https://www.c-sharpcorner.com/article/an-overview-of-cosmos-db-start-from-scratch/

<https://stackify.com/what-is-azure-cosmos-db/>

<https://cosmos.azure.com/>

<https://docs.microsoft.com/en-us/azure/cosmos-db/create-sql-api-dotnet>

# Set variables for the new SQL API account, database, and container

resourceGroupName='myResourceGroup'

location='southcentralus'

# The Azure Cosmos account name must be globally unique, make sure to update the `mysqlapicosmosdb` value before you run the command

accountName='mysqlapicosmosdb'

# Create a resource group

az group create \

--name $resourceGroupName \

--location $location

# Create a SQL API Cosmos DB account with session consistency and multi-master enabled

az cosmosdb create \

--resource-group $resourceGroupName \

--name $accountName \

--kind GlobalDocumentDB \

--locations regionName="South Central US" failoverPriority=0 --locations regionName="North Central US" failoverPriority=1 \

--default-consistency-level "Session" \

--enable-multiple-write-locations true

Create a Cosmos Account

Comos DB is a database service that is globally distributed. It allows you to manage your data even if you keep them in data centers that are scattered throughout the world. It provides the tools you need to scale both global distribution pattern and computational resources, and these tools are provided by Microsoft Azure.

It can [support multiple data models](https://azure.microsoft.com/en-us/services/cosmos-db/) using one backend. This means that it can be used for document, key value, relational, and graph models. It is more or less a NoSQL database because it does not rely on any schemas. However, because it uses query language similar to SQL and can easily support [ACID transactions](https://en.wikipedia.org/wiki/ACID), some people have been classifying it as a NewSQL type of database. What differentiates it from other NewSQL databases, however, is that it does not have a relational data model.

### What Problem Does It Solve?

Traditionally, it would take you a long time and a lot of tedious effort to build a similar globally distributed database that you would need to host on your own data centers, using your own connections and other resources. The planning in itself would deter most companies from even thinking about such a setup; many would think that it wouldn’t be worth the investment. What happens is that most companies would rather opt out of it than take advantage of the business and growth opportunities that such a database system can offer.

When cloud computing and platform-as-a-service came along, however, it became easier to come up with a globally distributed and scalable database for your company. So easy that it is possible for just one person to architect and manage it just with a few clicks of the mouse. [Azure Cosmos DB](https://docs.microsoft.com/en-us/azure/cosmos-db/introduction) takes it to the next level, giving you a turnkey database system that you can scale according to your needs.

In sum, it will help any enterprise establish a flexible database that would help them meet their business needs. But it is especially useful for companies that are looking for a database system that is scalable and globally distributed. Globally distributed means that all resources are partitioned horizontally in every region of the world, as well as replicated across different geographical areas. That means latency is at a minimum, and your users have a faster, seamless experience. Azure Cosmos DB offers 99.99% availability.

### Additional Features

Users also get to reap the benefits of several other features including:

* **Complete service and ready to use**: It gives you a complete product that is powered by Azure and can be automatically replicated in data centers worldwide.
* **Multi-API**: Because data is indexed automatically, users can access it using any API of their choice. They can see their data using SQL, Gremlin, JavaScript, Azure Table Storage, and MongoDB.
* **A number of consistency levels**: It uses five different consistency levels: bounded staleness, strong, session, eventual, and consistent-prefix.
* **Latency**: Very low latency is practically guaranteed at less than 10 milliseconds when reading data and less than 15 milliseconds when writing data.