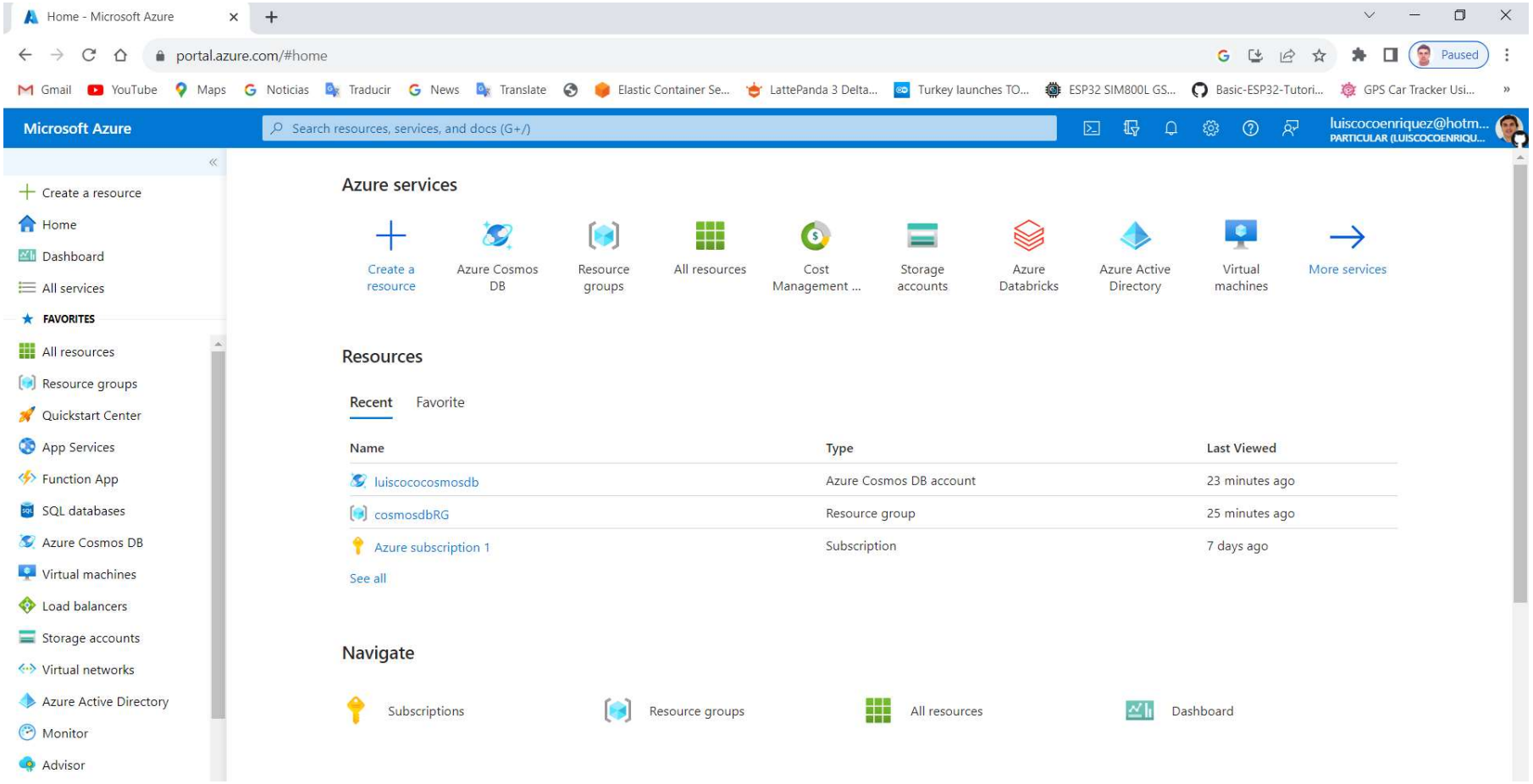
Cosmos DB

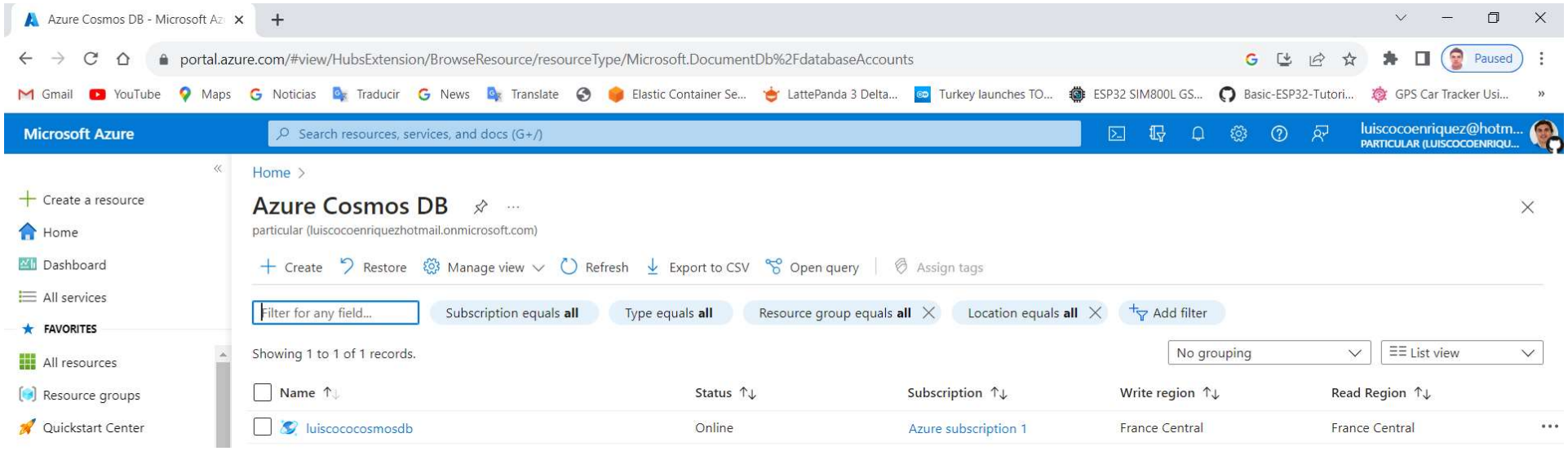
console application in C#

(quickStart)

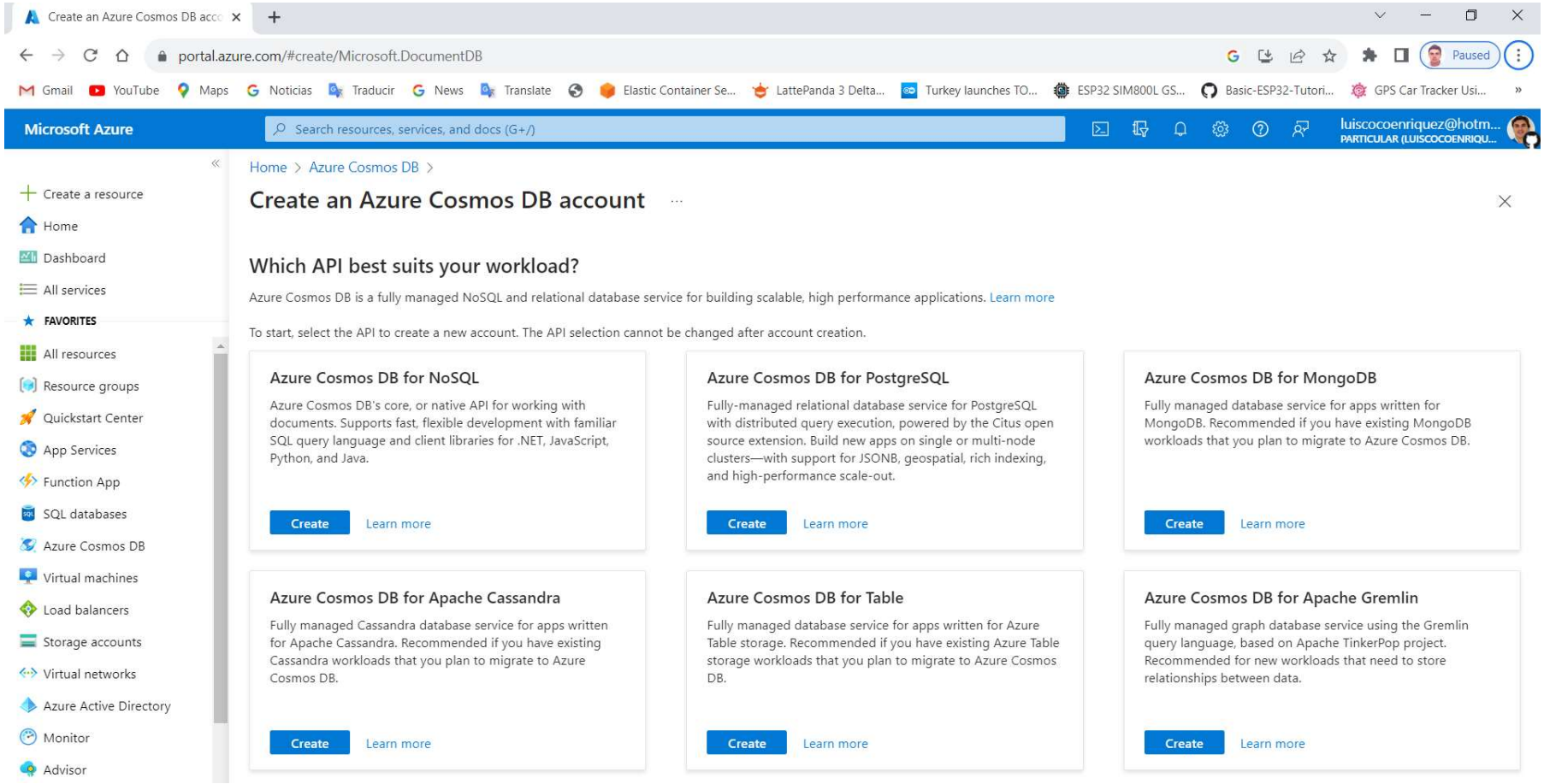
1. Login in the Azure portal and select **Azure CosmosDB**



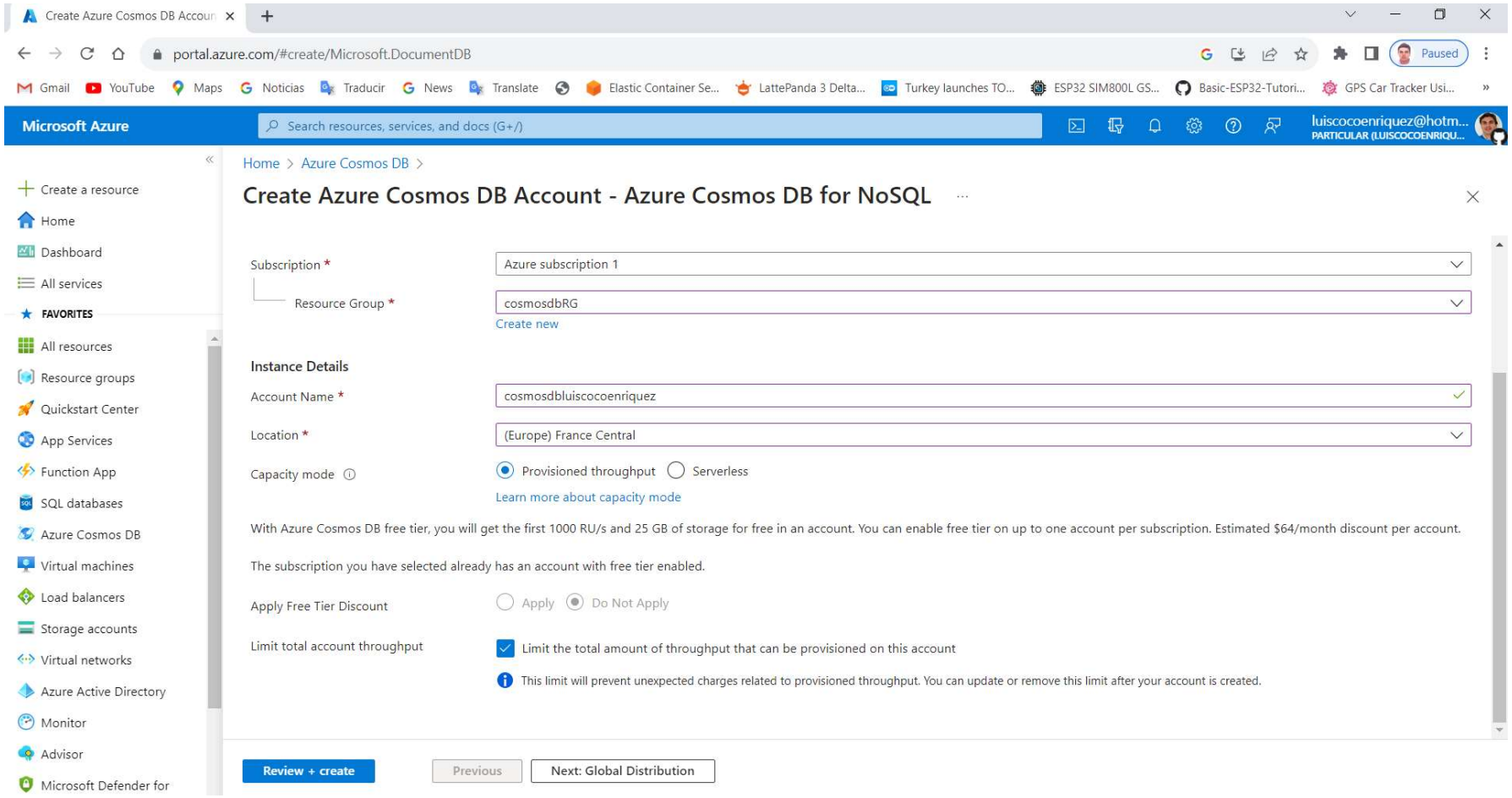
2. Press the “Create” button in the **Azure CosmosDB**



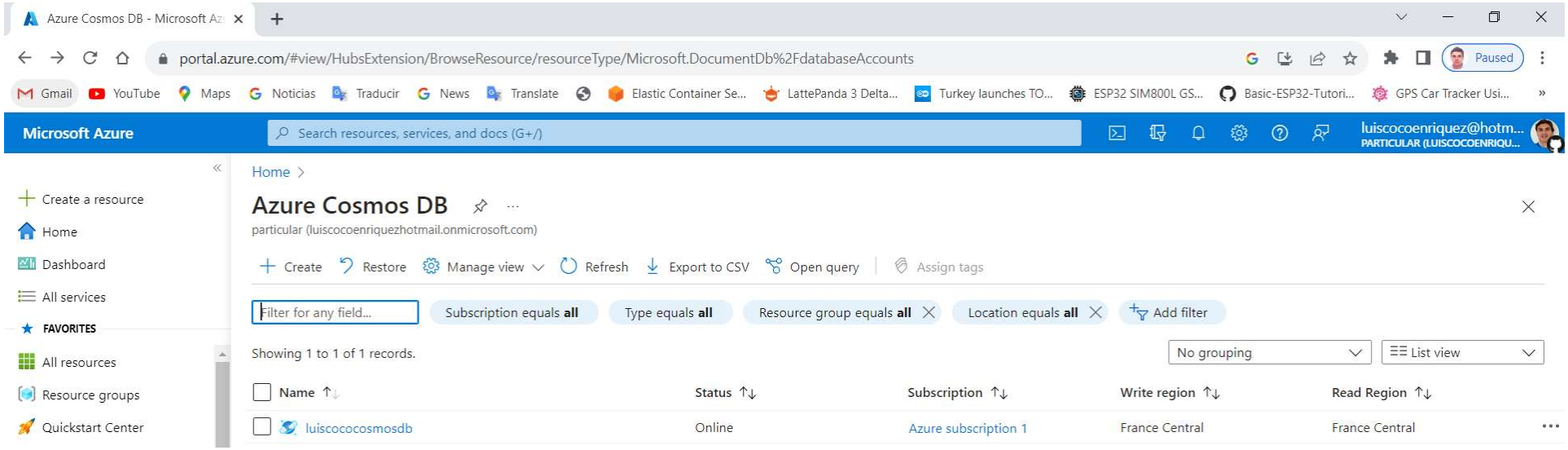
3. Press the “Create” button in the **Azure CosmosDB for NoSQL**



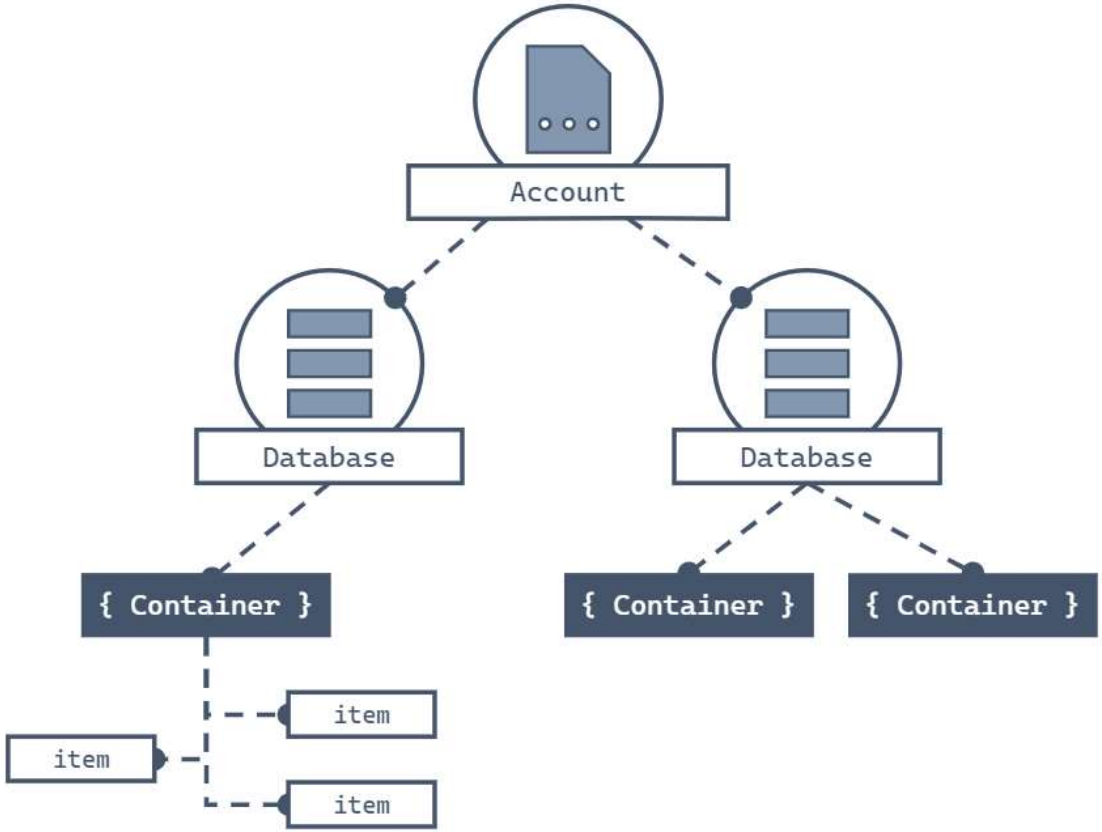
4. Set the Subscription, the resource group, the account name, the location, select the capacity mode, apply the free tier discount, limit total account throughput and press the button “Review + create”



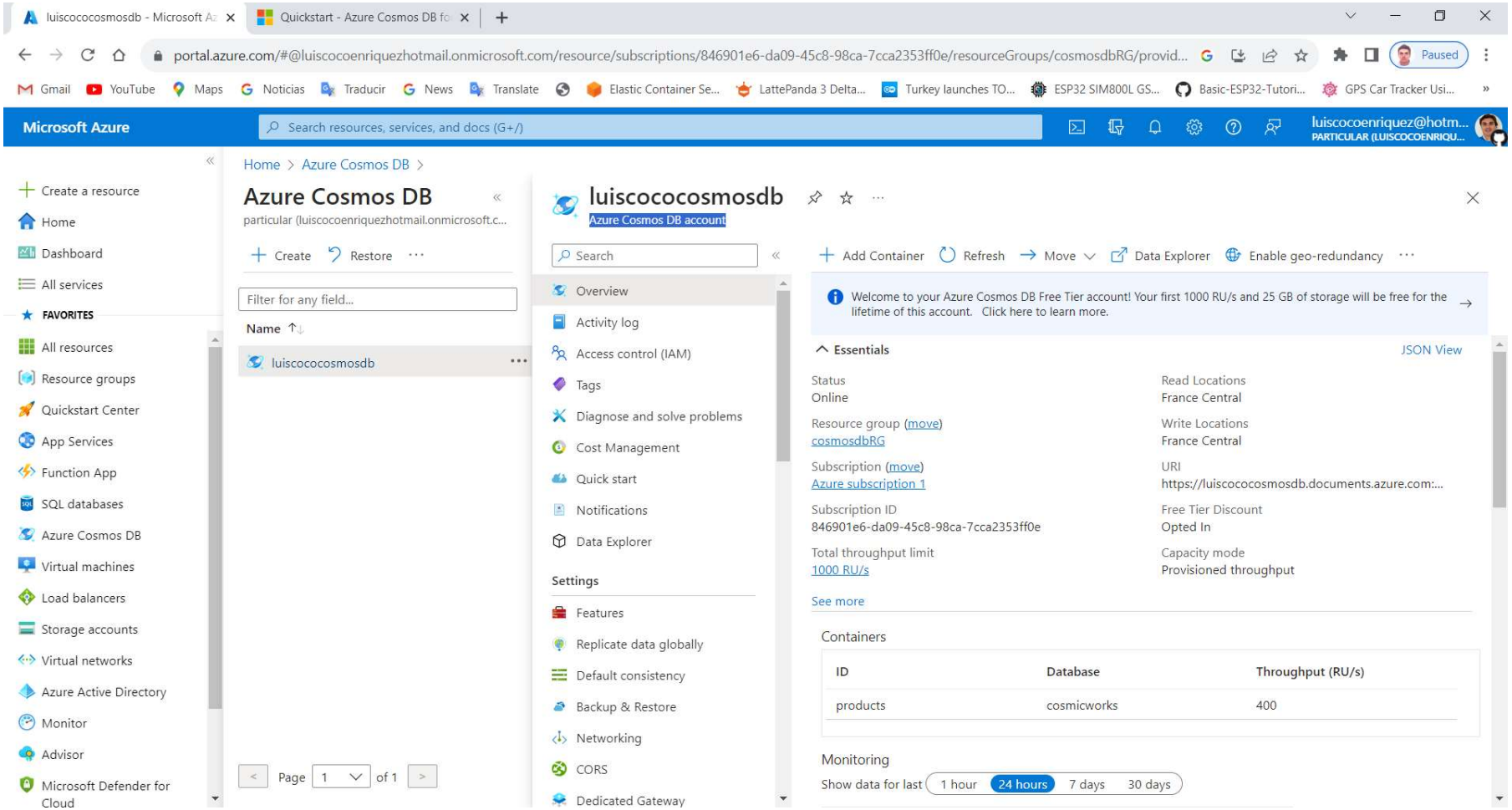
5. Press the CosmosDB link “luiscococosmosdb” to navigate to the **Azure CosmosDB account**



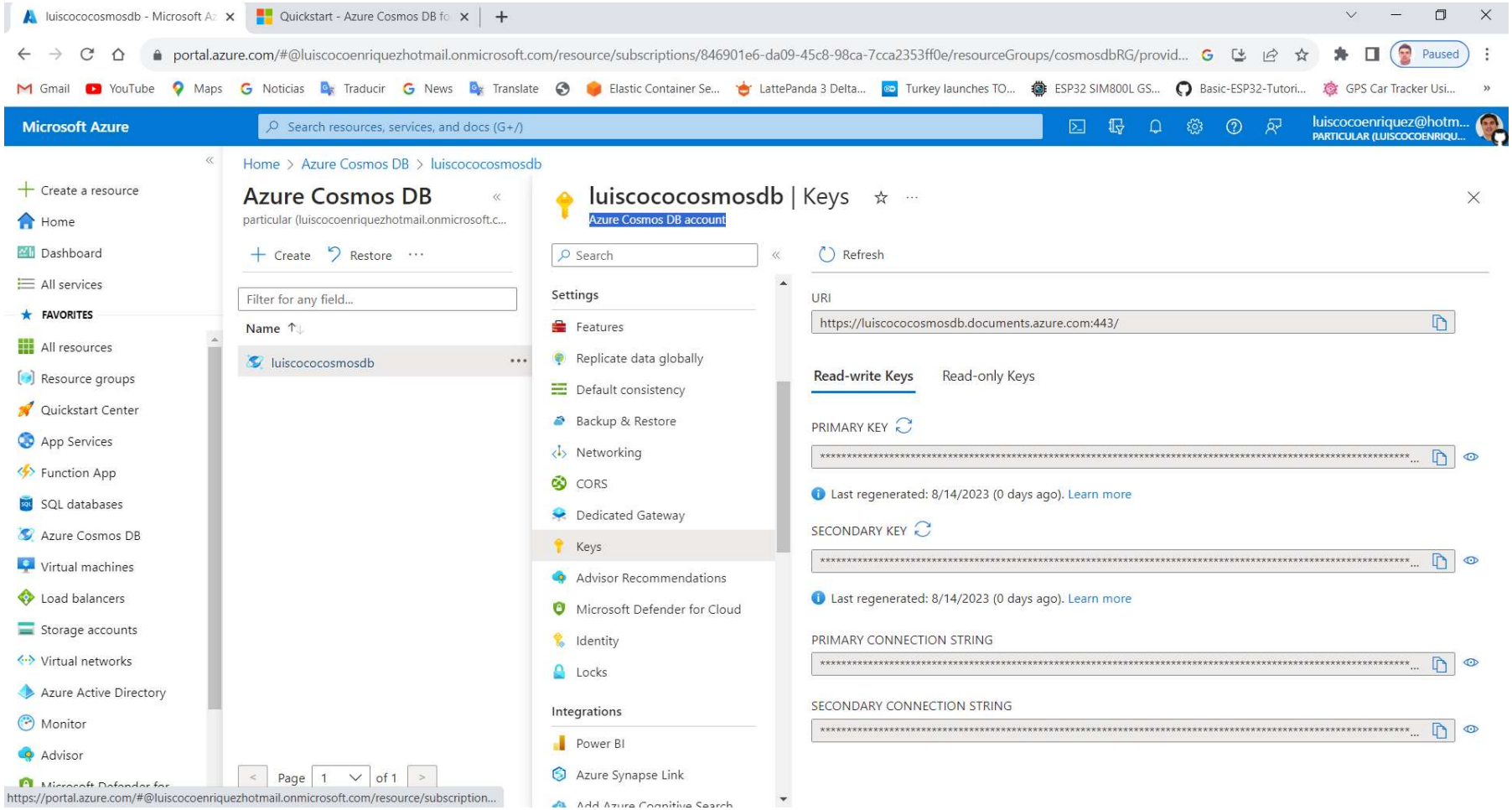
Cosmos DB. Azure Cosmos DB has a specific object model used to create and access resources. The Azure Cosmos DB creates resources in a hierarchy that consists of accounts, databases, containers, and items.



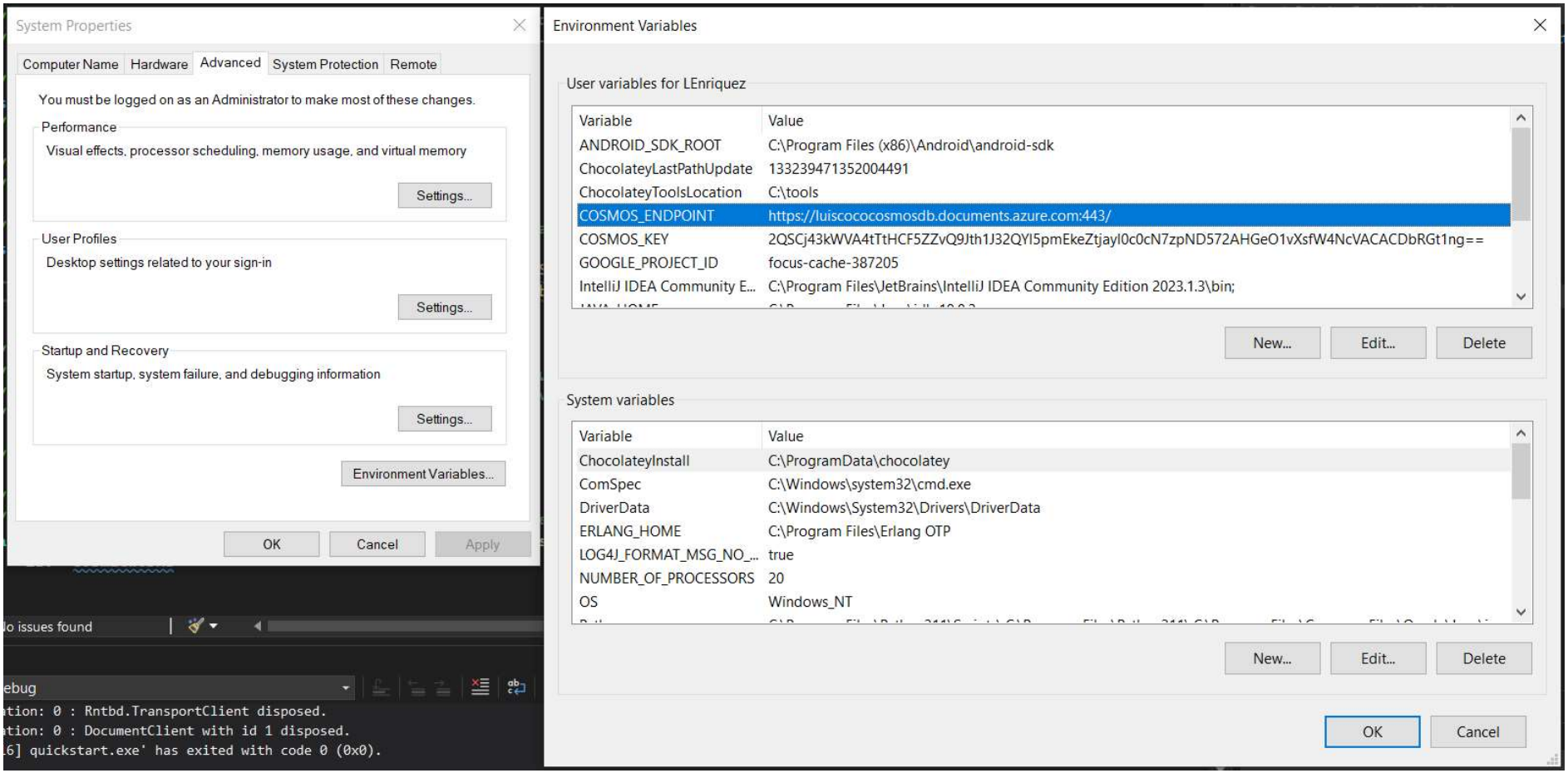
6. Press the CosmosDB link “luiscococosmosdb” to navigate to the **Azure CosmosDB account**



7. In the **Azure CosmosDB account** go to the “Keys” option



8. Set the environmental variables values: the COSMOS_ENDPOINT and the COSMOS_KEY



```
// ----------------------------------------------------------
// Copyright (c) Microsoft Corporation.  All rights reserved.
// ----------------------------------------------------------

// <using_directives>
using Microsoft.Azure.Cosmos;
// </using_directives>

// <client_credentials>
// New instance of CosmosClient class

//We need to set these two variable in the Environment Variables...
using CosmosClient client = new(
    accountEndpoint: Environment.GetEnvironmentVariable("COSMOS_ENDPOINT")!,
    authKeyOrResourceToken: Environment.GetEnvironmentVariable("COSMOS_KEY")!
);
// </client_credentials>

// <new_database>
// Database reference with creation if it does not already exist
Database database = await client.CreateDatabaseIfNotExistsAsync(
    id: "cosmicworks"
);
```

```
Console.WriteLine($"New database:\t{database.Id}");
// </new_database>

// <new_container>
// Container reference with creation if it does not already exist
Container container = await database.CreateContainerIfNotExistsAsync(
    id: "products",
    partitionKeyPath: "/categoryId",
    throughput: 400
);

Console.WriteLine($"New container:\t{container.Id}");
// </new_container>

// <new_item>
// Create new object and upsert (create or replace) to container
Product newItem = new(
    id: "70b63682-b93a-4c77-aad2-65501347265f",
    categoryId: "61dba35b-4f02-45c5-b648-c6badc0cbd79",
    categoryName: "gear-surf-surfboards",
    name: "Yamba Surfboard",
    quantity: 12,
    sale: false
);

Product createdItem = await container.CreateItemAsync<Product>(
    item: newItem,
    partitionKey: new PartitionKey("61dba35b-4f02-45c5-b648-c6badc0cbd79")
);
```

```
// <query_items>
// Create query using a SQL string and parameters
var query = new QueryDefinition(
    query: "SELECT * FROM products p WHERE p.categoryId = @categoryId"
)
    .WithParameter("@categoryId", "61dba35b-4f02-45c5-b648-c6badc0cbd79");

using FeedIterator<Product> feed = container.GetItemQueryIterator<Product>(
    queryDefinition: query
);

while (feed.HasMoreResults)
{
    FeedResponse<Product> response = await feed.ReadNextAsync();
    foreach (Product item in response)
    {
        Console.WriteLine($"Found item:\t{item.name}");
    }
}
// </query_items>
```