Analyze forms with prebuilt Azure AI Document Intelligence models

In this exercise, you'll set up an Azure AI Foundry project with all the necessary resources for document analysis. You'll use both the Azure AI Foundry portal and the Python SDK to submit forms to that resource for analysis.

While this exercise is based on Python, you can develop similar applications using multiple language-specific SDKs; including:

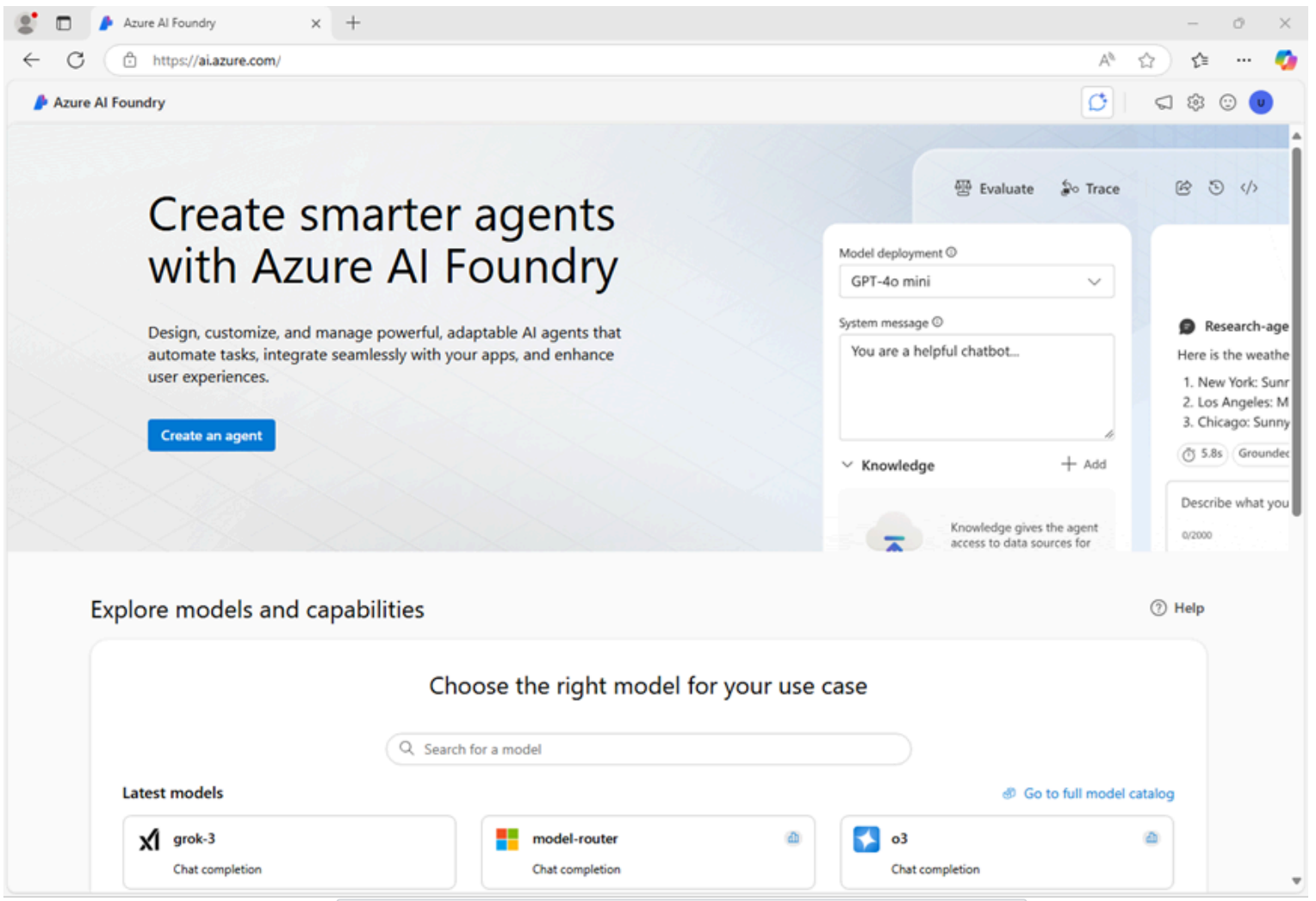
- Azure AI Document Intelligence client library for Python
- Azure AI Document Intelligence client library for Microsoft .NET
- Azure AI Document Intelligence client library for JavaScript

This exercise takes approximately **30** minutes.

Create an Azure AI Foundry project

Let's start by creating an Azure AI Foundry project.

1. In a web browser, open the Azure AI Foundry portal at `https://ai.azure.com` and sign in using your Azure credentials. Close any tips or quick start panes that are opened the first time you sign in, and if necessary use the **Azure AI Foundry** logo at the top left to navigate to the home page, which looks similar to the following image (close the **Help** pane if it's open):

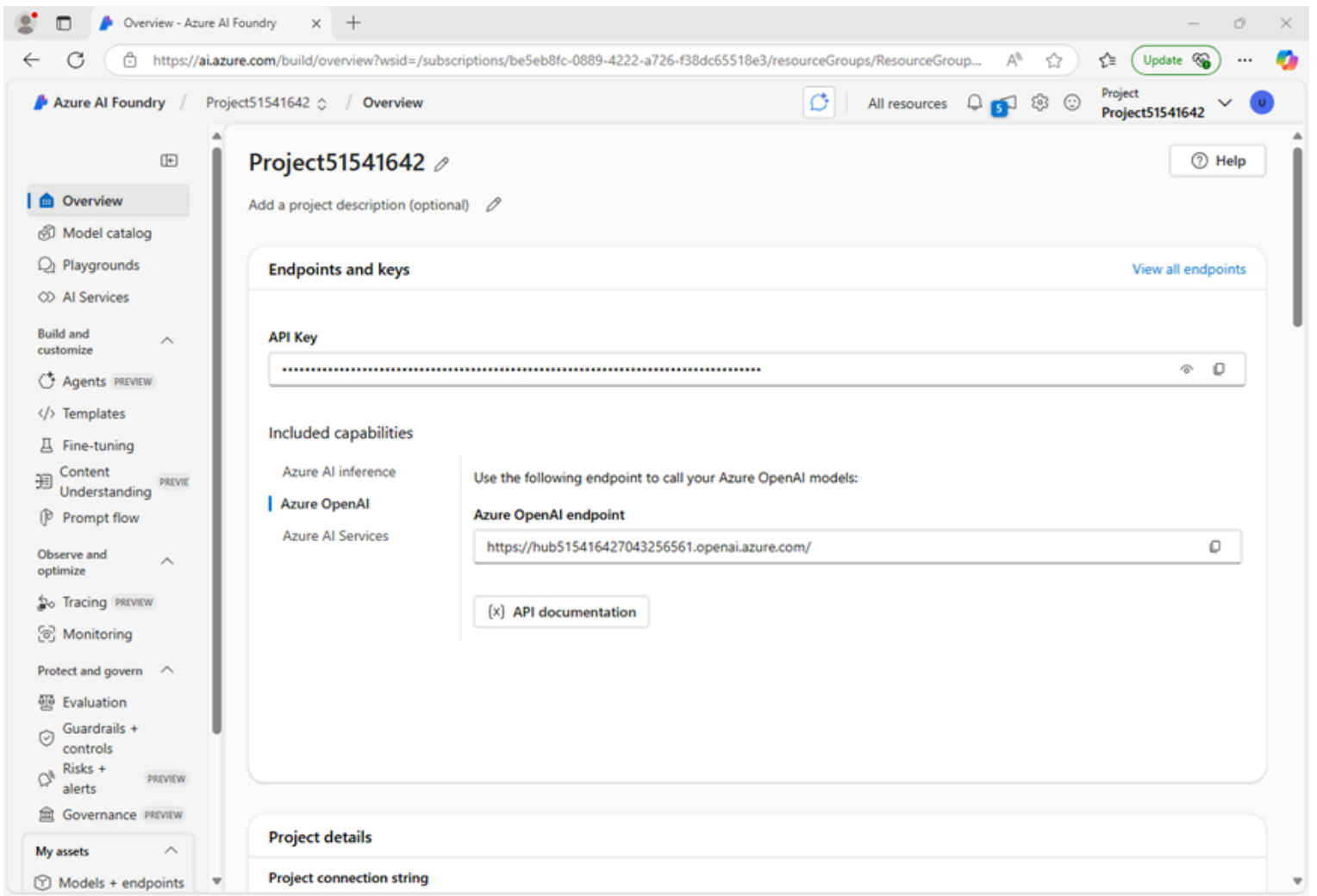


2. In the browser, navigate to `https://ai.azure.com/managementCenter/allResources` and select **Create new**. Then choose the option to create a new **AI hub resource**.
3. In the **Create a project** wizard, enter a valid name for your project, and select the option to create a new hub. Then use the **Rename hub** link to specify a valid name for your new hub, expand **Advanced options**, and specify the following settings for your project:
    - **Subscription**: *Your Azure subscription*
    - **Resource group**: *Create or select a resource group*
    - **Region**: *Any available region*

> **Note**: If you're working in an Azure subscription in which policies are used to restrict allowable resource names, you may need to use the link at the bottom of the **Create a new project** dialog box to create the hub using the Azure portal.

> **Tip**: If the **Create** button is still disabled, be sure to rename your hub to a unique alphanumeric value.

4. Wait for your project to be created.
5. When your project is created, close any tips that are displayed and review the project page in Azure AI Foundry portal, which should look similar to the following image:

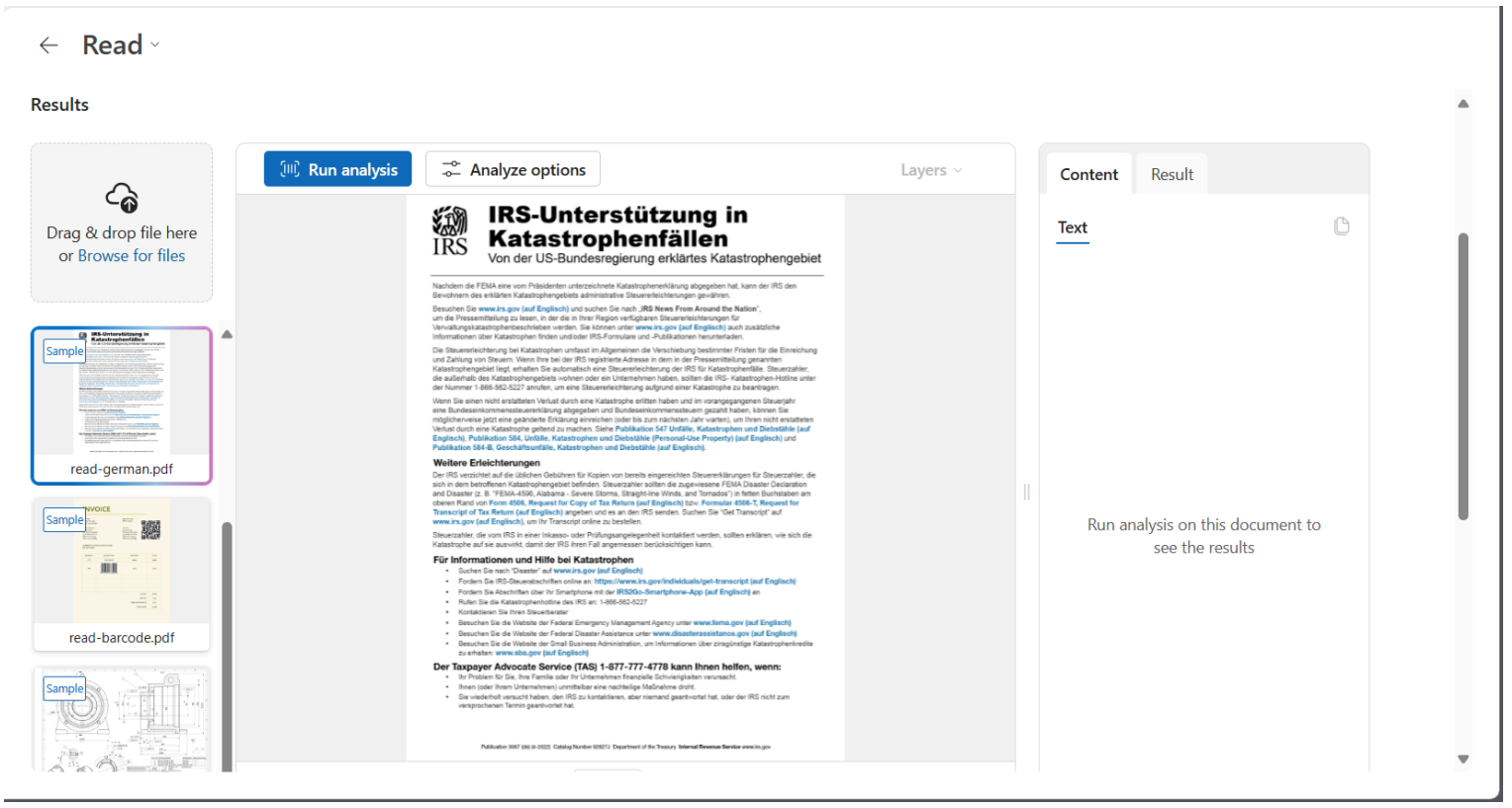


Use the Read model

Let's start by using the **Azure AI Foundry** portal and the Read model to analyze a document with multiple languages:

1. In the navigation panel on the left, select **AI Services**.
2. In the **Azure AI Services** page, select the **Vision + Document** tile.
3. In the **Vision + Document** page, verify that the **Document** tab is selected, then select the **OCR/Read** tile.

   In the **Read** page, the Azure AI Services resource created with your project should already be connected.
4. In the list of documents on the left, select **read-german.pdf**.



5. At the top toolbar, select **Analyze options**, then enable the **Language** check-box (under **Optional detection**) in the **Analyze options** pane and select **Save**.
6. At the top-left, select **Run Analysis**.
7. 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.
8. Select the **Result** tab. This tab displays the extracted JSON code.

Prepare to develop an app in Cloud Shell

Now let's explore the app that uses the Azure Document Intelligence service SDK. You'll develop your app using Cloud Shell. The code files for your app have been provided in a GitHub repo.

This is the invoice that your code will analyze.

CONTOSO LTD.

INVOICE

Contoso Headquarters
123 456th St
New York, NY, 10001

INVOICE: INV-100
DATE: 11/15/2019
DUE DATE: 12/15/2019
CUSTOMER NAME: MICROSOFT CORPORATION
CUSTOMER ID: CID-12345

Microsoft Corp
123 Other St,
Redmond WA, 98052

BILL TO:
Microsoft Finance
123 Bill St,
Redmond WA, 98052

SHIP TO:
Microsoft Delivery
123 Ship St,
Redmond WA, 98052

SERVICE ADDRESS:
Microsoft Services
123 Service St,
Redmond WA, 98052

SALESPERSON	P.O. NUMBER	REQUISITIONER	SHIPPED VIA	F.O.B. POINT	TERMS
	PO-3333				

QUANTITY	DESCRIPTION	UNIT PRICE	TOTAL
1	Test for 23 fields	1	$100.00
		SUBTOTAL	$100.00
		SALES TAX	$10.00
		TOTAL	$110.00

1. In the Azure AI Foundry portal, view the **Overview** page for your project.

2. In the **Endpoints and keys** area, select the **Azure AI Services** tab, and note the **API Key** and **Azure AI Services endpoint**. You'll use these credentials to connect to your Azure AI Services in a client application.
3. Open a new browser tab (keeping the Azure AI Foundry portal open in the existing tab). Then in the new tab, browse to the Azure portal at `https://portal.azure.com`; signing in with your Azure credentials if prompted.
4. Use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal.

   > **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

5. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.

6. In the PowerShell pane, enter the following commands to clone the GitHub repo for this exercise:

   Code Copy

   ```
   rm -r mslearn-ai-info -f
   git clone https://github.com/microsoftlearning/mslearn-ai-information-extraction mslearn-ai-info
   ```

   > **Tip**: As you paste commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

   Now follow the steps for your chosen programming language.

7. After the repo has been cloned, navigate to the folder containing the code files:

   Code Copy

   ```
   cd mslearn-ai-info/Labfiles/prebuilt-doc-intelligence/Python
   ```

8. In the cloud shell command line pane, enter the following command to install the libraries you'll use:

   Code Copy

   ```
   python -m venv labenv
   ./labenv/bin/Activate.ps1
   pip install -r requirements.txt azure-ai-formrecognizer==3.3.3
   ```

9. Enter the following command to edit the configuration file that has been provided:

   Code Copy

   ```
   code .env
   ```

   The file is opened in a code editor.
10. In the code file, replace the **YOUR_ENDPOINT** and **YOUR_KEY** placeholders with your Azure AI services endpoint and its API key (copied from the Azure AI Foundry portal).
11. After you've replaced the placeholders, within the code editor, use the **CTRL+S** command to save your changes and then use the **CTRL+Q** command to close the code editor while keeping the cloud shell command line open.

Add code to use the Azure Document Intelligence service

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

1. Enter the following command to edit the app file that has been provided:

```
code document-analysis.py
```

   The file is opened in a code editor.

2. In the code file, find the comment **Import the required libraries** and add the following code:

```
# Add references
from azure.core.credentials import AzureKeyCredential
from azure.ai.formrecognizer import DocumentAnalysisClient
```

3. Find the comment **Create the client** and add the following code (being careful to maintain the correct indentation level):

```
# Create the client
document_analysis_client = DocumentAnalysisClient(
     endpoint=endpoint, credential=AzureKeyCredential(key)
)
```

4. Find the comment **Analyze the invoice** and add the following code:

```
# Analyse the invoice
poller = document_analysis_client.begin_analyze_document_from_url(
     fileModelId, fileUri, locale=fileLocale
)
```

5. Find the comment **Display invoice information to the user**and add the following code:

```
# Display invoice information to the user
receipts = poller.result()

for idx, receipt in enumerate(receipts.documents):

     vendor_name = receipt.fields.get("VendorName")
     if vendor_name:
         print(f"\nVendor Name: {vendor_name.value}, with confidence
{vendor_name.confidence}.")

     customer_name = receipt.fields.get("CustomerName")
     if customer_name:
         print(f"Customer Name: '{customer_name.value}, with confidence
{customer_name.confidence}.")


     invoice_total = receipt.fields.get("InvoiceTotal")
     if invoice_total:
         print(f"Invoice Total: '{invoice_total.value.symbol}{invoice_total.value.amount},
 with confidence {invoice_total.confidence}.")
```

6. In the code editor, use the **CTRL+S** command or **Right-click > Save** to save your changes. Keep the code editor open in case you need to fix any errors in the code, but resize the panes so you can see the command line pane clearly.

7. In the command line pane, enter the following command to run the application.

Code Copy

```
python document-analysis.py
```

The program displays the vendor name, customer name, and invoice total with confidence levels. Compare the values it reports with the sample invoice you opened at the start of this section.

Clean up

If you’re done with your Azure resource, remember to delete the resource in the Azure portal (`https://portal.azure.com`) to avoid further charges.