

## Abstract

This HOL will teach you how to publish a pre-existing web application to an Azure Web App. The deployed solution will contain the following components:

1. A deployment for a pre-existing ASP.Net web application.
2. An Azure SQL database to maintain usernames/passwords.
3. A worker role that handles items placed into an Azure Storage queue
4. A deployment slot that can be used for staging web app updates
5. Setup of auto-scaling for the web app

In this lab the focus will be on the use of the Azure Preview portal and it's components. You will learn how to setup the structure of the environment through the Azure Preview portal and then deploy the application with Visual Studio.

Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarked, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, MS, Windows, Windows NT, MSDN, Active Directory, BizTalk, SQL Server, SharePoint, Outlook, PowerPoint, FrontPage, Visual Basic, Visual C++, Visual J++, Visual InterDev, Visual SourceSafe, Visual C#, Visual J#, and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.

## Contents

### Publishing an Azure Web App

Exercise 1 – Publishing a pre-existing web application to an Azure Web App

Prerequisites

Task 1 – Create a Storage Account

Task 2 – Create the Azure Web App

Task 3 – Retrieve your Storage account keys

Task 4 – Prepare the FixIt application for Azure

Task 5 – Publish the Worker Role into Azure

Task 6 – Publish the MyFitIt Web Application into an Azure Web App

Exercise 2 – Using Azure Web App Settings

Exercise 3 – Deploy the Web App to a Deployment Slot

Task 1 – Creating the Staging deployment slot

Task 2 – Publishing the web app to the Staging slot

Exercise 3 – Implementing Auto-Scaling for the Azure Web App

Task 1 – Setup Auto-Scaling using the Azure Portal

Task 2 – Changing the web server capacity

# Publishing an Azure Web App

## **Exercise 1 – Publishing a pre-existing web application to an Azure Web App**

In this lab, you will publish a pre-existing application named FixIt, to a Microsoft Azure Web App (website). This web app will use the following Microsoft Azure features:

1. Azure Web App
2. Azure Storage
3. Azure SQL Database

The purpose of this lab is not to teach you how to write web code, but instead how to take a project that you already have on-premises and deploy that into an Azure Web App configuration.

You'll learn:

- How to create an initially empty Azure Web App
- How to publish the web app in Azure
- How to setup the configuration options to use Azure storage
- How to setup the configuration options to use Azure SQL Database

## Prerequisites

The following is required to complete this hands-on lab:

- Microsoft Visual Studio 2013 Professional or Ultimate edition with Update 4
- Microsoft Azure SDK for .NET (VS 2013) - 2.6
- A Microsoft Azure subscription

## Logging in to the Virtual Machine

This lab is completed using virtual machines that run on Windows 8.1® in a Hyper-V™ environment. To log on to the virtual machines, press CTRL+ALT+END and enter your logon credentials.

Login Credentials:

Username – azstudent

Password – Pass@word1

## Task 1 – Create a Storage Account

Azure Storage can take on many forms for storing blobs, non-relational table data and queue data. In the case of this application, we will be using both blob and queue storage. For this, we need to create a new Azure Storage account.

Where possible, we will be using the new (preview) Azure portal for performing most of these lab exercise.

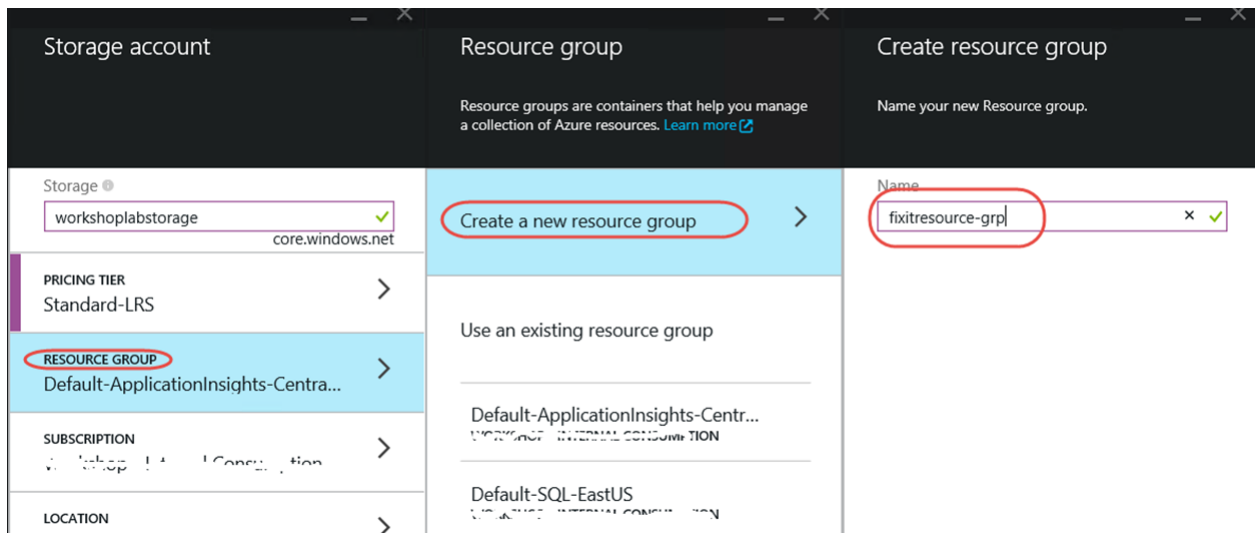
1. Log in to the Azure Preview Portal at <https://portal.azure.com>.
2. Click the **New+** button and select **Create->Data + Storage ->Storage** and then enter a unique name for the storage account (Don't select the **Create** button yet). The screen that you see in front of you will keep expanding to the right as you progress through the wizard. These individual sections that appear are known as 'blades'.

3. Select the **Pricing Tier** link and select the **L-Locally Redundant** selection. Click the **Select** button and the blade will close.

The screenshot shows the 'Storage account' configuration page in the Azure portal. The left sidebar contains configuration options: 'Storage' (with a text input 'core.windows.net'), 'PRICING TIER' (highlighted with a red circle and a right arrow), 'RESOURCE GROUP' (Default-SQL-EastUS), 'SUBSCRIPTION' (Azure Pass), 'LOCATION' (East US), and 'DIAGNOSTICS' (Not configured). The main area is titled 'Choose your pricing tier' with the subtitle 'BROWSE THE AVAILABLE PLANS AND THEIR FEATURES'. It displays two pricing tiers: 'L Locally Redundant' (selected with a blue checkmark and a red circle) and 'G Geo-Redundant'. Both tiers show '3 Local replicas' and '3 Geo-distributed rep'. The 'L Locally Redundant' tier is highlighted with a blue box and lists features: 'Block and page blobs', 'Table', 'Queue', '500 Max IOPS per disk', and '99.9% SLA'. The 'G Geo-Redundant' tier also lists these features. At the bottom of each tier's details, it says 'LOADING PRICING INFO...'.

PRICING TIER	Local replicas	Geo-distributed rep
Standard-GRS	3	3
Block and page blobs	Block and page blobs	Block and page blobs
Table	Table	Table
Queue	Queue	Queue
500 Max IOPS per disk	500 Max IOPS per disk	500 Max IOPS per disk
99.9% SLA	99.9% SLA	99.9% SLA

4. Click on the **Resource Group** link, then the **Create a new resource group** and then give the resource group a unique name. Choose the **Ok** button at the bottom of the resource group naming blade.



A Resource Group is an entity in the new Azure Preview Portal that is used to contain all of the resources associated with the application you will be deploying. This gives you a single view into all associated aspects of the application. You can find out more about them here <http://azure.microsoft.com/en-us/documentation/articles/azure-preview-portal-using-resource-groups/>

5. If you have multiple subscriptions, select the appropriate subscription by using the **Subscription** link. Also set your Location to the data center of your choosing. Choose the **Select** button at the bottom of the blade (not shown below). Wait for the storage account to be created (you will see an animated icon in the portal window that represents the storage account being created).

Storage account

Storage ⓘ

workshoplabstorage ✓

core.windows.net

PRICING TIER

Standard-LRS

>

RESOURCE GROUP

fixitresource-grp

>

SUBSCRIPTION

Microsoft Azure Subscription

>

LOCATION

East US

>

DIAGNOSTICS

Not configured

>

☒ Add to Startboard

Create



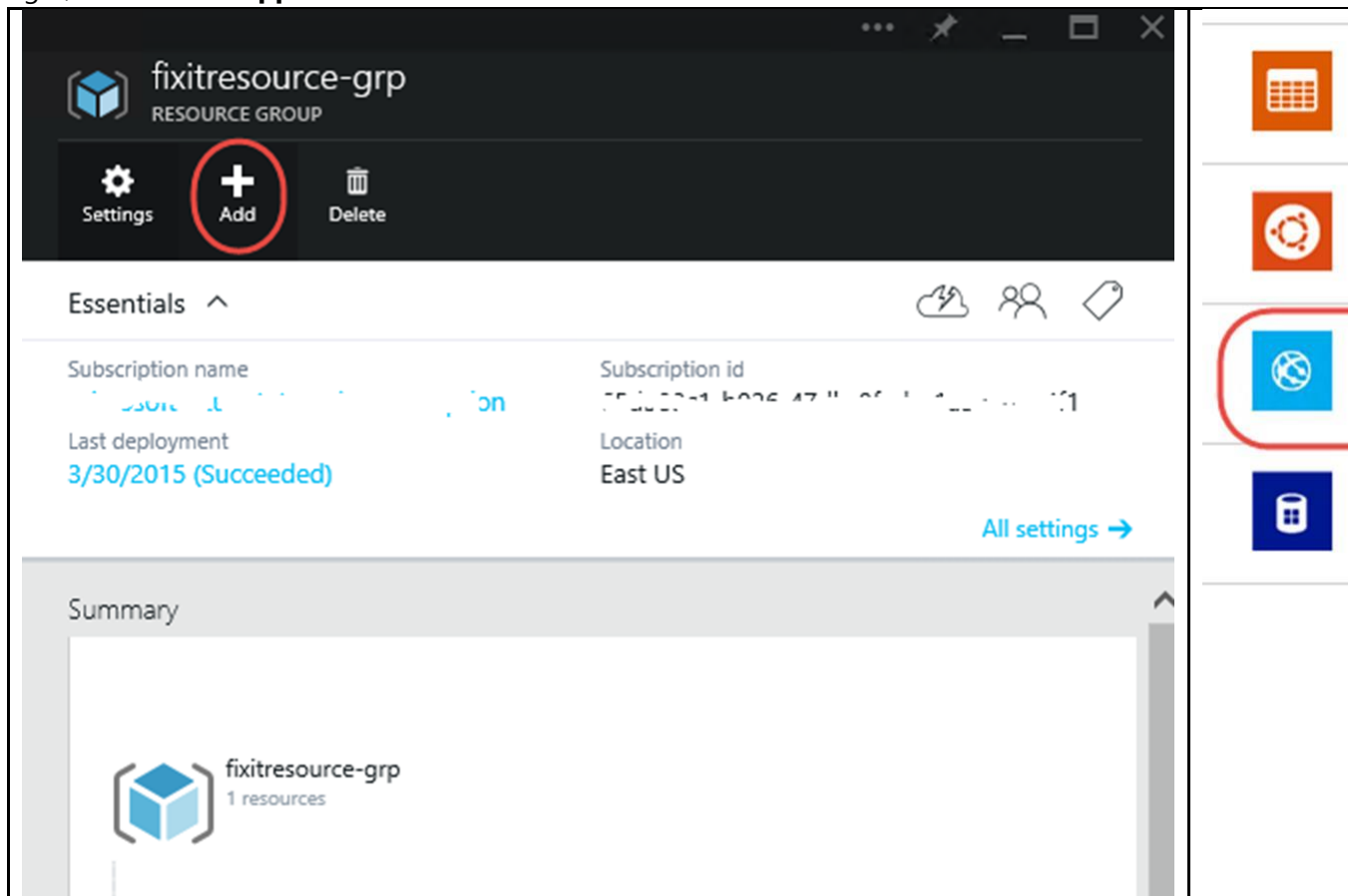
## Task 2 – Create the Azure Web App

In this task, you will create the web app environment. Note that this web app is empty of any content, we will publish the source code into the web app in a later task.

1. Click on the **Browse** button on the left side of the screen.
2. Select **Resource Groups** and then select the name of the resource group you created in the last step.

Reserved IP Addresses	fixitresource-2	Microsoft Azure Internal Consumption	East US
Resource groups	fixitresource-2	Microsoft Azure Internal Consumption	East US
Search services	fixitresource-2	Microsoft Azure Internal Consumption	East US
SQL Databases	sqlvm-cs	Microsoft Azure Internal Consumption	East US
SQL Servers	vmix-cs	Microsoft Azure Internal Consumption	East US
Storage Accounts	fixitresource-grp	Microsoft Azure Internal Consumption	East US

- When the resource group blade appears, select the **Add** button and then in the list on the right, choose **Web app**.



- Click on the **Create** button.

5. Enter a unique URL for the Azure Web App and also the name for a new **AppService Plan**.

Web app

URL  
myFixItA ✓  
.azurewebsites.net

CREATE NEW APPSERVICE PLAN ⓘ  
mySimpleWebHostPlan ✓  
[Or select existing](#)

PRICING TIER >  
S1 Standard

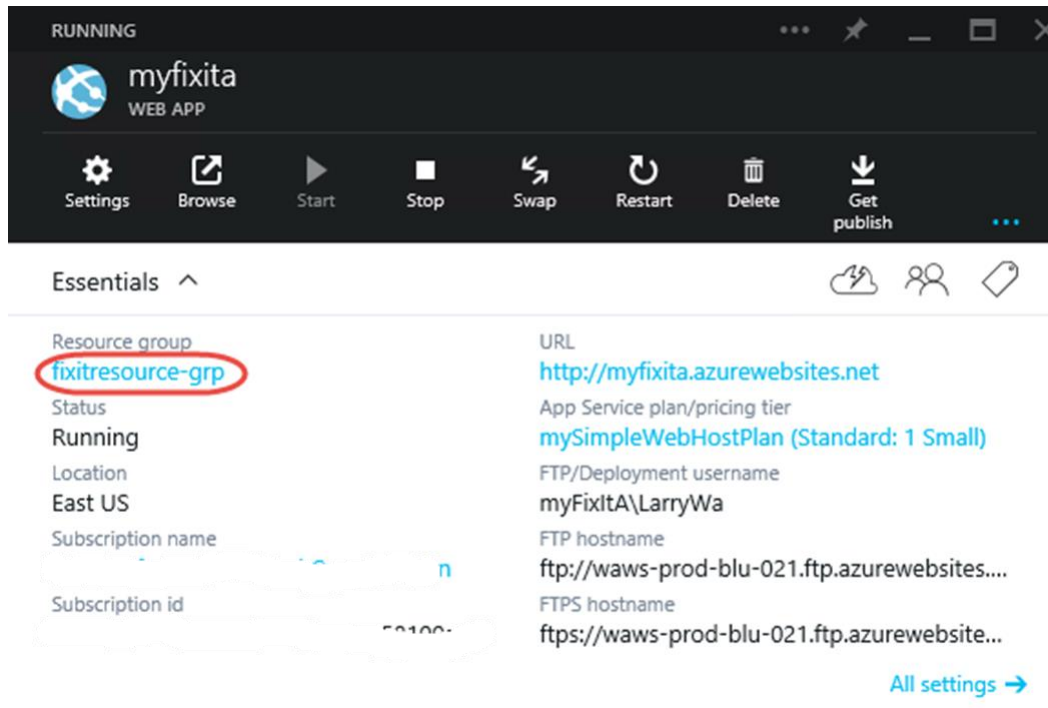
RESOURCE GROUP ⓘ >  
fixitresource-grp  
[Or create new](#)

SUBSCRIPTION >  
[Subscription Name]

LOCATION >  
East US

6. For the Pricing Tier, leave the setting as is (S1 Standard) and make sure the Location is in the same region where you created your storage account. Leave the rest the settings as is. Select the **Create** button at the bottom of the blade. Wait until the web app is created in the portal.

7. Within the web app blade, select the **resource group** that the web app was just created in. (You will see the blade like the screenshot below after the web app has completed creation)



8. Within the Resource Group, click on the **Add** button and then select **SQL Database**. Select the **Create** button.

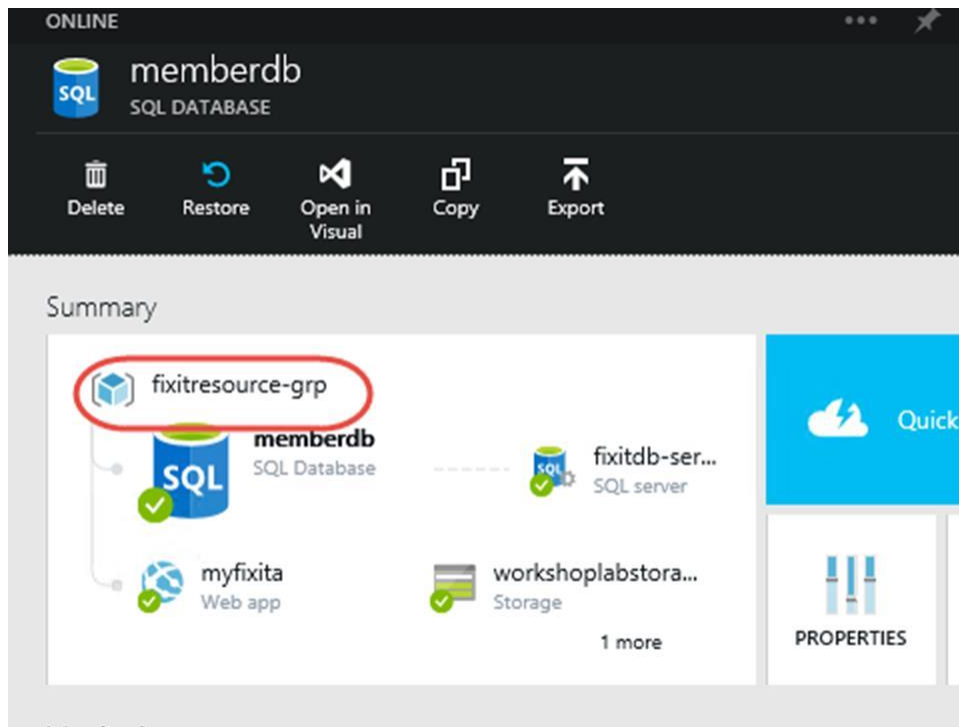
9. In the SQL Database blade, perform the following steps IN THIS ORDER:
- Select the subscription you will be using.
  - Select your resource group if not already selected to the correct group.
  - Enter the name **memberdb** in the **Name** field.
  - Select the **Server** link (to create a new server), select **Create a new server** and then enter your unique server name, login username/password and MAKE SURE you select the same data center you did for your web app.
  - Select the **OK** button at the bottom of the New Server blade.

The screenshot displays the Azure portal interface for configuring a new SQL Database. It is divided into three main sections: SQL Database, Server, and New server.

- SQL Database:** This section contains configuration options for the new database. The 'Name' field is set to 'memberdb' and is highlighted with a red box. Below it, the 'SERVER' link is highlighted with a red box, showing the option to 'Configure required settings'. Other options include 'SELECT SOURCE' (Blank Database), 'PRICING TIER' (Standard S0), 'OPTIONAL CONFIGURATION' (Collation), 'RESOURCE GROUP' (labwebsitegroup), and 'SUBSCRIPTION' (Azure Pass).
- Server:** This section shows the 'Create a new server' button, which is highlighted with a red box. Below it, the 'Use an existing server' section lists available servers, including 'East US Default-SQL-EastUS V12'.
- New server:** This section contains the configuration details for the new server. The 'SERVER NAME' field is set to 'fixitdb-server' and is highlighted with a red box. Below it, the 'SERVER ADMIN LOGIN' field is set to 'fixitdb-admin' and is highlighted with a red box. The 'PASSWORD' and 'CONFIRM PASSWORD' fields are both set to '\*\*\*\*\*' and are highlighted with a red box. The 'LOCATION' is set to 'East US' and is highlighted with a red box. The 'CREATE V12 SERVER (LATEST UPDATE)' checkbox is checked, and the 'ALLOW AZURE SERVICES TO ACCESS SERVER' checkbox is also checked.

10. On the SQL Database blade, leave everything else as is and select the **Create** button. Wait until the SQL Server has completed being created.

11. Once the database has been created, once again, click on the **Resource Group** link.



12. In the Resource group blade, select the **Add** button and create a new **SQL Database**. Follow the same sequence of steps you did in step 9 (where you started by setting your subscription name, resource group name first).
13. Give the database the name **appdb** and select to use the previously created database server. Select the **Create** button at the bottom of the SQL Database blade.

SQL Database

Name  
appdb ✓

SERVER  
fixitdb-server (East US) >

SELECT SOURCE ⓘ  
Blank Database >

PRICING TIER ⓘ  
Standard S0 >

OPTIONAL CONFIGURATION ⓘ  
Collation >

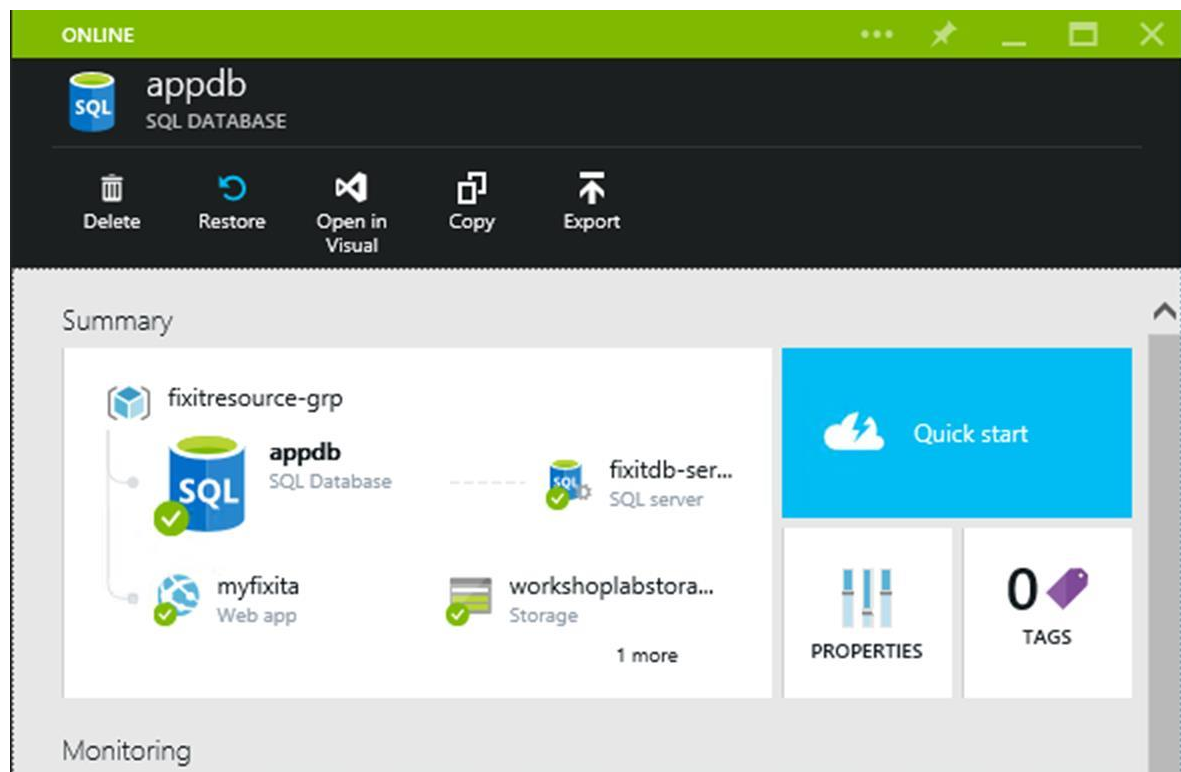
RESOURCE GROUP  
fixitresource-grp 🔒

SUBSCRIPTION  
p... 🔒

14. After the appdb database has been created, you should see your SQL databases, web app, storage account and your SQL server in the resource group.

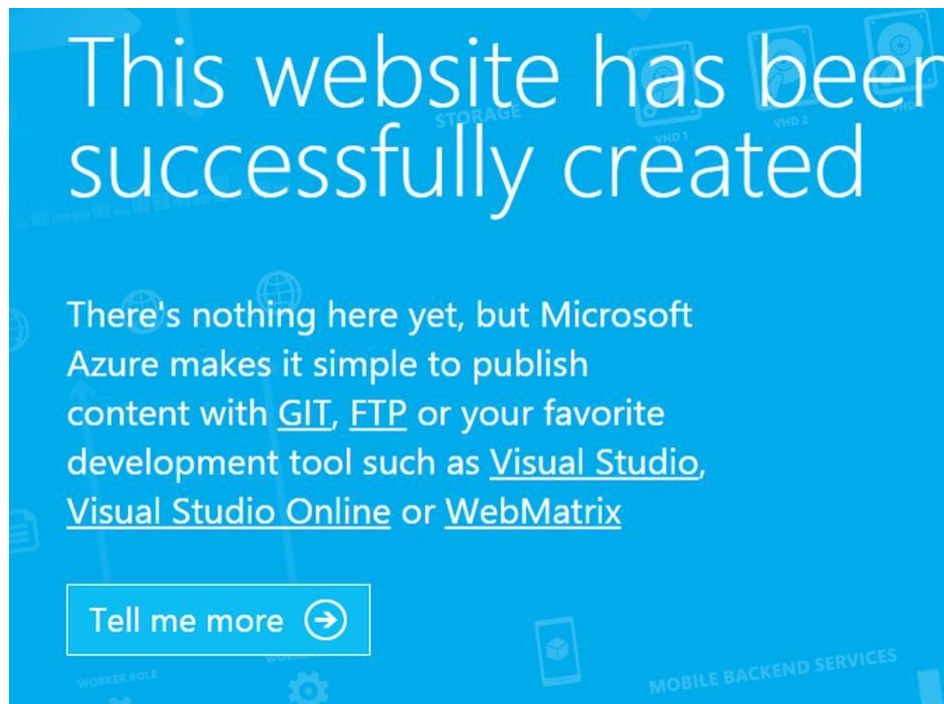
Because this portal is in Preview, you may not be able to see your SQL databases visually in the resource group. But don't be concerned, it's all there and any reference from the web app to the database will still succeed. The resource group is a logical grouping of all components.

If you want to make sure your databases have been created, you can go back to the main portal window, select Browse and then search for both databases and the database server.

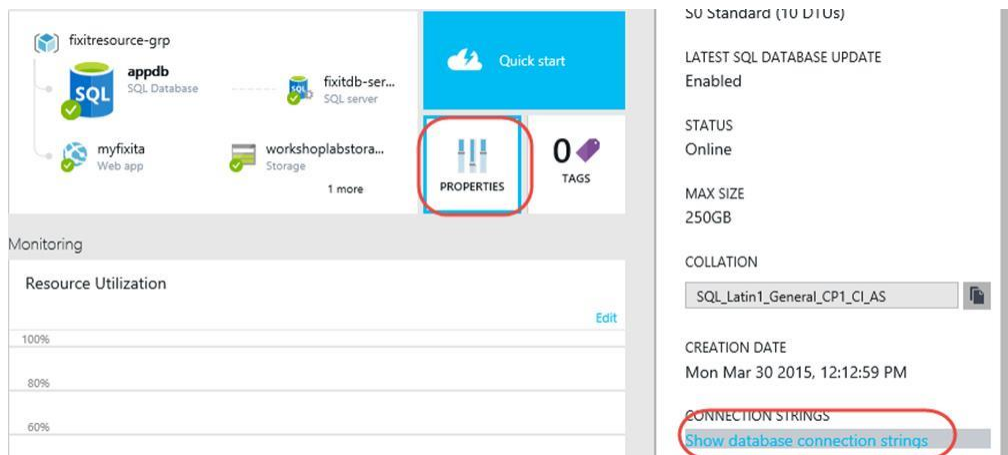





15. To assure that the empty web app has been created, select your web app icon in the resource group. When the web app blade appears, select the **Browse** button and you should see something like:



16. Go to the **appdb** database blade (you may need to use the Browse button from the main menu). We want to copy the connection string from the portal for later use in the application setup. Click on the Properties icon. Click on the Show database connection strings link and copy



17. Copy the ADO.Net connection string and paste it into Notepad.

 Database connection strings  
APPDB

ADO.NET

Server=tcp:localhost:1433;Database=appdb;User ID=sa;Password=;@h

ODBC (Includes Node.js)

Driver={SQL Server Native Client 11.0};Server=tcp:localhost:1433;Database=appdb;User ID=sa;Password=;@h

PHP

Server=localhost:1433 \r\nSQL Database: appdb\r\nUser Name: sa;Password=;@h

JDBC

jdbc:sqlserver://localhost:1433;database=appdb;user=sa;password=;

### Task 3 – Retrieve your Storage account keys

Each storage account contains 512 bit private keys that can be used to secure the storage account. You will need to copy the primary key of your storage account to Notepad for later use in the lab configuration files.

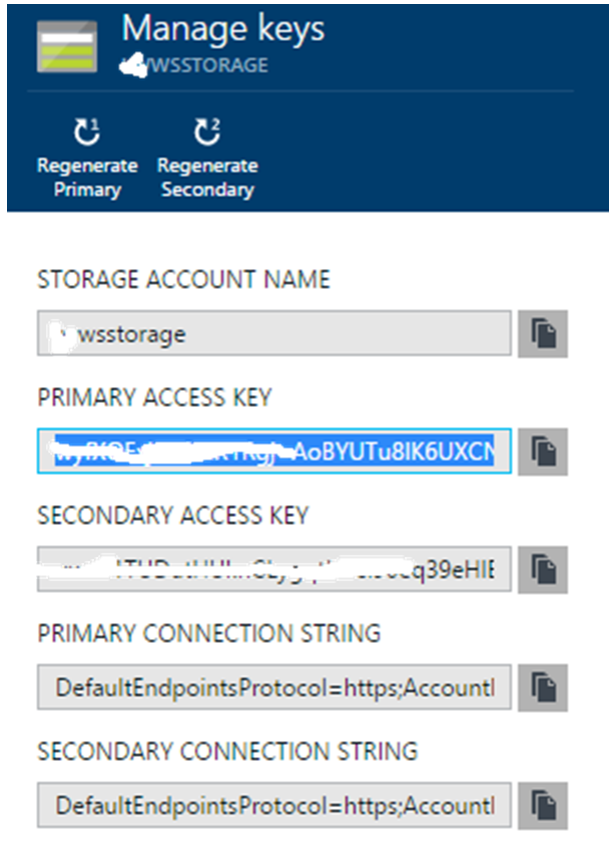
1. Click the **Browse** button located on the left hand side of the portal screen and then **Storage Accounts**.

The screenshot shows the Azure portal interface. On the left sidebar, the 'Storage Accounts' option is highlighted with a red circle. The main pane displays a table of storage accounts with the following data:

ACCOUNT	STATUS
myteststorage	Online
myteststorage	Online
myteststorage	Online
myteststorage	Online

2. Click on the name of the storage account that you created earlier and then the **Keys** icon.

3. Copy the **Primary Access Key** by selecting the **All Settings->Keys** menu item and then close the blade by selecting the 'X' button in the upper right of the blade. Paste the key into Notepad. You can continue to close all the blades associated with storage by clicking on their 'X' buttons. This will take you back to the main portal view.



The screenshot shows the 'Manage keys' blade for a storage account named 'wsstorage'. The blade has a dark blue header with the title 'Manage keys' and the 'WSSTORAGE' logo. Below the header, there are two buttons: 'Regenerate Primary' and 'Regenerate Secondary'. The main content area lists six fields, each with a text input and a copy icon:

- STORAGE ACCOUNT NAME**: wsstorage
- PRIMARY ACCESS KEY**: wY8Q5...AoBYUTu8IK6UXCN (The key is highlighted in blue)
- SECONDARY ACCESS KEY**: ...ITUD...q39eHIE
- PRIMARY CONNECTION STRING**: DefaultEndpointsProtocol=https;AccountI
- SECONDARY CONNECTION STRING**: DefaultEndpointsProtocol=https;AccountI

## Task 4 – Prepare the FixIt application for Azure

1. Open Visual Studio 2013 as an administrator and select **File->Open->Project Solution**....browse to/open the **C:\Labs\AZRHOL302-AzureWebsites\Source\Exercise1\Begin\MyFixIt.sln**.
2. Right click on the **MyFixIt** solution and select **Rebuild Solution**. Make sure there are no errors (do not be concerned about any warnings that appear).
3. Open the **app.config** file of the **MyFixIt.WorkerRole** project and put in your storage account name and key.

```
<add key="StorageAccountName" value="{StorageAccountName}" />
```

```
<add key="StorageAccountAccessKey" value="{StorageAccountAccessKey}" />
```

4. In the **app.config** file of the **MyFixIt.WorkerRole**, replace the **appdb** connection string with the following, providing the settings from your own environment. Recall that in previous steps you copied this connection string out of the portal and pasted it into NotePad. Notice that if you take the connection string from Notepad and paste into the config file, you still need to make some modifications to it (remove tcp:, ,1433 etc)

```
<add name="appdb" connectionString="Server=tcp:[Your DB Server  
name].database.windows.net,1433;Database=appdb;User  
ID=[login_user_name@server_name];Password=[Your  
Password];Trusted_Connection=False;Encrypt=True;Connection  
Timeout=30;" providerName="System.Data.SqlClient" />
```

5. Open the **app.config** file of the **MyFixIt.Persistence** project and put the worker role name '**MyFixIt.WorkerRole**' into the cache.

```
<dataCacheClients>
  <dataCacheClient name="default">
    <!--To use the in-role flavor of Windows Azure Cache, set identifier to be the cache
    cluster role name -->
    <!--To use the Windows Azure Cache Service, set identifier to be the endpoint of the
    cache cluster -->
    <autoDiscover isEnabled="true" identifier="MyFixIt.WorkerRole" />
  </dataCacheClient>
</dataCacheClients>
```

6. Open the **web.config** file of the **MyFixIt** web project and put in the storage account name, key and then set the *AppSettings->UseQueues* setting to **True**. Also set the AppSettings for the **StorageAccountName** and **StorageAccountAccessKey** to the values you used in the app.config file of the MyFixIt.WorkerRole.

```
<add key="StorageAccountName" value="{StorageAccountName}" />
<add key="StorageAccountAccessKey" value="{StorageAccountAccessKey}" />
```

7. In the web.config file, replace the following connection strings:

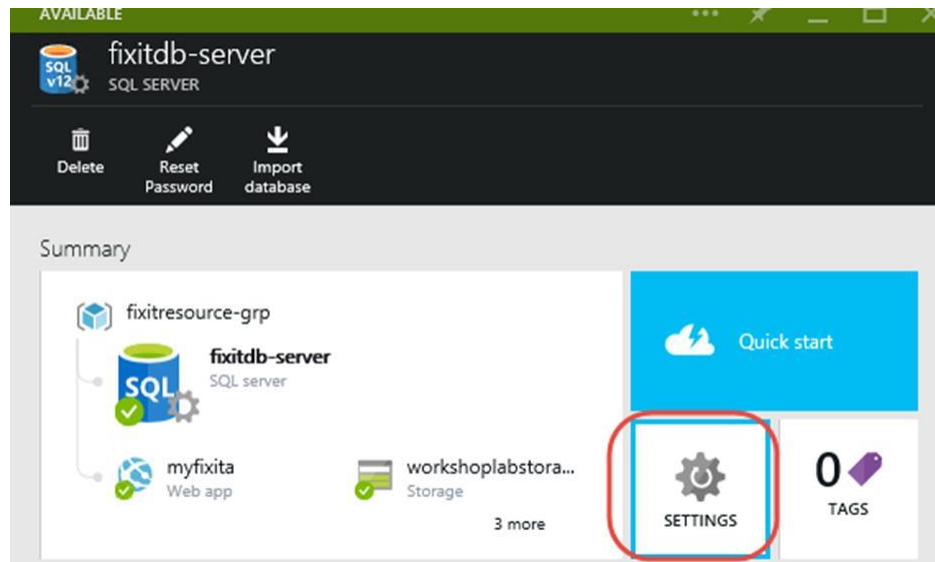
```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data
  Source=(LocalDb)\v11.0;Initial
  Catalog=MyFixItMembership;Integrated Security=True"
  providerName="System.Data.SqlClient" />
  <add name="appdb" connectionString="Data
  Source=(LocalDb)\v11.0;Initial Catalog=MyFixItTasks;Integrated
  Security=True" providerName="System.Data.SqlClient" />
</connectionStrings>
```

With these connection strings

```
<connectionStrings>
  <add name="DefaultConnection"
  connectionString="Server=tcp:[Your_Database_Server_Name].database
  .windows.net,1433;Database=memberdb;User
  ID=[Your_User_Name@Your_Server_Name];Password=[Your_Password];Tru
  sted_Connection=False;Encrypt=True;Connection Timeout=30;"
  providerName="System.Data.SqlClient" />
  <add name="appdb" connectionString="Server=tcp:
  [Your_Database_Server_Name].database.windows.net,1433;Database=ap
  pdb;User
  ID=[Your_User_Name@Your_Server_Name];Password=[Your_Password];Tru
  sted_Connection=False;Encrypt=True;Connection Timeout=30;"
  providerName="System.Data.SqlClient" />
```

</connectionStrings>

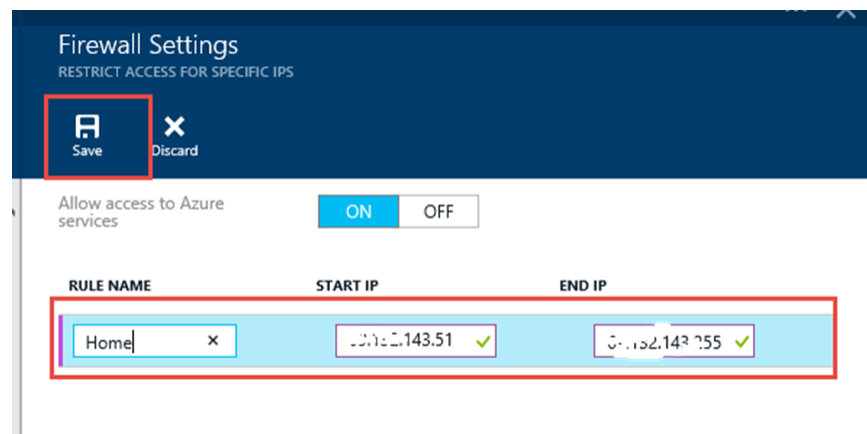
8. In order to test the app locally (meaning, before you deploy the web app to Azure), we will need to setup a firewall rule in your Azure SQL Database server so that it will allow our client IP address to access it. Go back to the Azure Preview portal and select **Browse->SQL Servers->[your db server name]** and then click on the **Settings** icon. Note that your screen may appear a bit different than what you see below due to preview changes in the preview portal.





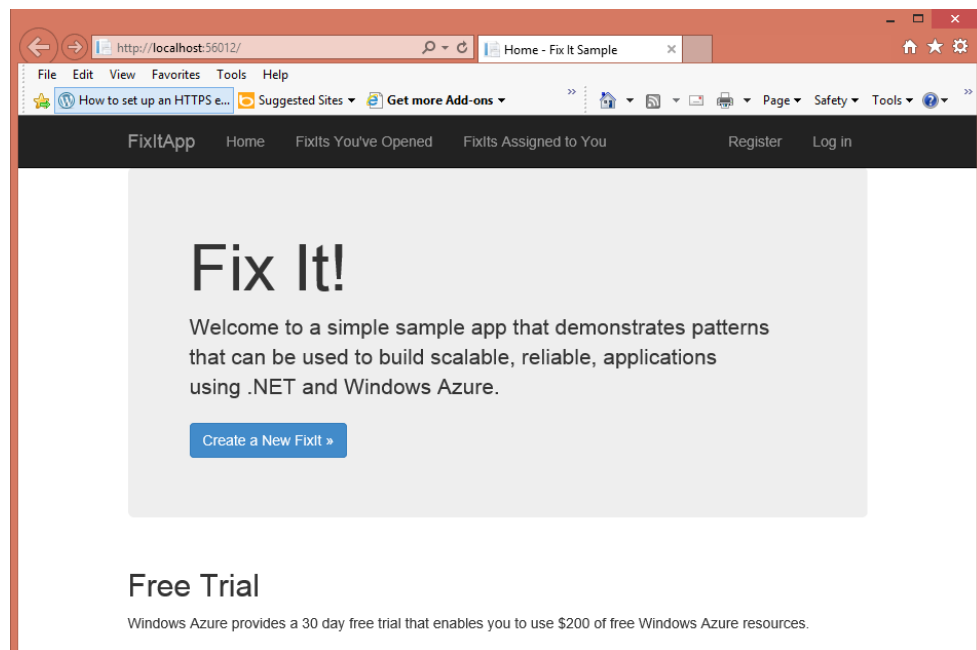
9. Click the **Firewall** link and then Add a new rule (name it whatever you want) and the IP address of your machine and select the **Save** button.

The IP address will be a public IP address, not an internal IP address (like 172.16.. or 192.168.. etc). If you do not know the external IP address you can go into the current portal (not the preview portal) and add a firewall rule. The current portal can detect your public IP address, whereas the preview portal cannot.



NOTE: Sometimes it can be difficult to figure out what the correct IP address is since your machine may be going through proxies to get to Azure. In a dev environment, it is almost easier to just try to access the database with the app and let it fail, and the error will show what IP address did not get through. From that, you can use that as a starting IP address and then for the ending IP address just set the last octet to .255. Note that sometimes it can take up to 3 minutes for a firewall rule to take effect in Azure.

10. The **MyFixIt** web project is currently selected as the Startup project. We want to test this web project locally just to make sure it runs with no errors. Press **F5** to run the web application. We are not doing a complete functionality test. What you should see if everything works correctly is:

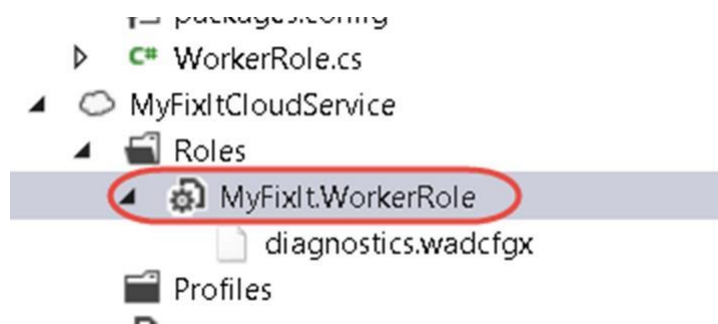


11. Close the web browser.

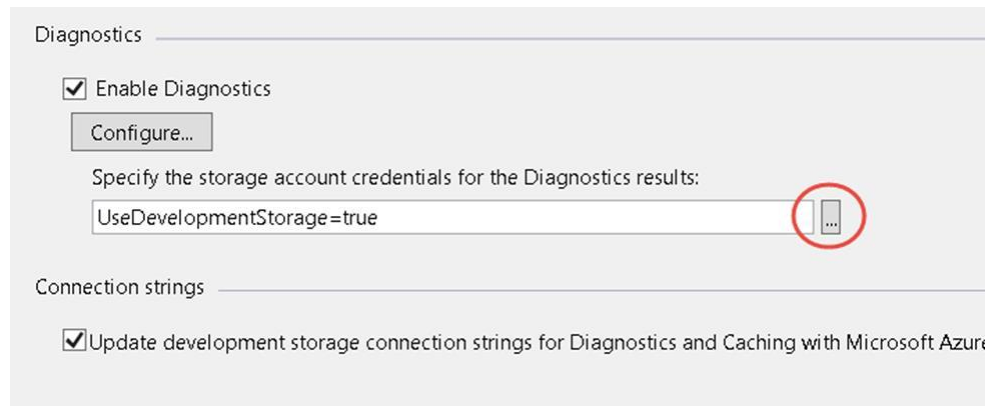
## Task 5 – Publish the Worker Role into Azure

We will first publish the **MyFixItCloudService** into an Azure Cloud Service. This will deploy our MyFixIt.WorkerRole that is used to pick up messages out of our Azure storage queue. This worker role also serves as our in-role cache.

1. In the **MyFixItCloudService** project, double click on the MyFixIt.WorkerRole role to bring up the role properties.



2. In the role properties window, click on the ellipse button for setting the diagnostics storage account location.



3. Click on the **Your Subscription** choice button and select your subscription and previously created storage account. If you have not signed in to Azure yet, you can do so from here. Select the **OK** button when finished.

Create Storage Connection String

Connect using:

- ☐ Microsoft Azure storage emulator
- ☒ Your subscription
- ☐ Manually entered credentials

Select a subscription and a storage account associated with it.

Signed in as: [email address]@gmail.com

Subscription:

[Subscription Name]

Account name:

[Account Name] storage (East US)

☒ Use HTTPS (Recommended)

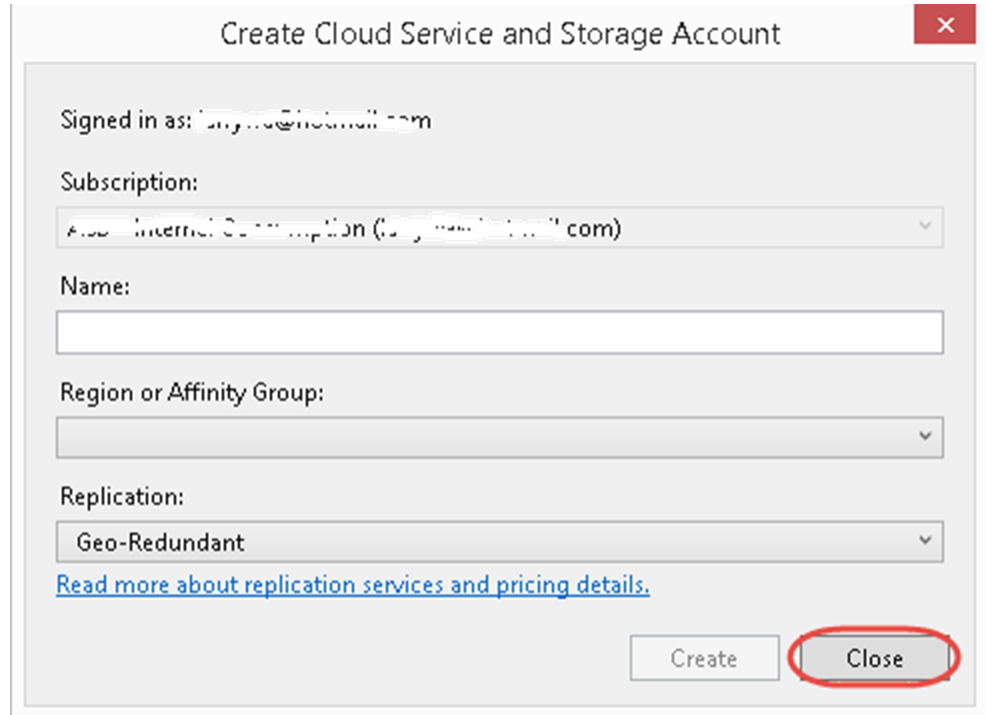
Preview connection string:

```
DefaultEndpointsProtocol=https;AccountName=[Account Name] storage;AccountKey=[Account Key]...;UseSSL=true
```

[Online privacy statement](#) OK Cancel

4. Right click on the **MyFixItCloudService** and select **Publish**.
5. You will be presented with a publish dialog box where you will need to log in to Azure (you should be prompted). Once you log in, you **may** be presented with a dialog box titled 'Create Cloud Service and Storage Account'. Click Close to close this dialog box. Move to step 3 if you do not see this dialog box appear.

Why are you closing this dialog box? Because, this dialog box wants you to create a additional storage account. You already have a storage account and we will use another process to make sure this deployment uses the correct storage account.



Create Cloud Service and Storage Account

Signed in as: anyone@hotmail.com

Subscription:  
Free Internet Connection (for home use)

Name:

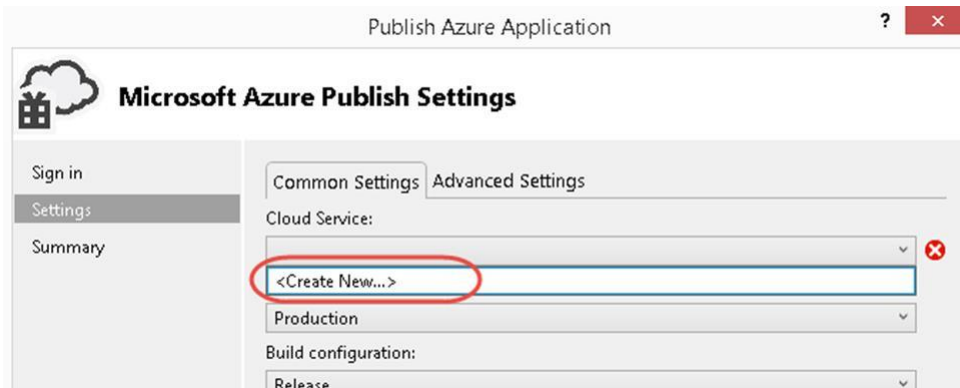
Region or Affinity Group:

Replication:  
Geo-Redundant

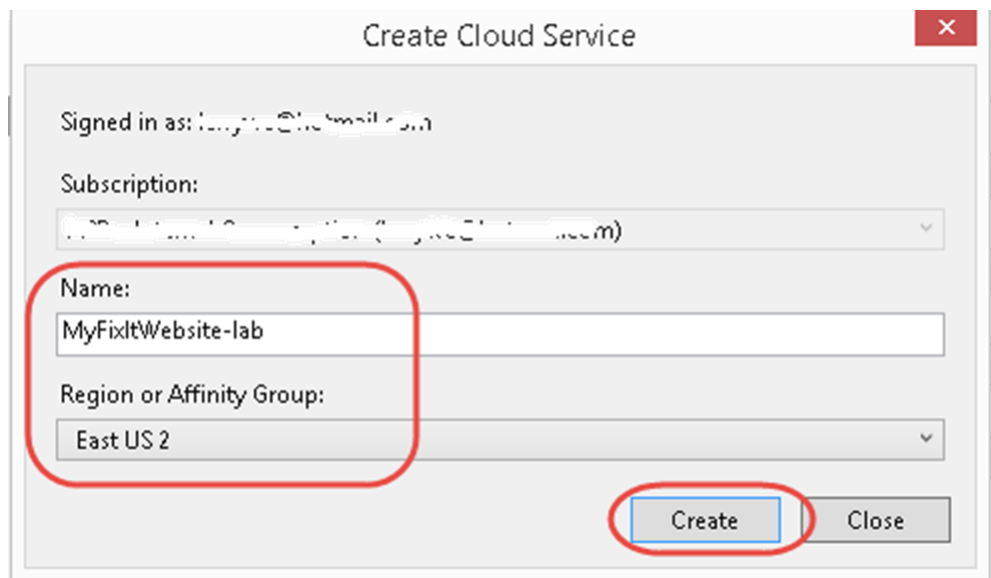
[Read more about replication services and pricing details.](#)

Create Close

6. In the **Common Settings** tab of the Publish Azure Application wizard, select the **Cloud Service** drop-down and then select the **<Create New>** link.



7. Enter a unique name for your Cloud Service, then select the data center/region. Select **Create**.



8. Select the **Advanced** tab and make sure the correct storage account is being used for this deployment. This will be the storage account you created earlier. Then click the **Next>** button.

### ft Azure Publish Settings

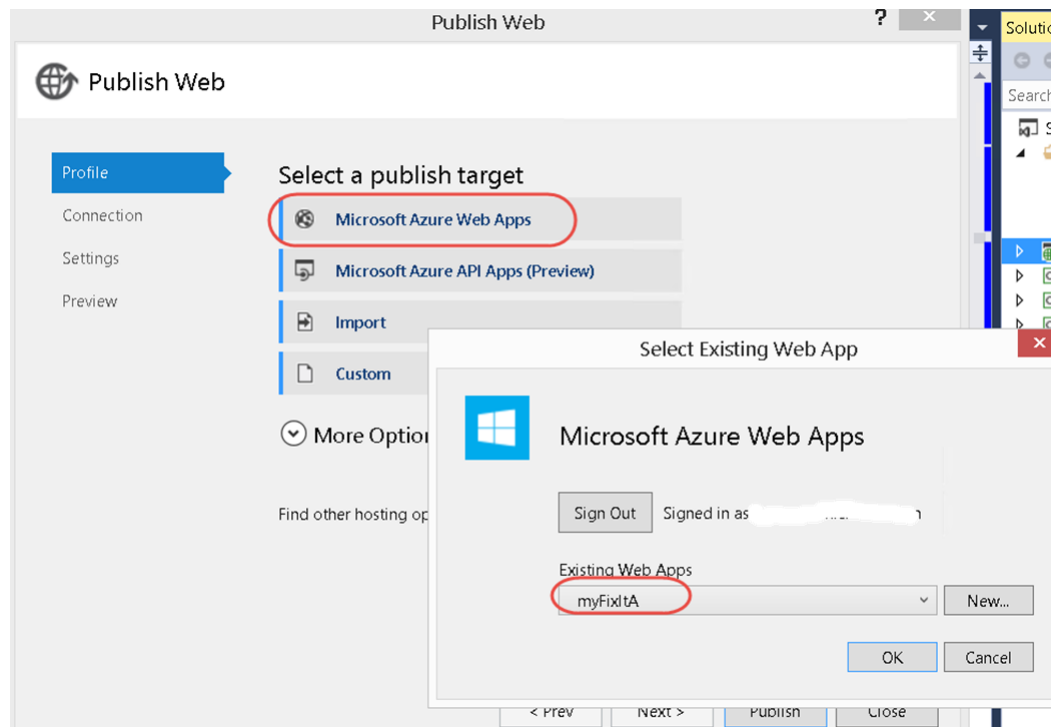
The screenshot shows the 'Advanced Settings' tab of the Azure Publish Settings dialog. The 'Deployment label' is 'MyFixItCloudService'. The 'Append current date and time' checkbox is checked. The 'Storage account' dropdown is set to 'workshoplabstorage (East US)'. Other options include 'Delete deployment on failure' (unchecked), 'Deployment update' (checked), 'Enable IntelliTrace' (unchecked), 'Enable profiling' (unchecked), and 'Enable Remote Debugger for all roles' (unchecked). At the bottom, the 'Next >' button is highlighted with a red circle.

9. Click the **Publish** button. The publish operation can take up to 10 minutes to complete but you can watch the progress from within Visual Studio.

The screenshot shows the Microsoft Azure Activity Log interface. The 'Deployment' tab is selected, showing a list of activities. The first activity is 'Deploying to labmyfixitworker - Production'. The 'Production' section shows the 'Website URL' as 'http://labmyfixitworker.cloudapp.net/' and the 'Deployment ID' as '1f80e85dc97447ada3c4ec2b15ed3511'. The 'History' section shows a list of events: '2:40:10 PM - Instance 0 of role MyFixIt.WorkerRole is stopped', '2:40:10 PM - Starting...', '2:40:26 PM - Initializing...', '2:40:26 PM - Instance 0 of role MyFixIt.WorkerRole is in an unknown state', and '2:40:58 PM - Instance 0 of role MyFixIt.WorkerRole is creating the virtual machine'.

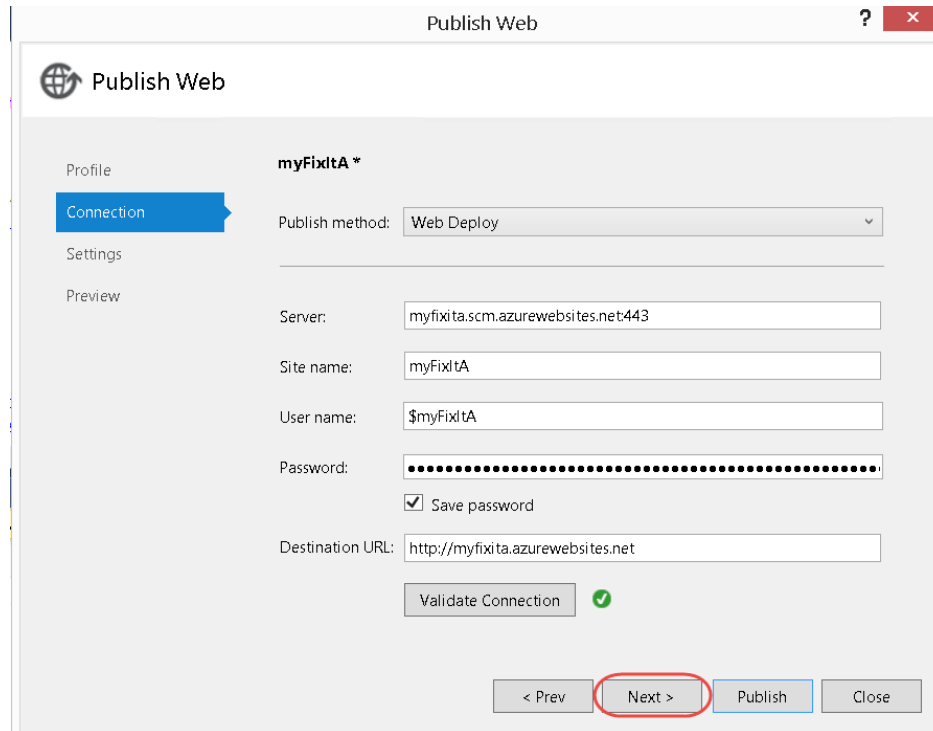
## Task 6 – Publish the MyFitIt Web Application into an Azure Web App

1. While the worker role is still in the process of publishing, right click on the **MyFitIt** web project and select **Publish**. (You can do the web app publish and the worker role publishing at the same time, but if you try to run the web app and it looks for the worker role it will fail).
2. You will be presented with the Publish dialog box where you will either have to log in to Azure, or you can just click on the **Microsoft Azure Web Apps** link and then choose the name of the web app you created earlier. Then select **OK**.





3. On the Connection page, select **Validate Connection** and then **Next>**.



The screenshot shows the 'Publish Web' dialog box with the 'Connection' tab selected. The profile is named 'myFixItA \*'. The 'Publish method' is set to 'Web Deploy'. The 'Server' is 'myfixita.scm.azurewebsites.net:443', 'Site name' is 'myFixItA', and 'User name' is '\$myFixItA'. The password is masked with dots, and the 'Save password' checkbox is checked. The 'Destination URL' is 'http://myfixita.azurewebsites.net'. A 'Validate Connection' button with a green checkmark is visible. At the bottom, the 'Next >' button is highlighted with a red circle.

Publish Web

Profile

myFixItA \*

Connection

Settings

Preview

Publish method: Web Deploy

Server: myfixita.scm.azurewebsites.net:443

Site name: myFixItA

User name: \$myFixItA

Password: .....

☒ Save password

Destination URL: http://myfixita.azurewebsites.net

Validate Connection ✓

< Prev Next > Publish Close

4. On the *Publish Web* page, for **Default Connection**, choose the connection string for the **memberdb** database (using the ellipse button) and **uncheck** the checkbox that would replace the connection string in the web.config file when it is published.

For the **appdb** connection string, use the ellipse button to choose the appdb connection string and **uncheck** the checkbox below it.

Click **Next>**.

The screenshot shows the 'Publish Web' dialog box for a project named 'testMyFixIt \*'. The 'Settings' tab is active. Under the 'Databases' section, there are two database configurations: 'DefaultConnection' and 'appdb'. Each configuration has a 'Data Source' dropdown menu and a checkbox labeled 'Use this connection string at runtime (update destination web.config)'. The 'DefaultConnection' dropdown is set to 'tcp:myfixitwebgroup-server.database.windows.net,1433;Initial Catalog=member'. The 'appdb' dropdown is set to 'tcp:myfixitwebgroup-server.database.windows.net,1433;Initial Catalog=appdb'. Both checkboxes are unchecked. At the bottom of the dialog, there are four buttons: '< Prev', 'Next >', 'Publish', and 'Close'. The 'Next >' button is highlighted with a red box.

5. Click the **Publish** button.
6. Once the application has been published to Azure, go to your web app at <http://<websitename>.azurewebsites.net> to test the web application. (Make sure the MyFixIt.WorkerRole has completed its deployment before testing).

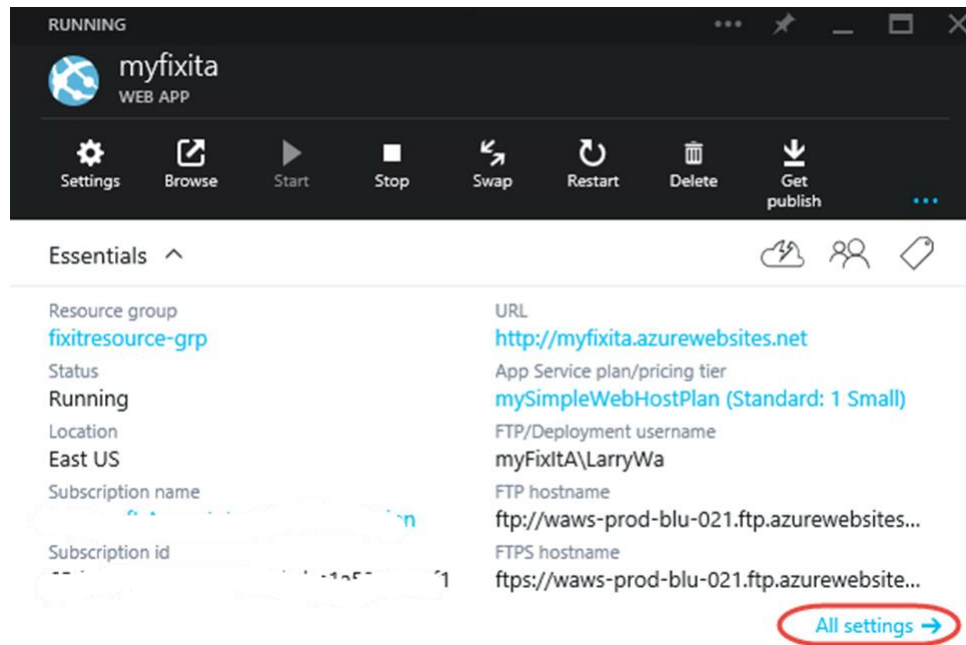
## Exercise 2 – Using Azure Web App Settings

In the previous exercise, your database connection strings for the web application were stored in the web.config file. This is typical of a web application, but Azure Web Apps allow for a more flexible mechanism where you can put this type of information in the Azure portal settings instead of the web.config file. This way, if you need to change the connection strings later, you do not have to redeploy the web application.

For more information on this functionality, see this article

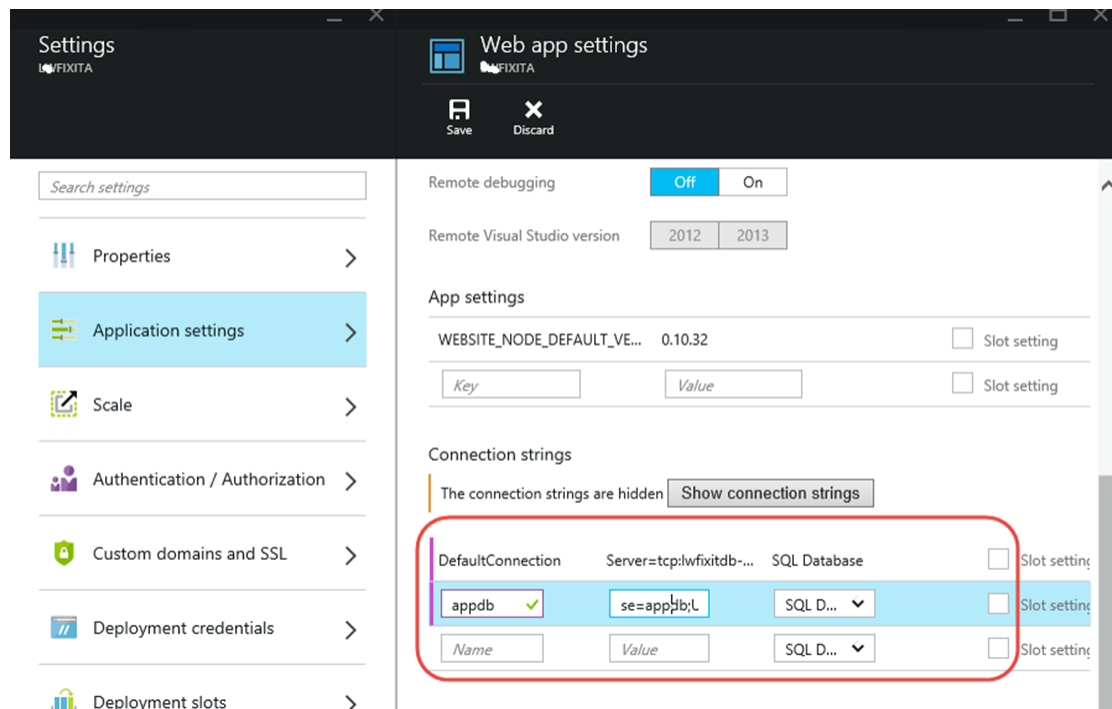
<http://azure.microsoft.com/blog/2013/07/17/windows-azure-web-sites-how-application-strings-and-connection-strings-work/>.

1. You need to have completed the previous exercise to perform this exercise. If you have closed Visual Studio, re-open Visual Studio and open the solution at **C:\Labs\AZRHOL302-AzureWebsites\Source\Exercise1\Begin\MyFixIt.sln** solution.
2. Log into the Azure Preview portal and go to the web app that you created earlier. Click on the **All Settings** icon.



- Click on the **Application Settings** link and then create two new connection string entries (copy the connection strings from your web.config file and make sure the Name is the same as what you have in the web.config file).

Chose a type of SQL Database. Click on the **Save** button at the top of the application settings blade.



- Go back to the **web.config** file of your web application in Visual Studio and comment out or delete the connection strings.
- Right click on the **MyFixIt** web application and select **Publish**. Click the **Publish** button to re-publish the web.config file with no connection strings.
- Test the application again. You should see that you can still perform the same functionality.

## Exercise 3 – Deploy the Web App to a Deployment Slot

In the previous exercise, you deployed your web app directly into the production location (slot) where your end users would be using the web app from.

With Microsoft Azure Web Apps, you have the capability of having multiple (5) deployment slots, 4 for staging/testing and 1 for production.

In this exercise we will create a staging deployment slot and then deploy changes to our FixIt application into the staging slot for testing, prior to swapping it to production.

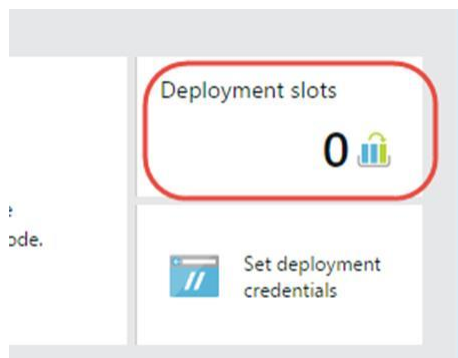
For more information on deployment slots, see <http://azure.microsoft.com/en-us/documentation/articles/web-sites-staged-publishing/>.

Pre-requisite: In order to do this exercise, you need to have completed Exercise 1 and 2.

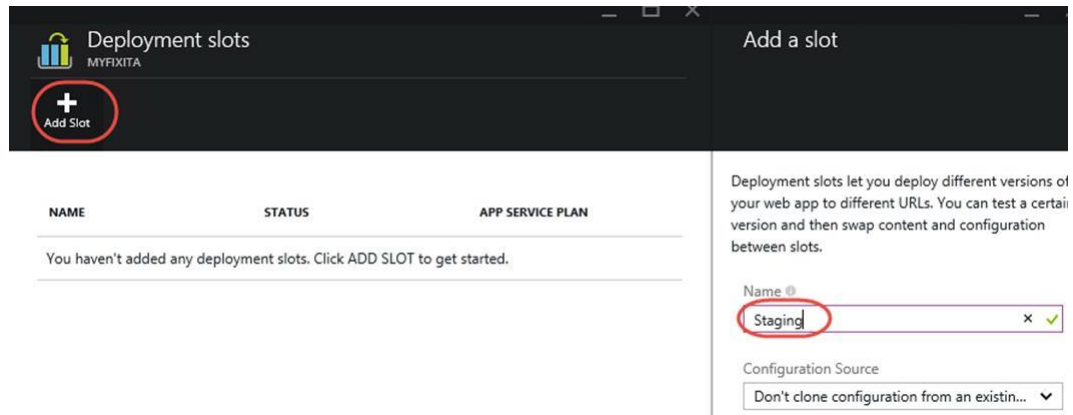
### Task 1 – Creating the Staging deployment slot

There are multiple ways to create a deployment slot. Through the Azure REST API, the Azure portal interface, Microsoft Azure PowerShell etc. For this exercise, we will create the deployment slot using the Azure Management portal (preview). For understanding how to create a deployment slot via PowerShell, go to the link given above and you will find a PowerShell cmdlet section at the bottom of the article.

1. Log into the Azure Management Preview portal at <https://portal.azure.com>.
2. Use the Browse menu item to browse to the web app you previously deployed.
3. In the web app blade, scroll down in the blade and find the **Deployments** icon. Click on this icon.



- Click on the **Add Slot** button and then give the slot the name **Staging** (you can call it whatever you want). Leave the *Configuration Source* settings as they are, we will be doing a re-deployment from Visual Studio.



- Select the **OK** button at the bottom of the *Add a slot* blade.
- Close the Deployment Slots blade once the deployment slot has been created.

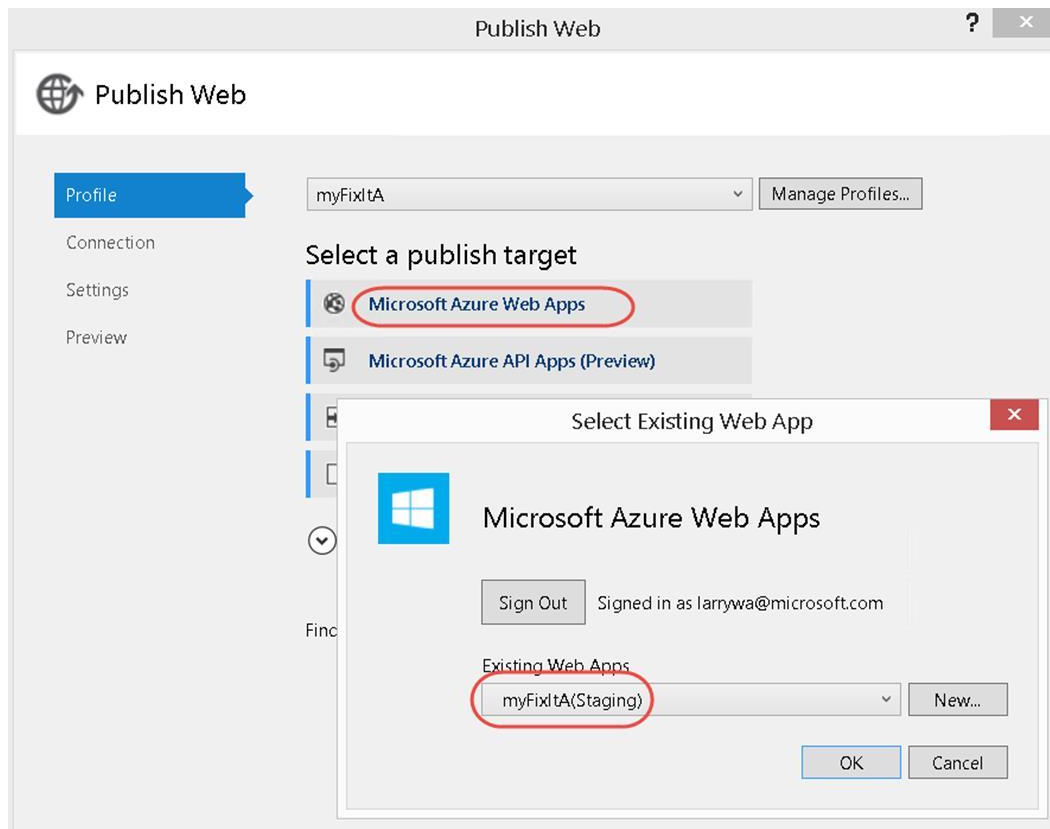
## Task 2 – Publishing the web app to the Staging slot

- Go back to your **MyFixIt** web project in Visual Studio.
- To show that there will be a difference between the Staging and Production slots, we will make a change to something in our web app. In the **MyFixIt** web project, open the **.\Views\Home\Index.cshtml** file.
- Change the header code in the file to look like this:

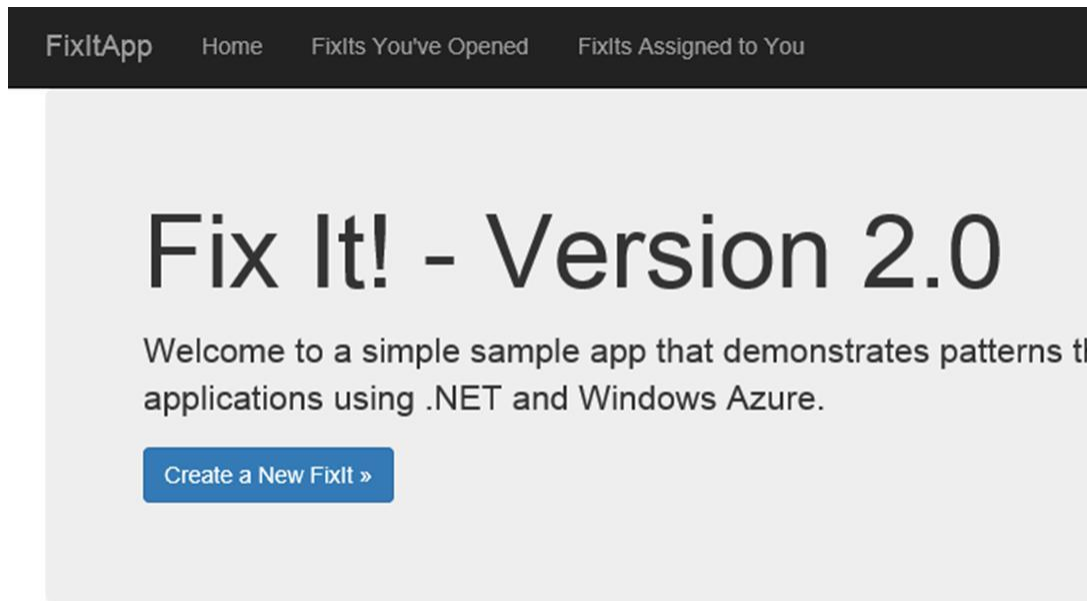
```
<div class="jumbotron">
<h1>Fix It! - Version 2.0</h1>
<p class="lead">
...
```

- Right click on the MyFixIt web application and select **Publish**.

5. In the Publish Web dialog box, you are going to want to click on the **Prev** button to take you back to the very first page of the publishing wizard. You will need to select the **Microsoft Azure Web Apps** link, and then select '*yourwebsite(deploymentslotname)*'. Select **Ok** and then the **Next** button. As before, make sure that the checkboxes for using the connection strings are un-selected. You can proceed in the wizard and Publish the web app to Staging.



6. Once the publishing has completed, you should see the new Index page in the Staging slot that looks like this.



7. To make this web app functional, we need to set the connection strings to the *appdb* and *memberdb* databases. Go back into the Azure Management portal (preview).
8. Find your web app (the first one you deployed) and then scroll down and select the **Deployment Slots** icon (which should by now show 1 deployment slot listed).
9. In the Deployment Slots blade, click on the **<yourwebsite name->Staging** deployment slot.



10. As before, when you were viewing your web app blade, you can now see the web app information as it is deployed in the Staging slot. Click on the **All settings** link.

myfixita-staging  
WEB APP

Settings Browse Start Stop Swap Restart Delete Get publish

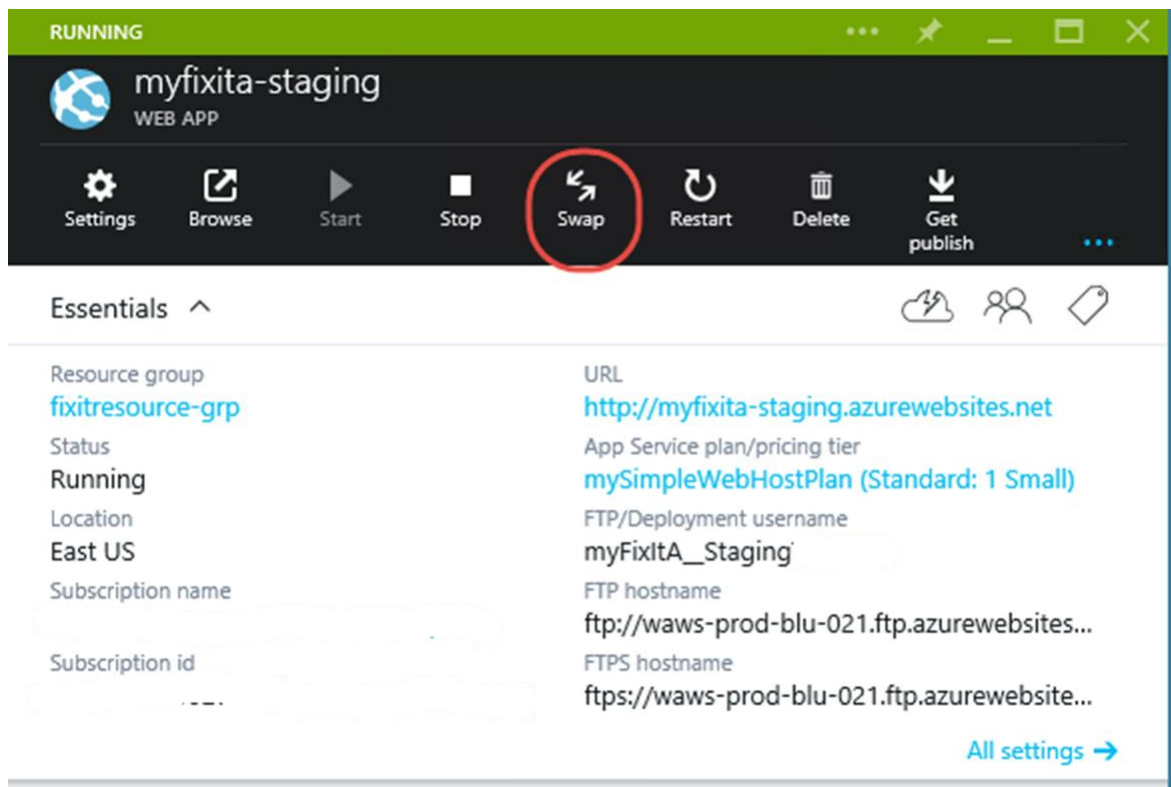
Essentials

Resource group	fixitresource-grp	URL	<a href="http://myfixita-staging.azurewebsites.net">http://myfixita-staging.azurewebsites.net</a>
Status	Running	App Service plan/pricing tier	mySimpleWebHostPlan (Standard: 1 Small)
Location	East US	FTP/Deployment username	myFixitA_Staging
Subscription name		FTP hostname	ftp://waws-prod-blu-021.azurewebsites...
Subscription id		FTPS hostname	ftps://waws-prod-blu-021.azurewebsites...

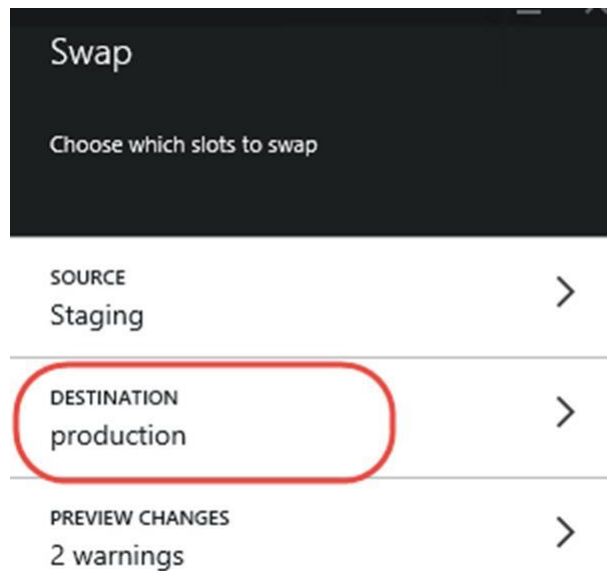
All settings →

11. Set your connection strings to the same values that you did in your production web app. You should be able to test the web application in the Staging deployment slot now.

12. Once you are satisfied that the application in the Staging slot is production ready, go to the web app blade that represents your staging web environment. If you click on the ellipse link, it will expand the toolbar at the top of the blade. Select the **Swap** button.



13. When you click the **Swap** button, you will be presented with a list of slots that you can 'swap' into. In our case, we only have one other slot, so we can select it (the production slot). Select the **Ok** button at the bottom of the Swap blade.



14. You should now be able to test your production web app at <http://<myproductionsite>.azurewebsites.net> and see the updated Index page. If you then go back to the <http://<myproductionwebsite-staging>.azurewebsites.net>, you should see the original production web app.

### Some critical things to note when doing a Swap

When you swap deployment slots, all of the web app content is swapped, but the same is not true of the configuration. The following configuration items will move to the destination slot:

General settings (for example, .NET framework version, Web Sockets, Always On)

- Connection strings
- Handler mappings
- Application and site diagnostics settings
- Monitoring settings

The following configuration items will not move to the destination slot:

- Publishing endpoints
- Custom domain names
- SSL certificates and bindings
- Scale settings

In effect, this means that settings such as your database connection strings should be configured with the production values prior to the swap. It also means that you do not have to worry about your non-production SSL certificate bindings or domain names accidentally overwriting your production settings. The fact that publishing endpoints do not swap also implies that if you are deploying to web apps via source control (for example, Visual Studio Online), your production slot will not suddenly be updated from source control just because you swapped into production a slot that is configured for deployment from source control.

### **How to create a deployment slot via PowerShell**

Using PowerShell, the deployment slot has to be created whenever the web app is created.

```
$wsQASlot = "Staging"
```

```
New-AzureWebsite -Location $wsLocation -Name $wsName -Slot $wsQASlot
```

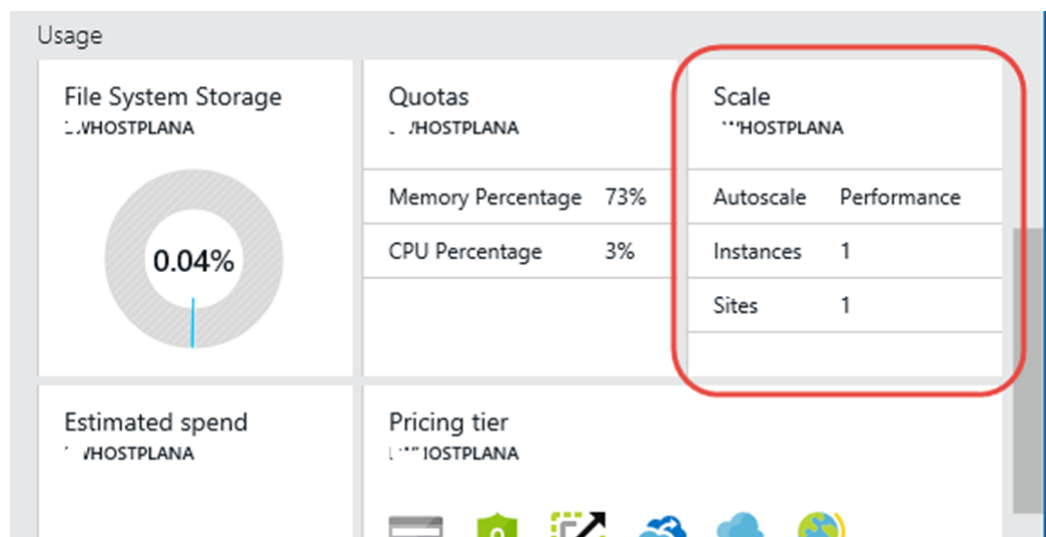
### Exercise 3 – Implementing Auto-Scaling for the Azure Web App

There are multiple ways to implement auto-scaling for an Azure web app, but in general, the app service plan that the web app is in needs to be using either the Basic or Standard tier. For the FixIt web app you deployed previously, the hosting plan you chose or created is already in the Standard tier.

For example, in the Standard tier, we can scale up to 10 machines, based on CPU performance or other metrics. In this exercise, we will scale up by CPU Performance.

#### Task 1 – Setup Auto-Scaling using the Azure Portal

1. Open the Azure Preview Portal at <https://portal.azure.com>. Log in using your credentials.
2. Click on the **Browse** button and select **Web Apps**. Find your web app in the blade that appears to the right which should take you to your web apps property blade.
3. In your web apps blade, scroll down to the **Usage** section and click on the **Scale** icon.



4. In the Scale blade, make sure that the Performance button is selected and then (as an example), move the instance range up to 10 with the minimal setting of 2. This will assure that you at least have two instances running at all times.

Choose scale

Autoscale Mode ⓘ

Off

Performance

Instance Range

2

10

- For the Target Metrics/CPU Percentage slider, set the low range to 50 and the upper range to 90. This means is that if the CPU level percentage exceeds 90%, Azure will add 8 more web app machines for you automatically. When the average CPU percentage goes below 50%, the instances will be scaled back down to 2 instances.

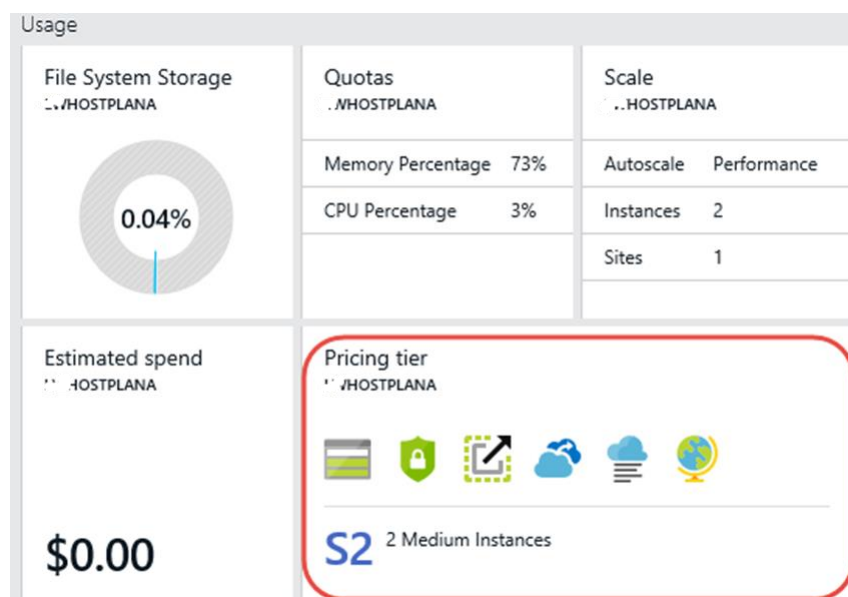


- Click on the **Save** button at the top of the blade. Note: It could take a few minutes for the Scale icon to update its status to 2 instances since Azure needs to create a new machine instance.

## Task 2 – Changing the web server capacity

Changing the web server capacity means changing the size of the machine that you are running your web app on. Typically, these settings are set whenever you create your hosting plan.

- In your web app blade, scroll down to the **Usage** section and click on the **Pricing Tier** icon.



2. Select the **S3 Standard** (Large machine) and then click on the **Select** button. Wait for your Pricing Tier icon to update to the new value.

Choose your pricing tier  
BROWSE THE AVAILABLE PLANS AND THEIR FEATURES

★ Recommended | View all

P1 Premium (Preview)	P2 Premium (Preview)	P3 Premium (Preview)
<b>1</b> Core	<b>2</b> Core	<b>4</b> Core
<b>1.75</b> GB RAM	<b>3.5</b> GB RAM	<b>7</b> GB RAM
BizTalk Services	BizTalk Services	BizTalk Services
250 GB Storage	250 GB Storage	250 GB Storage
Up to 20 instances * Subject to availability	Up to 20 instances * Subject to availability	Up to 20 instances * Subject to availability
20 slots Web app staging	20 slots Web app staging	20 slots Web app staging
50 times daily Backup	50 times daily Backup	50 times daily Backup
Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability
<b>74.40</b> USD/MONTH (ESTIMATED)	<b>148.80</b> USD/MONTH (ESTIMATED)	<b>297.60</b> USD/MONTH (ESTIMATED)

S1 Standard	S2 Standard	S3 Standard
<b>1</b> Core	<b>2</b> Core	<b>4</b> Core
<b>1.75</b> GB RAM	<b>3.5</b> GB RAM	<b>7</b> GB RAM
50 GB Storage	50 GB Storage	50 GB Storage
5 SNI, 1 IP Custom domains / SSL	5 SNI, 1 IP Custom domains / SSL	5 SNI, 1 IP Custom domains / SSL
Up to 10 instances Auto scale	Up to 10 instances Auto scale	Up to 10 instances Auto scale
Daily Backup	Daily Backup	Daily Backup
5 slots Web app staging	5 slots Web app staging	5 slots Web app staging
Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability
<b>74.40</b> USD/MONTH (ESTIMATED)	<b>148.80</b> USD/MONTH (ESTIMATED)	<b>297.60</b> USD/MONTH (ESTIMATED)