

**Lab Manual- Azure Language AI services Lab**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn916

**Contributor:**

Contents

[1. Objective 3](#_Toc167361527)

[2. Provision an Azure AI Services resource 3](#_Toc167361528)

[3. Copy Endpoint and Key 7](#_Toc167361529)

[4. Clone the repository in Visual Studio Code 8](#_Toc167361530)

[5. Configure Endpoint and Key 9](#_Toc167361531)

[6. Use REST 11](#_Toc167361532)

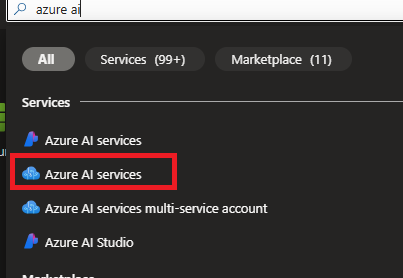
# Objective

In this exercise, you'll get started with Azure AI Services by creating an **Azure AI Services** resource in your Azure subscription and using it from a client application. The goal of the exercise is not to gain expertise in any particular service, but rather to become familiar with a general pattern for provisioning and working with Azure AI services as a developer.

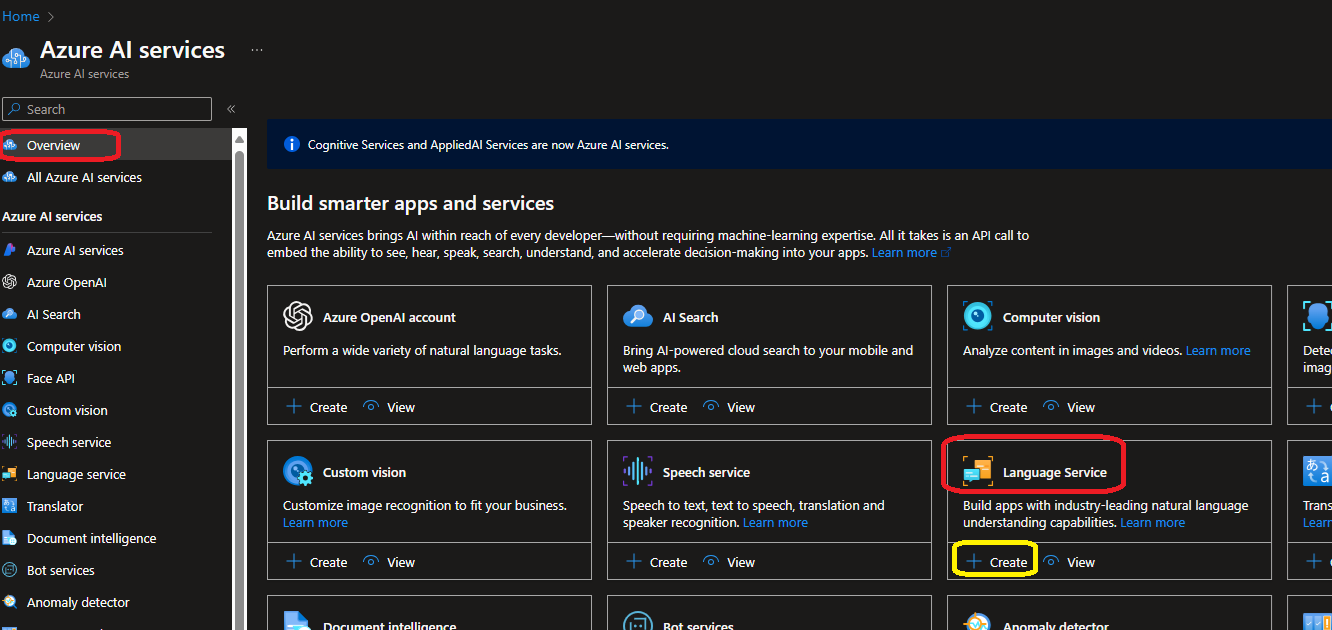
# Provision an Azure AI Services resource

Azure AI Services are cloud-based services that encapsulate artificial intelligence capabilities you can incorporate into your applications. You can provision **individual Azure AI services** resources for **specific APIs** (for example, **Language** or **Vision**), or you can provision a single Azure AI Services resource that provides access to multiple Azure AI services APIs through a single **endpoint** and **key**. In this case, you'll use a single Azure AI Services resource

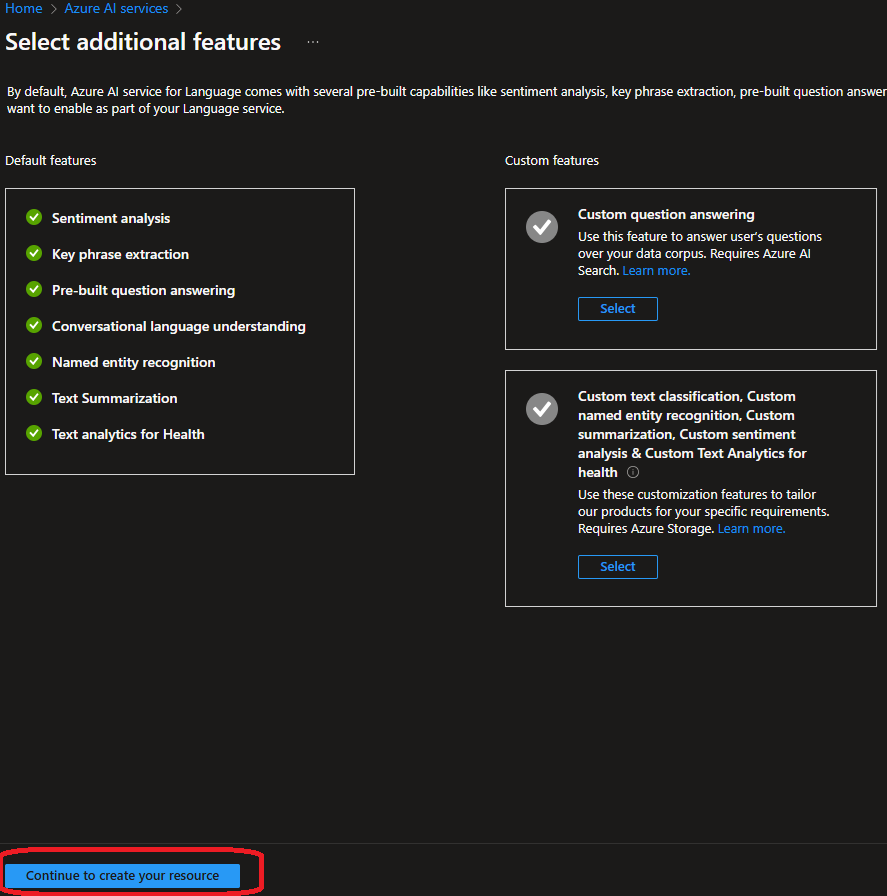
* Serach AI Service and select **Azure AI Services**



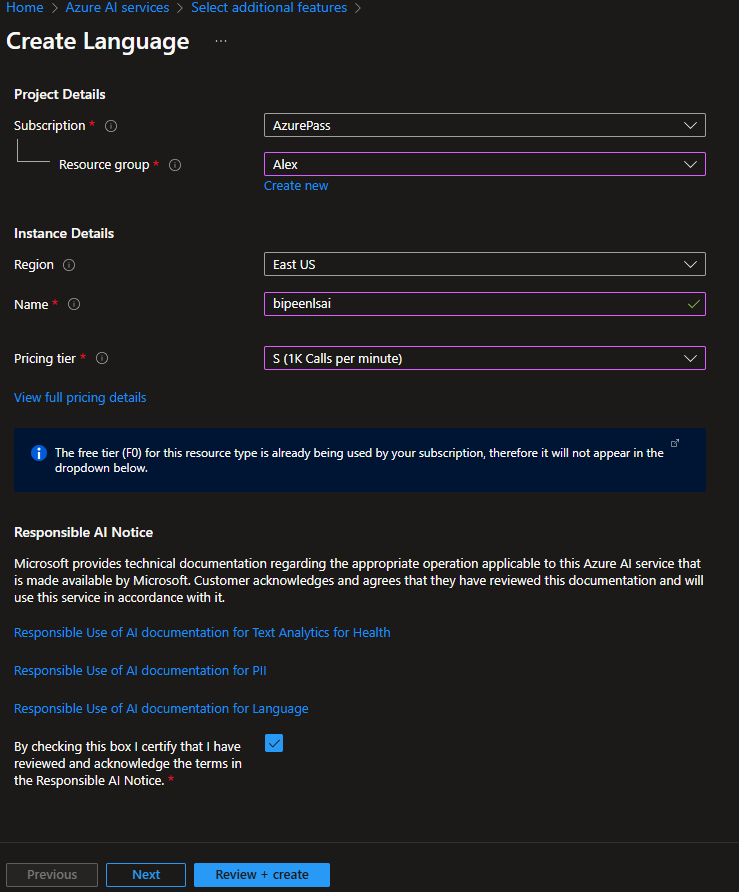
* Select **Language** and Click **create**



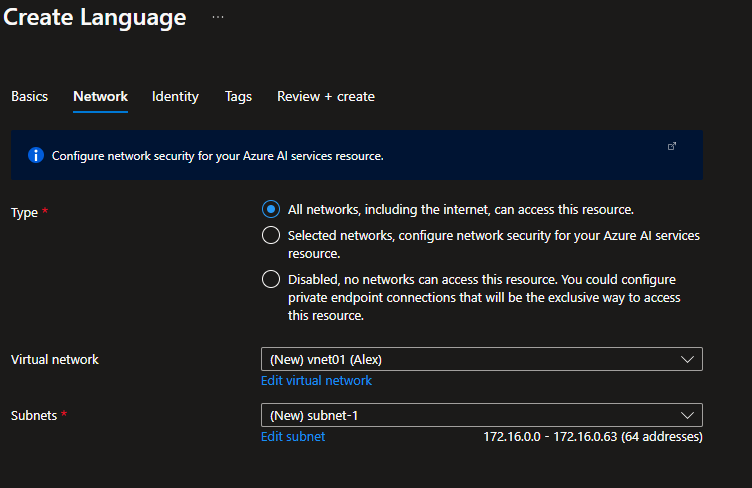
* Click **Continue to create your resource**



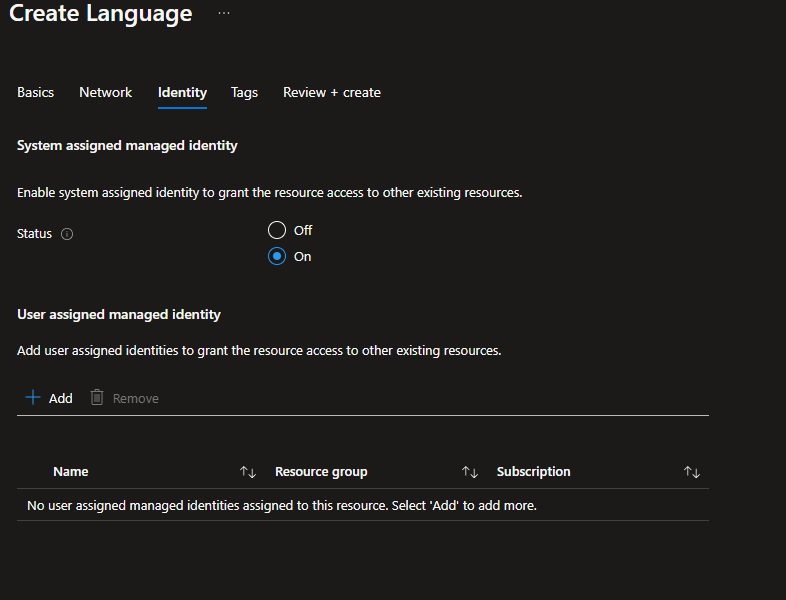
* Select your
  + Resource Group
  + Name of AI Instance
  + Pricing should be **FREE** ( Select S-Standard if free is not abalable)



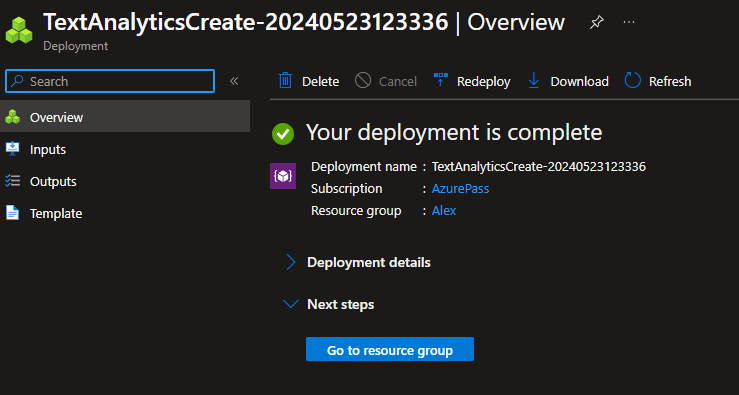
* Click **Next and Leave the default**



* Click **Next and Leave the default**

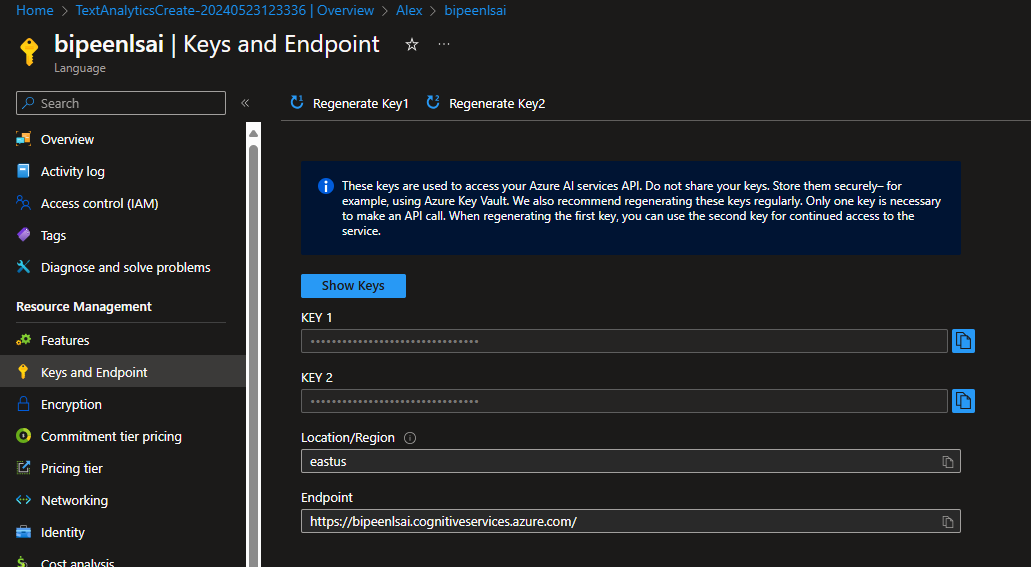


* Click **Review and Create**



# Copy Endpoint and Key

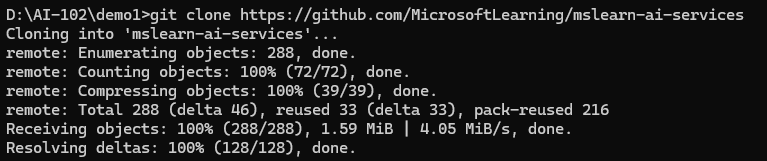
1. Wait for deployment to complete, and then view the deployment details.
2. Go to the resource and view its **Keys and Endpoint** page. This page contains the information that you will need to connect to your resource and use it from applications you develop. Specifically:
   * An HTTP *endpoint* to which client applications can send requests.
   * Two *keys* that can be used for authentication (client applications can use either key to authenticate).
   * The *location* where the resource is hosted. This is required for requests to some (but not all) APIs.



# Clone the repository in Visual Studio Code

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the https://github.com/MicrosoftLearning/mslearn-ai-services repository to a local folder (it doesn't matter which folder).

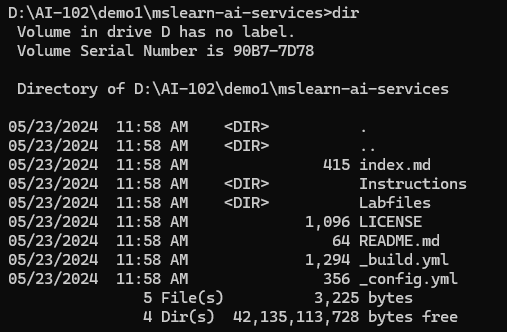
**git clone https://github.com/MicrosoftLearning/mslearn-ai-services**



1. Go Inside **mslearn-ai-services** directory

**cd mslearn-ai-services**

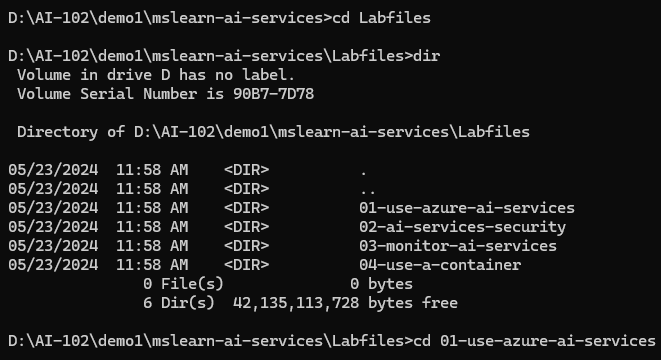
****

****

1. Go Inside **01-use-azure-ai-services** directory under **Labfiles**

**Cd Labfiles**

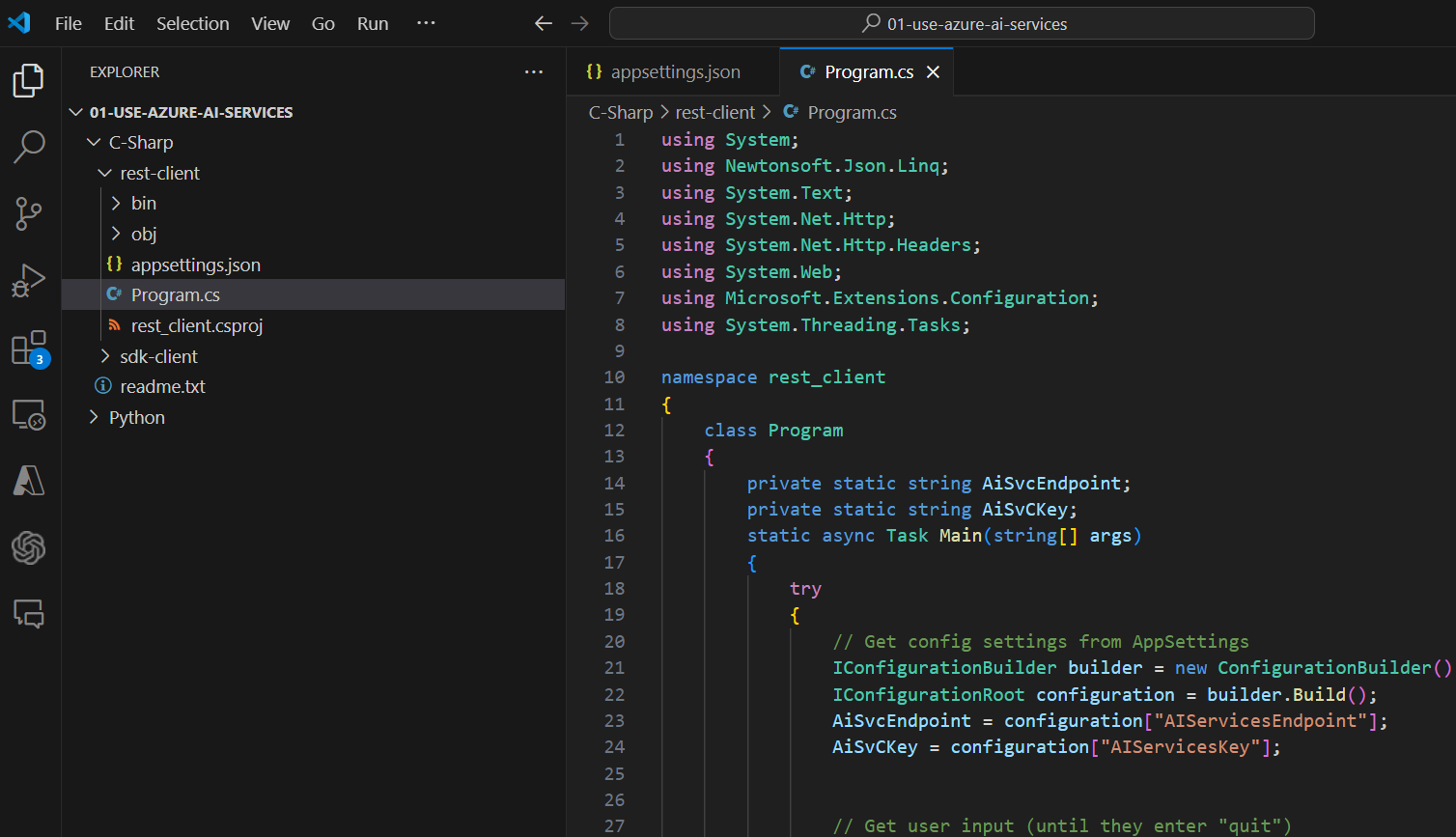
**cd 01-use-azure-ai-services**

****

1. Open the folder in Visual Studio Code.

**Code .**

****

****

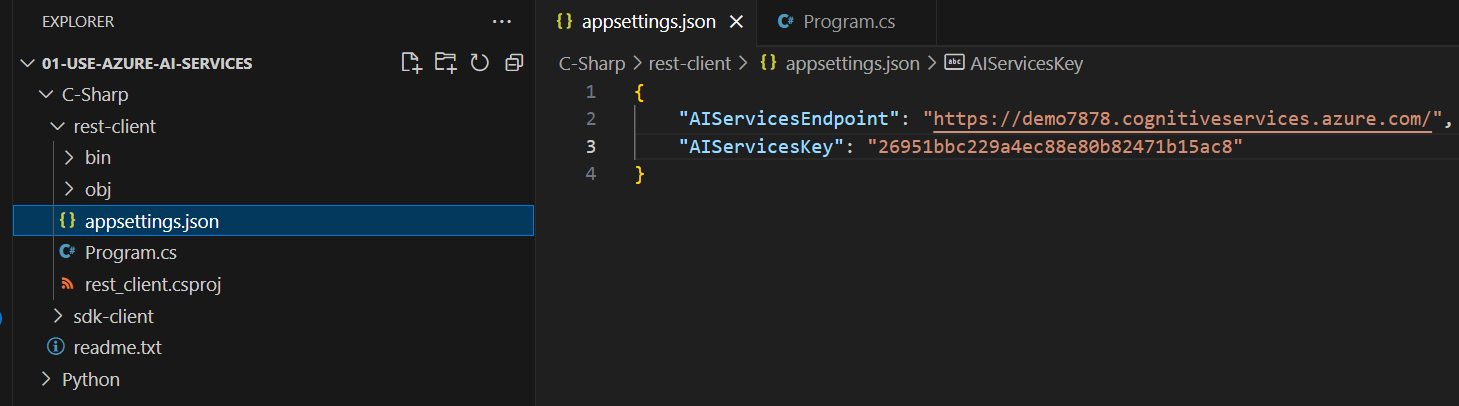
# Configure Endpoint and Key

The Azure AI services APIs are REST-based, so you can consume them by submitting JSON requests over HTTP. In this example, you'll explore a console application that uses the **Language** REST API to perform language detection; but the basic principle is the same for all of the APIs supported by the Azure AI Services resource.

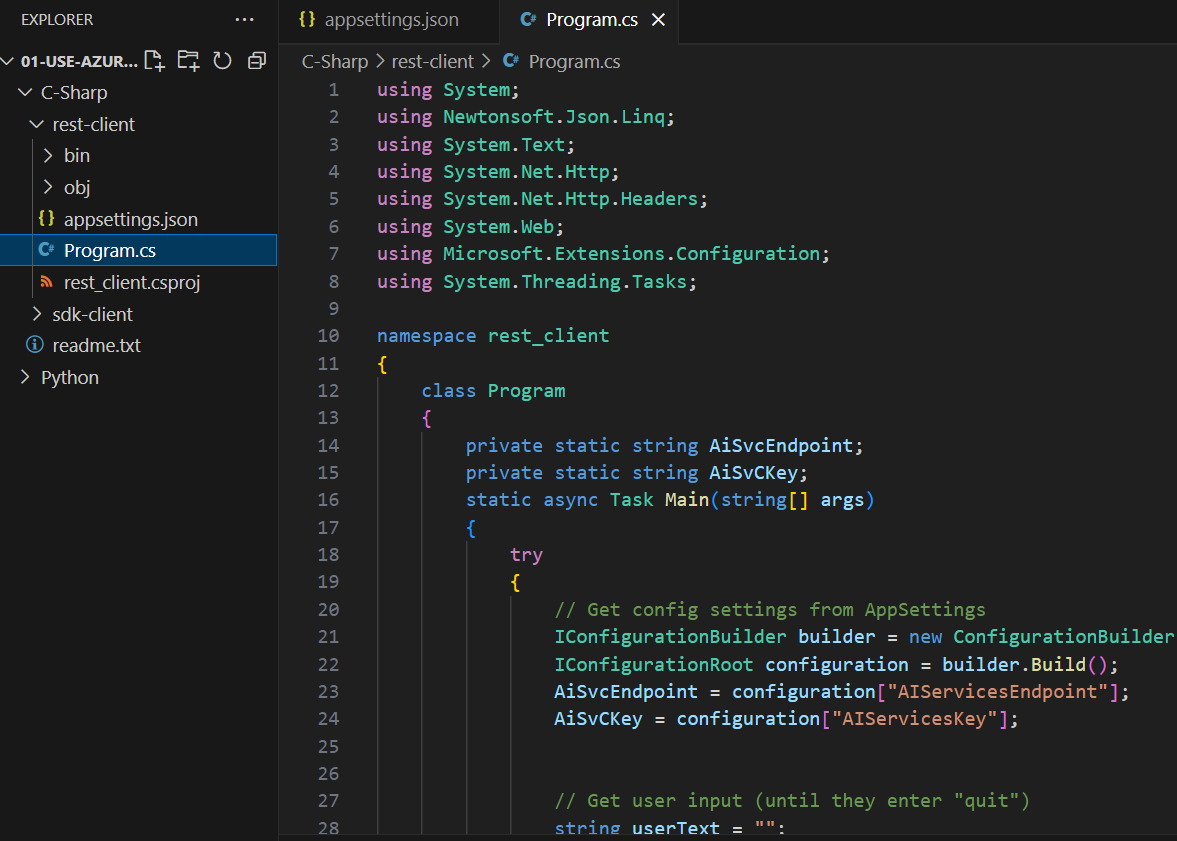
* In Visual Studio Code, expand the **C-Sharp**
* View the contents of the **rest-client** folder, and note that it contains a file for configuration settings:

appsettings.json

* Open the configuration file and update the configuration values it contains to reflect the **endpoint** and an authentication **key** for your Azure AI services resource. **Save your** changes.



Open the code file **Program.cs** and review the code it contains, noting the following details:

* Various namespaces are imported to enable HTTP communication
* Code in the **Main** function retrieves the endpoint and key for your Azure AI services resource - these will be used to send REST requests to the Text Analytics service.
* The program accepts user input, and uses the **GetLanguage** function to call the Text Analytics language detection REST API for your Azure AI services endpoint to detect the language of the text that was entered.
* The request sent to the API consists of a JSON object containing the input data - in this case, a collection of **document** objects, each of which has an **id** and **text**.
* The key for your service is included in the request header to authenticate your client application.
* The response from the service is a JSON object, which the client application can parse.

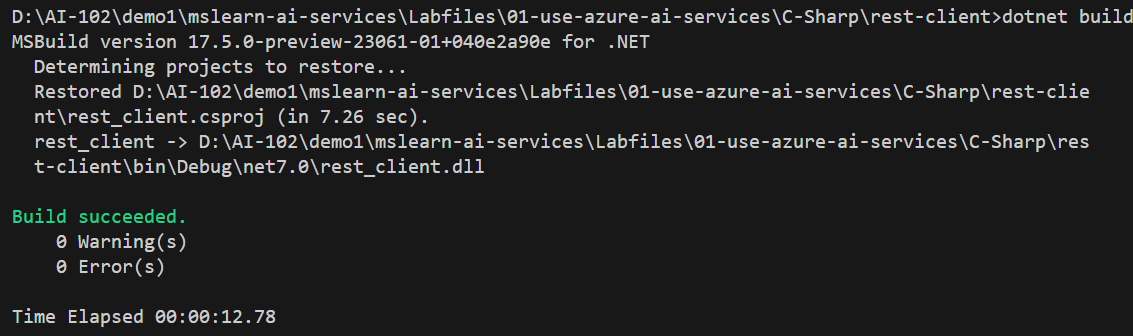
# Use REST

* In VS Code terminal go inside **C-Sharp\rest-client**



* Type

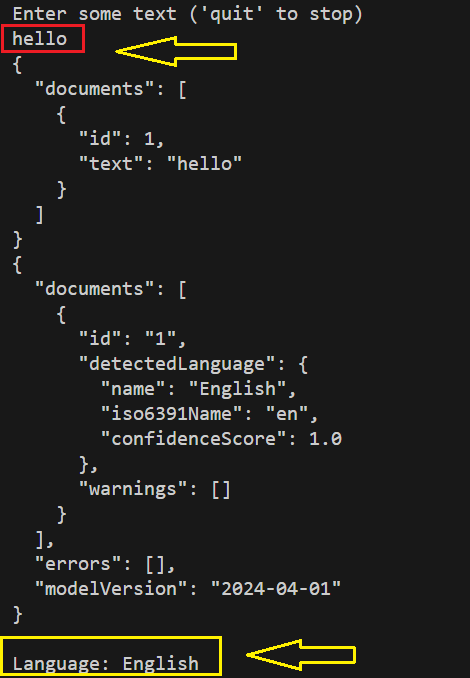
**Dotnet build**



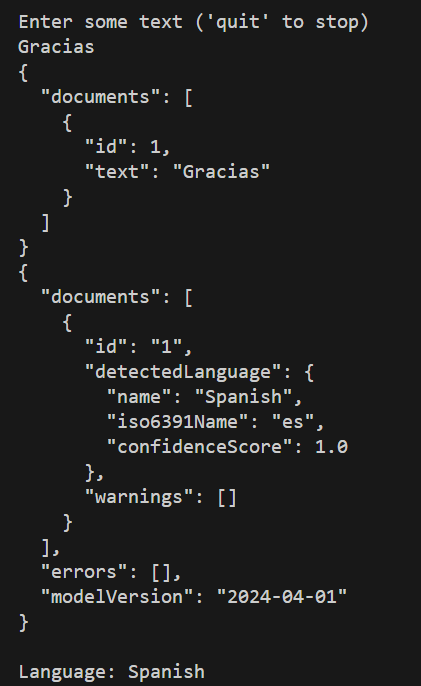
**Dotnet Run**

****

1. When prompted, enter some text and review the language that is detected by the service, which is returned in the JSON response. For example, try entering "**Hello**", "**Bonjour**", and "**Gracias**".







Enter quit to exit

