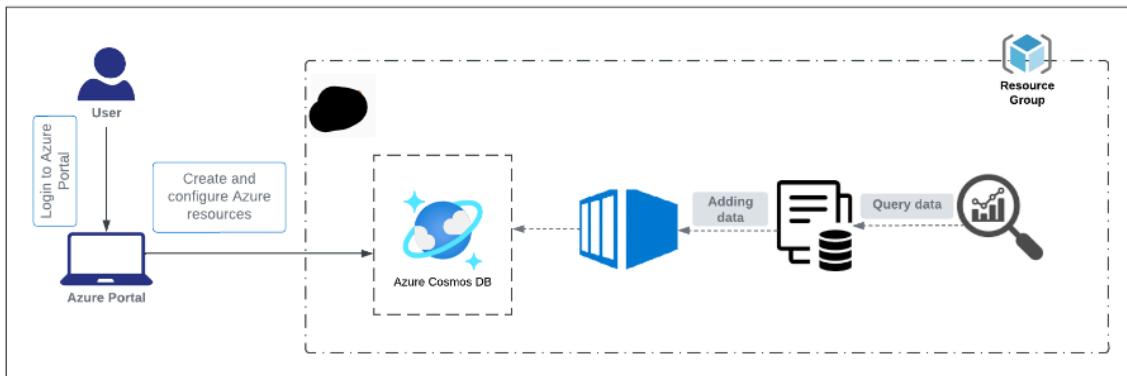


Introduction

What is Azure Cosmos DB ?

- Azure Cosmos DB is a fully managed NoSQL database that can be used for application development.
- It is highly scalable with very low latency.
- Management of Cosmos DB including updating and patching is done automatically without any manual intervention.
- With the help of Cosmos DB SQL API, one can query JSON files using SQL.
- Also, since SQL is quite popular among developers, it becomes pretty easy to query data in Cosmos DB.
- While storing the data in the Cosmos DB database, the data is split into logical partitions, thus making it easy to query the data.

Architecture Diagram



Lab Steps

Task 1: Sign in to Azure Portal

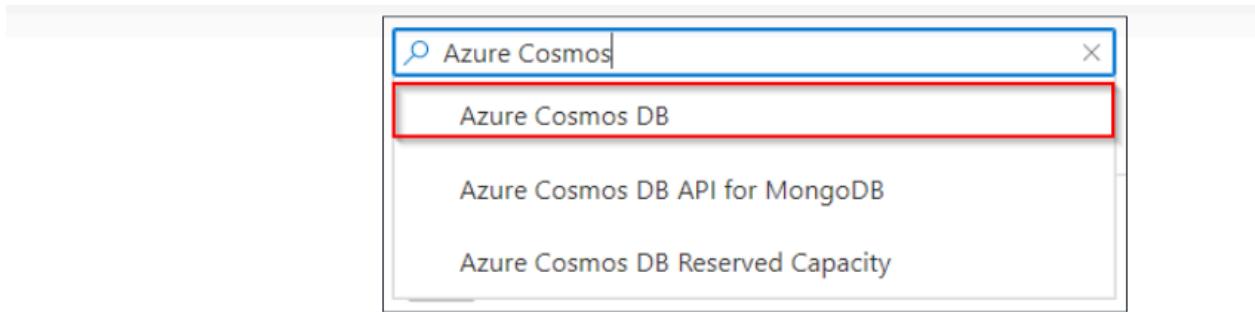
1. Go to the Azure portal by clicking on the **Open Console** button or by using URL <https://portal.azure.com>.
 - **Note:** It is recommended to use incognito mode to avoid Azure portal cache related issues.
2. If it automatically logs into any other azure account, please logout of it and clear cache.
3. Sign in with your given **username** and **password** on Azure portal.
4. If login is not working. Click on **End Lab** and start the lab again.

Task 2: Creating an Azure Cosmos DB Account

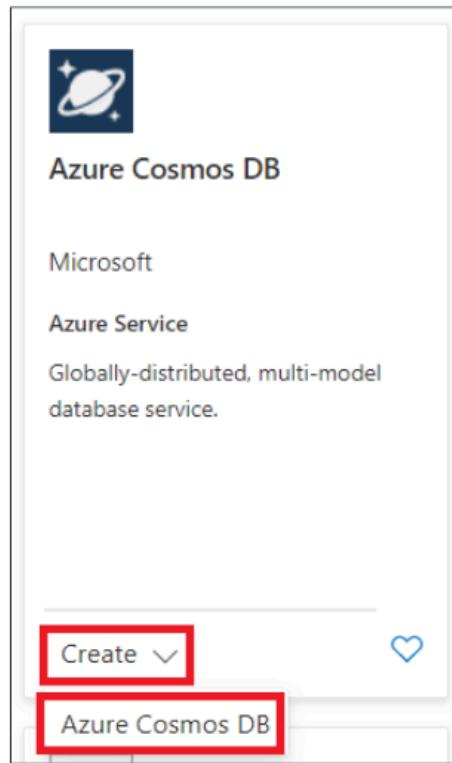
1. The Azure portal menu or from the Home page, select **Create a resource**.



2. On the search bar, enter Azure Cosmos and then select **Azure Cosmos DB**.



3. Click on **Create**.



4. On the **Create an Azure Cosmos DB account** page, click on **Create** under **Azure Cosmos DB for NoSQL**.

The screenshot shows the 'Create an Azure Cosmos DB account' page. At the top, it asks 'Which API best suits your workload?'. Below this, there are five options:

- Azure Cosmos DB for NoSQL**: Described as 'Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.' A 'Create' button is highlighted with a red box.
- Azure Cosmos DB for PostgreSQL**: Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.
- Azure Cosmos DB for MongoDB**: Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.
- Azure Cosmos DB for Apache Cassandra**: Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.
- Azure Cosmos DB for Table**: Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB.
- Azure Cosmos DB for Apache Gremlin**: Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data.

Each option has a 'Create' and 'Learn more' button.

5. Now, fill out the page with the following details:

- Resource group: Select **rg_westus_XXXXX**
- Account Name: Enter **whizcosmosdb**
- Apply Free Tier Discount: Select **Do Not Apply**

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Pay-As-You-Go

Resource Group * rg_westus_48231_1_1677935646379
Create new

Instance Details

Account Name * whizcosmosdb

Location * (US) West US

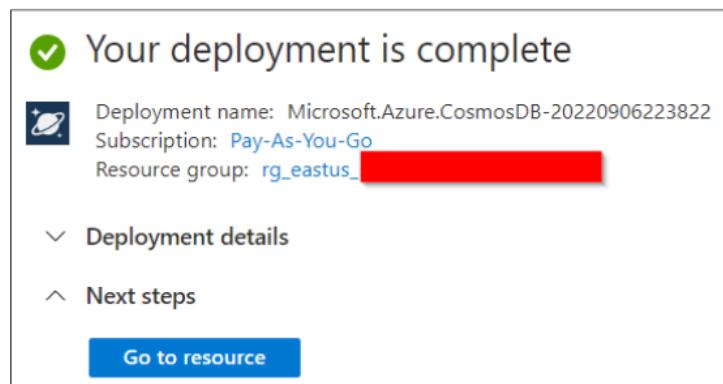
Capacity mode Provisioned throughput Serverless
[Learn more about capacity mode](#)

With Azure Cosmos DB free tier, you will get the first 1000 RU/s and 25 GB of storage for free in an account. You can enable free tier on up to one account per subscription. Estimated \$64/month

Apply Free Tier Discount Apply Do Not Apply

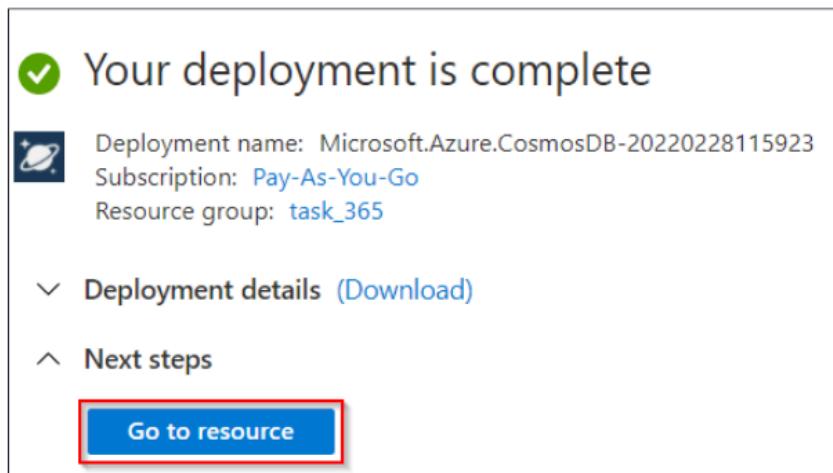
Limit total account throughput Limit the total amount of throughput that can be provisioned on this account
 ⓘ This limit will prevent unexpected charges related to provisioned throughput. You can update or remove this limit after your account is created.

6. Keeping rest in default, click on **Review+Create** and then click on **Create**.

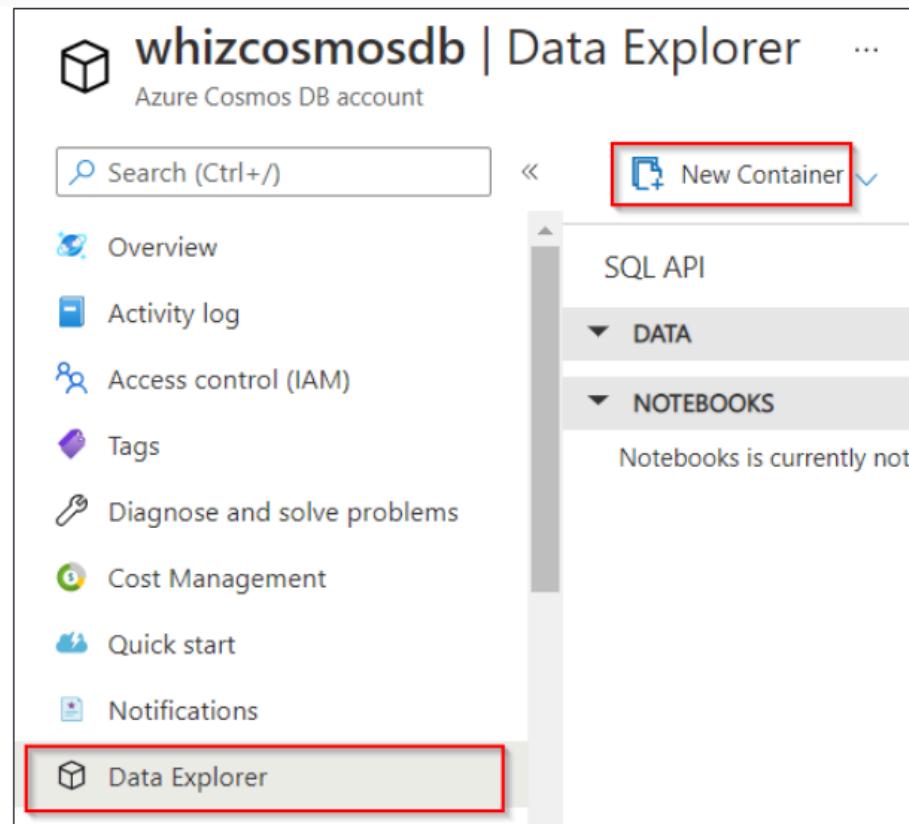


Task 3: Creating container within Cosmos DB Account

1. After successful deployment, click on **Go to Resource**.



2. On the left panel, click on **Data Explorer** and then click on **New Container**.



3. Fill the form with the following information:

- Database id: Enter **WhizID**
- Container id: Enter **Whizcustomer**
- Partition key: Enter **/Whizcustomercity**
- Click on **Ok.**

* Database id ⓘ

Create new Use existing

WhizID

Share throughput across containers ⓘ

* Container id ⓘ

Whizcustomer

* Indexing

Automatic Off

* Partition key ⓘ

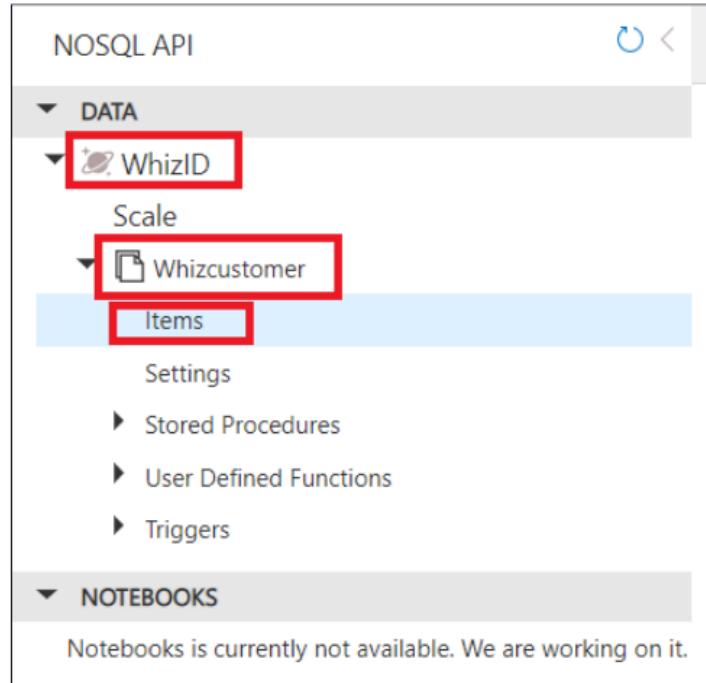
For small workloads, the item ID is a suitable choice for the partition key.

/Whizcustomercity

Unique keys ⓘ

Task 4: Adding default data into database

1. After the successful creation of the container/database, click on **WhizID** (Database ID) and then click on **Whizcustomer** and then on **items**.



2. On the top right corner, click on the Open Full Screen option.



3. Click on **Open** and click on **Sign In**. Then select your **Subscription** and **Cosmos DB Account Name**.

The screenshot shows the Microsoft Azure portal interface for Cosmos DB. At the top, it displays "Microsoft Azure | Cosmos DB > whizcosmosdb". On the left, there's a sidebar with "New Container" and "Enable" buttons, and a "NOSQL API" section. The main area is titled "Subscription" with a dropdown set to "Pay-As-You-Go". Below that is the "Cosmos DB Account Name" section, also with a dropdown set to "whizcosmosdb".

4. You will see a page like below.

The screenshot shows the "Welcome to Cosmos DB" page. The title is "Welcome to Cosmos DB" with the subtitle "Globally distributed, multi-model database service for any scale". There are three main buttons: "Launch quick start", "New Container", and "Connect". Below these are sections for "Recents", "Top 3 things you need to know" (with links to "Advanced Modeling Patterns" and "Get Started using an SDK"), and "Learning Resources".

5. Now, in the new tab, click on **WhizID** (Database ID) and then click on **Whizcustomer** and then on **Items**.

The screenshot shows the Microsoft Azure Cosmos DB NOSQL API interface. At the top, there are buttons for "New Container" and "Enable Azure Synapse Link". Below that, the "NOSQL API" section is visible. A dropdown menu is open under "WhizID", with "Whizcustomer" selected. The "Items" button is highlighted with a red box. Other options in the dropdown include "Scale", "Settings", and "Stored Procedures", "User Defined Functions", "Triggers".

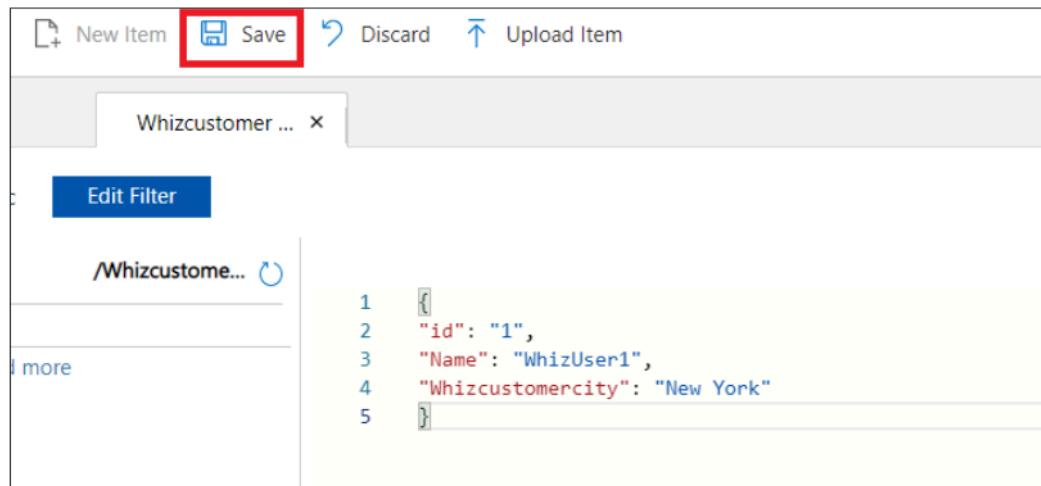
6. On the top panel, click on **New Item**.



7. Delete the text that is already displayed and then enter the following data:

```
{  
    "id": "1",  
    "Name": "WhizUser1",  
    "Whizcustomercity": "New York"  
}
```

8. Click on **Save**.



9. Again click on **New Item** and delete the data that is already displayed and then enter the following data:



```
{  
    "id": "2",  
    "Name": "WhizUser2",  
    "Whizcustomercity": "Atlanta"  
}
```

10. Click on **Save**.

Task 5: Querying the added data

1. After the data is successfully added in the database, click on **New SQL Query** on the top panel to use the query on the saved data.



2. Enter the following query to display all the records stored in the database:



```
SELECT * FROM c
```

3. Click on **Execute Query** on the top panel.



4. All the saved data would be displayed on the screen.

Results Query Stats

1 - 2

```
[{"id": "1", "Name": "WhizUser1", "Whizcustomercity": "New York", "_rid": "MzMqANqv92UBAAAAAAA==", "_self": "dbs/MzMqAA==/colls/MzMqANqv92U=/docs/MzMqANqv92UBAAAAAAA==/", "_etag": "\"07075cfa-0000-0700-0000-621c88240000\"", "_attachments": "attachments/", "_ts": 1646037028}, {"id": "2", "Name": "WhizUser2", "Whizcustomercity": "Atlanta", "_rid": "MzMqANqv92UCAAAAAAA==", "_self": "dbs/MzMqAA==/colls/MzMqANqv92U=/docs/MzMqANqv92UCAAAAAAA==/", "_etag": "\"08070b03-0000-0700-0000-621c8ce40000\"", "_attachments": "attachments/", "_ts": 1646038244}]
```

5. To display records where Whizcustomercity="Atlanta", enter the following command:

```
SELECT * FROM c WHERE c.Whizcustomercity="Atlanta"
```

6. Click on **Execute Query** on the top panel.



7. All the data that satisfies the condition will be displayed.

A screenshot of the Microsoft Power BI interface. At the top, there is a navigation bar with tabs for 'Items', 'Query 1', and a close button. Below the navigation bar, a query editor shows the following code:

```
1  SELECT * FROM c where c.Whizcustomercity="Atlanta"
```

The main area is titled 'Results' and shows a single row of data:

[{"id": "2", "Name": "WhizUser2", "Whizcustomercity": "Atlanta", "_rid": "MzMqANqv92UCAAAAAAAA==", "_self": "dbs/MzMqAA==/colls/MzMqANqv92U=/docs/MzMqANqv92UCAAAAAAAA==/", "_etag": "\"08070b03-0000-0700-0000-621c8ce40000\"", "_attachments": "attachments/", "_ts": 1646038244}]

The data row contains the following JSON object:

```
{"id": "2", "Name": "WhizUser2", "Whizcustomercity": "Atlanta", "_rid": "MzMqANqv92UCAAAAAAAA==", "_self": "dbs/MzMqAA==/colls/MzMqANqv92U=/docs/MzMqANqv92UCAAAAAAAA==/", "_etag": "\"08070b03-0000-0700-0000-621c8ce40000\"", "_attachments": "attachments/", "_ts": 1646038244}
```

Completion and Conclusions

1. You have successfully logged into Azure Portal.
2. You have successfully created an Azure Cosmos DB Account.
3. You have successfully created a container within Cosmos DB Account
4. You have successfully added default data into database
5. You have successfully queried the added data.
6. You have successfully tested the validation.
7. You have successfully deleted the resources.