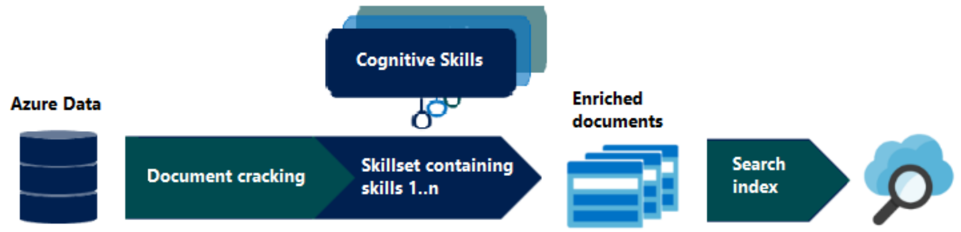
Data & AI Tech Immersion Workshop – Product Review Guide and Lab Instructions

Day 2, Experience 2 - Yield quick insights from unstructured data with Knowledge Mining and Cognitive Search

Technology overview

Cognitive search is an AI feature in Azure Search, used to extract text from images, blobs, and other unstructured data sources - enriching the content to make it more searchable in an Azure Search index. Extraction and enrichment are implemented through cognitive skills attached to an indexing pipeline. AI enrichments are supported in the following ways:

- **Natural language processing** skills include entity recognition, language detection, key phrase extraction, text manipulation, and sentiment detection. With these skills, unstructured text can assume new forms, mapped as searchable and filterable fields in an index.
- **Image processing** skills include Optical Character Recognition (OCR) and identification of visual features, such as facial detection, image interpretation, image recognition (famous people and landmarks) or attributes like colors or image orientation. You can create text-representations of image content, searchable using all the query capabilities of Azure Search.



Cognitive Search diagram

Cognitive skills in Azure Search are based on machine learning models in Cognitive Services APIs: Computer Vision and Text Analysis.

Natural language and image processing is applied during the data ingestion phase, with results becoming part of a document's composition in a searchable index in Azure Search. Data is sourced as an Azure data set and then pushed through an indexing pipeline using whichever built-in skills you need. The architecture is extensible so if the built-in skills are not sufficient, you can create and attach custom skills to integrate custom processing. Examples might be a custom entity module or document classifier targeting a specific domain such as finance, scientific publications, or medicine.

Scenario overview

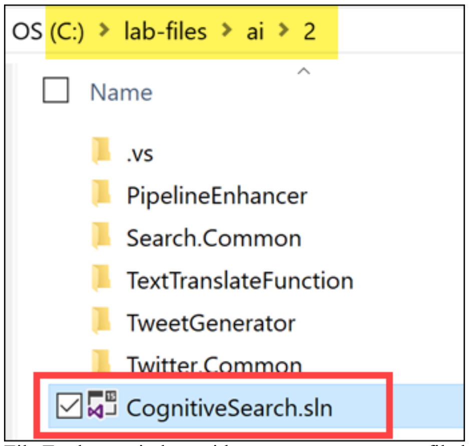
ContosoAuto is interested in using their unstructured data to gain further insights into how customers perceive their business, and the key things their customers are talking about. They are looking for a pilot that would use tweets streamed from Twitter into a `tweets` collection in their Cosmos DB instance to better understand what customers are saying about their organization on the platform. They are also looking to get a better understanding of whether the trend of messages is positive, negative, or neutral by performing sentiment analysis on the tweets.

In this experience, you will learn the mechanics of using Cognitive Search and Knowledge Mining to yield rapid insights into unstructured data. Using a combination of preconfigured and custom cognitive skills in Azure Search, you will create a Cognitive Search indexing pipeline that enriches source data in route to an index. 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 Search index, created by a cognitive search indexing pipeline. The output is a full-text searchable index on Azure Search.

Task 1: Populate Cosmos DB with tweets from a generator

For this experience, you will be using the `tweets` collection in ContosoAuto's Cosmos DB as the data source for your Cognitive Search pipeline. In order to use Cosmos DB as a data source, documents must exist in the target collection prior to creating the Data Source in Azure Search. In this task, you will populate the `tweets` collection in your Cosmos DB `ContosoAuto` database using a tweet generator application running in Visual Studio.

1. Open File Explorer and navigate to `C:\lab-files\ai\2`. Double-click on **`CognitiveSearch.sln`** to open the solution in Visual Studio. If you are prompted about how to open the file, choose **Visual Studio 2017**. If you are prompted by Visual Studio to log in, use the Azure credentials you are using for this workshop.

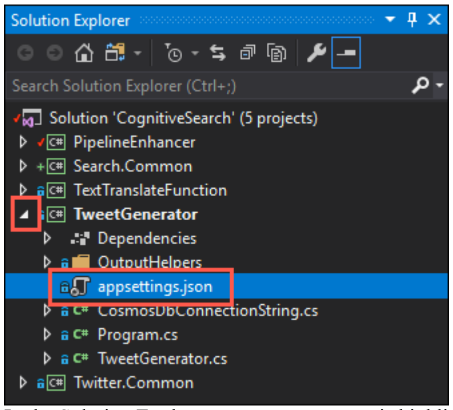


File Explorer window with `CognitiveSearch.sln` file highlighted.

The Visual Studio solution contains the following projects:

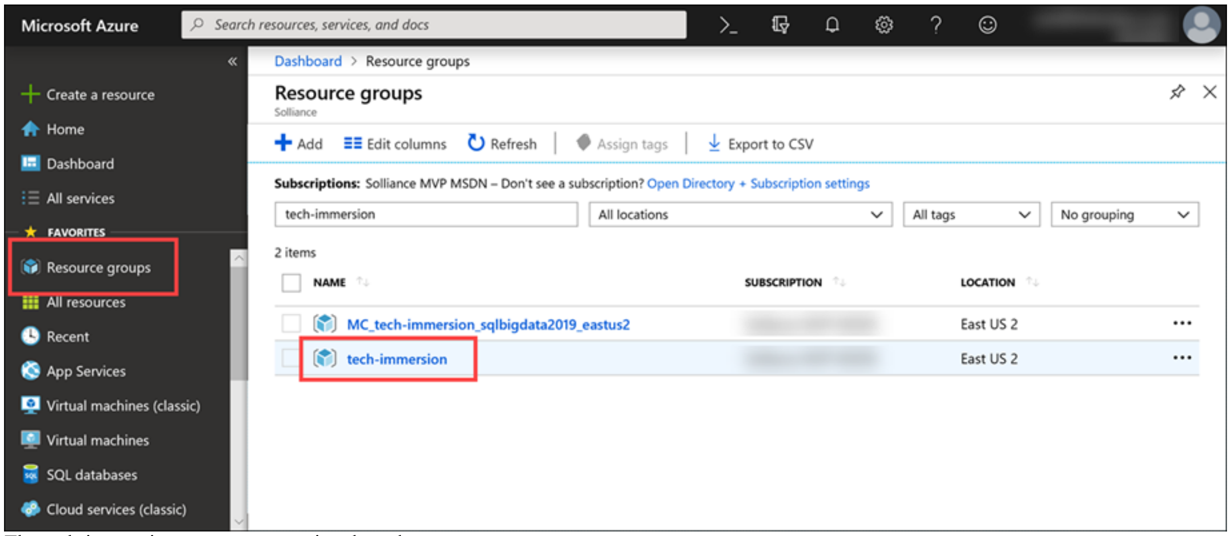
- **PipelineEnhancer**: Console app that interacts with Azure Search Service REST APIs to enhance the Cognitive Search pipeline.
- **Search.Common**: Common library containing models and structs used by the `PipelineEnhancer` to communicate with the Azure Search Service APIs.
- **TextTranslateFunction**: Azure Function with an HTTP Trigger that will be used translate non-English tweets to English.
- **TweetGenerator**: Console app that generates simulated tweets and sends them to Cosmos DB.
- **Twitter.Common**: Common library containing models and structs used by the other projects within the solution.

2. In the Solution Explorer on the left-hand side of Visual Studio, expand the **TweetGenerator** project, and then locate and open the `appsettings.json` file.



In the Solution Explorer, `appsettings.json` is highlighted in the TweetGenerator project.

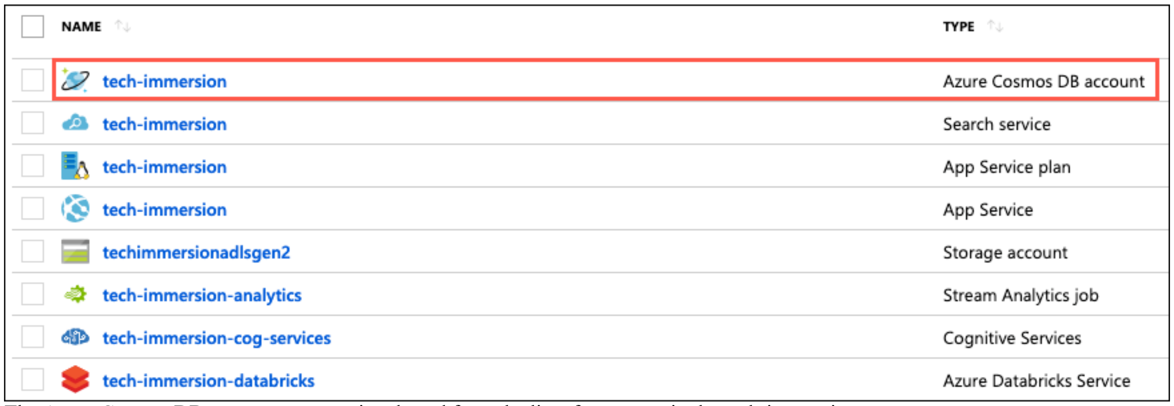
3. Next, you need to retrieve your Cosmos DB connection string. This will be used to enable the TweetGenerator to write data into your `tweets` collection. In the [Azure portal](), select **Resource groups** from the left-hand menu, and then select the **tech-immersion-XXXXX** resource group (where XXXXX is the unique identifier assigned to you for this workshop).



NAME	SUBSCRIPTION	LOCATION
MC_tech-immersion_sqlbigdata2019_eastus2	[redacted]	East US 2
tech-immersion	[redacted]	East US 2

The tech-immersion resource group is selected.

4. Select the **tech-immersionXXXXX** Azure Cosmos DB account from the list of resources (where XXXXX is the unique identifier assigned to you for this workshop).

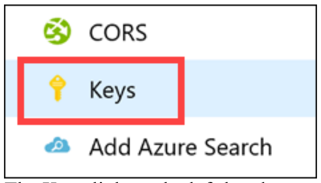


NAME	TYPE
tech-immersion	Azure Cosmos DB account
tech-immersion	Search service
tech-immersion	App Service plan
tech-immersion	App Service
techimmersionadlsgen2	Storage account
tech-immersion-analytics	Stream Analytics job
tech-immersion-cog-services	Cognitive Services
tech-immersion-databricks	Azure Databricks Service

The Azure Cosmos DB account resource is selected from the list of resources in the tech-immersion resource group.

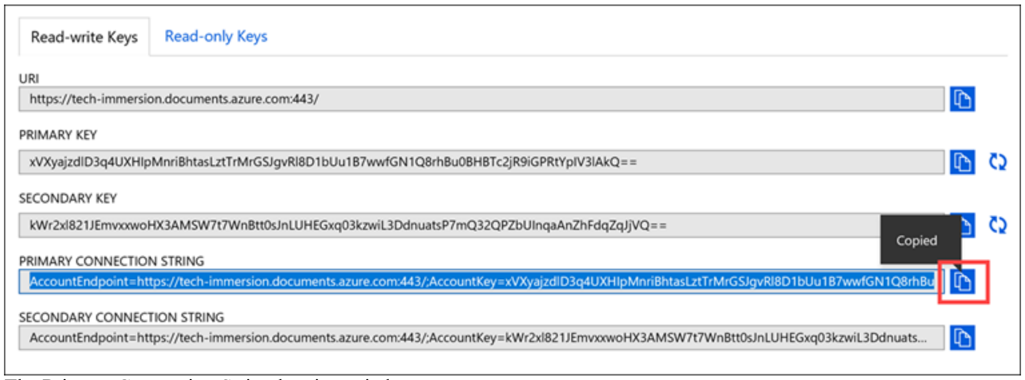
IMPORTANT: There may be two Cosmos DB accounts in your resource group. Select the Cosmos DB account named **tech-immersionXXXXX**, with no hyphen between immersion and XXXXX.

5. On your Cosmos DB blade, select **Keys** from the left-hand menu.



The Keys link on the left-hand menu is highlighted.

6. Copy the **Primary Connection String** value by selecting the copy button to the right of the field.



URI
https://tech-immersion.documents.azure.com:443/

PRIMARY KEY
xVXyajzdlD3q4UXHlpMnriBhtasLztTrMrGSJgvRl8D1bUu1B7wwfGN1Q8rhBu0BHBTc2jR9iGPRtYplV3lAkQ==

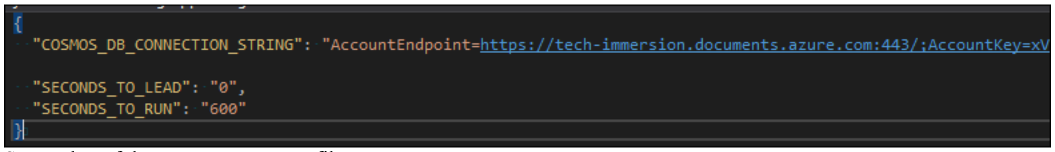
SECONDARY KEY
kWr2xl821JEmvxxwoHX3AMSW7t7WnBtt0sJnLUHEGxq03kzwiL3DdnuatsP7mQ32QPZbUlnqaAnZhFdqZqJjVQ==

PRIMARY CONNECTION STRING
AccountEndpoint=https://tech-immersion.documents.azure.com:443/;AccountKey=xVXyajzdlD3q4UXHlpMnriBhtasLztTrMrGSJgvRl8D1bUu1B7wwfGN1Q8rhBu

SECONDARY CONNECTION STRING
AccountEndpoint=https://tech-immersion.documents.azure.com:443/;AccountKey=kWr2xl821JEmvxxwoHX3AMSW7t7WnBtt0sJnLUHEGxq03kzwiL3Ddnuats...

The Primary Connection String key is copied.

7. Return to the `appsettings.json` file in Visual Studio and paste the Primary Connection String into the value for the `COSMOS_DB_CONNECTION_STRING` setting. Your `appsettings.json` file should look similar to the following:

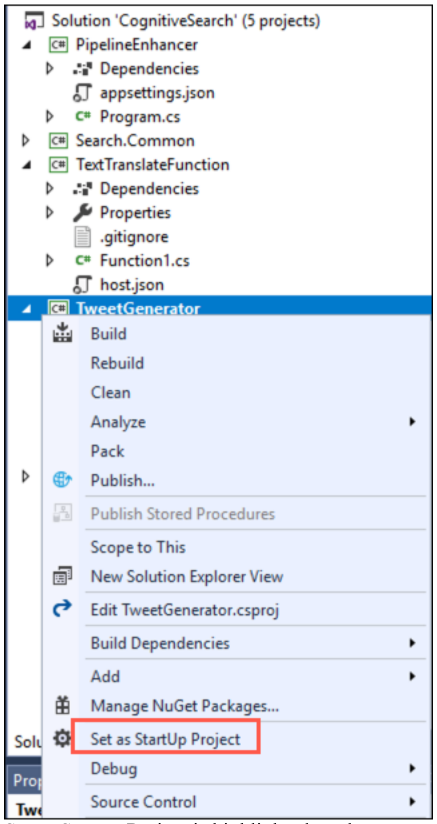


```
{
  "COSMOS_DB_CONNECTION_STRING": "AccountEndpoint=https://tech-immersion.documents.azure.com:443/;AccountKey=xV

  "SECONDS_TO_LEAD": "0",
  "SECONDS_TO_RUN": "600"
}
```

Screenshot of the `appsettings.json` file.

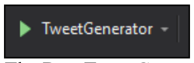
8. Save `appsettings.json`.

9. Right-click on the `TweetGenerator` project, and select **Set as StartUp Project** from the context menu.



Set as Startup Project is highlighted on the context menu for the TweetGenerator project.

10. Run the console app by selecting the button in the toolbar with the green arrow and text of **TweetGenerator**.



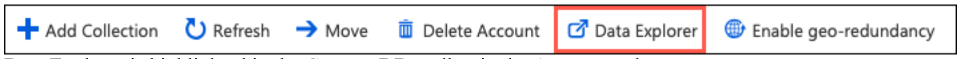
The Run TweetGenerator button is displayed.

Leave the `TweetGenerator` console app running in the background while you move on to the remaining tasks in this experience. The app will run for 10 minutes, sending random tweets into your Cosmos DB `tweets` collection, so you have data to work with in the remaining tasks of this experience. In the next task, you will set up an Azure Search Index which points to the `tweets` collection in Cosmos DB, so as new tweets are added, they will be indexed.

Task 2: Create a basic Cognitive Search pipeline using the Azure portal

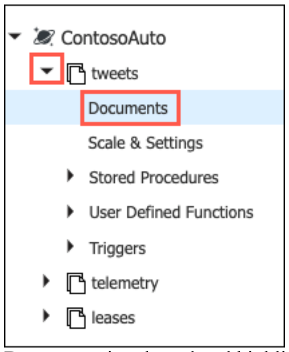
With data now streaming into your Cosmos DB `tweets` collection, you are ready to set up a basic Cognitive Search pipeline using the Azure portal. In this task, you will create an Azure Search Index and configure an Azure Search Indexer to read tweets from your Cosmos DB collection. You will also include several pre-configured skills linked to your Cognitive Services account to extract more information out of the tweets being indexed.

1. Return to your Azure Cosmos DB account blade in the Azure portal, and select **Data Explorer** from the toolbar on the overview blade.



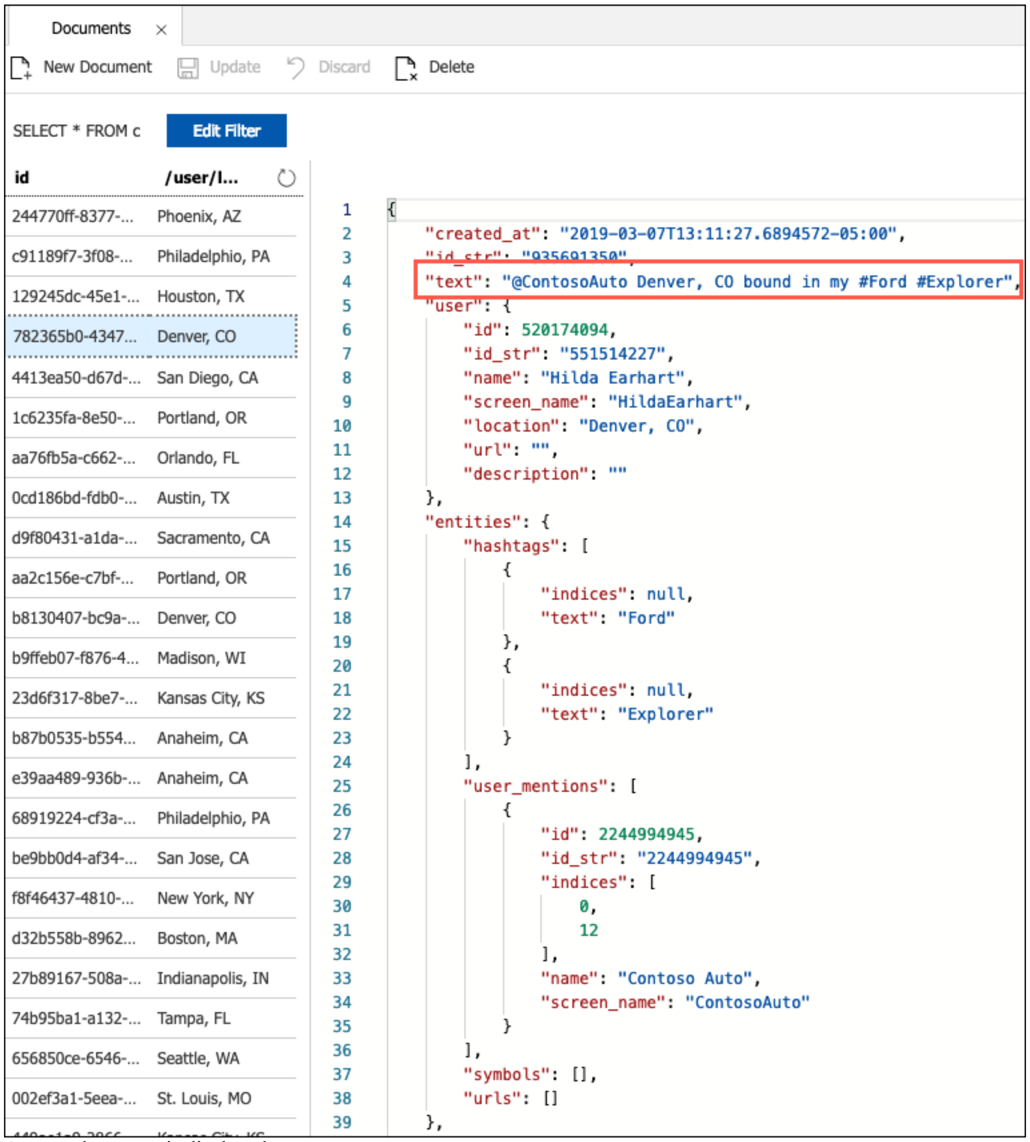
Data Explorer is highlighted in the Cosmos DB toolbar in the Azure portal.

2. Under the `ContosoAuto` database, expand the **tweets** collection and then select **Documents**.



Documents is selected and highlighted under the tweets collection in the `ContosoAuto` Cosmos DB database.

3. In the documents pane, select any of the documents listed and inspect a tweet document. Documents are stored in JSON (JavaScript Object Notation) format in Cosmos.



SELECT * FROM c

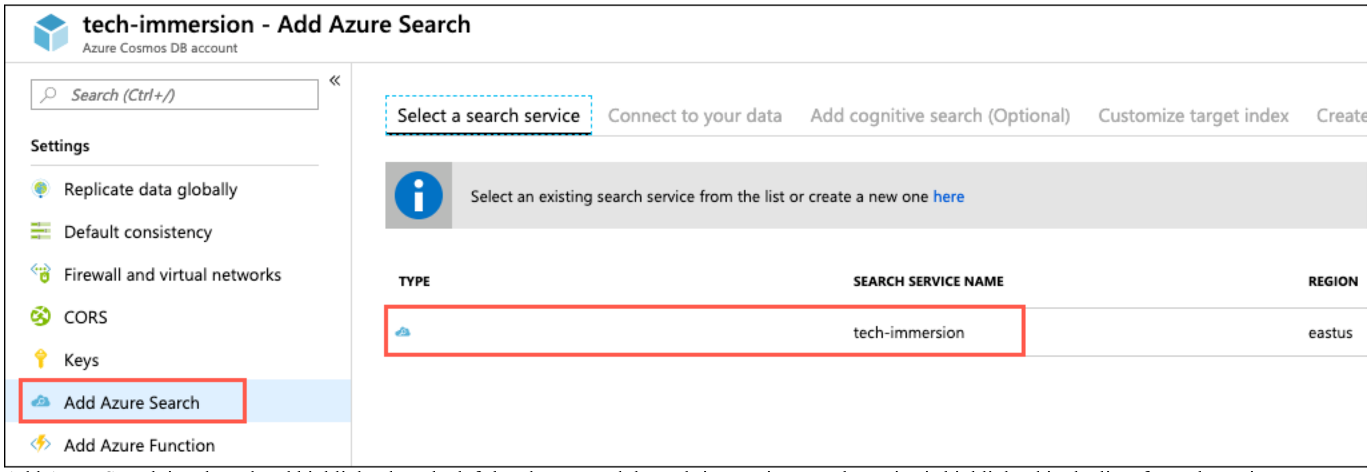
id	/user/l...
244770ff-8377-...	Phoenix, AZ
c91189f7-3f08-...	Philadelphio, PA
129245dc-45e1-...	Houston, TX
782365b0-4347...	Denver, CO
4413ea50-d67d-...	San Diego, CA
1c6235fa-8e50-...	Portland, OR
aa76fb5a-c662-...	Orlando, FL
0cd186bd-fdb0-...	Austin, TX
d9f80431-a1da-...	Sacramento, CA
aa2c156e-c7bf-...	Portland, OR
b8130407-bc9a-...	Denver, CO
b9ffeb07-f876-4...	Madison, WI
23d6f317-8be7-...	Kansas City, KS
b87b0535-b554...	Anaheim, CA
e39aa489-936b-...	Anaheim, CA
68919224-cf3a-...	Philadelphio, PA
be9bb0d4-af34-...	San Jose, CA
f8f46437-4810-...	New York, NY
d32b558b-8962...	Boston, MA
27b89167-508a-...	Indianapolis, IN
74b95ba1-a132-...	Tampa, FL
656850ce-6546-...	Seattle, WA
002ef3a1-5eea-...	St. Louis, MO

```
{
    "created_at": "2019-03-07T13:11:27.6894572-05:00",
    "id_str": "935691350",
    "text": "@ContosoAuto Denver, CO bound in my #Ford #Explorer",
    "user": {
        "id": 520174094,
        "id_str": "551514227",
        "name": "Hilda Earhart",
        "screen_name": "HildaEarhart",
        "location": "Denver, CO",
        "url": "",
        "description": ""
    },
    "entities": {
        "hashtags": [
            {
                "indices": null,
                "text": "Ford"
            },
            {
                "indices": null,
                "text": "Explorer"
            }
        ],
        "user_mentions": [
            {
                "id": 2244994945,
                "id_str": "2244994945",
                "indices": [
                    0,
                    12
                ],
                "name": "Contoso Auto",
                "screen_name": "ContosoAuto"
            }
        ],
        "symbols": [],
        "urls": []
    },
```

A tweet document is displayed.

The `text` field, which contains the content of the tweet, and is what you will be using as you begin building your Cognitive Search pipeline. You will be using other fields later, such as the `user` entity, so take time to review these structures in the document as well.

4. With a better understanding of the structure of the tweet documents stored in Cosmos DB, let's move on to creating a basic Cognitive Search pipeline. From your Cosmos DB blade in the Azure portal, select **Add Azure Search** from the left-hand menu, select your **tech-immersion** search service, and then select **Next: Connect to your data**.



Select an existing search service from the list or create a new one here

TYPE	SEARCH SERVICE NAME	REGION
	tech-immersion	eastus

Add Azure Search is selected and highlighted on the left-hand menu and the tech-immersion search service is highlighted in the list of search services.

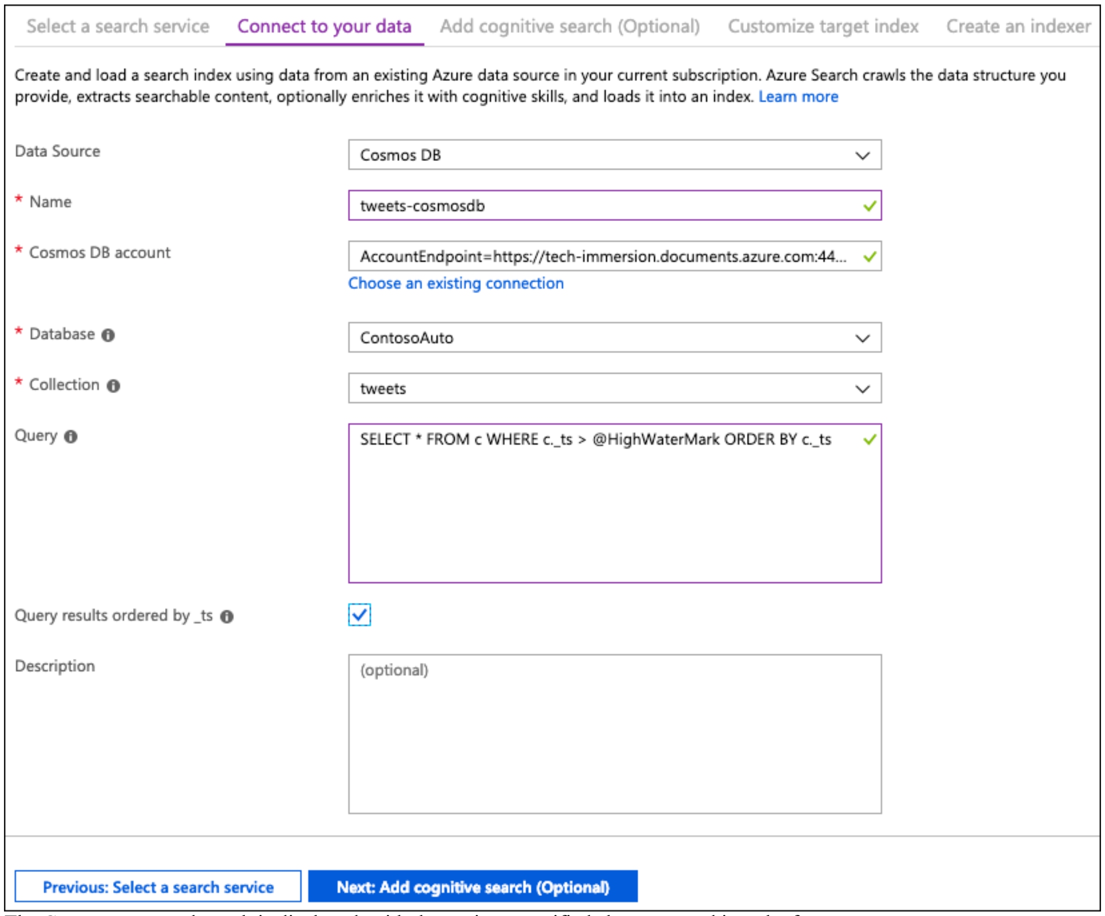
5. On the **Connect to your data** tab, enter the following:

    - **Data source**: This should be pre-populated with **Cosmos DB**. The data source object tells Azure Search how to retrieve external source data.

- **Name**: Enter **tweets-cosmosdb**.
- **Cosmos DB account**: This should be pre-populated with the connection string for your Cosmos DB account.
- **Database**: Select the **ContosoAuto** database.
- **Collection**: Select the **tweets** collection.
- **Query**: Paste the SQL statement below into the field.

  ```
  SELECT * FROM c WHERE c._ts > @HighWaterMark ORDER BY c._ts
  ```

- **Query results ordered by _ts**: Check this box.



Create and load a search index using data from an existing Azure data source in your current subscription. Azure Search crawls the data structure you provide, extracts searchable content, optionally enriches it with cognitive skills, and loads it into an index. Learn more

Data Source	Cosmos DB
* Name	tweets-cosmosdb
* Cosmos DB account	AccountEndpoint=https://tech-immersion.documents.azure.com:44... Choose an existing connection
* Database	ContosoAuto
* Collection	tweets
Query	SELECT * FROM c WHERE c._ts > @HighWaterMark ORDER BY c._ts
Query results ordered by _ts	✓
Description	(optional)

Previous: Select a search service
Next: Add cognitive search (Optional)

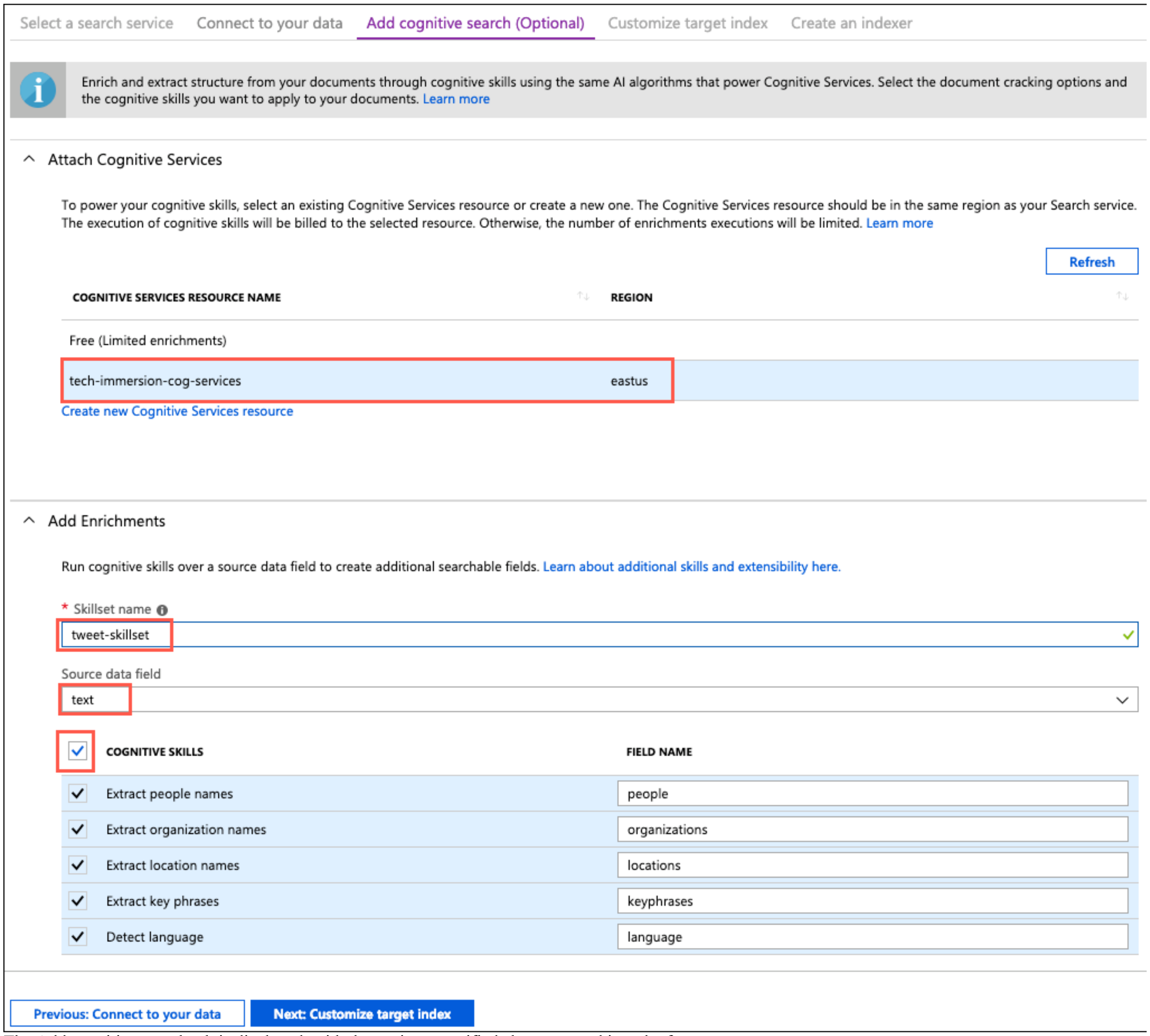
The Connect to your data tab is displayed, with the settings specified above entered into the form.

6. Select **Next: Add cognitive search (Optional)**.

   In this step, you will add a set of enrichment steps to the data being ingested from Cosmos DB. In a Cognitive Search pipeline, individual enrichment steps are called *skills*, and the collection of enrichment steps is a *skillset*. The predefined skills available at this step through the UI use pre-trained models to extract additional information from the documents. The EntityRecognitionSkill extracts entities (people, location, organization, emails, URLs, DateTime fields) from the document. The LanguageDetectionSkill is used to detect the primary language used in the document, and the KeyPhraseExtractionSkill detects important phrases based on term placement, linguistic rules, proximity to other terms, and how unusual the term is within the source data. Read about all of the available predefined cognitive skills to learn more.

7. On the **Add cognitive search (Optional)** tab:

   - Expand **Attached Cognitive Services** and select your **tech-immersion-cogserv** instance. This will associate your Cognitive Services account with the Skillset you are creating.
   - Expand **Add Enrichments**, enter **tweet-skillset** as the name, and select **text** as the source data field. Check the box next to Cognitive Skills to select all of the options.



Enrich and extract structure from your documents through cognitive skills using the same AI algorithms that power Cognitive Services. Select the document cracking options and the cognitive skills you want to apply to your documents. Learn more

Attach Cognitive Services

To power your cognitive skills, select an existing Cognitive Services resource or create a new one. The Cognitive Services resource should be in the same region as your Search service. The execution of cognitive skills will be billed to the selected resource. Otherwise, the number of enrichments executions will be limited. Learn more

COGNITIVE SERVICES RESOURCE NAME	REGION
Free (Limited enrichments)	
tech-immersion-cog-services	eastus

Add Enrichments

Run cognitive skills over a source data field to create additional searchable fields. Learn about additional skills and extensibility here.

* Skillset name: tweet-skillset

Source data field: text

✓ COGNITIVE SKILLS	FIELD NAME
✓ Extract people names	people
✓ Extract organization names	organizations
✓ Extract location names	locations
✓ Extract key phrases	keyphrases
✓ Detect language	language

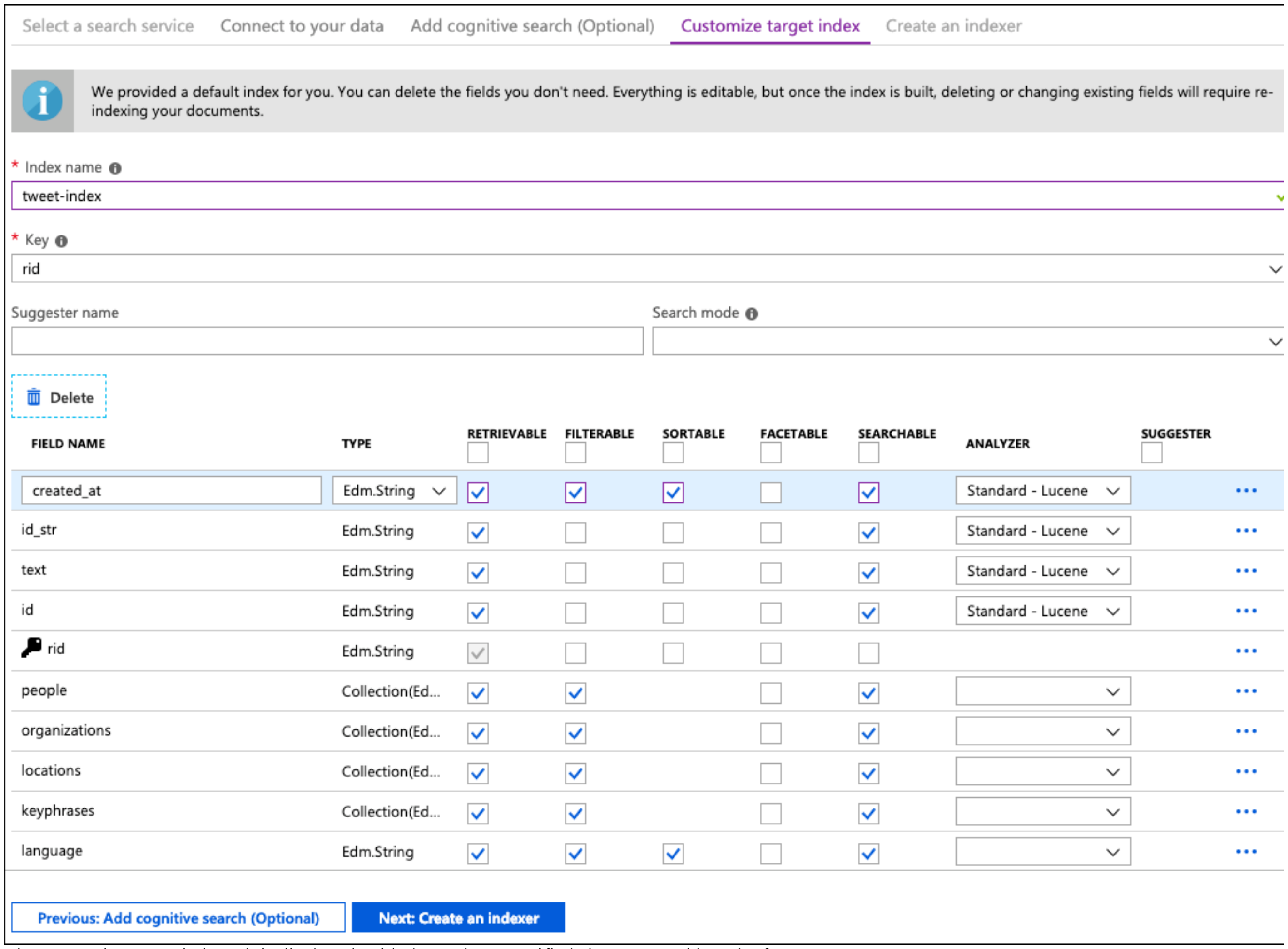
The Add cognitive search tab is displayed, with the settings specified above entered into the form.

Cognitive search is an AI feature in Azure Search, used to extract text from images, blobs, and other unstructured data sources - enriching the content to make it more searchable in an Azure Search index. Extraction and enrichment are implemented through cognitive skills attached to an indexing pipeline. Cognitive skills in Azure Search are based on machine learning models in Cognitive Services APIs: Computer Vision and Text Analysis. To learn more, read What is “cognitive search” in Azure Search.

8. Select **Next: Customize target index**.

9. On the **Customize target index** tab, do the following:

   - **Name**: Enter **tweet-index**.
   - **Key**: Leave this set to **rid**.
   - **Suggester name**: Leave this blank.
   - **Search mode**: Leave this blank.
   - Use the image below to set the checkboxes and settings for each field.


Select a search service | Connect to your data | Add cognitive search (Optional) | Customize target index | Create an indexer

We provided a default index for you. You can delete the fields you don't need. Everything is editable, but once the index is built, deleting or changing existing fields will require re-indexing your documents.

* Index name: tweet-index

* Key: rid

Suggester name:

Search mode:

Delete

FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACETABLE	SEARCHABLE	ANALYZER	SUGGESTER
created_at	Edm.String	✓	✓	✓	☐	✓	Standard - Lucene	
id_str	Edm.String	✓	☐	☐	☐	✓	Standard - Lucene	
text	Edm.String	✓	☐	☐	☐	✓	Standard - Lucene	
id	Edm.String	✓	☐	☐	☐	✓	Standard - Lucene	
rid	Edm.String	✓	☐	☐	☐	☐		
people	Collection(Ed...	✓	✓		☐	✓		
organizations	Collection(Ed...	✓	✓		☐	✓		
locations	Collection(Ed...	✓	✓		☐	✓		
keyphrases	Collection(Ed...	✓	✓		☐	✓		
language	Edm.String	✓	✓	✓	☐	✓		

Previous: Add cognitive search (Optional) | Next: Create an indexer

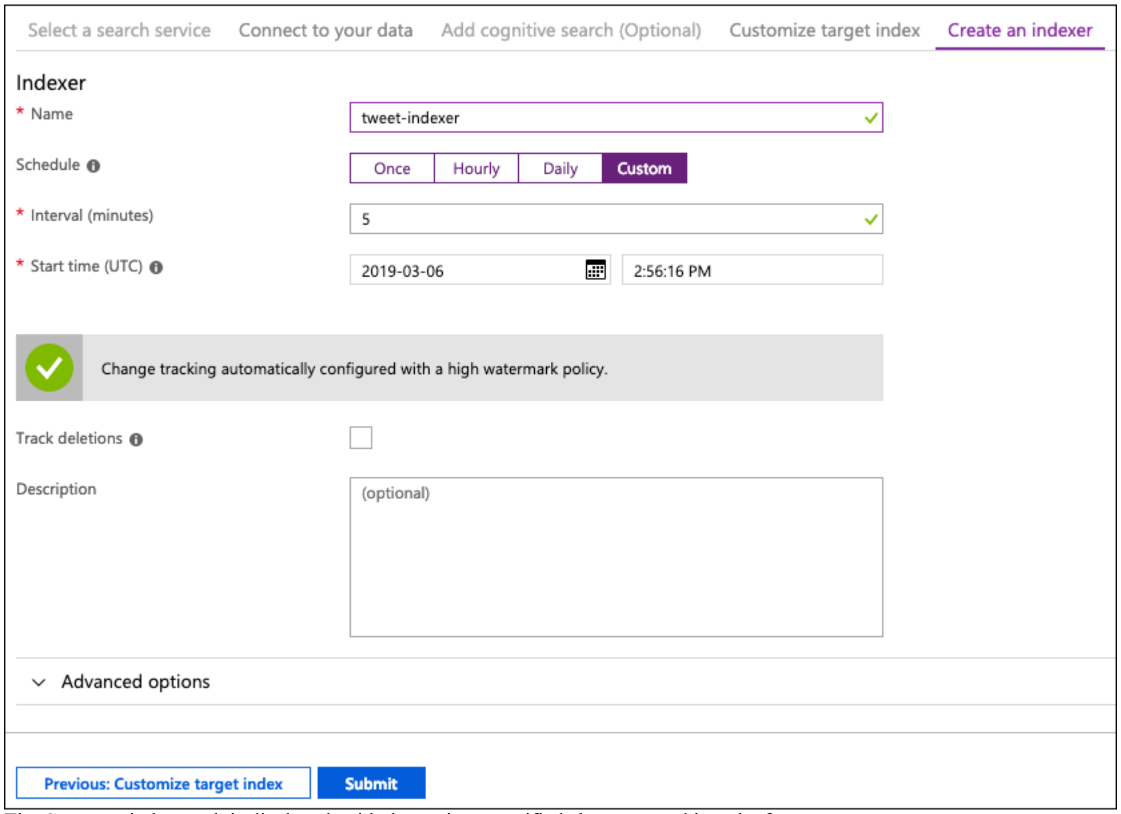
The Customize target index tab is displayed, with the settings specified above entered into the form.

On the Index page, you are presented with a list of fields with a data type and a series of checkboxes for setting index attributes. You can bulk-select attributes by clicking the checkbox at the top of an attribute column. Choose Retrievable and Searchable for every field that should be returned to a client app and subject to full text search processing. You'll notice that integers are not full text or fuzzy searchable (numbers are evaluated verbatim and are often useful in filters). Read the description of index attributes for more information.

10. Select **Next: Create an indexer**.

11. On the **Create an indexer** tab, set the following:

    - **Name**: Enter **tweet-indexer**.
    - **Schedule**: Select **Custom**, and enter **5** into the **Interval (minutes)** field.



Indexer

* Name	tweet-indexer
Schedule	Once Hourly Daily [✓ Custom]
* Interval (minutes)	5
* Start time (UTC)	2019-03-06 2:56:16 PM

Change tracking automatically configured with a high watermark policy.

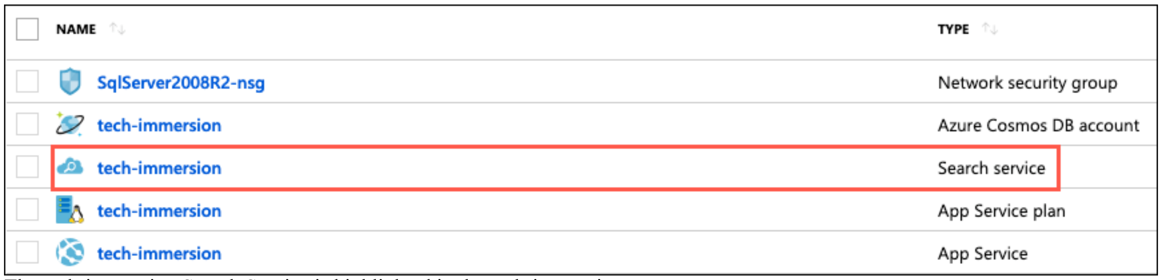
Track deletions	☐
Description	(optional)

The Create an indexer tab is displayed, with the settings specified above entered into the form.

An indexer in Azure Search is a crawler that extracts searchable data and metadata from an external Azure data source and populates an index based on field-to-field mappings between the index and your data source. This approach is sometimes referred to as a ‘pull model’ because the service pulls data in without you having to write any code that adds data to an index.

12. Select **Submit**.

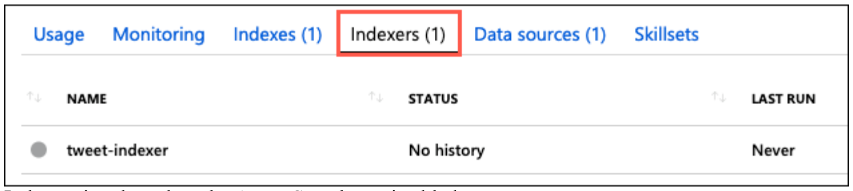
13. Next, navigate to the **techimmersionXXXXX** Azure Search Service (where XXXXX is the unique identifier assigned to you for this workshop) in the Azure portal by selecting it from the list of resources in the **tech-immersion-XXXXX** resource group.



NAME	TYPE
SqlServer2008R2-nsg	Network security group
tech-immersion	Azure Cosmos DB account
tech-immersion	Search service
tech-immersion	App Service plan
tech-immersion	App Service

The tech-immersion Search Service is highlighted in the tech-immersion resource group.

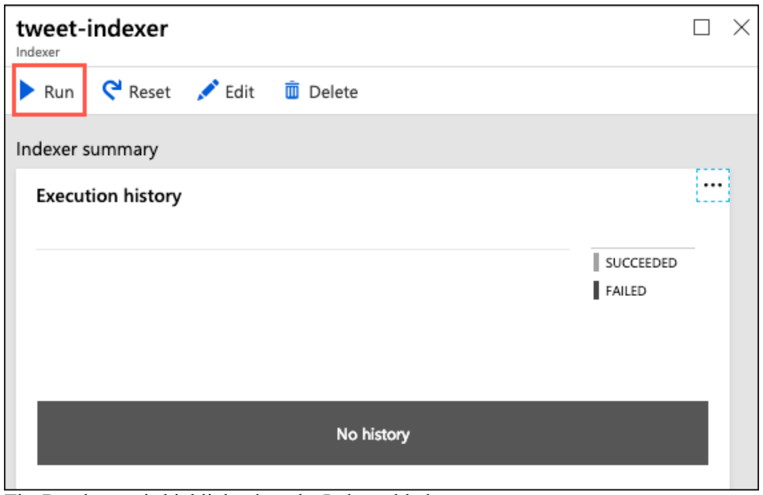
14. On the Azure Search service blade, select **Indexers**.



NAME	STATUS	LAST RUN
tweet-indexer	No history	Never

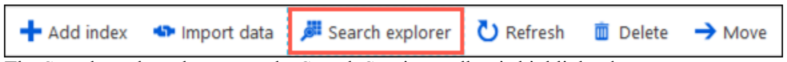
Indexers is selected on the Azure Search service blade.

15. Your indexer was scheduled to run every 5 minutes, which is the minimum allowable time. If your indexer has a status of **No history**, you can bypass the scheduled time, and force the indexer to run, by selecting the indexer, and then selecting **Run** on the Indexer blade.



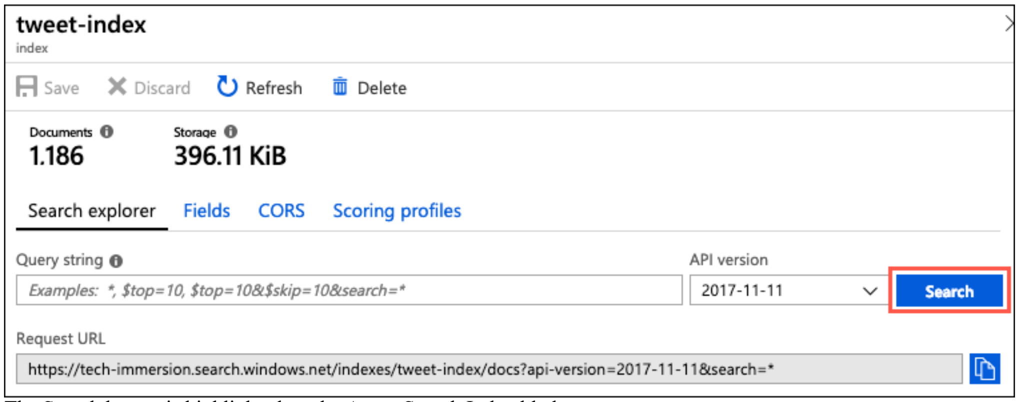
The Run button is highlighted on the Indexer blade.

16. Once your Indexer has run, select **Search explorer** on the Search Service toolbar.



The Search explorer button on the Search Service toolbar is highlighted.

17. On the Search explorer tab, select **Search** and observe the results.



The Search button is highlighted on the Azure Search Index blade.

18. Looking at the items in the search results, you will see that they are resemble the following:

```
{
  "@search.score": 1,
  "created_at": "3/7/2019 6:11:28 PM",
  "id_str": "707132469",
  "text": "@ContosoAuto Thank you for the amazing service at the Phoenix, AZ service center #Tesla #Model3",
  "id": "d9f80431-a1da-4cb8-84d8-093e9bdf9a8b",
  "rid": "YmpnZEFKMUhibTAtQUFBQUFBQUFBQT090",
  "people": [],
  "organizations": [],
  "locations": ["Phoenix", "AZ"],
  "keyphrases": [
    "amazing service",
    "AZ service center",
    "Phoenix",
    "Tesla"
  ],
  "language": "en"
}
```

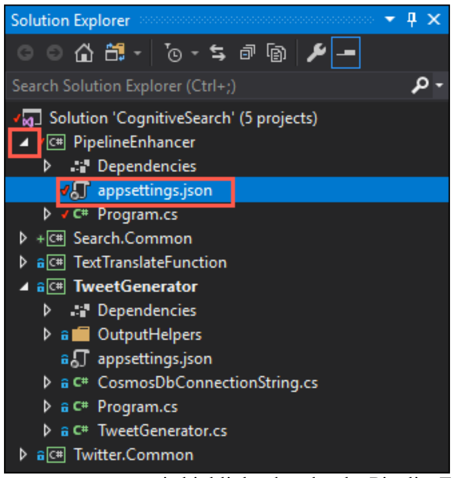
You may notice that several of the fields available in the `tweet` documents stored in Cosmos DB are not in the search result JSON object. For example, the `user` entity, which is a child object of a `tweet`, is not available to add to your search index when using the UI. To leverage the full capabilities of Cognitive Search in Azure, you will need to use the REST APIs. To learn more about using the REST API and how to use them for building a more robust Cognitive Search pipeline, you can review the [Azure Search Service REST API documentation]().

Task 3: Enhance the Cognitive Search pipeline

In the previous task, you created the beginnings of your Cognitive Search pipeline when you added Cognitive Skills for extracting people, organization and location names, key phrases and for detecting the primary language. In this task, you will enhance your Cognitive Search pipeline using functionality available only through the [Azure Search Service REST APIs]().

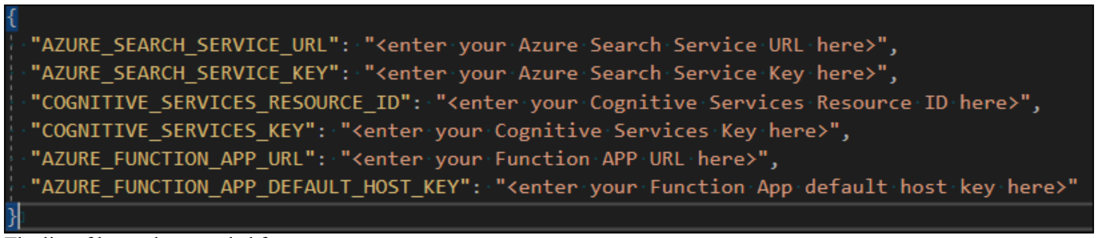
1. First, you will run a simple Visual Studio console application, `PipelineEnhancer`, which will make calls to the Azure Search Service REST APIs to update your Skillset, Index, and Indexer to accommodate the a new field named `sentiment`.

2. To prepare the console application, you need to update the `appsettings.json` file for the project in information about your search service. Return to Visual Studio, and open the `appsettings.json` file located under the `PipelineEnhancer` project.



`appsettings.json` is highlighted under the PipelineEnhancer solution.

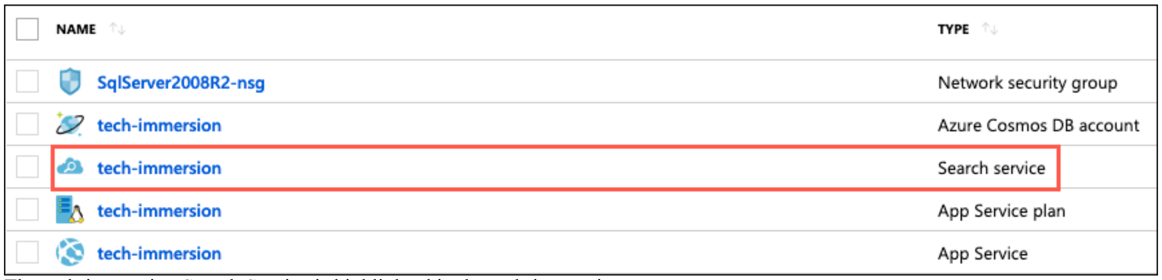
The `appsettings.json` file will look like the following, and here you can see the values that you need to retrieve before moving on.



```
{
  "AZURE_SEARCH_SERVICE_URL": "<enter your Azure Search Service URL here>",
  "AZURE_SEARCH_SERVICE_KEY": "<enter your Azure Search Service Key here>",
  "COGNITIVE_SERVICES_RESOURCE_ID": "<enter your Cognitive Services Resource ID here>",
  "COGNITIVE_SERVICES_KEY": "<enter your Cognitive Services Key here>",
  "AZURE_FUNCTION_APP_URL": "<enter your Function APP URL here>",
  "AZURE_FUNCTION_APP_DEFAULT_HOST_KEY": "<enter your Function App default host key here>"
}
```

The list of key values needed from `appsettings.json`.

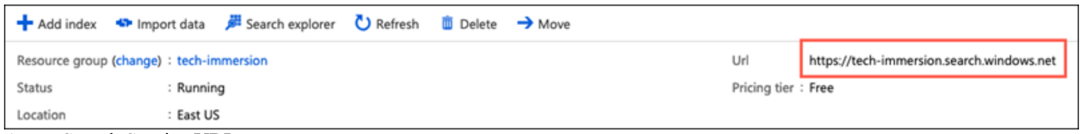
3. To retrieve values for the required settings for your Azure Search Service, navigate to your Azure Search Service in the Azure portal by selecting it from the list of resources in the **tech-immersion-XXXXX** resource group (where XXXXX is the unique identifier assigned to you for this workshop).



NAME	TYPE
SqlServer2008R2-nsg	Network security group
tech-immersion	Azure Cosmos DB account
tech-immersion	Search service
tech-immersion	App Service plan
tech-immersion	App Service

The tech-immersion Search Service is highlighted in the tech-immersion resource group.

4. On the overview blade of your search service, copy the **Url** field.



Resource group (change) : tech-immersion
Status : Running
Location : East US
Url https://tech-immersion.search.windows.net
Pricing tier : Free

Azure Search Service URL

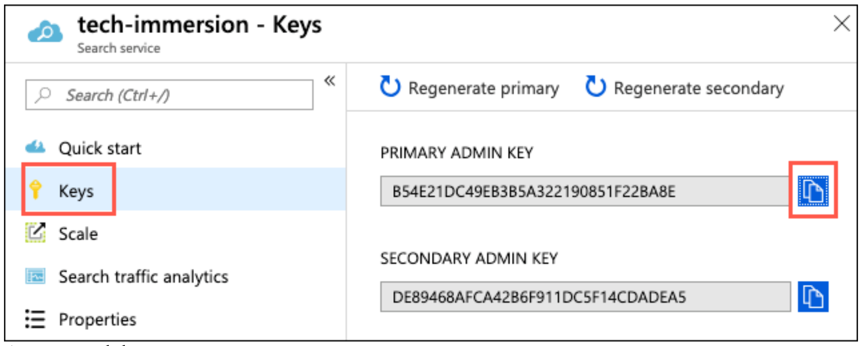
5. Return to Visual Studio, and in the `appsettings.json` file, locate the line that looks like the following:

```
"AZURE_SEARCH_SERVICE_URL": "<enter your Azure Search Service URL here>"
```

6. Replace the value of the `AZURE_SEARCH_SERVICE_URL` setting by pasting the copied URL within double-quotes. The line should now look similar to this:

```
"AZURE_SEARCH_SERVICE_URL": "https://tech-immersion.search.windows.net"
```

7. Return to your Search Service blade in the Azure portal, select **Keys** from the left-hand menu, and the select the **Copy** button for the **PRIMARY ADMIN KEY**.



PRIMARY ADMIN KEY: B54E21DC49EB3B5A322190851F22BA8E

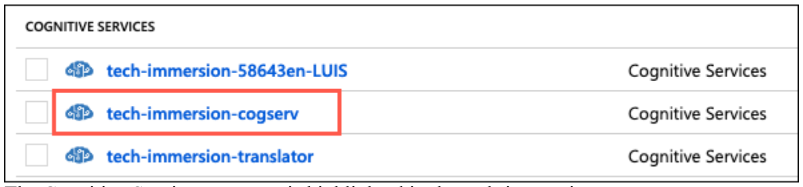
SECONDARY ADMIN KEY: DE89468AFCA42B6F911DC5F14CDADEA5

Azure search keys.

8. Return to Visual Studio and the `appsettings.json` file, and locate the `AZURE_SEARCH_SERVICE_KEY` setting. Paste the key you copied into the value for this setting. It should look similar to:

```
"AZURE_SEARCH_SERVICE_KEY": "B54E21DC49EB3B5A322190851F22BA8E"
```

9. Return to the Azure portal, and select the **Cognitive Services** account named **tech-immersion-cogserv** from the list of resources under the **tech-immersion-XXXXX** resource group (where XXXXX is the unique identifier assigned to you for this workshop).

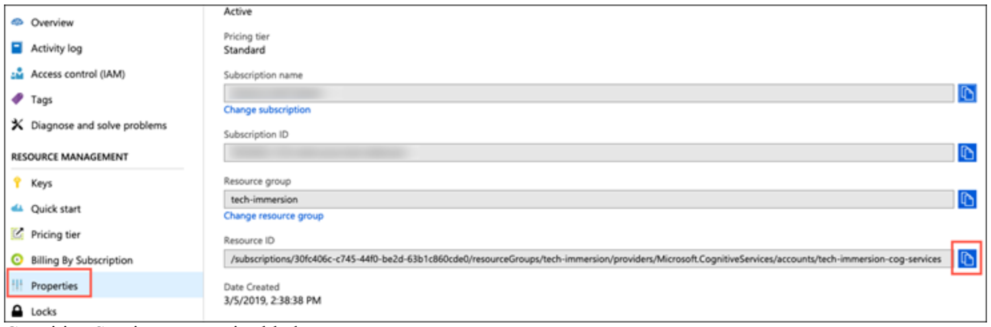


COGNITIVE SERVICES	
tech-immersion-58643en-LUIS	Cognitive Services
tech-immersion-cogserv	Cognitive Services
tech-immersion-translator	Cognitive Services

The Cognitive Services resource is highlighted in the tech-immersion resource group.

NOTE: There are three different Cognitive Services accounts in the resource group. You will want the one named **tech-immersion-cogserv** for this experience.

10. On the Cognitive Services blade, select **Properties** from the left-hand menu, and then copy the value for the **Resource ID** field by selecting the copy button to the right of the field.



Active

Pricing tier: Standard

Subscription name: [redacted]

Subscription ID: [redacted]

Resource group: tech-immersion

Resource ID: /subscriptions/30fc406c-c745-44f0-be2d-63b1c860cde0/resourceGroups/tech-immersion/providers/Microsoft.CognitiveServices/accounts/tech-immersion-cog-services

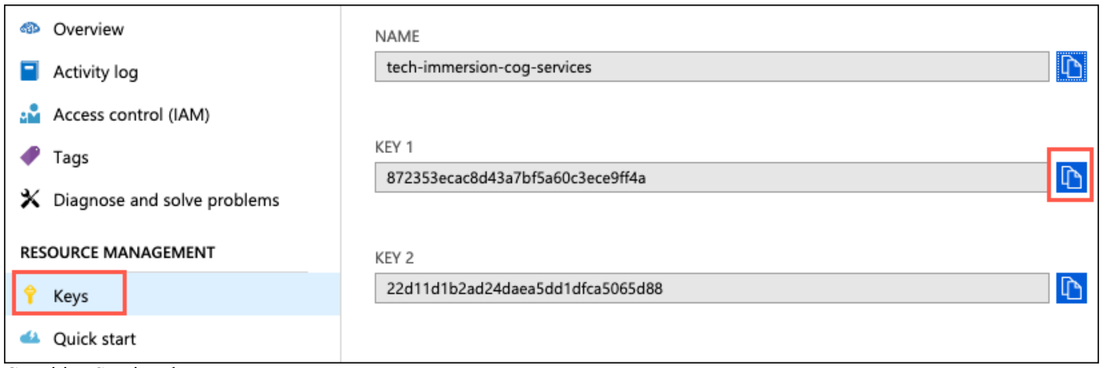
Date Created: 3/5/2019, 2:38:38 PM

Cognitive Services properties blade.

11. Return to Visual Studio and in the `appsettings.json` file, paste the **Resource ID** value into the `COGNITIVE_SERVICES_RESOURCE_ID` field value. It should look like the following:

```
"COGNITIVE_SERVICES_RESOURCE_ID": "/subscriptions/30fc406c-c745-44f0-be2d-63b1c860cde0/resourceGroups/tech-immersion/providers/Microsoft.CognitiveSer
```

12. Return to your Cognitive Services account in the Azure portal, and select **Keys** from the left-hand menu. On the Keys blade, copy the **Key 1** value by selecting the copy button to the right of the field.



NAME: tech-immersion-cog-services

KEY 1: 872353ecac8d43a7bf5a60c3ece9ff4a

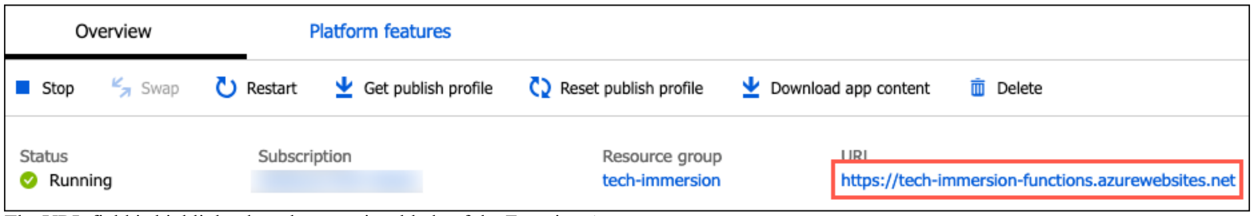
KEY 2: 22d11d1b2ad24daea5dd1dfca5065d88

Cognitive Services keys.

13. Return to Visual Studio and in the `appsettings.json` file, paste the Cognitive Services Key 1 value into the value for the `COGNITIVE_SERVICES_KEY` field. It will looks similar to the following:

```
"COGNITIVE_SERVICES_KEY": "872353ecac8d43a7bf5a60c3ece9ff4a"
```

14. The final settings you need to retrieve are those for your Azure Function App. In the Azure portal, navigate to your **ti-function-day2-XXXXX** Function App (where XXXXX is the unique identifier assigned to you for this workshop), and copy the **URL** on the Overview blade.



Status
Running

Subscription
[redacted]

Resource group
tech-immersion

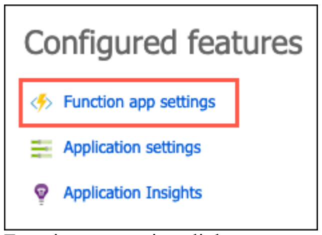
URL
https://tech-immersion-functions.azurewebsites.net

The URL field is highlighted on the overview blade of the Function App.

15. Return to the `appsettings.json` file for the `PipelineEnhancer` project in Visual Studio, and paste the value into the `AZURE_FUNCTION_APP_URL` setting. It should look similar to:

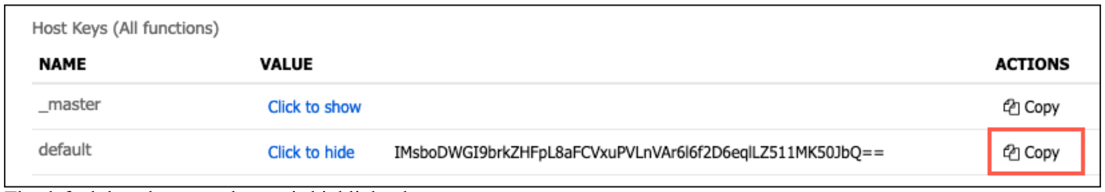
```
"AZURE_FUNCTION_APP_URL": "https://tech-immersion-functions.azurewebsites.net"
```

16. Back in the Azure portal, select **Function app settings** on the Overview blade.



Function app settings link.

17. On the Function app settings tab, select the **Copy** button next to the **default** Host Key.



Host Keys (All functions)

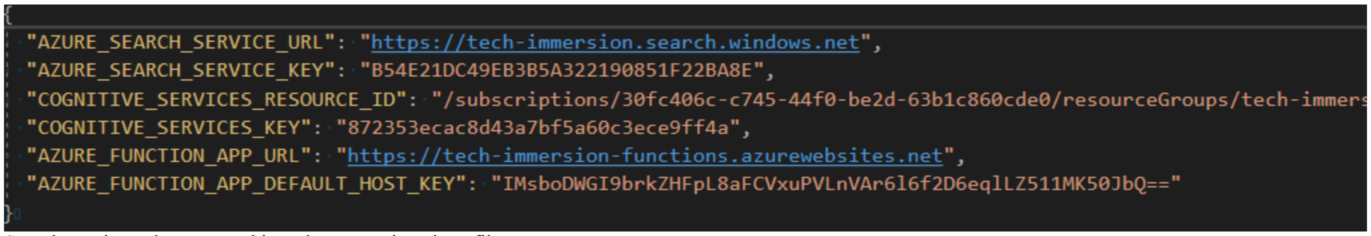
NAME	VALUE		ACTIONS
_master	Click to show		Copy
default	Click to hide	IMsboDWGI9brkZHFpL8aFCVxuPVLnVAr6l6f2D6eqlLZ511MK50JbQ==	Copy

The default host key copy button is highlighted.

18. Return to Visual Studio and the `appsettings.json` file for the `PipelineEnhancer` project, and paste default host key value into the `AZURE_FUNCTION_APP_DEFAULT_HOST_KEY` setting. It will look similar to the following:

```
"AZURE_FUNCTION_APP_DEFAULT_HOST_KEY": "IMsboDWGI9brkZHFpL8aFCVxuPVLnVAr6l6f2D6eqlLZ511MK50JbQ=="
```

19. Save `appsettings.json`. The file should now look like the following.



```
{
  "AZURE_SEARCH_SERVICE_URL": "https://tech-immersion.search.windows.net",
  "AZURE_SEARCH_SERVICE_KEY": "B54E21DC49EB3B5A322190851F22BA8E",
  "COGNITIVE_SERVICES_RESOURCE_ID": "/subscriptions/30fc406c-c745-44f0-be2d-63b1c860cde0/resourceGroups/tech-immers
  "COGNITIVE_SERVICES_KEY": "872353ecac8d43a7bf5a60c3ece9ff4a",
  "AZURE_FUNCTION_APP_URL": "https://tech-immersion-functions.azurewebsites.net",
  "AZURE_FUNCTION_APP_DEFAULT_HOST_KEY": "IMsboDWGI9brkZHFpL8aFCVxuPVLnVAr6l6f2D6eqlLZ511MK50JbQ=="
}
```

Search service values entered into the appsettings.json file.

20. You are now ready to move on to adding the enhancements to your pipeline. The create skillset API uses the following endpoint:

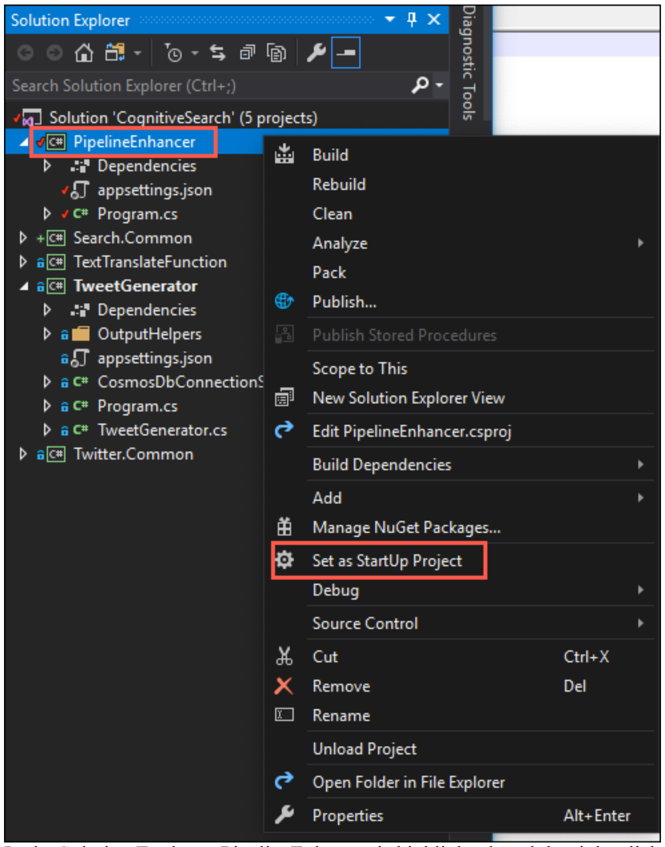
```
PUT https://[servicename].search.windows.net/skillsets/[skillset name]?api-version=2017-11-11-Preview
api-key: [admin key]
Content-Type: application/json
```

21. To add the sentiment analysis pre-built skill to your search pipeline, the `PipelineEnhancer` app will append the following JSON to the body of the Skillset you created through the Azure portal UI.

```
{
  "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",
  "inputs": [
    {
      "name": "text",
      "source": "/document/text"
    }
  ],
  "outputs": [
    {
      "name": "score",
      "targetName": "sentiment"
    }
  ]
}
```

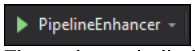
In the above JSON, the `inputs` specify the field in the source data document to send for analysis. The `outputs` section dictates that the `score` value returned by the Text Analytics in Cognitive Services should be output into a field named `sentiment` in the search results. This is sent into the REST API, along with the JSON from the previously built skillset to update or create the skillset.

22. To add sentiment analysis to your pipeline you will run the `PipelineEnhancer` project within the **CognitiveSearch** solution in Visual Studio. To run the project, right-click the `PipelineEnhancer` project in Visual Studio and select **Set as StartUp Project**.



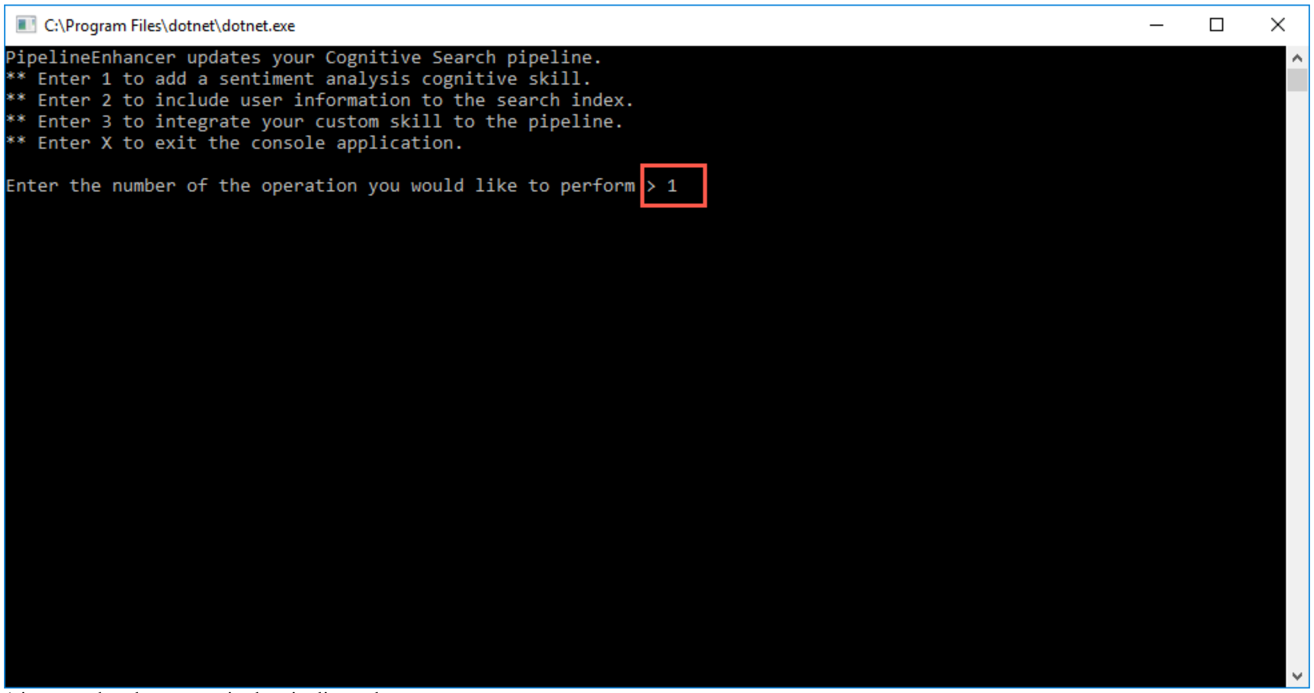
In the Solution Explorer, PipelineEnhancer is highlighted, and the right-click contenxt menu is displayed. Set as StartUp Project is highlighted in the context menu.

23. Now, select the run button on the Visual Studio toolbar, which is the one with a green arrow followed by the text "PipelineEnhancer."



The run button is displayed for PipelineEnhancer.

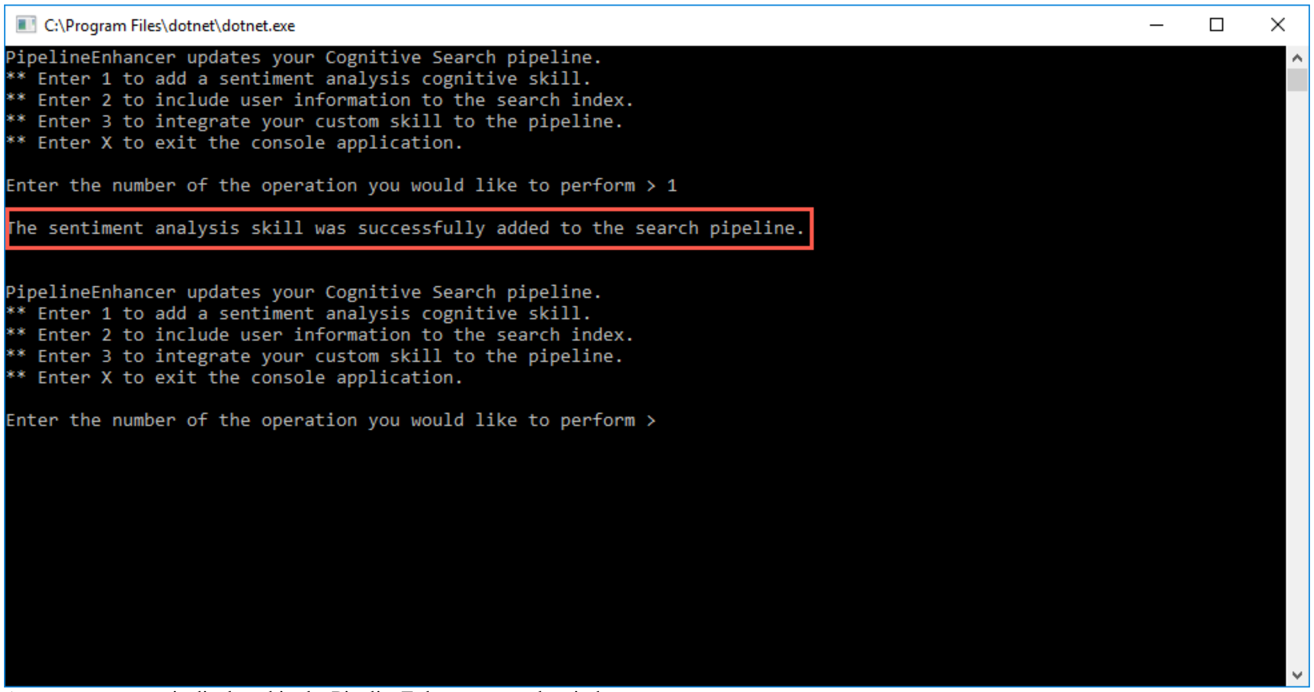
24. At the command prompt for the console app, enter **1** to incorporate the Sentiment cognitive skill to your pipeline.



```
PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 1
```

1 is entered at the prompt in the pipeline enhancer

25. When the console app completes you will receive a message stating that the sentiment analysis skill was successfully added.



```
C:\Program Files\dotnet\dotnet.exe
PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 1

The sentiment analysis skill was successfully added to the search pipeline.


PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform >
```

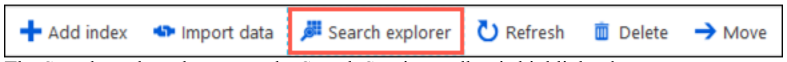
A success message is displayed in the PipelineEnhancer console window.

In addition to updating the Skillset JSON, the Indexer and Index were also be updated to include a new field named `sentiment`.

26. The process above deleted and recreated your Index, Indexer and Skillset, so you may need to select **Indexers** and the **tweet-indexer** on your Azure Search Service blade, and then select **Run** to force the Indexer to run against your tweet data again before attempting to run a search against the index in the next step.

    The Indexer **Run** screen does not refresh when the indexer has finished, so you can return to the overview blade of the Search service, and then select **Indexers**. Then, you can use the **Refresh** button on the Search service tool bar, next to Search explorer, to refresh the status. The **tweet-indexer** will display a status of **Success** when it finishes.

27. Return to your Azure Search service in the Azure portal and select **Search explorer** on the Search Service toolbar.



The Search explorer button on the Search Service toolbar is highlighted.

28. On the Search explorer tab, select **Search** and inspect one of the records in the search results.

```
{
  "@search.score": 1,
  "text": "@ContosoAuto Thank you for the amazing service at the Phoenix, AZ service center #Tesla #Model3",
  "rid": "YmpnZEFKMUhibTAtQUFBQUFBQUFBQT090",
  "people": [],
  "organizations": [],
  "locations": ["Phoenix", "AZ"],
  "keyphrases": [
    "amazing service",
    "AZ service center",
    "Phoenix",
    "Tesla"
  ],
  "language": "en",
  "sentiment": 0.994744300842285
}
```

    Notice the addition of the `sentiment` field to the bottom of the record. The value contained in this field is a numeric prediction made by a machine learning model about the sentiment of the contents of the `text` field in the tweet. Scores range from 0 to 1. Scores close to 1 indicate positive sentiment, and scores close to 0 indicate negative sentiment. Scores in the middle are considered to be neutral in the expression of sentiment. In the record above, the sentiment was determined to be very positive by the ML model.

29. Next, let's move on to adding user information from tweets to our search results. To accomplish this, you will use the delete index and delete indexer APIs to drop the index and indexer, and then use the create index and create indexer APIs to recreate them with a few fields added which point to the `user` object in the JSON tweet documents in Cosmos DB.

    For this workshop, we are going to add the user's `location` and `name` fields to our search index.

For the Index, the following JSON will be added to what was generated using the Azure portal UI:

```
{
  "name": "userLocation",
  "type": "Edm.String",
  "searchable": true,
  "filterable": true,
  "retrievable": true,
  "sortable": true,
    "facetable": false,
    "key": false,
    "indexAnalyzer": null,
    "searchAnalyzer": null,
    "analyzer": "standard.lucene",
    "synonymMaps": []
  },
  {
    "name": "userName",
    "type": "Edm.String",
    "searchable": true,
    "filterable": true,
    "retrievable": true,
    "sortable": true,
    "facetable": false,
    "key": false,
    "indexAnalyzer": null,
    "searchAnalyzer": null,
    "analyzer": "standard.lucene",
    "synonymMaps": []
  }
```

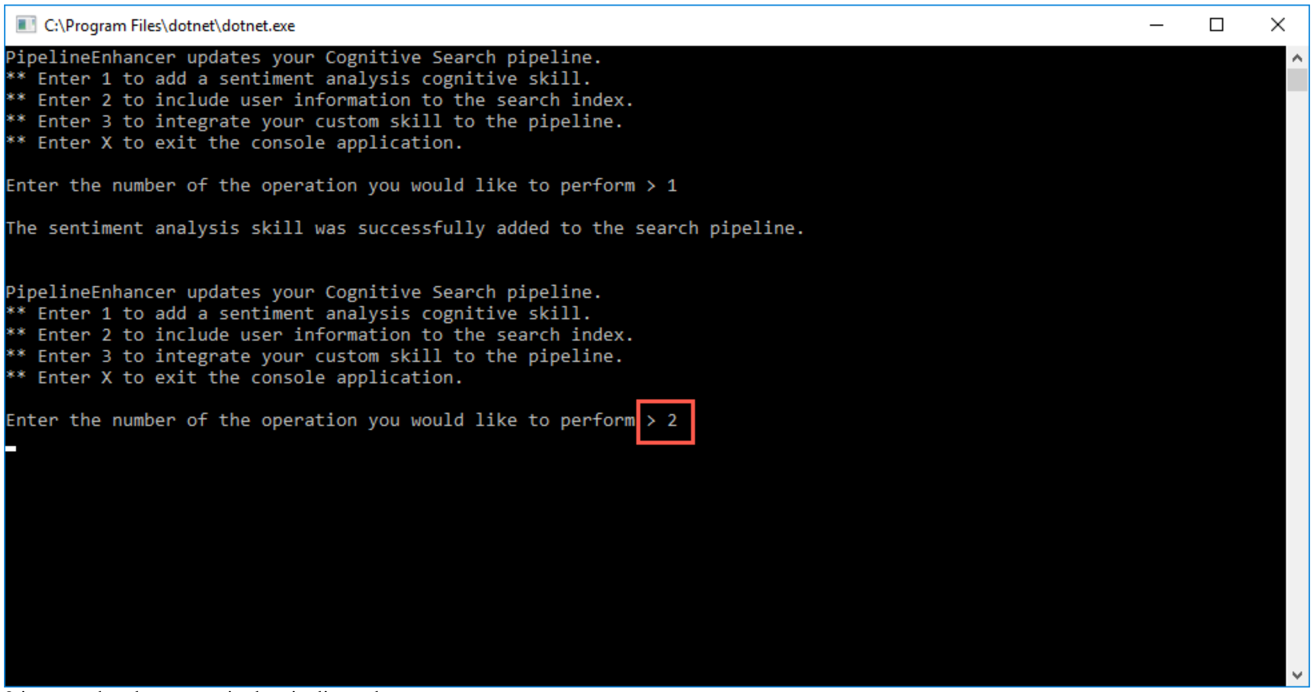
For each new field, `userLocation` and `userName`, the index properties are indicated by the additional fields.

For the Indexer, this JSON code will be added:

```
{
  "sourceFieldName": "/document/user/location",
  "targetFieldName": "userLocation",
  "mappingFunction": null
},
{
  "sourceFieldName": "/document/user/name",
  "targetFieldName": "userName",
  "mappingFunction": null
}
```

For each new field, `userLocation` and `userName`, an entry is added which maps the source field in the `tweet` document to an output field name.

30. Return to the `PipelineEnhancer` console app and enter **2** at the prompt to add the new user-related fields to your pipeline.



```
C:\Program Files\dotnet\dotnet.exe
PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 1

The sentiment analysis skill was successfully added to the search pipeline.


PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 2
```

2 is entered at the prompt in the pipeline enhancer

31. When the console app completes you will receive a success message. Navigate to your Azure Search service in the Azure portal and select **Search explorer**.

    As before, the console app deleted and recreated your Index and Indexer. You may need to select **Indexers** and the **tweet-indexer**, and then select **Run** to force the Indexer to run against your tweet data again before attempting to run a search against the index.

32. On the Search explorer blade, select **Search** and inspect one of the records in the search results.

```
{
  "@search.score": 1,
  "text": "@ContosoAuto Thank you for the amazing service at the Phoenix, AZ service center #Tesla #Model3",
  "rid": "YmpnZEFKMUhibTAtQUFBQUFBQUFBQT090",
  "people": [],
  "organizations": [],
  "locations": ["the Phoenix", "AZ", "Tesla"],
  "keyphrases": [
    "amazing service",
    "AZ service center",
    "Phoenix",
    "Tesla"
  ],
  "language": "en",
  "sentiment": 0.994744300842285,
  "userLocation": "Sacramento, CA",
  "userName": "Ellis Maginnis"
}
```

Note the addition of the `userLocation` and `userName` fields to the bottom of the record. The values in these fields are pulled directly from the `tweet` documents being indexed from Cosmos DB, but were not accessible using the Azure portal UI. Using the REST APIs, you are able to dig deeper into nested JSON documents, and index a richer set of the data stored in your documents.

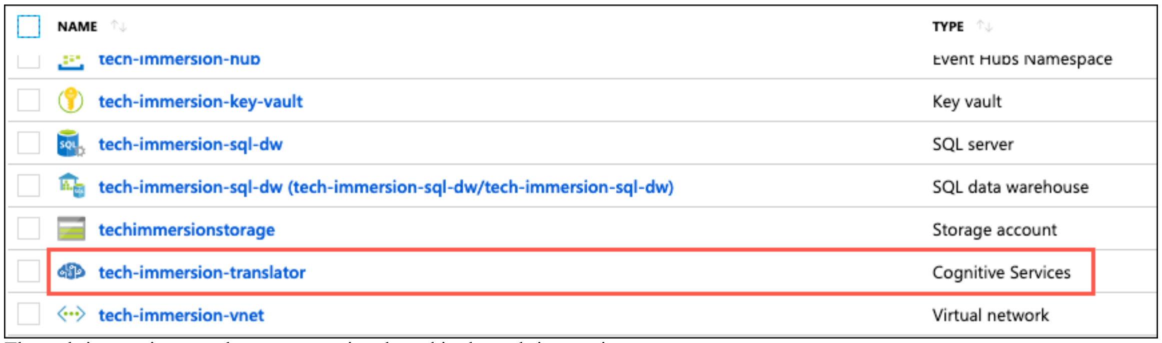
33. Return to the `PipelineEnhancer` console app, and enter **X** at the command prompt to exit the console application.

Task 4: Deploy a custom skill

In addition to predefined Cognitive skills, you can also integrate custom skills into your Cognitive Search pipeline. In this task, you will create a custom skill using the Text Translate API. The example uses an Azure Function to wrap the Translate Text API so that it implements the custom skill interface.

While this example uses an Azure Function to host a web API, you can use any approach as long as it meets the interface requirements for a cognitive skill. Azure Functions, however, make it very easy to create custom skills.

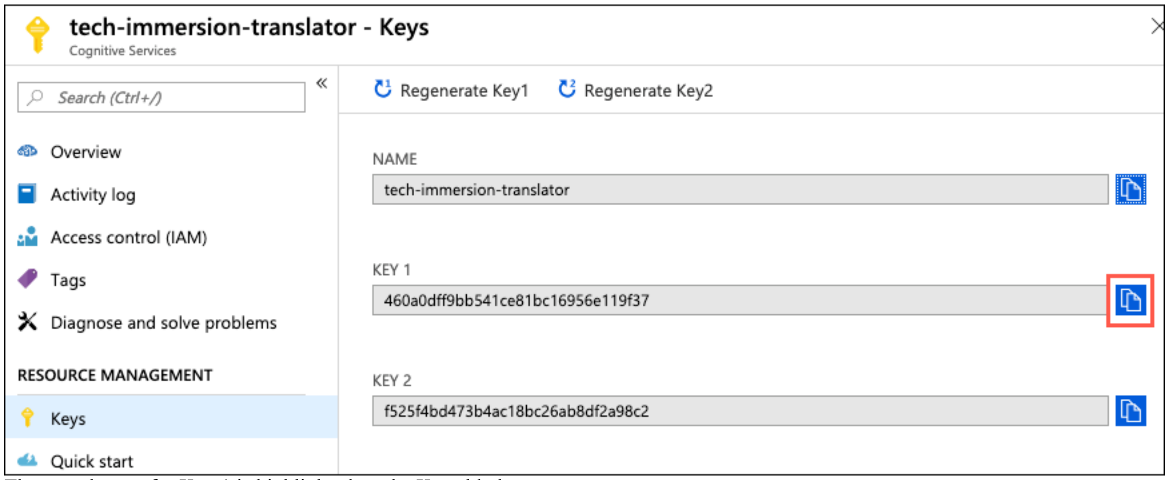
1. Since you will be using the Translator Text API, you need retrieve your API key from the Azure portal, so it can be used by the Azure Function. As you've done with previous resources, select the **tech-immersion-translator** Cognitive Services resource from the **tech-immersion-XXXXX** resource group (where XXXXX is the unique identifier assigned to you for this workshop).



NAME	TYPE
tech-immersion-hub	Event Hubs Namespace
tech-immersion-key-vault	Key vault
tech-immersion-sql-dw	SQL server
tech-immersion-sql-dw (tech-immersion-sql-dw/tech-immersion-sql-dw)	SQL data warehouse
techimmersionstorage	Storage account
tech-immersion-translator	Cognitive Services
tech-immersion-vnet	Virtual network

The tech-immersion-translator resource is selected in the tech-immersion resource group.

2. Once on your Cognitive Services blade, select **Keys** from the left-hand menu, and then select the Copy button next to the value for **Key 1**.



tech-immersion-translator - Keys
Cognitive Services

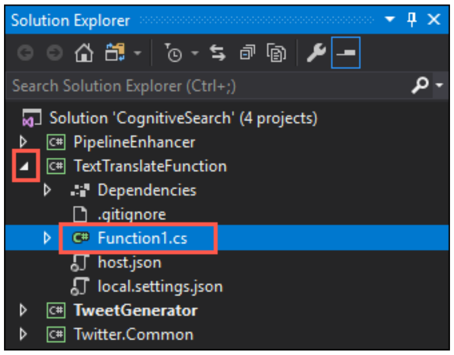
NAME
tech-immersion-translator

KEY 1
460a0dff9bb541ce81bc16956e119f37

KEY 2
f525f4bd473b4ac18bc26ab8df2a98c2

The copy button for Key 1 is highlighted on the Keys blade.

3. Return to Visual Studio, and in the Solution Explorer on the right-hand side, expand the `TextTranslateFunction` project and then double-click `Function1.cs` to open the file.



The `Function1.cs` file is highlighted in the Visual Studio Solution Explorer.

4. In the `Function1.cs` file, locate the line of code (line 22) that looks like the following:

```
// NOTE: Replace this example key with a valid subscription key.
static string key = "<enter your api key here>";
```

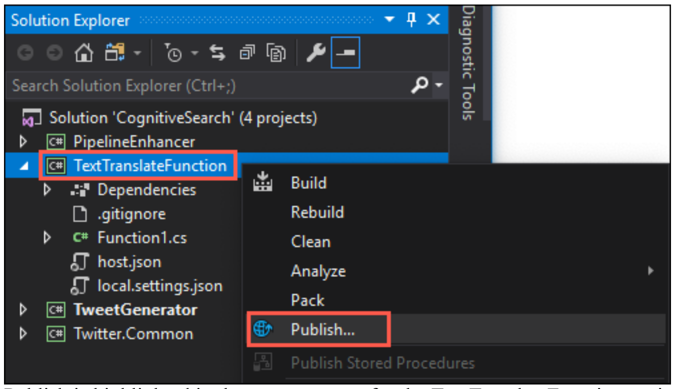
5. Replace the value with double-quotes (`<enter your api key here>`) with the API key you copied for the Translator Cognitive Service. The line should now look similar to this:

```
static string key = "460a0dff9bb541ce81bc16956e119f37";
```

6. Save `Function1.cs`.

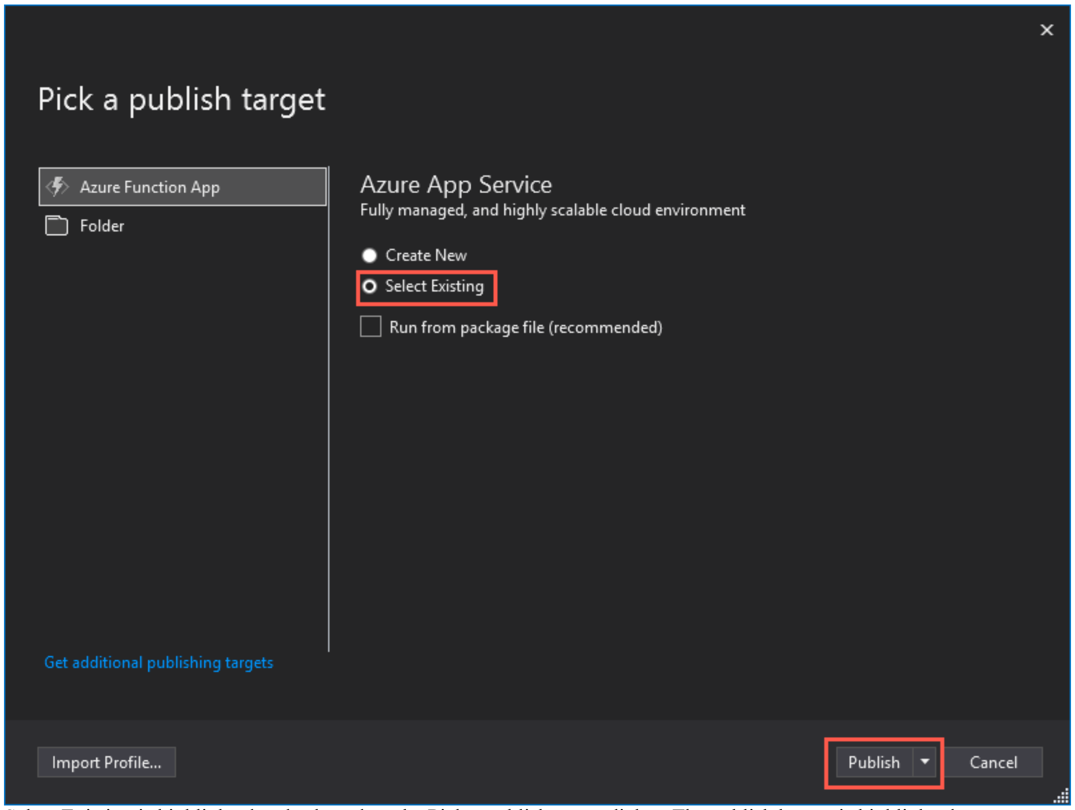
   Take a few minutes to look over the code in `Function1.cs`. This file defines the Function that will be deployed to your Azure Function App. The function code receives the tweet text and the detected language from the custom skill. Only those tweets with a language determined to not be English will be translated by the Translator Text API. For all English tweets, the original text is returned unmodified.

7. You are now ready to deploy the function into your Azure Function App. Right-click the `TextTranslateFunction` project, and select **Publish** from the context menu.



Publish is highlighted in the context menu for the TextTranslateFunction project.

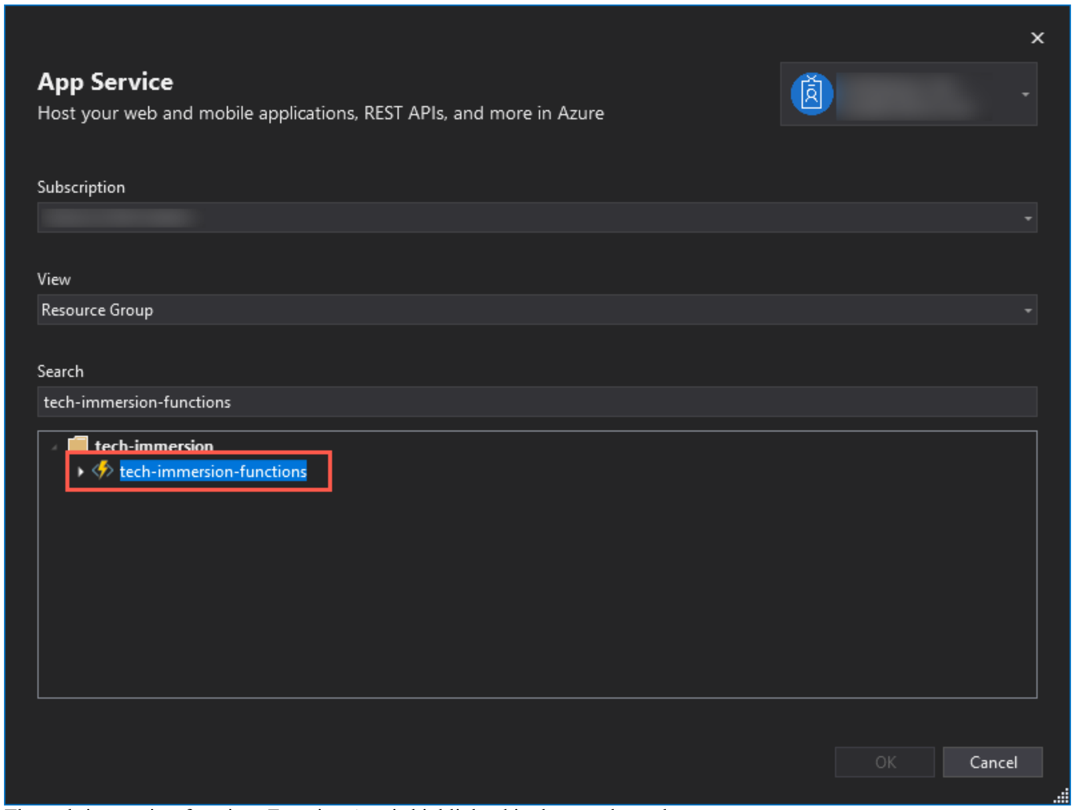
8. On the **Pick a publish target** dialog, choose **Select Existing** and select **Publish**.



Pick a publish target
Azure Function App
Folder
Azure App Service
Fully managed, and highly scalable cloud environment
Create New
Select Existing
Run from package file (recommended)
Get additional publishing targets
Import Profile...
Publish
Cancel

Select Existing is highlighted and selected on the Pick a publish target dialog. The publish button is highlighted.

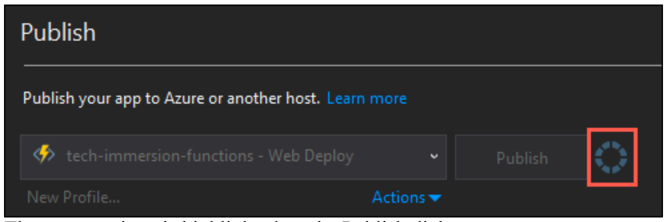
9. On the **App service** dialog, select your Subscription and locate the **tech-immersion-XXXXX** resource group (where XXXXX is the unique identifier assigned to you for this workshop) in the list. Select the Function App named **ti-function-day2-XXXXX** from the list of resources (where XXXXX is the unique identifier assigned to you for this workshop).



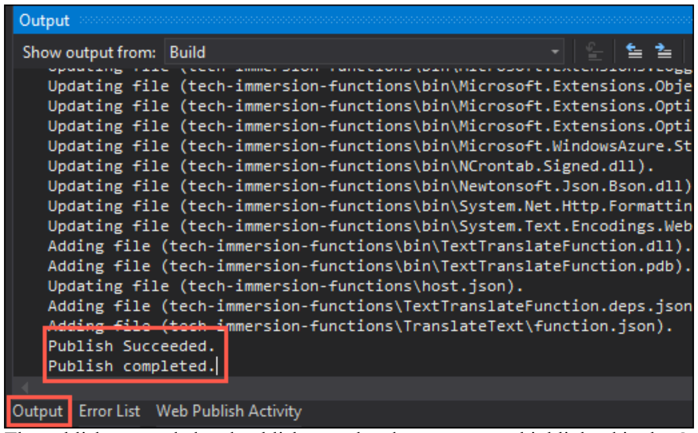
The tech-immersion-functions Function App is highlighted in the search results.

You may need to enter the credentials of the account you are using for this workshop before you can see any resources for your subscription.

10. Select **OK**. This will start the publish process. You will see an animated progress icon next to the Publish button while the deployment is in progress.


Publish
Publish your app to Azure or another host. Learn more
tech-immersion-functions - Web Deploy
Publish
New Profile...
Actions
The progress icon is highlighted on the Publish dialog.

11. When the publish is complete, you can open the **Output** window at the bottom left-hand corner of the Visual Studio window to observe the results. You should see messages that the **Publish Succeeded** and **Publish completed**.



```
Updating file (tech-immersion-functions\bin\Microsoft.Extensions.Obje
Updating file (tech-immersion-functions\bin\Microsoft.Extensions.Opti
Updating file (tech-immersion-functions\bin\Microsoft.Extensions.Opti
Updating file (tech-immersion-functions\bin\Microsoft.WindowsAzure.St
Updating file (tech-immersion-functions\bin\NCrontab.Signed.dll).
Updating file (tech-immersion-functions\bin\Newtonsoft.Json.Bson.dll)
Updating file (tech-immersion-functions\bin\System.Net.Http.Formattin
Updating file (tech-immersion-functions\bin\System.Text.Encodings.Web
Adding file (tech-immersion-functions\bin\TextTranslateFunction.dll).
Adding file (tech-immersion-functions\bin\TextTranslateFunction.pdb).
Updating file (tech-immersion-functions\host.json).
Adding file (tech-immersion-functions\TextTranslateFunction.deps.json
Adding file (tech-immersion-functions\TranslateText\function.json).
Publish Succeeded.
Publish completed.
```

The publish succeeded and publish completed messages are highlighted in the Output window.

If the Output window is not visible, you can display it by selecting the **View** menu in Visual Studio, and then selecting **Output**.

Task 5: Integrate custom skill into pipeline

With the Function App now in place to support your Text Translator custom skill, you are ready to integrate a new custom skill using the custom skill interface, which is accessed by adding a `WebApiSkill` to the skillset. In this task, you will add a custom skill into the Cognitive Search pipeline you created previously.

Another change that will be made as the new custom skill is added is to update all the previously added predefined skills, except the language detection skill, to use the new `textTranslated` field as input. This will allow all the Cognitive Skills to run against the English language version of the text.

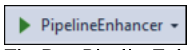
1. To add the custom skill to your search pipeline the following JSON will be appended to the body of the Skillset you built with the UI by the `PipelineEnhancer` app.

```
{
  "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
  "description": "Custom translator skill",
  "uri": "http://<your-function-app-name>.azurewebsites.net/api/Translate?code=<default-host-key>",
  "batchSize": 1,
  "context": "/document",
  "inputs": [
    {
      "name": "text",
      "source": "/document/text"
    },
    {
      "name": "language",
      "source": "/document/language"
    }
  ],
  "outputs": [
    {
      "name": "text",
      "targetName": "textTranslated"
    }
  ]
}
```

   In the above JSON, the `inputs` specify the field in the source data document to send for analysis. The `outputs` section dictates that the `text` value returned by your Function App should output into a field named `textTranslated` in the search results. This is sent into the REST API, along with the JSON from the previously built skillset to update or create the skillset.

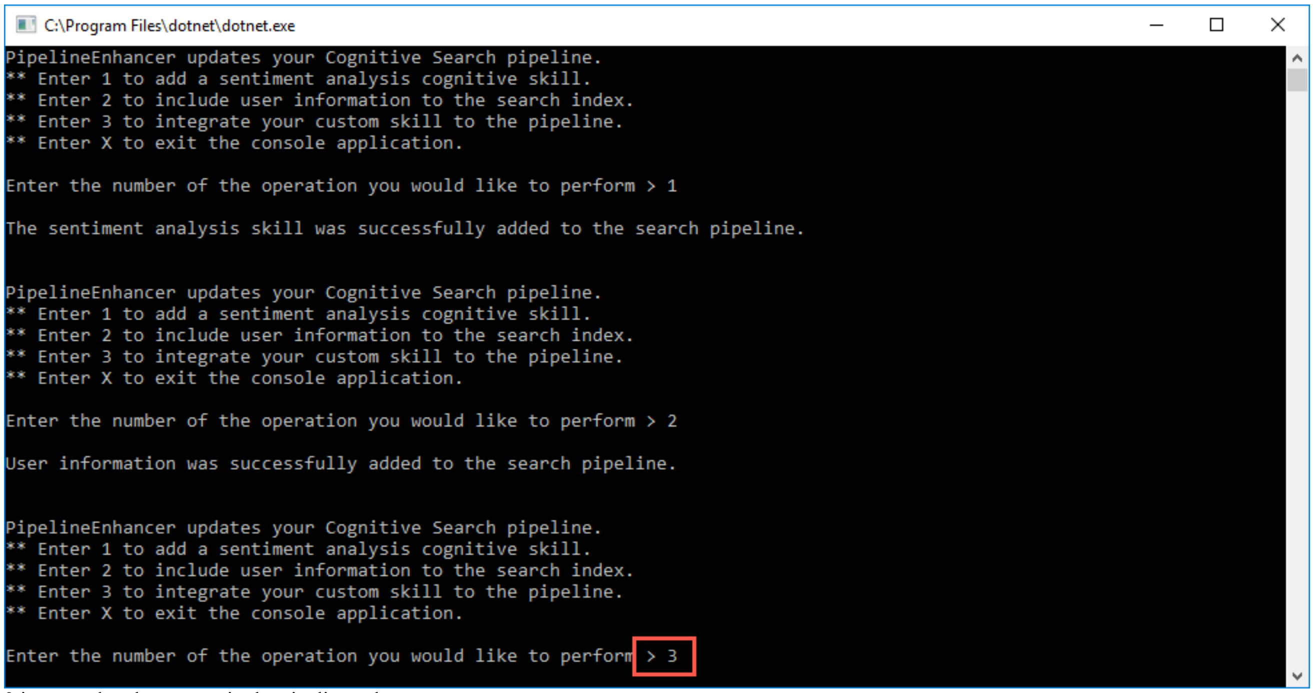
2. Note that there are two values within the `uri` field of your custom skill that will need to be supplied to your custom skill, so it can connect to your Function app: the Function App name and default host code. You already added these values to your `appsettings.json` file.

3. Relaunch the `PipelineEnhancer` console app by selecting the **Run** button on the Visual Studio toolbar.



   The Run PipelineEnhancer button from the Visual Studio toolbar is displayed.

4. In `PipelineEnhancer` console window and enter **3** at the prompt. This step will incorporate the custom cognitive skill into your pipeline.

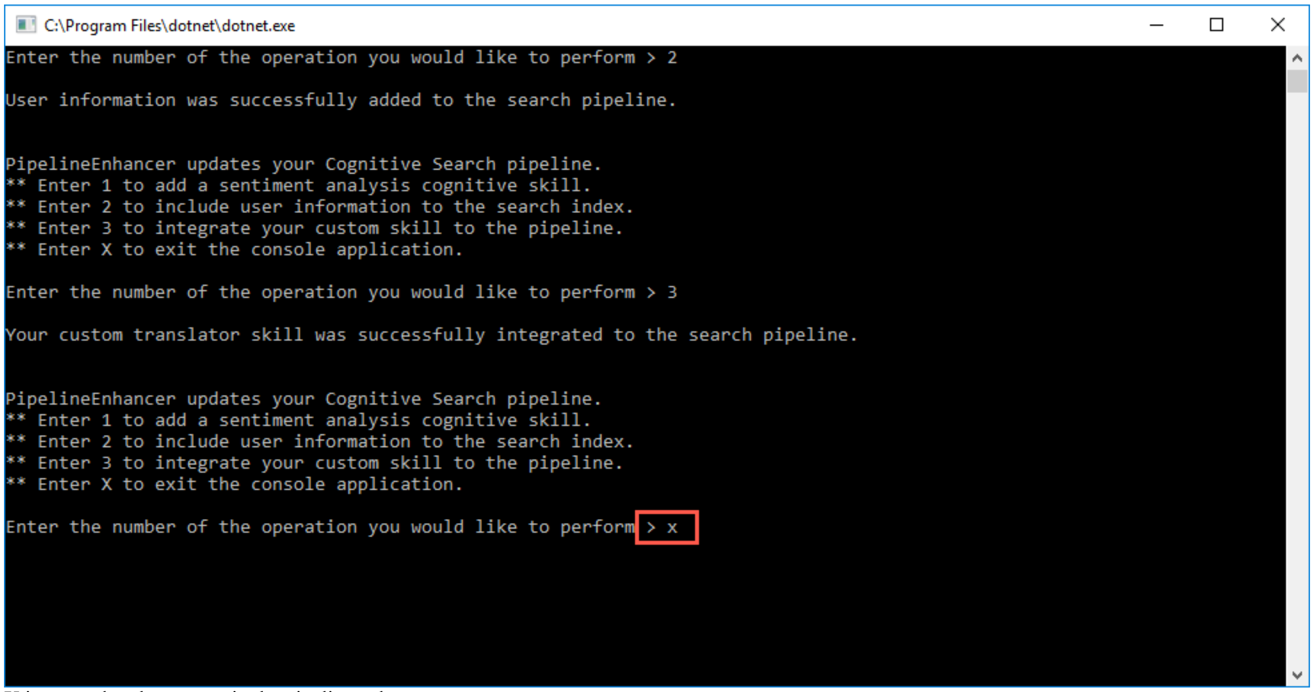


```
PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 1

The sentiment analysis skill was successfully added to the search pipeline.


PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 2

User information was successfully added to the search pipeline.


PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 3
```

   3 is entered at the prompt in the pipeline enhancer

   In addition to updating the Skillset JSON, the Indexer and Index were also be updated to include a new field named `textTranslated`.

5. When the console app completes, enter **X** at the prompt to close the application.



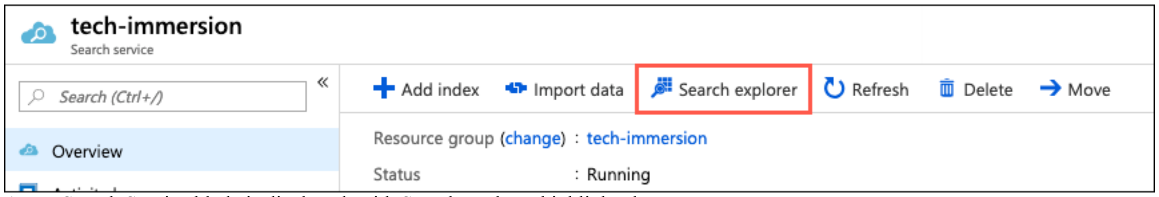
```
C:\Program Files\dotnet\dotnet.exe
Enter the number of the operation you would like to perform > 2

User information was successfully added to the search pipeline.


PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > 3

Your custom translator skill was successfully integrated to the search pipeline.


PipelineEnhancer updates your Cognitive Search pipeline.
** Enter 1 to add a sentiment analysis cognitive skill.
** Enter 2 to include user information to the search index.
** Enter 3 to integrate your custom skill to the pipeline.
** Enter X to exit the console application.

Enter the number of the operation you would like to perform > x
```

X is entered at the prompt in the pipeline enhancer

Task 6: Run indexer and query data

In this task, you will run your Search Indexer and then query data in the index.

1. In the Azure portal, navigate to your **Search service** resource and select **Search explorer** in the toolbar on the overview blade.

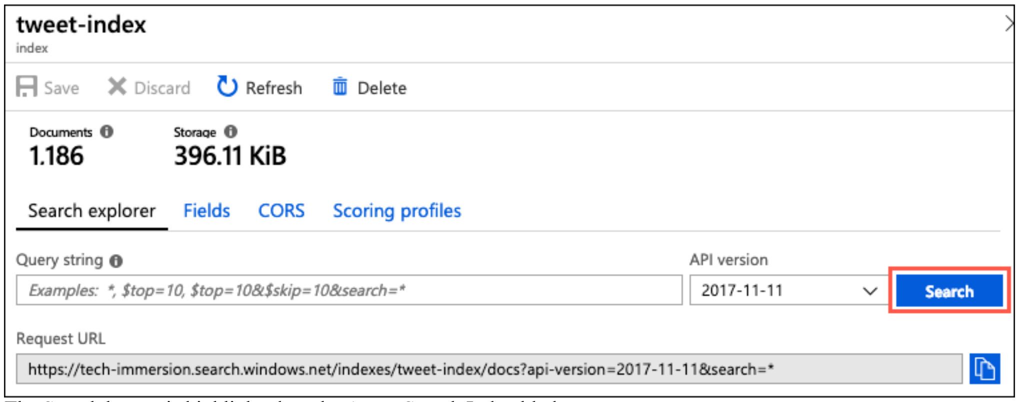


Resource group (change) : tech-immersion
Status : Running

Azure Search Service blade is displayed, with Search explorer highlighted.

The last step of Task 5 deleted and recreated your Index, Indexer and Skillset. If you don't see any search results, you may need to select **Indexers** on the Overview blade and then **tweet-indexer**. Select **Run** to force the Indexer to run against your tweet data again before attempting to run a search against the index.

2. On the Search explorer tab, select **Search** and observe the results.



Documents 1.186
Storage 396.11 KiB

Query string: Examples: *, $top=10, $top=10&$skip=10&search=*
API version: 2017-11-11

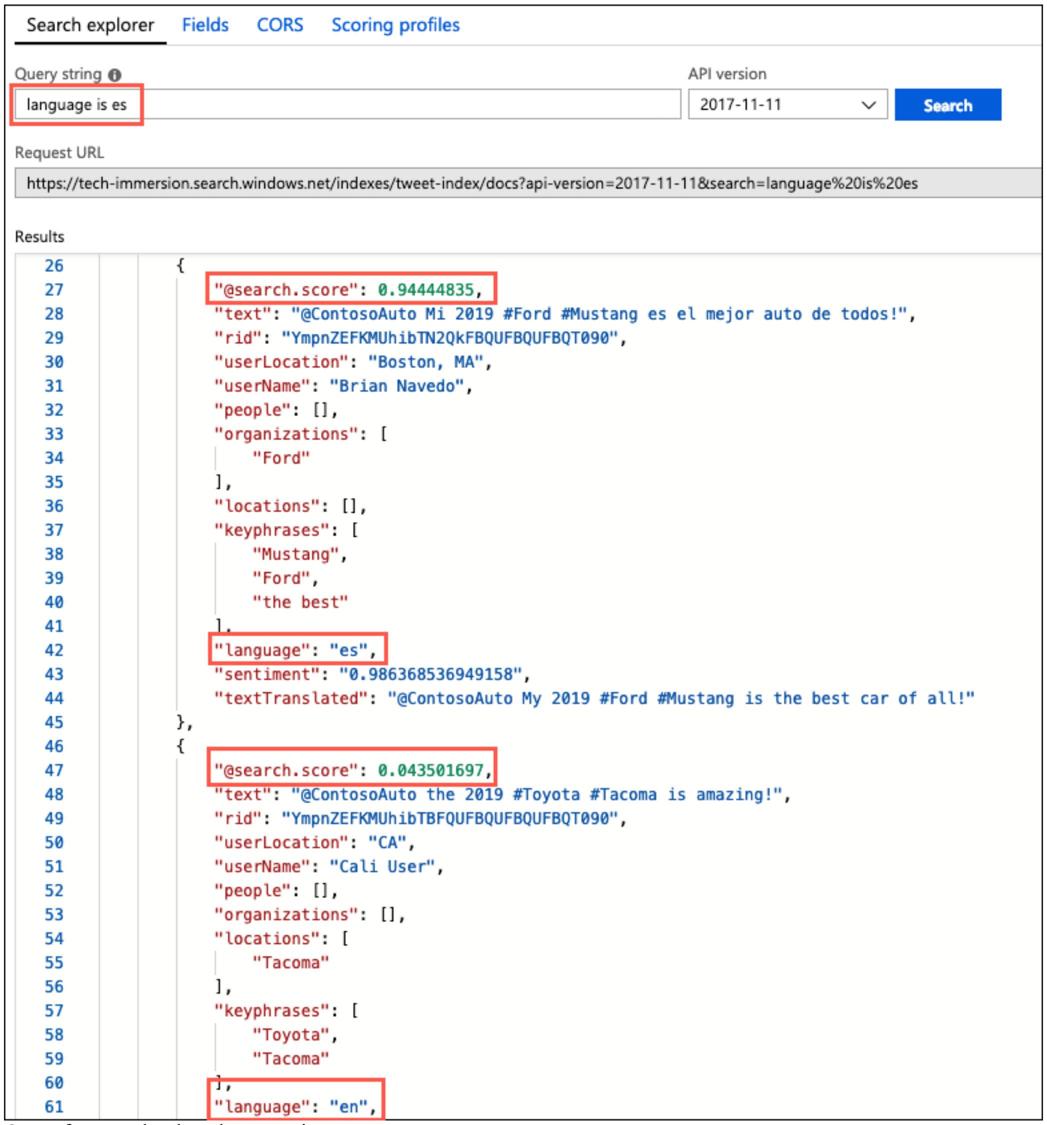
Request URL: https://tech-immersion.search.windows.net/indexes/tweet-index/docs?api-version=2017-11-11&search=*

The Search button is highlighted on the Azure Search Index blade.

3. You can now play around with the search functionality. Below, you will enter a few queries that simulate what an application user may enter. These queries tend to resemble natural language, so we will start there. In the Query string field, enter "language is es", and observe the results. This will return one of the records in the search results where the language was detected as Spanish.

```
{
  "@search.score": 0.94444835,
  "text": "@ContosoAuto Mi 2019 #Ford #Mustang es el mejor auto de todos!",
  "rid": "YmpnZEFKMUhibTN2QkFBQUFBQUFBQT090",
  "userLocation": "Boston, MA",
  "userName": "Brian Navedo",
  "people": [],
  "organizations": ["Ford"],
    "locations": [],
    "keyphrases": ["Mustang", "Ford", "the best"],
    "language": "es",
    "sentiment": 0.986368536949158,
    "textTranslated": "@ContosoAuto My 2019 #Ford #Mustang is the best car of all!"
}
```

Notice the addition of the `textTranslated` field to the bottom of the record. This contains the English translation of the contents of the `text` field in the tweet. You will also notice that the `keyphrases` field contains only English words and phrases. This is because the `KeyPhraseExtractionSkill` was pointed to the new `textTranslated` field with the latest updates to the search pipeline.



Query string: language is es

API version: 2017-11-11

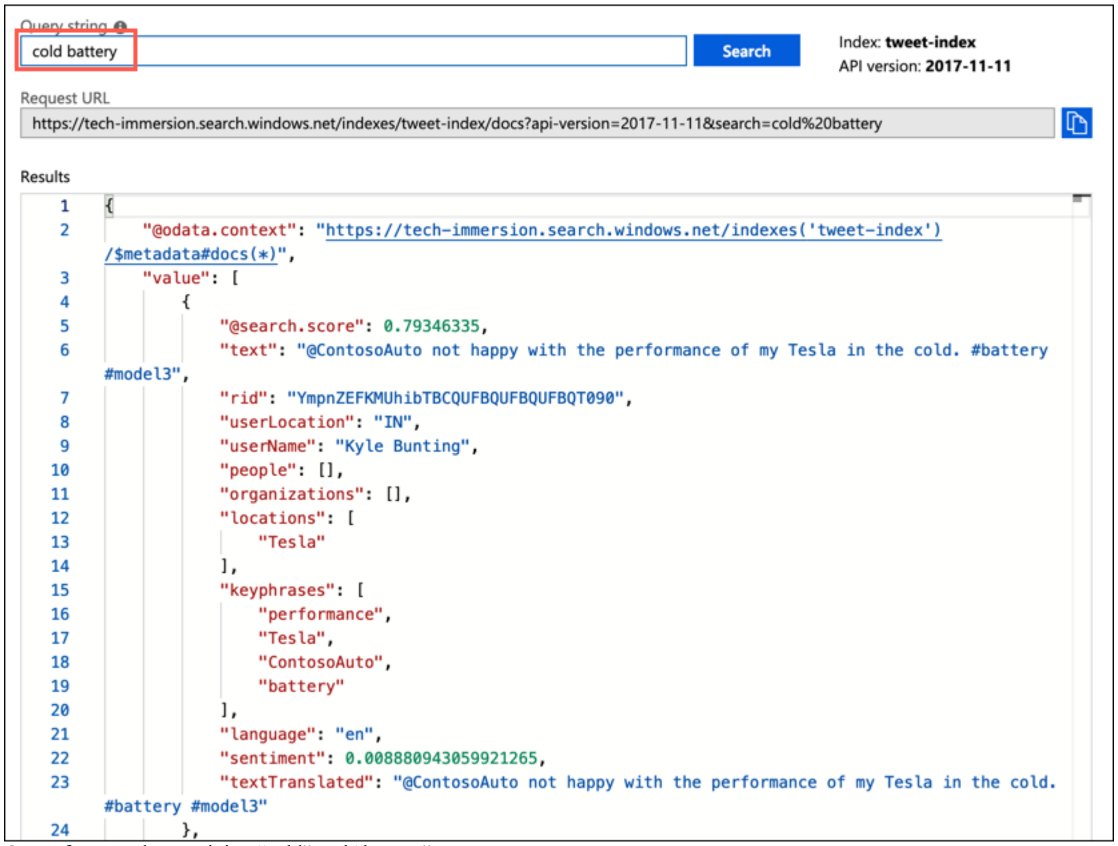
Request URL: https://tech-immersion.search.windows.net/indexes/tweet-index/docs?api-version=2017-11-11&search=language%20is%20es

```
        {
            "@search.score": 0.94444835,
            "text": "@ContosoAuto Mi 2019 #Ford #Mustang es el mejor auto de todos!",
            "rid": "YmpnZEFKMUhibTN2QkFBQUFBQUFBQT090",
            "userLocation": "Boston, MA",
            "userName": "Brian Navedo",
            "people": [],
            "organizations": [
                "Ford"
            ],
            "locations": [],
            "keyphrases": [
                "Mustang",
                "Ford",
                "the best"
            ],
            "language": "es",
            "sentiment": "0.986368536949158",
            "textTranslated": "@ContosoAuto My 2019 #Ford #Mustang is the best car of all!"
        },
        {
            "@search.score": 0.043501697,
            "text": "@ContosoAuto the 2019 #Toyota #Tacoma is amazing!",
            "rid": "YmpnZEFKMUhibTBFQUFBQUFBQUFBQT090",
            "userLocation": "CA",
            "userName": "Cali User",
            "people": [],
            "organizations": [],
            "locations": [
                "Tacoma"
            ],
            "keyphrases": [
                "Toyota",
                "Tacoma"
            ],
            "language": "en",
```

Query for records where language is es.

In addition, notice that each record returned contains a `@search.score` field with a numeric value. This value indicates the confidence of the match with the search query. The higher the value, the more likely it is to be a match for the query. If you scroll down past the records where `"language"="es"`, you will see the search score values drop, as those records don't match the query.

4. Try another search, such as "cold battery" and observe the results.



cold battery

Index: **tweet-index**
API version: **2017-11-11**

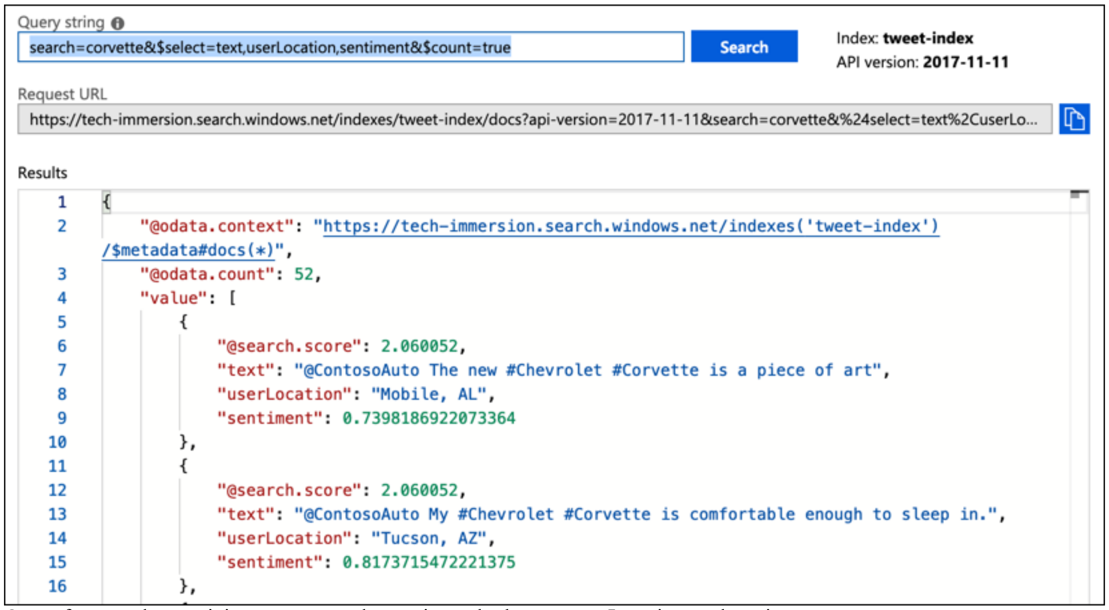
https://tech-immersion.search.windows.net/indexes/tweet-index/docs?api-version=2017-11-11&search=cold%20battery

```
{
    "@odata.context": "https://tech-immersion.search.windows.net/indexes('tweet-index')
/$metadata#docs(*)",
    "value": [
        {
            "@search.score": 0.79346335,
            "text": "@ContosoAuto not happy with the performance of my Tesla in the cold. #battery
#model3",
            "rid": "YmpnZEFKMUhibTBCQUFBQUFBQUFBQT090",
            "userLocation": "IN",
            "userName": "Kyle Bunting",
            "people": [],
            "organizations": [],
            "locations": [
                "Tesla"
            ],
            "keyphrases": [
                "performance",
                "Tesla",
                "ContosoAuto",
                "battery"
            ],
            "language": "en",
            "sentiment": 0.008880943059921265,
            "textTranslated": "@ContosoAuto not happy with the performance of my Tesla in the cold.
#battery #model3"
        },
```

Query for records containing “cold” and “battery”

5. Now, let’s try a slightly more advanced search. We want to look for only records that mention a “corvette”, and we only want to retrieve the `text`, `userLocation`, and `sentiment` fields in our results. Paste the following into the Query string box, and select **Search**:

```
search=corvette&$select=text,userLocation,sentiment&$count=true
```



search=corvette&$select=text,userLocation,sentiment&$count=true

Index: **tweet-index**
API version: **2017-11-11**

https://tech-immersion.search.windows.net/indexes/tweet-index/docs?api-version=2017-11-11&search=corvette&%24select=text%2CuserLo...

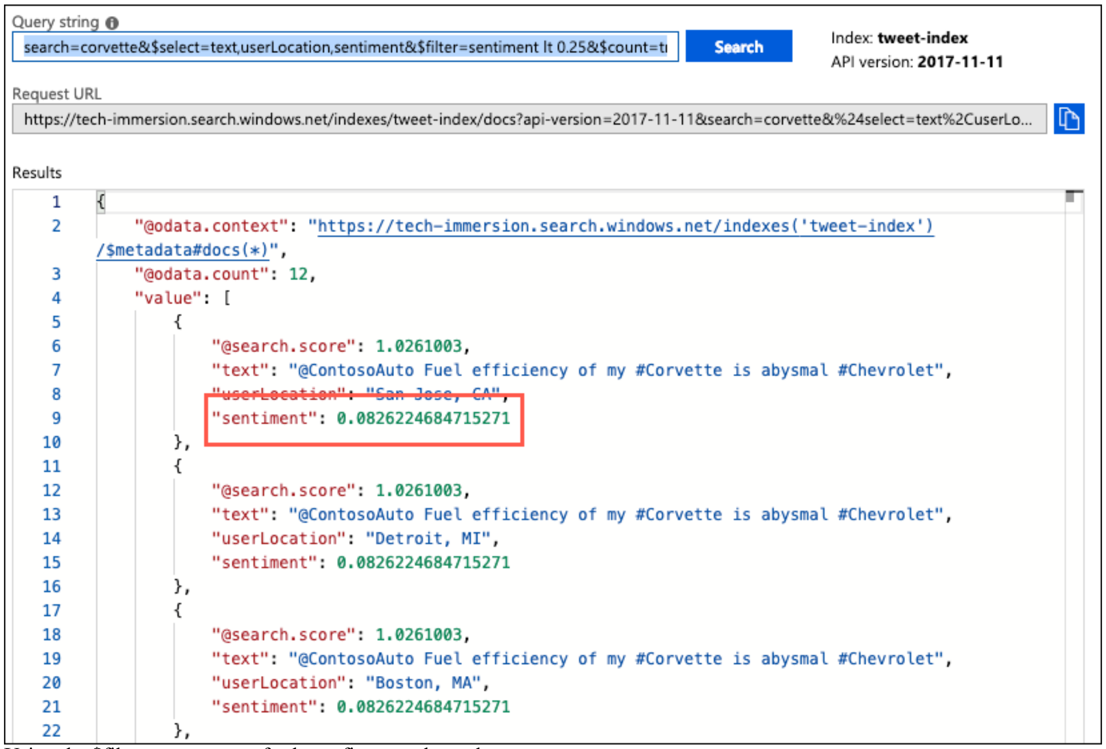
```
{
    "@odata.context": "https://tech-immersion.search.windows.net/indexes('tweet-index')
/$metadata#docs(*)",
    "@odata.count": 52,
    "value": [
        {
            "@search.score": 2.060052,
            "text": "@ContosoAuto The new #Chevrolet #Corvette is a piece of art",
            "userLocation": "Mobile, AL",
            "sentiment": 0.7398186922073364
        },
        {
            "@search.score": 2.060052,
            "text": "@ContosoAuto My #Chevrolet #Corvette is comfortable enough to sleep in.",
            "userLocation": "Tucson, AZ",
            "sentiment": 0.8173715472221375
        },
```

Query for records containing corvette, and returning only the text, userLocation, and sentiment.

Add `$select` limits results to the explicitly named fields for more readable output in Search explorer.

6. You can take that query even further by adding in the `$filter` parameter. Use the `$filter` parameter when you want to specify precise criteria rather than free text search. This example searches for sentiment less than 0.25, so we can target tweets with negative sentiment in the search results.

```
search=corvette&$select=text,userLocation,sentiment&$filter=sentiment lt 0.25&$count=true
```



```
search=corvette&$select=text,userLocation,sentiment&$filter=sentiment lt 0.25&$count=tı
```

Index: **tweet-index**
API version: **2017-11-11**

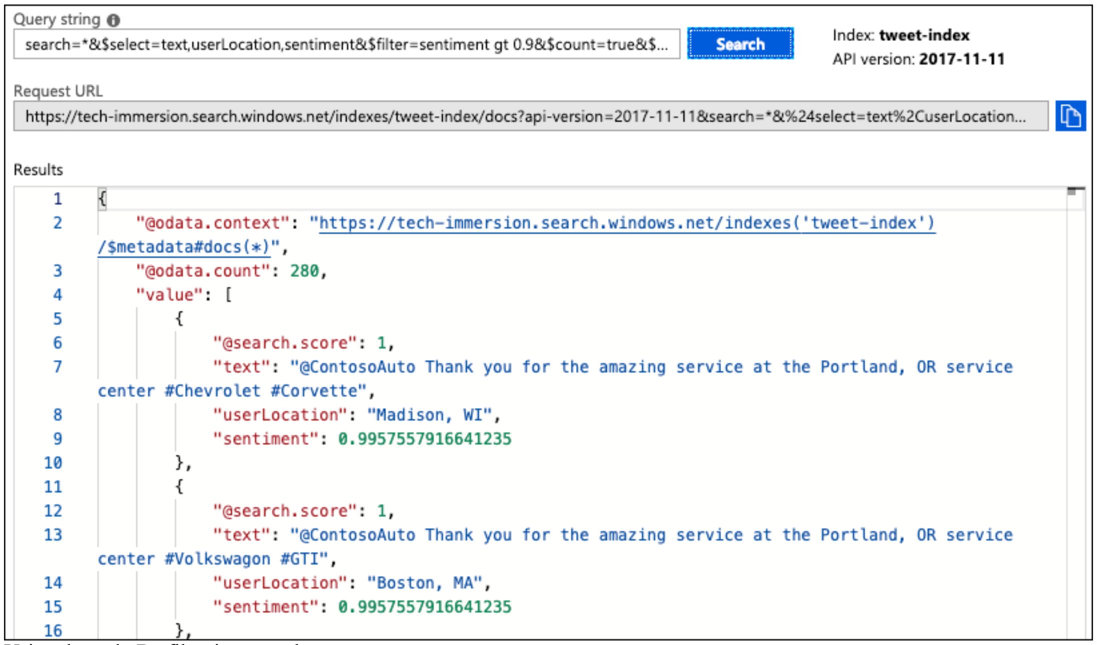
```
https://tech-immersion.search.windows.net/indexes/tweet-index/docs?api-version=2017-11-11&search=corvette&%24select=text%2CuserLo...
```

```
{
    "@odata.context": "https://tech-immersion.search.windows.net/indexes('tweet-index')
/$metadata#docs(*)",
    "@odata.count": 12,
    "value": [
        {
            "@search.score": 1.0261003,
            "text": "@ContosoAuto Fuel efficiency of my #Corvette is abysmal #Chevrolet",
            "userLocation": "San Jose, CA",
            "sentiment": 0.0826224684715271
        },
        {
            "@search.score": 1.0261003,
            "text": "@ContosoAuto Fuel efficiency of my #Corvette is abysmal #Chevrolet",
            "userLocation": "Detroit, MI",
            "sentiment": 0.0826224684715271
        },
        {
            "@search.score": 1.0261003,
            "text": "@ContosoAuto Fuel efficiency of my #Corvette is abysmal #Chevrolet",
            "userLocation": "Boston, MA",
            "sentiment": 0.0826224684715271
        },
```

Using the $filter parameter to further refine search results.

7. The final query we will run adds the `$orderBy` parameter, which allow you to specify the sort order of your results. In this case, let's search for records where the sentiment is the highest, filtering for records where the sentiment is greater than 0.9, and ordering the results in descending order.

```
search=*&$select=text,userLocation,sentiment&$filter=sentiment gt 0.9&$count=true&$orderby=sentiment desc
```



```
search=*&$select=text,userLocation,sentiment&$filter=sentiment gt 0.9&$count=true&$...
```

Index: **tweet-index**
API version: **2017-11-11**

```
https://tech-immersion.search.windows.net/indexes/tweet-index/docs?api-version=2017-11-11&search=*&%24select=text%2CuserLocation...
```

```
{
    "@odata.context": "https://tech-immersion.search.windows.net/indexes('tweet-index')
/$metadata#docs(*)",
    "@odata.count": 280,
    "value": [
        {
            "@search.score": 1,
            "text": "@ContosoAuto Thank you for the amazing service at the Portland, OR service
center #Chevrolet #Corvette",
            "userLocation": "Madison, WI",
            "sentiment": 0.9957557916641235
        },
        {
            "@search.score": 1,
            "text": "@ContosoAuto Thank you for the amazing service at the Portland, OR service
center #Volkswagon #GTI",
            "userLocation": "Boston, MA",
            "sentiment": 0.9957557916641235
        },
```

Using the orderBy filter in a search query

In the search results, observe that the sentiment values slowly decrease as you scroll down through the returned documents.

Wrap-up

In this experience, you learned how to leverage Azure Search and Cognitive Services to perform knowledge mining on unstructured data stored in Cosmos DB. Using a combination of preconfigured and custom cognitive skills, you built a Cognitive Search pipeline to enrich the source data in route to an Azure Search Index.

Using preconfigured cognitive skills, you were able to add language detection, sentiment analysis, and key phrase and entity extraction to your search pipeline. These skills enriched your search index with additional metadata about the tweets being indexed.

You then used an Azure Function App to create a custom cognitive skill, which used the Translator Text Cognitive Service to translate tweets into English. Using the Custom Web API skill, you integrated the custom skill to your cognitive search pipeline.

The end result is rich additional content in an Azure Search index, created by a cognitive search indexing pipeline. The output is a full-text searchable index on Azure Search.

Additional resources and more information

To continue learning and expand your understanding of Knowledge Mining with Cognitive Search, use the links below.

- Introduction to Azure Search
- Introduction to Cognitive Services
- Introduction to Cognitive Search
- Attach a Cognitive Services resource with a skillset in Azure Search
- Azure Search Service REST API
- Predefined Cognitive Search skills
  - Key Phrase Extraction
  - Language Detection
  - Entity Recognition
  - Text Merger
  - Text Split
  - Sentiment
  - Image Analysis
  - Optical Character Recognition (OCR)
  - Shaper
- Custom Web API skill
- How to add a custom skill to a cognitive search pipeline
- Learn how to call cognitive search APIs
- Learn Cognitive Search
- Enterprise Knowledge Mining Bootcamp
- Azure Search pricing