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 AZ-104](https://docs.microsoft.com/en-us/learn/certifications/exams/az-104)

🔧 Technologies & Tools

Cheatsheet with the most common Microsoft Azure PowerShell commands with examples.

Each section covers one specific set of resources you can manage with your Azure PowerShell Module Per each section you will find the following information:

* **Title**: The resources set name that you can manage with the Azure PowerShell Module
* **Command**: Remember this one. It's the basic thing to start. For instance, Get-AzResourceGroup is all about manage resource groups.
* **Basic actions**: Basic actions you can do in the command. Tip: mentally join command and basic action and you will be at half way to use the AZ Module to manage the resource.
* **Examples**: Set of examples using that command and the basic actions. First, it is described in human expression what you want to perform and the line after, the command that perform the action. Try to link the way of thinking and the sequence of the command and you will never forget a command again.

**Note**: In the examples is assumed that all the actions are based on your subscription account.

**Azure PowerShell Cheatsheet**

Check out the [Microsoft Azure PowerShell Overview](https://docs.microsoft.com/en-us/powershell/azure/azureps-support-lifecycle?view=azps-7.1.0) 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:

1. Downloading PowerShell and Installing Azure AZ Modules for PowerShell
2. Accounts and Subscriptions
3. Resource Groups
4. Governance
5. Storage
6. Virtual Machines
7. Networking
8. Azure Active Directory

\*\*Accounts and Subscriptions \*\*

**Azure Accounts**

**Login to Azure Account**

Login-AzAccount

**Logout of the Azure account you are connected with in yoursession**

Logout-AzAccount

**Subscription Selection**

**List all subscriptions in all tenants the account can access**

Get-AzSubscription

**Get subscriptions in a specific tenant**

Get-AzSubscription -TenantId "xxxx-xxxx-xxxxxxxx"

**Choose subscription**

Select-AzSubscription –SubscriptionID “SubscriptonID”

**Resource Groups**

**Retrieving Resource Groups**

**Get all resource groups (Gets the resource group and additional details which can also be stored for use by additional commands).**

Get-AzResourceGroup

**Get a specific resource group by name**

Get-AzResourceGroup -Name "myResourceGroup”

**Get resource groups where the name begins with "Production"**

Get-AzResourceGroup | Where ResourceGroupName -like Production\*

**Show resource groups by location**

Get-AzResourceGroup | Sort Location,ResourceGroupName | Format-Table -GroupBy Location ResourceGroupName,ProvisioningState,Tags

**Resources within RGs**

**Find resources of a type in resource groups with a specific name**

Get-AzResource -ResourceGroupName "myResourceGroup"

**Find resources of a type matching against the resource name string**

Note: The difference with this command vs the one above, is that this one does not look for a specific resource group, but rather just all resources with a name containing the text specified.

Get-AzResource -ResourceType

"microsoft.web/sites" -ResourceGroupName

"myResourceGroup"

**Resource Group Provisioning & Management**

###Create a new Resource Group

New-AzResourceGroup -Name 'myResourceGroup' -Location 'westeurope' #Creates a new resource group in West Europe called "myResourceGroup"

**Delete a Resource Group**

Remove-AzResourceGroup -Name "ResourceGroupToDelete"

**Moving Resources from One Resource Group to Another**

**Step 1: Retrieve existing Resource**

$Resource = Get-AzResource -ResourceType

"Microsoft.ClassicCompute/storageAccounts" - #Retrieves a storage account called "myStorageAccount"

ResourceName "myStorageAccount"

**Step 2: Move the Resource to the New Group**

Move-AzResource -ResourceId

$Resource.ResourceId -DestinationResourceGroupName - #Moves the resource from Step 1 into the destination resource group "NewResourceGroup"

"NewResourceGroup"

**Resource Group Tags**

**Display Tags associated with a specific resource group name**

(Get-AzResourceGroup -Name "myResourceGroup").Tags

**To get all Azure resource groups with a specific tag:**

(Get-AzResourceGroup -Tag @{Owner="DesiredOwner"}).Name

**To get specific resources with a specific tag:**

(Get-AzResource -TagName Dept -TagValue Finance).Name

**Adding Tags**

###Add Tags to an existing resource group that has no tags

Set-AzResourceGroup -Name examplegroup -Tag @{Dept="IT"; Environment="Production" }

**Adding tags to an existing**

resource group that has tags

1. Get Tags
2. Append
3. Update/Apply Tags

$tags = (Get-AzResourceGroup -Name

examplegroup).Tags

$tags += @{Status="Approved"}

Set-AzResourceGroup -Tag $tags -Name examplegroup

**Add tags to a specific resource without tags**

$r = Get-AzResource -ResourceName examplevnet -ResourceGroupName examplegroup Set-AzResource -Tag @{ Dept="IT";Environment="Production" } -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-AzResourceGroup foreach

($group in $groups)

{

Find-AzResource -

ResourceGroupNameEquals $g.ResourceGroupName |

ForEach-Object {Set-AzResource -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**

$groups = Get-AzResourceGroup foreach ($g

in $groups)

{

if ($g.Tags -ne $null) {

$resources = Find-AzResource

ResourceGroupNameEquals $g.ResourceGroupName

foreach ($r in $resources)

{

$resourcetags = (Get-AzResource

-ResourceId $r.ResourceId).Tags

foreach ($key in $g.Tags.Keys)

{

if

($resourcetags.ContainsKey($key)) {

$resourcetags.Remove($key) }

}

$resourcetags += $g.Tags

Set-AzResource -Tag

$resourcetags -ResourceId $r.ResourceId -Force

}

}

}

**Remove all tags**

**Removes all tags by passing an empty hash**

Set-AzResourceGroup -Tag @{} -Name exampleresourcegroup

***Governance***

\*\*Azure Policies: View Policies and Assignments

**See all policy definitions in your subscription**

Get-AzPolicyDefinition

**Retrieve assignments for a specific resource group**

$rg = Get-AzResourceGroup -Name

"ExampleGroup"

(Get-AzPolicyAssignment -Name

accessTierAssignment -Scope $rg.ResourceId

**Create Policies**

**Step 1 Create the policy in JSON**

**Step 2 pass the file using PowerShell**

$definition = New-AzPolicyDefinition `

-Name denyRegions `

-DisplayName "Deny specific regions" `

-Policy

'https://githublocation.com/azurepolicy.rules.js

on'

You can also use a local file as follows:

$definition = New-AzPolicyDefinition `

-Name denyCoolTiering `

-Description "Deny cool access tiering for

storage" `

-Policy "c:\policies\coolAccessTier.json"

**Assign Policies**

**Apply a policy from a definition created above**

$rg = Get-AzResourceGroup -Name

"ExampleGroup"

New-AzPolicyAssignment -Name denyRegions -

Scope $rg.ResourceId -PolicyDefinition

$definition

**Resource Locks**

**Create a new resource lock**

New-AzResourceLock -LockLevel ReadOnly -

LockNotes "Notes about the lock" -LockName "ReadOnlyLock" -ResourceName "Websites-PROD" ResourceType

"microsoft.web/sites" #Creates a new ReadOnly resource lock on a web site resource.

**Retrieve a resource lock.**

Get-AzResourceLock -LockName "ReadOnlyLock" -

ResourceName "Websites-PROD" -ResourceType

"microsoft.web/sites" -ResourceGroupName "RGWebSite"

***Storage***

##Retrieving Storage Accounts

Get-AzStorageAccount

***Create storage account***

**Create Storage Account**

New-AzStorageAccount -ResourceGroupName

“myResourceGroup” -Name “storage1” -Location

“westeurope”-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-AzStorageContainer -ResourceGroupName "storage" -AccountName "storageaccount1" -ContainerName "Container"

**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-AzStorageAccount -

ResourceGroupName "storage" -AccountName

"storageaccount1"

1. Make sure you have the right one

$storageaccount #This will show you the storage account object you stored in the variable $storageaccount

1. Create the container in the storage account object

NewAzStorageContainer -StorageAccount

$accountObject -ContainerName "Container"

**Remove Accounts and Containers**

**Delete a storage account**

Remove-AzStorageAccount -ResourceGroupName "storage" -AccountName "storageaccount1"

**Delete a storage container using storage account name and container name**

Remove-AzStorageContainer -ResourceGroupName "storage" -AccountName "storageaccount1" -ContainerName "container"

**Delete a storage container using the storage account object**

Remove-AzStorageContainer -StorageAccount $storageaccount -ContainerName "container"

Note: Make sure to storage the storage account as a

variable first using

$storageaccount = Get-AzStorageAccount -ResourceGroupName "storage" -AccountName "storageaccount1"

***Deploy and Manage Virtual Machines***

**Get information about VMs**

**List all VMs in current subscription**

Get-AzVM

**List VMs in a resource group See Resource Groups section above)**

Get -AzVM -ResourceGroupName $ResourceGroup

**Get a specific virtual machine**

Get-AzVM -ResourceGroupName “resourcegroup” -Name "myVM"

**Create a VM – Simplified**

Create a simple M

New-AzVM -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.

**Create a VM configuration**

$vmconfig = New-AzVMConfig -VMName “systemname” -VMSize "Standard\_D1\_v2"

**Add configuration settings This adds the operating system settings to the configuration**

$vmconfig = Set-AzVMOperatingSystem -VM $vmconfig -Windows -ComputerName “systemname” -Credential $cred -ProvisionVMAgent EnableAutoUpdate

**Add a network interface**

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

**Specify a platform image**

$vmconfig = Set-AzVMSourceImage -VM $vmconfig -PublisherName "publisher\_name" -Offer "publisher\_offer" -Skus "product\_sku" -Version "latest"

**Create a VM**

New-AzVM -ResourceGroupName “resourcegroup” -Location “westeurope

-VM $vmconfigconfig

All resources are created in the resource group. Before you run this command,

run New-AzVMConfig, Set-AzVMOperatingSystem, SetAzVMSourceImage, Add-AzVMNetworkInterface, and Set-AzVMOSDisk.

**VM Operations**

**Start a VM**

Start-AzVM -ResourceGroupName “resourcegroup” -Name “vmname”

**Stop a VM**

Stop-AzVM -ResourceGroupName “resourcegroup” -Name “vmname”

**Restart a running VM**

Restart-AzVM -ResourceGroupName “resourcegroup” -Name “vmname”

**Delete a VM**

Remove-AzVM -ResourceGroupName “resourcegroup” -Name “vmname”

**Networking** Get/List Networking

**List virtual networks**

Get-AzVirtualNetwork -ResourceGroupName “resourcegroup” #Lists all the virtual networks in the resource group.

**Get information about a virtual network**

Get-AzVirtualNetwork -Name "myVNet" -ResourceGroupName “resourcegroup”

**List subnets in a virtual network**

Get-AzVirtualNetwork -Name "myVNet" -ResourceGroupName “resourcegroup” | Select Subnets

**Get information about a subnet**

Get-AzVirtualNetworkSubnetConfig -Name "mySubnet1" VirtualNetwork $vnet #Gets information about the subnet in the specified virtual network. The $vnet

value represents the object returned by Get-AzVirtualNetwork you used

previously.

**Get all IP addresses from a resource group**

Get-AzPublicIpAddress -ResourceGroupName “resourcegroup”

**Get all load balancers from a resource group**

Get-AzLoadBalancer -ResourceGroupName “resourcegroup”

**Get all network interfaces from a resource group**

Get-AzNetworkInterface -ResourceGroupName “resourcegroup”

**Get information about a network interface**

Get-AzNetworkInterface -Name "NIC1" -ResourceGroupName “resourcegroup”

**Get the IP configuration of a network interface**

Get-AzNetworkInterfaceIPConfig -Name "NIC1" -NetworkInterface $nic #Gets information about the IP configuration of the specified network interface.

The $nic value represents the object returned by Get-AzNetworkInterface.

**Check provisioning status for Azure Express Route Circuit**

Get-AzExpressRouteCircuit -ResourceGroupName "Test-Resource" -Name "Test-Circuit"

**Upgrade ExpressRoute Gateway SKU**

Resize-AzVirtualNetworkGateway

**Create Network Resources**

**Create subnet configurations**

$subnet1 = New-AzVirtualNetworkSubnetConfig -Name "Subnet1" -AddressPrefix XX.X.X.X/XX

$subnet2 = New-AzVirtualNetworkSubnetConfig -Name "Subnet2" -AddressPrefix XX.X.X.X/XX

**Create a virtual network**

$vnet = New-AzVirtualNetwork -Name "myVNet" -ResourceGroupName “resourcegroup” -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-AzDnsAvailability -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 Azuremanaged 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-AzPublicIpAddress -Name "myPublicIp" -ResourceGroupName “resourcegroup” -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-AzLoadBalancerFrontendIpConfig -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-AzLoadBalancerBackendAddressPoolConfig -Name "myBackendAddressPool" #Provides internal addresses for the backend of the load balancer that are accessed through a network interface.

**Create a probe**

$healthProbe = New-AzLoadBalancerProbeConfig -Name "myProbe" RequestPath 'HealthProbe.aspx' -Protocol http -Port 80 -IntervalInSeconds 15 ProbeCount 2 #

**Create a load balancing rule**

$lbRule = New-AzLoadBalancerRuleConfig -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-AzLoadBalancerInboundNatRuleConfig -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-AzLoadBalancer -ResourceGroupName “resourcegroup”

-Name "myLoadBalancer" -Location $location -FrontendIpConfiguration $frontendIP

InboundNatRule $inboundNATRule -LoadBalancingRule $lbRule -BackendAddressPool

$beAddressPool -Probe $healthProbe

**Create a network interface**

$nic1= New-AzNetworkInterface -ResourceGroupName “resourcegroup” 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-AzVirtualNetwork -Name "myVNet" -ResourceGroupName “resourcegroup” #Removes the specified virtual network from the resource group

**Delete a network interface**

Remove-AzNetworkInterface -Name "myNIC" -ResourceGroupName “resourcegroup” #Removes the specified network interface from the resource group

**Delete a load balancer**

Remove-AzLoadBalancer -Name "myLoadBalancer" -ResourceGroupName “resourcegroup” #Removes the specified load balancer from the resource group

**Delete a public IP address**

Remove-AzPublicIpAddress-Name "myIPAddress" -ResourceGroupName “resourcegroup” #Removes the specified public IP address from the resource group.

**Reset an ExpressRoute circuit**

When an operation on an ExpressRoute circuit doesn't complete successfully, the circuit might go into a "failed" state. You can reset a failed ExpressRoute circuit using Azure PowerShell. You will need the latest version of the Azure Resource Manager PowerShell cmdlets.

$ckt = Get-AzExpressRouteCircuit -Name "ExpressRouteARMCircuit" -ResourceGroupName "ExpressRouteResourceGroup"

Set-AzExpressRouteCircuit -ExpressRouteCircuit $ckt

**Identify the root cause for ExpressRoute latency issues**

To troubleshoot latency issues with ExpressRoute, the Azure Connectivity Toolkit includes a tool called iPerf.

You use iPerf for basic performance tests, by copying the files to a directory on the host. To test performance, follow these steps:

1. Install the PowerShell Module.

(new-object Net.WebClient).DownloadString("https://aka.ms/AzureCT") | Invoke-Expression

This command downloads the PowerShell module and installs it locally.

1. Install the supporting applications.

Install-LinkPerformance

This AzureCT command installs iPerf and PSPing in a new directory, "C:\ACTTools". It also opens the Windows Firewall ports to allow ICMP and port 5201 (iPerf) traffic.

1. Run the performance test. First, on the remote host, you must install and run iPerf in server mode. Ensure the remote host is listening on either 3389 (RDP for Windows) or 22 (SSH for Linux) and allowing traffic on port 5201 for iPerf. If the remote host is Windows, install the AzureCT and run the Install-LinkPerformance command. The command will set up iPerf and the necessary firewall rules.

When the remote machine is ready, open PowerShell on the local machine and start the test:

Get-LinkPerformance -RemoteHost IP -TestSeconds 10

This command runs a series of concurrent load and latency tests to help estimate the bandwidth capacity and latency of your network link.

4.Review the output of the tests.

The detailed results of iPerf tests are in individual text files in the AzureCT tools directory at "C:\ACTTools."

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)

**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@contoso.com"

**Remove User**

Remove-AzureADUser -ObjectId "user@contoso.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 NewAzureADUser command

1. Create Password Profile

$PasswordProfile = New-Object -TypeName Microsoft.Open.AzureAD.Model.PasswordProfile

1. Set Password

$PasswordProfile.Password = "Password"

1. 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-AzADApplication -DisplayNameStartWith slappregistration

Once you have the application ID for the App registration, you can use it to create the SPN (Service Principal)

New-AzADServicePrincipal -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-AzRoleAssignment -ResourceGroupName “resourcegroup” -ObjectId 11111111-1111-1111-1111-11111111111 -RoleDefinitionName Reader

**View Current Role Assignment**

Get-AzRoleAssignment -ResourceGroupName “resourcegroup” -ObjectId 11111111-1111-1111-1111-11111111111