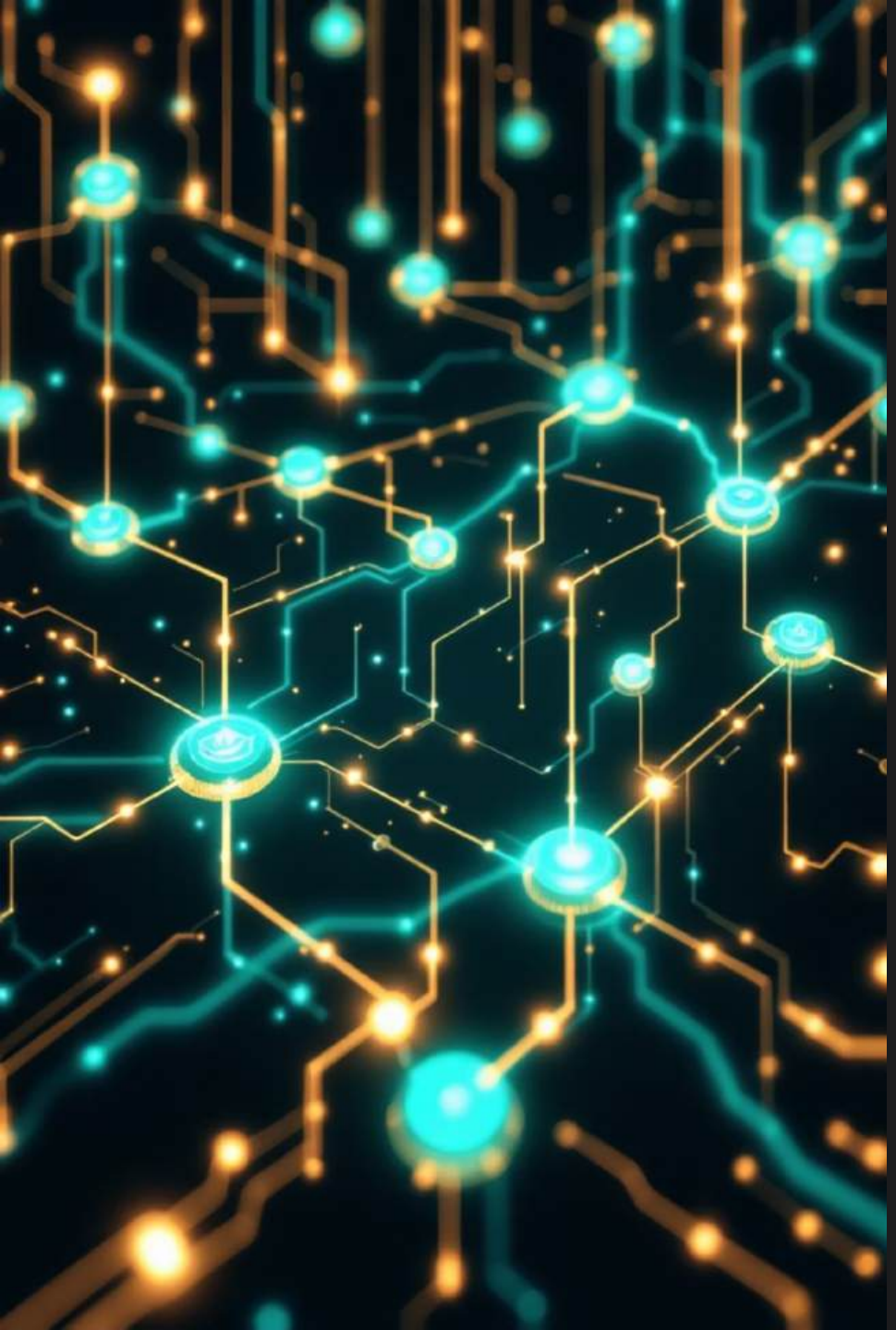


From Logs to LLMs

Building AI-Powered Search & RAG Systems with Elastic

How to turn your most ignored data source into the brain of your AI system

Saurin Patel · Data Scientist

LLMs: Reasoning Engines, Not Memory Systems

Understanding LLM Capabilities and Limitations

Core Functionality

LLMs excel at pattern recognition, language generation, and synthesizing information from their training data.

Key Limitations

Understanding what LLMs cannot do.

No Real-Time State Access

LLMs cannot directly access or interact with live system states, logs, or current environmental data.

Lack of Structured Retrieval

Not designed for deterministic, precise data fetching; they are not databases.

Prone to Hallucination

Can generate plausible but inaccurate information if not grounded with external, factual data.

LLMs are powerful reasoning engines, but their effectiveness is enhanced when integrated with external, factual data sources to mitigate limitations like hallucinations and lack of real-time access.

THE FOUNDATION

What Is RAG?

> "An LLM without your data is a brilliant doctor who has never seen your patient's chart."

RAG — Retrieval-Augmented Generation — solves this. Before answering, the LLM retrieves the most relevant real context from your own data. It reasons from evidence, not memory.

❌ Without RAG

User asks: "Why is payment-service failing?"

- LLM knows: everything from the internet up to training cutoff
- LLM doesn't know: YOUR logs, YOUR infra, TODAY's incident
- LLM answers: a confident-sounding guess — possibly completely wrong

✅ With RAG

User asks: "Why is payment-service failing?"

- Elastic finds: 12 most relevant log lines + your runbook
- LLM now knows: exact errors, timestamps, full root cause chain
- LLM answers: "db-primary-01 CPU spike at 3:47am — connection pool exhausted."

RAG doesn't make the LLM smarter. It gives the LLM the right evidence to reason from.

What RAG Actually Is: System Flow

Retrieval + Controlled Generation

Convert the user's question into a numerical representation.

Find relevant information using a combination of search methods.

Order the retrieved information by relevance.

Add the relevant retrieved information to the original query.

Create a coherent answer based on the augmented query.

WHY IT'S HARD TODAY

The Log Problem

Your systems write millions of lines of truth every second. You're reading almost none of it — not because engineers don't care, but because the tools make it nearly impossible.

📦 Fragile Pipelines

Hand-crafted Logstash configs break on every schema change. You spend Friday nights fixing ingest, not analysing data.

🔎 Grep at Petabyte Scale

`grep -r ERROR /logs` returns 400,000 lines. You open 6 terminals. You still can't find the root cause.

🏝️ Siloed Signals

Logs here. Metrics there. Traces somewhere else. Correlating them requires copy-paste and prayer.

💸 Cost vs. Retention Tradeoff

Can't keep logs beyond 7 days. That P2 incident? 10 days ago. The evidence is gone.

🤖 Zero Intelligence

Your log tool stores. It doesn't think. Every anomaly, every pattern — you find it manually.

Result: less than 1% of logs are ever analysed. The rest is invisible — until something breaks.


A smaller model with great retrieval beats a large model with bad retrieval.

What Real Production Systems Require

Hybrid Ranking (BM25 + Vector)

Combines keyword and semantic search for optimal relevance.

Cross-Index Correlation (ES|QL)

Enables queries and analysis across diverse data indices.

Time-Series Analysis

Understand trends, anomalies, and performance over time.

Aggregations & Deterministic Filters

Summarize data and ensure consistent, predictable results.

Unified Data Sources

Integrate disparate data types (logs, metrics, documents) fo comprehensive insights.

Why Elastic Is Uniquely Positioned

- **Hybrid Search**
  Combines BM25 with Vector Similarity for comprehensive results.
- **ES|QL**
  Enables cross-index joins, aggregations, and structured querying.
- **Observability Native**
  Integrates logs, metrics, and traces within a single platform.
- **Scalability**
  Features approximate kNN, a distributed architecture, and high reliability.

AI Incident Assistant Architecture

Leveraging Logs and RAG for Enhanced Incident Response

Collect and process relevant operational data from various sources.

Segment runbooks and convert them into vector embeddings for semantic understanding.

Store processed data in a searchable index supporting both semantic and keyword searches.

Combine keyword and semantic search to retrieve the most relevant information based on the user's query.

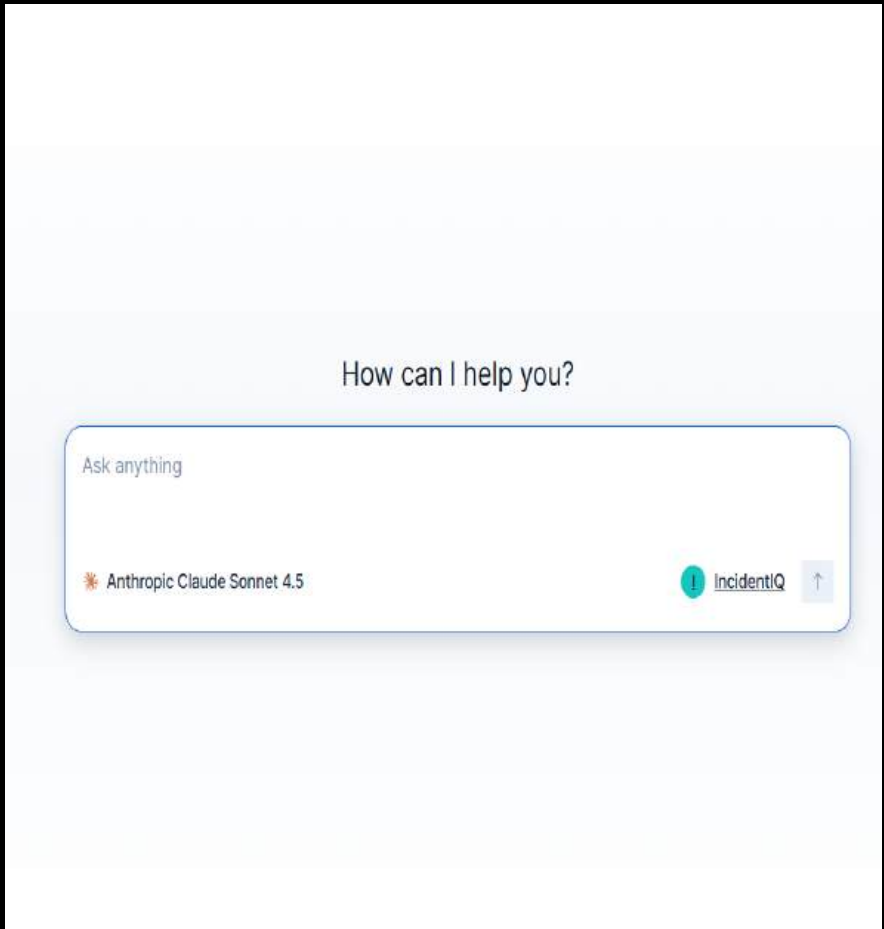
Utilize an LLM to analyze retrieved evidence and generate informed responses or actions for incident resolution.

This architecture streamlines incident response by intelligently combining operational data with knowledge from runbooks.

How it Works

Meet the AI Incident Assistant

Chat Interface Mockup



- User Question: 'Why is payment service throwing DB_CONN_TIMEOUT errors lately?'
- AI Response: Summarizing relevant logs and providing actionable insights.

Core Process

Ingest Logs: Automatically collects and processes incident-related data.

Embed Runbooks: Integrates existing operational knowledge and runbooks.

Query + Respond: Leverages AI to query logs and respond to user inquiries.

Hybrid Retrieval Example (Concrete)

Combining Search Techniques for Enhanced Relevance

Core Query Types

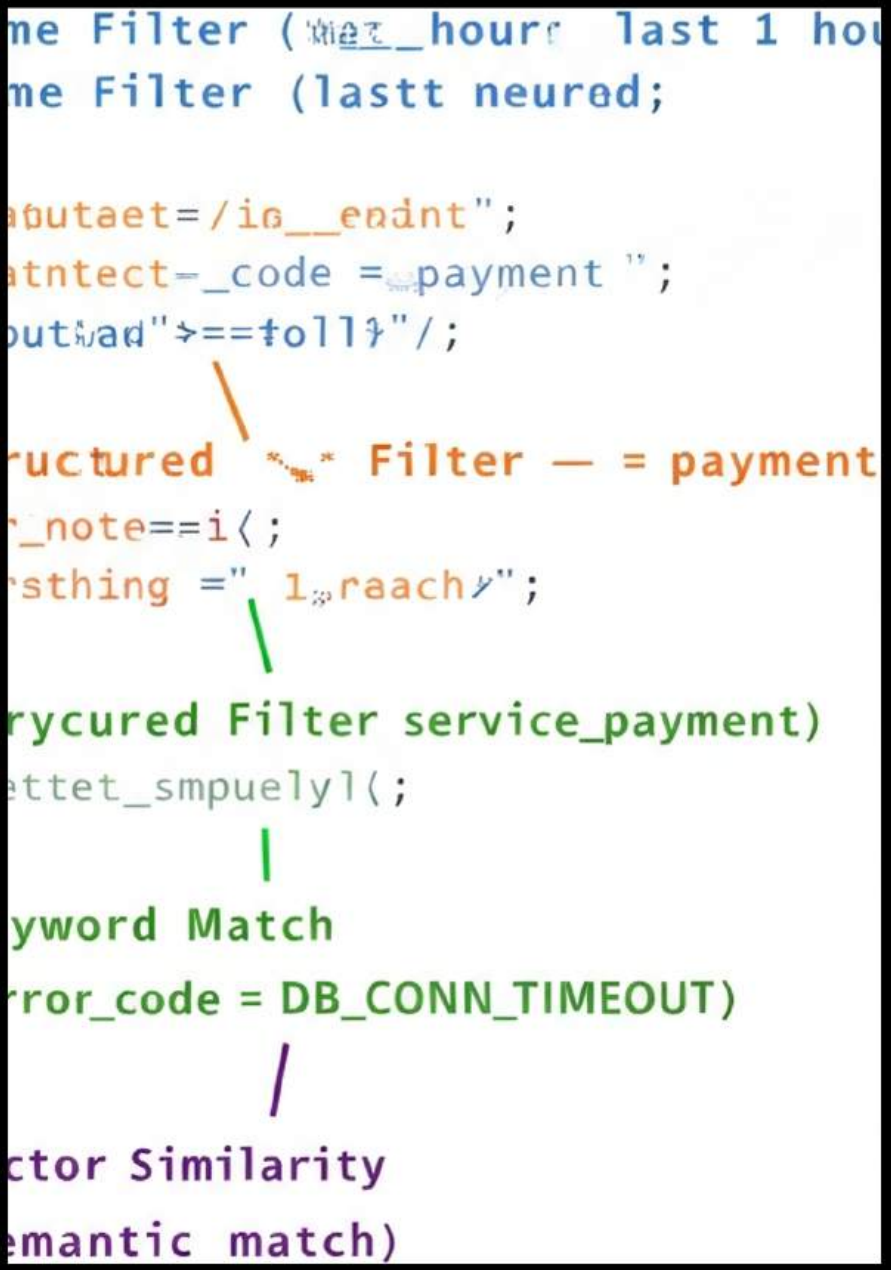
Combines keyword-based (match) and vector-based (knn) searches for exact terms and semantic similarity.

Filtering

Applies structured filters (e.g., `service: payment`) and time range filters (e.g., `last_1_hour`) to refine results.

Relevance Tuning

Utilizes boost weighting to adjust the importance of different query components and prioritize specific matches.



Objective

One Query, Four Dimensions of Retrieval

To illustrate how a single, complex search query is dissected into multiple, actionable dimensions to achieve highly relevant results.

Restore Up

Query Breakdown:

- Time Filter: timestamp: [now-1h TO now]
- Structured Filter: service: "payment"
- Keyword Match: error_code: "DB_CONN_TIMEOUT"
- Vector Similarity: semantic_match("database connection issues")

Achieving precision through multi-dimensional query analysis.

Demo Architecture

Data Flow and Key Components

- **Application Data Flow**
  Application logs are sent to Elastic for retrieval and ranking.
- **Runbooks Data Flow**
  Runbook content is converted into embeddings and stored in Elastic.
- **User Query Processing**
  User queries utilize hybrid retrieval methods before being processed by the LLM.
- **Elastic - Retrieval & Ranking Layer**
  Elasticsearch serves as the core for retrieving and ranking Information.
- **LLM - Reasoning Layer**
  The Large Language Model acts as the reasoning engine for generated responses.

Production RAG Challenges

Often-Overlooked Considerations for Deployment

1

Token Cost Control

Manage unforeseen high token consumption through prompt optimization, efficient data chunking, and retrieval refinement to control operational expenses.

2

Embedding Drift

Combat degradation in embedding accuracy over time by implementing proactive monitoring and periodic re-indexing to maintain retrieval quality.

3

Relevance Tuning

Achieve high semantic relevance beyond basic similarity matching with iterative tuning, re-ranking, and advanced retrieval techniques for pertinent results.

4

Evaluation & Monitoring

Establish robust, continuous evaluation frameworks tracking RAG-specific metrics to quickly identify and address performance degradation.

5

Index Lifecycle Management

Plan and execute strategies for data updates, sunsetting old information, schema evolution, and performance scaling for ongoing vector index management.

What Actually Powered That Answer?

Analyzing RAG Success Factors

- **Structured Filtering Effectiveness**
  Precision in identifying and selecting relevant source material.
- **Hybrid Ranking Accuracy**
  Sophisticated methods to prioritize the most pertinent retrieved information.
- **Time Correlation Relevance**
  Ensuring the recency and temporal alignment of information with the query.
- **Context Injection Quality**
  Seamlessly integrating retrieved data into the model's understanding.
- **Grounded Generation Reliability**
  Ensuring answers are factually accurate and directly supported by retrieved context.

The Bigger Vision: From Search to Agents

Evolution of AI Systems and Elastic's Role

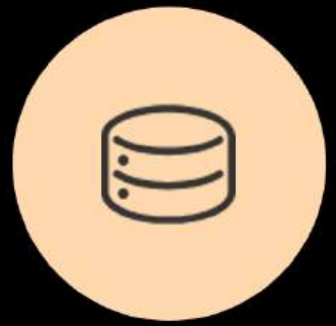
Foundational information retrieval.

Grounding generative AI with specific data for accuracy.

Enabling complex goal understanding, planning, and execution.

Provides persistent memory for context and learning, manages state for complex interactions, and enables intelligent, autonomous action.

Key Takeaways



LLMs need grounding — RAG gives them eyes into your data.

Retrieval-Augmented Generation (RAG) connects Large Language Models (LLMs) to your data, enabling them to provide more accurate and contextually relevant responses.

Elasticsearch is a complete retrieval engine.

Elasticsearch provides a robust and scalable solution for efficiently searching and retrieving information.

Better retrieval = better AI.

The quality of your AI's output is directly dependent on the quality and relevance of the data it can access and retrieve.

Enhancing data retrieval capabilities is crucial for improving the performance and accuracy of AI systems.

THANK YOU