kafka (opt) │ │ • Vector Search │

│ • App Uploads │ │ • Embedding Gen │ │ • Hybrid Query │

└─────────────────┘ └─────────────────┘ └─────────────────┘

│

┌─────────────────┐

│ Consumer Apps │

│ │

│ • Search APIs │

│ • Dashboards │

│ • Notebooks │

└─────────────────┘

3. Requirements

3.1 Functional Requirements

IDRequirementPriorityAcceptance CriteriaFR-001Text SearchHighSupport BM25 full-text search with sub-second responseFR-002Vector SearchHighSupport kNN search with cosine similarityFR-003Hybrid SearchMediumCombine text + vector search with configurable weightsFR-004Bulk IngestionHighProcess 10K+ documents per hourFR-005Real-time IngestionMediumNear real-time indexing (< 30s latency)FR-006Multi-source IntegrationHighSupport S3, NAS, DB, and upload sourcesFR-007Backup & RecoveryHighDaily snapshots with 30-day retention

3.2 Non-Functional Requirements

IDRequirementTargetMeasurementNFR-001Search Latency< 100msp95 response timeNFR-002Indexing Throughput1000 docs/minSustained rateNFR-003Availability99.5%Monthly uptimeNFR-004Data Retention12 monthsHot + cold storageNFR-005Concurrent Users50+Simultaneous searchesNFR-006Storage Growth20% monthlyCapacity planning

3.3 Technical Constraints

Platform: OpenShift 4.12+ on SILO-06 cluster

Namespace: es-poc (isolated environment)

Storage: NetApp (RWX) and SSD (RWO) storage classes

Network: Internal cluster networking with service mesh

Security: RBAC, network policies, Pod Security Standards

4. Detailed Design

4.1 Elasticsearch Cluster Architecture

Node Configuration

POC Phase (Current):

┌─────────────────────────────────┐

│ Single Node │

│ Roles: master + data + ingest │

│ Memory: 4GB (2GB heap) │

│ Storage: 150GB SSD │

│ CPU: 2 cores │

└─────────────────────────────────┘

Production Phase (Target):

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ Master Nodes │ │ Data Nodes │ │ Ingest Nodes │

│ (3 nodes) │ │ (3+ nodes) │ │ (2 nodes) │

│ │ │ │ │ │

│ • 2GB RAM │ │ • 8GB RAM │ │ • 4GB RAM │

│ • 10GB SSD │ │ • 200GB SSD │ │ • 50GB SSD │

│ • 1 CPU │ │ • 4 CPU │ │ • 2 CPU │

└──────────────┘ └──────────────┘ └──────────────┘

Deployment Specification

yamlapiVersion: elasticsearch.k8s.elastic.co/v1

kind: Elasticsearch

metadata:

name: es-vector-poc

namespace: es-poc

spec:

version: 8.11.0

http:

tls:

selfSignedCertificate:

disabled: false

nodeSets:

- name: default

count: 1

config:

node.store.allow\_mmap: false

xpack.security.enabled: true

xpack.monitoring.collection.enabled: true

podTemplate:

spec:

containers:

- name: elasticsearch

resources:

requests:

memory: 4Gi

cpu: 2

limits:

memory: 4Gi

cpu: 4

env:

- name: ES\_JAVA\_OPTS

value: "-Xms2g -Xmx2g"

volumeClaimTemplates:

- metadata:

name: elasticsearch-data

spec:

accessModes: ["ReadWriteOnce"]

storageClassName: ssd-rwo

resources:

requests:

storage: 150Gi

4.2 Index Design

Index Templates

json{

"name": "vector-documents-template",

"index\_patterns": ["vector-docs-\*", "vector-logs-\*"],

"template": {

"settings": {

"number\_of\_shards": 3,

"number\_of\_replicas": 1,

"index.codec": "best\_compression",

"index.refresh\_interval": "30s",

"index.mapping.total\_fields.limit": 2000

},

"mappings": {

"properties": {

"@timestamp": {

"type": "date",

"format": "strict\_date\_optional\_time||epoch\_millis"

},

"content": {

"type": "text",

"analyzer": "standard",

"fields": {

"keyword": {"type": "keyword", "ignore\_above": 256}

}

},

"title": {

"type": "text",

"analyzer": "standard"

},

"embedding": {

"type": "dense\_vector",

"dims": 768,

"index": true,

"similarity": "cosine",

"index\_options": {

"type": "hnsw",

"m": 16,

"ef\_construction": 100

}

},

"source\_system": {

"type": "keyword"

},

"document\_type": {

"type": "keyword"

},

"metadata": {

"type": "object",

"enabled": false

}

}

}

},

"composed\_of": ["logs-mappings", "vector-mappings"],

"priority": 200

}

Index Lifecycle Management

json{

"policy": {

"phases": {

"hot": {

"min\_age": "0ms",

"actions": {

"rollover": {

"max\_size": "10gb",

"max\_age": "7d",

"max\_docs": 1000000

},

"set\_priority": {

"priority": 100

}

}

},

"warm": {

"min\_age": "7d",

"actions": {

"shrink": {

"number\_of\_shards": 1

},

"allocate": {

"number\_of\_replicas": 0

},

"set\_priority": {

"priority": 50

}

}

},

"cold": {

"min\_age": "30d",

"actions": {

"allocate": {

"number\_of\_replicas": 0

},

"set\_priority": {

"priority": 0

}

}

},

"delete": {

"min\_age": "365d",

"actions": {

"delete": {}

}

}

}

}

}

4.3 Data Pipeline Architecture

ETL Processing Flow

Data Sources → Ingestion → Processing → Indexing → Search

│ │ │ │ │

│ │ │ │ │

▼ ▼ ▼ ▼ ▼

┌─────────┐ ┌─────────────┐ ┌────────┐ ┌────────┐ ┌──────┐

│ S3/NAS │ │ JupyterHub │ │Embedding│ │ ES │ │ APIs │

│ DB Dump │ │ Batch Jobs │ │Pipeline │ │Indexing│ │ UIs │

│ Uploads │ │ Kafka (opt) │ │ │ │ │ │ │

└─────────┘ └─────────────┘ └────────┘ └────────┘ └──────┘

JupyterHub Integration

yamlapiVersion: v1

kind: ConfigMap

metadata:

name: jupyter-etl-config

namespace: es-poc

data:

etl-pipeline.py: |

import os

import pandas as pd

from elasticsearch import Elasticsearch, helpers

from sentence\_transformers import SentenceTransformer

# Configuration

ES\_HOST = "es-vector-poc-es-http.es-poc.svc.cluster.local"

ES\_PORT = 9200

MODEL\_NAME = "sentence-transformers/all-mpnet-base-v2"

class DocumentProcessor:

def \_\_init\_\_(self):

self.es = self.\_init\_elasticsearch()

self.model = SentenceTransformer(MODEL\_NAME)

def \_init\_elasticsearch(self):

return Elasticsearch(

hosts=[{'host': ES\_HOST, 'port': ES\_PORT}],

http\_auth=('elastic', os.getenv('ELASTIC\_PASSWORD')),

use\_ssl=True,

verify\_certs=False

)

def process\_batch(self, documents, batch\_size=50):

for i in range(0, len(documents), batch\_size):

batch = documents[i:i + batch\_size]

self.\_process\_documents(batch)

def \_process\_documents(self, docs):

# Generate embeddings

texts = [doc['content'] for doc in docs]

embeddings = self.model.encode(texts)

# Prepare bulk actions

actions = []

for doc, embedding in zip(docs, embeddings):

action = {

"\_index": f"vector-docs-{pd.Timestamp.now().strftime('%Y-%m')}",

"\_source": {

\*\*doc,

"embedding": embedding.tolist(),

"@timestamp": pd.Timestamp.now().isoformat()

}

}

actions.append(action)

# Bulk index

helpers.bulk(self.es, actions, chunk\_size=50)

4.4 Search API Design

Hybrid Search Query Structure

pythondef hybrid\_search(query\_text, query\_vector, text\_weight=0.3, vector\_weight=0.7):

return {

"query": {

"script\_score": {

"query": {

"bool": {

"should": [

{

"match": {

"content": {

"query": query\_text,

"boost": text\_weight

}

}

}

]

}

},

"script": {

"source": """

double textScore = \_score;

double vectorScore = cosineSimilarity(params.query\_vector, 'embedding') + 1.0;

return (textScore \* params.text\_weight) + (vectorScore \* params.vector\_weight);

""",

"params": {

"query\_vector": query\_vector,

"text\_weight": text\_weight,

"vector\_weight": vector\_weight

}

}

}

},

"size": 20,

"\_source": ["title", "content", "source\_system", "@timestamp"],

"highlight": {

"fields": {

"content": {"fragment\_size": 150, "number\_of\_fragments": 3}

}

}

}

REST API Endpoints

yamlSearch API Specification:

Base URL: https://search-api.es-poc.svc.cluster.local

Endpoints:

POST /search/text

- Pure text search using BM25

- Query parameters: q, size, from, filters

POST /search/vector

- Pure vector similarity search

- Body: embedding vector, similarity threshold

POST /search/hybrid

- Combined text + vector search

- Body: query text, weights, filters, size

GET /search/suggest

- Auto-completion and query suggestions

- Query parameters: prefix, size

POST /search/similar

- Find similar documents to given document ID

- Body: document\_id, similarity\_threshold

5. Storage Design

5.1 Storage Classes & Allocation

Storage Requirements

yamlStorage Class Mapping:

ssd-rwo (ReadWriteOnce):

- Elasticsearch data volumes

- High IOPS for search operations

- Fast local SSD storage

netapp-rwx (ReadWriteMany):

- Snapshot repository

- Shared ETL staging area

- Multi-pod access scenarios

Volume Allocation:

POC Phase:

- ES Data: 150GB (ssd-rwo)

- Snapshots: 200GB (netapp-rwx)

- ETL Staging: 50GB (netapp-rwx)

Production Phase:

- ES Data: 600GB total (200GB × 3 nodes, ssd-rwo)

- ES Masters: 30GB total (10GB × 3 nodes, ssd-rwo)

- Snapshots: 1TB (netapp-rwx)

- ETL Staging: 200GB (netapp-rwx)

5.2 Backup & Recovery Strategy

Snapshot Configuration

json{

"repository\_settings": {

"type": "s3",

"settings": {

"bucket": "es-snapshots-silo06",

"region": "us-east-1",

"endpoint": "https://s3.netapp-internal.com",

"path\_style\_access": true,

"compress": true,

"chunk\_size": "1gb"

}

},

"snapshot\_policy": {

"schedule": "0 2 \* \* \*",

"name": "<daily-{now/d}>",

"repository": "s3-backup",

"config": {

"indices": ["vector-docs-\*", "vector-logs-\*"],

"ignore\_unavailable": false,

"include\_global\_state": true

},

"retention": {

"expire\_after": "30d",

"min\_count": 7,

"max\_count": 35

}

}

}

Disaster Recovery Procedures

markdownRecovery Time Objectives (RTO):

- Critical Data: 4 hours

- Full System: 8 hours

- Historical Data: 24 hours

Recovery Point Objectives (RPO):

- Critical Data: 1 hour

- Bulk Data: 24 hours

Recovery Procedures:

1. Automated daily snapshots to S3

2. Cross-region snapshot replication (optional)

3. Cluster rebuild from snapshots

4. Data validation and integrity checks

5. Application connectivity testing

6. Security Design

6.1 Authentication & Authorization

RBAC Configuration

yamlapiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

namespace: es-poc

name: elasticsearch-user

rules:

- apiGroups: [""]

resources: ["secrets"]

resourceNames: ["es-vector-poc-es-elastic-user"]

verbs: ["get"]

- apiGroups: [""]

resources: ["services"]

resourceNames: ["es-vector-poc-es-http"]

verbs: ["get"]

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: elasticsearch-client

namespace: es-poc

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: elasticsearch-binding

namespace: es-poc

subjects:

- kind: ServiceAccount

name: elasticsearch-client

namespace: es-poc

roleRef:

kind: Role

name: elasticsearch-user

apiGroup: rbac.authorization.k8s.io

6.2 Network Security

Network Policies

yamlapiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: elasticsearch-netpol

namespace: es-poc

spec:

podSelector:

matchLabels:

elasticsearch.k8s.elastic.co/cluster-name: es-vector-poc

policyTypes:

- Ingress

- Egress

ingress:

- from:

- namespaceSelector:

matchLabels:

name: es-poc

- podSelector:

matchLabels:

app: jupyter-hub

- podSelector:

matchLabels:

app: search-api

ports:

- protocol: TCP

port: 9200

- protocol: TCP

port: 9300

egress:

- to: []

ports:

- protocol: TCP

port: 53

- protocol: UDP

port: 53

- to:

- namespaceSelector:

matchLabels:

name: es-poc

6.3 Data Encryption

Encryption Configuration

yamlEncryption at Rest:

- Kubernetes secrets for credentials

- Encrypted storage volumes (storage class level)

- Elasticsearch index-level encryption (future)

Encryption in Transit:

- TLS 1.2+ for all ES communications

- Service mesh mTLS (optional)

- HTTPS for external APIs

Secret Management:

- Elasticsearch built-in security

- Kubernetes native secrets

- External secret operator (future)

7. Monitoring & Observability

7.1 Metrics & Alerting

Key Performance Indicators

yamlCluster Health Metrics:

- elasticsearch\_cluster\_health\_status

- elasticsearch\_cluster\_health\_number\_of\_nodes

- elasticsearch\_cluster\_health\_active\_shards

- elasticsearch\_cluster\_health\_unassigned\_shards

Performance Metrics:

- elasticsearch\_indices\_search\_query\_time\_seconds

- elasticsearch\_indices\_indexing\_index\_time\_seconds

- elasticsearch\_jvm\_memory\_used\_bytes

- elasticsearch\_filesystem\_data\_available\_bytes

Vector Search Metrics:

- Custom: vector\_search\_latency\_seconds

- Custom: embedding\_generation\_duration\_seconds

- Custom: hybrid\_search\_accuracy\_ratio

Business Metrics:

- Custom: documents\_processed\_total

- Custom: search\_requests\_per\_second

- Custom: user\_session\_duration\_seconds

Alerting Rules

yamlgroups:

- name: elasticsearch.rules

rules:

- alert: ElasticsearchClusterRed

expr: elasticsearch\_cluster\_health\_status{color="red"} == 1

for: 5m

labels:

severity: critical

annotations:

summary: "Elasticsearch cluster is in red state"

- alert: ElasticsearchHighSearchLatency

expr: elasticsearch\_indices\_search\_query\_time\_seconds > 1

for: 2m

labels:

severity: warning

annotations:

summary: "Search latency is above 1 second"

- alert: ElasticsearchLowDiskSpace

expr: elasticsearch\_filesystem\_data\_available\_bytes < 10737418240 # 10GB

for: 5m

labels:

severity: critical

annotations:

summary: "Elasticsearch node has less than 10GB disk space"

7.2 Logging Strategy

Log Configuration

yamlElasticsearch Logs:

- Level: INFO (production), DEBUG (development)

- Format: JSON structured logging

- Retention: 30 days

- Location: /usr/share/elasticsearch/logs/

Application Logs:

- ETL Pipeline: Structured logs with correlation IDs

- Search API: Request/response logging with timing

- JupyterHub: Session and job execution logs

Log Aggregation:

- Target: Existing cluster logging infrastructure

- Format: OpenShift logging standards

- Indexing: Separate indices for different log types

8. Performance Considerations

8.1 Capacity Planning

Sizing Guidelines

yamlDocument Volume Projections:

Month 1-3: 500K documents, 50GB storage

Month 4-6: 1.5M documents, 150GB storage

Month 7-12: 5M documents, 500GB storage

Year 2: 20M documents, 2TB storage

Search Load Projections:

Concurrent Users: 10 (POC) → 50 (production)

Queries per Second: 5 (POC) → 100 (production)

Peak Load Multiplier: 3x average load

Vector Processing:

Embedding Dimension: 768

Storage Overhead: ~3KB per document

Search Memory: 2-4GB per billion vectors

8.2 Performance Optimization

Query Optimization

json{

"search\_optimization": {

"index\_settings": {

"refresh\_interval": "30s",

"number\_of\_replicas": 1,

"codec": "best\_compression"

},

"query\_cache": {

"enabled": true,

"size": "20%"

},

"fielddata\_cache": {

"size": "40%"

},

"vector\_search": {

"ef\_search": 100,

"pre\_filter": true,

"rescore": {

"window\_size": 100,

"query\_weight": 0.7,

"rescore\_query\_weight": 1.3

}

}

}

}

JVM Tuning

yamlES\_JAVA\_OPTS:

Heap Size: "50% of available memory (max 32GB)"

GC Algorithm: "G1GC for balanced latency/throughput"

Memory Settings:

- "-Xms4g -Xmx4g" # Fixed heap size

- "-XX:+UseG1GC" # G1 garbage collector

- "-XX:MaxGCPauseMillis=200" # GC pause target

- "-XX:+UnlockExperimentalVMOptions"

- "-XX:+UseStringDeduplication" # Reduce memory usage

9. Testing Strategy

9.1 Test Plan

Unit Testing

Elasticsearch configuration validation

Index template and mapping tests

Query performance benchmarks

Embedding generation accuracy

Integration Testing

End-to-end data pipeline testing

Multi-source ingestion validation

Search API functionality testing

Backup and restore procedures

Performance Testing

Load testing with synthetic data

Concurrent user simulation

Search latency under load

Indexing throughput testing

Security Testing

RBAC permission validation

Network policy enforcement

TLS certificate validation

Data encryption verification

9.2 Test Data & Scenarios

Test Datasets

yamlSynthetic Data:

- 100K sample documents (various sizes)

- Multiple content types (PDF, text, JSON)

- Different languages and encodings

- Vector embeddings pre-generated

Real Data Samples:

- Anonymized production documents

- Log samples from existing systems

- Database export samples

- User upload simulations

Load Testing Scenarios:

- Baseline: 10 concurrent users, 1 QPS

- Normal Load: 25 concurrent users, 10 QPS

- Peak Load: 50 concurrent users, 50 QPS

- Stress Test: 100 concurrent users, 100 QPS

10. Deployment Plan

10.1 Deployment Phases

Phase 1: POC Deployment (Week 1-2)

yamlObjectives:

- Single-node Elasticsearch cluster

- Basic search functionality

- Sample data ingestion

- Performance baseline

Deliverables:

- Deployed ES cluster on SILO-06

- Basic monitoring setup

- Sample search API

- Initial performance metrics

Success Criteria:

- Cluster health: Green

- Search latency: < 500ms

- Index 10K test documents

- Basic vector search working

Phase 2: Production Preparation (Week 3-4)

yamlObjectives:

- Multi-node cluster setup

- Complete monitoring implementation

- Security hardening

- Backup procedures

Deliverables:

- 3-node production cluster

- Full observability stack

- Network policies and RBAC

- Automated backup/restore

Success Criteria:

- High availability verified

- Complete monitoring coverage

- Security compliance passed

- Disaster recovery tested

Phase 3: Production Deployment (Week 5-6)

yamlObjectives:

- Production workload migration

- Performance optimization

- User onboarding

- Documentation completion

Deliverables:

- Production data migration

- Optimized search performance

- User training materials

- Operational runbooks

Success Criteria:

- All SLAs met

- User acceptance achieved

- Operational handoff complete

- Documentation approved

10.2 Rollback Strategy

Rollback Triggers

Cluster health degradation

Performance SLA violations

Data corruption detected

Security breach identified

Rollback Procedures

Immediate: Traffic diversion to backup system

Short-term: Restore from latest snapshot

Long-term: Rebuild cluster from known-good configuration

Data recovery: Point-in-time restore procedures

11. Risk Assessment

11.1 Technical Risks

RiskProbabilityImpactMitigationPerformance degradation under loadMediumHighLoad testing, capacity planning, monitoringData loss during migrationLowCriticalMultiple backups, staged migration, validationVector search accuracy issuesMediumMediumModel evaluation, hybrid approach, tuningStorage capacity exhaustionHighHighMonitoring, ILM policies, alertingSecurity vulnerabilityLowCriticalSecurity scanning, regular updates, RBAC

11.2 Operational Risks

RiskProbabilityImpactMitigationInsufficient team expertiseMediumMediumTraining, documentation, vendor supportInadequate monitoringMediumHighComprehensive observability, alertingBackup failureLowCriticalMultiple backup methods, regular testingResource contentionMediumMediumResource limits, QoS policiesCompliance violationsLowHighSecurity reviews, audit trail

12. Success Criteria

12.1 Technical Success Metrics

Performance Targets

Search latency: p95 < 100ms

Indexing throughput: > 1000 docs/min

Cluster availability: > 99.5%

Storage efficiency: < 50% waste

Functional Targets

Hybrid search accuracy: > 85% relevance

Data ingestion success rate: > 99%

API response success rate: > 99.9%

Backup success rate: 100%

12.2 Business Success Metrics

User Adoption

Active users: 50+ within first month

Query volume: 1000+ searches/day

User satisfaction: > 4.0/5.0 rating

Time-to-insight: 50% reduction

Operational Efficiency

Reduced manual search time: 60%

Improved content discovery: 40%

Faster incident resolution: 30%

Knowledge sharing increase: 25%

13. Maintenance & Operations

13.1 Operational Procedures

Daily Operations

Cluster health monitoring

Performance metric review

Alert investigation and resolution

Backup verification

Weekly Operations

Capacity planning review

Security patch assessment

Performance trend analysis

User feedback collection

Monthly Operations

Disaster recovery testing

Security audit review

Capacity expansion planning

Documentation updates

13.2 Support & Escalation

Support Tiers

yamlTier 1 (Platform Team):

- Basic monitoring and alerting

- Routine maintenance tasks

- Initial troubleshooting

- User support requests

Tier 2 (Engineering Team):

- Complex performance issues

- Configuration changes

- Integration problems

- Capacity planning

Tier 3 (Vendor Support):

- Critical system failures

- Elasticsearch bugs

- Advanced performance tuning

- Architecture consultation

Escalation Matrix

P0 (Critical): Immediate response, 24/7 escalation

P1 (High): 2-hour response, business hours escalation

P2 (Medium): 8-hour response, next business day

P3 (Low): 24-hour response, planned maintenance window

Document Control

VersionDateAuthorChanges1.0Sep 2025Platform TeamInitial design document