Azure API Setup

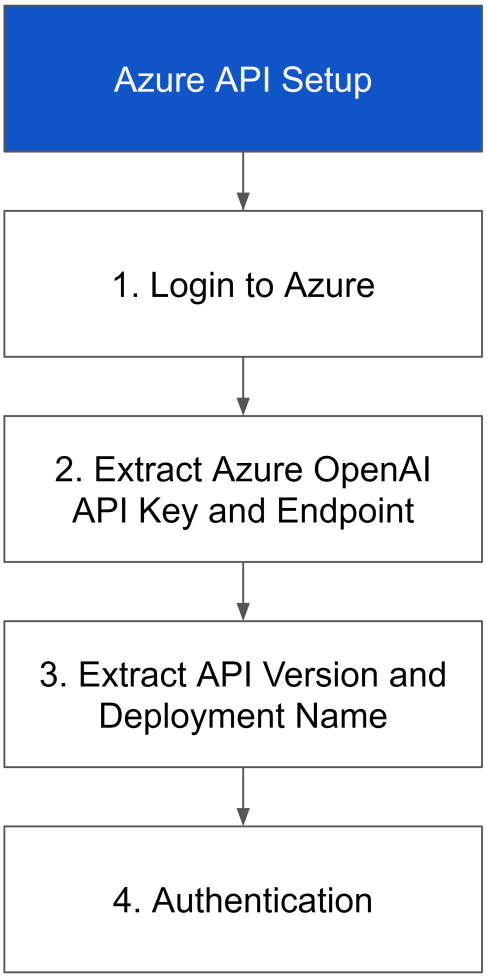


Figure 1

Section Overview: *This section will focus on preparing the groundwork before we dive into prompt engineering. We'll walk through* ***setting up the API*** *that gives us access to th LLM service, which is essential for our task. This process includes obtaining and verifyin the necessary credentials.*

To setup the API, we'll need an Azure account. We'll start by ***getting Azure account details*** *through Olympus. After that, we'll use these details to log in to Azure, where we'l* ***set up the OpenAI resource.*** *Within this service, we'll deploy the GPT model and* ***get the credentials****, which we'll later* ***authenticate*** *within the Colab notebook.*

Let's proceed!

***login credentials**. Let's see how we can obtain these credentials through Olympus:*

- Log in to the **Olympus** and on the Dashboard, you will find the **Labs** icon, mentioning Azure Labs.

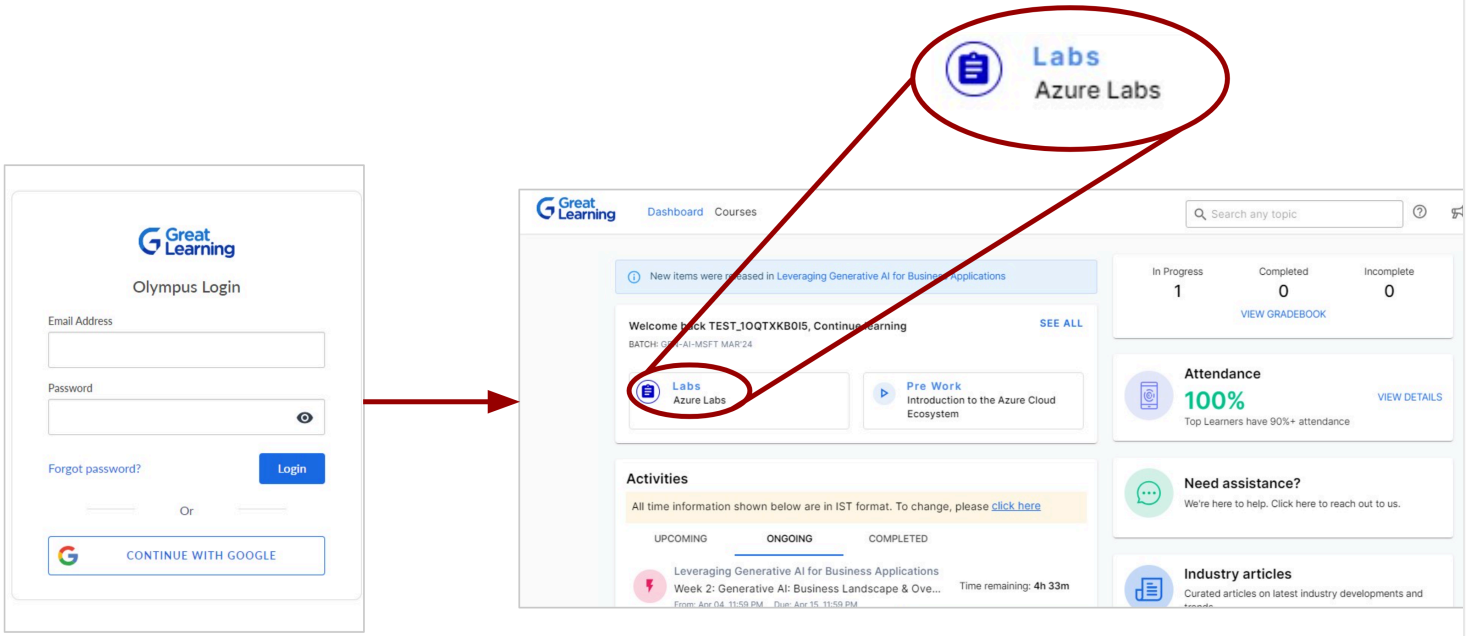
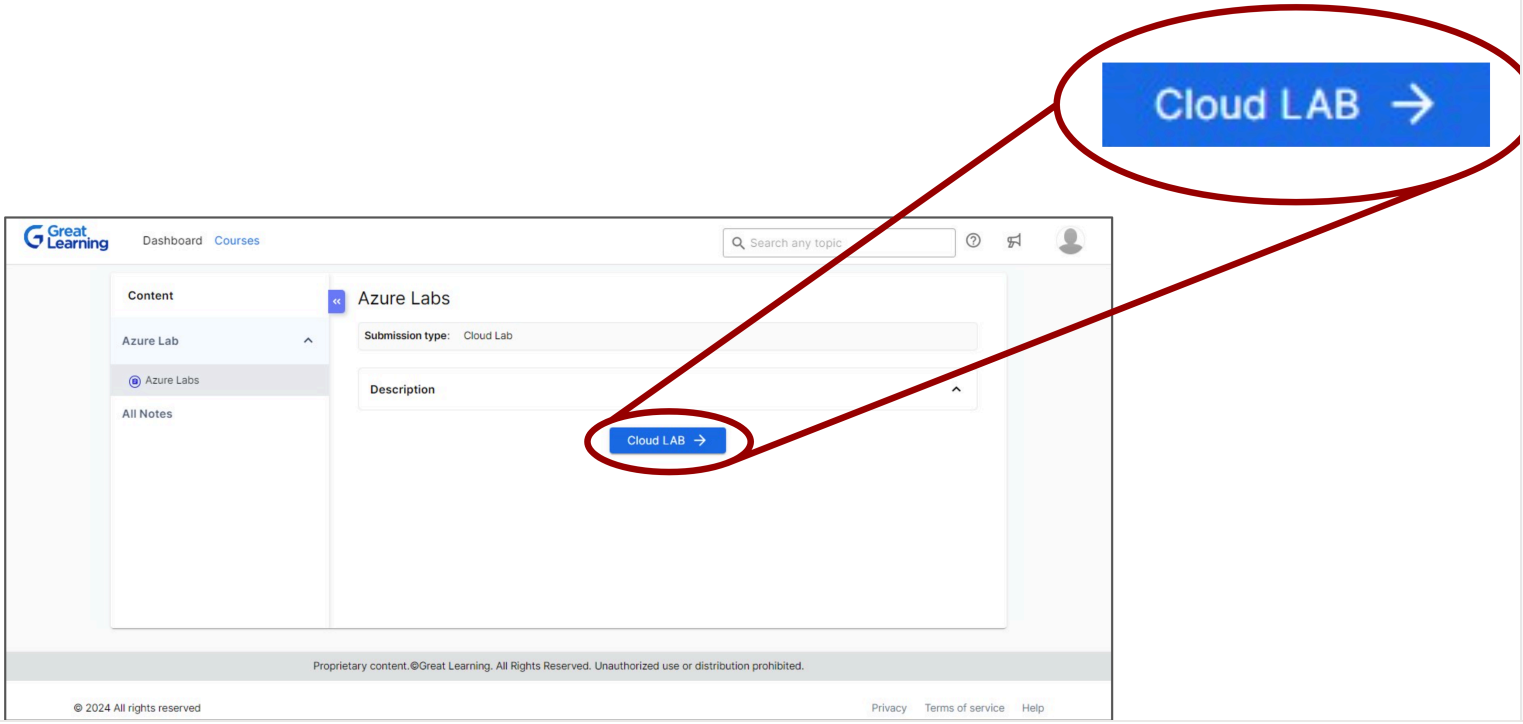


Figure 2

- Upon reaching the Azure Labs page, locate the **Cloud Lab** button and proceed by clicking on it.



Microsoft Azure Portal.

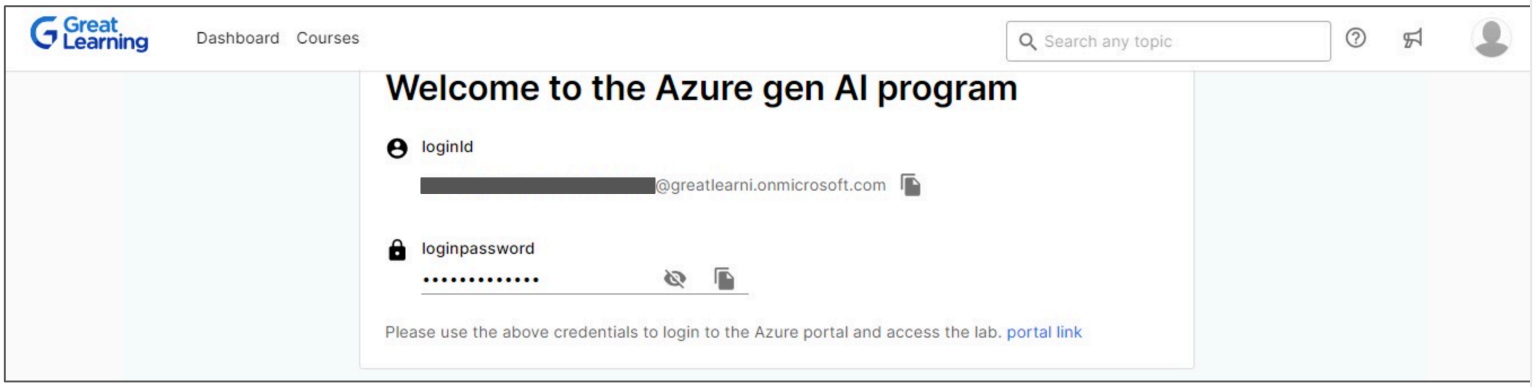


Figure 4

Now that we have our Microsoft Azure login details, our next task is to log in to **Microsoft Azure Portal** and get the necessary components to setup OpenAI: **API Key**, **Endpoint**, **Deployment Name**, and **API version**. These credentials are vital for setting up access to OpenAI in the code notebook. Let's proceed with obtaining them.

2. Extract Azure OpenAI Key and Endpoint

Section Overview: *To set up OpenAI API in your code, you'll require the following components:*

- AZURE_OPENAI_KEY
- AZURE_OPENAI_ENDPOINT
- CHATGPT_MODEL
- AZURE_OPENAI_APIVERSION

*In this section, we'll focus on obtaining the **OpenAI Key** and **Endpoint**. In the next section (section 3), we'll cover acquiring the **API version** and the **ChatGPT model**.*

To obtain the API Key and endpoint, we need to create a new OpenAI resource on the Azure portal, provided that you haven't already created any OpenAI resource. Let's proceed with creating it.

- Login to **Microsoft Azure Portal** using azure credentials.
- Once you're on the login dashboard, locate and click on the **Resource Group** Icon. A resource

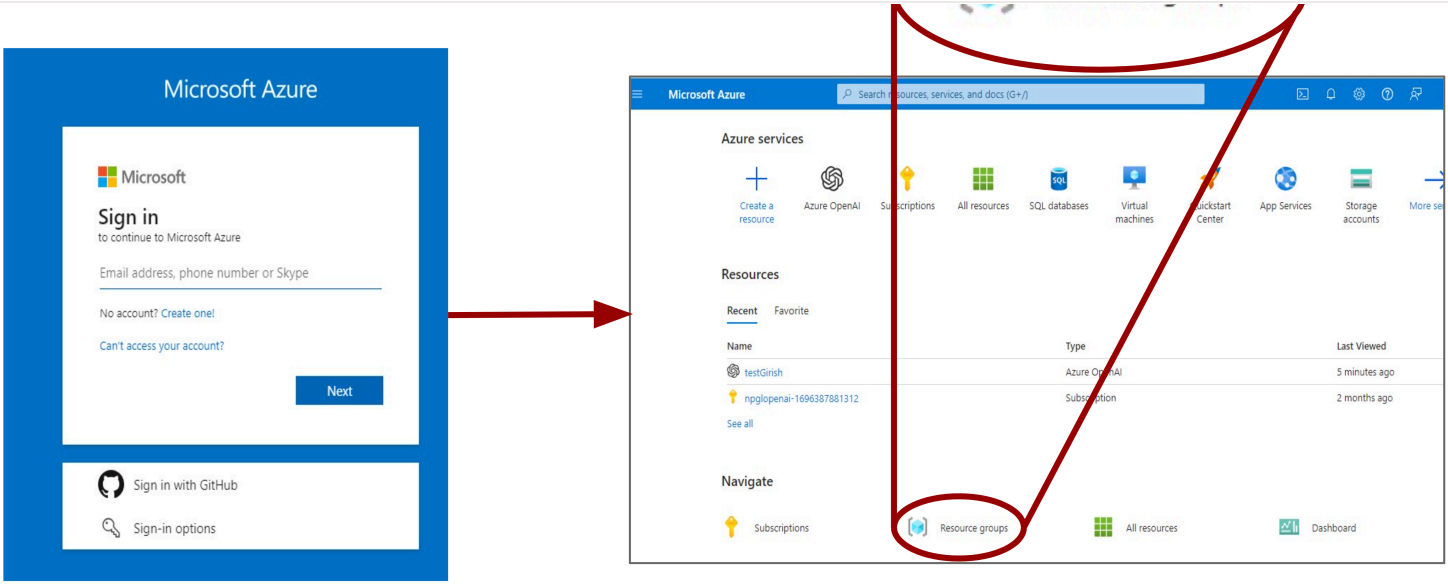


Figure 5

- On the resource group page, you'll find the option to select the **default resource group**.Click on it.

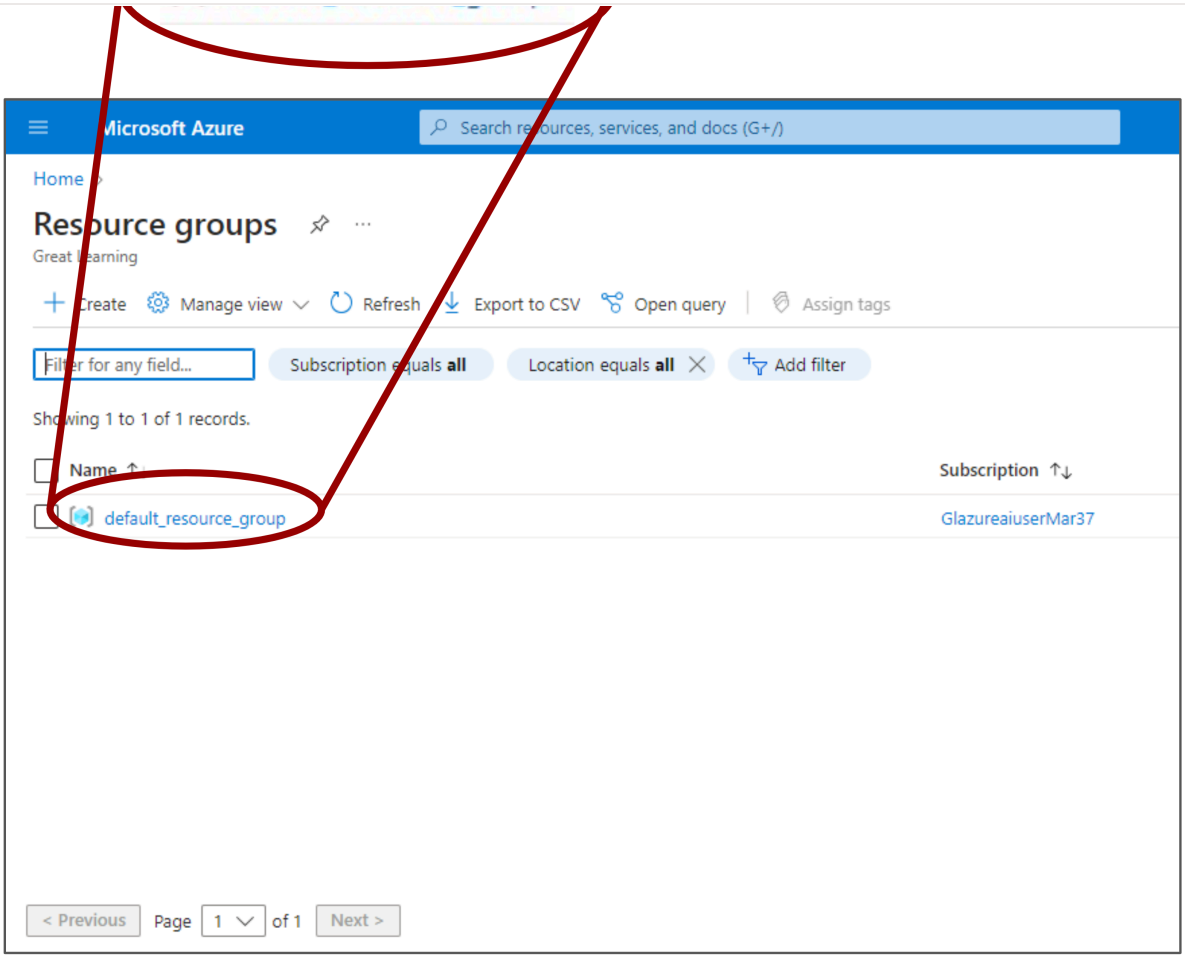


Figure 6

- Within the default resource group, locate and click on the **Create** button to initiate the process of creating a new resource.

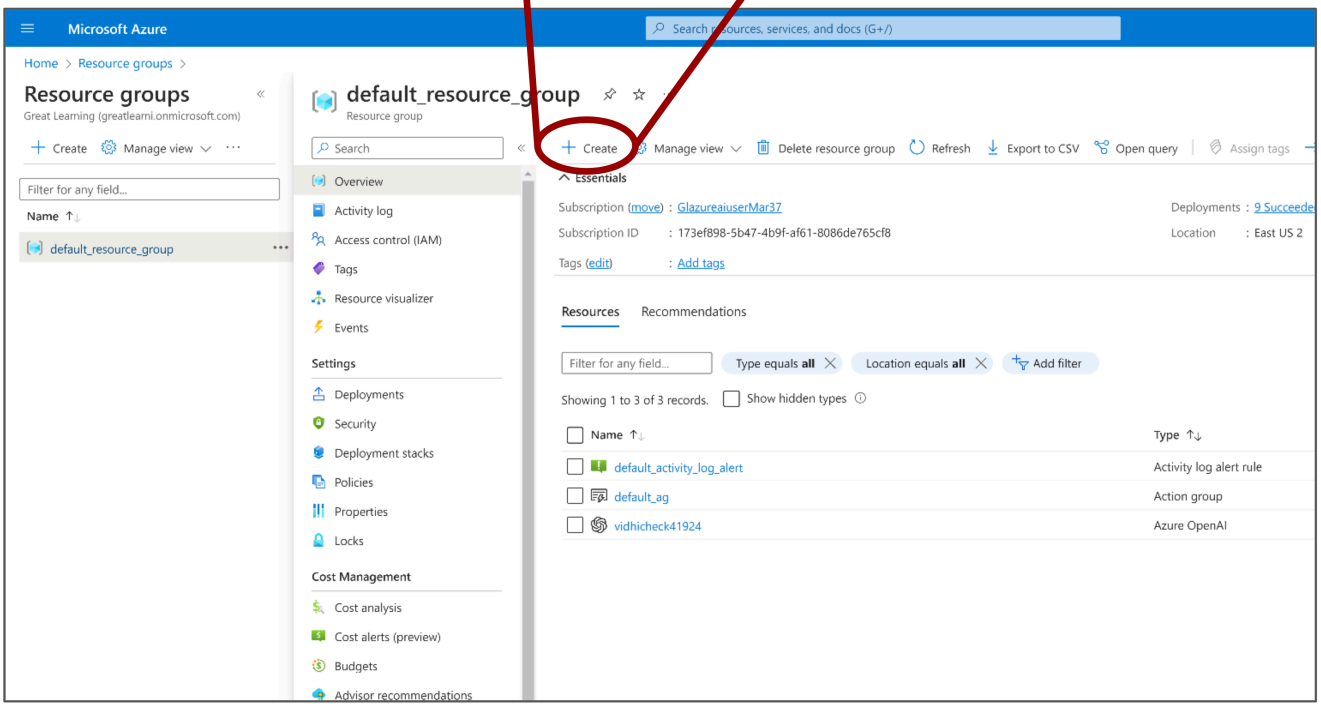


Figure 7

- On this page, you can explore all available resources that can be created. For our specific case, we'll search for **Azure OpenAI**.

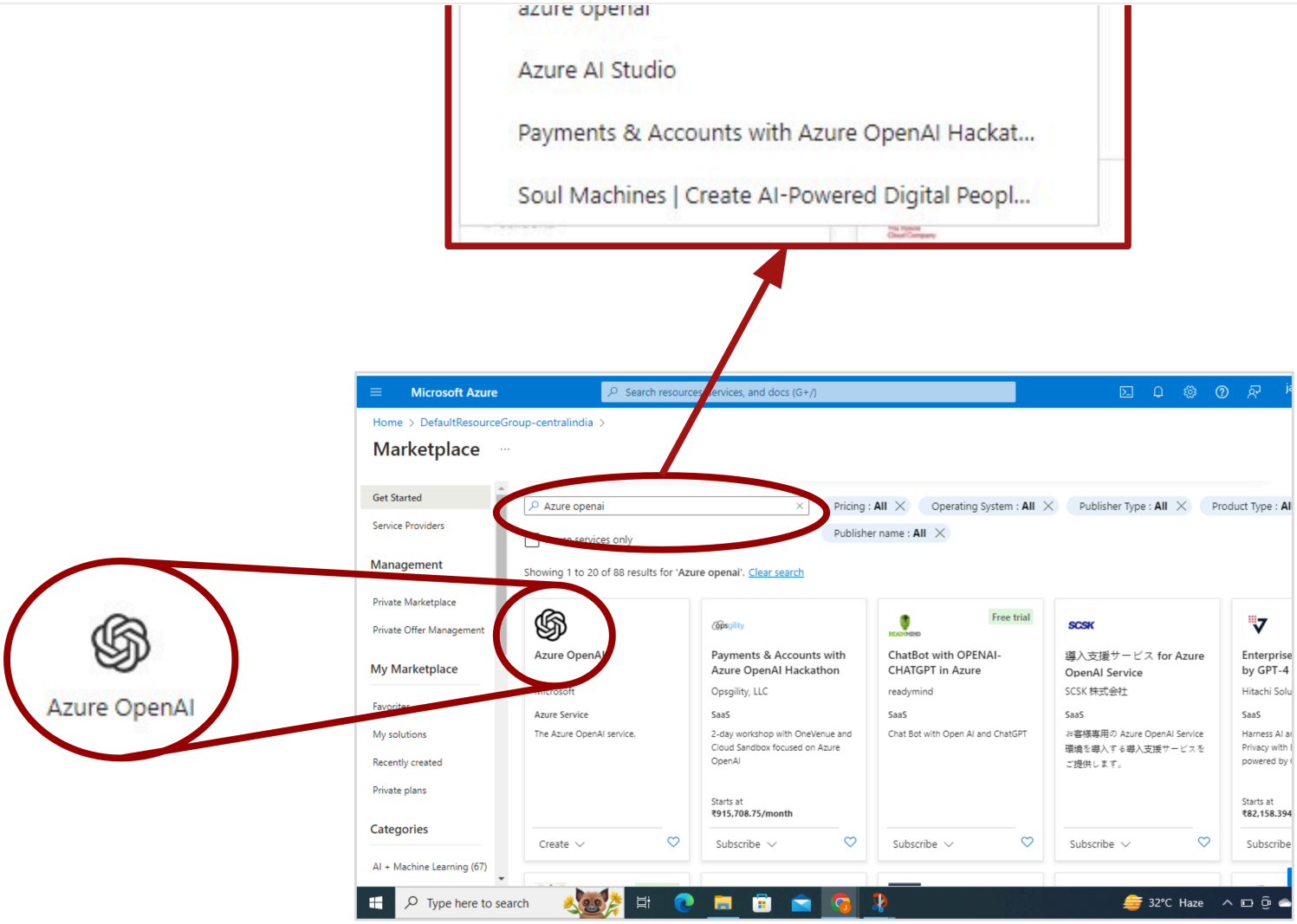


Figure 8

- Click on the **Create** button on the Azure OpenAI widget, then select the "**Azure OpenAI**" option to proceed with the creation process.

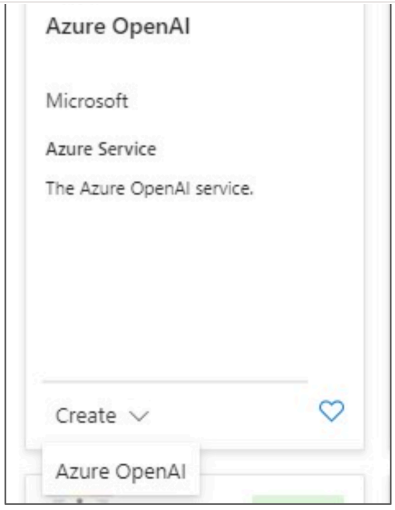


Figure 9

- Upon landing on the Create Azure OpenAI details page, you'll be prompted to provide the following mandatory details:
  - **Resource Group**: Choose the **default resource group** option available here.
  - **Resource Name**: Assign a unique name to the resource. This name will be decided by you. Fo the example in Figure 10, the resource name is "**test-openai-12345**".
  - **Pricing Tier**: Select the standard pricing tier ("**Standard S0**") here for pricing tier .
  - You can leave other details as default and proceed to complete the resource creation proces

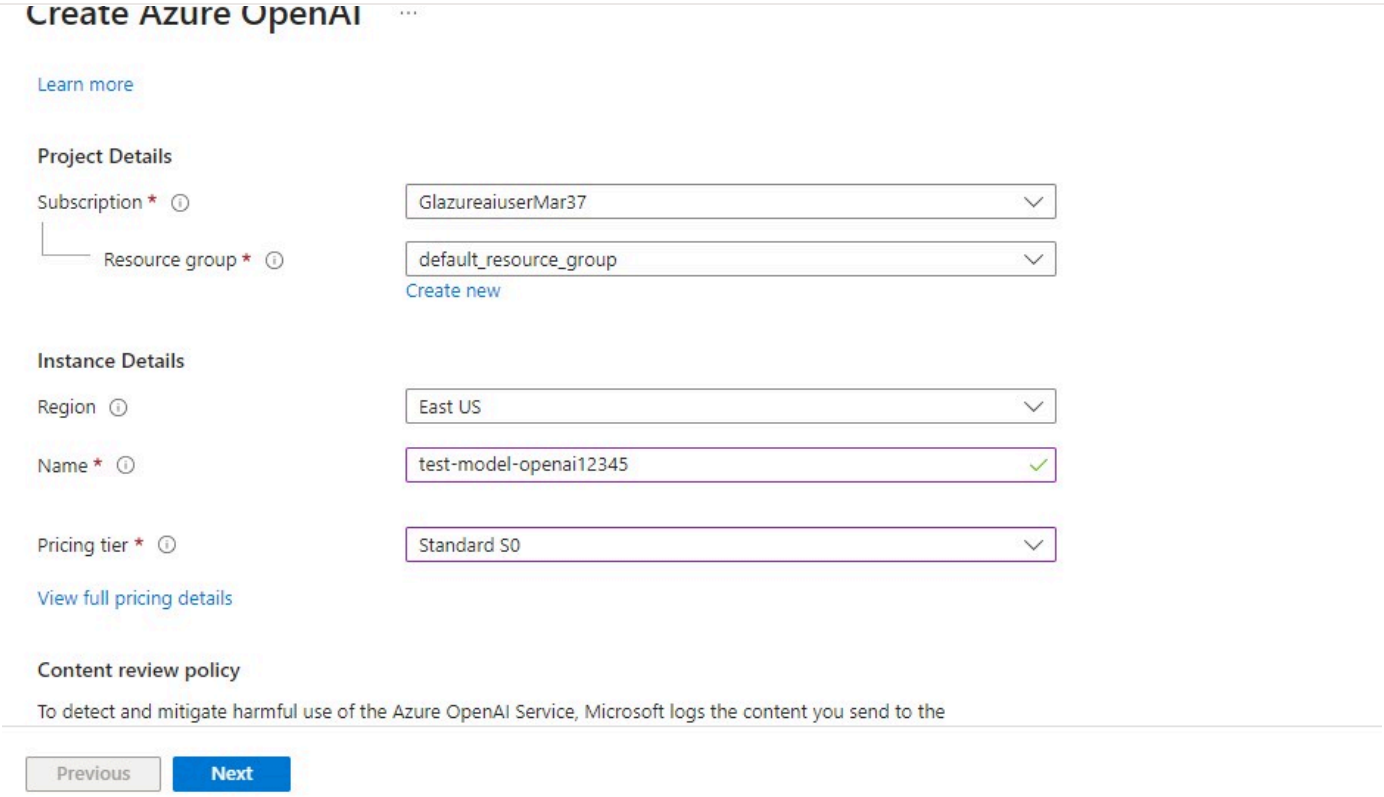
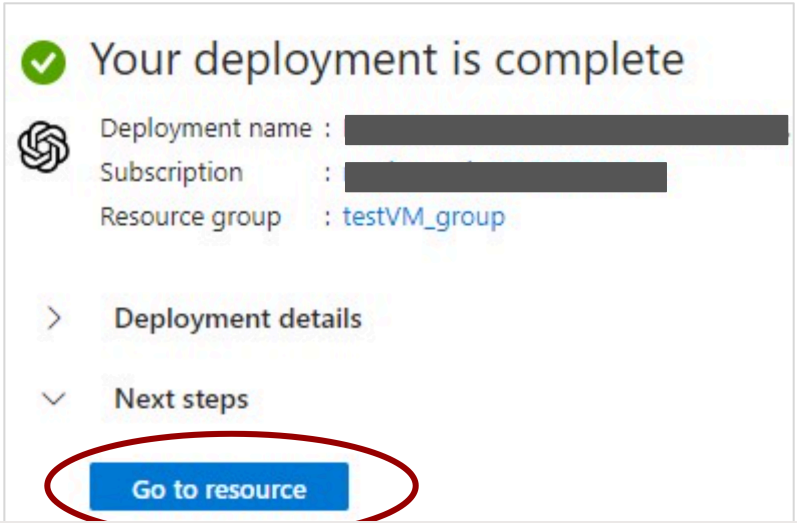


Figure 10

- Upon completing the resource creation process, you'll receive a notification stating "**Your deployment is complete**".
- **Note:** that this will deploy OpenAI resource for you, but you will still need to deploy a GPT model later in section 3 .
- You can then click on the "**Go to resource**" option to navigate to the homepage of the newly created resource.



- Having reached the resource page, you've successfully created the OpenAI resource. Now, proce by clicking on "**Keys and Endpoint**" to obtain the API Key and Endpoint for the resource. You will see multiple API Keys and can choose any one of them.

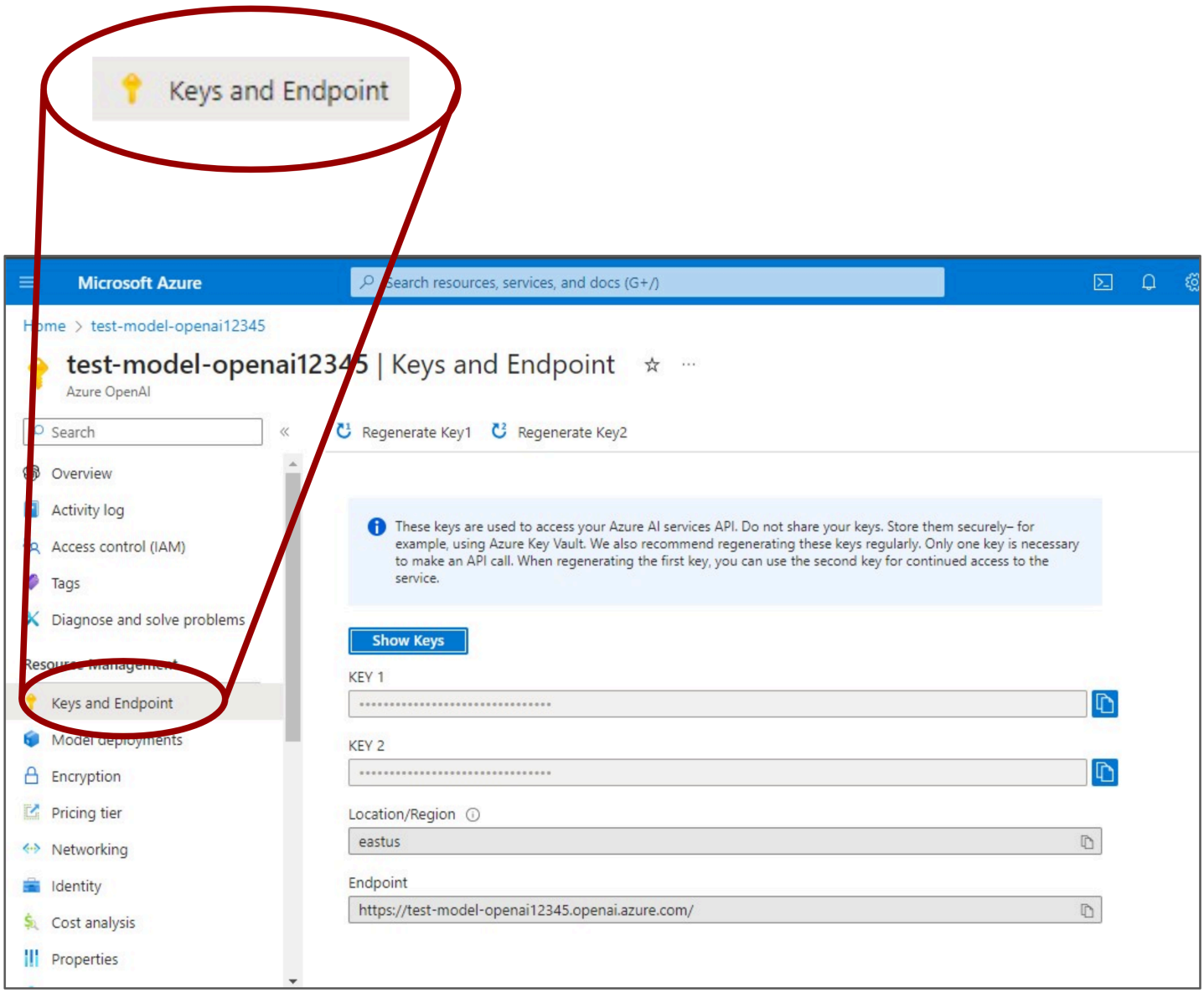


Figure 12

Above steps will provide us with two out of four necessary components. We can save the **API Key** and **Endpoint** separately to be used later in code.

- AZURE_OPENAI_KEY ✓
- AZURE_OPENAI_ENDPOINT ✓
- CHATGPT_MODEL
- AZURE_OPENAI_APIVERSION

Section Overview: *We now have API Key and Endpoint to be used for authentication late But we still require* ***API Version*** *and* ***Deployment Name (Chat GPT Model)****. We can get both of them from Azure studio.*

3.1 Get Deployment Name

Section Overview: *We will create a model deployment and get the name for it here. The* ***deployment name*** *helps the API know which specific resource you're trying to access, and makes sure your authentication credentials only work for that specific resource. This keeps your API requests safe and secure.*

- In the '**Overview**' section of the OpenAI Resource Page, click on "**Go to Azure OpenAI Studio**" option. This will take you to the AI Studio page.

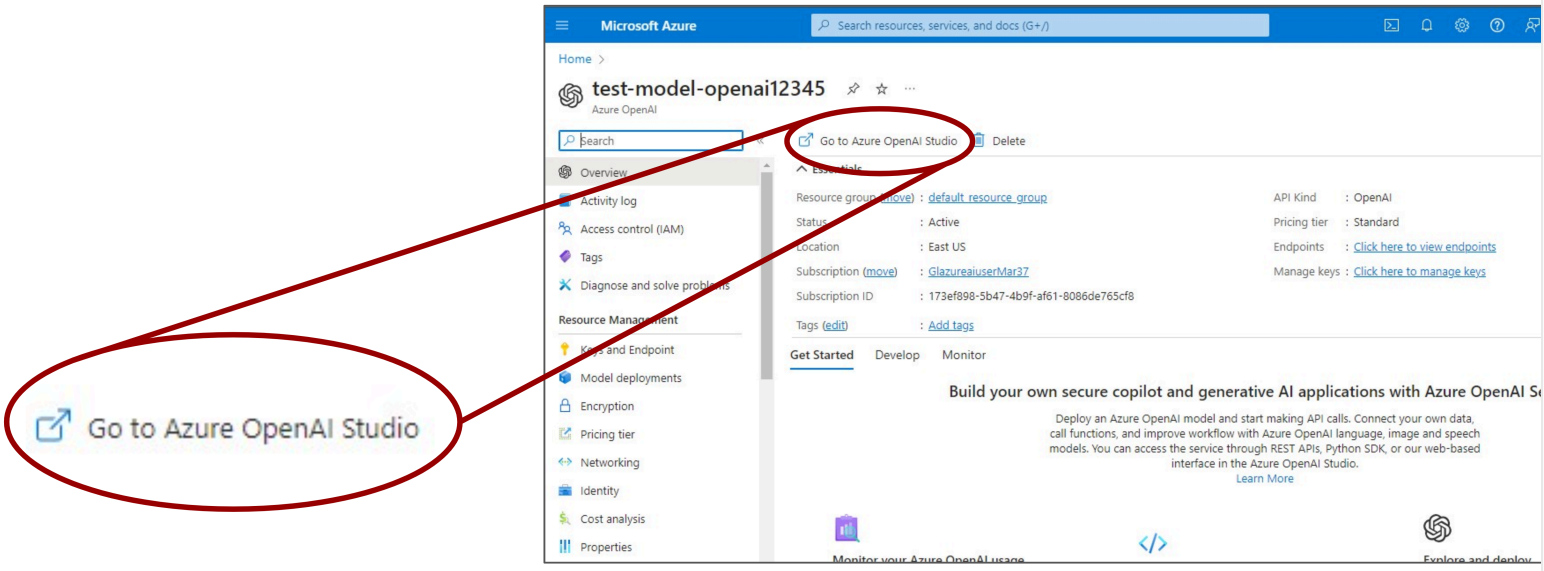


Figure 13

- In the same Azure Studio Inteface, you will see option **Deployments** in the left side menu.Click on
- In this step, we'll need to deploy at least one model. This deployed model will be utilized in the coc to generate text. We'll be deploying a GPT model. Click on "**Create new deployment**" to proceed with this deployment.

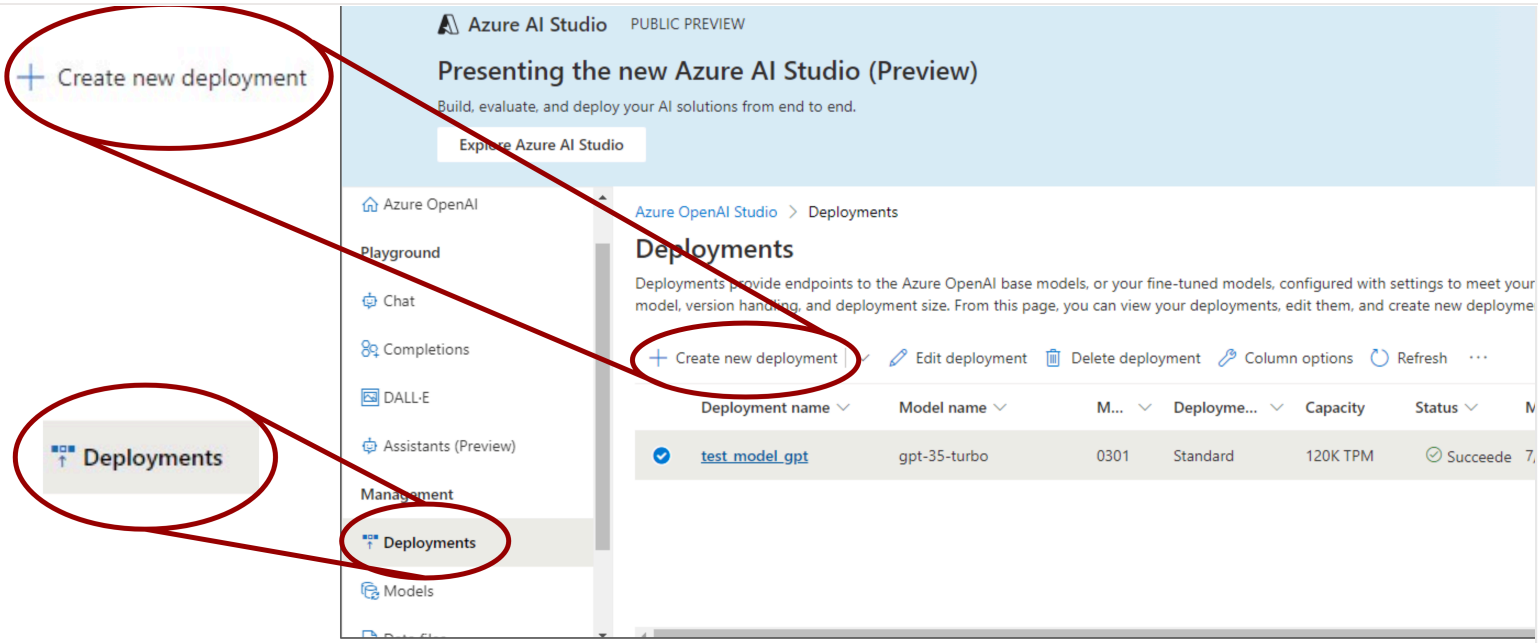
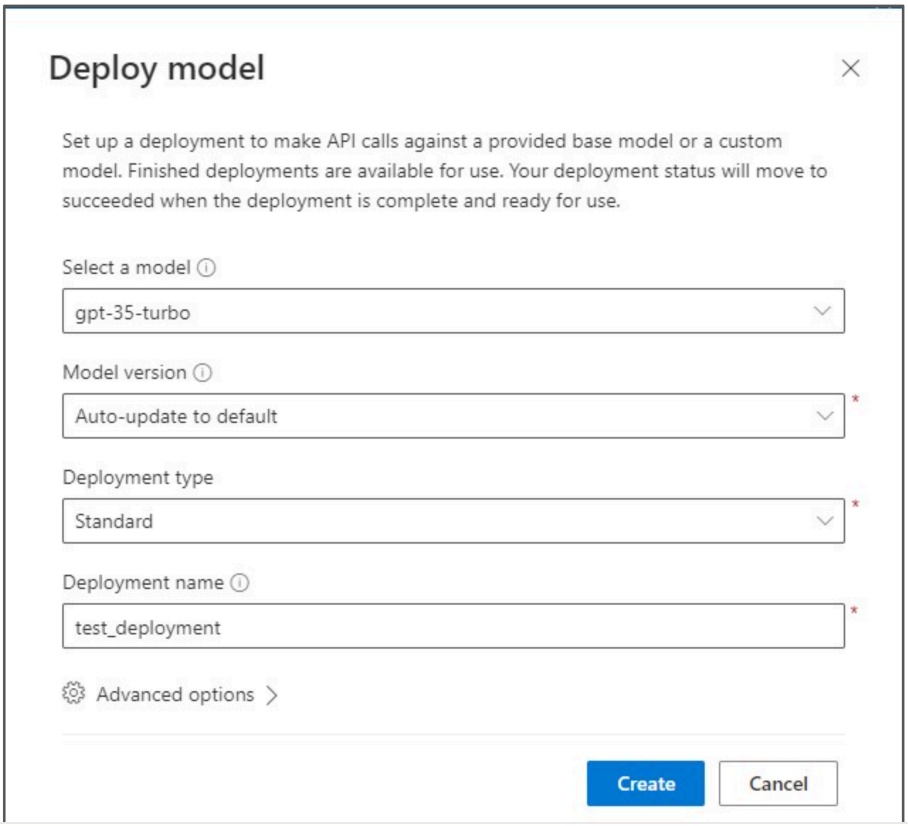


Figure 14

- On the details page for model deployment, select model: **gpt-turbo-35** and give deployment name Example: **test_deployment**. This deployment name will be later used as input in authentication.



- You can now observe that the new deployment named "**test_deployment**" has been successfully created.

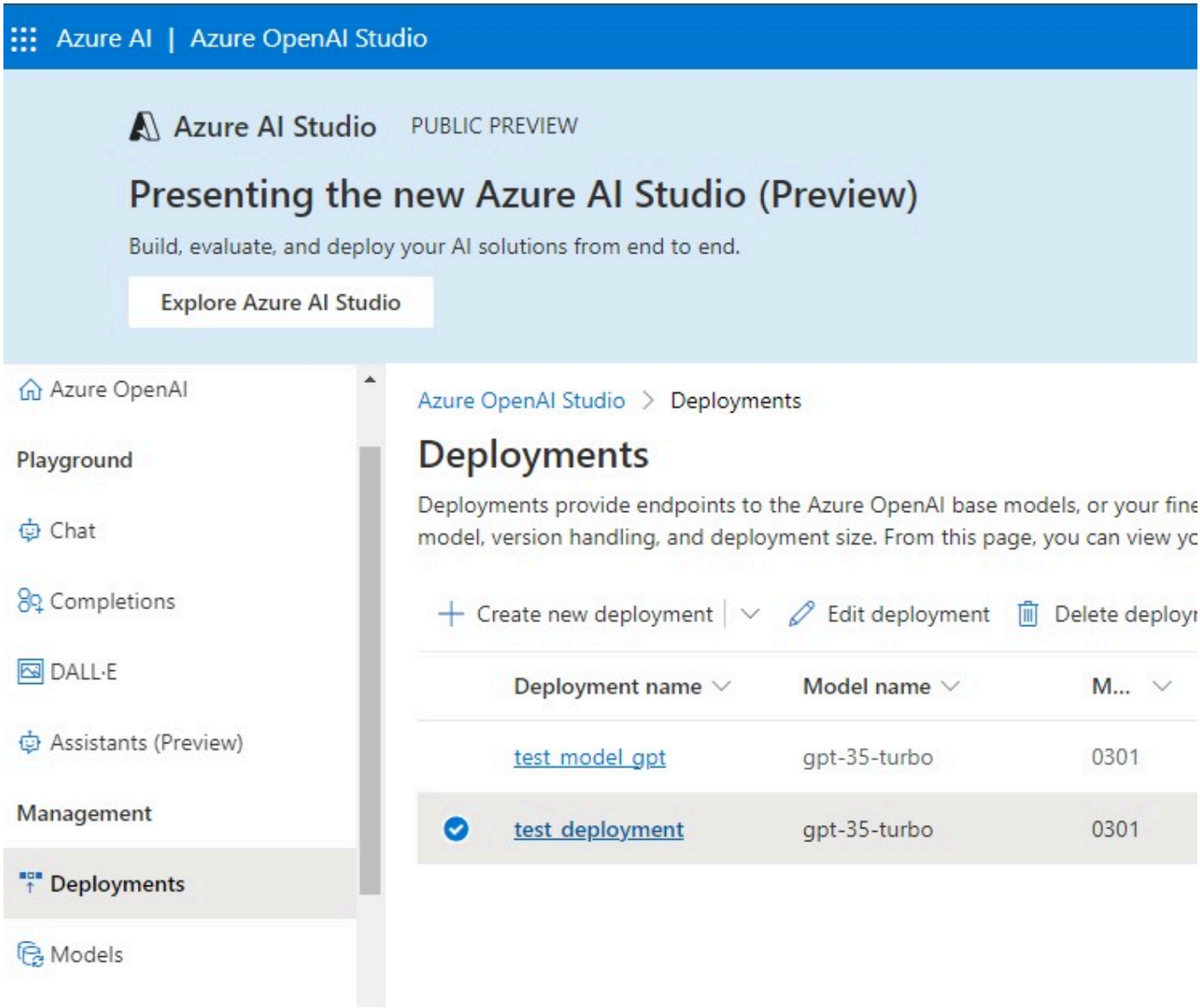


Figure 16

We have now the **deployment name (ChatGPT Model)** for the OpenAI resource. Let's get the last component, which is **API Version.**

- AZURE_OPENAI_KEY ✓
- AZURE_OPENAI_ENDPOINT ✓
- CHATGPT_MODEL ✓

*the model, we will get the **API version** for the model. API version is crucial for authentication, as it ensures that your requests are directed to the correct endpoint, and you have access to the features and functionality you need for your application.Follow th steps below:*

- In the same Azure AI studio page, navigate to the "**Chat**" option, where you'll find the chat playground. Scroll down and locate the "**View Code**" option, then click on it.

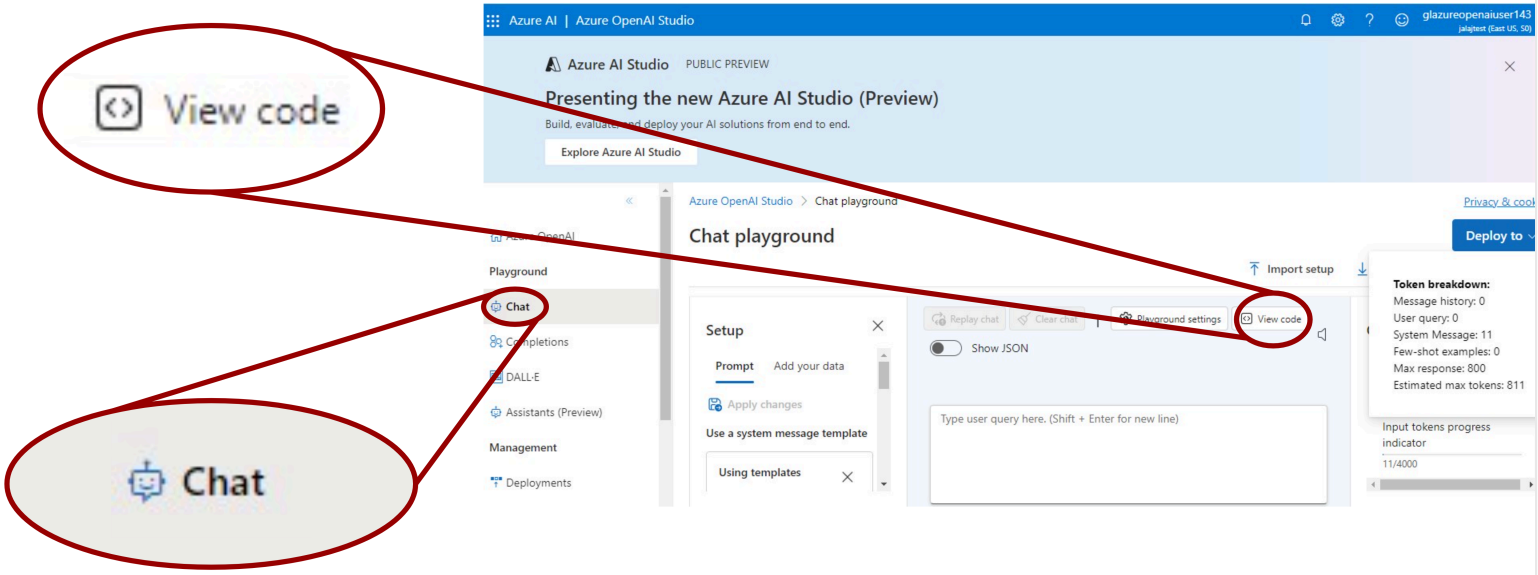


Figure 17

- You should be able to see the **API version** here. For the example shown below, API Version is "**2024-02-15-preview**".

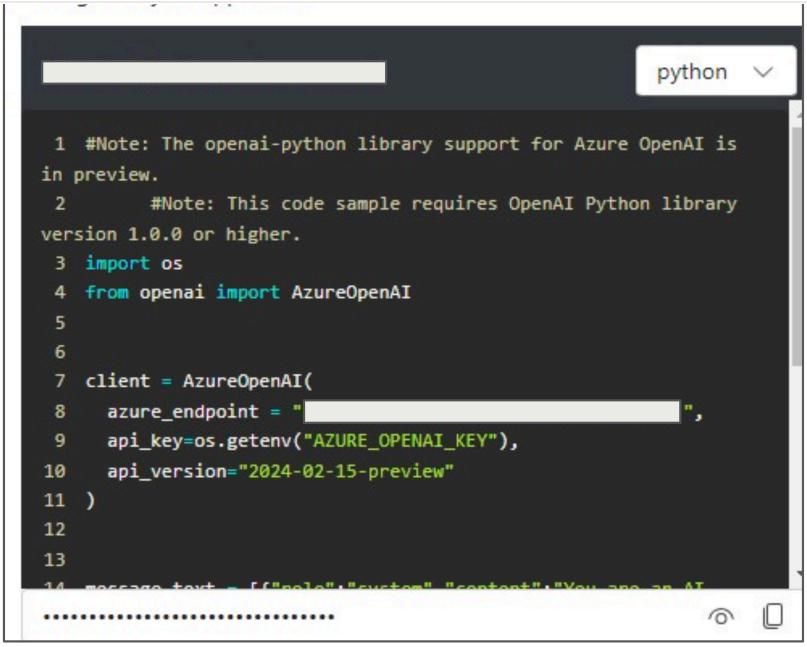


Figure 18

Now we have got all the necessary details to setup the azure openai:

- AZURE_OPENAI_KEY ✓
- AZURE_OPENAI_ENDPOINT ✓
- CHATGPT_MODEL ✓
- AZURE_OPENAI_APIVERSION ✓

Indeed, with the obtained details- **API Key**, **Endpoint**, **API Version**, and **Deployment Name** - we are now equipped to access the OpenAI service. Let us move to the code notebook for the authentication step.

4. Authentication

Section Overview: *Now that we have the necessary details for OpenAI, let's proceed to the **authentication** step in Google Colaboratory.*

4.1 Install and Import Necessary Libraries

Section Overview: *First, we'll need to import and install the necessary libraries required to work with OpenAI API in the code.*

Python----------

```
# Import all Python packages required to access the Azure Open AI API
from openai import AzureOpenAI
import json
import session_info
```

```
session_info.show()
```

OUTPUT:

```
▼ Click to view session information
-----
openai              1.2.0
session_info        1.0.0
-----
▶ Click to view modules imported as dependencies
-----
IPython             7.34.0
jupyter_client      6.1.12
jupyter_core        5.7.2
notebook            6.5.5
-----
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
Linux-6.1.58+-x86_64-with-glibc2.35
-----
Session information updated at 2024-04-19 07:59
```

Python----------

- **Install Required Libraries:** Python library installation is like adding a new tool to your coding toolbox.T
  - `!pip install openai session-info --quiet`: This line is installing the required Pytho packages (`openai`, `session-info`) using pip in the Google Colab environment.
  - `openai`: This package is required to interact with the Azure OpenAI API. It provides a Pytho client library to access the API.Here we will be using OpenAI version 1.2.0.
  - `session-info`: This package is used to display information about the current Python session, including installed packages and their versions.
  - `--quiet`: This flag tells pip to install the packages quietly, without showing us all the installation details. It's like telling pip to "be quiet" while it works.

- **Importing Packages:** Once the library is installed, you import it into your Python script to use its features. It's similar to taking a tool out of your toolbox and putting it to work.
  - `from openai import AzureOpenAI`: This line imports the `AzureOpenAI` from the

`import json` This line imports the `json` package, which is commonly used for parsing a manipulating JSON data. In the context of working with APIs, JSON is a prevalent data forma for exchanging information. Importing the `json` package enables us to handle JSON responses from the API effectively.

- `session_info`: This line imports the `session_info` package, which provides functionality t display the current session information, including the installed packages and their versions.

- **Show Session Information:** Now, let's take a look at the session details, like the version and the libraries we're using.
  - `session_info.show()`:This line calls the `show()` function from the `session_info` package to display the current session information.
  - By showing the installed packages and their versions, this step helps verify that the necessa dependencies are correctly set up and provides transparency regarding the environment configuration for the prompt engineering project using Azure Open AI API.

4.2 Write Credentials File

Section Overview: *Now that we've imported the required libraries, let's proceed by creating the configuration file to store all the OpenAI credentials we've obtained so far. This file will be in JSON format.*

Creating the JSON file here enhances security measures. By storing our credentials in th file, we can securely share the code notebook with others, ensuring that our credentials remain confidential. This setup allows the code notebook to access the credentials from the JSON file without exposing them directly in the notebook itself.

Python----------

CODE:

```
config_data = {
    "AZURE_OPENAI_KEY": "your_openai_key",
    "AZURE_OPENAI_ENDPOINT": "your_openai_endpoint",
    "AZURE_OPENAI_APIVERSION": "your_openai_apiversion",
    "CHATGPT_MODEL": "your_chat_model_name"
```

```
json.dump(config_data, json_file, indent=4)
```

Python----------

Here's a breakdown of the code:

- **Define Configuration Data:**
  - The first part of the code defines a Python dictionary called **config_data**. In this case, the `config_data` dictionary has four key-value pairs:
  - **AZURE_OPENAI_KEY**: This key maps to the value "your_openai_key", which should be replaced with your actual Azure OpenAI API key.
  - **AZURE_OPENAI_ENDPOINT**: This key maps to the value "your_openai_endpoint", which should be replaced with your actual Azure OpenAI API endpoint URL.
  - **AZURE_OPENAI_APIVERSION**: This key maps to the value "your_openai_apiversion", which should be replaced with your actual Azure OpenAI API version.
  - **CHATGPT_MODEL**: This key maps to the value "your_chat_model_name", which should be replaced with your actual ChatGPT model name (**test_deployment**).

- **File Path Definition:**
  - `file_path`: This variable stores the path where the configuration data will be saved.
  - It's set to `config.json`, indicating that the configuration data will be stored in a JSON file named **config.json**.

- **Writing Configuration Data to JSON File:**
  - `with open(file_path, 'w') as json_file`: This line opens the `config.json` file in write mode, creating it if it doesn't exist.
  - `with`: The `with` statement is used to open a file and perform some operations on it.he `with` statement also ensures that the file is properly closed after we're done with it, regardless of whether an exception is thrown or not.
  - `open(...)`: This part opens a file located at the path specified by `file_path`.
  - `file_path`: This is the path to the file that we want to open. In this case, it's the string valu 'config.json' that we assigned to the `file_path` variable earlier.
  - `'w'`: The 'w' mode stands for "write" mode, which means that we want to write data to the file. If the file doesn't exist, it will be created. If it does exist, its contents will be truncated (i.e deleted).
  - `as json_file`:This part assigns the opened file to a variable named `json_file`. It's like giving a name to the notebook we just opened, so we can refer to it easily.
  - `json.dump(config_data, json_file, indent=4)`: This line of code takes the

any other Python object that can be converted to JSON. Here, it is a Python dictionary we defined that contains the credentials.
- `json_file`: This is the file we opened earlier, where we want to write the JSON data.
- `indent=4`: This is an optional parameter that specifies how many spaces to use for indentation in the JSON output. In this case, we're using 4 spaces for readability.

- **Usage of `config.json` File:**
  - **config.json** is used to store sensitive information like API keys, endpoints, and model identifiers.
  - It's a common practice to store such information in a separate file, rather than hardcoding it directly into your code.
  - This approach enhances security by keeping sensitive data out of version-controlled files an making it easier to update credentials without modifying the code.
  - This approach enhances security by keeping sensitive data out of version-controlled files an making it easier to update credentials without modifying the code.

By executing this code snippet, a **config.json** file will be created in the specified file path, containing th OpenAI credentials and configuration data in a structured JSON format.

4.3 Access Config.json File in via Colab interface

Section Overview: *Once we've created the **config.json** file containing our OpenAI credentials, we can locate this and now download this file to avoid the need for creating every time we start a session in Colab. Let's see how we can save it:*

- **Navigate to the Files Section:**
  - On the left-hand side of the Google Colab interface, click on the folder icon. This opens the Files section where you can see the contents of your Colab environment.

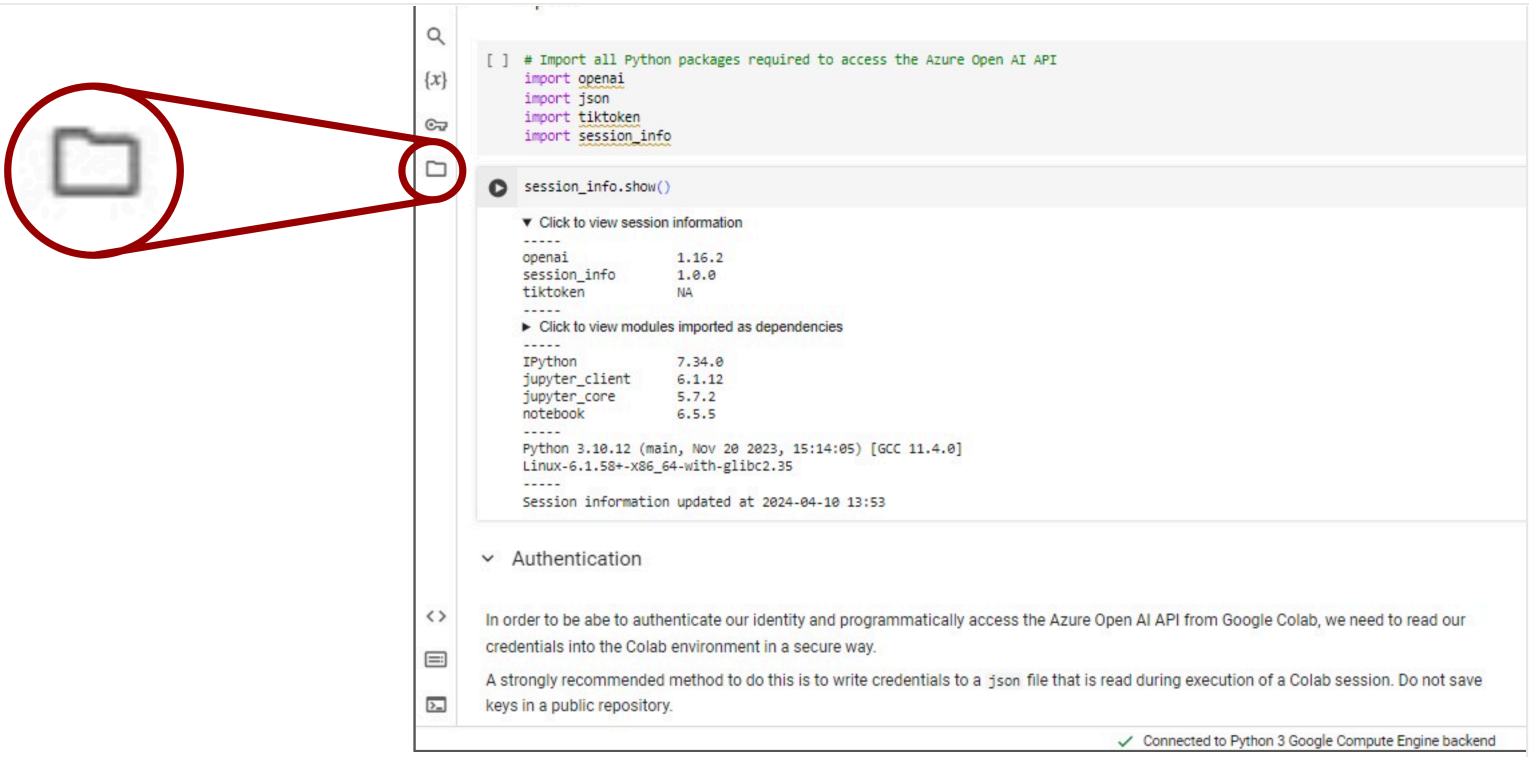
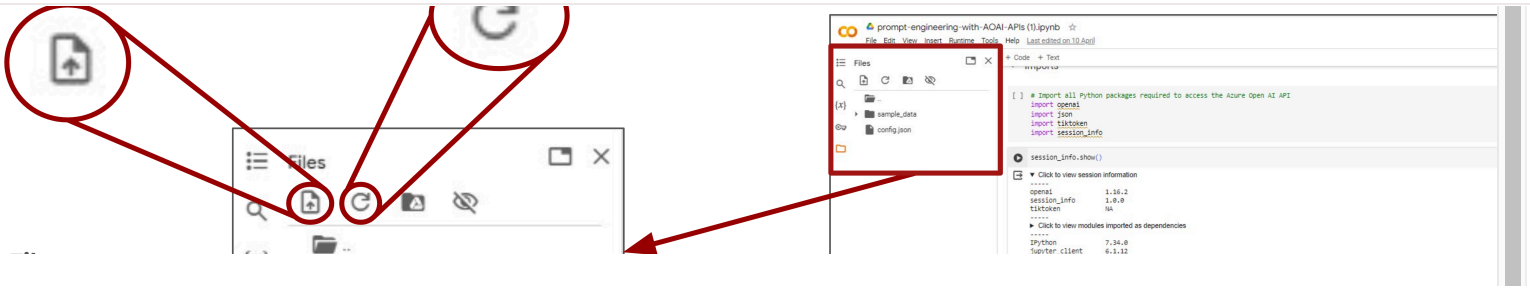


Figure 19

- **Locate the config.json File:**
  - Once in the Files section, locate the config.json file. It should be listed among other files and directories.

- **Refresh the Files List (if necessary):**
  - If you don't see the config.json file immediately, you may need to refresh the file list.
  - To do this, click on the circular arrow icon located at the top-right corner of the Files section. This refreshes the list and should display any newly created files, including config.json.


Files
prompt-engineering-with-AOAI-APIs (1).ipynb
Files
sample_data
config.json
# Import all Python packages required to access the Azure Open AI API
import openai
import json
import tiktoken
import session_info
session_info.show()
Click to view session information
openai 1.16.2
session_info 1.0.0
tiktoken 0.6
Click to view modules imported as dependencies
IPython 7.34.0
jupyter client 6.1.12