IT457 Cloud computing

Assignment - 7

Abhinav Agarwal (202101040)

**Q8:**

Code:-

using Azure.Storage.Blobs;

using System;

using System.IO;

using System.Security.Cryptography;

using System.Threading.Tasks;

class Program

{

    static async Task Main(string[] args)

    {

        string connectionString = "<your\_connection\_string>";

        string filePath = "<path\_to\_your\_file>";

        string containerName = "<your\_container\_name>";

        string blobName = "<your\_blob\_name>";

        await UploadLargeFileToBlob(connectionString, filePath, containerName, blobName);

    }

    static async Task UploadLargeFileToBlob(string connectionString, string filePath, string containerName, string blobName)

    {

        BlobServiceClient blobServiceClient = new BlobServiceClient(connectionString);

        BlobContainerClient containerClient = blobServiceClient.GetBlobContainerClient(containerName);

        BlobClient blobClient = containerClient.GetBlobClient(blobName);

        long blockSize = 4 \* 1024 \* 1024; // 4MB block size

        byte[] buffer = new byte[blockSize];

        using (FileStream fileStream = File.OpenRead(filePath))

        {

            long fileSize = fileStream.Length;

            long bytesUploaded = 0;

            string[] blockIds = new string[(int)Math.Ceiling((double)fileSize / blockSize)];

            while (bytesUploaded < fileSize)

            {

                int bytesRead = await fileStream.ReadAsync(buffer, 0, (int)Math.Min(blockSize, fileSize - bytesUploaded));

                // Calculate MD5 hash of the block

                using (MD5 md5 = MD5.Create())

                {

                    byte[] blockHash = md5.ComputeHash(buffer, 0, bytesRead);

                    string blockId = Convert.ToBase64String(blockHash);

                    // Upload the block

                    using (MemoryStream memoryStream = new MemoryStream(buffer, 0, bytesRead))

                    {

                        await blobClient.StageBlockAsync(blockId, memoryStream);

                    }

                    blockIds[bytesUploaded / blockSize] = blockId;

                }

                bytesUploaded += bytesRead;

            }

            // Commit the list of blocks

            await blobClient.CommitBlockListAsync(blockIds);

        }

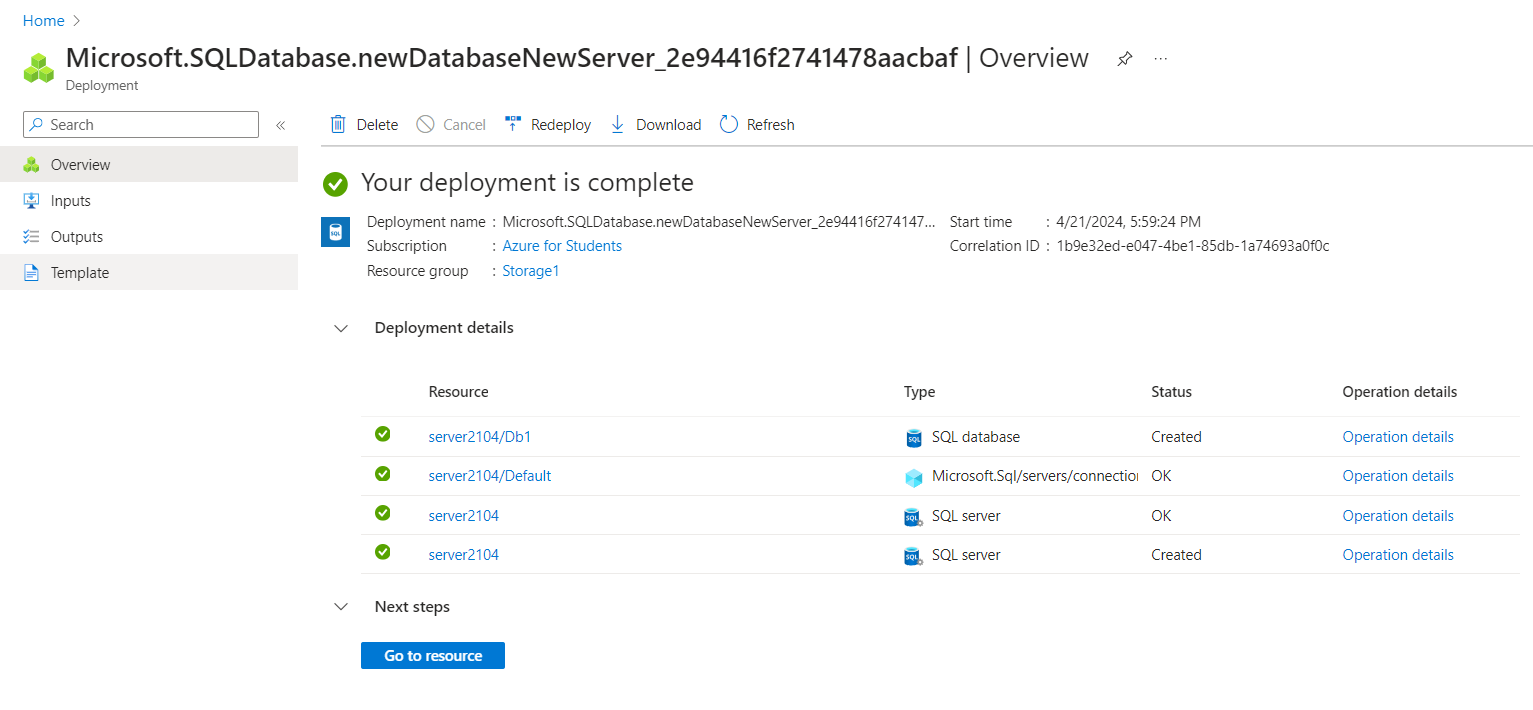
        Console.WriteLine("File uploaded successfully.");

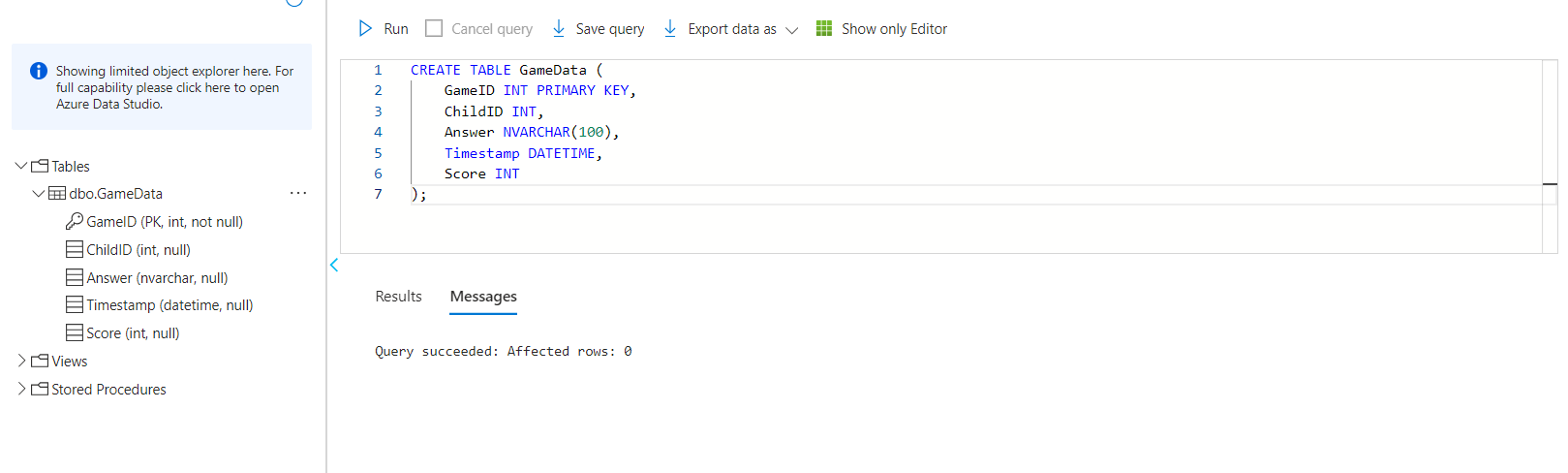
    }

}

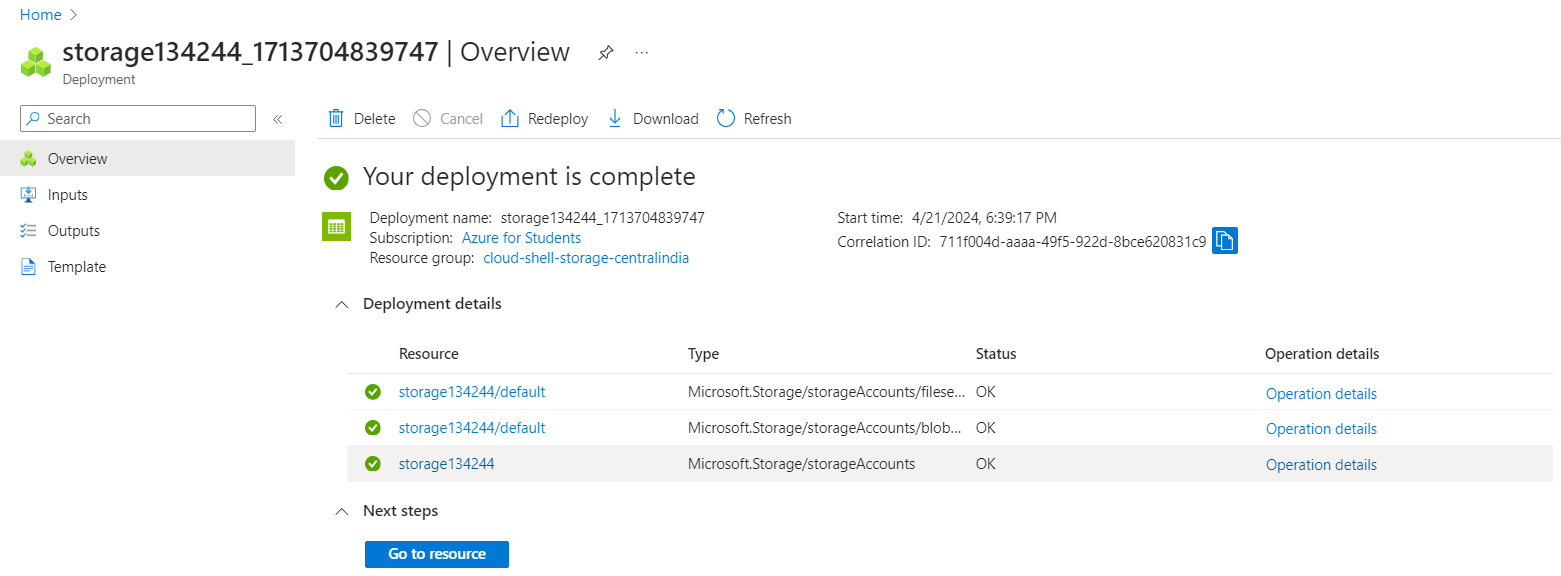
**Q11:**

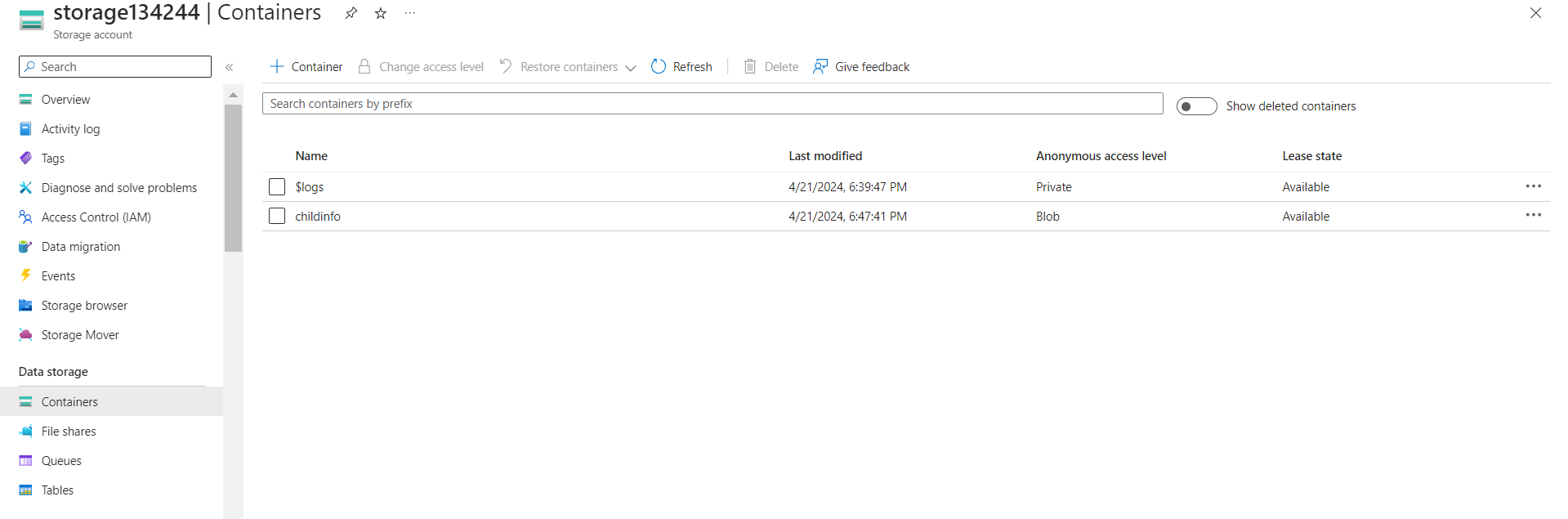
Creating SQL database:-



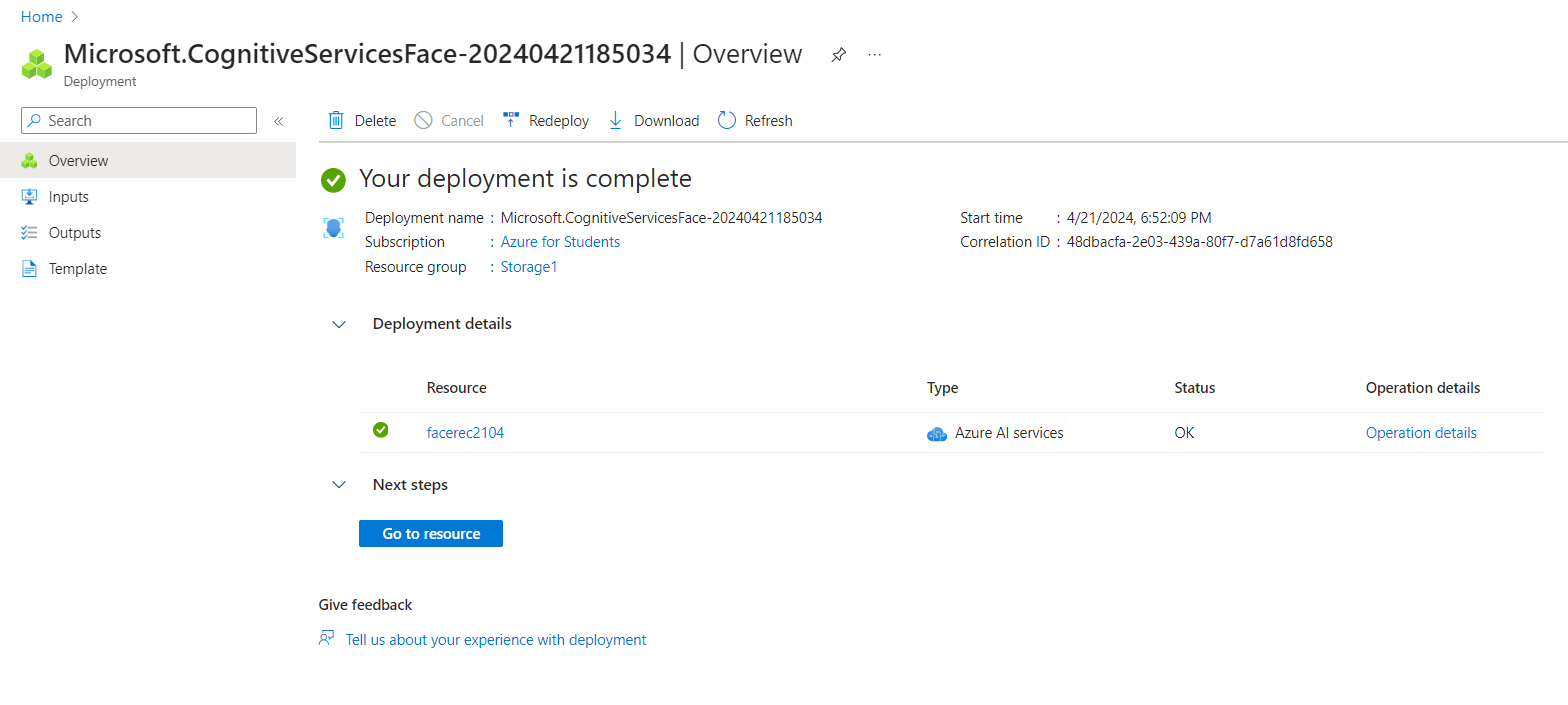


Creating blob storage:-

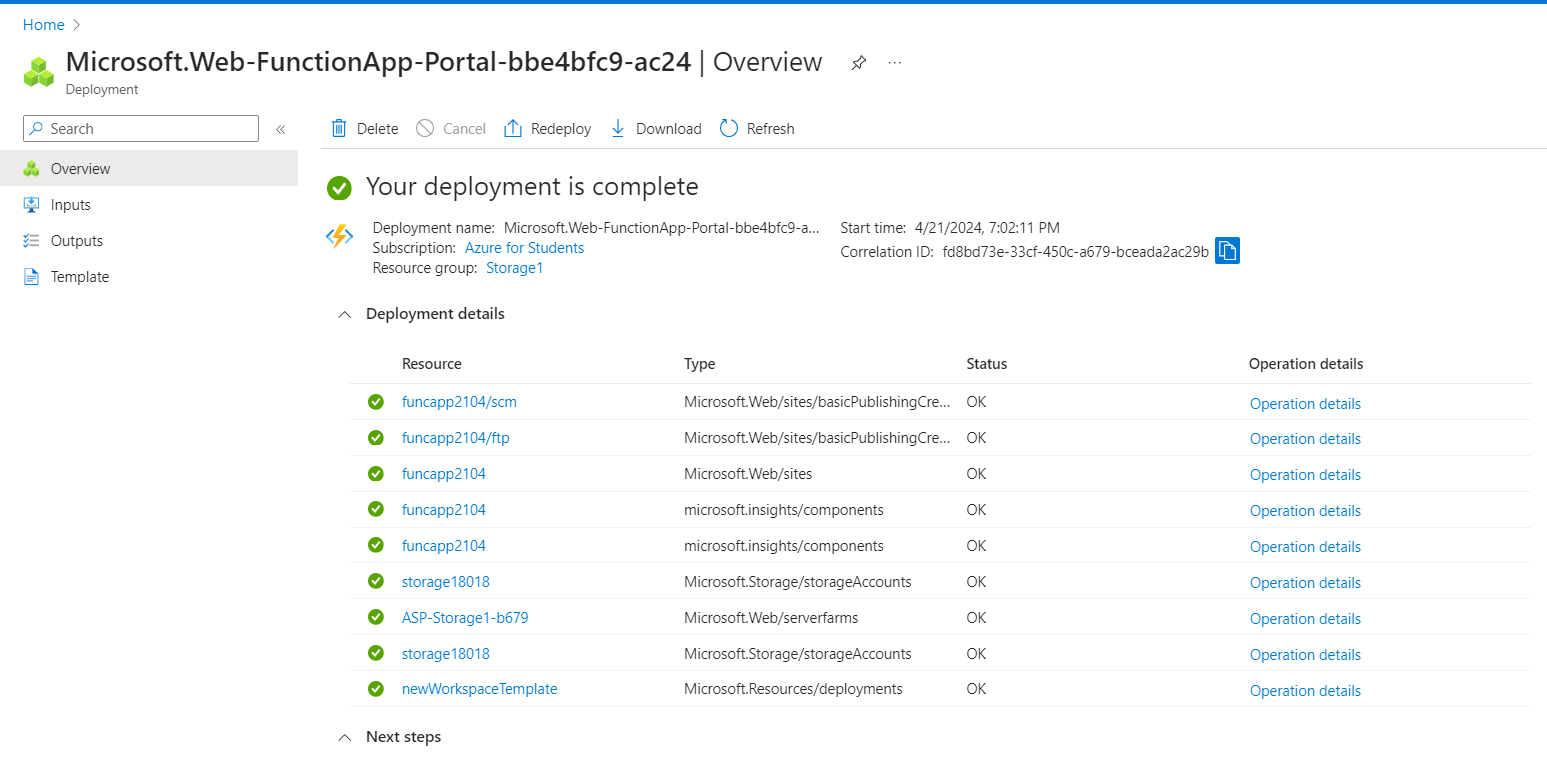




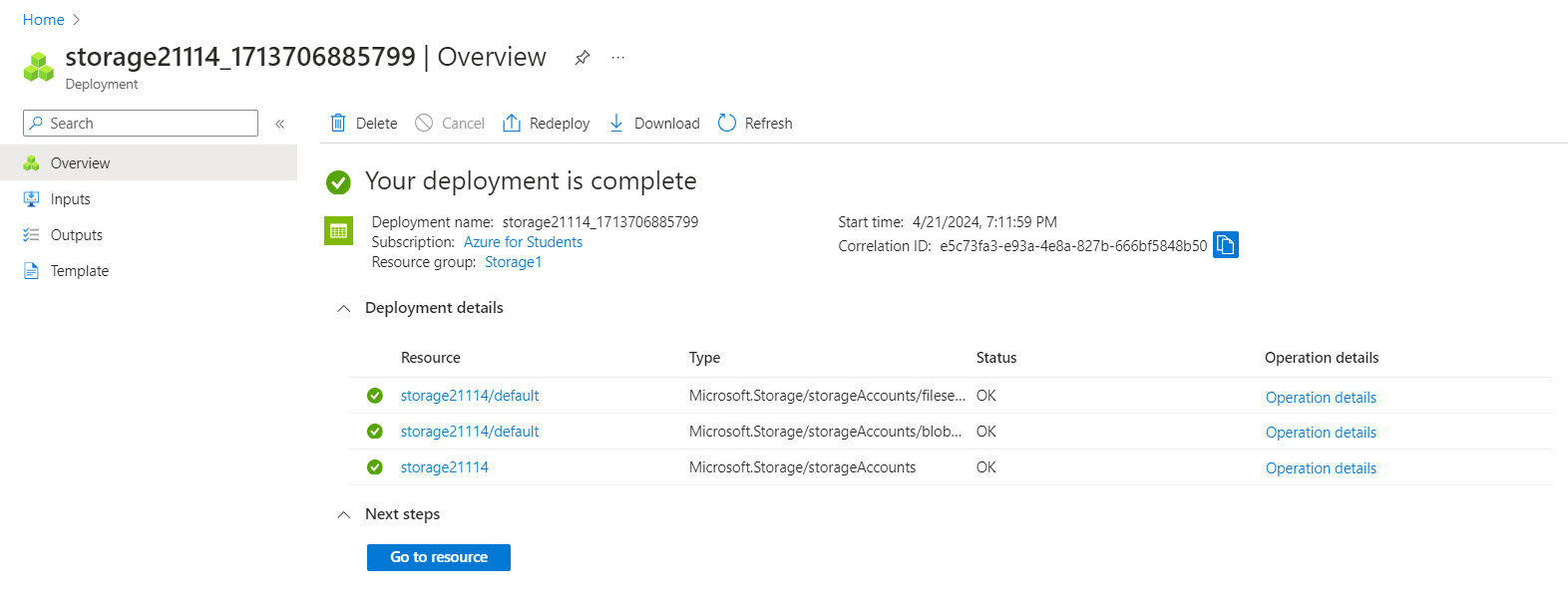
Creating Face API:-

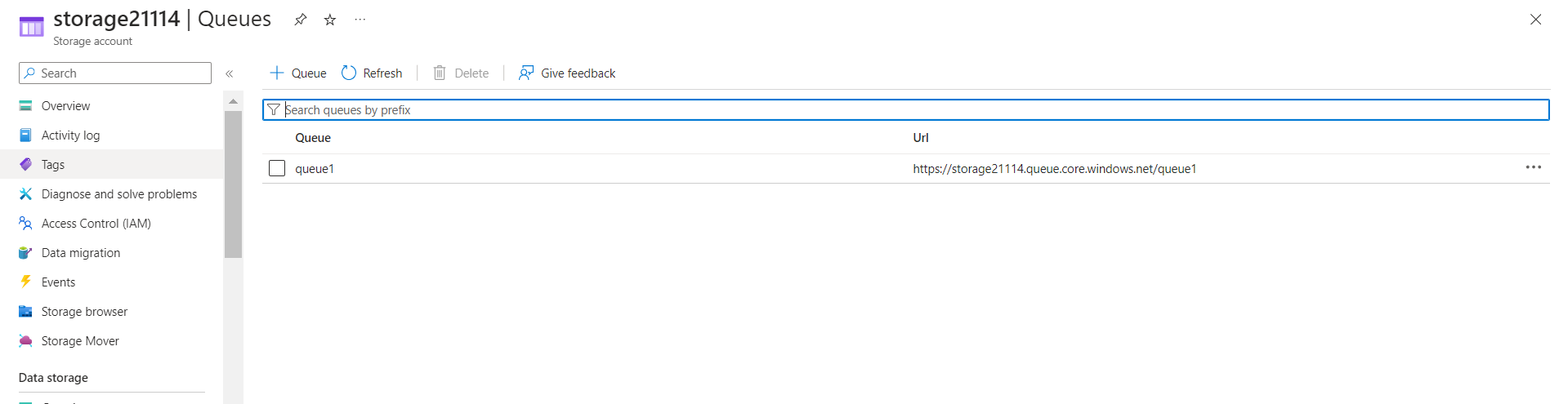


Creating Function App:-

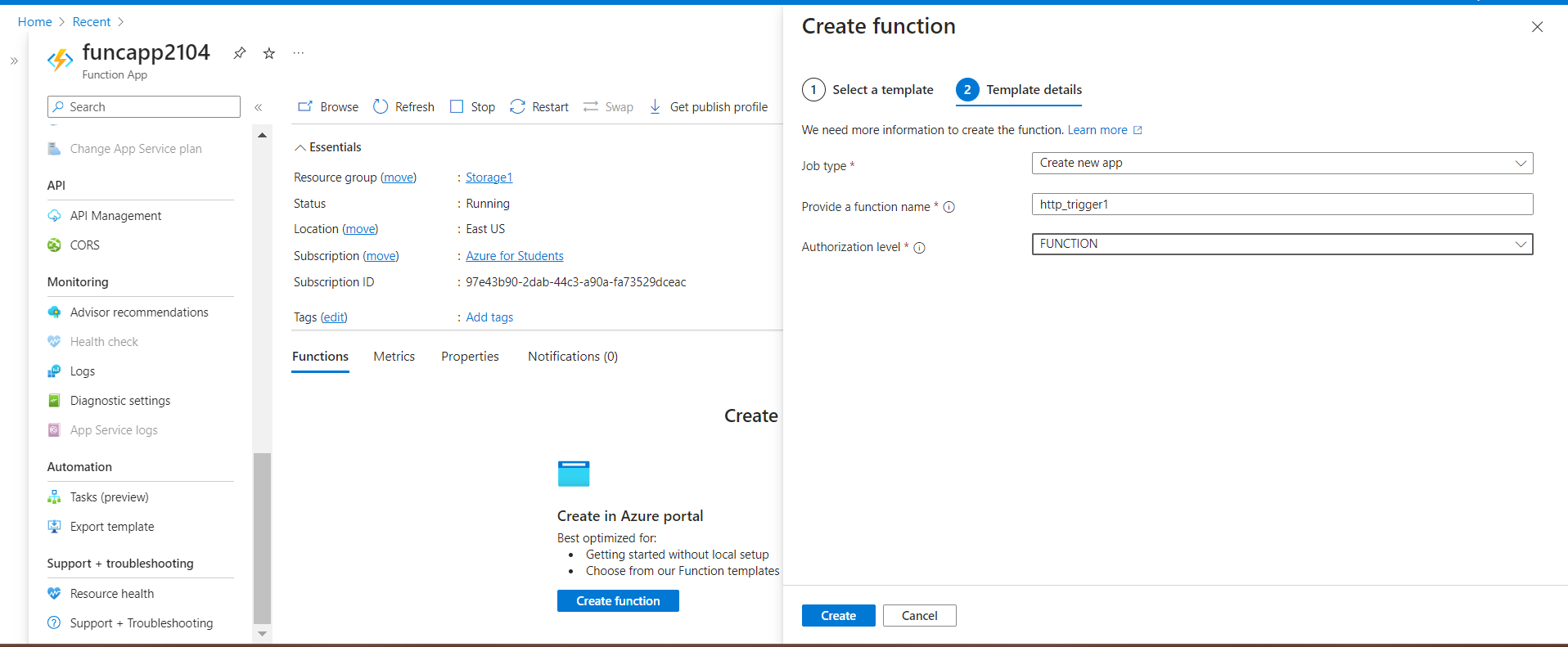


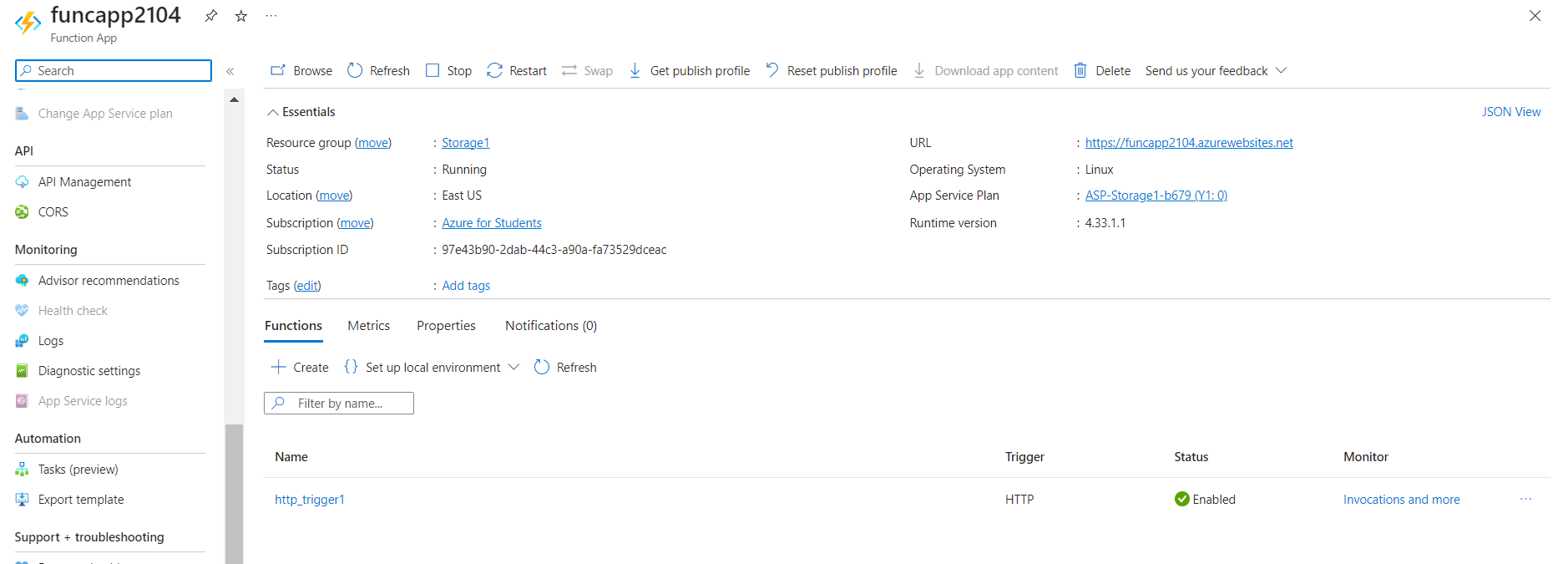
Creating Queue storage:-

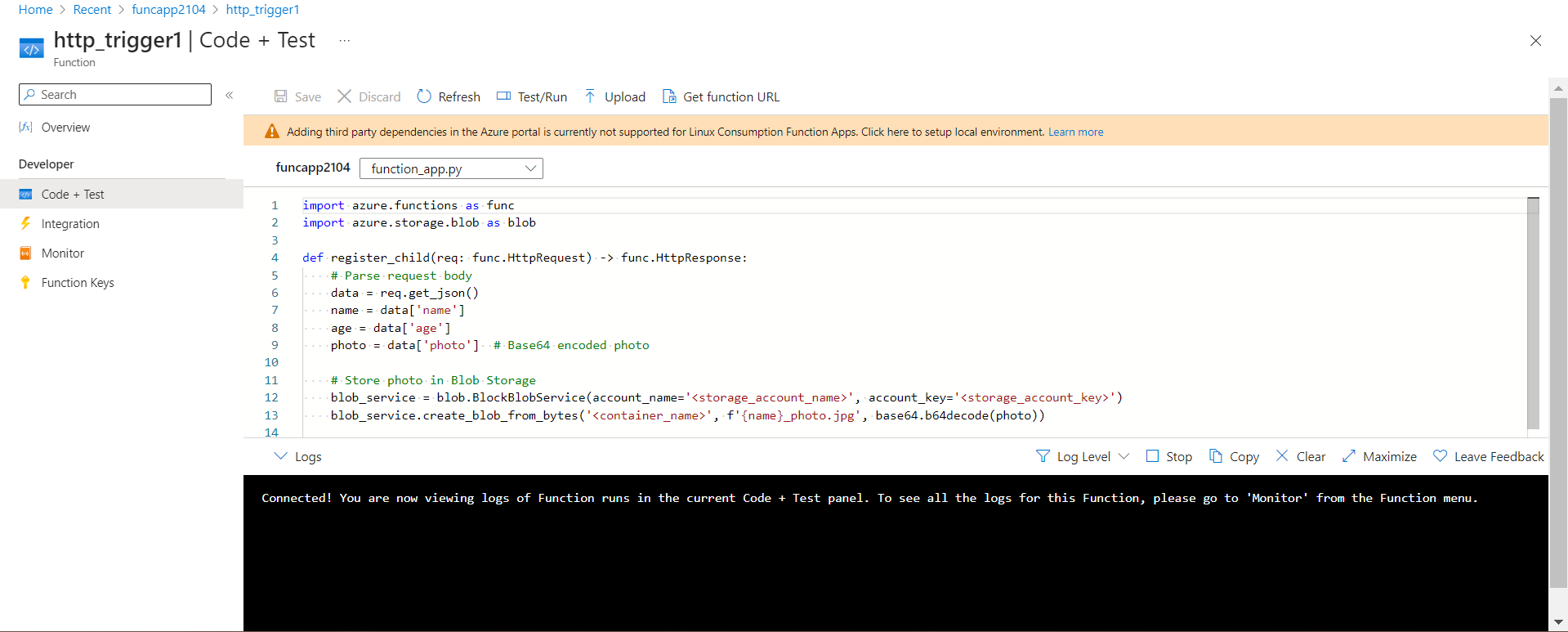




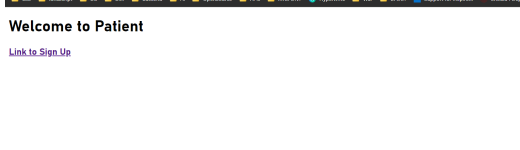
Creating function in function app:-

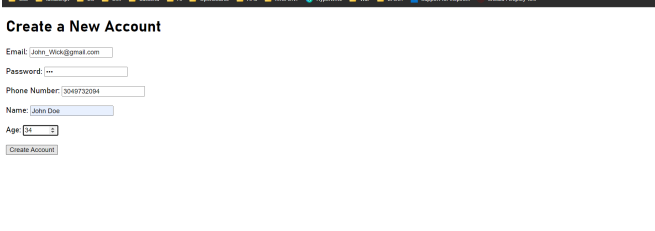




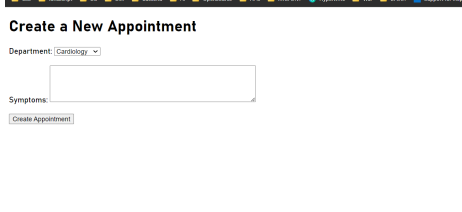


Result:-







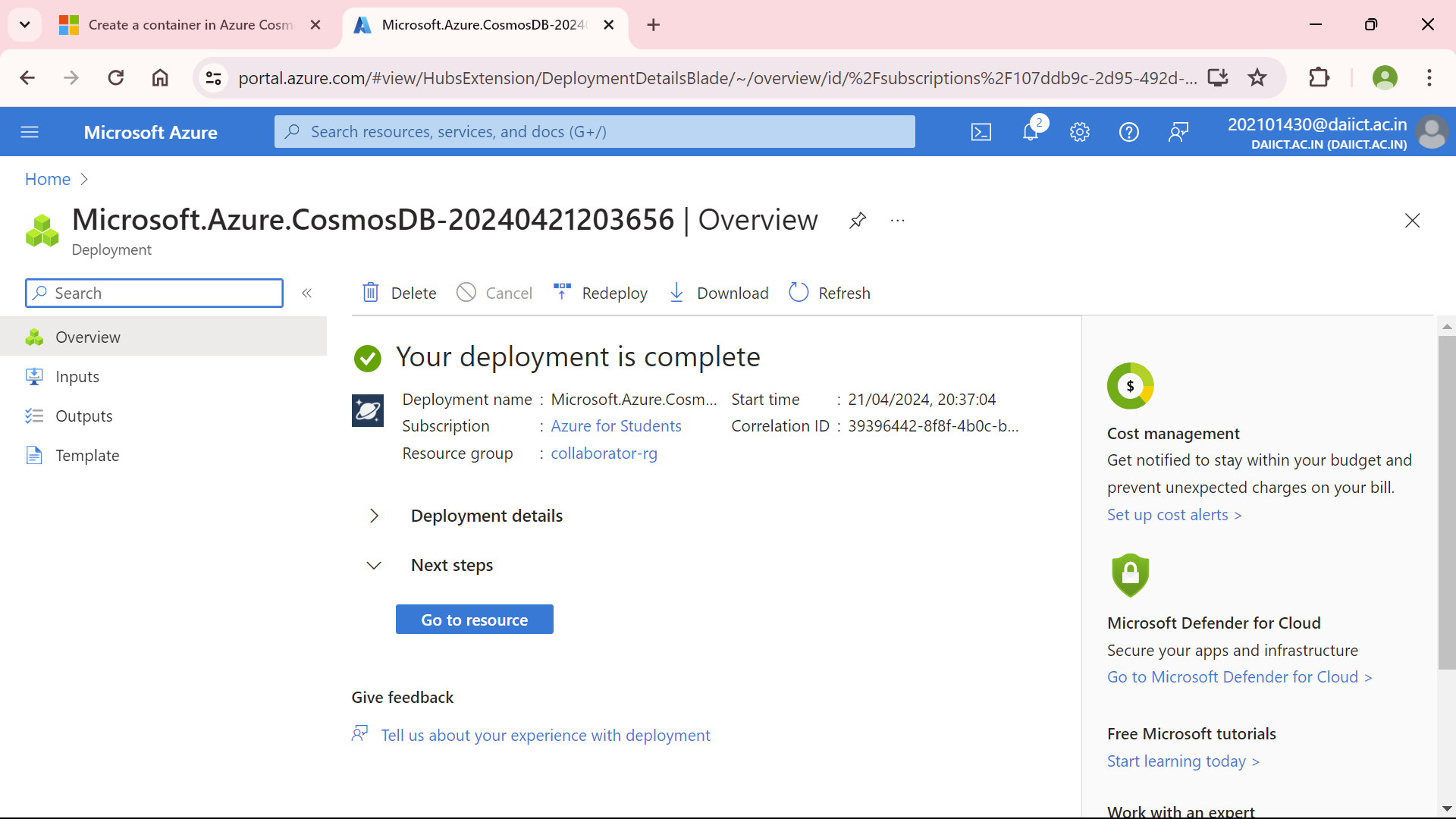




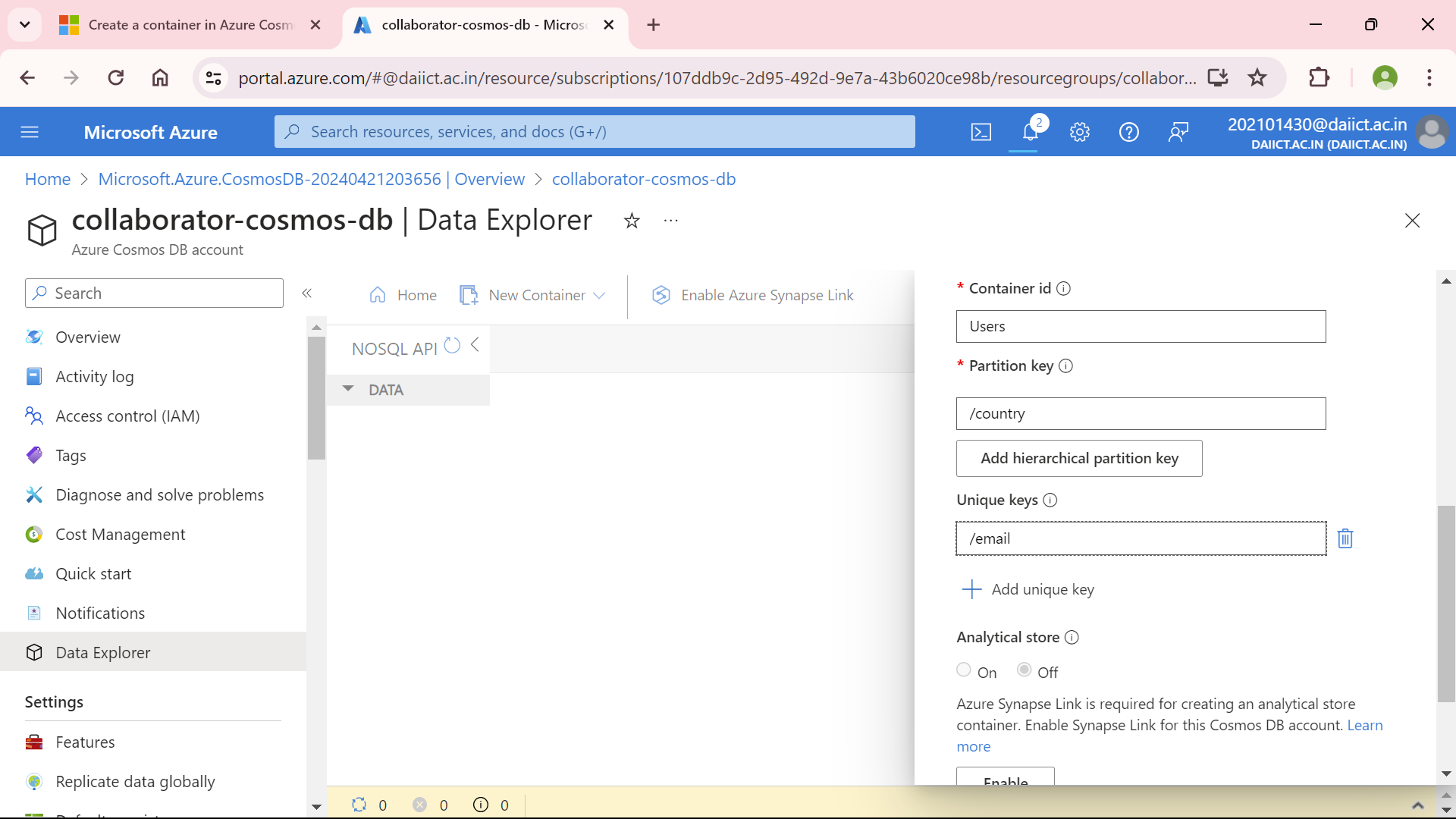


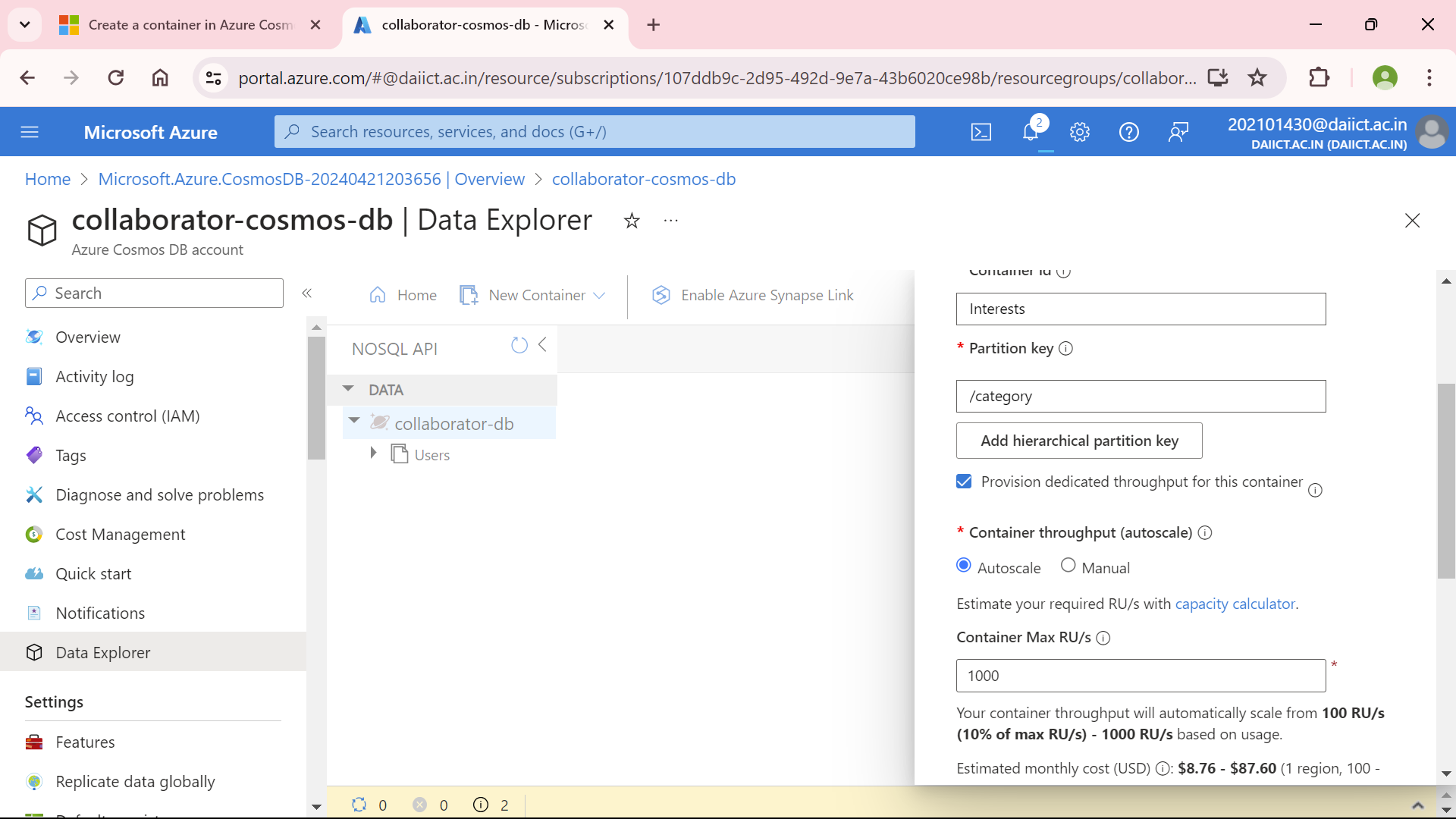
Q9:

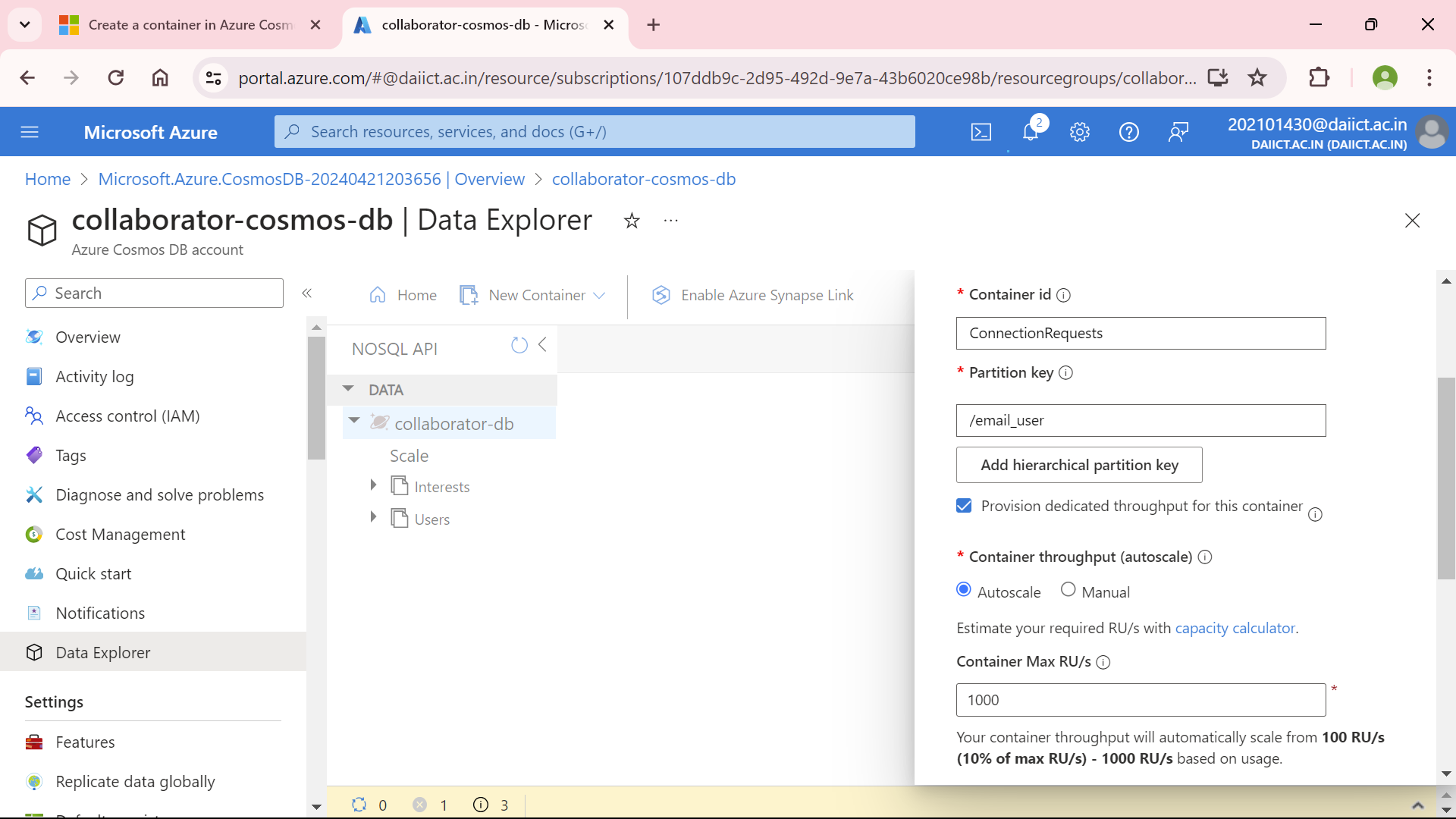
Creation of the CosmosDB database:

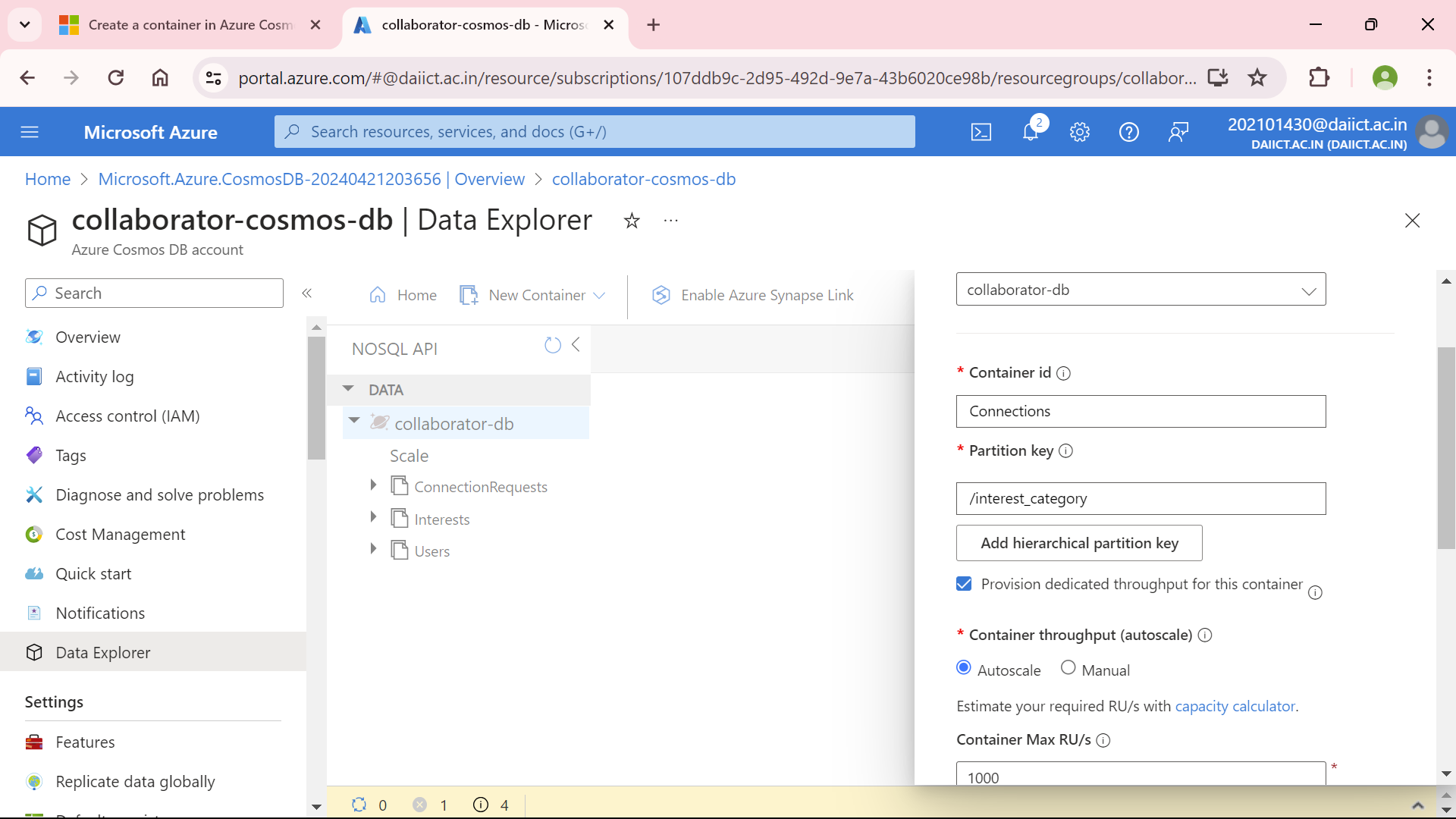


Creation of various tables:

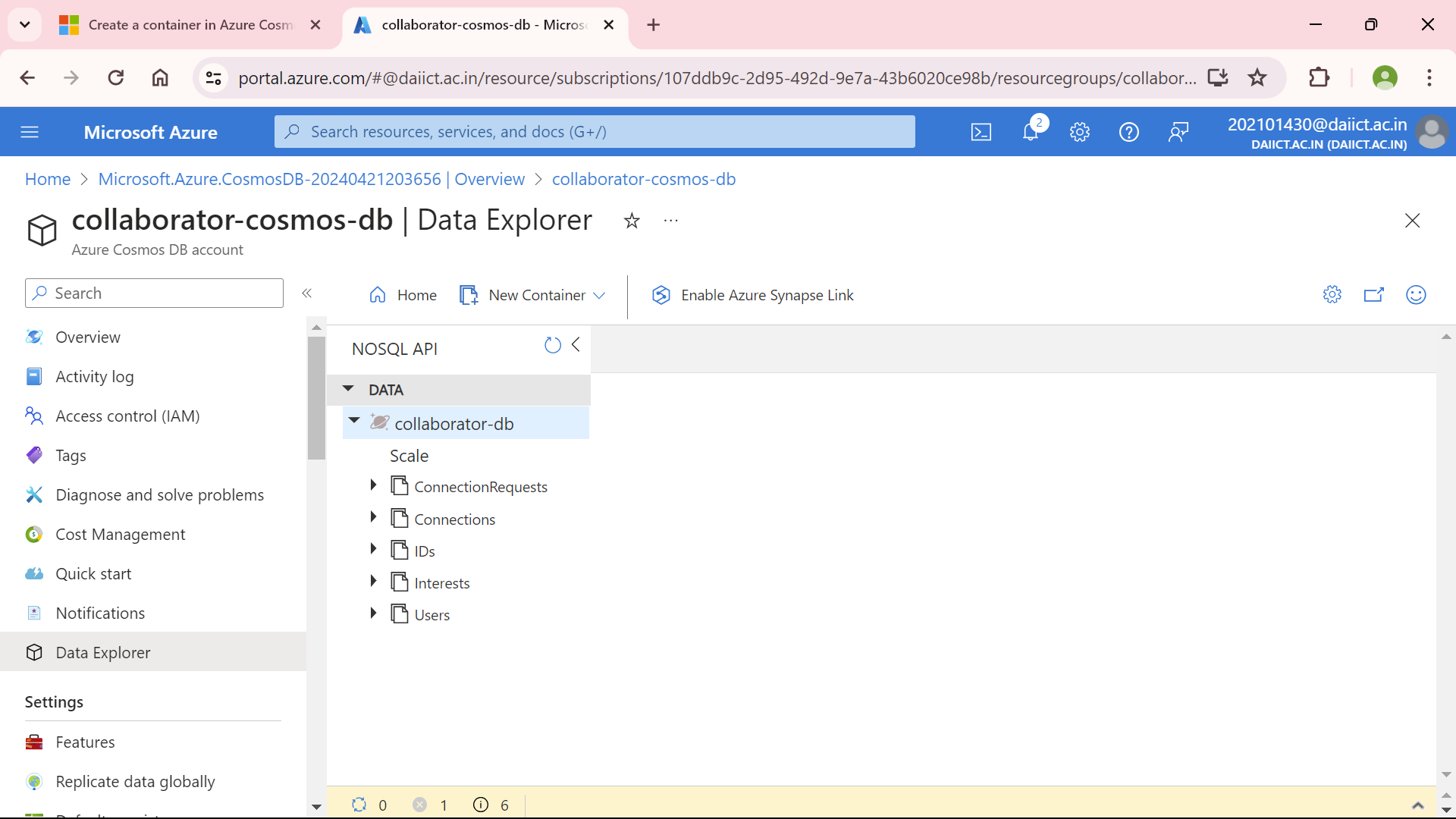








All tables required for the application:



Result:-

