

# Table of Contents

## Azure PowerShell

### Install

- Install on macOS and Linux

- Other installation methods

### Get started

### Cloud Shell

### Log in

- Create a service principal

- Using persistent credentials

### Queries

### Formatting

### Subscriptions

### Deploy

- Deploy using Resource Manager templates

- Export Resource Manager templates

- Deploy private Resource Manager template with SAS token

### Concepts

- Using experimental modules

- Using PowerShell jobs

### Release notes

- Breaking changes for Azure PowerShell 5.x

## Tutorials

- Create Virtual Machines

## Sample Scripts

- Linux Virtual Machines

- Windows Virtual Machines

- Web Apps

- SQL Databases

- Cosmos DB



# Overview of Azure PowerShell

4/10/2018 • 1 min to read • [Edit Online](#)

Azure PowerShell provides a set of cmdlets that use the [Azure Resource Manager](#) model for managing your Azure resources. You can use it in your browser with [Azure Cloud Shell](#), or you can install it on your local machine and use it in any PowerShell session.

Use the [Cloud Shell](#) to run the Azure PowerShell in your browser, or [install](#) it on own computer. Then read the [Get Started](#) article to begin using it. For information about the latest release, see the [release notes](#).

The following samples can help you learn how to perform common scenarios with Azure PowerShell:

- [Linux Virtual Machines](#)
- [Windows Virtual Machines](#)
- [Web Apps](#)
- [SQL Databases](#)

## NOTE

If you have deployments that use the classic deployment model that cannot be converted, you can install the Service Management version of Azure PowerShell. For more information, see [Install the Azure PowerShell Service Management module](#).

## Need help with PowerShell?

If you are unfamiliar with PowerShell, you may find an introduction to PowerShell helpful.

- [Installing PowerShell](#)
- [Scripting with PowerShell](#)

You can also watch this video: [PowerShell Basics: \(Part 1\) Getting Started with PowerShell](#).

Or attend the Microsoft Virtual Academy's [Getting Started with PowerShell Jumpstart](#).

## Other Azure PowerShell modules

- [Azure Active Directory](#)
- [Azure Information Protection](#)
- [Azure Service Fabric](#)
- [Azure ElasticDB](#)

# Install and configure Azure PowerShell

4/10/2018 • 4 min to read • [Edit Online](#)

This article explains the steps to install the Azure PowerShell modules in a Windows environment. If you want to use Azure PowerShell on macOS or Linux, see the following article: [Install and configure Azure PowerShell on macOS and Linux](#).

Installing Azure PowerShell from the PowerShell Gallery is the preferred method of installation.

## Step 1: Install PowerShellGet

Installing items from the PowerShell Gallery requires the PowerShellGet module. Make sure you have the appropriate version of PowerShellGet and other system requirements. Run the following command to see if you have PowerShellGet installed on your system.

```
Get-Module -Name PowerShellGet -ListAvailable | Select-Object -Property Name,Version,Path
```

You should see something similar to the following output:

Name	Version	Path
PowerShellGet	1.6.0	C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.6.0\PowerShellGet.psd1
PowerShellGet	1.0.0.1	C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PowerShellGet.psd1

You need PowerShellGet version 1.1.2.0 or higher. To update PowerShellGet, use the following command:

```
Install-Module PowerShellGet -Force
```

If you do not have PowerShellGet installed, see the [How to get PowerShellGet](#) section of this article.

### NOTE

Using PowerShellGet requires an Execution Policy that allows you to run scripts. For more information about PowerShell's Execution Policy, see [About Execution Policies](#).

## Step 2: Install Azure PowerShell

Installing Azure PowerShell from the PowerShell Gallery requires elevated privileges. Run the following command from an elevated PowerShell session:

```
# Install the Azure Resource Manager modules from the PowerShell Gallery
Install-Module -Name AzureRM -AllowClobber
```

By default, the PowerShell gallery is not configured as a Trusted repository for PowerShellGet. The first time you use the PSGallery you see the following prompt:

#### 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

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.

## Step 3: Load the AzureRM module

Once the module is installed, you need to load the module into your PowerShell session. You should do this in a normal (non-elevated) PowerShell session. Modules are loaded using the `Import-Module` cmdlet, as follows:

```
Import-Module -Name AzureRM
```

## Next Steps

For more information about using Azure PowerShell, see the following articles:

- [Get started with Azure PowerShell](#)

## Reporting issues and feedback

If you encounter any bugs with the tool, file an issue in the [Issues](#) section of our GitHub repo. To provide feedback from the command line, use the `Send-Feedback` cmdlet.

## Frequently asked questions

### How to get PowerShellGet

SCENARIO	INSTALL INSTRUCTIONS
Windows 10 Windows Server 2016	Built into Windows Management Framework (WMF) 5.0 included in the OS
I want to upgrade to PowerShell 5	<ol style="list-style-type: none"><li>1. <a href="#">Install the latest version of WMF</a></li><li>2. Run the following command: <pre>Install-Module PowerShellGet -Force</pre></li></ol>

SCENARIO	INSTALL INSTRUCTIONS
I am running on a version of Windows with PowerShell 3 or PowerShell 4	<a href="#">Get the PackageManagement modules</a> 1. Run the following command: <pre>Install-Module PowerShellGet -Force</pre>

## Checking the version of Azure PowerShell

Although we encourage you to upgrade to the latest version as early as possible, several versions of Azure PowerShell are supported. To determine the version of Azure PowerShell you have installed, run

```
Get-Module AzureRM
```

from your command line.

```
Get-Module AzureRM -ListAvailable | Select-Object -Property Name,Version,Path
```

## Support for classic deployment methods

If you have deployments that use the classic deployment model you can install the Service Management version of Azure PowerShell. For more information, see [Install the Azure PowerShell Service Management module](#). The Azure and AzureRM modules share common dependencies. If you use both the Azure and AzureRM modules, you should install the same version of each package.

## Updating to a new version of Azure PowerShell

If you have a previous version of Azure PowerShell installed that includes the Service Management module, you may receive the following error:

```
PackageManagement\Install-Package : A command with name 'Get-AzureStorageContainerAcl' is already
available on this system. This module 'Azure.Storage' may override the existing commands. If you
still want to install this module 'Azure.Storage', use -AllowClobber parameter.

At C:\Program Files\WindowsPowerShell\Modules\PowerShellGet\1.0.0.1\PSModule.psm1:1772 char:21
+ ...          $null = PackageManagement\Install-Package @PSBoundParameters
+ ~~~~~
+ CategoryInfo          : InvalidOperation: (Microsoft.Power....InstallPackage:InstallPackage) [Install-
Package], Exception
+ FullyQualifiedErrorId : CommandAlreadyAvailable,Validate-
ModuleCommandAlreadyAvailable,Microsoft.PowerShell.PackageManagement.Cmdlets.InstallPackage
```

As the error message states, you need to use the `-AllowClobber` parameter to install the module. Use the following command:

```
# Install the Azure Resource Manager modules from the PowerShell Gallery
Install-Module -Name AzureRM -AllowClobber
```

For more information, see the help topic for [Install-Module](#).

## Installing module versions side by side

The PowerShellGet method of installation is the only method that supports the installation of multiple versions. For example, you may have scripts written using a previous version of Azure PowerShell that you don't have the time or resources to updated. The following commands illustrate how to install multiple versions of Azure PowerShell:

```
Install-Module -Name AzureRM -RequiredVersion 3.7.0
Install-Module -Name AzureRM -RequiredVersion 1.2.9
```

Only one version of the module can be loaded in a PowerShell session. You must open a new PowerShell window

and use `Import-Module` to import a specific version of the AzureRM cmdlets:

```
Import-Module -Name AzureRM -RequiredVersion 1.2.9
```

#### NOTE

Version 2.1.0 and version 1.2.6 are the first module versions designed to be installed and used side by side. When loading an earlier version of the Azure PowerShell, incompatible versions of the **AzureRM.Profile** module are loaded. This results in the cmdlets prompting you to log in whenever you execute a cmdlet.

#### Other installation methods

For information about installing using the Web Platform Installer or the MSI Package, see [Other installation methods](#)

# Install and configure Azure PowerShell on macOS and Linux

4/10/2018 • 2 min to read • [Edit Online](#)

It is now possible to install PowerShell Core v6 and Azure PowerShell on non-Windows platforms. The process of installing Azure PowerShell on macOS and Linux is not that different from Windows, however, you must first install PowerShell Core v6.

## NOTE

At this time, both PowerShell Core v6 and Azure PowerShell for .NET Core are still in beta. Support for these products is limited. If you have problems or discover bugs, please file Issues in GitHub.

- [Issues for PowerShell Core v6](#)
- [Issues for Azure PowerShell](#)

## Step 1: Install PowerShell Core v6

The process of installing PowerShell Core v6 varies depending on the target operating system. While it is possible to install PowerShell Core v6 on Windows, this article focuses on macOS and Linux. If you want to use Azure PowerShell on Windows, see the [install](#) article for Windows.

Installing **PowerShell Core v6** on Linux or macOS varies depending on the Linux distribution and OS version. Detailed instructions can be found in the following article:

- [Installing PowerShell Core on macOS and Linux](#)

## Step 2: Install Azure PowerShell for .NET Core

PowerShell Core v6 comes with the PowerShellGet module already installed. This makes it easy to install any module that is published to the PowerShell Gallery. To install Azure PowerShell, open a new PowerShell session and run the following command:

```
Install-Module AzureRM.NetCore
```

## Step 3: Load the AzureRM.Netcore module

Once the module is installed, you need to load the module into your PowerShell session. Modules are loaded using the `Import-Module` cmdlet, as follows:

```
Import-Module AzureRM.Netcore
Import-Module AzureRM.Profile.Netcore
```

After the import completes, you can test your newly installed and module by attempting to sign into Azure using the following command:

```
Login-AzureRmAccount
```



The above command should prompt you to go to `https://aka.ms/devicelogin` and enter the provided code.

## Available cmdlets

The Azure PowerShell modules for .NET Standard are still in development. These modules do not provide the full set of cmdlets that are available for the Windows version of the modules. The following functions are implemented in AzureRM.Netcore modules:

- Account management
  - Login with Microsoft account, Organizational account, or Service Principal through Microsoft Azure Active Directory
  - Save Credentials to disk with Save-AzureRmContext and load saved credentials using Import-AzureRmContext
- Environment
  - Get the different out-of-box Microsoft Azure environments
  - Add/Set/Remove customized environments (like your Azure Stack or Windows Azure Pack environments)
- Management plane cmdlets for Azure services using Resource Manager and Service Management interfaces.
  - Virtual Machine
  - App Service (Websites)
  - SQL Database
  - Storage
  - Network

## Next Steps

For more information about using Azure PowerShell, see the [Get started with Azure PowerShell](#) article.

# Other installation methods

4/10/2018 • 2 min to read • [Edit Online](#)

Azure PowerShell has multiple installation methods. Using PowerShellGet with the PowerShell Gallery is the preferred method. Azure PowerShell can be installed on Windows using the Web Platform Installer (WebPI) or by using the MSI file available from GitHub. Azure PowerShell can also be installed in a Docker container.

## Install on Windows using the Web Platform Installer

Installing the latest Azure PowerShell from WebPI is the same as it was for previous versions. Download the [Azure PowerShell WebPI package](#) and start the install.

### NOTE

If you have previously installed Azure modules from the PowerShell Gallery, the installer automatically removes them. This simplifies your environment by ensuring that only one version of Azure PowerShell is installed. However, there are scenarios where you may need multiple versions installed at the same time.

PowerShell Gallery modules install modules in `$env:ProgramFiles\WindowsPowerShell\Modules`. In contrast, the WebPI installer installs the Azure modules in `$env:ProgramFiles(x86)\Microsoft SDKs\Azure\PowerShell\`.

If an error occurs during install, you can manually remove the Azure\* folders in your `$env:ProgramFiles\WindowsPowerShell\Modules` folder, and try the installation again.

Once the installation completes, your `$env:PSModulePath` setting should include the directories containing the Azure PowerShell cmdlets. The following command can be used to verify that the Azure PowerShell is installed properly.

```
# To make sure the Azure PowerShell module is available after you install
Get-Module -ListAvailable Azure* | Select-Object Name, Version, Path
```

### NOTE

There is a known issue that can occur when installing from WebPI. If your computer requires a restart due to system updates or other installations, it may cause updates to `$env:PSModulePath` to fail to include the path where Azure PowerShell is installed.

When attempting to load or execute cmdlets after installation, you can receive the following error message:

```
PS C:\> Connect-AzureRmAccount
Connect-AzureRmAccount : The term 'Connect-AzureRmAccount' is not recognized as the name of a cmdlet,
function, script file, or operable program. Check the spelling of the name, or if a path was
included, verify that the path is correct and try again.
At line:1 char:1
+ Connect-AzureRmAccount
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (Connect-AzureRmAccount:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

This error can be corrected by restarting the machine or importing the module using the fully qualified path. For example:

```
Import-Module "$env:ProgramFiles(x86)\Microsoft SDKs\Azure\PowerShell\AzureRM.psd1"
```

## Install on Windows using the MSI Package

Azure PowerShell can be installed using the MSI file available from [GitHub](#). If you have installed previous versions of Azure modules, the installer automatically removes them. The MSI package installs modules in

```
$env:ProgramFiles\WindowsPowerShell\Modules
```

 but does not create version-specific folders.

## Install in a Docker container

We maintain a Docker image preconfigured with Azure PowerShell.

Run the container with `docker run`.

```
docker run azuresdk/azure-powershell
```

In addition, we maintain a subset of cmdlets as a PowerShell Core container.

For Mac/Linux, use the `latest` image.

```
docker run azuresdk/azure-powershell-core:latest
```

For Windows, use the `nanoserver` image.

```
docker run azuresdk/azure-powershell-core:nanoserver
```

Azure PowerShell is installed on the image via `Install-Module` from the [PowerShell Gallery](#).

# Getting started with Azure PowerShell

4/10/2018 • 9 min to read • [Edit Online](#)

Azure PowerShell is designed for managing and administering Azure resources from the command line, and for building automation scripts that work against the Azure Resource Manager. You can use it in your browser with [Azure Cloud Shell](#), or you can install it on your local machine and use it in any PowerShell session. This article helps get you started using it, and teaches you the core concepts behind it.

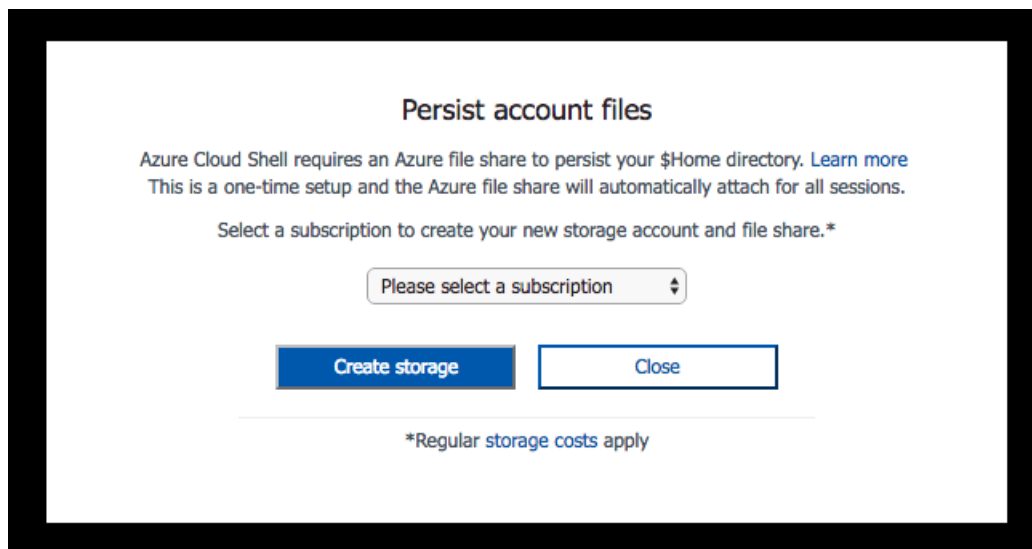
## Connect

The simplest way to get started is to [launch Cloud Shell](#).

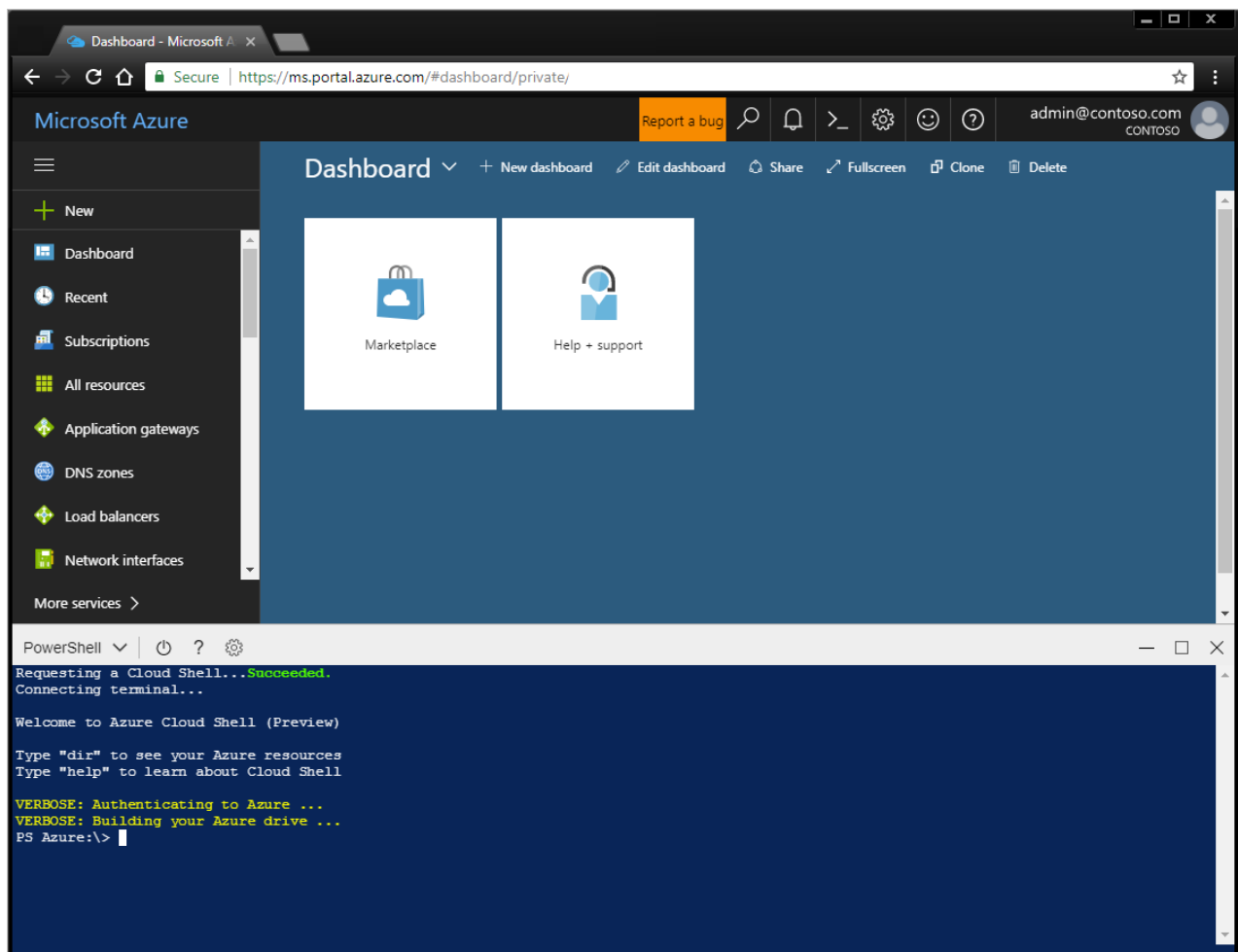
1. Launch Cloud Shell from the top navigation of the Azure portal.



2. Choose the subscription you want to use and create a storage account.

The screenshot shows a dialog box titled 'Persist account files'. The text inside reads: 'Azure Cloud Shell requires an Azure file share to persist your \$Home directory. [Learn more](#). This is a one-time setup and the Azure file share will automatically attach for all sessions.' Below this, it says 'Select a subscription to create your new storage account and file share.\*'. There is a dropdown menu with the text 'Please select a subscription' and a small arrow icon. At the bottom, there are two buttons: 'Create storage' (in blue) and 'Close' (in white with a blue border). A footnote at the bottom states '\*Regular [storage costs](#) apply'.

Once your storage has been created, the Cloud Shell will open a PowerShell session in the browser.



You can also install Azure PowerShell and use it locally in a PowerShell session.

## Install Azure PowerShell

The first step is to make sure you have the latest version of the Azure PowerShell installed. For information about the latest release, see the [release notes](#).

1. [Install Azure PowerShell](#).
2. To verify the installation was successful, run `Get-Module AzureRM -ListAvailable` from your command line.

## Log in to Azure

Sign on interactively:

1. Type `Connect-AzureRmAccount`. You will get dialog box asking for your Azure credentials. Option '-Environment' can let you login in Azure China or Azure Germany.  
e.g. `Connect-AzureRmAccount -Environment AzureChinaCloud`
2. Type the email address and password associated with your account. Azure authenticates and saves the credential information, and then closes the window.

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.

## Create a Windows virtual machine using simple defaults

The `New-AzureRmVM` cmdlet provides a simplified syntax making it easy to create a new virtual machine. There are only two parameter values you must provide: the name of the VM and a set of credentials for the local

administrator account on the VM.

First, create the credential object.

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

```
Windows PowerShell credential request.  
Enter a username and password for the virtual machine.  
User: localAdmin  
Password for user localAdmin: *****
```

Next, create the VM.

```
New-AzureRmVM -Name SampleVM -Credential $cred
```

```
ResourceGroupName      : SampleVM  
Id                     : /subscriptions/XXXXXXX-XXXX-XXXX-XXXX-XXXXXXX/resourceGroups/SampleVM/providers/Microsoft.Compute/virtualMachines/SampleVM  
VmId                   : 43f6275d-ce50-49c8-a831-5d5974006e63  
Name                   : SampleVM  
Type                   : Microsoft.Compute/virtualMachines  
Location               : eastus  
Tags                   : {}  
HardwareProfile        : {VmSize}  
NetworkProfile         : {NetworkInterfaces}  
OSProfile              : {ComputerName, AdminUsername, WindowsConfiguration, Secrets}  
ProvisioningState      : Succeeded  
StorageProfile         : {ImageReference, OsDisk, DataDisks}  
FullyQualifiedDomainName : samplevm-2c0867.eastus.cloudapp.azure.com
```

That was easy. But, you may wonder what else is created and how is the VM configured. First, let's look at our resource groups.

```
Get-AzureRmResourceGroup | Select-Object ResourceGroupName,Location
```

ResourceGroupName	Location
cloud-shell-storage-westus	westus
SampleVM	eastus

The **cloud-shell-storage-westus** resource group is created the first time you use the Cloud Shell. The **SampleVM** resource group was created by the `New-AzureRmVM` cmdlet.

Now, what other resources were created in this new resource group?

```
Get-AzureRmResource |  
Where ResourceGroupName -eq SampleVM |  
Select-Object ResourceGroupName,Location,ResourceType,Name
```

ResourceGroupName	Location	ResourceType	Name
-----	-----	-----	----
SAMPLEVM	eastus	Microsoft.Compute/disks	
SampleVM_OsDisk_1_9b286c54b168457fa1f8c47...			
SampleVM	eastus	Microsoft.Compute/virtualMachines	SampleVM
SampleVM	eastus	Microsoft.Network/networkInterfaces	SampleVM
SampleVM	eastus	Microsoft.Network/networkSecurityGroups	SampleVM
SampleVM	eastus	Microsoft.Network/publicIPAddresses	SampleVM
SampleVM	eastus	Microsoft.Network/virtualNetworks	SampleVM

Let's get some more details about the VM. This examples shows how to retrieve information about the OS Image used to create the VM.

```
Get-AzureRmVM -Name SampleVM -ResourceGroupName SampleVM |
  Select-Object -ExpandProperty StorageProfile |
    Select-Object -ExpandProperty ImageReference
```

```
Publisher : MicrosoftWindowsServer
Offer      : WindowsServer
Sku        : 2016-Datacenter
Version    : latest
Id         :
```

## Create a fully configured Linux Virtual Machine

The previous example used the simplified syntax and default parameter values to create a Windows virtual machine. In this example, we provide values for all options of the virtual machine.

### Create a resource group

For this example we want to create a Resource Group. Resource Groups in Azure provide a way to manage multiple resources that you want to logically group together. For example, you might create a Resource Group for an application or project and add a virtual machine, a database and a CDN service within it.

Let's create a resource group named "MyResourceGroup" in the westeurope region of Azure. To do so type the following command:

```
New-AzureRmResourceGroup -Name 'myResourceGroup' -Location 'westeurope'
```

```
ResourceGroupName : myResourceGroup
Location           : westeurope
ProvisioningState  : Succeeded
Tags               :
ResourceId         : /subscriptions/XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myResourceGroup
```

This new resource group will be used to contain all of the resources needed for the new VM we create. To create a new Linux VM we must first create the other required resources and assign them to a configuration. Then we can use that configuration to create the VM. Also, you will need to have an SSH public key named `id_rsa.pub` in the `.ssh` directory of your user profile.

### Create the required network resources

First we need to create a subnet configuration to be used with the virtual network creation process. We also create a public IP address so that we can connect to this VM. We create a network security group to secure access to the public address. Finally we create the virtual NIC using all of the previous resources.

```
# Variables for common values
$resourceGroup = "myResourceGroup"
$location = "westeurope"
$vmName = "myLinuxVM"

# Definer user name and blank password
$securePassword = ConvertTo-SecureString 'azurepassword' -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

# Create a subnet configuration
$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name mySubnet2 -AddressPrefix 192.168.2.0/24

# Create a virtual network
$vnet = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroup -Location $location `
    -Name MYvNET2 -AddressPrefix 192.168.0.0/16 -Subnet $subnetConfig

# Create a public IP address and specify a DNS name
$publicIp = New-AzureRmPublicIpAddress -ResourceGroupName $resourceGroup -Location $location `
    -Name "mypublicdns$(Get-Random)" -AllocationMethod Static -IdleTimeoutInMinutes 4
$publicIp | Select-Object Name,IpAddress

# Create an inbound network security group rule for port 22
$nsgRuleSSH = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleSSH -Protocol Tcp `
    -Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * `
    -DestinationPortRange 22 -Access Allow

# Create a network security group
$nsg = New-AzureRmNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location `
    -Name myNetworkSecurityGroup2 -SecurityRules $nsgRuleSSH

# Create a virtual network card and associate with public IP address and NSG
$nic = New-AzureRmNetworkInterface -Name myNic2 -ResourceGroupName $resourceGroup -Location $location `
    -SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $publicIp.Id -NetworkSecurityGroupId $nsg.Id
```

## Create the VM configuration

Now that we have the required resources we can create the VM configuration object.

```
# Create a virtual machine configuration
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize Standard_D1 |
    Set-AzureRmVMOperatingSystem -Linux -ComputerName $vmName -Credential $cred -DisablePasswordAuthentication |
    Set-AzureRmVMSourceImage -PublisherName Canonical -Offer UbuntuServer -Skus 14.04.2-LTS -Version latest |
    Add-AzureRmVMNetworkInterface -Id $nic.Id

# Configure SSH Keys
$sshPublicKey = Get-Content "$env:USERPROFILE\.ssh\id_rsa.pub"
Add-AzureRmVMSshPublicKey -VM $vmConfig -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"
```

## Create the virtual machine

Now we can create the VM using the VM configuration object.

```
New-AzureRmVM -ResourceGroupName $resourceGroup -Location $location -VM $vmConfig
```

Now that the VM has been created, you can log on to your new Linux VM using SSH with the public IP address of the VM you created:

```
ssh xx.xxx.xxx.xxx
```



```
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.19.0-65-generic x86_64)

* Documentation:  https://help.ubuntu.com/

System information as of Sun Feb 19 00:32:28 UTC 2017

System load: 0.31           Memory usage: 3%   Processes:      89
Usage of /:  39.6% of 1.94GB Swap usage:   0%   Users logged in: 0

Graph this data and manage this system at:
  https://landscape.canonical.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
  http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

my-login@MyLinuxVM:~/../..$
```

## Creating other resources in Azure

We've now walked through how to create a Resource Group, a Linux VM, and a Windows Server VM. You can create many other types of Azure resources as well.

For example, to create an Azure Network Load Balancer that we could then associate with our newly created VMs, we can use the following create command:

```
New-AzureRmLoadBalancer -Name MyLoadBalancer -ResourceGroupName myResourceGroup -Location westeurope
```

We could also create a new private Virtual Network (commonly referred to as a "VNet" within Azure) for our infrastructure using the following command:

```
$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name mySubnet2 -AddressPrefix 10.0.0.0/16
$vnet = New-AzureRmVirtualNetwork -ResourceGroupName myResourceGroup -Location westeurope `
-Name MYVNET3 -AddressPrefix 10.0.0.0/16 -Subnet $subnetConfig
```

What makes Azure and the Azure PowerShell powerful is that we can use it not just to get cloud-based infrastructure but also to create managed platform services. The managed platform services can also be combined with infrastructure to build even more powerful solutions.

For example, you can use the Azure PowerShell to create an Azure AppService. Azure AppService is a managed platform service that provides a great way to host web apps without having to worry about infrastructure. After creating the Azure AppService, you can create two new Azure Web Apps within the AppService using the following commands:

```
# Create an Azure AppService that we can host any number of web apps within
New-AzureRmAppServicePlan -Name MyAppServicePlan -Tier Basic -NumberofWorkers 2 -WorkerSize Small -
ResourceGroupName myResourceGroup -Location westeurope

# Create Two Web Apps within the AppService (note: name param must be a unique DNS entry)
New-AzureRmWebApp -Name MyWebApp43432 -AppServicePlan MyAppServicePlan -ResourceGroupName myResourceGroup -
Location westeurope
New-AzureRmWebApp -Name MyWebApp43433 -AppServicePlan MyAppServicePlan -ResourceGroupName myResourceGroup -
Location westeurope
```

## Listing deployed resources

You can use the `Get-AzureRmResource` cmdlet to list the resources running in Azure. The following example shows the resources we just created in the new resource group.

```
Get-AzureRmResource |
  Where-Object ResourceGroupName -eq myResourceGroup |
  Select-Object Name,Location,ResourceType
```

Name	Location	ResourceType
----	-----	-----
myLinuxVM_OsDisk_1_36ca038791f642ba91270879088c249a	westeurope	Microsoft.Compute/disks
myWindowsVM_OsDisk_1_f627e6e2bb454c72897d72e9632adf9a	westeurope	Microsoft.Compute/disks
myLinuxVM	westeurope	Microsoft.Compute/virtualMachines
myWindowsVM	westeurope	Microsoft.Compute/virtualMachines
myWindowsVM/BGInfo	westeurope	Microsoft.Compute/virtualMachines/extensions
myNic1	westeurope	Microsoft.Network/networkInterfaces
myNic2	westeurope	Microsoft.Network/networkInterfaces
myNetworkSecurityGroup1	westeurope	Microsoft.Network/networkSecurityGroups
myNetworkSecurityGroup2	westeurope	Microsoft.Network/networkSecurityGroups
mypublicdns245369171	westeurope	Microsoft.Network/publicIPAddresses
mypublicdns779537141	westeurope	Microsoft.Network/publicIPAddresses
MYvNET1	westeurope	Microsoft.Network/virtualNetworks
MYvNET2	westeurope	Microsoft.Network/virtualNetworks
micromyresomywi032907510	westeurope	Microsoft.Storage/storageAccounts

## Deleting resources

To clean up your Azure account, you want to remove the resources we created in this example. You can use the `Remove-AzureRm*` cmdlets to delete the resources you no longer need. To remove the Windows VM we created, using the following command:

```
Remove-AzureRmVM -Name myWindowsVM -ResourceGroupName myResourceGroup
```

You will be prompted to confirm that you want to remove the resource.

```
Confirm
Are you sure you want to remove resource group 'myResourceGroup'
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y
```

You can also use the delete many resources at one time. For example, the following command deletes all the resource group "MyResourceGroup" that we've used for all the samples in this Get Started tutorial. This removes the resource group and all of the resources in it.

```
Remove-AzureRmResourceGroup -Name myResourceGroup
```

Confirm

Are you sure you want to remove resource group 'myResourceGroup'

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

This can take several minutes to complete.

## Get samples

To learn more about ways to use the Azure PowerShell, check out our most common scripts for [Linux VMs](#), [Windows VMs](#), [Web Apps](#), and [SQL Databases](#).

## Next steps

- [Login with Azure PowerShell](#)
- [Manage Azure subscriptions with Azure PowerShell](#)
- [Create service principals in Azure using Azure PowerShell](#)
- Read the Release notes about migrating from an older release: <https://github.com/Azure/azure-powershell/tree/dev/documentation/release-notes>.
- Get help from the community:
  - [Azure forum on MSDN](#)
  - [stackoverflow](#)

# Log in with Azure PowerShell

4/10/2018 • 1 min to read • [Edit Online](#)

Azure PowerShell supports multiple login methods. The simplest way to get started is to log in interactively at the command line.

## Interactive log in

1. Type `Connect-AzureRmAccount`. You will get dialog box asking for your Azure credentials.
2. Type the email address and password associated with your account. Azure authenticates and saves the credential information, and then closes the window.

## Log in with a service principal

Service principals provide a way for you to create non-interactive accounts that you can use to manipulate resources. Service principals are like user accounts to which you can apply rules using Azure Active Directory. By granting the minimum permissions needed to a service principal, you can ensure your automation scripts are even more secure.

1. If you don't already have a service principal, [create one](#).
2. Log in with the service principal.

```
Connect-AzureRmAccount -ServicePrincipal -ApplicationId "http://my-app" -Credential $pscredential -
TenantId $tenantid
```

To get your TenantId, log in interactively and then get the TenantId from your subscription.

```
Get-AzureRmSubscription
```

```
Environment      : AzureCloud
Account          : username@contoso.com
TenantId         : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId    : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionName  : My Production Subscription
CurrentStorageAccount :
```

## Log in using an Azure VM Managed Service Identity

Managed Service Identity (MSI) is a preview feature of Azure Active Directory. You can use an MSI service principal for sign-in, and acquire an app-only access token to access other resources.

For more information about MSI, see [How to use an Azure VM Managed Service Identity \(MSI\) for sign-in and token acquisition](#).

## Log in to another Cloud

Azure cloud services provide different environments that adhere to the data-handling regulations of various governments. If your Azure account is in one the government clouds, you need to specify the environment when you sign in. For example, if you account is in the China cloud you sign on using the following command:

```
Connect-AzureRmAccount -Environment AzureChinaCloud
```

Use the following command to get a list of available environments:

```
Get-AzureRmEnvironment | Select-Object Name
```

```
Name
----
AzureCloud
AzureChinaCloud
AzureUSGovernment
AzureGermanCloud
```

## Learn more about managing Azure role-based access

For more information about authentication and subscription management in Azure, see [Manage Accounts, Subscriptions, and Administrative Roles](#).

Azure PowerShell cmdlets for role management

- [Get-AzureRmRoleAssignment](#)
- [Get-AzureRmRoleDefinition](#)
- [New-AzureRmRoleAssignment](#)
- [New-AzureRmRoleDefinition](#)
- [Remove-AzureRmRoleAssignment](#)
- [Remove-AzureRmRoleDefinition](#)
- [Set-AzureRmRoleDefinition](#)

# Create an Azure service principal with Azure PowerShell

4/10/2018 • 4 min to read • [Edit Online](#)

If you plan to manage your app or service with Azure PowerShell, you should run it under an Azure Active Directory (AAD) service principal, rather than your own credentials. This topic steps you through creating a security principal with Azure PowerShell.

## NOTE

You can also create a service principal through the Azure portal. Read [Use portal to create Active Directory application and service principal that can access resources](#) for more details.

## What is a 'service principal'?

An Azure service principal is a security identity used by user-created apps, services, and automation tools to access specific Azure resources. Think of it as a 'user identity' (username and password or certificate) with a specific role, and tightly controlled permissions. It only needs to be able to do specific things, unlike a general user identity. It improves security if you only grant it the minimum permissions level needed to perform its management tasks.

## Verify your own permission level

First, you must have sufficient permissions in both your Azure Active Directory and your Azure subscription. Specifically, you must be able to create an app in the Active Directory, and assign a role to the service principal.

The easiest way to check whether your account has adequate permissions is through the portal. See [Check required permission in portal](#).

## Create a service principal for your app

Once you are signed into your Azure account, you can create the service principal. You must have one of the following ways to identify your deployed app:

- The unique name of your deployed app, such as "MyDemoWebApp" in the following examples, or
- the Application ID, the unique GUID associated with your deployed app, service, or object

### Get information about your application

The `Get-AzureRmADApplication` cmdlet can be used to discover information about your application.

```
Get-AzureRmADApplication -DisplayNameStartWith MyDemoWebApp
```

```
DisplayName      : MyDemoWebApp
ObjectId         : 775f64cd-0ec8-4b9b-b69a-8b8946022d9f
IdentifierUri    : {http://MyDemoWebApp}
HomePage        : http://www.contoso.com
Type            : Application
ApplicationId    : 00c01aaa-1603-49fc-b6df-b78c4e5138b4
AvailableToOtherTenants : False
AppPermissions   :
ReplyUrls       : {}
```

## Create a service principal for your application

The `New-AzureRmADServicePrincipal` cmdlet is used to create the service principal.

```
Add-Type -Assembly System.Web
$password = [System.Web.Security.Membership]::GeneratePassword(16,3)
New-AzureRmADServicePrincipal -ApplicationId 00c01aaa-1603-49fc-b6df-b78c4e5138b4 -Password $password
```

DisplayName	Type	ObjectId
-----	----	-----
MyDemoWebApp	ServicePrincipal	698138e7-d7b6-4738-a866-b4e3081a69e4

## Get information about the service principal

```
$svcprincipal = Get-AzureRmADServicePrincipal -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4
$svcprincipal | Select-Object *
```

```
ServicePrincipalNames : {http://MyDemoWebApp, 00c01aaa-1603-49fc-b6df-b78c4e5138b4}
ApplicationId         : 00c01aaa-1603-49fc-b6df-b78c4e5138b4
DisplayName            : MyDemoWebApp
Id                    : 698138e7-d7b6-4738-a866-b4e3081a69e4
Type                  : ServicePrincipal
```

## Sign in using the service principal

You can now sign in as the new service principal for your app using the *appid* and *password* you provided. You need to supply the Tenant Id for your account. Your Tenant Id is displayed when you sign into Azure with your personal credentials.

```
$cred = Get-Credential -UserName $svcprincipal.ApplicationId -Message "Enter Password"
Connect-AzureRmAccount -Credential $cred -ServicePrincipal -TenantId XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXX
```

Run this command from a new PowerShell session. After a successfully signing on you see output something like this:

```
Environment      : AzureCloud
Account          : 00c01aaa-1603-49fc-b6df-b78c4e5138b4
TenantId         : XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId    :
SubscriptionName  :
CurrentStorageAccount :
```

Congratulations! You can use these credentials to run your app. Next, you need to adjust the permissions of the service principal.

# Managing roles

## NOTE

Azure Role-Based Access Control (RBAC) is a model for defining and managing roles for user and service principals. Roles have sets of permissions associated with them, which determine the resources a principal can read, access, write, or manage. For more information on RBAC and roles, see [RBAC: Built-in roles](#).

Azure PowerShell provides the following cmdlets to manage role assignments:

- [Get-AzureRmRoleAssignment](#)
- [New-AzureRmRoleAssignment](#)
- [Remove-AzureRmRoleAssignment](#)

The default role for a service principal is **Contributor**. It may not be the best choice depending on the scope of your app's interactions with Azure services, given its broad permissions. The **Reader** role is more restrictive and can be a good choice for read-only apps. You can view details on role-specific permissions or create custom ones through the Azure portal.

In this example, we add the **Reader** role to our prior example, and delete the **Contributor** one:

```
New-AzureRmRoleAssignment -ResourceGroupName myRG -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4 -  
RoleDefinitionName Reader
```

```
RoleAssignmentId    : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-  
XXXXXXXXXXXXXXXX/resourceGroups/myRG/providers/Microsoft.Authorization/roleAssignments/818892f2-d075-46a1-a3a2-  
3a4e1a12fcd5  
Scope                : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myRG  
DisplayName          : MyDemoWebApp  
SignInName           :  
RoleDefinitionName   : Reader  
RoleDefinitionId     : b24988ac-6180-42a0-ab88-20f7382dd24c  
ObjectId             : 698138e7-d7b6-4738-a866-b4e3081a69e4  
ObjectType           : ServicePrincipal
```

```
Remove-AzureRmRoleAssignment -ResourceGroupName myRG -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4 -  
RoleDefinitionName Contributor
```

To view the current roles assigned:

```
Get-AzureRmRoleAssignment -ResourceGroupName myRG -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4
```

```
RoleAssignmentId    : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-  
XXXXXXXXXXXXXXXX/resourceGroups/myRG/providers/Microsoft.Authorization/roleAssignments/0906bbd8-9982-4c03-8dae-  
aeaae8b13f9e  
Scope                : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myRG  
DisplayName          : MyDemoWebApp  
SignInName           :  
RoleDefinitionName   : Reader  
RoleDefinitionId     : acdd72a7-3385-48ef-bd42-f606fba81ae7  
ObjectId             : 698138e7-d7b6-4738-a866-b4e3081a69e4  
ObjectType           : ServicePrincipal
```

Other Azure PowerShell cmdlets for role management:



- [Get-AzureRmRoleDefinition](#)
- [New-AzureRmRoleDefinition](#)
- [Remove-AzureRmRoleDefinition](#)
- [Set-AzureRmRoleDefinition](#)

## Change the credentials of the security principal

It's a good security practice to review the permissions and update the password regularly. You may also want to manage and modify the security credentials as your app changes. For example, we can change the password of the service principal by creating a new password and removing the old one.

### Add a new password for the service principal

```
$password = [System.Web.Security.Membership]::GeneratePassword(16,3)
New-AzureRmADSpCredential -ServicePrincipalName http://MyDemoWebApp -Password $password
```

StartDate	EndDate	KeyId	Type
-----	-----	-----	----
3/8/2017 5:58:24 PM	3/8/2018 5:58:24 PM	6f801c3e-6fcd-42b9-be8e-320b17ba1d36	Password

### Get a list of credentials for the service principal

```
Get-AzureRmADSpCredential -ServicePrincipalName http://MyDemoWebApp
```

StartDate	EndDate	KeyId	Type
-----	-----	-----	----
3/8/2017 5:58:24 PM	3/8/2018 5:58:24 PM	6f801c3e-6fcd-42b9-be8e-320b17ba1d36	Password
5/5/2016 4:55:27 PM	5/5/2017 4:55:27 PM	ca9d4846-4972-4c70-b6f5-a4effa60b9bc	Password

### Remove the old password from the service principal

```
Remove-AzureRmADSpCredential -ServicePrincipalName http://MyDemoWebApp -KeyId ca9d4846-4972-4c70-b6f5-a4effa60b9bc
```

```
Confirm
Are you sure you want to remove credential with keyId '6f801c3e-6fcd-42b9-be8e-320b17ba1d36' for
service principal objectId '698138e7-d7b6-4738-a866-b4e3081a69e4'.
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): Y
```

### Verify the list of credentials for the service principal

```
Get-AzureRmADSpCredential -ServicePrincipalName http://MyDemoWebApp
```

StartDate	EndDate	KeyId	Type
-----	-----	-----	----
3/8/2017 5:58:24 PM	3/8/2018 5:58:24 PM	6f801c3e-6fcd-42b9-be8e-320b17ba1d36	Password

# Persisting user logins across PowerShell sessions

4/10/2018 • 5 min to read • [Edit Online](#)

In the September 2017 release of Azure PowerShell, Azure Resource Manager cmdlets introduce a new feature, **Azure Context Autosave**. This feature enables several new user scenarios, including:

- Retention of login information for reuse in new PowerShell sessions.
- Easier use of background tasks for executing long-running cmdlets.
- Switch between accounts, subscriptions, and environments without a separate login.
- Execution of tasks using different credentials and subscriptions, simultaneously, from the same PowerShell session.

## Azure contexts defined

An *Azure context* is a set of information that defines the target of Azure PowerShell cmdlets. The context consists of five parts:

- An *Account* - the Username or Service Principal used to authenticate communications with Azure
- A *Subscription* - The Azure Subscription containing the Resources being acted upon.
- A *Tenant* - The Azure Active Directory tenant that contains your subscription. Tenants are more important to ServicePrincipal authentication.
- An *Environment* - The particular Azure Cloud being targeted, usually the Azure global Cloud. However, the environment setting allows you to target National, Government, and on-premises (Azure Stack) clouds as well.
- *Credentials* - The information used by Azure to verify your identity and ensure your authorization to access resources in Azure

In previous releases, your Azure Context had to be created each time you opened a new PowerShell session. Beginning with Azure PowerShell v4.4.0, you can enable the automatic saving and reuse of Azure Contexts whenever you open a new PowerShell session.

## Automatically saving the context for the next login

By default, Azure PowerShell discards your context information whenever you close the PowerShell session.

To allow Azure PowerShell to remember your context after the PowerShell session is closed, use `Enable-AzureRmContextAutosave`. Context and credential information are automatically saved in a special hidden folder in your user directory ( `%AppData%\Roaming\Windows Azure PowerShell` ). Subsequently, each new PowerShell session targets the context used in your last session.

To set PowerShell to forget your context and credentials, use `Disable-AzureRmContextAutoSave`. You will need to log in to Azure every time you open a PowerShell session.

The cmdlets that allow you to manage Azure contexts also allow you fine grained control. If you want changes to apply only to the current PowerShell session ( `Process` scope) or every PowerShell session ( `CurrentUser` scope). These options are discussed in more detail in [Using Context Scopes](#).

## Running Azure PowerShell cmdlets as background jobs

The **Azure Context Autosave** feature also allows you to share your context with PowerShell background jobs. PowerShell allows you to start and monitor long-executing tasks as background jobs without having to wait for the tasks to complete. You can share credentials with background jobs in two different ways:

- Passing the context as an argument

Most AzureRM cmdlets allow you to pass the context as a parameter to the cmdlet. You can pass a context to a background job as shown in the following example:

```
PS C:\> $job = Start-Job { param ($ctx) New-AzureRmVm -AzureRmContext $ctx [... Additional parameters ...]} -ArgumentList (Get-AzureRmContext)
```

- Using the default context with Autosave enabled

If you have enabled **Context Autosave**, background jobs automatically use the default saved context.

```
PS C:\> $job = Start-Job { New-AzureRmVm [... Additional parameters ...]}
```

When you need to know the outcome of the background task, use `Get-Job` to check the job status and `Wait-Job` to wait for the Job to complete. Use `Receive-Job` to capture or display the output of the background job. For more information, see [about\\_Jobs](#).

## Creating, selecting, renaming, and removing contexts

To create a context, you must be logged in to Azure. The `Connect-AzureRmAccount` cmdlet (or its alias, `Login-AzureRmAccount`) sets the default context used by subsequent Azure PowerShell cmdlets, and allows you to access any tenants or subscriptions allowed by your login credentials.

To add a new context after login, use `Set-AzureRmContext` (or its alias, `Select-AzureRmSubscription`).

```
PS C:\> Set-AzureRmContext -Subscription "Contoso Subscription 1" -Name "Contoso1"
```

The previous example adds a new context targeting 'Contoso Subscription 1' using your current credentials. The new context is named 'Contoso1'. If you do not provide a name for the context, a default name, using the account ID and subscription ID is used.

To rename an existing context, use the `Rename-AzureRmContext` cmdlet. For example:

```
PS C:\> Rename-AzureRmContext '[user1@contoso.org; 123456-7890-1234-564321]' 'Contoso2'
```

This example renames the context with automatic name `[user1@contoso.org; 123456-7890-1234-564321]` to the simple name 'Contoso2'. Cmdlets that manage contexts also use tab completion, allowing you to quickly select the context.

Finally, to remove a context, use the `Remove-AzureRmContext` cmdlet. For example:

```
PS C:\> Remove-AzureRmContext Contoso2
```

Forgets the context that was named 'Contoso2'. You can recreate this context subsequently using

```
Set-AzureRmContext
```

## Removing credentials

You can remove all credentials and associated contexts for a user or service principal using `Remove-AzureRmAccount` (also known as `Logout-AzureRmAccount`). When executed without parameters, the `Remove-AzureRmAccount` cmdlet removes all credentials and contexts associated with the User or Service Principal in the current context. You may

pass in a Username, Service Principal Name, or context to target a particular principal.

```
Remove-AzureRmAccount user1@contoso.org
```

## Using context scopes

Occasionally, you may want to select, change, or remove a context in a PowerShell session without impacting other sessions. To change the default behavior of context cmdlets, use the `Scope` parameter. The `Process` scope overrides the default behavior by making it apply only for the current session. Conversely `CurrentUser` scope changes the context in all sessions, instead of just the current session.

As an example, to change the default context in the current PowerShell session without impacting other windows, or the context used the next time a session is opened, use:

```
PS C:\> Select-AzureRmContext Contoso1 -Scope Process
```

## How the context autosave setting is remembered

The context `AutoSave` setting is saved to the user Azure PowerShell directory ( `%AppData%\Roaming\Windows Azure PowerShell` ). Some kinds of computer accounts may not have access to this directory. For such scenarios, you can use the environment variable

```
$env:AzureRmContextAutoSave="true" | "false"
```

If set to 'true', the context is automatically saved. If set to 'false', the context is not saved.

## Changes to the AzureRM.Profile module

New cmdlets for managing context

- [Enable-AzureRmContextAutosave](#) - Allow saving the context between powershell sessions. Any changes alter the global context.
- [Disable-AzureRmContextAutosave](#) - Turn off autosaving the context. Each new PowerShell session is required to log in again.
- [Select-AzureRmContext](#) - Select a context as the default. All subsequent cmdlets use the credentials in this context for authentication.
- [Remove-AzureRmAccount](#) - Remove all credentials and contexts associated with an account.
- [Remove-AzureRmContext](#) - Remove a named context.
- [Rename-AzureRmContext](#) - Rename an existing context.

Changes to existing profile cmdlets

- [Add-AzureRmAccount](#) - Allow scoping of the login to the process or the current user. Allow naming the default context after login.
- [Import-AzureRmContext](#) - Allow scoping of the login to the process or the current user.
- [Set-AzureRmContext](#) - Allow selection of existing named contexts, and scope changes to the process or current user.

# Querying for Azure resources

4/10/2018 • 1 min to read • [Edit Online](#)

Querying in PowerShell can be completed by using built-in cmdlets. In PowerShell, cmdlet names take the form of **Verb-Noun**. The cmdlets using the verb **Get** are the query cmdlets. The cmdlet nouns are the types of Azure resources that are acted upon by the cmdlet verbs.

## Selecting simple properties

Azure PowerShell has default formatting defined for each cmdlet. The most common properties for each resource type are displayed in a table or list format automatically. For more information about formatting output, see [Formatting query results](#).

Use the `Get-AzureRmVM` cmdlet to query for a list of VMs in your account.

```
Get-AzureRmVM
```

The default output is automatically formatted as a table.

ResourceGroupName	Name	Location	VmSize	OsType	NIC	ProvisioningState
MYWESTEURG	MyUbuntu1610	westeurope	Standard_DS1_v2	Linux	myubuntu1610980	Succeeded
MYWESTEURG	MyWin2016VM	westeurope	Standard_DS1_v2	Windows	mywin2016vm880	Succeeded

The `Select-Object` cmdlet can be used to select the specific properties that are interesting to you.

```
Get-AzureRmVM | Select Name,ResourceGroupName,Location
```

Name	ResourceGroupName	Location
MyUbuntu1610	MYWESTEURG	westeurope
MyWin2016VM	MYWESTEURG	westeurope

## Selecting complex nested properties

If the property you want to select is nested deep in the JSON output you need to supply the full path to that nested property. The following example shows how to select the VM Name and the OS type from the `Get-AzureRmVM` cmdlet.

```
Get-AzureRmVM | Select Name,@{Name='OSType'; Expression={$_.StorageProfile.OSDisk.OSType}}
```

Name	OSType
MyUbuntu1610	Linux
MyWin2016VM	Windows

## Filter result using the Where-Object cmdlet

The `Where-Object` cmdlet allows you to filter the result based on any property value. In the following example, the filter selects only VMs that have the text "RGD" in their name.

```
Get-AzureRmVM | Where ResourceGroupName -like RGD* | Select ResourceGroupName,Name
```

ResourceGroupName	Name
RGDEM0001	KBDemo001VM
RGDEM0001	KBDemo0020

With the next example, the results will return the VMs that have the vmSize 'Standard\_DS1\_V2'.

```
Get-AzureRmVM | Where vmSize -eq Standard_DS1_V2
```

ResourceGroupName	Name	Location	VmSize	OsType	NIC	ProvisioningState
MYWESTEURG	MyUnbuntu1610	westeurope	Standard_DS1_v2	Linux	myunbuntu1610980	Succeeded
MYWESTEURG	MyWin2016VM	westeurope	Standard_DS1_v2	Windows	mywin2016vm880	Succeeded

# Formatting query results

4/10/2018 • 1 min to read • [Edit Online](#)

By default each PowerShell cmdlet has predefined formatting of output making it easy to read. PowerShell also provides the flexibility to adjust the output or convert the cmdlet output to a different format with the following cmdlets:

FORMATTING	CONVERSION
<code>Format-Custom</code>	<code>ConvertTo-Csv</code>
<code>Format-List</code>	<code>ConvertTo-Html</code>
<code>Format-Table</code>	<code>ConvertTo-Json</code>
<code>Format-Wide</code>	<code>ConvertTo-Xml</code>

## Formatting examples

In this example we get a list of Azure VMs in our default subscription. The `Get-AzureRmVM` command defaults output into a table format.

```
Get-AzureRmVM
```

ResourceGroupName	Name	Location	VmSize	OsType	NIC	ProvisioningState
MYWESTEURG	MyUbuntu1610	westeurope	Standard_DS1_v2	Linux	myubuntu1610980	Succeeded
MYWESTEURG	MyWin2016VM	westeurope	Standard_DS1_v2	Windows	mywin2016vm880	Succeeded

If you would like to limit the columns returned you can use the `Format-Table` cmdlet. In the following example we get the same list of virtual machines but restrict the output to just the name of the VM, the resource group, and the location of the VM. The `-AutoSize` parameter sizes the columns according to the size of the data.

```
Get-AzureRmVM | Format-Table Name,ResourceGroupName,Location -AutoSize
```

Name	ResourceGroupName	Location
MyUbuntu1610	MYWESTEURG	westeurope
MyWin2016VM	MYWESTEURG	westeurope

If you would prefer you can view information in a list format. The following example shows this using the `Format-List` cmdlet.

```
Get-AzureRmVM | Format-List Name,VmId,Location,ResourceGroupName
```

```
Name           : MyUbuntu1610
VmId           : 33422f9b-e339-4704-bad8-dbe094585496
Location       : westeurope
ResourceGroupName : MYWESTEURG
```

```
Name           : MyWin2016VM
VmId           : 4650c755-fc2b-4fc7-a5bc-298d5c00808f
Location       : westeurope
ResourceGroupName : MYWESTEURG
```

## Converting to other data types

PowerShell also offers multiple output format you can use to meet your needs. In the following example we use the `Select-Object` cmdlet to get attributes of the virtual machines in our subscription and convert the output to CSV format for easy import into a database or spreadsheet.

```
Get-AzureRmVM | Select-Object ResourceGroupName,Id,VmId,Name,Location,ProvisioningState | ConvertTo-Csv -
NoTypeInfo
```

```
"ResourceGroupName","Id","VmId","Name","Location","ProvisioningState"
"MYWESTUERG","/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTUERG/providers/Microsoft.Compute/virtualMachines/MyUbuntu1610","33422f9b-
e339-4704-bad8-dbe094585496","MyUbuntu1610","westeurope","Succeeded"
"MYWESTUERG","/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTUERG/providers/Microsoft.Compute/virtualMachines/MyWin2016VM","4650c755-
fc2b-4fc7-a5bc-298d5c00808f","MyWin2016VM","westeurope","Succeeded"
```

You can also convert the output into JSON format. The following example creates the same list of VMs but changes the output format to JSON.

```
Get-AzureRmVM | Select-Object ResourceGroupName,Id,VmId,Name,Location,ProvisioningState | ConvertTo-Json
```

```
[
  {
    "ResourceGroupName": "MYWESTEURG",
    "Id": "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTEURG/providers/Microsoft.Compute/virtualMachines/MyUbuntu1610",
    "VmId": "33422f9b-e339-4704-bad8-dbe094585496",
    "Name": "MyUbuntu1610",
    "Location": "westeurope",
    "ProvisioningState": "Succeeded"
  },
  {
    "ResourceGroupName": "MYWESTEURG",
    "Id": "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTEURG/providers/Microsoft.Compute/virtualMachines/MyWin2016VM",
    "VmId": "4650c755-fc2b-4fc7-a5bc-298d5c00808f",
    "Name": "MyWin2016VM",
    "Location": "westeurope",
    "ProvisioningState": "Succeeded"
  }
]
```



# Manage multiple Azure subscriptions

4/10/2018 • 1 min to read • [Edit Online](#)

If you are brand new to Azure, you probably only have a single subscription. But if you have been using Azure for a while, you may have created multiple Azure subscriptions. You can configure Azure PowerShell to execute commands against a particular subscription.

1. Get a list of all subscriptions in your account.

```
Get-AzureRmSubscription
```

```
Environment      : AzureCloud
Account          : username@contoso.com
TenantId         : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId    : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionName  : My Production Subscription
CurrentStorageAccount :
```

```
Environment      : AzureCloud
Account          : username@contoso.com
TenantId         : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId    : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionName  : My DevTest Subscription
CurrentStorageAccount :
```

```
Environment      : AzureCloud
Account          : username@contoso.com
TenantId         : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId    : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionName  : My Demos
CurrentStorageAccount :
```

2. Set the default.

```
Select-AzureRmSubscription -SubscriptionName "My Demos"
```

3. Verify the change by running the `Get-AzureRmContext` cmdlet.

```
Get-AzureRmContext
```

```
Environment      : AzureCloud
Account          : username@contoso.com
TenantId         : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId    : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionName  : My Demos
CurrentStorageAccount :
```

Once you set your default subscription, all subsequent Azure PowerShell commands run against this subscription.

# Using experimental Azure PowerShell modules

4/10/2018 • 3 min to read • [Edit Online](#)

With the emphasis on developer tools (especially CLIs) in Azure, the Azure PowerShell team is experimenting with many improvements to the Azure PowerShell experience.

## Experimentation methodology

To facilitate experimentation, we are creating new Azure PowerShell modules that implement existing Azure SDK functionality in new, easier to use ways. In most cases, the cmdlets exactly mirror existing cmdlets. However, the experimental cmdlets provide a shorthand notation and smarter default values that make it easier to create and manage Azure resources.

These modules can be installed side-by-side with existing Azure PowerShell modules. The cmdlet names have been shortened to provide shorter names and avoid name conflicts with existing, non-experimental cmdlets.

The experimental modules use the following naming convention: `AzureRM.*.Experiments`. This naming convention is similar to the naming of Preview modules: `AzureRM.*.Preview`. Preview modules differ from experimental modules. Preview modules implement new functionality of Azure services that is only available as a Preview offering. Preview modules replace existing Azure PowerShell modules and use the same cmdlet and parameter names.

## How to install an experimental module

Experimental modules are published to the PowerShell Gallery just like the existing Azure PowerShell modules. To see a list of experimental modules, run the following command:

```
Find-Module AzureRM.*.Experiments
```

Version	Name	Repository	Description
-----	----	-----	-----
1.0.25	AzureRM.Compute.Experiments	PSGallery	Azure Compute experiments for VM creation
1.0.0	AzureRM.Websites.Experiments	PSGallery	Create and deploy web applications using Azure App Services.

To install the experimental module, use the following commands from an elevated PowerShell session:

```
Install-Module AzureRM.Compute.Experiments
Install-Module AzureRM.Websites.Experiments
```

### Documentation and support

Documentation is included in the install package and is accessed using the `Get-Help` cmdlet. No official documentation is published for experimental modules.

We encourage you to test these modules. Your feedback allows us to improve usability and respond to your needs. However, being experimental, support for these modules is limited. Questions or problem reports can be submitted by creating an [issue](#) in the GitHub repository.

## Experiments and areas of improvement

These improvements were selected based on key differentiators in competing products. For example, Azure CLI 2.0 has made a point of basing commands on *scenarios* rather than *API surface area*. Azure CLI 2.0 use a number of smart defaults that make "getting started" scenarios easier for end users.

## Core improvements

The core improvements are regarded as "common sense", and little experimentation is needed to move forward in implementing these updates.

- Scenario-based Cmdlets - *\*All-* cmdlets should be designed around scenarios, not the Azure REST service.
- Shorter Names - Includes the names of cmdlets (for example, `New-AzureRmVm` => `New-AzVm` ) and the names of parameters (for example, `-ResourceGroupName` => `-Rg` ). Use aliases for compatibility with "old" cmdlets. Provide *backwards compatible* parameter sets.
- Smart Defaults - Create smart defaults to fill in "required" information. For example:
  - Resource Group
  - Location
  - Dependent resources

## Experimental improvements

The experimental improvements present a significant change that the team wants to validate via experimentation.

- Simple types - Create scenarios should move away from complex types and config objects. Simplify the configuration down to one or two parameters.
- "Smart Create" - All create scenarios implementing "Smart Create" would have *no* required parameters: all necessary information would be chosen by Azure PowerShell in an opinionated fashion.
- Gray Parameters - In many cases, some parameters could be "gray" or semi-optional. If the parameter is not specified, the user is asked if they want the parameter generated for them. It also makes sense for gray parameters to infer a value based on context with the user's consent. For example, it may make sense to have the gray parameter suggest the most recently used value.
- `-Auto` Switch - The `-Auto` switch would provide an "opt-in" way for users to *default all the things* while maintaining the integrity of required parameters in the mainline path.

## Feature-specific switches

For example, the "Create web app" scenario might have a `-Git` or `-AddRemote` switch that would automatically add an "azure" remote to an existing git repository.

- Settable Defaults - Users should have the ability to default certain ubiquitous parameters like `-ResourceGroupName` and `-Location` .
- Size Defaults - Resource "sizes" can be confusing to users since many Resource Providers use different names (for example, "Standard\_DS1\_v2" or "S1"). However, most users care more about cost. Therefore, it makes sense to define "universal" sizes based on a pricing schedule. Users can choose a specific size or let Azure PowerShell choose the *best option* based on the resource the budget.
- Output Format - Azure PowerShell currently returns `PSObject` s and there is little console output. Azure PowerShell may need to display some information to the user regarding the "smart defaults" used.

# Running cmdlets in parallel using PowerShell jobs

4/10/2018 • 3 min to read • [Edit Online](#)

PowerShell supports asynchronous action with [PowerShell Jobs](#). Azure PowerShell is heavily dependent on making, and waiting for, network calls to Azure. As a developer, you may often find yourself looking to make multiple non-blocking calls to Azure in a script, or you may find that you want to create Azure resources in the REPL without blocking the current session. To address these needs, Azure PowerShell provides first-class [PSJob](#) support.

## Context Persistence and PSJobs

PSJobs are run in separate processes, which means that information about your Azure connection must be properly shared with the jobs you create. Upon connecting your Azure account to your PowerShell session with

`Connect-AzureRmAccount`, you can pass the context to a job.

```
$creds = Get-Credential
$job = Start-Job { param($context,$vmadmin) New-AzureRmVM -Name MyVm -AzureRmContext $context -Credential
$vmadmin} -Arguments (Get-AzureRmContext),$creds
```

However, if you have chosen to have your context automatically saved with `Enable-AzureRmContextAutosave`, the context is automatically shared with any jobs you create.

```
Enable-AzureRmContextAutosave
$creds = Get-Credential
$job = Start-Job { param($vmadmin) New-AzureRmVM -Name MyVm -Credential $vmadmin} -Arguments $creds
```

## Automatic Jobs with `-AsJob`

As a convenience, Azure PowerShell also provides an `-AsJob` switch on some long-running cmdlets. The `-AsJob` switch makes creating PSJobs even easier.

```
$creds = Get-Credential
$job = New-AzureRmVM -Name MyVm -Credential $creds -AsJob
```

You can inspect the job and progress at any time with `Get-Job` and `Get-AzureRmVM`.

```
Get-Job $job
Get-AzureRmVM MyVm
```

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
1	Long Running Operation for 'New-AzureRmVM'	AzureLongRunningJob`1	Running	True	localhost	New-AzureRmVM

ResourceGroupName	Name	Location	VmSize	OsType	NIC	ProvisioningState	Zone
MyVm	MyVm	eastus	Standard_DS1_v2	Windows	MyVm	Creating	

Subsequently, upon completion, you can obtain the result of the job with `Receive-Job`.

## NOTE

`Receive-Job` returns the result from the cmdlet as if the `-AsJob` flag were not present. For example, the `Receive-Job` result of `Do-Action -AsJob` is of the same type as the result of `Do-Action`.

```
$vm = Receive-Job $job
$vm
```

```
ResourceGroupName      : MyVm
Id                     : /subscriptions/XXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MyVm/providers/Microsoft.Compute/virtualMachines/MyVm
VmId                   : dff1f79e-a8f7-4664-ab72-0ec28b9fbb5b
Name                   : MyVm
Type                   : Microsoft.Compute/virtualMachines
Location               : eastus
Tags                   : {}
HardwareProfile        : {VmSize}
NetworkProfile         : {NetworkInterfaces}
OSProfile              : {ComputerName, AdminUsername, WindowsConfiguration, Secrets}
ProvisioningState      : Succeeded
StorageProfile         : {ImageReference, OsDisk, DataDisks}
FullyQualifiedDomainName : myvmmyvm.eastus.cloudapp.azure.com
```

## Example Scenarios

Create multiple VMs at once.

```
$creds = Get-Credential
# Create 10 jobs.
for($k = 0; $k -lt 10; $k++) {
    New-AzureRmVm -Name MyVm$k -Credential $creds -AsJob
}

# Get all jobs and wait on them.
Get-Job | Wait-Job
"All jobs completed"
Get-AzureRmVM
```

In this example, the `Wait-Job` cmdlet causes the script to pause while jobs run. The script continues executing once all of the jobs have completed. This allows you to create several jobs running in parallel then wait for completion before continuing.

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
--	----	-----	----	-----	-----	-----
2	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
3	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
4	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
5	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
6	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
7	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
8	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
9	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
10	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
11	Long Running...	AzureLongRun...	Running	True	localhost	New-AzureRmVM
2	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
3	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
4	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
5	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
6	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
7	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
8	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
9	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
10	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM
11	Long Running...	AzureLongRun...	Completed	True	localhost	New-AzureRmVM

All Jobs completed.

ResourceGroupName	Name	Location	VmSize	OsType	NIC	ProvisioningState	Zone
-----	----	-----	-----	-----	---	-----	----
MYVM	MyVm	eastus	Standard_DS1_v2	Windows	MyVm	Succeeded	
MYVM0	MyVm0	eastus	Standard_DS1_v2	Windows	MyVm0	Succeeded	
MYVM1	MyVm1	eastus	Standard_DS1_v2	Windows	MyVm1	Succeeded	
MYVM2	MyVm2	eastus	Standard_DS1_v2	Windows	MyVm2	Succeeded	
MYVM3	MyVm3	eastus	Standard_DS1_v2	Windows	MyVm3	Succeeded	
MYVM4	MyVm4	eastus	Standard_DS1_v2	Windows	MyVm4	Succeeded	
MYVM5	MyVm5	eastus	Standard_DS1_v2	Windows	MyVm5	Succeeded	
MYVM6	MyVm6	eastus	Standard_DS1_v2	Windows	MyVm6	Succeeded	
MYVM7	MyVm7	eastus	Standard_DS1_v2	Windows	MyVm7	Succeeded	
MYVM8	MyVm8	eastus	Standard_DS1_v2	Windows	MyVm8	Succeeded	
MYVM9	MyVm9	eastus	Standard_DS1_v2	Windows	MyVm9	Succeeded	

# Release notes

4/10/2018 • 27 min to read • [Edit Online](#)

This is a list of changes made to Azure PowerShell in this release.

## Azure PowerShell 5.7.0

Azure PowerShell 5.7.0 Installer: [link](#)

Gallery Module for ARM Cmdlets: [link](#)

To install `AzureRM` from the PowerShell Gallery, run the following command:

```
Install-Module -Name AzureRM -Repository PSGallery -Force
```

To update from an older version of `AzureRM`, run the following command:

```
Update-Module -Name AzureRM
```

## Changes Since Last Release

### General

- Updated to the latest version of the Azure ClientRuntime

### Azure.Storage

- Fix the issue that upload Blob and upload File cmdlets fail on FIPS policy enabled machines
  - Set-AzureStorageBlobContent
  - Set-AzureStorageFileContent

### AzureRM.Billing

- New Cmdlet Get-AzureRmEnrollmentAccount
  - cmdlet to retrieve enrollment accounts

### AzureRM.CognitiveServices

- Integrate with Cognitive Services Management SDK version 4.0.0.
- Add Get-AzureRmCognitiveServicesAccountUsage operation.

### AzureRM.Compute

- `Get-AzureRmVmssDiskEncryptionStatus` supports encryption status at data disk level
- `Get-AzureRmVmssVmDiskEncryptionStatus` supports encryption status at data disk level
- Update for Zone Resilient
- 'New-AzureRmVm' and 'New-AzureRmVmss' (simple parameter set) support availability zones.

### AzureRM.ContainerRegistry

- Decouple reliance on Commands.Resources.Rest and ARM/Storage SDKs.

### AzureRM.DataLakeStore

- Add debug functionality
- Update the version of the ADLS dataplane SDK to 1.1.2
- Export-AzureRmDataLakeStoreItem - Deprecated parameters PerFileThreadCount, ConcurrentFileCount and

introduced parameter Concurrency

- Import-AzureRMDDataLakeStoreItem - Deprecated parameters PerFileThreadCount, ConcurrentFileCount and introduced parameter Concurrency
- Get-AzureRMDDataLakeStoreItemContent - Fixed the tail behavior for contents greater than 4MB
- Set-AzureRMDDataLakeStoreItemExpiry - Introduced new parameter set SetRelativeExpiry for setting relative expiration time
- Remove-AzureRmDataLakeStoreItem - Deprecated parameter Clean.

#### **AzureRM.EventHub**

- Fixed Alternamename in New-AzureRmEventHubGeoDRConfiguration

#### **AzureRM.KeyVault**

- Updated cmdlets to include piping scenarios
- Add deprecation messages for upcoming breaking change release

#### **AzureRM.Network**

- Fix error message with Network cmdlets

#### **AzureRM.ServiceBus**

- Added 'properties' in CorrelationFilter of Rules to support customproperties
- updated New-AzureRmServiceBusGeoDRConfiguration help and fixed Rules cmdlet output
- Fixed auto-forward properties in New-AzureRmServiceBusQueue and New-AzureRmServiceBusSubscription cmdlet

#### **AzureRM.Sql**

- Add new cmdlet 'Stop-AzureRmSqlElasticPoolActivity' to support canceling the asynchronous operations on elastic pool
- Update the response for cmdlets Get-AzureRmSqlDatabaseActivity and Get-AzureRmSqlElasticPoolActivity to reflect more information in the response

Changes since last release: <https://github.com/Azure/azure-powershell/compare/v5.6.0-March2018...v5.7.0-April2018>

## 5.6.0 - March 2018

#### **General**

- Fix issue with Default Resource Group in CloudShell
- Fix issue where incorrect startup scripts were being executed during module import

#### **AzureRM.Profile**

- Enable MSI authentication in unsupported scenarios
- Add support for user-defined Managed Service Identity

#### **AzureRM.AnalysisServices**

- Fixed issue with cleaning up scripts in build

#### **AzureRM.Cdn**

- Fixed issue with cleaning up scripts in build

#### **AzureRM.Compute**

- 'New-AzureRmVM' and 'New-AzureRmVMSS' support data disks.
- 'New-AzureRmVM' and 'New-AzureRmVMSS' support custom image by name or by id.
- Log analytic feature
  - Added 'Export-AzureRmLogAnalyticRequestRateByInterval' cmdlet
  - Added 'Export-AzureRmLogAnalyticThrottledRequests' cmdlet



#### **AzureRM.ContainerInstance**

- Fix parameter sets issue for container registry and azure file volume mount

#### **AzureRM.DataFactoryV2**

- Updated the ADF .Net SDK to version 0.6.0-preview containing the following changes:
  - Added new AzureDatabricks LinkedService and DatabricksNotebook Activity
  - Added headNodeSize and dataNodeSize properties in HDInsightOnDemand LinkedService
  - Added LinkedService, Dataset, CopySource for SalesforceMarketingCloud
  - Added support for SecureOutput on all activities
  - Added new BatchCount property on ForEach activity which control how many concurrent activities to run
  - Added new Filter Activity
  - Added Linked Service Parameters support

#### **AzureRM.Dns**

- Support for Private DNS Zones (Public Preview)
  - Adds ability to create DNS zones that are visible only to the associated virtual networks

#### **AzureRM.Network**

- Updating model types for compatibility with DNS cmdlets.

#### **AzureRM.RecoveryServices.SiteRecovery**

- Changes for ASR Azure to Azure Site Recovery (cmdlets are currently supporting operations for Enterprise to Enterprise, Enterprise to Azure, HyperV to Azure, VMware to Azure)
  - New-AzureRmRecoveryServicesAsrProtectionContainer
  - New-AzureRmRecoveryServicesAsrAzureToAzureDiskReplicationConfig
  - Remove-AzureRmRecoveryServicesAsrProtectionContainer
  - Update-AzureRmRecoveryServicesAsrProtectionDirection

#### **AzureRM.Storage**

- Obsolete following parameters in new and set Storage Account cmdlets: EnableEncryptionService and DisableEncryptionService, since Encryption at Rest is enabled by default and can't be disabled.
  - New-AzureRmStorageAccount
  - Set-AzureRmStorageAccount

#### **AzureRM.Websites**

- Fixed the help for Remove-AzureRmWebAppSlot

## **5.5.0 - March 2018**

#### **AzureRM.Profile**

- Fixed issue with importing aliases
- Load version 10.0.3 of Newtonsoft.Json side-by-side with version 6.0.8

#### **Azure.Storage**

- Support Soft-Delete feature
  - Enable-AzureStorageDeleteRetentionPolicy
  - Disable-AzureStorageDeleteRetentionPolicy
  - Get-AzureStorageBlob

#### **AzureRM.AnalysisServices**

- Fixed issue with importing aliases
- Add support of firewall and query scaleout feature, as well as support of 2017-08-01 api version.

#### **AzureRM.Automation**

- Fixed issue with importing aliases

#### **AzureRM.Cdn**

- Fixed issue with importing aliases

#### **AzureRM.CognitiveServices**

- Update notice.txt and notice message.

#### **AzureRM.Compute**

- 'New-AzureRmVMSS' prints connection strings in verbose mode.
- 'New-AzureRmVmss' supports public IP address, load balancing rules, inbound NAT rules.
- WriteAccelerator feature
  - Added WriteAccelerator switch parameter to the following cmdlets: Set-AzureRmVMOSDisk Set-AzureRmVMDataDisk Add-AzureRmVMDataDisk Add-AzureRmVmssDataDisk
  - Added OsDiskWriteAccelerator switch parameter to the following cmdlet: Set-AzureRmVmssStorageProfile.
  - Added OsDiskWriteAccelerator Boolean parameter to the following cmdlets: Update-AzureRmVM Update-AzureRmVmss

#### **AzureRM.DataFactories**

- Fix credential encryption issue that caused no meaningful error for some encryption operations
- Enable integration runtime to be shared across data factory

#### **AzureRM.DataFactoryV2**

- Add parameter "SetupScriptContainerSasUri" and "Edition" for "Set-AzureRmDataFactoryV2IntegrationRuntime" cmd to enable custom setup and edition selection functionality
- Fix credential encryption issue that caused no meaningful error for some encryption operations.
- Enable integration runtime to be shared across data factory

#### **AzureRM.HDInsight**

- Fixed issue with importing aliases

#### **AzureRM.KeyVault**

- Fixed example for Set-AzureRmKeyVaultAccessPolicy

#### **AzureRM.Network**

- Fixed issue with importing aliases

#### **AzureRM.Operationallnsights**

- Fixed issue with importing aliases

#### **AzureRM.RecoveryServices**

- Fixed issue with importing aliases

#### **AzureRM.RecoveryServices.SiteRecovery**

- Fixed issue with importing aliases

#### **AzureRM.Resources**

- Fixed issue with importing aliases

#### **AzureRM.ServiceBus**

- Added EnableBatchedOperations property to Queue
- Added DeadLetteringOnFilterEvaluationExceptions property to Subscriptions

#### **AzureRM.ServiceFabric**

- Service Fabric cmdlet refresh
  - Updated ARM templates

- Failed operations no longer rollback
- Add-AzureRmServiceFabricNodeType
  - VMs default to managed disks
  - Existing VMSS subnet used
  - All operations are idempotent
- Remove-AzureRmServiceFabricNodeType cleans up partially created VMSS and/or cluster node types
- Fixed output of PSCluster object for complex property types

#### **AzureRM.Sql**

- Fixed issue with importing aliases
- Get-AzureRmSqlServer, New-AzureRmSqlServer, and Remove-AzureRmSqlServer response now includes FullyQualifiedDomainName property.

#### **AzureRM.Websites**

- Fixed issue with importing aliases
- New-AzureRMWebApp - added parameter set for simplified WebApp creation, with local git repository support.

## 5.4.0 - February 2018

#### **AzureRM.Automation**

- Added alias from New-AzureRmAutomationModule to Import-AzureRmAutomationModule

#### **AzureRM.Compute**

- Fix ErrorAction issue for some of Get cmdlets.

#### **AzureRM.ContainerInstance**

- Apply Azure Container Instance SDK 2018-02-01
  - Support DNS name label

#### **AzureRM.DevTestLabs**

- Fixed all of the GET cmdlets which previously weren't working.

#### **AzureRM.EventHub**

- Fix bug in Get-AzureRmEventHubGeoDRConfiguration help

#### **AzureRM.Network**

- Added cmdlet to create a new connection monitor
  - New-AzureRmNetworkWatcherConnectionMonitor
- Added cmdlet to update a connection monitor
  - Set-AzureRmNetworkWatcherConnectionMonitor
- Added cmdlet to get connection monitor or connection monitor list
  - Get-AzureRmNetworkWatcherConnectionMonitor
- Added cmdlet to query connection monitor
  - Get-AzureRmNetworkWatcherConnectionMonitorReport
- Added cmdlet to start connection monitor
  - Start-AzureRmNetworkWatcherConnectionMonitor
- Added cmdlet to stop connection monitor
  - Stop-AzureRmNetworkWatcherConnectionMonitor
- Added cmdlet to remove connection monitor
  - Remove-AzureRmNetworkWatcherConnectionMonitor
- Updated Set-AzureRmApplicationGatewayBackendAddressPool documentation to remove deprecated example
- Added EnableHttp2 flag to Application Gateway
  - Updated New-AzureRmApplicationGateway: Added optional parameter -EnableHttp2

- Add IpTags to PublicIpAddress
  - Updated New-AzureRmPublicIpAddress: Added IpTags
  - New-AzureRmPublicIpTag to add Iptag
- Add DisableBgpRoutePropagation property in RouteTable and effectiveRoute.

#### AzureRM.Resources

- Register-AzureRmProviderFeature: Added missing example in the docs
- Register-AzureRmResourceProvider: Added missing example in the docs

#### AzureRM.Storage

- Obsolete following parameters in new and set Storage Account cmdlets: EnableEncryptionService and DisableEncryptionService, since Encryption at Rest is enabled by default and can't be disabled.
  - New-AzureRmStorageAccount
  - Set-AzureRmStorageAccount

## 5.3.0 - February 2018

#### AzureRM.Profile

- Added deprecation warning for PowerShell 3 and 4
- `Add-AzureRmAccount` has been renamed as `Connect-AzureRmAccount`; an alias has been added for the old cmdlet name, and other aliases ( `Login-AzAccount` and `Login-AzureRmAccount` ) have been redirected to the new cmdlet name.
- `Remove-AzureRmAccount` has been renamed as `Disconnect-AzureRmAccount`; an alias has been added for the old cmdlet name, and other aliases ( `Logout-AzAccount` and `Logout-AzureRmAccount` ) have been redirected to the new cmdlet name.
- Corrected Resource Strings to use `Connect-AzureRmAccount` instead of `Login-AzureRmAccount`
- `Add-AzureRmEnvironment` and `Set-AzureRmEnvironment`
  - Added `-AzureOperationalInsightsEndpoint` and `-AzureOperationalInsightsEndpointResourceId` as parameters for use with OperationalInsights data plane RP.

#### AzureRM.AnalysisServices

- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`

#### AzureRM.Compute

- Added `-AvailabilitySetName` parameter to the simplified parameterset of `New-AzureRmVM`.
- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`
- User assigned identity support for VM and VM scale set
  - `-IdentityType` and `-IdentityId` parameters are added to `New-AzureRmVMConfig`, `New-AzureRmVmssConfig`, `Update-AzureRmVM` and `Update-AzureRmVmss`
- Added `-EnableIPForwarding` parameter to `Add-AzureRmVmssNetworkInterfaceConfig`
- Added `-Priority` parameter to `New-AzureRmVmssConfig`

#### AzureRM.DataLakeAnalytics

- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`

#### AzureRM.DataLakeStore

- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`
- Corrected the error message of `Test-AzureRmDataLakeStoreAccount` when running this cmdlet without having logged in with `Login-AzureRmAccount`

#### AzureRM.EventGrid

- Updated to use the 2018-01-01 API version.

## AzureRM.EventHub

- Added below new commands for Geo Disaster Recovery operations.
  - Creating a new Alias (Disaster Recovery configuration):
    - `New-AzureRmEventHubGeoDRConfiguration`
  - Retrieve Alias (Disaster Recovery configuration) :
    - `Get-AzureRmEventHubGeoDRConfiguration`
  - Disabling the Disaster Recovery and stops replicating changes from primary to secondary namespaces
    - `Set-AzureRmEventHubGeoDRConfigurationBreakPair`
  - Invoking Disaster Recovery failover and reconfigure the alias to point to the secondary namespace
    - `Set-AzureRmEventHubGeoDRConfigurationFailOver`
  - Deleting an Alias(Disaster Recovery configuration)
    - `Remove-AzureRmEventHubGeoDRConfiguration`
- Added below new commands for checking the Namespace Name and GeoDr Configuration Name - Alias availability.
  - Check the Availability of Namespace name or Alias(Disaster Recovery configuration) name:
    - `Test-AzureRmEventHubName`

## AzureRM.Insights

- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`

## AzureRM.KeyVault

- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`

## AzureRM.Network

- Fix overwrite message 'Are you sure you want to overwriteresource'

## AzureRM.Operationallnsights

- Added support for V2 API querying via `Invoke-AzureRmOperationalInsightsQuery`. See <https://dev.loganalytics.io/> for more info on the new API.

## AzureRM.Resources

- `Get-AzureRmADServicePrincipal` : Removed `-ServicePrincipalName` from the default Empty parameter set as it was redundant with the SPN parameter set

## AzureRM.ServiceBus

- Added functionality fix for `Remove-AzureRmServiceBusRule` and `Get-AzureRmServiceBusKey`
- Added below new commandlets for Geo Disaster Recovery operations.
  - Creating a new Alias (Disaster Recovery configuration):
    - `New-AzureRmServiceBusDRConfigurations`
  - Retrieve Alias (Disaster Recovery configuration) :
    - `Get-AzureRmServiceBusDRConfigurations`
  - Disabling the Disaster Recovery and stops replicating changes from primary to secondary namespaces
    - `Set-AzureRmServiceBusDRConfigurationsBreakPairing`
  - Invoking Disaster Recovery failover and reconfigure the alias to point to the secondary namespace
    - `Set-AzureRmServiceBusDRConfigurationsFailOver`
  - Deleting an Alias(Disaster Recovery configuration)
    - `Remove-AzureRmServiceBusDRConfigurations`
- Updated `Test-AzureRmServiceBusName` commandlets to support Geo Disaster Recovery - Alias name check

availability operations.

- Check the Availability of Namespace name or Alias(Disaster Recovery configuration) name:
  - `Test-AzureRmServiceBusName`

### **AzureRM.UsageAggregates**

- Corrected usage of `Login-AzureRmAccount` to use `Connect-AzureRmAccount`

## **5.2.0 - January 2018**

### **AzureRM.Profile**

- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Add-AzureRmAccount
  - Added -MSI login for authenticationg using the credentials of the Managed Service Identity of the current VM / Service
  - Fixed KeyVault Authentication when logging in with user-provided access tokens

### **Azure.Storage**

- Add cmdlets to get and set Storage service properties
  - `Get-AzureStorageServiceProperty`
  - `Update-AzureStorageServiceProperty`

### **AzureRM.AnalysisServices**

- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

### **AzureRM.ApiManagement**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for `New-AzureRmApiManagementProperty`, `Set-AzureRmApiManagementProperty`, and `New-AzureRmApiManagement`

### **AzureRM.ApplicationInsights**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

### **AzureRM.Automation**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for `Set-AzureRmAutomationRunbook`

### **AzureRM.Backup**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

### **AzureRM.Batch**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

### **AzureRM.Cdn**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmCdnEndpoint and New-AzureRmCdnProfile

#### **AzureRM.CognitiveServices**

- Integrate with Cognitive Services Management SDK version 3.0.0.

#### **AzureRM.Compute**

- Added simplified parameter set to New-AzureRmVmss, which creates a Virtual Machine Scale Set and all required resources using smart defaults
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmVm and Update-AzureRmVm
- Fixed Get-AzureRmComputeResourceSku cmdlet when Zone is included in restriction.
- Updated Diagnostics Agent configuration schema for Azure Monitor sink support.

#### **AzureRM.ContainerInstance**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.ContainerRegistry**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.DataFactories**

- Enabled Azure Key Vault support for all data store linked services
- Added license type property for Azure SSIS integration runtime
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmDataFactory

#### **AzureRM.DataFactoryV2**

- Enabled Azure Key Vault support for all data store linked services
- Added license type property for Azure SSIS integration runtime
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Add parameter "LicenseType" for "Set-AzureRmDataFactoryV2IntegrationRuntime" cmd to enable AHUB functionality

#### **AzureRM.DataLakeAnalytics**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmDataLakeAnalyticsAccount and Set-AzureRmDataLakeAnalyticsAccount

#### **AzureRM.DataLakeStore**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations

- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmDataLakeStoreAccount and Set-AzureRmDataLakeStoreAccount

#### **AzureRM.DevTestLabs**

- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.Dns**

- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.EventGrid**

- Added the following new cmdlet:
  - Update-AzureRmEventGridSubscription
    - Update the properties of an Event Grid event subscription.
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.EventHub**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.HDInsight**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.Insights**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.IotHub**

- Add Certificate support for IoT Hub cmdlets
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.KeyVault**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Added -AsJob support for long-running KeyVault cmdlets. Allows selected cmdlets to run in the background and return a job to track and control progress.
  - Affected cmdlet is: Remove-AzureRmKeyVault
- Fixed bug in Set-AzureRmKeyVaultAccessPolicy where the AAD filter was setting SPN to the provided UPN, rather than setting the UPN
  - See the following issue for more information: <https://github.com/Azure/azure-powershell/issues/5201>

#### **AzureRM.LogicApp**



- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.MachineLearning**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for Update-AzureRmMLCommitmentPlan

#### **AzureRM.MachineLearningCompute**

- Add IncludeAllResources parameter to Remove-AzureRmMLOpCluster cmdlet
  - Using this switch parameter will remove all resources that were created with the cluster originally
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.Media**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for Set-AzureRmMediaService and New-AzureRmMediaService

#### **AzureRM.Network**

- Added -AsJob support for long-running Network cmdlets. Allows selected cmdlets to run in the background and return a job to track and control progress.
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.NotificationHubs**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmNotificationHubsNamespace and Set-AzureRmNotificationHubsNamespace

#### **AzureRM.Operationallnsights**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmOperationallnsightsSavedSearch, Set-AzureRmOperationallnsightsSavedSearch, New-AzureRmOperationallnsightsWorkspace, and Set-AzureRmOperationallnsightsWorkspace

#### **AzureRM.PowerBIEmbedded**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.RecoveryServices**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.RecoveryServices.Backup**

- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Added -UseOriginalStorageAccount option to the Restore-AzureRmRecoveryServicesBackupItem cmdlet.
  - Enabling this flag results in restoring disks to their original storage accounts which allows users to maintain the configuration of restored VM as close to the original VMs as possible.
  - It also helps in improving the performance of the restore operation.

#### **AzureRM.RedisCache**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Added 3 new cmdlets for firewall rules
- Added 3 new cmdlets for geo replication
- Added support for zones and tags
- Make ResourceGroup as optional whenever possible.

#### **AzureRM.Relay**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.Resources**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Added -AsJob support for long-running Resources cmdlets. Allows selected cmdlets to run in the background and return a job to track and control progress.
- Added alias from Get-AzureRmProviderOperation to Get-AzureRmResourceProviderAction to conform with naming conventions

#### **AzureRM.Scheduler**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.ServerManagement**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Obsoleted -Tags in favor of -Tag for New-AzureRmServerManagementNode and New-AzureRmServerManagementGateway

#### **AzureRM.ServiceBus**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.ServiceFabric**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.SiteRecovery**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.Sql**

- Update the Auditing commands parameters description
- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Added -AsJob parameter to long running cmdlets
- Obsoleted -DatabaseName parameter from Get-AzureRmSqlServiceObjective

#### **AzureRM.Storage**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Fix a null reference issue of run cmdlet New-AzureRMStorageAccount with parameter -EnableEncryptionService None
- Added -AsJob support for long-running Storage cmdlets. Allows selected cmdlets to run in the background and return a job to track and control progress.
  - Affected cmdlets are New-, Remove-, Add-, and Update- for Storage Account and Storage Account Network Rule.

#### **AzureRM.StreamAnalytics**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.TrafficManager**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription

#### **AzureRM.Websites**

- Added Location Completer to -Location parameters allowing tab completion through valid Locations
- Added ResourceGroup Completer to -ResourceGroup parameters allowing tab completion through resource groups in current subscription
- Added -AsJob support for long-running Websites cmdlets. Allows selected cmdlets to run in the background and return a job to track and control progress.
  - Affected cmdlets are New-, Remove-, Add-, and Set- for WebApps, AppServicePlan and Slots

## **2017.12.8 Version 5.1.1**

- AnalysisServices
  - Change validate set of location to dynamic lookup so that all clouds are supported.
- Automation
  - Update to Import-AzureRMAutomationRunbook
    - Support is now being provided for Python2 runbooks
- Batch
  - Fixed a bug where account operations without a resource group failed to auto-detect the resource group
- Compute
  - Get-AzureRmComputeResourceSku shows zone information.

- Update Disable-AzureRmVmssDiskEncryption to fix issue <https://github.com/Azure/azure-powershell/issues/5038>
- Added -AsJob support for long-running Compute cmdlets. Allows selected cmdlets to run in the background and return a job to track and control progress.
  - Affected cmdlets include: New-, Update-, Set-, Remove-, Start-, Restart-, Stop- cmdlets for Virtual Machines and Virtual Machine Scale Sets
  - Added simplified parameter set to New-AzureRmVM, which creates a Virtual Machine and all required resources using smart defaults
- ContainerInstance
  - Apply Azure Container Instance SDK 2017-10-01
    - Support container run-to-completion
    - Support Azure File volume mount
    - Support opening multiple ports for public IP
- ContainerRegistry
  - New cmdlets for geo-replication and webhooks
    - Get/New/Remove-AzureRmContainerRegistryReplication
    - Get/New/Remove/Test/Update-AzureRmContainerRegistryWebhook
- DataFactories
  - Credential encryption functionality now works with both "Remote Access" enabled (Over Network) and "Remote Access" disabled (Local Machine).
- DataFactoryV2
  - Added two new cmdlets: Update-AzureRmDataFactoryV2 and Stop-AzureRmDataFactoryV2PipelineRun
- DataLakeAnalytics
  - Added a parameter called ScriptParameter to Submit-AzureRmDataLakeAnalyticsJob
    - Detailed information about ScriptParameter can be found using Get-Help on Submit-AzureRmDataLakeAnalyticsJob
  - For New-AzureRmDataLakeAnalyticsAccount, changed the parameter MaxDegreeOfParallelism to MaxAnalyticsUnits
    - Added an alias for the parameter MaxAnalyticsUnits: MaxDegreeOfParallelism
  - For New-AzureRmDataLakeAnalyticsComputePolicy, changed the parameter MaxDegreeOfParallelismPerJob to MaxAnalyticsUnitsPerJob
    - Added an alias for the parameter MaxAnalyticsUnitsPerJob: MaxDegreeOfParallelismPerJob
  - For Set-AzureRmDataLakeAnalyticsAccount, changed the parameter MaxDegreeOfParallelism to MaxAnalyticsUnits
    - Added an alias for the parameter MaxAnalyticsUnits: MaxDegreeOfParallelism
  - For Submit-AzureRmDataLakeAnalyticsJob, changed the parameter DegreeOfParallelism to AnalyticsUnits
    - Added an alias for the parameter AnalyticsUnits: DegreeOfParallelism
  - For Update-AzureRmDataLakeAnalyticsComputePolicy, changed the parameter MaxDegreeOfParallelismPerJob to MaxAnalyticsUnitsPerJob
    - Added an alias for the parameter MaxAnalyticsUnitsPerJob: MaxDegreeOfParallelismPerJob
- MachineLearningCompute
  - Add Set-AzureRmMLOpCluster
    - Update a cluster's agent count or SSL configuration
  - Orchestrator properties are optional
    - The service will create a service principal if not provided, so the orchestrator properties are now optional
- PowerBIEmbedded

- Add support for Power BI Embedded Capacity cmdlets
- New Cmdlet Get-AzureRmPowerBIEmbeddedCapacity - Gets the details of a PowerBI Embedded Capacity.
- New Cmdlet New-AzureRmPowerBIEmbeddedCapacity - Creates a new PowerBI Embedded Capacity
- New Cmdlet Remove-AzureRmPowerBIEmbeddedCapacity - Deletes an instance of PowerBI Embedded Capacity
- New Cmdlet Resume-AzureRmPowerBIEmbeddedCapacity - Resumes an instance of PowerBI Embedded Capacity
- New Cmdlet Suspend-AzureRmPowerBIEmbeddedCapacity - Suspends an instance of PowerBI Embedded Capacity
- New Cmdlet Test-AzureRmPowerBIEmbeddedCapacity - Tests the existence of an instance of PowerBI Embedded Capacity
- New Cmdlet Update-AzureRmPowerBIEmbeddedCapacity - Modifies an instance of PowerBI Embedded Capacity
- Profile
  - Updated USGovernmentActiveDirectoryEndpoint to <https://login.microsoftonline.us/>
    - For more information about the Azure Government endpoint mappings, please see the following: <https://docs.microsoft.com/en-us/azure/azure-government/documentation-government-developer-guide#endpoint-mapping>
    - Added -AsJob support for cmdlets, enabling selected cmdlets to execute in the background and return a job to track and control progress
    - Added -AsJob parameter to Get-AzureRmSubscription cmdlet
- RecoveryServices.Backup
  - Fixed bug - Get-AzureRmRecoveryServicesBackupItem should do case insensitive comparison for container name filter.
  - Fixed bug - AzureVmItem now has a property that shows the last time a backup operation has happened - LastBackupTime.
- Resources
  - Fixed issue where Get-AzureRmRoleAssignment would result in a assignments without roledefiniton name for custom roles
    - Users can now use Get-AzureRmRoleAssignment with assignments having roledefinition names irrespective of the type of role
  - Fixed issue where Set-AzureRmRoleRoleDefinition used to throw RD not found error when there was a new scope in assignablescopes
    - Users can now use Set-AzureRmRoleRoleDefinition with assignable scopes including new scopes irrespective of the position of the scope
  - Allow scopes to end with "/"
    - Users can now use RoleDefinition and RoleAssignment commandlets with scopes ending with "/" ,consistent with API and CLI
  - Allow users to create RoleAssignment using delegation flag
    - Users can now use New-AzureRmRoleAssignment with an option of adding the delegation flag
  - Fix RoleAssignment get to respect the scope parameter
  - Add an alias for ServicePrincipalName in the New-AzureRmRoleAssignment Commandlet
    - Users can now use the ApplicationId instead of the ServicePrincipalName when using the New-AzureRmRoleAssignment commandlet
- SiteRecovery
  - Add deprecation warnings for all cmdlets in this module in preparation for the next breaking change release.

- Please see the upcoming breaking changes guide for more information on how to migrate your cmdlets from AzureRM.
- Sql
  - Added ability to rename database using Set-AzureRmSqlDatabase
  - Fixed issue <https://github.com/Azure/azure-powershell/issues/4974>
    - Providing invalid AUDIT\_CHANGED\_GROUP value for auditing cmdlets no longer throws an error and will be removed in an upcoming release.
  - Fixed issue <https://github.com/Azure/azure-powershell/issues/5046>
    - AuditAction parameter in auditing cmdlets is no longer being ignored
  - Fixed an issue in Auditing cmdlets when 'Secondary' StorageKeyType is provided
    - When setting blob auditing, the primary storage account key was used instead of the secondary key when providing 'Secondary' value for StorageKeyType parameter.
  - Changing the wording for confirmation message from Set-AzureRmSqlServerTransparentDataEncryptionProtector
- Azure (RDFE)
  - Removed all RemoteApp Cmdles
- Azure.Storage
  - Upgrade to Azure Storage Client Library 8.6.0 and Azure Storage DataMovement Library 0.6.5

## 2017.11.10 Version 5.0.1

- Fixed assembly loading issue that caused some cmdlets to fail when executing in the following modules:
  - AzureRM.ApiManagement
  - AzureRM.Backup
  - AzureRM.Batch
  - AzureRM.Compute
  - AzureRM.DataFactories
  - AzureRM.HDInsight
  - AzureRM.KeyVault
  - AzureRM.RecoveryServices
  - AzureRM.RecoveryServices.Backup
  - AzureRM.RecoveryServices.SiteRecovery
  - AzureRM.RedisCache
  - AzureRM.SiteRecovery
  - AzureRM.Sql
  - AzureRM.Storage
  - AzureRM.StreamAnalytics

## 2017.11.8 - Version 5.0.0

- NOTE: This is a breaking change release. Please see the migration guide (<https://aka.ms/azps-migration-guide>) for a full list of introduced breaking changes.
- All cmdlets in AzureRM now support online help
  - Run Get-Help with the -Online parameter to open the online help in your default Internet browser
- AnalysisServices
  - Fixed Synchronize-AzureAsInstance command to work with new AsAzure REST API for sync
- ApiManagement
  - Please see the migration guide for breaking changes made to ApiManagement this release

- Updated Cmdlet Get-AzureRmApiManagementUser to fix issue <https://github.com/Azure/azure-powershell/issues/4510>
- Updated Cmdlet New-AzureRmApiManagementApi to create Api with Empty Path <https://github.com/Azure/azure-powershell/issues/4069>
- ApplicationInsights
  - Add commands to get/create/remove applicaiton insights resource
    - Get-AzureRmApplicationInsights
    - New-AzureRmApplicationInsights
    - Remove-AzureRmApplicationInsights
  - Add commands to get/update pricing/daily cap of applicaiton insights resource
    - Get-AzureRmApplicationInsights -IncludeDailyCap
    - Set-AzureRmApplicationInsightsPricingPlan
    - Set-AzureRmApplicationInsightsDailyCap
  - Add commands to get/create/update/remove continuous export of applicaiton insights resource
    - Get-AzureRmApplicationInsightsContinuousExport
    - Set-AzureRmApplicationInsightsContinuousExport
    - New-AzureRmApplicationInsightsContinuousExport
    - Remove-AzureRmApplicationInsightsContinuousExport
  - Add commands to get/create/remove api keys of applicaiton insights resoruce
    - Get-AzureRmApplicationInsightsApiKey
    - New-AzureRmApplicationInsightsApiKey
    - Remove-AzureRmApplicationInsightsApiKey
- AzureBatch
  - Added new parameters to `New-AzureRmBatchAccount` .
    - `PoolAllocationMode` : The allocation mode to use for creating pools in the Batch account. To create a Batch account which allocates pool nodes in the user's subscription, set this to `UserSubscription` .
    - `KeyVaultId` : The resource ID of the Azure key vault associated with the Batch account.
    - `KeyVaultUrl` : The URL of the Azure key vault associated with the Batch account.
  - Updated parameters to `New-AzureBatchTask` .
    - Removed the `RunElevated` switch. The `UserIdentity` parameter has been added to replace `RunElevated` , and the equivalent behavior can be achieved by constructing a `PSUserIdentity` as shown below:
      - `$autoUser = New-Object Microsoft.Azure.Commands.Batch.Models.PSAutoUserSpecification -ArgumentList @("Task", "Admin")`
      - `$userIdentity = New-Object Microsoft.Azure.Commands.Batch.Models.PSUserIdentity $autoUser`
    - Added the `AuthenticationTokenSettings` parameter. This parameter allows you to request the Batch service provide an authentication token to the task when it runs, avoiding the need to pass Batch account keys to the task in order to issue requests to the Batch service.
    - Added the `ContainerSettings` parameter.
      - This parameter allows you to request the Batch service run the task inside a container.
    - Added the `OutputFiles` parameter.
      - This parameter allows you to configure the task to upload files to Azure Storage after it has finished.
  - Updated parameters to `New-AzureBatchPool` .

- Added the `UserAccounts` parameter.
  - This parameter defines user accounts created on each node in the pool.
- Added `TargetLowPriorityComputeNodes` and renamed `TargetDedicated` to `TargetDedicatedComputeNodes`.
  - A `TargetDedicated` alias was created for the `TargetDedicatedComputeNodes` parameter.
- Added the `NetworkConfiguration` parameter.
  - This parameter allows you to configure the pools network settings.
- Updated parameters to `New-AzureBatchCertificate`.
  - The `Password` parameter is now a `SecureString`.
- Updated parameters to `New-AzureBatchComputeNodeUser`.
  - The `Password` parameter is now a `SecureString`.
- Updated parameters to `Set-AzureBatchComputeNodeUser`.
  - The `Password` parameter is now a `SecureString`.
- Renamed the `Name` parameter to `Path` on `Get-AzureBatchNodeFile`, `Get-AzureBatchNodeFileContent`, and `Remove-AzureBatchNodeFile`.
  - A `Name` alias was created for the `Path` parameter.
- Changes to objects
  - Please see the Batch change log for the full list
- Added support for Azure Active Directory based authentication.
  - To use Azure Active Directory authentication, retrieve a `BatchAccountContext` object using the `Get-AzureRmBatchAccount` cmdlet, and supply this `BatchAccountContext` to the `-BatchContext` parameter of a Batch service cmdlet. Azure Active Directory authentication is mandatory for accounts with `PoolAllocationMode = UserSubscription`.
  - For existing accounts or for new accounts created with `PoolAllocationMode = BatchService`, you may continue to use shared key authentication by retrieving a `BatchAccountContext` object using the `Get-AzureRmBatchAccountKeys` cmdlet.
- Compute
  - Azure Disk Encryption Extension Commands
    - New Parameter for 'Set-AzureRmVmDiskEncryptionExtension': '-EncryptFormatAll' encrypt formats data disks
    - New Parameters for 'Set-AzureRmVmDiskEncryptionExtension': '-ExtensionPublisherName' and '-ExtensionType' allow switching to other versions of the extension
    - New Parameters for 'Disable-AzureRmVmDiskEncryption': '-ExtensionPublisherName' and '-ExtensionType' allow switching to other versions of the extension
    - New Parameters for 'Get-AzureRmVmDiskEncryptionStatus': '-ExtensionPublisherName' and '-ExtensionType' allow switching to other versions of the extension
- DataLakeAnalytics
  - Please see the migration guide for breaking changes made to DataLakeAnalytics this release
  - Changed one of the two OutputTypes of `Get-AzureRmDataLakeAnalyticsAccount`
    - `List<DataLakeAnalyticsAccount>` to `List<PSDataLakeAnalyticsAccountBasic>`
    - The properties of `PSDataLakeAnalyticsAccountBasic` is a strict subset of the properties of `DataLakeAnalyticsAccount`
    - The additional properties that are in `DataLakeAnalyticsAccount` are not returned by the service. Therefore, this change is to reflect this accurately. These additional properties are still in `PSDataLakeAnalyticsAccountBasic`, but they are tagged as `Obsolete`.
  - Changed one of the two OutputTypes of `Get-AzureRmDataLakeAnalyticsJob`
    - `List<JobInformation>` to `List<PSJobInformationBasic>`



- The properties of PSJobInformationBasic is a strict subset of the properties of JobInformation
  - The additional properties that are in JobInformation are not returned by the service. Therefore, this change is to reflect this accurately. These additional properties are still in PSJobInformationBasic, but they are tagged as Obsolete.
- DataLakeStore
  - Please see the migration guide for breaking changes made to DataLakeStore this release
  - Changed one of the two OutputTypes of Get-AzureRmDataLakeStoreAccount
    - List<PSDataLakeStoreAccount> to List<PSDataLakeStoreAccountBasic>
    - The properties of PSDataLakeStoreAccountBasic is a strict subset of the properties of PSDataLakeStoreAccount
    - The additional properties that are in PSDataLakeStoreAccount are not returned by the service. Therefore, this change is to reflect this accurately. These additional properties are still in PSDataLakeStoreAccountBasic, but they are tagged as Obsolete.
- Dns
  - Support for CAA record types in Azure DNS
    - Supports all operations on CAA record type
- EventHub
  - Please see the migration guide for breaking changes made to EventHub this release
- Insights
  - Please see the migration guide for breaking changes made to Insights this release
- Network
  - Please see the migration guide for breaking changes made to Network this release
  - Added cmdlet to list available internet service providers for a specified Azure region
    - Get-AzureRmNetworkWatcherReachabilityProvidersList
  - Added cmdlet to get the relative latency score for internet service providers from a specified location to Azure regions
    - Get-AzureRmNetworkWatcherReachabilityReport
- Profile
  - Set-AzureRmDefault
    - Use this cmdlet to set a default resource group. This will make the -ResourceGroup parameter optional for some cmdlets, and will use the default when a resource group is not specified
    - `Set-AzureRmDefault -ResourceGroupName "ExampleResourceGroup"`
    - If resource group specified exists in the subscription, this resource group will be set to default. Otherwise, the resource group will be created and then set to default.
  - Get-AzureRmDefault
    - Use this cmdlet to get the current default resource group (and other defaults in the future).
    - `Get-AzureRmDefault -ResourceGroup`
  - Clear-AzureRmDefault
    - Use this cmdlet to remove the current default resource group
    - `Clear-AzureRmDefault -ResourceGroup`
  - Add-AzureRmEnvironment and Set-AzureRmEnvironment
    - Add the BatchAudience parameter, which allows you to specify the Azure Batch Active Directory audience to use when acquiring authentication tokens for the Batch service.
- RecoveryServices.Backup
  - Added cmdlets to perform instant file recovery.
    - Get-AzureRmRecoveryServicesBackupRPMountScript
    - Disable-AzureRmRecoveryServicesBackupRPMountScript

- Updated RecoveryServices.Backup SDK version to the latest
- Updated tests for the Azure VM workload so that, all setups needed for test runs are done by the tests themselves.
- Fixes <https://github.com/Azure/azure-powershell/issues/3164>
- RecoveryServices.SiteRecovery
  - Changes for ASR VMware to Azure Site Recovery (cmdlets are currently supporting operations for Enterprise to Enterprise, Enterprise to Azure, HyperV to Azure)
    - New-AzureRmRecoveryServicesAsrPolicy
    - New-AzureRmRecoveryServicesAsrProtectedItem
    - Update-AzureRmRecoveryServicesAsrPolicy
    - Update-AzureRmRecoveryServicesAsrProtectionDirection
  - Added support to AAD-based vault
  - Added cmdlets to manage VCenter resources
    - Get-AzureRmRecoveryServicesAsrVCenter
    - New-AzureRmRecoveryServicesAsrVCenter
    - Remove-AzureRmRecoveryServicesAsrVCenter
    - Update-AzureRmRecoveryServicesAsrVCenter
  - Added other cmdlets
    - Get-AzureRmRecoveryServicesAsrAlertSetting
    - Get-AzureRmRecoveryServicesAsrEvent
    - New-AzureRmRecoveryServicesAsrProtectableItem
    - Set-AzureRmRecoveryServicesAsrAlertSetting
    - Start-AzureRmRecoveryServicesAsrResynchronizeReplicationJob
    - Start-AzureRmRecoveryServicesAsrSwitchProcessServerJob
    - Start-AzureRmRecoveryServicesAsrTestFailoverCleanupJob
    - Update-AzureRmRecoveryServicesAsrMobilityService
- ServiceBus
  - Please see the migration guide for breaking changes made to ServiceBus this release
- Sql
  - Adding support for list and cancel the asynchronous updateslo operation on the database
    - update existing cmdlet Get-AzureRmSqlDatabaseActivity to return DB updateslo operation status.
    - add new cmdlet Stop-AzureRmSqlDatabaseActivity for cancel the asynchronous updateslo operation on the database.
  - Adding support for Zone Redundancy for databases and elastic pools
    - Adding ZoneRedundant switch parameter to New-AzureRmSqlDatabase
    - Adding ZoneRedundant switch parameter to Set-AzureRmSqlDatabase
    - Adding ZoneRedundant switch parameter to New-AzureRmSqlElasticPool
    - Adding ZoneRedundant switch parameter to Set-AzureRmSqlElasticPool
  - Adding support for Server DNS Aliases
    - Adding Get-AzureRmSqlServerDnsAlias cmdlet which gets server dns aliases by server and alias name or a list of server dns aliases for an azure Sql Server.
    - Adding New-AzureRmSqlServerDnsAlias cmdlet which creates new server dns alias for a given Azure Sql server
    - Adding Set-AzureRmSqlServerDnsAlias cmdlet which allows updating a Azure Sql Server to which server dns alias is pointing
    - Adding Remove-AzureRmSqlServerDnsAlias cmdlet which removes a server dns alias for a Azure Sql Server

- Azure.Storage
  - Upgrade to Azure Storage Client Library 8.5.0 and Azure Storage DataMovement Library 0.6.3
  - Add File Share Snapshot Support Feature
    - Add 'SnapshotTime' parameter to Get-AzureStorageShare
    - Add 'IncludeAllSnapshot' parameter to Remove-AzureStorageShare

# Breaking changes for Microsoft Azure PowerShell 5.0.0

4/10/2018 • 11 min to read • [Edit Online](#)

This document serves as both a breaking change notification and migration guide for consumers of the Microsoft Azure PowerShell cmdlets. Each section describes both the impetus for the breaking change and the migration path of least resistance. For in-depth context, please refer to the pull request associated with each change.

## Table of Contents

- [Breaking changes to ApiManagement cmdlets](#)
- [Breaking changes to Batch cmdlets](#)
- [Breaking changes to Compute cmdlets](#)
- [Breaking changes to EventHub cmdlets](#)
- [Breaking changes to Insights cmdlets](#)
- [Breaking changes to Network cmdlets](#)
- [Breaking changes to Resources cmdlets](#)
- [Breaking Changes to ServiceBus Cmdlets](#)

## Breaking changes to ApiManagement cmdlets

### New-AzureRmApiManagementBackendProxy

- Parameters "UserName" and "Password" are being replaced in favor of a PSCredential

```
# Old
New-AzureRmApiManagementBackendProxy [other required parameters] -UserName "plain-text string" -Password
"plain-text string"

# New
New-AzureRmApiManagementBackendProxy [other required parameters] -Credential $PSCredentialVariable
```

### New-AzureRmApiManagementUser

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmApiManagementUser [other required parameters] -Password "plain-text string"

# New
New-AzureRmApiManagementUser [other required parameters] -Password $SecureStringVariable
```

### Set-AzureRmApiManagementUser

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
Set-AzureRmApiManagementUser [other required parameters] -Password "plain-text string"

# New
Set-AzureRmApiManagementUser [other required parameters] -Password $SecureStringVariable
```

## Breaking changes to Batch cmdlets

### New-AzureBatchCertificate

- Parameter `Password` being replaced in favor of a Secure string

```
# Old
New-AzureBatchCertificate [other required parameters] -Password "plain-text string"

# New
New-AzureBatchCertificate [other required parameters] -Password $SecureStringVariable
```

### New-AzureBatchComputeNodeUser

- Parameter `Password` being replaced in favor of a Secure string

```
# Old
New-AzureBatchComputeNodeUser [other required parameters] -Password "plain-text string"

# New
New-AzureBatchComputeNodeUser [other required parameters] -Password $SecureStringVariable
```

### Set-AzureRmBatchComputeNodeUser

- Parameter `Password` being replaced in favor of a Secure string

```
# Old
Set-AzureRmBatchComputeNodeUser [other required parameters] -Password "plain-text string"

# New
Set-AzureRmBatchComputeNodeUser [other required parameters] -Password $SecureStringVariable
```

### New-AzureBatchTask

- Removed the `RunElevated` switch and replaced it with `UserIdentity`.

```
# Old
New-AzureBatchTask -Id $taskId1 -JobId $jobId -CommandLine "cmd /c echo hello" -RunElevated $TRUE

# New
$autoUser = New-Object Microsoft.Azure.Commands.Batch.Models.PSAutoUserSpecification -ArgumentList @("Task",
"Admin")
$userIdentity = New-Object Microsoft.Azure.Commands.Batch.Models.PSUserIdentity $autoUser
New-AzureBatchTask -Id $taskId1 -JobId $jobId -CommandLine "cmd /c echo hello" -UserIdentity $userIdentity
```

This additionally impacts the `RunElevated` property on `PSCloudTask`, `PSStartTask`, `PSJobManagerTask`, `PSJobPreparationTask`, and `PSJobReleaseTask`.

### PSMultiInstanceSettings

- `PSMultiInstanceSettings` constructor no longer takes a required `numberOfInstances` parameter, instead it takes a required `coordinationCommandLine` parameter.

```
# Old
$settings = New-Object Microsoft.Azure.Commands.Batch.Models.PSMultiInstanceSettings -ArgumentList @(2)
$settings.CoordinationCommandLine = "cmd /c echo hello"
New-AzureBatchTask [other parameters] -MultiInstanceSettings $settings

# New
$settings = New-Object Microsoft.Azure.Commands.Batch.Models.PSMultiInstanceSettings -ArgumentList @("cmd /c
echo hello", 2)
New-AzureBatchTask [other parameters] -MultiInstanceSettings $settings
```

## Get-AzureBatchTask

- Removed the `RunElevated` property on `PSCloudTask`. The `UserIdentity` property has been added to replace `RunElevated`.

```
# Old
$task = Get-AzureBatchTask [parameters]
$task.RunElevated

# New
$task = Get-AzureBatchTask [parameters]
$task.UserIdentity.AutoUser.ElevationLevel
```

This additionally impacts the `RunElevated` property on `PSCloudTask`, `PSStartTask`, `PSJobManagerTask`, `PSJobPreparationTask`, and `PSJobReleaseTask`.

## Multiple types

- Renamed the `SchedulingError` property on `PSExitConditions` to `PreProcessingError`.

```
# Old
$task = Get-AzureBatchTask [parameters]
$task.ExitConditions.SchedulingError

# New
$task = Get-AzureBatchTask [parameters]
$task.ExitConditions.PreProcessingError
```

## Multiple types

- Renamed the `SchedulingError` property on `PSJobPreparationTaskExecutionInformation`, `PSJobReleaseTaskExecutionInformation`, `PSStartTaskInformation`, `PSSubtaskInformation`, and `PSTaskExecutionInformation` to `FailureInformation`.
  - `FailureInformation` is returned any time there is a task failure. This includes all previous scheduling error cases, as well as nonzero task exit codes, and file upload failures from the new output files feature.
  - This is structured the same as before, so no code change is needed when using this type.

```
# Old
$task = Get-AzureBatchTask [parameters]
$task.ExecutionInformation.SchedulingError

# New
$task = Get-AzureBatchTask [parameters]
$task.ExecutionInformation.FailureInformation
```

This additionally impacts: `Get-AzureBatchPool`, `Get-AzureBatchSubtask`, and `Get-AzureBatchJobPreparationAndReleaseTaskStatus`

## New-AzureBatchPool

- Removed `TargetDedicated` and replaced it with `TargetDedicatedComputeNodes` and `TargetLowPriorityComputeNodes` .
- `TargetDedicatedComputeNodes` has an alias `TargetDedicated` .

```
# Old
New-AzureBatchPool [other parameters] [-TargetDedicated <Int32>]

# New
New-AzureBatchPool [other parameters] [-TargetDedicatedComputeNodes <Int32>] [-TargetLowPriorityComputeNodes <Int32>]
```

This also impacts: Start-AzureBatchPoolResize

## Get-AzureBatchPool

- Renamed the `TargetDedicated` and `CurrentDedicated` properties on `PSCloudPool` to `TargetDedicatedComputeNodes` and `CurrentDedicatedComputeNodes` .

```
# Old
$pool = Get-AzureBatchPool [parameters]
$pool.TargetDedicated
$pool.CurrentDedicated

# New
$pool = Get-AzureBatchPool [parameters]
$pool.TargetDedicatedComputeNodes
$pool.CurrentDedicatedComputeNodes
```

## Type PSCloudPool

- Renamed `ResizeError` to `ResizeErrors` on `PSCloudPool` , and it is now a collection.

```
# Old
$pool = Get-AzureBatchPool [parameters]
$pool.ResizeError

# New
$pool = Get-AzureBatchPool [parameters]
$pool.ResizeErrors[0]
```

## New-AzureBatchJob

- Renamed the `TargetDedicated` property on `PSPoolSpecification` to `TargetDedicatedComputeNodes` .

```
# Old
$poolInfo = New-Object Microsoft.Azure.Commands.Batch.Models.PSPoolInformation
$poolInfo.AutoPoolSpecification = New-Object Microsoft.Azure.Commands.Batch.Models.PSAutoPoolSpecification
$poolInfo.AutoPoolSpecification.PoolSpecification = New-Object
Microsoft.Azure.Commands.Batch.Models.PSPoolSpecification
$poolInfo.AutoPoolSpecification.PoolSpecification.TargetDedicated = 5
New-AzureBatchJob [other parameters] -PoolInformation $poolInfo

# New
$poolInfo = New-Object Microsoft.Azure.Commands.Batch.Models.PSPoolInformation
$poolInfo.AutoPoolSpecification = New-Object Microsoft.Azure.Commands.Batch.Models.PSAutoPoolSpecification
$poolInfo.AutoPoolSpecification.PoolSpecification = New-Object
Microsoft.Azure.Commands.Batch.Models.PSPoolSpecification
$poolInfo.AutoPoolSpecification.PoolSpecification.TargetDedicatedComputeNodes = 5
New-AzureBatchJob [other parameters] -PoolInformation $poolInfo
```

## Get-AzureBatchNodeFile

- Removed `Name` and replaced it with `Path`.
- `Path` has an alias `Name`.

```
# Old
Get-AzureBatchNodeFile [other parameters] [[-Name] <String>]

# New
Get-AzureBatchNodeFile [other parameters] [[-Path] <String>]
```

This also impacts: `Get-AzureBatchNodeFileContent`, `Remove-AzureBatchNodeFile`

## Type PSNodeFile

- Renamed the `Name` property on `PSNodeFile` to `Path`.

```
# Old
$file = Get-AzureBatchNodeFile [parameters]
$file.Name

# New
$file = Get-AzureBatchNodeFile [parameters]
$file.Path
```

## Get-AzureBatchSubtask

- The `PreviousState` and `State` properties of `PSSubtaskInformation` are no longer of type `TaskState`, instead they are of type `SubtaskState`.
  - Unlike `TaskState`, `SubtaskState` has no `Active` value, since it is not possible for subtasks to be in an `Active` state.

```
# Old
$subtask = Get-AzureBatchSubtask [parameters]
if ($subtask.State -eq Microsoft.Azure.Batch.Common.TaskState.Running) { }

# New
$subtask = Get-AzureBatchSubtask [parameters]
if ($subtask.State -eq Microsoft.Azure.Batch.Common.SubtaskState.Running) { }
```

# Breaking changes to Compute cmdlets

## Set-AzureRmVMAccessExtension



- Parameters "UserName" and "Password" are being replaced in favor of a PSCredential

```
# Old
Set-AzureRmVMAccessExtension [other required parameters] -UserName "plain-text string" -Password "plain-text string"

# New
Set-AzureRmVMAccessExtension [other required parameters] -Credential $PSCredential
```

## Breaking changes to EventHub cmdlets

### New-AzureRmEventHubNamespaceAuthorizationRule

- The 'New-AzureRmEventHubNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'New-AzureRmEventHubAuthorizationRule' cmdlet

### Get-AzureRmEventHubNamespaceAuthorizationRule

- The 'Get-AzureRmEventHubNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'Get-AzureRmEventHubAuthorizationRule' cmdlet

### Set-AzureRmEventHubNamespaceAuthorizationRule

- The 'Set-AzureRmEventHubNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'Set-AzureRmEventHubAuthorizationRule' cmdlet

### Remove-AzureRmEventHubNamespaceAuthorizationRule

- The 'Remove-AzureRmEventHubNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'Remove-AzureRmEventHubAuthorizationRule' cmdlet

### New-AzureRmEventHubNamespaceKey

- The 'New-AzureRmEventHubNamespaceKey' cmdlet has been removed. Please use the 'New-AzureRmEventHubKey' cmdlet

### Get-AzureRmEventHubNamespaceKey

- The 'Get-AzureRmEventHubNamespaceKey' cmdlet has been removed. Please use the 'Get-AzureRmEventHubKey' cmdlet

### New-AzureRmEventHubNamespace

- The property 'Status' and 'Enabled' from the NamespaceAttributes will be removed.

```
# Old
# The $namespace has Status and Enabled property
$namespace = New-AzureRmEventHubNamespace <parameters>
$namespace.Status
$namespace.Enabled

# New
# The call remains the same, but the returned values Namespace object will not have the Status and Enabled property
$namespace = Get-AzureRmEventHubNamespace <parameters>
```

### Get-AzureRmEventHubNamespace

- The property 'Status' and 'Enabled' from the NamespaceAttributes will be removed.

```
# Old
# The $namespace has Status and Enabled property
$namespace = Get-AzureRmEventHubNamespace <parameters>
$namespace.Status
$namespace.Enabled

# New
# The call remains the same, but the returned values NameSpace object will not have the Status and Enabled
property
$namespace = Get-AzureRmEventHubNamespace <parameters>
```

### Set-AzureRmEventHubNamespace

- The property 'Status' and 'Enabled' from the NamespaceAttributes will be removed.

```
# Old
# The $namespace has Status and Enabled property
$namespace = Set-AzureRmEventHubNamespace <parameters>
$namespace.Status
$namespace.Enabled

# New
# The call remains the same, but the returned values NameSpace object will not have the Status and Enabled
property
$namespace = Set-AzureRmEventHubNamespace <parameters>
```

### New-AzureRmEventHubConsumerGroup

- The property 'EventHubPath' from the ConsumerGroupAttributes will be removed.

```
# Old
# The $consumerGroup has EventHubPath property
$consumerGroup = New-AzureRmEventHubConsumerGroup <parameters>
$consumerGroup.EventHubPath

# New
# The call remains the same, but the returned values ConsumerGroup object will not have the EventHubPath
property
$consumerGroup = New-AzureRmEventHubConsumerGroup <parameters>
```

### Set-AzureRmEventHubConsumerGroup

- The property 'EventHubPath' from the ConsumerGroupAttributes will be removed.

```
# Old
# The $consumerGroup has EventHubPath property
$consumerGroup = Set-AzureRmEventHubConsumerGroup <parameters>
$consumerGroup.EventHubPath

# New
# The call remains the same, but the returned values ConsumerGroup object will not have the EventHubPath
property
$consumerGroup = Set-AzureRmEventHubConsumerGroup <parameters>
```

### Get-AzureRmEventHubConsumerGroup

- The property 'EventHubPath' from the ConsumerGroupAttributes will be removed.

```
# Old
# The $consumergroup has EventHubPath property
$consumergroup = Get-AzureRmEventHubConsumerGroup <parameters>
$consumergroup.EventHubPath

# New
# The call remains the same, but the returned values ConsumerGroup object will not have the EventHubPath
property
$consumergroup = Get-AzureRmEventHubConsumerGroup <parameters>
```

## Breaking changes to Insights cmdlets

### Add-AzureRMLogAlertRule

- The **Add-AzureRMLogAlertRule** cmdlet has been deprecated
- After October 1st using this cmdlet will no longer have any effect as this functionality is being transitioned to Activity Log Alerts. Please see <https://aka.ms/migratemealerts> for more information.

### Get-AzureRMUsage

- The **Get-AzureRMUsage** cmdlet has been deprecated

### Get-AzureRmAlertHistory / Get-AzureRmAutoscaleHistory / Get-AzureRmLogs

- Output change: The field EventChannels from theEventData object (returned by these cmdlets) is being deprecated since it now returns a constant value (Admin,Operation.)

### Get-AzureRmAlertRule

- Output change: The output of this cmdlet will be flattened, i.e. elimination of the properties field, to improve the user experience.

```
# Old
$rules = Get-AzureRmAlertRule -ResourceGroup $resourceGroup
if ($rules -and $rules.count -ge 1)
{
    Write-Host -ForegroundColor Red "Error updating alert rule"
    Write-Host $rules[0].Id
    Write-Host $rules[0].Properties.IsEnabled
    Write-Host $rules[0].Properties.Condition
}

# New
$rules = Get-AzureRmAlertRule -ResourceGroup $resourceGroup
if ($rules -and $rules.count -ge 1)
{
    Write-Host -ForegroundColor red "Error updating alert rule"
    Write-Host $rules[0].Id

    # Properties will remain for a while
    Write-Host $rules[0].Properties.IsEnabled

    # But the properties will be at the top level too. Later Properties will be removed
    Write-Host $rules[0].IsEnabled
    Write-Host $rules[0].Condition
}
```

### Get-AzureRmAutoscaleSetting

- Output change: The AutoscaleSettingResourceName field will be deprecated since it always equals the Name field.

```
# Old
$s1 = Get-AzureRmAutoscaleSetting -ResourceGroup $resourceGroup -Name MySetting
if ($s1.AutoscaleSettingResourceName -ne $s1.Name)
{
    Write-Host "There is something wrong with the name"
}

# New
$s1 = Get-AzureRmAutoscaleSetting -ResourceGroup $resourceGroup -Name MySetting

# there won't be a AutoscaleSettingResourceName
Write-Host $s1.Name
```

### Remove-AzureRmAlertRule / Remove-AzureRmLogProfile

- Output change: The type of the output will change to return a single object containing the request Id and the status code.

```
# Old
$s1 = Remove-AzureRmAlertRule -ResourceGroup $resourceGroup -name $ruleName
if ($s1 -ne $null)
{
    $r = $s1[0].RequestId
    $s = $s1[0].StatusCode
}

# New
$s1 = Remove-AzureRmAlertRule -ResourceGroup $resourceGroup -name $ruleName
$r = $s1.RequestId
$s = $s1.StatusCode
```

## Breaking changes to Network cmdlets

### Add-AzureRmApplicationGatewaySslCertificate

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
Add-AzureRmApplicationGatewaySslCertificate [other required parameters] -Password "plain-text string"

# New
Add-AzureRmApplicationGatewaySslCertificate [other required parameters] -Password $SecureStringVariable
```

### New-AzureRmApplicationGatewaySslCertificate

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmApplicationGatewaySslCertificate [other required parameters] -Password "plain-text string"

# New
New-AzureRmApplicationGatewaySslCertificate [other required parameters] -Password $SecureStringVariable
```

### Set-AzureRmApplicationGatewaySslCertificate

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
Set-AzureRmApplicationGatewaySslCertificate [other required parameters] -Password "plain-text string"

# New
Set-AzureRmApplicationGatewaySslCertificate [other required parameters] -Password $SecureStringVariable
```

## Breaking changes to Resources cmdlets

### New-AzureRmADAppCredential

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmADAppCredential [other required parameters] -Password "plain-text string"

# New
New-AzureRmADAppCredential [other required parameters] -Password $SecureStringVariable
```

### New-AzureRmADApplication

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmADApplication [other required parameters] -Password "plain-text string"

# New
New-AzureRmADApplication [other required parameters] -Password $SecureStringVariable
```

### New-AzureRmADServicePrincipal

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmADServicePrincipal [other required parameters] -Password "plain-text string"

# New
New-AzureRmADServicePrincipal [other required parameters] -Password $SecureStringVariable
```

### New-AzureRmADSpCredential

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmADSpCredential [other required parameters] -Password "plain-text string"

# New
New-AzureRmADSpCredential [other required parameters] -Password $SecureStringVariable
```

### New-AzureRmADUser

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
New-AzureRmADUser [other required parameters] -Password "plain-text string"

# New
New-AzureRmADUser [other required parameters] -Password $SecureStringVariable
```

## Set-AzureRmADUser

- Parameter "Password" being replaced in favor of a SecureString

```
# Old
Set-AzureRmADUser [other required parameters] -Password "plain-text string"

# New
Set-AzureRmADUser [other required parameters] -Password $SecureStringVariable
```

## Breaking changes to ServiceBus cmdlets

### Get-AzureRmServiceBusTopicAuthorizationRule

- The 'Get-AzureRmServiceBusTopicAuthorizationRule' cmdlet has been removed. Please use the 'Get-AzureRmServiceBusAuthorizationRule' cmdlet.

### Get-AzureRmServiceBusTopicKey

- The 'Get-AzureRmServiceBusTopicKey' cmdlet has been removed. Please use the 'Get-AzureRmServiceBusKey' cmdlet.

### New-AzureRmServiceBusTopicAuthorizationRule

- The 'New-AzureRmServiceBusTopicAuthorizationRule' cmdlet has been removed. Please use the 'New-AzureRmServiceBusAuthorizationRule' cmdlet.

### New-AzureRmServiceBusTopicKey

- The 'New-AzureRmServiceBusTopicKey' cmdlet has been removed. Please use the 'New-AzureRmServiceBusKey' cmdlet.

### Remove-AzureRmServiceBusTopicAuthorizationRule

- The 'Remove-AzureRmServiceBusTopicAuthorizationRule' cmdlet has been removed. Please use the 'Remove-AzureRmServiceBusAuthorizationRule' cmdlet.

### Set-AzureRmServiceBusTopicAuthorizationRule

- The 'Set-AzureRmServiceBusTopicAuthorizationRule' cmdlet has been removed. Please use the 'Set-AzureRmServiceBusAuthorizationRule' cmdlet.

### New-AzureRmServiceBusNamespaceKey

- The 'New-AzureRmServiceBusNamespaceKey' cmdlet has been removed. Please use the 'New-AzureRmServiceBusKey' cmdlet.

### Get-AzureRmServiceBusQueueAuthorizationRule

- The 'Get-AzureRmServiceBusQueueAuthorizationRule' cmdlet has been removed. Please use the 'Get-AzureRmServiceBusAuthorizationRule' cmdlet.

### Get-AzureRmServiceBusQueueKey

- The 'Get-AzureRmServiceBusQueueKey' cmdlet has been removed. Please use the 'Get-AzureRmServiceBusKey' cmdlet.

### New-AzureRmServiceBusQueueAuthorizationRule

- The 'New-AzureRmServiceBusQueueAuthorizationRule' cmdlet has been removed. Please use the 'New-AzureRmServiceBusAuthorizationRule' cmdlet.

### New-AzureRmServiceBusQueueKey

- The 'New-AzureRmServiceBusQueueKey' cmdlet has been removed. Please use the 'New-AzureRmServiceBusKey' cmdlet.

### **Remove-AzureRmServiceBusQueueAuthorizationRule**

- The 'Remove-AzureRmServiceBusQueueAuthorizationRule' cmdlet has been removed. Please use the 'GRemove-AzureRmServiceBusAuthorizationRule' cmdlet.

### **Set-AzureRmServiceBusQueueAuthorizationRule**

- The 'Set-AzureRmServiceBusQueueAuthorizationRule' cmdlet has been removed. Please use the 'Set-AzureRmServiceBusAuthorizationRule' cmdlet.

### **Get-AzureRmServiceBusNamespaceAuthorizationRule**

- The 'Get-AzureRmServiceBusNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'Get-AzureRmServiceBusAuthorizationRule' cmdlet.

### **Get-AzureRmServiceBusNamespaceKey**

- The 'Get-AzureRmServiceBusNamespaceKey' cmdlet has been removed. Please use the 'Get-AzureRmServiceBusKey' cmdlet.

### **New-AzureRmServiceBusNamespaceAuthorizationRule**

- The 'New-AzureRmServiceBusNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'New-AzureRmServiceBusAuthorizationRule' cmdlet.

### **Remove-AzureRmServiceBusNamespaceAuthorizationRule**

- The 'Remove-AzureRmServiceBusNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'Remove-AzureRmServiceBusAuthorizationRule' cmdlet.

### **Set-AzureRmServiceBusNamespaceAuthorizationRule**

- The 'Set-AzureRmServiceBusNamespaceAuthorizationRule' cmdlet has been removed. Please use the 'Set-AzureRmServiceBusAuthorizationRule' cmdlet.

### **Type NamespaceAttributes**

- The following properties have been removed
  - Enabled
  - Status

```
# Old
# The $namespace has Status and Enabled property
$namespace = Get-AzureRmServiceBusNamespace <parameters>
$namespace.Status
$namespace.Enabled

# New
# The call remains the same, but the returned values Namespace object will not have the Enabled and Status
properties
$namespace = Get-AzureRmServiceBusNamespace <parameters>
```

### **Type QueueAttribute**

- The following properties are marked as obsolete:
  - EnableBatchedOperations
  - EntityAvailabilityStatus
  - IsAnonymousAccessible
  - SupportOrdering

```
# Old
# The $queue has EntityAvailabilityStatus, EnableBatchedOperations, IsAnonymousAccessible and SupportOrdering
properties
$queue = Get-AzureRmServiceBusQueue <parameters>
$queue.EntityAvailabilityStatus
$queue.EnableBatchedOperations
$queue.IsAnonymousAccessible
$queue.SupportOrdering

# New
# The call remains the same, but the returned values Queue object will not have the EntityAvailabilityStatus,
EnableBatchedOperations, IsAnonymousAccessible and SupportOrdering properties
$queue = Get-AzureRmServiceBusQueue <parameters>
```

## Type TopicAttribute

- The following properties are marked as obsolete:
  - Location
  - IsExpress
  - IsAnonymousAccessible
  - FilteringMessagesBeforePublishing
  - EnableSubscriptionPartitioning
  - EntityAvailabilityStatus

```
# Old
# The $topic has EntityAvailabilityStatus, EnableSubscriptionPartitioning, IsAnonymousAccessible, IsExpress,
Location and FilteringMessagesBeforePublishing properties
$topic = Get-AzureRmServiceBusTopic <parameters>
$topic.EntityAvailabilityStatus
$topic.EnableSubscriptionPartitioning
$topic.IsAnonymousAccessible
$topic.IsExpress
$topic.FilteringMessagesBeforePublishing
$topic.Location

# New
# The call remains the same, but the returned values Topic object will not have the EntityAvailabilityStatus,
EnableBatchedOperations, IsAnonymousAccessible and SupportOrdering properties
$topic = Get-AzureRmServiceBusTopic <parameters>
```

## Type SubscriptionAttribute

- The following properties are marked as obsolete
  - DeadLetteringOnFilterEvaluationExceptions
  - EntityAvailabilityStatus
  - IsReadOnly
  - Location



# Old

# The \$subscription has EntityAvailabilityStatus, EnableSubscriptionPartitioning, IsAnonymousAccessible, IsExpress, Location and FilteringMessagesBeforePublishing properties

\$subscription = Get-AzureRmServiceBusSubscription <parameters>

\$subscription.EntityAvailabilityStatus

\$subscription.EnableSubscriptionPartitioning

\$subscription.IsAnonymousAccessible

\$subscription.IsExpress

\$subscription.FilteringMessagesBeforePublishing

\$subscription.Location

# New

# The call remains the same, but the returned values Topic object will not have the EntityAvailabilityStatus, EnableBatchedOperations, IsAnonymousAccessible and SupportOrdering properties

\$subscription = Get-AzureRmServiceBusSubscription <parameters>