With the introduction of Azure Resource Manager (ARM), there has been a slight change in the way we remote into Azure Virtual Machines. If you've worked with Azure Service Management Virtual Machines, you would recollect that the default Powershell remoting (WinRM) endpoint was configured by default. With the ARM Virtual Machines this does not come by default. In Azure Service Management Virtual Machines had the WinRM pre-configured whereas in ARM Virtual Machines this is not the case. The scripts attached in this post will help you achieve the same so that you can remote into your ARM Virtual Machines. These scripts are targeted specifically for Azure Automation although, you can modify them as per your need in case you want to run it normally with Powershell. The scripts are in both the Azure Automation supported formats : Workflows as well as Powershell Scripts.

In a nutshell, there are 4 major steps required to configure and remote into your ARM Virtual Machine.

1. Configure your Virtual Machine's WinRM listener to listen to HTTPS requests
2. Open the firewall port for the incoming traffic
3. Add a Network Security Group Rule to allow inbound requests into the Virtual Machine
4. Get the public IP Address of the Virtual Machine and remote into it

There are four scripts attached with this post namely:

1. Connect-AzureARMVM (PowerShell Workflow) which performs steps 1,2,3 and 4 that were listed above
2. Remote-AzureARMVM (PowerShell Workflow) which uses the IP Address passed from Connect-AzureARMVM to remote into the VM and perform any operation
3. Connect-AzureARMVMPS (PowerShell Script) which performs steps 1,2,3 and 4 that were listed above
4. Remote-AzureARMVMPS (PowerShell Script) which uses the IP Address passed from Connect-AzureARMVM to remote into the VM and perform any operation

That being said, there are basically two scenarios that arise in this case when you are using this script.

* ARM Virtual Machine that has a standard storage account.
  1. The two attached scripts can be used as is in this scenario.
  2. The script uses Custom Script Extension to configure the WinRM Listener, enable the firewall rule in the VM and stores this configuration script in a Script container in the storage account where the VM is hosted so that the Custom Script Extension can pick it up from there.
* ARM Virtual Machine that has a premium storage account.
  1. To configure such Virtual Machines, we will have to perform the steps 1,2,3 and 4 manually by running the following code block in a powershell prompt on the VM

**PowerShell**

Edit|Remove

#POWERSHELL TO EXECUTE ON REMOTE SERVER BEGINS HERE

$DNSName = $env:COMPUTERNAME

#Ensure PS remoting is enabled, although this is enabled by default for Azure VMs

Enable-PSRemoting -Force

#Create rule in Windows Firewall

New-NetFirewallRule -Name "WinRM HTTPS" -DisplayName "WinRM HTTPS" -Enabled True -Profile "Any" -Action "Allow" -Direction "Inbound" -LocalPort 5986 -Protocol "TCP"

#Create Self Signed certificate and store thumbprint

$thumbprint = (New-SelfSignedCertificate -DnsName $DNSName -CertStoreLocation Cert:\LocalMachine\My).Thumbprint

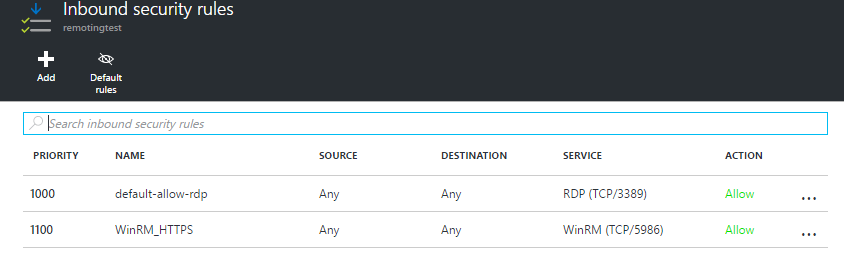
#Run WinRM configuration on command line. DNS name set to computer hostname, you may wish to use a FQDN

$cmd = "winrm create winrm/config/Listener?Address=\*+Transport=HTTPS @{Hostname=""$DNSName""; CertificateThumbprint=""$thumbprint""}"

cmd.exe /C $cmd

#POWERSHELL TO EXECUTE ON REMOTE SERVER ENDS HERE

* Add a Network Security Group Rule for the VM on port 5986



* You can then fetch the Public IP Address of the VM and remote into it in the same way as described earlier.

**PowerShell**

Edit|Remove

$IpAddress = .\Connect-AzureARMVMPS.ps1 -AzureSubscriptionId $AzureSubscriptionId -AzureOrgIdCredentialName $AzureOrgIdCredentialName -ResourceGroupName $ResourceGroupName -VMName $VMName

    Write-Output "The IP Address is $IpAddress. Attempting to remote into the VM.."

**if**($IpAddress -ne $null)

    {

            $sessionOptions = New-PSSessionOption -SkipCACheck -SkipCNCheck

            Invoke-Command -ComputerName $IpAddress -Credential $VMCredential -UseSSL -SessionOption $sessionOptions -ScriptBlock {

            #Code to be executed in the remote session goes here

            $hostname = hostname

            Write-Output "Hostname : $hostname"

            }

    }

Please Note: A pre-requisite to run these scripts is to have the Global Modules installed in your Azure Automation Account which at the time of writing this article was 1.0.3. You may have to manually download and upload the AzureRM.Network Module v(1.0.3) into your Automation Account if you dont have it already. Ensure that all your modules are of the same version.