

CASO PRACTICO – BENEFICIOS DE LA DESNORMALIZACION EN COSMOS DB

1-Creamos el recurso de CosmosDB, y seleccionamos la api con el se va a trabajar CORE SQL

[Inicio](#) > [udemyCourse](#) > [Marketplace](#) > [Azure Cosmos DB](#) >

Selección de la opción de API

¿Qué API resulta más adecuada para la carga de trabajo?

Azure Cosmos DB es un servicio de base de datos NoSQL totalmente administrado para la creación de aplicaciones escalables y de alto rendimiento. [Learn more](#)
Para empezar, seleccione la API para crear una cuenta nueva. La selección de la API no se puede cambiar tras crear la cuenta.

Núcleo (SQL): opción recomendada Núcleo o API nativa de Azure Cosmos DB para trabajar con documentos. Permite un desarrollo rápido y flexible con el lenguaje de consultas SQL y las bibliotecas cliente para .NET, JavaScript, Python y Java que ya conoce. Crear Más información	API de Azure Cosmos DB para MongoDB Servicio de base de datos de MongoDB totalmente administrado para aplicaciones escritas para MongoDB. Se recomienda si dispone de cargas de trabajo existentes de MongoDB que tenga previsto migrar a Azure Cosmos DB. Crear Más información	Cassandra Servicio de base de datos administrado para aplicaciones escritas para Cassandra. Se recomienda si dispone de cargas de trabajo existentes de Cassandra Cosmos DB. Crear Más información
Tabla de Azure Servicio de base de datos totalmente administrado para aplicaciones escritas para el almacenamiento de Azure Table. Se recomienda si dispone de cargas de trabajo existentes del almacenamiento de Azure Table que tenga previsto migrar a Azure Cosmos DB, pero no quiere volver a escribir la aplicación para usar la API SQL. Crear Más información	Gremlin (grafo) Servicio de base de datos de grafos totalmente administrado que emplea el lenguaje de consultas Gremlin, basado en el proyecto Apache TinkerPop. Se recomienda para nuevas cargas de trabajo en las que sea necesario almacenar relaciones entre datos. Crear Más información	Azure Cosmos DB f Fully-managed relational distributed query execution source extension. Build r clusters—with support for and high-performance s Crear Más información

2.-Seleccionamos el grupo de recursos, asignamos el nombre, y creamos el recurso. Al terminar de crearse seleccionamos go to resource.

[Inicio](#) > [Microsoft.Azure.CosmosDB-20221118125754](#) | Información general

Implementación

Buscar << [Eliminar](#) [Cancelar](#) [Volver a implementar](#) [Descargar](#) [Actualizar](#)

Información general

Entradas

Salidas

Plantilla

Se completó la implementación

Nombre de implementación: Microsoft.Azure.CosmosDB-20221118... Hora de inicio
Suscripción: [Azure for Students](#) Id. de correlac
Grupo de recursos: [udemyCourse](#)

Detalles de implementación

Pasos siguientes

[Ir al recurso](#)

3.- Creamos la base de datos, nos vamos a la opción data explorer

[Inicio](#) > [Microsoft.Azure.CosmosDB-20221118125754](#) | Información general > [cosmosaccountbluetab](#)

cosmosaccountbluetab | Explorador de datos

Cuenta de Azure Cosmos DB

Buscar << [New Container](#) [Enable Azure Synapse](#)

Etiquetas

Diagnosticar y solucionar problemas

Cost Management

Inicio rápido

Notificaciones

[New Container](#)

[New Database](#)

NOTEBOOKS

Notebooks is currently not available. We are working on it

New Database

Database id

☒ Provision throughput

☒ Database throughput (autoscale)

☒ Autoscale ☐ Manual

Estimate your required RU/s with [capacity calculator](#).

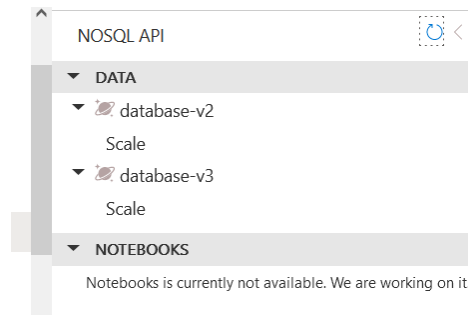
Database Max RU/s

Your database throughput will automatically scale from **100 RU/s** (10% of max RU/s) - **1000 RU/s** based on usage.

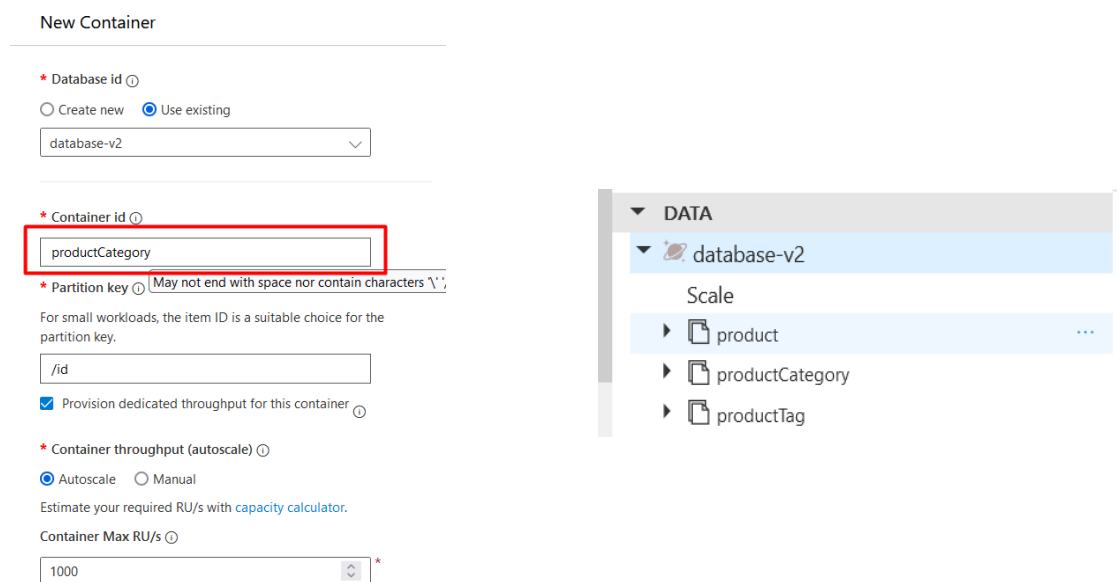
Estimated monthly cost (USD) (1 region, 100 - 1000 RU/s, \$0.00012/RU)

[OK](#)

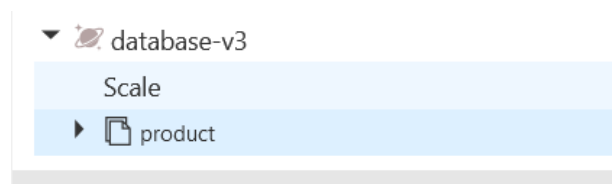
4.- Luego que ya tenemos creamos las bases de datos, refrescamos y nos aparecerá las dos bases de datos.



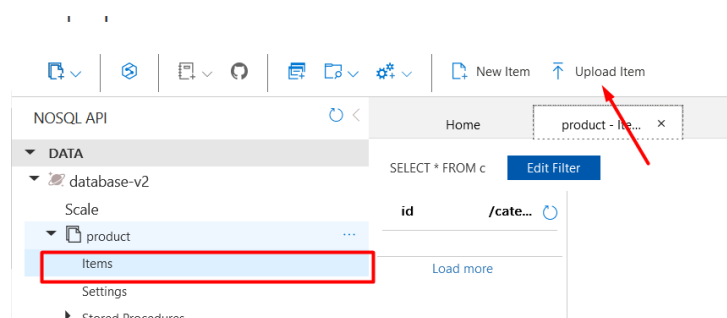
5.- Creando los contenedores database-v2 y verificamos que se haya creado. Repetimos los mismos pasos para producto y productTag



6.- Creamos los contenedores en database-v3,



7.- Nos redirigimos al contenedor producto del database-v2 y upload item



8.- Consulta a base de datos normalizada

productCategory

1

`SELECT * FROM c where c.type = 'category' and c.id = "AB952F9F-5ABA-4251-BC2D-AFF8DF412A4A"`

Results

Query Stats

Query Statistics

METRIC	VALUE
Request Charge	2.92 RUs

product

NOSQL API

HomeQuery 1Query 2

DATA

database-v2

Scale

product

productCategory

productTag

database-v3

Scale

product

NOTEBOOKS

1

`SELECT * FROM c where c.categoryId = "AB952F9F-5ABA-4251-BC2D-AFF8DF412A4A"`

Results

Query Stats

Query Statistics

METRIC	VALUE
Request Charge	2.89 RUs
Showing Results	1 - 3

productTag

Execute QuerySave Query

NOSQL API

HomeQuery 1Query 2Query 3

DATA

database-v2

Scale

product

productCategory

productTag

database-v3

Scale

product

NOTEBOOKS

1

`SELECT * FROM c where c.type = 'tag' and c.id IN ('87BC6842-2CCA-4CD3-994C-33AB101455F4', 'F07885AF-BD6C-4B71-88B1-F04295992176')`

2

Results

Query Stats

Query Statistics

METRIC	VALUE
Request Charge	3.02 RUs

NOSQL API

Home Query 1 Query 2 Query 3

DATA

- database-v2
 - Scale
 - product
 - productCategory
 - productTag
- database-v3
 - Scale
 - product

NOTEBOOKS

1 SELECT * FROM c where c.type = 'tag' and c.id IN ('18AC309F-F81C-4234-A752-5DD02BEAEE83', '1B387A00-57D3-4444-8331-18A90725E988', 'C6AB3E24-BA48-40F0-A260-CB04EB03D5B0', 'DAC25651-3DD3-4483-8FD1-581DC41EF34B', 'E6D5275B-8C42-47AE-BDEC-FC708D03E0AC')

Results Query Stats

Query Statistics

METRIC	VALUE
Request Charge	3.02 RUs

Requerimiento	RU
Req 1	2.92
Req 2	2.89
Req 3	3.02
Req 4	3.02
	10.85

9.- Consulta a base de datos no normalizada

Home Query 1

1 SELECT * FROM c where c.categoryId = "AB952F9F-5ABA-4251-BC2D-AFF8DF412A4A"

Results Query Stats

Query Statistics

METRIC	VALUE
Request Charge	2.89 RUs

Showing Results 1 - 3

Resumen consumo RU/s para la versión database-v3

Diferencia entre las distintas versiones: $10.85 - 2.89 = 7.96$ (RU/s de ahorro)