Getting Started with Azure PowerShell - Part 2

Sivaprasad Padisetty (July 2013)

*Windows Azure PowerShell is a powerful scripting environment that can be used to control and automate the deployment of virtual machines and other resources in Azure. In this blog I will cover how to run remote commands against VMs in Azure. You may want to read part1 first before reading this blog. In part 1, I covered the setup of azure PS module and create VM.*

# Check Configuration

Get-AzureVM, gets the list of all the VMs. If you want to select a specific VM, then specify the servicename and vmname.

PS C:\windows\system32> Get-AzureVM

ServiceName Name Status

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

sivablogvm1 sivablogvm1 ReadyRole

Now check if the end points are configured correctly. Endpoints are the way in which Azure VM can communicate with the external world. The Azure VMs are not directly exposed to internet, they are behind the external firewall/load balancer. The DNS name/public IP leads to the Azure’s load balancer. The end point maps to a public IP/public facing port number to VMs local IP address/VMs local port number. This means multiple VMs can share the same public facing IP address.

PS C:\windows\system32> Get-AzureVM -ServiceName "sivablogvm1" -Name "sivablogvm1" |

Get-AzureEndpoint

LBSetName :

LocalPort : 3389

Name : RemoteDesktop

Port : 61242

Protocol : tcp

Vip : 138.91.88.237

ProbePath :

ProbePort : 0

ProbeProtocol :

ProbeIntervalInSeconds :

ProbeTimeoutInSeconds :

EnableDirectServerReturn : False

Acl : {}

LBSetName :

LocalPort : 5986

Name : WinRmHTTPs

Port : 64344

Protocol : tcp

Vip : 138.91.88.237

ProbePath :

ProbePort : 0

ProbeProtocol :

ProbeIntervalInSeconds :

ProbeTimeoutInSeconds :

EnableDirectServerReturn : False

Acl : {}

Above output shows that sivablogvm1 has two endpoints configured. One for remote desktop and other for remote PowerShell (WinRmHTTPs). In case of remote PS, local port is 5986, whereas the public facing port is 64344.

You can use “**mstsc /v:sivablogvm1.cloudapp.net:61242**”, to connect to the remote desktop. Where 61242 is the public facing port number for the remote desktop.

# Remote command Execution

Get-AzureWinRMUri, is used to get the WInRMUri, which is needed for PS remoting. This cmdlet will return the Uri object, call ToString method to get the string representation. In our case the connection string is <https://sivablogvm1.cloudapp.net:64344>. This means the public DNS name is sivablogvm1.cloudapp.net, public WinRM port is 64344. Use the following script to remotely invoke “Get-Process” in the VM sivapblogvm1. (Note: Hardcoding password in the script is not a good idea. Use Get-Credential, instead)

#Get-AzureWinRMUri returns System.Uri object

#call ToString method to get the connection string for PS Remoting

$uri = Get-AzureWinRMUri -ServiceName "sivablogvm1" -Name "sivablogvm1"

$uri.ToString()

#Skip Certificate Authority check

#This is because of generated certificate with no trusted root

$opts = New-PSSessionOption -SkipCACheck

#create the securestring

$SecurePassword = ConvertTo-SecureString -String "Pass@word" -AsPlainText -Force

$cred = New-Object PSCredential -ArgumentList "sivablogvm1\siva", $SecurePassword

Invoke-Command -ConnectionUri $uri `

-Credential $cred `

-ScriptBlock {Get-Process} `

-SessionOption $opts

In the above case, PowerShell behind the scenes establishes connection, executes the command, and destroys the connection. Cost of establish and teardown the connection is not trivial. If you want to run multiple commands, there is more efficient way to do this.

#It is much more efficient to establish the session once

#and reuse the same session and execute multiple commands.

$s = New-PSSession -ConnectionUri $uri `

-Credential $cred `

-SessionOption $opts

#pass the session created above when invoke a command

Invoke-Command -Session $s {Get-Process}

#once you are done, remove the PS Session

Remove-PSSession $s

# Interactive Session

Thus far, I showed you how to run remote commands. Another option is to establish a remote interactive session. This is a convenient option for interactive debugging/exploring. Even the intellisense will work!

PS C:\windows\system32> Enter-PSSession -ConnectionUri $uri `

-Credential $cred `

-SessionOption $opts

[sivablogvm1.cloudapp.net]: PS C:\Users\Administrator\Documents> $env:COMPUTERNAME

SIVABLOGVM1

[sivablogvm1.cloudapp.net]: PS C:\Users\Administrator\Documents> get-process \*win\*

Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName

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

77 8 756 3420 20 0.06 440 wininit

113 7 1328 8308 63 0.11 468 winlogon

[sivablogvm1.cloudapp.net]: PS C:\Users\Administrator\Documents> Exit-PSSession

PS C:\windows\system32>

Explore & Enjoy!

/Siva