# Contents

# Azure Cognitive Search
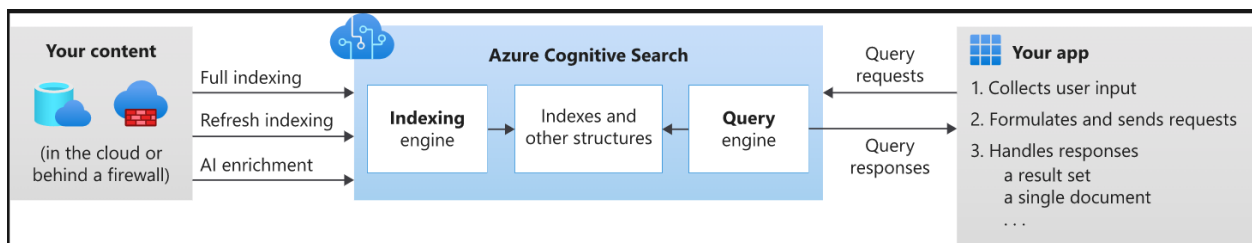
## What is Azure Cognitive Search?

Azure Cognitive Search ([formerly known as "Azure Search"](#)) is a cloud search service that gives developers infrastructure, APIs, and tools for building a rich search experience over private, heterogeneous content in web, mobile, and enterprise applications.

Search is foundational to any app that surfaces text content to users, with common scenarios including catalog or document search, online retail, or knowledge mining for data science.

When you create a search service, you'll work with the following capabilities:

- A search engine for full text search with storage for user-owned content in a search index
- Rich indexing, with [text analysis](#) and [optional AI enrichment](#) for advanced content extraction and transformation
- Rich query capabilities, including simple syntax, full Lucene syntax, and typeahead search
- Programmability through REST APIs and client libraries in Azure SDKs for .NET, Python, Java, and JavaScript
- Azure integration at the data layer, machine learning layer, and AI (Cognitive Services)

Architecturally, a search service sits between the external data stores that contain your un-indexed data, and your client app that sends query requests to a search index and handles the response.



Across the Azure platform, Cognitive Search can integrate with other Azure services in the form of *indexers* that automate data ingestion/retrieval from Azure data sources, and *skillsets* that incorporate consumable AI from Cognitive Services, such as image and natural language processing, or custom AI that you create in Azure Machine Learning or wrap inside Azure Functions.

Key terms used in the hackathon:

| | |
|---|---|
| Indexer | A configuration object specifying a data source, target index, an optional skillset, optional schedule, and optional configuration settings for error handing and base-64 encoding. |
| Data Source | Persists connection information to source data, including credentials. A data source object is used exclusively with indexers. |
| Index | Physical data structure used for full text search and other queries. |

# Import Data

The **Import data wizard** in the Azure portal creates multiple objects used for indexing and AI enrichment on a search service.

In the Azure portal, open the search service page from the dashboard or find your service in the service list. In the service Overview page at the top, click **Import data**.



When you click on Import data, it will start the wizard. The wizard is organized into four main steps:

1. Connect to a supported Azure data source.
    o Select Azure CosmosDB from the datasource drop down menu.



    o In the **data source** page, the source must be **Cosmos DB**, with the following specifications:
    o **Name** is the name of the data source object. Once created, you can choose it for other workloads.
    o **Cosmos DB account** should be in one of the following formats:
        ▪ The primary or secondary connection string from Cosmos DB with the following format: AccountEndpoint=https://<Cosmos DB account name>.documents.azure.com;AccountKey=<Cosmos DB auth key>;.
        ▪ Select Choose an existing connection and it will show you the CosmosDb instance that is created in the resource group
    o **Database** is an existing database from the account. - fsihack
    o **Collection** is a container of documents. Documents must exist in order for import to succeed. - claims
    o **Query** can be blank if you want all documents, otherwise you can input a query that selects a document subset. **Query** is only available for the SQL API.

2. Create an index schema, inferred by sampling source data.

3. Optionally, add AI enrichments to extract or generate content and structure. Inputs for creating a knowledge store are collected in this step. For our hackathon we will "Skipto : Cutsomize target index"

4. Change the index name to "claimsindex". The index that we will create will be used in the front-end UI application. If we want to make the field available to filter on, select the appropriate configuration. Select Next to create the indexer



5. Run the wizard to create objects, load data, set a schedule and other configuration options. Change it to run hourly and click submit.



6. Upon successful completion of the indexer it will process the document in cosmosdb and make it available as searchable index document in Azure search.

7. To view the documents, click on Index tab and select the index name



8. Click on the search and it will show you indexed documents

# claimsindex ...
index

Save   ✕ Discard   ↻ Refresh   ☐ Create Demo App   🗑 Delete

**Search explorer**   Fields   CORS   Scoring profiles   Index Definition (JSON)

Query string ⓘ                                                          API version

| Examples: *, $top=10, $top=10&$skip=10&search=* | 2020-06-30-Pre... ∨ | Search |

Request URL

| https://od478734azs.search.windows.net/indexes/claimsindex/docs?api-version=2020-06-30-Preview&search=* | 📋 |

Results

```
1    {
2        "@odata.context": "https://od478734azs.search.windows.net/indexes('claimsindex')/$metadata#docs(*)",
3        "value": [
4            {
5                "@search.score": 1,
6                "Address": null,
7                "CountryRegion": null,
8                "DateOfBirth": null,
9                "DateOfExpiration": null,
10               "DocumentNumber": null,
11               "DocumentType": null,
12               "FileName": "Georgia_Insurance.jpg",
13               "FirstName": null,
14               "LastName": null,
15               "MachineReadableZone": null,
16               "Nationality": null,
17               "Region": null,
18               "Sex": null,
19               "claimId": "Georgia",
20               "formtype": "Insurance",
21               "rid": "LXNON0FMcjMyWmNHQUFBQUFBQUFBQT090"
22           }
```