# Learn about storage and predictable performance provisioning in DocumentDB

DocumentDB is a fully managed, scalable document oriented NoSQL database service for JSON documents. With DocumentDB, you don’t have to rent virtual machines, deploy software, monitor databases or worry about disaster recovery. DocumentDB is operated and continuously monitored by Microsoft engineers to deliver world class availability, performance, and data protection.

You can get started with DocumentDB by [creating a database account](/documentation/articles/documentdb-create-account) through the [Azure Portal](https://portal.azure.com/). DocumentDB is offered in units of solid-state drive (SSD) backed storage and throughput. These units are provisioned by creating database collections within your database account. Each collection with reserved throughput. If the throughput requirements of your application change, you dynamically change this by setting the [performance level](/documentation/articles/documentdb-performance-levels) for each collection.

When your application exceeds performance levels for one or multiple collections, requests will be throttled on a per collection basis. This means that some application requests may succeed while others may be throttled.

This article provides an overview of the resources and metrics available to manage capacity and plan data storage.

## Database account and administrative resources

As an Azure subscriber, you can provision one or more DocumentDB database accounts. Each database account comes with a quota of administrative resources including databases, users, and permissions. These resources are subject to [limits and quotas](/documentation/articles/documentdb-limits). If you need additional administrative resources, please [contact support](/documentation/articles/documentdb-increase-limits).

The Azure Portal provides usage metrics for monitoring database accounts, databases and collections. For more information, see [Monitor a DocumentDB account](/documentation/articles/documentdb-monitor-accounts).

## Databases

A single DocumentDB database can contain practically an unlimited amount of document storage grouped into by collections. Collections provide performance isolation - each collection can be provisioned with throughput that is not shared with other collections in the same database or account. A DocumentDB database is elastic in size, ranging from GBs to TBs of SSD backed document storage and provisioned throughput. Unlike a traditional RDBMS database, a database in DocumentDB is not scoped to a single machine and can span multiple machines or clusters.

With DocumentDB, as you need to scale your applications, you can create more collections or databases or both.

## Database collections

Each DocumentDB database can contain one or more collections. Collections act as highly available data partitions for document storage and processing. Each collection can store documents with heterogeneous schema. DocumentDB’s automatic indexing and query capabilities allow you to easily filter and retrieve documents. A collection provides the scope for document storage and query execution. A collection is also a transaction domain for all the documents contained within it. Collections are allocated throughput based on their specified performance level. This can be adjusted dynamically through the Azure Portal or one of the SDKs.

Collections are automatically partitioned into one or more physical servers by DocumentDB. When you create a collection, you can specify the provisioned throughput in terms of request units per second and a partition key property. The value of this property will be used by DocumentDB to distribute documents among partitions and route requests like queries. The partition key value also acts as the transaction boundary for stored procedures and triggers. Each collection has a reserved amount of throughput specific to that collection, which is not shared with other collections in the same account. Therefore, you can scale out your application both in terms of storage and throughput.

Collections can be created through the [Azure Portal](https://portal.azure.com/) or through any one of the [DocumentDB SDKs](/documentation/articles/documentdb-sdk-dotnet).

## Request units and database operations

When you create a collection, you reserve throughput in terms of request units (RU) per second. Instead of thinking about and managing hardware resources, you can think of a **request unit** as a single measure for the resources required to perform various database operations and service an application request. A read of a 1 KB document consumes the same 1 RU regardless of the number of items stored in the collection or the number of concurrent requests running at the same. All requests against DocumentDB, including complex operations like SQL queries have a predictable RU value which can be determined at development time. If you know the size of your documents and the frequency of each operation (reads, writes and queries) to support for your application, you can provision the exact amount of throughput to meet your application’s needs, and scale your database up and down as your performance needs change.

Each collection can be reserved with throughput in blocks of 100 request units per second, from hundreds up to millions of request units per second. The provisioned throughput can be adjusted throughout the life of a collection to adapt to the changing processing needs and access patterns of your application. For more information, see [DocumentDB performance levels](/documentation/articles/documentdb-performance-levels).

[AZURE.IMPORTANT] Collections are billable entities. The cost is determined by the provisioned throughput of the collection measured in request units per second along with the total consumed storage in gigabytes.

How many request units will a particular operation like insert, delete, query, or stored procedure execution consume? A request unit is a normalized measure of request processing cost. A read of a 1 kB document is 1 RU, but a request to insert, replace or delete the same document will consume more processing from the service and thereby more request units. Each response from the service includes a custom header (x-ms-request-charge) that reports the request units consumed for the request. This header is also accessible through the [SDKs](/documentation/articles/documentdb-sdk-dotnet). In the .NET SDK, RequestCharge is a property of the ResourceResponse object.

[AZURE.NOTE] The baseline of 1 request unit for a 1 kB document corresponds to a simple GET of the document with Session Consistency.

There are several factors that impact the request units consumed for an operation against a DocumentDB database account. These factors include:

* Document size. As document sizes increase the units consumed to read or write the data will also increase.
* Property count. Assuming default indexing of all properties, the units consumed to write a document will increase as the property count increases.
* Data consistency. When using data consistency levels of Strong or Bounded Staleness, additional units will be consumed to read documents.
* Indexed properties. An index policy on each collection determines which properties are indexed by default. You can reduce your request unit consumption by limiting the number of indexed properties.
* Document indexing. By default each document is automatically indexed, you will consume fewer request units if you choose not to index some of your documents.

For more information, see [DocumentDB request units](/documentation/articles/documentdb-request-units).

For example, here’s a table that shows how many request units to provision at three different document sizes (1KB, 4KB, and 64KB) and at two different performance levels (500 reads/second + 100 writes/second and 500 reads/second + 500 writes/second). The data consistency was configured at Session, and the indexing policy was set to None.

<tbody>  
 <tr>  
 <td valign="top"><p><strong>Document size</strong></p></td>  
 <td valign="top"><p><strong>Reads/second</strong></p></td>  
 <td valign="top"><p><strong>Writes/second</strong></p></td>  
 <td valign="top"><p><strong>Request units</strong></p></td>  
 </tr>  
 <tr>  
 <td valign="top"><p>1 KB</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>100</p></td>  
 <td valign="top"><p>(500 \* 1) + (100 \* 5) = 1,000 RU/s</p></td>  
 </tr>  
 <tr>  
 <td valign="top"><p>1 KB</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>(500 \* 5) + (100 \* 5) = 3,000 RU/s</p></td>  
 </tr>  
 <tr>  
 <td valign="top"><p>4 KB</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>100</p></td>  
 <td valign="top"><p>(500 \* 1.3) + (100 \* 7) = 1,350 RU/s</p></td>  
 </tr>  
 <tr>  
 <td valign="top"><p>4 KB</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>(500 \* 1.3) + (500 \* 7) = 4,150 RU/s</p></td>  
 </tr>  
 <tr>  
 <td valign="top"><p>64 KB</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>100</p></td>  
 <td valign="top"><p>(500 \* 10) + (100 \* 48) = 9,800 RU/s</p></td>  
 </tr>  
 <tr>  
 <td valign="top"><p>64 KB</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>500</p></td>  
 <td valign="top"><p>(500 \* 10) + (500 \* 48) = 29,000 RU/s</p></td>  
 </tr>  
</tbody>

Queries, stored procedures, and triggers consume request units based on the complexity of the operations being performed. As you develop your application, inspect the request charge header to better understand how each operation is consuming request unit capacity.

## Choice of consistency level and throughput

The choice of default consistency level has an impact on the throughput and latency. You can set the default consistency level both programmatically and through the Azure Portal. You can also override the consistency level on a per request basis. By default, the consistency level is that of session which provides monotonic read/writes and read your write guarantees. Session consistency is great for user-centric applications and provides an ideal balance of consistency and performance trade-offs.

For instructions on changing your consistency level on the Azure Portal, see [How to Manage a DocumentDB Account](/documentation/articles/documentdb-manage-account#consistency). Or, for more information on consistency levels, see [Using consistency levels](/documentation/articles/documentdb-consistency-levels).

## Provisioned document storage and index overhead

DocumentDB supports the creation of both single-partition and partitioned collections. Each partition in DocumentDB supports up to 10 GB of SSD backed storage. The 10GB of document storage includes the documents plus storage for the index. By default, a DocumentDB collection is configured to automatically index all of the documents without explicitly requiring any secondary indices or schema. Based on applications using DocumentDB, the typical index overhead is between 2-20%. The indexing technology used by DocumentDB ensures that regardless of the values of the properties, the index overhead does not exceed more than 80% of the size of the documents with default settings.

By default all documents are indexed by DocumentDB automatically. However, if you want to fine tune the index overhead, you can chose to remove certain documents from being indexed at the time of inserting or replacing a document, as described in [DocumentDB indexing policies](/documentation/articles/documentdb-indexing-policies). You can configure a DocumentDB collection to exclude all documents within the collection from being indexed. You can also configure a DocumentDB collection to selectively index only a certain properties or paths with wildcards of your JSON documents, as described in [Configuring the indexing policy of a collection](/documentation/articles/documentdb-indexing-policies#configuring-the-indexing-policy-of-a-collection). Excluding properties or documents also improves the write throughput – which means you will consume fewer request units.  
## Next steps For instructions on monitoring performance levels on the Azure Portal, see [Monitor a DocumentDB account](/documentation/articles/documentdb-monitor-accounts).

For more information on choosing performance levels for collections, see [Performance levels in DocumentDB](/documentation/articles/documentdb-performance-levels).