WEBEX and Generative AI

Generative AI and Webex integration - Omer Ilyas

Omer Ilyas

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Table of contents

1. Lab 3
1.1 Overview - Understanding AI and Its Integration with Webex 3
1.2 Pre-Requisites and Setup 10
1.3 AI/ML Revolution Unveiled 32
1.4 Tokenization 36
1.5 Task 4: Embeddings and Vector Database 54
1.6 Task 5: Context Windows and Retrieval Augmented Generation (RAG) 78
1.7 Multimodal RAG 96
1.8 GenAI- Frameworks 0
1.9 Understanding Fine-Tuning - Large Language Models 0
1.10 Fine Tuning - Deep Dive into Quantization , LoRA and SFT 0
1.11 Task 8a - Configuring and Fine Tuning - Using llama2 0
1.12 Task 8b - Configuring and Fine Tuning - Using llama3 0
1.13 Task 8c - Configuring and Fine Tuning - Using ChatGPT 0
1.14 Language Model Merging 0
1.15 Logging Out and Ending the Lab Session 0
1.16 Conclusion 0

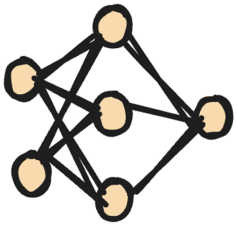
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1. Lab

1.1 Overview - Understanding AI and Its Integration with Webex

Technology and AI

Technology and AI are influencing every aspect of our lives, making the world around us more complex by the day



Artificial Intelligence (AI) is transforming the way we work, enabling innovative solutions and enhancing productivity. Cisco is committed to innovating responsibly, with responsible AI being non-negotiable. Our approach is grounded in the principles of transparency, fairness, accountability, reliability, security, and privacy. Cisco's Responsible AI Framework, aligned with the AI Risk Management Framework, ensures that all AI initiatives undergo rigorous assessments, particularly around privacy.

Remember data has value, Cisco's AI strategy emphasizes building ethical and trustworthy systems that mitigate risks while enhancing innovation across products and services.

IN THE AGE OF INFORMATION,
I DID NOT KNOW
IS NOT AN EXCUSE

Please refer to the below info for more detailed insights about Cisco Responsible AI:

- Responsible AI is built on a foundation of privacy
- Cisco's Responsible Artificial Intelligence Principles
- Cisco's Responsible Approach to Governing Artificial Intelligence
- Cisco's Responsible Artificial Intelligence Framework

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Responsible AI Framework

Cisco's goal is to provide clarity and consistency in informing users when AI is employed in our technologies

Cisco Trust Portal

trustportal.cisco.com

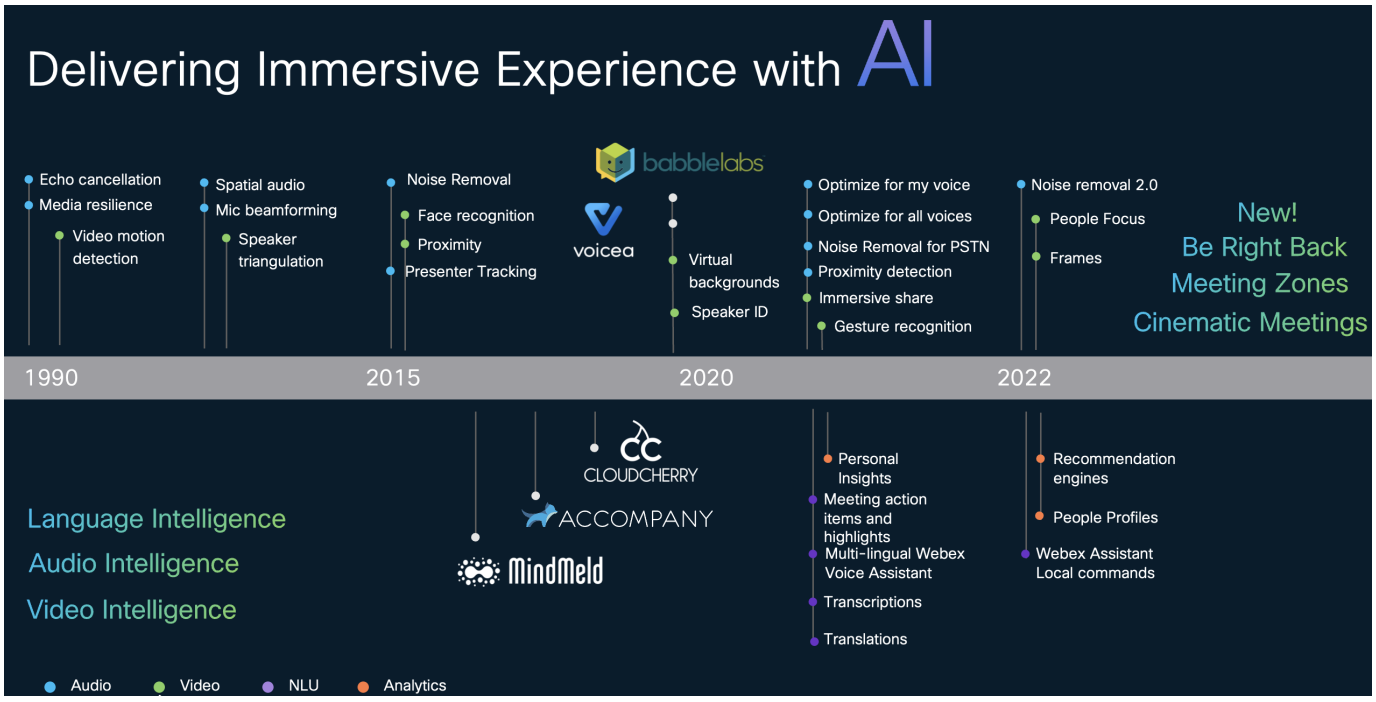
Your Data Belongs to You

Privacy, Security, and Human Rights by Design, Not by Chance

Trust Is Earned, Transparency Is Paramount

1.1.1 The Evolution of AI at Cisco

At Cisco, innovation is woven into the fabric of everything we do. We've been at the forefront of AI development long before it became a buzzword. Our journey spans decades, from early developments in audio and video intelligence, like echo cancellation, media resilience, to recent breakthroughs like noise removal, face recognition, and immersive experiences, just to name a few. We continue to push boundaries, enhancing AI across multiple domains to deliver immersive and intelligent experiences. Innovation in AI isn't just a trend for us—it's a longstanding commitment to excellence and progress.

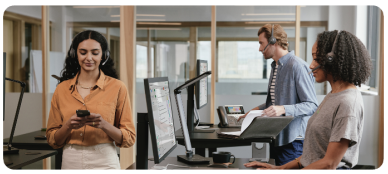


What we want everyone to understand is that AI is the core fabric that powers our platforms, enabling reimagining of work, workspaces, and customer experiences. Whether through the Webex Suite, advanced devices, or contact center solutions, AI drives the seamless, intelligent, and immersive experiences that define modern collaboration. Cisco's AI-powered Webex platform is designed to transform the way we work, ensuring that every interaction is smarter, more efficient, and more personalized.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

REIMAGINE WORK
WITH WEBEX SUITE

REIMAGINE WORKSPACES
WITH DEVICES



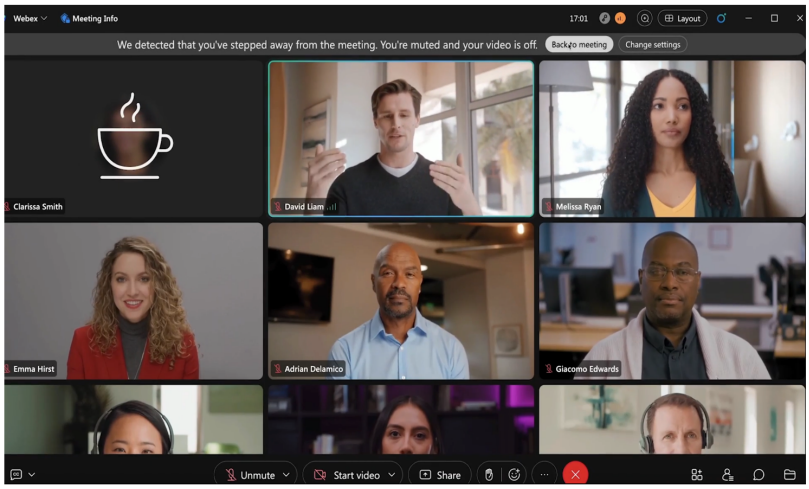
REIMAGINE CUSTOMER EXPERIENCE
WITH CONTACT CENTRE

AI-powered Webex Platform

In recent years, we've expanded our platform with several new AI capabilities. Here are a few highlights:

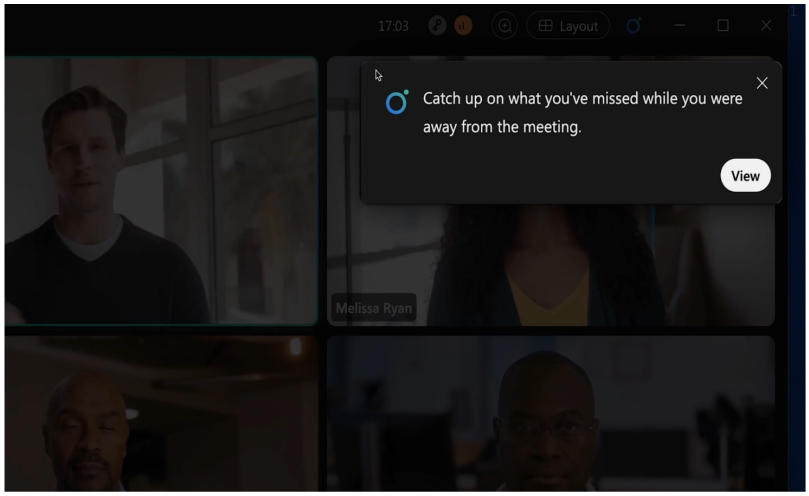
Catch me up in a meeting

Late to the meeting? Stepped away from the meeting? You can quickly catch up on what you missed.



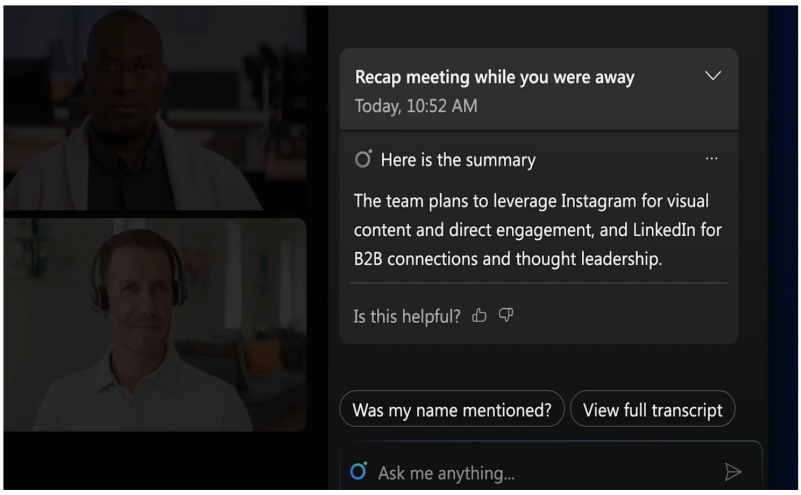
Catch me up in a meeting

Late to the meeting? Stepped away from the meeting? You can quickly catch up on what you missed.

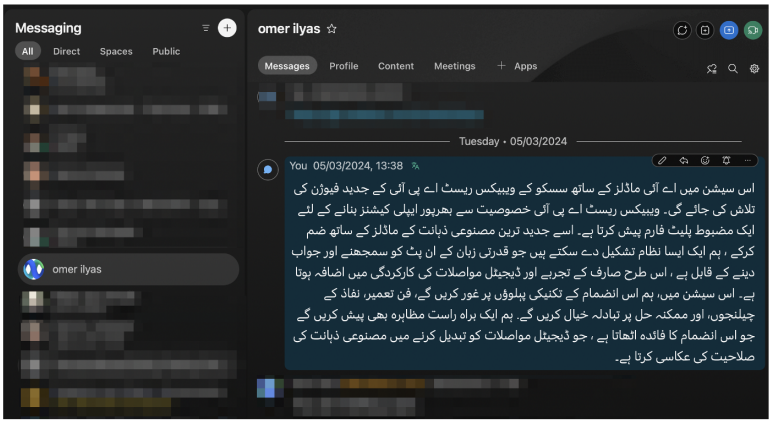


Catch me up in a meeting

Late to the meeting? Stepped away from the meeting? You can quickly catch up on what you missed.



Translate Messages with Ease



Cinematic Meetings

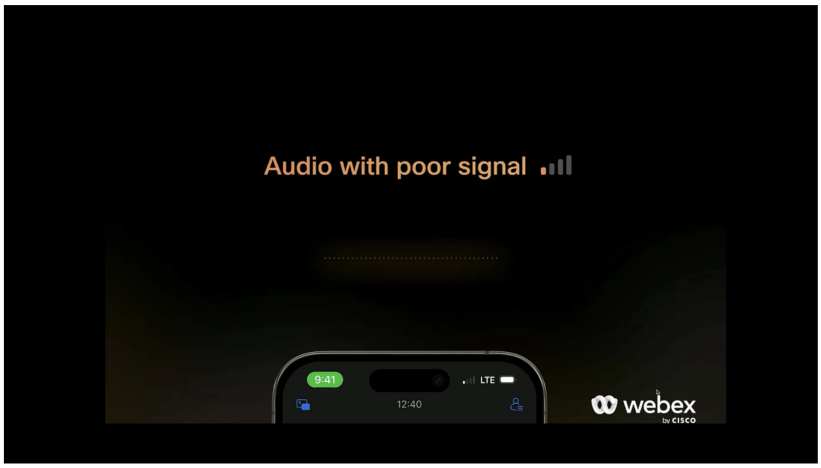
Keep everyone engaged and show the best view of people in the room from different angles through adaptive AI directed framing

AI Codec Improve audio quality in poor network conditions

Webex AI Codec

Be heard and seen clearly

New AI codec and Super resolution transform audio and video even in the poorest network connection



Webex AI Codec

Compatible Devices + Webex App (44.8) + Phase1 (App to App only)	Macbook
	Windows
	iPhone 11+
	Android
	iPad 13+

Min Version

Be heard and seen clearly

New AI codec and Super resolution transform audio and video even in the poorest network connection

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.1.2 Summary of newly integrated AI features within our platform

Reimagine work and workspaces

- Optimize for voice
- Smart Chapters
- Background Noise Removal
- AI Codec
- HD Voice
- Be right back
- Super Resolutions
- Meeting Summaries
- Space Summaries
- Rewrite/Translate messages
- Multi Stream
- Smart Relighting
- People Recognition in Rooms
- Slido Topic Summaries
- Video Highlights

Reimagine Customer Experience

- New Agent Desktop
- New Supervisor Desktop
- Topic Analysis
- Intelligent Summary
- Suggested Responses
- Coaching Highlights
- Agent Burnout Detection

REMINDER: Our product team is hosting AI sessions. Please review the agenda and join the ones that interest you.

1.1.3 Lab Guide Overview

In this lab guide you will go through the fundamental concepts of AI, various techniques, and how AI can be integrated with Webex to create efficient workflows.

1.1.4 Upon completion of this lab you will be able to

- Understand the basics of AI and its applications
- Learn about embedding techniques
- Vector databases
- Gain insights into Generative AI models
- Familiarize yourself with different AI frameworks
- Integrate AI with Webex to create seamless workflows - Call Recording
- Develop hands-on skills through practical exercises
- Understand Fine-tuning and Quantization
- Deploy Fine Tuning techniques
- Retrieval Augmented Generation (RAG) and Multimodal RAG

1.1.5 Prerequisites

- Basic understanding of AI concepts is helpful but not required.
- We will be using Google Colab for this lab. However, if you have your own machine with an Nvidia chipset and prefer to run the lab locally, feel free to do so.

1.1.6 Disclaimer

The lab design and configuration examples provided are for reference/learning purposes only. This is a sample deployment, and not all recommended features are used or enabled optimally. For design-related questions, please contact your representative at Cisco or a TME team.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.1.7 Lab Overview - Generative AI and Webex integration

- Lab Setup - Google Colab and Hugging Face
- AI/ML Revolution Unveiled: Task2.md
- Tokenization:
- Embeddings and Vector Database:
- Context Windows and Retrieval Augmented Generation (RAG)
- Multimodal RAG
- Understanding Fine-Tuning for Large Language Models
- Fine Tuning - Deep Dive into Quantization , LoRA and SFT
- Configuring and Fine Tuning - Using llama2
- Configuring and Fine Tuning - Using llama3
- Configuring and Fine Tuning - Using ChatGPT
- Model Merging (Theory Only)

Let's get started! Click on Task 1 - Google Collab- Accessing Google Collab and creating account.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.2 Pre-Requisites and Setup

1.2.1 Google Collab- Accessing Google Collab and creating account

Google Colab is a free, cloud-based platform that provides a convenient environment for running notebooks. If you want to create a machine learning model but don't have a computer that can handle the workload, Google Colab is the platform for you. In our lab, we will be using Google Colab to test and run our code. However, if you have your own Python environment and prefer to run the code on your local machine, please feel free to do so.

Here are some reasons why using Google Colab can be beneficial for this lab:

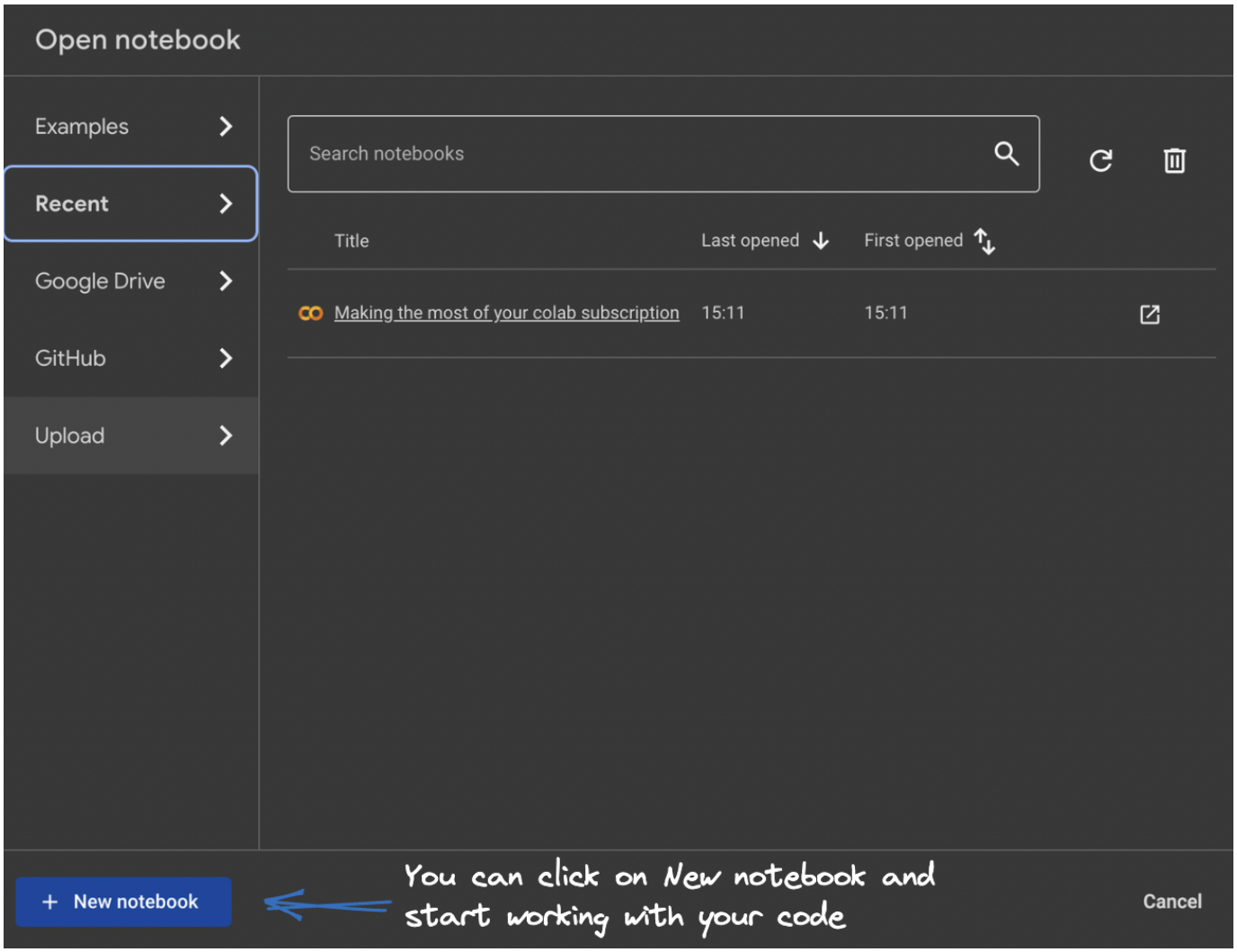
- Free Access to GPUs and TPUs: Google Colab offers free access to powerful GPUs and TPUs, which can significantly accelerate the training and fine-tuning of machine learning models.
- No Setup Required: With Colab, there is no need to set up your local environment. Everything runs in the cloud, which saves time and avoids configuration issues.
- Easy Collaboration: Colab notebooks can be easily shared and collaborated on with team members, making it an ideal tool for collaborative projects.
- Integration with Google Drive: Colab integrates seamlessly with Google Drive, allowing you to save and manage your work conveniently.
- Pre-installed Libraries: Many popular machine learning libraries, including TensorFlow and PyTorch, come pre-installed in Colab, making it easy to start working on your projects immediately.

1.2.2 Getting Started With Google Colab

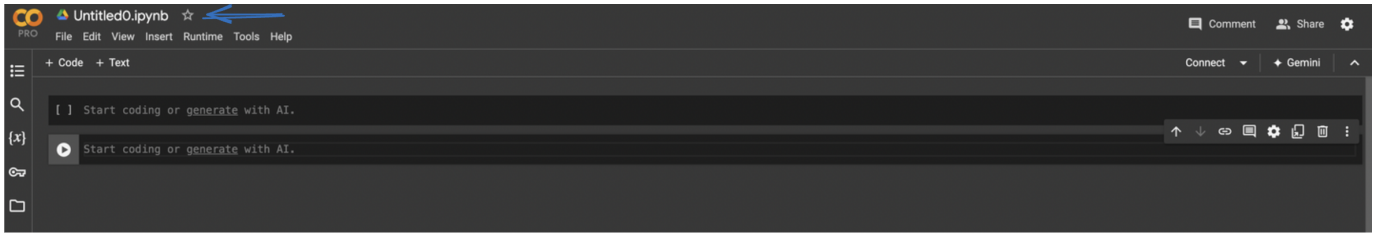
To start working with Google Collaboratory Notebook you first need to log in to your Google account, then go to this link Google Colab

- Create a new Jupyter Notebook

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- On creating a new notebook, it will create a Jupyter notebook with Untitled0.ipynb and save it to your google drive in a folder named Colab Notebooks. Now as it is essentially a Jupyter Notebook, all commands of Jupyter Notebooks will work here.

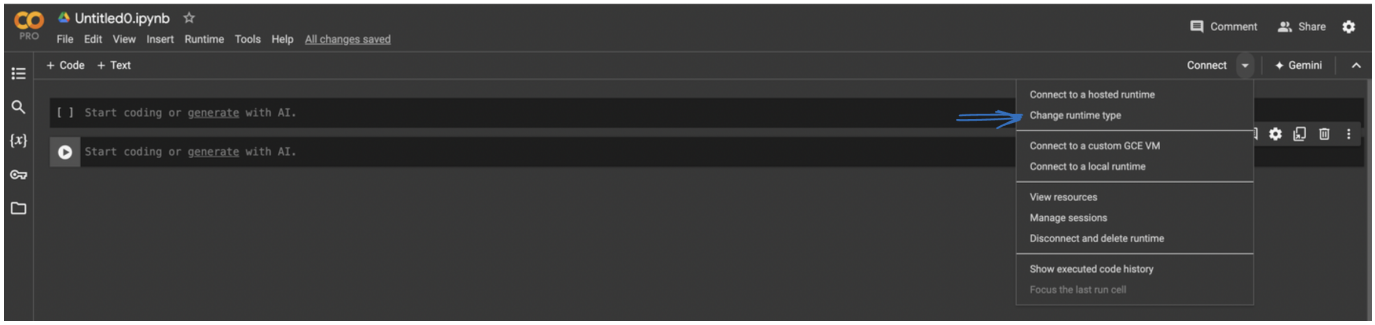


- There might be times when we need to fine-tune models or perform specific tasks that require changing the runtime environment in Colab. Google Colab offers different runtime environments that can be selected based on your requirements:
- Python Versions: You can select between different versions of Python (e.g., Python 2 or Python 3)depending on the compatibility of the code and libraries. We will be using Python3 for our lab.
- Hardware Accelerators: Colab provides access to hardware accelerators, which can be particularly useful for intensive computations. You can choose between:

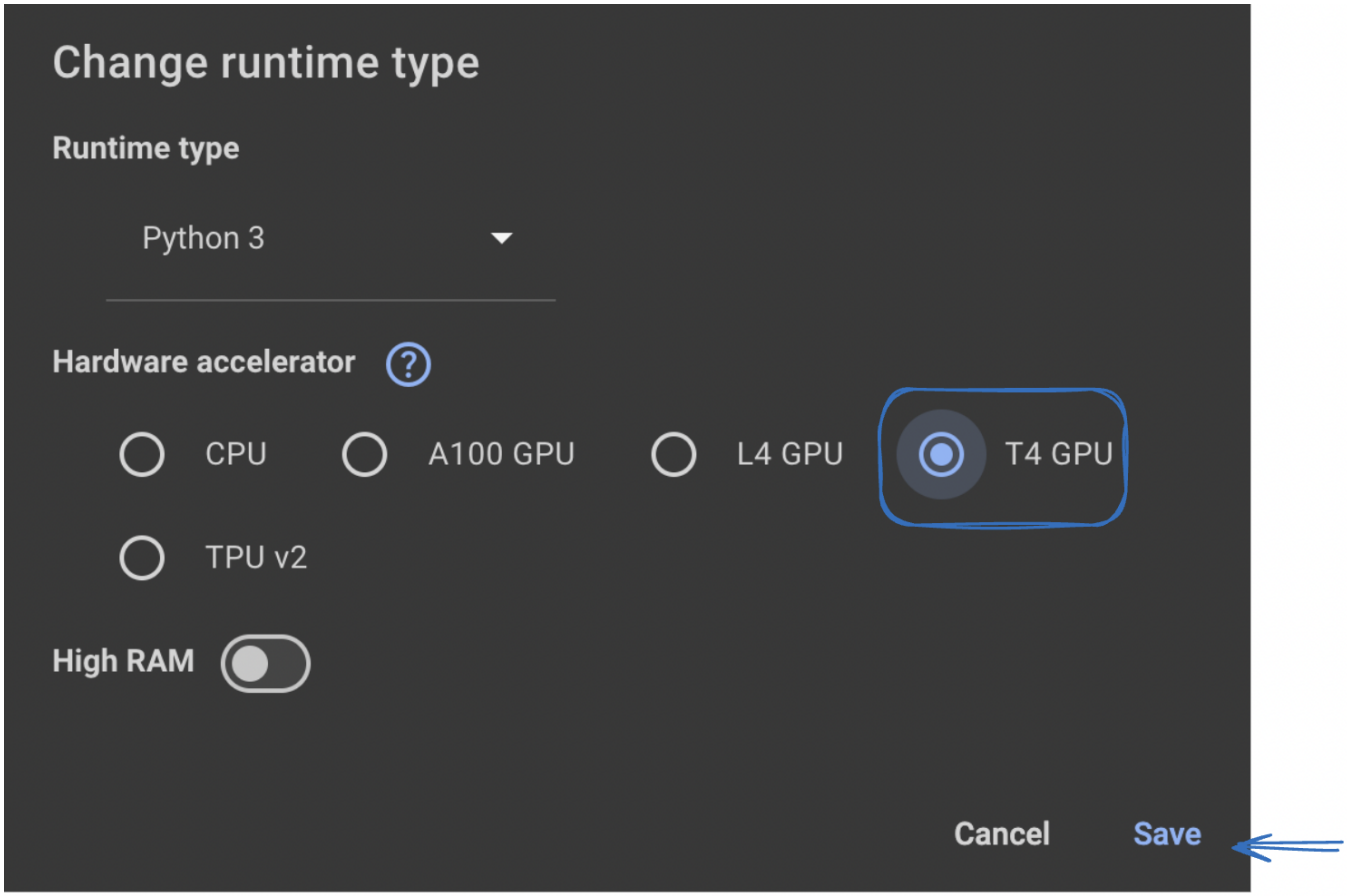
```
None: No hardware acceleration, suitable for basic tasks.

GPU: Accelerate your computations with a Graphics Processing Unit.

TPU: Use a Tensor Processing Unit for even faster performance, especially beneficial for deep learning tasks.
```

- Change Runtime Environment: Click the “Runtime” dropdown menu at the top of the Colab interface.

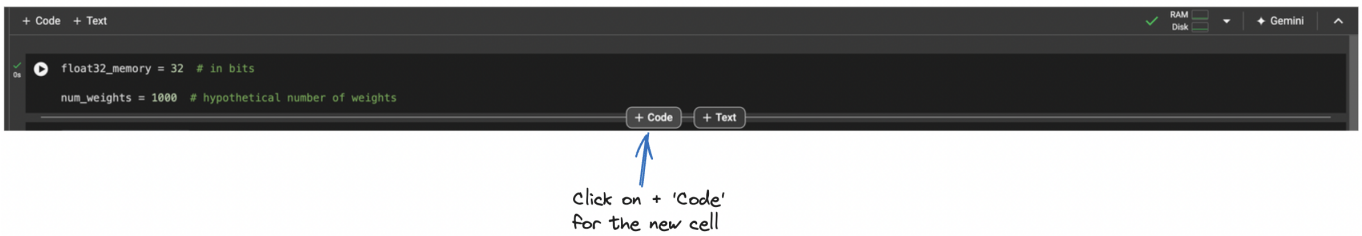
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Select “Change runtime type”: This will open a dialog box where you can configure the runtime environment.
- Select Python Version: Choose Python 3 from the “Runtime type” dropdown menu.
- Select Hardware Accelerator: From the “Hardware accelerator” dropdown menu, choose GPU, or TPU based on your needs.

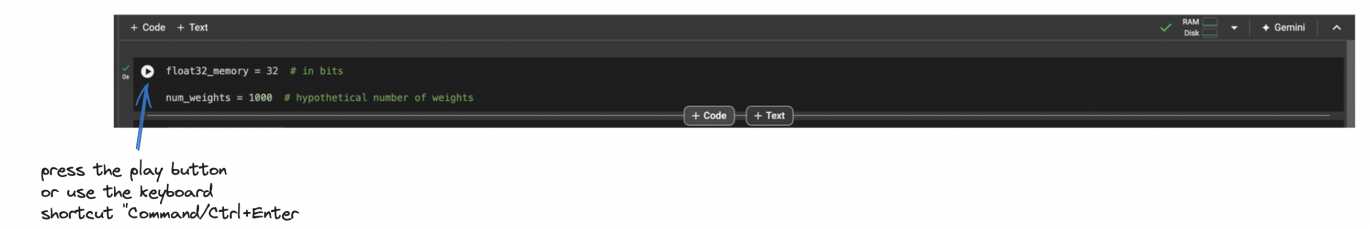


- Save Settings: Click “Save” to apply the changes.
- New Cell: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



- Execute Code: Click the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter" while the cell is selected.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



1.2.3 Notes: On GPU and TPU Access:

While Google Colab offers free access to GPUs and TPUs, there are limitations. For more consistent access to high-performance GPUs and TPUs, you might need to subscribe to Colab Pro or Colab Pro+ accounts. These paid plans provide priority access to better hardware, longer runtimes, and more memory.

1.2.4 Using Huggingface Hub to share our Datasets

In this lab, we will be utilizing the Hugging Face Hub to load our custom datasets. Hugging Face provides an extensive repository of datasets that can be easily integrated into your machine learning workflows. For the purposes of this lab, we will demonstrate how to access/upload and use our custom datasets effectively.

However, when fine-tuning models in your own work environment, especially if you are using private data, there are important considerations to keep in mind:

- Private Datastores: If you are working with proprietary or sensitive data, it is crucial to use your organization's secure datastores. Ensure that all data handling complies with your organization's data privacy policies and regulations.
- Hugging Face Datasets: If you prefer to use Hugging Face for dataset storage and management, make sure to mark your datasets as private. This setting ensures that your data cannot be accessed by anyone outside your organization, maintaining the confidentiality and integrity of your information. Please refer to Huggingface documentation for more info.

Few more Condsideration

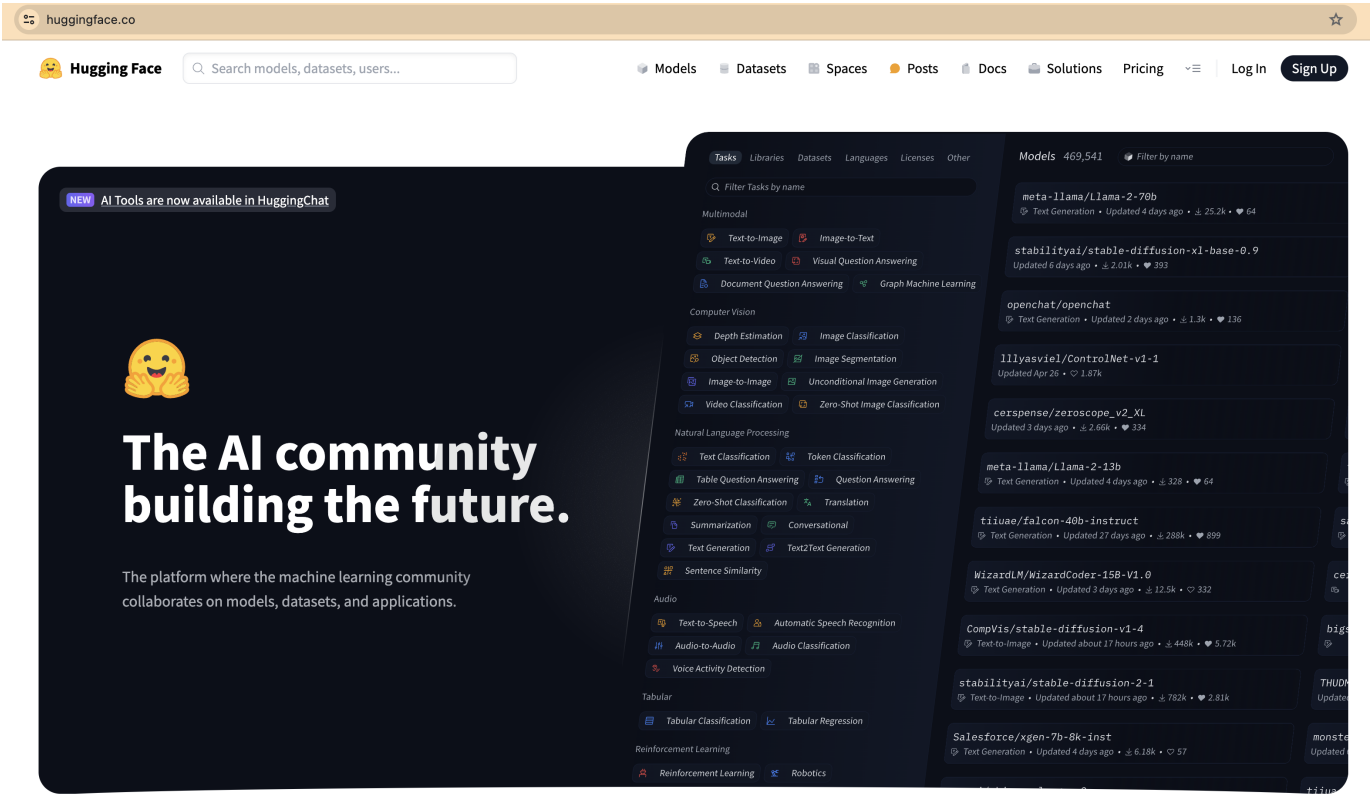
- Upload Dataset: When uploading your dataset to Hugging Face, choose the appropriate privacy settings. You can set your dataset to private during the upload process.
- Check Permissions: Regularly review and manage the permissions of your datasets to ensure they remain private and secure.
- Collaborator Access: If you need to share the dataset with specific team members, use the Hugging Face interface to grant access to trusted collaborators only.

By following these guidelines, you can ensure that your data remains secure while leveraging the powerful tools and resources provided by Hugging Face. This approach not only enhances your workflow efficiency but also upholds the best practices in data security and privacy.

1.2.5 Accessing Hugging Face Hub and creating account

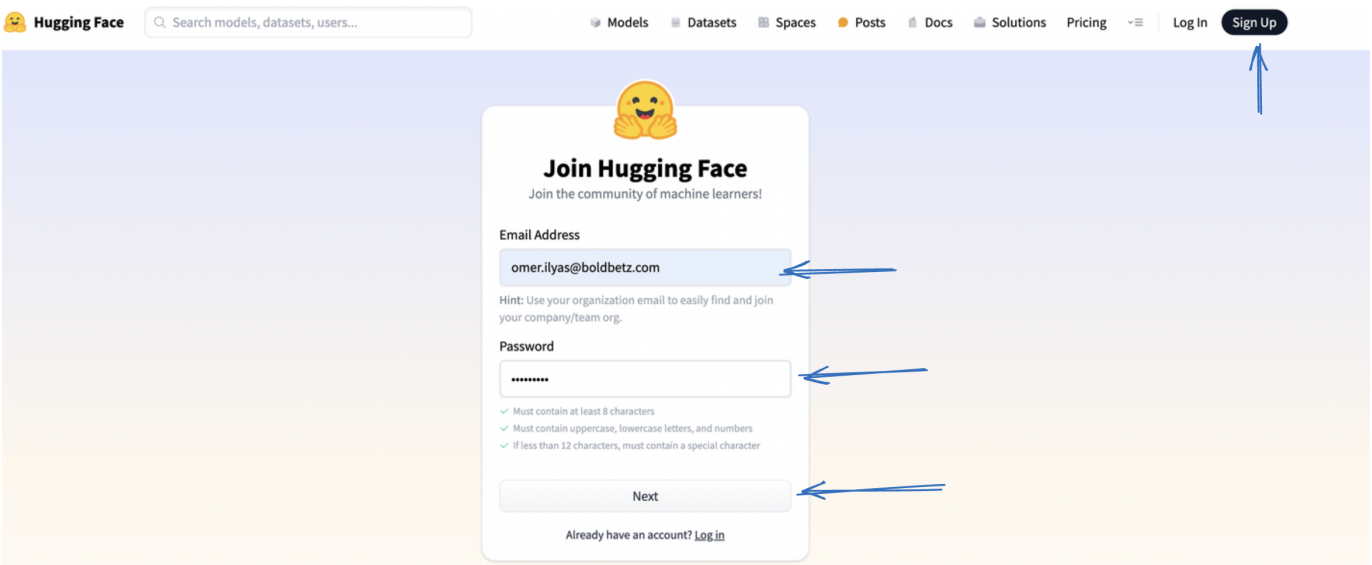
Hugging Face can be accessed by browsing to huggingface

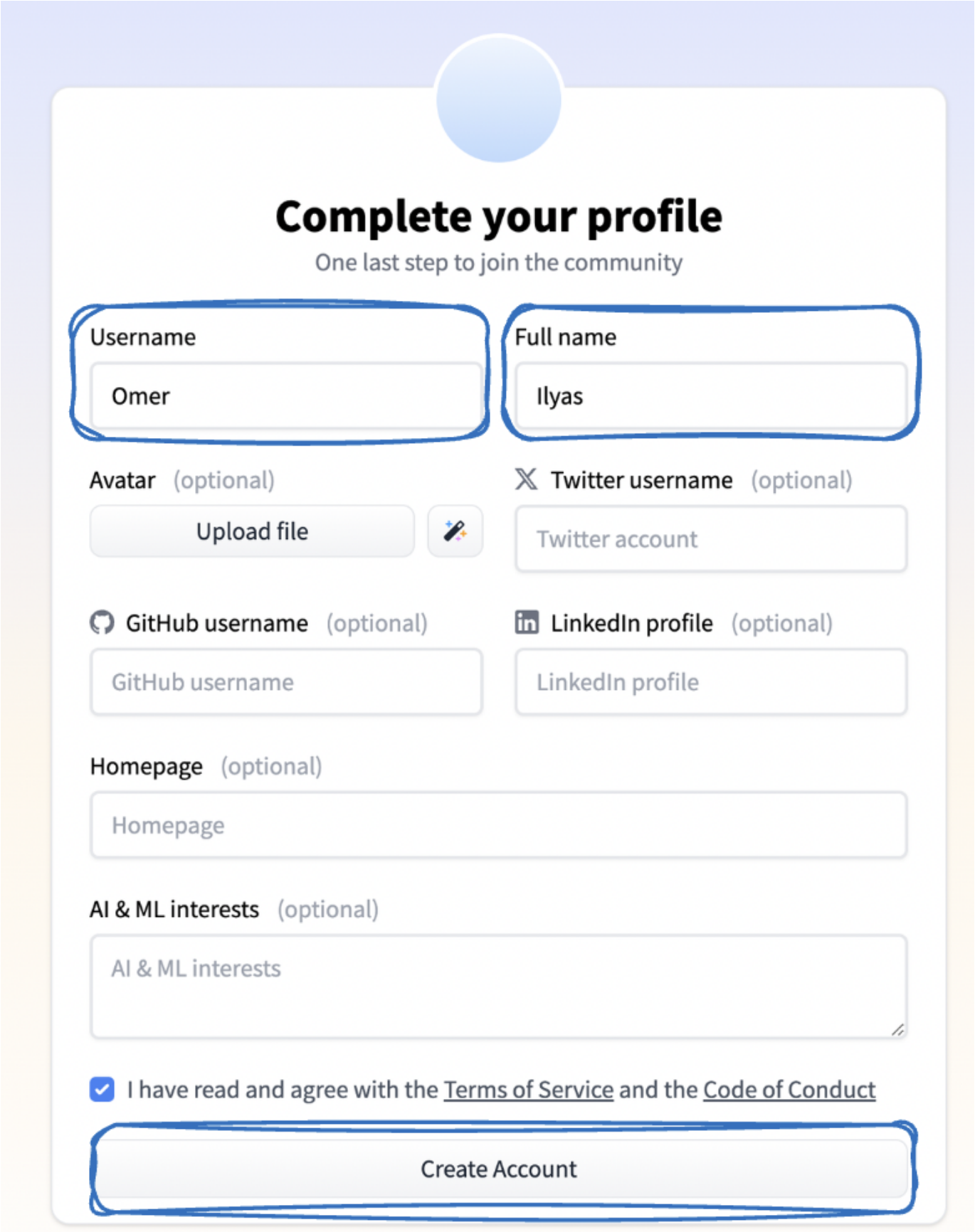
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



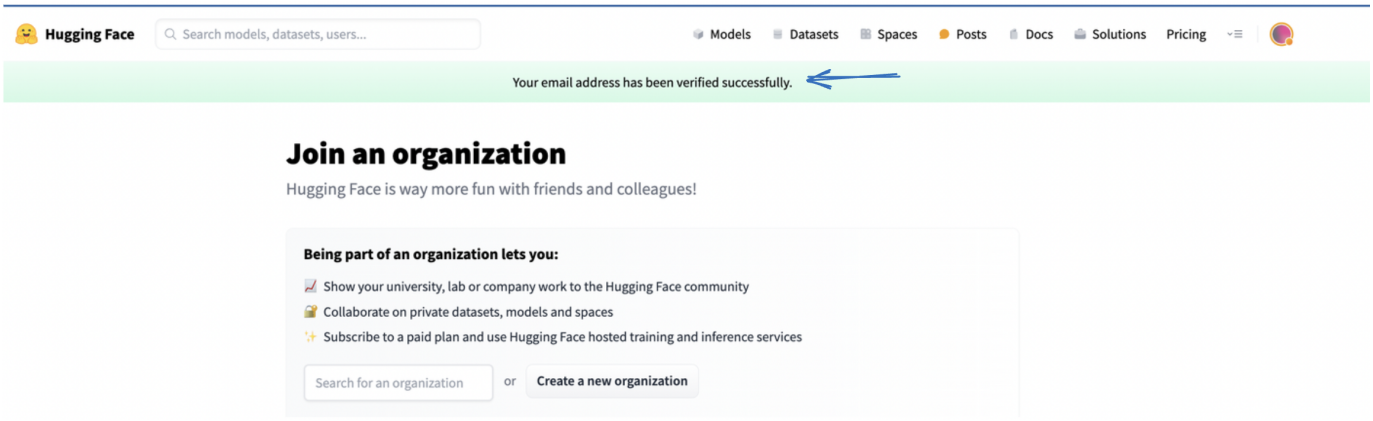
Signing up

- Browse to Hugging Face home page and click on Sign up. Follow the instructions as per below images



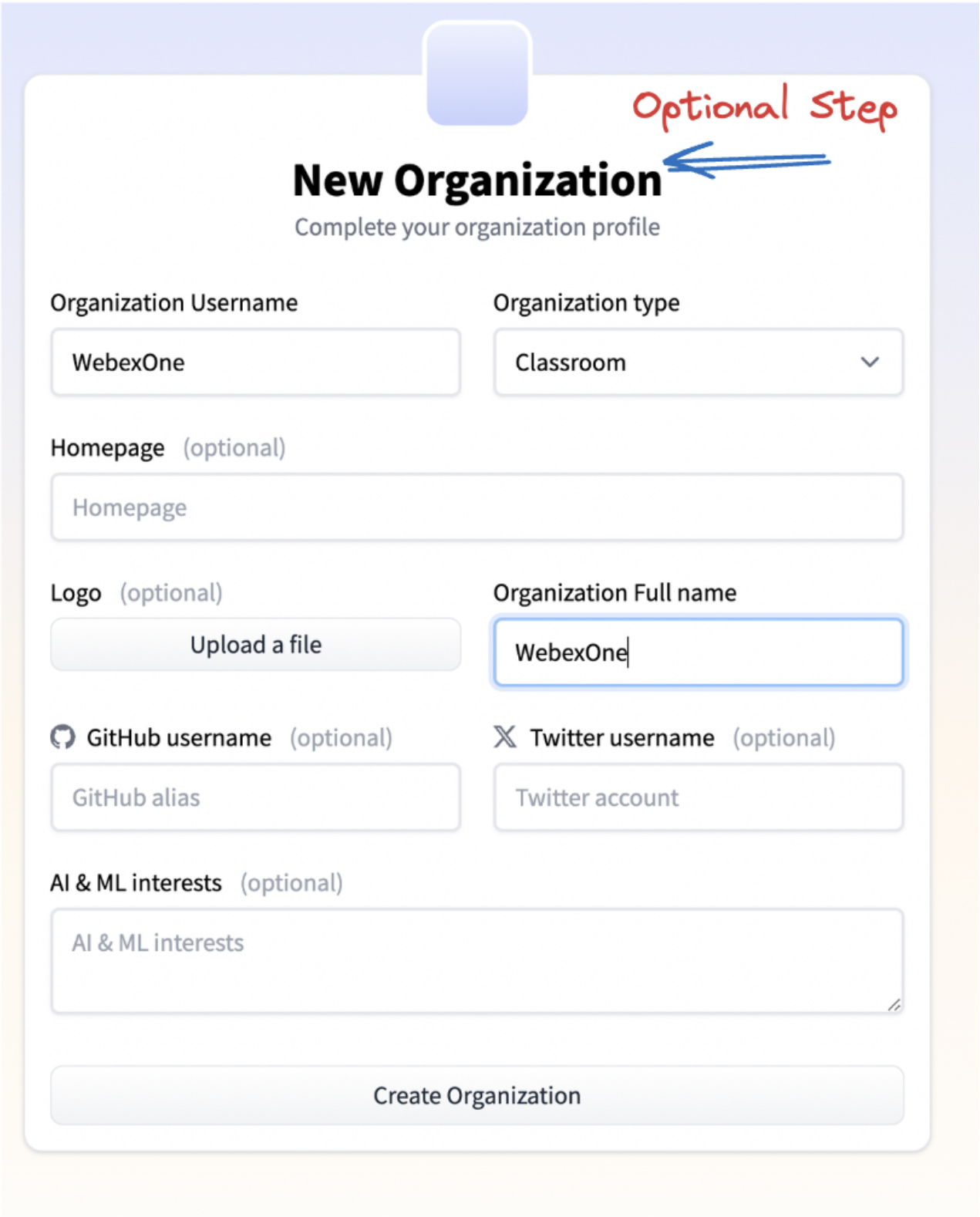


- Please check your email address for a confirmation link and click to verify your account



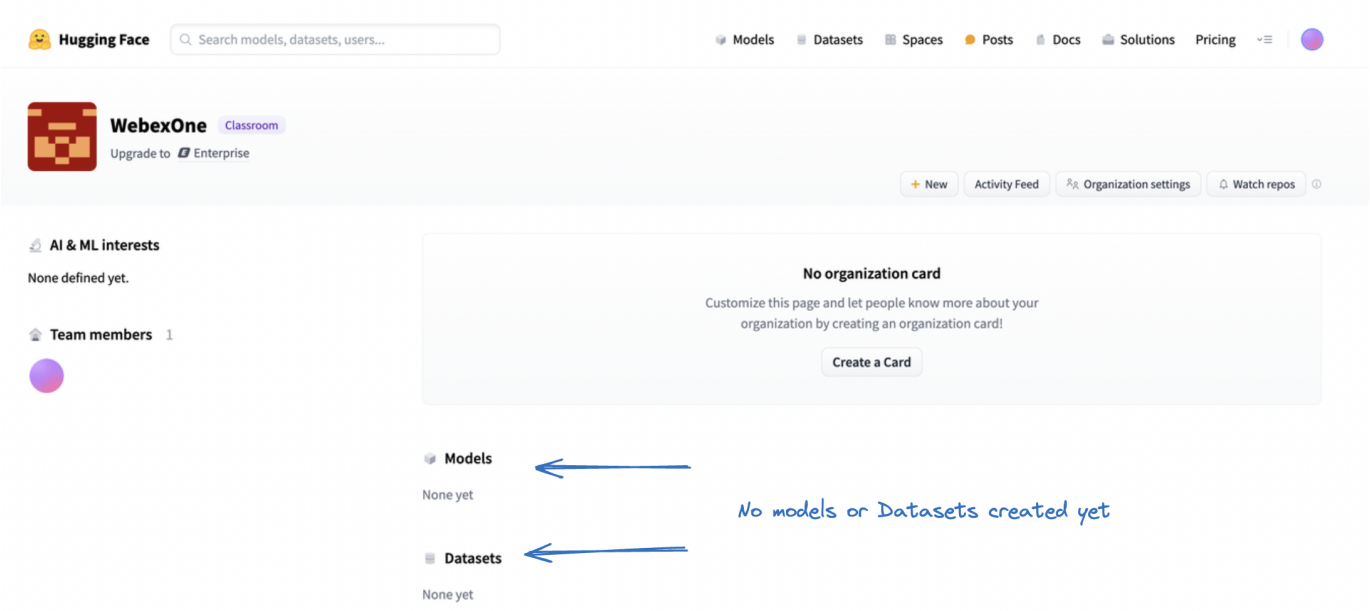
- Organization Creation (Optional): While you can upload datasets and fine-tune models directly on Hugging Face without creating an organization, you have the option to create an organization on Hugging Face. This can be particularly useful for team collaboration, as it allows you to upload all your datasets and models in one centralized location.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

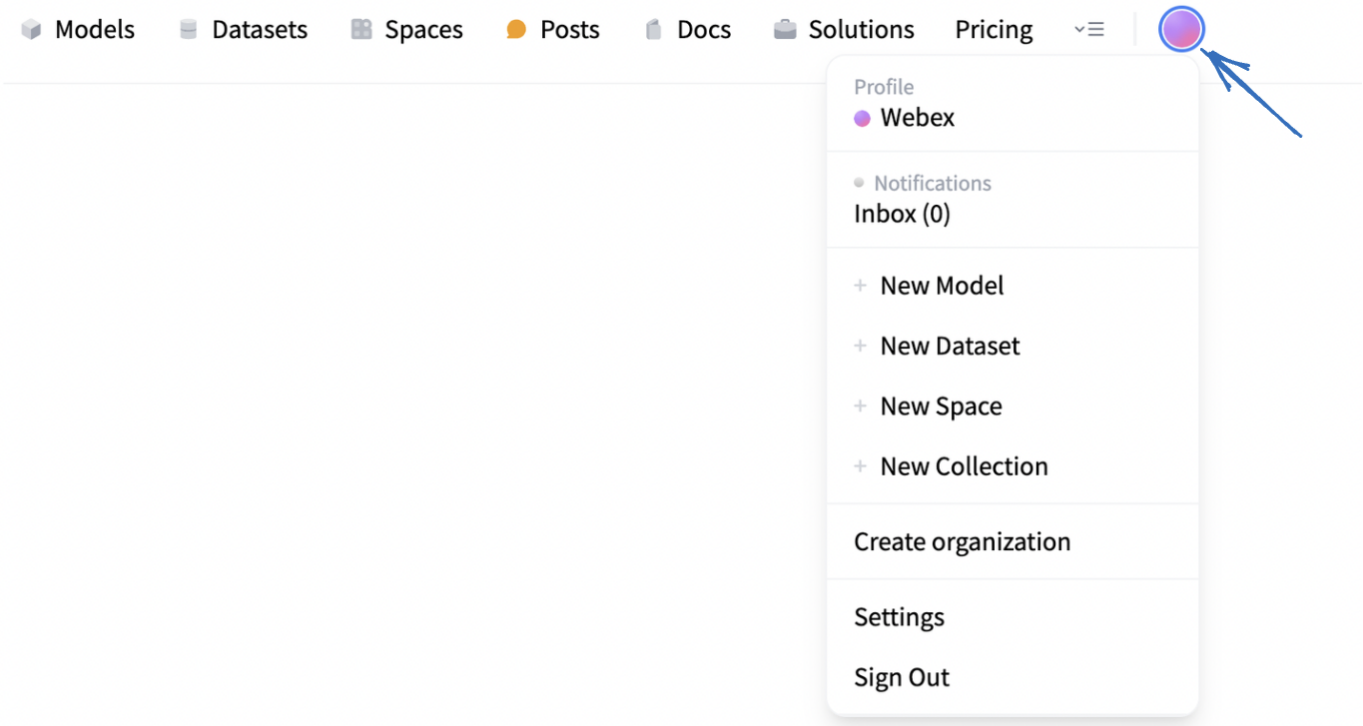


- At this stage, you will see no models or datasets created under your account.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Access Your Models and Datasets: The same can be accessed by clicking your profile picture on the top right corner of the Hugging Face website. This will take you to your personal dashboard where you can view and manage your models and datasets.



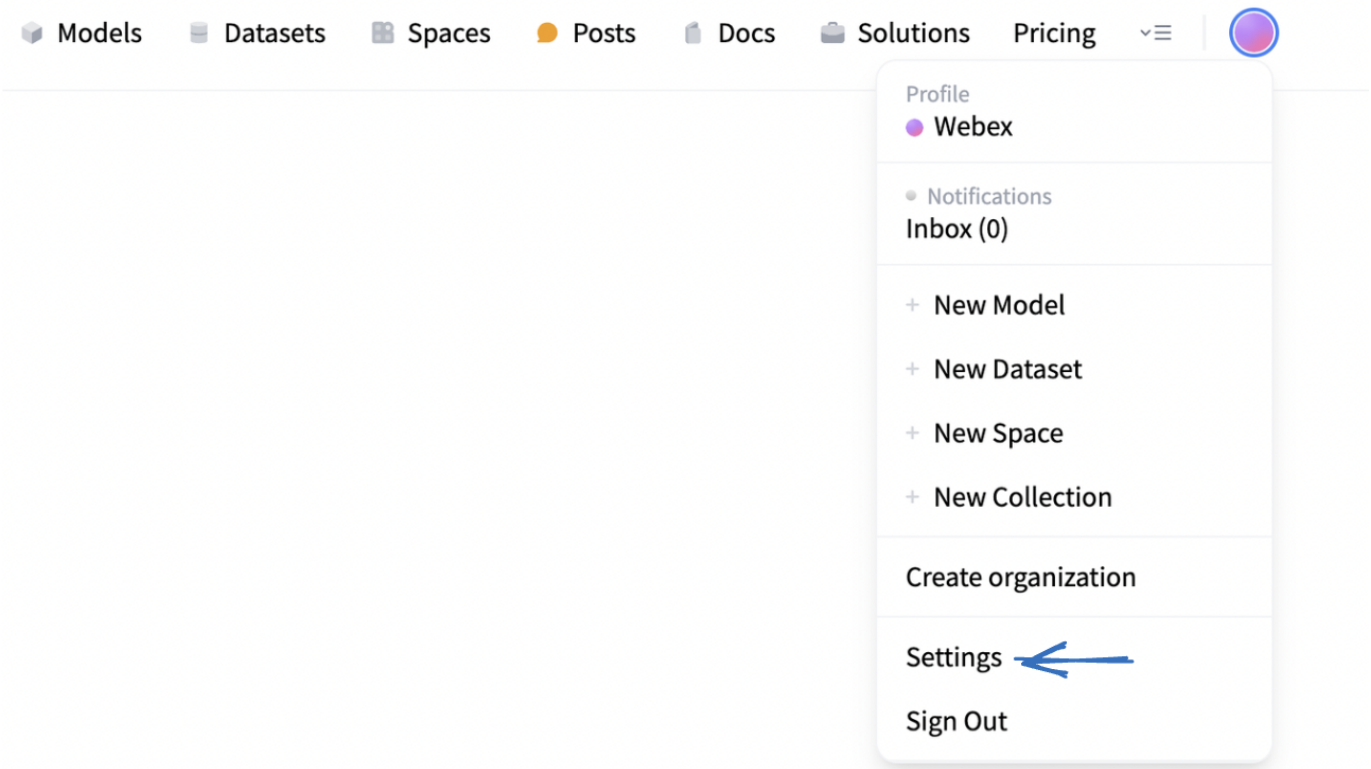
Hugging Face API Keys

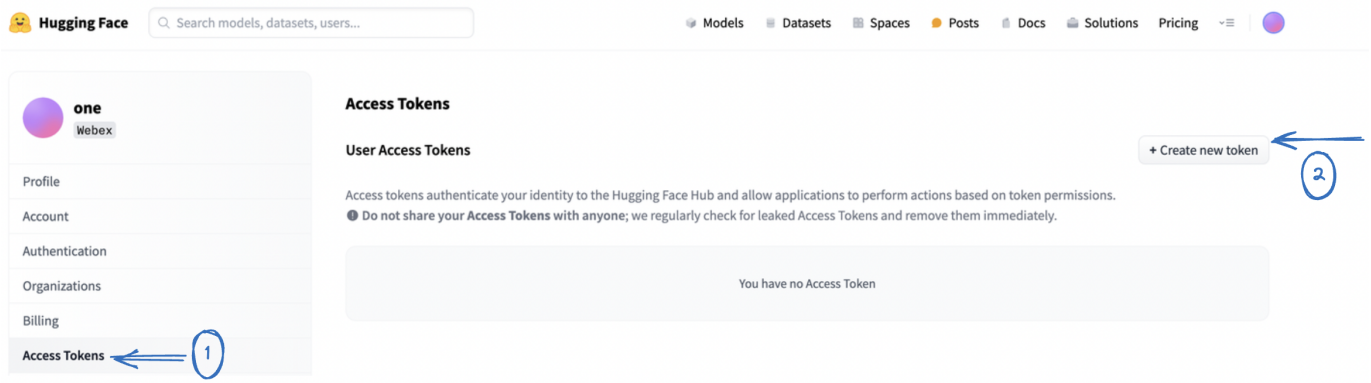
Create an API Key: As we will be uploading our datasets to the Hugging Face Hub, we need to create an API key for our account. This API key will be used to authenticate and interact with the Hugging Face services programmatically.

you can browse to huggingface API Key

or

Click on your profile picture > Settings > Access Tokens



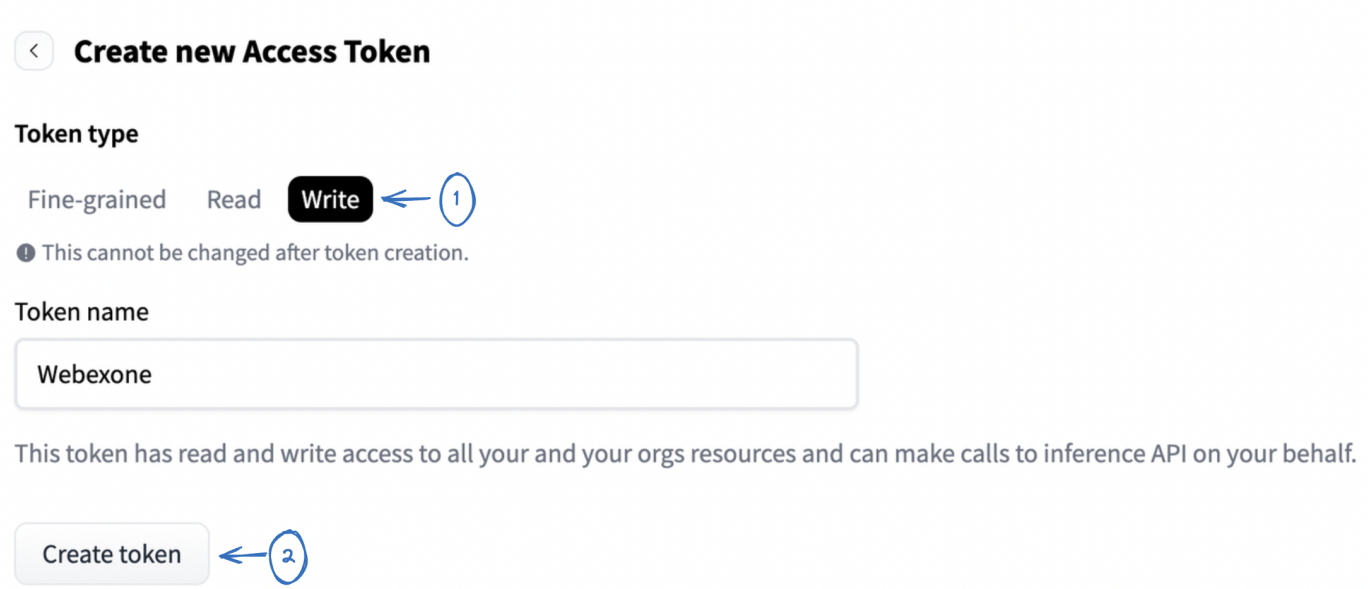


Under the "Access Tokens" section, click on "Create new token." You will see options to select the token type and provide a token name. For example, you might name your token "Webexone" and select the appropriate permissions.

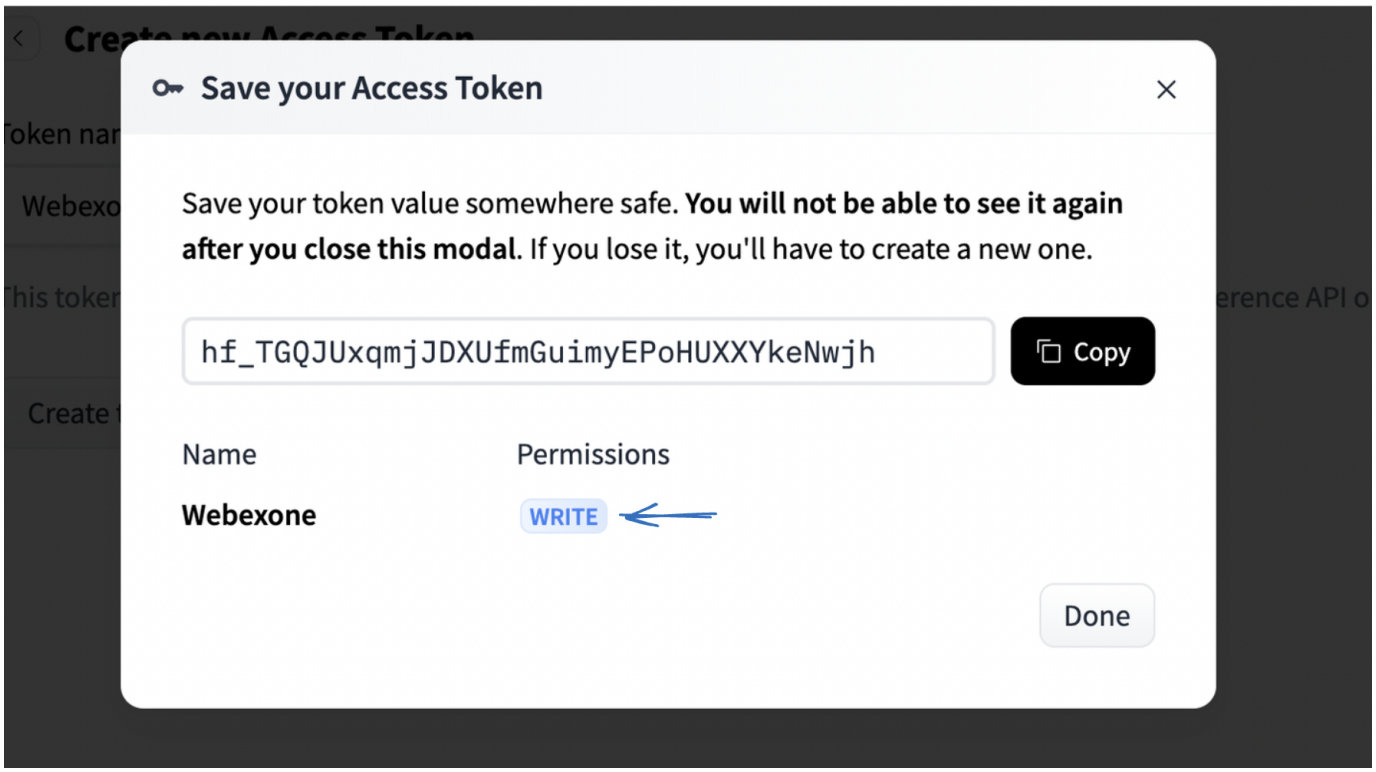
- fine-grained: tokens with this role can be used to provide fine-grained access to specific resources, such as a specific model or models in a specific organization. This type of token is useful in production environments, as you can use your own token without sharing access to all your resources.
- read: tokens with this role can only be used to provide read access to repositories you could read. That includes public and private repositories that you, or an organization you're a member of, own. Use this role if you only need to read content from the Hugging Face Hub (e.g. when downloading private models or doing inference).
- write: tokens with this role additionally grant write access to the repositories you have write access to. Use this token if you need to create or push content to a repository (e.g., when training a model or modifying a model card).

As we have a lab envoirnment we will be using the "Write" permission. This token will have read and write access to all your resources and can make calls to inference API on your behalf, as shown in the image below.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



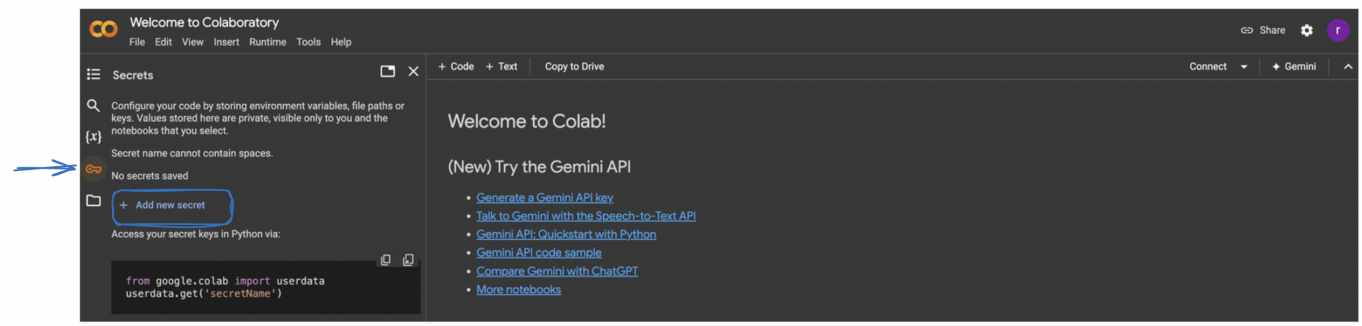
Save and Secure the Token: Once the token is generated, save it securely. This token will be required for accessing and managing your datasets via the API. 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



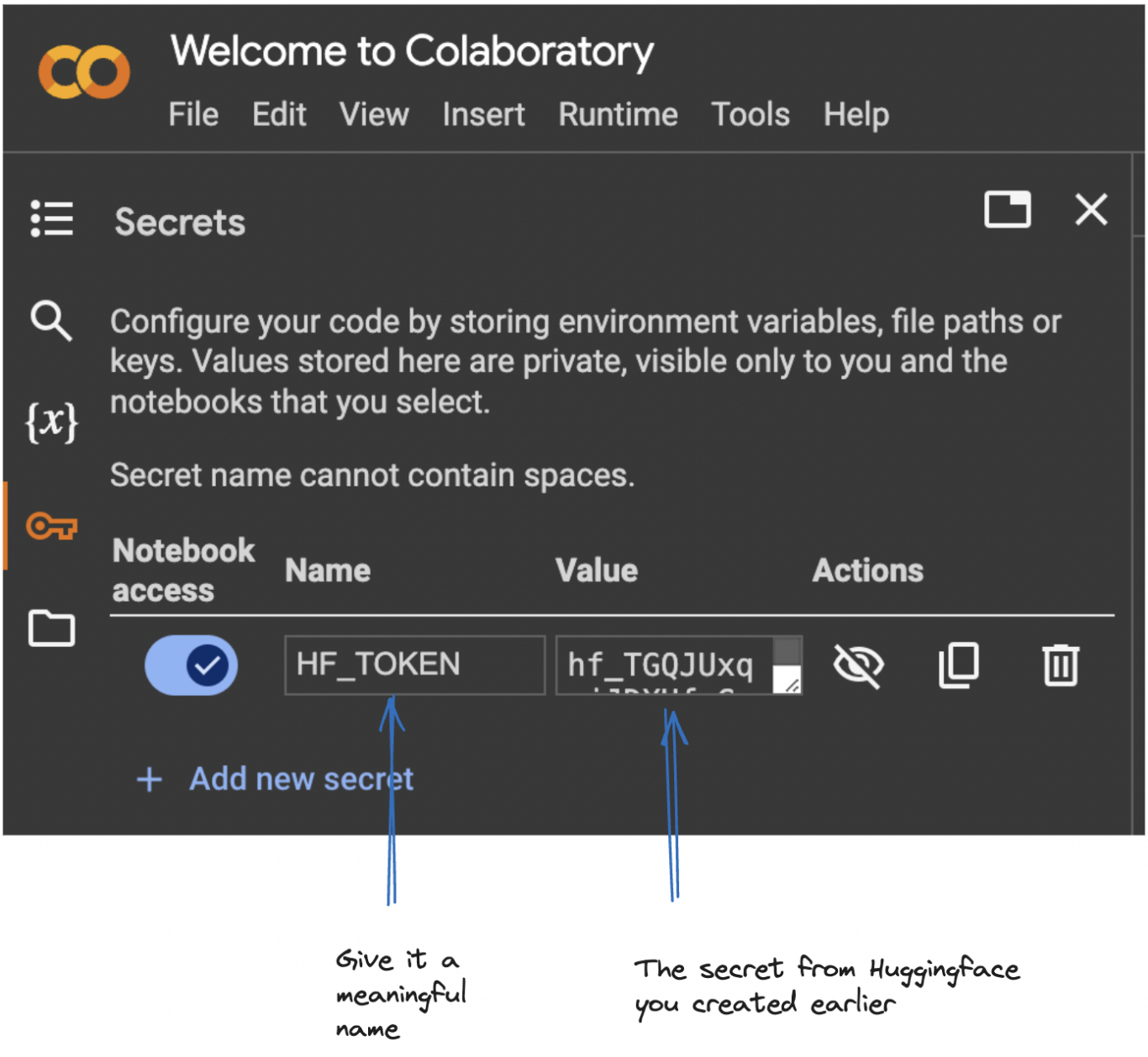
1.2.6 Accessing Hugging Face API in Google Colab

- Open the Google Colab notebook and navigate to the new “Secrets” section in the sidebar.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Click on “Add a new secret.” Enter the name example: HF_TOKEN and value of the secret. Note: The name is permanent once set.
- The list of secrets is global across all your notebooks.
- Use the “Notebook access” toggle to grant or revoke access to a secret for each notebook.



Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Optional Steps below

Incorporating Secrets into Your Code - We will use it later in our lab

- To use a secret in your notebook, use the following code snippet

```
from google.colab import userdata
my_secret_key = userdata.get('HF_TOKEN')
```

- Replace with your secret's name.

Using Secrets as Environment Variables - Optional Step , we will use it later in our lab

- For Python modules requiring API keys as environment variables, use the below snippet:

```
# Import Colab Secrets userdata module

from google.colab import userdata
import os

# Set other API keys similarly
os.environ["HF_TOKEN"] = userdata.get('HF_TOKEN')
```

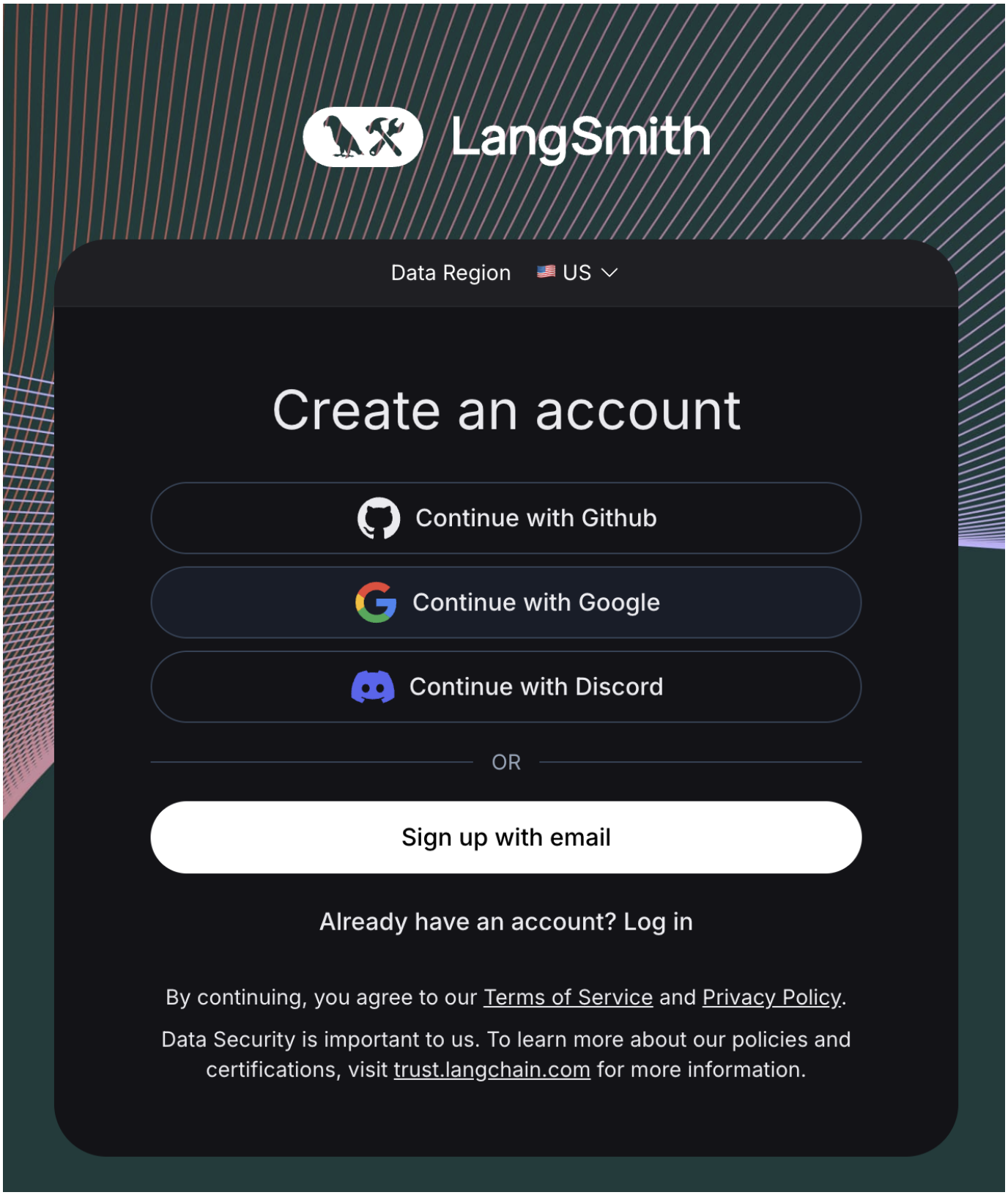
1.2.7 Getting Started with Langchain

Let's explore how to create a LangChain account and obtain the API key, which we will use later in our lab.

To start building applications with LangChain, you'll need an API key. This key allows your applications to securely connect with LangChain's services, ensuring proper authentication and usage tracking.

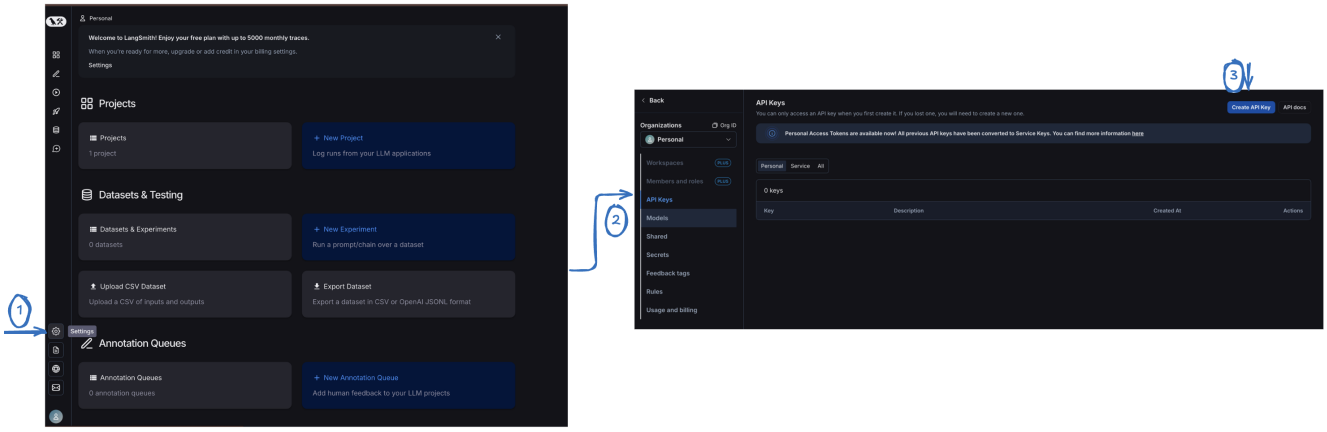
Lets start by visiting the official LangChain website and create an account. You'll need to enter some basic information about yourself or your organization. In this example, I will be using my Google account to sign up.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



To create an API key head to the Settings page. Then click Create API Key.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Setup Langchain envoirnment - Sample Code

- In the code snippet below, we'll configure our environment to enable tracing of AI calls and set up the API key for interacting with LangChain's services, as we will be using these in the upcoming section.

```
!pip install langchain-community
export LANGCHAIN_TRACING_V2=true
export LANGCHAIN_API_KEY=<your-api-key>
```

1.2.8 Optional Step - Running Ollama locally

Have you ever used GPT and amazed at its ability to understand and respond to your queries? But did you know that you can also harness the power of open-source models using Ollama? With Ollama, you can download and interact with these models directly, getting responses that are tailored to your needs.

In this lab, I'll show you how to install Ollama locally on your machine and start using it to generate responses. You'll learn how to download and load open-source models, and then use Ollama's intuitive interface to interact with them. More info for Ollama can be found here

Note: The steps below are provided for informational purposes. If you are using the demo laptop in this lab, feel free to follow these instructions. However, if you are using your personal or work machine, please ensure that you have the necessary privileges and authorization from your organization's administrator to install Ollama locally.

Prerequisites for Installing Ollama Locally

- Ensure that Docker is installed and running on your machine. It is available for various operating systems, including macOS, Windows, and Linux. You can download it from the official Docker website and follow the installation instructions for your specific OS.

Installing Ollama Locally

Note: Since we're using a Mac, I'll demonstrate how you can install it on macOS.

- First, you need to install Ollama. You can download it here and clicking on the download button. Follow the instructions.
- Once installed, Ollama functions as a command-line application, allowing you to interact with it directly through the terminal. To get started, open the terminal and enter the following command:

SHELL

```
ollama
```

It will output the below commands

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

```
Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve       Start ollama
  create      Create a model from a Modelfile
  show        Show information for a model
  run         Run a model
  pull        Pull a model from a registry
  push        Push a model to a registry
  list        List models
  cp          Copy a model
  rm          Remove a model
  help        Help about any command

Flags:
  -h, --help      help for ollama
  -v, --version   Show version information
```

- Ollama supports a list of models available here. Below are some example models that can be downloaded:

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

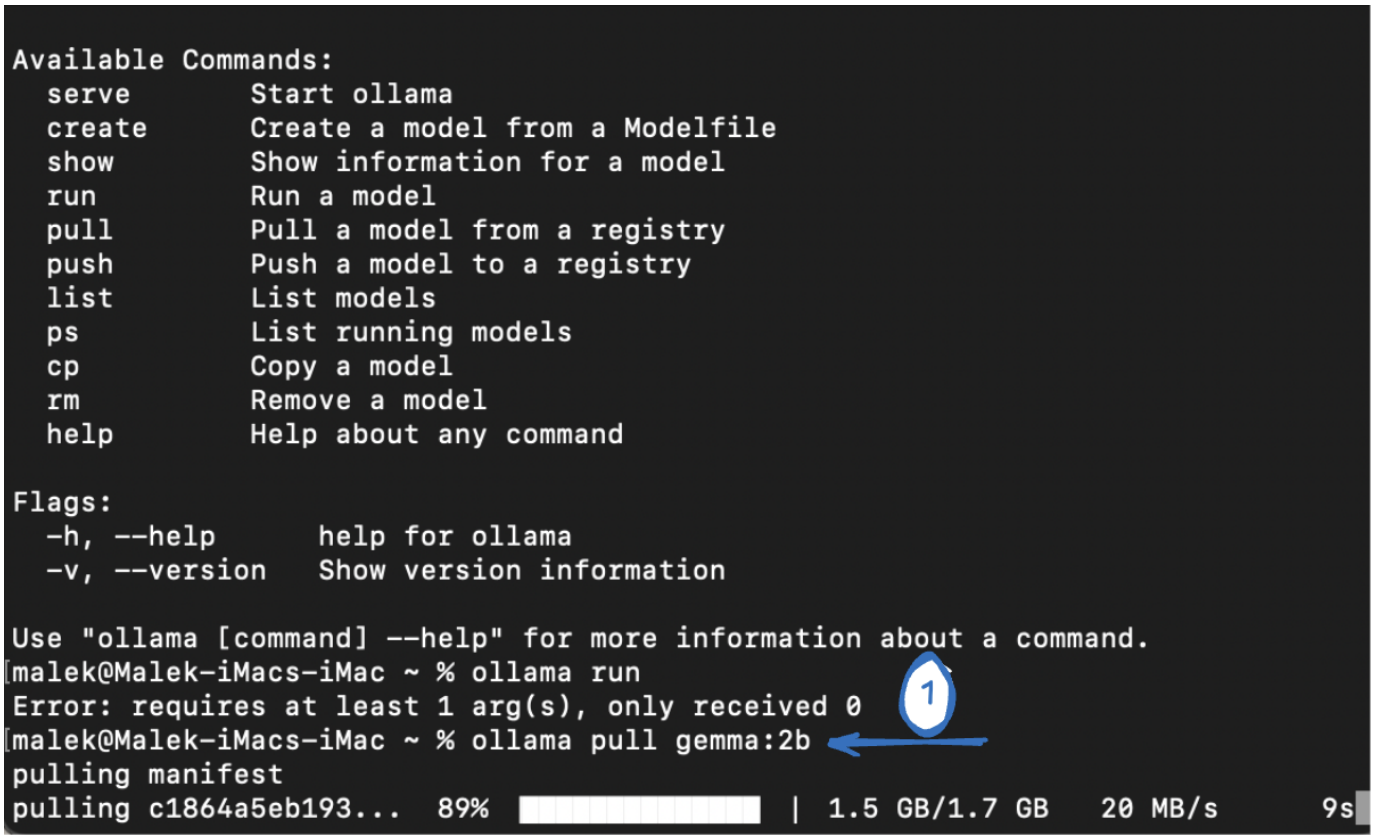
Model	Parameters	Size	Download
Llama 3.1	8B	4.7GB	`ollama run llama3.1`
Llama 3.1	70B	40GB	`ollama run llama3.1:70b`
Llama 3.1	405B	231GB	`ollama run llama3.1:405b`
Phi 3 Mini	3.8B	2.3GB	`ollama run phi3`
Phi 3 Medium	14B	7.9GB	`ollama run phi3:medium`
Gemma 2	2B	1.6GB	`ollama run gemma2:2b`
Gemma 2	9B	5.5GB	`ollama run gemma2`
Gemma 2	27B	16GB	`ollama run gemma2:27b`
Mistral	7B	4.1GB	`ollama run mistral`
Moondream 2	1.4B	829MB	`ollama run moondream`
Neural Chat	7B	4.1GB	`ollama run neural-chat`
Starling	7B	4.1GB	`ollama run starling-lm`
Code Llama	7B	3.8GB	`ollama run codellama`
Llama 2 Uncensored	7B	3.8GB	`ollama run llama2-uncensored`
LLaVA	7B	4.5GB	`ollama run llava`
Solar	10.7B	6.1GB	`ollama run solar`

- To download a model, for example, gemma:2b, which is a lighter model, we can simply type:

SHELL

```
ollama pull gemma:2b
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

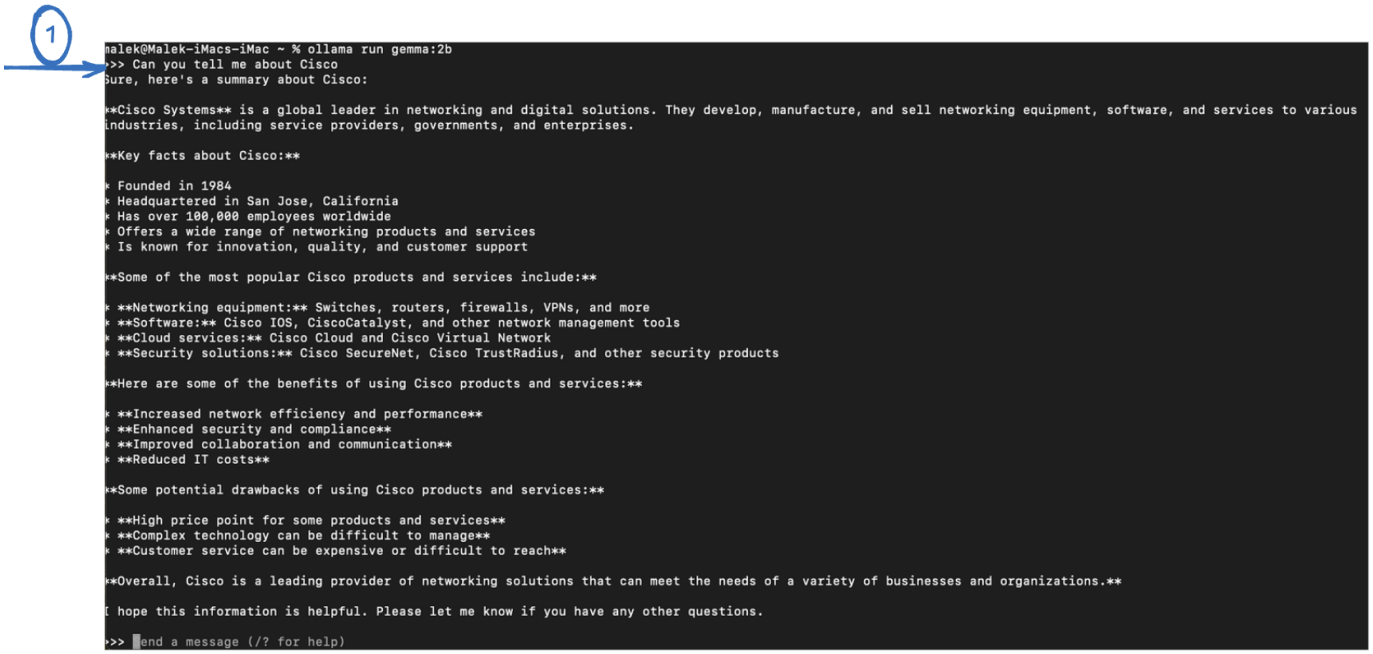


- After pulling the model we can interact with it in the terminal by typing:

SHELL

```
ollama run gemma:2b
```

- We can now ask our question directly in the terminal



- To remove model you can type

SHELL

```
ollama rm gemma:2b
```

- To see the models installed, just enter

SHELL

```
ollama list
```

Web Interface

Once we've installed Ollama, we can interact with it directly through the terminal, as demonstrated earlier. But what if you prefer a web interface? There are several open-source tools available, but I'll show you how to use Open WebUI, which was previously known as Ollama WebUI.

Open WebUI is a versatile, feature-rich, and user-friendly web interface that you can host yourself and use entirely offline. It offers a ChatGPT-style interface, allowing you to interact with language models running on locally (Ollama). This tool is especially useful for those who want to run language models locally or in a self-hosted environment, ensuring both data privacy and control.

- After installing Ollama, simply run the following Docker command to set up the interface

SHELL

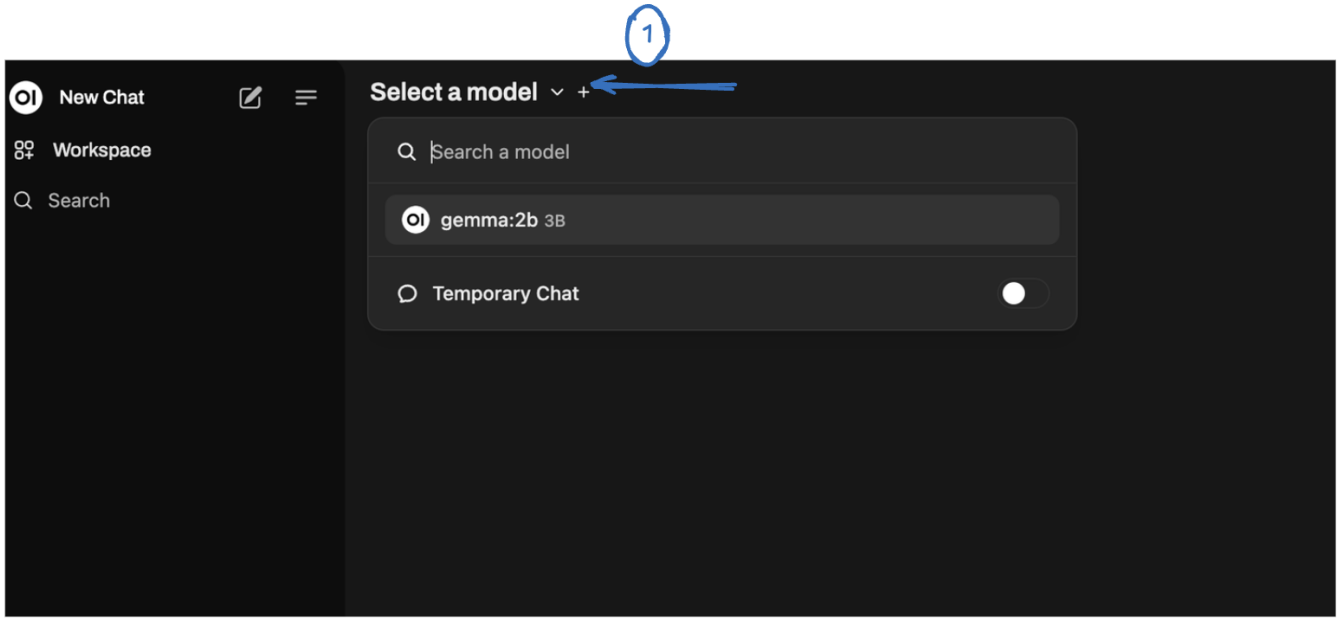
```
docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:main
```

```
malek@Malek-iMacs-iMac ~ % docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:main
Unable to find image 'ghcr.io/open-webui/open-webui:main' locally
main: Pulling from open-webui/open-webui
e4fff0779e6d: Pull complete
d97016d0706d: Pull complete
53db1713e5d9: Pull complete
a8cd795d9ccb: Pull complete
de3ba92de392: Pull complete
6f4d87c224b0: Pull complete
4f4fb700ef54: Pull complete
dd92a6022ddb: Pull complete
bbbfed48a772: Pull complete
a825beebdb5b: Downloading [==========================================>        ]  256.8MB/294.4MB
fd0f6bc6022b: Download complete
```

- Once installed, you can access Open WebUI at http://localhost:3000.

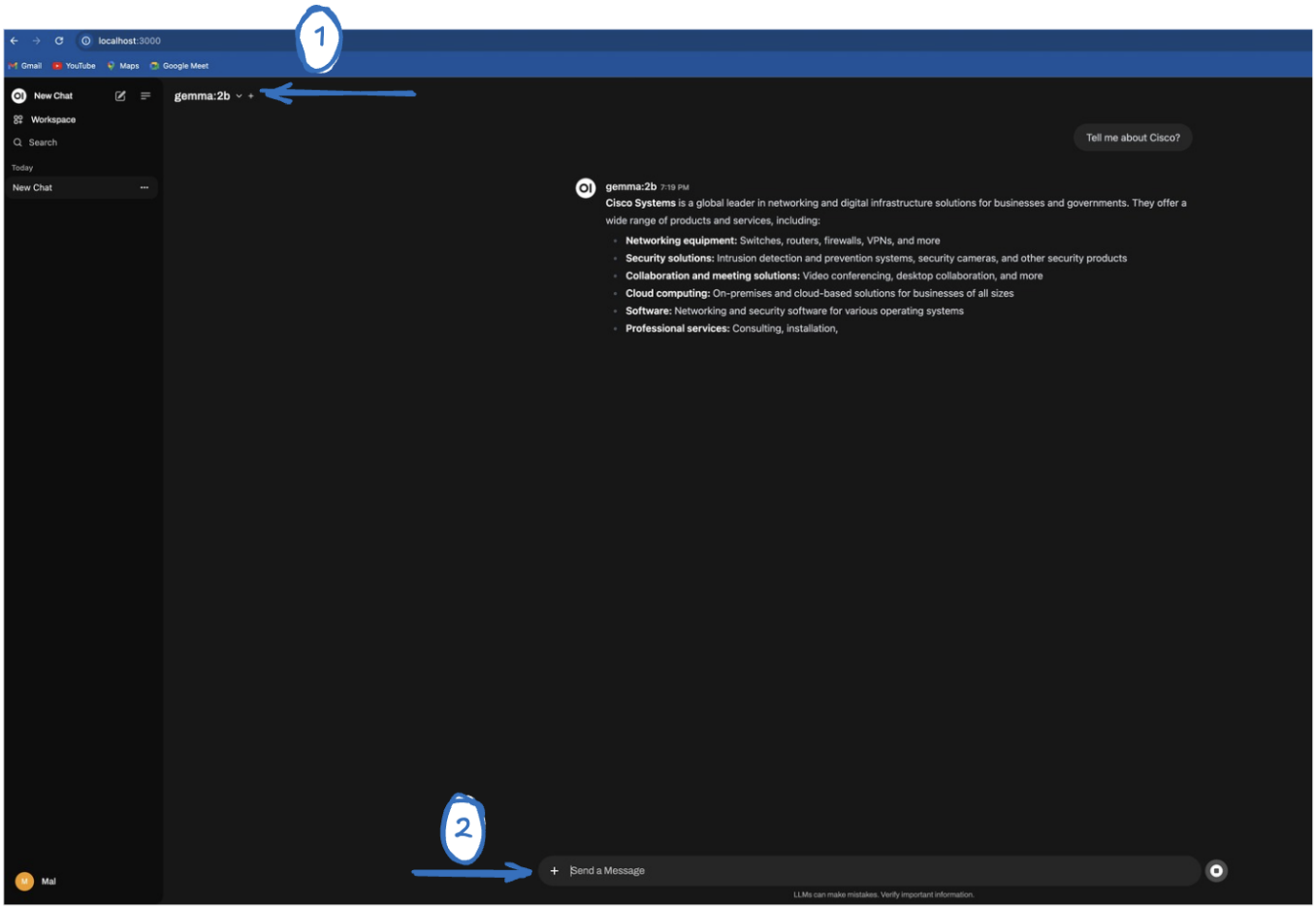
Note: Before accessing the web interface, you will be prompted to create an account. Please follow the instructions.

- In the dropdown menu, you'll see the model you previously installed via the terminal (gemma:2b).

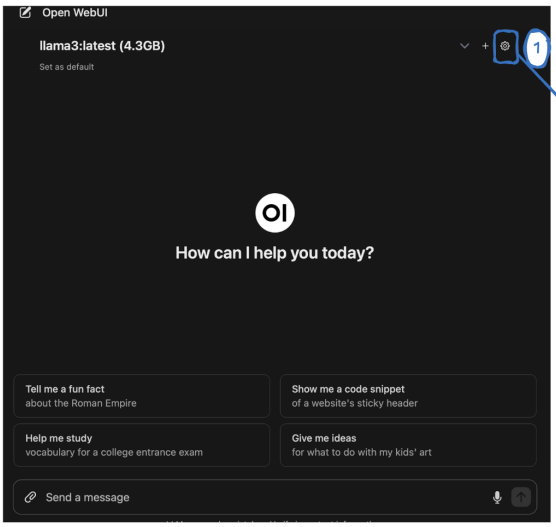


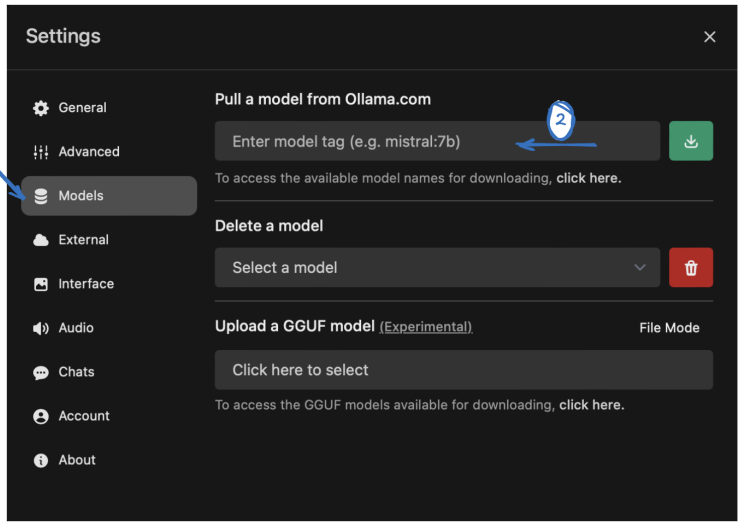
- You can now interact with the model via WebUi

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



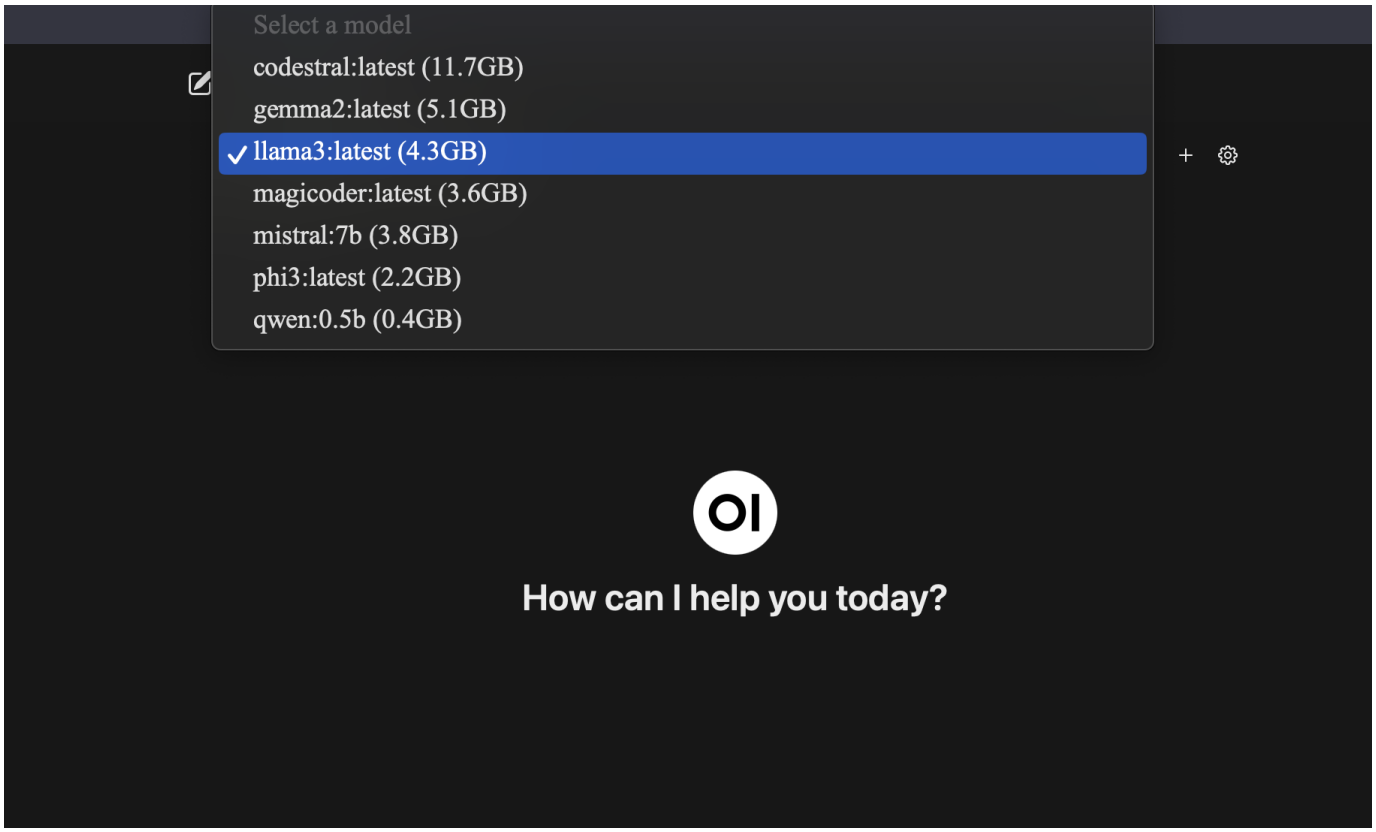
- You no longer need to use the terminal to download a model. Simply navigate to the “Settings --> Models” section in the WebUI, and select the specific model you want to download.





- After installing, all the installed models will be displayed in the UI.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

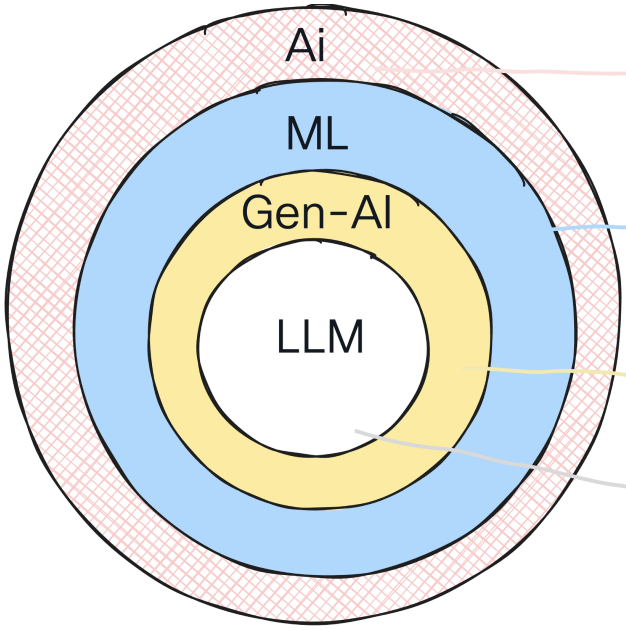


Note: Using LangChain with Ollama allows you to leverage Ollama's capabilities within the LangChain ecosystem. You can find more information here

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.3 AI/ML Revolution Unveiled

1.3.1 Different Types of AI



Teaching something to computers that we thought humans can do

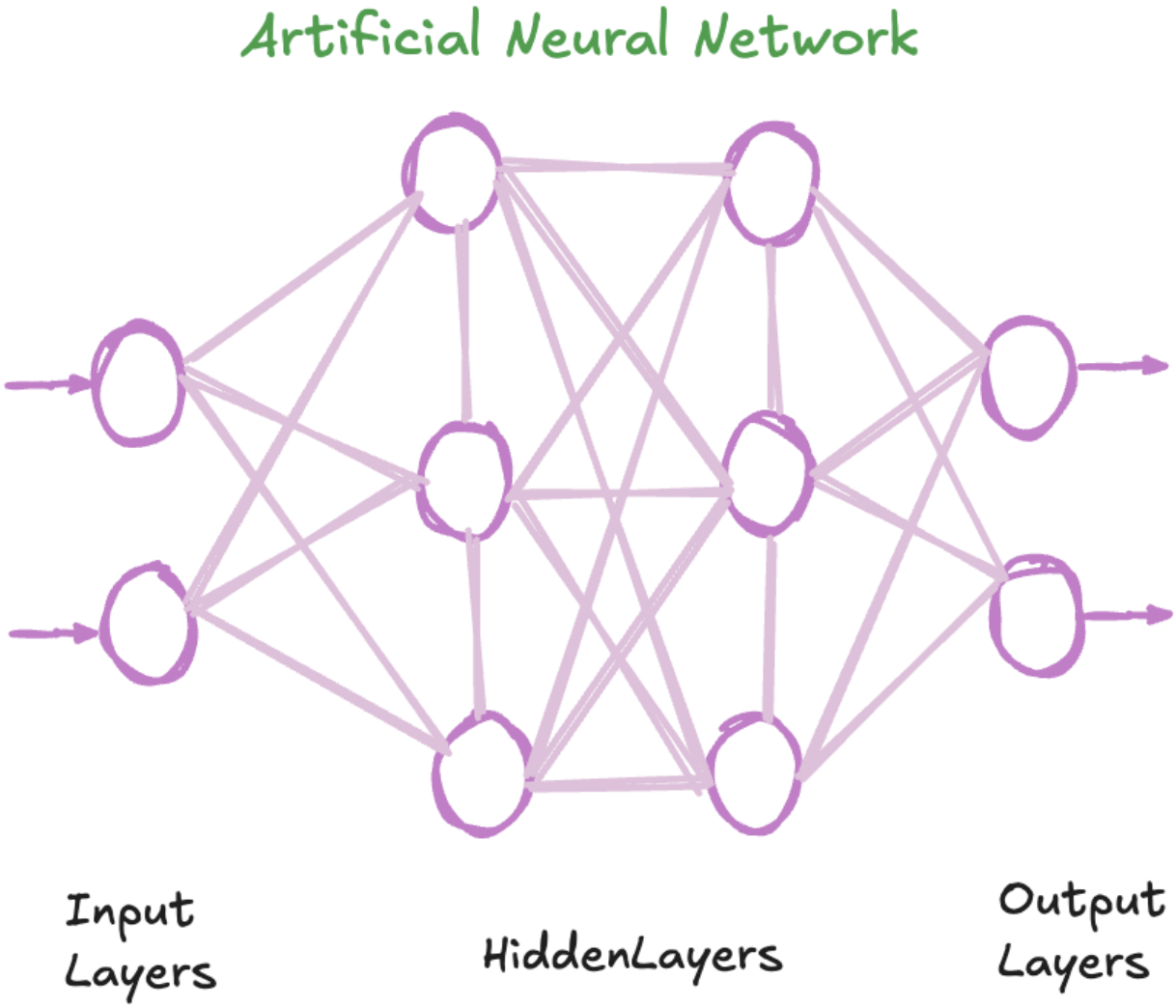
Algorithm technique where we build AI by learning from data

Subset of ML. Is an AI that generate things, e.g Image , music , language e.t.c

Are built after reading lots of data e.g GPT4, Gemini, Llama2 e.t.c

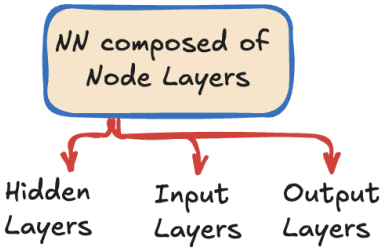
Remember: Subset of AI is ML and kind of ML is Gen-AI . LLM e.g GPT are subset of Gen-AI

1.3.2 Neural Network

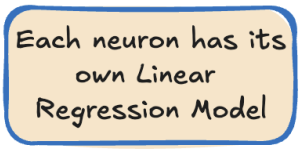


Note: The way nodes are connected called weights

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



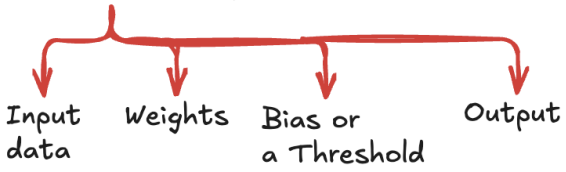
These neural network refelects the behaviour of human brain allowing computers to recognize patterns and solve common problems

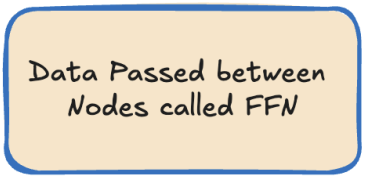


Linear Regression is the mathematical model that is used to predict future events

The weights of the connections between the nodes determine how much influence each input has on output

Each Node is composed of



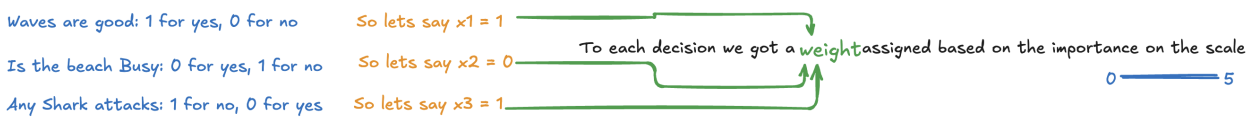


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

The data is passed in the neural network from one layer to another which is known as feed forward propogation

Example: Lets see how a single node in NN looks like when deciding if you wanna go surfing.
The decision to go or not wil be our predictive outcome or lets call it our yHat

Assume there are 3 factors influencing our decision:



Lets assign weights to our decisions:

So Waves are good we will assign a weight w1 = 5

Beach not busy not of great importance, w2 = 2

No Shark attacks, lets agive it a weight w3 = 4

Overall Result:

x1 = 1	x2 = 0	x3 =1
w1 = 5	w2 = 2	w3 =4

We can plus these values into our formulae to get the desired output:

$$yHat = (1*5) + (0*2) + (1*4)$$

$$yHat = 9 - 3 = 6$$

Threshold or Bias

As 6 is greater then 0 so the output of this node is 1. We are going surging

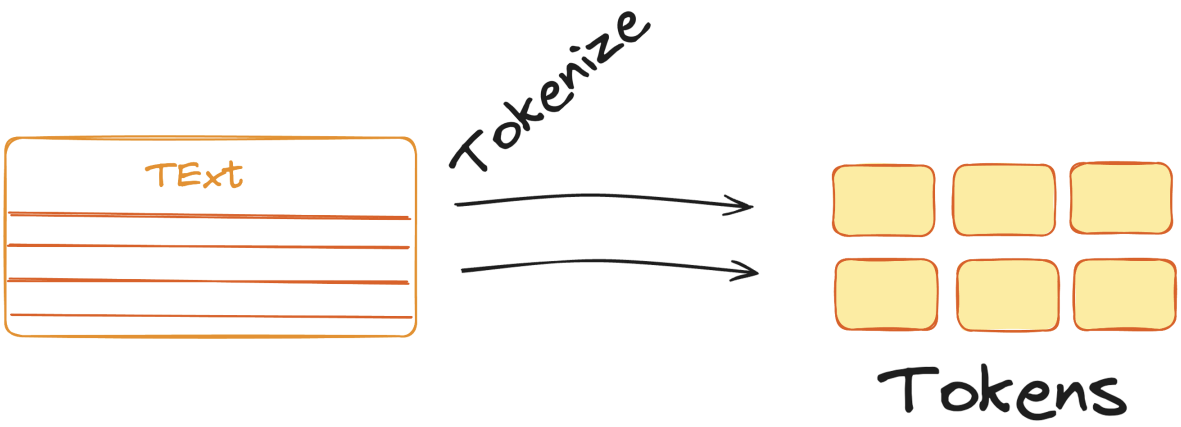
We can always adjust weights to get different outcomes

Note: Neural network rely on training data to learn and improve their accuracy over time. We used supervised learning to train the algorithm. There are multiple types of Neural network other than the Feed forward network that we have defined here. Example: CNN: Convolutional Neural Network - Unique architecture for identifying patterns like image recognition, or RNN: Recurrent neural network, that uses time series data to make prediction about future events like sales forecasting.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

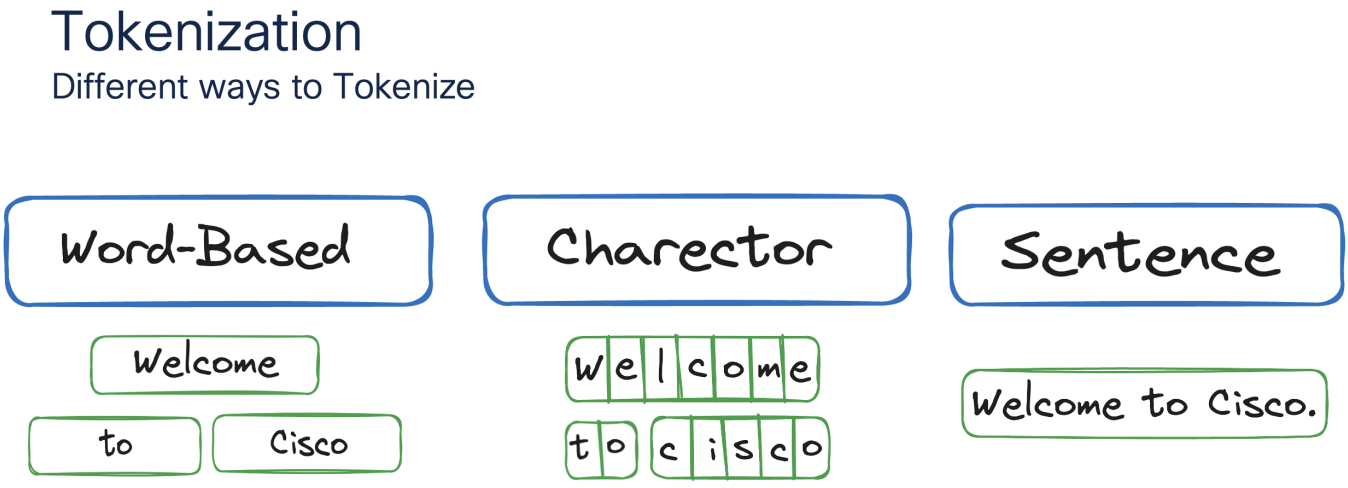
1.4 Tokenization

What is Tokenization and why its needed?



- Language models have a limit on how much text they can handle at once, known as their context window. While these limits are expanding, research shows that LLM's often perform better when provided with less, but more relevant, information. However, selecting the most relevant information is straightforward for humans but challenging for computers.

A common approach to manage large amounts of data is to break it down into smaller, more manageable parts a process often referred to as Tokenization or Chunking. Tokenization is a key step in this process, where raw text is divided into smaller units, called tokens, which can then be processed by a neural network.



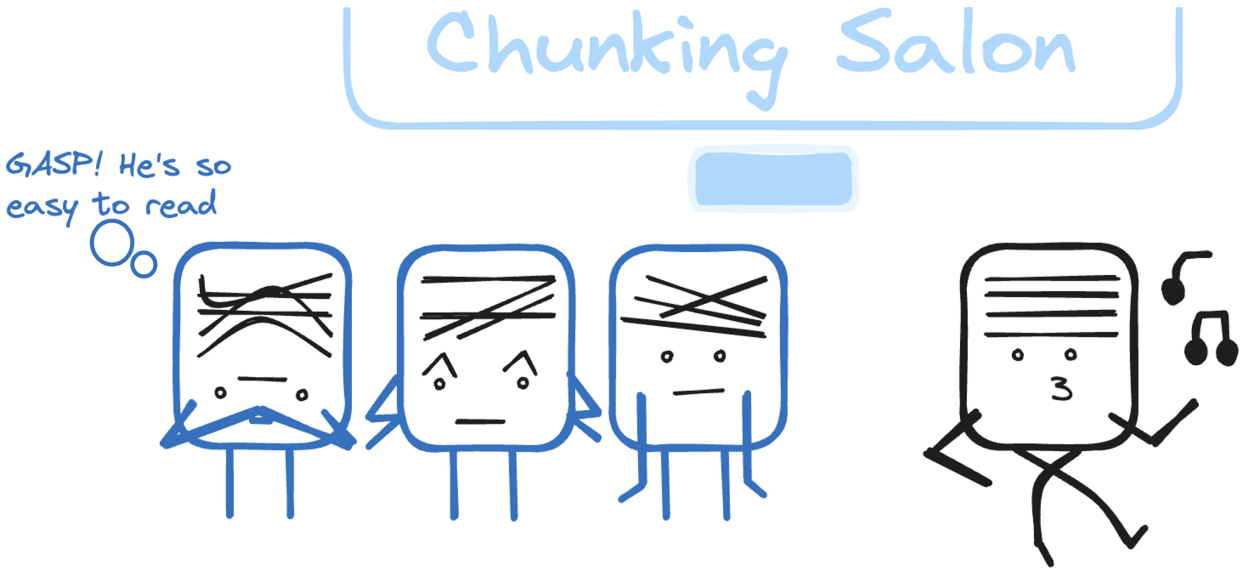
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

In order to do this you need to pick a chunk strategy. Just to name a few:

- Word-Based: A simple and straightforward method that most of us would propose is to use word-based tokens, splitting the text by spaces.
- Character based tokenization: Individual words are considered as tokens . Lot of computing resources needed as now e.g for a 3 word serentence where you might need 3 tokens now 15 - 20 tokens needed
- Sentence Based: We need a .(fullstop) for it to work

Note: We've all heard of GPT and OpenAI. They utilize a tokenization method called Byte Pair Encoding (BPE), which is a middle ground between word-based and character-based tokenization. In BPE, words are broken down into smaller character sequences that the model encountered during training, allowing it to make informed predictions.

1.4.1 Why we need Tokenization or Text Splitting?

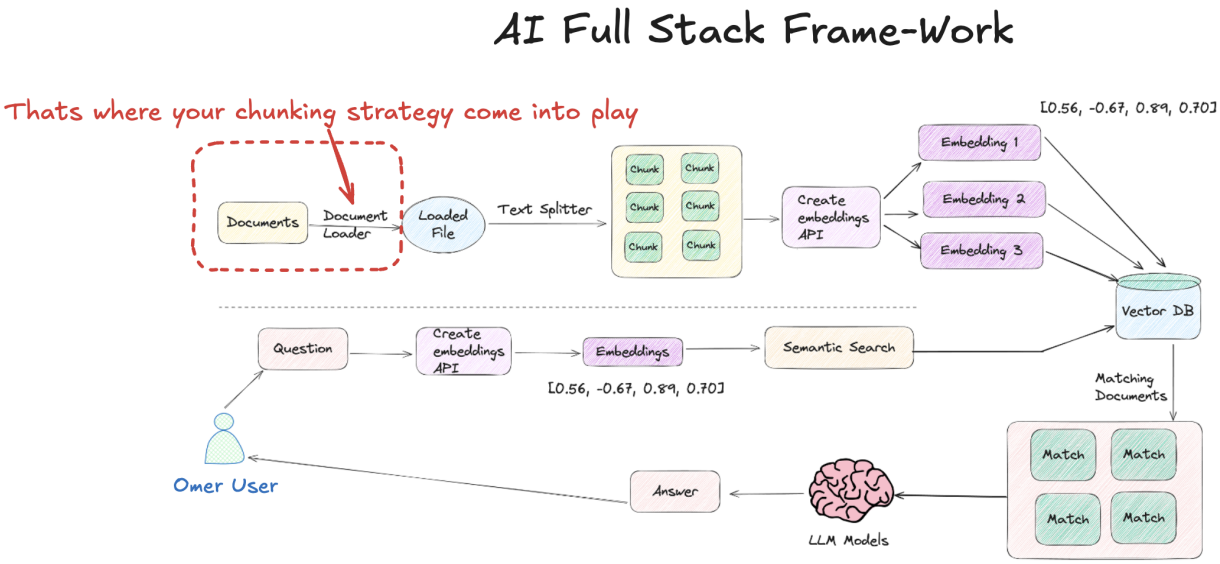


Historically, applications perform better when they are provided with your own data. However, you can't feed unlimited data to your LLMs due to two key limitations:

- Context window limit: LLMs have a finite context window for processing data.
- Signal-to-noise ratio (SNR): LLMs perform better when the SNR is high, meaning the information provided is useful, relevant, and clear. Clear, unambiguous instructions help the model deliver more accurate and detailed results, while ambiguous or complex input can lead to less accurate or incomplete outputs.

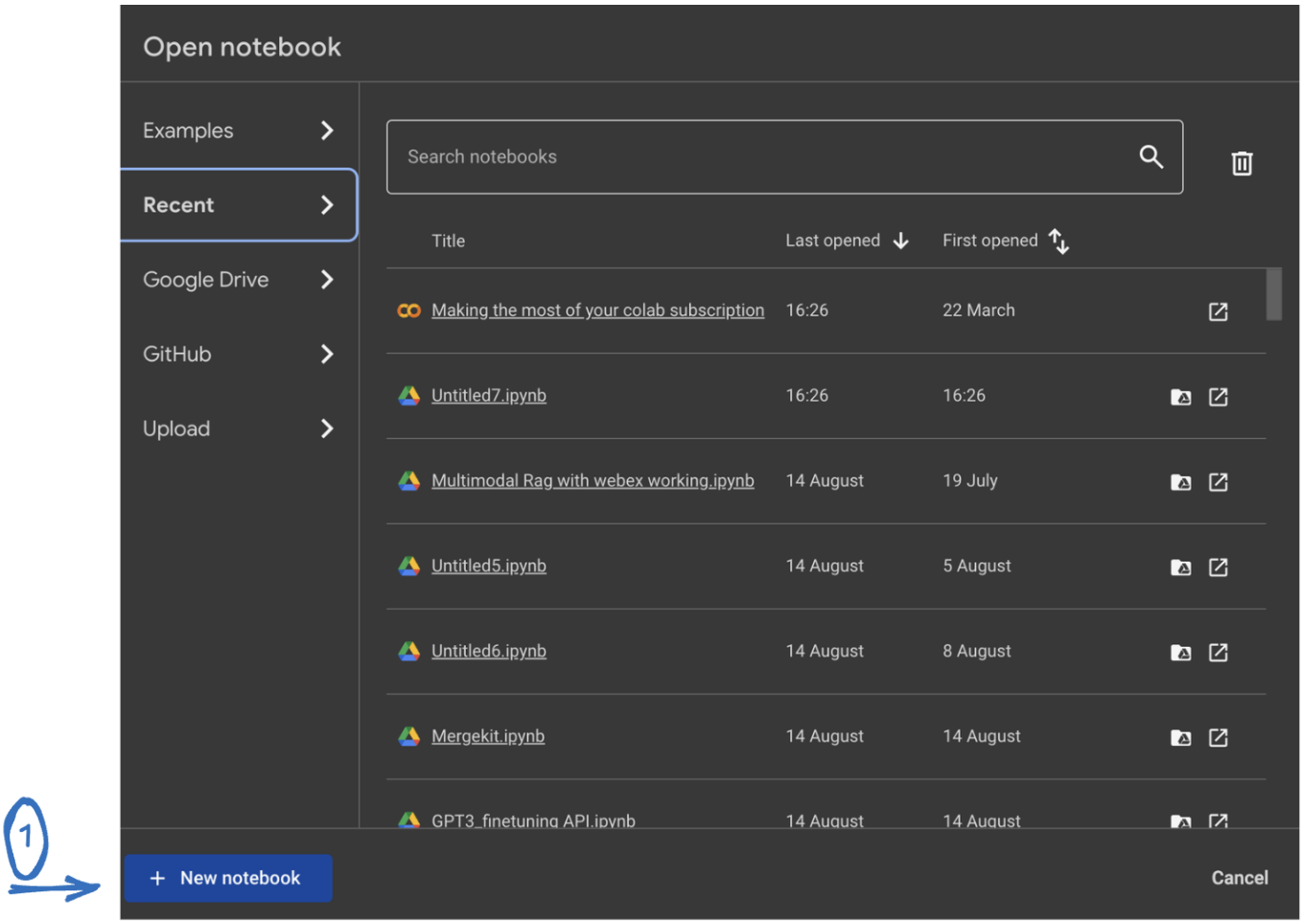
As noted, chunking or splitting refers to breaking your data into smaller, manageable pieces.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



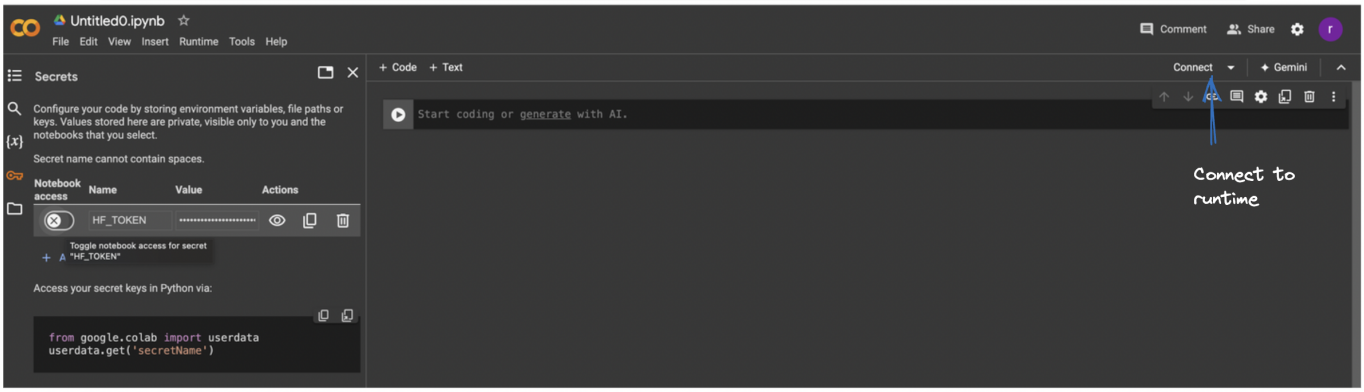
1.4.2 Task 1: Log into the Lab Environment

- Open Google Colab and create a new notebook. Click on "File" > "New notebook". Please refer to the following section to create Google Colab account.

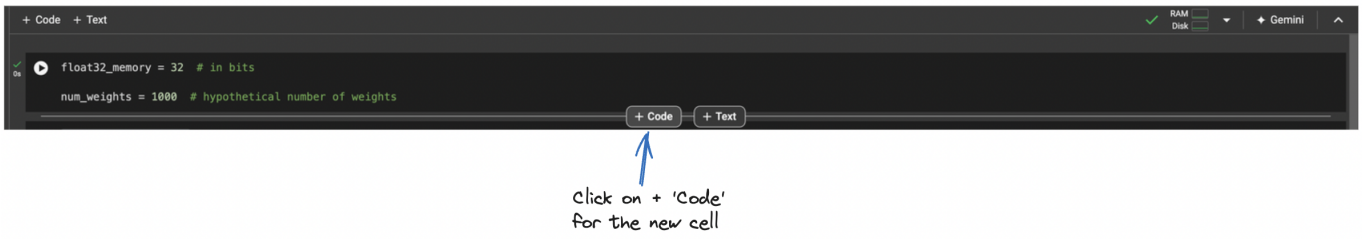


- Make sure you are connected to a runtime. For this task, you can use the CPU as the runtime environment.

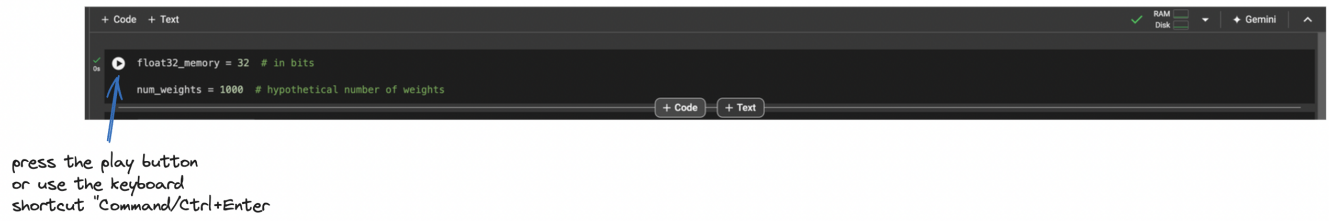
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Reminder: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



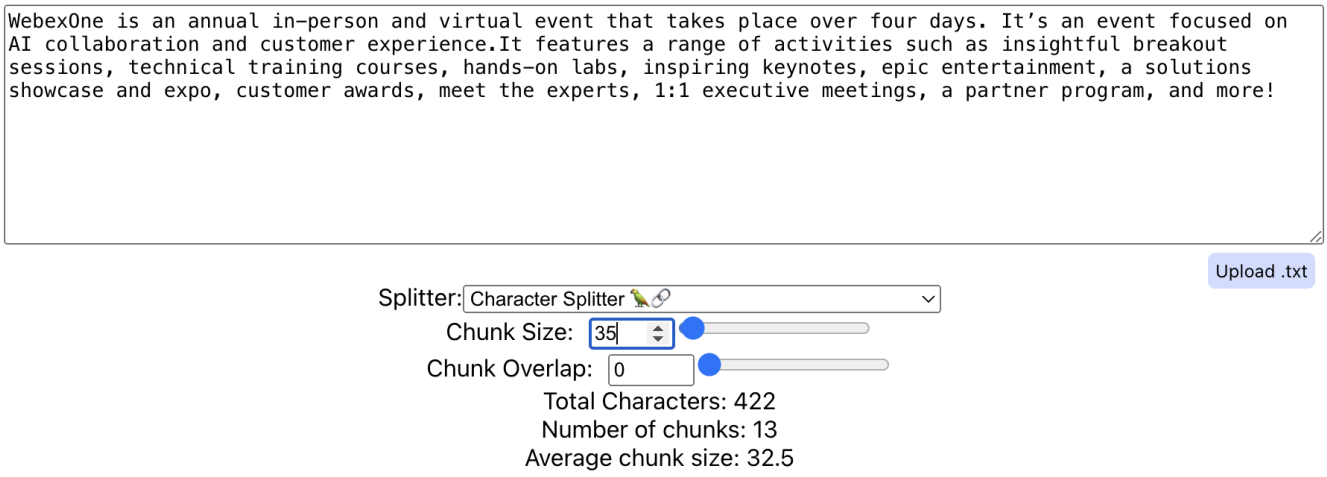
Reminder: Click the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter" while the cell is selected.



Manual testing for Chunking

We will create Chunks of 35 charectors, so first 35 charectors as chunk 1 , next 35 charectors chunk2 and so on

```
# Manual Splitting
text = """WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer
experience.
It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic
entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!"""

# Create a list that will hold your chunks
chunks = []

chunk_size = 35

# Run through the a range with the length of your text and iterate every chunk_size you want
for i in range(0, len(text), chunk_size):
    chunk = text[i:i + chunk_size]
    chunks.append(chunk)

print(chunks)
```



Note: The text contained a total of 422 characters, divided into 13 chunks. Problem with the above chunking technique is that it got split at 'r'. How we know when to chunk . Before we look into that. Lets look into Langchain Splitter

LangChain Text Splitter

- Let's retrieve the langchain library from the Python Package. In the example below, we'll configure the chunk_overlap parameter, which ensures that our chunks are blended together—meaning the end of chunk 1 will overlap with the beginning of chunk 2.

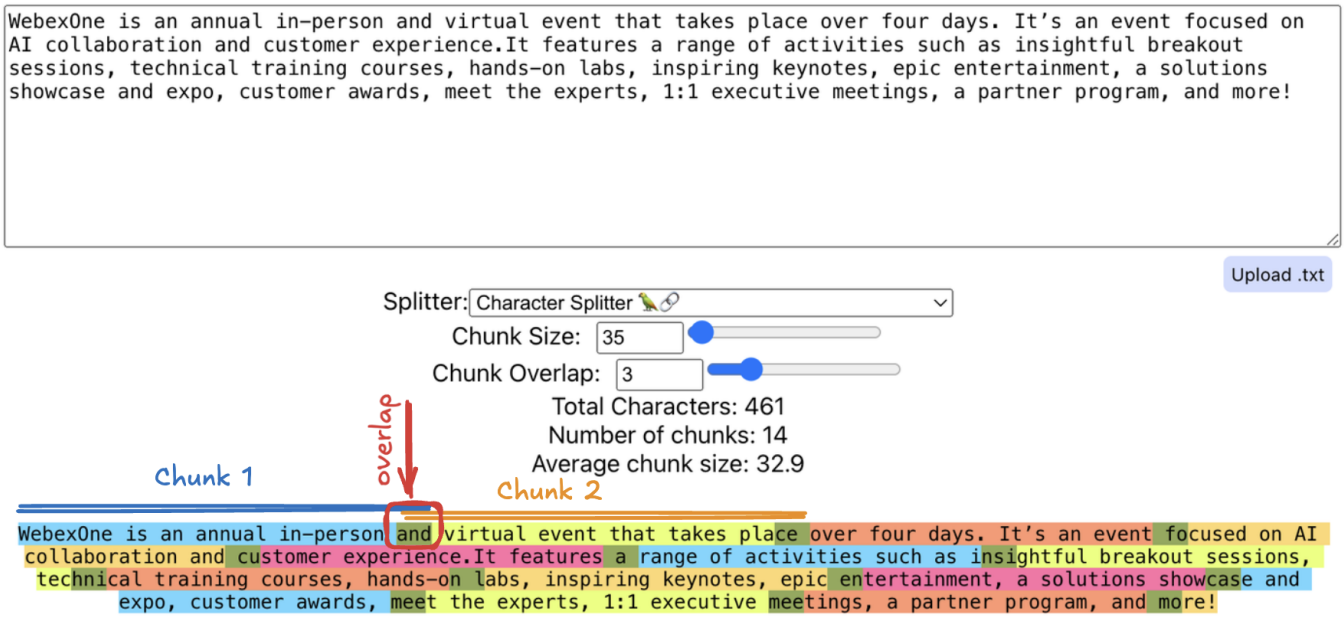
```
!pip install langchain
```

```
Collecting langchain
  Downloading langchain-0.2.14-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0.32)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (3.10.2)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Collecting langchain-core<0.3.0,>=0.2.32 (from langchain)
  Downloading langchain_core-0.2.33-py3-none-any.whl.metadata (6.2 kB)
Collecting langchain-text-splitters<0.3.0,>=0.2.0 (from langchain)
  Downloading langchain_text_splitters-0.2.2-py3-none-any.whl.metadata (2.1 kB)
Collecting langsmith<0.2.0,>=0.1.17 (from langchain)
  Downloading langsmith-0.1.99-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.26.4)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.8.2)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.32.3)
Collecting tenacity!=8.4.0,<9.0.0,>=8.1.0 (from langchain)
  Downloading tenacity-8.5.0-py3-none-any.whl.metadata (1.2 kB)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (2.3.5)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.4)
Collecting jsonpatch<2.0,>=1.33 (from langchain-core<0.3.0,>=0.2.32->langchain)
  Downloading jsonpatch-1.33-py2.py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.32->langchain) (24.1)
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/python3.10/dist-packages (from langchain-core<0.3.0,>=0.2.32->langchain) (4.12.2)
```

```
text = """WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer
experience.
It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic
entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!"""
```

```
from langchain.text_splitter import CharacterTextSplitter

text_splitter = CharacterTextSplitter(chunk_size=35, chunk_overlap=3, separator='', strip_whitespace=False) # seprate means you splt by charector

a = text_splitter.create_documents([text])

print(a)
```

Note: Seprators are charectors sequence you wanna split on.



Recursive Charector Text Splitting

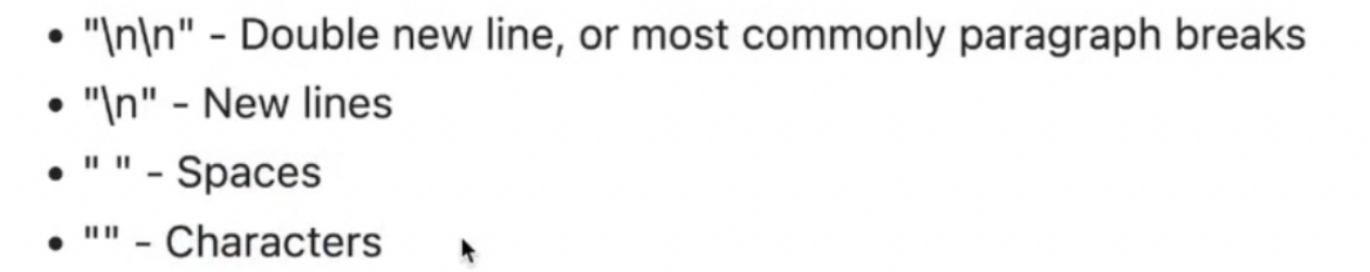
In the previous example, when we used Character Splitting, we split the text based on a fixed number of characters. Specifically, we divided the text into chunks of 35 characters each.

However, with Recursive Text Splitting (RTS), the process is more dynamic and considers the physical structure of the text to determine the appropriate chunk size.

Here's how RTS works:

- Instead of relying on a static number of characters, RTS examines the structure of the document.
- It begins by identifying the largest logical divisions, such as paragraphs, and splits the text at each double newline (indicating paragraph breaks).
- If any of these chunks are still too large, RTS will then move to the next level of separation, which is single newlines (often indicating sentences or list items).
- If necessary, it will continue to break down the text further, using spaces and eventually individual characters as separators.

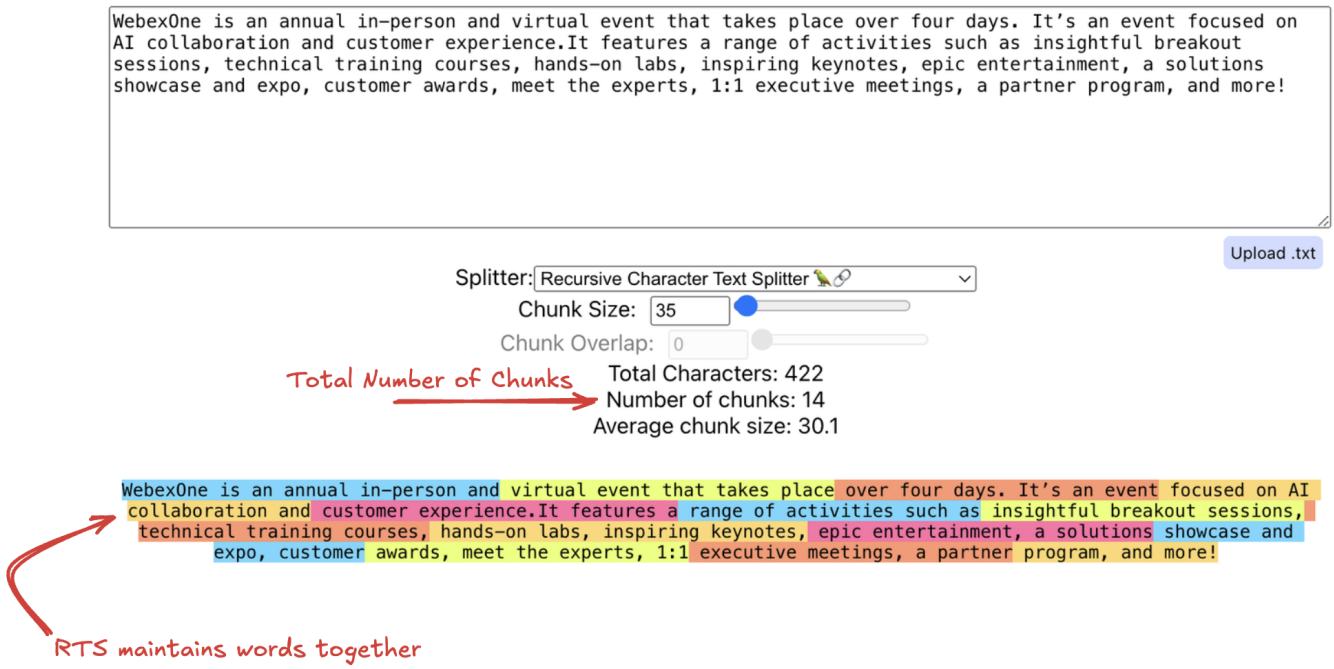
With RTS, you don't need to manually specify the chunk size by character count. You simply pass your text to the RTS process, and it will intelligently determine how to split the text based on its structure, resulting in chunks that are logically organized and more contextually meaningful.




```
text = """WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer
experience.
It features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic
entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!"""
```

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(chunk_size = 35, chunk_overlap = 0)
# how many chunks we have
print(len(text_splitter.create_documents([text])))

text_splitter.create_documents([text])
```

Note: We avoid splitting in the middle of words by using spaces as one of the separators, ensuring that each chunk ends on a complete word. This is crucial because RCS (Recursive Character Splitting) helps maintain the context within sentences by keeping related words together.



Note:If you're new to AI, I would personally recommend starting with Recursive Text Splitting (RTS).

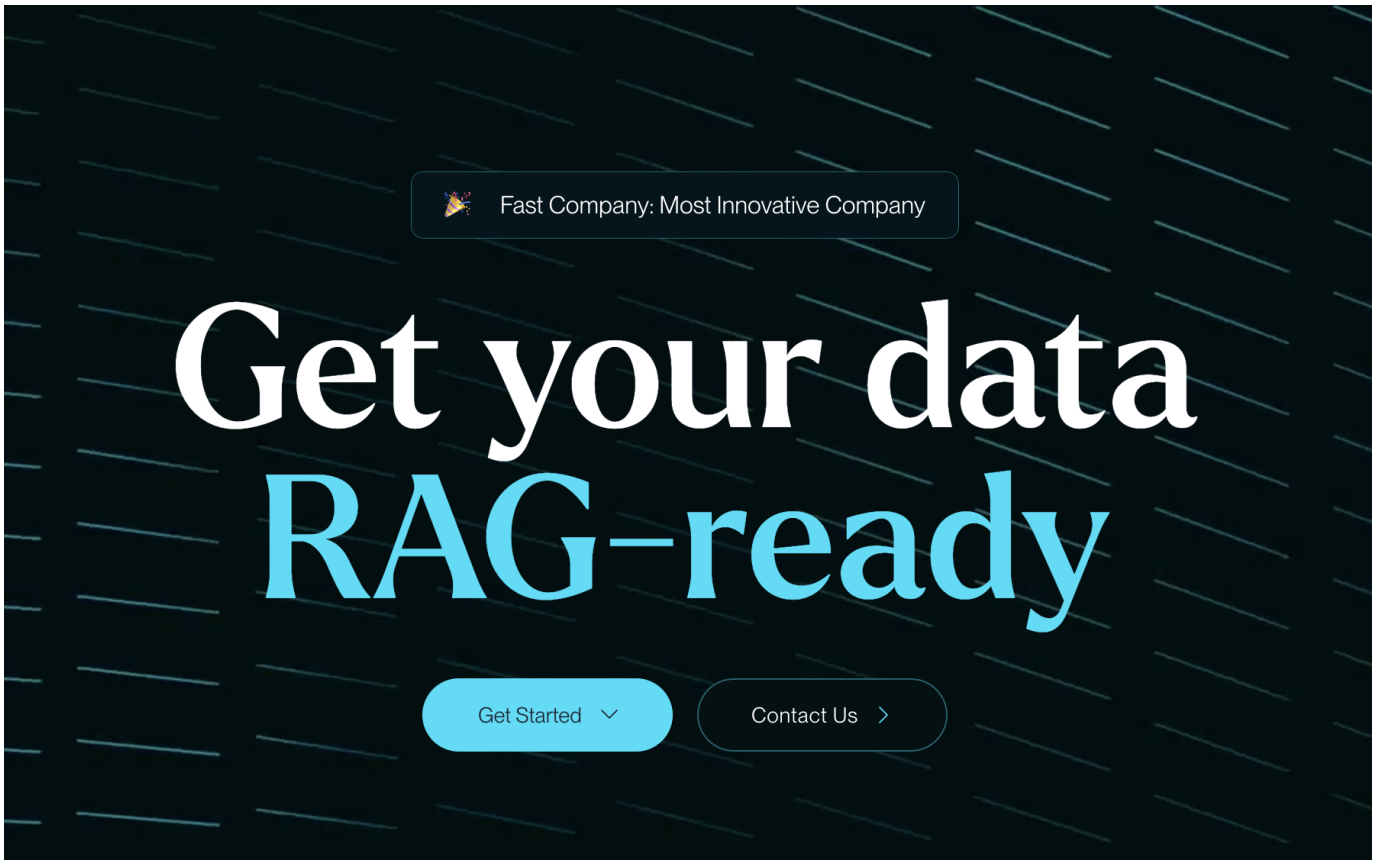
Document Level Splitting

So far, we've been working with splitting regular documents. But what if we have markdown files or PDF or Python documentation? There's a better way to handle those cases, and that's where specialized document splitting comes into play.

PDF WITH TABLE

PDFs often contain tables and other structured data that can be challenging to split accurately using character-based methods. For PDFs, it's important to extract and chunk all elements, including tables, effectively. We'll accomplish this using the Unstructured library, which is specifically designed for handling such tasks. If you have a large collection of PDFs, Unstructured is an excellent tool to manage them efficiently.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Install relevant libraries including Unstructured

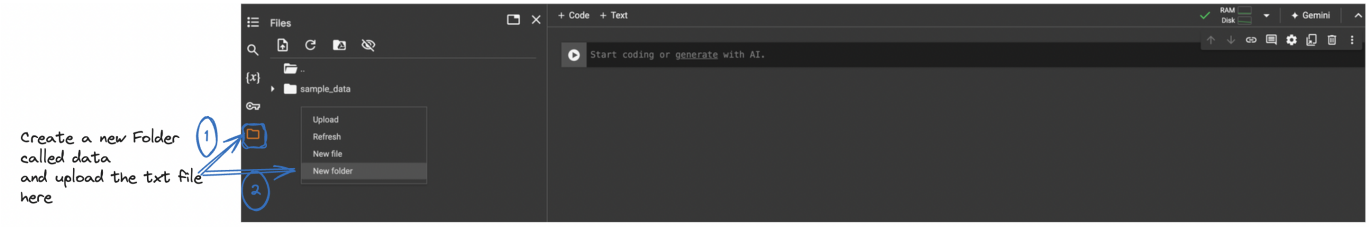
```
!pip install scikit-learn langchain_community unstructured[all-docs] unstructured pdfminer pdfminer.six pdf2image pillow_heif opencv-python
unstructured_inference pytesseract unstructured_pytesseract python-dotenv openai
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: langchain_community in /usr/local/lib/python3.10/dist-packages (0.2.12)
Requirement already satisfied: unstructured in /usr/local/lib/python3.10/dist-packages (0.15.5)
Requirement already satisfied: pdfminer in /usr/local/lib/python3.10/dist-packages (20191125)
Requirement already satisfied: pdfminer.six in /usr/local/lib/python3.10/dist-packages (20231228)
Requirement already satisfied: pdf2image in /usr/local/lib/python3.10/dist-packages (1.17.0)
Requirement already satisfied: pillow_heif in /usr/local/lib/python3.10/dist-packages (0.18.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: unstructured_inference in /usr/local/lib/python3.10/dist-packages (0.7.36)
Requirement already satisfied: pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.13)
Requirement already satisfied: unstructured_pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.13)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (2.0.32)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (3.10.2)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.6.7)
Requirement already satisfied: langchain<0.3.0,>=0.2.13 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.2.14)
Requirement already satisfied: langchain-core<0.3.0,>=0.2.30 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.2.33)
Requirement already satisfied: langsmith<0.2.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.1.99)
```

```
!apt-get install -y poppler-utils && apt-get install -y tesseract-ocr
```

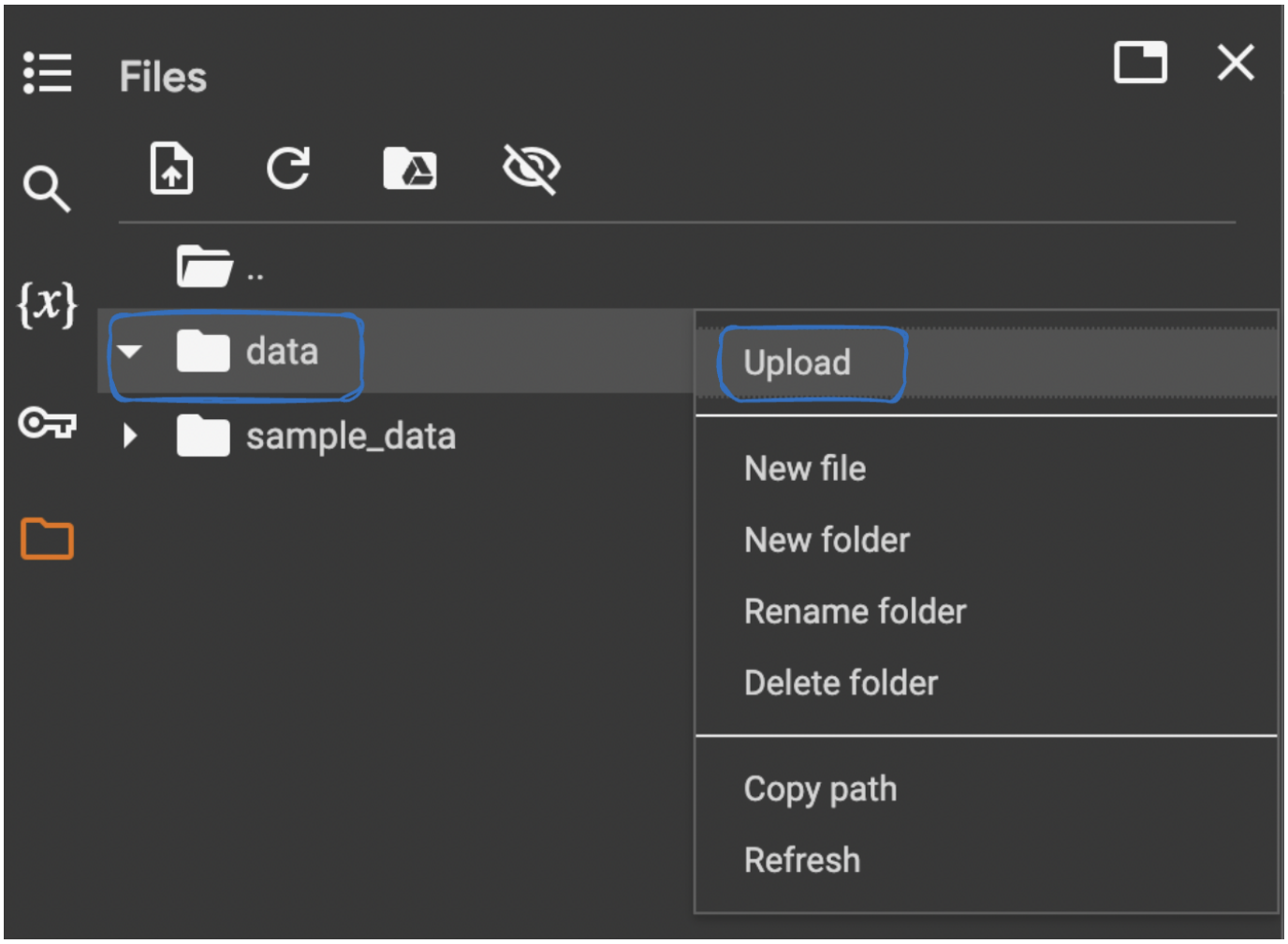
```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  poppler-utils
0 upgraded, 1 newly installed, 0 to remove and 45 not upgraded.
Need to get 186 kB of archives.
After this operation, 696 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 poppler-utils amd64 22.02.0-2ubuntu0.5 [186 kB]
Fetched 186 kB in 1s (228 kB/s)
Selecting previously unselected package poppler-utils.
(Reading database ... 123594 files and directories currently installed.)
Preparing to unpack .../poppler-utils_22.02.0-2ubuntu0.5_amd64.deb ...
Unpacking poppler-utils (22.02.0-2ubuntu0.5) ...
Setting up poppler-utils (22.02.0-2ubuntu0.5) ...
Processing triggers for man-db (2.10.2-1) ...
```

- Let's load our PDF files into Google Colab. For this example, we can use the Cisco Financial Results. Please Download the file here as we will be using in the next step.
- Within Google Colab, Click on Folder and create a new folder called "data"



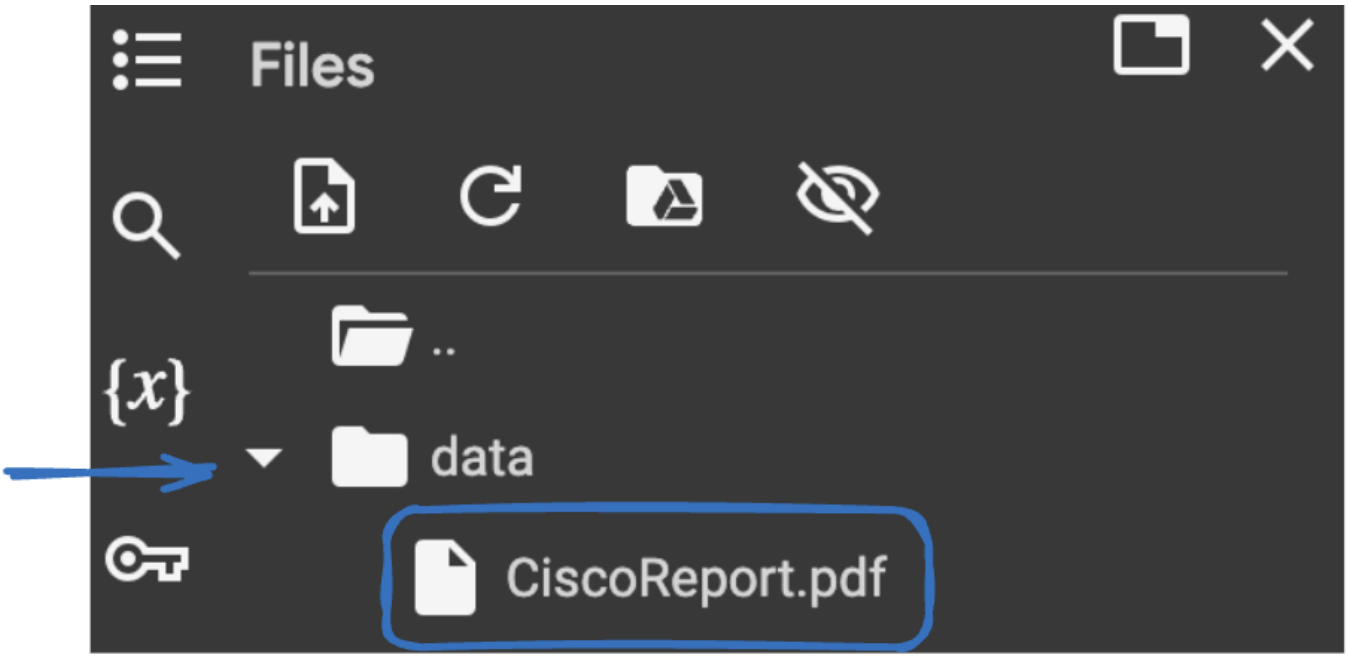
- Click on [...], select Upload

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Choose your CiscoReport.pdf file and click Open

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- We'll use the following code from the Unstructured library to demonstrate how tables can be extracted

```
import os
from unstructured.partition.pdf import partition_pdf
from unstructured.staging.base import elements_to_json

# Let's load up our PDF and then partition it.
filename = "/content/data/CiscoReport.pdf"# Use relative path since the file is in the same directory

# Extracts the elements from the PDF
elements = partition_pdf(
    filename=filename,
    extract_images_in_pdf=True,
    strategy="hi_res",
    infer_table_structure=True,
    hi_res_model_name="yolox"
)

# Let's look at our elements
print(elements)
```

```
config.json: 100% 1.47k/1.47k [00:00<00:00, 76.5kB/s]
model.safetensors: 100% 115M/115M [00:01<00:00, 125MB/s]
model.safetensors: 100% 46.8M/46.8M [00:00<00:00, 117MB/s]
<unstructured.documents.elements.Title at 0x7ba52cc9c5e0>,
<unstructured.documents.elements.ListItem at 0x7ba52cc9d7b0>,
<unstructured.documents.elements.ListItem at 0x7ba52cc9d3f0>,
<unstructured.documents.elements.Title at 0x7ba52cc9caf0>,
<unstructured.documents.elements.ListItem at 0x7ba4c0997ca0>,
<unstructured.documents.elements.Text at 0x7ba4c09943a0>,
<unstructured.documents.elements.NarrativeText at 0x7ba52cc9ebc0>,
<unstructured.documents.elements.NarrativeText at 0x7ba52cc9f370>,
<unstructured.documents.elements.NarrativeText at 0x7ba52cc9d720>,
<unstructured.documents.elements.Title at 0x7ba52cc9de10>,
<unstructured.documents.elements.Text at 0x7ba4c0997f10>,
<unstructured.documents.elements.Text at 0x7ba4c0994460>,
<unstructured.documents.elements.Text at 0x7ba4c0994520>,
<unstructured.documents.elements.Table at 0x7ba4c0994160>,
<unstructured.documents.elements.NarrativeText at 0x7ba4c09944f0>,
<unstructured.documents.elements.Title at 0x7ba4c09946a0>,
<unstructured.documents.elements.Text at 0x7ba4c09947f0>,
<unstructured.documents.elements.Text at 0x7ba4c0994730>,
<unstructured.documents.elements.Text at 0x7ba4c0994430>,
<unstructured.documents.elements.Table at 0x7ba4c0995150>,
<unstructured.documents.elements.Title at 0x7ba4c0994790>,
```

Regular Text

Table

- Lets grab element 37 as its the one where our table is

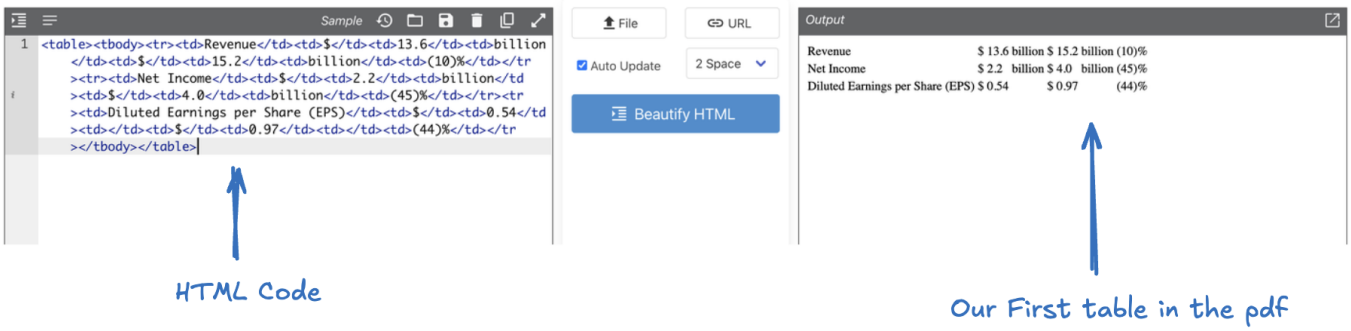
```
elements[37].metadata.text_as_html
```

OUTPUT

```
'<table><thead><tr><th>Revenue</th><th>$8.51 - $8.53 Billion</th><th>$34.5 - $34.7 Billion</th></tr></thead><tbody><tr><td>Y/Y Growth</td><td>~10%</td><td>~10%</td></tr><tr><td>FX Impact)</td><td>no impact</td><td>no impact</td></tr><tr><td>GAAP Operating Margin</td><td></td><td>~11.4%</td></tr><tr><td>Non-GAAP Operating Margin')</td><td></td><td>~28.0%</td></tr><tr><td>GAAP Earnings per Share?)</td><td>$0.79 - $0.80</td><td>$2.67 - $2.69</td></tr><tr><td>Non-GAAP Earnings per Share()</td><td>$1.89 - $1.90</td><td>$7.41 - $7.43</td></tr><tr><td>Operating Cash Flow Growth (Y/Y)°)</td><td></td><td>16% - 17%</td></tr><tr><td>Current Remaining Performance Obligation Growth (Y/Y)</td><td>~10%</td><td></td></tr></tbody></table>'
```

- Tables are straightforward for humans to read, but they aren't as easy for language models to interpret. Language models are typically trained on HTML tables, so when you pass HTML-formatted tables to an LLM, it will better understand the structure and content. You can paste the HTML into an HTML viewer to see how it looks.

HTML Viewer



Note: That's how you handle tables within a PDF.

Multi-Modal (Text + images)

What if there are images within a PDF or other documents? How can you extract them? We'll use the Unstructured library again to handle this.

EXTRACT IMAGES WITHIN PDF

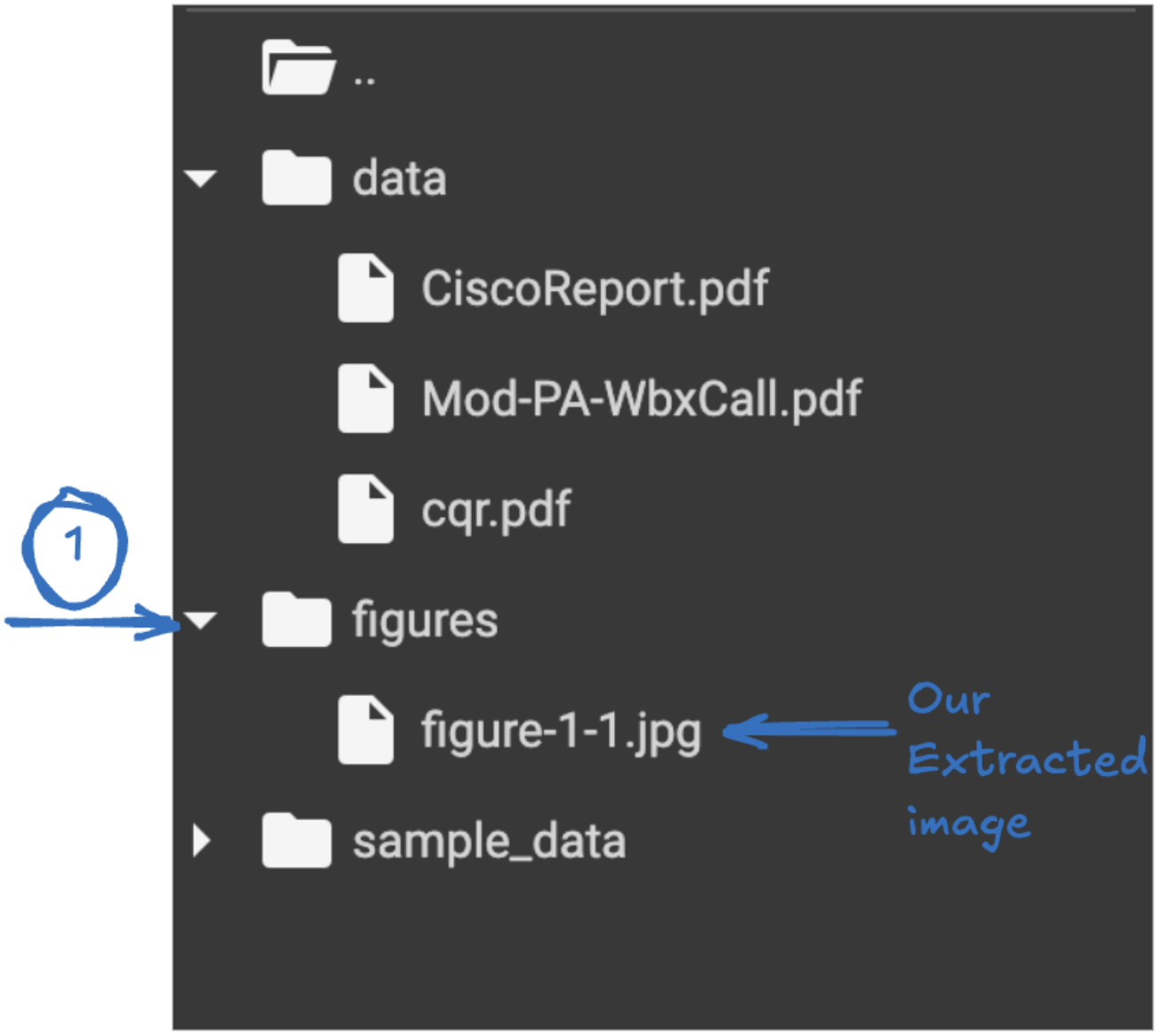
- Install relevant libraries

```
from typing import Any
from pydantic import BaseModel
from unstructured.partition.pdf import partition_pdf
```

```
filepath = "/content/data/CiscoReport.pdf"
# Get elements
raw_pdf_elements = partition_pdf(
    filename=filepath,

    # Using pdf format to find embedded image blocks
    extract_images_in_pdf=True,

    # Use layout model (YOLOX) to get bounding boxes (for tables) and find titles
    infer_table_structure=True,

    # Specifies the strategy to be used for chunking the text. In our case, it will chunk the text based on titles found in the document.
    chunking_strategy="by_title",

    # Sets a hard limit on the number of characters allowed in each chunk.
    max_characters=4000,
    # This function will attempt to create a new chunk after every 3,800 characters, allowing some flexibility while chunking.
    new_after_n_chars=3800,
    # If a chunk has fewer than 2,000 characters, the function will
    # attempt to combine it with neighboring text blocks to create a larger, more meaningful chunk.
    combine_text_under_n_chars=2000
)
```

Note: You'll notice that a folder called 'figures' is created, where all the extracted images are stored. To make these images more useful, we can generate embeddings for them, which can later be used for semantic search. Typically, embedding models are specialized–they either handle text or images, but not both. This means the vector lengths won't align, and using different models for text and image embeddings can complicate similarity searches. However, the CLIP model can generate embeddings for both images and text, making it a versatile option for this purpose. To simplify things, we can first create a text summary for each image, and then generate embeddings based on those summaries. This approach allows us to leverage text-based semantic search while still incorporating the visual information from the images.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

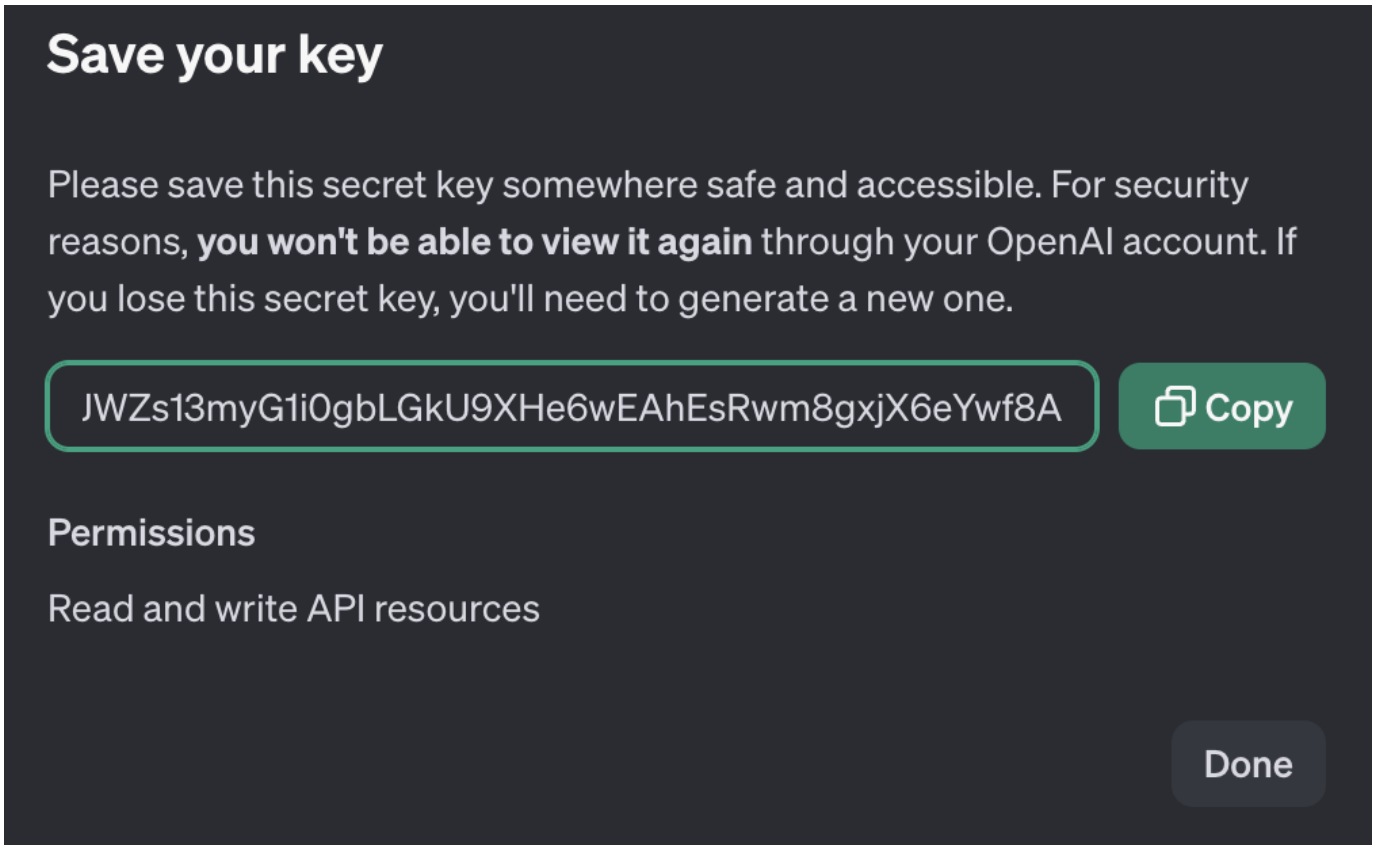


Note: After extracting images from a PDF or any other source, you can use the below method to send them to a multimodal model like GPT-4o to understand and generate insights about the images. This approach allows you to combine textual and visual data for a richer understanding of the content.

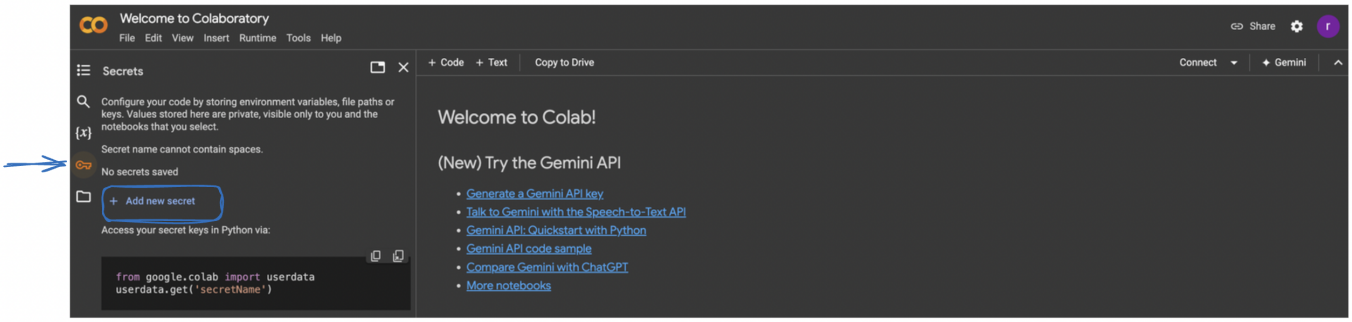
SET OPENAI TOKEN - MULTIMODAL

Note: First, create an account from the OpenAI official website.

- Create a new project API key by browsing to API Keys web page. Select Create new secret key. The API key is automatically generated. Save the APi Key as we will be using it in the later steps .

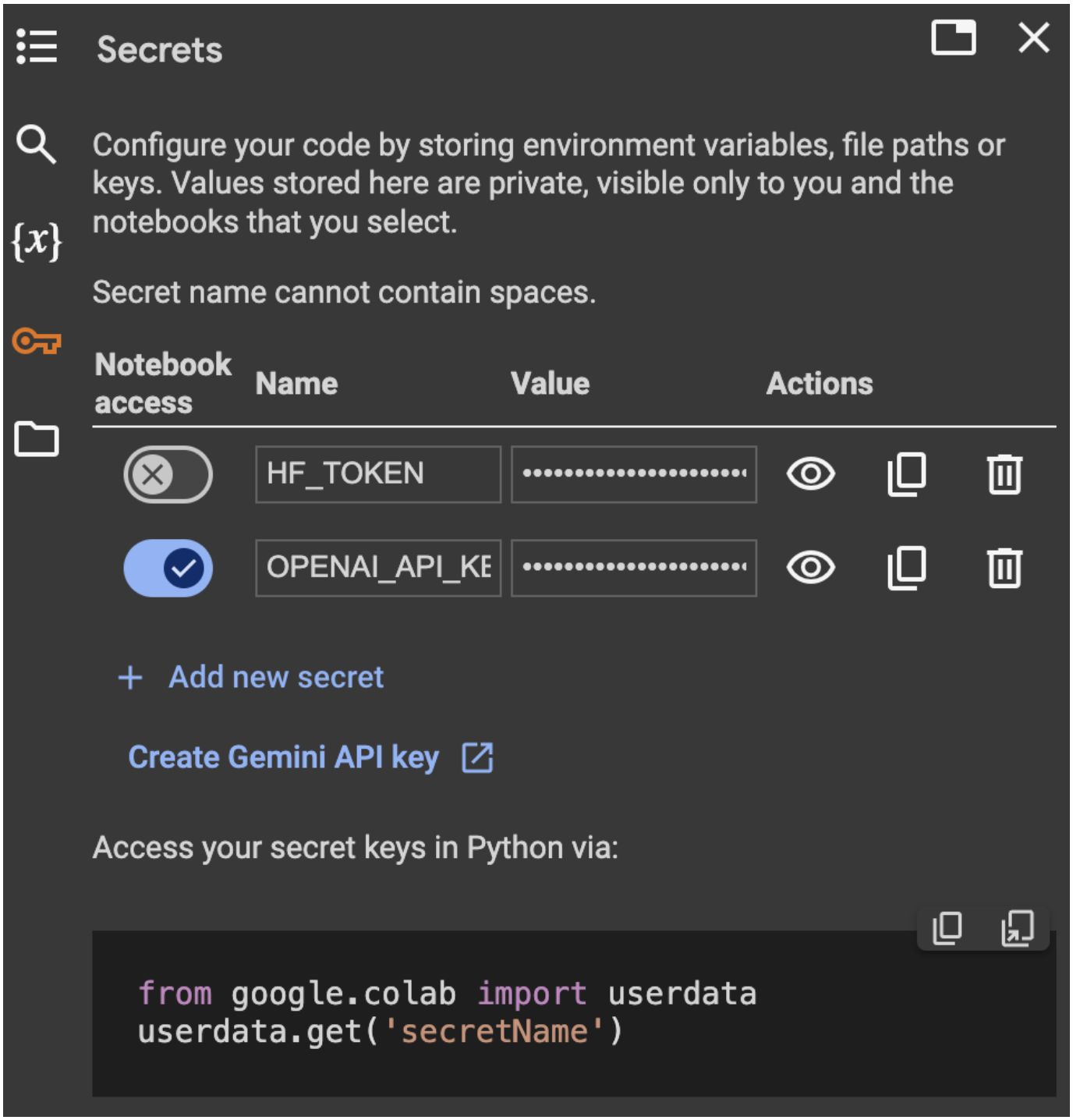


- Within your existing Google Colab notebook navigate to the new “Secrets” section in the sidebar.



- Click on “Add a new secret.” Enter the name example: OPENAI_API_KEY and value of the secret(the API key created above). Note: The name is permanent once set.
- The list of secrets is global across all your notebooks.
- Use the “Notebook access” toggle to grant or revoke access to a secret for each notebook.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Lets import and load the envoirnment variables

```
from langchain.chat_models import ChatOpenAI
from langchain.schema.messages import HumanMessage
import os
from dotenv import load_dotenv
from PIL import Image
import base64
import io
load_dotenv()
```

Incorporating Secrets into Your Code - We will use it later in our lab

```
import os
from google.colab import userdata
os.environ['OPENAI_API_KEY'] = userdata.get('OPENAI_API_KEY')
llm = ChatOpenAI(model="gpt-4o")
```

Function to Convert Image to Base64:

```
# Function to convert image to base64
def image_to_base64(image_path):
    with Image.open(image_path) as image:
        buffered = io.BytesIO()
        image.save(buffered, format=image.format)
        img_str = base64.b64encode(buffered.getvalue())
        return img_str.decode('utf-8')

image_str = image_to_base64("/content/figures/figure-1-1.jpg")
```

Note: This is the image that was extracted from our data and saved in the figures folder.

```
print(image_str)
```

Output: Our base64-encoded string


```
/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/
2wBDAQkJCQwLDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjL/wAARCAB1AOkDASIAAhEBAxEB/
8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/
8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTl
8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKk
HtL/un+VeNjpXqZbCMuZtHzmfVZx5Ixdtz1fQrx9R0a3uZBh2BDY7kHGf0rUrD8Jf8AItWn/Av/
AEI1t9K86qkqkku57mGk5UYSe7SHUUZpKg3FooooAKKKACiiigAooooAY5wpPpXk13rN7dXxuftDoQ3yBWICjsMV6vL/
q2+leMn7x+tenlkIycm0fO59UnFQUXbc9U8PXsmpaLBcSgeYcq2O5Bxn9K1RzWB4M/5FuD/AH3/APQjW/
0rgrJRqSS7ntYSTlQhKW7SHUUlLWZ0BRRRQAUUUUAFFJuHrS0ANP8AKue8SeIG0VY44Yw88oJG77qgd66GuC8e/wDIQtf+uZ/
nXThKcalZRlsefmdadHDSnB2ZpeHPFEuqXJtLmNBLtLI6cA47YrrK8y8Hf8jFF/uN/KvSx2p42lGnVtHYzynEVK+H5qju07DLn/j1l/3T/KvGe1ey3P8Ax6y/7h/lXjXauzK9pfI83iD4qfz/AEPT/CX/
ACLVr/wL/wBCNblYfhL/AJFq1/4F/wChGtyvOr/xZerPdwn8CHojnvEmvto0MaQxq88udu48KB3PrVPw74ol1S6+y3UaLKVLIyZAOOxFZ3jz/j/tP+ubfzrN8Jf8jHb/AEb/
ANBNd8cPTeF52tTxKuOrRzD2Sfu3Sseo0UUV5Z9IFFN3D1p1ACVgeI/EA0WJEjj8yeUEqD0AHc1v1wXj3/j+tP8Acb+YrowlONSqoy2ODMq86GHlOG5c0DxXNf3y2l4iB5M7HjBAzjOCK7HFeU+Gf+Rjs/
8AfP8A6Ca9VrTHUo06iUVbQwyjEVK9Fuo7tMST/Vt9K8XP3j9a9ok/1bfSvFz94/WurK95fI8/iD/l38/OPS/Bv/Itwf77/wDoRrf71geDf+Rbg/33/wDQjW/
Xn4j+LL1PcwX+7w9EYXiLXf7FtUZEDzSkhFPTjqTWX4f8Vz6hfrZ3scYaTOx4wQMjnBBJqDx/9+x/7af+y1heGv8AkY7L/fP/AKCa7qWGpywrm1rqeLicdWhj1Ti/
dulb1PVRS0lLXln0gUUhIoyPWgLnj02p3txctcPcy+YW3ZViMfT0r07Qrt73RrWeXmRl+Y+pBxmvJq9T8L/8i5Z/7p/ma9jMqcYwjZHy2R1ZyrTUne6v+JsmuB8e/wDIQtf+uZ/nXfGuA8e/8hC1/wCuR/
nXHgP46PUzn/dJfL8yj4O/5GKL/cb+VemV5n4O/wCRii/3G/lXplXmX8b5GWQ/7s/Vkdz/AMesv+4f5V4z2r2a5/49Zf8AcP8AKvGe1b5XtL5HFxB8VP5/oen+Ev8AkWrX/gX/AKEa3Kw/CX/ItWv/AAL/
ANCNbledW/iy9We/hP4EPRHA+Pf+Qhaf9c2/nWb4S/5GO3+jfyNaXj3/AJCFp/1zb+dZ3hL/AJGOD6N/KvXp/wC5/
JnzFf8A5GnzR6fUcz+XE74ztUnFSVDdf8esv+4f5V4i3PrJu0WeS3GqXt1cNcyXMokY5+VyNv09BXpfh+6kvtDtZ5WzIykMfUgkZ/SvKB0FeoeE/wDkXLT6N/6Ea9fMIRVKNl1Pmclqzl1JqTvdfqbhrg/H3/
H9Z/8AXNv5iu8NcH4+/wCP6z/65t/MVxYH+Oj1c4/3SXyMbwz/AMjHZ/75/wDQTXqgryvwx/yMdl/vn/0E16pW2Z/xV6HNkP8AAl6/ohJP9W30rxc/eP1r2iT/AFbfSvFz94/Wtcr3l8jm4g/5d/P9D0vwb/
yLcH++/wD6Ea3+9YHg3/kW4P8Aff8A9CNb/evOxH8WXqz3MF/u8PRHD+P/AL1h/wBtP/ZawfDX/Ix2X++f5Gt7x/8AesP+2n/stYXhn/kY7P8A3z/6Ca9bD/7m/RnzOM/5GS9V+h6qKbI21GPoM06mS/6t/
wDdNeItz66WkTyO61W9u7t7h7iQOxyArkBfYVL/AG/rH/P9J+f/ANas7tS19PGlTcVdI/PZ163O2pP7zuZ/AkL3fmR3TJATkxBckewP/
wBauqtLWOztY7eFdsca7VHtU4o6187Ur1KiSm72Pu6OEo0JOVONmxO9Y2uaDDrcS73Mc0edjjnGexFbVFZwnKEuaL1NatKFWDhNXTOe0LwxDo8jTtKZZyCobGAB7CuhFFLTqVJVJc0ndio0KdGHJTVkQ3P/
AB6y/wC4f5V4x2r2l1DoVPQjFeez+CdQS6KwPE0BJw7HG0e4/wAK9DL60KfMpux4udYWrW5HTje1zqfCX/ItWv8AwL/
0I1t9ao6XYrp2nQ2ifMI1wT6nqT+dXh1rz6slKba7nsYeDhSjGW6SMjXNCh1mBVdzHKmdjgZxn1HcVW0PwvHpErTvL585G0Nt2hR7CuhPTrQOnWqVaoociehMsJRlV9q4+93FprDIII4NOorM6TjbnwLDLeGWG5M
eP1r2dxlT7159deCtQW8K27RPAzEh2bBUe4/wrvy6tCm5c7sePnWFq1lB043tc6Twb/yLcH+8/wD6Ea3+tUNIsF0zTYbVTu2DlvUnk/
rV+uGrJSqSkurPWwsHCjGEt0kZetaLBrNqIpSUdDlJF6qf6is7RPCkelXP2qSczTDIQhdoXPtzzXTGk7VSrVFDkT0FPCUZ1VVlH3l1FoIyMUtFZHScdeeB4J70yw3LRRMctHtzj6GrH/CDaZ6z/
wDfyunpa6FiqqSXMcH9m4a7fItRaKKK5zvCiimk4+lAAeKBzWTdeJNJtWKSXiFgcFUy3P4VWXxjo7HBuHX3Mbf4VoqFRq6izmljKEXZzV/U3/
wpe1UbHU7PUVY2s6S7cbtp5GfUVezxUNNOzN4yjNc0XdCDpR0HSsy51/
TbOd4JrpElT7ykHjjNS2Wq2WolxaTrKUxuwDxmm4TSvbQhV6TlyKSv2uaFFJmkLAVJqB4oHNY9z4n0m1Yo92rMDgiMFsflUCeMNHc4Nw6+7Rt/
hWioVWrqLOaWLoRdnNX9Tf8Awpe1U7LULXUEaS1nSVAcEr2NXBUNNOzN4yUleLugpD9KM1Tu9RtLEA3NxHHnpubBP0FCTbsgIJRV5OxcPFA571gt4w0dGx9oZvcRtj+VTW3ibSbp1jju1DscBXBXJ/
GrdColdxZhHGUJOymr+ptUUUZrM6RKafwqO4uIrWJpZ5FRF5LMcAVjy+LdHjbH2ksf9lGI/lVxpzl8KuY1MRSp/HJL1NwH3FOrCh8V6PMcC7CH/popUfmRitmKZJkDxsGUjIZTkGiVOUfiVh069Op8Ek/
QloooqDUKKKACiikoAjlkWKJpHIVVGST2Fea674kudUneOGRorQHCqDguPU/4V1PjS7aDRfLQkGZwhx6ck/
yrzkDJAA9sV6uX0ItOpL5HzOd42amqEHbuA5IA6ngVO1ldom5rWdV9TGQP5V6NoOg2+mWqM0atcsoLueSD6D0Fbe1ewGaqpmajK0Y3RFDIXKClUnZs4rwCfmvv+2f/s1dxjiq8VtDDI8kcSI743MoALY6Z/
OrB6V5tar7Wo59z38Jh3h6Kpt3seV+Kf8AkZLv6r/6CK2vAP8Ar7//AHU/9mrF8U/8jHd/Vf8A0EVteAP9fff7qf1r1q/+5/JHzOE/5Gfzf6ncYwRXB+M9XmN7/
Z8MjJEigybTgsT2Ptiu9PNeZ+LrZ4PEEsjAhJlVlPY4GD/KuDARjKt7x7GdVJww/udXqY1vbT3UwigiaSQ9FUc1pHwtrQTd9ib6B1z/
ADqrpOpSaVfC5jRXOCrK3cHGfp0rtrLxpp9wQlwr27ercr+Yr08TVxFN/
u43R4WAw+ErRtVm1Ib4MsrmytLmO5heJjLkBhjIwK6roKiilSVA8bqyMMgqcgipGOFJ9K8SrNzm5PqfW4elGjSUIu6RzPifxEdNjFtakfaZBknrsHr9a8+mmluZWlmkeSRurMck1PqN219qNxctn945I9h2H5Vse
b/ABr2qUIYWjzy36nyeIq1swxPs4PTp29TCitp7jPkwSSY67EJ/lU9lG8WrWiSoyN56cMMH7wr1qOCOFAkcaqo6BRgCmyW0E2PMiR9pyNyg4Ncssyvdcuh6MMh5bNT1XkSqOB9Kzta1aLSLFp3wzH5Y0/
vN6VpV5z41vHm1gWxb93AgwP9o8k/yrjwtH21VRex6eYYl4bDua32Rjahqd3qc5lupi/ov8K/QVWSN5W2xozt1woyav6Hpn9rapHbsSIgN0hHXaP/
AK+K9QtbK3soFht4URF7Afzr1cRi4Ya0II+cweX1MdepOVl33bPH3R42KujKw7MMGr+l61e6VMrQSEx5+aJvun/CvTNQ0u11G3MVxErA9D3H0PavLdRs20/
UJ7VjkxtgHGMjqD+Rp0MTDFJwkhYvBVcvkqkJadz1TTdQh1GyjuYfusOh6qe4PvV3PWuD8C3pW5uLMn5WXzB7EcH+Y/Ku8ryMRS9lUcD6jA4n6xQjUe/UWiiisDsCiiigDkfHULHS4ZBnCy8/
iDXBI2x1bGcEGvXNVsI9S06a0k4Djg+h7H868pvLOexupLeddsiH8CPUe1e1llSMqbpvdHyeeUJQrqstn+Z61Y3Ud5ZxTwtlHUMKsivKNL129OjKwOrRE5McnI/D0rafx5cmPCWSB/
UuSPyxXJUy+qpe7qj0aGdUHTTqaM72l9a5fwnrN5q0t2bllwmzYqjAGc5/lXUVx1KbpycJbnq0K8a9NVIbM8r8U/8AIyXf1X/0EVteAP8AX33+6n9axfFP/IyXf1X/ANBFbXgH/X3/APup/wCzV7Nf/c/
kj5XCf8jP5v8AU7mqGqaXa6pb+TdJuAOVYHBU+xq+a5XUfFw07U57N7QuIyMMrdcgHpj3rx6MJzl+73R9PiqtGnD998L0Mu+8D3MQZrO4WVeoVxhvz6fyrmbmlns5jDcRtG46q3+ea7T/AITy1x/
x5TZ+q1y+t6s2sXonMQiVF2KucnHXk17WFlie1VaHy2Pp4FR5qEtexr+DNTmjvxYO26KQEqD/CwGeK79xmNh6ivOfBlnJPri3AU+XApLN2yRgD9T+VekY4rzseoqt7p7WTOcsL7/
AHdvQ8YliaCZ4nBDoxUg+oOK7HwLdxolzaMwEjOJFz1YYwcf1+tU/F+iSWt2+oQqWgmOXx/A3+BrmoZpLeZJYXZJEOVZTyK9NqOKoaP/AIc8BSnl+MvJbfij2fIxQfWvPrbxzeRR7bi2imI/
iB2H8etOj8Y6jeahbwokcEbyqpwNzYJHc/4V5TwNZbo+hjnOGaVm7vyO/rzLxdGyeIpiejqrD6Yx/
SvTR92uZ8W6K+oWq3MC7poQcqBy6+g96MFVVOsnLZ6FZth5V8M1HdanM+ELuO11wLIQBKhjBPY5BH8q9KVjk14uMqe4IP5Guhs/
GWoWsSxSqk6qMAtkN+YruxmDnVlzwPHyrM4YeDpVdujPRywC5NeVeIbtLzXbqaMgpuCqR3wAM1a1HxXqF/
G0S7YI2GCEPzEfWsNEaR1RAWZjgKByT6VWCwkqLc57kZpmMMU1SpbXOm8DwM+syTAfKkJBPuSMfyNeh4rB8MaOdK0797g3ExDv7eg/Ct/
Nebi6qqVXJbHv5Zh5UMPGMt9xaKKK5j0AooooAQ1maro1nqsWLiP51+668Mv0NFFOLad0Z1YRnBqSuee6too0648tZ969BlMH+dZYTPGaKK+koTk6auz4PE04xqtJaHeeCrMW0dxJv3GTbxtxjGf8a63tRRXhYt3
AGhe3VnZHpXmvim1/wCJ7cyb/vbeMdPlAoorky92qndnivh16mCV561r6PoS6ncBXuCig8gLnP60UV6+InKNJtM+ZwNOM6qUlc9D0/
TrbS7bybaMKg5Pqx9Sav8AaiivnZNt6n3kIRhBKKI5IkmjKOoZWHIIyDXF614StYkkuLaVogvJjI3D8OePloorbC1JRmrM4swo06lP3lc4502tjOa1NF08z6nbnzduyRW+7nODn1oor3qzapM+Qw8IuutOp6qPu0
2eVk37sHGcYoor2cBUk002fK5zRpweKsTWen/AGudY/NCBj125x+td9onh2z01fNUGWfH+scdPoO1FFTj6ktFfQvJqFNy5mtToBS0UV5B9QFFFFAH/9k=
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Initializing the Multimodal and sending the Image to GPT-4o for Analysis:

```
chat = ChatOpenAI(model="gpt-4o", max_tokens=1024)

msg = chat.invoke(
    [
        HumanMessage(
            content=[
                {"type": "text", "text": "Please give a summary of the image provided. Be descriptive"},
                {"type": "image_url", "image_url": {
                    "url": f"{image_str}"
                    }
                 },
            ]
        )
    ]
)
```

- Retrieving the Response:

```
msg.content
```

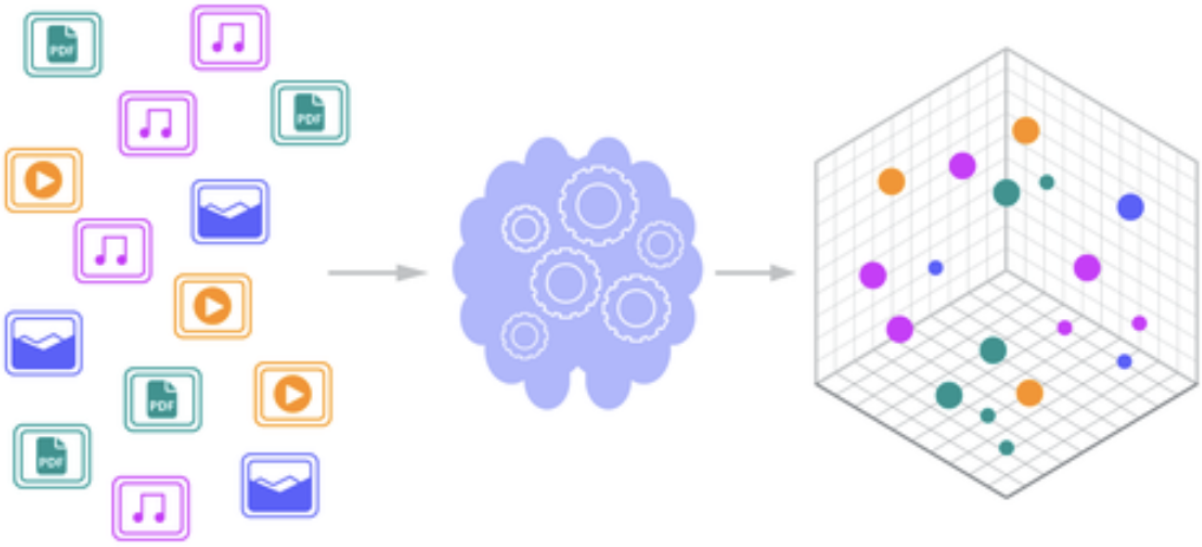
Output: The response from the model is stored in msg.content, which will contain the descriptive summary of the image.

The image is the logo of Cisco Systems, Inc., commonly known as Cisco. The logo features the company name "cisco" written in lowercase letters in a distinct, modern sans-serif font. Above the text, there is a series of vertical bars of varying heights, which is stylized to resemble the Golden Gate Bridge. The logo is rendered in a light blue color. The design is clean, professional, and easily recognizable, reflecting Cisco's identity as a major tech company specializing in networking hardware, telecommunications equipment, and other high-technology services and products.

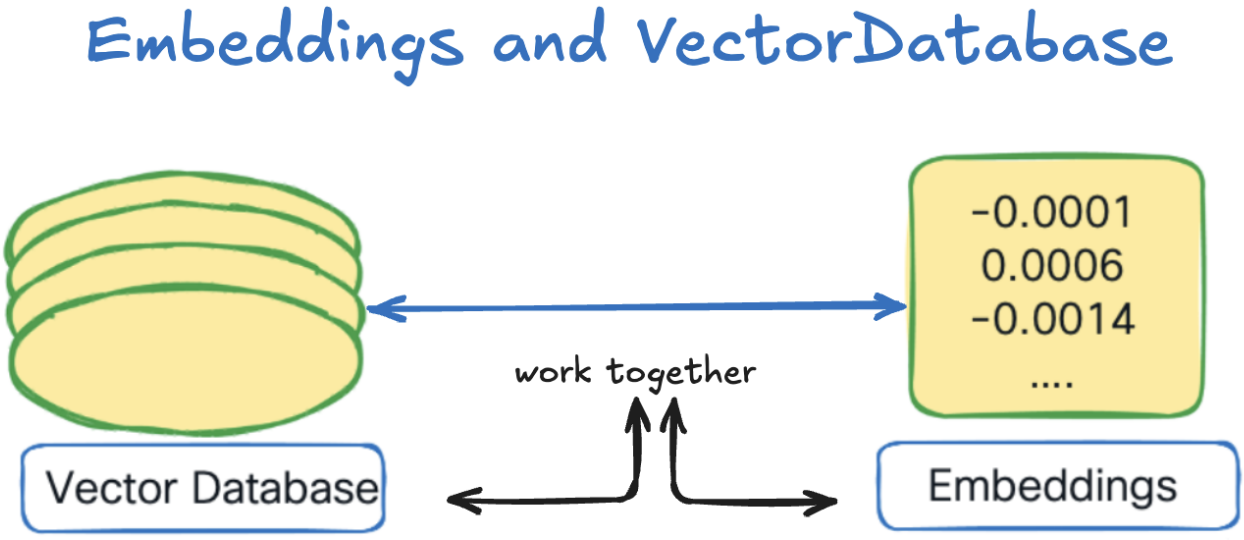
Note: The description of your image may vary depending on the image retrieved.

1.5 Task 4: Embeddings and Vector Database

- Vector databases are special databases meant to store content such as PDFs, Blogs, Word Documents, Images, Audio Files, and even Videos as embedding vectors, allowing for semantic based retrieval.



- In the process of a large language model, embedding is generally the first step to convert discrete tokens (like words or subwords) into dense vector representations, which will allow the rest of the network to do the math necessary to predict the next word.Embeddings and vector databases are essential if you are building any kind of AI product.



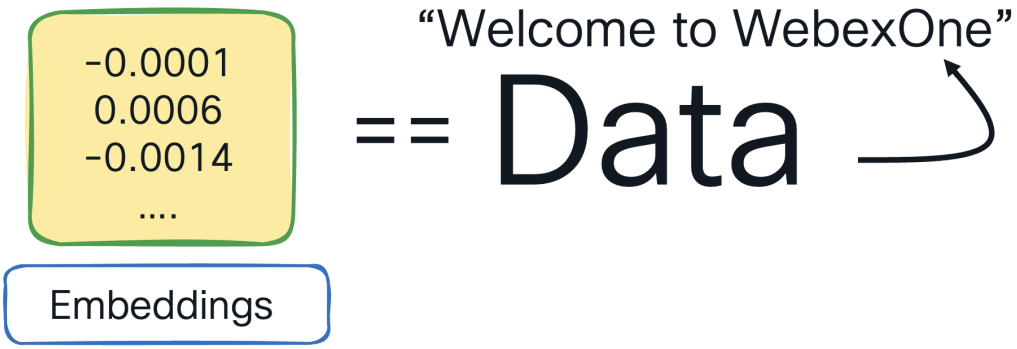
1.5.1 What are embeddings?

Embeddings are basically data.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Embeddings and Vector databases

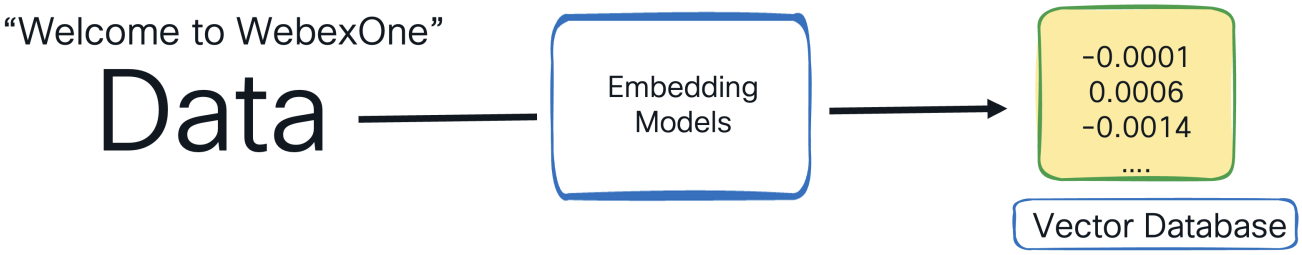
What are Embeddings?



That are converted into array of numbers called vectors thats contain pattern of relationship, and can be used for similarity search.

Embeddings and Vector databases

What are Embeddings?



Let me explain this concept using a 2D graph. In this graph, words like "Webex" and "collaboration" are often used in similar context, so their embeddings essentially vector representations are positioned close to each other. These vectors are numerical arrays that computers can easily interpret.

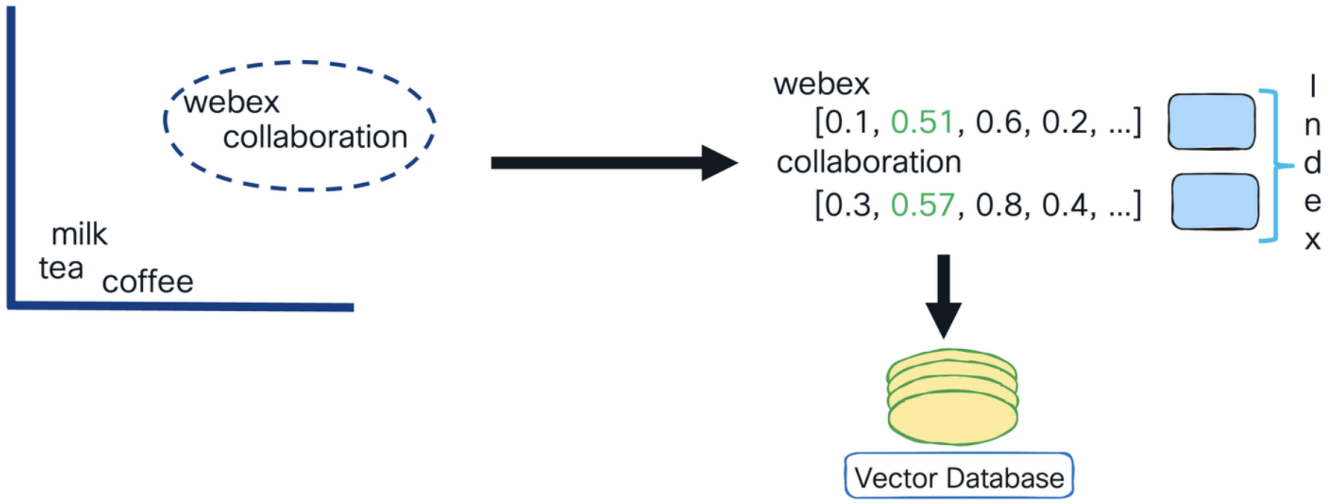
The advantage of using vectors is that you can find similar items by calculating the distances between their vectors, a process known as nearest neighbor search. However, simply storing these embeddings isn't enough. Performing queries across thousands of vectors can be incredibly slow. To overcome this, these vectors also need to be indexed in the vector database.

Note: In reality vectors can have hundred of dimensions

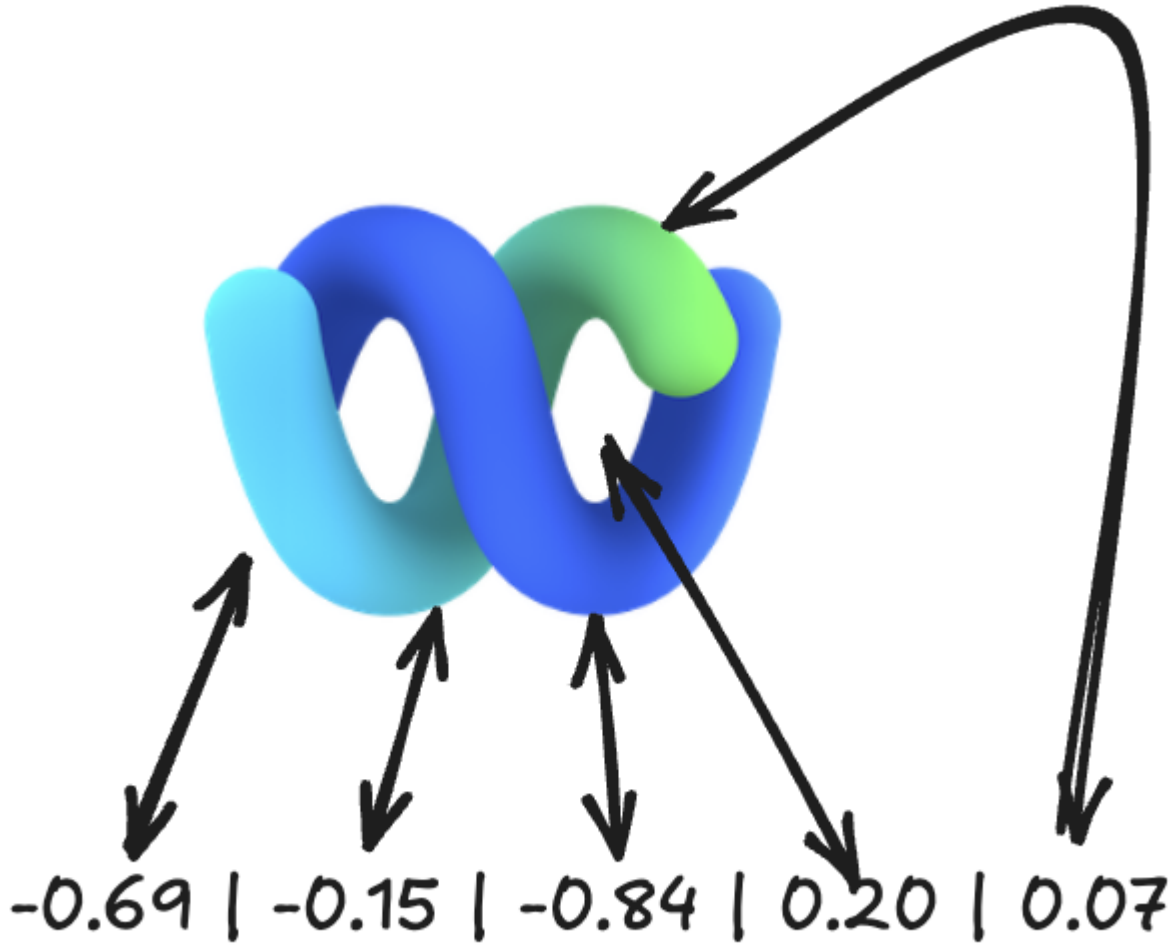
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Embeddings and Vector databases

What are Embeddings?



Similarly, images are also broken into vectors, which are arrays of numbers that machines can process. Once these embeddings (both for words and images) are generated, they are stored in a specialized database known as a vector database.

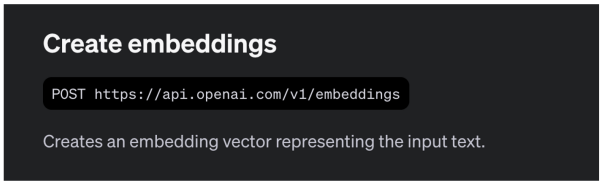


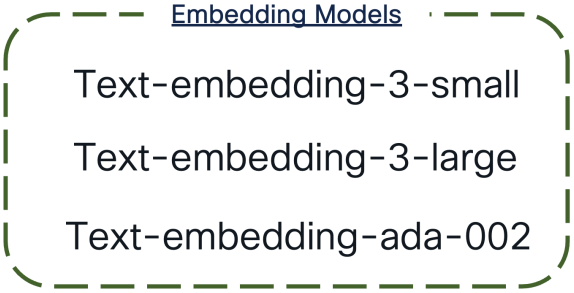
There are numerous embedding models available, such as Google's Word2Vec, CLIP (Contrastive Language-Image Pretraining), and even those provided by OpenAI, which offer excellent capabilities for generating embeddings. However, the challenge is that these models don't include tools for storing and managing those embeddings. That's where vector databases become essential.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Embeddings and Vector Database

OpenAI - Embeddings

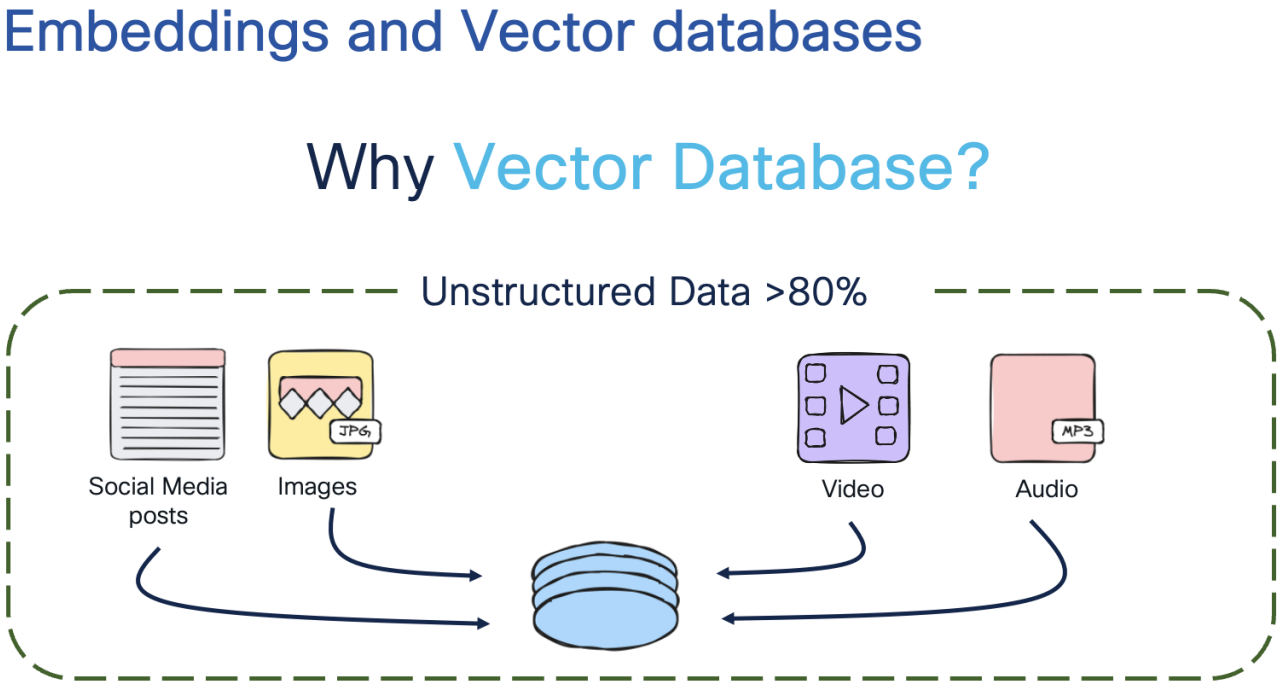




Note: In this lab session, we will be using OpenAI embeddings. More info can be found here

1.5.2 Why Vector Databases when we have relational DB's?

Around 80% of the data we encounter is unstructured, including social media posts, images, and videos. This type of data doesn't easily fit into traditional relational databases, which is where vector databases come into play.



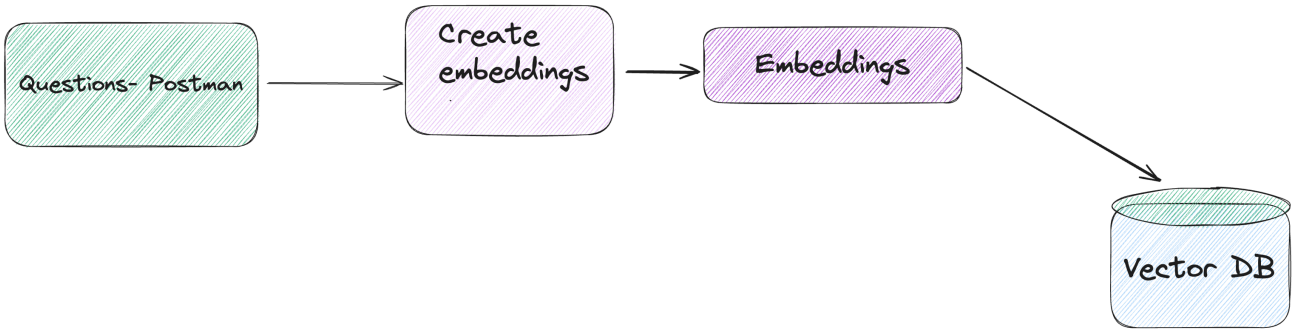
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.5.3 Understanding working of Embeddings and Vector Databases

Practical Example of Embedding Techniques Using Postman

Embeddings and Vector databases

High Level Overview



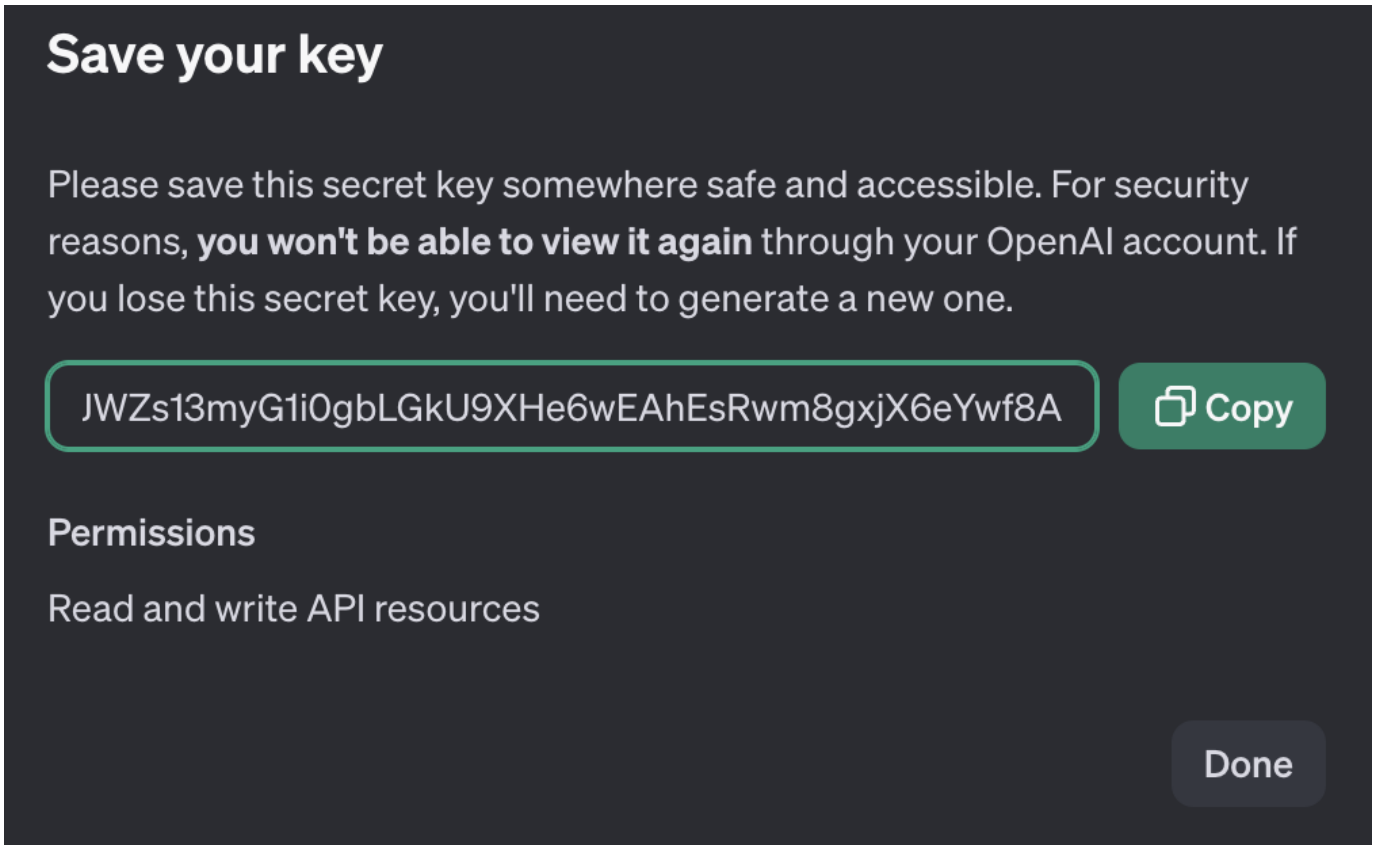
SET OPENAI TOKEN

Note: If you've already obtained the token in previous steps, you can skip this section

Note: First, create an account from the OpenAI official website.

- Create a new project API key by browsing to API Keys web page. Select Create new secret key. The API key is automatically generated. Save the APi Key as we will be using it in the later steps .

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Note: We will use the OpenAI key in Postman to generate embeddings.

LOGIN TO POSTMAN

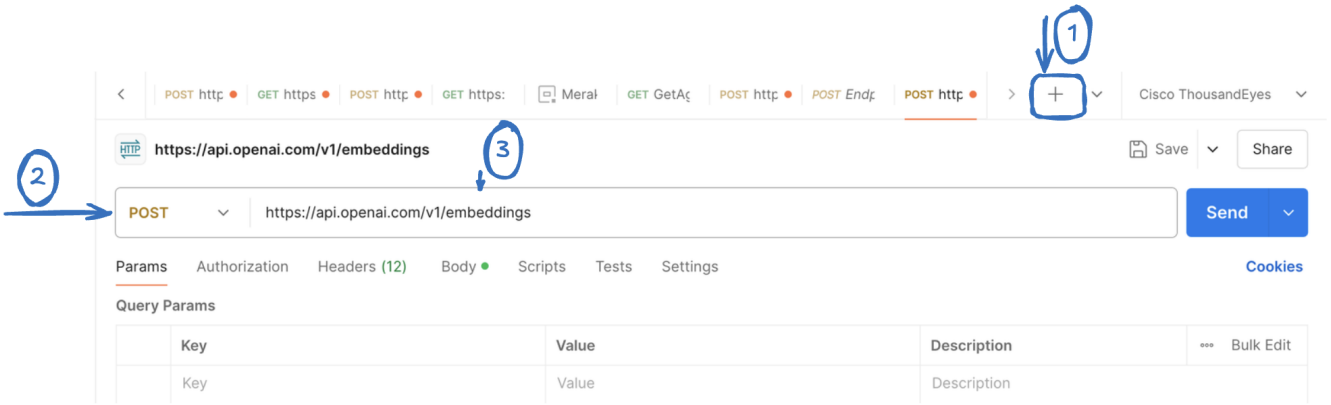
If you haven't installed Postman on your machine yet, you can download it from the following link

As we will be using embedding APIs to create an embedding vector that represents our input text, more information can be found at the following link

The POST request we will be using:

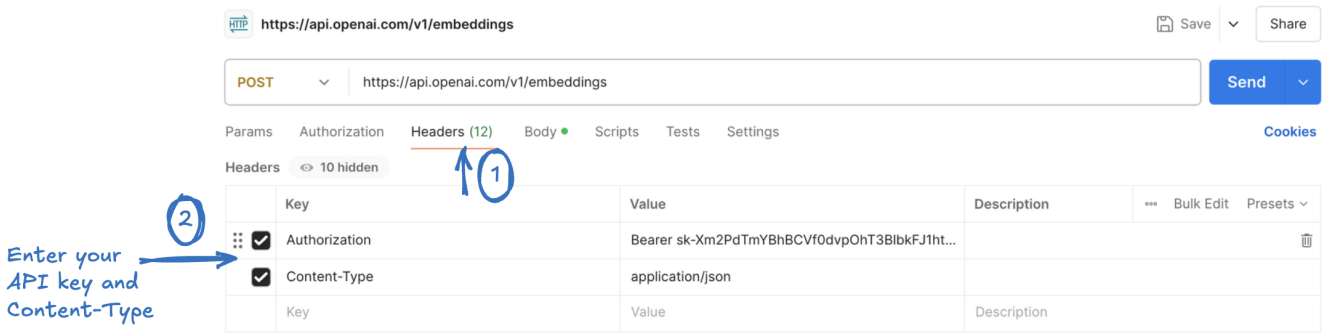
```
https://api.openai.com/v1/embeddings
```

* Open Postman, create a new request by pressing +. Select POST as your request and enter the request url

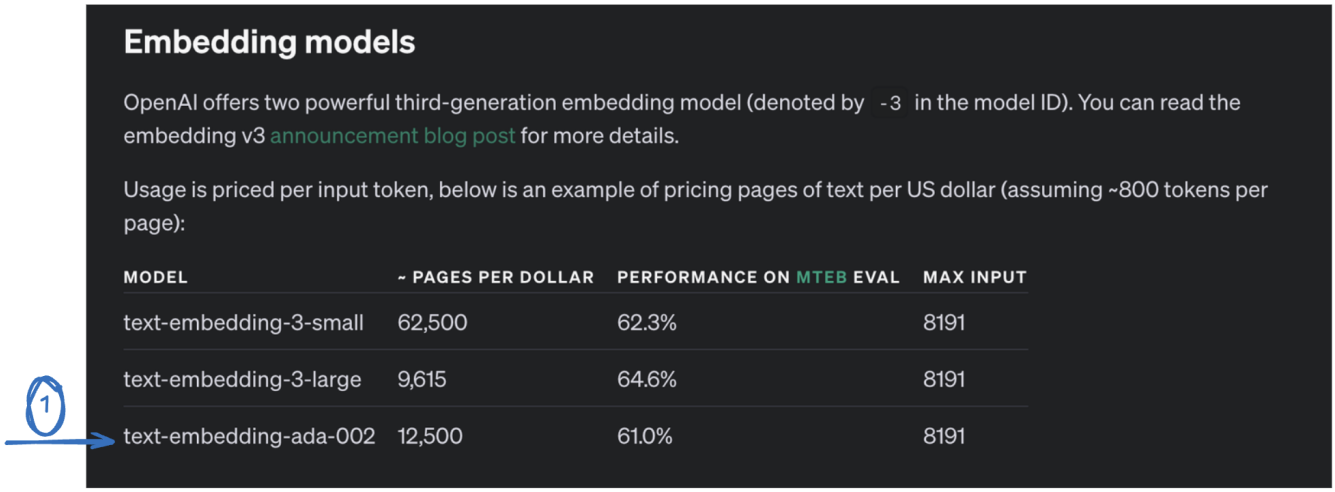


- Click on the Headers tab and enter the Following creds

```
Authorization: Bearer <replace with your openAi API key>
Content-Type : application/json
```

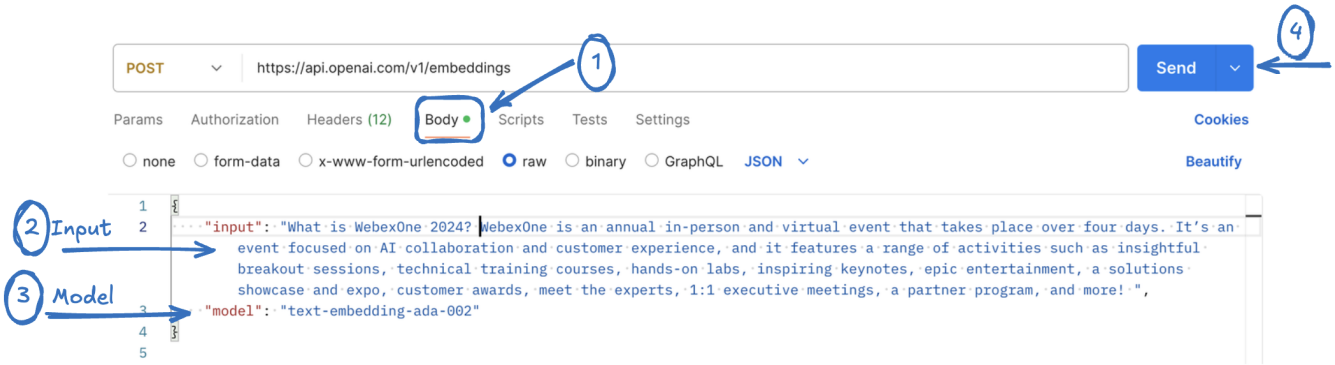


- In this lab, we will be using the text-embedding-ada-002 model, but feel free to use any other embedding model of your choice.



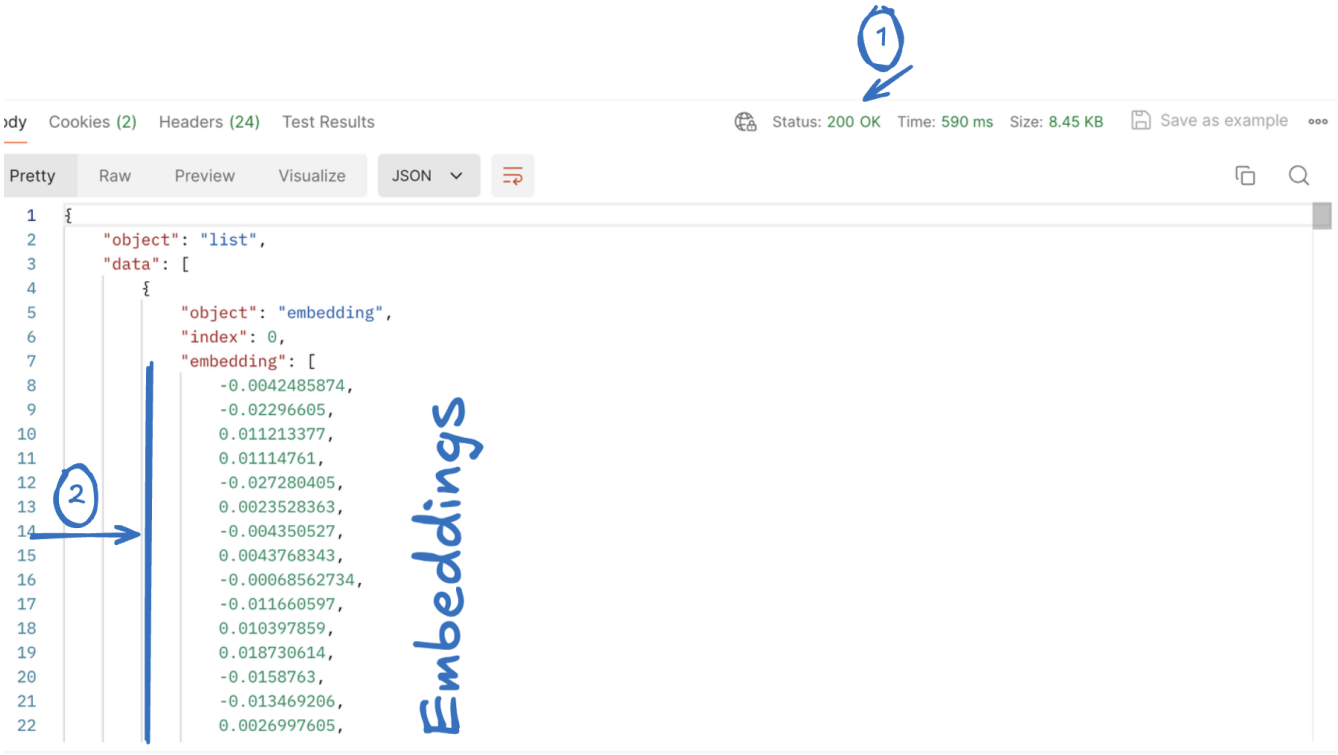
- In Postman, click on the "Body" tab, enter the following information, and then press "Send."

```
{
    "input": "What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days. It's an event focused on AI collaboration and customer experience, and it features a range of activities such as insightful breakout sessions, technical training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive meetings, a partner program, and more!",
    "model": "text-embedding-ada-002"
}
```



- You will receive a 200 OK message, confirming success, and you'll notice that our text has been converted into embeddings (vectors). In the upcoming steps, I will show you on how to manually save this information in a vector database.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Note: Text embedding models, convert text into numerical data(embeddings) that represent text meaning.

INSERTING VALUES IN VECTOR DATABASE

We have a variety of databases available. In the upcoming step, I will demonstrate how to use SingleStore as a vector database and save our embeddings there.

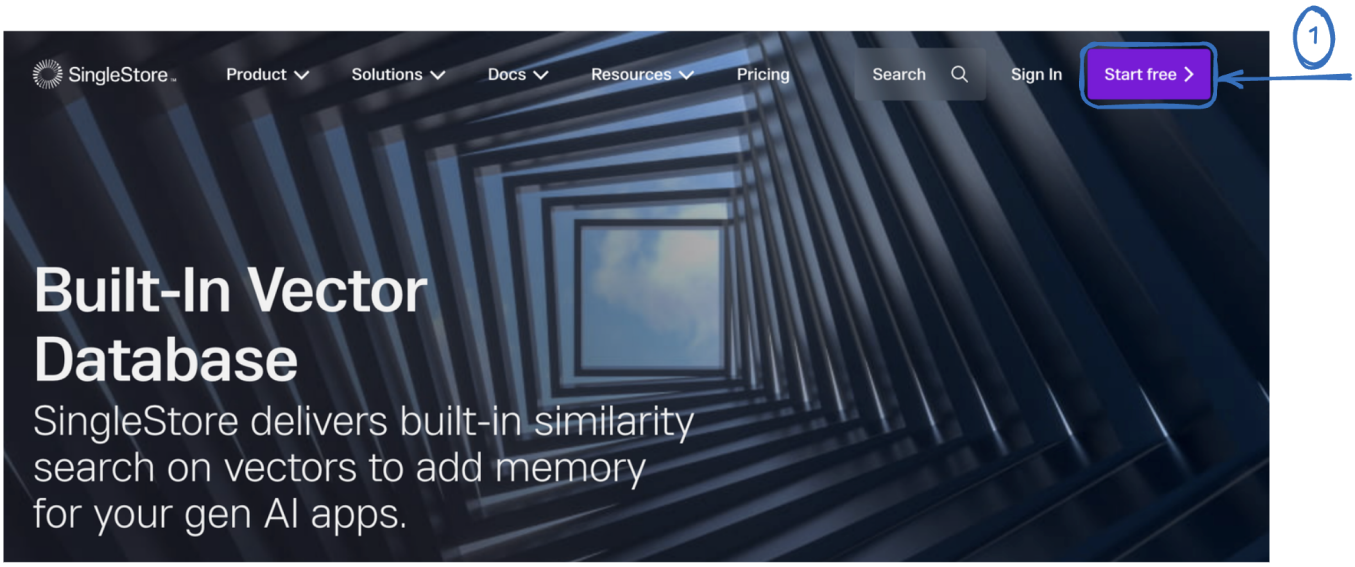
Note: This manual step is simply to illustrate how embeddings are stored in vector databases.

Embeddings and Vector Database

A few examples of Vector Database?

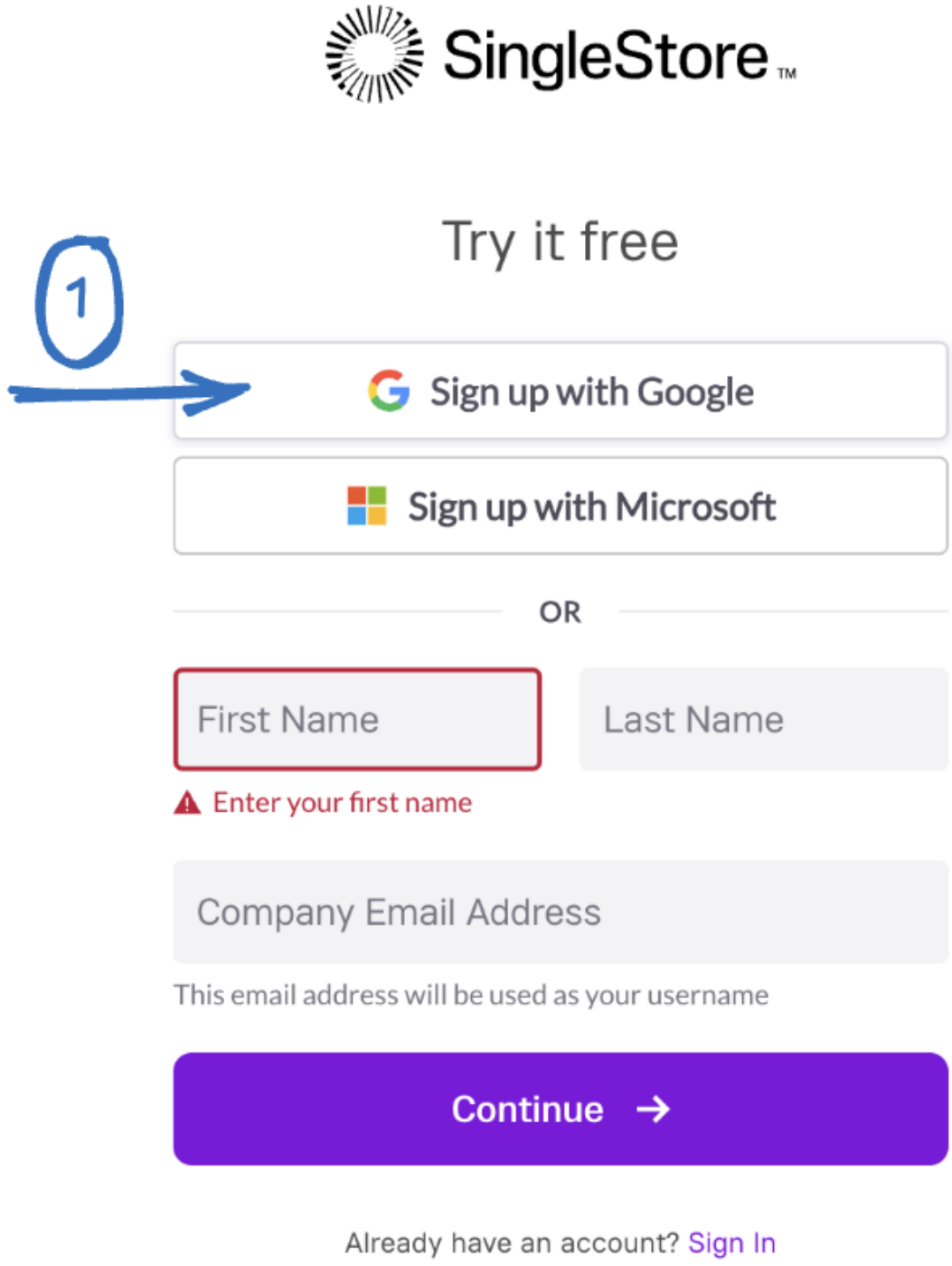
Note: We will create a Free account on SingleStoresince it provides some credits to set up a trial account.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

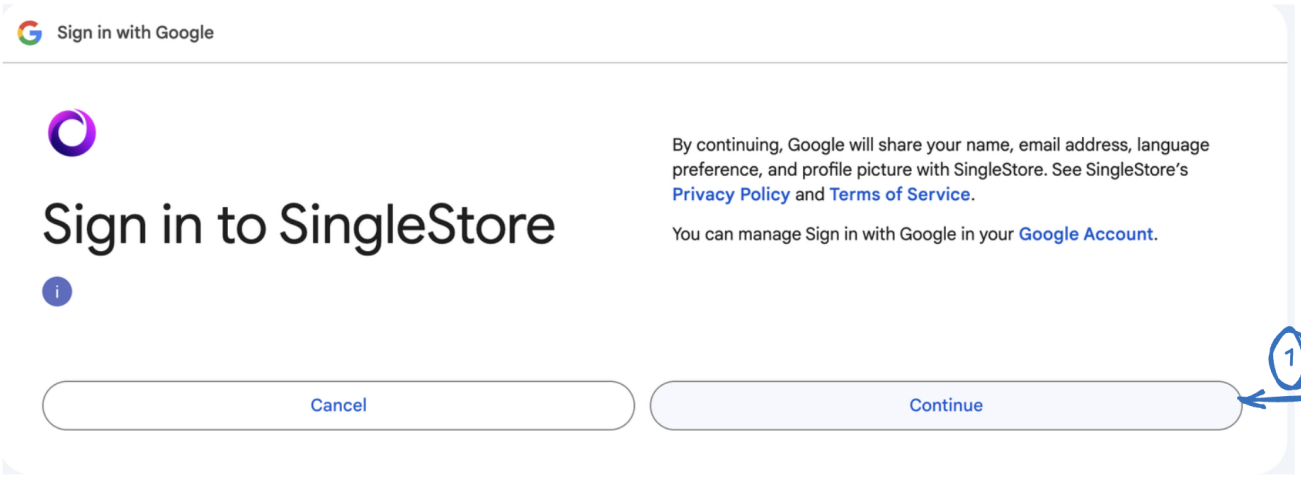


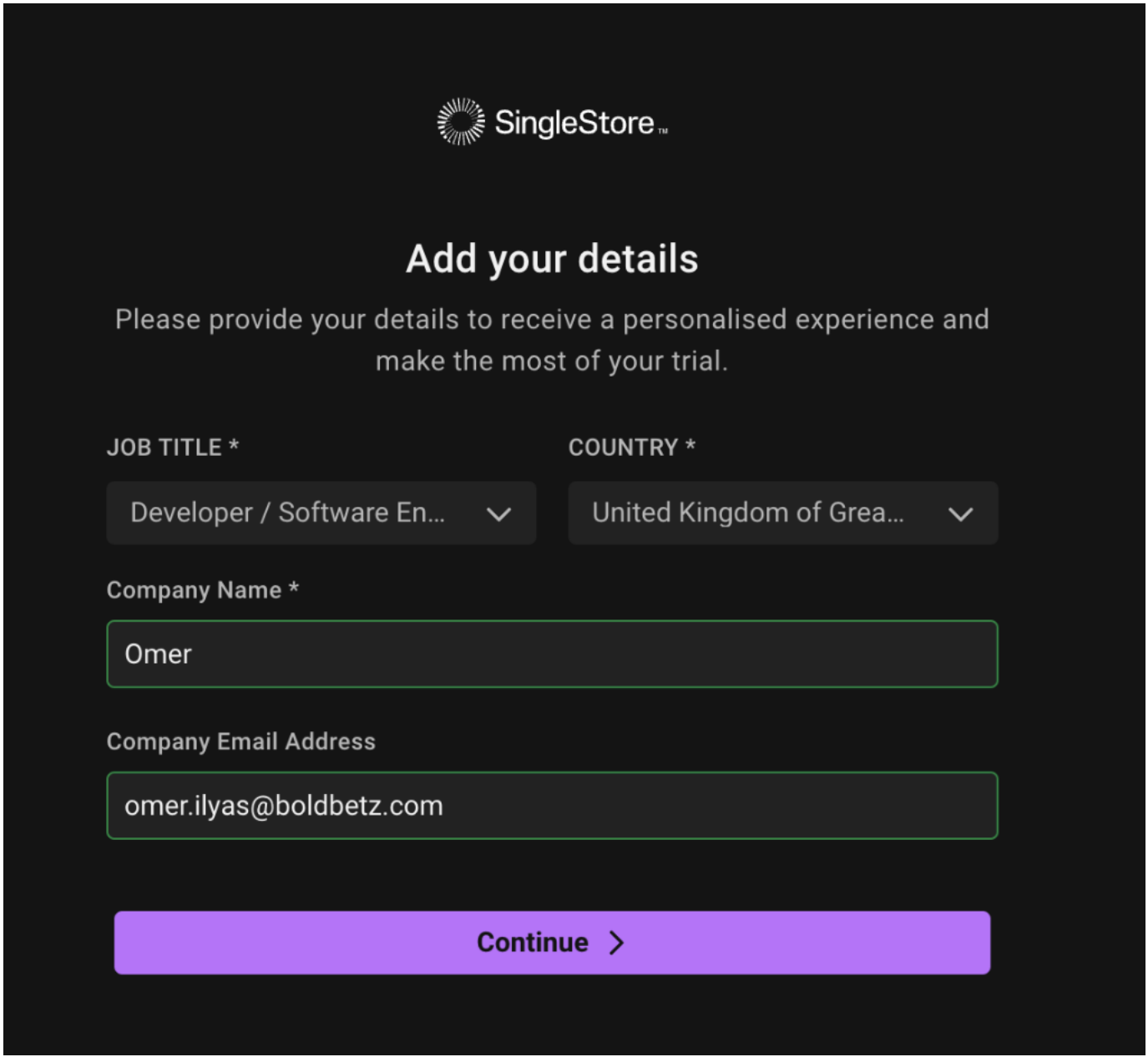
- I'm choosing to create an account using Google, but feel free to use any other method that works for you.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

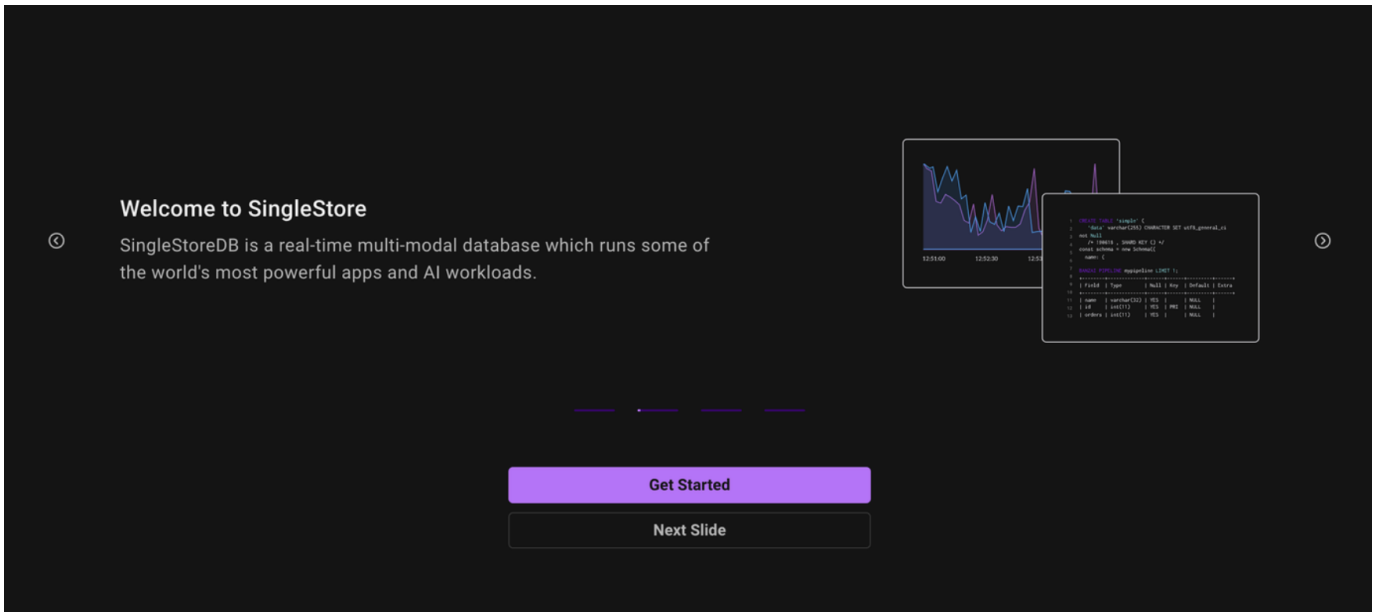


- Select Continue

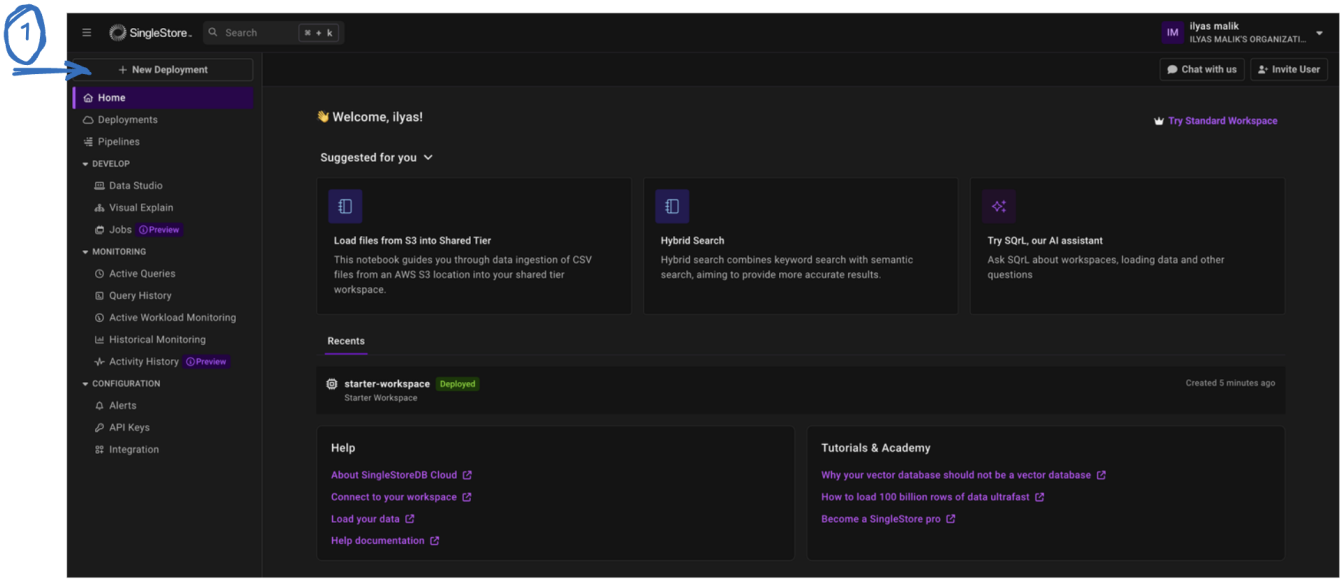




Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

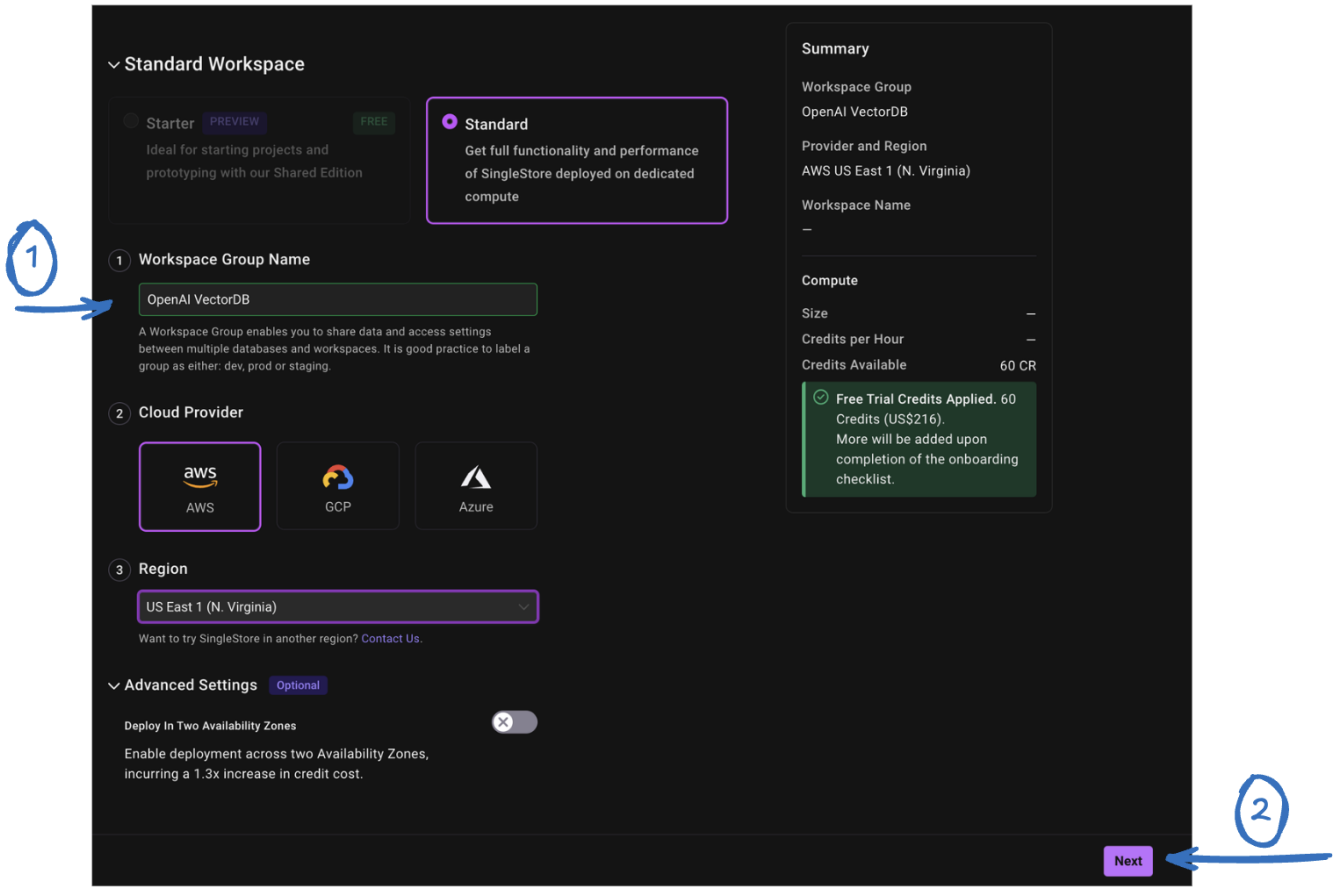


- Click on New Deployment



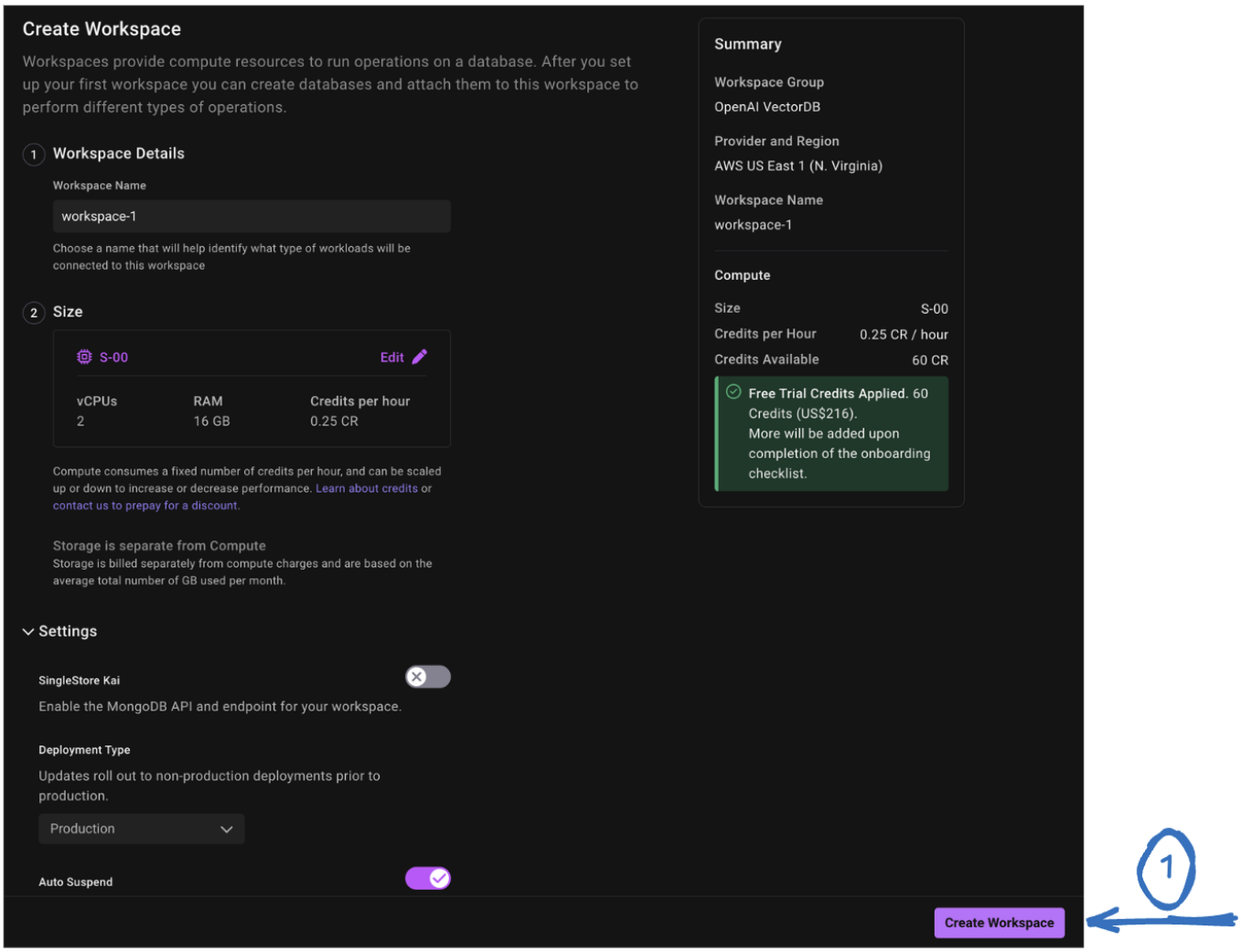
- Give your workspace a name, keep all other settings at their default, and click "Next."

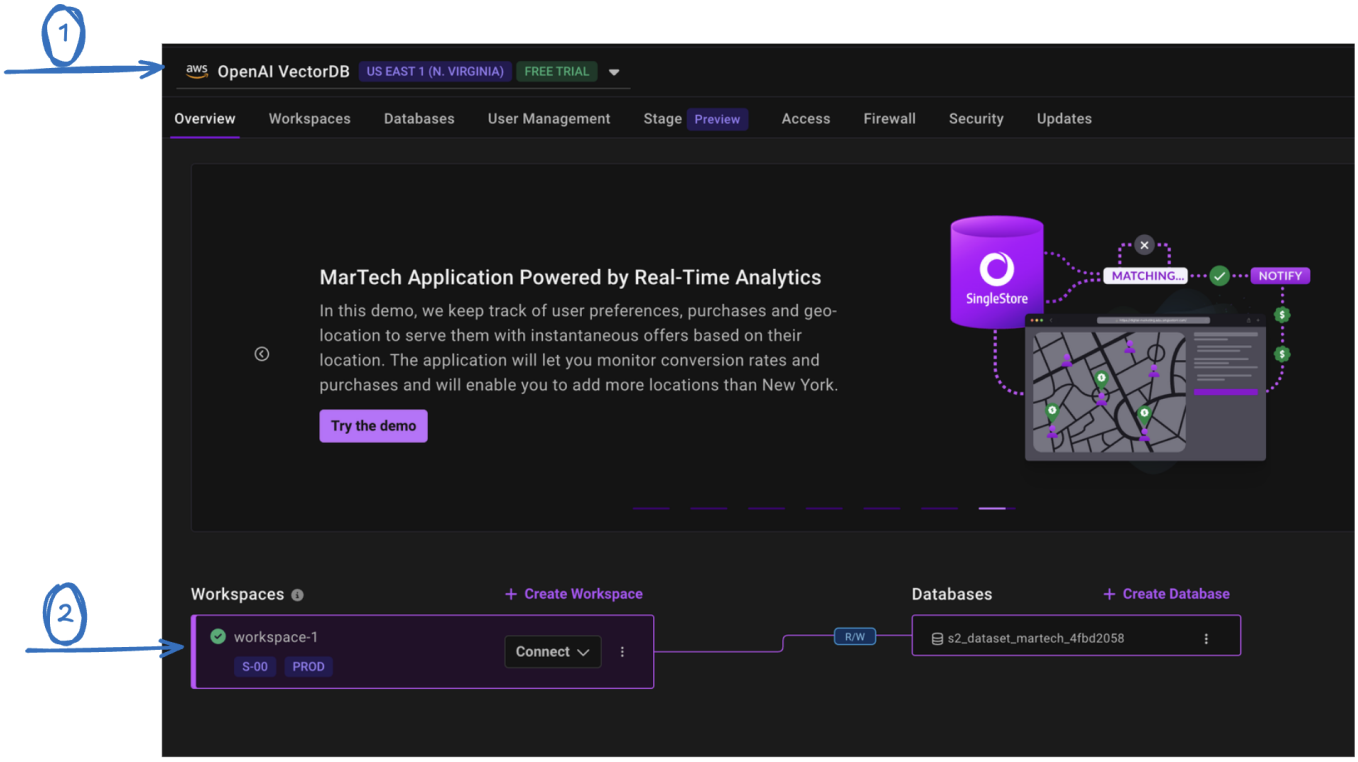
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Then, leave everything as is and click on "Create Workspace."

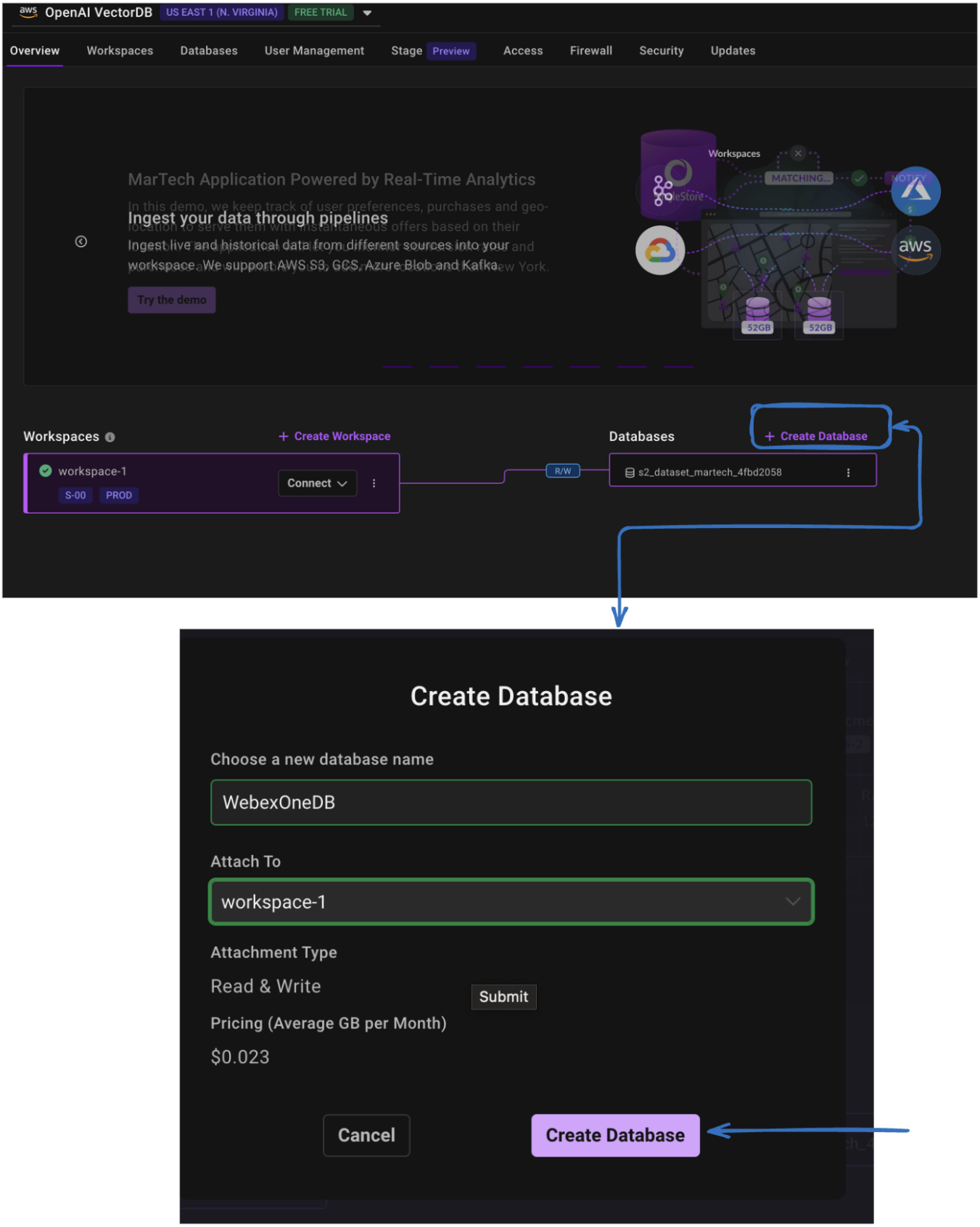
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com


Create Workspace
Workspaces provide compute resources to run operations on a database. After you set up your first workspace you can create databases and attach them to this workspace to perform different types of operations.
1 Workspace Details
Workspace Name
workspace-1
Choose a name that will help identify what type of workloads will be connected to this workspace
2 Size
S-00
Edit
vCPUs
2
RAM
16 GB
Credits per hour
0.25 CR
Compute consumes a fixed number of credits per hour, and can be scaled up or down to increase or decrease performance. Learn about credits or contact us to prepay for a discount.
Storage is separate from Compute
Storage is billed separately from compute charges and are based on the average total number of GB used per month.
Settings
SingleStore Kai
Enable the MongoDB API and endpoint for your workspace.
Deployment Type
Updates roll out to non-production deployments prior to production.
Production
Auto Suspend
Summary
Workspace Group
OpenAI VectorDB
Provider and Region
AWS US East 1 (N. Virginia)
Workspace Name
workspace-1
Compute
Size
S-00
Credits per Hour
0.25 CR / hour
Credits Available
60 CR
Free Trial Credits Applied. 60 Credits (US$216).
More will be added upon completion of the onboarding checklist.
Create Workspace
1

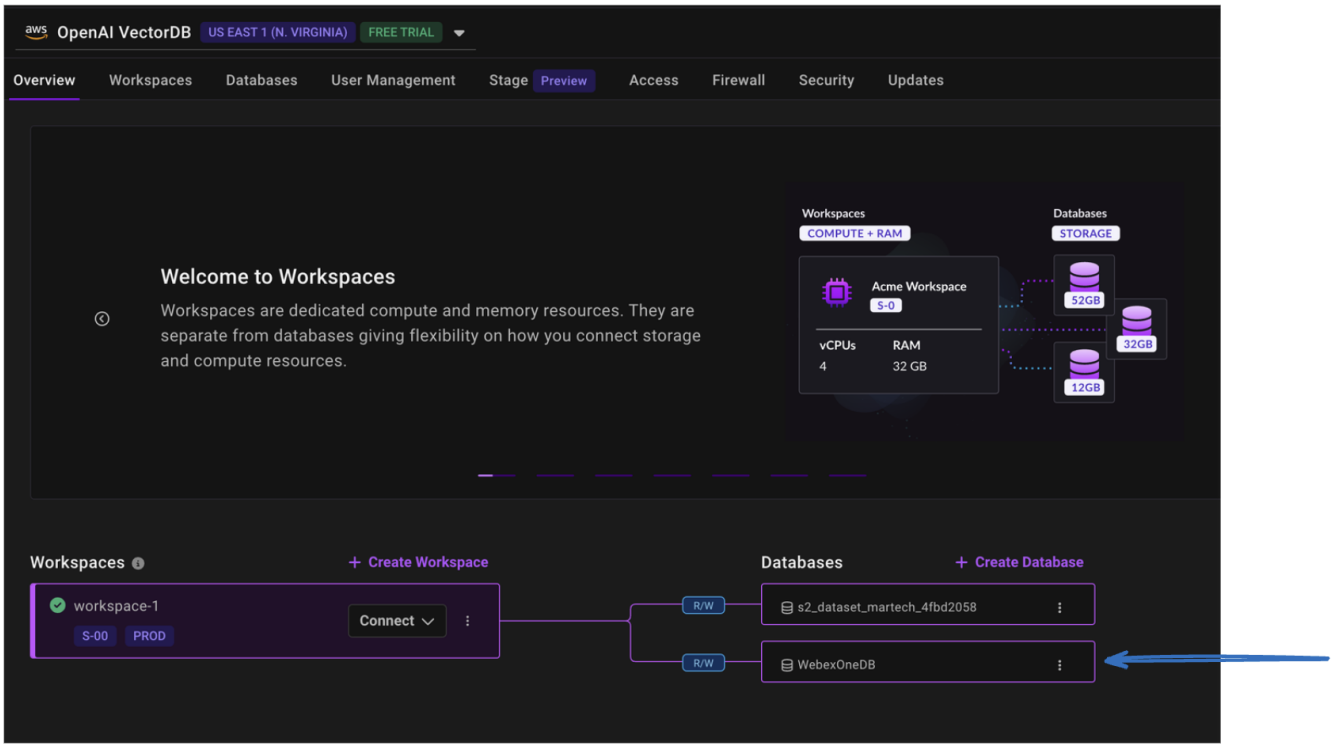


- Now we can create a database

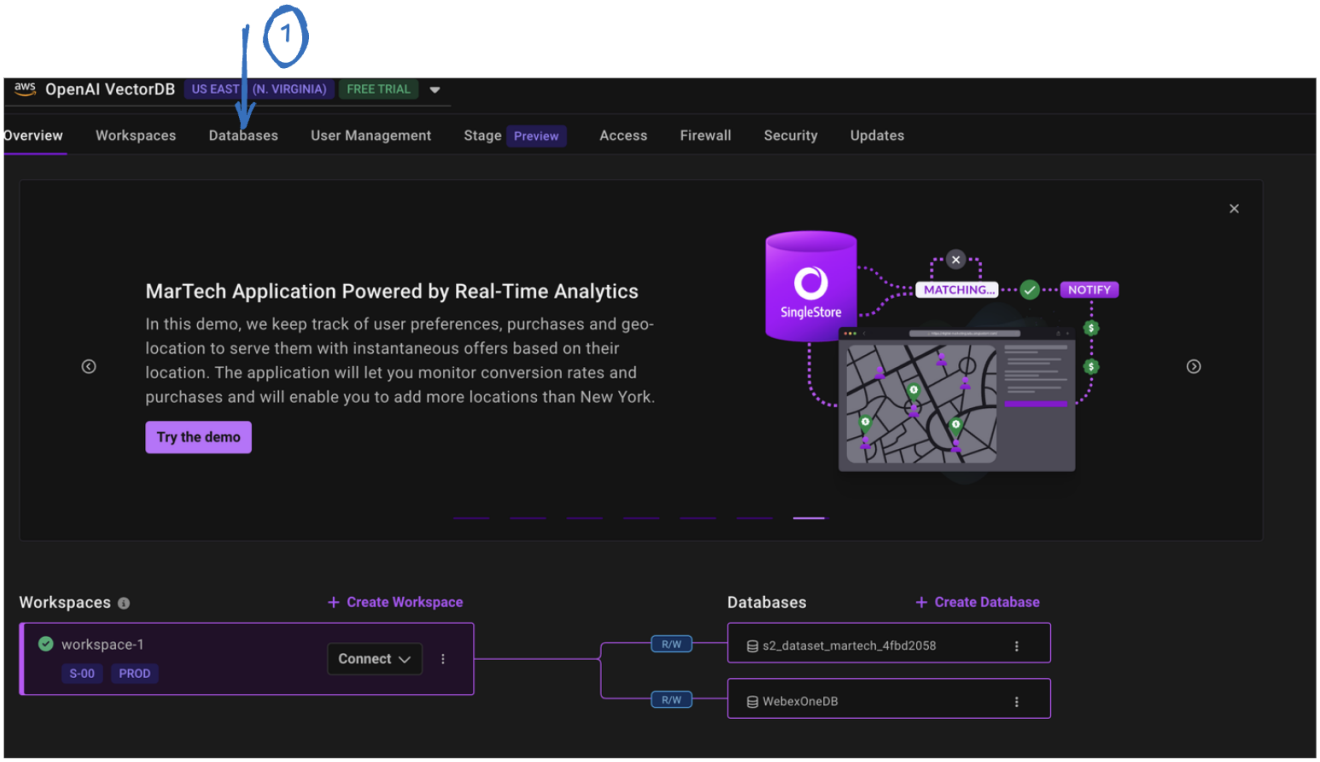
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



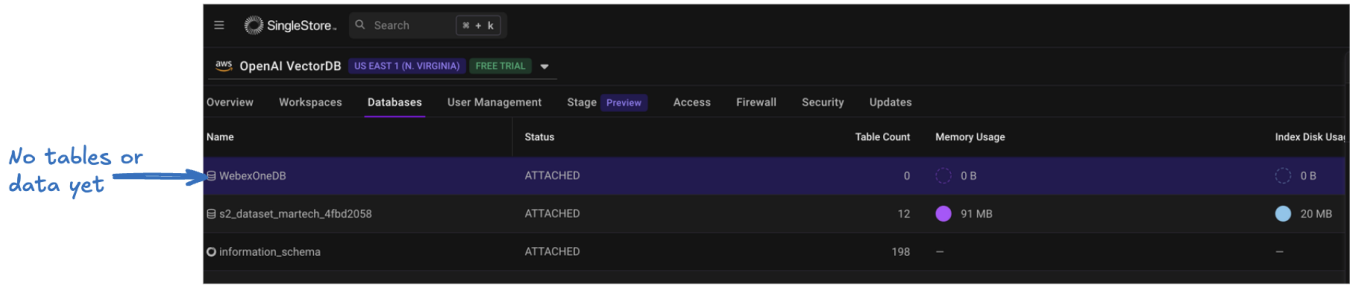
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



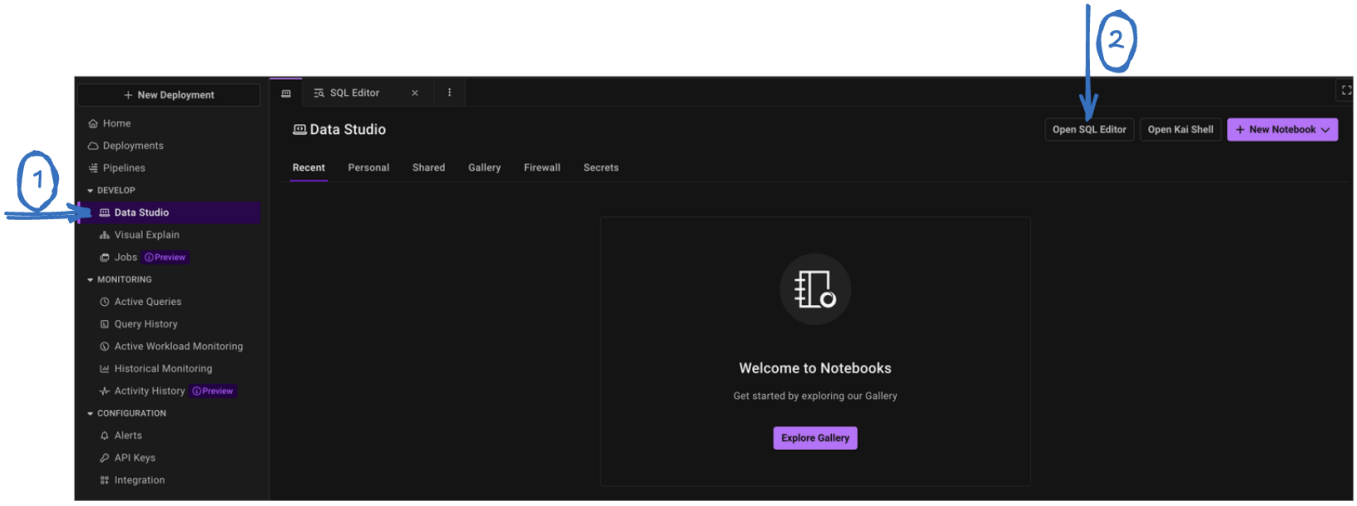
- Click on Database tab



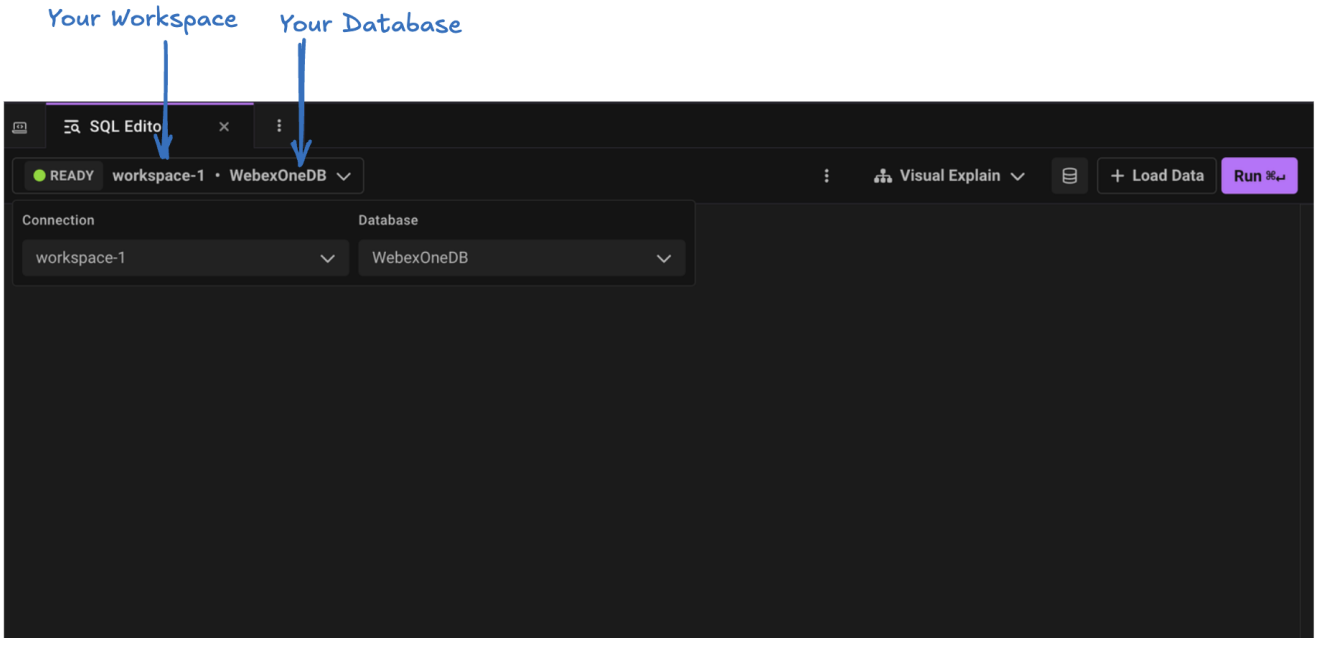
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Return to the home screen and navigate to Develop > Data Studio. Open the SQL Editor.



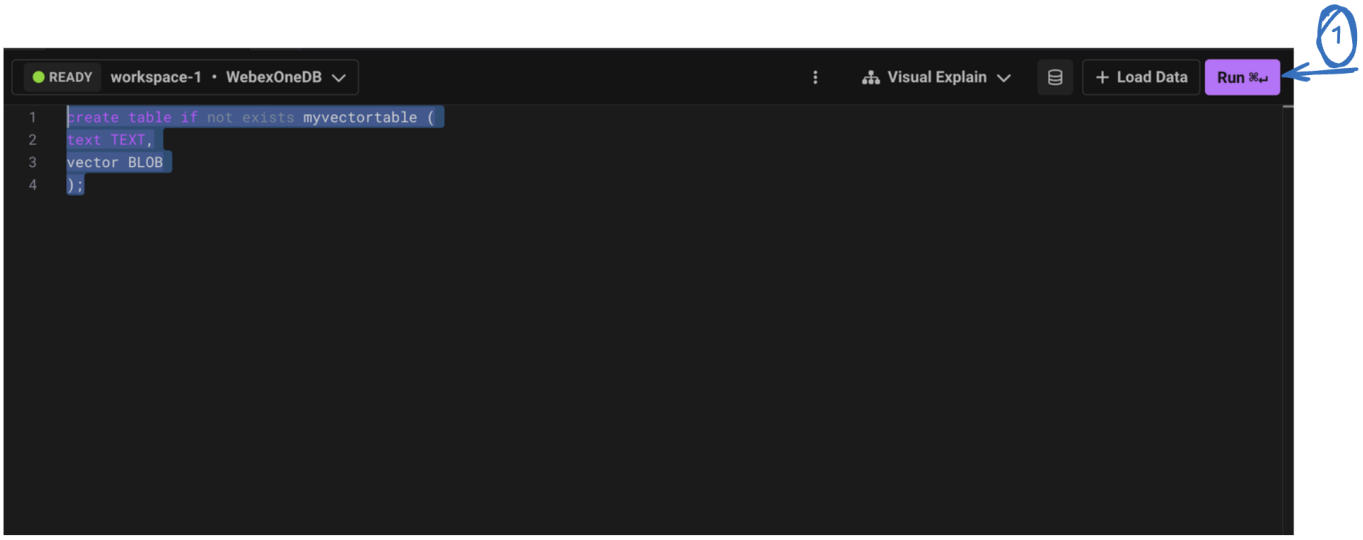
- Be sure to select your workspace and database.



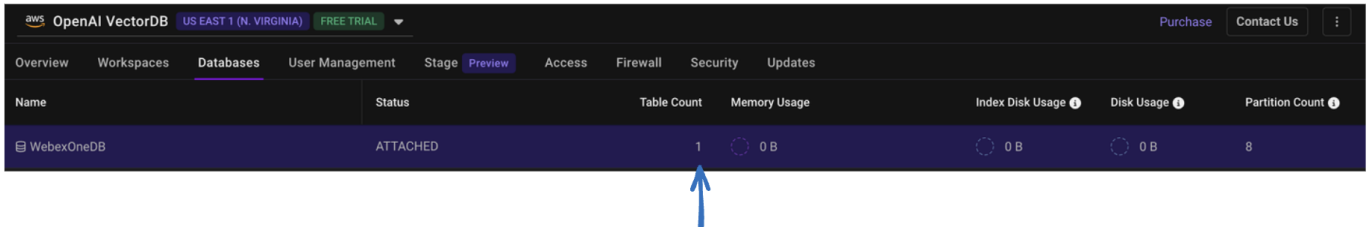
- Run a SQL command to create a table - Press Run

```
create table if not exists myvectortable (
text TEXT,
vector BLOB
);
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

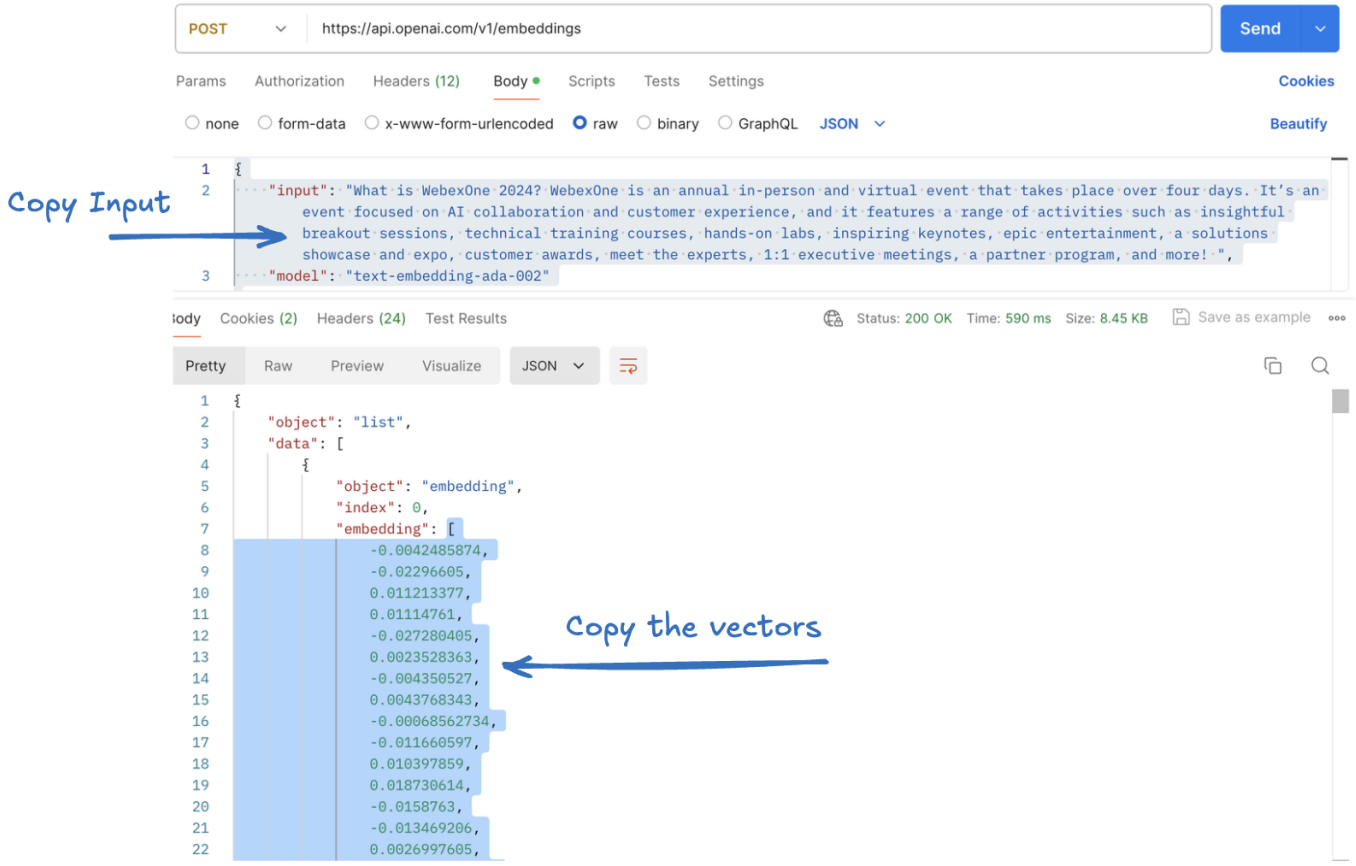


- You can navigate to the Databases tab and verify that the table has been created.



- Let's copy the embeddings we generated earlier using Postman so that we can insert them into our database.
- To proceed, copy the input text and all the embedding values, including the square brackets [], as shown in the image. These will be used for insertion into our database.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Let's head back to our SQL editor and insert the values into our database. Once done press "Run"

```
insert into myvectortable (text ,vector) values ("your_input_text", JSON_ARRAY_PACK("your_embeddings"))
```

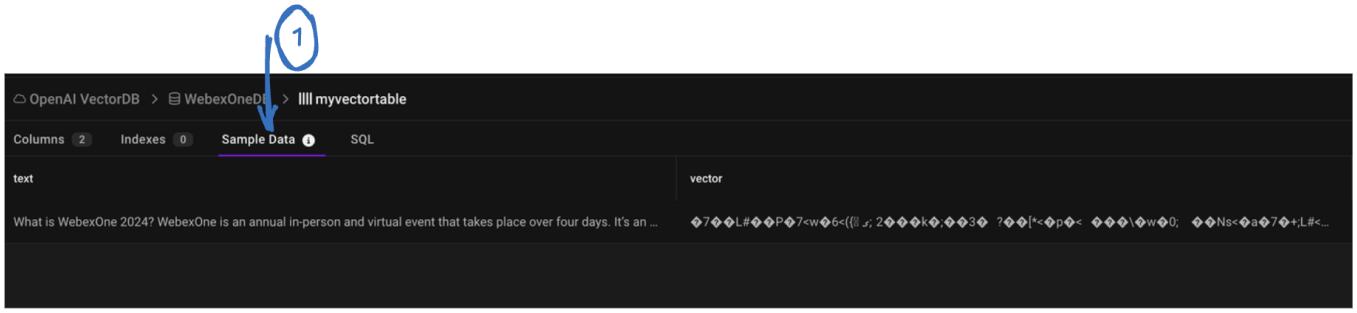
In this query:

```
* Replace "your_input_text" with the input values from Postman.

* Replace your_embeddings with the embeddings you copied earlier, including the square brackets [].
```

```
insert into myvectortable (text ,vector) values ("What is WebexOne 2024? WebexOne is an annual in-person and virtual event that takes place over four days.
It's an event focused on AI collaboration and customer experience, and it features a range of activities such as insightful breakout sessions, technical
training courses, hands-on labs, inspiring keynotes, epic entertainment, a solutions showcase and expo, customer awards, meet the experts, 1:1 executive
meetings, a partner program, and more!", JSON_ARRAY_PACK("[
                -0.0042485874,
                -0.02296605,
                0.011213377,
                0.01114761,
                -0.027280405,
                0.0023528363,
                -0.004350527,
                0.0043768343,
                -0.00068562734,
                -0.011660597,
                0.010397859,
                0.018730614,
                -0.0158763,
                -0.013469206,
                0.0026997605,
                -0.00003768792,
                0.014850326,
                -0.03183152,
                0.0026175508,
                -0.0459321,
                -0.0055803815,
                -0.008247258,
                -0.011713211,
                0.0006169824,
                -0.017112732,
                -0.002885554,
                -0.004113764,
                -0.01348236,
```

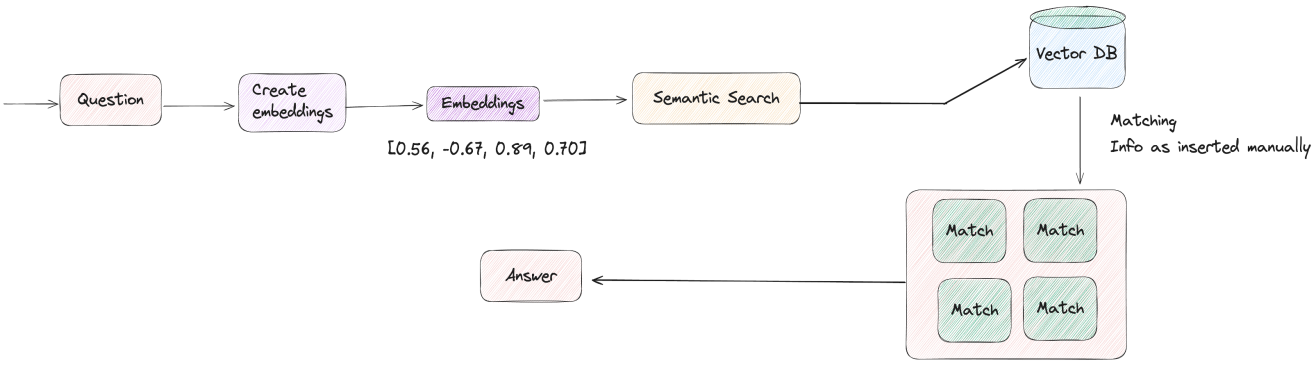
```
        ........
    ]"))
```

- After running the above command, you will see that our table now contains both the input text and the corresponding embeddings.



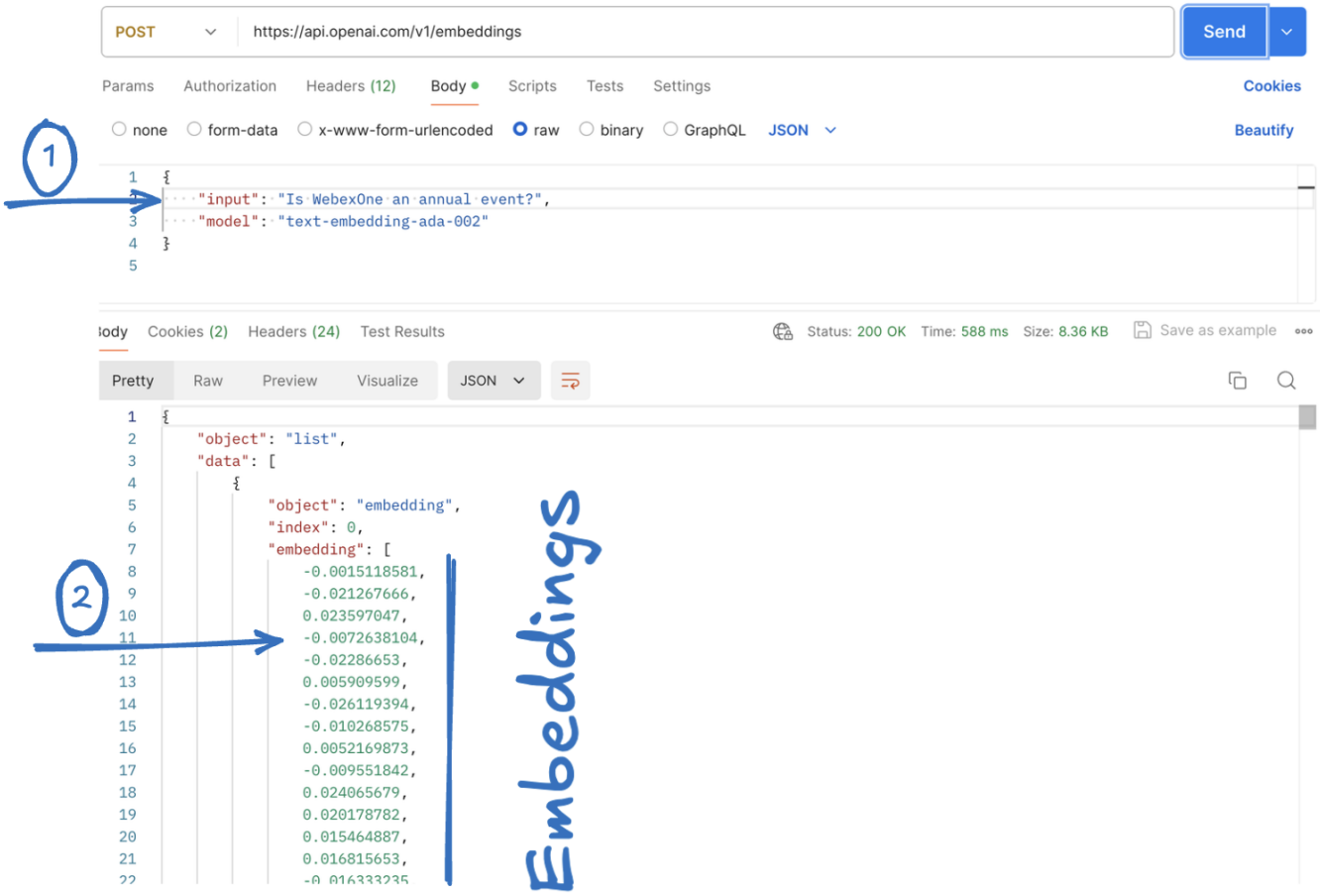
RETRIEVING VALUES FROM THE VECTOR DATABASE

High Level Overview



- Searching the vector database is quite simple. Example: we want to find anything related to WebexOne. To do this, we'll create embeddings for our query, "Is WebexOne an annual event?" and then search the vector database to find matches against the existing embeddings.
- Let's open Postman and create an embedding for our question. Be sure to click on "Send."

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



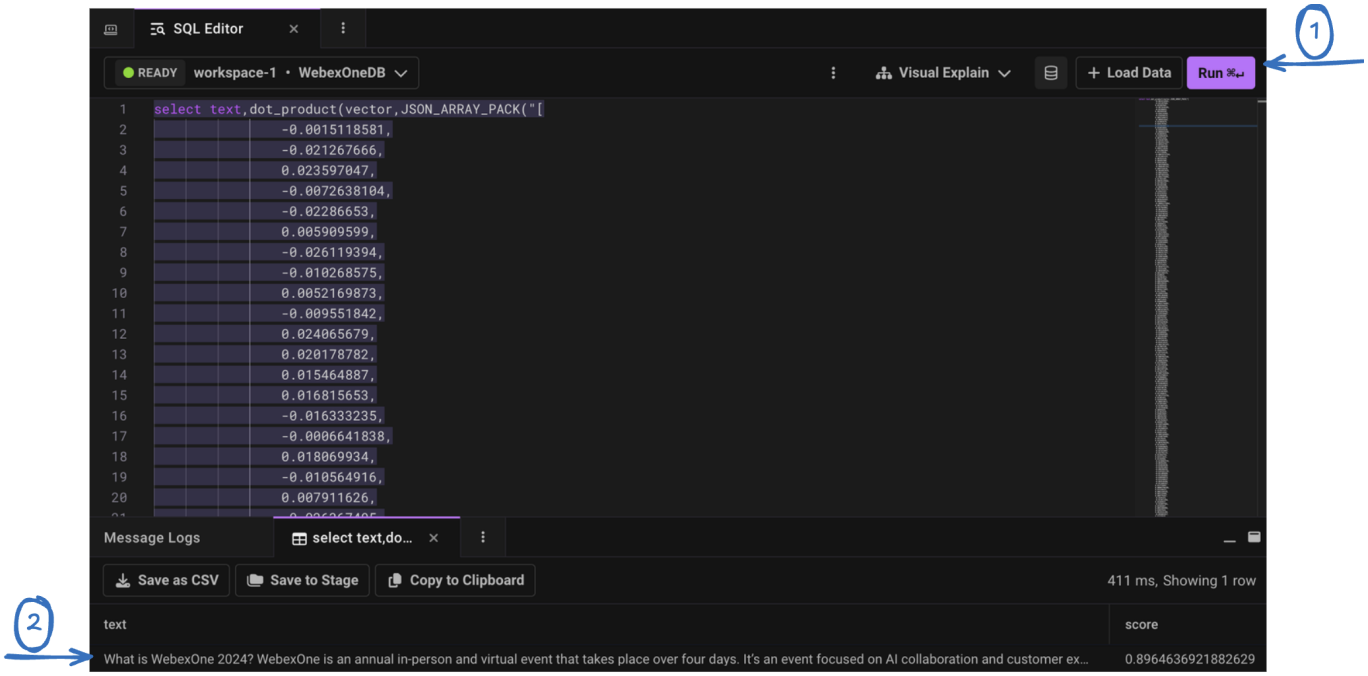
- Let's head over to the SQL editor and run a query for our search.

```
select text,dot_product(vector,JSON_ARRAY_PACK("[your_embeddings]")) as score
from myvectortable
order by score desc
limit 5;
```

* In this query: * Replace your_embeddings with the embeddings you copied earlier, including the square brackets [] for the question.

```
select text,dot_product(vector,JSON_ARRAY_PACK("[
                -0.0015118581,
                -0.021267666,
                0.023597047,
                -0.0072638104,
                -0.02286653,
                0.005909599,
                -0.026119394,
                -0.010268575,
                0.0052169873,
                -0.009551842,
                0.024065679,
                0.020178782,
                0.015464887,
                0.016815653,
                -0.016333235,
                -0.0006641838,
                0.018069934,
                -0.010564916,
                0.007911626,
                -0.026367495,
                -0.0019417254,
                -0.005547787,
                -0.012604848,
                0.0003715036,
                -0.013962504,
                0.011729606,
                -0.00078737224,
                -0.011881223,
                0.007815143,
                0.009992908,
```

```
                0.020316616,
                -0.0032580325,
                -0.0066401153,
                0.0097310245,
                -0.0028652078,
                -0.008145943,
                -0.0074843434,
                ........
            ])) as score
from myvectortable
order by score desc
limit 5;
```



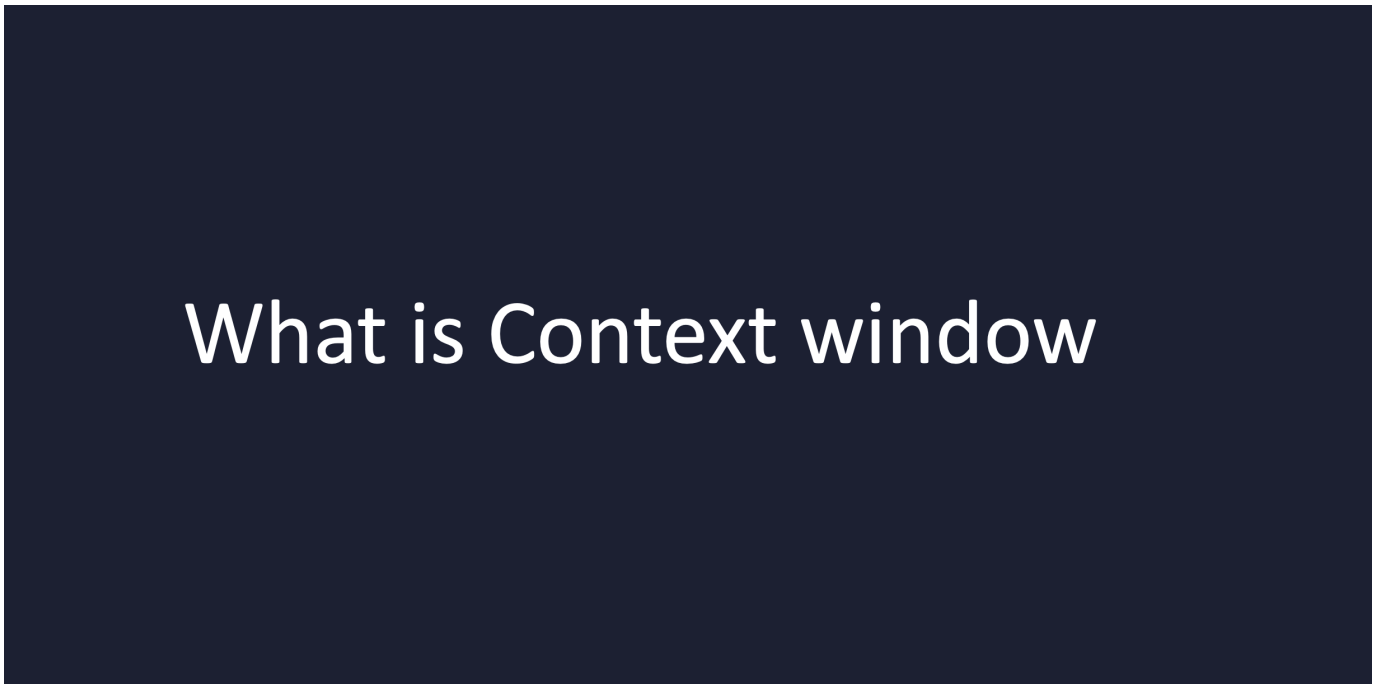
Note: You'll be able to see the success of the vector database—higher scores indicate better matches for the answer.

1.5.4 Conclusion

In summary, vector databases allow LLMs to have long-term memory. In this section, we explored how to use embeddings and vector databases by generating embeddings with the OpenAI API through Postman. After setting up a free SingleStore account, embeddings were stored in a vector database. The process included creating an embedding for a query, searching the database, and confirming that it effectively retrieved relevant information. This demonstrated how vector databases can efficiently manage and query embeddings, making it easier to find relevant information based on the data stored.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.6 Task 5: Context Windows and Retrieval Augmented Generation (RAG)

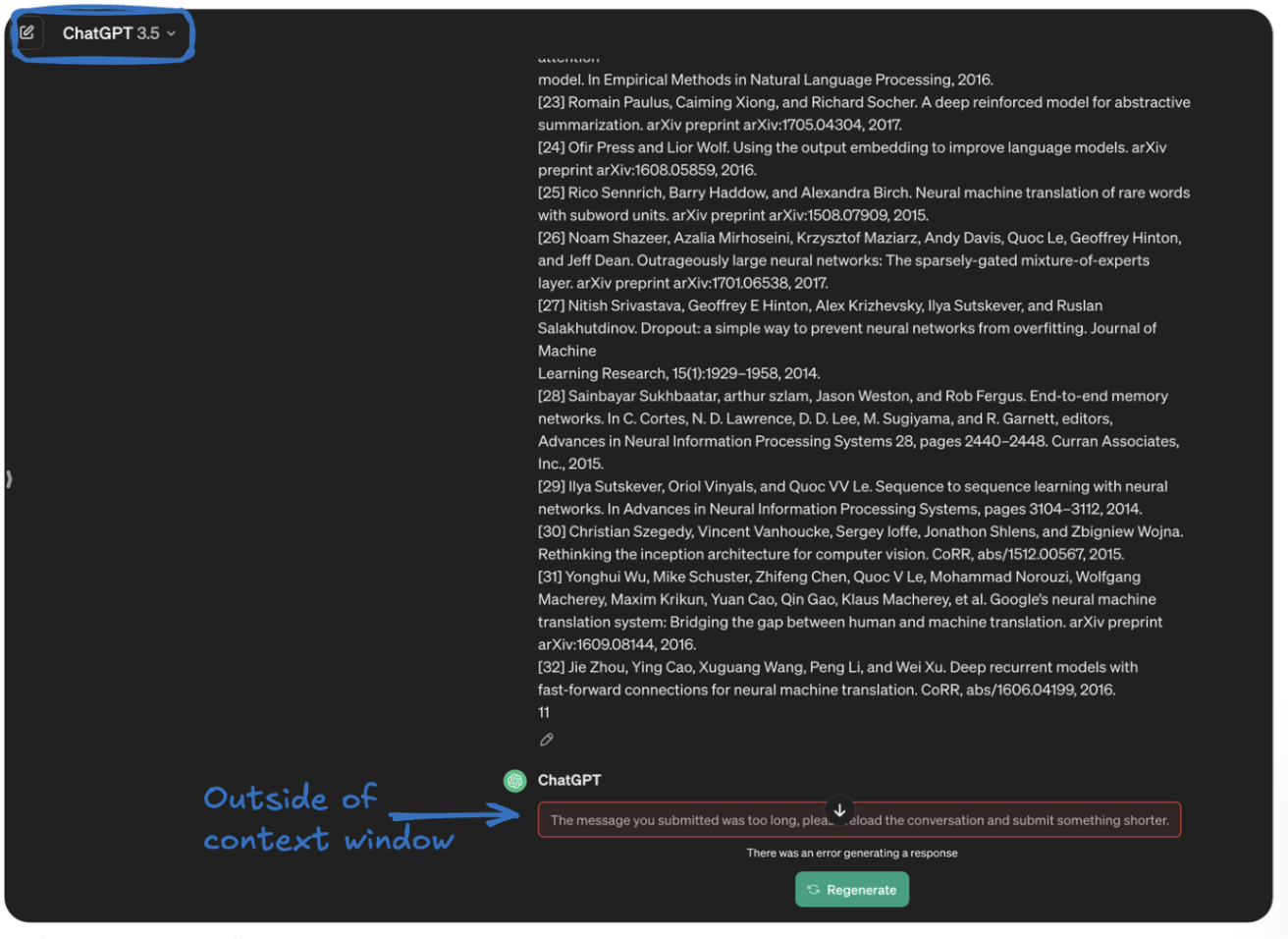


- The context window is the limit on how much text the model can keep in mind and process at once. It includes your messages, the model's replies, and any other text in the conversation. If the conversation gets too long and goes beyond this limit, the model might start to forget what was said earlier.
- Example
- For GPT-3.5, the context window limit is approximately 8,000 tokens (one token typically represents about 4 characters on average.). If the input text along with previous interactions (if any) exceeds the limit of LLM, the model may not be able to see or remember parts of the text that fall outside this window. This can result in errors or the inability to refer back to earlier parts of the conversation or document.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Context Window

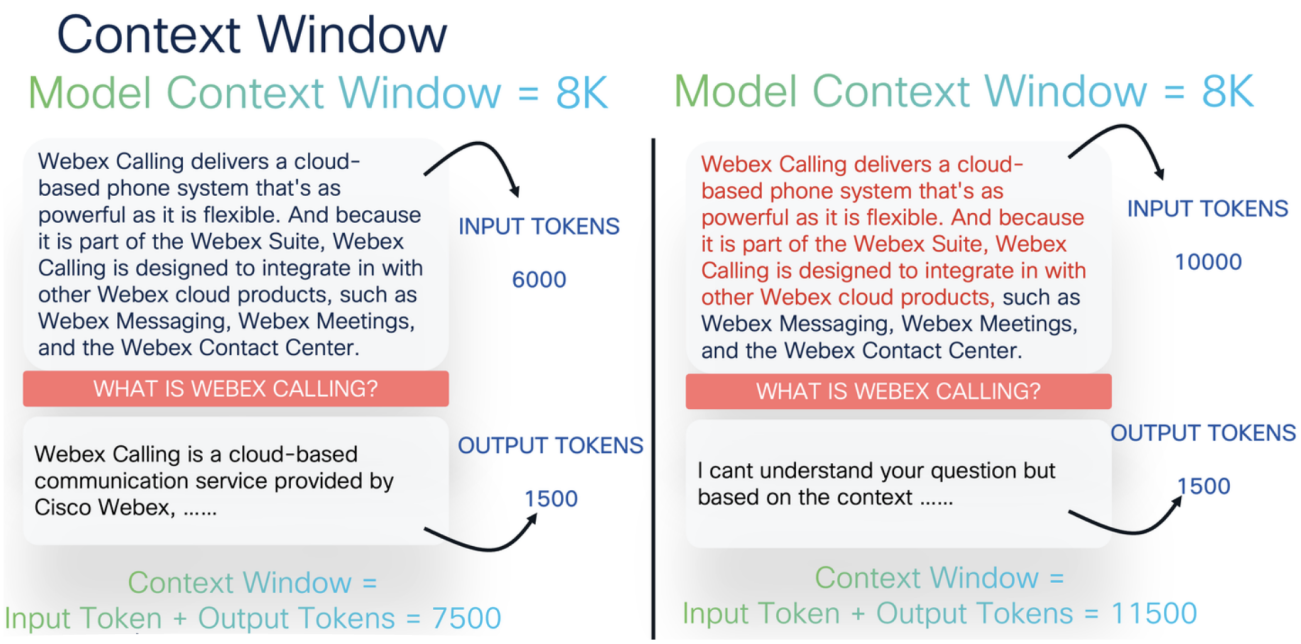
Model Context Window = 8K



Note: A token can be as small as a piece of a word or as large as a word itself, depending on the language and complexity of the text

- Consider a model with a context window limit of 8,000 tokens. If we input 6,000 tokens and the model generates a response of 1,500 tokens, the total token count is 7,500. This is within the model's context window, so all information is processed correctly. Conversely, if the input increases to 10,000 tokens and the response remains at 1,500 tokens, the total becomes 11,500 tokens. This exceeds the model's 8,000-token limit. Consequently, the model might lose context or be unable to respond properly as the excess information falls outside its context window. This demonstrates the importance of managing input size to stay within the model's processing capabilities.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



1.6.1 How Context window work

- In the below example , multiple potential predicted words exist, but the model chooses the words based on the surrounding context and their probability. However, if the context window size increases and the word "Webex" moves out of the window, the model may lose critical context. As a result, the next predicted word might be incorrect, as the model can no longer reference "Webex" to inform its predictions accurately. This highlights the importance of keeping key information within the context window to maintain the accuracy of the model's predictions.
- Additionally, the temperature setting in the model can influence the type of content it generates. Lower temperatures are useful for summarizing or generating more deterministic and concise outputs, while higher temperatures encourage creativity, making the model more suitable for writing stories or poems. Adjusting the temperature allows you to control the balance between predictability and creativity in the model's output.

	1	2	3	4	5	6	7	8	Predicted word
						Webex	Calling	is	cloud
					Webex	Calling	is	cloud	based
				Webex	Calling	is	cloud	based	telephony
			Webex	Calling	is	cloud	based	telephony	solution
		Webex	Calling	is	cloud	based	telephony	solution	that
	Webex	Calling	is	cloud	based	telephony	solution	that	offers
Webex	Calling	is	cloud	based	telephony	solution	that	offers	enterprise

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

How can we address the challenges related to the context window?

- Thats where RAG com einto play

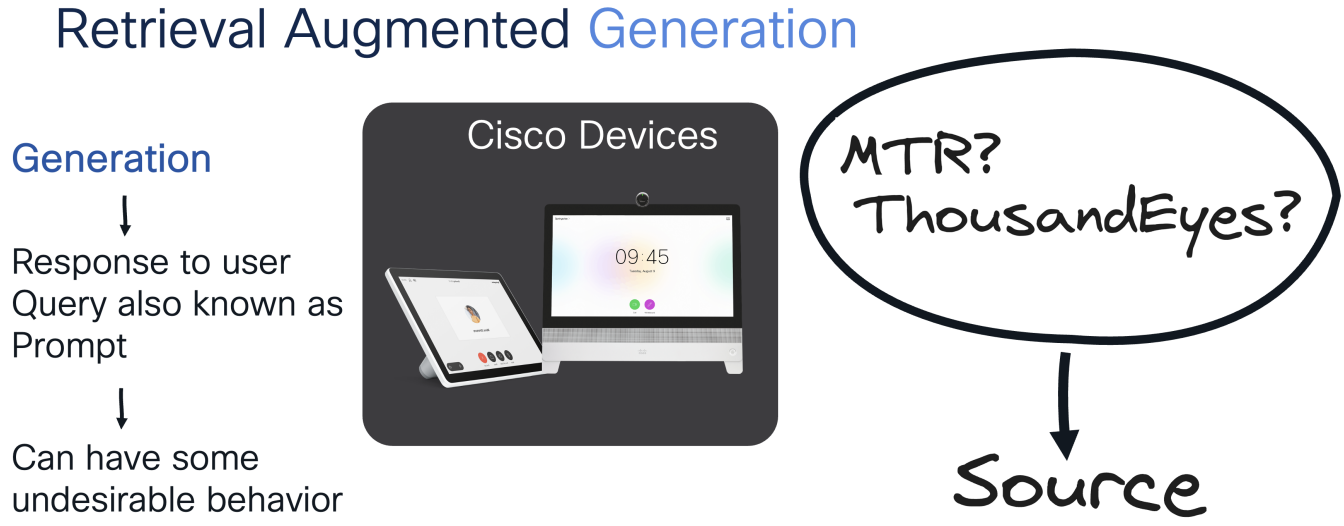
1.6.2 Retrieval Augmented Generation (RAG)

Lets try to undertsand what RAGs are all about before we can look how they can solve context window issues

RAG = Retrieval Augmented Generation

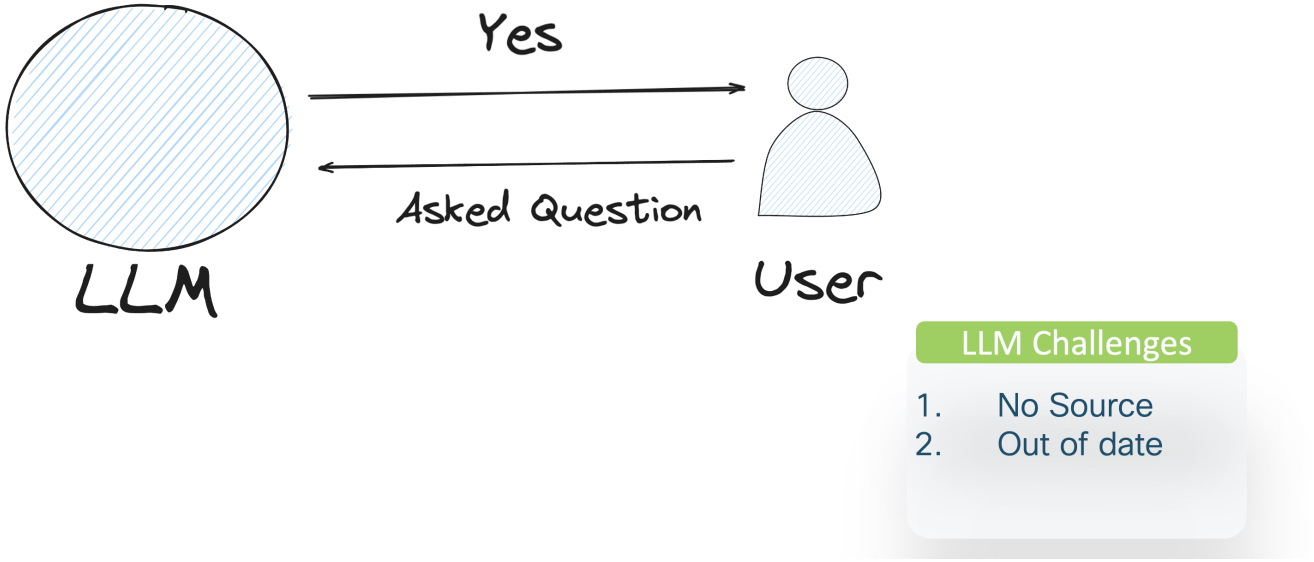
RAG = Retrieval Augmented Generation

Lets focus on the Generation part. Generation basically means responding to user query. For example, you might ask the model whether DX or Navigators are compatible with supporting ThousandEyes.

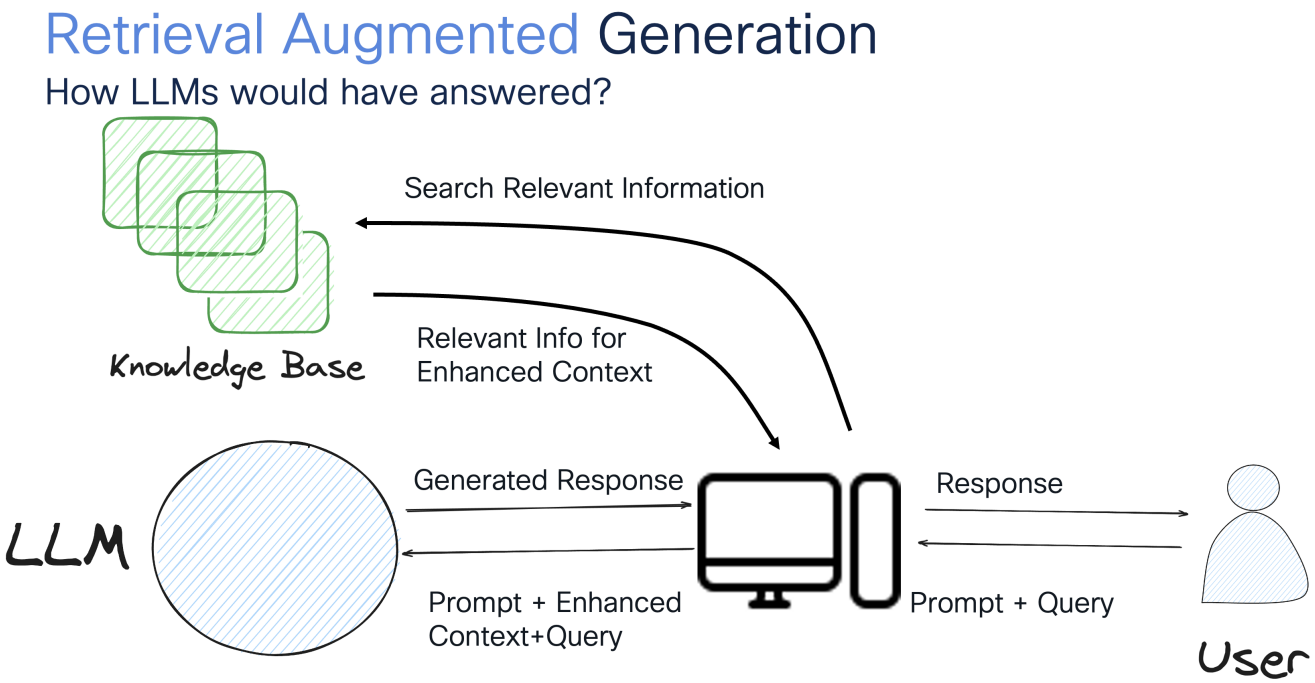


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

That's an excellent question. LLMs are trained on vast amounts of historical data, so they may recognize that DX and Navigators are well-regarded Cisco products. As a result, the LLM might confidently respond that you can install ThousandEyes on those devices. However, the issue is that the response may lack a verifiable source to back up that claim or the information can be out-of-date



What if, instead of just asking the question, I had first consulted a reputable source and only then informed the user that ThousandEyes can't be installed on those devices? This is where Retrieval-Augmented Generation comes into play. By integrating a knowledge base—such as PDFs, CSVs, images, and other resources—into the model, we ensure that any time a question is asked, the response is based on the most current and accurate information available.

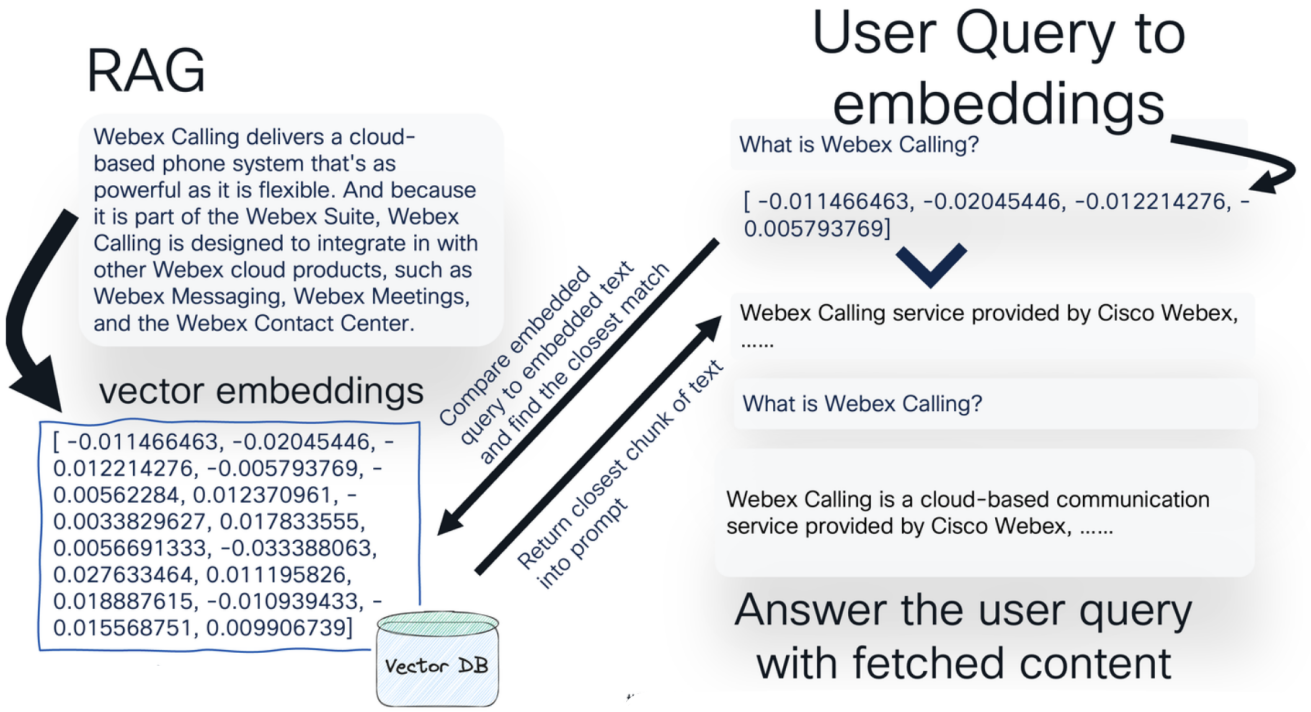


1.6.3 How can RAG address the limitations we've seen with the context window?

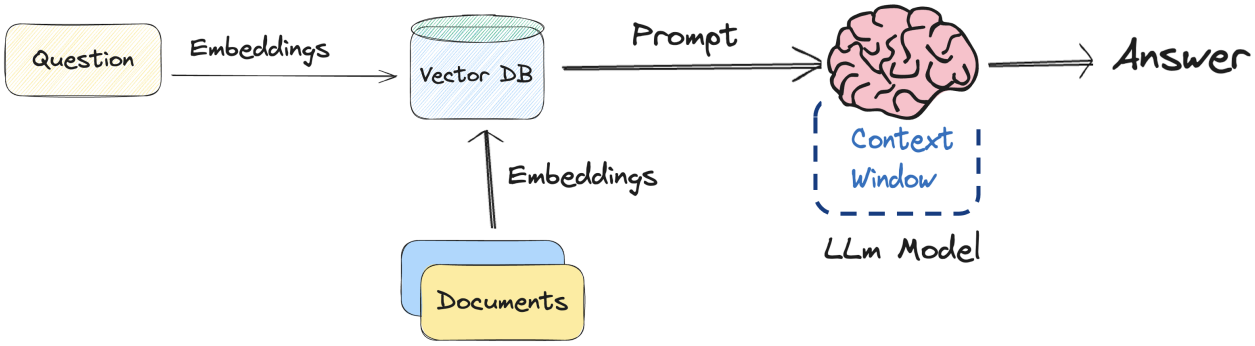
To address the context window limitation, we can use an embedding model to convert relevant text into embeddings and store them in a vector database. When a user submits a query, we transform it into an embedding and compare it to the stored embeddings to find the closest match.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Once we identify the match, we retrieve that specific chunk of text to respond to the query. This approach effectively overcomes the context window limitation by ensuring that our responses are accurate and contextually relevant.

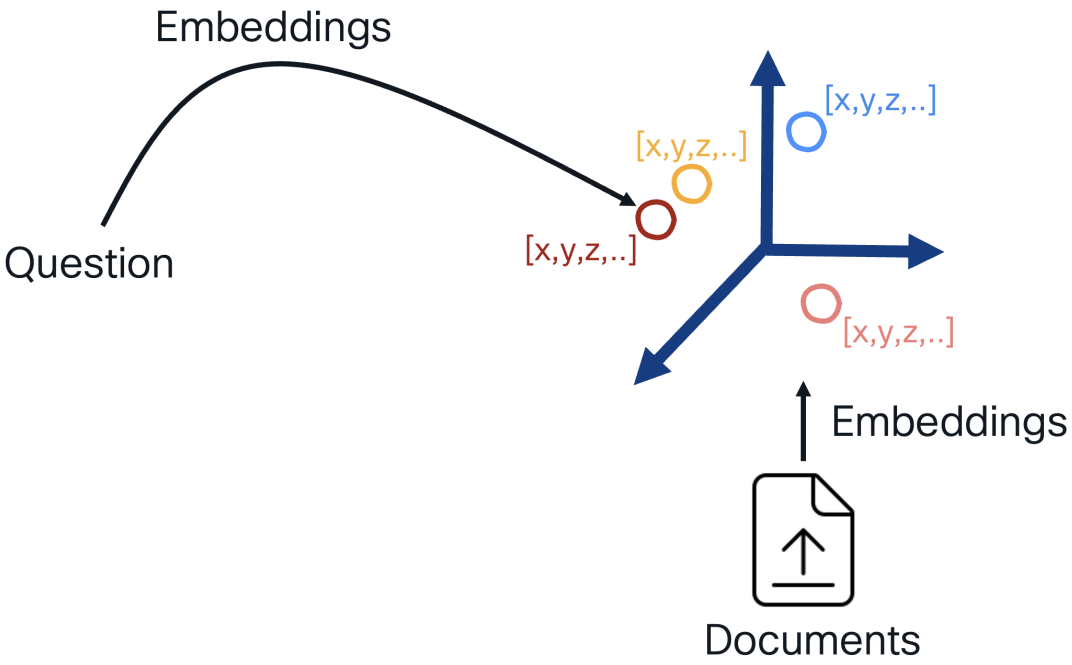


Now you can see how RAG can address some of the challenges with LLMs. Instead of re-training or fine-tuning the model with new information, we can augment our data sources to retrieve the most up-to-date information. Additionally, LLMs are now instructed to prioritize primary source data before generating responses, making them less likely to hallucinate and encouraging more accurate and reliable outputs.



Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.6.4 What's Happening Under the Hood



We normally start by taking multiple documents and converting them into embeddings. Imagine these embeddings as points in a 3D space, where each document is projected based on its semantic meaning or content. Documents located near each other in this space contain similar semantic information, which is crucial when we search for or retrieve information.

Now, when we receive a query, we also convert it into an embedding (numerical value). We then perform a similarity search in this 3D space, looking for the documents that are closest to our query's embedding. These nearby documents are likely to contain the most relevant information.

Once we've identified these matching documents, we retrieve the relevant sections (or "splits") and feed all this information into the LLM's context window, allowing it to generate a well-informed response.

Large context window size. Do we still need RAG?

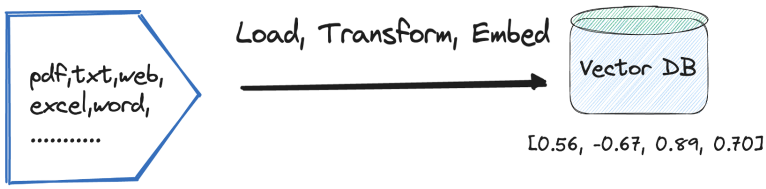
We've seen models like Gemini Pro with a context window size of millions, allowing them to retain more information. So, why use RAG? The answer is that RAG remains highly beneficial for sourcing real-time data,fact-checking, and accessing external knowledge that isn't contained within the context window.

1.6.5 Lets build a quick RAG application

While LLMs (Large Language Models) like GPT-4 , Llama, Gemini possess impressive general knowledge, they don't inherently have access to your data. To connect LLMs with your own data sources, we can use frameworks like LangChain. These frameworks enable us to leverage our own data by breaking down documents into smaller chunks and storing them as embeddings in vector databases.

This approach allows us to build applications that can effectively use language models. Example: When a user asks a question, we convert it into embeddings, perform a semantic search to find the most relevant answers, and then send both the question and the retrieved information to the LLM to generate a response.

In this section, we'll briefly introduce LangChain, but in the upcoming chapters, we'll dive deeper to gain a more comprehensive understanding of how it works.



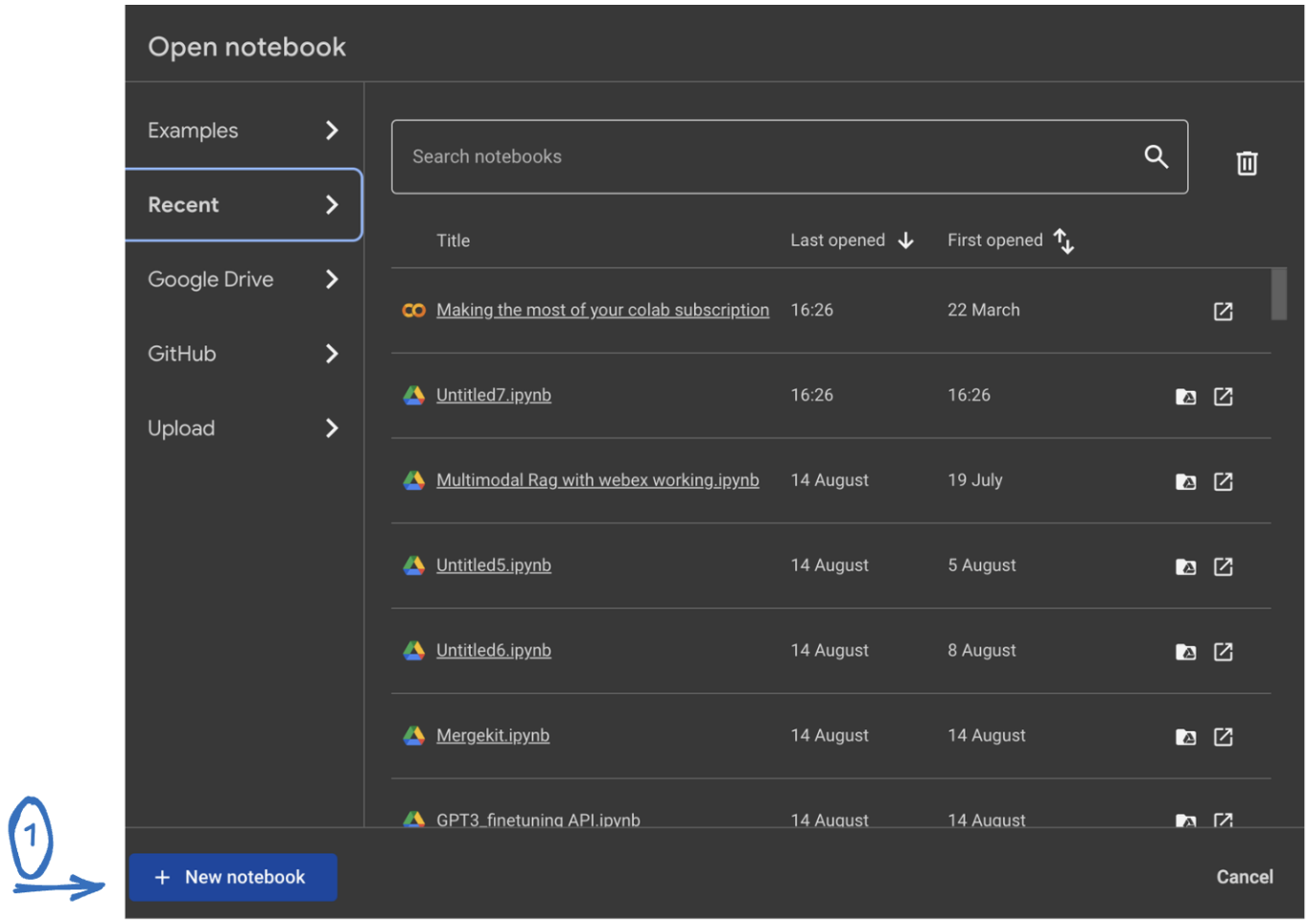
Step 1: Load data source also called Data ingestion
Step 2: Transform, where we break data into small chunks
Step 3: Convert Chunks into vectors also called Embeddings
Step 4: Save in Vector Database

Entire RAG pipeline

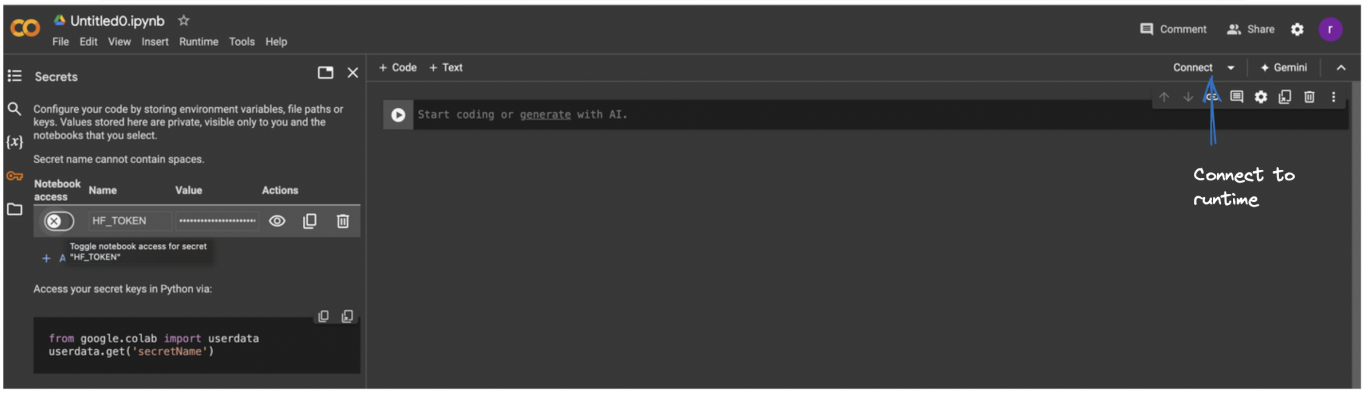
Note: In this section, we will be using the Cisco DECT 6800 Deployment Guide. For the purposes of this lab, I've modified the original PDF by removing a few pages and saving it as a new file. The PDF we are using in this lab can be downloaded from here

- Open Google Colab and create a new notebook. Click on "File" > "New notebook". Please refer to the following section to create Google Colab account.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

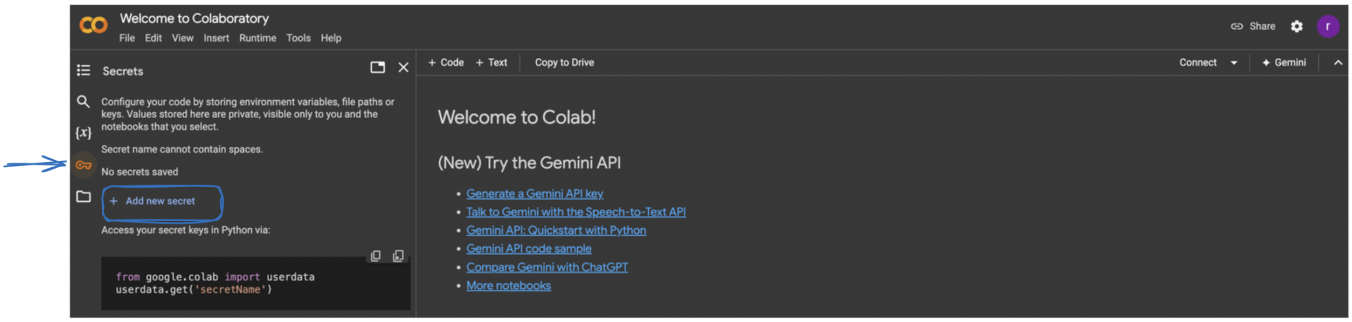


- Make sure you are connected to a runtime. For this task, you can use the CPU as the runtime environment.



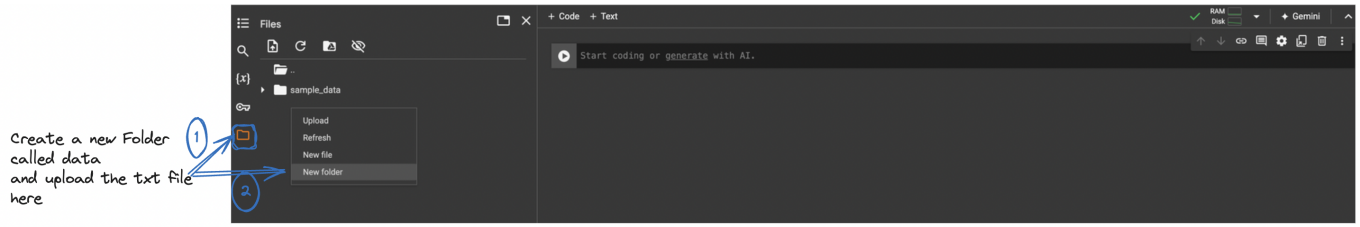
- Within your existing Google Colab notebook navigate to the new “Secrets” section in the sidebar.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



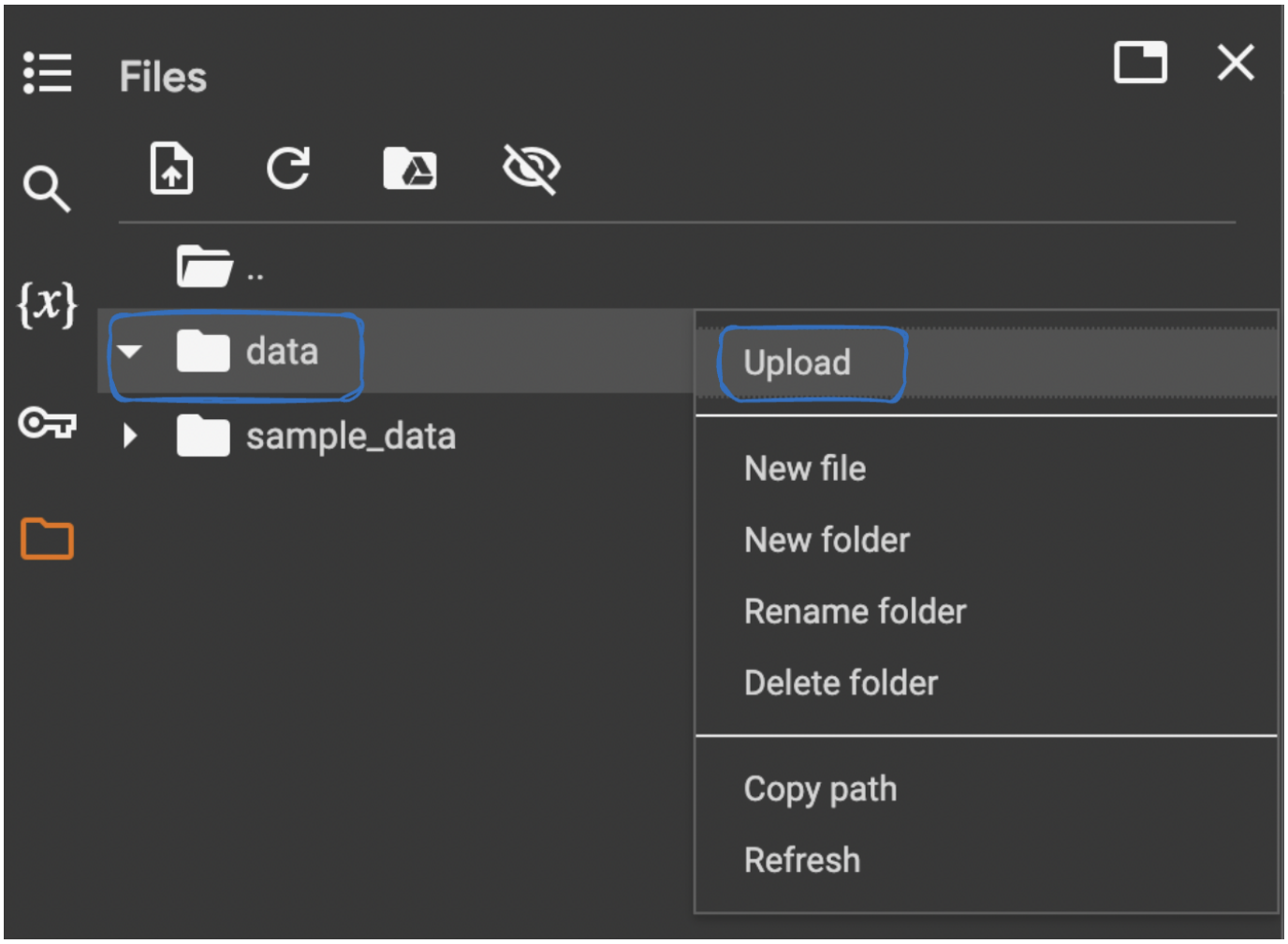

```
* If not already done, Click on “Add a new secret.” Enter the name example: OPENAI_API_KEY and value of the secret(the API key created earlier). Note: The name is permanent once set.

* The list of secrets is global across all your notebooks.

* Use the “Notebook access” toggle to grant or revoke access to a secret for each notebook.
```

- Let's load our PDF files into Google Colab. For this example, we can use the modified DECT guide
- Within Google Colab, Click on Folder and create a new folder called "data"



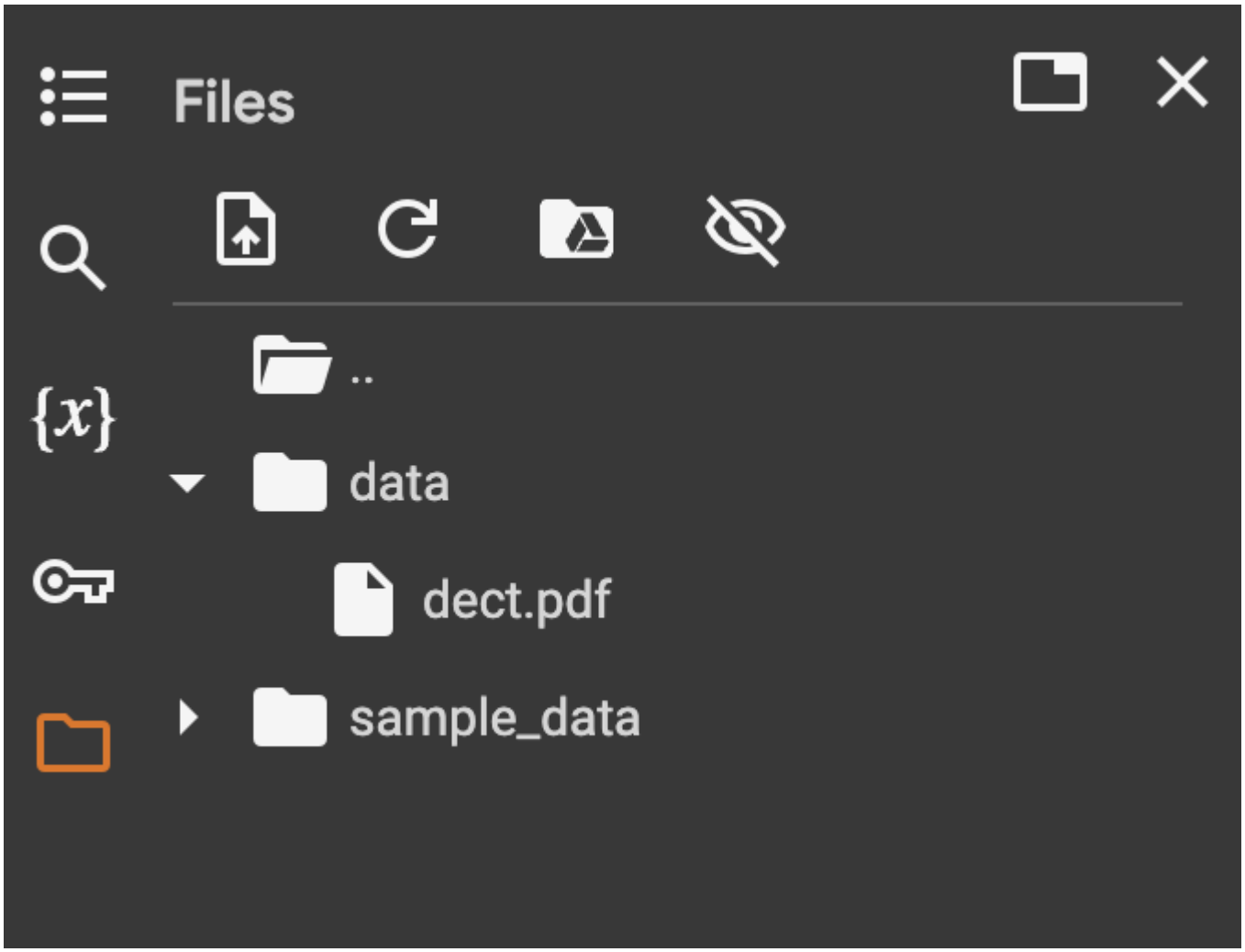
- Click on [...], select Upload

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Choose your dect.pdf file and click Open

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Install relevant Python packages

```
!pip install langchain langchain_community langchain-openai python-dotenv chromadb pymupdf
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

```
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-packages (0.2.14)
Requirement already satisfied: langchain_community in /usr/local/lib/python3.10/dist-packages (0.2.12)
Requirement already satisfied: langchain-openai in /usr/local/lib/python3.10/dist-packages (0.1.22)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.10/dist-packages (1.0.1)
Requirement already satisfied: chromadb in /usr/local/lib/python3.10/dist-packages (0.5.5)
Requirement already satisfied: pypdf in /usr/local/lib/python3.10/dist-packages (4.3.1)
Requirement already satisfied: pymupdf in /usr/local/lib/python3.10/dist-packages (1.24.9)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.2)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0.32)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (3.10.3)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Requirement already satisfied: langchain-core<0.3.0,>=0.2.32 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.2.34)
Requirement already satisfied: langchain-text-splitters<0.3.0,>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.2.2)
Requirement already satisfied: langsmith<0.2.0,>=0.1.17 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.1.101)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.26.4)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.8.2)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.32.3)
Requirement already satisfied: tenacity!=8.4.0,<9.0.0,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (8.5.0)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from langchain_community) (0.6.7)
Requirement already satisfied: openai<2.0.0,>=1.40.0 in /usr/local/lib/python3.10/dist-packages (from langchain-openai) (1.42.0)
Requirement already satisfied: tiktoken<1,>=0.7 in /usr/local/lib/python3.10/dist-packages (from langchain-openai) (0.7.0)
Requirement already satisfied: build>=1.0.3 in /usr/local/lib/python3.10/dist-packages (from chromadb) (1.2.1)
Requirement already satisfied: chroma-hnswlib==0.7.6 in /usr/local/lib/python3.10/dist-packages (from chromadb) (0.7.6)
Requirement already satisfied: fastapi>=0.95.2 in /usr/local/lib/python3.10/dist-packages (from chromadb) (0.112.1)
Requirement already satisfied: uvicorn>=0.18.3 in /usr/local/lib/python3.10/dist-packages (from uvicorn[standard]>=0.18.3->chromadb) (0.30.6)
Requirement already satisfied: posthog>=2.4.0 in /usr/local/lib/python3.10/dist-packages (from chromadb) (3.5.2)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from chromadb) (4.12.2)
Requirement already satisfied: onnxruntime>=1.14.1 in /usr/local/lib/python3.10/dist-packages (from chromadb) (1.19.0)
Requirement already satisfied: opentelemetry-api>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from chromadb) (1.26.0)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from chromadb) (1.26.0)
Requirement already satisfied: opentelemetry-instrumentation-fastapi>=0.41b0 in /usr/local/lib/python3.10/dist-packages (from chromadb) (0.47b0)
Requirement already satisfied: opentelemetry-sdk>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from chromadb) (1.26.0)
```

- Let's retrieve a OpenAI API key and set it as an environment variable within the Colab environment

```
import os
from google.colab import userdata
os.environ['OPENAI_API_KEY'] = userdata.get('OPENAI_API_KEY')
```

- Install relevant libraries for RAG

```
from langchain import hub
import os
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import WebBaseLoader
from langchain_community.vectorstores import Chroma
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from dotenv import load_dotenv
from langchain.document_loaders import PyMuPDFLoader
load_dotenv()
```

- Load data source also called Data ingestion

```
loader = PyMuPDFLoader("/content/data/dect.pdf")
docs = loader.load()
```

- Transform, where we break data into small chunks. We can use techniques like RecursiveCharacterTextSplitter or tiktoken for tokenizing text.

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
splits = text_splitter.split_documents(docs)
```

Note: RecursiveCharacterTextSplitter Covered in previous section

- Let's create our embeddings. As discussed in the Embedding section, there are multiple techniques available. In this lab, we will use OpenAI embeddings. By default, the text-embedding-ada-002 model is used, but we can also opt for the text-embedding-3-large model if desired. Once the embeddings are created, they can be stored in a Vector Database. While there are various databases available, as covered in the Embeddings and Vector DB section, in our example, we will use the Chroma Vector DB, which can be easily deployed locally on your machine. We will also create a retriever by using retriever = vectorstore.as_retriever() which converts the vector store into a retriever object. This retriever can then be used to find and retrieve documents or data from the vector store that are most relevant to a given query, based on the similarity of their embeddings.

```
vectorstore = Chroma.from_documents(documents=splits,
                                    embedding=OpenAIEmbeddings(model="text-embedding-3-large"))

retriever = vectorstore.as_retriever()
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Note: We are using Chroma dB but you have options to use other databases as well such as Faiss. Also we can use embeddings from Ollama, an open-source and cost-free embedding solution. Below is a code snippet for your reference only:

```
# from langchain_community.embeddings import OllamaEmbeddings
# embeddings = OllamaEmbeddings()
```

Note: More info on Vector Databases can be found here

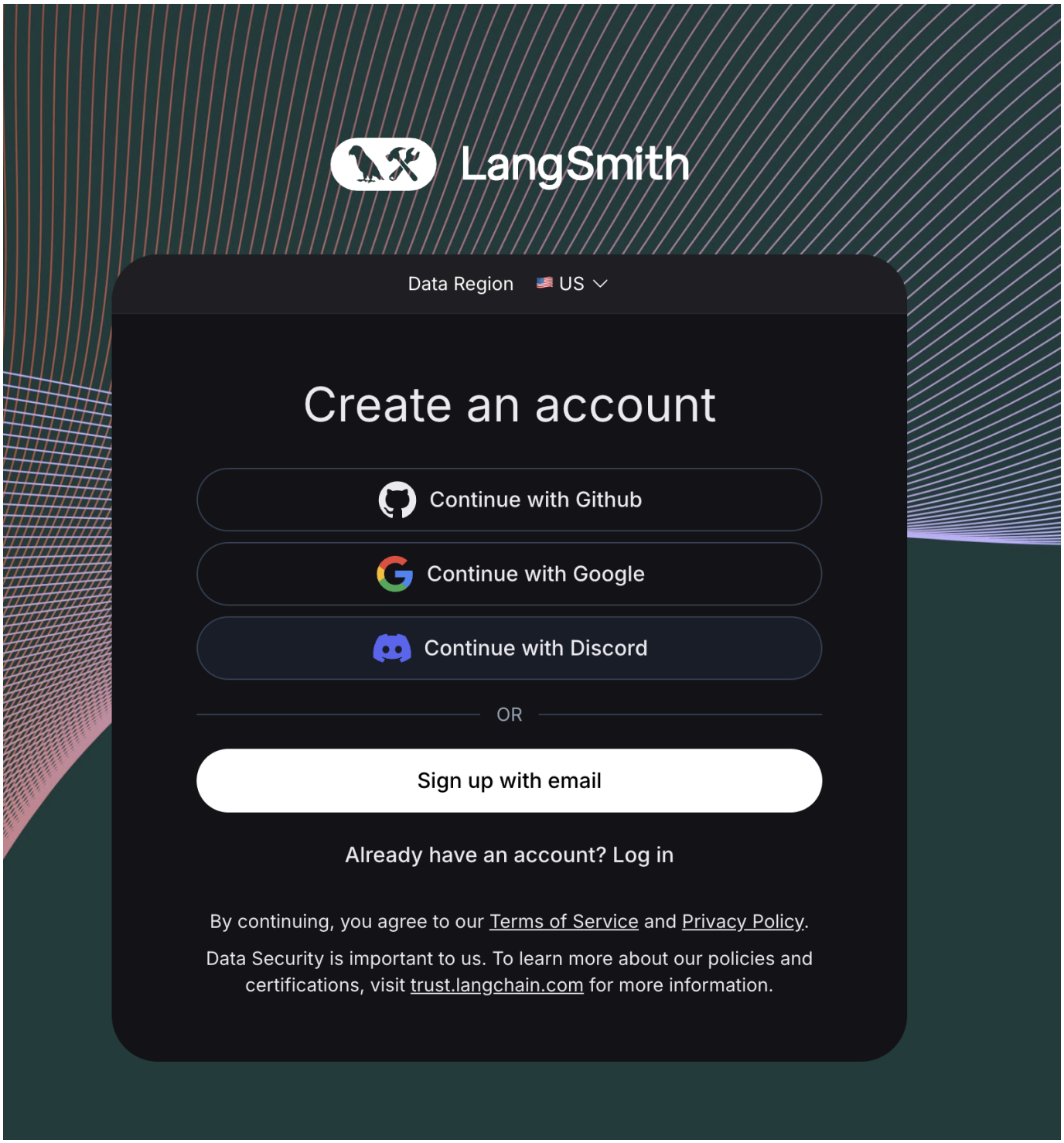
- At this stage, we've loaded the document, created splits, converted them into embeddings, and stored them in the Vector DB. Now, let's query the database to ensure we can retrieve the information.

```
query = "what is the Quick Set up and Installation Process for the Dect phones"
result = vectorstore.similarity_search(query)
print(result[0].page_content)
```

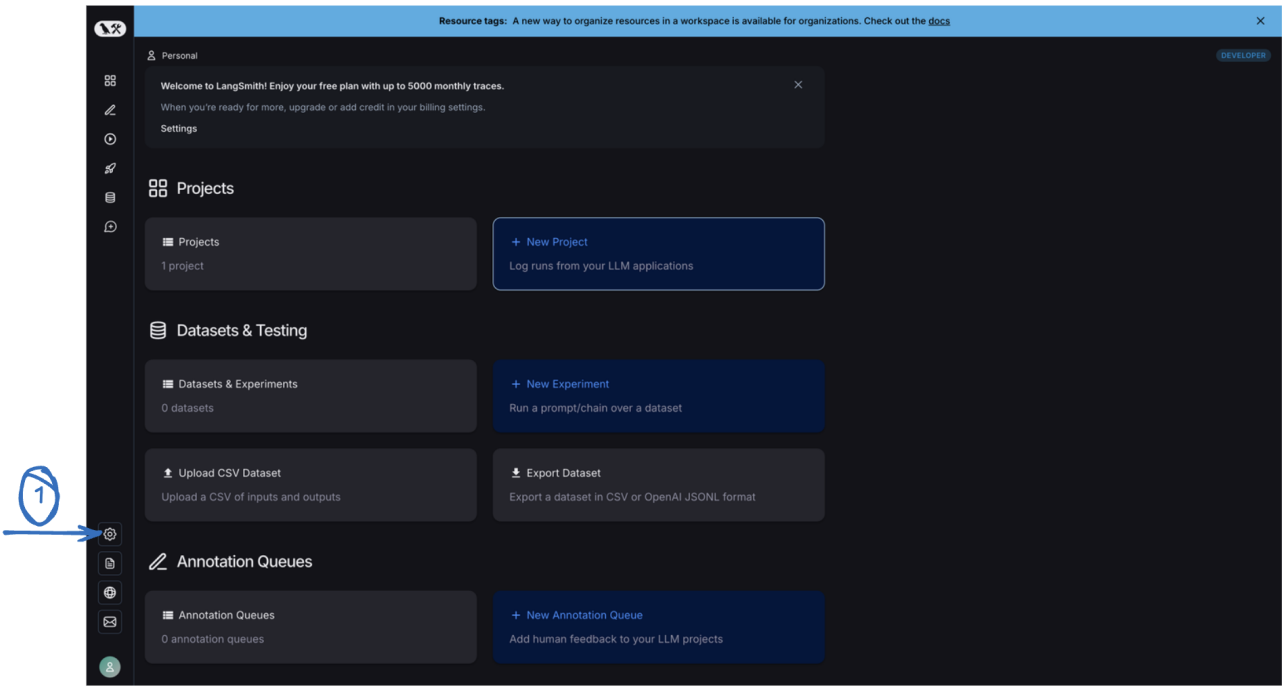
```
Quick Set up and Installation Process
This first section is a quick step-by-step guide. The details and context are found in the remainder
of this document. You must read the entire document before you start your deployment to
ensure success.
Note: The Cisco IP DECT 6800 Phone products support single cell, dual cell, and multicell
deployments. All call control systems may not support multicell deployment. Currently,
Cisco Webex Calling and Webex Calling Carrier support single cell and multicell
deployment. Before you begin, check with your call control provider to ensure that they
support the deployment models described in this guide.
Note: There are many ways to install the base stations. This guide provides the best practices
and the recommended deployment models supported by Cisco TAC.
Plan for the Device Installation
These instructions are for the planner:
1. Review the site to install the DECT system.
• Plan the location of the base stations.
```

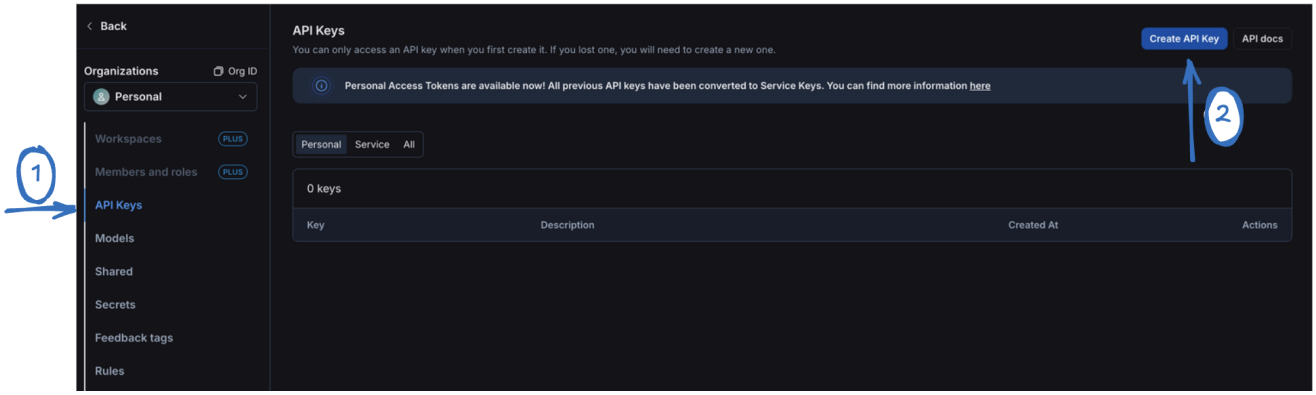
- Let's create our prompt. There are several ways to do this, you can either craft it manually or use prompts that have already been made available on the LangChain Hub. In this section, I'll demonstrate how to pull a prompt from the LangChain Hub and use it. Browse to Langmith and create an account. In this example, I'll use Google to sign up for a Langsmith account, but feel free to choose the option that works best for you.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Create your API Key





- You can either enter your key in the secret folder or paste it manually here.

```
os.environ["LANGCHAIN_API_KEY"] = "PASTE-YOUR-KEY-HERE"
```

- Lets pull our prompt

```
prompt = hub.pull("rlm/rag-prompt")
print(prompt)
```

OUTPUT "You are an assistant for question-answering tasks. Use the following pieces of retrieved context to answer the question. If you don't know the answer, just say that you don't know. Use three sentences maximum and keep the answer concise.\nQuestion: {question} \nContext: {context} \nAnswer:"

Here's an example of a manual prompt for your reference only—if you prefer not to use the downloaded prompt.

Reference ONLY

```
template = """Answer the question based only on the following context:
{context}
```

```
Question: {question}
"""
```

- We can now connect LLMs to our data sources. In this lab, we are using OpenAI models, but feel free to use open-source models if you prefer.

```
llm = ChatOpenAI(model_name="gpt-4o", temperature=0)
def format_docs(docs):
    return "\n\n".join(doc.page_content for doc in docs)
```

- Lets use the Retrieval-Augmented Generation (RAG) pipeline to put all together. We will use retriever | format_docs to retrieve relevant documents (or context) related to the input question and formats them. The RunnablePassthrough method will be used to simply pass the question without modifying it.

```
Question = "Quick Set up and Installation Process for dectphone and summarise"
#  Chain that wik take our input question from below
rag_chain = (
    {"context": retriever | format_docs, "question": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser()  # This component parses the LLM's output, typically converting it into a string format for easy use.
)
req = rag_chain.invoke(Question)
print(req)
```

Note: When we pass our question into the rag_chain.invoke() method, it first goes through the retriever component that we set up earlier. The retriever uses the same embedding model, text-embedding-3-large, to transform the question into an embedding vector. This vector is then compared with the embeddings of documents stored in the Chroma vector store to identify the most relevant ones. Once these relevant documents are retrieved based on the similarity of their embeddings, they are sent along with the original question to the language model (LLM), which then generates a response.

OUTPUT To set up and install a DECT phone system, first review the site and plan the location of the base stations, ensuring they are within 50 meters of each other for good coverage. Upgrade the base stations to the latest firmware, configure them according to the Cisco IP DECT 6800 Series Administration Guide, and unpack and prepare the handsets. Finally, mount the base stations, place the handsets in their cradles, and make a few test calls to ensure everything is working correctly.

1.6.6 Measuring Vector Similarity Using Cosine Similarity

Cosine similarity is a metric used to measure how similar two vectors are, regardless of their magnitude. It is particularly useful in the context of text analysis, information retrieval, and machine learning, where it is often used to compare the similarity of two text documents, sentences, or

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

any other data that can be represented as vectors. In our code, we will use it to match the similarity between the question we asked and the answer we receive.

- Lets create a function that calculates the cosine similarity between two vectors by using dot product a scalar value.

```
# Define the cosine similarity function
import numpy as np
# Define the cosine similarity function
def cosine_similarity(vec1, vec2):
    dot_product = np.dot(vec1, vec2)
    norm_vec1 = np.linalg.norm(vec1)
    norm_vec2 = np.linalg.norm(vec2)
    return dot_product / (norm_vec1 * norm_vec2)
```

- We will now convert the question into an embedding, retrieve the embedding of our question, and the embeddings of the documents retrieved by the retriever.

```
# Convert the question into an embedding
embedding_model = OpenAIEmbeddings(model="text-embedding-3-large")
question_embedding = embedding_model.embed_query(Question)
retrieved_docs = retriever.get_relevant_documents(Question)
```

- Lets calculate and print similarity between the question and each retrieved document

```
for doc in retrieved_docs:
    # Assuming your retriever does not provide embeddings directly, re-embed each document
    doc_embedding = embedding_model.embed_query(doc.page_content)
    similarity = cosine_similarity(question_embedding, doc_embedding)
    print(f"Document: {doc.page_content[:100]}...")  # Print a preview of the document
    print(f"Cosine Similarity: {similarity}\n")
```

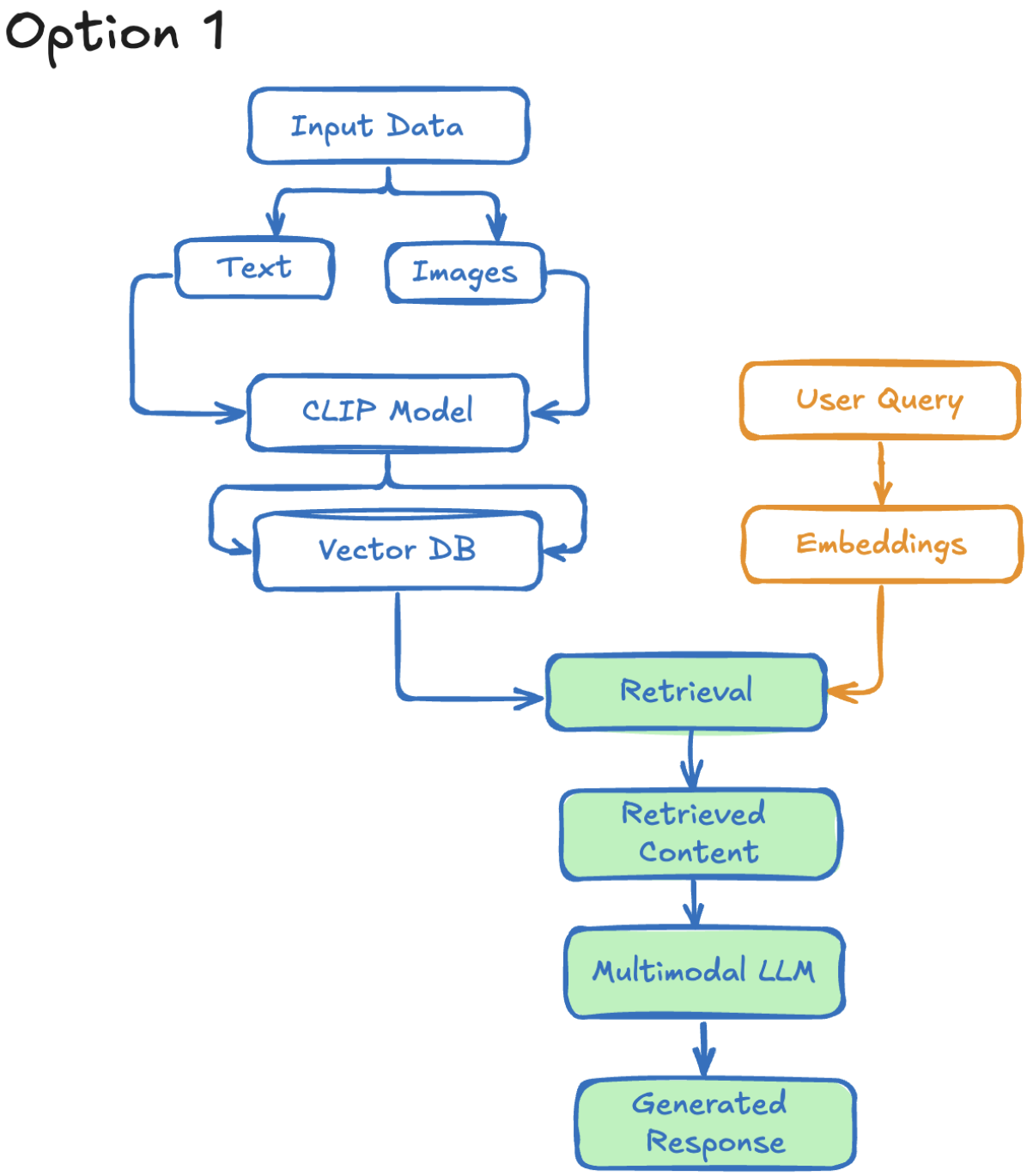
```
Document: 6
Quick Set up and Installation Process
This first section is a quick step-by-step guide. The deta...
Cosine Similarity: 0.6360771620271698

Document: 2. Upgrade the base stations to the desired firmware version, if necessary.
• The base stations and...
Cosine Similarity: 0.6281955726318081

Document: screen appears.

4. Press the Select softkey when the MAC address of the base station appears.
• ...
Cosine Similarity: 0.5779822517085555

Document: Plan for the Device Installation
These instructions are for the planner:
1. Review the site to ins...
Cosine Similarity: 0.5748595749246084
```

1.7 Multimodal RAG

In the previous section, we explored RAG systems primarily designed for handling text. However, in real-world scenarios, many documents within organizations contain valuable information in the form of images, tables, and other non-text elements. When building a robust RAG system, it's crucial to have the capability to retrieve not just text-based information but also relevant images and other visual data. This multimodal approach significantly enhances the effectiveness of information retrieval. In this section, we will explore methods to retrieve both images and text in response to user queries, enabling a more comprehensive and efficient RAG based system.

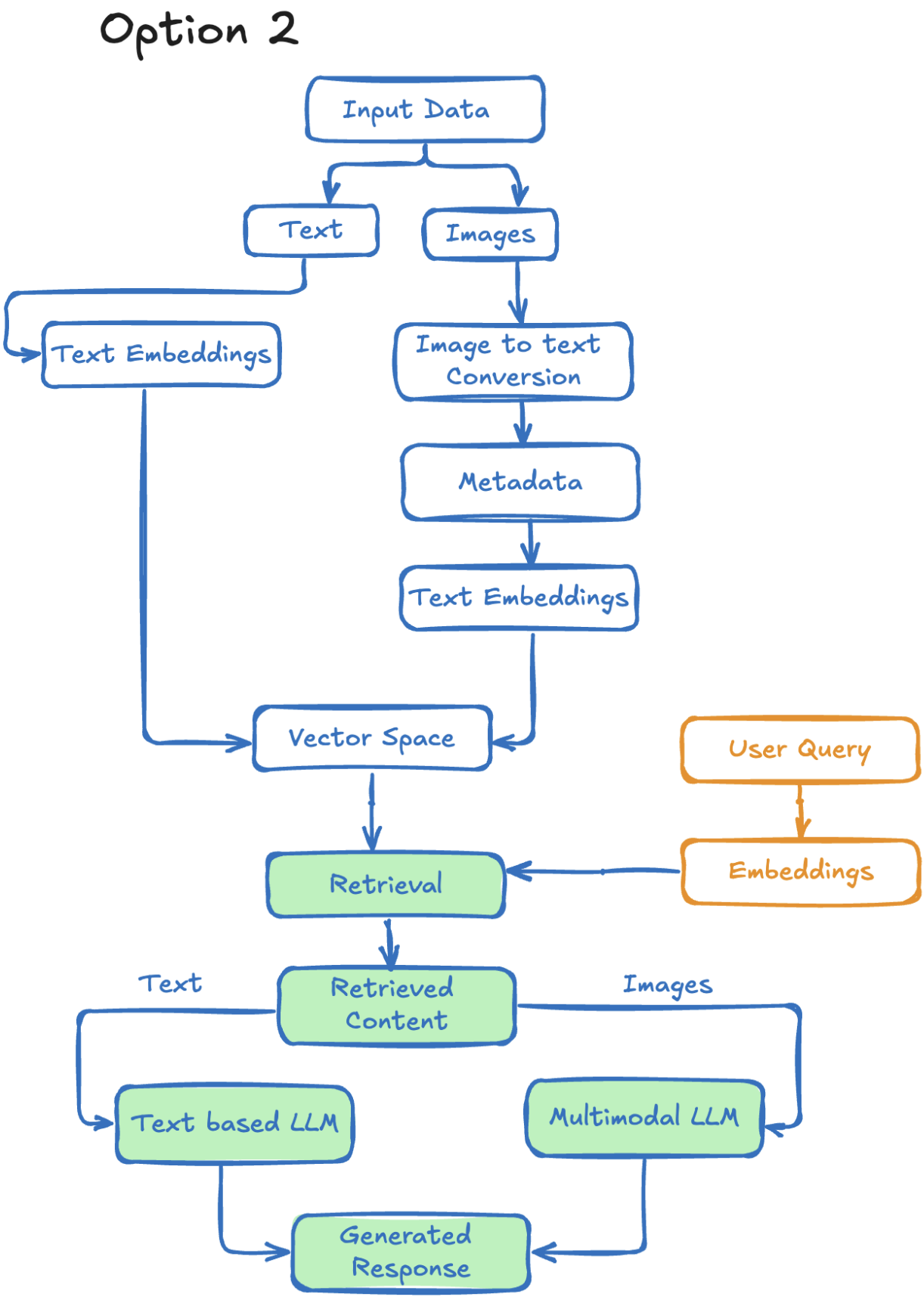
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

1.7.1 Three Different approaches for Multimodal RAG

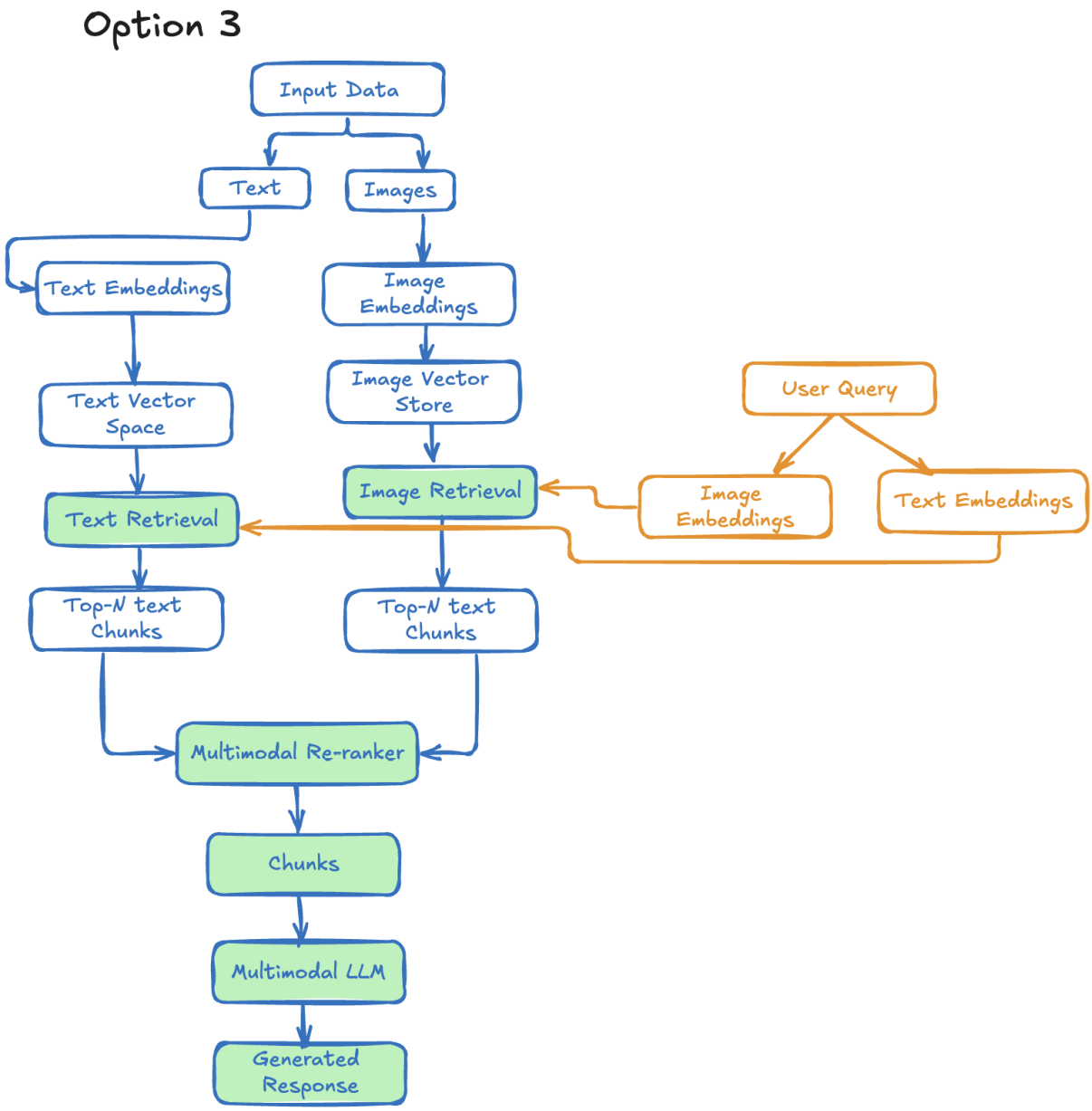


Embedding for images can be done via CLIP model.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



We will be using OpTion 2 for our lab



1.7.2 Contrastive Language-Image Pretraining - CLIP Model

Unlike other embedding models that focus on single modality (either text or images) the CLIP (Contrastive Language-Image Pretraining), was developed by OpenAI in 2021.

Note: CLIP models are embedding models, not traditional Large Language Models (LLMs).

The primary function of CLIP (Contrastive Language-Image Pretraining) is to create a shared embedding space where both text and images are represented as vectors. The key idea is to learn representations where images and their corresponding text descriptions are close to each other in this embedding space, while unrelated image-text pairs are far apart. This allows CLIP to perform tasks like image classification, zero-shot learning, and text-to-image retrieval based on the semantic similarity between text and images. So in summary CLIP models create a relationship between images and text.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

Zero-Shot Learning (ZSL)

Zero-Shot Learning (ZSL): This technique enhances the ability of AI systems to classify and recognize objects they have not been explicitly trained on. Instead of relying solely on trained data, ZSL uses auxiliary information to make predictions. For example, in an image classification scenario where a model is trained to recognize dogs and cats, if it encounters an image of a zebra (which the model has never seen before), it can still classify it correctly if it has auxiliary information describing a zebra as an animal with black and white stripes.

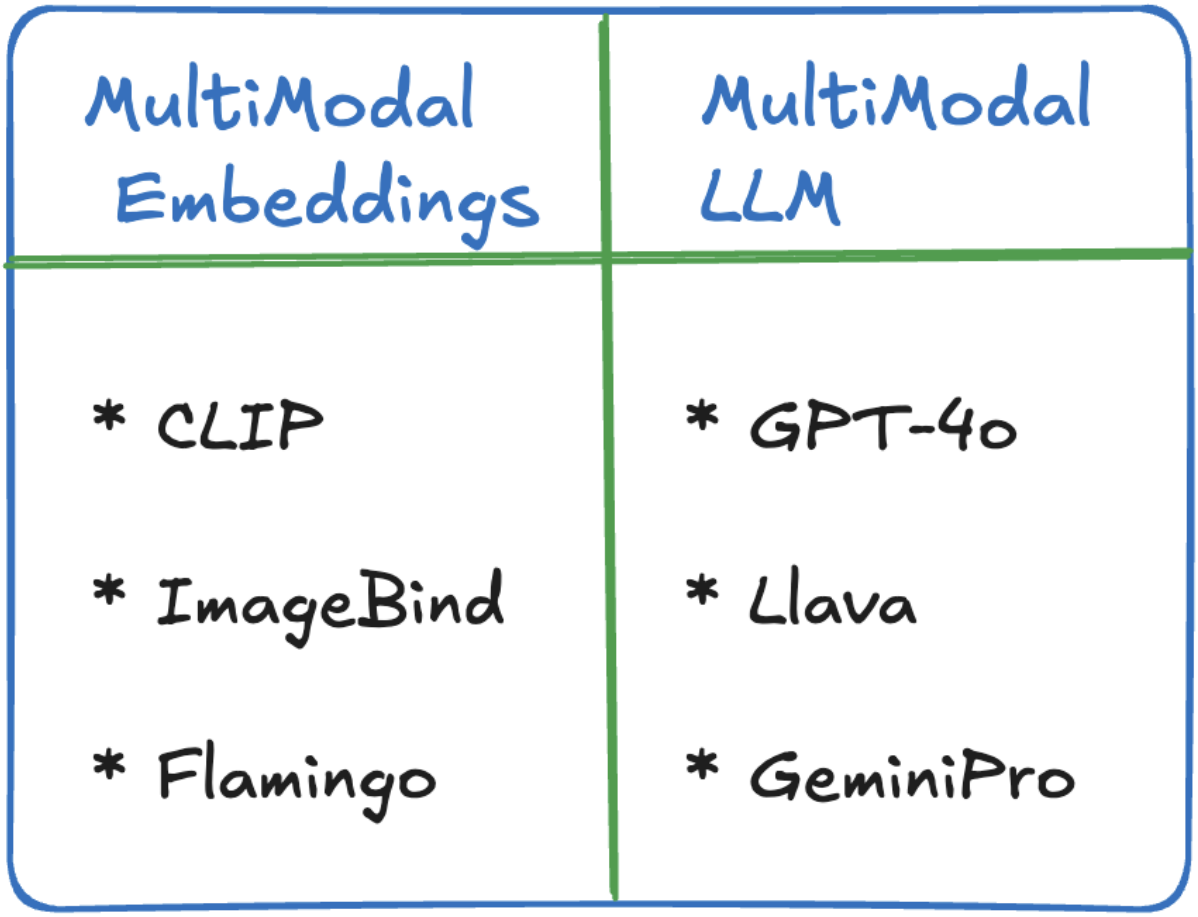
Using ZSL, CLIP models can perform tasks they were not specifically trained on, in real-time. Essentially, they have capabilities similar to text-based embedding models. Just as we create chunks of text and use a user query to find similar text chunks, the same concept applies to images using CLIP.

1.7.3 OpenCLIP Model

It's important to note that the original CLIP model by OpenAI is not publicly available. However, people have taken the concepts from OpenAI's CLIP paper and developed open-source models like OpenCLIP.

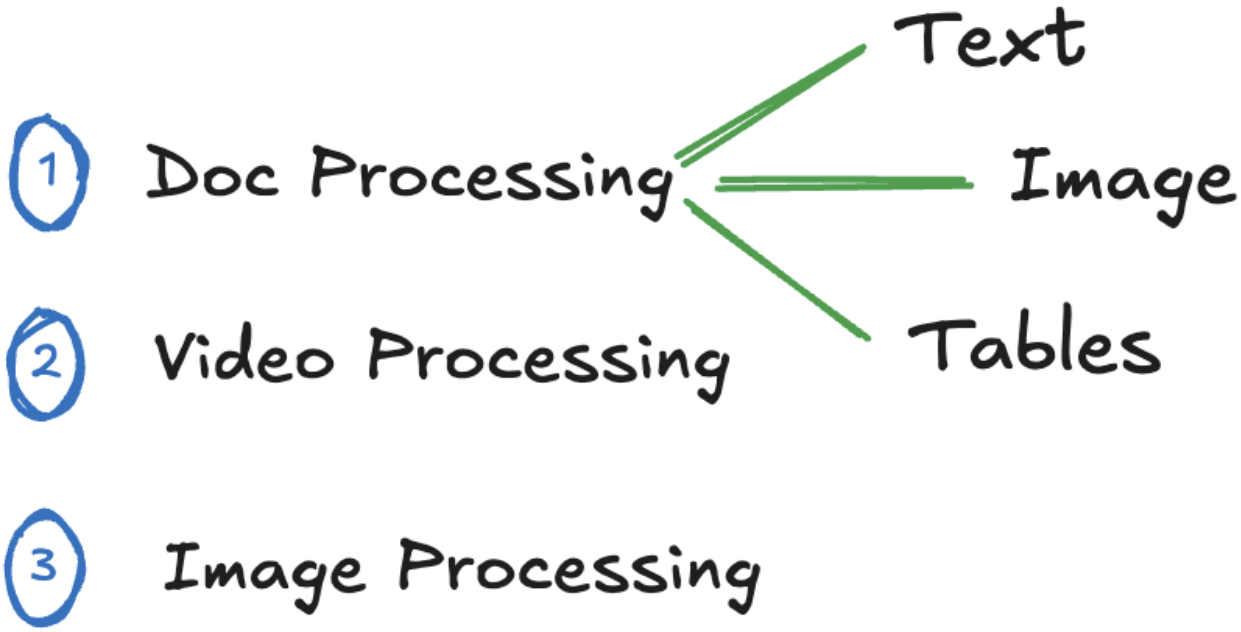
Note: OpenCLIP is an open-source implementation and extension of the original CLIP (Contrastive Language-Image Pretraining) model developed by OpenAI.

- Different types of Multimodal Embeddings and LLM



- Usecases for Multimodal RAG

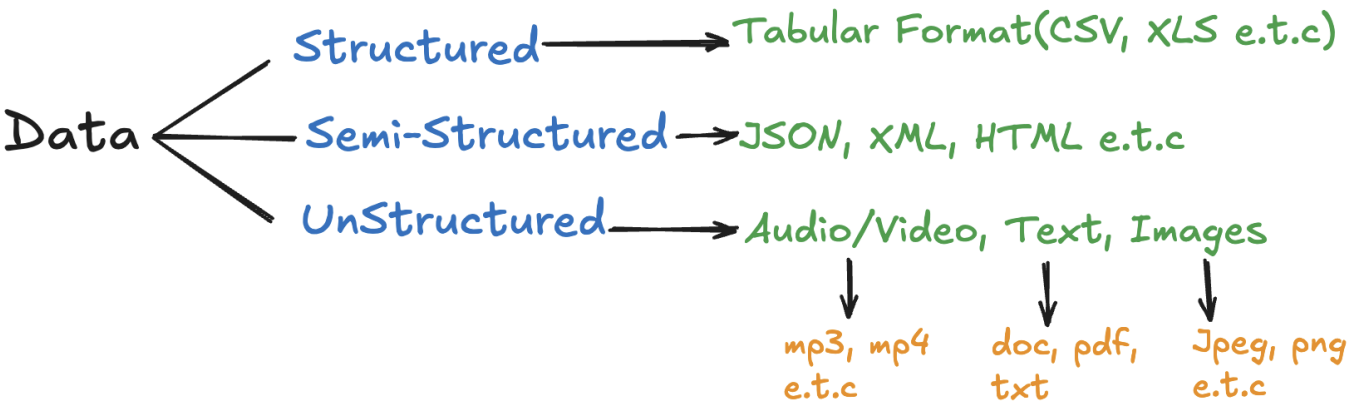
Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Lets contine and jump to our lab

1.7.4 Deploy Multimodal RAG

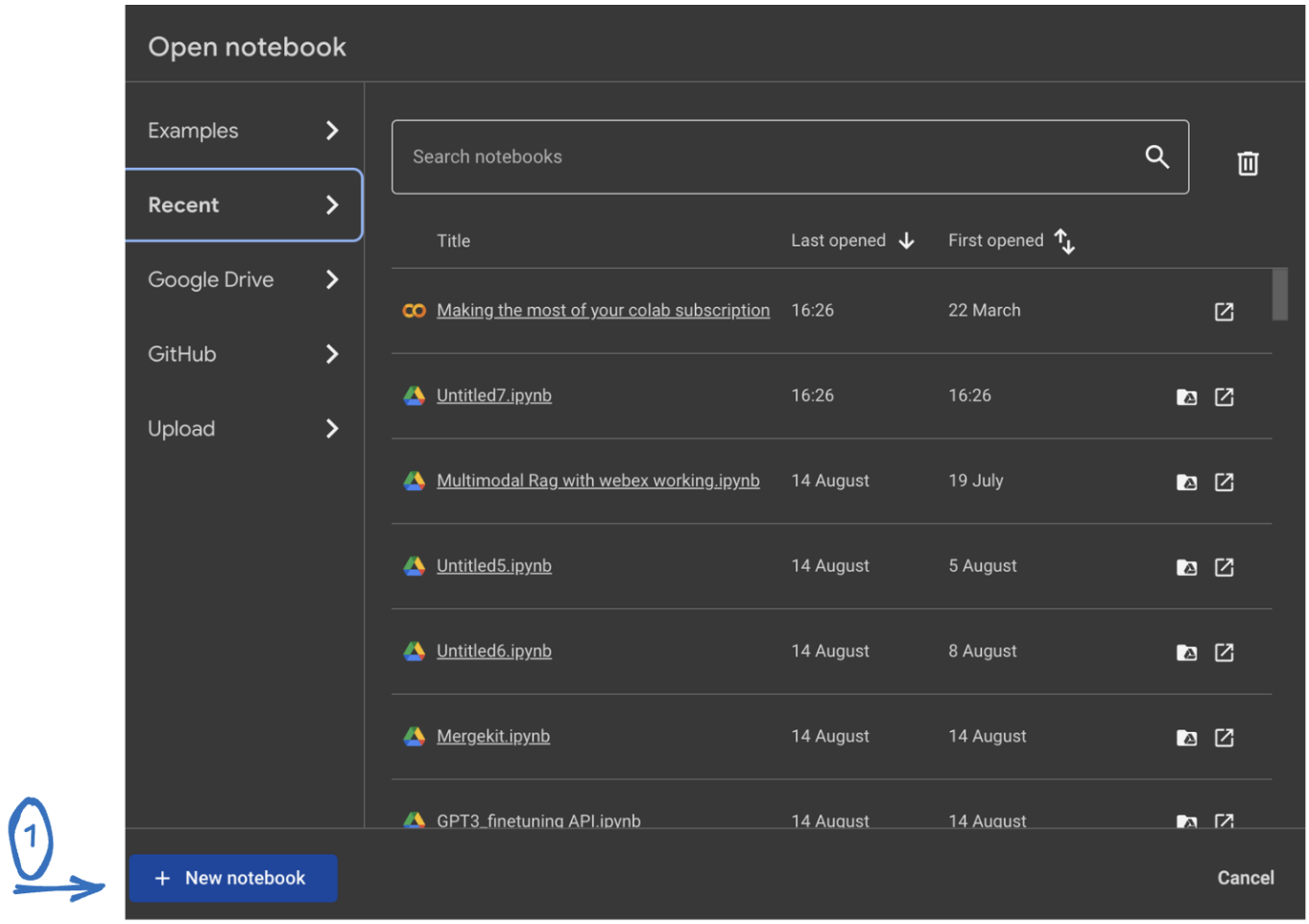
Whenever we develop applications based on Large Language Models (LLMs), handling data is a crucial aspect. As discussed in the previous section (RAG), data injection is the initial step in this process. It's important to note that there are typically three types of data structure.



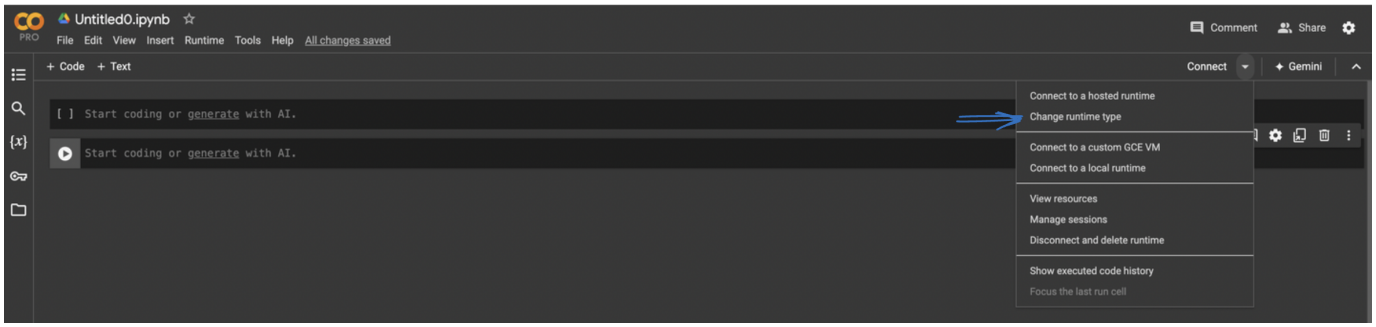
1.7.5 Task 1: Log into the Lab Environment

- Open Google Colab and create a new notebook. Click on "File" > "New notebook". Please refer to the following section to create Google Colab account.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

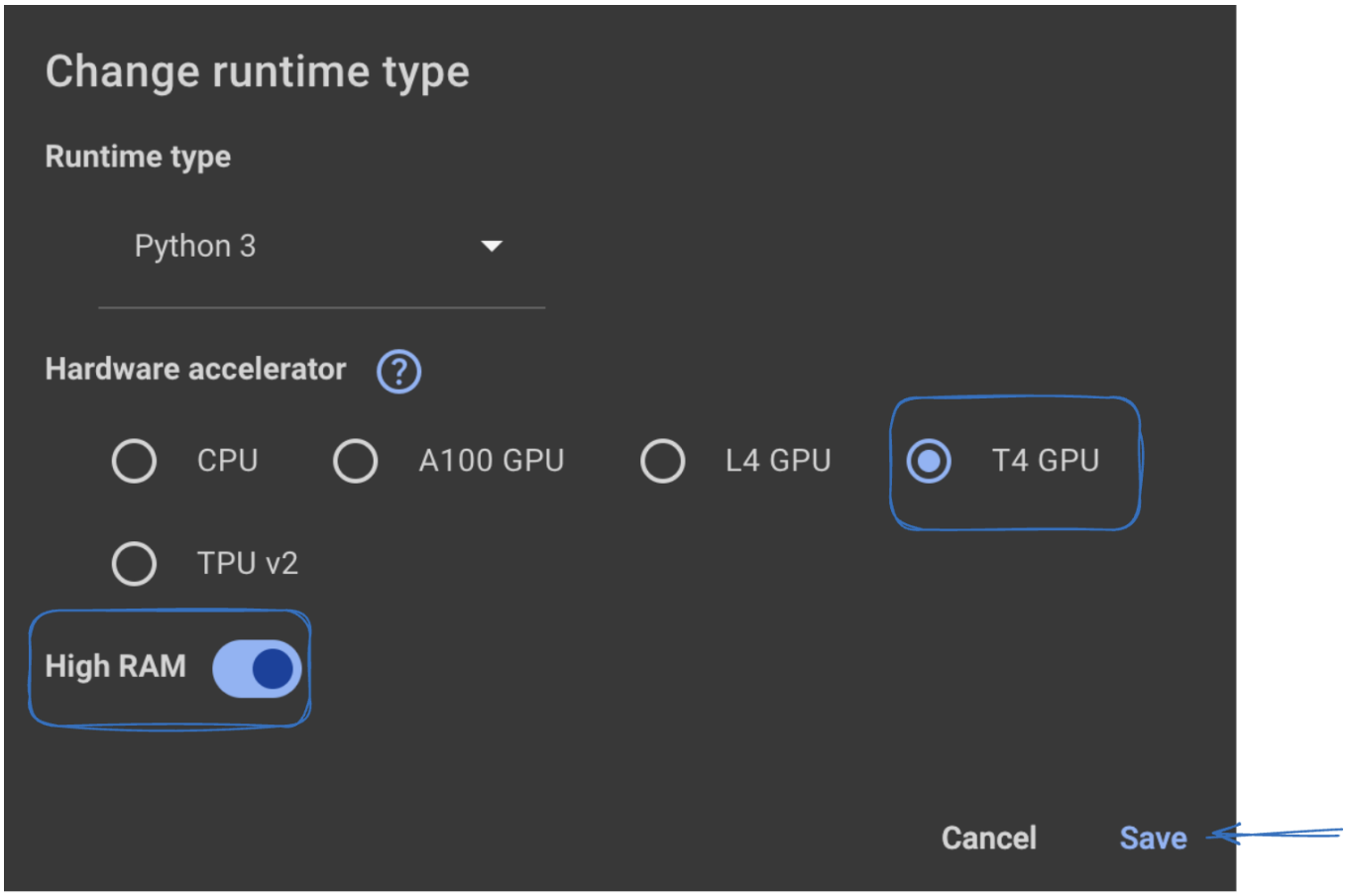


- Change Runtime Environment: Click the “Runtime” dropdown menu at the top of the Colab interface.



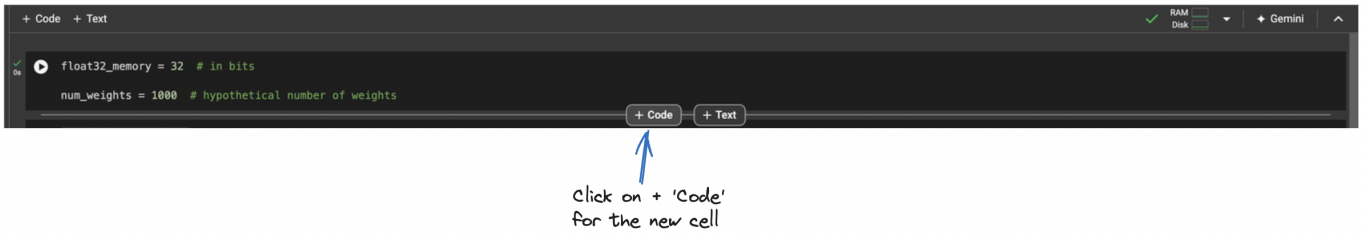
- Select “Change runtime type”: This will open a dialog box where you can configure the runtime environment.
- Select Hardware Accelerator: From the “Hardware accelerator” dropdown menu, choose >> T4 GPU and enable toggle for High RAM

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

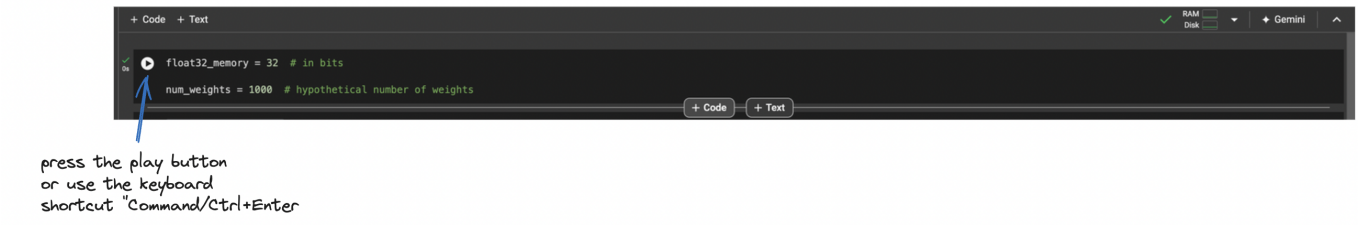


- Save Settings: Click “Save” to apply the changes.

Reminder: Whenever you want to copy the code in Google Colab and run it, be sure to click on + Code to add a new code cell.



Reminder: Click the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter" while the cell is selected.

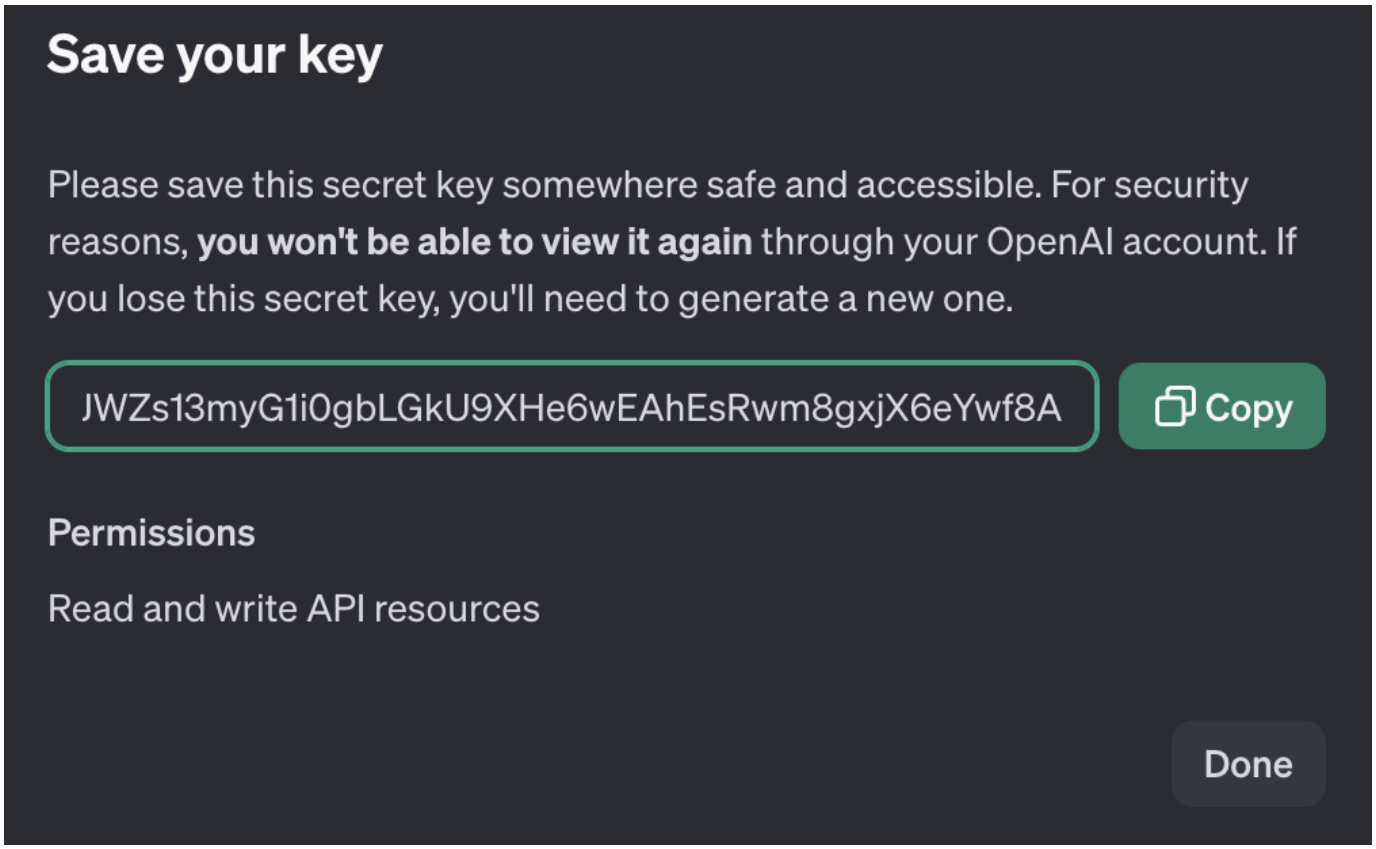


Set OpenAI token - MultiModal

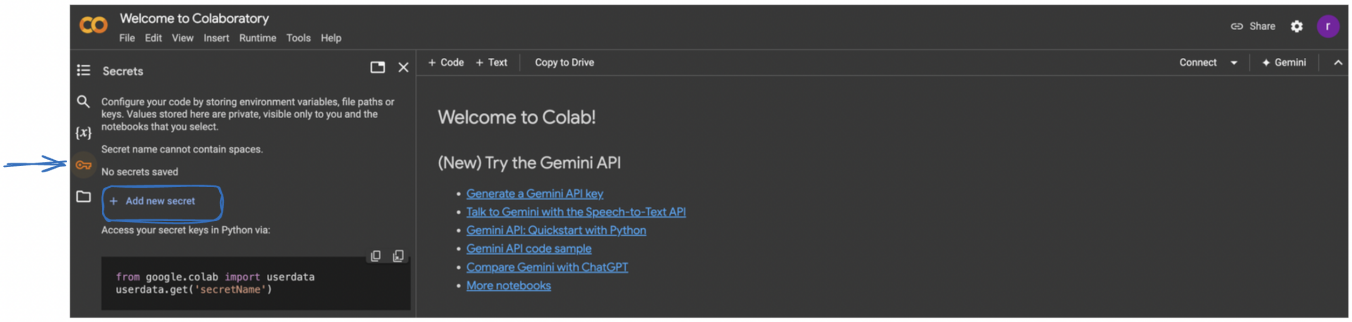
Note: First, create an account from the OpenAI official website.If you have already completed this step please ignore and jump to Update Google Colab envoirnment section.

- Create a new project API key by browsing to API Keys web page. Select Create new secret key. The API key is automatically generated. Save the APi Key as we will be using it in the later steps .

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

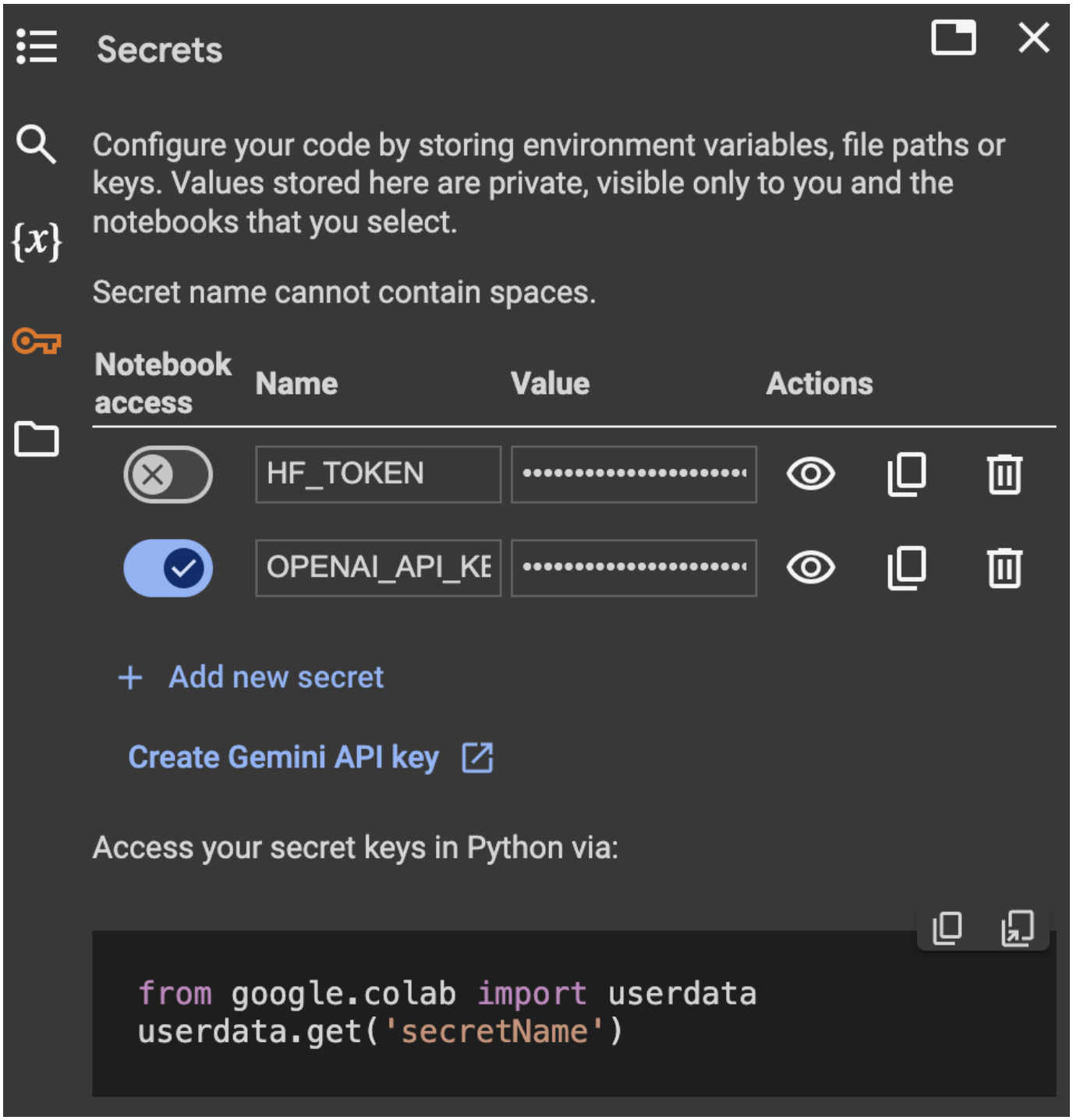


- Within your existing Google Colab notebook navigate to the new “Secrets” section in the sidebar.



- Click on “Add a new secret.” Enter the name example: OPENAI_API_KEY and value of the secret(the API key created above). Note: The name is permanent once set.
- The list of secrets is global across all your notebooks.
- Use the “Notebook access” toggle to grant or revoke access to a secret for each notebook.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Update Google Colab envoirnment and install packages

PDFs often contain tables and other structured data that can be challenging to split accurately using character-based embedding techniques. For PDFs, it's important to extract and chunk all elements, including tables, effectively. We'll accomplish this using the Unstructured library, which is specifically designed for handling such tasks. If you have a large collection of PDFs, Unstructured is an excellent tool to manage them efficiently.

- Install relevant libraries including Unstructured - ELT tool. Unstructured will partition PDF files by first removing all embedded image blocks. Then it will use a layout model (YOLOX) to get bounding boxes (for tables) as well as titles.

```
!pip install "unstructured[all-docs]" pillow pydantic lxml matplotlib langchain langchain_community chromadb langchain-experimental langchain_openai
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

```
Downloading requests_toolbelt-1.0.0-py2.py3-none-any.whl (54 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.5/54.5 kB 4.1 MB/s eta 0:00:00
Downloading timm-1.0.9-py3-none-any.whl (2.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 62.6 MB/s eta 0:00:00
Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (176.2 MB)
Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
Downloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Downloading XlsxWriter-3.2.0-py3-none-any.whl (159 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 159.9/159.9 kB 2.4 MB/s eta 0:00:00
Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Downloading layoutparser-0.3.4-py3-none-any.whl (19.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.2/19.2 MB 54.8 MB/s eta 0:00:00
Downloading olefile-0.47-py2.py3-none-any.whl (114 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.6/114.6 kB 5.7 MB/s eta 0:00:00
Downloading python_multipart-0.0.9-py3-none-any.whl (22 kB)
Downloading ordered_set-4.1.0-py3-none-any.whl (7.6 kB)
Downloading coloredlogs-15.0.1-py2.py3-none-any.whl (46 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 3.3 MB/s eta 0:00:00
Downloading pdfplumber-0.11.4-py3-none-any.whl (59 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.2/59.2 kB 4.8 MB/s eta 0:00:00
Downloading pdfminer.six-20231228-py3-none-any.whl (5.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.6/5.6 MB 62.9 MB/s eta 0:00:00
Downloading h11-0.14.0-py3-none-any.whl (58 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 4.7 MB/s eta 0:00:00
Downloading humanfriendly-10.0-py2.py3-none-any.whl (86 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 4.9 MB/s eta 0:00:00
Downloading pypdfium2-4.30.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 56.7 MB/s eta 0:00:00
Using cached nvidia_nvjitlink_cu12-12.6.20-py3-none-manylinux2014_x86_64.whl (19.7 MB)
Downloading portalocker-2.10.1-py3-none-any.whl (18 kB)
```

Note: Make sure to restart your notebook after installing teh packages

- Since Google Colab is built on top of an Ubuntu environment, it's necessary to update the Google Colab environment to ensure we can effectively extract information from images or PDFs for analysis and processing.

```
1  !sudo apt-get update
```

```
Get:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,626 B]
Get:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  InRelease [1,581 B]
Ign:3 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  Packages [945 kB]
Get:7 https://r2u.stat.illinois.edu/ubuntu jammy Release [5,713 B]
Get:8 https://r2u.stat.illinois.edu/ubuntu jammy Release.gpg [793 B]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:10 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Get:11 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [8,232 kB]
Get:12 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease [24.3 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Hit:14 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Get:15 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,560 kB]
Get:16 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy/main amd64 Packages [53.3 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,498 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,425 kB]
Fetched 16.1 MB in 2s (7,586 kB/s)
Reading package lists... Done
```

- Poppler-utils will help us extracting info from our pdf

```
1  !sudo apt-get install poppler-utils
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  poppler-utils
0 upgraded, 1 newly installed, 0 to remove and 58 not upgraded.
Need to get 186 kB of archives.
After this operation, 696 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 poppler-utils amd64 22.02.0-2ubuntu0.5 [186 kB]
Fetched 186 kB in 0s (458 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78, <> line
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package poppler-utils.
(Reading database ... 123595 files and directories currently installed.)
Preparing to unpack .../poppler-utils_22.02.0-2ubuntu0.5_amd64.deb ...
Unpacking poppler-utils (22.02.0-2ubuntu0.5) ...
Setting up poppler-utils (22.02.0-2ubuntu0.5) ...
Processing triggers for man-db (2.10.2-1) ...
```

```
!sudo apt-get install libleptonica-dev tesseract-ocr libtesseract-dev python3-pil tesseract-ocr-eng tesseract-ocr-script-latn
```

```
Unpacking libtesseract-dev:amd64 (4.1.1-2.1build1) ...
Selecting previously unselected package mailcap.
Preparing to unpack .../05-mailcap_3.70+nmu1ubuntu1_all.deb ...
Unpacking mailcap (3.70+nmu1ubuntu1) ...
Selecting previously unselected package mime-support.
Preparing to unpack .../06-mime-support_3.66_all.deb ...
Unpacking mime-support (3.66) ...
Selecting previously unselected package python3-olefile.
Preparing to unpack .../07-python3-olefile_0.46-3_all.deb ...
Unpacking python3-olefile (0.46-3) ...
Selecting previously unselected package python3-pil:amd64.
Preparing to unpack .../08-python3-pil_9.0.1-1ubuntu0.3_amd64.deb ...
Unpacking python3-pil:amd64 (9.0.1-1ubuntu0.3) ...
Selecting previously unselected package tesseract-ocr-eng.
Preparing to unpack .../09-tesseract-ocr-eng_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr-osd.
Preparing to unpack .../10-tesseract-ocr-osd_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-osd (1:4.00~git30-7274cfa-1.1) ...
Selecting previously unselected package tesseract-ocr.
Preparing to unpack .../11-tesseract-ocr_4.1.1-2.1build1_amd64.deb ...
Unpacking tesseract-ocr (4.1.1-2.1build1) ...
Selecting previously unselected package tesseract-ocr-script-latn.
Preparing to unpack .../12-tesseract-ocr-script-latn_1%3a4.00~git30-7274cfa-1.1_all.deb ...
Unpacking tesseract-ocr-script-latn (1:4.00~git30-7274cfa-1.1) ...
Setting up tesseract-ocr-script-latn (1:4.00~git30-7274cfa-1.1) ...
Setting up python3-olefile (0.46-3) ...
Setting up tesseract-ocr-eng (1:4.00~git30-7274cfa-1.1) ...
Setting up libleptonica-dev (1.82.0-3build1) ...
Setting up libraqm0:amd64 (0.7.0-4ubuntu1) ...
```

```
!pip install unstructured-pytesseract
!pip install tesseract-ocr
```

```
Requirement already satisfied: unstructured-pytesseract in /usr/local/lib/python3.10/dist-packages (0.3.13)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from unstructured-pytesseract) (24.1)
Requirement already satisfied: Pillow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from unstructured-pytesseract) (10.4.0)
Collecting tesseract-ocr
  Downloading tesseract-ocr-0.0.1.tar.gz (33 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: cython in /usr/local/lib/python3.10/dist-packages (from tesseract-ocr) (3.0.11)
Building wheels for collected packages: tesseract-ocr
  Building wheel for tesseract-ocr (setup.py) ... done
  Created wheel for tesseract-ocr: filename=tesseract_ocr-0.0.1-cp310-cp310-linux_x86_64.whl size=169769 sha256=ce48d78777bf67501e8e60975bf1e6bfb0b3d729b7d1d2b8bb1
  Stored in directory: /root/.cache/pip/wheels/bb/fd/f3/5c231ecbbb80a1fe33204ff3021d99b54ef6daf6f8099311b8
Successfully built tesseract-ocr
Installing collected packages: tesseract-ocr
Successfully installed tesseract-ocr-0.0.1
```

- In Google Colab, I noticed some challenges with the NLTK library, so let's set the path correctly to resolve these issues.

```
import os
# Set the NLTK_DATA environment variable to the correct path
os.environ['NLTK_DATA'] = '/root/nltk_data'
```

- Lets make sure we have nltk version 3.9.1 installed

```
pip install --upgrade nltk
```


Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com

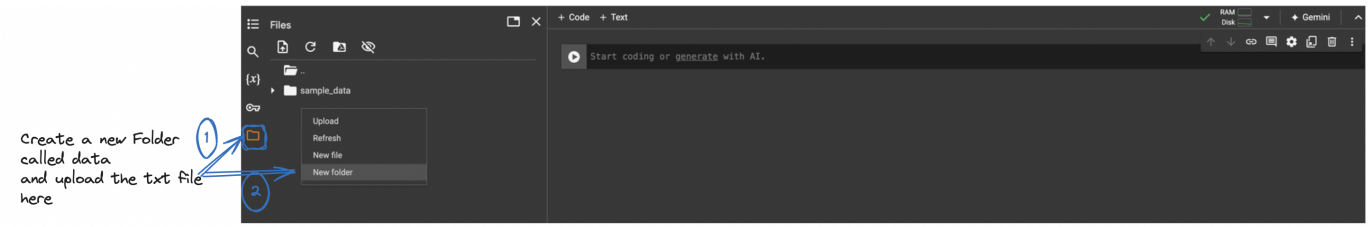
```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Collecting nltk
  Downloading nltk-3.9.1-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.5)
Downloading nltk-3.9.1-py3-none-any.whl (1.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 24.8 MB/s eta 0:00:00
Installing collected packages: nltk
  Attempting uninstall: nltk
    Found existing installation: nltk 3.8.1
    Uninstalling nltk-3.8.1:
      Successfully uninstalled nltk-3.8.1
Successfully installed nltk-3.9.1
```

PDF with table

- Let's load our PDF files into Google Colab. For this example, we can use the article titled "Webex Customer Experience Essentials" from the Webex Help Center. You can also download the modified article here as we will be using in the next step.

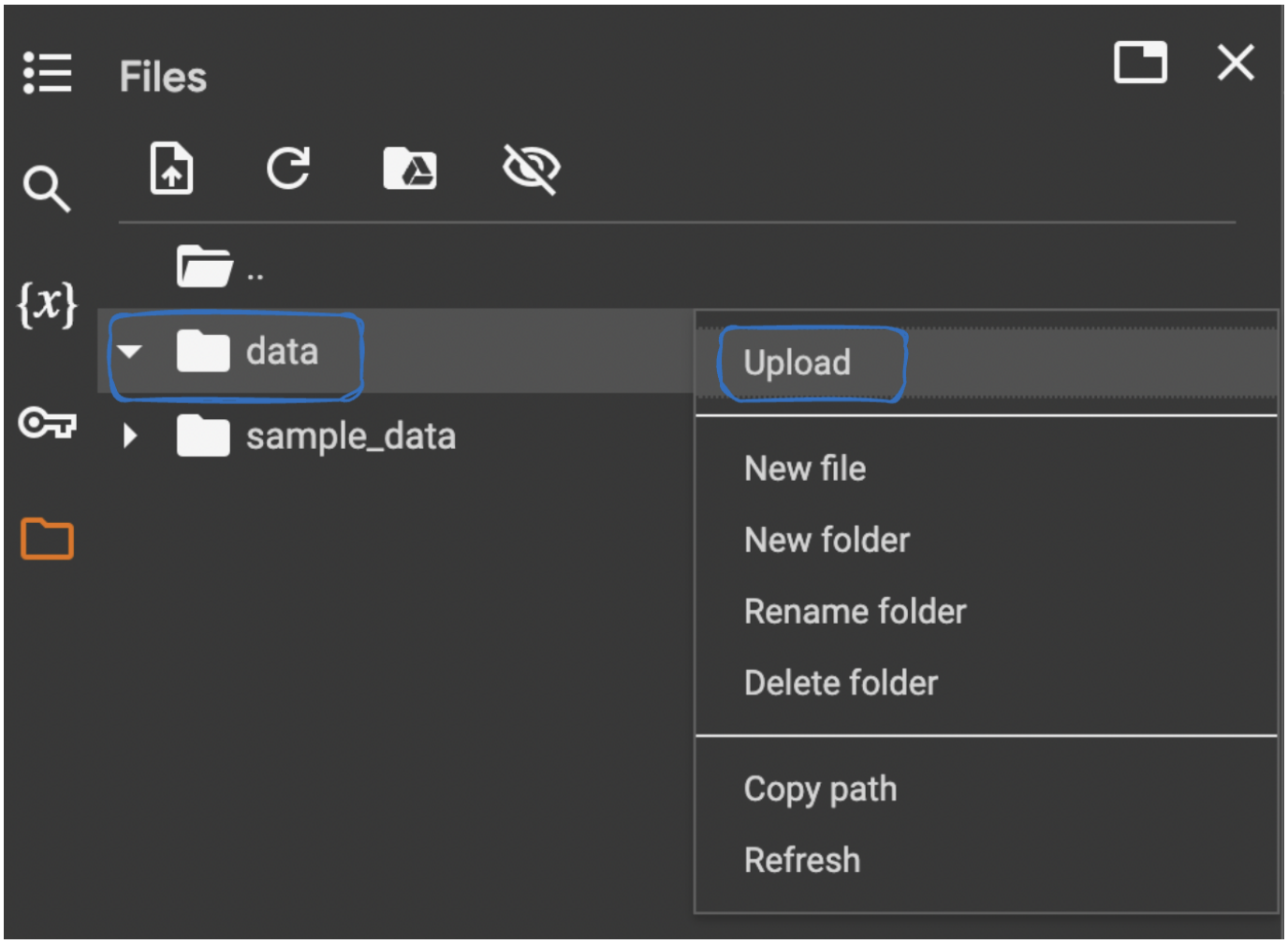
Note: To save time and speed up processing in this lab, I have modified the CX-Essentials.pdf file by reducing it to only a few pages. You can download this version from the modified link provided above.

- Within Google Colab, Click on Folder and create a new folder called "dat"



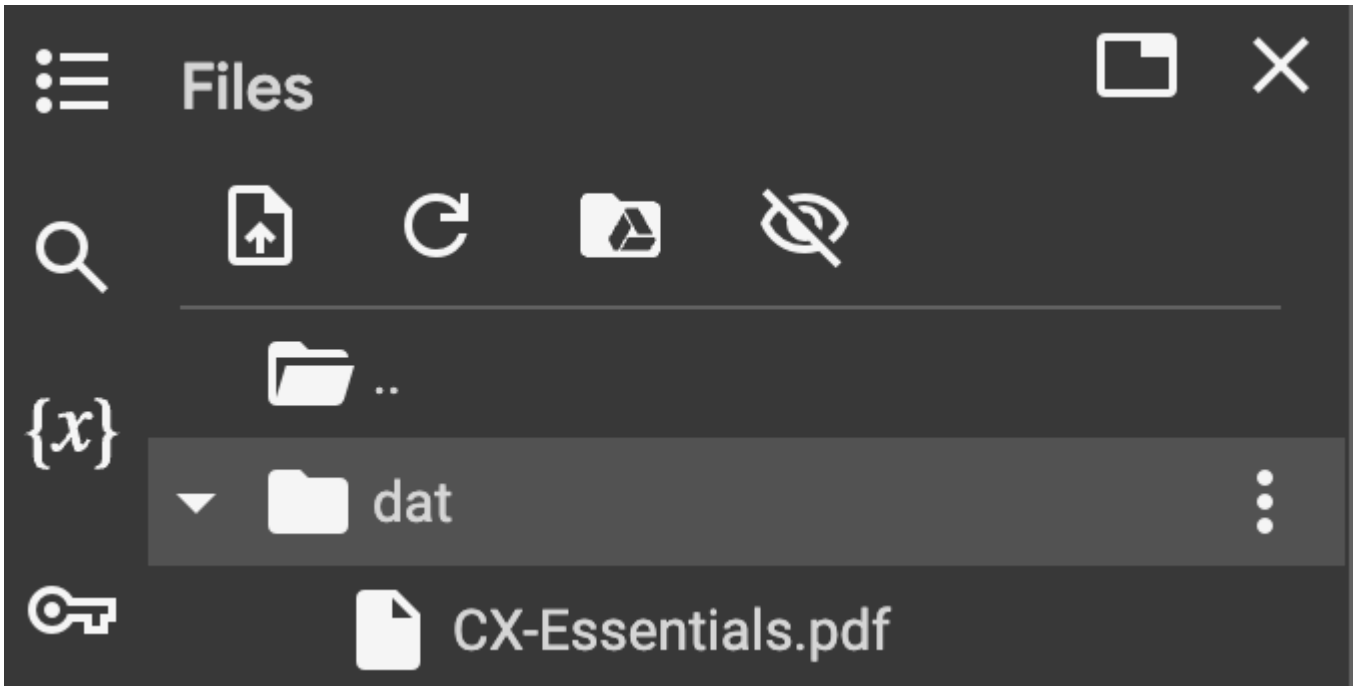
- Click on [...], select Upload

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Choose your CX-Essentials.pdf file and click Open

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



- Let's process our PDF document by splitting it into smaller, manageable chunks based on titles, extracting images, and handling text in a way that ensures the chunks are neither too large nor too small. We will store the extracted elements (text, images, etc.) in the pdf_elements variable, and images are saved in the specified path(image_path). To understand more about unstructured partition_pdf click here

Unstructured will partition PDF files by first extracting embedded images if specified. It then processes the document's layout, dividing the content into structured elements such as titles, tables, and paragraphs based on the layout and text structure. For more detailed layout analysis, it may use advanced models like YOLOX to identify and extract bounding boxes for elements like tables and titles. This allows for the precise extraction and organization of text and images, making the content more suitable for further analysis and processing.

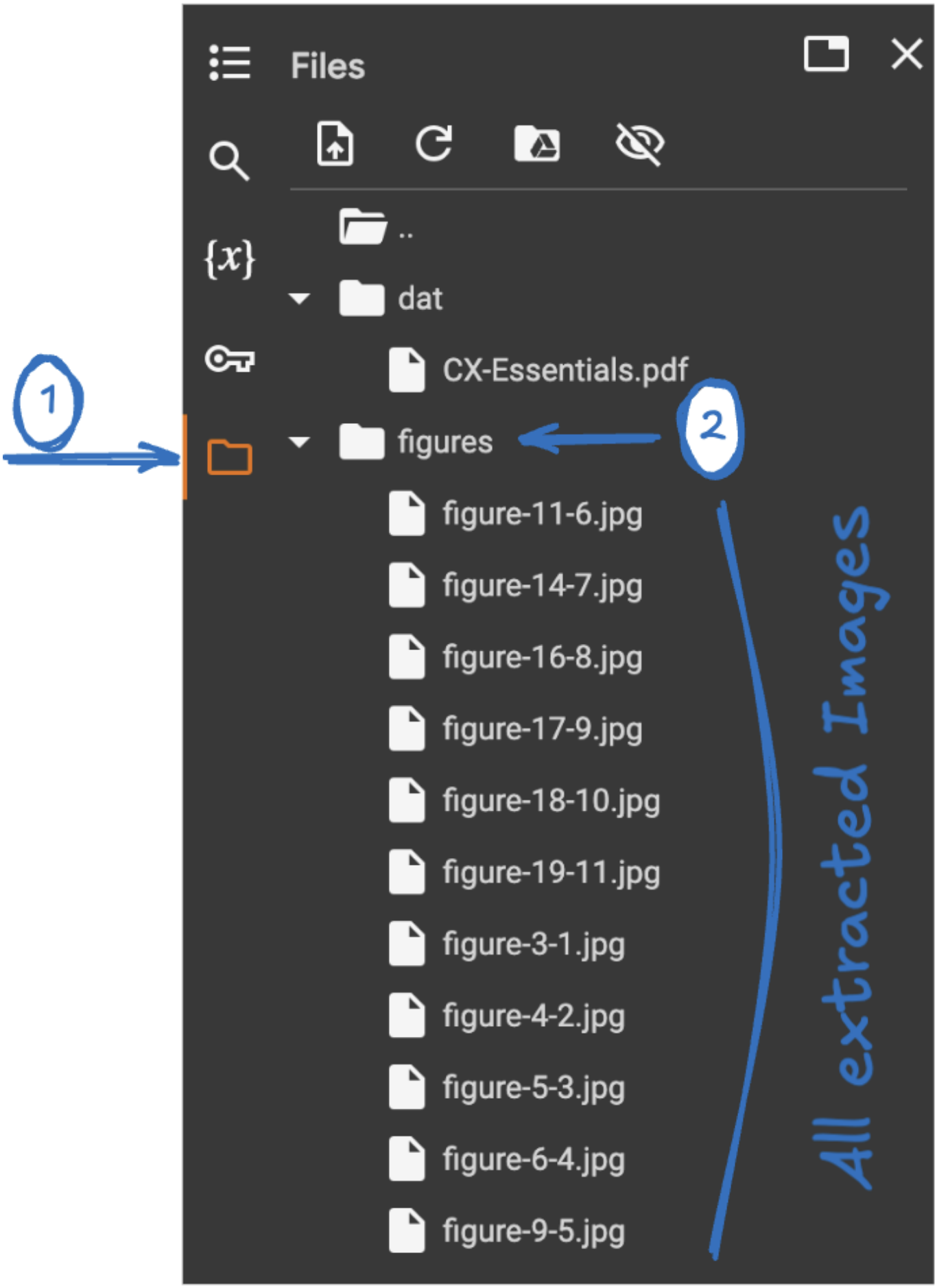
```
from unstructured.partition.pdf import partition_pdf
image_path = "./content/images"
pdf_elements = partition_pdf(
    filename="/content/dat/CX-Essentials.pdf",
    chunking_strategy="by_title",
    extract_images_in_pdf=True,
    # extract_image_block_types=["Image", "Table"],
    max_characters=3000, # Sets the maximum number of characters
    new_after_n_chars=2800, # Character threshold after which a new chunk will start. Ensures  chunks are created before the max_characters limit is reached
    combine_text_under_n_chars=2000, #If chunk of text is smaller than 2000 characters, it should be combined with the following chunk to create a more substantial piece
    image_output_dir_path=image_path
)
```

```
yolox_l0.05.onnx: 100% 217M/217M [00:05<00:00, 35.1MB/s]
```

Note: This process may take a few minutes to complete as we are extracting information from the PDF.

- You'll also notice that a folder has been created containing all the extracted images.

Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com



Omer Ilyas - Technical Marketing Engineer - oilyas@cisco.com