


Intelligent product search engine for e-commerce

09/14/2018 • 6 minutes to read • Contributors 

In this article

[Relevant use cases](#)

[Architecture](#)

[Considerations](#)

[Deploy the scenario](#)

[Pricing](#)

[Related resources](#)

This example scenario shows how using a dedicated search service can dramatically increase the relevance of search results for your e-commerce customers.

Search is the primary mechanism through which customers find and ultimately purchase products, making it essential that search results are relevant to the *intent* of the search query, and that the end-to-end search experience matches that of search giants by providing near-instant results, linguistic analysis, geo-location matching, filtering, faceting, autocomplete, hit highlighting, etc.

Imagine a typical e-commerce web application with product data stored in a relational database like SQL Server or Azure SQL Database. Search queries are often handled inside the database using LIKE queries or [Full-Text Search](#) features. By using [Azure Search](#) instead, you free up your operational database from the query processing and you can easily start taking advantage of those hard-to-implement features that provide your customers with the best possible search experience. Also, because Azure Search is a platform as a service (PaaS) component, you don't have to worry about managing infrastructure or becoming a search expert.

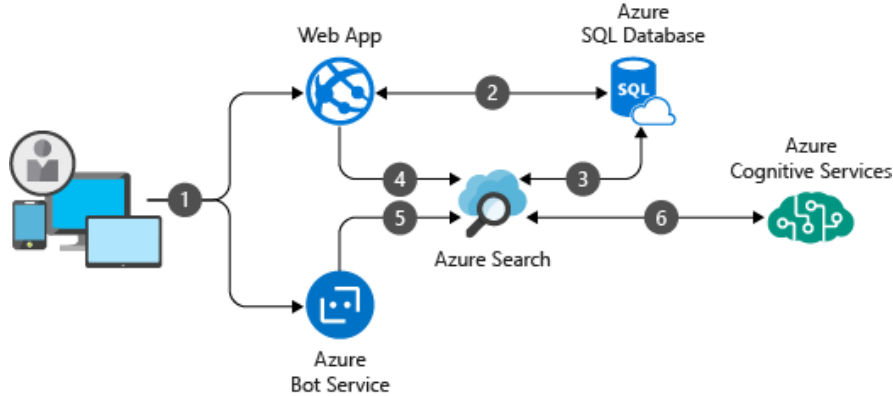
Relevant use cases

Other relevant use cases include:

- Finding real estate listings or stores near the user's physical location.
- Searching for articles in a news site or looking for sports results, with a higher preference for more *recent* information.
- Searching through large repositories for *document-centric* organizations like policy makers and notaries.

Ultimately, *any* application that has some form of search functionality can benefit from a dedicated search service.

Architecture



This scenario covers an e-commerce solution where customers can search through a product catalog.

1. Customers navigate to the **e-commerce web application** from any device.
2. The product catalog is maintained in an **Azure SQL Database** for transactional processing.
3. Azure Search uses a **search indexer** to automatically keep its search index up-to-date through integrated change tracking.
4. Customer's search queries are offloaded to the **Azure Search** service, which processes the query and returns the most relevant results.
5. As an alternative to a web-based search experience, customers can also use a **conversational bot** in social media or straight from digital assistants to search for products and incrementally refine their search query and results.
6. Optionally, the **Cognitive Search** feature can be used to apply artificial intelligence for even smarter processing.

Components

- [App Services - Web Apps](#) hosts web applications allowing autoscale and high availability without having to manage infrastructure.
- [SQL Database](#) is a general-purpose relational database-managed service in Microsoft Azure that supports structures such as relational data, JSON, spatial, and XML.
- [Azure Search](#) is a search-as-a-service cloud solution that provides a rich search experience over private, heterogenous content in web, mobile, and enterprise applications.
- [Bot Service](#) provides tools to build, test, deploy, and manage intelligent bots.
- [Cognitive Services](#) lets you use intelligent algorithms to see, hear, speak, understand, and interpret your user needs through natural methods of communication.

Alternatives

- You could use **in-database search** capabilities, for example, through SQL Server full-text search, but then your transactional store also processes queries (increasing the need for processing power) and the search capabilities inside the database are more limited.
- You could host the open-source [Apache Lucene](#) (on which Azure Search is built) on Azure Virtual Machines, but then you are back to managing Infrastructure-as-a-Service (IaaS) and don't benefit from the many features that Azure Search provides on top of Lucene.
- You could also consider deploying [Elastic Search](#) from the Azure Marketplace, which is an alternative and capable search product from a third-party vendor, but also in this case you are running an IaaS workload.

Other options for the data tier include:

- [Cosmos DB](#) - Microsoft's globally distributed, multi-model database. Cosmos DB provides a platform to run other data models such as Mongo DB, Cassandra, Graph data, or simple table storage. Azure Search also supports indexing the data from Cosmos DB directly.

Considerations

Scalability

The [pricing tier](#) of the Azure Search service doesn't determine the available features but is used mainly for [capacity planning](#) as it defines the maximum storage you get and how many partitions and replicas you can provision.

Partitions allow you to index more documents and get higher write throughputs, whereas **replicas** provide more Queries-Per-Second (QPS) and High Availability.

You can dynamically change the number of partitions and replicas but it's not possible to change the pricing tier, so you should carefully consider the right tier for your target workload. If you need to change the tier anyway, you will need to provision a new service side by side and reload your indexes there, at which point you can point your applications at the new service.

Availability

Azure Search provides a [99.9% availability SLA](#) for *reads* (that is, querying) if you have at least two replicas, and for *updates* (that is, updating the search indexes) if you have at least three replicas. Therefore you should provision at least two replicas if you want your customers to be able to *search* reliably, and 3 if actual *changes to the index* should also be considered high availability operations.

If there is a need to make breaking changes to the index without downtime (for example, changing data types, deleting or renaming fields), the index will need to be rebuilt. Similar to changing service tier, this means creating a new index, repopulating it with the data, and then updating your applications to point at the new index.

Security

Azure Search is compliant with many [security and data privacy standards](#), which makes it possible to be used in most industries.

For securing access to the service, Azure Search uses two types of keys: **admin keys**, which allow you to perform *any* task against the service, and **query keys**, which can only be used for read-only operations like querying. Typically, the application that performs the search does not update the index, so it should only be configured with a query key and not an admin key (especially if the search is performed from an end-user device like script running in a web browser).

Search Relevance

How successful your e-commerce application is depends largely on the relevance of the search results to your customers. Carefully tuning your search service to provide optimal results based on user research, or relying on built-in features such as [search traffic analysis](#) to understand your customer's search patterns allows you to make decisions based on data.

Typical ways to tune your search service include:

- Using [scoring profiles](#) to influence the relevance of search results, for example, based on which field matched the query, how recent the data is, the geographical distance to the user, ...
- Using [Microsoft provided language analyzers](#) that use an advanced Natural Language Processing (NLP) stack to better interpret queries
- Using [custom analyzers](#) to ensure your products are found correctly, especially if you want to search on non-language based information like a product's make and model.

Deploy the scenario

To deploy a more complete e-commerce version of this scenario, you can follow this [step-by-step tutorial](#) that provides a .NET sample application that runs a simple ticket purchasing application. It also includes Azure Search and uses many of the features discussed. Additionally, there is a Resource Manager template to automate the deployment of most of the Azure resources.

Pricing

To explore the cost of running this scenario, all the services mentioned above are pre-configured in the cost calculator. To see how the pricing would change for your particular use case change the appropriate variables to match your expected usage.

We have provided three sample cost profiles based on amount of traffic you expect to get:

- **Small:** In this profile, we're using a single `Standard S1` Web App to host the website, the free tier of the Azure Bot service, a single `Basic` Azure Search service, and a `Standard S2` SQL Database.
- **Medium:** Here we are scaling up the Web App to two instances of the `Standard S3` tier, upgrading the Search Service to a `Standard S1` tier, and using a `Standard S6` SQL Database.
- **Large:** In the largest profile, we use four instances of a `Premium P2V2` Web App, upgrade the Azure Bot service to the `Standard S1` tier (with 1.000.000 messages in Premium channels), use 2 units of the `Standard S3` Azure Search service, and a `Premium P6` SQL Database.

Related resources

To learn more about Azure Search, visit the [documentation center](#), check out the [samples](#), or see a full fledged [demo site](#) in action.