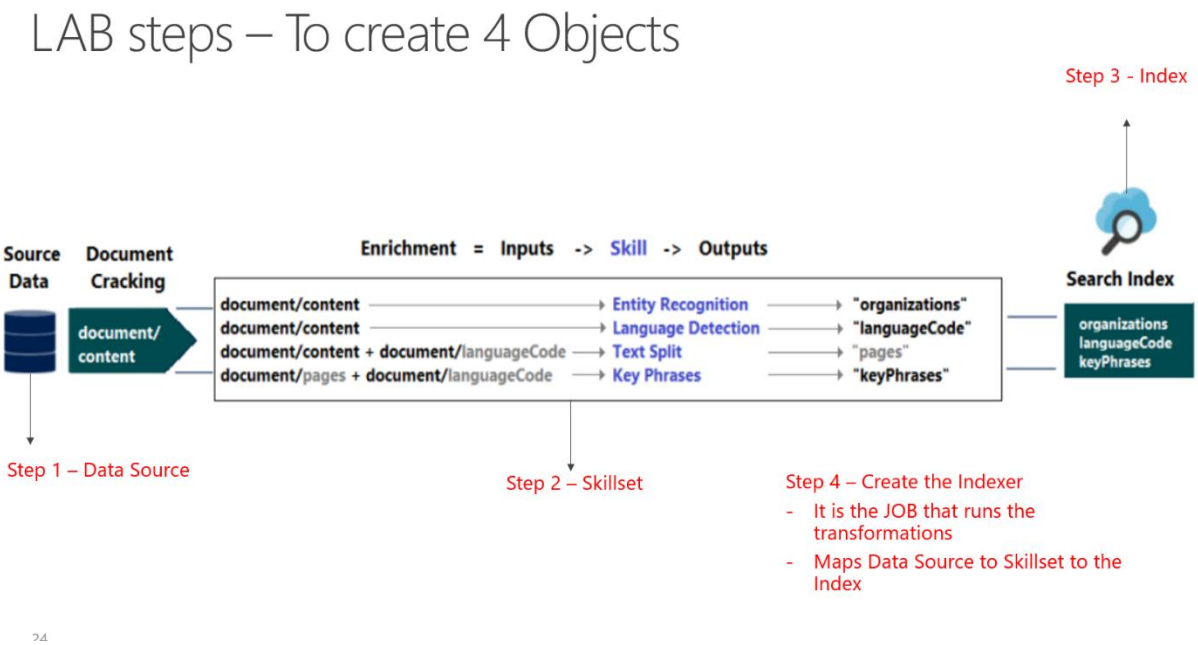
Create a Cognitive Search Enrichment Process with Text Skills

In this lab, you will learn the mechanics of programming data enrichment in Azure Cognitive Search using cognitive skills. Cognitive skills are natural language processing (NLP) and image analysis operations that extract text and text representations of an image, detect language, entities, key phrases, and more. The end result is rich additional content in an Azure Cognitive Search index, created by a cognitive search indexing pipeline.



Requirements

For this tutorial, you will use Postman to call Azure Cognitive Search service APIs. Using the POST method and Header of the Postman application, you will provide the service name and the api-key you used while creating the Azure Cognitive Search service, and you will define the content-type as JSON.

If you are not so familiar with Postman, perform the following detailed steps to define the POST method and Header settings.

- Add Datasource (replace the connection string in body)
- Add Skillset (replace cogs key in body)
- Add Index
- Add Indexer
- Check Indexer Status
- Execute Search

Instructions

Follow the steps and screenshots below to complete your experience.

Exercise 1.- Create a Data Source

Task 1. Define the POST method

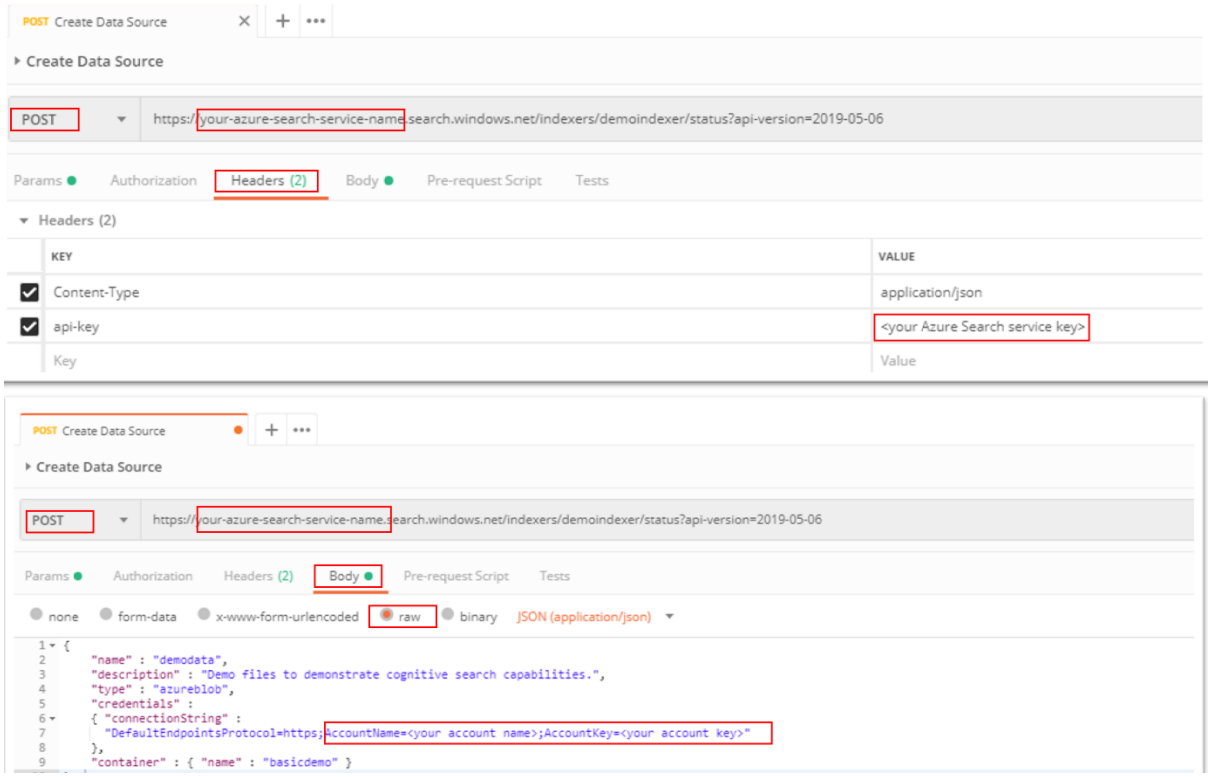
1. Open the Postman application. If a dialog box opens, close it down.
2. From the main menu, Select the New button, then select Request.
3. For the name, type **Azure Cognitive Search**.
4. For the collection, create a new collection called **Azure Cognitive Search**, then select **Save to Azure Cognitive Search**.
5. In the new request, for the request type, select the downward pointing arrow next to **GET**, and change it to select **POST**.
6. In the text box that shows the words "Enter request url" type in the following information, replacing [your service name] with the name of the Azure Cognitive Search service inside your Resource Group (xxx-search-service)

```
https://[your-service-name].search.windows.net/datasources?api-version=2019-05-06-Preview
```

7. Select the **Headers** tab. On this tab is a table of that has three columns with the titles of **KEY**, **VALUE** and **DESCRIPTION** respectively, and a single row.
8. Under the **KEY** column, in the first row, type in the following text **Content-Type**.
9. Under the **VALUE** column, in the first row, type in the following text **application/json**.
10. Under the **KEY** column, in the second row, type in the following text **api-key**.
11. Under the **VALUE** column, in the second row, paste in your **Azure Cognitive Search key**.
12. Select the Body tab. For the body type, select **Raw**
13. In the Request body, copy the following:

```
{
    "name" : "demodata",
    "description" : "Demo files to demonstrate cognitive search capabilities.",
    "type" : "azureblob",
    "credentials" :
    { "connectionString" :
      "DefaultEndpointsProtocol=https;AccountName=<your account name>;AccountKey=<your account key>"
    },
    "container" : { "name" : "basicdemo" }
}
```

Replace the connection string and container with the Azure Blob storage settings you created earlier. The following image can be used to confirm the settings you should define. Please take a look on all red boxes.



14. Select **Send**. The web test tool should return a status code of **201 Created** confirming success.
15. Switch back to the Azure portal. Confirm the data source was created in Azure Cognitive Search. On the Search service dashboard page, verify the Data Sources link has a new item. You might need to wait a few minutes for the portal page to refresh.

Tip: If you got a 403 or 404 error, check the request construction: api-version=2019-05-06 should be on the endpoint, api-key should be in the Header after Content-Type, and its value must be valid for a search service. You can reuse the header for the remaining steps in this lab.

Task 2. Create a Skillset

1. In the top navigation with tabs of your current requests, select the + to create a new request.
2. Change the request type to **PUT**.
3. Set the **URL** to:

https://[your-service-name].search.windows.net/skillsets/demoskillset?api-version=2019-05-06-Preview

4. Again, set the content-type and api-key as you did on the previous task.
5. In the Request body, you will use JSON to define the Language Detection, Text Split, Named Entity Recognition and Key Phrase Extraction Skills as follows:

```
{
  "description":
  "Extract entities, detect language and extract key-phrases",
  "skills":
  [
    {
      "@odata.type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
      "inputs": [
        {
          "name": "text", "source": "/document/content"
        }
      ],
      "outputs": [
        {
          "name": "languageCode",
          "targetName": "languageCode"
        }
      ]
    },
    {
      "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
      "textSplitMode" : "pages",
      "maximumPageLength": 4000,
      "inputs": [
      {
        "name": "text",
        "source": "/document/content"
      },
      {
        "name": "languageCode",
        "source": "/document/languageCode"
      }
    ],
    "outputs": [
      {
            "name": "textItems",
            "targetName": "pages"
      }
    ]
  },
{
      "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
      "categories": [ "Organization" ],
      "defaultLanguageCode": "en",
      "context": "/document/pages/*",
      "inputs": [
        {
          "name": "text", "source": "/document/pages/*"
        }
      ],
      "outputs": [
        {
          "name": "organizations", "targetName": "organizations"
        }
      ]
    },
  {
      "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
      "context": "/document/pages/*",
      "inputs": [
        {
          "name": "text", "source": "/document/pages/*"
        },
        {
          "name":"languageCode", "source": "/document/languageCode"
        }
      ],
      "outputs": [
        {
          "name": "keyPhrases",
          "targetName": "keyPhrases"
        }
      ]
    }
  ],
  "cognitiveServices": {
        "@odata.type":
"#Microsoft.Azure.Search.CognitiveServicesByKey",
        "description": "my-cog-serv",
        "key": "your-api-key-here"
  }
}
```

6. Towards the bottom of the body content, replace the cognitive services key with your service key from your previous lab.
7. Select **Send**. The web test tool should return a status code of **201 Created** confirming success.

Task 3. Create an index

In this section, you define the index schema by specifying which fields to include in the searchable index, and the search attributes for each field. Fields have a type and can take attributes that determine how the field is used (searchable, sortable, and so forth). Field names in an index are not required to identically match the field names in the source. In a later step, you add field mappings in an indexer to connect source-destination fields. For this step, define the index using field naming conventions pertinent to your search application.

This exercise uses the following fields and field types:

field-names:	id	content	languageCode	keyPhrases	organizations
field-types:	Edm.String	Edm.String	Edm.String	List	List

This request creates an index. Use the index name **demoindex** for the rest of this tutorial.

1. Select the + to create a new request
2. Change the request type to **PUT.**
3. Set the **URL** to:

https://[your-service-name].search.windows.net/indexes/demoindex?api-version=2019-05-06-Preview

4. Again set the content-type and api-key as you did above.
5. In the **Request body**, in the json below you are defining the same properties of the previous lab, but through the API instead of the Azure Portal:

```
{
  "fields": [
    {
      "name": "id",
      "type": "Edm.String",
      "key": true,
      "searchable": true,
      "filterable": false,
      "facetable": false,
      "sortable": true
    },
    {
      "name": "blob_uri",
      "type": "Edm.String",
      "searchable": true,
      "filterable": false,
      "facetable": false,
      "sortable": true
    },
    {
      "name": "content",
      "type": "Edm.String",
      "sortable": false,
      "searchable": true,
      "filterable": false,
      "facetable": false

    },
    {
      "name": "languageCode",
      "type": "Edm.String",
      "searchable": true,
      "filterable": false,
      "facetable": false
    },
    {
      "name": "keyPhrases",
      "type": "Collection(Edm.String)",
      "searchable": true,
      "filterable": false,
      "facetable": false
    },
    {
      "name": "organizations",
      "type": "Collection(Edm.String)",
      "searchable": true,
      "sortable": false,
      "filterable": false,
      "facetable": false
    }
  ]
}
```

6. Select **Send**. The web test tool should return a status code of **201 Created** confirming success.

Check the Azure portal to confirm the index was created in Azure Cognitive Search. On the **Search service dashboard page**, verify the **Indexes** tile has a 2 items. You might need to wait a few minutes for the portal page to refresh. Select **Indexes** to confirm that the `demoindex` appears.

Task 4. Create an indexer, map fields, and execute transformations

So far you have created a data source, a skillset, and an index. These three components become part of an indexer that pulls each piece together into a single multi-phased operation. To tie these together in an indexer, you must define field mappings. Field mappings are part of the indexer definition and execute the transformations when you submit the request.

Before you make this REST call, remember to replace the service name and the admin key in the request below if your tool does not preserve the request header between calls.

Also, provide the name of your indexer. You can reference it as `demoindexer` for the rest of this lab.

1. Select the **+** to create a new request.
2. Change the request type to **PUT.**
3. Set the **URL** to:

https://[your-service-name].search.windows.net/indexers/demoindexer?api-version=2019-05-06-Preview

4. Again set the content-type and api-key as you did above.
5. Set the **body** to the following:

```
{
  "dataSourceName" : "demodata",
  "targetIndexName" : "demoindex",
  "skillsetName" : "demoskillset",
  "fieldMappings" : [
        {
          "sourceFieldName" : "metadata_storage_path",
          "targetFieldName" : "id",
          "mappingFunction" :
            { "name" : "base64Encode" }
        },
        {
          "sourceFieldName" : "content",
          "targetFieldName" : "content"
        },
        {
          "sourceFieldName" : "metadata_storage_path",
          "targetFieldName" : "blob_uri"
        }
    ],
  "outputFieldMappings" :
  [
        {
          "sourceFieldName" : "/document/pages/*/organizations/*",
          "targetFieldName" : "organizations"
        },
        {
          "sourceFieldName" : "/document/pages/*/keyPhrases/*",
          "targetFieldName" : "keyPhrases"
        },
        {
            "sourceFieldName": "/document/languageCode",
            "targetFieldName": "languageCode"
        }
  ],
  "parameters":
  {
    "maxFailedItems":-1,
    "maxFailedItemsPerBatch":-1,
    "configuration":
    {
      "dataToExtract": "contentAndMetadata",
       "imageAction": "generateNormalizedImages"
    }
  }
}
```

6. Select **Send**. The web test tool should return a status code of **201 Created** confirming success.

Expect this step to take a minute or two to complete. Even though the data set is small, analytical skills are computation-intensive. Some skills, such as image analysis, are long-running.

Check the Azure portal to confirm the index was created in Azure Cognitive Search. On the Search service dashboard page, verify if the Indexers tile has 2 indexes (one from your previous lab). You might need to wait a few minutes for the portal page to refresh. Select Indexers to confirm that the **demoindexer** appears.

Task 5. Check indexer status

Once the indexer is defined, it runs automatically when you submit the request. Depending on which cognitive skills you defined, indexing can take longer than you expect. To find out whether the indexer is still running, send the following request to check the indexer status.

1. Select the **+** to create a new request.
2. Ensure the request type is **GET**.
3. Set the **URL** to:

https://[your-service-name].search.windows.net/indexers/demoindexer/status?api-version=2019-05-06-Preview

4. Again set the content-type and api-key as you did above.

The response tells you whether the indexer is running. Once indexing is finished, the response to the same call (as above) will result in a report of any errors and warnings that occurred during enrichment.

Warnings are common with some source file and skill combinations and do not always indicate a problem. In this lab, the warnings are benign (for example, no text inputs from the JPEG files). You can review the status response for verbose information about warnings emitted during indexing. You can also expect warnings about text been truncated or long words. If the status field has “success”, we don’t need to worry about any of the warnings.

Since the index columns and skillsets are smaller than in the previous lab, you will also notice the index size is smaller for the same set of indexed documents.

Task 6. Verify Content

After indexing is finished, run queries that return the contents of individual fields. By default, Azure Cognitive Search returns the top 50 results. The sample data is small so the default works fine. However, when working with larger data sets, you might need to include parameters in the query string to return more results - you can read how to page results in Azure Cognitive Search here.

As a verification step, query the index for all of the fields.

1. Select the **+** to create a new request.
2. Ensure the request type is **GET**.
3. Set the **URL** to:

https://[your-service-name].search.windows.net/indexes/demoindex?api-version=2019-05-06-Preview

4. Again set the content-type and api-key as you did above.

The output is the index schema, with the name, type, and attributes of each field.

Submit the second query below, to verify the metadata created with AI. Please notice that API calls are case sensitive, so it is mandatory to use exactly the same field names of the index definition.

1. Select the **+** to create a new request.
2. Ensure the request type is **GET**.
3. Set the **URL** to:

https://[your-service-name].search.windows.net/indexes/demoindex/docs?search=*&api-version=2019-05-06-Preview

4. Again set the content-type and api-key as you did above.

Note As you can see, it is possible to return multiple fields via **$select** using a comma-delimited list.

You can use **GET** or **POST**, depending on query string complexity and length. For more information, see Query using the **REST API**.