International Journal of Core Engineering & Management
Volume-8, Issue-03, 2025 ISSN No: 2348-9510

ENHANCING SALES ENABLEMENT WITH RAG AND VECTOR SEARCH: REAL-TIME KNOWLEDGE RETRIEVAL FOR GTM TEAMS

Adish Rai
New York City, USA
rai.adish@gmail.com

Abstract

Go-to-market teams sit on large volumes of enablement content across wikis, PDFs, decks, product notes, CRM fields, and conversation highlights, yet reps still struggle to find the few paragraphs that matter for a specific account and persona. Retrieval Augmented Generation pairs vector search with a language model so the model only sees the most relevant snippets and can answer with citations. This paper presents a practical, AWS aligned design that ingests common GTM sources, chunks and embeds them, retrieves the best matches in real time, and generates seller ready answers and short draft emails. We cover a lightweight in memory mode for quick wins and a scalable Open Search based mode for large libraries, along with privacy and governance guardrails. The pattern is portable to other stacks with equivalent services. The approach converts scattered enablement material into reliable, real-time knowledge retrieval for go-to-market teams while maintaining enterprise security and compliance standards.

Index Terms – Sales enablement; retrieval augmented generation; vector search; Amazon Bedrock; Titan Embeddings; Open Search; go-to-market; sales playbooks

I. INTRODUCTION

Two everyday problems slow down go-to-market teams. First, it takes too long for sellers to find relevant information because content is scattered across knowledge bases, PDFs, internal chats, CRM notes, and recorded meetings. Second, on boarding new reps or handing off accounts is rarely clean. Context lives in many places, and the last mile answer that a rep needs for a call or email is often buried.

RAG addresses both problems. Instead of sending an entire library to a model, the system retrieves a few highly relevant passages by meaning and then generates a grounded response with citations. The result is fast, consistent answers with a link back to source material so managers and reps can trust and verify.

II. BACKGROUND IN PLAIN TERMS

A. Embeddings

An embedding model turns text into a vector, a list of numbers that captures meaning. Passages with similar meaning end up with vectors those are close to each other. This enables search by meaning, not only by keyword.

B. Vector Search
Vector search finds the passages whose vectors are closest to a query vector. For go-to-market teams, that means show the three sections most relevant to this question or persona even if the exact words do not match.

C. Retrieval Augmented Generation
RAG retrieves the top passages first, and then gives those passages to the language model to write a short answer or draft. Because the answer is built from retrieved text, it can cite where each point came from.

III. GTM USE CASES

Sales enablement questions are typically narrow and time sensitive. A representative preparing for a call may need a short explanation of a capability, a summary of how a product addresses a specific objection, or the most recent change in positioning for a given segment. Retrieval augmented generation supports these tasks by returning a brief, cited answer drawn from the enablement library and recent meeting highlights. When communication is required, the same retrieved passages can condition a concise email draft tailored to a selected persona, with citations so that the reviewer can verify the source material. The approach also assists onboarding and handoffs by synthesizing selected CRM notes, win or loss summaries, and product updates into a compact brief for a new account owner. During product launches the system prioritizes recent documents so that queries surface the latest messaging, value proof points, and enablement materials for the relevant industry or persona.

IV. METHODS: KNOWLEDGE BASES FIRST

The primary design uses Amazon Bedrock Knowledge Bases to handle chunking, embeddings, vector storage, retrieval, and grounding. This approach reduces custom code and shortens time to value while preserving essential controls such as metadata filters and model choice.

A. Data Sources
An organization registers one or more S3 prefixes as knowledge sources (for example, enablement PDFs, product notes, CRM note snapshots, and short meeting highlights). Each object includes lightweight metadata such as account identifier, industry, product, and date to support filtered retrieval.

B. Indexing and Retrieval
The Knowledge Base performs segmentation and embedding using an embedding model (for example, Amazon Titan Embeddings) and stores vectors in a managed vector store. At query time, an application calls Retrieve and Generate, which retrieves the most relevant passages by meaning and prompts a Bedrock text model to produce a concise answer that includes citations back to the source objects.

C. Delivery and Access
Answers are returned to the calling application for delivery in Slack or as a Salesforce side panel.

Access policies on the S3 prefixes and the Knowledge Base ensure that users only retrieve content they are permitted to see. Encryption at rest (KMS) and private connectivity (VPC endpoints) keep traffic inside the AWS boundary.

V. ALTERNATIVE: DIY VECTOR RETRIEVAL WITH OPENSEARCH

Some teams require fine grained control over segmentation, ranking logic, or multi-tenant isolation. In that case a do it yourself pattern uses AWS Lambda to ingest and segment documents, Amazon Titan Embeddings to create vectors, and Amazon OpenSearch Serverless with vector fields for nearest neighbour retrieval. The application embeds the query, retrieves the top passages from OpenSearch, and calls a Bedrock text model to generate a cited answer. This alternative offers tighter control and custom filters, at the cost of additional configuration and operations.

A. Tier A: In Memory Retrieval (Pilot Scale)

A pilot can operate without a persistent vector index. A scheduled Lambda extracts text, segments it into passages of approximately five hundred to one thousand words using natural section boundaries, and stores passage JSON with metadata in S3. The same function creates embeddings with Titan Embeddings and stores the vectors alongside each passage. At query time a Lambda loads the small set of vectors into memory, computes similarity with the embedded query, selects the top passages, and calls the Bedrock text model to produce a short answer with citations. This configuration minimizes setup and operational overhead and is suited to small corpora. As the collection grows, retrieval latency and accuracy can degrade because similarity search is performed in memory rather than through a dedicated index.

B. Tier B: OpenSearch Based Retrieval (Managed Scale)

For ongoing use at larger scale the ingestion process writes passages and embeddings to an OpenSearch Serverless collection with vector fields enabled. Each query is embedded and issued to OpenSearch for nearest neighbor search. The retrieved passages and metadata are supplied to the Bedrock text model to generate a cited answer. This configuration provides low latency retrieval for larger libraries and supports filtering by attributes such as product, industry, or date, as well as multi-tenant isolation. The improved control and performance require additional configuration and routine management.

VI. FRESHNESS AND STREAMING UPDATES

A steady flow of new content keeps retrieval useful.

A. With Knowledge Bases

Meeting systems post a web hook when a meeting ends. An Event Bridge rule triggers a Lambda that converts the payload into a short highlight with minimal PII, adds metadata such as account, opportunity, persona, and timestamp, and writes the object to an S3 prefix used by the Knowledge Base. The Knowledge Base syncs on a schedule or via API to pick up new objects and update the index. Recent content can be prioritized with regency filters at query time.

B. With DIY OpenSearch

The same Event Bridge and Lambda flow writes highlights to S3 and computes embeddings with

Titan Embeddings. The Lambda upserts vectors and metadata into an OpenSearch Serverless collection. Dedupe keys prevent duplicates, and a simple retention policy removes stale items. Metadata such as account, industry, and date enables filtered retrieval.

VII. OUTPUT FORMAT

The system returns a compact answer card followed by optional communication text. The answer card presents three to five concise statements that address the query, each supported by a short quotation and a source reference that links to the originating document. When requested, the system also produces a brief email draft of approximately ninety to one hundred thirty words tailored to the selected persona. The draft references one or two quotations and includes source pointers so that a reviewer can verify the claims. Links to the underlying sections allow the reader to continue to the full documents.

VIII. EVALUATION

Evaluation emphasizes accuracy, usefulness, and operational behavior. Accuracy is measured as the share of answers with citations in which the quoted text supports the corresponding claim. Usefulness is captured through a short rating scale and an estimate of time saved per question. Operational behavior is summarized by latency per answer, reported as median and ninety fifth percentile, the token cost per response, and the retrieval hit rate, defined as the share of questions for which the relevant passage appears among the top results. A simple study can compare a treatment group that receives the system's responses to a control group over several weeks.

IX. PRIVACY, RISK, AND GUARDRAILS

Data handling follows a minimization principle. Only the fields required by the prompt are passed to the model, and meeting highlights are preferred over full transcripts. Names and email addresses are avoided unless they are necessary for the use case. Applications should prevent personal data from being written to logs by redacting common identifiers at capture. When prompts must include names, traffic remains within AWS using VPC endpoints and KMS encryption at rest, and Bedrock data protection controls are applied. Access to sources respects document permissions at indexing and at query time. A simple recrawl schedule maintains freshness, and every answer carries citations so reviewers can verify assertions.

X. LIMITATIONS AND CHALLENGES

Conflicting sources may lead to inconsistent guidance; when this occurs the system should surface both views and label the conflict explicitly. Knowledge freshness is a practical concern and can be addressed with change detection and re-indexing for frequently updated content. Multilingual embeddings support global deployment where multiple languages are present. In some scenarios it is beneficial to combine unstructured passages with structured tables such as product or pricing data to improve specificity.

XI. FUTURE SCOPE

Future enhancements will address current limitations and expand system capabilities. Multilingual embeddings support will enable global deployment where multiple languages are present. Integration with structured tables such as product specifications and pricing data will improve response specificity and accuracy. Advanced conflict detection algorithms will automatically identify and flag contradictory information from different sources. Real-time change detection and incremental reindexing will maintain knowledge freshness for frequently updated content without full corpus reprocessing. The framework will be extended to support additional cloud platforms beyond AWS, with equivalent implementations for Microsoft Azure and Google Cloud Platform to support multi-cloud enterprise requirements.

XII. CONCLUSION

Retrieval augmented generation enables sales teams to obtain concise, verifiable answers at the point of need. A knowledge base first design simplifies deployment by managing segmentation, embeddings, vector storage, retrieval, and grounding within a managed service, while an OpenSearch based alternative offers additional control where required. With clear output formats, straightforward evaluation, and appropriate privacy controls, the approach converts scattered enablement material into reliable, real-time knowledge retrieval for go-to-market teams.

REFERENCES

1. Amazon Web Services, "What is retrieval augmented generation," 2025. [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/ (Accessed: September 24, 2025)
2. Amazon Web Services, "Data protection for Amazon Bedrock," 2025. [Online]. Available: https://docs.aws.amazon.com/bedrock/latest/userguide/data-protection.html (Accessed: September 24, 2025)
3. Amazon Web Services, "Amazon Titan embeddings models," 2025. [Online]. Available: https://docs.aws.amazon.com/bedrock/latest/userguide/titan-embedding-models.html (Accessed: September 24, 2025)
4. Amazon Web Services, "Amazon Titan embeddings G1 text parameters," 2025. [Online]. Available: https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-titan-embed-text.html (Accessed: September 24, 2025)
5. Amazon Web Services, "Amazon OpenSearch serverless vector search collections," 2025. [Online]. Available: https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless-vector-search.html (Accessed: September 24, 2025)
6. OpenSearch Project, "k-NN vector field type," 2025. [Online]. Available: https://docs.opensearch.org/latest/field-types/supported-field-types/knn-vector/ (Accessed: September 24, 2025)
7. Amazon Web Services, "Amazon OpenSearch serverless overview," 2025. [Online]. Available: https://docs.aws.amazon.com/opensearch-service/latest/developerguide/serverless.html (Accessed: September 24, 2025)
8. AWS Machine Learning Blog, "Build an end to end RAG solution using Amazon Bedrock knowledge bases," 2024. [Online]. Available: https://aws.amazon.com/blogs/machine-

learning/build-an-end-to-end-rag-solution-using-knowledge-bases-for-amazon-bedrock-and-aws-cloudformation/ (Accessed: September 24, 2025)

9. AWS Machine Learning Blog, "Evaluate the reliability of RAG applications using Amazon Bedrock," 2024. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/evaluate-the-reliability-of-retrieval-augmented-generation-applications-using-amazon-bedrock/ (Accessed: September 24, 2025)