PowerShell and Azure CLI Reference

Introduction:

Welcome to the PowerShell Reference Guide. This guide will provide you with a reference to key PowerShell commands necessary for Azure administrators as well as required to pass the Azure Administrator certification exams from Microsoft.

If you are completely new to PowerShell, we highly recommend you check out the Microsoft Azure PowerShell Overview which has a number of tutorials and guides for learning the basics. This guide is made up of several PowerShell commands which have been reference from the Microsoft documentation and other sources. Before running any of these commands in production, please be sure to test them out in an Azure test account. Some commands are destructive in nature (e.g. removing resource groups, tags etc.) and you need to make sure you fully understand the commands that you execute.

The guide is divided up into the following sections:

- Downloading PowerShell and Installing Azure ARM Modules for PowerShell
- Accounts and Subscriptions
- Resource Groups
- Governance
- Storage
- Virtual Machines
- Networking
- Azure Active Directory

If you spot any errors in this guide, please submit them via the Contact Us page on the Skylines Academy web site.

Thank you,

Skylines Academy Team
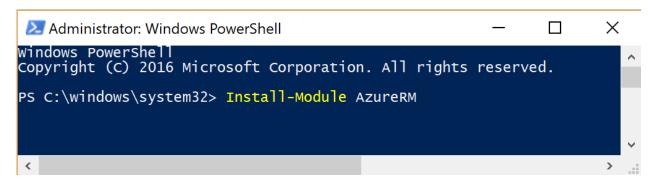
Downloading PowerShell:

https://azure.microsoft.com/en-gb/downloads/

All Azure administrators will require PowerShell along with the AzureRM module installed on their laptops.

Installing AzureRM Module (Windows Example)

Installing Azure PowerShell from the PowerShell Gallery requires elevated privileges. Run the following command from an elevated PowerShell session (Search for PowerShell → Right Click → Run as Administrator)

```
Administrator: Windows PowerShell                                    —    □    ✕

Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\windows\system32> Install-Module AzureRM
```

By default, the PowerShell gallery is not configured as a Trusted repository for PowerShellGet. You will see the following prompts. Enter Yes to all.

```
Select Administrator: Windows PowerShell                              —    □    ✕

Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

 Installing package 'AzureRM'
    Installing dependent package 'AzureRM.SiteRecovery'
    [oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo  ]
 Installing package 'AzureRM.SiteRecovery'
       Downloaded 0.20 MB out of 1.95 MB.
       [ooooooo                                                                 ]

NuGet provider now?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its InstallationPolicy
value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y
```

Untrusted repository

Make sure to choose yes when prompted to install modules from the untrusted repositories. You can make these repos trusted by using the Set-PSRepository cmdlet and changing the installation policy if you desire given that the source is PSGallery.

Are you sure you want to install the modules from 'PSGallery'?

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y

Answer 'Yes' or 'Yes to All' to continue with the installation.

Note

If you have a version older than 2.8.5.201 of NuGet, you are prompted to download and install the latest version of NuGet.+

The AzureRM module is a rollup module for the Azure Resource Manager cmdlets. When you install the AzureRM module, any Azure PowerShell module not previously installed is downloaded and from the PowerShell Gallery.+

If you have a previous version of Azure PowerShell installed you may receive an error. To resolve this issue, see the Updating to a new version of Azure PowerShell section of this article.+

Reference: https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-4.4.0#step-2-install-azure-powershell

Azure Cloud Shell

Reference content from following: https://docs.microsoft.com/en-us/azure/cloud-shell/overview?view=azurermps-4.4.0

Accounts and Subscriptions

<u>Azure Accounts</u>

Login to Azure Account	`Login-AzureRMAccount`
Logout of the Azure account you are connected with in your session	`Disconnect-AzureRmAccount` `Note:` Upon entering this command, you will be presented with a popup window to complete your login process and any MFA requirements.

Upon entering this command, you will be presented with a popup window to complete your login process and any MFA requirements.

Subscription Selection

List all subscriptions in all tenants the account can access	`Get-AzureRmSubscription`
Get subscriptions in a specific tenant	`Get-AzureRmSubscription -TenantId "xxxx-xxxx-xxxx-xxxx"`
Choose subscription	`Select-AzureRmSubscription –SubscriptionID "SubscriptonID"` Note: Use Get-AzureRMSubscription to identity the subscriptionID.

name containing the text specified.	

Resource Group Provisioning & Management

Create a new Resource Group	**New-AzureRmResourceGroup -Name 'SkylinesRG' -Location 'northcentral'** `#Creates a new resource group in North Central called "Skylines RG"`
Delete a Resource Group	`Remove-AzureRmResourceGroup -Name "SL-RGToDelete"`

Moving Resources from one Resource Group to another

Step 1: Retrieve existing Resource	`$Resource = Get-AzureRmResource -ResourceType "Microsoft.ClassicCompute/storageAccounts" -ResourceName "SkylinesStorageAccount"` **# Retrieves a storage account called "SkylinesStorageAccount"**

Step 2: Move the Resource to the New Group	`Move-AzureRmResource -ResourceId $Resource.ResourceId -DestinationResourceGroupName "SL-NewRG"`
	
	`# Moves the resource from Step 1 into the destination resource group "SL-NewRG"`

Resource Group Tags

Display Tags associated with a specific **resource group** name	`(Get-AzureRmResourceGroup -Name "SkylinesRG").Tags`
To get all Azure **resource groups** with a specific tag:	`(Find-AzureRmResourceGroup -Tag @{ Owner="Skylines Academy" }).Name`

To get specific **resources** with a specific tag:	```(Find-AzureRmResource -TagName Dept -TagValue Finance).Name```

Adding Tags

Add Tags to an existing resource group that has no tags	```Set-AzureRmResourceGroup -Name examplegroup -Tag @{ Dept="IT"; Environment="Test" }```

Adding tags to an existing resource group that has tags
 	
1. Get Tags
2. Append
3. Update/Apply Tags
	```$tags = (Get-AzureRmResourceGroup -Name examplegroup).Tags```
	```$tags += @{Status="Approved"}```
	```Set-AzureRmResourceGroup -Tag $tags -Name examplegroup```

Add tags to a specific resource without tags	```$r = Get-AzureRmResource -ResourceName examplevnet -ResourceGroupName examplegroup``` ```Set-AzureRmResource -Tag @{ Dept="IT"; Environment="Test" } -ResourceId $r.ResourceId -Force```

Apply all tags from an existing resource group to the resources beneath. (Note: this overrides all existing tags on the resources inside the RG)	```$groups = Get-AzureRmResourceGroup```
	```foreach ($group in $groups)```
	```{```
	``` Find-AzureRmResource -ResourceGroupNameEquals $g.ResourceGroupName |```
	```ForEach-Object {Set-AzureRmResource -ResourceId $_.ResourceId -Tag $g.Tags -Force }```
	```}```

Apply all tags from a resource group to its resources, but retain tags on resources that are not duplicates	```$groups = Get-AzureRmResourceGroup```
	```foreach ($g in $groups)```
	```{```
	``` if ($g.Tags -ne $null) {```
	``` $resources = Find-AzureRmResource -ResourceGroupNameEquals $g.ResourceGroupName```
	``` foreach ($r in $resources)```
	``` {```
	``` $resourcetags = (Get-AzureRmResource -ResourceId $r.ResourceId).Tags```
	``` foreach ($key in $g.Tags.Keys)```
	``` {```
	``` if```

```
($resourcetags.ContainsKey($key)) {
$resourcetags.Remove($key) }
                }
                $resourcetags += $g.Tags
                Set-AzureRmResource -Tag
$resourcetags -ResourceId $r.ResourceId -Force
            }
        }
}
```

Remove all tags (**Caution**)

Removes all tags by passing an empty hash	`Set-AzureRmResourceGroup -Tag @{} -Name exampleresourcegroup`

Governance

<u>Azure Policies: View Policies and Assignments</u>

See all policy definitions in your subscription	`Get-AzureRmPolicyDefinition`
Retrieve assignments for a specific resource group	`$rg = Get-AzureRmResourceGroup -Name "ExampleGroup" (Get-AzureRmPolicyAssignment -Name accessTierAssignment -Scope $rg.ResourceId`

<u>Create Policies</u>

Step 1	`Create the policy in JSON`

Step 2	`Pass the file using  Powershell`
	
	`Example:`
	
	`$definition = New-AzureRmPolicyDefinition` `` ` ``
	`    -Name denyRegions` `` ` ``
	`    -DisplayName "Deny specific regions"` `` ` ``
	`    -Policy 'https://githublocation.com/azurepolicy.rules.json'`
	
	`You can also use a local file as follows:`
	
	`$definition = New-AzureRmPolicyDefinition` `` ` ``
	`    -Name denyCoolTiering` `` ` ``
	`    -Description "Deny cool access tiering for storage"` `` ` ``
	`    -Policy "c:\policies\coolAccessTier.json"`

<u>Assign Policies</u>

Apply a policy from a definition created above	```$rg = Get-AzureRmResourceGroup -Name "ExampleGroup" New-AzureRMPolicyAssignment -Name denyRegions -Scope $rg.ResourceId -PolicyDefinition $definition```

Resource Locks

Create a new resource lock	```New-AzureRmResourceLock -LockLevel ReadOnly -LockNotes "Notes about the lock" -LockName "SL-WebSiteLock" -ResourceName "SL-WebSite" -ResourceType "microsoft.web/sites"``` # Creates a new ReadOnly resource lock on a web site resource.
Retrieve a resource lock	```Get-AzureRmResourceLock -LockName "SL-WebSiteLock" -ResourceName "SL-WebSite" -ResourceType "microsoft.web/sites" -ResourceGroupName "SL-RGWebSite"```

Storage

Retrieving Storage Accounts

Lists all storage accounts in the current subscription	`Get-AzureRMStorageAccount`

Create Storage Account

Create Storage Account Requires the resource group name, storage account name, valid Azure location, and type (SkuName).	`New-AzureRmStorageAccount -ResourceGroupName` "slstoragerg" `-Name` "slstorage1" `-Location` "eastus"`-SkuName` "Standard_LRS"
SKU Options	• Standard_LRS. Locally-redundant storage. • Standard_ZRS. Zone-redundant storage. • Standard_GRS. Geo-redundant storage. • Standard_RAGRS. Read access geo-redundant storage. • Premium_LRS. Premium locally-redundant storage.
Optional Key Parameters	-Kind The kind parameter will allow you to specify the type of Storage Account. • **Storage -** General purpose Storage account that supports storage of Blobs, Tables, Queues, Files and Disks. • **StorageV2 -** General Purpose Version 2 (GPv2) Storage account that supports Blobs, Tables, Queues, Files, and Disks, with advanced features like data tiering. • **BlobStorage** -Blob Storage account which supports storage of Blobs only. The default value is Storage. -Access Tier

	If you specify BlobStorage as the "Kind" then you must also include an access tier • **Hot** • **Cold**
Create a storage container in a storage Account (using storage account name)	New-AzureRmStorageContainer -ResourceGroupName "slstoragerg" -AccountName "slstorageaccount" -ContainerName "slContainer"
Create a storage container in a storage account (using the storage account object)	1. Get the storage account and store it as a variable ➢ $storageaccount = Get-AzureRmStorageAccount -ResourceGroupName "slstoragerg" -AccountName "slstorageaccount" 2. Make sure you have the right one ➢ $storageaccount This will show you the storage account object you stored in the variable $storageaccount 3. Create the container in the storage account object ➢ New-AzureRmStorageContainer -StorageAccount $accountObject -ContainerName "slContainer" -

Remove Accounts and Containers

Delete a storage account	`Remove-AzureRmStorageAccount -ResourceGroupName "slstoragerg" -AccountName "slstorageaccount"`
Delete a storage container using storage account name and container name	`Remove-AzureRmStorageContainer -ResourceGroupName "slstoragerg" -AccountName "slstorageaccount" -ContainerName "slcontainer"`

Delete a storage container using the storage account object	`Remove-AzureRmStorageContainer -StorageAccount $storageaccount -ContainerName "slcontainer"`
	
	`Note: Make sure to storage the storage account as a variable first using`
	
	➢ `$storageaccount = Get-AzureRmStorageAccount -ResourceGroupName "slstoragerg" -AccountName "slstorageaccount"`

Deploy and Manage Virtual Machines

Get Information About VMs

Task	Command
List all VMs in current subscription	Get-AzureRmVM
List VMs in a resource group (See Resource Groups section above)	Get-AzureRmVM -ResourceGroupName $slResourceGroup
Get a specific virtual machine	Get-AzureRmVM -ResourceGroupName "slresourcegroup" -Name "myVM"

Create a VM – Simplified

I put this command here as it is a quick way to create a VM, but you are far better off using VM configurations to create your VMs with more specific parameters applied. Try out both of them and you will see the difference.

Task	Command
Create a simple VM	New-AzureRmVM -Name "vmname" Typing in this simple command will create a VM and populate names for all the associated objects based on the VM name specified.

Create a VM Configuration Before Creating the Virtual Machine

Use the following tasks to create a new VM configuration before creating your Virtual Machine based on that config.

Task	Command
Create a VM configuration	$vmconfig = New-AzureRmVMConfig -VMName "systemname" -VMSize "Standard_D1_v2"
Add configuration settings This adds the operating system settings to the configuration.	$vmconfig = Set-AzureRmVMOperatingSystem -VM $vmconfig -Windows -ComputerName "systemname" -Credential $cred -ProvisionVMAgent -EnableAutoUpdate
Add a network interface	$vmconfig = Add-AzureRmVMNetworkInterface -VM $vmconfig -Id $nic.Id
Specify a platform image	$vmconfig = Set-AzureRmVMSourceImage -VM $vmconfig -PublisherName "publisher_name" -Offer "publisher_offer" -Skus "product_sku" -Version "latest"
Create a VM	New-AzureRmVM -ResourceGroupName "slresourcegroup" -Location "eastus" -VM $vmconfigconfig All resources are created in the resource group. Before you run this command, run New-AzureRmVMConfig, Set-AzureRmVMOperatingSystem, Set-AzureRmVMSourceImage, Add-AzureRmVMNetworkInterface, and Set-AzureRmVMOSDisk.

VM Operations

Task	Command
Start a VM	Start-AzureRmVM -ResourceGroupName "slresourcegroup" -Name "vmname"
Stop a VM	Stop-AzureRmVM -ResourceGroupName "slresourcegroup" -Name "vmname"
Restart a running VM	Restart-AzureRmVM -ResourceGroupName "slresourcegroup" -Name "vmname"
Delete a VM	Remove-AzureRmVM -ResourceGroupName "slresourcegroup" -Name "vmname"

Networking

Get/List Networking

List virtual networks	Get-AzureRmVirtualNetwork -ResourceGroupName "slresourcegroup" Lists all the virtual networks in the resource group.
Get information about a virtual network	Get-AzureRmVirtualNetwork -Name "myVNet" -ResourceGroupName "slresourcegroup"
List subnets in a virtual network	Get-AzureRmVirtualNetwork -Name "myVNet" -ResourceGroupName "slresourcegroup" | Select Subnets
Get information about a subnet	Get-AzureRmVirtualNetworkSubnetConfig -Name "mySubnet1" - VirtualNetwork $vnet Gets information about the subnet in the specified virtual network. The $vnet value represents the object returned by Get-AzureRmVirtualNetwork you used previously.
Get all IP addresses from a resource group	Get-AzureRmPublicIpAddress -ResourceGroupName "slresourcegroup"
Get all load balancers from a resource group	Get-AzureRmLoadBalancer -ResourceGroupName "slresourcegroup"
Get all network interfaces from a resource group	Get-AzureRmNetworkInterface -ResourceGroupName "slresourcegroup"
Get information about a network interface	Get-AzureRmNetworkInterface -Name "slNIC" -ResourceGroupName "slresourcegroup"
Get the IP configuration of a network interface	Get-AzureRmNetworkInterfaceIPConfig -Name "slNICIP" -NetworkInterface $nic

	Gets information about the IP configuration of the specified network interface. The $nic value represents the object returned by Get-AzureRmNetworkInterface.

Create Network Resources

Create subnet configurations	$subnet1 = New-AzureRmVirtualNetworkSubnetConfig -Name "slSubnet1" -AddressPrefix XX.X.X.X/XX $subnet2 = New-AzureRmVirtualNetworkSubnetConfig -Name "slSubnet2" -AddressPrefix XX.X.X.X/XX
Create a virtual network	$vnet = New-AzureRmVirtualNetwork -Name "myVNet" -ResourceGroupName "slresourcegroup" -Location $location -AddressPrefix XX.X.X.X/XX -Subnet $slsubnet1, $slsubnet2 Note: Make sure to create the subnets first as per the previous command above.
Test for a unique domain name	Test-AzureRmDnsAvailability -DomainNameLabel "myDNS" -Location $location You can specify a DNS domain name for a public IP resource, which creates a mapping for domainname.location.cloudapp.azure.com to the public IP address in the Azure-managed DNS servers. The name can contain only letters, numbers, and hyphens. The first and last character must be a letter or number and the domain name must be unique within its Azure location. If **True** is returned, your proposed name is globally unique.
Create a public IP address	$pip = New-AzureRmPublicIpAddress -Name "myPublicIp" -ResourceGroupName "slresourcegroup" -DomainNameLabel "myDNS" -Location $location -AllocationMethod Dynamic The public IP address uses the domain name that you previously tested and is used by the frontend configuration of the load balancer.
Create a frontend IP configuration	$frontendIP = New-AzureRmLoadBalancerFrontendIpConfig -Name "myFrontendIP" -PublicIpAddress $pip

	The frontend configuration includes the public IP address that you previously created for incoming network traffic.
Create a backend address pool	$beAddressPool = New-AzureRmLoadBalancerBackendAddressPoolConfig -Name "myBackendAddressPool" Provides internal addresses for the backend of the load balancer that are accessed through a network interface.
Create a probe	$healthProbe = New-AzureRmLoadBalancerProbeConfig -Name "myProbe" -RequestPath 'HealthProbe.aspx' -Protocol http -Port 80 -IntervalInSeconds 15 -ProbeCount 2 Contains health probes used to check availability of virtual machines instances in the backend address pool.
Create a load balancing rule	$lbRule = New-AzureRmLoadBalancerRuleConfig -Name HTTP -FrontendIpConfiguration $frontendIP -BackendAddressPool $beAddressPool -Probe $healthProbe -Protocol Tcp -FrontendPort 80 -BackendPort 80 Contains rules that assign a public port on the load balancer to a port in the backend address pool.
Create an inbound NAT rule	$inboundNATRule = New-AzureRmLoadBalancerInboundNatRuleConfig -Name "myInboundRule1" -FrontendIpConfiguration $frontendIP -Protocol TCP -FrontendPort 3441 -BackendPort 3389 Contains rules mapping a public port on the load balancer to a port for a specific virtual machine in the backend address pool.
Create a load balancer	$loadBalancer = New-AzureRmLoadBalancer -ResourceGroupName "slresourcegroup" -Name "myLoadBalancer" -Location $location -FrontendIpConfiguration $frontendIP -InboundNatRule $inboundNATRule -LoadBalancingRule $lbRule -BackendAddressPool $beAddressPool -Probe $healthProbe
Create a network interface	$nic1= New-AzureRmNetworkInterface -ResourceGroupName "slresourcegroup" -Name "myNIC" -Location $location -PrivateIpAddress XX.X.X.X -Subnet $subnet2 -LoadBalancerBackendAddressPool $loadBalancer.BackendAddressPools[0] -

| | LoadBalancerInboundNatRule $loadBalancer.InboundNatRules[0]

Create a network interface using the public IP address and virtual network subnet that you previously created. |
|---|---|

Remove Network Resources

| Delete a virtual network | Remove-AzureRmVirtualNetwork -Name "myVNet" -ResourceGroupName "slresourcegroup"

Removes the specified virtual network from the resource group. |
|---|---|
| Delete a network interface | Remove-AzureRmNetworkInterface -Name "myNIC" -ResourceGroupName "slresourcegroup"

Removes the specified network interface from the resource group. |
| Delete a load balancer | Remove-AzureRmLoadBalancer -Name "myLoadBalancer" -ResourceGroupName "slresourcegroup"

Removes the specified load balancer from the resource group. |
| Delete a public IP address | Remove-AzureRmPublicIpAddress-Name "myIPAddress" -ResourceGroupName "slresourcegroup"

Removes the specified public IP address from the resource group. |

Azure Active Directory Commands

Install Azure AD Module

In order to use the Azure AD commands, you first need to install the Azure AD module. Use the following procedure to get it installed:

1. Open PowerShell

2. Type "Install-Module AzureAD"

3. Press Y to accept the untrusted repository (PSGallery).

```
PS C:\> Install-Module AzureAD

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y
```

Connect to Azure AD

Connect to Azure Active Directory	Connect-AzureAD Note: You will be prompted to enter your credentials and any additional authentication steps required.
Disconnect from Azure Active Directory	Disconnect-AzureAD

User and Service Principal Management

Get all users	Get-AzureADUser
Get specific user	Get-AzureADUser -ObjectId "user@skylinesexam.com"
Remove User	Remove-AzureADUser -ObjectId "user@skylinesexam.com"
New User Creation This is a 3 step process that requires first creating a password profile, setting the password, and then passing these into the New-AzureADUser command	1. Create Password Profile $PasswordProfile = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordProfile 2. Set Password

	`$PasswordProfile.Password = "Password"`
	
	3. Create User
	
	`New-AzureADUser -DisplayName "New User" -PasswordProfile $PasswordProfile -UserPrincipalName "user@contoso.com" -AccountEnabled $true -MailNickName "Newuser"`

Service Principal Creation	First you need to create your application registration in AzureAD then you retrieve it with this command.
	
	`Get-AzureRmADApplication -DisplayNameStartWith slappregistration`
	
	Once you have the application ID for the App registration, you can use it to create the SPN (Service Principal)
	
	`New-AzureRmADServicePrincipal -ApplicationId 11111111-1111-1111-1111-11111111111 -Password $securePassword`

Assign Role This will be scoped to the resource group name you type in with the role definition assigned to the SPN i.e. The SPN is allowed to do X at the RG named Y	`New-AzureRmRoleAssignment -ResourceGroupName "slresourcegroup" -ObjectId 11111111-1111-1111-1111-11111111111 -RoleDefinitionName Reader`
View Current Role Assignment	`Get-AzureRmRoleAssignment -ResourceGroupName "slresourcegroup" -ObjectId 11111111-1111-1111-1111-11111111111`