Elastic's AI Year in Review

January to December 2025

One million messages later: Lessons from building AI assistants on the Elasticsearch Platform

Published

May 14, 2026

Authors

Chris Blaisure and Riya Juneja

Acknowledgments

Steve Mayzak, Cory Mangini, Dominik Toepfer, Sandra Leonard, and the entire Field Technology team who build and maintain these tools every day. Thank you.

elastic.co | © 2026 elasticsearch B.V. All rights reserved.

Table of contents

 | © 2026 elasticsearch B.V. All rights reserved.

AI on Elastic: What we built and why it matters

In 2024, Elastic's Field Technology team set out to use generative AI to meaningfully improve the support experience for our customers and employees. We didn't want to summarize a doc or surface a link. We wanted a system that could reason through a problem and provide a useful, accurate answer the moment a user needs it.

The result was the **Elastic Support Assistant**, a retrieval augmented generation (RAG)-powered chat experience built entirely on Elastic's own technology stack, which launched publicly in September 2024. To power it, we:

1. **Indexed the entirety of Elastic's documentation:** 114 major and minor product versions, the full product knowledge base, technical blogs, and onboarding guides into Elasticsearch
2. **Built a hybrid search retrieval layer:** Combining keyword-based search (BM25) with ELSER, Elastic's sparse vector retrieval model, running on Elasticsearch as the underlying vectorDB and search platform
3. **Developed a custom UI:** An EUI-powered chat interface accessible through the Support Hub, Elastic's customer-facing support portal

We weren't building for customers alone. The same assistant was deployed internally for Elastic employees (Elasticians) through a separate deployment wired to internal tooling, Salesforce, and an expanded knowledge base. This architecture further expanded into an **AI Sales Assistant** for revenue teams, a **Case Summarizer** for support agents, and a **Knowledge Drafter** for documentation authors. Each tool shares the same underlying Elastic infrastructure but is scoped to a specific workflow and user population to accelerate efficiency.

 | © 2026 elasticsearch B.V. All rights reserved.

At the core of our architecture is a centralized AI gateway that evaluates all large language model (LLM) traffic for sentiment, answer quality, and technical accuracy. By indexing this enriched data back into Elasticsearch, we defined and created a process we call Context Performance Monitoring or simply **Context Observability**. Using an LLM, we transform raw interaction logs into use case patterns, user engagement paths, and response accuracy metrics. By analyzing these interactions, we can now rapidly augment solutions to align with real-world user needs.

By the end of 2025, we had 955,020 messages (a combination of user questions and AI responses), 209,220 threads (systematic grouping of messages into topics), and 149,432 unique user sessions. Threads were enriched with structured quality metadata: a dataset we could analyze using the same aggregation and search tools we used for everything else. This inaugural report, **Elastic's AI Year in Review**, is built entirely from those insights. It is intended to share our learnings and observations and provide specific guidance to other teams building conversational AI experiences on Elasticsearch.

We are able to perform the complex, multidimensional analysis presented in this report because the entire ecosystem was built on a shared infrastructure. This architecture choice enables us to maintain an integrated data and telemetry layer, allowing us to close the loop between user intent and system response. Building on a shared infrastructure did more than just simplify deployment; it turned our data into a strategic asset. For a more detailed understanding of our architecture, refer to the Appendix.

 | © 2026 elasticsearch B.V. All rights reserved.

Patterns from AI adoption

Overall adoption distribution

Throughout 2025, our AI tools handled 209,220 conversation threads spanning five tools, multiple user populations, and dozens of distinct use cases. Before diving into usage and performance, it is worth understanding the distribution of activity and dynamics of adoption itself, as it directly dictates how we optimized and prioritized workloads.

The first major takeaway from the data: AI tool adoption does not distribute evenly. It concentrates.

By the end of 2025, 7.8% of users (675 people) had generated 79.2% of all sessions with an average of 174 sessions a year or 245 messages. This reveals a power-law distribution, a pattern where a small number of actors account for a disproportionately large share of activity. It clearly highlights a significant divide between "casual users" and "power users" who have fully integrated AI into their daily workflows.

209K	7.8%	9.45
conversation threads logged	**of users drove 79.2% of volume**	**global avg. quality score (out of 10)**

 | © 2026 elasticsearch B.V. All rights reserved.

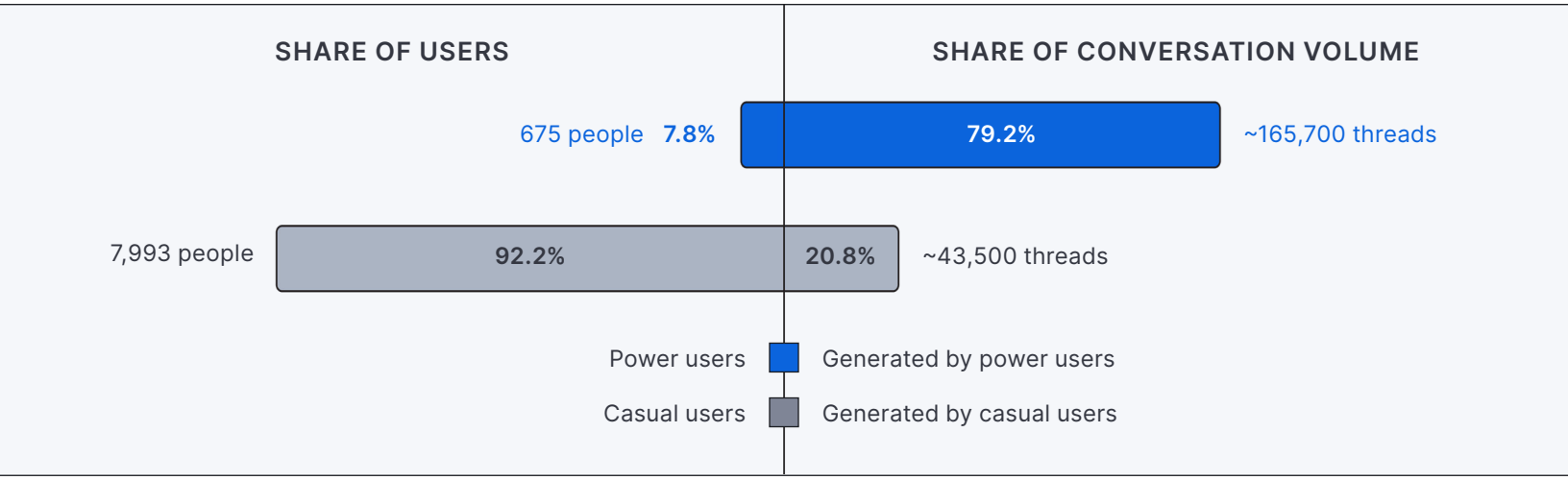


*Fig. 1 — **Usage breakdowns by "casual users" and "power users."** Casual users represent a large proportion of the user population and log only a handful of sessions per year. Power users represent the larger population who log on consistently and drive the majority of search volume.*

Those 675 "power users" averaged roughly 245 searches per year, nearly one interaction per business day. They are not occasional visitors consulting a knowledge base; instead, they have integrated the assistant into their daily workflow in the same way they use Slack or a terminal. The rest of the user population averaged fewer than nine searches for the entire year.

> **Power users drive the majority of volume: Understanding who they are and what they need is a significant input to your feature roadmap.**

What sits on the other end of this distribution is worth examining carefully: 52.9% of users logged only a single session across the entire year, and 62.2% were active in only one calendar month. This is not a failure signal. Many of these users may have had a single, specific problem, found their answer, and moved on. That is a legitimate and successful use of the tool. To better understand this, we are correlating session data with case outcomes and ticket deflection rates to see how many of those single-session users left satisfied.

 | © 2026 elasticsearch B.V. All rights reserved.

Internal vs. external: 2 different tools, 2 different jobs

Our dataset is 69.4% internal users (Elastic employees) and 30.6% external (customers). While they share the same infrastructure, it is important to treat these cohorts as fundamentally distinct. Internal and external users are using the same tooling but for different purposes. The use cases:

1. **External customers (reactive):** External customers access the Support Assistant through the Support Hub, directly alongside the form they use to open a support ticket. That placement generally defines the use case; customers reach for it reactively. A customer hits a problem, checks the assistant before filing a ticket, gets an answer or doesn't, and moves on. The expectation is not daily engagement but on-demand, problem-specific access. Measuring that population by session frequency and finding it lower than internal users is not a retention story. It is an accurate reflection of the product's role in that context. The **right success metric for the external deployment is deflection**: Did the user get an answer good enough that they did not need to open a ticket?

2. **Internal support engineers (proactive):** Internal support engineers use the same assistant daily as a live research tool, pulling it up during case review to cross-reference documentation, troubleshoot configurations, and synthesize answers across multiple product versions simultaneously. That is a fundamentally different interaction model than a customer checking it once before filing a ticket. **High session frequency is the expected result of a tool embedded in a daily workflow.**

For practical and technical purposes, we at Elastic instrumented our deployments internally first, drove internal adoption, and gained valuable feedback from our employees as testers. Since our AI applications serve different user bases, there is a delta between internal and external usage in this report. Deploying internally first was the right decision because the problems our internal users surfaced were problems customers never saw in production.

 | © 2026 elasticsearch B.V. All rights reserved.

Types of use cases across all tools

To understand how users are using these tools and what they need, we need to analyze the conversations. The visualization below is the raw output of the classification pipeline, which we call unconstrained labels. Every conversation thread is evaluated by an LLM that assigns a free-form use case description with no predefined taxonomy. This provides a granular, unfiltered picture of demand from our use case, which then can be consolidated into categories.

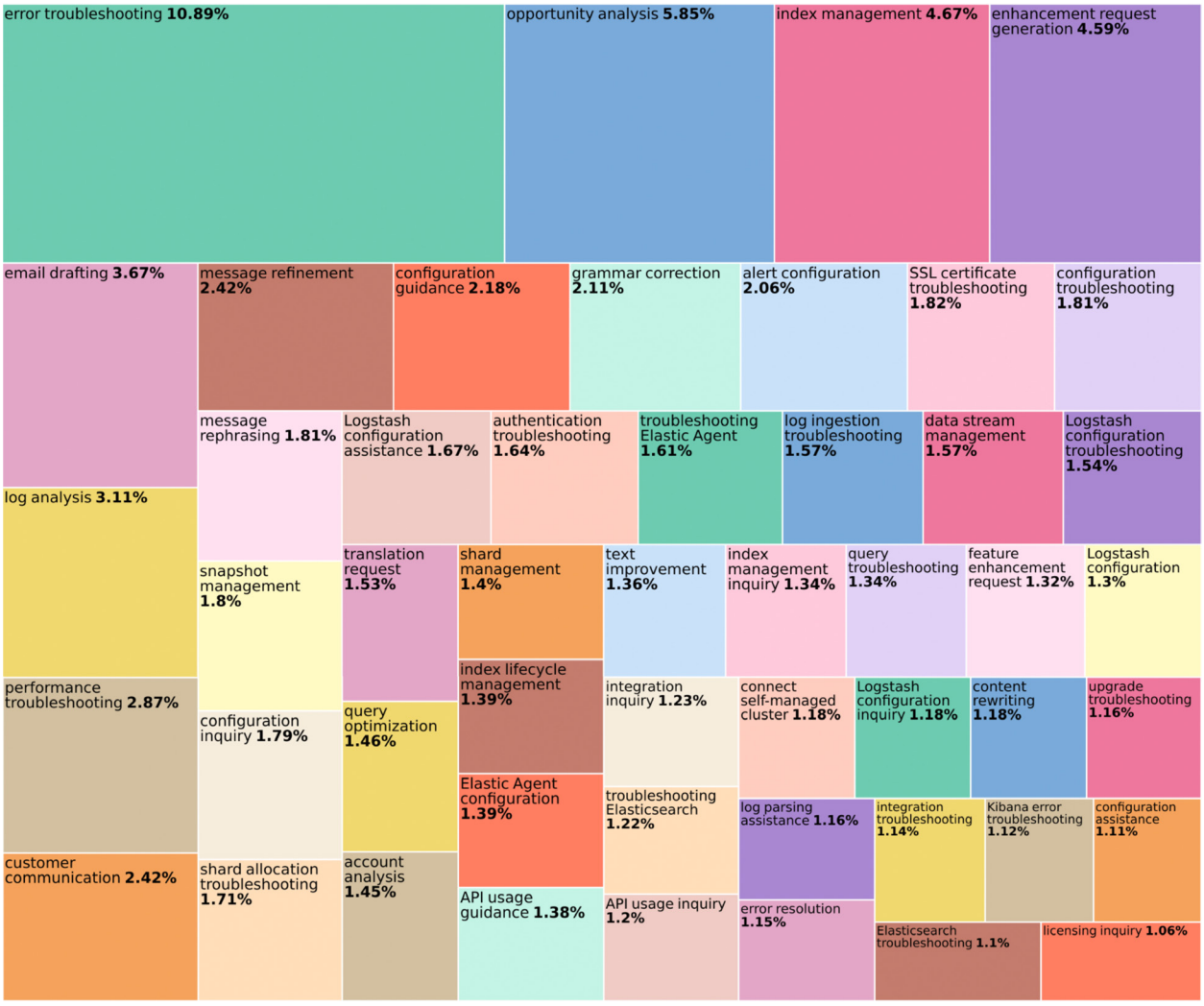


***Fig. 2 — Top 50 unconstrained use case classifications**, visualized directly from Kibana. Each tile represents a distinct AI-generated label; tile size is proportional to thread volume. These labels are the raw output of the LLM classification step before rollup into constrained categories for dashboards and trend tracking. Error troubleshooting leads at 10.89%, but the distribution flattens quickly. The top 5 categories account for roughly 30% of volume, and the remaining 45 tiles share the rest in a long, even tail.*

 | © 2026 elasticsearch B.V. All rights reserved.

Error troubleshooting leads at 10.89%, nearly double the second-place category, but the more telling signal is how quickly the distribution flattens after that. The top 5 categories account for roughly 30% of volume, and the remaining 45 divide the rest in a long, even tail where no single use case dominates. That shape is an accurate reflection of how people actually use general-purpose assistants, meaning a wide variety of specific, well-formed questions that do not reduce neatly to a short list of use cases. When all technical troubleshooting-adjacent categories are combined, 118,955 sessions involve some form of diagnostic task.

> **Users reach for AI when they are stuck: Earning trust at that moment of frustration is one of the fastest paths to adoption.**

Looking across all technical interaction use cases, a clear ratio emerges. For roughly every one failure report (a user saying "it's broken"), the assistants received three or more implementation questions: "How do I build this?" "What is the right architecture for this use case?" "How should I configure this for production?" The dominant demand is not reactive support. It is proactive architectural guidance. Users are not primarily reaching for the assistant when things break. They are reaching for it when they are building.

We used this same technology stack to launch a **Sales Assistant** for internal users, which revealed where AI value concentrates inside an organization. Revenue teams primarily used the tool for:

1. **Opportunity analysis:** In-depth deal intelligence that retrieves and synthesizes context at the individual opportunity level (Use Case Constrained 2.7% or 4,916 sessions)

2. **Account analysis:** A strategic 360-degree view that aggregates historical customer data and cross-product trends at the account level (Use Case Constrained 1.51% or 2,909 sessions)

 | © 2026 elasticsearch B.V. All rights reserved.

Together, those two use cases represented the overwhelming majority of Sales Assistant usage, nearly equal in volume. Sellers wanted both forward-looking deal intelligence and backward-looking account context in roughly equal measure. By passing real-time CRM data, Salesforce opportunity, and account IDs directly into the context window, the assistant moved up the value chain from fixing problems to generating revenue-relevant insight. We now track this as a distinct ROI category similarly to Support ticket deflection and measure it via an AI-influenced pipeline metric.

Evolution of intent: How questions mature over time

One of the clearest signals in the data was not a single metric, but a pattern. The questions users asked in January 2025 were fundamentally different from the questions they asked in December 2025. Early in the year, the dominant query type was navigational and setup-oriented. By Q4, approximately 30% of conversations focused on what we classify as Elasticsearch internals (e.g., shard allocation failures, heap utilization, and cluster performance under load). Diagnostic "internals" queries outnumbered basic "how-to" queries by 2.5 to 1. Our assumption is that this shift is driven by two distinct, parallel trends:

1. **Customer maturity:** As users on our platform progress past initial setup, the demand for complex production optimization surpasses basic onboarding support.

2. **Broader market:** As hybrid search and semantic relevance became more commonly known as critical components to power AI applications, the broader market, including our customers, shifted their focus toward the rigors of production-scale operation and optimization.

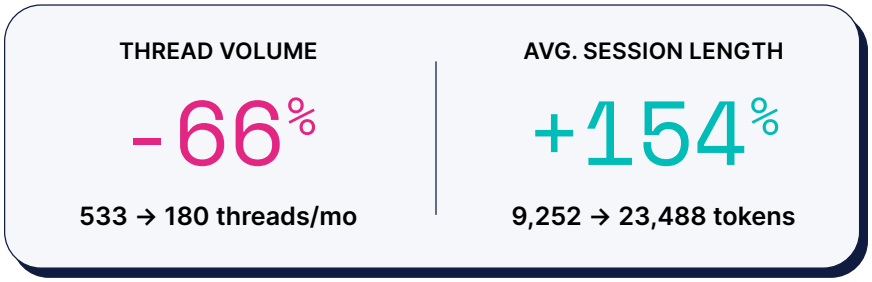
> **Knowledge bases are not static: Users who stay on the platform do not stay asking the same questions. They go deeper, and your knowledge base needs to go with them.**

The documentation and knowledge base had to keep pace. The available content was optimized for setup and onboarding, which is where significant documentation effort goes, but the users had moved into production operations mode. The gap between what users were asking and what the knowledge base contained had shifted from "Where is the doc?" to "There is no doc for this yet." This data point was passed on to our Support and Product Documentation teams to review and adjust as needed.

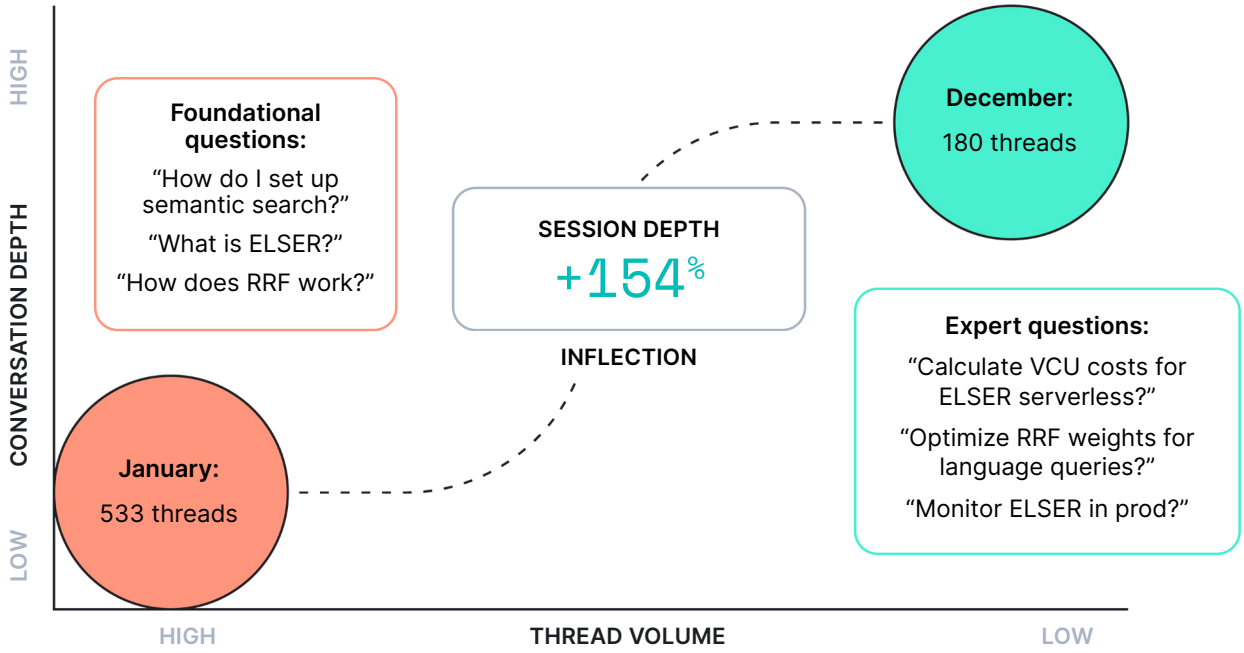
 | © 2026 elasticsearch B.V. All rights reserved.

Case study: The evolution of "hybrid search" as a topic

The support evolution for hybrid search provides a concrete example of this shift. From January 2025 to December 2025 there was a material shift and inverse relationship between volume and depth.



Hybrid search maturity curve — from high volume low depth to low volume high depth



9,252 avg. tokens/session → 23,488 avg. tokens/session · Jan.–Dec. 2025

*Fig. 3 — **Evolution of "hybrid search" as a support topic** across 2025, measured by monthly thread volume and average session length in tokens. In January, the assistant fielded 533 threads averaging 9,252 tokens — questions were foundational: "How do I set up semantic search?" "What is ELSER?" "How does RRF work?" By December, volume had fallen to 180 threads while average session length grew to 23,488 tokens, a 154% increase. The questions had changed entirely: "How do I calculate VCU costs for ELSER in serverless?" "How do I optimize RRF weights for foreign language queries?" "How do I monitor ELSER memory usage in a production cluster?" Fewer users were asking about hybrid search because the ones still asking had moved well past the basics, a 66% drop in volume masking a 154% increase in depth.*

 | © 2026 elasticsearch B.V. All rights reserved.

> **Plan for fewer threads and deeper conversation: The topic isn't shrinking, but the user base is maturing its use case.**

A hidden use case for a multilingual user base

There was one pattern in the data that did not appear in any product roadmap. Users were using the assistant as a translation and language tool. More than 879 threads were explicitly for translation services. When the definition is expanded to include grammar assistance, proofreading, drafting support, and rephrasing, that figure grows to account for approximately 22,192 threads.

Elastic operates across dozens of countries, and a significant portion of our employees and customers work in English as a second or third language. The assistant provided a way to draft technically precise support cases, internal communications, and documentation with confidence, something that previously required either a native speaker colleague, an additional translation service, or worst case, going without. The quantity and quality of these language interactions proves this is no anomaly; it is a clear signal of a multilingual user base adopting the assistant as a communication equalizer.

> **Language and translation is invisible: In traditional analytics, these communication-type use cases leave no trail of tickets, escalations, or complaints. They only show up if you are classifying conversation intent at the thread level. If you are building for a multilingual userbase, consider building targeted experiences to support these users.**

 | © 2026 elasticsearch B.V. All rights reserved.

Analysis of AI quality

Quality measurement

Before diving into the specific findings of this review, it is essential to define the metrics used to evaluate system efficacy. For the purposes of this report, we anchor our analysis on two primary definitions:

1. **Quality:** Quantified via a sentiment analysis model that evaluates the helpfulness, tone, and resolution of an interaction

2. **Conversation session:** Defined as a sequence of one or more question and answer pairs centered on a single topic or problem statement

By correlating these quality scores with user interaction patterns, several counterintuitive findings emerged. These insights have direct implications for anyone building production-scale AI systems on the Elasticsearch Platform or any other technology stack.

High token count is not always a cost problem

Our initial hypothesis was that long sessions potentially indicated a struggling user, someone who failed to receive a satisfactory answer and was "going in circles." The data countered this assumption. Sessions exceeding 50,000 tokens (what we categorize as "deep sessions") averaged a quality score of 9.74 out of 10, significantly higher than the global average of 9.45. Our top 10 sessions (averaging 583,000 tokens) averaged 9.8 out of 10. Nine of the top ten highest-token sessions in the dataset scored 10/10.

 | © 2026 elasticsearch B.V. All rights reserved.

Session type	Avg. quality	Sessions
All sessions	9.45/10	149,432
Deep sessions (>50K tokens)	9.74/10	4,026
Top 10 sessions (average 583K tokens)	9.8/10	10

What was driving the longest sessions? The primary driver was in-depth sessions on key technical architect decisions, such as Elastic Common Schema (ECS) mapping. Users were pasting complete log schemas with sometimes 50 or more fields and asking the assistant to generate full ECS mappings and ingest pipelines. The largest session in the dataset reached 640,659 tokens and was entirely ECS-related. These were not confused users; they were architects using the assistant as a technical partner for a task that would otherwise require hours of manual, field-by-field mapping.

 | © 2026 elasticsearch B.V. All rights reserved.

This trend tracks directly with the maturity of our user base. As users moved hybrid search deployments into production, they required ECS-compliant data for semantic indexing to function correctly. The increased complexity in the second half of the year correlates with the reciprocal rank fusion (RRF) adoption on the search maturity curve. Users were not simply using the assistant more; they were using it to solve progressively harder problems.

> **Token volume is a value signal: For us, the sessions that consume the most tokens often represent the highest-value engineering work. In these contexts, high token consumption correlates with near-perfect accuracy and significant time savings for senior technical staff.**

Rethinking the cost model

Treating a 400,000-token session as an expensive anomaly to be optimized away is, in our opinion, the wrong conclusion. The right question to ask is: What is the equivalent human cost for a task and what did we save? Consider the following illustrative scenario based on our observations:

Metric	Assumed value
Estimated time saving for each technical support session	15 minutes
Observed AI use cases that equate to a technical support session	5,000
Estimated time saved for content generation	7 minutes
Observed AI use cases that equate to a content generation session	3,000
Full-time employee (FTE) burdened cost (salary, bonus, benefits)	$100 per hour

 | © 2026 elasticsearch B.V. All rights reserved.

$$\text{Total savings} = \left(\frac{(5{,}000 \times 15) + (3{,}000 \times 7)}{60} \right) \text{x } 100 = \$160{,}000$$

In the above scenario, we achieved a theoretical dollar savings of time of $160,000. This figure is calculated by multiplying the total hours reclaimed across the user population by the average hourly rate of the professionals performing the work. Taken a step further, total return on investment can further be calculated by subtracting LLM token and infrastructure costs. While these numbers illustrate calculated savings (no budget returned to the bottom line), we used these calculations to explain our impact to the business, which for the first six months of 2025 was roughly $2.5 million.

Organizations building AI tools must weigh the cost of tokens against the value delivered to the user and organization. A deep session that resolves a complex architectural problem provides a measurable return on investment through saved engineering time and faster time to production. For Elastic, customers leveraging our Support Assistant to do similar activities are saving us from time spent on support tickets. **The metric that matters is value delivered per interaction, not cost per token.**

Partial context retrieval degrades quality

Perhaps the most significant finding in our analysis of RAG was the impact of search relevance and answer quality on the LLM's reasoning. We evaluated answer quality across three distinct retrieval outcomes:

1. **Helpful:** Documents retrieved were directly relevant to the question being asked. For example, a search for apples returns articles on apples.

2. **Partial:** Documents retrieved were tangentially related but not directly useful. For example, a search for apples returns articles on pears.

3. **None:** No documents were retrieved at all. For example, a search on apples returns no articles and relies only on the knowledge of the LLM about apples.

 | © 2026 elasticsearch B.V. All rights reserved.

The relationship between these outcomes and quality scores reveals an interesting trend:

RAG retrieval outcome	Avg. quality score	Thread count
Helpful (directly relevant)	9.81/10	80,887
None (no documents retrieved)	9.18/10	65,624
Partial (tangentially relevant)	8.15/10	9,732

> **Partial retrieval is a reliability risk: A retrieval pipeline that always returns something is not a safe default. It can be a risk and reduce quality of response.**

When provided with partial context, the model produces slightly worse answers than no context at all. When given documents that were adjacent but not directly relevant to the question, the model felt an obligation to use them. This results in contextual drift, where the assistant confidently synthesizes conclusions the source material did not support or absorbs misleading framing from the source material. In short, the model performs better relying on its own training than it does when distracted by adjacent but irrelevant data.

The design implication is clear: Retrieval thresholds matter more than retrieval volume. A system configured to return the top 10 documents regardless of relevance score will routinely deliver partial context and can degrade quality and performance. The correct behavior is to set a strict confidence threshold and return nothing when no document clears it. "I don't know" is a much more valuable and safer response than a confidently wrong one.

 | © 2026 elasticsearch B.V. All rights reserved.

Retrieval quality degrades as product surface area grows

Despite Elasticsearch core capabilities being the most discussed product capability in the dataset by a wide margin (118,723 threads, ~3x more than the next largest product), it recorded one of the lowest RAG hit rates (percentage of queries where relevant documents appear in the results) in the system. In contrast, Beats, a smaller and more bounded product, outperformed it substantially.

Product	Threads	RAG hit rate
Elasticsearch	118,723	54.3%
Kibana	43,199	59.3%
Beats	10,344	66.1%
Logstash	10,124	57.0%

This is not a documentation failure; it is a long-tail problem. As a product's surface area grows, its question surface area becomes effectively infinite. In our context, Elasticsearch's question surface area is effectively infinite. The intersection of version specificity, configuration permutations, SDK behaviors, and cloud deployment variants creates a category of edge-case questions that no knowledge base can fully anticipate. The larger the product, the harder it is to maintain retrieval coverage, and the harder it is to know where the gaps are.

 | © 2026 elasticsearch B.V. All rights reserved.

Visualizing context gaps

To make these gaps visible, we performed semantic clustering of the 159,084 RAG searches and produced 740 distinct topic clusters. Each cluster represents a potential "user manual", where a subject has high demand, but documentation coverage is thin or absent.

For example, the largest cluster by volume, "Kibana configuration validation," contained 3,470 searches across 3,340 unique query variations. Upon inspection, this was a catch-all for all diverse Kibana queries: troubleshooting (30%), alerting (15%), security (11%), dashboards (8%). The signal was clear: Kibana generates broad, diverse demand across its entire feature surface that no single document can satisfy. Similarly, clusters focused on Elasticsearch indexing performance and version upgrade paths highlighted areas where production operations content was thin and users were arriving with questions the knowledge base could not fully answer.

> **Query clustering can drive knowledge base roadmap: By analyzing where retrieval fails or clusters around thin content, organizations can prioritize documentation based on actual user friction.**

Query clustering run on a regular cadence is a documentation roadmap ranked by actual user demand. This is a more reliable content prioritization signal than editorial judgment or page view analytics because it reflects what users could not find rather than what they already knew to look for.

Specialized tools outperform generalists and grow faster

While our Toolbox strategy was born out of operational necessity, addressing the distinct workflows of different teams, the resulting adoption validated its primary design principle. Purpose-built tools consistently achieved higher quality scores and significantly faster growth and adoption trajectories than our general assistant. What we did not fully anticipate was how clearly the data would validate the approach.

The Knowledge Drafter, a tool built specifically for documentation authoring, achieved an average answer quality of 9.85 out of 10. Meanwhile, the general-purpose Support

 | © 2026 elasticsearch B.V. All rights reserved.

Tool	Avg. quality score	Usage trajectory
Knowledge Drafter	9.85/10	11x growth Q1 → Q4 (73 → 815 threads)
Case Summarizer	9.46/10	6.9x growth Q1 → Q2 (244 → 1,674 threads)
Support Assistant	9.46/10	46% decline from Q1 peak to Q4

Assistant, processing approximately 196,000 threads, averaged 9.46 in quality. Although the gap is modest in absolute terms and consistent across the full year, the growth trajectory was more telling. Purpose-built tools such as the Case Summarizer grew 6.9x from Q1 to Q2, and the Knowledge Drafter grew 11x from Q1 to Q4. In the same period, the general Support Assistant declined 46% from its Q1 peak as users migrated toward alternate tools, either specialized internal tools or external general-purpose assistants.

> **Purpose-built tools win at quality: Users prefer tools that feel scoped to the specific workflow and require specialized assistants to provide better quality.**

The pattern is consistent enough to be a design principle: Users prefer tools that feel scoped to their specific workflow. A general assistant is a good starting point and a useful fallback, but the ceiling for both quality and adoption is lower than a tool that does one thing well. Each specialized tool you build has a faster adoption curve than the general tool it supplements because users arrive with a clear job to be done and the tool is configured to do exactly that job.

 | © 2026 elasticsearch B.V. All rights reserved.

It is also worth acknowledging that Support Assistant's 46% decline is not a failure signal of the platform, but a reflection of a maturing market. During 2025, broad "zero-shot" AI assistants from major vendors became significantly more capable and widely available externally but also internally at Elastic. Users turned to those tools for generic questions but returned to the Elastic toolset for domain-specific, Elastic-focused work.

This shift allowed our foundational infrastructure to move up the value chain. By focusing on tuning RAG search results, the application of these capabilities to new, specialized use cases allowed for AI acceleration throughout the company. Specialized tools that leverage your proprietary data are the true differentiators in the era of AI.

Version boundaries expose a class of failure unique to code generation

While approximately 2.4% of threads received quality scores at or below 3 out of 10, these failures are not random. They clustered tightly around SDK version boundaries, the exact edges where the model's training data and the current API surface diverge.

The most representative example in our dataset was a .NET Client v8.19 RequestParameters failure, a pattern we observed across multiple threads. In these cases, users asked how to add querystring parameters to the GetTaskAsync method. The assistant generated syntactically valid C#:

```
var requestParameters = new RequestParameters();
requestParameters.AddQueryString("wait_for_completion", "true");
```

On the surface, this looks perfect. It references real classes and follows standard conventions. However, the code could not execute. RequestParameters is abstract in v8.19 and cannot be instantiated. While the code might compile correctly, it fails at runtime. In one example, the user caught it and corrected the assistant, but the thread ended with a technical accuracy score of 4 out of 10 against a platform baseline of 9.46.

 | © 2026 elasticsearch B.V. All rights reserved.

> **In code-generation, "mostly right" is functionally "completely wrong": One hallucinated method in a working codebase can undo a lot of goodwill that accurate answers build up.**

The architectural response to this class of failure is version-specific retrieval. The knowledge base needs to be indexed not just by product but by version, with retrieval logic that matches the SDK version referenced in the user's query to the corresponding documentation. This is not optional for codebases with active release cycles; it is the minimum viable safeguard. The cost of adding a validation step is low. The cost of a developer shipping non-executable code that is traced back to your assistant is not.

Version freshness is not a nice-to-have

The .NET hallucination is the most documented version-boundary failure in this dataset, but it is not the only signal of a broader challenge: Version specificity drives a disproportionate share of user need. Upgrade-related queries (users asking about migration paths, version compatibility, and what changed between releases) account for approximately 1,600 threads annually. These questions have an incredibly short shelf life. An answer that was accurate for version 8.17 may be wrong or incomplete for 8.19. A RAG system that does not actively prioritize current documentation will serve stale answers to exactly the users who need accuracy the most: those in the middle of an upgrade.

The practical implications and cost of stale context goes beyond the upgrade workflow. Every product release is also a knowledge base debt event. Any content that references version-specific behavior (API parameters, configuration syntax, SDK classes) begins to decay the moment a new version is tagged. Without version-aware freshness scoring in your retrieval pipeline, this decay remains invisible until a user experiences a failure. And any version-based failure represents a measurable cost of letting version freshness slip.

 | © 2026 elasticsearch B.V. All rights reserved.

Your retrieval pipeline is hiding knowledge gaps from you

Most RAG pipelines are configured to always return a fixed number of documents (the top "k" results) regardless of relevance score. While this is the default behavior in most implementations, it creates a significant blind spot that is easy to miss: When the pipeline always returns something, you never see a zero-result event. Without zero-result events, you cannot identify where your knowledge base genuinely fails to cover a topic.

In our pipeline, genuine knowledge gaps (low RAG hit rate) were masked by low-relevance documents that still cleared the return threshold. The system dutifully returned 10 documents in response to every query, even when none of them were meaningfully relevant. That not only degraded answer quality, as established in our partial context finding, but it ensured the coverage gap remained invisible by design.

> **A zero-result log is a strategic asset, not a system failure: Every query that returns nothing is a topic your users are asking about that your knowledge base cannot answer. That is exactly the signal you need to prioritize documentation investment.**

As mentioned earlier, the fix is a confidence threshold: Return documents only when relevance clears a defined minimum floor and return nothing when nothing does. This design change serves two purposes: It prevents the partial-context quality degradation described earlier, and it makes knowledge gaps visible. Implementing a strict confidence threshold transforms these failed retrievals into a data-driven content roadmap, allowing teams to prioritize documentation based on verified user demand.

 | © 2026 elasticsearch B.V. All rights reserved.

Tactical takeaways: An operational roadmap

Derived directly from our analysis of 209,220 threads, the following actions represent the most impactful changes a developer can make to move an AI system from a successful prototype to a production-grade tool. These findings did not emerge from a research environment. They came from a production system handling real customer and employee interactions at scale, built on the same Elasticsearch infrastructure available to anyone reading this report. The recommendations below are not theoretical; they are the operational conclusions we drew from a year of running this system and analyzing the data accumulated.

On data strategy and retrieval architecture

To ensure response accuracy and long-term utility, developers must treat their knowledge base as a dynamic data product and their retrieval pipeline as a precision-weighted gateway. High-quality RAG requires a dual focus on context retrieval and relevance.

- **Implement a strict relevance floor:** The partial context finding is the most immediately actionable result in this dataset. A retrieval system configured to always return the top N documents, regardless of relevance score, will routinely degrade answer quality below what the model would achieve without any retrieved context at all. Tune your thresholds to be strict. The impact on AI response quality from saying "I don't know" is a far more valuable response than a confidently incorrect one.

- **Pivot from onboarding to customer maturity:** Every user base follows a maturity curve. Early on, they ask foundational "how to" questions. As they stay with your product or platform, they transition to complex high-stake queries. Your data strategy must evolve from basic FAQs to include deep-dive procedural guides, edge case documentation, and any other content that supports complex problem-solving. In our case, that means evolving our roadmap from simple trouble shooting to architectural guidance.

- **Treat context as a versioned asset:** In any fast-moving environment, accuracy has a shelf life. Whether you are dealing with software versions, insurance policy upgrades, or shifting legal regulations, serving stale data is the primary driver of inaccurate responses. Tag your knowledge base documentation with additional metadata (e.g., version, validity dates) to ensure that when a user asks, they receive relevant responses.

 | © 2026 elasticsearch B.V. All rights reserved.

- **Analyze documentation gaps:** Every query that returns no results is a direct signal of an unmet user need. By performing semantic clustering on your search queries, you can move from guessing what documentation to write to building a context roadmap. This transforms your retrieval logs into a ranked priority list.

On product and deployment architecture

- **Build a toolbox, not a single chatbot:** Embedded or specialized solutions have better quality scores and faster adoption curves than general-purpose tools. Identify your highest-frequency, most clearly scoped workflows (case summarization, document drafting, opportunity analysis) and build focused agents for each. A general assistant is a useful starting point to deliver usage data that tells you which workflow to target next.

- **Deploy to internal users first:** Internal deployment always matters as a proving ground: The failure modes your employees surface are the ones your customers should never encounter.

- **Delight users that are stuck:** Users reach for AI most often when they are stuck. Ensure your agents are embedded at the point of highest friction. Success at this specific moment is the fastest path to long-term adoption.

On measurement

It's crucial to be deliberate about how you define success metrics for each user population because the right metric depends on the use case your tool is designed for. Define success by user population:

- **External users (customers):** In a support context, users are engaging reactively. Success should be measured by ticket deflection and first-response resolution. High session frequency here might actually indicate a failure in the UI or retrieval, whereas a single, resolved session is the ideal outcome.

- **Internal users (employees):** For employees, true success depends on how seamlessly AI integrates into their workflows and empowers them to take action. Rather than focusing solely on frequency, prioritize measuring task completion and depth of use. Usage patterns naturally fluctuate across different applications. A general assistant, for instance, will inevitably record more sessions than a specialized opportunity analysis tool. Judge each tool by its specific purpose rather than a single standard.

 | © 2026 elasticsearch B.V. All rights reserved.

Once you've defined success by user type, then use the following techniques to capture and enrich your measurement model:

- **Treat high token counts as a value signal:** Session depth correlates positively with quality and user satisfaction. Before implementing rate limits, analyze your longest sessions. Treat high token counts as a signal to investigate, not a cost to optimize away. In our case, these represented the highest-value work being done such as complex architectural analysis and schema mapping.

- **Prioritize first-session quality as your primary retention metric:** A user decides whether to return based on their first interaction. Every investment in retrieval precision and fallback behavior is ultimately an investment in converting a first time user into a habitual power user.

- **Instrument RAG retrieval quality per interaction:** Capture, index, and store your generative AI pipelines for introspection. This includes calculating retrieval context quality per interaction by building a structured field in your evaluation index that annotates whether the retrieved document was helpful, partial, or absent. Without this data, you may see that quality is degrading but fail to understand the root cause.

- **Track power users, not just volume:** The formation of a power user cohort, defined as a small group of habitual daily users, is the leading indicator of sustained ROI. If that cohort is not forming, the problem is almost always first interaction quality. The fix likely requires a mix of context refinement and retrieval optimization.

- **Deploy an LLM quality judge from day one:** Modern sentiment analysis on a 1–10 scale is more valuable than user systems tagging good, neutral, or bad labels. The infrastructure required to score threads in near real time is modest while the value of maintaining a continuous quality signal is substantial. Retroactive evaluation of a year of conversation threads is possible but expensive. By the time a quality problem becomes visible through user feedback, it has already affected thousands of interactions.

 | © 2026 elasticsearch B.V. All rights reserved.

Concluding thoughts: Driving business needs through technical precision

The path forward

The analysis of one million messages reveals a fundamental truth: The success of an AI initiative is all about the feedback loop, not solely about selecting the right LLM. It's an ongoing engineering discipline focused on monitoring and Context Observability.

Our data shows that the most impactful improvements do not come from larger models, but from more precise retrieval, a deeper understanding of the user's maturity curve, and a shared infrastructure for document retrieval and performance monitoring. As we look toward the next year of operation, our focus continues to shift from simply "answering questions" to building a toolbox of specialized agents. These agents, built on a foundation of strict relevance and demand-driven documentation roadmaps, will continue to evolve alongside our users. By treating raw interaction logs as a strategic asset, any organization can transform their AI from a novel experiment into a mission-critical engine for revenue-relevant insight.

Driving business needs

If you are building a business case for a deployment of this scale, the data in this report, along with our own experience, suggests three distinct business impact categories worth measuring:

1. **Self-service support:** This represents the count of interactions where a user successfully resolved their query without needing to submit a request or escalate to a human.
2. **Productivity as reclaimed capacity:** This measures skilled human time returned as raw hours, dollar value, or total ROI. For example, an engineer cross-referencing multiple product versions in a single AI session instead of manual research across multiple documents. This is measurable using a time estimation model illustrated earlier in this report.
3. **Knowledge insights:** Every query an assistant receives is an unfiltered signal about what your customers and employees are actually trying to accomplish. Aggregated and clustered at scale, those signals provide a demand ranked map of where your context is strong and incomplete. This is a clear indicator back to your business on how to best retain customers and improve customer success.

 | © 2026 elasticsearch B.V. All rights reserved.

The data in this report is yours to build on, but you do not have to build alone. Whether you are currently architecting your first RAG pipeline or scaling a global AI toolbox, the most valuable insights often come from the peers who are in the trenches with you.

We invite you to join our Slack Community, a dedicated space for engineers, architects, and product leaders to discuss the architectural patterns, retrieval strategies, and operational challenges described in this review.

Join the Elastic Slack Community

 | © 2026 elasticsearch B.V. All rights reserved.

Appendix

Data corpus and collection

This report is drawn from data collected between January 1 and December 31, 2025, across five tools: the external Support Assistant, the internal Support Assistant, the Sales Assistant, the Case Summarizer, and the Knowledge Drafter. The following core metrics define the scope of the dataset:

- **955,020 messages:** The total count of individual inputs and outputs, combining user questions and AI responses across all tools and sessions
- **209,220 conversation threads:** Systematic groupings of messages organized around a single topic or problem statement from within a user session; the primary unit of analysis used throughout this report
- **149,432 unique user sessions:** Discrete periods of user engagement, each representing one continuous interaction with a tool

The knowledge base that underpins all tools draws from three primary source categories:

1. **Technical Support knowledge base:** Knowledge base articles are authored continuously by support engineers following a knowledge-centered service approach and stored entirely on Elasticsearch with semantic search enabled via ELSER and vector embeddings. The full Elastic support knowledge base is covered, including internal and customer-facing articles. Articles are authored in real time as support engineers resolve cases, meaning the knowledge base grows daily.
2. **Product documentation and web content:** Documentation and web content are Ingested via a combination of direct API, Elastic Connector, and web crawler. This corpus includes **114 major and minor product versions** of Elastic product documentation, technical blogs from elastic.co and onboarding guides.
3. **LLM enrichment layer:** Each ingested document was processed through an AI-powered pipeline to generate structured metadata and document-level summaries before indexing, improving hybrid search relevance across the full corpus. **More than 300,000+ AI-generated summaries** were produced and indexed for retrieval augmentation. Enrichment metadata was stored as structured fields alongside source documents in Elasticsearch.

 | © 2026 elasticsearch B.V. All rights reserved.

The full build process, from knowledge library construction through retrieval tuning, was documented in the four-part technical blog series that accompanied the September 2024 public launch of the Support Assistant.

For the Sales Assistant, additional real-time context was injected directly into the LLM context window at query time. Salesforce opportunity and account IDs were passed alongside the user's query, enabling deal-level and account-level intelligence without requiring those records to be statically indexed into the knowledge base. This pattern, dynamic context injection alongside static retrieval, was a design choice for both real-time data use cases and enforcing additional document-level security constraints from Salesforce.

Technical stack

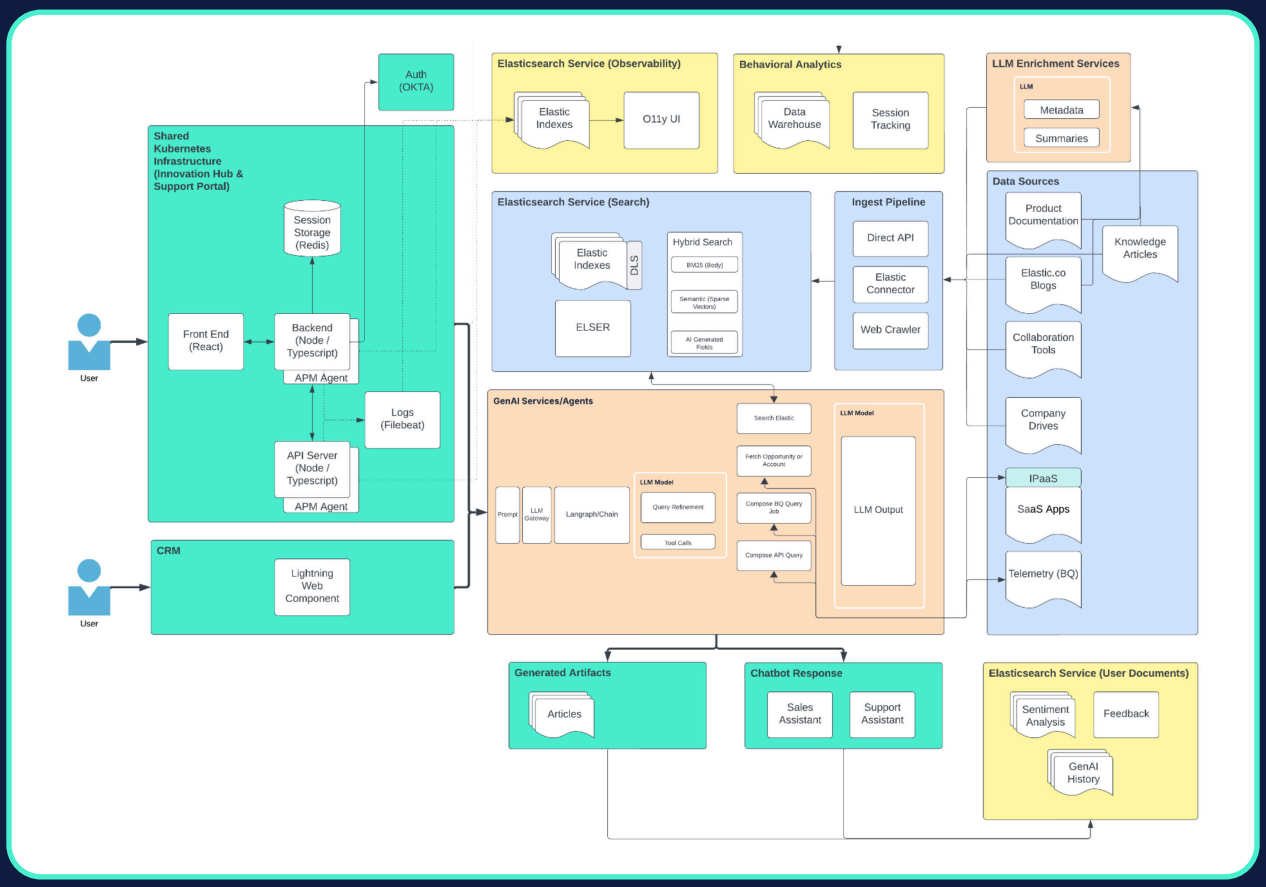


***Fig. 4 — GenAI platform architecture as of 2025**. The full platform as it operated during the 2025 reporting period.*

 | © 2026 elasticsearch B.V. All rights reserved.

All tools share a common RAG architecture built on Elasticsearch and deployed on a shared Kubernetes infrastructure (referred to internally as the Innovation Hub). The diagram above illustrates the full platform as it operated during the 2025 reporting period. The front end is a React application; backend services run on Node/TypeScript and include an API server and LLM gateway layer. All services are instrumented with APM agents for observability, with logs shipped via Filebeat and session state managed through Redis. Authentication is handled via Okta.

The retrieval layer uses hybrid search combining BM25 keyword search with ELSER sparse vector embeddings. For semantic search, **`text_expansion`** queries run against both title and summary embedding fields. Keyword relevance uses **`cross_fields`** with a tuned **`minimum_should_match`** parameter, with phrase matches receiving a higher boost to signal stronger relevance. The top results are assembled into a structured system prompt and passed to the LLM for generation.

Orchestration of multistep agent workflows uses LangGraph/LangChain. Tool calls within the GenAI Services layer enable discrete actions including Elasticsearch search, Salesforce opportunity and account retrieval, BigQuery, API query construction, and Google Sheet parsing. LLM output is routed back through the gateway before delivery to the user interface or Salesforce Lightning Web Component, depending on the access surface.

GPT-5 was the primary generation model throughout the 2025 reporting period; a small volume of interactions used earlier GPT model versions as a technical fallback. All LLM traffic was routed through a centralized AI gateway.

Behavioral analytics are captured separately via a session tracking layer connected to a data warehouse, with user interaction events encoded as JSON and stored in BigQuery for cross-team analysis. Observability for the Elasticsearch layer itself is handled via a dedicated Elasticsearch Service (Observability) instance feeding an Observability UI.

Quality evaluation

Each thread was scored on a 1–10 scale assessing helpfulness, tone, and resolution quality using a sentiment analysis model operating as an LLM quality judge. Threads were additionally annotated with a RAG retrieval outcome (Helpful, Partial, or None), a free-form use case label generated by a classification LLM with no predefined taxonomy, and a token count. This structured metadata was written back as fields in a dedicated Elasticsearch evaluation index, making the full quality dataset queryable using standard Elasticsearch aggregations.

 | © 2026 elasticsearch B.V. All rights reserved.

Adoption analysis used standard Elasticsearch aggregations: terms aggregations for use case distribution, date histograms for growth trends, and percentile aggregations for token distribution. The 740-cluster documentation gap analysis used k-means clustering over ELSER embeddings applied to the 159,084 RAG search queries in the corpus. Use case classification was performed using unconstrained LLM labeling followed by rollup into constrained categories for trend tracking and dashboards.

Privacy and data handling

Building a measurement system at the scale described in this report requires a deliberate approach to data governance from the start. Privacy architecture was not a post-hoc compliance consideration but a design constraint baked into the telemetry pipeline before the first interaction was logged.

All analysis in this report was conducted without the use of personally identifiable information. User identities are represented as hashed tokens throughout the pipeline. No names, email addresses, or account identifiers appear in the telemetry indices used for this analysis. This applies equally to internal Elastic employees and external customers, as both populations are treated as anonymized cohorts for the purposes of measurement and reporting.

Conversation content is evaluated at the thread level and summarized into structured metadata fields. The raw conversation text that produces those fields is retained separately under standard data governance controls and was not used in any of the aggregations that produced the findings in this report. What enters the evaluation index is the derived signal: a quality score, a use case label, question and answer summaries, a retrieval outcome classification, and a token count. The word for word text of the conversation does not travel with it.

Where this report references user behavior, session frequency, adoption patterns, power user cohort formation, and retention rates, those figures are derived exclusively from anonymized, aggregated telemetry. Every metric in this report describes a population, not an individual. The power user analysis that identifies 675 users driving 79.2% of volume is a statistical observation about a behavioral cohort, not a profile of identifiable people.

For teams building similar systems, this architecture has a practical benefit beyond compliance. By separating raw conversation content from derived evaluation metadata at the point of ingestion, the evaluation index remains lightweight, fast to query, and safe to share broadly across teams for analysis. The insights in this report were accessible to authors, product teams, and leadership without any of those stakeholders requiring access to raw conversation logs.

 | © 2026 elasticsearch B.V. All rights reserved.