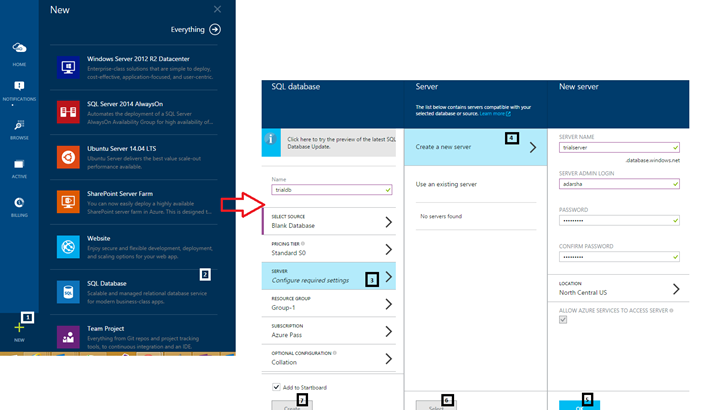
**Creating and Managing Azure SQL:**

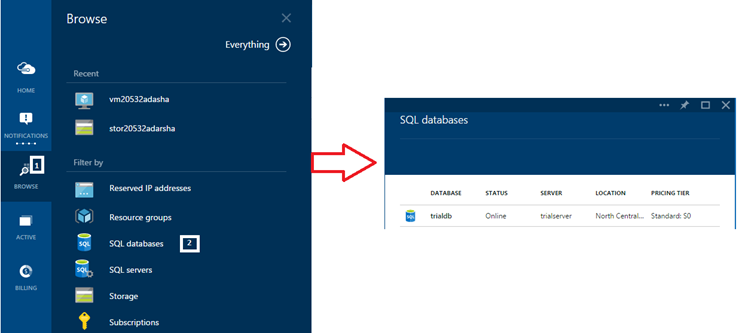
Step 1: Create and access Azure SQL using Management Studio

1. Login to the portal.azure.com
2. Select New-> SQL Database



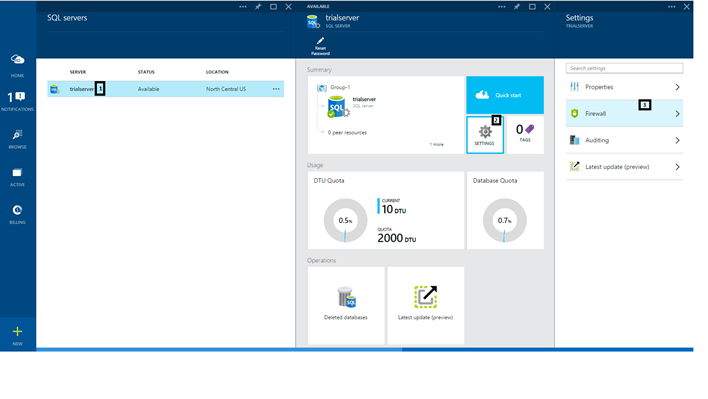
1. Fill in the details:
   1. Create a new server or use an existing one
   2. Select Pricing Tier
   3. Select Resource Group, if you would want to collate all your resources together
   4. Click on Create

Once created, the SQL database can be accessed as follows:

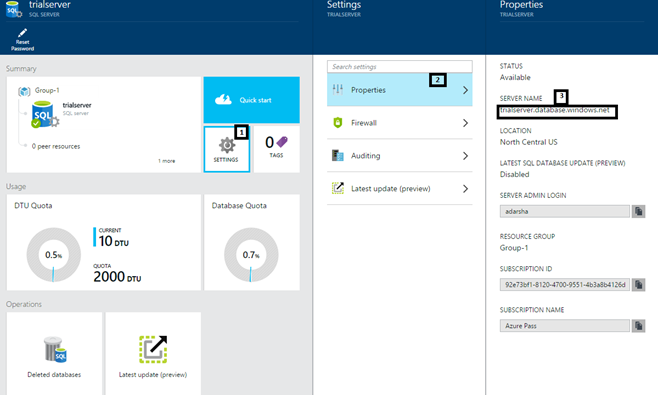


Step 2: Using SQL Management Studio, let us connect to the server

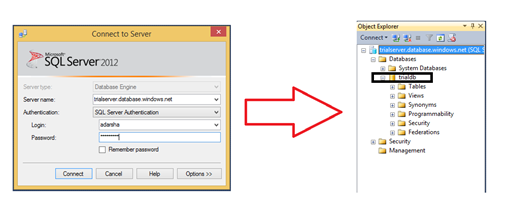
1. Give Firewall access to database.
   1. Click on Server -> Settings-> Firewall. Enter range of acceptable IP



* 1. Once Firewall access is given, note down the server name for us to access from Management Studio:



1. Login to Management Studio and use the Server Name from above to connect and use the Azure SQL. You will be able to see the created DB on the server that you just created



1. Write a simple piece of code to create an additional database from Management Studio and get back to the Azure portal to check if it added the database



**Connecting Azure SQL to WebApp**

Now that we have created Azure SQL Storage Server, Database. Let us create a simple ASP.Net Webapp and connect it to the Azure SQL DB. In order to do this, there are essentially 2 steps:

Step 1: Create the DB/table on Azure SQL using any of the utilities such as Management Studio or sqlcmd

Step 2: Create a simple Webapp to connect and read from the Azure SQL table

Step 1:

Let us create Azure SQL DBs and tables using the sql command utility:

C:\Users\addatta>sqlcmd –U <admin>@<storage server name>

-P <password> -S <storage server name> -d master

1> CREATE DATABASE TRIALDB

2> GO

1> CREATE DATABASE TestDb;

2> GO

1> quit

C:\Users\addatta>sqlcmd -U <admin>@<storage server name>

-P <password> -S <server name> -d TestDb

1> CREATE TABLE Table1 (Col1 int primary key, Col2 varchar(20));

2> GO

1> INSERT INTO Table1 (Col1, Col2) VALUES (1, 'string1'), (2, 'string2');

2> GO

(2 rows affected)

1> Quit

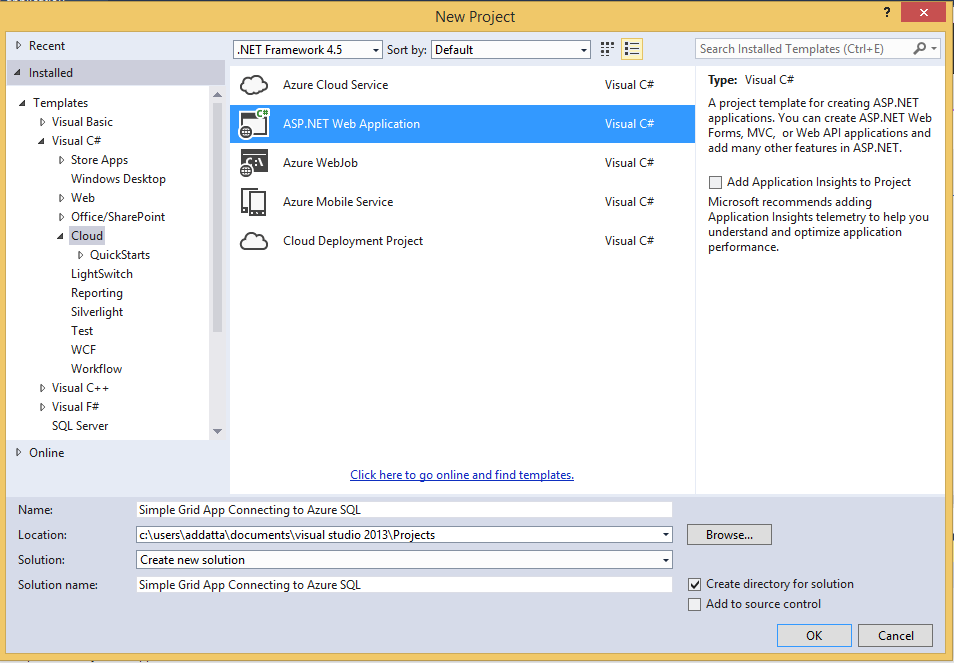
C:\Users\addatta>

Step 2:

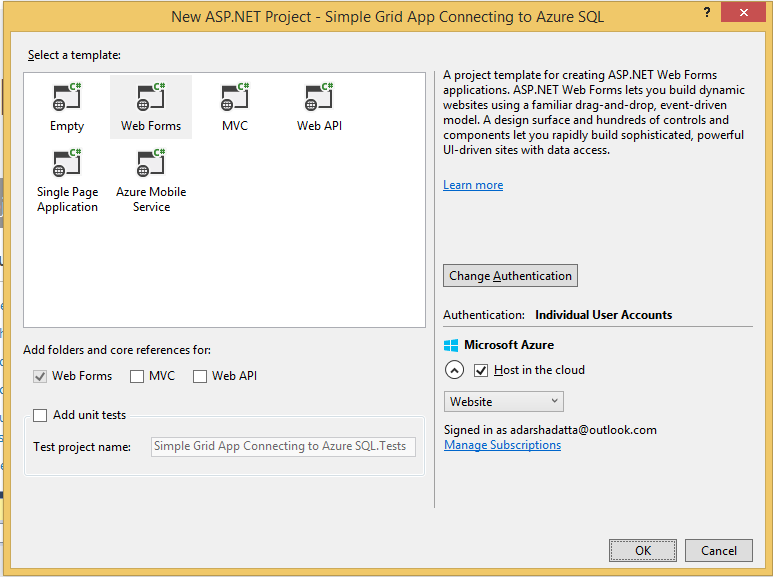
Visual Studio integrates completely with Azure.

1. Create a new Web App template:

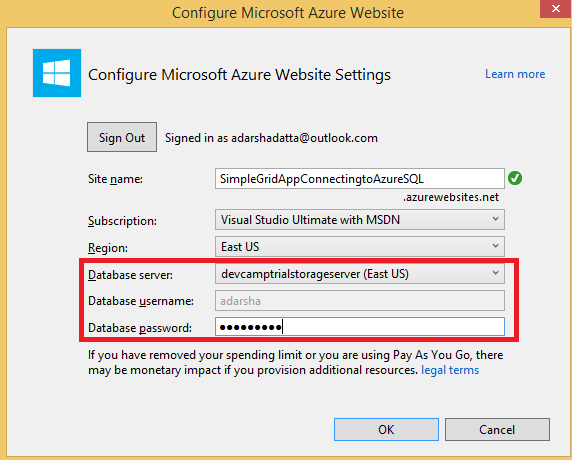
New Project -> Installed Templates -> Cloud -> ASP.Net Web Application, enter name, click on ‘ok’.



1. Select ‘Web Forms’, check ‘Host in Cloud’ and click ‘OK’



1. If you have not signed in to your Azure account, you will be prompted to enter your Azure credentials. Once you have entered your Azure credentials, your app is connected to your Azure account. Select your Azure SQL server, enter admin username and password and click ‘OK’



1. Once the project is created. Double click on Default.aspx. Replace the first <div>:

<div class="jumbotron">

<h1>ASP.NET</h1>

<p class="lead">ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS, and JavaScript.</p>

<p><a href="http://www.asp.net" class="btn btn-primary btn-lg">Learn more &raquo;</a></p>

</div>

With:

<div class="jumbotron">

<h1>Connect Grid View with Azure SQL</h1>

</div>

In the following section of <div class =row>, delete the entire content to ensure that there are no columns. The entire Default.aspx should look like this:

<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Simple\_Grid\_App\_Connecting\_to\_Azure\_SQL.\_Default" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<div class="jumbotron">

<h1>Connect Grid View with Azure SQL</h1>

</div>

<div class="row">

</div>

</asp:Content>

1. Click on Design View. From ToolBox, select GridView and drag it to the <div class =”row”> section. In the Properties window for the GridView, set AutoGenerateDeleteButton and AutoGenerateEditButton to true.
2. In Default.aspx, Source View: Add the following markup inside the <asp:Content> tag to create a SqlDataSource control for your connection: (Note: Remember to replace the Connection String)

<asp:SqlDataSource ID="SqlAzureDataSource" runat="server"

ConnectionString="<%$ ConnectionStrings:SQLAzureConnection %>"

InsertCommand="INSERT INTO [Table1] ([Col1], [Col2]) VALUES (@Col1, @Col2)"

SelectCommand="SELECT \* FROM [Table1]"

UpdateCommand="UPDATE [Table1] SET [Col2] = @Col2 WHERE [Col1] = @Col1"

DeleteCommand="DELETE FROM [Table1] WHERE [Col1] = @Col1">

<UpdateParameters>

<asp:Parameter Name="Col2" Type="String" />

<asp:Parameter Name="Col1" Type="Int32" />

</UpdateParameters>

<InsertParameters>

<asp:formParameter Name="Col1" FormField="TextBox1" />

<asp:formParameter Name="Col2" FormField="TextBox2" />

</InsertParameters>

<DeleteParameters>

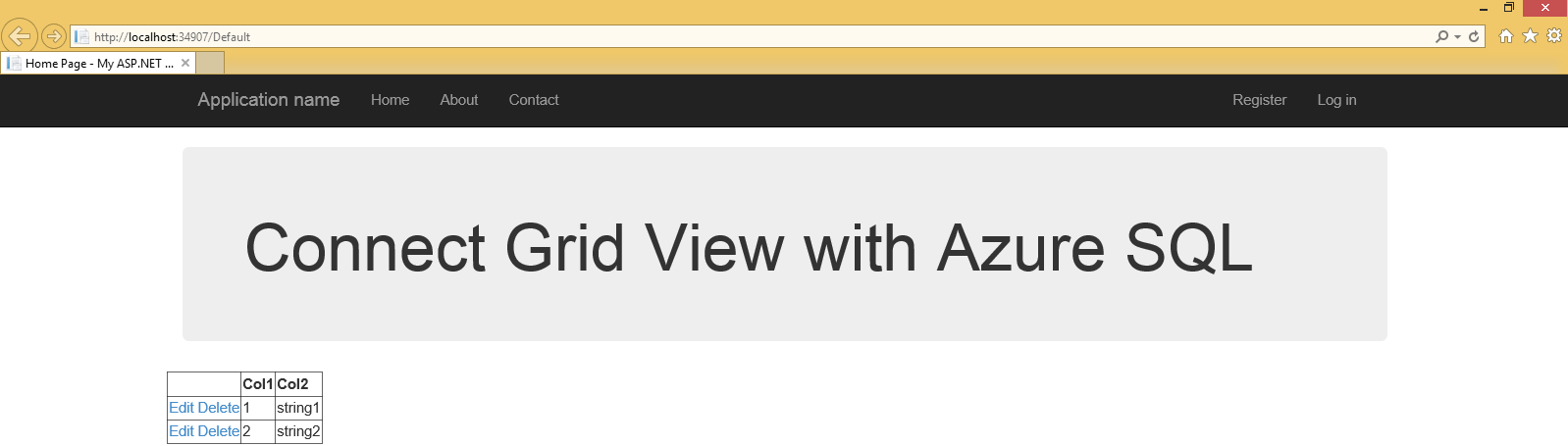
<asp:Parameter Name="Col1" Type="Int32" />

</DeleteParameters>

</asp:SqlDataSource>

1. Add the following attributes to the markup of your GridView control in Default.aspx: DataSourceID="SqlAzureDataSource"DataKeyNames="Col1".
2. Press F5 to run the solution. The browser opens and shows the contents of the table, and enables the user to edit and delete the values in the table.

This is the output of the solution:



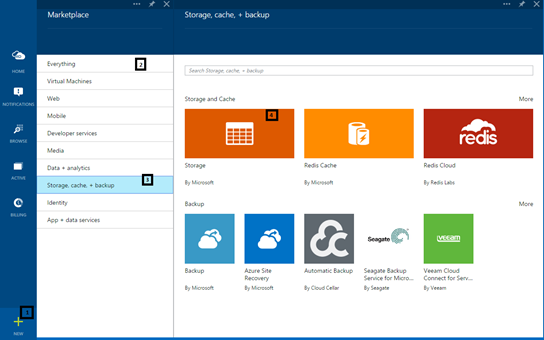
You should be able to see the small table with the 2 string data that we created in Step 1.

**Storage Containers: Blob & Queues:**

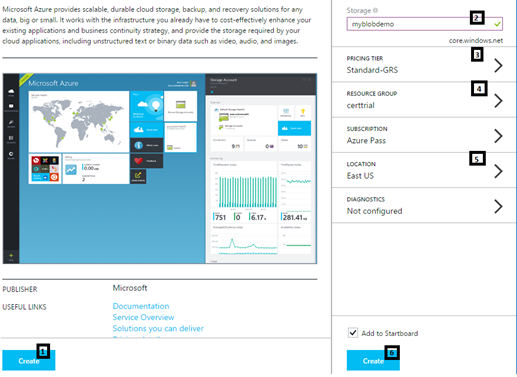
In this section, we will create a storage container and then a blob and a queue to show to use it.

1. Create the Storage Container:

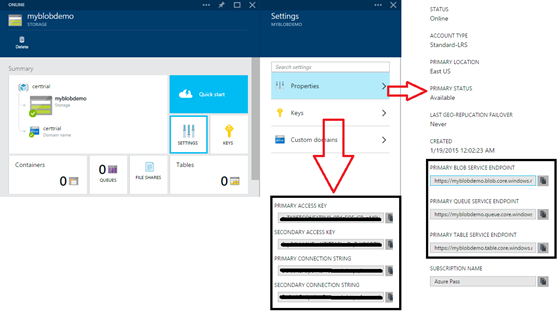
Login to the Azure management portal. Click on New -> Everything-> Storage cache, backup -> Storage.



Enter the details and click on Create.



1. Once the storage account is created, you can look into the settings tab and note the access points of each storage containers as well as the access keys as shown:



1. Let us open Visual Studio and play around with these storage containers and find out the different kind of options that we can play with. We will be starting with a blank console app, create an object to access the storage account and then create a container for each of the different kind of storage.
   1. Open Visual Studio -> New Project ->Visual C# -> Console App. Install Windows Azure Storage Nuget Package and add the following using statements:

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Auth;

* 1. Create an object to access the storage account

CloudStorageAccount account = new CloudStorageAccount(new StorageCredentials("<storage ac name", "<primary access key"), true);

* 1. Blob

We will be using the following code to access the blob container:

//Create a blob container within the storage account that is created

var blobclient = account.CreateCloudBlobClient();

//Get hte container reference which is created in order to store the image in the container

var container = blobclient.GetContainerReference("images");

//This creates the blob if it does not already exist.

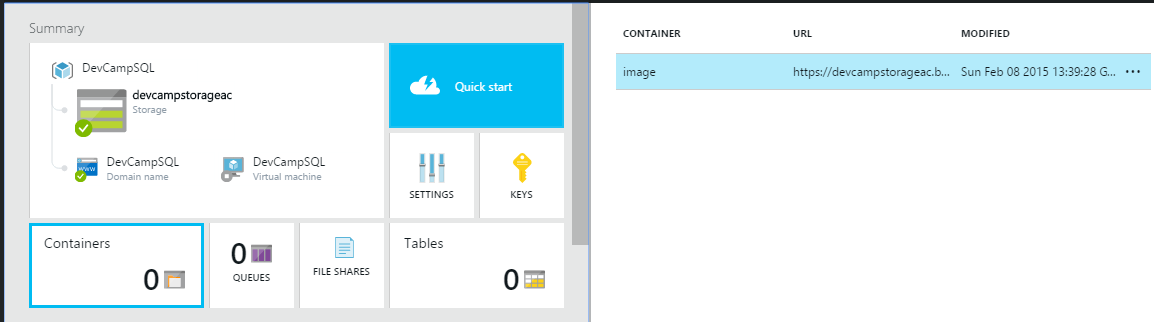
container.CreateIfNotExists();

//Get the blob reference and then upload the image to the continer using the blob reference

var blob = container.GetBlockBlobReference("diagrams.png");

blob.UploadFromFile(@"C:\Users\addatta\Desktop\blob.jpeg", FileMode.Open);

Console.Read();

* 1. After you have executed the above code, you will be able to see the blob updated on the Azure portal.
  2. Queue:

You can add the following using statement:

using Microsoft.WindowsAzure.Storage.Queue;

* 1. Instead of creating a blobclient, we will create a queue client as above and access as above:

var queueclient = account.CreateCloudQueueClient();

var queue = queueclient.GetQueueReference("myqueue");

queue.CreateIfNotExists();

* 1. Now let us conduct some operations on the queue:

queue.AddMessage(

new CloudQueueMessage( "Message 1")

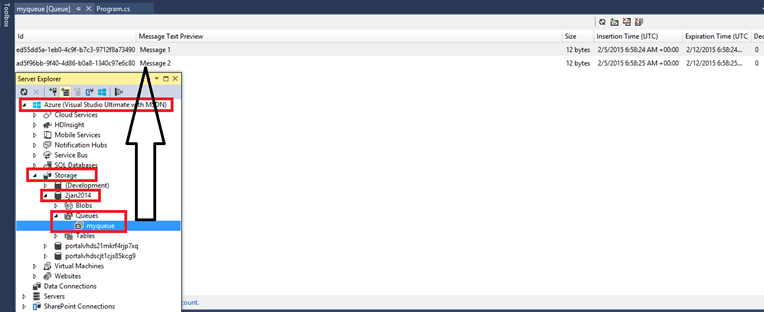
);

queue.AddMessage(

new CloudQueueMessage( "Message 2")

);

* 1. These messages can be viewed in the Server explorer:



* 1. You can consume the messages using the following:

var msg = queue.GetMessage();

Console.WriteLine(msg.AsString);

Console.ReadLine();

* 1. You can delete the messages using:

var msg = queue.GetMessage();

queue.DeleteMessage(msg);

console.WriteLine(msg.AsString);