

**Lab Manual- Analyze a document using Azure AI Document Intelligence**

**Prepared for**:

**Date:9th May** 2024

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn018

**Contributor:**

Contents

[1. Objective 3](#_Toc167809888)

[1. Learning objectives 3](#_Toc167809889)

[2. Provision an Azure AI Document Intelligence resource 3](#_Toc167809890)

[3. Use the Read model 7](#_Toc167809891)

[4. Prepare to develop an app in Visual Studio Code 13](#_Toc167809892)

[5. Configure your application 13](#_Toc167809893)

[6. Add code to use the Azure Document Intelligence service 15](#_Toc167809894)

# Objective

Many forms and documents that your business handles are common across disparate companies in different sectors. For example, most companies use invoices and receipts. Microsoft Azure AI Document Intelligence includes prebuilt models so you can handle common document types easily.

You work for a company that conducts polls for private companies and political parties. Participants submit their responses as paper forms or as online PDFs. You've decided to deploy Azure AI Document Intelligence to streamline data entry and you need to know if you can use the prebuilt models to generate meaningful data from your forms.

In this module, you'll learn about the capabilities of the prebuilt models in Azure AI Document Intelligence and how to use them.

## Learning objectives

At the end of this module, you'll be able to:

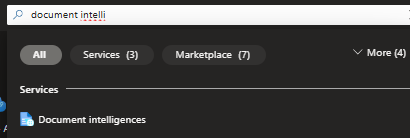
* Identify business problems that you can solve by using prebuilt models in Azure AI Document Intelligence.
* Analyze forms by using the General Document, Read, and Layout models.
* Analyze forms by using financial, ID, and tax prebuilt models.

In this exercise, you'll set up an Azure AI Document Intelligence resource in your Azure subscription. You'll use both the Azure AI Document Intelligence Studio and C# to submit forms to that resource for analysis.

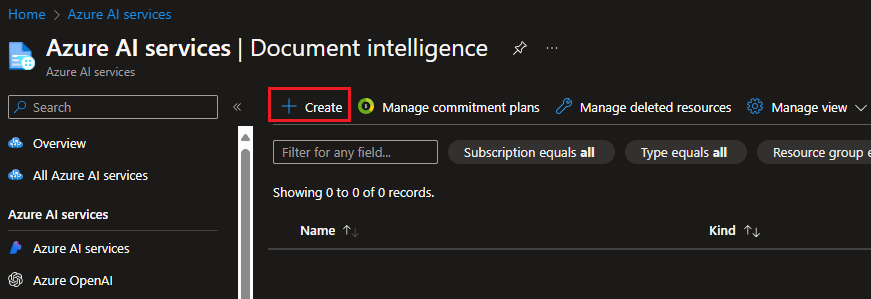
# Provision an Azure AI Document Intelligence resource

Before you can call the Azure AI Document Intelligence service, you must create a resource to host that service in Azure:

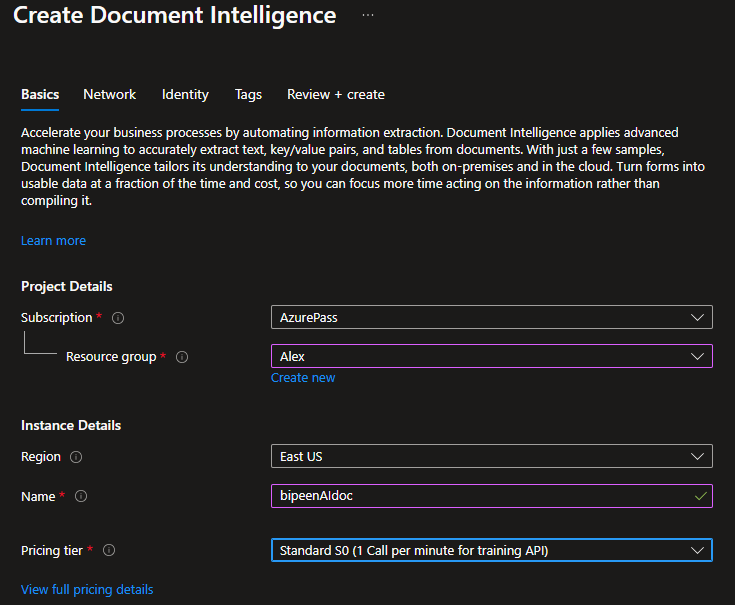
1. In a browser tab, open the Azure portal at [https://portal.azure.com](https://portal.azure.com/?azure-portal=true), signing in with the Microsoft account associated with your Azure subscription.
2. On the Azure portal home page, navigate to the top search box and type **Document Intelligence** and then press **Enter**.



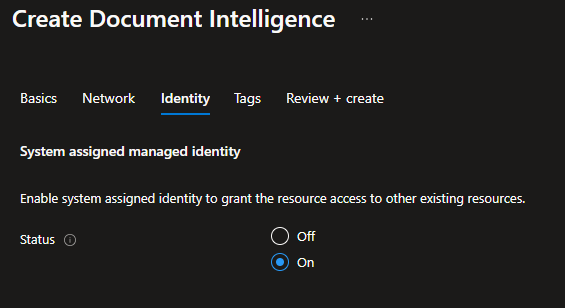
1. On the **Document Intelligence** page, select **Create**.



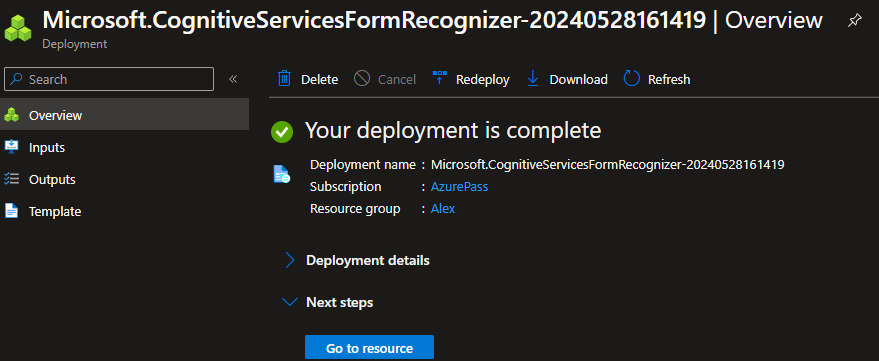
1. On the **Create Document Intelligence** page, use the following to configure your resource:
   * **Subscription**: Your Azure subscription.
   * **Resource group**: Select or create a resource group with a unique name such as DocIntelligenceResources.
   * **Region**: select a region near you.
   * **Name**: Enter a globally unique name.
   * **Pricing tier**: select **Free F0** (if you don't have a Free tier available, select **Standard S0**).

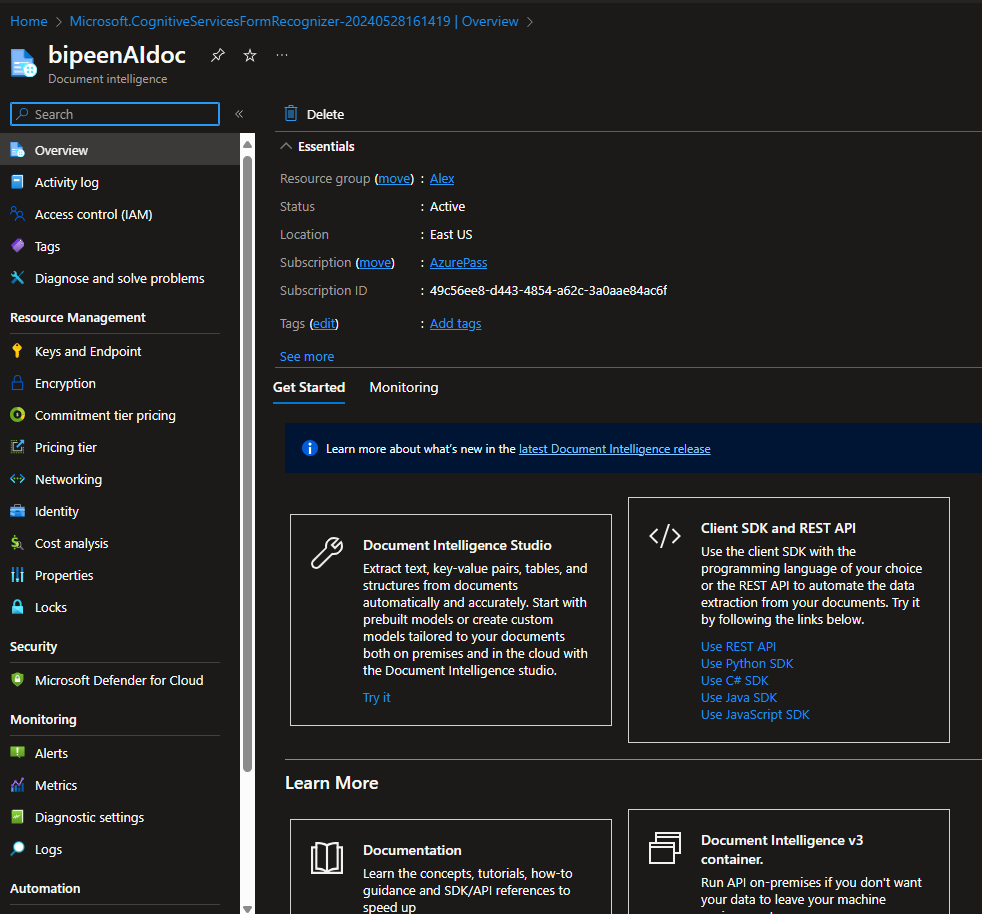


1. Click Next to identity and on the System Assigned Identity (Optional)



1. Then select **Review + create**, and **Create**. Wait while Azure creates the Azure AI Document Intelligence resource.
2. When the deployment is complete, select **Go to resource**. Keep this page open for the rest of this exercise.

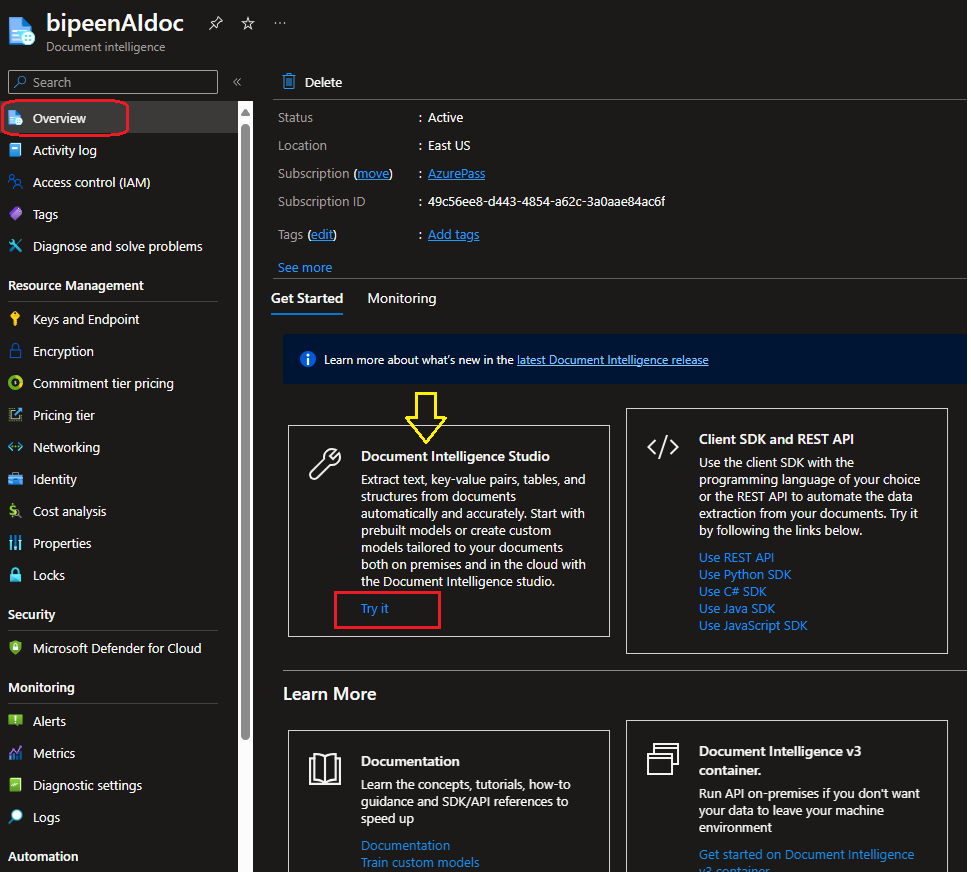


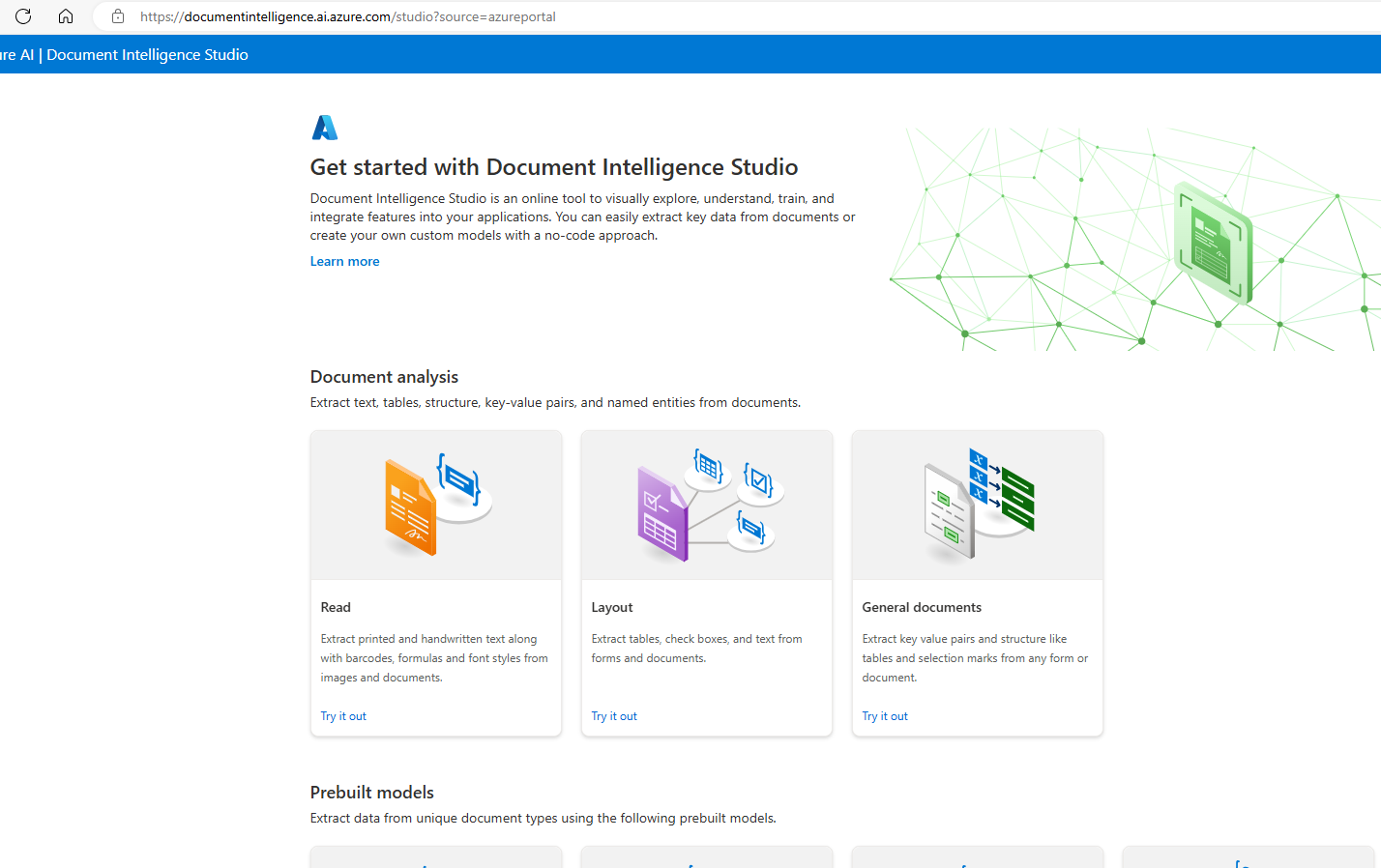
****

# Use the Read model

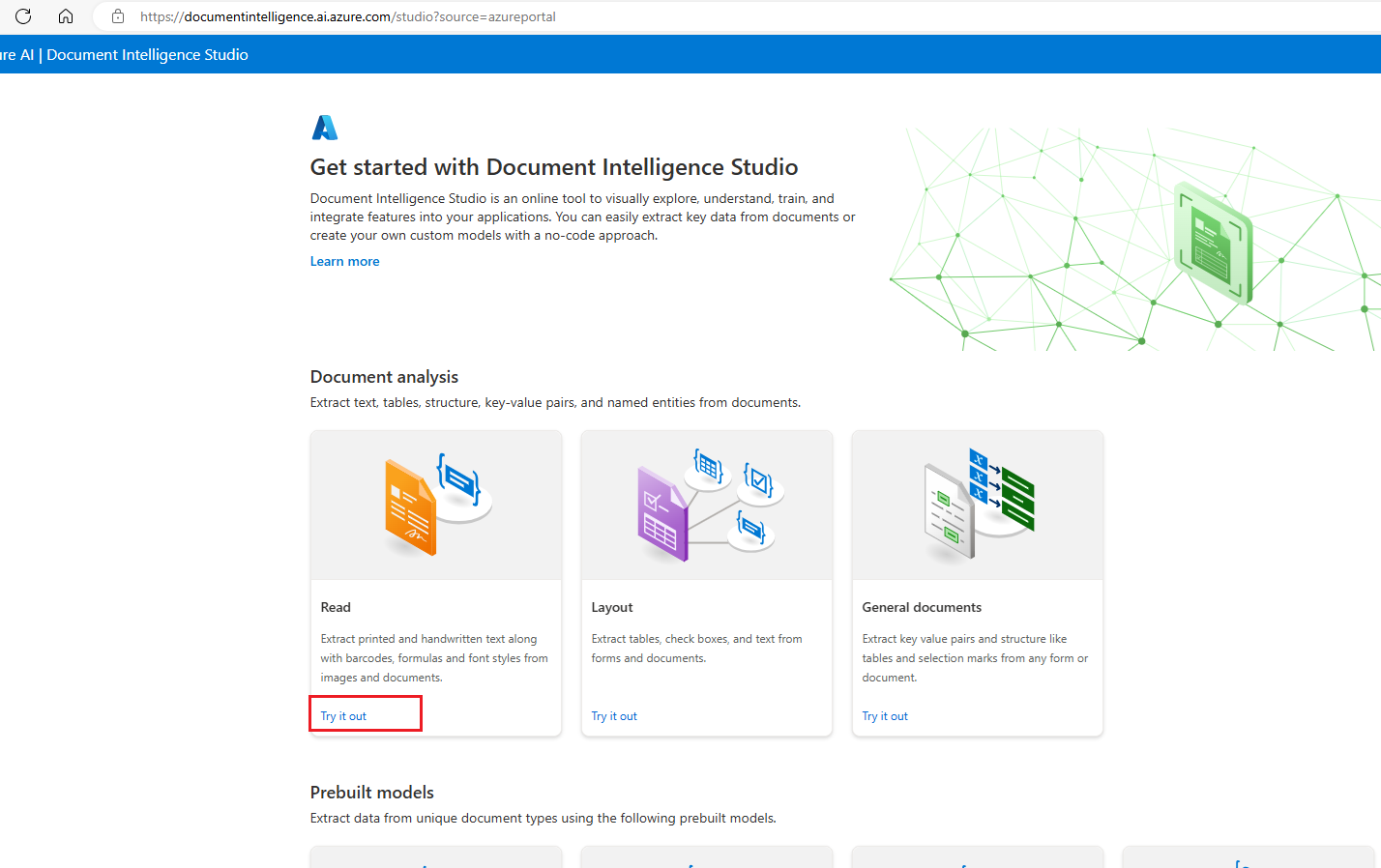
Let's start by using the **Azure AI Document Intelligence Studio** and the Read model to analyze a document with multiple languages. You'll connect Azure AI Document Intelligence Studio to the resource you just created to perform the analysis:

1. Open a new browser tab and go to the **Azure AI Document Intelligence Studio** at <https://documentintelligence.ai.azure.com/studio>. ( You can click the Try it from portal also

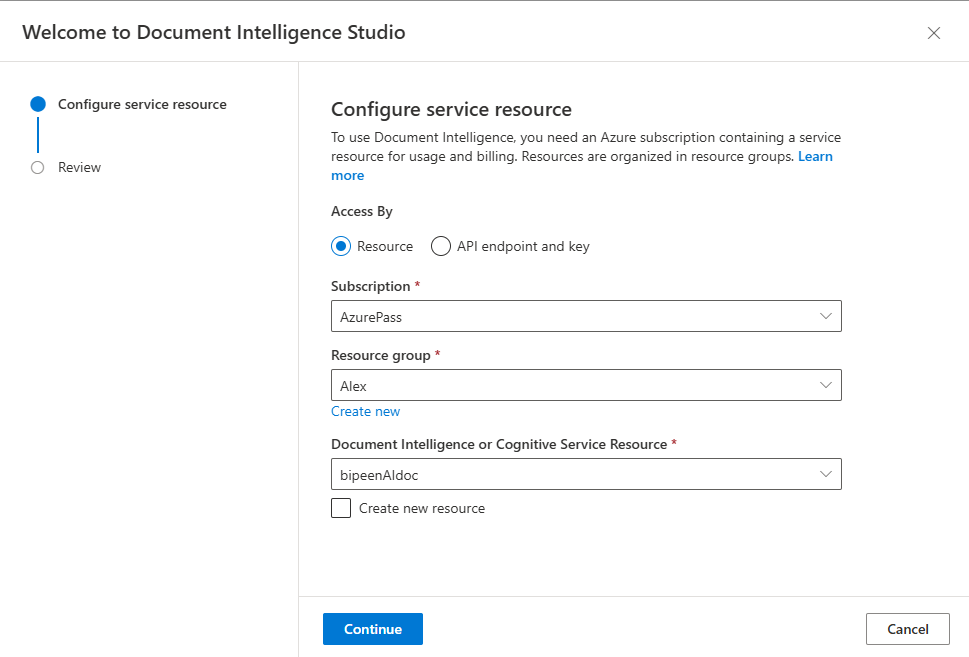




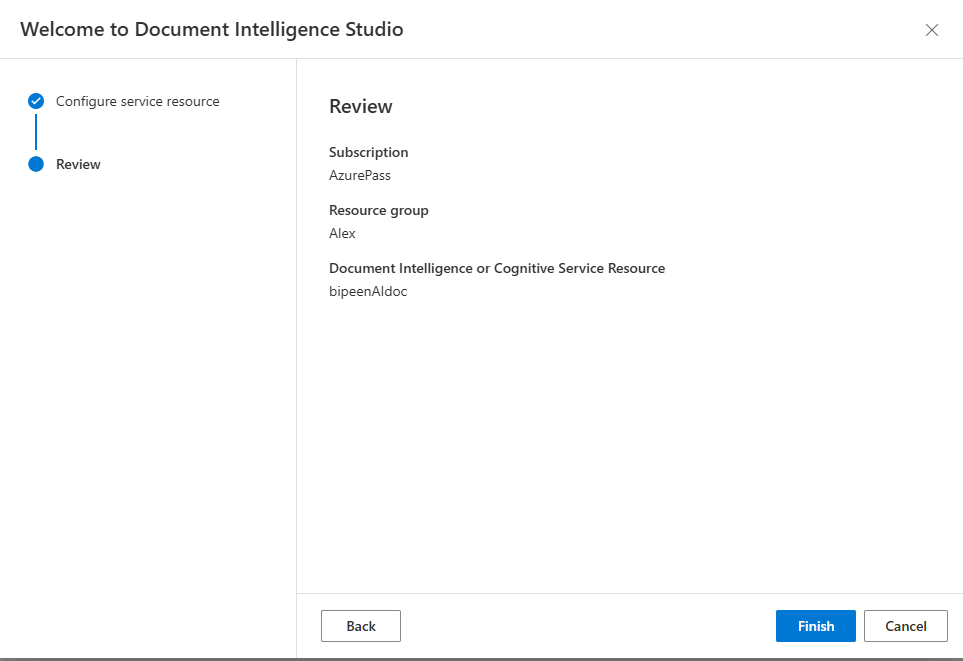
1. Under **Document Analysis**, select the **Read** tile.



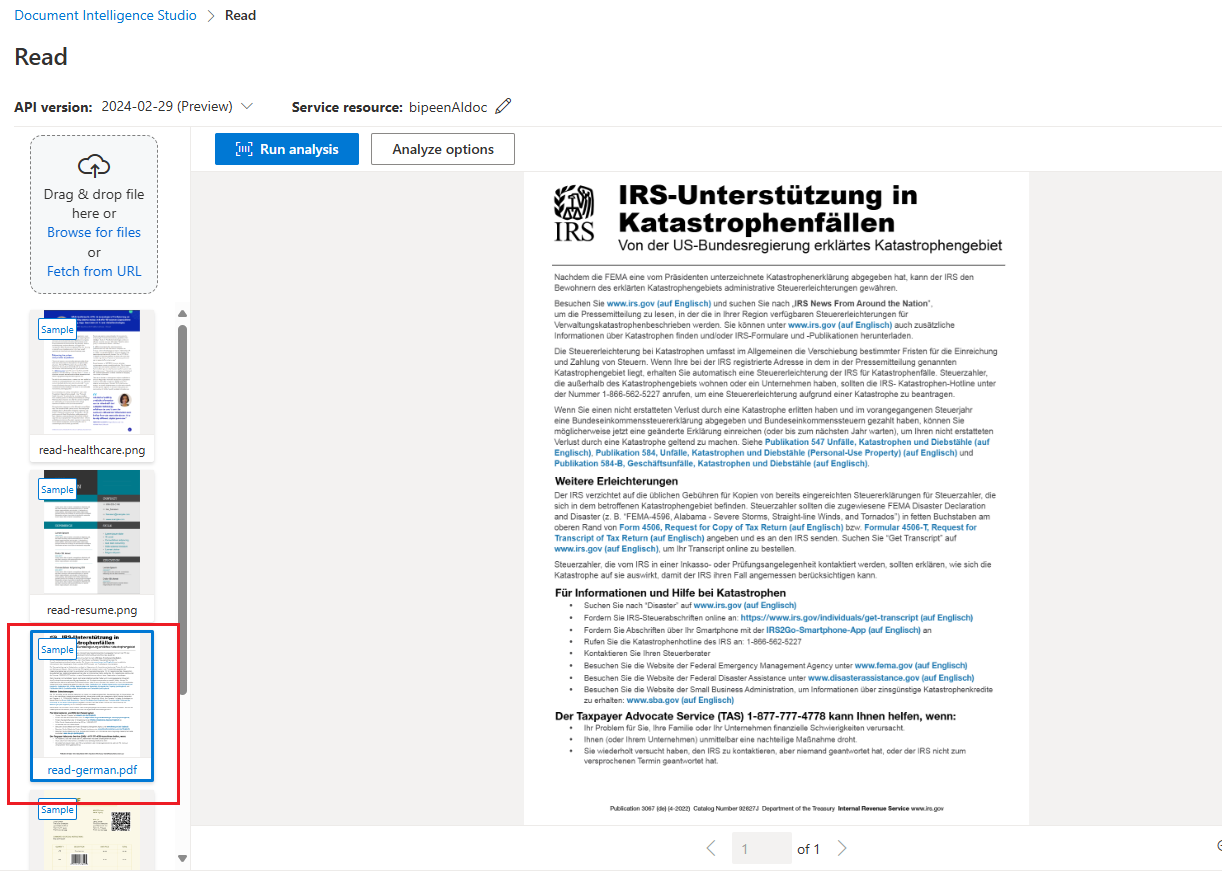
1. If you are asked to sign into your account, use your Azure credentials.
2. If you are asked which Azure AI Document Intelligence resource to use, select the subscription and resource name you used when you created the Azure AI Document Intelligence resource.



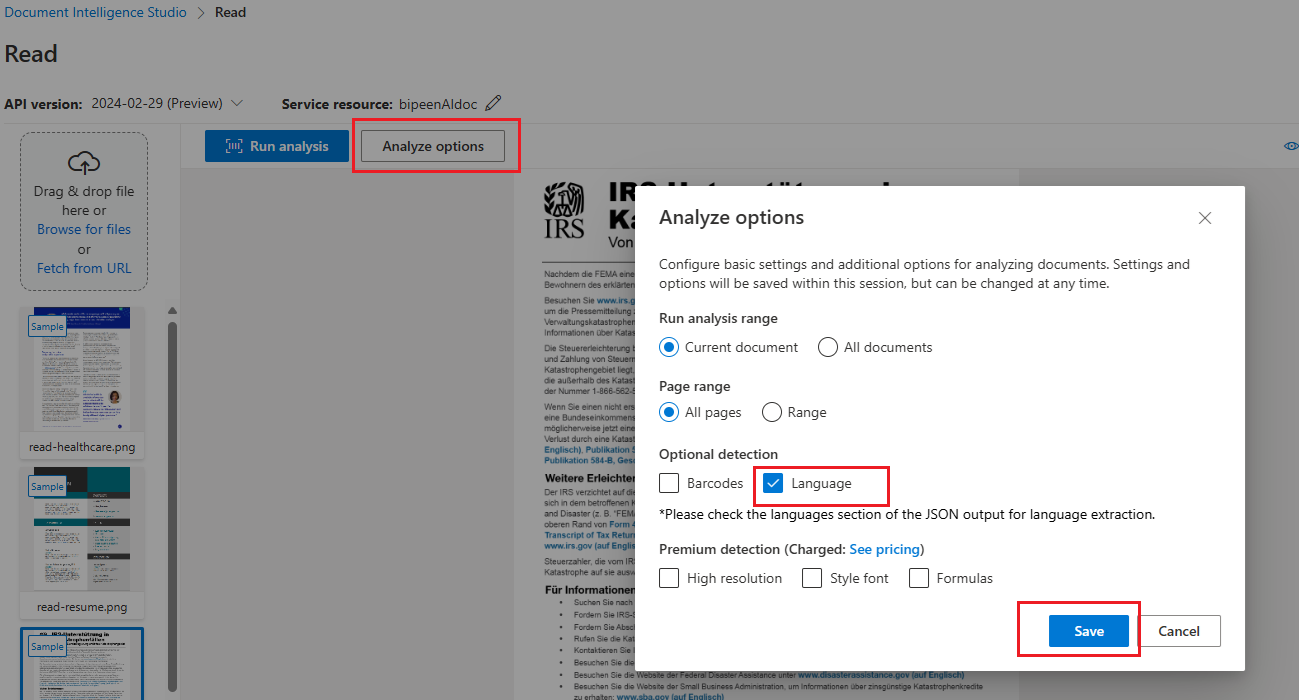
1. Click **Continue** button and Click **Finish** to Next Screen



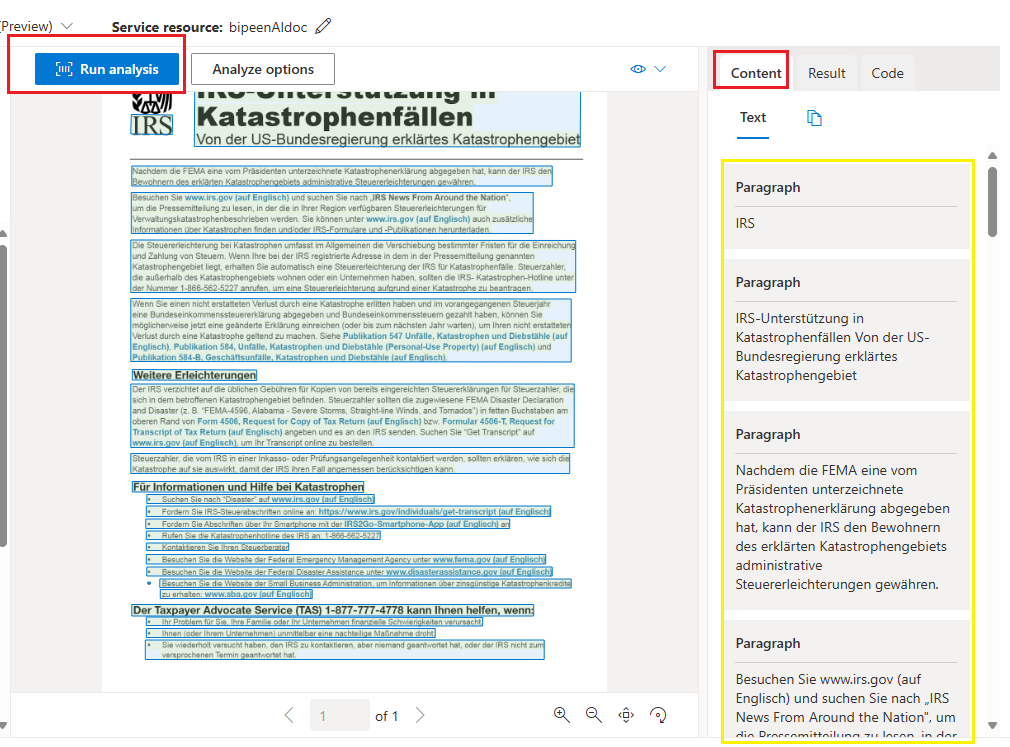
1. In the list of documents on the left, select **read-german.pdf**.



1. At the top-left, select **Analyze options**, then enable the **Language** check-box (under **Optional detection**) in the **Analyze options** pane and click on **Save**.



1. At the top-left, select **Run Analysis**.
2. When the analysis is complete, the text extracted from the image is shown on the right in the **Content** tab. Review this text and compare it to the text in the original image for accuracy.



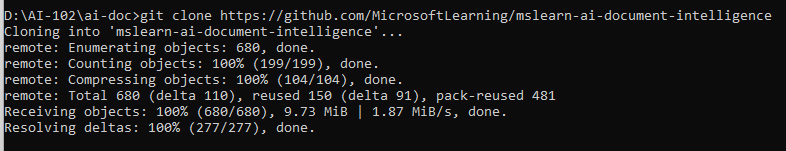
1. Select the **Result** tab. This tab displays the extracted JSON code.
2. Scroll to the bottom of the JSON code in the **Result** tab. Notice that the read model has detected the language of each span. Most spans are in German (language code de) but you can find other language codes in the spans (e.g. English - language code en - in one of the last span).



# Prepare to develop an app in Visual Studio Code

1. Open the command prompt and run a **Git: Clone** command to clone the https://github.com/MicrosoftLearning/mslearn-ai-document-intelligence repository to a local folder (it doesn't matter which folder).

**Git clone** [**https://github.com/MicrosoftLearning/mslearn-ai-document-intelligence**](https://github.com/MicrosoftLearning/mslearn-ai-document-intelligence)



1. When the repository has been cloned, go inside clone directory and launch VS Code

**cd mslearn-ai-document-intelligence**

**code .**

****

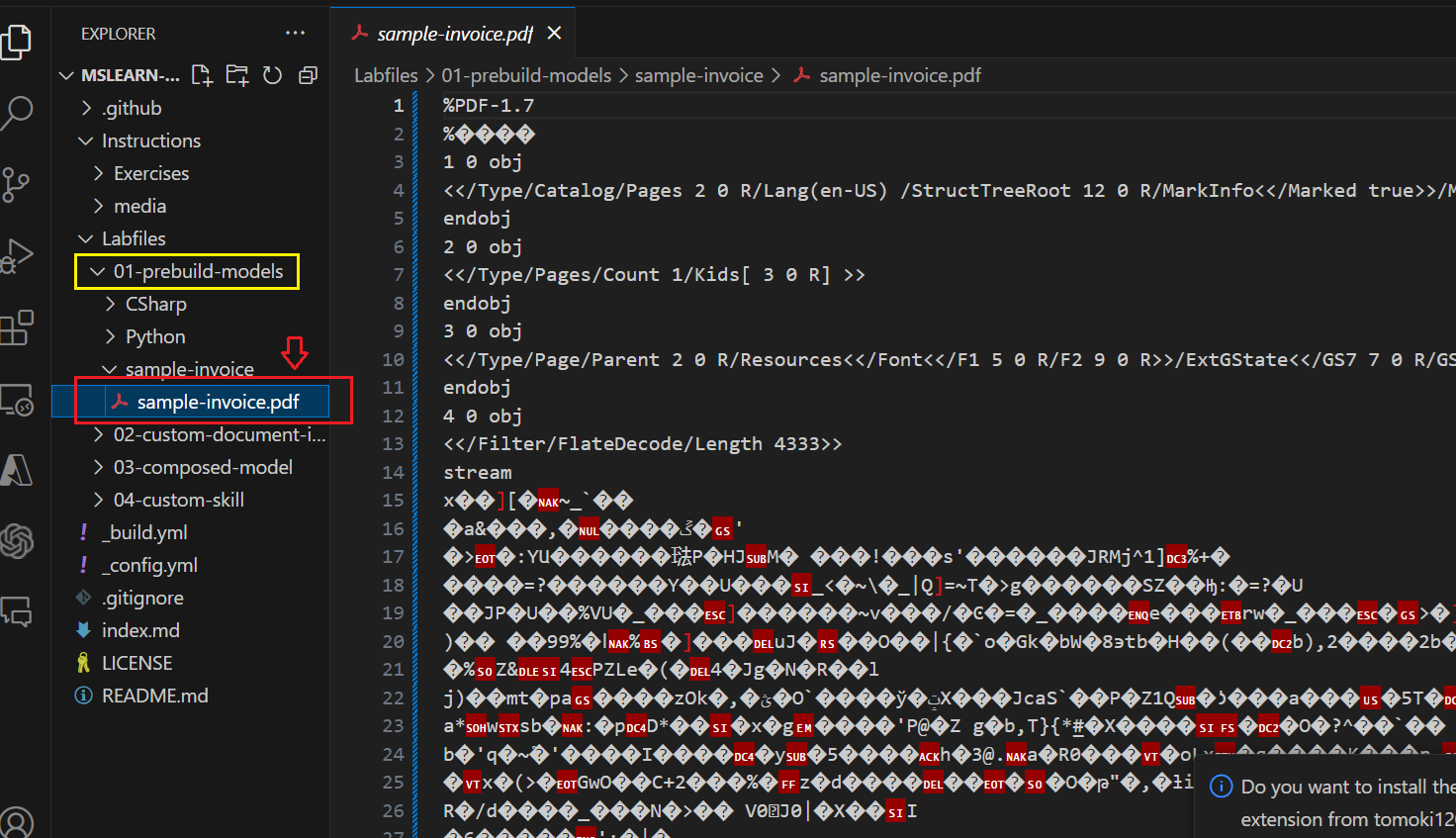
1. Wait while additional files are installed to support the C# code projects in the repo.

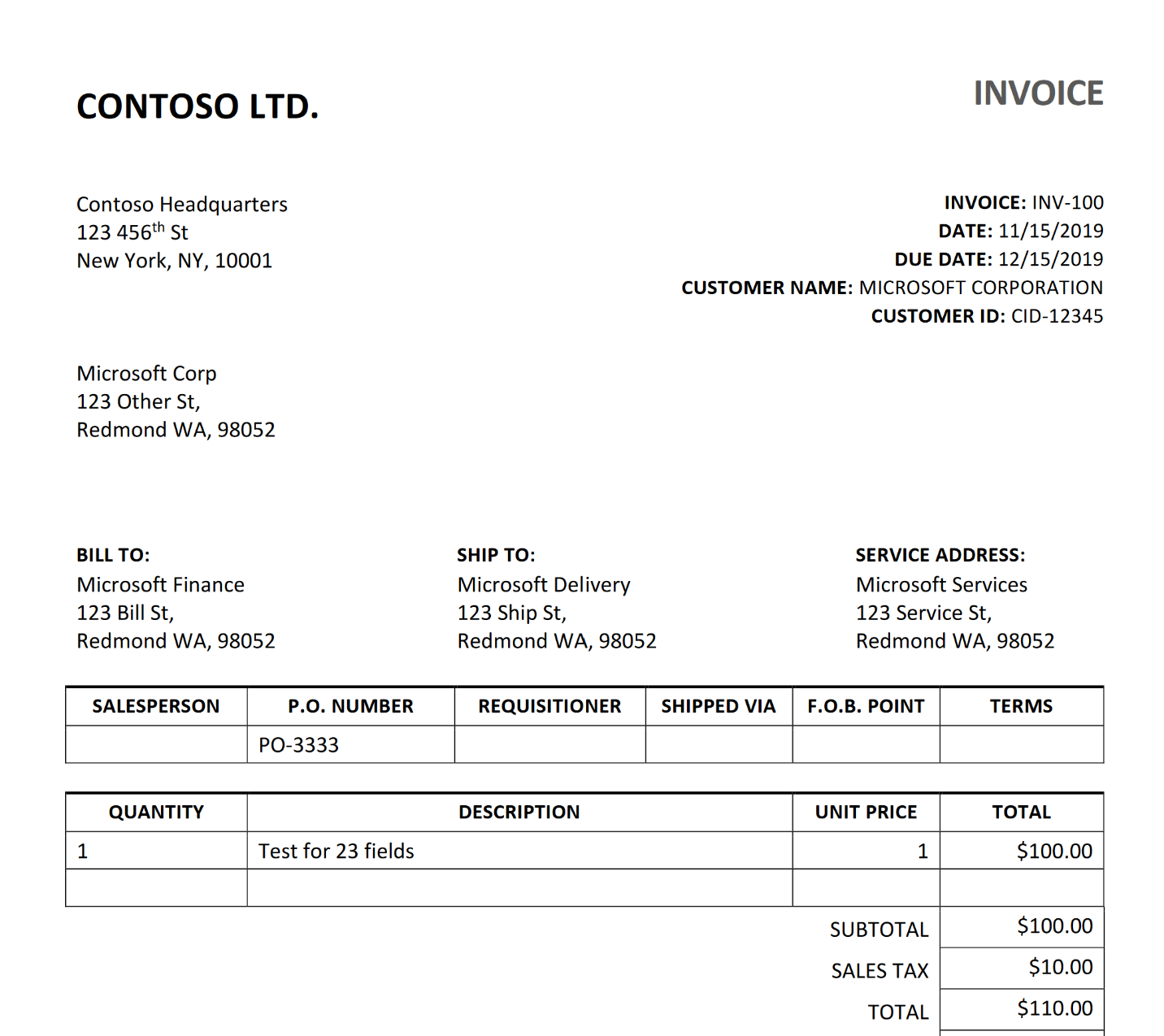
**Note**: If you are prompted to add required assets to build and debug, select **Not Now**. If you are prompted with the Message Detected an Azure Function Project in folder, you can safely close that message.

# Configure your application

Applications for C# have been provided, as well as a sample pdf file you'll use to test Document Intelligence. Both apps feature the same functionality. First, you'll complete some key parts of the application to enable using your Azure Document Intelligence resource.

1. Examine the following invoice located inside **Labfiles🡪 01 Prebuild Model** and note some of its fields and values. This is the invoice that your code will analyze.



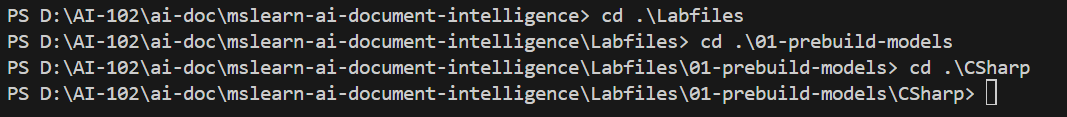


1. In Visual Studio Code, in the **Explorer** pane, browse to the **Labfiles/01-prebuild-models** folder and expand the **CSharp** . Each folder contains the language-specific files for an app into which you're you're going to integrate Azure Document Intelligence functionality.

**cd .\Labfiles**

**cd .\01-prebuild-models**

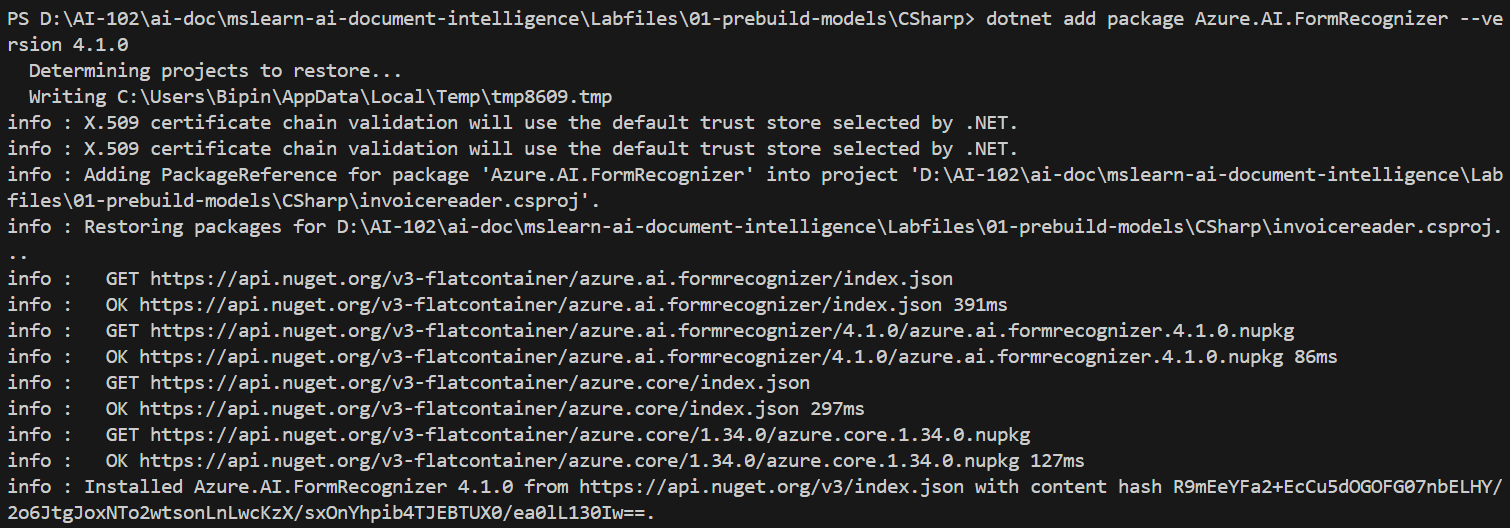
**cd .\CSharp**



1. Then install the **Azure Form Recognizer** (the previous name for Document Intelligence) SDK package by running the appropriate command for your language preference:

**C#**:

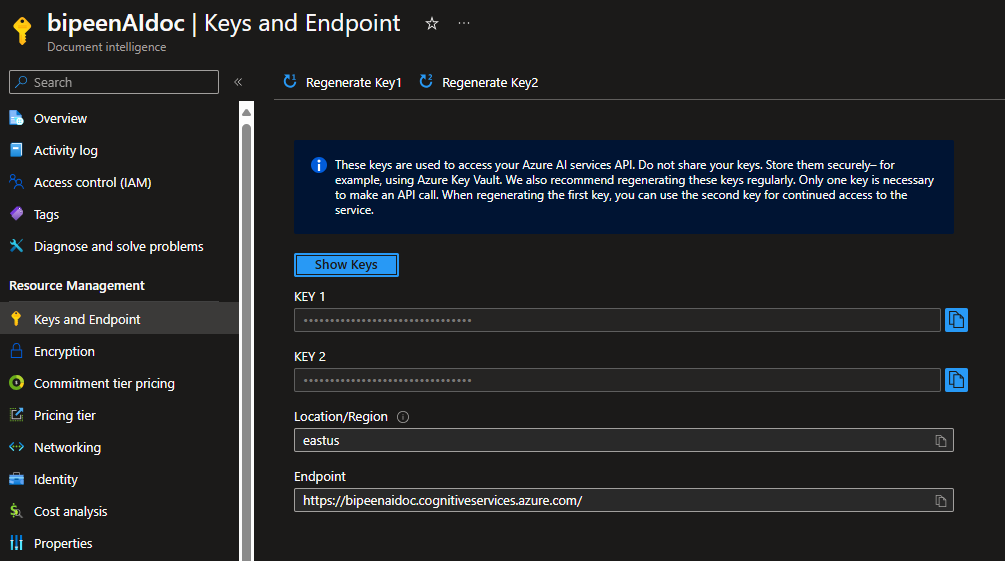
**dotnet add package Azure.AI.FormRecognizer --version 4.1.0**

****

# Add code to use the Azure Document Intelligence service

Now you're ready to use the SDK to evaluate the pdf file.

1. Switch to the browser tab that displays the Azure AI Document Intelligence overview in the Azure portal. On the left pane, under Resource Management, select **Keys and Endpoint**. To the right of the **Endpoint** value, click the **Copy to clipboard** button.

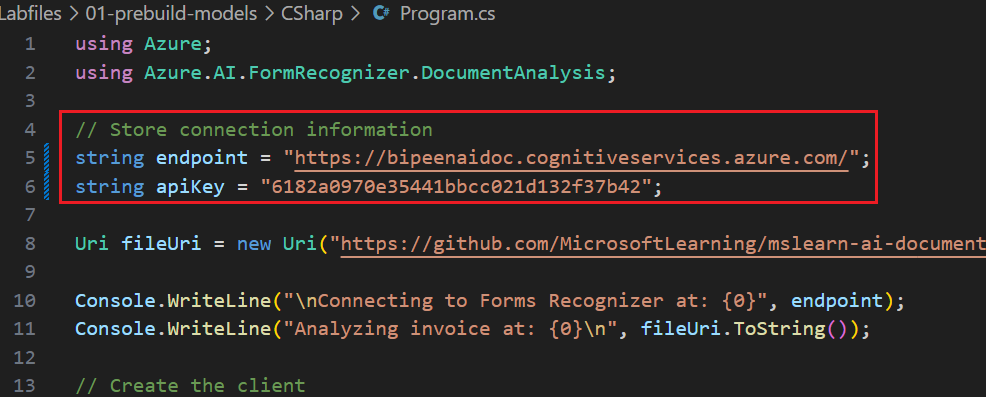


1. In the **Explorer** pane, in the **CSharp** or **Pythn** folder, open the code file for your preferred language, and replace <Endpoint URL>  and API Key with the string and Key you just copied:

**C#**: **Program.cs**

string endpoint = "<Endpoint URL>";

string apiKey = "<API Key>";

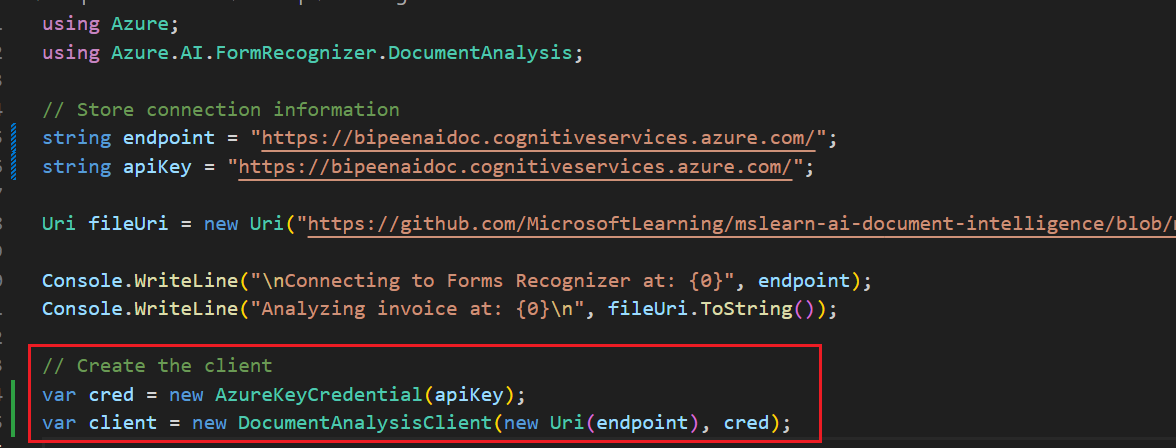


1. Locate the comment Create the client. Following that, on new lines, enter the following code:

**C#**

var cred = new AzureKeyCredential(apiKey);

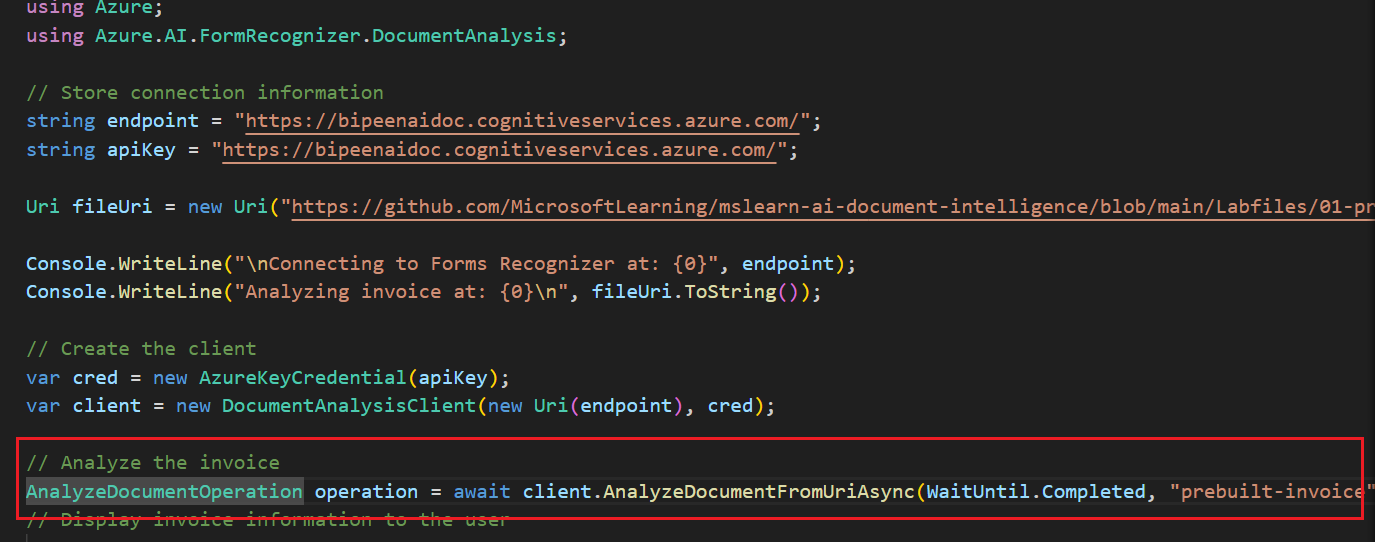
var client = new DocumentAnalysisClient(new Uri(endpoint), cred);



1. Locate the comment Analyze the invoice. Following that, on new lines, enter the following code:

**C#**

AnalyzeDocumentOperation operation = await client.AnalyzeDocumentFromUriAsync(WaitUntil.Completed, "prebuilt-invoice", fileUri);



1. Locate the comment Display invoice information to the user. Following that, on news lines, enter the following code:

AnalyzeResult result = operation.Value;

foreach (AnalyzedDocument invoice in result.Documents)

{

if (invoice.Fields.TryGetValue("VendorName", out DocumentField? vendorNameField))

{

if (vendorNameField.FieldType == DocumentFieldType.String)

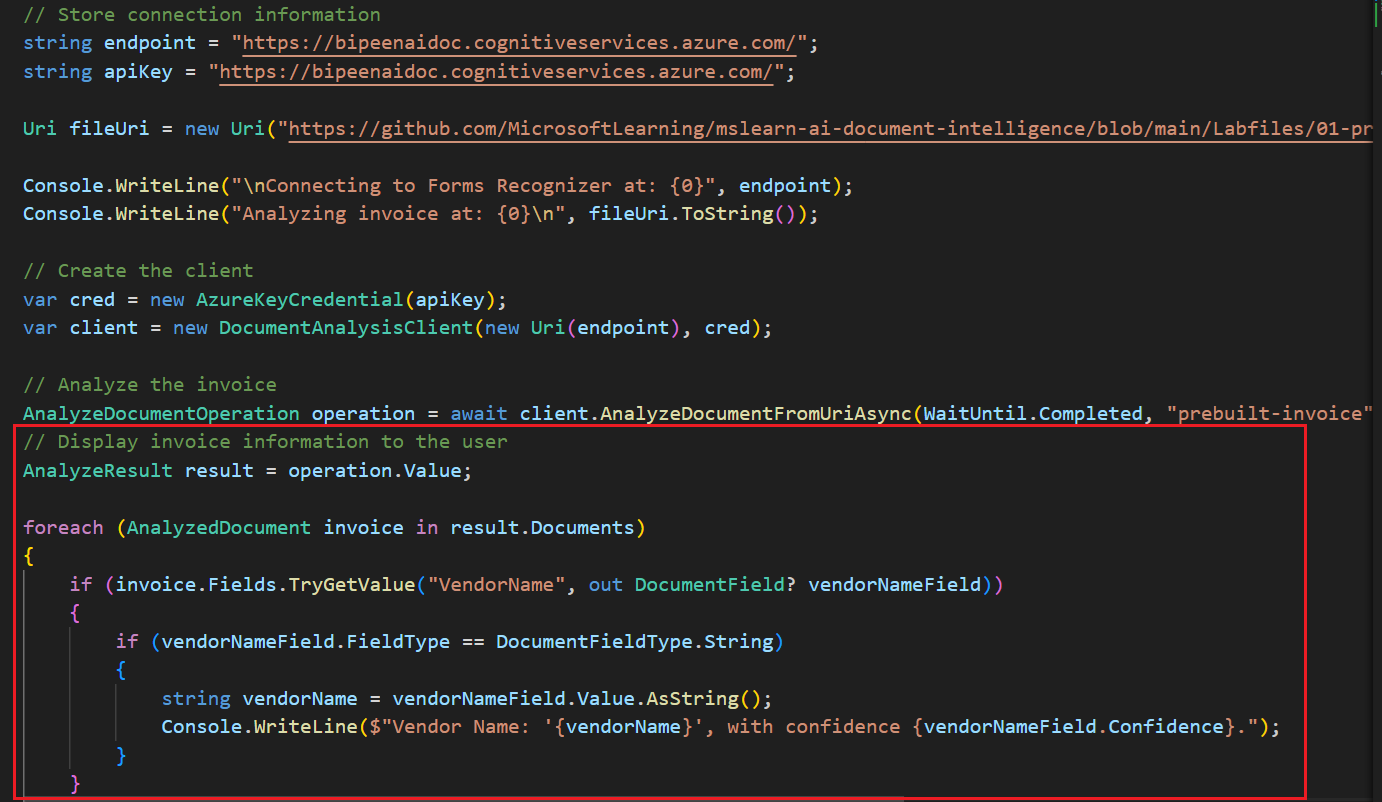
{

string vendorName = vendorNameField.Value.AsString();

Console.WriteLine($"Vendor Name: '{vendorName}', with confidence {vendorNameField.Confidence}.");

}

}



You've added code to display the vendor name. The starter project also includes code to display the customer name and invoice total.

**Complete Code :**

using Azure;

using Azure.AI.FormRecognizer.DocumentAnalysis;

// Store connection information

string endpoint = "https://bipeenaidoc.cognitiveservices.azure.com/";

string apiKey = "https://bipeenaidoc.cognitiveservices.azure.com/";

Uri fileUri = new Uri("https://github.com/MicrosoftLearning/mslearn-ai-document-intelligence/blob/main/Labfiles/01-prebuild-models/sample-invoice/sample-invoice.pdf?raw=true");

Console.WriteLine("\nConnecting to Forms Recognizer at: {0}", endpoint);

Console.WriteLine("Analyzing invoice at: {0}\n", fileUri.ToString());

// Create the client

var cred = new AzureKeyCredential(apiKey);

var client = new DocumentAnalysisClient(new Uri(endpoint), cred);

// Analyze the invoice

AnalyzeDocumentOperation operation = await client.AnalyzeDocumentFromUriAsync(WaitUntil.Completed, "prebuilt-invoice", fileUri);

// Display invoice information to the user

AnalyzeResult result = operation.Value;

foreach (AnalyzedDocument invoice in result.Documents)

{

    if (invoice.Fields.TryGetValue("VendorName", out DocumentField? vendorNameField))

    {

        if (vendorNameField.FieldType == DocumentFieldType.String)

        {

            string vendorName = vendorNameField.Value.AsString();

            Console.WriteLine($"Vendor Name: '{vendorName}', with confidence {vendorNameField.Confidence}.");

        }

    }

    if (invoice.Fields.TryGetValue("CustomerName", out DocumentField? customerNameField))

    {

        if (customerNameField.FieldType == DocumentFieldType.String)

        {

            string customerName = customerNameField.Value.AsString();

            Console.WriteLine($"Customer Name: '{customerName}', with confidence {customerNameField.Confidence}.");

        }

    }

    if (invoice.Fields.TryGetValue("InvoiceTotal", out DocumentField? invoiceTotalField))

    {

        if (invoiceTotalField.FieldType == DocumentFieldType.Currency)

        {

            CurrencyValue invoiceTotal = invoiceTotalField.Value.AsCurrency();

            Console.WriteLine($"Invoice Total: '{invoiceTotal.Symbol}{invoiceTotal.Amount}', with confidence {invoiceTotalField.Confidence}.");

        }

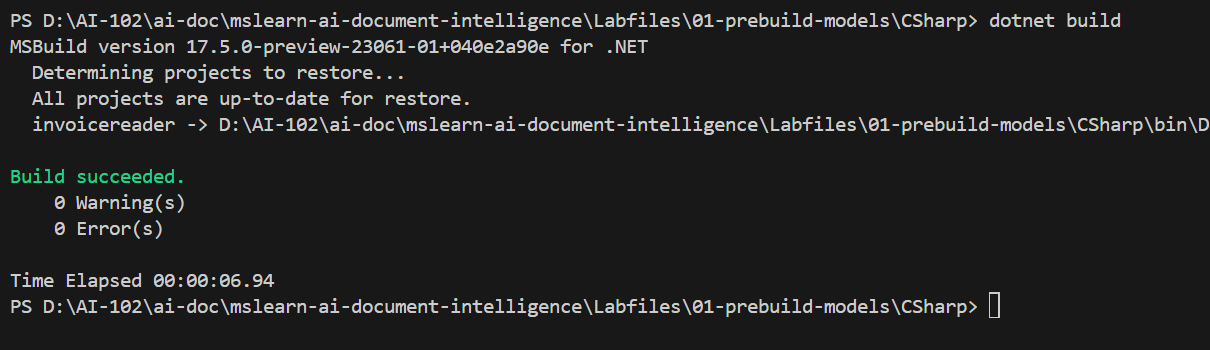
    }

}

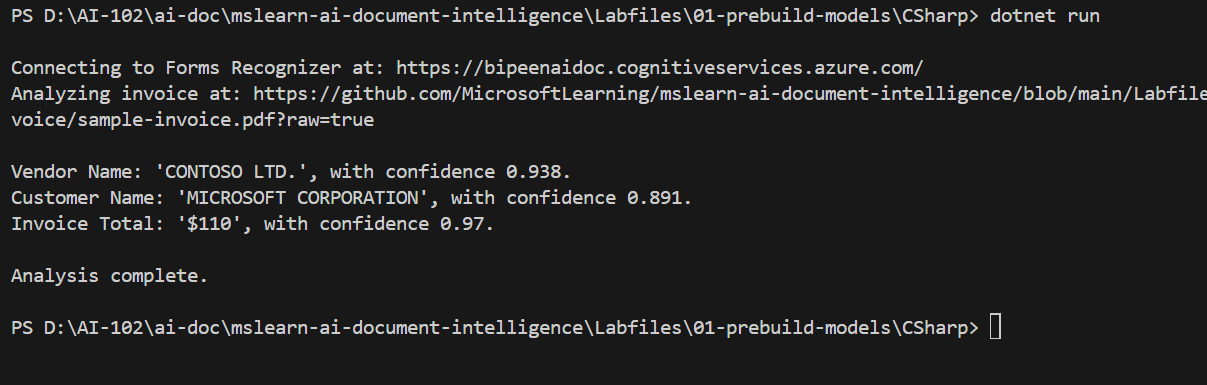
Console.WriteLine("\nAnalysis complete.\n");

1. Save the changes to the code file.
2. In the interactive terminal pane enter the following command to run the application.

**Dotnet build**



**Dotnet run**

****