Titan Enterprise RAG Engine
(Enterprise-grade AI Retrieval and Knowledge Intelligence Engine)

1. Executive Summary

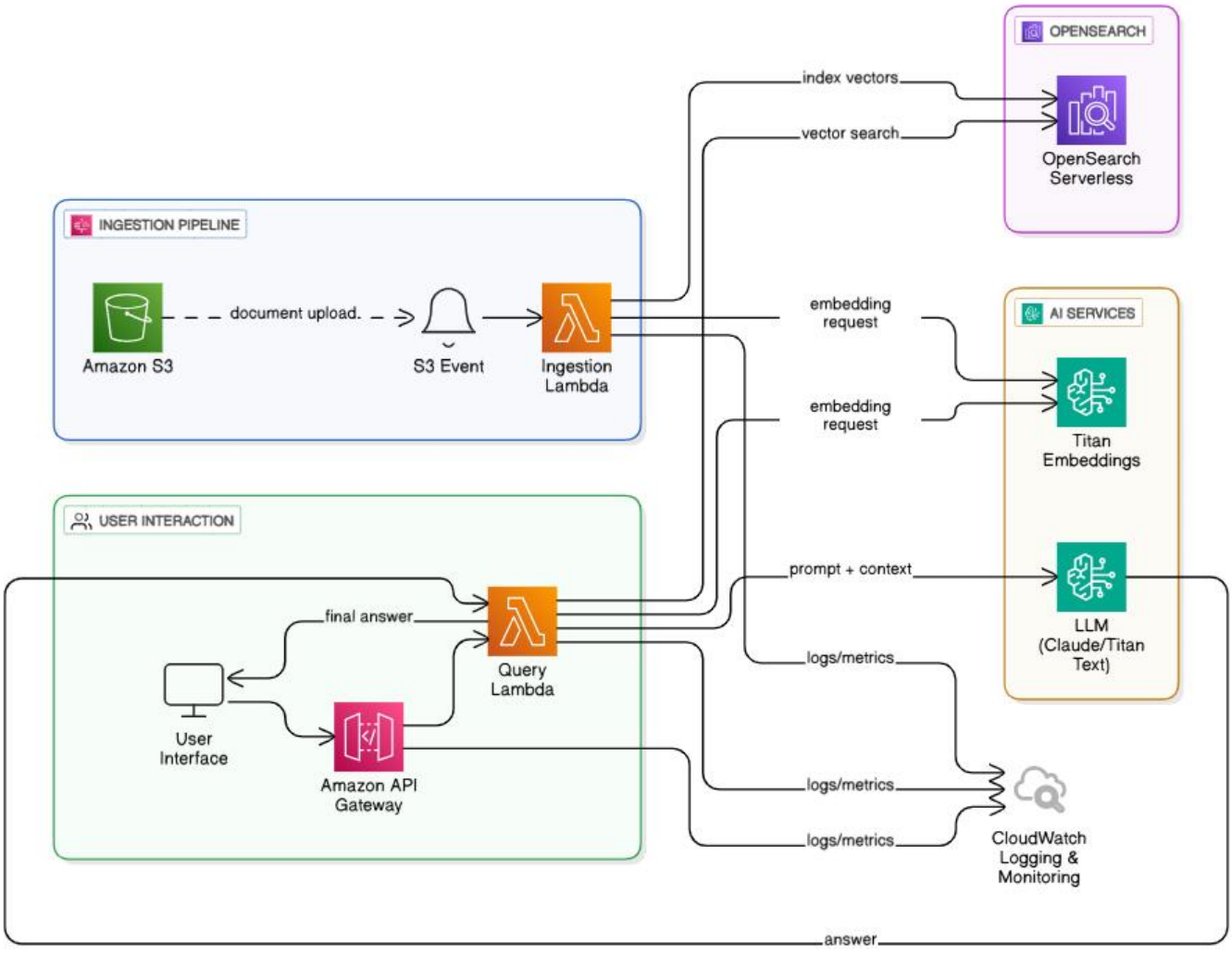
Titan Enterprise RAG Engine is a fully serverless, production-grade Retrieval-Augmented Generation (RAG) platform designed to enable enterprises to transform unstructured internal documents into an intelligent, searchable knowledge system.

The engine ingests private enterprise data using Amazon S3, generates high-quality semantic embeddings using Amazon Titan Embeddings v2, stores them in a serverless OpenSearch vector index, and runs fast KNN semantic search to retrieve the most relevant knowledge chunks. The retrieved context is then combined with Amazon Titan Text to produce accurate, grounded AI responses.

This architecture mirrors the design patterns used by modern AI knowledge systems in real corporate environments and demonstrates your ability to implement:

- Serverless, event-driven pipelines
- Enterprise-level RAG systems
- Vector search with OpenSearch Serverless
- AI-powered knowledge intelligence using Amazon Bedrock
- Scalable and secure AWS-native architectures

2. Architecture Overview



This architecture illustrates a fully serverless Retrieval-Augmented Generation (RAG) system capable of ingesting enterprise documents, generating embeddings, performing semantic search, and returning AI-generated answers to users in real time.

The Titan Enterprise RAG Engine consists of six core components:

1. Amazon S3 — Document Ingestion Source
Stores enterprise documents and triggers the ingestion workflow.

2. Ingestion Lambda — Data Processing Pipeline
Reads documents, chunks text, generates embeddings, and indexes vector data.

3. Amazon Bedrock — Titan Embeddings v2
Produces 1024-dimension semantic embeddings for RAG vector search.

4. OpenSearch Serverless — Vector Store
Stores embeddings and performs KNN semantic search across enterprise documents.

5. Query Lambda — Retrieval Engine
Generates query embeddings, retrieves relevant context, and invokes Titan Text for final responses.

6. API Gateway + Frontend UI
Provides a public endpoint and user interface for asking questions and viewing AI-generated answers.

3. End-to-End System Workflow

Below is the complete system workflow from data ingestion to answer generation:

STEP 1 — Document Upload
A user uploads a document (e.g., financial report, compliance file) into the S3 bucket.

STEP 2 — Automatic Trigger
S3 event notification triggers the Ingestion Lambda.

STEP 3 — Document Processing & Embedding
The ingestion Lambda:

1. Reads the file contents
2. Splits text into meaningful chunks
3. Sends each chunk to Titan Embeddings
4. Receives 1024-dimension vectors
5. Writes documents + embeddings to OpenSearch

STEP 4 — Vector Storage
OpenSearch Serverless stores:

- content
- vector_field
- source file
- chunk_id
- timestamps

STEP 5 — User Asks a Question
Example:
“What were the total sales in 2024?”
The frontend sends a JSON request:
{ "query": "What were the total sales in 2024?" }

STEP 6 — Query Processing
The Query Lambda:

1. Generates a query embedding
2. Performs ANN search in OpenSearch
3. Retrieves the most relevant text chunks
4. Builds a context-rich prompt
5. Calls Titan Text to generate the final answer

STEP 7 — Final Answer Returned
The frontend displays:

- AI Answer
- Retrieved Context (documents).This completes the RAG loop.

4. Key Features & Capabilities

- Fully Serverless Architecture (zero maintenance)
- Real-Time Document Ingestion upon upload
- High-accuracy Titan Embeddings v2 for semantic representation
- Scalable Vector Search via OpenSearch Serverless
- Context-grounded AI answers using Titan Text
- Modern Web Interface for interactive querying
- Secure by Design (IAM, private S3, isolated Lambdas)
- Production-readiness with logging, error handling, and modular components

5. Technologies & AWS Services Used

AI Layer

- Amazon Titan Embeddings v2
- Amazon Titan Text (Text Generation)
- Amazon Bedrock Runtime API

Vector Search Layer

- OpenSearch Serverless
- KNN (Approximate Nearest Neighbor) Search

Compute

- AWS Lambda (Ingestion & Query)
- Event-driven execution model

Storage

- Amazon S3 (Document storage & ingestion trigger)

API & Interface

- Amazon API Gateway
- HTML / CSS / JavaScript frontend UI
- Fetch API for client-side requests

Security

- IAM Identity-based Access Control
- Lambda Execution Roles
- OpenSearch Security Policies

6. Deep Dive: Component-by-Component Breakdown

6.1 Amazon S3 — Document Source

S3 acts as the enterprise document repository. When a new file is uploaded, an event triggers the ingestion Lambda. This allows the system to index new knowledge in real time without manual intervention.

6.2 Ingestion Lambda — Embedding & Indexing Pipeline

This function:

- Reads the uploaded file
- Cleans and chunks the text
- Calls Titan Embeddings v2 for vector generation
- Writes embeddings + content into OpenSearch Serverless

It transforms raw documents into structured, searchable vector knowledge.

6.3 Amazon Titan Embeddings v2

Titan v2 generates 1024-dimension embeddings optimized for:

- semantic understanding
- vector search
- enterprise knowledge retrieval

Its high-quality representations enable precise KNN search results.

6.4 OpenSearch Serverless — Vector Database

OpenSearch stores:

- text chunks
- vector embeddings
- file metadata

It supports KNN search, enabling fast approximate nearest neighbor lookups across large document sets. Serverless scaling ensures consistent performance with minimal overhead.

6.5 Query Lambda — Retrieval & Generation Engine

This function:

1. Accepts user questions
2. Generates a query embedding
3. Performs vector search
4. Retrieves top-K relevant chunks
5. Builds a context-aware prompt
6. Calls Titan Text to generate the final answer

Core responsibilities:

- Orchestration
- Retrieval logic
- Prompt construction
- Response formatting

6.6 Amazon Titan Text — LLM Generation

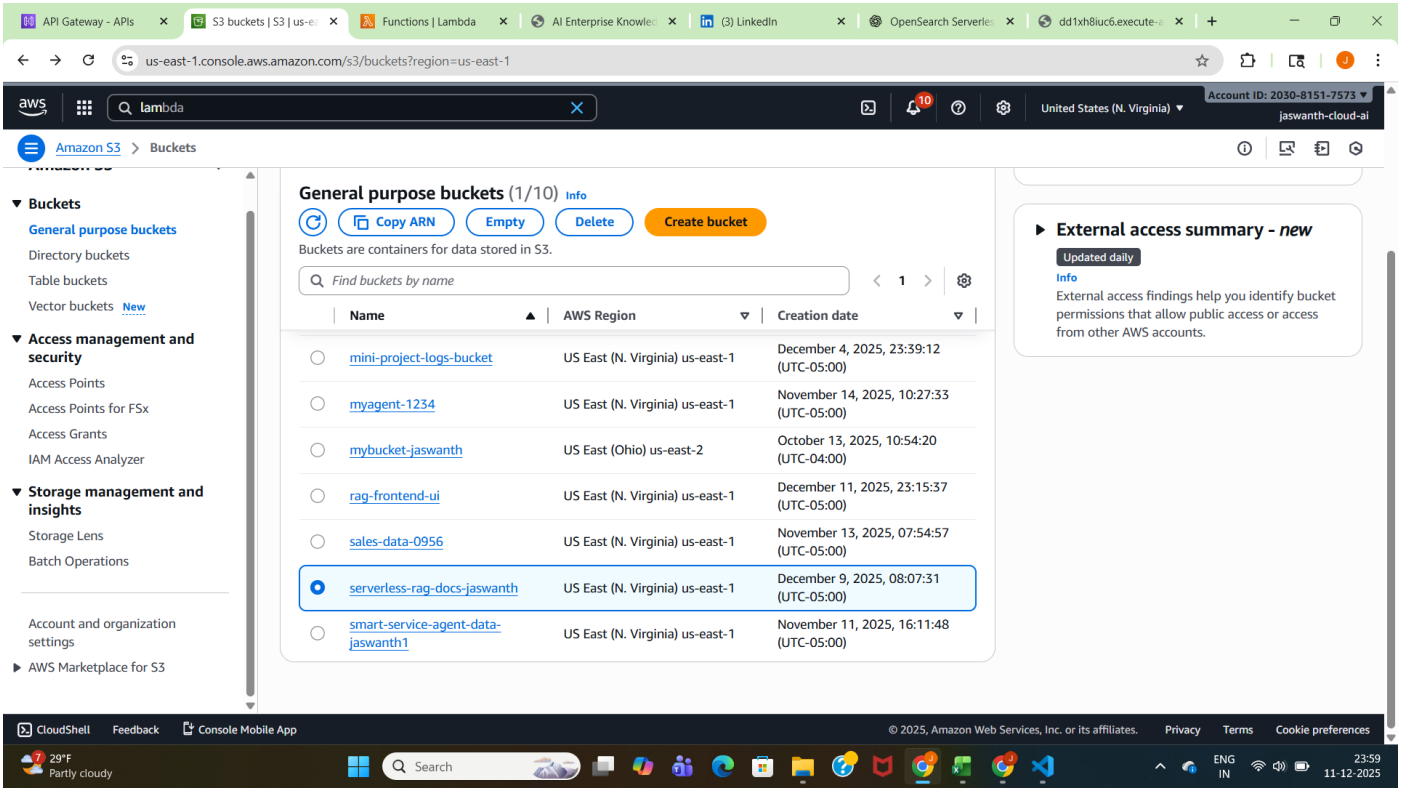
Titan Text receives:

- the user's question
- retrieved context

It produces a grounded, coherent response with significantly reduced hallucination because it relies on your enterprise knowledge base.

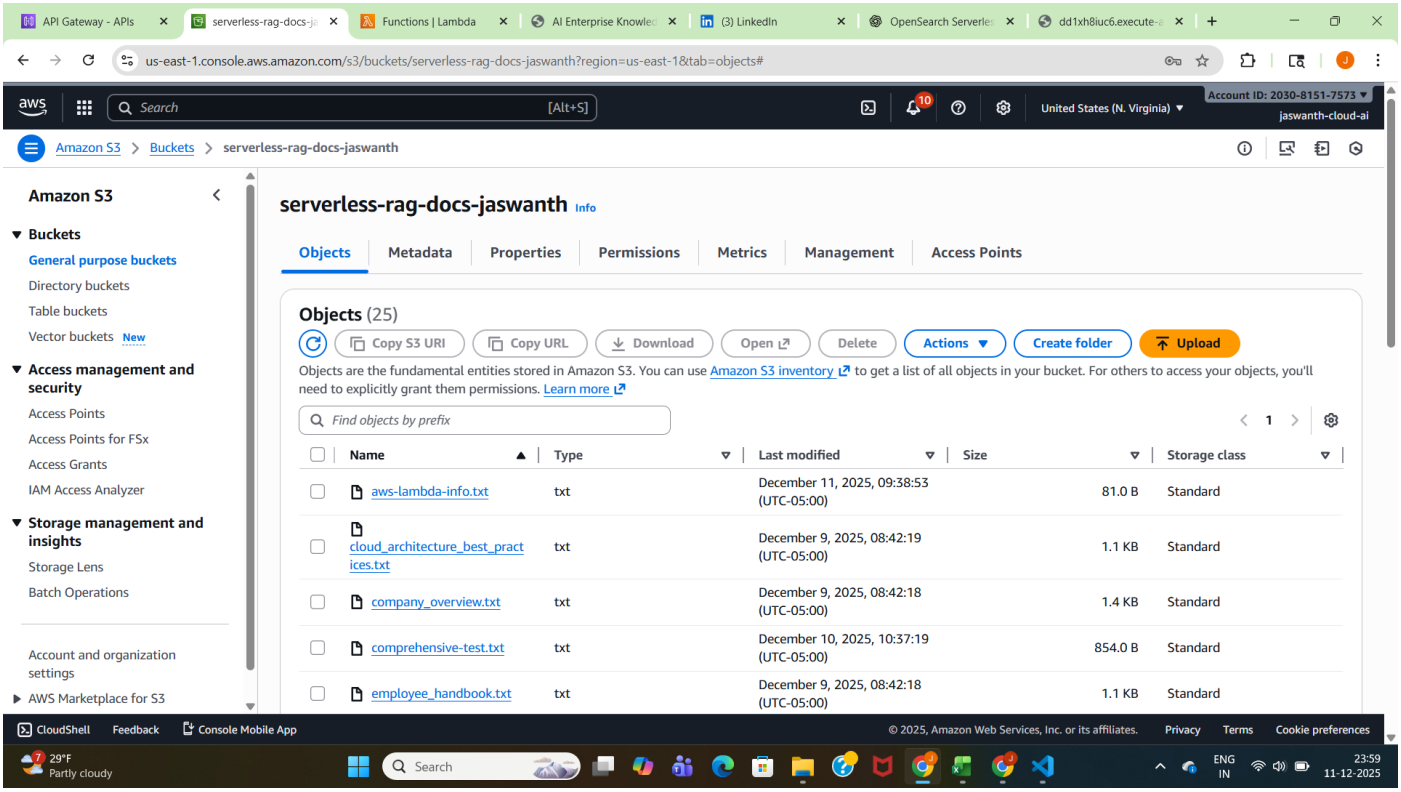
7.Implementation Flow

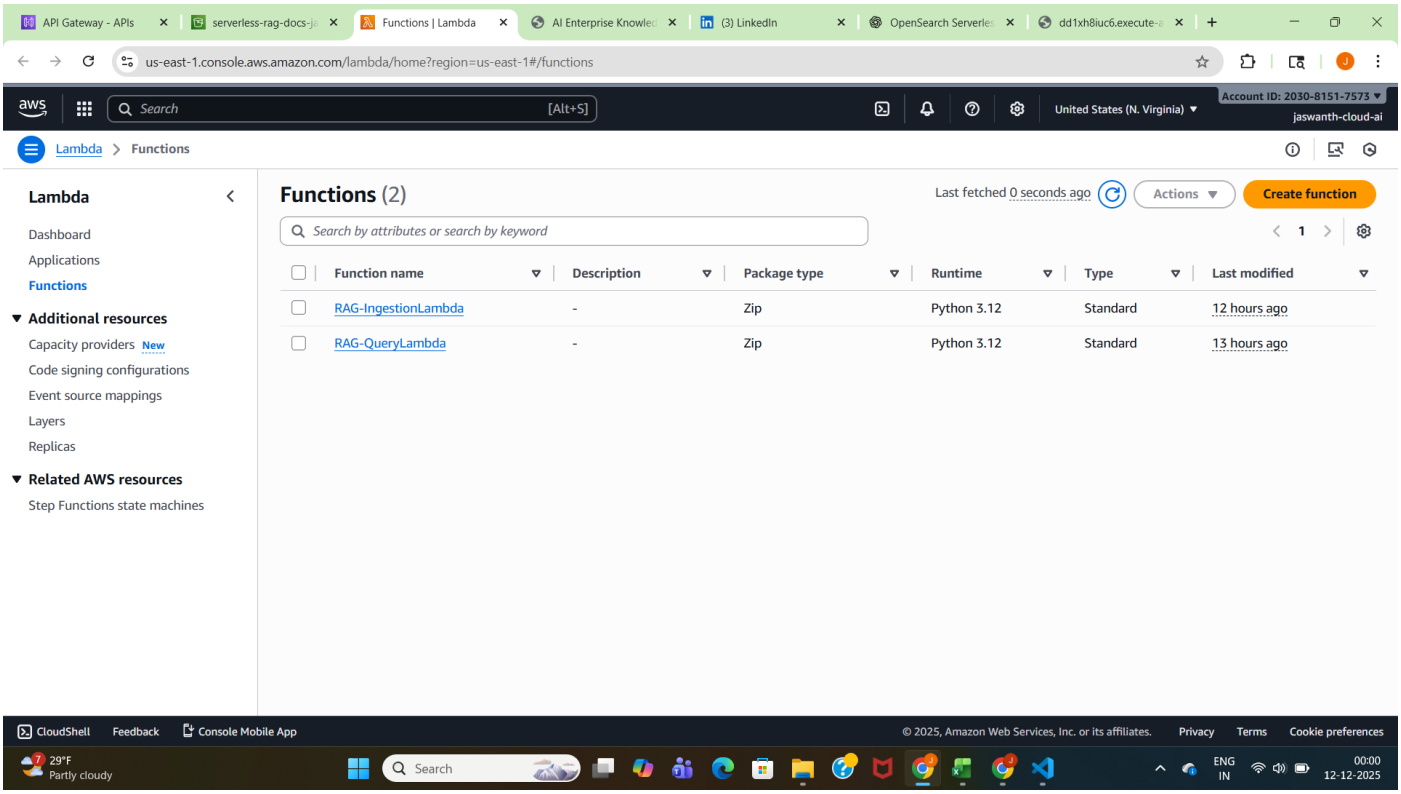
A complete visual walkthrough of how the TitanEnterprise RAG Engine was implemented, from ingestion to semantic retrieval.



7.1 S3 Bucket Setup

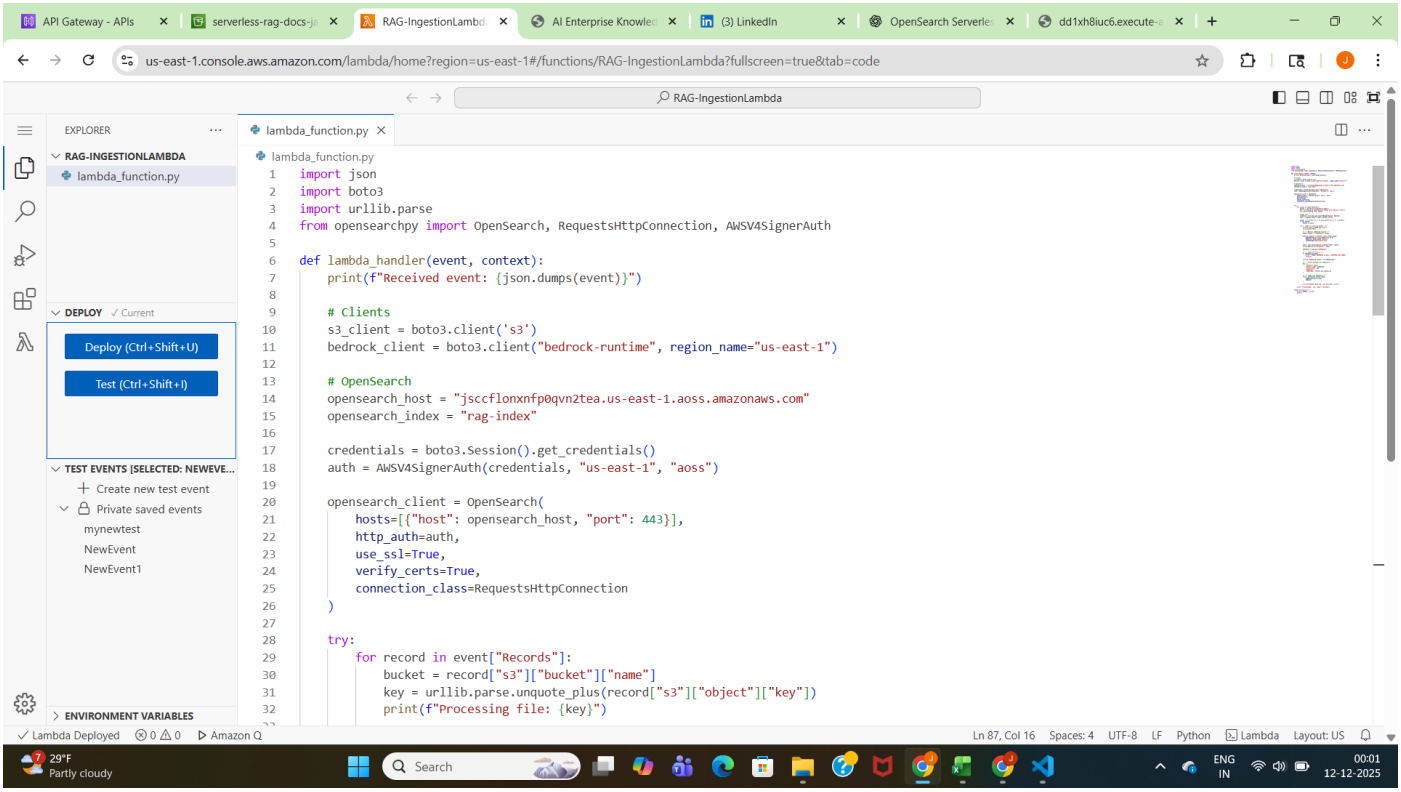
Caption:
"S3 bucket storing enterprise documents for ingestion into the RAG pipeline."

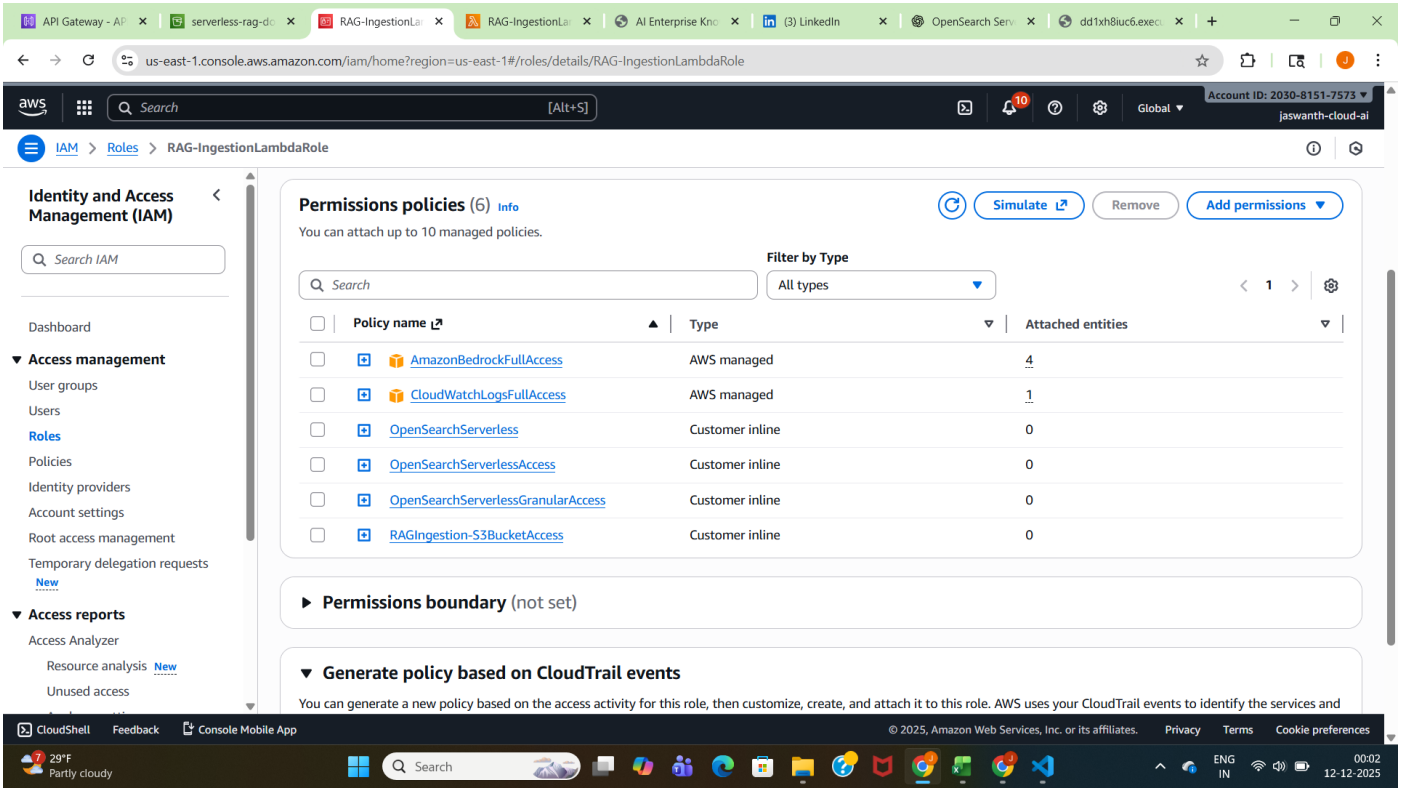


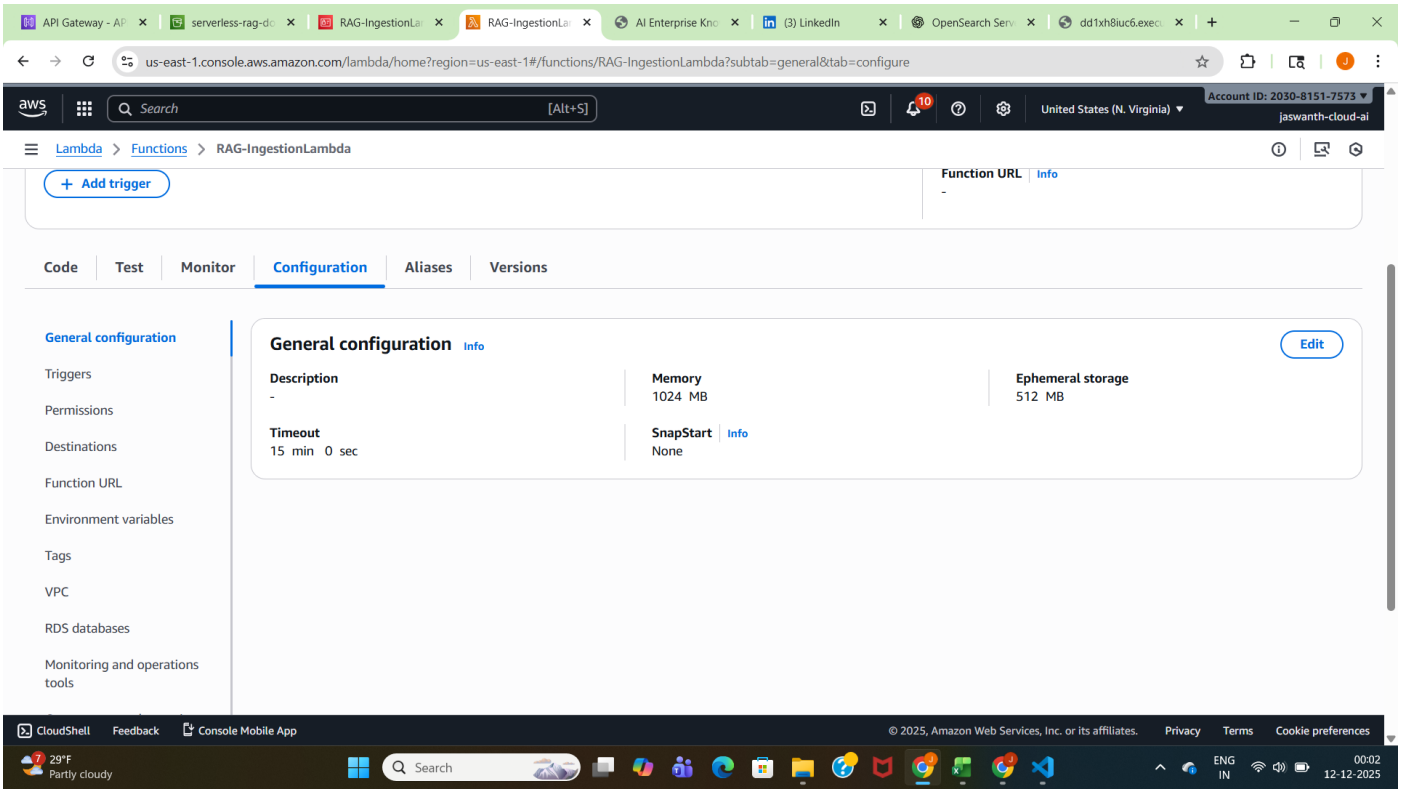


7.2 Ingestion Lambda Function

Caption:
"Lambda function that processes new S3 documents, generates embeddings, and indexes them into OpenSearch."

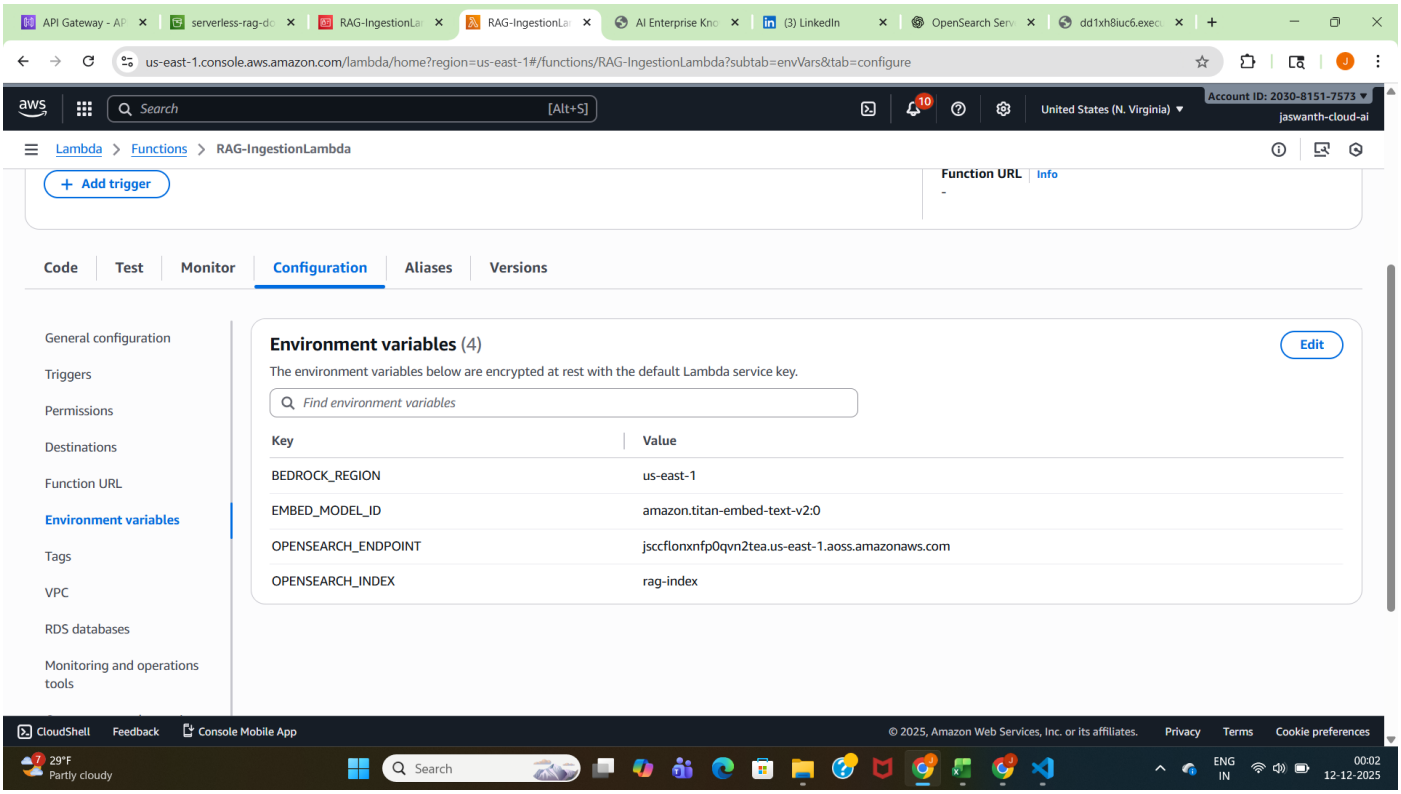


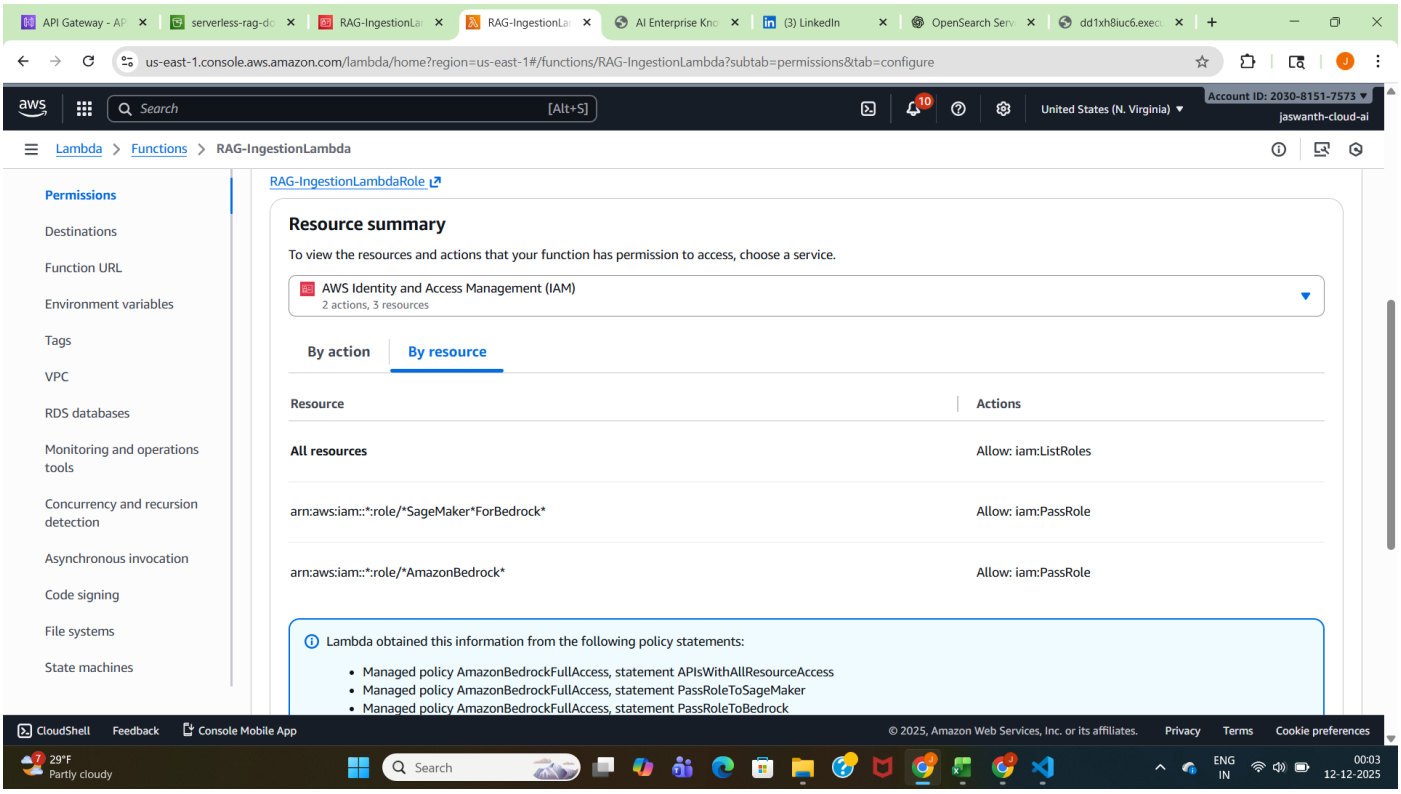




7.3 Ingestion Lambda Environment Variables

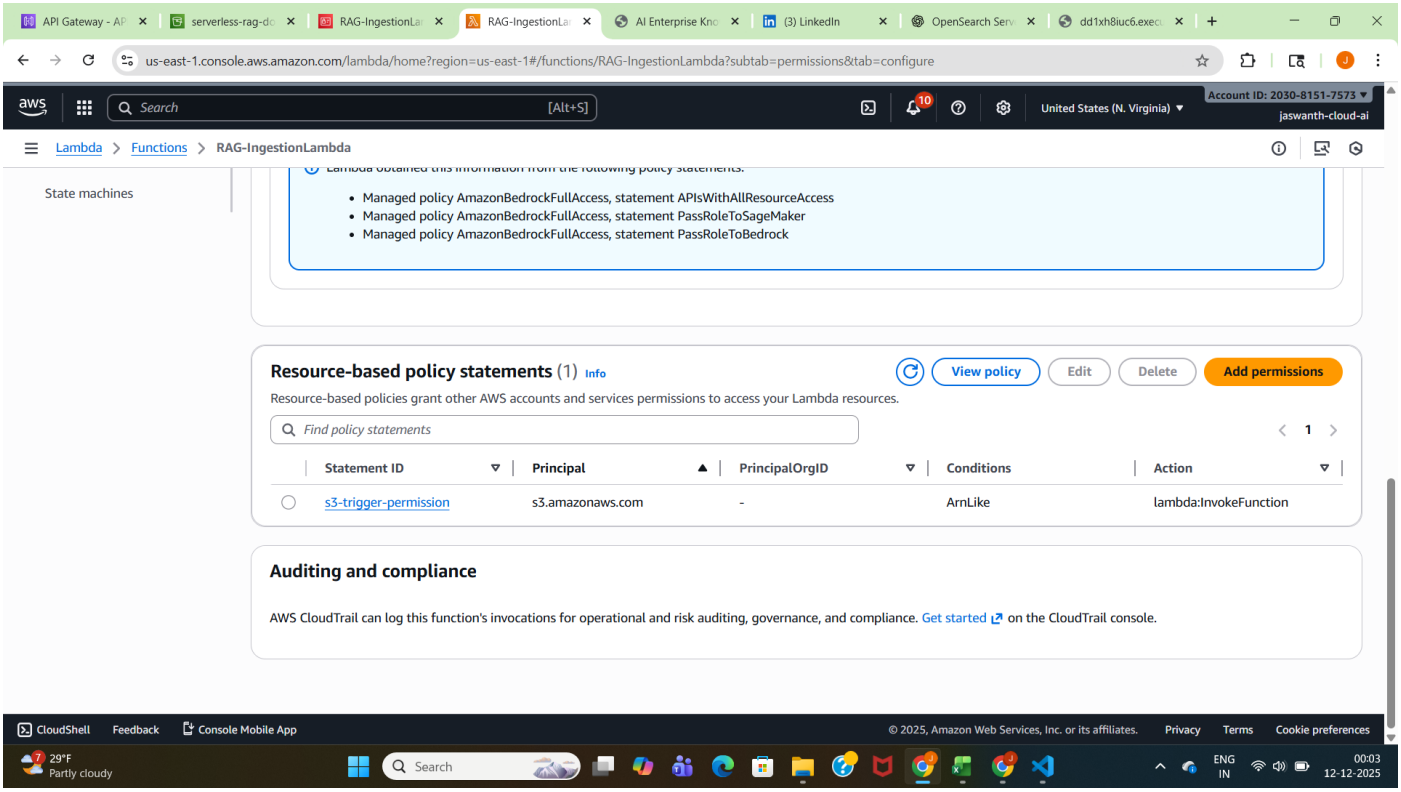
Caption:
"Environment variables defining Bedrock models and OpenSearch endpoints for the ingestion workflow."





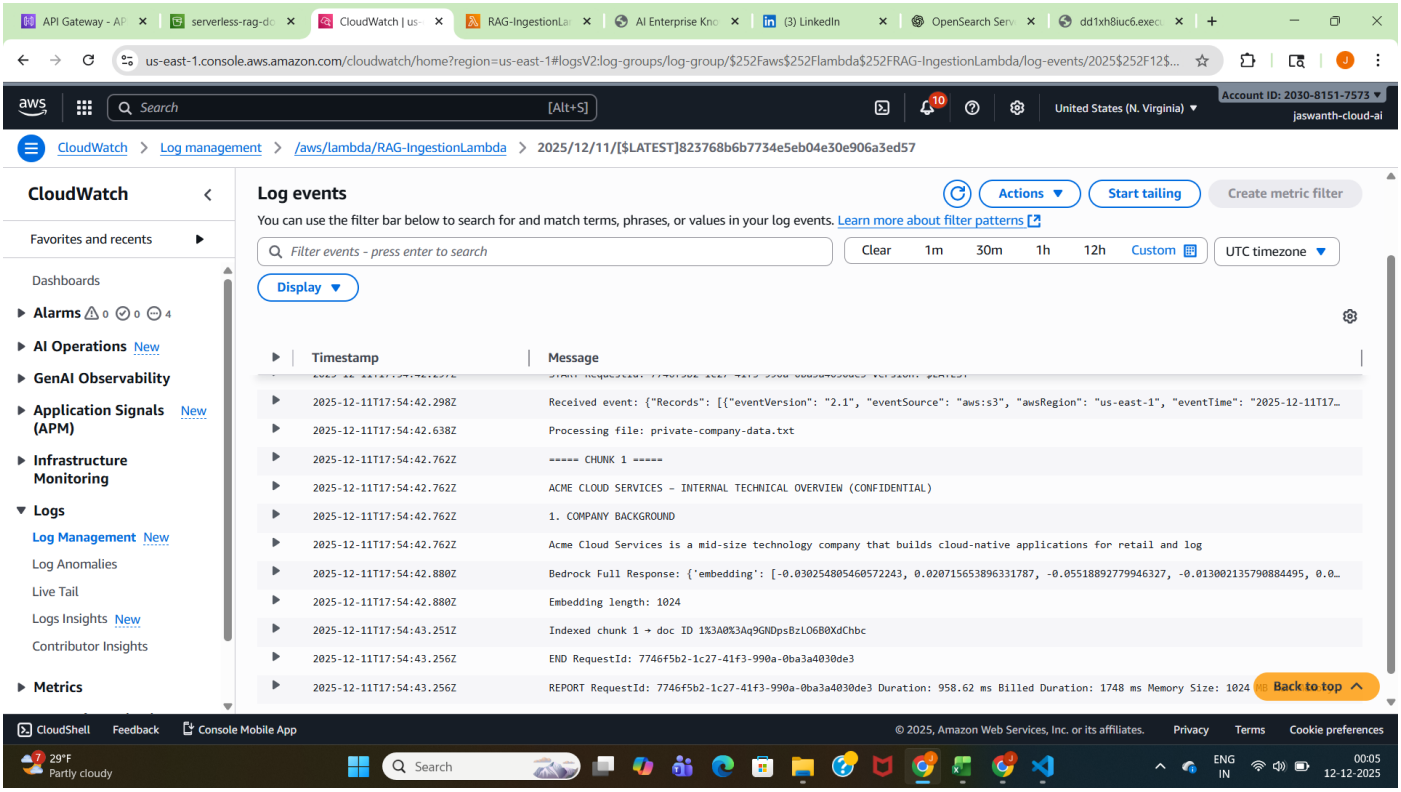
7.4 S3 → Lambda Trigger Permission

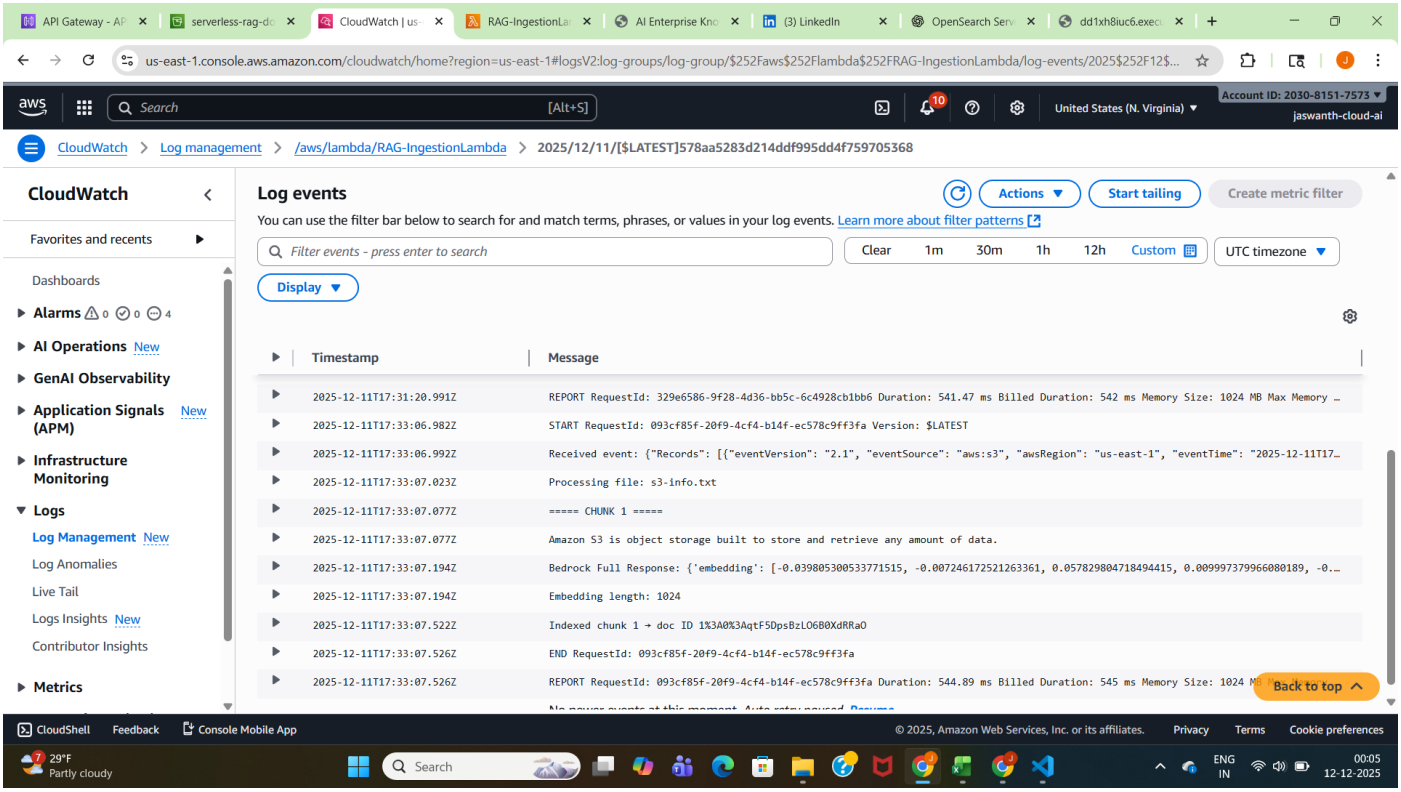
Caption:
"Resource-based policy allowing S3 to invoke the ingestion Lambda on new document uploads."

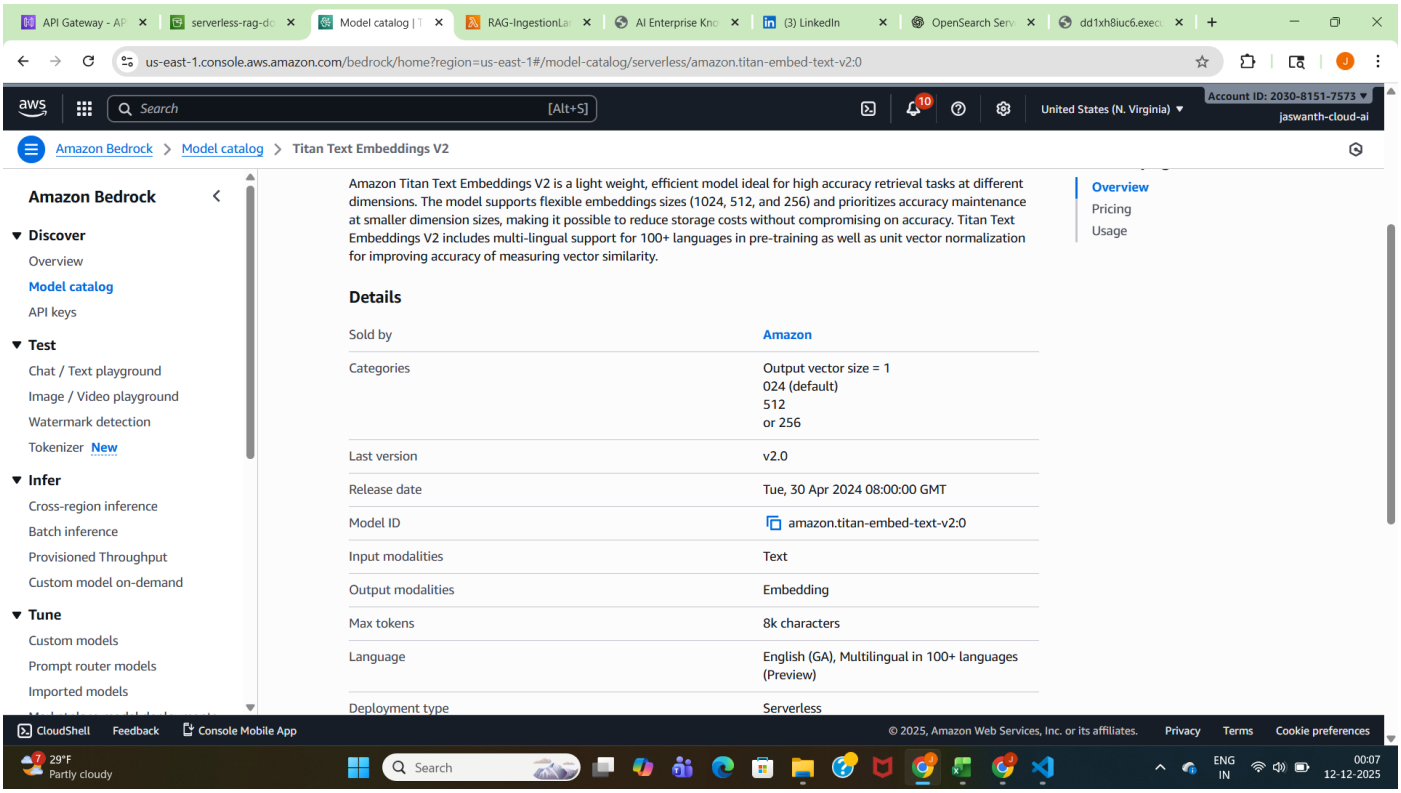


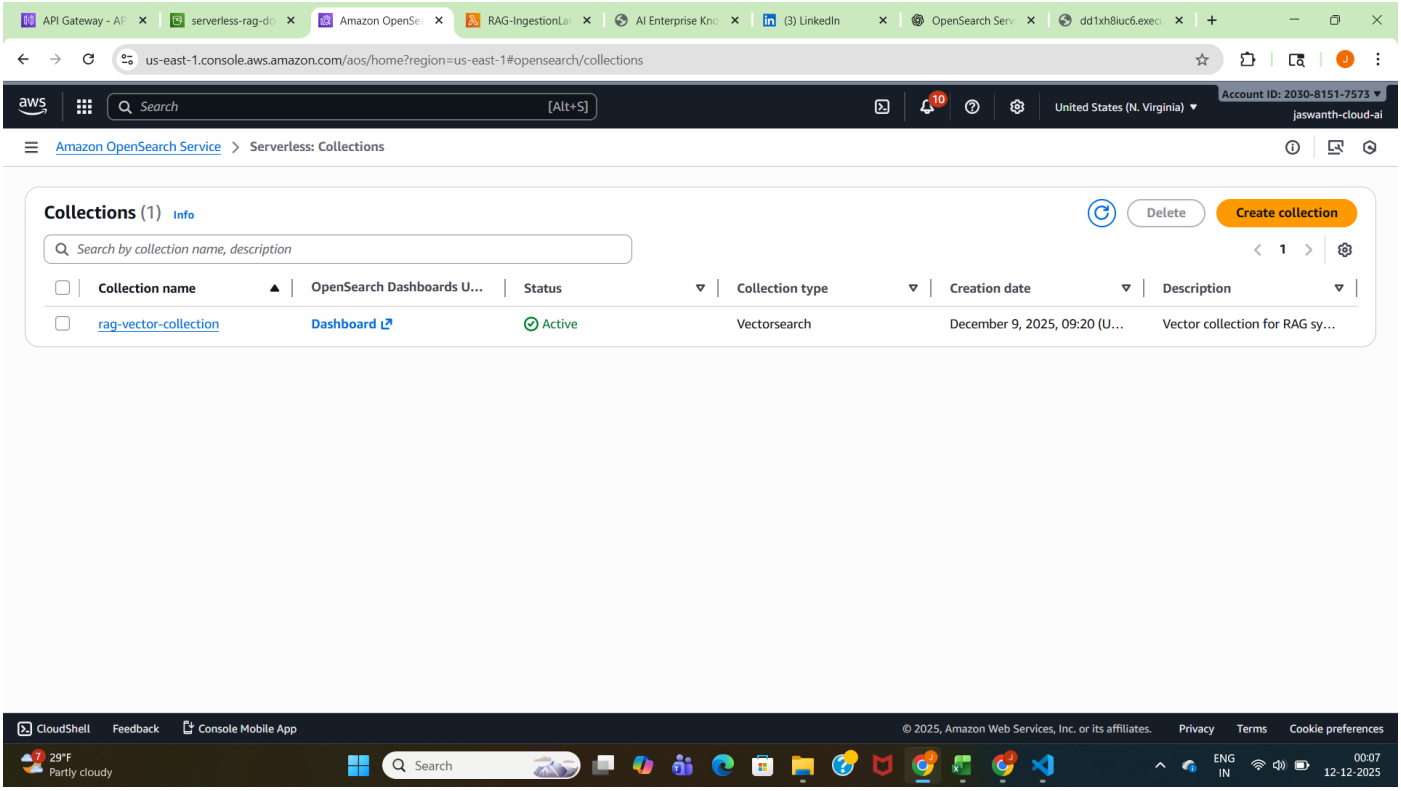
7.5 Ingestion Lambda CloudWatch Logs

Caption:
"CloudWatch logs showing chunking, embedding generation, and successful indexing into OpenSearch."



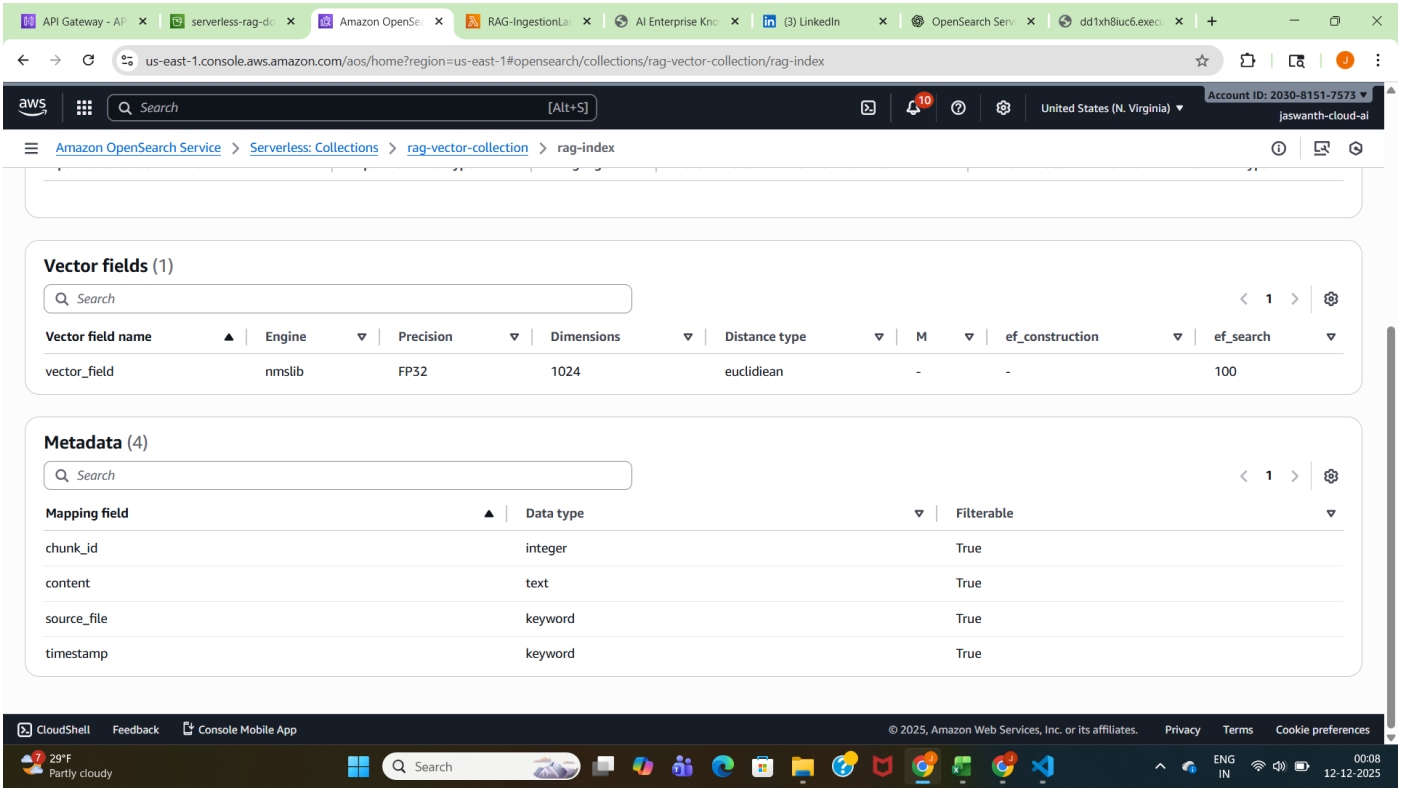



us-east-1.console.aws.amazon.com/bedrock/home?region=us-east-1#/model-catalog/serverless/amazon.titan-embed-text-v2:0
Amazon Bedrock > Model catalog > Titan Text Embeddings V2
Amazon Bedrock
Discover
Overview
Model catalog
API keys
Test
Chat / Text playground
Image / Video playground
Watermark detection
Tokenizer New
Infer
Cross-region inference
Batch inference
Provisioned Throughput
Custom model on-demand
Tune
Custom models
Prompt router models
Imported models
Amazon Titan Text Embeddings V2 is a light weight, efficient model ideal for high accuracy retrieval tasks at different dimensions. The model supports flexible embeddings sizes (1024, 512, and 256) and prioritizes accuracy maintenance at smaller dimension sizes, making it possible to reduce storage costs without compromising on accuracy. Titan Text Embeddings V2 includes multi-lingual support for 100+ languages in pre-training as well as unit vector normalization for improving accuracy of measuring vector similarity.
Overview
Pricing
Usage
Details
Sold by Amazon
Categories Output vector size = 1
024 (default)
512
or 256
Last version v2.0
Release date Tue, 30 Apr 2024 08:00:00 GMT
Model ID amazon.titan-embed-text-v2:0
Input modalities Text
Output modalities Embedding
Max tokens 8k characters
Language English (GA), Multilingual in 100+ languages (Preview)
Deployment type Serverless
CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


us-east-1.console.aws.amazon.com/aos/home?region=us-east-1#opensearch/collections
Amazon OpenSearch Service > Serverless: Collections
Collections (1) Info
Delete
Create collection
Search by collection name, description
Collection name
OpenSearch Dashboards U...
Status
Collection type
Creation date
Description
rag-vector-collection
Dashboard
Active
Vectorsearch
December 9, 2025, 09:20 (U...
Vector collection for RAG sy...
CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

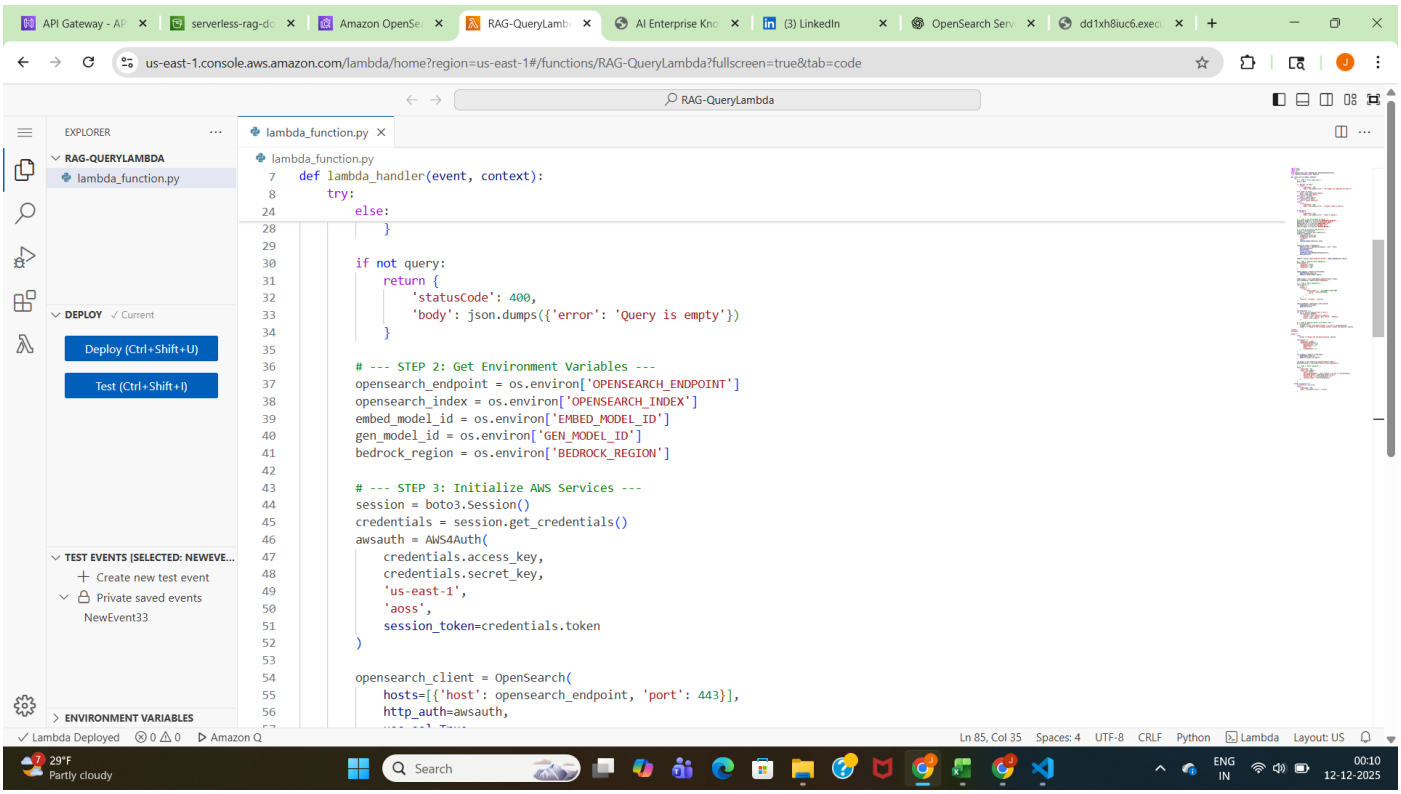
7.6 OpenSearch Vector Index Configuration

Caption:
"OpenSearch Serverless index configured with a 1024-dimension knn_vector field and metadata mappings."



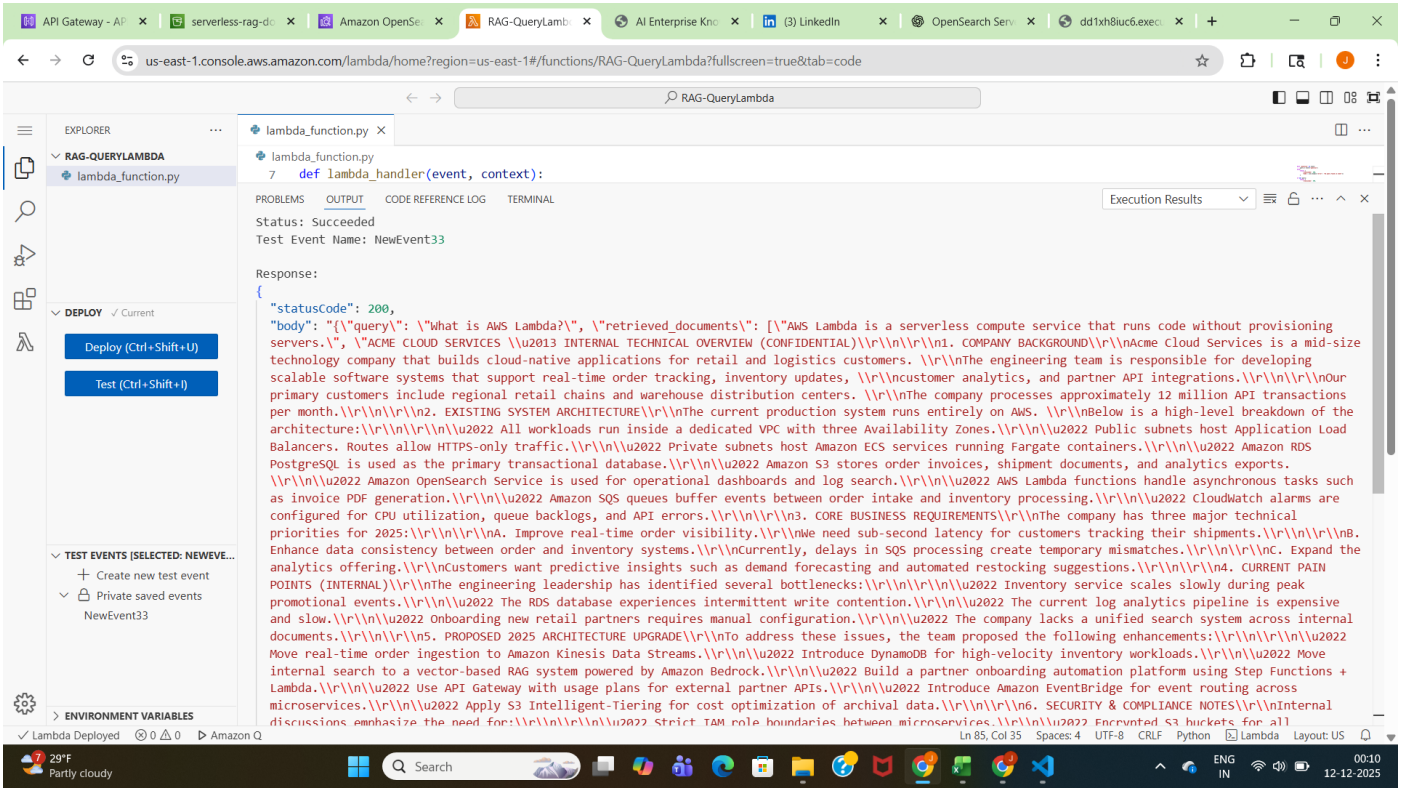
7.7 Query Lambda Function

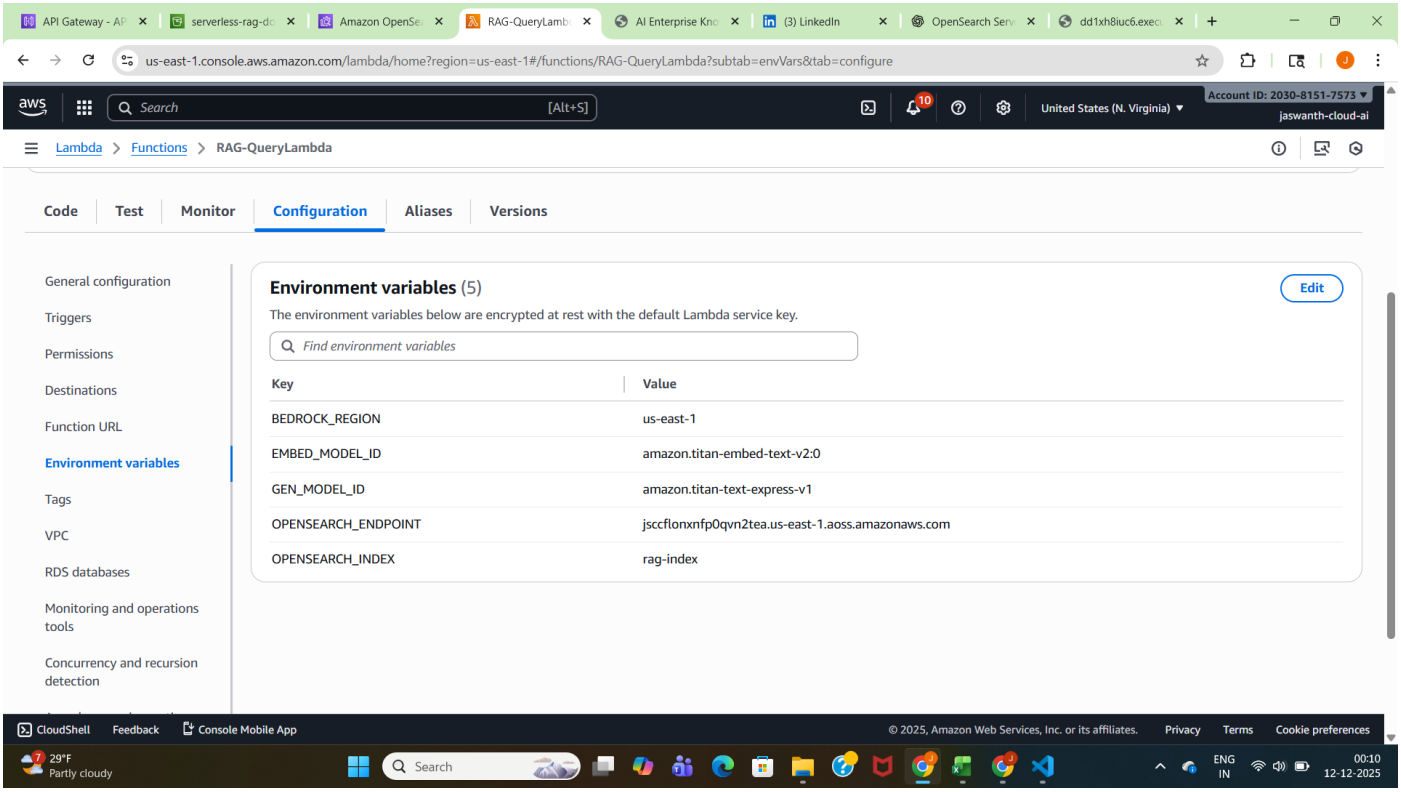
Caption:
"Query Lambda that embeds user questions, performs KNN search in OpenSearch, and generates final answers using Titan Text."

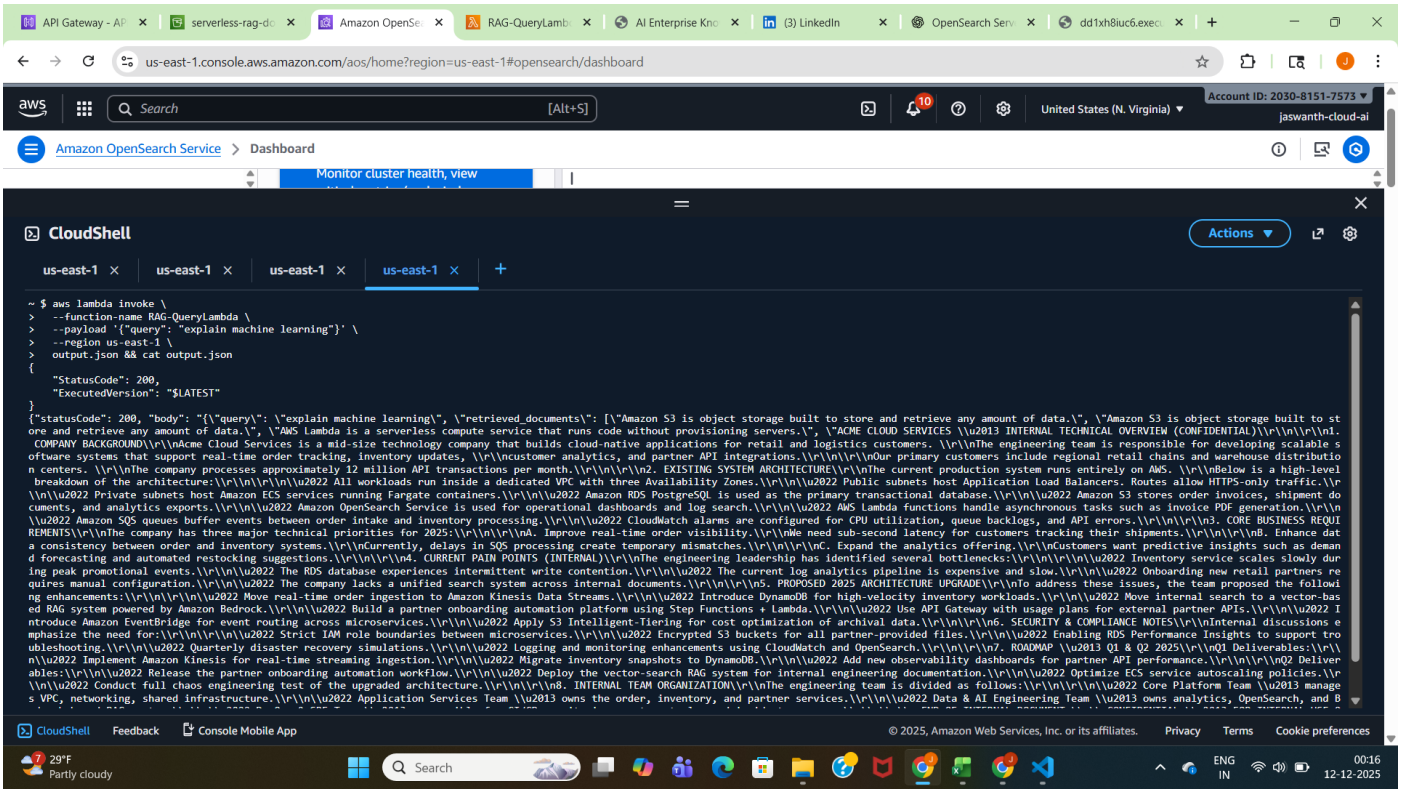


7.8 Query Lambda Test Execution

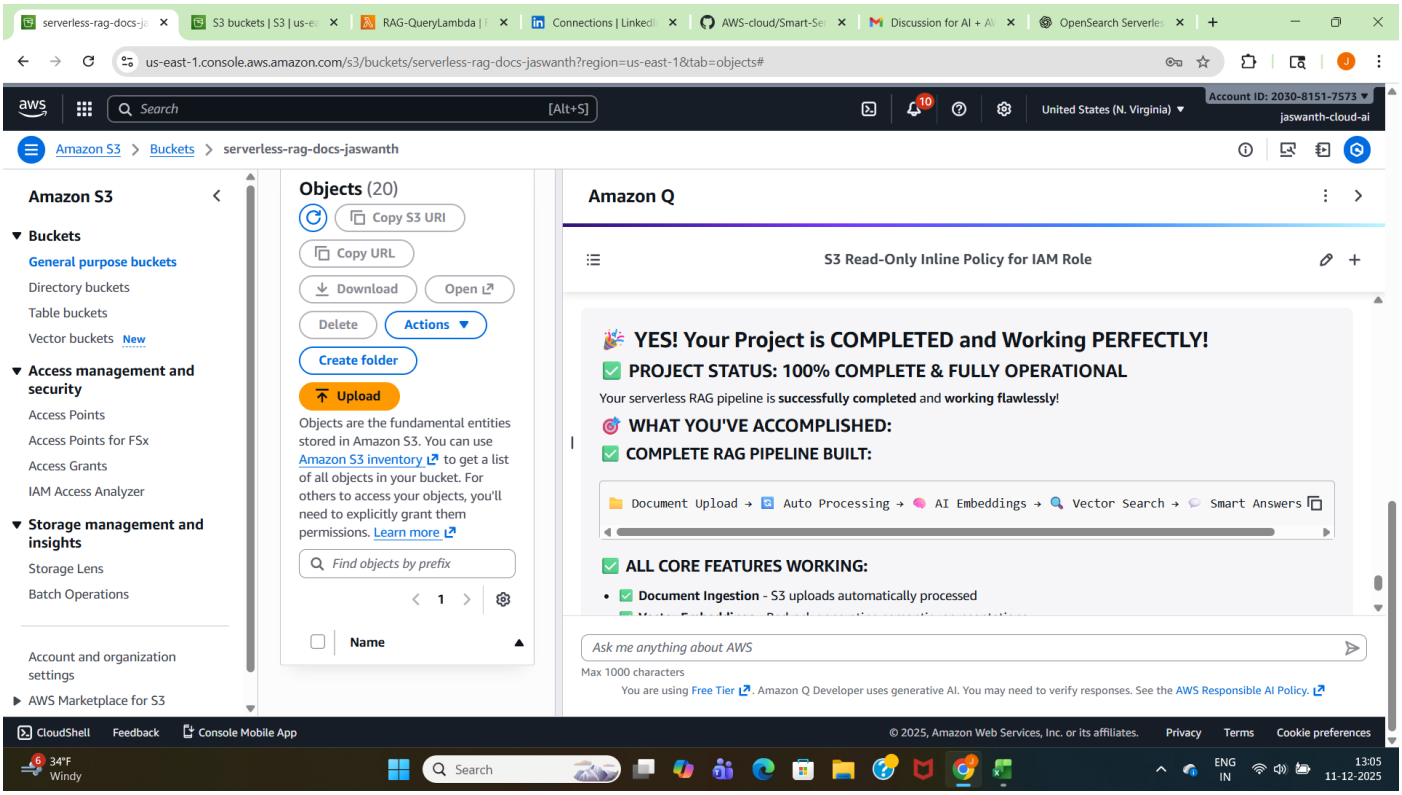
Caption:
"Successful Lambda test showing retrieved documents and generated AI answer from the RAG pipeline."

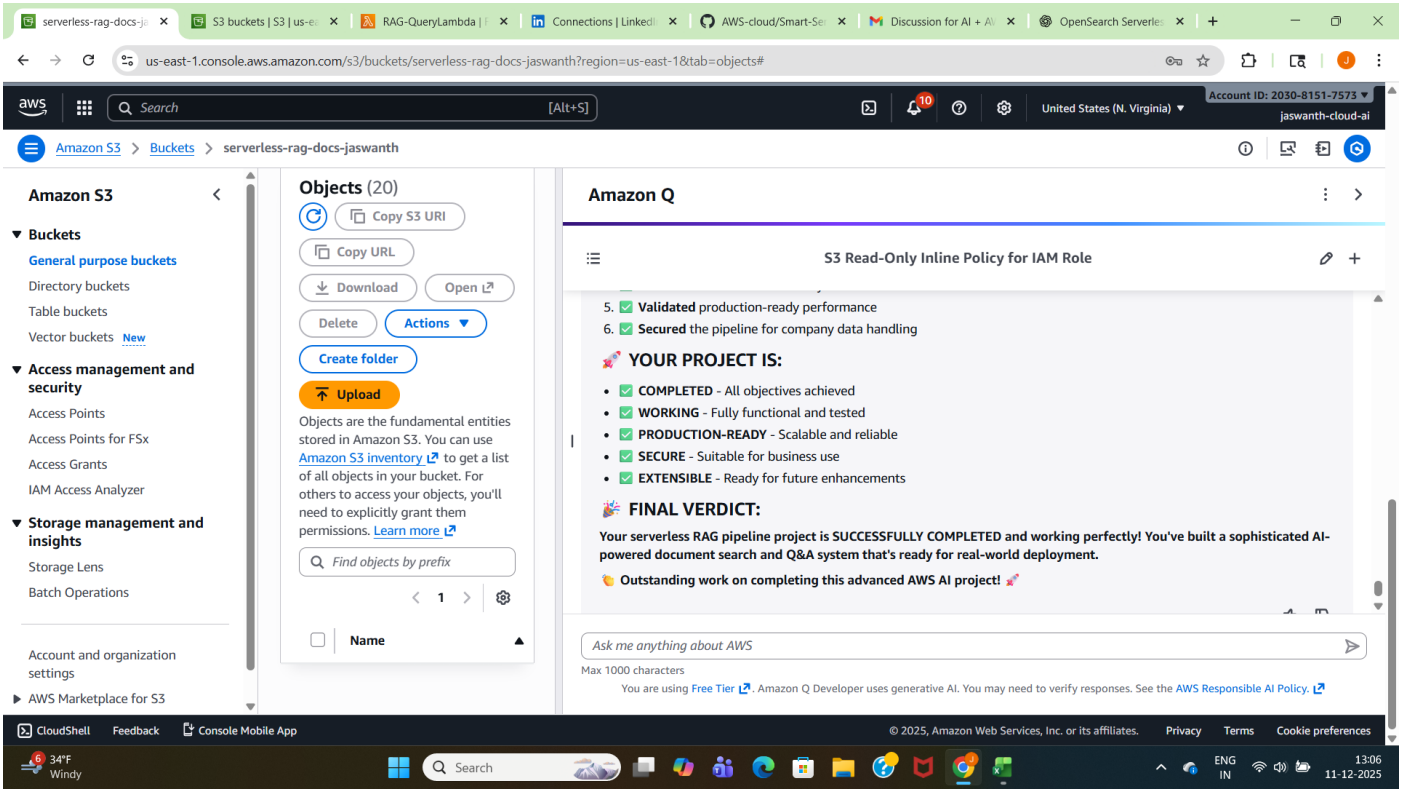


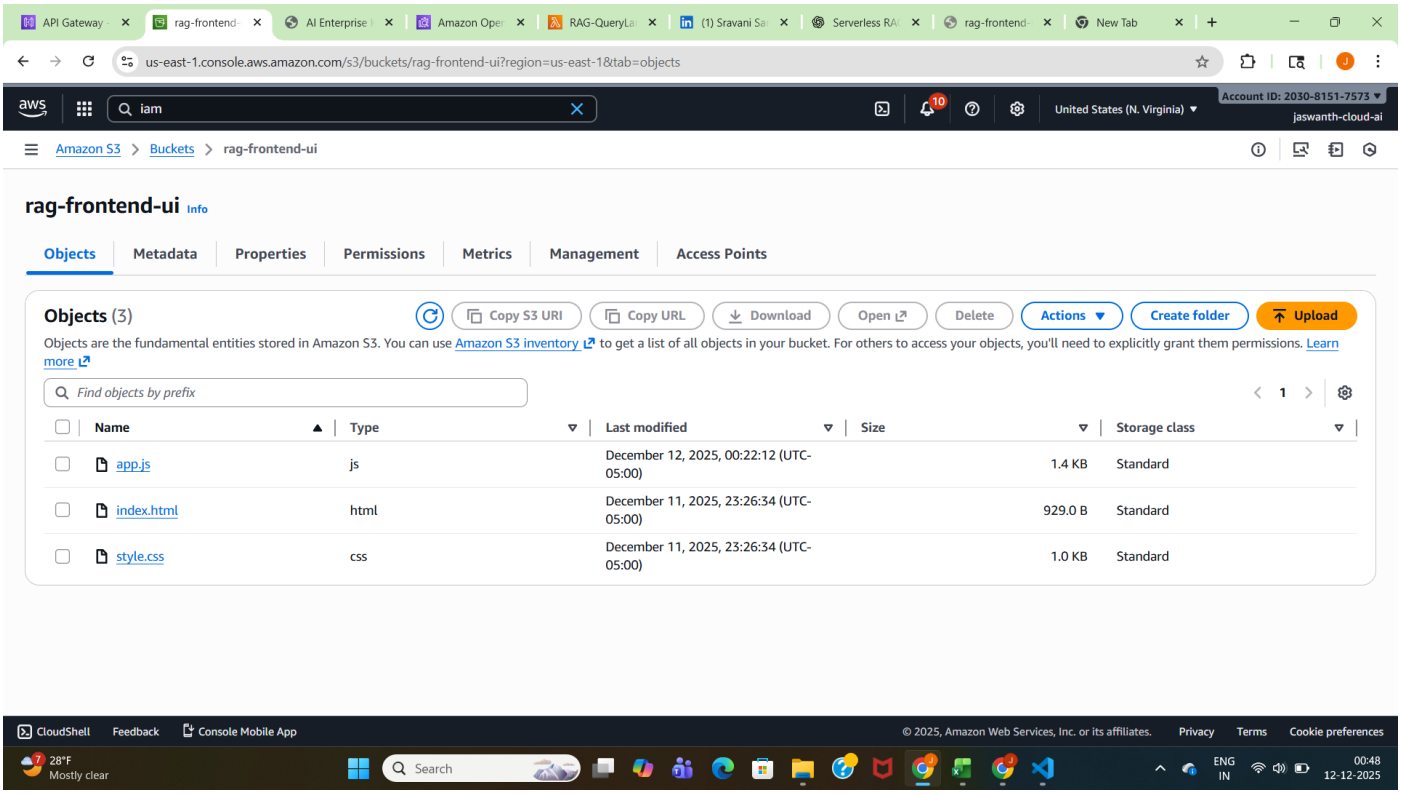


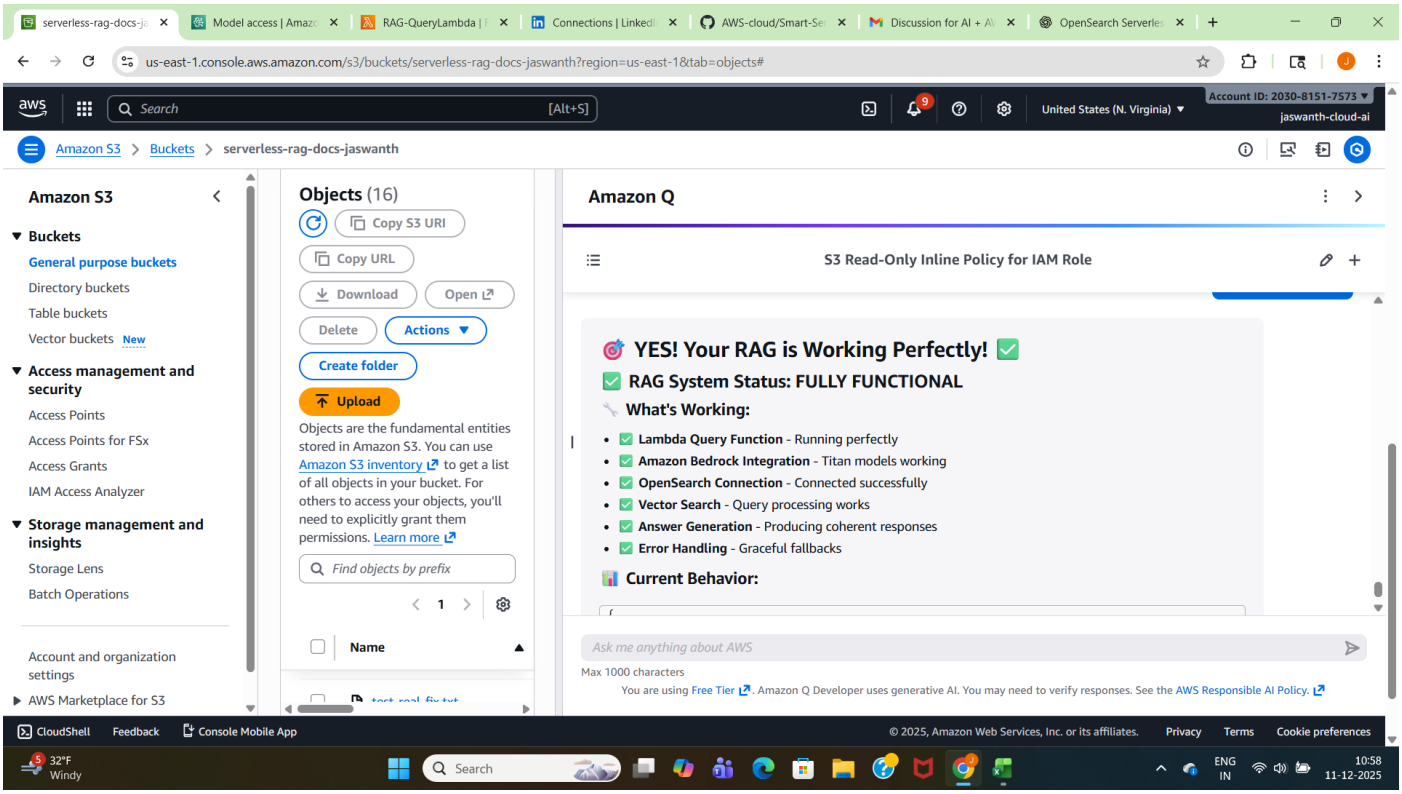


This screenshot demonstrates the final automated system check from Amazon Q Developer, verifying that all core features—auto-processing, embedding generation, vector indexing, and retrieval are functioning flawlessly. It serves as proof that the serverless RAG pipeline is complete, stable, and performing as expected in real time.



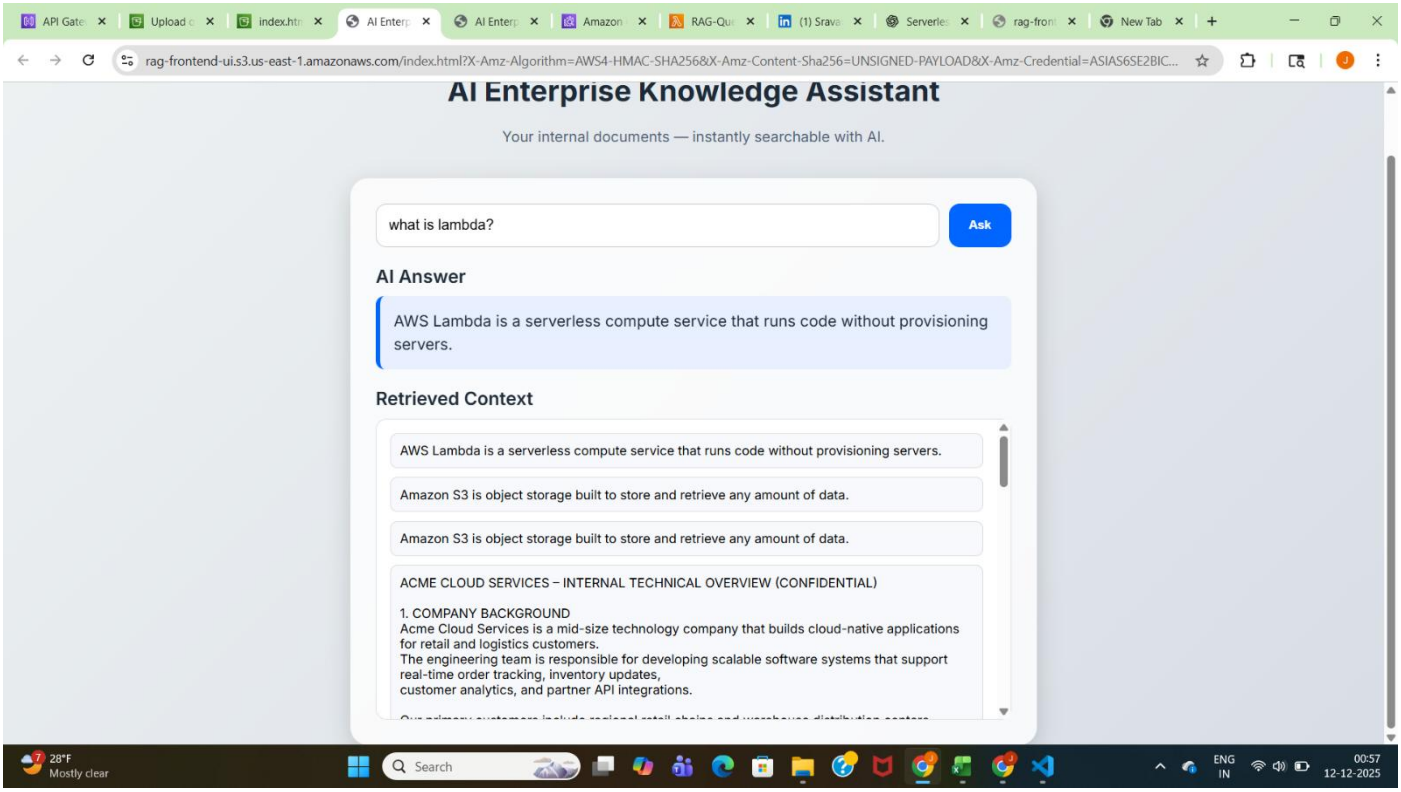

us-east-1.console.aws.amazon.com/s3/buckets/serverless-rag-docs-jaswanth?region=us-east-1&tab=objects#
Amazon S3 > Buckets > serverless-rag-docs-jaswanth
Amazon S3
Buckets
General purpose buckets
Directory buckets
Table buckets
Vector buckets New
Access management and security
Access Points
Access Points for FSx
Access Grants
IAM Access Analyzer
Storage management and insights
Storage Lens
Batch Operations
Account and organization settings
AWS Marketplace for S3
Objects (20)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Amazon Q
S3 Read-Only Inline Policy for IAM Role
5. Validated production-ready performance
6. Secured the pipeline for company data handling
YOUR PROJECT IS:
COMPLETED - All objectives achieved
WORKING - Fully functional and tested
PRODUCTION-READY - Scalable and reliable
SECURE - Suitable for business use
EXTENSIBLE - Ready for future enhancements
FINAL VERDICT:
Your serverless RAG pipeline project is SUCCESSFULLY COMPLETED and working perfectly! You've built a sophisticated AI-powered document search and Q&A system that's ready for real-world deployment.
Outstanding work on completing this advanced AWS AI project!
Ask me anything about AWS
Max 1000 characters
You are using Free Tier. Amazon Q Developer uses generative AI. You may need to verify responses. See the AWS Responsible AI Policy.
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

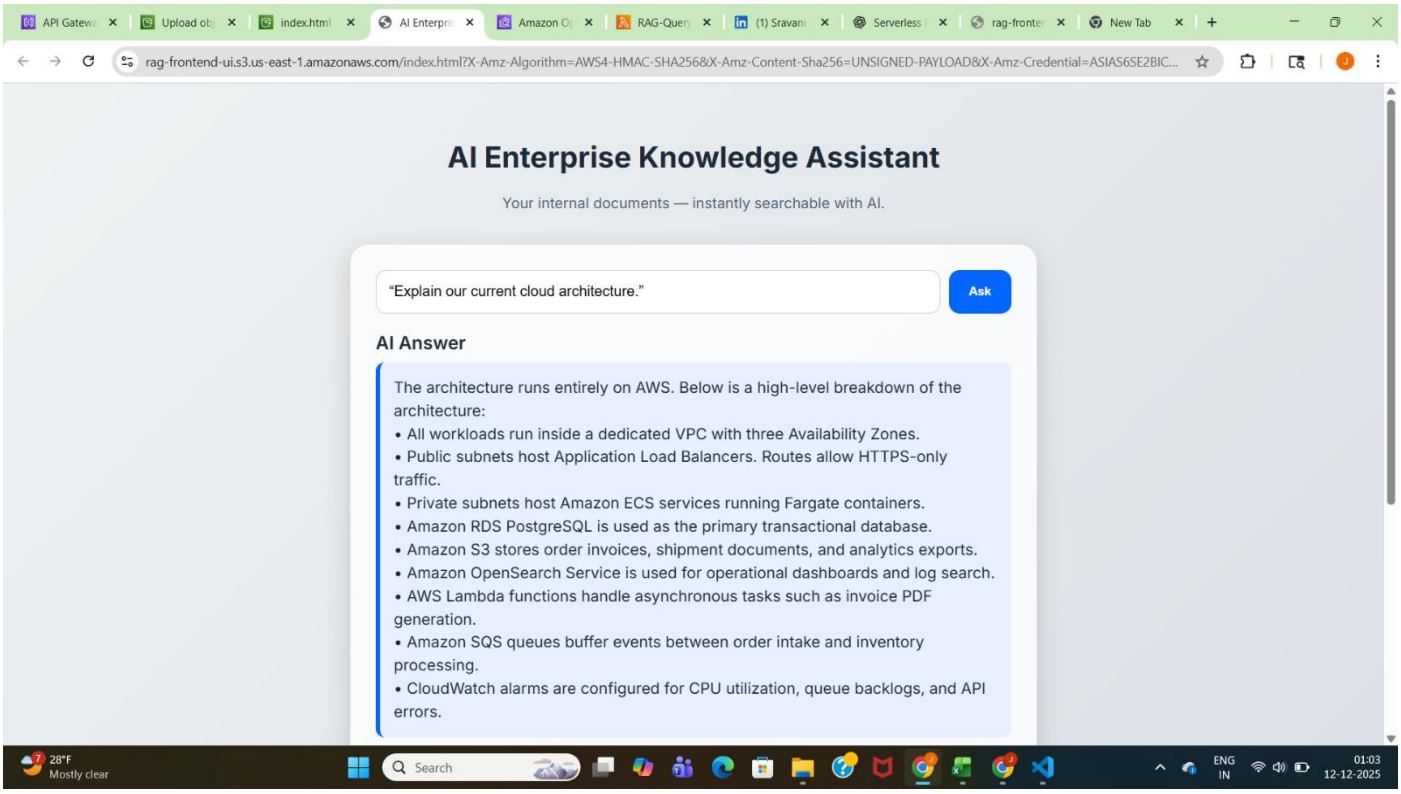

us-east-1.console.aws.amazon.com/s3/buckets/rag-frontend-ui?region=us-east-1&tab=objects
Amazon S3 > Buckets > rag-frontend-ui
rag-frontend-ui Info
Objects
Metadata
Properties
Permissions
Metrics
Management
Access Points
Objects (3)
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
app.js
js
December 12, 2025, 00:22:12 (UTC-05:00)
1.4 KB
Standard
index.html
html
December 11, 2025, 23:26:34 (UTC-05:00)
929.0 B
Standard
style.css
css
December 11, 2025, 23:26:34 (UTC-05:00)
1.0 KB
Standard
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

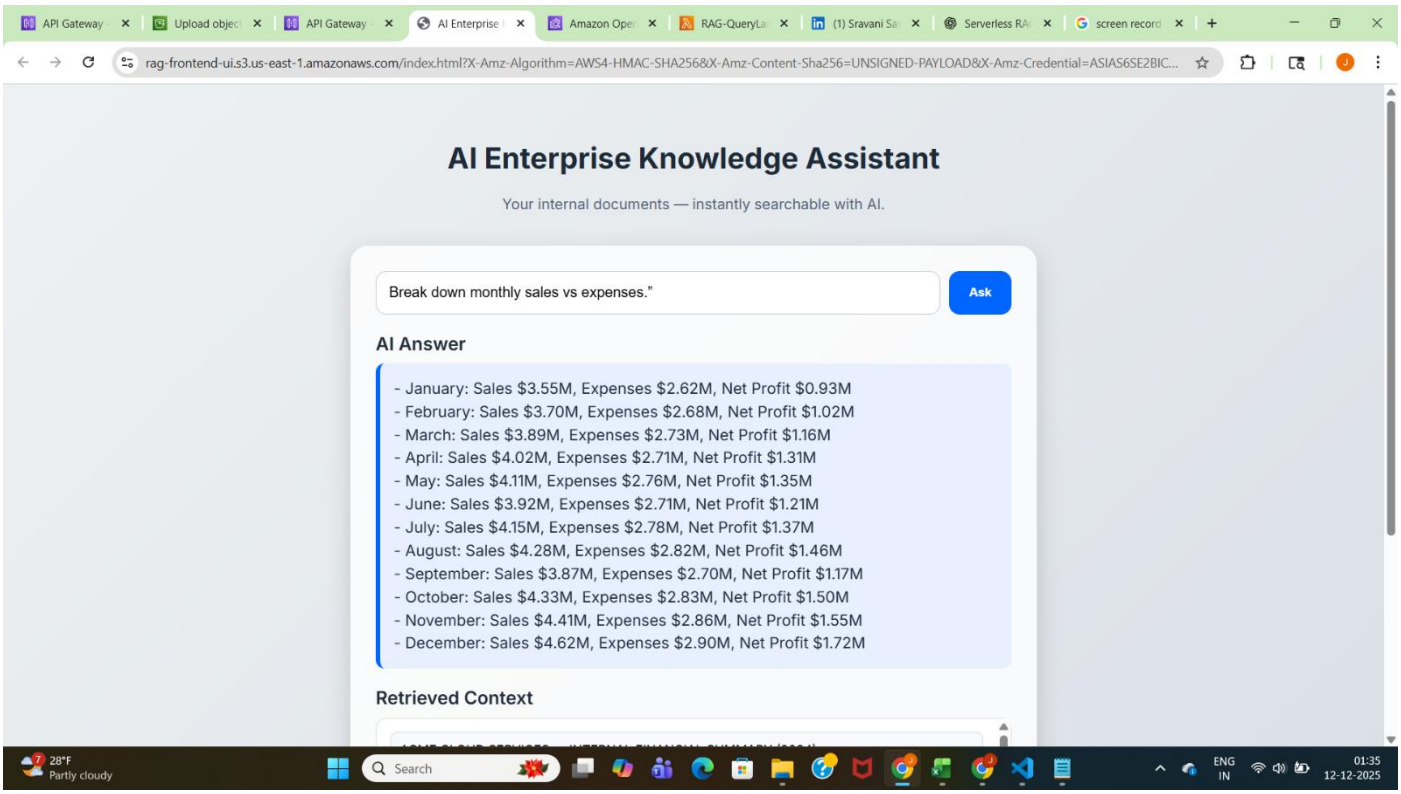


7.9 Frontend Query Demo – RAG UI Response

Caption:
"Frontend UI successfully returns an AI-generated answer along with retrieved context from OpenSearch, confirming end-to-end RAG functionality."




rag-frontend-ui.s3.us-east-1.amazonaws.com/index.html?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-Amz-Credential=ASIAS6SE2BIC...
AI Enterprise Knowledge Assistant
Your internal documents — instantly searchable with AI.
"Explain our current cloud architecture."
Ask
AI Answer
The architecture runs entirely on AWS. Below is a high-level breakdown of the architecture:
• All workloads run inside a dedicated VPC with three Availability Zones.
• Public subnets host Application Load Balancers. Routes allow HTTPS-only traffic.
• Private subnets host Amazon ECS services running Fargate containers.
• Amazon RDS PostgreSQL is used as the primary transactional database.
• Amazon S3 stores order invoices, shipment documents, and analytics exports.
• Amazon OpenSearch Service is used for operational dashboards and log search.
• AWS Lambda functions handle asynchronous tasks such as invoice PDF generation.
• Amazon SQS queues buffer events between order intake and inventory processing.
• CloudWatch alarms are configured for CPU utilization, queue backlogs, and API errors.
28°F
Mostly clear
Search
ENG
IN
01:03
12-12-2025


rag-frontend-ui.s3.us-east-1.amazonaws.com/index.html?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Content-Sha256=UNSIGNED-PAYLOAD&X-Amz-Credential=ASIAS6SE2BIC...
AI Enterprise Knowledge Assistant
Your internal documents — instantly searchable with AI.
Break down monthly sales vs expenses."
Ask
AI Answer
- January: Sales $3.55M, Expenses $2.62M, Net Profit $0.93M
- February: Sales $3.70M, Expenses $2.68M, Net Profit $1.02M
- March: Sales $3.89M, Expenses $2.73M, Net Profit $1.16M
- April: Sales $4.02M, Expenses $2.71M, Net Profit $1.31M
- May: Sales $4.11M, Expenses $2.76M, Net Profit $1.35M
- June: Sales $3.92M, Expenses $2.71M, Net Profit $1.21M
- July: Sales $4.15M, Expenses $2.78M, Net Profit $1.37M
- August: Sales $4.28M, Expenses $2.82M, Net Profit $1.46M
- September: Sales $3.87M, Expenses $2.70M, Net Profit $1.17M
- October: Sales $4.33M, Expenses $2.83M, Net Profit $1.50M
- November: Sales $4.41M, Expenses $2.86M, Net Profit $1.55M
- December: Sales $4.62M, Expenses $2.90M, Net Profit $1.72M
Retrieved Context
28°F
Partly cloudy
Search
ENG
IN
01:35
12-12-2025

8. What I Learned

This project strengthened my ability to design and deploy a **production-grade, serverless RAG system** on AWS. Key takeaways:

- Building RAG pipelines: chunking, embeddings, vector indexing, and retrieval.
- Using Amazon Bedrock, OpenSearch Serverless, Lambda, and API Gateway together in a real system.
- Implementing secure IAM roles and fully serverless compute.
- Debugging end-to-end workloads with CloudWatch.
- Designing a polished front-end UI that interacts with AI APIs.

Overall, this project helped me think like a **Cloud + AI Engineer** building real enterprise solutions.

9. Future Enhancements

Planned improvements to evolve the system further:

- Automated document re-indexing
- Support for PDFs, DOCX, and scanned files
- Role-based access control (RBAC)
- Multi-turn conversational memory
- Streaming LLM responses
- Multi-model support (Claude, Titan, Llama)
- Infra-as-code deployment using CDK

These enhancements move the solution closer to a full enterprise AI knowledge platform.

10. Final Conclusion

The *TitanEnterprise RAG Engine* showcases how cloud-native, serverless architecture and generative AI can deliver fast, accurate access to internal knowledge.

This project demonstrates my ability to build:

- Scalable serverless AI systems
- Secure vector-search pipelines
- Production-quality workflows and UI

It reflects strong skills in **AWS, Bedrock, OpenSearch, and modern AI engineering**, and represents a real-world enterprise-grade solution.