**CosmoDB : Lab Manual handbook [I]**

**Prepared for**: TechPledge

**Date:……….**, 2020

**Prepared by:** Shruti Sinhaa

Document Name: Lab Manual

**Document Number** AZ204\_04

**Contributor:**

Bipeen Sinhaa

Contents

[\* Introduction: 3](#_Toc56022387)

[Reference GitHub repository: 3](#_Toc56022388)

[Clone the Labs repository to local device: 3](#_Toc56022389)

[Create Container from VsCode 5](#_Toc56022390)

[Update Program.cs 6](#_Toc56022391)

[Update primary key and endpoint uri in following script 6](#_Toc56022392)

[Create Database : 8](#_Toc56022393)

[Populate Container with Data of Different Types 13](#_Toc56022394)

# 

# \* Introduction:

In this Lab we would be performing following tasks:

* Create COSMODB account with PowerShell.
* Create database container and put data using PowerShell.

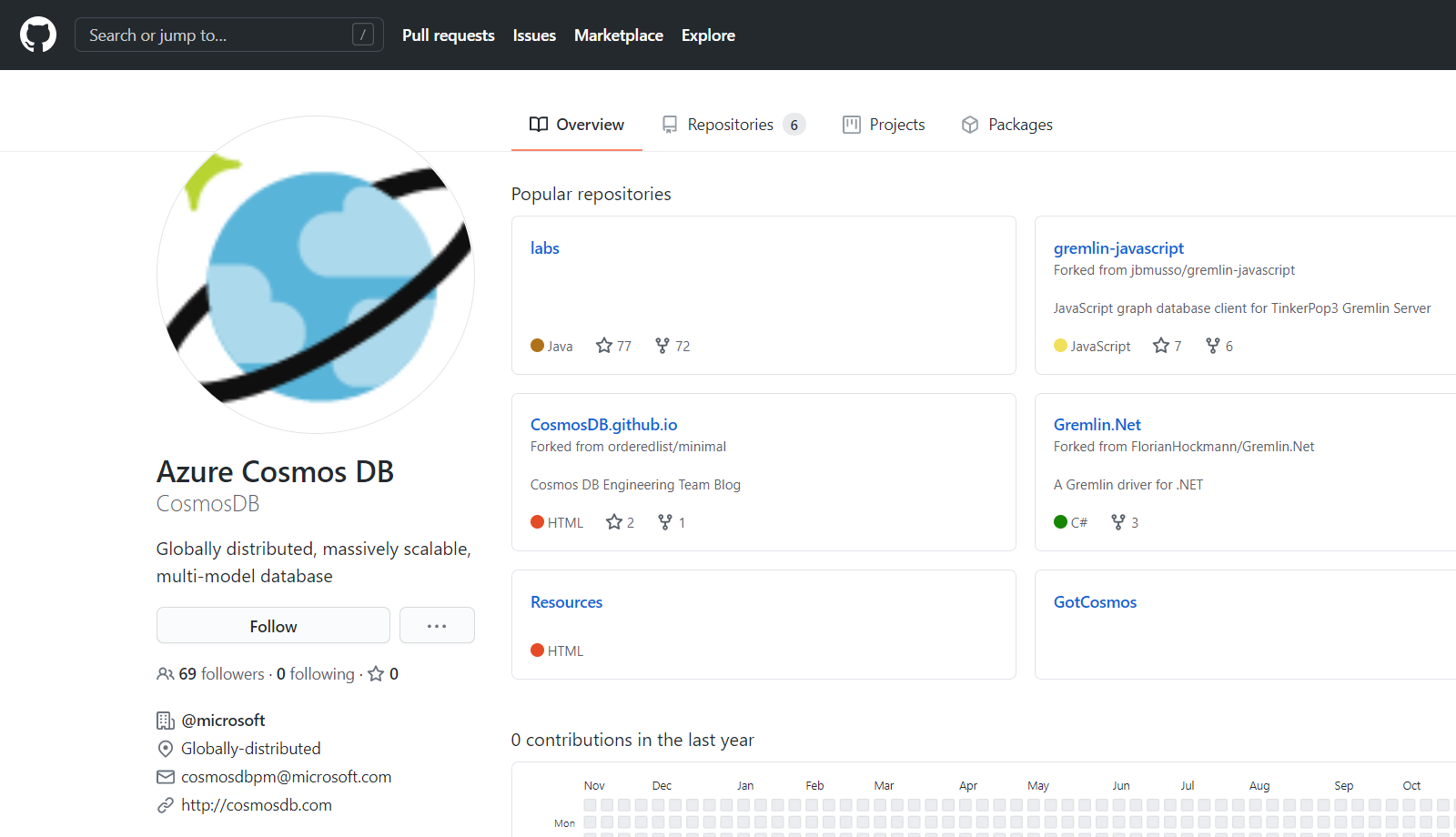
# Reference GitHub repository:

<https://github.com/CosmosDB/labs>

\* <https://cosmosdb.github.io/labs/dotnet/labs/01-creating_partitioned_collection.html>

\* <https://cosmosdb.github.io/labs/dotnet/labs/00-account_setup.html#lab-content-setup>

This repository is from Microsoft which stores PS for managing cosmoDB



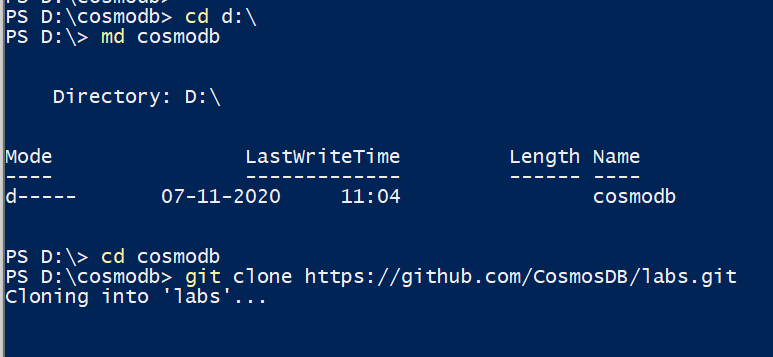
# Clone the Labs repository to local device:

Open Windows PowerShell in admin mode

md cosmodb

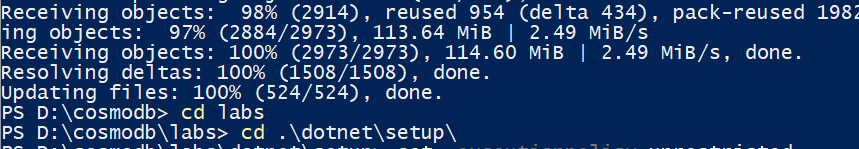
cd cosmodb

git clone <https://github.com/CosmosDB/labs.git>

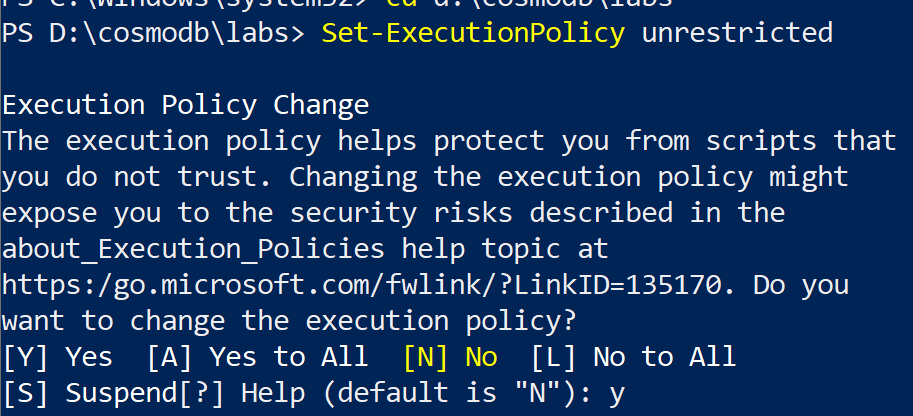


cd labs

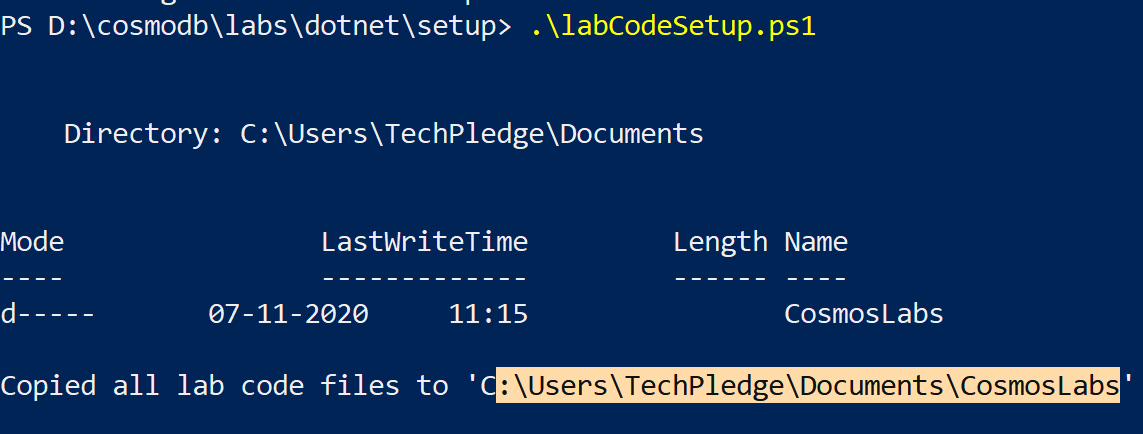
cd .\dotnet\setup\



Set-ExecutionPolicy Unrestricted

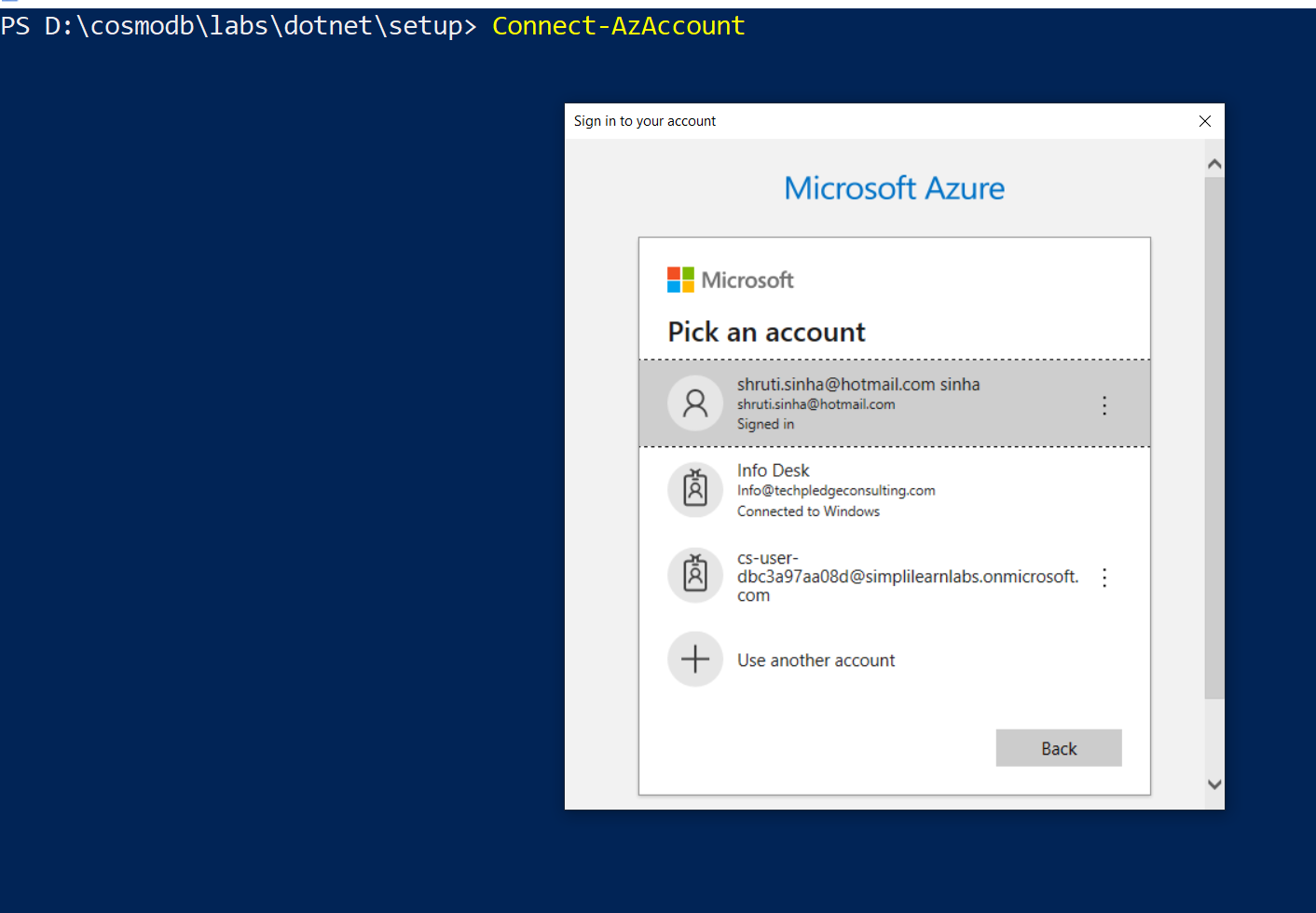


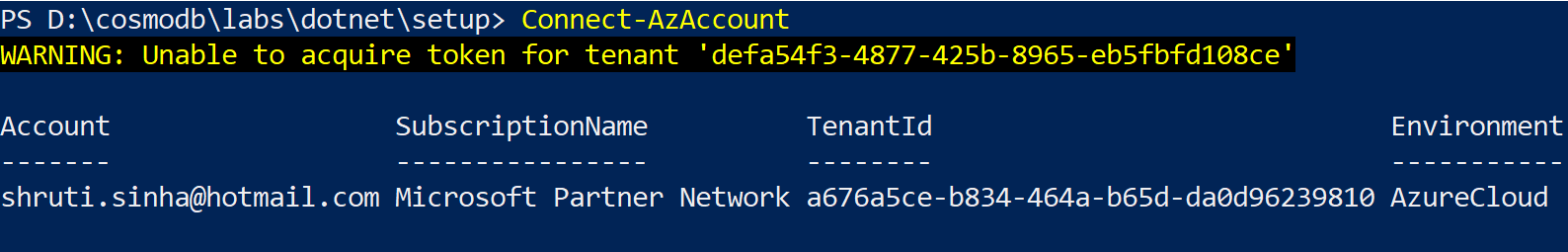
.\labCodeSetup.ps1



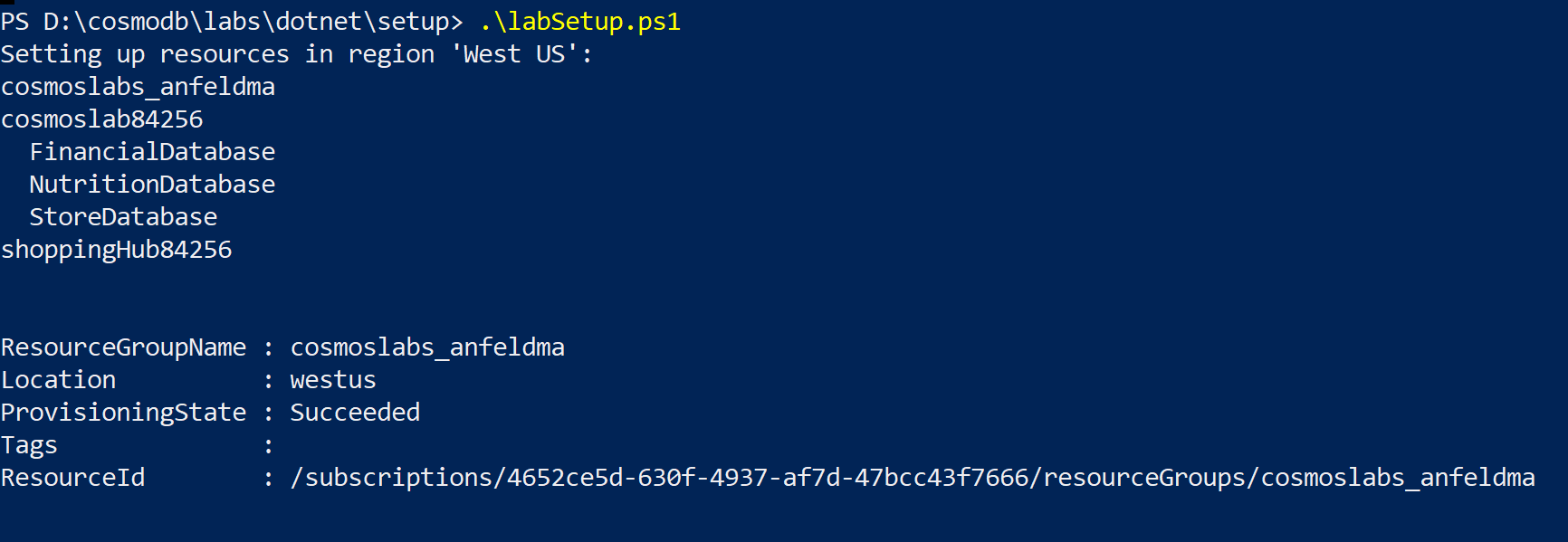
Install-Module -Name Az -AllowClobber

Connect-AzAccount





.\labSetup.ps1

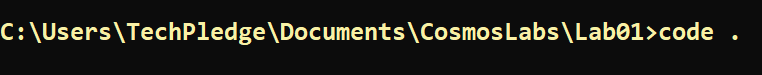


We have created a Cosmo DB account using the PS.

Now lets create container inside:

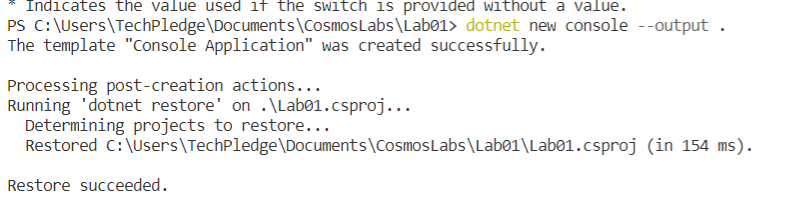
# Create Container from VsCode

open "DataTypes.cs" in VsCode from C:\Users\Admin\Documents\CosmosLabs\Lab01

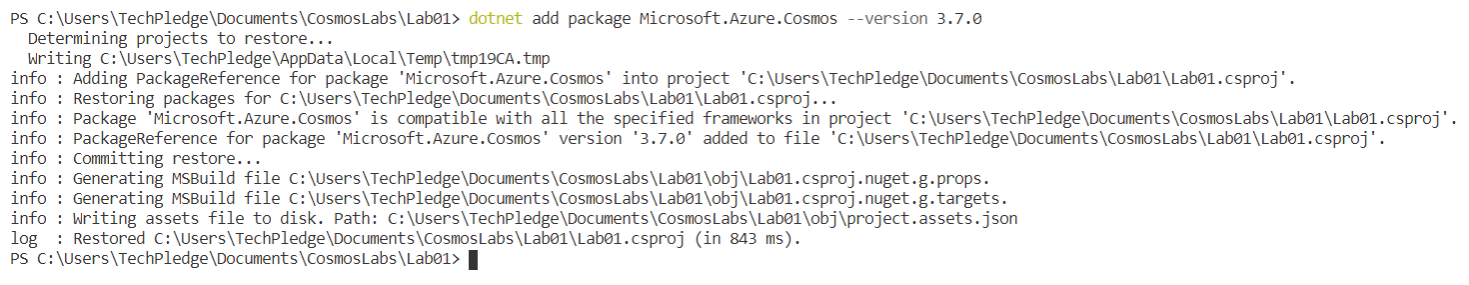


in vscode terminal type

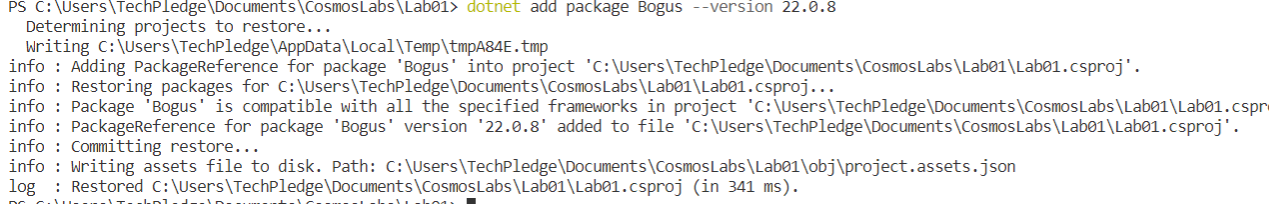
dotnet new console --output .



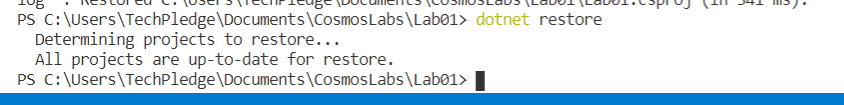
dotnet add package Microsoft.Azure.Cosmos --version 3.7.0



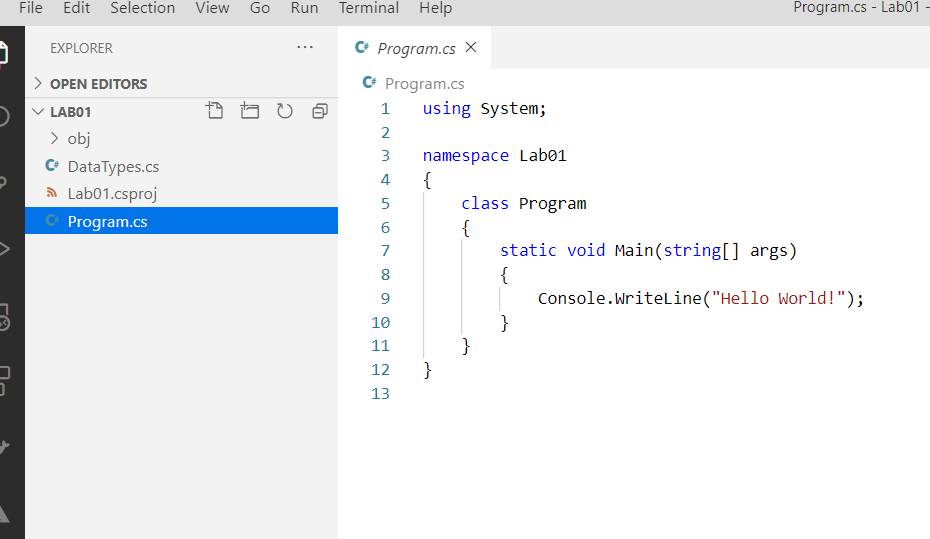
dotnet add package Bogus --version 22.0.8

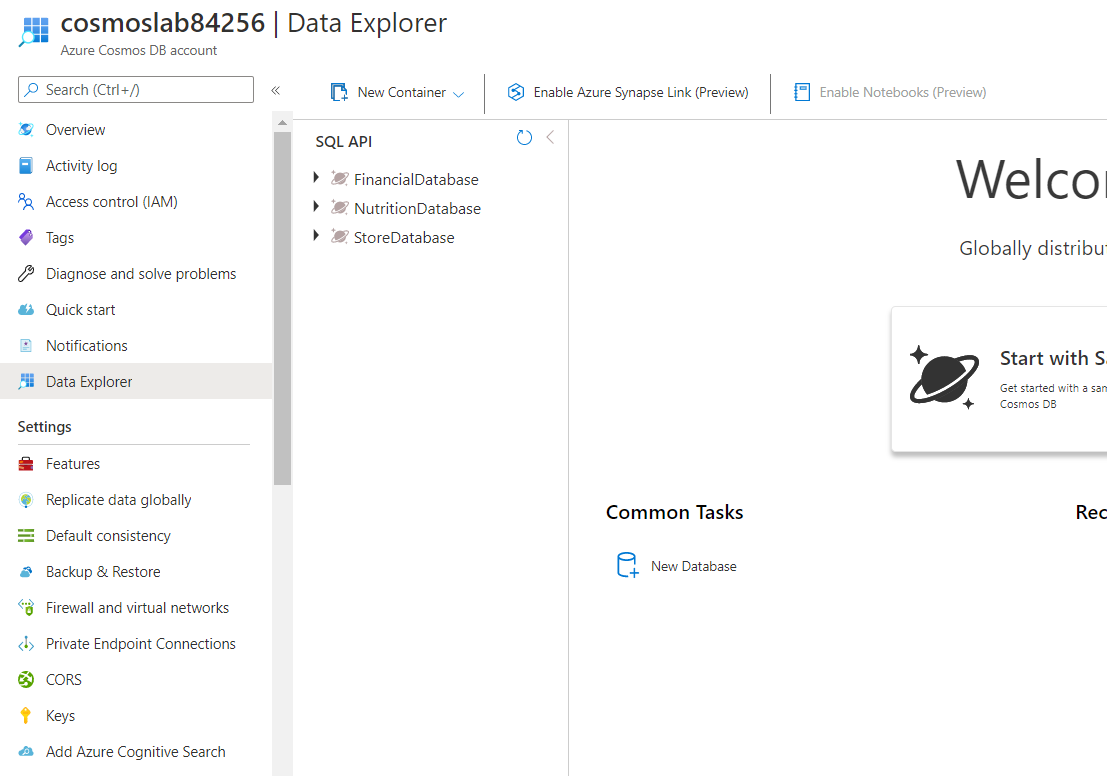


dotnet restore

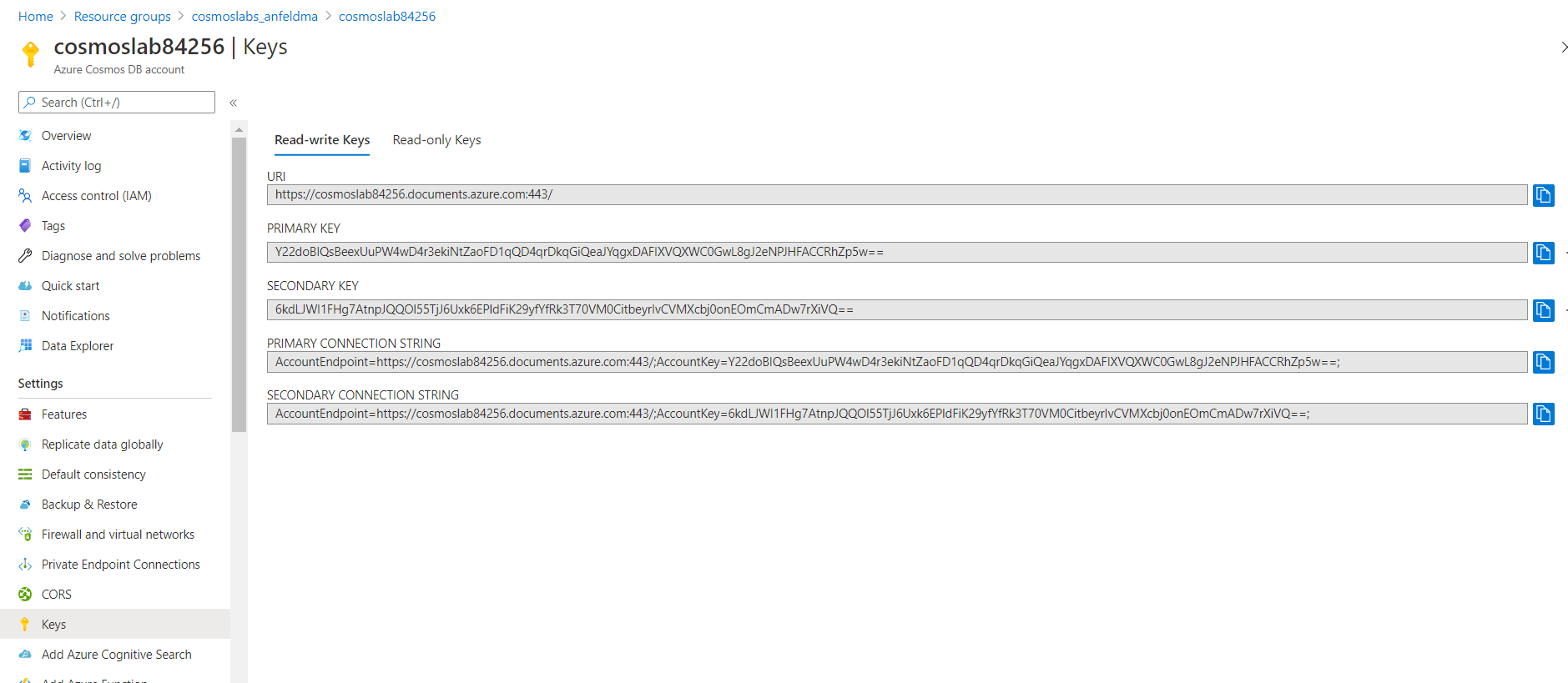


## Update Program.cs





## Update primary key and endpoint uri in following script



==========Program.cs=============

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Azure.Cosmos;

using System;

namespace Lab01

{

public class Program

{

private static readonly string \_endpointUri = "https://cosmoslab22681.documents.azure.com:443/";

private static readonly string \_primaryKey = "09xCU4jPjbh58KFpdkeMUefkopgUZxyMuLyq2p5rhZJ6s288TX9yJnFi07nghxNzCXT8wsRw68ERXYUJ9llNQg==";

public static async Task Main(string[] args)

{

using (CosmosClient client = new CosmosClient(\_endpointUri, \_primaryKey))

{

}

}

}

}



====Save the File and Run Dotnet Build ===

dotnet build



## Create Database :

=======Program.cs to Create the Database=========

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Azure.Cosmos;

using System;

namespace Lab01

{

public class Program

{

private static readonly string \_endpointUri = "https://cosmoslab22681.documents.azure.com:443/";

private static readonly string \_primaryKey = "09xCU4jPjbh58KFpdkeMUefkopgUZxyMuLyq2p5rhZJ6s288TX9yJnFi07nghxNzCXT8wsRw68ERXYUJ9llNQg==";

public static async Task Main (string[] args)

{

using (CosmosClient client = new CosmosClient(\_endpointUri, \_primaryKey))

{

DatabaseResponse databaseResponse = await client.CreateDatabaseIfNotExistsAsync("EntertainmentDatabase");

Database targetDatabase = databaseResponse.Database;

await Console.Out.WriteLineAsync($"Database Id:\t{targetDatabase.Id}");

}

}

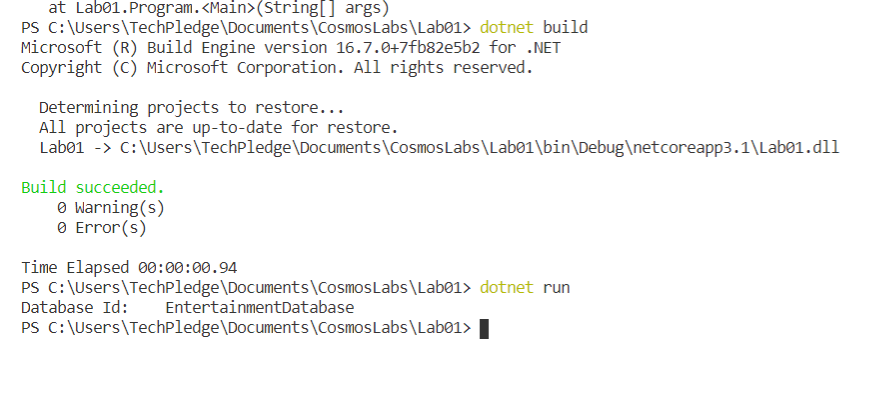
}

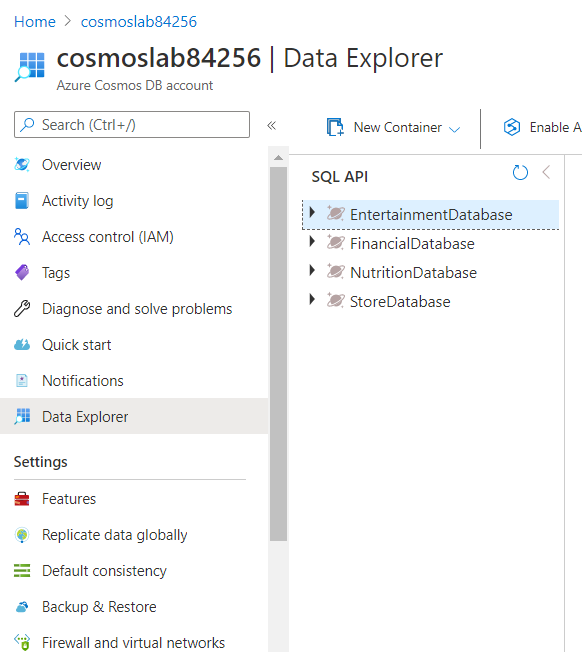
}

===========Now Build and Run=======

Dotnet Build

Dotnet Run





========Create a Partitioned Container using the SDK==

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Azure.Cosmos;

using System;

namespace Lab01

{

public class Program

{

private static readonly string \_endpointUri = "https://cosmoslab22681.documents.azure.com:443/";

private static readonly string \_primaryKey = "09xCU4jPjbh58KFpdkeMUefkopgUZxyMuLyq2p5rhZJ6s288TX9yJnFi07nghxNzCXT8wsRw68ERXYUJ9llNQg==";

public static async Task Main (string[] args)

{

using (CosmosClient client = new CosmosClient(\_endpointUri, \_primaryKey))

{

// DatabaseResponse databaseResponse = await client.CreateDatabaseIfNotExistsAsync("EntertainmentDatabase");

//Database targetDatabase = databaseResponse.Database;

// await Console.Out.WriteLineAsync($"Database Id:\t{targetDatabase.Id}");

Database targetDatabase = client.GetDatabase("EntertainmentDatabase");

IndexingPolicy indexingPolicy = new IndexingPolicy

{

IndexingMode = IndexingMode.Consistent,

Automatic = true,

IncludedPaths =

{

new IncludedPath

{

Path = "/\*"

}

}

};

var containerProperties = new ContainerProperties("CustomCollection", "/type")

{

IndexingPolicy = indexingPolicy

};

var containerResponse = await targetDatabase.CreateContainerIfNotExistsAsync(containerProperties, 10000);

var customContainer = containerResponse.Container;

await Console.Out.WriteLineAsync($"Custom Container Id:\t{customContainer.Id}");

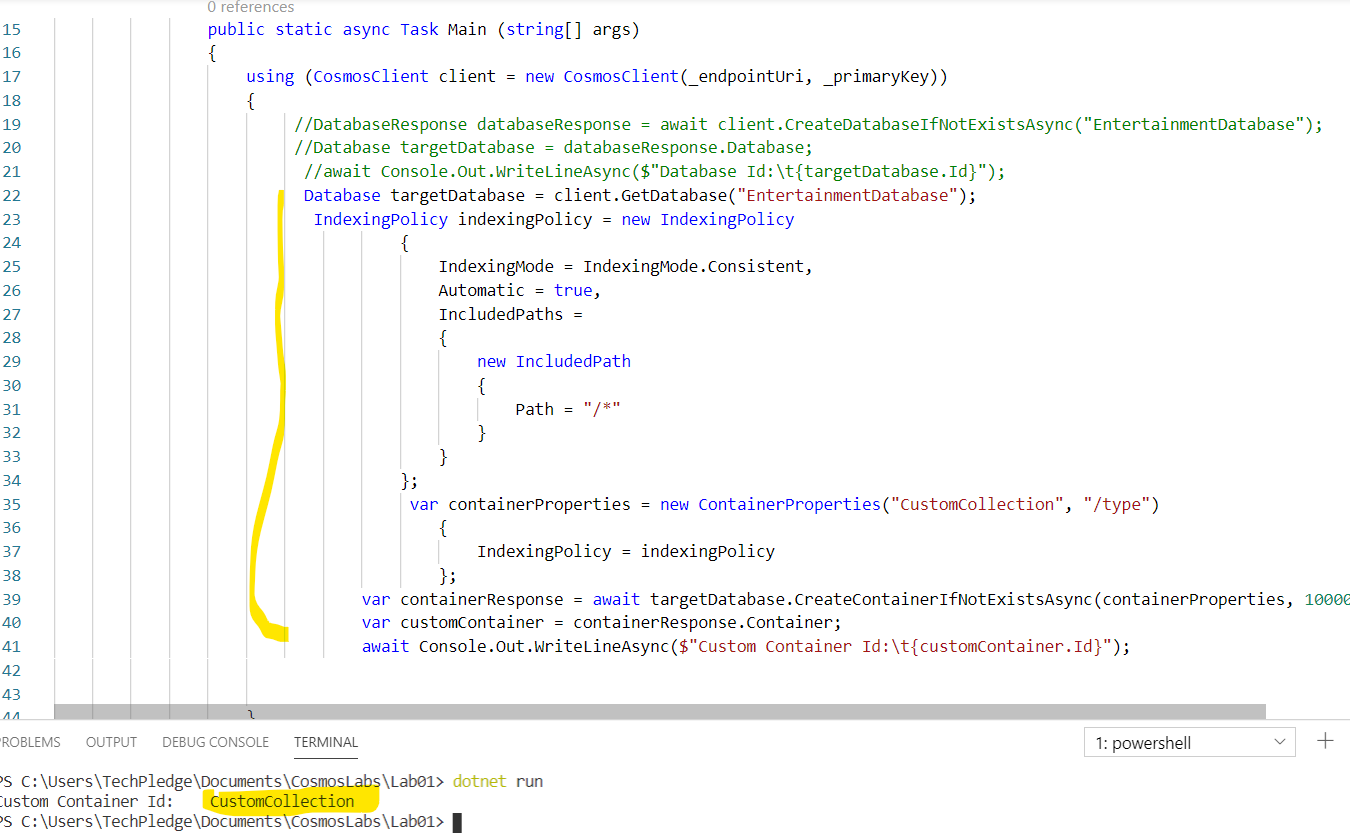
}

}

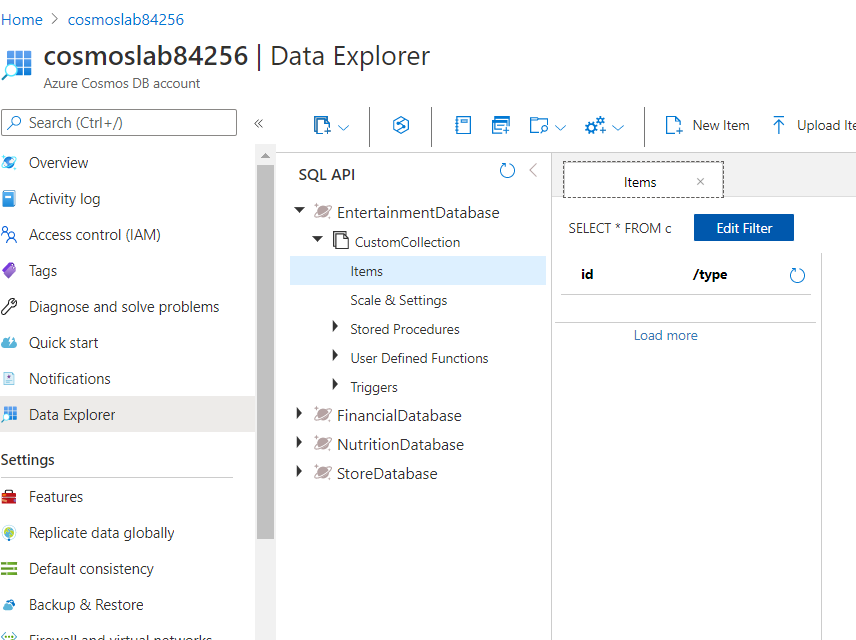
}

}

dotnet run



Observe changes on Azure Cosmodb account



# Populate Container with Data of Different Types

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Azure.Cosmos;

using System;

namespace Lab01

{

public class Program

{

private static readonly string \_endpointUri = "https://cosmoslab22681.documents.azure.com:443/";

private static readonly string \_primaryKey = "09xCU4jPjbh58KFpdkeMUefkopgUZxyMuLyq2p5rhZJ6s288TX9yJnFi07nghxNzCXT8wsRw68ERXYUJ9llNQg==";

public static async Task Main (string[] args)

{

/\* using (CosmosClient client = new CosmosClient(\_endpointUri, \_primaryKey))

{

// DatabaseResponse databaseResponse = await client.CreateDatabaseIfNotExistsAsync("EntertainmentDatabase");

//Database targetDatabase = databaseResponse.Database;

// await Console.Out.WriteLineAsync($"Database Id:\t{targetDatabase.Id}");

//Database targetDatabase = client.GetDatabase("EntertainmentDatabase");

/\* IndexingPolicy indexingPolicy = new IndexingPolicy

{

IndexingMode = IndexingMode.Consistent,

Automatic = true,

IncludedPaths =

{

new IncludedPath

{

Path = "/\*"

}

}

};

var containerProperties = new ContainerProperties("CustomCollection", "/type")

{

IndexingPolicy = indexingPolicy

};

var containerResponse = await targetDatabase.CreateContainerIfNotExistsAsync(containerProperties, 10000);

var customContainer = containerResponse.Container;

await Console.Out.WriteLineAsync($"Custom Container Id:\t{customContainer.Id}");\*/

/\* var targetDatabase = client.GetDatabase("EntertainmentDatabase");

var customContainer = targetDatabase.GetContainer("CustomCollection");

var foodInteractions = new Bogus.Faker<PurchaseFoodOrBeverage>()

.RuleFor(i => i.id, (fake) => Guid.NewGuid().ToString())

.RuleFor(i => i.type, (fake) => nameof(PurchaseFoodOrBeverage))

.RuleFor(i => i.unitPrice, (fake) => Math.Round(fake.Random.Decimal(1.99m, 15.99m), 2))

.RuleFor(i => i.quantity, (fake) => fake.Random.Number(1, 5))

.RuleFor(i => i.totalPrice, (fake, user) => Math.Round(user.unitPrice \* user.quantity, 2))

.GenerateLazy(500);

foreach(var interaction in foodInteractions)

{

ItemResponse<PurchaseFoodOrBeverage> result = await customContainer.CreateItemAsync(interaction);

await Console.Out.WriteLineAsync($"Document [PurchaseFoodOrBeverage] Created\t{result.Resource.id}");

}\*/

using (CosmosClient client = new CosmosClient(\_endpointUri, \_primaryKey))

{

var targetDatabase = client.GetDatabase("EntertainmentDatabase");

var customContainer = targetDatabase.GetContainer("CustomCollection");

var tvInteractions = new Bogus.Faker<WatchLiveTelevisionChannel>()

.RuleFor(i => i.id, (fake) => Guid.NewGuid().ToString())

.RuleFor(i => i.type, (fake) => nameof(WatchLiveTelevisionChannel))

.RuleFor(i => i.minutesViewed, (fake) => fake.Random.Number(1, 45))

.RuleFor(i => i.channelName, (fake) => fake.PickRandom(new List<string> { "NEWS-6", "DRAMA-15", "ACTION-12", "DOCUMENTARY-4", "SPORTS-8" }))

.GenerateLazy(500);

foreach(var interaction in tvInteractions)

{

ItemResponse<WatchLiveTelevisionChannel> result = await customContainer.CreateItemAsync(interaction);

await Console.Out.WriteLineAsync($"Document [WatchLiveTelevisionChannel] Created\t{result.Resource.id}");

}

}

}

}

}



dotnet run

