WorkshopPLUS

Microsoft Azure Infrastructure as a Service (IaaS)

Azure Resource Manager Deployments

Student Lab Manual

V1.8, December 2, 2015

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.

© 2014 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

[Exercise 1 – Using the Visual Studio Templates 5](#_Toc435731378)

[Prerequisites 5](#_Toc435731379)

[Task 1 – Create the Azure Resource Group project 5](#_Toc435731380)

[Task 2 – Deploy to Azure 8](#_Toc435731381)

[Task 3 – Confirm Deployment of Virtual Machines with Load Balancer 10](#_Toc435731382)

[Exercise 2 – Using PowerShell with ARM Templates 13](#_Toc435731383)

[Task 1 – Setting up the PowerShell deployment 13](#_Toc435731384)

[Task 2 – Deploying with PowerShell 15](#_Toc435731385)

**Introduction to Microsoft Azure Resource Manager**

In this lab, you will create a new Azure Resource Group with Visual Studio and then add a project that will deploy two virtual machines within a load balanced set.

You'll learn:

* How to create an Azure Resource Group via Visual Studio templates

# Exercise 1 – Using the Visual Studio Templates

## Prerequisites

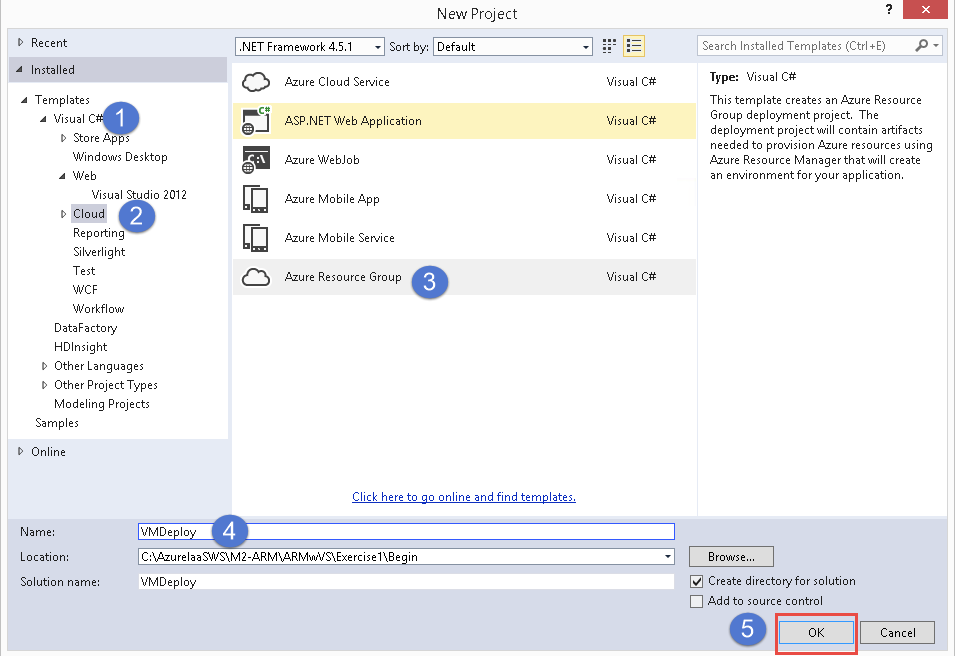
The following is required to complete this hands-on lab:

* Microsoft V[isual Studio 2013 Professional or Ultimate edition](http://www.microsoft.com/visualstudio/) with Update 5
* [Microsoft Azure SDK for .NET (VS 2013) - 2.](http://www.microsoft.com/windowsazure/sdk/)8
* Microsoft Azure PowerShell
* A Microsoft Azure subscription

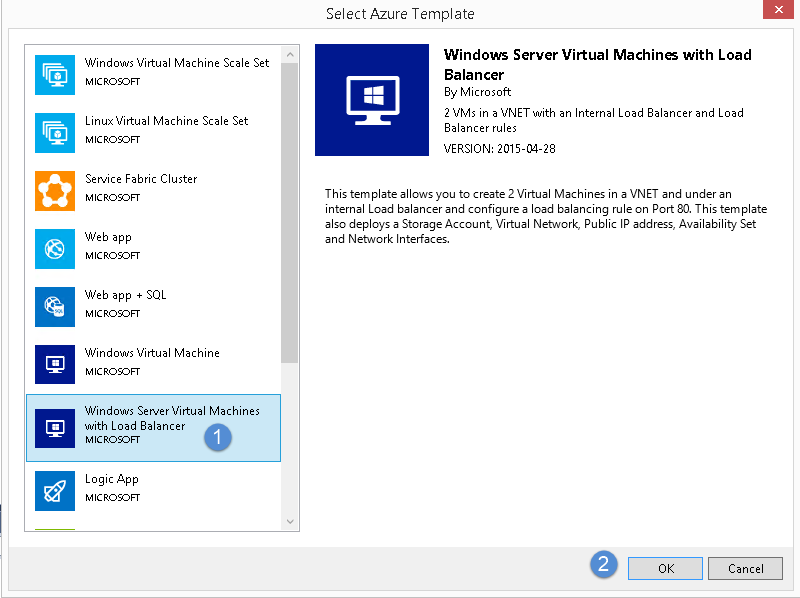
## Task 1 – Create the Azure Resource Group project

Although the .json script for an Azure Resource Manager (ARM) template can be created by hand, it is easier to use pre-existing templates that are provided by Visual Studio. There will still be modification required in order to achieve the end goal of deploying a web application using the ARM template.

1. Open Visual Studio as an Administrator.
2. Select **File->New->Project** and then select **Visual C#->Cloud->Azure Resource Group**. Give the project the name **VMDeploy**. Set the location of the project to the **.\AzureIaaSWS\M2-ARM\Labs\Begin**. Select the **OK** button.



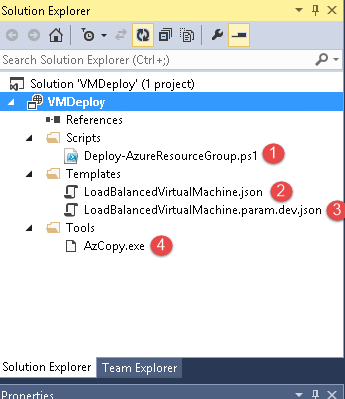
1. In the Select Azure Template dialog box, select **Windows Server Virtual Machines with Load Balancer** and then select the **OK** button.



1. View the contents of the project that has been created.

You will see:

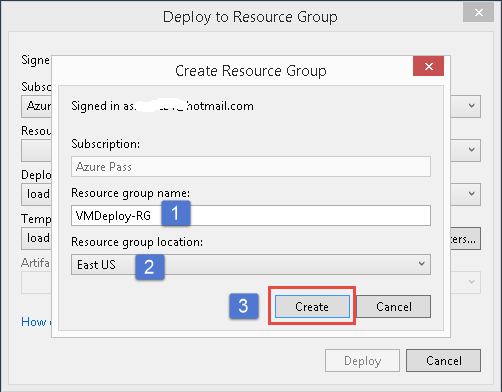
1. **Deploy-AzureResourceGroup.ps1** – the PS script used to actually deploy the template and any application referenced.
2. The .json template file for the project.
3. The .json template parameter file used by the main template.
4. AzCopy.exe, used to copy the template and files up to Azure storage for deployment



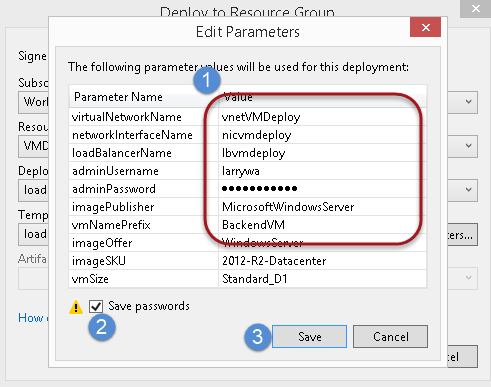
## Task 2 – Deploy to Azure

The version of Microsoft Azure PowerShell that you will be using is generally newer than the PowerShell script used in the Visual Studio template. Because of this, we will be replacing the Deploy-AzureResourceGroup.ps1 template with one that has the appropriate PowerShell commands.

1. Right click on the **VMDeploy** project in the Solution Explorer and select **Deploy->New Deployment.**
2. In the *Deploy to Resource Group* dialog box, if you have not already signed in to Azure, click the **Sign In** button and sign in.
3. Once you are signed in to Azure, select the dropdown in the Resource Group box and create a new, unique resource group name. A dialog box will appear that allows you to select the resource group location also. Then, select the **Create** button.



1. Now select the **Edit Parameters** button. The parameters that you enter here will be passed to the deployment, and will also be placed in the .json parameters file in your Visual Studio project.



The parameters that need to be set are circled above.

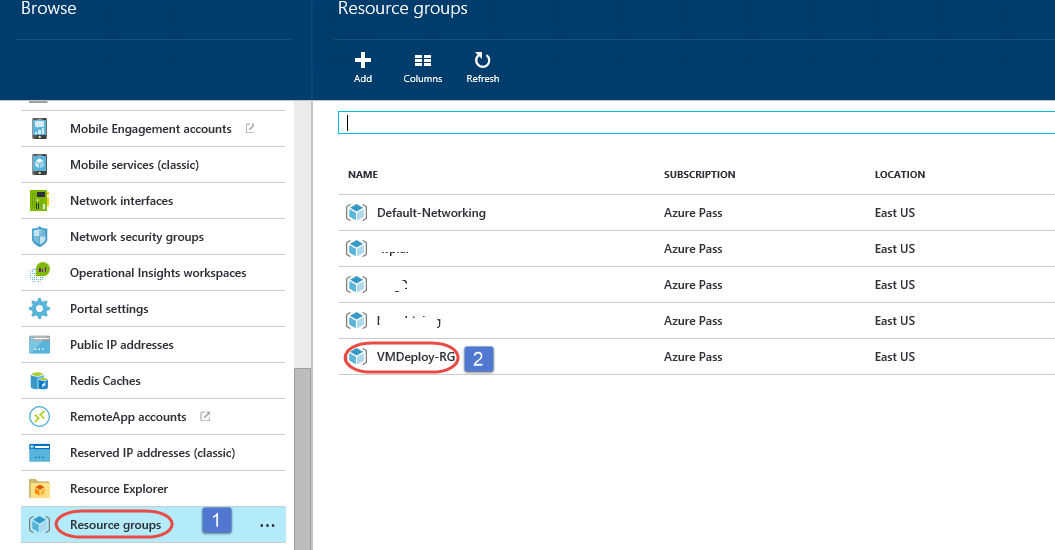
* virtualNetworkName – name of the virtual network to place the VMs in
* networkInterfaceName – NIC card name
* loadBalancerName – load balancer name
* adminUsername – RDP admin username
* adminPassword – RDP admin password
* imagePublisher – company name of the publisher of the image. This is the same value for all Windows VMs. Auto-populated.
* vmNamePrefix – each VM will have an auto generated name but will begin with this prefix
* imageOffer – Auto-populated. Image offer name
* imageSKU – version of Windows VM to use
* newStorageAccountName – V2 virtual machines require a V2 storage account. The Visual Studio tooling and template does not allow you to enter a pre-created storage account. This storage account name must be all small letters and unique within all of Azure
* vmSize – size of the VM. If you know the correct naming convention, you can enter a different size

Make sure you check the ‘**Save Password’** checkbox and select **Save**.

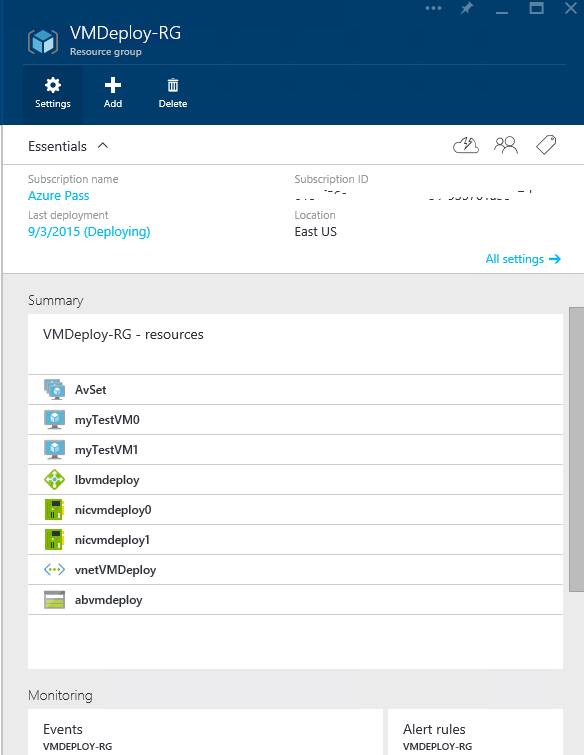
1. Click the **Deploy** button. If the Edit Parameters box pops up again, just make sure you select the Save Passwords checkbox again and select **Save**.
2. You will be prompted, when the script begins to execute, to log in to Azure again. This login is just to authenticate you in to Azure. The subscription name and resource group you previously selected will still be used just as you had originally selected them.

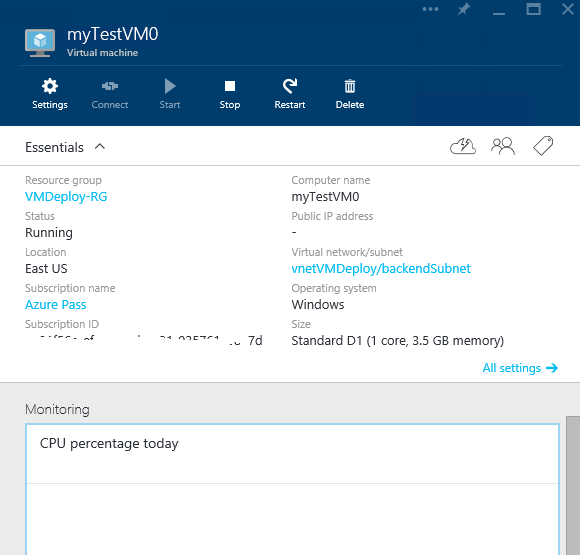
## Task 3 – Confirm Deployment of Virtual Machines with Load Balancer

1. Log in to the Azure portal at <https://portal.azure.com>. You will not be able to see resource groups from the *current* portal.
2. On the left hand side, click the **Browse** button. Select **Resource groups** and then click on the name of the resource group you created for the deployment.



1. Your resource group blade will open. It will display all resources that belong to this resource group. Click on the name of the first VM (ex. myTestVM0). The VM blade will appear.





From the VM blade, there are a variety of things you can do, such as:

1. Add data disks (select **All settings->Disks**)
2. Add role based access control permissions
3. Stop and Restart the machines
4. More settings to explore..click **All Settings**

Notice in the toolbar that the **Connect** button is disabled. This is because this machine does not have a public IP address (see *Essentials* area of blade). In this case you can only RDP into the machine if you do so with a machine within the same virtual network. You will learn more about this in later labs.

1. Close the VM blade.

# Exercise 2 – Using PowerShell with ARM Templates

In Exercise 1, you used Visual Studio and one of the Microsoft provided deployment templates to deploy a set of VMs with a load balancer.

In a typical IT environment, developers will hand off the files/packages needed to someone else whose responsibility it is to actually do the deployments. From an IT perspective, this is generally performed through the use of PowerShell.

In this exercise, you are going to first make sure you have created the files necessary for deployment and then you will use PowerShell for the deployment.

## Task 1 – Setting up the PowerShell deployment

1. You can either continue working with the project from Exercise 1, or open the project located at **.\Labs\ARMwVS\Exercise2\Begin\VMDeploy**. Open the project with Visual Studio as an Administrator.

NOTE: If you are going to use the project at **.\Labs\ARMwVS\Exercise2\Begin\VMDeploy,** you are going to have to open the **.\Labs\ARMwVS\Exercise2\Begin\VMDeploy\VMDeploy\Templates\LoadBalancedVirtualMachine.param.json** file and update the parameters to you own custom values:

{

"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",

"contentVersion": "1.0.0.0",

"parameters": {

"virtualNetworkName": {

"value": ""

},

"networkInterfaceName": {

"value": ""

},

"loadBalancerName": {

"value": ""

},

"adminUsername": {

"value": ""

},

"adminPassword": {

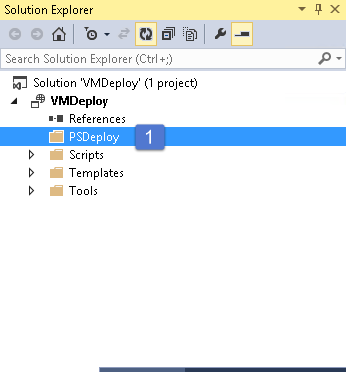
"value": ""

}

}

}

Otherwise, if you are continuing with the same project you had open from Exercise 1, you would still want to modify the storage account name. Storage accounts must be unique in all of Azure.

1. Within the Visual Studio VMDeploy project, right click on VMDeploy and select the **Add->New Folder** and create a new folder named **PSDeploy**. Your project should now look like this:  
     
   
2. Open Windows Explorer and browse to your **PSDeploy** folder.
3. Copy the **".\M2-ARM\Labs\ARMwVS\Source\Ex2\Deploy-AzureResourceGroup.ps1"** to the **.\PSDeploy** folder. This is a modified version of the script we have in the Visual Studio .\scripts folder. The script has been modified to remove most of the boilerplate PowerShell that Visual Studio uses for other types of templates.
4. Copy the **.\VMDeploy\Templates\LoadBalancedVirtualMachine.json** and the **.\VMDeploy\Templates\ LoadBalancedVirtualMachine.param.json** files to the **.\PSDeploy** folder.

## Task 2 – Deploying with PowerShell

1. Open PowerShell ISE as an Administrator.
2. In the PowerShell ISE command prompt window, change the directory to point to your **.\PSDeploy** directory.
3. When deploying through Visual Studio, you are typically already logged in to Azure, so PowerShell does not ask for any authentication. Using PowerShell ISE however, we need to perform two steps:
   1. In the PowerShell ISE command prompt window, type in **Clear-AzureProfile**. This will make sure that any profile information previously stored is removed from the machine.
   2. In the PowerShell ISE command prompt window, type in **Login-AzureRMAccount** and log in to Azure. This command only logs you in, and if you have multiple subscriptions, will just grab the first subscription ID it finds. Note, you may not want to use the subscription that this command selects, so follow the rest of the steps.
   3. Type in

**Get-AzureRMSubscription | Select SubscriptionName,SubscriptionId**

This will give you a list of subscription names and associated IDs that you can choose from for the next command.

* 1. Enter the command:  
       
     **Select-AzureRMSubscription –SubscriptionId <subscriptionId-from-last-command>**

This command will select the subscription associated with the ID into the currently running PowerShell instance. All commands you run from this point on will be executed against this subscription.

* 1. To give PowerShell permission to execute a script on your machine, type in (you will be prompted to confirm this permission):  
     **set-executionpolicy unrestricted**

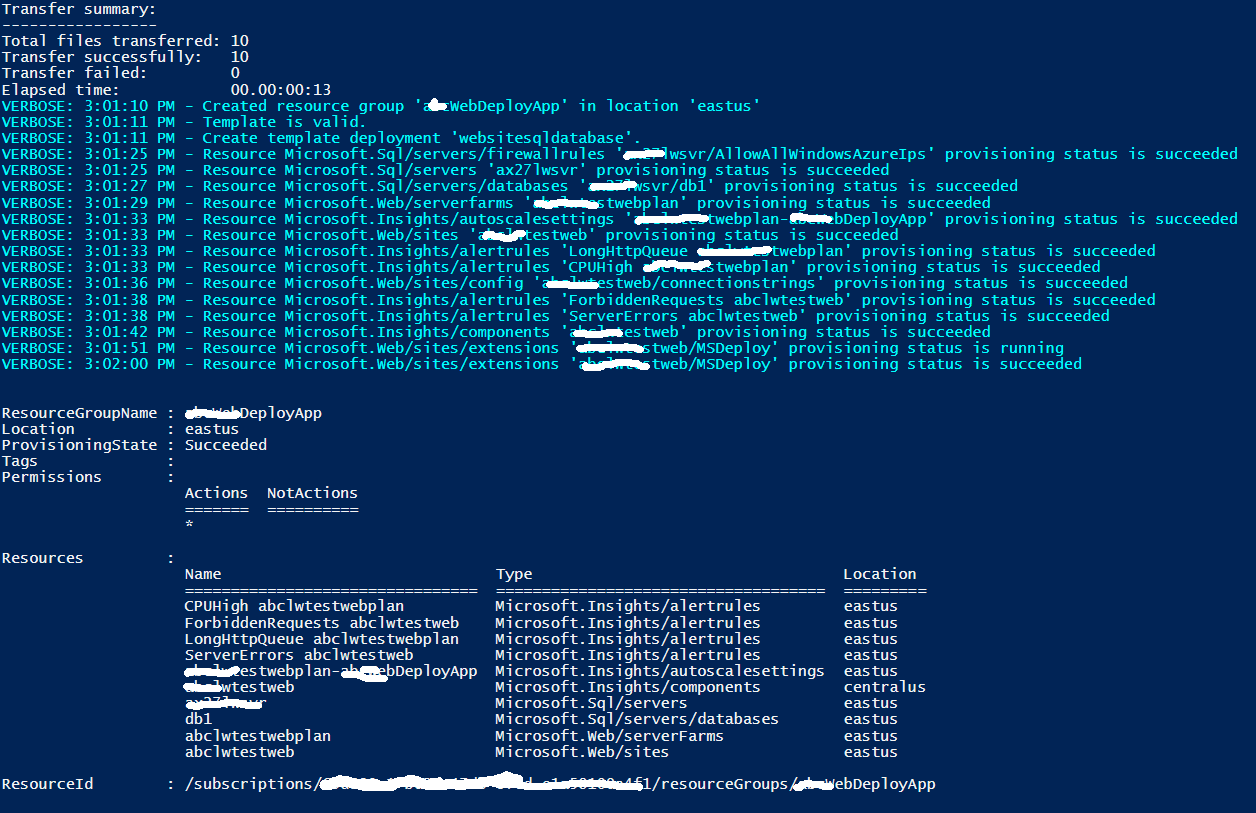
1. Using the PowerShell ISE script window, open the **Deploy-AzureResourceGroup.ps1** file.
2. You may have noticed that during the Visual Studio deployment, a PowerShell command window briefly pops up. In the Visual Studio output window, you can actually see the PowerShell command being executed that calls our deployment script. This is the command we will be running from the PowerShell ISE command prompt.

'\Deploy-AzureResourceGroup.ps1' -StorageAccountName '[your-storage-acct-name]' -ResourceGroupName '[your-resourcegroup-name]' -ResourceGroupLocation 'eastus' -TemplateFile '.\LoadBalancedVirtualMachine.json' -TemplateParametersFile '.\LoadBalancedVirtualMachine.param.json' -UploadArtifacts

1. Fill in the appropriate command line parameters and run the command in PowerShell ISE command prompt window (you may want to check to be sure that you don’t already have a resource group of the same name in your parameters file). If you are following along from the previous exercise you will need a new resource group name and you’ll need to confirm that your admin username and password are in the parameters file. You may also decide to change the –ResourceGroupLocation to a different region.

.\Deploy-AzureResourceGroup.ps1 -ResourceGroupName '[your-resourcegroup-name]' -ResourceGroupLocation 'eastus' -TemplateFile '.\LoadBalancedVirtualMachine.json' -TemplateParametersFile '.\LoadBalancedVirtualMachine.param.json'

1. If you deployment succeeds, you should see a PowerShell display such as this:



1. You can now log in to the Azure portal at <https://portal.azure.com> to look for your resource group and test your web app.