

Create an knowledge mining solution

In this exercise, you use AI Search to index a set of documents maintained by Margie’s Travel, a fictional travel agency. The indexing process involves using AI skills to extract key information to make them searchable, and generating a knowledge store containing data assets for further analysis.

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

- [Azure AI Search client library for Python](#)
- [Azure AI Search client library for Microsoft .NET](#)
- [Azure AI Search client library for JavaScript](#)

This exercise takes approximately **40** minutes.

Create Azure resources

The solution you will create for Margie’s Travel requires multiple resources in your Azure subscription. In this exercise, you’ll create them directly in the Azure portal. You could also create them by using a script, or an ARM or BICEP template; or you could create an Azure AI Foundry project that includes an Azure AI Search resource.

Important: Your Azure resources should be created in the same location!

Create an Azure AI Search resource

1. In a web browser, open the [Azure portal](#) at `https://portal.azure.com`, and sign in using your Azure credentials.
2. Select the **+ Create a resource** button, search for `Azure AI Search`, and create an **Azure AI Search** resource with the following settings:
 - **Subscription:** *Your Azure subscription*
 - **Resource group:** *Create or select a resource group*
 - **Service name:** *A valid name for your search resource*
 - **Location:** *Any available location*
 - **Pricing tier:** Free
3. Wait for deployment to complete, and then go to the deployed resource.
4. Review the **Overview** page on the blade for your Azure AI Search resource in the Azure portal. Here, you can use a visual interface to create, test, manage, and monitor the various components of a search solution; including data sources, indexes, indexers, and skillsets.

Create a storage account

1. Return to the home page, and then create a **Storage account** resource with the following settings:
 - **Subscription:** *Your Azure subscription*
 - **Resource group:** *The same resource group as your Azure AI Search and Azure AI Services resources*
 - **Storage account name:** *A valid name for your storage resource*
 - **Region:** *The same region as your Azure AI Search resource*
 - **Primary service:** Azure Blob Storage or Azure Data Lake Storage Gen 2
 - **Performance:** Standard
 - **Redundancy:** Locally-redundant storage (LRS)
2. Wait for deployment to complete, and then go to the deployed resource.

Tip: Keep the storage account portal page open - you will use it in the next procedure.

[Create Azure resources](#)

[Upload documents to Azure Storage](#)

[Create and run an indexer](#)

[Search the index](#)

[Create a search client application](#)

[View the knowledge store](#)

[Clean-up](#)

[More information](#)

Upload documents to Azure Storage

Your knowledge mining solution will extract information from travel brochure documents in an Azure Storage blob container.

1. In a new browser tab, download [documents.zip](https://github.com/microsoftlearning/mslearn-ai-information-extraction/raw/main/Labfiles/knowledge/documents.zip) from

`https://github.com/microsoftlearning/mslearn-ai-information-extraction/raw/main/Labfiles/knowledge/documents.zip`

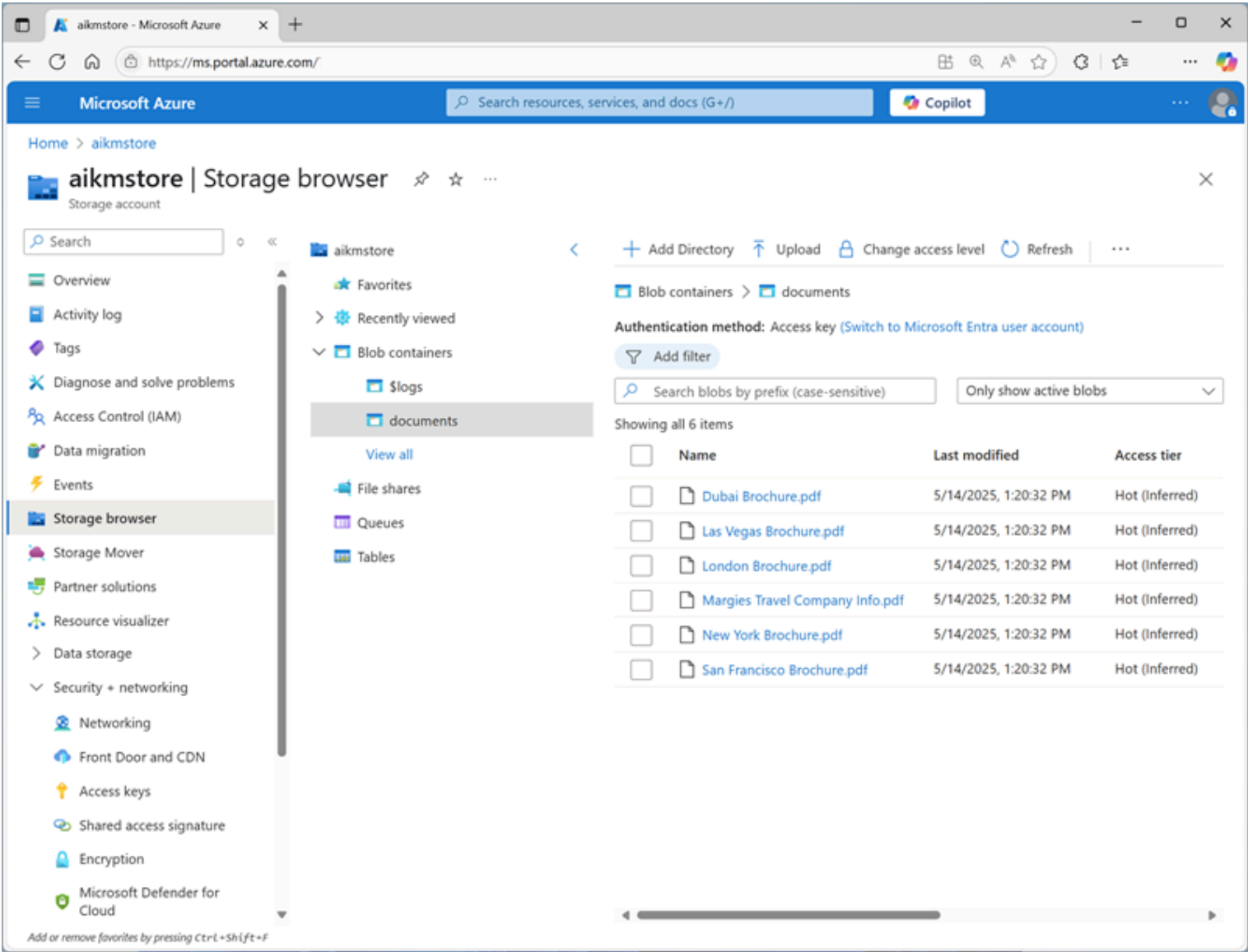
 and save it in a local folder.
2. Extract the downloaded *documents.zip* file and view the travel brochure files it contains. You'll extract and index information from these files.
3. In the browser tab containing the Azure portal page for your storage account, in the navigation pane on the left, select **Storage browser**.
4. In the storage browser, select **Blob containers**.

Currently, your storage account should contain only the default **\$logs** container.

5. In the toolbar, select **+ Container** and create a new container with the following settings:
 - o **Name:** `documents`
 - o **Anonymous access level:** Private (no anonymous access)*

Note: *Unless you enabled the option to allow anonymous container access when creating your storage account, you won't be able to select any other setting!

6. Select the **documents** container to open it, and then use the **Upload** toolbar button to upload the .pdf files you extracted from **documents.zip** previously into the root of the container, as shown here:



Create and run an indexer


Now that you have the documents in place, you can create an indexer to extract information from them.

1. In the Azure portal, browse to your Azure AI Search resource. Then, on its **Overview** page, select **Import data**.
2. On the **Connect to your data** page, in the **Data Source** list, select **Azure Blob Storage**. Then complete the data store details with the following values:

- **Data Source:** Azure Blob Storage
- **Data source name:** `margies-documents`
- **Data to extract:** Content and metadata
- **Parsing mode:** Default
- **Subscription:** *Your Azure subscription*
- **Connection string:**
 - Select **Choose an existing connection**
 - Select your storage account
 - Select the **documents** container
- **Managed identity authentication:** None
- **Container name:** documents
- **Blob folder:** *Leave this blank*
- **Description:** `Travel brochures`

3. Proceed to the next step (**Add cognitive skills**), which has three expandable sections to complete.

4. In the **Attach Azure AI Services** section, select **Free (limited enrichments)***.

 **Note:** *The free Azure AI Services resource for Azure AI Search can be used to index a maximum of 20 documents. In a real solution, you should create an Azure AI Services resource in your subscription to enable AI enrichment for a larger number of documents.

5. In the **Add enrichments** section:

- Change the **Skillset name** to `margies-skillset`.
- Select the option **Enable OCR and merge all text into merged_content field**.
- Ensure that the **Source data field** is set to **merged_content**.
- Leave the **Enrichment granularity level** as **Source field**, which is set the entire contents of the document being indexed; but note that you can change this to extract information at more granular levels, like pages or sentences.
- Select the following enriched fields:

Cognitive Skill	Parameter	Field name
Text Cognitive Skills		
Extract people names		people
Extract location names		locations
Extract key phrases		keyphrases
Image Cognitive Skills		
Generate tags from images		imageTags
Generate captions from images		imageCaption

Double-check your selections (it can be difficult to change them later).

6. In the **Save enrichments to a knowledge store** section:

- Select only the following checkboxes (an **error** will be displayed, you'll resolve that shortly):
 - **Azure file projections:**
 - Image projections
 - **Azure table projections:**
 - Documents
 - Key phrases
 - **Azure blob projections:**

- Document
- Under **Storage account connection string** (beneath the **error messages**):
 - Select **Choose an existing connection**
 - Select your storage account
 - Select the **documents** container (*this is only required to select the storage account in the browse interface - you'll specify a different container name for the extracted knowledge assets!*)
 - Change the **Container name** to `knowledge-store`.
 - Proceed to the next step (**Customize target index**), where you'll specify the fields for your index.
 - Change the **Index name** to `margies-index`.
 - Ensure that the **Key** is set to `metadata_storage_path`, leave the **Suggester name** blank, and ensure **Search mode** is **analyzingInfixMatching**.
 - Make the following changes to the index fields, leaving all other fields with their default settings (**IMPORTANT**: you may need to scroll to the right to see the entire table):

Field name	Retrievable	Filterable	Sortable	Facetable	Searchable
metadata_storage_size	✓	✓	✓		
metadata_storage_last_modified	✓	✓	✓		
metadata_storage_name	✓	✓	✓		✓
locations	✓	✓			✓
people	✓	✓			✓
keyphrases	✓	✓			✓

- Double-check your selections, paying particular attention to ensure that the correct **Retrievable**, **Filterable**, **Sortable**, **Facetable**, and **Searchable** options are selected correctly for each field (it can be difficult to change them later).
- Proceed to the next step (**Create an indexer**), where you'll create and schedule the indexer.
 - Change the **Indexer name** to `margies-indexer`.
 - Leave the **Schedule** set to **Once**.
 - Select **Submit** to create the data source, skillset, index, and indexer. The indexer is run automatically and runs the indexing pipeline, which:
 - Extracts the document metadata fields and content from the data source
 - Runs the skillset of cognitive skills to generate additional enriched fields
 - Maps the extracted fields to the index.
 - Saves the extracted data assets to the knowledge store.
 - In the navigation pane on the left, under **Search management** view the **Indexers** page, which should show the newly created **margies-indexer**. Wait a few minutes, and click **↻ Refresh** until the **Status** indicates **Success**.

Search the index

Now that you have an index, you can search it.

- Return to the **Overview** page for your Azure AI Search resource, and on the toolbar, select **Search explorer**.
- In Search explorer, in the **Query string** box, enter `*` (a single asterisk), and then select **Search**.

This query retrieves all documents in the index in JSON format. Examine the results and note the fields for each document, which contain document content, metadata, and enriched data extracted by the cognitive skills you selected.

- In the **View** menu, select **JSON view** and note that the JSON request for the search is shown, like this:

CodeCopy

```
{
  "search": "*",
  "count": true
}
```

4. The results include a **@odata.count** field at the top of the results that indicates the number of documents returned by the search.
5. Modify the JSON request to include the **select** parameter as shown here:

CodeCopy

```
{
  "search": "*",
  "count": true,
  "select": "metadata_storage_name,locations"
}
```

This time the results include only the file name and any locations mentioned in the document content. The file name is in the **metadata_storage_name** field, which was extracted from the source document. The **locations** field was generated by an AI skill.

6. Now try the following query string:

CodeCopy

```
{
  "search": "New York",
  "count": true,
  "select": "metadata_storage_name,keyphrases"
}
```

This search finds documents that mention “New York” in any of the searchable fields, and returns the file name and key phrases in the document.

7. Let’s try one more query:

CodeCopy

```
{
  "search": "New York",
  "count": true,
  "select": "metadata_storage_name,keyphrases",
  "filter": "metadata_storage_size lt 380000"
}
```

This query returns the filename and key phrases for any documents mentioning “New York” that are smaller than 380,000 bytes in size.

Create a search client application

Now that you have a useful index, you can use it from a client application. You can do this by consuming the REST interface, submitting requests and receiving responses in JSON format over HTTP; or you can use the software development kit (SDK) for your preferred programming language. In this exercise, we’ll use the SDK.

Note: You can choose to use the SDK for either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

Get the endpoint and keys for your search resource

1. In the Azure portal, close the search explorer page and return to the **Overview** page for your Azure AI Search resource.

Note the **Url** value, which should be similar to **https://your_resource_name.search.windows.net**. This is the endpoint for your search resource.

2. In the navigation pane on the left, expand **Settings** and view the **Keys** page.

Note that there are two **admin** keys, and a single **query** key. An *admin* key is used to create and manage search resources; a *query* key is used by client applications that only need to perform search queries.

*You will need the endpoint and **query** key for your client application.*

Prepare to use the Azure AI Search SDK

1. Use the [**>**] button to the right of the search bar at the top of the Azure portal to create a new Cloud Shell in the Azure portal, selecting a **PowerShell** environment with no storage in your subscription.

The cloud shell provides a command-line interface in a pane at the bottom of the Azure portal. You can resize or maximize this pane to make it easier to work in. Initially, you'll need to see both the cloud shell and the Azure portal (so you can find and copy the endpoint and key you'll need).

2. 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.

3. In the cloud shell pane, enter the following commands to clone the GitHub repo containing the code files for this exercise (type the command, or copy it to the clipboard and then right-click in the command line and paste as plain text):

CodeCopy

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

Tip: As you enter commands into the cloudshell, the output 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.

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

CodeCopy


```
cd mslearn-ai-info/Labfiles/knowledge/python
ls -a -l
```

5. Install the Azure AI Search SDK and Azure identity packages by running the following commands:

CodeCopy


```
python -m venv labenv
./labenv/bin/Activate.ps1
pip install -r requirements.txt azure-identity azure-search-documents==11.5.1
```

6. Run the following command to edit the configuration file for your app:

Code  Copy


```
code .env
```

The configuration file is opened in a code editor.


7. Edit the configuration file to replace the following placeholder values:

- **your_search_endpoint** (replace with the endpoint for your Azure AI Search resource)
- **your_query_key** (replace with the query key for your Azure AI Search resource)
- **your_index_name** (replace with the name of your index, which should be `margies-index`)

8. When you’ve updated the placeholders, use the **CTRL+S** command to save the file and then use the **CTRL+Q** command to close it.

 **Tip:** Now that you’ve copied the endpoint and key from the Azure portal, you might want to maximize the cloud shell pane to make it easier to work in.

9. Run the following command to open the code file for your app:

Code  Copy

```
code search-app.py
```


The code file is opened in a code editor.

10. Review the code, and note that it performs the following actions:

- Retrieves the configuration settings for your Azure AI Search resource and index from the configuration file you edited.
- Creates a **SearchClient** with the endpoint, key, and index name to connect to your search service.
- Prompts the user for a search query (until they enter “quit”)
- Searches the index using the query, returning the following fields (ordered by metadata_storage_name):
 - metadata_storage_name
 - locations
 - people
 - keyphrases
- Parses the search results that are returned to display the fields returned for each document in the result set.

11. Close the code editor pane (**CTRL+Q**), keeping the cloud shell command line console pane open

12. Enter the following command to run the app:

Code  Copy

```
python search-app.py
```

13. When prompted, enter a query such as `London` and view the results.

14. Try another query, such as `flights`.

15. When you’re finished testing the app, enter `quit` to close it.

16. Close the Cloud shell, returning to the Azure portal.

View the knowledge store

After you have run an indexer that uses a skillset to create a knowledge store, the enriched data extracted by the indexing process is persisted in the knowledge store projections.

View object projections

The *object* projections defined in the Margie’s Travel skillset consist of a JSON file for each indexed document. These files are stored in a blob container in the Azure Storage account specified in the skillset definition.

1. In the Azure portal, view the Azure Storage account you created previously.
2. Select the **Storage browser** tab (in the pane on the left) to view the storage account in the storage explorer interface in the Azure portal.
3. Expand **Blob containers** to view the containers in the storage account. In addition to the **documents** container where the source data is stored, there should be two new containers: **knowledge-store** and **margies-skillset-image-projection**. These were created by the indexing process.
4. Select the **knowledge-store** container. It should contain a folder for each indexed document.
5. Open any of the folders, and then select the **objectprojection.json** file it contains and use the **Download** button on the toolbar to download and open it. Each JSON file contains a representation of an indexed document, including the enriched data extracted by the skillset as shown here (formatted to make it easier to read).

Code Copy

```
{
  "metadata_storage_content_type": "application/pdf",
  "metadata_storage_size": 388622,
  "<more_metadata_fields>": "...",
  "key_phrases":[
    "Margie’s Travel",
    "Margie's Travel",
    "best travel experts",
    "world-leading travel agency",
    "international reach"
  ],
  "locations":[
    "Dubai",
    "Las Vegas",
    "London",
    "New York",
    "San Francisco"
  ],
  "image_tags":[
    "outdoor",
    "tree",
    "plant",
    "palm"
  ],
  "more_fields": "..."
}
```

The ability to create *object* projections like this enables you to generate enriched data objects that can be incorporated into an enterprise data analysis solution.

View file projections

The *file* projections defined in the skillset create JPEG files for each image that was extracted from the documents during the indexing process.

1. In the *Storage browser* interface in the Azure portal, select the **margies-skillset-image-projection** blob container. This container contains a folder for each document that contained images.
2. Open any of the folders and view its contents - each folder contains at least one *.jpg file.
3. Open any of the image files, and download and view it to see the image.

The ability to generate *file* projections like this makes indexing an efficient way to extract embedded images from a large volume of documents.

View table projections

The *table* projections defined in the skillset form a relational schema of enriched data.

1. In the *Storage browser* interface in the Azure portal, expand **Tables**.
2. Select the **margiesSkillsetDocument** table to view data. This table contains a row for each document that was indexed:
3. View the **margiesSkillsetKeyPhrases** table, which contains a row for each key phrase extracted from the documents.

The ability to create *table* projections enables you to build analytical and reporting solutions that query a relational schema. The automatically generated key columns can be used to join the tables in queries - for example to return all of the key phrases extracted from a specific document.

Clean-up

Now that you've completed the exercise, delete all the resources you no longer need. Delete the Azure resources:

1. In the Azure portal, select **Resource groups**.
2. Select the resource group you don't need, then select **Delete resource group**.

More information

To learn more about Azure AI Search, see the [Azure AI Search documentation](https://docs.microsoft.com/en-us/azure/search/).