Quickstart: Create an Azure Cosmos DB for NoSQL account using the Azure portal

In this quickstart, you create a new Azure Cosmos DB for NoSQL account in the Azure portal. You then use the Data Explorer experience within the Azure portal to create a database and container configuring all required settings. Finally, you add sample data to the container and issue a basic query.

Prerequisites

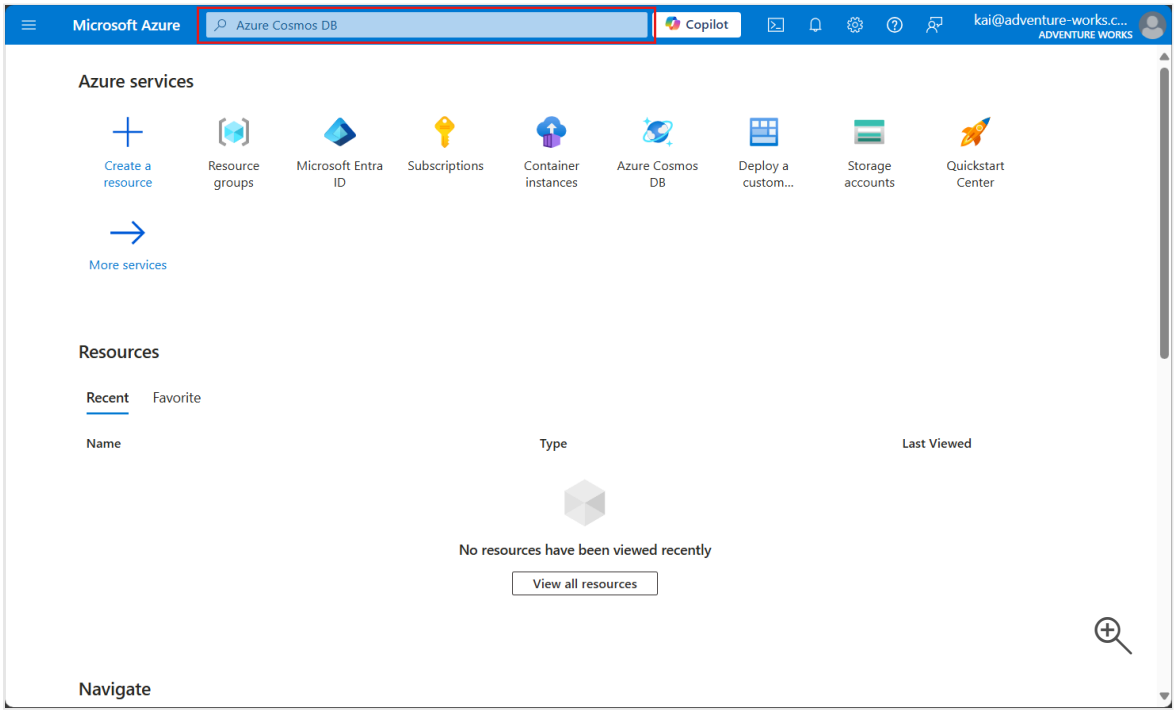
- An Azure account with an active subscription. Create an account for free .

Create an account

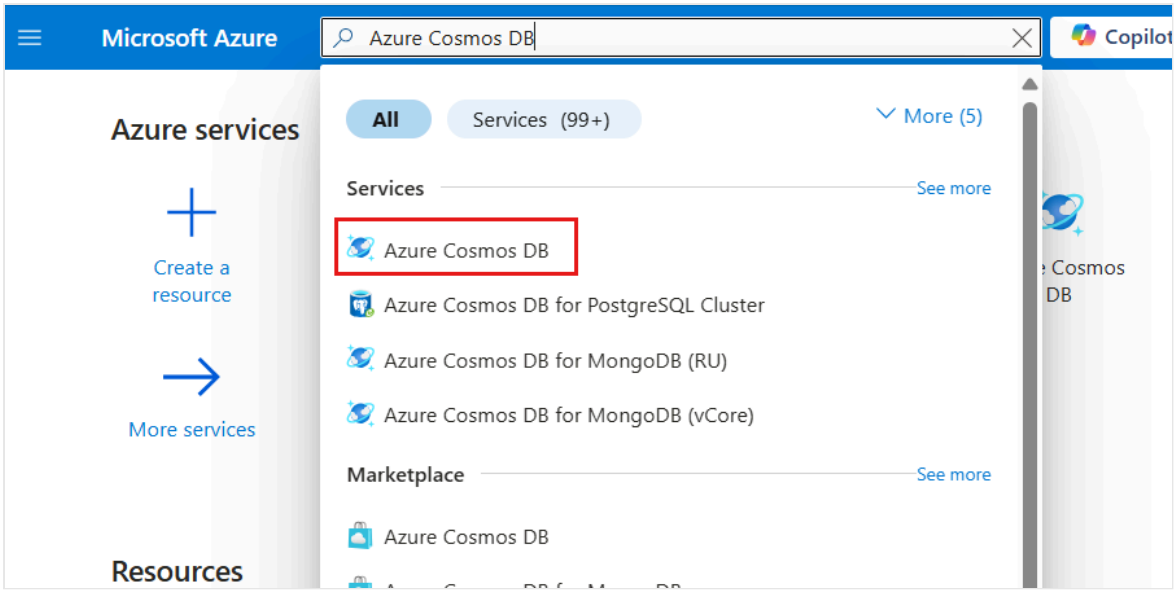
Start by creating a new Azure Cosmos DB for NoSQL account

1. Sign in to the Azure portal (https://portal.azure.com).

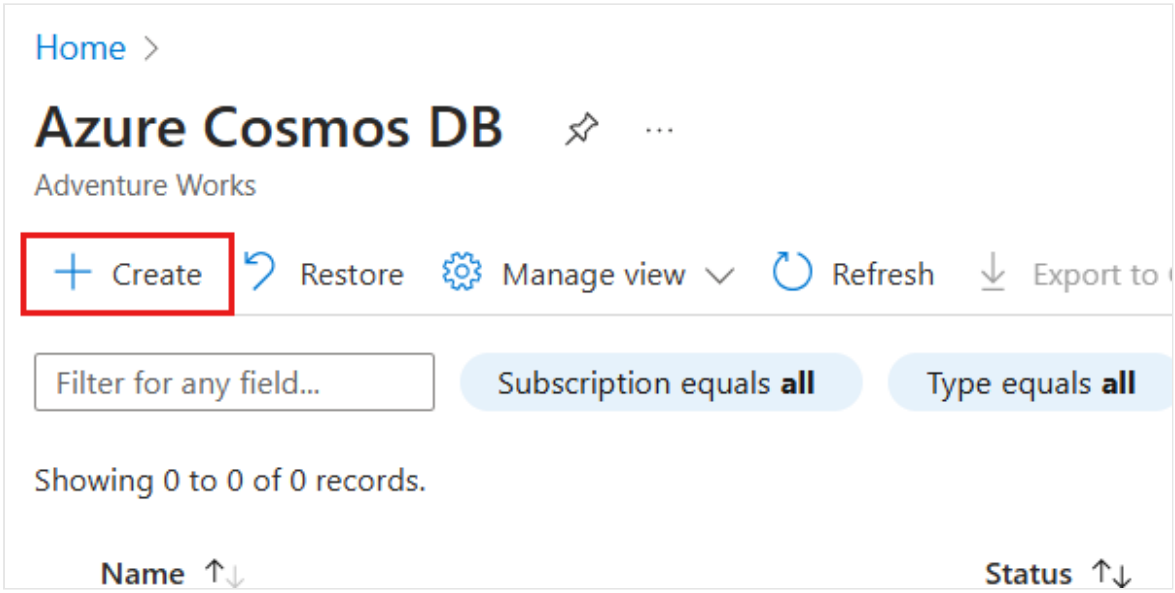
2. Enter *Azure Cosmos DB* in the global search bar.

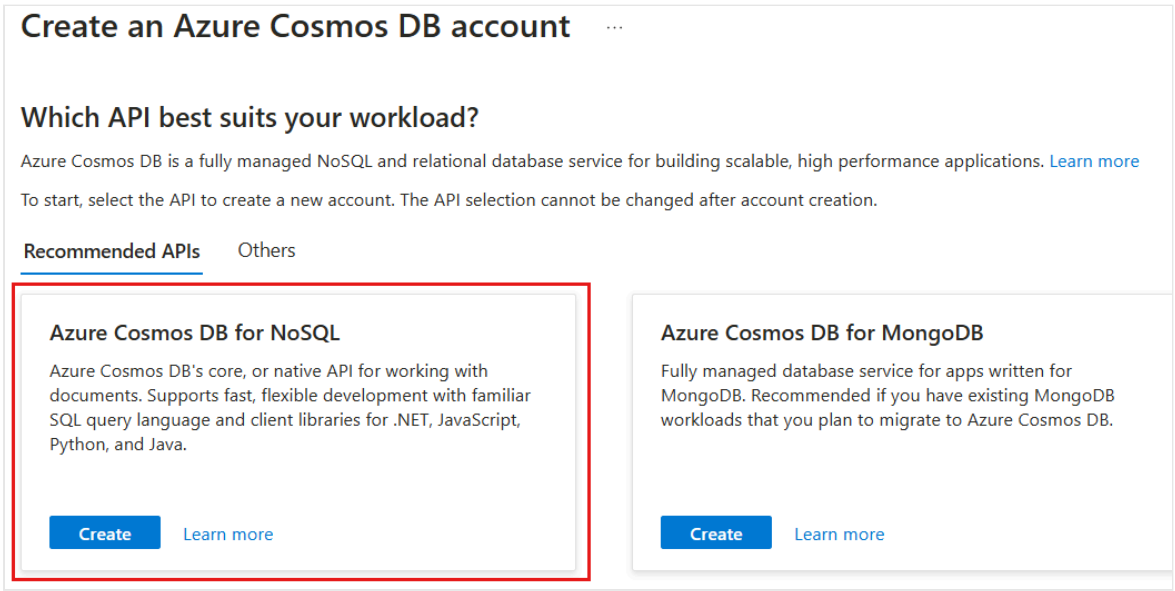


3. Within **Services**, select **Azure Cosmos DB**.



4. In the **Azure Cosmos DB** pane, select **Create**, and then **Azure Cosmos DB for NoSQL**.

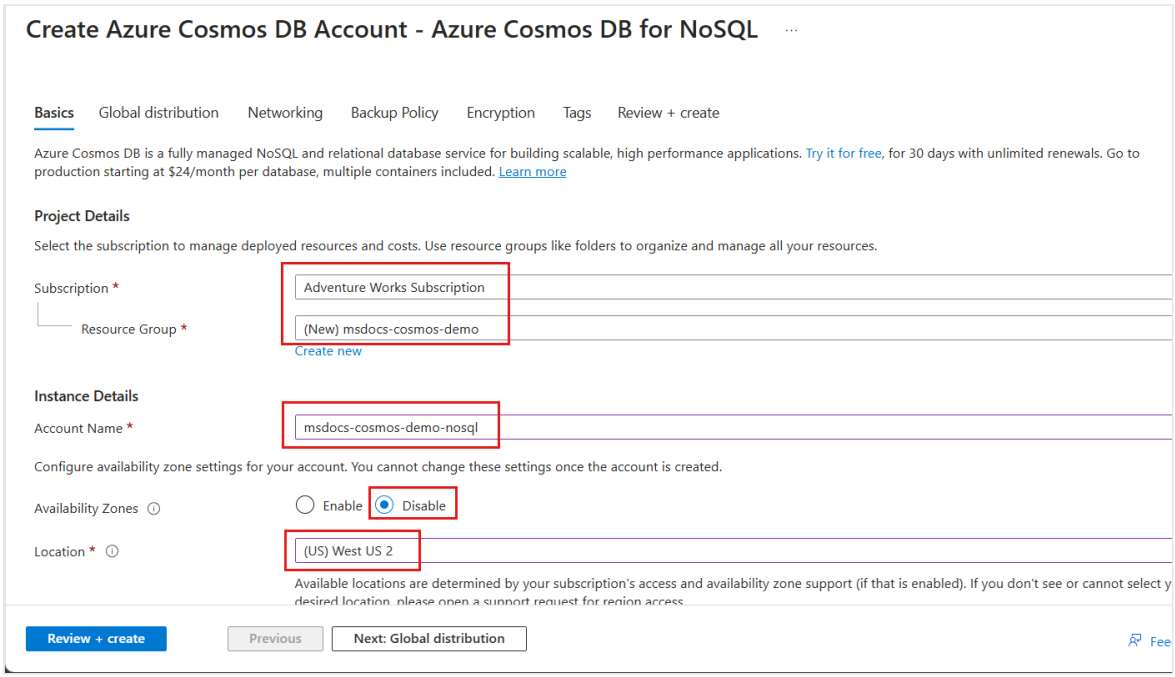




5. Within the **Basics** pane, configure the following options, and then select **Review + create**:

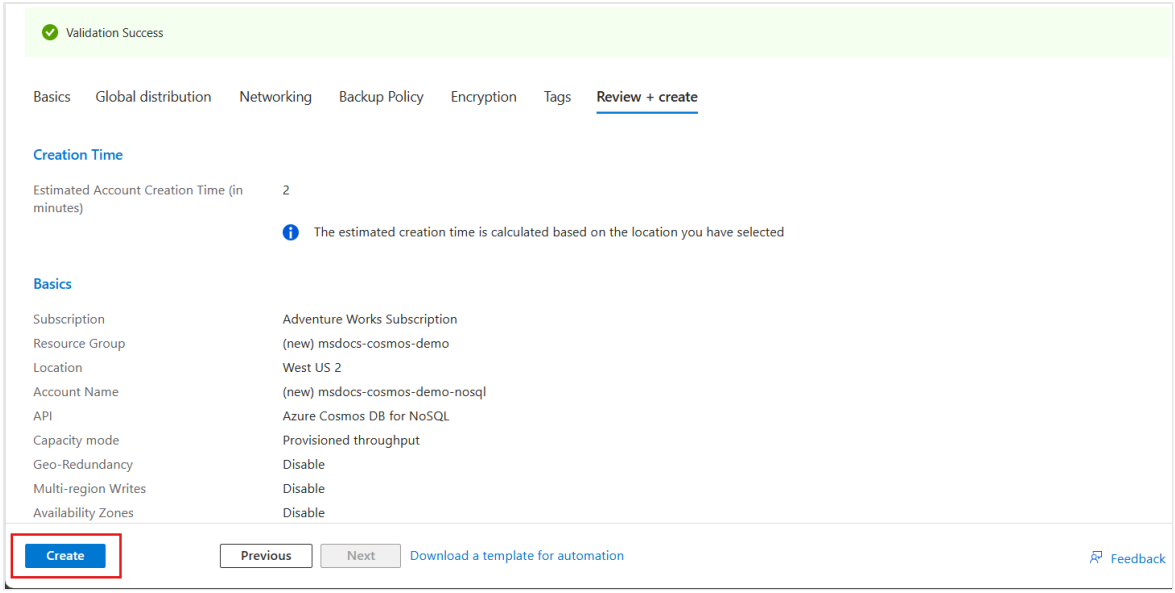
Expand table

	Value
Subscription	Select your Azure subscription
Resource Group	Create a new resource group or select an existing resource group
Account Name	Provide a globally unique name
Availability Zones	*Disable*
Location	Select a supported Azure region for your subscription

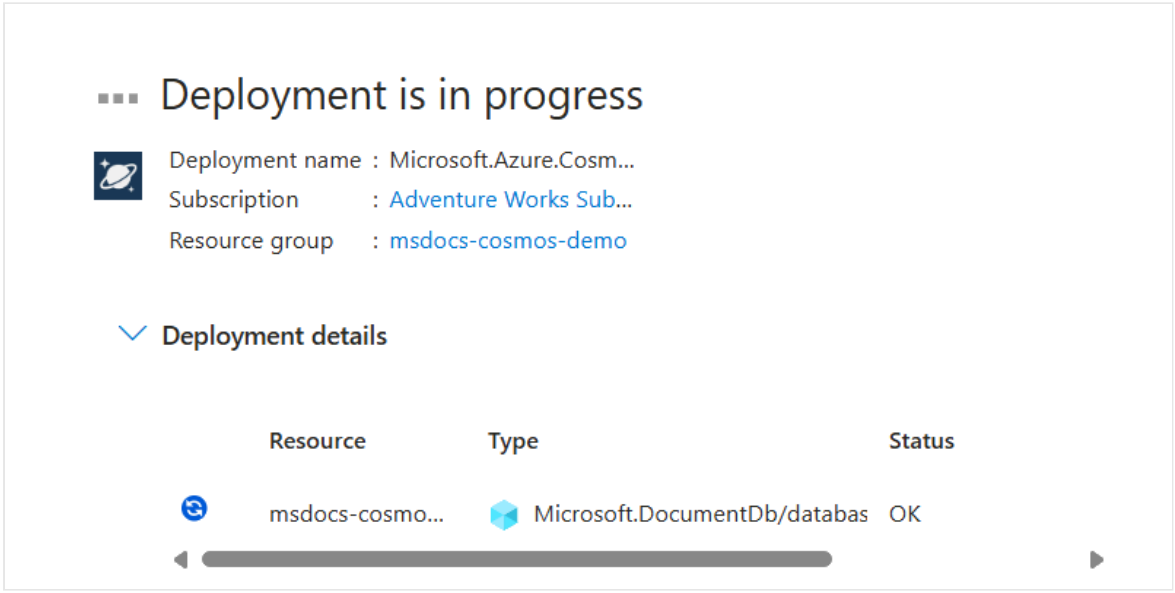


> **Tip**
>
> You can leave any unspecified options to their default values. You can also configure the account to limit total account throughput to 1,000 request units per second (RU/s) and enable free tier to minimize your costs.

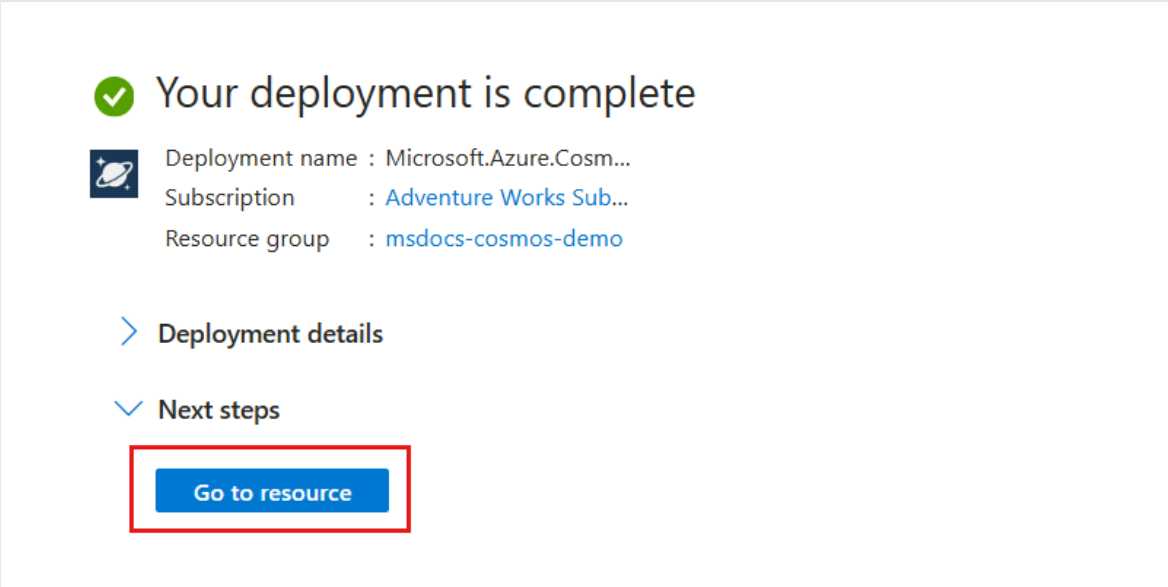
6. On the **Review + create** pane, wait for validation of your account to finish successfully, and then select **Create**.



7. The portal automatically navigates to the **Deployment** pane. Wait for the deployment to complete.



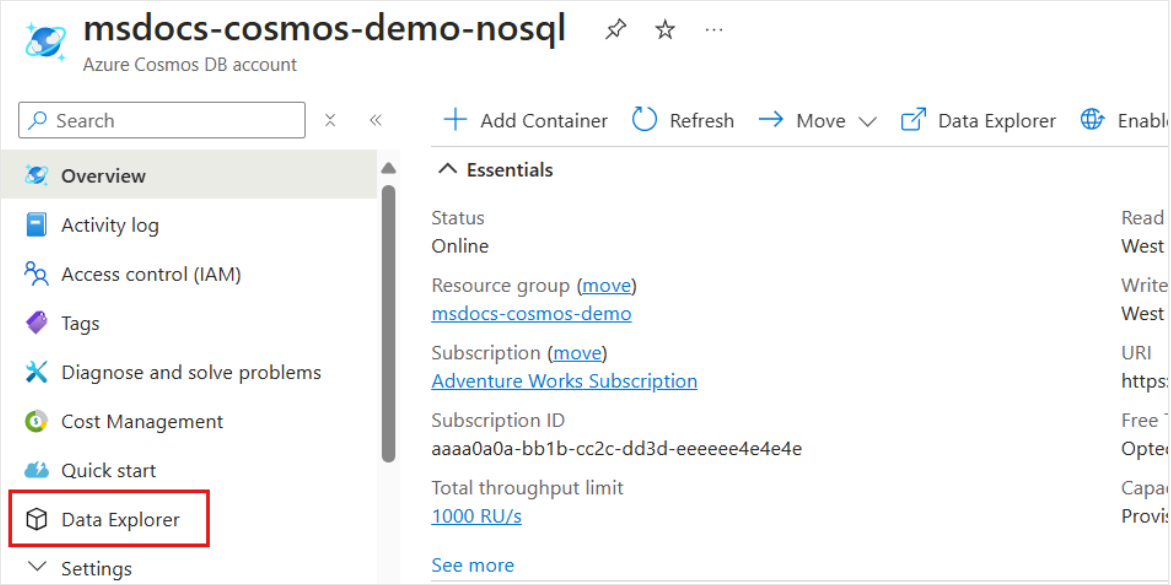
8. Once the deployment is complete, select **Go to resource** to navigate to the new Azure Cosmos DB for NoSQL account.



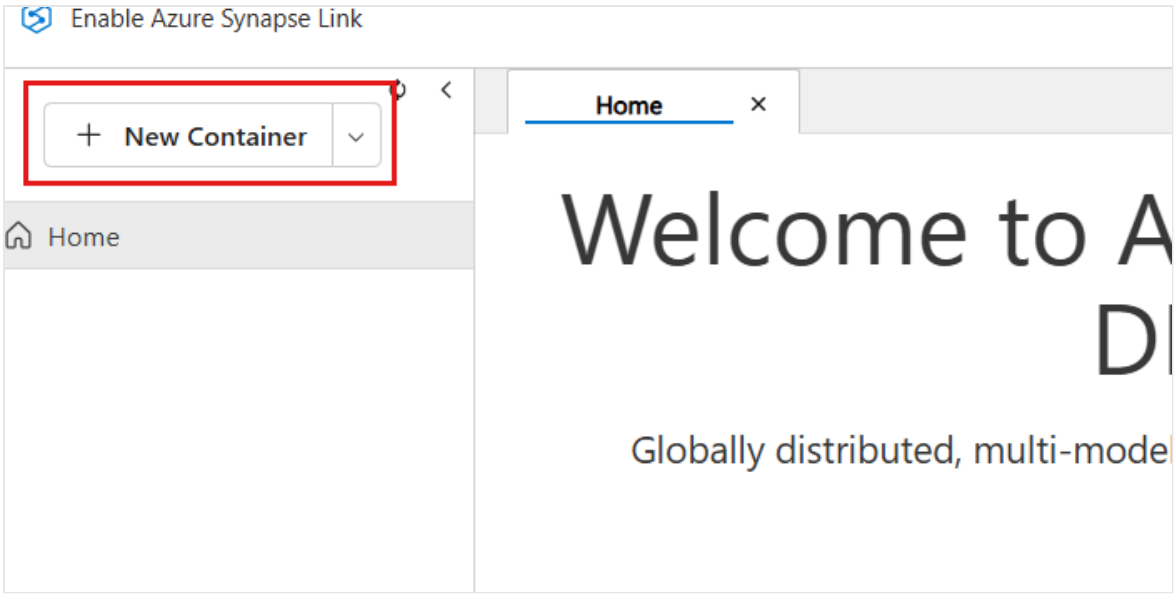
Create a database and container

Next, use the Data Explorer to create a database and container in-portal.

1. In the account resource pane, select **Data Explorer** in the service menu.



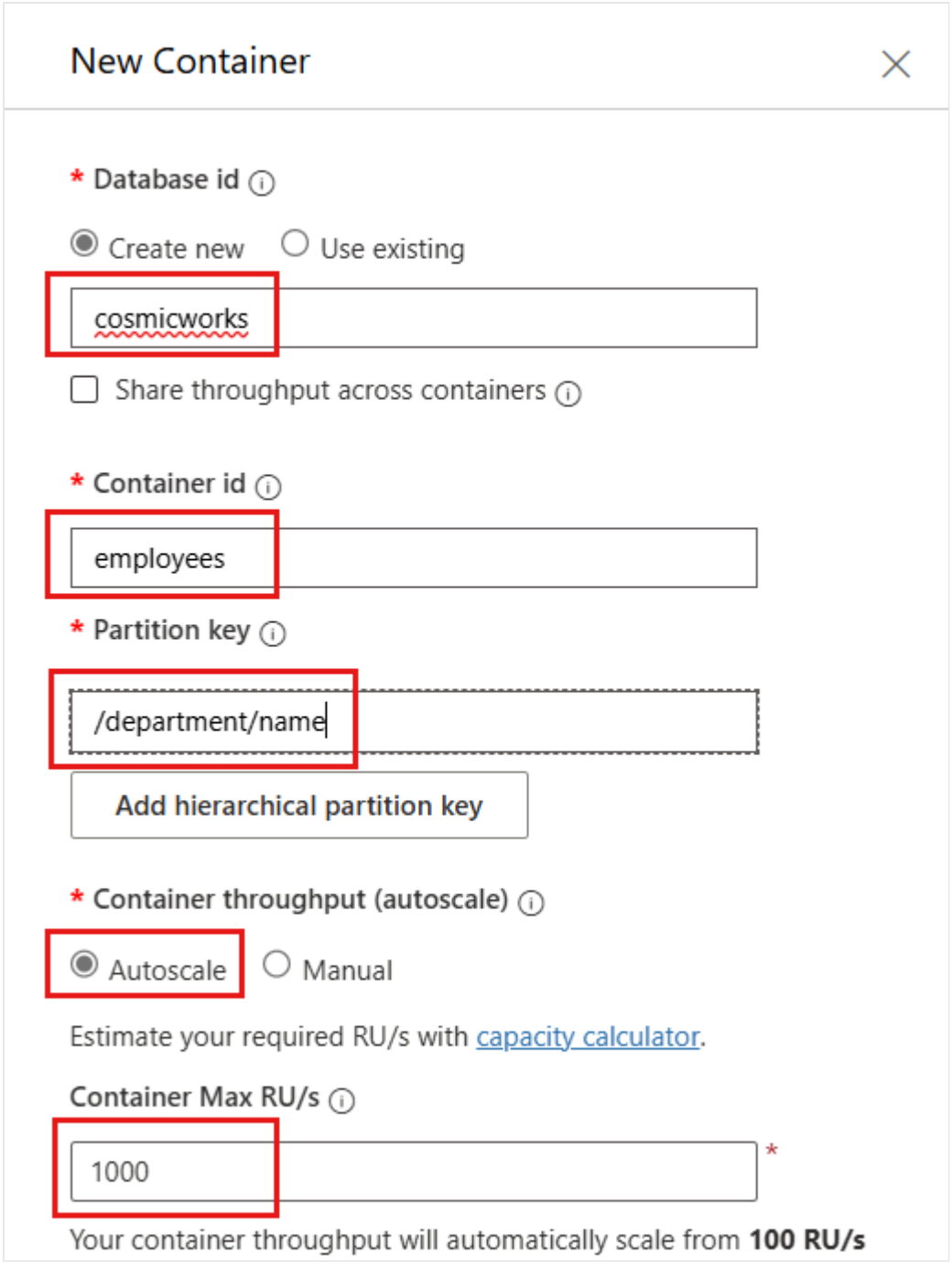
2. In the **Data Explorer** pane, select the **New Container** option.



3. In the **New Container** dialog, configure the following values and then select **OK**:

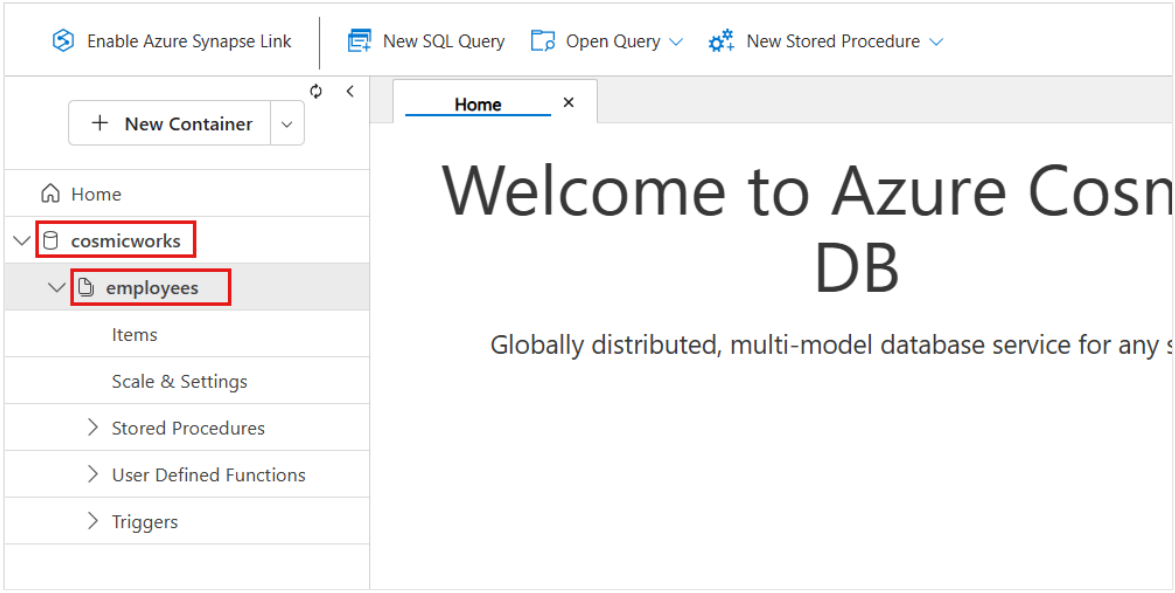
Expand table

	Value
Database	*Create new*
Database id	`cosmicworks`
Share throughput across containers	Don't select
Container id	`employees`
Partition key	`department/name`
Container throughput (autoscale)	*Autoscale*
Container Max RU/s	`1000`



4. Create a new file named *demo.bicepparam* or (`demo.bicepparam`).

5. Observe the newly created database and container in the Data Explorer's hierarchy.

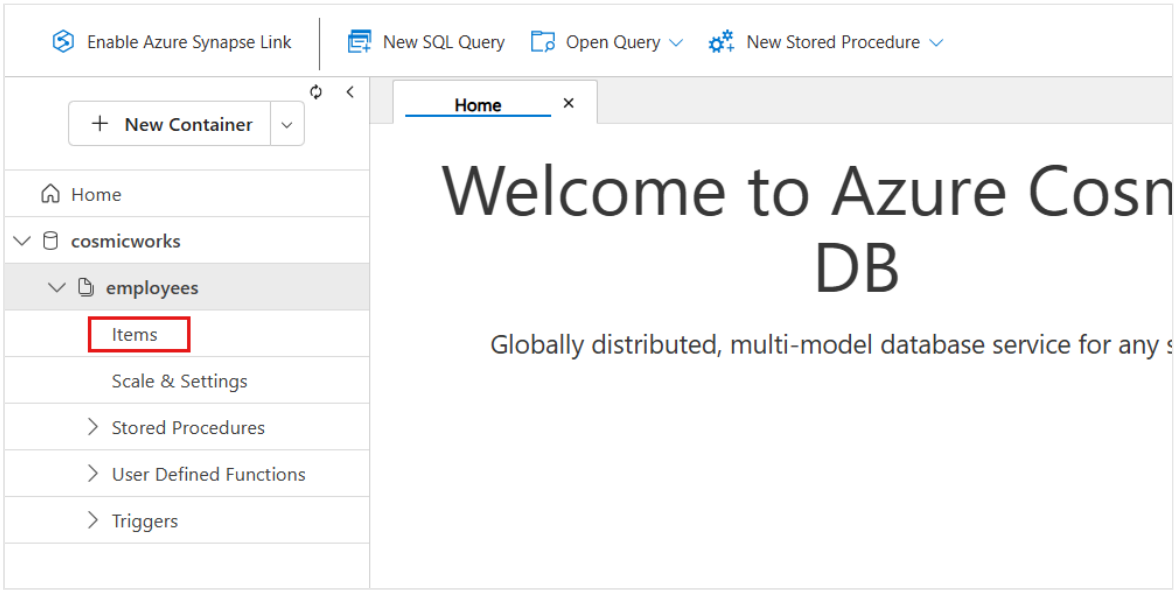


> **Tip**
>
> Optionally, you can expand the container node to observe additional properties and configuration settings.

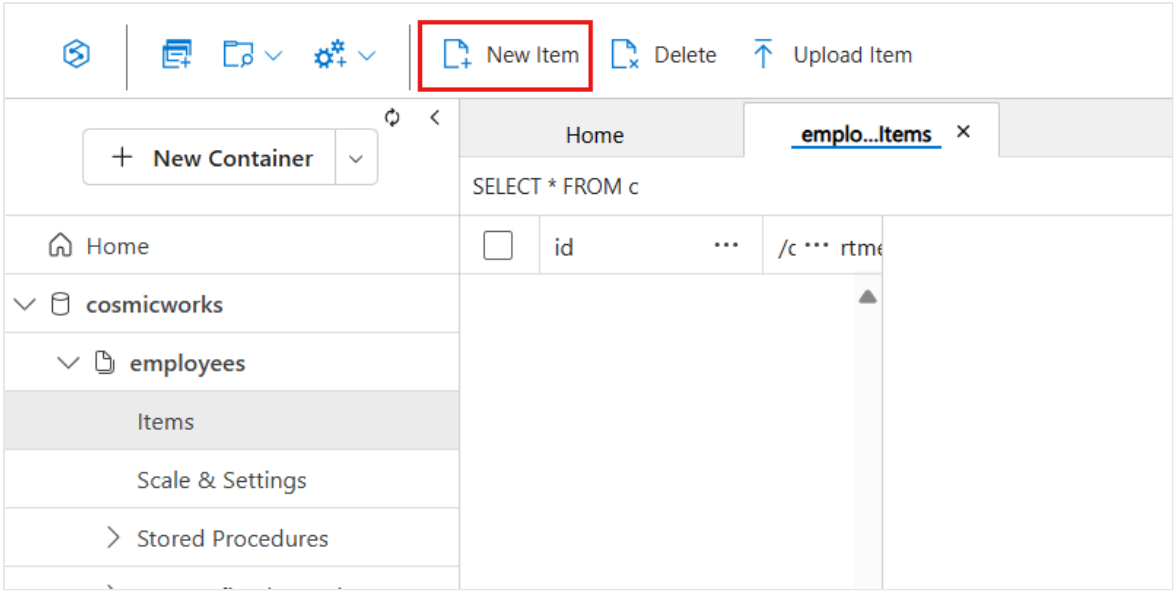
Add and query sample data

Finally, use the Data Explorer to create a sample item and then issue a basic query to the container.

1. Expand the node for the **employees** container in the tree of the Data Explorer. Then, select the **Items** option.



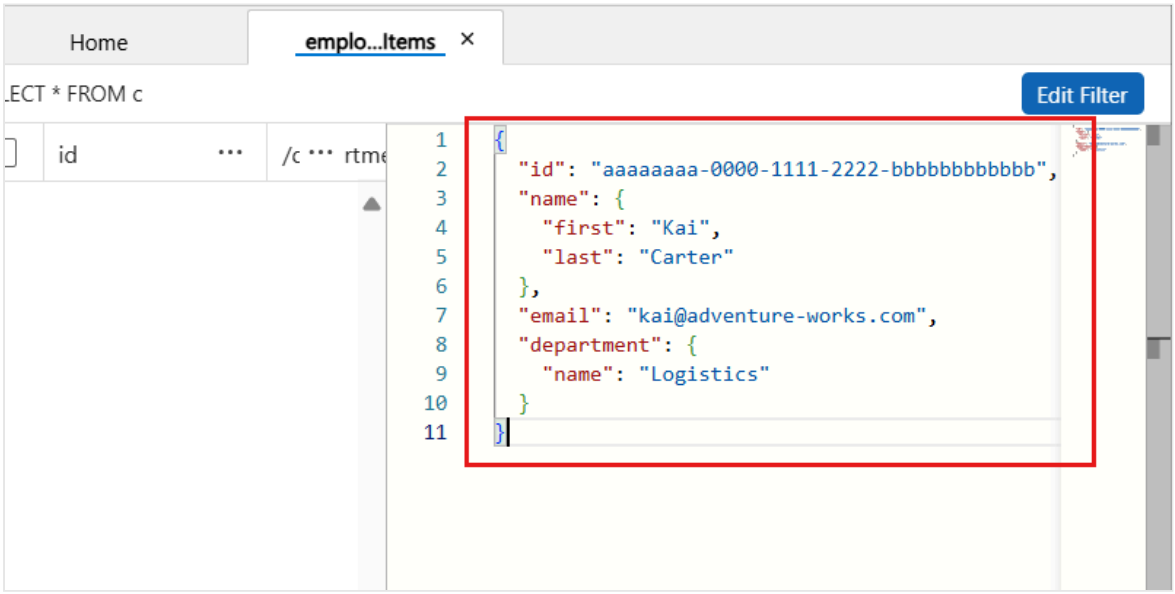
2. In the Data Explorer's menu, select **New Item**.



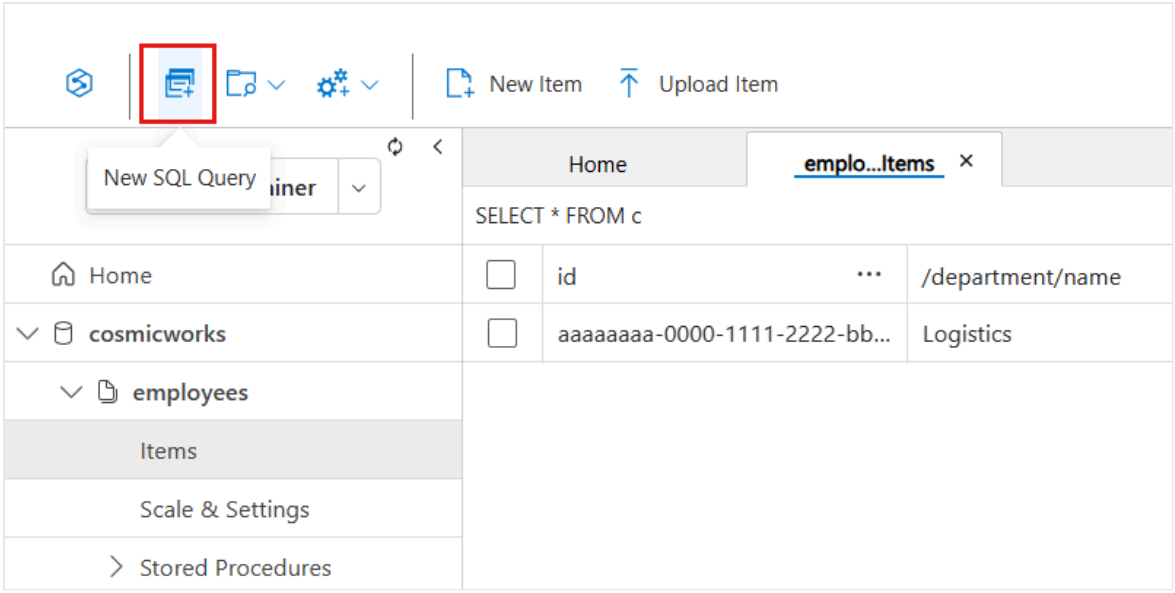
3. Now, insert the following JSON for a new item in the **employees** container and then select **Save**:

JSON

```
{
  "id": "aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb",
  "name": {
    "first": "Kai",
    "last": "Carter"
  },
  "email": "<kai@adventure-works.com>",
  "department": {
    "name": "Logistics"
  }
}
```



4. In the Data Explorer's menu, select **New SQL Query**.



5. Now, insert the following NoSQL query to get all items for the `logistics` department using a case-insensitive search. The query then formats the output as a structured JSON object. Run the query by selecting **Execute Query**:

```
NoSQL

SELECT VALUE {
    "name": CONCAT(e.name.last, " ", e.name.first),
    "department": e.department.name,
    "emailAddresses": [
        e.email
    ]
}
FROM
    employees e
WHERE
    STRINGEQUALS(e.department.name, "logistics", true)
```

6. Observe the JSON array output from the query.

JSON

```
[
  {
    "name": "Carter Kai",
    "department": "Logistics",
    "emailAddresses": [
      "kai@adventure-works.com"
    ]
  }
]
```