Agentic retrieval in Azure AI Search

> **Note:** This feature is currently in public preview. This preview is provided without a service-level agreement and isn't recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see Supplemental Terms of Use for Microsoft Azure Previews.

What is agentic retrieval? In Azure AI Search, *agentic retrieval* is a new multi-query pipeline designed for complex questions posed by users or agents in chat and copilot apps. It's intended for Retrieval Augmented Generation (RAG) patterns and agent-to-agent workflows.

Here's what it does:

- Uses a large language model (LLM) to break down a complex query into smaller, focused subqueries for better coverage over your indexed content. Subqueries can include chat history for extra context.
- Runs subqueries in parallel. Each subquery is semantically reranked to promote the most relevant matches.
- Combines the best results into a unified response that an LLM can use to generate answers with your proprietary content.
- The response is modular yet comprehensive in how it also includes a query plan and source documents. You can choose to use just the search results as grounding data, or invoke the LLM to formulate an answer.

This high-performance pipeline helps you generate high quality grounding data (or an answer) for your chat application, with the ability to answer complex questions quickly.

Programmatically, agentic retrieval is supported through a new Knowledge Base object in the 2025-11-01-preview and in Azure SDK preview packages that provide the feature. A knowledge base's retrieval response is designed for downstream consumption by other agents and chat apps.

Why use agentic retrieval

There are two use cases for agentic retrieval. First, it's the basis of the Foundry IQ experience in the Microsoft Foundry (new) portal. It provides the knowledge layer for agent solutions in Microsoft Foundry. Second, it's the basis for custom agentic solutions that you create using the Azure AI Search APIs.

You should use agentic retrieval when you want to provide agents and apps with the most relevant content for answering harder questions, leveraging chat context and your proprietary content.

The *agentic* aspect is a reasoning step in query planning processing that's performed by a supported large language model (LLM) that you provide. The LLM analyzes the entire chat thread to identify the underlying information need. Instead of a single, catch-all query, the LLM breaks down compound questions into focused subqueries based on: user questions, chat history, and parameters on the request. The subqueries target your indexed documents (plain text and vectors) in Azure AI Search. This hybrid approach ensures you surface both keyword matches and semantic similarities at once, dramatically improving recall.

The *retrieval* component is the ability to run subqueries simultaneously, merge results, semantically rank results, and return a three-part response that includes grounding data for the next conversation turn, reference data so that you can inspect the source content, and an activity plan that shows query execution steps.

Query expansion and parallel execution, plus the retrieval response, are the key capabilities of agentic retrieval that make it the best choice for generative AI (RAG) applications.

[[IMG:agentric-retrieval-example.png]]

Agentic retrieval adds latency to query processing, but it makes up for it by adding these capabilities:

- Reads in chat history as an input to the retrieval pipeline.
- Deconstructs a complex query that contains multiple "asks" into component parts. For example: "find me a hotel near the beach, with airport transportation, and that's within walking distance of vegetarian restaurants."
- Rewrites an original query into multiple subqueries using synonym maps (optional) and LLM-generated paraphrasing.
- Corrects spelling mistakes.
- Executes all subqueries simultaneously.
- Outputs a unified result as a single string. Alternatively, you can extract parts of the response for your solution. Metadata about query execution and reference data is included in the response.

Agentic retrieval invokes the entire query processing pipeline multiple times for each subquery, but it does so in parallel, preserving the efficiency and performance necessary for a reasonable user experience.

> **Note:** Including an LLM in query planning adds latency to a query pipeline. You can mitigate the effects by using faster models, such as gpt-4o-mini, and summarizing the message threads. You can minimize latency and costs by setting properties that limit LLM processing. You can also exclude LLM processing altogether for just text and hybrid search and your own query planning logic.

Architecture and workflow

Agentic retrieval is designed for conversational search experiences that use an LLM to intelligently break down complex queries. The system coordinates multiple Azure services to deliver comprehensive search results.

[[IMG:agentic-retrieval-architecture.png]]

How it works

The agentic retrieval process works as follows:

1. **Workflow initiation:** Your application calls a knowledge base with retrieve action that provides a query and conversation history.
2. **Query planning:** A knowledge base sends your query and conversation history to an LLM, which analyzes the context and breaks down complex questions into focused subqueries. This step is automated and not customizable.
3. **Query execution:** The knowledge base sends the subqueries to your knowledge sources. All subqueries run simultaneously and can be keyword, vector, and hybrid search. Each subquery undergoes semantic reranking to find the most relevant matches. References are extracted and retained for citation purposes.
4. **Result synthesis:** The system combines all results into a unified response with three parts: merged content, source references, and execution details.

Your search index determines query execution and any optimizations that occur during query execution. Specifically, if your index includes searchable text and vector fields, a hybrid query executes. If the only searchable field is a vector field, then only pure vector search is used. The index semantic configuration, plus optional scoring profiles, synonym maps, analyzers, and normalizers (if you add filters) are all used during query execution. You must have named defaults for a semantic configuration and a scoring profile.

Required components

Component	Service	Role
LLM	Azure OpenAI	Creates subqueries from conversation context and later uses grounding data for answer generation
Knowledge base	Azure AI Search	Orchestrates the pipeline, connecting to your LLM and managing query parameters
Knowledge source	Azure AI Search	Wraps the search index with properties pertaining to knowledge base usage
Search index	Azure AI Search	Stores your searchable content (text and vectors) with semantic configuration
Semantic ranker	Azure AI Search	Required component that reranks results for relevance (L2 reranking)

Integration requirements

Your application drives the pipeline by calling the knowledge base and handling the response. The pipeline returns grounding data that you pass to an LLM for answer generation in your conversation interface. For implementation details, see Tutorial: Build an end-to-end agentic retrieval solution.

> **Note:** Only gpt-4o, gpt-4.1, and gpt-5 series models are supported for query planning. You can use any model for final answer generation.

How to get started

To create an agentic retrieval solution, you can use the Azure portal, the latest preview REST APIs, or a preview Azure SDK package that provides the functionality.

Quickstarts

- Quickstart: Agentic retrieval in the Azure portal
- Quickstart: Agentic retrieval (C#, Java, JavaScript, Python, TypeScript, REST)

How-to guides

- Create a knowledge source:
  - Blob
  - OneLake
  - Remote SharePoint
  - Indexed SharePoint
  - Search index
  - Web
- Create a knowledge base
- Use answer synthesis for citation-backed responses
- Use a knowledge base to retrieve data

Tutorials

- Tutorial: Build an end-to-end agentic retrieval solution

Code samples

- Quickstart-Agentic-Retrieval: Python
- Quickstart-Agentic-Retrieval: .NET
- Quickstart-Agentic-Retrieval: REST

- End-to-end with Azure AI Search and Foundry Agent Service

REST API references

- Knowledge Sources
- Knowledge Bases
- Knowledge Retrieval

Demos

- Azure OpenAI Demo has been updated to use agentic retrieval.

Availability and pricing

Agentic retrieval is available in selected regions. Knowledge sources and knowledge bases also have maximum limits that vary by pricing tier and retrieval reasoning effort.

It has a dependency on premium features. If you disable semantic ranker for your search service, you effectively disable agentic retrieval.

Plan	Description
Free	A free tier search service provides 50 million free agentic reasoning tokens per month. On higher tiers, you can choose between the free plan (default) and the standard plan.
Standard	The standard plan is pay-as-you-go pricing once the monthly free quota is consumed. After the free quota is used up, you are charged an additional fee for each additional one million agentic reasoning tokens. You aren't notified when the transition occurs. For more information about charges by currency, see the Azure AI Search pricing page.

Token-based billing for LLM-based query planning and answer synthesis (optional) is pay-as-you-go in Azure OpenAI. It's token based for both input and output tokens. The model you assign to the knowledge base is the one charged for token usage. For example, if you use gpt-4o, the token charge appears in the bill for gpt-4o.

Token-based billing for agentic retrieval is the number of tokens returned by each subquery.

Aspect	Classic single-query pipeline	Agentic retrieval multi-query pipeline
Unit	Query based (1,000 queries) per unit of currency	Token based (1 million tokens per unit of currency)
Cost per unit	Uniform cost per query	Uniform cost per token
Cost estimation	Estimate query count	Estimate token usage
Free tier	1,000 free queries	50 million free tokens

Example: Estimate costs

Agentic retrieval has two billing models: billing from Azure OpenAI (query planning and, if enabled, answer synthesis) and billing from Azure AI Search for agentic retrieval.

This pricing example omits answer synthesis, but helps illustrate the estimation process. Your costs could be lower. For the actual price of transactions, see Azure OpenAI pricing.

Estimated billing costs for query planning

To estimate the query plan costs as pay-as-you-go in Azure OpenAI, let's assume gpt-4o-mini:

- 15 cents for 1 million input tokens.
- 60 cents for 1 million output tokens.
- 2,000 input tokens for average chat conversation size.
- 350 tokens for average output plan size.

Estimated billing costs for query execution

To estimate agentic retrieval token counts, start with an idea of what an average document in your index looks like. For example, you might approximate:

- 10,000 chunks, where each chunk is one to two paragraphs of a PDF.
- 500 tokens per chunk.
- Each subquery reranks up to 50 chunks.
- On average, there are three subqueries per query plan.

Calculating price of execution

1. Assume we make 2,000 agentic retrievals with three subqueries per plan. This gives us about 6,000 total queries.
2. Rerank 50 chunks per subquery, which is 300,000 total chunks.
3. Average chunk is 500 tokens, so the total tokens for reranking is 150 million.
4. Given a hypothetical price of 0.022 per token, $3.30 is the total cost for reranking in US dollars.
5. Moving on to query plan costs: 2,000 input tokens multiplied by 2,000 agentic retrievals equal 4 million input tokens for a total of 60 cents.
6. Estimate the output costs based on an average of 350 tokens. If we multiply 350 by 2,000 agentic retrievals, we get 700,000 output tokens total for a total of 42 cents.

Putting it all together, you'd pay about $3.30 for agentic retrieval in Azure AI Search, 60 cents for input tokens in Azure OpenAI, and 42 cents for output tokens in Azure OpenAI, for $1.02 for query planning total. The combined cost for the full execution is $4.32.

Tips for controlling costs

- Review the activity log in the response to find out what queries were issued to which sources and the parameters used. You can reissue those queries against your indexes and use a public tokenizer to estimate tokens and compare to API-reported usage. Precise reconstruction of a query or response isn't guaranteed however. Factors include the type of knowledge source, such as public web data or a remote SharePoint knowledge source that's predicated on a user identity, which can affect query reproduction.
- Reduce the number of knowledge sources (indexes); consolidating content can lower fan-out and token volume.
- Lower the reasoning effort to reduce LLM usage during query planning and query expansion (iterative search).
- Organize content so the most relevant information can be found with fewer sources and documents (For example, curated summaries or tables).