

# Quickstart: Use built-in portal tools for Azure Search import, indexing, and queries

05/01/2019 • 11 minutes to read • Contributors      all

## In this article

[Prerequisites](#)

[Create an index and load data](#)

[Monitor progress](#)

[View the index](#)

[Query using Search explorer](#)

[Example queries](#)

[Takeaways](#)

[Clean up](#)

[Next steps](#)

Portal 

For a fast ramp up on Azure Search concepts, try the built-in tools in the Azure portal. Wizards and editors do not offer full parity with the .NET and REST APIs, but you can get started quickly with a code-free introduction, writing interesting queries against a sample data within minutes.

- ✓ Start with a free public sample data set hosted on Azure
- ✓ Run the **Import data** wizard in Azure Search to load data and generate an index
- ✓ Monitor indexing progress in the portal
- ✓ View an existing index and options for modifying it
- ✓ Explore full text search, filters, facets, fuzzy search, and geosearch with **Search explorer**

If the tools are too limiting, you can consider a [code-based introduction to programming Azure Search in .NET](#) or use [Postman or Fiddler for making REST API calls](#).

If you don't have an Azure subscription, create a [free account](#) before you begin. You could also watch a 6-minute demonstration of the steps in this tutorial, starting at about three

minutes into this [Azure Search Overview video](#).

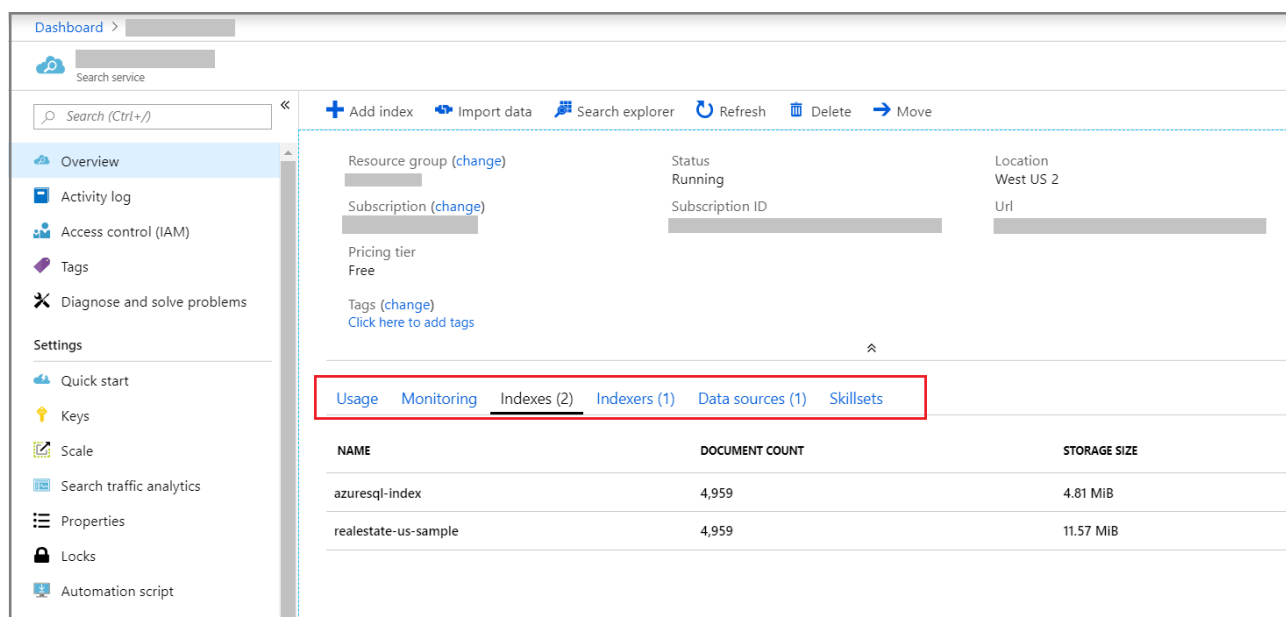
## Prerequisites

[Create an Azure Search service](#) or [find an existing service](#) under your current subscription. You can use a free service for this quickstart.

### Check for space

Many customers start with the free service. This version is limited to three indexes, three data sources, and three indexers. Make sure you have room for extra items before you begin. This tutorial creates one of each object.

Sections on the service dashboard show how many indexes, indexers, and data sources you already have.



The screenshot shows the Azure Search service dashboard. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Quick start, Keys, Scale, Search traffic analytics, Properties, Locks, and Automation script. The main content area displays service details like Resource group, Subscription, Status (Running), Location (West US 2), and Pricing tier (Free). Below this, a tabbed interface shows 'Usage', 'Monitoring', 'Indexes (2)', 'Indexers (1)', 'Data sources (1)', and 'Skillsets'. The 'Indexes (2)' tab is active, displaying a table with the following data:

NAME	DOCUMENT COUNT	STORAGE SIZE
azuresql-index	4,959	4.81 MIB
realestate-us-sample	4,959	11.57 MIB

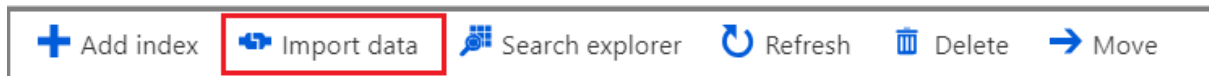
## Create an index and load data

Search queries iterate over an [index](#) that contains searchable data, metadata, and additional constructs that optimize certain search behaviors.

For this tutorial, we use a built-in sample dataset that can be crawled using an [indexer](#) via the **Import data** wizard. An indexer is a source-specific crawler that can read metadata and content from supported Azure data sources. Normally, indexers are used programmatically, but in the portal, you can access them through the **Import data** wizard.

## Step 1 - Start the Import data wizard and create a data source

1. On the Azure Search service dashboard, click **Import data** on the command bar to create and populate a search index.



2. In the wizard, click **Connect to your data** > **Samples** > **realestate-us-sample**. This data source is built-in. If you were creating your own data source, you would need to specify a name, type, and connection information. Once created, it becomes an "existing data source" that can be reused in other import operations.

TYPE	NAME
	realestate-us-sample

3. Continue to the next page.

**Next: Add cognitive search (Optional)**

## Step 2 - Skip Cognitive skills

The wizard supports the creation of a [cognitive skills pipeline](#) for incorporating the Cognitive Services AI algorithms into indexing.

We'll skip this step for now, and move directly on to **Customize target index**.

**Skip to: Customize target index**

### Tip

You can step through an AI-indexing example in a [quickstart](#) or [tutorial](#).

## Step 3 - Configure index

Typically, index creation is a code-based exercise, completed prior to loading data. However, as this tutorial indicates, the wizard can generate a basic index for any data source it can crawl. Minimally, an index requires a name and a fields collection; one of the fields should be marked as the document key to uniquely identify each document. Additionally, you can specify language analyzers or suggesters if you want autocomplete or suggested queries.

Fields have data types and attributes. The check boxes across the top are *index attributes* controlling how the field is used.

- **Retrievable** means that it shows up in search results list. You can mark individual fields as off limits for search results by clearing this checkbox, for example when fields used only in filter expressions.
- **Key** is the unique document identifier. It's always a string, and it is required.
- **Filterable**, **Sortable**, and **Facetable** determine whether fields are used in a filter, sort, or faceted navigation structure.
- **Searchable** means that a field is included in full text search. Strings are searchable. Numeric fields and Boolean fields are often marked as not searchable.

Storage requirements do not vary as a result of your selection. For example, if you set the **Retrievable** attribute on multiple fields, storage requirements do not go up.

By default, the wizard scans the data source for unique identifiers as the basis for the key field. *Strings* are attributed as **Retrievable** and **Searchable**. *Integers* are attributed as **Retrievable**, **Filterable**, **Sortable**, and **Facetable**.

1. Accept the defaults.

If you rerun the wizard a second time using an existing realestate data source, the index won't be configured with default attributes. You'll have to manually select attributes on future imports.

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

**i** 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 ⓘ  
realestate-us-sample ✓

\* Key ⓘ  
listingId ✓

Suggester name  
sg

Search mode ⓘ  
[Dropdown]

Delete

FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACETABLE	SEARCHABLE	ANALYZER	SUGGESTER	
listingId	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	...
beds	Edm.Int32	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				...
baths	Edm.Int32	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				...
description	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	English - Microsoft [v]	<input type="checkbox"/>	...

2. Continue to the next page.

**Next: Create an indexer**

## Step 4 - Configure indexer

Still in the **Import data** wizard, click **Indexer** > **Name**, and type a name for the indexer.


This object defines an executable process. You could put it on recurring schedule, but for now use the default option to run the indexer once, immediately.


Click **Submit** to create and simultaneously run the indexer.

[Connect to your data](#) [Add cognitive search \(Optional\)](#) [Customize target index](#) [Create an indexer](#)

### Indexer

\* Name

 A schedule can't be configured on samples or existing data sources without change tracking. Edit your data source to add change tracking if you wish to set up a schedule.

Schedule 

Description


▼ Advanced options

[Previous: Customize target index](#) [Submit](#)

## Monitor progress

The wizard should take you to the Indexers list where you can monitor progress. For self-navigation, go to the Overview page and click **Indexers**.

It can take a few minutes for the portal to update the page, but you should see the newly created indexer in the list, with status indicating "in progress" or success, along with the number of documents indexed.

<a href="#">Usage</a>	<a href="#">Monitoring</a>	<a href="#">Indexes (2)</a>	<a href="#">Indexers (1)</a>	<a href="#">Data sources (1)</a>	<a href="#">Skillsets</a>
NAME	STATUS	LAST RUN	DOCS SUCCEEDED		
 myindexer	In progress	Just now	0/0		

## View the index

The main service page provides links to the resources created in your Azure Search service. To view the index you just created, click **Indexes** from the list of links.

Usage Monitoring Indexes (2) Indexers (2) Data sources (2) Skillsets

NAME	DOCUMENT COUNT	STORAGE SIZE
azureblob-index	14	228.59 KiB
realestate-us-sample	4,959	11.8 MiB

From this list, you can click on the *realestate-us-sample* index that you just created, view the index schema, and optionally add new fields.

The **Fields** tab shows the index schema. Scroll to the bottom of the list to enter a new field. In most cases, you cannot change existing fields. Existing fields have a physical representation in Azure Search and are thus non-modifiable, not even in code. To fundamentally change an existing field, create a new index, dropping the original.

**realestate-us-sample**  
index

[Save](#) [Discard](#) [Refresh](#) [Delete](#)

Documents **4959** Storage **11.57 MiB**

[Search explorer](#) [Fields](#) [CORS](#) [Scoring profiles](#)

Suggester name  Search mode **1** ▼

[Delete](#)

FIELD NAME	TYPE	RETRIEVABLE	FILTERABLE	SORTABLE	FACETABLE	SEARCHABLE	ANALYZER	SUGGESTER
listingId	Edm.String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/> ...
beds	Edm.Int32	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			...
baths	Edm.Int32	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			...

Other constructs, such as scoring profiles and CORS options, can be added at any time.

To clearly understand what you can and cannot edit during index design, take a minute to view index definition options. Grayed-out options are an indicator that a value cannot be modified or deleted.

## Query using Search explorer

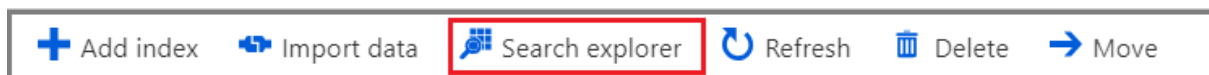
Moving forward, you should now have a search index that's ready to query using the built-in [Search explorer](#) query page. It provides a search box so that you can test arbitrary query strings.

Search explorer is only equipped to handle [REST API requests](#), but it accepts syntax for both [simple query syntax](#) and [full Lucene query parser](#), plus all the search parameters available in [Search Document REST API](#) operations.

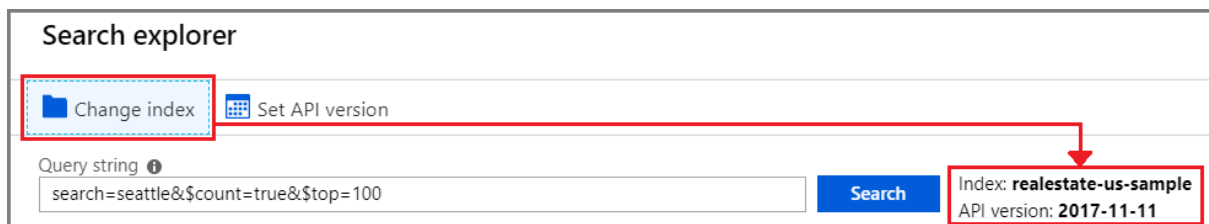
### 💡 Tip

The following steps are demonstrated at 6m08s into the [Azure Search Overview video](#).

1. Click **Search explorer** on the command bar.



2. Click **Change index** on the command bar to switch to *realestate-us-sample*. Click **Set API version** on the command bar to see which REST APIs are available. For the queries below, use the generally available version (2019-05-06).



3. In the search bar, paste in the query strings below and click **Search**.



## Example queries

You can enter terms and phrases, similar to what you might do in a Bing or Google search, or fully-specified query expressions. Results are returned as verbose JSON documents.

### Simple query with top N results



### Example (string query): `search=seattle`

- The **search** parameter is used to input a keyword search for full text search, in this case, returning listings in King County, Washington state, containing *Seattle* in any searchable field in the document.
- **Search explorer** returns results in JSON, which is verbose and hard to read if documents have a dense structure. This is intentional; visibility into the entire document is important for development purposes, especially during testing. For a better user experience, you will need to write code that [handles search results](#) to bring out important elements.
- Documents are composed of all fields marked as "retrievable" in the index. To view index attributes in the portal, click *realestate-us-sample* in the **Indexes** list.

### Example (parameterized query): `search=seattle&$count=true&$top=100`

- The **&** symbol is used to append search parameters, which can be specified in any order.
- The **\$count=true** parameter returns the total count all documents returned. This value appears near the top of the search results. You can verify filter queries by monitoring changes reported by **\$count=true**. Smaller counts indicate your filter is working.
- The **\$top=100** returns the highest ranked 100 documents out of the total. By default, Azure Search returns the first 50 best matches. You can increase or decrease the amount via **\$top**.

## Filter the query

Filters are included in search requests when you append the **\$filter** parameter.

### Example (filtered): `search=seattle&$filter=beds gt 3`

- The **\$filter** parameter returns results matching the criteria you provided. In this case, bedrooms greater than 3.
- Filter syntax is an OData construction. For more information, see [Filter OData syntax](#).

## Facet the query

Facet filters are included in search requests. You can use the facet parameter to return an aggregated count of documents that match a facet value you provide.

**Example (faceted with scope reduction):** `search=*&facet=city&$top=2`

- `search=*` is an empty search. Empty searches search over everything. One reason for submitting an empty query is to filter or facet over the complete set of documents. For example, you want a faceting navigation structure to consist of all cities in the index.
- `facet` returns a navigation structure that you can pass to a UI control. It returns categories and a count. In this case, categories are based on the number of cities. There is no aggregation in Azure Search, but you can approximate aggregation via `facet`, which gives a count of documents in each category.
- `$top=2` brings back two documents, illustrating that you can use `top` to both reduce or increase results.

**Example (facet on numeric values):** `search=seattle&facet=beds **`

- This query is facet for beds, on a text search for *Seattle*. The term *beds* can be specified as a facet because the field is marked as retrievable, filterable, and facetable in the index, and the values it contains (numeric, 1 through 5), are suitable for categorizing listings into groups (listings with 3 bedrooms, 4 bedrooms).
- Only filterable fields can be faceted. Only retrievable fields can be returned in the results.

## Highlight search results

Hit highlighting refers to formatting on text matching the keyword, given matches are found in a specific field. If your search term is deeply buried in a description, you can add hit highlighting to make it easier to spot.

**Example (highlighter):** `search=granite countertops&highlight=description`

- In this example, the formatted phrase *granite countertops* is easier to spot in the description field.

**Example (linguistic analysis):** `search=mice&highlight=description`

- Full text search finds word forms with similar semantics. In this case, search results contain highlighted text for "mouse", for homes that have mouse infestation, in response to a keyword search on "mice". Different forms of the same word can appear in results because of linguistic analysis.
- Azure Search supports 56 analyzers from both Lucene and Microsoft. The default used by Azure Search is the standard Lucene analyzer.

## Try fuzzy search

By default, misspelled query terms, like *samamish* for the Sammamish plateau in the Seattle area, fail to return matches in typical search. The following example returns no results.

**Example (misspelled term, unhandled):** `search=samamish`

To handle misspellings, you can use fuzzy search. Fuzzy search is enabled when you use the full Lucene query syntax, which occurs when you do two things: set **queryType=full** on the query, and append the `~` to the search string.

**Example (misspelled term, handled):** `search=samamish~&queryType=full`

This example now returns documents that include matches on "Sammamish".

When **queryType** is unspecified, the default simple query parser is used. The simple query parser is faster, but if you require fuzzy search, regular expressions, proximity search, or other advanced query types, you will need the full syntax.

Fuzzy search and wildcard search have implications on search output. Linguistic analysis is not performed on these query formats. Before using fuzzy and wildcard search, review [How full text search works in Azure Search](#) and look for the section about exceptions to lexical analysis.

For more information about query scenarios enabled by the full query parser, see [Lucene query syntax in Azure Search](#).

## Try geospatial search

Geospatial search is supported through the [edm.GeographyPoint data type](#) on a field containing coordinates. Geosearch is a type of filter, specified in [Filter OData syntax](#).

### Example (geo-coordinate filters):

```
search=*&$count=true&$filter=geo.distance(location,geography'POINT(-122.121513  
47.673988)') 1e 5
```

The example query filters all results for positional data, where results are less than 5 kilometers from a given point (specified as latitude and longitude coordinates). By adding `$count`, you can see how many results are returned when you change either the distance or the coordinates.

Geospatial search is useful if your search application has a "find near me" feature or uses map navigation. It is not full text search, however. If you have user requirements for searching on a city or country/region by name, add fields containing city or country/region names, in addition to coordinates.

## Takeaways

This tutorial provided a quick introduction to Azure Search using the Azure portal.

You learned how to create a search index using the **Import data** wizard. You learned about [indexers](#), as well as the basic workflow for index design, including [supported modifications to a published index](#).

Using the **Search explorer** in the Azure portal, you learned some basic query syntax through hands-on examples that demonstrated key capabilities such as filters, hit highlighting, fuzzy search, and geo-search.

You also learned how to find indexes, indexers, and data sources in the portal. Given any new data source in the future, you can use the portal to quickly check its definitions or field collections with minimal effort.

## Clean up

If this tutorial was your first use of the Azure Search service, delete the resource group containing the Azure Search service. If not, look up the correct resource group name from the list of services and delete the appropriate one.

## Next steps

You can explore more of Azure Search using the programmatic tools:

- [Create an index using .NET SDK](#)
- [Create an index using REST APIs](#)
- [Create an index using Postman or Fiddler and the Azure Search REST APIs](#)