

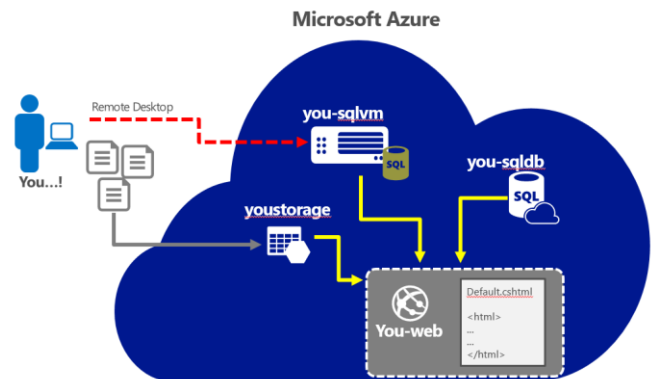
# CDP-H209

## Intro to App Dev on Azure – the Basics

### 1. What you will build...

A web application that displays a list of movies using HTML5 and CSS. Your app will display images and read/display data from a database.

1. You will create a storage account, a SQL Server Virtual Machine and a web site for your application.
2. You will upload content to your storage account
3. Develop a web application on Azure and connect your web application to the STORAGE account (for your web images).
- 4.
5. You will create a database in your Virtual Machine, configure the VM for access and modify your static web application to read/display data from the database.
6. You will migrate your database to Azure SQLDB and configure your web application to use SQLDB.



There are a number of exercises in this lab. If you do all the exercises, the lab will take you around 90 minutes.

### 2. Login to the Azure Management Portal

The first task is to get you signed into the Azure Management Portal – and to do that you need a valid subscription to Azure. You will be given an Azure subscription for this lab – PLEASE USE THE SUPPLIED SUBSCRIPTION.

On your lab computer, fire up Internet Explorer and browse to <http://manage.windowsazure.com> and login using the supplied subscription ID and password.

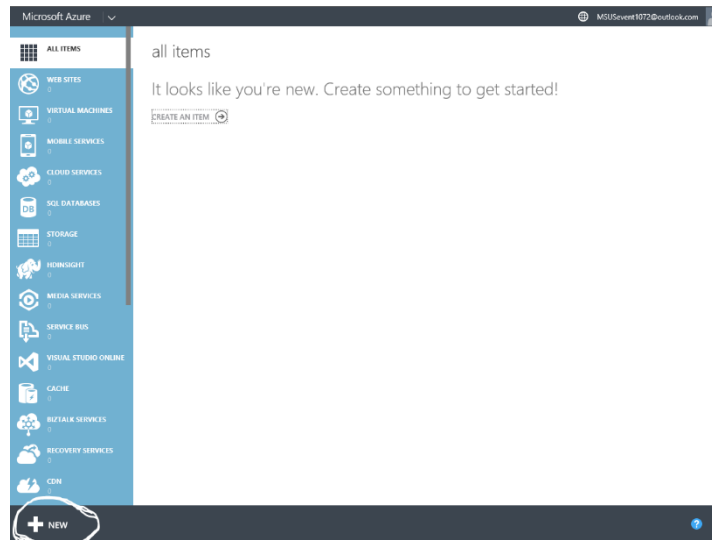
### 3. <sup>1.</sup> The Azure Portal and Azure Services

#### 3.1 The Management Portal and Your First Service

After **login**, you might get the Azure getting started tour if this is the first time you have been to the portal using this subscription. **Click through this tour** and you will see the Management Portal. This is the **CURRENT** production portal, there is also a **new preview portal** in development – you will use that later.

The various Azure Services are in the left nav bar – you click these to see the instances you have of these services, you click the “+ NEW” link to create new services.

2.



You don't have any services... Let's create one...

3.

Click the “ + NEW ” icon at the bottom left and select **DATA SERVICES** and **STORAGE** and **QUICK CREATE**

4.

In the **URL** box, enter a name for your storage service... use <youralias>store...

**For example**, if your name is Ann Green, your work email alias is [agreen@contoso.com](mailto:agreen@contoso.com), use agreenstore as the storage account name (there can be NO UPPERCASE letters or symbols).

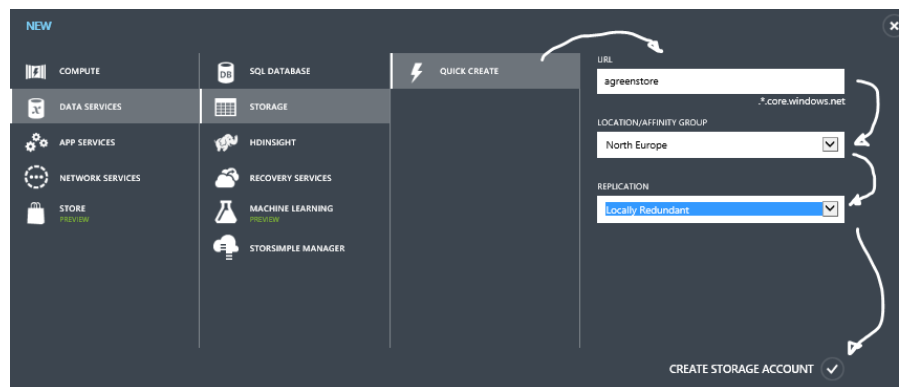
5.

You will get a red “tick” next to the URL name if it is OK.

6.

Choose a **location** – this is which DataCenter in the world you want to place your storage account... Your subscription MIGHT NOT allow all locations.

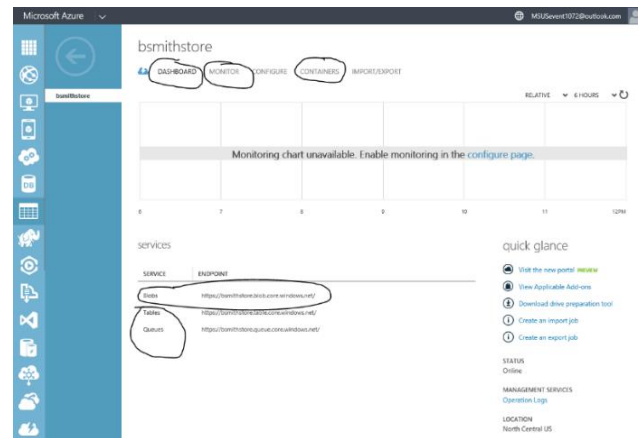
Select **Locally Redundant** replication – this means data in your storage account is NOT replicated to another Azure data center (we don't need it for this lab, it's also cheaper and faster).



Click on **CREATE STORAGE ACCOUNT**. It will take around 30 seconds for the account to get created (status: ONLINE).

A storage account has a few different types of services you can now use. One of them, is BLOB storage – essentially a big file system...

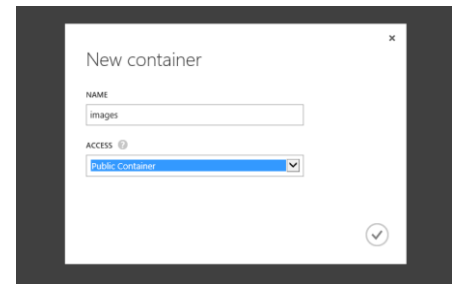
7. Each Service has a set of activities you can perform, grouped into sections. **Click the Name of your storage account**, which will take you to the overview page. Then click the **DASHBOARD** tab –
8. this is an overview of some key service metrics/health for your service and other configuration information.




Click on the **CONTAINERS TAB** (at the top) and create a new container called **"images"** (LOWER CASE ONLY) and set the access to **Public BLOB**

9.

Ok so you have what is now like a folder on your computer, only it lives in the cloud... and it is very resilient - there will be 3 copies of everything you store in this folder in redundant racks in the data center (and if you had turned on geo-replication - another 3 copies in the replicated DC). But you don't have anything YET in your "folder".



10.

To get content in your "container" you need to **install** a tool that understands Azure storage. There are many tools available that do this, most are free. Back on the Azure Portal, click on the Quick Start link in the portal (the  icon).

11.

Click on the Storage Explorers link in the "Get the Tools" section. You can pick any of the links – in this lab we will use the first one – the Azure Storage Explorer. Follow the instructions to download and install the Azure Storage Explorer Tool on your machine:

12.

<http://azurestorageexplorer.codeplex.com/releases/view/125870> and download the latest release.

13.

Run the Tool (Press the **Window** button->type **Storage**) and launch the Azure Storage Explorer application. Click the **Add Account** button on the toolbar. The tool needs the storage account name and your storage account key. Back in Azure, go to your storage account and at the bottom, click on **MANAGE ACCESS KEYS**.

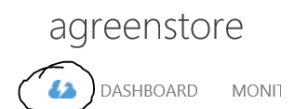
14.

15.

**Copy** the **storage account name** and the **primary access key** into the relevant fields in your storage explorer tool and click on **Add Storage Account**.

Get the set of assets for this lab from the **lab download folder** here: <http://1drv.ms/1DcUEnI>

Check the **"Azure\_Intro\_to\_App\_Dev"** folder and select **DOWNLOAD** in the header. Save the file to your desktop, right click the file on your desktop and select **EXTRACT ALL...**



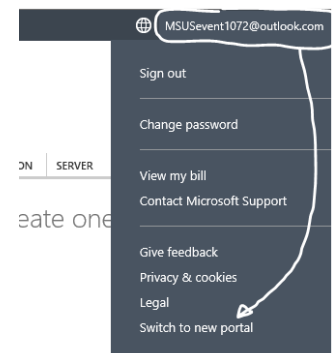
Using the storage explorer tool, upload the files in the images folder to your images folder in your blob storage account.

Your BLOB storage account is really just a file system on Azure – but a highly resilient one. Many companies first experience Azure simply by using BLOB storage as a way to backup their on-premise data and have a simple, cost effective off-premise backup location. You can connect backup in Windows easily to backup your servers and file shares to Azure.

All the other services in terms of creation and management work like this, each has its own set of needs and commands but creating and managing the services happens in this way.

## 3.2 The New Preview Portal

The management portal is being re-designed to support more flexible compositions of services as well as new workflows to create and manage more complex applications. This “Preview” provides much richer information when you create as well as manage your services. Let’s take a quick look as increasingly this will become the place to do everything (even though not all services and capabilities are available in the preview portal at this time).



1. On the current portal, click on your login ID at the top left and select the **Switch to New Portal** option at the bottom.

A new browser tab will open...

2. The first/homepage is called the **Startboard** – this can be customized to show the things that are important to you.

Let’s first FIND your storage account. Click on the **BROWSE** icon in the left nav, and select **STORAGE**. A new panel opens up alongside the Startboard – this is called a **BLADE**.

3. NOTE: Not all services are available yet in the New Portal, these will be added over time which means going between both Portals regularly.

4. On the **Storage BLADE**, click the **PIN icon** (top right on the blade) – this will add the blade to your Startboard so you can access it quickly.



**Right Click** anywhere on the **Startboard** and select **CUSTOMIZE...** Drag your Storage Icon to a different place on the startboard, select **DONE** to save your changes.

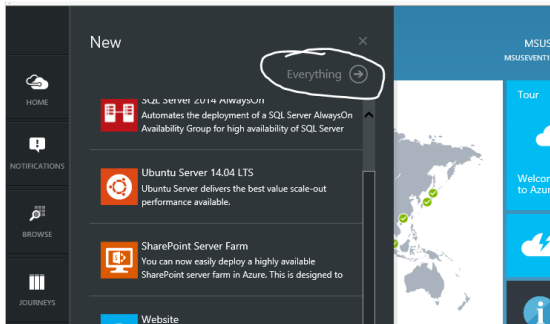
Now you will create a new service in the New Portal – this time you will create a new Virtual Machine. You still have the familiar “+ **NEW**” icon, click it...

This time you get a list of just “some” of the services you can create but notice this includes more complex things – like “SharePoint Server Farm” and Website + SQL

So this allows collections of services, multiple services to be provisioned all at once.

Click the **EVERYTHING** link

6.



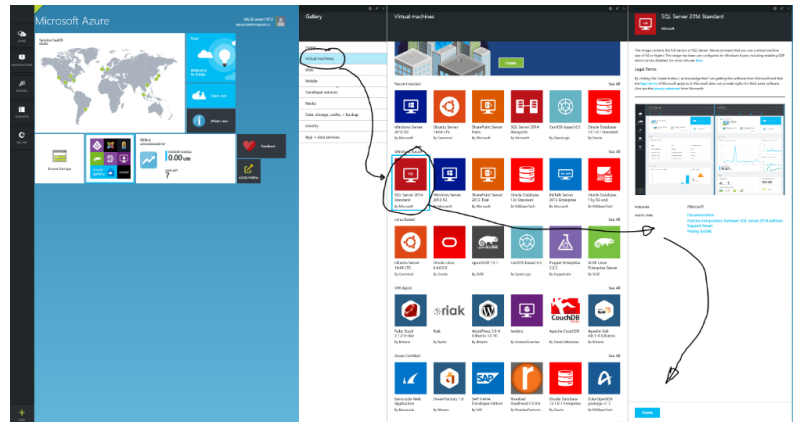
This displays the Gallery Blade (which is also by default added to your StartBoard). It is like the categories in the old portal and then when you click a category, you see many more options that include not just individual services but collections of services.

7.

Click on the **Virtual Machines** category.

8.

Select the **SQL Server 2014 Standard** item – this will open up another blade with details about that item – click the **CREATE** button.



Another BLADE appears – this time it is for information you need to enter about the resource – in this case a VM. It needs the name you want to call the machine and a user account/password that will be the admin account on the box.

- For the Host Name – use **<alias>-sqlvm** – e.g. agreeen-sqlvm
- For username - use **azureadmin** – just so you don't forget...!
- For password – use **1stAzure** – again – just so you don't forget

9.

Click on the Pricing Tier and select BROWSE ALL PRICING TIERS in the blade – you will see a very rich view of the estimated costs, capacities and other limits for a particular VM size. Select the **Standard A2** option.

10.

Click on OPTIONAL CONFIGURATION. On the new blade, select the **STORAGE ACCOUNT section** and then select the EXISTING storage account you created earlier. This will also mean your VM will be created in the same data center location where your storage account was created. Click OK.

11.

Click on RESOURCE GROUP and Click CREATE A NEW RESOURCE GROUP. Enter **alias-rg** as the name for the resource group – e.g. agreeen-rg. Click **OK..**

12.

The click **CREATE**. The blade will close and an icon for the VM will be created on your StartBoard with updating status.

13.

This will take a while – you will do something else while it fires up...!

Create VM

HOST NAME  
bsmith-sqlvm

USER NAME  
azureadmin

PASSWORD  
\*\*\*\*\*

PRICING TIER  
Standard A2

OPTIONAL CONFIGURATION  
Network, storage, diagnostics

RESOURCE GROUP  
Group-1

SUBSCRIPTION  
Azdem207B37118C

LOCATION  
East Asia

☒ Add to Startboard

Create

## 4. Very Basic Web Site

Many business applications today, internal and external ones are web applications. Azure has some special services that really make deploying and running web applications simple. You don't have to create your own Virtual Machines and install any web software and configure everything – you just spin up a web site and get going in literally seconds... So in this next part you will not only create the web server on Azure (using the website service – I know – confusing or what), you will BUILD your very own web application using HTML.

Let's stay on the Preview Portal... Click “+  
**NEW**” -> **WEB SITE**

1. Your website needs a name – this will be the first part of the public url to your site – to be consistent with our other naming – use **<alias>** - **web** e.g. psmith-web
- 2.

3. Click on WEB HOSTING PLAN. In the Create New section, enter a name for the plan – use “basic-northeurope” and select the B1 plan. Click OK.

4. (A hosting plan is a collection of web sites that you put in the same data center region and with the same web site mode). If you change the web site mode, you change the mode for all web sites in that plan.

5. Click on RESOURCE GROUP – select the existing resource group <alias-rg> that you created when you created your Virtual Machine. Remember, Resource groups allow you to put related resources together and see/manage all resources for a particular group.
- 6.

Finally, select your data center location – choose “North Europe”. Click OK

Now click **CREATE**.

After the web site creation completes – about 30-40 seconds, your site will show up on the StartBoard and a blade for managing your web site will be displayed.

The screenshot shows the 'Website' and 'Web hosting plan' creation wizard in the Azure Portal. The 'Website' pane on the left contains the following fields:

- URL:  (with a .azurewebsites.net suffix)
- WEB HOSTING PLAN: **basic-northeurope (Small: Basic)** (selected)
- RESOURCE GROUP: **agreen-rg** (selected)
- SUBSCRIPTION: **Azpas337A5W7724** (selected)
- LOCATION: **North Europe** (selected)

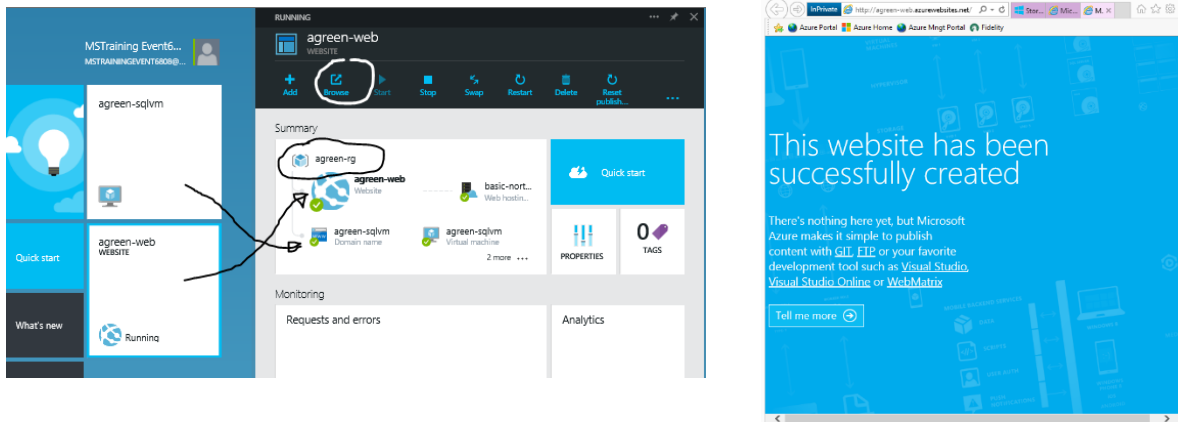
The 'Web hosting plan' pane on the right shows the 'Create new' section with the following details:

- NAME:
- Plans: **B1 BASIC** (selected), S1 STANDARD, S3 STANDARD
- 1 Core
- 1.75 GB RAM
- Storage: 10 GB
- Custom domains: ☐
- Manual scale: ☐ Up to 3 instances

At the bottom, there is a checkbox for 'Add to Startboard' (checked) and a 'Create' button. The 'OK' button is at the bottom right of the 'Web hosting plan' pane.

Click the **BROWSE** button at the top of the blade. This opens a browser pointing at your web site url... It just has a simple default page right now.

7.

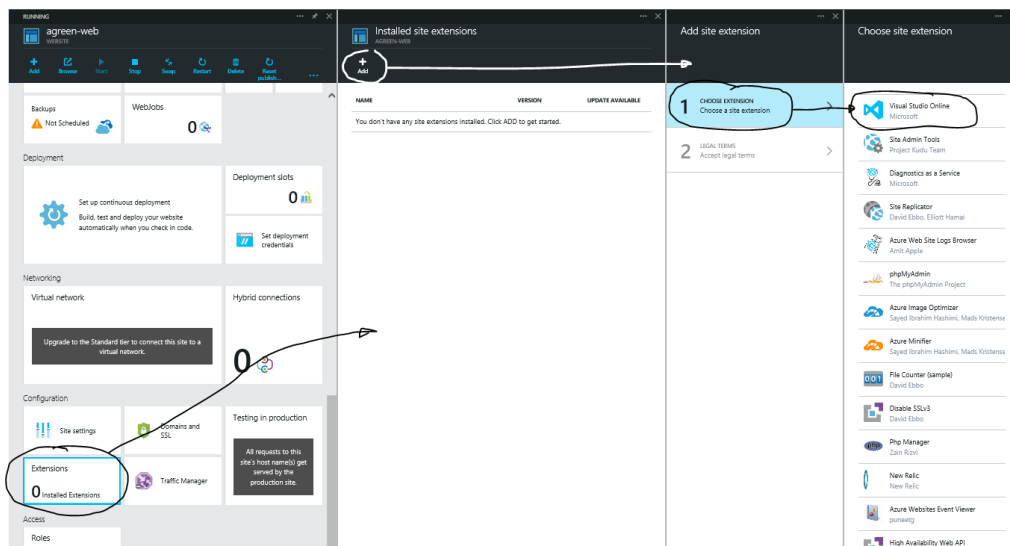


Congratulations, you just did something that would have taken about a month in the past. You have a web server running (it's running Windows Server and IIS), you have a domain name and entries in DNS, the Web Server is getting and responding to traffic from the Internet, you have some code on your web server for your application (the start page that was just shown). It took all of a few seconds to make that happen.

Now you are going to start building your web site but you need a development tool to write your web code. There is one built into the Portal, let's turn it on for your web site.

8.

Go back to your **website blade**. Scroll down to the bottom of the blade and you will see a **configuration** section. Click on the Extensions tile and the Extension blade will open.





Click **Add** in the header, click on **Choose Extension** and then select the **Visual Studio Online** option and **click OK** at the bottom of the blade to accept the legal agreement. **Click OK again** to add the site extension.

Click on the **Visual Studio Online extension** and in the blade header, click the **BROWSE** button. A new browser window opens which is the VS Online editor experience or “Monaco”. You are now in real-time editing your web site code directly on the web server in Azure.

9.

IF MONACO does not load, go back to your web site blade, restart your web site and then try to browse to MONACO again from the extensions blade.

10.

There are many other development options, like the full Visual Studio developer tools. For this lab though, you will stay in VSONline “Monaco” as it’s more about simple web development and using Azure Services than getting to experience the full “premier” tooling available in Visual Studio.

In “Monaco”, **Right Click** the **hostingstart.html** (#1 opposite) file and select **Delete**. This is the page that was displayed when you browsed to your empty site).

11.

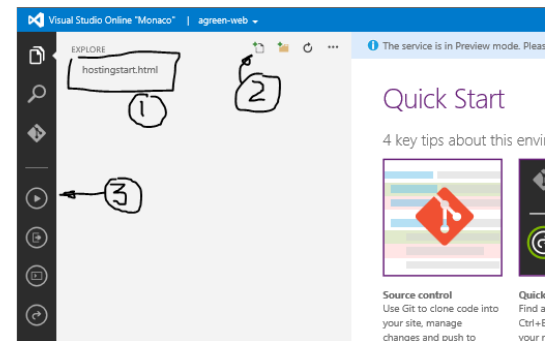
Click the **New File** icon (#2 in the picture opposite) and enter **default.cshtml** as the filename (**this is important you call it default.cshtml**)

12.

You are going to create a web site of your favorite movies... **Copy the HTML below and paste it into your new page.**

13.

```
<html>
<head>
  <title>My Fav Movies</title>
</head>
<body>
  <h1>A list of my Favorite Movies</h1>
  <div id="center">
    <div id="bio">
      <p>
        This is me and this is my bio that tells you all about me
        and it might give you some clue as to why my list of favorite
        movies is the way it is...
      </p>
    </div>
    <div id="movielist">
      <ol id="mylist">
        <li><a href="http://www.imdb.com/title/tt0038650">It's a wonderful life</a></li>
        <li><a href="http://www.imdb.com/title/tt0059742">The Sound of Music</a></li>
        <li><a href="http://www.imdb.com/title/tt0088247">The Terminator</a></li>
        <li><a href="http://www.imdb.com/title/tt0110357">The Lion King</a></li>
      </ol>
    </div>
  </div>
</body>
</html>
```



Click the **RUN** button (#3 in the picture above) on the left nav. You need to hit **REFRESH** on your browser since IE cached the default page you just deleted. Note: if you EVER expect something to appear and it does not in the browser – try hitting REFRESH first. The browser caches pages.

Switch back to **Monaco** and change something in your app – **change the bio text** so it's about you...

14. Switch back to your **web site** and click the **refresh** button on your browser – you should see your change instantly. You are live on the Internet with your new web site and you did everything from the browser – even web development. Pretty Cool.

15. Laying out web pages in various sections and making the pages and content look attractive is half the battle in  
16. modern web sites and applications. Cascading Style Sheets (CSS) is the way to separate out the layout/look of your web pages from the content of the page.

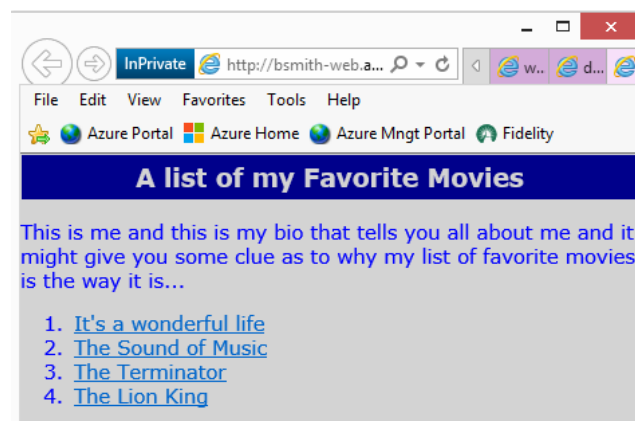
In your default.cshtml file, right after the <title>...</title> line, add a new line and paste in all this code.

17. 

```
<style>
  body { color: blue; font-family: Verdana; font-size: 11pt; margin: 0; background-
color: lightgrey;}
  h1 { background-color: darkblue; border: 1px solid lightgrey; color: lightgrey; text-
align: center; font-size: 14pt; margin: 0; padding: 5px;}
</style>
```

18. The body {...} and the h1 {...} are selectors – they say find ALL the body/h1 tags in the HTML file. You can also put your styles in a separate file and then share styles across pages – we are just doing it the simple way.

Go back to your website and click REFRESH to see your changes.



As you can see so far, Windows Azure is really just running your site – all the real work is in building your application. We want to display the content side by side and make our <bio> block a little more attractive... To do this, we need to style each of the sections of our page that we identify using the <div> tag and identify which div tag using an ID that we create.

**Copy and Paste these styles below** into your list of styles so that it resembles the picture to the right...

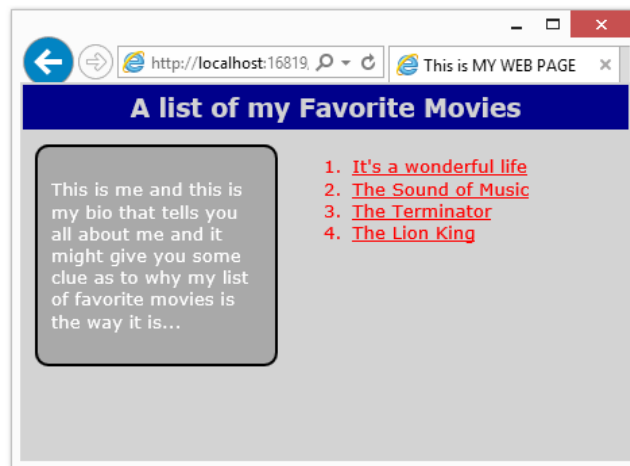
19. 

```
#center{display: flex;}
#bio{border: 2px solid black; border-radius: 10px; background-color: darkgrey; color: white; margin: 10px; padding: 10px; width: 40%;}
#movielist{margin: 5px; width: 60%}
```

**REFRESH** your browser tab pointing at your web site. You will now see these new styles applied. OK – that’s the basics of HTML and CSS. Now let’s start using other Azure Services...

20.

```
default.csshtml /
1 <html>
2 <head>
3   <title>My Fav Movies</title>
4   <style>
5     body { color: blue; font-
6       h1 { background-color: dar
7     #center{display: flex;}
8     #bio{border: 2px solid bl
9     #movielist{margin: 5px; wj
10  </style>
11 </head>
12 <body>
13   <h1>A list of my Favorite M
14   <div id="center">
```



## 5. Using your BLOB storage Account

1. You have a number of image files in your BLOB storage account. Look at these files in your storage explorer tool and click on one of the files to see it’s URL. Since the permissions on your “images” folder were set to BLOB level permissions, each of these files is accessible so long as you have the URL. You will simply put one of these file URL’s into your web application.

Copy the <img> tag below... You can change the src="" attribute to point to one of the images in your BLOB storage account.

IF you leave the code below unchanged – it will still work

since the teazurestore storage account is a real storage account, it’s just owned by someone else and in a

```
8   <div id="center">
9     <div id="bio">
10    <p>
11    
12    This is me and this is my bio that tells you all about me
13    and it might give you some clue as to why my list of favorite
14    movies is the way it is...
15    </p>
16    </div>
17  <div id="movielist">
```

different Azure subscription.

```

```

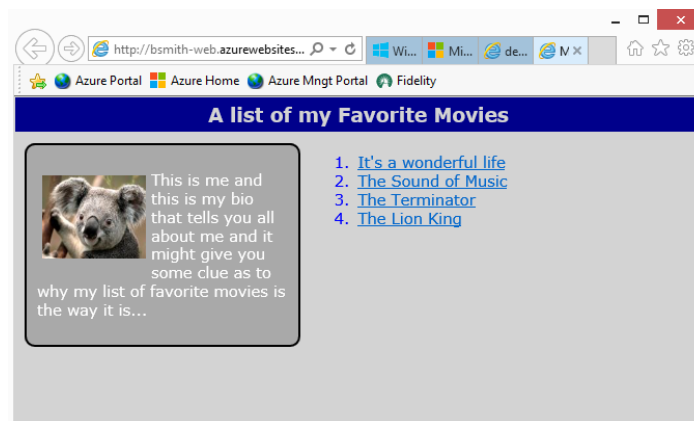
The image needs to be controlled and the text on the page needs to properly FLOW around the image. Add in the following CSS style for your <img> tag to the list of styles in your default.cshtml page.

```
#bioimg {width: 100px;height: 80px;float: left;padding: 5px;margin: 0;}
```

2. The key style here is **float: left** – this puts the element on the left of the containing box – which is the #bio div and then makes other elements float around it.

**REFRESH** your browser you should have some picture in the right place and sized nicely in the #bio tag. It should look something like this:

3. Use some of the other images in your storage account – all you need to change in your src="" line of code is the name of the .jpg file.
- 4.



## 6. Using Your Virtual Machine

### 6.1 Connect to your VM it should be ready...

Your Web Site just has a static list of movies. Many Web Sites are dynamic and get their data from a database. There are a number of data storage options you can use in Azure. The two most often used are using a database in a VM and using the database service – Azure SQL Database. Since you already created a VM and put SQLServer in it, let's start there.

Go back to the **Azure portal**, and click on your – sqlvm Virtual Machine tile from the Startboard.

You will see the main blade for the VM, everything should be running.

You will now connect to your machine – click the **CONNECT** button at the top of the blade

1.

Your browser will prompt you to Open/SAVE a .rdp file – this is a configuration file with the details the remote desktop application needs to connect to your VM.

2.

3.

4.

Select **OPEN** and then click **CONNECT** on the next



5.

You will then see a logon dialog. Click the **USE ANOTHER ACCOUNT** option and enter the **username** and **password** you entered when you created the VM – (the suggestion was username: **azureadmin** and password: **1stAzure**). If you forgot these, you will have to delete and re-create your VM.

6.

Select **YES** on the certificate warning. You will then be logged into the machine.

7.

A little DEMO trick – if you click the restore-down button on the remote desktop window and then on the window icon click the smart-sizing option – you will be able to dynamically change the screen size and all contents will auto-scale to the new window size. Click **NO** on the network discovery panel.

8.

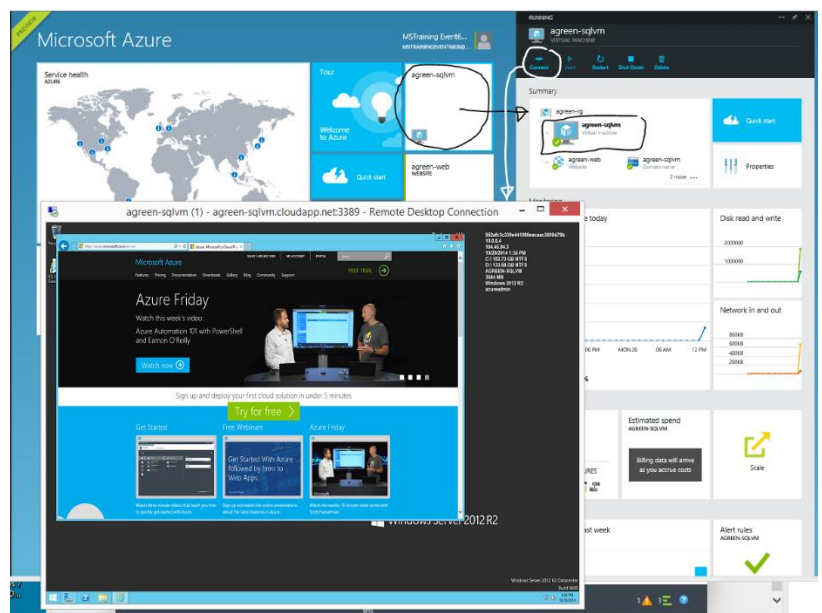
This is just a computer – just like your laptop. It's running Windows Server and SQLServer. Let's use this VM to browse to your new web site on the internet. Since this machine is a server though, it has some EXTRA protections that we need to turn off.

9.

Server Manager in the VM will be started. Click on **Local Server** (Top left in Server Manager) and then find the **IE Enhanced Security Configuration** option in the middle top section. Click it and turn it **OFF** for admins and users.

10.

Now open up **Internet Explorer** (From the START screen). Go to a web page – like [azure.microsoft.com](http://azure.microsoft.com). As I said – it's just a computer...



## 6.2 Creating Your Database and Configuring your VM for Access

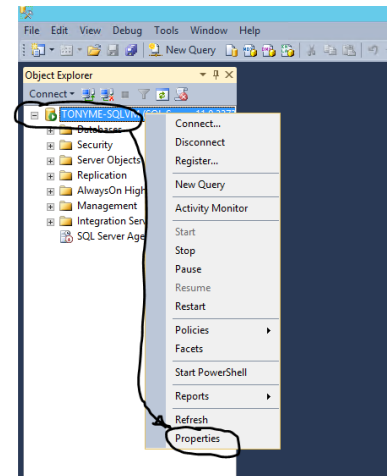
You will use this machine to create a movies database and then in your web application to read records from this database to display in your web page instead of your static list of movies. So, you need a database and then you need to configure your VM/SQLServer so that your web site can communicate with your database securely.

On your Azure VM click the **Window** button and **type** “sql management”. Click on the SQL Server 2014 Management Studio application. This application is the classic tool to manage all aspects of SQLServer.

After starting up you will be prompted to login and by default SQL Mngt Studio will try and connect to the SQLServer on your VM and use the VM credentials. SQLServer is setup to use Windows Authentication by default. **Click Connect.**

1. **Click Connect.**
2. You will change the authentication method first, because you will be connecting to SQLServer from your Web Site – you cannot use Windows Authentication. In the **Object Explorer** on the left, right click the **top level SQLServer node** and select **properties.**
3. **Click on Security** in the Select a Page area and in the **server authentication section**, select **SQL Server and Windows Authentication mode.** You will get a warning about needing to restart SQL Server – click OK.

4. **Click on Security** in the Select a Page area and in the **server authentication section**, select **SQL Server and Windows Authentication mode.** You will get a warning about needing to restart SQL Server – click OK.
5. Right Click again as you did above on the database server and select **RESTART** and select Yes to do it.



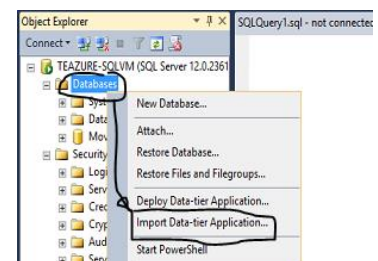
Next you will create a database on the server. In the set of files you downloaded for the lab to your desktop, there is a databases folder and two .bacpac files. These files contain the databases for this lab including database user 6.accounts as well as the data itself.

7. The file you need is the **MoviesSQLVM.bacpac** file in the **databases** folder from your **download** (on your lab machine). Locate and right-click this file and select **copy.**
8. **Switch** to your VM. On your VM desktop, right click and select **PASTE** (pretty cool eh?).

9. **Switch** to your VM. On your VM desktop, right click and select **PASTE** (pretty cool eh?).

Back in **SQL Management studio** on your VM, **right click** the **databases** node and select **Import Data-Tier Application.**

10. Follow the Wizard, pointing at the MoviesSQLVM.bacpac file to **import** from your desktop and accepting all other defaults. After completing the Wizard you will have a new database.



NOTE: You would normally install the database files on a separate data disk in your VM – you are just doing it this way for convenience/learning.

There is one last thing you need to do. To improve security, logins in the bacpac file are stored without passwords. We have a special login “Movies\_reader” that is assigned with database reader permissions on the movies database. You need to reset the password for this login. In **SQL Mngt Studio**, expand

the **Security** folder and the **Logins** folder. Right click the **Movies\_reader** login, click **Properties** and change the **password to "1stAzure"**. Click **OK**.

**Browse** to the **Movies** table in the database and view the data in the table (right click the table and click on **Select Top 1000 Rows**).

If you tried to connect to the database from outside the VM, you would not be able to. **SQLServer** communicates over a specific TCP/IP port (number 1433). The firewall built into Windows does not by default have this port enabled – so traffic cannot flow into the VM.

11.

Go to the **Windows Start** screen **in your VM** and type **Firewall** and click the **Windows Firewall with Advanced Security** icon.

12.

You will need to **create an inbound** rule (so SQL traffic can flow into your VM). Click on **Inbound Rules** and select **New Rule** in the Actions tab. **Follow the Wizard** creating new rules of type **1433**, applied to **TCP** and using the **specific local port** number **1433**. Accept all other defaults. Call your rule **SQLIN**.

So now your Windows VM will allow TCP 1433 port traffic to it. However Windows **Azure** WILL NOT – you have to enable that as well. I know right ! It's for your own good, you can never have too much security 😊.

13.

In the **Azure portal**, on **your VM blade**, scroll down and find the **Endpoints** tile. Click on the **ENDPOINTS** tile and click **ADD** in the new blade. Call the endpoint "SQL", the protocol is TCP and enter 1433 for both the public and private ports. Click **OK**. The notifications area in the portal will tell you the progress. It will take a couple of minutes to add the endpoint and you will see it listed in the new blade.

Finally you are done configuring your VM, **SQLServer** and **Azure**.

## 7. Programming your web site to display data from a database in a VM...

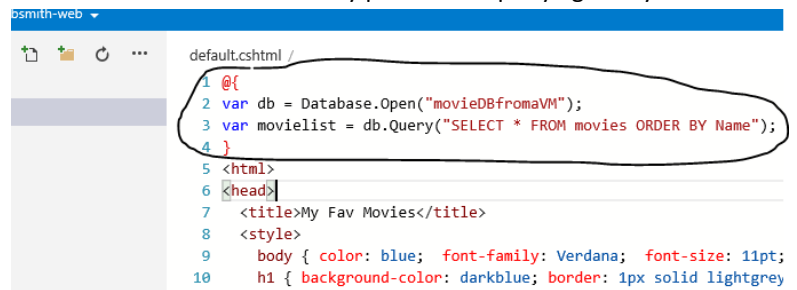
1.

So now you have a database, you will connect your web application to that database and read the movie records in the database to display them on your web page. For this, you need to do some programming.

Back in your **default.cshtml** page, first you need to create a connection to the database. At the very start of your **default.cshtml** file, before the **<html>** tag, add in these 4 lines of code...(well 2 lines – the **@{ ..}** lines tell the web server that this is a code section (and the **.cshtml** file extension, tells the web server that it's **.NET** code and what module to run to process this code – **cs** means **C SHARP** code.)...

```
@{
var db = Database.Open("movieDBfromaVM");
var movielist = db.Query("SELECT * FROM movies ORDER BY Name");
}
```

The first statement creates the connection. The second statement actually performs a query against your database and all the records that are returned in this query are assigned to the variable movielist. So movielist is actually not a single value, it is a collection of values.



You might be wondering how your web knows how to connect to your VM and SQLServer – how does it know this from the name “movieDBfromaVM” value. Well it does not – YET. You will have to add this configuration to your web site and your code will then pick up this value.

Next, you need to **loop through** all the values that were returned in the movielist variable (the highlighted area in the picture below), and for each one of these values, inject the value of the name and link into your HTML page from the database in the right place. Fortunately, looping through collections of

- values is one of things we do most often in programming and so there are looping commands that help us.

- When we look at the HTML, we see that in the highlighted section, this is the HTML that keeps repeating, so this is where we need to inject our loop and within the loop, we will



have a single line that is similar to each line above, but instead of a fixed value we get a value from the database. So **replace the whole highlighted section** in your code with this:

```
@foreach(var movieItem in movielist)
{
    <li><a href=@movieItem.imdblink>@movieItem.name</a></li>
}
```

It should look like this...

- ```

26 and it might give you some clue as to why my list of favorite
27 movies is the way it is...
28 </p>
29 </div>
30 <div id="movielist">
31   <ol id="mylist">
32     @foreach(var movieItem in movielist)
33     {
34       <li><a href=@movieItem.imdblink>@movieItem.name</a></li>
35     }
36   </ol>
37 </div>
38 </div>

```

As you can probably guess, **foreach** is a loop programming construct and everything inside the {..} is executed for each value in the movielist variable. So that you can reference these values in your code, in



each iteration of the loop the current values in movielist are assigned to the new variable movieitem. One important point – CASE is SENSITIVE...! So movieitem and movieltem (upper case "I") are different variables.

Right you are almost ready. The only thing left is to tell your code where to find your database using the name **movieDBfromaVM**. The way to connect to a database and provide the connection information and security credentials is using a connection string. You can do all this in the Azure Portal.

5.

Go to the **blade for your web site**. Scroll down to the bottom of the blade and find the **Configuration** section.

6.

Click the **Site Settings** tile to open the Site Settings blade. You will see a **Connections Strings** section.

7.

In the first row enter in the **NAME** field the value: **movieDBfromaVM**

8.

**Copy/Paste** the connection string below into the **Value** box **REPLACING** teazure-sqlvm with the name of **your VM**

9.

```
Server=tcp:teazure-sqlvm.cloudapp.net,1433;Database=moviessqlvm;User ID=movies_reader;Password=1stAzure;Trusted_Connection=False;Encrypt=False;Connection Timeout=30;
```

10.

11.

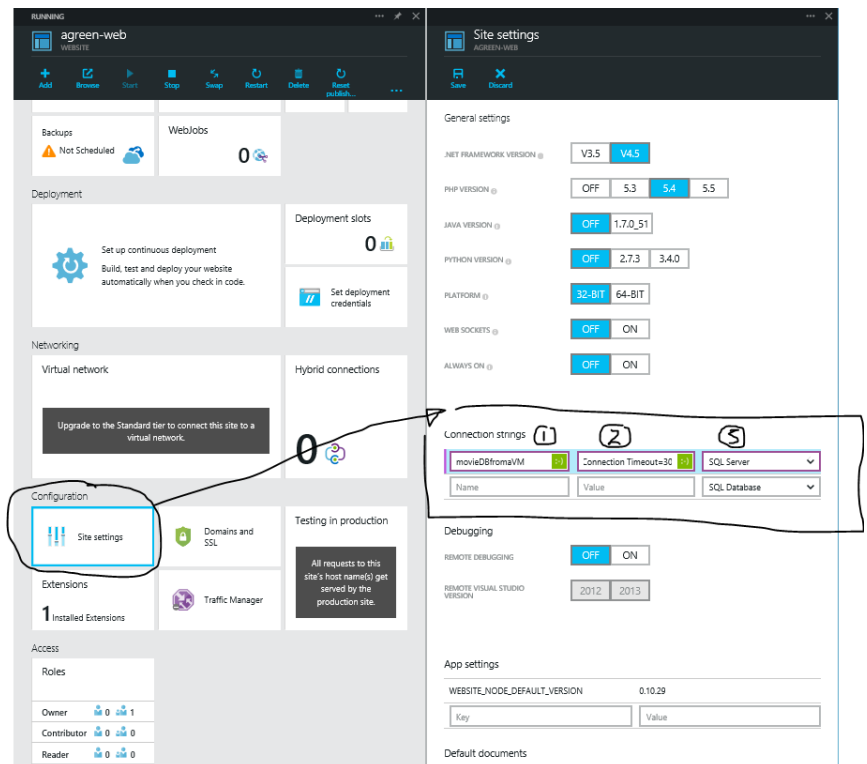
If you don't change this, you will find that it magically still works. That's because you are using someone else's database – the author of this lab.

Set the next box to be **SQL Server**. Click **SAVE at the TOP**.

Back on your web browser for your web site – click REFRESH. Did it work...?

Now the magic of making your web site talk to your database is that it's now dynamic, any time you add/change/delete records in the database, they are immediately reflected for users who go to your site, and you don't have to change any web code or anything.

You can try this by making a change to the data on your SQLServer.



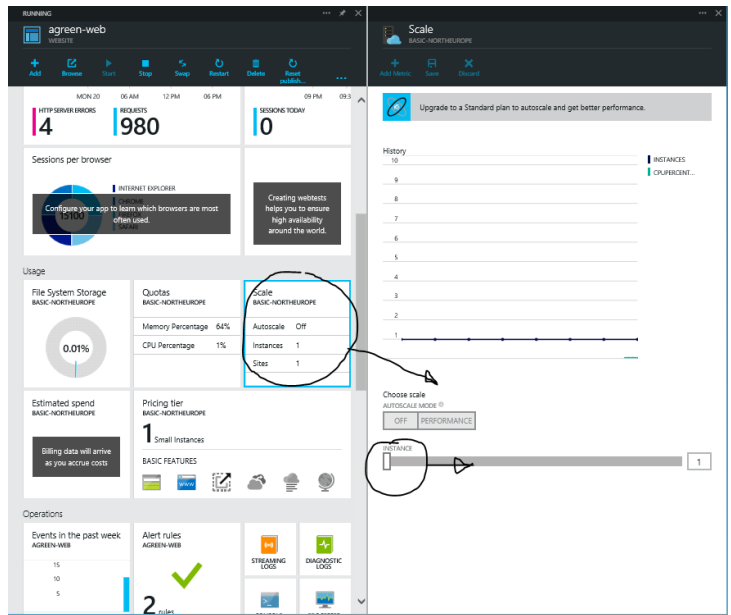
## 8. A Quick Lesson in Resilience and Scale

You now have a web application built, running on a web server (the Web Sites Service) in Azure, talking to a Virtual Machine running SQLServer that is hosting your database and connected to Azure BLOB storage for some of your web content. Pretty Cool...

One problem though is that apart from BLOB storage, your web application is not very resilient. If your web server crashes, you have lost your application until Azure detects this and spins a new one back up.

Worse, if SQLServer crashes in your VM or Windows Server itself crashes, Azure won't know about this – you are responsible for managing everything in your VM. You can solve the web part of the problem by adding additional web site instances – giving you both better scale/performance as well as resilience.

1. You do this on your web site blade by clicking the **SCALE** tile.
2. Scaling to multiple instances requires at least BASIC web site mode.  
Drag the instance slider to **3 instances** and click **SAVE**.  
Now all requests to your web site are served across the 3 instances in round-robin fashion, although web sites also implements application request routing so the sessions are quite sticky to a particular client session.



However, your application also has a database. That is NOT resilient. You can of course make the database perform better by having a larger VM and configuring your VM disks to have better IO, but it still won't be resilient. To make it resilient, you would have to implement a SQL14 always-on cluster, so you would need to spin up additional VM's and configure these to get this resilience.

In this next section, we will show you the database service – Azure SQL Database – which is BOTH resilient and scalable based on your needs.

## 9. <sup>1.</sup> Moving your database from a VM to Azure SQL Database.

<sup>2.</sup>

The first step is to create a new Azure SQL database. You will then migrate your current database in your VM to Azure SQLDB.

In the **Azure Preview Portal**, select + **NEW** -> **SQL Database** to get the New Database blade.

Enter <alias>-sqldb as the database name – e.g. agreen-sqldb

Click on **Pricing Tier** and select the **BASIC** tier.

Notice that each tier gives you a number of DTU's – basic has 5 DTU's. A DTU stands for a Database Transaction Unit – and it's an approximate measure of performance. The database is also resilient by design, there are actually three underlying databases spread across redundant physical hardware in the data center – all managed and kept in sync by Azure.

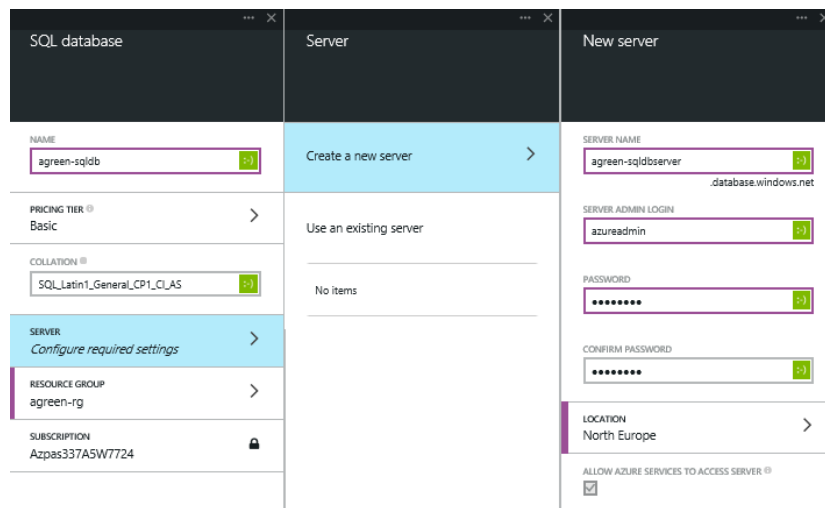
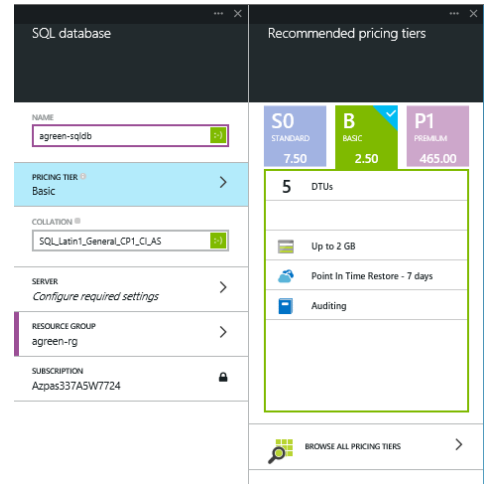
Click on **SERVER/Configure Required Settings**

Click to **create a new server** and enter the details for the new server using a name of **<alias>-sqldbserver**, a username of **azureadmin** and a password of **1stAzure**.

A "server" is a logical concept in Azure – a unit of management and configuration options for all databases attached to the logical server.

Make sure to select the **SAME location** as your Web Site location (you want to put your web and database tiers close together for lowest latency communication and fastest data pipe between them).

Complete the blades and create your new database.



Go back to your **SQLServer Virtual Machine** – you should still have the remote desktop console open (connect back to it if you closed it down).

In **SQL Management Studio**, right click the MoviesSQLVM database, select **Tasks** and select **"Deploy Database to Windows Azure SQL Database"**.

In the **Target Connection/Server Connection**, click the **Connect** button. The server name will be **<alias>-sqldbserver.database.windows.net**

Change the authentication to **SQL Authentication** and enter the username/password for the server you created above (**azureadmin/1stAzure**).

In the **New Database Name**, change it to **MoviesSQLDB**. Complete the rest of the Wizard.

To validate that this all worked, use SQL Management Studio in your VM to connect to your Azure SQL Database. In the **object explorer**, click the **Connect** Button and select database engine. Enter the same server name, username and password you entered above when you migrated the database.

**Browse** to the databases (yes you have TWO - <alias>-sqlldb and MoviesSQLDB – you can't just create a server by itself in the current portal implementation.

Expand the **MoviesSQLDB** to find the **Movies** table and then right click and select Script Table As -> Select To ->New Query Editor Window. And then Click to **Execute** this script.

Notice this is a different experience to when you could just select the top 1000 rows for your SQLServer – there is not parity of the SQL database management tool between SQLServer and Azure SQLDB yet – this is on the roadmap.

So now you have a database you need to use this database in your web application. Go back to the **CONFIGURATION/Site Settings** section for your **web site** and find the **connection strings** section.

Add a NEW row by entering in a blank NAME column the value **movieDBfromSQLDB**

In the VALUE field, copy/paste the connection string below:

```
Server=tcp:teazure-sqlserver.database.windows.net,1433;  
Database=moviessqlldb; User ID=azureadmin;Password=1stAzure;  
Trusted_Connection=False; Encrypt=False; Connection Timeout=30;
```

Click **SAVE** at the bottom of the Portal.

Now go back to **Monaco** and your default.cshtml file. **Run** the site so you know its working (remember the site is currently getting data from your VM/SQLServer.

Now, **change the value** of the var db=Database.Open("movieDBfromaVM") and replace it with the new connection string name to your Azure SQLDB – which is **movieDBfromSQLDB**

Press **RUN**. Did you get your data from your new Azure SQLDB...?

It's the same data of course. To prove you are using your new database, let's delete some records. Go back to your SQLServer VM – you should be connected to your Azure SQLDB.

Expand the **MoviesSQLDB** to find the **Movies** table and then right click and select Script Table As -> Delete To ->New Query Editor Window. In the Query Editor window, change the WHERE clause under the DELETE FROM line to "WHERE likeorder > 5". Click the **Execute** button on the toolbar.

Go back to your browser tab for your website and click REFRESH. You should see fewer records.

**THE END.**

## Appendix One: the full code for default.cshtml

```
@{
    var db = Database.Open("MovieDBfromaVM");
    var movielist = db.Query("SELECT * FROM movies ORDER BY Name");
}
<html>
<head>
    <title>My Fav Movies</title>
    <style>
        body { color: blue; font-family: Verdana; font-size: 11pt; margin: 0; background-
color: lightgrey;}
        h1 { background-color: darkblue; border: 1px solid lightgrey; color: lightgrey; text-
align: center; font-size: 14pt; margin: 0; padding: 5px;}
        #center{display: flex;}
        #bio{border: 2px solid black; border-radius: 10px; background-color: darkgrey; color:
white; margin: 10px; padding: 10px; width: 40%;}
        #movielist{margin: 5px; width: 60%}
        #bioimg {width: 100px;height: 80px;float: left;padding: 5px;margin: 0;}
    </style>
</head>
<body>
    <h1>A list of my Favorite Movies</h1>
    <div id="center">
        <div id="bio">
            <p>
                
                This is me and this is my bio that tells you all about me
                and it might give you some clue as to why my list of favorite
                movies is the way it is...
            </p>
        </div>
        <div id="movielist">
            <ol id="mylist">
                @foreach(var movieItem in movielist)
                {
                    <li><a href=@movieItem.imdblink>@movieItem.name</a></li>
                }
            </ol>
        </div>
    </div>
</body>
</html>
```