Installing Aneka on Azure

1.VNET

…

…

2. Setup your machine

2.1 You should enable ICMP (PING) through the Windows Firewall
Open Control Panel>Windows Firewall>Advanced Settings>Inbound Rules
New Rule>Custom>All programs
Protocol type=ICMPv4
Allow the connection
Give a name and finish.
Repeat these steps for ICMPv6.

2.2 Create a powershell script as follows

Create a file and name it as WHATEVER.ps1
copy this content to the file:

```
Write-Host "Commands to enable network and file sharing"
Set-Service FDResPub -startuptype "manual"
Set-Service SSDPSRV -startuptype "manual"
Set-Service upnphost -startuptype "manual"
Start-Service -displayname "Function Discovery Resource Publication"
Start-Service -displayname "SSDP Discovery"
Start-Service -displayname "UPnP Device Host"
netsh advfirewall firewall set rule group="File and Printer Sharing"
new enable=Yes
netsh advfirewall firewall set rule group="network discovery" new
enable=yes
netsh advfirewall firewall set rule group="network discovery" new
enable=yes

Write-Host "Commands to open Aneka ports"
netsh advfirewall firewall add rule name="Aneka Daemon" dir=in
action=allow protocol=TCP localport=9000
netsh advfirewall firewall add rule name="Aneka Daemon" dir=out
action=allow protocol=TCP localport=9000
netsh advfirewall firewall add rule name="Aneka Container" dir=in
action=allow protocol=TCP localport=9090
netsh advfirewall firewall add rule name="Aneka Container" dir=out
action=allow protocol=TCP localport=9090
netsh advfirewall firewall add rule name="Aneka storage" dir=in
action=allow protocol=TCP localport=9091
```

```
netsh advfirewall firewall add rule name="Aneka storage" dir=out
action=allow protocol=TCP localport=9091
netsh advfirewall firewall add rule name="Aneka Adi" dir=in
action=allow protocol=TCP localport=9092
netsh advfirewall firewall add rule name="Aneka Adi" dir=out
action=allow protocol=TCP localport=9092

Write-Host "Commands to open PowerShell ports (This is needed in to
manage remote machines via PowerShell)"
netsh advfirewall firewall add rule name="Aneka PS1" dir=in
action=allow protocol=TCP localport=5986
netsh advfirewall firewall add rule name="Aneka PS1" dir=out
action=allow protocol=TCP localport=5986
netsh advfirewall firewall add rule name="Aneka PS2" dir=in
action=allow protocol=TCP localport=5985
netsh advfirewall firewall add rule name="Aneka PS2" dir=out
action=allow protocol=TCP localport=5985

Write-Host "Commands to open File Sharing ports"
netsh advfirewall firewall add rule name="SMB TCP" dir=in
action=allow protocol=TCP localport=135-139
netsh advfirewall firewall add rule name="SMB TCP" dir=out
action=allow protocol=TCP localport=135-139
netsh advfirewall firewall add rule name="SMB UDP" dir=in
action=allow protocol=UDP localport=135-139
netsh advfirewall firewall add rule name="SMB UDP" dir=out
action=allow protocol=UDP localport=135-139
netsh advfirewall firewall add rule name="DIRECT SMB TCP" dir=in
action=allow protocol=TCP localport=445
netsh advfirewall firewall add rule name="DIRECT SMB TCP" dir=out
action=allow protocol=TCP localport=445
netsh advfirewall firewall add rule name="DIRECT SMB UDP" dir=in
action=allow protocol=UDP localport=445
netsh advfirewall firewall add rule name="DIRECT SMB UDP" dir=out
action=allow protocol=UDP localport=445

Write-Host "command to install .Net Framework"
Install-WindowsFeature Net-Framework-Core -source
\\network\share\sxs?

Write-Host "command to install .Net Framework in Windows server 2012
R2"
wusa.exe /uninstall /kb:2966828 /quiet /log /norestart
```

```
$SourcePath = "%Temp%\sxs"
Install-WindowsFeature NET-Framework-Core –Source $SourcePath
```

Save the file and return to the Powershell window.

2.3 Run the script on the Powershell

In order to run the script, the most common method is by calling it:
& “WHATEVER.ps1”

You might get an error like this

```
WHATEVER.ps1 cannot be loaded because running scripts is disabled on
this system.  …
```

You should change the execution policy, we will need to reopen Powershell as an Administrator (the command will fail otherwise) and run the following command:

```
Set-ExecutionPolicy RemoteSigned
```

Then run WHATEVER.ps1.
Everything should be ok.

2.4

Capture a Windows virtual machine image in the Resource Manager deployment model

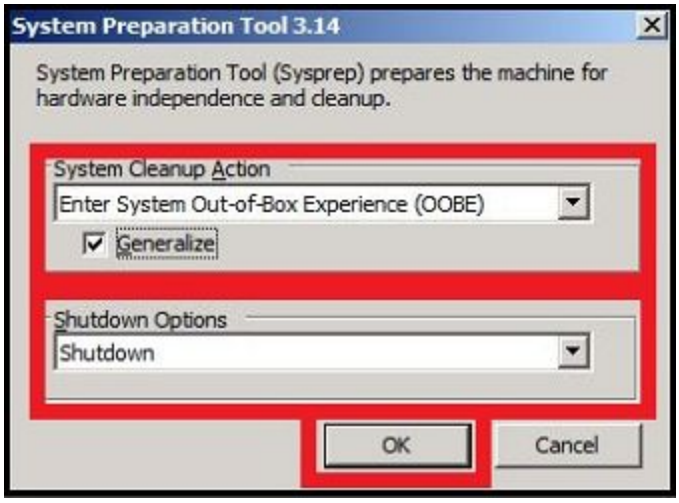
Prepare the VM for image capture

This section shows you how to generalize your Windows virtual machine. This removes all your personal account information, among other things. You will typically want to do this when you want to use this VM image to quickly deploy similar virtual machines.

Note that the virtual machine cannot be logged in via RDP once it is generalized, since the process removes all user accounts.

Sign in to your Windows virtual machine. In the Azure portal, navigate through Browse > Virtual machines > Your Windows virtual machine >Connect.

1. Open a Command Prompt window as an administrator.
2. Change the directory to `%windir%\system32\sysprep`, and then run sysprep.exe.
3. In the System Preparation Tool dialog box, do the following:
    - In System Cleanup Action, select Enter System Out-of-Box Experience (OOBE) and make sure that Generalize is checked. For more information about using Sysprep, see How to Use Sysprep: An Introduction.
    - In Shutdown Options, select Shutdown.
    - Click OK.

4. 

Sysprep shuts down the virtual machine. Its status changes to Stopped in the Azure portal.

Capture the VM

You can capture the generalized Windows VM by using either Azure PowerShell or the new Azure Resource Manager Explorer tool. This section will show you the steps for both.

Using PowerShell

Use ca

You have to install Azure PowerShell before proceeding this part.

Open PowerShell and sign in to your Azure account.

```
Login-AzureRmAccount
```

This command will open a pop-up window to enter your Azure credentials.

If the subscription ID that is selected by default is different from the one you want to work in, use either of the following to set the right subscription.

```
Set-AzureRmContext -SubscriptionId "xxxx-xxxx-xxxx-xxxx"
```

You can find the subscriptions that your Azure account has by using the command `Get-AzureRmSubscription`.

Now you will need to deallocate the resources that are used by this virtual machine by using this commmand.

```
Stop-AzureRmVM -ResourceGroupName YourResourceGroup -Name YourWindowsVM
```

You will see that the *Status* for the VM on the Azure portal has changed from Stopped to Stopped (deallocated).

You can also find out the status of your virtual machine in PowerShell by using:

```
$vm = Get-AzureRmVM -ResourceGroupName YourResourceGroup -Name YourWindowsVM
-status
```

```
$vm.Statuses
```

The DisplayStatus field corresponds to the Status shown in the Azure portal.

Next, you need to set the status of the virtual machine to Generalized. Note that you will need to do this because the generalization step above (`sysprep`) does not do it in a way that Azure can understand.

```
Set-AzureRmVm -ResourceGroupName YourResourceGroup -Name YourWindowsVM
-Generalized
```

The generalized state as set above will not be shown on the portal. However, you can verify it by using the Get-AzureRmVM command as shown in the tip above.

Capture the virtual machine image to a destination storage container by using this command.

```
Save-AzureRmVMImage -ResourceGroupName YourResourceGroup -VMName YourWindowsVM
-DestinationContainerName YourImagesContainer -VHDNamePrefix YourTemplatePrefix
-Path Yourlocalfilepath\Filename.json
```

```
Example:
```

```
PS C:\windows\system32> Save-AzureRmVMImage -ResourceGroupName Adel_Aneka_Test
-VMName anekatest2 -DestinationContainerName anekaimagescontainer -VHDNamePrefix
aneka -Path C:\aneka.json
```

The `-Path` variable is optional. You can use it to save the JSON template locally. The `-DestinationContainerName` variable is the name of the container that you want to hold your images in. The URL of the image that is stored will be similar to `https://YourStorageAccountName.blob.core.windows.net/system/Microsoft.Compute/Images/YourImagesContainer/YourTemplatePrefix-osDisk.xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxx`

`xxxxx.vhd`. It will be created in the same storage account as that of the original virtual machine.

To find the location of your image, open the local JSON file template. Go to the resources > storageProfile > osDisk > image> uri section for the complete path of your image. As of now, there is no easy way to check these images on the portal, since the *system* container in the storage account is hidden. For this reason, although the `-Path` variable is optional, you definitely want to use it to save the template locally and to easily find out the image URL.

Create network resources

Use the following sample PowerShell script to set up a virtual network and NIC for your new VM. Use values for the variables (represented by the $ sign) as appropriate to your application.

Copy

```
$pip = New-AzureRmPublicIpAddress -Name $pipName -ResourceGroupName $rgName
-Location $location -AllocationMethod Dynamic

$subnetconfig = New-AzureRmVirtualNetworkSubnetConfig -Name $subnet1Name
-AddressPrefix $vnetSubnetAddressPrefix

$vnet = New-AzureRmVirtualNetwork -Name $vnetName -ResourceGroupName $rgName
-Location $location -AddressPrefix $vnetAddressPrefix -Subnet $subnetconfig

$nic = New-AzureRmNetworkInterface -Name $nicname -ResourceGroupName $rgName
-Location $location -SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $pip.Id
```

Create a new VM from the Captured image

The following PowerShell script shows how to set up the virtual machine configurations and use the captured VM image as the source for the new installation.

```
#Enter a new user name and password in the pop-up for the following
$cred = Get-Credential

#Get the storage account where the captured image is stored
$storageAcc = Get-AzureRmStorageAccount -ResourceGroupName $rgName -AccountName
$storageAccName
```

If you do not know your storage account open the local JSON file template. Go to the resources > storageProfile > osDisk > image> uri section for the complete path of your image. You can find it here in the URI:

`https://>>>>>YourStorageAccountName<<<<<.blob.core. … xxxxxxxxx.vhd`.

```
#Set the VM name and size
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize "Standard_A2"

#Set the Windows operating system configuration and add the NIC
$vm = Set-AzureRmVMOperatingSystem -VM $vmConfig -Windows -ComputerName
$computerName -Credential $cred -ProvisionVMAgent -EnableAutoUpdate

$vm = Add-AzureRmVMNetworkInterface -VM $vm -Id $nic.Id

#Create the OS disk URI
$osDiskUri = '{0}vhds/{1}{2}.vhd' -f
$storageAcc.PrimaryEndpoints.Blob.ToString(), $vmName.ToLower(), $osDiskName

#Configure the OS disk to be created from image (-CreateOption fromImage) and
give the URL of the captured image VHD for the -SourceImageUri parameter.
#We found this URL in the local JSON template in the previous sections.
```

```
$vm = Set-AzureRmVMOSDisk -VM $vm -Name $osDiskName -VhdUri $osDiskUri
-CreateOption fromImage -SourceImageUri $urlOfCapturedImageVhd -Windows

#Create the new VM
New-AzureRmVM -ResourceGroupName $rgName -Location $location -VM $vm
```

You should see the newly created VM in either the Azure portal under Browse >Virtual machines, OR by using the following PowerShell commands:

Copy

```
$vmList = Get-AzureRmVM -ResourceGroupName $rgName
$vmList.Name
```

$pip = New-AzureRmPublicIpAddress -Name anekapip -ResourceGroupName Adel_Aneka_Test -Location "Australia Southeast" -AllocationMethod Dynamic
#Get the virtual network information you already created.
$vnet = Get-AzureRmVirtualNetwork -Name Aneka-VNET-SITE -ResourceGroupName Adel_Aneka_Test
#Get the subnet information for virtual network
$subnetconfig = Get-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet
#Create a new nic address for you machine

```
$nic = New-AzureRmNetworkInterface -Name anekanic -ResourceGroupName
Adel_Aneka_Test -Location "Australia Southeast" -SubnetId $vnet.Subnets[0].Id
-PublicIpAddressId $pip.Id
#Enter a new user name and password in the pop-up for the following
$cred = Get-Credential

#Get the storage account where the captured image is stored
#If you do not know your storage account open the local JSON file template. Go to the resources > storageProfile > osDisk > image> uri section for the complete path of your image. You can find it here in the URI:
```

```
https://>>>>>YourStorageAccountName<<<<<.blob.core. … xxxxxxxxx.vhd.
$storageAcc = Get-AzureRmStorageAccount -ResourceGroupName $rgName -AccountName
$storageAccName
#Set the VM name and size
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize "Standard_A2"
#Set the Windows operating system configuration and add the NIC
$vm = Set-AzureRmVMOperatingSystem -VM $vmConfig -Windows -ComputerName
$computerName -Credential $cred -ProvisionVMAgent -EnableAutoUpdate

$vm = Add-AzureRmVMNetworkInterface -VM $vm -Id $nic.Id

#Create the OS disk URI
$osDiskUri = '{0}vhds/{1}{2}.vhd' -f
$storageAcc.PrimaryEndpoints.Blob.ToString(), $vmName.ToLower(), $osDiskName

#Configure the OS disk to be created from image (-CreateOption fromImage) and
give the URL of the captured image VHD for the -SourceImageUri parameter.
#We found this URL in the local JSON template in the previous sections.
$vm = Set-AzureRmVMOSDisk -VM $vm -Name $osDiskName -VhdUri $osDiskUri
-CreateOption fromImage -SourceImageUri $urlOfCapturedImageVhd -Windows

#Create the new VM
New-AzureRmVM -ResourceGroupName $rgName -Location $location -VM $vm
```

Calling the Azure REST API

Solution: you can either create a Service Principal account and give it just the set of permissions that it needs or use a Management Certificate, which has full power over the subscription.

Azure supports two different REST APIs:

- the old one, known as RDFE
- the new one, known as ARM (Azure Resource Manager)

Many people are still using RDFE, as it's been the only API for years. But ARM is what the cool kids are doing now, and it is worth taking the time to switch over. Besides being a nicer REST API, it supports new key concept like Resource Groups, Deployment Templates and RBAC (Role Based Access Control).

Calling the Azure ARM API using a service principal

1. The first thing you'll need to do is create an AAD application. To do this, go to the Current Azure portal (`https://manage.windowsazure.com/`). For this purpose you should use global admin user.
2. Choose Active Directory in the left pane
3. Click on default directory.
4. Then click on APPLICATIONS tab. It'll probably say that you don't have any. Click to add one.
5. In the next dialog, choose Add an application that my organization is developing.
6. Now you'll need to give an answer that probably won't make more sense. Our goal is to automate Azure from our client, yet here you have to tell it to create a Web app. Just go along with it. I warn you some steps would not be too logical!
7. Give a name to your application, e.g., **aneka**

8. Now it's asking you for two URLs. In our scenario, using URLs here doesn't really make sense (it does for other scenario). But you'll want to enter some recognizable URL, as we'll need it later during role assignment. e.g. I use **`http://aneka`**, which is bogus as a URL, but is recognizable to represent my app (this will get cleaner in the future). Copy the same for APP ID URI.
9. Now you now have an AAD application! In there, click on the CONFIGURE tab
10. Find the Client ID and save it. This will basically be your username: e.g., **`Username: 90158e70-3d93-4237-a225-52df16ca8f9b`**
11. Now go to the Keys section, click on the drop down, and pick 1 or 2 years and hit the save button at the bottom, it will display your key, which is basically your Service Principal account password. Save it and store it in a secure place (like a password manager). You will never see it again in the portal! E.g., **`Password: 5S7Re1A9zINGIeCsPY3mvuW11fNxX4PQc2mAJ/tGQ4Q=`**
12. One last thing you need to do is get your tenant ID. The way to do this is a bit harder than it should be. Click on the View Endpoints button in the bottom bar. It will show you lots of URLs. Click copy on the first one (any of them will do too). I will look like this:
    `https://login.microsoftonline.com/b33823ea-6faf-4f0b-967f-de5f4933f22b/federationmetadata/2007-06/federationmetadata.xml`
    The GUID in there is your tenant ID, which you'll need later.
    `Tenant ID :b33823ea-6faf-4f0b-967f-de5f4933f22b`

It was complex to get here but the summary is that you now have a Service Principal account with a username and a password. And we also have our tenant ID:

`Client id : 90158e70-3d93-4237-a225-52df16ca8f9b`
`Password: 5S7Re1A9zINGIeCsPY3mvuW11fNxX4PQc2mAJ/tGQ4Q=`
`Tenant ID :b33823ea-6faf-4f0b-967f-de5f4933f22b`

Assigning roles to your Service Principal

You have a Service Principal account, but right now it's not allowed to do anything. You'll need to use Azure PowerShell to do this (until the Preview Portal adds support for it).

Here, you'll want to log in as your Microsoft identity in order to grant roles to your Service Principal identity (conceptually: you're the boss, and you set permissions for your 'employee').

```
Login-AzureRmAccount # This will pop up a login dialog
```

Now, you can assign roles to your Service Principal. e.g. let's give it access to one of the resource groups in our subscription. You can use either App ID Uri or Client ID as the value for the -ServicePrincipalNameparameter.

```
New-AzureRmRoleAssignment -ServicePrincipalName http://aneka
-RoleDefinitionName Contributor -Scope
/subscriptions/deb6d126-5cb8-470f-b967-6540279d470e/resourceGroups/Adel_Aneka_Test
```

Or if you want it to have access to the whole subscription, just leave out the Scope:

```
Select-AzureRmSubscription -SubscriptionId <subscription-id>
New-AzureRmRoleAssignment -ServicePrincipalName http://aneka
-RoleDefinitionName Contributor
```

If you run Get-AzureRmRoleAssignment, you should see the assignment.

Using your Service Principal account

We're going to check if the Service Principal identity works ok.

1. Open a new PowerShell window.
2. Now, let's get our Service Principal creds into a PSCredential object, as described below:

   ```
   $secpasswd = ConvertTo-SecureString "5S7Re1A9zINGIeCsPY3mvuW11fNxX4PQc2mAJ/tGQ4Q=" -AsPlainText -Force
   $mycreds = New-Object System.Management.Automation.PSCredential ("90158e70-3d93-4237-a225-52df16ca8f9b", $secpasswd)
   ```
3. We are now able to add the Service Principal account, e.g.,

   ```
   Login-AzureRmAccount -ServicePrincipal -Tenant b33823ea-6faf-4f0b-967f-de5f4933f22b  -Credential $mycreds
   ```
4. Run the following command to see all resources this principal has access to it.

   ```
   Get-AzureRmResource
   ```

Calling the Azure RFDE API using a Management Certificate

Find Subscription name and subscription id:

1. Login in the azure portal (https://manage.windowsazure.com)
2. Go to the SETTINGS section. This is the last option in the menu that is in left hand of the Microsoft azure console. Then found the subscription section and the column SUBSCRIPTION is the subscription name and the following column is SUBSCRIPTION ID.

Install Certificate:

3. Access the following link -> https://manage.windowsazure.com/publishsettings/Index?client=vs&SchemaVersion=1.0 that automatically saves a xxx.publishsettings file.
5. Login with your account

6. Save the file in the following path -> C:\Program Files (x86)\Manjrasoft\Aneka.4.0\Runtime\Certificate
7. You need Microsoft Azure Powershell Command-line to do run the following commands. If you do not have one installed download and install it from: http://azure.microsoft.com/en-us/downloads/#cmd-line-tools
   Open Powershell and execute the followings commands:

```
cd "C:\Program Files
(x86)\Manjrasoft\Aneka.4.0\Runtime\Certificate"
Set-ExecutionPolicy Unrestricted
Import-AzurePublishSettingsFile $filename
Set-ExecutionPolicy Restricted
```

   $filename is the xxx.publishsettings that you downloaded in the previous step.
8. Find the certificate name in the azure portal. For this you need to log in https://manage.windowsazure.com and go to the SETTINGS section. This is the last option in the menu that is in left hand of the Microsoft azure portal. Then in the top select the option MANAGEMENT CERTIFICATES. Find the one that have the same name of the file downloaded (xxx.publishsettings) and in the option THUMBPRINT you can find the thumbprint to configure the azure cloud connection.

Install the certificate as a trusted certificate:

9. Windows key+R -> certmgr.msc -> Personal->Certificate.
10. Double click over the certificate associated to the file downloaded.
11. Then open the tab Details ->copy file to. Could be copy in "C:\Program Files (x86)\Manjrasoft\Aneka.4.0\Runtime\Certificate"
12. After copying the file, double click on the file then Install certificate.
13. Select the option “Place all the certificates in the following store”. Browse and select Trusted Root Certification Authorities. Next and finish and yes.

14. If the machines will be acquired via auto provision, so it will be needed to authenticate the daemon service with the current user, and in this way it could find the certificate. Otherwise this error can appear in the container log when the container tried to obtain the new machine:
    System.Exception: Provision failed: System.Exception: Uri: The client certificate was not found. Please review thumbprint:
15. To log the daemon: -windows+R -> services.msc
16. -Search for the service Aneka::Daemon - [Port: 9000]
17. -Sight click properties
18. -Select the tab Log on.
19. -Check this account
20. -Write the user and password of the current user
21. -Restart the service
22. Add existent machine
23. -use chrome to download the certificate
24. -save the certificate in Trusted Root Certification Authorities

Step 1: Create a Key Vault

Learn how to create a Key Vault using PowerShell at the links below

NOTE: *You'll need to set the "EnabledForDeployment" switch to enable the Microsoft.Compute resource provider to download your secret from the Key Vault*

- PowerShell - https://azure.microsoft.com/en-us/documentation/articles/key-vault-get-started/#vault

Step 2: Create a self-signed certificate

The self-signed certificate can be generated by following this guide - https://msdn.microsoft.com/en-us/library/ff699202.aspx

Step 3: Upload your self-signed certificate to the Key Vault

Before uploading the certificate to the Key Vault created in step 1, it needs to converted into a format the Microsoft.Compute resource provider will understand. The below PowerShell script will allow you do that

```
$fileName = "<Path to the .pfx file>"
$fileContentBytes = get-content $fileName -Encoding Byte
$fileContentEncoded = [System.Convert]::ToBase64String($fileContentBytes)

$jsonObject = @"
{
    "data": "$filecontentencoded",
    "dataType" :"pfx",
    "password": "<password>"
}
"@

$jsonObjectBytes = [System.Text.Encoding]::UTF8.GetBytes($jsonObject)
$jsonEncoded = [System.Convert]::ToBase64String($jsonObjectBytes)

$secret = ConvertTo-SecureString -String $jsonEncoded -AsPlainText –Force
```

```
Set-AzureKeyVaultSecret -VaultName "<vault name>" -Name "<secret name>" -SecretValue $secret
```

Step 4: Get the URL for your self-signed certificate in the Key Vault

The Microsoft.Compute resource provider needs a URL to the secret inside the Key Vault while provisioning the VM. This enables the Microsoft.Compute resource provider to download the secret and create the equivalent certificate on the VM.

Templates

You can get the link to the URL in the template using the below code

```
"certificateUrl": "[reference(resourceId(resourceGroup().name,
'Microsoft.KeyVault/vaults/secrets', '<vault-name>', '<secret-name>'),
'2015-06-01').secretUriWithVersion]"
```

PowerShell

You can get this URL using the below PowerShell command

```
$secretURL = (Get-AzureKeyVaultSecret -VaultName "<vault name>" -Name "<secret name>").Id
```

Step 5: Reference your self-signed certificates URL while creating a VM

ARM Templates

While creating a VM through templates, the certificate gets referenced in the secrets section and the winRM section as below

```
"osProfile": {
      ...
      "secrets": [
        {
          "sourceVault": {
            "id": "<resource id of the Key Vault containing the secret>"
          },
          "vaultCertificates": [
            {
              "certificateUrl": "<URL for the certificate you got in Step 4>",
              "certificateStore": "<Name of the certificate store on the VM>"
            }
          ]
        }
      ],
      "windowsConfiguration": {
        ...
        "winRM": {
          "listeners": [
            {
              "protocol": "http"
            },
            {
              "protocol": "https",
```

```
            "certificateUrl": "<URL for the certificate you got in Step 4>"
          }
        ]
      },
      ...
    }
  },
```

A sample template for the above can be found here -

https://azure.microsoft.com/en-us/documentation/templates/201-winrm-windows-vm

Source code for this template can be found in Github -

https://github.com/Azure/azure-quickstart-templates/tree/master/201-winrm-windows-vm

PowerShell

```
$vm = New-AzureRmVMConfig -VMName "<VM name>" -VMSize "<VM Size>"
$credential = Get-Credential
$secretURL = (Get-AzureKeyVaultSecret -VaultName "<vault name>" -Name "<secret name>").Id
$vm = Set-AzureRmVMOperatingSystem -VM $vm  -Windows -ComputerName "<Computer Name>"
-Credential $credential -WinRMHttps -WinRMCertificateUrl $secretURL
$sourceVaultId = (Get-AzureRmKeyVault -ResourceGroupName "<Resource Group name>" -VaultName
"<Vault Name>").ResourceId
$CertificateStore = "My"
$vm = Add-AzureRmVMSecret -VM $vm -SourceVaultId $sourceVaultId -CertificateStore
$CertificateStore -CertificateUrl $secretURL
```

On the local computer to make the powershell session

```
$username = "aneka"
$password = "aneka1234$" | ConvertTo-SecureString -AsPlainText -Force
$cred = new-object -typename System.Management.Automation.PSCredential
```

```
-argumentlist $username,$password
Enter-PSSession -ComputerName 192.168.1.8 -UseSSL -Credential $cred
-SessionOption (New-PSSessionOption -SkipCACheck -SkipCNCheck
-SkipRevocationCheck)
```