Deploy your Model to a WebApp on Azure

Agenda

- Analyze the deployment process of an LLM model using a banking dataset to understand customer intents.
- Step-by-step guide on deploying the LLM model as a web app on Azure Chat Playground.
- Review and test the responses generated by the deployed web app.

Steps Performed

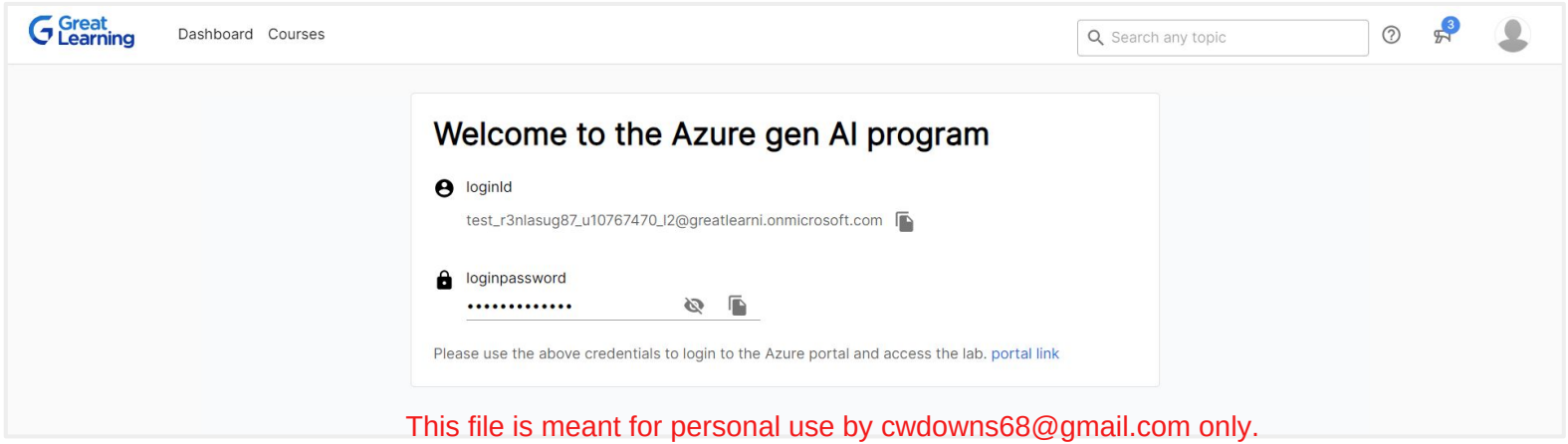
- Provision the Azure OpenAI resource
- Deploy a model
- Use the chat playground
- Create Azure AI search service
- Add your data
- Case Study
- Architecture of Web App Deployment
- Deploy your model to a Web App
- Test the Web App
- Summary & Clean Up

Provision an Azure OpenAI Resource

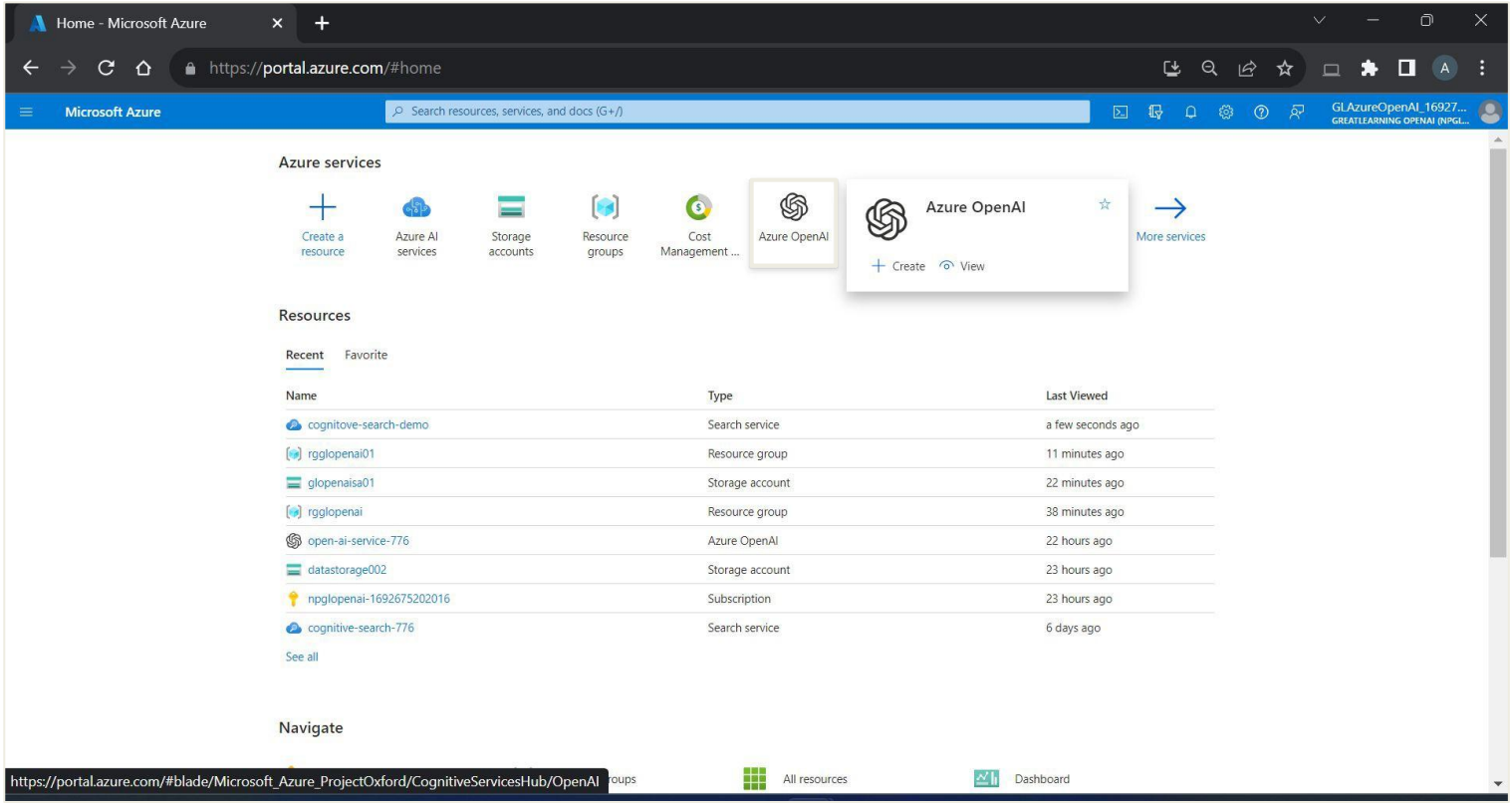
Ground Rules : Important

Ground rules for setting up an OpenAI Resource:

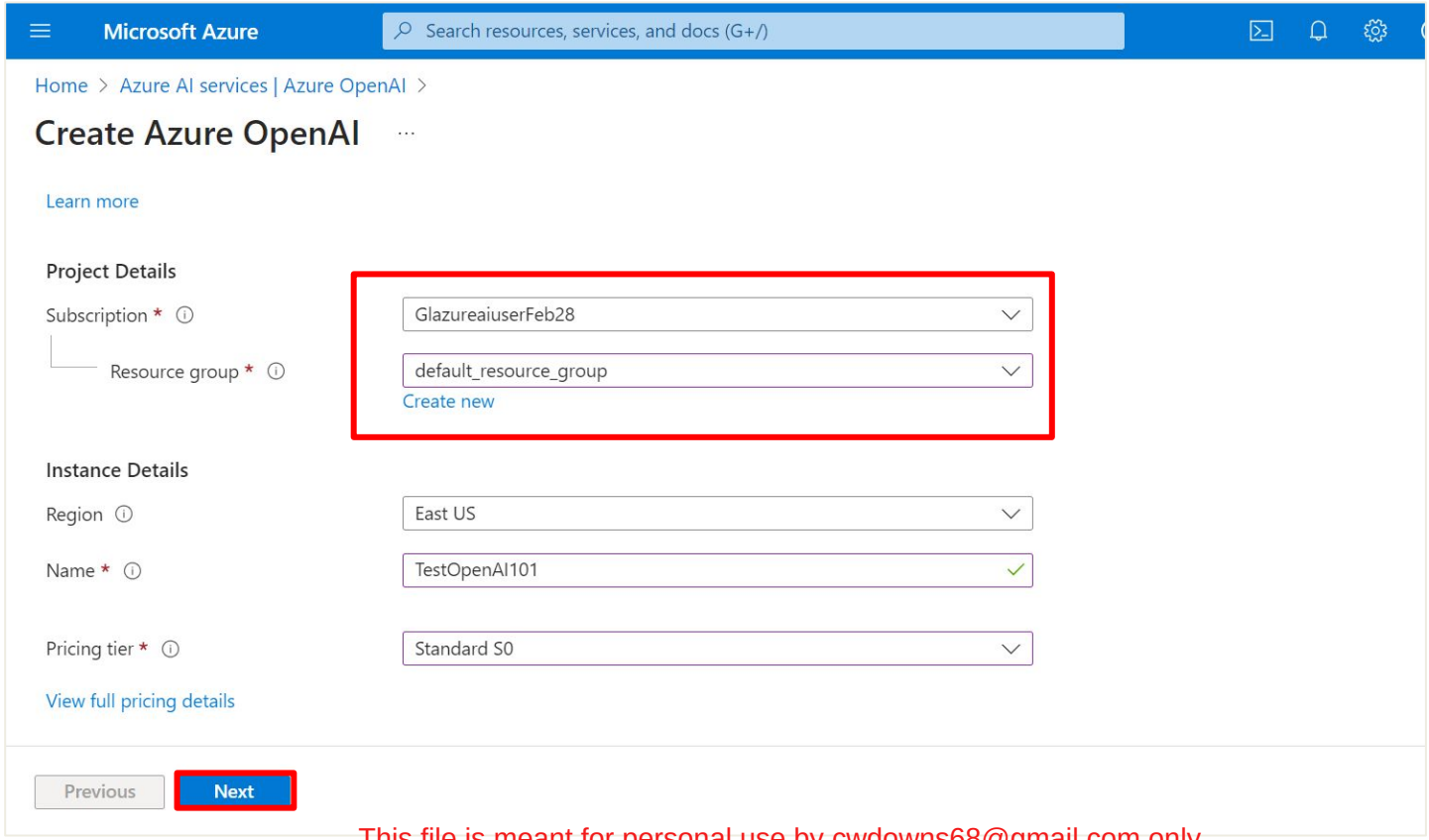
- Utilize the **default resource group** for all your tasks.
- Select **Region** = **East US** for the Open AI resource
- **Pricing tier**: **Standard S0 pricing tier**
- **Allow 10 to 15 minutes** for the resource to become fully operational following its creation.
- If the microsoft credentials are required at any point, employ the credentials found within the **"Welcome to the Azure Gen AI" section on Olympus** to complete the Microsoft Login process.
- It is advised to create and use a **single resource** for all future tasks within the program.



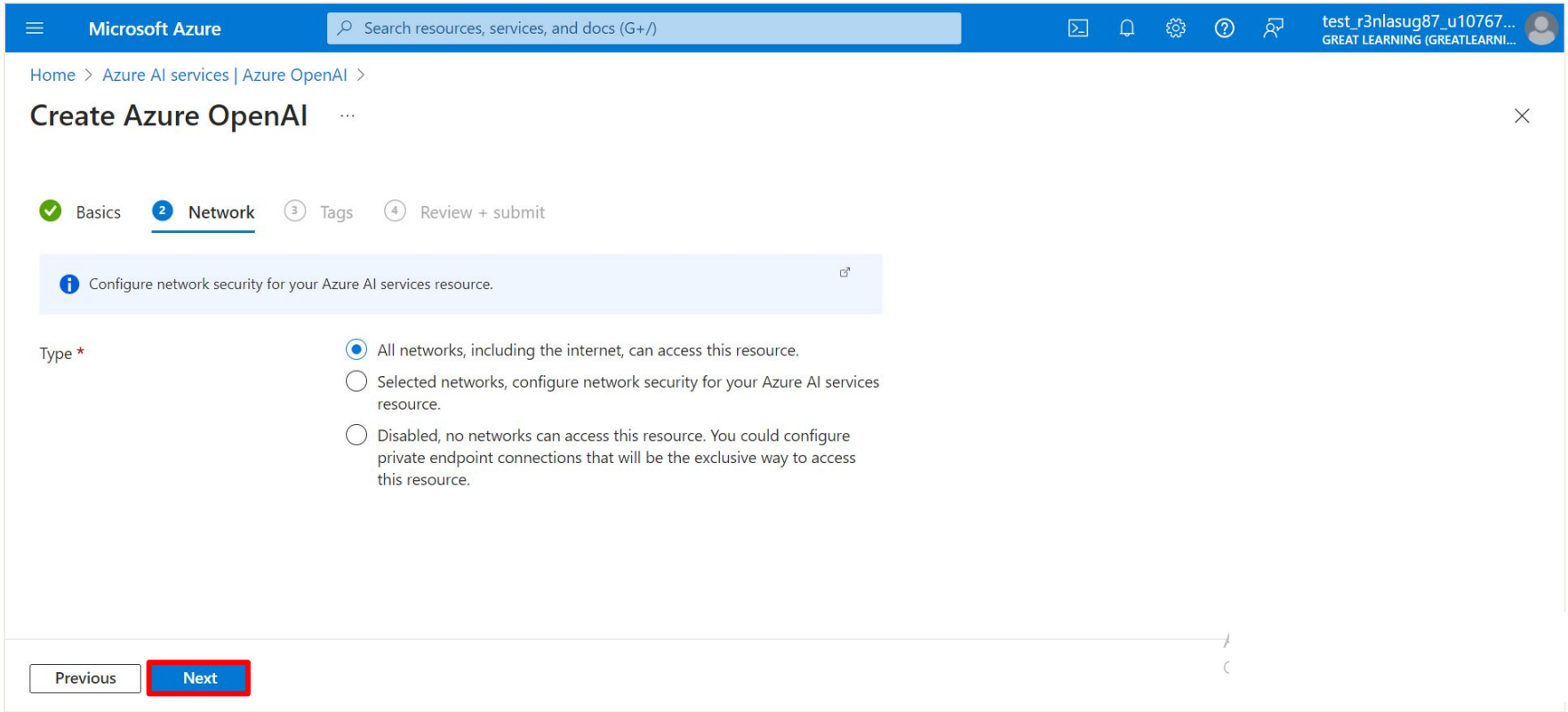
Click on Azure OpenAI service



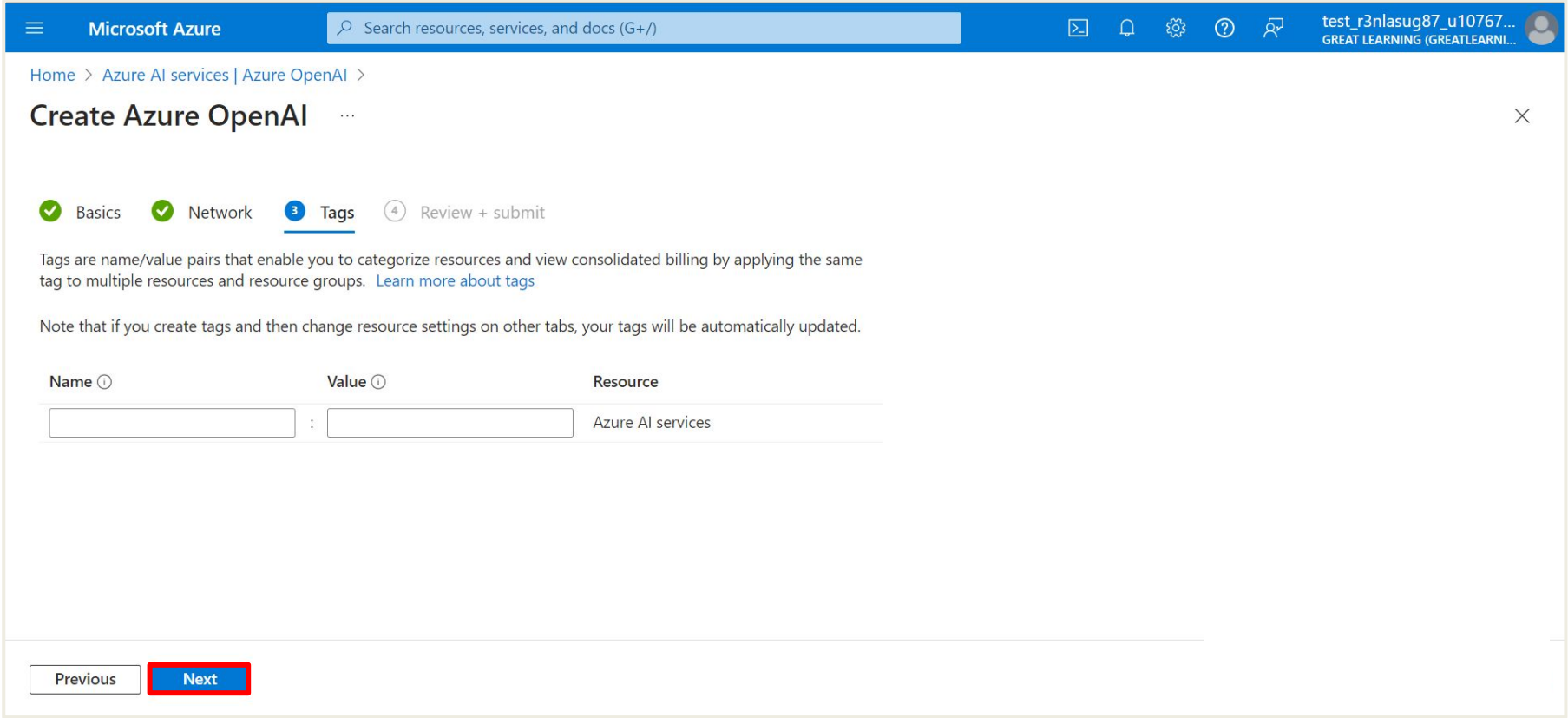
Select the details as below



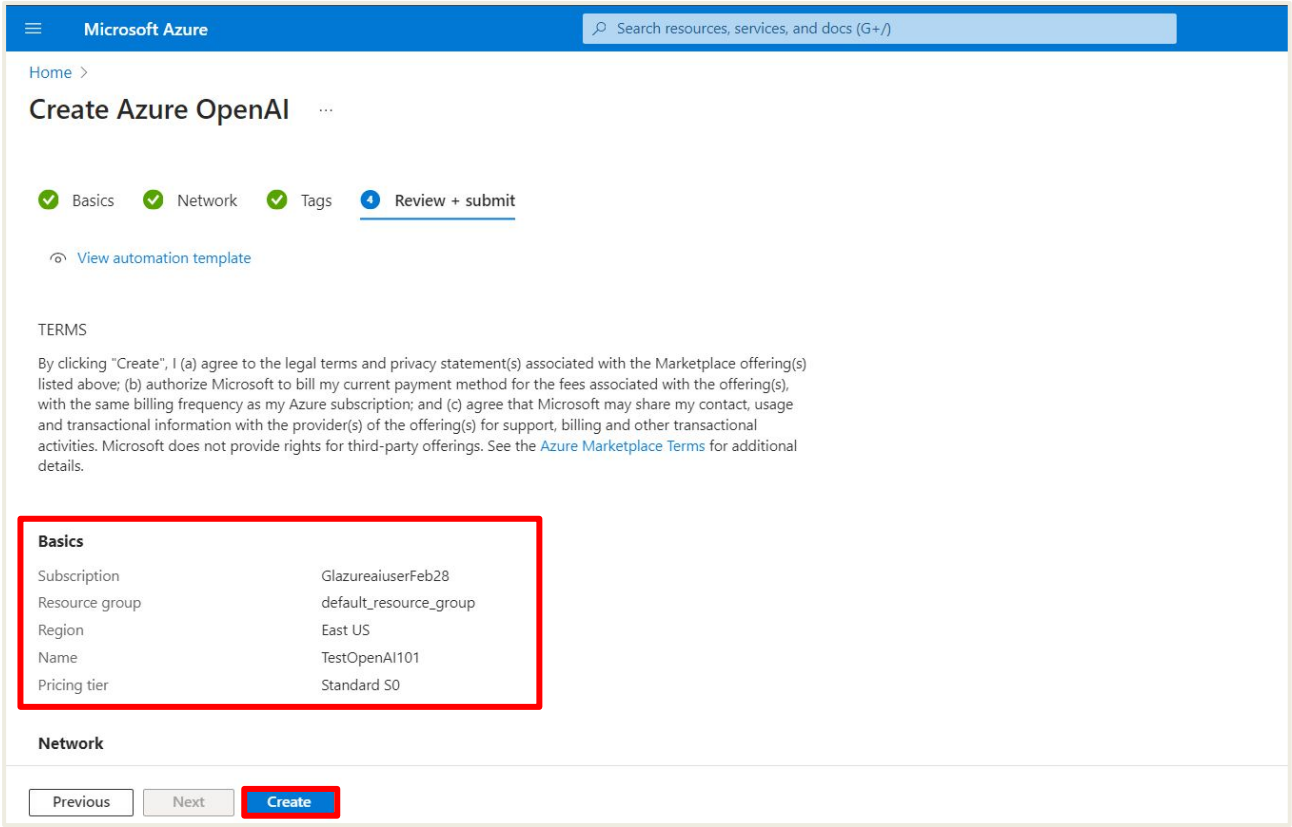
Next



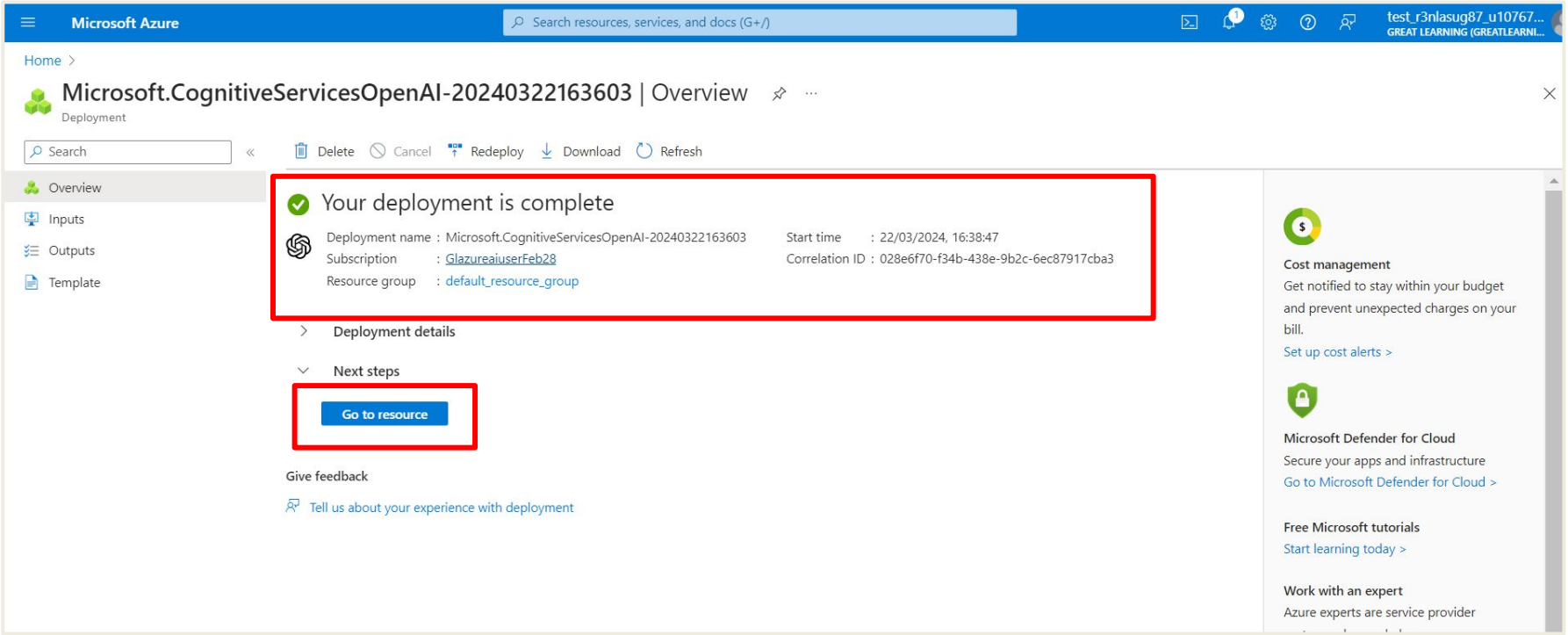
Next



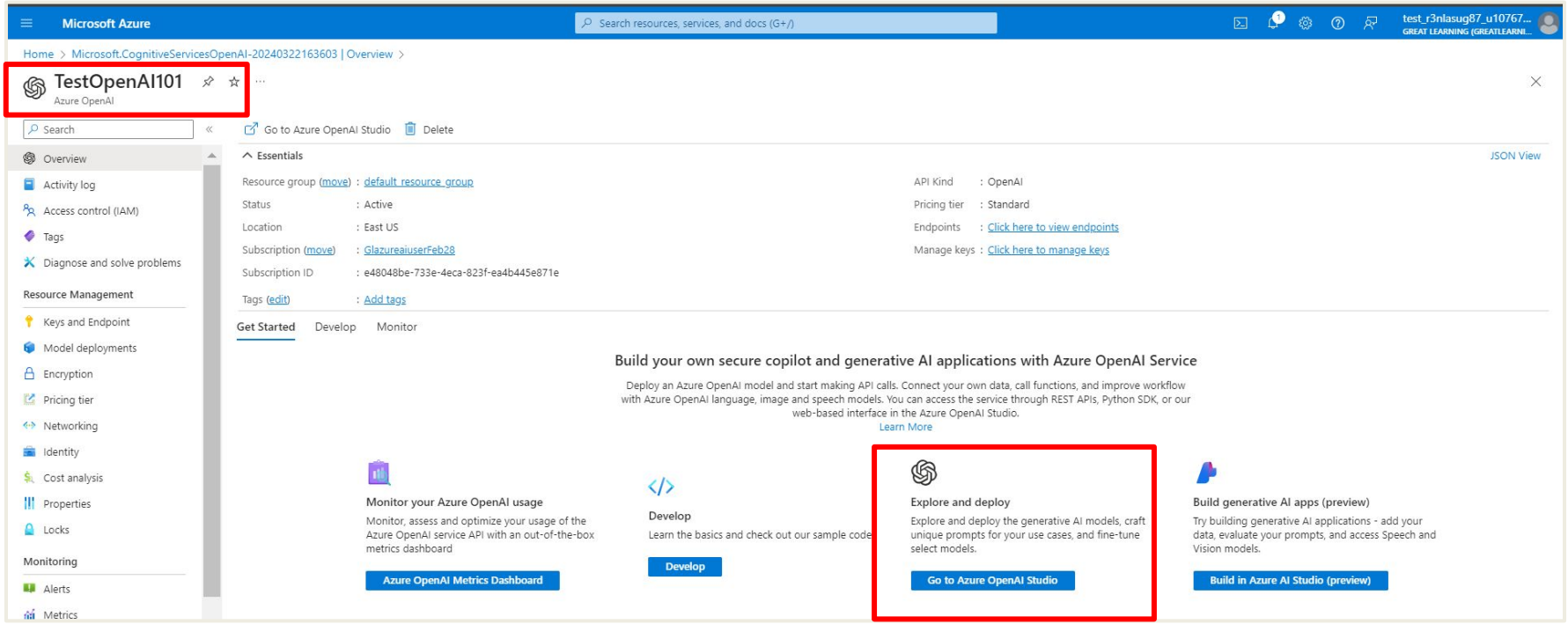
Click on Create



Deployment will take few mins



Click on Explore once the deployment is complete



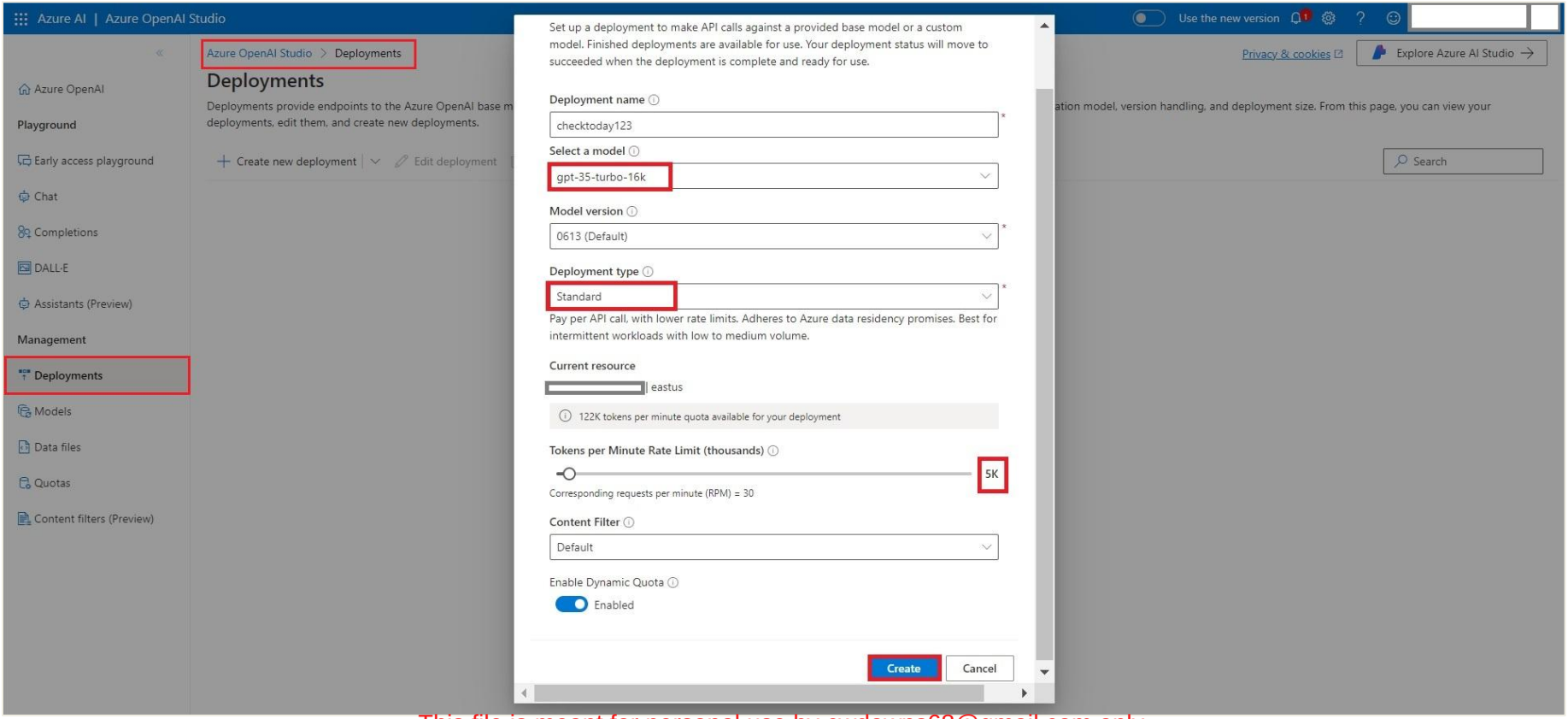
Deploy a model to use Chat Playground

Ground Rules : Important

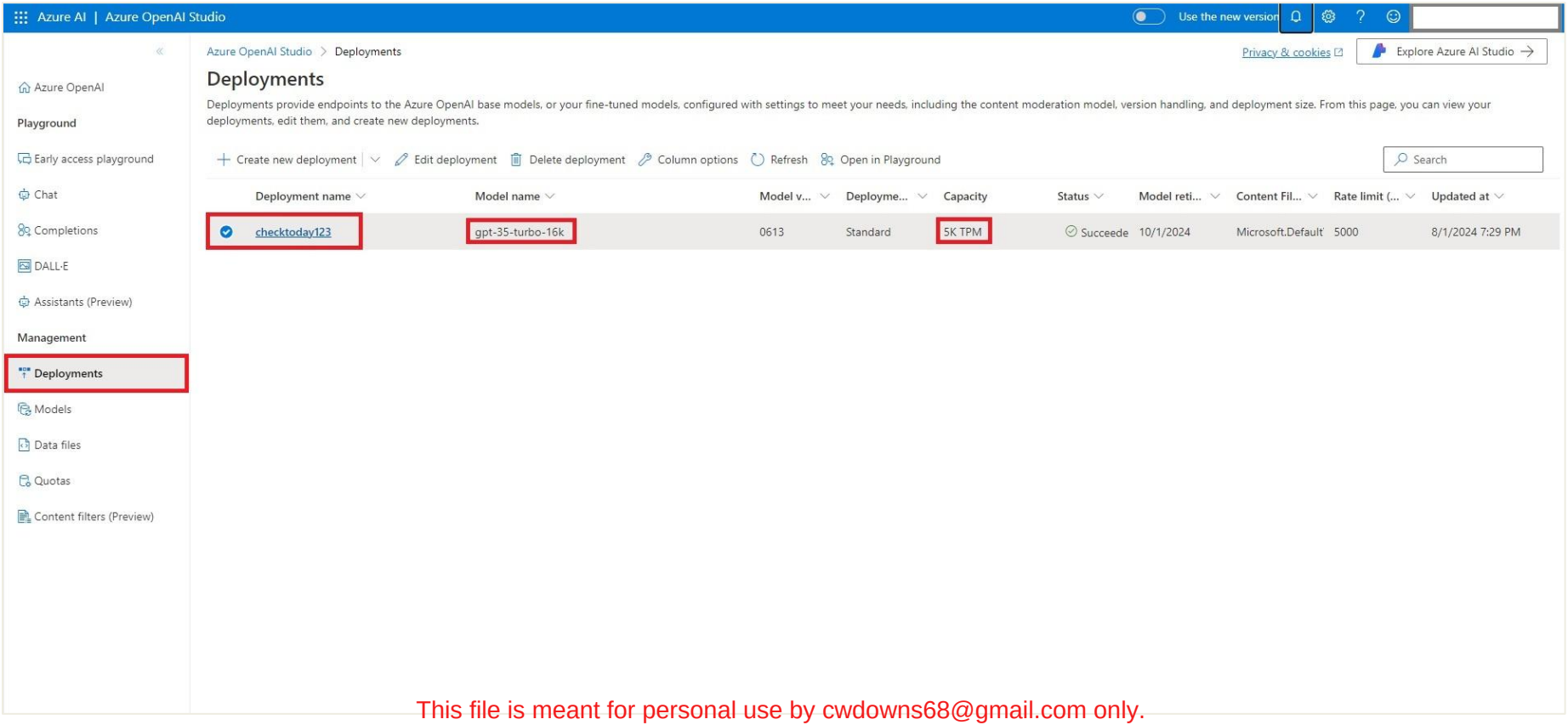
Ground rules for deploying a model in Azure OpenAI Resource:

- **Deployment name:** A unique name of your choice
- Select the **Model: gpt-35-turbo-16k** only
- Model version: **Auto-update to default**
- Deployment type: **Standard**
- **Tokens per minute rate limit: 5K**
- Content filter: Default
- Enable dynamic quota: Enabled
- **Allow 10 to 15 minutes** for the deployment to become fully operational following its creation.
- Please ensure that you have **only 1 ACTIVE DEPLOYMENT**. This is because the total token limit summed up across all the deployments is only **240k** and if you exceed this, you will not be able deploy any model.

Deploy a model



You will be able to view this



Use the Chat Playground

Use the Chat Playground

1. In the Playground section, select the Chat page. The Chat playground page consists of three main panels (which may be arranged right-to-left horizontally, or top-to-bottom vertically depending on your screen resolution):
    - **Setup -** used to set the context for the model's responses.
    - **Chat session** - used to submit chat messages and view responses.
    - **Configuration** - used to configure settings for the model deployment.

2. In the Configuration panel, ensure that your **gpt-35-turbo-16k model deployment** is selected.

3. In the Setup panel, review the default System message, which should be ***"You are an AI assistant that helps people find information".*** The system message is included in prompts submitted to the model, and provides context for the model's responses; setting expectations about how an AI agent based on the model should interact with the user.

Use the system message

In the previous weeks, you've engaged in a chat conversation with your model based on the default system message. You can customize the system setup to have more control over the kinds of responses generated by your model with respect to the case study covered that will be covering here.

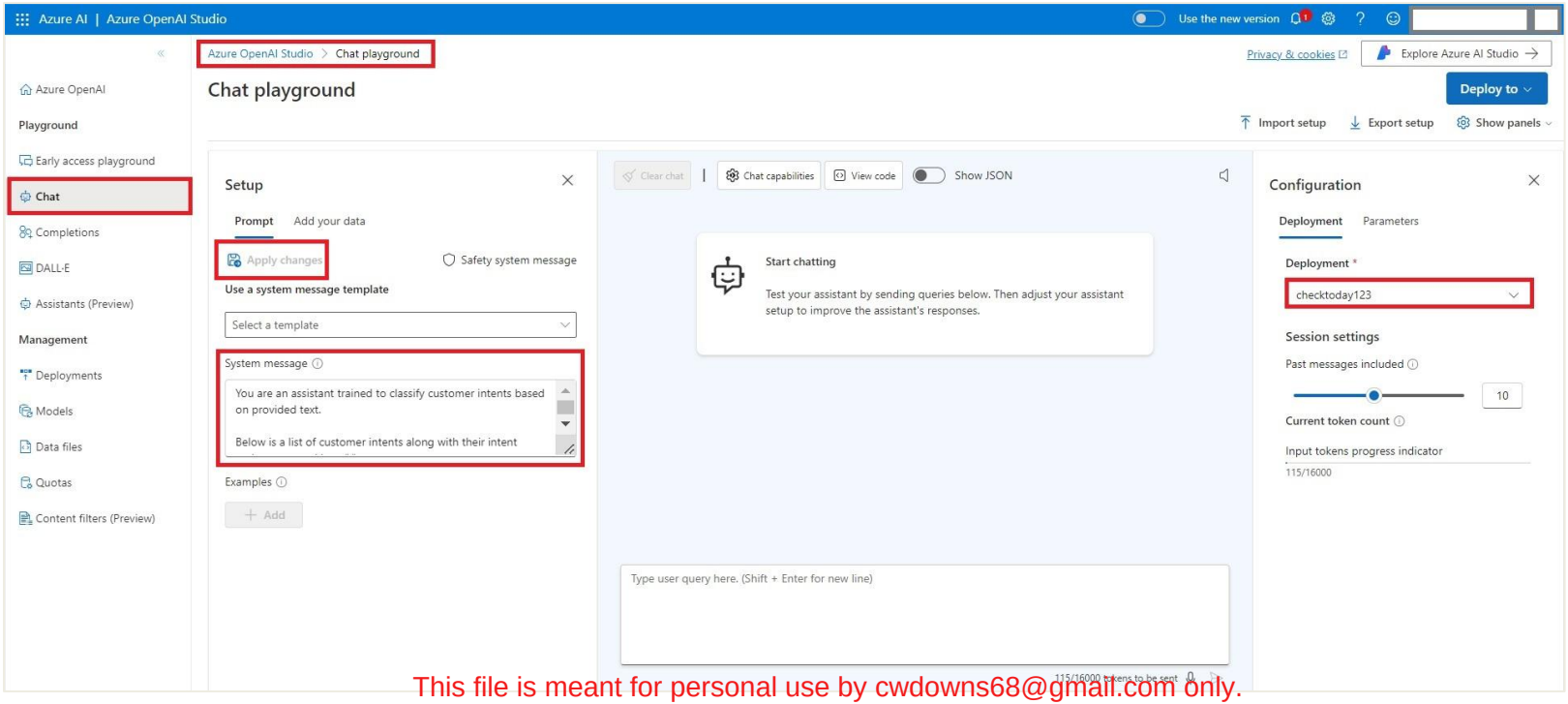
You are an assistant trained to classify customer intents based on provided text.

Below is a list of customer intents along with their intent codes separated by a ":":
- top_up_by_bank_transfer_charge:56
- top_up_by_card_charge:57
- top_up_by_cash_or_cheque:58
- top_up_failed:59
- top_up_limits:60
- top_up_reverted:61
- topping_up_by_card:62

Given the following text, classify it and output only one intent code:
Customer: ```{text}```
Intent Code:

Use the Chat Playground

When the model deployment is completed, navigate to the “Chat Playground”.
Enter the system message provided and then click on **“Apply Changes”.**



Create Azure AI search service

Need of Azure AI search service

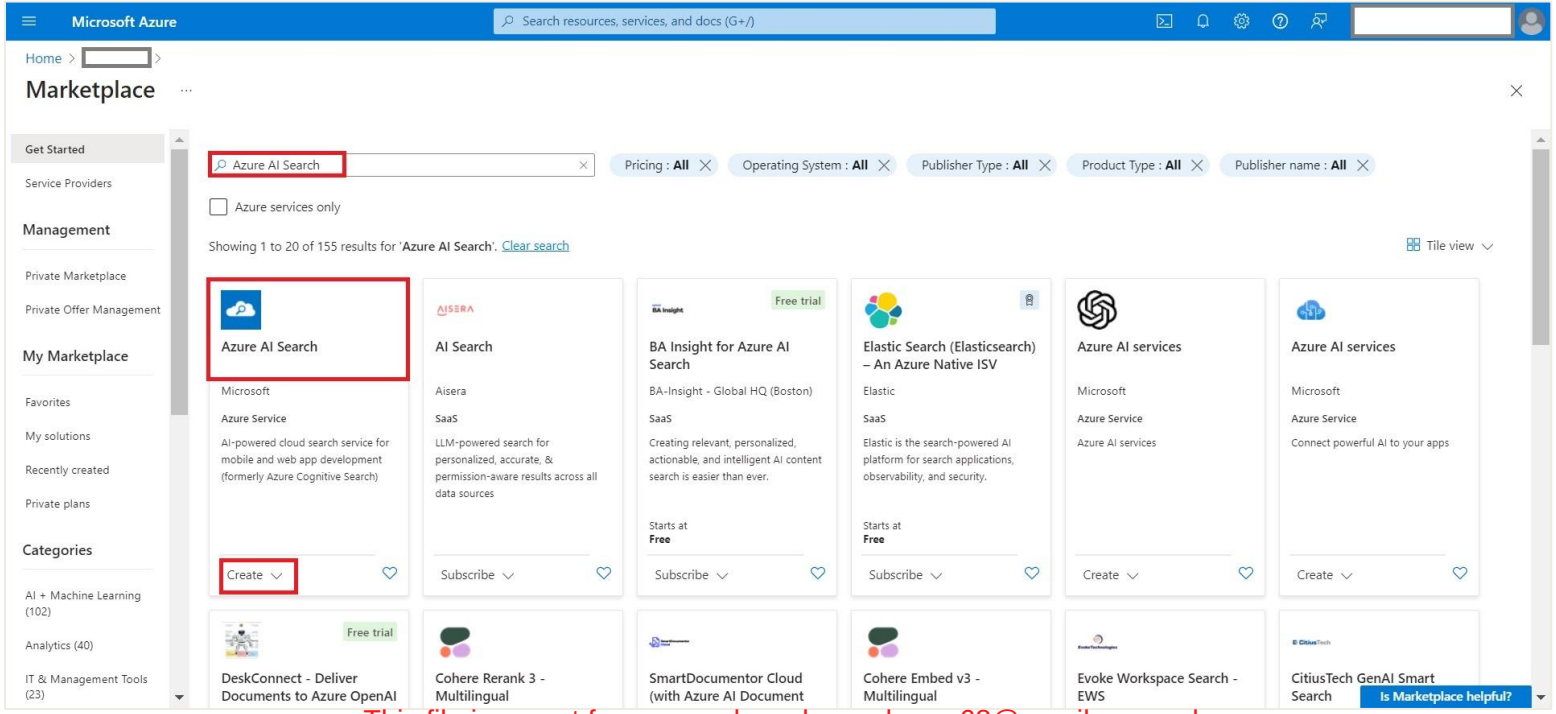
- Integrating Azure AI Search Service with your deployed webapp significantly enhances its capability to retrieve and utilize data efficiently.

- This service enables quick access to relevant information from large datasets, supporting powerful search functionalities like full-text search, filtering, and navigation. By providing precise and contextually relevant answers, it improves the overall user experience, ensuring users receive accurate and helpful responses.

- Seamlessly integrating with extensive datasets from Hugging Face, Azure AI Search Service ensures optimal data utilization and real-time information retrieval. Its scalability and flexibility allows the deployed webapp to handle large volumes of data and adapt to complex queries, making it an indispensable tool for enhancing the webapp''s performance and user satisfaction.

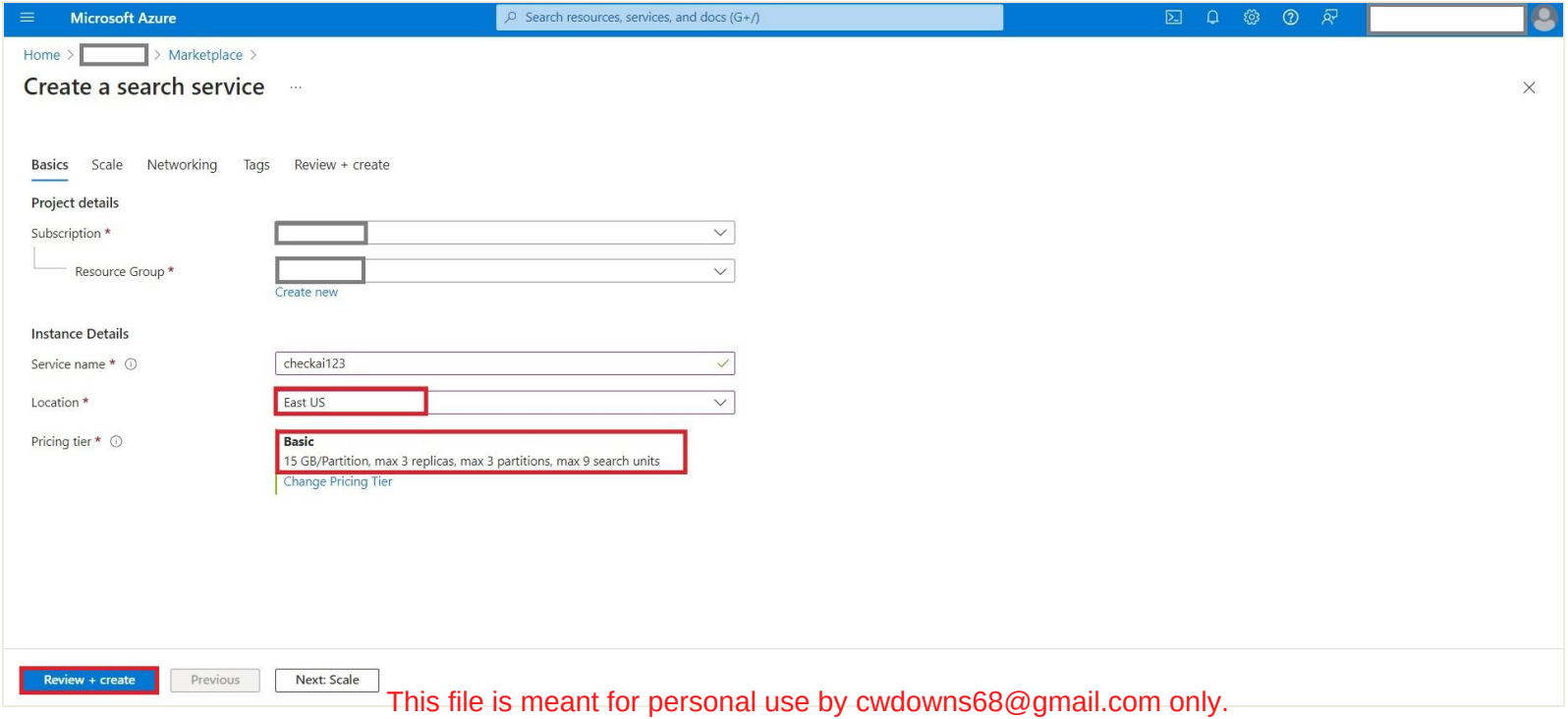
Create Azure AI search service

1. Navigate to **Azure AI search service** under the resource group provisioned.
2. Click on “**Create”** button.



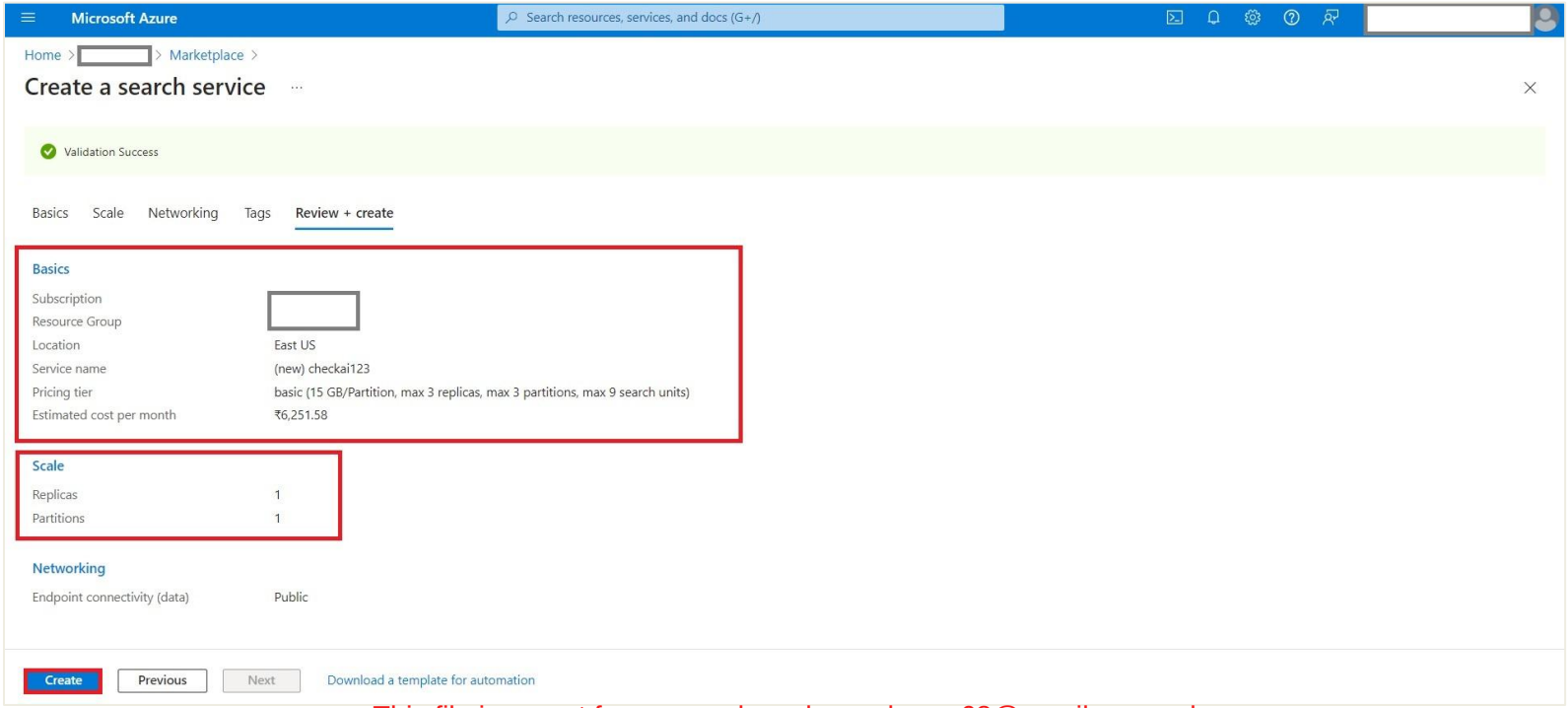
Create Azure AI search service

1. Use **East US location** ad **Basic Pricing Tier**.
2. Click on **“Review + create”** button.



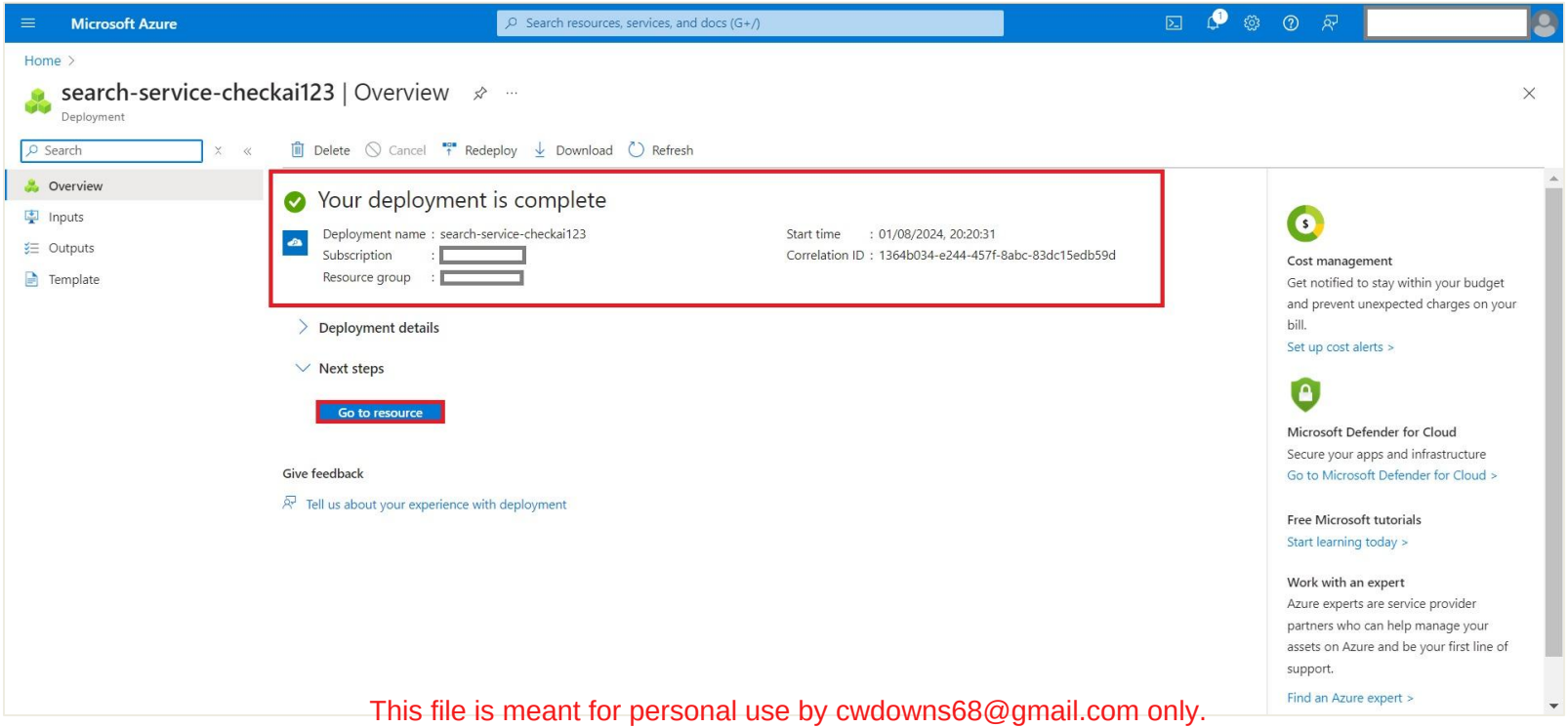
Create Azure AI search service

Click on “**Create”** button.



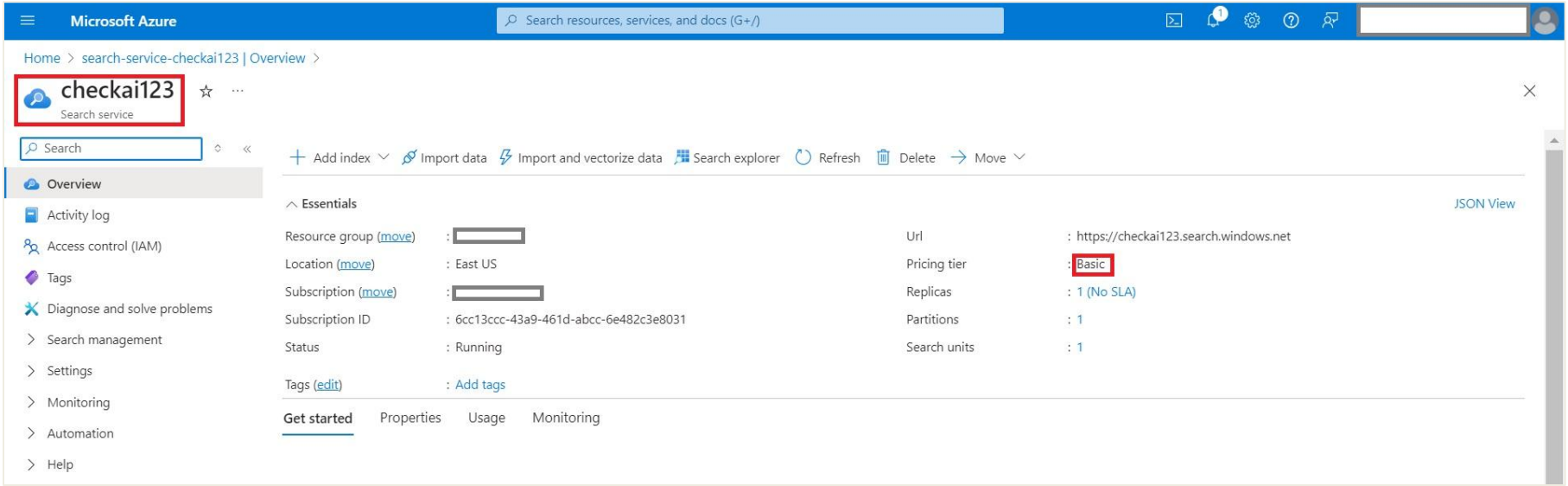
Create Azure AI search service

Deployment will take few minutes. Once it is done, click on **“Go to resource”**.



Create Azure AI search service

You will be able to view the search service created.



Case Study

Case Study

Problem Statement: In the banking sector, efficiently managing customer queries is vital for enhancing service quality and customer satisfaction. Given the diversity of customer intents, ranging from general inquiries to specific service requests, accurately identifying these intents is crucial. This case study focuses on a subset of customer intents related to credit card top-ups. The challenge is to deploy a large language model (LLM) based multi-class classification model and utilize few-shot learning prompts to accurately classify customer queries related to credit card top-ups.

Objective: To improve customer service efficiency and satisfaction by:

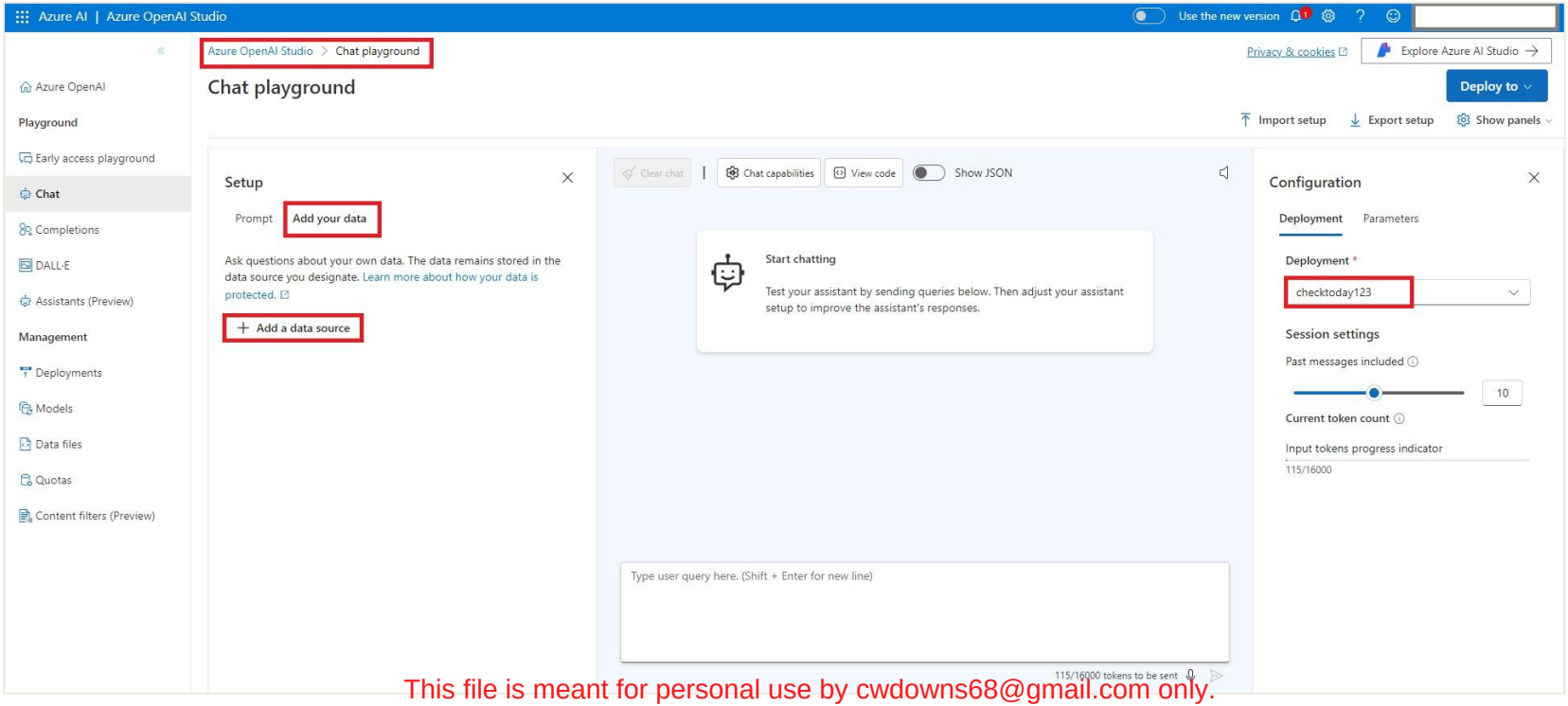
1. **Deploying a Multi-Class Classification Model:** Implement a large language model (LLM) capable of classifying a range of customer intents, specifically targeting queries related to credit card top-ups.
2. **Utilizing Few-Shot Learning Prompts:** Apply few-shot learning techniques to train the model on a banking dataset, enabling it to effectively understand and classify customer intents with minimal examples.
3. **Real-Time Deployment:** Deploy the model using the Azure Chat Playground for real-time analysis of customer queries and categorizing the intents.

By achieving these objectives, the model aims to enhance the accuracy of intent identification and streamline the handling of credit card top-up queries, leading to improved customer service and satisfaction.

Add your Data

Add your Data

Next, navigate back to the Azure OpenAI Chat Playground.
Click on **“+Add a data source”**

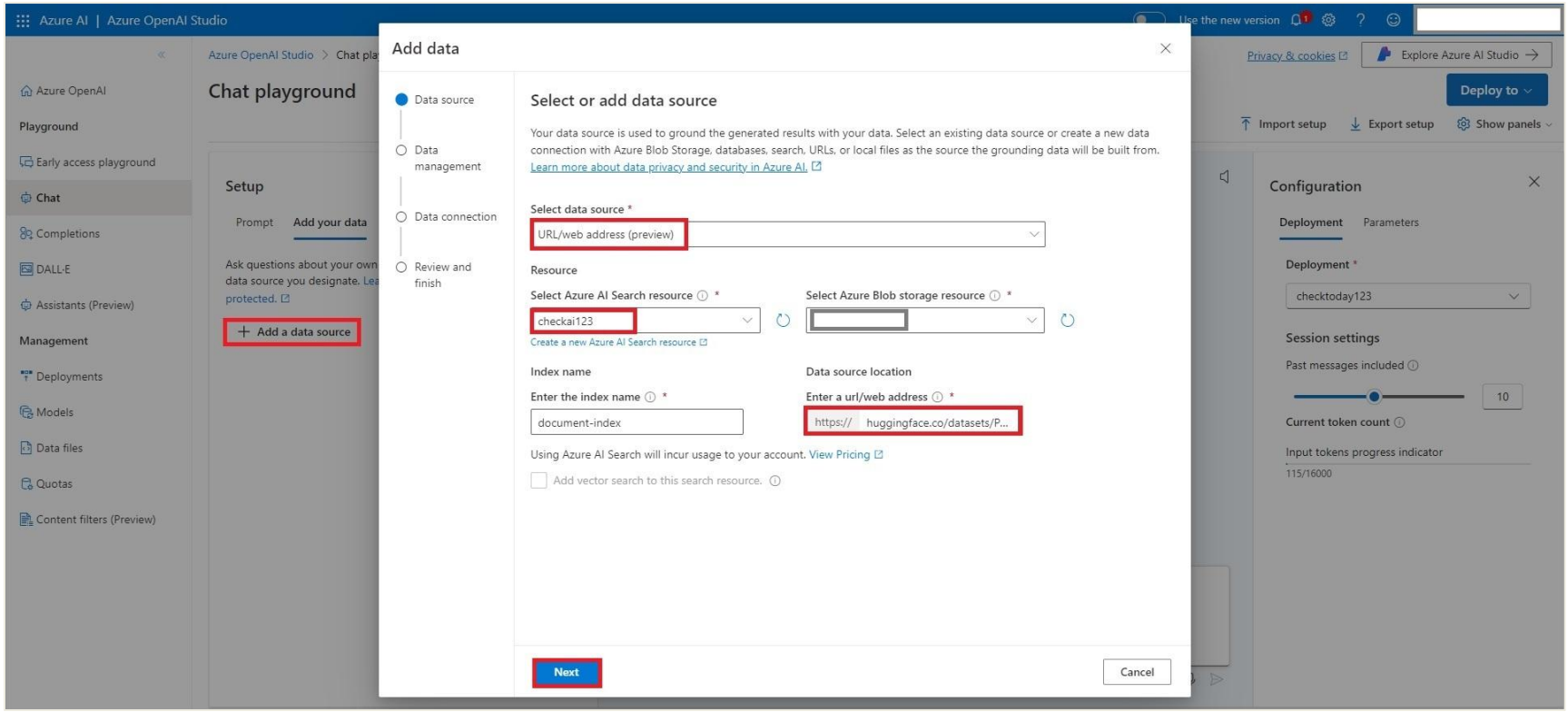


Add your Data

1. We will be using the a **hugging face dataset called banking77** from their dataset repository.
2. Explore the dataset at https://huggingface.co/datasets/PolyAI/banking77
3. There are a total of 77 classes or intents which have been defined. For the purpose of learning, we will pick a few classes (7) which are related to credit card top-ups.
4. **Select data source :** URL/web address (preview)
5. Select the Azure AI search resource created previously
6. Select the Blob Storage resource
7. Enter an index-name of your choice
8. Copy and paste the dataset mentioned in **Step2** in the section **"Enter a url/web address"**
9. Click on **Next**

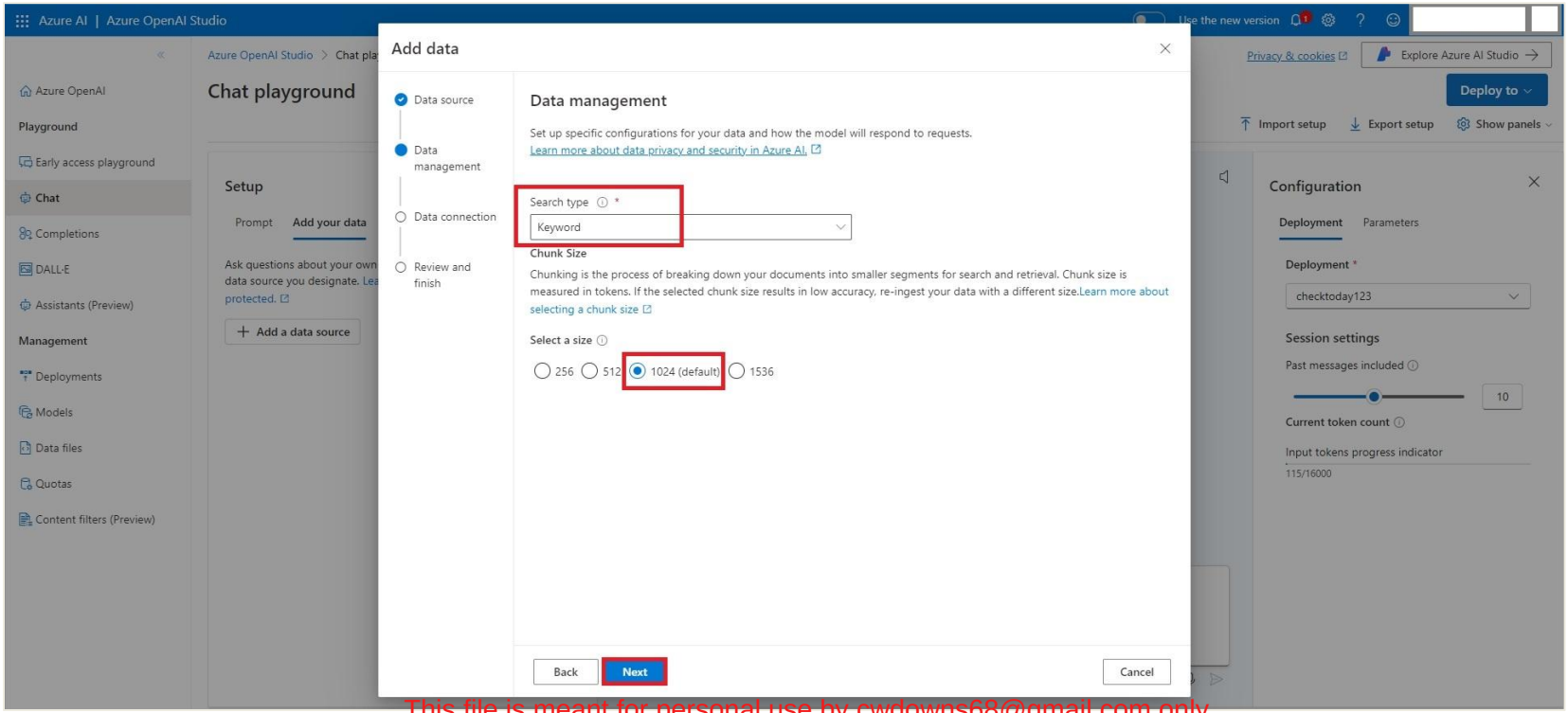
56	top_up_by_bank_transfer_charge
57	top_up_by_card_charge
58	top_up_by_cash_or_cheque
59	top_up_failed
60	top_up_limits
61	top_up_reverted
62	topping_up_by_card

Add your Data



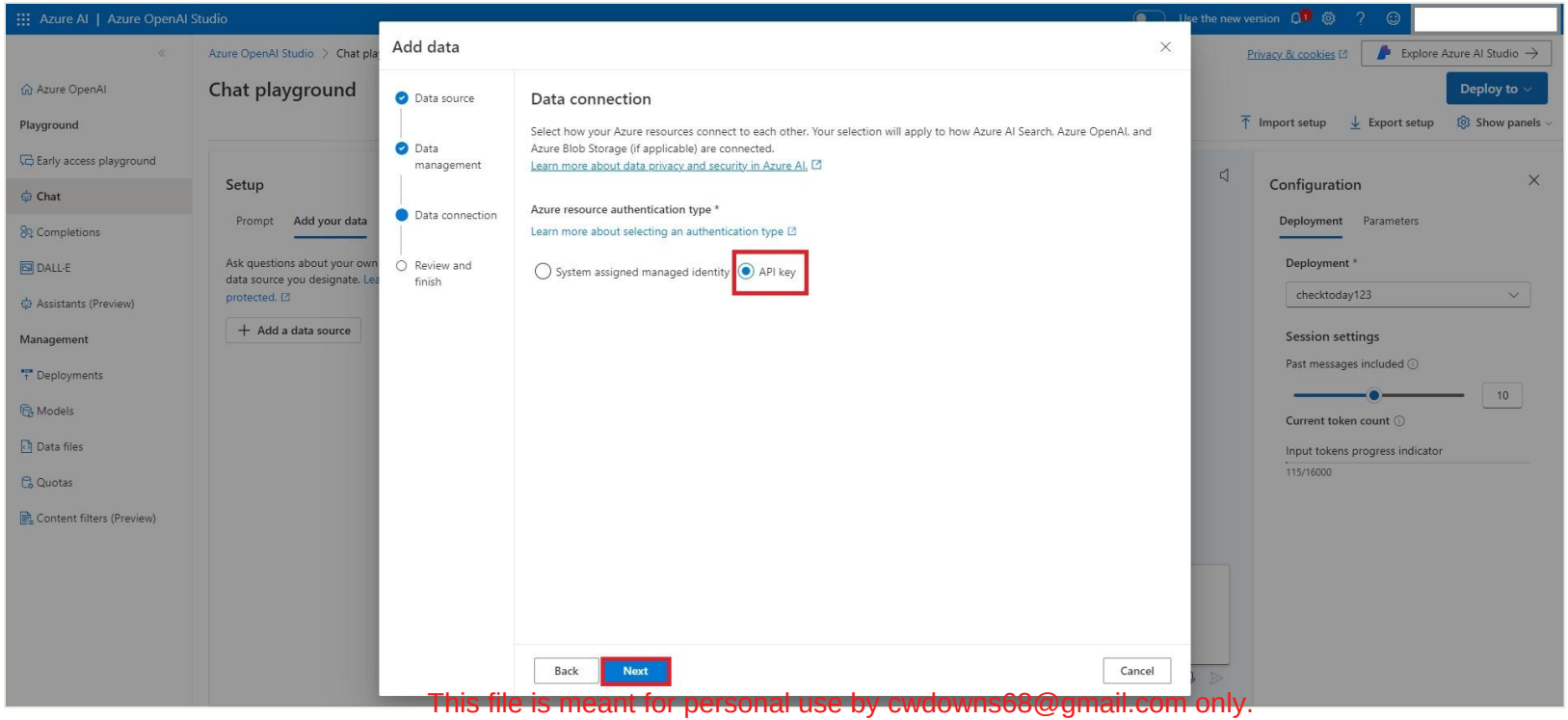
Add your Data

Select **“Keyword”** in Search type and **1024** as Chunk size.
Click on **Next**



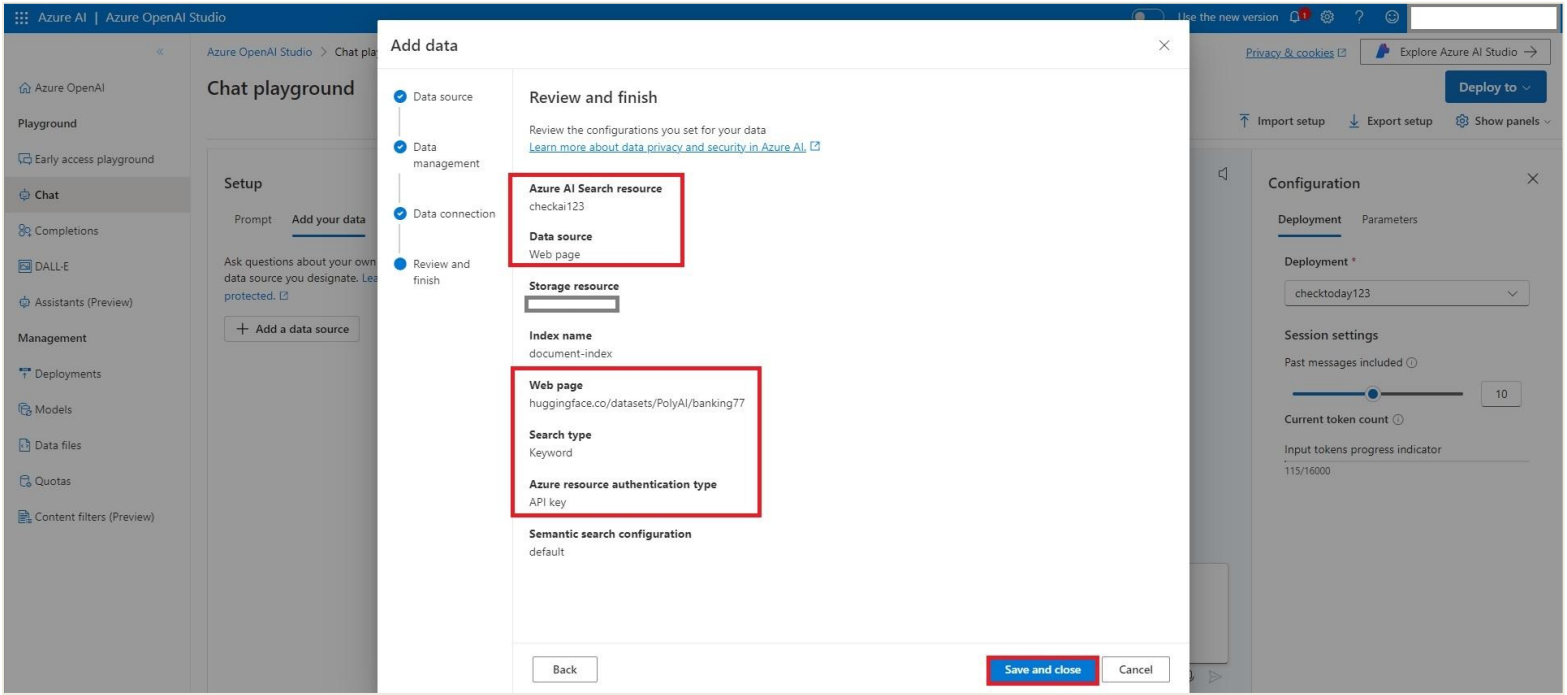
Add your Data

Select **“API Key”** in resource authentication type
Click on **Next**



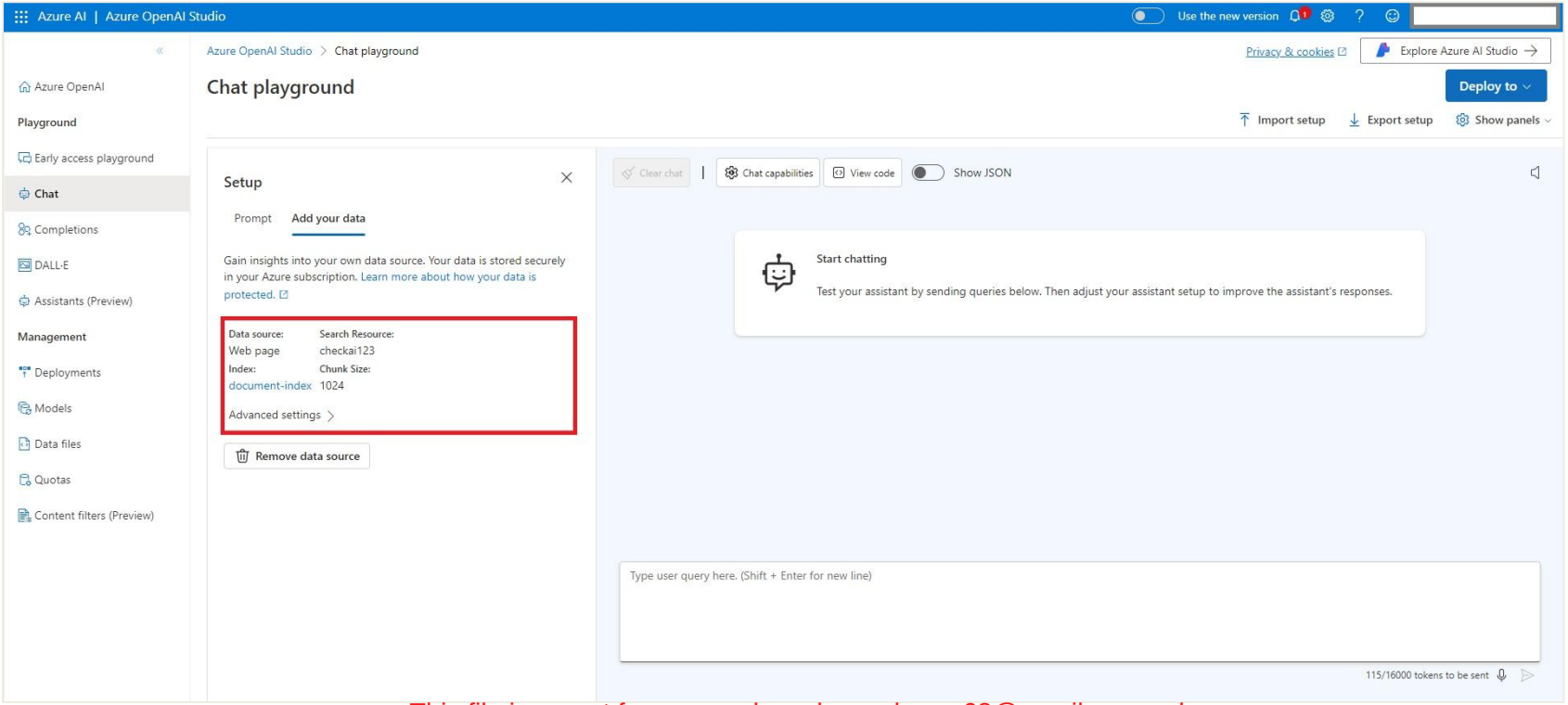
Add your Data

Review the settings.
Click on **Save and Close. Allow 10-15 minutes for the ingestion process.**



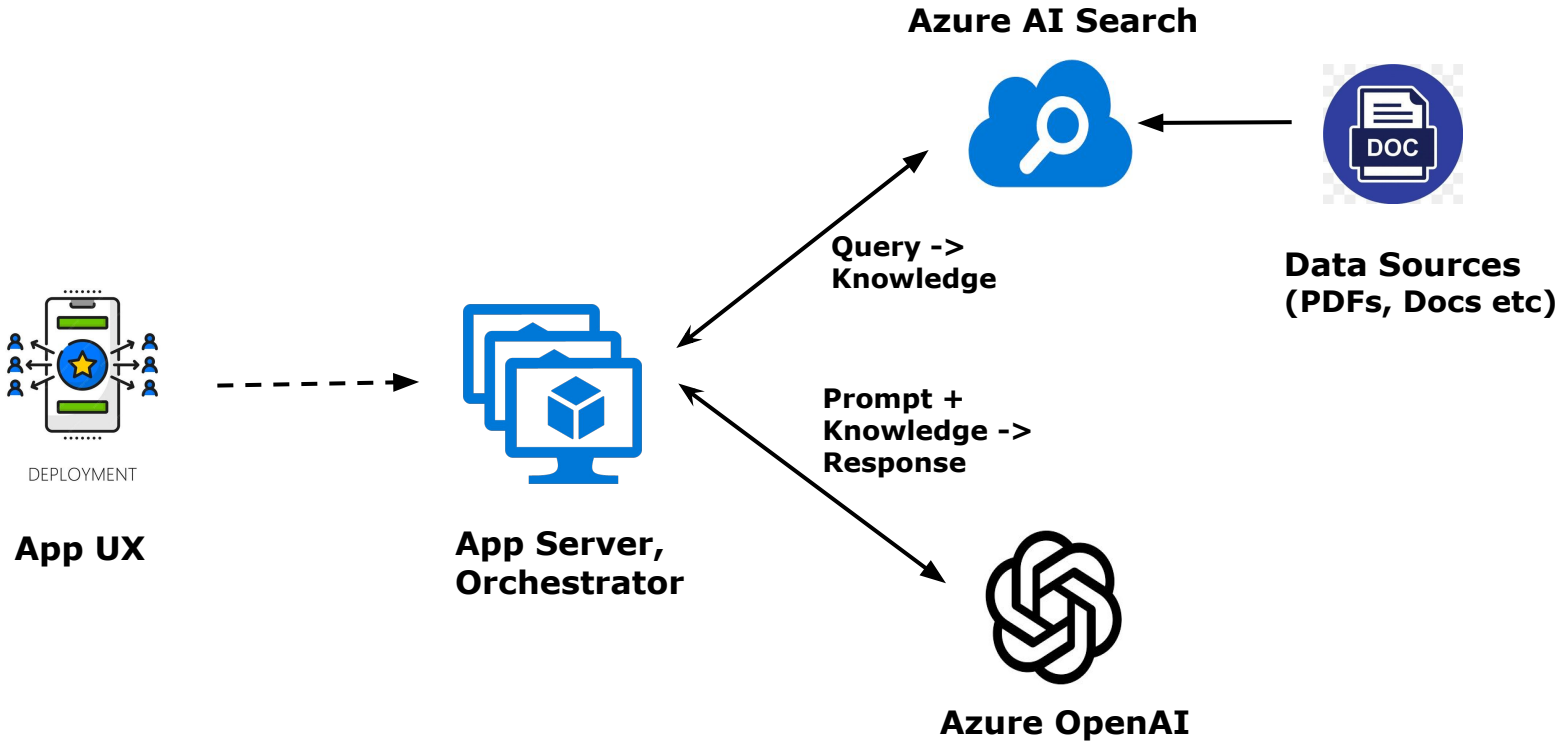
Add your Data

You will now be able to see the data source added under the **“Add your data”** section.



Architecture of Web App Deployment

Architecture of Web App Deployment



Architecture of Web App Deployment

1. **User Query Submission**
   - Users enter queries through the web app interface (App UX).

2. **Query Routing**
   - The App Server & Orchestrator receive and direct the query to Azure AI Search.

3. **Knowledge Retrieval**
   - Azure AI Search processes the query, searches through data sources (e.g., PDFs, documents), and retrieves relevant information.

4. **Prompt Creation & Response Generation**
   - The retrieved knowledge is combined with the original query to create a prompt. Azure OpenAI processes this prompt to generate a detailed response.

5. **Response Delivery**
   - The response is sent back through the App Server & Orchestrator, then delivered to the user via the App UX.

Deploy your model to a web app

Deploy your model to a WebApp

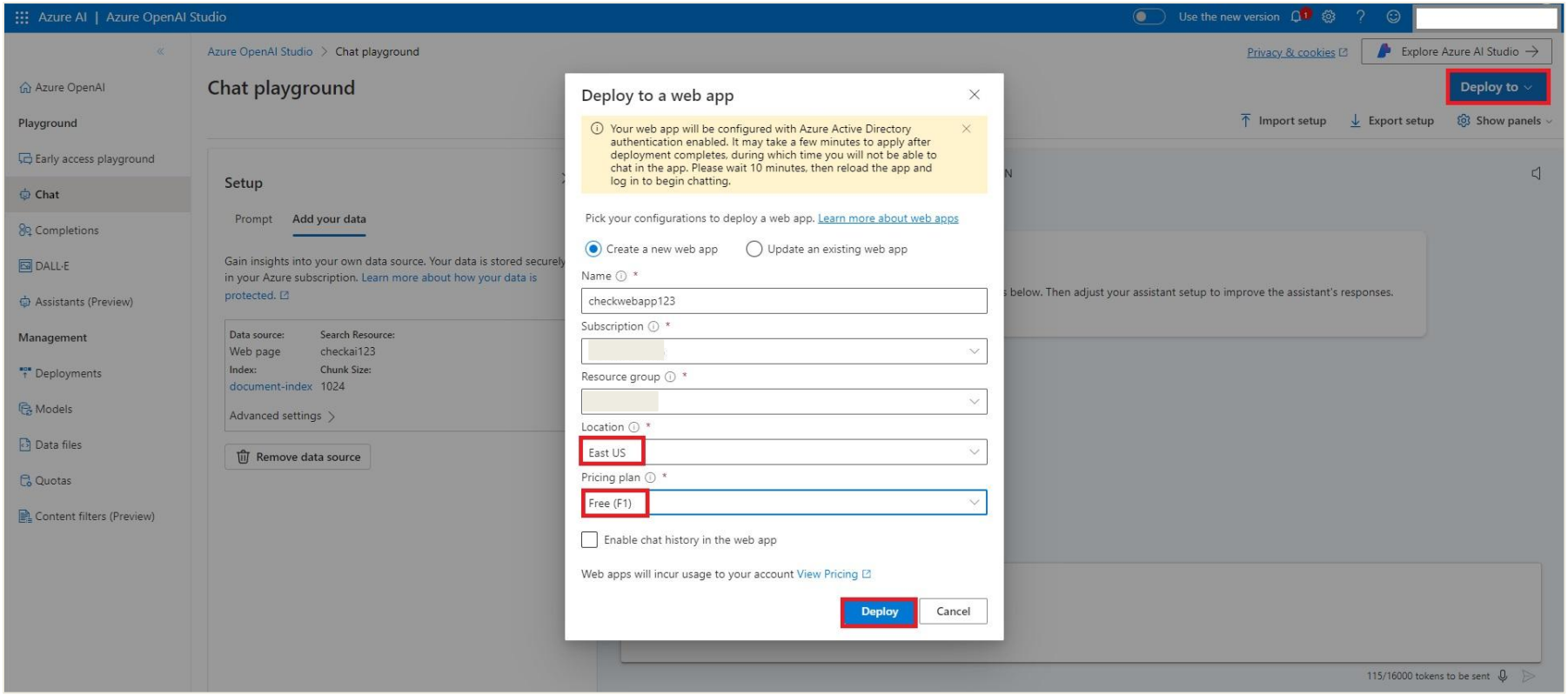
1. At the top right of the Chat playground page, in the **Deploy to menu**, select A new web app.

2. In the Deploy to a web app dialog box, create a new web app with the following settings:
   - Name: *A unique name*
   - Subscription: *Your Azure subscription*
   - Resource group: *The resource group in which you provisioned your Azure OpenAI resource*
   - Locations: *The region where you provisioned your Azure OpenAI resource*
   - Pricing plan: **Free (F1)**
   - Enable chat history in the web app: Unselected
   - I acknowledge that web apps will incur usage to my account: Selected

3. Deploy the new web app and wait for deployment to complete (which may take 15 minutes or so)

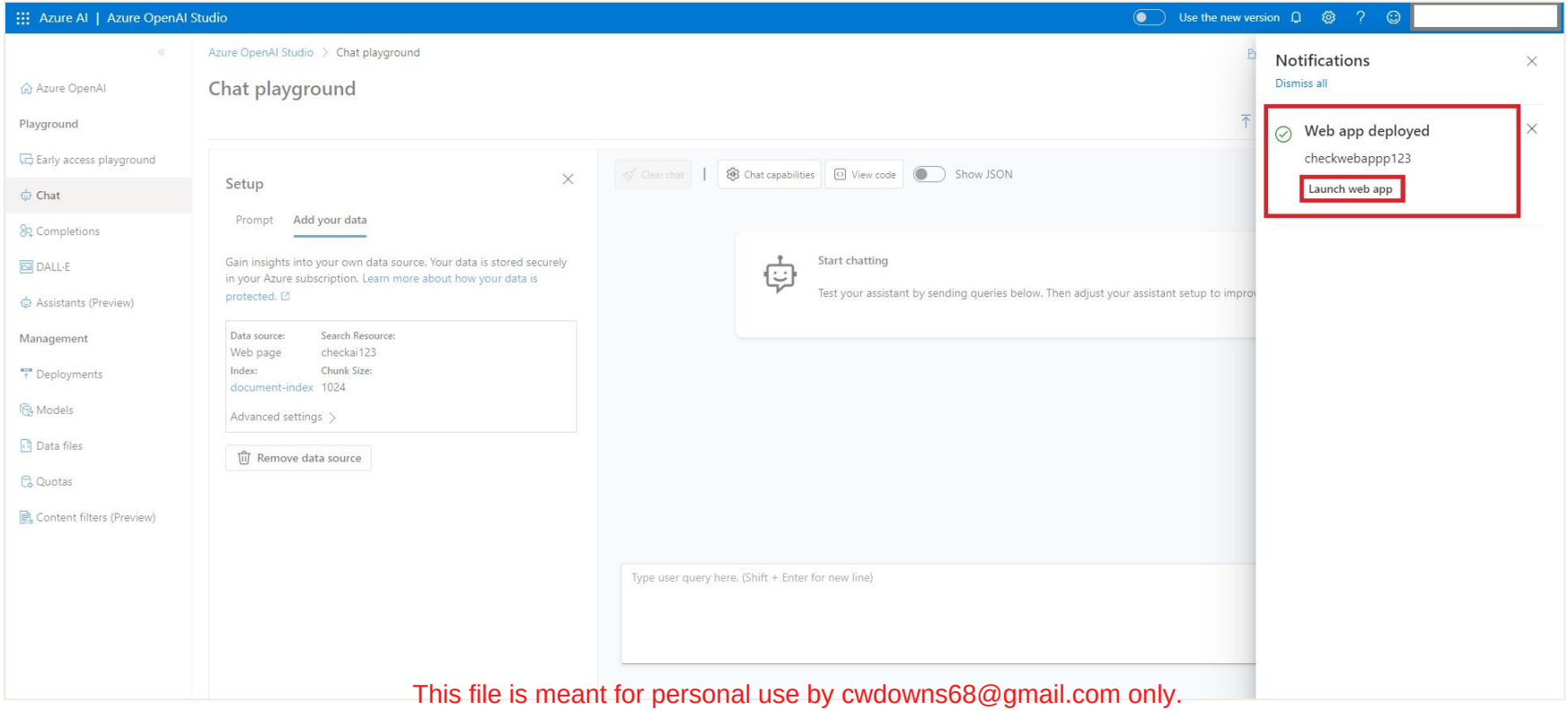
4. After your web app has deployed successfully, use the button at the top right of the Chat playground page to launch the web app. The app may take a few minutes to launch. If prompted, accept the permissions request.

Deploy your model to a WebApp



Deploy your model to a Web App

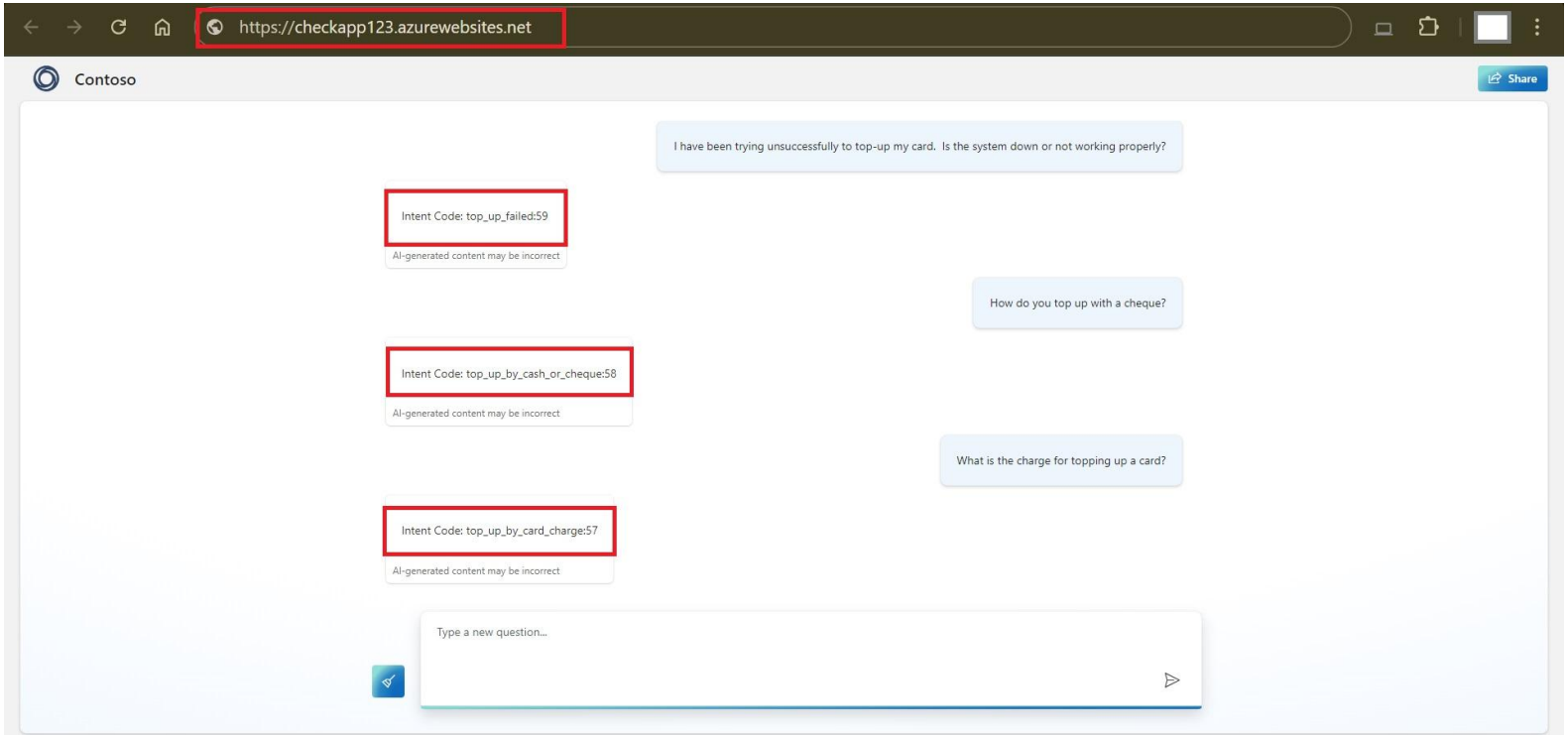
Click on **“Launch web app”**. Wait for 15-20 minutes to launch it.



Test the Web App

Test the Web App

Now, it’s time to test the web app’s response by interacting with it.



Understanding the Responses generated

The outputs you're getting indicate how well your model is classifying the customer intents based on the provided examples. Here's a brief overview of the results:

1. **Top-Up Failure**
   - **Question**: "I have been trying unsuccessfully to top-up my card. Is the system down or not working properly?"
   - **Expected Output**: **59 (top_up_failed)**
   - **Explanation**: This question addresses issues related to failed top-up attempts. The model should recognize this intent and return the code **59.**

2. **Top-Up by Cheque**
   - **Question**: "How do you top up with a cheque?"
   - **Expected Output**: **58 (top_up_by_cash_or_cheque)**
   - **Explanation**: This question is about the process of topping up with a cheque. The model should classify this under the intent for cash or cheque top-ups, returning **58**.

3. **Top-Up Card Charge**
   - **Question**: "What is the charge for topping up a card?"
   - **Expected Output**: **57 (top_up_by_card_charge)**
   - **Explanation**: This question inquires about fees associated with card top-ups. The model should classify this intent and return **57**.

Summary

Summary

1. The model's outputs reveal its proficiency in categorizing customer queries related to card top-ups.

2. It effectively identifies issues with unsuccessful top-ups, questions about using cheques, and inquiries about card charges.

3. This accurate classification enables the business to promptly address customer concerns, streamline support processes, and provide precise information on transaction issues and methods.

4. By ensuring that customer queries are classified correctly, the model enhances user satisfaction and operational efficiency, allowing the business to respond more effectively to diverse customer needs.

Clean Up

Clean Up

When you're done with your Azure OpenAI resource, remember to delete the deployment or the entire resource in the Azure portal at https://portal.azure.com.

Great Learning

Happy Learning !