

IBM TechXchange

Chat with Documents in watsonx.ai

Hands-on lab guide

Catherine Cao

catherine.cao@ibm.com

Senior WW Data and AI Technical Specialist

Caroline Gorham

cgorham@ibm.com

Senior watsonx Technical SME

Elena Lowery

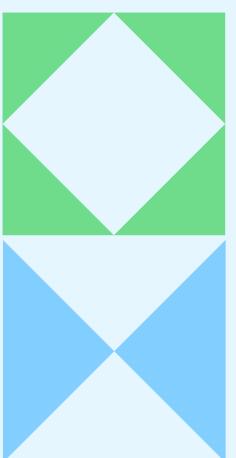
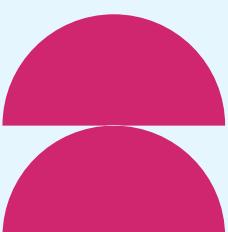
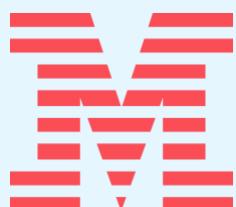
elowery@us.ibm.com

Principal WW Data and AI Technical Specialist

Joe Kozhaya

kozhaya@us.ibm.com

Data and AI CSM Architect





Contents

Overview.....	3
Required software, access, and files	3
RAG Overview.....	4
Implement RAG in watsonx.ai.....	11
Part 1: RAG with an in-memory vector store	11
Creating Vector Index	11
RAG Solution Development	18
RAG solution Deployment.....	24
RAG streamlit Application	31
Part 2: RAG with a static vector store	37
Part 3 (Optional)	43
Review RAG scripts and applications	43
RAG on documents with LangChain and ChromaDB	43
Review a Streamlit application with RAG using a document.....	50
Conclusion	53



Overview

In this lab, you will learn how to implement the Retrieval-Augmented Generation (RAG) pattern for working with LLMs using the *Chat with Documents* feature of watsonx.ai. In the first part of the lab, you will use an in-memory vector database to store a document, and in the second part, you will create a document collection in a static vector database.

Required software, access, and files

For this hands-on lab, you need the following:

- Access to *watsonx.ai* in IBM Cloud.
- The second part of the lab requires access to a vector database in *watsonx.data*. Your instructor will provide connection details.
- A Python IDE with Python 3.11 environment (*Visual Studio Code* or *PyCharm*).
- Access to sample prompts and code samples from the Lab Guide (linked on your lab landing page).

All of these pre-requisites are provided to you by the instructors and prepared in the environments you will be given access to. Specifically, you will have access to an IBM Cloud account which provides you with the required access to *watsonx.ai*. Additionally, you will have access to a Windows virtual machine (VM) which you can access via the browser and that VM will include Visual Studio Code (VSCode) and PyCharm IDEs as well as the Python 3.11 environments with required libraries for the lab.

The Windows Virtual Machine provided to you should already have the required lab files downloaded under **~/Downloads/2231_Lab_Files** but if for some reason it is not, or if you would like to work directly on the local machine (not the Windows VM), you can download the required files from the Lab Guide link as described below.

Important note: Some screenshots in the lab may be slightly different from the product. If you have questions, please ask your workshop instructor.

Accessing Environments

To execute all the steps in this lab, you will need access to two environments, (i) watsonx.ai SaaS environment and (ii) Windows VM for developing and executing Python code in Visual Studio Code IDE. After you launch your Chrome browser (it is setup to navigate to the Lab URL) and enter the lab access code (provided by the instructor), you will be re-directed to the following landing page for your lab.

Click the **Lab Guide** button (annotated with red arrow) to access and download the lab instructions and supporting lab files.

The screenshot shows the IBM Technology Zone interface. On the left, there's a sidebar with links like 'My dashboard', 'My created collections', 'My bookmarked searches', 'My favorites', 'My reservations', 'My workshops', and 'My events'. The main content area is titled 'Student 1 - Chat with documents' and 'Lab 2231'. It contains a brief description of Retrieval Augmented Generation, followed by two timestamped buttons: 'October 23, 2024 11:45 AM' and 'October 23, 2024 1:15 PM'. Below this is a 'Contact Info' section with 'Lab instructor(s)' (Ryan Kather, ryan.kather@ibm.com, Lin Ju, linju@ca.ibm.com) and 'Lab developer(s)' (Elena Lowery, elowery@us.ibm.com). A large button at the bottom labeled 'Lab Guide' has a red arrow pointing to it from the right.

Scroll down on that page to review the two environments provided for you:

- 1- watsonx.ai SaaS environment: This is where you develop and deploy prompts using large language models (LLMs). Capture the **IBM Cloud User Name** (annotated with red oval), the **IBM Cloud Password** (annotated with red arrow) and the **URLs** (annotated with red rectangle). These credentials are required for you to access watsonx.ai SaaS environment.

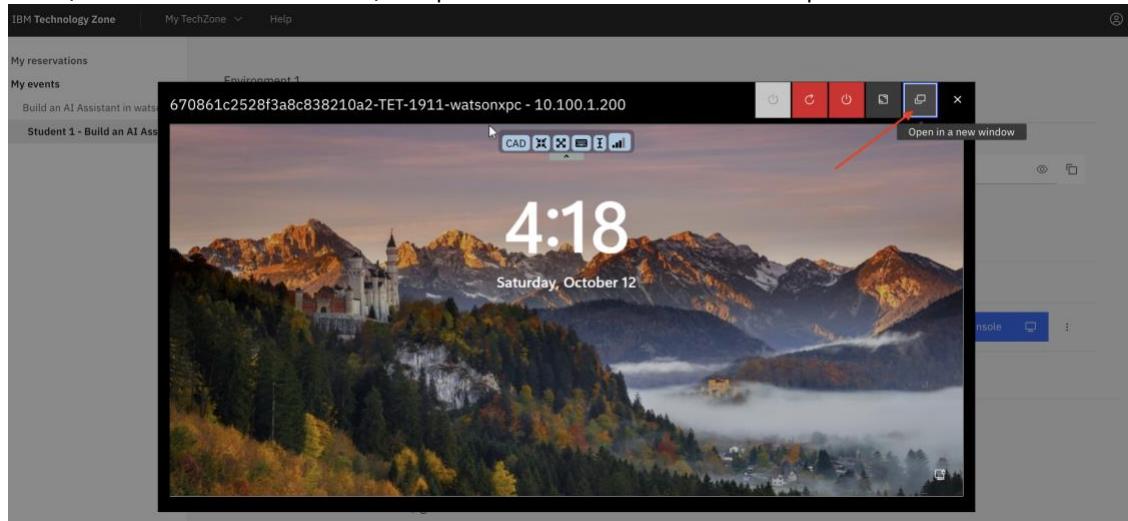
The screenshot shows the environment details for 'Environment 1'. It includes sections for 'Environment Identifier' (txv-lab-khl84u_670dcfaed), 'Environment Details' (WatsonX Instance: txv-lab-khl84u_670dcfaed, WatsonX Username: student_khl84u, WatsonX Password: nhqj4dmcvnfvgez), 'Instructions' (warning about using the default IBM Cloud login), 'Service Identifier' (txv-lab-khl84u_670dcfaed), 'Service API Key' (redacted), 'Watson AI' (a long URL), 'URLs' (https://cloud.ibm.com/logout), and 'Service Accounts' (student_khl84u and nhqj4dmcvnfvgez). Red annotations include a red oval around 'student_khl84u', a red arrow pointing to 'nhqj4dmcvnfvgez', and a red rectangle around the 'URLs' section.



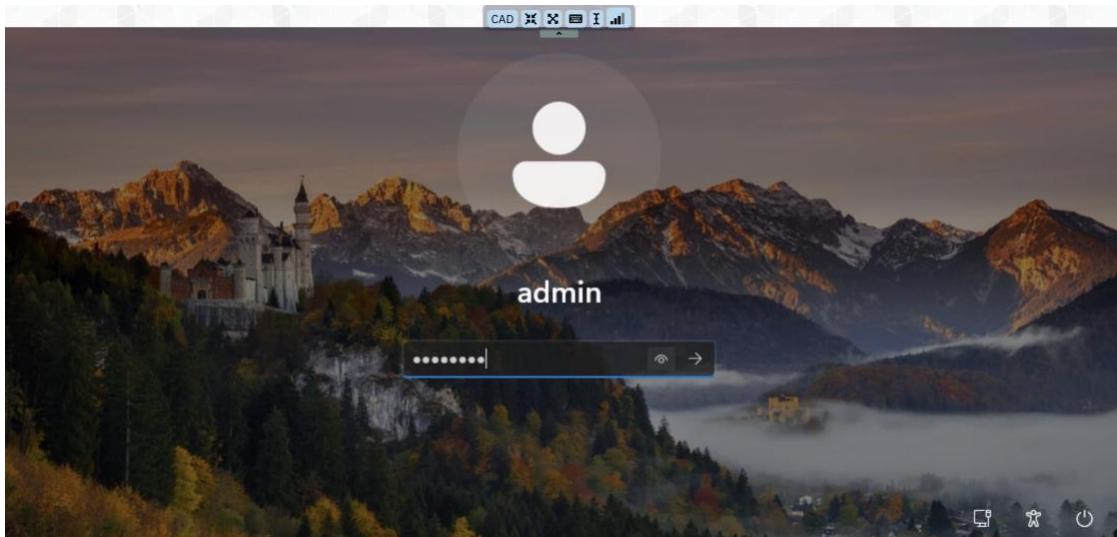
- 2- Windows VM: This is where you develop Python web application that offers a User Interface (UI) and orchestration logic to interact with your LLM prompt templates. You will need this Windows VM in later sections of the lab where you develop and execute the Python application. However, it is a good idea to launch it now to verify you can access. Click the **Console** button (annotated with red arrow).

The screenshot shows the IBM Technology Zone interface. At the top, there's a header with a dropdown arrow, the ID '670dcfafd325d5b5d1448999', and a refresh icon. To the right, it says 'Ready'. Below the header, the title 'Environment 2' and subtitle 'Student environment 1 - watsonxai_client_tools' are displayed. A table follows, with columns: Published Services, Router WAN IP, Router User Name, and Router Password. Under 'Published Services', there's a link to 'watsonxai_client_tools'. In the 'Router WAN IP' column, '10.170.126.25' is listed. In the 'Router User Name' column, 'admin' is listed. In the 'Router Password' column, '*****' is listed. Below the table, another table lists VM details: Name ('670dcfafd325d5b5d1448999-TET-1911-watsonxpc'), OS ('Microsoft Windows 10 (64-bit)'), IP ('10.100.1.200'), Status ('Running'), and a 'Console' button (annotated with a red arrow). There's also a 'Copy' icon next to the IP address.

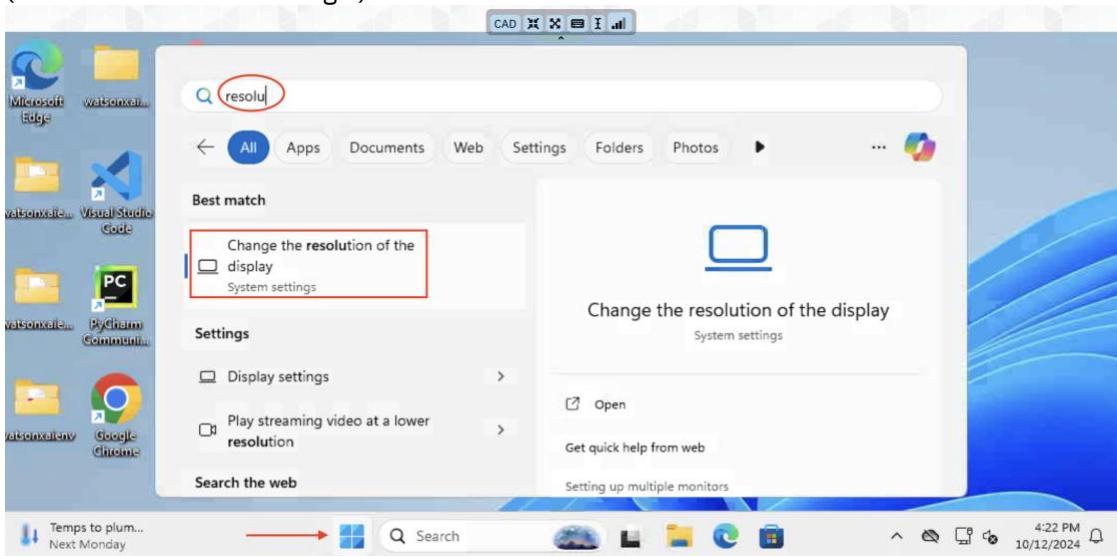
The VM will launch in a pop-up window as shown in figure below. Click the **Open in a new window** icon (annotated with red arrow) to open this Virtual Machine in a separate browser tab.



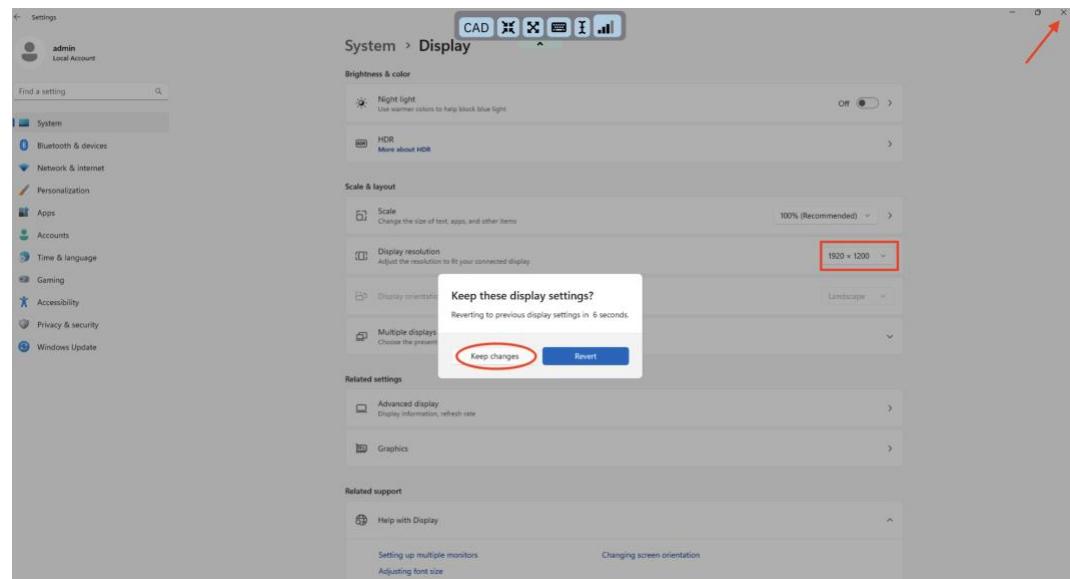
When you first launch the VM (or when not used for a period of time), it will be locked. To unlock the VM, click anywhere and you will be prompted to enter the password for the admin user. Type in the password **IBMDemOs** and hit Enter to unlock the VM.



(Optional) Once logged in, you can change the resolution of the Windows VM if you would like. To do so, click the **Windows icon** (annotated with red arrow) and start typing **resolution** in the search bar (annotated with red oval) and select the **Change the resolution of the display** setting (annotated with red rectangle).



Select your preferred display resolution (in figure below, we select **1920x1200** annotated with red rectangle). On the confirmation pop up window, select **Keep changes** (annotated with red oval) to preserve the display resolution setting. Lastly, click the **x** icon (annotated with red arrow) to close this setting.



RAG Overview

RAG (Retrieval-Augmented Generation) is one of the most common use cases in generative AI because it allows us to work with data that is “external to the model,” i.e., data that was not used during model training. Many use cases require working with proprietary company data, which is one of the reasons why RAG is frequently used in generative AI applications. Additionally, RAG allows us to add guardrails to generated output and reduce hallucinations.

When describing the RAG pattern, AI engineers often use the term “grounding”, which means that the output of the LLM is derived from the provided document collection, rather than from the LLM's internal knowledge.

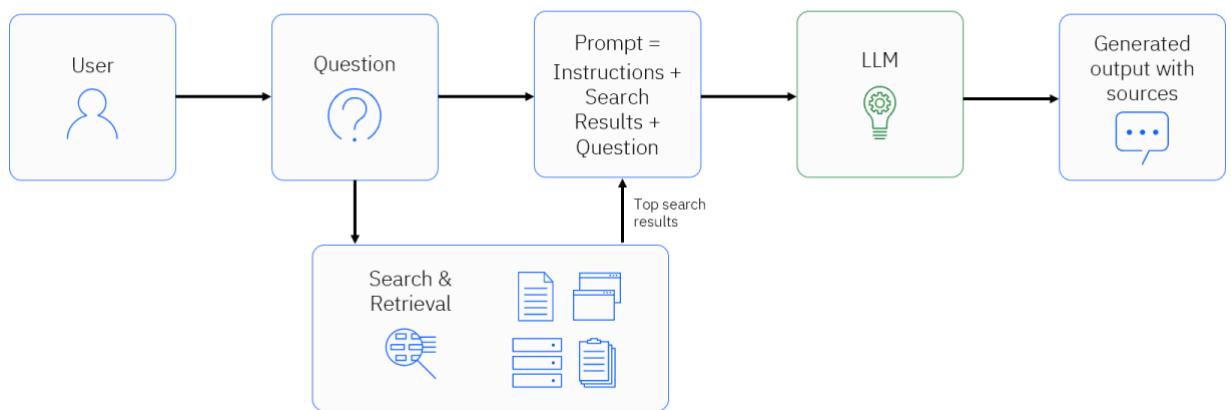
RAG can be applied to several generative AI use cases, including:

- Question answering
- Summarization
- Content generation.

Note: A “human interaction” analogy of RAG is providing a document to a person and asking them to answer a question based on the information in the document.

RAG is an application pattern that can be implemented with multiple technologies. It has two steps:

- **Retrieval:** During the **retrieval** step we search through a knowledge base (documents, Websites, databases, etc.) to find information relevant to the instructions sent to the model. Retrieval can be based on keyword search or more advanced methods, like semantic similarity.
- **Generation:** The **generation** step is similar to generation in non-RAG use cases. The main difference is that information retrieved in the first step is provided as “context” (included in the prompt).



The *knowledge base* for the retrieval step is typically implemented as a collection of documents in a *vector database*.

Some of the popular vector databases are:

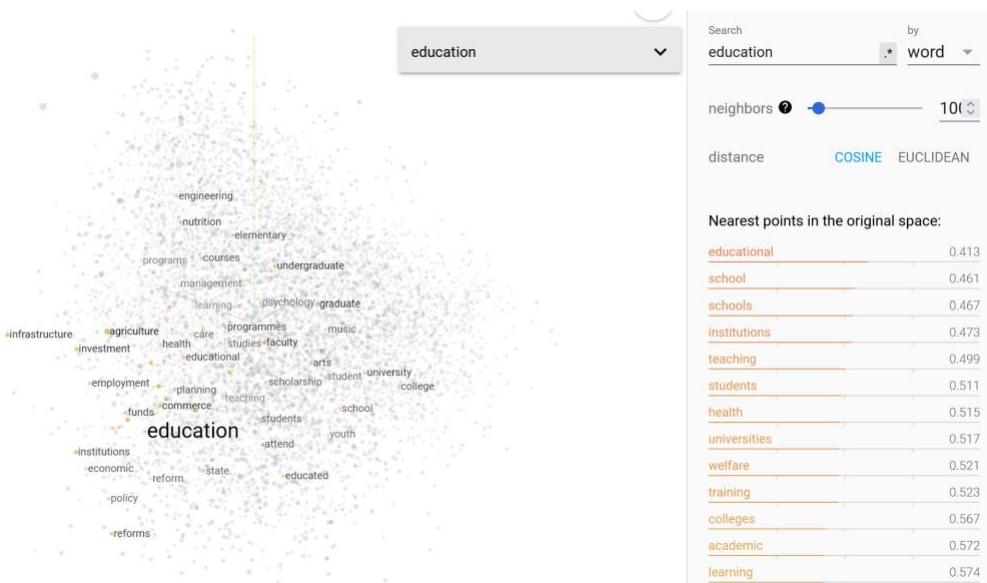
- *Chroma*
- *Milvus*
- *Elasticsearch*
- *SingleStore*
- *FAISS*
- *Pinecone*
- *Redis*
- *Postgres*

Some vector databases, like *Chroma*, provide an in-memory option, which makes it a good choice for experimenting and prototyping RAG use cases. In-memory databases do not need to be installed or configured, but the downside is that we need to load data each time we run an application.

Vector databases store unstructured data in numeric format. For AI use cases, we use a specific term to describe the converted unstructured data – *embeddings* or *vectors*. Embeddings are created by using embedding models, such as a popular open-source model *all-MiniLM-L6-v2*.

Here's an example of semantic search for the word “education”. Notice the “nearest point” words in the table. The measurement is in numbers because the search was performed using vector (numeric) data. You can experiment with more words on this [site](#).

When we implement the RAG pattern, we will perform semantic search based on the entire question asked by a user.



Based on what we have discussed so far, we can make the following conclusions:

- We have a choice of *vector databases* for implementation of RAG use cases.
- We have a choice of several *embedding models* for converting unstructured data to vectors.
- Retrieval of relevant information is the first step in the RAG pattern, and the quality may be affected by the choice of a vector database and an embedding model.



In addition to the mentioned factors, loading data into the vector database requires “chunking”, which is the process of dividing text into sections that are loaded into the vector database. There is no single guideline for defining chunks because the best size depends on the content of the document collection.

For generative AI use cases, we typically start with fixed-size chunks that are semantically meaningful. For example, if each paragraph in your document has approximately 200 words, convert this number to characters and use it as the initial chunk size. In addition to chunk size, we also need to specify the number of overlapping characters (the same content in more than one chunk). Keep in mind that embedding models have token limits, so the chunk size should not exceed the maximum supported tokens. If you plan to use watsonx.ai embedding models, you can find information about token limits [here](#).

Embedding model providers should publish information on the recommended chunk size and overlap percentage for specific use cases. However, even if this information is available, developers will still need to experiment to find the optimal chunk size. If you can't find the information on the recommended overlap, you can start with 10 percent: for example, set the overlap to 100 if your chunk size is 1,000.

To summarize, we need to complete these steps when implementing the RAG use case:

1. Select a vector database. When working with watsonx.ai, we should confirm that the selected vector database is supported. [Check IBM documentation for the latest updates](#).
2. Select the embedding model.
3. Experiment with chunk size.

As we discussed earlier, the second component of the RAG pattern, sending a prompt to an LLM, is similar to non-RAG patterns. Developers should pay special attention to token limits because the content retrieved from a vector database will be added to the prompt. If the token limit is exceeded on input, then the LLM will not be able to generate output. The check for token limit is not automatic, it needs to be implemented in your applications.

Example: watsonx.ai API returns errors if token size is exceeded.

```
raise ApiRequestFailure('Failure during {}.'.format(operation_name), response)
ibm_watson_machine_learning.wml_client_error.ApiRequestFailure: Failure during generate. (POST https://us-south.ml.cloud.ibm.com/ml/v1-beta/generation/text?version=2022-08-01)
Status code: 400, body: {"errors": [{"code": "invalid_input_argument", "message": "Invalid input argument for model meta-llama/llama-2-70b-chat: input tokens (6233) plus prefix length (0) must be \u003c 4096"}]}, "trace": "6f12c353d374273dc4a68218"
```

Implement RAG in watsonx.ai

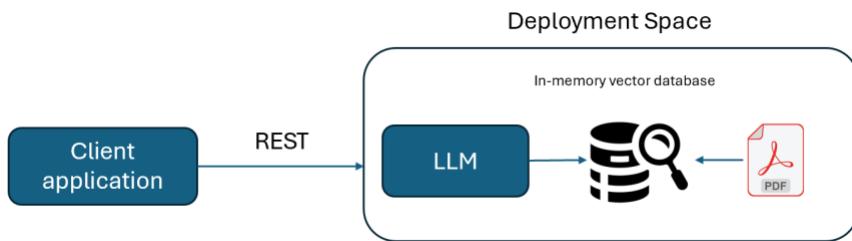
While it's possible to implement the RAG pattern with open-source APIs, it requires advanced knowledge and considerable development effort. Watsonx.ai supports implementing this pattern without writing code, using either an in-memory or persistent vector store. In this lab, we will review both approaches.

Part 1: RAG with an in-memory vector store

Using an in-memory vector store is a useful approach for

- Prototyping
- Working with small documents
- Working with documents that are useful for a limited timeframe.

In watsonx.ai implementation of the RAG pattern, we create a *document collection* which is loaded into an in-memory vector database during the development phase and inference.



Let's walk through the process of creating and deploying this pattern.

Creating Vector Index

1. Log in to your IBM Cloud account.

- On your lab landing page (after you enter the access code provided by your instructor), capture the credentials you need to access an IBM Cloud account with required services for this lab. Specifically, capture the **IBM Cloud User Name** (annotated with red oval) and the **IBM Cloud Password** (annotated with red arrow). Next, click the **URLs** (annotated with red rectangle).

The screenshot shows the IBM Cloud Lab interface with the following details:

- Environment Identifier:** txv-lab-khl84u_670dcfaed
- Environment Details:** WatsonX Instance: txv-lab-khl84u_670dcfaed, WatsonX Username: student_khl84u, WatsonX Password: nhqj4dmcvnfvgez
- Instructions:** **IMPORTANT**: DO NOT USE the default IBM Cloud login. Use the IBM Cloud Login URL from this reservation. You must be logged out of IBM Cloud before authenticating, or use an Incognito/Private browser session.
- Service Identifier:** txv-lab-khl84u_670dcfaed
- Service API Key:** (red oval)
- WatsonX AI:** <https://cloud.ibm.com/services/data-science-experience/cm%3Av%3Abluebeam%3Apublic%3/science-experience%3Aus>
- URLs:** <https://cloud.ibm.com/authorize/itzwatsonx031> (red rectangle)
- Logout URL:** https://cloud.ibm.com/logout
- Links at the bottom:**
 - <https://cloud.ibm.com/authorize/itzwatsonx031> (red oval)
 - [student_khl84u](#)
 - [nhqj4dmcvnfvgez](#) (red arrow)

- Enter the Username and Password you copied in the previous step and click **Sign in** (annotated with red arrow) to log into your IBM Cloud account.



Username
 student_8idb2l

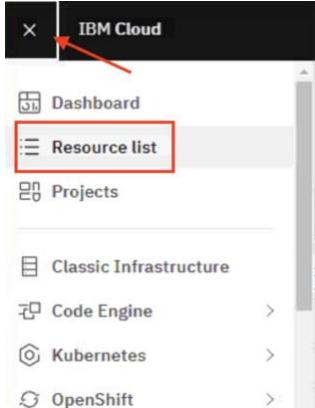
Password

→ **Sign in**

Powered by IBM Technology Zone

At this time, you are logged into your IBM Cloud account. Next, you need to navigate to your watsonx.ai service.

2. To launch your watsonx.ai service, click on the top left **navigation menu** (hamburger menu) in your IBM Cloud UI and select **Resource list** in the drop down (highlighted with red rectangle).



- On the Resource list page, under **Product** column, search for *watson studio* (annotated with red rectangle) to quickly find your watsonx.ai service. You should see a Watson Studio service provisioned in your environment (annotated with red arrow). Click on that service (annotated with red arrow).

Resource list

Name	Group	Location	Product	Status	Tags
ws-txv-lab-8idb2l_670861c45	txv-lab-8idb2l_670861c45	Dallas	Watson Studio	Active	-

- On the *Watson Studio* service page, make sure to select *Launch in IBM watsonx* by selecting the drop down and choosing **IBM watsonx** (annotated with red arrow).

Resource list / ws-txv-lab-8idb2l_670861c45

Watson Studio in Cloud Pak for Data and watsonx

Build and deploy machine learning models on either platform. Work with foundation models on watsonx as a Service.

IBM Watson Studio in Cloud Pak for Data and watsonx

IBM Watson Studio is part of IBM Cloud Pak for Data and watsonx, and serves as the AI capability of the data fabric architecture.

IBM Cloud Pak for Data, watsonx, Unifying platforms

IBM Cloud Base cloud infrastructure

Launch in

- IBM Cloud Pak for Data
- IBM watsonx**
- Help links

- If the page “Preparing IBM watsonx.ai ...” page doesn’t load within a minute, please refresh that browser page.

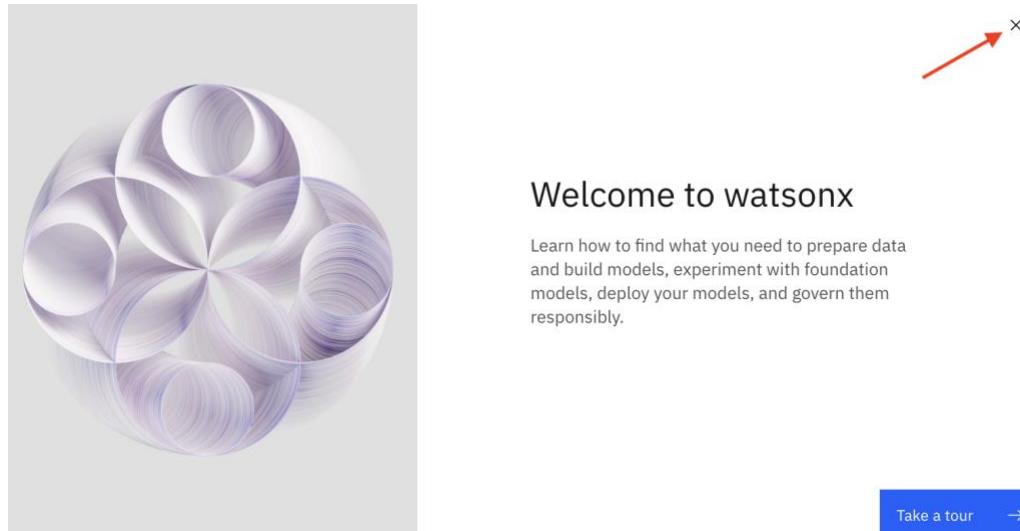
Preparing IBM watsonx.ai...

- Experiment with prompts for foundation models
- Build machine learning models your way
- Manage your AI lifecycle

Registering you with IBM watsonx.ai

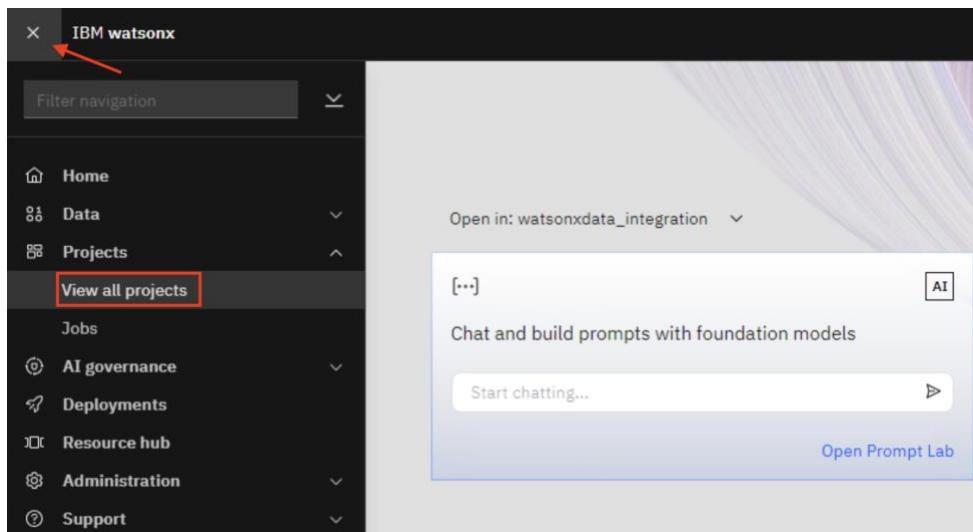
Provisioning your IBM watsonx.ai services.

- If you get the **Welcome to watsonx** pop up (you get that first time you access), close the pop-up by selecting the x on the top right corner of the pop-up window (annotated with red arrow).



At this point, you are logged into watsonx.ai user interface.

- In watsonx.ai, click the top left navigation menu (hamburger menu) in the top left corner of your watsonx.ai user interface, select **Projects -> View All Projects**.



- Click on the **New Project** button. Provide a name for the project, a description (optional) and select the Cloud Object Storage instance (you should only have one instance in your drop-down list). Please include your initials in the name of the project since this watsonx.ai service is a multi-tenant shared service for multiple users; for example, watsonx-chat-with-docs-<YOUR INITIALS>. Click **Create** to create the project.
In watsonx.ai, a project is a collaborative environment where you can work with other members of your team to develop traditional machine learning and generative AI solutions. It allows you to work with different types of assets such as Jupyter notebooks, LLM prompts, and other assets needed for developing traditional and generative AI solutions.

Create a project

Start with a new, blank project or select from where to import an existing project.

Define details

Name: watsonx-chat-with-docs-jk

Description (optional): A sample project for illustrating chat with documents capability in [watsonx.ai](#)

Tags (optional): Add tags

Storage: cos-techzone
Project includes integration with [Cloud Object Storage](#) for storing project assets.

Advanced settings

Cancel Create

- Once the project is created, you will be at the project landing page which is the **Overview**. Switch to the **Services and Integrations** page by clicking the **Manage** tab (annotated with red rectangle) and selecting **Services & integrations** (annotated with red arrow). On the Services & Integrations page, click **Associate Service** (annotated with red oval).

Projects / watsonx-chat-with-docs-jk

Overview Assets Deployments Jobs **Manage**

Project

- General
- Access control
- Environments
- Resource usage
- Services & integrations**

Services & integrations

IBM services (1) Third-party integrations

Find services

Name Service type

Associate service

- On the Associate service pop-up window, select the displayed **Watson Machine Learning** service (annotated with red arrow) and click **Associate**.

*Note: if the service is already associated, click **Cancel** and proceed to the next step.*

The Watson Machine Learning service is required to enable you to access and work with the prompt lab and interact with large language models hosted on [watsonx.ai](#).

Associate service

Choose an existing or add a new service to associate with your project.

Resource Groups Locations

Name	Type	Plan	Location	Status	Group
watsonx Assistant-itz	watsonx Assistant	Enterprise	Dallas	Not associated	watsonx
Data Replication-itz	Data Replication	Lite	Global	Not associated	watsonx
Watson Discovery-itz	Watson Discovery	Enterprise	Dallas	Not associated	watsonx
Watson Machine Learning-itz	Watson Machine Learning	Essentials	Dallas	Not associated	watsonx
Speech to Text-itz	Speech to Text	Plus - NEW!	Dallas	Not associated	watsonx
Text to Speech-itz	Text to Speech	Standard	Dallas	Not associated	watsonx

New service +

Cancel Associate

7. Switch to the **Assets** tab (annotated with red rectangle), then click the **New asset** button (annotated with red arrow).

The screenshot shows the WatsonX interface with the 'Projects' dropdown set to 'watsonx-chat-with-docs-jk'. The 'Assets' tab is highlighted with a red box. In the top right, there is a 'New asset' button with a red arrow pointing to it. The main area shows '0 assets' and a 'All assets' section with a 'Start working' callout.

8. Under Prepare data, select **Ground gen AI with vectorized documents** tile (annotated with red rectangle).

The screenshot shows the 'Prepare data' section with several tiles. One tile, 'Ground gen AI with vectorized documents', is highlighted with a red box. Other tiles include 'Connect to a data source', 'Prepare and visualize data', and 'Define reusable sets of parameters'.

9. Click the **Browse** button (annotated with red oval) in order to select the document to upload to the vector index. Navigate to the **401K_Updates.pdf** document on your Virtual Machine under **2231_Lab_Files/Documents/** folder. If you downloaded the lab files to your Downloads folder, the file path would be under **~/Downloads/2231_Lab_Files/Documents/401K_Updates.pdf**

Keep the default vector store selection as **In memory** (annotated with red rectangle). Provide a **Name** and a **Description** (optional) for your vector index. Expand the **Advanced settings** drop down (annotated with red arrow) and review / keep the default selection for **Embeddings model** (all-MiniLM-L6-v2), **Text chunk size** (2000 characters), and **Text chunk overlap** (200 characters).

Click **Create** button to create the in-memory vector index using the provided document (401K_Updates.pdf) and the settings you reviewed for embeddings model, chunk size, and chunk overlap.

Note the size limits for the different file formats as well and check the [documentation](#) for further details.

Note: building a document collection is an iterative process, and if during testing we find out that semantic search does not return optimal results, we can rebuild the collection with a different embedding model and different chunk sizes.

Ground gen AI with vectorized documents

Add documents to vectorize and create a vector index in memory. Otherwise, select your vector database and specify index details.

10. It takes a minute or so to import the pdf file, parse the text, chunk it into 2000-character chunks, and store it as a vector using the in-memory vector index. When finished processing, your vector store will look similar to the following screenshot.

11. Next, you will test the semantic search functionality. Switch to the **Test** tab (annotated with red oval) and type the following questions one-by-one in the **Search vector index** field (annotated with red rectangle):

- *What is an ESG fund?*
- *What are the top 3 ESG funds?*
- *What is the most conservative fund?*
- *How do I withdraw money in retirement?*



After testing each question, add it as a *sample question* by clicking the **Add as sample question** button (annotated with red arrow).

We also recommend that you review search results to make sure that the question is answered. While it's not a necessary step (when we test with the LLM, it will let us know if there is no answer), it's still a good practice when you first create a document collection. You can also automate testing of semantic search by using an API.

Note that these results are strictly testing the retrieval part of the RAG part; they are searching the vector index to find the most semantically similar chunks of text to the input query. We will review the LLM generation component in later steps in this lab.

Projects / watsonx-chat-with-docs-jk / 401K_data_jk

Details Sample questions **Test**

Search vector index ⓘ

Add as sample question

Adjust search settings

Content	Document	Location	Score ⓘ
particular ESG fund may not reflect your values as an investor or even show favorable ESG qualities. Before investing in any investment options of the IBM 401(k) Plus Plan, please carefully consider its investment objectives, risks, and fees and read the applicable fund...	401K_Updates.pdf	Page: 10, Line: 31 - 37	0.6
A new U.S. bond market index fund (passively managed fund) To satisfy participants that want to continue to invest in a passively managed bond fund, the Vanguard Total Bond Market Index Fund has been added to your investment options. The fund seeks to track...	401K_Updates.pdf	Page: 10, Line: 1 - 33	0.53
Build a blueprint for your future—Whatever that future may hold How much should you be saving? IBMers' top questions of 2021 Would you like to consolidate your retirement assets? It's a digital world. Is your beneficiary information still on paper? Changes to yo...	401K_Updates.pdf	Page: 8, Line: 1 - 39	0.52

12. Switch to the **Sample questions** tab (annotated with red oval) and verify the questions you tested earlier are all saved as sample questions.

Projects / watsonx-chat-with-docs-jk / 401K_data_jk

Details **Sample questions** Test

Sample questions ⓘ

Search sample questions

Add sample question +

Question	⋮
<input type="checkbox"/> How do I withdraw money in retirement?	⋮
<input type="checkbox"/> What is the most conservative fund?	⋮
<input type="checkbox"/> What are the top 3 ESG funds?	⋮
<input type="checkbox"/> What is an ESG fund?	⋮

RAG Solution Development

13. Next, we will test the complete RAG solution results including the generation component with an LLM. To do so, navigate back to your project **Assets** tab (annotated with red rectangle) and click **New asset** button (annotated with red arrow).

Note the 401K_Updates.pdf document and the 401K_data_jk vector index are assets in your project.

Projects / watsonx-chat-with-docs-jk

Overview Assets Deployments Jobs Manage

Import assets Import assets New asset +

2 assets All assets

All assets

Name	Last modified
401K_data_jk Vector index	4 minutes ago Modified by you
401K_Updates.pdf PDF	19 minutes ago Modified by you

14. Under **Work with models** category (annotated with red oval), find and select the **Chat and build prompts with foundation models** tile (annotated with red rectangle).

What do you want to do?
Select a task based on your goal. You'll use a tool to create an asset for that goal.

All

Prepare data

Work with models

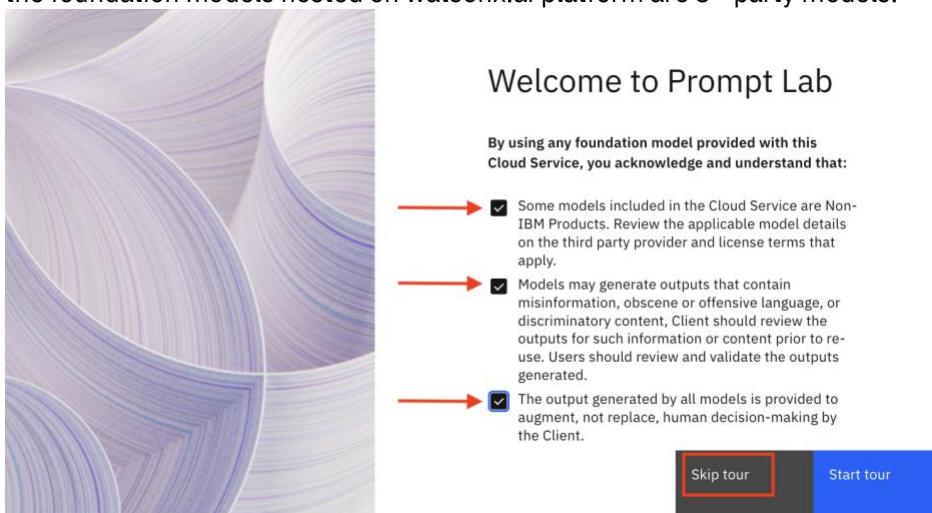
Automate model lifecycles

Search for a task or tool.

Work with models

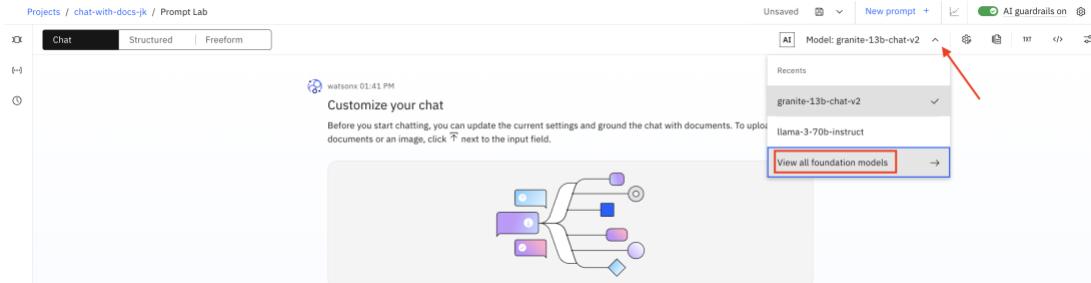
Build machine learning models automatically with AutoAI	Chat and build prompts with foundation models with Prompt Lab	Tune a foundation model with labeled data with Tuning Studio	Build models as a visual flow with SPSS Modeler
Work with data and models in Python or R notebooks	Solve optimization problems	Train models on distributed data	Write R scripts

15. If you get the **Welcome to Prompt Lab** pop-up, review the details on this pop-up, select the check boxes (annotated with red arrows) and click **Skip tour** (annotated with red rectangle). This text is effectively a warning that the output from foundation models may not be always factual and some of the foundation models hosted on watsonx.ai platform are 3rd party models.





16. Once the Prompt Lab UI launches, choose the llama-3-70b-instruct model as the large language model (LLM) to work with in this lab (you can also explore leveraging other LLMs if you would like but the instructions were written leveraging llama-3-70b-instruct LLM). Click the model drop down (annotated with red arrow) and select **View all foundation models** (annotated with red rectangle).



Filter the models by typing **llama** in the search box (annotated with red oval) and **select llama-3-70b-instruct model** (annotated with red rectangle).

Select a foundation model

To choose a model, review characteristics such as tasks that models perform. Compare model benchmarks with scores in the range 0–100. Higher scores are better.

All models Model benchmarks

Search bar: **llama** (highlighted with a red oval)

Model	Provider	Type	Description
llama-2-13b-chat	Meta	Provided model	llama-2-13b-chat is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-1-70b-instruct	Meta	Provided model	llama-3-1-70b-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-1-8b-instruct	Meta	Provided model	llama-3-1-8b-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-2-11b-vision-instruct	Meta	Provided model	llama-3-2-11b-vision-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-2-1b-instruct	Meta	Provided model	llama-3-2-1b-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-2-3b-instruct	Meta	Provided model	llama-3-2-3b-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-2-90b-vision-instruct	Meta	Provided model	llama-3-2-90b-vision-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-405b-instruct	Meta	Provided model	llama-3-405b-instruct is Meta's largest open-sourced foundation model to date, with 405 billion parameters, optimized for dialogue use...
llama-3-70b-instruct	Meta	Provided model	llama-3-70b-instruct is an auto-regressive language model that uses an optimized transformer architecture.
llama-3-8b-instruct	Meta	Provided model	llama-3-8b-instruct is an auto-regressive language model that uses an optimized transformer architecture.

Review the model details and click **Select model** button (annotated with red arrow).

llama-3-70b-instruct

Provider: Meta | Type: Provided model

Question answering Summarization Retrieval-Augmented Generation Classification Generation Code generation and conversion Extraction

Note: This model is a Non-IBM Product governed by a third-party license that may impose use restrictions and other obligations. By using this model you agree to these terms. [Read terms](#)

Model Datasheet

Model Details

Built with Meta Llama 3. Meta Llama 3 is licensed under the META LLAMA 3 Community License. Copyright © Meta Platforms, Inc. All Rights Reserved.

Meta developed and released the Meta Llama 3 family of large language models (LLMs), a collection of pretrained and instruction tuned generative text models in 8 and 70B sizes. The Llama 3 instruction tuned models are optimized for dialogue use cases and outperform many of the available open source chat models on common industry benchmarks. Further, in developing these models, we took great care to optimize helpfulness and safety.

Model developers Meta

Variations Llama 3 comes in two sizes — 8B and 70B parameters — in pre-trained and instruction tuned variants.

Input Models input text only.

Output Models generate text and code only.

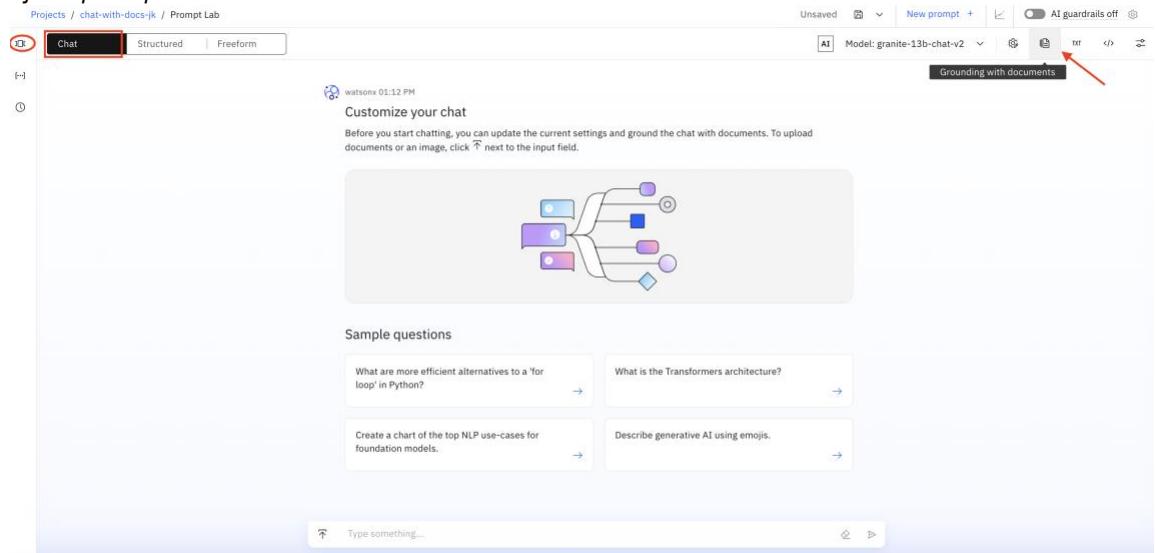
Model Architecture Llama 3 is an auto-regressive language model that uses an optimized transformer architecture. The tuned versions use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align with human preferences for helpfulness and safety.

Training Data	Params	Context length	GQA	Token count	Knowledge cutoff
llama-3-70b-instruct	~70B	~100K	~100K	~70B	~100K

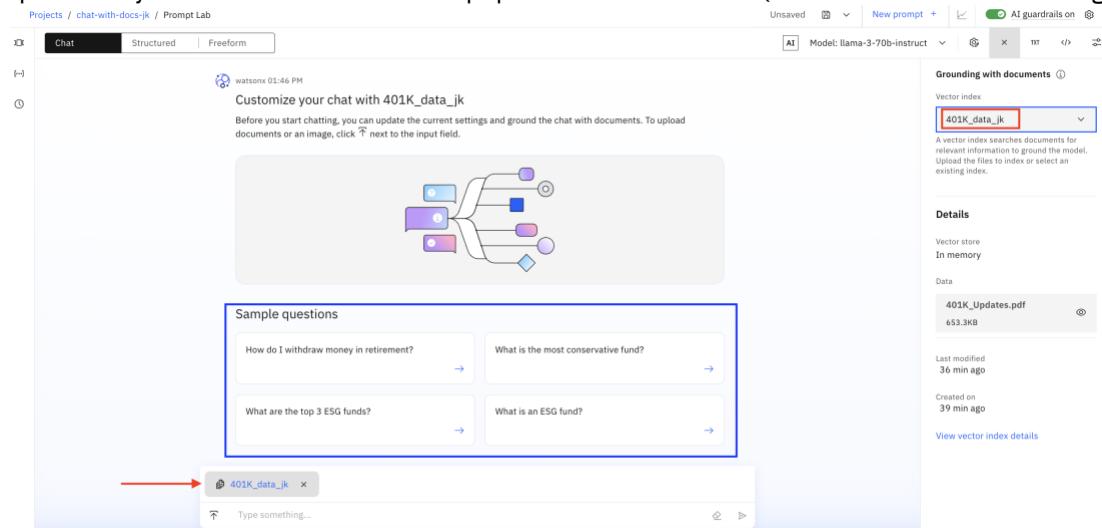
Back **Select model** (highlighted with a red arrow)

17. If the sample prompts window is open, click the **Sample Prompts** icon (annotated with red oval) to close it. Also, switch to the **Chat** mode of the prompt lab UI (annotated with red rectangle) and click the **Grounding with documents** icon (annotated with red arrow) top right of the page.

Note: The watsonx.ai Prompt Lab supports multiple modes including Chat, Structured, and Freeform which appeal to different use cases and scenarios. For this lab, we'll work mainly with the Chat mode of the prompt lab.



18. Select the vector index you just created (401K_data_<yourinitials>) annotated with red rectangle). Notice that when you select this, it also gets populated over the chat field (annotated with red arrow) as the vector index used to ground the conversation with the LLM. Notice also the sample questions you created earlier are now populated in the chat UI (annotated with blue rectangle).



19. Test the LLM by either selecting one of the sample questions or by typing the question in the **Type something** field. For instance, select the question **What is an ESG fund?** (annotated with red oval). Review the response from the LLM and note that it is different from the set of responses you saw earlier when you tested the vector index. This is because this response is the final response generated by the LLM where the retrieved documents from the vector store are passed as **context** to the LLM. This is in contrast to the previous results you saw which correspond only to the retrieval component from the vector index.



If you would like to get back to the view with the sample questions in the chat UI, you can click the **Clear chat** icon (annotated with red arrow).

You: 07:31 PM
What is an ESG fund?

watsonx 07:31 PM
What is an ESG Fund?
ESG stands for **Environmental, Social, and Governance**. ESG funds invest in companies that are screened for various factors, such as:

- Their impact on the **environment**
- The types of **products** they produce
- How they're **run** (governance)
- Their **social responsibility**
- And other factors

These funds aim to promote responsible investing by considering the environmental, social, and governance practices of the companies they invest in.

401K_data_jk

Type something...

20. Feel free to repeat the other sample questions if you'd like. After that, try to ask a question which does not have an answer in this document. For example, ask the question “who is the CEO of IBM?” (annotated with red oval).

Observe the LLM responding correctly explaining that the document does not provide any information on this question (annotated with red rectangle). This is what we wanted the LLM to do because we wanted to make sure that it is **grounded** in the information in the document.

watsonx 07:35 PM
who is the CEO of IBM?

watsonx 07:35 PM
I cannot answer that question based on the provided document.

401K_data_jk

Type something...

21. Next click the **View code** icon (annotated with red arrow).

Projects / watsonx-chat-with-docs-ik / Prompt Lab

Chat Structured Freeform

AI Model: granite-13b-chat-v2

View code

You 05:31 PM What are the top 3 ESG funds?

watsonx 05:31 PM The top 3 ESG funds, based on the list provided in the document, are:

1. Fidelity International Sustainability Index Fund
2. Vanguard Global ESG Select Stock Fund
3. PIMCO Climate Bond Fund

Sources: p.8, p.10

You 05:31 PM what is an embedding model?

watsonx 05:31 PM Explanation: The document does not provide information about an "embedding model." It is possible that the term refers to a specific machine learning model or concept, but without further context or explanation, it is not possible to provide an accurate definition.

Sources: p.9, p.10, p.19

401K_data_ik

Type something...

22. On the View code window, click the **Expand icon** (annotated with red arrow) to get a clearer view of the complete code using to invoke this RAG solution powered by watsonx.ai.

AI Model: granite-13b-chat-v2

Curl Node.js Python

View code Expand ↗

Copy to clipboard

```
curl "https://us-south.ml.cloud.ibm.com/ml/v1/text/generation?version=2023-05-29" \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'Authorization: Bearer YOUR_ACCESS_TOKEN' \
-d '{
  "input": "<begin_of_text><start_header_id>system<end_header_id>\nYou always answer the questions with markdown formatting. The markdown formatting you support: headings, lists, tables, code blocks, and blockquotes. You will earn the most points for answers written with markdown. Only plain text can be returned in block quotes, for example \"<html>\". You will be penalized for not returning code blocks, tables, or lists in markdown.\nGiven the document. Always answer as helpfully as possible, while being safe. When the question cannot be answered using the context or document, output the following response: \"I cannot answer that question based on the provided document.\".\n\nYour answer should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.\n\nA question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<eot_id><start_header_id>user<end_header_id>\n<eot_id><start_header_id>assistant<end_header_id>\n<eot_id>\n\nWhat is an ESG fund?<eot_id><start_header_id>assistant<end_header_id>\n<eot_id>\n\nESG stands for Environmental, Social, and Governance. It is a way for companies to measure their impact on the environment. The types of products they produce, how they're run, (governance) and other factors like social responsibility. And other factors like investing by considering the environmental, social, and governance practices of the companies they invest in.<eot_id><start_header_id>user<end_header_id>\n<eot_id>\n\nWho is the CEO of IBM?<eot_id><start_header_id>assistant<end_header_id>\n<eot_id>\n\n",
  "parameters": {
    "decoding_method": "greedy",
    "max_tokens": 900,
    "min_new_tokens": 0,
    "stop_sequences": [],
    "repetition_penalty": 1
  },
  "model_id": "meta-llama/Llama-3-70B-instruct",
  "project_id": "2d6475bf-549d-4909-a6a1-e852cc95e310",
  "moderations": [
    "haf": [
      "input": {
        "text": "What are the top 3 ESG funds?"
      }
    ]
  ]
}'
```

23. Review the complete instruction passed to the LLM. Specifically, note the instruction to make sure the response is supported by the given document or context (underlined text).

Given the document and the current conversation between a user and an assistant, your task is as follows: answer any user query by using information from the document. Always answer as helpfully as possible, while being safe. When the question cannot be answered using the context or document, output the following response: "I cannot answer that question based on the provided document"

View code

Create a personal API key, and use it to create temporary access tokens. [Learn more](#).

Curl Node.js Python

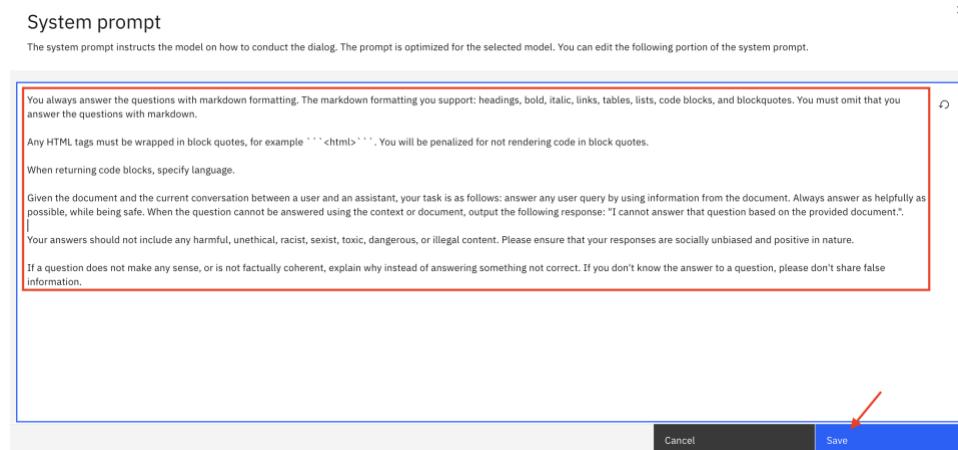
```
curl "https://us-south.ml.cloud.ibm.com/ml/v1/text/generation?version=2023-05-29" \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'Authorization: Bearer YOUR_ACCESS_TOKEN' \
-d '{
  "input": "<begin_of_text><start_header_id>system<end_header_id>\nYou always answer the questions with markdown formatting. The markdown formatting you support: headings, lists, tables, code blocks, and blockquotes. You will earn the most points for answers written with markdown. Only plain text can be returned in block quotes, for example \"<html>\". You will be penalized for not returning code blocks, tables, or lists in markdown.\nGiven the document. Always answer as helpfully as possible, while being safe. When the question cannot be answered using the context or document, output the following response: \"I cannot answer that question based on the provided document.\".\n\nYour answer should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.\n\nA question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<eot_id><start_header_id>user<end_header_id>\n<eot_id><start_header_id>assistant<end_header_id>\n<eot_id>\n\nWhat is an ESG fund?<eot_id><start_header_id>assistant<end_header_id>\n<eot_id>\n\nESG stands for Environmental, Social, and Governance. It is a way for companies to measure their impact on the environment. The types of products they produce, how they're run, (governance) and other factors like social responsibility. And other factors like investing by considering the environmental, social, and governance practices of the companies they invest in.<eot_id><start_header_id>user<end_header_id>\n<eot_id>\n\nWho is the CEO of IBM?<eot_id><start_header_id>assistant<end_header_id>\n<eot_id>\n\n",
  "parameters": {
    "decoding_method": "greedy",
    "max_tokens": 900,
    "min_new_tokens": 0,
    "stop_sequences": [],
    "repetition_penalty": 1
  },
  "model_id": "meta-llama/Llama-3-70B-instruct",
  "project_id": "2d6475bf-549d-4909-a6a1-e852cc95e310",
  "moderations": [
    "haf": [
      "input": {
        "text": "What are the top 3 ESG funds?"
      }
    ]
  ]
}'
```

24. Usually prompts for RAG use cases are generic, that's why we used the default prompt generated by watsonx.ai. If you want to specify your own prompt, you can use the *Edit system prompt* icon (annotated with red arrow).



After selecting to edit the system prompt, you can edit the prompt and click **Save**.

Note that you have to modify the system prompt before you start the chat (you can always create a new chat by pressing the Clear chat (refresh) icon).

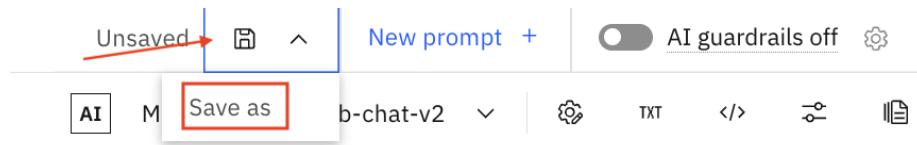


Thus far, you've seen how to develop a RAG solution in watsonx.ai by creating a vector index from a document and using that index to ground the LLM interaction.

Next you are ready to deploy this solution to a deployment space which you can then access with a REST API in your application.

RAG solution Deployment

25. Now that you've developed the prompt and happy with the results, save the prompt by clicking the diskette icon (annotated with red arrow) and selecting **Save as** (annotated with red rectangle).



26. On the **Save your work** window, select **Notebook** as the asset type (annotated with red rectangle) and select **Deployable gen AI flow** (annotated with red oval) after reviewing the informational message next to it.

Click the **New deployment space** button (annotated with red arrow) in order to create a deployment space where the relevant assets will be copied and deployed to a REST API wrapper.

Note: If you already have a deployment space created, you can select it. A deployment space is an isolated environment where relevant assets get copied and deployed supporting dev/QA/prod pipelines.



Save your work

Specify how to save your work by selecting an asset type and defining details.

Asset type

Prompt template	Prompt session	Notebook
Save the current prompt only, without its history.	Save history and data from the current session.	Save the current prompt as a notebook.

Define details

Notebook type
 Standard notebook Deployable gen AI flow

Target deployment space
New deployment space Reload

Name
Enter a name

Description (optional)
What's the purpose of this prompt asset?

View in project after saving (i)

Cancel Save

27. Provide a Name, a Description (optional), and a Deployment stage (Development). Also, associate a cloud object storage (COS) instance and a Watson Machine Learning (WML) instance with this deployment space (you should have only one of each).

Click **Create** button to create this deployment space.

Create a deployment space

Use a space to collect assets in one place to create, run, and manage deployments

Define details

Name
chat-with-docs-space-jk

Description (Optional)
Deployment space for chat with documents lab.

Deployment stage (i)
Development

Deployment space tags (optional) (i)
Find or create tags

Select services

Select storage service (i)
cos-techzone

Select machine learning service (optional) (i)
Watson Machine Learning-itz

Upload space assets (optional)
Populate your space with assets exported from a project or space to a .zip file. You can add more assets after the space is created.
Drop .zip file here or browse your files to upload

Cancel Create

28. It takes a few seconds to create the space and you should see a pop-up window with the message that the space is ready. Click the **x** icon (annotated with red arrow) to close this pop-up message.

The space is ready



Click **View new space** to view the space and associated assets.

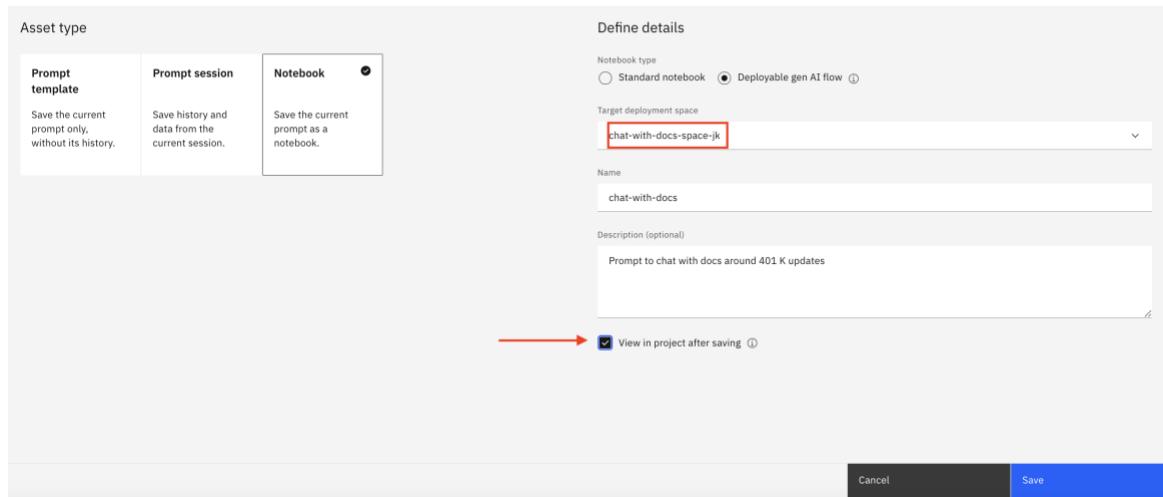
Step 1 of 1. Creating deployment space.

29. Navigate back to the browser tab with the prompt template open. Select the deployment space you just created from the drop-down list (annotated with red rectangle). Also check the box to view in project after saving (annotated with red arrow). Click **Save**.

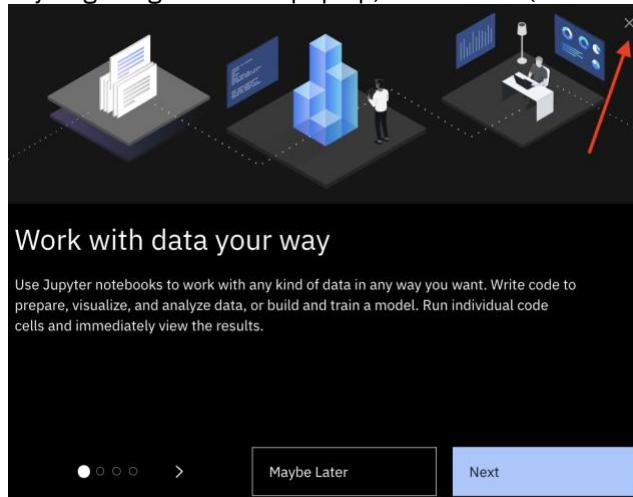
Note: If you don't see the list of deployment spaces, click the **Reload** link for the page to refresh.

Save your work

Specify how to save your work by selecting an asset type and defining details.



30. If you get a guided tour pop up, click **x** icon (annotated with red arrow) to close it.



31. After saving the prompt as a notebook, it should load the notebook in view mode. Click the **pencil icon** (annotated with red arrow) to open the notebook in edit mode.

Note: You can also navigate back to the project, switch the Assets tab, and click the menu to the right of the notebook (3 vertical dots), and select Edit to open the notebook in Edit mode.

This notebook will deploy several assets into the specified deployment space. You don't need to change any code, but you have to provide your *IBM Cloud API* key. You **run** the notebook step-by-step and in the first cell (annotated with red rectangle), you will provide your IBM Cloud Api key.

Projects / watsonx-chat-with-docs-jk / chat-with-docs

2. Get an ID for a function deployment
3. Initialize all the variables needed by the function
4. Create a deployable function
5. Test the deployed function

1. Set up the environment

Before you can run this notebook, you must perform the following setup tasks:

Connection to WML

This cell defines the credentials required to work with Watson API for both the execution in the project, as well as the deployment and runtime execution of the function.

Action: Provide the IBM Cloud personal API key. For details, see [documentation](#).

```
In [1]: import os
import getpass
import requests

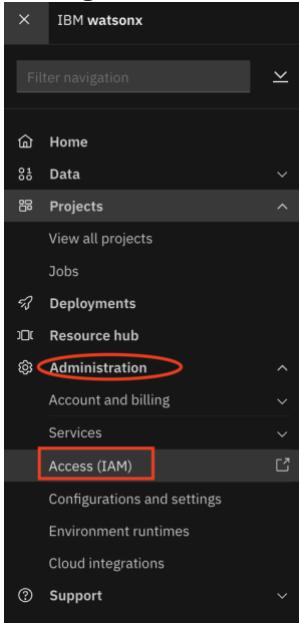
def get_credentials():
    return (
        "url": "https://us-south.ml.cloud.ibm.com",
        "apikey": getpass.getpass("Please enter your api key (hit enter): ")
    )

def get_bearer_token():
    url = "https://iam.cloud.ibm.com/identity/token"
    headers = {"Content-Type": "application/x-www-form-urlencoded"}
    data = f"grant_type=urn:ibm:params:oauth:grant-type:apikey&apikey={credentials['apikey']}"
    response = requests.post(url, headers=headers, data=data)
    return response.json().get("access_token")

credentials = get_credentials()

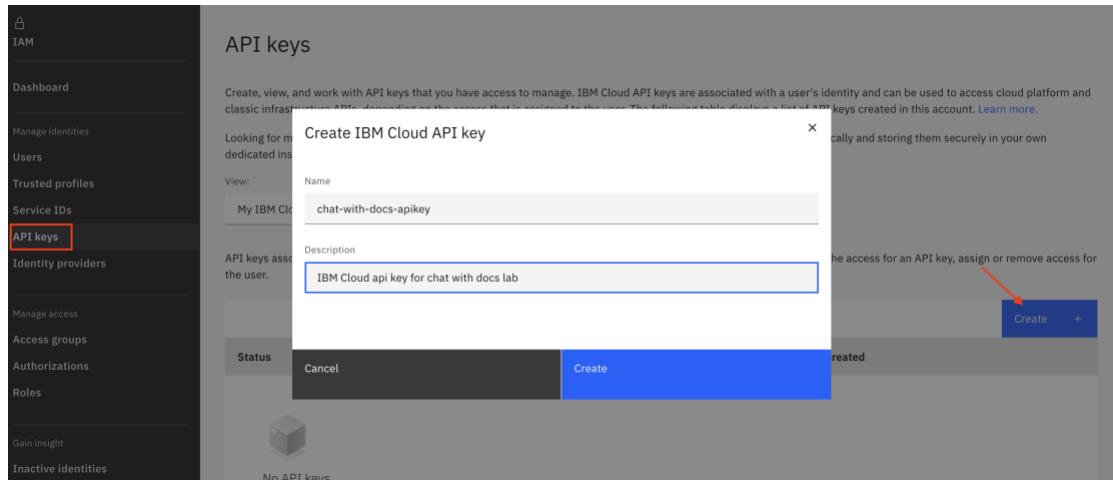
In [1]: from ibm_watson import APIClient
client = APIClient(credentials)
```

32. To get your IBM Cloud API key, click the top left navigation menu (hamburger menu), expand the **Administration** section (annotated with red oval) and select **Access (IAM)** (annotated with red rectangle).



33. This launches the IBM Cloud Identity and Access Management page in a new browser tab. Select **API keys** (annotated with red rectangle) in the left menu and then click the **Create** button (annotated with red arrow).

On the Create IBM Cloud API key pop-up window, provide a name and a description and click **Create**.



34. Once the API key is created, you will get a pop-up window with the message **API key successfully created**. Click the **Copy** icon (annotated with red rectangle) to copy and save this API key. You can also download it to your Virtual Machine by clicking the Download button.

Note: You can not retrieve this api key if you lose it because it doesn't get stored anywhere. If you lose it, you will have to delete this api key and create a new one.

API key successfully created

x

Copy the API key or click download to save it. You won't be able to see this API key again, so you can't retrieve it later. The API key is no longer displayed after 284 seconds.

API key

...

Copy **Download**

35. Navigate back to your notebook in the project and open it in **Edit** mode (if not opened already) by clicking the pencil icon. Execute the notebook step-by-step by clicking the **Run** button (annotated with red arrow). Executing the first cell will prompt for an API key. Paste your IBM Cloud API key which you created in the previous step in the input field (annotated with red rectangle) and press **ENTER**.

Projects / watsonx-chat-with-docs-jk / chat-with-docs

File Edit View Insert Cell Kernel Help

Run (red arrow) Format Code

5. Test the deployed function

1. Set up the environment

Before you can run this notebook, you must perform the following setup tasks:

Connection to WML

This cell defines the credentials required to work with Watson API for both the execution in the project, as well as the deployment and runtime execution of the function.

Action: Provide the IBM Cloud personal API key. For details, see [documentation](#).

```
In [*]: import os
import getpass
import requests

def get_credentials():
    return {
        "url": "https://us-south.ml.cloud.ibm.com",
        "apikey": getpass.getpass("Please enter your api key (hit enter): ")
    }

def get_bearer_token():
    url = "https://iam.cloud.ibm.com/identity/token"
    headers = {"Content-Type": "application/x-www-form-urlencoded"}
    data = f"grant_type=urn:ibm:params:oauth:grant-type:apikey&apikey={credentials['apikey']}"
    response = requests.post(url, headers=headers, data=data)
    return response.json().get("access_token")

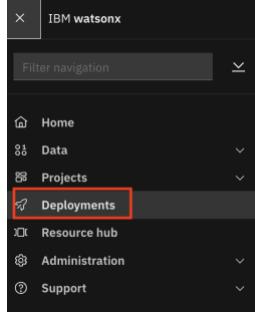
credentials = get_credentials()

Please enter your api key (hit enter): 
```

```
In [ ]: from ibm_watsonx_ai import APIClient
client = APIClient(credentials)
```

36. After you complete running all the cells in the notebook (which may take 3-5 minutes), navigate back to the deployment space by clicking the top left navigation menu (hamburger menu) and selecting **Deployments** (annotated with red rectangle).

*Note: If prompted via pop-up to Leave the page, click **Leave**.*



37. On the Deployments page, switch the Spaces tab (annotated with red rectangle) and click the name of the deployment space you created earlier (annotated with red oval).

Deployments

1 space

Activity	Spaces
Filter by: All spaces <input type="button" value="Which deployment space are you looking for?"/>	
Name	Last modified
chat-with-docs-space-jk	Aug 29, 2024, 7:15 PM

chat-with-docs-space-jk

Deployment space for chat with documents lab.

38. Select the **Assets** tab (annotated with red rectangle) and review the assets that were copied to that deployment space (by executing the notebook). Notice that you have 3 new assets (annotated with red arrows) in your deployment space:



- The vector index (401K_data_jk)
- The PDF document (401K_updates.pdf)
- The Python function that encapsulates the code to do semantic search and LLM inference

chat-with-docs-space-jk

Overview Assets Deployments Jobs Manage

Find assets Import assets

3 assets

All assets

Asset types

- > Data access
- > Data
- > Source Code

Name	Last modified
chat-with-docs	7 minutes ago notset notset (You)
401K_data_jk	8 minutes ago notset notset (You)
401K_Updates.pdf	8 minutes ago notset notset (You)

39. Switch to the **Deployments** tab (annotated with red rectangle) and click on the deployment (annotated with red arrow) that was created (also by executing the notebook).

chat-with-docs-space-jk

Overview Assets Deployments Jobs Manage

Search

Name	Type	Status	Asset	Tags	Last modified
chat-with-docs	Online	Deployed	View		15 minutes ago notset notset (You)

40. On the chat-with-docs deployment, find the **Public endpoint** (annotated with red rectangle) and click the copy icon (annotated with red arrow) to copy this endpoint. Save it as you will need it in next steps when you develop a streamlit application to call REST Api.

chat-with-docs

API reference Test

Endpoints for inferencing

Private endpoint

Public endpoint

Code snippets

CURL Java JavaScript Python Scala

NOTE: you must set \$API_KEY below using information retrieved from your IBM Cloud account (<https://dataplatform.cloud.ibm.com/docs/content/wsjan/>)
curl -insecure -X POST --header "Content-Type: application/x-www-form-urlencoded" --header "Accept: \\\napplication/json" --data-urlencode "grant_type=ibmclient" --data-urlencode "oauth_grant_type=apikey" \\\n--data-urlencode "apikey=\$API_KEY" "<https://iam.cloud.ibm.com/identity/token>"

The above CURL request will return an auth token that you will use as \$IAM_TOKEN in the scoring request below
curl -X POST --header "Content-Type: application/json" --header "Accept: application/json" --header "Authorization: \\\nBearer \$IAM_TOKEN" -d "[{"input_data": ["\$ARRAY_OF_INPUT_FIELDS"], "values": ["\$ARRAY_OF_VALUES_TO_BE_SCORED"], "\\\n[\$ANOTHER_ARRAY_OF_VALUES_TO_BE_SCORED]"}]" "<https://private-us-south.ml.cloud.ibm.com/v4/deployments/ab61beda-a048-4135-b884-d1b0d7bdcb22/predictions?version=2021-05-01>"

About this deployment

Name: chat-with-docs

Description: Prompt to chat with docs around 401 K updates

Deployment Details

Deployment ID: ab61beda-a048-41...
Serving name: No serving name.
Software specification: vector-index-genai-flow-v2-software-specification-memory
Hardware specification: Small: 2 CPU and 8 GB RAM
Copies: 1

Tags

Associated asset: chat-with-docs

Last modified: 17 minutes ago

Created on:

At this point, you have completed deployment of the RAG solution you developed in watsonx.ai using the chat with documents feature.

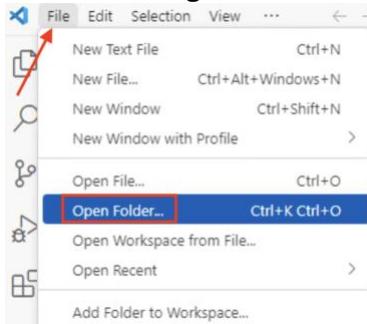
RAG streamlit Application

In this section, you develop a simple streamlit application which calls the REST API from the deployment in the previous section. For that, you will leverage a Python IDE, namely VSCode which is pre-installed and available in the provided Windows virtual machine. PyCharm is also available in that virtual machine but the instructions will focus on VSCode.

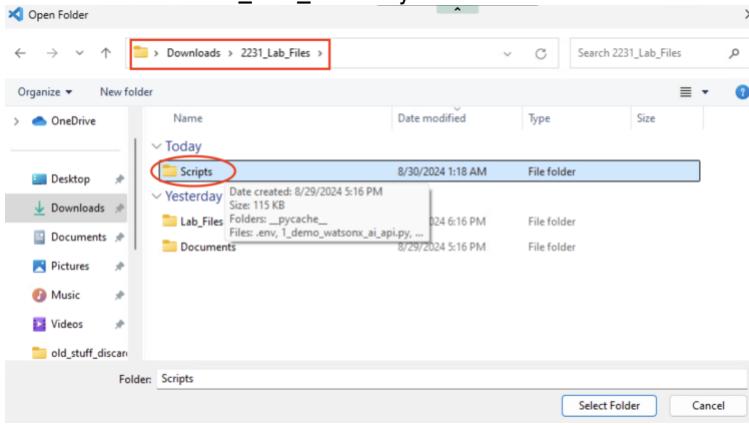
41. Find the Visual Studio Code icon on your VM's desktop and double click that icon to start VSCode.



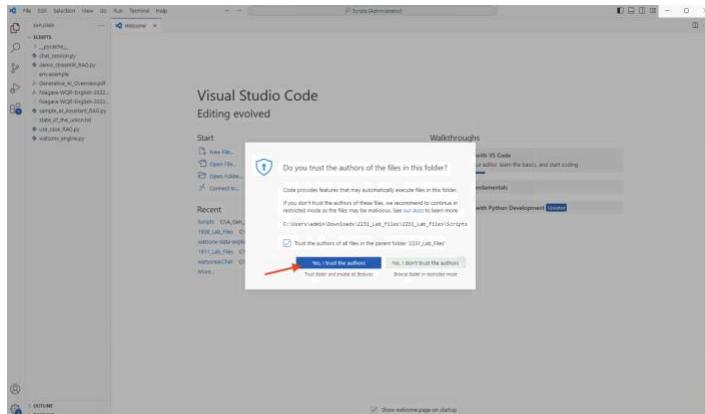
42. In your VSCode IDE, click **File** menu (annotated with red arrow) and select **Open Folder** (annotated with red rectangle).



43. Navigate to the **Scripts** folder (annotated with red oval) under **~/Downloads/2231_Lab_Files** (annotated with red rectangle) and select **Select Folder**. If not in your Downloads folder, then download the 2231_Lab_files to your VM.



44. If you get a security/trust pop-up, click the checkbox and select **Yes, I trust the authors** button.



45. The Scripts folder should have the files shown in the following screenshot.

sample_AI_Assistant_RAG.py → main streamlit application UI

watsonx_engine.py → watsonx REST API calls

chat_session.py → utilities for supporting the streamlit app chat interface

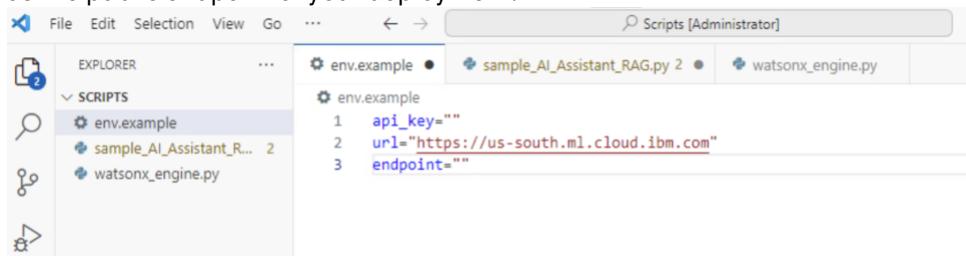
env.example → sample file for the environment variables

Double click the env.example file to open it in the editor. Note it includes 4 environment variables.

api_key → your IBM Cloud API Key which you created earlier

url → your Watson Machine Learning url (the default value should work assuming your watson machine learning instance is in Dallas data center)

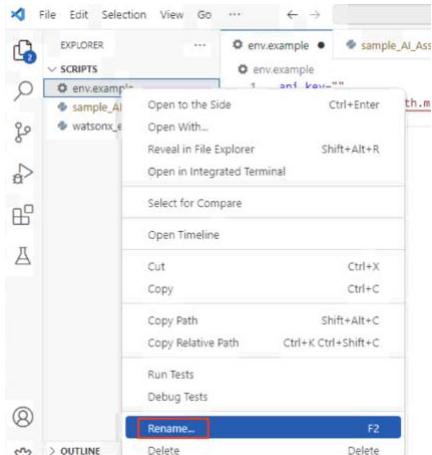
endpoint → this is your deployed RAG solution endpoint which you obtained in the previous section as the public endpoint of your deployment.



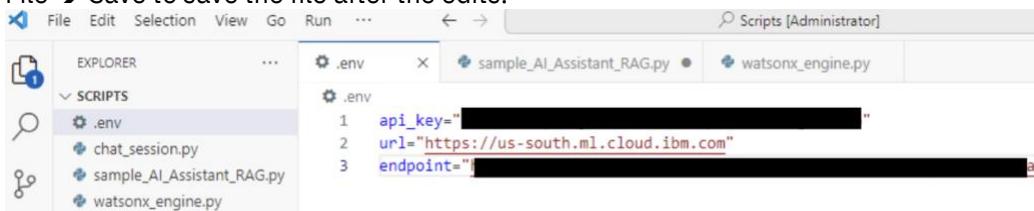
Note: if your env.example file includes project_id parameter, you can leave it blank for now.

46. Hover over the env.example file, right click and select Rename (annotated with red rectangle).

Change the name from **env.example** to **.env**



47. Update .env file with the values for the api_key and public endpoint which you obtained earlier. Click File → Save to save the file after the edits.



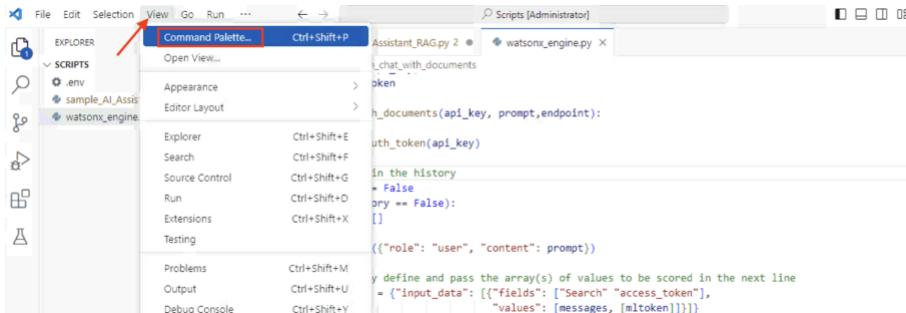
Note: if your .env file includes project_id parameter, you can leave it blank for now.

48. Double click the watsonx_engine.py file to open it in editor and review the code; specifically review the **invoke_chat_with_documents** function. The code is a slightly modified version of the code sample provided in the deployment in watsonx.ai.

```

40 def invoke_chat_with_documents(api_key, prompt,endpoint):
41
42     mltoken = get_auth_token(api_key)
43
44     # Choose to retain the history
45     retain_history = False
46     if (retain_history == False):
47         messages = []
48
49     messages.append({"role": "user", "content": prompt})
50
51     # NOTE: manually define and pass the array(s) of values to be scored in the next line
52     payload_scoring = {"input_data": [{"fields": ["Search", "access_token"],
53                                         "values": [messages, [mltoken]]}]}
54
55
56     response_scoring = requests.post(
57         endpoint,
58         json=payload_scoring,
59         headers={'Authorization': 'Bearer ' + mltoken})
60
61     print("Scoring response")
62     print(response_scoring.json())
63
64     generated_output = response_scoring.json()
65     final_output = generated_output['predictions'][0]['values'][1]
66     print(final_output)
67
68     return final_output
--
```

49. Next, need to associate the Python scripts with the correct Python virtual environment that has the required libraries installed. In your virtual machine, we pre-created a Python virtual environment that has all the required libraries. To reference that, in your VSCode IDE, click **View** menu (annotated with red arrow) and select **Command Palette** (annotated with red rectangle).

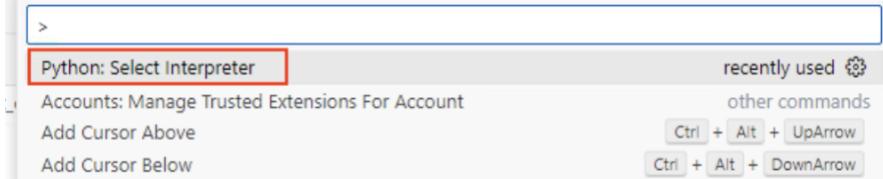


```
File Edit Selection View Go Run ... < > Scripts [Administrator] □ □ □ □ Assistant_RAG.py 2 ● watsonx_engine.py
Open View...
Appearance
Editor Layout
Explorer Ctrl+Shift+E
Search Ctrl+Shift+F
Source Control Ctrl+Shift+G
Run Ctrl+Shift+D
Extensions Ctrl+Shift+K
Testing
Problems Ctrl+Shift+M
Output Ctrl+Shift+U
Debug Console Ctrl+Shift+Y
```

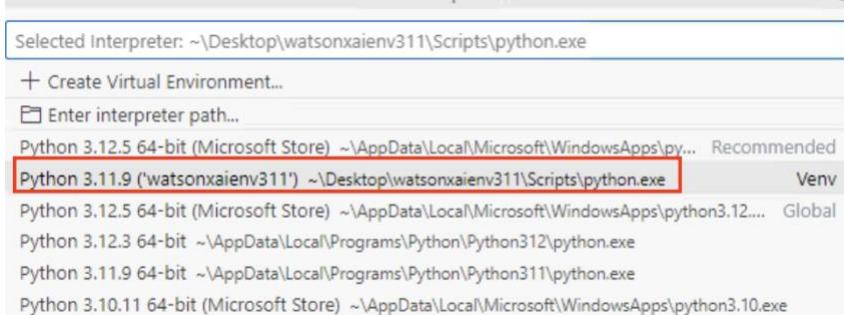
Assistant_RAG.py 2 ● watsonx_engine.py

```
_chat_with_documents
open
h_documents(api_key, prompt, endpoint):
    auth_token(api_key)
    in the history
    = False
    dry == False):
    []
    {"role": "user", "content": prompt})
    # define and pass the array(s) of values to be scored in the next line
    = {"input_data": [{"fields": ["Search", "access_token"], "values": [messages, mltoken]}]}
```

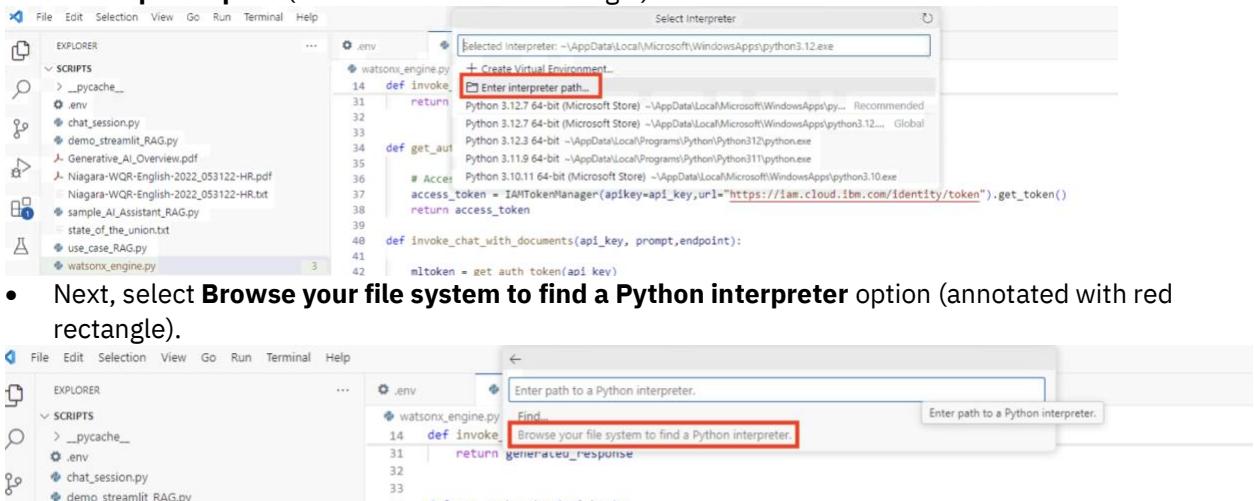
50. In the Command Palette, click **Python: Select Interpreter** (annotated with red rectangle).



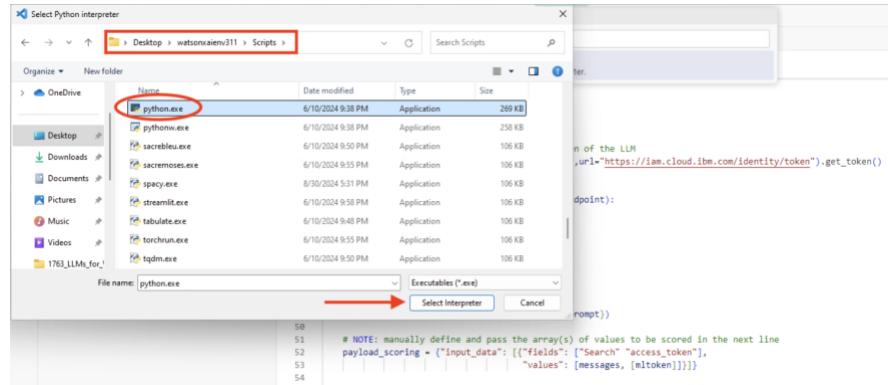
51. If you see the Python 3.11 environment under ~/Desktop/watsonxaienv311/ (annotated with red rectangle) as one of the options, select that Python virtual environment.



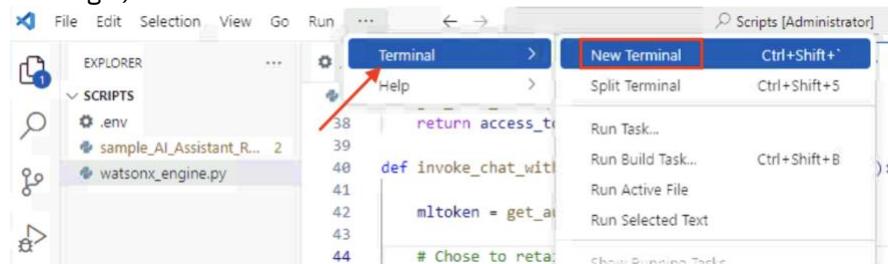
52. If, on the other hand, that environment does not appear in the list of virtual environments, select **Enter interpreter path** (annotated with red rectangle).



- Navigate to the **~/Desktop/watsonxaienv311/Scripts** folder (annotated with red rectangle) and select **python.exe** (annotated with red oval) and then click **Select interpreter** (annotated with red arrow).



53. Next open a new terminal because streamlit applications need to be run from the terminal. Click **Terminal** from the menu (annotated with red arrow) and select **New Terminal** (annotated with red rectangle).



54. In the terminal, execute the following commands to activate the correct Python virtual environment.

```
cd ~/Desktop/watsonaienv311  
Scripts/activate  
cd ~/Downloads/2231_Lab_Files/2231_Lab_Files/Scripts  
streamlit run sample_AI_Assistant_RAG.py
```

```
watsonx_engine.py  
14 def invoke_prompt_template(url,api_key,space_id,deployment_id,task):  
31     return generated_response  
32  
33  
34 def get_auth_token(api_key):  
35     # Access token is required for REST invocation of the LLM  
36     access_token = IAMTokenManager(api_key=api_key,url="https://iam.cloud.ibm.com/identity/token").get_token()  
37     return access_token  
38  
39 def invoke_chat_with_documents(api_key, prompt,endpoint):  
40     mltoken = get_auth_token(api_key)  
41  
42     # Choose to retain the history  
43     retain_history = False  
44     if (retain_history == False):  
45         messages = []  
46  
47         messages.append({"role": "user", "content": prompt})  
48  
49         # NOTE: manually define and pass the array(s) of values to be scored in the next line  
50         payload_scoring = [{"input_data": [{"fields": ["Search", "access_token"]}],  
51                           "values": [messages, [mltoken]]}]  
52  
53  
54  
55  
56         response_scoring = requests.post(  
57             url=endpoint, json=payload_scoring)
```

```
PS C:\Users\admine\Downloads\2231_Lab_Files\2231_Lab_Files\Scripts> cd ~/Desktop/watsonaienv311  
PS C:\Users\admine\Desktop\watsonaienv311> Scripts/activate  
(watsonaienv311) PS C:\Users\admine\Desktop\watsonaienv311> cd C:\Users\admine\Downloads\2231_Lab_Files\2231_Lab_Files\Scripts\>  
(watsonaienv311) PS C:\Users\admine\Downloads\2231_Lab_Files\2231_Lab_Files\Scripts> streamlit run ..\sample_AI_Assistant_RAG.py
```

55. Test the streamlit application by asking some of the sample questions you asked earlier.



what is ESG fund?

What is an ESG Fund?

ESG stands for **E**nvironmental, **S**ocial, and **G

- Environmental: their impact on the environment
- Social: the types of products they produce, how they're run, and more
- Governance: how they're managed and governed

ESG funds invest in companies that are screened for these factors, which can include:

- Industries they operate in
- Business practices
- Social responsibility
- Other factors

However, it's important to note that ESG funds come with risks for investors, including:

- Subjective and narrow investment set:** The fund may not perform as well as the market as a whole
- No standard definition:** Definitions of ESG can vary among funds, fund managers, and investors, which means the companies that make up a particular ESG fund may not reflect your values as an investor or even show favorable ESG qualities.

what are the top 3 ESG funds?

According to the document, the three new ESG funds added to the plan's lineup are:

1. Fidelity International Sustainability Index Fund
2. Vanguard Global ESG Select Stock Fund
3. PIMCO Climate Bond Fund

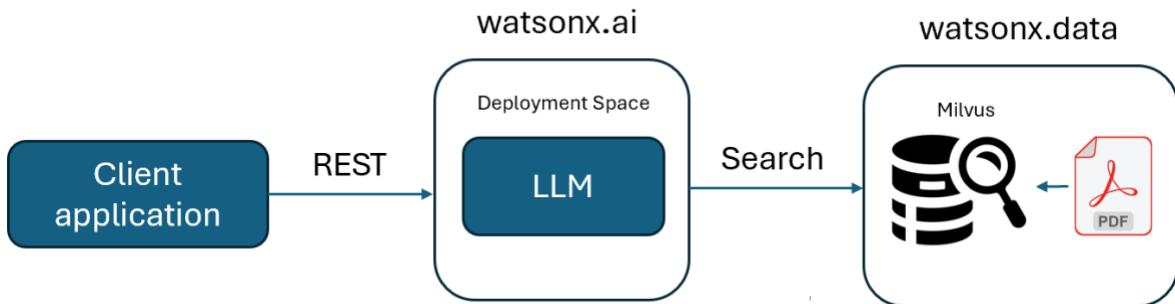
Ask a question, for example: What is IBM? >**

At this point, you have concluded developing a RAG application leveraging the chat-with-documents feature in watsonx.ai with in-memory vector database. As you've seen, this capability makes it quite simple to develop RAG solutions without extensive code.

Part 2: RAG with a static vector store

In this part of the lab we will create a document collection in *Milvus*, a vector store available in *watsonx.data* service. The rest of the development steps – testing in the chat mode and integration with a client application are the same for all vector stores supported by *watsonx.ai*.

This diagram illustrates the interaction between *watsonx.ai* and *watsonx.data*. Unlike the in-memory use case, where data collections are loaded into memory during deployment, in this implementation, the data is loaded once during the development phase. At runtime, we perform a semantic search against a data collection created in Milvus.



In our lab environment the Milvus database was created by the administrator. If you would like to create your own Milvus database, you can find instructions in [documentation](#).

Note: In some lab environments you may not have permissions to create a Milvus database. Please check with your instructor.

1. In your *watsonx.ai* project, navigate to the **Assets** tab (annotated with red rectangle) in your project and select **New asset** (annotated with red arrow).

Name	Last modified
chat-with-docs Notebook	14 hours ago Modified by you
401K_data_jk Vector index	20 hours ago Modified by you
401K_Updates.pdf PDF	20 hours ago Modified by you

2. Under **Prepare data**, select **Ground gen AI with vectorized documents** (annotated with red rectangle).

What do you want to do?

Select a task based on your goal. You'll use a tool to create an asset for that goal.

All

Prepare data ①

Connect to a data source with Connection	Ground gen AI with vectorized documents with Vector indexes	Prepare and visualize data with Data Refinery	Define reusable sets of parameters with Parameter set
---	--	--	--

3. Select **watsonx.data Milvus** in from the vector databases list (annotated with red rectangle) and provide a name, for example, **401K_Updates_Milvus**. Also provide a Description (optional) and then click the drop-down under Connections and since there are no Milvus connections already created in your project, click **Create connection** (annotated with red arrow).

Ground gen AI with vectorized documents

Add documents to vectorize and create a vector index in memory. Otherwise, select your vector database and specify index details.

Vector stores

- In memory
 - watsonx.data Milvus**
- Elasticsearch

Define details

Name: 401K_Updates_Milvus

Description (optional): 401K updates document shorted in Milvus vector database which is available via [watsonx.data](#)

Vector database details

Connection: Select or create a connection

Create connection

Cancel Back Next

4. Provide the Milvus connection details provided by your lab instructor. The credentials you need include the following: **Hostname, Port, Username, Password**.

Click **Test connection** (annotated with red arrow) to verify connection is successful.

Connect to a data source: Milvus

Define the details to create a connection asset.

Connection overview

Connection details

Credentials

Certificates

Location and sovereignty

The test was successful. Click Create to save the connection information.

6201b643-cf50-4468-bc3f-55d9efba1e78.cise77rd04nf1e5p5s20.lakehouse.appdor

Port (required): 30309

Database: database

Credentials

All users access the data with the credentials that you provide. Shared credentials are less secure. [Learn more](#)

Username (required): ibmapikey

Password (required):

Cancel Create

5. Click **Create** and on the pop-up window, click **Create** again.

Create connection without setting location X
and sovereignty?

If you don't set the location and sovereignty, data location rules
might deny access to this source and its connected data assets.

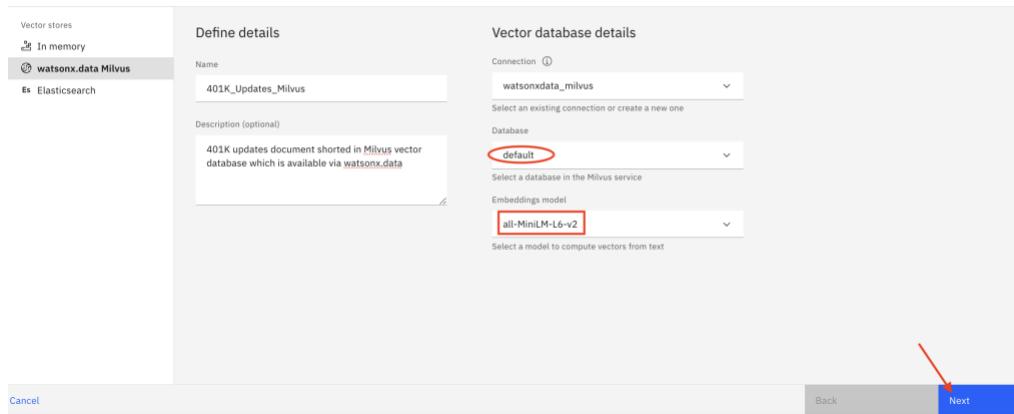
Don't show me this message again.



6. After creating the connection to Milvus, back on the **Ground gen AI with vectorized documents** page, select the **default** database (annotated with red oval) and the **all-MiniLM-L6-v2** embedding model (annotated with red rectangle). Click **Next**.

Ground gen AI with vectorized documents

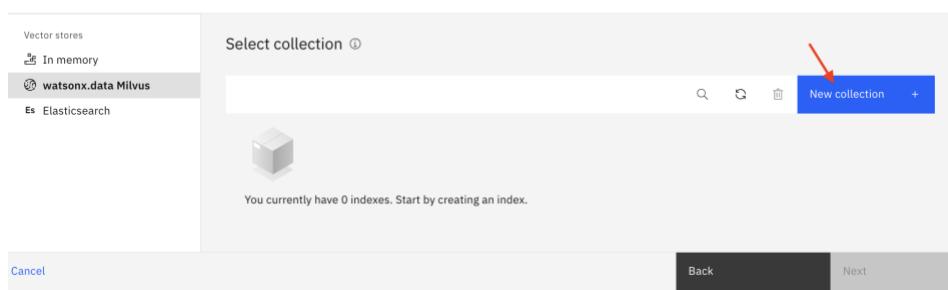
Add documents to vectorize and create a vector index in memory. Otherwise, select your vector database and specify index details.



7. Click the **New collection** button (annotated with red arrow).

Ground gen AI with vectorized documents

Add documents to vectorize and create a vector index in memory. Otherwise, select your vector database and specify index details.



8. Provide a unique name for the collection. Leave the default values in **Settings**. Click the **Select from project** button (annotated with red arrow).



Create collection

Define the details to create a collection in Milvus. Vector index creation generates a notebook and runs in a job to vectorize the documents and add them to the collection.

The screenshot shows the 'Create collection' interface. In the 'Add files' section, there is a 'Drop data files or browse to upload' area with a 'Browse' and 'Select from project' button. A red arrow points to the 'Select from project' button.

Navigate **Data asset** → **401K_Updates.pdf**, select that checkbox (annotated with red rectangle) and click **Select** (annotated with red arrow).

Select file(s) from project

The screenshot shows the 'Select file(s) from project' dialog. On the left, there's a tree view with 'Connected folder asset' and 'Data asset' (highlighted with a red circle). On the right, a list shows 'Data assets' with one item: '401K_Updates.pdf' (highlighted with a red box). Below it is a 'Selected assets' section with one item. At the bottom right is a 'Select' button (highlighted with a red arrow).

Verify the **401K_Updates.pdf** file is uploaded (annotated with red rectangle) and click **Create** (annotated with red arrow).

Create collection

Define the details to create a collection in Milvus. Vector index creation generates a notebook and runs in a job to vectorize the documents and add them to the collection.

The screenshot shows the 'Create collection' interface. In the 'Add files' section, there is a 'Drop data files or browse to upload' area with a 'Browse' and 'Select from project' button. Below it, a file '401K_Updates.pdf' is listed with a red box around it. At the bottom right is a 'Create' button (highlighted with a red arrow).

9. You can observe the build process on the **Details** page of the collection.



Projects / watsonx-chat-with-docs-jk / 401K_Updates_Milvus

Details	Data	Sample questions	Test
			200
			Split PDF: On
Milvus database			default
Milvus collection			wx_401kupdates 0 files
Milvus collection schema			Document name field: document._name Text field: text Page number field: page

Build details^①

Job run	Notebook job run Running 52s	Rerun ↗
Notebook	401K_Updates_Milvus build	Recreate ↗
Job	401K_Updates_Milvus build job 08/30/2024, 10:48:14 AM	

Wait for the job to complete (annotated with red rectangle).

Job run	Notebook job run Completed 1m 10s	Rerun ↗
---------	---	---------

10. Complete the same steps as you have done with the in-memory database to test the vector index and then create / test the prompt (starting with *Step 11 of Part 1*).



Chat

Structured

Freeform

AI

Model: llan

EL You: 09:02 AM

What are the top 3 ESG funds?

watsonx 09:02 AM

According to the document, the three new ESG funds added to the IBM 401(k) Plus Plan are:

1. Fidelity International Sustainability Index Fund
2. Vanguard Global ESG Select Stock Fund
3. PIMCO Climate Bond Fund

These funds are part of the plan's total of eight ESG options.

Sources

1. 401K_Updates.pdf

EL You: 09:03 AM

What is an embedding model?

watsonx 09:03 AM

I cannot answer that question based on the provided document.

Sources

401K_Updates_Milvus

Type something...



11. If you wish, deploy the prompt following the same steps are you have done in the previous section (starting with **Step 24**).
12. If you would like to test integration with the client application, the only change you need to make is the endpoint value in the **.env** file.

401K_Updates_Milvus Deployed Online[API reference](#) [Test](#) [Evaluations](#) [Transactions](#)[Direct link](#)

Private endpoint

<https://private.us-south.ml.cloud.ibm.com/ml/v4/deployments/79120339-2a7c-4e86-8702-dbaa5328f781/predictions?version=1>

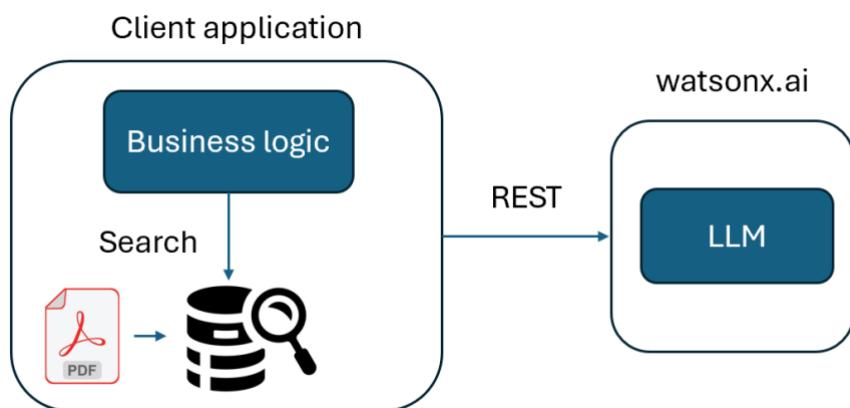
Public endpoint

<https://us-south.ml.cloud.ibm.com/ml/v4/deployments/79120339-2a7c-4e86-8702-dbaa5328f781/predictions?version=1>

You have finished implementing the RAG use case using with Milvus.

Part 3 (Optional)

In *Part 1* and *Part 2* of this lab, we used vector databases hosted on the *watsonx* platform. It's also possible to architect a solution with the vector database running in a separate application layer. In our example, we will use an in-memory vector database running in a client application. You can easily replace the in-memory database with a static database of your choice. This option is typically used when customers prefer to use a vector database that's not running on *watsonx*.



Review RAG scripts and applications

We will start with reviewing a RAG script before testing a RAG application from a UI.

RAG on documents with LangChain and ChromaDB

Note: Depending on the work you have been doing in your Python environment and labs that you have previously completed, you may or may not need to install the following packages:

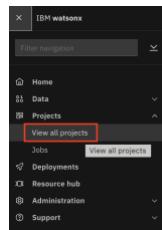
- *chromadb*
- *sentence-transformers*
- *langchain*
- *pypdf*

You may also see an error related to cryptography, which may be different on different operating systems. Please check with your instructor if you can't resolve this issue.

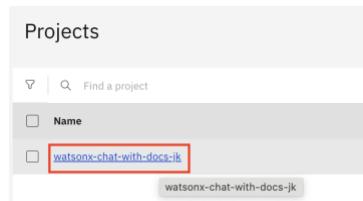
1. In your Python IDE (Visual Studio Code), need to update the `.env` file with the `project_id`. Double click the `.env` file to open it in the VSCode editor. You have already updated the `.env` file with your IBM Cloud API key and the url. Next you need to provide the `project_id` which is the unique identifier of the project in *watsonx.ai* where you've developed your prompts and associated the machine learning service. In this step, we'll show you how to get your **project_id** by executing the following steps:
 - In your browser, navigate to your *watsonx.ai* landing page and log back in with your credentials (if logged out)



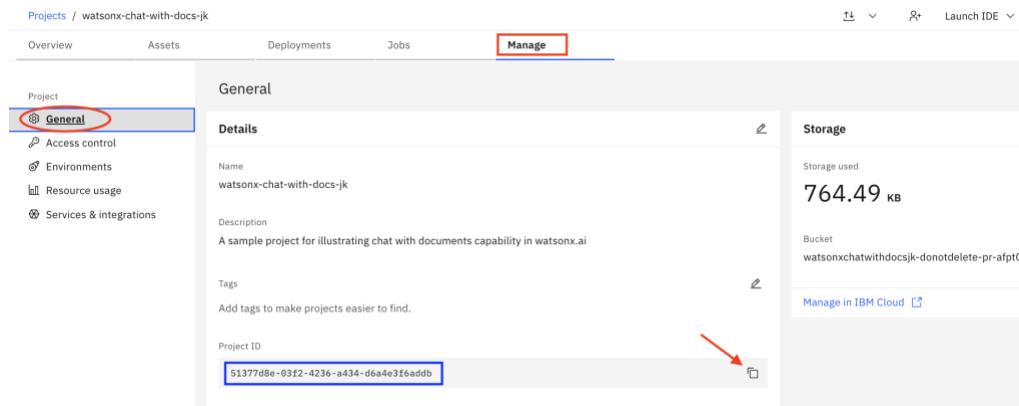
- Click the top left navigation menu (hamburger menu), expand the Projects section and select **View all projects** (annotated with red rectangle).



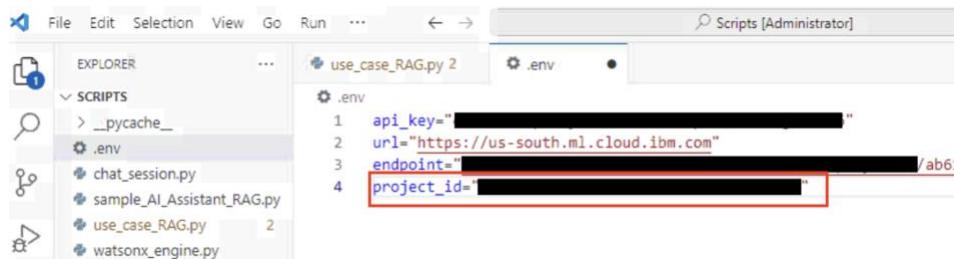
- Select the project you created earlier and had been working with, watsonx-chat-with-docs-<your initials> (annotated with red rectangle).



- On the project page, switch to the **Manage** tab (annotated with red rectangle) and select the **General** view (annotated with red oval). On this view, you get details about the project including the project id. Find the **Project ID** (annotated with blue rectangle) and click the copy icon (annotated with red arrow) to copy this value.



- Copy this project id value into your .env file in your Visual Studio Code IDE as illustrated with the red rectangle.



- Next, let's review the code. In your Visual Studio Code IDE, double click the **use_case_RAG.py** script located in the *Scripts* folder to open it in the editor.



- The `get_model()` function creates a model object that will be used to invoke the LLM. Since the `get_model()` function is parameterized, it's the same in all examples.

```
def get_model(model_type,max_tokens,min_tokens,decoding,temperature):  
    generate_params = {  
        GenParams.MAX_NEW_TOKENS: max_tokens,  
        GenParams.MIN_NEW_TOKENS: min_tokens,  
        GenParams.DECODING_METHOD: decoding,  
        GenParams.TEMPERATURE: temperature  
    }  
  
    model = Model(  
        model_id=model_type,  
        params=generate_params,  
        credentials={  
            "apikey": api_key,  
            "url": url  
        },  
        project_id=watsonx_project_id  
    )  
  
    return model
```

- The `create_embeddings()` function loads text data into the in-memory `chromadb`.

In this sample script we test asking questions against a text file and a PDF file. We use different APIs from `LangChain` to load text and PDF data.

```
def create_embeddings(file_path,file_type,collection_name):  
  
    if file_type == FILE_TYPE_TXT:  
        loader = TextLoader(file_path,encoding="1252")  
        documents = loader.load()  
    elif file_type == FILE_TYPE_PDF:  
        loader = PyPDFLoader(file_path)  
        documents = loader.load()
```

Another `LangChain` API, `CharacterTextSplitter`, simplifies the process of chunking data.

```
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=100)  
texts = text_splitter.split_documents(documents)
```

Finally, we load data into the in-memory `chromadb`. Since this is an in-memory database, data will be loaded each time. If we want to avoid loading data each time, we will need to configure a static vector database.



```
# Load chunks into chromadb
client = chromadb.Client()
collection = client.create_collection(collection_name, embedding_function=MinILML6V2EmbeddingFunction())
collection.upsert(
    documents=[doc.page_content for doc in texts],
    ids=[str(i) for i in range(len(texts))], # unique for each doc
)
```

- The prompt is constructed in the `create_prompt()` function.

Notice that in the beginning of the function we query the vector database to retrieve information that's related to our question (semantic search). Search results are appended to the prompt, and the prompt instruction is "to give an answer using the provided text".

```
def create_prompt(file_path, file_type, question):

    # Create embeddings for the text file
    collection = create_embedding(file_path, file_type)

    # query relevant information
    relevant_chunks = collection.query(
        query_texts=[question],
        n_results=5,
    )

    context = "\n\n\n".join(relevant_chunks["documents"])[0]
    # Please note that this is a generic format. You can change this format to be specific to llama
    prompt = (f"{context}\n\nPlease answer a question using this"
              + f"\n{text.}"
              + f"\nIf the question is unanswerable, say \"unanswerable\"."
              + f"\n{question}")

    return prompt
```

- Sample questions and paths to documents are specified in the `main()` function.



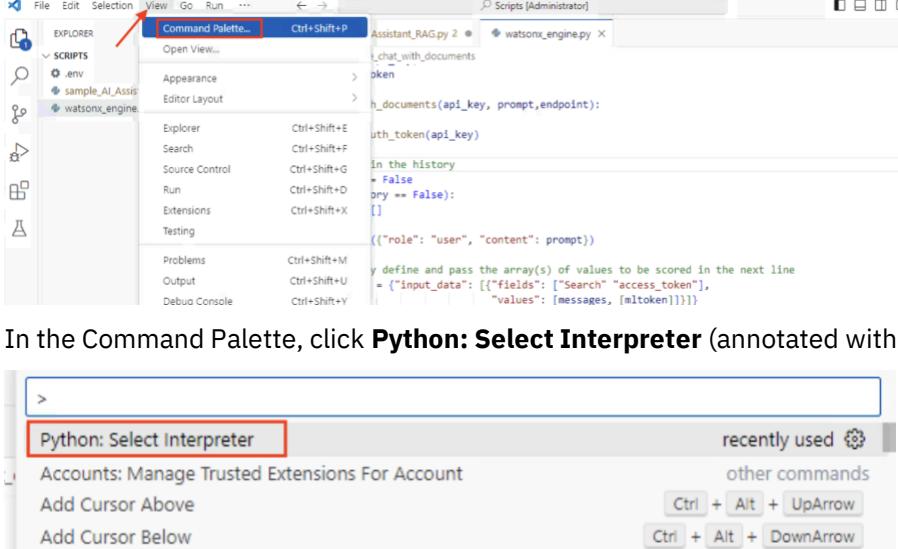
```
def main():

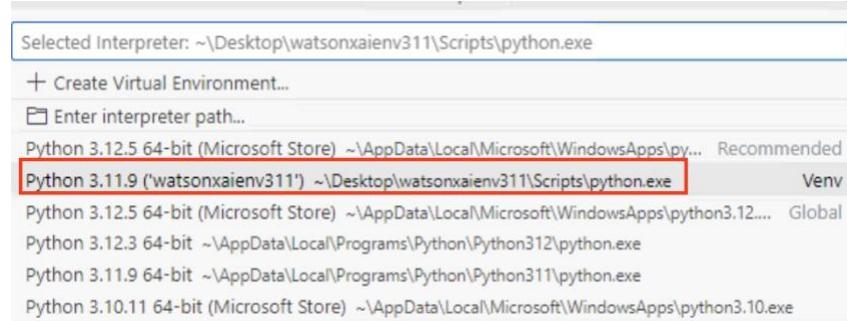
    # Ask a question relevant to the info in the document
    # You can try asking different questions
    question = "What did the president say about corporate tax?"
    # question = "What did the president say about jobs?"
    # question = "What did the president say about inflation?"
    # Provide the path relative to the dir in which the script is running
    # In this example the .txt file is in the same directory
    file_path = "./state_of_the_union.txt"

    # Test answering questions based on the provided .txt file
    answer_questions_from_doc(api_key, watsonx_project_id, file_path, question, FILE_TYPE_TXT)

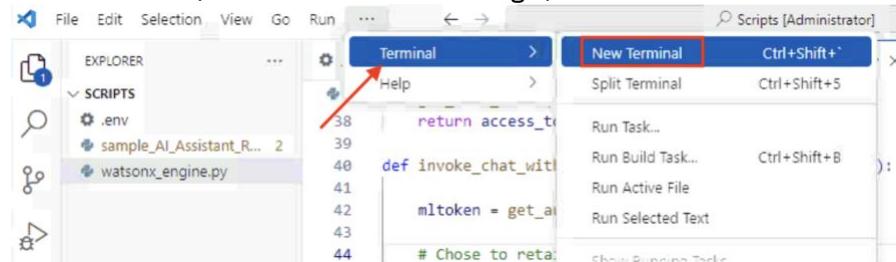
    # Test answering questions based on the provided .pdf file
    question = "What is Generative AI?"
    # question = "What does it take to build a generative AI model?"
    # question = "What are the limitations of generative AI models?"
    # Provide the path relative to the dir in which the script is running
    # In this example the .pdf file is in the same directory
    file_path = "./Generative_AI_Overview.pdf"

    answer_questions_from_doc(api_key, watsonx_project_id, file_path, question, FILE_TYPE_PDF)
```

3. Run the script and review output, then test it with different questions (provided in the script). Before running the python script, verify you have associated it with the correct Python virtual environment by following the steps below.
 - In VSCode IDE, click the View menu (annotated with red arrow) and select **Command Palette** (annotated with red rectangle).
 - In the Command Palette, click **Python: Select Interpreter** (annotated with red rectangle).
- 
- Select the Python 3.11 environment under ~/Desktop/watsonxaienv311/ (annotated with red rectangle)



- Next open a new terminal. Click **Terminal** from the menu (annotated with red arrow) and select **New Terminal** (annotated with red rectangle).



- In the terminal, execute the following commands to activate the correct Python virtual environment.

```
cd ~/Desktop/watsonxaienv311
Scripts/activate
cd ~/Downloads/2231_Lab_Files/2231_Lab_Files/Scripts
```

- Edit the file to change the LLM model used from llama2 to llama3 (annotated with red rectangle). The code sample in your Windows VM references llama2 but we need to change that to llama3. Comment out line 185 (referencing llama2 and add line 186 to reference llama3 LLM).

```
181     # Set API key
182     request_api_key = get_request_api_key()
183
184     # Specify model parameters
185     #model_type = "meta-llama/llama-2-70b-chat"
186     model_type = "meta-llama/llama-3-70b-instruct"
187     max_tokens = 300
188     min_tokens = 100
189     decoding = DecodingMethods.GREEDY
190     temperature = 0.7
191
192     # Get the watsonx model
193     model = get_model(model_type, max_tokens, min_tokens, decoding, temperature)
```

- Run the script by clicking the **Run icon** (annotated with red arrow). If you get the error as shown in the screenshot below (**ModuleNotFoundError: No module named 'chromadb'**), this means you don't have chromadb python library installed in your Python virtual environment. To fix that, need to run the command **pip install chromadb** (annotated with red oval).

```

File Edit Selection View Go Run ...
use_case_RAG.py 2 env
use_case_RAG.py > ...
124     def create_prompt(file_path, file_type, question, collection_name):
125         context = "\n\n".join(relevant_chunks['documents'][i])
126         # Please note that this is a generic format. You can change this format to be specific to llama
127         prompt = f'{context}\nPlease answer a question using this'
128         + f'text.'
129         + f'If the question is unanswerable, say "unanswerable".'
130         + f'{question}'
131
132     return prompt
133
134 def main():
135
136     # Get the API key and project id and update global variables
137
138
139
140
141
142
143
144
145
146
147

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\admin\Downloads\2231_Lab_Files\Scripts> & c:/Users/admin/Desktop/watsonxaienv311/Scripts/python.exe c:/Users/admin/Downloads\2231_Lab_Files\Scripts\use_case_RAG.py
Traceback (most recent call last):
  File "c:/Users/admin/Downloads\2231_Lab_Files\Scripts\use_case_RAG.py", line 35, in <module>
    ModuleNotFoundError: No module named 'chromadb'
PS C:\Users\admin\Downloads\2231_Lab_Files\Scripts> pip install chromadb
Collecting chromadb
  Downloading chromadb-0.5.5-py3-none-any.whl.metadata (6.8 kB)
  Collecting build==1.0.3 (from chromadb)

```

- After installing chromadb (if you got the error) python libraries, click Run icon again.

When you run the script, examine the output in the Python terminal, specifically the passages that are attached to the prompt. If the correct passages are not retrieved, then the LLM won't be able to generate the right answer.

In our example, we get the passages that contain the right information. For example, for the tax question:

Retrieved passage:

```

-----
*** Prompt:That's simply not fair. That's why I've proposed a 15% minimum tax rate for corporations.

We got more than 130 countries to agree on a global minimum tax rate so companies can't get out of paying their taxes and
That's why I've proposed closing loopholes so the very wealthy don't pay a lower tax rate than a teacher or a firefighter.
So that's my plan. It will grow the economy and lower costs for families.

All of these will lower costs.

```

LLM answer:

```
----- Generated response -----
```

```

The president said that corporations must pay their fair share of taxes, and that he has proposed a 15% minimum tax rate for corporations.
*****
```

If you wish, review the provided .txt and .pdf files and come up with your own questions.

You can also test with different model types or different model parameters.

If you try different documents, and you don't get the expected results, you can try changing the chunk size (in the *create_embeddings()* function) and the number of retrieved passages setting (in the *create_prompt()* function).

The chunk size corresponds to a “logical unit” in text, and the number is the token count. Overlap is repeating tokens in chunks (used to preserve context).

```

text_splitter = CharacterTextSplitter(chunk_size=500, chunk_overlap=50)
texts = text_splitter.split_documents(documents)

# Query relevant information
# You can try retrieving different number of chunks (n_results)
relevant_chunks = collection.query(
    query_texts=[question],
    n_results=3,
)

```

Review a Streamlit application with RAG using a document

In this section you will review a Streamlit application with RAG and observe the impact of file processing on the quality of LLM output.

1. In your Python IDE open the ***demo_streamlit_RAG.py*** script located in the *repo/Scripts* folder.
2. For testing, you will use two files from the *repo/Documents* folder (you don't need to copy them):
 - *Niagara-WQR-English-2022_053122-HR.pdf*
 - *Niagara-WQR-English-2022_053122-HR.txt*

Review the content of these files so that you can ask questions.

Note: this application is an example, and it doesn't have robust error handling. It may not work with complex PDF files that include graphics, tables, etc. Errors will happen on PDF file load. You can try using your own PDF files that contain primarily text. If you're an experienced Python programmer, you can also modify the code to load/read PDF files.

3. Run the *Streamlit* application by executing the command ***streamlit run demo_streamlit_RAG.py*** (annotated with red rectangle) in the terminal.



The screenshot shows a Python IDE interface with the following details:

- File Explorer:** Shows a folder structure with 'SCRIPTS' containing '_pycache__', '.env', 'chat_session.py', 'demo_streamlit_RAG.py', 'Generative_AI_Overview.ipynb', 'Niagara-WQR-English-2022_053122-HR.pdf', 'Niagara-WQR-English-2022_053122-HR.txt', 'sample_AI_Assistant.ipynb', 'state_of_the_union.txt', 'use_case_RAG.py', and 'watsonx_engine.py'.
- Terminal:** The command `PS C:\Users\admin\Downloads\2231_Lab_Files\Scripts> streamlit run ./demo_streamlit_RAG.py` is entered, with the entire command highlighted by a red rectangle.
- Status Bar:** Shows 'Scripts [Administrator]'.

*Note: you may need to install the following packages if they are not already installed in your Python virtual environment. You install these packages by executing ***pip install <package>***:*

- *fitz, frontend, pymupdf*

This application uses the Python module that you tested in the previous section of the lab, which means that it uses the same hardcoded model settings.

Most code in this application creates the UI and loads files. In this application the UI component of the application (*demo_streamlit_RAG.py*) writes PDF and txt files to the current directory on your laptop. The *use_case_RAG.py* module reads from this directory, loads data into *chromadb*, and invokes the LLM.

Demo of RAG with files and in-memory chromadb

Please upload one file type at a time and delete the file if switching types of files.

Upload a PDF File

 Drag and drop file here
Limit 200MB per file + PDF

Browse files

Upload a TXT File

 Drag and drop file here
Limit 200MB per file + TXT

Browse files

Question

Response

4. Load the *Niagara PDF* file to the UI and try asking the following questions:
 1. *How much CO₂ was saved?*
 2. *What is the name of the company mentioned in the document?*
 3. *How many bottles does the company produce per minute?*
 4. *What is spring water?*

Drag and drop file here
Limit 200MB per file • PDF

Niagara-WQR-English-2022_053122-HR.pdf 6.7MB

Upload a TXT File

Drag and drop file here
Limit 200MB per file • TXT

Question

How much CO2 was saved?

Answer the question

Response

Answer: 789 Million Pounds of CO2 in the last 10 years.

After asking each question, you can view the output in the Python terminal – it will show the prompts, which contain retrieved chunks.

```
Copyright © 2022 Niagara Bottling, LLC. All rights reserved. Each bottle of water begins with a carefully selected well or municipal water source. The water is initially processed using a multi-step filtration process depending on the source. Next, the water is purified using a reverse osmosis or distillation system and then it is disinfected with ozone just prior to bottling.  
Alkaline products are made using a similar process, however, there is an additional electrolysis step to increase the pH. Both alkaline and purified with minerals added products have one additional step, just after the water is disinfected with ozone, it is re-mineralized. We use just the right combination of minerals to give the product a smooth and refreshing taste.Distilled, Purified, Purified with Minerals Added and Alkaline Water  
Please answer a question using this text. If the question is unanswerable, say "unanswerable".How much CO2 was saved?***  
----- Generated response -----  
  
Answer: 789 Million Pounds of CO2 in the last 10 years.
```

You will notice that the model does not answer some questions when using the PDF file. One of the ways to troubleshoot it is to look at the retrieved passages from the PDF that are attached to the prompt (in the Python terminal).

For some questions, the retrieved chunks do not contain relevant information for the model to answer the question.

When troubleshooting this issue, we need to consider potential problems in several parts of the process:

1. Loading a PDF file
2. Chunking
3. Choice of embeddings API and vector database.

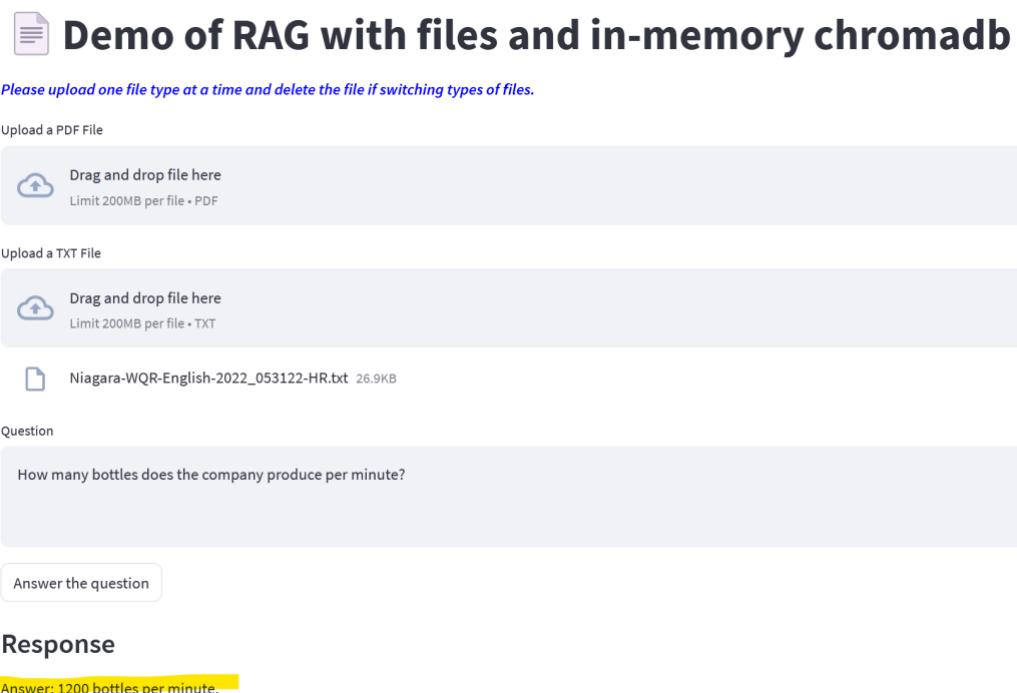
To rule out the issue with the processing of the PDF file, we will load a txt file with the same information.

5. Load the *Niagara txt file* to the UI and ask all the questions again:

1. *How much CO₂ was saved?*
2. *What is the name of the company mentioned in the document?*
3. *How many bottles does the company produce per minute?*
4. *What is spring water?*

Now we get results for the questions that were not answered when we used the PDF, which confirms the issue with loading and processing a PDF file.

In our sample application we used a basic PDF API. There are several open source and commercial applications for converting PDF to text, and this component of a generative AI application should be thoroughly tested to avoid issues that you've seen in our example.



The screenshot shows a user interface for demonstrating RAG (Retrieval-Augmented Generation) with in-memory storage. At the top, there's a title "Demo of RAG with files and in-memory chromadb" with a document icon. Below it, a note says "Please upload one file type at a time and delete the file if switching types of files." There are two main upload sections: "Upload a PDF File" and "Upload a TXT File". Each section has a "Drag and drop file here" button and a note "Limit 200MB per file - PDF/TXT". Under the PDF section, a file named "Niagara-WQR-English-2022_053122-HR.txt" is listed with a size of 26.9KB. Below these sections is a "Question" input field containing the text "How many bottles does the company produce per minute?". A "Answer the question" button is located below the question input. Finally, under the "Response" heading, the text "Answer: 1200 bottles per minute." is displayed in a yellow-highlighted box.

Conclusion

You have finished the *Chat with Documents* lab. In this lab you learned how to implement RAG use cases using the features of watsonx.ai.

Check [IBM documentation](#) for the latest code samples, tutorials, and other information related to RAG.