Introduction to Azure AI Search

Azure AI Search is a cloud-based search-as-a-service solution that provides powerful, AI-driven search capabilities for building and integrating sophisticated search experiences into applications. It leverages the scale and flexibility of the Azure cloud, allowing developers to add search functionality to websites, applications, and other digital experiences without managing the underlying infrastructure.

Key Features of Azure AI Search

1. **Full-Text Search**: Enables users to perform text-based searches across a wide range of content types, including documents, web pages, and databases.
2. **Natural Language Processing (NLP)**: Enhances search relevance and precision by understanding user queries in natural language. This includes capabilities like synonyms, stemming, and language detection.
3. **AI-Enriched Search**: Integrates AI capabilities such as cognitive skills (image and text analysis), entity recognition, and sentiment analysis to enhance the search index with enriched metadata and annotations.
4. **Faceted Navigation and Filtering**: Allows users to filter and navigate search results based on facets like categories, price ranges, and other structured data.
5. **Autocomplete and Suggestions**: Provides real-time query suggestions and auto-complete options to help users find relevant results faster.
6. **Customizable Scoring Profiles**: Enables fine-tuning of search result rankings based on custom criteria, improving the relevance of search results for specific applications.
7. **Scalability**: Azure AI Search is fully scalable, capable of handling vast amounts of data and high query volumes with ease.
8. **Security**: Provides built-in security features, including encryption, access control, and role-based access, ensuring that only authorized users can access sensitive information.

Common Use Cases

1. **Enterprise Search**: Organizations use Azure AI Search to index and search through large repositories of documents, emails, intranet pages, and databases, making it easier for employees to find relevant information.
2. **E-commerce Search**: Retailers integrate Azure AI Search into their online stores to enhance product search capabilities, enabling customers to find products quickly and accurately.
3. **Website Search**: Content-heavy websites, such as news sites or knowledge bases, utilize Azure AI Search to provide users with powerful search functionality, improving user engagement and satisfaction.
4. **Application Integration**: Developers can embed search functionality within mobile or web applications, offering users seamless access to information, whether it’s customer data, product catalogs, or media libraries.

How Azure AI Search Works

1. **Data Ingestion**: Azure AI Search indexes data from various sources, such as databases, file storage, or custom data sources. The indexed data is stored in a search index, which is optimized for fast querying.
2. **Querying**: Users perform searches by submitting queries to the Azure AI Search service. These queries can be simple keyword searches or complex queries with filters and operators.
3. **Indexing and Enrichment**: Azure AI Search can enrich data during the indexing process using AI cognitive skills. For example, it can extract key phrases, detect language, analyze sentiment, and more.
4. **Search Results**: The service returns search results that match the user's query, sorted by relevance. Developers can customize the ranking and presentation of search results to fit the application's needs.
5. **Administration and Monitoring**: Azure AI Search provides tools for managing indexes, monitoring query performance, and scaling the service as needed.

Benefits of Azure AI Search

- **Improved Search Accuracy**: AI-driven features like NLP and cognitive skills help return more relevant search results, improving the user experience.
- **Faster Time to Market**: Being a fully managed service, Azure AI Search allows developers to quickly integrate search capabilities into applications without worrying about the underlying infrastructure.
- **Scalability**: The service automatically scales to handle increased data volume and query load, ensuring consistent performance.
- **Security and Compliance**: Azure AI Search adheres to Azure's security and compliance standards, providing robust protection for sensitive data.

Getting Started with Azure AI Search

To start using Azure AI Search, you'll need to:

1. **Create an Azure AI Search resource** in the Azure portal.
2. **Index your data** by setting up data sources and defining indexers.
3. **Query the search index** using the REST API or Azure SDKs.
4. **Customize and tune** the search experience to meet your application's needs.

Manage capacity

Managing Capacity in Azure AI Search is crucial for ensuring that your search service can handle the load, perform efficiently, and scale according to the needs of your application. Azure AI Search offers various tools and strategies to manage and optimize the capacity of your search service.

1. Understanding Capacity in Azure AI Search

Capacity in Azure AI Search primarily refers to the allocation of resources such as **partitions** and **replicas**:

- **Partitions**: These are units of storage and compute that handle a portion of your data. As your data grows, you may need more partitions to store and process it effectively.
- **Replicas**: These are copies of your index that handle query load. Having more replicas improves the query performance and provides redundancy for high availability.

2. Managing Partitions

Partitions are critical for scaling the storage and indexing capabilities of your Azure AI Search service.

- **Determine the Number of Partitions**:
    - The number of partitions you need depends on the amount of data you plan to index. Each partition has a storage limit, and increasing the number of partitions allows you to store more data.
    - Start with a single partition and monitor your storage usage. As your index grows, add more partitions to accommodate the data.
- **Add Partitions**:
    - You can add partitions through the Azure portal or programmatically using the Azure Management APIs.
    - Adding more partitions helps distribute the load, which can improve indexing performance.
- **Monitor Partition Usage**:
    - Use Azure Monitor to keep track of partition usage. Pay attention to metrics like storage utilization, index size, and indexing latency to determine if more partitions are needed.

3. Managing Replicas

Replicas are essential for scaling the query performance and ensuring high availability.

- **Determine the Number of Replicas**:
    - The number of replicas required depends on the query load your application expects. More replicas allow for better distribution of queries and faster response times.
    - A minimum of two replicas is recommended for production workloads to ensure high availability.
- **Add Replicas**:
    - Increase the number of replicas through the Azure portal or programmatically using the Azure Management APIs.
    - Adding replicas can directly improve query throughput and reduce latency.
- **Monitor Replica Performance**:
    - Use Azure Monitor to observe query latency, query load, and error rates. If your queries are taking too long or if you experience timeouts, it may be time to add more replicas.

4. Scaling Strategies

Azure AI Search allows for both **manual scaling** and **auto-scaling**:

- **Manual Scaling**:
    - Manually adjust the number of partitions and replicas based on your application's needs.
    - This approach gives you control over scaling but requires careful monitoring to ensure optimal performance.
- **Auto-Scaling**:
    - While Azure AI Search does not currently support automatic scaling directly within the service, you can implement auto-scaling logic using Azure Automation, Logic Apps, or Azure Functions.
    - By monitoring key metrics (like query volume or index size), you can automate the process of adding or removing replicas and partitions.

5. Capacity Planning

Capacity planning involves forecasting your resource needs based on anticipated growth in data volume and query traffic:

- **Estimate Future Growth**: Consider how much your data and query load are likely to grow. Use this to estimate the number of partitions and replicas you might need in the future.
- **Test and Optimize**: Conduct load testing to determine how well your current configuration handles peak loads. Use this information to optimize your capacity planning.
- **Budgeting**: Be mindful of the costs associated with adding partitions and replicas. Plan your budget to ensure you can scale your Azure AI Search service as needed.

6. Monitoring and Alerts

- **Azure Monitor and Alerts**:
    - Set up **Azure Monitor** to track key performance metrics like CPU usage, memory usage, query latency, and error rates.
    - Configure **alerts** to notify you when certain thresholds are exceeded, allowing you to take action before performance degrades.
- **Log Analytics**:
    - Use **Azure Log Analytics** to gather and analyze logs from your Azure AI Search service. This helps in identifying trends and potential issues that could affect capacity.

7. High Availability and Disaster Recovery

- **High Availability**: Ensure high availability by deploying multiple replicas across different Azure regions. This setup helps maintain service continuity during regional outages.
- **Disaster Recovery**: Implement disaster recovery strategies by backing up your indexes and having a recovery plan in place. Regularly test your disaster recovery processes to ensure they work as expected.

Search components

Azure AI Search Components are the building blocks that allow you to create, configure, and manage powerful search experiences within your applications. Understanding these components is essential for effectively utilizing Azure AI Search to meet your application's requirements.

1. Search Service

The **Search Service** is the core component of Azure AI Search. It is a cloud-based service that provides search capabilities through a REST API or SDKs. When you create a Search Service, it includes the following components:

- **Indexes**
- **Data Sources**
- **Indexers**
- **Skillsets**

2. Indexes

An **Index** is a structure that stores searchable content. It is similar to a database table in that it defines the schema (fields) and stores the data that users can search against.

- **Fields**: Each index consists of fields that define the types of data stored (e.g., strings, numbers, booleans). Fields can be searchable, filterable, sortable, facetable, and retrievable, depending on the requirements.
- **Documents**: The actual data stored in an index is called a document. Each document is an instance of the schema defined by the index's fields.
- **Index Management**: You can create, update, delete, and manage indexes through the Azure portal, REST API, or Azure SDKs.

3. Data Sources

A **Data Source** is the connection to the original data that will be indexed. Azure AI Search supports various types of data sources, including:

- **Azure SQL Database**: Index data from a SQL database.
- **Azure Blob Storage**: Index data from files stored in blob storage (e.g., PDFs, images, or CSV files).
- **Azure Cosmos DB**: Index data from a NoSQL database.

- **Table Storage**: Index data from Azure Table Storage.

4. Indexers

An **Indexer** is a component that automatically extracts data from a data source and populates the index. Indexers can be scheduled to run at intervals, ensuring that your index is always up to date with the latest data.

- **Indexing Schedule**: You can configure the indexer to run on a schedule (e.g., every hour or daily) to keep the index in sync with the data source.
- **Field Mappings**: Indexers allow you to map fields from the data source to the fields in your index, enabling custom data transformations during the indexing process.

5. Skillsets

Skillsets are a unique feature of Azure AI Search that allow you to enrich data during the indexing process using AI cognitive skills. A skillset is a collection of skills that process and enrich the content before it's stored in the index.

- **Cognitive Skills**: Skills are predefined operations like language detection, image analysis, entity recognition, sentiment analysis, and more.
- **Custom Skills**: You can create custom skills using Azure Functions to perform specific data processing tasks that aren't covered by the predefined cognitive skills.

6. Search Queries

Search Queries are how users interact with the indexed data. Azure AI Search supports a rich query language that allows users to search for documents based on keywords, phrases, filters, and more.

- **Full-Text Search**: Allows users to perform free-text searches on searchable fields.
- **Filters**: Use filters to narrow down search results based on specific criteria (e.g., price range, category).
- **Facets**: Faceted navigation helps users explore data by grouping results into categories, such as by brand, price, or date.
- **Autocomplete**: Provides suggestions as users type, improving the search experience.

7. Synonyms

Synonyms allow you to define alternative terms or phrases that should be considered equivalent during a search query. For example, a search for "laptop" might also return results for "notebook" if synonyms are defined.

- **Synonym Maps**: Synonym maps are managed lists of synonyms that you can associate with your indexes. These maps help improve search relevance by capturing variations in terminology.

8. Security

Security in Azure AI Search is implemented through role-based access control (RBAC) and encryption.

- **Role-Based Access Control (RBAC)**: Manage access to the search service and its resources by assigning roles to users or applications.
- **Encryption**: Data in Azure AI Search is encrypted both in transit and at rest. You can also use your own encryption keys with Azure Key Vault for added security.

9. Monitoring and Logging

Monitoring and logging are essential for managing and optimizing your search service:

- **Azure Monitor**: Track the performance of your search service, including query latency, document count, and storage usage.
- **Application Insights**: Use Application Insights for deeper analysis of search query performance and user behavior.
- **Log Analytics**: Collect and analyze logs from your search service to troubleshoot issues and gain insights into search usage pattern.

10. API and SDKs

Azure AI Search provides a robust set of APIs and SDKs for interacting with the service:

- **REST API**: The REST API allows you to manage the search service, create and query indexes, and more.
- **SDKs**: Azure AI Search SDKs are available for various programming languages, including .NET, Java, Python, and JavaScript, enabling easy integration into your applications.

Indexing process

The indexing process in Azure AI Search is a crucial step that involves ingesting data from various sources, transforming and enriching that data, and storing it in a searchable format within an index. The indexing process is essential for enabling fast and efficient search queries. Below is a detailed explanation of the indexing process in Azure AI Search:

1. Data Source Identification

The first step in the indexing process is identifying the **data source** from which the data will be ingested. Azure AI Search supports several types of data sources:

- **Azure SQL Database**: Structured data stored in a relational database.
- **Azure Blob Storage**: Unstructured data like documents, images, and CSV files.
- **Azure Cosmos DB**: NoSQL databases with JSON data.
- **Azure Table Storage**: Structured data in table format.
- **Custom Data Sources**: Any data that can be accessed via a custom data ingestion process, such as through an API.

2. Creating an Index

Before indexing data, you must define the **index** that will store your searchable content. An index in Azure AI Search is similar to a database table and consists of:

- **Fields**: Define the structure of the data (e.g., text, number, boolean). Each field can be configured to be searchable, filterable, sortable, facetable, and retrievable.
- **Schema**: The schema outlines the organization of the fields and how they relate to each other.

3. Defining an Indexer

An **indexer** is a component that automates the process of pulling data from the data source, transforming it if necessary, and loading it into the index. The indexer can be scheduled to run at specific intervals to keep the index updated.

- **Field Mappings**: Map fields from the data source to the fields in the index. This may involve transforming or normalizing data (e.g., converting a date format).
- **Indexing Schedule**: Configure the indexer to run on a schedule that suits your application's needs, such as hourly, daily, or weekly.

4. Enrichment with Skillsets (Optional)

Skillsets allow you to enrich your data using cognitive skills during the indexing process. This is particularly useful for unstructured data like documents and images. Skillsets can perform tasks such as:

- **Language Detection**: Automatically detect the language of the text.
- **Entity Recognition**: Identify entities like names, locations, and dates within the text.
- **Sentiment Analysis**: Determine the sentiment of the text (positive, negative, neutral).
- **Image Analysis**: Extract information from images, such as text using OCR or recognizing objects.

5. Running the Indexer

When the indexer runs, it performs the following steps:

- **Data Extraction**: The indexer connects to the data source and retrieves the data. For structured data, this might involve querying a database; for unstructured data, it could involve reading files from blob storage.
- **Transformation**: If field mappings or transformations are defined, the indexer applies them to the data before indexing.
- **Enrichment**: If a skillset is defined, the indexer sends the data through the cognitive skills for enrichment.
- **Indexing**: The processed and enriched data is then loaded into the index.

6. Index Storage and Optimization

Once the data is indexed, it is stored in the Azure AI Search service, optimized for fast search queries:

- **Sharding**: The index might be divided into partitions (shards) to improve performance and scalability, especially if the dataset is large.
- **Compression**: Data within the index is often compressed to reduce storage requirements and improve query performance.

7. Querying the Index

After indexing, the data is available for search queries. Users can perform full-text searches, filter queries, and more, depending on how the index fields are configured.

- **Searchable Fields**: Users can search across fields that are marked as searchable.
- **Faceted Navigation**: If facetable fields are defined, users can filter search results by categories like date ranges, price, etc.

8. Monitoring and Updating the Index

Continuous monitoring is necessary to ensure the indexing process remains efficient and the search results are relevant:

- **Monitor Indexer Performance**: Use Azure Monitor to track indexer execution, success rates, and errors.
- **Update Indexes**: As your data evolves, you may need to update your index schema, field mappings, or skillsets. The indexer can be re-run to update the index with new data.

9. Handling Indexer Failures

If the indexer encounters errors (e.g., due to data inconsistencies or connectivity issues), Azure AI Search provides logs and diagnostics to help identify and resolve the issues.

- **Retry Logic**: Indexers often include built-in retry logic for transient failures.
- **Error Handling**: Define how to handle specific types of errors, such as ignoring certain documents or halting the indexing process.

The indexing process in Azure AI Search is designed to be flexible, allowing for the efficient ingestion, transformation, enrichment, and storage of data in a format optimized for search. By understanding and effectively managing each step of the indexing process, you can ensure that your search service delivers accurate and relevant results to users, while also maintaining performance and scalability as your data grows.

Search an Index

Searching an Index in Azure AI Search is the process of querying the data that has been indexed to retrieve relevant information based on user input. This process involves using various query types and parameters to refine and optimize search results. Here's how you can search an index in Azure AI Search:

1. Basic Search Query

A **basic search query** involves performing a full-text search on the documents within the index. You can search across all fields or specify particular fields to target.

- **All Fields Search**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=search-term

Example: Search for the term "laptop" across all searchable fields in the index.

- **Specific Field Search**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=search-term&searchFields=field1,field2

Example: Search for "laptop" specifically within the **productName** and **description** fields.

2. Filter Queries

Filters allow you to narrow down search results based on specific criteria, such as numeric ranges, dates, or categorical data.

- **Filter by Numeric Range**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&$filter=price gt 100 and price lt 500

Example: Find all products priced between $100 and $500.

- **Filter by Date**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&$filter=releaseDate ge 2023-01-01

Example: Find all products released on or after January 1, 2023.

- **Filter by Category**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&$filter=category eq 'Electronics'

Example: Find all products in the "Electronics" category.

3. Sorting Results

You can **sort** the search results based on specific fields, such as by price or release date.

- **Sort by Price Ascending**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&$orderby=price asc

Example: Sort the results by price in ascending order.

- **Sort by Release Date Descending**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&$orderby=releaseDate desc

Example: Sort the results by the most recent release date first

4. Faceted Navigation

Faceted navigation allows users to refine their search results by grouping them into categories (facets) like price ranges, brands, or product types.

- **Request Facets**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&facet=category,count:10&facet=price,interval:50

Example: Request facets for the **category** field (with the top 10 categories) and for the **price** field (with intervals of $50).

Using Facets in the UI: After retrieving facet values, you can present them in the user interface, allowing users to filter results by clicking on a specific category or price range.

5. Autocomplete and Suggestions

Azure AI Search supports **autocomplete** and **suggestions** to enhance the user experience by providing real-time search suggestions as users type.

- **Autocomplete**:

HTTP:

GET /indexes/{index-name}/docs/suggest?api-version=2021-04-30-Preview&suggesterName=mySuggester&search=lap

Example: Provide suggestions for the prefix "lap", which might include "laptop", "lap desk", etc.

- **Suggestions**:

HTTP:

GET /indexes/{index-name}/docs/suggest?api-version=2021-04-30-Preview&search=lap&suggesterName=mySuggester&top=5

Example: Return the top 5 suggestions based on the user's input.

6. Highlighting Search Results

Highlighting allows you to emphasize the search terms within the search results, making it easier for users to identify relevant content.

- **Highlight Search Terms**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=search-term&highlight=description

Example: Highlight occurrences of the search term within the **description** field in the results.

7. Geospatial Search

Azure AI Search supports **geospatial search** queries, which allow you to search based on geographic location.

- **Search by Location**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=*&$filter=geo.distance(location, geography'POINT(-122.131577 47.678581)') le 10

Example: Find all items within 10 kilometers of the given coordinates.

8. Combining Queries

You can **combine** different query types to create more sophisticated searches.

- **Combined Query Example**

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=laptop&$filter=price lt 1000&$orderby=releaseDate desc

Example: Search for "laptop", filter results to those priced under $1000, and sort by the most recent release date

9. Search Scoring Profiles

Scoring profiles allow you to customize how search results are ranked based on specific criteria, such as boosting certain fields or adjusting the influence of certain data on the search score.

- **Applying a Scoring Profile**:

HTTP:

GET /indexes/{index-name}/docs?api-version=2021-04-30-Preview&search=search-term&scoringProfile=myScoringProfile

Example: Apply a scoring profile named "**myScoringProfile**" to influence the ranking of search results.

10. Search Security

Search Security in Azure AI Search ensures that only authorized users can query specific data or fields.

- **API Keys**: Use API keys to authenticate and authorize access to your search service.
- **Role-Based Access Control (RBAC)**: Implement RBAC to control access to different aspects of the search service.

Applying Filtering & Sorting in Azure AI Search

Filtering and sorting are essential techniques in Azure AI Search that allow users to refine search results based on specific criteria and order them in a meaningful way. Here's a simple and clear explanation of how these work, along with examples of expected output.

1. Filtering in Azure AI Search

Filtering allows you to narrow down search results by specifying conditions that must be met for a document to be included in the search results.

- **Example Scenario**: You have an index of products, and you want to filter the results to show only products that cost less than $500.
- **Sample Query:**

HTTP:

```
GET /indexes/products/docs?api-version=2021-04-30-Preview&search=*&$filter=price lt 500
```

- Expected Output: The search results will include only the products where the price is less than $500.

JSON:

```
{
  "value": [
    {
      "productId": "1",
      "productName": "Smartphone A",
      "price": 299
    },
    {
      "productId": "2",
      "productName": "Tablet B",
      "price": 450
    }
  ]
}
```

2. Sorting in Azure AI Search

Sorting allows you to order the search results based on a particular field, such as price, rating, or date.

- **Example Scenario**: You want to sort the filtered products by price in ascending order.
- **Sample Query**:

HTTP:

```
GET /indexes/products/docs?api-version=2021-04-30-Preview&search=*&$filter=price lt 500&$orderby=price asc
```

- **Expected Output**: The search results will be sorted with the cheapest product first.

JSON:


```
{
  "value": [
    {
      "productId": "1",
      "productName": "Smartphone A",
      "price": 299
    },
    {
      "productId": "2",
      "productName": "Tablet B",
      "price": 450
    }
  ]
}
```


3. Combining Filtering and Sorting

You can **combine filtering and sorting** to refine and organize your search results simultaneously.

- **Example Scenario**: Filter products priced under $500 and sort them by release date, with the newest first.
- **Sample Query**:

HTTP:

```
GET /indexes/products/docs?api-version=2021-04-30-Preview&search=*&$filter=price lt 500&$orderby=releaseDate desc
```

- **Expected Output**: The search results will show products under $500, with the most recently released products appearing first.

JSON:


```
{
  "value": [
    {
      "productId": "3",
      "productName": "Smartwatch C",
      "price": 350,
      "releaseDate": "2024-08-01"
    },
    {
      "productId": "1",
      "productName": "Smartphone A",
      "price": 299,
      "releaseDate": "2024-07-15"
    }
  ]
}
```


Summary

- **Filtering** helps you narrow down results by specific conditions (e.g., price < $500).
- **Sorting** allows you to order results (e.g., by price or date).
- **Combining filtering and sorting** gives you refined and organized results based on multiple criteria.

Enhancing the Index in Azure AI Search

Enhancing an index in Azure AI Search involves optimizing the search experience by adding functionalities such as analyzers, scoring profiles, and cognitive skills. These enhancements can make search results more relevant, faster, and more useful to end users.

1. Analyzers

Analyzers are tools that process the text in the index fields, breaking it down into tokens (words or terms) and applying filters (e.g., lowercasing, removing stop words) to make searches more effective.

- **Example**: Using a custom analyzer to support searches in multiple languages.
- **How to Implement**: When defining your index, specify an analyzer for fields that require special processing.

JSON:

```
{
  "name": "description",
  "type": "Edm.String",
  "analyzer": "standard.lucene"
}
```

Expected Impact: Searches become more accurate, especially in multilingual contexts, where specific language rules are applied.

2. Scoring Profiles

Scoring profiles allow you to customize how search results are ranked. You can boost certain fields or use functions to influence the score of documents based on numeric values or dates.

- **Example**: Boosting the relevance of newer products by giving higher scores to recent release dates.
- **How to Implement**: Define a scoring profile in your index configuration.

JSON:

```
{
  "name": "recentBoost",
```

```
 "text": {
  "weights": {
   "productName": 1.5,
   "description": 1.0
  }
 },
 "functions": [
  {
   "type": "freshness",
   "fieldName": "releaseDate",
   "boost": 2,
   "interpolation": "linear",
   "parameters": {
    "boostingDuration": "365d"
   }
  }
 ]
}
```

- **Expected Impact**: Products released more recently will appear higher in search results, improving the visibility of new items.

3. Cognitive Skills and Enrichment

Cognitive skills add AI-powered capabilities to your index, such as language detection, sentiment analysis, and image recognition. These skills can enrich the data before it is indexed, making it more searchable and informative.

- **Example**: Automatically detecting the language of a document and adding it as a searchable field.
- **How to Implement**: Define a skillset and attach it to the indexer.

JSON:

```
{
  "skills": [
    {
      "type": "#Microsoft.Skills.Text.LanguageDetectionSkill",
      "inputs": [
        { "name": "text", "source": "/document/content" }
      ],
      "outputs": [
        { "name": "languageCode", "targetName": "language" }
      ]
    }
  ]
}
```

- **Expected Impact**: The index now includes the detected language of each document, allowing users to filter or boost search results based on language.

4. Indexer Schedules

Keeping the index up-to-date is crucial for delivering accurate search results. **Indexer schedules** allow you to automate the process of updating the index by regularly crawling the data source for new or changed content.

- **Example**: Scheduling an indexer to run every day at midnight to keep the product catalog current.
- **How to Implement**: Configure the indexer with a schedule.

JSON:

```
{
  "schedule": {
```

```
  "interval": "P1D", // Every 1 day
  "startTime": "2024-08-27T00:00:00Z"
 }
}
```

Expected Impact: The index stays fresh, ensuring that users always see the most up-to-date information.

5. Geo-Spatial Indexing

If your data includes geographic locations, **geo-spatial indexing** allows users to search for items based on their proximity to a specified location.

- **Example**: Allowing users to search for nearby stores or products available within a certain radius.
- **How to Implement**: Include a geographyPoint field in your index schema.

JSON:

```
{
 "name": "location",
 "type": "Edm.GeographyPoint"
}
```

- **Expected Impact**: Users can perform location-based searches, enhancing the search experience for services like store locators or delivery zones.

Summary

Enhancing an index in Azure AI Search involves adding and configuring features that improve search accuracy, relevance, and user experience. By implementing analyzers, scoring profiles, cognitive skills, and more, you can create a powerful and efficient search solution that meets the specific needs of your application.

These enhancements ensure that your search results are not only accurate but also tailored to the specific needs of your users, making the overall search experience more effective and user-friendly.

Creating a Custom Skill for Azure AI Search

Custom skills in Azure AI Search allow you to define specific operations that enrich your search index by applying unique transformations or analyses to the data during the indexing process. Custom skills are part of the broader set of cognitive skills used in Azure Cognitive Search to process and enrich content.

Here's a step-by-step guide to creating a custom skill for Azure AI Search.

1. Understand the Custom Skill Scenario

Before creating a custom skill, you need to identify the specific scenario where it will be applied. For example, you might want to:

- **Example**: Extract key phrases from a document and store them as a searchable field in the index.

2. Create an Azure Function for the Custom Skill

Custom skills are typically implemented using Azure Functions. The Azure Function will process the input data and return the enriched output.

- **Step-by-Step**:
    1. **Create an Azure Function**: Go to the Azure portal and create a new Azure Function App.
    2. **Develop the Function**:
        - Choose the HTTP trigger template.
        - Write the code to process the input (e.g., extract key phrases) and return the output.

Sample Azure Function Code:

Python:

```
import json

from azure.functions import HttpRequest, HttpResponse

def main(req: HttpRequest) -> HttpResponse:

  req_body = req.get_json()
```

```
    documents = req_body['values']
    results = []
    for doc in documents:
        text = doc['data']['text']
        key_phrases = extract_key_phrases(text)  # Implement this function
        results.append({
            "recordId": doc["recordId"],
            "data": { "keyPhrases": key_phrases }
        })

    return HttpResponse(
        json.dumps({"values": results}),
        mimetype="application/json"
    )

def extract_key_phrases(text):
    # Implement key phrase extraction logic
    return ["example phrase 1", "example phrase 2"]
```

- **Deploy the Function**: Deploy the function to Azure and note the function's URL endpoint.

3. Define the Custom Skill in Your Search Indexer

Once your custom skill is ready, you need to define it as part of your indexer's skillset.

- **Sample Skillset Definition**:

JSON:

```
{
  "skills": [
    {
```

```
    "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",
    "uri": "https://<your-function-app>.azurewebsites.net/api/<function-name>?code=<function-key>",
    "httpMethod": "POST",
    "timeout": "PT30S",
    "batchSize": 1,
    "context": "/document",
    "inputs": [
     {
      "name": "text",
      "source": "/document/content"
     }
    ],
    "outputs": [
     {
      "name": "keyPhrases",
      "targetName": "keyPhrases"
     }
    ]
   }
 ]
}
```

Explanation:

- **uri**: The endpoint of your Azure Function.
- **httpMethod**: The method used to call the function, typically POST.
- **inputs**: The data passed to the function (e.g., text from the document).
- **outputs**: The enriched data returned by the function (e.g., key phrases).

4. Integrate the Custom Skill into the Indexing Pipeline

After defining the custom skill, integrate it into the indexing pipeline by updating the indexer configuration.

- **Sample Indexer Definition**

JSON:

```
{
  "dataSourceName": "your-datasource",
  "targetIndexName": "your-index",
  "skillsetName": "your-skillset",
  "fieldMappings": [
    {
      "sourceFieldName": "text",
      "targetFieldName": "content"
    }
  ],
  "outputFieldMappings": [
    {
      "sourceFieldName": "/document/keyPhrases",
      "targetFieldName": "keyPhrases"
    }
  ],
  "schedule": {
    "interval": "P1D"  // Run daily
  }
}
```

5. Test and Validate the Custom Skill

- **Run the Indexer**: Execute the indexer and ensure that the custom skill is applied during the indexing process.
- **Check the Output**: Validate that the enriched data (e.g., key phrases) is correctly indexed and searchable.

6. Use the Custom Skill in Search Queries

After successfully indexing the data with the custom skill, you can use the enriched fields in your search queries.

- **Sample Query**:

HTTP:

```
GET /indexes/your-index/docs?api-version=2021-04-30-Preview&search=example+phrase
```

- **Expected Output**: Documents containing the key phrase "example phrase" will be returned as part of the search results.

Summary

Creating a custom skill in Azure AI Search involves building an Azure Function to process and enrich your data, defining the custom skill in a skillset, and integrating it into the indexing pipeline. By doing so, you can tailor the search experience to your specific needs, making the search results more relevant and informative for your users.

Creating a Knowledge Store with Azure AI Search

A **Knowledge Store** in Azure AI Search allows you to store and organize enriched content extracted during the indexing process. It supports advanced data processing scenarios where you can store and analyze data in different formats like tables, blobs, or even as structured data in Azure Synapse Analytics.

Here's a step-by-step guide to creating a Knowledge Store with Azure AI Search.

1. Understand the Knowledge Store Scenario

Before setting up a Knowledge Store, identify your scenario. For example, you may want to:

- Store and analyze extracted entities from documents.
- Archive images or other media files with enriched metadata.
- Store text analytics data for further processing.

2. Provision Azure Resources

To create a Knowledge Store, you need the following Azure resources:

- **Azure Cognitive Search Service**: To perform indexing and data enrichment.
- **Azure Blob Storage**: To store documents, images, or other files.
- **Azure Table Storage or Azure Synapse Analytics**: To store structured data.

3. Define the Knowledge Store in Your Search Indexer

The Knowledge Store is configured as part of your search indexer. You will specify which enriched content to store, how to store it, and where to store it.

- **Sample Skillset Configuration**:

Here's a sample configuration for a skillset that outputs data to a Knowledge Store.

JSON:

```
{
  "skillset": {
    "name": "my-skillset",
    "skills": [
```

```
  {
    "@odata.type": "#Microsoft.Skills.Text.EntityRecognitionSkill",
    "inputs": [
      { "name": "text", "source": "/document/content" }
    ],
    "outputs": [
      { "name": "entities", "targetName": "entities" }
    ]
  }
],
"knowledgeStore": {
  "storageAccountConnectionString": "<your-blob-storage-connection-string>",
  "projections": [
    {
      "tables": [
        {
          "tableName": "entitiesTable",
          "generatedKeyName": "documentId",
          "source": "/document/entities"
        }
      ],
      "objects": [
        {
          "storageContainer": "enriched-content",
          "generatedKeyName": "documentId",
          "source": "/document"
        }
      ]
    }
```

```
    ]
   }
  }
}
```

Explanation:

- **storageAccountConnectionString**: The connection string for your Azure Blob Storage account.
- **projections**: Defines how the enriched data is stored.
  - **tables**: Specifies the table where structured data (like entities) will be stored.
  - **objects**: Specifies the blob container where documents or images will be stored.

4. Configure and Run the Indexer

After defining the skillset with the Knowledge Store, you need to configure and run the indexer to process data and store it in the Knowledge Store.

- **Sample Indexer Definition:**

JSON:

```
{
 "dataSourceName": "my-data-source",
 "targetIndexName": "my-index",
 "skillsetName": "my-skillset",
 "outputFieldMappings": [
  {
   "sourceFieldName": "/document/entities",
   "targetFieldName": "entities"
  }
 ],
 "knowledgeStore": {
  "storageAccountConnectionString": "<your-blob-storage-connection-string>",
```

```
  "projections": [
   {
    "tables": [
     {
      "tableName": "entitiesTable",
      "generatedKeyName": "documentId",
      "source": "/document/entities"
     }
    ],
    "objects": [
     {
      "storageContainer": "enriched-content",
      "generatedKeyName": "documentId",
      "source": "/document"
     }
    ]
   }
  ]
 }
}
```

- **Run the Indexer**: Execute the indexer to begin processing and storing the data.

5. Query and Analyze Data in the Knowledge Store

Once the data is stored in the Knowledge Store, you can query and analyze it using tools like Azure Synapse Analytics, Power BI, or custom applications.

- **Example Queries**:
    - **Query Structured Data**: You can query the table storage directly or via Azure Synapse to analyze structured data like entities.

SQL:

```
SELECT * FROM entitiesTable WHERE EntityType = 'Location'
```

- **Access Blob Data**: Access enriched documents or media files stored in the blob container through Azure Blob Storage Explorer or directly via APIs.

Summary

Creating a Knowledge Store with Azure AI Search involves setting up a skillset to process and enrich your data, configuring how and where this enriched data should be stored, and running the indexer to populate the Knowledge Store. The stored data can then be queried and analyzed for deeper insights, making it a powerful tool for scenarios like text analytics, entity recognition, and media metadata extraction.

For reference purpose use below link

https://learn.microsoft.com/en-us/azure/search/search-what-is-azure-search