
AWS
re:Invent
NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

ANT210

Improve your search with vector capabilities in OpenSearch Service

Jon Handler
(he/him)
Sr. Principal Solutions Architect
AWS

Achal Kumar
Director – Data Services
Intuit

Aruna Govindaraju
(she/her)
Specialist Solutions Architect
AWS

Artificial Intelligence and Machine Learning boom

Large language models providing good baseline NLP

ChatBots hit the public's awareness

"Semantic" capabilities used for search workloads

Other AI/ML techniques used for search as well

Amazon OpenSearch Service enables working with vector embeddings

Amazon OpenSearch Service

Intelligent search and log analytics solution to help you get the most out of your data

Search: Improve the relevancy of your search results in near real time with cutting edge search innovations

Analytics: Securely, easily, and efficiently visualize and analyze your operational data

Integrations: Quickly and easily connect all of your data for faster, better insights

Cost effective: Optimize time and resources for strategic work

Deployment options: Serverless simplicity or managed control

aws

OpenSearch Service vector capabilities

OpenSearch

- Apache v2 Licensed, open-source project
- 300K+ downloads, 70+ partners
- K-Nearest-Neighbor (kNN) plugin providing FAISS, nmslib, and Lucene, for exact kNN, HNSW, and IVF
- Support for ML-Commons, and Learning to Rank (LTR)
- Neural plugin for simplifying vector usage

Amazon OpenSearch Service

- Fully managed domains
- Connectors to third-party, model-hosting services
- Support for the kNN plugin, Neural plugin, and LTR plugin

Amazon OpenSearch Serverless

- Auto-scaled, serverless experience
- Vector engine for OpenSearch Serverless
- Support for the kNN plugin

OpenSearch Service vector workloads

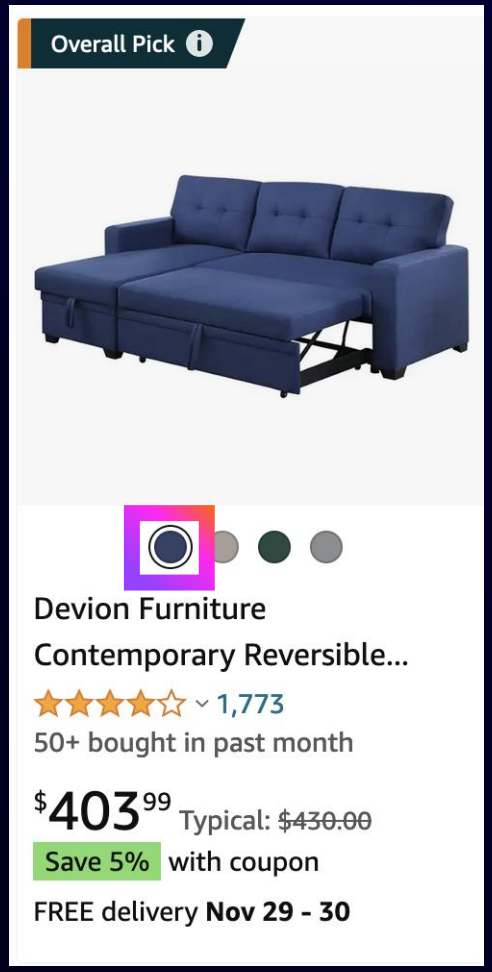
Semantic search

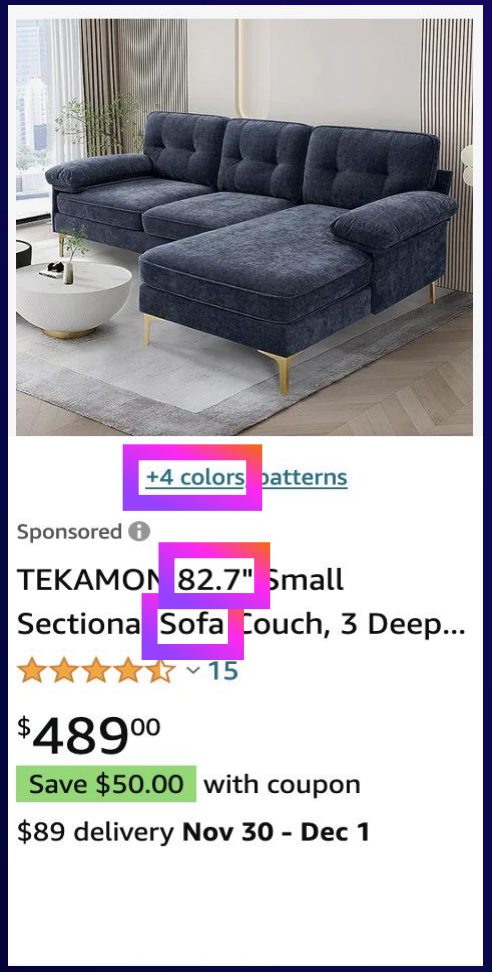
Improve search relevance by storing, matching, and scoring with LLM-generated (or other) vector embeddings. Supports multi-modal, and hybrid search.

Generative AI

Use OpenSearch Service as a component in a generative AI application, as part of the prompt engineering. Can incorporate semantic search for improved prompt data.

Find me an 8-foot, blue sofa





Without vectors, search engines work by matching query terms (words) to document terms

What makes search results better or worse?

Search relevance is the **measure**
of how well the **information retrieved**
meets the user's **information goal**

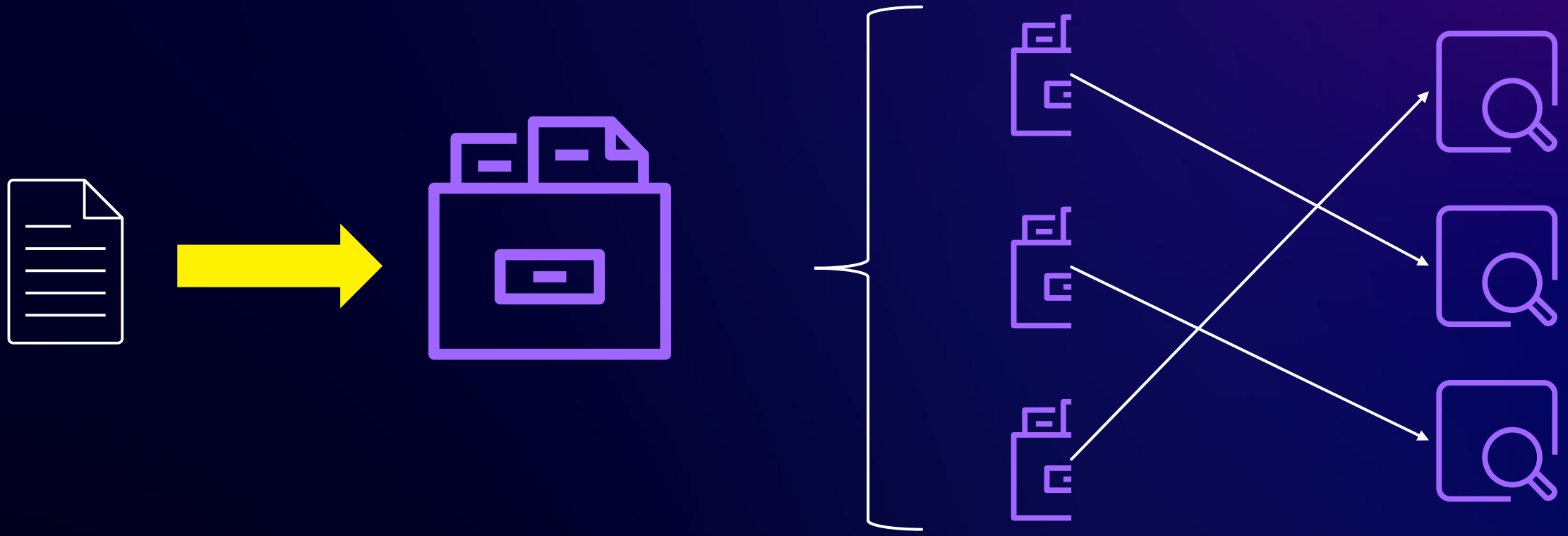
Search basics

OpenSearch Service documents are JSON

https://registry.opendata.aws/amazon-pqa/


```
{
    "asin": "B003YFHQI2",
    "question_id": "Tx3UX24HTBF20DP",
    "brand_name": "Belle Epoque",
    "item_name": "Belle Epoque 420 Full Thread Count Sheet Set with Hemstitch,
    Taupe",
    "question_text": "are these sheets 100% cotton?",
    "answers": [
        {
            "answer_text": "Yes, but the cotton is a bit rough. I've washed
            the sheets about six times and they are finally softening."
        },
        {
            "answer_text": "Yes!"
        }
    ],
    "bullet_point1": "Incredible luxury: Unbelievable soft hand comes from
    Belle Epoque processing formula combined with highest quality compact
    combed cotton yarns",
    "bullet_point2": "Soft mirror finish, Special processing ensures that
    these sheets will be satin-soft and pill-free after many wash and dry
    cycles",
    "bullet_point3": "Oversized sheets, Flat sheet: 1 foot longer than
    standard flat sheet with 6-inch hem at the top and 1-inch mitered hems on
    the sides and foot of the sheet, 6-inch hem features meticulous hemstitch
    for added beauty, Extra length provides additional turn",
    "bullet_point4": "Fitted sheet: Deep pocket fits up to a 20-inch mattress,
    Elastic is sewn in all around for secure fit",
    "bullet_point5": "Pillowcases: Oversized with 6-Inch cuff and hemstitch
    detailing to match flat sheet Full Sheet Set",
    "product_description": "",
    "question_type": "yes-no",
    "answer_aggregated": "yes"
}
```

OpenSearch Service is a distributed database

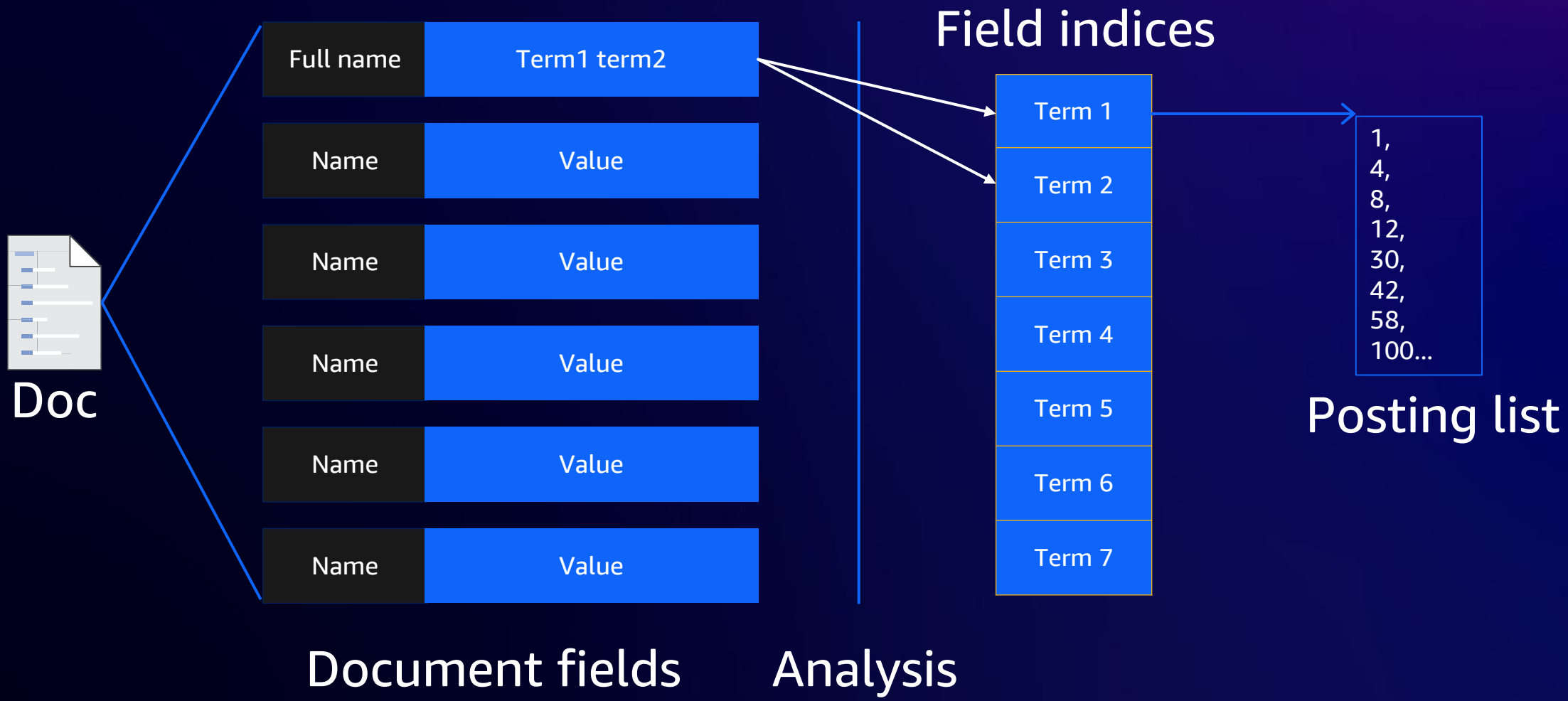


Send documents to an index

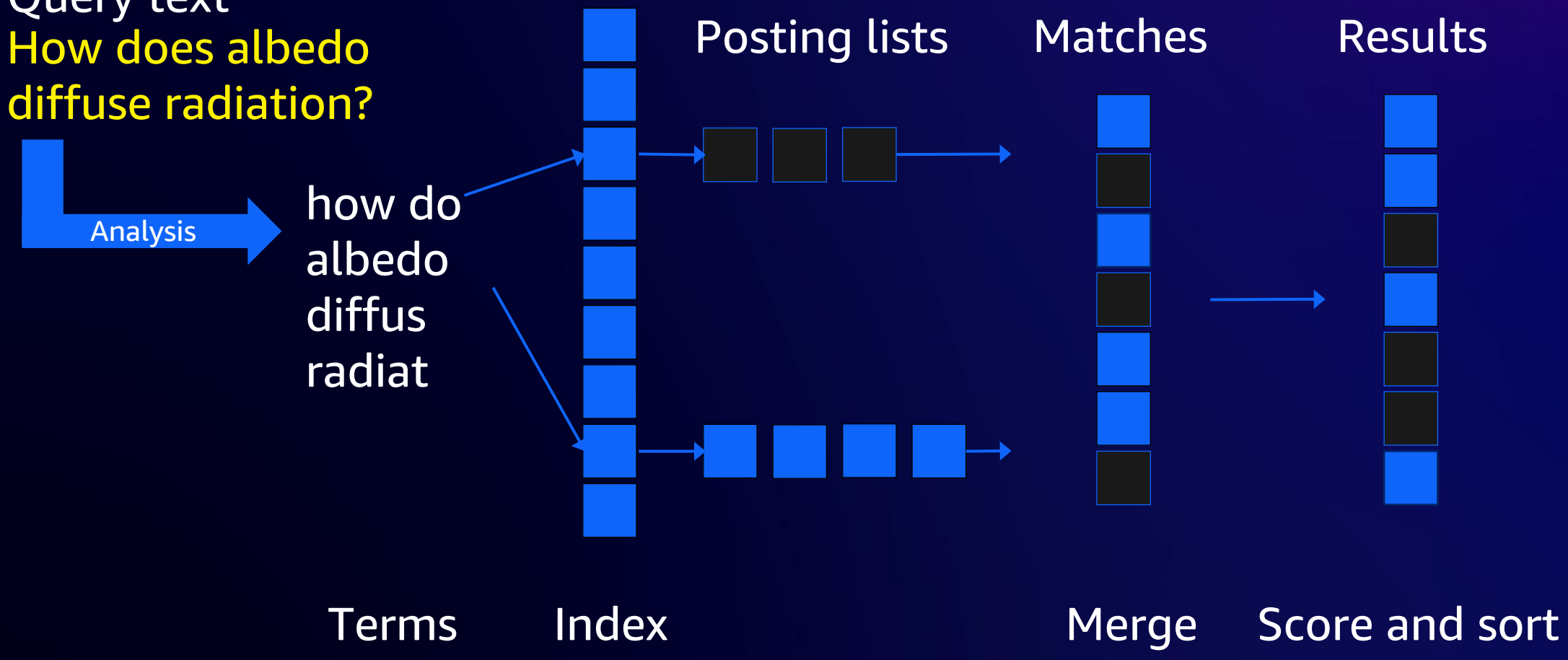
An index is composed of shards

Shards are distributed to data nodes

Lucene indices provide the core search functionality



Query execution



Text analysis (Wikipedia data)

Source text

The '''albedo''' of an object is the extent to which it diffusely reflects light, defined as

the ratio of [[diffusely reflected]] to incident [[electromagnetic radiation]]. It is a [[Dimensionless number|unitless]] measure indicative of a surface's or body's diffuse [[reflectivity]]. The word is derived from [[Latin]] ''albedo'' "whiteness", in turn from ''albus'' "white".

Standard analyzer

the albedo of an object is the extent to which it diffusely reflects light defined as the ratio of diffusely reflected to incident electromagnetic radiation it is a dimensionless number unitless measure indicative of a surface's or body's diffuse reflectivity the word is derived from latin albedo quot whiteness quot in turn from albus quot white quot

English analyzer

albedo object extent which diffus reflect light defin ratio diffus reflect incid electromagnet radiat dimensionless number unitless measur indic surfac bodi diffus reflect word deriv from latin albedo quot white quot turn from albu quot white quot

Text:text matching

Query text

How does albedo diffuse radiation?

Analyzed

how do albedo diffus radiat

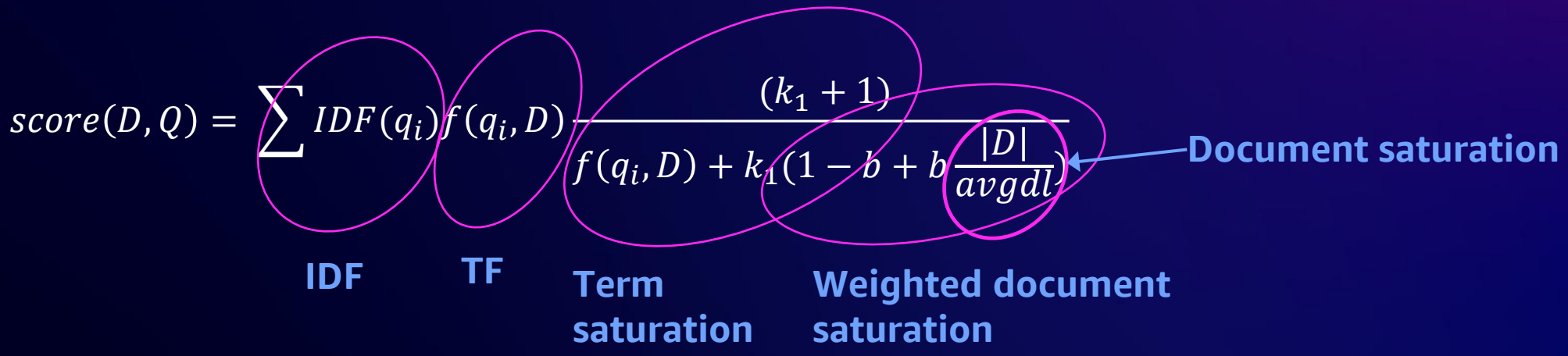
“Albedo”

albedo bject extent which
diffus reflect light defin ratio
diffus eflect incid
electromagne radiat
dimensionless number
unitless measur indic surfac
boc diffus reflect word deriv
from latir albedo quot white
quot turn from albu quot
white quot

“Albert Einstein”

1906 patent offic promot
einstein technic examin
second class he give up
academia 1908 he becam
privatdoz univers bern lt ref
gt citat last pai first abraham
author link abraham pai year
1982 titl subtl lord scienc life
albert einstein publish oxford
univers 7 lt ref gt 1910 he
wrote paper critic opalesc
describ cumul effect light
scatter individu molecul
atmospher diffus sky radiat
why sky blue lt ref name quot
levenson quot gt levenson
thoma quot http
www.pbs.org wgbh nova
einstein geniu einstein

Ranking with Okapi BM25



Based on probabilistic retrieval model

Considers term saturation & document length

Reward short documents, while penalizing matches in long documents

Pros and cons of text-based relevance

BM25 provides good accuracy

Exact match is appropriate in many cases

What if the query doesn’t have the terms?

The intent of the query is “hidden” behind the words

Vectors, generated by Large Language Models (LLMs) can capture more than just the words to improve search results

“Find me a cozy place to sit by the fire”

Vector engine key concepts

Vector: An array of values

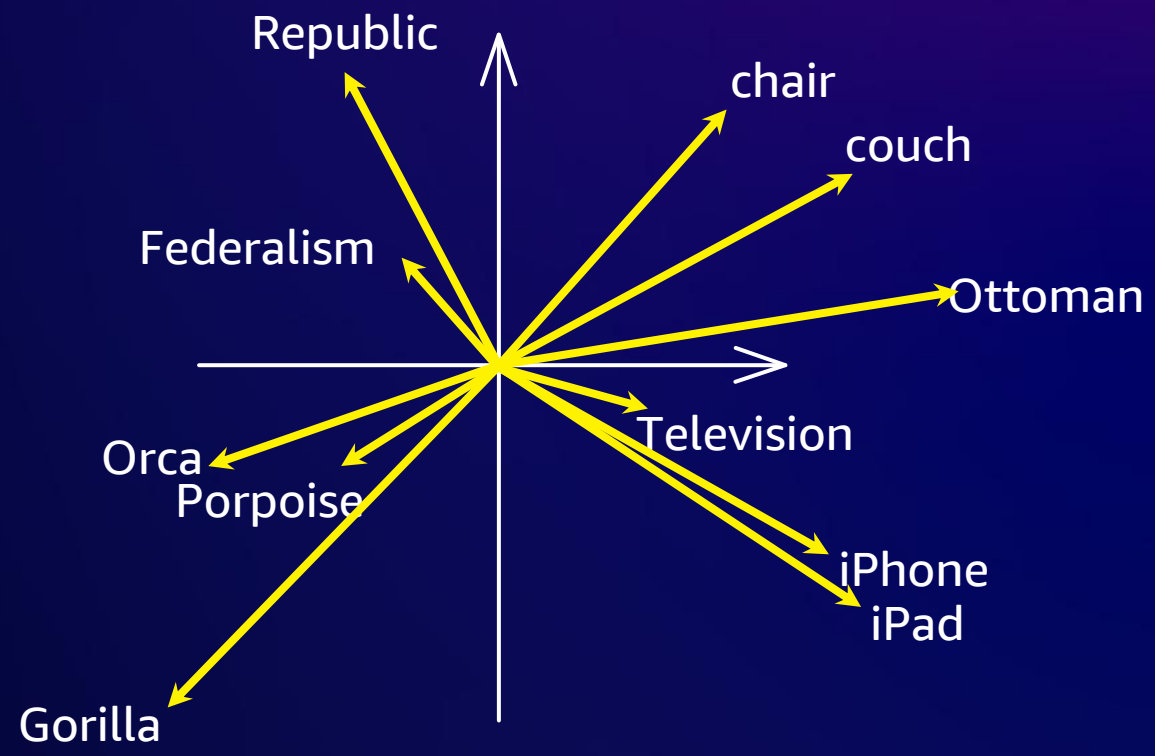
Model: A function that learns the relationship inputs -> outputs

Large language model (LLM): A model forward- and backward-codes text

Embedding: A vector added to a database entity that codes it

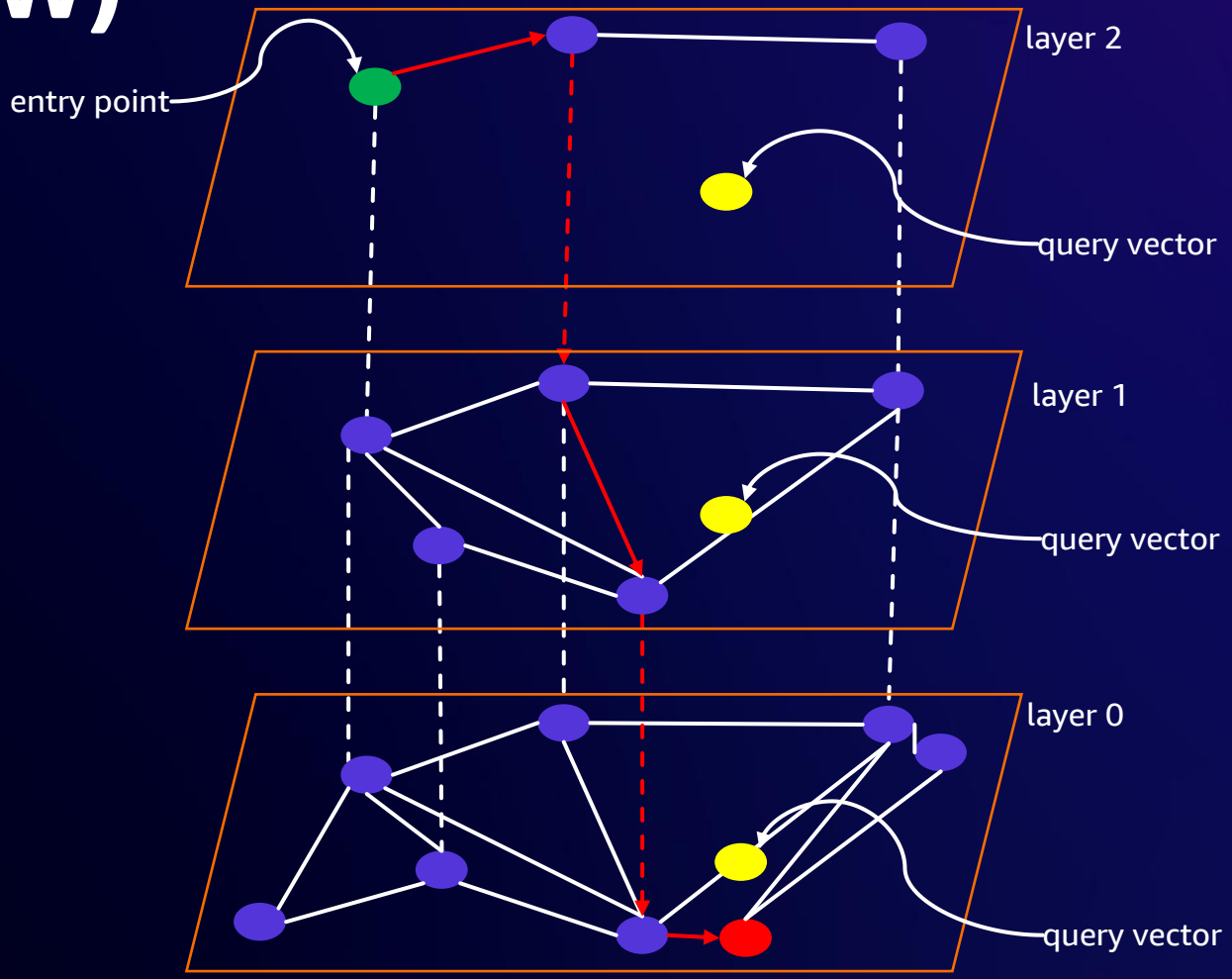
Engine/algorithm: The storage and similarity capability for vectors

Space type/similarity metric: The scoring function

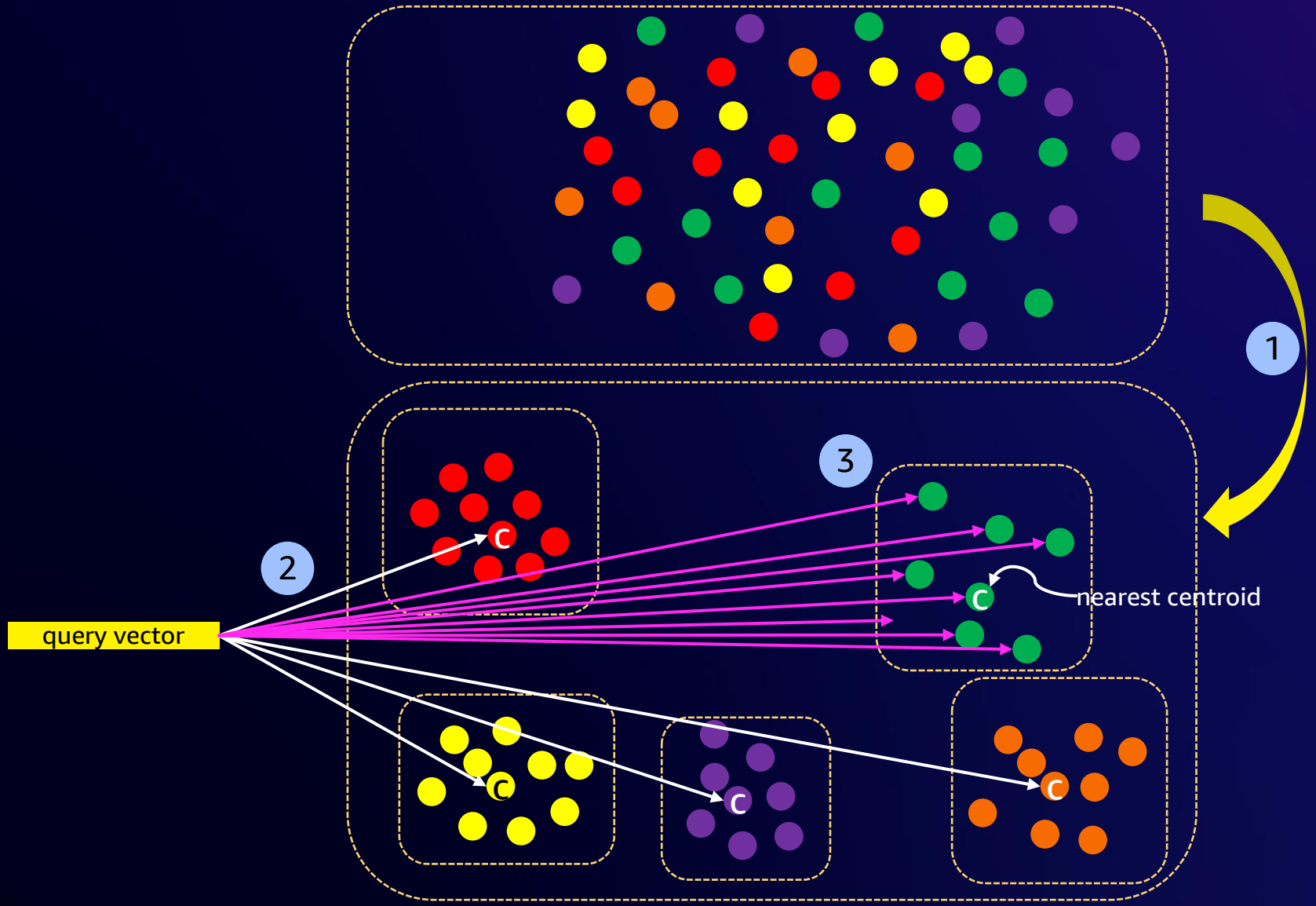


[0.46289062, -0.39257812, 0.25195312, 0.016845703, 0.20996094, 0.13867188, -0.20410156, 0.00041770935, 0.002029419, -0.15332031, 0.046142578, 0.4765625, 0.08642578, -0.07763672, -0.009399414, -0.123535156, 0.23730469, 0.15234375, 0.115234375, 0.079589844, 0.15527344, 0.078125, -0.24023438, 0.38867188, 0.078125, -0.3203125, -0.0058898926, 0.061035156, 0.33007812, -0.29101562, 0.46875, -0.078125, -0.11816406, -0.31640625, 0.20898438, -0.13183594, 0.07714844, 0.1640625, 0.34765625, -0.19824219, 0.35351562, -0.26171875, -0.12988281, -0.27148438, 0.0007209778, -0.16992188, -0.22460938, 0.14746094, 0.16894531, 0.28515625, -0.12109375, -0.07421875, 0.609375, -0.15039062, 0.55859375, 0.47265625, 0.20996094, 0.029541016, -0.15625, 0.064941406, 0.296875, 0.100097656, -0.31640625, 0.4375, 0.063964844, -0.011413574, 0.06201172, -0.14453125, 0.111328125, -0.038330078, -0.4296875, 0.09423828, -0.27734375, -0.5703125, 0.17285156, 0.0020141602, -0.048095703, 0.18066406, 0.30078125, -0.38085938, -0.107421875, 0.34765625, -0.045654297, 0.29101562, -0.0061035156, 0.31640625, -0.26953125, -0.2421875, 0.00013828278, -0.32226562, 0.19921875, 0.38476562, 0.62890625, -0.035888672, 0.025634766, 0.12792969, 0.33789062, -0.076660156, -0.91015625, -0.17382812, -0.15527344, -0.16503906, 0.06225586, 0.08300781, -0.010559082, 0.24121094, 0.46484375, 0.24121094, 0.16015625, 0.11621094, 0.328125, 0.026855469, -0.41210938, 0.24804688, -0.22753906, -0.34765625, -0.23730469, -0.20996094, -0.39453125, 0.36132812, 0.5625, 0.0065612793, 0.034179688, 0.018554688, -0.05834961, -0.24804688, -0.03173828, 0.19140625, 0.06640625, 0.18261719, 0.09667969, -0.265625, 0.006652832, 0.29101562, 0.203125, -0.19042969, -0.25, -0.09863281, 0.22363281, -0.0022125244, -0.40234375, -0.20996094, -0.1796875, -0.28320312, 0.011108398, -0.42773438, -0.10595703, 0.1015625, -0.24023438, -0.18359375, -0.087890625, -0.20898438, -0.3359375, 0.6875, 0.4453125, 0.025512695, -0.4921875, 0.50390625, -0.16210938, -0.3671875, 0.15234375, -0.16894531, 0.23339844, 0.03515625, -0.22070312, -0.15625, 0.19921875, 0.32421875, 0.46679688, 0.025268555, -0.032226562, 0.19726562, -0.16601562, -0.13085938, 0.08544922, 0.56640625, 0.40625, -0.13476562, -0.74609375, 0.33007812, 0.12988281, -0.5234375, 0.036132812, -0.049072266, 0.5390625, 0.16308594, 0.41601562, 0.009460449, -0.16503906, 0.14941406, -0.026733398, -0.118652344, 0.045410156, 0.036376953, 0.15625, -0.2578125, 0.28320312, -0.06738281, -0.24511719, -0.110839844, -0.037597656, -0.08300781, -0.39453125, -0.22753906, -0.30273438, -0.11376953, -0.21386719, -0.056396484, 0.067871094, 0.20117188, -0.32421875, -0.46289062, -0.14941406, -0.2578125, 0.26953125, 0.080566406, 0.1015625, 0.22851562, -0.33398438, 0.46484375, -0.06738281, -0.30078125, -0.06933594, -0.44921875, -0.24511719, -0.099121094, 0.28515625, -0.13671875, 0.23046875, -0.07910156, 0.52734375, -0.048583984, 0.19628906, -0.66015625, 0.08203125, -0.16699219, -0.27148438,...

ANN algorithms – Hierarchical navigable small world (HNSW)



ANN algorithms – Inverted file (IVF)

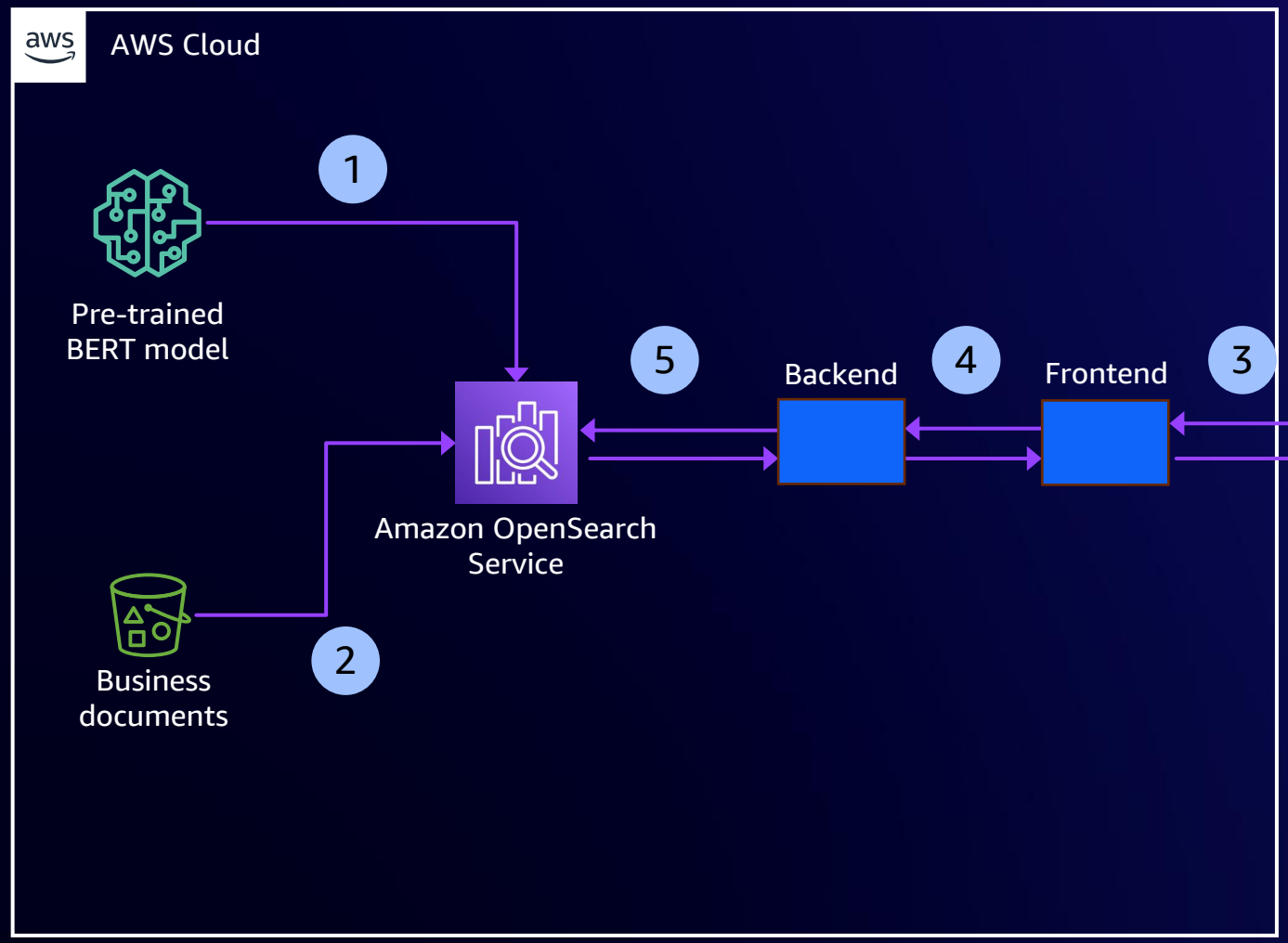


1. Cluster the vectors into buckets, each bucket has centroid
2. Find the nearest centroid by comparing distance of query vector and all centroid
3. Compare the distance of query vector and other vectors in the same bucket identified by step 2

OpenSearch ANN supported algorithms

	NMSLIB-HNSW	FAISS-HNSW	FAISS-IVF	Lucene-HNSW
Max dimension	16,000	16,000	16,000	1024
Filter	Post-filter	Post-filter	Post-filter	Filter while search
Training required	No	No	Yes	No
Distance formula	l2, innerproduct, cosinesimil, l1, linf	l2, innerproduct	l2, innerproduct	l2, cosinesimil
Vector volume	Tens of billions	Tens of billions	Tens of billions	< Ten million
Indexing latency	Low	Low	Lowest	Low
Query latency & quality	Low latency & high quality	Low latency & high quality	Low latency & low quality	High latency & high quality
Vector compression	Flat	Flat Product Quantization	Flat Product Quantization	Flat
Memory consumption	High	High Low with PQ	Medium Low with PQ	High

Semantic search (neural search plugin)



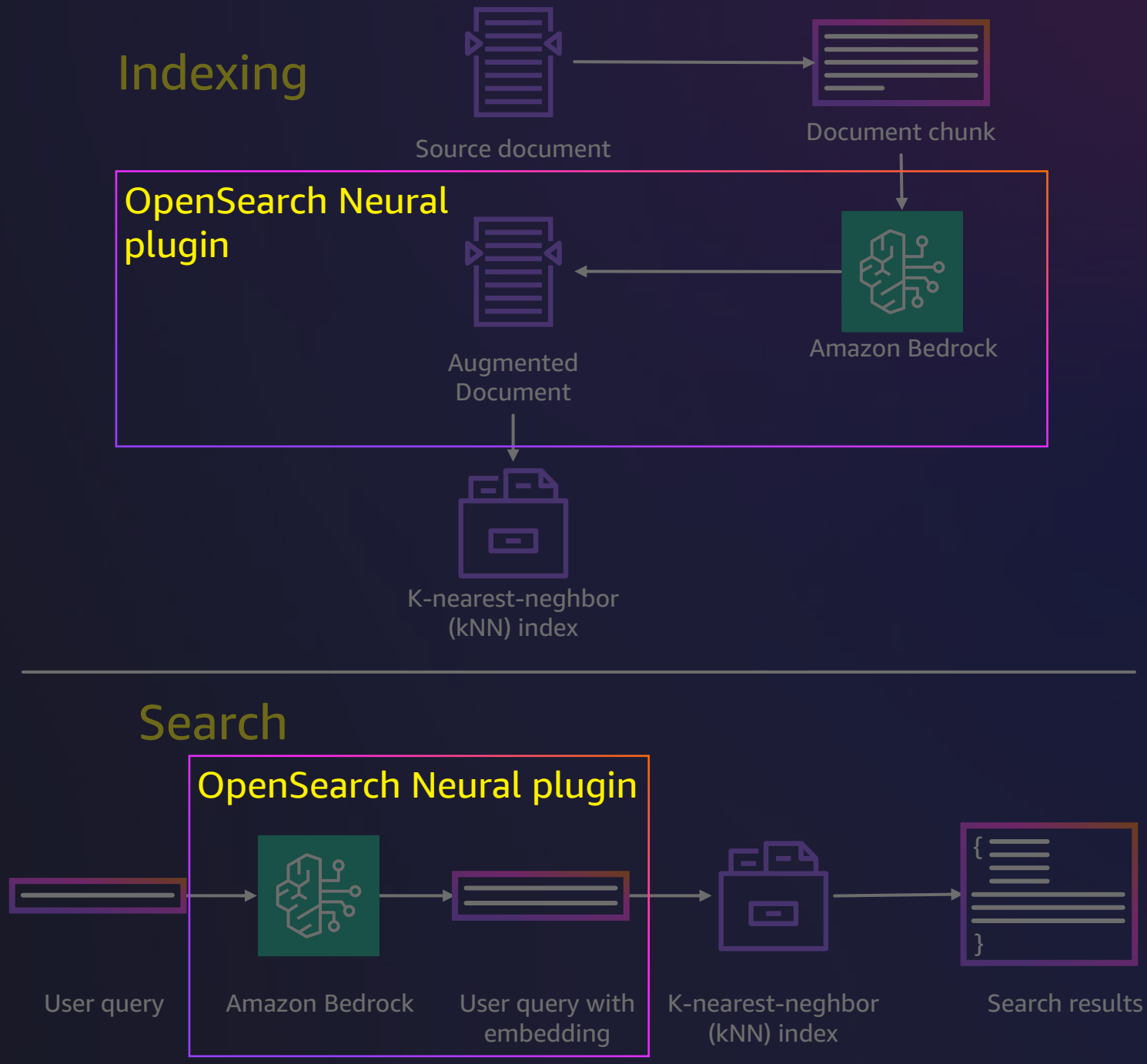
1. Create a connection to a 3P model hosting service
2. Run neural search pipeline to ingest documents into OpenSearch Service
3. Client submits a search request to API Gateway
4. Amazon API Gateway calls AWS Lambda backend service in Lambda
5. Backend service calls neural search API to get similar documents and return to client

Working with embeddings

OpenSearch Service provides connectors to 3P model-hosting systems – e.g., Amazon Bedrock

Indexing: Select chunks from the source document, send to the 3P system for embeddings

Search: Create embedding for the query then find nearest neighbors



Neural plugin

Load model, then

```
{
  'settings': { 'index.knn': True },
  'mappings': {
    'properties': {
      "plot_text": { "type": "text" },
      "plot_embedding": {
        "type": "knn_vector",
        "dimension": 384
      },
      "year": { "type": "integer" },
      "rank": { "type": "integer" },
      "rating": { "type": "float" },
      "running_time_secs":
        { "type": "integer" },…
```

Map

```
{
  "processors" :
  [
    {
      "text_embedding":
      {
        "model_id": "<model id>",
        "field_map": {
          "plot": "plot_embedding"
        }
      }
    }
  ]
}
```

Pipeline and ingest

```
{
  "query": {
    "bool": {
      "should": [{
        "neural": {
          "plot_embedding": {
            "query_text": "{{query}}",
            "model_id": "<<MODEL_ID>>",
            "k": 10
          }
        }
      }]
    }
  }
}
```

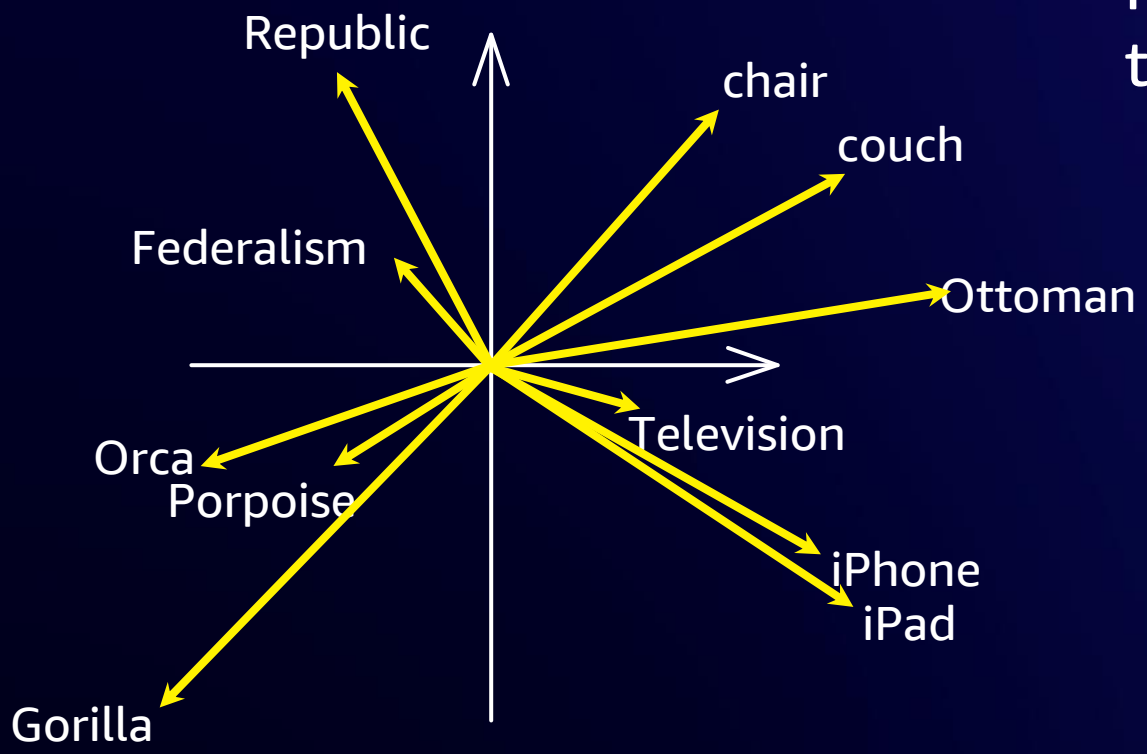
Query

Results: Transformer + BM25, NDCG@10

	BM25	Harmonic	Fine-tuned, arithmetic	Fine-tuned, geometric
NFCorpus	0.343	0.346	0.369	0.367
Trec-Covid	0.688	0.731	0.752	0.79
ArguAna	0.472	0.482	0.527	0.526
FiQA	0.254	0.281	0.364	0.350
Scifact	0.691	0.673	0.728	0.727
DBPedia	0.313	0.395	0.373	0.392
Quora	0.789	0.847	0.874	0.872
Scidocs	0.165	0.173	0.184	0.184
CQADupStack	0.325	0.333	0.3673	0.377
Amazon ESCI	0.081	0.088	0.091	0.091
	N/A	**6.42%**	**14.14%**	**14.93%**

Source: https://opensearch.org/blog/semantic-science-benchmarks/

Sparse vectors for the neural plugin



Find me a cozy place to sit by the fire

Chair	.901
Couch	.930
Ottoman	.802
Television	.311
iPhone	.013
iPad	.014
Gorilla	.021
Porpoise	.001
Orca	.002
Federalism	.010
Republic	.009

Ranking with sparse vectors

$$relevance = \sum_{t \in q \cap d} \omega_t^q \omega_t^d$$

Improves BM25 scoring, while preserving text relevance

Uses less RAM

Faster than dense vector ranking

10.2% to 17.4% improvement over BM25 on the BEIR test

14.7% improvement over dense vector ranking

Personalization – Amazon Music

100 Million

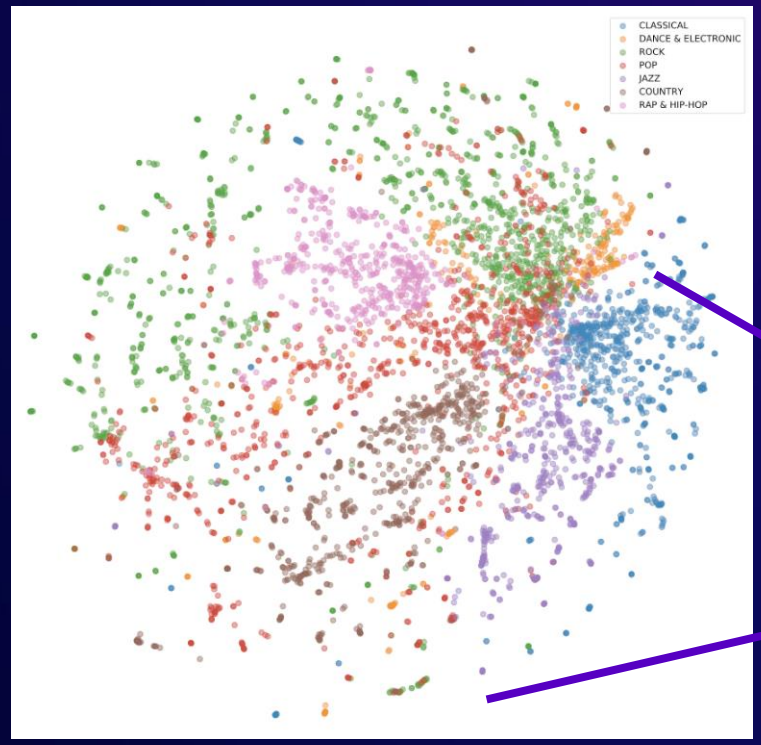
music tracks with recommendations based on user listening history

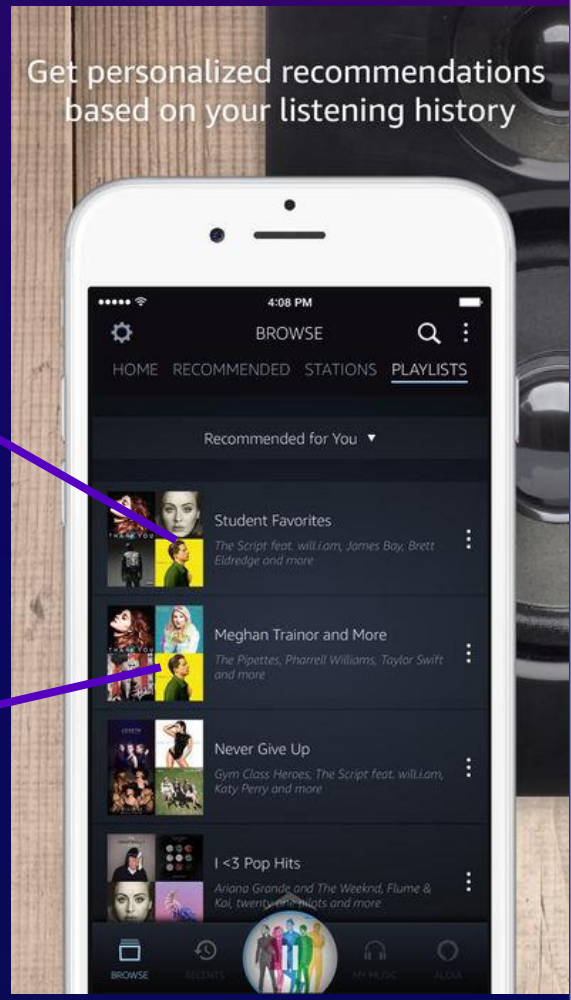
1.05 Billion

vectors indexed into OpenSearch Service to power item-item collaborative filtering

7.1K QPS

at peak to power recommendations.





Source: https://aws.amazon.com/blogs/big-data/amazon-opensearch-services-vector-database-capabilities-explained/

IP infringement detection – Amazon.com

8 Billion

items scanned daily

99%

of discovered infringements were automatically found or blocked through proactive controls

68 Billion

vectors encoded from product information indexed into OpenSearch Service to power vector search

Source: https://aws.amazon.com/blogs/big-data/amazon-opensearch-services-vector-database-capabilities-explained/

Vector search at Intuit


INTUIT
turbotax
creditkarma
quickbooks
mailchimp
Who we serve:
Consumers
Small businesses
Self-employed

INTUIT

Intuit is the **global financial technology platform** that **powers prosperity for the people and communities** we serve with Intuit TurboTax, Credit Karma, QuickBooks, and Mailchimp





Who we are

The data organization at Intuit offers fully managed paved roads for different database technologies such as NoSQL, search, relational & graph

Vector database use cases

Intuit aims to enhance the search experience by leveraging advanced techniques to identify user intent, rather than relying solely on traditional text-based matching. This approach will enable us to deliver more relevant and personalized search results, ultimately improving the overall user journey.

Generative QA

Build retrieval enhanced generative QA systems

Entity resolution

Product matching, entity matching, identity resolution

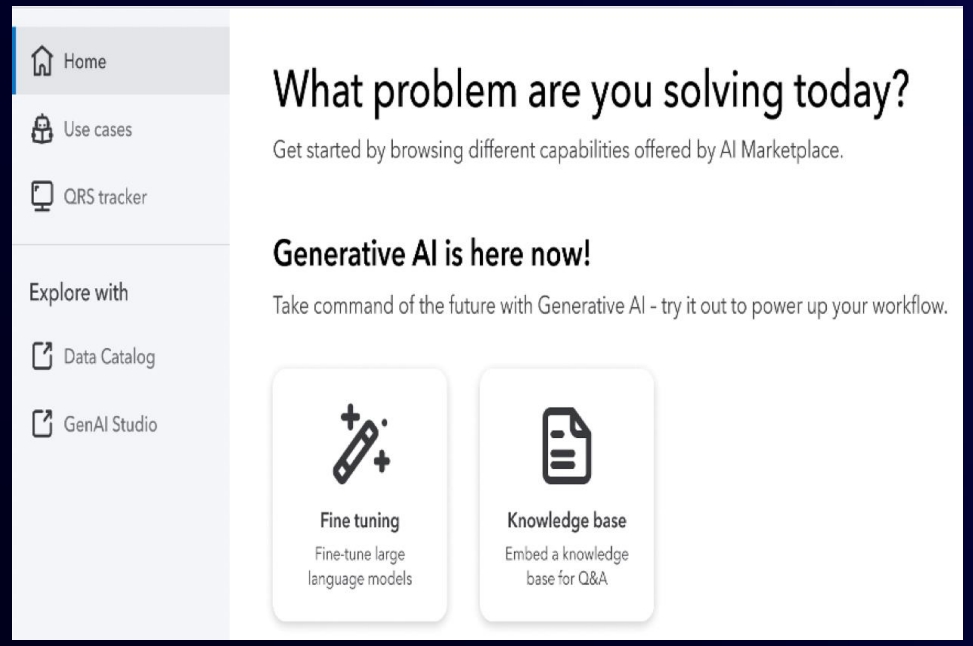
Document discovery

Finding similar documents and later processing to retrieve embedded information

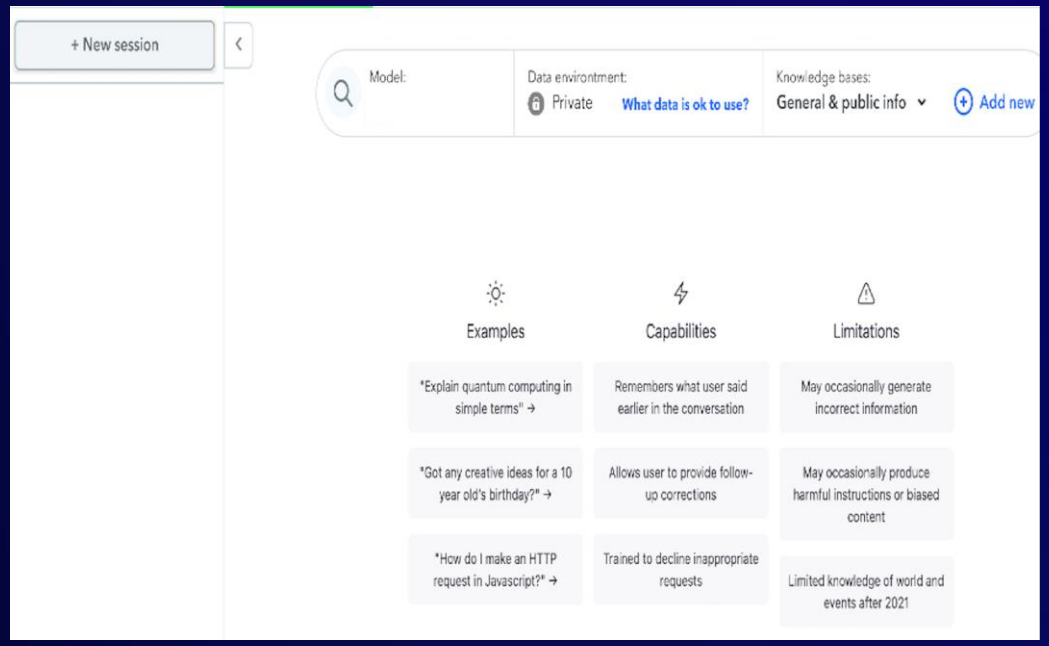
Generative QA

Paved path for building a knowledge base using a RAG model architecture

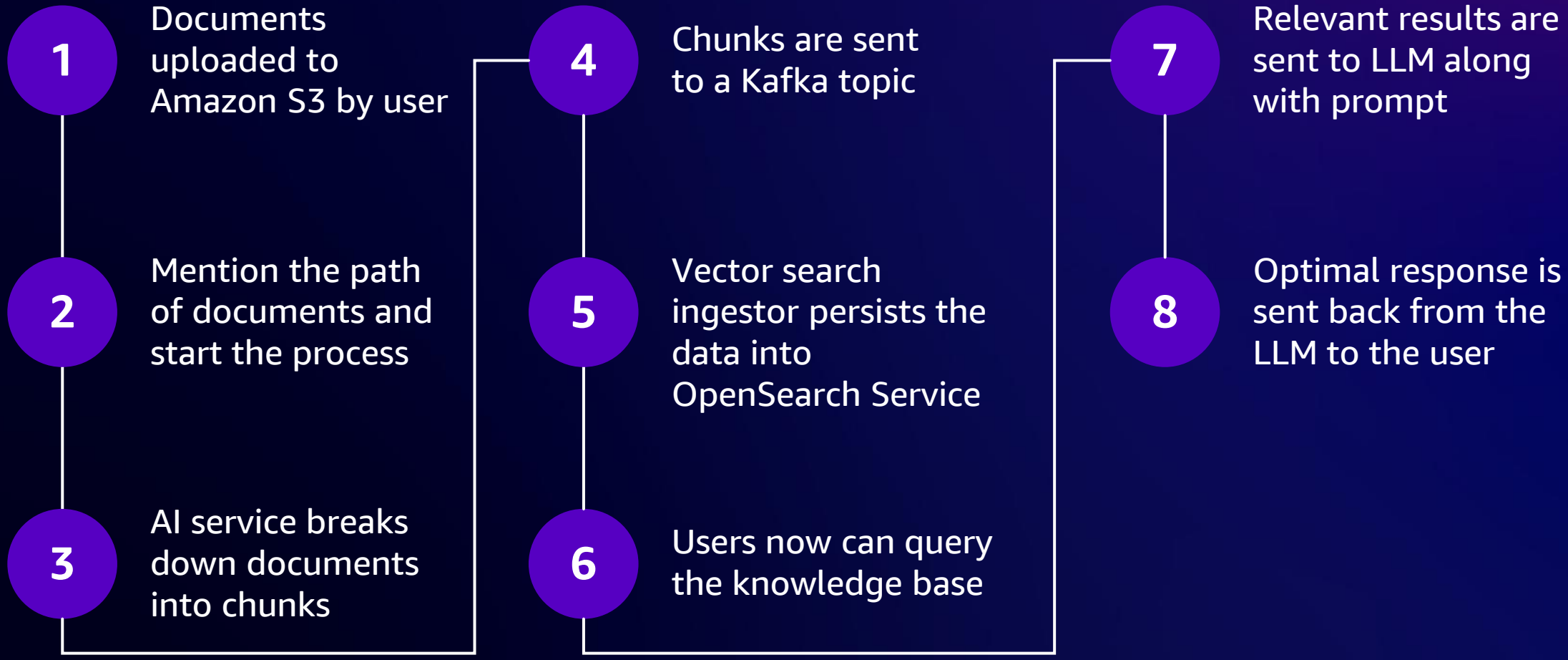
Create Knowledge Base



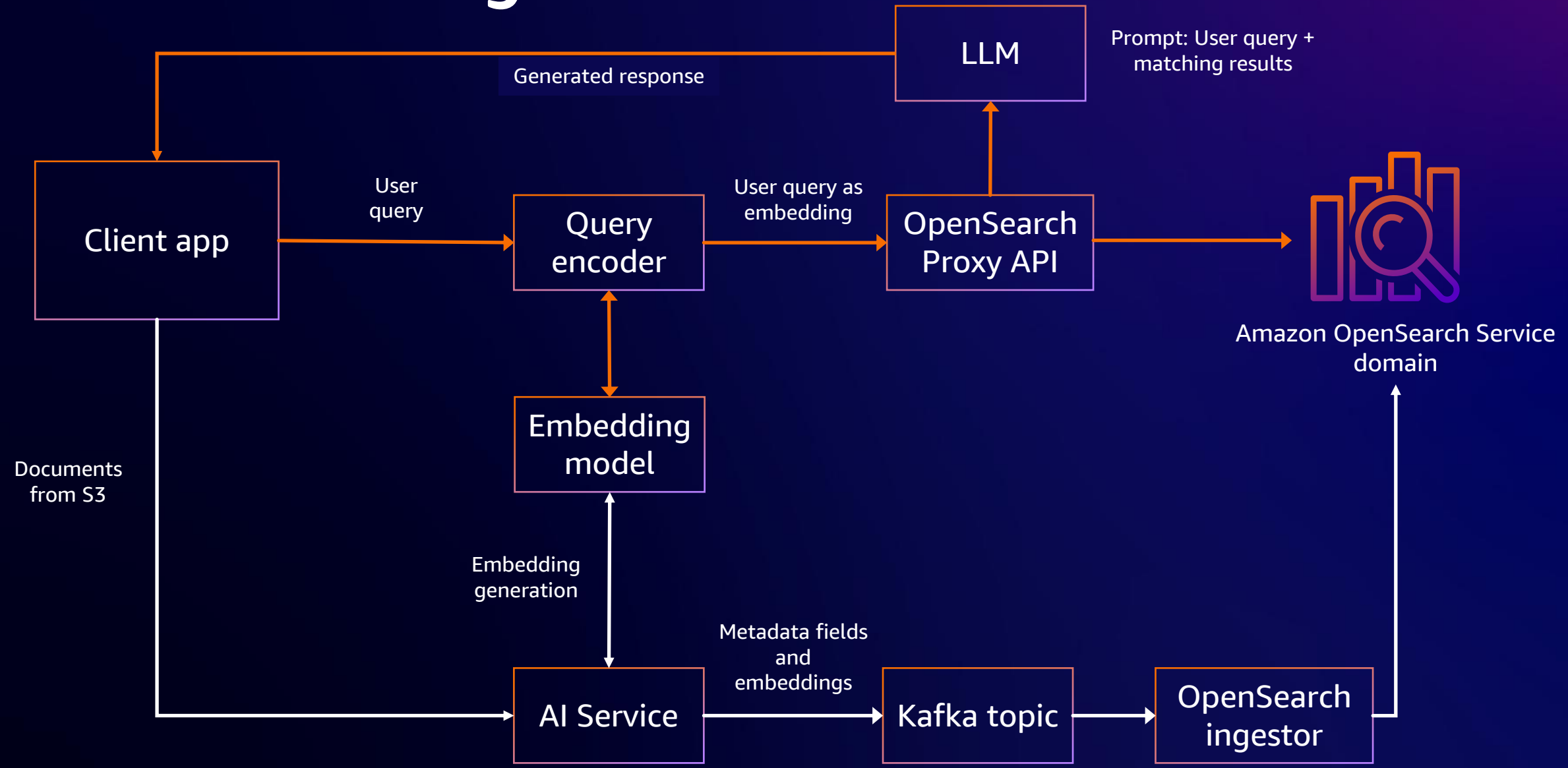
Query Knowledge Base



Use case and data flow



Architecture diagram



Metrics

- 7 use cases onboarded to production
- Exact k-NN, approximate k-NN, approximate k-NN with pre-filtering queries are in use
- No P1, P2 incidents after launch
- 46 use cases in pipeline
- 2 customers are experimenting with product quantization compression

Learnings from adopting Amazon OpenSearch Service as a vector store

- Leverage monitoring capabilities of OpenSearch Service
- Leverage data operations such as reindexing, backup, index management

- Scoring disparity during hybrid search
- Support higher dimensions on Lucene engine
- Higher latency when documents scale to billions

Desired features from OpenSearch Service

- Easier configuration to enable different compression techniques
- Reduce memory footprint by storing subset of the vectors to the disk
- Support model-serving framework from OpenSearch open source project

Demo: OpenSearch Service with ML Connectors

Simplify & operationalize vector hydration

Demo scope

Create a connector to your external model

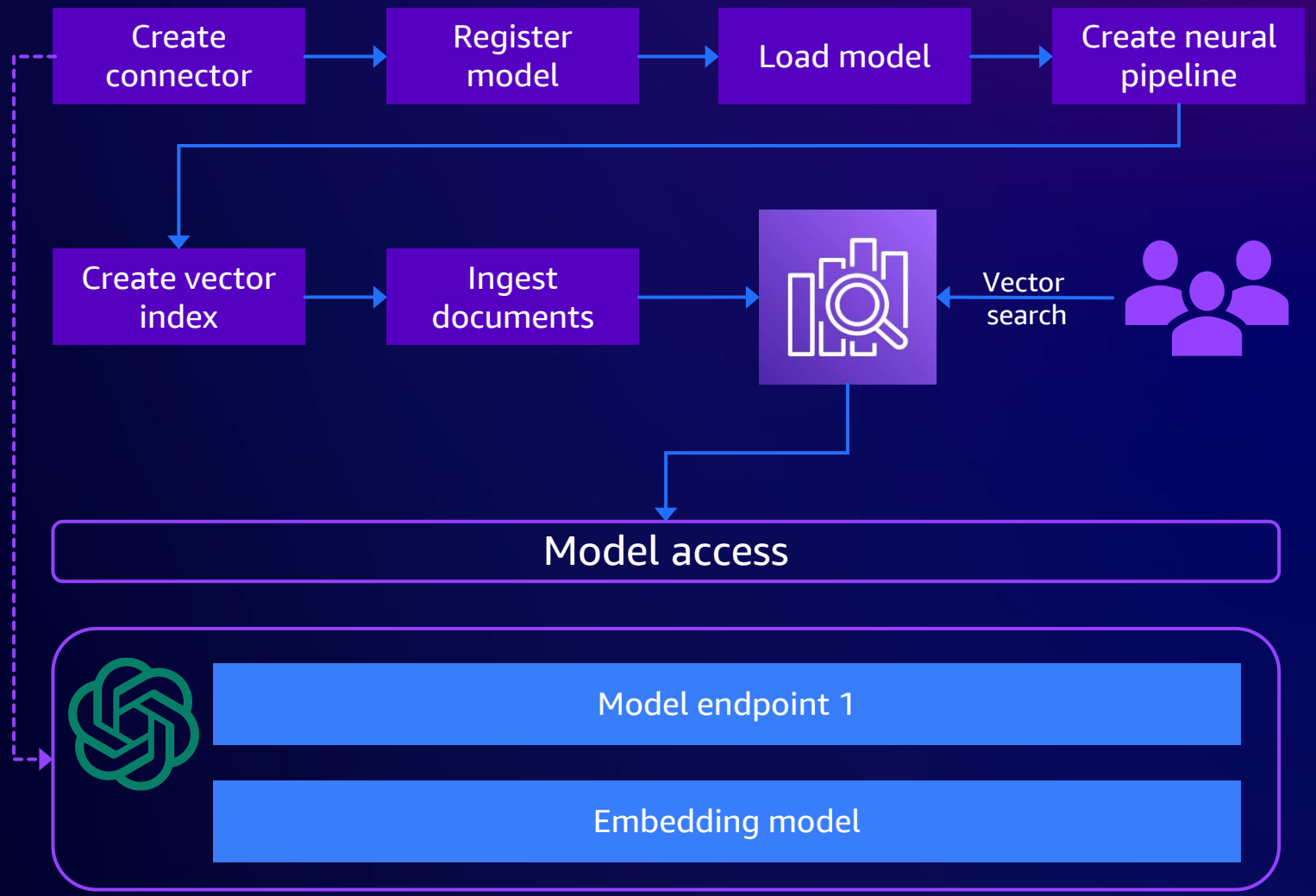
Securely connect to it using AWS Secrets Manager and AWS Identity and Access Management (IAM)

Register the model in OpenSearch using the connector

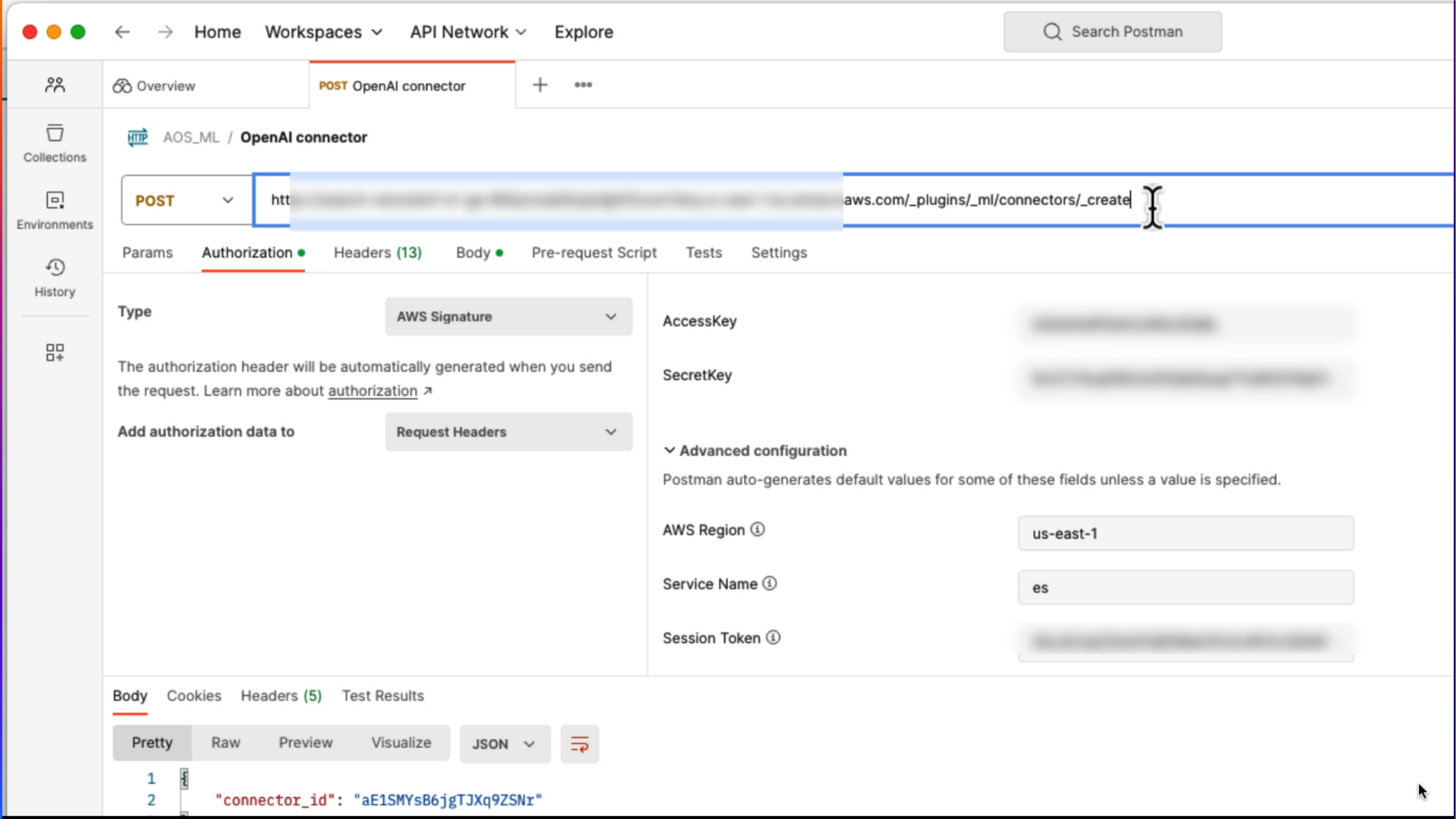
Load the model to memory

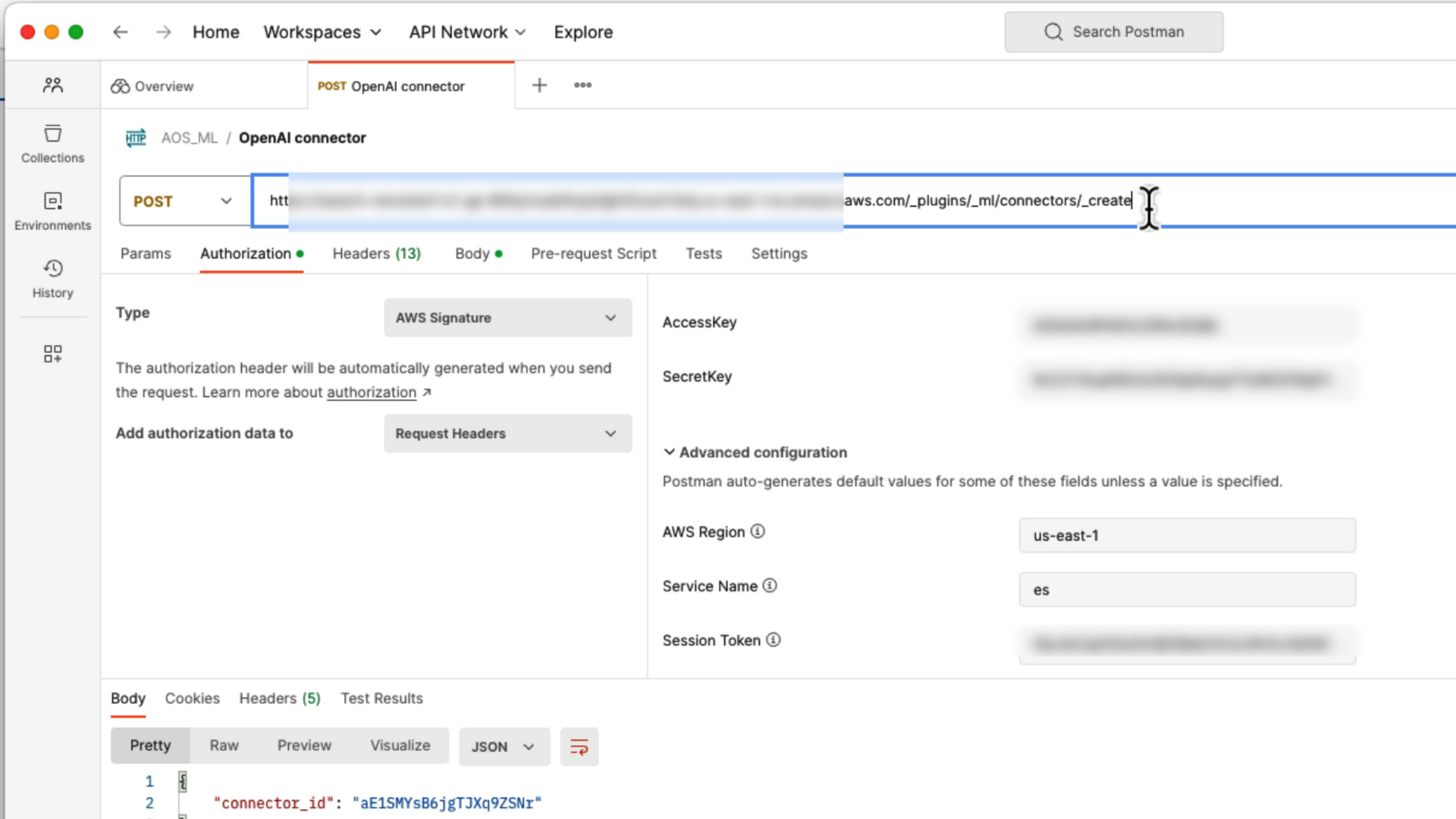
Create a **neural ingestion pipeline**

Ingest & search



Demo!


Home
Workspaces
API Network
Explore
Search Postman
Overview
POST OpenAI connector
Collections
Environments
History
AOS_ML / OpenAI connector
POST
aws.com/_plugins/_ml/connectors/_create
Params
Authorization
Headers (13)
Body
Pre-request Script
Tests
Settings
Type
AWS Signature
The authorization header will be automatically generated when you send the request. Learn more about authorization
Add authorization data to
Request Headers
AccessKey
SecretKey
Advanced configuration
Postman auto-generates default values for some of these fields unless a value is specified.
AWS Region
us-east-1
Service Name
es
Session Token
Body
Cookies
Headers (5)
Test Results
Pretty
Raw
Preview
Visualize
JSON
1
2
"connector_id": "aE1SMYsB6jgTJXq9ZSNr"


Home
Workspaces
API Network
Explore
Search Postman
Overview
POST OpenAI connector
Collections
Environments
History
AOS_ML / OpenAI connector
POST
aws.com/_plugins/_ml/connectors/_create
Params
Authorization
Headers (13)
Body
Pre-request Script
Tests
Settings
Type
AWS Signature
The authorization header will be automatically generated when you send the request. Learn more about authorization
Add authorization data to
Request Headers
AccessKey
SecretKey
Advanced configuration
Postman auto-generates default values for some of these fields unless a value is specified.
AWS Region
us-east-1
Service Name
es
Session Token
Body
Cookies
Headers (5)
Test Results
Pretty
Raw
Preview
Visualize
JSON
1
2
"connector_id": "aE1SMYsB6jgTJXq9ZSNr"

OpenSearch ML Connector

STREAMLINING ML INTEGRATIONS

Simplify & operationalize vector hydration

Build native integrations with embedding services

Leverage vector search to power your generative AI applications

Secure and manage access to the ML models

Leverage the distributed design of OpenSearch to build a **stable, scalable** vector database

OpenSearch Service vector search public content

OpenSearch Service's vector DB capabilities

OpenSearch Serverless vector engine

Semantic workshop

Vector search video

Multi-modal

Semantic benchmarks

Scaling for vectors

Building chatbots

aws

Wrap up

Use Amazon OpenSearch Service's core capabilities for text-text matching with good relevance to provide low-latency, high-throughput query processing

Add vector embeddings, from LLMs for improved relevance

OpenSearch Service brings multiple vector algorithms and engines with connectors to third-party model-hosting services to make it simple to work with embeddings

You've just completed an AWS Analytics Superheroes session

Scan QR code to learn more
about the AWS Analytics Superheroes

Thank you!

Please complete the session survey in the mobile app

Jon Handler
@_searchgeek
handler@amazon.com

Achal Kumar

Aruna Govindaraju
arunagov@amazon.com