Intel® AI for Enterprise RAG 2.0.0 User Guide

Version 1.0 | November 20, 2025

Table of Contents

1. Welcome & Purpose

Welcome to the Intel® AI for Enterprise RAG User Guide. This guide is designed to help you understand and use Intel's open-source Enterprise Retrieval-Augmented Generation (RAG) application effectively.

The application supports two user roles:

- End Users - who query the questions and consume results.
- Administrators (Admins) - who configure, manage, and maintain the system, ensuring smooth operation and accurate results.

This guide provides:

- Step-by-step instructions for both end users and admins.
- Explanations of key features and workflows.
- Troubleshooting advice and resources for further support.

1.1 Who Should Use This Guide?

- **End Users**: Business professionals, analysts, or employees who need to search for and retrieve information from enterprise knowledge sources.

- **Admins**: Technical staff responsible for setting up, configuring, and maintaining the RAG system.

1.2 What This Guide Does Not Cover

This guide does not include installation or deployment instructions. For details on how to set up Intel® AI for Enterprise RAG, please refer to the GitHub repository: https://github.com/opea-project/Enterprise-RAG

2. About Intel® AI for Enterprise RAG

Intel® AI for Enterprise RAG is an open-source application that combines large language models (LLMs) with your organization's knowledge sources to deliver accurate, context-aware answers. This approach is known as Retrieval-Augmented Generation (RAG). Instead of relying only on a model's built-in knowledge, RAG retrieves relevant information from your enterprise data and uses it to generate grounded, verifiable responses.

2.1 Key Benefits

- Enterprise-Ready: Scales to handle diverse and complex data sources.
- Accurate Responses: Answers are supported with citations from your own content.
- Flexible & Modular: Works with multiple backends, connectors, and storage options.
- Role-Based Access: Supports distinct profiles for end users and admins.

2.2 Core Concepts

- Retrieval-Augmented Generation (RAG) - A method where relevant documents are retrieved and combined with a user query before being processed by a language model, without the need to train the model.
- Embeddings - Numerical representations of text that allow semantic search.
- Vector Store - A database optimized for storing and retrieving embeddings.
- Connectors - Integrations that link the system to data sources (files, databases, APIs).

3. End User Guide

3.1 Accessing the System

3.1.1 Logging In

After your first login, the system will prompt you to change the default password for security.

Follow these requirements when creating your new password:

- Minimum 12 characters in length
- At least one digit (0-9)
- At least one uppercase letter (A-Z)
- At least one lowercase letter (a-z)
- At least one special character (for example: ! @ # $ % ^ & *)
- Must be different from your last five passwords

If the password you choose does not meet these rules, the system will display an error and you'll be asked to try again.

3.1.2 Navigating the Interface

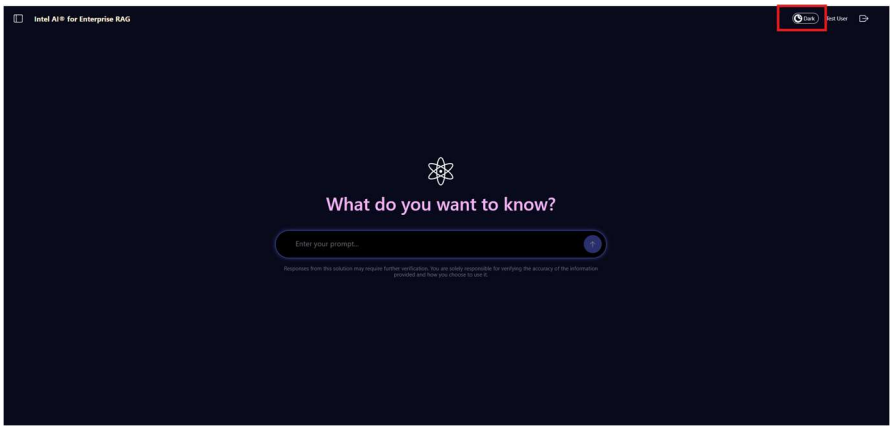
When you sign in, the application opens to a clean, search-focused dashboard designed to help you ask questions immediately.

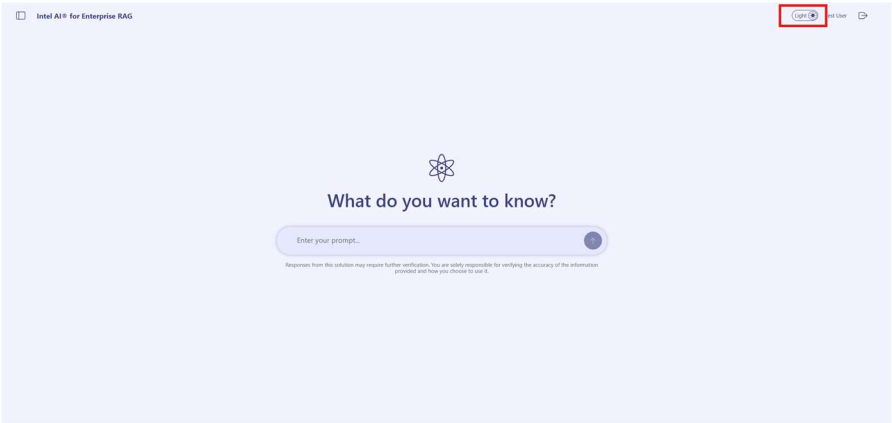
Top Left Navigation

- The Intel® AI for Enterprise RAG logo appears in the upper-left corner for easy identification.
- **Open Side Menu** - Far left icon allows you to open Chat History (conversation history) side menu.

Top Right Navigation

- **Theme Toggle** - The first upper-right corner switch lets you change between light and dark modes.





- **Username** - Next to the theme toggle, your username appears (for example, *Test User*).
- **Logout Icon** - A small arrow icon to the far right provides a quick way to sign out.

Main View
The center of the screen displays the prompt "What do you want to know?" above a single large search bar.
This is the area where you can enter a question in natural language.

A brief disclaimer beneath the bar reminds you that responses may require verification.

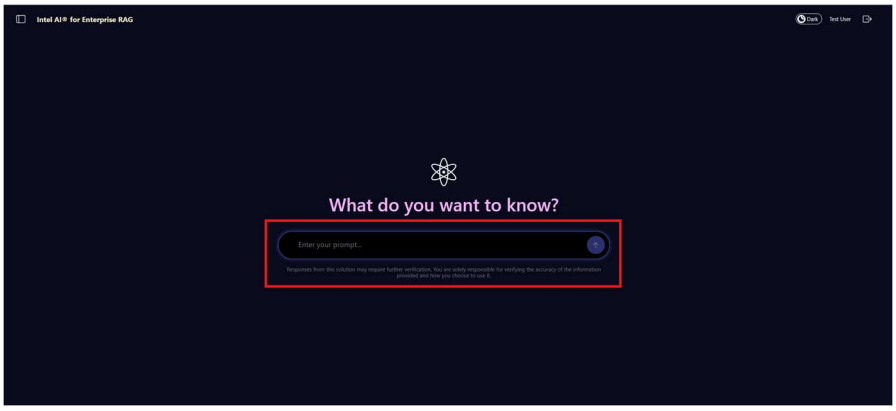
This minimal layout keeps the focus on your question, making it simple to start a search without distractions.

3.2 Asking Questions

3.2.1 Entering a Question

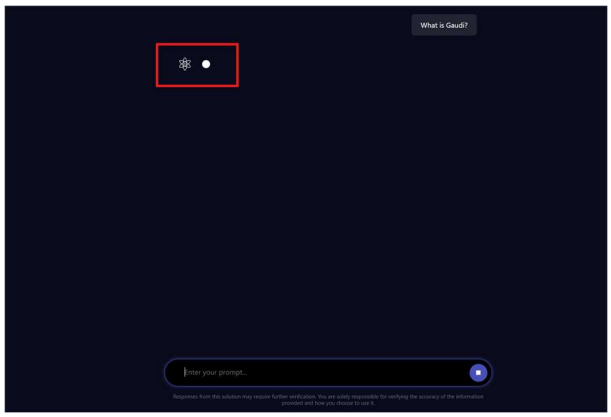
Type your question (prompt) into the text input field in the middle of the screen.

When ready, click the circle button with the arrow to send it.

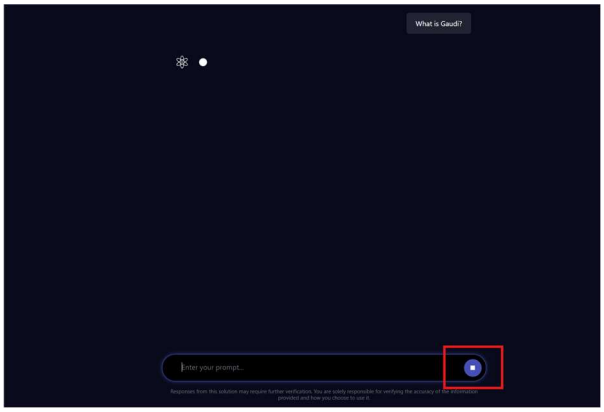


3.2.2 Waiting for a Response

After you submit your question, a **pulsing dot next to the atom icon** appears to show that the system is processing your request.



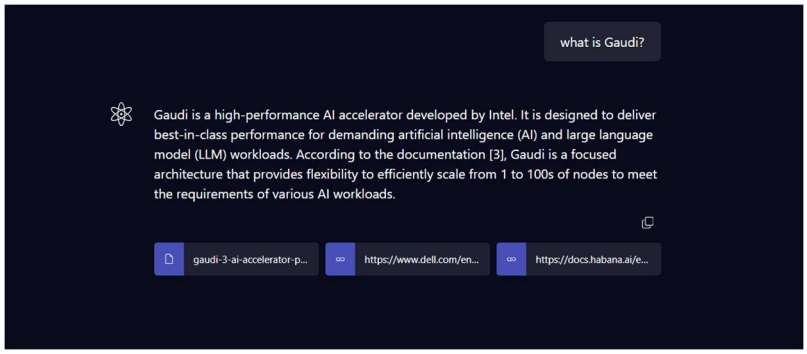
If you need to stop the request - for example, if you realize you want to rephrase - click the **Stop button** located at the bottom-right corner of the input field.



3.2.3 Reviewing the Answer

When the system provides a response, read it carefully and verify that it addresses your question.

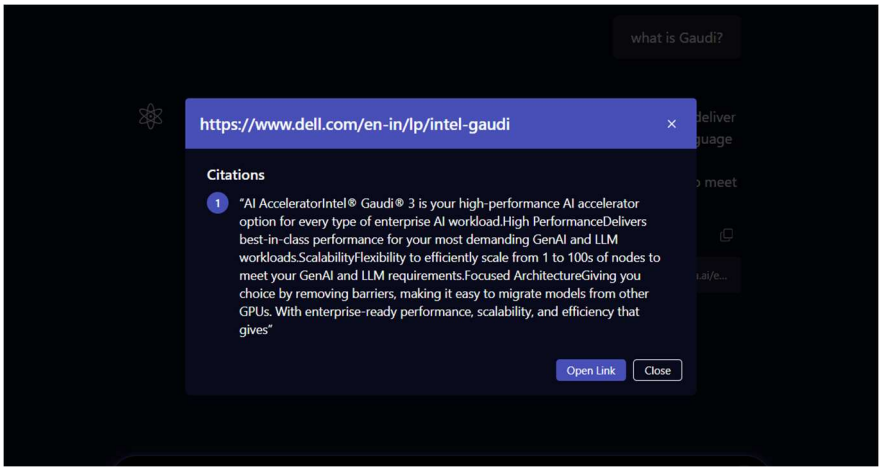
Each answer also includes sources (citations) the system used to generate the reply.



The sources are clearly listed, and each source shows the exact text chunks cited by the system.

If more than three sources are displayed for a chat response, a **Show less/Show all sources** button will appear. You can use this button to expand or collapse the list of sources for easier viewing.

Clicking on a source opens a window where you can view these chunks. Depending on the type of data source, the window provides either a **Download** button or an **Open Link** button, allowing you to save the sources for your own verification or record keeping. Use this information to validate the accuracy of the response and to explore the original reference materials if needed.



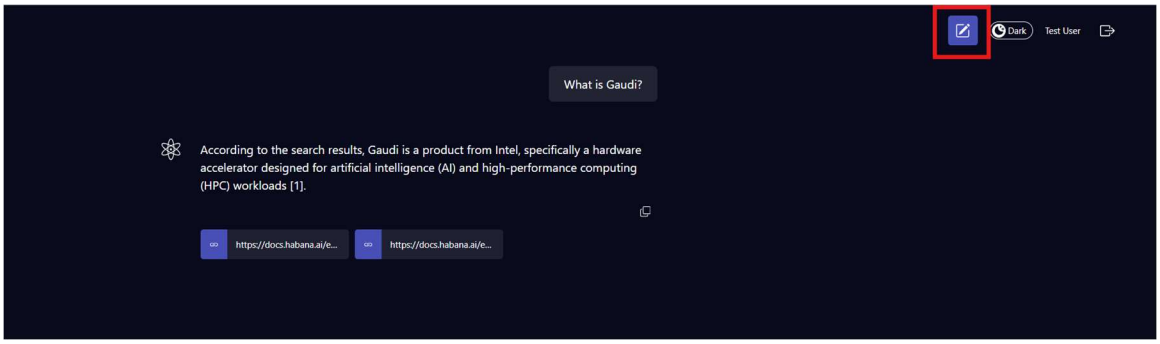
If necessary, you can enter a follow-up question or rephrase your prompt for more detail.

3.2.4 Starting a New Chat

To begin a different conversation, use the **Create New Chat** button in the upper-right corner of the screen.

This button appears only after a chat has been started or continued and is not visible on the initial or welcome page.

Click this button to clear the current conversation and open a fresh chat window.



Your previous conversation is automatically saved in **Chat History**, so you can return to it at any time.

This feature lets you move quickly from one topic to another while keeping earlier discussions safely stored.

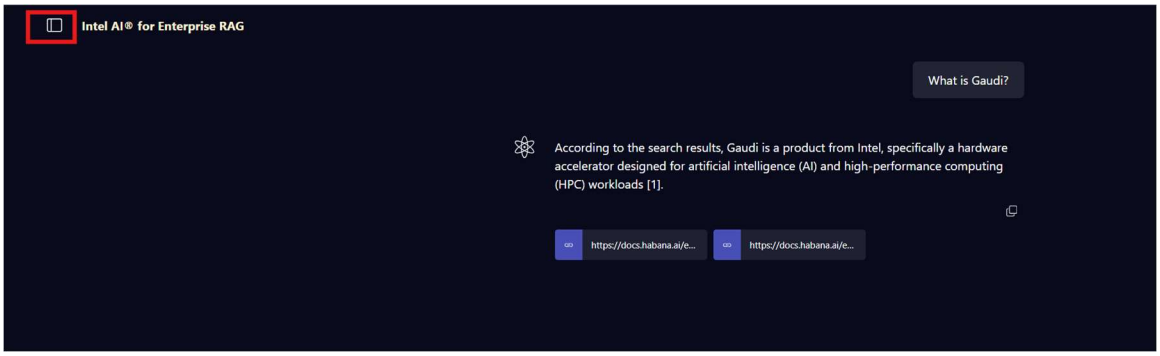
3.3 Chat History

Every conversation is automatically saved so you can revisit it later.

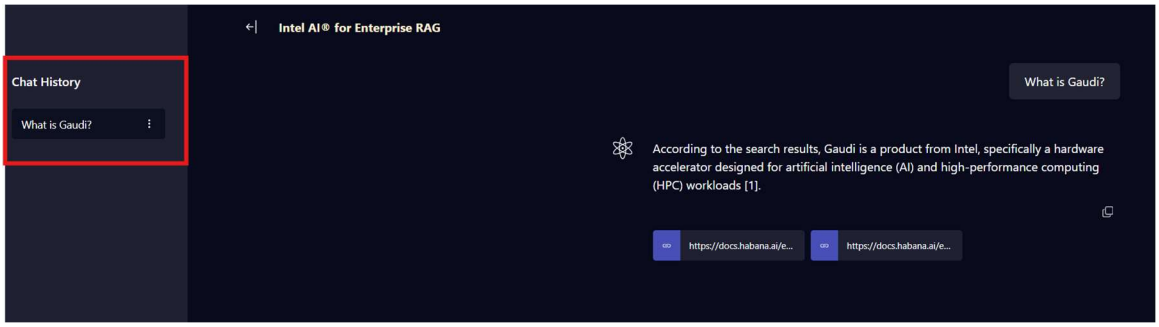
3.3.1 Opening Chat History

After you receive a response, your conversation is stored in **Chat History**.

Click the **Open Side Menu** in the top-left corner of the screen to open the panel.



The menu displays a list of all your saved chats.



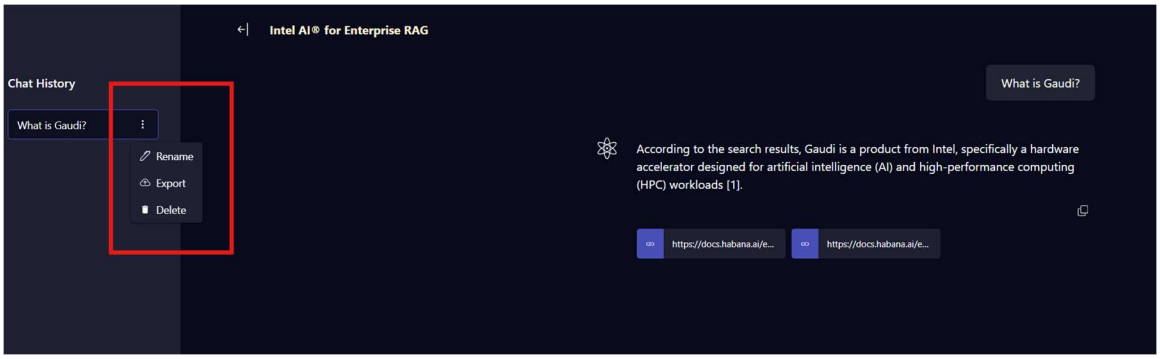
3.3.2 Managing a Chat

Locate the chat you want to manage and click the **three dots** next to its name.

A menu appears with these options:

- **Rename** - Give the chat a custom title for easier reference.
- **Export** - Download the entire conversation in JSON format for archiving, analysis, or debugging.
- **Delete** - Permanently remove the chat from your history.

Use these tools to keep your saved conversations organized and to back up important exchanges when needed.



4. Admin Guide

4.1 Accessing the Admin Interface

4.1.1 Logging In

Open https://erag.com in your browser.

You will be redirected to the Keycloak login page, where you must sign in with the one-time credentials generated during deployment.

These credentials are stored in the file: deployment/ansible-logs/default_credentials.txt

For the first time login use the appropriate set depending on the role:

- **Admin account:** KEYCLOAK_ERAG_ADMIN_USERNAME and KEYCLOAK_ERAG_ADMIN_PASSWORD
- **User account:** KEYCLOAK_ERAG_USER_USERNAME and KEYCLOAK_ERAG_USER_PASSWORD

After your first login, the system will prompt you to change the default password for security.

Follow these requirements when creating your new password:

- Minimum 12 characters in length
- At least one digit (0-9)
- At least one uppercase letter (A-Z)
- At least one lowercase letter (a-z)
- At least one special character (for example: ! @ # $ % ^ & *)
- Must be different from your last five passwords

If the password you choose does not meet these rules, the system will display an error and you'll be asked to try again.

4.1.2 Navigating the Admin Interface

When you sign in, the application opens to a clean, search-focused dashboard designed to help you ask questions immediately.

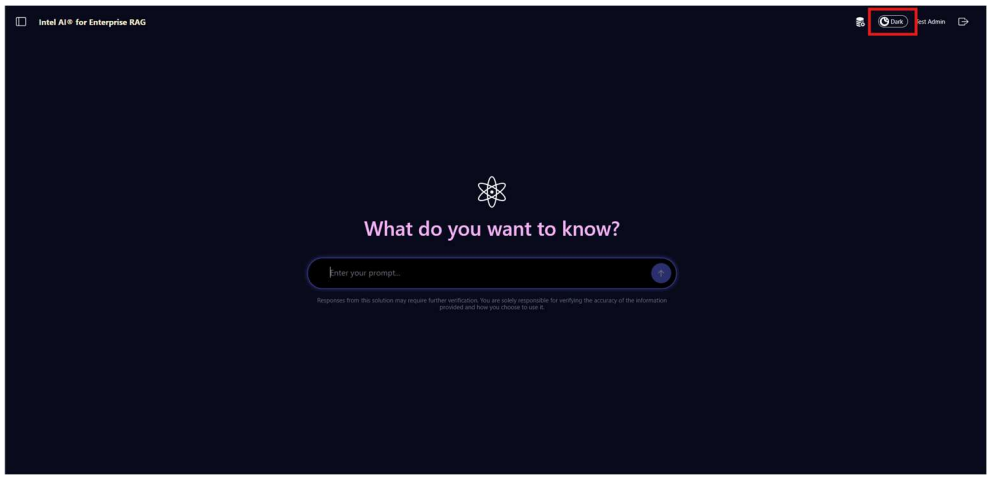
Top Left Navigation

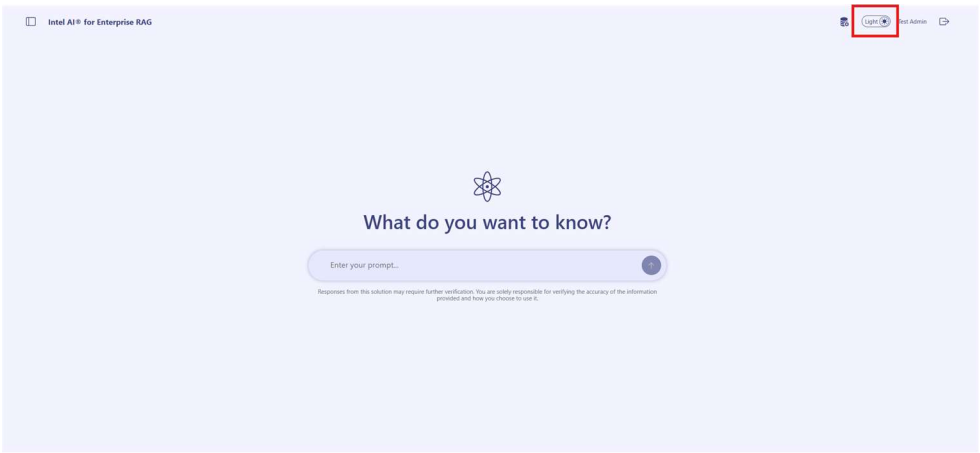
- The Intel® AI for Enterprise RAG logo appears in the upper-left corner for easy identification.
- **Open Side Menu** - Far left icon allows you to open Chat History (conversation history) side menu.

Top Right Navigation

- **Switch to Admin Panel** - The first upper-right corner icon opens the administration dashboard for managing system configuration, handle data sources, and monitor activity.

- **Theme Toggle** - The switch lets you change between light and dark modes.





- **Username** - Next to the theme toggle, your username appears (for example, *Test Admin*).
- **Logout Icon** - A small arrow icon to the far right provides a quick way to sign out.

Main View
The center of the screen displays the prompt “What do you want to know?” above a single large search bar.
This is the area where you can enter a question in natural language.

A brief disclaimer beneath the bar reminds you that responses may require verification.

This minimal layout keeps the focus on your question, making it simple to start a search without distractions.

4.1.3 Admin Panel Views

Within the **Admin Panel**, you can switch between three primary views - **Control Plane**, **Data Ingestion**, and **Telemetry & Authentication** - to manage system configuration, handle data sources, and monitor activity.

4.2 Control Plane

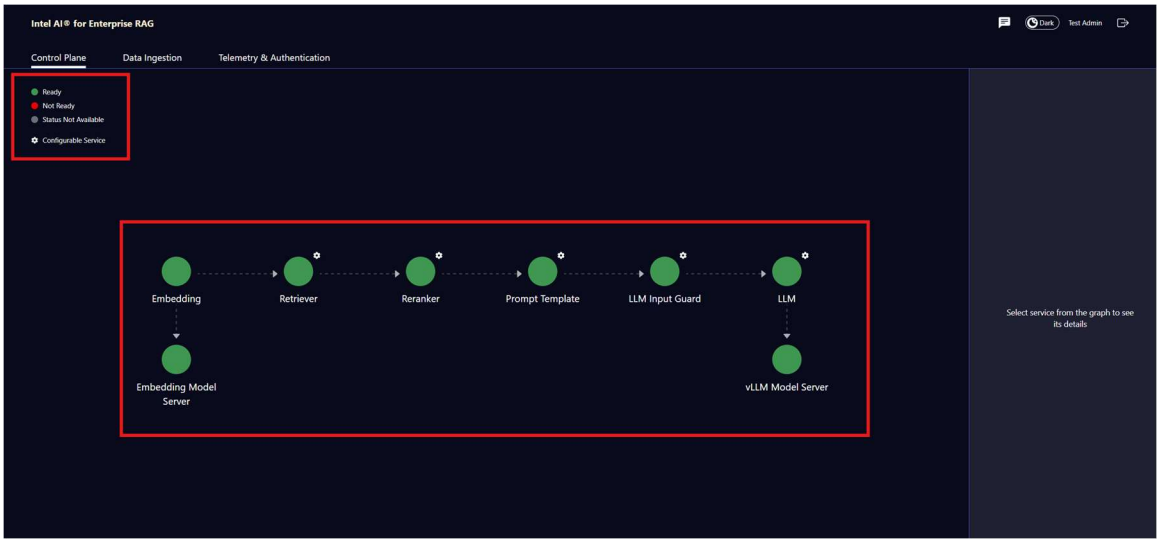
The **Control Plane** displays all components of the deployed pipeline as an interactive graph.

Each circle represents a running service - for example, Embedding, Retriever, Reranker, Prompt Template, LLM Input Guard, and LLM.

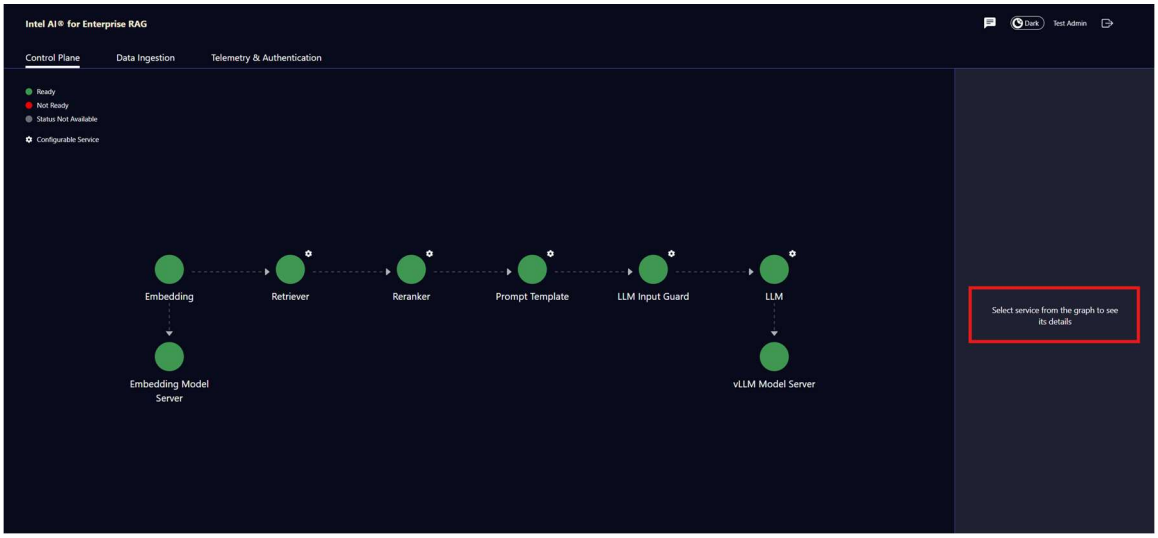
Services are connected in order that data flows through the pipeline, giving you a clear picture of the entire process.

A legend in the upper-left corner shows the health of each service:

- **Green - Ready:** The service is running normally.
- **Red - Not Ready:** The service is unavailable or has an error.
- **Gray - Status Not Available:** No status information is currently reported.



Services marked with a **gear-wheel icon** are configurable. For these components, you can adjust key parameters directly in the right-side panel to fine-tune chat behavior without redeploying the pipeline.



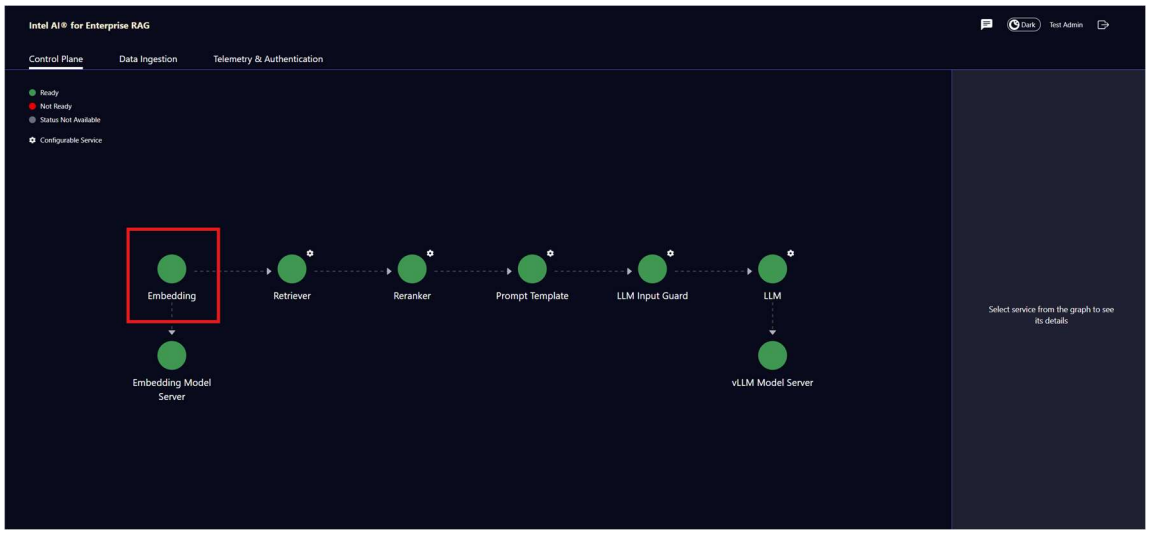
The Control Plane layout allows you, as an administrator, to monitor overall health at a glance, inspect individual services, and make targeted adjustments quickly and safely.

4.2.1 Embedding Service

The **Embedding** service is the first stage of the RAG pipeline.

- **Function** - It converts incoming text (from uploaded documents or other data sources) into *embeddings* - compact numerical vectors that capture the meaning of the text. These vectors make it possible to perform fast, accurate semantic search later in the pipeline.

- **Process** - Every document ingested into the system is passed through this service to produce embeddings that are stored in the vector database for retrieval.
- **Monitoring** - In the Control Plane you can view the Embedding service status as *Ready*, *Not Ready*, or *Status Not Available*. This lets you quickly confirm whether the service is operating normally.
- **Configuration** - No settings can be edited from the UI. Any changes to the embedding model or its parameters must be made during deployment or through infrastructure-level updates.
- **Relation to Other Components** - The Embedding service feeds directly into the Retriever node, which uses these stored vectors to locate the most relevant pieces of information for each query.

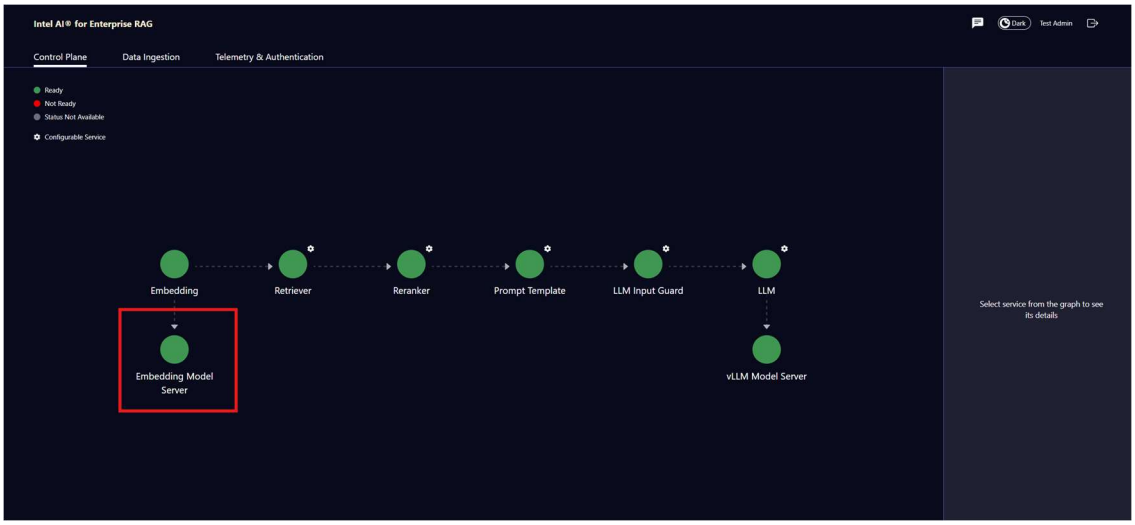


4.2.2 Embedding Model Server

The **Embedding Model Server** generates numerical representations ("embeddings") that allow the system to compare text by meaning rather than exact words. It supports the Embedding service by turning both your source documents and user questions into vectors that the Retriever can quickly match for relevance.

Behind the scenes, the server runs on TorchServe and hosts an embedding model optimized for Xeon processors, ensuring fast and efficient processing.

- **Purpose** - Provides the raw embeddings that power semantic search across all ingested data.
- **Role in the pipeline** - Runs behind the Embedding node, supplying vector data to the Retriever so it can locate the most relevant document chunks.
- **Configuration** - Read-only. No settings can be changed from the Control Plane.

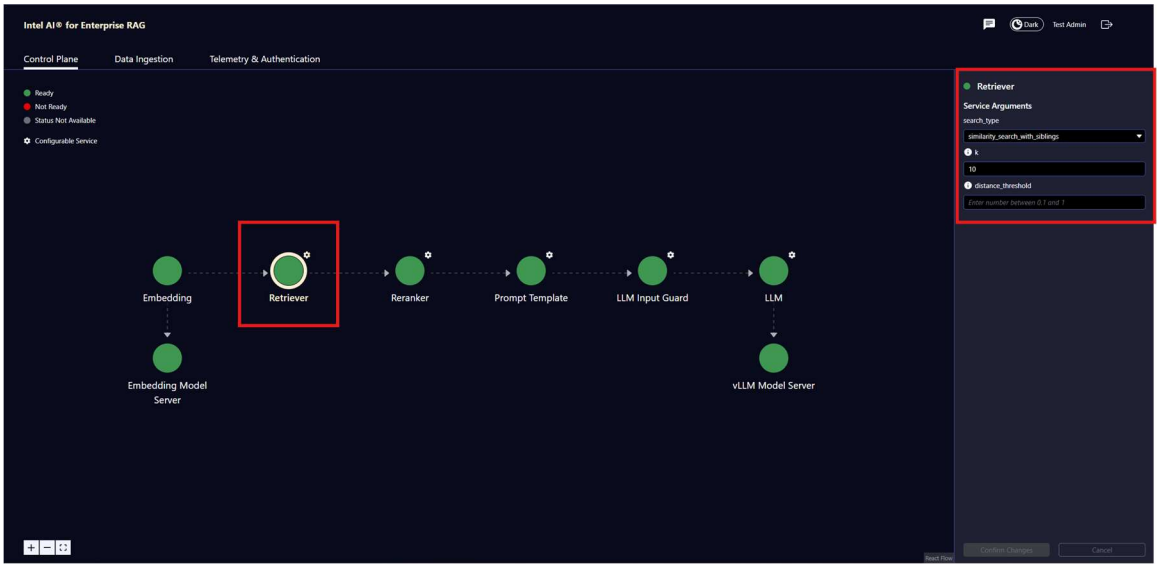


4.2.3 Retriever Service

The **Retriever** service locates the most relevant information from connected data sources before a response is generated.

Accessing Retriever Settings

- In the **Control Plane** graph, click the **Retriever** node.
  The node will be highlighted, and its configuration panel opens on the right.



Service Arguments

- **search_type** - Determines how the system selects and ranks documents for each query.

  Choose the option that best fits your needs:

    - **similarity** - Returns the top *k* documents that are most similar to the user's question.
      Use this when you want straightforward "closest match" results.
    - **similarity_search_with_siblings** - Finds the most similar chunk and also retrieves the immediate chunks that come before and after it in the original source.
      Helpful when context around the main match is important.
    - **similarity_distance_threshold** - Retrieves all documents that meet a minimum similarity score you define with the distance_threshold setting (value between 0.1 and 1).
      Ideal when you prefer to capture every document above a certain relevance level instead of a fixed number of top results.

- **k** - Sets the number of top results to return.
- **distance_threshold** - Available with *similarity_search_with_siblings* and *similarity_distance_threshold*; controls how strict the similarity requirement is (lower numbers return fewer, more closely matched results).

Editing and Saving Changes

1. Make the desired adjustments to *search_type*, *k*, or *distance_threshold*.
2. Click the green **Confirm Changes** button to apply your updates.
   The changes take effect immediately - no pipeline redeployment is required.
3. To discard edits, click **Cancel**.

This process lets you fine-tune how much supporting context is gathered for each query.

4.2.4 Reranker Service

The **Reranker** service evaluates the candidate documents returned by the Retriever and reorders them by relevance before they are sent to the language model.

Fine-tuning its parameters lets you control how many top results move forward and how strictly they are filtered.

It runs on a **TorchServe** inference server, which hosts the trained reranking model and serves predictions in real time.

Accessing Reranker Settings

- In the **Control Plane** graph, click the **Reranker** node.
  The node will be highlighted, and a configuration panel appears on the right side of the screen.

Service Arguments

- **top_n** - Specifies the number of highest-scoring documents that should be kept after reranking.
  Example: a value of 3 means only the three most relevant results will be passed on.
- **rerank_score_threshold** - Sets a minimum relevance score (for example, 0.02). Only documents scoring at or above this threshold are retained, even if the *top_n* value is larger.
  Raising this value makes the filter stricter; lowering it allows more documents through.

Editing and Saving Changes

1. Adjust *top_n* and/or *rerank_score_threshold* as needed.
2. Click the green **Confirm Changes** button to save and apply the updates immediately - no pipeline redeployment is required.
3. Click **Cancel** to discard any unsaved edits.

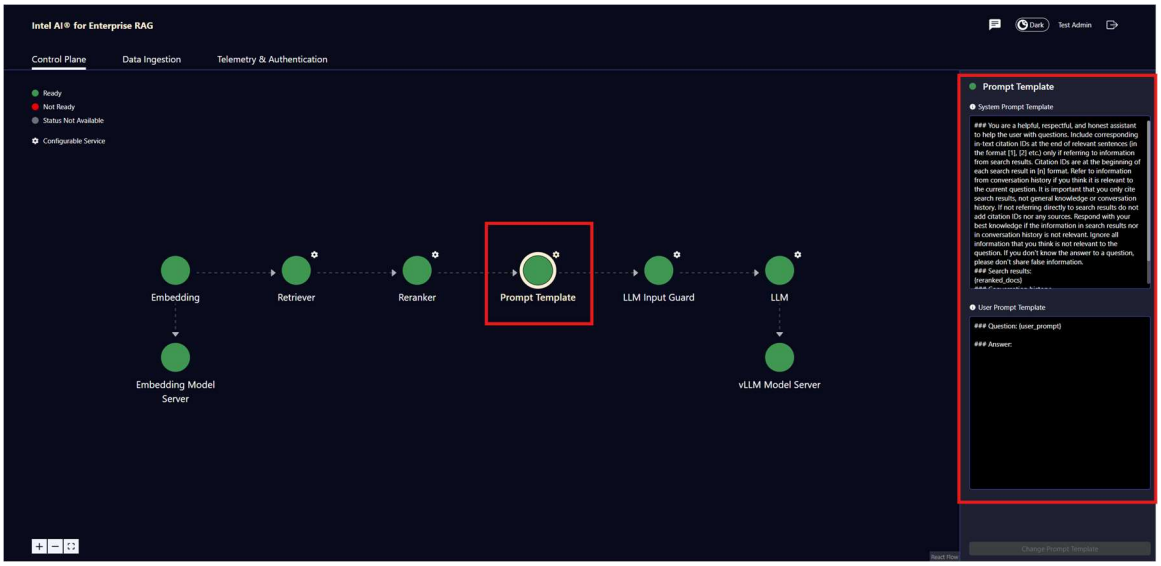
This configuration gives you precise control over the **quality and quantity** of information that reaches the next stage of the pipeline.

4.2.5 Prompt Template Service

The **Prompt Template** defines the instructions and format given to the language model (LLM) whenever it generates a response.

Accessing Prompt Template Settings

- In the **Control Plane** graph, click the **Prompt Template** node.
  The node will be highlighted, and a configuration panel appears on the right side of the screen.



Editable Fields

- **System Prompt Template** - Contains the core instructions sent to the model before every query.
  This section defines the assistant's behavior and the rules for citations and formatting.

  By default, it includes detailed guidance such as:
  - how to insert citation IDs at the end of relevant sentences when referring to search results,
  - when and how to use conversation history if it is relevant to the current question, and
  - directives to ignore unrelated information and avoid sharing unsupported content.

- **User Prompt Template** - Defines how the user's question is wrapped before it is sent to the model.
  The default format passes the user's prompt and reserves space for the model's answer.

Editing and Saving Changes

1. Click inside the **System Prompt Template** or **User Prompt Template** field and make the desired edits.
   As soon as you modify either field, the **Change Prompt Template** button at the bottom of the panel becomes active.
2. When you are ready to apply your updates, click **Change Prompt Template**.
   - The changes are saved immediately - there is no additional confirmation step.
   - After saving, the interface automatically returns you to the main **Control Plane** view.
3. To discard edits instead of saving them, simply leave the panel without clicking **Change Prompt Template**.

Adjusting these templates lets you, as an administrator, control the tone, content rules, and structure of the answers provided to end users.

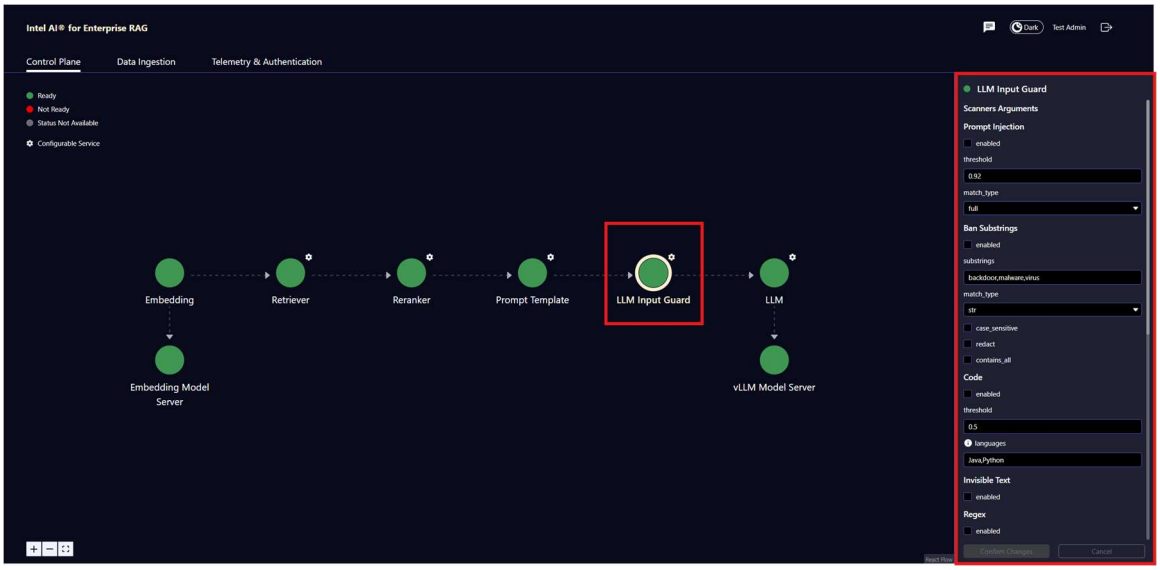
4.2.6 LLM Input Guard

The **LLM Input Guard** is a protective layer that reviews every user query before it reaches the main language model.

Its role is to filter or flag inputs that might cause unwanted behavior - such as prompts containing disallowed content or patterns known to produce unreliable answers.

Accessing LLM Input Guard Settings

- In the **Control Plane** graph, click the **LLM Input Guard** node.
  The node will highlight, and a configuration panel opens on the right side of the screen.



Some scanners use AI models, while others are purely algorithmic:

- **AI-powered scanners** use machine learning to detect complex patterns or context-dependent issues that may not be obvious from simple rules: Prompt Injection, Code, TokenLimit, Toxicity.
- **Algorithmic scanners** rely on predefined rules and patterns to catch straightforward or well-defined issues: BanSubstring, InvisibleText, Regex, Secrets, Sentiment.

Prompt Injection Service Arguments

The Prompt Injection scanner protects the system from malicious or misleading instructions that might be hidden inside a user's query or retrieved documents. When enabled, it checks every incoming prompt against detection rules and blocks or sanitizes suspicious input before it reaches the language model.

- **enabled** - Toggle to turn the scanner on or off.
  Checked means protection is active; *unchecked* disables scanning.

- **threshold** - Sets the sensitivity of detection (for example, 0.92).
  Higher values make detection stricter, reducing false positives but possibly missing subtle attacks.
  Lower values increase sensitivity, catching more potential issues but risking unnecessary blocking.
- **match_type** - Controls how the scanner compares text to its detection patterns.

  Options:
  - **full** - Evaluates the entire prompt as a single block.
  - **sentence** - Checks each sentence individually.
  - **truncate_token_head_tail** - Examines both the start and end of the prompt, where attacks often appear.
  - **truncate_head_tail** - Similar to the above but trims by character count rather than tokens.
  - **chunks** - Splits the prompt into segments and evaluates each one separately.

Ban Substrings Service Arguments
The **Ban Substrings** scanner blocks any prompt that contains specific words or phrases you define. This is useful for filtering out sensitive terms - such as security exploits or prohibited topics - before they reach the language model.

- **enabled** - Turns this filter on or off.
  Checked means scanning is active; *unchecked* disables it.
- **substrings** - A comma-separated list of words or phrases to block.
  Example: *backdoor,malware,virus* prevents prompts containing any of those terms.
- **match_type** - Determines how matching is performed:
  - **str** - Direct string match.
  - **word** - Matches complete words only, ignoring partial matches inside longer words.
- **case_sensitive** - When enabled, matches only if the capitalization is exactly the same as listed in **substrings**. Leave unchecked to ignore case.
- **redact** - Specifies how blocked terms are handled.

Checked means matching terms are replaced with a placeholder (e.g., ***);
unchecked blocks the entire prompt if a match is found.

- **contains_all** - Specifies whether all listed substrings must appear to trigger the filter.
  Checked means action is taken only if every listed substring is present in the prompt;
  unchecked means action is taken if any listed substring appears.

Code Service Arguments

The **Code** scanner detects programming code within user prompts and can restrict or monitor specific programming languages. This helps prevent accidental or unauthorized submission of code snippets or allows the system to flag and handle them differently.

- **enabled** - Turns this scanner on or off.
  Checked means the system will analyze incoming prompts for code; *unchecked* disables detection.
- **threshold** - Sets the sensitivity for detecting code-like patterns (e.g., 0.5).
  Higher values make detection stricter, reducing false positives but possibly missing subtle code fragments.
  Lower values increase sensitivity, catching more potential code at the risk of false positives.
- **languages** - A comma-separated list of programming languages to monitor. The tooltip lists all supported programming languages, so you can quickly verify which language names can be used in this argument.
  Example: *Java,Python* will specifically check for prompts containing Java or Python code.
  Leave blank to scan for any language.

Invisible Text Service Argument
The **Invisible Text** scanner detects hidden, or zero-width characters often used to smuggle instructions or data into a prompt.

- **enabled** - Turns the scanner on or off.
  Checked means the scanner is active and will flag or block inputs containing invisible characters; *unchecked* disables this check and allows all text, even if it contains hidden characters.

Regex Service Arguments
The **Regex** scanner uses regular expressions to detect custom text patterns in user prompts. It's ideal for spotting sensitive formats such as API keys, access tokens, or other strings that match a defined pattern.

- **enabled** - Turns this scanner on or off.
  Checked means prompts are scanned for the specified regex patterns; *unchecked* disables it.
- **patterns** - One or more regular expressions used to find matches.
  Example: *Bearer [A-Za-z0-9._~+/-]+* flags strings that look like Bearer tokens.
- **match_type** - Controls how the scanner evaluates the regex against the prompt:
  - **search** - Returns a match if the pattern appears anywhere within the text.
  - **fullmatch** - Requires the entire text to match the pattern.
  - **all** - If multiple patterns are listed, all of them must match for the action to trigger.
- **redact** - Controls how matched text is handled.
  If *checked*, the scanner redacts the matched content by replacing it with a placeholder (such as ***), and the prompt is allowed to proceed. *Unchecked* blocks the entire prompt if a regex match is found.

Secrets Service Arguments

The **Secrets** scanner monitors incoming text for patterns that resemble sensitive credentials - such as API keys, passwords, or tokens - and can automatically redact them. Enable this scanner to prevent accidental exposure of confidential information in prompts.

- **enabled** - Turns this scanner on or off.
  Checked means the scanner is active and will inspect inputs for potential secrets; *unchecked* disables this check.
- **redact_mode** - Determines how detected secrets are handled:
  - **all** - Entire detected secret is replaced with a placeholder.
  - **partial** - Only part of the detected secret is masked (for example, showing the last few characters).
  - **hash** - Replaces the secret with a secure hash value instead of the raw text.

Sentiment Service Arguments

The **Sentiment** scanner monitors user input for strongly negative sentiment. Use this scanner if you need to detect and respond to messages that carry aggressive, harmful, or otherwise negative tone before they reach the model.

- **enabled** - Turns this scanner on or off.
  Checked means the sentiment scanner is active; *unchecked* disables this check.
- **threshold** - Sets the negativity score (0 to 1) required to trigger a match.
  Lower values (for example, 0.2) flag milder negative content.
  Higher values (for example, 0.5) only flag strongly negative input.

Token Limit Service Arguments

The **Token Limit** scanner enforces a maximum token count for incoming text. Use this scanner to prevent extremely long prompts that might strain the model or system resources.

- **enabled** - Turns this scanner on or off.
  Checked means the scanner is active and will block or flag any request that exceeds the token limit; *unchecked* disables token limit checks.
- **limit** - The maximum number of tokens allowed (for example, 4096). Tokens roughly correspond to chunks of text; higher limits permit longer input.

Toxicity Service Arguments
The **Toxicity** scanner detects harmful or offensive language. Enable it to prevent toxic or abusive language from reaching the model.

- **enabled** - Turns this scanner on or off.
  Checked means the scanner is active and will evaluate input for toxic content; *unchecked* disables toxicity checks.
- **threshold** - Confidence level (for example, 0.5) that determines when text is flagged as toxic. Lower values make detection more sensitive.
- **match_type** - How the input is evaluated:
  - full - Flags the entire input if overall toxicity exceeds the threshold.
  - sentence - Flags individual sentences.

Editing and Saving Changes

1. Adjust any values as needed.
2. Click **Confirm Changes** to apply updates immediately - no pipeline redeployment required.
3. Click **Cancel** to discard unsaved edits.

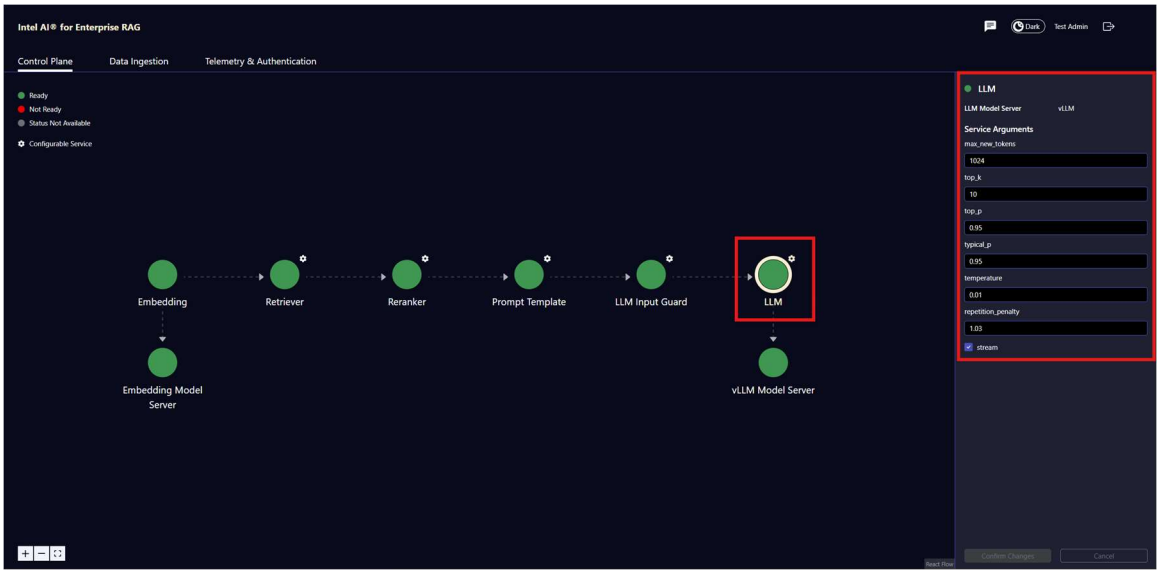
4.2.7 LLM Service

The **LLM Service** hosts the core language model that generates final responses for user queries.

It runs on a **vLLM model server**, which provides high-performance inference and supports live streaming of tokens for faster, more interactive replies.

Accessing LLM Settings

- In the **Control Plane** graph, click the **LLM** node.
  The node will highlight, and a configuration panel opens on the right side of the screen.



Service Arguments
Each field lets you fine-tune how the language model creates text:

- **max_new_tokens** - Maximum number of tokens the model can generate in a single response. Higher values allow longer answers.
- **top_k** - Limits sampling to the top *k* most probable next tokens. Lower values make responses more focused; higher values increase variety.
- **top_p** - Nucleus sampling threshold (0–1). The model considers the smallest set of tokens whose combined probability is at least *p*. Lower values narrow the range of possible outputs.
- **typical_p** - Similar to top_p, but balances probability mass with typicality to avoid overly predictable text.
- **temperature** - Controls randomness. Lower than 1.0 makes output more deterministic; higher than 1.0 adds creativity.
- **repetition_penalty** - Penalizes repeating the same phrases. Values above 1 reduce repetition.
- **streaming** - If enabled, the model sends tokens as they are generated so users see the answer appear in real time.

Editing and Saving Changes

1. Adjust any values as needed.
2. Click **Confirm Changes** to apply updates immediately - no pipeline redeployment required.
3. Click **Cancel** to discard unsaved edits.

Use these controls to balance speed, creativity, and reliability of responses according to your deployment's needs.

4.2.8 vLLM Model Server

The **vLLM Model Server** is the engine that runs the large language model (LLM) at the end of the pipeline.

- **Function** - It receives the fully prepared prompt - your user's question combined with retrieved documents and any system instructions - and generates the final answer.
- **Performance** - vLLM is built for fast response and efficient memory use, allowing low-latency replies even when many users are active.
- **Monitoring** - In the Control Plane you can see the vLLM service status as *Ready*, *Not Ready*, or *Status Not Available*. This lets you confirm at a glance that the model server is healthy.
- **Configuration** - This service cannot be edited from the UI. Any model updates or scaling changes must be made during deployment or through infrastructure settings outside the Control Plane.

4.3 Data Ingestion

The **Data Ingestion** tab gives you a clear view of all files and web links that have been synchronized and processed into the RAG knowledge base.

The interface is designed for lightweight administrative management - monitoring ingestion jobs, checking processing results, and adding small sample files or links when needed.

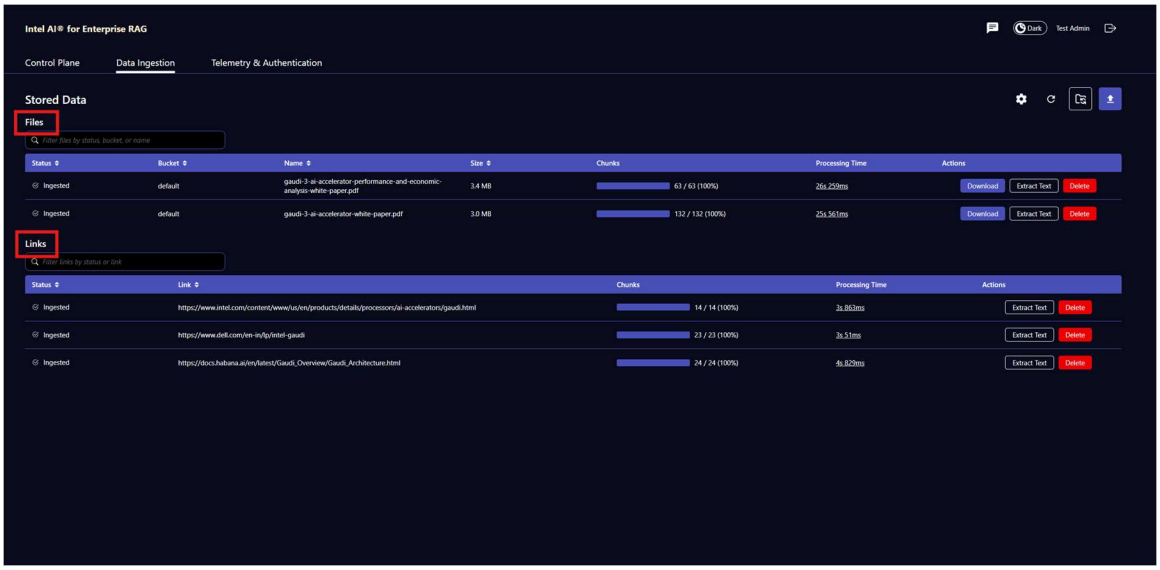
For best performance and reliability, files should be uploaded directly to the configured storage endpoint (e.g., S3 bucket) rather than through the UI. The UI is intended primarily for small sample files or occasional uploads.

The main panel is split into two sections:

1. **Files** - Lists tracked documents. Columns include:
    - **Status** - e.g., *Ingested* when processing is complete.
    - **Bucket** - The S3 bucket or endpoint where the file resides.
    - **Name** - File name.
    - **Size** - File size.

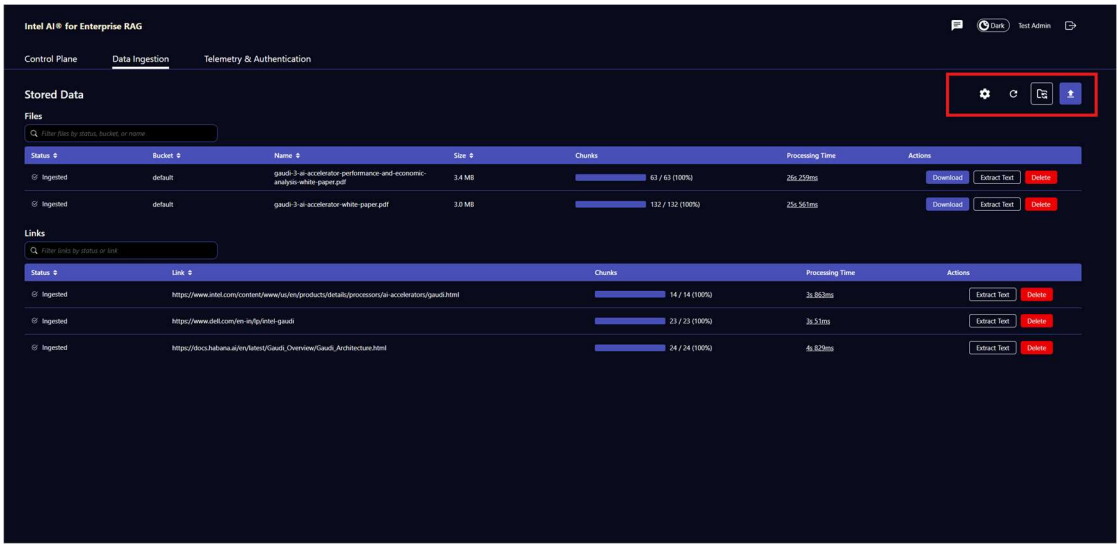
- **Chunks** - Number of text chunks created during processing, with a completion percentage.
- **Processing Time** - Total time to process the file.
- **Actions** - Options to download the processed file or delete it from the system.

2. **Links** - Displays ingested web pages. Columns include:
    - **Status** - e.g., *Ingested* when processing is complete.
    - **Link** - The URL of the ingested page.
    - **Chunks** - Number of text chunks created during processing, with a completion percentage.
    - **Processing Time** - Total time required to process the file.
    - **Actions** - Option to delete the ingested page from the system.



Data Ingestion Action Buttons
In the upper-right corner of the **Data Ingestion** page you'll find four buttons you can use to manage ingestion tasks:

- **Settings** - Open configuration options for data ingestion, such as bucket connections and other ingestion parameters.
- **Refresh Data** - Reload the file and link tables to see the latest ingestion status without refreshing the entire page.
- **Synchronize Buckets** - Manually trigger a check of all configured S3 buckets so any new or updated files are detected and registered.
- **Upload Data** - Open the upload dialog to add a supported file or provide a web link to ingest a single document or page.

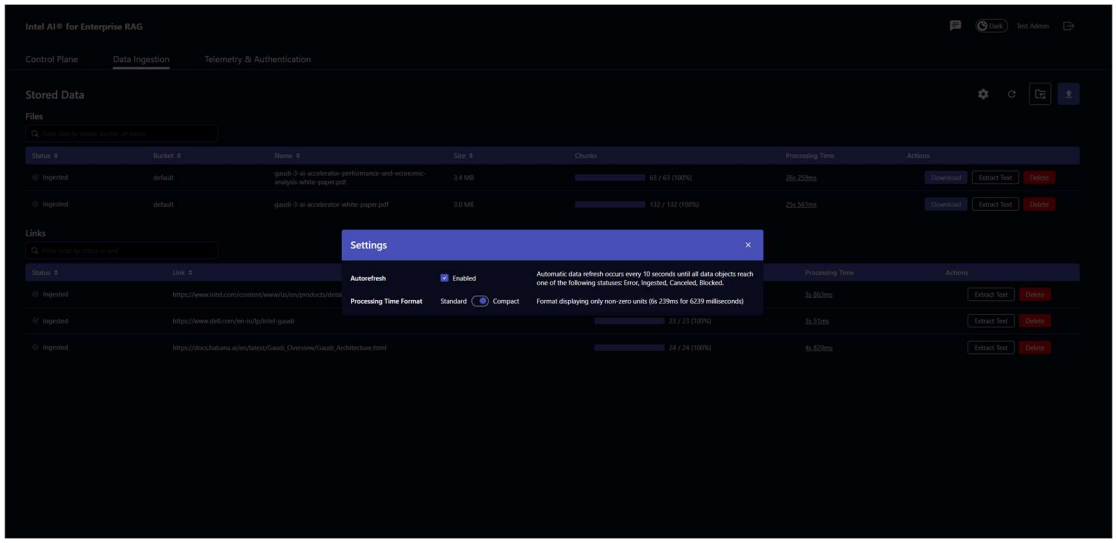


4.3.1 Settings

Open the **Settings** panel (gear icon in the top-right corner of the Data Ingestion page) to control how the ingestion table refreshes and how processing times are displayed.

Options

- **Autorefresh**
  - **Enabled** - The Data Ingestion tables automatically refresh every 10 seconds until each file or link reaches a final status (Error, Ingested, Canceled, or Blocked).
  - **Disabled** - You must manually click **Refresh Data** to update the tables.
- **Processing Time Format**
  - **Standard** - Shows full duration values, including all units.
  - **Compact** - Displays only non-zero units (for example, 6s 239ms instead of 0h 0m 6s 239ms).

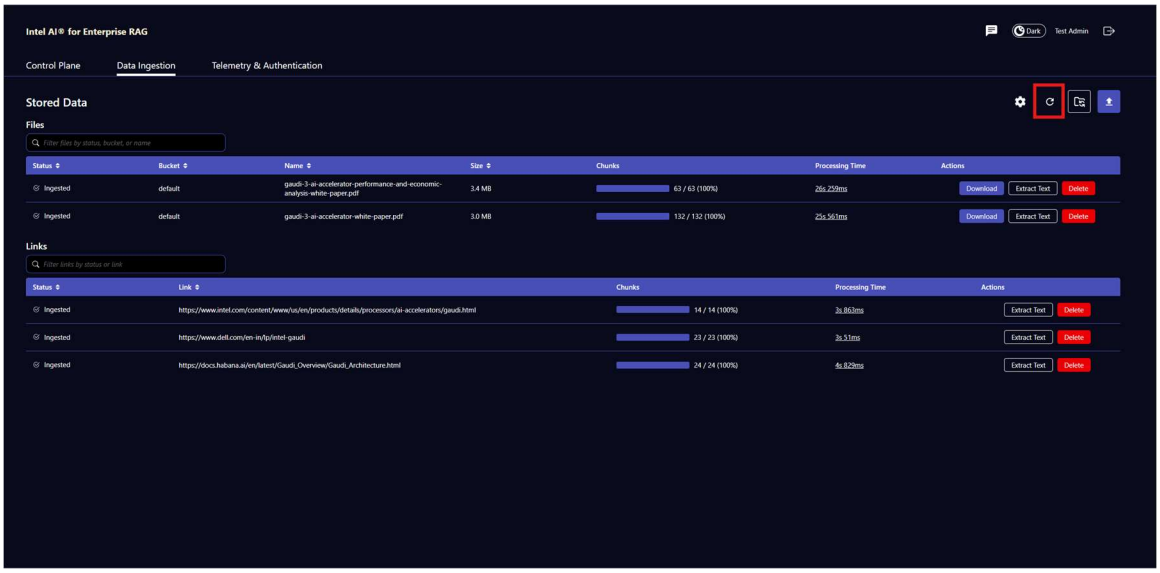


4.3.2 Data Refresh

Use **Refresh Data** button to manually reload the Data Ingestion table.

Click Refresh Data immediately requests the latest status from the backend and updates the Files and Links lists with any new processing results, chunk counts, or errors.

This is useful if you have recently uploaded files or links and want to see their current ingestion state without waiting for the next automatic refresh.



4.3.3 Buckets Synchronization

Use **Synchronize Buckets** to manually update the RAG knowledge base so it matches its configured S3 buckets. This is useful in cases where bucket notifications are not available on the S3 storage endpoint.

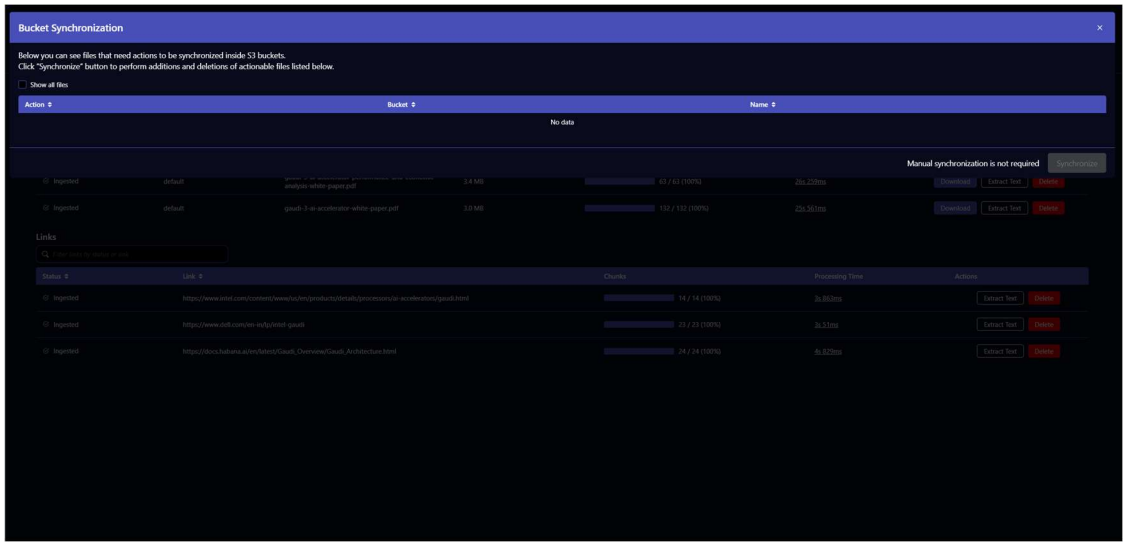
How it works

- The table lists any files that require action to match the contents of the connected S3 buckets.
- For each file you see the **Action** (add or remove), the **Bucket** name, and the file **Name**.
- Selecting **Show all files** reveals every file in scope, including those that are already synchronized.

To perform a manual sync

1. Open the dialog by clicking **Synchronize Buckets** in the top-right corner of the Data Ingestion tab.
2. Review the listed items.
3. Click **Synchronize** to apply all required additions or deletions.

If there are no differences between the S3 buckets and the local index, the table will be empty, and the message *Manual synchronization is not required* appears. In this state, the **Synchronize** button remains disabled.



4.3.4 Data Upload

Use the **Upload Data** to add files or web links directly into the RAG knowledge base.

Files

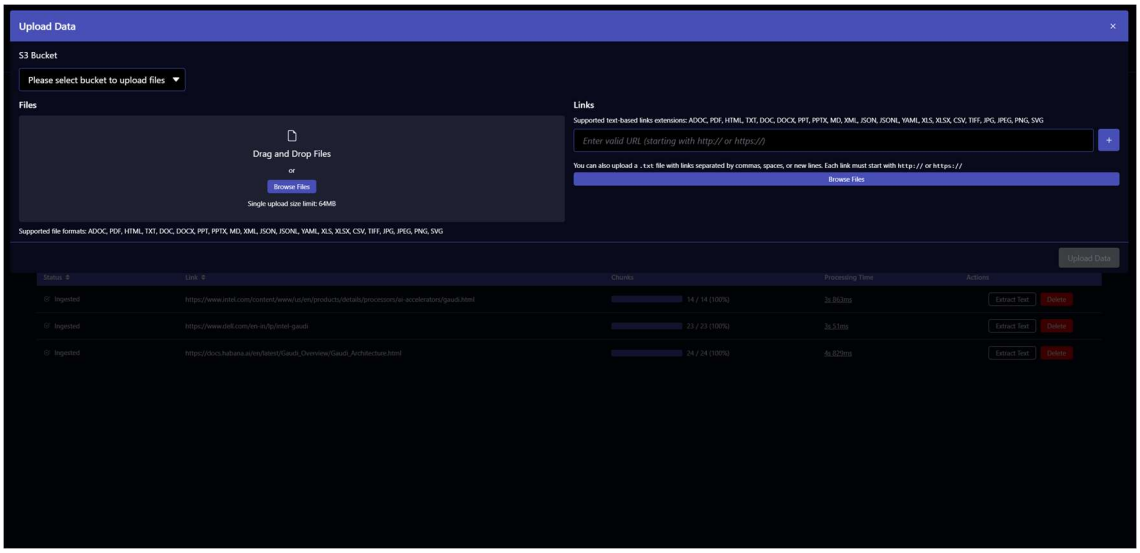
1. From the **S3 Bucket** drop-down, select the bucket where the new data will be stored.
2. Drag and drop files into the upload area or click **Browse Files** to pick them from your computer.
3. Supported formats include ADOC, PDF, HTML, TXT, DOC, DOCX, PPT, PPTX, MD, XML, JSON, JSONL, YAML, XLS, XLSX, CSV, TIFF, JPG, JPEG, PNG, and SVG.
4. The upload limit applies to a single request. Multiple files can be uploaded together as long as the total size does not exceed 64 MB.

Links

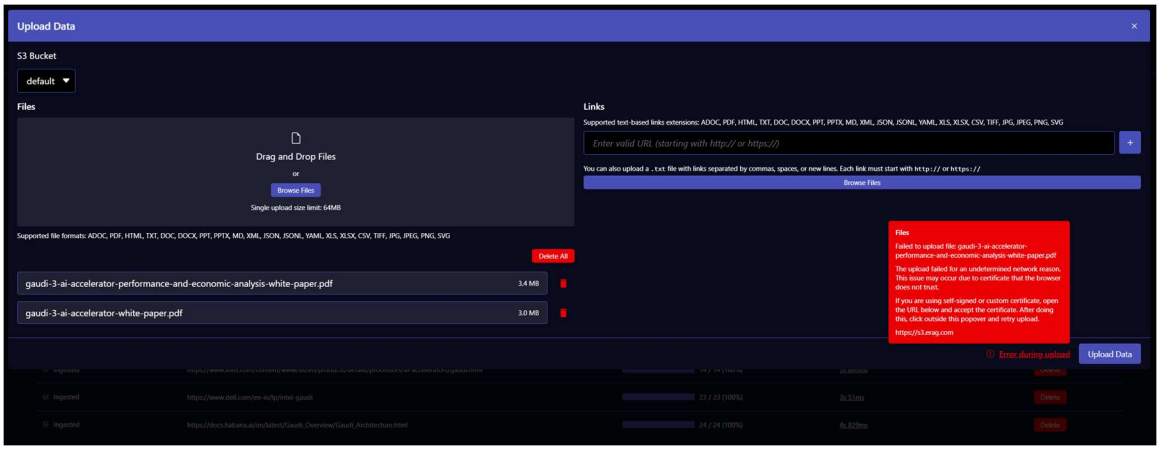
- Enter one or more valid URLs (starting with http:// or https://) in the text field and click the **+** button to add them.
- Alternatively, upload a .txt file that lists URLs separated by commas, spaces, or new lines.

After selecting all desired files or links, click **Upload Data** to start ingestion.

The system immediately begins processing and chunking the new content, and progress can be monitored from the main **Data Ingestion** view.



When you access a locally deployed pipeline that uses a self-signed certificate, an error about an **untrusted** or **self-signed certificate** will appear. This does not happen on production pipelines, which use trusted certificates.



To resolve this, just follow the instructions shown in the error message:

1. Click the URL shown in the error message.
2. Accept the certificate in your browser.
3. Return to the upload page and click **Upload Data** again.

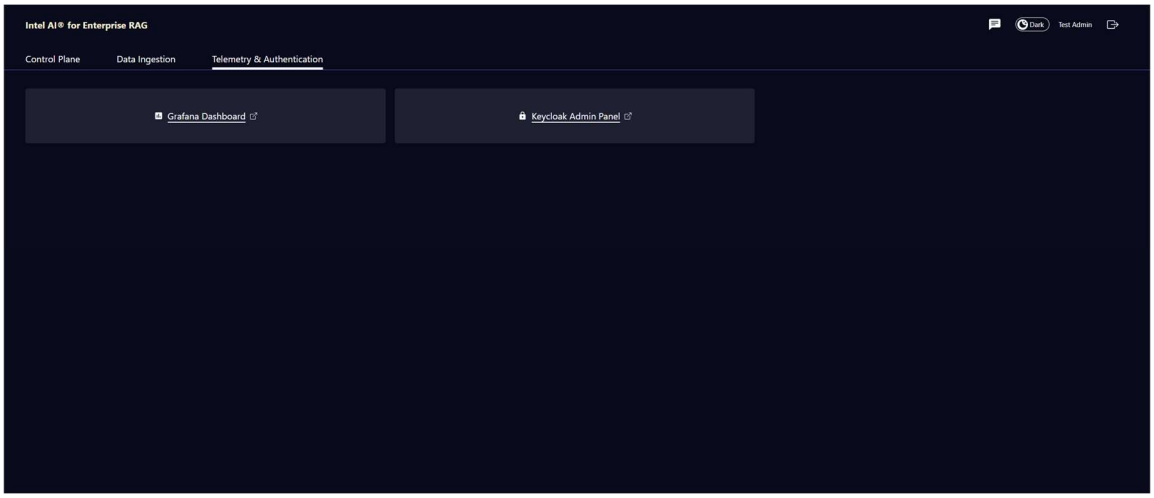
4.4 Telemetry & Authentication

The **Telemetry & Authentication** tab provides quick access to supporting services used for system monitoring and identity management.

- **Grafana Dashboard** - Opens a new browser tab with the Grafana interface, where you can view performance metrics, system health, and usage statistics.

- **Keycloak Admin Panel** - Opens a new browser tab with the Keycloak interface, allowing you to manage users, roles, and authentication settings.

Simply click the block for the service you want to use, and a new tab will launch automatically with the correct URL and credentials prompt.

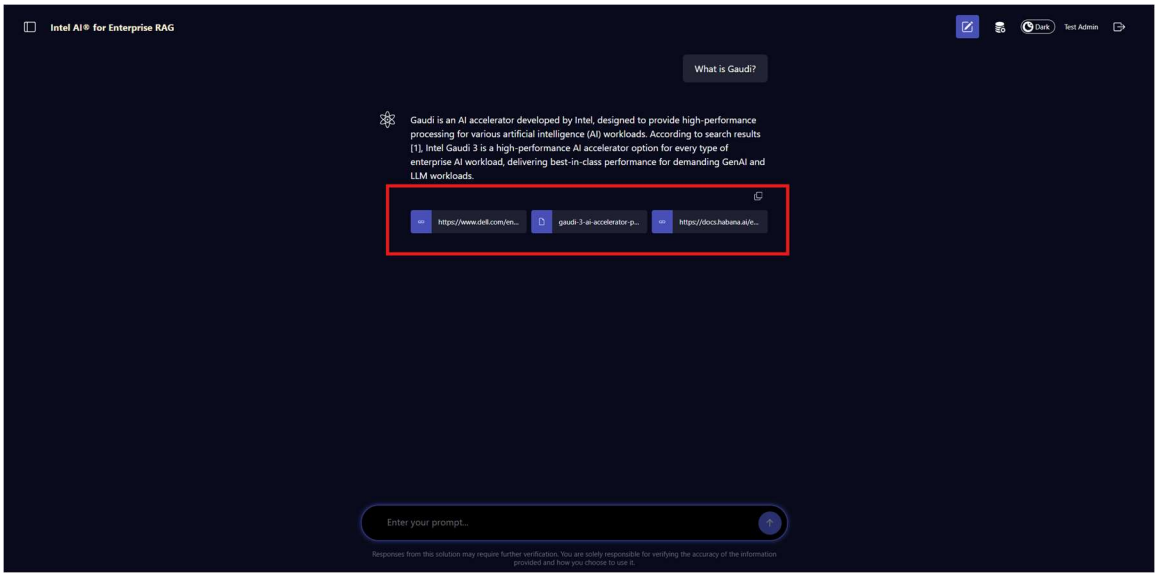


4.5 Troubleshooting

This section helps you identify and resolve common issues in the RAG system. It provides simple steps and visuals so you can monitor, inspect, and improve system behavior safely.

4.5.1 Debug Mode

When you ask a question in Chat Q&A, the system returns an answer along with references to the source documents. This allows you to verify information against the original sources.



If a question is unanswered or a response seems incorrect, you can use **Debug Mode** to inspect how documents are retrieved and ranked.

Debug Mode lets you:

- See how the system processes documents and retrieves knowledge.
- Review the RAG pipeline's "thought path."
- Experiment with pipeline parameters safely - changes do not affect production.

To enable Debug Mode:

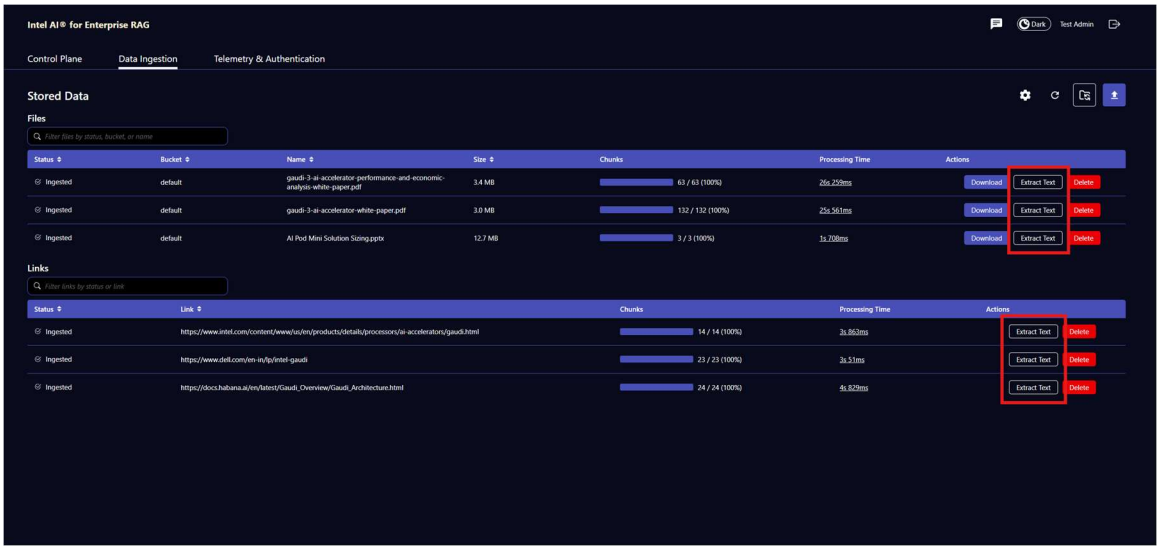
1. Open the Admin Panel in your browser.
2. Add **?debug=true** to the end of the URL and press Enter.
3. The page reloads with Debug Mode enabled.

4.5.2 Controlling Text Extraction and Chunking

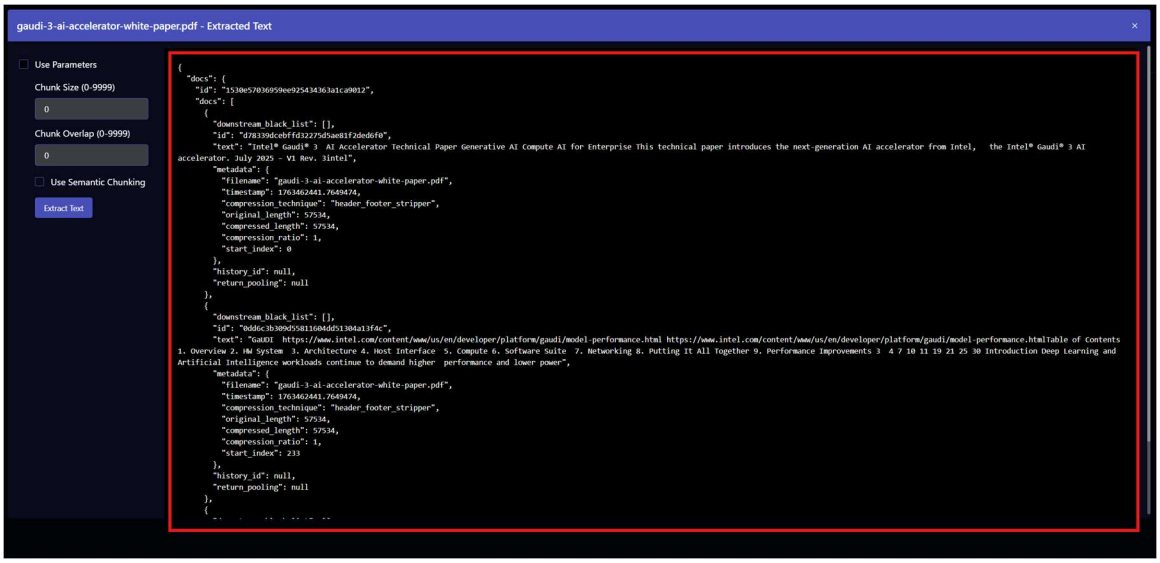
Use the Data Ingestion tab in the Admin Panel to review uploaded documents. In Debug Mode, you can inspect how text is extracted and chunked.

To inspect **text extraction:**

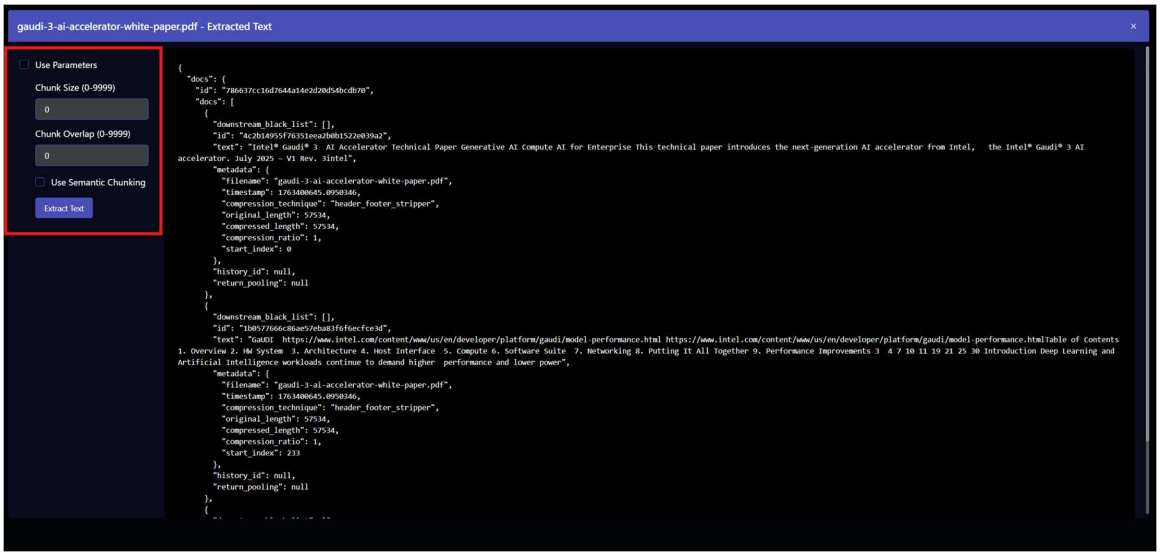
1. Click **Extract Text** next to the file.



2. Review the extracted content and how it has been chunked.



3. Use **Use Parameters** to experiment with chunk size, overlaps, and semantic chunking.



Changes made here affect only the Debug Mode and do not impact production.

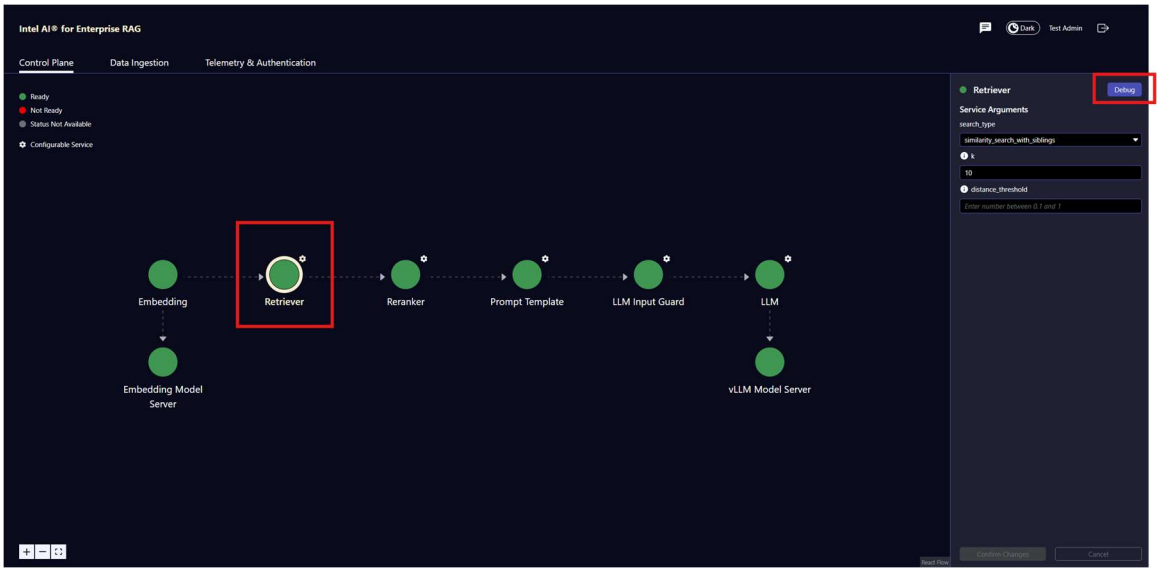
Previewing extracted content helps ensure documents and images are processed correctly and are ready to generate accurate answers.

4.5.3 Monitoring Pipeline Components

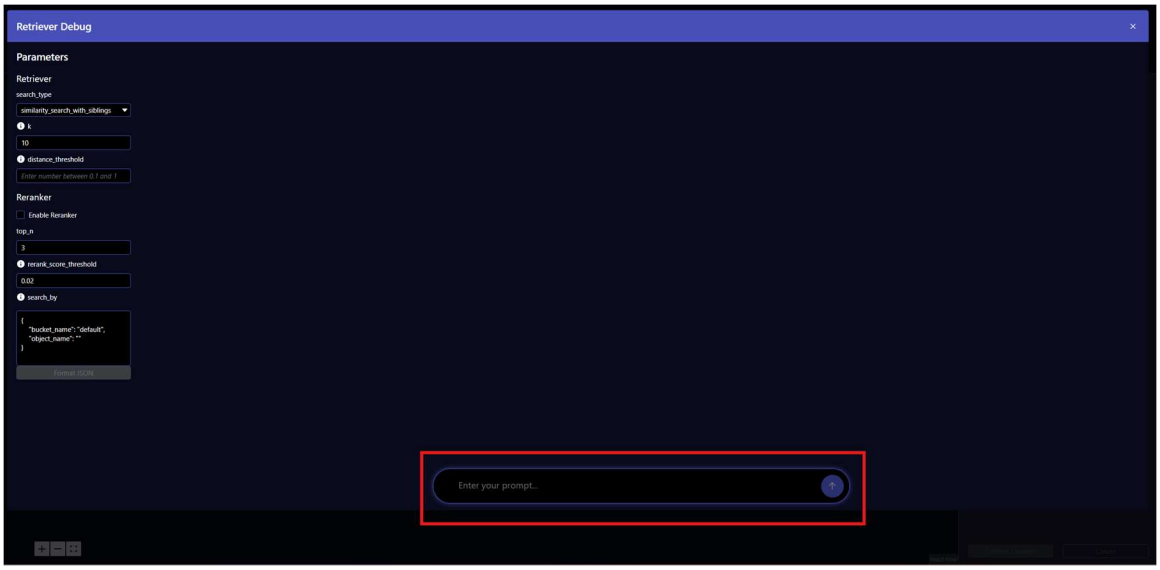
The Control Plane tab shows all components of the Chat Q&A pipeline and their status. In Debug Mode, you can inspect how queries move through the pipeline and troubleshoot issues with retrieval or ranking.

To check retrieval:

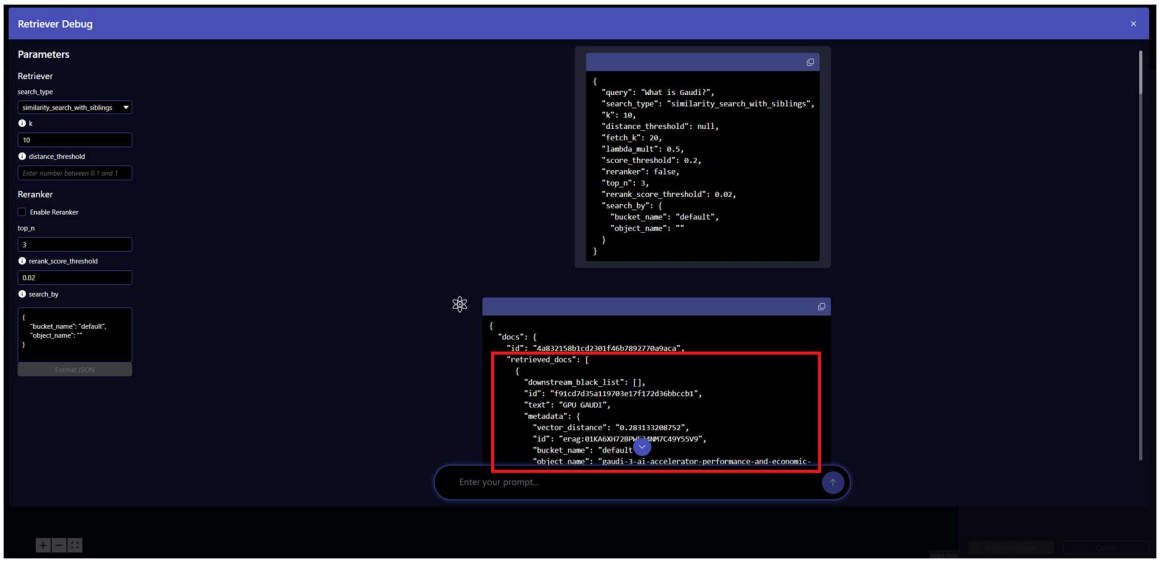
1. Open **Retriever** and click the **Debug** button.



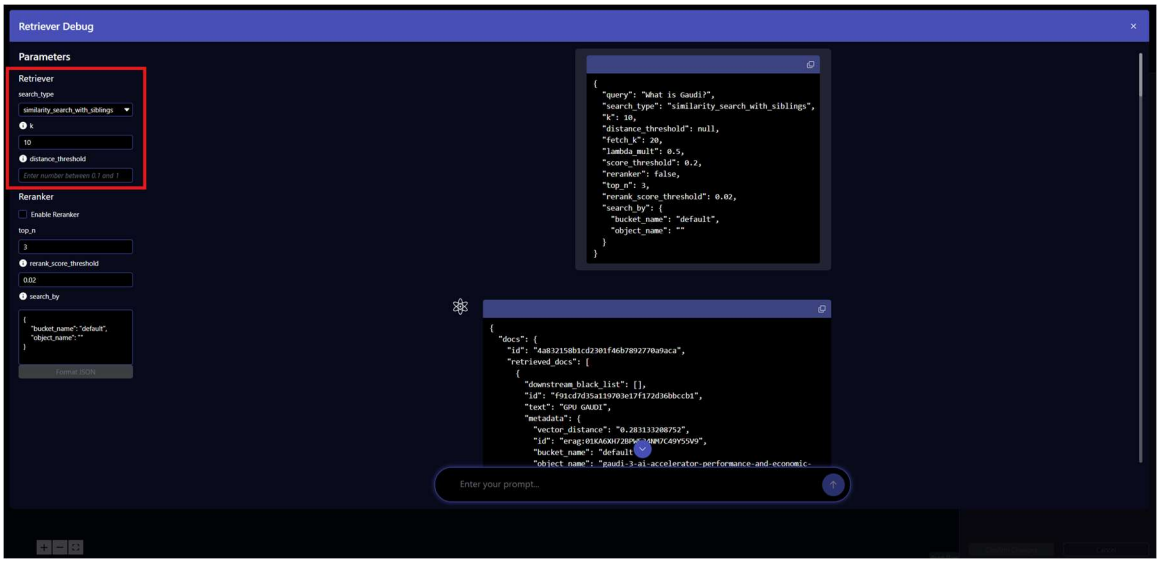
2. Enter a query.



3. Review which documents were selected under **retrieved_docs.**

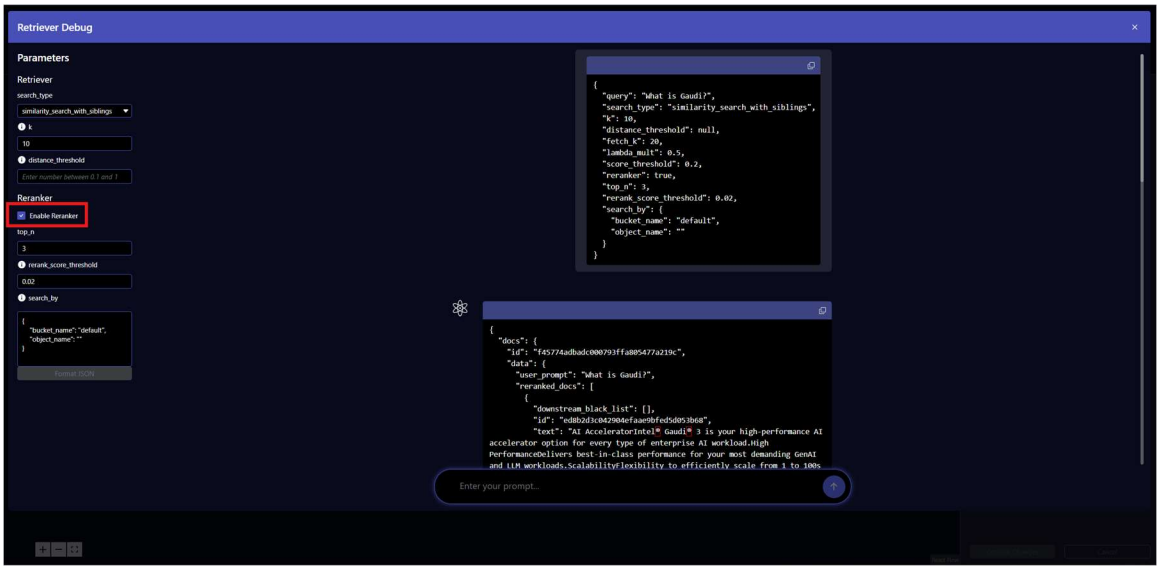


4. Adjust retrieval parameters to test different strategies and see how results change.



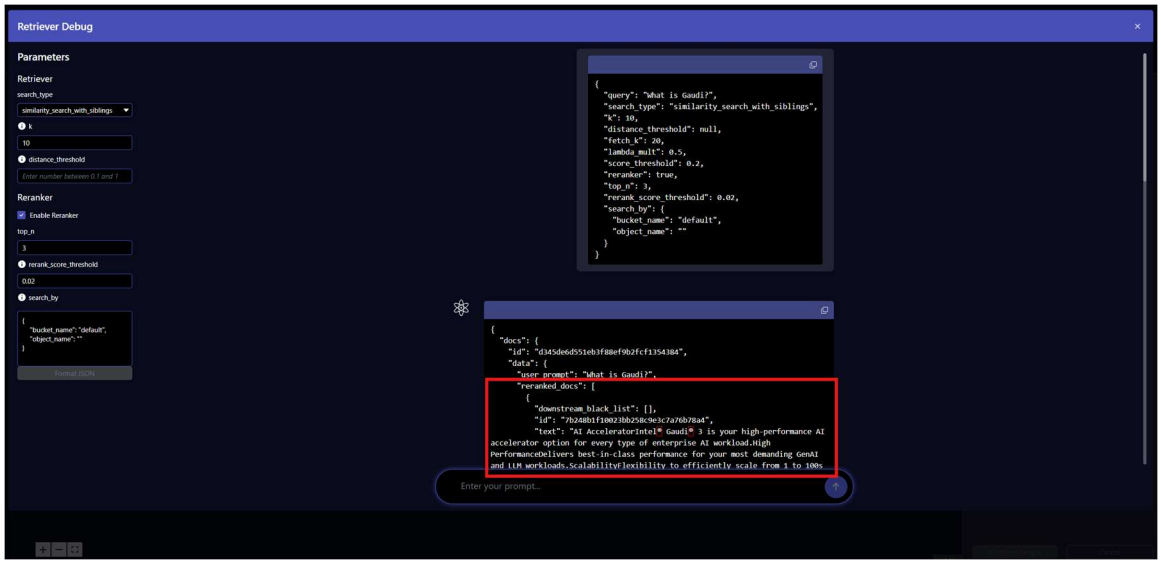
To include reranking in your inspection:

1. Mark **Enable Reranker.**



2. The response will show documents under **reranked_docs** along with their **reranker_score** (ranging from 0 to 1).

   The reranker identifies the top **n** documents with the highest score, prioritizing content most likely to contribute to an accurate answer.



3. Adjust **top_n** and **rerank_score_threshold** to improve relevance if needed.

This environment is a safe sandbox for experimentation without affecting production.

4.5.4 Role Based Access Control (RBAC) Testing

Chat QnA responses may vary depending on the user profile. Role-Based Access Control (RBAC) ensures that users see only the documents they are permitted to access.

Verify that users can only retrieve authorized documents by simulating a user's interaction with the RAG pipeline. This can be done by restricting the search to specific buckets.

To test **RBAC:**

1. Restrict retrieval to a specific bucket or list of buckets using the **search_by** parameter (for example, change from *default* to *hr-department*) to observe the pipeline behaving as if accessed by a user with limited permissions.

2. Simulate a user query.
3. Confirm that only documents the user is authorized to access are returned.

4.5.5 Common Issues and Recommended Actions

When using the RAG pipeline, there are a few important things to watch out for. Here's what to look for, possible issues, and what you can do:

1. **Content Gaps**
   - What it is: Sometimes the retriever or reranker cannot find relevant information in the vector database.
   - Possible issue: Some important information may be missing, which can lead to incomplete or incorrect answers.
   - What you can do:
     - Review the vector database and add missing content.
     - Adjust retriever settings (like **k** or **search_type**) to improve coverage.

2. **Conflicting Information**
   - What it is: When retrieved documents contain contradictory content.
   - Possible issue: This may cause inconsistent or misleading answers.
   - What you can do:
     - Check your source documents and resolve conflicts.
     - Remove outdated documents.
     - Ensure user access is correctly set with RBAC.

3. **Chunk Quality Issues**
   - What it is: How text is split into chunks for the prompt.

- Possible issues:
    - Chunks too small: important info may be split and missed.
    - Chunks too large: retrieval may be less precise, and the LLM may struggle to focus on key content.
- What you can do:
    - Adjust chunk size and overlap.
    - Test chunking on sample documents.
    - Refine the chunking approach as needed.

4. **Reranker Evaluation**
    - What it is: How the reranker scores retrieved documents and decides which are most relevant.
    - Possible issue: Poor scoring may prioritize irrelevant documents or miss high-value content.
    - What you can do:
        - Tune reranker settings (like **top_n** or **score_threshold**).
        - Monitor outputs to ensure relevance.
        - Consider using a stronger reranker model if needed.

5. Support & Resources

5.1 Online Resources

- Project Repository - Intel® AI for Enterprise RAG on GitHub:
  https://github.com/opea-project/Enterprise-RAG
  Access the latest source code, release notes, and technical documentation.

- SW Catalog - https://swcatalog.intel.com/
  Tools and frameworks optimized for RAG implementations.

5.2 Reporting Bugs or Suggesting Features

- Submit a detailed report through the project's GitHub Issues page:
  https://github.com/opea-project/Enterprise-RAG/issues
  Include steps to reproduce the problem and any relevant logs or screenshots.

6. Glossary

6.1 General Terms

Chunk
A small segment of text created when documents are split into manageable pieces for embedding and retrieval.

Embedding
The process of converting text into numerical vectors ("embeddings") for semantic search.

Knowledge Base
The collection of enterprise documents or data that the RAG system can search to provide evidence for its responses.

Large Language Model (LLM)
A type of AI model trained in massive amounts of text to understand and generate natural language response.

Retriever
The service that searches the vector database for text chunks most relevant to a user's question.

Retrieval-Augmented Generation (RAG)
An approach that retrieves relevant documents from a knowledge base and uses them as context for the AI model to produce grounded answers.

Semantic Search
A search method that looks for text with the same meaning, not just the same keywords. Instead of matching exact words, it compares the *concepts* in a query to numerical representations ("embeddings") of documents, so it can find relevant passages even when different wording is used.

Vector Database / Vector Store
A specialized database that stores embeddings so the system can quickly find semantically similar documents.

6.2 End-User Terms

Citation
The reference or link to a specific document that supports the AI's answer.

Conversation/Chat History
A record of previous questions and answers within a session.

End User
A person who interacts with the system to submit questions and view answers.

Natural Language Query
A question phrased in everyday language instead of using special commands.

6.3 Admin Terms

Administrator (Admin)
A user with permissions to configure, monitor, and maintain the Intel® AI for Enterprise RAG system.

Authentication/Authorization
The process of verifying a user's identity and assigning the correct permissions.

Bucket (S3 Bucket)
A container in Amazon S3 or compatible storage where files and data are stored for ingestion.

Connector
A built-in integration that links the RAG system to an external data source (e.g., a document repository or database).

Control Plane
The administrative dashboard which displays all deployed services as an interactive graph and allows viewing of system health. A visual interface for tracking system health, performance, and logs.

Data Ingestion
The process and interface for synchronizing files and web links from external storage into the knowledge base.

Data Pipeline
The series of steps used to ingest, process, and store data before it can be queried by the RAG system.

Deployment
The act of installing and running the application in a chosen environment (cloud, on-premises, etc.).

Embedding Model Server
The service that generates embeddings for storage and retrieval.

Grafana
A monitoring and visualization tool used to display performance metrics and system health.

Keycloak
An open-source identity and access management service used for authentication and user management.

Latency
The time it takes from sending a query to receiving a response.

LLM Input Guard
The protective layer that scans each user query for risks (prompt injection, toxic language, hidden characters, or secrets) before it reaches the model.

Prompt Template
A set of reusable instructions that frames both the system's behavior and how user questions are presented to the model.

Reranker
Evaluates and reorders the Retriever's results so the most relevant chunks are sent to the language model.

Scaling
Increasing or decreasing computing resources to handle demand.

S3 (Amazon Simple Storage Service)
A cloud object storage system commonly used for holding ingested documents and other large files.

vLLM Model Server
The high-performance inference engine that runs the large language model and streams responses.