Getting Started with Azure PowerShell - Part 1

Sivaprasad Padisetty (July 2013)

*Windows Azure PowerShell is a powerful scripting environment that you can use to control and automate the deployment of virtual machines and other resources in Azure. This blog post will cover the basics to get started. By the end you should able to create a VM in the cloud and manage its lifecycle. In the next blog I will cover how to run remote commands against this VM. Since you are going to pay for the actual duration you are using, this automation will be extremely useful.*

# Install Azure Module

Install the PS module from <http://www.windowsazure.com/en-us/downloads>. This installation happens via Web Platform Installer and it will automatically install all the dependencies.

Make sure to enable script execution in PS if not already done. This can be done by running “Set-ExecutionPolicy RemoteSigned” from an elevated prompt.

Import the azure module, to access Azure cmdlets. On a 64 bit version of windows, the command would be:

Import-Module 'C:\Program Files (x86)\Microsoft SDKs\Windows Azure\PowerShell\Azure'

Once imported you can check the module with Get-Module cmdlet.

PS C:\windows\system32> Get-Module

ModuleType Name ExportedCommands

---------- ---- ----------------

Binary Azure {Add-AzureCacheWorkerRole, Add-AzureCertificate...

Script ISE {Get-IseSnippet, Import-IseSnippet, New-IseSnip...

Manifest Microsoft.PowerShell.Management {Add-Computer, Add-Content, Checkpoint-Computer...

Manifest Microsoft.PowerShell.Utility {Add-Member, Add-Type, Clear-Variable, Compare-...

You can run Get-Command, to get the list of cmdlets supported by the azure module.

PS C:\windows\system32> Get-Command -Module azure

CommandType Name ModuleName

----------- ---- ----------

Cmdlet Add-AzureCacheWorkerRole Azure

Cmdlet Add-AzureCertificate Azure

Cmdlet Add-AzureDataDisk Azure

Cmdlet Add-AzureDisk Azure

<... output truncated ...>

# Configure your workstation to connect with Azure

You need to do this setup only once in your workstation. To connect with Azure you need three pieces of information a) Subscription name b) Subscription ID and c) Certificate. You can either supply this information manually or from a publishing settings file. Manual configuration can be done by cmdlet “Set-AzureSubscription” Let’s do the easy way! Run the cmdlet “Get-AzurePublishSettingsFile” to download the settings file, and import into PowerShell. (Note: It will open the browser, you need to login to your azure account if not already logged in).

#Run the following cmdlets once per workstation.

Get-AzurePublishSettingsFile

Import-AzurePublishSettingsFile 'C:\temp\azure.publishsettings’

If you have more than one subscription then, use Get-AzureSubscription to get current subscription and Select-AzureSubscription to select active subscription. Other azure cmdlets uses current subscription in talking to azure. Now you are ready to talk to Azure!

# VM Lifecycle management

Get-AzureVM lists the VMs in your current subscription. Note you don’t have to specify any credentials as you have configured the certificate for authentication in the previous step.

PS C:\windows\system32> Get-AzureVM

ServiceName Name Status

----------- ---- ------

sivadomain azurevm1 StoppedDeallocated

To create a new VHD you need to specific what OS image to use. Azure comes with tons of stock images. If you are not happy with that you even upload your own image. In this case I am going to use the stock image for Windows Server 2012 Datacenter. As there can be multiple datacenter images, I am going to select the last one. “?” is a short form for where. Following script gets the image, saves it into $image, and outputs the label to console.

$image = Get-AzureVMImage |

? label -Like "Windows Server 2012 datacenter\*" |

select -Last 1

$image.Label

Test if the servicename is available. Since service name maps to DNS name it has to be globally unique.

PS C:\windows\system32> Test-AzureName -Service -name sivablogvm1

False

Following script, creates two VMs. They are created differently to illustrate both ways.

#get the Server 2012 datacenter image

#select the last one that will likely be the latest

$image = Get-AzureVMImage |

? label -Like "Windows Server 2012 datacenter\*" |

select -Last 1

$image.Label

#location where to create the VM

$location = Get-AzureLocation | ? Name -EQ 'West US'

#There are two ways to create the VM

#blogvm1 is created using New-AzureQuickVM

#blogvm2 is created using New-AzureVM

#service name is also the DNS name and hence has to be globally unique.

New-AzureQuickVM -Name "sivablogvm1" `

-Windows `

-ServiceName "sivablogvm1" `

-Location $location.Name `

-ImageName $image.ImageName `

-InstanceSize ExtraSmall `

-AdminUsername "siva" `

-Password "Pass@word" `

-EnableWinRMHttp `

-WaitForBoot

New-AzureVMConfig -Name "sivablogvm2" -InstanceSize ExtraSmall -ImageName $image.ImageName |

Add-AzureProvisioningConfig –Windows -AdminUsername "siva" –Password "Pass@word" -EnableWinRMHttp |

New-AzureVM -ServiceName "sivablogvm2" -WaitForBoot -Location $location.Name

You can login to the portal and see these VMs. To check the VMs from PS use Get-AzureVM:

PS C:\windows\system32> Get-AzureVM

ServiceName Name Status

----------- ---- ------

sivablogvm1 sivablogvm1 ReadyRole

sivablogvm2 sivablogvm2 ReadyRole

Finally you can delete sivablogvm2 using “Remove-AzureVM”:

PS C:\windows\system32> Get-AzureVM -ServiceName "sivablogvm2" | Remove-AzureVM

PS C:\windows\system32> Get-AzureVM

ServiceName Name Status

----------- ---- ------

sivablogvm1 sivablogvm1 ReadyRole

# Start/Stop VM

|  |
| --- |
| #stop the VM  Stop-AzureVM -ServiceName "sivablogvm1" -Name "sivablogvm1" -Force  #start the VM  Start-AzureVM -ServiceName "sivablogvm1" -Name "sivablogvm1"  #restart the VM  Restart-AzureVM -ServiceName "sivablogvm1" -Name "sivablogvm1" |

# Azure Disks

Every VM in Azure is backed by a disk where the state is stored. Disk can either be an OS disk or data disk. In our case, there is no data disk, the OS disk is implicitly created. Deleting a VM does not delete the associated disk. The following script queries the disk and finally deletes the OS disk associated with sivablogvm2.

#to get OS disk associated with the VM

Get-AzureVM -ServiceName "sivablogvm1" -Name "sivablogvm1" |

Get-AzureOsDisk

#to get data disk associated with the VM

#since we did not associate a seperate data disk, this will be empty

Get-AzureVM -ServiceName "sivablogvm1" -Name "sivablogvm1" |

Get-AzureDataDisk

#get the disk associated with the deleted sivablogvm2

#deleting the VM will not delete the associated disk

Get-AzureDisk | ? diskname -Like 'sivablogvm2\*'

#now delete the associated disk

Get-AzureDisk | `

? diskname -Like 'sivablogvm2\*' |

Remove-AzureDisk

In part 2, I will go over automating and connecting to the VM.

Explore & Enjoy!

/Siva