Build a C# Console App that Uses Azure AI Services to Process and Analyze Survey Results

A guide for beginners who want to learn how to use Azure Cognitive Services for text analysis

# Introduction

In this lab, you will learn how to use Azure AI Services to process a form of survey results and do sentiment analysis on the results. By the end of this lab, you will have a report that contains the survey results, as well as the sentiment score.

# Prerequisites

To complete this lab, you will need:

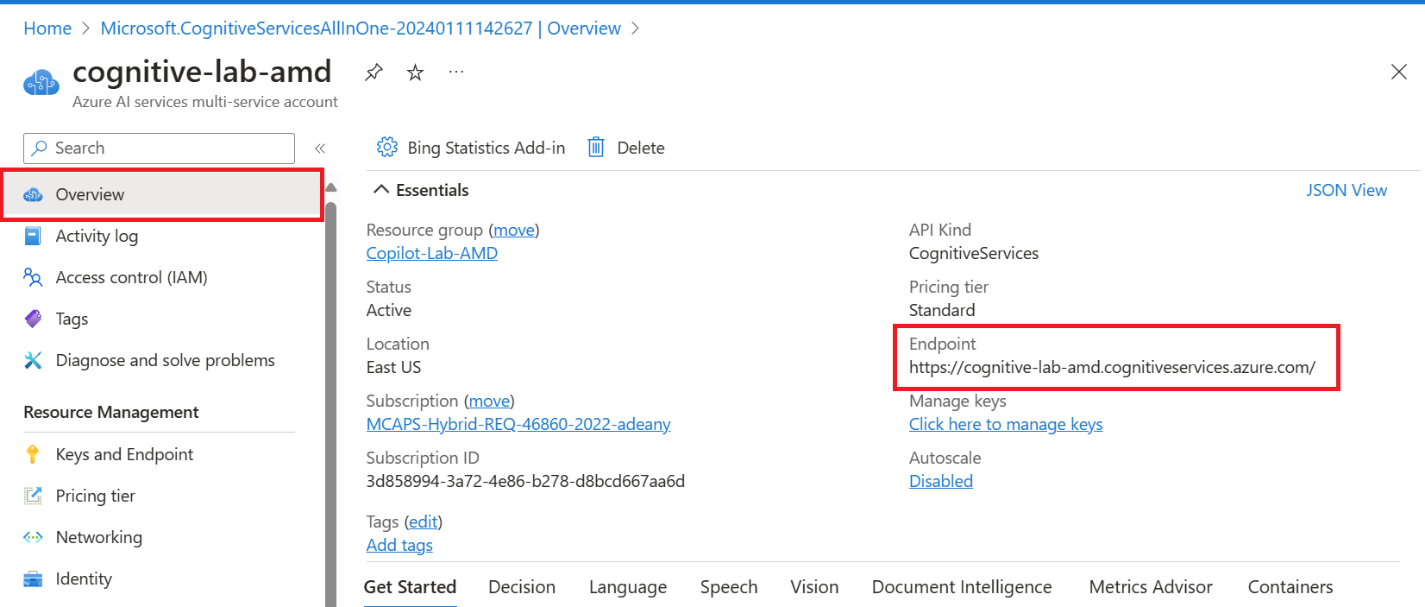
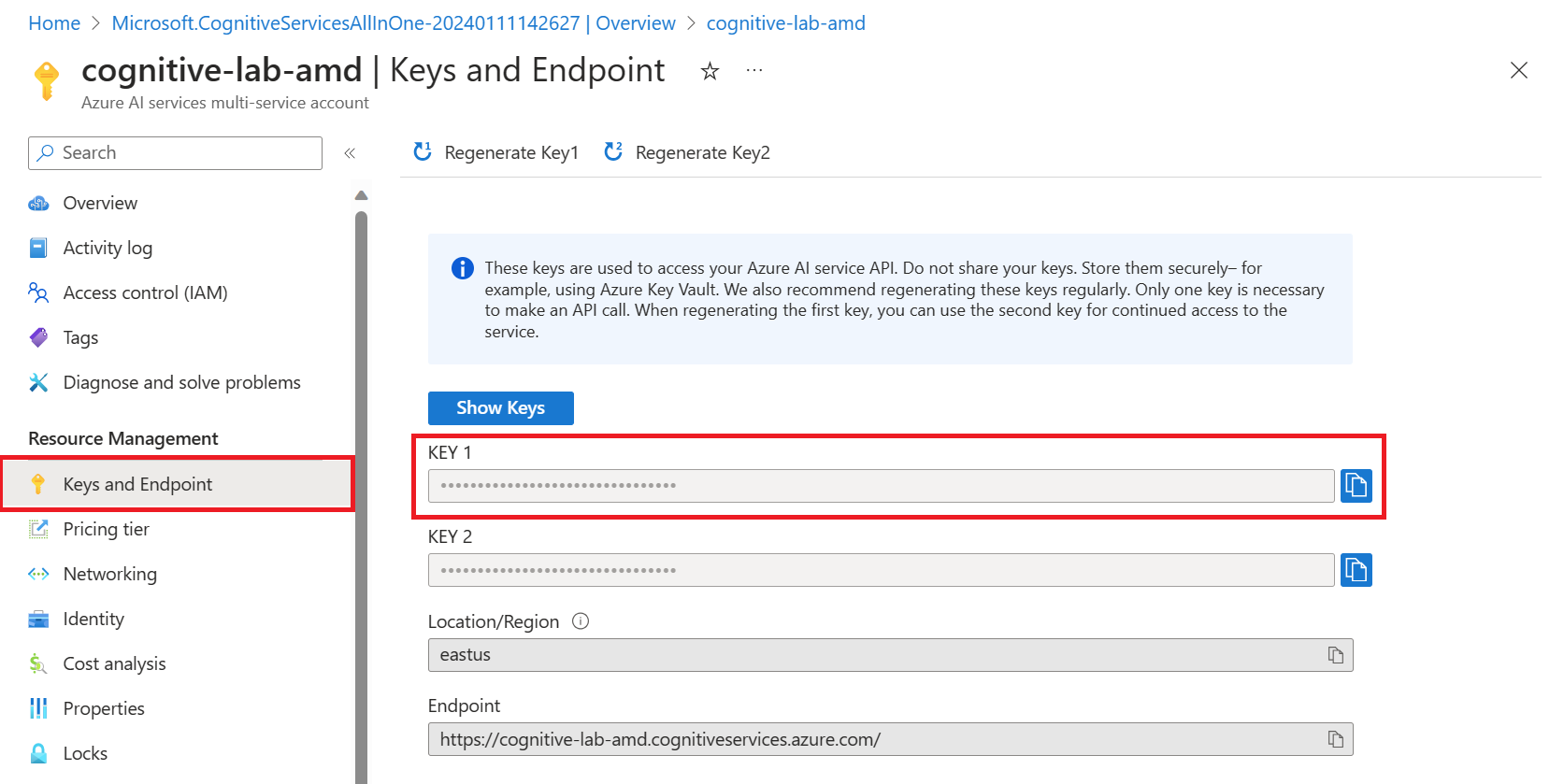
* An Azure subscription
* VSCode with dotnet

# Step 1: Create an Azure AI Services Resource

The first step is to create an Azure Cognitive Services resource that will provide access to the Text Analytics and Translator APIs. To do this, follow these steps:

1. Log in to the **Azure portal** (https://portal.azure.com) and click on the "**Create a resource**" button on the top left corner.
2. Search for "**Azure AI Services**" and select the option with the same name. **A screenshot of a chat

   Description automatically generated**
3. Click on the "**Create**" button and fill in the required fields, such as the resource name, the subscription, the resource group, the location, and the pricing tier. You can choose any name and location, but make sure to select the "S0" pricing tier, which gives you 5,000 transactions per month for free.  
     
   A screenshot of a computer

   Description automatically generated
4. Click on the "**Review + create**" button and then on the "**Create**" button to confirm the creation of the resource.
5. Once the resource is created, click “**Go to resource**”
6. On the **Overview page**, copy the endpoint URL. You will need this later to connect to the APIs.  
     
   
7. On the **Keys and** Endpoints page, copy one of the keys. You will need this later to connect to the APIs.  
     
   

# Step 2: Create a C# Console Application Project

Follow these steps to complete the lab:

1. Open VS Code and navigate to a new folder.
2. Open a new terminal and type: **dotnet new console --framework net8.0 --use-program-main**. A new console app template should be created. **A screen shot of a computer program

   Description automatically generated**
3. Open Copilot Chat and prompt Copilot to create a survey form. For example, you could type something like: **Create a C# application that creates a survey form. The survey will ask 5 questions that ask employees to rate their satisfaction on a scale of 1 to 10.**  
     
   A black background with white text

   Description automatically generated
4. Copilot should return something like this:  
     
   A screenshot of a computer program

   Description automatically generated
5. Hover over the suggested code and click “**Copy**”. **Paste** the contents of the Program class in your existing Program.cs file. Click **Ctrl + S** to save.
6. On your terminal, run the program by typing: **dotnet run.** You will be prompted to fill out your survey. The output will look something like this:  
     
   A screenshot of a computer program

   Description automatically generated
7. Next, we will want to save the results. Prompt Github Copilot to save the results to a text file. You can say something like this: **Modify Program.cs to write the results of my survey to a txt file.**  
     
   A black screen with white text

   Description automatically generated
8. Copilot will suggest how to add this to your code. You should see something like this:  
     
   A screenshot of a computer program

   Description automatically generated
9. **Copy** the changes into your Program.cs file. You can test these changes by running **dotnet run.** After you take the survey, a new file named **survey\_results.txt** should show up in your folder. The contents of the file should look something like this:  
     
   A screenshot of a computer

   Description automatically generated
10. Now that we have survey results available, let’s use the Azure AI Services API with the results. First, we will need to import the namespaces for the APIs. At the top of your **Program.cs** file, prompt Copilot to add the using statements for you by prompting it with an in-line comment. You can type something like: **// Use Azure AI Text Analytics API.** After the prompt, you can enter **Ctrl + Tab** and Copilot should add the following: **A screen shot of a computer code

    Description automatically generated**
11. Prompt Copilot to create the client object by typing something like: **// Create a Text Analytics client.** After the prompt, you can enter **Ctrl + Tab** and Copilot should add the following:   
      
    A screen shot of a computer

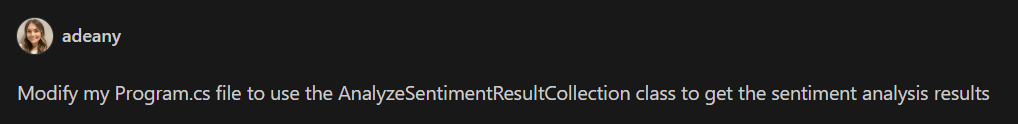
    Description automatically generated
12. Make sure **to replace the analytics key and endpoint** with the values you copied in Step 1.
13. Prompt Copilot to read the results, call the sentiment analysis method and get the scores for each response, and print the sentiment score. It should look something like this:  
    A screen shot of a computer code

    Description automatically generated
14. In the terminal, run **dotnet run.** You may see errors. If so, we need to add the Nuget packages for the Azure API. Run **dotnet add package Azure.AI.TextAnalytics –version 5.2.0.** You will also need to run **dotnet add package Azure.Core.** The packages will be added. Your .csproj file should look something like this:   
      
    A screen shot of a computer program

    Description automatically generated
15. Let’s try this again. Run **dotnet run.** I got some errors regarding my use of SentimentBatchResult.  
    A screen shot of a computer

    Description automatically generated
16. Let’s ask Copilot how to fix these errors. I typed: **How do I fix the error “\_\_\_\_”.**  
      
    A black background with white text

    Description automatically generated
17. Copilot informed me that the SentimentBatchResult class was not a part of the TextAnalytics namespace and provided an alternative solution: to use the AnayzeSentimentResultCollection class.  
      
    A screenshot of a computer program

    Description automatically generated
18. Looks like we have to use **AnalyzeSentimentResultCollection** class. Let’s ask Copilot how we would do this with a prompt: **Modify my Program.cs file to use the AnalyzeEntimentResultCollection class to get the sentiment analysis results.**  
      
    
19. **Review** and **copy** the suggestion. Your new Program class should look something like this:  
      
    A screenshot of a computer program

    Description automatically generated
20. Run **dotnet run.** The application should run without error. If we open up **survey\_results.txt,** we will see that the sentiment analysis (Positive, Negative) is now added to each of the ratings:  
      
    A screen shot of a computer

    Description automatically generated

# Conclusion

In this lab, you learned how to use Azure AI Services to process a form of survey results to do sentiment analysis on the results. You used the Azure AI Service to perform these tasks. You also created a report that contains the survey results, as well as the overall sentiment score. You can use these skills to create similar reports for other types of forms and documents.