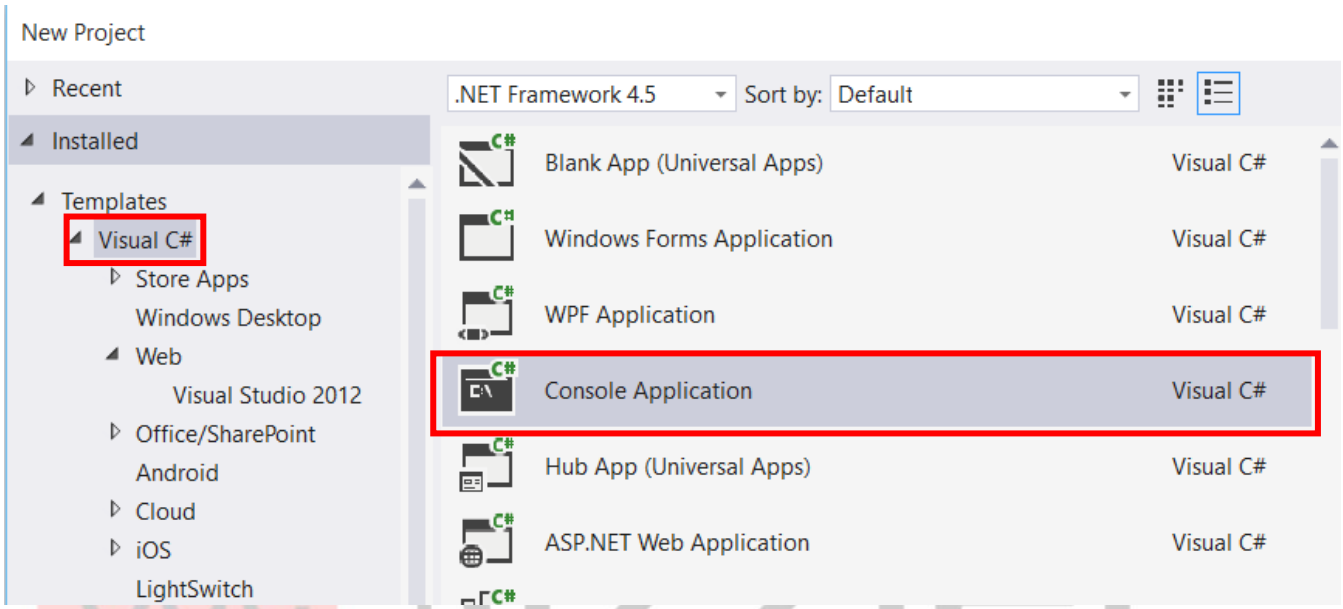
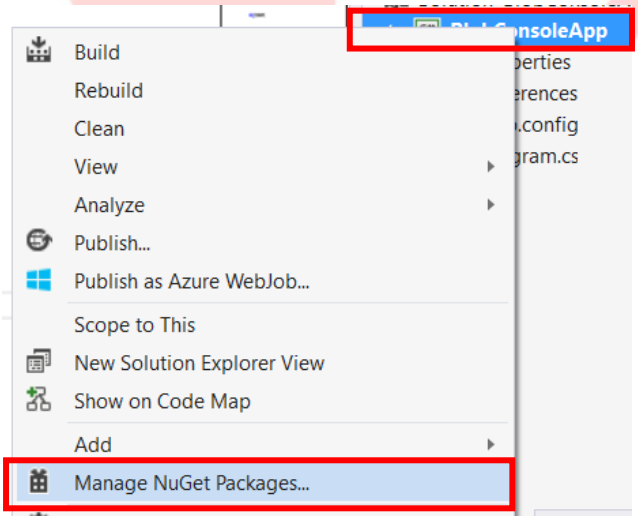
Azure Storage Blob – Console App

Step 1: Create New Console App from Visual Studio Template

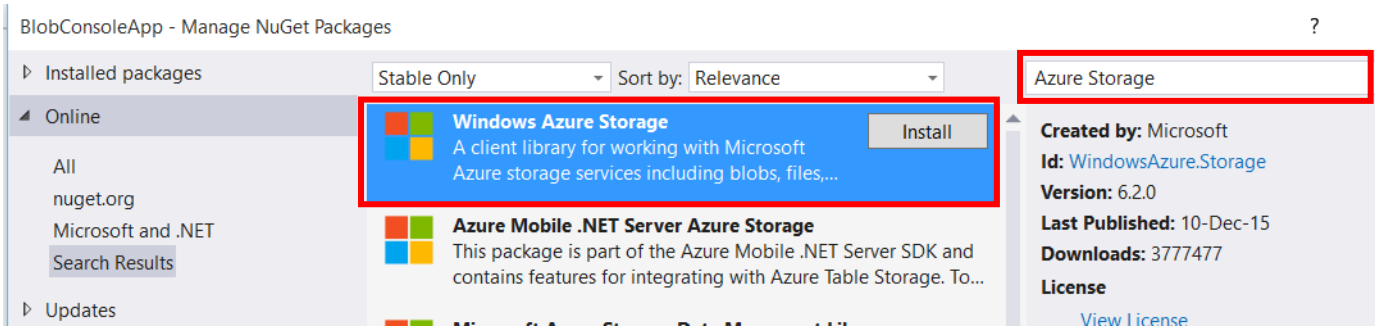


Step 2: Right click on Project name & select "Manage NuGet Packages…"



Step 3: Manage NuGet Packages dialog box will open & search for “Azure Storage”

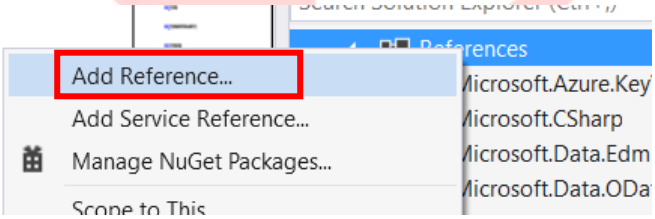
Install “Windows Azure Storage”



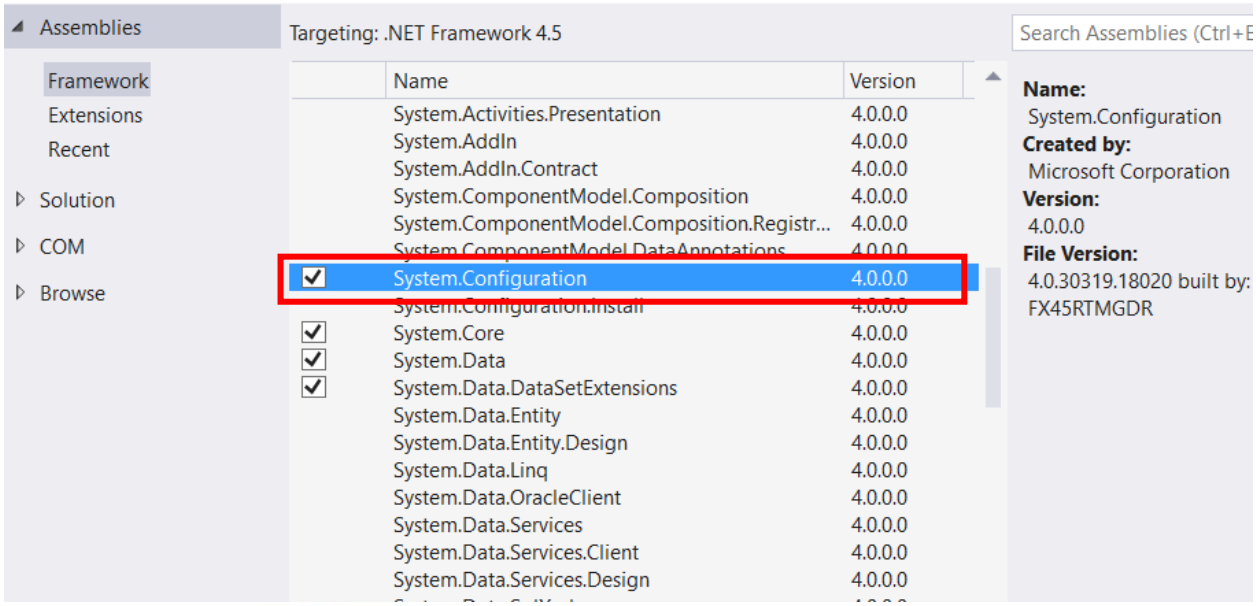
Step 4: Open app.config file, add an entry under the Configuration element, replacing the account name and key with your own storage account details:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <appSettings>
      <add key="StorageConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=storageaccountname;AccountKey=storag
ekey" />
    </appSettings>
</configuration>
```

Step 5: Add one reference



Select “System.Configuration”

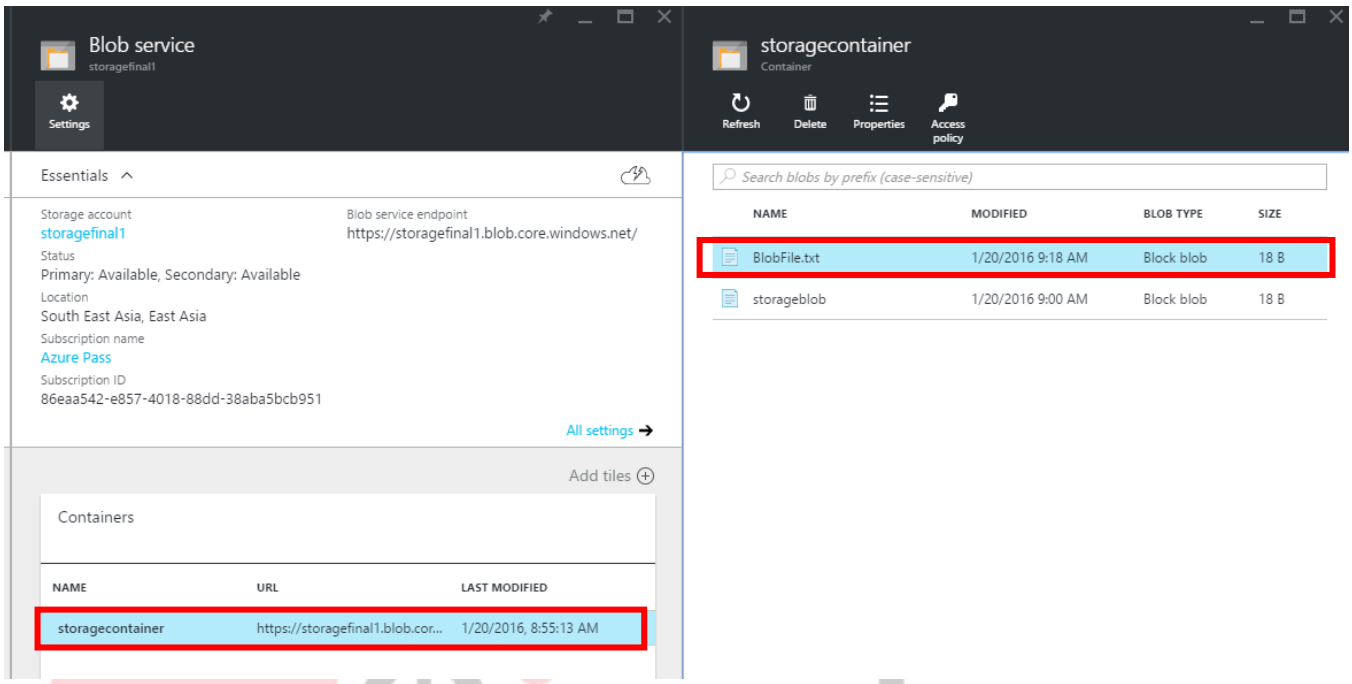


Step 6: Open Program.cs file

Add references of Azure Storage

```
using System.Configuration;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure.Storage.Blob;


static void Main(string[] args)
        {
            //Retrieve Storage account from connection string
            CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(ConfigurationManager.AppSettings["StorageConnectionString"])
;

            //Create the blob client
            CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

            //Retrieve a reference to a container
            CloudBlobContainer container =
blobClient.GetContainerReference("storagecontainer");

            //Create the container if it doesn't already exist
            container.CreateIfNotExists();

            //Retrieve reference to a blob named "storageblob"
            CloudBlockBlob blockBlob =
container.GetBlockBlobReference("BlobFile.txt");

            //Create or overwrite the "storageblob" blob with contents from a local
file
            using (var filestream = System.IO.File.OpenRead(@"C:\azure\BlobFile.txt"))
            {
                blockBlob.UploadFromStream(filestream);
            }
```

```
}
```

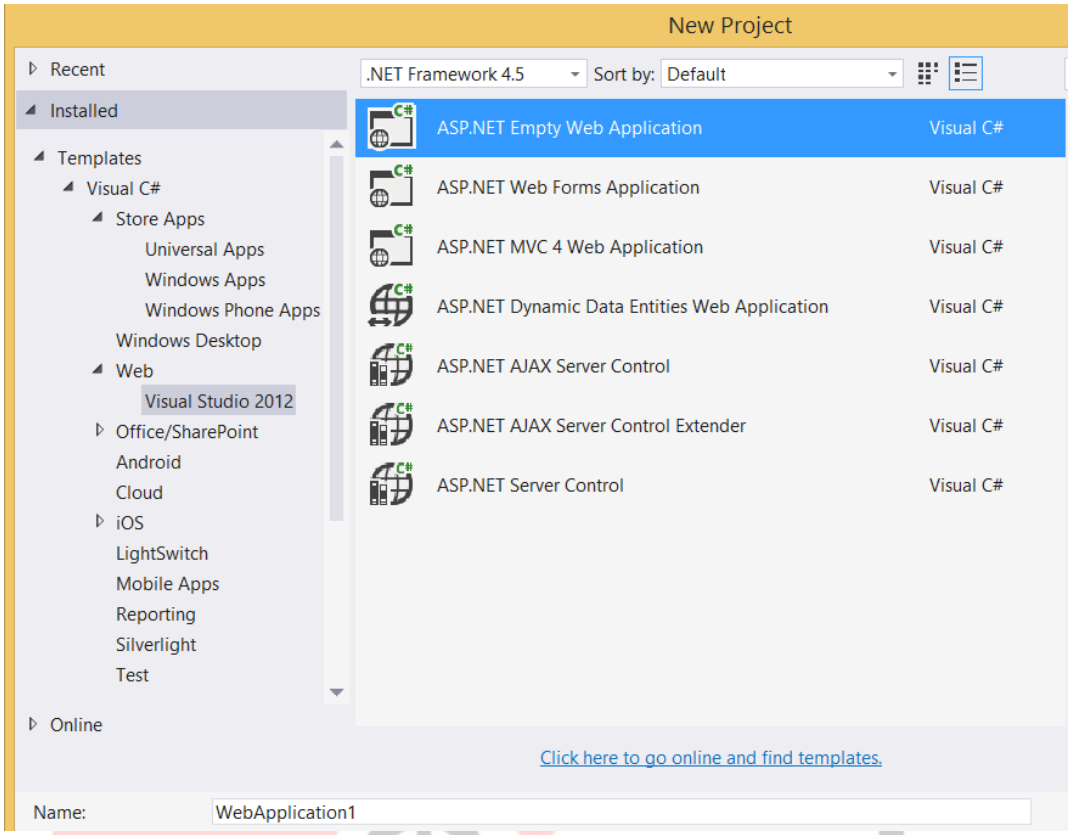


Azure Blob Storage Image Uploading:

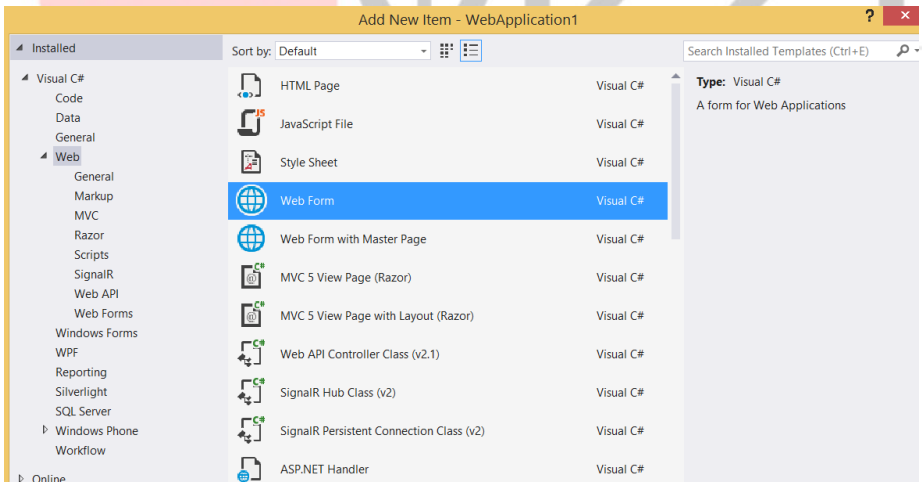
Step 1:

Create New ASP.NET Empty Web Application

Templates -> Web -> Visual Studio 2012 -> ASP.NET Empty Web Application



Step 2: Add New WebForm Page



Step 3:

Add one button & file upload control in WebForm1.aspx page also add async="true" at first line

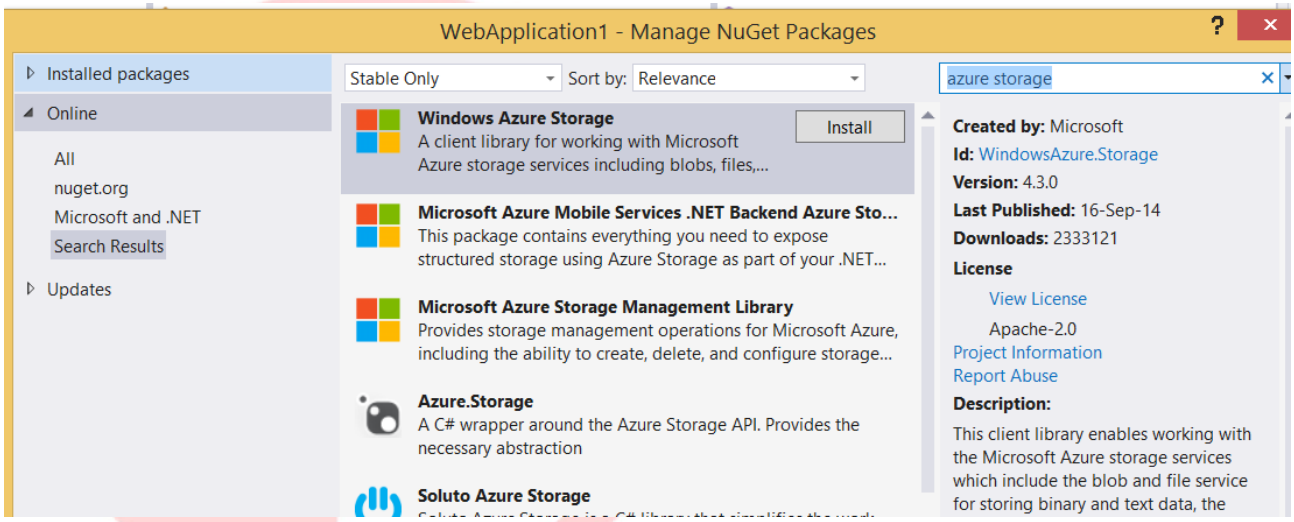
```
<body>
    <form id="form1" runat="server">
        <div>
            Photo Upload : <asp:FileUpload ID="fuTest" runat="server" />
```

```
            <br/>
            <asp:Button ID="btnUpload" Text="Upload File" runat="server"
OnClick="btnUpload_Click"/>
            <asp:Label ID="lblPhotoUpload" Text="Click on Upload File button"
runat="server" />
        </div>
    </form>
</body>
```

Step 4:

Right click on project name and select "Manage NuGet Packages…"

Search for "Azure Storage"



Install the Package.

Step 5:

Open WebForm1.aspx.cs

Add the reference of Azure storage

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

```
protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected async void btnUpload_Click(object sender, EventArgs e)
        {
            // Retrieve storage account from connection string
            CloudStorageAccount storageAccount = new CloudStorageAccount(
                            new StorageCredentials("storagename",
                                "storageaccesskey"), true);

            // Create the blob client
            CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

            // Retrieve reference to a previously created container
            CloudBlobContainer container =
blobClient.GetContainerReference("photocontainer");
            await container.CreateIfNotExistsAsync();

            CloudBlockBlob blockBlob =
container.GetBlockBlobReference(fuTest.PostedFile.FileName);

            // Create or overwrite the "myblob" blob with contents from a local file.
            using (var fileStream = fuTest.PostedFile.InputStream)
            {
                blockBlob.UploadFromStream(fileStream);
            }

            lblPhotoUpload.Text = "Photo Uploaded Successfully.";

        }
```

Now run the project

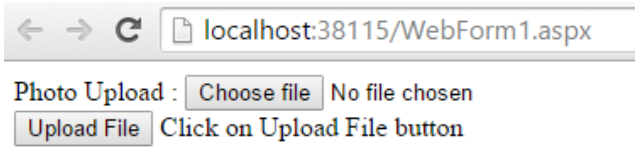
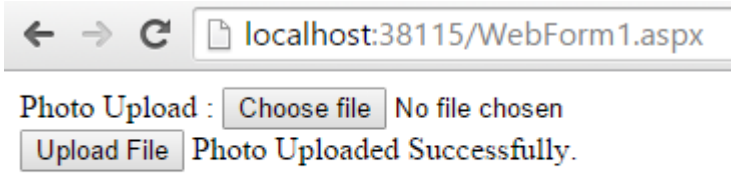
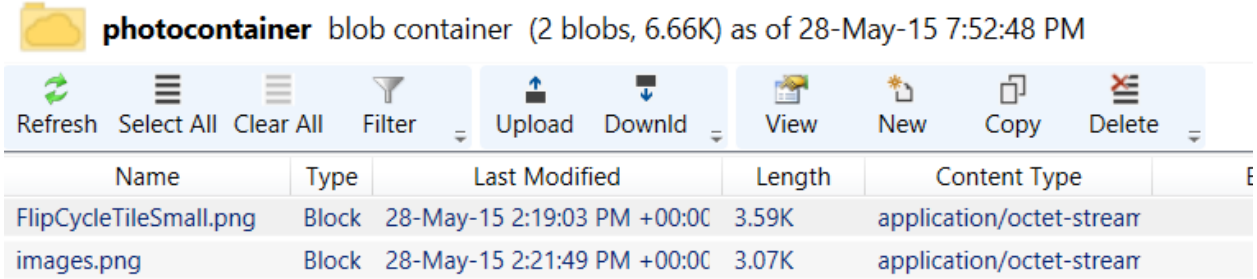


Image successfully uploaded



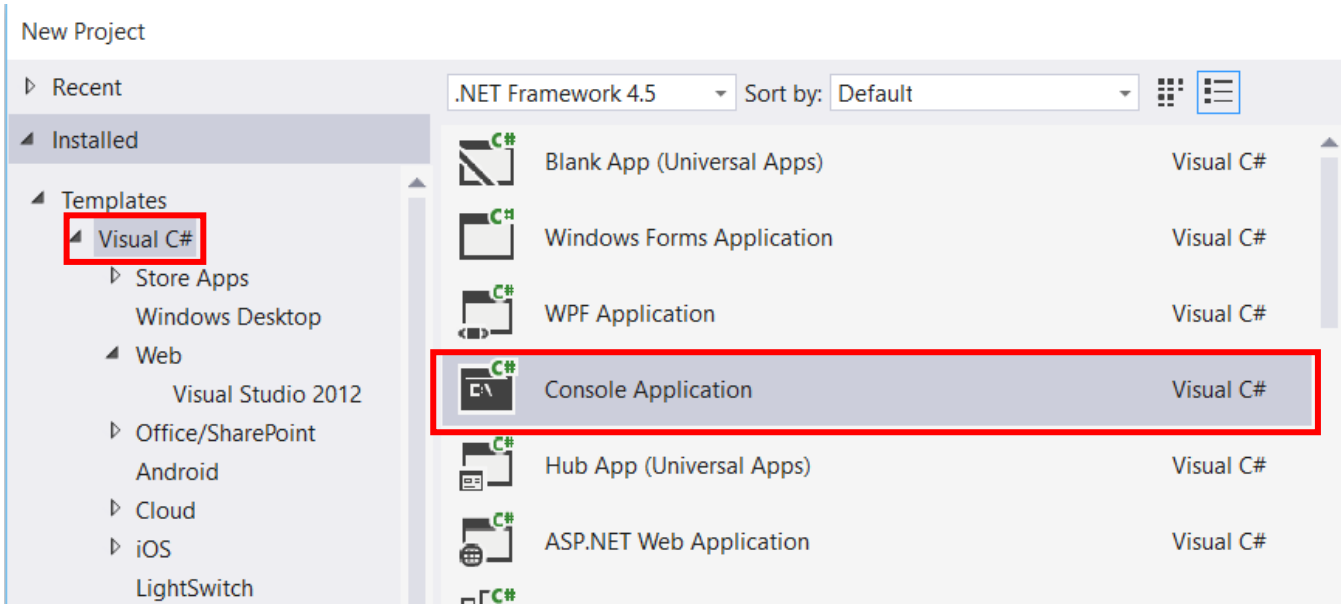
Using Azure Storage Explorer



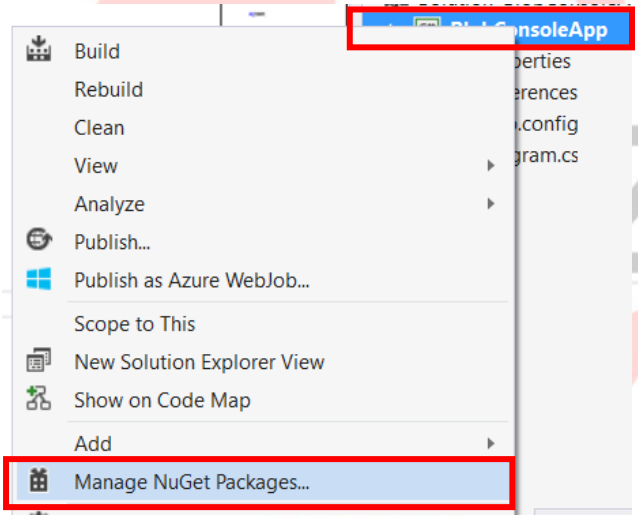
Azure Storage Queue – Console App

The Azure Storage Queue service provides a mechanism for reliable inter-application messaging to support asynchronous distributed application workflows. This section covers a few fundamental features of the Queue service for adding messages to a queue, processing those messages individually or in a batch, and scaling the service.

Step 1: Create New Console App from Visual Studio Template

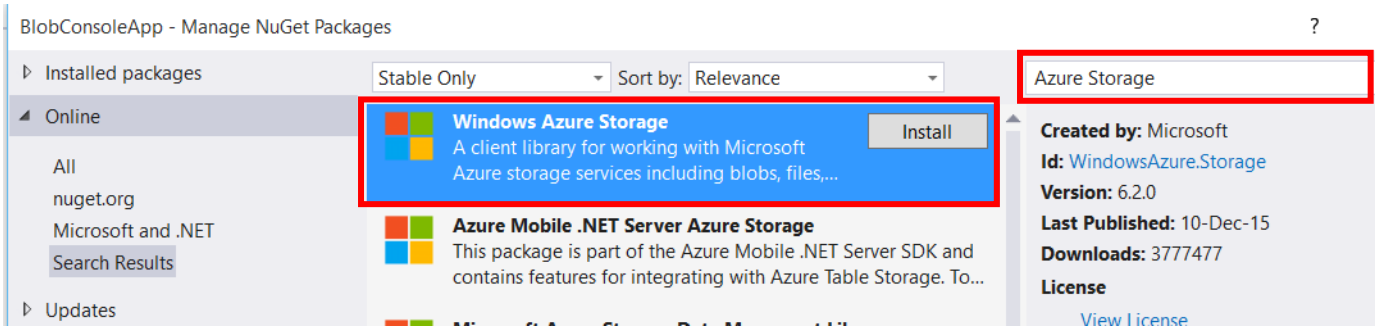


Step 2: Right click on Project name & select "Manage NuGet Packages…"



Step 3: Manage NuGet Packages dialog box will open & search for "Azure Storage"

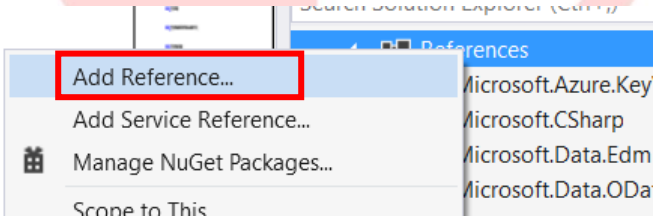
Install "Windows Azure Storage"



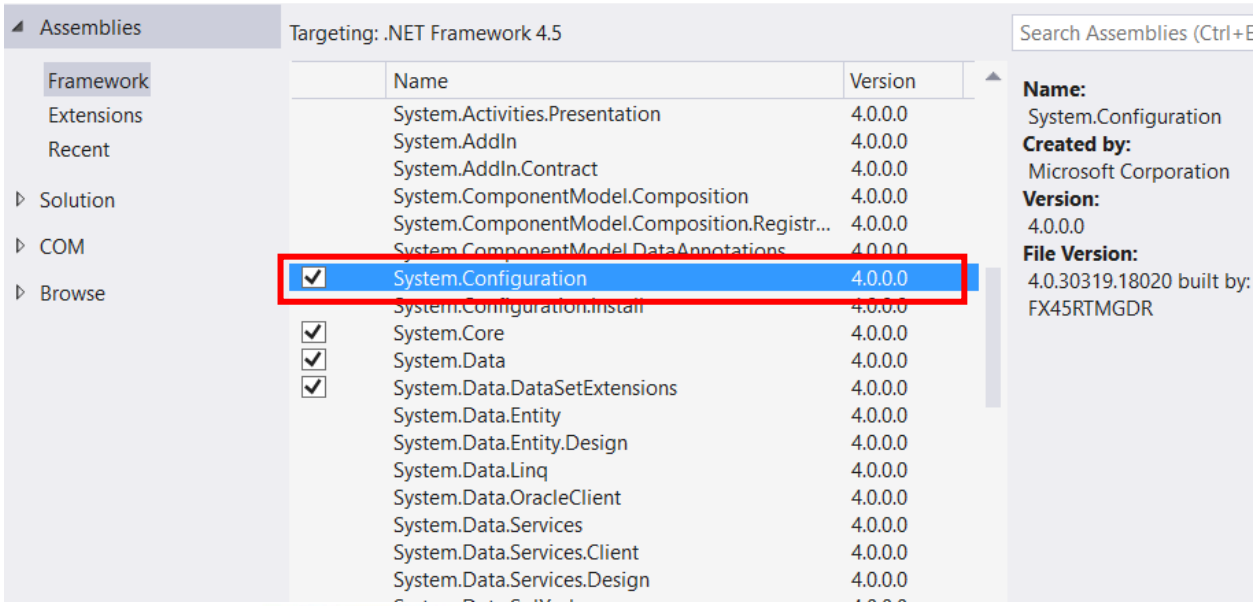
Step 4: Open app.config file, add an entry under the Configuration element, replacing the account name and key with your own storage account details:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <appSettings>
      <add key="StorageConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=storageaccountname;AccountKey=storag
ekey" />
    </appSettings>
</configuration>
```

Step 5: Add one reference



Select "System.Configuration"



Step 6: Open Program.cs file

Add references of Azure Storage

```
using System.Configuration;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure.Storage.Queue;


static void Main(string[] args)
        {
            //Retrieve Storage account from connection string
            CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(ConfigurationManager.AppSettings["StorageConnectionString"])
;

            //Get a reference to the Queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

            //Get a reference to a Queue object
            CloudQueue queue = queueClient.GetQueueReference("storagequeue");
            queue.CreateIfNotExists();

        }


Output : storagequeue will generate
```

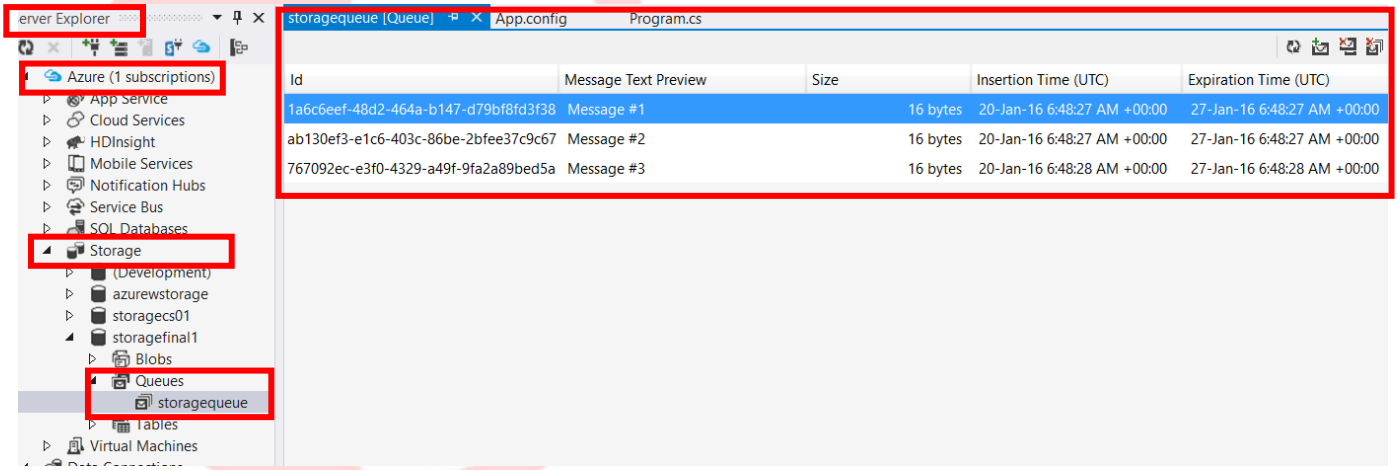
Adding messages to a queue

You can access your storage queues and add messages to a queue using many storage browsing tools; however, it is more likely you will add messages programmatically as part of your application workflow.

```
static void Main(string[] args)
        {
            //Retrieve Storage account from connection string
            CloudStorageAccount storageAccount =
CloudStorageAccount.Parse(ConfigurationManager.AppSettings["StorageConnectionString"])
;

            //Get a reference to the Queue client
            CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();

            //Get a reference to a Queue object
            CloudQueue queue = queueClient.GetQueueReference("storagequeue");
            queue.CreateIfNotExists();

            queue.AddMessage(new CloudQueueMessage("Message #1"));
            queue.AddMessage(new CloudQueueMessage("Message #2"));
            queue.AddMessage(new CloudQueueMessage("Message #3"));

        }
```

Run the project

Server Explorer -> Azure Subscription -> Storage -> StorageName -> Queues



Azure Storage Queue vs Azure Service Bus Queue

Azure Storage Queue	**Service Bus Queue**
Delivers message and works with in the Azure Services and from on-premise to azure cloud.	This works with azure messaging to publish web service or to integrate patterns.
Get message, put message, peek message,	Works with all environment
Queue	Broker Relay
200TB	80GB
7 days	Unlimited

Azure Table - Azure SQL Database – DocumentDB

	Azure Table	**SQL Database**	**DocumentDB**
Storage	Non-Relational	Relational	Non-Relational
	Entity (record) – XML	Entity	Documents – JSON object
	Unlimited	500GB – single db	250GB- per collection
Indexing	Supported	Supported	Supported
	Single Index on it partition key	Define index with multiple columns	Automatically generated indexes
Server – Side Programming	Not Supported	Supported	Supported
		T-SQL	T-SQL