

Module 05 - Azure Resource Manager

Exercise 1: Install Visual Studio Code

Introduction

In this lab, we will look at how to install Visual Studio Code and the Azure Resource Manager extension.

Summary

Visual Studio is a free and cross-platform editor. It has an excellent extension to help write Azure Resource Manager (ARM) templates. In this lab we will install Visual Studio Code and the ARM extension.

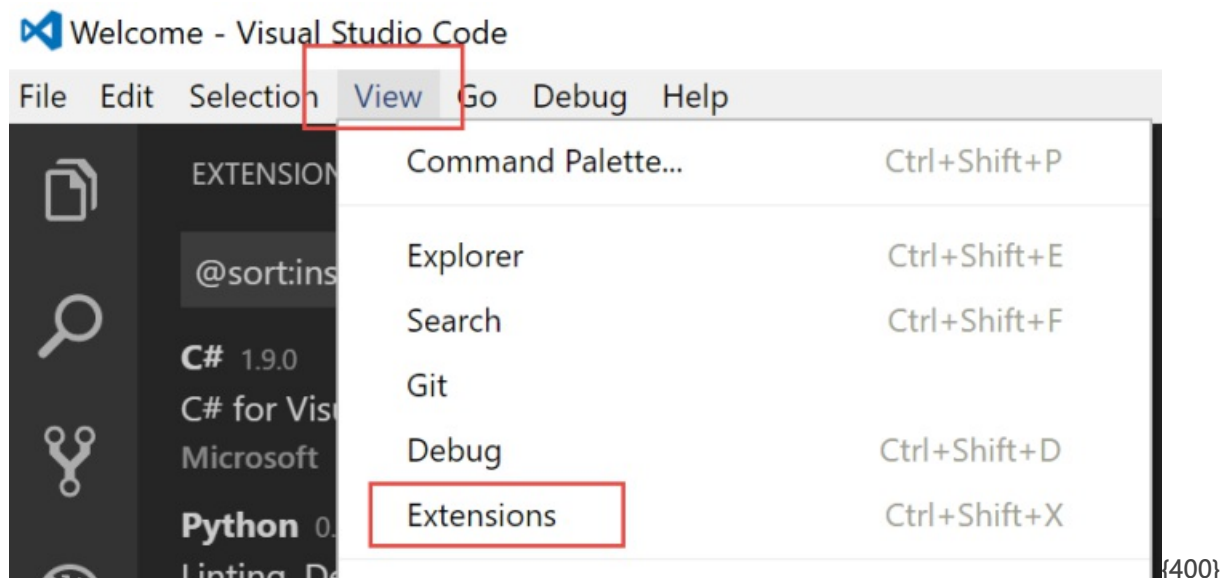
Estimated Time to Complete This Lab

10 minutes

Task Description

[!note] If you already have Visual Studio code installed on your Host PC skip to step 4 unless you are on an LOD machine in which case the extensions are already installed, and you can skip to Lab 2

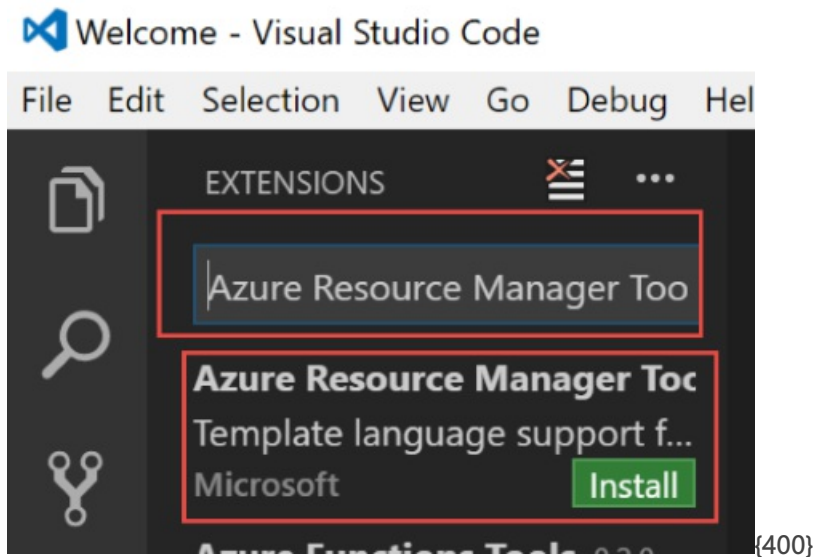
1. [] On your lab machine, open an Internet browser and go to <https://code.visualstudio.com/>.
2. [] Click the download link to download and for you to choose your platform of choice. Run the installer and complete the installation of Visual Studio Code on your lab machine.
3. [] Once installed, open Visual Studio Code.
4. [] Select **View** > **Extensions**.



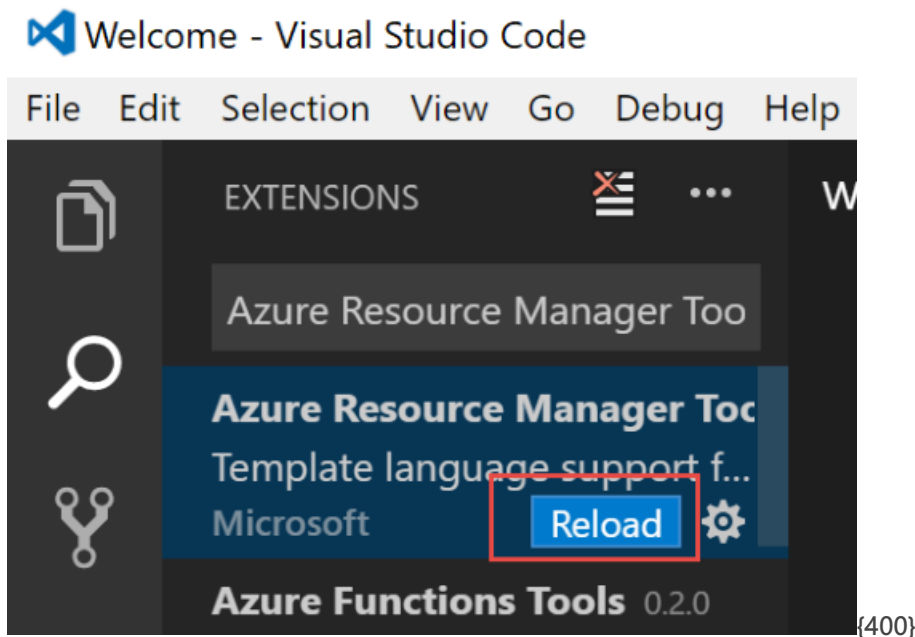
5. [] Delete the entry in the search box and type in **Azure Resource Manager Tools** then click on **Install**

Module 5: Azure Resource Manager (ARM)

on the Azure Resource Manager Tools extension.



6. Click **Reload** to activate the extension and select **Reload Window**.



7. You have now installed Visual Studio Code and the Azure Resource Manager extension. We will make use of these in the next lab.

Exercise 2: Deploy Azure Resource Manager Template

Introduction

Azure Resource Manager (ARM) templates define your *Infrastructure as Code*. In this lab, we will download an example ARM template from the Quick Start Templates. We will then make some simple edits to the template, and deploy it to Azure.

Scenario

In this lab, we will:

Module 5: Azure Resource Manager (ARM)

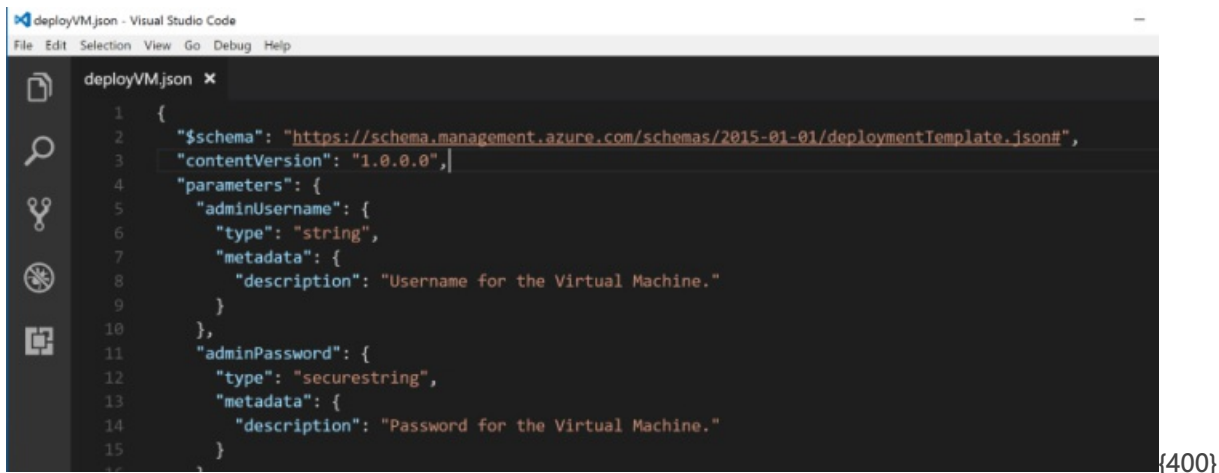
- Download an ARM template from the Quick Start Templates site
- Open the template in Visual Studio Code
- Set parameters
- Deploy the ARM template

Estimated Time to Complete This Lab

30 minutes

Task Description


1. ☐ On your lab machine, go to the GitHub site for Azure Quick Start Templates to view sample templates: <https://github.com/Azure/azure-quickstart-templates>
2. ☐ For this exercise we will use the templates in C:\labs\module5\Templates
3. ☐ Double click the file **deployVM.json** file
4. ☐ You should now be able to see the contents of deployVM.json in VS Code.
5. ☐ You should notice that Visual Studio Code has formatted the JSON file and added coloring:

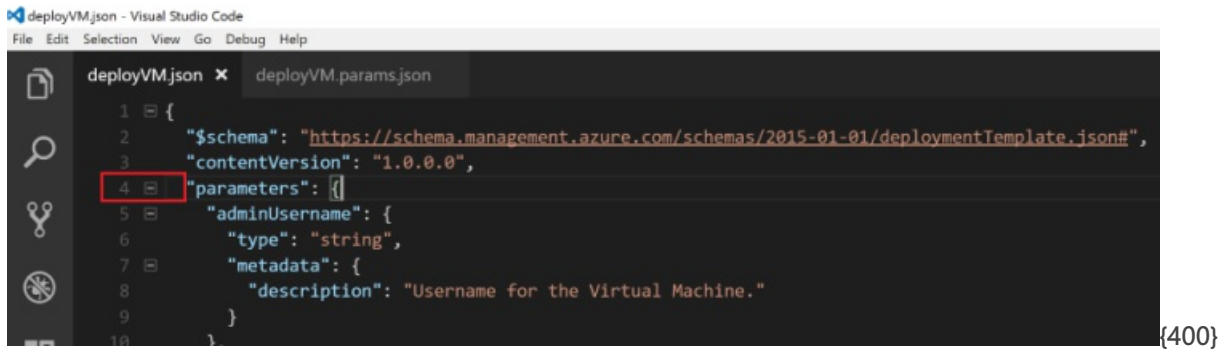


```
1 {
2   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "adminUsername": {
6       "type": "string",
7       "metadata": {
8         "description": "Username for the Virtual Machine."
9       }
10    },
11    "adminPassword": {
12      "type": "securestring",
13      "metadata": {
14        "description": "Password for the Virtual Machine."
15      }
16    }
17  }
```

6. ☐ The ARM language server starts automatically when it detects an ARM template. If you check around line 3 in the file you will see an option to *Select or create a parameter file to enable full validation....* Click on it
7. ☐ In the dropdown menu that appears select **New**
8. ☐ Select **Only required parameters**
9. ☐ Click **Save**
10. ☐ You should now have 2 files saved locally
 - deployVM.json
 - deployVM.parameters.json
11. ☐ In Visual Studio Code, open **deployVM.json**.

Module 5: Azure Resource Manager (ARM)

12. ☐ Notice in Visual Studio Code, that by moving the mouse near the line numbers, you can collapse sections of the JSON file. Towards the top of the file, click the  to hide the **parameters** section:



13. ☐ You should now be able to see the **variables** section of the template. Notice how this has several variables and values set:



14. ☐ Let's alter some of the settings in the JSON template. We can alter the address space of the Virtual Network created by the template: Change the value of **addressPrefix** to **10.71.0.0/16**
15. ☐ Next, let's change the address space of the Subnet it creates: Change the value of **subnetPrefix** to **10.71.0.0/24**
16. ☐ Scroll up to the top of the **deployVM.json** file, expand the **parameters** section, if it is still collapsed from earlier.
17. ☐ Search the file for **vmSize**. Change the value from **Standard_D2_v3** to **Standard_D1_v2**
18. ☐ Here, you can see the several parameters which can be supplied to this template at Runtime. We can see **adminUsername**, **adminPassword**, **dnsLabelPrefix**, **WindowsOSVersion**. We can define the value of these in the parameters file. In Visual Studio Code, go to the **deployVM.params.json** file, which you saved earlier.
19. ☐ You should see there are references for the parameter values. Change the values:

- **adminUsername**: student
- **adminPassword**: RedDwarf2017
- **dnsLabelPrefix**: This needs to be a globally unique name for the VM, as this template gives the VM a public DNS entry. *NOTE*: this can only use *lower case* a-z and numbers 0-9. Use your name and the date to generate a random string:

Module 5: Azure Resource Manager (ARM)

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": {
      "value": "student"
    },
    "adminPassword": {
      "value": "Password.111"
    },
    "dnsLabelPrefix": {
      "value": "johndoe090717"
    }
  }
}
```

{400}

20. [] Save both files in Visual Studio Code.
21. [] We can deploy from PowerShell installed locally - but the Cloud Shell also has the Powershell modules installed and we can upload files to the file share there. In the Azure portal click on the Cloud Shell icon. Click **Reconnect** if prompted.
22. [] Start PowerShell by typing *pwsh* and pressing **Enter**
23. [] Click on the file upload button at the top of the Cloud Shell interface and select Upload.

Bash ▾ | 🔌 ? ⚙️   {} {400}

24. [] Navigate to where you have saved the JSON files, select both of them and click **Open**. The files will be uploaded to the file share in the Cloud Shell.

×

Upload destination: /home/anthony/

deployVM.json	COMPLETE	
deployVM.parameters.json	COMPLETE	{400}

25. [] Change to the directory containing the files by running the command below.

```
Set-Location $HOME
```

26. [] Now we can create the Resource Group we will use to deploy to, enter the following command to create a new resource group in a location close to you.

```
$location = "" #Enter a location to deploy your resources
New-AzResourceGroup -Name 'ARMDeployRG' -Location $location
```

27. [] Next, we are ready to deploy our ARM templates. Enter in the following command to deploy the template file and parameter file to the Resource Group we just created:

```
New-AzResourceGroupDeployment -ResourceGroupName 'ARMDeployRG' -TemplateFile .\deployVM.json
-TemplateParameterFile .\deployVM.parameters.json -Verbose
```

28. [] You should now see that your template is validated and the deployment is happening. Keep an eye on

Module 5: Azure Resource Manager (ARM)

the output to ensure this is successful. It will take several minutes to complete.

29. ☐ Once completed, you should see a success message similar to below:

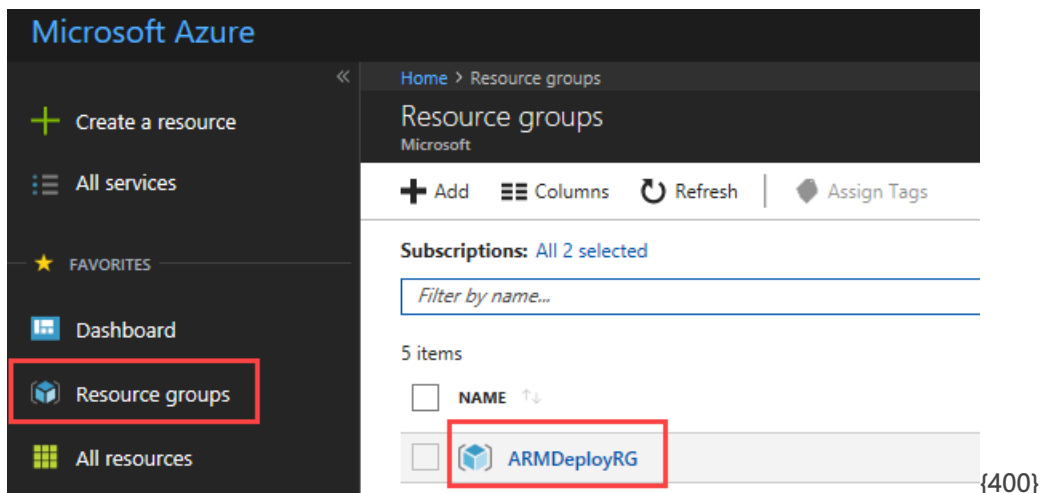
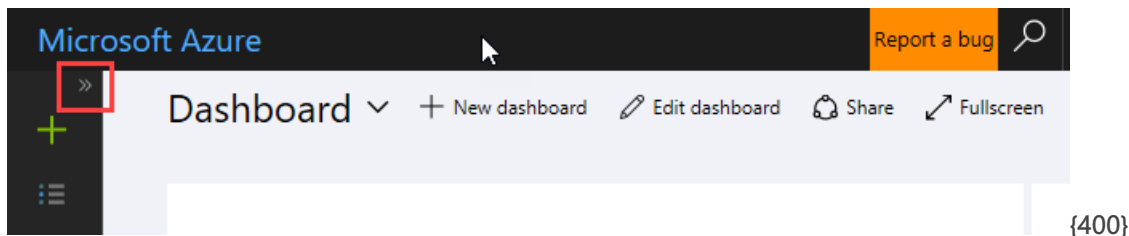
```
DeploymentName      : deployVM
ResourceGroupName   : ARMDeployRG
ProvisioningState    : Succeeded
Timestamp           : 8/21/18 7:19:03 AM
Mode                 : Incremental
TemplateLink         :
Parameters           :
                    Name      Type      Value
                    =====
                    adminUsername String    student
                    adminPassword SecureString
                    dnsLabelPrefix String    awsaug8212018
                    windowsOSVersion String    2016-Datacenter
                    location    String    southeastasia

Outputs              :
                    Name      Type      Value
                    =====
                    hostname    String
awsaug8212018.southeastasia.cloudapp.azure.com

DeploymentDebugLogLevel :
```

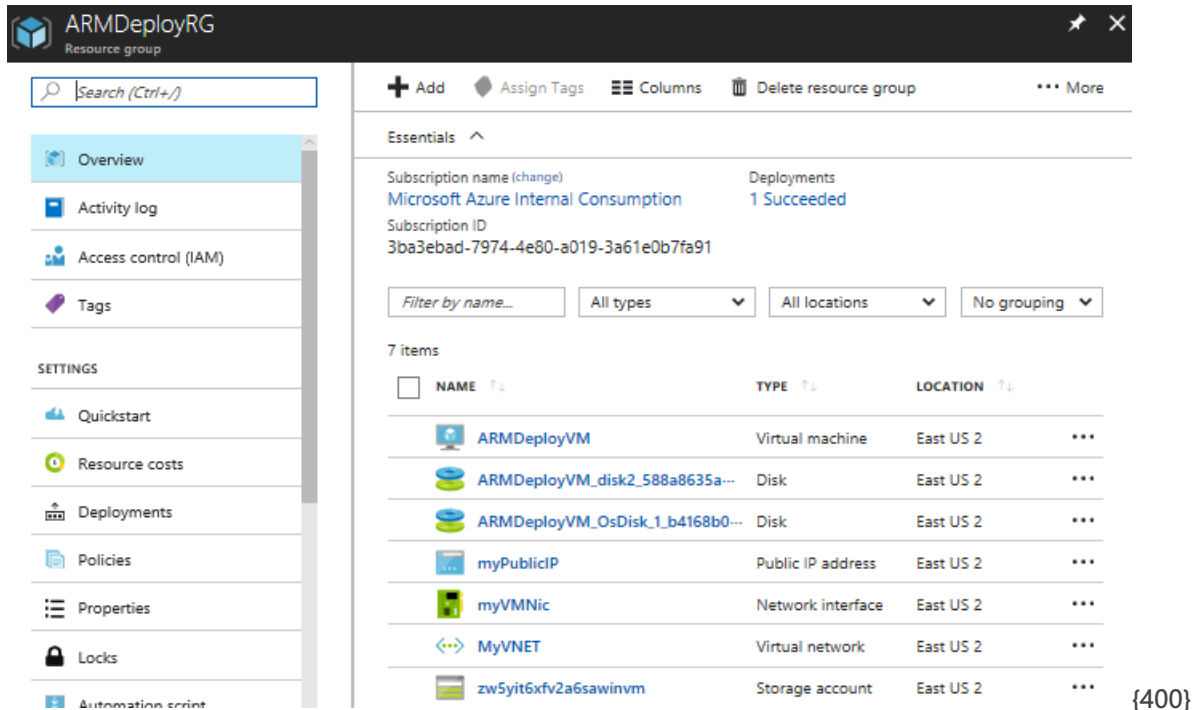
30. ☐ Now, we can validate the resources which were created in the Azure Portal. Go to <https://portal.azure.com>

31. ☐ From the left of the screen, expand the side panel and click **Resource Groups**, then click **ARMDeployRG**



Module 5: Azure Resource Manager (ARM)

32. In your Resource Group, you should find the new Virtual Machine, Network and other components which were deployed:



ARMDeployRG
Resource group

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

SETTINGS

Quickstart

Resource costs

Deployments

Policies

Properties

Locks

Automation script

Essentials

Subscription name (change)
Microsoft Azure Internal Consumption

Deployments
1 Succeeded

Subscription ID
3ba3ebad-7974-4e80-a019-3a61e0b7fa91

Filter by name... All types All locations No grouping

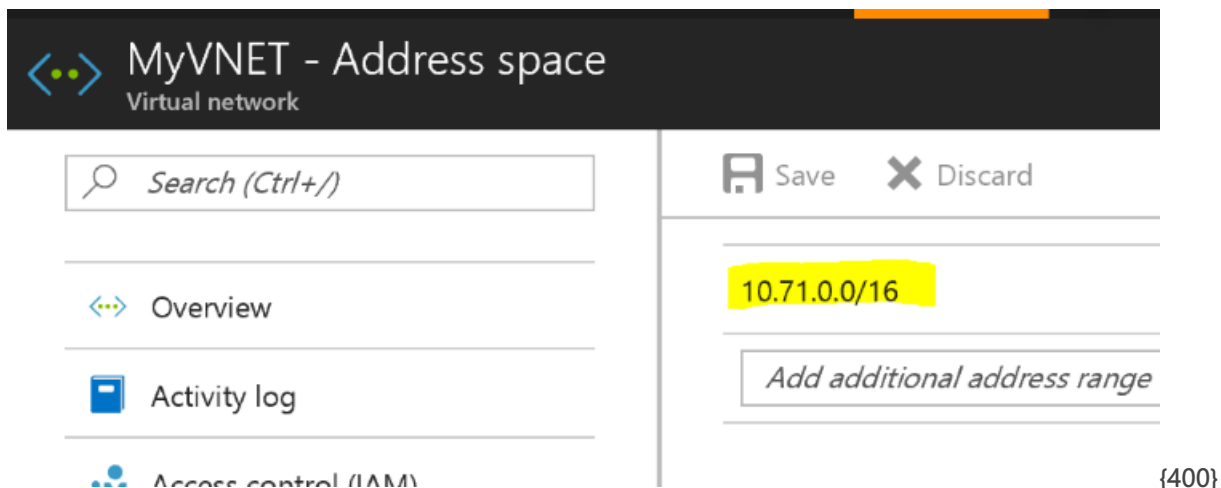
7 items

NAME	TYPE	LOCATION
ARMDeployVM	Virtual machine	East US 2
ARMDeployVM_disk2_588a8635a...	Disk	East US 2
ARMDeployVM_OsDisk_1_b4168b0...	Disk	East US 2
myPublicIP	Public IP address	East US 2
myVMNic	Network interface	East US 2
MyVNET	Virtual network	East US 2
zw5yit6xfv2a6sawinvm	Storage account	East US 2

{400}

33. Click the **MyVNET** resource, to open the Virtual Network created by your template. From the Virtual Network, click **Address space**.

34. Notice that the Address Space is as you defined in your ARM Template:



MyVNET - Address space
Virtual network

Search (Ctrl+/)

Save Discard

10.71.0.0/16

Add additional address range

Overview

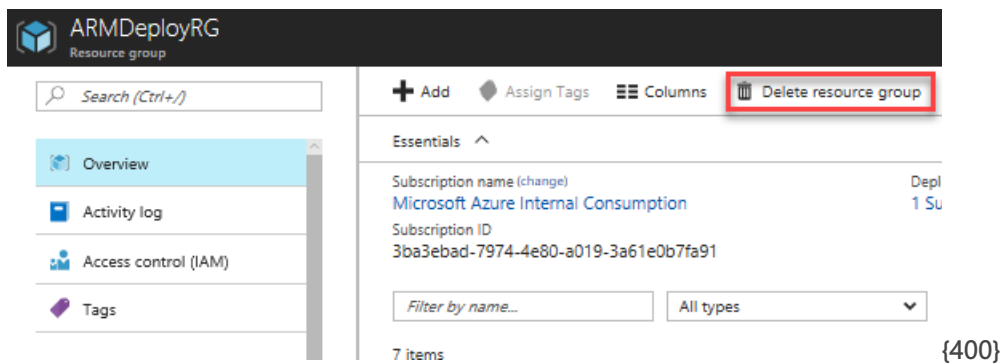
Activity log

Access control (IAM)

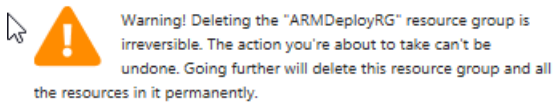
{400}

35. Close the Virtual Network pane to get back to your **Resource Group**. Now we are finished with the deployment, we can delete the Resource Group. Click the **Delete resource group** button:

Module 5: Azure Resource Manager (ARM)



36. [] Enter in the name of the Resource Group and click the Delete button to fully delete the Resource Group and all resources inside it:



37. [] You can also delete the resource group by running the code below.

```
Remove-AzResourceGroup -ResourceGroupName ARMDeployRG -Force
```

In this lab, you downloaded an ARM template from the Quick Start Templates, edited and deployed it. We then deleted the Resource Group to remove all the deployed resources.

Exercise 3: Deploy Adventure Works Website and Database

Introduction

In the previous lab, we saw a simple example of deploying a VM via an ARM Template. In this lab, we will use ARM templates to deploy a full business application. This application will include a Windows Server running an IIS website, and an Azure SQL Database hosting data for the site.

This deployed service will be used in later labs, to perform automation. Please ensure this lab is completed fully. If you have any questions, please contact your instructor.

Scenario

In this lab, we will:

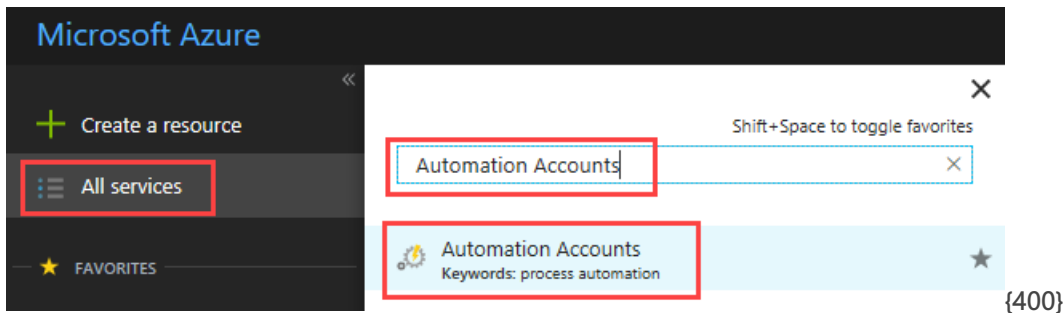
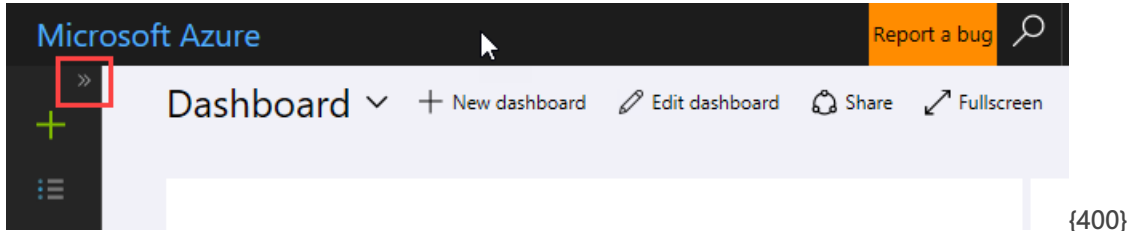
- Upload DSC Configuration
- Deploy an ARM template
- Connect a website to an Azure SQL database
- Access the site to ensure its successful

Estimated Time to Complete This Lab

45 minutes

Task Description

1. Go to <https://portal.azure.com> and log in with your account.
2. In the Azure Portal, expand the menu on the left by clicking >> if isn't already expanded. Click **All Services**. Search for **Automation Accounts** then click **Automation Accounts**.



3. Click **ContosoAutomationAccount** to open your Automation Account.
4. In your Automation Account click the **Module Gallery** under **Shared Resources**.
5. In Modules Gallery, search for **xPSDesiredStateConfiguration**. and click it.
6. Click **Import** and **OK**.
7. Module import can take several minutes. Wait here for all the activities to be extracted successfully.
8. Make sure you have also got the **xWebAdministration** module imported into your Automation Account's Modules. If you completed the DSC Module, you may have already imported it. If you don't, repeat the above steps to import **xWebAdministration**.
9. Next, from your Automation Account, click **State configuration (DSC)**.
10. Click on the **Configurations** tab and click **Add**. Click the folder icon.
11. From your lab files, on **C:\Labs\Module5** folder, locate **DeployDBWebsite.ps1**. Upload this file as the DSC Configuration and click **OK**. This DSC configuration will deploy the Adventure Works website.
12. From the **DSC Configurations** screen, click **DeployDBWebsite**.
13. On your lab machine, open **Windows PowerShell** from the start menu.
14. In your PowerShell console, change directory to **C:\Labs\Module5**
15. Run the following command to log into your lab Azure Subscription:

Connect-AzAccount

16. [] When prompted to do so, login with the account you have your Lab Subscription configured with.
17. [] Before we compile the configuration we need to upload the website files and generate a URL for the DSC configuration. This URL will be provided to the configuration as a parameter. We can create the Resource Group we will use to deploy to. These commands will also create a storage account and upload the website files for use by the DSC extension. Copy and paste the block into the PowerShell window. Press **Enter**.

[!note] The `Clip` command in the script below simply copies the output of the `$key` variable to the current clipboard. If you need to access the value of the SAS token again - simply enter `$key` at the PowerShell prompt and it will display the value again.

```
$resourceGroupName = "ContosoWebApp"
$location = Get-AzLocation | Select Location | Out-GridView -PassThru
New-AzResourceGroup -Name $resourceGroupName -Location $location.Location
$storageAccount = New-AzStorageAccount -Name ($resourceGroupName.ToLower() + (Get-Random -
Maximum 1000)) -ResourceGroupName $resourceGroupName -SkuName Standard_LRS -Location
$location.Location
$context = (Get-AzStorageAccount -ResourceGroupName $resourceGroupName -Name
$storageAccount.StorageAccountName).Context
Set-Location C:\Labs\Module5
New-AzStorageContainer -Name dbfiles -Permission Blob -Context $context
Set-AzStorageBlobContent -File .\DBWebsite.zip -Container dbfiles -Blob DBWebsite.zip -
Context $Context
$key = New-AzStorageBlobSASToken -Container "dbfiles" -Blob "DBWebsite.zip" -Permission r -
Context $context -FullUri -ExpiryTime (Get-Date).AddDays(30)
$key | Clip
```

18. [] From **DeployDBWebsite**, click the **Compile** button. In the blade that opens - use Ctrl-V to paste in the blob URI from the above PowerShell commands. Click **OK**
19. [] The compilation job will be queued, In Progress and then Completed. Wait for a few minutes, for the compilation to be completed successfully.
20. [] At this point, we have imported a configuration to deploy the Adventure Works site on a Virtual Machine. Next, we will deploy an ARM template, which will deploy a VM, Azure SQL Database and apply the DSC configuration to the VM.
21. [] In the lab files for this workshop, navigate to the **Module5** folder on `C:\Labs\Module5`. In there, you will find two .json files:
 - azuredeploy.json
 - azuredeploy.parameters.json
22. [] Open the parameters file `azuredeploy.parameters.json` in **Visual Studio Code** on your lab machine.
23. [] This defines the parameters used for the Website and Database deployment. You should see that the following parameters are already populated with values:

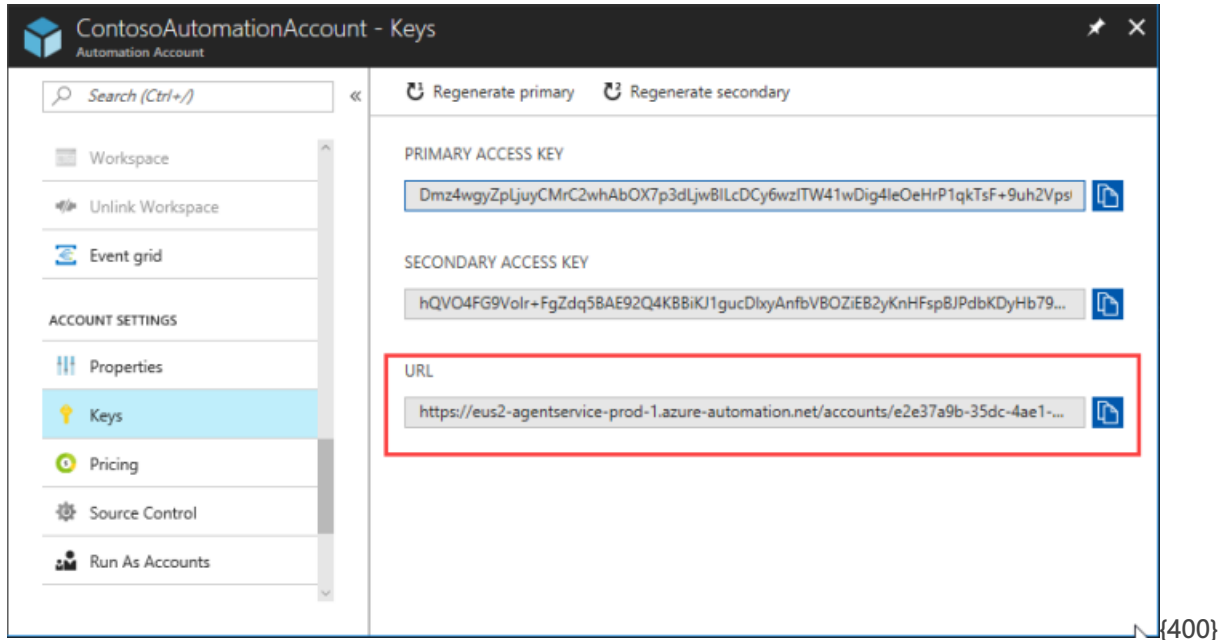
Module 5: Azure Resource Manager (ARM)

- vmName (name of the VM to deploy)
- adminUsername (username for VM)
- adminPassword (password for VM)

24. □ Next we will populate the 2 other parameters, **AutomationRegistrationUrl** and **AutomationRegistrationKey**. In the Portal navigate to your **ContosoAutomationAccount** Automation Account.

25. □ From the menu on the left of your Automation Account, click **Keys** under ACCOUNT SETTINGS

26. □ From the Keys copy the URL value:



27. □ Navigate back to your **azuredeploy.parameters.json** file in Visual Studio Code. Paste in the URL into the value for **AutomationRegistrationUrl**.

28. □ Go back to your Automation Account, from **Keys** copy either the Primary or Secondary **Access Key**.

29. □ Paste the access key into **azuredeploy.parameters.json** for the value of variable **AutomationRegistrationKey**.

30. □ Your parameter file should look similar to the following. Save the file.

```

1  {
2    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json",
3    "contentVersion": "1.0.0.0",
4    "parameters": {
5      "vmName": {
6        "value": "WebAppVM"
7      },
8      "adminUsername": {
9        "value": "student"
10     },
11     "adminPassword": {
12       "value": "Password.111"
13     },
14     "AutomationRegistrationUrl": {
15       "value": "https://eus2-agent-service-prod-1.azure-automation.net/accounts/6fb088b3-f5
16     },
17     "AutomationRegistrationKey": {
18       "value": "oSfxIcB43WUXYdRKbK5IFZ/MNA/JePIGL2d5IgG1DI5ymtQv+Q9fXwiWrDsOHZ9YpNF8t+41h8
19     }
20   }
21 }

```

31. [] In the same directory as the parameters file, you should find **azuredeploy.json**. This is the JSON template file, which the parameters will be passed into. Open **azuredeploy.json** in Visual Studio Code.
32. [] Review the contents of **azuredeploy.json**. You should see several resources are deployed by this:
 - A Virtual Network
 - A Public IP Address
 - A Network Interface
 - A Virtual Machine
 - Within the Virtual Machine, the DSC extension which deploys the DSC Node Configuration and DeployDBWebsite, which you uploaded previously
 - An Azure SQL Server
 - The example **Adventureworks** Azure SQL Database
33. [] Next, we are ready to deploy our ARM templates. Enter in the following command, to deploy the template file and parameter file to the Resource Group we just created:

```
New-AzResourceGroupDeployment -ResourceGroupName 'ContosoWebApp' -TemplateFile
.\azuredeploy.json -TemplateParameterFile .\azuredeploy.parameters.json -Verbose
```

34. [] You should now see that your template is validated, and the deployment is happening. Keep an eye on the output to ensure this is successful. It will take several minutes to complete

```

PS C:\labs\module5> New-AzureRmResourceGroupDeployment -ResourceGroupName 'ContosoWebApp' -TemplateFile
.\azuredeploy.json -TemplateParameterFile .\azuredeploy.parameters.json -Verbose
VERBOSE: Performing the operation "Creating Deployment" on target "ContosoWebApp".
VERBOSE: 1:12:28 PM - Template is valid.
VERBOSE: 1:12:30 PM - Create template deployment 'azuredeploy'

```

35. [] Once completed, you should see a success message similar to below:

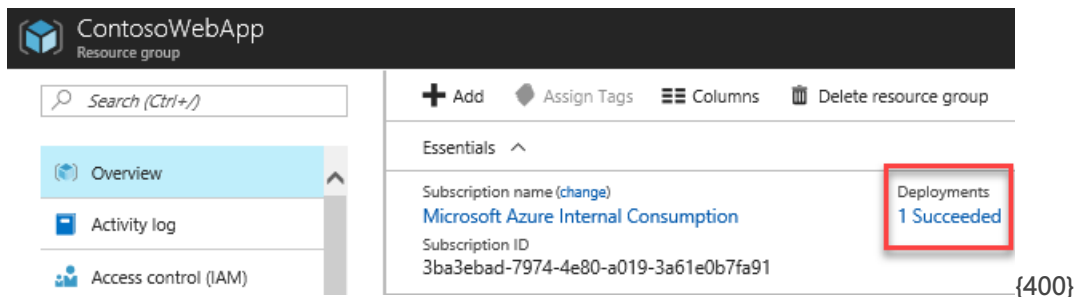
Module 5: Azure Resource Manager (ARM)

```
DeploymentName      : azuredeploy
ResourceGroupName  : ContosoWebApp
ProvisioningState   : Succeeded
Timestamp          : 1/30/2018 11:40:09 PM
Mode               : Incremental
TemplateLink       :
Parameters         :
                    Name      Type      Value
                    =====
                    vmName    String   WebAppVM
                    vmSize    String   Standard_D2_V2
                    adminPassword SecureString
                    adminUsername String   student
                    windowsOSVersion String   2016-Datacenter
                    sqlAdministratorLogin String   student
                    sqlAdministratorLoginPassword SecureString
                    automationRegistrationUrl String   https://
                    rvice-prod-1.azure-automation.net/accounts/4c51eea8-b68b-4315
                    33ff0f
                    automationRegistrationKey SecureString

Outputs
DeploymentDebugLogLevel : {400}
```

36. [] From the Azure Portal, navigate to the new **ContosoWebApp** Resource Group.

37. [] From the ContosoWebApp Resource Group, click the 1 **Succeeded** link.



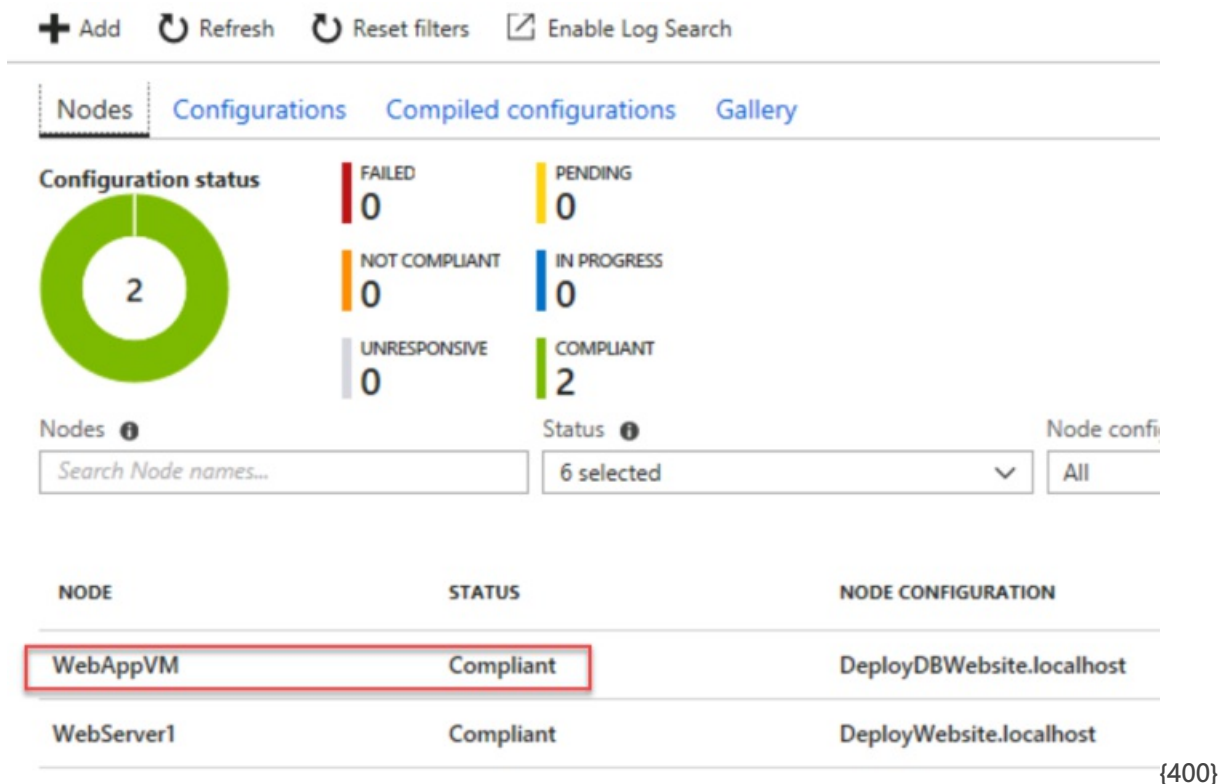
The screenshot shows the Azure Portal interface for the 'ContosoWebApp' Resource group. The left sidebar contains links for 'Overview', 'Activity log', and 'Access control (IAM)'. The main area shows the 'Essentials' section with 'Subscription name (change)' as 'Microsoft Azure Internal Consumption' and 'Subscription ID' as '3ba3ebad-7974-4e80-a019-3a61e0b7fa91'. The 'Deployments' tab is selected, showing '1 Succeeded' deployment. The 'Deployments' link is highlighted with a red box.

38. [] Click the **azuredeploy**. From here, you can see all of the deployment logs, input parameters and results. Browse through the logging, to see what was deployed.

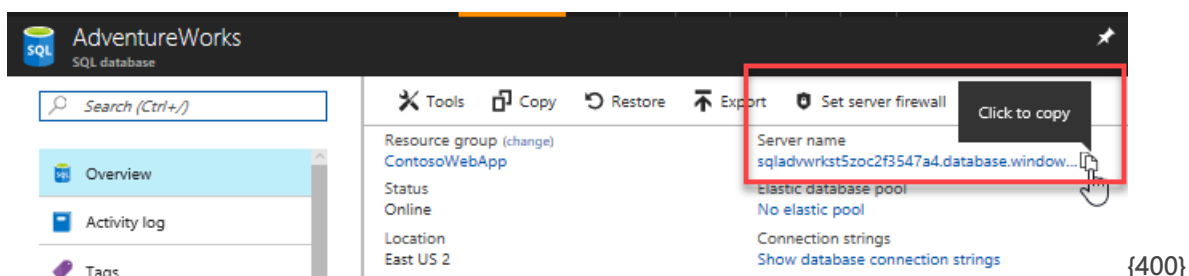
39. [] Next in the Azure Portal, navigate to your **ContosoAutomationAccount** Azure Automation account.

40. [] In Azure Automation, click **State configuration (DSC)**.

41. [] In the **Nodes** tab you should see that your new **WebAppVM** has been added to DSC.



42. □ Wait for a few minutes for the DSC Node to complete if it has not already. This should change to the status **Compliant**. This means the Adventure Works website has been installed onto your Virtual Machine.
43. □ We have got one final step to complete this deployment. Next, we have to connect the Adventure Works website on the VM, to the Azure SQL Database which was deployed. From the Azure Portal, expand left pane, and click **SQL databases**.
44. □ You should see the **AdventureWorks** database. Click this to open up the details of the SQL database.
45. □ From here, we can see the database is online, and the Azure SQL Server on which it is located. We need to copy the SQL Server name. Locate **Server name** and copy the name. Paste this into notepad, as we will need it later.



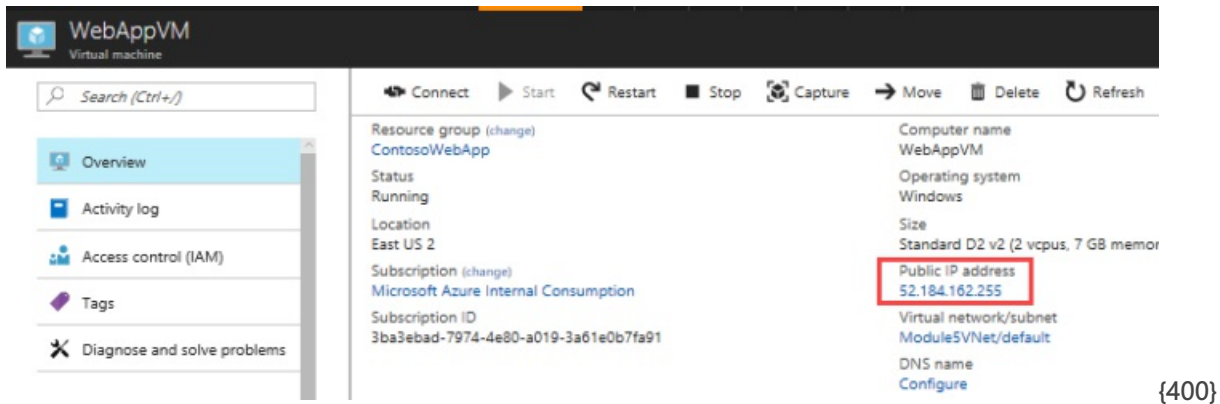
46. □ The website configuration needs to be updated to insert the SQL server name into the connection string. For this we can use `Invoke-AzVMRunCommand` which allows us to execute a script on a virtual machine.

In the PowerShell window run the command below ensuring that you update the SQL server name with the one you have copied from the portal.

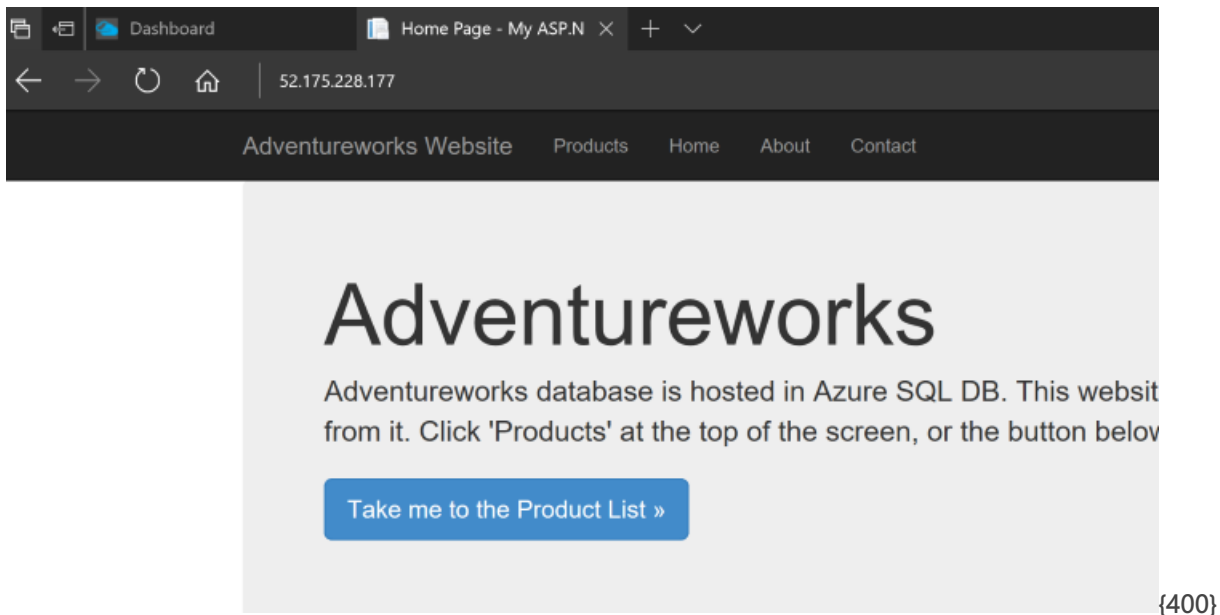
Module 5: Azure Resource Manager (ARM)

```
Invoke-AzVmRunCommand -ResourceGroupName 'ContosoWebApp' -VMName 'WebAppVm' -  
CommandId 'RunPowerShellScript' -ScriptPath .\updateConnectionString.ps1 -Parameter  
@{sqlServerName="YOUR SQL SERVER NAME"}
```

47. [] Now, we should be able to access the website we deployed. Go back to the **Azure Portal** and navigate back to your **WebAppVM**.
48. [] Your virtual machine should have a **Public IP address**. Make a note of your IP address:



49. [] From a Web Browser, open up the IP Address of your VM. You should see the Adventureworks website:



50. [] In the Adventureworks website, click the **Products** link at the top.
51. [] This should take a moment to load, but you should be presented with the Adventureworks product list. This data is being read from the Azure SQL DB and being displayed in the Website, running on your VM:

Module 5: Azure Resource Manager (ARM)

Dashboard

Index - My ASP.NET App

52.175.228.177/Products

Adventureworks WebsiteProductsHomeAboutContact

Index

Create New

Name	ProductNumber	Color	StandardCost	ListPrice	Size	Weight	ProductCategoryID	ProductModelID	SellStartDate
HL Road Frame - Black, 58	FR-R92B-58	Black	1059.31	1431.50	58	1016.04	18	6	6/1/2002 12:00:00 AM
HL Road Frame - Red, 58	FR-R92R-58	Red	1059.31	1431.50	58	1016.04	18	6	6/1/2002 12:00:00 AM
Sport-100 Helmet, Red	HL-U509-R	Red	13.09	34.99			35	33	7/1/2005 12:00:00 AM
Sport-100	HL-U509	Black	13.09	34.99			35	33	7/1/2005

{400}

52. [] You have now successfully deployed a Web Application, via Azure Resource Manager templates, and downloaded and installed the website using Azure Automation DSC!