MultiServerDeploy Solution Documentation

Prepared for

9/14/2016

Prepared by

Boklyn Wong

boklynw@microsoft.com

Table of Contents

[1 Overview 1](#_Toc451547954)

[2 Pre-requisites 1](#_Toc451547955)

[2.1 script-createdeploystorage 1](#_Toc451547956)

[3 Configure multideploy parameters file. 2](#_Toc451547957)

[4 Multi-server deploy usage 4](#_Toc451547958)

1. Overview

The multideploy server solution allows you to deploy any number of servers with the following baseline support:

* Provision any virtual machines from the Microsoft Marketplace galary images.
  + Currently validated to provision Windows Server 2012 and SQL combinations.
  + In the appendix provides instructions for additional customizations.
* Provision up to 16 additional data disks
  + Provides capabilities to initializes those disks in storage pools in any volume size you specify as long as the volume size(s) are less than or equal to the amount of availabile disk space registered on the OS level.
  + The solution requires that any server provisioned has to have at least one additional disk. Machines with 0 additional disks would not be supported in this solution.
* Allow you to specific different storage account placement for each virtual machine.

The following addons are available:

* Keyvault integration in parameters file to pass in secrets of passwords as well a dependent template url paths.
* Joins your provisioned machine to the domain of your choosing as long as vnet connectivity is available upon provisioning.
* Leverages keyvault for retrieving passwords and secondary template URL information(s).

This documentation has been modified to provide modification steps need to deploy the nMarket application for ABB Enterprise

1. Pre-requisites

The multiserver deploy solution depends on an external location to retrieve secondary templates, custom script extensions, and other assets in order to function. You can either use a github repositiory or blob storage for storing your assets. This guide will cover the steps involved in setting up your blob storage to store your asset.

The following assumptions are made by the multideploy solution:

1. The Vnet is already provisioned with the corresponding network information is available.
   1. There is a sample vnet with a single subnet solution provided.
   2. Configure deployment storage account.

The entire multideploy solution depends on the availability of an external public location to pull secondary assets. This section would cover the script needed to provision this secondary location.

* + 1. script-createdeploystorage

This script will create the deployment storage account needed by the multideploy solution. The container that will be created will be public. The following information needs to be entered in order for this script to work.

* Subid, needs to reference the subscription id of your environment
  + Please see appendex for instructions for retrieving subid value
* Tenid, needs to reference the tenant id of your environment
  + Please see appendex for instructions for retrieving tenid value
* Storlocation, the location where you want the storage account to be placed.
* Resourcegrp, the resource group where you want the storage account is being placed.
* Storagename, the name of the storage account to be provisioned.
  + This value is important because the solution depends on this value
* Containername, the name of the container in which the assets are going to be used.
  + This value is important because the solution depends on this value

This script will transfer all files needed by the asset folder into the storage account created by the script.

1. Configure multideploy parameters file.

The following parameter values need to be changed/validated for using the Multi-Deploy template:

* diagStorageAccountName
  + This solution seperates out the location of the VM diagnostic files to a different storage account than the account in which the VHDs are located.
* VHDStorageAccountNames
  + This is an array of storage accounts names that you want each machine provisioned to be placed under.
  + If you are provisioning two machines and you want them on two different storage accounts, the value would then be: [ "storagename1", "storagename2" ]
  + If you are provisioning two machines and you want them on the same storage account then the value would be: [ "storagename1", "storagename2" ]
* existingVNETName
  + The Vnet in which the tier of servers will be deployed to.
* Vnetrgpname
  + The resource group in which the vNet you want to depoloy your servers is a member of.
* existingSubnetName
  + the subnet of the vnet that will house your servers.
* domainToJoin
  + This is the domain in which your machines will be joined to.
* domainUsername
  + This is a domain user that has the rights to join the machine to.
  + This is an optional value when provisioning stand alone servers.
* domainPassword
  + You need to ensure that the ID of the KeyVault is correct
  + You need to ensure that the secret of the KeyVault is correct. If choosing the keyvault option.
  + This domain password would be retrieved from KeyVault.
  + This is an optional value when provisioning stand alone servers.
* vmAdminUsername
  + The local useradmin name
* vmAdminPassword
  + You need to ensure that the ID of the KeyVault is correct.
  + You need to ensure that the secret of the KeyVault is correct.
* optionsselectpath
  + You need to ensure that the ID of the KeyVault is correct.
  + You need to ensure that the secret of the KeyVault is correct.
  + This solution depends on this secondary template to enable the ability to provision
* Nicoffset
  + The NIC offset enables the deployment solution to append a numberical number to the nicnameformat. If your nickname format is specificed as cust1w1dr-nic- and the nickname offset is 1, then the first nic will have cust1w1dr-nic-1 and the second will have cust1w1dr-nic-2
* Sitecount
  + The sitecount value represents the number of systems expected to be deployed.
* Nicnameformat
  + This represents the name format of the nic(s) to be provisioned by the solution.
* Vmnames
  + This represents the name(s) of virtual machines to be provisioned. If your building two servers, then the value for the is parameter will be [ "cust1e1-bld-05", "cust1e1-bld-06" ]
* vmSizes
  + This parameter value repsents the size(s) of the virtual machines you need to be provisioned. For the two server example, the value will be [ "Standard\_DS2", "Standard\_DS6"], These values can from any officially supported vmSizes.
* Vmstaticips
  + This solution assumes that the virtual machines you will provision will leverage static ip(s). The value for the two server will be [ "172.26.14.17", "172.26.14.18" ].
* diskcount
  + The nature of this solution requires you to always have at least 1 data disk. The scenario for 0 data disk is not covered by this solution.
* Disksizes
  + This value represents the sizes of VHDs you want the solution to provision. Bear in mind that the limitation for disk sizes is 1 TB. This solution supports up to 16 disks. The following example would create two disks with each disk having 1000 GB for the first server and two disks with each disk having 200 GB for the second server.

[

[ 1000, 1000, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],

[ 200, 200, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ]

]

You have to specifiy all 16 disks options. In scenarios where you need less than 16 disks you would have to specify the other disk with 0 values.

* Initdisksizes
  + This solution also allows you to also initialize the disks on the OS level through Windows Server storage pool. The solution will use this value to provision the virtual disks. The following example will create 4 virtual disks each of size of 450 GB for the first server and 4 virtual disks each of size 75 GB. Bear in mind the total sizes of the virtual disks must be less than the size of the total disks requestd in the Disksizes parameters.

[

"450,450,450,450",

"75,75,75,75"

]

* windowsOSversion
  + This is where you would specify the type of OS version to install on the virtual machines. Currently 2012-R2-Datacenter, has been tested and validated.

1. Multi-server deploy usage with demo templates/scripts

The multi deploy solution includes scripts and templates to provision a sample environment leveraging the multi-deploy template solution. Ensure that you have followed instruction from Section 2.1 to execute script-createdeploystorage script. The scripts and templates needs to be executed and deploy in the following order:

script-createdeploystorage.ps1

This script will provision the deployment storage account and resource group needed by all the steps below.

cust1-script-createkeyvault.ps1

This script will create the keyvault for your environment

cust1-script-addkeyvaultsecrets.ps1

This script will add the secrets to the keyvault in step 2.

script-deploynetwork.ps1

This script leverages the following template/parameter files:

cust1-template-network.json

cust1-parameters-network.json

script-deploydomaincontrollers.ps1

This script leverages the following template/parameter files:

cust1-template-domaincontrollers\_withkey.json

cust1-parameters-key.domaincontrollers.json

script-deploymachines.ps1

This script leverages the following template/parameter files:

cust1-template-multisvrdeploy\_withkey.cse.json

cust1-parameters-key.admintier.svrs.json

* 1. cust1-script-createkeyvault.ps1

This section will highlight the bare minimum variable values you need to change in order deploy the demo network in your environment.

* Subid
  + Your specific subscription id
* Tenid
  + Your specific tenant id
* Kvaultname
  + This will have to be a unique name to provision your keyvault specific to your organization. This value needs to be updated in the addkeyvaultsecrets script.

At the end of executing the script you would need to record the value ResourceId that would needed, when you update the parameters file that depends on keyvault secrets. Before you proceed to the next step of 4.4 you might need to wait 5-10 minutes for the keyvault provisioning process to complete.

* 1. cust1-script-addkeyvaultsecrets.ps1

This section will highlight the bare minimum variable values you need to change in order deploy the demo network in your environment.

* Subid
  + Your specific subscription id
* Tenid
  + Your specific tenant id
* Kvaultname
  + This is the value entered in section 4.3

**NOTE:** You might get the following error, this might be a scenario where the keyvault has not been provisioned successfully. You can execute the script a moment later and the issues should be resolved.

dd-AzureKeyVaultKey : The remote name could not be resolved:

* 1. script-deploynetwork configuration instructions

This section will highlight the bare minimum variable values you need to change in order deploy the demo network in your environment.

* Subid
  + Your specific subscription id
* Tenid
  + Your specific tenant id
* Workingdir
  + The working directory path where the multideploy solution is located.
  1. script-deploydomaincontrollers configuration instructions

This section will highlight the bare minimum variable values you need to change in order deploy the demo network in your environment.

* Subid
  + Your specific subscription id
* Tenid
  + Your specific tenant id
* Workingdir
  + The working directory path where the multideploy solution is located.
    1. Update keyvault parameters file

You would neeed to modify the cust1-parameters-key.domaincontrollers file to update the following parameters with the keyvault id with the value retrieved from section 4.3.

* adminPassword
* pdcdscpkgurl
* bdcdscpkgurl
* nictemplateurl
* vnetdnstemplateurl
* vnetdnsparamurl

The following example would show you the place where you need to change the value:

"keyVault": {

"id": "VALUE TO BE CHANGED "

},

* 1. script-deploymachines configuration instructions

This section will highlight the bare minimum variable values you need to change in order deploy the demo network in your environment.

* Subid
  + Your specific subscription id
* Tenid
  + Your specific tenant id
* Workingdir
  + The working directory path where the multideploy solution is located.
    1. Update keyvault parameters file

You would neeed to modify the cust1-parameters-key.admintier.svrs.json file to update the following parameters with the keyvault id with the value retrieved from section 4.3.

* domainPassword
* vmAdminPassword
* optionsselectpath
* initidiskpath

The following example would show you the place where you need to change the value:

"keyVault": {

"id": "VALUE TO BE CHANGED "

},

1. Nmarket modifications
   1. Prepare the network

The solution comes with a base template “base-template-network.json” that you would need to modify for your environment. This base template will provision a network with only 1 subnet. In the event that you need to add additional subnets you would need to add the following code snippets for each additional subnets:

* Parameters
  + Replace the apptiersubnetname and apptiersubnetrange to the new name of the new subnet.

"newtiersubnetname": {

"type": "string",

"metadata": {

"description": "new tier Subnet Name"

}

},

"newtiersubnetrange": {

"type": "string",

"metadata": {

"description": "new tier Subnet Range"

}

},

* Variable code
  + After line 54 you would need to add the new variable representing the new subnet.
  + You would need to replace the two parameters reference of subnetname and subnetrange with the correct values.

"newtiersubname": "[concat(parameters('newtiersubnetname'),'\_',replace(parameters('newtiersubnetrange'),'/','\_'))]"

* Subnet Code
  + Line 68 holds the subnet array. For each additional subnet you would need to add the following code:
  + You would need to replace thew variable newtiersubname and the parameter value with the correct one.

{

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

"properties": {

"addressPrefix": "[parameters('newtiersubnetrange')]"

}

}

* 1. Prepare the deployment account

nMarket has two primary servers and as part of their provisioning phase would require the execution of additional steps to pull the software and install it on the provisioned server. The script will perform the following actions:

* Initialized the disk for use.
* Use the initiatlized disk by copying the software from blob storage and installing the required files onto the initialized disk.
* Execute the necessary steps to pull the nmarket installation zip file from blob storage and install it onto the provisioned server.

You would need to rename the nmarket\_app\_basescript.ps1 to the servername of the app server so that the deployment sequence will know where to pull the script from. Since we have two servers, there should be two files of the following names:

* abbsc-nmap-01
* abbsc-nmap-02
  1. Create your region specific parameter files.

nMarket template files are region specific. This requires you to provide region specific parameter files to provision the complete solution for a specific regtion. The following parameter files are needed for each region:

* nmarket-parameters-network<region\_name>
  + The values are pretty self-explaintory. Please see the other region’s file for reference.
* nmarket-parameters-key.svrsnodomain<region\_name>
  + This parameter file governs the type(s) of server(s) you are provisioning. Please see section 3 for additional information about how to configure this file. For nMarket you can leverage other region’s files for reference.
* nmarket-parameters-ip<region\_name>
  + This parameters file will allow you to provision the public ip address used by the servers. The only value of particular importance is the vmnames. For this parameter the value must be the names of the server(s) you are provision for the region. This has no template functionality, but is used by the deployment script. For the south west region, the following value was used: [ "abbsc-nmap-04", "abbsc-nmap-05", "abbsc-nmis-06" ]
* nmarket-parameters-storage<region\_name>
  + This parameters file will allow you to provision any number of storage account to be used by your region deployments. The only values of importance is the name(s) and the storage account type(s) for each account(s).
* nmarket-parameters-tm<region\_name>
  + This parameters file will allow you to provision the traffic manager profile to govern the failover of the app servers. The value that needs to be modified will be the type of traffic manager profile you want, and the unique dns name that the traffic manager will respond to.
  1. Configure the script-deploynmarket

This solution will orchestrate the steps needed to successfully deploy nMarket’s infrastructure and software. The following values must be known prior the execution of the script:

* Subscription id, This is the subscription identifier where you plan to install nmarket onto.
  + Please see appendix 6.4, for more information on retrieving this value.
* Tenet id, This is the tenet identifier where you plan to install nmarket onto.
  + Please see appendix 6.4 for more information on retrieving this value.
* Azure SQL server name, This is the sql server name in which the nMarket application will connect to.
  + This value is passed from the template to the initialization script.
* Dbname, This is the database name that the nMarket application will connect to.
  + This value is passed from the template to the initialization script.
* Nmarketstorage, this value is the name of the storage account that contains the nMarket installation files.
* Nmarketgrp, This value is the resource group that holds the storage account that contains the nMarket installation files. This value is used to generate the one time access token to the installation files.
* Networkrggrp: This value holds the name of the resource group to hold the network components
* Svrrggrp, This value holds the name of the resource group to hold the server and dependent components.
* Location, This is the location in which you are deploying the nmarket region.

1. Appendix

This appendix will cover additional scenarios cover by multi-deploy but would require some modification of the template file. Each scenario will have detailed instructions on what you need to change to achieve the additional functionality.

* 1. Dynamic IP

If you want all machines in your environment to leverage dynamic ip instead of the static ip option you would perform the following steps.

* + - 1. Make a copy of the cust1-template-multisvrdeploy\_withkey.cse.json file with the name cust1-template-multisvrdeploydynip\_withkey.cse
      2. To switch the current template from static ip you would need to swap out the following code at line 195:

The following code allows for static ip:

"properties": {

"privateIPAllocationMethod": "Static",

"privateIPAddress": "[parameters('vmstaticips')[copyIndex()]]",

"subnet": {

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

}

}

Replace it with the following code:

"properties": {

"privateIPAllocationMethod": "Dynamic",

"subnet": {

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

}

}

Now you have a template that supports provisioning of multiple servers and all servers provisioned will leverage dynamic ip allocation method. This solution would work wheter you choose/not choose to remove the following section of your parameters file:

"vmstaticips": {

"value": [ "172.26.14.17" ]

},

**NOTE:** We do however reccomend you remove the value to provide a clean and easy to understand parameters file.

The template in the event that it does not detect this parameter value will substitute a empty value since it is no longer being consumed.

* 1. Non-Domain Join servers

There are scenarios in which you need to provision a series of servers in which they do not have to be joined to any domain controllers. To leverage multideploy without domain join just remove code lines from 313-338. Also remove the comman on line 312. These lines are replicated in the following section:

{

"apiVersion": "[variables('apiVersion')]",

"type": "Microsoft.Compute/virtualMachines/extensions",

"name": "[concat(parameters('vmnames')[copyIndex()],'/joindomain')]",

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

"dependsOn": [

"[concat('Microsoft.Compute/virtualMachines/', parameters('vmnames')[copyIndex()])]",

"[resourceId('Microsoft.Resources/deployments', concat(toLower(parameters('vmnames')[copyIndex()]),'-disksel'))]",

"[resourceId('Microsoft.Compute/virtualMachines/extensions', parameters('vmnames')[copyIndex()],'cseexec')]"

],

"properties": {

"publisher": "Microsoft.Compute",

"type": "JsonADDomainExtension",

"typeHandlerVersion": "1.0",

"settings": {

"Name": "[parameters('domainToJoin')]",

"OUPath": "[parameters('ouPath')]",

"User": "[concat(parameters('domainToJoin'), '\\', parameters('domainUsername'))]",

"Restart": "true",

"Options": "[parameters('domainJoinOptions')]"

},

"protectedsettings": {

"Password": "[parameters('domainPassword')]"

}

}

}

This option would work if you coose to remove or not remove the domain specific parameters of the following

* 'domainToJoin'
* 'domainJoinOptions'
* 'domainPassword'

**NOTE:** We do however reccomend you remove the value to provide a clean and easy to understand parameters file.

* 1. No-keyvault integration
  2. Retrieve subid and tenid values