b'

Microsoft Azure \xe2\x80\x93 Working with PowerShell in Cosmos DB

- Last Updated :\n26 Jan, 2021

Azure Cosmos DB is a fully managed NoSQL database for build applications designed by Microsoft. It is highly responsive, scalable, and fully automated. **Azure Cloud Shell** is an in-browser terminal used to manage cloud instances in Azure. The **PowerShell** is an application used for the same purpose but is installed locally.

In this article, we will look into Azure Cosmos DB with PowerShell where we will create a new Cosmos DB account, a database, and a container with PowerShell. Here we\xe2\x80\x99ll use the Azure Cloud Shell, and you can also use a local installation of **PowerShell**.Some of the operations that you can perform using PowerShell in Cosmos DB are listed below:

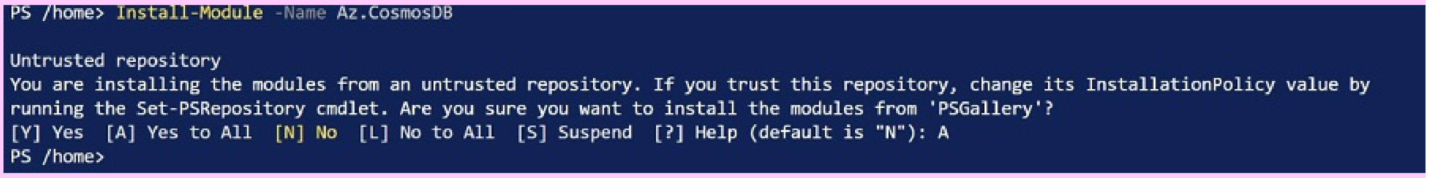
- Create a Cosmos DB database
- Update Cosmos DB database
- Delete Cosmos DB database

To create a Cosmos DB database with Powershell/ Cloud Shell follow the below steps:

- **Step 1:** Before we can start, we need to install the Cosmos DB PowerShell module like with the below command:

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

At the end of the installation you need to verify again if to install the same as shown below:



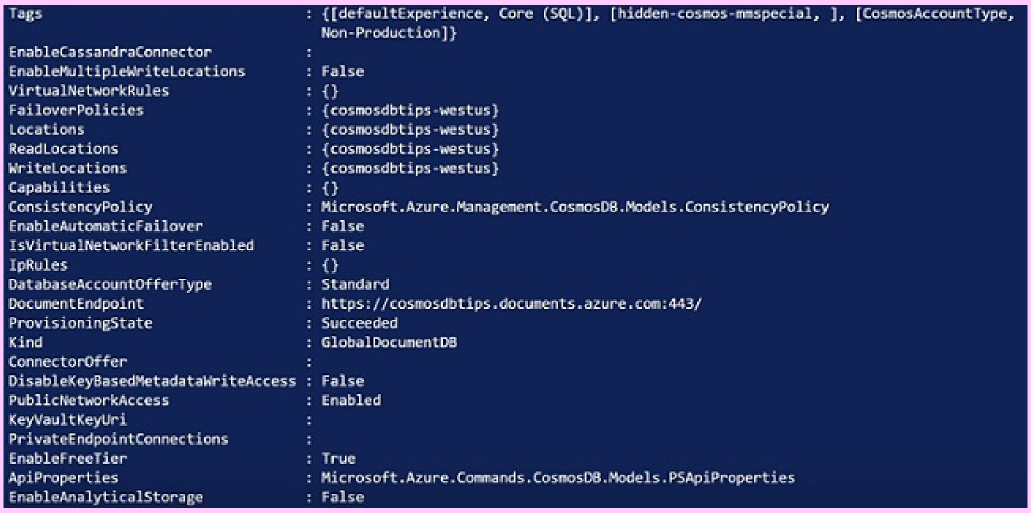
```
PS /home> Install-Module -Name Az.CosmosDB

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its InstallationPolicy value by
running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
PS /home>
```

- **Step 2:** First, to see if it works, we will run the below command to list all Cosmos DB accounts in this resource group:

```
Get-AzCosmosDBAccount -ResourceGroupName "RESOURCE NAME"
```

This will list the accounts as shown below:



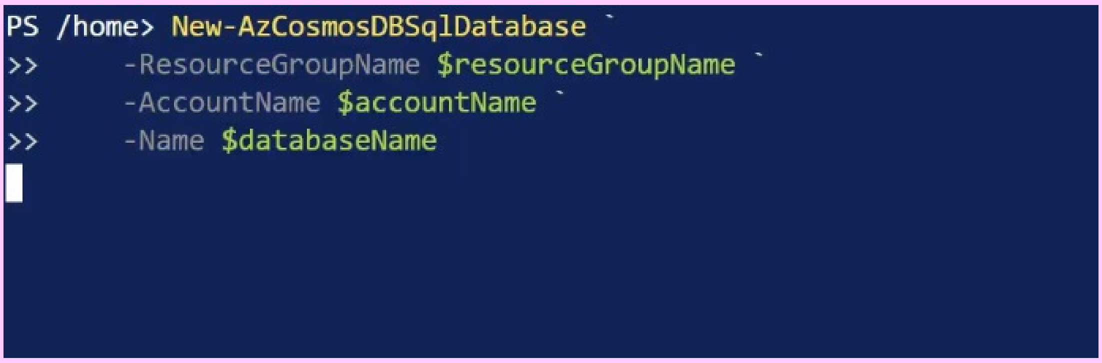
```
Tags                               : {[defaultExperience, Core (SQL)], [hidden-cosmos-mmspecial, ], [CosmosAccountType,
                                     Non-Production]}
EnableCassandraConnector           :
EnableMultipleWriteLocations       : False
VirtualNetworkRules                : {}
FailoverPolicies                   : {cosmosdbtips-westus}
Locations                          : {cosmosdbtips-westus}
ReadLocations                      : {cosmosdbtips-westus}
WriteLocations                     : {cosmosdbtips-westus}
Capabilities                       : {}
ConsistencyPolicy                  : Microsoft.Azure.Management.CosmosDB.Models.ConsistencyPolicy
EnableAutomaticFailover            : False
IsVirtualNetworkFilterEnabled      : False
IpRules                            : {}
DatabaseAccountOfferType           : Standard
DocumentEndpoint                   : https://cosmosdbtips.documents.azure.com:443/
ProvisioningState                  : Succeeded
Kind                               : GlobalDocumentDB
ConnectorOffer                     :
DisableKeyBasedMetadataWriteAccess : False
PublicNetworkAccess                : Enabled
KeyVaultKeyUri                     :
PrivateEndpointConnections         :
EnableFreeTier                     : True
ApiProperties                      : Microsoft.Azure.Commands.CosmosDB.Models.PSApiProperties
EnableAnalyticalStorage            : False
```

And we can see that there is one in this resource group.

- **Step 3:** Now, let\xe2\x80\x99s create a new Azure Cosmos DB account. This will contain a database that will contain containers with documents in it. This will be an account that uses the **SQL API** to work with data. This can take a while. At this stage, the Cosmos DB account is created.
- **Step 4:** Now we need a database for the account. To do so use the below command:

```
New-AzCosmosDBSqlDatabase
```

It would result in something like below after completion:



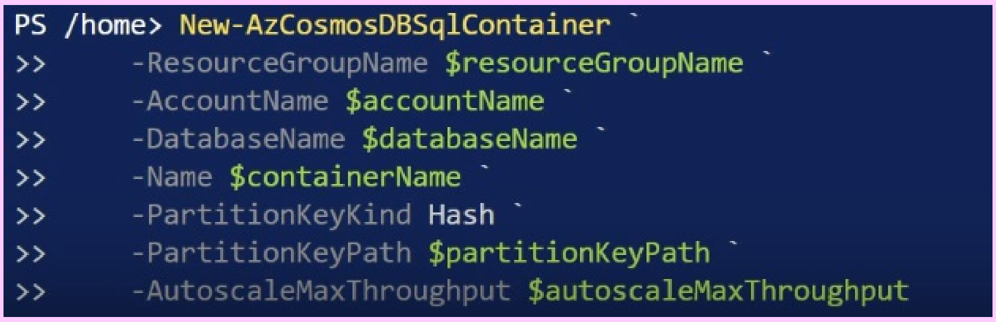
```
PS /home> New-AzCosmosDBSqlDatabase `
>>     -ResourceGroupName $resourceGroupName `
>>     -AccountName $accountName `
>>     -Name $databaseName
```

This creates a database, and it is done.\xc2\xa0

- **Step 5:** Now, we can add a container to the database with the below command:

```
New-AzCosmosDBSqlContainer
```

This will result in the following:

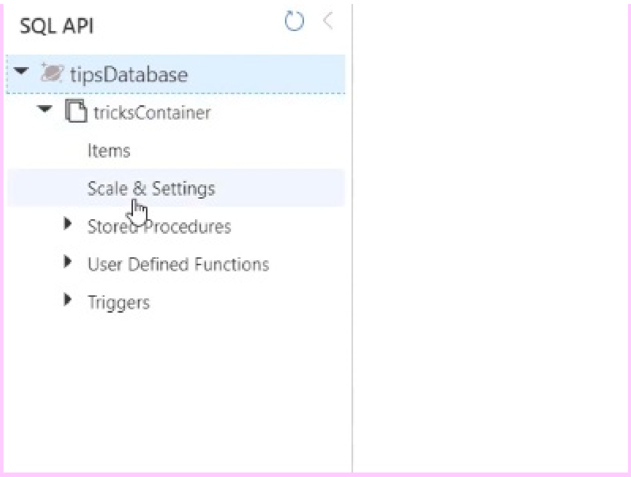


```
PS /home> New-AzCosmosDBSqlContainer `
>>     -ResourceGroupName $resourceGroupName `
>>     -AccountName $accountName `
>>     -DatabaseName $databaseName `
>>     -Name $containerName `
>>     -PartitionKeyKind Hash `
>>     -PartitionKeyPath $partitionKeyPath `
>>     -AutoscaleMaxThroughput $autoscaleMaxThroughput
```

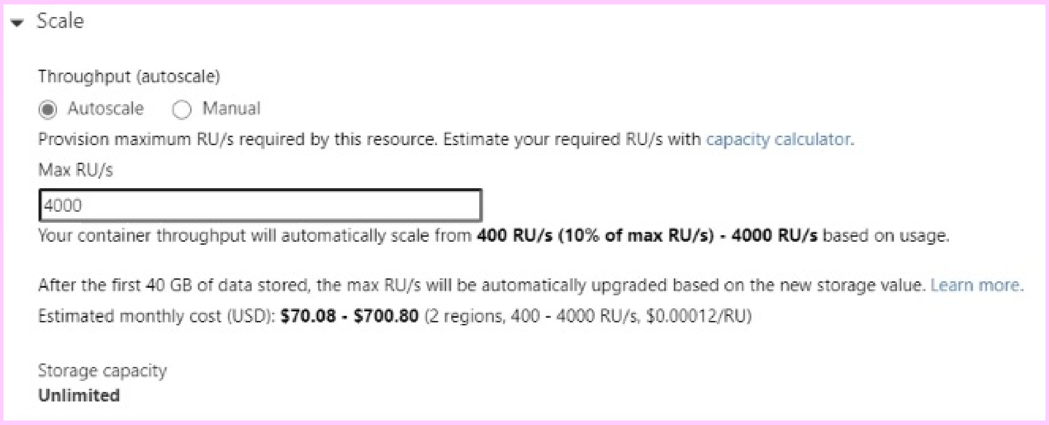
\xc2\xa0You can create multiple containers in a database, and this one would use the **Autoscale feature** as shown in the above image:

```
$ autoscaleMaxThroughput = 4000 #minimum = 4000
```

Now let\xe2\x80\x99s go to the Azure portal to look at the result. We are in the Azure Cosmos DB account in the Data Explorer. Here is the database, and under that is the container.



It also has autoscale enabled.\xc2\xa0



Scale

Throughput (autoscale)

[✓ Autoscale] Manual

Provision maximum RU/s required by this resource. Estimate your required RU/s with capacity calculator.

Max RU/s

4000

Your container throughput will automatically scale from **400 RU/s (10% of max RU/s) - 4000 RU/s** based on usage.

After the first 40 GB of data stored, the max RU/s will be automatically upgraded based on the new storage value. Learn more.

Estimated monthly cost (USD): **$70.08 - $700.80** (2 regions, 400 - 4000 RU/s, $0.00012/RU)

Storage capacity
Unlimited

We\xe2\x80\x99ve used PowerShell to list Azure Cosmos DB accounts and create a new one with a database and autoscale container.

My Personal Notes\n*arrow_drop_up*

Add your personal notes her

Save

'