Managing Microsoft Azure by using   
Windows PowerShell

**Aditee Verma**

Seidenberg School of Computer Science and Information Systems,

Pace University

Author Note

Correspondence concerning this article should be addressed to Aditee Verma,

Department of Computer Science and Information Systems, Pace University,

New York City, NY 10038

Contact: av11813n@pace.edu

Abstract

This paper explores how the Azure PowerShell module is used to create and manage Azure resources from the PowerShell command line using Windows PowerShell Integrated Scripting Environment(ISE). It outlines the benefits of using Windows Azure from PowerShell as compared to portal. This paper also showcases basic Azure virtual machine(VM) deployment items such as selecting a VM size, selecting a VM image, and deploying a VM.

Keywords: Virtual Machine, Windows PowerShell Integrated Scripting Environment

**Managing Microsoft Azure by using   
Windows PowerShell**

Microsoft Azure (formerly Windows Azure) is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language.

# Azure PowerShell

Azure PowerShell(APS) is basically an extension of Windows PowerShell. It lets Windows PowerShell users control Azure’s robust functionality. From the command line, Azure PowerShell programmers use preset scripts called cmdlets to perform complex tasks like provisioning virtual machines (VMs) or creating cloud services. APS can work programmatically too, to automate tasks. While some users complain the terminal feels “unfinished” and support is lacking, proponents point out the ease of use aids typically intensive tasks.

### Getting started with Azure PowerShell

The simplest way to get started is to launch Cloud Shell which can be launched from the top navigation of the Azure portal. Then choose the most suitable subscription and create a storage account. Once the storage has been created, the Cloud Shell will open a PowerShell session in the browser. We can also install Azure PowerShell and use it locally in a PowerShell session. The next step is to make sure we have the latest version of the Azure PowerShell installed. If not install Azure PowerShell.

#### Introducing the Windows PowerShell ISE

The Windows PowerShell Integrated Scripting Environment (ISE) is a host application for Windows PowerShell. In Windows PowerShell ISE, we can run commands and write, test, and debug scripts in a single Windows-based graphic user interface with multiline editing, tab completion, syntax coloring, selective execution, context-sensitive help, and support for right-to-left languages.

##### Cmdlets are the heart-and-soul of Windows PowerShell

A cmdlet is a lightweight command that is used in the Windows PowerShell environment. To get started hit Start on your Windows operating system and type “Windows PowerShell ISE” and run it as administrator. Running Windows PowerShell ISE as an administrator is required to install the azure modules and change execution policy. The below listed tables include step by step description of cmdlets which must be run in Windows PowerShell ISE to manage Microsoft Azure.

Table 1

Cmdlets to install Azure and AzureRM Modules and login to Azure account.

|  |  |
| --- | --- |
| PowerShell Cmdlet | Cmdlet Description and Output |
| Start-Transcript -Path "C:\transcripts\transcript0.txt" -NoClobber | Creates a record of all or part of a Windows PowerShell session to a text file. The transcript includes all commands that the user types and all output that appears on the console. This is especially useful when your enterprise's logging is centralized. |
| get-ExecutionPolicy | Output: Restricted  Tells you which of the four execution policies is currently in-force. The Windows PowerShell execution policies include the following:  Restricted - No scripts can be run. Windows PowerShell can be used only in interactive mode.  AllSigned - Only scripts signed by a trusted publisher can be run.  RemoteSigned - Downloaded scripts must be signed by a trusted publisher before they can be run.  Unrestricted - No restrictions; all Windows PowerShell scripts can be run. |
| Set-ExecutionPolicy RemoteSigned | Sets the execution policy as RemoteSigned to download azure modules. |
| Install-Module AzureRM | Installing AzureRM PowerShell repository from the PowerShell Gallery. AzureRM includes the new school repository. |
| Install-Module Azure -AllowClobber | Installing Azure PowerShell repository from the PowerShell Gallery. Azure is the classic version.  AzureRM and Azure have conflicting repository hence we use AllowClobber to overwrite them. |
| get-command -Module AzureRM | Displays the repository. |
| (get-command -Module AzureRM).count | Output: 2038 |
| (get-command -Module Azure).count | Output: 687 |
| Login-AzureRmAccount | You will get dialog box asking for your Azure credentials. Once you have signed in to an Azure account, you can use the Azure PowerShell cmdlets to access and manager the resources in your subscription. |

Table 2

Cmdlets to Create and Manage Windows VMs with the Azure PowerShell module.

|  |  |
| --- | --- |
| PowerShell Cmdlet | Cmdlet Description and Output |
| New-AzureRmResourceGroup -ResourceGroupName myResourceGroupVM -Location EastUS | An Azure resource group is a logical container into which Azure resources are deployed and managed. A resource group must be created before a virtual machine. In this example, a resource group named myResourceGroupVM is created in the EastUS region. The resource group is specified when creating or modifying a VM. |
| $subnetConfig = New-AzureRmVirtualNetworkSubnetConfig `  -Name mySubnet `  -AddressPrefix 192.168.1.0/24 | A virtual machine must be connected to a virtual network. To create a virtual network, we need to create a subnet first. |
| $vnet = New-AzureRmVirtualNetwork `  -ResourceGroupName myResourceGroupVM `  -Location EastUS `  -Name myVnet `  -AddressPrefix 192.168.0.0/16 `  -Subnet $subnetConfig | Creates a virtual network. |
| $pip = New-AzureRmPublicIpAddress `  -ResourceGroupName myResourceGroupVM `  -Location EastUS `  -AllocationMethod Static `  -Name myPublicIPAddress | We communicate with the virtual machine using a public IP address through a network interface card. Hence creating a public IP address. |
| $nic = New-AzureRmNetworkInterface `  -ResourceGroupName myResourceGroupVM `  -Location EastUS `  -Name myNic `  -SubnetId $vnet.Subnets[0].Id `  -PublicIpAddressId $pip.Id | Creating a network interface card. |
| $nsgRule = New-AzureRmNetworkSecurityRuleConfig `  -Name myNSGRule `  -Protocol Tcp `  -Direction Inbound `  -Priority 1000 `  -SourceAddressPrefix \* `  -SourcePortRange \* `  -DestinationAddressPrefix \* `  -DestinationPortRange 3389 `  -Access Allow | An Azure network security group (NSG) controls inbound and outbound traffic for one or many virtual machines. Network security group rules allow or deny network traffic on a specific port or port range. These rules can also include a source address prefix so that only traffic originating at a predefined source can communicate with a virtual machine. To access the IIS webserver that you are installing, you must add an inbound NSG rule.  To create an inbound NSG rule, use Add-AzureRmNetworkSecurityRuleConfig. This cmdlet creates an NSG rule named myNSGRule that opens port 3389 for the virtual machine. |
| $nsg = New-AzureRmNetworkSecurityGroup `  -ResourceGroupName myResourceGroupVM `  -Location EastUS `  -Name myNetworkSecurityGroup `  -SecurityRules $nsgRule | Creating the NSG using myNSGRule. |
| Set-AzureRmVirtualNetworkSubnetConfig `  -Name mySubnet `  -VirtualNetwork $vnet `  -NetworkSecurityGroup $nsg `  -AddressPrefix 192.168.1.0/24 | Adding the NSG to the subnet in the virtual network. |
| Set-AzureRmVirtualNetwork -VirtualNetwork $vnet | Updating the virtual network. |
| $cred = Get-Credential | Setting the username and password needed for the administrator account on the virtual machine. |
| $vm = New-AzureRmVMConfig -VMName myVM -VMSize Standard\_D1 | Creating the initial configuration for the virtual machine. |
| $vm = Set-AzureRmVMOperatingSystem `  -VM $vm `  -Windows `  -ComputerName myVM `  -Credential $cred `  -ProvisionVMAgent -EnableAutoUpdate | Add the operating system information to the virtual machine configuration. |
| $vm = Set-AzureRmVMSourceImage `  -VM $vm `  -PublisherName MicrosoftWindowsServer `  -Offer WindowsServer `  -Skus 2016-Datacenter `  -Version latest | Adding the image information to the virtual machine configuration. |
| $vm = Set-AzureRmVMOSDisk `  -VM $vm `  -Name myOsDisk `  -DiskSizeInGB 128 `  -CreateOption FromImage `  -Caching ReadWrite | Adding the operating system disk settings to the virtual machine configuration. |
| $vm = Add-AzureRmVMNetworkInterface -VM $vm -Id $nic.Id | Adding the network interface card that you previously created to the virtual machine configuration. |
| New-AzureRmVM -ResourceGroupName myResourceGroupVM -Location EastUS -VM $vm | Creating the virtual machine |
| Get-AzureRmPublicIpAddress -ResourceGroupName myResourceGroupVM | Select IpAddress | Output: 2.170.220.255  After the deployment has completed, create a remote desktop connection with the virtual machine.  Run this command to return the public IP address of the virtual machine. |
| mstsc /v: 2.170.220.255 | Using this command, on the local machine, we create a remote desktop session with the virtual machine. When prompted, enter the credentials used when creating the virtual machine. |
| Get-AzureRmVMImagePublisher -Location "EastUS" | The Azure marketplace includes many virtual machine images that can be used to create a new virtual machine. This command returns a list of image publishers. |
| Get-AzureRmVMImageOffer -Location "EastUS" -PublisherName "MicrosoftWindowsServer" | This command returns a list of image offers filtered on the specified publisher. |
| Get-AzureRmVMImageSku -Location "EastUS" -PublisherName "MicrosoftWindowsServer" -Offer "WindowsServer" | This command filter on the publisher and offer name to return a list of image names. |
| $vm = Set-AzureRmVMSourceImage `  -VM $vm `  -PublisherName MicrosoftWindowsServer `  -Offer WindowsServer `  -Skus 2016-Datacenter-with-Containers `  -Version latest | Using information from above three outputs we can deploy a VM with a specific image. This command sets the image name on the VM object. |
| Get-AzureRmVMSize -Location EastUS | A virtual machine size determines the amount of compute resources such as CPU, GPU, and memory that are made available to the virtual machine. Virtual machines need to be created with a size appropriate for the expect work load. If workload increases, an existing virtual machine can be resized.  Use this command to see a list of VM sizes available in a region. |
| Get-AzureRmVMSize -ResourceGroupName myResourceGroupVM -VMName myVM | After a VM has been deployed, it can be resized to increase or decrease resource allocation.  Before resizing a VM, check if the desired size is available on the current VM cluster. This command returns a list of sizes. |
| $vm = Get-AzureRmVM -ResourceGroupName myResourceGroupVM -VMName myVM  $vm.HardwareProfile.VmSize = "Standard\_D4"  Update-AzureRmVM -VM $vm -ResourceGroupName myResourceGroupVM | If the desired size is available, the VM can be resized from a powered-on state, however it is rebooted during the operation. |
| Stop-AzureRmVM -ResourceGroupName myResourceGroupVM -Name "myVM" -Force  $vm = Get-AzureRmVM -ResourceGroupName myResourceGroupVM -VMName myVM  $vm.HardwareProfile.VmSize = "Standard\_F4s"  Update-AzureRmVM -VM $vm -ResourceGroupName myResourceGroupVM  Start-AzureRmVM -ResourceGroupName myResourceGroupVM -Name $vm.name | If the desired size is not on the current cluster, the VM needs to be deallocated before the resize operation can occur. Note, when the VM is powered back on, any data on the temp disk are removed, and the public IP address change unless a static IP address is being used. |
| Get-AzureRmVM `  -ResourceGroupName myResourceGroupVM `  -Name myVM `  -Status | Select @{n="Status"; e={$\_.Statuses[1].Code}} | Output: PowerState/running  An Azure VM can have one of many power states. This state represents the current state of the VM from the standpoint of the hypervisor.  To retrieve the state of a VM, use this command. |
| Stop-AzureRmVM -ResourceGroupName myResourceGroupVM -Name "myVM" -Force | Stop and deallocate a virtual machine.  If you want to keep the virtual machine in a provisioned state, use the -StayProvisioned parameter. |
| Start-AzureRmVM -ResourceGroupName myResourceGroupVM -Name myVM | Starting a virtual machine. |
| Remove-AzureRmResourceGroup -Name myResourceGroupVM -Force | Deleting a resource group also deletes all resources contained within. |

I performed similar steps and created another VM with the name myVM2 under the same resource group “myResourceGroupVM”. Below screen capture shows the two VMs added to my portal:

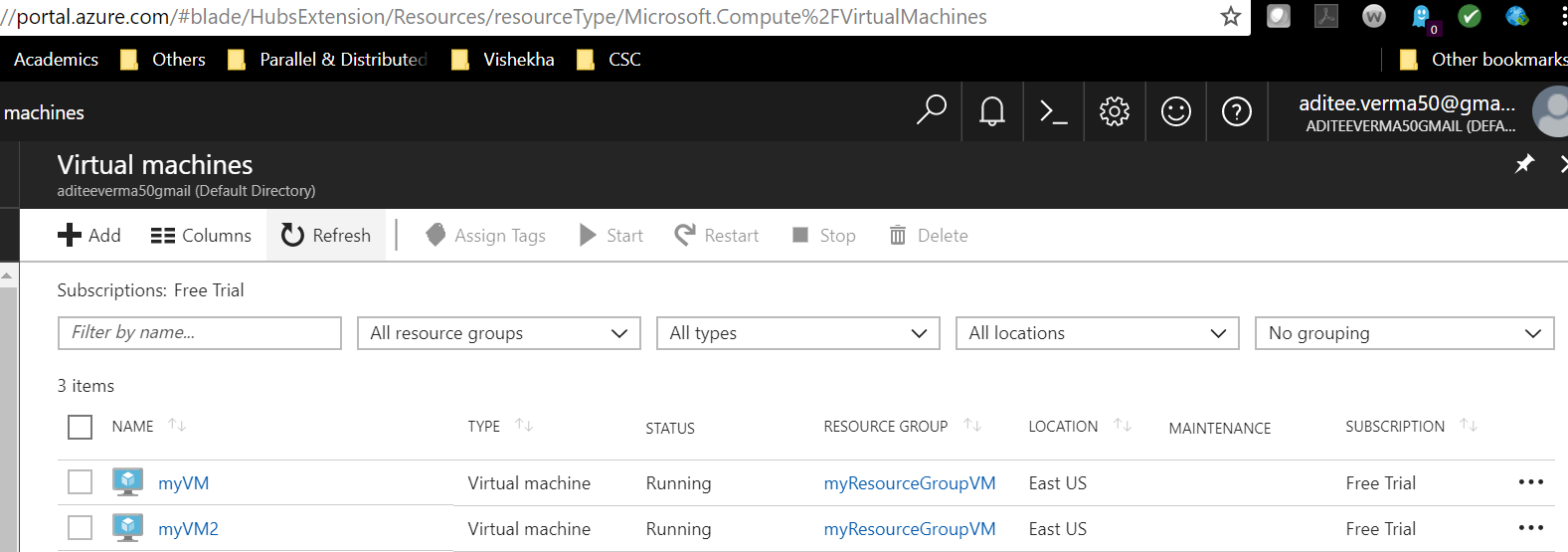


Table 3

Cmdlets to start and stop multiple VMs running using a single command - Automation

|  |  |
| --- | --- |
| PowerShell Cmdlet | Cmdlet Description and Output |
| get-azurermvm -ResourceGroupName MYRESOURCEGROUPVM | foreach {start-azurermvm -Name $\_.name -resourcegroupname MYRESOURCEGROUPVM} | Starts all the VMs under the resource group “MYRESOURCEGROUPVM” |
| get-azurermvm -ResourceGroupName MYRESOURCEGROUPVM | foreach {stop-azurermvm -Name $\_.name -resourcegroupname MYRESOURCEGROUPVM -confirm:$false -force} | Stops all the VMs under the resource group “MYRESOURCEGROUPVM”  -confirm:$false-force : this attribute deallocate the resources as well  -StayProvisioned: This attribute can be used when you don’t want to deallocate resources. So time taken to restart the VMs would be less. |

## Azure PowerShell key features

1. Automation: PowerShell is all about automation. Query, manage and configure Windows Azure resources (VMs, Cloud Services, Websites, Storage, Queues, Databases etc..) across multiple subscriptions, multiple storage accounts and data centers.

2. Provision fully composed virtual machines: When we work through the portal we first add the virtual machine and then add disc to it which is a separate operation. Similarly, for adding five discs to a virtual machine it would require five different operations. From PowerShell we can create a virtual machine with all the discs and all the networking configured all at once. So, whenever we’re doing big deployments on multiple servers, we can do this in a very repeatable fashion. Also, Virtual machine can be started domain joined at the provision time using PowerShell.

3. Virtual Networking: We can configure and manage Virtual Network configuration and VPN gateway using Azure PowerShell.

4. Storage: We can upload and download VHDs from windows Azure storage accounts to the on-premises services. Also, VHDs can be copied between storage accounts and subscriptions.

References

1. https://stackify.com/what-is-azure-powershell/
2. https://www.youtube.com/watch?v=3Q0jG1Doa-s
3. https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-manage-vm
4. https://docs.microsoft.com/en-us/powershell/azure/get-started-azureps?view=azurermps-5.0.0
5. https://en.wikipedia.org/wiki/PowerShell
6. https://en.wikipedia.org/wiki/Microsoft\_Azure
7. https://docs.microsoft.com/en-us/powershell/scripting/core-powershell/ise/introducing-the-windows-powershell-ise?view=powershell-5.1
8. https://technet.microsoft.com/en-us/library/ee176847.aspx