Contents

- Azure PowerShell
  - Introducing the new Az module
    - Install Az
    - Migrate to Az
  - Install
    - Install on macOS or Linux
    - Install with MSI
  - Uninstall
  - Get started
  - Cloud Shell
  - Sign in
    - Create a service principal
    - Persistent credentials
  - Queries
  - Format output
  - Manage subscriptions
  - Deploy
    - Deploy Resource Manager templates
    - Export Resource Manager templates
    - Deploy private Resource Manager templates
  - Concepts
    - Experimental modules
    - PowerShell jobs
  - Release notes
    - Breaking changes for Azure PowerShell 6.x
- Tutorials
  - Create Virtual Machines
- Sample Scripts
  - Linux Virtual Machines

Windows Virtual Machines

Web Apps

SQL Databases

Cosmos DB

Overview of Azure PowerShell

11/20/2018 • 2 minutes to read • Edit Online

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

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

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

- Linux Virtual Machines
- Windows Virtual Machines
- Web Apps
- SQL Databases

NOTE

After November 2018, there will be no new features or cmdlets added to the `AzureRM` module. `AzureRM` will continue to be supported and receive bugfixes. New features will be provided in the `Az` module, which will reach 1.0 at the same time. `Az` has a backwards compatibility mode with `AzureRM`, and is designed to be easy to switch to. To learn more about this new module and how to upgrade, see:

- Introducing the Azure PowerShell Az module
- Install Azure PowerShell Az module
- Migrate to the new Azure PowerShell Az module

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

Learn PowerShell basics

If you're unfamiliar with PowerShell, an introduction to PowerShell may be helpful.

- Installing PowerShell
- Scripting with PowerShell

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

Or attend the Microsoft Virtual Academy's Getting Started with PowerShell Jumpstart.

Build your skills with Microsoft Learn

- Automate Azure tasks using scripts with PowerShell
- More interactive learning...

Other Azure PowerShell modules

- Azure Active Directory
- Azure Information Protection

- Azure Service Fabric
- Azure ElasticDB

Introducing the new Azure PowerShell Az module

11/20/2018 • 2 minutes to read • Edit Online

Starting in November 2018, the Azure PowerShell `Az` module is available for full public preview. Az offers shorter commands, improved stability, and supports Windows, macOS, and Linux. Az also offers feature parity and an easy migration path from AzureRM.

Az uses the .NET Standard library, which means it runs on PowerShell 5.x and PowerShell 6.x. Since PowerShell 6.x can run on Linux, macOS, and Windows, that means Az is available for all platforms. Using .NET Standard allows us to unify the code base of Azure PowerShell with minimal impact on users.

Az is a new module, so the version has been reset. The first stable release will be 1.0, but the module has feature parity with AzureRm as of November 2018.

Upgrade to Az

It's recommended that users upgrade to the new `Az` module. To do so:

- Uninstall the Azure PowerShell AzureRM module
- Install the Azure PowerShell Az module
- Enable compatibility mode for AzureRM with `Enable-AzureRMAlias` while you become familiar with the new command set.

Migrate existing scripts to Az

Major updates can be inconvenient. However, the `Az` module has a compatibility mode to help you use existing scripts while you work on updates to the new syntax. Use the `Enable-AzureRmAlias` cmdlet to enable the `AzureRM` compatibility mode. This cmdlet defines `AzureRM` cmdlet names as aliases for the new `Az` cmdlet names.

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`.

For a full description of the migration process, see Migrate from AzureRM to Az.

The future of support for AzureRM

The existing `AzureRM` module will no longer receive new cmdlets or features when `Az` version 1.0 is released in December 2018. However, `AzureRM` is still officially maintained and will get bug fixes. To keep up with the latest Azure services and features, you should switch to the `Az` module.

Install the Azure PowerShell 'Az' module

11/27/2018 • 3 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 preview release of Az, no other install methods are supported.

Requirements

Azure PowerShell works with either PowerShell 5.x on Windows, or PowerShell 6.x on any platform. To check the version of PowerShell running on your machine, run the following command:

```
$PSVersionTable.PSVersion
```

If you have an outdated version or need to install PowerShell, see Installing various versions of PowerShell. Install information for your platform is linked from that page.

Install the Azure PowerShell module

IMPORTANT

You can have both the `AzureRM` and `Az` modules installed at the same time. If you have both modules installed, **don't enable aliases**. Enabling aliases will cause conflicts between `AzureRM` cmdlets and `Az` command aliases, and could cause unexpected behavior. It's recommended that before installing the `Az` module, you uninstall `AzureRM`. You can always uninstall `AzureRM` or enable aliases at any time. For uninstall instructions, see Uninstall the Azure PowerShell module (AzureRM).

To install modules at a global scope, you need elevated privileges to install modules from the PowerShell Gallery. To install Azure PowerShell, run the following command in an elevated session ("Run as Administrator" on Windows, or with superuser privileges on macOS or Linux):

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

If you don't have access to administrator privileges, you can install for the current user by adding the `-Scope` argument.

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

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.

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, you need to 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

You can update your Azure PowerShell installation by running Update-Module. This command does **not** uninstall earlier versions.

```
Update-Module -Name Az
```

If you want to remove older 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-Module -Name Az -List | select Name,Version
```

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

You can load a specific version of the `Az` module by using the `-RequiredVersion` argument with `Install-Module` and `Import-Module`:

```
Install-Module -Name Az -RequiredVersion 0.4.0
Import-Module -Name Az -RequiredVersion 0.4.0
```

If you have more than one version of the module installed, the latest version is loaded by default.

Provide feedback

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

Next Steps

To get started using Azure PowerShell, see Get Started with Azure PowerShell to learn more about the module and its features.

Migrate from AzureRM to Azure PowerShell Az

11/27/2018 • 2 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.

Ensure your existing scripts work with the latest AzureRM release

This is the most important step! Run your existing scripts, and make sure that they work with the *latest* release of AzureRM (**6.13.0**). If your scripts don't work, make sure to read the AzureRM migration guide.

Install the Azure PowerShell Az module

The first step is to install the Az module on your platform. To install Az, you need to 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 these steps:

- Uninstall the AzureRM module. Make sure that you remove *all* installed versions of AzureRM, not just the most recent version.
- Install the Az module

Enable AzureRM alias mode

With AzureRM uninstalled and your scripts working with the latest AzureRM version, now is the time 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 user profile for the selected scope. If no user profile exists, one is created.

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 could also cause issues for other users which 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.

Change module imports and cmdlet names

In general, the module names have been changed so that `AzureRM` and `Azure` become `Az`, and the same for cmdlets. For example, the `AzureRM.Compute` module has been renamed to `Az.Compute`. `New-AzureRmVM` has become `New-AzVM`, and `Get-AzureStorageBlob` is now `Get-AzStorageBlob`.

There are exceptions to this naming change that you should be aware of before doing any renaming:

AZURERM MODULE	AZ MODULE
AzureRM.DataFactories	Az.DataFactory
AzureRM.DataFactoryV2	Az.DataFactory
AzureRM.RecoveryServices.Backup	Az.RecoveryServices
AzureRM.RecoveryServices.SiteRecovery	Az.RecoveryServices

Summary

By following these steps, you can update all of your existing scripts to use the new module. If you have any questions or problems with these steps that made your migration difficult, please comment on this article so that we can improve the instructions.

Install Azure PowerShell on Windows with PowerShellGet

12/10/2018 • 3 minutes to read • Edit Online

This article explains the steps to install the Azure PowerShell modules in a Windows environment using PowerShellGet. PowerShellGet and module management is the preferred way to install Azure PowerShell but if you would rather install with the Web Platform Installer or MSI package, see Other installation methods.

For instructions to install Azure PowerShell on other platforms, see Install and configure Azure PowerShell on macOS and Linux.

The Azure classic deployment model is not supported by this version of Azure PowerShell. For support for classic deployments, follow the instructions in Install the Azure PowerShell Service Management module.

NOTE

After November 2018, there will be no new features or cmdlets added to the `AzureRM` module. `AzureRM` will continue to be supported and receive bugfixes. New features will be provided in the `Az` module, which will reach 1.0 at the same time. `Az` has a backwards compatibility mode with `AzureRM`, and is designed to be easy to switch to. To learn more about this new module and how to upgrade, see:

- Introducing the Azure PowerShell Az module
- Install Azure PowerShell Az module
- Migrate to the new Azure PowerShell Az module

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

Requirements

Starting with Azure PowerShell version 6.0, Azure PowerShell requires PowerShell version 5.0. To check the version of PowerShell running on your machine, run the following command:

```
$PSVersionTable.PSVersion
```

If you have an outdated version, see Upgrading existing Windows PowerShell.

IMPORTANT

The module described in this document, AzureRM, uses .NET Framework. This makes it incompatible with PowerShell 6.0, which uses .NET Core. If you are using PowerShell 6.0, follow the installation instructions for macOS and Linux.

Install the Azure PowerShell module

You need elevated privileges to install modules from the PowerShell Gallery. To install Azure PowerShell, run the following command in an elevated session:

```
Install-Module -Name AzureRM -AllowClobber
```

NOTE

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

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 `AzureRM` 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.

Sign in

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

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

NOTE

If you've disabled module autoloading, you need to manually import the module with `Import-Module AzureRM`. 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

You can update your Azure PowerShell installation by running Update-Module. This command does **not** uninstall earlier versions.

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

If you want to remove older 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-Module -Name AzureRM -ListAvailable | select Name,Version
```

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

You might need more than one version if you work with on-premises Azure Stack resources, run an older version of Windows, or use the Azure classic deployment model. To install an older version, provide the `-RequiredVersion` argument when installing.

```
# Install version 2.3.0 of Azure PowerShell
Install-Module -Name AzureRM -RequiredVersion 2.3.0
```

When loading the Azure PowerShell module the latest version is loaded by default. To load a different version, provide the `-RequiredVersion` argument.

```
# Load version 2.3.0 of Azure PowerShell
Import-Module -Name AzureRM -RequiredVersion 2.3.0
```

Provide feedback

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

Next Steps

To get started using Azure PowerShell, see Get Started with Azure PowerShell to learn more about the module and its features.

Install the Azure PowerShell 'Az' module

11/27/2018 • 3 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 preview release of Az, no other install methods are supported.

Requirements

Azure PowerShell works with either PowerShell 5.x on Windows, or PowerShell 6.x on any platform. To check the version of PowerShell running on your machine, run the following command:

```
$PSVersionTable.PSVersion
```

If you have an outdated version or need to install PowerShell, see Installing various versions of PowerShell. Install information for your platform is linked from that page.

Install the Azure PowerShell module

IMPORTANT

You can have both the `AzureRM` and `Az` modules installed at the same time. If you have both modules installed, **don't enable aliases**. Enabling aliases will cause conflicts between `AzureRM` cmdlets and `Az` command aliases, and could cause unexpected behavior. It's recommended that before installing the `Az` module, you uninstall `AzureRM`. You can always uninstall `AzureRM` or enable aliases at any time. For uninstall instructions, see Uninstall the Azure PowerShell module (AzureRM).

To install modules at a global scope, you need elevated privileges to install modules from the PowerShell Gallery. To install Azure PowerShell, run the following command in an elevated session ("Run as Administrator" on Windows, or with superuser privileges on macOS or Linux):

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

If you don't have access to administrator privileges, you can install for the current user by adding the `-Scope` argument.

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

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.

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, you need to 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

You can update your Azure PowerShell installation by running Update-Module. This command does **not** uninstall earlier versions.

```
Update-Module -Name Az
```

If you want to remove older 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-Module -Name Az -List | select Name,Version
```

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

You can load a specific version of the `Az` module by using the `-RequiredVersion` argument with `Install-Module` and `Import-Module`:

```
Install-Module -Name Az -RequiredVersion 0.4.0
Import-Module -Name Az -RequiredVersion 0.4.0
```

If you have more than one version of the module installed, the latest version is loaded by default.

Provide feedback

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

Next Steps

To get started using Azure PowerShell, see Get Started with Azure PowerShell to learn more about the module and its features.

Install Azure PowerShell on Windows with MSI

12/10/2018 • 2 minutes to read • Edit Online

This article explains how to install Azure PowerShell on Windows using an MSI installer.
Use these installation methods only if they're necessary for your system. The recommended way to install Azure PowerShell on Windows is with PowerShellGet. For instructions on using PowerShellGet to install Azure PowerShell, see Install Azure PowerShell with PowerShellGet.

NOTE

The Web Platform Installer method of installation is no longer available for versions of Azure PowerShell 6.x and higher. If you require use of the Web Platform Installer please consider using the MSI instead, or you can install an earlier version of Azure PowerShell.

To install on Linux or macOS environments, see Install Azure PowerShell on macOS or Linux.

Install or update on Windows using the MSI Package

Azure PowerShell for Windows PowerShell 5.x can be installed using the MSI file available from GitHub. If you have installed previous versions of Azure modules as an MSI, the installer automatically removes them. The MSI package installs modules in `${env:ProgramFiles}\WindowsPowerShell\Modules`. Both the `AzureRM` and `Azure` modules are installed.

NOTE

Only use the `Azure` module if you are working with the Azure classic deployment model.

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

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

NOTE

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

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.

Uninstall the Azure PowerShell module

11/30/2018 • 2 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 encounter a bug, we'd appreciate it if you file a GitHub issue.

Uninstall from PowerShell

If you installed Azure PowerShell using PowerShellGet, you can 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 AzureRM -AllVersions
```

```
Version             Name                                Repository           Description
-------             ----                                ----------           -----------
6.11.0              AzureRM                             PSGallery            Azure Resource Manager Module
6.13.1              AzureRM                             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
  $target.Dependencies | ForEach-Object {
    if ($_.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 $version
      if ($candidate) {
        $AllModules += New-Object -TypeName psobject -Property @{name=$_.name; version=$version}
      }
      else {
        Write-Warning ("Could not find uninstall candidate for {0}:{1} - module may require manual
uninstall" -f $_.name,$version)
      }
    }
  }
  $AllModules += New-Object -TypeName psobject -Property @{name=$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 AzureRM -Version 4.4.1 -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 AzureRM.Profile version 3.4.1
Uninstalling Azure.Storage version 3.4.1
Uninstalling AzureRM.AnalysisServices version 0.4.7
Uninstalling Azure.AnalysisServices version 0.4.7
...
```

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

```
$versions = (get-installedmodule AzureRM -AllVersions | Select-Object Version)
$versions[1..($versions.Length-1)]  | foreach { Uninstall-AllModules -TargetModule AzureRM -Version
($_.Version) -Force }
```

Uninstall 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, and can uninstall from there.

Get started with Azure PowerShell

12/10/2018 • 8 minutes to read • Edit Online

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

Install Azure PowerShell

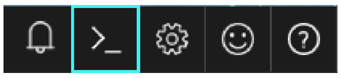
The first step is to make sure you have the latest version of the Azure PowerShell installed. For information about the latest release, see the release notes.

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

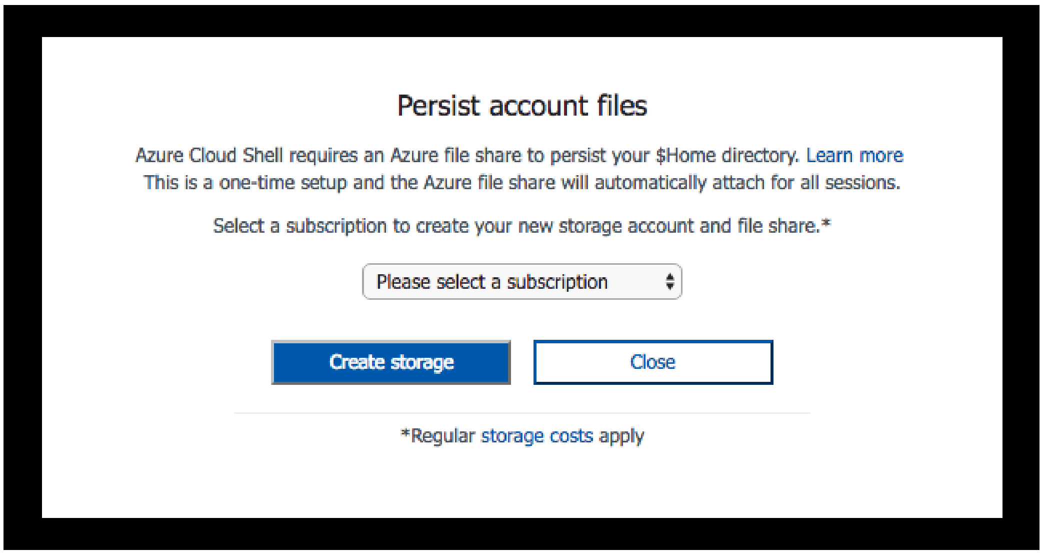
Azure Cloud Shell

The simplest way to get started is to launch Cloud Shell.

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



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



Persist account files

Azure Cloud Shell requires an Azure file share to persist your $Home directory. Learn more
This is a one-time setup and the Azure file share will automatically attach for all sessions.

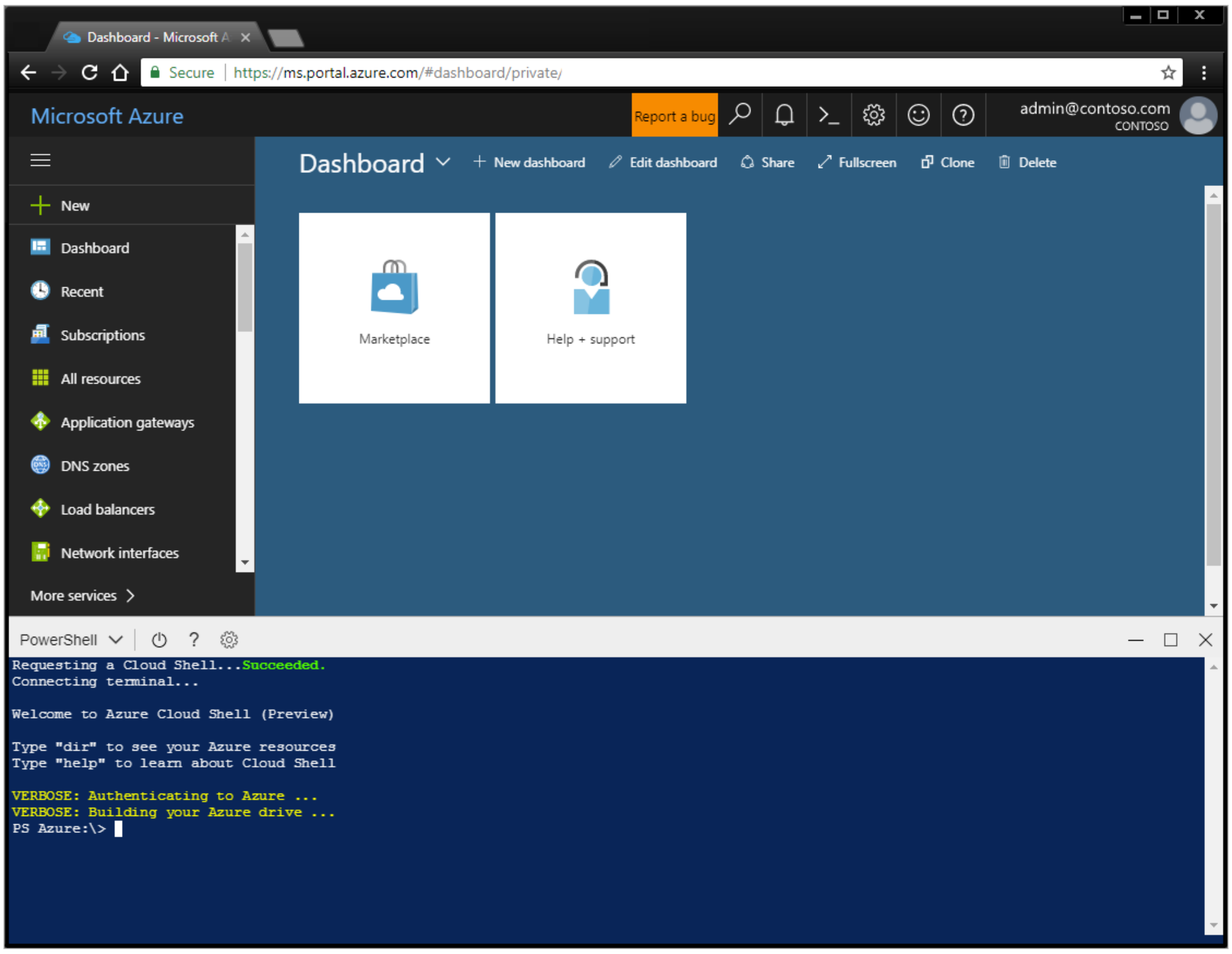
Select a subscription to create your new storage account and file share.*

Please select a subscription

Create storage | Close

*Regular storage costs apply

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



```
Requesting a Cloud Shell...Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell (Preview)

Type "dir" to see your Azure resources
Type "help" to learn about Cloud Shell

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS Azure:\>
```

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

Sign in to Azure

Sign on interactively:

1. Type `Connect-AzureRmAccount`. You'll get a dialog box asking for your Azure credentials. Option '-Environment' can let you authenticate for Azure China or Azure Germany.

   for example, Connect-AzureRmAccount -Environment AzureChinaCloud

2. Type the email address and password associated with your account. Azure authenticates and saves the credential information, and then closes the window.

Once you have signed in to an Azure account, you can use the Azure PowerShell cmdlets to access and manage the resources in your subscription.

Create a Windows virtual machine using simple defaults

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

First, create the credential object.

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

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

Next, create the VM.

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

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

You may wonder what else is created and how is the VM configured. First, let's look at our resource groups.

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

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

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

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

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

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

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

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

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

Create a fully configured Linux Virtual Machine

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

Create a resource group

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

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

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

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

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

Create the required network resources

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

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

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

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

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

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

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

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

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

Create the VM configuration

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

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

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

Create the virtual machine

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

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

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

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

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

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

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

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

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

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

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



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

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

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

Creating other resources in Azure

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

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

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

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

```
$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name mySubnet2 -AddressPrefix 10.0.0.0/16
$vnet = New-AzureRmVirtualNetwork -ResourceGroupName myResourceGroup -Location westeurope `
  -Name MYvNET3 -AddressPrefix 10.0.0.0/16 -Subnet $subnetConfig
```

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

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

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

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

Listing deployed resources

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

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

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

Deleting resources

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

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

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

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

You can also delete many resources at once. For example, the following command deletes the resource group "MyResourceGroup" that we've used for all the samples so far. All resources in the group are also deleted.

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

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

The task can take several minutes to complete, depending on the number and type of resources.

Get samples

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

Next steps

- Sign in with Azure PowerShell
- Manage Azure subscriptions with Azure PowerShell
- Create service principals in Azure using Azure PowerShell
- Read the release notes about migrating from an older release: https://github.com/Azure/azure-powershell/tree/dev/documentation/release-notes.
- Get help from the community:
  - Azure forum on MSDN
  - stackoverflow

Sign in with Azure PowerShell

11/20/2018 • 2 minutes to read • Edit Online

Azure PowerShell supports several authentication methods. The simplest way to get started is to sign in interactively at the command line.

Sign in interactively

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

```
Connect-AzureRmAccount
```

When run, this cmdlet will bring up a dialog box prompting you for your email address and password associated with your Azure account. This authentication lasts for the current PowerShell session.

IMPORTANT

As of Azure PowerShell 6.3.0, your credentials are shared among multiple PowerShell sessions as long as you remain signed in to Windows. For more information, see the article on Persistent Credentials.

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-AzureRmAccount` cmdlet. You'll also need the service principal's application ID, sign-in credentials, and the tenant ID associate with the service principal. To get the service principal's credentials as the appropriate object, use the Get-Credential cmdlet. This cmdlet will display a dialog box to enter the service principal user ID and password into.

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

Sign in using an Azure Managed Service Identity

Managed identities for Azure resources is a feature of Azure Active Directory. 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 virtual machines running in an Azure cloud.

For more information 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 as a Cloud Solution Provider (CSP)

A Cloud Solution Provider (CSP) sign-in requires the use of `-TenantId`. Normally, this parameter can be provided as either a tenant ID or a domain name. However, for CSP sign-in, it must be provided a **tenant ID**.

```
Connect-AzureRmAccount -TenantId 'xxxx-xxxx-xxxx-xxxx'
```

Sign in to another Cloud

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

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

The following command gets a list of available environments:

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

Learn more about managing Azure role-based access

For more information about authentication and subscription management in Azure, see Manage Accounts, Subscriptions, and Administrative Roles.

Azure PowerShell cmdlets for role management:

- Get-AzureRmRoleAssignment
- Get-AzureRmRoleDefinition
- New-AzureRmRoleAssignment
- New-AzureRmRoleDefinition
- Remove-AzureRmRoleAssignment
- Remove-AzureRmRoleDefinition
- Set-AzureRmRoleDefinition

Create an Azure service principal with Azure PowerShell

11/27/2018 • 4 minutes to read • Edit Online

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

NOTE

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

What is a 'service principal'?

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

Verify your own permission level

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

The easiest way to check whether your account has the right permissions is through the portal. See Check required permission in portal.

Create a service principal for your app

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

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

Get information about your application

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

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

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

Create a service principal for your application

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

```
$servicePrincipal = New-AzureRmADServicePrincipal -ApplicationId 00c01aaa-1603-49fc-b6df-b78c4e5138b4
```

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

From here, you can either directly use the $servicePrincipal.Secret property in Connect-AzureRmAccount (see "Sign in using the service principal" below), or you can convert this SecureString to a plain text string for later usage:

```
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($servicePrincipal.Secret)
$password = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
[Runtime.InteropServices.Marshal]::ZeroFreeBSTR($BSTR)
```

Sign in using the service principal

You can now sign in as the new service principal for your app using the *appId* you provided and *password* that was automatically generated. You also need the Tenant ID for the service principal. Your Tenant ID is displayed when you sign into Azure with your personal credentials. To sign in with a service principal, use the following commands:

```
$cred = New-Object System.Management.Automation.PSCredential ("00c01aaa-1603-49fc-b6df-b78c4e5138b4",
$servicePrincipal.Secret)
Connect-AzureRmAccount -Credential $cred -ServicePrincipal -TenantId XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

After a successful sign-in you see output like:

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

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

Managing roles

NOTE

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

Azure PowerShell provides the following cmdlets to manage role assignments:

- Get-AzureRmRoleAssignment
- New-AzureRmRoleAssignment
- Remove-AzureRmRoleAssignment

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

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

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

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

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

To view the current roles assigned:

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

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

Other Azure PowerShell cmdlets for role management:

- Get-AzureRmRoleDefinition
- New-AzureRmRoleDefinition
- Remove-AzureRmRoleDefinition
- Set-AzureRmRoleDefinition

Change the credentials of the security principal

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

Add a new password for the service principal

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

```
Secret    : System.Security.SecureString
StartDate : 11/16/2018 12:38:23 AM
EndDate   : 11/16/2019 12:38:23 AM
KeyId     : 6f801c3e-6fcd-42b9-be8e-320b17ba1d36
Type      : Password
```

Get a list of credentials for the service principal

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

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

Remove the old password from the service principal

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

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

Verify the list of credentials for the service principal

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

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

Get information about the service principal

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

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

Persist user credentials across PowerShell sessions

11/20/2018 • 5 minutes to read • Edit Online

Azure PowerShell offers a feature called **Azure Context Autosave**, which gives the following features:

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

Azure contexts defined

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

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

In previous releases, an Azure Context had to be created each time you opened a new PowerShell session. Beginning with Azure PowerShell v4.4.0, Azure Contexts can automatically be saved whenever opening a new PowerShell session.

Automatically save the context for the next sign-in

In versions 6.3.0 and later, Azure PowerShell retains your context information automatically between sessions. To set PowerShell to forget your context and credentials, use `Disable-AzureRmContextAutoSave`. You'll need to sign in to Azure every time you open a PowerShell session.

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

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

Running Azure PowerShell cmdlets as background jobs

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

- Passing the context as an argument

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

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

- Using the default context with Autosave enabled

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

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

When you need to know the outcome of the background task, use `Get-Job` to check the job status and `Wait-Job` to wait for the Job to complete. Use `Receive-Job` to capture or display the output of the background job. For more information, see about_Jobs.

Creating, selecting, renaming, and removing contexts

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

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

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

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

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

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

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

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

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

Forgets the context that was named 'Contoso2'. You can recreate this context using `Set-AzureRmContext`

Removing credentials

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

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

Using context scopes

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

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

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

How the context autosave setting is remembered

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

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

When set to 'true', the context is automatically saved. If set to 'false', the context isn't saved.

Changes to the AzureRM.Profile module

New cmdlets for managing context

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

Changes to existing profile cmdlets

- Add-AzureRmAccount - Allow scoping of the sign-in to the process or the current user. Allow naming the default context after authentication.
- Import-AzureRmContext - Allow scoping of the sign-in to the process or the current user.
- Set-AzureRmContext - Allow selection of existing named contexts, and scope changes to the process or current user.

Query output of Azure PowerShell cmdlets

11/20/2018 • 2 minutes to read • Edit Online

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

Select simple properties

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

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

```
Get-AzureRmVM
```

The default output is automatically formatted as a table.

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

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

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

```
Name          ResourceGroupName Location
----          ----------------- --------
MyUnbuntu1610 MYWESTEURG        westeurope
MyWin2016VM   MYWESTEURG        westeurope
```

Select complex nested properties

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

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

```
Name           OSType
----           ------
MyUnbuntu1610   Linux
MyWin2016VM   Windows
```

Filter results with the Where-Object cmdlet

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

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

```
ResourceGroupName  Name
-----------------  ----
RGDEMO001          KBDemo001VM
RGDEMO001          KBDemo020
```

With the next example, the results will return the VMs that have the vmSize 'Standard_DS1_V2'.

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

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

Format AzurePowerShell cmdlet output

11/20/2018 • 2 minutes to read • Edit Online

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

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

Format examples

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

```
Get-AzureRmVM
```

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

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

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

```
Name          ResourceGroupName Location
----          ----------------- --------
MyUnbuntu1610 MYWESTEURG        westeurope
MyWin2016VM   MYWESTEURG        westeurope
```

Output can also be formatted into a list. The following example shows this using the `Format-List` cmdlet.

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

```
Name              : MyUnbuntu1610
VmId              : 33422f9b-e339-4704-bad8-dbe094585496
Location          : westeurope
ResourceGroupName : MYWESTEURG

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

Convert to other data types

PowerShell also allows taking command output and converting it into multiple data formats. In the following example, the `Select-Object` cmdlet is used to get attributes of the virtual machines in our subscription and convert the output to CSV format for easy import into a database or spreadsheet.

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

```
"ResourceGroupName","Id","VmId","Name","Location","ProvisioningState"
"MYWESTUERG","/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTUERG/providers/Microsoft.Compute/virtualMachines/MyUnbuntu1610","33422f9b-
e339-4704-bad8-dbe094585496","MyUnbuntu1610","westeurope","Succeeded"
"MYWESTUERG","/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTUERG/providers/Microsoft.Compute/virtualMachines/MyWin2016VM","4650c755-
fc2b-4fc7-a5bc-298d5c00808f","MyWin2016VM","westeurope","Succeeded"
```

Output can also be converted into the JSON format. The following example creates the same list of VMs but changes the output format to JSON.

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

```
[
    {
        "ResourceGroupName":  "MYWESTEURG",
        "Id":  "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTEURG/providers/Microsoft.Compute/virtualMachines/MyUnbuntu1610",
        "VmId":  "33422f9b-e339-4704-bad8-dbe094585496",
        "Name":  "MyUnbuntu1610",
        "Location":  "westeurope",
        "ProvisioningState":  "Succeeded"
    },
    {
        "ResourceGroupName":  "MYWESTEURG",
        "Id":  "/subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXXXXX/resourceGroups/MYWESTEURG/providers/Microsoft.Compute/virtualMachines/MyWin2016VM",
        "VmId":  "4650c755-fc2b-4fc7-a5bc-298d5c00808f",
        "Name":  "MyWin2016VM",
        "Location":  "westeurope",
        "ProvisioningState":  "Succeeded"
    }
]
```

Manage multiple Azure subscriptions

11/20/2018 • 2 minutes to read • Edit Online

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

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

```
Get-AzureRmSubscription
```

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

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

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

2. Set the default.

```
Select-AzureRmSubscription -Subscription "My Demos"
```

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

```
Get-AzureRmContext
```

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

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

Use experimental Azure PowerShell modules

11/20/2018 • 3 minutes to read • Edit Online

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

Experimentation methodology

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

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

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

How to install an experimental module

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

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

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

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

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

Documentation and support

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

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

Experiments and areas of improvement

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

Core improvements

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

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

Experimental improvements

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

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

Feature-specific switches

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

- Settable Defaults - Users should be able to set defaults for common parameters like `-ResourceGroupName` and `-Location`.
- Size Defaults - Resource "sizes" can be confusing to users since many Resource Providers use different names (for example, "Standard_DS1_v2" or "S1"). However, most users care more about cost. So it makes sense to define "universal" sizes based on a pricing schedule. Users can choose a specific size or let Azure PowerShell choose the *best option* based on the resource the budget.
- Output Format - Azure PowerShell currently returns `PSObject`s and there's little console output. Azure PowerShell may need to display some information to the user about the "smart defaults" used.

Running cmdlets in parallel using PowerShell jobs

11/20/2018 • 3 minutes to read • Edit Online

PowerShell supports asynchronous action with PowerShell Jobs. Azure PowerShell is heavily dependent on making, and waiting for, network calls to Azure. You may often find yourself needing to make non-blocking calls. To address this need, Azure PowerShell provides first-class PSJob support.

Context Persistence and PSJobs

Since PSJobs are run as separate processes, your Azure connection must be shared with them. After signing in to your Azure account with `Connect-AzureRmAccount`, pass the context to a job.

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

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

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

Automatic Jobs with `-AsJob`

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

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

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

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

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

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

When the job completes, get the result of the job with `Receive-Job`.

NOTE

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

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

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

Example Scenarios

Create several VMs at once:

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

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

In this example, the `Wait-Job` cmdlet causes the script to pause while jobs run. The script continues executing once all of the jobs have completed. Several jobs run in parallel then the script waits for completion before continuing.

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

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

Release notes

11/20/2018 • 21 minutes to read • Edit Online

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

6.13.0 - November 2018

AzureRM.Profile

- Update common code to use latest version of ClientRuntime

AzureRM.ApiManagement

- Update dependencies for type mapping issue

AzureRM.Automation

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

AzureRM.Compute

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

AzureRM.ContainerInstance

- Update dependencies for type mapping issue

AzureRM.MarketplaceOrdering

- update the examples description for marketplace cmdlets

AzureRM.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

AzureRM.RecoveryServices.Backup

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

AzureRM.RecoveryServices.SiteRecovery

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

AzureRM.Relay

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

AzureRM.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

AzureRM.ServiceFabric

- Add deprecation messages for upcoming breaking changes

AzureRM.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

6.12.0 - November 2018

AzureRM.Profile

- Update common code to use latest version of ClientRuntime
- Rename param TenantId in cmdlet Connect-AzureRmAccount to Tenant and add an alias for TenantId
- Updated TenantId description for Connect-AzureRmAccount
- 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-AzureRmAccount' would throw if not connected
  - https://github.com/Azure/azure-powershell/issues/7167

AzureRM.Automation

- Renamed cmdlet DLL filename to Microsoft.Azure.Commands.Automation.dll

AzureRM.CognitiveServices

- Add Get-AzureRmCognitiveServicesAccountSkus operation.

AzureRM.Compute

- Add Add-AzureRmVmssVMDataDisk and Remove-AzureRmVmssVMDataDisk cmdlets
- Get-AzureRmVMImage shows AutomaticOSUpgradeProperties
- Fixed SetAzureRmVMChefExtension -BootstrapOptions and -JsonAttribute option values are not setting in json format.

AzureRM.DataLakeStore

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

AzureRM.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-AzureRMDiagnosticSetting 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

AzureRM.Network

- Changed PeeringType to be a mandatory parameter for the following cmdlets:-
  - Get-AzureRmExpressRouteCircuitRouteTable
  - Get-AzureRmExpressRouteCircuitARPTable
  - Get-AzureRmExpressRouteCircuitRouteTableSummary
  - Get-AzureRMExpressRouteCrossConnectionArpTable
  - Get-AzureRMExpressRouteCrossConnectionRouteTable
  - Get-AzureRMExpressRouteCrossConnectionRouteTableSummary

AzureRM.PolicyInsights

- Added policy remediation cmdlets

AzureRM.RecoveryServices.Backup

- Added support for azure file shares in recovery services.

AzureRM.Resources

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

AzureRM.ServiceBus

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

AzureRM.ServiceFabric

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

6.11.0 - October 2018

AzureRM.Profile

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

AzureRM.Backup

- Deprecated Azure Backup cmdlets.

AzureRM.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-AzureRmVm'
- Added ResourceName argument completer to all cmdlets.

AzureRM.DataLakeStore

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

AzureRM.Network

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

AzureRM.Resources

- Fix isssue where Get-AzureRMRoleDefinition 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'.

6.10.0 - October 2018

Azure.Storage

- Fix Copy Blob/File won't copy metadata when destination has metadata issue
    - Start-AzureStorageBlobCopy
    - Start-AzureStorageFileCopy

AzureRM.CognitiveServices

- Support Get-AzureRmCognitiveServicesAccountSkus without an existing account.

AzureRM.Compute

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

AzureRM.DataFactoryV2

- Updated the ADF .Net SDK version to 2.3.0.

AzureRM.Network

- Added NetworkProfile functionality. new cmdlets added
    - Get-AzureRMNetworkProfile
    - New-AzureRMNetworkProfile
    - Remove-AzureRMNetworkProfile

  - Set-AzureRMNetworkProfile
  - New-AzureRMContainerNicConfig
  - New-AzureRmContainerNicConfigIpConfig
- Added service association link on Subnet Model
- Added cmdlet New-AzureRmVirtualNetworkTap, Get-AzureRmVirtualNetworkTap, Set-AzureRmVirtualNetworkTap, Remove-AzureRmVirtualNetworkTap
- Added cmdlet Set-AzureRmNEtworkInterfaceTapConfig, Get-AzureRmNEtworkInterfaceTapConfig, Remove-AzureRmNEtworkInterfaceTapConfig

AzureRM.RedisCache

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

AzureRM.Resources

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

AzureRM.Sql

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

AzureRM.Storage

- Support get the Storage resource usage of a specific location, and add warning message for get global Storage resource usage is obsolete.
  - Get-AzureRmStorageUsage

AzureRM.Websites

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

6.9.0 - September 2018

General

- AzureRM.SignalR was added to the AzureRM rollup module

AzureRM.Profile

- Minor changes to the storage common code
- Updated help files to include full parameter types.
- Changed -ServicePrincipal to non-mandatory in the ServicePrincipalCertificateWithSubscriptionId parameter set

Azure.Storage

- Support create Storage Context with OAuth.
  - New-AzureStorageContext

AzureRM.Cdn

- Added Standard_Microsoft in Cdn pricing sku.

AzureRM.Compute

- Move dependencies on Keyvault and Storage to the common dependencies
- Add support for more virutal machine sizes to AEM cmdlets
- Add PublicIPPrefix parameter to New-AzureRmVmssIpConfig
- Add ResourceId parameter to Invoke-AzureRmVMRunCommand cmdelt
- Add Invoke-AzureRmVmssVMRunCommand cmdlet

AzureRM.Dns

- Added support for alias record during dns record creation

AzureRM.Insights

- Fixed issues #6833 and #7102 (Diagnostic Settings area)
  - Issues with the default name, i.e. 'service', during creation and listing/getting of diagnostic settings
  - Issues creating diagnostic settings with categories
- Added deprecation message for metrics time grains parameters
  - Timegrains parameters are still being accepted (this is a non-breaking change,) but they are ignored in the backend since only PT1M is valid

AzureRM.Network

- Changes to LoadBalancer cmdlets
  - LoadBalancerInboundNatPoolConfig: added parameters IdleTimeoutInMinutes, EnableFloatingIp and EnableTcpReset
  - LoadBalancerInboundNatRuleConfig: added parameter EnableTcpReset
  - LoadBalancerRuleConfig: added parameter EnableTcpReset
  - LoadBalancerProbeConfig: added support for value "Https" for parameter Protocol
- Added new commands for new LoadBalancer's subresource OutboundRule
  - Add-AzureRmLoadBalancerOutboundRuleConfig
  - Get-AzureRmLoadBalancerOutboundRuleConfig
  - New-AzureRmLoadBalancerOutboundRuleConfig
  - Set-AzureRmLoadBalancerOutboundRuleConfig
  - Remove-AzureRmLoadBalancerOutboundRuleConfig
- Added new HostedWorkloads property for PSNetworkInterface
- Added new cmdlets for feature: Azure Firewall via ARM
  - Added Get-AzureRmFirewall
  - Added Set-AzureRmFirewall
  - Added New-AzureRmFirewall
  - Added Remove-AzureRmFirewall
  - Added New-AzureRmFirewallApplicationRuleCollection
  - Added New-AzureRmFirewallApplicationRule
  - Added New-AzureRmFirewallNatRuleCollection
  - Added New-AzureRmFirewallNatRule
  - Added New-AzureRmFirewallNetworkRuleCollection
  - Added New-AzureRmFirewallNetworkRule
- Added support for Trusted Root certificate and Autoscale configuration in Application Gateway
  - New Cmdlets added:
    - Add-AzureRmApplicationGatewayTrustedRootCertificate
    - Get-AzureRmApplicationGatewayTrustedRootCertificate
    - New-AzureRmApplicationGatewayTrustedRootCertificate
    - Remove-AzureRmApplicationGatewayTrustedRootCertificate
    - Set-AzureRmApplicationGatewayTrustedRootCertificate
    - Get-AzureRmApplicationGatewayAutoscaleConfiguration
    - New-AzureRmApplicationGatewayAutoscaleConfiguration
    - Remove-AzureRmApplicationGatewayAutoscaleConfiguration
    - Set-AzureRmApplicationGatewayAutoscaleConfiguration
  - Cmdlets updated with optonal parameter -TrustedRootCertificate
    - New-AzureRmApplicationGateway

        - New-AzureRmApplicationGateway
        - Set-AzureRmApplicationGateway
        - New-AzureRmApplicationGatewayBackendHttpSetting
        - Set-AzureRmApplicationGatewayBackendHttpSetting
    - Cmdlets updated with optonal parameter -AutoscaleConfiguration
        - New-AzureRmApplicationGateway
        - Set-AzureRmApplicationGateway
- Add cmdlet for Interface Endpoint Get-AzureInterfaceEndpoint
- Added support for multiple address prefixes in a subnet. Updated cmdlets:
    - New-AzureRmVirtualNetworkSubnetConfig
    - Set-AzureRmVirtualNetworkSubnetConfig
    - Add-AzureRmVirtualNetworkSubnetConfig
    - Get-AzureRmVirtualNetworkSubnetConfig
    - Add-AzureRmApplicationGatewayAuthenticationCertificate
    - Add-AzureRmApplicationGatewayFrontendIPConfig
    - New-AzureRmApplicationGatewayFrontendIPConfig
    - Set-AzureRmApplicationGatewayFrontendIPConfig
    - Add-AzureRmApplicationGatewayIPConfiguration
    - New-AzureRmApplicationGatewayIPConfiguration
    - Set-AzureRmApplicationGatewayIPConfiguration
    - Add-AzureRmNetworkInterfaceIpConfig
    - New-AzureRmNetworkInterfaceIpConfig - Set-AzureRmNetworkInterfaceIpConfig
    - New-AzureRmVirtualNetworkGatewayIpConfig
    - Add-AzureRmVirtualNetworkGatewayIpConfig
    - Set-AzureRmLoadBalancerFrontendIpConfig
    - Add-AzureRmLoadBalancerFrontendIpConfig
    - New-AzureRmLoadBalancerFrontendIpConfig
    - New-AzureRmNetworkInterface
- Adding cmdlets for subnet delegation.
    - New-AzureRmDelegation: Creates a new delegation, which can be added to a subnet
    - Remove-AzureRmDelegation: Takes in a subnet and removes the provided delegation name from that subnet
    - Add-AzureRmDelegation: Takes in a subnet and adds the provided service name as a delegation to that subnet
    - Get-AzureRmDelegation
    - Get-AzureRmAvailableServiceDelegations

AzureRM.RecoveryServices.SiteRecovery

- Support for managed Managed disk

AzureRM.RedisCache

- Updated Insights dependency.

AzureRM.Resources

- Update New-AzureRmResourceGroupDeployment with new parameter RollbackAction
    - Add support for OnErrorDeployment with the new parameter.
- Support managed identity on policy assignments.
- Parameters with default values are no longer requred when assigning a policy with 'New-AzureRmPolicyAssignment'
- Add new cmdlet Get-AzureRmPolicyAlias for retrieving policy aliases

AzureRM.ServiceBus

- Fixed issue #7161

AzureRM.SignalR

- Update SKU names to Free_F1 and Standard_S1
- Add version field to the PSSignalRResource object and connection string to the PSSignalRKeys object.

AzureRM.Storage

- Support Immutability Policy in AzureRm.Storage
  - Remove-AzureRmStorageAccountNetworkRule
  - Get-AzureRmStorageContainer
  - Update-AzureRmStorageContainer
  - New-AzureRmStorageContainer
  - Remove-AzureRmStorageContainer
  - Add-AzureRmStorageContainerLegalHold
  - Remove-AzureRmStorageContainerLegalHold
  - Set-AzureRmStorageContainerImmutabilityPolicy
  - Get-AzureRmStorageContainerImmutabilityPolicy
  - Remove-AzureRmStorageContainerImmutabilityPolicy
  - Lock-AzureRmStorageContainerImmutabilityPolicy

AzureRM.Websites

- Added two new cmdlets: Get-AzureRmDeletedWebApp and Restore-AzureRmDeletedWebApp
- New-AzureRmAppServicePlan -HyperV switch is added for create app service plan with windows container
- New-AzureRmWebApp/ New-AzureRmWebAppSlot/ Set-AzureRmWebApp/ Set-AzureRmWebAppSlot - New parameters (–ContainerRegistryUser string -ContainerRegistryPassword secureString - EnableContainerContinuousDeployment) added for creating and managing windows container app

6.8.1 - August 2018

General

- Fixed issue with default resource groups not being set.
- Updated common runtime assemblies

AzureRM.ApiManagement

- Fixed issue with default resource groups not being set.
- Fixed issue https://github.com/Azure/azure-powershell/issues/6603
  - Import-AzureRmApiManagementApi and *-AzureRmApiManagementCertificate cmdlets now handle relative Paths
- Fixed issue https://github.com/Azure/azure-powershell/issues/6879
  - The CertificateInformation is a settable property allowing for Set-AzureRmApiManagement cmdlet to work property. Fixed by upgrading to 4.0.4-preview nuget
- Fixed issue https://github.com/Azure/azure-powershell/issues/6853
  - Fixed the Odata filter for Search by Name on Product
- Fixed issue https://github.com/Azure/azure-powershell/issues/6814
  - Fixed the Odata filter for Search by Name on Api
- Added support for AzureMonitor logger

AzureRM.Compute

- Fixed the issue that target is missing in error output.
- Fixed issue with storage account type for VM with managed disk
- Fixed issue with default resource groups not being set.

- Fix AEM Extension cmdlets for other environments, for example Azure China

AzureRM.Network

- Changed default cmdlet output presentation to table view

AzureRM.PowerBIEmbedded

- Fix failure in Update-AzureRmPowerBIEmbeddedCapacity when trying to scale paused capacity

AzureRM.Resources

- Fixed issue with creating managed applications from the MarketPlace.

AzureRM.ServiceBus

- Fixed issues
  - https://github.com/Azure/azure-powershell/issues/5058
  - https://github.com/Azure/azure-powershell/issues/5055
  - https://github.com/Azure/azure-powershell/issues/6891

AzureRM.TrafficManager

- Added Support for the MultiValue routing method
  - New parameter 'MaxReturn' for MultiValue routing
- Added Support for the Subnet routing method
  - Support for IP address ranges (subnets) in endpoints
- Added Support for Custom Headers in profiles
- Added Support for Expected status code ranges in profiles
- Added Support for Custom Headers in endpoints

6.8.0 - August 2018

General

- Fixed issue with default resource groups not being set.

AzureRM.Profile

- Added expiration property to tokens returned during Connect-AzureRmAccount

AzureRM.Compute

- Fixed the issue that target is missing in error output.
- Fixed issue with storage account type for VM with managed disk
- Fix AEM Extension cmdlets for other environments, for example Azure China

AzureRM.IotHub

- Fix examples for New-AzureRmIotHubExportDevices and New-AzureRmIotHubImportDevices

AzureRM.Network

- Changed default models representation to table-view

AzureRM.PowerBIEmbedded

- Fix failure in Update-AzureRmPowerBIEmbeddedCapacity when trying to scale paused capacity

AzureRM.Resources

- Fixed issue with creating managed application from the MarketPlace.

AzureRM.ServiceBus

- Fix for issues
  - https://github.com/Azure/azure-powershell/issues/5058
  - https://github.com/Azure/azure-powershell/issues/5055
  - https://github.com/Azure/azure-powershell/issues/6891

AzureRM.TrafficManager

- Support for the MultiValue routing method
  - New parameter 'MaxReturn' for MultiValue routing
- Support for the Subnet routing method
  - Support for IP address ranges (subnets) in endpoints
- Support for Custom Headers in profiles
- Support for Expected status code ranges in profiles
- Support for Custom Headers in endpoints

AzureRM.Websites

- Fixed issue with default resource group being set incorrectly.

6.7.0 - August 2018

AzureRM.Profile

- Updated to the latest version of the Azure ClientRuntime.
- Add user id to default context name to avoid context clashing
  - https://github.com/Azure/azure-powershell/issues/6489
- Fix issues with Clear-AzureRmContext that caused issues with selecting a context #6398
- Enable tenant domain to be passed to '-TenantId' parameter for 'Connect-AzureRmAccount'
  - https://github.com/Azure/azure-powershell/issues/3974
  - https://github.com/Azure/azure-powershell/issues/6709

Azure.Storage

- Remove the 5TB limitation for Azure File Share quota
- Set-AzureStorageShareQuota

AzureRM.AnalysisServices

- Updated to the latest version of the Azure ClientRuntime.

Azure.AnalysisServices

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.ApiManagement

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.ApplicationInsights

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Automation

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Backup

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Batch

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Billing

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Cdn

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.CognitiveServices

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Compute

- Updated to the latest version of the Azure ClientRuntime.
- Add EvictionPolicy parameter to New-AzureRmVmssConfig
- Use default location in the DiskFileParameterSet of New-AzureRmVm if no Location is specified.
- Fix parameter description in Save-AzureRmVMImage
- Fix Get-AzureRmVMDiskEncryptionStatus cmdlet for certain singlepass related scenarios

AzureRM.Consumption

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.ContainerInstance

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.ContainerRegistry

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.DataFactories

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.DataFactoryV2

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.DataLakeAnalytics

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.DataLakeStore

- Fix debugging when DebugPreference is set from powershell command line
- Update example for Set-AzureRmDataLakeStoreItemAcl
- Updated to the latest version of the Azure ClientRuntime.
- Update example for Set-AzureRmDataLakeStoreItemAclEntry

AzureRM.DevTestLabs

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Dns

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.EventGrid

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.EventHub

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.HDInsight

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Insights

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.IotHub

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.KeyVault

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.LogicApp

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.MachineLearning

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.MachineLearningCompute

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.MarketplaceOrdering

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Media

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Network

- Added example for Set-AzureRmLocalNetworkGateway
- Added examples and descriptions for Add-AzureRmVirtualNetworkGatewayIpConfig, Get-AzureRmVirtualNetworkGatewayConnectionSharedKey and New-AzureRmVirtualNetworkGatewayConnection
- Added examples for Remove-AzureRmVirtualNetworkGatewayIpConfig and Reset-AzureRmVirtualNetworkGateway
- Added example for Reset-AzureRmVirtualNetworkGatewayConnectionSharedKey
- Added example for Set-AzureRmVirtualNetworkGatewayConnectionSharedKey
- Added example for Set-AzureRmVirtualNetworkGatewayConnection
- Re-generated cmdlets for ApplicationSecurityGroup, RouteTable and Usage using latest code generator
- Clarified error message for Get-AzureRmVirtualNetworkSubnetConfig when attempting to get a subnet that does not exitc

AzureRM.NotificationHubs

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.OperationalInsights

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.PolicyInsights

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.PowerBIEmbedded

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.RecoveryServices

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.RecoveryServices.Backup

- Added policy filter to Get-AzureRmRecoveryServicesBackItem cmdlet. The command returns the list of backup items protected by the given policy id.
- Updated Microsoft.Azure.Management.RecoveryServices.Backup to version 3.0.0-preview.
- Updated to the latest version of the Azure ClientRuntime.
- Added TargetResourceGroupName parameter to Restore-AzureRmRecoveryServicesBackupItem. The resource group to which the managed disks are restored. Applicable to backup of VM with managed disks.

AzureRM.RecoveryServices.SiteRecovery

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.RedisCache

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Relay

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Resources

- Support template deployment at subscription scope. Add new Cmdlets:
  - New-AzureRmDeployment
  - Get-AzureRmDeployment
  - Test-AzureRmDeployment
  - Remove-AzureRmDeployment
  - Stop-AzureRmDeployment
  - Save-AzureRmDeploymentTemplate
  - Get-AzureRmDeploymentOperation
- Fix issue where error is thrown when passing a context to Set-AzureRmResource
  - https://github.com/Azure/azure-powershell/issues/5705
- Fix example in New-AzureRmResourceGroupDeployment
- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Scheduler

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.ServiceBus

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.ServiceFabric

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Sql

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Storage

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.StreamAnalytics

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Tags

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.TrafficManager

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.UsageAggregates

- Updated to the latest version of the Azure ClientRuntime.

AzureRM.Websites

- Updated to the latest version of the Azure ClientRuntime.

6.6.0 - July 2018

General

- Updated all help files to include full parameter types and correct input/output types.

AzureRM.Profile

- Updated Common.Strategy library to be able to validate that the current config for a resource is compatible with the target resource.
- Added ps1xml types to Common.Storage

Azure.Storage

- Added support for getting Storage Context from DefaultProfile
- Added Ps1XmlAttribute to cmdlets output types properties.

AzureRM.ApiManagement

- Fixed issue https://github.com/Azure/azure-powershell/issues/6370
  - Fixed bug in Automapper to translate PsApiManagementApi to ApiContract
- Fixed issue https://github.com/Azure/azure-powershell/issues/6515
  - Fixed bug in File.Save to not overload with Encoding Type
- Fixed issue https://github.com/Azure/azure-powershell/issues/6560
  - Upgraded to 4.0.3 Nuget version which fixes the pattern exception on apiId

AzureRM.Compute

- Fix issue with creating a vm using DiskFileParameterSet in New-AzureRmVm failing because of PremiumLRS storage account type renaming.
- Fix Invoke-AzureRmVMRunCommand cmdlet
- Update Get-AzureRmAvailabilitySet to enable list all availability sets in a subscription. (ResouceGroupName parameter is now optional.)
- Update SimpleParameterSet of 'New-AzureRmVm' to enable Accelerated Network on qualifying vms.
- Update New-AzureRmVmss simple parameter set to fail creating the vmss when a user specified LB already exists.
- Update example for New-AzureRmDisk
- Add example for 'New-AzureRmVM'
- Update description for Set-AzureRmVMOSDisk
- Update Example 1 for Set-AzureRmVMBginfoExtension to correct spelling and prefix.

AzureRM.DataFactoryV2

- Updated the ADF .Net SDK version to 1.1.0.
- Support self-hosted integration runtime sharing across data factories.
  - Add new parameter -SharedIntegrationRuntimeResourceId to Set-AzureRmDataFactoryV2IntegrationRuntime cmdlet.
  - Add new optional parameter -LinkedDataFactoryName to Remove-AzureRmDataFactoryV2IntegrationRuntime cmdlet.

AzureRM.DataLakeStore

- Updated the DataPlane SDK (Microsoft.Azure.DataLake.Store) version to 1.1.9

AzureRM.EventHub

- Updated piping for InputObject and ResourceId in remove cmdlets

AzureRM.Insights

- Fixed formatting of OutputType in help files
- Using Microsoft.Azure.Management.Monitor SDK 0.19.1-preview

AzureRM.KeyVault

- Fix piping issue in Set-AzureRmKeyVaultAccessPolicy

AzureRM.Network

- Added examples for LoadBalancerInboundNatPoolConfig cmdlets.

AzureRM.Resources

- Fix issue when providing both tag name and value for 'Get-AzureRmResource'
  - https://github.com/Azure/azure-powershell/issues/6765
- Fix piping scenario with 'Set-AzureRmResource'

AzureRM.ServiceBus

- Updated piping for InputObject and ResourceId in remove cmdlets
- fixed few issues

  - https://github.com/Azure/azure-powershell/issues/3780
  - https://github.com/Azure/azure-powershell/issues/4340

AzureRM.Sql

- Adding Server Advanced Threat Protection support with the following cmdlets:
  - Enable-AzureRmSqlServerAdvancedThreatProtection; Disable-AzureRmSqlServerAdvancedThreatProtection; Get-AzureRmSqlServerAdvancedThreatProtectionPolicy
- Adding Vulnerability Assessment support with the following cmdlets:
  - Update-AzureRmSqlDatabaseVulnerabilityAssessmentSettings; Get-AzureRmSqlDatabaseVulnerabilityAssessmentSettings; Clear-AzureRmSqlDatabaseVulnerabilityAssessmentSettings
  - Set-AzureRmSqlDatabaseVulnerabilityAssessmentRuleBaseline; Get-AzureRmSqlDatabaseVulnerabilityAssessmentRuleBaseline; Clear-AzureRmSqlDatabaseVulnerabilityAssessmentRuleBaseline
  - Convert-AzureRmSqlDatabaseVulnerabilityAssessmentScan; Get-AzureRmSqlDatabaseVulnerabilityAssessmentScanRecord; Start-AzureRmSqlDatabaseVulnerabilityAssessmentScan
- Fixed example in Remove-AzureRmSqlServerFirewallRule
- Fix datetime handling incorrectly for non-us base culture in Get-AzureSqlSyncGroupLog

AzureRM.Storage

- Add Ps1XmlAttribute to cmdlets output types properties
- Show StorageAccount cmdlet output in table view
  - Get-AzureRmStorageAccount
  - New-AzureRmStorageAccount
  - Set-AzureRmStorageAccount

AzureRM.Tags

- Remove incorrect statement from Tag cmdlet help
  - https://github.com/Azure/azure-powershell/issues/3878

6.5.0 - July 2018

AzureRM.Profile

- Updated help for 'Get-AzureRmContextAutosaveSetting'

Azure.Storage

- Support Upload Blob or File with write only Sas token
- Set-AzureStorageBlobContent
- Set-AzureStorageFileContent

AzureRM.AnalysisServices

- Add required property ResourceGroupName to AS.

AzureRM.Automation

- Update help and add example for 'New-AzureRMAutomationSchedule'

AzureRM.Compute

- Add -Tag parameter to Update/New-AzureRmAvailabilitySet
- Add example for 'Add-AzureRmVmssExtension'
- Add examples for 'Remove-AzureRmVmssExtension'
- Update help for 'Set-AzureRmVMAccessExtension'
- Update SimpleParameterSet for New-AzureRmVmss to set SinglePlacementGroup to false by default and add

switch parameter 'SinglePlacementGroup' that enables the user to create the VMSS in a single placement group.

AzureRM.EventHub

- Added a readonly property 'PendingReplicationOperationsCount' to PSEventHubDRConfigurationAttributes class, which gives the pending replication operations count while replication is in progress

AzureRM.KeyVault

- Update error message for Set-AzureRmKeyVaultAccessPolicy

AzureRM.LogicApp

- Fixed "parameter set could not be resolved" error in New-AzureRmLogicApp

AzureRM.Network

- Enable peering across Virtual Networks in multiple Tenants for Set/Add-AzureRmVirtualNetworkPeering
- Updated below cmdlets for Application Gateway
  - New-AzureRmApplicationGateway : Added EnableFIPS flag and Zones support
  - New-AzureRmApplicationGatewaySku : Added new skus Standard_v2 and WAF_v2
  - Set-AzureRmApplicationGatewaySku : Added new skus Standard_v2 and WAF_v2
- Regenerated RouteTable cmdlets with the latest generator version

AzureRM.Relay

- Updated markdown files, fix for the parameter name issue in example

AzureRM.Resources

- Update Roleassignment and roledefinition cmdlets:
  - Remove extra roledefinition calls done as part of paging.
- Fix Get-AzureRmRoleAssignment cmdlet
  - Fix -ExpandPrincipalGroups command parameter functionality
- Fix issue with 'Get-AzureRmResource' where '-ResourceType' parameter was case sensitive

AzureRM.ServiceBus

- Added top and skip parameter to list cmdlets
- Added Standard to Premium NameSpace migration cmdlets :
  - Start-AzureRmServiceBusMigration
  - Get-AzureRmServiceBusMigration
  - Complete-AzureRmServiceBusMigration
  - Stop-AzureRmServiceBusMigration
  - Remove-AzureRmServiceBusMigration
- Added a readonly property 'PendingReplicationOperationsCount' to PSServiceBusDRConfigurationAttributes class, which gives the pending replication operations count while replication is in progress

AzureRM.ServiceFabric

- Update example for 'New-AzureRmServiceFabricCluster'

AzureRM.Sql

- Adding new Cmdlets for Management.Sql to allow customers to add TDE Certificate to Sql Server instance or a Managed Instance
  - Add-AzureRmSqlServerTransparentDataEncryptionCertificate
  - Add-AzureRmSqlManagedInstanceTransparentDataEncryptionCertificate

AzureRM.Websites

- `Set-AzureRmWebApp -AssignIdentity` and `Set-AzureRmWebAppSlot -AssignIdentity` when set to false will now remove the Identity property from the site object.Removing preview tag as well.

- `Get-AzureRmWebAppMetrics`, `Get-AzureRmAppServicePlanMetrics` example updated
- `Set-AzureRmWebApp -PhpVersion` supports off as a valid php version

6.4.0 - July 2018

General

- Fixed formatting of OutputType in help files for most modules

AzureRM.Profile

- Ps1Xml attribute added to the basic output types

AzureRM.Compute

- IP Tag feature for VMSS
  - 'New-AzureRmVmssIpTagConfig' cmdlet is added
  - IpTag parameter is added to New-AzureRmVmssIpConfig
- Auto OS Rollback feature for VMSS
  - DisableAutoRollback parameters are added to New-AzureRmVmssConfig and Update-AzureRmVmss
- OS Upgrade History feature for Vmss
  - OSUpgradeHistory switch parameter is added to Get-AzureRmVmss

AzureRM.DataLakeAnalytics

- Add support for Catalog ACLs through the following commands:
  - Get-AzureRmDataLakeAnalyticsCatalogItemAclEntry
  - Set-AzureRmDataLakeAnalyticsCatalogItemAclEntry
  - Remove-AzureRmDataLakeAnalyticsCatalogItemAclEntry

AzureRM.DataLakeStore

- Add cancellation support and progress tracking for Set-AzureRmDataLakeStoreItemAclEntry, Remove-AzureRmDataLakeStoreItemAclEntry, Set-AzureRmDataLakeStoreItemAcl
- Add cancellation support for Export-AzureRmDataLakeStoreChildItemProperties
- Fix flushing of debug messages for cmdlets that does recursive operations
- Fix location of test of DataLake cmdlets

AzureRM.EventHub

- Added Optional MaxCount parameter to List Operations cmdlet Get-AzureRmEventHub and Get-AzureRmEventHubConsumerGroup
- Fixed issue in New-AzureRmEventHub cmdlet where at least one parameter needed while creating New EventHub. Provided Default Parameter set.
- Added optional Parameter -KeyValue to New-AzureRmEventHubKey cmdlet, which enables user to provide KeyValue.

AzureRM.KeyVault

- Fix issue where all resources were being returned by Get-AzureRmKeyVault -Tag

AzureRM.Network

- Expose new Skus for Zone-Redundant VirtualNetworkGateways
- Added new commands for feature: ExpressRoute Partner APIs via ARM
  - Added Get-AzureRmExpressRouteCrossConnection
  - Added Set-AzureRmExpressRouteCrossConnection
  - Added Add-AzureRmExpressRouteCrossConnectionPeering
  - Added Get-AzureRmExpressRouteCrossConnectionPeering
  - Added Remove-AzureRmExpressRouteCrossConnectionPeering
  - Added Get-AzureRMExpressRouteCrossConnectionArpTable

  - Added Get-AzureRMExpressRouteCrossConnectionRouteTable
  - Added Get-AzureRMExpressRouteCrossConnectionRouteTableSummary

AzureRM.RecoveryServices.Backup

- Added Get-AzureRmRecoveryServicesBackupStatus cmdlet. This cmdlet takes a VM ID and checks if the VM is protected by some vault in the subscription. If there exists such a vault, the cmdlet outputs the vault details.

AzureRM.Resources

- Update Get-AzureRmPolicyAssignment cmdlets:
  - Add support for listing -Scope values at management group level
  - Add support for retrieving individual assignments with -Scope values at management group level
  - Add -Effective and -All switches to control parameter
- Update Get/New/Remove/Set-AzureRmPolicyDefinition cmdlets
  - Add -ManagementGroupName parameter to apply operations to a given management group
  - Add -SubscriptionId parameter to apply operations to a given subscription
- Update Get/New/Remove/Set-AzureRmPolicySetDefinition cmdlets
  - Add -ManagementGroupName parameter to apply operations to a given management group
  - Add -SubscriptionId parameter to apply operations to a given subscription
- Add KeyVault secret reference support in parameters when using 'TemplateParameterObject' in 'New-AzureRmResourceGroupDeployment'
- Fix issue where '-EndDate' parameter was ignored for 'New-AzureRmADAppCredential'
  - https://github.com/Azure/azure-powershell/issues/6505
- Fix issue where 'Add-AzureRmADGroupMember' used incorrect URL to make request
  - https://github.com/Azure/azure-powershell/issues/6485

AzureRM.ServiceBus

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

AzureRM.Sql

- Clarified User-Defined Restore Points for SQLDW in New-AzureRmSqlDatabaseRestorePoint help
- Updated documentation of -ComputeGeneration parameter in several cmdlets

6.3.0 - June 2018

AzureRM.Profile

- Updated error messages for Enable-AzureRmContextAutoSave
- Create a context for each subscription when running 'Connect-AzureRmAccount' with no previous context

Azure.Storage

- Added additional information about -Permissions parameter in help files.

AzureRM.Compute

- 'Get-AzureRmVmDiskEncryptionStatus' fixes an issue observed for VMs with no data disks
- Update Compute client library version to fix following cmdlets
  - Grant-AzureRmDiskAccess
  - Grant-AzureRmSnapshotAccess
  - Save-AzureRmVMImage
- Fixed following cmdlets to show 'operation ID' and 'operation status' correctly:
  - Start-AzureRmVM
  - Stop-AzureRmVM
  - Restart-AzureRmVM

    - Set-AzureRmVM
    - Remove-AzuerRmVM
    - Set-AzureRmVmss
    - Start-AzureRmVmssRollingOSUpgrade
    - Stop-AzureRmVmssRollingUpgrade
    - Start-AzureRmVmss
    - Restart-AzureRmVmss
    - Stop-AzureRmVmss
    - Remove-AzureRmVmss
    - ConvertTo-AzureRmVMManagedDisk
    - Revoke-AzureRmSnapshotAccess
    - Remove-AzureRmSnapshot
    - Revoke-AzureRmDiskAccess
    - Remove-AzureRmDisk
    - Remove-AzureRmContainerService
    - Remove-AzureRmAvailabilitySet

AzureRM.EventGrid

- Remove ValidateNotNullOrEmpty validation conditions for SubjectBeginsWith/SubjectEndsWith in Update-AzureRmEventGridSubscription cmdlet to allow changing these parameters to empty string if needed.

AzureRM.KeyVault

- Fix issue where no Tags are being returned when Get-AzureRmKeyVault -Tag is run

AzureRM.PolicyInsights

- Public release of Policy Insights cmdlets
    - Use API version 2018-04-04
    - Add PolicyDefinitionReferenceId to the results of Get-AzureRmPolicyStateSummary

AzureRM.RecoveryServices.Backup

- Added -Vault parameter to RecoveryServices.Backup cmdlets. When passed, this will override the Set-AzureRmRecoveryServicesContext cmdlet.

AzureRM.Sql

- Updated example in the help file for Get-AzureRmSqlDatabaseExpanded

AzureRM.TrafficManager

- Updated the help file for Add-AzureRmTrafficManagerEndpointConfig

AzureRM.Websites

- 'Set-AzureRmWebApp' is updated to not overwrite the AppSettings when using -AssignIdentity
- 'New-AzureRmWebAppSlot' is updated to honor AppServicePlan as an optional parameter

6.2.1 - June 2018

AzureRM.OperationalInsights

- Updated PSWorkspace model to allow Network to use type as a parameter

6.2.0 - June 2018

AzureRM.Profile

- Fix issue where version 10.0.3 of Newtonsoft.Json wasn't being loaded on module import

AzureRM.Compute

- VMSS VM Update feature
  - Added 'Update-AzureRmVmssVM' and 'New-AzureRmVMDataDisk' cmdlets
  - Add VirtualMachineScaleSetVM parameter to 'Add-AzureRmVMDataDisk' cmdlet to support adding a data disk to Vmss VM.

AzureRM.DataFactoryV2

- Updated the ADF .Net SDK version to 0.8.0-preview containing following changes:
  - Added Configure factory repository operation
  - Updated QuickBooks LinkedService to expose consumerKey and consumerSecret properties
  - Updated Several model types from SecretBase to Object
  - Added Blob Events trigger

AzureRM.KeyVault

- Update documentation with example output

AzureRM.Network

- Enable Traffic Analytics parameters on Network Watcher cmdlets

AzureRM.Resources

- Fix issue with 'Properties' property of 'PSResource' object(s) returned from 'Get-AzureRmResource'

AzureRM.Scheduler

- Fix issue with update ServiceBusQueueJob not setting new Auth values

AzureRM.Sql

- Updated the following cmdlets with optional LicenseType parameter
  - New-AzureRmSqlDatabase; Set-AzureRmSqlDatabase
  - New-AzureRmSqlElasticPool; Set-AzureRmSqlElasticPool
  - New-AzureRmSqlDatabaseCopy
  - New-AzureRmSqlDatabaseSecondary
  - Restore-AzureRmSqlDatabase

AzureRM.Websites

- 'New-AzureRMWebApp' is updated to use common algorithms from the Strategy library.

6.1.0 - May 2018

AzureRM.Profile

- Fix issue where running 'Clear-AzureRmContext' would keep an empty context with the name of the previous default context, which prevented the user from creating a new context with the old name

AzureRM.AnalysisServices

- Enable Gateway assocaite/disassociate operations on AS.

AzureRM.ApiManagement

- Added support for ApiVersions, ApiReleases and ApiRevisions
- Added suppport for ServiceFabric Backend
- Added support for Application Insights Logger
- Added support for recognizing 'Basic' sku as a valid sku of Api Management service
- Added support for installing Certificates issued by private CA as Root or CA
- Added support for accepting Custom SSL certificates via KeyVault and Multiple proxy hostnames
- Added support for MSI identity
- Added support for accepting Policies via Url NOTE: The following cmdlets will be deprecated in future release
  - Import-AzureRmApiManagementHostnameCertificate

    - New-AzureRmApiManagementHostnameConfiguration
    - Set-AzureRmApiManagementHostnames
    - Update-AzureRmApiManagementDeployment

AzureRM.Batch

- Release new cmdlet Get-AzureBatchPoolNodeCounts
- Release new cmdlet Start-AzureBatchComputeNodeServiceLogUpload

AzureRM.Consumption

- Add new parameters Expand, ResourceGroup, InstanceName, InstanceId, Tags, and Top on Cmdlet Get-AzureRmConsumptionUsageDetail

AzureRM.DataLakeStore

- Fix example for Export-AzureRmDataLakeStoreChildItemProperties
- Fix null parameter exception for Recurse case in Set-AzureRmDataLakeStoreItemAclEntry
- Fix the help files for Set-AzureRmDataLakeStoreItemAclEntry, Set-AzureRmDataLakeStoreItemAcl, Remove-AzureRmDataLakeStoreItemAclEntry

AzureRM.Network

- Bump up Network SDK version from 18.0.0-preview to 19.0.0-preview
- Added cmdlet to create protocol configuration
    - New-AzureRmNetworkWatcherProtocolConfiguration
- Added cmdlet to add a new circuit connection to an existing express route circuit.
    - Add-AzureRmExpressRouteCircuitConnectionConfig
- Added cmdlet to remove a circuit connection from an existing express route circuit.
    - Remove-AzureRmExpressRouteCircuitConnectionConfig
- Added cmdlet to retrieve a circuit connection
    - Get-AzureRmExpressRouteCircuitConnectionConfig

AzureRM.ServiceFabric

- Fixed server authentication usage with generated certificates (Issue #5998)

AzureRM.Sql

- Updated Auditing cmdlets to allow removing AuditActions or AuditActionGroups
- Fixed issue with Set-AzureRmSqlDatabaseBackupLongTermRetentionPolicy when setting a new flexible retention policy where the command would fail with 'Configure long term retention policy with azure recovery service vault and policy is no longer supported. Please submit request with the new flexible retention policy'.
- Update all Azure Sql Database/ElasticPool Creation/Update related cmdlets to use the new Database API, which support Sku property for scale and tier-related properties.
- The updated cmdlets including:
    - New-AzureRmSqlDatabase; Set-AzureRmSqlDatabase
    - New-AzureRmSqlElasticPool; Set-AzureRmSqlElasticPool
    - New-AzureRmSqlDatabaseCopy
    - New-AzureRmSqlDatabaseSecondary
    - Restore-AzureRmSqlDatabase

AzureRM.TrafficManager

- Update the parameters for 'Get-AzureRmTrafficManagerProfile' so that -ResourceGroupName parameter is required when using -Name parameter.

Breaking changes for Microsoft Azure PowerShell 6.0.0

11/20/2018 • 8 minutes to read • Edit Online

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

Table of Contents

- General breaking changes
  - Minimum PowerShell version required bumped to 5.0
  - Context autosaved enabled by default
  - Removal of Tags alias
- Breaking changes to AzureRM.Compute cmdlets
- Breaking changes to AzureRM.DataLakeStore cmdlets
- Breaking changes to AzureRM.Dns cmdlets
- Breaking changes to AzureRM.Insights cmdlets
- Breaking changes to AzureRM.KeyVault cmdlets
- Breaking changes to AzureRM.Network cmdlets
- Breaking changes to AzureRM.RedisCache cmdlets
- Breaking changes to AzureRM.Resources cmdlets
- Breaking changes to AzureRM.Storage cmdlets
- Removed modules
  - `AzureRM.ServerManagement`
  - `AzureRM.SiteRecovery`

General breaking changes

Minimum PowerShell version required bumped to 5.0

Previously, Azure PowerShell required *at least* version 3.0 of PowerShell to run any cmdlet. Moving forward, this requirement will be raised to version 5.0 of PowerShell. For information on upgrading to PowerShell 5.0, please see this table.

Context autosave enabled by default

Context autosave is the storage of Azure sign in information that can be used between new and different PowerShell sessions. For more information on context autosave, please see this document.

Previously by default, context autosave was disabled, which meant the user's Azure authentication information was not stored between sessions until they ran the `Enable-AzureRmContextAutosave` cmdlet to turn on context persistence. Moving forward, context autosave will be enabled by default, which means that users *with no saved context autosave settings* will have their context stored the next time they sign in. Users can opt out of this functionality by using the `Disable-AzureRmContextAutosave` cmdlet.

Note: users that previously disabled context autosave or users with context autosave enabled and existing contexts will not be affected by this change

Removal of Tags alias

The alias `Tags` for the `Tag` parameter has been removed across numerous cmdlets. Below is a list of modules (and the corresponding cmdlets) affected by this:

`AzureRM.ApiManagement`

- `New-AzureRmApiManagement`
- `New-AzureRmApiManagementProperty`
- `Set-AzureRmApiManagementProperty`

`AzureRM.Automation`

- `Set-AzureRmAutomationRunbook`

`AzureRM.Cdn`

- `New-AzureRmCdnEndpoint`
- `New-AzureRmCdnProfile`

`AzureRM.Compute`

- `New-AzureRmVM`
- `Update-AzureRmVM`

`AzureRM.DataFactories`

- `New-AzureRmDataFactories`

`AzureRM.DataLakeAnalytics`

- `New-AzureRmDataLakeAnalyticsAccount`

`AzureRM.DataLakeStore`

- `New-AzureRmDataLakeStoreAccount`
- `Set-AzureRmDataLakeStoreAccount`

`AzureRM.MachineLearning`

- `Update-AzureRmMlCommitmentPlan`

`AzureRM.Media`

- `Set-AzureRmMediaService`

`AzureRM.OperationalInsights`

- `New-AzureRmOperationalInsightsSavedSearch`
- `New-AzureRmOperationalInsightsWorkspace`
- `Set-AzureRmOperationalInsightsSavedSearch`
- `Set-AzureRmOperationalInsightsWorkspace`

Breaking changes to AzureRM.Compute cmdlets

Miscellaneous

- The sku name property nested in types `PSDisk` and `PSSnapshot` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

```
$disk = Get-AzureRmDisk -ResourceGroupName "MyResourceGroup" -DiskName "MyDiskName"
$disk.Sku.Name       # This will now return Standard_LRS or Premium_LRS

$snapshot = Get-AzureRmSnapshot -ResourceGroupName "MyResourceGroup" -SnapshotName "MySnapshotName"
$snapshot.Sku.Name   # This will now return Standard_LRS or Premium_LRS
```

- The storage account type property nested in types `PSVirtualMachine`, `PSVirtualMachineScaleSet` and `PSImage` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

```
$vm = Get-AzureRmVM -ResourceGroupName "MyResourceGroup" -Name "MyVM"
$vm.StorageProfile.DataDisks[0].ManagedDisk.StorageAccountType   # This will now return Standard_LRS or
Premium_LRS
```

Add-AzureRmImageDataDisk

- The accepted values for parameter `StorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Add-AzureRmVMDataDisk

- The accepted values for parameter `StorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Add-AzureRmVmssDataDisk

- The accepted values for parameter `StorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

New-AzureRmAvailabilitySet

- The parameter `Managed` was removed in favor of `Sku`

```
# Old
New-AzureRmAvailabilitySet -ResourceGroupName "MyRG" -Name "MyAvailabilitySet" -Location "West US" -Managed

# New
New-AzureRmAvailabilitySet -ResourceGroupName "MyRG" -Name "MyAvailabilitySet" -Location "West US" -Sku
"Aligned"
```

New-AzureRmDiskConfig

- The accepted values for parameter `SkuName` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

New-AzureRmDiskUpdateConfig

- The accepted values for parameter `SkuName` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

New-AzureRmSnapshotConfig

- The accepted values for parameter `SkuName` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

New-AzureRmSnapshotUpdateConfig

- The accepted values for parameter `SkuName` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Set-AzureRmImageOsDisk

- The accepted values for parameter `StorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Set-AzureRmVMAEMExtension

- The parameter `DisableWAD` was removed
  - Windows Azure Diagnostics is disabled by default

Set-AzureRmVMDataDisk

- The accepted values for parameter `StorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Set-AzureRmVMOSDisk

- The accepted values for parameter `StorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Set-AzureRmVmssStorageProfile

- The accepted values for parameter `ManagedDisk` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Update-AzureRmVmss

- The accepted values for parameter `ManagedDiskStorageAccountType` changed from `StandardLRS` and `PremiumLRS` to `Standard_LRS` and `Premium_LRS`, respectively

Breaking changes to AzureRM.DataLakeStore cmdlets

Export-AzureRmDataLakeStoreItem

- Parameters `PerFileThreadCount` and `ConcurrentFileCount` were removed. Please use the `Concurrency` parameter moving forward

```
# Old
Export-AzureRmDataLakeStoreItem -Account contoso -Path /test -Destination C:\test -Recurse -Resume -
PerFileThreadCount 2 -ConcurrentFileCount 80

# New
Export-AzureRmDataLakeStoreItem -Account contoso -Path /test -Destination C:\test -Recurse -Resume -
Concurrency 160
```

Import-AzureRmDataLakeStoreItem

- Parameters `PerFileThreadCount` and `ConcurrentFileCount` were removed. Please use the `Concurrency` parameter moving forward

```
# Old
Import-AzureRmDataLakeStoreItem -Account contoso -Path C:\test -Destination /test -Recurse -Resume -
ForceBinary -PerFileThreadCount 2 -ConcurrentFileCount 80

# New
Import-AzureRmDataLakeStoreItem -Account contoso -Path C:\test -Destination /test -Recurse -Resume -
ForceBinary -Concurrency 160
```

Remove-AzureRmDataLakeStoreItem

- Parameter `Clean` was removed

```
# Old
Remove-AzureRmDataLakeStoreItem -Account "ContosoADL" -path /myFolder -Recurse -Clean

# New
Remove-AzureRmDataLakeStoreItem -Account "ContosoADL" -path /myFolder -Recurse
```

Breaking changes to AzureRM.Dns cmdlets

New-AzureRmDnsRecordSet

- The parameter `Force` was removed

Remove-AzureRmDnsRecordSet

- The parameter `Force` was removed

Remove-AzureRmDnsZone

- The parameter `Force` was removed

Breaking changes to AzureRM.Insights cmdlets

Add-AzureRmAutoscaleSetting

- The parameter aliases `AutoscaleProfiles` and `Notifications` were removed

Add-AzureRmLogProfile

- The parameter aliases `Categories` and `Locations` were removed

Add-AzureRmMetricAlertRule

- The parameter alias `Actions` was removed

Add-AzureRmWebtestAlertRule

- The parameter alias `Actions` was removed

Get-AzureRmLog

- The parameter aliases `MaxRecords` and `MaxEvents` were removed

Get-AzureRmMetricDefinition

- The parameter alias `MetricNames` was removed

New-AzureRmAlertRuleEmail

- The parameter aliases `CustomEmails` and `SendToServiceOwners` were removed

New-AzureRmAlertRuleWebhook

- The parameter alias `Properties` was removed

New-AzureRmAutoscaleNotification

- The parameter aliases `CustomEmails`, `SendEmailToSubscriptionCoAdministrators` and `Webhooks` were removed

New-AzureRmAutoscaleProfile

- The parameter aliases `Rules`, `ScheduleDays`, `ScheduleHours` and `ScheduleMinutes` were removed

New-AzureRmAutoscaleWebhook

- The parameter alias `Properties` was removed

Breaking changes to AzureRM.KeyVault cmdlets

Add-AzureKeyVaultCertificate

- The `CertificatePolicy` parameter has become mandatory.

Set-AzureKeyVaultManagedStorageSasDefinition

- The cmdlet no longer accepts individual parameters that compose the access token; instead, the cmdlet replaces explicit token parameters, such as `Service` or `Permissions`, with a generic `TemplateUri` parameter, corresponding to a sample access token defined elsewhere (presumably using Storage PowerShell cmdlets, or composed manually according to the Storage documentation.) The cmdlet retains the `ValidityPeriod` parameter.

For more information on composing shared access tokens for Azure Storage, please refer to the documentation pages, respectively:

- Constructing a Service SAS
- Constructing an Account SAS

```
# Old
$sas = Set-AzureKeyVaultManagedStorageSasDefinition -VaultName myVault -Name myKey -Service Blob -Permissions
'rcw' -ValidityPeriod 180d

# New
$sctx=New-AzureStorageContext -StorageAccountName $sa.StorageAccountName -Protocol Https -StorageAccountKey
Key1
$start=[System.DateTime]::Now.AddDays(-1)
$end=[System.DateTime]::Now.AddMonths(1)
$at=New-AzureStorageAccountSasToken -Service blob -ResourceType Service,Container,Object -Permission
"racwdlup" -Protocol HttpsOnly -StartTime $start -ExpiryTime $end -Context $sctx
$sas=Set-AzureKeyVaultManagedStorageSasDefinition -AccountName $sa.StorageAccountName -VaultName $kv.VaultName
-Name accountsas -TemplateUri $at -SasType 'account' -ValidityPeriod ([System.Timespan]::FromDays(30))
```

Set-AzureKeyVaultCertificateIssuer

- The `IssuerProvider` parameter has become mandatory.

Undo-AzureKeyVaultCertificateRemoval

- The output of this cmdlet has changed from `CertificateBundle` to `PSKeyVaultCertificate`.

Undo-AzureRmKeyVaultRemoval

- `ResourceGroupName` has been removed from the `InputObject` parameter set, and is instead obtained from the `InputObject` parameter's `ResourceId` property.

Set-AzureRmKeyVaultAccessPolicy

- The `all` permission was removed from `PermissionsToKeys`, `PermissionsToSecrets`, and `PermissionsToCertificates`.

General

- The `ValueFromPipelineByPropertyName` property was removed from all cmdlets where piping by `InputObject` was enabled. The cmdlets affected are:

- `Add-AzureKeyVaultCertificate`
- `Add-AzureKeyVaultCertificateContact`
- `Add-AzureKeyVaultKey`
- `Backup-AzureKeyVaultKey`
- `Backup-AzureKeyVaultSecret`
- `Get-AzureKeyVaultCertficate`
- `Get-AzureKeyVaultCertificateContact`
- `Get-AzureKeyVaultCertificateIssuer`
- `Get-AzureKeyVaultCertificateOperation`
- `Get-AzureKeyVaultCertificatePolicy`
- `Get-AzureKeyVaultKey`
- `Get-AzureKeyVaultManagedStorageAccount`
- `Get-AzureKeyVaultManagedStorageSasDefinition`
- `Get-AzureKeyVaultSecret`
- `Remove-AzureRmKeyVault`
- `Remove-AzureRmKeyVaultAccessPolicy`
- `Remove-AzureKeyVaultCertificate`
- `Remove-AzureKeyVaultCertificateContact`
- `Remove-AzureKeyVaultCertificateIssuer`
- `Remove-AzureKeyVaultCertificateOperation`
- `Remove-AzureKeyVaultKey`
- `Remove-AzureKeyVaultManagedStorageAccount`
- `Remove-AzureKeyVaultManagedStorageSasDefinition`
- `Remove-AzureKeyVaultSecret`
- `Restore-AzureKeyVaultKey`
- `Restore-AzureKeyVaultSecret`
- `Set-AzureRmKeyVaultAccessPolicy`
- `Set-AzureKeyVaultCertificateAttribute`
- `Set-AzureKeyVaultCertificateIssuer`
- `Set-AzureKeyVaultCertificatePolicy`
- `Set-AzureKeyVaultKeyAttribute`
- `Set-AzureKeyVaultManagedStorageSasDefinition`
- `Set-AzureKeyVaultSecret`
- `Set-AzureKeyVaultSecretAttribute`
- `Stop-AzureKeyVaultCertificateOperation`
- `Undo-AzureKeyVaultCertificateRemoval`
- `Undo-AzureKeyVaultKeyRemoval`
- `Undo-AzureRmKeyVaultRemoval`
- `Undo-AzureKeyVaultSecretRemoval`
- `Update-AzureKeyVaultManagedStorageAccount`
- `Update-AzureKeyVaultManagedStorageAccountKey`

- `ConfirmImpact` levels were removed from all cmdlets. The cmdlets affected are:
  - `Remove-AzureRmKeyVault`
  - `Remove-AzureKeyVaultCertificate`

    - `Remove-AzureKeyVaultCertificateIssuer`
    - `Remove-AzureKeyVaultCertificateOperation`
    - `Remove-AzureKeyVaultKey`
    - `Remove-AzureKeyVaultManagedStorageAccount`
    - `Remove-AzureKeyVaultManagedStorageSasDefinition`
    - `Remove-AzureKeyVaultSecret`
    - `Stop-AzureKeyVaultCertificateOperation`
    - `Update-AzureKeyVaultManagedStorageAccountKey`
- The `IKeyVaultDataServiceClient` was updated so all Certificate operations return PSTypes instead of SDK types. This includes:
    - `SetCertificateContacts`
    - `GetCertificateContacts`
    - `GetCertificate`
    - `GetDeletedCertificate`
    - `MergeCertificate`
    - `ImportCertificate`
    - `DeleteCertificate`
    - `RecoverCertificate`
    - `EnrollCertificate`
    - `UpdateCertificate`
    - `GetCertificateOperation`
    - `DeleteCertificateOperation`
    - `CancelCertificateOperation`
    - `GetCertificatePolicy`
    - `UpdateCertificatePolicy`
    - `GetCertificateIssuer`
    - `SetCertificateIssuer`
    - `DeleteCertificateIssuer`

Breaking changes to AzureRM.Network cmdlets

Add-AzureRmApplicationGatewayBackendHttpSettings

- The parameter `ProbeEnabled` was removed

Add-AzureRmVirtualNetworkPeering

- The parameter alias `AlloowGatewayTransit` was removed

New-AzureRmApplicationGatewayBackendHttpSettings

- The parameter `ProbeEnabled` was removed

Set-AzureRmApplicationGatewayBackendHttpSettings

- The parameter `ProbeEnabled` was removed

Breaking changes to AzureRM.RedisCache cmdlets

New-AzureRmRedisCache

- The parameters `Subnet` and `VirtualNetwork` were removed in favor of `SubnetId`
- The parameter `RedisVersion` was removed
- The parameter `MaxMemoryPolicy` was removed in favor of `RedisConfiguration`

```
# Old
New-AzureRmRedisCache -ResourceGroupName "MyRG" -Name "MyRedisCache" -Location "North Central US" -
MaxMemoryPolicy "allkeys-lru"

# New
New-AzureRmRedisCache -ResourceGroupName "MyRG" -Name "MyRedisCache" -Location "North Central US" -
RedisConfiguration @{"maxmemory-policy" = "allkeys-lru"}
```

Set-AzureRmRedisCache

- The parameter `MaxMemoryPolicy` was removed in favor of `RedisConfiguration`

```
# Old
Set-AzureRmRedisCache -ResourceGroupName "MyRG" -Name "MyRedisCache" -MaxMemoryPolicy "allkeys-lru"

# New
Set-AzureRmRedisCache -ResourceGroupName "MyRG" -Name "MyRedisCache" -RedisConfiguration @{"maxmemory-policy"
= "allkeys-lru"}
```

Breaking changes to AzureRM.Resources cmdlets

Find-AzureRmResource

- This cmdlet was removed and the functionality was moved into `Get-AzureRmResource`

```
# Old
Find-AzureRmResource -ResourceType "Microsoft.Web/sites" -ResourceGroupNameContains "ResourceGroup"
Find-AzureRmResource -ResourceType "Microsoft.Web/sites" -ResourceNameContains "test"

# New
Get-AzureRmResource -ResourceType "Microsoft.Web/sites" -ResourceGroupName "*ResourceGroup*"
Get-AzureRmResource -ResourceType "Microsoft.Web/sites" -Name "*test*"
```

Find-AzureRmResourceGroup

- This cmdlet was removed and the functionality was moved into `Get-AzureRmResourceGroup`

```
# Old
Find-AzureRmResourceGroup
Find-AzureRmResourceGroup -Tag @{ "testtag" = $null }
Find-AzureRmResourceGroup -Tag @{ "testtag" = "testval" }

# New
Get-AzureRmResourceGroup
Get-AzureRmResourceGroup -Tag @{ "testtag" = $null }
Get-AzureRmResourceGroup -Tag @{ "testtag" = "testval" }
```

Get-AzureRmRoleDefinition

- Parameter `AtScopeAndBelow` was removed.

```
# Old
Get-AzureRmRoleDefinition [other required parameters] -AtScopeAndBelow

# New
Get-AzureRmRoleDefinition [other required parameters]
```

Breaking changes to AzureRM.Storage cmdlets

New-AzureRmStorageAccount

- The parameter `EnableEncryptionService` was removed

Set-AzureRmStorageAccount

- The parameters `EnableEncryptionService` and `DisableEncryptionService` were removed

Removed modules

`AzureRM.ServerManagement`

The Server Management Tools service was retired last year, and as a result, the corresponding module for SMT, `AzureRM.ServerManagement`, was removed from `AzureRM` and will stop shipping moving forward.

`AzureRM.SiteRecovery`

The `AzureRM.SiteRecovery` module is being superseded by `AzureRM.RecoveryServices.SiteRecovery`, which is a functional superset of the `AzureRM.SiteRecovery` module and includes a new set of equivalent cmdlets. The full list of mappings from old to new cmdlets can be found below:

DEPRECATED CMDLET	EQUIVALENT CMDLET	ALIASES
`Edit-AzureRmSiteRecoveryRecoveryPlan`	`Edit-AzureRmRecoveryServicesAsrRecoveryPlan`	`Edit-ASRRecoveryPlan`
`Get-AzureRmSiteRecoveryFabric`	`Get-AzureRmRecoveryServicesAsrFabric`	`Get-ASRFabric`
`Get-AzureRmSiteRecoveryJob`	`Get-AzureRmRecoveryServicesAsrJob`	`Get-ASRJob`
`Get-AzureRmSiteRecoveryNetwork`	`Get-AzureRmRecoveryServicesAsrNetwork`	`Get-ASRNetwork`
`Get-AzureRmSiteRecoveryNetworkMapping`	`Get-AzureRmRecoveryServicesAsrNetworkMapping`	`Get-ASRNetworkMapping`
`Get-AzureRmSiteRecoveryPolicy`	`Get-AzureRmRecoveryServicesAsrPolicy`	`Get-ASRPolicy`
`Get-AzureRmSiteRecoveryProtectableItem`	`Get-AzureRmRecoveryServicesAsrProtectableItem`	`Get-ASRProtectableItem`
`Get-AzureRmSiteRecoveryProtectionContainer`	`Get-AzureRmRecoveryServicesAsrProtectionContainer`	`Get-ASRProtectionContainer`
`Get-AzureRmSiteRecoveryProtectionContainer`	`Get-AzureRmRecoveryServicesAsrProtectionContainerMapping`	`Get-ASRProtectionContainerMapping`

DEPRECATED CMDLET	EQUIVALENT CMDLET	ALIASES
`Get-AzureRmSiteRecoveryProtectionEntity`	`Get-AzureRmRecoveryServicesAsrProtectableItem`	`Get-ASRProtectableItem`
`Get-AzureRmSiteRecoveryRecoveryPlan`	`Get-AzureRmRecoveryServicesAsrRecoveryPlan`	`Get-ASRRecoveryPlan`
`Get-AzureRmSiteRecoveryRecoveryPoint`	`Get-AzureRmRecoveryServicesAsrRecoveryPoint`	`Get-ASRRecoveryPoint`
`Get-AzureRmSiteRecoveryReplicationProtecte`	`Get-AzureRmRecoveryServicesAsrReplicationProtectedItem`	`Get-ASRReplicationProtectedItem`
`Get-AzureRmSiteRecoveryServer`	`Get-AzureRmRecoveryServicesAsrServicesProvider`	`Get-ASRServicesProvider`
`Get-AzureRmSiteRecoveryServicesProvider`	`Get-AzureRmRecoveryServicesAsrServicesProvider`	`Get-ASRServicesProvider`
`Get-AzureRmSiteRecoverySite`	`Get-AzureRmRecoveryServicesAsrFabric`	`Get-ASRFabric`
`Get-AzureRmSiteRecoveryStorageClassificati`	`Get-AzureRmRecoveryServicesAsrStorageClassification`	`Get-ASRStorageClassification`
`Get-AzureRmSiteRecoveryStorageClassificati`	`Get-AzureRmRecoveryServicesAsrStorageClass`	`Get-ASRStorageClassificationMapping`
`Get-AzureRmSiteRecoveryVault`	`Get-AzureRmRecoveryServicesVault`	
`Get-AzureRmSiteRecoveryVaultSettings`	`Get-AzureRmRecoveryServicesAsrVaultContext`	
`Get-AzureRmSiteRecoveryVaultSettingsFile`	`Get-AzureRmRecoveryServicesVaultSettingsFile`	
`Get-AzureRmSiteRecoveryVM`	`Get-AzureRmRecoveryServicesAsrReplicationProtectedItem`	`Get-ASRReplicationProtectedItem`
`Import-AzureRmSiteRecoveryVaultSettingsFile`	`Import-AzureRmRecoveryServicesAsrVaultSettingsFile`	
`New-AzureRmSiteRecoveryFabric`	`New-AzureRmRecoveryServicesAsrFabric`	`New-ASRFabric`
`New-AzureRmSiteRecoveryNetworkMapping`	`New-AzureRmRecoveryServicesAsrNetworkMapping`	`New-ASRNetworkMapping`
`New-AzureRmSiteRecoveryPolicy`	`New-AzureRmRecoveryServicesAsrPolicy`	`New-ASRPolicy`
`New-AzureRmSiteRecoveryProtectionContainer`	`New-AzureRmRecoveryServicesAsrProtectionContainerMapping`	`New-ASRProtectionContainerMapping`
`New-AzureRmSiteRecoveryRecoveryPlan`	`New-AzureRmRecoveryServicesAsrRecoveryPlan`	`New-ASRRecoveryPlan`

DEPRECATED CMDLET	EQUIVALENT CMDLET	ALIASES
`New-AzureRmSiteRecoveryReplicationProtecte`	`New-AzureRmRecoveryServicesAsrReplicationProtectedItem`	`New-ASRReplicationProtectedItem`
`New-AzureRmSiteRecoverySite`	`New-AzureRmRecoveryServicesAsrFabric`	`New-ASRFabric`
`New-AzureRmSiteRecoveryStorageClassificati`	`New-AzureRmRecoveryServicesAsrStorageClass`	`New-ASRStorageClassificationMapping`
`New-AzureRmSiteRecoveryVault`	`New-AzureRmRecoveryServicesVault`	
`Remove-AzureRmSiteRecoveryFabric`	`Remove-AzureRmRecoveryServicesAsrFabric`	`Remove-ASRFabric`
`Remove-AzureRmSiteRecoveryNetworkMapping`	`Remove-AzureRmRecoveryServicesAsrNetworkMapping`	`Remove-ASRNetworkMapping`
`Remove-AzureRmSiteRecoveryPolicy`	`Remove-AzureRmRecoveryServicesAsrPolicy`	`Remove-ASRPolicy`
`Remove-AzureRmSiteRecoveryProtectionContainer`	`Remove-AzureRmRecoveryServicesAsrProtectionCo`	`Remove-ASRProtectionContainerMapping`
`Remove-AzureRmSiteRecoveryRecoveryPlan`	`Remove-AzureRmRecoveryServicesAsrRecoveryPlan`	`Remove-ASRRecoveryPlan`
`Remove-AzureRmSiteRecoveryReplicationProtecte`	`Remove-AzureRmRecoveryServicesAsrReplicationP`	`Remove-ASRReplicationProtectedItem`
`Remove-AzureRmSiteRecoveryServer`	`Remove-AzureRmRecoveryServicesAsrServicesProvider`	
`Remove-AzureRmSiteRecoveryServicesProvider`	`Remove-AzureRmRecoveryServicesAsrServicesProvider`	`Remove-ASRServicesProvider`
`Remove-AzureRmSiteRecoverySite`	`Remove-AzureRmRecoveryServicesAsrFabric`	`Remove-ASRFabric`
`Remove-AzureRmSiteRecoveryStorageClassificati`	`Remove-AzureRmRecoveryServicesAsrStorageClass`	`Remove-ASRStorageClassificationMapping`
`Remove-AzureRmSiteRecoveryVault`	`Remove-AzureRmRecoveryServicesVault`	
`Restart-AzureRmSiteRecoveryJob`	`Restart-AzureRmRecoveryServicesAsrJob`	`Restart-ASRJob`
`Resume-AzureRmSiteRecoveryJob`	`Resume-AzureRmRecoveryServicesAsrJob`	`Resume-ASRJob`
`Set-AzureRmSiteRecoveryProtectionEntity`	`New-AzureRmRecoveryServicesAsrReplicationProtectedItem`	`New-ASRReplicationProtectedItem`
`Set-AzureRmSiteRecoveryReplicationProtecte`	`Set-AzureRmRecoveryServicesAsrReplicationProtectedItem`	`Set-ASRReplicationProtectedItem`

DEPRECATED CMDLET	EQUIVALENT CMDLET	ALIASES
`Set-AzureRmSiteRecoveryVaultSettings`	`Set-AzureRmRecoveryServicesAsrVaultContext`	`Set-ASRVaultContext`
`Set-AzureRmSiteRecoveryVM`	`Set-AzureRmRecoveryServicesAsrReplicationProtectedItem`	`Set-ASRReplicationProtectedItem`
`Start-AzureRmSiteRecoveryApplyRecoveryPoint`	`Start-AzureRmRecoveryServicesAsrApplyRecoveryPoint`	`Start-ASRApplyRecoveryPoint`
`Start-AzureRmSiteRecoveryCommitFailoverJob`	`Start-AzureRmRecoveryServicesAsrCommitFailoverJob`	`Start-ASRCommitFailoverJob`
`Start-AzureRmSiteRecoveryPlannedFailoverJob`	`Start-AzureRmRecoveryServicesAsrPlannedFailoverJob`	`Start-ASRPlannedFailoverJob`
`Start-AzureRmSiteRecoveryPolicyAssociationJo`	`New-AzureRmRecoveryServicesAsrProtectionContainerMapping`	`New-ASRProtectionContainerMapping`
`Start-AzureRmSiteRecoveryPolicyDissociationJ`	`Remove-AzureRmRecoveryServicesAsrProtectionCo`	`Remove-ASRProtectionContainerMapping`
`Start-AzureRmSiteRecoveryTestFailoverJob`	`Start-AzureRmRecoveryServicesAsrTestFailoverJob`	`Start-ASRTestFailoverJob`
`Start-AzureRmSiteRecoveryUnplannedFailoverJo`	`Start-AzureRmRecoveryServicesAsrUnplannedFailoverJob`	`Start-ASRUnplannedFailoverJob`
`Stop-AzureRmSiteRecoveryJob`	`Stop-AzureRmRecoveryServicesAsrJob`	`Stop-ASRJob`
`Update-AzureRmSiteRecoveryPolicy`	`Update-AzureRmRecoveryServicesAsrPolicy`	`Update-ASRPolicy`
`Update-AzureRmSiteRecoveryProtectionDirection`	`Update-AzureRmRecoveryServicesAsrProtectionDirection`	`Update-ASRProtectionDirection`
`Update-AzureRmSiteRecoveryRecoveryPlan`	`Update-AzureRmRecoveryServicesAsrRecoveryPlan`	`Update-ASRRecoveryPlan`
`Update-AzureRmSiteRecoveryServer`	`Update-AzureRmRecoveryServicesAsrServicesProvider`	`Update-ASRServicesProvider`
`Update-AzureRmSiteRecoveryServicesProvider`	`Update-AzureRmRecoveryServicesAsrvCenter`	`Update-ASRvCenter`