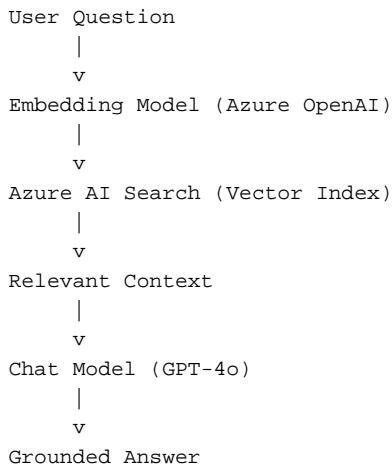


Azure RAG Chatbot – Manual Text Ingestion

A clean, interview-ready, step-by-step reference for building a Retrieval-Augmented Generation (RAG) chatbot using Azure OpenAI and Azure AI Search with manual text ingestion. This document reflects the latest working Azure SDKs and UI (Foundry-based deployments).

1. High-Level Architecture



This flow ensures the LLM answers strictly from indexed documents, preventing hallucination.

2. Azure Resource Setup

Resources created in Azure Portal:

- Resource Group: rg-ai102-rag
- Azure OpenAI (Standard S0)
- Azure AI Search (Basic tier – vector search enabled)

Interview note: Azure OpenAI handles access, billing, and endpoints. Azure AI Search acts as the vector database for retrieval.

3. Model Deployment (Azure AI Foundry)

- Deployments are created in Azure AI Foundry, not directly in the Azure OpenAI blade.
- Chat model: gpt-4o → deployment name: chat-model
- Embedding model: text-embedding-3-small → deployment name: embed-model

Interview note: Deployment names, not model names, are used in API calls.

4. Local Project Structure

```
rag-chatbot/
  create_index.py
  ingest_docs.py
  chat.py
  .env
```

5. Environment Configuration

```
AZURE_OPENAI_ENDPOINT=https://<openai>.openai.azure.com/
AZURE_OPENAI_KEY=<key>
AZURE_OPENAI_CHAT_DEPLOYMENT=chat-model
AZURE_OPENAI_EMBED_DEPLOYMENT=embed-model
```

```
AZURE_SEARCH_ENDPOINT=https://<search>.search.windows.net
AZURE_SEARCH_KEY=<admin-key>
AZURE_SEARCH_INDEX=rag-index
```

6. Vector Index Creation (Latest API)

A vector-enabled index is created using a vectorSearchProfile with HNSW algorithm.

```
SearchField(
  name="embedding",
  type=Collection(Single),
  dimensions=1536,
  vector_search_profile_name="vector-profile"
)
```

Interview note: Azure AI Search now requires vectorSearchProfile. Older vector_search_configuration is deprecated.

7. Manual Text Ingestion

```
embedding = embeddings.create(input=text)
search_client.upload_documents([
    "id": uuid,
    "content": text,
    "embedding": embedding
]))
```

8. Retrieval Using VectorizedQuery

```
vector_query = VectorizedQuery(
    vector=q_embedding,
    k_nearest_neighbors=3,
    fields="embedding"
)
```

Interview note: Passing vector= directly to search() is deprecated. VectorizedQuery is mandatory.

9. RAG Chat Completion

```
response = chat.completions.create(
    model="chat-model",
    messages=[system, user],
    temperature=0
)
```

10. One-Page Interview Summary

| Topic | Key Point |
|-----------------------|--|
| RAG | Retrieval + Generation using external data |
| Embeddings | text-embedding-3-small for vectorization |
| Search | Azure AI Search with HNSW vector index |
| Chat Model | GPT-4o for grounded responses |
| Hallucination Control | Context-only answering + temperature=0 |
| Cost Control | Standard S0 + budgets |