

Creating and Configuring Azure Web Apps

Lab Time: 60 minutes

Lab Folder: C:\Student\Modules\02_AzureWebApps\Lab

Lab Overview: In this module you will work with your Azure subscription to create and configure several different web apps. You will also create a simple ASP.NET MVC application in Visual Studio 2017 and work through the steps to deploy the application to an Azure web app using deployment slots and staged deployments.

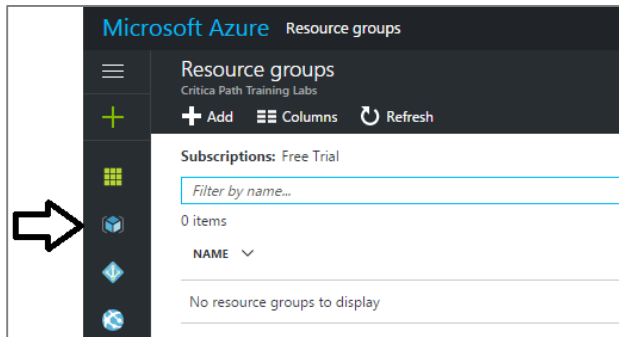
Exercise 1: Create a Web App by Hand using the Azure Portal

In this exercise you will create an app service plan and an Azure Web App by hand in the Azure portal.

1. Login into the Azure portal.
 - a) In the browser, navigate the following URL.

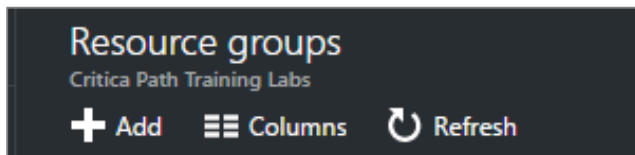
<https://portal.azure.com>

- b) Login using the credentials associated with your Azure subscription.
2. Create a new resource group named **lab02**.
 - a) In the left navigation, click on the resource group button to see the list of existing resource groups.

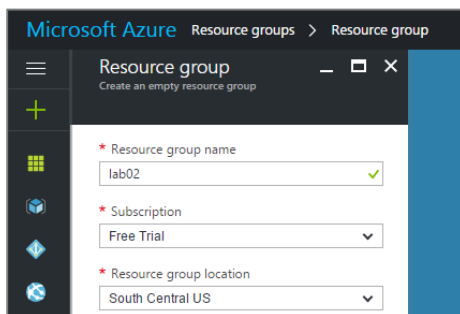


At this point, the list is likely empty because you have not created anything in your Azure subscription yet.

- b) Click the **Add** button to create a new resource group.



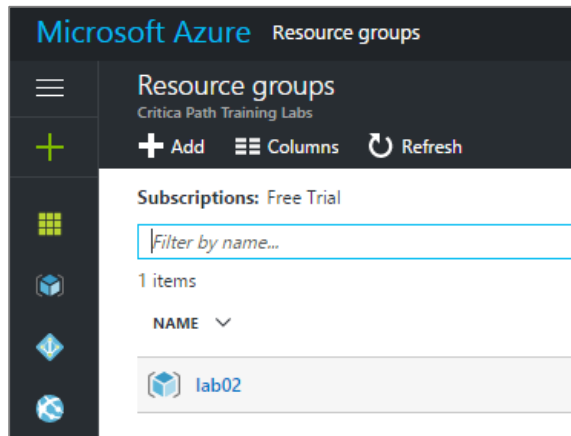
- c) Give the new resource group a name of **lab02**. Assign a resource group location that is appropriate for where you live.



- d) Click the **Create** button below to create the new resource group

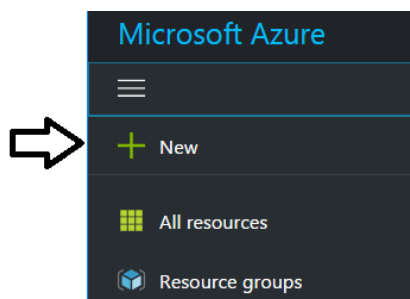


- e) You should now be able to see the new resource group in the list of resource groups.

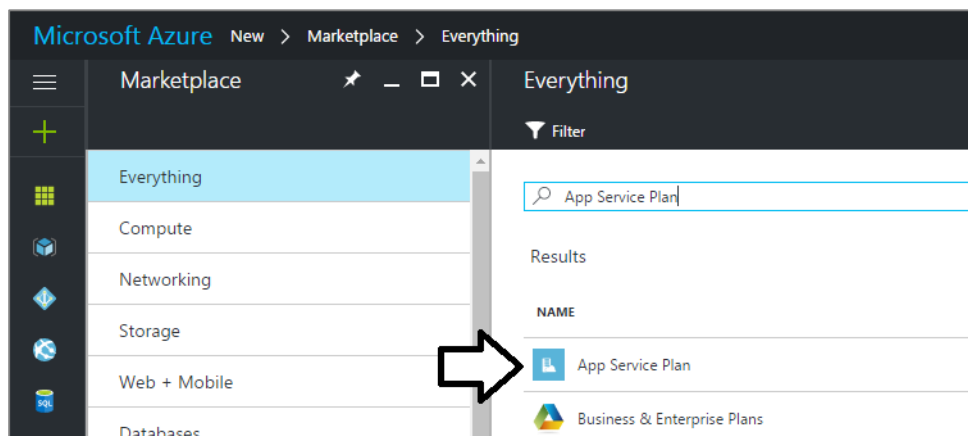


3. Create an App Service Plan named **lab02-plan**.

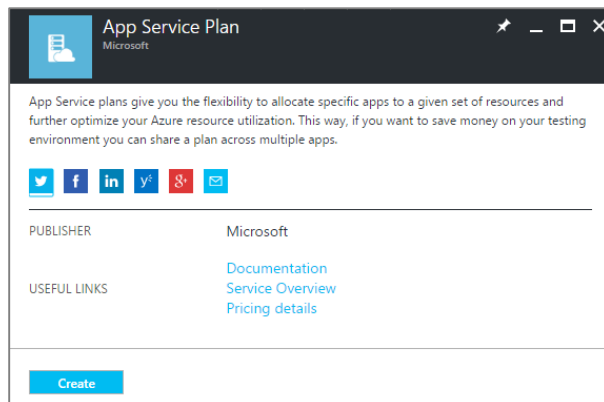
- a) Click the **New** button at the top of the left navigation menu.



- b) Type **App Service Plan** into the search box and wait until you see **App Service Plan** as a search result.
c) Click on **App Service Plan** to begin the creation process.



- d) Click the **Create** button to continue creating a new app service plan.

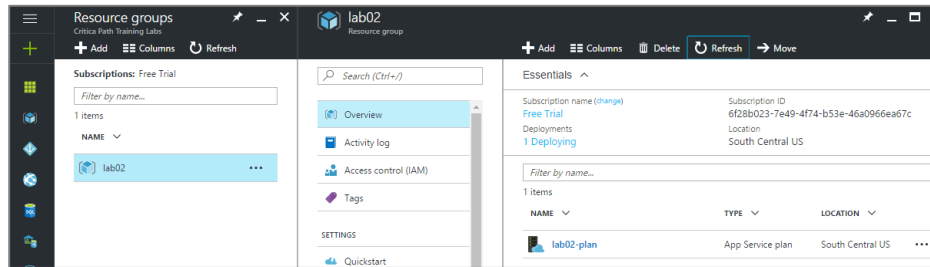


- e) Give the new app service plan a name of **lab02-plan**.
f) Add the new app service plan to the resource group you created named **lab02**.
g) Make sure the **Operating System** is set to **Windows**.
h) Assign the same location that you assigned to the resource group named **lab02**.

- i) Assign the new app service plan a **Pricing Tier** of **S1 Standard**.

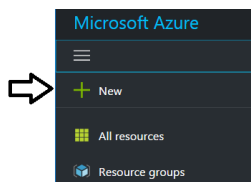
- j) Click the **Create** button to create the new app service plan.

- k) After you have created the new app service plan, click on the resource group named **lab02** in the resource group list. You should be able to verify that the app service plan named **lab02-plan** has been added to this resource group.

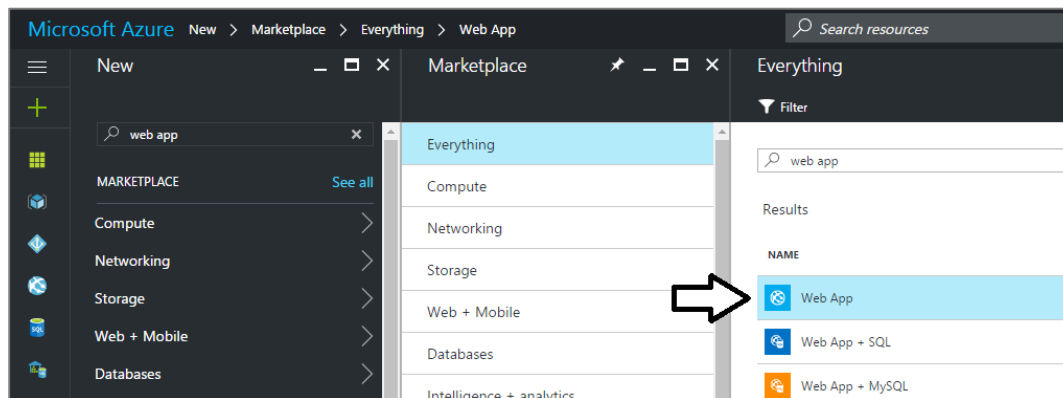


4. Create a new azure Web App.

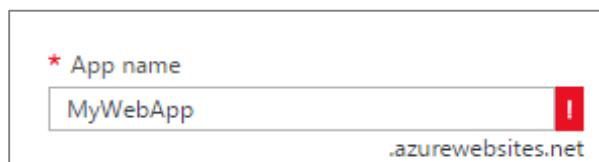
- a) Click the **New** button at the top of the left navigation menu.



- b) Type **web app** into the search box and wait until you see **Web App** as a search result.
c) Click on **Web App** to begin the creation process for the new azure Web App.



- d) Type in the value of **MyWebApp** for the **App Name**. When you do this, you will most likely see a red indicator that tells you that this **App name** value has already been taken and is not available for use in a new Azure web app.



- e) Add a few digits to the end of the **App name** until you see the green checkmark indicating that you have created a unique name that can be used to create a new web app.

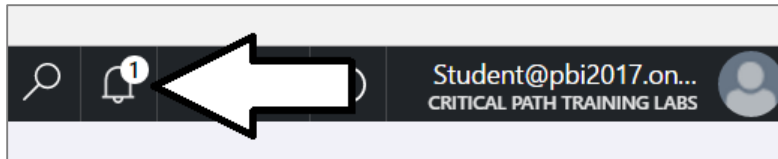


- f) Configure the **Resource Group** setting with the existing resource group named **lab02**.
- g) Configure the **App service plan/Location** setting with the app service plan named **lab02-plan**.
- h) Leave the **Application Insights** setting with its default of **Off**.

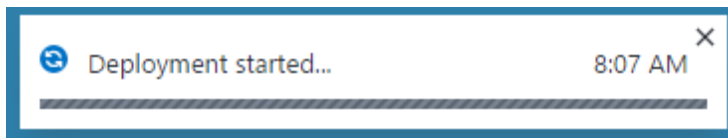
The screenshot shows the 'New Web App' configuration form in the Azure portal. The fields are as follows:

- App name:** MyWebApp714 (with a green checkmark and .azurewebsites.net domain)
- Subscription:** Free Trial (dropdown menu)
- Resource Group:** Use existing (radio button selected), lab02 (dropdown menu)
- App Service plan/Location:** lab02-plan(South Central US) (dropdown menu)
- Application Insights:** Off (toggle switch)

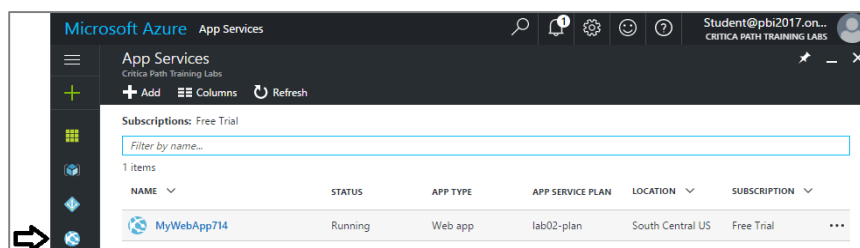
- i) Click the **Create** button below to create the new web app.
- j) While the web app is being provisioned, you can check on its status. Click on the Notification menu with the bell icon at the top right of the Azure portal window.



- k) You should see a status indicator telling you that the deployment has started. You can use the Notification menu to monitor the provisioning process and determine when the web app has been fully provisioned and is ready for use.



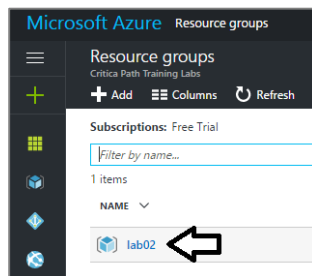
- l) Click on the **App Services** button in the left navigation menu. You should be able to see your new web app.



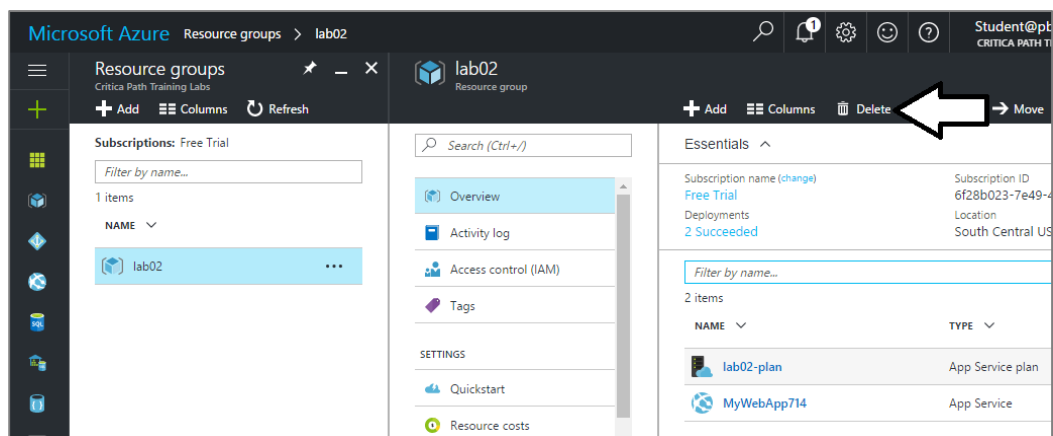
In the next step you will delete the resource group named **lab02** that you just created. This might seem like a strange step because you will be throwing away all the work you have done so far. However, there is a reason for this. In the next exercise you will be creating the exact same set of Azure resources using a PowerShell script. The main point that this lab is emphasizing is that anything you can do by hand in the Azure portal can also be automated by writing a PowerShell script.

- 5. Delete the resource group named **lab02**.

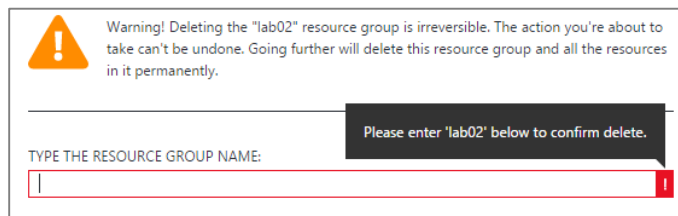
- a) In the left navigation, click on the resource group button to see the list of existing.
- b) Click on the resource group named **lab02** to select it.



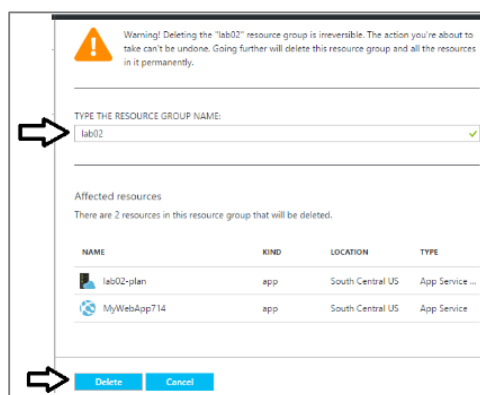
- c) Once the resource group has been selected, click the **Delete** button as shown in the following screenshot.



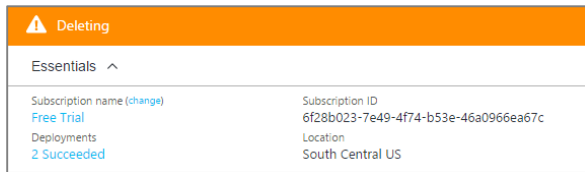
- d) The Azure portal requires that you type in the name of the resource group to confirm you want to delete it..



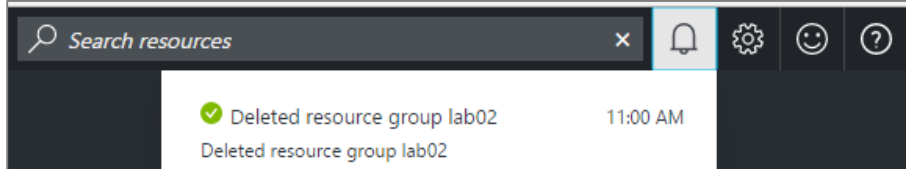
- e) Type in the resource group name of **lab02** to confirm you want to delete it and then click **Delete** button below.



- f) At this point, the deleting process should have started.



- g) Wait until you can confirm that the resource group has been deleted.



Now you have learned how to create Azure resources by hand in the Azure portal. In the next exercise you will learn to accomplish the same result in a much faster and convenient way using a PowerShell script.

Exercise 2: Create a New Azure Web App using PowerShell

In this exercise you will create a new resource group, app service plan and web app using a PowerShell script.

1. Open and update the PowerShell script named **Create-Web-App.ps1**.
 - a) Launch the PowerShell ISE.
 - b) In the PowerShell ISE, open the script named **Create-Web-App.ps1** located at the following path.

```
C:\Student\Modules\02_AzureWebApps\Lab\Create-Web-App.ps1
```

In the next step make sure to use the same Web App name you use in the previous exercise. The key point here is to use a web app name that you already know is unique.

- c) Update the first two lines with your tenant name and your primary office 365 account name. Also update the **\$webAppName** variable using the web app name you used in the previous lab.

```
$tenantName = "[Your Tenant]"
$tenantAdminAccountName = "[Your User Account]"
$tenantDomain = $tenantName + ".onmicrosoft.com"
$tenantAdminSPN = $tenantAdminAccountName + "@" + $tenantDomain

$location = "southcentralus"
$resourceGroupName = "lab02"
$appServicePlanName = "lab02-plan"
$webAppName = "MyWebApp714"
$webAppSlotName = "staging"
```

- d) Save your changes to **Create-Web-App.ps1**.

The next few steps will provide you with a quick walkthrough of the code in this PowerShell script named **Create-Web-App.ps1**. Note that you are not required to make any more modifications to this script. However, it's important you understand what this script does.

- e) The code in **Create-Web-App.ps1** begins by calling **Get-Credential** and passing in your user name. That means you will be prompted to provide your password but you will not be required to type in your user name when the script runs. After that, the script calls **Login-AzureRmAccount** to establish a login session with Microsoft Azure.

```
# establish login
$credential = Get-Credential -UserName $tenantAdminSPN -Message "Enter password"
Login-AzureRmAccount -Credential $credential | Out-Null
```

- f) Next, the script checks to see if a resource group named **lab02** already exists. If the resource group named **lab02** does not already exist, the script creates it using the **New-AzureRmResourceGroup** cmdlet.

```
# Create resource group if it doesn't already exist

$resourceGroup = Get-AzureRmResourceGroup -Name $resourceGroupName -ErrorAction Ignore

if(!$resourceGroup){
    Write-Host "Resource group named" $resourceGroupName "does not exist - now creating it"
    $resourceGroup = New-AzureRmResourceGroup -Name $resourceGroupName -Location $location
}
```

- g) Next, the script checks if an app service plan named **lab02-plan** already exists. If the app service plan named **lab02-plan** does not already exist, the script creates it using the **New-AzureRmAppServicePlan** cmdlet. Note that the app service plan is created with a pricing tier of **S1 Standard**.

```
# create app service plan if it doesn't already exist
$appservicePlan = Get-AzureRmAppServicePlan -Name $appservicePlanName -ErrorAction Ignore
if(!$appservicePlan){

    Write-Host "App servie Plan named" $appservicePlanName "does not exist - now creating it"

    $appservicePlan = New-AzureRmAppServicePlan `
        -ResourceGroupName $resourceGroupName `
        -Location $location `
        -Name $appservicePlanName `
        -Tier Standard `
        -WorkerSize Small `
        -NumberofWorkers 1

}
```

- h) At the end, the script checks to see if a web app already exists with the name of the variable **\$webAppName**. If a web app of that name does not already exist, the script creates it using the **New-AzureRmWebApp** cmdlet. Note that the script also extends the web app by creating a deployment slot named **staging** with a call to **New-AzureRmWebAppSlot**.

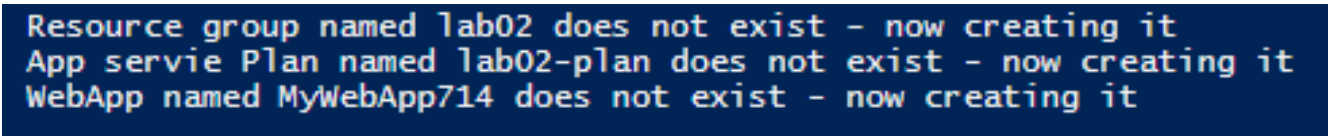
```
# create web app if it doesn't already exist
$webApp = Get-AzureRmWebApp -Name $webAppName -ResourceGroupName $resourceGroupName -ErrorAction Ignore
if(!$webApp) {

    Write-Host "WebApp named" $webAppName "does not exist - now creating it"
    $webApp = New-AzureRmWebApp `
        -ResourceGroupName $resourceGroupName `
        -Location $location `
        -AppServicePlan $appservicePlanName `
        -Name $webAppName

    $slot = New-AzureRmWebAppSlot `
        -ResourceGroupName $resourceGroupName `
        -Name $webAppName `
        -Slot $webAppSlotName

}
```

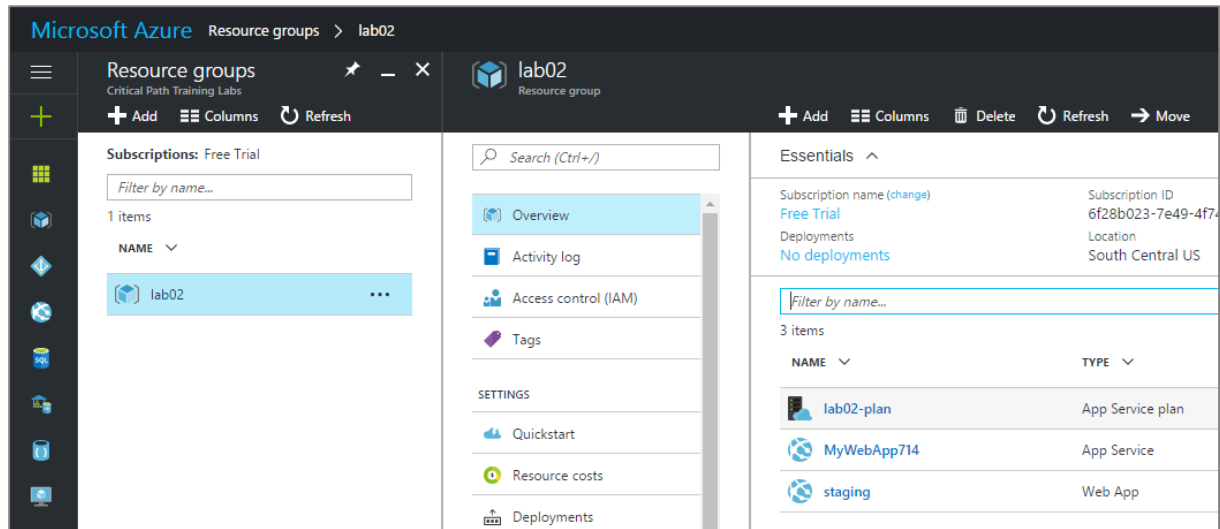
2. Execute the script named to **Create-Web-App.ps1** using the PowerShell ISE. When the script executes, it should output text to the console indicating success as shown in the following screenshot.



```
Resource group named lab02 does not exist - now creating it
App servie Plan named lab02-plan does not exist - now creating it
WebApp named MyWebApp714 does not exist - now creating it
```

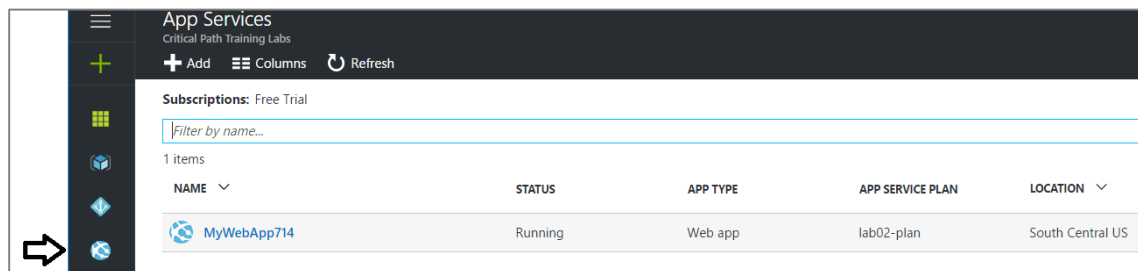
3. After the script named **Create-Web-App.ps1** has executed, return to the Azure portal and see what has been created.
- Return to the Azure portal and inspect the list of resource groups.
 - You should be able to verify that the resource group named **lab02** has been created.

- c) You should also be able to verify that the **lab02** resource group contains the app service plan, web app and deployment slot.

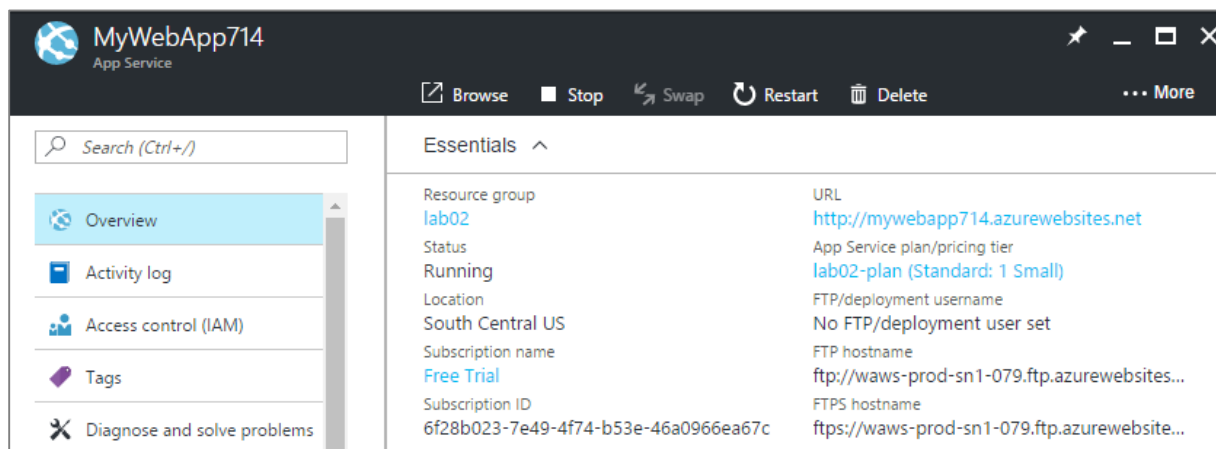


4. Inspect the properties of the new web app.

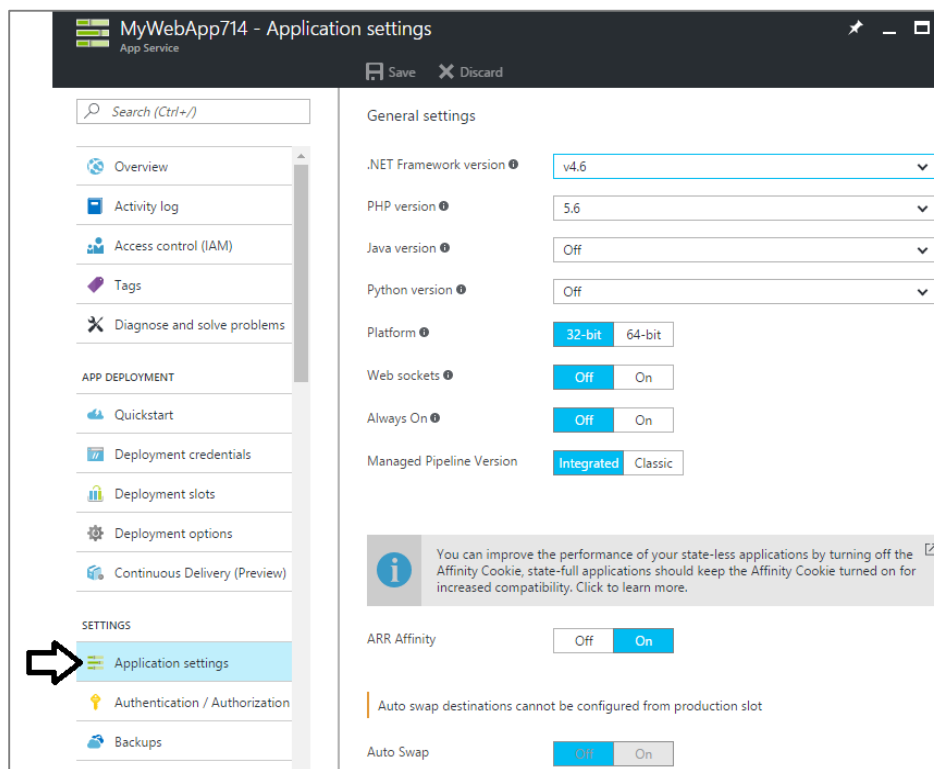
- a) Click on the **App Service** button in the left navigation and locate your web app.



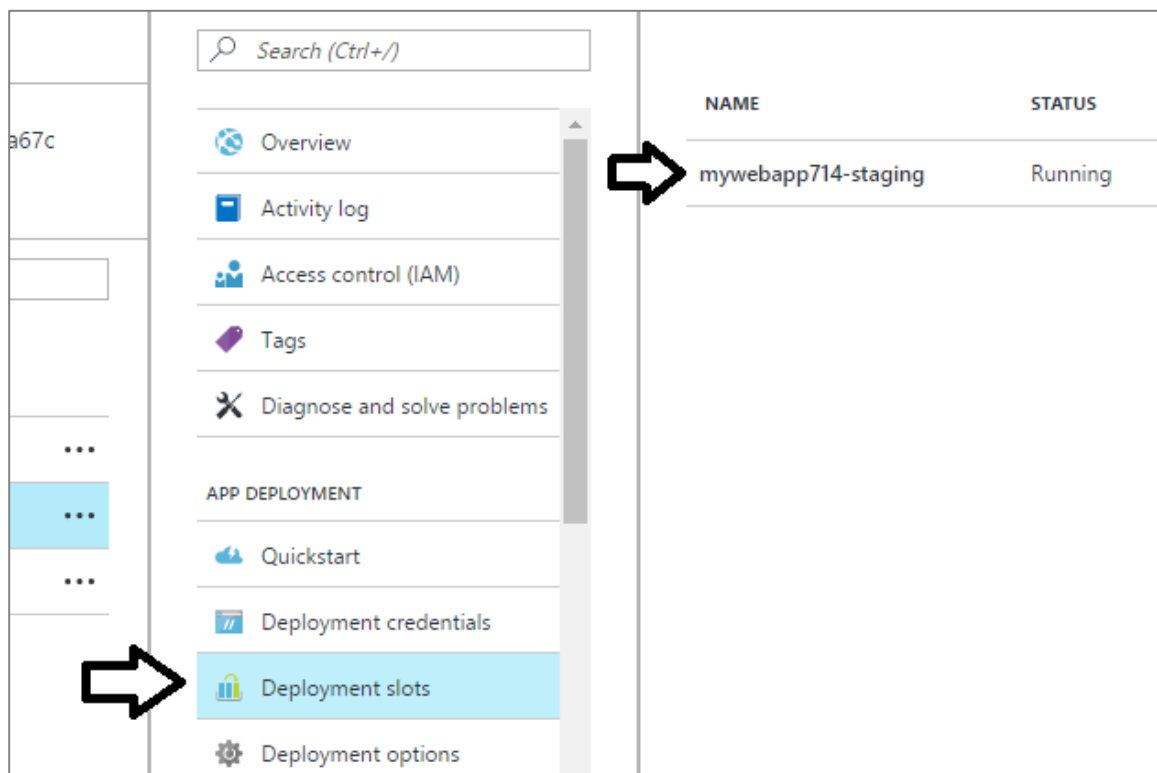
- b) Click on the web app in the **App Service** list to view its properties.
c) Inspect the web app properties shown in the **Overview** section.



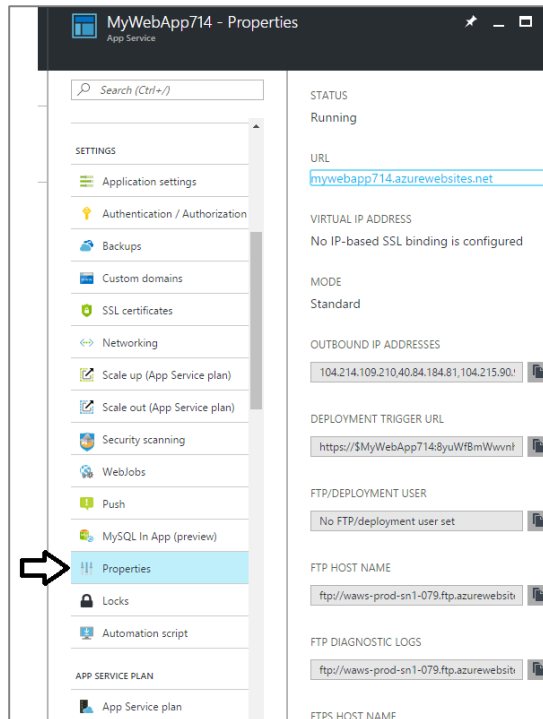
- d) Click on the **Application settings** link on the left to view the application settings for the web app. There is no need to change anything. The purpose of this step is for you to become familiar viewing web app properties in the Azure portal.



- e) Click on the **Deployment slots** link on the left to view the deployment slots for the web app. You should be able to verify that there is a single deployment slot named **staging**.



- f) Click on the **Properties** link on the left to view the general properties for the web app.

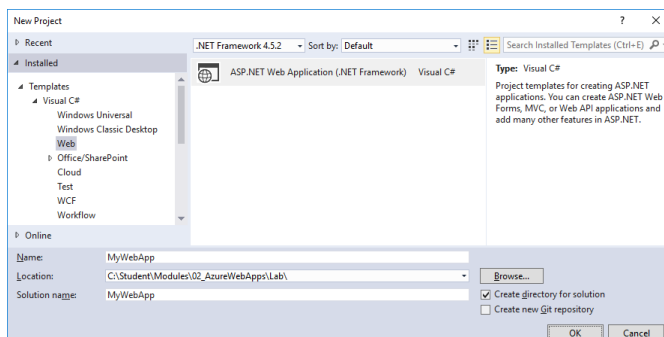


At this point you have seen it's possible to create a web app by hand or by using a PowerShell script. In the next exercise, you will create a new ASP.NET MVC application and then you will move through the steps to deploy it to your new web app.

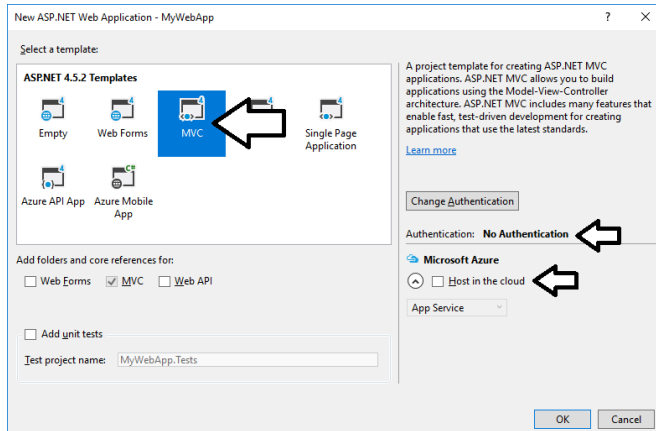
Exercise 3: Deploy a Visual Studio Project to an Azure Web App

In this exercise you will create a new ASP.NET MVC application named **MyWebApp**.

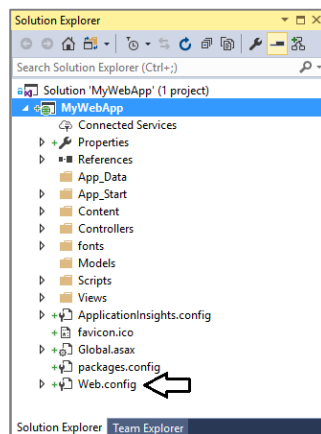
1. Launch Visual Studio 2017.
2. Create a new MVC application named **MyWebApp**.
 - a) From the Visual Studio **File** menu, select the **File > New > Project...** command.
 - b) In the **New Project** dialog, select **Installed > Templates > Visual C# > Web** in the left template menu.
 - c) Select the **ASP.NET Web Application** project template.
 - d) Give the new project a name of **MyWebApp**.
 - e) Set the **Location** to the folder for this lab inside the Student folder.
 - f) Click **OK** to display the **New ASP.NET Web Application** dialog.



- g) In the **New ASP.NET Web Application** dialog, select the template named **MVC**.
- h) Make sure **Authentication** is set to **No Authentication**.
- i) Make sure the **Host in the cloud** checkbox is unchecked.
- j) Click the **OK** button to create the new project.



- k) Wait until Visual Studio has finished creating the new project.
 - l) Once the **MyWebApp** project has been created, take a moment to examine its folder structure in the Solution Explorer.
3. Add an App Setting to the project's main **Web.config** file.
- a) Double click on the **Web.config** file at the bottom of the project to open it in an editor window.



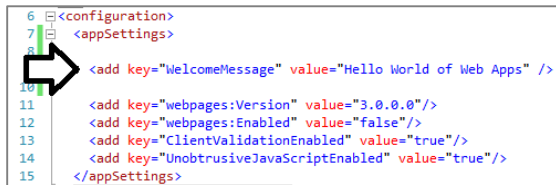
- b) Locate the **appSettings** section at the top of the file.



- c) Add the following XML element to the top of the **appSettings** section.

```
<add key="welcomeMessage" value="Hello world of web Apps" />
```

d) Your edit should now appear in the **appSettings** section as shown in the following screenshot.

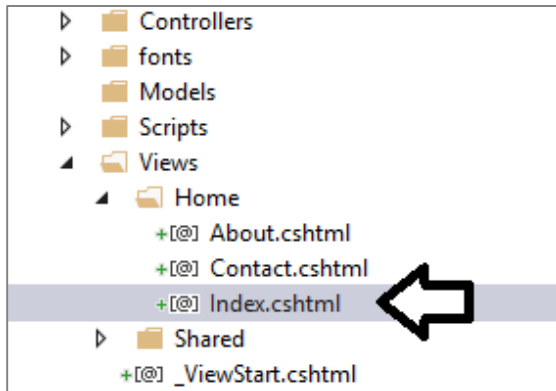


```
6 <configuration>
7 <appSettings>
8   <add key="WelcomeMessage" value="Hello World of Web Apps" />
9
10
11   <add key="webpages:Version" value="3.0.0.0"/>
12   <add key="webpages:Enabled" value="false"/>
13   <add key="ClientValidationEnabled" value="true"/>
14   <add key="UnobtrusiveJavaScriptEnabled" value="true"/>
15 </appSettings>
```

e) Save your changes and close the **WebResource.config** file

4. Modify the view template of the **Home** controller named **Index.cshtml**.

a) In the Solution Explorer, expand the folder path of **Views > Home** and locate the view template file named **Index.cshtml**.



b) Double click on **Index.cshtml** to open it in an editor window.

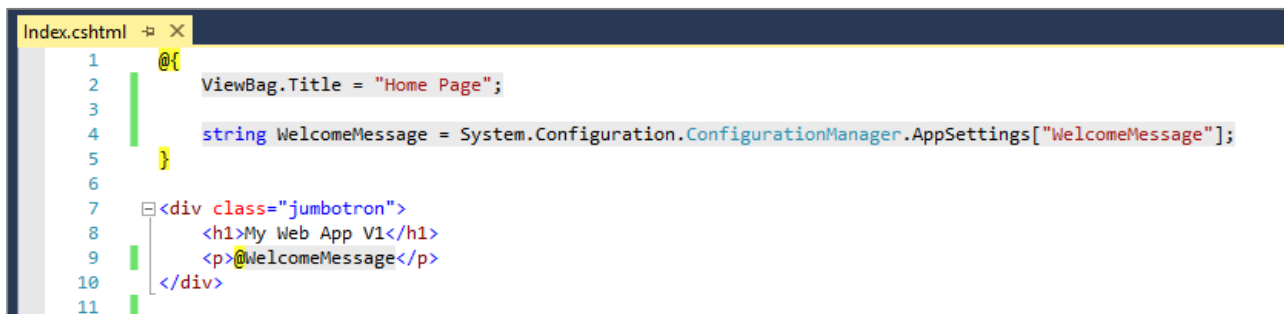
c) Delete all the code inside **Index.cshtml** and replace it with the following code.

```
@{
    ViewBag.Title = "Home Page";

    string welcomeMessage = System.Configuration.ConfigurationManager.AppSettings["WelcomeMessage"];
}

<div class="jumbotron">
    <h1>My Web App V1</h1>
    <p>@welcomeMessage</p>
</div>
```

d) The code you have added to **Index.cshtml** should match the following screenshot.



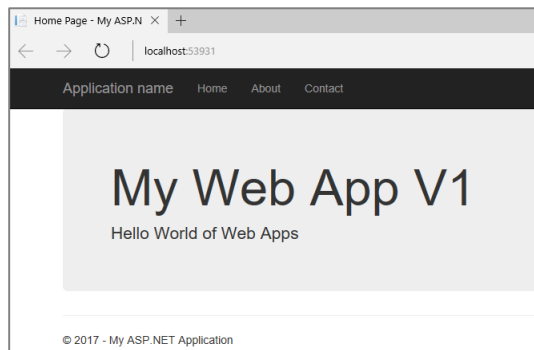
```
Index.cshtml
1  @{
2      ViewBag.Title = "Home Page";
3
4      string WelcomeMessage = System.Configuration.ConfigurationManager.AppSettings["WelcomeMessage"];
5  }
6
7  <div class="jumbotron">
8      <h1>My Web App V1</h1>
9      <p>@WelcomeMessage</p>
10 </div>
11
```

e) Save your work and close **Index.cshtml**.

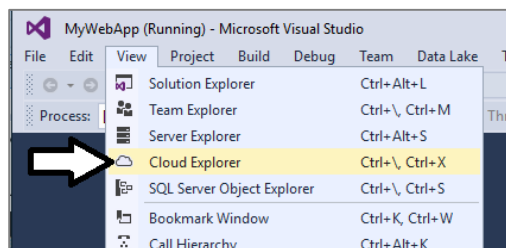
5. Test the **MyWebApp** project using the Visual Studio debugger.

a) Press **{F5}** to start a debugging session.

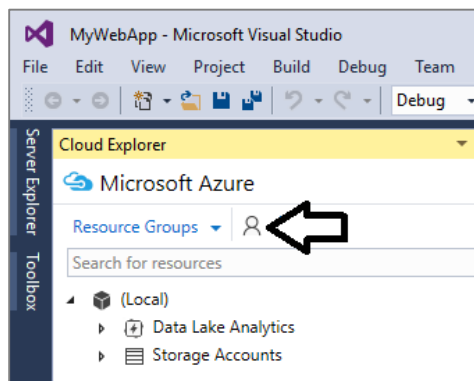
- b) Once the **MyWebApp** project has started up in the Visual Studio debugger, it should match the following screenshot.



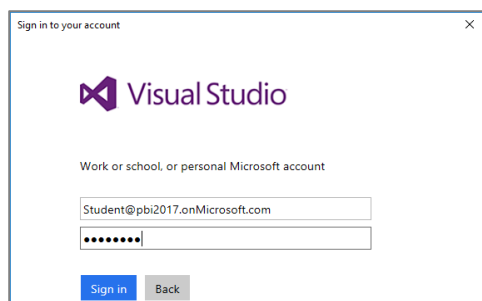
- c) Once you have tested the application, close the browser and then return to Visual Studio and stop the debugger.
6. Connect to Azure from Visual Studio 2017 using the Cloud Explorer.
- a) From the Visual Studio **View** menu, select **View > Cloud Explorer** to display the Cloud Explorer.



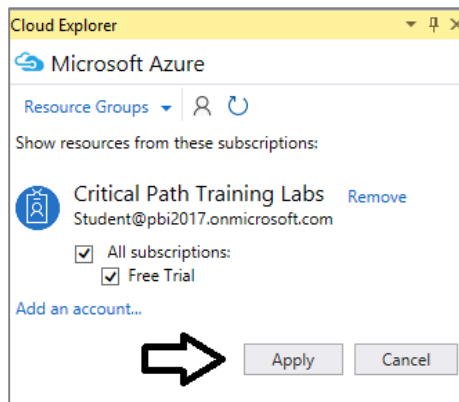
- b) Click the **Azure Accounts Settings** button in the **Cloud Explorer**.



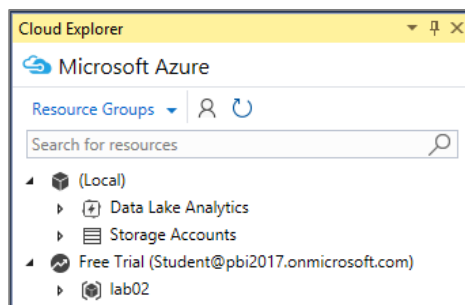
- c) When prompted log into Azure using your primary Office 365 account.



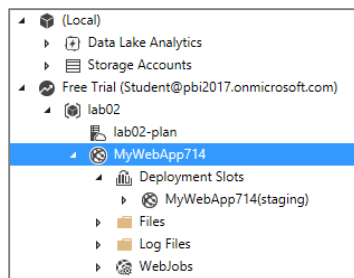
- d) Select your subscription and then click the **Apply** button.



- e) You should now see a new node in Cloud Explorer showing your Azure subscription and the resource group named **lab02**.

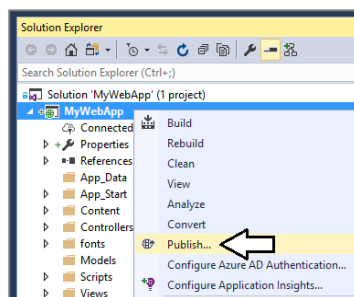


- f) If you drill into the node for your Azure subscription, you should also be able to locate the app service plan, the web app and the deployment slot that were created earlier using the PowerShell script.

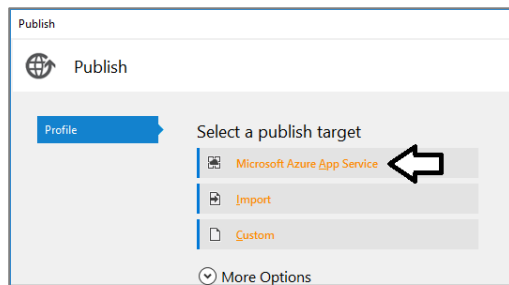


7. Publish the **MyWebApp** project to your web app in Azure.

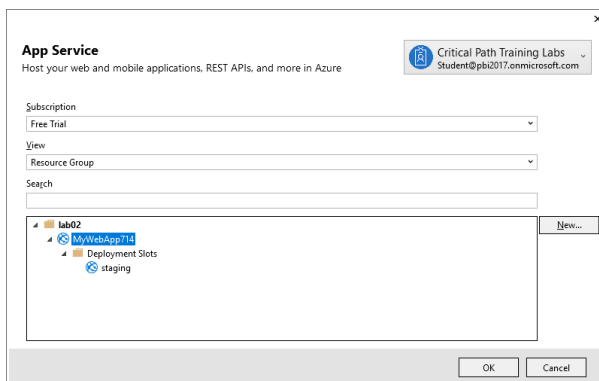
- a) In Solution Explorer, right-click on the project node of the **MyWebApp** project and select the **Publish...** command.



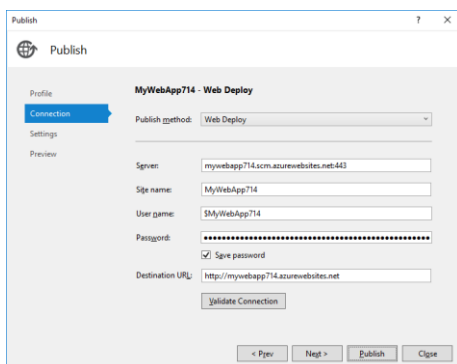
- b) When the **Publish** dialog appears, select the **publish target** named **Microsoft Azure App Service**.



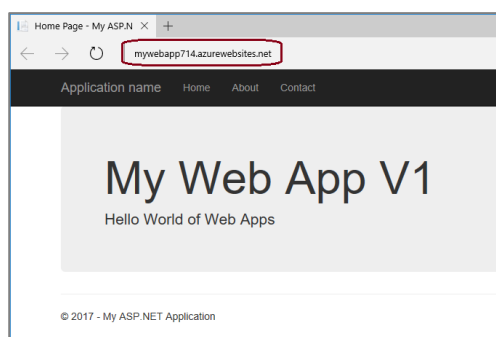
- c) When the **App Service** dialog appears, select your web app as shown in the following screenshot and then click **OK**.



- d) Review the connection settings in the **Connections** tab of the **Publish** dialog and then click the **Publish** button.

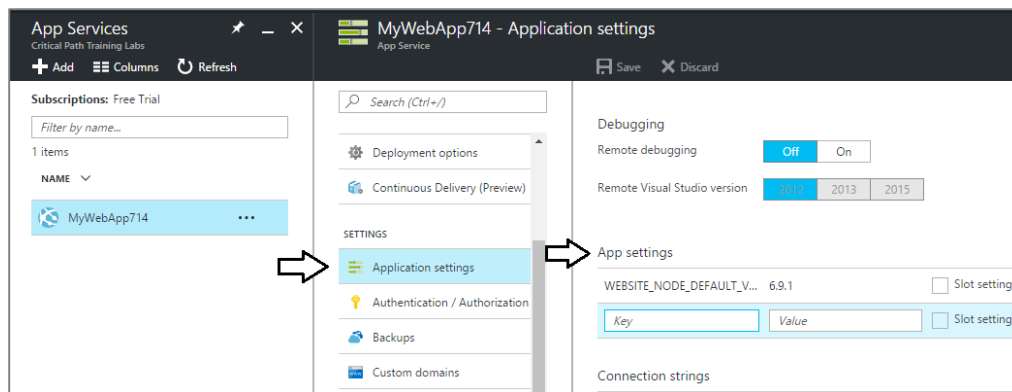


- e) When the publish process completes, Visual Studio should launch a browser and redirect you to the URL of the web app.

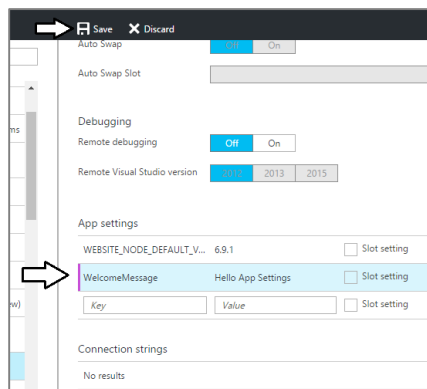


You should see the web app appear just as it did in the Visual Studio Debugger. The only difference is that now the web app has been deployed to a URL where it is accessible to any user on the Internet.

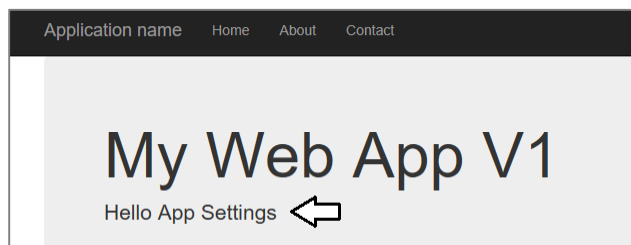
- f) Leave the browser window with the web app open because you will be returning back to it in the next few steps.
8. Modify the **WelcomePage** application setting using the Azure portal.
- a) In a separate browser window, navigate to the Azure portal at <https://portal.azure.com>.
- b) Click on the **App Services** button in the left navigation menu.
- c) Click on your web app to select it.
- d) Click on **Application setting** on the right.
- e) Scroll down and locate the **App settings** section for the web app.



- f) Add a new **App setting** named **WelcomeMessage** with a value of **Hello App Settings**.
- g) Click the **Save** button at the top of the page as shown in the following screenshot.



- h) Return to the browser window with your web app and refresh the page. The text at the bottom should now match the App setting value you just entered. A key point here is that an app setting value added to a web app configuration will always override an **applicationSetting** value with the same name in the project's **Web.config** file.



9. Modify the **WelcomeMessage** application setting using a PowerShell Script.

- Launch the PowerShell ISE.
- In the PowerShell ISE, open the script named **Update-AppSetting.ps1** located at the following path.

```
C:\Student\Modules\02_AzureWebApps\Lab\Update-AppSetting.ps1
```

- Update the first two lines with your tenant name and your primary office 365 account name. Also update the **\$webAppName** variable using the web app name you used in the previous lab.

```
$tenantName = "[Your Tenant]"
$tenantAdminAccountName = "[Your User Account]"
$tenantDomain = $tenantName + ".onmicrosoft.com"
$tenantAdminSPN = $tenantAdminAccountName + "@" + $tenantDomain

$resourceGroupName = "lab02"
$webAppName = "MyWebApp714"
```

- Look at the two lines of code at the bottom of the script which demonstrate how to update an app setting with PowerShell.

```
$AppSettings = @{"WelcomeMessage" = "Hello App Settings Modified by a PowerShell Script" }
Set-AzureRmWebApp -ResourceGroupName $resourceGroupName -Name $webAppName -AppSettings $AppSettings
```

- Save your changes to **Update-AppSetting.ps1**.
- Execute **Update-AppSetting.ps1** using the PowerShell ISE to update the **WelcomeMessage** app setting.
- After **Update-AppSetting.ps1** has executed, return to the browser window with your web app and refresh the page. You should now see the updated welcome message indicating the PowerShell script was able to modify the app setting.

My Web App V1

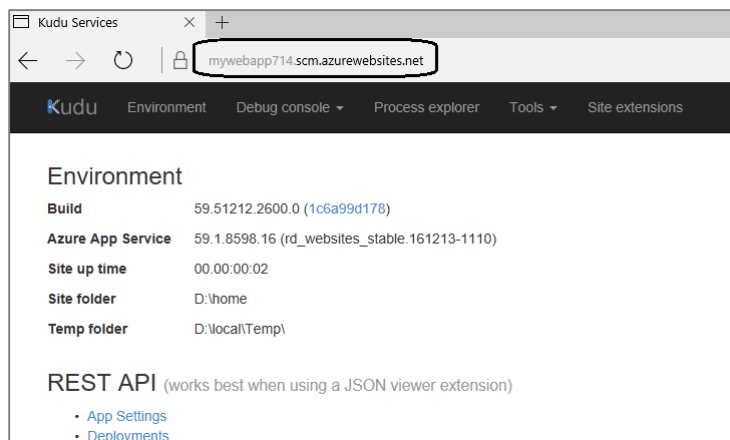
Hello App Settings Modified by a PowerShell Script

10. Examine your Azure web app using the **Kudo** web application.

- In the browser window that is hosting your web app, modify the URL in the address bar by adding **.scm** and a dot (.) just before the last part of the URL which is **azurewebsites.net**. For example, if your website is named **mywebapp714**, you should modify the URL to match the following URL.

```
https://mywebapp714.scm.azurewebsites.net/
```

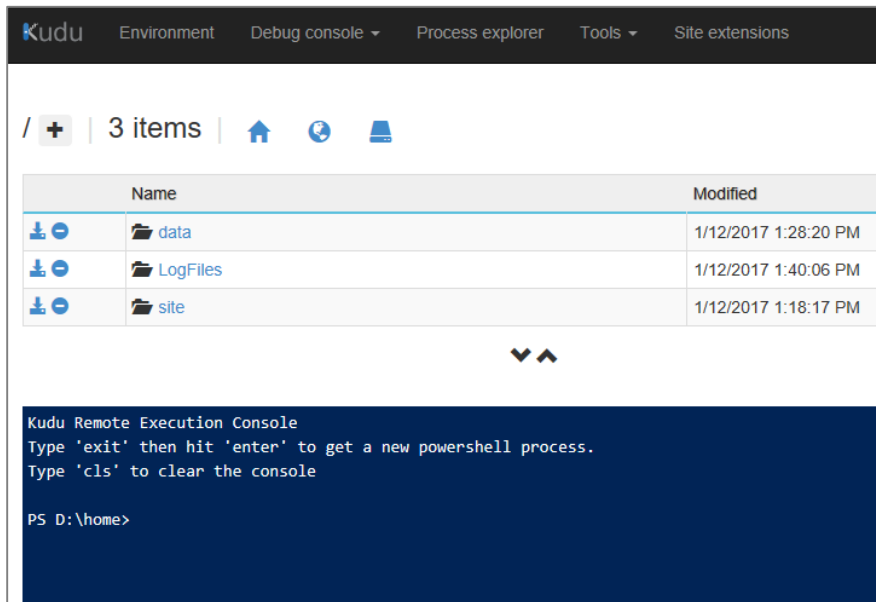
- When you attempt to navigate to the new URL for the Kudo application, you will be prompted to authenticate. Unlike the web app you just deployed, the Kudo app does not support anonymous access. Log in using your primary Office 365 account.
- Take a minute to explore the Kudo application. The first page shows information about the web app environment.



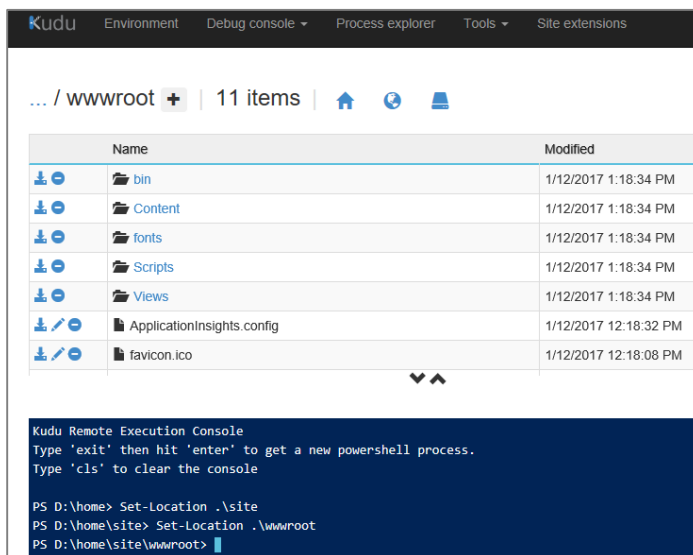
- d) Click the **Debug console** dropdown menu in the top navigation menu and click PowerShell.



- e) You can see that the PowerShell debug console displays the top level folders in the web app's underlying storage structure.



- f) Place your cursor at the prompt in the PowerShell console and type **Set-Location .\site** to move into the **site** folder.
- g) Next, type **Set-Location .\wwwroot** to move into the **wwwroot** folder which is the root folder of the **MyWebApp** project.

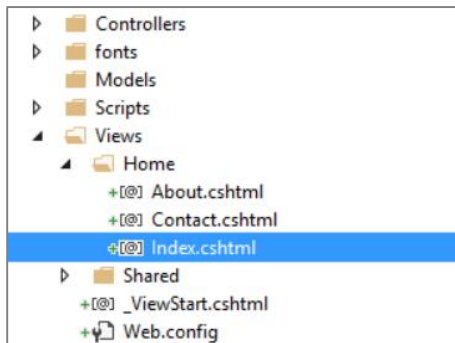


At this point, the lab does not have you do anything else using the Kudu application. However, you should keep in mind that Kudu can be used to perform maintenance operation on a live site such as making edits to files such as **Index.cshtml** and **Web.config**.

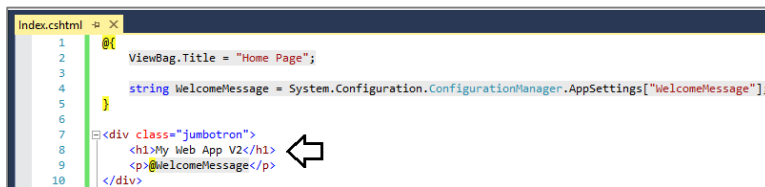
Exercise 4: Managing Staged Deployments using Deployment Slots

In this exercise you will make a few minor changes to the **MyWebApp** project in order to create an updated version of the application code. This will give you an opportunity to deploy the second version of the application code in a staged fashion using deployment slots.

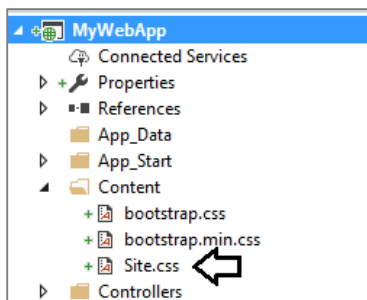
1. Return to Visual Studio and make sure the **MyWebApp** project is still open.
2. Make a small (yet obvious) update to the view template file named **Index.cshtml**.
 - a) In Solution Explorer, double-click on the **Index.cshtml** file in the **Views > Home** folder to open it inside an editor window.



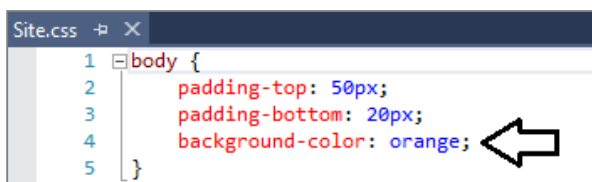
- b) Change the contents of the **h1** tag from "**My Web App V1**" to "**My Web App V2**".



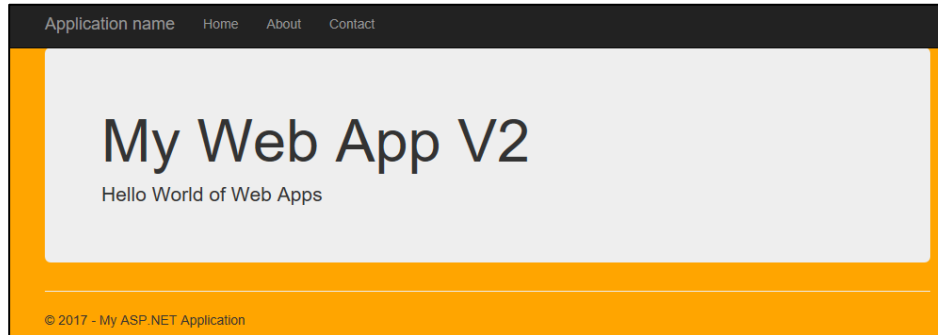
- c) Save your changes and close **Index.cshtml**.
3. Make another small (yet obvious) update to the application's main CSS file named **Site.css**.
 - a) In Solution Explorer, double-click on the **Site.css** file in the **Content** folder to open it inside an editor window.



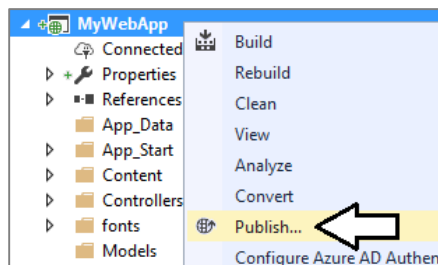
- b) You should be able to see that that **body** tag at the top already has style properties for **padding-top** and **padding-bottom**.
- c) Add a third style property for **background-property** and give it a value of **orange** (or another color that you like better).



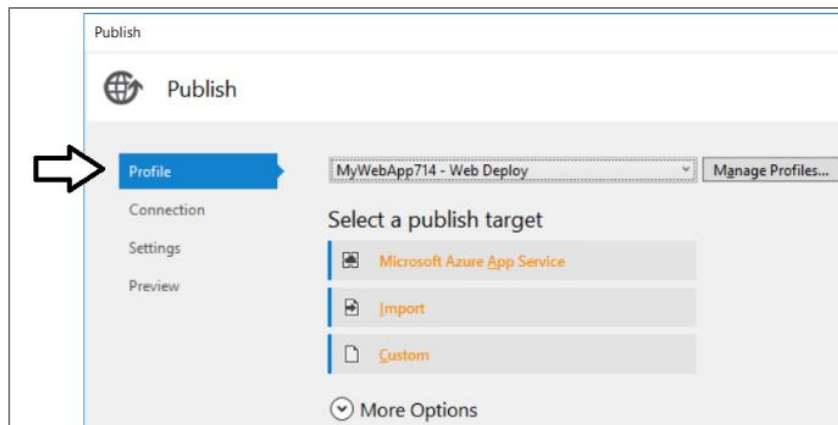
- d) Save your changes and close **Site.css**.
- 4. Test the updated version of the **MyWebApp** project in the Visual Studio debugger.
 - a) Press the **{F5}** key to begin a new debugging session.
 - b) One the application loads, verify that you see the changes to **Index.cshtml** and **Site.css**.



- c) Once you have tested the application, close the browser, return to Visual Studio and stop the debugger.
- 5. Deploy the updated version of the **MyWebApp** project to the **staged** deployment slot.
 - a) In Solution Explorer, right-click on the project node of the **MyWebApp** project and select the **Publish...** command.

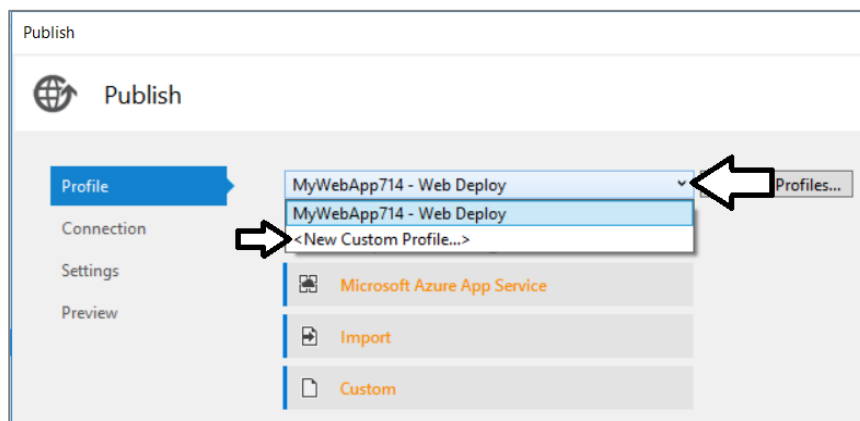


- b) When the **Publish** dialog appears, select the **Profile** tab as shown in the following screenshot.

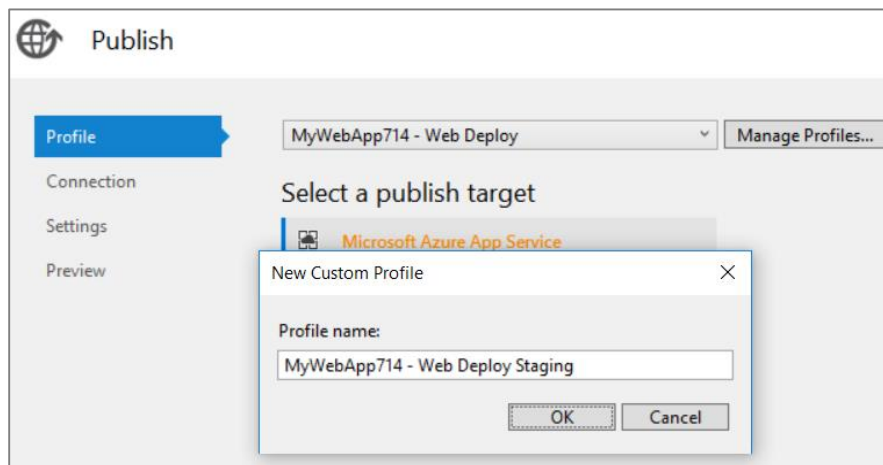


You have already created the publishing profile named **MyWebApp714 - Web Deploy** which has been configured to deploy the project's application files to the default deployment slot named **production** for the target web app. You will now create a second publishing profile to deploy the project's application files to the secondary deployment slot named **staging**.

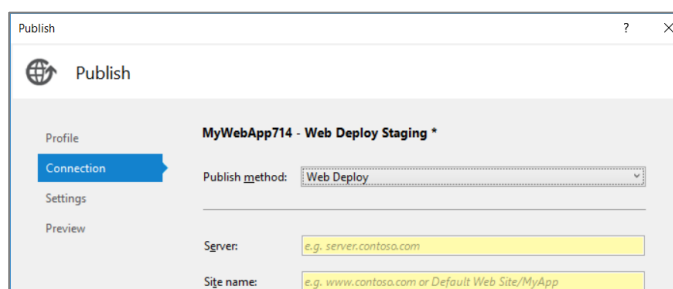
- c) Click the dropdown menu with the profile name and select **<New Custom Profile... >**.



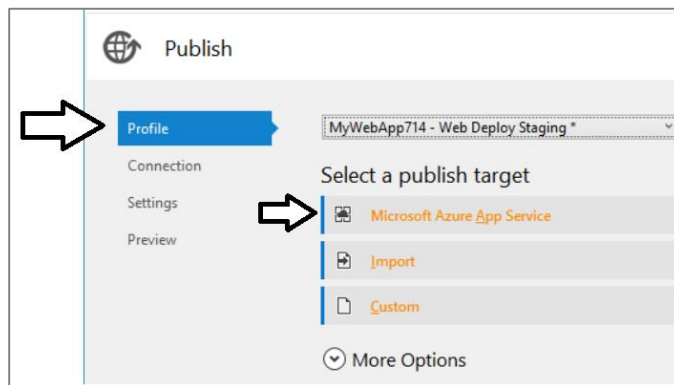
- d) In the **New Custom Profile** dialog, enter a name such as "**MyWebApp714 – Web Deploy Staging**" and click **OK**.



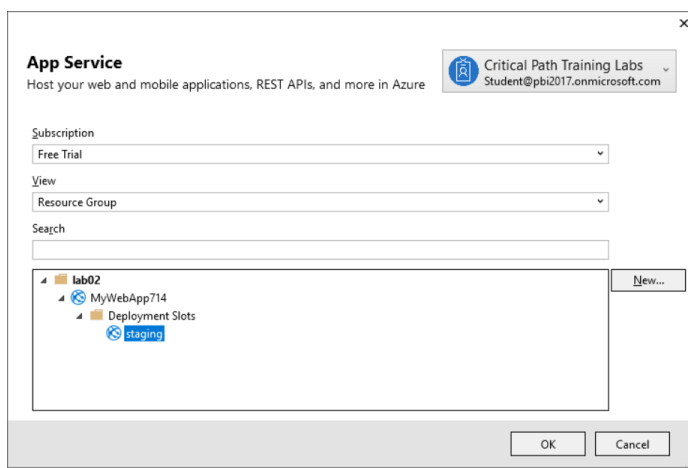
- e) When you click **OK** to create the new profile, Visual Studio moves ahead to the **Connections** tab. However, you can see there is no connection information yet for important settings such as **Server** and **Site name**.



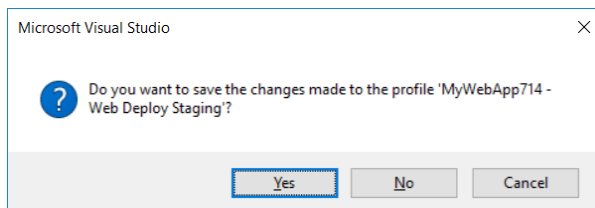
- f) Click on the **Profile** tab of the **Publish** dialog and then select the **publish target** of **Microsoft Azure App Service**.



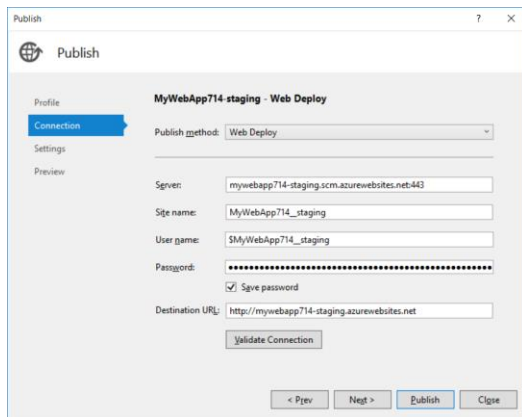
g) In the **App Service** dialog, select the **staging** deployment slot as shown in the following screenshot.



h) When prompted whether you want to save the changes to your profile, click **Yes**.



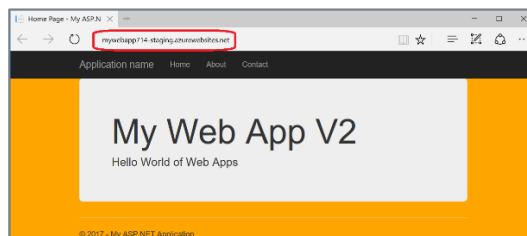
i) Now you should see that Visual Studio has configured the connection information on the **Connection** tab.



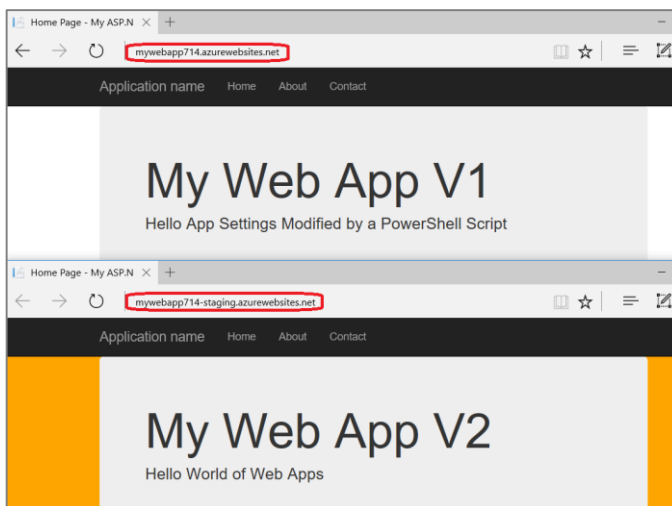
- j) Click the **Publish** button in the **Publish** dialog to deploy updated code for **MyWebApp** to the **staging** deployment slot.

When the publishing process completes, Visual Studio will redirect you to the **staging** deployment slot.

- k) You should now see the updated version of **MyWebApp** running inside the browser.
l) Examine the URL in the address bar and verify the URL ends with **-staging.azurewebsites.net**.



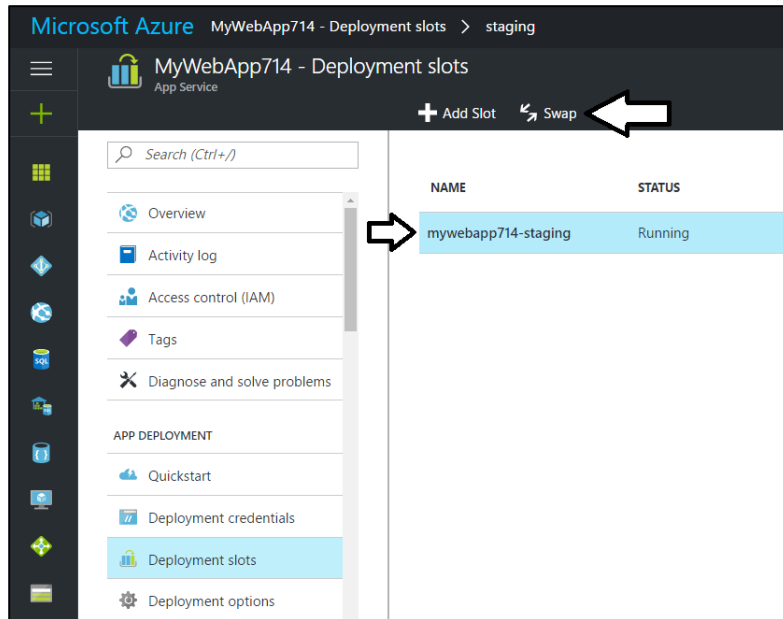
6. Position two browser windows on the screen so you can compare the production slot with the staging slot.
a) In the first browser window, navigation to the production slot at <http://mywebapp714.azurewebsites.net/>.
b) In the second browser window, navigation to the production slot at <http://mywebapp714-staging.azurewebsites.net/>.



At this point, the **production** slot of the web app has the files for **V1** of the MVC application while the **staging** slot has the files for **V2**. Over the next few steps, you will test swapping out these two slots and you will see that V1 and V2 switch places.

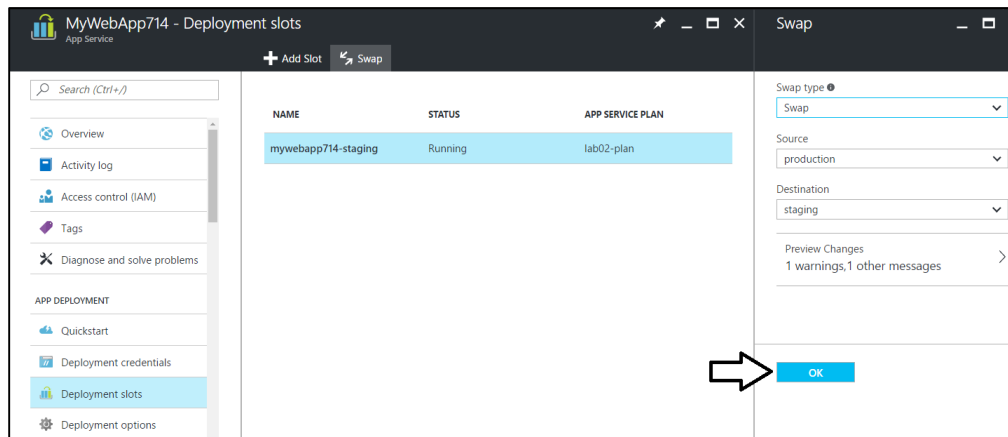
7. Swap the **staging** and **production** deployment slots using the Azure portal.

- Open a third browser window and navigate to the Azure portal.
- Locate and select the web app you created earlier.
- Click on **Deployment slots** and then select the **staging** deployment slot.
- Click the **Swap** button above in the toolbar.

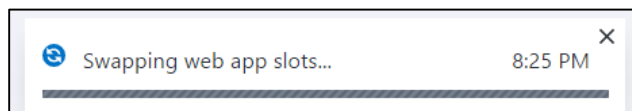


Note that clicking the **Swap** button does not actually perform the swap. Instead, it opens the **Swap** blade.

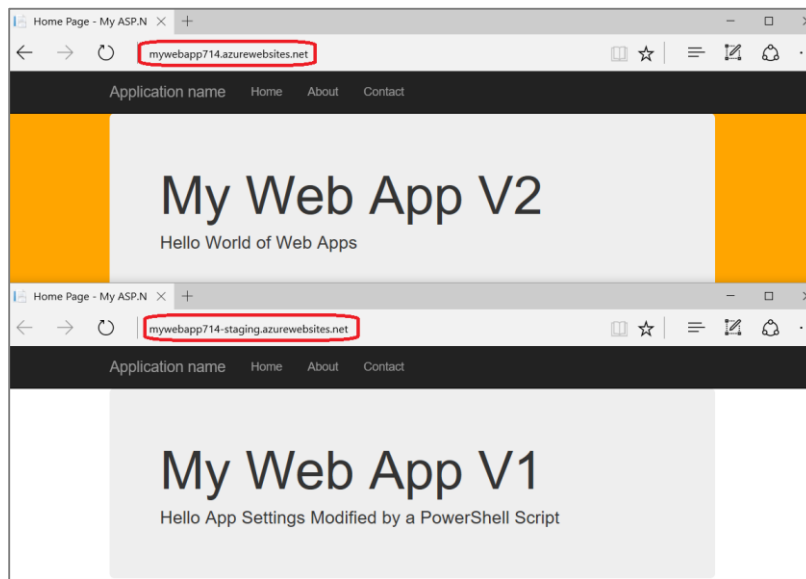
- On the **Swap** blade, click **OK** to perform the swap.



- Note that the Azure portal will provide notification that the swap operation is underway as well as when it completes.



- Wait until the swap operation completes and then move ahead to the next step.
- Return to the two browser windows and refresh them. You should be able to verify that the **production** slot now contains the **V2** deployment while the **staging** slot contains the **V1** deployment.



- i) Return to Azure portal perform the swap operation one more time.
 - j) Return to the two browser windows and refresh them. You should be able to verify that the **production** slot now contains the **V1** deployment while the **staging** slot contains the **V2** deployment.
8. Swap the deployment slots using PowerShell.
- a) Launch the PowerShell ISE.
 - b) In the PowerShell ISE, open the script named **Swap-Deployment-Slots.ps1** located at the following path.

```
C:\Student\Modules\02_AzureWebApps\Lab\Swap-Deployment-Slots.ps1
```

- c) Update the first two lines with your tenant name and your primary office 365 account name. Also update the **\$webAppName** variable using the web app name you used earlier in this lab.

```
$tenantName = "[Your Tenant]"
$tenantAdminAccountName = "[Your User Account]"
$tenantDomain = $tenantName + ".onmicrosoft.com"
$tenantAdminSPN = $tenantAdminAccountName + "@" + $tenantDomain

$resourceGroupName = "lab02"
$webAppName = "MyWebApp714"
$sourceSlotName = "staging"
$destinationSlotName = "production"
```

- d) Examine the PowerShell code that executes the **Swap-AzureRmWebAppSlot** cmdlet to perform the swap operation.

```
Write-Host "Swapping deployment slots..."
Swap-AzureRmWebAppSlot `
  -ResourceGroupName $resourceGroupName `
  -Name $webAppName `
  -SourceSlotName $sourceSlotName `
  -DestinationSlotName $destinationSlotName
```

- e) Execute the PowerShell script.
- f) Return to the two browser windows and refresh them. You should be able to verify that the **production** slot now contains the **V2** deployment while the **staging** slot contains the **V1** deployment.

You can continue to run this PowerShell script to toggle the versions back and forth between the **production** slot and the **staging** slot.

Exercise 5: Create a Virtual Machine using PowerShell

In this exercise you will modify and run a PowerShell script to create an Azure virtual machine. As you will see, this can become a complex process because creating a virtual machines requires creating several other resources for storage and network connectivity.

1. Open and update the PowerShell script named **Create-And-Start-VM.ps1**.
 - a) Launch the PowerShell ISE if it is not already running.
 - b) In the PowerShell ISE, open the script named **Create-And-Start-VM.ps1** located at the following path.

```
C:\Student\Modules\02_AzureWebApps\Lab\Create-And-Start-VM.ps1
```

- c) Update the first two lines with your tenant name and your primary office 365 account name.
 - d) Also update the **\$location** variable if that is appropriate given where you live.

```
$tenantName = "[Your Tenant]"
$tenantAdminAccountName = "[Your User Account]"
$tenantDomain = $tenantName + ".onmicrosoft.com"
$tenantAdminSPN = $tenantAdminAccountName + "@" + $tenantDomain

$location = "southcentralus"
$resourceGroupName = "lab02-vm"
```

- e) Take some time to review this script to see all that is involved with creating a virtual machine instance.

While you reviewing the script make sure to look at the virtual machine image SKU. That is the setting in the PowerShell script that determines what type of virtual machine will be created.

- f) Look down in the script and locate the variable named **\$vmAdminLoginName** and **\$vmAdminPassword**. These credential values are important because you will be using them later when you log into the VM with an RDP connection.

```
$vmAdminLoginName = "CptStudent"
$vmAdminPassword = "pass@word1234"
```

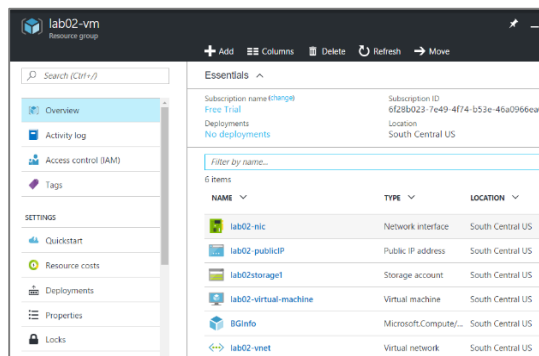
- g) Save your changes to **Create-And-Start-VM.ps1**.
2. Execute the PowerShell Script **Create-And-Start-VM.ps1** from within the PowerShell ISE.
 - a) Inside the PowerShell ISE, execute the script named **Create-And-Start-VM.ps1**.
 - b) Wait until the script completed its execution.

This script will likely take 5-10 minutes to run due to all the resources that must be provisioned.

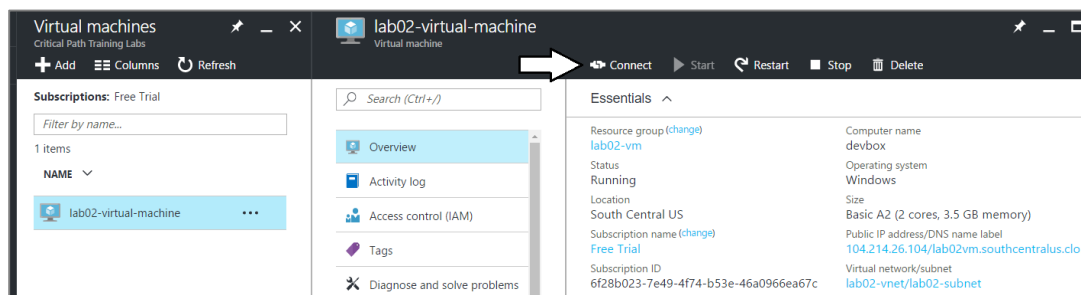
- c) When the script completes, you should be able to see output in the console window that matches the following screenshot.

```
Resource group named lab02-vm does not exist - now creating it
Storage account name lab02storage already in use
Calling New-AzureRmStorageAccount to create a storage account named lab02storage1
Calling New-AzureRmVirtualNetworkSubnetConfig to create subnet named lab02-subnet
Calling New-AzureRmVirtualNetwork to create Virtual network name lab02-vnet
Calling New-AzureRmPublicIpAddress to create static public IP address with domain label name of lab02vm
Calling New-AzureRmNetworkInterface to create network interface named lab02-nic
Calling New-AzureRmVMConfig to create new VM configuration with name of lab02-virtual-machine and size of Basic_A2
Calling Set-AzureRmVMSourceImage to configure VM template
Calling Add-AzureRmVMNetworkInterface to configure VM with network connectivity
Calling Set-AzureRmVMOSDisk to configure creating VM from VM image
Calling New-AzureRmVM to create and start VM
All done - VM has been provisioned
```

3. Navigate to the Azure portal in the browser and examine the set of resources in the resource group named **lab02-vm**. These are the resources that were created in order to support the new virtual machine.

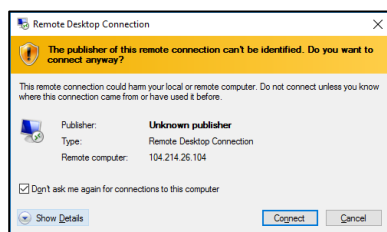


4. Examine the new virtual machine and establish an RDP connection.
 - a) Select the virtual machine named **lab02-virtual-machine** and examine its **Overview** blade.
 - b) Examine the property settings for the VM in the **Essentials** blade.
 - c) Find and click the **Connect** button to establish an RDP connection with the VM.

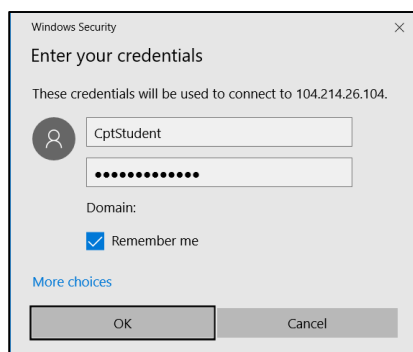


You might have to enable the browser to allow all popup windows from portal.azure.com for the **Connect** button to work properly.

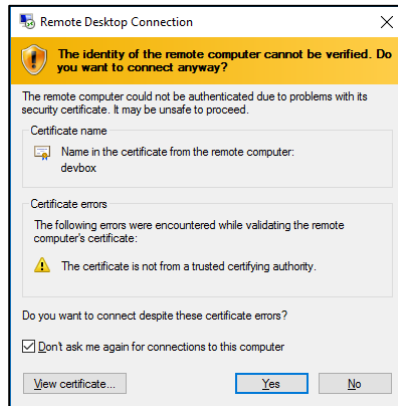
- d) When prompted by the **Remote Desktop Connection** dialog, click **Connect**.



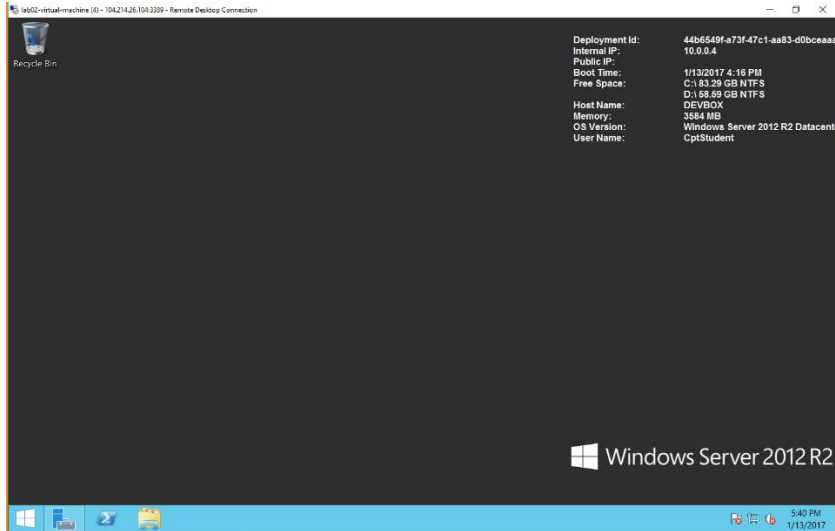
- e) In the **Enter your credentials** dialog, enter a user name of "**CptStudent**" and a password of "**pass@word1234**".



- f) In the **Remote Desktop Connection** dialog, check the checkbox with the caption **Don't ask me again for connections to this computer** and click Yes to login into the VM.

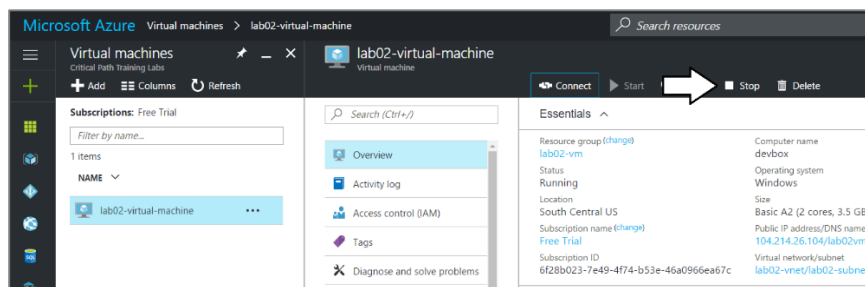


- g) You should now be able to access the VM using the new RDP connection.



Once you are able to access the VM, the lab exercise steps are over. Feel free to play around with the VM. However, make sure you don't leave this virtual machine running overnight if it isn't essential because running VMs in Azure can be expensive.

5. Return to the Azure portal and stop the VM from running by clicking the Stop button.



6. Note you can also stop all your VMs at once using the following PowerShell code.

```
foreach($group in Get-AzureRmResourceGroup){  
    foreach($vm in Get-AzureRmVM -ResourceGroupName $group.ResourceGroupName){
```

```
    write-Host "Stopping VM named" $vm.Name "..."  
    Stop-AzureRmVM -ResourceGroupName $resourceGroupName -Name $vm.Name -Force  
  }  
}  
  
write-Host "All VMs have been stopped"
```

7. Once again, remember to shut down your VMs when you are not using them to keep your costs down.

Congratulations. You have now completed this lab.