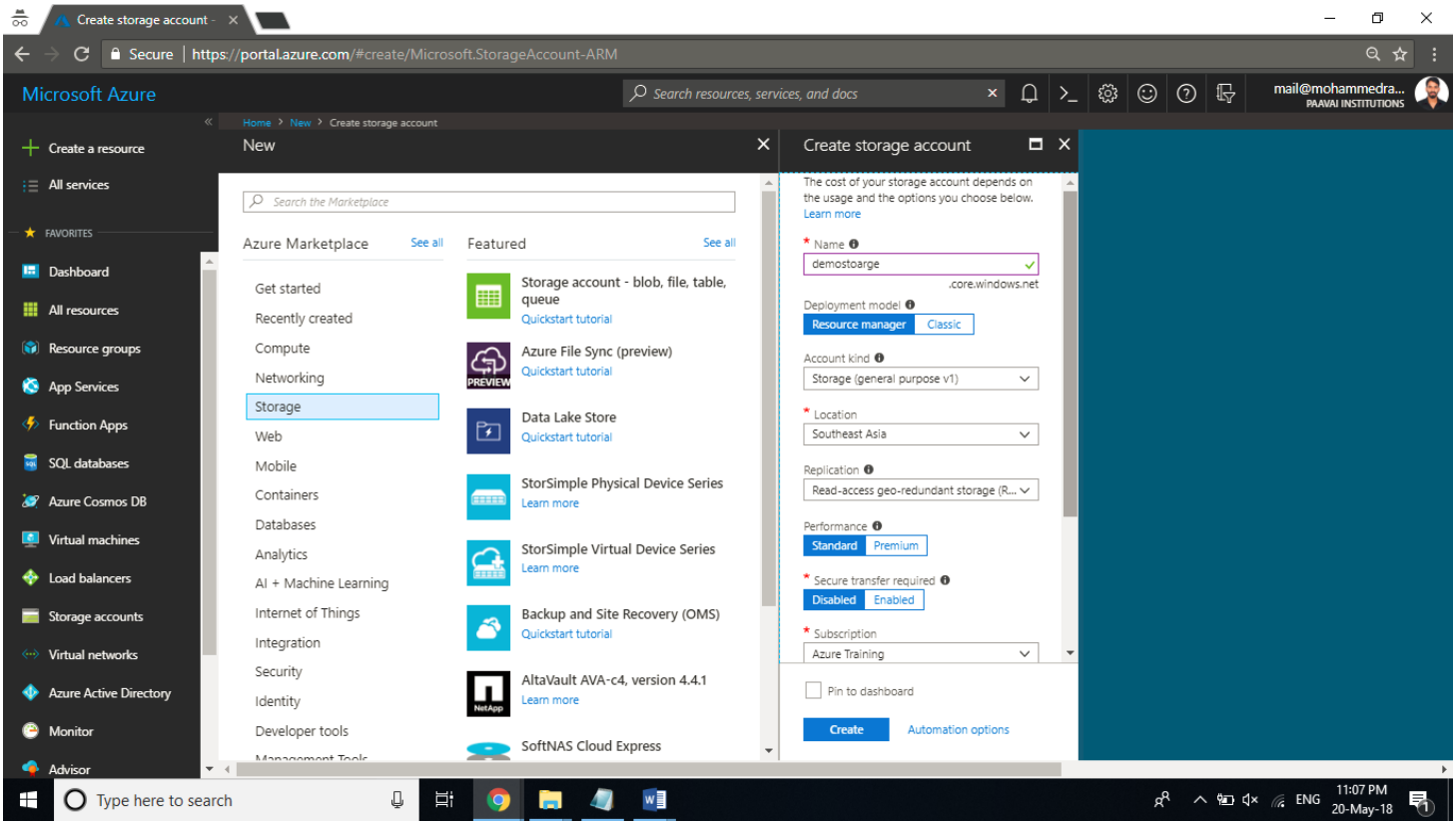
Azure Blob Storage - Creating Storage Account and Container

Step 1: Login to your Microsoft Azure portal, click on **Create new resource** and select **Storage account - blob, file, table, queue** from **Storage** category to create a **Storage account** and give the necessary details as follows.

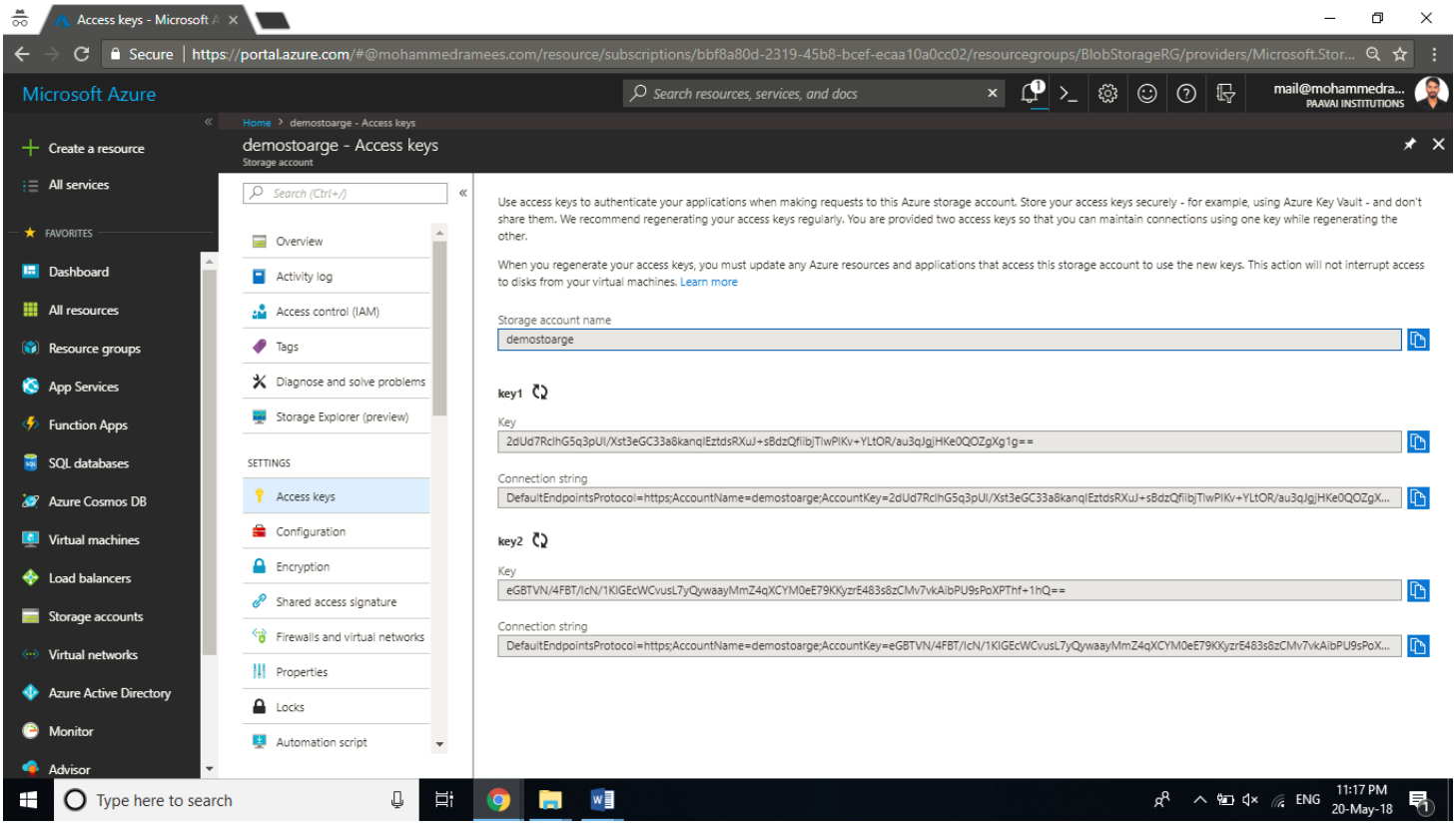
Name: A Unique Name
Deployment: Resource manager
Account kind: General purpose (Can't work with Files, Tables or Queues in case of Blob Storage)
Location: Nearest to you
Replication: Read-access geo-redundant storage

Locally-redundant storage- Three copies within the same data center
Zone-redundant storage- Three copies within two data centers of same region
Geo-redundant storage- Three copies in a primary data center and three copies in another data center

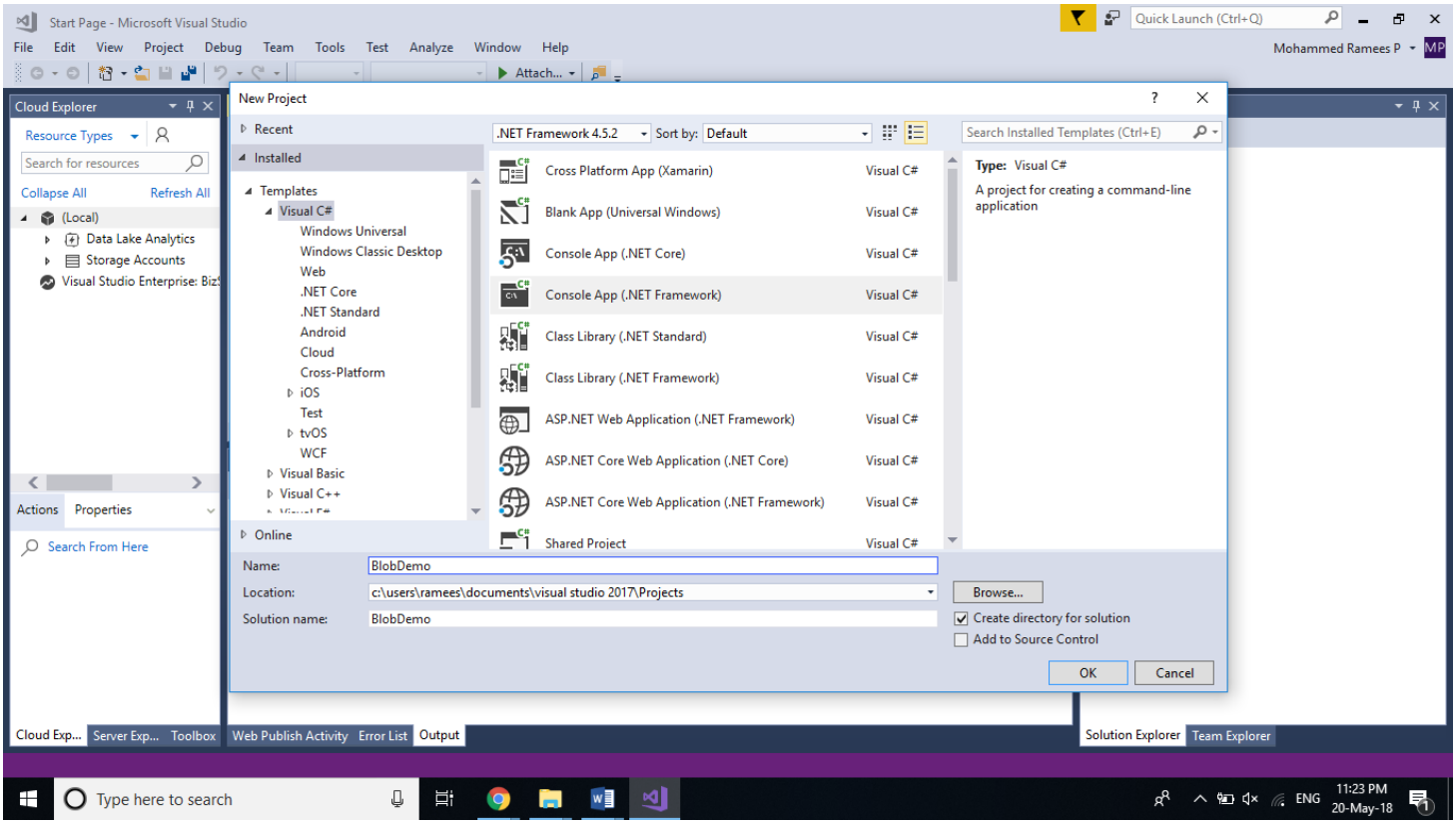
Performance: Standard (For premium only LRS replication is available as of now)
Secure Transfer Required: Disabled
Resource group: Create a new as BlobStorageRG



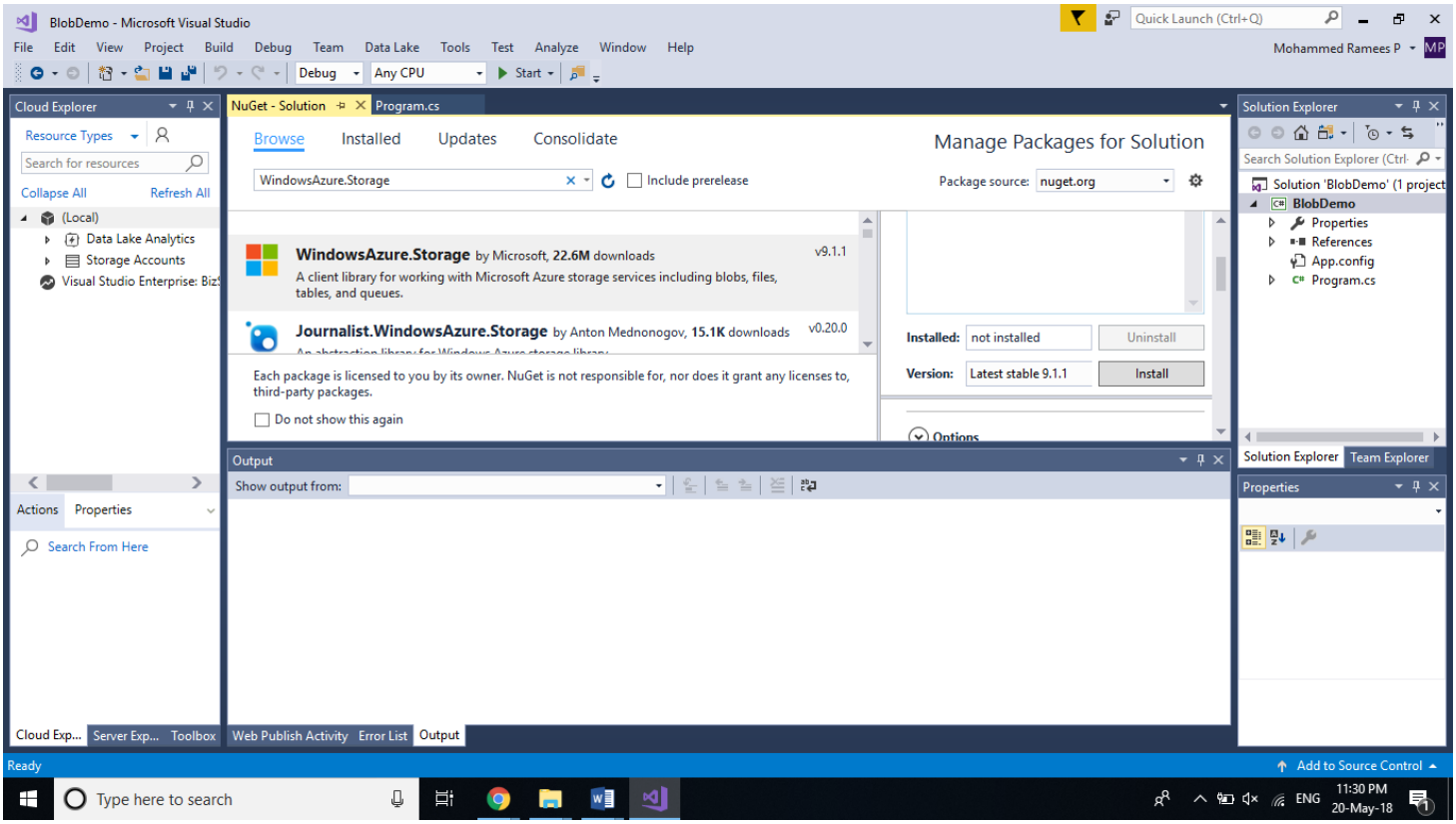
Step 2: Once the deployment is succeeded, open the resource and click on **Access keys** and note a key for a later purpose.

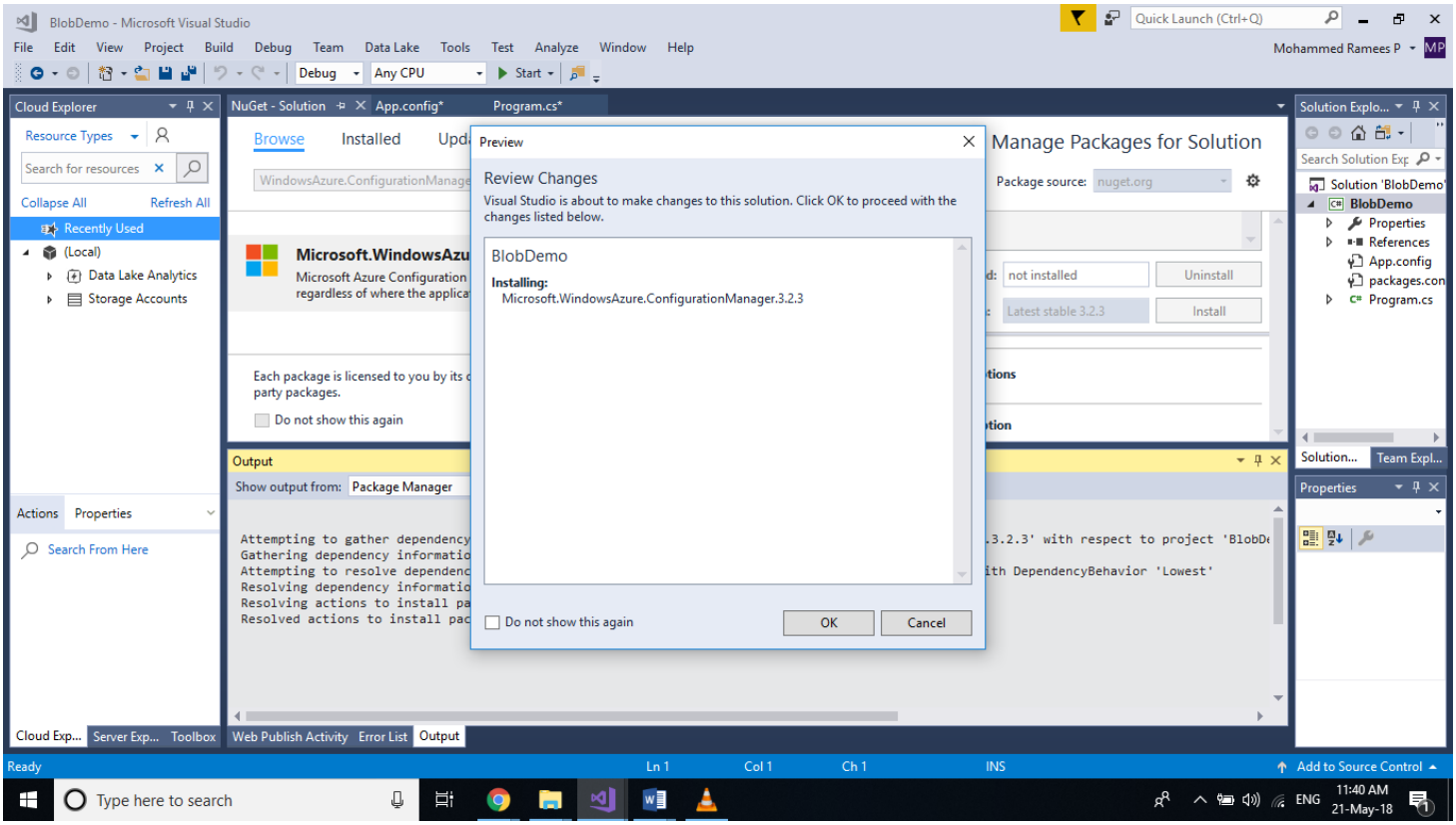


Step 3: Create a new C# **Console Application** within Visual Studio and name it as BlobDemo.



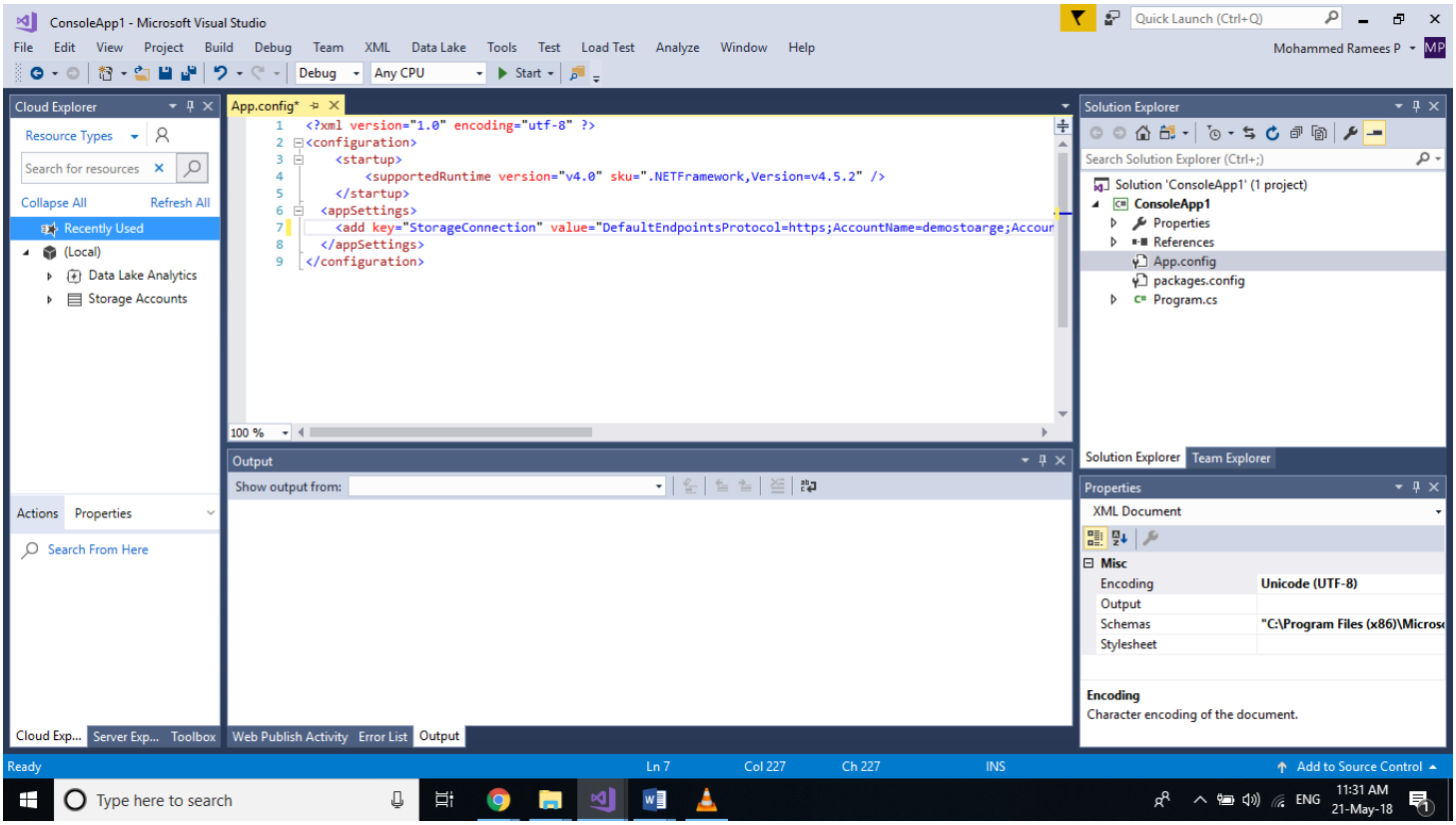
Step 4: Once the project is created, **install two NuGet Packages** named **WindowsAzure.Storage** and **WindowsAzure.ConfigurationManager** to your solution. You can do this by selecting Tools > NuGet Package Manager > Manage NuGet Packages for Solution. In the Browse tab search for WindowsAzure.Storage and WindowsAzure.ConfigurationManager to get the packages complete the installation by reviewing the changes and accepting the license term for both.





Step 5: Now we have to add the **Connection String** to the storage account. For that open the App.config file and add following code snippet within the configuration. Replace the [Connection string] with the connection string value you copied in Step 2.

```
<appSettings>
    <add key="StorageConnection" value="[Connection String]" />
  </appSettings>
```

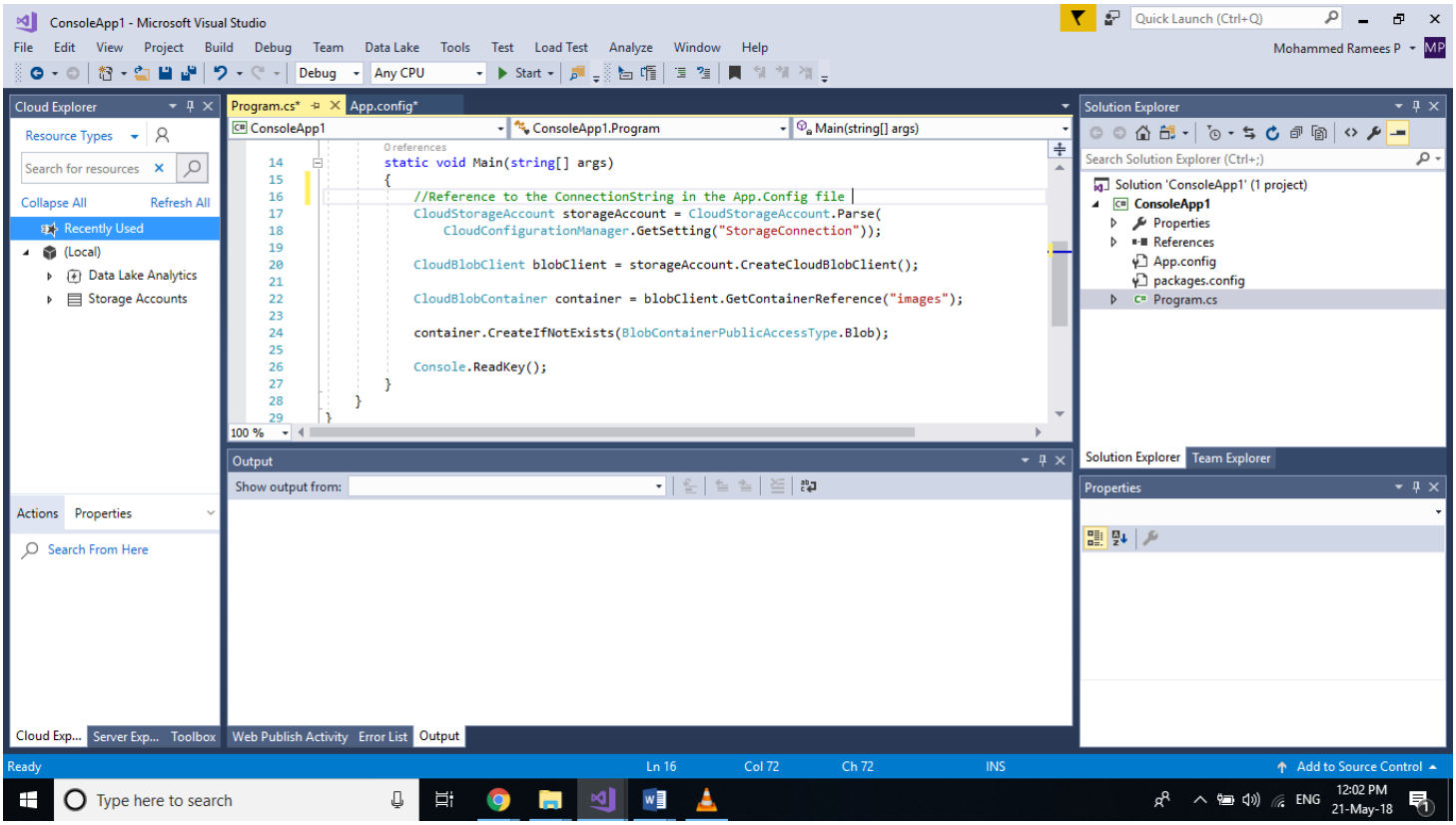


Step 6: Now go to the **Program.cs** Page and add the following code to the Main method. This code will create a connection with the storage account and create a container named images in our blob storage with a Public (Blob) access type.

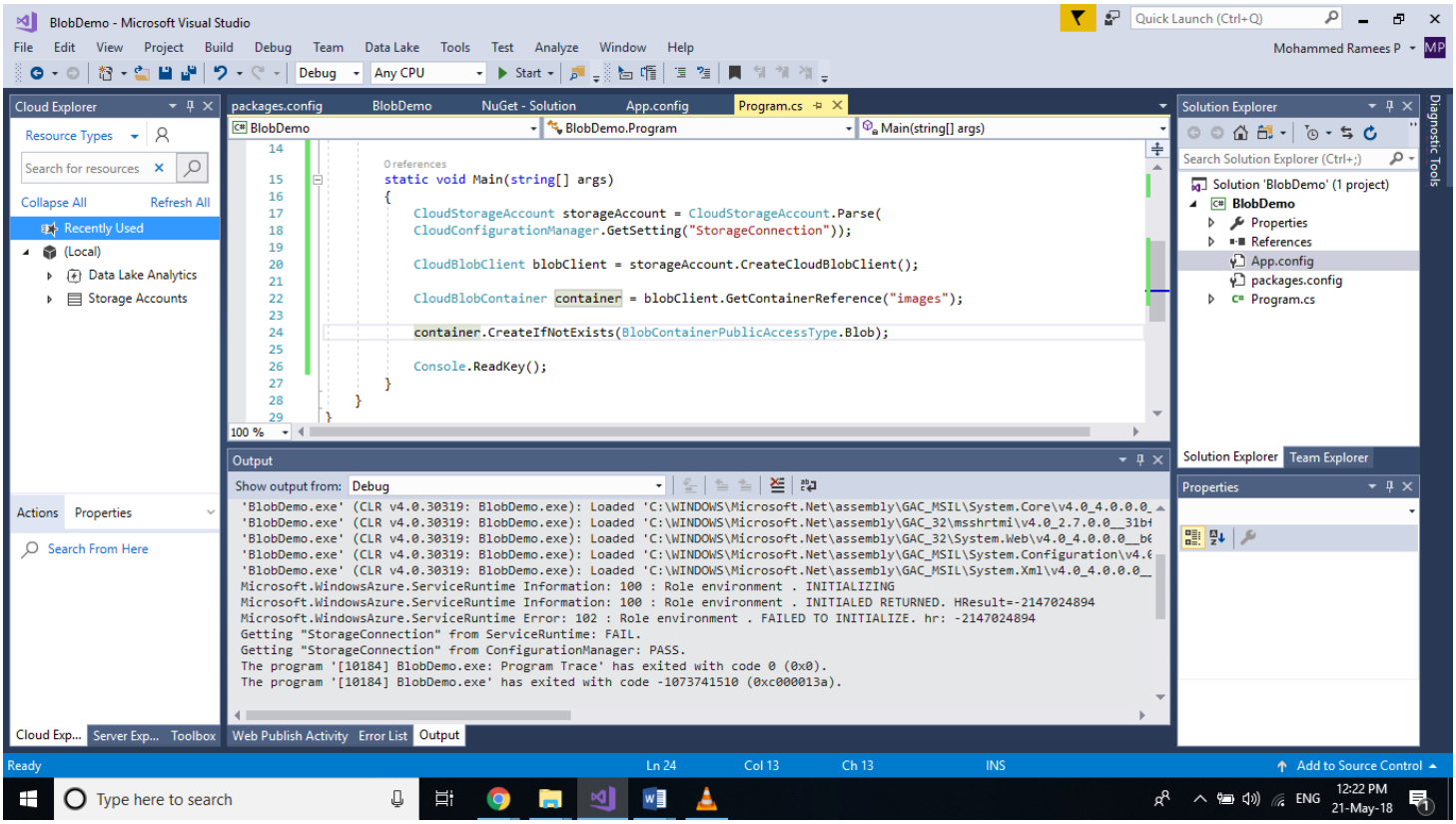
```
//Reference to the ConnectionString in the App.Config file
            CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
                CloudConfigurationManager.GetSetting("StorageConnection"));

            CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

            CloudBlobContainer container = blobClient.GetContainerReference("images");

            container.CreateIfNotExists(BlobContainerPublicAccessType.Blob);

            Console.ReadKey();
```

Add the following namespaces also

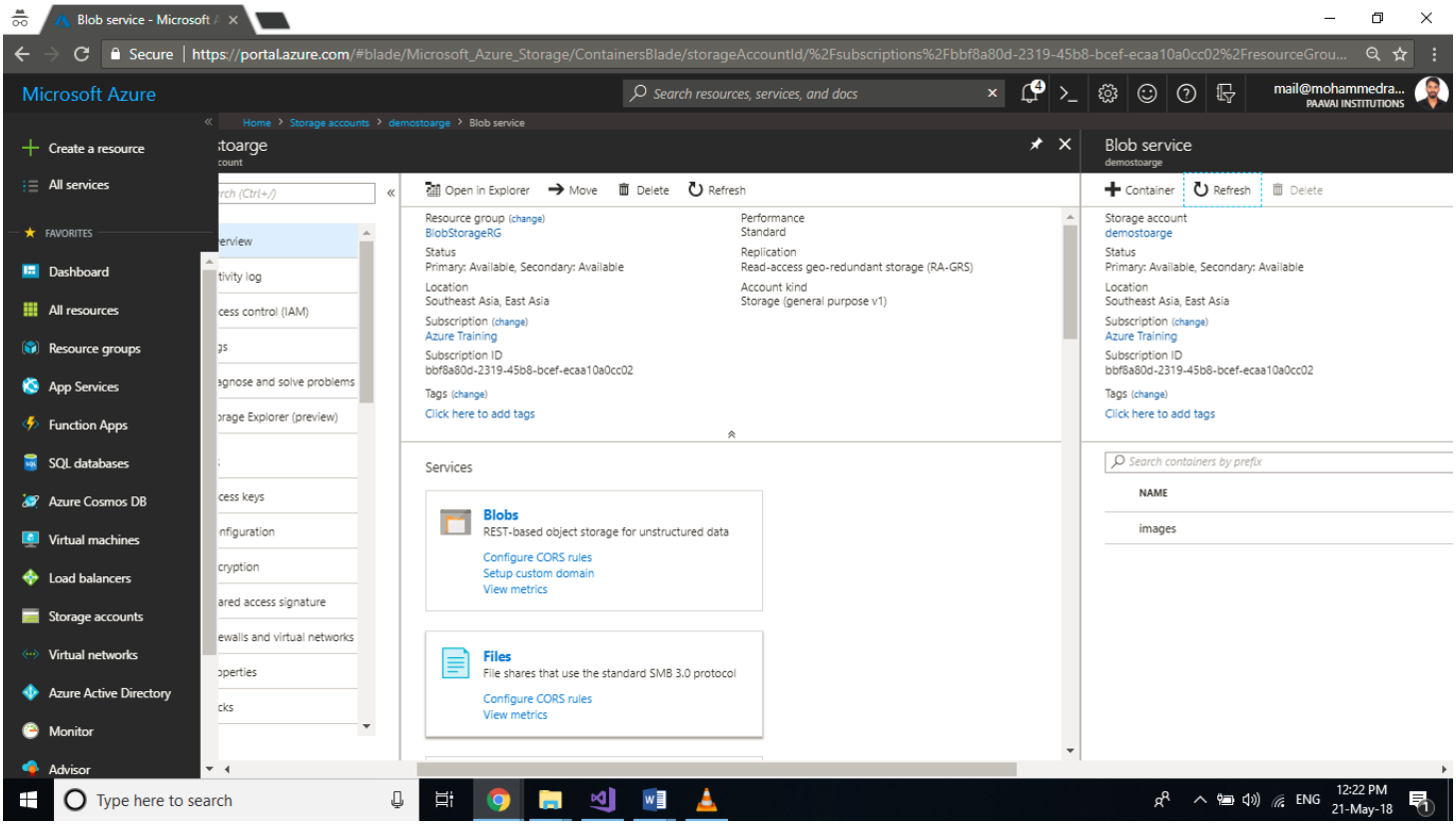
```
using Microsoft.Azure;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;
```

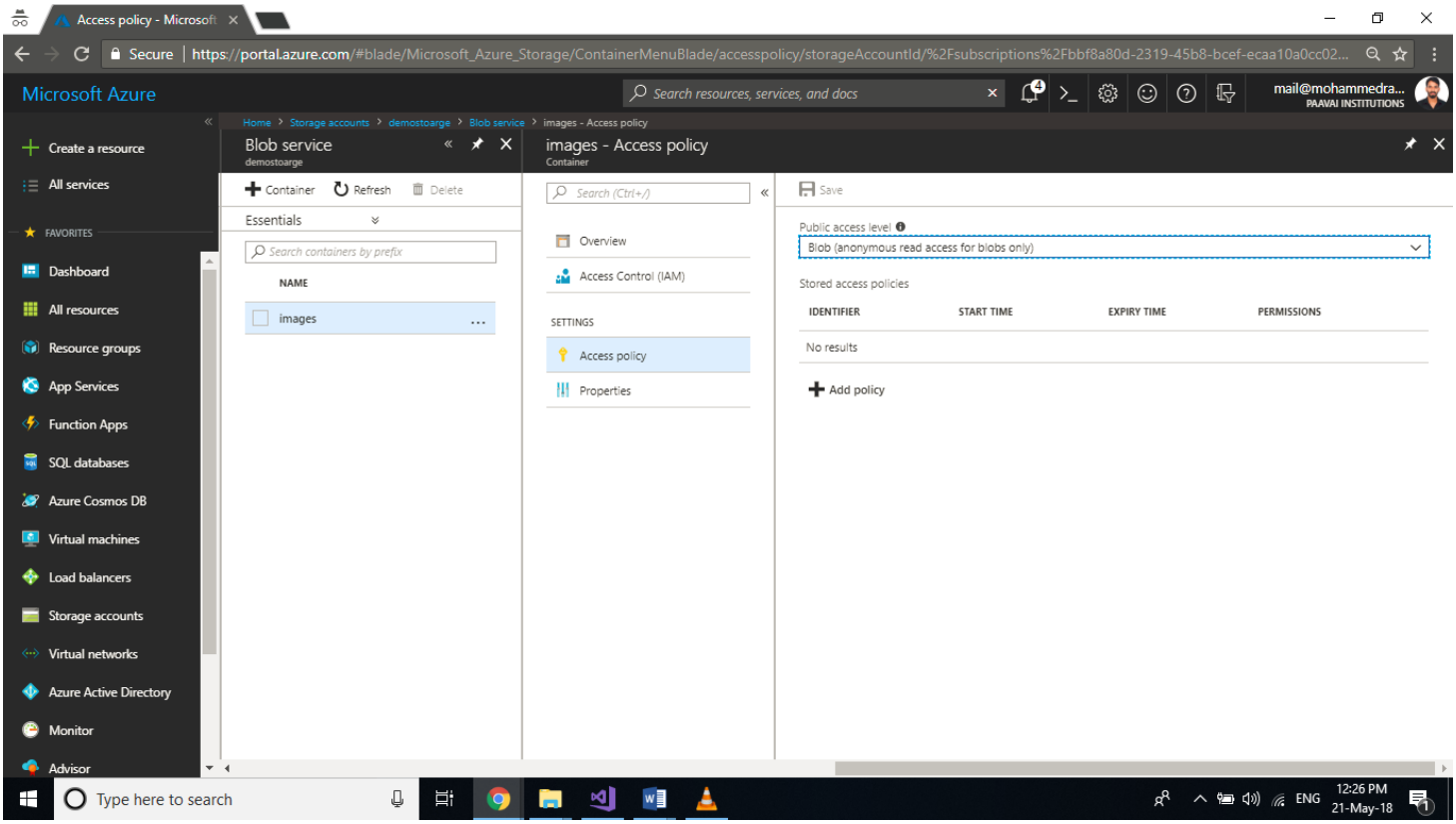


Step 7: Click on **Start** to run the program and you will see a console window waiting for an input as we given Console.Readkey() at the end of the code.



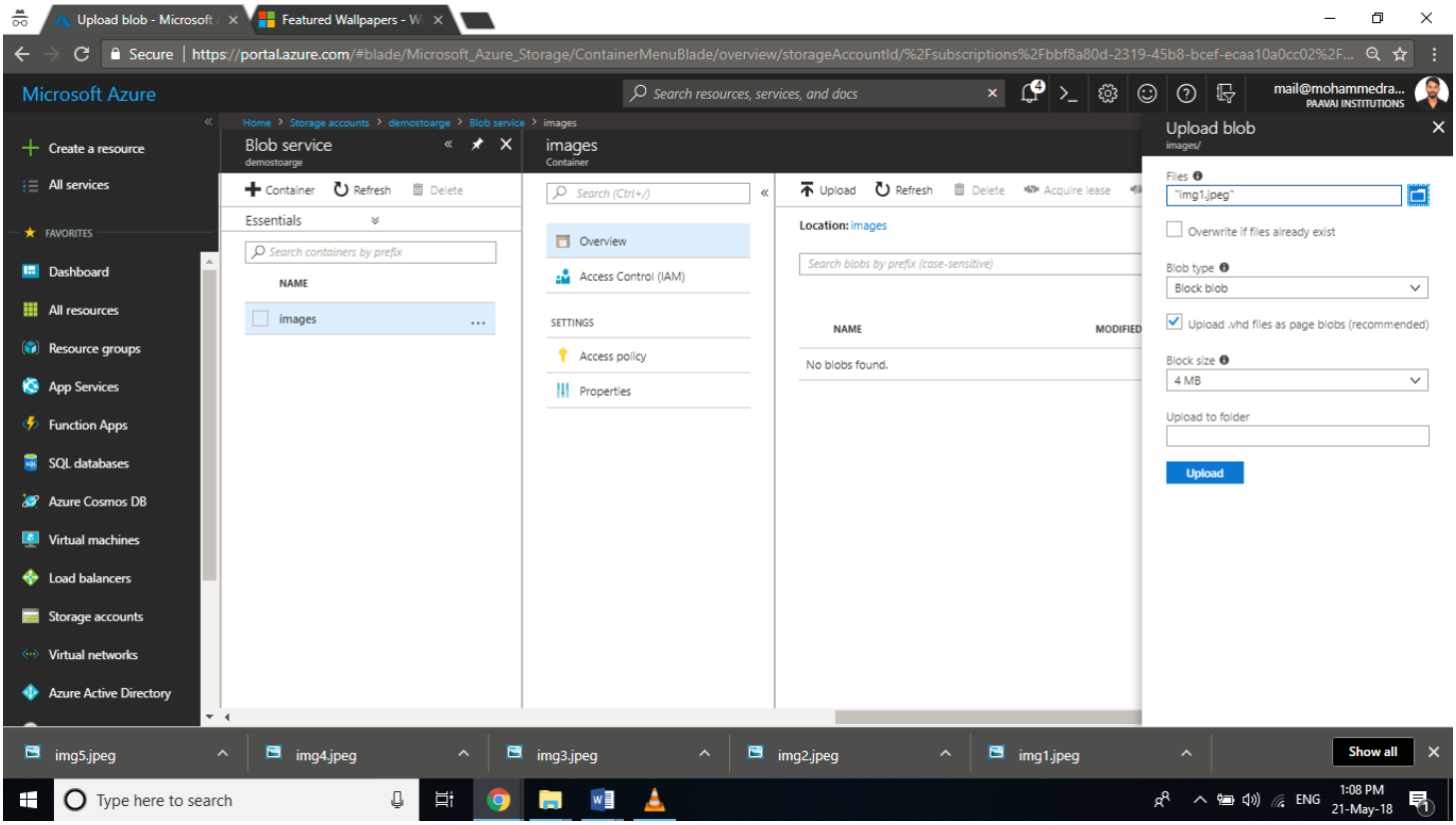
Step 8: Go back to your storage account and select **Blobs** service from the overview tab and you can see that we now have an **images** container there and we can see that the **access policy** is set to the blob for the container as we set in the code.



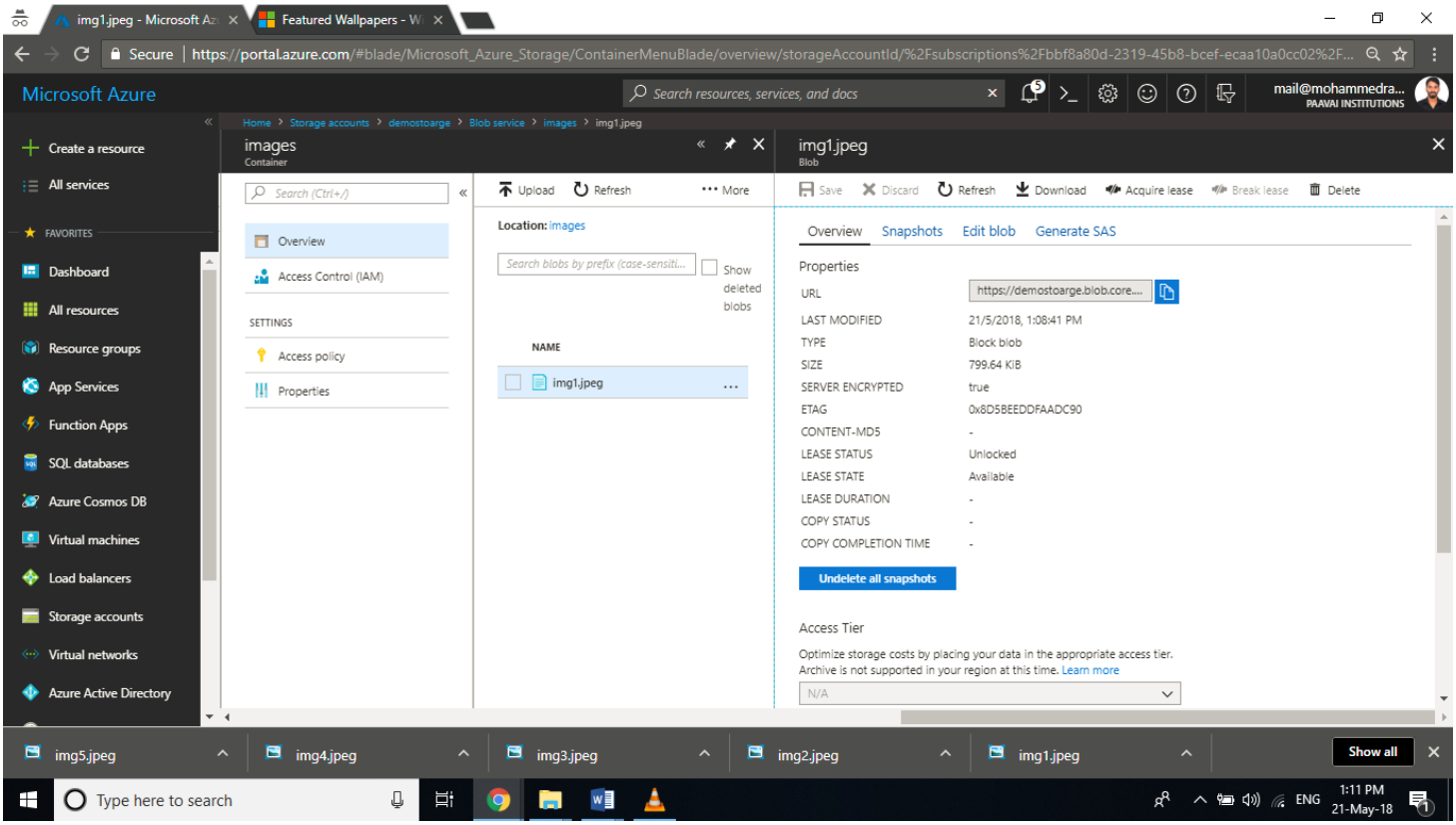


Blob Storage: Uploading, Downloading & Listing Blobs

It is very easy to upload or download data to Containers in the Blob through the portal. If you select the container you can see an option to do the same. You can click on **upload** and browse for an item from your local machine to add the blob, and in the advanced option you can select the blob type as Blob, Page or Append and specify the block size.



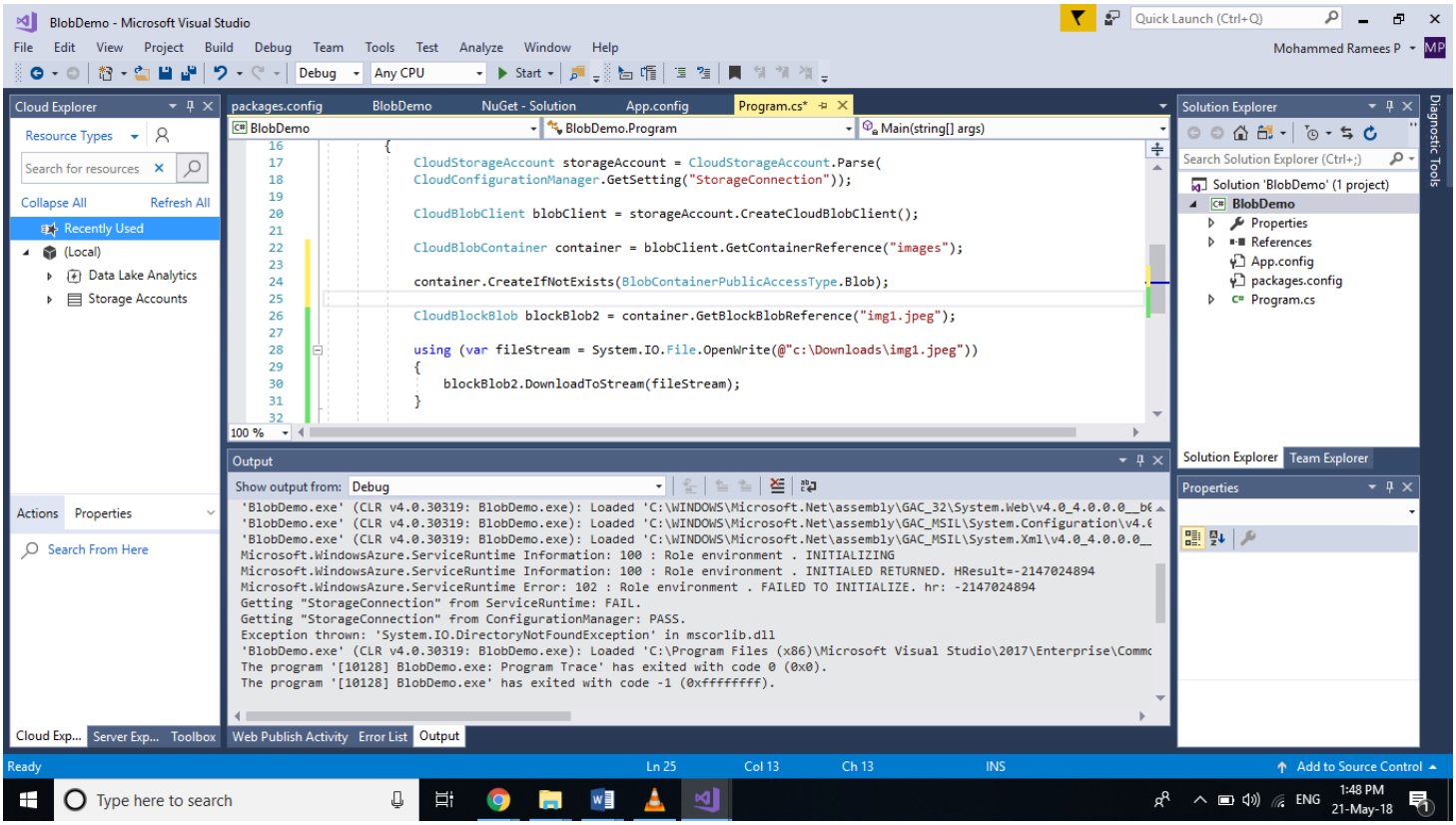
Once uploaded, the image will be listed in the container and on selecting you will be able to **download** the file through the portal and also you can get the **URL** to access the file for the public. If you scroll down you can see an option to add metadata also.



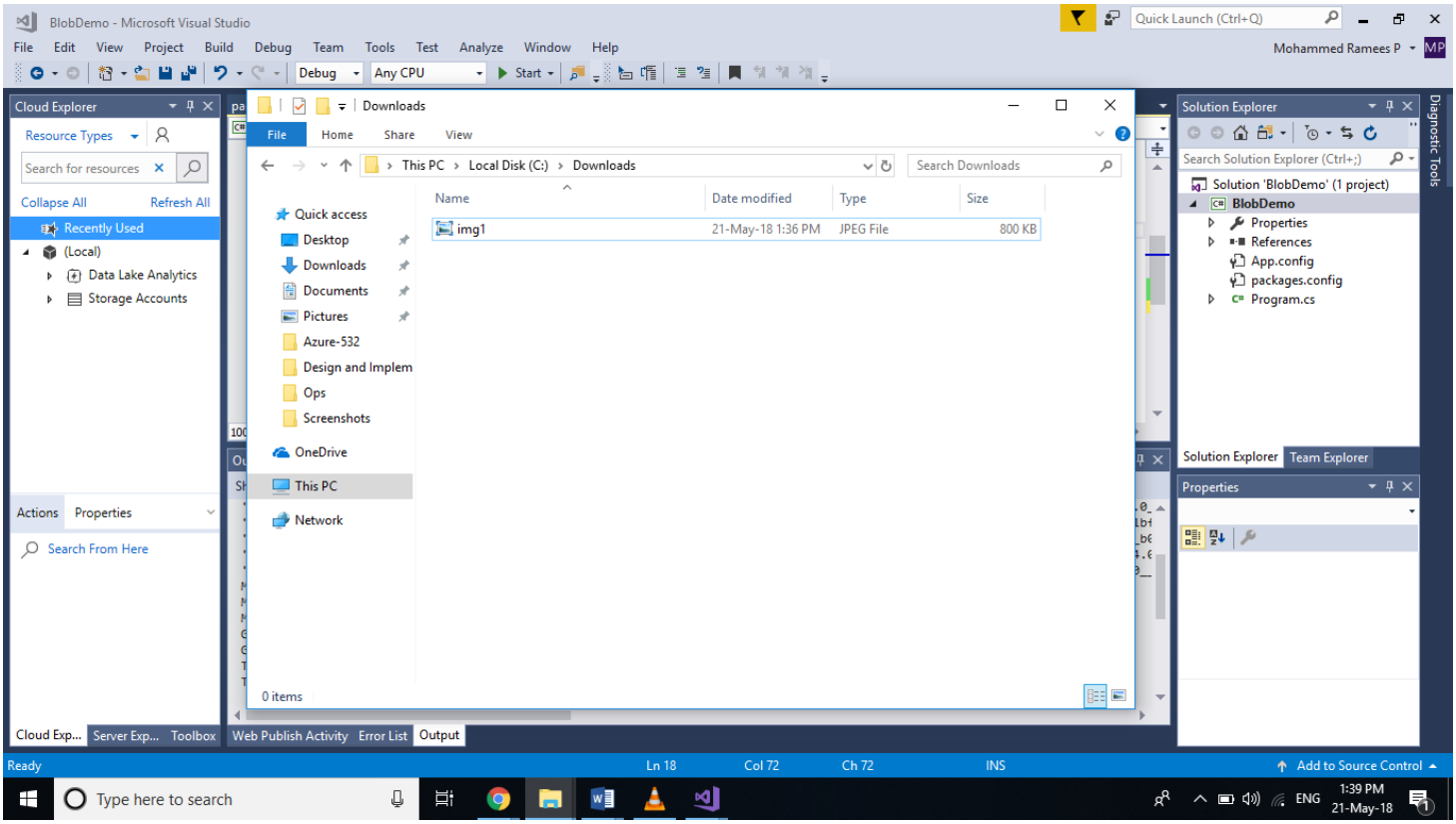
Now here in this demo, we will see how to download the existing image that we uploaded, upload a new image through a C# application and also list all the files in a container.

Step 1: In this step, we will **download the file** that we just added (Make sure you are having the image in the container) to downloads folder on our C drive. Append the following code to your previous exercise in the Main method.

```
CloudBlockBlob blockBlob2 = container.GetBlockBlobReference("img1.jpeg");

using (var fileStream = System.IO.File.OpenWrite(@"c:\Downloads\img1.jpeg"))
{
    blockBlob2.DownloadToStream(fileStream);
}
```

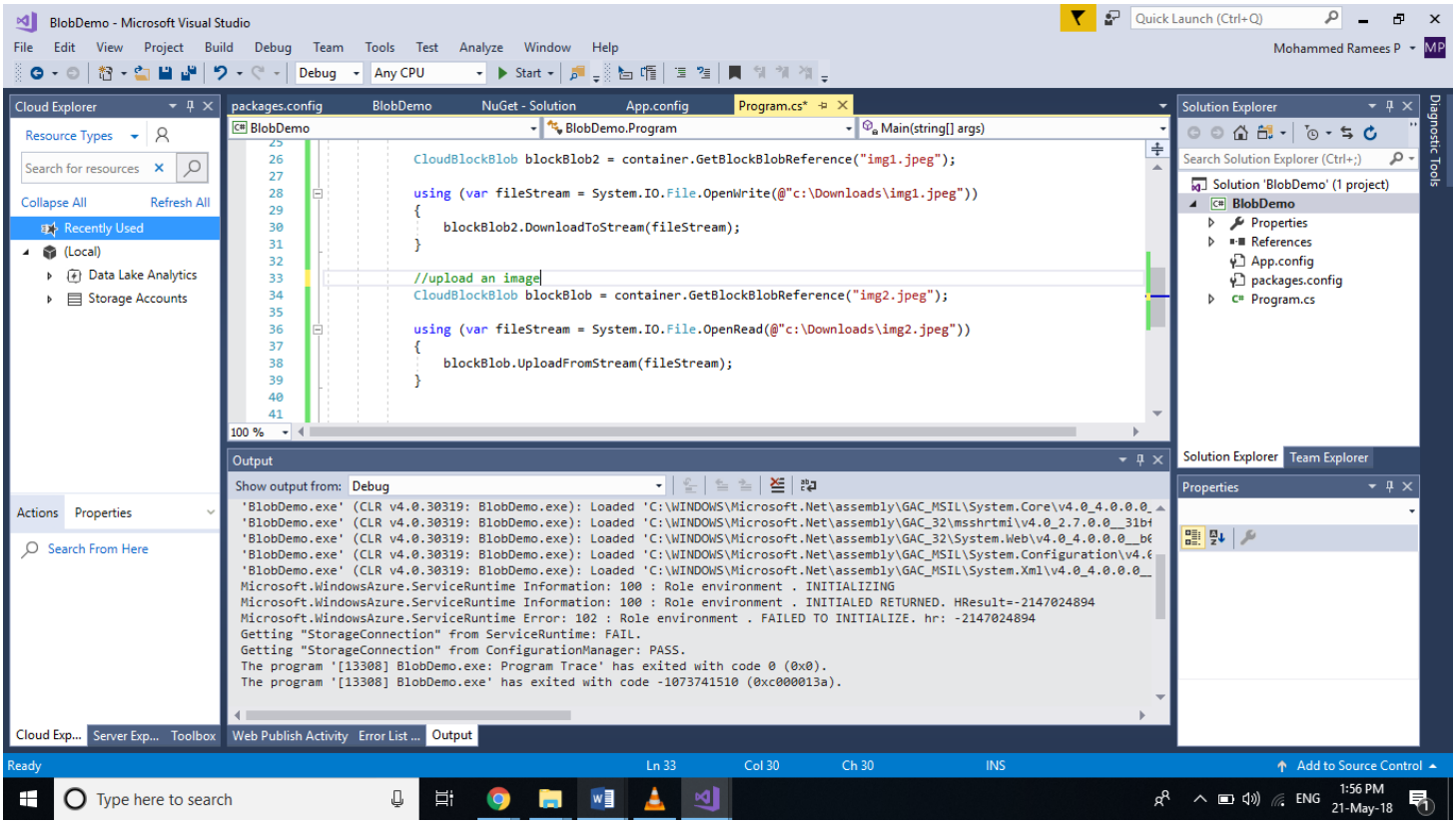


Step 2: Run your application and you can see the file downloaded in the folder specified. (Make sure you have created a Downloads folder prior to running your application, else will throw an exception).

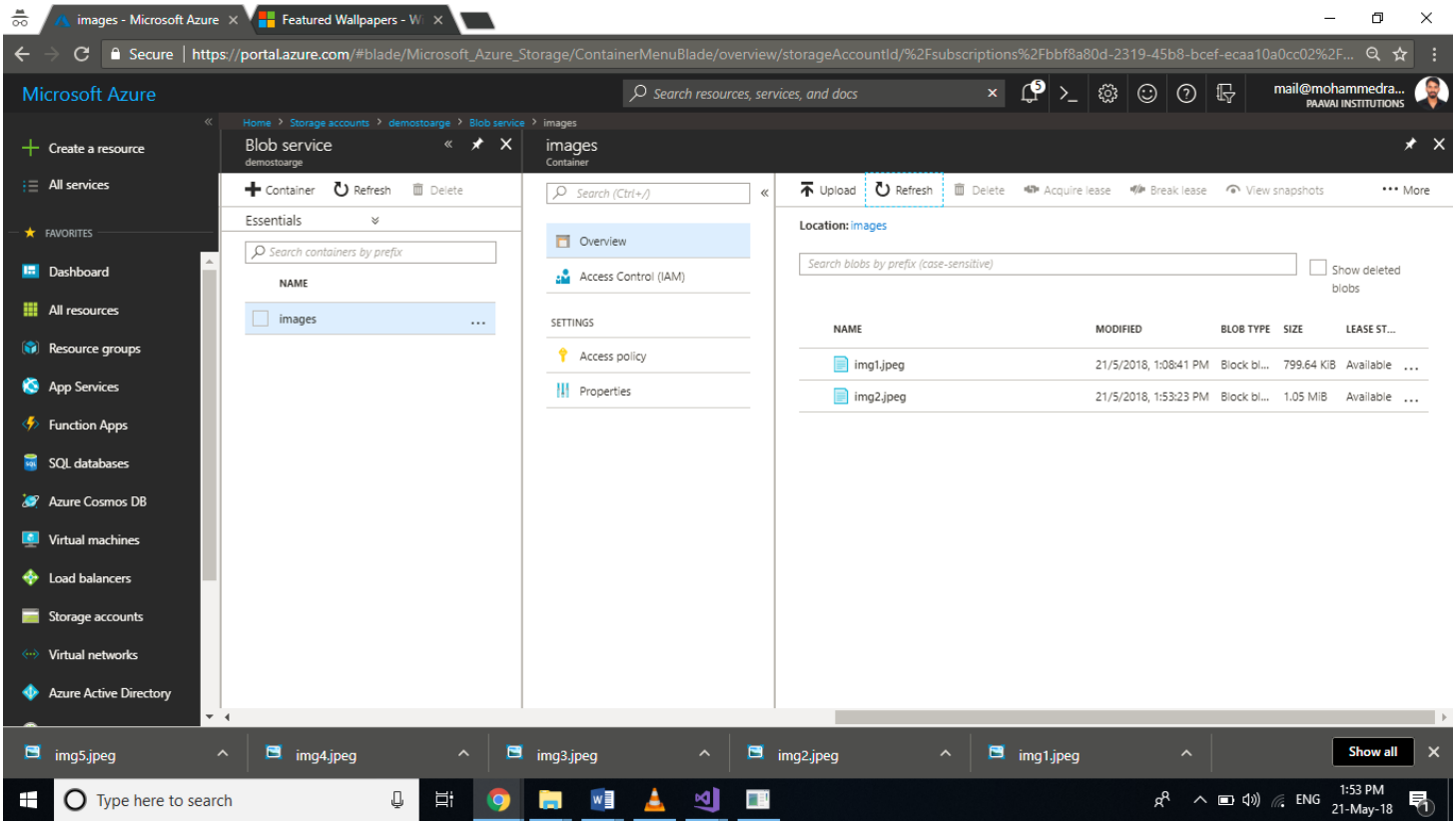


Step 3: In this step will **upload a new file** to the container from the same folder. For that, I have already added a new file named img2.jpeg to the same folder and append the following code to your main method.

```
CloudBlockBlob blockBlob = container.GetBlockBlobReference("img2.jpeg");

using (var fileStream = System.IO.File.OpenRead(@"c:\Downloads\img2.jpeg"))
{
    blockBlob.UploadFromStream(fileStream);
}
```

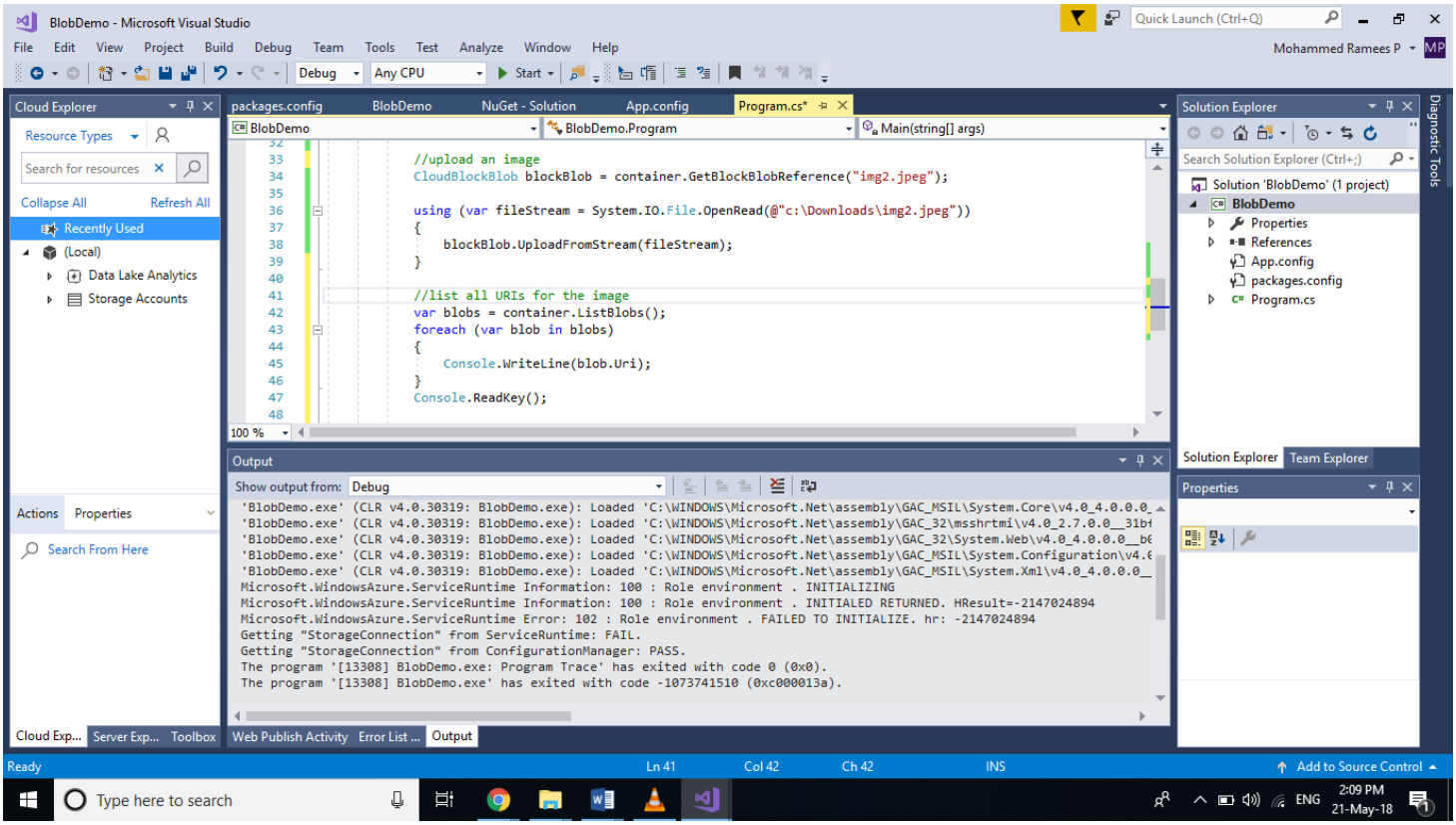


Step 4: **Run** your application and if you check the container in the portal you can see a new image added



Step 5: Now in this step we will write some code to **list all the blobs** that are up there in that container. As the files are of image type we will list the URI for the images. Append the following code to the main method.

```
var blobs = container.ListBlobs();
foreach (var blob in blobs)
{
    Console.WriteLine(blob.Uri);
}
Console.ReadKey();
```



Step 6: **Run** the application and you can see the URIs of our two images in the container listed