|  |
| --- |
|  |
| Capstone Project Document |

**DANDELION**

Report #6 – User Manuals

|  |  |  |
| --- | --- | --- |
| **Dandelion** | | |
| **Group Members** | Vũ Công Chính | SE02585 |
| Lưu Ngọc Mạnh | SE02619 |
| Nguyễn Minh Huy | SE02723 |
| Dương Đức Anh | SE02700 |
| Vũ Ngọc Trung | SE02967 |
| Cao Thị Phương Mai | SE02908 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Project code** | DDL | |

**- Hanoi, 09/2015 -**

# SIGNATURE PAGE

AUTHOR: Dương Đức Anh 7/12/2015

Project Manager

REVIEWERS: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 7/12/2015

Team member

APPROVAL: Nguyễn Văn Sang --/--/--15

Supervisor

Record of change

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Effective Date | Changed Item | A,M,D | Change Description | Reason for Change | Rev. Number |
| 19/Sep/2015 | Create User Manual | A | First version | Create User Manual | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE CONTENTS

# INTRODUCTION

## Purpose

This document contains guide-lines step by step to deploy DDL App to Azure website

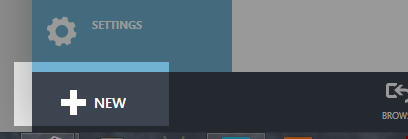
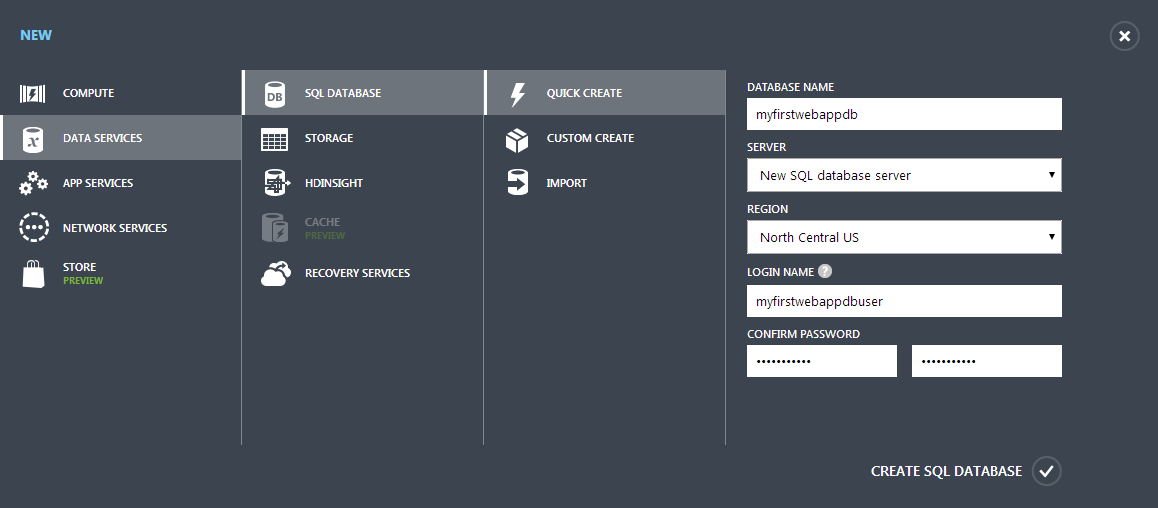
## Environment

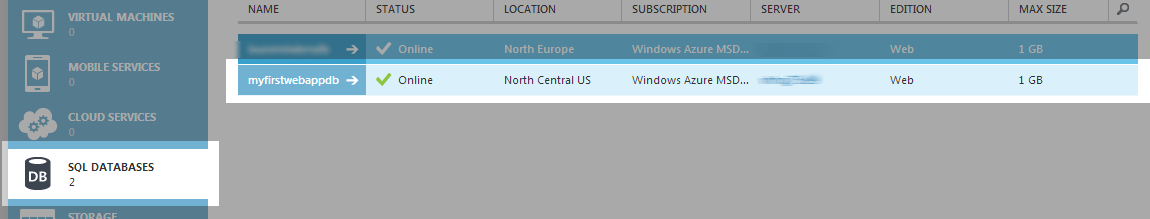
* Following are the software required to start the FAP system:
* Operating System: Windows 7, Windows 8
* Browsers: Firefox 40, Chrome 44 or higher.
* Database: SQL 2010 or higher
* .NET Framework 4.5

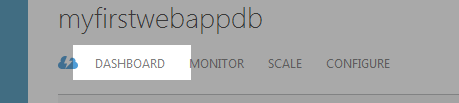
# [Connecting an ASP.NET MVC Web App with SQL Azure](http://www.codenutz.com/mvc-web-app-with-sql-azure/)

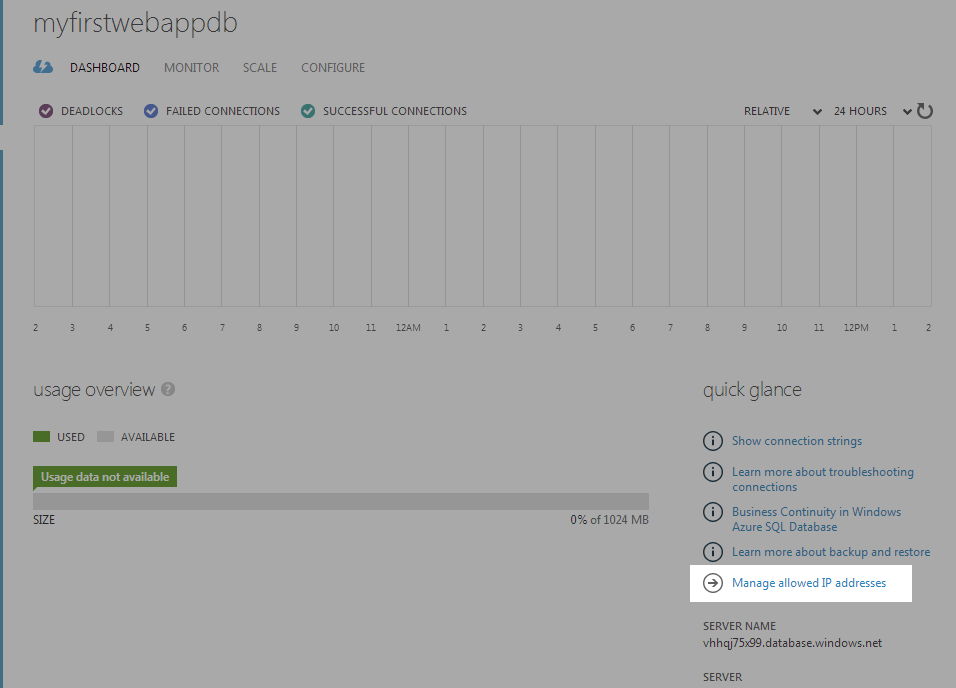
## Creating the SQL Azure Database

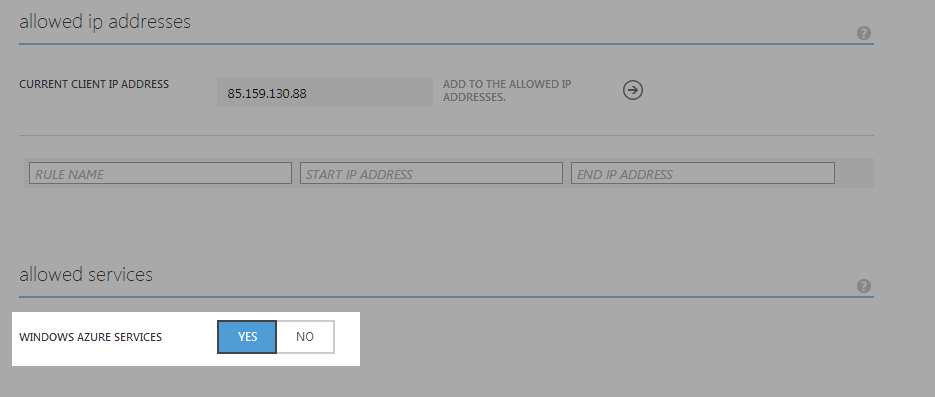
The first thing to do is creat your sql database on azure:

1. go to your azure portal, and click the new button in the bottom left corner
2. 
3. Then choose Data Services >> SQL Database >> Quick Create and fill in the form. My settings were:   
   a. Database Name: myfirstwebappdb   
   b. Server: Create New   
   c. Region: North Central US   
   d. **Make sure you remember the username and password you choose**   
   e. Press “Create Database” button   
   
4. After waiting a few seconds your database should be created.
5. Click SQL Databases on the left column, then select your database in the main pane

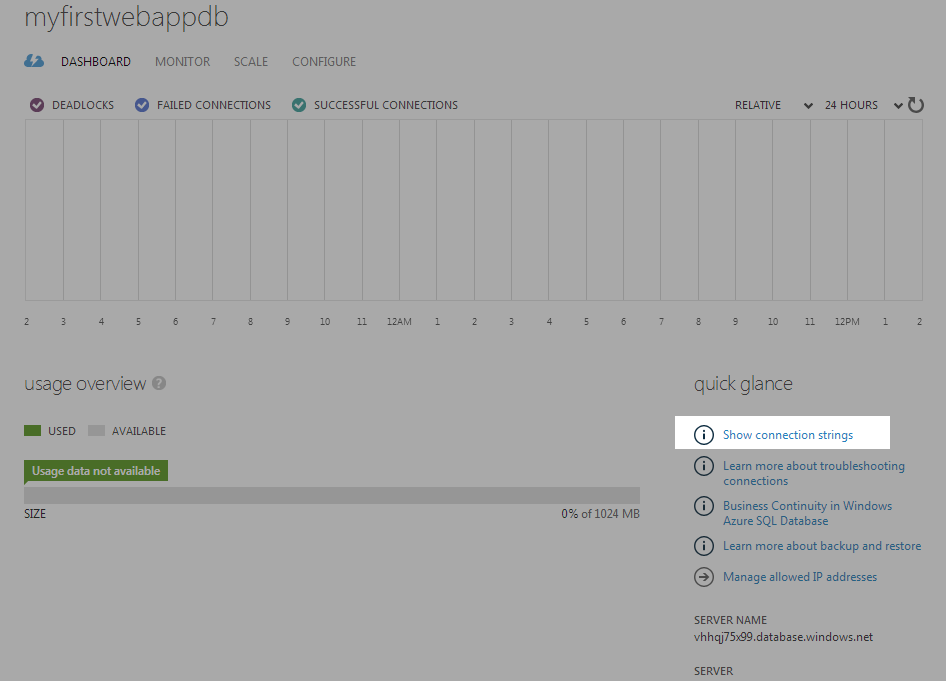


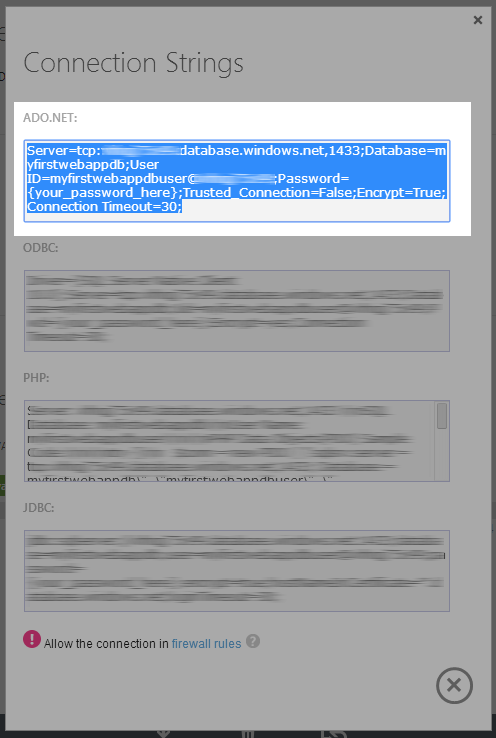
1. Go to the database dashboard   
   
2. Once on the dashboard you need to ensure that other azure services are allowed to communicate with you database (this should be set by default). If this option is not selected the firewall on the sql azure database will disallow connections from your website:   
   a. Select ‘Manage allowed IP addresses’

  
b. Ensure that ‘Windows Azure Services’ under ‘allowed services’ is set to Yes



1. Finally you need to get the connection string for the database:   
   a. From the database dashboard (see point 5), select “Show connection strings”

  
b. On the window that pops up copy the connection string labelled ‘ADO.NET’ for later use:



Your database is now set-up and ready for you to configure in your web app.

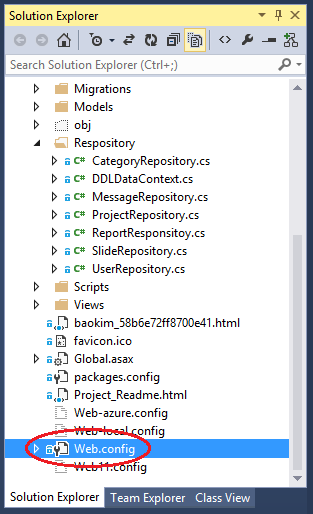
## Updating Connection Details in Your Web App

The next step is to update you mvc appication to point at the newly created sql azure database. There is one major thing we should take note of here though – generally speaking when we’re developing and working on a web app its better to be working locally (i.e. not connecting to the azure database). There are several reasons for this:

* Any communication with sql azure will have to happen over an internet connection, and will be comparatively slow to a local connection. Once you’ve deployed your website to azure this problem goes away as it is likely that your database will be located in the same data-centre as your website.
* Developing directly against the sql azure database will also require a constant internet connection, so you cannot work offline (or work at all if you temporarily lose your connection). developing against a local database alleviates this.
* Finally sql azure costs money – although the volumes may be low, it all adds up!

So the changes we make will allow us to continue to work off LocalDb when developing, but have our azure hosted website point at out sql azure database. To do this we’re going to take advantage of web.config transformations.

The web.config is the main configuration point for any asp.net website – its where you can store application settings, connection strings and much more. Web.config transformations allow us to change these settings seemlessly based on predetermined criteria.

1. Within your visual studio project, find the file called Web.config and open it   
   
2. Add the following code into the file directly

<connectionStrings>

<add name="DefaultConnection"

connectionString=”\*enter your connection string here\*”

xdt:Transform=”SetAttributes” xdt:Locator=”Match(name)” />

</connectionStrings>

1. Ensure that you replace ‘enter your connection string here’ with the one copied from sql azure earlier
2. There should also be a placeholer inside the connection string that read ‘{your\_password\_here}’ – this should be replaced with the password you set earlier.
3. Save the file

### How does this work?

The connection string represents a set of instructions about how your app will find and access a database. The connection string we provided tells the app how to get to the sql azure database we just created. This connection string will only be used when in the ‘Release’ configuration – note that we have 2 configurations ‘Debug’ and ‘Release’, debug is for local development, release is for our production environment.

The connection string for our LocalDb is contained within the Web.config file, and is used by default. When we decide to build and publish the release configuration, the instructions we entered into the Web.Release.config will replace those in the Web.config. Whenever we publish to azure, we are always using the release configuration.

### What do the instructions in the Web.Release.config actually mean?

The xml code we entered in the Web.release.config roughly says this

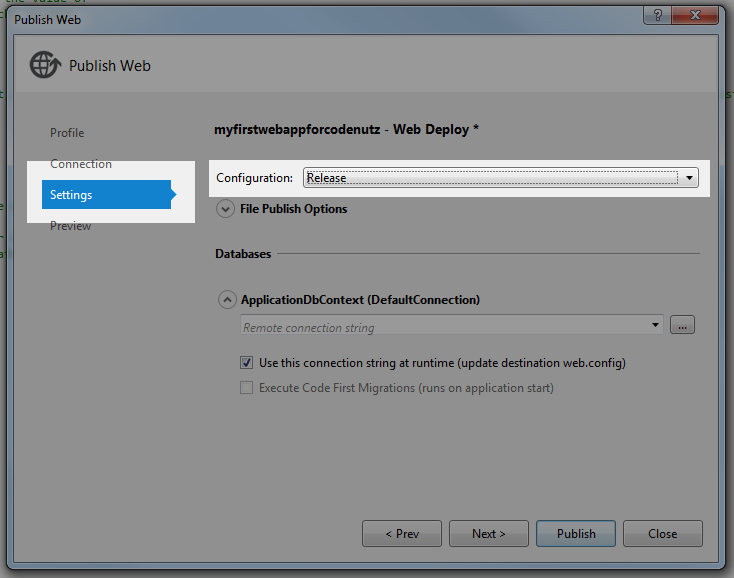
* Within the connectionStrings element
* Locate the element using ‘Match(name)’: locate any elements with the name attribute that matches the name attribute on this element, i.e. ‘DefaultConnection’
* Transform those elements using ‘SetAttributes’: set all attributes equal to the attributes on this element, i.e. set the connectionString attribute to our sql azure connection string.

These transformations aren’t written to disk, but simply applied as the website is published

## Deploy your website

To deploy your website you should just be able to right click on your project in the solution explorer, click ‘Publish…’ and follow the wizard through. for a more detailed step-by-step, see [Deploying to the Azure Website](http://www.codenutz.com/creating-deploying-asp-net-mvc-web-app-windows-azure/#deploying-to-the-azure-website).

On thing worth noting is within the publish dialog, if you go to the ‘Settings’ tab you should notice that the configuration selected says ‘Release’:

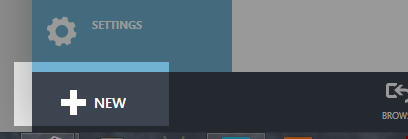


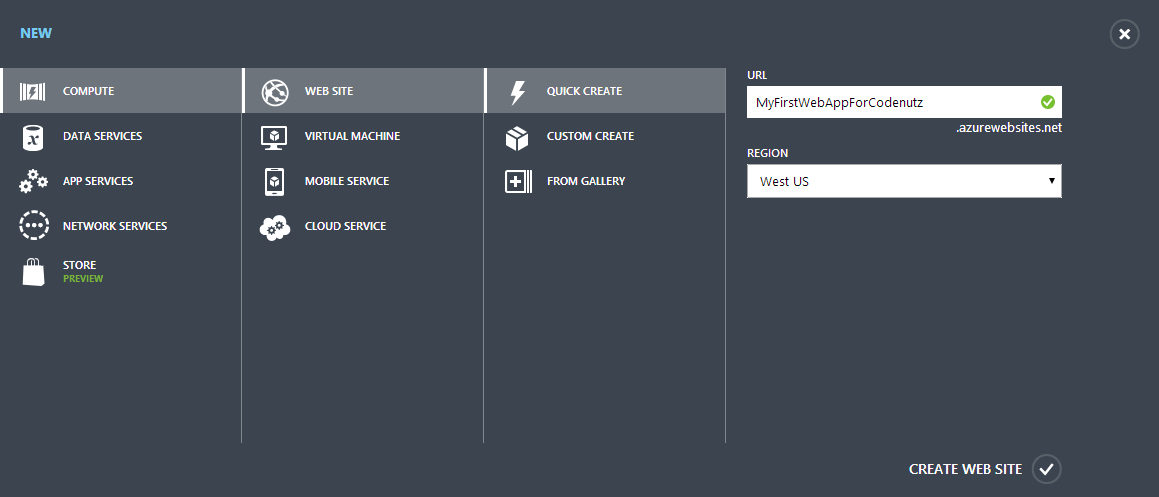
# Create & Deploying the Azure Website

## Create the Azure Website

In order to deploy your website to windows azure you’re going to need to set-up an account. At [Windows Azure website](http://www.codenutz.com/azure)you can create an account for free and have a free trial. Furthermore, at this point in time you can set-up up to 10 azure websites for free – you cannot use custom domain names, ssl or several other features, but they are perfect for experimentation.

Onceyou have created you account head to [http://manage.windowsazure.com](http://manage.windowsazure.com/) where you can set-up your first website:

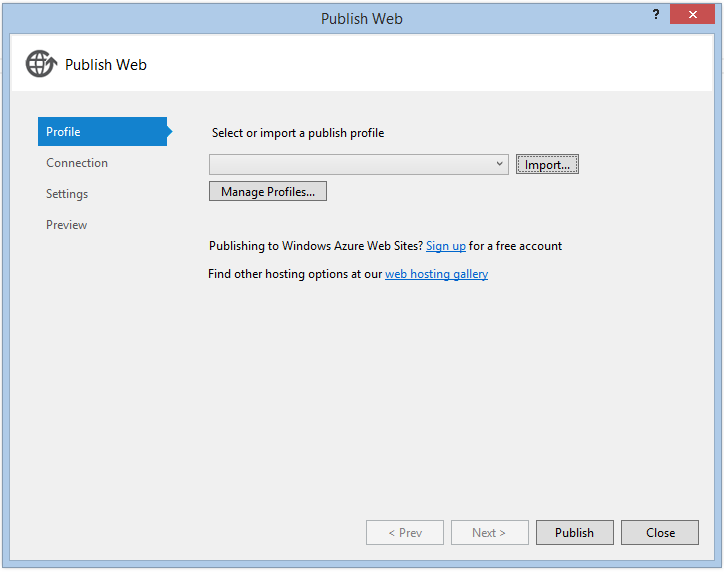
1. From the main screen click the new button in the bottom left corner   
   
2. The select
   1. **Compute>>Web Site>>Quick Create**
   2. Enter the name for your website, I used MyFirstWebAppForCodenutz (note that you have to choose a name thats available).
   3. Select a region where you want your site to be hosted, I chose West US (this isn’t important for this exercise)
   4. Press create website

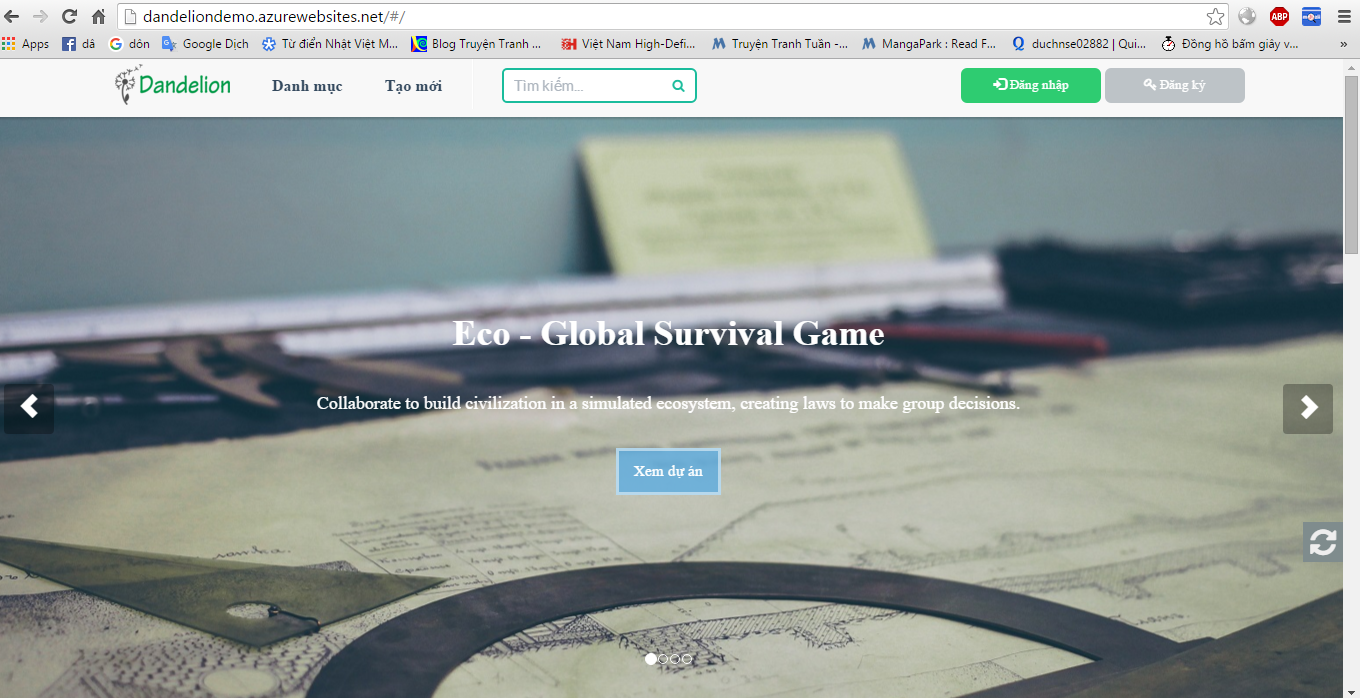


1. You’ll have to wait a few seconds for azure to spin up the website, but you will be able to see in the interface a green tick denoting that it is ready to use:   
   Azure website ready
2. Once it is ready click on it, and select ‘Download publish profile’ – this should download a file to your computer called something like MyFirstWebAppForCodenutz.azurewebsites.net.PublishSettings

Now you’ve got an azure website ready, we can get to publishing your mvc site to the azure service.

## Deploying to the Azure Website

1. From visual studio, right click the website project and choose the ‘Publish…’ option. You should be presented with a publish dialog like this:   
   
2. Choose ‘Import…’ – on this screen you can choose to import the profile directly from azure, or from a profile file. Either is very simple, but as we downloaded this file we’ll choose ‘Import from a publish profile file’. Click the browse button, select the file you just downloaded and click Ok. *This will prepopulate a number of fields in the publish dialog which for now we dont need to worry about.*
3. Finally just click the ‘Publish’ button, and you website will be deployed to windows azure. \*You should see the progress reported from within the visual studio output window with various bits of information like ‘Adding file…’ and ‘Adding ACLs’ etc. The first time you deploy tends to be the longest as all of the files have to be uploaded – subsequent deployments are much quicker as only the changed files are transferred.
4. Once your website has finished deploying, visual studio should automatically open it in your browser for you, which should look something like this:

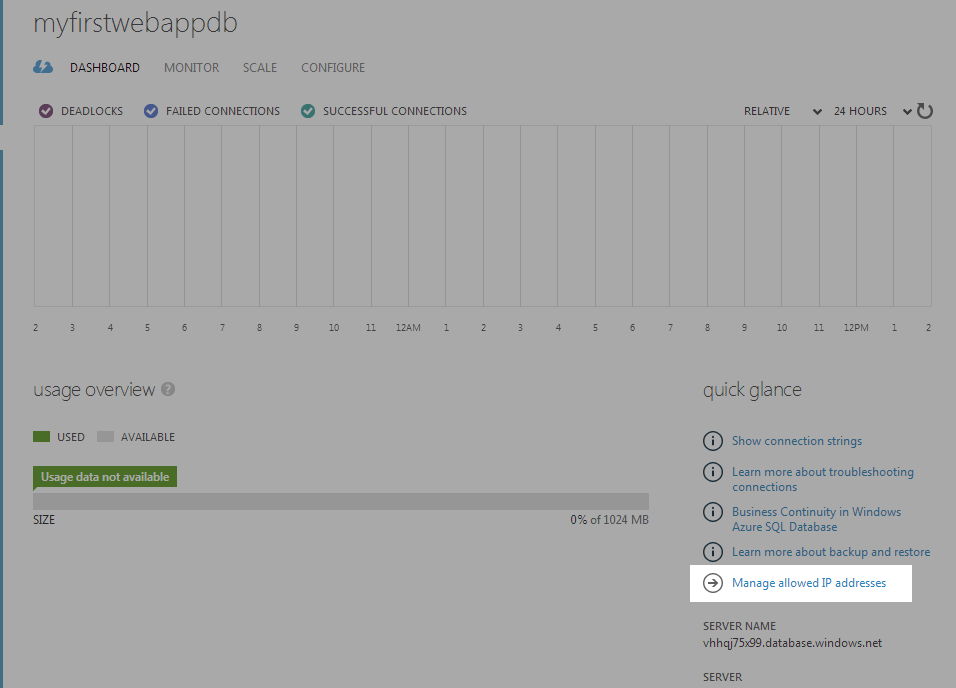


# Connecting SQL Management Studio to SQL Azure

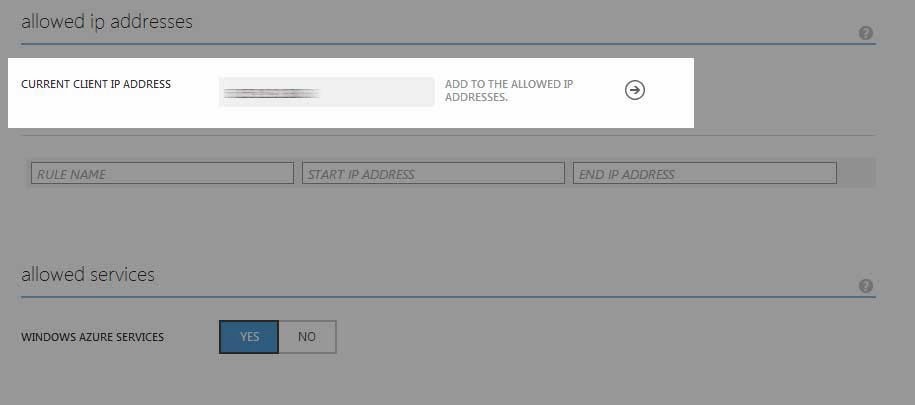
## Allowing access to SQL Azure

Here we’re going to set the SQL Azure fierwall to allow connections from our local IP.

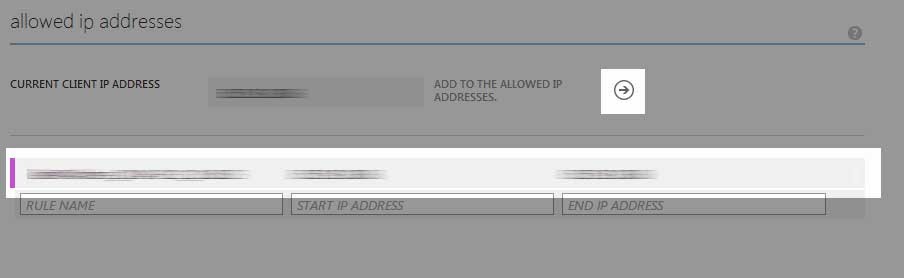
1. Navigate to your SQL Azure database dashboard (more detail instructions can be found in [Creating a SQL Azure Database](http://www.codenutz.com/mvc-web-app-with-sql-azure#creating-the-sql-azure-database).
2. Next go to ‘Manage allowed IP addresses’



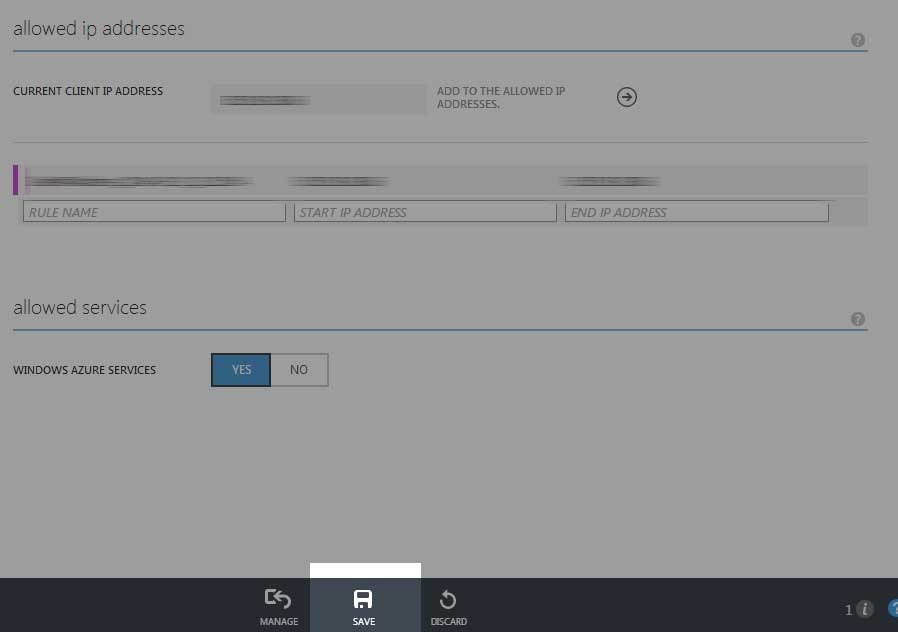
1. You should see on this screen that your IP address is listed as ‘Current client IP Address’. Here Azure has recognised your ip address and listed it for you so that you can easily add it to the firewall rules:



1. To add the firewall rule simple click right arrow and you should see the IP address move into the list of rules:



1. Finally click the Save button at the bottom of the page



## Connecting with SQL Server Management Studio

Connecting with SSMS should be pretty straight forward:

1. In the previous article we [discussed how to create and azure database](http://www.codenutz.com/mvc-web-app-with-sql-azure#creating-the-sql-azure-database). In this step we collected the ADO.NET connection string, and our logon credentials (username and password). You’ll need these for the next steps.
2. From the connection string you need to extract the server name. The server name part of the connection string should start with “tcp:” and end with “,1433”:

*The “tcp:” portion refers to the protocol used to establish the connection. The “,1433” is the port the connection will be made over.*

Server=\*\*{This is your server name}\*\*;Database=myfirstwebappdb;User ID=\*\*{your username is here}\*\*;Password=\*\*{your\_password\_here}\*\*;Trusted\_Connection=False;Encrypt=True;Connection Timeout=30;

1. Open SQL Server Management Studio. You should be prompted to connect to a server:   
   a. Enter your server name from the previous step in the server name box   
   b. Make sure you select “SQL Server Authentication” for the authentication option.   
   c. Enter your user name and password in the login and password boxes   
   d. Click connect   
   

You connection should now open up within SSMS.

## Troubleshooting

In my experience this works pretty well with one common issue – that being the configuration of the firewall on your development machine and/or your company network. If the crrect ports are not open your connection will fail generall with a message reading something like:

*Cannot connect to {your server name}*

*Additional Information*

*A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not foundor was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections.*

In order for you to be able to make this connection your firewall(s) must be configured to allow connections via TCP port 1433 – this is the default port sql server uses for TCP connections, and the one that SQL Azure uses as a standard.

## How to determine if port 1433 is open

A really simple way to test if it is the port that’s getting in the way, you can try to open a telnet session to the database server. To complete this your just need the server name as mentioned earlier, removing the protocol and port portions.

So if your server name from earlier looked like:

tcp:somecrazystring.database.windows.net,1433

You’d just need the part that says

somecrazystring.database.windows.net

Then follow these simple steps:

1. Open the command prompt (press the Windows key, then type ‘cmd’ and press enter)
2. Enter the following telnet somecrazystring.database.windows.net 1433
3. If the command prompt window turns blank then you have esxtablished the connection, and your port configuration is fine.
4. If the command prompt takes some time, and then outputs a message like Could not open connection to the host, on port 1433: Connect failed then there is some problem connecting via port 1433 and it is most likely some sort of firewall issue.