Microsoft Cloud Workshops

Azure Resource Manager Lab Guide

February 2017

© 2017 Microsoft Corporation. All rights reserved. This document is confidential and proprietary to Microsoft. Internal use only. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

## Contents

[Azure resource manager hackathon 1](#_Toc474933657)

[Azure resource manager hackathon answers 10](#_Toc474933658)

# Azure resource manager hackathon

## Overview

Contoso has asked you to define an Azure Resource Manager template that can deploy their application CloudShop and its associated database using Azure Virtual Machines.

## Requirements

* Microsoft Azure subscription
* Local machine or a virtual machine configured with:
  + Visual Studio 2015 Community Edition
  + Azure SDK 2.9+ for Visual Studio
    - <https://www.microsoft.com/en-us/download/details.aspx?id=51657>
  + Azure PowerShell (installed as part of Azure SDK)
  + Reboot after installing the Azure SDK

## Lab structure

This lab has two sets of instructions. The first is a high-level set of instructions that is designed for students that have previous experience authoring templates in Azure. The second is a traditional hands-on lab guide that is designed for users that are new to Azure Resource Manger template authoring.

## Help references

|  |  |
| --- | --- |
| Authoring ARM Templates | <https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/> |
| Azure Resource Manager templates with VS 2015 | <http://blogs.msdn.com/b/kaevans/archive/2015/07/06/azure-resource-manager-templates-with-visual-studio-2015.aspx> |
| Virtual Machine Scale Set Samples | <https://github.com/gbowerman/azure-myriad> |
| Azure Quick Start Templates | <https://github.com/Azure/azure-quickstart-templates> |
| Network Security Groups | <https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-nsg/> |

## Exercise 0: Environment setup

### Overview

In this lab, you will create a developer environment and download the required files for this course if you do not already have one that meets the requirements.

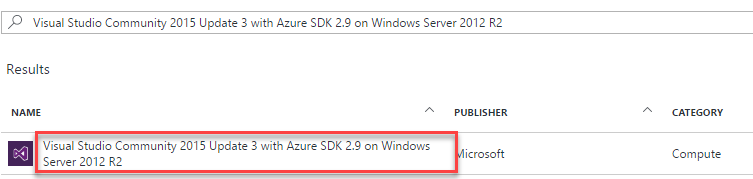
### Prerequisites

* Microsoft Azure subscription: <http://azure.microsoft.com/en-us/pricing/free-trial/>
* Client computer with Windows 10 or later with Visual Studio 2015 - SDK 2.9.6+
  + Ensure you reboot after installing the SDK or Azure PowerShell will not work correctly

#### Task 1: Configure a Development Environment

If you do not have a machine setup with Visual Studio 2015 Community and Azure SDK 2.9.6+ complete this task.

1. Create a virtual machine in Azure using the Visual Studio Community 2015 Update 3 and SDK 2.9 on Windows Server 2012 R2 image.



It is highly recommended to use a DS2\_V2 or D2\_V2 instance size for this VM

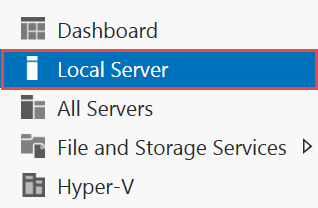
#### Task 2: Disable IE Enhanced Security

Note: Sometimes this image has IE ESC disabled. Sometimes it does not.

1. On the new VM you just created, click the Server Manager icon.



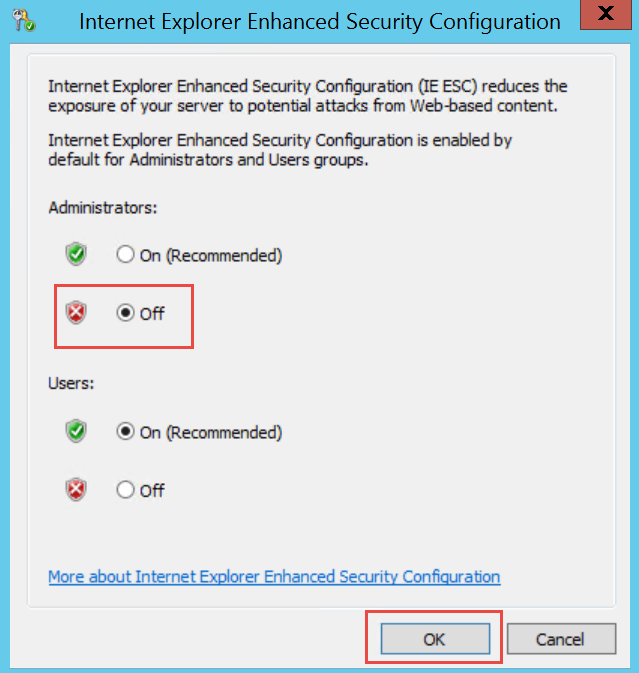
Click Local Server



1. On the right side of the pane, click **On** by IE Enhanced Security Configuration.

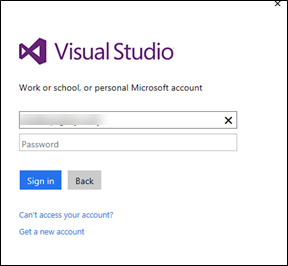


1. Change to **Off** for Administrators and click **OK**.

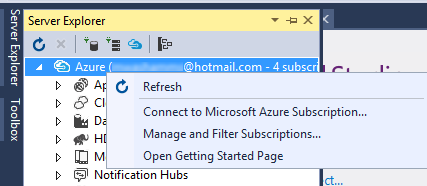


#### Task 3: Validate Connectivity to Azure

1. From within the virtual machine, Launch Visual Studio 2015 and validate that you can login with your Microsoft Account when prompted.
   1. Note: You will have to search for “Visual Studio”, after clicking the Start Button.



1. Validate connectivity to your Azure subscription. Launch Visual Studio, open Server Explorer from the View menu, and ensure that you can connect to your Azure subscription.



**Download the exercise files**

1. Download the exercise files for the training (from within the virtual machine).
   1. Create a new folder on your computer named **C:\Hackathon**
   2. Download the support files (.zip format), <https://opsgilityweb.blob.core.windows.net/cebootcamp-feb-2017/Lift%20and%20Shift%20ARM%20Hackathon%20-%20Student%20Files.zip> to the new folder.
   3. Extract the contents to the same folder.

Tip: In the labs when adding resources to the Azure Resource Manager template, ensure there are no spaces at the end of the resource names. Visual Studio will include the space in all of the code it generates.

## Summary

In this lab, you created a developer environment and downloaded the required files for this course.

### Exercise 1: Configure Automation Account

Contoso has asked you to create an Azure Resource Manager template that will configure the network for a proof of concept deployment of their new application CloudShop. Before creating the ARM Template, you must first create and configure an Azure Automation account using the Azure Portal.

#### Task 1: Create Azure Automation Account

*Tasks to complete*

* Create Automation Account in Azure portal
* Upload the Web and SQL .ps1 files
* Compile the files so they are available to virtual machines that check-in

*Exit criteria*

* You log into the portal and verify creation of the Azure Automation account, taking note of the Registration URL, Registration Keys, and the names of the nodes you compiled.

### Exercise 2: Define the network foundation

Contoso has asked you to create an Azure Resource Manager template that will configure the network for a proof of concept deployment of their new application CloudShop.

#### Task 1: Define the network foundation

##### Tasks to complete

* Create a new Azure Resource Manager template that creates a virtual network.
* Validate the JSON file with <http://jsonlint.com>.
* Deploy the virtual network to an Azure subscription.

##### Exit criteria

* You navigated to <http://jsonlint.com>, pasted the contents of your azuredeploy.json file into the text box, and clicked **Validate**.

Note: In future lab steps if you have errors deploying a template, repeat this process and check for validation errors.

* A template deployed to your Azure subscription that has the following virtual network characteristics.
  + Address space: 10.0.0.0/16
  + Subnet: FrontEndNet – 10.0.0.0/24
  + Subnet: DatabaseNet – 10.0.1.0/24

### Exercise 3: Extend with compute

The next task is to update the template by adding a storage account, a public IP, and virtual machines for the web server(s) and a SQL server. The servers will be automatically configured using a PowerShell DSC script supplied to you by the infrastructure team.

#### Task 1: Add storage and a location parameter

##### Tasks to complete

* Add a new storage account to the template. All virtual machines in the template will use this storage account.
* Create a location parameter for the template that all resources in the template use. Update the existing virtual network to use this parameter and as you add new resources to the template that require a location, update them to reference this parameter.
* Deploy the updated template.

##### Exit criteria

* A new storage account in the template.
* A location parameter that defines the list of regions the template can deploy to.
* The new storage account should be deployed to your subscription after deployment.

#### Task 2: Add a virtual machine for the web server

##### Tasks to complete

* Add the following resources to the template
  + A Windows virtual machine for the web front end
  + Public IP with a unique DNS name linked to the NIC of the web VM.
  + Azure DSC Extension
    - DSC Extension named **hackathonDSC** for the web servers that use the CloudShopWeb.WebServer configuration node of your Automation account.

##### Exit criteria

* The template should accept parameters for the Registration URL, Registration Key, and Configuration Node name(s) of the Automation account.
  + The scripts for the DSC Configurations are located in C:\Hackathon
* The web site should load when you browse the DNS name for the public IP.

#### Task 3: Add a virtual machine for the SQL Server

##### Tasks to complete

* Add the following resources to the template
  + A Windows virtual machine
  + Azure DSC Extension
    - DSC Extension named **hackathonDSCSql** for the SQL database that configures the SQL Server and restores the AdventureWorks database.
    - The scripts for the DSC Configurations are located in C:\Hackathon
    - Add a dependency on the DSC extension for the web tier to the DSC extension for the SQL server. This allows the virtual machines to be provisioned in parallel but the customizations to be orchestrated.

##### Exit criteria

* SQL Server virtual machine
  + Should have a parameter that specifies the size of the virtual machine. The default value should be Standard\_D2.
    - Standard\_D1, Standard\_D2, Standard\_D3, Standard\_D4, Standard\_D5
  + Should accept a parameter named **hackathonSqlVMSKU** that allows the user to specify the SQL Server SKU using one of the following values. The default value should be Web.
    - Web, Standard, Enterprise
  + The image configuration should use the following:
    - Image Publisher: MicrosoftSQLServer
    - Image Offer: SQL2014-WS2012R2
  + The virtual machine should have 4 x 1 TB disks attached to it formatted a 4 TB volume using Windows Storage Spaces.
  + The SQL Server should have the default path for data, logs, and backup referring to the F: drive and the database should be restored to the F: drive.
  + SQL Server should be in mixed authentication mode.

### Exercise 4: Lock down the environment

In this portion of the exercise, you will deploy a network security group to restrict the network attack surface for the deployment.

#### Task 1: Restrict traffic to the web server

##### Tasks to complete

* Create and deploy a network security group that allows the following traffic on the FrontEndNet subnet.
  + Source: INTERNET, Destination port: 80

##### Exit criteria

* Validate restrictions by connecting to the web server using a web browser. Traffic should be allowed in and the site should load. Attempt to connect using remote desktop, click **Connect** in the portal. This connection should fail.

#### Task 2: Update the network security group to allow Windows Remote Desktop

##### Tasks to complete

* Update the network security group on the deployed resources to allow the remote desktop protocol (port 3389).

##### Exit Criteria

* Access to the web front end should be successful by clicking **Connect** in the Azure Preview Portal.

### Exercise 5: Scale out the deployment

In this portion of the exercise, you will use the Azure Virtual Machine Scale Sets feature to scale out the web front end and the storage sub system.

Note: This is an *advanced* configuration based on a new feature (scale sets). It is recommended that unless you have configured scale sets before that you jump to the answers section of the lab to complete this exercise.

#### Task 1: Parameterize the size of the environment and add load balancing

##### Tasks to complete

* Add a new parameter called instanceCount to the template. This variable should define how many web servers should be created.
* Add a new resource that can create up to five storage accounts.
* Add a load balancer resource and associate it with the public IP.
* Add the virtual machine scale set configuration that will use the storage accounts and creates a number of virtual machines based on the instanceCount parameter.
* Remove the existing virtual machine and network adapter because they are replaced by the scale set.

##### Exit criteria

* The scale set should provision the number of virtual machines based on the instanceCount parameter.
* The virtual machines provisioned by the scale set should execute the same DSC script used in the initial web server configuration.

# Azure resource manager hackathon answers

## Overview

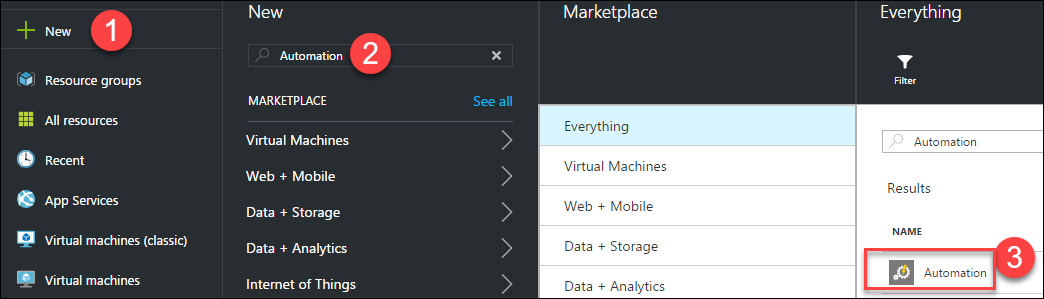
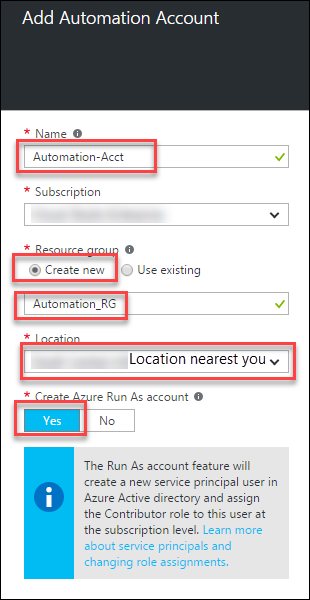
This portion of the lab is designed to help you if you are blocked or have limited experience with Azure Resource Manager.

## Exercise 1: Configure Automation Account

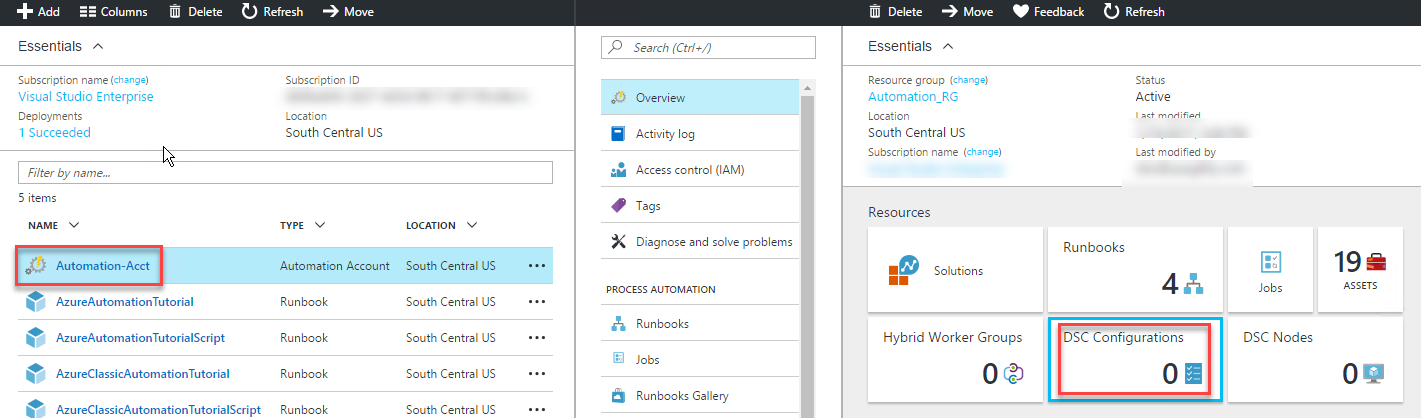
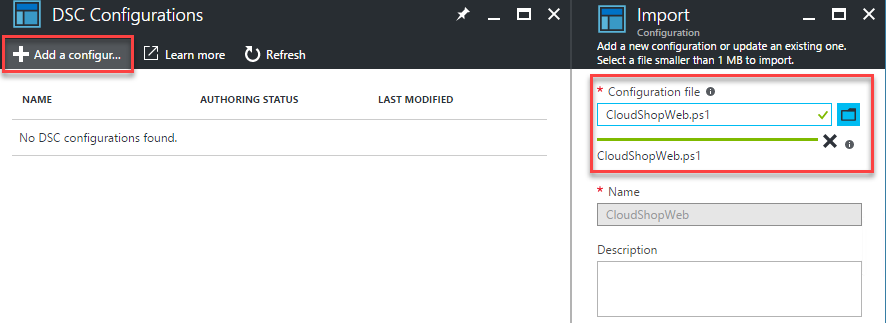
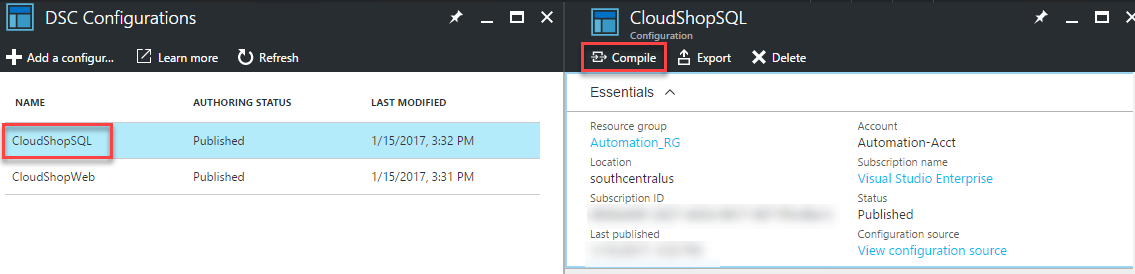
### Overview

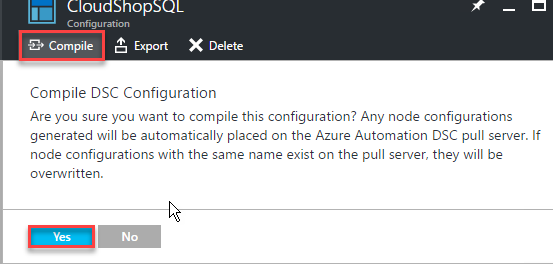
In this exercise, you will create and configure an Azure Automation account in the Azure Portal before configuring and deploying the resources of your ARM template.

### Task1: Create Automation Account

1. Browse to the Azure Portal and authenticate at <https://portal.azure.com/>
2. Click **+New** and type **Automation** in the search box. Choose **Automation** from the results.  
   
3. Click **Create** on the Automation blade so that the **Add Automation Account** blade is displayed. Specify the following information and click **Create**.  
     
   

### Task 2: Upload DSC Configurations into Automation account

1. Click **Resource groups > Automation\_RG > Automation-Acct** and click the **DSC Configuration** tile  
   
2. Click Add a configuration to upload **C:\Hackathon\CloudShopSQL.ps1** and **C:\Hackathon\CloudShopWeb.ps1**
3. After importing the .ps1 files, click the **CloudShopSQL** DSC Configuration and click **Compile** on the toolbar (*click* ***Yes*** *on the overwrite prompt*). Do the same for **CloudShopWeb**



### Summary

In this exercise, you configured an Automation account, and configured DSC configuration scripts that will be leveraged by the virtual machine resources.

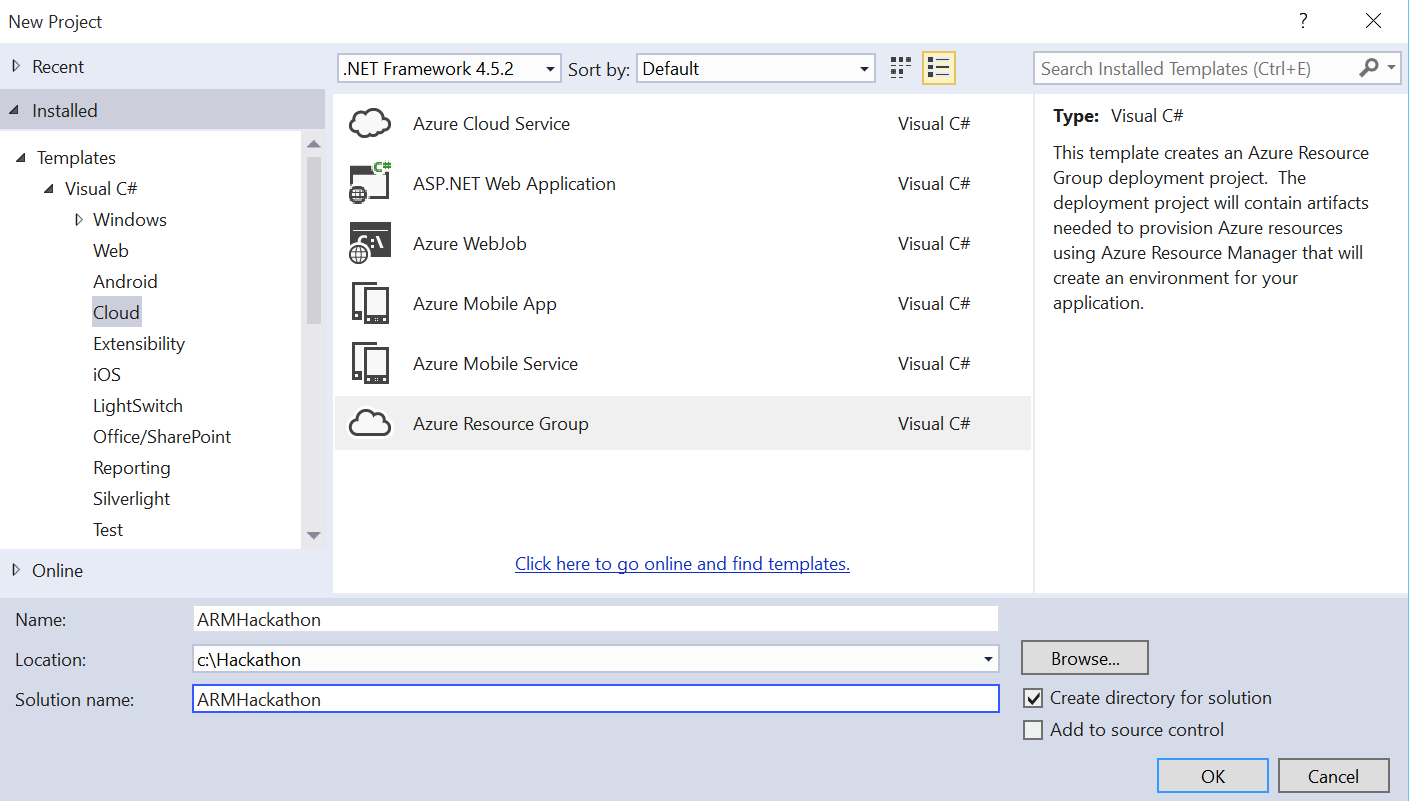
## Exercise 2: Define the network foundation

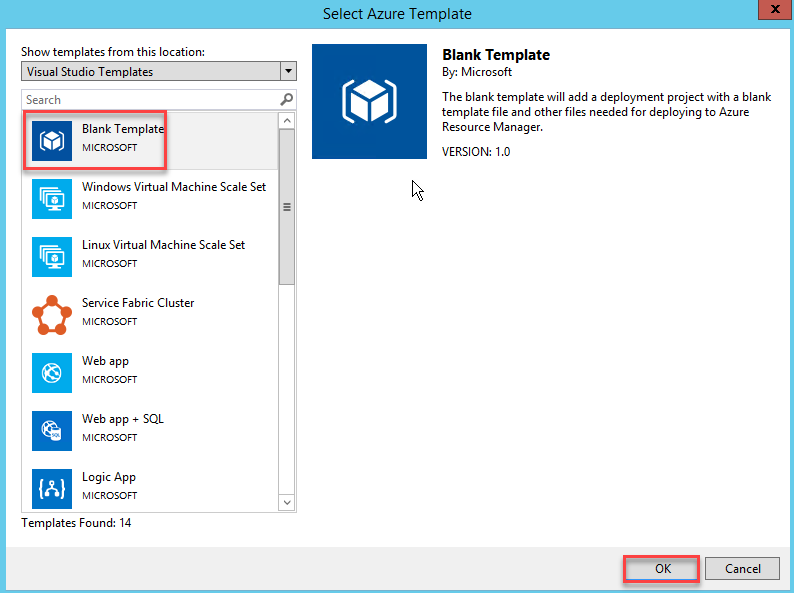
### Overview

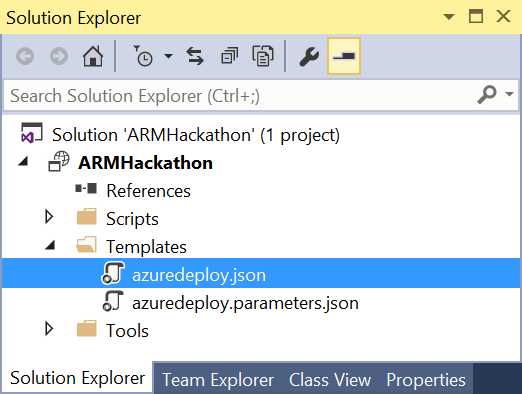
Your first ARM template task is to create a virtual network template using Visual Studio and deploy it to your Azure account.

### Task 1: Deploy a virtual network with a template

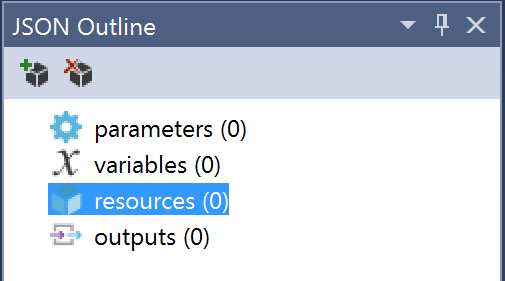
1. Open Visual Studio. The shortcut should be available on the desktop.
2. Choose **File**, **New Project**, and then choose **Cloud**, and then **Azure Resource Group**.



1. Name the project **ARMHackathon**,specify **C:\Hackathon** for the location, and click **OK**.
2. On the Select Azure Template dialog box, choose **Blank Template** and click **OK**. 
3. In the **Solution Explorer**, open the **azuredeploy.json** file from the **Templates** folder.



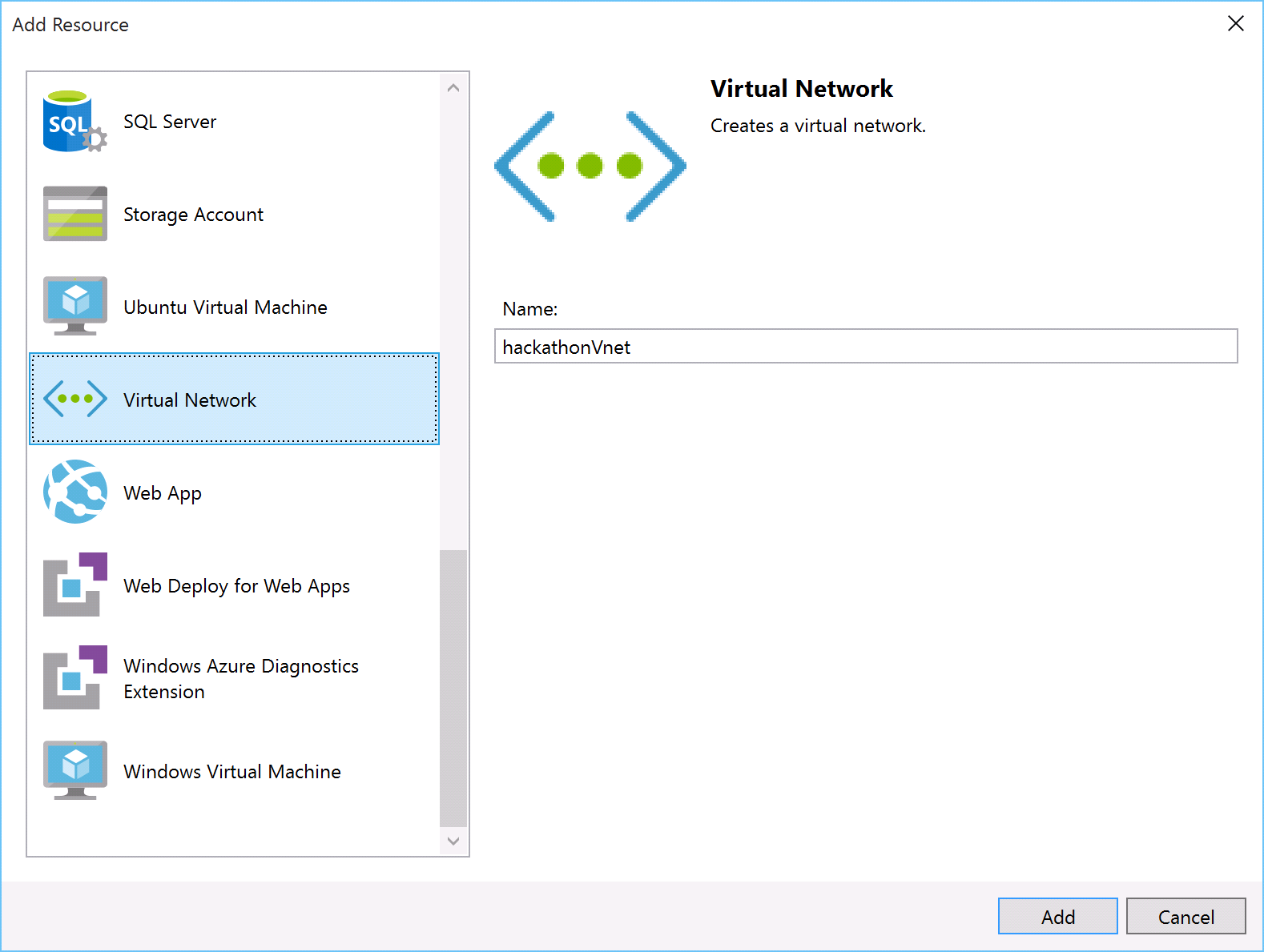
1. The file should contain four different sections: parameters, variables, resources, and outputs. On the left side, a new window called **JSON Outline** should have been opened as well. If this was not the case, go to the **View** menu, select **Other Windows**, and choose **JSON Outline**. The window should look like the following image.



1. On the **JSON Outline** window, click **Add Resource** in the upper-left corner or right-click the **resources** and choose **Add New Resource**.



1. On the **Add Resource** dialog box, choose **Virtual Network**, enter **hackathonVnet** in the **Name** field, and click **Add**.



1. Go to the **azuredeploy.json** file, and inspect its content. Review the **variables** section. It should look like the following file.

"variables": {

    "hackathonVnetPrefix": "10.0.0.0/16",

    "hackathonVnetSubnet1Name": "Subnet-1",

    "hackathonVnetSubnet1Prefix": "10.0.0.0/24",

    "hackathonVnetSubnet2Name": "Subnet-2",

    "hackathonVnetSubnet2Prefix": "10.0.1.0/24"

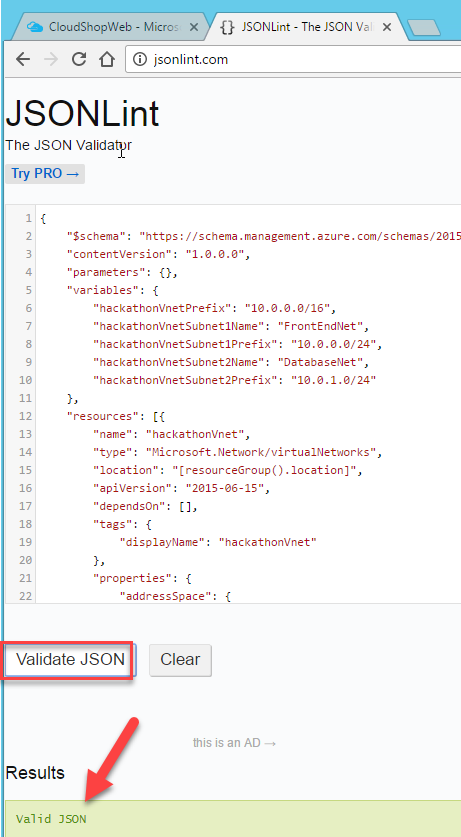
}

1. Change the name of **Subnet-1** to **FrontEndNet**, and the name of **Subnet-2** to **DatabaseNet**.

"hackathonVnetSubnet1Name": "FrontEndNet",

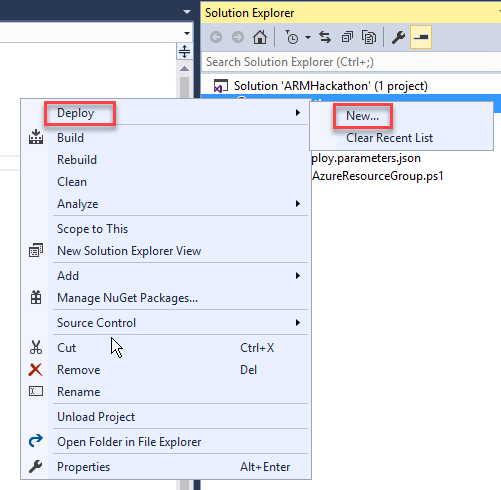
"hackathonVnetSubnet2Name": "DatabaseNet",

1. Launch a browser and navigate to <http://jsonlint.com>. Copy the entire contents of **azuredeploy.json**, paste it in the text box, and click **Validate**.

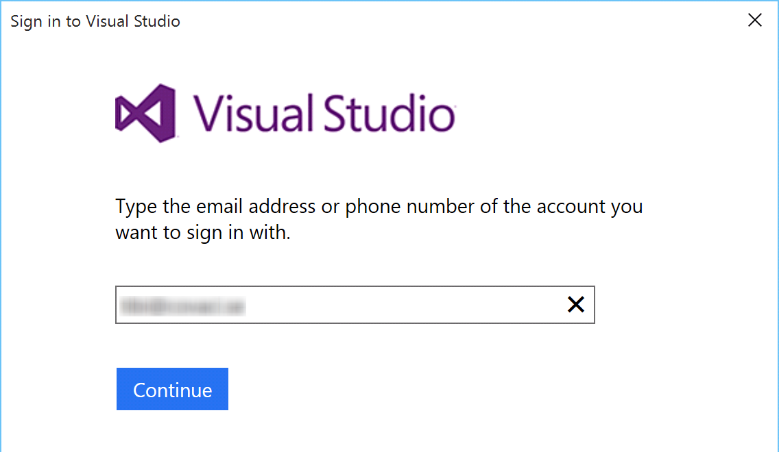


Note: In future lab steps if you have errors deploying a template, repeat this process and check for validation errors.

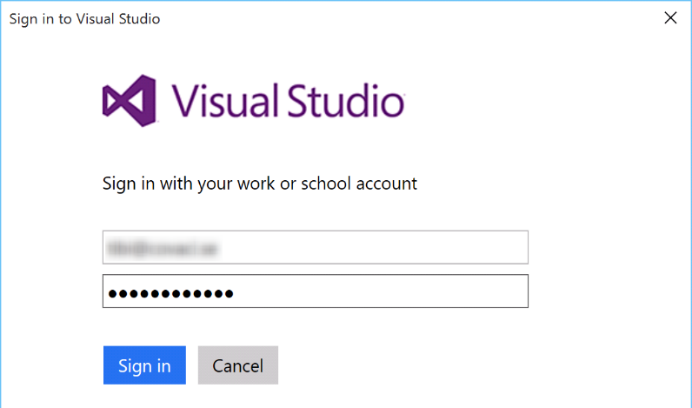
1. Deploy the template by right-clicking the **ARMHackathon** project, and choosing **Deploy** **>** **New Deployment**.



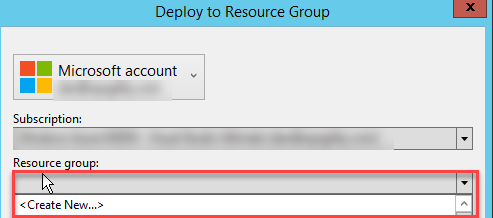
1. If you did not sign in to your Microsoft Azure account already, you will be asked to do so now.
2. Fill in the email address associated with the Azure account and click **Continue**.



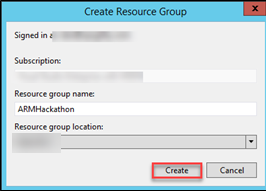
1. You might have to choose between a Work/School account and a Microsoft account. Microsoft account refers to a LiveId account. Depending on what kind of account you have, you should choose one or the other.
2. Enter your password and click **Sign In**.



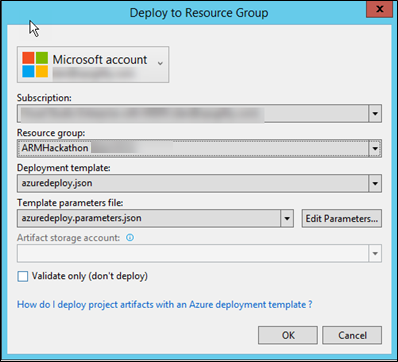
1. If you have several subscriptions, choose the one that you want your VNet to be deployed to, and on the Resource group choose **Create New**.



1. On the Create Resource Group dialog box, accept the default value for the name; and for the location, choose the **closest location to you**, and click **Create**.

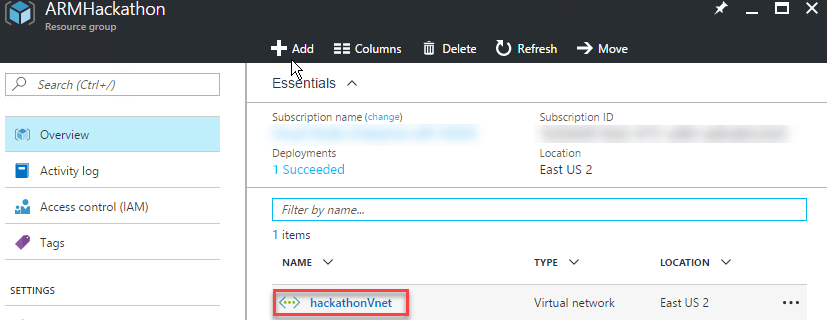


1. When you are back on the Deploy to Resource Group dialog box, click **OK**. After about a minute, your virtual network will be deployed to Azure.



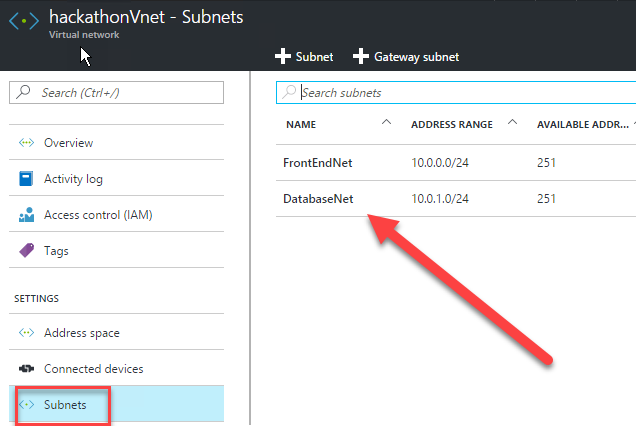
1. View the created resource group and virtual network in the Azure Management Portal by clicking **Resource Groups** and clicking the **ARMHackathon**.





### Summary

In this exercise, you have created a new virtual network with two different subnets. To verify this, sign in to the **Microsoft Azure** portal and go to **Virtual Networks**. Your new network should be listed there.



## Exercise 3: Extend with Compute

### Overview

In this task you will continue the work you started in the previous task by creating a storage account and adding virtual machines for the web application and for the database, and then configuring the machines for the roles.

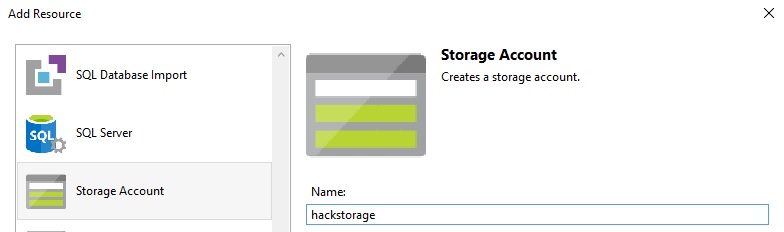
### Task 1: Add an Azure storage account

1. On the **JSON Outline** window, click **Add Resource** in the upper-left corner, or right-click the **resources** and choose **Add New Resource**.



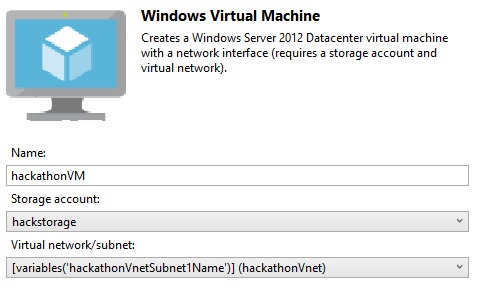
1. Add a new **Storage Account** resource to the template named ***hackstorage***.

Note: The template generated in the Azure SDK appends a unique value (13 characters in length) to the storage account name. Ensure the name specified is 11 characters or less in length.



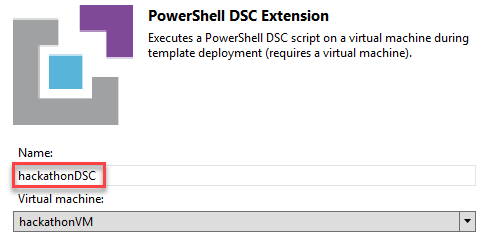
### Task 2: Add a virtual machine and configure as a web server

1. Add a new **Windows Virtual Machine** called **hackathonVM** and choose ***hackstorage*** as the Storage Account and **FrontEndNet** subnet as the Virtual network/subnet. The **FrontEndNet** is the value of **hackathonVnetSubnet1Name** variable.

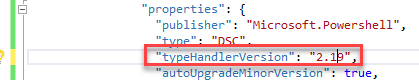


1. A **Network Interface** named ***hackathonVMNic*** was automatically added to the configuration when the virtual machine resource was added to connect the virtual machine to the virtual network. Add a public IP address called ***ok*** to the **hackathonVMNic**. This will allow you to connect to the machine using remote desktop client, or to access the web server.

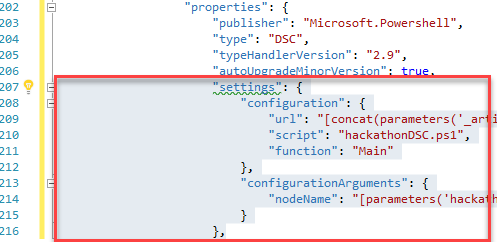
Next you will add the PowerShell DSC Extension to the **azuredeploy.json** file. This will register the VM with Azure Automation DSC Extension



1. Change the Type Handler from 2.9 to 2.19 and make sure that autoUpgradeMinorVersion is true.



1. Find the settings code within the PowerShell DSC section you just added and replace it with this code:



"settings": {

"modulesUrl": "https://opsgilityweb.blob.core.windows.net/20160629-arm-hackathon/RegistrationMetaConfigV2.zip",

"configurationFunction": "RegistrationMetaConfigV2.ps1\\RegistrationMetaConfigV2",

"Properties": [

{

"Name": "RegistrationKey",

"Value": {

"UserName": "PLACEHOLDER\_DONOTUSE",

"Password": "PrivateSettingsRef:registrationKeyPrivate"

},

"TypeName": "System.Management.Automation.PSCredential"

},

{

"Name": "RegistrationUrl",

"Value": "[parameters('registrationUrl')]",

"TypeName": "System.String"

},

{

"Name": "NodeConfigurationName",

"Value": "[parameters('nodeConfigurationName')]",

"TypeName": "System.String"

},

{

"Name": "ConfigurationMode",

"Value": "[parameters('configurationMode')]",

"TypeName": "System.String"

},

{

"Name": "ConfigurationModeFrequencyMins",

"Value": "[parameters('configurationModeFrequencyMins')]",

"TypeName": "System.Int32"

},

{

"Name": "RefreshFrequencyMins",

"Value": "[parameters('refreshFrequencyMins')]",

"TypeName": "System.Int32"

},

{

"Name": "RebootNodeIfNeeded",

"Value": "[parameters('rebootNodeIfNeeded')]",

"TypeName": "System.Boolean"

},

{

"Name": "ActionAfterReboot",

"Value": "[parameters('actionAfterReboot')]",

"TypeName": "System.String"

},

{

"Name": "AllowModuleOverwrite",

"Value": "[parameters('allowModuleOverwrite')]",

"TypeName": "System.Boolean"

},

{

"Name": "Timestamp",

"Value": "[parameters('timestamp')]",

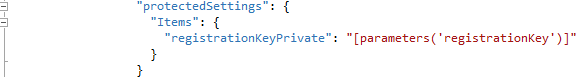
"TypeName": "System.String"

}

]

},

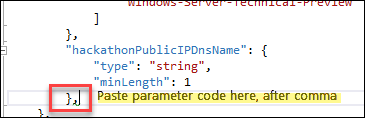
1. Next in the protectedSettings section delete the “configurationUrlSasToken” line; replacing it with this code.



"Items": {

"registrationKeyPrivate": "[parameters('registrationKey')]"

}

1. You will now append the following parameters to your json template (*after the* ***hackathonPublicIPDnsName*** *parameter*)  
     
     
     
   "registrationKey": {

"type": "string",

"metadata": {

"description": "Registration key of Automation account"

}

},

"registrationUrl": {

"type": "string",

"metadata": {

"description": "Registration URL of Automation account"

}

},

"nodeConfigurationName": {

"type": "string",

"metadata": {

"description": "Name of configuration to apply"

}

},

"rebootNodeIfNeeded": {

"type": "bool",

"metadata": {

"description": "Reboot if needed"

}

},

"allowModuleOverwrite": {

"type": "bool",

"metadata": {

"description": "Allow Module Overwrite"

}

},

"configurationMode": {

"type": "string",

"defaultValue": "ApplyAndMonitor",

"allowedValues": [

"ApplyAndMonitor",

"ApplyOnly",

"ApplyandAutoCorrect"

],

"metadata": {

"description": "Configuration Mode"

}

},

"configurationModeFrequencyMins": {

"type": "int",

"metadata": {

"description": "Allow Module Overwrite"

}

},

"refreshFrequencyMins": {

"type": "int",

"metadata": {

"description": "Refresh frequency in minutes"

}

},

"actionAfterReboot": {

"type": "string",

"defaultValue": "ContinueConfiguration",

"allowedValues": [

"ContinueConfiguration",

"StopConfiguration"

],

"metadata": {

"description": "Action after reboot"

}

},

"timestamp": {

"type": "string",

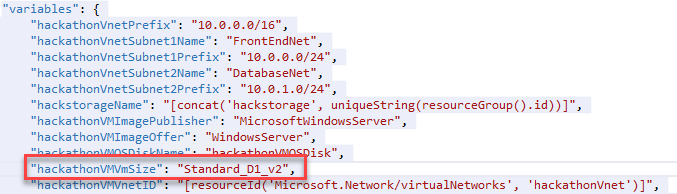
"metadata": {

"description": "Time stamp MM/dd/YYYY H:mm:ss"

}

}

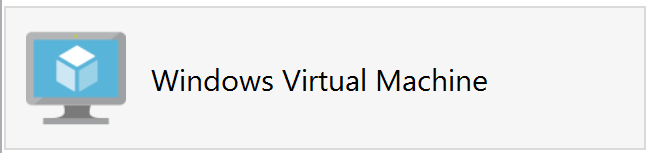
1. In the “**variables**” section, change the value of hackathonVMVmSize" to "Standard\_D1\_v2".



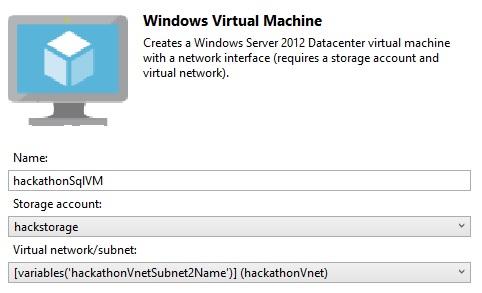
1. Save your changes to the **azuredeploy.json** template file.

### Task 3: Add a Windows virtual machine for the database server

1. Add another virtual machine to the template by clicking **Add Resource** and then selecting **Windows Virtual Machine**.



1. Name this virtual machine resource **hackathonSqlVM** and reference the parameters **hackStorage** and **hackathonVnetSubnet2Name** respectively.



1. Navigate to the **variables** section and find the following variables:



1. Modify the following **hackathonSqlVMImagePublisher** and **hackathonSqlVMImageOffer** variables to the following SQL Server image values:

"hackathonSqlVMImagePublisher": "MicrosoftSQLServer",

"hackathonSqlVMImageOffer": "f",

1. Find the **hackathonSqlVMWindowsOSVersion** parameter.



1. Replace the **hackathonSqlVMWindowsOSVersion** parameter with the following:

"hackathonSqlVMSKU": {

"type": "string",

"defaultValue": "Web",

"allowedValues": [

"Web",

"Standard",

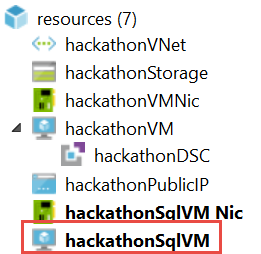
"Enterprise"

]

}

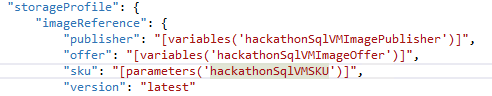


1. Click the **hackathonSqlVM** resource to move to its properties.



1. Update the **SKU** property to point to the new parameter: **hackathonSqlVMSKU**.

"sku": "[parameters('hackathonSqlVMSKU')]",



1. Navigate to the **parameters** section of the template and add a new parameter called **vmSizeSQL** to define the size of the virtual machine.

**Tip:** Do not forget the preceding comma.

"vmSizeSql": {

"type": "string",

"defaultValue": "Standard\_D1\_v2",

"allowedValues": [

"Standard\_D1\_v2",

"Standard\_D2\_v2",

"Standard\_D3\_v2",

"Standard\_D4\_v2",

"Standard\_D5\_v2"

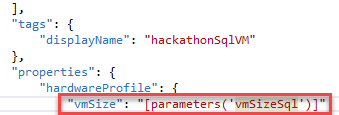
]

}



1. Navigate to the **resources** section for the **hackathonSqlVM**. Find the **hardwareProfile** section and replace the **vmSize** property with the following:

"vmSize": "[parameters('vmSizeSql')]"



1. Next, add two data disks to the **hackathonSqlVM** by first adding the storage paths as variables in the variables section of the template.

"dataDisk1VhdName": "[concat('http://',variables('hackStorageName'),'.blob.core.windows.net/','vhds','/','dataDisk1.vhd')]",

"dataDisk2VhdName": "[concat('http://',variables('hackStorageName'),'.blob.core.windows.net/','vhds','/','dataDisk2.vhd')]"

Do not forget to add a comma after the last variable first and then hitting enter. This will start a new line where you can paste.



1. In order to deploy two 1 TB disks, add the following section to the properties, storage profile section of the **hackathonSqlVM** (right after osDisk).

Tip: Do not forget to add a comma at the end of the osDisk section.



"dataDisks": [

{

"name": "datadisk1",

"diskSizeGB": "1023",

"lun": 0,

"vhd": { "uri": "[variables('dataDisk1VhdName')]" },

"createOption": "Empty"

},

{

"name": "datadisk2",

"diskSizeGB": "1023",

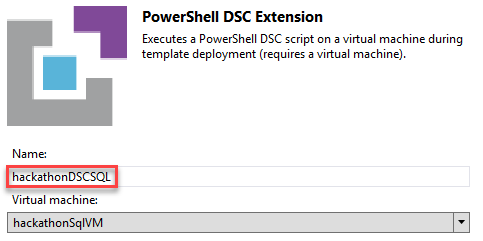
"lun": 1,

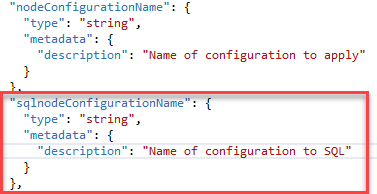
"vhd": {"uri": "[variables('dataDisk2VhdName')]"},

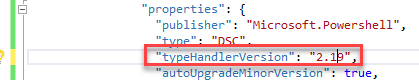
"createOption": "Empty"

}]

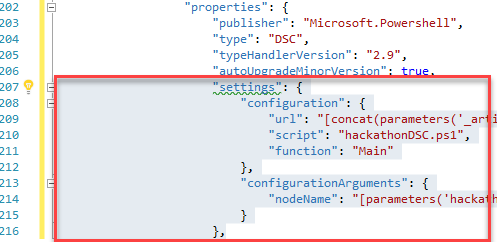
1. Next you will add the PowerShell DSC Extension to the **azuredeploy.json** file for the SQL VM. This will register the VM with Azure Automation DSC Extension.



1. We do need to create a new parameter that will be different for the SQL VM. In the parameters section duplicate the **nodeConfigurationName** parameter and change its name to **sqlnodeConfigurationName**
2. Save your changes to the **azuredeploy.json** template file.
3. Change the Type Handler from 2.9 to 2.19 and make sure that autoUpgradeMinorVersion is true.



1. Find the settings code within the PowerShell DSC section you just added and replace it with this code:



"settings": {

"modulesUrl": "https://opsgilityweb.blob.core.windows.net/20160629-arm-hackathon/RegistrationMetaConfigV2.zip",

"configurationFunction": "RegistrationMetaConfigV2.ps1\\RegistrationMetaConfigV2",

"Properties": [

{

"Name": "RegistrationKey",

"Value": {

"UserName": "PLACEHOLDER\_DONOTUSE",

"Password": "PrivateSettingsRef:registrationKeyPrivate"

},

"TypeName": "System.Management.Automation.PSCredential"

},

{

"Name": "RegistrationUrl",

"Value": "[parameters('registrationUrl')]",

"TypeName": "System.String"

},

{

"Name": "NodeConfigurationName",

"Value": "[parameters('sqlnodeConfigurationName')]",

"TypeName": "System.String"

},

{

"Name": "ConfigurationMode",

"Value": "[parameters('configurationMode')]",

"TypeName": "System.String"

},

{

"Name": "ConfigurationModeFrequencyMins",

"Value": "[parameters('configurationModeFrequencyMins')]",

"TypeName": "System.Int32"

},

{

"Name": "RefreshFrequencyMins",

"Value": "[parameters('refreshFrequencyMins')]",

"TypeName": "System.Int32"

},

{

"Name": "RebootNodeIfNeeded",

"Value": "[parameters('rebootNodeIfNeeded')]",

"TypeName": "System.Boolean"

},

{

"Name": "ActionAfterReboot",

"Value": "[parameters('actionAfterReboot')]",

"TypeName": "System.String"

},

{

"Name": "AllowModuleOverwrite",

"Value": "[parameters('allowModuleOverwrite')]",

"TypeName": "System.Boolean"

},

{

"Name": "Timestamp",

"Value": "[parameters('timestamp')]",

"TypeName": "System.String"

}

]

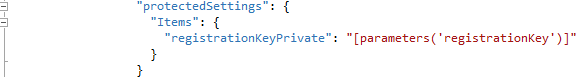
},

1. Next in the protectedSettings section delete the “configurationUrlSasToken” line; replacing it with this code.

"Items": {

"registrationKeyPrivate": "[parameters('registrationKey')]"

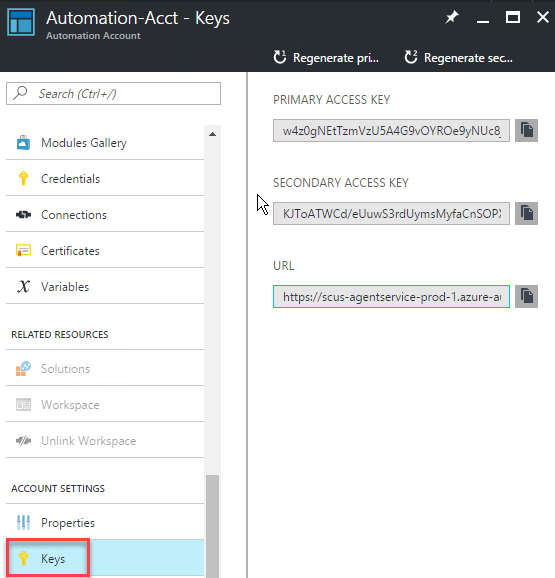
}



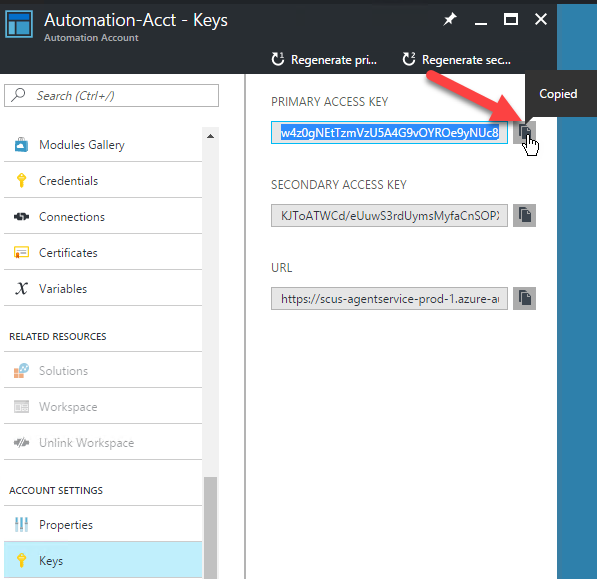
1. Save your changes to the **azuredeploy.json** template file.

### Task 4: Deploy your updated template to Azure

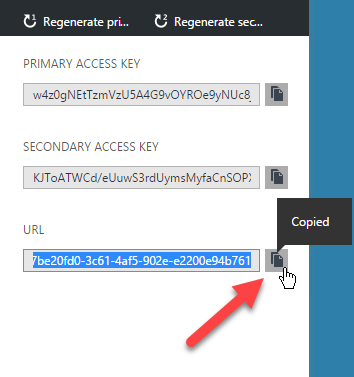
1. Before deploying your updated template, you should make note of your Automation key and registration URL. In the Azure Portal, click **Resource groups > Automation\_RG > Automation-Acct**, and then in the Account Settings area click the **Keys** icon.



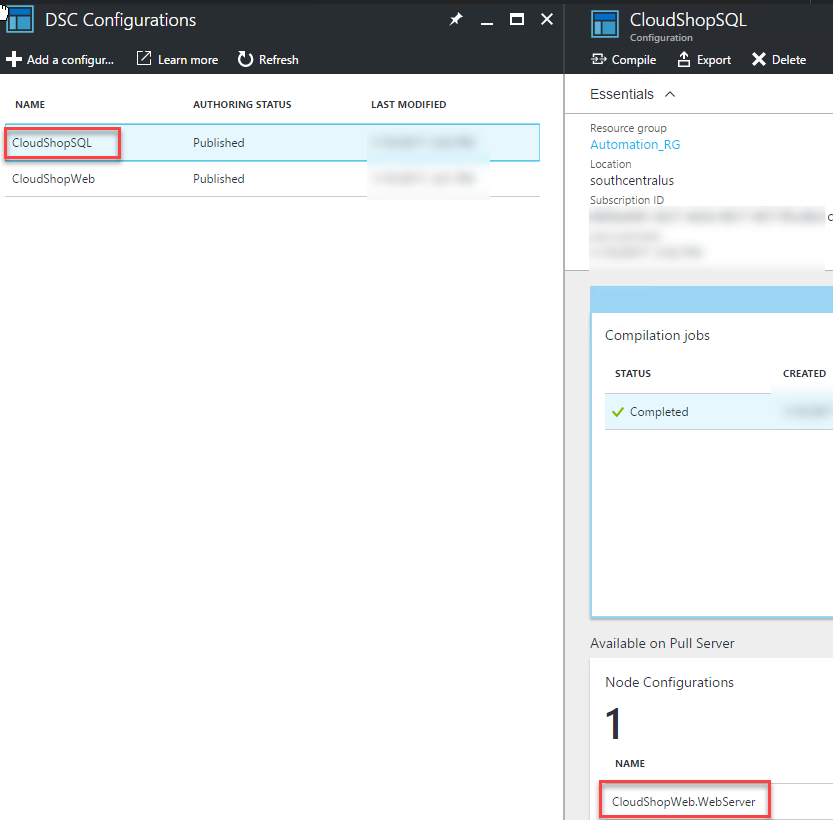
1. The information you will need to deploy your template is on the **Manage Keys** blade to the right. When doing the deployment from within Visual Studio a Key need to be provided. The **Primary Access Key** can be copied to the clipboard by clicking the button next to the Key on this blade.

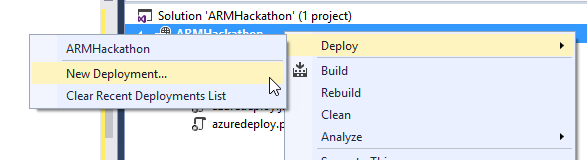
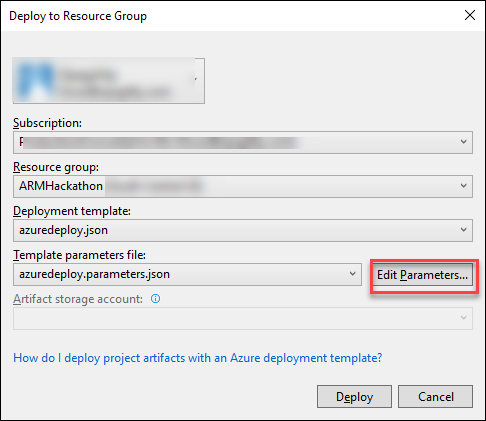


1. Using this same process, the **URL** can also be copied to use with Visual Studio.

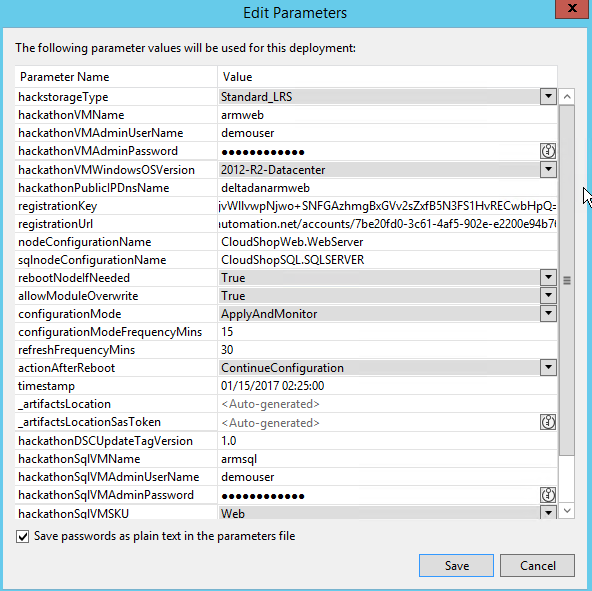


1. You will also need the name of your Node Configurations which you uploaded using the portal during Exercise 1. To find these click the DSC Configuration tile on the Azure Automation Blade. Then click the name of each to find the Node Name.



1. From within Visual Studio create a new deployment (*specify the same Resource group as before ARMHackathon)*.   
   
2. On the **Deploy to Resource Group** dialog box, click **Edit Parameters** and populate the empty values.  
     
   

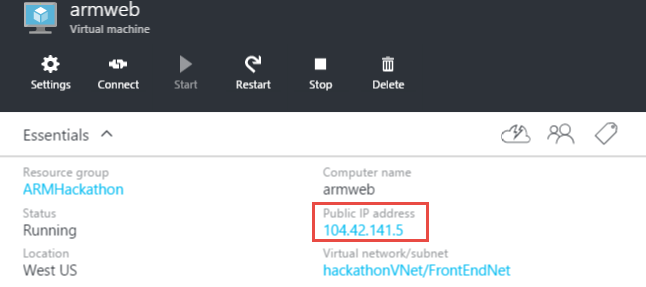
* hackathonVMName: **armweb**
* hackathonVMAdminUserName: **demouser**
* hackathonVMAdminPassword: **demo@pass123**
* hackathonPublicIPDnsName: **Choose a unique looking DNS name (must be lowercase)**
* registrationKey: **Automation account key**
* registrationUrl: **Automation registration URL**
* nodeConfigurationName: **CloudShopWeb.WebServer**
* sqlnodeConfigurationName: **CloudShopSQL.SQLSERVER**
* rebootNodeIfNeeded: **True**
* allowModuleOverwrite: **True**
* configurationModeFrequencyMins: **15**
* refreshFrequencyMins: **30**
* timestamp: **<enter current value in format like screenshot below>**
* hackathonSQLVMName: **armsql**
* hackathonVMSQLAdminUserName: **demouser**
* hackathonVMSQLAdminPassword: **demo@pass123**
* hackathonVMSQLSKU: **Web**
* vmsizeSQL: **Standard\_D3\_v2**
* **Check: Save passwords as plain text in the parameters file**



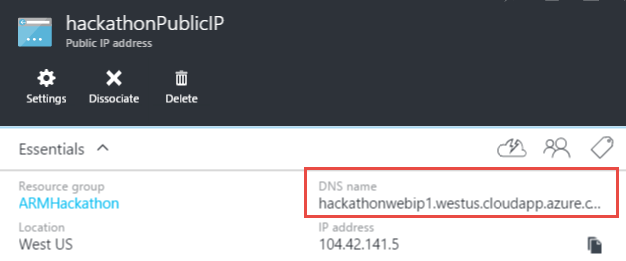
Note: The deployment may take 20 to 30 minutes to complete.

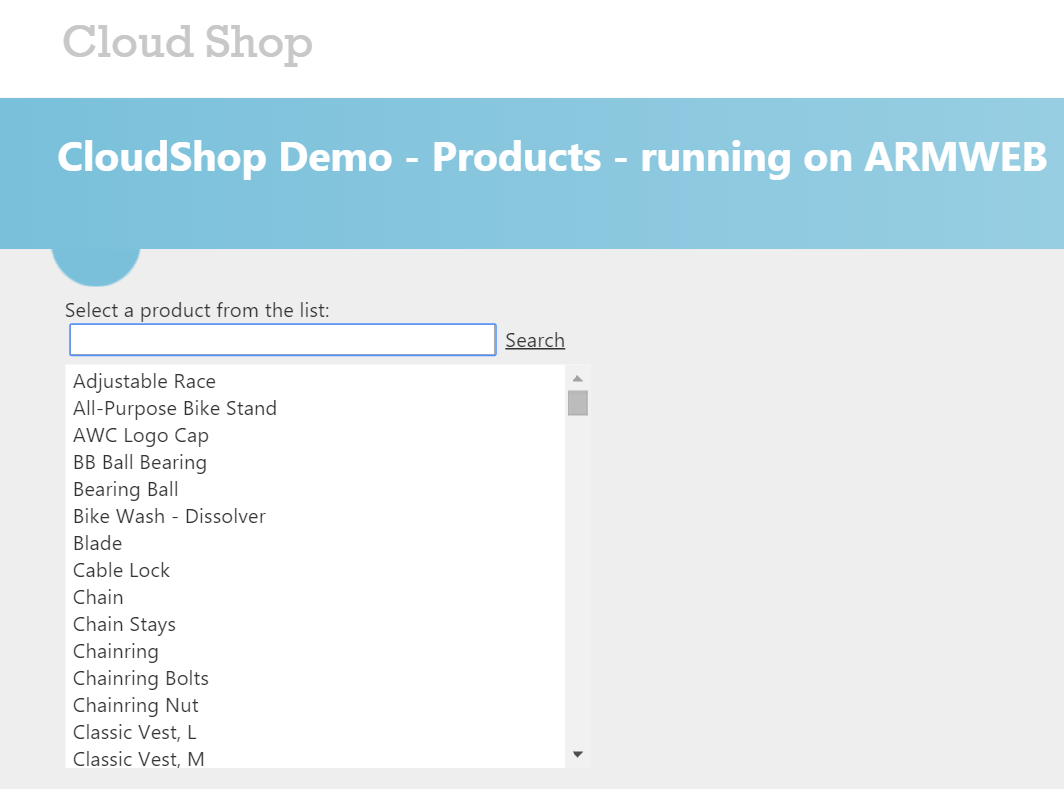
Extension Troubleshooting tip: If you make a mistake with either of the Azure DSC extensions and need to redeploy, open the virtual machine in the portal, under all settings, click extensions and remove the failed extension before deploying and also in your Automation Account under the DSC Nodes Unregister the Node.

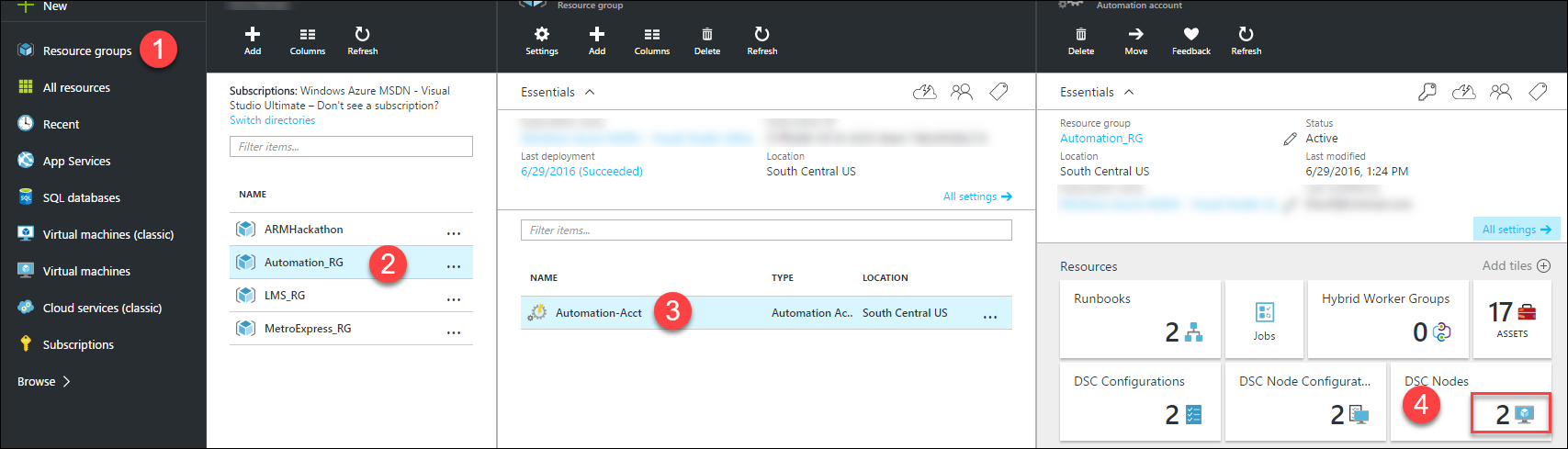
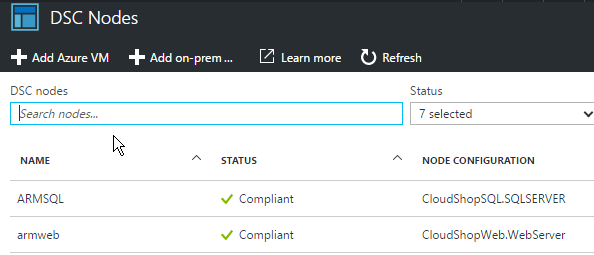
1. Launch the **Azure Management Portal** <http://portal.azure.com> and navigate to the resource group you deployed to. Click the **virtual machine** for the web server and then click the **public IP**.



1. Copy the **DNS Name** and navigate to it in the browser.





1. You should also verify that your VMs are registered as DSC Nodes in your Automation account. In the Azure Portal, click **Resource groups > Automation\_RG > Automation-Acct**, and then click the **DSC Nodes** tile   
     
     
   

### Summary

In this lab, you provisioned two virtual machines that were customized with the PowerShell DSC extension to automatically deploy a web application and a SQL Server database.

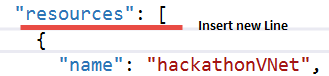
## Exercise 4: Lock down the environment

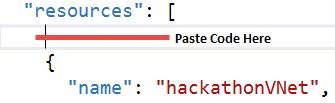
### Overview

In this portion of the exercise, you will deploy a network security group to restrict the network attack surface for the deployment.

### Task 1: Restrict traffic to the web server

1. Add the following at the beginning of the JSON template as the first item under the **resources** node. This will deploy the network security group resource and add a rule so that the only port open on the public IP is port 80.





{

"apiVersion": "2016-03-30",

"type": "Microsoft.Network/networkSecurityGroups",

"name": "hackathonNetworkSecurityGroup",

"location": "[resourceGroup().location]",

"properties": {

"securityRules": [

{

"name": "webrule",

"properties": {

"description": "This rule allows traffic in on port 80",

"protocol": "Tcp",

"sourcePortRange": "\*",

"destinationPortRange": "80",

"sourceAddressPrefix": "INTERNET",

"destinationAddressPrefix": "10.0.0.0/24",

"access": "Allow",

"priority": 100,

"direction": "Inbound"

}

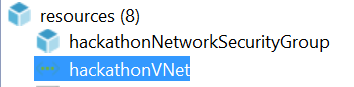
}

]

}

},

1. Click the **hackathonVNet** resource to go to its configuration.

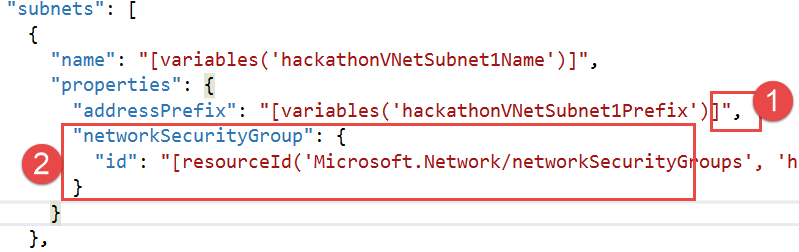


1. Associate the network security group with the **hackathonVnetSubnet1Name** subnet by adding a comma at the end of the **addressPrefix** block, and pasting in the **networkSecurityGroup** reference.

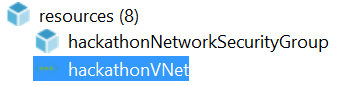
"networkSecurityGroup": {

"id": "[resourceId('Microsoft.Network/networkSecurityGroups', 'hackathonNetworkSecurityGroup')]"

}



1. Update the virtual network to have a dependency on the network security group.
2. Click the **hackathonVNet** resource to view the configuration.



1. Change the **dependsOn** configuration to refer back to the network security group.

"dependsOn": [

"[resourceId('Microsoft.Network/networkSecurityGroups', 'hackathonNetworkSecurityGroup')]"

],

1. Right-click the Visual Studio project and select **Deploy** from the context menu, followed by **New Deployment**. Click **Deploy** to update the existing deployment with the network security group.
2. To validate the network security group is working:
   * Browse to the public IP or DNS name of the web server. The site should load because traffic is allowed on port 80.
   * Connect to the web virtual machine by clicking **Connect** in the preview portal. This should fail because port 3389 is not allowed in.

### Task 2: Update the network security group to allow Windows Remote Desktop

1. Add a new rule to the network security group to allow in traffic on port 3389 Remote Desktop Protocol (RDP) by adding a comma at the end of the web rule and add the following code:

{

"name": "rdprule",

"properties": {

"description": "This rule allows traffic on port 3389 from the web",

"protocol": "Tcp",

"sourcePortRange": "\*",

"destinationPortRange": "3389",

"sourceAddressPrefix": "INTERNET",

"destinationAddressPrefix": "10.0.0.0/24",

"access": "Allow",

"priority": 200,

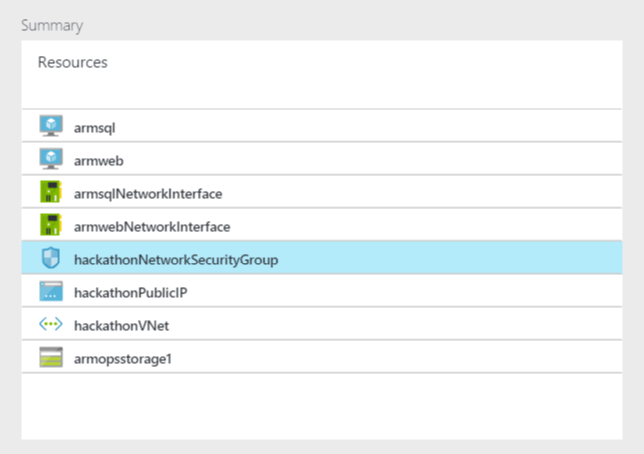
"direction": "Inbound"

}

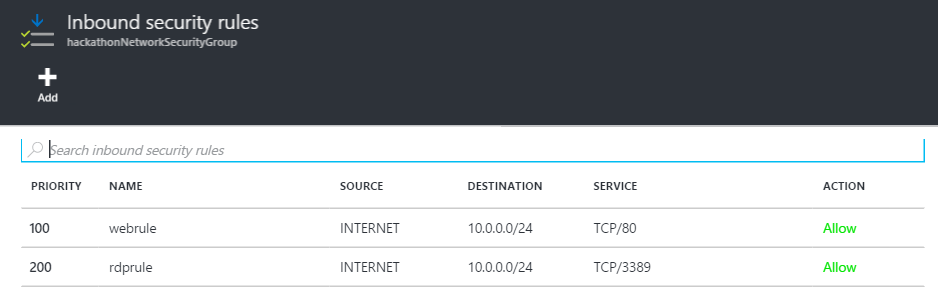
}



1. Deploy the updated rule and test RDP connectivity again.
2. Open the Azure Preview Portal, and navigate to the resource group containing your deployment.
3. Click the **hackathonNetworkSecurityGroup** in the resources summary.



1. Examine the created rule.



### Summary

In this lab, you created a network security group and applied it to the resource group to further constrain traffic to the network.

## Exercise 5: Scale out the deployment

### Overview

In this exercise, you will configure the template to scale out the web front end using a scalable number of virtual machines and storage accounts. For this, you will use the load balancer and the scale sets feature.

### Task 1: Parameterize and scale out the environment

1. Add the following variables to the end of the **variables** section of the **azuredeploy.json** file:

Tip: Do not forget to put a comma after the previous variables.

"vmSSName": "webset",

"publicIPAddressID": "[resourceId('Microsoft.Network/publicIPAddresses',variables('hackathonPublicIPName'))]",

"lbName": "loadBalancer1",

"lbID": "[resourceId('Microsoft.Network/loadBalancers',variables('lbName'))]",

"lbFEName": "loadBalancerFrontEnd",

"lbWebProbeName": "loadBalancerWebProbe",

"lbBEAddressPool": "loadBalancerBEAddressPool",

"lbFEIPConfigID": "[concat(variables('lbID'),'/frontendIPConfigurations/',variables('lbFEName'))]",

"lbBEAddressPoolID": "[concat(variables('lbID'),'/backendAddressPools/',variables('lbBEAddressPool'))]",

"lbWebProbeID": "[concat(variables('lbID'),'/probes/',variables('lbWebProbeName'))]",

"storageAccountPrefix": [

"a",

"g",

"m",

"s",

"y"

]

1. Add the following parameters to the end of the **parameters** section of the **azuredeploy.json** file (do not forget to add the comma after the last parameter).

"instanceCount": {

"type": "string",

"metadata": {

"description": "Number of VM instances"

}

},

"newStorageAccountSuffix": {

"type": "string",

"metadata": {

"description": "The Prefix for the names of the new storage accounts created"

}

}

1. Add a new storage account resource using the copy function by pasting the following code as the first resource in the list.



{

"type": "Microsoft.Storage/storageAccounts",

"name": "[concat(variables('StorageAccountPrefix')[copyIndex()],parameters('newStorageAccountSuffix'))]",

"apiVersion": "2015-06-15",

"copy": {

"name": "storageLoop",

"count": 5

},

"location": "[resourceGroup().location]",

"properties": {

"accountType": "[parameters('hackStorageType')]"

}

},

Note: This code will create five storage accounts. The virtual machine scale set will distribute the virtual machine disks across the storage accounts to ensure the VMs do not run out of IO capacity.

1. Add a load balancer resource by pasting the following code as the first resource in the list.



{

"apiVersion": "2016-03-30",

"name": "[variables('lbName')]",

"type": "Microsoft.Network/loadBalancers",

"location": "[resourceGroup().location]",

"dependsOn": [

"[concat('Microsoft.Network/publicIPAddresses/',variables('hackathonPublicIPName'))]"

],

"properties": {

"frontendIPConfigurations": [

{

"name": "[variables('lbFEName')]",

"properties": {

"publicIPAddress": {

"id": "[variables('publicIPAddressID')]"

}

}

}

],

"backendAddressPools": [

{

"name": "[variables('lbBEAddressPool')]"

}

],

"loadBalancingRules": [

{

"name": "weblb",

"properties": {

"frontendIPConfiguration": {

"id": "[variables('lbFEIPConfigID')]"

},

"backendAddressPool": {

"id": "[variables('lbBEAddressPoolID')]"

},

"probe": {

"id": "[variables('lbWebProbeID')]"

},

"protocol": "Tcp",

"frontendPort": 80,

"backendPort": 80,

"enableFloatingIP": false

}

}

],

"probes": [

{

"name": "[variables('lbWebProbeName')]",

"properties": {

"protocol": "Http",

"port": 80,

"intervalInSeconds": 15,

"numberOfProbes": 5,

"requestPath": "/"

}

}

]

}

},

Note: This code creates a load balancer resource that is listening on port 80.

1. Add the virtual machine scale set to the **resources** section using the following configuration:

{

"type": "Microsoft.Compute/virtualMachineScaleSets",

"apiVersion": "2015-06-15",

"name": "[variables('vmSSName')]",

"location": "[resourceGroup().location]",

"tags": {

"vmsstag1": "Myriad"

},

"dependsOn": [

"[concat('Microsoft.Storage/storageAccounts/a',parameters('newStorageAccountSuffix'))]",

"[concat('Microsoft.Storage/storageAccounts/g',parameters('newStorageAccountSuffix'))]",

"[concat('Microsoft.Storage/storageAccounts/m',parameters('newStorageAccountSuffix'))]",

"[concat('Microsoft.Storage/storageAccounts/s',parameters('newStorageAccountSuffix'))]",

"[concat('Microsoft.Storage/storageAccounts/y',parameters('newStorageAccountSuffix'))]",

"[concat('Microsoft.Network/loadBalancers/',variables('lbName'))]",

"[concat('Microsoft.Network/virtualNetworks/','hackathonVnet')]"

],

"sku": {

"name": "Standard\_A2",

"tier": "Standard",

"capacity": "[parameters('instanceCount')]"

},

"properties": {

"upgradePolicy": {

"mode": "Manual"

},

"virtualMachineProfile": {

"storageProfile": {

"osDisk": {

"vhdContainers": [

"[concat('https://a',parameters('newStorageAccountSuffix'),'.blob.core.windows.net/vmss')]",

"[concat('https://g',parameters('newStorageAccountSuffix'),'.blob.core.windows.net/vmss')]",

"[concat('https://m',parameters('newStorageAccountSuffix'),'.blob.core.windows.net/vmss')]",

"[concat('https://s',parameters('newStorageAccountSuffix'),'.blob.core.windows.net/vmss')]",

"[concat('https://y',parameters('newStorageAccountSuffix'),'.blob.core.windows.net/vmss')]"

],

"name": "vmssosdisk",

"caching": "ReadOnly",

"createOption": "FromImage"

},

"imageReference": {

"publisher": "[variables('hackathonVMImagePublisher')]",

"offer": "[variables('hackathonVMImageOffer')]",

"sku": "[parameters('hackathonVMWindowsOSVersion')]",

"version": "latest"

}

},

"osProfile": {

"computerNamePrefix": "[variables('vmSSName')]",

"adminUsername": "[parameters('hackathonVMAdminUserName')]",

"adminPassword": "[parameters('hackathonVMAdminPassword')]"

},

"networkProfile": {

"networkInterfaceConfigurations": [

{

"name": "nic1",

"properties": {

"primary": true,

"ipConfigurations": [

{

"name": "ip1",

"properties": {

"subnet": {

"id": "[variables('hackathonVMSubnetRef')]"

},

"loadBalancerBackendAddressPools": [

{ "id": "[variables('lbBEAddressPoolID')]" }

]

}

}

]

}

}

]

},

"extensionProfile": {

"extensions": [

{

"name": "hackathonDSC",

"properties": {

"publisher": "Microsoft.Powershell",

"type": "DSC",

"typeHandlerVersion": "2.19",

"autoUpgradeMinorVersion": true,

"protectedSettings": {

"Items": {

"registrationKeyPrivate": "[parameters('registrationKey')]"

}

},

"settings": {

"modulesUrl": "https://opsgilityweb.blob.core.windows.net/20160629-arm-hackathon/RegistrationMetaConfigV2.zip",

"configurationFunction": "RegistrationMetaConfigV2.ps1\\RegistrationMetaConfigV2",

"Properties": [

{

"Name": "RegistrationKey",

"Value": {

"UserName": "PLACEHOLDER\_DONOTUSE",

"Password": "PrivateSettingsRef:registrationKeyPrivate"

},

"TypeName": "System.Management.Automation.PSCredential"

},

{

"Name": "RegistrationUrl",

"Value": "[parameters('registrationUrl')]",

"TypeName": "System.String"

},

{

"Name": "NodeConfigurationName",

"Value": "CloudShopWeb.WebServer",

"TypeName": "System.String"

},

{

"Name": "ConfigurationMode",

"Value": "[parameters('configurationMode')]",

"TypeName": "System.String"

},

{

"Name": "ConfigurationModeFrequencyMins",

"Value": "[parameters('configurationModeFrequencyMins')]",

"TypeName": "System.Int32"

},

{

"Name": "RefreshFrequencyMins",

"Value": "[parameters('refreshFrequencyMins')]",

"TypeName": "System.Int32"

},

{

"Name": "RebootNodeIfNeeded",

"Value": "[parameters('rebootNodeIfNeeded')]",

"TypeName": "System.Boolean"

},

{

"Name": "ActionAfterReboot",

"Value": "[parameters('actionAfterReboot')]",

"TypeName": "System.String"

},

{

"Name": "AllowModuleOverwrite",

"Value": "[parameters('allowModuleOverwrite')]",

"TypeName": "System.Boolean"

},

{

"Name": "Timestamp",

"Value": "[parameters('timestamp')]",

"TypeName": "System.String"

}

]

}

}

}

]

}

}

}

},

This code creates a virtual machine scale sets resource that will create as many instances of the virtual machine as specified in the instanceCount parameter. The DSC extension will execute on each VM when it is created to configure the cloud shop web application. The scale set will distribute the VM disks across the previously created storage accounts.

1. Delete the existing **hackathonVM** and the **hackathonVMNic** resources by right-clicking each resource and clicking **Delete**.



This VM and NIC will be replaced by the VMs in the scale set.

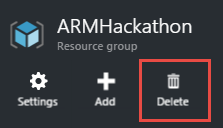
1. Delete the existing deployment (to save on core quota) by opening the Azure portal (portal.azure.com) in your browser.
2. Click **Resource groups**.



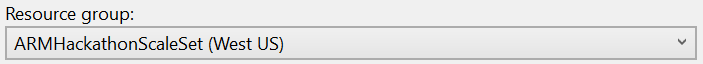
1. Click the **ARMHackathon** resource group (or whatever you named your deployment).



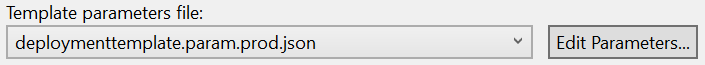
1. Click **Delete**, and then confirm by typing in the name of the resource group.



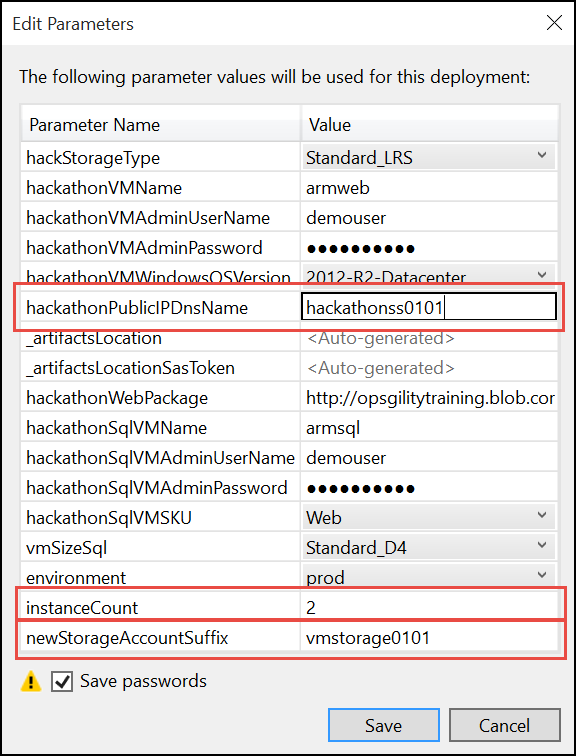
1. Create a **new deployment** and choose a new **resource group**. Name the new resource group **ARMHackathonScaleSet**.



1. Choose any of the template parameters files, and click **Edit Parameters**.



1. Provide a unique value for the **hackathonPublicIPDnsName** and **newStorageAccountSuffix** parameters. Enter a value of 2 for the **instanceCount**. Click **Save** and then **Deploy**.



Note: The deployment may take 20 to 30 minutes to complete. If Visual Studio fails monitoring the solution with an error about the SAS Token expiring you can open the resource group in the Portal and you can monitor the deployment by clicking the link under the Last Deployment lab on the essentials pane.

1. Within the **Azure Management Portal**, open the **resource group** and click the **hackathonPublicIP** resource.



1. Copy the **DNS name** and navigate to it in a browser to validate that the load balancer and the scale set are working. Click **Refresh** several times and the page should flip from WEBSET-0 to WEBSET-1.

