100 XP

Provision Azure Cosmos DB

12 minutes

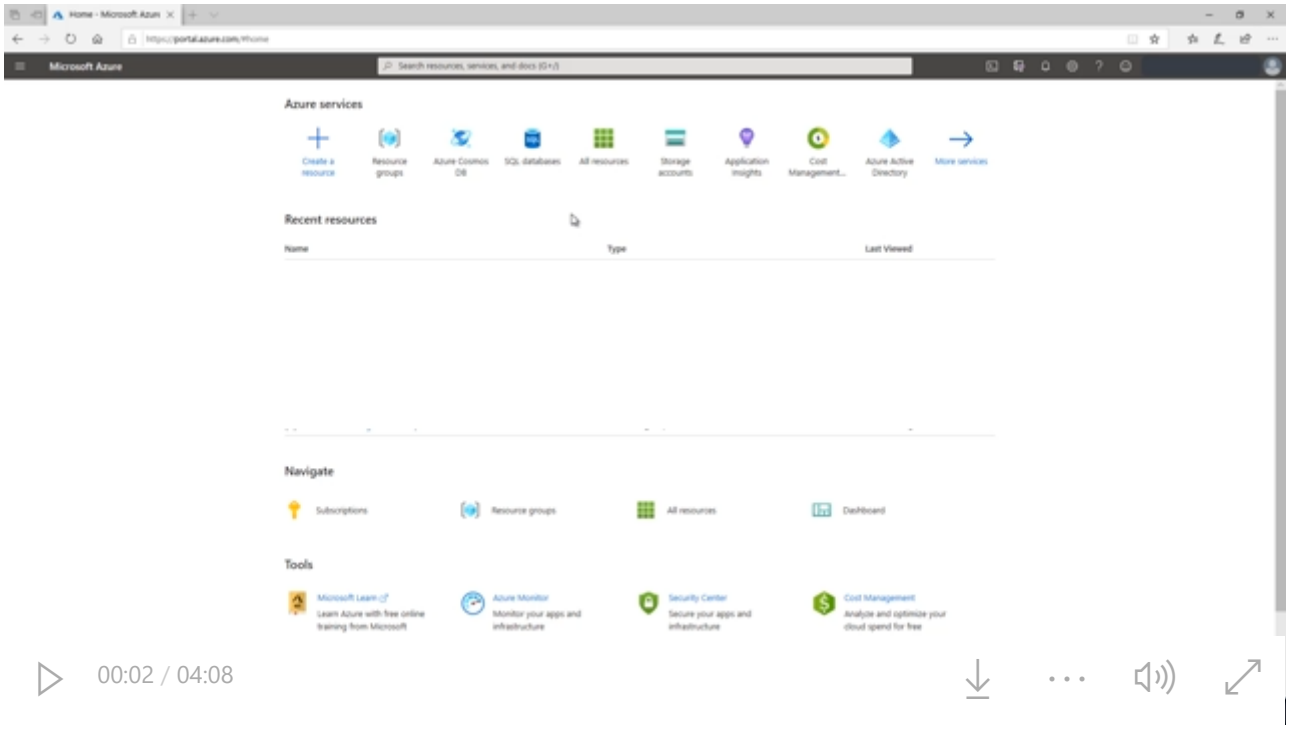
Azure Cosmos DB is a document database, suitable for a range of applications. In the sample scenario, Contoso decided to use Cosmos DB for at least part of their data storage and processing.

In Cosmos DB, you organize your data as a collection of documents stored in containers. Containers are held in a database. A database runs in the context of a Cosmos DB account. You must create the account before you can set up any databases.

This unit describes how to provision a Cosmos DB account, and then create a database and a container in this account.

How to provision a Cosmos DB account

You can provision a Cosmos DB account interactively using the Azure portal, or you can perform this task programmatically through the Azure CLI, Azure PowerShell, or an Azure Resource Manager template. The following video describes how to use the Azure portal.



If you prefer to use the Azure CLI or Azure PowerShell, you can run the following commands to create a Cosmos DB account. The parameters to these commands correspond to many of the options you can select using the Azure portal. The examples shown below create an account for the Core(SQL) API, with geo-redundancy between the EastUS and WestUS regions,

and support for multi-region writes. For more information about these commands, see the az cosmosdb create page for the Azure CLI, or the [New-AzCosmosDBAccount page for PowerShell.

Azure CLI Copy

```
## Azure CLI

az cosmosdb create \
  --subscription <your-subscription> \
  --resource-group <resource-group-name> \
  --name <cosmosdb-account-name> \
  --locations regionName=eastus failoverPriority=0 \
  --locations regionName=westus failoverPriority=1 \
  --enable-multiple-write-locations
```

PowerShell Copy

```
## Azure PowerShell

New-AzCosmosDBAccount `
  -ResourceGroupName "<resource-group-name>" `
  -Name "<cosmosbd-account-name>" `
  -Location @("West US", "East US") `
  -EnableMultipleWriteLocations
```

Note

To use Azure PowerShell to provision a Cosmos DB account, you must first install the Az.CosmosDB PowerShell module:

PowerShell Copy

```
Install-Module -Name Az.CosmosDB
```

The other deployment option is to use an Azure Resource Manager template. The template for Cosmos DB can be rather lengthy, because of the number of parameters. To make life easier, Microsoft has published a number of example templates for handling different configurations. You can download these templates from the Microsoft web site, at Manage Azure Cosmos DB Core (SQL) API resources with Azure Resource Manager templates.

How to create a database and a container

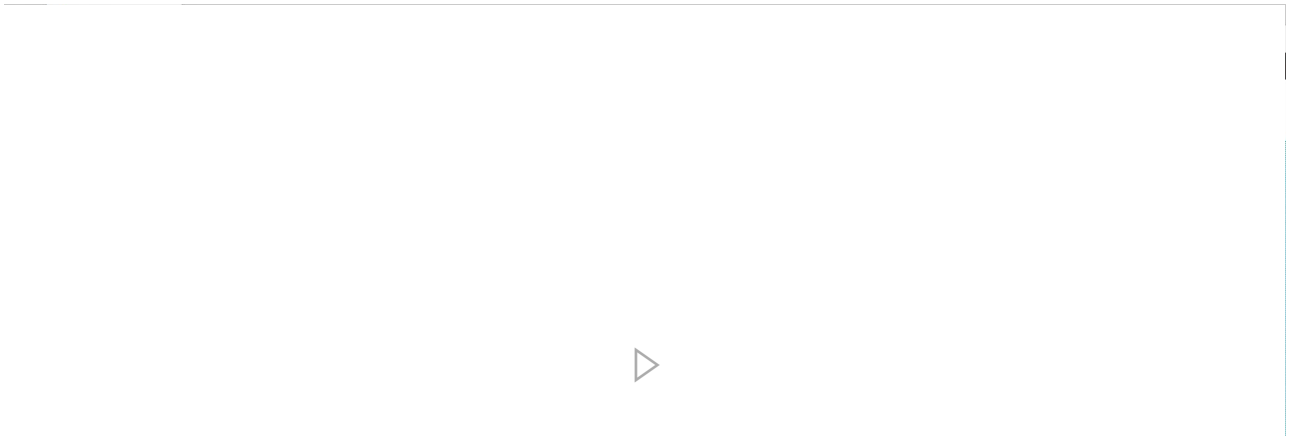
An Azure Cosmos DB account by itself doesn't really provide any resources other than a few pieces of static infrastructure. Databases and containers are the primary resource consumers. Resources are allocated in terms of the storage space required to hold your databases and containers, and the processing power required to store and retrieve data. Azure Cosmos DB uses the concept of Request Units per second (RU/s) to manage the performance and cost of databases. This measure abstracts the underlying physical resources that need to be provisioned to support the required performance.

You can think of a request unit as the amount of computation and I/O resources required to satisfy a simple read request made to the database. Microsoft gives a measure of approximately one RU as the resources required to read a 1-KB document with 10 fields. So a throughput of one RU per second (RU/s) will support an application that reads a single 1-KB document each second. You can specify how many RU/s of throughput you require when you create a database or when you create individual containers in a database. If you specify throughput for a database, all the containers in that database share that throughput. If you specify throughput for a container, the container gets that throughput all to itself.

If you underprovision (by specifying too few RU/s), Cosmos DB will start throttling performance. Once throttling begins, requests will be asked to retry later when hopefully there are available resources to satisfy it. If an application makes too many attempts to retry a throttled request, the request could be aborted. The minimum throughput you can allocate to a database or container is 400 RU/s. You can increase and decrease the RU/s for a container at any time. Allocating more RU/s increases the cost. However, once you allocate throughput to a database or container, you'll be charged for the resources provisioned, whether you use them or not.

> **Note**
>
> If you applied the Free Tier Discount to your Cosmos DB account, you get the first 400 RU/s for a single database or container for free. 400 RU/s is enough capacity for most small to moderate databases.

The next video shows how to use the Azure portal to create a database and container:



If you prefer to use the Azure CLI or Azure PowerShell, you can run the following commands to create documents and containers. The code below shows some examples:

Azure CLI Copy

```
## Azure CLI - create a database

az cosmosdb sql database create \
  --account-name <cosmos-db-account-name> \
  --name <database-name> \
  --resource-group <resource-group-name> \
  --subscription <your-subscription> \
  --throughput <number-of-RU/s>

## Azure CLI - create a container

az cosmosdb sql container create \
  --account-name <cosmos-db-account-name> \
  --database-name <database-name> \
  --name <container-name> \
  --resource-group <resource-group-name> \
  --partition-key-path <key-field-in-documents>
```

PowerShell Copy

```
## Azure PowerShell - create a database

New-AzCosmosDBSqlDatabase `
    -ResourceGroupName "<resource-group-name>" `
    -AccountName "<cosmos-db-account-name>" `
    -Name "<database-name>" `
    -Throughput <number-of-RU/s>

## Azure PowerShell - create a container

New-AzCosmosDBSqlContainer `
    -ResourceGroupName "<resource-group-name>" `
    -AccountName "<cosmos-db-account-name>" `
    -DatabaseName "<database-name>" `
    -Name "<container-name>" `
    -PartitionKeyKind Hash `
    -PartitionKeyPath "<key-field-in-documents>"
```

Next unit: Provision other non-relational data services

Continue >