### Click on following links tp go directly to scripts

### [Create a Resource Group](#RG)

### [Creating an Azure Virtual Machine](#VM)

### [Create an IIS VM with PowerShell](#VM_WITH_IIS)

### [Script to upload a VHD to Azure and create a new VM](#UPLOAD_VHD_CREATE_VM)

### [Create a virtual machine using an existing managed OS disk](#VM_WITH_MANAGED_OS_DISK)

**Windows**

PowerShell is included with Windows, however there may be an update available for your machine. The Azure support we are going to use requires PowerShell with a major version of 5.0 or higher. You can check the version you have installed through the following steps:

1. Open the **Start** menu and type **Windows PowerShell**. There may be multiple shortcut links available:
2. Select the Windows PowerShell icon.
3. Type the following command to determine the version of PowerShell installed.

$PSVersionTable.PSVersion

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> $PSVersionTable.PSVersion

Major Minor Build Revision

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

5 1 18362 752

If the major version number is lower than 5.0, follow these instructions for [upgrading existing Windows PowerShell](https://docs.microsoft.com/en-us/powershell/scripting/install/installing-windows-powershell#upgrading-existing-windows-powershell).

Now we have setup our local machine to support PowerShell.

Now, we will talk about additional commands we can add including the Azure module.

PowerShell lets you write commands and execute them immediately. This is known as **interactive mode**.

A PowerShell command is called a **cmdlet** (pronounced "command-let"). A cmdlet is a command that manipulates a single feature. The term **cmdlet** is intended to imply "small command".

Cmdlets are shipped in modules. A PowerShell Module is a DLL that includes the code to process each available cmdlet. You load cmdlets into PowerShell by loading the module they are contained in. You can get a list of loaded modules using the Get-Module command: Cmdlets are shipped in modules. A PowerShell Module is a DLL that includes the code to process each available cmdlet. You load cmdlets into PowerShell by loading the module they are contained in. You can get a list of loaded modules using the Get-Module command:

PS C:\WINDOWS\system32> Get-Module

ModuleType Version Name ExportedCommands

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

Manifest 3.1.0.0 Microsoft.PowerShell.Utility {Add-Member, Add-Type, Clear-Variable, Compare-Object...}

Script 2.0.0 PSReadline {Get-PSReadLineKeyHandler, Get-PSReadLineOption, Remove-PS...

## What is the Az module?

**Az** is the formal name for the Azure PowerShell module containing cmdlets to work with Azure features. It contains hundreds of cmdlets that let you control nearly every aspect of every Azure resource. You can work with resource groups, storage, virtual machines, Azure Active Directory, containers, machine learning, and so on. This module is an open source component [available on GitHub](https://github.com/Azure/azure-powershell).

You may have seen or used Azure PowerShell commands that used a -AzureRM format. In October 2018 we announced the replacement of the **AzureRM** module with the **Az** module. This new module has several features, notably a shortened cmdlet noun prefix of -Az instead of -AzureRM. The **Az** module ships with backwards compatibility with the **AzureRM** module so the -AzureRM cmdlet format will work, but you should transition to the **Az** module and use the -Az commands going forward.

### Install the Az module

The Az module is available from a global repository called the PowerShell Gallery. You can install the module onto your local machine through the Install-Module command. You need an elevated PowerShell shell to install modules from the PowerShell Gallery.

To install the latest Azure PowerShell module, run the following commands:

1. Open the **Start** menu and type **Windows PowerShell**.
2. Right-click the **Windows PowerShell** icon and select **Run as administrator**.
3. In the **User Account Control** dialog, select **Yes**.
4. Type the following command, and then press Enter:

PowerShellCopy

Install-Module -Name Az -AllowClobber

## create a resource group with Azure PowerShell

There are four steps we need to perform:

1. Import the Azure cmdlets.
2. Connect to your Azure subscription.
3. Create the resource group.
4. Verify that creation was successful

Import-Module Az

Connect-AzAccount

Select-AzSubscription -Subscription "Visual Studio Enterprise"

Get-AzResourceGroup

Get-AzResourceGroup | Format-Table

New-AzResourceGroup -Name <name> -Location <location>

Get-AzResource

Get-AzResource | ft

Get-AzResource -ResourceGroupName ExerciseResources

### Creating an Azure Virtual Machine

Azure PowerShell provides the New-AzVm cmdlet to create a virtual machine. The cmdlet has many parameters to let it handle the large number of VM configuration settings. Most of the parameters have reasonable default values so we only need to specify five things:

* **ResourceGroupName**: The resource group into which the new VM will be placed.
* **Name**: The name of the VM in Azure.
* **Location**: Geographic location where the VM will be provisioned.
* **Credential**: An object containing the username and password for the VM admin account. We will use the Get-Credential cmdlet. This cmdlet will prompt for a username and password and package it into a credential object.
* **Image**: The operating system image to use for the VM. This is often a Linux distribution, or Windows Server.

# Variables for common values

$resourceGroup = "myResourceGroup"

$location = "westeurope"

$vmName = "myVM"

# Create user object

$cred = Get-Credential -Message "Enter a username and password for the virtual machine."

# Create a resource group

New-AzResourceGroup -Name $resourceGroup -Location $location

# Create a virtual machine

New-AzVM `

-ResourceGroupName $resourceGroup `

-Name $vmName `

-Location $location `

-Image "Win2016Datacenter" `

-VirtualNetworkName "myVnet" `

-SubnetName "mySubnet" `

-SecurityGroupName "myNetworkSecurityGroup" `

-PublicIpAddressName "myPublicIp" `

-Credential $cred `

-OpenPorts 3389

Remove a Resource Group

Remove-AzResourceGroup -Name myResourceGroup

# Create an IIS VM with PowerShell

# Variables for common values

$resourceGroup = "myResourceGroup"

$location = "westeurope"

$vmName = "myVM"

# Create user object

$cred = Get-Credential -Message "Enter a username and password for the virtual machine."

# Create a resource group

New-AzResourceGroup -Name $resourceGroup -Location $location

# Create a virtual machine

New-AzVM `

-ResourceGroupName $resourceGroup `

-Name $vmName `

-Location $location `

-ImageName "Win2016Datacenter" `

-VirtualNetworkName "myVnet" `

-SubnetName "mySubnet" `

-SecurityGroupName "myNetworkSecurityGroup" `

-PublicIpAddressName "myPublicIp" `

-Credential $cred `

-OpenPorts 80

# Install IIS

$PublicSettings = '{"commandToExecute":"powershell Add-WindowsFeature Web-Server"}'

Set-AzVMExtension -ExtensionName "IIS" -ResourceGroupName $resourceGroup -VMName $vmName `

-Publisher "Microsoft.Compute" -ExtensionType "CustomScriptExtension" -TypeHandlerVersion 1.4 `

-SettingString $PublicSettings -Location $location

# Script to upload a VHD to Azure and create a new VM

This script takes a local .vhd file from a generalized VM and uploads it to Azure, creates a Managed Disk image and uses the to create a new VM.

# Provide values for the variables

$resourceGroup = 'myResourceGroup'

$location = 'EastUS'

$storageaccount = 'mystorageaccount'

$storageType = 'Standard\_LRS'

$containername = 'mycontainer'

$localPath = 'C:\Users\Public\Documents\Hyper-V\VHDs\generalized.vhd'

$vmName = 'myVM'

$imageName = 'myImage'

$vhdName = 'myUploadedVhd.vhd'

$diskSizeGB = '128'

$subnetName = 'mySubnet'

$vnetName = 'myVnet'

$ipName = 'myPip'

$nicName = 'myNic'

$nsgName = 'myNsg'

$ruleName = 'myRdpRule'

$computerName = 'myComputerName'

$vmSize = 'Standard\_DS1\_v2'

# Get the username and password to be used for the administrators account on the VM.

# This is used when connecting to the VM using RDP.

$cred = Get-Credential

# Upload the VHD

New-AzResourceGroup -Name $resourceGroup -Location $location

New-AzStorageAccount -ResourceGroupName $resourceGroup -Name $storageAccount -Location $location `

-SkuName $storageType -Kind "Storage"

$urlOfUploadedImageVhd = ('https://' + $storageaccount + '.blob.core.windows.net/' + $containername + '/' + $vhdName)

Add-AzVhd -ResourceGroupName $resourceGroup -Destination $urlOfUploadedImageVhd `

-LocalFilePath $localPath

# Note: Uploading the VHD may take awhile!

# Create a managed image from the uploaded VHD

$imageConfig = New-AzImageConfig -Location $location

$imageConfig = Set-AzImageOsDisk -Image $imageConfig -OsType Windows -OsState Generalized `

-BlobUri $urlOfUploadedImageVhd

$image = New-AzImage -ImageName $imageName -ResourceGroupName $resourceGroup -Image $imageConfig

# Create the networking resources

$singleSubnet = New-AzVirtualNetworkSubnetConfig -Name $subnetName -AddressPrefix 10.0.0.0/24

$vnet = New-AzVirtualNetwork -Name $vnetName -ResourceGroupName $resourceGroup -Location $location `

-AddressPrefix 10.0.0.0/16 -Subnet $singleSubnet

$pip = New-AzPublicIpAddress -Name $ipName -ResourceGroupName $resourceGroup -Location $location `

-AllocationMethod Dynamic

$rdpRule = New-AzNetworkSecurityRuleConfig -Name $ruleName -Description 'Allow RDP' -Access Allow `

-Protocol Tcp -Direction Inbound -Priority 110 -SourceAddressPrefix Internet -SourcePortRange \* `

-DestinationAddressPrefix \* -DestinationPortRange 3389

$nsg = New-AzNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location `

-Name $nsgName -SecurityRules $rdpRule

$nic = New-AzNetworkInterface -Name $nicName -ResourceGroupName $resourceGroup -Location $location `

-SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $pip.Id -NetworkSecurityGroupId $nsg.Id

$vnet = Get-AzVirtualNetwork -ResourceGroupName $resourceGroup -Name $vnetName

# Start building the VM configuration

$vm = New-AzVMConfig -VMName $vmName -VMSize $vmSize

# Set the VM image as source image for the new VM

$vm = Set-AzVMSourceImage -VM $vm -Id $image.Id

# Finish the VM configuration and add the NIC.

$vm = Set-AzVMOSDisk -VM $vm  -DiskSizeInGB $diskSizeGB -CreateOption FromImage -Caching ReadWrite

$vm = Set-AzVMOperatingSystem -VM $vm -Windows -ComputerName $computerName -Credential $cred `

-ProvisionVMAgent -EnableAutoUpdate

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

# Create the VM

New-AzVM -VM $vm -ResourceGroupName $resourceGroup -Location $location

# Verify that the VM was created

$vmList = Get-AzVM -ResourceGroupName $resourceGroup

$vmList.Name

# Create a virtual machine using an existing managed OS disk with PowerShell (Windows)

This script creates a virtual machine by attaching an existing managed disk as OS disk. Use this script in preceding scenarios:

* Create a VM from an existing managed OS disk that was copied from a managed disk in different subscription
* Create a VM from an existing managed disk that was created from a specialized VHD file
* Create a VM from an existing managed OS disk that was created from a snapshot

#Provide the subscription Id

$subscriptionId = 'yourSubscriptionId'

#Provide the name of your resource group

$resourceGroupName ='yourResourceGroupName'

#Provide the name of the snapshot that will be used to create OS disk

$snapshotName = 'yourSnapshotName'

#Provide the name of the OS disk that will be created using the snapshot

$osDiskName = 'yourOSDiskName'

#Provide the name of an existing virtual network where virtual machine will be created

$virtualNetworkName = 'yourVNETName'

#Provide the name of the virtual machine

$virtualMachineName = 'yourVMName'

#Provide the size of the virtual machine

#e.g. Standard\_DS3

#Get all the vm sizes in a region using below script:

#e.g. Get-AzVMSize -Location westus

$virtualMachineSize = 'Standard\_DS3'

#Set the context to the subscription Id where Managed Disk will be created

Select-AzSubscription -SubscriptionId $SubscriptionId

$snapshot = Get-AzSnapshot -ResourceGroupName $resourceGroupName -SnapshotName $snapshotName

$diskConfig = New-AzDiskConfig -Location $snapshot.Location -SourceResourceId $snapshot.Id -CreateOption Copy

$disk = New-AzDisk -Disk $diskConfig -ResourceGroupName $resourceGroupName -DiskName $osDiskName

#Initialize virtual machine configuration

$VirtualMachine = New-AzVMConfig -VMName $virtualMachineName -VMSize $virtualMachineSize

#Use the Managed Disk Resource Id to attach it to the virtual machine. Please change the OS type to linux if OS disk has linux OS

$VirtualMachine = Set-AzVMOSDisk -VM $VirtualMachine -ManagedDiskId $disk.Id -CreateOption Attach -Windows

#Create a public IP for the VM

$publicIp = New-AzPublicIpAddress -Name ($VirtualMachineName.ToLower()+'\_ip') -ResourceGroupName $resourceGroupName -Location $snapshot.Location -AllocationMethod Dynamic

#Get the virtual network where virtual machine will be hosted

$vnet = Get-AzVirtualNetwork -Name $virtualNetworkName -ResourceGroupName $resourceGroupName

# Create NIC in the first subnet of the virtual network

$nic = New-AzNetworkInterface -Name ($VirtualMachineName.ToLower()+'\_nic') -ResourceGroupName $resourceGroupName -Location $snapshot.Location -SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $publicIp.Id

$VirtualMachine = Add-AzVMNetworkInterface -VM $VirtualMachine -Id $nic.Id

#Create the virtual machine with Managed Disk

New-AzVM -VM $VirtualMachine -ResourceGroupName $resourceGroupName -Location $snapshot.Location