Contents

Introducing the new Azure PowerShell Az module

11/4/2019 • 3 minutes to read • Edit Online

Starting in December 2018, the Azure PowerShell Az module is in general release and is now the intended PowerShell module for interacting with Azure. Az offers shorter commands, improved stability, and cross-platform support. Az also has feature parity with AzureRM, giving you an easy migration path.

With the Az module, Azure PowerShell is now compatible with PowerShell 5.1 on Windows and PowerShell Core 6.x and later on all supported platforms - including Windows, macOS, and Linux.

Az is a new module, so the version has been reset to 1.0.0.

Why a new module?

Major updates can be inconvenient, so it's important that we let you know why the decision was made to introduce a new set of modules, with new cmdlets, for interacting with Azure from PowerShell.

The biggest and most important change is that PowerShell has been a cross-platform product since the introduction of PowerShell Core 6.x, based on the .NET Standard library. We're committed to bringing Azure support to all platforms, which means that the Azure PowerShell modules needed to be updated to use .NET Standard and be compatible with PowerShell Core. Rather than taking the existing AzureRM module and introduce complex changes to add this support, the Az module was created.

Creating a new module also gave our engineers the opportunity to make the design and naming of cmdlets and modules consistent. All modules now start with the `Az.` prefix and cmdlets all use the *Verb*-`Az`*Noun* form. Previously, cmdlet names were not only longer, there were inconsistencies in cmdlet names.

The number of modules was also reduced: Some modules which worked with the same services have been rolled together, and management plane and data plane cmdlets are now contained all within single modules for their services. For those of you who manually manage dependencies and imports, this makes things much simpler.

By making these important changes that required building a new Azure PowerShell module, the team has committed to making it easier than ever, and on more platforms than previously possible, to use Azure with PowerShell cmdlets.

Upgrade to Az

To keep up with the latest Azure features in PowerShell, you should migrate to the Az module as soon as possible. If you're not ready to install the Az module as a replacement for AzureRM, you have a couple of options available to experiment with Az:

- Use a `PowerShell` environment with Azure Cloud Shell. Azure Cloud Shell is a browser-based shell environment which comes with the Az module installed and `Enable-AzureRM` compatibility aliases enabled.
- Keep the AzureRM module installed with PowerShell 5.1 for Windows, but install the Az module for PowerShell Core 6.x or later. PowerShell 5.1 for Windows and PowerShell Core use separate collections of modules. Follow the instructions to install PowerShell Core and then install the Az module from a PowerShell Core terminal.

To upgrade from an existing AzureRM install:

1. Uninstall the Azure PowerShell AzureRM module
2. Install the Azure PowerShell Az module
3. **OPTIONAL**: Enable compatibility mode to add aliases for AzureRM cmdlets with Enable-AzureRMAlias while

you become familiar with the new command set. See the next section or Start migration from AzureRM to Az for more details.

Migrate existing scripts to Az

The new cmdlet names have been designed to be easy to learn. Instead of using `AzureRm` or `Azure` in cmdlet names, use `Az`. For example, the old command `New-AzureRMVm` has become `New-AzVm`. Migration is more than just becoming familiar with the new cmdlet names, though: There are renamed modules, parameters, and other important changes.

To help you with the process of migration from AzureRM to Az, we've got a number of resources:

- Get started with migration from AzureRM to Az
- Full list of breaking changes from AzureRM to Az 1.0.0
- The Enable-AzureRmAlias cmdlet

The Az module has a compatibility mode to help you use existing scripts while you update to the new syntax. The Enable-AzureRmAlias cmdlet enables a compatibility mode through aliases, to allow you to use existing scripts with minimal modification while working towards a full migration to Az.

> **IMPORTANT**
>
> Even though the cmdlet names are aliased, there may still be new (or renamed) parameters or changed return values for the Az cmdlets. Don't expect enabling aliases to take care of the migration for you! See the full breaking changes list to find where your scripts may require updates.

Continued support for AzureRM

AzureRM will no longer receive new cmdlets or features. However, the AzureRM module is still officially maintained and will get bug fixes through December 2020.

Install the Azure PowerShell module

11/4/2019 • 5 minutes to read • Edit Online

This article tells you how to install the Azure PowerShell modules using PowerShellGet. These instructions work on Windows, macOS, and Linux platforms. For the Az module, currently no other installation methods are supported.

Requirements

Azure PowerShell works with PowerShell 5.1 or higher on Windows, or PowerShell Core 6.x and later on all platforms. If you aren't sure if you have PowerShell, or are on macOS or Linux, install the latest version of PowerShell Core.

To check your PowerShell version, run the command:

```
$PSVersionTable.PSVersion
```

To run Azure PowerShell in PowerShell 5.1 on Windows:

1. Update to Windows PowerShell 5.1 if needed. If you're on Windows 10, you already have PowerShell 5.1 installed.
2. Install .NET Framework 4.7.2 or later.

There are no additional requirements for Azure PowerShell when using PowerShell Core.

Install the Azure PowerShell module

WARNING

You **can't** have both the AzureRM and Az modules installed for PowerShell 5.1 for Windows at the same time. If you need to keep AzureRM available on your system, install the Az module for PowerShell Core 6.x or later. To do this, install PowerShell Core 6.x or later and then follow these instructions in a PowerShell Core terminal.

The recommended install method is to only install for the active user:

```
Install-Module -Name Az -AllowClobber -Scope CurrentUser
```

If you want to install for all users on a system, this requires administrator privileges. From an elevated PowerShell session either run as administrator or with the `sudo` command on macOS or Linux:

```
Install-Module -Name Az -AllowClobber -Scope AllUsers
```

By default, the PowerShell gallery isn't 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"):
```

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

The Az module is a rollup module for the Azure PowerShell cmdlets. Installing it downloads all of the available Azure Resource Manager modules, and makes their cmdlets available for use.

Install offline

In some environments it's not possible to connect to the PowerShell Gallery. In those situations, you can still install offline using one of these methods:

- Download the modules to another location and use that as an installation source on your network. This can be a complicated process, but will let you cache PowerShell modules on a single server or file share to be deployed with PowerShellGet to any disconnected systems. Learn how to set up a local repository and install on disconnected systems with Working with local PowerShellGet repositories.
- Download the Azure PowerShell MSI to a machine connected to the network, and then copy the installer to systems without access to PowerShell Gallery. Keep in mind that the MSI installer only works for PowerShell 5.1 on Windows.
- Save the module with Save-Module to a file share, or save it to another source and manually copy it to other machines:

  ```
  Save-Module -Name Az -Path '\\someshare\PowerShell\modules' -Force
  ```

Troubleshooting

Here are some common problems seen when installing the Azure PowerShell module. If you experience a problem not listed here, please file an issue on GitHub.

Proxy blocks connection

If you get errors from `Install-Module` that indicate the PowerShell Gallery is unreachable, you may be behind a proxy. Different operating systems will have different requirements for configuring a system-wide proxy, which are not covered in detail here. Contact your system administrator for your proxy settings and how to configure them for your OS.

PowerShell itself may not be configured to use this proxy automatically. With PowerShell 5.1 and later, configure the proxy to use for a PowerShell session with the following command:

```
(New-Object System.Net.WebClient).Proxy.Credentials = `
  [System.Net.CredentialCache]::DefaultNetworkCredentials
```

If your operating system credentials are configured correctly, this will route PowerShell requests through the proxy. In order to have this setting persist between sessions, add the command to a PowerShell profile.

In order to install the package, your proxy needs to allow HTTPS connections to the following address:

- `https://www.powershellgallery.com`

Sign in

To start working with Azure PowerShell, sign in with your Azure credentials.

```
# Connect to Azure with a browser sign in token
Connect-AzAccount
```

> **NOTE**
> If you've disabled module autoloading, manually import the module with `Import-Module Az`. Because of the way the module is structured, this can take a few seconds.

You'll need to repeat these steps for every new PowerShell session you start. To learn how to persist your Azure sign-in across PowerShell sessions, see Persist user credentials across PowerShell sessions.

Update the Azure PowerShell module

Because of how the Az module is packaged, the Update-Module command won't update your installation correctly. When you install the Az module, it actually collects and installs all of its dependent submodules, and which provide the cmdlets for each service. That means that to update the Azure PowerShell module, you will need to **reinstall**, rather than just **update**. This is done in the same way as installing, but you may need to add the `-Force` argument:

```
Install-Module -Name Az -AllowClobber -Force
```

Although this can overwrite installed modules, you may still have older versions left on your system. To learn how to remove old versions of Azure PowerShell from your system, see Uninstall the Azure PowerShell module.

Use multiple versions of Azure PowerShell

It's possible to install more than one version of Azure PowerShell. To check if you have multiple versions of Azure PowerShell installed, use the following command:

```
Get-InstalledModule -Name Az -AllVersions | select Name,Version
```

To remove a version of Azure PowerShell, see Uninstall the Azure PowerShell module.

You can install or load a specific version of the `Az` module by using the `-RequiredVersion` argument:

```
# Install Az version 0.7.0
Install-Module -Name Az -RequiredVersion 0.7.0
# Load Az version 0.7.0
Import-Module -Name Az -RequiredVersion 0.7.0
```

If you have more than one version of the module installed, module autoload and `Import-Module` load the latest version by default.

Provide feedback

If you find a bug in Azure Powershell, file an issue on GitHub. To provide feedback from the command line, use the Send-Feedback cmdlet.

Next Steps

To learn more about the Azure PowerShell modules and their features, see Get Started with Azure PowerShell. If you're familiar with Azure PowerShell and need to migrate from AzureRM, see Migrate from AzureRM to Az.

Install Azure PowerShell on Windows with MSI

11/4/2019 • 2 minutes to read • Edit Online

This article explains how to install Azure PowerShell on Windows using an MSI installer. The MSI installer is provided for environments where the PowerShell Gallery may be blocked by a firewall, or an offline installer is needed. The recommended way to install Azure PowerShell is with PowerShellGet. For instructions on using PowerShellGet to install Azure PowerShell, see Install Azure PowerShell with PowerShellGet.

Requirements

The MSI installer for Azure PowerShell works **only** for PowerShell 5.1 on Windows. For installation on non-Windows platforms or later versions of powershell, Install with PowerShellGet. To check your PowerShell version, run the command:

```
$PSVersionTable.PSVersion
```

To use Azure PowerShell in PowerShell 5.1, you need to:

1. Update to Windows PowerShell 5.1 if needed. If you're on Windows 10, you already have PowerShell 5.1 installed.
2. Install .NET Framework 4.7.2 or later.

Install or update on Windows using the MSI Package

Azure PowerShell for Windows is installed using the MSI file available from GitHub. If you have installed earlier versions of Azure modules as an MSI, the installer automatically removes them. The MSI package installs modules in `${env:ProgramFiles}\WindowsPowerShell\Modules`.

To start working with Azure PowerShell, sign in with your Azure credentials.

```
# Connect to Azure with an interactive dialog for sign-in
Connect-AzAccount
```

> **NOTE**
> If you've disabled module autoloading, you need to manually import the module with `Import-Module Az`. Because of the way the module is structured, this can take up to a minute.

You'll need to repeat this step for every new PowerShell session you start. To learn how to persist your Azure sign-in across PowerShell sessions, see Persist user credentials across PowerShell sessions.

Provide feedback

If you find a bug in Azure Powershell, file an issue on GitHub. To provide feedback from the command line, use the Send-Feedback cmdlet.

Next Steps

To learn more about the Azure PowerShell modules and their features, see Get Started with Azure PowerShell. If

you're familiar with Azure PowerShell and need to migrate from AzureRM, see Migrate from AzureRM to Az.

Uninstall the Azure PowerShell module

11/13/2019 • 4 minutes to read • Edit Online

This article tells you how to uninstall an older version of Azure PowerShell, or completely remove it from your system. If you've decided to completely uninstall the Azure PowerShell, give us some feedback through the Send-Feedback cmdlet. If you encountered a bug, we'd appreciate it if you file a GitHub issue so that it can be fixed.

Uninstall Azure PowerShell from MSI

If you installed Azure PowerShell using the MSI package, you must uninstall through the Windows system rather than PowerShell.

PLATFORM	INSTRUCTIONS
Windows 10	Start > Settings > Apps
Windows 7 Windows 8	Start > Control Panel > Programs > Uninstall a program

Once on this screen you should see **Azure PowerShell** in the program listing. This is the app to uninstall. If you don't see this program listed, then you installed through PowerShellGet, and should follow the next set of instructions.

Uninstall Azure PowerShell from PowerShell Get

To uninstall the Az modules, use the Uninstall-Module cmdlet. However, `Uninstall-Module` only uninstalls one module. To remove Azure PowerShell completely, you must uninstall each module individually. Uninstallation can be complicated if you have more than one version of Azure PowerShell installed.

To check which versions of Azure PowerShell you currently have installed, run the following command:

```
Get-InstalledModule -Name Az -AllVersions
```

```
Version           Name                Repository          Description
-------           ----                ----------          -----------
0.7.0             Az                  PSGallery           Azure Resource Manager Module
1.0.0             Az                  PSGallery           Azure Resource Manager Module
```

The following script queries the PowerShell Gallery to get a list of dependent submodules. Then, the script uninstalls the correct version of each submodule. You will need to have administrator access to run this script in a scope other than `Process` or `CurrentUser`.

```
function Uninstall-AllModules {
  param(
    [Parameter(Mandatory=$true)]
    [string]$TargetModule,

    [Parameter(Mandatory=$true)]
    [string]$Version,

    [switch]$Force,

    [switch]$WhatIf
  )

  $AllModules = @()

  'Creating list of dependencies...'
  $target = Find-Module $TargetModule -RequiredVersion $version
  $target.Dependencies | ForEach-Object {
    if ($_.PSObject.Properties.Name -contains 'requiredVersion') {
      $AllModules += New-Object -TypeName psobject -Property @{name=$_.name; version=$_.requiredVersion}
    }
    else { # Assume minimum version
      # Minimum version actually reports the installed dependency
      # which is used, not the actual "minimum dependency." Check to
      # see if the requested version was installed as a dependency earlier.
      $candidate = Get-InstalledModule $_.name -RequiredVersion $version -ErrorAction Ignore
      if ($candidate) {
        $AllModules += New-Object -TypeName psobject -Property @{name=$_.name; version=$version}
      }
      else {
        $availableModules = Get-InstalledModule $_.name -AllVersions
        Write-Warning ("Could not find uninstall candidate for {0}:{1} - module may require manual uninstall.
Available versions are: {2}" -f $_.name,$version,($availableModules.Version -join ', '))
      }
    }
  }
  $AllModules += New-Object -TypeName psobject -Property @{name=$TargetModule; version=$Version}

  foreach ($module in $AllModules) {
    Write-Host ('Uninstalling {0} version {1}...' -f $module.name,$module.version)
    try {
      Uninstall-Module -Name $module.name -RequiredVersion $module.version -Force:$Force -ErrorAction Stop -
WhatIf:$WhatIf
    } catch {
      Write-Host ("`t" + $_.Exception.Message)
    }
  }
}
```

To use this function, copy and paste the code into your PowerShell session. The following example shows how to run the function to remove an older version of Azure PowerShell.

```
Uninstall-AllModules -TargetModule Az -Version 0.7.0 -Force
```

As the script runs, it will display the name and version of each submodule that is being uninstalled. To run the script to only see what would be deleted, without removing it, use the `-WhatIf` option.

```
Creating list of dependencies...
Uninstalling Az.Profile version 0.7.0
Uninstalling Az.Aks version 0.7.0
Uninstalling Az.AnalysisServices version 0.7.0
...
```

> **NOTE**
> If this script can't match an exact dependency with the same version to uninstall, it won't uninstall *any* version of that dependecy. This is because there may be other versions of the target module on your system which rely on these dependencies. In this case, the available versions of the dependency are listed. You can then remove any old versions manually with `Uninstall-Module`.

Run this command for every version of Azure PowerShell that you want to uninstall. For convenience, the following script will uninstall all versions of Az **except** for the latest.

```
$versions = (Get-InstalledModule Az -AllVersions | Select-Object Version)
$versions[0..($versions.Length-2)]  | foreach { Uninstall-AllModules -TargetModule Az -Version ($_.Version) -
Force }
```

Uninstall the AzureRM module

If you have the Az module installed on your system and would like to uninstall AzureRM, there are two options that don't require running the `Uninstall-AllModules` script above. Which method you follow depends on how you installed the AzureRM module. If you're not sure of your original install method, follow the steps for uninstalling an MSI first.

Uninstall Azure PowerShell MSI

If you installed the Azure PowerShell AzureRM modules using the MSI package, you must uninstall through the Windows system rather than PowerShell.

PLATFORM	INSTRUCTIONS
Windows 10	Start > Settings > Apps
Windows 7 Windows 8	Start > Control Panel > Programs > Uninstall a program

Once on this screen you should see **Azure PowerShell** or **Microsoft Azure PowerShell - Month Year** in the program listing. This is the app to uninstall. If you don't see this program listed, then you installed through PowerShellGet, and should follow the next set of instructions.

Uninstall from PowerShell

If you installed AzureRM with PowerShellGet, then you can remove the modules with the Uninstall-AzureRM command, available as part of the `Az.Accounts` module. This removes *all* AzureRM modules from your machine, but requires administrator privileges.

```
Uninstall-AzureRm
```

If you can't successfully run the `Uninstall-AzureRM` command, use the `Uninstall-AllModules` script provided in this article with the following invocation:

```
$versions = (Get-InstalledModule AzureRM -AllVersions | Select-Object Version)
$versions | foreach { Uninstall-AllModules -TargetModule AzureRM -Version ($_.Version) -Force }
```

Migrate Azure PowerShell from AzureRM to Az

11/4/2019 • 4 minutes to read • Edit Online

The Az module has feature parity with AzureRM, but uses shorter and more consistent cmdlet names. Scripts written for the AzureRM cmdlets won't automatically work with the new module. To make the transition easier, Az offers tools to allow you to run your existing scripts using AzureRM. No migration to a new command set is ever convenient, but this article will help you get started on transitioning to the new module.

To see the full list of breaking changes between AzureRM and Az, see the full changes from AzureRM to Az.

Ensure existing scripts work with the latest AzureRM release

Before taking any migration steps, check which versions of AzureRM are installed on your system. Doing so allows you to make sure scripts are already running on the latest release, and let you know which versions of AzureRM must be uninstalled.

To check which version(s) of AzureRM you have installed, run the command:

```
Get-InstalledModule -Name AzureRM -AllVersions
```

The **latest** available release of AzureRM is **6.13.1**. If you don't have this version installed, your existing scripts may need additional modification to work with the Az module beyond what's described here and in the breaking changes list.

If your scripts don't work with AzureRM 6.13.1, update them according to the AzureRM 5.x to 6.x migration guide. If you use an earlier version of the AzureRM module, there are migration guides available for each major version.

Uninstall AzureRM

The Az module is not guaranteed to be compatible with any existing AzureRM installs in PowerShell 5.1 for Windows. Before you install the Az module, uninstall AzureRM.

> **IMPORTANT**
>
> If you're not ready to remove the AzureRM module from your system, you can install the Az module for PowerShell Core 6.x or later instead. PowerShell Core and PowerShell 5.1 for Windows use different module libraries, so there will be no conflicts. You can still enable aliases in PowerShell Core.

Install the Azure PowerShell Az module

The first step is to install the Az module on your platform. When you install Az, it's recommended that you uninstall AzureRM. In the following steps, you'll learn how to keep running your existing scripts and enable compatibility for old cmdlet names.

To install the Azure PowerShell Az module, follow the instructions in Install the Az module.

> **NOTE**
>
> At this point, you might want to run the Uninstall-AzureRM cmdlet provided in the Az module, just to make sure that all versions of AzureRM have been uninstalled and won't cause conflicts.

Enable AzureRM compatibility aliases

With AzureRM uninstalled and your scripts working with the latest AzureRM version, the next step is to enable the compatibility mode for the Az module. Compatibility is enabled with the command:

```
Enable-AzureRmAlias -Scope CurrentUser
```

Aliases enable the ability to use old cmdlet names with the Az module installed. These aliases are written to the profile for the selected scope. If no profile exists, one is created. When using a `-Scope` broader than `CurrentUser`, the appropriate permissions are required to create or update the corresponding profile file.

> **IMPORTANT**
> **Only** cmdlet names are aliased - module names aren't! If you're using `#Requires`, `Import-Module`, dependency lists in a `.psd1`, or fully-qualified cmdlet names, make sure that you migrate them at this point by following the process outlined in the breaking changes list regarding module names.

> **WARNING**
> You can use a different `-Scope` for this command, but it's not recommended. Aliases are written to the user profile for the selected scope, so keep enabling them to as limited a scope as possible. Enabling aliases system-wide can cause issues for other users who have AzureRM installed in their local scope.

Once the alias mode is enabled, run your scripts again to confirm that they still function as expected. Some parameter names have been changed, added, or made required by the Az module. Output types of cmdlets may have changed as well. These changes are detailed in the breaking changes list.

Update cmdlets, modules, and parameters

With scripts updated and running under aliases, you can take your time to update them to use the new cmdlets and take advantage of other changes like new features. For most scripts, you will only need to update cmdlet names, following the new cmdlet naming scheme in Az. There may also be some other changes that you need to make in order to have your scripts work, depending on what they do and which Azure PowerShell features they take advantage of.

For example, the Blob Storage cmdlets have been completely reworked to use a new asynchronous model, so scripts using them will take more work to update than those where the only relevant changes were cmdlet names.

Even if you've only had to make small, simple changes to your scripts up to this point - or they even work without additional modification when aliases are enabled - read the full list of breaking changes in Az 1.0.0 to make sure that you're not relying on 'transparent' behavior of aliases which could disappear after you change cmdlet names and disable aliases.

Disable aliases

Once you've completed your migration and are no longer relying on aliasing behavior, it's recommended that you disable aliases. This is done with the Disable-AzureRmAlias cmdlet.

> **IMPORTANT**
> When running this cmdlet, **make sure** that you invoke it for each `-Scope` that `Enable-AzureRmAlias` was invoked for, otherwise there may still be scripts on your system relying on the aliasing behavior.

Breaking changes for Az 1.0.0

11/4/2019 • 8 minutes to read • Edit Online

This document provides detailed information on the changes between AzureRM 6.x and the new Az module, version 1.x and later. The table of contents will help guide you through a full migration path, including module-specific changes that may affect your scripts.

For general advice on getting started with a migration from AzureRM to Az, see Start migration from AzureRM to Az.

> **IMPORTANT**
> There have been breaking changes between Az 1.0.0 and Az 2.0.0 as well. After following this guide for updating from AzureRM to Az, see the Az 2.0.0 breaking changes to find out if you need to make additional changes.

Table of Contents

General breaking changes

This section details the general breaking changes that are part of the redesign of the Az module.

Cmdlet Noun Prefix Changes

In the AzureRM module, cmdlets used either `AzureRM` or `Azure` as a noun prefix. Az simplifies and normalizes cmdlet names, so that all cmdlets use 'Az' as their cmdlet noun prefix. For example:

```
Get-AzureRMVM
Get-AzureKeyVaultSecret
```

Has changed to:

```
Get-AzVM
Get-AzKeyVaultSecret
```

To make the transition to these new cmdlet names simpler, Az introduces two new cmdlets, Enable-AzureRmAlias and Disable-AzureRmAlias. `Enable-AzureRmAlias` creates aliases for the older cmdlet names in AzureRM that map to the newer Az cmdlet names. Using the `-Scope` argument with `Enable-AzureRmAlias` allows you to choose where aliases are enabled.

For example, the following script in AzureRM:

```
#Requires -Modules AzureRM.Storage
Get-AzureRmStorageAccount | Get-AzureStorageContainer | Get-AzureStorageBlob
```

Can be run with minimal changes using `Enable-AzureRmAlias`:

```
#Requires -Modules Az.Storage
Enable-AzureRmAlias -Scope Process
Get-AzureRmStorageAccount | Get-AzureStorageContainer | Get-AzureStorageBlob
```

Running `Enable-AzureRmAlias -Scope CurrentUser` will enable the aliases for all PowerShell sessions you open, so that after executing this cmdlet, a script like this would not need to be changed at all:

```
Get-AzureRmStorageAccount | Get-AzureStorageContainer | Get-AzureStorageBlob
```

For complete details on the usage of the alias cmdlets, see the Enable-AzureRmAlias reference.

When you're ready to disable aliases, `Disable-AzureRmAlias` removes the created aliases. For complete details, see the Disable-AzureRmAlias reference.

> **IMPORTANT**
> When disabling aliases, make sure that they are disabled for *all* scopes which had aliases enabled.

Module Name Changes

The module names have changed from `AzureRM.*` to `Az.*`, except for the following modules:

AZURERM MODULE	AZ MODULE

AZURERM MODULE	AZ MODULE
Azure.Storage	Az.Storage
Azure.AnalysisServices	Az.AnalysisServices
AzureRM.Profile	Az.Accounts
AzureRM.Insights	Az.Monitor
AzureRM.DataFactories	Az.DataFactory
AzureRM.DataFactoryV2	Az.DataFactory
AzureRM.RecoveryServices.Backup	Az.RecoveryServices
AzureRM.RecoveryServices.SiteRecovery	Az.RecoveryServices
AzureRM.Tags	Az.Resources
AzureRM.MachineLearningCompute	Az.MachineLearning
AzureRM.UsageAggregates	Az.Billing
AzureRM.Consumption	Az.Billing

The changes in module names mean that any script that uses `#Requires` or `Import-Module` to load specific modules will need to be changed to use the new module instead. For modules where the cmdlet suffix has not changed, this means that although the module name has changed, the suffix indicating the operation space has *not*.

Migrating #Requires and Import-Module Statements

Scripts that use `#Requires` or `Import-Module` to declare a dependency on AzureRM modules must be updated to use the new module names. For example:

```
#Requires -Module AzureRM.Compute
```

Should be changed to:

```
#Requires -Module Az.Compute
```

For `Import-Module`:

```
Import-Module -Name AzureRM.Compute
```

Should be changed to:

```
Import-Module -Name Az.Compute
```

Migrating Fully-Qualified Cmdlet Invocations

Scripts that use module-qualified cmdlet invocations, such as:

```
AzureRM.Compute\Get-AzureRmVM
```

Must be changed to use the new module and cmdlet names:

```
Az.Compute\Get-AzVM
```

Migrating module manifest dependencies

Modules that express dependencies on AzureRM modules through a module manifest (.psd1) file will need to updated the module names in their `RequiredModules` section:

```
RequiredModules = @(@{ModuleName="AzureRM.Profile"; ModuleVersion="5.8.2"})
```

Must be changed to:

```
RequiredModules = @(@{ModuleName="Az.Profile"; ModuleVersion="1.0.0"})
```

Removed modules

The following modules have been removed:

- `AzureRM.Backup`
- `AzureRM.Compute.ManagedService`
- `AzureRM.Scheduler`

The tools for these services are no longer actively supported. Customers are encouraged to move to alternative services as soon as it is convenient.

Windows PowerShell 5.1 and .NET 4.7.2

Using Az with PowerShell 5.1 for Windows requires the installation of .NET Framework 4.7.2. Using PowerShell Core 6.x or later does not require .NET Framework.

Temporary removal of User login using PSCredential

Due to changes in the authentication flow for .NET Standard, we are temporarily removing user login via PSCredential. This capability will be re-introduced in the 1/15/2019 release for PowerShell 5.1 for Windows. This is discussed in detail in this GitHub issue.

Default device code login instead of web browser prompt

Due to changes in the authentication flow for .NET Standard, we are using device login as the default login flow during interactive login. Web browser based login will be re-introduced for PowerShell 5.1 for Windows as the default in the 1/15/2019 release. At that time, users will be able to choose device login using a Switch parameter.

Module breaking changes

This section details specific breaking changes for individual modules and cmdlets.

Az.ApiManagement (previously AzureRM.ApiManagement)

- Removed the following cmdlets:
  - New-AzureRmApiManagementHostnameConfiguration
  - Set-AzureRmApiManagementHostnames
  - Update-AzureRmApiManagementDeployment
  - Import-AzureRmApiManagementHostnameCertificate

    - Use **Set-AzApiManagement** cmdlet to set these properties instead
- Removed the following properties:
    - Removed property `PortalHostnameConfiguration`, `ProxyHostnameConfiguration`, `ManagementHostnameConfiguration` and `ScmHostnameConfiguration` of type `PsApiManagementHostnameConfiguration` from `PsApiManagementContext`. Instead use `PortalCustomHostnameConfiguration`, `ProxyCustomHostnameConfiguration`, `ManagementCustomHostnameConfiguration` and `ScmCustomHostnameConfiguration` of type `PsApiManagementCustomHostNameConfiguration`.
    - Removed property `StaticIPs` from PsApiManagementContext. The property has been split into `PublicIPAddresses` and `PrivateIPAddresses`.
    - Removed required property `Location` from New-AzureApiManagementVirtualNetwork cmdlet.

Az.Billing (previously AzureRM.Billing, AzureRM.Consumption, and AzureRM.UsageAggregates)

- The `InvoiceName` parameter was removed from the `Get-AzConsumptionUsageDetail` cmdlet. Scripts will need to use other identity parameters for the invoice.

Az.CognitiveServices (previously AzureRM.CognitiveServices)

- Removed `GetSkusWithAccountParamSetName` parameter set from `Get-AzCognitiveServicesAccountSkus` cmdlet. You must get Skus by Account Type and Location, instead of using ResourceGroupName and Account Name.

Az.Compute (previously AzureRM.Compute)

- `IdentityIds` are removed from `Identity` property in `PSVirtualMachine` and `PSVirtualMachineScaleSet` objects Scripts should no longer use the value of this field to make processing decisions.
- The type of `InstanceView` property of `PSVirtualMachineScaleSetVM` object is changed from `VirtualMachineInstanceView` to `VirtualMachineScaleSetVMInstanceView`
- `AutoOSUpgradePolicy` and `AutomaticOSUpgrade` properties are removed from `UpgradePolicy` property
- The type of `Sku` property in `PSSnapshotUpdate` object is changed from `DiskSku` to `SnapshotSku`
- `VmScaleSetVMParameterSet` is removed from `Add-AzVMDataDisk` cmdlet, you cna no longer add a data disk individually to a ScaleSet VM.

Az.DataFactory (previously AzureRM.DataFactories and AzureRM.DataFactoryV2)

- The `GatewayName` parameter has become mandatory in the `New-AzDataFactoryEncryptValue` cmdlet
- Removed `New-AzDataFactoryGatewayKey` cmdlet
- Removed `LinkedServiceName` parameter from `Get-AzDataFactoryV2ActivityRun` cmdlet Scripts should no longer use the value of this field to make processing decisions.

Az.DataLakeAnalytics (previously AzureRM.DataLakeAnalytics)

- Removed deprecated cmdlets: `New-AzDataLakeAnalyticsCatalogSecret`, `Remove-AzDataLakeAnalyticsCatalogSecret`, and `Set-AzDataLakeAnalyticsCatalogSecret`

Az.DataLakeStore (previously AzureRM.DataLakeStore)

- The following cmdlets have had the `Encoding` parameter changed from the type `FileSystemCmdletProviderEncoding` to `System.Text.Encoding`. This change removes the encoding values `String` and `Oem`. All the other prior encoding values remain.
    - New-AzureRmDataLakeStoreItem
    - Add-AzureRmDataLakeStoreItemContent
    - Get-AzureRmDataLakeStoreItemContent
- Removed deprecated `Tags` property alias from `New-AzDataLakeStoreAccount` and `Set-AzDataLakeStoreAccount` cmdlets

  Scripts using

```
New-AzureRMDataLakeStoreAccount -Tags @{TagName="TagValue"}
```

Should be changed to

```
New-AzDataLakeStoreAccount -Tag @{TagName="TagValue"}
```

- Removed deprecated properties `Identity`, `EncryptionState`, `EncryptionProvisioningState`, `EncryptionConfig`, `FirewallState`, `FirewallRules`, `VirtualNetworkRules`, `TrustedIdProviderState`, `TrustedIdProviders`, `DefaultGroup`, `NewTier`, `CurrentTier`, `FirewallAllowAzureIps` from `PSDataLakeStoreAccountBasic` object. Any script that uses the `PSDatalakeStoreAccount` returned from `Get-AzDataLakeStoreAccount` should not reference these properties.

Az.KeyVault (previously AzureRM.KeyVault)

- The `PurgeDisabled` property was removed from the `PSKeyVaultKeyAttributes`, `PSKeyVaultKeyIdentityItem`, and `PSKeyVaultSecretAttributes` objects Scripts should no longer reference the `PurgeDisabled` property to make processing decisions.

Az.Media (previously AzureRM.Media)

- Remove deprecated `Tags` property alias from `New-AzMediaService` cmdlet Scripts using

```
New-AzureRMMediaService -Tags @{TagName="TagValue"}
```

Should be changed to

```
New-AzMediaService -Tag @{TagName="TagValue"}
```

Az.Monitor (previously AzureRM.Insights)

- Removed plural names `Categories` and `Timegrains` parameter in favor of singular parameter names from `Set-AzDiagnosticSetting` cmdlet Scripts using

```
Set-AzureRmDiagnosticSetting -Timegrains PT1M -Categories Category1, Category2
```

Should be changed to

```
Set-AzDiagnosticSetting -Timegrain PT1M -Category Category1, Category2
```

Az.Network (previously AzureRM.Network)

- Removed deprecated `ResourceId` parameter from `Get-AzServiceEndpointPolicyDefinition` cmdlet
- Removed deprecated `EnableVmProtection` property from `PSVirtualNetwork` object
- Removed deprecated `Set-AzVirtualNetworkGatewayVpnClientConfig` cmdlet

Scripts should no longer make processing decisions based on the values fo these fields.

Az.OperationalInsights (previously AzureRM.OperationalInsights)

- Default parameter set for `Get-AzOperationalInsightsDataSource` is removed, and `ByWorkspaceNameByKind` has become the default parameter set

  Scripts that listed data sources using

```
Get-AzureRmOperationalInsightsDataSource
```

Should be changed to specify a Kind

```
Get-AzOperationalInsightsDataSource -Kind AzureActivityLog
```

Az.RecoveryServices (previously AzureRM.RecoveryServices, AzureRM.RecoveryServices.Backup, and AzureRM.RecoveryServices.SiteRecovery)

- Removed `Encryption` parameter from `New/Set-AzRecoveryServicesAsrPolicy` cmdlet
- `TargetStorageAccountName` parameter is now mandatory for managed disk restores in `Restore-AzRecoveryServicesBackupItem` cmdlet
- Removed `StorageAccountName` and `StorageAccountResourceGroupName` parameters in `Restore-AzRecoveryServicesBackupItem` cmdlet
- Removed `Name` parameter in `Get-AzRecoveryServicesBackupContainer` cmdlet

Az.Resources (previously AzureRM.Resources)

- Removed `Sku` parameter from `New/Set-AzPolicyAssignment` cmdlet
- Removed `Password` parameter from `New-AzADServicePrincipal` and `New-AzADSpCredential` cmdlet
  Passwords are automatically generated, scripts that provided the password:

```
New-AzAdSpCredential -ObjectId 1f99cf81-0146-4f4e-beae-2007d0668476 -Password $secPassword
```

Should be changed to retrieve the password from the output:

```
$credential = New-AzAdSpCredential -ObjectId 1f99cf81-0146-4f4e-beae-2007d0668476
$secPassword = $credential.Secret
```

Az.ServiceFabric (previously AzureRM.ServiceFabric)

- The following cmdlet return types have been changed:
  - The property `ServiceTypeHealthPolicies` of type `ApplicationHealthPolicy` has been removed.
  - The property `ApplicationHealthPolicies` of type `ClusterUpgradeDeltaHealthPolicy` has been removed.
  - The property `OverrideUserUpgradePolicy` of type `ClusterUpgradePolicy` has been removed.
  - These changes affect the following cmdlets:
    - Add-AzServiceFabricClientCertificate
    - Add-AzServiceFabricClusterCertificate
    - Add-AzServiceFabricNode
    - Add-AzServiceFabricNodeType
    - Get-AzServiceFabricCluster
    - Remove-AzServiceFabricClientCertificate
    - Remove-AzServiceFabricClusterCertificate
    - Remove-AzServiceFabricNode
    - Remove-AzServiceFabricNodeType
    - Remove-AzServiceFabricSetting
    - Set-AzServiceFabricSetting
    - Set-AzServiceFabricUpgradeType
    - Update-AzServiceFabricDurability

- Update-AzServiceFabricReliability

Az.Sql (previously AzureRM.Sql)

- Removed `State` and `ResourceId` parameters from `Set-AzSqlDatabaseBackupLongTermRetentionPolicy` cmdlet
- Removed deprecated cmdlets: `Get/Set-AzSqlServerBackupLongTermRetentionVault`, `Get/Start/Stop-AzSqlServerUpgrade`, `Get/Set-AzSqlDatabaseAuditingPolicy`, `Get/Set-AzSqlServerAuditingPolicy`, `Remove-AzSqlDatabaseAuditing`, `Remove-AzSqlServerAuditing`
- Removed deprecated parameter `Current` from `Get-AzSqlDatabaseBackupLongTermRetentionPolicy` cmdlet
- Removed deprecated parameter `DatabaseName` from `Get-AzSqlServerServiceObjective` cmdlet
- Removed deprecated parameter `PrivilegedLogin` from `Set-AzSqlDatabaseDataMaskingPolicy` cmdlet

Az.Storage (previously Azure.Storage and AzureRM.Storage)

- To support creating an Oauth storage context with only the storage account name, the default parameter set has been changed to `OAuthParameterSet`
  - Example: `$ctx = New-AzureStorageContext -StorageAccountName $accountName`
- The `Location` parameter has become mandatory in the `Get-AzStorageUsage` cmdlet
- The Storage API methods now use the Task-based Asynchronous Pattern (TAP), instead of synchronous API calls. The following examples demonstrate the new asynchronous commands:

Blob Snapshot

AzureRM:

```
$b = Get-AzureStorageBlob -Container $containerName -Blob $blobName -Context $ctx
$b.ICloudBlob.Snapshot()
```

Az:

```
$b = Get-AzStorageBlob -Container $containerName -Blob $blobName -Context $ctx
$task = $b.ICloudBlob.SnapshotAsync()
$task.Wait()
$snapshot = $task.Result
```

Share Snapshot

AzureRM:

```
$Share = Get-AzureStorageShare -Name $containerName -Context $ctx
$snapshot = $Share.Snapshot()
```

Az:

```
$Share = Get-AzStorageShare -Name $containerName -Context $ctx
$task = $Share.SnapshotAsync()
$task.Wait()
$snapshot = $task.Result
```

Undelete soft-deleted blob

AzureRM:

```
$b = Get-AzureStorageBlob -Container $containerName -Blob $blobName -IncludeDeleted -Context $ctx
$b.ICloudBlob.Undelete()
```

Az:

```
$b = Get-AzStorageBlob -Container $containerName -Blob $blobName -IncludeDeleted -Context $ctx
$task = $b.ICloudBlob.UndeleteAsync()
$task.Wait()
```

Set Blob Tier

AzureRM:

```
$blockBlob = Get-AzureStorageBlob -Container $containerName -Blob $blockBlobName -Context $ctx
$blockBlob.ICloudBlob.SetStandardBlobTier("hot")

$pageBlob = Get-AzureStorageBlob -Container $containerName -Blob $pageBlobName -Context $ctx
$pageBlob.ICloudBlob.SetPremiumBlobTier("P4")
```

Az:

```
$blockBlob = Get-AzStorageBlob -Container $containerName -Blob $blockBlobName -Context $ctx
$task = $blockBlob.ICloudBlob.SetStandardBlobTierAsync("hot")
$task.Wait()

$pageBlob = Get-AzStorageBlob -Container $containerName -Blob $pageBlobName -Context $ctx
$task = $pageBlob.ICloudBlob.SetPremiumBlobTierAsync("P4")
$task.Wait()
```

Az.Websites (previously AzureRM.Websites)

- Removed deprecated properties from the `PSAppServicePlan`, `PSCertificate`, `PSCloningInfo`, and `PSSite` objects

Get started with Azure PowerShell

11/4/2019 • 3 minutes to read • Edit Online

Azure PowerShell is designed for managing and administering Azure resources from the command line. Use Azure PowerShell when you want to build automated tools that use the Azure Resource Manager model. Try it out in your browser with Azure Cloud Shell, or install on your local machine.

This article helps you get started with Azure PowerShell and teaches the core concepts behind it.

Install or run in Azure Cloud Shell

The easiest way to get started with Azure PowerShell is by trying it out in an Azure Cloud Shell environment. To get up and running with Cloud Shell, see Quickstart for PowerShell in Azure Cloud Shell. Cloud Shell runs PowerShell 6 on a Linux container, so Windows-specific functionality isn't available.

When you're ready to install Azure PowerShell on your local machine, follow the instructions in Install the Azure PowerShell module.

Sign in to Azure

Sign in interactively with the `Connect-AzAccount` cmdlet. Skip this step if you use Cloud Shell: Your Azure Cloud Shell session is already authenticated for the environment, subscription, and tenant that launched the Cloud Shell session.

```
Connect-AzAccount
```

If you're in a non-US region, use the `-Environment` parameter to sign in. Get the name of the environment for your region by using the Get-AzEnvironment cmdlet. For example, to sign in to Azure China 21Vianet:

```
Connect-AzAccount -Environment AzureChinaCloud
```

In PowerShell 5.1 environments, you'll get a sign-in dialog to provide a username and password for your Azure account. On every other version of PowerShell, you'll get a token to use on [https://microsoft.com/devicelogin]. Open this page in your browser and enter the token, then sign in with your Azure account credentials and authorize Azure PowerShell.

After signing in, you'll see information indicating which of your Azure subscriptions is active. If you have multiple Azure subscriptions in your account and want to select a different one, get your available subscriptions with Get-AzSubscription and use the Set-AzContext cmdlet with your subscription ID. For more information about managing your Azure subscriptions in Azure PowerShell, see Use multiple Azure subscriptions.

Once signed in, use the Azure PowerShell cmdlets to access and manage resources in your subscription. To learn more about the sign-in process and authentication methods, see Sign in with Azure PowerShell.

Find commands

Azure PowerShell cmdlets follow a standard naming convention for PowerShell, `VERB-NOUN`. The verb describes the action (examples include `New`, `Get`, `Set`, `Remove`) and the noun describes the resource type (examples include `AzVM`, `AzKeyVaultCertificate`, `AzFirewall`, `AzVirtualNetworkGateway`). Nouns in Azure PowerShell always start with the prefix `Az`. For the full list of standard verbs, see Approved verbs for PowerShell Commands.

Knowing the nouns, verbs, and the Azure PowerShell modules available help you find commands with the Get-Command cmdlet. For example, to find all VM-related commands that use the `Get` verb:

```
Get-Command -Verb Get -Noun AzVM* -Module Az.Compute
```

To help you find common commands, this table lists the resource type, corresponding Azure PowerShell module, and noun prefix to use with `Get-Command`:

RESOURCE TYPE	AZURE POWERSHELL MODULE	NOUN PREFIX
Resource group	Az.Resources	`AzResourceGroup`
Virtual machines	Az.Compute	`AzVM`
Storage accounts	Az.Storage	`AzStorageAccount`
Key Vault	Az.KeyVault	`AzKeyVault`
Web applications	Az.Websites	`AzWebApp`
SQL databases	Az.Sql	`AzSqlDatabase`

For a full list of the modules in Azure PowerShell, see the Azure PowerShell modules list hosted on GitHub.

Learn Azure PowerShell basics with quickstarts and tutorials

To get started with Azure PowerShell, try an in-depth tutorial for setting up virtual machines and learning how to query them.

Create virtual machines with Azure PowerShell

There are also Azure PowerShell quickstarts for other popular Azure services:

- Create a storage account
- Transfer objects to/from Azure Blob storage
- Create and retrieve secrets from Azure Key Vault
- Create an Azure SQL database and firewall
- Run a container in Azure Container Instances
- Create a Virtual Machine Scale Set (VMSS)
- Create a standard load balancer

Next steps

- Sign in with Azure PowerShell
- Manage Azure subscriptions with Azure PowerShell
- Create service principals with Azure PowerShell
- Get help from the community:
  - Azure forum on MSDN
  - Stack Overflow

Sign in with Azure PowerShell

11/4/2019 • 4 minutes to read • Edit Online

Azure PowerShell supports several authentication methods. The easiest way to get started is with Azure Cloud Shell, which automatically logs you in. With a local install, you can sign in interactively through your browser. When writing scripts for automation, the recommended approach is to use a service principal with the necessary permissions. When you restrict sign-in permissions as much as possible for your use case, you help keep your Azure resources secure.

After signing in, commands are run against your default subscription. To change your active subscription for a session, use the Set-AzContext cmdlet. To change the default subscription used when logging in with Azure PowerShell, use Set-AzDefault.

> **IMPORTANT**
> Your credentials are shared among multiple PowerShell sessions as long as you remain signed in. For more information, see the article on Persistent Credentials.

Sign in interactively

To sign in interactively, use the Connect-AzAccount cmdlet.

```
Connect-AzAccount
```

When run, this cmdlet will present a token string. To sign in, copy this string and paste it into https://microsoft.com/devicelogin in a browser. Your PowerShell session will be authenticated to connect to Azure.

> **IMPORTANT**
> Username/password credential authorization has been removed in Azure PowerShell due to changes in Active Directory authorization implementations and security concerns. If you use credential authorization for automation purposes, instead create a service principal.

Sign in with a service principal

Service principals are non-interactive Azure accounts. Like other user accounts, their permissions are managed with Azure Active Directory. By granting a service principal only the permissions it needs, your automation scripts stay secure.

To learn how to create a service principal for use with Azure PowerShell, see Create an Azure service principal with Azure PowerShell.

To sign in with a service principal, use the `-ServicePrincipal` argument with the `Connect-AzAccount` cmdlet. You'll also need the service principal's application ID, sign-in credentials, and the tenant ID associate with the service principal. How you sign in with a service principal will depend on whether it's configured for password-based or certificate-based authentication.

Password-based authentication

To get the service principal's credentials as the appropriate object, use the Get-Credential cmdlet. This cmdlet will

present a prompt for a username and password. Use the service principal ID for the username.

```
$pscredential = Get-Credential
Connect-AzAccount -ServicePrincipal -Credential $pscredential -Tenant $tenantId
```

For automation scenarios, you need to create credentials from a user name and secure string:

```
$passwd = ConvertTo-SecureString <use a secure password here> -AsPlainText -Force
$pscredential = New-Object System.Management.Automation.PSCredential('service principal name/id', $passwd)
Connect-AzAccount -ServicePrincipal -Credential $pscredential -Tenant $tenantId
```

Make sure that you use good password storage practices when automating service principal connections.

Certificate-based authentication

Certificate-based authentication requires that Azure PowerShell can retrieve information from a local certificate store based on a certificate thumbprint.

```
Connect-AzAccount -ApplicationId $appId -Tenant $tenantId -CertificateThumbprint <thumbprint>
```

When using a service principal instead of a registered application, add the `-ServicePrincipal` argument and provide the service principal's ID as the `-ApplicationId` parameter's value.

```
Connect-AzAccount -ServicePrincipal -ApplicationId $servicePrincipalId -Tenant $tenantId -
CertificateThumbprint <thumbprint>
```

In PowerShell 5.1, the certificate store can be managed and inspected with the PKI module. For PowerShell Core 6.x and later, the process is more complicated. The following scripts show you how to import an existing certificate into the certificate store accessible by PowerShell.

Import a certificate in PowerShell 5.1

```
# Import a PFX
$credentials = Get-Credential -Message "Provide PFX private key password"
Import-PfxCertificate -FilePath <path to certificate> -Password $credentials.Password -CertStoreLocation
cert:\CurrentUser\My
```

Import a certificate in PowerShell Core 6.x and later

```
# Import a PFX
$storeName = [System.Security.Cryptography.X509Certificates.StoreName]::My
$storeLocation = [System.Security.Cryptography.X509Certificates.StoreLocation]::CurrentUser
$store = [System.Security.Cryptography.X509Certificates.X509Store]::new($storeName, $storeLocation)
$certPath = <path to certificate>
$credentials = Get-Credential -Message "Provide PFX private key password"
$flag = [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable
$certificate = [System.Security.Cryptography.X509Certificates.X509Certificate2]::new($certPath,
$credentials.Password, $flag)
$store.Open([System.Security.Cryptography.X509Certificates.OpenFlags]::ReadWrite)
$store.Add($Certificate)
$store.Close()
```

Sign in using a managed identity

Managed identities are a feature of Azure Active Directory. Managed identities are service principals assigned to resources that run in Azure. You can use a managed identity service principal for sign-in, and acquire an app-only

access token to access other resources. Managed identities are only available on resources running in an Azure cloud.

To learn more about managed identities for Azure resources, see How to use managed identities for Azure resources on an Azure VM to acquire an access token.

Sign in with a non-default tenant or as a Cloud Solution Provider (CSP)

If your account is associated with more than one tenant, sign-in requires the use of the `-Tenant` parameter when connecting. This parameter will work with any sign-in method. When logging in, this parameter value can either be the Azure object ID of the tenant (Tenant ID) or the fully qualified domain name of the tenant.

If you're a Cloud Solution Provider (CSP), the `-Tenant` value **must** be a tenant ID.

```
Connect-AzAccount -Tenant 'xxxx-xxxx-xxxx-xxxx'
```

Sign in to another Cloud

Azure cloud services offer environments compliant with regional data-handling laws. For accounts in a regional cloud, set the environment when you sign in with the `-Environment` argument. This parameter will work with any sign-in method. For example, if your account is in the China cloud:

```
Connect-AzAccount -Environment AzureChinaCloud
```

The following command gets a list of available environments:

```
Get-AzEnvironment | Select-Object Name
```

Create an Azure service principal with Azure PowerShell

11/4/2019 • 5 minutes to read • Edit Online

Automated tools that use Azure services should always have restricted permissions. Instead of having applications sign in as a fully privileged user, Azure offers service principals.

An Azure service principal is an identity created for use with applications, hosted services, and automated tools to access Azure resources. This access is restricted by the roles assigned to the service principal, giving you control over which resources can be accessed and at which level. For security reasons, it's always recommended to use service principals with automated tools rather than allowing them to log in with a user identity.

This article shows you the steps for creating, getting information about, and resetting a service principal with Azure PowerShell.

Create a service principal

Create a service principal with the New-AzADServicePrincipal cmdlet. When creating a service principal, you choose the type of sign-in authentication it uses.

> **NOTE**
>
> If your account doesn't have permission to create a service principal, `New-AzADServicePrincipal` will return an error message containing "Insufficient privileges to complete the operation." Contact your Azure Active Directory admin to create a service principal.

There are two types of authentication available for service principals: Password-based authentication, and certificate-based authentication.

Password-based authentication

Without any other authentication parameters, password-based authentication is used and a random password created for you. If you want password-based authentication, this method is recommended.

```
$sp = New-AzADServicePrincipal -DisplayName ServicePrincipalName
```

The returned object contains the `Secret` member, which is a `SecureString` containing the generated password. Make sure that you store this value somewhere secure to authenticate with the service principal. Its value **won't** be displayed in the console output. If you lose the password, reset the service principal credentials.

The following code will allow you to export the secret:

```
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($sp.Secret)
$UnsecureSecret = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
```

For user-supplied passwords, the `-PasswordCredential` argument takes `Microsoft.Azure.Commands.ActiveDirectory.PSADPasswordCredential` objects. These objects must have a valid `StartDate` and `EndDate`, and take a plaintext `Password`. When creating a password, make sure you follow the Azure Active Directory password rules and restrictions. Don't use a weak password or reuse a password.

```
Import-Module Az.Resources # Imports the PSADPasswordCredential object
$credentials = New-Object Microsoft.Azure.Commands.ActiveDirectory.PSADPasswordCredential -Property @{
StartDate=Get-Date; EndDate=Get-Date -Year 2024; Password=<Choose a strong password>}
$sp = New-AzAdServicePrincipal -DisplayName ServicePrincipalName -PasswordCredential $credentials
```

The object returned from `New-AzADServicePrincipal` contains the `Id` and `DisplayName` members, either of which can be used for sign in with the service principal.

> **IMPORTANT**
> Signing in with a service principal requires the tenant ID which the service principal was created under. To get the active tenant when the service principal was created, run the following command **immediately after** service principal creation:
>
> ```
> (Get-AzContext).Tenant.Id
> ```

Certificate-based authentication

Service principals using certificate-based authentication are created with the `-CertValue` parameter. This parameter takes a base64-encoded ASCII string of the public certificate. This is represented by a PEM file, or a text-encoded CRT or CER. Binary encodings of the public certificate aren't supported. These instructions assume that you already have a certificate available.

```
$cert = <public certificate as base64-encoded string>
$sp = New-AzADServicePrincipal -DisplayName ServicePrincipalName -CertValue $cert
```

You can also use the `-KeyCredential` parameter, which takes `PSADKeyCredential` objects. These objects must have a valid `StartDate`, `EndDate`, and have the `CertValue` member set to a base64-encoded ASCII string of the public certificate.

```
$cert = <public certificate as base64-encoded string>
$credentials = New-Object Microsoft.Azure.Commands.ActiveDirectory.PSADKeyCredential -Property @{
StartDate=Get-Date; EndDate=Get-Date -Year 2024; KeyId=New-Guid; CertValue=$cert}
$sp = New-AzADServicePrincipal -DisplayName ServicePrincipalName -KeyCredential $credentials
```

The object returned from `New-AzADServicePrincipal` contains the `Id` and `DisplayName` members, either of which can be used for sign in with the service principal. Clients which sign in with the service principal also need access to the certificate's private key.

> **IMPORTANT**
> Signing in with a service principal requires the tenant ID which the service principal was created under. To get the active tenant when the service principal was created, run the following command **immediately after** service principal creation:
>
> ```
> (Get-AzContext).Tenant.Id
> ```

Get an existing service principal

A list of service principals for the currently active tenant can be retrieved with Get-AzADServicePrincipal. By default this command returns **all** service principals in a tenant, so for large organizations, it may take a long time to return results. Instead, using one of the optional server-side filtering arguments is recommended:

- `-DisplayNameBeginsWith` requests service principals that have a *prefix* that match the provided value. The

display name of a service principal is the value set with `-DisplayName` during creation.

- `-DisplayName` requests an *exact match* of a service principal name.

Manage service principal roles

Azure PowerShell has the following cmdlets to manage role assignments:

- Get-AzRoleAssignment
- New-AzRoleAssignment
- Remove-AzRoleAssignment

The default role for a service principal is **Contributor**. This role has full permissions to read and write to an Azure account. The **Reader** role is more restrictive, with read-only access. For more information on Role-Based Access Control (RBAC) and roles, see RBAC: Built-in roles.

This example adds the **Reader** role and removes the **Contributor** one:

```
New-AzRoleAssignment -ApplicationId <service principal application ID> -RoleDefinitionName "Reader"
Remove-AzRoleAssignment -ApplicationId <service principal application ID> -RoleDefinitionName "Contributor"
```

> **IMPORTANT**
> Role assignment cmdlets don't take the service principal object ID. They take the associated application ID, which is generated at creation time. To get the application ID for a service principal, use `Get-AzADServicePrincipal`.

> **NOTE**
> If your account doesn't have permission to assign a role, you see an error message that your account "does not have authorization to perform action 'Microsoft.Authorization/roleAssignments/write'." Contact your Azure Active Directory admin to manage roles.

Adding a role *doesn't* restrict previously assigned permissions. When restricting a service principal's permissions, the **Contributor** role should be removed.

The changes can be verified by listing the assigned roles:

```
Get-AzRoleAssignment -ServicePrincipalName ServicePrincipalName
```

Sign in using a service principal

Test the new service principal's credentials and permissions by signing in. To sign in with a service principal, you need the `applicationId` value associated with it, and the tenant it was created under.

To sign in with a service principal using a password:

```
# Use the application ID as the username, and the secret as password
$credentials = Get-Credential
Connect-AzAccount -ServicePrincipal -Credential $credentials -Tenant <tenant ID>
```

Certificate-based authentication requires that Azure PowerShell can retrieve information from a local certificate store based on a certificate thumbprint.

```
Connect-AzAccount -ServicePrincipal -Tenant <tenant ID> -CertificateThumbprint <thumbprint>
```

For instructions on importing a certificate into a credential store accessible by PowerShell, see Sign in with Azure PowerShell

Reset credentials

If you forget the credentials for a service principal, use New-AzADSpCredential to add a new credential. This cmdlet takes the same credential arguments and types as `New-AzADServicePrincipal`. Without any credential arguments, a new `PasswordCredential` with a random password is created.

IMPORTANT

Before assigning any new credentials, you may want to remove existing credentials to prevent sign in with them. To do so, use the Remove-AzADSpCredential cmdlet:

```
Remove-AzADSpCredential -DisplayName ServicePrincipalName
```

```
$newCredential = New-AzADSpCredential -ServicePrincipalName ServicePrincipalName
```

Azure PowerShell context objects

11/4/2019 • 5 minutes to read • Edit Online

Azure PowerShell uses *Azure PowerShell context objects* (Azure contexts) to hold subscription and authentication information. If you have more than one subscription, Azure contexts let you select the subscription to run Azure PowerShell cmdlets on. Azure contexts are also used to store sign-in information across multiple PowerShell sessions and run background tasks.

This article covers managing Azure contexts, not the management of subscriptions or accounts. If you're looking to manage users, subscriptions, tenants, or other account information, see the Azure Active Directory documentation. To learn about using contexts for running background or parallel tasks, see Use Azure PowerShell cmdlets in PowerShell jobs after becoming familiar with Azure contexts.

Overview of Azure context objects

Azure contexts are PowerShell objects representing your active subscription to run commands against, and the authentication information needed to connect to an Azure cloud. With Azure contexts, Azure PowerShell doesn't need to reauthenticate your account each time you switch subscriptions. An Azure context consists of:

- The *account* that was used to sign in to Azure with Connect-AzAccount. Azure contexts treat users, application IDs, and service principals the same from an account perspective.
- The active *subscription*, a service agreement with Microsoft to create and run Azure resources, which are associated with a *tenant*. Tenants are often referred to as *organizations* in documentation or when working with Active Directory.
- A reference to a *token cache*, a stored authentication token for accessing an Azure cloud. Where this token is stored and how long it persists for is determined by the context autosave settings.

For more information on these terms, see Azure Active Directory Terminology. Authentication tokens used by Azure contexts are the same as other stored tokens that are part of a persistent session.

When you sign in with `Connect-AzAccount`, at least one Azure context is created for your default subscription. The object returned by `Connect-AzAccount` is the default Azure context used for the rest of the PowerShell session.

Get Azure contexts

Available Azure contexts are retrieved with the Get-AzContext cmdlet. List all of the available contexts with `-ListAvailable`:

```
Get-AzContext -ListAvailable
```

Or get a context by name:

```
$context = Get-Context -Name "mycontext"
```

Context names may be different from the name of the associated subscription.

> **IMPORTANT**
> The available Azure contexts **aren't** always your available subscriptions. Azure contexts only represent locally-stored information. You can get your subscriptions with the Get-AzSubscription cmdlet.

Create a new Azure context from subscription information

The Set-AzContext cmdlet is used to both create new Azure contexts and set them as the active context. The easiest way to create a new Azure context is to use existing subscription information. The cmdlet is designed to take the output object from `Get-AzSubscription` as a piped value and configure a new Azure context:

```
Get-AzSubscription -SubscriptionName 'MySubscriptionName' | Set-AzContext -Name 'MyContextName'
```

Or give the subscription name or ID and the tenant ID if necessary:

```
Set-AzContext -Name 'MyContextName' -Subscription 'MySubscriptionName' -Tenant '.......'
```

If the `-Name` argument is omitted, then the subscription's name and ID are used as the context name in the format `Subscription Name (subscription-id)`.

Change the active Azure context

Both `Set-AzContext` and Select-AzContext can be used to change the active Azure context. As described in Create a new Azure context, `Set-AzContext` creates a new Azure context for a subscription if one doesn't exist, and then switches to use that context as the active one.

`Select-AzContext` is meant to be used with existing Azure contexts only and works similarly to using `Set-AzContext -Context`, but is designed for use with piping:

```
Set-AzContext -Context $(Get-AzContext -Name "mycontext") # Set a context with an inline Azure context object
Get-AzContext -Name "mycontext" | Select-AzContext # Set a context with a piped Azure context object
```

Like many other account and context management commands in Azure PowerShell, `Set-AzContext` and `Select-AzContext` support the `-Scope` argument so that you can control how long the context is active. `-Scope` lets you change a single session's active context without changing the default:

```
Get-AzContext -Name "mycontext" | Select-AzContext -Scope Process
```

To avoid switching contexts for a whole PowerShell session, all Azure PowerShell commands can be run against a given context with the `-AzContext` argument:

```
$context = Get-AzContext -Name "mycontext"
New-AzVM -Name ExampleVM -AzContext $context
```

The other main use of contexts with Azure PowerShell cmdlets is to run background commands. To learn more about running PowerShell Jobs using Azure PowerShell, see Run Azure PowerShell cmdlets in PowerShell Jobs.

Save Azure contexts across PowerShell sessions

By default, Azure contexts are saved for use between PowerShell sessions. You change this behavior in the

following ways:

- Sign in using `-Scope Process` with `Connect-AzAccount`.

  ```
  Connect-AzAccount -Scope Process
  ```

  The Azure context returned as part of this sign in is valid for the current session *only* and will not be automatically saved, regardless of the Azure PowerShell context autosave setting.

- Disable AzurePowershell's context autosave with the Disable-AzContextAutosave cmdlet. Disabling context autosave **doesn't** clear any stored tokens. To learn how to clear stored Azure context information, see Remove Azure contexts and credentials.

- Explicitly enable Azure context autosave can be enabled with the Enable-AzContextAutosave cmdlet. With autosave enabled, all of a user's contexts are stored locally for later PowerShell sessions.

- Manually save contexts with Save-AzContext to be used in future PowerShell sessions, where they can be loaded with Import-AzContext:

  ```
  Save-AzContext -Path current-context.json # Save the current context
  Save-AzContext -Profile $profileObject -Path other-context.json # Save a context object
  Import-AzContext -Path other-context.json # Load the context from a file and set it to the current
  context
  ```

> **WARNING**
> Disabling context autosave **doesn't** clear any stored context information that was saved. To remove stored information, use the Clear-AzContext cmdlet. For more on removing saved contexts, see Remove contexts and credentials.

Each of these commands supports the `-Scope` parameter, which can take a value of `Process` to only apply to the current running process. For example, to ensure that newly created contexts aren't saved after exiting a PowerShell session:

```
Disable-AzContextAutosave -Scope Process
$context2 = Set-AzContext -Subscription "sub-id" -Tenant "other-tenant"
```

Context information and tokens are stored in the `$env:USERPROFILE\.Azure` directory on Windows, and on `$HOME/.Azure` on other platforms. Sensitive information such as subscription IDs and tenant IDs may still be exposed in stored information, through logs or saved contexts. To learn how to clear stored information, see the Remove contexts and credentials section.

Remove Azure contexts and stored credentials

To clear Azure contexts and credentials:

- Sign out of an account with Disconnect-AzAccount. You can sign out of any account either by account or context:

  ```
  Disconnect-AzAccount # Disconnect active account
  Disconnect-AzAccount -Username "user@contoso.com" # Disconnect by account name

  Disconnect-AzAccount -ContextName "subscription2" # Disconnect by context name
  Disconnect-AzAccount -AzureContext $contextObject # Disconnect using context object information
  ```

Disconnecting always removes stored authentication tokens and clears saved contexts associated with the disconnected user or context.

- Use Clear-AzContext. This cmdlet is guaranteed to always remove stored contexts and authentication tokens, and will also sign you out.
- Remove a context with Remove-AzContext:

```
Remove-AzContext -Name "mycontext" # Remove by name
Get-AzContext -Name "mycontext" | Remove-AzContext # Remove by piping Azure context object
```

If you remove the active context, you will be disconnected from Azure and need to reauthenticate with `Connect-AzAccount`.

See also

- Run Azure PowerShell cmdlets in PowerShell Jobs
- Azure Active Directory Terminology
- Az.Accounts reference

Query output of Azure PowerShell

11/4/2019 • 2 minutes to read • Edit Online

The results of each Azure PowerShell cmdlet are an Azure PowerShell object. Even cmdlets that aren't explicitly `Get-` operations might return a value that can be inspected, to give information about a resource that was created or modified. While most cmdlets return a single object, some return an array that should be iterated through.

In almost all cases, you query output from Azure PowerShell with the Select-Object cmdlet, often abbreviated to `select`. Output can be filtered with Where-Object, or its alias `where`.

Select simple properties

In the default table format, Azure PowerShell cmdlets don't display all of their available properties. You can get the full properties by using the Format-List cmdlet, or by piping output to `Select-Object *`:

```
Get-AzVM -Name TestVM -ResourceGroupName TestGroup | Select-Object *
```

```
ResourceGroupName        : TESTGROUP
Id                       : /subscriptions/711d8ed1-b888-4c52-8ab9-
66f07b87eb6b/resourceGroups/TESTGROUP/providers/Micro
                           soft.Compute/virtualMachines/TestVM
VmId                     : 711d8ed1-b888-4c52-8ab9-66f07b87eb6b
Name                     : TestVM
Type                     : Microsoft.Compute/virtualMachines
Location                 : westus2
LicenseType              :
Tags                     : {}
AvailabilitySetReference :
DiagnosticsProfile       :
Extensions               : {}
HardwareProfile          : Microsoft.Azure.Management.Compute.Models.HardwareProfile
InstanceView             :
NetworkProfile           : Microsoft.Azure.Management.Compute.Models.NetworkProfile
OSProfile                : Microsoft.Azure.Management.Compute.Models.OSProfile
Plan                     :
ProvisioningState        : Succeeded
StorageProfile           : Microsoft.Azure.Management.Compute.Models.StorageProfile
DisplayHint              : Compact
Identity                 :
Zones                    : {}
FullyQualifiedDomainName :
AdditionalCapabilities   :
RequestId                : 711d8ed1-b888-4c52-8ab9-66f07b87eb6b
StatusCode               : OK
```

Once you know the names of the properties that you're interested in, you can use those property names with `Select-Object` to get them directly:

```
Get-AzVM -Name TestVM -ResourceGroupName TestGroup | Select-Object Name,VmId,ProvisioningState
```

```
Name   VmId                                 ProvisioningState
----   ----                                 -----------------
TestVM 711d8ed1-b888-4c52-8ab9-66f07b87eb6b Succeeded
```

Output from using `Select-Object` is always formatted to display the requested information. To learn about using formatting as part of querying cmdlet results, see Format Azure PowerShell cmdlet output.

Select nested properties

Some properties in Azure PowerShell cmdlet output use nested objects, like the `StorageProfile` property of `Get-AzVM` output. To get a value from a nested property, provide a display name and the full path to the value you want to inspect as part of a dictionary argument to `Select-Object`:

```
Get-AzVM -ResourceGroupName TestGroup | `
    Select-Object Name,@{Name="OSType"; Expression={$_.StorageProfile.OSDisk.OSType}}
```

```
Name     OSType
----     ------
TestVM    Linux
TestVM2   Linux
WinVM   Windows
```

Each dictionary argument selects one property from the object. The property to extract must be part of an expression.

Filter results

The `Where-Object` cmdlet allows you to filter the result based on any property value, including nested properties. The next example shows how to use `Where-Object` to find the Linux VMs in a resource group.

```
Get-AzVM -ResourceGroupName TestGroup | `
    Where-Object {$_.StorageProfile.OSDisk.OSType -eq "Linux"}
```

```
ResourceGroupName    Name Location          VmSize OsType        NIC ProvisioningState Zone
-----------------    ---- --------          ------ ------        --- ----------------- ----
TestGroup           TestVM  westus2 Standard_D2s_v3  Linux  testvm299         Succeeded
TestGroup          TestVM2  westus2 Standard_D2s_v3  Linux testvm2669         Succeeded
```

You can pipe the results of `Select-Object` and `Where-Object` to each other. For performance purposes, it's always recommended to put the `Where-Object` operation before `Select-Object`:

```
Get-AzVM -ResourceGroupName TestGroup | `
    Where-Object {$_.StorageProfile.OsDisk.OsType -eq "Linux"} | `
    Select-Object Name,VmID,ProvisioningState
```

```
Name    VmId                                 ProvisioningState
----    ----                                 -----------------
TestVM  711d8ed1-b888-4c52-8ab9-66f07b87eb6  Succeeded
TestVM2 cbcee769-dd78-45e3-a14d-2ad11c647d0  Succeeded
```

Format Azure PowerShell cmdlet output

11/4/2019 • 3 minutes to read • Edit Online

By default each Azure PowerShell cmdlet formats output to be easy to read. PowerShell allows you to convert or format cmdlet output by piping to one of the following cmdlets:

FORMATTING	CONVERSION
Format-Custom	ConvertTo-Csv
Format-List	ConvertTo-Html
Format-Table	ConvertTo-Json
Format-Wide	ConvertTo-Xml

Formatting is used for display in a PowerShell terminal, and conversion is used for generating data to be consumed by other scripts or programs.

Table output format

By default, Azure PowerShell cmdlets output in the table format. This format doesn't display all information of the requested resource:

```
Get-AzVM
```

```
ResourceGroupName           Name Location          VmSize  OsType              NIC ProvisioningState Zone
-----------------           ---- --------          ------  ------              --- ----------------- ----
QueryExample      ExampleLinuxVM  westus2        Basic_A0   Linux examplelinuxvm916         Succeeded
QueryExample         RHELExample  westus2  Standard_D2_v3   Linux    rhelexample469         Succeeded
QueryExample        WinExampleVM  westus2 Standard_DS1_v2 Windows   winexamplevm268         Succeeded
```

The amount of data displayed by `Format-Table` can be affected by the width of your PowerShell session window. To restrict the output to specific properties and order them, property names can be provided as arguments to `Format-Table`:

```
Get-AzVM -ResourceGroupName QueryExample | Format-Table Name,ResourceGroupName,Location
```

```
Name           ResourceGroupName Location
----           ----------------- --------
ExampleLinuxVM QueryExample      westus2
RHELExample    QueryExample      westus2
WinExampleVM   QueryExample      westus2
```

List output format

List output format produces two columns, property names followed by the value. For complex objects, the type of the object is displayed instead.

```
Get-AzVM | Format-List
```

The following output has some fields removed.

```
ResourceGroupName        : QueryExample
Id                       :
/subscriptions/.../resourceGroups/QueryExample/providers/Microsoft.Compute/virtualMachines/ExampleLinuxVM
VmId                     : ...
Name                     : ExampleLinuxVM
Type                     : Microsoft.Compute/virtualMachines
Location                 : westus2
...
HardwareProfile          : Microsoft.Azure.Management.Compute.Models.HardwareProfile
InstanceView             :
NetworkProfile           : Microsoft.Azure.Management.Compute.Models.NetworkProfile
OSProfile                : Microsoft.Azure.Management.Compute.Models.OSProfile
...
StatusCode               : OK

ResourceGroupName        : QueryExample
Id                       :
/subscriptions/.../resourceGroups/QueryExample/providers/Microsoft.Compute/virtualMachines/RHELExample
VmId                     : ...
Name                     : RHELExample
Type                     : Microsoft.Compute/virtualMachines
Location                 : westus2
...
```

Like `Format-Table`, property names can be provided to order and restrict the output:

```
Get-AzVM | Format-List ResourceGroupName,Name,Location
```

```
ResourceGroupName : QueryExample
Name              : ExampleLinuxVM
Location          : westus2

ResourceGroupName : QueryExample
Name              : RHELExample
Location          : westus2

ResourceGroupName : QueryExample
Name              : WinExampleVM
Location          : westus2
```

Wide output format

Wide output format produces only one property name per query. Which property is displayed can be controlled by giving a property as an argument.

```
Get-AzVM | Format-Wide
```

```
ExampleLinuxVM                              RHELExample
WinExampleVM
```

```
Get-AzVM | Format-Wide ResourceGroupName
```

```
QueryExample                                QueryExample
QueryExample
```

Custom output format

The `Custom-Format` output type is meant for formatting custom objects. Without any arguments, it behaves like `Format-List` but displays the property names of custom classes.

```
Get-AzVM | Format-Custom
```

The following output has some fields removed.

```
ResourceGroupName : QueryExample
Id                :
/subscriptions/.../resourceGroups/QueryExample/providers/Microsoft.Compute/virtualMachines/ExampleLinuxVM
VmId              : ...
Name              : ExampleLinuxVM
Type              : Microsoft.Compute/virtualMachines
Location          : westus2
Tags              : {}
HardwareProfile   : {VmSize}
NetworkProfile    : {NetworkInterfaces}
OSProfile         : {ComputerName, AdminUsername, LinuxConfiguration, Secrets,
AllowExtensionOperations}
ProvisioningState : Succeeded
StorageProfile    : {ImageReference, OsDisk, DataDisks}
...
```

Giving property names as arguments to `Custom-Format` displays the property/value pairs for custom objects set as values:

```
Get-AzVM | Format-Custom Name,ResourceGroupName,Location,OSProfile
```

The following output has some fields removed.

```
class PSVirtualMachineList
{
  Name = ExampleLinuxVM
  ResourceGroupName = QueryExample
  Location = westus2
  OSProfile =
    class OSProfile
    {
      ComputerName = ExampleLinuxVM
      AdminUsername = ...
      AdminPassword =
      CustomData =
      WindowsConfiguration =
      LinuxConfiguration =
        class LinuxConfiguration
        {
          DisablePasswordAuthentication = False
          Ssh =
          ProvisionVMAgent = True
        }
      Secrets =
        [
        ]

      AllowExtensionOperations = True
    }
}

...

class PSVirtualMachineList
{
  Name = WinExampleVM
  ResourceGroupName = QueryExample
  Location = westus2
  OSProfile =
    class OSProfile
    {
      ComputerName = WinExampleVM
      AdminUsername = ...
      AdminPassword =
      CustomData =
      WindowsConfiguration =
        class WindowsConfiguration
        {
          ProvisionVMAgent = True
          EnableAutomaticUpdates = True
          TimeZone =
          AdditionalUnattendContent =
          WinRM =
        }
      LinuxConfiguration =
      Secrets =
        [
        ]

      AllowExtensionOperations = True
    }
}
```

Conversion to other data formats

The `ConvertTo-*` family of cmdlets allows for converting the results of Azure PowerShell cmdlets to machine-readable formats. To get only some properties from the Azure PowerShell results, use the `Select-Object` command in a pipe before performing the conversion. The following examples demonstrate the different kinds of

output that each conversion produces.

Conversion to CSV

```
Get-AzVM | ConvertTo-CSV
```

```
#TYPE Microsoft.Azure.Commands.Compute.Models.PSVirtualMachineList
"ResourceGroupName","Id","VmId","Name","Type","Location","LicenseType","Tags","AvailabilitySetReference","Diag
nosticsProfile","Extensions","HardwareProfile","InstanceView","NetworkProfile","OSProfile","Plan","Provisionin
gState","StorageProfile","DisplayHint","Identity","Zones","FullyQualifiedDomainName","AdditionalCapabilities",
"RequestId","StatusCode"
"QUERYEXAMPLE","/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/Exa
mpleLinuxVM","...","ExampleLinuxVM","Microsoft.Compute/virtualMachines","westus2",,"System.Collections.Generic
.Dictionary`2[System.String,System.String]",,,"System.Collections.Generic.List`1[Microsoft.Azure.Management.Co
mpute.Models.VirtualMachineExtension]","Microsoft.Azure.Management.Compute.Models.HardwareProfile",,"Microsoft
.Azure.Management.Compute.Models.NetworkProfile","Microsoft.Azure.Management.Compute.Models.OSProfile",,"Succe
eded","Microsoft.Azure.Management.Compute.Models.StorageProfile","Compact",,"System.Collections.Generic.List`1
[System.String]",,,"...","OK"
"QUERYEXAMPLE","/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/RHE
LExample","...","RHELExample","Microsoft.Compute/virtualMachines","westus2",,"System.Collections.Generic.Dicti
onary`2[System.String,System.String]",,,"System.Collections.Generic.List`1[Microsoft.Azure.Management.Compute.
Models.VirtualMachineExtension]","Microsoft.Azure.Management.Compute.Models.HardwareProfile",,"Microsoft.Azure
.Management.Compute.Models.NetworkProfile","Microsoft.Azure.Management.Compute.Models.OSProfile",,"Succeeded",
"Microsoft.Azure.Management.Compute.Models.StorageProfile","Compact",,"System.Collections.Generic.List`1[Syste
m.String]",,,"...","OK"
"QUERYEXAMPLE","/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/Win
ExampleVM","...","WinExampleVM","Microsoft.Compute/virtualMachines","westus2",,"System.Collections.Generic.Dic
tionary`2[System.String,System.String]",,,"System.Collections.Generic.List`1[Microsoft.Azure.Management.Comput
e.Models.VirtualMachineExtension]","Microsoft.Azure.Management.Compute.Models.HardwareProfile",,"Microsoft.Azu
re.Management.Compute.Models.NetworkProfile","Microsoft.Azure.Management.Compute.Models.OSProfile",,"Succeeded
","Microsoft.Azure.Management.Compute.Models.StorageProfile","Compact",,"System.Collections.Generic.List`1[Sys
tem.String]",,,"...","OK"
```

Conversion to JSON

JSON output doesn't expand all properties by default. To change the depth of properties expanded, use the `-Depth` argument. By default, the expansion depth is `2`.

```
Get-AzVM|ConvertTo-JSON
```

The following output has some fields removed.

```
[
    {
        "ResourceGroupName":  "QUERYEXAMPLE",
        "Id":
"/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/ExampleLinuxVM",
        "VmId":  "...",
        "Name":  "ExampleLinuxVM",
        "Type":  "Microsoft.Compute/virtualMachines",
        "Location":  "westus2",
        ...
        "OSProfile":  {
                          "ComputerName":  "ExampleLinuxVM",
                          "AdminUsername":  "...",
                          "AdminPassword":  null,
                          "CustomData":  null,
                          "WindowsConfiguration":  null,
                          "LinuxConfiguration":
"Microsoft.Azure.Management.Compute.Models.LinuxConfiguration",
                          "Secrets":  "",
                          "AllowExtensionOperations":  true
                      },
        "Plan":  null,
        "ProvisioningState":  "Succeeded",
        "StorageProfile":  {
                               "ImageReference":  "Microsoft.Azure.Management.Compute.Models.ImageReference",
                               "OsDisk":  "Microsoft.Azure.Management.Compute.Models.OSDisk",
                               "DataDisks":  ""
                           },
        "DisplayHint":  0,
        "Identity":  null,
        "Zones":  [

                  ],
        "FullyQualifiedDomainName":  null,
        "AdditionalCapabilities":  null,
        "RequestId":  "...",
        "StatusCode":  200
    },
    ...
]
```

Conversion to XML

The `ConvertTo-XML` cmdlet converts the Azure PowerShell response object into a pure XML object, which can be handled like any other XML object within PowerShell.

```
Get-AzVM | ConvertTo-XML
```

```
xml                          Objects
---                          -------
version="1.0" encoding="utf-8" Objects
```

Conversion to HTML

Converting an object to HTML produces output that will be rendered as an HTML table. Rendering of the HTML will depend on your browser behavior for rendering tables which contain no width information. No custom class objects are expanded.

```
Get-AzVM | ConvertTo-HTML
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>HTML TABLE</title>
</head><body>
<table>
<colgroup><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/><col/>
<col/><col/><col/><col/><col/><col/><col/><col/><col/></colgroup>
<tr><th>ResourceGroupName</th><th>Id</th><th>VmId</th><th>Name</th><th>Type</th><th>Location</th>
<th>LicenseType</th><th>Tags</th><th>AvailabilitySetReference</th><th>DiagnosticsProfile</th>
<th>Extensions</th><th>HardwareProfile</th><th>InstanceView</th><th>NetworkProfile</th><th>OSProfile</th>
<th>Plan</th><th>ProvisioningState</th><th>StorageProfile</th><th>DisplayHint</th><th>Identity</th>
<th>Zones</th><th>FullyQualifiedDomainName</th><th>AdditionalCapabilities</th><th>RequestId</th>
<th>StatusCode</th></tr>
<tr><td>QUERYEXAMPLE</td>
<td>/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/ExampleLinuxVM<
/td><td>...</td><td>ExampleLinuxVM</td><td>Microsoft.Compute/virtualMachines</td><td>westus2</td><td></td>
<td>System.Collections.Generic.Dictionary`2[System.String,System.String]</td><td></td><td></td>
<td>System.Collections.Generic.List`1[Microsoft.Azure.Management.Compute.Models.VirtualMachineExtension]</td>
<td>Microsoft.Azure.Management.Compute.Models.HardwareProfile</td><td></td>
<td>Microsoft.Azure.Management.Compute.Models.NetworkProfile</td>
<td>Microsoft.Azure.Management.Compute.Models.OSProfile</td><td></td><td>Succeeded</td>
<td>Microsoft.Azure.Management.Compute.Models.StorageProfile</td><td>Compact</td><td></td>
<td>System.Collections.Generic.List`1[System.String]</td><td></td><td></td><td></td><td>...</td><td>OK</td></tr>
<tr><td>QUERYEXAMPLE</td>
<td>/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/RHELExample</td
><td>...</td><td>RHELExample</td><td>Microsoft.Compute/virtualMachines</td><td>westus2</td><td></td>
<td>System.Collections.Generic.Dictionary`2[System.String,System.String]</td><td></td><td></td>
<td>System.Collections.Generic.List`1[Microsoft.Azure.Management.Compute.Models.VirtualMachineExtension]</td>
<td>Microsoft.Azure.Management.Compute.Models.HardwareProfile</td><td></td>
<td>Microsoft.Azure.Management.Compute.Models.NetworkProfile</td>
<td>Microsoft.Azure.Management.Compute.Models.OSProfile</td><td></td><td>Succeeded</td>
<td>Microsoft.Azure.Management.Compute.Models.StorageProfile</td><td>Compact</td><td></td>
<td>System.Collections.Generic.List`1[System.String]</td><td></td><td></td><td></td><td>...</td><td>OK</td></tr>
<tr><td>QUERYEXAMPLE</td>
<td>/subscriptions/.../resourceGroups/QUERYEXAMPLE/providers/Microsoft.Compute/virtualMachines/WinExampleVM</t
d><td>...</td><td>WinExampleVM</td><td>Microsoft.Compute/virtualMachines</td><td>westus2</td><td></td>
<td>System.Collections.Generic.Dictionary`2[System.String,System.String]</td><td></td><td></td>
<td>System.Collections.Generic.List`1[Microsoft.Azure.Management.Compute.Models.VirtualMachineExtension]</td>
<td>Microsoft.Azure.Management.Compute.Models.HardwareProfile</td><td></td>
<td>Microsoft.Azure.Management.Compute.Models.NetworkProfile</td>
<td>Microsoft.Azure.Management.Compute.Models.OSProfile</td><td></td><td>Succeeded</td>
<td>Microsoft.Azure.Management.Compute.Models.StorageProfile</td><td>Compact</td><td></td>
<td>System.Collections.Generic.List`1[System.String]</td><td></td><td></td><td></td><td>...</td><td>OK</td></tr>
</table>
</body></html>
```

Use multiple Azure subscriptions

11/4/2019 • 2 minutes to read • Edit Online

Most Azure users will only ever have a single subscription. However, if you are part of more than one organization or your organization has divided up access to certain resources across groupings, you may have multiple subscriptions within Azure. The CLI supports selecting a subscription both globally and per command.

For detailed information on subscriptions, billing, and cost management, see the billing and cost management documentation.

Tenants, users, and subscriptions

You might have some confusion over the difference between tenants, users, and subscriptions within Azure. A *tenant* is the Azure Active Directory entity that encompasses a whole organization. This tenant has at least one *subscription* and *user*. A user is an individual and is associated with only one tenant, the organization that they belong to. Users are those accounts that sign in to Azure to create, manage, and use resources. A user may have access to multiple *subscriptions*, which are the agreements with Microsoft to use cloud services, including Azure. Every resource is associated with a subscription.

To learn more about the differences between tenants, users, and subscriptions, see the Azure cloud terminology dictionary. To learn how to add a new subscription to your Azure Active Directory tenant, see How to add an Azure subscription to Azure Active Directory. To learn how to sign in to a specific tenant, see Sign in with Azure PowerShell.

Change the active subscription

In Azure PowerShell, accessing the resources for a subscription requires changing the subscription associated with your current Azure session. This is done by modifying the active session context, the information about which tenant, subscription, and user cmdlets should be run against. In order to change subscriptions, an Azure PowerShell Context object first needs to be retrieved with Get-AzSubscription and then the current context changed with Set-AzContext.

The next example shows how to get a subscription in the currently active tenant, and set it as the active session:

```
$context = Get-AzSubscription -SubscriptionId ...
Set-AzContext $context
```

To learn more about Azure PowerShell contexts, including how to save them and quickly switch between them for working with multiple subscriptions easily, see Persist credentials with Azure PowerShell contexts.

Run Azure PowerShell cmdlets in PowerShell Jobs

11/4/2019 • 2 minutes to read • Edit Online

Azure PowerShell depends on connecting to an Azure cloud and waiting for responses, so most of these cmdlets block your PowerShell session until they get a response from the cloud. Powershell Jobs let you run cmdlets in the background or do multiple tasks on Azure at once, from inside a single PowerShell session.

This article is a brief overview of how to run Azure PowerShell cmdlets as PowerShell Jobs and check for completion. Running commands in Azure PowerShell requires the use of Azure PowerShell contexts, which are covered in detail in Azure contexts and sign-in credentials. To learn more about PowerShell Jobs, see About PowerShell Jobs.

Azure contexts with PowerShell jobs

PowerShell Jobs are run as separate processes without an attached PowerShell session, so your Azure credentials must be shared with them. Credentials are passed as Azure context objects, using one of these methods:

- Automatic context persistence. Context persistence is enabled by default and preserves your sign-in information across multiple sessions. With context persistence enabled, the current Azure context is passed to the new process:

  ```
  Enable-AzContextAutosave # Enables context autosave if not already on
  $creds = Get-Credential
  $job = Start-Job { param($vmadmin) New-AzVM -Name MyVm -Credential $vmadmin } -ArgumentList $creds
  ```

- Use the `-AzContext` parameter with any Azure PowerShell cmdlets to provide an Azure context object:

  ```
  $context = Get-AzContext -Name 'mycontext' # Get an Azure context object
  $creds = Get-Credential
  $job = Start-Job { param($context, $vmadmin) New-AzVM -Name MyVm -AzContext $context -Credential
  $vmadmin} -ArgumentList $context,$creds }
  ```

  If context persistence is disabled, the `-AzContext` argument is required.

- Use the `-AsJob` switch provided by some Azure PowerShell cmdlets. This switch automatically starts the cmdlet as a PowerShell Job, using the currently active Azure context:

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

  To see if a cmdlet supports `-AsJob`, check its reference documentation. The `-AsJob` switch doesn't require context autosave to be enabled.

You can check the status of a running job with the Get-Job cmdlet. To get the output from a job so far, use the Receive-Job cmdlet.

To check an operation's progress remotely on Azure, use the `Get-` cmdlets associated with the type of resource being modified by the job:

```
$creds = Get-Credential
$context = Get-AzContext -Name 'mycontext'
$vmName = "MyVm"

$job = Start-Job { param($context, $vmName, $vmadmin) New-AzVM -Name $vmName -AzContext $context -Credential
$vmadmin} -ArgumentList $context,$vmName,$creds }

Get-Job $job
Get-AzVM -Name $vmName
```

See Also

- Azure PowerShell contexts
- About PowerShell Jobs
- Get-Job reference
- Receive-Job reference

3.0.0 - November 2019

11/4/2019 • 49 minutes to read • Edit Online

General

- Az.PrivateDns 1.0.0 released

Az.Accounts

- Add a deprecation message for 'Resolve-Error' alias.

Az.Advisor

- Added new category 'Operational Excellence' to Get-AzAdvisorRecommendation cmdlet.

Az.Batch

- Renamed `CoreQuota` on `BatchAccountContext` to `DedicatedCoreQuota`. There is also a new `LowPriorityCoreQuota`.
  - This impacts **Get-AzBatchAccount**.
- **New-AzBatchTask** `-ResourceFile` parameter now takes a collection of `PSResourceFile` objects, which can be constructed using the new **New-AzBatchResourceFile** cmdlet.
- New **New-AzBatchResourceFile** cmdlet to help create `PSResourceFile` objects. These can be supplied to **New-AzBatchTask** on the `-ResourceFile` parameter.
  - This supports two new kinds of resource file in addition to the existing `HttpUrl` way:
    - `AutoStorageContainerName` based resource files download an entire auto-storage container to the Batch node.
    - `StorageContainerUrl` based resource files download the container specified in the URL to the Batch node.
- Removed `ApplicationPackages` property of `PSApplication` returned by **Get-AzBatchApplication**.
  - The specific packages inside of an application now can be retrieved using **Get-AzBatchApplicationPackage**. For example:

    ```
    Get-AzBatchApplication -AccountName myaccount -ResourceGroupName myresourcegroup -ApplicationId
    myapplication
    ```

    .
- Renamed `ApplicationId` to `ApplicationName` on **Get-AzBatchApplicationPackage**, **New-AzBatchApplicationPackage**, **Remove-AzBatchApplicationPackage**, **Get-AzBatchApplication**, **New-AzBatchApplication**, **Remove-AzBatchApplication**, and **Set-AzBatchApplication**.
  - `ApplicationId` now is an alias of `ApplicationName`.
- Added new `PSWindowsUserConfiguration` property to `PSUserAccount`.
- Renamed `Version` to `Name` on `PSApplicationPackage`.
- Renamed `BlobSource` to `HttpUrl` on `PSResourceFile`.
- Removed `OSDisk` property from `PSVirtualMachineConfiguration`.
- Removed **Set-AzBatchPoolOSVersion**. This operation is no longer supported.
- Removed `TargetOSVersion` from `PSCloudServiceConfiguration`.
- Renamed `CurrentOSVersion` to `OSVersion` on `PSCloudServiceConfiguration`.
- Removed `DataEgressGiB` and `DataIngressGiB` from `PSPoolUsageMetrics`.
- Removed **Get-AzBatchNodeAgentSku** and replaced it with **Get-AzBatchSupportedImage**.
  - **Get-AzBatchSupportedImage** returns the same data as **Get-AzBatchNodeAgentSku** but in a more friendly format.
  - New non-verified images are also now returned. Additional information about `Capabilities` and `BatchSupportEndOfLife` for each image is also included.

- Added ability to mount remote file-systems on each node of a pool via the new `MountConfiguration` parameter of **New-AzBatchPool**.
- Now support network security rules blocking network access to a pool based on the source port of the traffic. This is done via the `SourcePortRanges` property on `PSNetworkSecurityGroupRule`.
- When running a container, Batch now supports executing the task in the container working directory or in the Batch task working directory. This is controlled by the `WorkingDirectory` property on `PSTaskContainerSettings`.
- Added ability to specify a collection of public IPs on `PSNetworkConfiguration` via the new `PublicIPs` property. This guarantees nodes in the Pool will have an IP from the list user provided IPs.
- When not specified, the default value of `WaitForSuccess` on `PSSTartTask` is now `$True` (was `$False`).
- When not specified, the default value of `Scope` on `PSAutoUserSpecification` is now `Pool` (was `Task` on Windows and `Pool` on Linux).

Az.Cdn

- Introduced UrlRewriteAction and CacheKeyQueryStringAction to RulesEngine.
- Fixed several bugs like missing 'Selector' Input in New-AzDeliveryRuleCondition cmdlet.

Az.Compute

- Disk Encryption Set feature
  - New cmdlets: New-AzDiskEncryptionSetConfig New-AzDiskEncryptionSet Get-AzDiskEncryptionSet Remove-AzDiskEncryptionSet
  - DiskEncryptionSetId parameter is added to the following cmdlets: Set-AzImageOSDisk Set-AzVMOSDisk Set-AzVmssStorageProfile
    Add-AzImageDataDisk New-AzVMDataDisk Set-AzVMDataDisk Add-AzVMDataDisk Add-AzVmssDataDisk Add-AzVmssVMDataDisk
  - DiskEncryptionSetId and EncryptionType parameters are added to the following cmdlets: New-AzDiskConfig New-AzSnapshotConfig
- Add PublicIPAddressVersion parameter to New-AzVmssIPConfig
- Move FileUris of custom script extension from public setting to protected setting
- Add ScaleInPolicy to New-AzVmss, New-AzVmssConfig and Update-AzVmss cmdlets
- Breaking changes
  - UploadSizeInBytes parameter is used instead of DiskSizeGB for New-AzDiskConfig when CreateOption is Upload
  - PublishingProfile.Source.ManagedImage.Id is replaced with StorageProfile.Source.Id in GalleryImageVersion object

Az.DataFactory

- Update ADF .Net SDK version to 4.3.0

Az.DataLakeStore

- Update ADLS SDK version (https://github.com/Azure/azure-data-lake-store-net/blob/preview-alpha/CHANGELOG.md#version-123-alpha), brings following fixes
- Avoid throwing exception while unable to deserialize the creationtime of the trash or directory entry.
- Expose setting per request timeout in adlsclient
- Fix passing the original syncflag for badoffset recovery
- Fix EnumerateDirectory to retrieve continuation token once response is checked
- Fix Concat Bug

Az.FrontDoor

- Fixed miscellaneous typos across module

Az.HDInsight

- Fixed the bug that customer will get 'Not a valid Base-64 string' error when using Get-AzHDInsightCluster to

get the cluster with ADLSGen1 storage.

- Add a parameter named 'ApplicationId' to three cmdlets Add-AzHDInsightClusterIdentity, New-AzHDInsightClusterConfig and New-AzHDInsightCluster so that customer can provide the service principal application id for accessing Azure Data Lake.
- Changed Microsoft.Azure.Management.HDInsight from 2.1.0 to 5.1.0
- Removed five cmdlets:
  - Get-AzHDInsightOMS
  - Enable-AzHDInsightOMS
  - Disable-AzHDInsightOMS
  - Grant-AzHDInsightRdpServicesAccess
  - Revoke-AzHDInsightRdpServicesAccess
- Added three cmdlets:
  - Get-AzHDInsightMonitoring to replace Get-AzHDInsightOMS.
  - Enable-AzHDInsightMonitoring to replace Enable-AzHDInsightOMS.
  - Disable-AzHDInsightMonitoring to replace Disable-AzHDInsightOMS.
- Fixed cmdlet Get-AzHDInsightProperties to support get capabilities information from a specific location.
- Removed parameter sets('Spark1', 'Spark2') from Add-AzHDInsightConfigValue.
- Add examples to the help documents of cmdlet Add-AzHDInsightSecurityProfile.
- Changed output type of the following cmdlets:
- - Changed the output type of Get-AzHDInsightProperties from CapabilitiesResponse to AzureHDInsightCapabilities.
- - Changed the output type of Remove-AzHDInsightCluster from ClusterGetResponse to bool.
- - Changed the output type of Set-AzHDInsightGatewaySettings HttpConnectivitySettings to GatewaySettings.
- Added some scenario test cases.
- Remove some alias: 'Add-AzHDInsightConfigValues', 'Get-AzHDInsightProperties'.

Az.IotHub

- Breaking changes:
  - The cmdlet 'Add-AzIotHubEventHubConsumerGroup' no longer supports the parameter 'EventHubEndpointName' and no alias was found for the original parameter name.
  - The parameter set '__AllParameterSets' for cmdlet 'Add-AzIotHubEventHubConsumerGroup' has been removed.
  - The cmdlet 'Get-AzIotHubEventHubConsumerGroup' no longer supports the parameter 'EventHubEndpointName' and no alias was found for the original parameter name.
  - The parameter set '__AllParameterSets' for cmdlet 'Get-AzIotHubEventHubConsumerGroup' has been removed.
  - The property 'OperationsMonitoringProperties' of type 'Microsoft.Azure.Commands.Management.IotHub.Models.PSIotHubProperties' has been removed.
  - The property 'OperationsMonitoringProperties' of type 'Microsoft.Azure.Commands.Management.IotHub.Models.PSIotHubInputProperties' has been removed.
  - The cmdlet 'New-AzIotHubExportDevice' no longer supports the alias 'New-AzIotHubExportDevices'.
  - The cmdlet 'New-AzIotHubImportDevice' no longer supports the alias 'New-AzIotHubImportDevices'.
  - The cmdlet 'Remove-AzIotHubEventHubConsumerGroup' no longer supports the parameter 'EventHubEndpointName' and no alias was found for the original parameter name.
  - The parameter set '__AllParameterSets' for cmdlet 'Remove-AzIotHubEventHubConsumerGroup' has been removed.
  - The cmdlet 'Set-AzIotHub' no longer supports the parameter 'OperationsMonitoringProperties' and no alias was found for the original parameter name.

- The parameter set 'UpdateOperationsMonitoringProperties' for cmdlet 'Set-AzIotHub' has been removed.

Az.RecoveryServices

- Azure Site Recovery support to configure networking resources like NSG, public IP and internal load balancers for Azure to Azure.
- Azure Site Recovery Support to write to managed disk for vMWare to Azure.
- Azure Site Recovery Support to NIC reduction for vMWare to Azure.
- Azure Site Recovery Support to accelerated networking for Azure to Azure.
- Azure Site Recovery Support to agent auto update for Azure to Azure.
- Azure Site Recovery Support to Standard SSD for Azure to Azure.
- Azure Site Recovery Support to Azure Disk Encryption two pass for Azure to Azure.
- Azure Site Recovery Support to protect newly added disk for Azure to Azure.
- Added SoftDelete feature for VM and added tests for softdelete

Az.Resources

- Update dependency assembly Microsoft.Extensions.Caching.Memory from 1.1.1 to 2.2

Az.Network

- Change all cmdlets for PrivateEndpointConnection to support generic service provider.
  - Updated cmdlet:
    - Approve-AzPrivateEndpointConnection
    - Deny-AzPrivateEndpointConnection
    - Get-AzPrivateEndpointConnection
    - Remove-AzPrivateEndpointConnection
    - Set-AzPrivateEndpointConnection
- Add new cmdlet for PrivateLinkResource and it also support generic service provider.
  - New cmdlet:
    - Get-AzPrivateLinkResource
- Add new fields and parameter for the feature Proxy Protocol V2.
  - Add property EnableProxyProtocol in PrivateLinkService
  - Add property LinkIdentifier in PrivateEndpointConnection
  - Updated New-AzPrivateLinkService to add a new optional parameter EnableProxyProtocol.
- Fix incorrect parameter description in 'New-AzApplicationGatewaySku' reference documentation
- New cmdlets to support the azure firewall policy
- Add support for child resource RouteTables of VirtualHub
  - New cmdlets added:
    - Add-AzVirtualHubRoute
    - Add-AzVirtualHubRouteTable
    - Get-AzVirtualHubRouteTable
    - Remove-AzVirtualHubRouteTable
    - Set-AzVirtualHub
- Add support for new properties Sku of VirtualHub and VirtualWANType of VirtualWan
  - Cmdlets updated with optional parameters:
    - New-AzVirtualHub : added parameter Sku
    - Update-AzVirtualHub : added parameter Sku
    - New-AzVirtualWan : added parameter VirtualWANType
    - Update-AzVirtualWan : added parameter VirtualWANType
- Add support for EnableInternetSecurity property for HubVnetConnection, VpnConnection and

ExpressRouteConnection
    - New cmdlets added:
        - Update-AzureRmVirtualHubVnetConnection
    - Cmdlets updated with optional parameters:
        - New-AzureRmVirtualHubVnetConnection : added parameter EnableInternetSecurity
        - New-AzureRmVpnConnection : added parameter EnableInternetSecurity
        - Update-AzureRmVpnConnection : added parameter EnableInternetSecurity
        - New-AzureRmExpressRouteConnection : added parameter EnableInternetSecurity
        - Set-AzureRmExpressRouteConnection : added parameter EnableInternetSecurity
- Add support for Configuring TopLevel WebApplicationFirewall Policy
    - New cmdlets added:
        - New-AzApplicationGatewayFirewallPolicySetting
        - New-AzApplicationGatewayFirewallPolicyExclusion
        - New-AzApplicationGatewayFirewallPolicyManagedRuleGroupOverride
        - New-AzApplicationGatewayFirewallPolicyManagedRuleOverride
        - New-AzApplicationGatewayFirewallPolicyManagedRule
        - New-AzApplicationGatewayFirewallPolicyManagedRuleSet
    - Cmdlets updated with optional parameters:
        - New-AzApplicationGatewayFirewallPolicy : added parameter PolicySetting, ManagedRule
- Added support for Geo-Match operator on CustomRule
    - Added GeoMatch to the operator on the FirewallCondition
- Added support for perListener and perSite Firewall policy
    - Cmdlets updated with optional parameters:
        - New-AzApplicationGatewayHttpListener : added parameter FirewallPolicy, FirewallPolicyId
        - New-AzApplicationGatewayPathRuleConfig : added parameter FirewallPolicy, FirewallPolicyId
- Fix required subnet with name AzureBastionSubnet in 'PSBastion' can be case insensitive
- Support for Destination FQDNs in Network Rules and Translated FQDN in NAT Rules for Azure Firewall
- Add support for top level resource RouteTables of IpGroup
    - New cmdlets added:
        - New-AzIpGroup
        - Remove-AzIpGroup
        - Get-AzIpGroup
        - Set-AzIpGroup

Az.ServiceFabric

- Remove Add-AzServiceFabricApplicationCertificate cmdlet as this scenario is covered by Add-AzVmssSecret.

Az.Sql

- Added support for restore of dropped databases on Managed Instances.
- Deprecated from code old auditing cmdlets.
- Removed deprecated aliases:
- Get-AzSqlDatabaseIndexRecommendations (use Get-AzSqlDatabaseIndexRecommendation instead)
- Get-AzSqlDatabaseRestorePoints (use Get-AzSqlDatabaseRestorePoint instead)
- Remove Get-AzSqlDatabaseSecureConnectionPolicy cmdlet
- Remove aliases for deprecated Vulnerability Assessment Settings cmdlets
- Deprecate Advanced Threat Detection Settings cmdlets
- Adding cmdlets to Disable and enable sensitivity recommendations on columns in a database.

Az.Storage

- Support enable Large File share when create or update Storage account
  - New-AzStorageAccount
  - Set-AzStorageAccount
- When close/get File handle, skip check the input path is File directory or File, to avoid failure with object in DeletePending status
  - Get-AzStorageFileHandle
  - Close-AzStorageFileHandle

2.8.0 - October 2019

General

- Az.HealthcareApis 1.0.0 release

Az.Accounts

- Update telemetry and url rewriting for generated modules, fix windows unit tests.

Az.ApiManagement

- **Set-AzApiManagementApi** - Added support for Updating Api into ApiVersionSet
  - Fix for issue https://github.com/Azure/azure-powershell/issues/10068

Az.Automation

- Fixed New-AzureAutomationSoftwareUpdateConfiguration cmdlet for Linux reboot setting parameter.

Az.Batch

- **Get-AzBatchNodeAgentSku** is deprecated and will be replaced by **Get-AzBatchSupportImage** in version 2.0.0.

Az.Compute

- Add Priority, EvictionPolicy, and MaxPrice parameters to New-AzVM and New-AzVmss cmdlets
- Fix warning message and help document for Add-AzVMAdditionalUnattendContent and Add-AzVMSshPublicKey cmdlets
- Fix -skipVmBackup exception for Linux VMs with managed disks for Set-AzVMDiskEncryptionExtension.
- Fix bug in update encryption settings in Set-AzVMDiskEncryptionExtension, two pass scenario.

Az.DataFactory

- Adding CRUD commands for ADF V2 data flow: Set-AzDataFactoryV2DataFlow, Remove-AzDataFactoryV2DataFlow, and Get-AzDataFactoryV2DataFlow.
- Adding action commands for ADF V2 data flow debug Session: Start-AzDataFactoryV2DataFlowDebugSession, Get-AzDataFactoryV2DataFlowDebugSession, Add-AzDataFactoryV2DataFlowDebugSessionPackage, Invoke-AzDataFactoryV2DataFlowDebugSessionCommand and Stop-AzDataFactoryV2DataFlowDebugSession.
- Update ADF .Net SDK version to 4.2.0

Az.DataLakeStore

- Fix account validation so that accounts with '-' can be passed without domain

Az.HealthcareApis

- Updated the powershell version to 1.0.0
- Updated the SDK version to 1.0.2
- Update in tests to refer to new SDK version
- Updated the output structure from nested to flattened.

Az.IotHub

- Add new routing source: DigitalTwinChangeEvents
- Minor bug fix: Get-Azlothub not returning subscriptionId

Az.Monitor

- New action group receivers added for action group -ItsmReceiver -VoiceReceiver -ArmRoleReceiver -AzureFunctionReceiver -LogicAppReceiver -AutomationRunbookReceiver -AzureAppPushReceiver
- Use common alert schema enabled for the receivers. This is not applicable for SMS, Azure App push , ITSM and Voice recievers
- Webhooks now supports Azure active directory authentication .

Az.Network

- Add new cmdlet Get-AzAvailableServiceAlias which can be called to get the aliases that can be used for Service Endpoint Policies.
- Added support for the adding traffic selectors to Virtual Network Gateway Connections
  - New cmdlets added:
    - New-AzureRmTrafficSelectorPolicy
  - Cmdlets updated with optional parameter -TrafficSelectorPolicies -New-AzureRmVirtualNetworkGatewayConnection -Set-AzureRmVirtualNetworkGatewayConnection
- Add support for ESP and AH protocols in network security rule configurations
  - Updated cmdlets:
    - Add-AzNetworkSecurityRuleConfig
    - New-AzNetworkSecurityRuleConfig
    - Set-AzNetworkSecurityRuleConfig
- Improve handling of exceptions in Cortex cmdlets
- New Generations and SKUs for VirtualNetworkGateways
  - Introduce new Generations for VirtualNetworkGateways.
  - Introduce new high throughput SKUs for VirtualNetworkGateways.

Az.RedisCache

- Updated 'Set-AzRedisCache' reference documentation to include missing values for '-Size' parameter

Az.Sql

- Add support for setting Active Directory Administrator on Managed Instance

Az.Storage

- Upgrade Storage Client Library to 11.1.0
- List containers with Management plane API, will list with NextPageLink
  - Get-AzRmStorageContainer
- List Storage accounts from subscription, will list with NextPageLink
  - Get-AzStorageAccount

Az.StorageSync

- Fix Issue 9810 in Reset-AzStorageSyncServerCertificate.

Az.Websites

- Set-AzWebApp updating ASP of an app was failing

2.7.0 - September 2019

Az.ApiManagement

- Update '-Format' parameter description in 'Set-AzApiManagementPolicy' reference documentation
- Removed references of deprecated cmdlet 'Update-AzApiManagementDeployment' from reference documentation. Use 'Set-AzApiManagement' instead.

Az.Automation

- Fixed example typo in reference documentation for 'Register-AzAutomationDscNode'

- Added clarification on OS restriction to Register-AzAutomationDSCNode
- Fixed Start-AzAutomationRunbook cmdlet Null reference exception for -Wait option.

Az.Compute

- Add UploadSizeInBytes parameter tp New-AzDiskConfig
- Add Incremental parameter to New-AzSnapshotConfig
- Add a low priority virtual machine feature:
  - MaxPrice, EvictionPolicy and Priority parameters are added to New-AzVMConfig.
  - MaxPrice parameter is added to New-AzVmssConfig, Update-AzVM and Update-AzVmss cmdlets.
- Fix VM reference issue for Get-AzAvailabilitySet cmdlet when it lists all availability sets in the subscription.
- Fix the null exception for Get-AzRemoteDesktopFile.
- Fix VHD Seek method for end-relative position.
- Fix UltraSSD issue for New-AzVM and Update-AzVM.

Az.DataFactory

- Adding 3 new commands for ADF V2 - Add-AzDataFactoryV2TriggerSubscription, Remove-AzDataFactoryV2TriggerSubscription, and Get-AzDataFactoryV2TriggerSubscriptionStatus
- Updated ADF .Net SDK version to 4.1.3

Az.HDInsight

- Call out breaking changes

Az.IotHub

- Add support to invoke failover for an IotHub to the geo-paired disaster recovery region.
- Add support to manage message enrichment for an IotHub. New cmdlets are:
  - Add-AzIotHubMessageEnrichment
  - Get-AzIotHubMessageEnrichment
  - Remove-AzIotHubMessageEnrichment
  - Set-AzIotHubMessageEnrichment

Az.Monitor

- Pointing to the most recent Monitor SDK, i.e. 0.24.1-preview
  - Adds non-braking changes to the Metrics cmdlets, i.e. the Unit enumeration supports several new values. These are read-only cmdlets, so there would be no change in the input of the cmdlets.
  - The api-version of the **ActionGroups** requests is now **2019-06-01**, before it was **2018-03-01**. The scenario tests have been updated to accommodate for this change.
  - The constructors for the classes **EmailReceiver** and **WebhookReceiver** added one new mandatory argument, i.e. a Boolean value called **useCommonAlertSchema**. Currently, the value is fixed to **false** to hide this breaking change from the cmdlets. **NOTE**: this is a temporary change that must be validated by the Alerts team.
  - The order of the arguments for the constructor of the class **Source** (related to the **ScheduledQueryRuleSource** class) changed from the previous SDK. This change required two unit tests to the be fixed: they compiled, but failed to pass the tests.
  - The order of the arguments for the constructor of the class **AlertingAction** (related to the **ScheduledQueryRuleSource** class) changed from the previous SDK. This change required two unit tests to the be fixed: they compiled, but failed to pass the tests.
- Support Dynamic Threshold criteria for metric alert V2
  - New-AzMetricAlertRuleV2Criteria: now creats dynamic threshold criteria also
  - Add-AzMetricAlertRuleV2: now accept dynamic threshold criteria also
- Improvements in Scheduled Query Rule cmdlets (SQR)
- Cmdlets will accept 'Location' paramater in both formats, either the location (e.g. eastus) or the location display

name (e.g. East US)
- Illustrated 'Enabled' parameter in help files properly
- Added examples for 'ActionGroup' optional parameter
- Overall improved help files
- Fix bug in determining scope type for 'Set-AzActionRule'

Az.Network

- Fix incorrect example in 'New-AzApplicationGateway' reference documentation
- Add note in 'Get-AzNetworkWatcherPacketCapture' reference documentation about retrieving all properties for a packet capture
- Fixed example in 'Test-AzNetworkWatcherIPFlow' reference documentation to correctly enumerate NICs
- Improved cloud exception parsing to display additional details if they are present
- Improved cloud exception parsing to handle additional type of SDK exception
- Fixed incorrect mapping of Security Rule models
- Added properties to network interface for private ip feature
  - Added property 'PrivateEndpoint' as type of PSResourceId to PSNetworkInterface
  - Added property 'PrivateLinkConnectionProperties' as type of PSIpConfigurationConnectivityInformation to PSNetworkInterfaceIPConfiguration
  - Added new model class PSIpConfigurationConnectivityInformation
- Added new ApplicationRuleProtocolType 'mssql' for Azure Firewall resource
- MultiLink support in Virtual WAN
  - New cmdlets
    - New-AzVpnSiteLink
    - New-AzVpnSiteLinkConnection
  - Updated cmdlet:
    - New-VpnSite
    - Update-VpnSite
    - New-VpnConnection
    - Update-VpnConnection
- Fixed documents for some PowerShell examples to use Az cmdlets instead of AzureRM cmdlets

Az.RecoveryServices

- Update AzureVMpolicy Object with ProtectedItemsCount Attribute
- Added Tests for VM policy and Original Storage Account Restore

Az.Resources

- Fix bug where New-AzRoleAssignment could not be called without parameter Scope.

Az.ServiceFabric

- Fixed typo in example for 'Update-AzServiceFabricReliability' reference documentation
- Adding new cmdlets to manage appliaction and services:
  - New-AzServiceFabricApplication
  - New-AzServiceFabricApplicationType
  - New-AzServiceFabricApplicationTypeVersion
  - New-AzServiceFabricService
  - Update-AzServiceFabricApplication
  - Get-AzServiceFabricApplication
  - Get-AzServiceFabricApplicationType
  - Get-AzServiceFabricApplicationTypeVersion

    - Get-AzServiceFabricService
    - Remove-AzServiceFabricApplication
    - Remove-AzServiceFabricApplicationType
    - Remove-AzServiceFabricApplicationTypeVersion
    - Remove-AzServiceFabricServic
- Upgraded Service Fabric SDK to version 1.2.0 which uses service fabric resource provider api-version 2019-03-01.

Az.SignalR

- Add Update, Restart, CheckNameAvailability, GetUsage Cmdlets

Az.Sql

- Update example in reference documentation for 'Get-AzSqlElasticPool'
- Added vCore example to creating an elastic pool (New-AzSqlElasticPool).
- Remove the validation of EmailAddresses and the check that EmailAdmins is not false in case EmailAddresses is empty in Set-AzSqlServerAdvancedThreatProtectionPolicy and Set-AzSqlDatabaseAdvancedThreatProtectionPolicy
- Enabled removal of server/database auditing settings when multiple diagnostic settings that enable audit category exist.
- Fix email addresses validation in multiple Sql Vulnerability Assessment cmdlets (Update-AzSqlDatabaseVulnerabilityAssessmentSetting, Update-AzSqlServerVulnerabilityAssessmentSetting, Update-AzSqlInstanceDatabaseVulnerabilityAssessmentSetting and Update-AzSqlInstanceVulnerabilityAssessmentSetting).

Az.Storage

- Updated example in reference documentation for 'Get-AzStorageAccountKey'
- In upload/Downalod Azure File,support perserve the source File SMB properties (File Attributtes, File Creation Time, File Last Write Time) in the destination file
    - Set-AzStorageFileContent
    - Get-AzStorageFileContent
- Fix Upload block blob with properties/metadate fail on container enabled ImmutabilityPolicy.
    - Set-AzStorageBlobContent
- Support manage Azure File shares with Management plane API
    - New-AzRmStorageShare
    - Get-AzRmStorageShare
    - Update-AzRmStorageShare
    - Remove-AzRmStorageShare

Az.Websites

- Fixing issue where webapp Tags were getting deleted when migrating App to new ASPwhere webapp Tags were getting deleted when migrating App to new ASP
- Fixing the Publish-AzureWebapp to work across Linux and windows
- Update example in 'Get-AzWebAppPublishingProfile' reference documentation

2.6.0 - August 2019

General

- Fixed miscellaneous typos across numerous modules

Az.Accounts

- Support user-assigned MSI in Azure Functiosn Authentication (#9479)

Az.Aks

- Fix issue with output for 'Get-AzAks'
  - More information here: https://github.com/Azure/azure-powershell/issues/9847

Az.ApiManagement

- Fix for issue https://github.com/Azure/azure-powershell/issues/9351
  - Update .net nuget version, which does not enforce restrictions on productId, apiId, groupId and userId
- **Get-AzApiManagementProduct** - Added support for querying products using Api. https://github.com/Azure/azure-powershell/issues/9482
- **New-AzApiManagementApiRevision** - Fix for issue where ApiRevisionDescription was not being set when creating new api revision https://github.com/Azure/azure-powershell/issues/9752
- Fixed typo in model 'PsApiManagementOAuth2AuthrozationServer' to 'PsApiManagementOAuth2AuthorizationServer'

Az.Batch

- Fixed typo in help message and documentation to capitalize Windows

Az.Cdn

- Fixed a typo in CDN module conversion helper

Az.Compute

- Add VmssId to New-AzVMConfig cmdlet
- Add TerminateScheduledEvents and TerminateScheduledEventNotBeforeTimeoutInMinutes parameters to New-AzVmssConfig and Update-AzVmss
- Add HyperVGeneration property to VM image object
- Add Host and HostGroup features
  - New cmdlets: New-AzHostGroup New-AzHost Get-AzHostGroup Get-AzHost Remove-AzHostGroup Remove-AzHost
  - HostId parameter is added to New-AzVMConfig and New-AzVM
- Update example in 'Invoke-AzVMRunCommand' documentation to use correct parameter name
- Update '-VolumeType' description in 'Set-AzVMDiskEncryptionExtension' and 'Set-AzVmssDiskEncryptionExtension' reference documentation

Az.DataFactory

- Fix typo to capitalize 'Windows' in 'New-AzDataFactoryEncryptValue' documentation
- Updated ADF .Net SDK version to 4.1.2
- Add parameter 'DataProxyIntegrationRuntimeName', 'DataProxyStagingLinkedServiceName' and 'DataProxyStagingPath' for 'Set-AzureRmDataFactoryV2IntegrationRuntime' cmd to enable set up Self-Hosted Integration Runtime as a proxy for SSIS Integration Runtime
- Updated PSTriggerRun to show the triggered pipelines, message and properties, and PSActivityRun to show the activity type

Az.DataLakeStore

- Fix hanging of Get-DataLakeStoreDeletedItem for any errors or remote exceptions.

Az.EventHub

- Fix for issue #9658 : Typo VirtualNteworkRule parameter in Set-AzEventHubNetworkRuleSet
- Fix for issue #9558 : Set-AzEventHubNamespace is using PATCH instead of PUT
- added EnableKafka parameter to Set-AzEventHubNamespace cmdlet
- Fix for issue #9786 : cannot create a rule with Listen only rights

Az.MarketplaceOrdering

- Fixed documentation typo where 'Azure' was all lowercase letters

Az.Monitor

- Fixed incorrect parameter name in help documentation

Az.Network

- Updated New-AzPrivateLinkServiceIpConfig
  - Deprecated the paramster 'PublicIpAddress' since this is never used in the server side.
  - Added one optional parameter 'Primary' that indicate the current ip configuration is primary one or not.
- Improved handling of request error exception from SDK -Fixes the issue that previously SDK exceptions aren't handled correctly which results in key error details not being displayed
- Adjusted validation logic for Ipv6 IP Prefix to check for correct IPv6 prefix length.
- Updated Get-AzVirtualNetworkSubnetConfig: Added parameter set to get by subnet resource id.
- Updated description of Location parameter for AzNetworkServiceTag

Az.OperationalInsights

- Updated documentation for 'New-AzOperationalInsightsLinuxSyslogDataSource'
  - Added example
  - Updated description for '-Name' parameter
- Added an example for New-AzOperationalInsightsWindowsEventDataSource
- Changed the description of the -Name parameter for New-AzOperationalInsightsWindowsEventDataSource

Az.RecoveryServices

- Update 'Get-AzRecoveryServicesBackupJobDetail.md'

Az.Resources

- Add support for new api version 2019-05-10 for Microsoft.Resource
  - Add support for 'copy.count = 0' for variables, resources and properties
  - Resources with 'condition = false' or 'copy.count = 0' will be deleted in complete mode
- Add an example of assigning policy at subscription level to help doc

Az.ServiceBus

- Fix for issue #9658 : Typo VirtualNetworkRule parameter in Set-AzServiceBusNetworkRuleSet
- Fix for issue #9786 : cannot create a rule with Listen only rights
- Added new command 'Test-AzServiceBusNameAvailability' to check the name availability for queue and topic

Az.ServiceFabric

- Fix add node type cmdlet bugs:
  - NullReferenceException bug when resource group had other vmss not related to the service fabric cluster. Fixes issue: https://github.com/Azure/azure-powershell/issues/8681
  - Fix bug where cmdlet failed if virtualNetwork was in a different resource group that the cluster. fixes issue: https://github.com/Azure/azure-powershell/issues/8407
  - Deprecating Add-AzServiceFabricApplicationCertificate cmdlet

Az.Sql

- Update documentation of old Auditing cmdlets.

Az.Storage

- Update help for Get/Close-AzStorageFileHandle, by add more scenarios to cmdlet examples and update parameter descriptions
- Support StandardBlobTier in upload blob and copy blob
  - Set-AzStorageBlobContent
  - Start-AzStorageBlobCopy
- Support Rehydrate Priority in copy blob
  - Start-AzStorageBlobCopy

Az.Websites

- Add clarification around -AppSettings parameter in Set-AzWebApp and Set-AzWebAppSlot

2.5.0 - July 2019

Az.Accounts

- Update common code to use latest version of ClientRuntime

Az.ApplicationInsights

- Fix example typo in 'Remove-AzApplicationInsightsApiKey' documentation

Az.Automation

- Fix typo in resource string

Az.CognitiveServices

- Added NetworkRuleSet support.

Az.Compute

- Add missing properties (ComputerName, OsName, OsVersion and HyperVGeneration) of VM instance view object.

Az.ContainerRegistry

- Fix typo in Remove-AzContainerRegistryReplication for Replication parameter
  - More information here https://github.com/Azure/azure-powershell/issues/9633

Az.DataFactory

- Updated ADF .Net SDK version to 4.1.0
- Fix typo in documentation for 'Get-AzDataFactoryV2PipelineRun'

Az.EventHub

- Added new cmmdlet added for generating SAS token : New-AzEventHubAuthorizationRuleSASToken
- added verification and error message for authorizationrules rights if only 'Manage' is assigned

Az.KeyVault

- Added support to specify the KeySize for Certificate Policies

Az.LogicApp

- Fix for Get-AzIntegrationAccountMap to list all map types
  - Added new MapType parameter for filtering

Az.ManagedServices

- Added support for api version 2019-06-01 (GA)

Az.Network

- Add support for private endpoint and private link service
  - New cmdlets
    - Set-AzPrivateEndpoint
    - Set-AzPrivateLinkService
    - Approve-AzPrivateEndpointConnection
    - Deny-AzPrivateEndpointConnection
    - Get-AzPrivateEndpointConnection
    - Remove-AzPrivateEndpointConnection
    - Test-AzPrivateLinkServiceVisibility
    - Get-AzAutoApprovedPrivateLinkService
- Updated below commands for feature: PrivateEndpointNetworkPolicies/PrivateLinkServiceNetworkPolicies flag on Subnet in Virtualnetwork
  - Updated New-AzVirtualNetworkSubnetConfig/Set-AzVirtualNetworkSubnetConfig/Add-

AzVirtualNetworkSubnetConfig
    - Added optional parameter -PrivateEndpointNetworkPoliciesFlag that configures whether to apply network policies on private endpoint in this subnet.
    - Added optional parameter -PrivateLinkServiceNetworkPoliciesFlag that configures whether to apply network policies network policies on private link service in this subnet.
- AzPrivateLinkService's cmdlet parameter 'ServiceName' was renamed to 'Name' with an alias 'ServiceName' for backward compatibility
- Enable ICMP protocol for network security rule configurations
    - Updated cmdlets
        - Add-AzNetworkSecurityRuleConfig
        - New-AzNetworkSecurityRuleConfig
        - Set-AzNetworkSecurityRuleConfig
- Add ConnectionProtocolType (Ikev1/Ikev2) as a configurable parameter for New-AzVirtualNetworkGatewayConnection
- Add PrivateIpAddressVersion in LoadBalancerFrontendIpConfiguration
    - Updated cmdlet:
        - New-AzLoadBalancerFrontendIpConfig
        - Add-AzLoadBalancerFrontendIpConfig
        - Set-AzLoadBalancerFrontendIpConfig
- Application Gateway New-AzApplicationGatewayProbeConfig command update for supporting custom port in Probe
    - Updated New-AzApplicationGatewayProbeConfig: Added optional parameter Port which is used for probing backend server. This parameter is applicable for Standard_V2 and WAF_V2 SKU.

Az.OperationalInsights

- Updated default version for saved searches to be 1.
- Fixed custom log null regex handling

Az.RecoveryServices

- Update 'Get-AzRecoveryServicesBackupJob.md'
- Update 'Get-AzRecoveryServicesBackupContainer.md'
- Update 'Get-AzRecoveryServicesVault.md'
- Update 'Wait-AzRecoveryServicesBackupJob.md'
- Update 'Set-AzRecoveryServicesVaultContext.md'
- Update 'Get-AzRecoveryServicesBackupItem.md'
- Update 'Get-AzRecoveryServicesBackupRecoveryPoint.md'
- Update 'Restore-AzRecoveryServicesBackupItem.md'
- Updated service call for Unregistering container for Azure File Share
- Update 'Set-AzRecoveryServicesAsrAlertSetting.md'

Az.Resources

- Remove missing cmdlet referenced in 'New-AzResourceGroupDeployment' documentation
- Updated policy cmdlets to use new api version 2019-01-01

Az.ServiceBus

- Added new cmmdlet added for generating SAS token : New-AzServiceBusAuthorizationRuleSASToken
- added verification and error message for authorizationrules rights if only 'Manage' is assigned

Az.Sql

- Fix missing examples for Set-AzSqlDatabaseSecondary cmdlet
- Fix set Vulnerability Assessment recurring scans without providing any email addresses

- Fix a small typo in a warining message.

Az.Storage

- Update example in reference documentation for 'Get-AzStorageAccount' to use correct parameter name

Az.StorageSync

- Adding Invoke-AzStorageSyncChangeDetection cmdlet.
- Fix Issue 9551 for honoring TierFilesOlderThanDays

Az.Websites

- Fixing a bug where some SiteConfig properties were not returned by Get-AzWebApp and Set-AzWebApp
- Adds a new Location parameter to Get-AzDeletedWebApp and Restore-AzDeletedWebApp
- Fixes a bug with cloning web app slots using New-AzWebApp -IncludeSourceWebAppSlots

2.4.0 - July 2019

Az.Accounts

- Add support for profile cmdlets
- Add support for environments and data planes in generated cmdlets
- Fix bug where incorrect endpoint was being used in some cases for data plane cmdlets in Windows PowerShell

Az.Advisor

- GA release of Az.Advisor
- This module is now included as a part of the roll-up `Az` module

Az.ApiManagement

- Fix for issue https://github.com/Azure/azure-powershell/issues/8671
  - **Get-AzApiManagementSubscription**
    - Added support for querying subscriptions by User and Product
    - Added support for querying using Scope '/', '/apis', '/apis/echo-api'
- Fix for issue https://github.com/Azure/azure-powershell/issues/9307 and https://github.com/Azure/azure-powershell/issues/8432
  - **Import-AzApiManagementApi**
    - Added support for specifying 'ApiVersion' and 'ApiVersionSetId' when importing Apis

Az.Automation

- Fixed Set-AzAutomationConnectionFieldValue cmdlet bug to handle string value.

Az.Compute

- Add HyperVGeneration parameter to New-AzImageConfig

Az.DataFactory

- Updating the output of get activity runs, get pipeline runs, and get trigger runs ADF cmdlets to support Select-Object pipe.

Az.EventGrid

- Fix typo in 'New-AzEventGridSubscription' documentation

Az.IotHub

- Add support to regenerate authorization policy keys.

Az.Network

- Added 'RoutingPreference' to public ip tags
- Improve examples for 'Get-AzNetworkServiceTag' reference documentation

Az.PolicyInsights

- Fix null reference issue in Get-AzPolicyState
  - More information here: https://github.com/Azure/azure-powershell/issues/9446

Az.OperationalInsights

- Fixed CustomLog datasource model returned in Get-AzOperationalInsightsDataSource

Az.RecoveryServices

- Fix for get-policy command for IaaSVMs

Az.Resources

```
- Fix help text for Get-AzPolicyState -Top parameter
- Add client-side paging support for Get-AzPolicyAlias
- Add new parameters for Set-AzPolicyAssignment, -PolicyParameters and -PolicyParametersObject
- Handful of doc and example updates for Policy cmdlets
```

Az.ServiceBus

- Fix for issue #4938 - New-AzureRmServiceBusQueue returns BadRequest when setting MaxSizeInMegabytes

Az.Sql

- Add Instance Failover Group cmdlets from preview release to public release
- Support Azure SQL Server\Database Auditing with new cmdlets.
  - Set-AzSqlServerAudit
  - Get-AzSqlServerAudit
  - Remove-AzSqlServerAudit
  - Set-AzSqlDatabaseAudit
  - Get-AzSqlDatabaseAudit
  - Remove-AzSqlDatabaseAudit
- Remove email constraints from Vulnerability Assessment settings

Az.Storage

- Change 2 parameters '-IndexDocument' and '-ErrorDocument404Path' from required to optional in cmdlet:
  - Enable-AzStorageStaticWebsite
- Update help of Get-AzStorageBlobContent by add an example
- Show more error information when cmdlet failed with StorageException
- Support create or update Storage account with Azure Files AAD DS Authentication
  - New-AzStorageAccount
  - Set-AzStorageAccount
- Support list or close file handles of a file share, file directory or a file
  - Get-AzStorageFileHandle
  - Close-AzStorageFileHandle

Az.StorageSync

- This module is now included as a part of the roll-up `Az` module

2.3.2 - June 2019

Az.Accounts

- Fix bug with incorrect URL being used in some cases for Functions calls
  - More information here: https://github.com/Azure/azure-powershell/issues/8983
- Fix Issue with aliases from AzureRM to Az cmdlets
  - Set-AzureRmVMBootDiagnostics -> Set-AzVMBootDiagnostic
  - Export-AzureRMLogAnalyticThrottledRequests -> Export-AzLogAnalyticThrottledRequest

Az.Compute

- New-AzVm and New-AzVmss simple parameter sets now accept the 'ProximityPlacementGroup' parameter.
- Fix typo in 'New-AzVM' reference documentation

Az.Dns

- Fixed a typo in 'Set-AzDnsZone' help examples.

Az.EventGrid

- Updated to use the 2019-06-01 API version.
- New cmdlets:
  - New-AzureRmEventGridDomain
    - Creates a new Azure Event Grid Domain.
  - Get-AzureRmEventGridDomain
    - Gets the details of an Event Grid Domain, or gets a list of all Event Grid Domains in the current Azure subscription.
  - Remove-AzureRmEventGridDomain
    - Removes an Azure Event Grid Domain.
  - New-AzureRmEventGridDomainKey
    - Regenerates the shared access key for an Azure Event Grid Domain.
  - Get-AzureRmEventGridDomainKey
    - Gets the shared access keys used to publish events to an Event Grid Domain.
  - New-AzureRmEventGridDomainTopic:
    - Creates a new Azure Event Grid Domain Topic.
  - Get-AzureRmEventGridDomainTopic
    - Gets the details of an Event Grid Domain Topic, or gets a list of all Event Grid Domain Topics under specific Event Grid Domain in the current Azure
  - Remove-AzureRmEventGridDomainTopic:
    - Removes an existing Azure Event Grid Domain Topic.
- Updated cmdlets:
  - New-AzureRmEventGridSubscription/Update-AzureRmEventGridSubscription:
    - Add new mandatory parameters to support piping for the new Event Grid Domain and Event Grid Domain Topic to allow creating new event subscription under these resources.
    - Add new mandatory parameters for specifying the new Event Grid Domain name and/or Event Grid Domain Topic name to allow creating new event subscription under these resources.
    - Add new Parameter sets for domains and domain topics to allow reusing existing parameters (e.g., EndPointType, SubjectBeginsWith, etc).
    - Add new optional parameters for specifying:
      - Event subscription expiration date,
      - Advanced filtering parameters.
    - Add new enum for servicebusqueue as destination.
    - Disallow usage of 'All' in -IncludedEventType option and replace it with
  - Get-AzEventGridTopic, Get-AzEventGridDomain, Get-AzEventGridDomainTopic, Get-AzEventGridSubscription:
    - Add new optional parameters (Top, ODataQuery and NextLink) to support results pagination and filtering.
  - Remove-AzureRmEventGridSubscription
    - Add new mandatory parameters to support piping for Event Grid Domain and Event Grid Domain Topic to allow removing existing event subscription under these resources.
    - Add new mandatory parameters for specifying the Event Grid Domain name and/or Event Grid

Domain Topic name to allow removing existing event subscription under these resources.

Az.FrontDoor

- New-AzFrontDoorWafMatchConditionObject
  - Add transforms support and new operator auto-complete value (RegEx)
- New-AzFrontDoorWafManagedRuleObject
  - Add new auto-complete values

Az.Network

- Add support for Virtual Network Gateway Resource
  - New cmdlets
    - Get-AzVirtualNetworkGatewayVpnClientConnectionHealth
- Add AvailablePrivateEndpointType
  - New cmdlets
    - Get-AzAvailablePrivateEndpointType
- Add PrivatePrivateLinkService
  - New cmdlets
    - Get-AzPrivateLinkService
    - New-AzPrivateLinkService
    - Remove-AzPrivateLinkService
    - New-AzPrivateLinkServiceIpConfig
    - Set-AzPrivateEndpointConnection
- Add PrivateEndpoint
  - New cmdlets
    - Get-AzPrivateEndpoint
    - New-AzPrivateEndpoint
    - Remove-AzPrivateEndpoint
    - New-AzPrivateLinkServiceConnection
- Updated below commands for feature: UseLocalAzureIpAddress flag on VpnConnection
  - Updated New-AzVpnConnection: Added optional parameter -UseLocalAzureIpAddress to indicate that local azure ip address should be used as source address while initiating connection.
  - Updated Set-AzVpnConnection: Added optional parameter -UseLocalAzureIpAddress to indicate that local azure ip address should be used as source address while initiating connection.
- Added readonly field PeeredConnections in ExpressRoute peering.
- Added readonly field GlobalReachEnabled in ExpressRoute.
- Added breaking change attribute to call out deprecation of AllowGlobalReach field in ExpressRouteCircuit model
- Fixed Issue 8756 Error using TargetListenerID with AzApplicationGatewayRedirectConfiguration cmdlets
- Fixed bug in New-AzApplicationGatewayPathRuleConfig that prevented the rewrite ruleset from being set.
- Fixed displaying of VirtualNetworkTaps in NetworkInterfaceIpConfiguration
- Fixed Cortex Get cmdlets for list all part
- Fixed VirtualHub reference creation for ExpressRouteGateways, VpnGateway
- Added support for Availability Zones in AzureFirewall and NatGateway
- Added cmdlet Get-AzNetworkServiceTag
- Add support for multiple public IP addresses for Azure Firewall
  - Updated New-AzFirewall cmdlet:
    - Added parameter -PublicIpAddress which accepts one or more Public IP Address objects
    - Added parameter -VirtualNetwork which accepts a Virtual Network object

    - Added methods AddPublicIpAddress and RemovePublicIpAddress on firewall object - these accept a Public IP Address object as input
    - Deprecated parameters -PublicIpName and -VirtualNetworkName
- Updated below commands for feature: Set VpnClient AAD authentication options to Virtual network gateway resource.
    - Updated New-AzVirtualNetworkGateway: Added optional parameters AadTenantUri,AadAudienceId,AadIssuerUri to set VpnClient AAD authentication options on Gateway.
    - Updated Set-AzVirtualNetworkGateway: Added optional parameter AadTenantUri,AadAudienceId,AadIssuerUri to set VpnClient AAD authentication options on Gateway.
    - Updated Set-AzVirtualNetworkGateway: Added optional switch parameter RemoveAadAuthentication to remove VpnClient AAD authentication options from Gateway.

Az.OperationalInsights

- Enable **pergb2018** pricing tier in 'New-AzureRmOperationalInsightsWorkspace' command

Az.Resources

- Support for additional Template Export options
    - Add '-SkipResourceNameParameterization' parameter to Export-AzResourceGroup
    - Add '-SkipAllParameterization' parameter to Export-AzResourceGroup
    - Add '-Resource' parameter to Export-AzResourceGroup for exported resource filtering

Az.ServiceFabric

- Fix add certificate ByExistingKeyVault getting the wrong thumbprint in some cases

Az.Sql

- Fix Advanced Threat Protection storage endpoint suffix
- Fix Advanced Data Security enable overrides Advanced Threat Protection policy
- New Cmdlets for Management.Sql to allow customers to add TDE keys and set TDE protector for managed instances
    - Add-AzSqlInstanceKeyVaultKey
    - Get-AzSqlInstanceKeyVaultKey
    - Remove-AzSqlInstanceKeyVaultKey
    - Get-AzSqlInstanceTransparentDataEncryptionProtector
    - Set-AzSqlInstanceTransparentDataEncryptionProtector

Az.Storage

- Support Kind FileStorage and SkuName Premium_ZRS when create Storage account
    - New-AzStorageAccount
- Clarified description of blob immutability cmdlet
    - Remove-AzRmStorageContainerImmutabilityPolicy

Az.Websites

- Optimizes Get-AzWebAppCertificate to filter by resource group on the server instead of the client
- Adds -UseDisasterRecovery switch parameter to Get-AzWebAppSnapshot

2.2.0 - June 2019

Az.Cdn

- Updated cmdlets to support rulesEngine feature based on API version 2019-04-15.

Az.Compute

- Added `NoWait` parameter that starts the operation and returns immediately, before the operation is completed.
    - Updated cmdlets: Export-AzLogAnalyticRequestRateByInterval Export-AzLogAnalyticThrottledRequest

Remove-AzVM Remove-AzVMAccessExtension Remove-AzVMAEMExtension Remove-AzVMChefExtension Remove-AzVMCustomScriptExtension Remove-AzVMDiagnosticsExtension Remove-AzVMDiskEncryptionExtension Remove-AzVMDscExtension Remove-AzVMSqlServerExtension Restart-AzVM Set-AzVM Set-AzVMAccessExtension Set-AzVMADDomainExtension Set-AzVMAEMExtension Set-AzVMBginfoExtension Set-AzVMChefExtension Set-AzVMCustomScriptExtension Set-AzVMDiagnosticsExtension Set-AzVMDscExtension Set-AzVMExtension Start-AzVM Stop-AzVM Update-AzVM

Az.EventHub

- Fix for #9231 - Get-AzEventHubNamespace does not return tags
- Fix for #9230 - Get-AzEventHubNamespace returns ResourceGroup instead of ResourceGroupName

Az.Network

- Update ResourceId and InputObject for Nat Gateway
  - Add alias for ResourceId and InputObject

Az.PolicyInsights

- Fix Null reference issue in Get-AzPolicyEvent

Az.RecoveryServices

- IaaSVM policy minimum retention in days changed to 7 from 1

Az.ServiceBus

- Fix for issue #9182 - Get-AzServiceBusNamespace returns ResourceGroup instead of ResourceGroupName

Az.ServiceFabric

- Fix typo in error message for 'Update-AzServiceFabricReliability'
- Fix missing character in Service Fabric cmdlines

Az.Sql

- Add DnsZonePartner Parameter for New-AzureSqlInstance cmdlet to support AutoDr for Managed Instance.
- Deprecating Get-AzSqlDatabaseSecureConnectionPolicy cmdlet
- Rename Threat Detection cmdlets to Advanced Threat Protection
- New-AzSqlInstance -StorageSizeInGB and -LicenseType parameters are now optional.

Az.Websites

- fixes the issue where using Set-AzWebApp and Set-AzWebAppSlot with -WebApp property was removing the tags

2.1.0 - May 2019

Az.ApiManagement

- Created new Cmdlets for managing diagnostics at the global and API Scope
  - **Get-AzApiManagementDiagnostic** - Get the diagnostics configured a global or api Scope
  - **New-AzApiManagementDiagnostic** - Create new diagnostics at the global scope or api Scope
  - **New-AzApiManagementHttpMessageDiagnostic** - Create diagnostic setting for which Headers to log and the size of Body Bytes
  - **New-AzApiManagementPipelineDiagnosticSetting** - Create Diagnostic settings for incoming/outgoing HTTP messages to the Gateway.
  - **New-AzApiManagementSamplingSetting** - Create Sampling Setting for the requests/response for a diagnostic
  - **Remove-AzApiManagementDiagnostic** - Remove a diagnostic entity at global or api scope
  - **Set-AzApiManagementDiagnostic** - Update a diagnostic Entity at global or api scope
- Created new Cmdlets for managing Cache in ApiManagement service

  - **Get-AzApiManagementCache** - Get the details of the Cache specified by identifier or all caches
  - **New-AzApiManagementCache** - Create a new 'default' Cache or Cache in a particular azure 'region'
  - **Remove-AzApiManagementCache** - Remove a cache
  - **Update-AzApiManagementCache** - Update a cache
- Created new Cmdlets for managing API Schema
  - **New-AzApiManagementSchema** - Create a new Schema for an API
  - **Get-AzApiManagementSchema** - Get the schemas configured in the API
  - **Remove-AzApiManagementSchema** - Remove the schema configured in the API
  - **Set-AzApiManagementSchema** - Update the schema configured in the API
- Created new Cmdlet for generating a User Token.
  - **New-AzApiManagementUserToken** - Generate a new User Token valid for 8 hours by default.Token for the 'GIT' user can be generated using this cmdlet./
- Created a new cmdlet to retrieving the Network Status
  - **Get-AzApiManagementNetworkStatus** - Get the Network status connectivity of resources on which API Management service depends on. This is useful when deploying ApiManagement service into a Virtual Network and validing whether any of the dependencies are broken.
- Updated cmdlet **New-AzApiManagement** to manage ApiManagement service
  - Added support for the new 'Consumption' SKU
  - Added support to turn the 'EnableClientCertificate' flag on for 'Consumption' SKU
  - The new cmdlet **New-AzApiManagementSslSetting** allows configuring 'TLS/SSL' setting on the 'Backend' and 'Frontend'. This can also be used to configure 'Ciphers' like '3DES' and 'ServerProtocols' like 'Http2' on the 'Frontend' of an ApiManagement service.
  - Added support for configuring the 'DeveloperPortal' hostname on ApiManagement service.
- Updated cmdlets **Get-AzApiManagementSsoToken** to take 'PsApiManagement' object as input
- Updated the cmdlet to display Error Messages inline

```
PS D:\github\azure-powershell> Set-AzApiManagementPolicy -Context -PolicyFilePath
C:\wrongpolicy.xml -ApiId httpbin Set-AzApiManagementPolicy : Error Code: ValidationError Error
Message: One or more fields contain incorrect values: Error Details: [Code=ValidationError,
Message=Error in element 'log-to-eventhub' on line 3, column 10: Logger not found, Target=log-to-
eventhub]
```

- Updated cmdlet **Export-AzApiManagementApi** to export APIs in 'OpenApi 3.0' format
- Updated cmdlet **Import-AzApiManagementApi**
  - To import Api from 'OpenApi 3.0' document specification
  - To override the 'PsApiManagementSchema' property specified in any ('Swagger', 'Wadl', 'Wsdl', 'OpenApi') document.
  - To override the 'ServiceUrl' property specified in any document.
- Updated cmdlet **Get-AzApiManagementPolicy** to return policy in Non-Xml escaped 'format' using 'rawxml'
- Updated cmdlet **Set-AzApiManagementPolicy** to accept policy in Non-Xml escaped 'format' using 'rawxml' and Xml escaped using 'xml'
- Updated cmdlet **New-AzApiManagementApi**
  - To configure API with 'OpenId' authorization server.
  - To create an API in an 'ApiVersionSet'
  - To clone an API using 'SourceApiId' and 'SourceApiRevision'.
  - Ability to configure 'SubscriptionRequired' at the Api scope.
- Updated cmdlet **Set-AzApiManagementApi**
  - To configure API with 'OpenId' authorization server.

  - To updated an API into an 'ApiVersionSet'
  - Ability to configure 'SubscriptionRequired' at the Api scope.
- Updated cmdlet **New-AzApiManagementRevision**
  - To clone (copy tags, products, operations and policies) an existing revision using 'SourceApiRevision'. The new Revision assumes the 'ApiId' of the parent.
  - To provide an 'ApiRevisionDescription'
  - To override the 'ServiceUrl' when cloning an API.
- Updated cmdlet **New-AzApiManagementIdentityProvider**
  - To configure 'AAD' or 'AADB2C' with an 'Authority'
  - To setup 'SignupPolicy', 'SigninPolicy', 'ProfileEditingPolicy' and 'PasswordResetPolicy'
- Updated cmdlet **New-AzApiManagementSubscription**
  - To account for the new SubscriptonModel using 'Scope' and 'UserId'
  - To account for the old subscription model using 'ProductId' and 'UserId'
  - Add support to enable 'AllowTracing' at the subscription level.
- Updated cmdlet **Set-AzApiManagementSubscription**
  - To account for the new SubscriptonModel using 'Scope' and 'UserId'
  - To account for the old subscription model using 'ProductId' and 'UserId'
  - Add support to enable 'AllowTracing' at the subscription level.
- Updated following cmdlets to accept 'ResourceId' as input
  - 'New-AzApiManagementContext'

    ```
    New-AzApiManagementContext -ResourceId
    /subscriptions/subid/resourceGroups/rgName/providers/Microsoft.ApiManagement/service/con
    toso
    ```

  - 'Get-AzApiManagementApiRelease'

    ```
    Get-AzApiManagementApiRelease -ResourceId
    /subscriptions/subid/resourceGroups/rgName/providers/Microsoft.ApiManagement/service/con
    toso/apis/echo-api/releases/releaseId
    ```

  - 'Get-AzApiManagementApiVersionSet'

    ```
    Get-AzApiManagementApiVersionSet -ResourceId
    /subscriptions/subid/resourceGroups/rgName/providers/Microsoft.ApiManagement/service/con
    stoso/apiversionsets/pathversionset
    ```

  - 'Get-AzApiManagementAuthorizationServer'
  - 'Get-AzApiManagementBackend'

    ```
    Get-AzApiManagementBackend -ResourceId
    /subscriptions/subid/resourceGroups/rgName/providers/Microsoft.ApiManagement/service/con
    toso/backends/servicefabric
    ```

  - 'Get-AzApiManagementCertificate'
  - 'Remove-AzApiManagementApiVersionSet'
  - 'Remove-AzApiManagementSubscription'

Az.Automation

- Updated Get-AzAutomationJobOutputRecord to handle JSON and Text record values.

    - Fix for issue https://github.com/Azure/azure-powershell/issues/7977
    - Fix for issue https://github.com/Azure/azure-powershell/issues/8600
- Changed behavior for Start-AzAutomationDscCompilationJob to just start the job instead of waiting for its completion.
    - Fix for issue https://github.com/Azure/azure-powershell/issues/8347
- Fix for Get-AzAutomationDscNode when using -Name returns all node. Now it returns matching node only.

Az.Compute

- Add ProtectFromScaleIn and ProtectFromScaleSetAction parameters to Update-AzVmssVM cmdlet.
- New-AzVM wimple parameter set now uses by default an available location if 'East US' is not supported

Az.DataLakeStore

- Update the ADLS sdk to use httpclient, integrate dataplane testing with azure framework

Az.Monitor

- Fixed incorrect parameter names in help examples

Az.Network

- Add DisableBgpRoutePropagation flag to Effective Route Table output
    - Updated cmdlet:
        - Get-AzEffectiveRouteTable
- Fix double dash in New-AzApplicationGatewayTrustedRootCertificate documentation

Az.Resources

- Add new cmdlet Get-AzureRmDenyAssignment for retrieving deny assignments

Az.Sql

- Rename Advanced Threat Protection cmdlets to Advanced Data Security and enable Vulnerability Assessment by default

2.0.0 - May 2019

Az.Accounts

- Update Authentication Library to fix ADFS issues with username/password auth

Az.CognitiveServices

- Only display Bing disclaimer for Bing Search Services.
- Improve error when create account failed.

Az.Compute

- Proximity placement group feature.
    - The following new cmdlets are added: New-AzProximityPlacementGroup Get-AzProximityPlacementGroup Remove-AzProximityPlacementGroup
    - The new parameter, ProximityPlacementGroupId, is added to the following cmdlets: New-AzAvailabilitySet New-AzVMConfig New-AzVmssConfig
- StorageAccountType parameter is added to New-AzGalleryImageVersion.
- TargetRegion of New-AzGalleryImageVersion can contain StorageAccountType.
- SkipShutdown switch parameter is added to Stop-AzVM and Stop-AzVmss
- Breaking changes
    - Set-AzVMBootDiagnostics is changed to Set-AzVMBootDiagnostic.
    - Export-AzLogAnalyticThrottledRequests is changed to Export-AzLogAnalyticThrottledRequests.

Az.DeploymentManager

- First Generally Available release of Azure Deployment Manager cmdlets

Az.Dns

- Automatic DNS NameServer Delegation
  - Create DNS zone cmdlet accepts parent zone name as additional optional parameter.
  - Adds NS records in the parent zone for newly created child zone.

Az.FrontDoor

- First Generally Available Release of Azure FrontDoor cmdlets
- Rename WAF cmdlets to include 'Waf'
  - `Get-AzFrontDoorFireWallPolicy --> Get-AzFrontDoorWafPolicy`
  - `New-AzFrontDoorCustomRuleObject --> New-AzFrontDoorWafCustomRuleObject`
  - `New-AzFrontDoorFireWallPolicy --> New-AzFrontDoorWafPolicy`
  - `New-AzFrontDoorManagedRuleObject --> New-AzFrontDoorWafManagedRuleObject`
  - `New-AzFrontDoorManagedRuleOverrideObject --> New-AzFrontDoorWafManagedRuleOverrideObject`
  - `New-AzFrontDoorMatchConditionObject --> New-AzFrontDoorWafMatchConditionObject`
  - `New-AzFrontDoorRuleGroupOverrideObject --> New-AzFrontDoorWafRuleGroupOverrideObject`
  - `Remove-AzFrontDoorFireWallPolicy --> Remove-AzFrontDoorWafPolicy`
  - `Update-AzFrontDoorFireWallPolicy --> Update-AzFrontDoorWafPolicy`

Az.HDInsight

- Removed two cmdlets:
  - Grant-AzHDInsightHttpServicesAccess
  - Revoke-AzHDInsightHttpServicesAccess
- Added a new cmdlet Set-AzHDInsightGatewayCredential to replace Grant-AzHDInsightHttpServicesAccess
- Update cmdlet Get-AzHDInsightJobOutput to distinguish reader role and hdinsight operator role:
  - Users with reader role need to specify 'DefaultStorageAccountKey' parameter explicitly, otherwise error occurs.
  - Users with hdinsight operator role will not be affected.

Az.Monitor

- New cmdlets for SQR API (Scheduled Query Rule)
  - New-AzScheduledQueryRuleAlertingAction
  - New-AzScheduledQueryRuleAznsActionGroup
  - New-AzScheduledQueryRuleLogMetricTrigger
  - New-AzScheduledQueryRuleSchedule
  - New-AzScheduledQueryRuleSource
  - New-AzScheduledQueryRuleTriggerCondition
  - New-AzScheduledQueryRule
  - Get-AzScheduledQueryRule
  - Set-AzScheduledQueryRule
  - Update-AzScheduledQueryRule
  - Remove-AzScheduledQueryRule
  - More information about SQR API
  - Updated Az.Monitor.md to include cmdlets for GenV2(non classic) metric-based alert rule

Az.Network

- Add support for Nat Gateway Resource
  - New cmdlets
    - New-AzNatGateway
    - Get-AzNatGateway

    - Set-AzNatGateway
    - Remove-AzNatGateway
  - Updated cmdlets - New-AzureVirtualNetworkSubnetConfigCommand - Add-AzureVirtualNetworkSubnetConfigCommand
- Updated below commands for feature: Custom routes set/remove on Brooklyn Gateway.
  - Updated New-AzVirtualNetworkGateway: Added optional parameter -CustomRoute to set the address prefixes as custom routes to set on Gateway.
  - Updated Set-AzVirtualNetworkGateway: Added optional parameter -CustomRoute to set the address prefixes as custom routes to set on Gateway.

Az.PolicyInsights

- Support for querying policy evaluation details.
  - Add '-Expand' parameter to Get-AzPolicyState. Support '-Expand PolicyEvaluationDetails'.

Az.RecoveryServices

- Support for Cross subscription Azure to Azure site recovery.
- Marking upcoming breaking changes for Azure Site Recovery.
- Fix for Azure Site Recovery recovery plan end action plan.
- Fix for Azure Site Recovery Update network mapping for Azure to Azure.
- Fix for Azure Site Recovery update protection direction for Azure to Azure for managed disk.
- Other minor fixes.

Az.Relay

- Fix typos in customer-facing messages

Az.ServiceBus

- Added new cmdlets for NetworkRuleSet of Namespace

Az.Storage

- Upgrade to Storage Client Library 10.0.1 (the namespace of all objects from this SDK change from 'Microsoft.WindowsAzure.Storage.' *to* '*Microsoft.Azure.Storage.*')
- Upgrade to Microsoft.Azure.Management.Storage 11.0.0, to support new API version 2019-04-01.
- The default Storage account Kind in Create Storage account change from 'Storage' to 'StorageV2'
  - New-AzStorageAccount
- Change the Storage account cmdlet output Sku.Name to be aligned with input SkuName by add '-', like 'StandardLRS' change to 'Standard_LRS'
  - New-AzStorageAccount
  - Get-AzStorageAccount
  - Set-AzStorageAccount

Az.Websites

- 'Kind' property will now be set for PSSite objects returned by Get-AzWebApp
- Get-AzWebApp*Metrics and Get-AzAppServicePlanMetrics marked deprecated

1.8.0 - April 2019

Highlights since the last major release

- General availability of `Az` module
- For more information about the `Az` module, please visit the following: https://aka.ms/azps-announce
- Added Location, ResourceGroup, and ResourceName completers: https://azure.microsoft.com/blog/completers-in-azure-powershell/
- Added wildcard support to Get cmdlets for Az.Compute and Az.Network

- Added interactive and username/password authentication for Windows PowerShell 5.1 only
- Added support for Python 2 runbooks in Az.Automation
- Az.LogicApp: New cmdlets for Integration Account Assemblies and Batch Configuration

Az.Accounts

- Update Uninstall-AzureRm to correctly delete modules in Mac

Az.Batch

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.Cdn

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.CognitiveServices

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.Compute

- Fix issue with AEM installation if resource ids of disks had lowercase resourcegroups in resource id
- Updated cmdlets with plural nouns to singular, and deprecated plural names.
- Fix documentation for wildcards

Az.DataFactory

- Add SsisProperties if NodeCount not null for managed integration runtime.

Az.DataLakeStore

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.EventGrid

- Updated the help text for endpoint to indicate that resources should be created before using the create/update event subscription cmdlets.

Az.EventHub

- Added new cmdlets for NetworkRuleSet of Namespace

Az.HDInsight

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.IotHub

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.KeyVault

- Updated cmdlets with plural nouns to singular, and deprecated plural names.
- Fix documentation for wildcards

Az.MachineLearning

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.Media

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.Monitor

- New cmdlets for GenV2(non classic) metric-based alert rule
  - New-AzMetricAlertRuleV2DimensionSelection
  - New-AzMetricAlertRuleV2Criteria
  - Remove-AzMetricAlertRuleV2
  - Get-AzMetricAlertRuleV2
  - Add-AzMetricAlertRuleV2
- Updated Monitor SDK to version 0.22.0-preview

Az.Network

- Updated cmdlets with plural nouns to singular, and deprecated plural names.
- Fix documentation for wildcards

Az.NotificationHubs

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.OperationalInsights

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.PowerBIEmbedded

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.RecoveryServices

- Updated cmdlets with plural nouns to singular, and deprecated plural names.
- Updated table format for SQL in azure VM
- Added alternate method to fetch location in AzureFileShare
- Updated ScheduleRunDays in SchedulePolicy object according to timezone

Az.RedisCache

- Updated cmdlets with plural nouns to singular, and deprecated plural names.

Az.Resources

- Fix documentation for wildcards

Az.Sql

- Replace dependency on Monitor SDK with common code
- Updated cmdlets with plural nouns to singular, and deprecated plural names.
- Enhanced process of multiple columns classification.
- Include sku properties (sku name, family, capacity) in response from Get-AzSqlServerServiceObjective and format as table by default.
- Ability to Get-AzSqlServerServiceObjective by location without needing a preexisting server in the region.
- Support for time zone parameter in Managed Instance create.
- Fix documentation for wildcards

Az.Websites

- fixes the Set-AzWebApp and Set-AzWebAppSlot to not remove the tags on execution
- Updated cmdlets with plural nouns to singular, and deprecated plural names.
- Updated the WebSites SDK.
- Removed the AdminSiteName property from PSAppServicePlan.

1.7.0 - April 2019

Highlights since the last major release

- General availability of `Az` module
- For more information about the `Az` module, please visit the following: https://aka.ms/azps-announce
- Added Location, ResourceGroup, and ResourceName completers: https://azure.microsoft.com/blog/completers-in-azure-powershell/
- Added wildcard support to Get cmdlets for Az.Compute and Az.Network
- Added interactive and username/password authentication for Windows PowerShell 5.1 only
- Added support for Python 2 runbooks in Az.Automation
- Az.LogicApp: New cmdlets for Integration Account Assemblies and Batch Configuration

Az.Accounts

- Updated Add-AzEnvironment and Set-AzEnvironment to accept parameter AzureAnalysisServicesEndpointResourceId

Az.AnalysisServices

- Using ServiceClient in dataplane cmdlets and removing the original authentication logic
- Making Add-AzureASAccount a wrapper of Connect-AzAccount to avoid a breaking change

Az.Automation

- Fixed New-AzAutomationSoftwareUpdateConfiguration cmdlet bug for Inclusions. Now parameter IncludedKbNumber and IncludedPackageNameMask should work.
- Bug fix for azure automation update management dynamic group

Az.Compute

- Add HyperVGeneration parameter to New-AzDiskConfig and New-AzSnapshotConfig
- Allow VM creation with galley image from other tenants.

Az.ContainerInstance

- Fixed issue in the -Command parameter of New-AzContainerGroup which added a trailing empty argument

Az.DataFactory

- Updated ADF .Net SDK version to 3.0.2
- Updated Set-AzDataFactoryV2 cmdlet with extra parameters for RepoConfiguration related settings.

Az.Resources

- Improve handling of providers for 'Get-AzResource' when providing '-ResourceId' or '-ResourceGroupName', '-Name' and '-ResourceType' parameters
- Improve error handling for 'Test-AzDeployment' and 'Test-AzResourceGroupDeployment'
  - Handle errors thrown outside of deployment validation and include them in output of command instead
  - More information here: https://github.com/Azure/azure-powershell/issues/6856
- Add '-IgnoreDynamicParameters' switch parameter to set of deployment cmdlets to skip prompt in script and job scenarios
  - More information here: https://github.com/Azure/azure-powershell/issues/6856

Az.Sql

- Support Database Data Classification.

Az.Storage

- Report detail error when create Storage context with parameter -UseConnectedAccount, but without login Azure account
  - New-AzStorageContext
- Support Manage Blob Service Properties of a specified Storage account with Management plane API
  - Update-AzStorageBlobServiceProperty
  - Get-AzStorageBlobServiceProperty
  - Enable-AzStorageBlobDeleteRetentionPolicy
  - Disable-AzStorageBlobDeleteRetentionPolicy
- -AsJob support for Blob and file upload and download cmdlets
  - Get-AzStorageBlobContent
  - Set-AzStorageBlobContent
  - Get-AzStorageFileContent
  - Set-AzStorageFileContent

1.6.0 - March 2019

Highlights since the last major release

- General availability of `Az` module
- For more information about the `Az` module, please visit the following: https://aka.ms/azps-announce
- Added Location, ResourceGroup, and ResourceName completers: https://azure.microsoft.com/blog/completers-in-azure-powershell/
- Added wildcard support to Get cmdlets for Az.Compute and Az.Network
- Added interactive and username/password authentication for Windows PowerShell 5.1 only
- Added support for Python 2 runbooks in Az.Automation
- Az.LogicApp: New cmdlets for Integration Account Assemblies and Batch Configuration

Az.Automation

- Azure automation update management change to support the following new features :
  - Dynamic grouping
  - Pre-Post script
  - Reboot Setting

Az.Compute

- Fix issue with path resolution in Get-AzVmBootDiagnosticsData
- Update Compute client library to 25.0.0.

Az.KeyVault

- Added wildcard support to KeyVault cmdlets

Az.Network

- Add Threat Intelligence support for Azure Firewall
- Add Application Gateway Firewall Policy top level resource and Custom Rules

Az.RecoveryServices

- Added SnapshotRetentionInDays in Azure VM policy to support Instant RP
- Added pipe support for unregister container

Az.Resources

- Update wildcard support for Get-AzResource and Get-AzResourceGroup
- Update credentials used when making generic calls to ARM

Az.Sql

- changed Threat Detection's cmdlets param (ExcludeDetectionType) from DetectionType to string[] to make it future proof when new DetectionTypes are added and to support autocomplete as well.

Az.Storage

- Support Get/Set/Remove Management Policy on a Storage account
  - Set-AzStorageAccountManagementPolicy
  - Get-AzStorageAccountManagementPolicy
  - Remove-AzStorageAccountManagementPolicy
  - Add-AzStorageAccountManagementPolicyAction
  - New-AzStorageAccountManagementPolicyFilter
  - New-AzStorageAccountManagementPolicyRule

Az.Websites

- Fix ARM template bug that breaks cloning all slots using 'New-AzWebApp -IncludeSourceWebAppSlots'

1.5.0 - March 2019

Az.Accounts

- Add 'Register-AzModule' command to support AutoRest generated cmdlets
- Update examples for Connect-AzAccount

Az.Automation

- Fixed issue when retreiving certain monthly schedules in several Azure Automation cmdlets
- Fix Get-AzAutomationDscNode returning just top 20 nodes. Now it returns all nodes

Az.Cdn

- Added new Powershell cmdlets for Enable/Disable Custom Domain Https and deprecated the old ones

Az.Compute

- Add wildcard support to Get cmdlets

Az.DataFactory

- Updated ADF .Net SDK version to 3.0.1

Az.LogicApp

- Fix for ListWorkflows only retrieving the first page of results

Az.Network

- Add wildcard support to Network cmdlets

Az.RecoveryServices

- Added Sql server in Azure VM support
- SDK Update
- Removed VMappContainer check in Get-ProtectableItem
- Added Name and ServerName as parameters for Get-ProtectableItem

Az.Resources

- Add `-TemplateObject` parameter to deployment cmdlets
  - More information here: https://github.com/Azure/azure-powershell/issues/2933
- Fix issue when piping the result of `Get-AzResource` to `Set-AzResource`
  - More information here: https://github.com/Azure/azure-powershell/issues/8240
- Fix issue with JSON data type change when running `Set-AzResource`
  - More information here: https://github.com/Azure/azure-powershell/issues/7930

Az.Sql

- Updating AuditingEndpointsCommunicator.
  - Fixing the behavior of an edge case while creating new diagnostic settings.

Az.Storage

- Support Kind BlockBlobStorage when create Storage account - New-AzStorageAccount

1.4.0 - February 2019

Az.AnalysisServices

- Deprecated AddAzureASAccount cmdlet

Az.Automation

- Update help for Import-AzAutomationDscNodeConfiguration
- Added configuration name validation to Import-AzAutomationDscConfiguration cmdlet
- Improved error handling for Import-AzAutomationDscConfiguration cmdlet

Az.CognitiveServices

- Added CustomSubdomainName as a new optional parameter for New-AzCognitiveServicesAccount which is used to specify subdomain for the resource.

Az.Compute

- Fix issue with ID parameter sets
- Update Get-AzVMExtension to list all installed extension if Name parameter is not provided
- Add Tag and ResourceId parameters to Update-AzImage cmdlet
- Get-AzVmssVM without instance ID and with InstanceView can list VMSS VMs with instance view.

Az.DataLakeStore

- Add cmdlets for ADL deleted item enumerate and restore

Az.EventHub

- Added new boolean property SkipEmptyArchives to Skip Empty Archives in CaptureDescription class of Eventhub

Az.KeyVault

- Fix tagging on Set-AzKeyVaultSecret

Az.LogicApp

- Add in Basic sku for Integration Accounts
- Add in XSLT 2.0, XSLT 3.0 and Liquid Map Types
- New cmdlets for Integration Account Assemblies
  - Get-AzIntegrationAccountAssembly
  - New-AzIntegrationAccountAssembly
  - Remove-AzIntegrationAccountAssembly
  - Set-AzIntegrationAccountAssembly
- New cmdlets for Integration Account Batch Configuration
  - Get-AzIntegrationAccountBatchConfiguration
  - New-AzIntegrationAccountBatchConfiguration
  - Remove-AzIntegrationAccountBatchConfiguration
  - Set-AzIntegrationAccountBatchConfiguration
- Update Logic App SDK to version 4.1.0

Az.Monitor

- Update help for Get-AzMetric

Az.Network

- Update help example for Add-AzApplicationGatewayCustomError

Az.OperationalInsights

- Additional support for New and Get ApplicationInsights data source.
  - Added new 'ApplicationInsights' kind to support Get specific and Get all ApplicationInsights data sources for given workspace.
  - Added New-AzOperationalInsightsApplicationInsightsDataSource cmdlet for creating data source by given Application-Insights resource parameters: subscription Id, resourceGroupName and name.

Az.Resources

- Fix for issue https://github.com/Azure/azure-powershell/issues/8166
- Fix for issue https://github.com/Azure/azure-powershell/issues/8235
- Fix for issue https://github.com/Azure/azure-powershell/issues/6219
- Fix bug preventing repeat creation of KeyCredentials

Az.Sql

- Add support for SQL DB Hyperscale tier
- Fixed bug where restore could fail due to setting unnecessary properties in restore request

Az.Websites

- Correct example in Get-AzWebAppSlotMetrics

1.3.0 - February 2019

Az.Accounts

- Update to latest version of ClientRuntime

Az.AnalysisServices

General availability for Az.AnalysisServices module.

Az.Compute

- AEM extension: Add support for UltraSSD and P60,P70 and P80 disks
- Update help description for Set-AzVMBootDiagnostics
- Update help description and example for Update-AzImage

Az.RecoveryServices

General availability for Az.RecoveryServices module.

Az.Resources

- Fix tagging for resource groups
  - More information here: https://github.com/Azure/azure-powershell/issues/8166
- Fix issue where `Get-AzureRmRoleAssignment` doesn't respect -ErrorAction
  - More information here: https://github.com/Azure/azure-powershell/issues/8235

Az.Sql

- Add Get/Set AzSqlDatabaseBackupShortTermRetentionPolicy
- Fix issue where not being logged into Azure account would result in nullref exception when executing SQL cmdlets
- Fixed null ref exception in Get-AzSqlCapability

1.2.1 - January 2019

Az.Accounts

- Release with correct version of Authentication

Az.AnalysisServices

- Release with updated Authentication dependency

Az.RecoveryServices

- Release with updated Authentication dependency

1.2.0 - January 2019

Az.Accounts

- Add interactive and username/password authentication for Windows PowerShell 5.1 only
- Update incorrect online help URLs
- Add warning message in PS Core for Uninstall-AzureRm

Az.Aks

- Update incorrect online help URLs

Az.Automation

- Added support for Python 2 runbooks
- Update incorrect online help URLs

Az.Cdn

- Update incorrect online help URLs

Az.Compute

- Add Invoke-AzVMReimage cmdlet
- Add TempDisk parameter to Set-AzVmss
- Fix the warning message of New-AzVM

Az.ContainerRegistry

- Update incorrect online help URLs

Az.DataFactory

- Updated ADF .Net SDK version to 3.0.0

Az.DataLakeStore

- Fix issue with ADLS endpoint when using MSI
  - More information here: https://github.com/Azure/azure-powershell/issues/7462
- Update incorrect online help URLs

Az.IotHub

- Add Encoding format to Add-IotHubRoutingEndpoint cmdlet.

Az.KeyVault

- Update incorrect online help URLs

Az.Network

- Update incorrect online help URLs

Az.Resources

- Fix incorrect examples in 'New-AzADAppCredential' and 'New-AzADSpCredential' reference documentation
- Fix issue where path for '-TemplateFile' parameter was not being resolved before executing resource group deployment cmdlets
- Az.Resources: Correct documentation for New-AzureRmPolicyDefinition -Mode default value
- Az.Resources: Fix for issue https://github.com/Azure/azure-powershell/issues/7522
- Az.Resources: Fix for issue https://github.com/Azure/azure-powershell/issues/5747
- Fix formatting issue with 'PSResourceGroupDeployment' object
  - More information here: https://github.com/Azure/azure-powershell/issues/2123

Az.ServiceFabric

- Rollback when a certificate is added to VMSS model but an exception is thrown this is to fix bug: https://github.com/Azure/service-fabric-issues/issues/932
- Fix some error messages.
- Fix create cluster with default ARM template for New-AzServiceFabriCluster which was not working with migration to Az.
- Fix add cluster/application certificate to only add to VM Scale Sets that correspond to the cluster by checking cluster id in the extension.

Az.SignalR

- Update incorrect online help URLs

Az.Sql

- Update incorrect online help URLs
- Updated parameter description for LicenseType parameter with possible values
- Fix for updating managed instance identity not working when it is the only updated property
- Support for custom collation on managed instance

Az.Storage

- Update incorrect online help URLs
- Give detail error message when get/set classic Logging/Metric on Premium Storage Account, since Premium Storage Account not supoort classic Logging/Metric.
  - Get/Set-AzStorageServiceLoggingProperty
  - Get/Set-AzStorageServiceMetricsProperty

Az.TrafficManager

- Update incorrect online help URLs

Az.Websites

- Update incorrect online help URLs
- Fixes 'New-AzWebAppSSLBinding' to upload the certificate to the correct resourcegroup+location if the app is hosted on an ASE.
- Fixes 'New-AzWebAppSSLBinding' to not overwrite the tags on binding an SSL certificate to an app

1.1.0 - January 2019

Az.Accounts

- Add 'Local' Scope to Enable-AzureRmAlias

Az.Compute

- Name is now optional in ID parameter set for Restart/Start/Stop/Remove/Set-AzVM and Save-AzVMImage
- Updated the description of ID in help files
- Fix backward compatibility issue with Az.Accounts module

Az.DataLakeStore

- Update the sdk version of dataplane to 1.1.14 for SDK fixes.
  - Fix handling of negative acesstime and modificationtime for getfilestatus and liststatus, Fix async cancellation token

Az.EventGrid

- Updated to use the 2019-01-01 API version.
- Update the following cmdlets to support new scenario in 2019-01-01 API version
  - New-AzureRmEventGridSubscription: Add new optional parameters for specifying:
    - Event Time-To-Live,
    - Maximum number of delivery attempts for the events,
    - Dead letter endpoint.
  - Update-AzureRmEventGridSubscription: Add new optional parameters for specifying:
    - Event Time-To-Live,
    - Maximum number of delivery attempts for the events,
    - Dead letter endpoint.
- Add new enum values (namely, storageQueue and hybridConnection) for EndpointType option in New-AzureRmEventGridSubscription and Update-AzureRmEventGridSubscription cmdlets.
- Show warning message if creating or updating the event subscription is expected to entail manual action from user.

Az.IotHub

- Updated to the latest version of the IotHub SDK

Az.LogicApp

- Get-AzLogicApp lists all without specified Name

Az.Resources

- Fix parameter set issue when providing '-ODataQuery' and '-ResourceId' parameters for 'Get-AzResource'

    - More information here: https://github.com/Azure/azure-powershell/issues/7875
- Fix handling of the -Custom parameter in New/Set-AzPolicyDefinition
- Fix typo in New-AzDeployment documentation
- Made '-MailNickname' parameter mandatory for 'New-AzADUser'
    - More information here: https://github.com/Azure/azure-powershell/issues/8220

Az.SignalR

- Fix backward compatibility issue with Az.Accounts module

Az.Sql

- Converted the Storage management client dependency to the common SDK implementation.

Az.Storage

- Set the StorageAccountName of Storage context as the real Storage Account Name, when it's created with Sas Token, OAuth or Anonymous
    - New-AzStorageContext
- Create Sas Token of Blob Snapshot Object with '-FullUri' parameter, fix the returned Uri to be the sanpshot Uri
    - New-AzStorageBlobSASToken

Az.Websites

- Fixed a date parsing bug in 'Get-AzDeletedWebApp'
- Fix backward compatibility issue with Az.Accounts module

1.0.0 - December 2018

General

- General Availability of Az Module
- Online help for each module
- For more details and a roadmap, see the Az Announcement page
- See the Migration Guide for information on migrating from AzureRM

Az.Accounts

- Changed from Az.Profile
- Fixed table formats for profile and context types

Az.ApiManagement

- Fixes for #7002
- Minor breaking changes, see the Migration Guide for details

Az.Batch

- Added the ability to see what version of the Azure Batch Node Agent is running on each of the VMs in a pool, via the new `NodeAgentInformation` property on `PSComputeNode`.
- The `Caching` default for `PSDataDisk` is now `ReadWrite` instead of `None`.
- Minor breaking changes, see the Migration Guide for details

Az.Billing

- Combines Billing, Consumption, and UsageAggregates cmdlets, see the Migration Guide for details

Az.CognitivServices

- Add completers for SkuName and Typem available on New-AzureRmCognitiveServicesAccount operation
- Removed GetSkusWithAccountParamSetName parameter set from Get-AzCognitiveServicesAccountSkus

Az.ContainerInstance

- Added ManagedIdentity support

Az.DataLakeAnalytics

- Minor breaking changes, see the Migration Guide for details

Az.DataLakeStore

- Minor breaking changes, see the Migration Guide for details

Az.Monitor

- Renamed Az.Insights to Az.Monitor and other minor breaking changes, see the Migration Guide for details

Az.KeyVault

- Removed the deprecated 'PurgeDisabled' property from output types

Az.MachineLearning

- Included cmdlets from Az.MachineLearningCompute module

Az.Media

- Remove deprecated -Tags alias from New-AzMediaService

Az.Network

Added support for the configuring RewriteRuleSets in the Application Gateway - New cmdlets added: - Add-AzureRmApplicationGatewayRewriteRuleSet - Get-AzureRmApplicationGatewayRewriteRuleSet - New-AzureRmApplicationGatewayRewriteRuleSet - Remove-AzureRmApplicationGatewayRewriteRuleSet - Set-AzureRmApplicationGatewayRewriteRuleSet - New-AzureRmApplicationGatewayRewriteRule - New-AzureRmApplicationGatewayRewriteRuleActionSet - New-AzureRmApplicationGatewayRewriteRuleHeaderConfiguration - Cmdlets updated with optional parameter - RewriteRuleSet - New-AzureRmApplicationGateway - New-AzureRmApplicationGatewayRequestRoutingRule - Add-AzureRmApplicationGatewayRequestRoutingRule - New-AzureRmApplicationGatewayPathRuleConfig - Add-AzureRmApplicationGatewayUrlPathMapConfig - New-AzureRmApplicationGatewayUrlPathMapConfig Added KeyVault Support to Application Gateway using Identity. - Cmdlets updated with optonal parameter - KeyVaultSecretId, -KeyVaultSecret - Add-AzApplicationGatewaySslCertificate - New-AzApplicationGatewaySslCertificate - Set-AzApplicationGatewaySslCertificate - New-AzApplicationGateway cmdlet updated with optional parameter -UserAssignedIdentity

- Minor breaking changes, see the Migration Guide for details

Az.OperationalInsights

- Minor breaking changes, see the Migration Guide for details

Az.Profile

- Changed module name to Az.Accounts

Az.RecoveryServices

- Minor breaking changes, see the Migration Guide for details

Az.Resources

- Minor breaking changes, see the Migration Guide for details

Az.ServiceFabric

- Support specfying certificate by common name and thumbprint
- Mnor breaking changes, see the Migration Guide for details

Az.SIgnalR

- General Availability for PowerShell cmdlets for SIgnalR

Az.Sql

- Added new Data_Exfiltration and Unsafe_Action detection types to Threat Detection's cmdlets
- Updated documentation examples for Sql Auditing cmdlets
- Minor breaking changes, see the Migration Guide for details

Az.Storage

- Minor breaking changes, see the Migration Guide for details

Az.Websites

- Minor breaking changes, see the Migration Guide for details

0.7.0 - December 2018

General

- Minor changes for upcoming AzureRM to Az transition

Az.Compute

- Add support for UltraSSD and Gallery Images in the simple param sets for `New-AzVm(ss)` cmdlets.

Az.DataLakeStore

- Fix the trailing slash of the domain of adls account

Az.FrontDoor

- Fixed some broken links
  - In the New-AzureRmFrontDoor and Set-AzureRmFrontDoor articles, fixed the link to the New-AzureRmFrontDoorHealthProbeSettingObject cmdlet article.
  - In the New-AzureRmFrontDoorManagedRuleObject article, fixed the link to the New-AzureRmFrontDoorRuleGroupOverrideObject cmdlet article.

Az.RecoveryServices

- Added client side validations for Azure File Share restore operations.
- Made storageAccountName and storageAccountResourceGroupName optional for afs restore.

Az.Resources

- Fix for https://github.com/Azure/azure-powershell/issues/7679
  - Update Get-AzureRmRoleAssignment to use the subscription scope if it is provided when requesting classic administrators.

Az.Sql

- Minor changes for upcoming AzureRM to Az transition
- Fixed issue with using Get-AzureRmSqlDatabaseVulnerabilityAssessment with DotNet core
- Modified documentation of help messages related to SQL Auditing cmdlets.

Az.Storage

- Add -EnableHierarchicalNamespace to New-AzureRmStorageAccount
- Fix issue that Copy File cmdlet can't reuse source context in destination when not input -DestContext
  - Start-AzureStorageFileCopy
- Support Static Website configuration
  - Enable-AzureStorageStaticWebsite
  - Disable-AzureStorageStaticWebsite

Az.Websites

- Set-AzureRmWebApp and Set-AzureRmWebAppSlot

- New parameter (-AzureStoragePath) added to specify Azure Storage paths to be mounted in Windows and Linux container apps. Use the output of the new cmdlet New-AzureRmWebAppAzureStoragePath as a parameter to set the Azure Storage paths.

0.6.1 - November 2018

Az.ApiManagement

- Update dependencies for type mapping issue

Az.Automation

- Swagger based Azure Automation cmdlets
- Added Update Management cmdlets
- Added Source Control cmdlets
- Added Remove-AzureRmAutomationHybridWorkerGroup cmdlet
- Fixed the DSC Register Node command

Az.Compute

- Fixed identity issue for SystemAssigned identity
- Update dependencies for type mapping issue

Az.ContainerInstance

- Update dependencies for type mapping issue

Az.MarketplaceOrdering

- update the examples description for marketplace cmdlets

Az.Network

- Added cmdlet New-AzureRmApplicationGatewayCustomError, Add-AzureRmApplicationGatewayCustomError, Get-AzureRmApplicationGatewayCustomError, Set-AzureRmApplicationGatewayCustomError, Remove-AzureRmApplicationGatewayCustomError, Add-AzureRmApplicationGatewayHttpListenerCustomError, Get-AzureRmApplicationGatewayHttpListenerCustomError, Set-AzureRmApplicationGatewayHttpListenerCustomError, Remove-AzureRmApplicationGatewayHttpListenerCustomError
- Added ICMP back to supported AzureFirewall Network Protocols
- Update cmdlet Test-AzureRmNetworkWatcherConnectivity, add validation on destination id, address and port.
- Fix issues with memory usage in VirtualNetwork map

Az.RecoveryServices.Backup

- Fix for modifying policy for a protected file share.
- Converted policy timezone to uppercase.

Az.RecoveryServices.SiteRecovery

- Corrected example in New-AzureRmRecoveryServicesAsrProtectableItem
- Update dependencies for type mapping issue

Az.Relay

- Added optional Parameter -KeyValue to New-AzureRmRelayKey cmdlet, which enables user to provide KeyValue.

Az.Resources

- Update help documentation for resource identity related parameters in `New-AzureRmPolicyAssignment` and

`Set-AzureRmPolicyAssignment`

- Add an example for New-AzureRmPolicyDefinition that uses -Metadata
- Fix to allow case preservation in Tag keys in NetStandard: #7678 #7703

Az.ServiceFabric

- Add deprecation messages for upcoming breaking changes

Az.Sql

- Added new cmdlets for CRUD operations on Azure Sql Database Managed Instance and Azure Sql Managed Database
  - Get-AzureRmSqlInstance
  - New-AzureRmSqlInstance
  - Set-AzureRmSqlInstance
  - Remove-AzureRmSqlInstance
  - Get-AzureRmSqlInstanceDatabase
  - New-AzureRmSqlInstanceDatabase
  - Restore-AzureRmSqlInstanceDatabase
  - Remove-AzureRmSqlInstanceDatabase
- Enabled Extended Auditing Policy management on a server or a database.
  - New parameter (PredicateExpression) was added to enable filtering of audit logs.
  - Cmdlets were modified to use SQL clients instead of Legacy clients.
  - Set-AzureRmSqlServerAuditing.
  - Get-AzureRmSqlServerAuditing.
  - Set-AzureRmSqlDatabaseAuditing.
  - Get-AzureRmSqlDatabaseAuditing.
- Fixed issue with using Update-AzureRmSqlDatabaseVulnerabilityAssessmentSettings with storage account name parameter set

0.5.0 - November 2018

General

- Added Resource Completers to many core cmdlets - these alloow you to tab through existing resource names when invoking cmdlets interactively

Az.Profile

- Update common code to use latest version of ClientRuntime
- Rename param TenantId in cmdlet Connect-AzAccount to Tenant and add an alias for TenantId
- Updated TenantId description for Connect-AzAccount
- Fix error message for failed login when providing tenant domain
  - https://github.com/Azure/azure-powershell/issues/6936
- Fix issue with context name clashing for accounts with no subscriptions in tenant
  - https://github.com/Azure/azure-powershell/issues/7453
- Fix issue with DataLake endpoints when using MSI
  - https://github.com/Azure/azure-powershell/issues/7462
- Fix issue where 'Disconnect-AzAccount' would throw if not connected
  - https://github.com/Azure/azure-powershell/issues/7167

Az.CognitiveServices

- Add Get-AzCognitiveServicesAccountSkus operation.

Az.Compute

- Add Add-AzVmssVMDataDisk and Remove-AzVmssVMDataDisk cmdlets
- Get-AzVMImage shows AutomaticOSUpgradeProperties
- Fixed SetAzVMChefExtension -BootstrapOptions and -JsonAttribute option values are not setting in json format.

Az.DataLakeStore

- Update the DataLake package to 1.1.10.
- Add default Concurrency to multithreaded operations.

Az.Insights

- Fixed issue #7267 (Autoscale area)
  - Issues with creating a new autoscale rule not properly setting enumerated parameters (would always set them to the default value).
- Fixed issue #7513 [Insights] Set-AzDiagnosticSetting requires explicit specification of categories during creation of setting
  - Now the cmdlet does not require explicit indication of the categories to enable during creation, i.e. it works as it is documented

Az.Network

- Changed PeeringType to be a mandatory parameter for the following cmdlets:-
  - Get-AzExpressRouteCircuitRouteTable
  - Get-AzExpressRouteCircuitARPTable
  - Get-AzExpressRouteCircuitRouteTableSummary
  - Get-AzExpressRouteCrossConnectionArpTable
  - Get-AzExpressRouteCrossConnectionRouteTable
  - Get-AzExpressRouteCrossConnectionRouteTableSummary

Az.PolicyInsights

- Added policy remediation cmdlets

Az.Resources

- Fix for https://github.com/Azure/azure-powershell/issues/7402
  - Allow listing resources using the '-ResourceId' parameter for 'Get-AzResource'

Az.ServiceBus

- Added MigrationState read-only property to PSServiceBusMigrationConfigurationAttributes which will help to know the Migration state.

Az.ServiceFabric

- Fix add certificate to Linux Vmss.
- Fix 'Add-AzServiceFabricClusterCertificate'
  - Using correct thumbprint from new certificate (Azure/service-fabric-issues#932).
  - Display exception correctly (Azure/service-fabric-issues#1054).
- Fix 'Update-AzServiceFabricDurability' to update cluster configuration before starting Vmss CreateOrUpdate operation.

0.4.0 - October 2018

Az.Profile

- Fix issue with Get-AzSubscription in CloudShell
- Update common code to use latest version of ClientRuntime

Az.Compute

- Added new sizes to the whitelist of VM sizes for which accelerated networking will be turned on when using the

simple param set for 'New-AzVm'

- Added ResourceName argument completer to all cmdlets.

Az.DataLakeStore

- Adding support for Virtual Network Rules
  - Get-AzDataLakeStoreVirtualNetworkRule: Gets or Lists Azure Data Lake Store virtual network rule.
  - Add-AzDataLakeStoreVirtualNetworkRule: Adds a virtual network rule to the specified Data Lake Store account.
  - Set-AzDataLakeStoreVirtualNetworkRule: Modifies the specified virtual network rule in the specified Data Lake Store account.
  - Remove-AzDataLakeStoreVirtualNetworkRule: Deletes an Azure Data Lake Store virtual network rule.

Az.Network

- Update cmdlet Test-AzNetworkWatcherConnectivity, pass the protocol value to backend.
- Added ResourceName argument completer to all cmdlets.

Az.Resources

- Fix isssue where Get-AzRoleDefinition throws an unintelligible exception (when the default profile has no subscription in it and no scope is specified) by adding a meaningful exception in the scenario. Also set the default param set to 'RoleDefinitionNameParameterSet'.

0.3.0 - October 2018

Azure.Storage

- Fix Copy Blob/File won't copy metadata when destination has metadata issue
  - Start-AzureStorageBlobCopy
  - Start-AzureStorageFileCopy
- Support get the Storage resource usage of a specific location, and add warning message for get global Storage resource usage is obsolete.
  - Get-AzStorageUsage

Az.CognitiveServices

- Support Get-AzCognitiveServicesAccountSkus without an existing account.

Az.Compute

- Fix Get-AzVM -ResourceGroupName to return more than 50 results if needed
- Added an example of the 'SimpleParameterSet' to New-AzVmss cmdlet help.
- Fixed a typo in the Azure Disk Encryption progress message

Az.DataFactoryV2

- Updated the ADF .Net SDK version to 2.3.0.

Az.Network

- Added NetworkProfile functionality. new cmdlets added
  - Get-AzNetworkProfile
  - New-AzNetworkProfile
  - Remove-AzNetworkProfile
  - Set-AzNetworkProfile
  - New-AzContainerNicConfig
  - New-AzContainerNicConfigIpConfig
- Added service association link on Subnet Model
- Added cmdlet New-AzVirtualNetworkTap, Get-AzVirtualNetworkTap, Set-AzVirtualNetworkTap, Remove-AzVirtualNetworkTap

- Added cmdlet Set-AzNEtworkInterfaceTapConfig, Get-AzNEtworkInterfaceTapConfig, Remove-AzNEtworkInterfaceTapConfig

Az.RedisCache

- Allow any string as Size parameter going forward. Add P5 in PSArgumentCompleter popup

Az.Resources

- Add missing -Mode parameter to Set-AzPolicyDefinition
- Fix Get-AzProviderOperation commandlet bug for operations with Origin containing User

Az.Sql

- Fixed issue where some backup cmdlets would not recognize the current azure subscription

Az.Websites

- New Cmdlet Get-AzWebAppContainerContinuousDeploymentUrl - Gets the Container Continuous Deployment Webhook URL
- New Cmdlets New-AzWebAppContainerPSSession and Enter-WebAppContainerPSSession - Initiates a PowerShell remote session into a windows container app

0.2.0 - September 2018

Initial Release

Migration Guide for Az 2.0.0

11/4/2019 • 5 minutes to read • Edit Online

This document describes the changes between the 1.0.0 and 2.0.0 versions of Az

Table of Contents

Module breaking changes

Az.Compute

- Removed `Managed` Parameter from `New-AzAvailabilitySet` and `Update-AzAvailabilitySet` cmdlets in favor of using `Sku = Aligned`

  Before

  ```
  Update-AzAvailabilitySet -Managed
  ```

  After

  ```
  Update-AzAvailabilitySet -Sku Aligned
  ```

- For consistency, removed `Image` parameter from 'ByName' and 'ByResourceId' parameter sets in `Update-AzImage`

  Before

  Note that the below code is functional, but the passed-in ImageName is not used, so removing this parameter has no functional impact.

  ```
  Update-AzImage -ResourceGroupName $Rg -ImageName $Name -Image $Image -Tag $tags

  Update-AzImage -ResourceId $Id -Image $Image -Tag $tags
  ```

  After

  ```
  Update-AzImage -ResourceGroupName $Rg -ImageName $Name -Tag $tags

  Update-AzImage -ResourceId $Id -Tag $tags
  ```

- For consistency, removed `Name` parameter from 'ByObject' and 'ByResourceId' parameter sets in `Restart-AzVM`

  Before

  Note that the below code is functional, but the passed-in Name is not used, so removing this parameter has no functional impact.

```
Restart-AzVM -InputObject $VM -Name $Name

Restart-AzVM -ResourceId $Id -Name $Name
```

After

```
Restart-AzVM -InputObject $VM

Restart-AzVM -ResourceId $Id
```

- For consistency, removed `Name` parameter from 'ByObject' and 'ByResourceId' parameter sets in `Start-AzVM`

  Before

  Note that the below code is functional, but the passed-in Name is not used, so removing this parameter has no functional impact.

  ```
  Start-AzVM -InputObject $VM -Name $Name

  Start-AzVM -ResourceId $Id -Name $Name
  ```

  After

  ```
  Start-AzVM -InputObject $VM

  Start-AzVM -ResourceId $Id
  ```

- For consistency, removed `Name` parameter from 'ByObject' and 'ByResourceId' parameter sets in `Stop-AzVM`

  Before

  Note that the below code is functional, but the passed-in Name is not used, so removing this parameter has no functional impact.

  ```
  Stop-AzVM -InputObject $VM -Name $Name

  Stop-AzVM -ResourceId $Id -Name $Name
  ```

  After

  ```
  Stop-AzVM -InputObject $VM

  Stop-AzVM -ResourceId $Id
  ```

- For consistency, removed `Name` parameter from 'ByObject' and 'ByResourceId' parameter sets in `Remove-AzVM`

  Before

  Note that the below code is functional, but the passed-in Name is not used, so removing this parameter has no functional impact.

  ```
  Remove-AzVM -InputObject $VM -Name $Name

  Remove-AzVM -ResourceId $Id -Name $Name
  ```

After

```
Remove-AzVM -InputObject $VM

Remove-AzVM -ResourceId $Id
```

- For consistency, removed `Name` parameter from 'ByObject' and 'ByResourceId' parameter sets in `Set-AzVM`

  Before

  Note that the below code is functional, but the passed-in Name is not used, so removing this parameter has no functional impact.

```
Set-AzVM -InputObject $VM -Name $Name ...

Set-AzVM -ResourceId $Id -Name $Name ...
```

  After

```
Set-AzVM -InputObject $VM ...

Set-AzVM -ResourceId $Id ...
```

- For consistency, removed `Name` parameter from 'ByObject' and 'ByResourceId' parameter sets in `Save-AzVMImage`

  Before

  Note that the below code is functional, but the passed-in Name is not used, so removing this parameter has no functional impact.

```
Save-AzVMImage -InputObject $VM -Name $Name ...

Save-AzVMImage -ResourceId $Id -Name $Name ...
```

  After

```
Save-AzVMImage -InputObject $VM ...

Save-AzVMImage -ResourceId $Id ...
```

- Added ProtectionPolicy property to encapsulate `ProtectFromScaleIn` property in `PSVirtualMachineScaleSetVM`

  Before

```
$vmss = Get-AzVMssVM ...
$vmss.ProtectFromScaleIn = $true

$vmss = Update-AzVMssVM ...
$vmss.ProtectFromScaleIn = $true

$vmss = Remove-AzVMssVMDataDisk ...
$vmss.ProtectFromScaleIn = $true
```

  After

```
$vmss = Get-AzVMssVM ...
$vmss.ProtectionPolicy.ProtectFromScaleIn = $true

$vmss = Update-AzVMssVM ...
$vmss.ProtectionPolicy.ProtectFromScaleIn = $true

$vmss = Remove-AzVMssVMDataDisk ...
$vmss.ProtectionPolicy.ProtectFromScaleIn = $true
```

- Added `EncryptionSettingsCollection` Property to enclose `EncryptionSettings` property in `PSDisk`

Before

```
$disk = New-AzDisk ... | Set-AzDiskDiskEncrytionKey ...
$disk.EncryptionSettings

$disk = New-AzDisk ... | Set-AzDiskKeyEncrytionKey ...
$disk.EncryptionSettings

$update = New-AzDiskUpdateConfig | Set-AzDiskUpdateDiskEncryptionKey ...
$update.EncryptionSettings

$update = New-AzDiskUpdateConfig | Set-AzDiskUpdateKeyEncryptionKey ...
$update.EncryptionSettings
```

After

```
$disk = New-AzDisk ... | Set-AzDiskDiskEncrytionKey ...
$disk.EncryptionSettingsCollection.EncryptionSettings

$disk = New-AzDisk ... | Set-AzDiskKeyEncrytionKey ...
$disk.EncryptionSettingsCollection.EncryptionSettings

$update = New-AzDiskUpdateConfig | Set-AzDiskUpdateDiskEncryptionKey ...
$update.EncryptionSettingsCollection.EncryptionSettings

$update = New-AzDiskUpdateConfig | Set-AzDiskUpdateKeyEncryptionKey ...
$update.EncryptionSettingsCollection.EncryptionSettings
```

- Added `EncryptionSettingsCollection` Property to enclose `EncryptionSettings` property in `PSSnapshot`

Before

```
$snap = New-AzSnapshotConfig ... | Set-AzSnapshotDiskEncryptionKey ...
$snap.EncryptionSettings

$snap = New-AzSnapshotConfig ... | Set-AzSnapshotKeyEncryptionKey ...
$snap.EncryptionSettings

$update = New-AzSnapshotUpdateConfig ... | Set-AzSnapshotUpdateDiskEncryptionKey ...
$update.EncryptionSettings

$update = New-AzSnapshotUpdateConfig ... | Set-AzSnapshotUpdateKeyEncryptionKey ...
$update.EncryptionSettings
```

After

```
$snap = New-AzSnapshotConfig ... | Set-AzSnapshotDiskEncryptionKey ...
$snap.EncryptionSettingsCollection.EncryptionSettings

$snap = New-AzSnapshotConfig ... | Set-AzSnapshotKeyEncryptionKey ...
$snap.EncryptionSettingsCollection.EncryptionSettings

$update = New-AzSnapshotUpdateConfig ... | Set-AzSnapshotUpdateDiskEncryptionKey ...
$update.EncryptionSettingsCollection.EncryptionSettings

$update = New-AzSnapshotUpdateConfig ... | Set-AzSnapshotUpdateKeyEncryptionKey ...
$update.EncryptionSettingsCollection.EncryptionSettings
```

- Removed `VirtualMachineProfile` property from `PSVirtualMachineScaleSet`

  Before

  ```
  $vmss = New-AzVMSSConfig ...
  $vmss.VirtualMachineProfile.AdditionalCapabilities.UltraSSDEnabled = $true
  ```

  After

  ```
  $vmss = New-AzVMSSConfig ...
  $vmss.AdditionalCapabilities.UltraSSDEnabled = $true
  ```

- Cmdlet `Set-AzVMBootDiagnostic` removed alias to `Set-AzVMBootDiagnostics`

  Before

  Using deprecated alias

  ```
  Set-AzVMBootDiagnostics
  ```

  After

  ```
  Set-AzVMBootDIagnostic
  ```

- Cmdlet `Export-AzLogAnalyticThrottledRequest` removed alias to `Export-AzLogAnalyticThrottledRequests`

  Before

  Using deprectaed alias

  ```
  Export-AzLogAnalyticThrottledRequests
  ```

  After

  ```
  Export-AzLogAnalyticThrottledRequest
  ```

Az.HDInsight

- Removed the `Grant-AzHDInsightHttpServicesAccess` and `Revoke-AzHDInsightHttpServicesAccess` cmdlets. These are no longer necessary because HTTP access is always enabled on all HDInsight clusters.
- Added a new `Set-AzHDInsightGatewayCredential` cmdlet. Use this cmdlet to change the gateway HTTP username and password (replaces `Grant-AzHDInsightHttpServicesAccess`).
- Updated the `Get-AzHDInsightJobOutput` cmdlet to support granular role-based access to the storage key.
  - Users with HDInsight Cluster Operator, Contributor, or Owner roles will not be affected.
  - Users with only the Reader role will need to specify `DefaultStorageAccountKey` parameter explicitly.

For more information about these role-based access changes, see aka.ms/hdi-config-update

Before

```
Grant-AzHDInsightHttpServicesAccess -ClusterName $cluster -HttpCredential $credential
```

After

```
Set-AzHDInsightGatewayCredential -ClusterName $cluster -HttpCredential $credential
```

Users with only Reader role for cmdlet Get-AzHDInsightJobOutput

Before

```
Get-AzHDInsightJobOutput  -ClusterName $clusterName -JobId $jobId
```

After

```
Get-AzHDInsightJobOutput  -ClusterName $clusterName -JobId $jobId -DefaultStorageAccountKey $storageAccountKey
```

Az.Storage

- Namespaces for types returned from Blob, Queue, and File cmdlets have changed their namespace from `Microsoft.WindowsAzure.Storage` to `Microsoft.Azure.Storage`. While this is not technically a breaking change according to the breaking change policy, it may require some changes in code that uses the methods from the Storage .Net SDK to interact with the objects returned from these cmdlets.

  Example 1: Add a message to a Queue (change CloudQueueMessage object namespace)

  Before:

  ```
  $queue = Get-AzStorageQueue –Name $queueName –Context $ctx
  $queueMessage = New-Object -TypeName
  "Microsoft.WindowsAzure.Storage.Queue.CloudQueueMessage,$($queue.CloudQueue.GetType().Assembly.FullName)
  " -ArgumentList "This is message 1"
  $queue.CloudQueue.AddMessageAsync($QueueMessage)
  ```

  After:

  ```
  $queue = Get-AzStorageQueue –Name $queueName –Context $ctx
  $queueMessage = New-Object -TypeName
  "Microsoft.Azure.Storage.Queue.CloudQueueMessage,$($queue.CloudQueue.GetType().Assembly.FullName)"  -
  ArgumentList "This is message 1"
  $queue.CloudQueue.AddMessageAsync($QueueMessage)
  ```

  Example 2: Fetch Blob/File Attributes with AccessCondition (change AccessCondition object namespace)

  Before:

  ```
  $accessCondition= New-Object Microsoft.WindowsAzure.Storage.AccessCondition

  $blob = Get-AzureStorageBlob -Container $containerName -Blob $blobName
  $blob.ICloudBlob.FetchAttributes($accessCondition)

  $file = Get-AzureStorageFile -ShareName $shareName -Path $filepath
  $file.FetchAttributes($accessCondition)
  ```

  After:

```
$accessCondition= New-Object Microsoft.Azure.Storage.AccessCondition

$blob = Get-AzureStorageBlob -Container $containerName -Blob $blobName
$blob.ICloudBlob.FetchAttributes($accessCondition)

$file = Get-AzureStorageFile -ShareName $shareName -Path $filepath
$file.FetchAttributes($accessCondition)
```

- While not technically a breaking change, you will notice output differences in the Sku.Name property of Storage Accounts returned from `New/Get/Set-AzStorageAccount` changes are as follows. (After the change, output and input SkuName are aligned.)
  - "StandardLRS" -> "Standard_LRS";
  - "StandardGRS" -> "Standard_GRS";
  - "StandardRAGRS" -> "Standard_RAGRS";
  - "StandardZRS" -> "Standard_ZRS";
  - "PremiumLRS" -> "Premium_LRS";
- The default service behavior when creating a storage account withous specifying a Kind has changed. In previous versions, when a storage account was created with no `Kind` specified, the Storage account Kind of `Storage` was used, in the new version `StorageV2` is the default `Kind` value. If you need to create a V1 Storage account with Kind 'Storage', add parameter '-Kind Storage'

  Example : Create a storage Account (Default Kind change)

  Before:

```
PS c:\> New-AzStorageAccount -ResourceGroupName groupname -Name accountname -SkuName Standard_LRS -
Location "westus"

StorageAccountName ResourceGroupName Location SkuName     Kind      AccessTier CreationTime
ProvisioningState EnableHttpsTrafficOnly
------------------ ----------------- -------- -------     ----      ---------- ------------          ---
-------------- ----------------------
accountname        groupname         westus   StandardLRS Storage   Hot        4/17/2018 10:34:32 AM
Succeeded         False
```

  After:

```
PS c:\> New-AzStorageAccount -ResourceGroupName groupname -Name accountname -SkuName Standard_LRS -
Location "westus"

StorageAccountName ResourceGroupName Location SkuName      Kind      AccessTier CreationTime
ProvisioningState EnableHttpsTrafficOnly
------------------ ----------------- -------- -------      ----      ----------  ------------          -
---------------- ----------------------
accountname        groupname         westus   Standard_LRS StorageV2 Hot        4/17/2018 10:34:32 AM
Succeeded         False
```