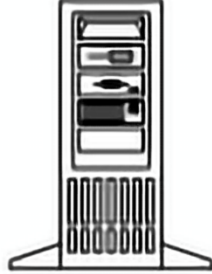


## Chapter 1: Installing and Configuring PowerShell 7



**SRV1**

Windows Server 2022

**Initial Reskit Workgroup**

```
PS C:\WINDOWS\system32> # 2. Update help text for Windows PowerShell
PS C:\WINDOWS\system32> Update-Help -Force | Out-Null


Update-Help : Failed to update Help for the module(s) 'ConfigDefender, ConfigDefenderPerformance, Dism, Get-NetView, Kds, Microsoft.ServerCore.SConfig, NetQos, PcsvDevice, PKI, PSReadline, RemoteDesktop, StorageBusCache, VMDirectStorage, Whea, WindowsUpdate' with UI culture(s) {en-US} : Unable to retrieve the HelpInfo XML file for UI culture en-US.
Make sure the HelpInfoUri property in the module manifest is valid or check your network connection and then try the command again.

At line:1 char:1
+ Update-Help -Force |
+ ~~~~~
+ CategoryInfo          : ResourceUnavailable: (:) [Update-Help], Exception
+ FullyQualifiedErrorId : UnableToRetrieveHelpInfoXml,Microsoft.PowerShell.Commands.UpdateHelpCommand
```

```
PS C:\WINDOWS\system32> # 5. Viewing Installation Script Help
PS C:\WINDOWS\system32> Get-Help -Name C:\Foo\Install-PowerShell.ps1
Install-PowerShell.ps1 [-Destination <string>] [-Daily] [-DoNotOverwrite] [-AddToPath] [-Preview] [<CommonParameters>]
Install-PowerShell.ps1 [-UseMSI] [-Quiet] [-AddExplorerContextMenu] [-EnablePSRemoting] [-Preview] [<CommonParameters>]
```

```
PS C:\WINDOWS\system32> # 6. Installing PowerShell 7.2
PS C:\WINDOWS\system32> EXHT = @{
    UseMSI           = $true
    Quiet            = $true
    AddExplorerContextMenu = $true
    EnablePSRemoting = $true
}
PS C:\WINDOWS\system32> C:\Foo\Install-PowerShell.ps1 @EXHT | Out-Null

VERBOSE: About to download package from 'https://github.com/PowerShell/PowerShell/releases/download/v7.2.2/PowerShell-7.2.2-win-x64.msi'
```

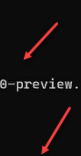


```
PS C:\WINDOWS\system32> # 7. Installing the preview and daily builds (for the adventurous)
PS C:\WINDOWS\system32> C:\Foo\Install-PowerShell.ps1 -Preview -Destination C:\PSPreview | Out-Null

Destination: C:\PSPreview
About to download package from 'https://github.com/PowerShell/PowerShell/releases/download/v7.3.0-preview.3/PowerShell-7.3.0-preview.3-win-x64.zip'
PowerShell has been installed at C:\PSPreview

PS C:\WINDOWS\system32> C:\Foo\Install-PowerShell.ps1 -Daily -Destination C:\PSDailyBuild | Out-Null

Destination: C:\PSDailyBuild
About to download package from 'https://pscoretestdata.blob.core.windows.net/v7-3-0-daily20220329-1/PowerShell-7.3.0-daily20220329.1-win-x64.zip'
PowerShell has been installed at C:\PSDailyBuild
```





```
PS C:\Foo> # 9. Checking versions of PowerShell 7 loaded
Get-ChildItem -Path C:\pwsh.exe -Recurse -ErrorAction SilentlyContinue
```

```
Directory: C:\Program Files\PowerShell\7
```

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a----	08/03/2022 23:21	287632	pwsh.exe

```
Directory: C:\PSDailyBuild
```

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a----	29/03/2022 18:24	286104	pwsh.exe

```
Directory: C:\PSPreview
```

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a----	18/03/2022 17:04	281512	pwsh.exe

```
PS C:\WINDOWS\system32> # 2. Viewing the installation help file
PS C:\WINDOWS\system32> C:\Foo\Install-Chocolatey.ps1 -?
```

NAME

C:\Foo\Install-Chocolatey.ps1

SYNOPSIS

Downloads and installs Chocolatey on the local machine.

SYNTAX

```
C:\Foo\Install-Chocolatey.ps1 [-ChocolateyDownloadUrl <String>]
                                [-ChocolateyVersion <String>]
                                [-UseNativeUnzip]
                                [-IgnoreProxy]
                                [<CommonParameters>]
```

```
C:\Foo\Install-Chocolatey.ps1 [-ChocolateyDownloadUrl <String>]
                                [-ChocolateyVersion <String>]
                                [-UseNativeUnzip]
                                [-IgnoreProxy]
                                [-ProxyUrl <String>]
                                [-ProxyCredential <PSCredential>]
                                [<CommonParameters>]
```

DESCRIPTION

Retrieves the Chocolatey nupkg for the latest or a specified version, and downloads and installs the application to the local machine.

RELATED LINKS

For organizational deployments of Chocolatey, please see  
<https://docs.chocolatey.org/en-us/guides/organizations/organizational-deployment-guide>

REMARKS

To see the examples, type: "get-help C:\Foo\Install-Chocolatey.ps1 -examples".  
For more information, type: "get-help C:\Foo\Install-Chocolatey.ps1 -detailed".  
For technical information, type: "get-help C:\Foo\Install-Chocolatey.ps1 -full".  
For online help, type: "get-help C:\Foo\Install-Chocolatey.ps1 -online"

```
PS C:\WINDOWS\system32> # 3. Installing Chocolatey
PS C:\WINDOWS\system32> C:\Foo\Install-Chocolatey.ps1
```

```
Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org)
Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/1.1.0.
Downloading https://community.chocolatey.org/api/v2/package/chocolatey/1.1.0 to
  C:\Users\ADMINI~1\AppData\Local\Temp\2\chocolatey\chocoInstall\chocolatey.zip
Not using proxy.
Extracting C:\Users\ADMINI~1\AppData\Local\Temp\2\chocolatey\chocoInstall\chocolatey.zip to
  C:\Users\ADMINI~1\AppData\Local\Temp\2\chocolatey\chocoInstall
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
  Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
  before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
  (i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
  and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

WARNING: You can safely ignore errors related to missing log files when
  upgrading from a version of Chocolatey less than 0.9.9.
  'Batch file could not be found' is also safe to ignore.
  'The system cannot find the file specified' - also safe.
chocolatey.nupkg file not installed in lib.
  Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
Adding Chocolatey to the profile. This will provide tab completion, refreshenv, etc.
WARNING: Chocolatey profile installed. Reload your profile - type . $profile
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
  first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
```

```
PS C:\WINDOWS\system32> # 4. Configuring Chocolatey
PS C:\WINDOWS\system32> choco feature enable -n allowGlobalConfirmation
Chocolatey v1.1.0
Enabled allowGlobalConfirmation
```

```
PS C:\WINDOWS\system32> # 5. Finding PowerShell (PWSH) on Chocolatey
PS C:\WINDOWS\system32> choco find pwsh
Chocolatey v1.1.0
pwsh 7.2.2 [Approved]
powershell.portable 7.1.3 [Approved] Downloads cached for licensed users
powershell-core 7.2.2 [Approved] Downloads cached for licensed users ←
powershell-preview 7.2.4.20210411 [Approved] Downloads cached for licensed users
4 packages found.
```

```

PS C:\WINDOWS\system32> # 6. Installing PowerShell-7 using choco.exe
PS C:\WINDOWS\system32> choco install powershell-core
Chocolatey v1.1.0
Installing the following packages:
powershell-core
By installing, you accept licenses for the packages.

Progress: Downloading powershell-core 7.2.2... 4%
Progress: Downloading powershell-core 7.2.2... 100%

powershell-core v7.2.2 [Approved]
powershell-core package files install completed. Performing other installation steps.
7.2.2
WARNING: If you started this package under PowerShell core, replacing an in-use version may be unpredictable, require multiple attempts or produce errors.
Downloading powershell-core 64 bit
  from 'https://github.com/PowerShell/PowerShell/releases/download/v7.2.2/PowerShell-7.2.2-win-x64.msi'

Progress: 0% - Saving 13.39 KB of 101.8 MB
Progress: 100% - Completed download of C:\Users\Administrator\AppData\Local\Temp\2\chocolatey\powershell-core\7.2.2\PowerShell-7.2.2-win-x64.msi (101.8 MB).
Download of PowerShell-7.2.2-win-x64.msi (101.8 MB) completed.
Hashes match.
Installing powershell-core...
powershell-core has been installed.
*****
* INSTRUCTIONS: Your system default WINDOWS PowerShell version has not been changed.
* PowerShell CORE 7.2.2, was installed to: "C:\Program Files\PowerShell\7"
* To start PowerShell Core 7.2.2, at a prompt or the start menu execute:
* "pwsh.exe"
* Or start it from the desktop or start menu shortcut installed by this package.
* This is your new default version of PowerShell CORE (pwsh.exe).
*****
* As of OpenSSH 0.0.22.0 Universal Installer, a script is distributed that allows
* setting the default shell for openssh. You could call it with code like this:
* If (Test-Path "C:\Program Files\openssh-win64\Set-SSHDefaultShell.ps1")
* {& "C:\Program Files\openssh-win64\Set-SSHDefaultShell.ps1" [PARAMETERS]}
* Learn more with this:
* Get-Help "C:\Program Files\openssh-win64\Set-SSHDefaultShell.ps1"
* Or here:
* https://github.com/DarwinJS/ChocoPackages/blob/main/openssh/readme.md
*****
powershell-core may be able to be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type 'refreshenv').
The install of powershell-core was successful.
Software installed as 'msi', install location is likely default.

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

Enjoy using Chocolatey? Explore more amazing features to take your
experience to the next level at
https://chocolatey.org/compare

```

```

PS C:\Users\Administrator> # 1. Viewing the PowerShell version
PS C:\Users\Administrator> $PSVersionTable

```

Name	Value
PSVersion	7.2.2
PSEdition	Core
GitCommitId	7.2.2
OS	Microsoft Windows 10.0.20348
Platform	Win32NT
PSCompatibleVersions	{1.0, 2.0, 3.0, 4.0}
PSRemotingProtocolVersion	2.3
SerializationVersion	1.1.0.1
WSManStackVersion	3.0



```

PS C:\Users\Administrator> # 2. Viewing the $Host variable
PS C:\Users\Administrator> $Host

Name           : ConsoleHost
Version        : 7.2.2
InstanceId      : e9582c28-fc21-46fa-8e93-4c1738c9eed3
UI             : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : en-GB
CurrentUICulture : en-US
PrivateData    : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
DebuggerEnabled : True
IsRunspacePushed : False
Runspace       : System.Management.Automation.Runspaces.LocalRunspace

```

```

PS C:\Users\Administrator> # 3. Looking at the PowerShell process (PWSH)
PS C:\Users\Administrator> Get-Process -Id $PID |
    Format-Custom -Property MainModule -Depth 1

class Process
{
    MainModule =
        class ProcessModule
        {
            ModuleName = pwsh.exe
            FileName = C:\Program Files\PowerShell\7\pwsh.exe
            BaseAddress = 140702355292160
            ModuleMemorySize = 303104
            EntryPointAddress = 140702355373392
            FileVersionInfo = File: C:\Program Files\PowerShell\7\pwsh.exe
            InternalName: pwsh.dll
            OriginalFilename: pwsh.dll
            FileVersion: 7.2.2.500
            FileDescription: pwsh
            Product: PowerShell
            ProductVersion: 7.2.2 SHA: 9027d1a433831dcabd8e108f65a893bec63b0c1b
            Debug: False
            Patched: False
            PreRelease: False
            PrivateBuild: False
            SpecialBuild: False
            Language: Language Neutral

            Site =
            Container =
            Size = 296
            Company = Microsoft Corporation
            FileVersion = 7.2.2.500
            ProductVersion = 7.2.2 SHA: 9027d1a433831dcabd8e108f65a893bec63b0c1b
            Description = pwsh
            Product = PowerShell
        }
}

```

```
PS C:\Users\Administrator> # 4. Looking at resource usage statistics
PS C:\Users\Administrator> Get-Process -Id $PID |
    Format-List CPU,*Memory*
```

```
CPU : 2.71875
NonpagedSystemMemorySize64 : 66888
NonpagedSystemMemorySize : 66888
PagedMemorySize64 : 48328704
PagedMemorySize : 48328704
PagedSystemMemorySize64 : 428712
PagedSystemMemorySize : 428712
PeakPagedMemorySize64 : 49778688
PeakPagedMemorySize : 49778688
PeakVirtualMemorySize64 : 2204178198528
PeakVirtualMemorySize : 859975680
PrivateMemorySize64 : 48328704
PrivateMemorySize : 48328704
VirtualMemorySize64 : 2204171730944
VirtualMemorySize : 853508096
```

```
PS C:\Users\Administrator> # 5. Updating the PowerShell 7 help files
PS C:\Users\Administrator> $Before = Get-Help -Name about_*
PS C:\Users\Administrator> Update-Help -Force | Out-Null
Update-Help: Failed to update Help for the module(s) 'ConfigDefenderPerformance, Dism, kds,
NetQos, PcsvDevice, PRI, PSReadline, Whea, WindowsUpdate' with UI culture(s) {en-US} :
One or more errors occurred. (Response status code does not indicate success:
404 (The specified blob does not exist).).
English-US help content is available and can be installed using: Update-Help -UICulture en-US.
PS C:\Users\Administrator> $After = Get-Help -Name about_*
PS C:\Users\Administrator> $Delta = $After.Count - $Before.Count
PS C:\Users\Administrator> "{0} Conceptual Help Files Added" -f $Delta
136 Conceptual Help Files Added
```

```
PS C:\Users\Administrator> # 6. Determining available commands
PS C:\Users\Administrator> Get-Command |
    Group-Object -Property CommandType
```

Count	Name	Group
-----	----	-----
58	Alias	{Add-AppPackage, Add-AppPackageVolume, Add-AppProvisionedPackage,...}
1136	Function	{A:, Add-BCDataCacheExtension, Add-DnsClientDohServerAddress...}
587	Cmdlet	{Add-AppxPackage, Add-AppxProvisionedPackage, Add-AppxVolume...}

```

PS C:\WINDOWS\system32> # 7. Examining the Path Variable
PS C:\WINDOWS\system32> $env:path.split(';')
C:\WINDOWS\system32>
C:\WINDOWS
C:\WINDOWS\system32\Wbem
C:\WINDOWS\system32\WindowsPowerShell\v1.0
C:\WINDOWS\System32\OpenSSH\
C:\ProgramData\chocolatey\bin
C:\Program Files\PowerShell\7\
C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps

```

```

PS C:\Users\Administrator> # 1. Discovering the profile file names
PS C:\Users\Administrator> $ProfileFiles = $PROFILE | Get-Member -MemberType NoteProperty
PS C:\Users\Administrator> $ProfileFiles | Format-Table -Property Name, Definition

Name                                     Definition
----
AllUsersAllHosts                        string AllUsersAllHosts=C:\Program Files\PowerShell\7\profile.ps1
AllUsersCurrentHost                     string AllUsersCurrentHost=C:\Program Files\PowerShell\7\Microsoft.PowerShell_profile.ps1
CurrentUserAllHosts                     string CurrentUserAllHosts=C:\Users\Administrator\Documents\PowerShell\profile.ps1
CurrentUserCurrentHost                   string CurrentUserCurrentHost=C:\Users\Administrator\Documents\PowerShell\Microsoft.Power

```

```

PS C:\Users\Administrator> # 2. Checking for the existence of each PowerShell profile files
PS C:\Users\Administrator> Foreach ($ProfileFile in $ProfileFiles){
    "Testing $($ProfileFile.Name)"
    $ProfilePath = $ProfileFile.Definition.split('=')[1]
    If (Test-Path $ProfilePath){
        "$($ProfileFile.Name) DOES EXIST"
        "At $ProfilePath"
    }
    Else {
        "$($ProfileFile.Name) DOES NOT EXIST"
    }
    ""
}

Testing AllUsersAllHosts
AllUsersAllHosts DOES NOT EXIST

Testing AllUsersCurrentHost
AllUsersCurrentHost DOES NOT EXIST

Testing CurrentUserAllHosts
CurrentUserAllHosts DOES NOT EXIST

CurrentUserCurrentHost DOES NOT EXIST

```

```

PS C:\Users\Administrator> # 3. Discovering Current User/Current Host Profile
PS C:\Users\Administrator> $CUCHProfile = $PROFILE.CurrentUserCurrentHost
PS C:\Users\Administrator> "Current User/Current Host profile path: [$CUCHPROFILE]"
Current User/Current Host profile path: [C:\Users\Administrator\Documents\PowerShell\Microsoft.PowerShell_profile.ps1]

```

```

PS C:\Foo> # 6. Restarting the PowerShell 7 console and viewing the profile output at startup
PS C:\Foo> Get-ChildItem -Path $PROFILE

Directory: C:\Users\Administrator\Documents\PowerShell

Mode                LastWriteTime         Length Name
----                -
-a---             01/04/2022   12:01         1225 Microsoft.PowerShell_profile.ps1

```

```
PS C:\Foo> # 1. Checking the version table for PowerShell 7 console
PS C:\Foo> $PSVersionTable

Name                           Value
----                           -
PSVersion                      7.2.2
PSEdition                      Core
GitCommitId                    7.2.2
OS                              Microsoft Windows 10.0.20348
Platform                       Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0, 5.0, 5.1.10032.0, 6.0.0, 6.1.0, 6.2.0, 7.0.0, 7.1.0, 7.2.2}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0
```

```
PS C:\Foo> # 2. Examining the PowerShell 7 installation folder
PS C:\Foo> Get-ChildItem -Path $env:ProgramFiles\PowerShell\7 -Recurse |
Measure-Object -Property Length -Sum
```

```
Count      : 1013
Average    :
Sum        : 267906359
Maximum    :
Minimum    :
Property   : Length
```

```
PS C:\Foo> # 3. Viewing PowerShell 7 configuration JSON file
PS C:\Foo> Get-ChildItem -Path $env:ProgramFiles\PowerShell\7\powershell*.json |
Get-Content

{
  "Microsoft.PowerShell:ExecutionPolicy": "RemoteSigned",
  "WindowsPowerShellCompatibilityModuleDenylist": [
    "PSScheduledJob",
    "BestPractices",
    "UpdateServices"
  ]
}
```

```
PS C:\Foo> # 4. Checking initial Execution Policy for PowerShell 7
PS C:\Foo> Get-ExecutionPolicy
RemoteSigned
```



```

PS C:\Foo> # 5. Viewing module folders
PS C:\Foo> $1 = 0
PS C:\Foo> $ModPath = $env:PSModulePath -split ';'
PS C:\Foo> $ModPath |
    Foreach-Object {
        [{0:N0}] {1} -f $I++, $_
    }

[0] C:\Users\Administrator\Documents\PowerShell\Modules
[1] C:\Program Files\PowerShell\Modules
[2] c:\program files\powershell\7\Modules
[3] C:\Program Files\WindowsPowerShell\Modules
[4] C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules

```

```

PS C:\Foo> # 6. Checking the modules
PS C:\Foo> $TotalCommands = 0
PS C:\Foo> Foreach ($Path in $ModPath){
    Try { $Modules = Get-ChildItem -Path $Path -Directory -ErrorAction Stop
        "Checking Module Path: [$Path]"
    }
    Catch [System.Management.Automation.ItemNotFoundException] {
        "Module path [$path] DOES NOT EXIST ON $(hostname)"
    }
    $TotalCommands = 0
    Foreach ($Module in $Modules) {
        $Cmds = Get-Command -Module ($Module.name)
        $TotalCommands += $Cmds.Count
    }
}

Module path [C:\Users\Administrator\Documents\PowerShell\Modules] DOES NOT EXIST ON SRV1
Module path [C:\Program Files\PowerShell\Modules] DOES NOT EXIST ON SRV1
Checking Module Path: [c:\program files\powershell\7\Modules]
Checking Module Path: [C:\Program Files\WindowsPowerShell\Modules]
Checking Module Path: [C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules]

```

```

PS C:\Foo> # 7. Viewing totals of commands and modules
PS C:\Foo> $Mods = (Get-Module * -ListAvailable | Measure-Object).count
PS C:\Foo> "{0} modules providing {1} commands" -f $Mods,$TotalCommands
69 modules providing 1562 commands

```

```

PS C:\Foo> # 2. Reviewing the installation help details
PS C:\Foo> Get-Help -Name C:\Foo\Install-VSCode.ps1

NAME
    C:\Foo\Install-VSCode.ps1

SYNOPSIS
    Installs Visual Studio Code, the PowerShell extension, and optionally
    a list of additional extensions.

SYNTAX
    C:\Foo\Install-VSCode.ps1 [[-Architecture] <String>] [[-BuildEdition] <String>] [[-AdditionalExtensions]
    <String[]>] [-LaunchWhenDone] [-EnableContextMenus] [-WhatIf] [-Confirm] [<CommonParameters>]

DESCRIPTION
    This script can be used to easily install Visual Studio Code and the
    PowerShell extension on your machine. You may also specify additional
    extensions to be installed using the -AdditionalExtensions parameter.
    The -LaunchWhenDone parameter will cause VS Code to be launched as
    soon as installation has completed.

    Please contribute improvements to this script on GitHub!

    https://github.com/PowerShell/vscode-powershell/blob/master/scripts/Install-VSCode.ps1

RELATED LINKS

REMARKS
    To see the examples, type: "Get-Help C:\Foo\Install-VSCode.ps1 -Examples"
    For more information, type: "Get-Help C:\Foo\Install-VSCode.ps1 -Detailed"
    For technical information, type: "Get-Help C:\Foo\Install-VSCode.ps1 -Full"

```

```

PS C:\Foo> # 3. Running the installation script and adding in some popular extensions
PS C:\Foo> $Extensions = 'Streetsidesoftware.code-spell-checker',
                        'yzhang.markdown-all-in-one',
                        'hediet.vscode-drawio'

PS C:\Foo> $InstallHT = @{
    BuildEdition      = 'Stable-System'
    AdditionalExtensions = $Extensions
    LaunchWhenDone    = $true
}

PS C:\Foo> .\Install-VSCode.ps1 @InstallHT
Installing extensions...
Installing extension 'ms-vscode.powershell'...
Extension 'ms-vscode.powershell' v2021.12.0 was successfully installed.

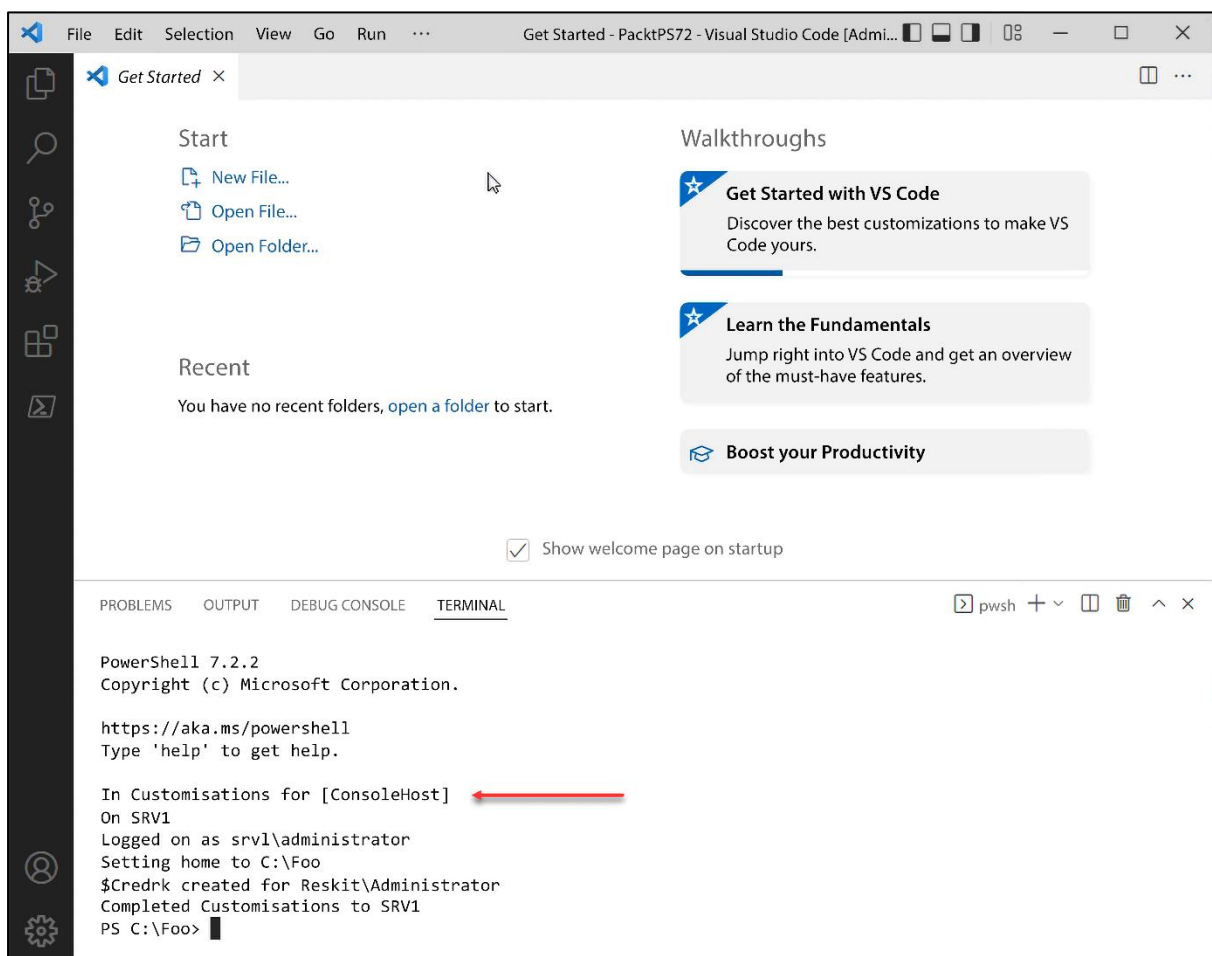
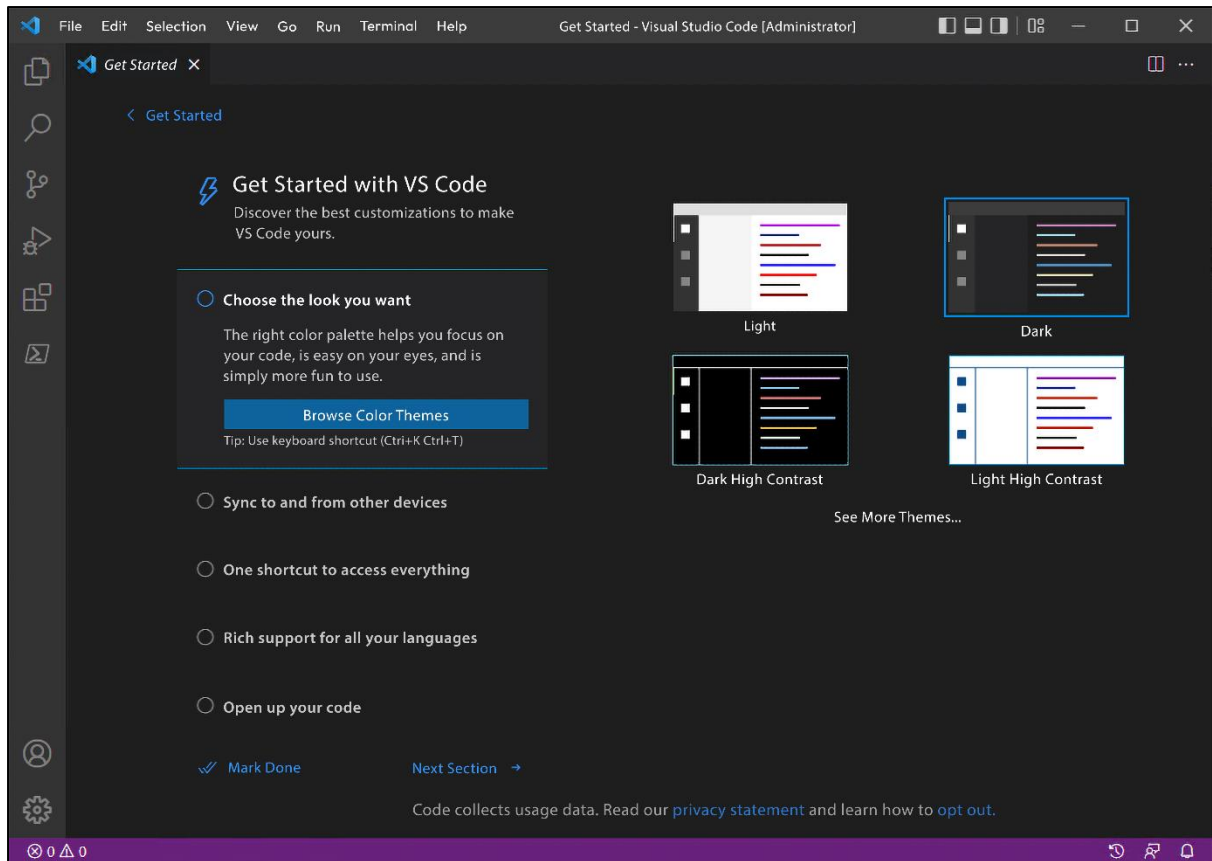
Installing extension Streetsidesoftware.code-spell-checker...
Installing extensions...
Installing extension 'streetsidesoftware.code-spell-checker'...
Extension 'streetsidesoftware.code-spell-checker' v2.1.11 was successfully installed.

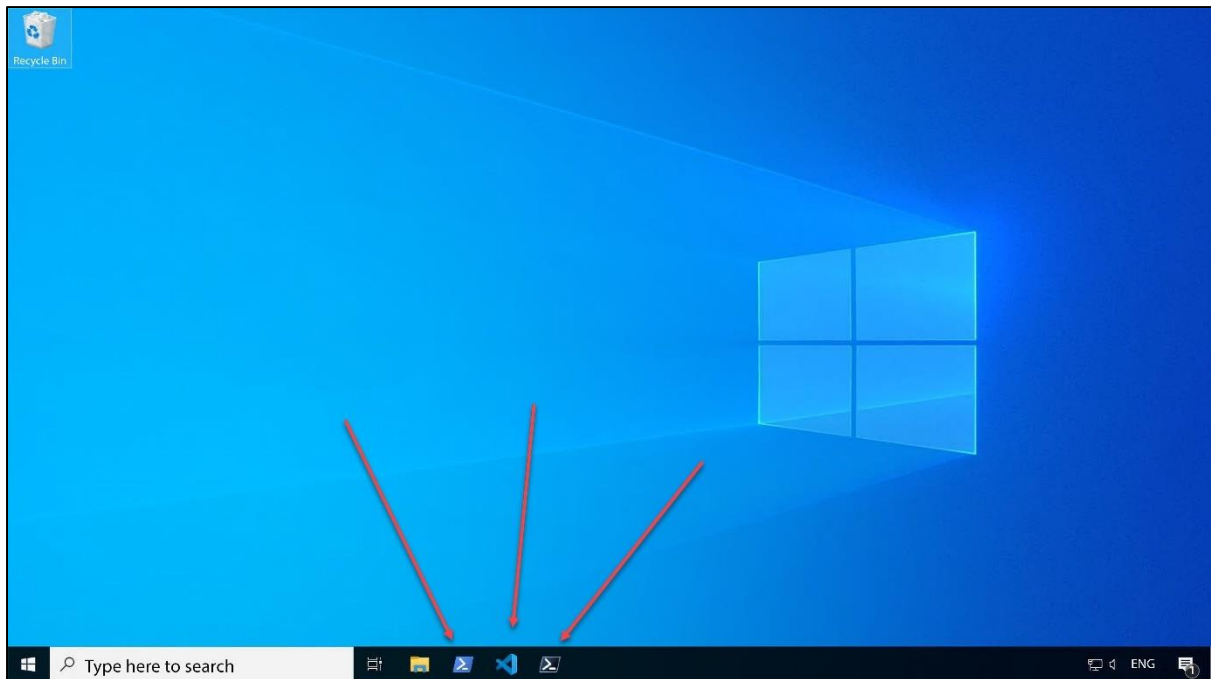
Installing extension yzhang.markdown-all-in-one...
Installing extensions...
Installing extension 'yzhang.markdown-all-in-one'...
Extension 'yzhang.markdown-all-in-one' v3.4.0 was successfully installed.

Installing extension hediet.vscode-drawio...
Installing extensions...
Installing extension 'hediet.vscode-drawio'...
Extension 'hediet.vscode-drawio' v1.6.4 was successfully installed.

Installation complete, starting Visual Studio Code (64-bit)...

```





```
PS C:\Foo> # 1. Getting commands in the PSReadLine module
PS C:\Foo> Get-Command -Module PSReadLine
```

CommandType	Name	Version	Source
Function	PSConsoleHostReadLine	2.1.0	PSReadLine
Cmdlet	Get-PSReadLinekeyHandler	2.1.0	PSReadLine
Cmdlet	Get-PSReadLineOption	2.1.0	PSReadLine
Cmdlet	Remove-PSReadLinekeyHandler	2.1.0	PSReadLine
Cmdlet	Set-PSReadLinekeyHandler	2.1.0	PSReadLine
Cmdlet	Set-PSReadLineOption	2.1.0	PSReadLine

```
PS C:\Foo> # 2. Getting the first 10 PSReadLine key handlers
PS C:\Foo> Get-PSReadLinekeyHandler |
Select-Object -First 10
Sort-Object -Property Key |
Format-Table -Property Key, Function, Description
```

Basic editing functions  
=====

Key	Function	Description
Enter	AcceptLine	Accept the input or move to the next line if input is missing a closing token.
Shift+Enter	AddLine	Move the cursor to the next line without attempting to execute the input
Backspace	BackwardDeleteChar	Delete the character before the cursor
Ctrl+h	BackwardDeleteChar	Delete the character before the cursor
Ctrl+Home	BackwardDeleteLine	Delete text from the cursor to the start of the line
Ctrl+Backspace	BackwardKillWord	Move the text from the start of the current or previous word to the cursor to the kill ring
Ctrl+w	BackwardKillWord	Move the text from the start of the current or previous word to the cursor to the kill ring
Ctrl+c	Copy	Copy selected region to the system clipboard. If no region is selected, copy the whole line
Ctrl+c	CopyOrCancelLine	Either copy selected text to the clipboard, or if no text is selected, cancel editing the line with Cancelline.
Ctrl+x	Cut	Delete selected region placing deleted text in the system clipboard

```
PS C:\Foo> # 3. Discovering a count of unbound key handlers
PS C:\Foo> $Unbound = (Get-PSReadLinekeyHandler -Unbound).count
PS C:\Foo> "$Unbound unbound key handlers"
116 unbound key handlers
```

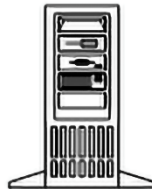
```
PS C:\Foo> # 4. Getting the PSReadline options
PS C:\Foo> Get-PSReadLineOption
```

```
EditMode : Windows
AddToHistoryHandler : System.Func`2[System.String,System.Object]
HistoryNoDuplicates : True
HistorySavePath : C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\
BellStyle : Audible
DingDuration : 50
DingTone : 1221
CommandsToValidateScriptBlockArguments : {ForEach-Object, %, Invoke-Command, icm, Measure-Command, New-Module, nmo,
Register-EngineEvent, Register-ObjectEvent, Register-WMIEvent, Set-PSBreakpoint,
sbp, Start-Job, sajb, Trace-Command, trcm, Use-Transaction, Where-Object, ?, where}

CommandValidationHandler :
CompletionQueryItems : 100
MaximumKillRingCount : 10
ShowToolTips : True
ViModeIndicator : None
WordDelimiters : ;, . [] {} () ^ & * - + ' " ---
AnsiEscapeTimeout : 100
PredictionSource : None
CommandColor : "`e[91m"
CommentColor : "`e[32m"
ContinuationPromptColor : "`e[37m"
DefaultTokenColor : "`e[37m"
EmphasisColor : "`e[33m"
ErrorColor : "`e[91m"
KeywordColor : "`e[92m"
MemberColor : "`e[34m"
NumberColor : "`e[34m"
OperatorColor : "`e[35m"
ParameterColor : "`e[35m"
InlinePredictionColor : "`e[38;5;238m"
SelectionColor : "`e[30;47m"
StringColor : "`e[36m"
TypeColor : "`e[37m"
VariableColor : "`e[33m"
```



## Chapter 2: Managing PowerShell 7 in the Enterprise



**SRV1**

Windows Server 2022

**Initial Reskit Workgroup**

```
PS C:\Foo> # 1. Importing the ServerManager Module
```

```
PS C:\Foo> Import-Module -Name ServerManager
```

WARNING: Module ServerManager is loaded in Windows PowerShell using WinPSCompatSession remoting session; please note that all input and output of commands from this module will be deserialized objects. If you want to load this module into PowerShell please use 'Import-Module -SkipEditionCheck' syntax.

```
PS C:\Foo> # 2. Viewing module details
```

```
PS C:\Foo> Get-Module -Name ServerManager |  
Format-List
```

```
Name           : ServerManager  
Path            : C:\Users\Administrator\AppData\Local\Temp\2\remoteIpMoProxy_ServerManager_2.0.0.0_  
                localhost_c7b5e52a-1429-4fcf-841f-d8800a63ff9f\remoteIpMoProxy_ServerManager_2.0.0.0_  
                localhost_c7b5e52a-1429-4fcf-841f-d8800a63ff9f.psm1  
Description     : Implicit remoting for  
ModuleType      : Script  
Version         : 1.0  
PreRelease      :  
NestedModules   : {}  
ExportedFunctions : {Disable-ServerManagerStandardUserRemoting, Enable-ServerManagerStandardUserRemoting,  
                    Get-WindowsFeature, Install-WindowsFeature, Uninstall-WindowsFeature}  
ExportedCmdlets  :  
ExportedVariables :  
ExportedAliases  : {Add-WindowsFeature, Remove-WindowsFeature}
```

```
PS C:\Foo> # 3. Displaying a Windows Feature
PS C:\Foo> Get-WindowsFeature -Name 'TFTP-Client'
```

Display Name	Name	Install State
	TFTP-Client	Available

```
PS C:\Foo> # 4. Running the same command in a remoting session
PS C:\Foo> $Session = Get-PSSession -Name WinPSCompatSession
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WindowsFeature -Name 'TFTP-Client' |
    Format-Table
}
```

Display Name	Name	Install State
[ ] TFTP Client	TFTP-Client	Available

```
PS C:\Foo> # 5. Getting the path to Windows PowerShell modules
PS C:\Foo> $Paths = $env:PSModulePath -split ';'
PS C:\Foo> $S32Path = $Paths |
    Where-Object {$_.ToString() -match 'system32'}
PS C:\Foo> "System32 path: [$S32Path]"
System32 path: [C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules]
```

```
PS C:\Foo> # 6. Displaying path to the format XML for Server Manager module
PS C:\Foo> $FXML = "$S32path/ServerManager"
PS C:\Foo> $FF = Get-ChildItem -Path $FXML\*.format.ps1xml
PS C:\Foo> "    $($FF.Name)"
PS C:\Foo> "Format XML files:"
Format XML files:
Feature.format.ps1xml
```

```
PS C:\Foo> # 8. Using the command with improved output
PS C:\Foo> Get-WindowsFeature -Name TFTP-Client
```

Display Name	Name	Install State
[ ] TFTP-Client	TFTP-Client	Available



```

PS C:\Foo> # 1. Displaying counts of available PowerShell commands
PS C:\Foo> $CommandsBeforeRSAT = Get-Command
PS C:\Foo> $CmdletsBeforeRSAT = $CommandsBeforeRSAT |
    Where-Object CommandType -eq 'Cmdlet'
PS C:\Foo> $CommandCountBeforeRSAT = $CommandsBeforeRSAT.Count
PS C:\Foo> $CmdletCountBeforeRSAT = $CmdletsBeforeRSAT.Count
PS C:\Foo> "On Host: [$(hostname)]"
PS C:\Foo> "Total Commands available before RSAT installed [$CommandCountBeforeRSAT]"
PS C:\Foo> "Cmdlets available before RSAT installed [$CmdletCountBeforeRSAT]"
On Host: [SRV1]
Commands available before RSAT installed [1809]
Cmdlets available before RSAT installed [597]

```

```

PS C:\Foo> # 2. Getting command types returned by Get-Command
PS C:\Foo> $CommandsBeforeRSAT |
    Group-Object -Property CommandType

```

Count	Name	Group
58	Alias	{Add-AppPackage, Add-AppPackageVolume, Add-AppProvisionedPackage, ...}
1154	Function	{A:, Add-BCDataCacheExtension, Add-DnsClientDohServerAddress, Add...
597	Cmdlet	{Add-AppxPackage, Add-AppxProvisionedPackage, Add-AppxVolume, Add...

```

PS C:\Foo> # 3. Checking the object type details
PS C:\Foo> $CommandsBeforeRSAT |
    Get-Member |
    Select-Object -ExpandProperty TypeName -Unique
System.Management.Automation.AliasInfo
System.Management.Automation.FunctionInfo
System.Management.Automation.CmdletInf

```

```

PS C:\Foo> # 5. Displaying a count of modules available
PS C:\Foo> # before adding the RSAT tools
PS C:\Foo> $CountOfModulesBeforeRSAT = $ModulesBefore.Count
PS C:\Foo> "$CountOfModulesBeforeRSAT modules available"
75 modules available

```

```

PS C:\Foo> # 7. Displaying counts of features installed
PS C:\Foo> "On Host [$(hostname)]"
PS C:\Foo> "Total features available [{0}]" -f $Features.count
PS C:\Foo> "Total features installed [{0}]" -f $FeaturesInstalled.count
PS C:\Foo> "Total RSAT features available [{0}]" -f $RSATFeatures.count
PS C:\Foo> "Total RSAT features installed [{0}]" -f $RSATFeaturesInstalled.count
On Host [SRV1]
Total features available [266]
Total features installed [12]
Total RSAT features available [50]
Total RSAT features installed [0]

```



```
PS C:\Foo> # 8. Adding ALL RSAT tools to SRV1
PS C:\Foo> Get-WindowsFeature -Name *RSAT* |
Install-WindowsFeature
```

Success	Restart Needed	Exit Code	Feature Result
True	Yes	SuccessRestar...	{BitLocker Drive Encryption, RAS Connection ...

WARNING: You must restart this server to finish the installation process.

```
PS C:\Foo> # 10. Displaying counts of commands after installing the RSAT tools
PS C:\Foo> "After Installation of RSAT tools on SRV1"
PS C:\Foo> $INS = 'Features installed on SRV1'
PS C:\Foo> "$($InstalledOnSRV1.Count) $INS"
PS C:\Foo> "$($RsatInstalledOnSRV1.Count) $INS"
After Installation of RSAT tools on SRV1
76 features installed on SRV1 ←
50 RSAT features installed on SRV1 ←
```

```

PS C:\Foo> # 11. Displaying RSAT tools on SRV1
PS C:\Foo> $MODS = "$env:windir\system32\windowspowershell\v1.0\modules"
PS C:\Foo> $SMMOD = "$MODS\ServerManager"
PS C:\Foo> Update-FormatData -PrependPath "$SMMOD\*.format.ps1xml"
PS C:\Foo> Get-WindowsFeature |
    Where-Object Name -Match 'RSAT'

```

Display Name	Name	Install State
[X] Remote Server Administration Tools	RSAT	Installed
[X] Feature Administration Tools	RSAT-Feature-Tools	Installed
[X] SMTP Server Tools	RSAT-SMTP	Installed
[X] BitLocker Drive Encryption Administration ...	RSAT-Feature-Tools-Bit...	Installed
[X] BitLocker Drive Encryption Tools	RSAT-Feature-Tools-Bit...	Installed
[X] BitLocker Recovery Password Viewer	RSAT-Feature-Tools-Bit...	Installed
[X] BITS Server Extensions Tools	RSAT-Bits-Server	Installed
[X] DataCenterBridging LLDP Tools	RSAT-DataCenterBridgin...	Installed
[X] Failover Clustering Tools	RSAT-Clustering	Installed
[X] Failover Cluster Management Tools	RSAT-Clustering-Mgmt	Installed
[X] Failover Cluster Module for Windows Po...	RSAT-Clustering-PowerS...	Installed
[X] Failover Cluster Automation Server	RSAT-Clustering-Automa...	Installed
[X] Failover Cluster Command Interface	RSAT-Clustering-CmdInt...	Installed
[X] Network Load Balancing Tools	RSAT-NLB	Installed
[X] Shielded VM Tools	RSAT-Shielded-VM-Tools	Installed
[X] SNMP Tools	RSAT-SNMP	Installed
[X] Storage Migration Service Tools	RSAT-SMS	Installed
[X] Storage Replica Module for Windows PowerSh...	RSAT-Storage-Replica	Installed
[X] System Insights Module for Windows PowerSh...	RSAT-System-Insights	Installed
[X] WINS Server Tools	RSAT-WINS	Installed
[X] Role Administration Tools	RSAT-Role-Tools	Installed
[X] AD DS and AD LDS Tools	RSAT-AD-Tools	Installed
[X] Active Directory module for Windows Po...	RSAT-AD-PowerShell	Installed
[X] AD DS Tools	RSAT-ADDS	Installed
[X] Active Directory Administrative Ce...	RSAT-AD-AdminCenter	Installed
[X] AD DS Snap-Ins and Command-Line To...	RSAT-ADDS-Tools	Installed
[X] AD LDS Snap-Ins and Command-Line Tools	RSAT-ADLDS	Installed
[X] Hyper-V Management Tools	RSAT-Hyper-V-Tools	Installed
[X] Remote Desktop Services Tools	RSAT-RDS-Tools	Installed
[X] Remote Desktop Gateway Tools	RSAT-RDS-Gateway	Installed
[X] Remote Desktop Licensing Diagnoser Too...	RSAT-RDS-Licensing-Dia...	Installed
[X] Windows Server Update Services Tools	UpdateServices-RSAT	Installed
[X] Active Directory Certificate Services Tools	RSAT-ADCS	Installed
[X] Certification Authority Management Too...	RSAT-ADCS-Mgmt	Installed
[X] Online Responder Tools	RSAT-Online-Responder	Installed
[X] Active Directory Rights Management Service...	RSAT-ADRMS	Installed
[X] DHCP Server Tools	RSAT-DHCP	Installed
[X] DNS Server Tools	RSAT-DNS-Server	Installed
[X] Fax Server Tools	RSAT-Fax	Installed
[X] File Services Tools	RSAT-File-Services	Installed
[X] DFS Management Tools	RSAT-DFS-Mgmt-Con	Installed
[X] File Server Resource Manager Tools	RSAT-FSRM-Mgmt	Installed
[X] Services for Network File System Manag...	RSAT-NFS-Admin	Installed
[X] Network Controller Management Tools	RSAT-NetworkController	Installed
[X] Network Policy and Access Services Tools	RSAT-NPAS	Installed
[X] Print and Document Services Tools	RSAT-Print-Services	Installed
[X] Remote Access Management Tools	RSAT-RemoteAccess	Installed
[X] Remote Access GUI and Command-Line Too...	RSAT-RemoteAccess-Mgmt	Installed
[X] Remote Access module for Windows Power...	RSAT-RemoteAccess-Powe...	Installed
[X] Volume Activation Tools	RSAT-VA-Tools	Installed



```
PS C:\Foo> # 1. Reviewing the cmdlets in the PackageManagement module
PS C:\Foo> Get-Command -Module PackageManagement
```

CommandType	Name	Version	Source
-----	----	-----	-----
Cmdlet	Find-Package	1.4.7	PackageManagement
Cmdlet	Find-PackageProvider	1.4.7	PackageManagement
Cmdlet	Get-Package	1.4.7	PackageManagement
Cmdlet	Get-PackageProvider	1.4.7	PackageManagement
Cmdlet	Get-PackageSource	1.4.7	PackageManagement
Cmdlet	Import-PackageProvider	1.4.7	PackageManagement
Cmdlet	Install-Package	1.4.7	PackageManagement
Cmdlet	Install-PackageProvider	1.4.7	PackageManagement
Cmdlet	Register-PackageSource	1.4.7	PackageManagement
Cmdlet	Save-Package	1.4.7	PackageManagement
Cmdlet	Set-PackageSource	1.4.7	PackageManagement
Cmdlet	Uninstall-Package	1.4.7	PackageManagement
Cmdlet	Unregister-PackageSource	1.4.7	PackageManagement

```
PS C:\Foo> # 2. Reviewing installed providers with Get-PackageProvider
PS C:\Foo> Get-PackageProvider |
```

```
Format-Table -Property Name,
               Version,
               SupportedFileExtensions,
               FromTrustedSource
```

Name	Version	SupportedFileExtensions	FromTrustedSource
-----	-----	-----	-----
NuGet	3.0.0.1	{nupkg}	False
PowerShellGet	2.2.5.0	{}	False

```
PS C:\Foo> # 3. Examining available Package Providers
PS C:\Foo> $PROVIDERS = Find-PackageProvider
PS C:\Foo> $PROVIDERS |
    Select-Object -Property Name,Summary |
    Format-Table -AutoSize -Wrap
```

Name	Summary
PowerShellGet	PowerShell module with commands for discovering, installing, updating and publishing the PowerShell artifacts like Modules, DSC Resources, Role Capabilities and Scripts.
ChocolateyGet	Package Management (OneGet) provider that facilitates installing Chocolatey packages from any NuGet repository.
ContainerImage	This is a PackageManagement provider module which helps in discovering, downloading and installing Windows Container OS images. For more details and examples refer to our project site at <a href="https://github.com/PowerShell/ContainerProvider">https://github.com/PowerShell/ContainerProvider</a> .
NanoServerPackage	A PackageManagement provider to Discover, Save and Install Nano Server Packages on-demand
Chocolatier	Package Management (OneGet) provider that facilitates installing Chocolatey packages from any NuGet repository.
WinGet	Package Management (OneGet) provider that facilitates installing WinGet packages from any NuGet repository.
DotNetGlobalToolProvider	OneGet package provider for dotnet global tools.

```
PS C:\Foo> # 4. Discovering and counting available packages
PS C:\Foo> $PACKAGES = Find-Package
PS C:\Foo> "Discovered {0:N0} packages" -f $PACKAGES.Count
```

Discovered 7,880 packages

```
PS C:\Foo> # 5. Showing the first 5 packages discovered
PS C:\Foo> $PACKAGES |
    Select-Object -First 5 |
    Format-Table -AutoSize -Wrap
```

Name	Version	Source	Summary
SpeculationControl	1.0.14	PSGallery	This module provides the ability to query the speculation control settings for the system.
DellBIOSProvider	2.6.0	PSGallery	The 'Dell Command   PowerShell Provider' provides native configuration capability of Dell Optiplex, Latitude, Precision, XPS Notebook and Venue 11 systems within PowerShell.
PSWindowsUpdate	2.2.0.3	PSGallery	This module contain cmdlets to manage Windows Update Client.
NetworkingDsc	8.2.0	PSGallery	DSC resources for configuring settings related to networking.
PackageManagement	1.4.7	PSGallery	PackageManagement (a.k.a. OneGet) is a new way to discover and install software packages from around the web. It is a manager or multiplexor of existing package managers (also called package providers) that unifies Windows package management with a single Windows PowerShell interface. With PackageManagement, you can do the following. - Manage a list of software repositories in which packages can be searched, acquired and installed - Discover software packages - Seamlessly install, uninstall, and inventory packages from one or more software repositories



```

PS C:\Foo> # 7. Verifying ChocolateyGet is in the list of installed providers
PS C:\Foo> Import-PackageProvider -Name ChocolateyGet
PS C:\Foo> Get-PackageProvider -ListAvailable |
    Select-Object -Property Name,Version

```

Name	Version
ChocolateyGet	4.0.0.0
NuGet	3.0.0.1
PowerShellGet	2.2.5.0
PowerShellGet	1.0.0.1

```

PS C:\Foo> # 8. Discovering Packages using the ChocolateyGet provider
PS C:\Foo> $CPackages = Find-Package -ProviderName ChocolateyGet -Name *
PS C:\Foo> "($CPackages.Count) packages available via ChocolateyGet"
6842 packages available via ChocolateyGet

```

```

PS C:\Foo> # 1. Reviewing the commands available in the PowerShellGet module
PS C:\Foo> Get-Command -Module PowerShellGet

```

CommandType	Name	Version	Source
Function	Find-Command	2.2.5	PowerShellGet
Function	Find-DscResource	2.2.5	PowerShellGet
Function	Find-Module	2.2.5	PowerShellGet
Function	Find-RoleCapability	2.2.5	PowerShellGet
Function	Find-Script	2.2.5	PowerShellGet
Function	Get-CredsFromCredentialProvider	2.2.5	PowerShellGet
Function	Get-InstalledModule	2.2.5	PowerShellGet
Function	Get-InstalledScript	2.2.5	PowerShellGet
Function	Get-PSRepository	2.2.5	PowerShellGet
Function	Install-Module	2.2.5	PowerShellGet
Function	Install-Script	2.2.5	PowerShellGet
Function	New-ScriptFileInfo	2.2.5	PowerShellGet
Function	Publish-Module	2.2.5	PowerShellGet
Function	Publish-Script	2.2.5	PowerShellGet
Function	Register-PSRepository	2.2.5	PowerShellGet
Function	Save-Module	2.2.5	PowerShellGet
Function	Save-Script	2.2.5	PowerShellGet
Function	Set-PSRepository	2.2.5	PowerShellGet
Function	Test-ScriptFileInfo	2.2.5	PowerShellGet
Function	Uninstall-Module	2.2.5	PowerShellGet
Function	Uninstall-Script	2.2.5	PowerShellGet
Function	Unregister-PSRepository	2.2.5	PowerShellGet
Function	Update-Module	2.2.5	PowerShellGet
Function	Update-ModuleManifest	2.2.5	PowerShellGet
Function	Update-Script	2.2.5	PowerShellGet
Function	Update-ScriptFileInfo	2.2.5	PowerShellGet

```
PS C:\Foo> # 2. Discovering Find-* cmdlets in PowerShellGet module
PS C:\Foo> Get-Command -Module PowerShellGet -Verb Find
```

CommandType	Name	Version	Source
Function	Find-Command	2.2.5	PowerShellGet
Function	Find-DscResource	2.2.5	PowerShellGet
Function	Find-Module	2.2.5	PowerShellGet
Function	Find-RoleCapability	2.2.5	PowerShellGet
Function	Find-Script	2.2.5	PowerShellGet

```
PS C:\Foo> # 4. Reporting on results
PS C:\Foo> "On Host [$(hostname)]"
PS C:\Foo> "Commands found:           [{0:N0}]" -f $Commands.Count
PS C:\Foo> "Modules found:             [{0:N0}]" -f $Modules.Count
PS C:\Foo> "DSC Resources found:        [{0:N0}]" -f $DSCResources.Count
PS C:\Foo> "Scripts found:              [{0:N0}]" -f $Scripts.Count
```

```
On Host [SRV1]
Commands found:           [146,638]
Modules found:           [7,886]
DSC Resources found:     [2,146]
Scripts found:           [1,545]
```

```
PS C:\Foo> # 5. Discovering NTFS-related modules
PS C:\Foo> $Modules |
Where-Object Name -match NTFS
```

Version	Name	Repository	Description
4.2.6	NTFSSecurity	PSGallery	Windows PowerShell M...
1.4.1	cNtfsAccessControl	PSGallery	The cNtfsAccessContr...
1.0	NTFSPermissionMigration	PSGallery	This module is used ...



```
PS C:\Foo> # 7. Reviewing module contents
PS C:\Foo> Get-Command -Module NTFSSecurity
```

CommandType	Name	Version	Source
-----	----	-----	-----
Cmdlet	Add-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Add-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Clear-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Clear-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Copy-Item2	4.2.6	NTFSSecurity
Cmdlet	Disable-NTFSAccessInheritance	4.2.6	NTFSSecurity
Cmdlet	Disable-NTFSAuditInheritance	4.2.6	NTFSSecurity
Cmdlet	Disable-Privileges	4.2.6	NTFSSecurity
Cmdlet	Enable-NTFSAccessInheritance	4.2.6	NTFSSecurity
Cmdlet	Enable-NTFSAuditInheritance	4.2.6	NTFSSecurity
Cmdlet	Enable-Privileges	4.2.6	NTFSSecurity
Cmdlet	Get-ChildItem2	4.2.6	NTFSSecurity
Cmdlet	Get-DiskSpace	4.2.6	NTFSSecurity
Cmdlet	Get-FileHash2	4.2.6	NTFSSecurity
Cmdlet	Get-Item2	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSEffectiveAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSHardLink	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSInheritance	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOrphanedAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOrphanedAudit	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOwner	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSSecurityDescriptor	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSSimpleAccess	4.2.6	NTFSSecurity
Cmdlet	Get-Privileges	4.2.6	NTFSSecurity
Cmdlet	Move-Item2	4.2.6	NTFSSecurity
Cmdlet	New-NTFSHardLink	4.2.6	NTFSSecurity
Cmdlet	New-NTFSSymbolicLink	4.2.6	NTFSSecurity
Cmdlet	Remove-Item2	4.2.6	NTFSSecurity
Cmdlet	Remove-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Remove-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSInheritance	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSOwner	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSSecurityDescriptor	4.2.6	NTFSSecurity
Cmdlet	Test-Path2	4.2.6	NTFSSecurity

```
S C:\Foo> # 8. Testing the Get-NTFSAccess cmdlet
PS C:\Foo> Get-NTFSAccess -Path C:\Foo
```

Path: C:\Foo (Inheritance enabled)

Account	Access Rights	Applies to	Type	IsInherited	InheritedFrom
NT AUTHORITY\SYSTEM	FullControl	ThisFolderSubfoldersAndF...	Allow	True	C:
BUILTIN\Administrators	FullControl	ThisFolderSubfoldersAndF...	Allow	True	C:
BUILTIN\Users	ReadAndExecute, ...	ThisFolderSubfoldersAndF...	Allow	True	C:
BUILTIN\Users	CreateDirectories	ThisFolderAndSubfolders	Allow	True	C:
BUILTIN\Users	CreateFiles	ThisFolderAndSubfolders	Allow	True	C:
CREATOR OWNER	GenericAll	SubfoldersAndFilesOnly	Allow	True	C:

```
PS C:\Foo> # 11. Viewing the contents of the download folder
PS C:\Foo> Get-ChildItem -Path $DownloadFolder -Recurse |
Format-Table -Property FullName
```

FullName

```
-----
C:\Foo\DownloadedModules\PSLogging
C:\Foo\DownloadedModules\PSLogging\2.5.2
C:\Foo\DownloadedModules\PSLogging\2.5.2\PSLogging.psd1
C:\Foo\DownloadedModules\PSLogging\2.5.2\PSLogging.psm1
```

```
PS C:\Foo> # 12. Checking commands in the module
PS C:\Foo> Get-Command -Module PSLogging
```

CommandType	Name	Version	Source
Function	Send-Log	0.0	PSLogging
Function	Start-Log	0.0	PSLogging
Function	Stop-Log	0.0	PSLogging
Function	Write-LogError	0.0	PSLogging
Function	Write-LogInfo	0.0	PSLogging
Function	Write-LogWarning	0.0	PSLogging



```

PS C:\Foo> # 2. Sharing the folder
PS C:\Foo> $SMBHT = @{
    Name           = 'RKRepo'
    Path           = $PATH
    Description    = 'Reskit Repository'
    FullAccess     = 'Everyone'
}
PS C:\Foo> New-SmbShare @SMBHT

```

Name	ScopeName	Path	Description
RKRepo	*	C:\RKRepo	Reskit Repository

```

PS C:\Foo> # 4. Viewing configured repositories
PS C:\Foo> Get-PSRepository

```

Name	InstallationPolicy	SourceLocation
RKRepo	Trusted	\\SRV1\RKRepo
PSGallery	Untrusted	https://www.powershellgallery.com/api/v2

```

PS C:\Foo> # 7. Testing the module locally
PS C:\Foo> Import-Module -Name $HWDIR\HW.PSM1 -Verbose
VERBOSE: Importing function 'Get-HelloWorld'.
VERBOSE: Importing alias 'GHW'.
PS C:\Foo> GHW
Hello World

```

```

PS C:\Foo> # 10. Viewing the results of publishing
PS C:\Foo> Find-Module -Repository RKRepo

```

Version	Name	Repository	Description
1.0.1	HW	RKRepo	Hello World module

```
PS C:\Foo> # 11. Checking the repository's home folder
PS C:\Foo> Get-ChildItem -Path $LPATH
```

Directory: C:\RKRepo

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a---	25/04/2022 17:38	3462	HW.1.0.1.nupkg

```
PS C:\Foo> # 2. Displaying the newly created certificate
PS C:\Foo> $Cert = Get-ChildItem -Path Cert:\CurrentUser\my -CodeSigningCert
PS C:\Foo> $Cert |
    Where-Object {$_.SubjectName.Name -match $CHT.Subject}
```

PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\my

Thumbprint	Subject	EnhancedKeyUsageList
-----	-----	-----
55694D1AA117028A739E9F5CC7AE BBB1209D45E5	CN=Reskit Code Sign...	Code Signing

```
PS C:\Foo> # 3. Creating and viewing a simple script
PS C:\Foo> $Script = @"
    # Sample Script
    'Hello World from PowerShell 7!'
    "Running on [$(Hostname)]"
"@
PS C:\Foo> $Script | Out-File -FilePath C:\Foo\Signed.ps1
PS C:\Foo> Get-ChildItem -Path C:\Foo\Signed.ps1
```

Directory: C:\Foo

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a---	26/04/2022 17:22	78	Signed.ps1

```
PS C:\Foo> # 4. Signing your new script
PS C:\Foo> $SHT = @{
    Certificate = $cert
    FilePath   = 'C:\Foo\Signed.ps1'
}
PS C:\Foo> Set-AuthenticodeSignature @SHT
```

Directory: C:\foo

SignerCertificate	Status	StatusMessage	Path
55694D1AA117028A739E9F5CC7AEBB1209D45E5	UnknownError	A certificate chain processed, but terminated in a root certificate which is not trusted by the trust provider.	signed.ps1

```
PS C:\Foo> # 5. Checking script after signing
PS C:\Foo> Get-ChildItem -Path C:\Foo\Signed.ps1
```

Directory: C:\Foo

Mode	LastWriteTime	Length	Name
-a---	26/04/2022 17:25	2136	Signed.ps1



```
PS C:\Foo> # 6. Viewing the signed script
PS C:\Foo> Get-Content -Path C:\Foo\Signed.ps1
```

```
# Sample Script
'Hello World from PowerShell 7!'
"Running on [SRV1]"
```

```
# SIG # Begin signature block
# MIIFeQYJKoZIhvcNAQcCoIIFajCCBWYCAQExCzAJBgUrDgMCGGUAMGkGCisGAQQB
# gjcCAQSGWzBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUiXIZ7que8HGeN6y3v9batOmI
# VzygggMQMIIDDDCCAfSgAwIBAgIQESy8ARMRF6RNfapBk//p+zANBgkqhkiG9w0B
# AQsFADAeMRwwGgYDVQQDDDBNSZXNraXQgQ29kZSBTaWduaW5nMB4XDTEyMDQyNjE2
# MTEyM1oXDTEyMDQyNjE2MzExM1owHjEcmBoGA1UEAwVUMVza2l0IENvZGUgU2ln
# bmluZzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMQkI0Zo8ctEiPwd
# R5LdCw7QQ0WVK6gsqAvgX0pMVYe9V70Y+h3jyomxA3svRAKJVriyfsp+uEaCN5WS
# aT863u553qMBYkgvokJzmQIxWJV01B1dBbSWutn9wihSjzaqUm3KWh/k8XS8ow9W
# KvWChT7Vn5fmsfX/6tXtmJxaG47DirWZUZ/9V4RJADQL24RT5SsE16WboCKanRAL
# can60JK80CbIQvM0uXne+7bv1n+3x+ZfBptJFB8Sf0CfGiUYe/q3P1rWZS+do1hd
# bPFRxca44SU4zxLSnllvIXeQgCTPe93NHPOfkgDbliZhtbjFSF03S0dxj7UmOE1/
# RQcLy00CAwEAAaNGMEQwDgYDVR0PAQH/BAQDAgeAMBGA1UdJQQMMAoGCCsGAQUF
# BwMDMB0GA1UdDgQWBBSbIu3CywcdPmTZRkdV647Q6XrXjANBgkqhkiG9w0BAQsF
# AAOCAQEApHLEvSTYsxdBYWir+/3ShQ05454IfIIgAtNar9RTGmdobpeMdGTWIGLS
# mXRUVnTGIWNPqEdd3m+Pm9hUuwW4Av/F0LMwIE0WsqvS5MoepKY6oinpaa4rUDZi
# XbLBTdutNTa4u7YExjVxYgOUXmdbgG0L/OjiWmqQ7BC40zmJr1LrJ+NfQkMDf9LR
# VjkrLX1MD/V6ZqZtJK57XjmX1VmiZeS0pNNpvvSIGE70/N37Bf9iBmg2HkLz/cm0
# Aqo6IrIHmK3UuG1VEjCPh7479KqWSEcqqGaVWTBuuA4xhfl5mjJXaK17MqCYD6bZ
# BpTrBLs1G3ULSeCqzVhSXPGGnSQ5AzGCAdMwggHPAgEBMDIwHjEcmBoGA1UEAwVU
# MVza2l0IENvZGUgU2lnbmluZwIQESy8ARMRF6RNfapBk//p+zAJBgUrDgMCGGU
# oHgwGAYKKwYBBAGCNwIBDDEKMAigAoAAoQKAADAZBgkqhkiG9w0BCQMxDAYKKwYB
# BAGCNwIBBDACBgorBgEEAYI3AgELMQ4wDAYKKwYBBAGCNwIBFTAjBgkqhkiG9w0B
# CQQxFgQUtXvYPUYhpvHmSjkz4NMRIU+3M6QwDQYJKoZIhvcNAQEBBQAEggEAAo7K
# r2w2oNathBRiZ4mohR44DqMFgg5dfkbW9+wlglQ5C2opNSWFPbKJWJJWRHv7r21P
# 0t0myr/TGo1MQ093afZPJvpeDP1pA/Gc+VNQJwk3YG6yl+HnhthaxuyNzHCZqDdX
# ydzT05VazDdHIOU63ZN8RD17JZQ0LXHiPxSbpkoIvAkQJ5NPK6X/Rbi8Y0u69/U
# Q71jptE0I9HYL8htjns0Mp41DHJp0pKPsL5JmNyilj6xj/vad0RrrPvrHhKbXWOP
# TAwvZro2J6BK0zSuchNn0X2AYpGdnvkVOYnCEa60nQg1EXBvkYH8M4PoY457ULCy
# qN+2J0RqrkyF1AF85Q==
# SIG # End signature block
```

```
PS C:\Foo> # 7. Testing the signature
PS C:\Foo> Get-AuthenticodeSignature -FilePath C:\Foo\Signed.ps1 |
Format-List
```

```
SignerCertificate      : [Subject]
                        CN=Reskit Code Signing

                        [Issuer]
                        CN=Reskit Code Signing

                        [Serial Number]
                        112CBC01131117A44D7DAA4193FFE9FB

                        [Not Before]
                        26/04/2022 17:11:13

                        [Not After]
                        26/04/2023 17:31:13

                        [Thumbprint]
                        55694D1AA117028A739E9F5CC7AEBBB1209D45E5

TimeStamperCertificate :
Status                 : UnknownError
StatusMessage          : A certificate chain processed, but terminated in
                        a root certificate which is not trusted by the
                        trust provider.

Path                   : C:\Foo\Signed.ps1
SignatureType          : Authenticode
IsOSBinary             : False
```

```
PS C:\Foo> # 8. Running the signed script
PS C:\Foo> C:\Foo\Signed.ps1
Hello World from PowerShell 7!
Running on [SRV1]
```

```
PS C:\Foo> # 10. Running the signed script
PS C:\Foo> C:\Foo\Signed.ps1
C:\Foo\Signed.ps1
Line |
  2  | C:\Foo\Signed.ps1
      | ~~~~~
      | File C:\Foo\Signed.ps1 cannot be loaded. A certificate chain processed, but
      | terminated in a root certificate which is not trusted by the trust provider.
```

## Security Warning



You are about to install a certificate from a certification authority (CA) claiming to represent:

**Reskit Code Signing**

Windows cannot validate that the certificate is actually from "Reskit Code Signing". You should confirm its origin by contacting "Reskit Code Signing". The following number will assist you in this process:

Thumbprint (sha1): 55694D1A A117028A 739E9F5C C7AE BBB1  
209D45E5

**Warning:**

If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

Yes

No



```
PS C:\Foo> # 12. Checking the signature
PS C:\Foo> Get-AuthenticodeSignature -FilePath C:\Foo\Signed.ps1 |
Format-List
```

```
SignerCertificate      : [Subject]
                        CN=Reskit Code Signing

                        [Issuer]
                        CN=Reskit Code Signing

                        [Serial Number]
                        112CBC01131117A44D7DAA4193FFE9FB

                        [Not Before]
                        26/04/2022 17:11:13

                        [Not After]
                        26/04/2023 17:31:13

                        [Thumbprint]
                        55694D1AA117028A739E9F5CC7AEBBB1209D45E5

TimeStampCertificate :
Status               : Valid
StatusMessage        : Signature verified.
Path                 : C:\Foo\Signed.ps1
SignatureType        : Authenticode
IsOSBinary           : False
```

```
PS C:\Foo> # 13. Running the signed script
PS C:\Foo> C:\Foo>.\Foo\Signed.ps1
Do you want to run software from this untrusted publisher? ←
File C:\Foo\Signed.ps1 is published by CN=Reskit Code Signing and is
not trusted on your system. Only run scripts from trusted publishers.
[V] Never run [D] Do not run [R] Run once [A] Always run [?] Help
(default is "Do not run"):
```

```
PS C:\Foo> # 15. Running the signed script
PS C:\Foo> C:\Foo\Signed.ps1
Hello World from PowerShell 7!
Running on [SRV1]
```

```
PS C:\Foo> # 1. Finding the PSShortCut module
PS C:\Foo> Find-Module -Name '*Shortcut'
```

Version	Name	Repository	Description
2.2.0	DSCR_Shortcut	PSGallery	PowerShell DSC Resource to create shortcut file.
0.2.1	PSAdvancedShortcut	PSGallery	Advanced shortcut appliance to create and modify shortcut file properties that are not easily...
2.2.0	Remove-iCloudPhotosShortcut	PSGallery	A PowerShell module for removing iCloud Photos shortcuts from This PC and Quick access on Win...
1.0.6	PSShortcut	PSGallery	This module eases working with Windows shortcuts (LNK and URL) files.
0.1.0	oneShortcut	PSGallery	PowerShell module to query, create, and remove OneDrive shortcuts to SharePoint document libr...

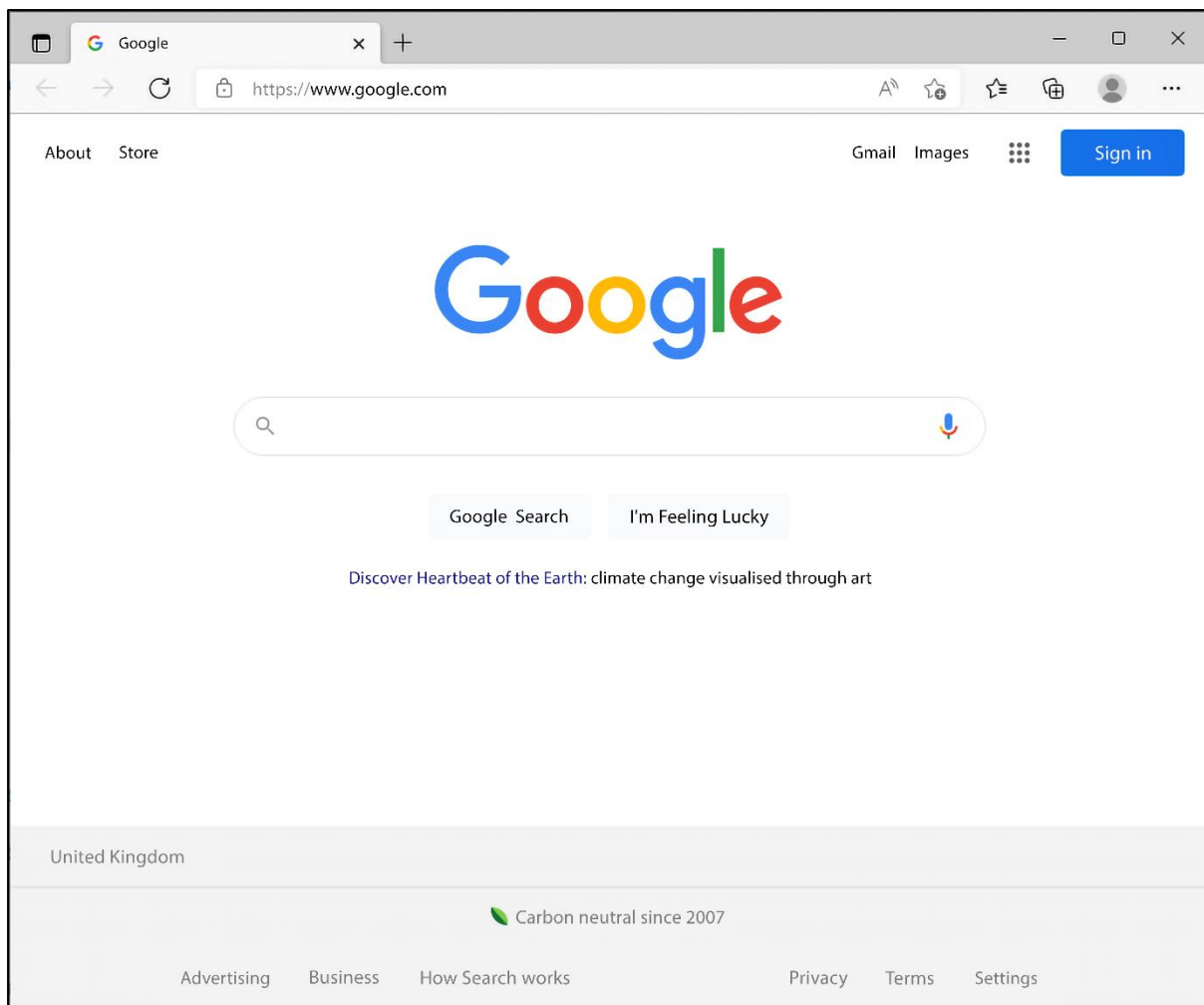
```
PS C:\Foo> # 3. Reviewing PSShortcut module
PS C:\Foo> Get-Module -Name PSShortCut -ListAvailable |
Format-List
```

```
Name           : PSShortcut
Path            : C:\Users\Administrator\Documents\PowerShell\Modules\PSShortcut\1.0.6\PSShortcut.psd1
Description     : This module eases working with Windows shortcuts (LNK and URL) files.
ModuleType      : Script
Version        : 1.0.6
PreRelease     :
NestedModules   : {}
ExportedFunctions : {Get-Shortcut, Set-Shortcut}
ExportedCmdlets  :
ExportedVariables :
ExportedAliases  :
```

```
PS C:\Foo> # 4. Discovering commands in PSShortcut module
PS C:\Foo> Get-Command -Module PSShortcut
```

CommandType	Name	Version	Source
Function	Get-Shortcut	1.0.6	PSShortcut
Function	Set-Shortcut	1.0.6	PSShortcut





```
PS C:\Foo> # 6. Discovering PWSH shortcuts
PS C:\Foo> $SHORTCUTS | Where-Object Name -match '^PWSH'
```

Directory: C:\Users\Administrator\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar

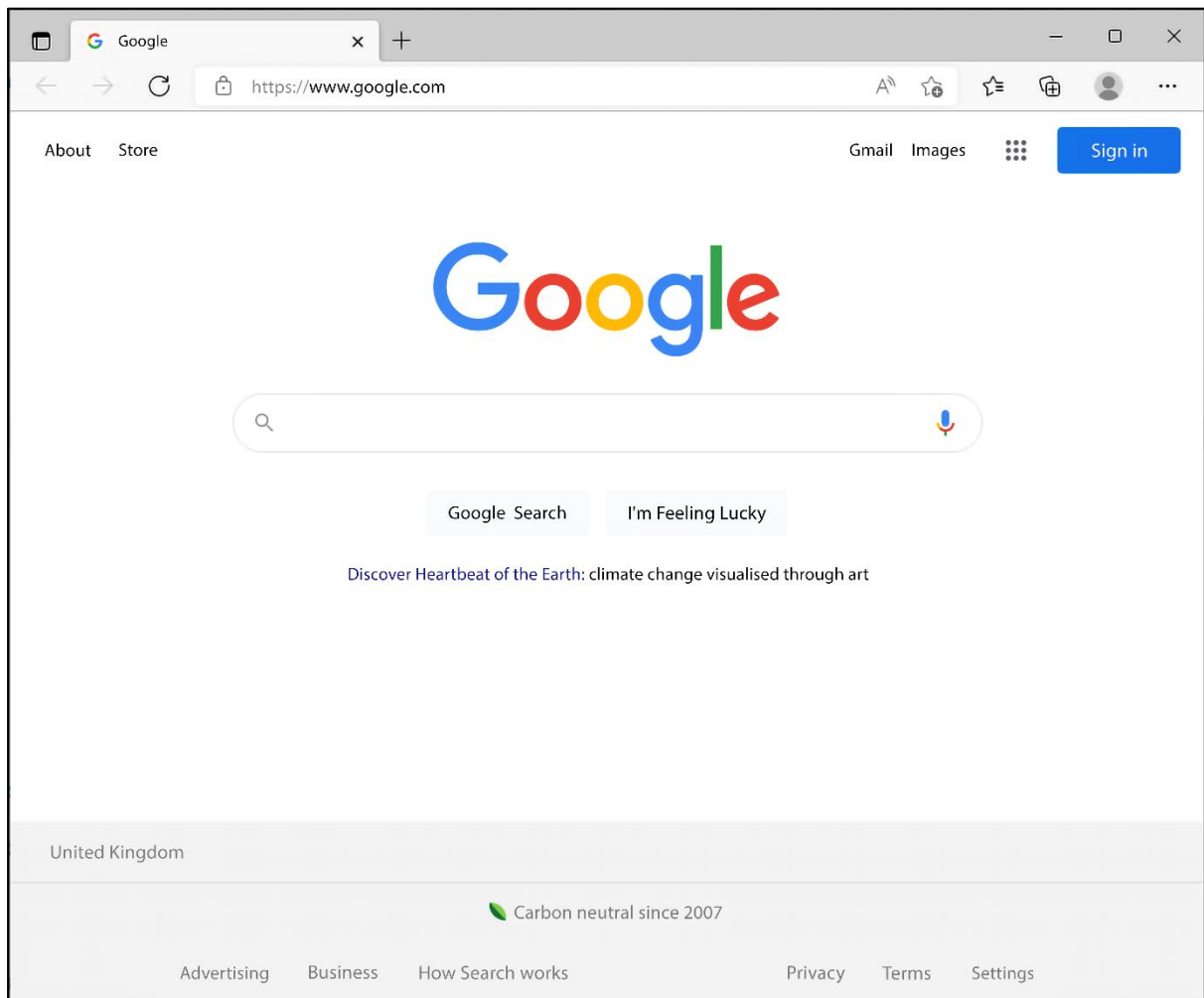
Mode	LastWriteTime	Length	Name
-a---	09/04/2022 16:11	1030	pwsh.lnk

```
PS C:\Foo> # 7. Discovering URL shortcut
PS C:\Foo> $URLSC = Get-Shortcut -FilePath *.url
PS C:\Foo> $URLSC
```

Directory: C:\Users\Administrator\Favorites

Mode	LastWriteTime	Length	Name
-a---	31/03/2022 17:30	208	Bing.url

```
PS C:\Foo> # 8. Viewing content of shortcut
PS C:\Foo> $URLSC | Get-Content
[{000214A0-0000-0000-C000-000000000046}]
Prop3=19,2
[InternetShortcut]
IDList=
URL=http://go.microsoft.com/fwlink/p/?LinkId=255142
IconIndex=0
IconFile=%ProgramFiles%\Internet Explorer\Images\bing.ico
```





```
PS C:\Foo> # 1. Getting archive module
PS C:\Foo> Get-Module -Name Microsoft.PowerShell.Archive -ListAvailable
```

Directory: C:\program files\powershell\7\Modules

ModuleType	Version	PreRelease	Name	PSEdition	ExportedCommands
Manifest	1.2.5		Microsoft.PowerShell.Archive	Desk	{Compress-Archive, Expand-Archive}

```
PS C:\Foo> # 2. Discovering commands in archive module
PS C:\Foo> Get-Command -Module Microsoft.PowerShell.Archive
```

CommandType	Name	Version	Source
Function	Compress-Archive	1.2.5	Microsoft.PowerShell.Archive
Function	Expand-Archive	1.2.5	Microsoft.PowerShell.Archive

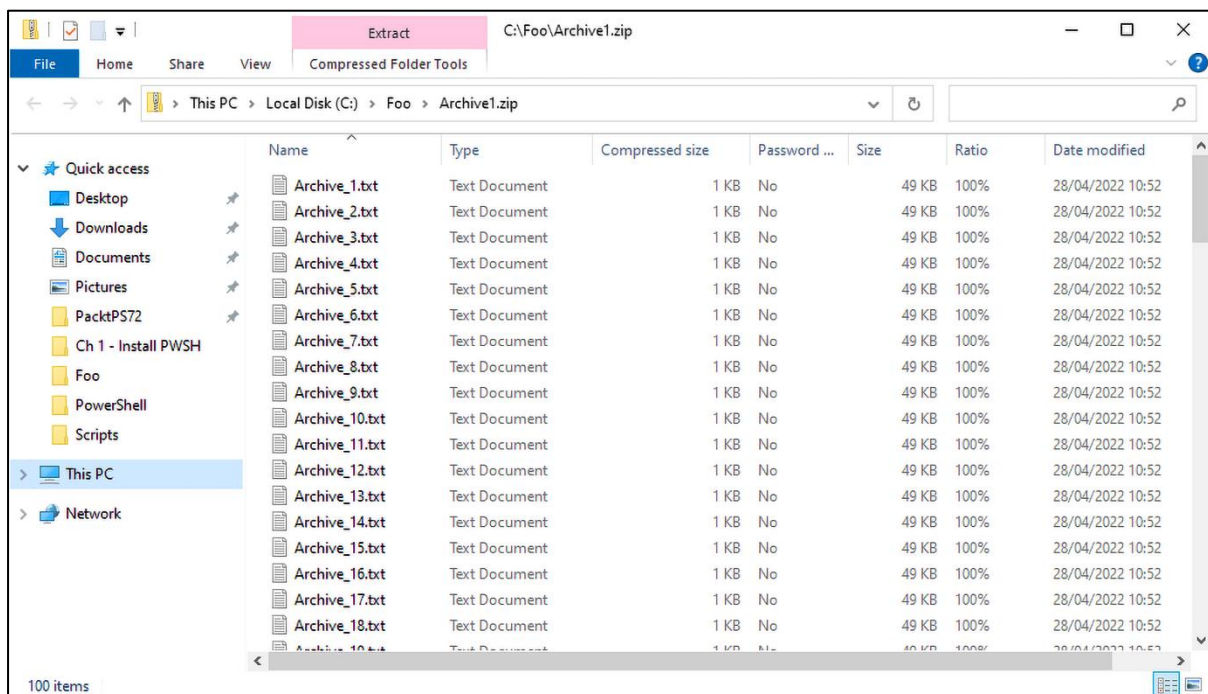
```
PS C:\Foo> # 5. Measuring files to archive
PS C:\Foo> $Files = Get-ChildItem -Path 'C:\Foo\Archive'
PS C:\Foo> $Count = $Files.Count
PS C:\Foo> $LenKB = (($Files | Measure-Object -Property length -Sum).Sum)/1mb
PS C:\Foo> "[{0}] files, occupying {1:n2}mb" -f $Count, $LenKB
[100] files, occupying 4.77mb
```

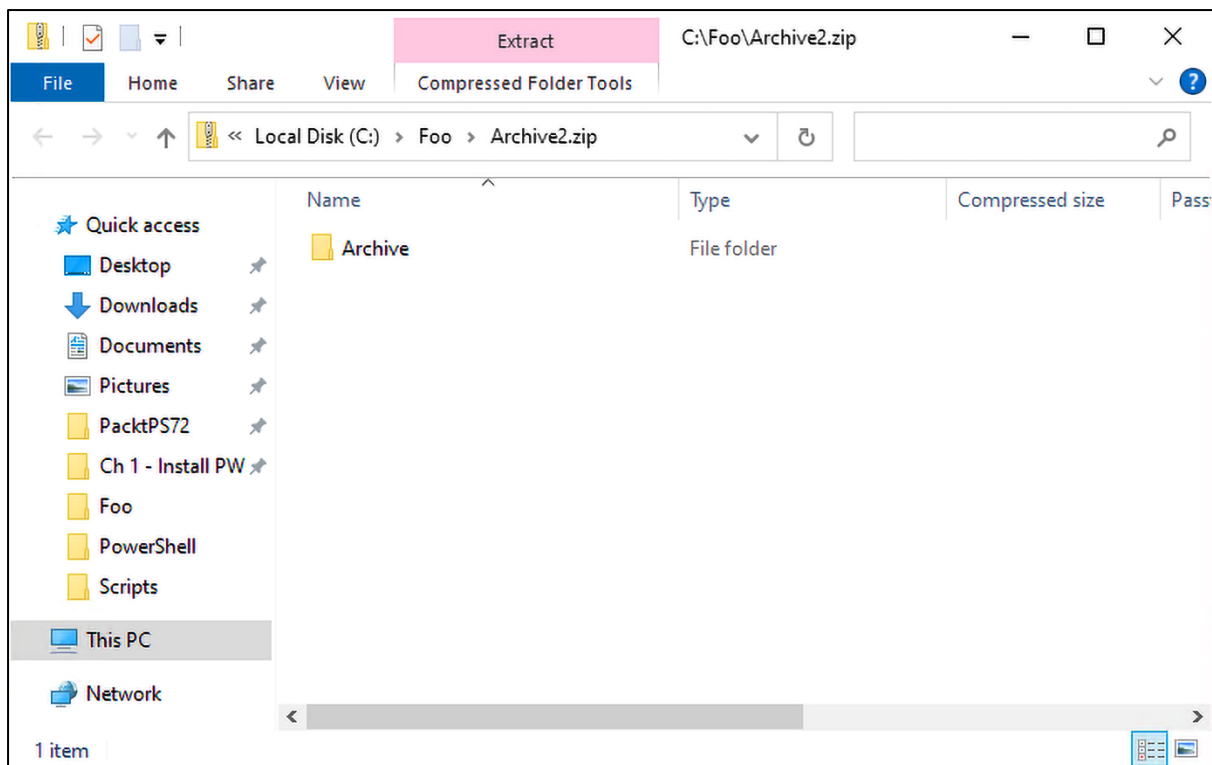


```
PS C:\Foo> # 8. Viewing the archive files
PS C:\Foo> Get-ChildItem -Path $AFILE1, $AFILE2
```

Directory: C:\Foo

Mode	LastWriteTime	Length	Name
-a---	28/04/2022 10:52	50106	Archive1.zip
-a---	28/04/2022 10:53	51706	Archive2.zip





```

PS C:\Foo> # 13. Measuring the size of the decompressed files
PS C:\Foo> $Files = Get-ChildItem -Path $Opath
PS C:\Foo> $Count = $Files.Count
PS C:\Foo> $LenKB = (($Files |
                    Measure-Object -Property length -Sum).Sum)/1mb
PS C:\Foo> "[{0}] decompressed files, occupying {1:n2}mb" -f $Count, $LenKB
[100] decompressed files, occupying 4.77mb

```

Name	Path	Size	Date Modified
dmrc.idx	C:\ProgramData\Microsoft\Windows\Device...	699 KB	30/04/2022 11:59
lastalive0.dat	C:\Windows\ServiceState\EventLog\Data	2 KB	30/04/2022 11:59
edb.chk	C:\Users\Administrator\AppData\Local\Micr...	8 KB	30/04/2022 11:59
fba94e759052658216786bfababced1b67a...	C:\Users\Administrator\AppData\Local\Pack...	3 KB	30/04/2022 11:59
CacheStorage.edb	C:\Users\Administrator\AppData\Local\Pack...	1,536 KB	30/04/2022 11:59
.inUse	C:\Users\Administrator\AppData\Local\Micr...	0 KB	30/04/2022 11:59
edb.log	C:\Users\Administrator\AppData\Local\Micr...	512 KB	30/04/2022 11:59
CacheStorage.jfm	C:\Users\Administrator\AppData\Local\Pack...	16 KB	30/04/2022 11:59
IndexedDB.edb	C:\Users\Administrator\AppData\Local\Pack...	2,048 KB	30/04/2022 11:59
microsoft.windows[1].xml	C:\Users\Administrator\AppData\Local\Pack...	1 KB	30/04/2022 11:59
WebCacheV01.dat	C:\Users\Administrator\AppData\Local\Micr...	26,112 KB	30/04/2022 11:59
lastalive1.dat	C:\Windows\ServiceState\EventLog\Data	2 KB	30/04/2022 11:58
service.0.etl	C:\Windows\Logs\NetSetup	2,368 KB	30/04/2022 11:58
UpdateHeartbeatScan\$	C:\ProgramData\Microsoft\Diagnosis\Aggre...	1 KB	30/04/2022 11:57
MoUxCoreWorker.143831e6-2916-49b6-a6...	C:\ProgramData\USOShared\Logs\System	4 KB	30/04/2022 11:56
MoUsoCoreWorker.a228b5ff-a582-4a69-8...	C:\ProgramData\USOShared\Logs\System	36 KB	30/04/2022 11:56
store.db	C:\ProgramData\USOPrivate\UpdateStore	8,200 KB	30/04/2022 11:55
DataStore.edb	C:\Windows\SoftwareDistribution\DataStore	20,992 KB	30/04/2022 11:55
WindowsUpdate.log	C:\Windows	1 KB	30/04/2022 11:55

286,265 objects

```
PS C:\Foo> # 5. Find the PSEverything module
PS C:\Foo> Find-Module -Name PSEverything
```

Version	Name	Repository	Description
3.3.0	PSEverything	PSGallery	Powershell access to Everything - Blazingly fast file system se...

```
PS C:\Foo> # 7. Find commands in the module
PS C:\Foo> Get-Command -Module PSEverything
```

CommandType	Name	Version	Source
Cmdlet	Search-Everything	3.3.0	PSEverything
Cmdlet	Select-EverythingString	3.3.0	PSEverything

```
PS C:\Foo> # 8. Getting a count of files in folders below C:\Foo
PS C:\Foo> Search-Everything |
           Get-Item |
           Group-Object DirectoryName |
           Where-Object name -ne '' |
           Format-Table -Property Name, Count
```

Name	Count
C:\Foo	17
C:\Foo\Archive	100
C:\Foo\CascadiaCode\otf\static	48
C:\Foo\CascadiaCode\ttf	8
C:\Foo\CascadiaCode\ttf\static	48
C:\Foo\CascadiaCode\woff2	8
C:\Foo\CascadiaCode\woff2\static	48
C:\Foo\Decompressed	100
C:\Foo\DownloadedModules\PSLogging\2.5.2	3

```
PS C:\Foo> # 9. Finding PowerShell scripts using wild cards
PS C:\Foo> Search-Everything *.ps1 |
           Measure-Object
```

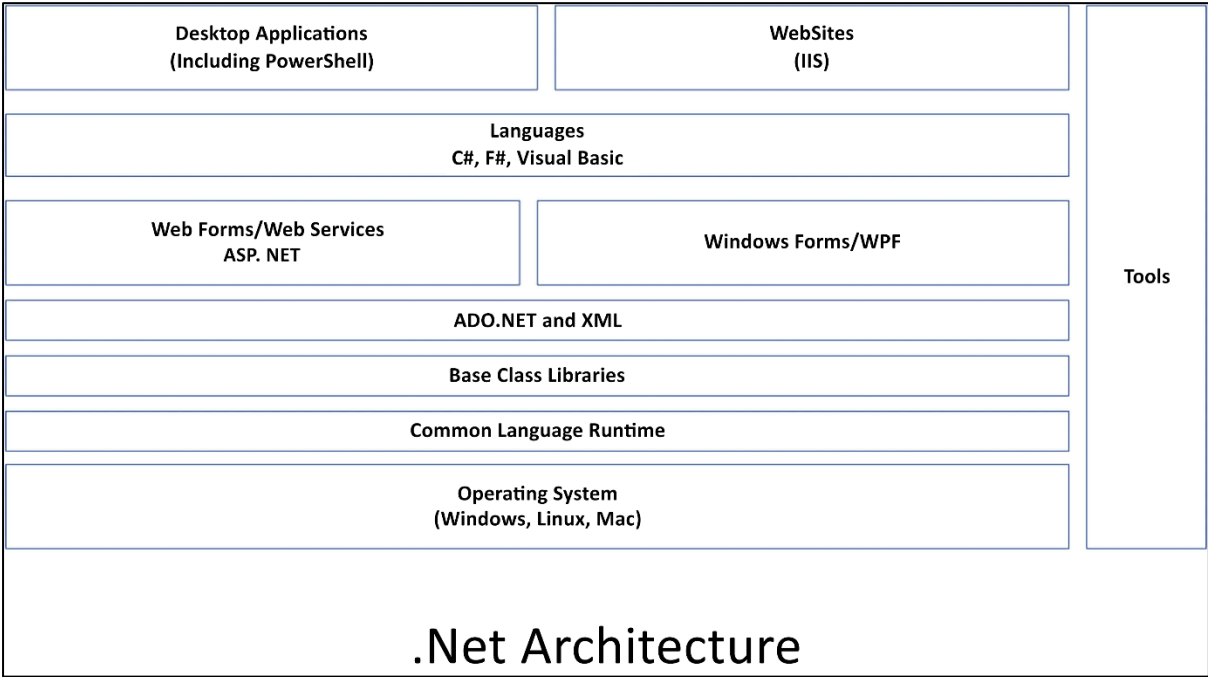
```
Count           : 4
Average         :
Sum             :
Maximum         :
Minimum         :
StandardDeviation :
Property        :
```



```
PS C:\Foo> # 10. Finding all PowerShell scripts using regular expression
PS C:\Foo> Search-Everything -RegularExpression '\.ps1$' -Global |
Measure-Object
```

```
Count          : 1629
Average        :
Sum            :
Maximum        :
Minimum        :
StandardDeviation :
Property       :
```

# Chapter 3: Exploring .NET



```

PS C:\Foo> # 1. Counting loaded assemblies
PS C:\Foo> $Assemblies = [System.AppDomain]::CurrentDomain.GetAssemblies()
PS C:\Foo> "Assemblies loaded: {0:n0}" -f $Assemblies.Count
Assemblies loaded: 86

```

```

PS C:\Foo> # 2. Viewing first 10
PS C:\Foo> $Assemblies | Select-Object -First 10

```

GAC	Version	Location
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Private.CoreLib.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\pwsh.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Runtime.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\Microsoft.PowerShell.ConsoleHost.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Management.Automation.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Threading.Thread.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Runtime.InteropServices.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Threading.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Diagnostics.Process.dll
False	v4.0.30319	C:\Program Files\PowerShell\7\System.Text.RegularExpressions.dll

```

PS C:\Foo> # 3. Checking assemblies in Windows PowerShell
PS C:\Foo> $ScriptBlock = {
    [System.AppDomain]::CurrentDomain.GetAssemblies()
}
PS C:\Foo> $PS51 = New-PSSession -UseWindowsPowerShell
PS C:\Foo> $Assin51 = Invoke-Command -Session $PS51 -ScriptBlock $ScriptBlock
PS C:\Foo> "Assemblies loaded in Windows PowerShell: {0:n0}" -f $Assin51.Count

```

Assemblies loaded in Windows PowerShell: 16

```

PS C:\Foo> # 4. Viewing Microsoft.PowerShell assemblies
PS C:\Foo> $Assin51 |
    Where-Object FullName -Match "Microsoft\PowerShell" |
    Sort-Object -Property Location

```

GAC	Version	Location	PSComputerName
True	v4.0.30319	C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\Microsoft.PowerS...	localhost
True	v4.0.30319	C:\WINDOWS\Microsoft.Net\assembly\GAC_MSIL\Microsoft.PowerS...	localhost



```

PS C:\Foo> # 5. Exploring the Microsoft.PowerShell.Management module
PS C:\Foo> $Mod = Get-Module -Name Microsoft.PowerShell.Management -ListAvailable
PS C:\Foo> $Mod | Format-List

Name           : Microsoft.PowerShell.Management
Path           : C:\program files\powershell\7\Modules\Microsoft.PowerShell.Management\Microsoft.PowerShell.Management.psd1
Description    :
ModuleType     : Manifest
Version        : 7.0.0.0
PreRelease     :
NestedModules  : {Microsoft.PowerShell.Commands.Management}
ExportedFunctions :
ExportedCmdlets : {Add-Content, Clear-Content, Get-Clipboard, Set-Clipboard, Clear-ItemProperty, Join-Path, Convert-Path,
                  Copy-ItemProperty, Get-ChildItem, Get-Content, Get-ItemProperty, Get-ItemPropertyValue, Move-ItemProperty, Get-Location,
                  Set-Location, Push-Location, Pop-Location, New-PSDrive, Remove-PSDrive, Get-PSDrive, Get-Item, New-Item, Set-Item,
                  Remove-Item, Move-Item, Rename-Item, Copy-Item, Clear-Item, Invoke-Item, Get-PSProvider, New-ItemProperty, Split-Path,
                  Test-Path, Test-Connection, Get-Process, Stop-Process, Wait-Process, Debug-Process, Start-Process, Remove-ItemProperty,
                  Rename-ItemProperty, Resolve-Path, Get-Service, Stop-Service, Start-Service, Suspend-Service, Resume-Service,
                  Restart-Service, Set-Service, New-Service, Remove-Service, Set-Content, Set-ItemProperty, Restart-Computer,
                  Stop-Computer, Rename-Computer, Get-ComputerInfo, Get-TimeZone, Set-TimeZone, Get-HotFix, Clear-RecycleBin}
ExportedVariables :
ExportedAliases  : {gcb, gin, gtz, scb, stz}

```

```

PS C:\Foo> # 6. Viewing module manifest
PS C:\Foo> $Manifest = Get-Content -Path $Mod.Path
PS C:\Foo> $Manifest | Select-Object -First 20
@{
GUID="EEFCB906-B326-4E99-9F54-8B4BB6EF3C6D"
Author="PowerShell"
CompanyName="Microsoft Corporation"
Copyright="Copyright (c) Microsoft Corporation."
ModuleVersion="7.0.0.0"
CompatiblePSEditions = @("Core")
PowerShellVersion="3.0"
NestedModules="Microsoft.PowerShell.Commands.Management.dll"
HelpInfoURI = 'https://aka.ms/powershell72-help'
FunctionsToExport = @()
AliasesToExport = @("gcb", "gin", "gtz", "scb", "stz")
CmdletsToExport=@("Add-Content",
                  "Clear-Content",
                  "Get-Clipboard",
                  "Set-Clipboard",
                  "Clear-ItemProperty",
                  "Join-Path",
                  "Convert-Path",
                  "Copy-ItemProperty",

```

```

PS C:\Foo> # 7. Discovering the module's assembly
PS C:\Foo> Import-Module -Name Microsoft.PowerShell.Management
PS C:\Foo> $Match = $Manifest | Select-String Modules
PS C:\Foo> $Line = $Match.Line
PS C:\Foo> $DLL = ($Line -Split '"')[1]
PS C:\Foo> Get-Item -Path $PSHOME\$DLL

```

Directory: C:\Program Files\PowerShell\7

Mode	LastWriteTime	Length	Name
-a---	08/03/2022 23:22	1134480	Microsoft.PowerShell.Commands.Management.dll

```

PS C:\Foo> # 8. Viewing associated loaded assembly
PS C:\Foo> $Assemblies2 = [System.AppDomain]::CurrentDomain.GetAssemblies()
PS C:\Foo> $Assemblies2 | Where-Object Location -match $DLL

```

GAC	Version	Location
False	v4.0.30319	C:\Program Files\PowerShell\7\Microsoft.PowerShell.Commands.Management.dll

```

PS C:\Foo> # 9. Getting details of a PowerShell command inside a module DLL
PS C:\Foo> $Commands = $Assemblies |
    Where-Object Location -match Commands.Management\*.dll
PS C:\Foo> $Commands.GetType() |
    Where-Object Name -match "AddContentCommand$"

```

IsPublic	IsSerial	Name	BaseType
True	False	AddContentCommand	Microsoft.PowerShell.Commands.WriteContentCommandBase

```

PS C:\Foo> # 1. Creating a FileInfo object
PS C:\Foo> $FILE = Get-ChildItem -Path $PSHOME\pwsh.exe
PS C:\Foo> $FILE

```

Directory: C:\Program Files\PowerShell\7

Mode	LastWriteTime	Length	Name
-a---	08/03/2022 23:21	287632	pwsh.exe

```

PS C:\Foo> # 2. Discovering the underlying class
PS C:\Foo> $Type = $FILE.GetType().FullName
PS C:\Foo> ".NET Class name: $Type"
.NET Class name: System.IO.FileInfo

```



```
PS C:\Foo> # 3. Getting member types of FileInfo object
PS C:\Foo> $File |
    Get-Member |
    Group-Object -Property MemberType |
    Sort-Object -Property Count -Descending
```

Count	Name	Group
23	Method	{System.IO.StreamWriter AppendText(), System.IO.FileInfo CopyTo(string destFileName), System.IO.FileInfo Copy...
16	Property	{System.IO.FileAttributes Attributes {get;set;}, datetime CreationTime {get;set;}, datetime CreationTimeUtc {...
6	NoteProperty	{string PSChildName=pwsh.exe, PSDriveInfo PSDrive=C, bool PSIsContainer=False, string PSParentPath=Microsoft...
3	CodeProperty	{System.String LinkType{get=GetLinkType;}, System.String Mode{get=Mode;}, System.String ModeWithoutHardLink{g...
2	ScriptProperty	{System.Object BaseName {get;if (\$this.Extension.Length -gt 0){\$this.Name.Remove(\$this.Name.Length - \$this.Ex...
1	AliasProperty	{Target = LinkTarget}

```
PS C:\Foo> # 4. Discovering properties of a Windows service
PS C:\Foo> Get-Service |
    Get-Member -MemberType Property
```

TypeName: System.Service.ServiceController#StartupType

Name	MemberType	Definition
BinaryPathName	Property	System.String {get;set;}
CanPauseAndContinue	Property	bool CanPauseAndContinue {get;}
CanShutdown	Property	bool CanShutdown {get;}
CanStop	Property	bool CanStop {get;}
Container	Property	System.ComponentModel.IContainer Container {get;}
DelayedAutoStart	Property	System.Boolean {get;set;}
DependentServices	Property	System.ServiceProcess.ServiceController[] DependentServices {get;}
Description	Property	System.String {get;set;}
DisplayName	Property	string DisplayName {get;set;}
MachineName	Property	string MachineName {get;set;}
ServiceHandle	Property	System.Runtime.InteropServices.SafeHandle ServiceHandle {get;}
ServiceName	Property	string ServiceName {get;set;}
ServicesDependedOn	Property	System.ServiceProcess.ServiceController[] ServicesDependedOn {get;}
ServiceType	Property	System.ServiceProcess.ServiceType ServiceType {get;}
Site	Property	System.ComponentModel.ISite Site {get;set;}
StartType	Property	System.ServiceProcess.ServiceStartMode StartType {get;}
StartupType	Property	Microsoft.PowerShell.Commands.ServiceStartupType {get;set;}
Status	Property	System.ServiceProcess.ServiceControllerStatus Status {get;}
UserName	Property	System.String {get;set;}

```
PS C:\Foo> # 5. Discovering the underlying type of an integer
PS C:\Foo> $I = 42
PS C:\Foo> $IntType = $I.GetType()
PS C:\Foo> $TypeName = $IntType.FullName
PS C:\Foo> $BaseType = $IntType.BaseType.Name
PS C:\Foo> ".NET Class name      : $TypeName"
PS C:\Foo> ".NET Class base type : $BaseType"
```

```
.NET Class name      : System.Int32
.NET Class base type : ValueType
```



```

PS C:\Foo> # 6. Looking at Process objects
PS C:\Foo> $PWSH = Get-Process -Name pwsh |
Select-Object -First 1
PS C:\Foo> $PWSH |
Get-Member |
Group-Object -Property MemberType |
Sort-Object -Property Count -Descending

```

Count	Name	Group
52	Property	{int BasePriority {get;}, System.ComponentModel.IContainer Container {get;}, bool EnableRaisingEvents {get;se...
19	Method	{void BeginErrorReadLine(), void BeginOutputReadLine(), void CancelErrorRead(), void CancelOutputRead(), void...
8	ScriptProperty	{System.Object CommandLine {get=...
7	AliasProperty	{Handles = Handlecount, Name = ProcessName, NPM = NonpagedSystemMemorySize64, PM = PagedMemorySize64, SI = Se...
4	Event	{System.EventHandler Disposed(System.Object, System.EventArgs), System.Diagnostics.DataReceivedEventHandler E...
2	PropertySet	{PSConfiguration {Name, Id, PriorityClass, FileVersion}, PSResources {Name, Id, Handlecount, WorkingSet, NonP...
1	CodeProperty	{System.Object Parent{get=GetParentProcess;}}
1	NoteProperty	{string __NounName=Process}

```

PS C:\Foo> # 7. Looking at static properties within a class
PS C:\Foo> $Max = [Int32]::MaxValue
PS C:\Foo> $Min = [Int32]::MinValue
PS C:\Foo> "Minimum value [$Min]"
PS C:\Foo> "Maximum value [$Max]"
Minimum value [-2147483648]
Maximum value [2147483647]

```



```
PS C:\Foo> # 2. Obtaining methods on the Notepad process
PS C:\Foo> $Notepad = Get-Process -Name Notepad
PS C:\Foo> $Notepad | Get-Member -MemberType Method
```

TypeName: System.Diagnostics.Process

Name	MemberType	Definition
BeginErrorReadLine	Method	void BeginErrorReadLine()
BeginOutputReadLine	Method	void BeginOutputReadLine()
CancelErrorRead	Method	void CancelErrorRead()
CancelOutputRead	Method	void CancelOutputRead()
Close	Method	void Close()
CloseMainWindow	Method	bool CloseMainWindow()
Dispose	Method	void Dispose(), void IDisposable.Dispose()
Equals	Method	bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetLifetimeService	Method	System.Object GetLifetimeService()
GetType	Method	type GetType()
InitializeLifetimeService	Method	System.Object InitializeLifetimeService()
Kill	Method	void Kill(), void Kill(bool entireProcessTree)
Refresh	Method	void Refresh()
Start	Method	bool Start()
ToString	Method	string ToString()
WaitForExit	Method	void WaitForExit(), bool WaitForExit(int milliseconds)
WaitForExitAsync	Method	System.Threading.Tasks.Task WaitForExitAsync(System.Threading.CancellationToken cancellationToken)
WaitForInputIdle	Method	bool WaitForInputIdle(), bool WaitForInputIdle(int milliseconds)

```
PS C:\Foo> # 4. Confirming Notepad process is destroyed
PS C:\Foo> Get-Process -Name Notepad
```

Get-Process: Cannot find a process with the name "Notepad". Verify the process name and call the cmdlet again.

```
PS C:\Foo> # 6. Viewing files in $Path folder
PS C:\Foo> $Files = Get-ChildItem -Path $Path
PS C:\Foo> $Files | Format-Table -Property Name, Attributes
```

Name	Attributes
SecureFile1.txt	Archive
SecureFile2.txt	Archive
SecureFile3.txt	Archive

```
PS C:\Foo> # 8. Viewing file attributes
PS C:\Foo> Get-ChildItem -Path $Path |
Format-Table -Property Name, Attributes
```

Name	Attributes
SecureFile1.txt	Archive Encrypted
SecureFile2.txt	Archive Encrypted
SecureFile3.txt	Archive Encrypted

```
PS C:\Foo> # 10. Viewing the -File attributes
PS C:\Foo> Get-ChildItem -Path $Path |
    Format-Table -Property Name, Attributes
```

Name	Attributes
----	-----
SecureFile1.txt	Archive
SecureFile2.txt	Archive
SecureFile3.txt	Archive

```
PS C:\Foo> # 1. Examining overloaded method definition
PS C:\Foo> ("a string").Trim
```

OverloadDefinitions

-----

```
string Trim()
string Trim(char trimChar)
string Trim(Params char[] trimChars)
```

```
PS C:\Foo> # 4. Examining method definition
PS C:\Foo> [Reskit.Hello]::World
```

OverloadDefinitions

-----

```
static void World()
```

```
PS C:\Foo> # 5. Using the class's method
PS C:\Foo> [Reskit.Hello]::World()
Hello World!
```



```
PS C:\Foo> # 8. Viewing method definitions
PS C:\Foo> [Reskit.Hello2]::World
```

OverloadDefinitions

-----

```
static void World()
static void World(string name)
```

```
PS C:\Foo> # 9. Calling with no parameters specified
PS C:\Foo> [Reskit.Hello2]::World()
Hello World!
```

```
PS C:\Foo> # 10. Calling new method with a parameter
PS C:\Foo> [Reskit.Hello2]::World('Jerry')
Hello Jerry!
```

Download .NET (Linux, macOS, a x

https://dotnet.microsoft.com/en-us/download

Microsoft

.NET

Why .NET

Features

Learn

Docs

More

LIVE TV

All Microsoft

Home > Download

.NET 7 Preview

Want to try out the latest preview? .NET 7.0.0-preview.3 is available. [Get .NET 7 Preview >](#)

X

# Download .NET

Downloads for .NET, including ASP.NET Core

?

Not sure where to start? See the [Hello World in 5 minutes tutorial](#) to install .NET and build your first app.

Windows

Linux

macOS

Docker

.NET is a free, cross-platform, open-source developer platform for building many different types of applications.

## .NET 6.0

LTS

Download .NET SDK  
x64

Download .NET  
Runtime

All .NET 6.0 downloads

All .NET versions

Download .NET (Linux, macOS, a xDownload .NET 6.0 SDK (v6.0.202 x

https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/s...A

Microsoft | .NET Why .NET Features Learn

Home > Download > .NET > 6.0 > .NET 6.0 SDK (v6.0.202)

Downloadsdotnet-sdk-6.0.202-win-x64.exeOpen file

Try .NET on Azure for freeGet started with 12 months of free services and build .NET cloud apps with your Azure free account.Start free > X

Thanks for downloading  
.NET 6.0 SDK (v6.0.202) -  
Windows x64 Installer!

⚠ Using Visual Studio? This release is only compatible with Visual Studio 2022 (v17.1). Using a different version? See [.NET SDKs for Visual Studio](#).

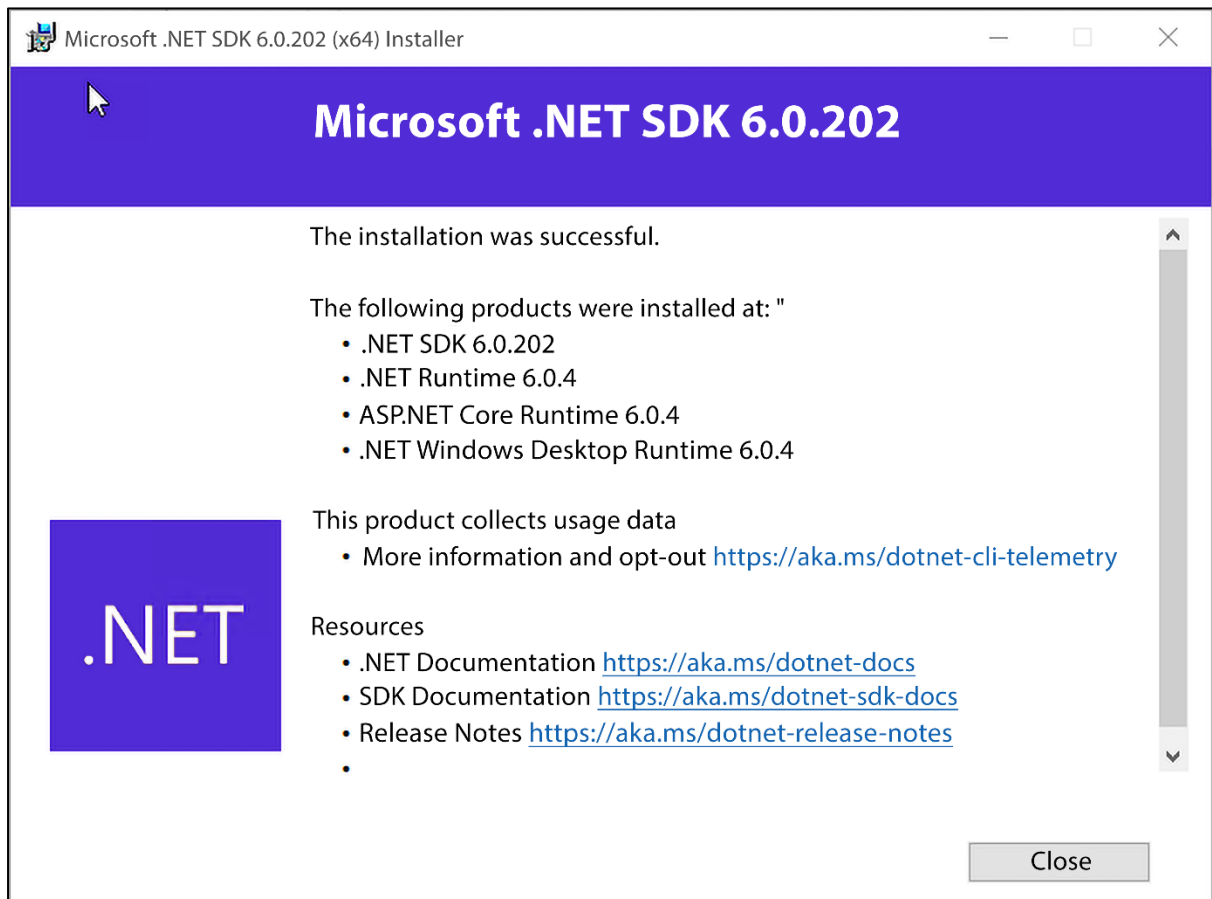
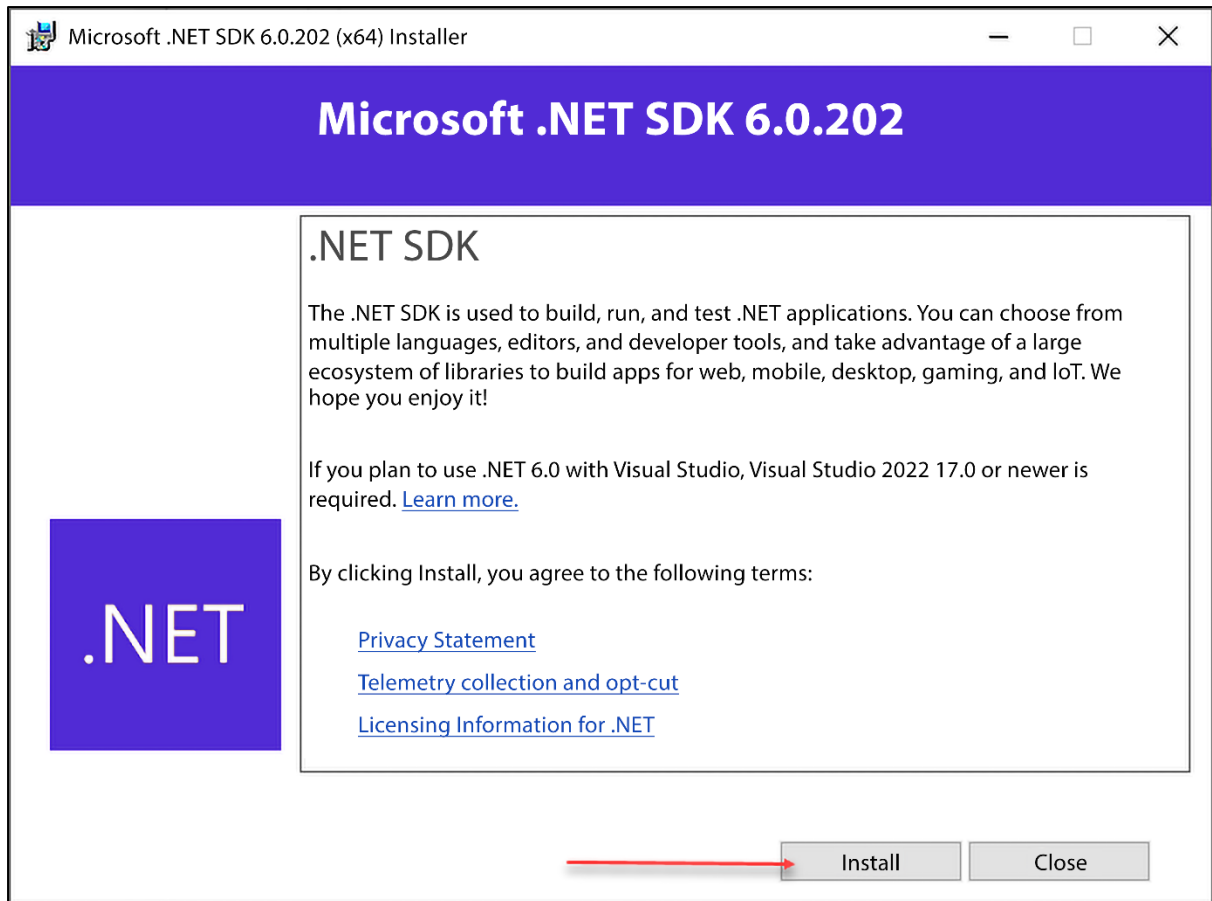
If your download doesn't start after 30 seconds, [click here to download manually](#).

Direct linkhttps://download.visualstudio.microsoft.com/download/pr/e4f4bbac-

Copy

Checksum (SHA512)54e9a70cfcc b69f4e442fd8822c1233f0672599cbb91664ff7e31c

Copy





```
PS C:\Foo> # 2. Creating the cmdlet folder
PS C:\Foo> New-Item -Path C:\Foo\Cmdlet -ItemType Directory -Force
```

Directory: C:\Foo

Mode	LastWriteTime	Length	Name
d----	02/05/2022 16:59		Cmdlet

```
PS C:\Foo> # 3. Creating a new class library project
```

```
PS C:\Foo> Set-Location C:\Foo\Cmdlet
```

```
PS C:\Foo\Cmdlet> dotnet new classlib --name SendGreeting
```

Welcome to .NET 6.0!

SDK Version: 6.0.202

Telemetry

The .NET tools collect usage data in order to help us improve your experience. It is collected by Microsoft and shared with the community. You can opt-out of telemetry by setting the DOTNET\_CLI\_TELEMETRY\_OPTOUT environment variable to '1' or 'true' using your favorite shell.

Read more about .NET CLI Tools telemetry: <https://aka.ms/dotnet-cli-telemetry>

Installed an ASP.NET Core HTTPS development certificate.

To trust the certificate run 'dotnet dev-certs https --trust' (Windows and macOS only).

Learn about HTTPS: <https://aka.ms/dotnet-https>

Write your first app: <https://aka.ms/dotnet-hello-world>

Find out what's new: <https://aka.ms/dotnet-whats-new>

Explore documentation: <https://aka.ms/dotnet-docs>

Report issues and find source on GitHub: <https://github.com/dotnet/core>

Use 'dotnet --help' to see available commands or visit: <https://aka.ms/dotnet-cli>

The template "Class Library" was created successfully.

Processing post-creation actions...

Running 'dotnet restore' on C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj...

Determining projects to restore...

Restored C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj (in 95 ms).

Restore succeeded.

```
PS C:\Foo\Cmdlet> # 4. Viewing contents of new folder
PS C:\Foo\Cmdlet> Set-Location -Path .\SendGreeting
PS C:\Foo\Cmdlet\SendGreeting> Get-ChildItem
```

Directory: C:\Foo\Cmdlet\SendGreeting

Mode	LastWriteTime	Length	Name
d----	02/05/2022 16:20		obj
-a---	02/05/2022 16:20	57	Class1.cs
-a---	02/05/2022 16:20	215	SendGreeting.csproj

```
PS C:\Foo\Cmdlet\SendGreeting> # 5. Creating and displaying global.json
PS C:\Foo\Cmdlet\SendGreeting> dotnet new globaljson
The template "global.json file" was created successfully.
PS C:\Foo\Cmdlet\SendGreeting> Get-Content -Path .\global.json
```

```
{
  "sdk": {
    "version": "6.0.202"
  }
}
```

```
PS C:\Foo\Cmdlet\SendGreeting> # 6. Adding PowerShell package
PS C:\Foo\Cmdlet\SendGreeting> $SourceName = 'Nuget.org https://api.nuget.org/v3/index.json'
PS C:\Foo\Cmdlet\SendGreeting> dotnet nuget add source --name $SourceName
Package source with Name: nuget.org added successfully.
PS C:\Foo\Cmdlet\SendGreeting> dotnet add package PowerShellStandard.Library
Determining projects to restore...
Writing C:\Users\Administrator\AppData\Local\Temp\2\tmpCD91.tmp
info : Adding PackageReference for package 'PowerShellStandard.Library' into
       project 'C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj'.
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/powershellstandard.library/index.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/powershellstandard.library/index.json 133ms
info : Restoring packages for C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj...
info :   GET https://api.nuget.org/v3-flatcontainer/powershellstandard.library/index.json
info :   OK https://api.nuget.org/v3-flatcontainer/powershellstandard.library/index.json 133ms
info :   GET https://api.nuget.org/v3-flatcontainer/powershellstandard.library/5.1.1/
       powershellstandard.library.5.1.1.nupkg
info :   OK https://api.nuget.org/v3-flatcontainer/powershellstandard.library/5.1.1/
       powershellstandard.library.5.1.1.nupkg 54ms
info : Installed PowerShellStandard.Library 5.1.1 from https://api.nuget.org/v3/index.json with content hash
       e31xJjG+Kjbv6YF3Yq6D4DL3or8v7LrNF41k3CXrWozW6hR1zc0e5KYuZJaGSiAgLnwP8wclw+I3+IWzMPZKXQ==.
info : Package 'PowerShellStandard.Library' is compatible with all the specified frameworks in
       project 'C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj'.
info : PackageReference for package 'PowerShellStandard.Library' version '5.1.1' added to
       file 'C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj'.
info : Writing assets file to disk. Path: C:\Foo\Cmdlet\SendGreeting\obj\project.assets.json
log  : Restored C:\Foo\Cmdlet\SendGreeting\SendGreeting.csproj (in 820 ms).
```

```
PS C:\Foo\Cmdlet\SendGreeting> # 9. Building the cmdlet
PS C:\Foo\Cmdlet\SendGreeting> dotnet build
Microsoft (R) Build Engine version 17.1.1+a02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
SendGreeting -> C:\Foo\Cmdlet\SendGreeting\bin\Debug\net6.0\SendGreeting.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

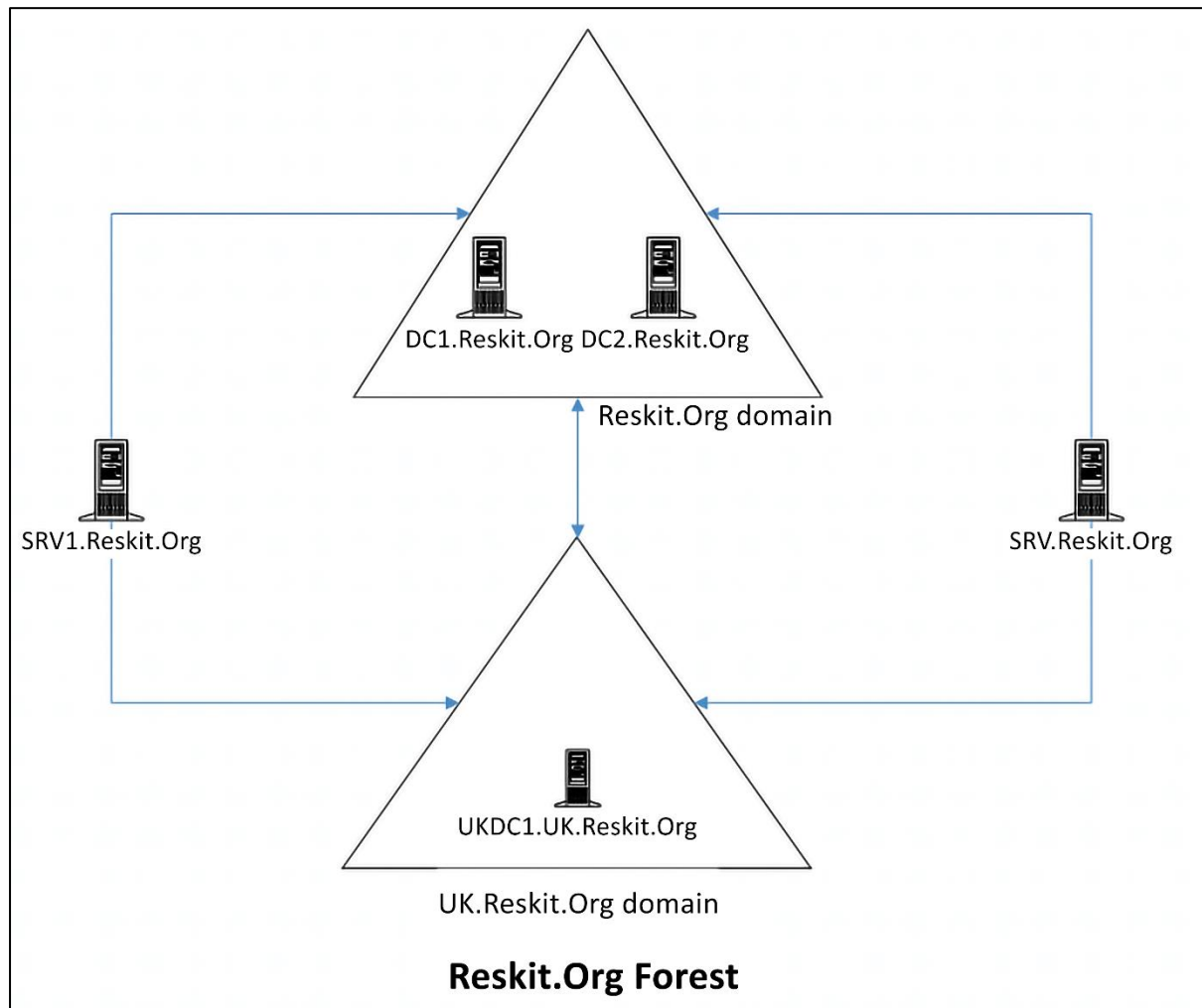
Time Elapsed 00:00:03.79
```

```
PS C:\Foo\Cmdlet\SendGreeting> # 11. Examining the module's details
PS C:\Foo\Cmdlet\SendGreeting> Get-Module SendGreeting
```

ModuleType	Version	PreRelease	Name	ExportedCommands
Binary	1.0.0.0		SendGreeting	Send-Greeting

```
PS C:\Foo\Cmdlet\SendGreeting> # 12. Using the cmdlet
PS C:\Foo\Cmdlet\SendGreeting> Send-Greeting -Name 'Jerry Garcia'
Hello Jerry Garcia - have a nice day!
```

## Chapter 4: Managing Active Directory



```
PS C:\Foo> # 1. Installing the AD Domain Services feature and management tools
PS C:\Foo> Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

Success	Restart Needed	Exit Code	Feature	Result
---------	----------------	-----------	---------	--------

True	No	Success	{Active Directory Domain Services, Group Pot...	
------	----	---------	---	--

```
PS C:\Foo> # 2. Importing the ADDeployment module
PS C:\Foo> Import-Module -Name ADDSDeployment
WARNING: Module ADDSDeployment is loaded in Windows PowerShell using WinPSCompatSession remoting
session; please note that all input and output of commands from this module will be deserialized
objects. If you want to load this module into PowerShell please use 'Import-Module -SkipEditionCheck'
syntax.
```



```
PS C:\Foo> # 3. Examining the commands in the ADDSDeployment module
PS C:\Foo> Get-Command -Module ADDSDeployment
```

CommandType	Name	Version	Source
Function	Add-ADDSDomainControllerAccount	1.0	ADDSDeployment
Function	Install-ADDSDomain	1.0	ADDSDeployment
Function	Install-ADDSDomainController	1.0	ADDSDeployment
Function	Install-ADDSDomainControllerForest	1.0	ADDSDeployment
Function	Test-ADDSDomainControllerInstallation	1.0	ADDSDeployment
Function	Test-ADDSDomainControllerUninstallation	1.0	ADDSDeployment
Function	Test-ADDSDomainInstallation	1.0	ADDSDeployment
Function	Test-ADDSDomainForestInstallation	1.0	ADDSDeployment
Function	Test-ADDSDomainControllerAccountCreation	1.0	ADDSDeployment
Function	Uninstall-ADDSDomainController	1.0	ADDSDeployment

```
PS C:\Foo> # 5. Testing DC Forest installation starting on DC1
```

```
PS C:\Foo> $ForestHT = @{
    DomainName           = 'Reskit.Org'
    InstallDNS            = $true
    NoRebootOnCompletion = $true
    SafeModeAdministratorPassword = $PSS
    ForestMode            = 'WinThreshold'
    DomainMode            = 'WinThreshold'
}
```

```
PS C:\Foo> Test-ADDSForestInstallation @ForestHT -WarningAction SilentlyContinue
```

```
RunspaceId      : 463237a4-f04b-41a6-b871-bd2da91e68fe
Message         : Operation completed successfully
Context         : Test.VerifyDcPromoCore.DCPromo.General.3
RebootRequired  : False
Status          : Success
```

```
PS C:\Foo> # 6. Creating Forest Root DC on DC1
```

```
PS C:\Foo> $NewActiveDirectoryParameterHashTable = @{
    DomainName                = 'Reskit.Org'
    SafeModeAdministratorPassword = $PSS
    InstallDNS                = $true
    DomainMode                = 'WinThreshold'
    ForestMode                = 'WinThreshold'
    Force                    = $true
    NoRebootOnCompletion      = $true
    WarningAction              = 'SilentlyContinue'
}
```

```
PS C:\Foo> Install-ADDSForest @NewActiveDirectoryParameterHashTable
```

```
RunspaceId      : 463237a4-f04b-41a6-b871-bd2da91e68fe
Message        : You must restart this computer to complete the operation.

Context        : DCPromo.General.4
RebootRequired : True
Status         : Success
```

```
PS C:\Foo> # 7. Checking key AD and related services
PS C:\Foo> Get-Service -Name DNS, Netlogon
```

Status	Name	DisplayName
Running	DNS	DNS Server
Stopped	Netlogon	Netlogon

```
PS C:\Foo> # 8. Checking DNS zones
PS C:\Foo> Get-DnsServerZone
```

ZoneName	ZoneType	IsAutoCreated	IsDslIntegrated	IsReverseLookupZone	IsSigned
_msdcs.Reskit.Org	Primary	False	False	False	False
0.in-addr.arpa	Primary	True	False	True	False
127.in-addr.arpa	Primary	True	False	True	False
255.in-addr.arpa	Primary	True	False	True	False
Reskit.Org	Primary	False	False	False	False

```
PS C:\Foo> # 1. Examining Root Directory Service Entry (DSE)
PS C:\Foo> Get-ADRootDSE -Server DC1.Reskit.Org
```

```
configurationNamingContext : CN=Configuration,DC=Reskit,DC=Org
currentTime                : 03/05/2022 19:00:28
defaultNamingContext        : DC=Reskit,DC=Org
dnsHostName                 : DC1.Reskit.Org
domainControllerFunctionality : Windows2016
domainFunctionality         : Windows2016Domain
dsServiceName               : CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,
                             CN=Sites,CN=Configuration,DC=Reskit,DC=Org
forestFunctionality         : Windows2016Forest
highestCommittedUSN         : 12829
isGlobalCatalogReady       : {TRUE}
isSynchronized              : {TRUE}
ldapServiceName             : Reskit.Org:dc1$@RESKIT.ORG
namingContexts              : {DC=Reskit,DC=Org, CN=Configuration,DC=Reskit,DC=Org,
                             CN=Schema,CN=Configuration,DC=Reskit,DC=Org,
                             DC=DomainDnsZones,DC=Reskit,DC=Org, DC=ForestDnsZones,DC=Reskit,DC=Org}
rootDomainNamingContext     : DC=Reskit,DC=Org
schemaNamingContext         : CN=Schema,CN=Configuration,DC=Reskit,DC=Org
serverName                  : CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,
                             CN=Configuration,DC=Reskit,DC=Org
subschemaSubentry           : CN=Aggregate,CN=Schema,CN=Configuration,DC=Reskit,DC=Org
supportedCapabilities        : {1.2.840.113556.1.4.800 (LDAP_CAP_ACTIVE_DIRECTORY_OID),
                             1.2.840.113556.1.4.1670 (LDAP_CAP_ACTIVE_DIRECTORY_V51_OID),
                             1.2.840.113556.1.4.1791 (LDAP_CAP_ACTIVE_DIRECTORY_LDAP_INTEG_OID),
                             1.2.840.113556.1.4.1935 (LDAP_CAP_ACTIVE_DIRECTORY_V61_OID),
                             1.2.840.113556.1.4.2080, 1.2.840.113556.1.4.2237}
supportedControl             : {1.2.840.113556.1.4.319 (LDAP_PAGED_RESULT_OID_STRING),
                             1.2.840.113556.1.4.801 (LDAP_SERVER_SD_FLAGS_OID),
```



```
PS C:\Foo> # 2. Viewing AD forest details
PS C:\Foo> Get-ADForest
```

```
ApplicationPartitions : {DC=ForestDnsZones,DC=Reskit,DC=Org, DC=DomainDnsZones,DC=Reskit,DC=Org}
CrossForestReferences : {}
DomainNamingMaster    : DC1.Reskit.Org
Domains               : {Reskit.Org}
ForestMode            : Windows2016Forest
GlobalCatalogs       : {DC1.Reskit.Org}
Name                  : Reskit.Org
PartitionsContainer    : CN=Partitions,CN=Configuration,DC=Reskit,DC=Org
RootDomain            : Reskit.Org
SchemaMaster          : DC1.Reskit.Org
Sites                 : {Default-First-Site-Name}
SPNSuffixes           : {}
UPNSuffixes           : {}
```

```
PS C:\Foo> # 3. Viewing AD Domain details
PS C:\Foo> Get-ADDomain -Current LocalComputer
```

```
AllowedDNSSuffixes    : {}
ChildDomains          : {}
ComputersContainer     : CN=Computers,DC=Reskit,DC=Org
DeletedObjectsContainer : CN=Deleted Objects,DC=Reskit,DC=Org
DistinguishedName      : DC=Reskit,DC=Org
DNSRoot               : Reskit.Org
DomainControllersContainer : OU=Domain Controllers,DC=Reskit,DC=Org
DomainMode            : Windows2016Domain
DomainSID             : S-1-5-21-3837990179-1095414155-523858238
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=Reskit,DC=Org
Forest                : Reskit.Org
InfrastructureMaster   : DC1.Reskit.Org
LastLogonReplicationInterval :
LinkedGroupPolicyObjects : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},
                          CN=Policies,CN=System,DC=Reskit,DC=Org}
LostAndFoundContainer  : CN=LostAndFound,DC=Reskit,DC=Org
ManagedBy             :
Name                   : Reskit
NetBIOSName           : RESKIT
ObjectClass            : domainDNS
ObjectGUID             : 641ac984-a9e6-410f-9f41-d8be35cc217b
ParentDomain           :
PDCEmulator           : DC1.Reskit.Org
PublicKeyRequiredPasswordRolling : True
QuotasContainer        : CN=NTDS Quotas,DC=Reskit,DC=Org
ReadOnlyReplicaDirectoryServers : {}
ReplicaDirectoryServers : {DC1.Reskit.Org}
RIDMaster              : DC1.Reskit.Org
SubordinateReferences  : {DC=ForestDnsZones,DC=Reskit,DC=Org,
                          DC=DomainDnsZones,DC=Reskit,DC=Org,
                          CN=Configuration,DC=Reskit,DC=Org}
SystemsContainer       : CN=System,DC=Reskit,DC=Org
UsersContainer         : CN=Users,DC=Reskit,DC=Org
```



```
PS C:\Foo> # 4. Checking Netlogon, ADWS, and DNS services
PS C:\Foo> Get-Service NetLogon, ADWS, DNS
```

Status	Name	DisplayName
Running	ADWS	Active Directory Web Services
Running	DNS	DNS Server
Running	Netlogon	Netlogon

```
PS C:\Foo> # 5. Getting initial AD users
PS C:\Foo> Get-ADUser -Filter * |
Sort-Object -Property Name |
Format-Table -Property Name, DistinguishedName
```

Name	DistinguishedName
Administrator	CN=Administrator,CN=Users,DC=Reskit,DC=Org
Guest	CN=Guest,CN=Users,DC=Reskit,DC=Org
krbtgt	CN=krbtgt,CN=Users,DC=Reskit,DC=Org

```
PS C:\Foo> # 6. Getting initial AD groups
PS C:\Foo> Get-ADGroup -Filter * |
Sort-Object -Property GroupScope,Name |
Format-Table -Property Name, GroupScope
```

Name	GroupScope
Access Control Assistance Operators	DomainLocal
Account Operators	DomainLocal
Administrators	DomainLocal
Allowed RODC Password Replication Group	DomainLocal
Backup Operators	DomainLocal
Cert Publishers	DomainLocal
Certificate Service DCOM Access	DomainLocal
Cryptographic Operators	DomainLocal
Denied RODC Password Replication Group	DomainLocal
Distributed COM Users	DomainLocal
DnsAdmins	DomainLocal
Event Log Readers	DomainLocal
Guests	DomainLocal
Hyper-V Administrators	DomainLocal
IIS_IUSRS	DomainLocal
Incoming Forest Trust Builders	DomainLocal
Network Configuration Operators	DomainLocal
Performance Log Users	DomainLocal
Performance Monitor Users	DomainLocal
Pre-Windows 2000 Compatible Access	DomainLocal
Print Operators	DomainLocal
RAS and IAS Servers	DomainLocal
RDS Endpoint Servers	DomainLocal
RDS Management Servers	DomainLocal
RDS Remote Access Servers	DomainLocal
Remote Desktop Users	DomainLocal
Remote Management Users	DomainLocal
Replicator	DomainLocal
Server Operators	DomainLocal
Storage Replica Administrators	DomainLocal
Terminal Server License Servers	DomainLocal
Users	DomainLocal
Windows Authorization Access Group	DomainLocal
Cloneable Domain Controllers	Global
DnsUpdateProxy	Global
Domain Admins	Global
Domain Computers	Global
Domain Controllers	Global
Domain Guests	Global
Domain Users	Global
Group Policy Creator Owners	Global
Key Admins	Global
Protected Users	Global
Read-only Domain Controllers	Global
Enterprise Admins	Universal
Enterprise Key Admins	Universal
Enterprise Read-only Domain Controllers	Universal
Schema Admins	Universal

```
PS C:\Foo> # 7. Examining Enterprise Admins group membership
PS C:\Foo> Get-ADGroupMember -Identity 'Enterprise Admins'
```

```
distinguishedName : CN=Administrator,CN=Users,DC=Reskit,DC=Org
name               : Administrator
objectClass        : user
objectGUID         : 1d9f4694-5a75-42eb-82f2-7fdf3d4b98d0
SamAccountName     : Administrator
SID                : S-1-5-21-3837990179-1095414155-523858238-500
```

```
PS C:\Foo> # 8. Checking DNS zones on DC1
PS C:\Foo> Get-DnsServerZone -ComputerName DC1
```

ZoneName	ZoneType	IsAutoCreated	IsDsIntegrated	IsReverseLookupZone	IsSigned
_msdcs.Reskit.Org	Primary	False	True	False	False
0.in-addr.arpa	Primary	True	False	True	False
127.in-addr.arpa	Primary	True	False	True	False
255.in-addr.arpa	Primary	True	False	True	False
Reskit.Org	Primary	False	True	False	False

```
PS C:\Foo> # 9. Testing domain name DNS resolution
PS C:\Foo> Resolve-DnsName -Name Reskit.Org
```

Name	Type	TTL	Section	IPAddress
Reskit.Org	AAAA	600	Answer	2a02:8010:6386:0:f55d:e38c:ceb0:8efb
Reskit.Org	A	600	Answer	10.10.10.10

```
PS C:\Foo> # 2. Checking DC1 can be resolved
PS C:\Foo> Resolve-DnsName -Name DC1.Reskit.Org -Type A
```

Name	Type	TTL	Section	IPAddress
DC1.Reskit.Org	A	3600	Answer	10.10.10.10



```
PS C:\Foo> # 3. Testing the network connection to DC1
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 445
```

```
ComputerName      : DC1.Reskit.Org
RemoteAddress     : 10.10.10.10
RemotePort        : 445
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.11
TcpTestSucceeded  : True
```

```
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389
```

```
ComputerName      : DC1.Reskit.Org
RemoteAddress     : 10.10.10.10
RemotePort        : 389
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.11
TcpTestSucceeded  : True
```

```
PS C:\Foo> # 4. Adding the AD DS features on DC2
PS C:\Foo> Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	{Active Directory Domain Services, Group Pol...

```
PS C:\Foo> # 6. Checking the computer objects in AD
PS C:\Foo> Get-ADComputer -Filter * |
Format-Table DNSHostName, DistinguishedName
```

DNSHostName	DistinguishedName
DC1.Reskit.Org	CN=DC1,OU=Domain Controllers,DC=Reskit,DC=Org
DC2.Reskit.Org	CN=DC2,OU=Domain Controllers,DC=Reskit,DC=Org

```
PS C:\Foo> # 8. Checking DCs in Reskit.Org
PS C:\Foo> $SearchBase = 'OU=Domain Controllers,DC=Reskit,DC=Org'
PS C:\Foo> Get-ADComputer -Filter * -SearchBase $SearchBase - Properties * |
Format-Table -Property DNSHostName, Enabled
```

DNSHostName	Enabled
DC1.Reskit.Org	True
DC2.Reskit.Org	True

```
PS C:\Foo> # 9. Viewing Reskit.Org domain DCs
PS C:\Foo> Get-ADDomain |
    Format-Table -Property Forest, Name,
    ReplicaDirectoryServers
```

Forest	Name	ReplicaDirectoryServers
Reskit.Org	Reskit	{DC1.Reskit.Org, DC2.Reskit.Org}

```
PS C:\Foo> # 2. Checking DC1 can be resolved
PS C:\Foo> Resolve-DnsName -Name DC1.Reskit.Org -Type A
```

Name	Type	TTL	Section	IPAddress
DC1.Reskit.Org	A	3600	Answer	10.10.10.10

```
PS C:\Foo> # 3. Checking network connection to DC1
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 445
```

```
ComputerName      : DC1.Reskit.Org
RemoteAddress     : 10.10.10.10
RemotePort        : 445
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.12
TcpTestSucceeded  : True
```

```
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389
```

```
ComputerName      : DC1.Reskit.Org
RemoteAddress     : 10.10.10.10
RemotePort        : 389
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.12
TcpTestSucceeded  : True
```

```
PS C:\Foo> # 4. Adding the AD DS -features on UKDC1
PS C:\Foo> $Features = 'AD-Domain-Services'
PS C:\Foo> Install-WindowsFeature -Name $Features -IncludeManagementTools
```

Success	Restart Needed	Exit Code	Feature	Result
True	No	Success	{Active Directory Domain Services, Group Pol...	

```

PS C:\Foo> # 7. Looking at the AD forest
PS C:\Foo> Get-ADForest -Server UKDC1.UK.Reskit.Org
ApplicationPartitions : {DC=ForestDnsZones,DC=Reskit,DC=Org, DC=DomainDnsZones,DC=Reskit,DC=Org}
CrossForestReferences : {}
DomainNamingMaster    : DC1.Reskit.Org
Domains               : {Reskit.Org, UK.Reskit.Org}
ForestMode            : Windows2016Forest
GlobalCatalogs        : {DC1.Reskit.Org, DC2.Reskit.Org, UKDC1.UK.Reskit.Org}
Name                  : Reskit.Org
PartitionsContainer    : CN=Partitions,CN=Configuration,DC=Reskit,DC=Org
RootDomain            : Reskit.Org
SchemaMaster          : DC1.Reskit.Org
Sites                 : {Default-First-Site-Name}
SPNSuffixes           : {}
UPNSuffixes           : {}

```

```

PS C:\Foo> # 8. Looking at the UK domain
PS C:\Foo> Get-ADDomain -Server UKDC1.UK.Reskit.Org
AllowedDNSSuffixes    : {}
ChildDomains          : {}
ComputersContainer    : CN=Computers,DC=UK,DC=Reskit,DC=Org
DeletedObjectsContainer : CN=Deleted Objects,DC=UK,DC=Reskit,DC=Org
DistinguishedName      : DC=UK,DC=Reskit,DC=Org
DNSRoot               : UK.Reskit.Org
DomainControllersContainer : OU=Domain Controllers,DC=UK,DC=Reskit,DC=Org
DomainMode            : Windows2016Domain
DomainSID             : S-1-5-21-2334424375-2449464812-1680912663
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=UK,DC=Reskit,DC=Org
Forest                : Reskit.Org
InfrastructureMaster   : UKDC1.UK.Reskit.Org
LastLogonReplicationInterval :
LinkedGroupPolicyObjects : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=UK,DC=Reskit,DC=Org}
LostAndFoundContainer : CN=LostAndFound,DC=UK,DC=Reskit,DC=Org
ManagedBy            :
Name                  : UK
NetBIOSName           : UK
ObjectClass            : domainDNS
ObjectGUID             : 247c3c57-6869-4347-9594-04fe466542ae
ParentDomain           : Reskit.Org
PDCEmulator           : UKDC1.UK.Reskit.Org
PublicKeyRequiredPasswordRolling : True
QuotasContainer        : CN=NTDS Quotas,DC=UK,DC=Reskit,DC=Org
ReadOnlyReplicaDirectoryServers : {}
ReplicaDirectoryServers : {UKDC1.UK.Reskit.Org}
RIDMaster              : UKDC1.UK.Reskit.Org
SubordinateReferences  : {}
SystemsContainer       : CN=System,DC=UK,DC=Reskit,DC=Org
UsersContainer         : CN=Users,DC=UK,DC=Reskit,DC=Org

```

```

PS C:\Foo> # 9. Checking on user accounts in UK domain
PS C:\Foo> Get-ADUser -Filter * |
Format-Table -Property SamAccountName, DistinguishedName

```

SamAccountName	DistinguishedName
Administrator	CN=Administrator,CN=Users,DC=UK,DC=Reskit,DC=Org
Guest	CN=Guest,CN=Users,DC=UK,DC=Reskit,DC=Org
krbtgt	CN=krbtgt,CN=Users,DC=UK,DC=Reskit,DC=Org
RESKIT\$	CN=RESKIT\$,CN=Users,DC=UK,DC=Reskit,DC=Org



```
PS C:\Foo> # 10. Checking on user accounts in parent domain
PS C:\Foo> # Get-ADUser -Filter * -Server DC1.Reskit.Org |
Format-Table -Property SamAccountName, DistinguishedName
```

SamAccountName	DistinguishedName
Administrator	CN=Administrator,CN=Users,DC=Reskit,DC=Org
Guest	CN=Guest,CN=Users,DC=Reskit,DC=Org
krbtgt	CN=krbtgt,CN=Users,DC=Reskit,DC=Org
UK\$	CN=UK\$,CN=Users,DC=Reskit,DC=Org

```
PS C:\Foo> # 7. Viewing existing AD users
PS C:\Foo> Get-ADUser -Filter * -Property * |
Format-Table -Property Name, Displayname, SamAccountName
```

Name	Displayname	SamAccountName
Administrator		Administrator
Guest		Guest
krbtgt		krbtgt
UK\$		UK\$
ThomasL	Thomas Lee (IT)	ThomasL
Rebecca Tanner	Rebecca Tanner (IT)	RLT
Jerry Garcia	Jerry Garcia (IT)	JerryG
TBR1	User to be removed	TBR1
TBR2	User to be removed	TBR2

```
PS C:\Foo> # 11. Viewing updated user
PS C:\Foo> Get-ADUser -Identity ThomasL -Properties * |
Format-Table -Property DisplayName,Name,Office,
OfficePhone,EmailAddress
```

DisplayName	Name	Office	OfficePhone	EmailAddress
Thomas Lee (IT)	ThomasL	Cookham HQ	4416835420	ThomasL@Reskit.Org

```
PS C:\Foo> # 14. Display members of the IT Team group
PS C:\Foo> Get-ADGroupMember -Identity 'IT Team' |
Format-Table SamAccountName, DistinguishedName
```

SamAccountName	DistinguishedName
JerryG	CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org
RLT	CN=Rebecca Tanner,OU=IT,DC=Reskit,DC=Org
ThomasL	CN=ThomasL,OU=IT,DC=Reskit,DC=Org



```
PS C:\Foo> # 1. Getting computers in the Reskit Domain
PS C:\Foo> Get-ADComputer -Filter * |
    Format-Table -Property Name, DistinguishedName
```

Name	DistinguishedName
DC1	CN=DC1,OU=Domain Controllers,DC=Reskit,DC=Org
DC2	CN=DC2,OU=Domain Controllers,DC=Reskit,DC=Org
UKDC1	CN=UKDC1,CN=Computers,DC=Reskit,DC=Org

```
PS C:\Foo> # 2. Getting computers in the UR Domain
PS C:\Foo> Get-ADComputer -Filter * -Server URDC1 UK.Reskit.Org |
    Format-Table -Property Name, DistinguishedName
```

Name	DistinguishedName
UKDC1	CN=UKDC1,OU=Domain Controllers,DC=UK,DC=Reskit,DC=Org

```
PS C:\Foo> # 6. Testing to ensure SRV1 is on line
PS C:\Foo> Test-NetConnection -ComputerName SRV1
```

```
ComputerName      : SRV1
RemoteAddress     : 10.10.1.13
InterfaceAlias    : Ethernet 2
SourceAddress     : 10.10.1.2
PingSucceeded     : True
PingReplyDetails (RTT) : 0 ms
```

```
PS C:\Foo> Test-NetConnection -ComputerName SRV1 -port 5985
```

```
ComputerName      : SRV1
RemoteAddress     : 10.10.10.50
RemotePort        : 5985
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.10
TcpTestSucceeded  : True
```

```
PS C:\Foo> # 9. Viewing the resulting computer accounts for Reskit.Org
PS C:\Foo> Get-ADComputer -Filter * -Properties DNSHostName |
Format-Table
```

Name	DNSHostName	Enabled
DC1	DC1.Reskit.Org	True
DC2	DC2.Reskit.Org	True
UKDC1	UKDC1.Reskit.Org	False
Wolf	Wolf.Reskit.Org	True
SRV1	SRV1.Reskit.Org	True

```
PS C:\Foo> # 2. Importing and displaying the CSV
PS C:\Foo> $Users = Import-CSV -Path C:\Foo\Users.Csv |
Sort-Object -Property Alias
PS C:\Foo> $Users | Format-Table
```

FirstName	Initials	LastName	UserPrincipalName	Alias	Description	Password
Billy	Bob	JoeBob	BBJB	BillyBob	One of the Bobs	Christmas42
C	F	Smith	CFS	Claire	Receptionist	Christmas42
Malcolm	D	Duewrong	Malcolm	Malcolm	Mr. Danger	Christmas42
P	D	Rowley	PDR	Peter	Data Team	Christmas42

```
PS C:\Foo> # 3. Adding the users using the CSV
PS C:\Foo> $Users |
ForEach-Object -Parallel {
    $User = $_
    # Create a hash table of properties to set on created user
    $Prop = @{}
    # Fill in values
    $Prop.GivenName = $User.FirstName
    $Prop.Initials = $User.Initials
    $Prop.Surname = $User.LastName
    $Prop.UserPrincipalName = $User.UserPrincipalName + "@Reskit.Org"
    $Prop.DisplayName = $User.FirstName.Trim() + " " +
        $User.LastName.Trim()
    $Prop.Description = $User.Description
    $Prop.Name = $User.Alias
    $PW = ConvertTo-SecureString -AsPlainText $User.Password -Force
    $Prop.AccountPassword = $PW
    $Prop.ChangePasswordAtLogon = $true
    $Prop.Path = 'OU=IT,DC=Reskit,DC=ORG'
    $Prop.Enabled = $true
    # Now Create the User
    New-ADUser @Prop
    # Finally, Display User Created
    "Created $($Prop.Name)"
}
```

```
Created BillyBob
Created Malcolm
Created Claire
Created Peter
```

```
PS C:\Foo> # 4. Showing all users in AD (Reskit.Org)
PS C:\Foo> Get-ADUser -Filter * -Property Description |
    Format-Table -Property Name, UserPrincipalName, Description
```

Name	UserPrincipalName	Description
Administrator		Built-in account for administering the computer/domain
Guest		Built-in account for guest access to the computer/domain
krbtgt		Key Distribution Center Service Account
UK\$		
ThomasL	thomasl@reskit.org	
Rebecca Tanner	rlt@reskit.org	
Jerry Garcia	jerryg@reskit.org	Virtualization Team
BillyBob	BBJB@Reskit.Org	One of the Bobs
Malcolm	Malcolm@Reskit.Org	Mr. Danger
Claire	CFS@Reskit.Org	Receptionist
Peter	PDR@Reskit.Org	Data Team

```
PS C:\Foo> # 7. Displaying the GPOs in the domain
PS C:\Foo> Get-GPO -All -Domain Reskit.Org |
    Sort-Object -Property DisplayName |
    Format-Table -Property DisplayName, Description, GpoStatus
```

DisplayName	Description	GpoStatus
Default Domain Controllers Policy		AllSettingsEnabled
Default Domain Policy		AllSettingsEnabled
ITPolicy	IT GPO	AllSettingsEnabled
Screen Saver Time Out		AllSettingsEnabled

file:///C:/Foo/GPOReport1.HTML#

Default Domain Policy

### Default Domain Policy

Data collected on: 07/05/2022 14:57:43

**General** [hide](#)

**Details** [show](#)

**Links** [show](#)

**Security Filtering** [show](#)

**Delegation** [show](#)

**Computer Configuration (Enabled)** [hide](#)

**Policies** [hide](#)

**Windows Settings** [hide](#)

**Security Settings** [show](#)

**User Configuration (Enabled)** [hide](#)

No settings defined.

### Screen Saver Time Out

Data collected on: 07/05/2022 14:57:44

**General** [hide](#)

**Details** [show](#)

**Links** [show](#)

**Security Filtering** [show](#)

**Delegation** [show](#)

```
PS C:\Foo> # 10. Creating simple GPO report
PS C:\Foo> $RPath2 = 'C:\Foo\GPOReport2.XML'
PS C:\Foo> $FMTS = "{0,-33} {1,-30} {2,-10} {3}"
PS C:\Foo> $FMTS -f 'Name','Linked To','Enabled','No Override'
PS C:\Foo> $FMTS -f '-----','-----','-----','-----'
PS C:\Foo> $XML.report.GPO |
    Sort-Object -Property Name |
    ForEach-Object {
        $Gname = $_.Name
        $SOM = $_.linksto.SomPath
        $ENA = $_.linksto.enabled
        $NOO = $_.linksto.nooverride
        $FMTS -f $Gname, $SOM, $ENA, $NOO
    }
```

Name	Linked To	Enabled	No Override
-----	-----	-----	-----
Default Domain Controllers Policy	Reskit.Org/Domain Controllers	true	false
Default Domain Policy	Reskit.Org	true	false
ITPolicy	Reskit.Org/IT	true	false
Screen Saver Time Out	Reskit.Org/IT	true	false



PS C:\Foo> # 10. Displaying the final report

PS C:\Foo> \$RKReport

\*\*\* Reskit.Org AD Report

\*\*\* Generated [05/07/2022 21:51:09]

\*\*\*\*\*

\*\*\* Disabled Users

SamAccountName DisplayName

-----

Guest

krbtgt

\*\*\* Users Not logged in since 04/30/2022 21:51:09

SamAccountName LastLogonDate

-----

UK\$

RLT

JerryG

BillyBob

Claire

Peter

\*\*\* High Number of Bad Password Attempts

SamAccountName BadPwdCount

-----

ThomasL 7

Malcolm 9

\*\*\* Privileged User Report

Name	Group	whenCreated	LastLogonDate
----	-----	-----	-----
Administrator	Enterprise Admins	03/05/2022 13:51:52	03/05/2022 13:58:10
Malcolm	Enterprise Admins	06/05/2022 16:00:42	06/05/2022 16:22:42
ThomasL	Enterprise Admins	06/05/2022 17:23:46	07/05/2022 18:01:48
Administrator	Domain Admins	03/05/2022 13:51:52	03/05/2022 13:58:10
ThomasL	Domain Admins	06/05/2022 17:23:46	07/05/2022 18:01:48
Administrator	Schema Admins	03/05/2022 13:51:52	03/05/2022 13:58:10

```

PS C:\Foo> # 10. Displaying the report
PS C:\Foo> $RKReport
*** Reskit.Org AD Daily AD Computer Report
*** Generated [05/08/2022 21:31:52]
*****

```

```

Computers that have never logged on
Name                               LastLogonDate
----                               -
Wolf                               Never

```

```

Computers that have not logged in over 6 months
Name                               LastLogonDate
----                               -
NLIComputer3_6month              08/11/2021 21:00:04

```

```

Computers that have not logged in 1-6 months
Name                               LastLogonDate
----                               -
NLIComputer2_1month              08/04/2022 21:00:04

```

```

Computers that have between one week and one month ago
Name                               LastLogonDate
----                               -
NLIComputer1_1week              01/05/2022 21:00:04

```

```

PS C:\Foo> # 1. Checking replication partners for DC1
PS C:\Foo> Get-ADReplicationPartnerMetadata -Target DC1.Reskit.Org |
    Format-List -Property Server, PartnerType, Partner,
    Partition, LastRep*

Server                : DC1.Reskit.Org
PartnerType           : Inbound
Partner               : CN=NTDS
                     : Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=Reskit,DC=Org
Partition             : DC=Reskit,DC=Org
LastReplicationAttempt : 08/05/2022 23:47:46
LastReplicationResult  : 0
LastReplicationSuccess : 08/05/2022 23:47:46

```

```
PS C:\Foo> # 2. Checking AD replication partner metadata in the domain
PS C:\Foo> Get-ADReplicationPartnerMetadata -Target Reskit.Org -Scope Domain |
Format-Table -Property Server, P*Type, Last*
```

Server	PartnerType	LastChangeUsn	LastReplicationAttempt	LastReplicationResult	LastReplicationSuccess
DC1.Reskit.Org	Inbound	29228	08/05/2022 23:47:46		0 08/05/2022 23:47:46
DC2.Reskit.Org	Inbound	20987	08/05/2022 23:48:24		1722 08/05/2022 20:25:4

```
PS C:\Foo> # 3. Investigating group membership metadata
PS C:\Foo> $REPLHT = @{
    Object           = (Get-ADGroup -Identity 'IT Team')
    Attribute        = 'Member'
    ShowAllLinkedValues = $true
    Server           = (Get-ADDomainController)
}
PS C:\Foo> Get-ADReplicationAttributeMetadata @REPLHT |
Format-Table -Property A*NAME, A*VALUE, *TIME
```

AttributeName	AttributeValue	FirstOriginatingCreateTime	LastOriginatingChangeTime	LastOriginatingDeleteTime
member	CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org	06/05/2022 17:28:42	06/05/2022 17:28:42	01/01/1601 00:00:00
member	CN=Rebecca Tanner,OU=IT,DC=Reskit,DC=Org	06/05/2022 17:28:42	06/05/2022 17:28:42	01/01/1601 00:00:00
member	CN=ThomasL,OU=IT,DC=Reskit,DC=Org	06/05/2022 17:28:42	06/05/2022 17:28:42	01/01/1601 00:00:00

```
PS C:\Foo> # 5 Checking updated metadata
PS C:\Foo> Get-ADReplicationAttributeMetadata @REPLHT |
Format-Table -Property A*NAME,A*VALUE, *TIME
```

AttributeName	AttributeValue	FirstOriginatingCreateTime	LastOriginatingChangeTime	LastOriginatingDeleteTime
member	CN=Claire,OU=IT,DC=Reskit,DC=Org	08/05/2022 23:52:38	09/05/2022 00:25:01	09/05/2022 00:25:01
member	CN=Malcolm,OU=IT,DC=Reskit,DC=Org	08/05/2022 23:52:38	09/05/2022 00:25:01	01/01/1601 00:00:00
member	CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org	06/05/2022 17:28:42	06/05/2022 17:28:42	01/01/1601 00:00:00
member	CN=Rebecca Tanner,OU=IT,DC=Reskit,DC=Org	06/05/2022 17:28:42	06/05/2022 17:28:42	01/01/1601 00:00:00
member	CN=ThomasL,OU=IT,DC=Reskit,DC=Org	06/05/2022 17:28:42	06/05/2022 17:28:42	01/01/1601 00:00:00

```
PS C:\Foo> # 7. Checking updated metadata
PS C:\Foo> $0 = Get-ADUser -Identity $User
PS C:\Foo> # From DC1
PS C:\Foo> Get-ADReplicationAttributeMetadata -Object $0 -Server DC1 |
Where-Object AttributeName -match 'Office'
```

```

AttributeName           : physicalDeliveryOfficeName
AttributeValue           : Marin Office
FirstOriginatingCreateTime :
IsLinkValue              : False
LastOriginatingChangeDirectoryServerIdentity : CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configur
ation,DC=Reskit,DC=Org
LastOriginatingChangeDirectoryServerInvocationId : d5f95fc9-5fad-4290-a08f-4dffa5b62bfbf
LastOriginatingChangeTime : 09/05/2022 11:39:48
LastOriginatingChangeUsn : 23224
LastOriginatingDeleteTime :
LocalChangeUsn           : 23224
Object                   : CN=Malcolm,OU=IT,DC=Reskit,DC=Org
Server                   : DC1.Reskit.Org
Version                  : 3

```

```
PS C:\Foo> # From DC2
PS C:\Foo> Get-ADReplicationAttributeMetadata -Object $0 -Server DC2 |
Where-Object AttributeName -match 'Office'
```

```

AttributeName           : physicalDeliveryOfficeName
AttributeValue           : Marin Office
FirstOriginatingCreateTime :
IsLinkValue              : False
LastOriginatingChangeDirectoryServerIdentity : CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configur
ation,DC=Reskit,DC=Org
LastOriginatingChangeDirectoryServerInvocationId : d5f95fc9-5fad-4290-a08f-4dffa5b62bfbf
LastOriginatingChangeTime : 09/05/2022 11:39:48
LastOriginatingChangeUsn : 23224
LastOriginatingDeleteTime :
LocalChangeUsn           : 34015
Object                   : CN=Malcolm,OU=IT,DC=Reskit,DC=Org
Server                   : DC2.Reskit.Org

```



```

PS C:\Foo> # 8. Examine Replication partners for both DC1, UKDC1
PS C:\Foo> Get-ADReplicationConnection |
    Format-List -Property Name,ReplicateFromDirectoryServer

Name : bae320e1-f2af-4530-aac5-647ced3129a4
ReplicateFromDirectoryServer : CN=NTDS
    Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=Reskit,DC=Org

Name : c70c8749-56db-4990-93f6-db482e6be3b9
ReplicateFromDirectoryServer : CN=NTDS
    Settings,CN=UKDC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=Reskit,DC=Org

PS C:\Foo> Get-ADReplicationConnection -Server UKDC1 |
>> Format-List -Property Name,ReplicateFromDirectoryServer

Name : 9c16d02c-8297-4b8a-89e9-fc63b7bccde1
ReplicateFromDirectoryServer : CN=NTDS
    Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=Reskit,DC=Org

Name : 05c0c646-9b40-4bcf-bb59-f3a2418333f4
ReplicateFromDirectoryServer : CN=NTDS
    Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=Reskit,DC=Org

```

```

PS C:\Foo> # 9. Use repadmin to check replication summary
PS C:\Foo> repadmin /replsummary

```

Replication Summary Start Time: 2022-05-09 13:28:14

Beginning data collection for replication summary, this may take awhile:

.....

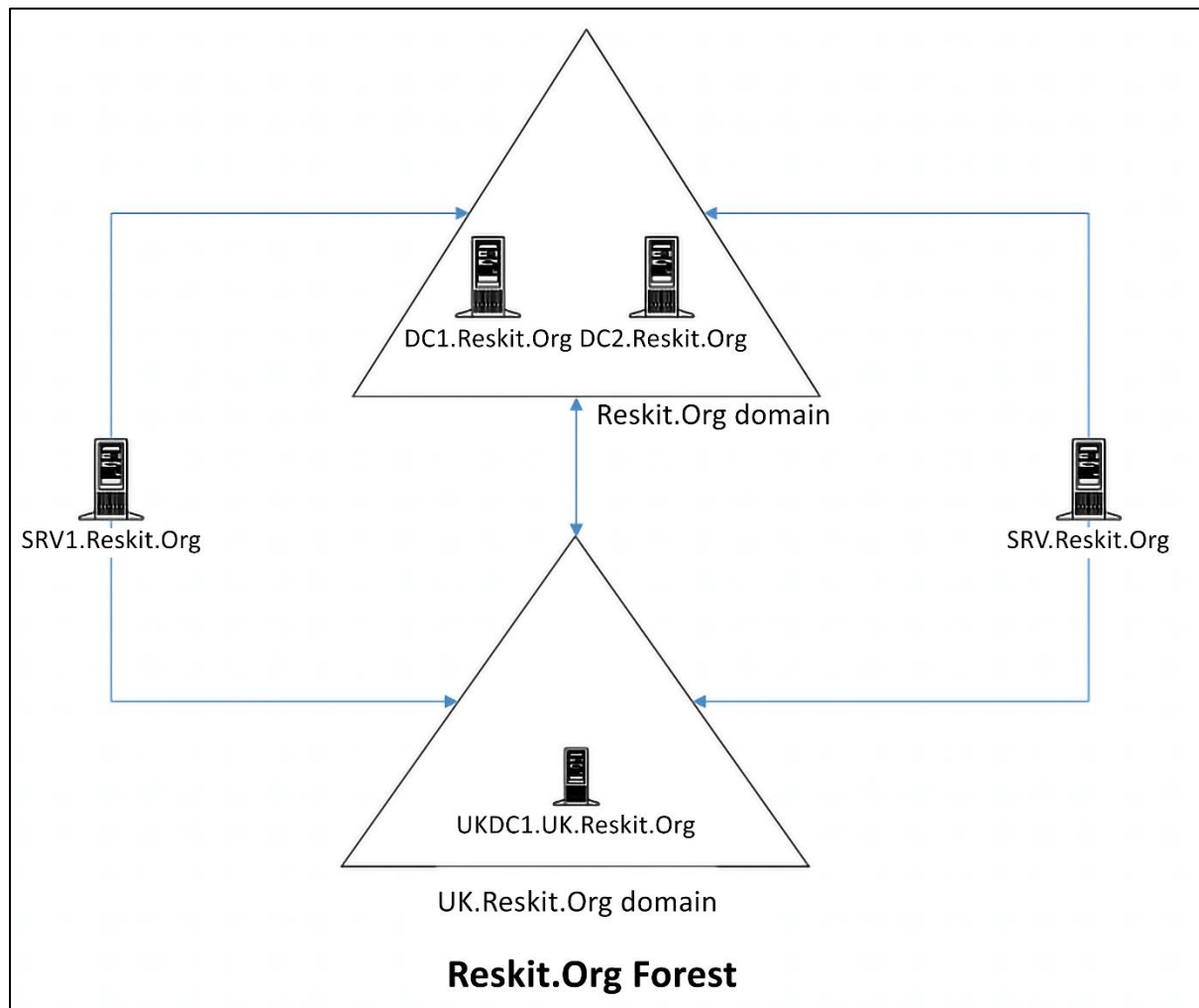
Source DSA	largest delta	fails/total	%%	error
DC1	34m:16s	0 / 7	0	
DC2	43m:02s	0 / 7	0	
UKDC1	43m:02s	0 / 6	0	

Destination DSA	largest delta	fails/total	%%	error
DC1	43m:02s	0 / 7	0	
DC2	34m:16s	0 / 7	0	
UKDC1	31m:24s	0 / 6	0	



## Chapter 5: Managing Networking



```
PS C:\Foo> # 1. Discovering the adapter, adapter interface and adapter interface index
PS C:\Foo> $IPIType = 'IPv4'
PS C:\Foo> $Adapter = Get-NetAdapter | Where-Object Status -eq 'Up'
PS C:\Foo> $Interface = $Adapter | Get-NetIPInterface -AddressFamily $IPIType
PS C:\Foo> $Index = $Interface.IfIndex
PS C:\Foo> Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPIType |
Format-Table -Property Interface*, IPAddress, PrefixLength
```

InterfaceAlias	InterfaceIndex	IPAddress	PrefixLength
Ethernet	5	169.254.118.167	16

```

PS C:\Foo> # 2. Setting a new IP address for the NIC
PS C:\Foo> $IPHT = @{
    InterfaceIndex = $Index
    PrefixLength   = 24
    IPAddress      = '10.10.10.51'
    DefaultGateway = '10.10.10.254'
    AddressFamily  = $IPType
}
PS C:\Foo> New-NetIPAddress @IPHT

```

```

IPAddress      : 10.10.10.51
InterfaceIndex  : 5
InterfaceAlias  : Ethernet
AddressFamily   : IPv4
Type            : Unicast
PrefixLength    : 24
PrefixOrigin    : Manual
SuffixOrigin    : Manual
AddressState    : Tentative
ValidLifetime   : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource    : False
PolicyStore     : ActiveStore

```

```

IPAddress      : 10.10.10.51
InterfaceIndex  : 5
InterfaceAlias  : Ethernet
AddressFamily   : IPv4
Type            : Unicast
PrefixLength    : 24
PrefixOrigin    : Manual
SuffixOrigin    : Manual
AddressState    : Invalid
ValidLifetime   : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource    : False
PolicyStore     : PersistentStore

```

```
PS C:\Foo> # 3. Verifying the new IP address
PS C:\Foo> Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |
    Format-Table IPAddress, InterfaceIndex, PrefixLength
```

IPAddress	InterfaceIndex	PrefixLength
10.10.10.51	5	24

```
PS C:\Foo> # 5. Verifying the new IP configuration
PS C:\Foo> Get-NetIPAddress -InterfaceIndex $Index -AddressFamily $IPType |
    Format-Table
```

ifIndex	IPAddress	PrefixLength	PrefixOrigin	SuffixOrigin	AddressState	PolicyStore
5	10.10.10.51	24	Manual	Manual	Preferred	ActiveStore

```
PS C:\Foo> # 6. Testing that SRV2 can see the domain controller
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org |
    Format-Table
```

ComputerName	RemotePort	RemoteAddress	PingSucceeded	PingReplyDetails (RTT)	TcpTestSucceeded
DC1.Reskit.Org	0	10.10.10.10	True	0 ms	False

```
PS C:\Foo> # 13. Testing the DNS server on DC1.Reskit.Org correctly resolves SRV2
PS C:\Foo> Resolve-DnsName -Name SRV2.Reskit.Org -Type 'A' -Server DC1.Reskit.Org
```

Name	Type	TTL	Section	IPAddress
SRV2.Reskit.Org	A	1200	Answer	10.10.10.51

```
PS C:\Foo> # 14. Checking the computer account in the AD
PS C:\Foo> Invoke-Command -ComputerName DC1 -ScriptBlock {
    Get-ADComputer -Identity SRV2}
```

```
PSComputerName      : DC1
RunspaceId          : 4f45b87c-b5df-4d35-b87d-f2f1cfbf72b4
DistinguishedName   : CN=SRV2,CN=Computers,DC=Reskit,DC=Org
DNSHostName         :
Enabled             : True
Name                : SRV2
ObjectClass         : computer
ObjectGUID          : 7b0128ff-7a95-4cf8-a8e2-cd7257928661
SamAccountName      : SRV2$
SID                 : S-1-5-21-3837990179-1095414155-523858238-3101
UserPrincipalName   :
```

```
PS C:\Foo> # 15. Adding SRV2 to the domain and restarting
PS C:\Foo> Add-Computer -DomainName Reskit.Org -Credential $Cred
WARNING: The changes will take effect after you restart the computer SRV2
```

```
PS C:\Foo> # 1. Verifying SRV2 itself is up, and that loopback is working
PS C:\Foo> Test-Connection -ComputerName SRV2 -Count 1 -IPv4
```

Destination: SRV2

Ping	Source	Address	Latency (ms)	BufferSize (B)	Status
1	SRV2	10.10.10.51	0	32	Success

```
PS C:\Foo> # 2. Testing connection to local host's WinRM port
PS C:\Foo> Test-NetConnection -ComputerName SRV2 -CommonTCPPort WinRM
```

```
ComputerName      : SRV2
RemoteAddress     : fe80::b51e:f281:f518:76a7%5
RemotePort        : 5985
InterfaceAlias    : Ethernet
SourceAddress     : fe80::b51e:f281:f518:76a7%5
TcpTestSucceeded  : True
```

```
PS C:\Foo> # 3. Testing basic connectivity to DC1
PS C:\Foo> Test-Connection -ComputerName DC1.Reskit.Org -Count 1
```

Destination: DC1.Reskit.Org

Ping	Source	Address	Latency (ms)	BufferSize (B)	Status
1	SRV2	10.10.10.10	0	32	Success

```
PS C:\Foo> # 4. Checking connectivity to SMB port on DC1
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -CommonTCPPort SMB
```

```
ComputerName      : DC1.Reskit.Org
RemoteAddress     : 10.10.10.10
RemotePort        : 445
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.51
TcpTestSucceeded  : True
```



```
PS C:\Foo> # 5. Checking connectivity to the LDAP port on DC1
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389
```

```
ComputerName      : DC1.Reskit.Org
RemoteAddress     : 10.10.10.10
RemotePort        : 389
InterfaceAlias    : Ethernet
SourceAddress     : 10.10.10.51
TcpTestSucceeded  : True
```

```
PS C:\Foo> # 6. Examining the path to a remote server on the Internet
PS C:\Foo> $NCHT = @{
    ComputerName      = 'www.Packt.Com'
    TraceRoute        = $true
    InformationLevel   = 'Detailed'
}
PS C:\Foo> Test-NetConnection @NCHT # Check our wonderful publisher
```

```
ComputerName      : www.Packt.Com
RemoteAddress     : 141.193.213.20
NameResolutionResults : 141.193.213.20
                  141.193.213.21
InterfaceAlias    : Ethernet 2
SourceAddress     : 10.10.1.17
NetRoute (NextHop) : 10.10.1.100
PingSucceeded     : True
PingReplyDetails (RTT) : 9 ms
TraceRoute        : 10.10.1.100
                  51.148.72.22
                  51.148.73.160
                  51.148.73.153
                  5.57.81.75
                  141.101.71.2
                  141.193.213.20
```

```
PS C:\Foo> # 1. Installing the DHCP feature on DC1 and add the management tools
PS C:\Foo> Import-Module -Name ServerManager -WarningAction SilentlyContinue
PS C:\Foo> Install-WindowsFeature -Name DHCP -IncludeManagementTools
```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	{DHCP Server, DHCP Server Tools}

```

PS C:\Foo> # 4. Restarting DHCP server
PS C:\Foo> Restart-Service -Name DHCPServer -Force
WARNING: Waiting for service 'DHCP Server (DHCPServer)' to start...
WARNING: Waiting for service 'DHCP Server (DHCPServer)' to start...
PS C:\Foo>

```

```

PS C:\Foo> # 5. Testing service availability
PS C:\Foo> Get-Service -Name DHCPServer |
Format-List -Property *

UserName           : NT AUTHORITY\NetworkService
Description        : Performs TCP/IP configuration for DHCP clients, including dynamic assignments of
                    IP addresses, specification of the WINS and DNS servers, and connection-specific
                    DNS names. If this service is stopped, the DHCP server will not perform
                    TCP/IP configuration for clients. If this service is disabled, any services
                    that explicitly depend on it will fail to start.
DelayedAutoStart   : False
BinaryPathName     : C:\WINDOWS\system32\svchost.exe -k DHCPServer -p
StartupType        : Automatic
Name               : DHCPServer
RequiredServices   : {RpcSs, Tcpip, SamSs, EventLog, EventSystem}
CanPauseAndContinue : True
CanShutdown        : True
CanStop            : True
DisplayName        : DHCP Server
DependentServices  : {}
MachineName        : .
ServiceName        : DHCPServer
ServicesDependedOn : {RpcSs, Tcpip, SamSs, EventLog, EventSystem}
StartType          : Automatic
ServiceHandle       : Microsoft.Win32.SafeHandles.SafeServiceHandle
Status             : Running
ServiceType        : Win32OwnProcess, Win32ShareProcess
Site               :
Container          :

```

```

PS C:\Foo> # 3. Getting IPV4 scopes from the server
PS C:\Foo> Get-DhcpServerv4Scope -ComputerName DC1.Reskit.Org

ScopeId      SubnetMask      Name      State      StartRange      EndRange      LeaseDuration
-----
10.10.10.0    255.255.255.0   ReskitOrg Active      10.10.10.150    10.10.10.199

```

```

PS C:\Foo> # 6. Viewing server options
PS C:\Foo> Get-DhcpServerv4OptionValue | Format-Table -AutoSize

OptionId Name      Type      Value      VendorClass UserClass PolicyName
-----
15      DNS Domain Name String     {Reskit.Org}
6       DNS Servers  IPv4Address {10.10.10.10}

```

```
PS C:\Foo> # 7. Viewing scope specific options
PS C:\Foo> Get-DhcpServerv4OptionValue -ScopeId '10.10.10.0' |
Format-Table -AutoSize
```

OptionId	Name	Type	Value	VendorClass	UserClass	PolicyName
51	Lease	DWord	{691200}			
3	Router	IPv4Address	{10.10.10.254}			

```
PS C:\Foo> # 8. Viewing DHCPv4 option definitions
PS C:\Foo> Get-DhcpServerv4OptionDefinition | Format-Table -AutoSize
```

Name	OptionId	Type	VendorClass	MultiValued
Classless Static Routes	121	BinaryData		False
Subnet Mask	1	IPv4Address		False
Time Offset	2	DWord		False
Router	3	IPv4Address		True
Time Server	4	IPv4Address		True
Name Servers	5	IPv4Address		True
DNS Servers	6	IPv4Address		True
Log Servers	7	IPv4Address		True
Cookie Servers	8	IPv4Address		True
LPR Servers	9	IPv4Address		True
Impress Servers	10	IPv4Address		True
Resource Location Servers	11	IPv4Address		True
Host Name	12	String		False
Boot File Size	13	Word		False
Merit Dump File	14	String		False
DNS Domain Name	15	String		False
Swap Server	16	IPv4Address		False
Root Path	17	String		False
Extensions Path	18	String		False
IP Layer Forwarding	19	Byte		False
Nonlocal Source Routing	20	Byte		False
Policy Filter Masks	21	IPv4Address		True
Max DG Reassembly Size	22	Word		False
Default IP Time-to-live	23	Byte		False
Path MTU Aging Timeout	24	DWord		False
Path MTU Plateau Table	25	Word		True
MTU Option	26	Word		False
All subnets are local	27	Byte		False
Broadcast Address	28	IPv4Address		False
Perform Mask Discovery	29	Byte		False
Mask Supplier Option	30	Byte		False
Perform Router Discovery	31	Byte		False
Router Solicitation Address	32	IPv4Address		False
Static Route Option	33	IPv4Address		True
Trailer Encapsulation	34	Byte		False
ARP Cache Timeout	35	DWord		False
Ethernet Encapsulation	36	Byte		False
TCP Default Time-to-live	37	Byte		False
Keepalive Interval	38	DWord		False
Keepalive Garbage	39	Byte		False
NIS Domain Name	40	String		False
NIS Servers	41	IPv4Address		True
NTP Servers	42	IPv4Address		True
Vendor Specific Info	43	BinaryData		False
WINS/NBNS Servers	44	IPv4Address		True
Rebinding (T2) Time Value	59	DWord		False
NIS+ Domain Name	64	String		False
NIS+ Servers	65	IPv4Address		True
Boot Server Host Name	66	String		False
Bootfile Name	67	String		False
Mobile IP Home Agents	68	IPv4Address		True
Simple Mail Transport Protocol (SMTP) Servers	69	IPv4Address		True
Post Office Protocol (POP3) Servers	70	IPv4Address		True
Network News Transport Protocol (NNTP) Servers	71	IPv4Address		True
World Wide Web (WWW) Servers	72	IPv4Address		True
Finger Servers	73	IPv4Address		True
Internet Relay Chat (IRC) Servers	74	IPv4Address		True
StreetTalk Servers	75	IPv4Address		True
StreetTalk Directory Assistance (STDA) Servers	76	IPv4Address		True



```
PS C:\Foo> # 1. Adding DHCP RSAT tools
PS C:\Foo> Import-Module -Name ServerManager -WarningAction SilentlyContinue
PS C:\Foo> Install-WindowsFeature -Name RSAT-DHCP
Success Restart Needed Exit Code Feature Result
-----
True      No                Success    {Remote Server Administration Tools, DHCP Se...
```

```
PS C:\Foo> # 3. Viewing the scopes on DC1
PS C:\Foo> Get-DhcpServerv4Scope -ComputerName DC1
```

ScopeId	SubnetMask	Name	State	StartRange	EndRange	LeaseDuration
10.10.10.0	255.255.255.0	ReskitOrg	Active	10.10.10.150	10.10.10.199	

```
PS C:\Foo> # 4. Getting V4 scope statistics from DC1
PS C:\Foo> Get-DhcpServerv4ScopeStatistics -ComputerName DC1
```

ScopeId	Free	InUse	PercentageInUse	Reserved	Pending	SuperscopeName
10.10.10.0	50	0	0	0	0	

```
PS C:\Foo> # 5. Discovering a free IP address
PS C:\Foo> Get-DhcpServerv4FreeIPAddress -ComputerName dc1 -ScopeId 10.10.10.42
10.10.10.150
```

```
PS C:\Foo> # 7. Getting IP interface
PS C:\Foo> $NIC |
Get-NetIPInterface |
Where-Object AddressFamily -eq 'IPv4'
```

ifIndex	InterfaceAlias	AddressFamily	NlMtu(Bytes)	InterfaceMetric	Dhcp	ConnectionState	PolicyStore
5	Ethernet	IPv4	1500	15	Disabled	Connected	ActiveStore

```
PS C:\Foo> # 9. Checking IP address assigned
PS C:\Foo> Get-NetIPAddress -InterfaceAlias "Ethernet" |
Where-Object AddressFamily -eq 'IPv4'
```

```
IPAddress           : 10.10.10.150
InterFaceIndex      : 5
InterFaceAlias       : Ethernet
AddressFamily        : IPv4
Type                 : Unicast
PrefixLength         : 24
PrefixOrigin         : Dhcp
SuffixOrigin         : Dhcp
AddressState         : Preferred
ValidLifetime        : 7.23:59:23
PreferredLifetime    : 7.23:59:23
SkipAsSource         : False
PolicyStore          : ActiveStore
```



```
PS C:\Foo> # 10. Getting updated V4 scope statistics from DC1
PS C:\Foo> Get-DhcpServerv4ScopeStatistics -ComputerName DC1
```

ScopeId	Free	InUse	PercentageInUse	Reserved	Pending	SuperscopeName
10.10.10.0	49	1	2	0	0	

```
PS C:\Foo> # 11. Discovering the next free IP address
PS C:\Foo> Get-DhcpServerv4FreeIPAddress -ComputerName dc1 -ScopeId 10.10.10.42
10.10.10.151
```

```
PS C:\Foo> # 12. Checking IPv4 DNS name resolution
PS C:\Foo> Resolve-DnsName -Name SRV2.Reskit.Org -Type A
```

Name	Type	TTL	Section	IPAddress
SRV2.Reskit.Org	A	1200	Question	10.10.10.150

```
PS C:\Foo> # 1. Importing the DHCP Server module explicitly
PS C:\Foo> Import-Module -Name DHCPServer
WARNING: Module DHCPServer is loaded in Windows PowerShell using WinPSCompatSession
remoting session; please note that all input and output of commands from this
module will be deserialized objects. If you want to load this module into
PowerShell please use 'Import-Module -SkipEditionCheck' syntax.
```

```
PS C:\Foo> # 5. Testing net connection to SRV2
PS C:\Foo> Clear-DnsClientCache
PS C:\Foo> Resolve-DnsName -Name SRV2.Reskit.Org -Type A
```

Name	Type	TTL	Section	IPAddress
SRV2.Reskit.Org	A	1200	Answer	10.10.10.199

```
PS C:\Foo> Test-Connection -TargetName SRV2.Reskit.Org
```

Destination: SRV2.Reskit.Org

Ping	Source	Address	Latency(ms)	BufferSize(B)	Status
1	DC1	10.10.10.199	0	32	Success
2	DC1	10.10.10.199	0	32	Success
3	DC1	10.10.10.199	1	32	Success
4	DC1	10.10.10.199	0	32	Success

```

PS C:\Foo> # 1. Installing the DHCP server feature on DC2
PS C:\Foo> Import-Module -Name ServerManager -WarningAction SilentlyContinue
PS C:\Foo> $FEATUREHT = @{
    Name = 'DHCP'
    IncludeManagementTools = $True
}
PS C:\Foo> Install-WindowsFeature @FEATUREHT

```

Success	Restart	Needed	Exit Code	Feature Result
True	No		Success	{DHCP Server, DHCP Server Tools}

```

PS C:\Foo> # 4. Viewing authorized DHCP servers in the Reskit domain
PS C:\Foo> Get-DhcpServerinDC

```

IPAddress	DnsName
10.10.10.10	dc1.reskit.org
10.10.10.11	dc2.reskit.org

```

PS C:\Foo> # 5. Configuring fail-over and load balancing
PS C:\Foo> $FAILOVERHT = @{
    ComputerName = 'DC1.Reskit.Org'
    PartnerServer = 'DC2.Reskit.Org'
    Name = 'DC1-DC2'
    ScopeID = '10.10.10.0'
    LoadBalancePercent = 60
    SharedSecret = 'j3RryIsTheB3est!'
    Force = $true
    Verbose = $True
}
PS C:\Foo> Invoke-Command -ComputerName DC1.Reskit.Org -ScriptBlock {
    Add-DhcpServerv4Failover @Using:FAILOVERHT
}

```

```

VERBOSE: A new failover relationship will be created between servers DC1.Reskit.Org and DC2.Reskit.Org. The
configuration of the specified scopes on server DC1.Reskit.Org will be replicated to the partner server.
VERBOSE: Add scopes on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update properties for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update properties for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update delay offer for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update delay offer for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update NAP properties for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update NAP properties for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update superscope for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update superscope for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update IP ranges for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update IP ranges for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update exclusions for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update exclusions for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update reservations for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update reservations for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update policies for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update policies for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Update options for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Update options for scope 10.10.10.0 (1 of 1) on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Add scopes on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Disable scopes on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Disable scopes on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Creation of failover configuration on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Creation of failover configuration on partner server DC2.Reskit.Org .....Successful.
VERBOSE: Creation of failover configuration on host server DC1.Reskit.Org .....In progress.
VERBOSE: Creation of failover configuration on host server DC1.Reskit.Org .....Successful.
VERBOSE: Activate scopes on partner server DC2.Reskit.Org .....In progress.
VERBOSE: Activate scopes on partner server DC2.Reskit.Org .....Successful.

```



```

PS C:\Foo> # 6. Getting active scopes (from both servers!)
PS C:\Foo> $DHCPServers = 'DC1.Reskit.Org', 'DC2.Reskit.Org'
PS C:\Foo> $DHCPServers |
    ForEach-Object {
        "Server: $_" | Format-Table
        Get-DhcpServerv4Scope -ComputerName $_ | Format-Table
    }

```

Server: DC1.Reskit.Org

ScopeId	SubnetMask	Name	State	StartRange	EndRange	LeaseDuration
10.10.10.0	255.255.255.0	ReskitOrg	Active	10.10.10.150	10.10.10.199	

Server: DC2.Reskit.Org

ScopeId	SubnetMask	Name	State	StartRange	EndRange	LeaseDuration
10.10.10.0	255.255.255.0	ReskitOrg	Active	10.10.10.150	10.10.10.199	

```

PS C:\Foo> # 7. Viewing DHCP server statistics from both DHCP Servers
PS C:\Foo> $DHCPServers |
    ForEach-Object {
        "Server: $_" | Format-Table
        Get-DhcpServerv4ScopeStatistics -ComputerName $_ | Format-Table
    }

```

Server: DC1.Reskit.Org

ScopeId	Free	InUse	PercentageInUse	Reserved	Pending	SuperscopeName
10.10.10.0	49	1	2	1	0	

Server: DC2.Reskit.Org

ScopeId	Free	InUse	PercentageInUse	Reserved	Pending	SuperscopeName
10.10.10.0	49	1	2	1	0	

```

PS C:\Foo> # 8. Viewing DHCP reservations from both DHCP Servers
PS C:\Foo> $DHCPServers |
    ForEach-Object {
        "Server: $_" | Format-Table
        Get-DhcpServerv4Reservation -scope 10.10.10.42 -ComputerName $_ |
            Format-Table
    }

```

Server: DC1.Reskit.Org

IPAddress	ScopeId	ClientId	Name	Type	Description
10.10.10.199	10.10.10.42	00-15-5d-01-01-4c	SRV2.Reskit.Org	Both	

Server: DC2.Reskit.Org

IPAddress	ScopeId	ClientId	Name	Type	Description
10.10.10.199	10.10.10.42	00-15-5d-01-01-4c	SRV2.Reskit.Org	Both	

```
PS C:\Foo> # 1. Installing the DNS feature on DC2
PS C:\Foo> Import-Module -Name ServerManager -WarningAction SilentlyContinue
PS C:\Foo> Install-WindowsFeature -Name DNS -IncludeManagementTools
```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	{DNS Server, DNS Server Tools}

```
PS C:\Foo> # 10. Viewing recursion settings
PS C:\Foo> $DNSRV |
Select-Object -ExpandProperty ServerRecursion
```

Enable	True
AdditionalTimeout(s)	4
RetryInterval(s)	3
Timeout(s)	8
SecureResponse	True

```
PS C:\Foo> # 11. Viewing server cache settings
PS C:\Foo> $DNSRV |
Select-Object -ExpandProperty ServerCache
```

MaxTTL	: 1.00:00:00
MaxNegativeTTL	: 00:15:00
MaxRBSize	: 20480
EnablePollutionProtection	: True
LockingPercent	: 100
StoreEmptyAuthenticationResponse	: True
IgnorePolicies	: False

```
PS C:\Foo> # 12. Viewing ENDS Settings
PS C:\Foo> $DNSRV |
Select-Object -ExpandProperty ServerEdns
```

CacheTimeout	EnableProbes	EnableReception
00:15:00	True	True



```
PS C:\Foo> # 14. Resolving SRV2.Reskit.Org on DC2
PS C:\Foo> Resolve-DnsName -Name SRV2.Reskit.Org -Type A -Server DC2
```

Name	Type	TTL	Section	IPAddress
SRV2.Reskit.Org	A	1200	Answer	10.10.10.199

```
PS C:\Foo> # 1. Creating a new primary forward DNS zone for Cookham.Net
PS C:\Foo> $ZHT1 = @{
```

```
    Name = 'Cookham.Net'
    ResponsiblePerson = 'dnsadmin.cookham.net.'
    ReplicationScope = 'Domain'
    ComputerName = 'DC1.Reskit.Org'
}
```

```
PS C:\Foo> Add-DnsServerPrimaryZone @ZHT1 -Verbose
```

```
VERBOSE: Adding DNS primary (AD integrated/file-backed forward/reverse lookup) zone Cookham.Net on DC1.Reskit.Org server.
VERBOSE: AllowUpdate successfully set on server DC1.Reskit.Org.
```

```
PS C:\Foo> # 4. Checking the DNS zones on DC1
PS C:\Foo> Get-DNSServerZone -ComputerName DC1
```

ZoneName	ZoneType	IsAutoCreated	IsDsIntegrated	IsReverseLookupZone	IsSigned
_msdcs.Reskit.Org	Primary	False	True	False	False
0.in-addr.arpa	Primary	True	False	True	False
10.10.10.in-addr.arpa	Primary	False	True	True	False
127.in-addr.arpa	Primary	True	False	True	False
255.in-addr.arpa	Primary	True	False	True	False
Cookham.Net	Primary	False	True	False	False
Reskit.Org	Primary	False	True	False	False
TrustAnchors	Primary	False	True	False	Fals

```
PS C:\Foo> # 4. Checking the DNS zones on DC1
PS C:\Foo> Get-DNSServerZone -ComputerName DC1
```

ZoneName	ZoneType	IsAutoCreated	IsDsIntegrated	IsReverseLookupZone	IsSigned
_msdcs.Reskit.Org	Primary	False	True	False	False
0.in-addr.arpa	Primary	True	False	True	False
10.10.10.in-addr.arpa	Primary	False	True	True	False
127.in-addr.arpa	Primary	True	False	True	False
255.in-addr.arpa	Primary	True	False	True	False
Cookham.Net	Primary	False	True	False	False
Reskit.Org	Primary	False	True	False	False
TrustAnchors	Primary	False	True	False	False

```
PS C:\Foo> # 8. Testing DNS resolution on DC2, DC1
PS C:\Foo> # Testing The CNAME from DC1
PS C:\Foo> Resolve-DnsName -Server DC1.Reskit.Org -Name 'Mail.Cookham.Net'
```

Name	Type	TTL	Section	NameHost
Mail.Cookham.Net	CNAME	3600	Answer	Home.Cookham.Net

```
Name      : Home.Cookham.Net
QueryType : A
```

```
PS C:\Foo> # Testing the MX on DC2
PS C:\Foo> Resolve-DnsName -Server DC2.Reskit.Org -Name 'Cookham.Net'
```

Name	Type	TTL	Section	PrimaryServer	NameAdministrator	SerialNumber
Cookham.Net	SOA	3600	Authority	dc2.reskit.org	dnsadmin.Cookham.Net	4

```
PS C:\Foo> # 9. Testing the reverse lookup zone
PS C:\Foo> Resolve DnsName -Name '10.10.10.10'
```

Name	Type	TTL	Section	NameHost
10.10.10.10.in-addr.arpa.	PTR	1200	Question	DC1.Reskit.Org

```
PS C:\Foo> # 1. Obtaining the IP addresses of DNS servers for Packt.Com
PS C:\Foo> $NameServers = Resolve-DnsName -Name Packt.Com -Type NS |
Where-Object Name -eq 'Packt.Com'
PS C:\Foo> $NameServers
```

Name	Type	TTL	Section	NameHost
Packt.Com	NS	1461	Answer	eva.ns.cloudflare.Com
Packt.Com	NS	1461	Answer	max.ns.cloudflare.Com

```
PS C:\Foo> # 2. Obtaining the IPV4 addresses for these hosts
PS C:\Foo> $NameServerIPs = foreach ($Server in $NS) {
    (Resolve-DnsName -Name $Server.NameHost -Type A),IPAddress
}
PS C:\Foo> $NameServerIPs
```

```
108.162.192.114
172.64.32.114
173.245.58.114
108.162.193.132
172.64.33.132
173.245.59.132
```

```
PS C:\Foo> # 4. Checking zone on DC1
PS C:\Foo> Get-DnsServerZone -Name Packt.Com
```

ZoneName	ZoneType	IsAutoCreated	IsDnsIntegrated	IsReverseLookupZone	IsSigned
Packt.Com	Forwarder	False	False	False	

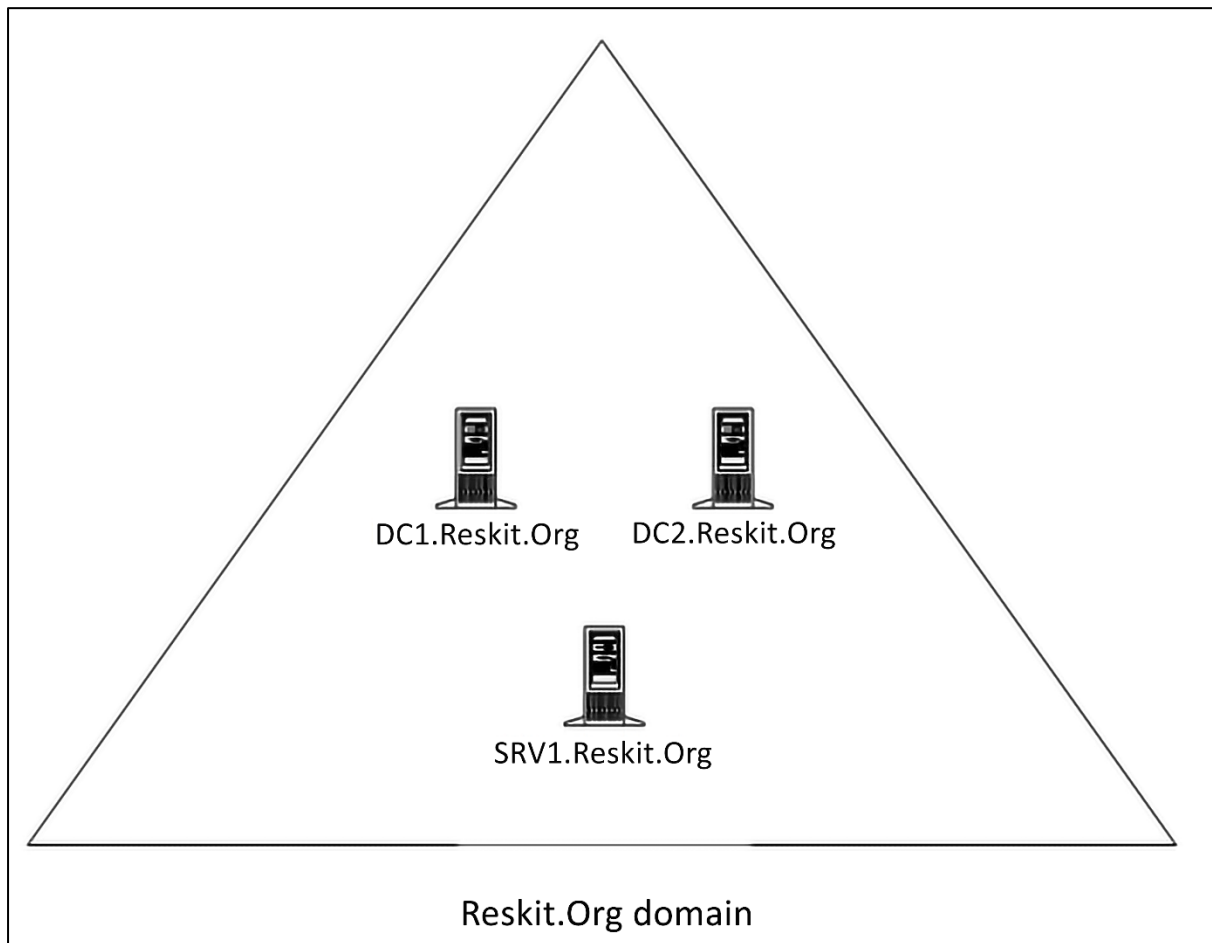
```
PS C:\Foo> # 5. Testing conditional forwarding
```

```
PS C:\Foo> Resolve-DnsName -Name WWW.Packt.Com -Server DC1 |
Format-Table
```

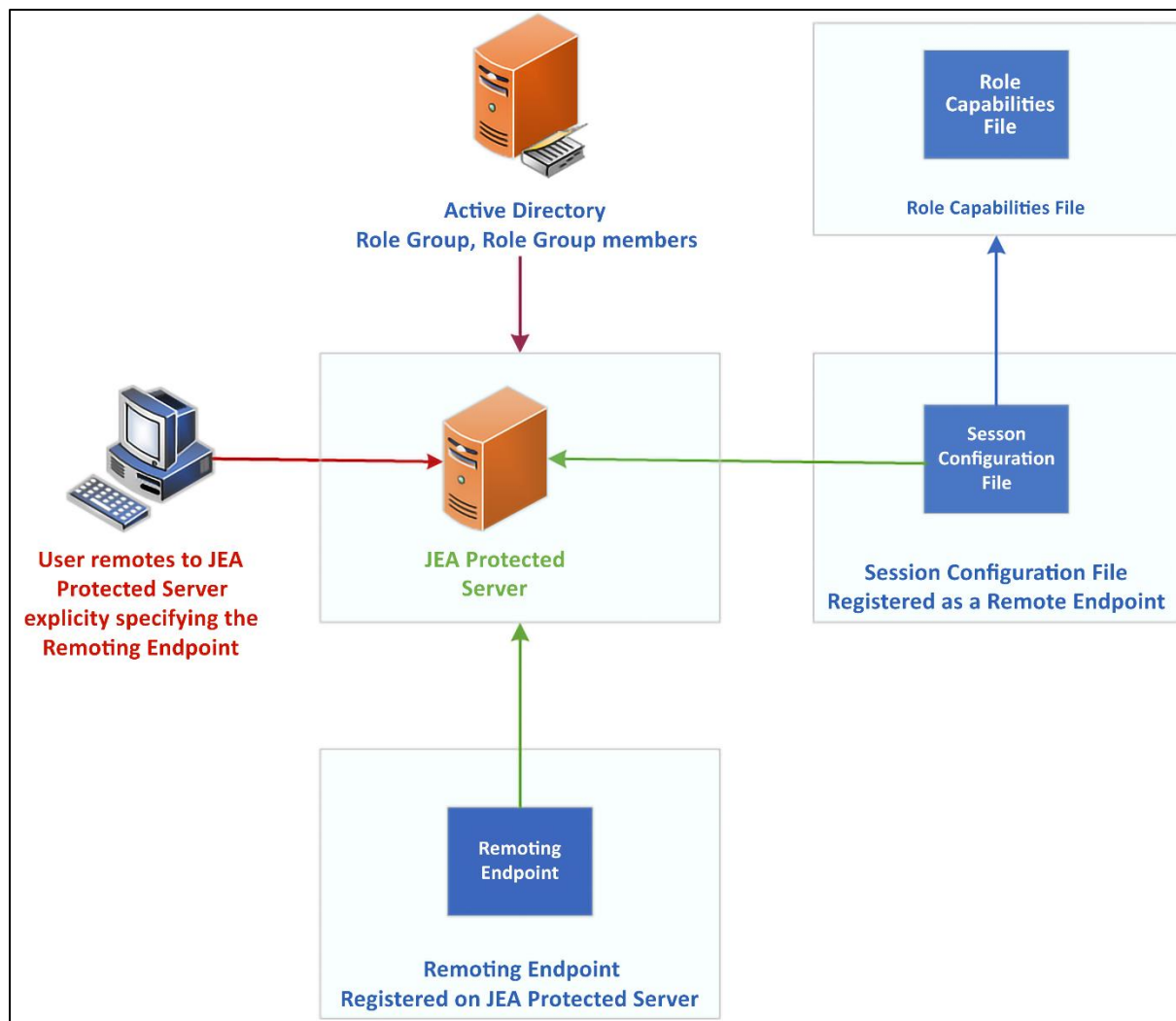
Name	Type	TTL	Section	NameHost
WWW.Packt.Com	CNAME	300	Answer	packtcommerce.wpengine.Com

```
Name          : packtcommerce.wpengine.Com
QueryType     : A
TTL           : 120
Section       : Answer
IP4Address    : 34.105.173.16
```

## Chapter 6: Implementing Enterprise Security







```
PS C:\Foo> # 8. Testing the session configuration file
PS C:\Foo> Test-PSSessionConfigurationFile -Path $P
True
```

```
PS C:\Foo> # 9. Enabling remoting on DC1
PS C:\Foo> Enable-PSRemoting -Force |
Out Null
```

WARNING: PowerShell remoting has been enabled only for PowerShell 6+ configurations and does not affect Windows PowerShell remoting configurations. Run this cmdlet in Windows PowerShell to affect all PowerShell remoting configurations.

```

PS C:\Foo> # 10. Registering the JEA session configuration remoting endpoint
PS C:\Foo> $SCHT = @{
    Path = $P
    Name = 'DnsAdminsJEA'
    Force = $true
}
PS C:\Foo> Register-PSSessionConfiguration @SCHT

WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Plugin

Type      keys      Name
-----
Container {Name=DnsAdminsJEA} DnsAdminsJEA

```

```

PS C:\Foo> # 11. Viewing remoting endpoints
PS C:\Foo> Get-PSSessionConfiguration |
    Format-Table -Property Name, PSVersion, Run*Account

```

Name	PSVersion	RunAsVirtualAccount
DnsAdminsJEA	7.2	True
PowerShe11.7	7.2	false
PowerShe11.7.2.5.7.2		false

```

PS C:\Foo> # 12. Verifying what the user can do
PS C:\Foo> $SCHT = @{
    ConfigurationName = 'DnsAdminsJEA'
    Username = 'Reskit\JerryG'
}
PS C:\Foo> Get-PSSessionCapability @SCHT |
    Sort-Object -Property Module

```

CommandType	Name	Version	Source
Alias	clear -> Clear-Host		
Function	Stop-Service		
Function	Start-Service		
Function	Select-Object		
Function	Restart-Computer		
Function	Out-Default		
Function	Measure-Object		
Function	Get-HW		
Function	Get-Help		
Application	ipconfig.exe	10.0.2034...	C:\WINDOWS\system32\ipconfig.exe
Function	Get-Command		
Function	Exit-PSSession		
Function	Clear-Host		
Function	Get-FormatData		
Application	whoami.exe	10.0.2034...	C:\WINDOWS\system32\whoami.exe
Alias	gh -> Get-Help		
Alias	gcm -> Get-Command		
Alias	select -> Select-Object		
Alias	cls -> Clear-Host		
Alias	exsn -> Exit-PSSession		
Alias	measure -> Measure-Object		
Function	Set-DnsServerClientSubnet	2.0.0.0	DnsServer
Function	Set-DnsServerDiagnostics	2.0.0.0	DnsServer
Function	Set-DnsServerCache	2.0.0.0	DnsServer
Function	Set-DnsServerDnsSecZoneSetting	2.0.0.0	DnsServer
Function	Set-DnsServer	2.0.0.0	DnsServer

```
PS C:\Foo> # 15. Getting commands available within the JEA session
PS C:\Foo> Invoke-Command -ScriptBlock $SB1 @ICMHT |
Sort-Object -Property Module |
Select-Object -First 15
```

CommandType	Name	Version	Source	PSComputerName
Function	Get-HW			DC1.Reskit.Org
Function	Get-Help			DC1.Reskit.Org
Function	Get-FormatData			DC1.Reskit.Org
Function	Stop-Service			DC1.Reskit.Org
Function	Measure-Object			DC1.Reskit.Org
Function	Start-Service			DC1.Reskit.Org
Function	Out-Default			DC1.Reskit.Org
Function	Get-Command			DC1.Reskit.Org
Function	Exit-PSSession			DC1.Reskit.Org
Function	Restart-Computer			DC1.Reskit.Org
Function	Clear-Host			DC1.Reskit.Org
Function	Select-Object			DC1.Reskit.Org
Function	Add-DnsServerClientSubnet	2.0.0.0	DnsServer	DC1.Reskit.Org
Function	Reset-DnsServerZoneKeyMasterRole	2.0.0.0	DnsServer	DC1.Reskit.Org
Function	Restore-DnsServerPrimaryZone	2.0.0.0	DnsServer	DC1.Reskit.Org

```
PS C:\Foo> # 16. Invoking a JEA-defined function in a JEA session as JerryG
PS C:\Foo> Invoke-Command -ScriptBlock $SB2 @ICMHT
Hello JEA World
```

```
PS C:\Foo> # 17. Getting DNSServer commands available to JerryG
PS C:\Foo> $C = Invoke-Command -ScriptBlock $SB3 @ICMHT
PS C:\Foo> "$($C.Count) DNS commands available"
131 DNS commands available
```

```
PS C:\Foo> # 18. Examining the contents of the transcripts folder
PS C:\Foo> Get-ChildItem -Path $PSCHT.TranscriptDirectory
```

Directory: C:\JEATranscripts

Mode	LastWriteTime	Length	Name
-a---	28/06/2022 17:08	12819	PowerShell_transcript.DC1.LLTy91d2.20220628170803.txt
-a---	28/06/2022 17:02	12845	PowerShell_transcript.DC1.n+EhEOFS.20220628170207.txt
-a---	28/06/2022 17:06	13108	PowerShell_transcript.DC1.Uaexd6D5.20220628170559.txt
-a---	28/06/2022 17:07	779	PowerShell_transcript.DC1.xs+t5kPA.20220628170730.txt

```

PS C:\Foo> # 19. Examining a transcript
PS C:\Foo> Get-ChildItem -Path $PSCHT.TranscriptDirectory |
    Select-Object -First 1 |
    Get-Content
WSManStackVersion: 3.0
*****
PS>CommandInvocation(Get-Command): "Get-Command"
>> ParameterBinding(Get-Command): name="Name"; value="*-DNSSERVER*"
>> ParameterBinding(Get-Command): name="ListImported"; value="False"
>> ParameterBinding(Get-Command): name="ShowCommandInfo"; value="False"

```

CommandType	Name	Version	Source
Function	Add-DnsServerClientSubnet	2.0.0.0	DnsServer
Function	Add-DnsServerConditionalForwarderZone	2.0.0.0	DnsServer
Function	Add-DnsServerDirectoryPartition	2.0.0.0	DnsServer
Function	Add-DnsServerForwarder	2.0.0.0	DnsServer
Function	Add-DnsServerPrimaryZone	2.0.0.0	DnsServer
Function	Add-DnsServerQueryResolutionPolicy	2.0.0.0	DnsServer

```

PS C:\Foo> # 2. Discovering classic event logs on SRV1
PS C:\Foo> Get-EventLog -LogName *

```

Max(K)	Retain	OverflowAction	Entries	Log
20,480	0	OverwriteAsNeeded	1,241	Application
20,480	0	OverwriteAsNeeded	0	HardwareEvents
512	7	OverwriteOlder	0	Internet Explorer
20,480	0	OverwriteAsNeeded	0	Key Management Service
20,480	0	OverwriteAsNeeded	10,334	Security
20,480	0	OverwriteAsNeeded	7,692	System
15,360	0	OverwriteAsNeeded	475	Windows PowerShell

```

PS C:\Foo> # 3. Discovering and measuring all event logs on this host
PS C:\Foo> $Logs = Get-WinEvent -ListLog *
PS C:\Foo> "There are $($Logs.Count) total event logs on SRV1"
There are 423 total event logs on SRV1

```

```

PS C:\Foo> # 4. Discovering and measuring all event logs on DC1
PS C:\Foo> $SB1 = {Get-WinEvent -ListLog *}
PS C:\Foo> $LogsDC1 = Invoke-Command -ComputerName DC1 -ScriptBlock $SB1
PS C:\Foo> "There are $($LogsDC1.Count) total event logs on DC1"
There are 415 total event logs on DC1

```



```
PS C:\Foo> # 5. Discovering SRV1 log member details
PS C:\Foo> $Logs | Get-Member
```

TypeName: System.Diagnostics.Eventing.Reader.EventLogConfiguration

Name	MemberType	Definition
Dispose	Method	void Dispose(), void IDisposable.Dispose()
Equals	Method	bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetType	Method	type GetType()
SaveChanges	Method	void SaveChanges()
ToString	Method	string ToString()
FileSize	NoteProperty	long FileSize=1118208
IsLogFull	NoteProperty	bool IsLogFull=False
LastAccessTime	NoteProperty	datetime LastAccessTime=30/06/2022 12:45:34
LastWriteTime	NoteProperty	datetime LastWriteTime=30/06/2022 12:45:34
OldestRecordNumber	NoteProperty	long OldestRecordNumber=1
RecordCount	NoteProperty	long RecordCount=475
IsClassicLog	Property	bool IsClassicLog {get;}
IsEnabled	Property	bool IsEnabled {get;set;}
LogFilePath	Property	string LogFilePath {get;set;}
LogIsolation	Property	System.Diagnostics.Eventing.Reader.EventLogIsolation LogIsolation {get;}
LogMode	Property	System.Diagnostics.Eventing.Reader.EventLogMode LogMode {get;set;}
LogName	Property	string LogName {get;}
LogType	Property	System.Diagnostics.Eventing.Reader.EventLogType LogType {get;}
MaximumSizeInBytes	Property	long MaximumSizeInBytes {get;set;}
OwningProviderName	Property	string OwningProviderName {get;}
ProviderBufferSize	Property	System.Nullable[int] ProviderBufferSize {get;}
ProviderControlGuid	Property	System.Nullable[guid] ProviderControlGuid {get;}
ProviderKeywords	Property	System.Nullable[long] ProviderKeywords {get;set;}
ProviderLatency	Property	System.Nullable[int] ProviderLatency {get;}
ProviderLevel	Property	System.Nullable[int] ProviderLevel {get;set;}
ProviderMaximumNumberOfBuffers	Property	System.Nullable[int] ProviderMaximumNumberOfBuffers {get;}
ProviderMinimumNumberOfBuffers	Property	System.Nullable[int] ProviderMinimumNumberOfBuffers {get;}
ProviderNames	Property	System.Collections.Generic.IEnumerable[string] ProviderNames {get;}
SecurityDescriptor	Property	string SecurityDescriptor {get;set;}

```
PS C:\Foo> # 6. Measuring enabled logs on SRV1
PS C:\Foo> $Logs |
    Where-Object IsEnabled |
    Measure-Object |
    Select-Object -Property Count
```

```
Count
-----
343
```

```

PS C:\Foo> # 7. Measuring enabled logs on DC1
PS C:\Foo> $LogsDC1 |
    Where-Object IsEnabled |
    Measure-Object |
    Select-Object -Property Count

```

Count

340

```

PS C:\Foo> # 9. Discovering PowerShell-related logs
PS C:\Foo> $Logs |
    Where-Object LogName -match 'Powershell'

```

LogMode	MaximumSizeInBytes	RecordCount	LogName
Circular	15728640	475	Windows PowerShell
Circular	15728640	2630	PowerShellCore/Operational
Circular	15728640	401	Microsoft-Windows-PowerShell/Operational
Retain	1048985600	0	Microsoft-Windows-PowerShell/Admin
Circular	1052672	0	Microsoft-Windows-PowerShell-DesiredStateConfiguration-FileDownloadManager/Operational

```

PS C:\Foo> # 10. Examining PowerShellCore event log
PS C:\Foo> Get-WinEvent -LogName 'PowerShellCore/Operational' |
    Select-Object -First 10

```

ProviderName: PowerShellCore

TimeCreated	Id	Level	DisplayName	Message
30/06/2022 12:50:31	12039	Information		Modifying activity Id and correlating
30/06/2022 12:50:31	8196	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	12039	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	8196	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	12039	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	8196	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	12039	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	8196	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	12039	Information		Modifying activity Id and correlating
30/06/2022 12:50:30	8196	Information		Modifying activity Id and correlating

```

PS C:\Foo> # 1. Getting Security log events
PS C:\Foo> $SecLog = Get-WinEvent -ListLog Security
PS C:\Foo> "Security Event log entries: [{10,10:N0}]" -f $Seclog.RecordCount
Security Event log entries: [ 80,792]

```

```

PS C:\Foo> # 2. Getting all Windows Security log event details
PS C:\Foo> $SecEvents = Get-WinEvent -LogName Security
PS C:\Foo> "Found $($SecEvents.count) security events on DC1"
Found 80792 security events on DC1

```

```

PS C:\Foo> # 3: Examining Security event log event members
PS C:\Foo> $SecEvents |
    Get-Member

```

TypeName: System.Diagnostics.Eventing.Reader.EventLogRecord

Name	MemberType	Definition
Dispose	Method	void Dispose(), void IDisposable.Dispose()
Equals	Method	bool Equals(System.Object obj)
FormatDescription	Method	string FormatDescription(), string FormatDescription(System.Collections.Generic.IEnumerable[System.Object] values)
GetHashCode	Method	int GetHashCode()
GetPropertyValues	Method	System.Collections.Generic.IList[System.Object] GetPropertyValues(System.Diagnostics.Eventing.Reader.EventLogPropertySelector propertySelector)
GetType	Method	type GetType()
ToString	Method	string ToString()
ToXml	Method	string ToXml()
Message	NoteProperty	string Message=An account was logged off...
ActivityId	Property	System.Nullable[guid] ActivityId {get;}
Bookmark	Property	System.Diagnostics.Eventing.Reader.EventBookmark Bookmark {get;}
ContainerLog	Property	string ContainerLog {get;}
Id	Property	int Id {get;}
Keywords	Property	System.Nullable[long] Keywords {get;}
KeywordsDisplayNames	Property	System.Collections.Generic.IEnumerable[string] KeywordsDisplayNames {get;}
Level	Property	System.Nullable[byte] Level {get;}
LevelDisplayName	Property	string LevelDisplayName {get;}
LogName	Property	string LogName {get;}
MachineName	Property	string MachineName {get;}
MatchedQueryIds	Property	System.Collections.Generic.IEnumerable[int] MatchedQueryIds {get;}
Opcode	Property	System.Nullable[short] Opcode {get;}
OpcodeDisplayName	Property	string OpcodeDisplayName {get;}
ProcessId	Property	System.Nullable[int] ProcessId {get;}
Properties	Property	System.Collections.Generic.IList[System.Diagnostics.Eventing.Reader.EventProperty] Properties {get;}
ProviderId	Property	System.Nullable[guid] ProviderId {get;}
ProviderName	Property	string ProviderName {get;}
Qualifiers	Property	System.Nullable[int] Qualifiers {get;}
RecordId	Property	System.Nullable[long] RecordId {get;}
RelatedActivityId	Property	System.Nullable[guid] RelatedActivityId {get;}
Task	Property	System.Nullable[int] Task {get;}
TaskDisplayName	Property	string TaskDisplayName {get;}
ThreadId	Property	System.Nullable[int] ThreadId {get;}
TimeCreated	Property	System.Nullable[datetime] TimeCreated {get;}
UserId	Property	System.Security.Principal.SecurityIdentifier UserId {get;}
Version	Property	System.Nullable[byte] Version {get;}



```

PS C:\Foo> # 4. Summarizing security events by event ID
PS C:\Foo> $SecEvents |
    Sort-Object -Property Id |
    Group-Object -Property ID |
    Sort-Object -Property Name |
    Format-Table -Property Name, Count

```

Name	Count
1100	7
1101	2
4608	11
4616	23
4624	23829
4625	5
4634	22346
4647	6
4648	171
4662	242
4672	21887
4688	122
4696	11
4713	1

```

PS C:\Foo> # 5. Getting all successful logon events on DC1
PS C:\Foo> $Logons = $SecEvents | Where-Object ID -eq 4624 # logon event
PS C:\Foo> "Found $($Logons.Count) logon events on DC1"
Found 23829 logon events on DC1

```

```

PS C:\Foo> # 6. Getting all failed logon events on DC1
PS C:\Foo> $FLogons = $SecEvents | Where-Object ID -eq 4625 # failed logon event
PS C:\Foo> "Found $($FLogons.Count) failed logon events on DC1"
Found 5 failed logon events on DC1

```



```

PS C:\Foo> # 8. Summarizing successful logon events on DC1
PS C:\Foo> $LogonEvents |
    Group-Object -Property LogonType |
    Sort-Object -Property Name |
    Select-Object -Property Name,Count

```

Name	Count
0	11
10	6
2	99
3	22383
5	1328
7	2

```

PS C:\Foo> # 10. Summarizing failed logon events on DC1
PS C:\Foo> $FLogonEvents |
    Group-Object -Property Account |
    Sort-Object -Property Name |
    Format-Table Name, Count

```

Name	Count
Reskit\Malcolm	2
Reskit\BobbyW	3

```

PS C:\Foo> # 1. Discovering the GPO-related files
PS C:\Foo> Get-ChildItem -Path $PSHOME -Filter *Core*Policy*

```

Directory: C:\Program Files\PowerShell\7

Mode	LastWriteTime	Length	Name
-a---	15/06/2022 17:24	17386	InstallPSCorePolicyDefinitions.ps1
-a---	15/06/2022 17:08	9675	PowerShellCoreExecutionPolicy.adml
-a---	15/06/2022 17:08	6198	PowerShellCoreExecutionPolicy.admx

```
PS C:\Foo> # 2. Installing the PowerShell 7 group policy files
PS C:\Foo> $LOC = $PSHOME + '\InstallPSCorePolicyDefinitions.ps1'
PS C:\Foo> & $LOC -VERBOSE
VERBOSE: Copying C:\Program Files\PowerShell\7\PowerShellCoreExecutionPolicy.admx
to C:\WINDOWS\PolicyDefinitions
VERBOSE: PowerShellCoreExecutionPolicy.admx was installed successfully
VERBOSE: Copying C:\Program Files\PowerShell\7\PowerShellCoreExecutionPolicy.adml
to C:\WINDOWS\PolicyDefinitions\en-US
VERBOSE: PowerShellCoreExecutionPolicy.adml was installed successfully
```

```
PS C:\Foo> # 3. Creating and displaying a new GPO for the IT group
PS C:\Foo> $GPOName = 'PowerShell GPO for IT'
PS C:\Foo> $PshGPO = New-GPO -Name $GPOName
PS C:\Foo> Get-GPO -Name $GPOName
```

```
DisplayName      : PowerShell GPO for IT
DomainName       : Reskit.Org
Owner            : RESKIT\Domain Admins
Id               : 694960eb-41a8-43de-95e1-1a12ffe4006e
GpoStatus        : AllSettingsEnabled
Description       :
CreationTime     : 01/07/2022 13:59:47
ModificationTime : 01/07/2022 14:03:30
UserVersion      :
ComputerVersion  :
WmiFilter        :
```

RESKITJERRYG

File | C:/Foo/GPOReport.html

No data available.

User Details

hide

General

hide

User name

RESKITJERRYG

Domain

Reskit.Org

Organizational Unit

Reskit.Org/IT

Security Group Membership

show

Component Status

show

Settings

hide

Policies

hide

Administrative Templates

hide

Policy definitions (ADMX files) retrieved from the local computer.

PowerShell Core

hide

Policy	Setting	Winning GPO
Turn on Module Logging	Enabled	PowerShell GPO for IT
Use Windows PowerShell Policy Setting. <div>             To turn on logging for one or more modules, click Show, and then type the module names in the list. Wildcards are supported.             <div>               Module Names               <div>ITModule1</div> <div>ITModule2</div> </div> </div>		
To turn on logging for the PowerShell Core core modules, type the following module names in the list: <div>             Microsoft.PowerShell.*             Microsoft.WSMan.Management           </div>		
Turn on PowerShell Script Block Logging	Enabled	PowerShell GPO for IT
Use Windows PowerShell Policy setting. <div>Log script block invocation start / stop events:</div>		
Turn on Script Execution	Enabled	PowerShell GPO for IT
Use Windows PowerShell Policy Setting. <div>             Execution Policy             <div>Allow all scripts</div> </div>		

Extra Registry Settings

hide

Display names for some settings cannot be found. You might be able to resolve this issue by updating the .ADM files used by Group Policy Management.

Setting	State	Winning GPO
Software\Policies\Microsoft\Windows\Control Panel\Desktop\ScreenSaveTimeOut	900	Screen Saver Time Out

Group Policy Objects

show

WMI Filters

show

```
PS C:\Foo> # 3. Examining the PowerShell Core event log for 4104 events
PS C:\Foo> Get-Winevent -LogName 'PowerShellCore/Operational' |
Where-Object Id -eq 4104
```

ProviderName: PowerShellCore

TimeCreated	Id	LevelDisplayName	Message
10/07/2022 15:39:03	4104	Warning	Creating Scriptblock text (1 of 1):...
10/07/2022 15:39:03	4104	Warning	Creating Scriptblock text (1 of 1):...
10/07/2022 15:37:21	4104	Warning	Creating Scriptblock text (1 of 1):...
10/07/2022 15:37:21	4104	Warning	Creating Scriptblock text (1 of 1):...

```

PS C:\Foo> # 4. Examining logged event details
PS C:\Foo> Get-Winevent -LogName 'PowerShellCore/Operational' |
    Where-Object Id -eq 4104 |
    Select-Object -First 1 |
    Format-List -Property ID, Logname, Message

Id      : 4104
LogName : PowerShellCore/Operational
Message : Creating Scriptblock text (1 of 1):
        Set-ItemProperty $SBLPath -Name EnableScriptBlockLogging -Value '1'

ScriptBlock ID: 0ea65eb3-e6b5-4b36-9ceb-d8076b9e80a4

```

```

PS C:\Foo> # 6. Comparing the events before and after you invoke the command
PS C:\Foo> "Before: $($Before.Count) events"
Before: 4 events
PS C:\Foo> "After : $($After.Count) events"
After : 4 events

```

```

PS C:\Foo> # 1. Discovering the current domain password policy
PS C:\Foo> Get-ADDefaultDomainPasswordPolicy -

ComplexityEnabled           : True
DistinguishedName           : DC=Reskit,DC=Org
LockoutDuration              : 00:30:00
LockoutObservationWindow    : 00:30:00
LockoutThreshold             : 0
MaxPasswordAge               : 42.00:00:00
MinPasswordAge               : 1.00:00:00
MinPasswordLength            : 7
objectClass                  : {domainDNS}
objectGuid                   : ceb54656-2e25-4bec-b0f1-1c562e03230e
PasswordHistoryCount         : 24
ReversibleEncryptionEnabled : False

```

```

PS C:\Foo> # 2. Discovering if there is a fine-grained password policy for JerryG
PS C:\Foo> Get-ADFineGrainedPasswordPolicy -Identity 'JerryG'
Get-ADFineGrainedPasswordPolicy: Cannot find an object with identity: 'JerryG' under: 'DC=Reskit,DC=Org'.

```



```
PS C:\Foo> # 4. Checking updated default password policy
PS C:\Foo> Get-ADDefaultDomainPasswordPolicy
```

```
ComplexityEnabled           : True
DistinguishedName           : DC=Reskit,DC=Org
LockoutDuration              : 00:45:00
LockoutObservationWindow    : 00:30:00
LockoutThreshold             : 0
MaxPasswordAge               : 42.00:00:00
MinPasswordAge               : 1.00:00:00
MinPasswordLength            : 6
objectClass                  : {domainDNS}
objectGuid                   : ceb54656-2e25-4bec-b0f1-1c562e03230e
PasswordHistoryCount         : 24
ReversibleEncryptionEnabled  : False
```

```
PS C:\Foo> # 8. Checking on policy application for the group
PS C:\Foo> Get-ADGroup 'DNSAdmins' -Properties * |
    Select-Object -Property msDS-PSOApplied
```

```
msDS-PSOApplied
```

```
-----
{CN=DNSPWP,CN=Password Settings Container,CN=System,DC=Reskit,DC=Org}
```

```
PS C:\Foo> # 9. Checking on policy application for the user
PS C:\Foo> Get-ADUser JerryG -Properties * |
    Select-Object -Property msDS-PSOApplied
```

```
msDS-PSOApplied
```

```
-----
{CN=DNSPWP,CN=Password Settings Container,CN=System,DC=ReskitIDC=Org}
```

```
PS C:\Foo> # 10. Getting DNS Admins policy
PS C:\Foo> Get-ADFineGrainedPasswordPolicy -Identity DNSPWP
```

```
AppliesTo           : {CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org, CN=DnsAdmins,CN=Users,DC=Reskit,DC=Org}
ComplexityEnabled   : True
DistinguishedName    : CN=DNSPWP,CN=Password Settings Container,CN=System,DC=Reskit,DC=Org
LockoutDuration      : 12:00:00
LockoutObservationWindow : 00:42:00
LockoutThreshold     : 3
MaxPasswordAge       : 42.00:00:00
MinPasswordAge       : 1.00:00:00
MinPasswordLength    : 7
Name                 : DNSPWP
ObjectClass           : msDS-PasswordSettings
ObjectGUID            : f75e3310-6526-4957-b697-550c749b9717
PasswordHistoryCount : 24
Precedence            : 500
ReversibleEncryptionEnabled : True
```

```
PS C:\Foo> # 11. Checking on JerryG's resultant password policy
PS C:\Foo> Get-ADUserResultantPasswordPolicy -Identity JerryG
```

```
AppliesTo           : {CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org, CN=DnsAdmins,CN=Users,DC=Reskit,DC=Org}
ComplexityEnabled   : True
DistinguishedName    : CN=DNSPWP,CN=Password Settings Container,CN=System,DC=Reskit,DC=Org
LockoutDuration      : 12:00:00
LockoutObservationWindow : 00:42:00
LockoutThreshold     : 3
MaxPasswordAge       : 42.00:00:00
MinPasswordAge       : 1.00:00:00
MinPasswordLength    : 7
Name                 : DNSPWP
ObjectClass           : msDS-PasswordSettings
ObjectGUID            : f75e3310-6526-4957-b697-550c749b9717
PasswordHistoryCount : 24
Precedence            : 500
ReversibleEncryptionEnabled : True
```

```
PS C:\Foo> # 2. Discovering the cmdlets in the Defender module
PS C:\Foo> Import-Module -Name Defender
```

WARNING: Module Defender is loaded in Windows PowerShell using WinPSCompatSession remoting session; please note that all input and output of commands from this module will be deserialized objects. If you want to load this module into PowerShell please use 'Import-Module -SkipEditionCheck' syntax.

```
PS C:\Foo> Get-Command -Module Defender
```

CommandType	Name	Version	Source
Function	Add-MpPreference	1.0	Defender
Function	Get-MpComputerStatus	1.0	Defender
Function	Get-MpPreference	1.0	Defender
Function	Get-MpThreat	1.0	Defender
Function	Get-MpThreatCatalog	1.0	Defender
Function	Get-MpThreatDetection	1.0	Defender
Function	Remove-MpPreference	1.0	Defender
Function	Remove-MpThreat	1.0	Defender
Function	Set-MpPreference	1.0	Defender
Function	Start-MpScan	1.0	Defender
Function	Start-MpWDOScan	1.0	Defender
Function	Update-MpSignature	1.0	Defender

```
PS C:\Foo> # 3. Checking the Defender service status
PS C:\Foo> Get-Service -Name WinDefend
```

Status	Name	DisplayName
Running	WinDefend	Microsoft Defender Antivirus Service

```
PS C:\Foo> # 4. Checking the operational status of Defender on this host
PS C:\Foo> Get-MpComputerStatus
```

```
RunspaceId           : 130ecf76-9081-44e3-86ef-993b5fa3d9a1
AMEngineVersion       : 1.1.19300.2
AMProductVersion      : 4.18.2205.7
AMRunningMode         : Normal
AMServiceEnabled      : True
AMServiceVersion      : 4.18.2205.7
AntispywareEnabled    : True
AntispywareSignatureAge : 0
AntispywareSignatureLastUpdated : 11/07/2022 05:27:53
AntispywareSignatureVersion : 1.369.1143.0
AntivirusEnabled      : True
AntivirusSignatureAge : 0
AntivirusSignatureLastUpdated : 11/07/2022 05:27:53
AntivirusSignatureVersion : 1.369.1143.0
BehaviorMonitorEnabled : True
ComputerID            : CA7655A0-BFDF-4F85-B366-5CB7BA2D982B
ComputerState         : 0
DefenderSignaturesOutOfDate : False
DeviceControlDefaultEnforcement : Unknown
DeviceControlPoliciesLastUpdated : 11/07/2022 14:39:15
DeviceControlState    : Disabled
FullScanAge           : 4294967295
FullScanEndTime       : 
FullScanOverdue       : False
FullScanRequired      : False
FullScanSignatureVersion : 
FullScanStartTime     : 
IoavProtectionEnabled : True
IsTamperProtected     : False
IsVirtualMachine      : True
LastFullScanSource    : 0
LastQuickScanSource   : 2
NISEnabled            : True
NISEngineVersion      : 1.1.19300.2
NISSignatureAge       : 0
NISSignatureLastUpdated : 11/07/2022 05:27:53
NISSignatureVersion   : 1.369.1143.0
OnAccessProtectionEnabled : True
ProductStatus         : 524288
QuickScanAge          : 0
QuickScanEndTime      : 11/07/2022 04:09:01
QuickScanOverdue      : False
QuickScanSignatureVersion : 1.369.1085.0
QuickScanStartTime    : 11/07/2022 04:06:04
RealTimeProtectionEnabled : True
RealTimeScanDirection : 0
RebootRequired        : False
TamperProtectionSource : Signatures
TDTMode               : N/A
TDTStatus             : N/A
TDTTelemetry          : N/A
TroubleShootingDailyMaxQuota : 
TroubleShootingDailyQuotaLeft : 
TroubleShootingEndTime : 
TroubleShootingExpirationLeft : 
TroubleShootingMode    : 
TroubleShootingModeSource : 
TroubleShootingQuotaResetTime : 
TroubleShootingStartTime : 
```



```

PS C:\Foo> # 5. Getting and counting threat catalog
PS C:\Foo> $ThreatCatalog = Get-MpThreatCatalog
PS C:\Foo> "There are $($ThreatCatalog.count) threats in the catalog"
There are 234010 threats in the catalog

```

```

PS C:\Foo> # 6. Viewing five threats in the catalog
PS C:\Foo> $ThreatCatalog |
    Select-Object -First 5 |
    Format-Table -Property SeverityID, ThreatID, ThreatName

```

SeverityID	ThreatID	ThreatName
5	1605	Dialer:Win32/Aconti
5	1622	MonitoringTool:Win32/ActiveKeylogger
5	1624	Dialer:Win32/ActiveStripPlayer
5	1625	MonitoringTool:Win32/ActivityXCustomControl
5	1626	MonitoringTool:Win32/ActivityMonitor

```

PS C:\Foo> # 8. Creating a false positive threat
PS C:\Foo> $TF = 'C:\Foo\FalsePositive1.Txt'
PS C:\Foo> $FP = 'X50!P%AP[4\PZX54(P^)7CC)7}$EICAR-' +
    'STANDARD-ANTIVIRUS-TEST-FILE!$H+H*'
PS C:\Foo> $FP | Out-File -FilePath $TF
PS C:\Foo> Get-Content -Path $TF
Get-Content: Operation did not complete successfully because the file contains
a virus or potentially unwanted software. : 'C:\Foo\FalsePositive1.Txt'

```

```

PS C:\Foo> # 10. Viewing detected threats
PS C:\Foo> Get-MpThreat

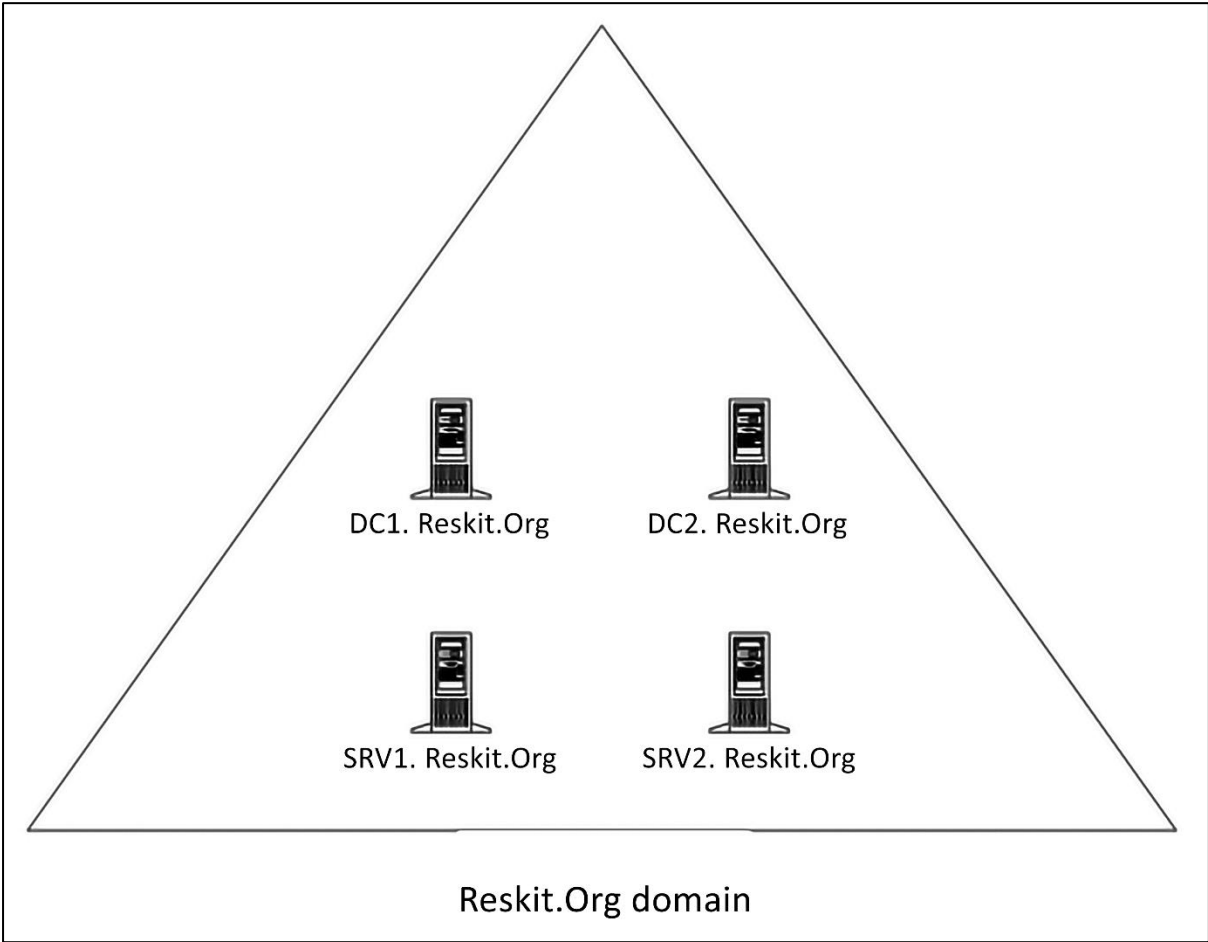
```

```

RunspaceId      : 130ecf76-9081-44e3-86ef-993b5fa3d9a1
CategoryID      : 42
DidThreatExecute : False
IsActive        : False
Resources       :
RollupStatus    : 1
SchemaVersion   : 1.0.0.0
SeverityID      : 5
ThreatID        : 2147519003
ThreatName      : Virus:DOS/EICAR_Test_File
TypeID          : 0

```

# Chapter 7: Managing Storage



```
PS C:\Foo> # 1. Displaying the disks on SRV1
PS C:\Foo> Get-Disk
```

Number	Friendly Name	Serial Number	HealthStatus	OperationalStatus	Total Size	Partition Style
0	Msft Virtual Disk		Healthy	Online	128 GB	GPT
1	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
2	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
3	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
4	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
5	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
6	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
7	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
8	Msft Virtual Disk		Healthy	Offline	64 GB	RAW

```

PS C:\Foo> # 2. Get first usable disk
PS C:\Foo> $Disk = Get-Disk |
    Where-Object PartitionStyle -eq Raw |
    Select-Object -First 1
PS C:\Foo> $Disk | Format-List

UniqueId       : 60022480E8258902AEAD5E128062318A
Number         : 1
Path           : \\?\scsi#disk&ven_msft&prod_virtual_disk#
                5&2132ca1&0&000000#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
Manufacturer   : Msft
Model          : Virtual Disk
SerialNumber    :
Size           : 64 GB
AllocatedSize   : 0
LogicalSectorSize : 512
PhysicalSectorSize : 4096
NumberOfPartitions : 0
PartitionStyle  : RAW
IsReadOnly      : True
IsSystem        : False
IsBoot          : False

```

```

PS C:\Foo> # 4. Re-displaying all disks in SRV1
PS C:\Foo> Get-Disk

```

Number	Friendly Name	Serial Number	HealthStatus	OperationalStatus	Total Size	Partition Style
0	Msft Virtual Disk		Healthy	Online	128 GB	GPT
1	Msft Virtual Disk		Healthy	Online	64 GB	GPT
2	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
3	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
4	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
5	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
6	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
7	Msft Virtual Disk		Healthy	Offline	64 GB	RAW
8	Msft Virtual Disk		Healthy	Offline	64 GB	RAW

```

PS C:\Foo> # 5. Viewing volumes on SRV1
PS C:\Foo> Get-Volume | Sort-Object -Property DriveLetter

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
		FAT32	Fixed	Healthy	OK	67.25 MB	96 MB
C		NTFS	Fixed	Healthy	OK	111.25 GB	127.9 GB

```
PS C:\Foo> # 6. Viewing partitions on SRV1
PS C:\Foo> Get-Partition
```

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#5&1ebf9ebb&0&000000#
{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

PartitionNumber	DriveLetter	Offset	Size	Type
1	C	1048576	127.9 GB	Basic
2		137333047296	100 MB	System

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000000#
{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

PartitionNumber	DriveLetter	Offset	Size	Type
1		17408	15.98 MB	Reserved

```
PS C:\Foo> # 7. Examining details of a volume
PS C:\Foo> Get-Volume | Select-Object -First 1 | Format-List
```

```
ObjectId           : {1}\\SRV1\root\Microsoft\Windows\Storage\Providers_v2\
WSP_Volume.ObjectId="{004bc44d-ed2-11ec-a79c-806e6f6e6963}":
V0:\\?\Volume{ebeaff83-af69-11ec-be24-5cf37091be18}\
PassThroughClass    :
PassThroughIds       :
PassThroughNamespace :
PassThroughServer    :
UniqueId            : \\?\Volume{ebeaff83-af69-11ec-be24-5cf37091be18}\
AllocationUnitSize   : 4096
DedupMode            : NotAvailable
DriveLetter          : C
DriveType            : Fixed
FileSystem            : NTFS
FileSystemLabel       :
FileSystemType        : NTFS
HealthStatus         : Healthy
OperationalStatus    : OK
Path                 : \\?\Volume{ebeaff83-af69-11ec-be24-5cf37091be18}\
Size                 : 137331994624
SizeRemaining        : 119453777920
PSComputerName       :
```



```

PS C:\Foo> # 8. Examining details of a partition
PS C:\Foo> Get-Partition | Select-Object -First 1 | Format-List

UniqueId          : {00000000-0000-0000-0000-100000000000}6002248005588E716BE40737CCDCDD7C
AccessPaths       : {C:\, \\?\Volume{ebeaff83-af69-11ec-be24-5cf37091be18}\}
DiskNumber        : 0
DiskPath          : \\?\scsi#disk&ven_msft&prod_virtual_disk#5&1ebf9ebb&0&000000#
                  {53f56307-b6bf-11d0-94f2-00a0c91efb8b}
DriveLetter       : C
Guid              : {ebeaff83-af69-11ec-be24-5cf37091be18}
IsActive          : False
IsBoot            : True
IsHidden          : False
IsOffline         : False
IsReadOnly        : False
IsShadowCopy      : False
IsDAX             : False
IsSystem          : False
NoDefaultDriveLetter : False
Offset            : 1048576
OperationalStatus : Online
PartitionNumber   : 1
Size              : 127.9 GB
Type              : Basic

```

```

PS C:\Foo> # 10. Examining disks in SRV1
PS C:\Foo> Get-Disk

Number Friendly Name      Serial Number HealthStatus OperationalStatus Total Size Partition Style
-----
0       Msft Virtual Disk             Healthy      Online      128 GB GPT
1       Msft Virtual Disk             Healthy      Online      64 GB GPT
2       Msft Virtual Disk             Healthy      Online      64 GB MBR
3       Msft Virtual Disk             Healthy      Offline     64 GB RAW
4       Msft Virtual Disk             Healthy      Offline     64 GB RAW
5       Msft Virtual Disk             Healthy      Offline     64 GB RAW
6       Msft Virtual Disk             Healthy      Offline     64 GB RAW
7       Msft Virtual Disk             Healthy      Offline     64 GB RAW
8       Msft Virtual Disk             Healthy      Offline     64 GB RAW

```

```

PS C:\Foo> # 1. Getting the second disk
PS C:\Foo> $Disk = Get-Disk | Select-Object -Skip 1 -First 1
PS C:\Foo> $Disk | Format-List

UniqueId          : 60022480E8258902AEAD5E128062318A
Number            : 1
Path              : \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000000#
                  {53f56307-b6bf-11d0-94f2-00a0c91efb8b}
Manufacturer      : Msft
Model             : Virtual Disk
SerialNumber       :
Size              : 64 GB
AllocatedSize      : 17825792
LogicalSectorSize  : 512
PhysicalSectorSize : 4096
NumberOfPartitions : 1
PartitionStyle     : GPT
IsReadOnly        : False
IsSystem          : False
IsBoot            : False

```

```

PS C:\Foo> # 2. Creating a new volume in this disk
PS C:\Foo> $NewVolumeHT1 = @{
    DiskNumber = $Disk.Disknumber
    DriveLetter = 'S'
    FriendlyName = 'Files'
}
PS C:\Foo> New-Volume @NewVolumeHT1

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
S	Files	NTFS	Fixed	Healthy	OK	63.84 GB	63.98 GB

```

PS C:\Foo> # 3. Getting next available disk to use on SRV1
PS C:\Foo> $Disk2 = Get-Disk |
    Where-Object PartitionStyle -eq 'MBR' |
    Select-Object -First 1
PS C:\Foo> $Disk2 | Format-List

```

```

UniqueId          : 60022480F74243AB1AD9CFB6E1B06E28
Number            : 2
Path              : \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000001#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
Manufacturer      : Msft
Model             : Virtual Disk
SerialNumber      :
Size              : 64 GB
AllocatedSize     : 2097152
LogicalSectorSize : 512
PhysicalSectorSize : 4096
NumberOfPartitions : 0
PartitionStyle    : MBR
IsReadOnly        : False
IsSystem          : False
IsBoot            : False

```

```
PS C:\Foo> # 4. Creating 4 new partitions on third (MBR) disk
PS C:\Foo> $UseMaxHT= @{UseMaximumSize = $true}
PS C:\Foo> New-Partition -DiskNumber $Disk2.DiskNumber -DriveLetter W -Size 1gb
```

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000001#
{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

PartitionNumber	DriveLetter	Offset	Size	Type
1	W	1048576	1 GB	Logical

```
PS C:\Foo> New-Partition -DiskNumber $Disk2.DiskNumber -DriveLetter X -Size 15gb
```

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000001#
{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

PartitionNumber	DriveLetter	Offset	Size	Type
2	X	1074790400	15 GB	Logical

```
PS C:\Foo> New-Partition -DiskNumber $Disk2.DiskNumber -DriveLetter Y -Size 15gb
```

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000001#
{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

PartitionNumber	DriveLetter	Offset	Size	Type
3	Y	17180917760	15 GB	Logical

```
PS C:\Foo> New-Partition -DiskNumber $Disk2.DiskNumber -DriveLetter Z @UseMaxHT
```

```
DiskPath: \\?\scsi#disk&ven_msft&prod_virtual_disk#5&2132ca1&0&000001#
{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

PartitionNumber	DriveLetter	Offset	Size	Type
4	Z	33288093696	33 GB	Logical



```

PS C:\Foo> # 5. Formatting each volume on this disk
PS C:\Foo> $FormatHT1 = @{
    DriveLetter      = 'W'
    FileSystem        = 'FAT'
    NewFileSystemLabel = 'w-fat'
}
PS C:\Foo> Format-Volume @FormatHT1

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
W	W-FAT	FAT	Fixed	Healthy	OK	1023.69 MB	1023.72 MB

```

PS C:\Foo> $FormatHT2 = @{
    DriveLetter      = 'X'
    FileSystem        = 'exFAT'
    NewFileSystemLabel = 'x-exFAT'
}
PS C:\Foo> Format-Volume @FormatHT2

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
X	x-exFAT	exFAT	Fixed	Healthy	OK	15 GB	15 GB

```

PS C:\Foo> $FormatHT3 = @{
    DriveLetter      = 'Y'
    FileSystem        = 'FAT32'
    NewFileSystemLabel = 'Y-FAT32'
}
PS C:\Foo> Format-Volume @FormatHT3

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
Y	Y-FAT32	FAT32	Fixed	Healthy	OK	14.98 GB	14.98 GB

```

PS C:\Foo> $FormatHT4 = @{
    DriveLetter      = 'Z'
    FileSystem        = 'ReFS'
    NewFileSystemLabel = 'Z-ReFS'
}
PS C:\Foo> Format-Volume @FormatHT4

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
Z	Z-ReFS	ReFS	Fixed	Healthy	OK	31.81 GB	32.94 GB

```

PS C:\Foo> # 7. Getting all volumes on SRV1
PS C:\Foo> Get-Volume | Sort-Object DriveLetter

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
C		FAT32	Fixed	Healthy	OK	67.25 MB	96 MB
S	Files	NTFS	Fixed	Healthy	OK	111.23 GB	127.9 GB
W	W-FAT	FAT	Fixed	Healthy	OK	63.84 GB	63.98 GB
X	x-exFAT	exFAT	Fixed	Healthy	OK	1023.69 MB	1023.72 MB
Y	Y-FAT32	FAT32	Fixed	Healthy	OK	15 GB	15 GB
Z	Z-ReFS	ReFS	Fixed	Healthy	OK	14.98 GB	14.98 GB
						31.81 GB	32.94 GB

```

PS C:\Foo> # 1. Getting PowerShell providers
PS C:\Foo> Get-PSPProvider

```

Name	Capabilities	Drives
Registry	ShouldProcess	{HKLM, HKCU}
Alias	ShouldProcess	{Alias}
Environment	ShouldProcess	{Env}
FileSystem	Filter, ShouldProcess, Credentials	{C, Temp, S, W, X, Y, Z}
Function	ShouldProcess	{Function}
Variable	ShouldProcess	{Variable}
Certificate	ShouldProcess	{Cert}
WSMan	Credentials	{WSMan}



```
PS C:\Foo> # 2. Getting drives from the registry provider
PS C:\Foo> Get-PSDrive | Where-Object Provider -match 'Registry'
```

Name	Used (GB)	Free (GB)	Provider	Root	CurrentLocation
HKCU			Registry	HKEY_CURRENT_USER	
HKLM			Registry	HKEY_LOCAL_MACHINE	

```
PS C:\Foo> # 4. Getting registered owner
PS C:\Foo> (Get-ItemProperty -Path $Path -Name RegisteredOwner).RegisteredOwner
Book Readers
```

```
PS C:\Foo> # 5. Counting aliases in the Alias: drive
PS C:\Foo> Get-Item Alias:* | Measure-Object
```

```
Count           : 164
Average          :
Sum              :
Maximum          :
Minimum          :
StandardDeviation :
Property         :
```

```
PS C:\Foo> # 6. Finding aliases for Remove-Item
PS C:\Foo> Get-Childitem Alias:* |
Where-Object ResolvedCommand -match 'Remove-Item$'
```

CommandType	Name	Version	Source
Alias	ri -> Remove-Item		
Alias	rm -> Remove-Item		
Alias	rmdir -> Remove-Item		
Alias	del -> Remove-Item		
Alias	erase -> Remove-Item		
Alias	rd -> Remove-Item		

```
PS C:\Foo> # 7. Counting environment variables on SRV1
PS C:\Foo> Get-Item ENV:* | Measure-Object
```

```
Count           : 45
Average          :
Sum              :
Maximum          :
Minimum          :
StandardDeviation :
Property         :
```

```
PS C:\Foo> # 8. Displaying Windows installation folder
PS C:\Foo> "Windows installation folder is [Senv:windir]"
Windows installation folder is [C:\WINDOWS]
```

```
PS C:\Foo> # 9. Checking on FileSystem provider drives on SRV1
PS C:\Foo> Get-PSPProvider -PSPProvider FileSystem |
Select-Object -ExpandProperty Drives |
Sort-Object -Property Name
```

Name	Used (GB)	Free (GB)	Provider	Root	CurrentLocation
C	16.68	111.22	FileSystem	C:\	Foo
S	0.14	63.84	FileSystem	S:\	
Temp	16.68	111.22	FileSystem	C:\Users\administrator.RESKIT\AppData\Local\Temp\	
W	0.00	1.00	FileSystem	W:\	
X	0.00	15.00	FileSystem	X:\	
Y	0.00	14.98	FileSystem	Y:\	
Z	1.16	31.78	FileSystem	Z:\	

```
PS C:\Foo> # 10. Getting home folder for FileSystem provider
PS C:\Foo> $HomeFolder = Get-PSPProvider -PSPProvider FileSystem |
Select-Object -ExpandProperty Home
PS C:\Foo> $HomeFolder
C:\Foo
```

```
PS C:\Foo> # 11. Checking Function drive
PS C:\Foo> Get-Module | Remove-Module -WarningAction SilentlyContinue
PS C:\Foo> $Functions = Get-ChildItem -Path Function:
PS C:\Foo> "Functions available [ $($Functions.Count) ]"
Functions available [36]
```

```
PS C:\Foo> # 13. Checking Function drive again
PS C:\Foo> $Functions2 = Get-ChildItem -Path Function:
PS C:\Foo> "Functions now available [ $($Functions2.Count) ]"
Functions now available [37]
```

```
PS C:\Foo> # 14. Counting defined variables
PS C:\Foo> $Variables = Get-ChildItem -Path Variable:
PS C:\Foo> "Variables defined [ $C$Variables.count ]"
Variables defined [70]
```

```
PS C:\Foo> # 15. Counting defined variables
PS C:\Foo> $Variables = Get-ChildItem -Path Variable:
PS C:\Foo> "Variables defined [ $($Variables.count) ]"
Variables defined [70]
```

```
PS C:\Foo> # 16. Getting trusted root certificates for the local machine
PS C:\Foo> Get-ChildItem -Path Cert:\LocalMachine\Root |
Format-Table FriendlyName, Thumbprint
```

FriendlyName	Thumbprint
Microsoft Root Certificate Authority	CDD4EEAE6000AC7F40C3802C171E30148030C072
Thawte Timestamping CA	BE36A4562FB2EE05DBB3D32323ADF445084ED656
Microsoft Root Authority	A43489159A520F0D93D032CCAF37E7FE20A8B419
	92B46C76E13054E104F230517E6E504D43AB10B5
Microsoft Root Certificate Authority 2011	8F43288AD272F3103B6FB1428485EA3014C0BCFE
Microsoft Authenticode(tm) Root	7F88CD7223F3C813818C994614A89C99FA3B5247
Microsoft Root Certificate Authority 2010	3B1EFD3A66EA28B16697394703A72CA340A05BD5
Microsoft ECC TS Root Certificate Authority 2018	31F9FC8BA3805986B721EA7295C65B3A44534274
Microsoft Timestamp Root	245C97DF7514E7CF2DF8BE72AE957B9E04741E85
VeriSign Time Stamping CA	18F7C1FCC3090203FD5BAA2F861A754976C8DD25
Microsoft ECC Product Root Certificate Authority 2018	06F1AA330B927B753A40E68CDF22E34BCBEF3352
Microsoft Time Stamp Root Certificate Authority 2014	0119E81BE9A14CD8E22F40AC118C687ECBA3F4D8
DigiCert Global Root G2	DF3C24F9BFD666761B268073FE06D1CC8D4F82A4
DST Root CA X3	DAC9024F54D8F6DF94935FB1732638CA6AD77C13
GlobalSign Root CA - R3	D69B561148F01C77C54578C10926DF5B856976AD
DigiCert Baltimore Root	D4DE20D05E66FC53FE1A50882C78DB2852CAE474
Sectigo (AAA)	D1EB23A46D17D68FD92564C2F1F1601764D8E349
ISRG Root X1	CABD2A79A1076A31F21D253635CB039D4329A5E8
GlobalSign Root CA - R1	B1BC968BD4F49D622AA89A81F2150152A41D829C
Starfield Class 2 Certification Authority	AD7E1C28B064EF8F6003402014C3D0E3370EB58A
DigiCert	A8985D3A65E5E5C4B2D7D66D040C6DD2FB19C5436
Entrust.net	8CF427FD790C3AD166068DE81E57EFBB932272D4
VeriSign Class 3 Public Primary CA	742C3192E607E424EB4549542BE1BBC53E6174E2
DigiCert	5FB7EE0633E259DBAD0C4C9AE6D38F1A61C7DC25
VeriSign	4EB6D578499B1CCF5F581EAD56BE3D986744A5E5
VeriSign Universal Root Certification Authority	3679CA35668772304D30A5FB873B0FA77BB70D54
Go Daddy Class 2 Certification Authority	2796BAE63F1801E277261BA0D77770028F20EEE4
QuoVadis Root CA 2 G3	093C61F38B8BDC7D55DF7538020500E125F5C836
DigiCert	0563B8630D62D75ABBC8AB1E4BDFB5A899B24D43

```
PS C:\Foo> # 17. Examining ports in use by WinRM
PS C:\Foo> Get-ChildItem -Path WSMan:\localhost\Client\DefaultPorts
```

WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Client\DefaultPorts

Type	Name	SourceOfValue	Value
System.String	HTTP		5985
System.String	HTTPS		5986

```
PS C:\Foo> # 21. Importing the CimPSDrive module and creating a drive
PS C:\Foo> Import-Module -Name CimPSDrive
PS C:\Foo> New-PSDrive -Name CIM -PSProvider SHiPS -Root CIMPSDrive#CMRoot
```

Name	Used (GB)	Free (GB)	Provider	Root	CurrentLocation
CIM			SHiPS	CIMPSDrive#CMRoot	



```
PS C:\Foo> # 22. Examining BIOS using the CimPSDrive module
PS C:\Foo> Get-ChildItem CIM:\localhost\CIMV2\Win32_Bios
```

```
SMBIOSBIOSVersion : Hyper-V UEFI Release v4.1
Manufacturer      : Microsoft Corporation
Name              : Hyper-V UEFI Release v4.1
SerialNumber      : 1292-4012-4928-5640-9420-3487-94
Version           : VIRTUAL - 1
PSComputerName    : localhost
```

```
PS C:\Foo> # 1. Getting disk number of the disk holding the S partition
PS C:\Foo> $Part = Get-Partition -DriveLetter S
PS C:\Foo> "S drive on disk [$C$Part.DiskNumber]"
S drive on disk [1]
```

```
PS C:\Foo> # 2. Creating S: drive on SRV2
PS C:\Foo> $ScriptBlock = {
    Initialize-Disk -Number $using:Part.DiskNumber -PartitionStyle GPT
    $NewVolHT = @{
        DiskNumber = $using:Part.DiskNumber
        FriendlyName = 'Files'
        FileSystem = 'NTFS'
        DriveLetter = 'S'
    }
    New-Volume @NewVolHT
}
PS C:\Foo> Invoke-Command -ComputerName SRV2 -ScriptBlock $ScriptBlock
```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size	PSComputerName
S	Files	NTFS	Fixed	Healthy	OK	63.89 GB	63.98 GB	SRV2

```
PS C:\Foo> # 4. Counting files/folders on S:
PS C:\Foo> Get-ChildItem -Path S:\ -Recurse | Measure-Object
```

```
Count           : 10100
Average         :
Sum             :
Maximum         :
Minimum         :
StandardDeviation :
Property        :
```



```

PS C:\Foo> # 5. Examining the same drives remotely on SRV2
PS C:\Foo> $ScriptBlock2 = {
    Get-ChildItem -Path S:\ -Recurse |
    Measure-Object
}
PS C:\Foo> Invoke-Command -ComputerName SRV2 -ScriptBlock $ScriptBlock2

```

```

Count          : 0
Average        :
Sum            :
Maximum        :
Minimum        :
Property       :
PSComputerName : SRV2

```

```

PS C:\Foo> # 6. Adding the storage replica feature to SRV1
PS C:\Foo> Add-WindowsFeature -Name Storage-Replica | Out-Null
WARNING: You must restart this server to finish the installation process

```

```

PS C:\Foo> # 7. Adding the Storage Replica Feature to SRV2
PS C:\Foo> $SB= {
    Add-WindowsFeature -Name Storage-Replica | Out-Null
}
PS C:\Foo> Invoke-Command -ComputerName SRV2 -ScriptBlock $SB

```

WARNING: You must restart this server to finish the installation process.

```

PS C:\Foo> # 10. Creating a G: volume in disk 3 on SRV1
PS C:\Foo> $ScriptBlock3 = {
    Initialize-Disk -Number 3 | Out-Null
    $VolumeHT = @{
        DiskNumber = 3
        FriendlyName = 'SRLOGS'
        DriveLetter = 'G'
    }
    New-Volume @VolumeHT
}
PS C:\Foo> Invoke-Command -ComputerName SRV1 -ScriptBlock $ScriptBlock3

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size	PSComputerName
G	SRLOGS	NTFS	Fixed	Healthy	OK	63.84 GB	63.98 GB	SRV1

```

PS C:\Foo> # 11. Creating G: volume on SRV2
PS C:\Foo> Invoke-Command -ComputerName SRV2 -ScriptBlock $ScriptBlock3

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size	PSComputerName
G	SRLOGS	NTFS	Fixed	Healthy	OK	63.89 GB	63.98 GB	SRV2

```

PS C:\Foo> # 12. Viewing volumes on SRV1
PS C:\Foo> Get-Volume | Sort-Object -Property DriveLetter

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
		FAT32	Fixed	Healthy	OK	67.25 MB	96 MB
C		NTFS	Fixed	Healthy	OK	111.61 GB	127.9 GB
G	SRLOGS	NTFS	Fixed	Healthy	OK	63.84 GB	63.98 GB
S	Files	NTFS	Fixed	Healthy	OK	63.83 GB	63.98 GB
W	W-FAT	FAT	Fixed	Healthy	OK	1023.66 MB	1023.72 MB
X	x-exFAT	exFAT	Fixed	Healthy	OK	15 GB	15 GB
Y	Y-FAT32	FAT32	Fixed	Healthy	OK	14.98 GB	14.98 GB
Z	Z-ReFS	ReFS	Fixed	Healthy	OK	31.78 GB	32.94 GB

```
PS C:\Foo> # 13. Viewing volumes on SRV2
PS C:\Foo> Invoke-Command -Computer SRV2 -Scriptblock {
    Get-Volume | Sort-Object -Property DriveLetter
}
```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size	PSComputerName
C		FAT32	Fixed	Healthy	OK	67.26 MB	96 MB	SRV2
G	SRLOGS	NTFS	Fixed	Healthy	OK	115.03 GB	127.9 GB	SRV2
S	Files	NTFS	Fixed	Healthy	OK	63.89 GB	63.98 GB	SRV2

```
PS C:\Foo> # 14. Creating an SR replica group
PS C:\Foo> $NewSRHT = @{
```

```
    SourceComputerName      = 'SRV1'
    SourceRGName             = 'SRV1RG2'
    SourceVolumeName         = 'S:'
    SourceLogVolumeName      = 'G:'
    DestinationComputerName  = 'SRV2'
    DestinationRGName        = 'SRV2RG2'
    DestinationVolumeName    = 'S:'
    DestinationLogVolumeName = 'G:'
    LogSizeInBytes           = 2gb
}
```

```
PS C:\Foo> New-SRPartnership @SRHT
```

```
RunspaceId      : 0da18bca-0a49-4f1a-9bbf-cdd44129984c
DestinationComputerName : SRV2
DestinationRGName : SRV2RG2
Id              : 5c52459b-38a9-44ae-8ebd-867d6e19ece0
SourceComputerName : SRV1
SourceRGName      : SRV1RG2
```

```
PS C:\Foo> # 15. Examining the volumes on SRV2
PS C:\Foo> $ScriptBlock3 = {
    Get-Volume |
    Sort-Object -Property DriveLetter |
    Format-Table
}
PS C:\Foo> Invoke-Command -ComputerName SRV2 -ScriptBlock $ScriptBlock3
```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
C		FAT32	Fixed	Healthy	OK	67.26 MB	96 MB
G	SRLOGS	NTFS	Fixed	Healthy	OK	115.03 GB	127.9 GB
S		Unknown	Fixed	Healthy	Unknown	61.89 GB	63.98 GB

```
PS C:\Foo> # 17. Viewing the SR Partnership
PS C:\Foo> Get-SRPartnership
```

```
RunspaceId      : 0da18bca-0a49-4f1a-9bbf-cdd44129984c
DestinationComputerName : SRV1
DestinationRGName : SRV1RG2
Id              : 5c52459b-38a9-44ae-8ebd-867d6e19ece0
SourceComputerName : SRV2
SourceRGName      : SRV2RG2
```

```
PS C:\Foo> # 1. Viewing disks available for pooling
PS C:\Foo> $Disks = Get-PhysicalDisk -CanPool $true
PS C:\Foo> $Disks | Sort-Object -Property Deviceid
```

Number	FriendlyName	SerialNumber	MediaType	CanPool	OperationalStatus	HealthStatus	Usage	Size
4	Msft Virtual Disk		Unspecified	True	OK	Healthy	Auto-Select	64 GB
5	Msft Virtual Disk		Unspecified	True	OK	Healthy	Auto-Select	64 GB
6	Msft Virtual Disk		Unspecified	True	OK	Healthy	Auto-Select	64 GB
7	Msft Virtual Disk		Unspecified	True	OK	Healthy	Auto-Select	64 GB
8	Msft Virtual Disk		Unspecified	True	OK	Healthy	Auto-Select	64 GB

```
PS C:\Foo> # 2. Creating a storage pool
PS C:\Foo> $NewPoolHT = @{
    FriendlyName           = 'RKSP'
    StorageSubsystemFriendlyName = "Windows Storage*"
    PhysicalDisks          = $Disks
}
PS C:\Foo> New-StoragePool @NewPoolHT
```

FriendlyName	OperationalStatus	HealthStatus	IsPrimordial	IsReadOnly	Size	AllocatedSize
RKSP	OK	Healthy	False	False	317.42 GB	1.25 GB

```
PS C:\Foo> # 3. Creating a mirrored hard disk named Mirror1
PS C:\Foo> $VDisk1HT = @{
    StoragePoolFriendlyName = 'RKSP'
    FriendlyName            = 'Mirror1'
    ResiliencySettingName   = 'Mirror'
    Size                    = 8GB
    ProvisioningType        = 'Thin'
}
PS C:\Foo> New-VirtualDisk @VDisk1HT
```

FriendlyName	ResiliencySettingName	FaultDomainRedundancy	OperationalStatus	HealthStatus	Size	FootprintOnPool	StorageEfficiency
Mirror1	Mirror	1	OK	Healthy	8 GB	1.5 GB	33.33%

```
PS C:\Foo> # 4. Creating a three way mirrored disk named Mirror2
PS C:\Foo> $VDisk2HT = @{
    StoragePoolFriendlyName = 'RKSP'
    FriendlyName            = 'Mirror2'
    ResiliencySettingName   = 'Mirror'
    NumberOfDataCopies      = 3
    Size                    = 8GB
    ProvisioningType        = 'Thin'
}
PS C:\Foo> New-VirtualDisk @VDisk2HT
```

FriendlyName	ResiliencySettingName	FaultDomainRedundancy	OperationalStatus	HealthStatus	Size	FootprintOnPool	StorageEfficiency
Mirror2	Mirror	2	OK	Healthy	8 GB	1.5 GB	16.67%

```
PS C:\Foo> # 5. Creating a volume in Mirror1
PS C:\Foo> Get-VirtualDisk -FriendlyName 'Mirror1' |
    Get-Disk |
    Initialize-Disk -PassThru |
    New-Partition -AssignDriveLetter -UseMaximumSize |
    Format-Volume
```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
D		NTFS	Fixed	Healthy	OK	7.95 GB	7.98 GB

```
PS C:\Foo> # 6. Creating a volume in Mirror2
PS C:\Foo> Get-VirtualDisk -FriendlyName 'Mirror2' |
    Get-Disk |
    Initialize-Disk -PassThru |
    New-Partition -AssignDriveLetter -UseMaximumSize |
    Format-Volume
```

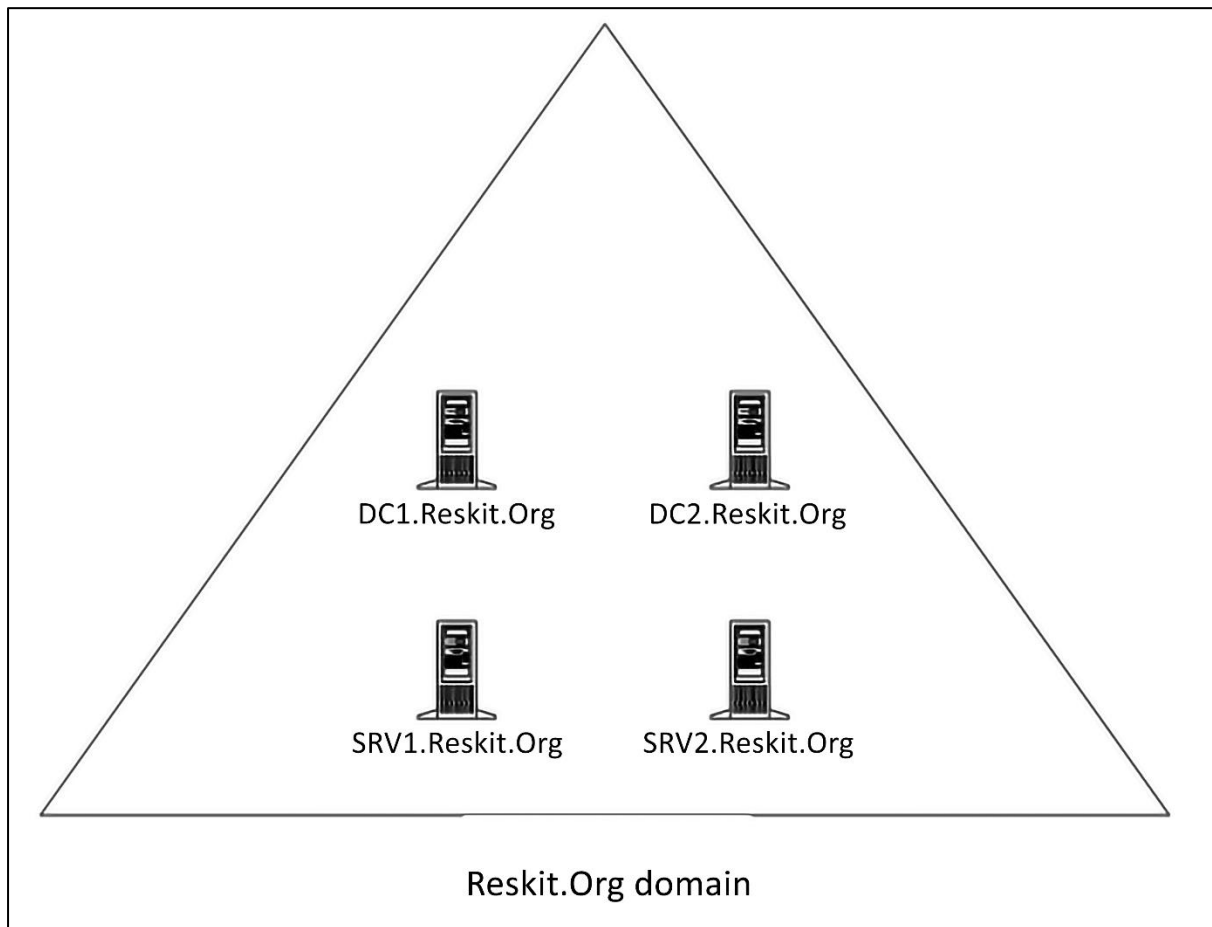
DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
E		NTFS	Fixed	Healthy	OK	7.95 GB	7.98 GB

```
PS C:\Foo> # 7. Viewing volumes on SRV1
PS C:\Foo> Get-Volume | Sort-Object -Property DriveLetter
```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
		FAT32	Fixed	Healthy	OK	67.24 MB	96 MB
C		NTFS	Fixed	Healthy	OK	108.07 GB	127.9 GB
D		NTFS	Fixed	Healthy	OK	7.95 GB	7.98 GB
E		NTFS	Fixed	Healthy	OK	7.95 GB	7.98 GB
G	SRLOGS	NTFS	Fixed	Healthy	OK	61.84 GB	63.98 GB
S		Unknown	Fixed	Healthy	Unknown	0 B	0 B
W	W-FAT	FAT	Fixed	Healthy	OK	1023.66 MB	1023.72 MB
X	x-exFAT	exFAT	Fixed	Healthy	OK	15 GB	15 GB
Y	Y-FAT32	FAT32	Fixed	Healthy	OK	14.98 GB	14.98 GB
Z	Z-ReFS	ReFS	Fixed	Healthy	OK	31.78 GB	32.94 GB



## Chapter 8: Managing Shared Data



```
PS C:\Foo> # 3. Getting commands in the module
PS C:\Foo> Get-Command -Module NTFSSecurity
```

CommandType	Name	Version	Source
Cmdlet	Add-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Add-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Clear-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Clear-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Copy-Item2	4.2.6	NTFSSecurity
Cmdlet	Disable-NTFSAccessInheritance	4.2.6	NTFSSecurity
Cmdlet	Disable-NTFSAuditInheritance	4.2.6	NTFSSecurity
Cmdlet	Disable-Privileges	4.2.6	NTFSSecurity
Cmdlet	Enable-NTFSAccessInheritance	4.2.6	NTFSSecurity
Cmdlet	Enable-NTFSAuditInheritance	4.2.6	NTFSSecurity
Cmdlet	Enable-Privileges	4.2.6	NTFSSecurity
Cmdlet	Get-ChildItem2	4.2.6	NTFSSecurity
Cmdlet	Get-DiskSpace	4.2.6	NTFSSecurity
Cmdlet	Get-FileHash2	4.2.6	NTFSSecurity
Cmdlet	Get-Item2	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSEffectiveAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSHardLink	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSInheritance	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOrphanedAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOrphanedAudit	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOwner	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSSecurityDescriptor	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSSimpleAccess	4.2.6	NTFSSecurity
Cmdlet	Get-Privileges	4.2.6	NTFSSecurity
Cmdlet	Move-Item2	4.2.6	NTFSSecurity
Cmdlet	New-NTFSHardLink	4.2.6	NTFSSecurity
Cmdlet	New-NTFSSymbolicLink	4.2.6	NTFSSecurity
Cmdlet	Remove-Item2	4.2.6	NTFSSecurity
Cmdlet	Remove-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Remove-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSInheritance	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSOwner	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSSecurityDescriptor	4.2.6	NTFSSecurity
Cmdlet	Test-Path2	4.2.6	NTFSSecurity

```

PS C:\Foo> # 4. Creating a new folder and a file in the folder
PS C:\Foo> New-Item -Path F:\Secure1 -ItemType Directory |
    Out-Null
PS C:\Foo> "Secure" | Out-File -FilePath F:\Secure1\Secure.Txt
PS C:\Foo> Get-ChildItem -Path F:\Secure1

```

Directory: F:\Secure1

Mode	LastWriteTime	Length	Name
-a---	10/08/2022	13:55	8 Secure.Txt

```

PS C:\Foo> # 5. Viewing ACL of the folder
PS C:\Foo> Get-NTFSAccess -Path F:\Secure1 |
    Format-Table -AutoSize

```

Path: F:\Secure1 (Inheritance enabled)

Account	Access Rights	Applies to	Type	IsInherited	InheritedFrom
BUILTIN\Administrators	FullControl	ThisFolderOnly	Allow	False	
BUILTIN\Administrators	FullControl	ThisFolderSubfoldersAndFiles	Allow	True	F:
NT AUTHORITY\SYSTEM	FullControl	ThisFolderSubfoldersAndFiles	Allow	True	F:
CREATOR OWNER	GenericAll	SubfoldersAndFilesOnly	Allow	True	F:
BUILTIN\Users	ReadAndExecute, Synchronize	ThisFolderSubfoldersAndFiles	Allow	True	F:
BUILTIN\Users	CreateDirectories	ThisFolderAndSubfolders	Allow	True	F:
BUILTIN\Users	CreateFiles	ThisFolderAndSubfolders	Allow	True	F:

```

PS C:\Foo> # 6. Viewing ACL of the file
PS C:\Foo> Get-NTFSAccess F:\Secure1\Secure.Txt |
    Format-Table -AutoSize

```

Path: F:\Secure1\Secure.Txt (Inheritance enabled)

Account	Access Rights	Applies to	Type	IsInherited	InheritedFrom
BUILTIN\Administrators	FullControl	ThisFolderOnly	Allow	True	F:
NT AUTHORITY\SYSTEM	FullControl	ThisFolderOnly	Allow	True	F:
BUILTIN\Users	ReadAndExecute, Synchronize	ThisFolderOnly	Allow	True	F:

```

PS C:\Foo> # 8. Displaying Sales AD Group
PS C:\Foo> Invoke-Command -ComputerName DC1 -ScriptBlock {
    Get-ADGroup -Identity Sales}

```

```

PSComputerName      : DC1
RunspaceId          : 50ee7037-cd9f-4270-b663-96cca880c708
DistinguishedName    : CN=Sales,CN=Users,DC=Reskit,DC=Org
GroupCategory        : Security
GroupScope           : Global
Name                 : Sales
ObjectClass          : group
ObjectGUID           : 7f426f57-c616-4555-8e75-d1c8a6d60a33
SamAccountName       : Sales
SID                  : S-1-5-21-140053678-4069492383-922506915-3602

```

```
PS C:\Foo> # 13. Getting ACL of the Secure1 folder
PS C:\Foo> Get-NTFSAccess -Path F:\Secure1 |
Format-Table -AutoSize
```

Path: F:\Secure1 (Inheritance disabled)

Account	Access Rights	Applies to	Type	IsInherited	InheritedFrom
BUILTIN\Administrators	FullControl	ThisFolderOnly	Allow	False	
RESKIT\Domain Admins	FullControl	ThisFolderSubfoldersAndFiles	Allow	False	
RESKIT\Sales	FullControl	ThisFolderSubfoldersAndFiles	Allow	False	

```
PS C:\Foo> # 14. Getting resulting ACL on the Secure1.Txt file
PS C:\Foo> Get-NTFSAccess -Path F:\Secure1\Secure.Txt |
Format-Table -AutoSize
```

Path: F:\Secure1\Secure.Txt (Inheritance enabled)

Account	Access Rights	Applies to	Type	IsInherited	InheritedFrom
RESKIT\Domain Admins	FullControl	ThisFolderOnly	Allow	True	F:\Secure1
RESKIT\Sales	FullControl	ThisFolderOnly	Allow	True	F:\Secure1

```
PS C:\Foo> # 1. Adding File Server features to FS1
PS C:\Foo> $Features = 'FileAndStorage-Services',
                        'File-Services',
                        'FS-FileServer',
                        'RSAT-File-Services'
PS C:\Foo> Add-WindowsFeature -Name $Features
```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	{File and iSCSI Services, File Server, Remot...



```
PS C:\Foo> # 2. Viewing the SMB server settings
PS C:\Foo> Get-SmbServerConfiguration
```

```
AnnounceComment           :
AnnounceServer             : False
AsynchronousCredits        : 512
AuditSmb1Access            : False
AutoDisconnectTimeout      : 15
AutoShareServer            : True
AutoShareWorkstation       : True
CachedOpenLimit            : 10
DisableSmbEncryptionOnSecureConnection : True
DurableHandleV2TimeoutInSeconds : 180
EnableAuthenticateUserSharing : False
EnableDownlevelTimewarp    : False
EnableForcedLogoff         : True
EnableLeasing              : True
EnableMultiChannel         : True
EnableOplocks              : True
EnableSecuritySignature    : False
EnableSMB1Protocol         : False
EnableSMB2Protocol         : True
EnableStrictNameChecking   : True
EncryptData                : False
IrpStackSize               : 15
KeepAliveTime              : 2
MaxChannelPerSession       : 32
MaxMpxCount                : 50
MaxSessionPerConnection    : 16384
MaxThreadsPerQueue         : 20
MaxWorkItems               : 1
NullSessionPipes           :
NullSessionShares          :
OplockBreakWait            : 35
PendingClientTimeoutInSeconds : 120
RejectUnencryptedAccess    : True
RequireSecuritySignature   : False
ServerHidden               : True
Smb2CreditsMax             : 8192
```

```
PS C:\Foo> # 1. Discovering existing shares and access rights
PS C:\Foo> Get-SmbShare -Name * |
    Get-SmbShareAccess |
    Format-Table -GroupBy Name
```

Name: IPC\$

Name	ScopeName	AccountName	AccessControlType	AccessRight
IPC\$	*	BUILTIN\Administrators	Allow	Full
IPC\$	*	BUILTIN\Backup Operators	Allow	Full
IPC\$	*	NT AUTHORITY\INTERACTIVE	Allow	Full

```
PS C:\Foo> # 2. Creating and sharing a new folder
PS C:\Foo> New-Item -Path F: -Name ITShare -ItemType Directory |
    Out-Null
PS C:\Foo> New-SmbShare -Name ITShare -Path F:\ITShare
```

Name	ScopeName	Path	Description
ITShare	*	F:\ITShare	

```
PS C:\Foo> # 6. Removing all access to ITShare share for the Everyone group
PS C:\Foo> $AdminHT1 = @{
    Name      = 'ITShare'
    AccountName = 'Everyone'
    Confirm   = $false
}
PS C:\Foo> Revoke-SmbShareAccess @AdminHT1
```

Name	ScopeName	AccountName	AccessControlType	AccessRight
ITShare	*	Everyone	Deny	Full

```
PS C:\Foo> # 7. Adding Reskit\Administrators to have read permission
PS C:\Foo> $AHT2 = @{
    Name      = 'ITShare'
    AccessRight = 'Read'
    AccountName = 'Reskit\ADMINISTRATOR'
    ConFirm   = $false
}
PS C:\Foo> Grant-SmbShareAccess @AHT2
```

Name	ScopeName	AccountName	AccessControlType	AccessRight
ITShare	*	RESKIT\Administrator	Allow	Read

```
PS C:\Foo> # 11. Reviewing share access
PS C:\Foo> Get-SmbShareAccess -Name ITShare |
Sort-Object AccessRight
```

Name	ScopeName	AccountName	AccessControlType	AccessRight
ITShare	*	NT AUTHORITY\SYSTEM	Allow	Full
ITShare	*	CREATOR OWNER	Allow	Full
ITShare	*	RESKIT\Administrator	Allow	Read
ITShare	*	RESKIT\Sales	Allow	Read

```
PS C:\Foo> # 15. Viewing file ACL
PS C:\Foo> Get-NTFSAccess -Path F:\ITShare\File.Txt |
Format-Table -AutoSize
```

Path: F:\ITShare\File.Txt (Inheritance enabled)

Account	Access Rights	Applies to	Type	IsInherited	InheritedFrom
BUILTIN\Administrators	FullControl	ThisFolderOnly	Allow	True	F:\ITShare
NT AUTHORITY\SYSTEM	FullControl	ThisFolderOnly	Allow	True	F:\ITShare
RESKIT\Administrator	ReadAndExecute, Synchronize	ThisFolderOnly	Allow	True	F:\ITShare
RESKIT\Sales	ReadAndExecute, Synchronize	ThisFolderOnly	Allow	True	F:\ITShare
BUILTIN\Users	ReadAndExecute, Synchronize	ThisFolderOnly	Allow	True	F:



```
PS C:\Foo> # 1. Examining the SMB client's configuration on SRV1
PS C:\Foo> Get-SmbClientConfiguration
```

```
SkipCertificateCheck           : False
ConnectionCountPerRssNetworkInterface : 4
DirectoryCacheEntriesMax       : 16
DirectoryCacheEntrySizeMax     : 65536
DirectoryCacheLifetime         : 10
DormantFileLimit               : 1023
EnableBandwidthThrottling      : True
EnableByteRangeLockingOnReadOnlyFiles : True
EnableInsecureGuestLogons      : True
EnableLargeMtu                 : True
EnableLoadBalanceScaleOut      : True
EnableMultiChannel              : True
EnableSecuritySignature        : True
ExtendedSessionTimeout         : 1000
FileInfoCacheEntriesMax        : 64
FileInfoCacheLifetime          : 10
FileNotFoundCacheEntriesMax    : 128
FileNotFoundCacheLifetime      : 5
ForceSMBEncryptionOverQuic     : False
KeepConn                       : 600
MaxCmds                        : 50
MaximumConnectionCountPerServer : 32
OplocksDisabled                : False
RequireSecuritySignature        : False
SessionTimeout                 : 60
UseOpportunisticLocking        : True
WindowSizeThreshold            : 1
```

```
PS C:\Foo> # 3. Examining SMB client's network interface
PS C:\Foo> Get-SmbClientNetworkInterface |
Format-Table
```

Interface	Index	RSS Capable	RDMA Capable	Speed	IpAddresses	Friendly Name
7		True	False	10 Gbps	{fe80::8d9c:754b:9c00:54, 10.10.10.101}	Ethernet

```
PS C:\Foo> # 4. Examining the shares provided by FS1
PS C:\Foo> net view \\FS1
Shared resources at \\FS1
```

Share name	Type	Used as	Comment
ITShare	Disk	R:	File Share for IT

The command completed successfully.



```
PS C:\Foo> # 5. Creating a drive mapping, mapping R: to the share on server FS1
PS C:\Foo> New-SmbMapping -LocalPath R: -RemotePath \\FS1\ITShare
```

Status	Local Path	Remote Path
OK	R:	\\FS1\ITShare

```
PS C:\Foo> # 6. Viewing the shared folder mapping
PS C:\Foo> Get-SmbMapping
```

Status	Local Path	Remote Path
OK	R:	\\FS1\ITShare

```
PS C:\Foo> # 7. Viewing the shared folder contents
PS C:\Foo> Get-ChildItem -Path R:
```

Directory: R:\

Mode	LastWriteTime	Length	Name
-a---	13/08/2022 11:23	15	File.Txt

```
PS C:\Foo> # 8. Viewing existing connections
PS C:\Foo> Get-SmbConnection
```

ServerName	ShareName	UserName	Credential	Dialect	NumOpens
FS1	ITShare	RESKIT\Administrator	RESKIT\Administrator	3.1.1	1

```
PS C:\Foo> # 1. Installing the iSCSI target feature on SS1
PS C:\Foo> Import-Module -Name ServerManager -WarningAction SilentlyContinue
PS C:\Foo> Install-WindowsFeature FS-iSCSITarget-Server
```

Success	Restart Needed	Exit Code	Feature Result
True	Yes	SuccessRestar...	{File and iSCSI Services, File Server, iSCSI... WARNING: You must restart this server to finish the installation process.

```
PS C:\Foo> # 3. Exploring iSCSI target server settings
PS C:\Foo> Get-IscsiTargetServerSetting
```

```
RunspaceId      : dcd86396-786d-435e-a056-6f3b7b1d94b8
ComputerName    : SS1.Reskit.Org
IsClustered     : False
Version         : 10.0
DisableRemoteManagement : False
Portals         : {+10.10.1.17:3260, -[fe80::5817:e84:6e63:f824%6]:32601}
```

```
PS C:\Foo> # 5. Creating an iSCSI virtual disk (aka a LUN)
PS C:\Foo> $VDiskPath = 'C:\iSCSI\ITData.Vhdx'
PS C:\Foo> $VDHT = @{
    Path          = $VDiskPath
    Description    = 'LUN For IT Group'
    SizeBytes      = 500MB
}
PS C:\Foo> New-IscsiVirtualDisk @VDHT
```

```
RunspaceId      : dcd86396-786d-435e-a056-6f3b7b1d94b8
ClusterGroupName :
ComputerName     : SS1.Reskit.Org
Description       : LUN For IT Group
DiskType         : Dynamic
HostVolumeId     : {BF124141-EFEA-11EC-BE5E-E454E88CB586}
LocalMountDeviceId :
OriginalPath      :
ParentPath        :
Path             : C:\iSCSI\ITData.Vhdx
SerialNumber      : C8C701D3-E94C-40FF-B195-72B091357B4C
Size             : 524288000
SnapshotIds       :
Status           : NotConnected
VirtualDiskIndex  : 1694099853
```

```

PS C:\Foo> # 6. Setting the iSCSI target, specifying who can initiate an iSCSI connection
PS C:\Foo> $TargetName = 'ITTarget'
PS C:\Foo> $NewTargetHT = @{
    TargetName = $TargetName
    InitiatorIds = 'IQN:*'
}
PS C:\Foo> New-IscsiServerTarget @NewTargetHT

RunspaceId          : dcd86396-786d-435e-a056-6f3b7b1d94b8
ChapUserName         :
ClusterGroupName    :
ComputerName        : SS1.Reskit.Org
Description          :
EnableChap           : False
EnableReverseChap    : False
EnforceIdleTimeoutDetection : True
FirstBurstLength     : 65536
IdleDuration         : 00:00:00
InitiatorIds         : {Iqn:*}
LastLogin            :
LunMappings          : {}
MaxBurstLength       : 262144
MaxReceiveDataSegmentLength : 65536
ReceiveBufferCount   : 10
ReverseChapUserName  :
Sessions             : {}
Status               : NotConnected
TargetIqn            : iqn.1991-05.com.microsoft:ssl-ittarget-target
TargetName           : ITTarget

```

```

PS C:\Foo> # 2. Setting up the portal to SS1
PS C:\Foo> $PortalHT = @{
    TargetPortalAddress = 'SS1.Reskit.Org'
    TargetPortalPortNumber = 3260
}
PS C:\Foo> New-IscsiTargetPortal @PortalHT

RunspaceId          : 903f3c98-40a9-4094-9360-2a710fd27b3a
InitiatorInstanceName :
InitiatorPortalAddress :
IsDataDigest         : False
IsHeaderDigest        : False
TargetPortalAddress   : SS1.Reskit.Org
TargetPortalPortNumber : 3260

```

```

PS C:\Foo> # 3. Finding and viewing the ITTarget on the portal
PS C:\Foo> $Target = Get-IscsiTarget |
    Where-Object NodeAddress -Match 'ITTarget'
PS C:\Foo> $Target

RunspaceId : 903f3c98-40a9-4094-9360-2a710fd27b3a
IsConnected : False
NodeAddress : iqn.1991-05.com.microsoft:ssl-ittarget-target

```

```

PS C:\Foo> # 4. Connecting to the target on SS1
PS C:\Foo> $ConnectHT = @{
    TargetPortalAddress = 'SS1.Reskit.Org'
    NodeAddress         = $Target.NodeAddress
}
PS C:\Foo> Connect-IscsiTarget @ConnectHT

RunspaceId          : 903f3c98-40a9-4094-9360-2a710fd27b3a
AuthenticationType  : NONE
InitiatorInstanceName : ROOT\ISCSIPRT\0000_0
InitiatorNodeAddress : iqn.1991-05.com.microsoft:fs1.reskit.org
InitiatorPortalAddress : 0.0.0.0
InitiatorSideIdentifier : 400001370000
IsConnected         : True
IsDataDigest        : False
IsDiscovered         : False
IsHeaderDigest       : False
IsPersistent         : False
NumberOfConnections : 1
SessionIdentifier    : fffff7045daff010-40000137000000002
TargetNodeAddress    : iqn.1991-05.com.microsoft:ss1-ittarget-target
TargetSideIdentifier  : 0100

```

```

PS C:\Foo> # 5. Viewing the iSCSI disk from FS1 on SS1
PS C:\Foo> $RemoteDisk = Get-Disk |
    Where-Object BusType -eq 'iscsi'
PS C:\Foo> $RemoteDisk |
    Format-Table -AutoSize

```

Number	Friendly Name	Serial Number	HealthStatus	OperationalStatus	Total Size	Partition Style
2	MSFT Virtual HD	C8C701D3-E94C-40FF-B195-72B091357B4C	Healthy	Offline	500 MB	RAW

```

PS C:\Foo> # 7. Formatting the volume on SS1
PS C:\Foo> $NewVolumeHT = @{
    FriendlyName = 'ITData'
    FileSystem   = 'NTFS'
    DriveLetter  = 'I'
}
PS C:\Foo> $RemoteDisk |
    New-Volume @NewVolumeHT

```

DriveLetter	FriendlyName	FileSystemType	DriveType	HealthStatus	OperationalStatus	SizeRemaining	Size
I	ITData	NTFS	Fixed	Healthy	OK	467.73 MB	483.93 MB



```

PS C:\Foo> # 8. Using the drive as a local drive
PS C:\Foo> Set-Location -Path I:
PS I:\> New-Item -Path I:\ -Name ITData -ItemType Directory |
        Out-Null
PS I:\> 'Testing 1-2-3' |
        Out-File -FilePath I:\ITData\Test.Txt
PS I:\> Get-ChildItem I:\ITData

```

Directory: I:\ITData

Mode	LastWriteTime	Length	Name
-a---	19/08/2022 20:26	15	Test.Txt

```

PS C:\Foo> # 1. Installing FS Resource Manager feature on FS1
PS C:\Foo> Import-Module -Name ServerManager -WarningAction 'SilentlyContinue'
PS C:\Foo> $InstallIHT = @{
                Name = 'FS-Resource-Manager'
                IncludeManagementTools = $True
                WarningAction = 'SilentlyContinue'
            }
PS C:\Foo> Install-WindowsFeature @InstallIHT

```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	{File Server Resource Manager, Remote Server...

```
PS C:\Foo> # 2. Viewing default FSRM settings
PS C:\Foo> Get-FsrmSetting
```


```
AdminEmailAddress           :
CommandNotificationLimit    : 60
EmailNotificationLimit      : 60
EventNotificationLimit      : 60
FromEmailAddress            :
ReportClassificationFormat   : DHtml
ReportClassificationLog      : {ClassificationsInLogFile, ErrorsInLogFile}
ReportClassificationMailTo   :
ReportFileGroupIncluded     :
ReportFileOwnerFilePattern   :
ReportFileOwnerUser         :
ReportFileScreenAuditDaysSince : 0
ReportFileScreenAuditEnable  : False
ReportFileScreenAuditUser    :
ReportLargeFileMinimum       : 5242880
ReportLargeFilePattern       :
ReportLeastAccessedFilePattern :
ReportLeastAccessedMinimum   : 90
ReportLimitMaxDuplicateGroup : 100
ReportLimitMaxFile           : 1000
ReportLimitMaxFileGroup      : 10
ReportLimitMaxFileScreenEvent : 1000
ReportLimitMaxFilesPerDuplicateGroup : 10
ReportLimitMaxFilesPerFileGroup : 100
ReportLimitMaxFilesPerOwner  : 100
ReportLimitMaxFilesPerPropertyValue : 100
ReportLimitMaxOwner          : 10
ReportLimitMaxPropertyValue  : 10
ReportLimitMaxQuota           : 1000
ReportLocationIncident       : C:\StorageReports\Incident
ReportLocationOnDemand       : C:\StorageReports\Interactive
ReportLocationScheduled      : C:\StorageReports\Scheduled
ReportMostAccessedFilePattern :
ReportMostAccessedMaximum    : 7
ReportNotificationLimit      : 60
ReportPropertyFilePattern    :
ReportPropertyName           :
ReportQuotaMinimumUsage      : 0
Server                       : Reserved
SmtptServer                  :
PSComputerName               :
```

Test notification message for File Server Resource Manager - Message (Plain Text)

File Message Help Tell me what you want to do

Delete Archive Reply Reply All Forward Respond Share to Teams Quick Steps Move Tags Editing Immersive Zoom Send to OneNote Viva Insights Add-in

Fri 26/08/2022 16:16

 tfl@psp.co.uk

To Thomas Lee

Test notification message for File Server Resource Manager

This email address has been configured as a default for quota, file screening, and file management notifications on server FS1. To update this information, in File Server Resource Manager, on the Action menu, click Configure Options, and then update the email addresses on the Email Notifications tab.

```

PS C:\Foo> # 5. Creating a new FSRM quota template for a 10MB hard limit
PS C:\Foo> $QuotaHT1 = @{
    Name      = '10 MB Reskit Quota'
    Description = 'Filestore Quota (10mb)'
    Size      = 10MB
}
PS C:\Foo> New-FsrmQuotaTemplate @QuotaHT1

```

```

Description      : Filestore Quota (10mb)
Name             : 10 MB Reskit Quota
Size             : 10485760
SoftLimit        : False
Threshold        :
UpdateDerived    : False
UpdateDerivedMatching : False
PSComputerName   :

```

```

PS C:\Foo> # 6. Viewing available FSRM quota templates
PS C:\Foo> Get-FsrmQuotaTemplate |
    Format-Table -Property Name, Description, Size, SoftLimit

```

Name	Description	Size	SoftLimit
100 MB Limit		104857600	False
200 MB Limit Reports to User		209715200	False
Monitor 200 GB Volume Usage		214748364800	True
Monitor 500 MB Share		524288000	True
200 MB Limit with 50 MB Extension		209715200	False
250 MB Extended Limit		262144000	False
2 GB Limit		2147483648	False
5 GB Limit		5368709120	False
10 GB Limit		10737418240	False
Monitor 3 TB Volume Usage		3298534883328	True
Monitor 5 TB Volume Usage		5497558138880	True
Monitor 10 TB Volume Usage		10995116277760	True
10 MB Reskit Quota	Filestore Quota (10mb)	10485760	False

```

PS C:\Foo> # 10. Building a quota for the C:\Quota folder
PS C:\Foo> $NewQuotaHT1 = @{
    Path      = 'C:\Quota'
    Template  = '10 MB Reskit Quota'
    Threshold = $Thresh
}
PS C:\Foo> New-FsrmQuota @NewQuotaHT1

```

```

Description      :
Disabled         : False
MatchesTemplate  : False
Path             : C:\Quota
PeakUsage        : 1024
Size             : 10485760
SoftLimit        : False
Template         : 10 MB Reskit Quota
Threshold        : {MSFT_FSRMQuotaThreshold}
Usage            : 1024
PSComputerName   :

```

FSRM Over limit RESKIT\Administrator -...

File Message Help Tell me what you want to do

Delete Archive Reply Reply All Forward Move Tags Editing Read Aloud Speech Zoom

Sun 07/03/2021 15:58

**F** FSRM@Reskit.Org

FSRM Over limit RESKIT\Administrator

To Doctordns@gmail.Com

*We removed extra line breaks from this message.*

User RESKIT\Administrator has exceeded the 85% quota threshold for the quota on C:\Quota on server SRV1.  
The quota limit is 10.00 MB, and 8.51 MB currently is in use (85% of limit).

```

PS C:\Foo> # 12. Testing the hard limit quota
PS C:\Foo> $Test1 | Out-File -FilePath C:\Quota\Demo3.Txt
out-lineoutput: There is not enough space on the disk. : 'C:\Quota\Demo3.Txt'

```



```
PS C:\Foo> # 13. Viewing the contents of the C:\Quota folder
PS C:\Foo> Get-ChildItem -Path C:\Quota
```

Directory: C:\Quota

Mode		LastWriteTime	Length	Name
-a---		26/08/2022 17:16	8388610	Demo1.Txt
-a---		26/08/2022 17:16	692062	Demo2.Txt
-a---		26/08/2022 17:17	1441792	Demo3.Txt

```
PS C:\Foo> # 1. Creating a new FSRM storage report for large files on C:\ on FS1
PS C:\Foo> $NewReportHT = @{
```

```
    Name           = 'Large Files on FS1'
    Namespace      = 'C:\'
    ReportType     = 'LargeFiles'
    LargeFileMinimum = 10MB
    Interactive     = $true
}
```

```
PS C:\Foo> New-FsrmStorageReport @NewReportHT
```

```
FileGroupIncluded      :
FileOwnerFilePattern   :
FileOwnerUser          :
FileScreenAuditDaysSince : 0
FileScreenAuditUser    :
FolderPropertyName     :
Interactive            : True
LargeFileMinimum       : 10485760
LargeFilePattern       :
LastError              :
LastReportPath         :
LastRun                :
LeastAccessedFilePattern :
LeastAccessedMinimum   : 0
MailTo                 :
MostAccessedFilePattern :
MostAccessedMaximum    : 0
Name                   : Large Files on FS1
Namespace              : {C:\}
PropertyFilePattern    :
PropertyName           :
QuotaMinimumUsage      : 0
ReportFormat           : {DHtml, XML}
ReportType             : LargeFiles
Schedule               :
Status                 : Queued
PSComputerName         :
```

```
PS C:\Foo> # 2. Getting existing FSRM reports
PS C:\Foo> Get-FsrmStorageReport -Name * |
    Format-Table -Property Name, Namespace,
    ReportType, ReportFormat
```

Name	Namespace	ReportType	ReportFormat
Large Files on FS1	{C:\}	LargeFiles	{DHtml, XML}

```
PS C:\Foo> # 3. Viewing Interactive reports available on FS1
PS C:\Foo> $Path = 'C:\StorageReports\Interactive'
PS C:\Foo> Get-ChildItem -Path $Path
```

Directory: C:\StorageReports\Interactive

Mode	LastWriteTime	Length	Name
d----	28/08/2022 13:04		LargeFiles2_2022-08-28_13-03-47_files
-a---	28/08/2022 13:04	235866	LargeFiles2_2022-08-28_13-03-47.html
-a---	28/08/2022 13:04	430926	LargeFiles2_2022-08-28_13-03-47.xml

LargeFiles3\_2022-08-28\_13-05-18

File

C:/StorageReports/Interactive/LargeFiles3\_2022-08-28\_13-05-18.html

### Large Files Report

Generated at: 28/08/2022 13:05:18

**Report Description:** Lists files that are a specified size or larger. Use this report to quickly identify the files that are consuming the most disk space on the server. These can help you quickly reclaim large quantities of disk space.

**Machine:** FS1

**Report Folders:** 'C:\'

**Parameters:** Minimum file size: 10.00 MB

[Large Files Report Table of Contents](#)

[Report Totals](#)  
[Size by Owner](#)  
[Size by File Group](#)  
[Report statistics](#)

Report Totals			
Files shown in the report		All files matching report criteria	
Files	Total size on Disk	Files	Total size on Disk
251	13,774 MB	251	13,774 MB

[To top of the current report](#)

#### Size By Owner

BUILTIN\Administrators	8,411 MB; (61.07%)
NT AUTHORITY\SYSTEM	2,503 MB; (18.17%)
NT SERVICE\TrustedInstaller	2,459 MB; (17.85%)
NT AUTHORITY\LOCAL SERVICE	349 MB; (2.53%)
Others	51.0 MB; (0.37%)

```
PS C:\Foo> # 5. Extracting key information from the FSRM XML output
PS C:\Foo> $Files | Where-Object Path -NotMatch '^Windows|^Program|^Users' |
    Format-Table -Property name, path,
        @{ Name = 'SizeMB';
            Expression = {((([int]$_).size)/1mb).tostring('N2')}};
        DaysSinceLastAccessed -AutoSize
```

Name	Path	SizeMB	DaysSinceLastAccessed
2{3808876b-c176-4e48-b7ae-04046e6cc752}	System Volume Information		0
pagefile.sys		1,536.00	2
winre.wim	Recovery\WindowsRE	439.15	2
CascadiaCode.zip	Foo	23.75	18
PresentationFramework.dll	PSPreview	15.49	18
PresentationFramework.dll	PSDailyBuild	15.48	18
System.Windows.Forms.dll	PSPreview	12.70	18
System.Windows.Forms.dll	PSDailyBuild	12.67	18
System.Private.CoreLib.dll	PSDailyBuild	11.13	18
System.Private.CoreLib.dll	PSPreview	10.92	18

```
PS C:\Foo> # 7. Getting details of the task
PS C:\Foo> Get-ScheduledTask |
    Where-Object TaskName -Match 'Monthly' |
    Format-Table -AutoSize
```

TaskPath	TaskName	State
\Microsoft\Windows\File Server Resource Manager\	StorageReport-Monthly Files by files group report	Ready

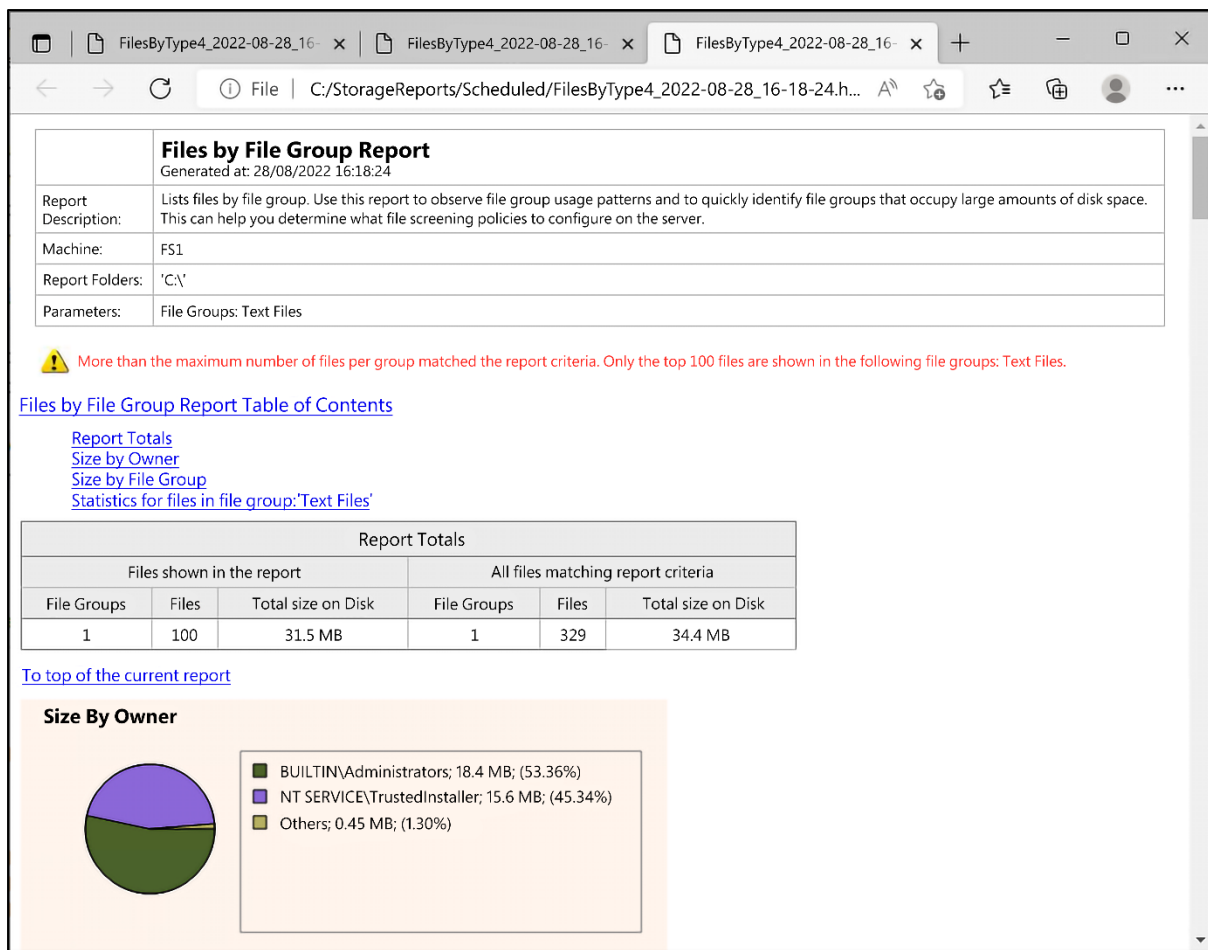
```
PS C:\Foo> # 8. Running the task now
PS C:\Foo> Get-ScheduledTask -TaskName '*Monthly*' |
    Start-ScheduledTask
PS C:\Foo> Get-ScheduledTask -TaskName '*Monthly*'
```

TaskPath	TaskName	State
\Microsoft\Windows\File Server Resource Manag...	StorageReport-Monthly Files by f...	Running

```
PS C:\Foo> # 9. Viewing the report in the StorageReports folder
PS C:\Foo> $Path = 'C:\StorageReports\Scheduled'
PS C:\Foo> $Report = Get-ChildItem -Path $Path\*.html
PS C:\Foo> $Report
```

Directory: C:\StorageReports\Scheduled

Mode	LastWriteTime	Length	Name
-a---	28/08/2022 16:13	94620	FilesByType4_2022-08-28_16-12-40.html



```
PS C:\Foo> # 1. Examining the existing FSRM file groups
PS C:\Foo> Get-FsrmFileGroup |
Format-Table -Property Name, IncludePattern

Name                                IncludePattern
----
Audio and Video Files {*.aac, *.aif, *.aiff, *.asf, *.asx, *.au, *.avi, *.flac, *.m3u, *.mid, *.midi, *.mov,
*.mp1, *.mp2, *.mp3, *.mp4, *.mpa, *.mpe, *.mpeg, *.mpeg2, *.mpeg3, *.mpg, *.ogg, *.qt,
*.qtw, *.ram, *.rm, *.rmi, *.rmvb, *.snd, *.swf, *.vob, *.wav, *.wax, *.wma, *.wmv,
*.wvx}
Image Files {*.bmp, *.dib, *.eps, *.gif, *.img, *.jfif, *.jpe, *.jpeg, *.jpg, *.pcx, *.png, *.ps,
*.psd, *.raw, *.rif, *.spiff, *.tif, *.tiff}
Office Files {*.accdb, *.accde, *.accdr, *.accdt, *.adn, *.adp, *.doc, *.docm, *.docx, *.dot, *.dotm,
*.dotx, *.grv, *.gsa, *.gta, *.mad, *.maf, *.mda, *.mda, *.mda, *.mdb, *.mde, *.mdf,
*.mdf, *.mdm, *.mdt, *.mdw, *.mdw, *.mdw, *.mdz, *.mpd, *.mpp, *.mpt, *.obt, *.odb,
*.one, *.onepkg, *.pot, *.potm, *.potx, *.ppa, *.ppam, *.pps, *.ppsm, *.ppsx, *.ppt,
*.pptm, *.pptx, *.pub, *.pwz, *.rqt, *.rtf, *.rwz, *.sldm, *.sldx, *.slk, *.thmx, *.vdx,
*.vsd, *.vsl, *.vss, *.vst, *.vsu, *.vsw, *.vsx, *.vtx, *.wbk, *.wri, *.xla, *.xlam,
*.xlb, *.xlc, *.xld, *.xll, *.xlm, *.xls, *.xlsb, *.xlsm, *.xlsx, *.xlt, *.xltn,
*.xltx, *.xlv, *.xlw, *.xsf, *.xsn}
E-mail Files {*.eml, *.idx, *.mbox, *.mbx, *.msg, *.oft, *.ost, *.pab, *.pst}
Executable Files {*.bat, *.cmd, *.com, *.cpl, *.exe, *.inf, *.js, *.jse, *.msh, *.msi, *.msp, *.ocx,
*.pif, *.pl, *.psl, *.scr, *.vb, *.vbs, *.wsf, *.wsh}
System Files {*.acm, *.dll, *.ocx, *.sys, *.vxd}
Compressed Files {*.ace, *.arc, *.arj, *.bhx, *.bz2, *.cab, *.gz, *.gzip, *.hpk, *.hqx, *.jar, *.lha,
*.lzh, *.lzx, *.pak, *.pit, *.rar, *.sea, *.sit, *.sqz, *.tgz, *.uu, *.uue, *.z, *.zip,
*.zoo}
Web Page Files {*.asp, *.aspx, *.cgi, *.css, *.dhtml, *.hta, *.htm, *.html, *.mht, *.php, *.php3,
*.shtml, *.url}
Text Files {*.asc, *.text, *.txt}
Backup Files {*.bak, *.bck, *.bkf, *.old}
Temporary Files {*.temp, *.tmp, ~*}
```



```
PS C:\Foo> # 2. Examining the existing file screening templates
PS C:\Foo> Get-FsrmFileScreenTemplate |
    Format-Table -Property Name, IncludeGroup, Active
```

Name	IncludeGroup	Active
Block Audio and Video Files	{Audio and Video Files}	True
Block Executable Files	{Executable Files}	True
Block Image Files	{Image Files}	True
Block E-mail Files	{E-mail Files}	True
Monitor Executable and System Files	{Executable Files, System Files}	False

```
PS C:\Foo> # 4. Creating a new file screen
PS C:\Foo> $FileScreenHT = @{
    Path          = $Path
    Description    = 'Block Executable Files'
    IncludeGroup   = 'Executable Files'
}
PS C:\Foo> New-FsrmFileScreen @FileScreenHT
```

```
Active          : True
Description      : Block Executable Files
IncludeGroup     : {Executable Files}
MatchesTemplate  : False
Notification     :
Path             : C:\FileScreen
Template         :
PSComputerName   :
```

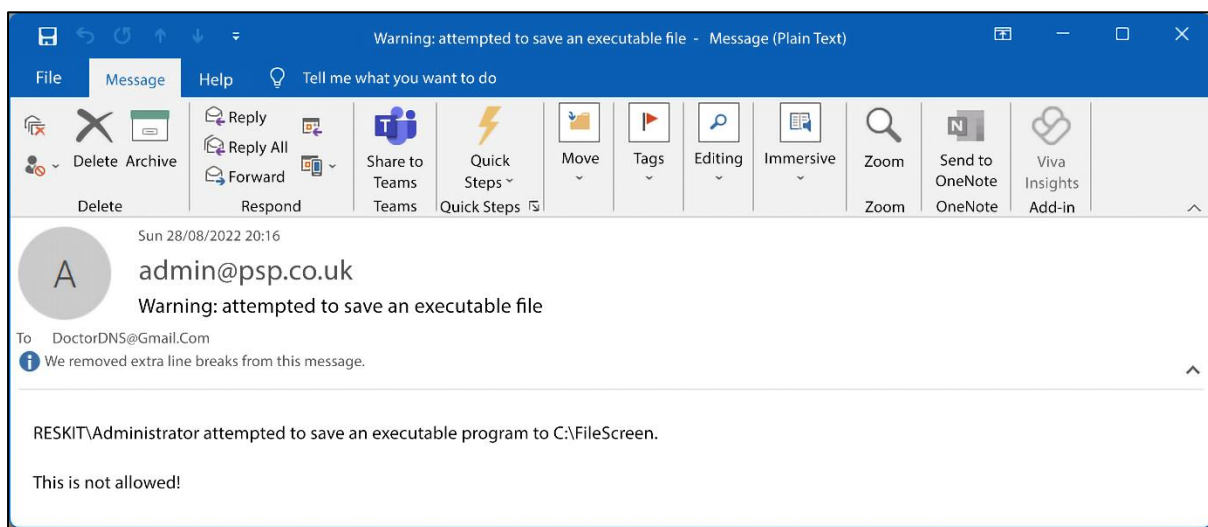
```
PS C:\Foo> # 5. Testing file screen by copying notepad.exe
PS C:\Foo> $FSTestHT = @{
    Path          = "$Env:windir\notepad.exe"
    Destination    = 'C:\FileScreen\notepad.exe'
}
```

```
PS C:\Foo> Copy-Item @FSTestHT
Copy-Item: Access to the path 'C:\FileScreen\notepad.exe' is denied.
```

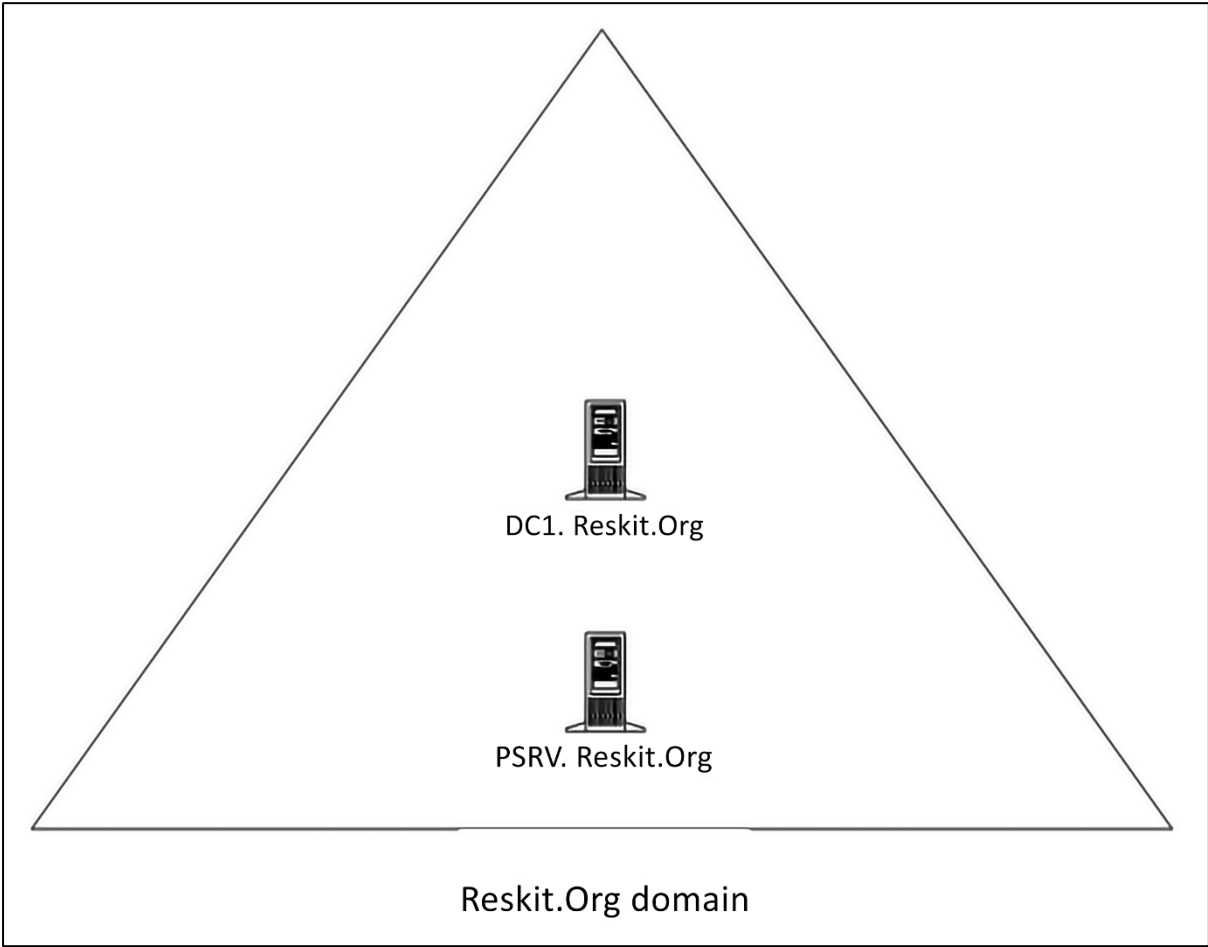
```
PS C:\Foo> # 7. Getting FSRM Notification Limits
PS C:\Foo> Get-FsrmSetting |
    Format-List -Property "*NotificationLimit"
```

```
CommandNotificationLimit : 60
EmailNotificationLimit    : 60
EventNotificationLimit    : 60
ReportNotificationLimit   : 60
```

```
PS C:\Foo> # 9. Re-testing the file screen to check the action
PS C:\Foo> Copy-Item @FSTestHT
Copy-Item: Access to the path 'C:\FileScreen\notepad.exe' is denied.
```



# Chapter 9: Managing Printing



```
PS C:\Foo> # 1. Installing the Print-Server features on PSRV
PS C:\Foo> Install-WindowsFeature -Name Print-Server,
                                     Print-Services,
                                     RSAT-Print-Services
```

Success	Restart Needed	Exit Code	Feature Result
True	No	Success	{Print Server, Print and Document Services, ...

```
PS C:\Foo> # 9. Reviewing what you have done
PS C:\Foo> Get-PrinterPort -Name SalesPP |
Format-Table -AutoSize -Property Name, Description,
PrinterHostAddress, PortNumber
```

Name	Description	PrinterHostAddress	PortNumber
SalesPP	Standard TCP/IP Port	10.10.10.61	9100

```
PS C:\Foo> Get-PrinterDriver -Name xerox* |
Format-Table -Property Name, Manufacturer,
DriverVersion, PrinterEnvironment
```

Name		Manufacturer	DriverVersion	PrinterEnvironment
Xerox Phaser 6510 PCL6		Xerox	1581047950660861952	Windows x64
Xerox WorkCentre 6515 PCL6		Xerox	1581047950660861952	Windows x64

```
PS C:\Foo> Get-Printer -ComputerName PSRV -Name SalesPrinter1 |
Format-Table -Property Name, ComputerName,
Type, PortName, Location, Shared
```

Name	ComputerName	Type	PortName	Location	Shared
SalesPrinter1	PSRV	Local	SalesPP		True

```
PS C:\Foo> # 10. Checking the status of the shared printer
PS C:\Foo> net view \\PSRV
Shared resources at \\PSRV
```

Share name	Type	Used as	Comment
SalesPrinter1	Print		SalesPrinter1

The command completed successfully.

```
PS C:\Foo> # 2. Viewing the printer details
PS C:\Foo> $Printer | Format-Table -Property Name, Published
```

Name	Published
SalesPrinter1	False


```
PS C:\Foo> # 4. Viewing the updated publication status
PS C:\Foo> Get-Printer -Name SalesPrinter1 |
Format-Table -Property Name, Location, Drivename, Published
```

Name	Location	Drivename	Published
SalesPrinter1	10th floor 10E4	Xerox Phaser 6510 PCL6	True



✕

←

 Add Printer

Find a printer by other options

☐ My printer is a little older. Help me find it.

☒ Find a printer in the directory, based on location or feature

☐ Select a shared printer by name

Browse...

Example \\computername\printername or  
http://computername/printers/printername/.printer

☐ Add a printer using a TCP/IP address or hostname

☐ Add a Bluetooth, wireless or network discoverable printer

☐ Add a local printer or network printer with manual settings

Next

Cancel

**Find Printers**

File Edit View

In: Entire Directory Browse...

Printers Features Advanced

Name:


Location:

Model:

Find Now


Stop

Clear All



OK

**Search results:**

Name	Location	Model
 SalesPrinter1	10th floor 10E4	Xerox WorkCentre 6515 PCL6

<  >

1 item(s) found

```
PS C:\Foo> # 4. Displaying print server properties
PS C:\Foo> $PrintServer
```

```
Name : \\COOKHAM216
SubSystemVersion : 0
RestartJobOnPoolEnabled : True
RestartJobOnPoolTimeout : 600
MinorVersion : 0
MajorVersion : 3
EventLog : LogPrintingErrorEvents
NetPopup : False
BeepEnabled : False
DefaultSchedulerPriority : Normal
SchedulerPriority : Normal
DefaultPortThreadPriority : Normal
PortThreadPriority : Normal
DefaultSpoolDirectory : C:\Windows\system32\spool\PRINTERS
Parent :
PropertiesCollection : {EventLog, RestartJobOnPoolTimeout, SchedulerPriority,
MinorVersion, Name, MajorVersion, DefaultSpoolDirectory,
DefaultPortThreadPriority, DefaultSchedulerPriority,
PortThreadPriority, BeepEnabled, RestartJobOnPoolEnabled, NetPopup}
```

```
PS C:\Foo> # 9. Restarting the Spooler to accept the new folder
PS C:\Foo> Restart-Service -Name Spooler
WARNING: Waiting for service 'Print Spooler (Spooler)' to start...
```

```
PS C:\Foo> # 10. Verifying the new spooler folder
PS C:\Foo> New-Object -TypeName System.Printing.PrintServer |
    Format-Table -Property Name,
    DefaultSpoolDirectory
```

Name	DefaultSpoolDirectory
\\PSRV C:\SpoolPath	

```
PS C:\Foo> # 15. Viewing the results
PS C:\Foo> New-Object -TypeName System.Printing.PrintServer |
    Format-Table -Property Name, DefaultSpoolDirectory
```

Name	DefaultSpoolDirectory
\\PSRV C:\WINDOWS\system32\spool\PRINTERS	

```
PS C:\Foo> # 2. Viewing loaded printer drivers:
PS C:\Foo> Get-PrinterDriver
```

Name	PrinterEnvironment	MajorVersion	Manufacturer
Microsoft XPS Document Writer v4	Windows x64	4	Microsoft
Microsoft Software Printer Driver	Windows x64	4	Microsoft
Microsoft Print To PDF	Windows x64	4	Microsoft
Xerox WorkCentre 6515 PCL6	Windows x64	3	Xerox
Xerox Phaser 6510 PCL6	Windows x64	3	Xerox
Remote Desktop Easy Print	Windows x64	3	Microsoft
Microsoft Shared Fax Driver	Windows x64	3	Microsoft
Microsoft enhanced Point and Print...	Windows x64	3	Microsoft
Microsoft enhanced Point and Print...	Windows NT x86	3	Microsoft

```
PS C:\Foo> Get-Printer -Name $PrinterName |
    Format-Table -Property Name, DriverName, PortName,
    Published, Shared
```

Name	DriverName	PortName	Published	Shared
SalesPrinter1	Xerox WorkCentre 6515 PCL6	SalesPP	True	True

```
PS C:\Foo> # 2. Displaying the number of printers defined on PSRV
PS C:\Foo> '{0} Printers defined on this system' -f $Printers.Count
7 Printers defined on this system
```

```
PS C:\Foo> # 4. Displaying the printer's details
PS C:\Foo> $Printer | Format-Table -AutoSize
```

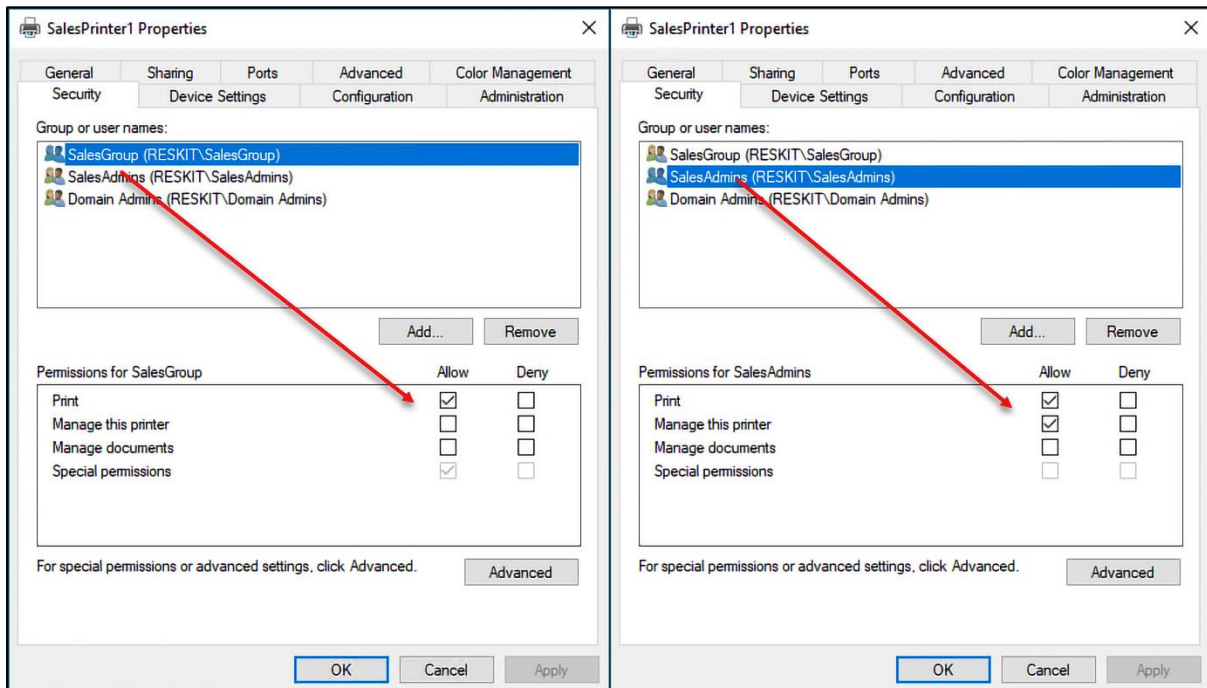
Name	ShareName	SystemName	PrinterState	PrinterStatus	Location
SalesPrinter1	SalesPrinter1	PSRV	0	3	10th floor 10E4

```
PS C:\Foo> # 5. Printing a test page
PS C:\Foo> Invoke-CimMethod -InputObject $Printer -MethodName PrintTestPage
```

```
ReturnValue PSComputerName
-----
0
```

```
PS C:\Foo> # 6. Checking on test page print job
PS C:\Foo> Get-PrintJob -PrinterName SalesPrinter1
```


Id	ComputerName	PrinterName	DocumentName	SubmittedTime	JobStatus
2		SalesPrinter1	Test Page	03/09/2022 13:16:18	Printing, Reta...




```
PS C:\Foo> Get-Printer $Printer |
Format-Table -Property Name, Type, DriverName, PortName -AutoSize
```

Name	Type	DriverName	PortName
SalesPrinter1	Local	Xerox WorkCentre 6515 PCL6	SalesPP,SalesPP2



 SalesPrinter1 Properties ✕

Security    Device Settings    Configuration    Administration  
General    Sharing    Ports    Advanced    Color Management

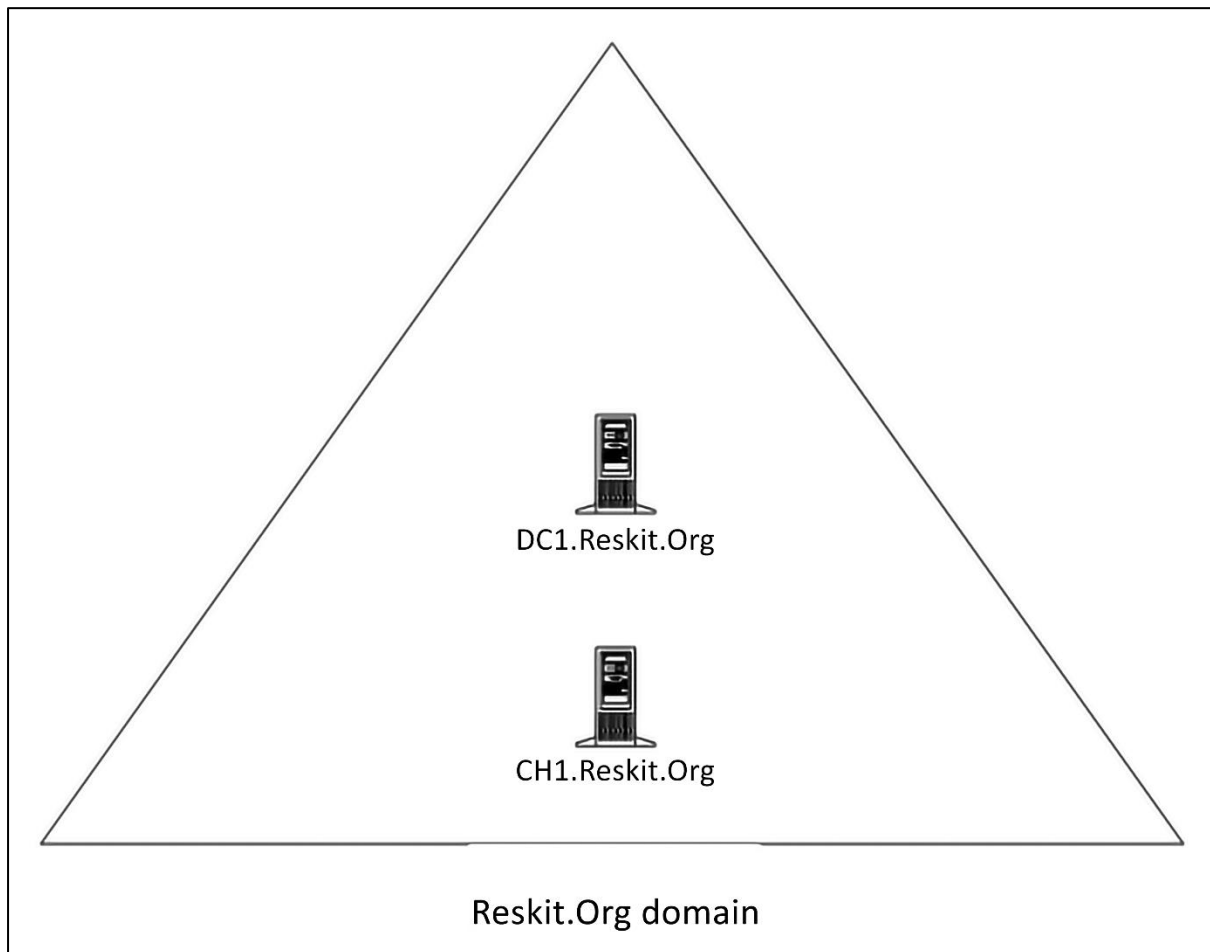
 SalesPrinter1

Print to the following port(s). Documents will print to the first free checked port.

Port	Description	Printer
<input type="checkbox"/> TS011	COOKHAM24: PRN2:...	
<input type="checkbox"/> TS012	Inactive TS Port	
<input type="checkbox"/> TS013	Inactive TS Port	
<input type="checkbox"/> TS014	Inactive TS Port	
<input checked="" type="checkbox"/> SalesPP	Standard TCP/IP Port	SalesPrinter1
<input checked="" type="checkbox"/> SalesPP2	Standard TCP/IP Port	SalesPrinter1
<input type="checkbox"/> PORTP...	Local Port	Microsoft XPS Document Write...

☐ Enable bidirectional support  
☒ Enable printer pooling

## Chapter 10: Exploring Windows Containers



```
PS C:\Foo> # 2. Installing the latest version of the docker package
PS C:\Foo> $InstallHT2 = @{
    Name           = 'Docker'
    ProviderName   = 'DockerMSFTProvider'
    Force          = $True
}
PS C:\Foo> Install-Package @InstallHT2
```

WARNING: A restart is required to enable the containers feature. Please restart your machine.

Name	Version	Source	Summary
Docker	20.10.9	DockerDefault	Contains Docker EE for use with Windows Server.

```
PS C:\Foo> # 2. Installing the latest version of the docker package
PS C:\Foo> $InstallHT2 = @{
    Name           = 'Docker'
    ProviderName   = 'DockerMSFTProvider'
    Force          = $True
}
PS C:\Foo> Install-Package @InstallHT2
```

WARNING: A restart is required to enable the containers feature. Please restart your machine.

Name	Version	Source	Summary
Docker	20.10.9	DockerDefault	Contains Docker EE for use with Windows Server.

```
PS C:\Foo> # 6. Checking Windows Containers and Hyper-V features are installed on CH1
PS C:\Foo> Get-WindowsFeature -Name Containers, Hyper-V
```

Display Name	Name	Install State
	Hyper-V	Installed
	Containers	Installed

```
PS C:\Foo> # 7. Checking Docker service
PS C:\Foo> Get-Service -Name Docker
```

Status	Name	DisplayName
Running	Docker	Docker Engine

```
PS C:\Foo> # 8. Checking Docker Version information
PS C:\Foo> docker version
```

```
Client: Mirantis Container Runtime
Version: 20.10.9
API version: 1.41
Go version: go1.16.12m2
Git commit: 591094d
Built: 12/21/2021 21:34:30
OS/Arch: windows/amd64
Context: default
Experimental: true
```

```
Server: Mirantis Container Runtime
Engine:
Version: 20.10.9
API version: 1.41 (minimum version 1.24)
Go version: go1.16.12m2
Git commit: 9b96ce992b
Built: 12/21/2021 21:33:06
OS/Arch: windows/amd64
Experimental: false
```

```

PS C:\Foo> # 9. Displaying Docker configuration information
PS C:\Foo> docker info
Client:
Context:    default
Debug Mode: false
Plugins:
  app: Docker App (Docker Inc., v0.9.1-beta3)
  cluster: Manage Mirantis Container Cloud clusters (Mirantis Inc., v1.9.0)
  registry: Manage Docker registries (Docker Inc., 0.1.0)

Server:
Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
Images: 7
Server Version: 20.10.9
Storage Driver: windowsfilter
  Windows:
Logging Driver: json-file
Plugins:
  Volume: local
  Network: ics internal l2bridge l2tunnel nat null overlay private transparent
  Log: awslogs etwlogs fluentd gcplogs gelf json-file local logentries splunk syslog
Swarm: inactive
Default Isolation: process
Kernel Version: 10.0 20348 (20348.1.amd64fre.fe_release.210507-1500)
Operating System: Windows Server 2022 Datacenter Version 2009 (OS Build 20348.169)
OSType: windows
Architecture: x86_64
CPUs: 6
Total Memory: 4.124GiB
Name: CH1
ID: VUDQ:UPUI:ZMI4:G53V:WTFT:VTZK:6YBB:SQI2:K34F:F70Z:DXW7:NEJ2
Docker Root Dir: C:\ProgramData\docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false

```

```

PS C:\Foo> # 1. Finding hello-world container images at the Docker Hub
PS C:\Foo> docker search hello-world

```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
hello-world	Hello World! (an example of minimal Dockeriz...	1835	[OK]	
kitematic/hello-world-nginx	A light-weight nginx container that demonstr...	152		
tutum/hello-world	Image to test docker deployments. Has Apache...	89		[OK]
dockercloud/hello-world	Hello World!	19		[OK]
crccheck/hello-world	Hello World web server in under 2.5 MB	15		[OK]
vadlmo/hello-world-rest	A simple REST Service that echoes back all t...	5		[OK]
ppc64le/hello-world	Hello World! (an example of minimal Dockeriz...	2		
rancher/hello-world		2		
ansibleplaybookbundle/hello-world-db-apb	An APB which deploys a sample Hello World! a...	2		[OK]
thomaspoignant/hello-world-rest-json	This project is a REST hello-world API to bu...	1		
ansibleplaybookbundle/hello-world-apb	An APB which deploys a sample Hello World! a...	1		[OK]
strimzi/hello-world-consumer		0		
armswdev/c-hello-world	Simple hello-world C program on Alpine Linux...	0		
strimzi/hello-world-producer		0		
koudaiii/hello-world		0		
businessgeeks00/hello-world-nodejs		0		
strimzi/hello-world-streams		0		
tacc/hello-world		0		
garystafford/hello-world	Simple hello-world Spring Boot service for t...	0		[OK]
freddiedevops/hello-world-spring-boot		0		
tsepotesting123/hello-world		0		
okteto/hello-world		0		
rsperling/hello-world3		0		
dandando/hello-world-dotnet		0		
kevindockercompany/hello-world		0		



```
PS C:\Foo> # 2. Pulling the Docker official hello-world image
PS C:\Foo> docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2ebf439f800c: Pull complete
59d9f62c09b7: Pull complete
d6884bc3f6c7: Pull complete
Digest: sha256:7d246653d0511db2a6b2e0436cfd0e52ac8c066000264b3ce63331ac66dca625
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

```
PS C:\Foo> # 3. Checking the Image just downloaded
PS C:\Foo> docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	d4d88879abb0	3 weeks ago	297MB

```
PS C:\Foo> # 4. Running the hello-world container image
PS C:\Foo> docker run hello-world
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(windows-amd64, nanoserver-ltsc2022)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run a Windows Server container with:

```
PS C:\> docker run -it mcr.microsoft.com/windows/servercore:ltsc2022 powershell
```

Share images, automate workflows, and more with a free Docker ID:  
<https://hub.docker.com/>

For more examples and ideas, visit:  
<https://docs.docker.com/get-started/>

```
PS C:\Foo> # 5. Getting Server Core image base image
PS C:\Foo> docker pull mcr.microsoft.com/windows/servercore:ltsc2022
ltsc2022: Pulling from windows/servercore
97f65a0ec59e: Pull complete
97b25a378238: Pull complete
Digest: sha256:35c3cb29ef2c9f05e36070de4c79d7fc861c035fa5df2df64ae607a276db42c6
Status: Downloaded newer image for mcr.microsoft.com/windows/servercore:ltsc2022
mcr.microsoft.com/windows/servercore:ltsc2022
```

```
PS C:\Foo> # 6. Checking the images available now on CH1
PS C:\Foo> docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	d4d88879abb0	3 weeks ago	297MB
mcr.microsoft.com/windows/servercore	ltsc2022	5798b78d003a	4 weeks ago	5.08GB

```
PS C:\Foo> # 7. Running the servercore container image
PS C:\Foo> docker run $ServerCore
Microsoft Windows [Version 10.0.20348.887]
(c) Microsoft Corporation. All rights reserved.
```

```
PS C:\Foo> # 9. Inspecting Server Core Image
PS C:\Foo> $ServerCoreImage = Get-DockerImage | Where-Object name -match servercore
PS C:\Foo> docker inspect $ServerCoreImage.ImageId | ConvertFrom-Json

Id                : sha256:5798b78d003a0eb4c52ddc590a333254e974bdc400f262bd7b4442bb2c6e49a2
RepoTags          : {mcr.microsoft.com/windows/servercore:ltsc2022}
RepoDigests       : {mcr.microsoft.com/windows/servercore@sha256:35c3cb29ef2c9f05e36070d04c79d7fc861c035fa5df2df64ae607a276db42c6}
Parent            :
Comment           :
Created           : 06/08/2022 02:59:35
Container         :
ContainerConfig    : @{Hostname=; Domainname=; User=; AttachStdin=False; AttachStdout=False;
                    AttachStderr=False; Tty=False; OpenStdin=False; StdinOnce=False; Env=;
                    Cmd=; Image=; Volumes=; WorkingDir=; Entrypoint=; OnBuild=; Labels=}
DockerVersion     :
Author            :
Config            : @{Hostname=; Domainname=; User=; AttachStdin=False; AttachStdout=False;
                    AttachStderr=False; Tty=False; OpenStdin=False; StdinOnce=False; Env=;
                    Cmd=System.Object[]; Image=; Volumes=; WorkingDir=; Entrypoint=; OnBuild=; Labels=}
Architecture      : amd64
Os                : windows
OsVersion         : 10.0.20348.887
Size              : 5083872027
VirtualSize       : 5083872027
GraphDriver       : @{Data=; Name=windowsfilter}
RootFS            : @{Type=layers; Layers=System.Object[]}
Metadata          : @{{LastTagTime=01/01/0001 00:00:00}}
```

```
PS C:\Foo> # 10. Pulling a Server 2019 container image
PS C:\Foo> $Server2019Image = mcr.microsoft.com/windows:1809
PS C:\Foo> docker pull $Server2019Image
1809: Pulling from windows
b079fa252589: Pull complete
3100d4854554: Pull complete
Digest: sha256:14241ad3587eb63e81c07e227adfc5b1ee4702d5b047599886fd82144210c479
Status: Downloaded newer image for mcr.microsoft.com/windows:1809
mcr.microsoft.com/windows:1809
```

```
PS C:\Foo> # 11. Running older server image
PS C:\Foo> docker run $Server2019Image
docker: Error response from daemon: hcsshim::CreateComputeSystem
3c8495f8debb5bf0cb39d141dbf2ba85c20e4c94b1c465e7228331dacf5de2b3:
The container operating system does not match the host operating system.
```

```
PS C:\Foo> # 12. run it with isolation
PS C:\Foo> PS C:\Foo> docker run --isolation=hyperv $Server2019Image
Microsoft Windows [Version 10.0.17763.3287]
(c) 2018 Microsoft Corporation. All rights reserved.
```

```

PS C:\Foo> # 13. Checking difference in run times with Hyper-V
PS C:\Foo> # Running with no isolation
PS C:\Foo> $Start1 = Get-Date
PS C:\Foo> docker run hello-world | Out-Null
PS C:\Foo> $End1 = Get-Date
PS C:\Foo> $Time1 = ($End1-$Start1).TotalMilliseconds
PS C:\Foo> # Running with isolation
PS C:\Foo> $Start2 = Get-Date
PS C:\Foo> docker run --isolation=hyperv hello-world | Out-Null
PS C:\Foo> $End2 = get-date
PS C:\Foo> $Time2 = ($End2-$Start2).TotalMilliseconds
PS C:\Foo> # Displaying the time differences
PS C:\Foo> "Without isolation, took : $Time1 milliseconds"
PS C:\Foo> "With isolation, took      : $Time2 milliseconds"
Without isolation, took : 2989.7237 milliseconds
With isolation, took      : 5881.3702 milliseconds

```

```

PS C:\Foo> # 14. Viewing system disk usage
PS C:\Foo> docker system df

```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	3	3	21.14GB	0B (0%)
Containers	5	0	0B	0B
Local Volumes	0	0	0B	0B
Build Cache	0	0	0B	0B

```

PS C:\Foo> # 15. Viewing active containers
PS C:\Foo> docker container ls -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
13e6b98c2a2a	hello-world	"cmd /C 'type C:\\heL..."	4 minutes ago	Exited (0) 4 minutes ago		dreamy_cannon
8287cd70830b	hello-world	"cmd /C 'type C:\\heL..."	4 minutes ago	Exited (0) 4 minutes ago		objective_ardinghelli
6c0cb5e6505f	mcr.microsoft.com/windows:1809	"c:\\windows\\system32..."	8 minutes ago	Exited (0) 7 minutes ago		condescending_hugle
3c8495f8debb	mcr.microsoft.com/windows:1809	"c:\\windows\\system32..."	9 minutes ago	Created		kind_bassi
c42a3e2b23aa	mcr.microsoft.com/windows/servercore:ltsc2022	"c:\\windows\\system32..."	47 minutes ago	Exited (0) 46 minutes ago		vigilant_swirles

```

PS C:\Foo> # 18. Removing other docker detritus
PS C:\Foo> docker system prune -f
Total reclaimed space: 0B

```

```

PS C:\Foo> # 19. Checking images and containers
PS C:\Foo> docker image ls

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE

```

PS C:\Foo> docker container ls

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES



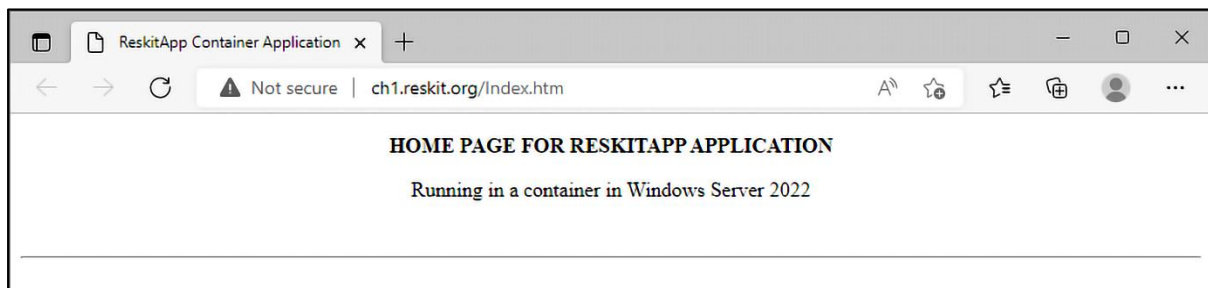
```
PS C:\Foo> # 1. Creating the reskitapp folder
PS C:\Foo> $EA = @{ErrorAction='SilentlyContinue'}
PS C:\Foo> New-Item -Path C:\ReskitApp -ItemType Directory @EA
```

Directory: C:\

Mode	LastWriteTime	Length	Name
d----	05/09/2022 16:13		ReskitApp

```
PS C:\Foo> # 3. Getting a server core with IIS image from the Docker registry:
PS C:\Foo> $Image = 'mcr.microsoft.com/windows/servercore/iis'
PS C:\Foo> docker pull $Image
Using default tag: latest
latest: Pulling from windows/servercore/iis
97f65a0ec59e: Pull complete
97b25a378238: Pull complete
7ebd66ebabd1: Pull complete
fa560e2e7835: Pull complete
39278cebaf6: Pull complete
Digest: sha256:d1821f5d785e5e17f4cb4194525dbcb57b7ec2e819d4db4738c14b6f2f2c2ad0
Status: Downloaded newer image for mcr.microsoft.com/windows/servercore/iis:latest
mcr.microsoft.com/windows/servercore/iis:latest
```

```
PS C:\Foo> # 4. Running the image as a container named rkwebc
PS C:\Foo> docker run -d -p80:80 --name rkwebc "$Image"
244189ade083393e734bf9aff4fb3339e4a6f340922ba812504327255fcaab20
```





```

PS C:\RKWebContainer> # 5. Build the Images
PS C:\RKWebContainer> docker build -t rkwebc .
Sending build context to Docker daemon 3.584kB
Step 1/7 : FROM mcr.microsoft.com/windows/servercore/iis
latest: Pulling from windows/servercore/iis
97f65a0ec59e: Pull complete
97b25a378238: Pull complete
7ebd66ebabd1: Pull complete
fa560e2e7835: Pull complete
39278cebafe6: Pull complete
Digest: sha256:d1821f5d785e5e17f4cb4194525dbcb57b7ec2e819d4db4738c14b6f2f2c2ad0
Status: Downloaded newer image for mcr.microsoft.com/windows/servercore/iis:lates
--> 8397e926fa67
Step 2/7 : LABEL Description="RKWEB Container" Vendor="PS Partnership" Version="1
--> Running in 451b28258dc5
Removing intermediate container 451b28258dc5
--> b18bdcdfe1df
Step 3/7 : RUN powershell -Command Add-WindowsFeature Web-Server
--> Running in bc01c1831a2e

Success Restart Needed Exit Code      Feature Result
-----
True      No              NoChangeNeeded {}

Removing intermediate container bc01c1831a2e
--> fbe8fcd9327a
Step 4/7 : RUN powershell -Command GIP
--> Running in ae9fa7a4d55c

InterfaceAlias      : vEthernet (Ethernet)
InterfaceIndex      : 21
InterfaceDescription : Hyper-V Virtual Ethernet Container Adapter
IPv4Address         : 172.26.216.230
IPv6DefaultGateway  :
IPv4DefaultGateway  : 172.26.208.1
DNSServer           : 172.26.208.1
                   10.10.10.10

Removing intermediate container ae9fa7a4d55c
--> f87f6d9a3836
Step 5/7 : WORKDIR C:\\RKWebContainer
--> Running in baee270860dd
Removing intermediate container baee270860dd
--> 15c07d026d4a
Step 6/7 : COPY Config.ps1 \\Config.ps1
--> a2c8e08456d6
Step 7/7 : RUN powershell -command ".\\Config.ps1"
--> Running in 7382f69c7cda

Name      ID      State      Physical Path      Bindings
-----
RKWeb     1299 Started    C:\RKWebContainer http *:80:RKWeb
          8361
          53          .Reskit.Org

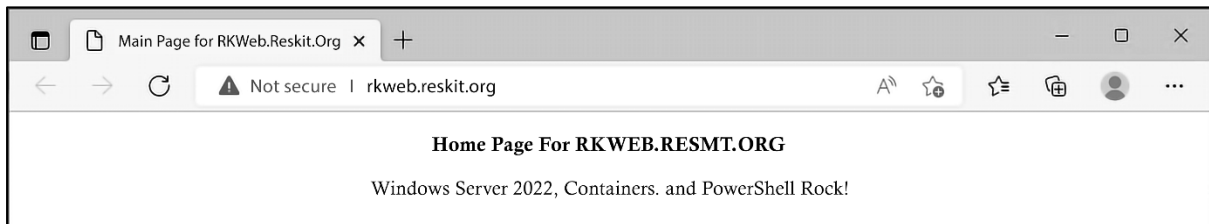
Removing intermediate container 7382f69c7cda
--> 8e710ffdf906
Successfully built 8e710ffdf906
Successfully tagged rkwebc:latest

```

```
PS C:\RKWebContainer> # 6. Running the image
PS C:\RKWebContainer> docker run -d --name rkwebc -p 80:80 rkwebc
e48e53b8bd46e7ca86991ca7c161b57037df3e2dc2989f44946a8d03942865ba
```

```
PS C:\RKWebContainer> # 7. Navigating to the container
PS C:\RKWebContainer> Invoke-WebRequest -UseBasicParsing HTTP://RKweb.Reskit.Org

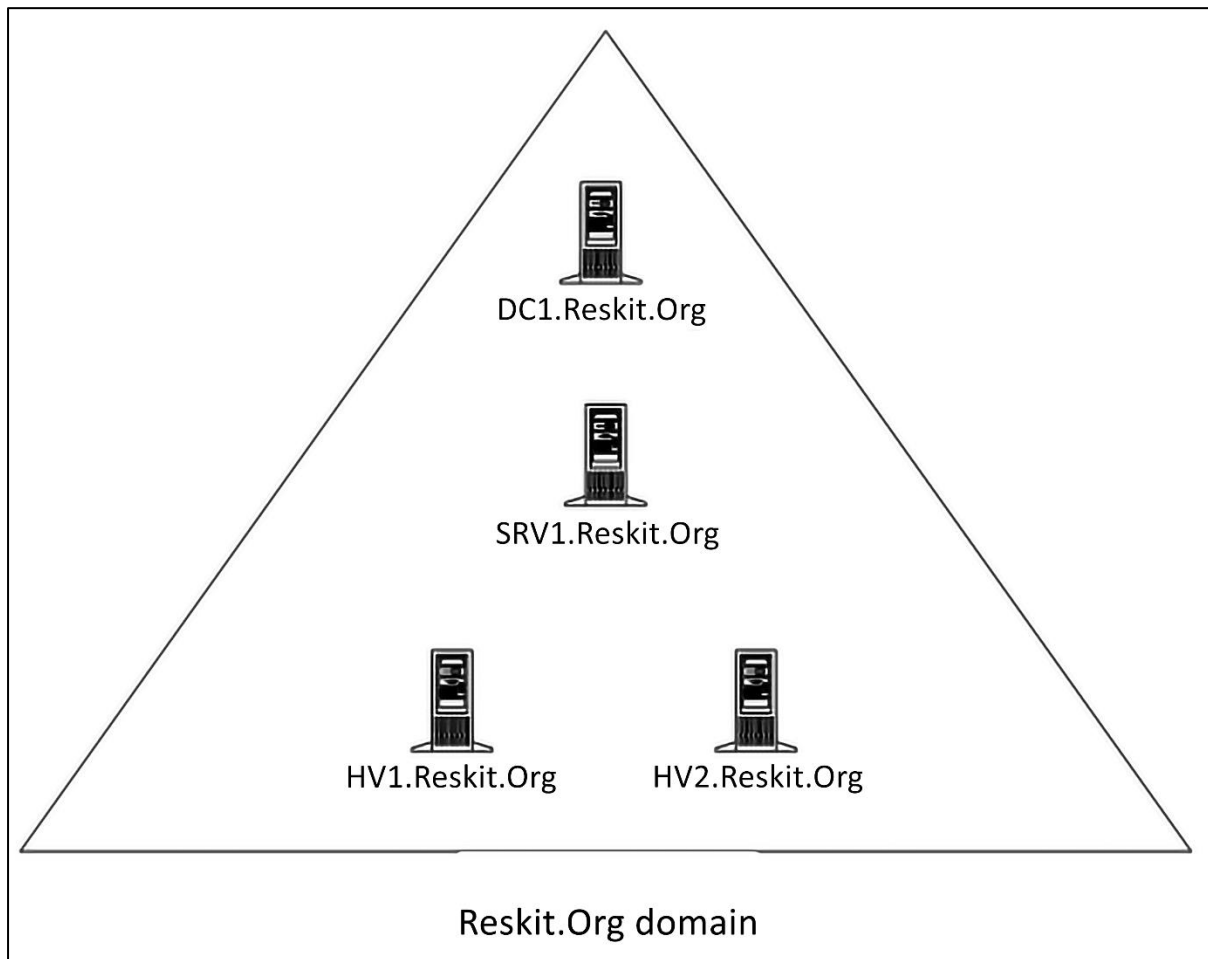
StatusCode      : 200
StatusDescription : OK
Content         : yp<!DOCTYPE html>
                  <html>
                  <head><title>Main Page for RKWeb.Reskit.Org</title></head>
                  <body><p><cen...
RawContent      : HTTP/1.1 200 OK
                  Accept-Ranges: bytes
                  ETag: "bae994ef43c1d81:0"
                  Server: Microsoft-IIS/10.0
                  Date: Mon, 05 Sep 2022 16:44:23 GMT
InputFields     : {}
Links           : {}
RawContentLength : 416
RelationLink    : {}
```



```
PS C:\RKWebContainer> # 9. Testing network connection
PS C:\RKWebContainer> Test-NetConnection -ComputerName localhost -Port 80

ComputerName      : localhost
RemoteAddress     : 127.0.0.1
RemotePort        : 80
InterfaceAlias    : Loopback Pseudo-Interface 1
SourceAddress     : 127.0.0.1
TcpTestSucceeded  : True
```

## Chapter 11: Managing Hyper-V



```
PS C:\Foo> # 1. Installing the Hyper-V feature on HV1, HV2
PS C:\Foo> $InstallSB = {
    Install-WindowsFeature -Name Hyper-V -IncludeManagementTools
}
PS C:\Foo> Invoke-Command -ComputerName HV1, HV2 -ScriptBlock $InstallSB

PSComputerName : HV2
RunspaceId     : 628a05c7-b80b-40e4-8feb-d5bdb93c694d
Success        : True
RestartNeeded  : Yes
FeatureResult   : {Hyper-V, Hyper-V Module for Windows PowerShell,
                  Hyper-V GUI Management Tools, Remote Server Administration Tools,
                  Hyper-V Management Tools, Role Administration Tools}
ExitCode       : SuccessRestartRequired

WARNING: You must restart this server to finish the installation process

PSComputerName : HV1
RunspaceId     : 3f0df3e9-33bd-4cbb-9b5e-63454533d361
Success        : True
RestartNeeded  : Yes
FeatureResult   : {Hyper-V, Hyper-V Module for Windows PowerShell,
                  Hyper-V GUI Management Tools, Remote Server Administration Tools,
                  Hyper-V Management Tools, Role Administration Tools}
ExitCode       : SuccessRestartRequired
```

```

PS C:\Foo> # 9. Reviewing key VM host settings
PS C:\Foo> $CheckVMHostSB = {
    Get-VMHost
}
PS C:\Foo> $Properties = 'Name', 'V*Path', 'Numasp*', 'Ena*', 'RES*'
PS C:\Foo> Invoke-Command -Scriptblock $CheckVMHostSB -Session $Sessions |
    Format-Table -Property $Properties

```

Name	VirtualHardDiskPath	VirtualMachinePath	NumaSpanningEnabled	EnableEnhancedSessionMode	ResourceMeteringSaveInterval
HV2	C:\VM\VHDS	C:\VM\VMS	True	True	01:00:00
HV1	C:\VM\VHDS	C:\VM\VMS	True	True	01:00:00

```

PS C:\Foo> # 2. Creating a new VM
PS C:\Foo> New-VM -Name $VMName -Path $VMLocation -MemoryStartupBytes 1GB

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status
PSDirect	Off	0	0	00:00:00	Operating normally 10.0

```

PS C:\Foo> # 7. Viewing the VM
PS C:\Foo> Get-VM -Name $VMName

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	4	1024	00:00:21.8460000	Operating normally	10.0

```

PS C:\Foo> # 2. Viewing the PSDirect VM
PS C:\Foo> Get-VM -Name PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	0	1024	02:18:03.4980000	Operating normally	10.0

```

PS C:\Foo> # 3. Invoking a command on the VM specifying VM name
PS C:\Foo> $CommandHT = @{
    VMName      = 'PSDirect'
    Credential   = $Cred
    ScriptBlock  = {HOSTNAME.EXE}
}
PS C:\Foo> Invoke-Command @CommandHT
WIN-LRBCORRT59T

```

```

PS C:\Foo> # 4. Invoking a command based on VMID
PS C:\Foo> $VMID = (Get-VM -VMName PSDirect).VMId.Guid
PS C:\Foo> Invoke-Command -VMid $VMID -Credential $Cred -ScriptBlock {ipconfig}

```

Windows IP Configuration

Ethernet adapter Ethernet:

```

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

```



```

PS C:\Foo> # 5. Entering a PS remoting session with the psdirect VM
PS C:\Foo> Enter-PSSession -VMName PSDirect -Credential $Cred
[PSDirect]: PS C:\Users\Administrator\Documents> Get-CimInstance -Class Win32_ComputerSystem

Name                PrimaryOwnerName Domain      TotalPhysicalMemory Model              Manufacturer
-----                -
WIN-LKBCORRT59T     Windows User      WORKGROUP  1073270784          Virtual Machine    Microsoft Corporation

[PSDirect]: PS C:\Users\Administrator\Documents> Exit

```

```

PS C:\Foo> # 8. Entering an interactive session with HV1
PS C:\Foo> Enter-PSSession $RS
[HV1]: PS C:\Users\Administrator\Documents>

```

```

[HV1]: PS C:\Users\Administrator\Documents> # 10. Entering and using the remoting session inside PSDirect
[HV1]: PS C:\Users\Administrator\Documents> $PSDRS = New-PSSession -VMName PSDirect -Credential $Cred
[HV1]: PS C:\Users\Administrator\Documents> Enter-PSSession -Session $PSDRS
[HV1]: [PSDirect]: PS C:\Users\Administrator\Documents> HOSTNAME.EXE
WIN-LRBCORRT59T
[HV1]: [PSDirect]: PS C:\Users\Administrator\Documents>

```

```

[HV1]: [PSDirect]: PS C:\Users\Administrator\Documents> # 11. Closing sessions
[HV1]: [PSDirect]: PS C:\Users\Administrator\Documents> Exit-PSSession # exits from session to PSDirect
[HV1]: PS C:\Users\Administrator\Documents> Exit-PSSession # exits from session to HV1
PS C:\Foo>

```

```

PS C:\Foo> # 1. Creating five VMs on HV2
PS C:\Foo> $VMLocation = 'C:\Vm\VMs' # Created in earlier recipe
PS C:\Foo> # Create SQLAcct1
PS C:\Foo> $VMN1 = 'SQLAcct1'
PS C:\Foo> New-VM -Name $VMN1 -Path "$VMLocation\$VMN1"

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLAcct1	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # Create SQLAcct2
PS C:\Foo> $VMN2 = 'SQLAcct2'
PS C:\Foo> New-VM -Name $VMN2 -Path "$VMLocation\$VMN2"

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLAcct2	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # Create SQLAcct3
PS C:\Foo> $VMN3 = 'SQLAcct3'
PS C:\Foo> New-VM -Name $VMN3 -Path "$VMLocation\$VMN3"

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLAcct3	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # Create SQLMfg1
PS C:\Foo> $VMN4 = 'SQLMfg1'
PS C:\Foo> New-VM -Name $VMN4 -Path "$VMLocation\$VMN4"

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLMfg1	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # Create SQLMfg2
PS C:\Foo> $VMN5 = 'SQLMfg2'
PS C:\Foo> New-VM -Name $VMN5 -Path "$VMLocation\$VMN5"

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLMfg2	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # 2. Viewing SQL VMs
PS C:\Foo> Get-VM -Name SQL*

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLAcct3	Off	0	0	00:00:00	Operating normally	10.0
SQLMfg2	Off	0	0	00:00:00	Operating normally	10.0
SQLMfg1	Off	0	0	00:00:00	Operating normally	10.0
SQLAcct1	Off	0	0	00:00:00	Operating normally	10.0
SQLAcct2	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 4. Displaying the VM groups on HV2
PS C:\Foo> Get-VMGroup |
    Format-Table -Property Name, *Members, ComputerName
```

Name	VMMembers	VMGroupMembers	ComputerName
SQLMfgVMG	{}		HV2
SQLAccVMG	{}		HV2

```
PS C:\Foo> # 8. Viewing VM groups on HV2
PS C:\Foo> Get-VMGroup |
    Format-Table -Property Name, *Members, ComputerName
```

Name	VMMembers	VMGroupMembers	ComputerName
SQLMfgVMG	{SQLMfg2, SQLMfg1}		HV2
SQLAccVMG	{SQLAcct3, SQLAcct1, SQLAcct2}		HV2

```
PS C:\Foo> # 12. Viewing VM groups by type
PS C:\Foo> Get-VMGroup | Sort-Object -Property GroupType |
    Format-Table -Property Name, GroupType, VMGroupMembers,
    VMMembers
```

Name	GroupType	VMGroupMembers	VMMembers
SQLMfgVMG	VMCollectionType		{SQLMfg2, SQLMfg1}
SQLAccVMG	VMCollectionType		{SQLAcct3, SQLAcct1, SQLAcct2}
VMMGSQL	ManagementCollectionType	{SQLMfgVMG, SQLAccVMG}	

```
PS C:\Foo> # 16. Checking processor counts for all VMs sorted by CPU count
PS C:\Foo> $VMS = (Get-VMGroup -Name VMMGSQL).VMGroupMembers.VMMembers
PS C:\Foo> Get-VMProcessor -VMName $VMS.Name |
    Sort-Object -Property Count -Descending |
    Format-Table -Property VMName, Count
```

VMName	Count
SQLAcct3	6
SQLAcct1	6
SQLAcct2	6
SQLMfg2	4
SQLMfg1	4

```
PS C:\Foo> # 1. Turning off the PSDirect VM
PS C:\Foo> Stop-VM -VMName PSDirect
PS C:\Foo> Get-VM -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # 2. Setting the startup order in the VM's BIOS
PS C:\Foo> $Order = 'IDE','CD','LegacyNetworkAdapter','Floppy'
PS C:\Foo> Set-VMbios -VMName PSDirect -StartupOrder $Order
PS C:\Foo> Get-VMbios PSDirect

```

VMName	StartupOrder	NumLockEnabled
PSDirect	{IDE, CD, LegacyNetworkAdapter, Floppy}	False

```

PS C:\Foo> # 3. Setting and viewing CPU count for PSDirect
PS C:\Foo> Set-VMProcessor -VMName PSDirect -Count 2
PS C:\Foo> Get-VMProcessor -VMName PSDirect |
    Format-Table VMName, Count

```

VMName	Count
PSDirect	2

```

PS C:\Foo> # 4. Setting and viewing PSDirect memory
PS C:\Foo> $VMConfigurationHT = [ordered] @{
    VMName                = 'PSDirect'
    DynamicMemoryEnabled  = $true
    MinimumBytes           = 512MB
    StartupBytes           = 1GB
    MaximumBytes           = 2GB
}
PS C:\Foo> Set-VMemory @VMConfigurationHT
PS C:\Foo> Get-VMemory -VMName PSDirect

```

VMName	DynamicMemoryEnabled	Minimum(M)	Startup(M)	Maximum(M)
PSDirect	True	512	1024	2048

```

PS C:\Foo> # 5. Adding and viewing a ScsiController in the PSDirect VM
PS C:\Foo> Add-VMScsiController -VMName PSDirect
PS C:\Foo> Get-VMScsiController -VMName PSDirect

```

VMName	ControllerNumber	Drives
PSDirect	0	{}
PSDirect	1	{}



```

PS C:\Foo> # 7. Creating a new VHDX file for the PSDirect VM
PS C:\Foo> $VHDPATH = 'C:\VM\VHDs\PSDirect-D.VHDX'
PS C:\Foo> New-VHD -Path $VHDPATH -SizeBytes 8GB -Dynamic

```

```

ComputerName      : HV1
Path              : C:\VM\VHDs\PSDirect-D.VHDX
VhdFormat         : VHDX
VhdType           : Dynamic
FileSize          : 4194304
Size              : 8589934592
MinimumSize       :
LogicalSectorSize : 512
PhysicalSectorSize : 4096
BlockSize         : 33554432
ParentPath        :
DiskIdentifier     : 49F1E70B-A66D-4F60-A536-53FB87042BAC
FragmentationPercentage : 0
Alignment         : 1
Attached          : False
DiskNumber        :
IsPMEMCompatible  : False
AddressAbstractionType : None
Number            :

```

```

PS C:\Foo> # 10. Viewing drives in the PSDirect VM
PS C:\Foo> Get-VMScsiController -VMName PSDirect
Select-Object -ExpandProperty Drives

```

VMName	ControllerType	ControllerNumber	ControllerLocation	DiskNumber	Path
PSDirect	SCSI	0	0		C:\VM\VHDs\PSDirect-D.VHDX

```

PS C:\Foo> # 2. Getting NIC details and any IP addresses from the PSDirect VM
PS C:\Foo> Get-VMNetworkAdapter -VMName PSDirect

```

Name	IsManagementOs	VMName	SwitchName	MacAddress	Status	IPAddresses
Network Adapter	False	PSDirect		00155D0AC900	{0k}	{169.254.17.230, fe80::351e:a5ce:b8d7:11e6}

```

PS C:\Foo> # 3. Creating a credential then getting VM networking details
PS C:\Foo> $RKAdministrator = 'localhost\Administrator'
PS C:\Foo> $Password = 'Pa$$w0rd'
PS C:\Foo> $RKPassword = ConvertTo-SecureString -String $Password -AsPlainText -Force
PS C:\Foo> $RKCred = [System.Management.Automation.PSCredential]::new(
                                $RKAdministrator,
                                $RKPassword)

PS C:\Foo> $VMHT = @{
    VMName      = 'PSDirect'
    ScriptBlock = {ipconfig}
    Credential   = $RKCred
}

PS C:\Foo> Invoke-Command @VMHT

```

Windows IP Configuration

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected  
 Connection-specific DNS Suffix . :

```

PS C:\Foo> # 4. Creating a virtual switch on HV1
PS C:\Foo> $VirtSwitchHT = @{
    Name              = 'iNTGERNAL'
    NetAdapterName    = 'Ethernet'
    Notes              = 'Created on HV1'
}

PS C:\Foo> New-VMSwitch @VirtSwitchHT

```

Name	SwitchType	NetAdapter	InterfaceDescription
Internal	External	Microsoft	Hyper-V Network Adapter

```

PS C:\Foo> # 6. Viewing VM networking inFormation
PS C:\Foo> Get-VMNetworkAdapter -VMName PSDirect

```

Name	IsManagementOs	VMName	SwitchName	MacAddress	Status	IPAddresses
Network Adapter	False	PSDirect	External	00155D0AC900	{OK}	{10.10.10.179, fe80::351e:a5ce:b8d7:11e6}

```
PS C:\Foo> # 7. Observing the IP address in the PSDirect VM
PS C:\Foo> $CommandHT = @{
    VMName      = 'PSDirect'
    ScriptBlock = {ipconfig}
    Credential   = $RKCred
}
PS C:\Foo> Invoke-Command @CommandHT
```

Windows IP Configuration

Ethernet adapter Ethernet 2:

```
Connection-specific DNS Suffix  . : Reskit.Org
Link-local IPv6 Address . . . . . : fe80::94d5:e366:7bf7:320a%7
IPv4 Address. . . . . : 10.10.10.179
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
```

```
PS C:\Foo> # 8. Viewing the hostname on PSDirect
PS C:\Foo> $CommandHT.ScriptBlock = {hostname}
PS C:\Foo> Invoke-Command @CommandHT
WIN-LKBCØRRT59T
```

```
PS C:\Foo> # 9. Changing the name of the host in the PSDirect VM
PS C:\Foo> $CommandHT.ScriptBlock = {Rename-Computer -NewName Wolf -Force}
PS C:\Foo> Invoke-Command @CommandHT
WARNING: The changes will take effect after you restart the computer WIN-LKBCØRRT59T.
```

```
PS C:\Foo> # 11. Getting hostname of the PSDirect VM
PS C:\Foo> $CommandHT.ScriptBlock = {HOSTNAME}
PS C:\Foo> Invoke-Command @CommandHT
Wolf
```

```

PS C:\Foo> # 2. Setting the VM's processor to support virtualization
PS C:\Foo> $VMHT = @{
    VMName = 'PSDirect'
    ExposeVirtualizationExtensions = $true
}
PS C:\Foo> Set-VMProcessor @VMHT
PS C:\Foo> Get-VMProcessor -VMName PSDirect |
    Format-Table -Property Name, Count,
        ExposeVirtualizationExtensions

```

Name	Count	ExposeVirtualizationExtensions
Processor	2	True

```

PS C:\Foo> # 3. Starting the PSDirect VM
PS C:\Foo> Start-VM -VMName PSDirect
PS C:\Foo> Wait-VM -VMName PSDirect -For Heartbeat
PS C:\Foo> Get-VM -VMName PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	23	1024	00:00:35.7060000	Operating normally	10.0

```

PS C:\Foo> # 7. Installing Hyper-V inside PSDirect
PS C:\Foo> $InstallHT = @{
    Session = $Session
    ScriptBlock = $ScriptBlock
}
PS C:\Foo> Invoke-Command @InstallHT

```

```

PSComputerName : PSDirect
RunspaceId      : 2449adca-f238-4333-a0c2-425f4e7e2f2a
Success         : True
RestartNeeded   : Yes
FeatureResult    : {Hyper-V, Hyper-V Module for Windows PowerShell,
                    Hyper-V GUI Management Tools, Remote Server Administration Tools,
                    Hyper-V Management Tools, Role Administration Tools}
ExitCode        : SuccessRestartRequired

```

WARNING: You must restart this server to finish the installation process.

```

PS C:\Foo> # 8. Restarting the VM to finish adding Hyper-V to PSDirect
PS C:\Foo> Stop-VM -VMName PSDirect
PS C:\Foo> Start-VM -VMName PSDirect
PS C:\Foo> Wait-VM -VMName PSDirect -For IPAddress
PS C:\Foo> Get-VM -VMName PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	29	1024	00:00:41.8930000	Operating normally	10.0



```

PS C:\Foo> # 9. Creating a nested VM inside the PSDirect VM
PS C:\Foo> $ScriptBlock2 = {
    $VMName = 'NestedVM'
    New-VM -Name $VMName -MemoryStartupBytes 1GB | Out-Null
    Get-VM
}
PS C:\Foo> $InstallHT2 = @{
    VMName = 'PSDirect'
    ScriptBlock = $ScriptBlock2
}
PS C:\Foo> Invoke-Command @InstallHT2 -Credential $Cred

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version	PSComputerName
NestedVM	Off	0	0	00:00:00	Operating normally	10.0	PSDirect

```

PS C:\Foo> # 1. Getting the VM's state to check if it is off
PS C:\Foo> Stop-VM -Name PSDirect -WarningAction SilentlyContinue
PS C:\Foo> Get-VM -Name PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```

PS C:\Foo> # 2. Starting the VM
PS C:\Foo> Start-VM -VMName PSDirect
PS C:\Foo> Wait-VM -VMName PSDirect -For IPAddress
PS C:\Foo> Get-VM -VMName PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	27	1024	00:00:42.4010000	Operating normally	10.0

```

PS C:\Foo> # 3. Suspending and viewing the PSDirect VM
PS C:\Foo> Suspend-VM -VMName PSDirect
PS C:\Foo> Get-VM -VMName PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Paused	0	1024	00:01:16.8060000	Operating normally	10.0

```

PS C:\Foo> # 4. Resuming the PSDirect VM
PS C:\Foo> Resume-VM -VMName PSDirect
PS C:\Foo> Get-VM -VMName PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	0	1024	00:01:17.1490000	Operating normally	10.0

```

PS C:\Foo> # 5. Saving the VM
PS C:\Foo> Save-VM -VMName PSDirect
PS C:\Foo> Get-VM -VMName PSDirect

```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Saved	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 6. Resuming the saved VM and viewing the status
PS C:\Foo> Start-VM -VMName PSDirect
PS C:\Foo> Get-VM -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	0	1024	00:00:00.1320000	Operating normally	10.0

```
PS C:\Foo> # 7. Restarting the PSDirect VM
PS C:\Foo> Restart-VM -VMName PSDirect -Force
PS C:\Foo> Get-VM -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	0	1024	00:00:00.2240000	Operating normally	10.0

```
PS C:\Foo> # 8. Waiting for the PSDirect VM to get an IP address
PS C:\Foo> Wait-VM -VMName PSDirect -For IPaddress
PS C:\Foo> Get-VM -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	5	1024	00:00:52.6320000	Operating normally	10.0

```
PS C:\Foo> # 9. Performing a hard power off on the PSDirect VM
PS C:\Foo> Stop-VM -VMName PSDirect -TurnOff
PS C:\Foo> Get-VM -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 1. Viewing the PSDirect VM on HV1 and verifying that it is turned on and running
PS C:\Foo> Start-VM -VMName PSDirect
PS C:\Foo> Get-VM -Name PSDirect -Computer HV1
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	0	1024	00:00:00.3600000	Operating normally	10.0

```
PS C:\Foo> # 2. Getting the VM configuration location
PS C:\Foo> (Get-VM -Name PSDirect).ConfigurationLocation
C:\VM\VMS\PSDirect
```

```
PS C:\Foo> # 3. Getting the virtual hard drive locations
PS C:\Foo> Get-VMHardDiskDrive -VMName PSDirect |
Format-Table -Property VMName, ControllerType, Path
```

VMName	ControllerType	Path
PSDirect	IDE	C:\VM\VHDS\PSDirect.Vhdx
PSDirect	SCSI	C:\VM\VHDS\PSDirect-D.VHDX

```
PS C:\Foo> # 5. Viewing the configuration details after moving the VM's storage
PS C:\Foo> (Get-VM -Name PSDirect).ConfigurationLocation C:\PSDirectNew
PS C:\Foo> Get-VMHardDiskDrive -VMName PSDirect |
    Format-Table -Property VMName, ControllerType, Path
```

VMName	ControllerType	Path
PSDirect	IDE	C:\PSDirectNew\Virtual Hard Disks\PSDirect.Vhdx
PSDirect	SCSI	C:\PSDirectNew\Virtual Hard Disks\PSDirect-D.VHDX

```
PS C:\Foo> # 6. Getting the VM details for VMs from HV2
PS C:\Foo> Get-VM -ComputerName HV2
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
SQLAcct1	Off	0	0	00:00:00	Operating normally	10.0
SQLAcct2	Off	0	0	00:00:00	Operating normally	10.0
SQLAcct3	Off	0	0	00:00:00	Operating normally	10.0
SQLMfg1	Off	0	0	00:00:00	Operating normally	10.0
SQLMfg2	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 7. Creating Internal virtual switch on HV2
PS C:\Foo> $ScriptBlock = {
    $NSHT = @{
        Name = 'Internal'
        NetAdapterName = 'Ethernet'
        ALLOWmAnagementOS = $true
    }
    New-VMSwitch @NSHT
}
PS C:\Foo> Invoke-Command -ScriptBlock $ScriptBlock -ComputerName HV2
```

Name	SwitchType	NetAdapterInterfaceDescription	PSComputerName
Internal	External	Microsoft Hyper-V Network Adapter	HV2

```
PS C:\Foo> # 11. Displaying the time taken to migrate
PS C:\Foo> $OS = "Migration took: [{0:n2}] minutes"
PS C:\Foo> ($OS -f (($Finish-$Start).TotalMinutes))
Migration took: [1.16] minutes
```

```
PS C:\Foo> # 13. Checking the VMs on HV2
PS C:\Foo> Get-VM -ComputerName HV2
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	0	830	00:06:34.0690000	Operating normally	10.0
SQLAcct1	Off	0	0	00:00:00	Operating normally	10.0
SQLAcct2	Off	0	0	00:00:00	Operating normally	10.0
SQLAcct3	Off	0	0	00:00:00	Operating normally	10.0
SQLMfg1	Off	0	0	00:00:00	Operating normally	10.0
SQLMfg2	Off	0	0	00:00:00	Operating normally	10.0



```
PS C:\Foo> # 14. Looking at the details of the PSDirect VM on HV2
PS C:\Foo> ((Get-VM -Name PSDirect -Computer HV2).ConfigurationLocation)
C:\PSDirect
PS C:\Foo> Get-VMHardDiskDrive -VMName PSDirect -Computer HV2 |
Format-Table -Property VMName, Path
```

VMName	Path
PSDirect	C:\PSDirect\Virtual Hard Disks\PSDirect.Vhdx
PSDirect	C:\PSDirect\Virtual Hard Disks\PSDirect-D.VHDX

```
PS C:\Foo> # 16. Displaying the time taken to migrate back to HV1
PS C:\Foo> $OS = "Migration back to HV1 took: [{0:n2}] minutes"
PS C:\Foo> ($OS -F ($($fINISH2 - $Start2).TotalMinutes))
Migration back to HV1 took: [0.96] minutes
```

```
PS C:\Foo> # 5. Viewing the replication status of HV1
PS C:\Foo> Get-VMReplicationServer -ComputerName HV1
```

RepEnabled	AuthType	KerbAuthPort	CertAuthPort	AllowAnyServer
True	Kerb	42000	443	True

```
PS C:\Foo> # 6. Checking PSDirect on Hyper-V hosts
PS C:\Foo> Get-VM -ComputerName HV1 -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Running	3	1024	00:11:43.8010000	Operating normally	10.0

```
PS C:\Foo> Get-VM -ComputerName HV2 -VMName PSDirect
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 8. Examining the initial replication state on HV1 just after
PS C:\Foo> # you start the initial replication
PS C:\Foo> Measure-VMReplication -ComputerName HV1
```

VMName	State	Health	LReplTime	PReplSize(M)	AvgLatency	AvgReplSize(M)	Relationship
PSDirect	InitialReplicationInProgress	Normal		7,650.41		0.00	Simple

```
PS C:\Foo> # 9. Examining the replication status on HV1 after replication completes
PS C:\Foo> Measure-VNReplication -ComputerName HV1
```

VMName	State	Health	LReplTime	PReplSize(M)	AvgLatency	AvgReplSize(M)	Relationship
PSDirect	Replicating	Normal	28/09/2022 12:14:53	16.01	00:01:24	2,248.00	Simple



```
PS C:\Foo> # 11. Viewing the status of PSDirect VMs on HV2
PS C:\Foo> $VMsOnHV1 = Get-VM -ComputerName HV2 -VMName PSDirect*
PS C:\Foo> $VMsOnHV1
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect - Test	Running	3	1024	00:05:34.2340000	Operating normally	10.0
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 12. Examining networking on PS Direct Test VM
PS C:\Foo> $TestVM = $VMsOnHV1 | Where-Object Name -Match 'Test'
PS C:\Foo> Get-VMNetworkAdapter -CimSession HV2 -VMName $TestVM.Name
```

Name	IsManagementOs	VMName	SwitchName	MacAddress	Status	IPAddresses
Network Adapter	False	PSDirect - Test		00155D0ACA03	{0k}	{169.254.12.154, fe80::904d:130b:5e89:c9a}

```
PS C:\Foo> # 18. Viewing VM Status on both Hyper-V Hosts
PS C:\Foo> Get-VM -ComputerName HV1 -VMName PSDirect*
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> Get-VM -ComputerName HV2 -VMName PSDirect*
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status	Version
PSDirect	Off	0	0	00:00:00	Operating normally	10.0

```
PS C:\Foo> # 20. Checking PSDirect VM networking for HV2
PS C:\Foo> Get-VMNetworkAdapter -ComputerName HV2 -VMName PSDirect
```

Name	IsManagementOs	VMName	SwitchName	MacAddress	Status	IPAddresses
Network Adapter	False	PSDirect		00155D0ACA04	{0K}	{169.254.1.162, fe80::3d85:2893:e7f2:1a2}

```
PS C:\Foo> # 22. Checking PSDirect VM networking for HV2
PS C:\Foo> Get-VMNetworkAdapter -ComputerName HV2 -VMName PSDirect
```

Name	IsManagementOs	VMName	SwitchName	MacAddress	Status	IPAddresses
Network Adapter	False	PSDirect	Internal	00155D0ACA04	{0k}	{10.10.10.150, fe80::3d85:2893:e7f2:1a2}

```

PS C:\Foo> # 3. Examining the C:\ in the PSDirect VM before we start
PS C:\Foo> $ScriptBlock = { Get-ChildItem -Path C:\ | Format-Table}
PS C:\Foo> $InvocationHT = @{
    VMName      = 'PSDirect'
    ScriptBlock = $ScriptBlock
    Credential   = $RKCred
}
PS C:\Foo> Invoke-Command @InvocationHT

```

Directory: C:\

Mode	LastWriteTime		Length	Name
d-----	5/8/2021	1:20 AM		PerfLogs
d-r----	9/26/2022	9:47 AM		Program Files
d-----	5/8/2021	2:39 AM		Program Files (x86)
d-r----	9/16/2022	5:50 AM		Users
d-----	9/17/2022	5:55 AM		Windows

```

PS C:\Foo> # 5. Examining the files created to support the checkpoints
PS C:\Foo> $Parent = Split-Path -Parent (Get-VM -Name PSDirect |
    Select-Object -ExpandProperty HardDrives).Path |
    Select-Object -First 1
PS C:\Foo> Get-ChildItem -Path $Parent

```

Directory: C:\VM\VHDS\PSDirect\Virtual Hard Disks

Mode	LastWriteTime		Length	Name
-a----	18/10/2022	17:23	37748736	PSDirect_FB257654-AAF1-4BF0-815D-BA73754FA7FA.avhdx
-a----	18/10/2022	17:22	4194304	PSDirect-D_13E2C8CF-E339-4D79-B2B9-9414EB86AB3F.avhdx
-a----	18/10/2022	17:13	4194304	PSDirect-D.VHDX
-a----	18/10/2022	17:22	11815354368	PSDirect.Vhdx

```

PS C:\Foo> # 6. Creating some content in a file on PSDirect and displaying it
PS C:\Foo> $ScriptBlock = {
    $FileName = 'C:\File_After_Checkpoint_1'
    Get-Date | Out-File -FilePath $FileName
    Get-Content -Path $FileName
}
PS C:\Foo> $InvocationHT = @{
    VMName      = 'PSDirect'
    ScriptBlock = $ScriptBlock
    Credential   = $RKCred
}
PS C:\Foo> Invoke-Command @InvocationHT

```

Tuesday, October 18, 2022 4:20:55 AM

```

PS C:\Foo> # 8. Viewing the VM checkpoint details for PSDirect
PS C:\Foo> Get-VMSnapshot -VMName PSDirect

```

VMName	Name	SnapshotType	CreationTime	ParentSnapshotName
PSDirect	Snapshot1	Standard	18/10/2022 17:23:03	
PSDirect	Snapshot2	Standard	18/10/2022 17:27:54	Snapshot1

```
PS C:\Foo> # 9. Looking at the files supporting the two checkpoints
PS C:\Foo> Get-ChildItem -Path $Parent
```

Directory: C:\VM\VHDS\PSDirect\Virtual Hard Disks

Mode	LastWriteTime		Length	Name
-a---	18/10/2022	17:28	71303168	PSDirect_30F65FC1-08E3-478E-8D33-291E1001F8B4.avhdx
-a---	18/10/2022	17:27	442499072	PSDirect_FB257654-AAF1-4BF0-815D-BA73754FA7FA.avhdx
-a---	18/10/2022	17:27	4194304	PSDirect-D_13E2C8CF-E339-4D79-B2B9-9414EB86AB3F.avhdx
-a---	18/10/2022	17:27	4194304	PSDirect-D_78C6477B-5603-4639-B0A7-9C37D6063830.avhdx
-a---	18/10/2022	17:13	4194304	PSDirect-D.VHDX
-a---	18/10/2022	17:22	11815354368	PSDirect.Vhdx

```
PS C:\Foo> # 10. Creating and displaying another file in PSDirect
PS C:\Foo> # (i.e. after you have taken Snapshot2)
```

```
PS C:\Foo> $ScriptBlock2 = {
    $FileName2 = 'C:\File_After_Checkpoint_2'
    Get-Date | Out-File -FilePath $FileName2
    Get-ChildItem -Path C:\ -File | Format-Table
}
```

```
PS C:\Foo> $InvocationHT2 = @{
    VMName      = 'PSDirect'
    ScriptBlock = $ScriptBlock2
    Credential   = $RKCred
}
```

```
PS C:\Foo> Invoke-Command @InvocationHT2
```

Directory: C:\

Mode	LastWriteTime		Length	Name
-a----	10/18/2022	9:25 AM	90	File_After_Checkpoint_1
-a----	10/18/2022	9:30 AM	90	File_After_Checkpoint_2

```
PS C:\Foo> # 12. Seeing what files we have now on PSDirect
```

```
PS C:\Foo> $ScriptBlock3 = {
    Get-ChildItem -Path C:\ | Format-Table
}
```

```
PS C:\Foo> $InvocationHT3 = @{
    VMName      = 'PSDirect'
    ScriptBlock = $ScriptBlock3
    Credential   = $RKCred
}
```

```
PS C:\Foo> Invoke-Command @InvocationHT3
```

Directory: C:\

Mode	LastWriteTime		Length	Name
d-----	5/8/2021	1:20 AM		PerfLogs
d-r----	9/26/2022	9:47 AM		Program Files
d-----	5/8/2021	2:39 AM		Program Files (x86)
d-r----	9/16/2022	5:50 AM		Users
d-----	9/17/2022	5:55 AM		Windows



```

PS C:\Foo> # 14. Observe the files you now have supporting PSDirect
PS C:\Foo> $ScriptBlock4 = {
    Get-ChildItem -Path C:\ | Format-Table
}
PS C:\Foo> $InvocationHT4 = @{
    VMName      = 'PSDirect'
    ScriptBlock = $ScriptBlock4
    Credential   = $RKCred
}
PS C:\Foo> Invoke-Command @InvocationHT4

```

Directory: C:\

Mode		LastWriteTime		Length	Name
d-----		5/8/2021	1:20 AM		PerfLogs
d-r----		9/26/2022	9:47 AM		Program Files
d-----		5/8/2021	2:39 AM		Program Files (x86)
d-r----		9/16/2022	5:50 AM		Users
d-----		9/17/2022	5:55 AM		Windows
-a-----		10/18/2022	9:25 AM	90	File_After_Checkpoint_1

```

PS C:\Foo> # 16. Checking checkpoints and VM data files again
PS C:\Foo> Get-VMSnapshot -VMName PSDirect

```

VMName	Name	SnapshotType	CreationTime	ParentSnapshotName
PSDirect	Snapshot1	Standard	18/10/2022 17:23:03	
PSDirect	Snapshot2	Standard	18/10/2022 17:27:54	Snapshot1

```

PS C:\Foo> Get-ChildItem -Path $Parent | Format-Table

```

Directory: C:\VM\ VHDS\PSDirect\Virtual Hard Disks

Mode		LastWriteTime		Length	Name
-a----	18/10/2022	20:41	473956352		PSDirect_414F9F70-521F-42F0-9325-EEA16590E12F.avhdx
-a----	18/10/2022	17:27	442499072		PSDirect_FB257654-AAF1-4BF0-815D-BA73754FA7FA.avhdx
-a----	18/10/2022	17:27	4194304		PSDirect-D_13E2C8CF-E339-4D79-B2B9-9414EB86AB3F.avhdx
-a----	18/10/2022	20:40	4194304		PSDirect-D_37ADA8DF-DDB0-4F1E-9E16-CD5103C7A7A4.avhdx
-a----	18/10/2022	17:13	4194304		PSDirect-D.VHDX
-a----	18/10/2022	17:22	11815354368		PSDirect.Vhdx

```

PS C:\Foo> # 18. Checking VM data files again
PS C:\Foo> Get-ChildItem -Path $Parent

```

Directory: C:\VM\ VHDS\PSDirect\Virtual Hard Disks

Mode		LastWriteTime		Length	Name
-a----		18/10/2022	20:42	4194304	PSDirect-D.VHDX
-a----		18/10/2022	20:42	11815354368	PSDirect.Vhdx



```
PS C:\Foo> # 14. Displaying final report
PS C:\Foo> $Report
```

VM Host Details:

Name	Value
HostName	HV1
Maker	Microsoft Corporation
Model	Virtual Machine
PSVersion	7.2.6
OSEdition	Microsoft Windows Server 2022 Datacenter
OSArch	64-bit
OSLang	1033
LastBootTime	28/09/2022 11:41:47
UpTimeDays	23.171
CPUCount	6
HostCPUUsage	63
HostMemoryGB	8
AllocatedMemoryGB	1

VM Details:

VMName	Status	Uptime	VMCPUUsage	ReplMode	ReplState
PSDirect	Operating normally	2.19:06:58.0580000	6	None	Disabled
VM2	Operating normally	00:00:00	0	None	Disabled
VM3	Operating normally	00:00:00	0	None	Disabled

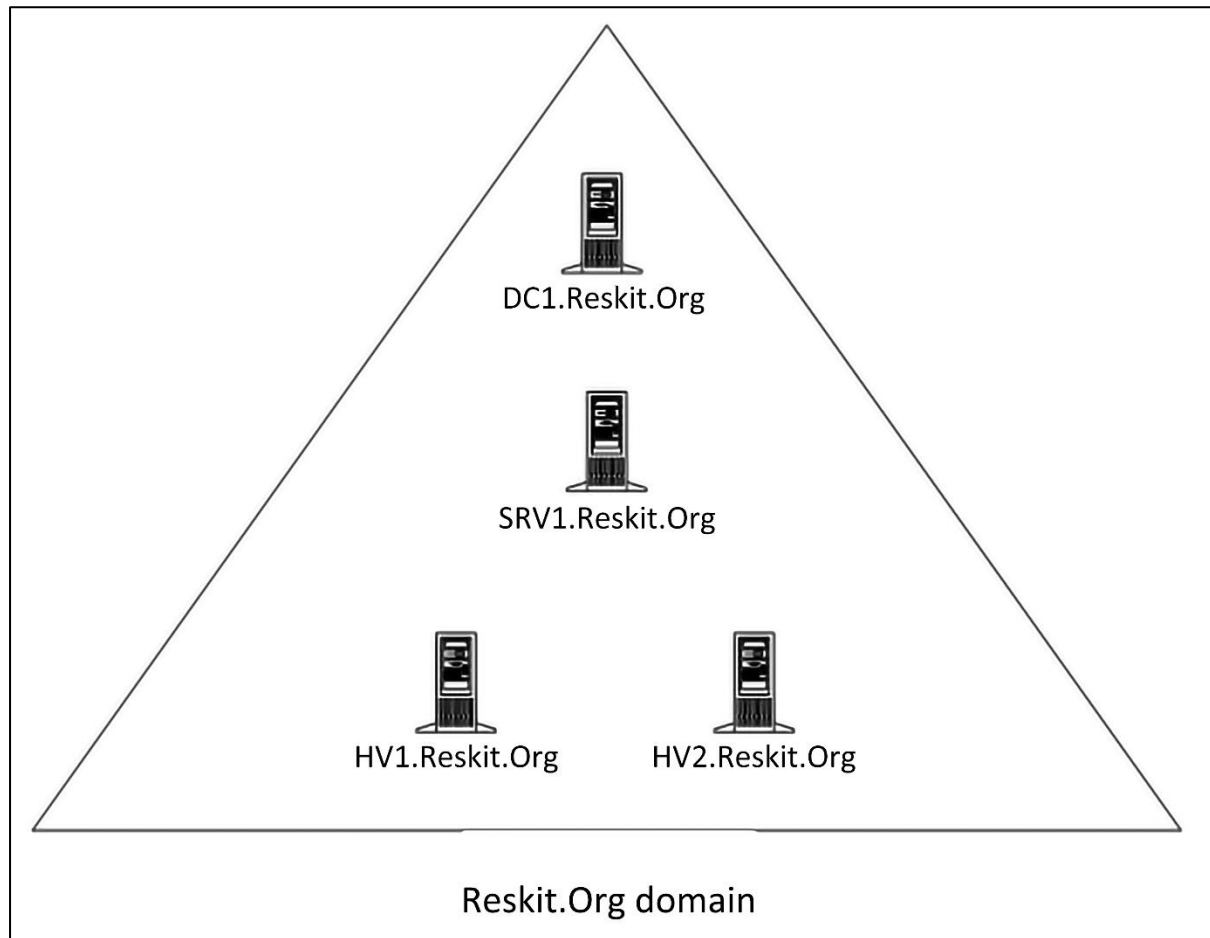
VM Host Details:

Name	Value
HostName	COOKHAM216
Maker	Dell Inc.
Model	Precision 7920 Tower
PSVersion	7.2.6
OSEdition	Microsoft Windows 11 Enterprise Insider Preview
OSArch	64-bit
OSLang	1033
LastBootTime	24/09/2022 18:02:05
UpTimeDays	27.101
CPUCount	64
HostCPUUsage	3
HostMemoryGB	127.51
AllocatedMemoryGB	27.29

VM Details:

VMName	Status	Uptime	VMCPUUsage	ReplMode	ReplState
***Cookham1	Operating normally	27.02:25:13.2360000	0	None	Disabled
CH1	Operating normally	00:00:00	0	None	Disabled
DC1	Operating normally	27.02:09:26.9510000	0	None	Disabled
DC2	Operating normally	27.02:09:26.4410000	0	None	Disabled
FS1	Operating normally	00:00:00	0	None	Disabled
FS2	Operating normally	00:00:00	0	None	Disabled
HV1	Operating normally	23.08:46:13.1760000	0	None	Disabled
HV2	Operating normally	23.08:46:17.8320000	0	None	Disabled
PSRV	Operating normally	00:00:00	0	None	Disabled
SMTP-2019	Operating normally	00:00:00	0	None	Disabled
SRV1	Operating normally	00:00:00	0	None	Disabled
SRV2	Operating normally	00:00:00	0	None	Disabled
SS1	Operating normally	00:00:00	0	None	Disabled
UKDC1	Operating normally	00:00:00	0	None	Disabled

## Chapter 12: Debugging and Troubleshooting Windows Server



```
PS C:\Foo> # 1. Discovering the Powershell Script Analyzer module
PS C:\Foo> Find-Module -Name PSScriptAnalyzer |
    Format-List Name, Version, Type, Desc*, Author, Company*, *Date, *URI*

Name           : PSScriptAnalyzer
Version        : 1.21.0
Type           : Module
Description    : PSScriptAnalyzer provides script analysis and checks for
                  potential code defects in the scripts by applying a group
                  of built-in or customized rules on the scripts being analyzed.
Author         : Microsoft Corporation
CompanyName    : {PowerShellTeam, JamesTruher-MSFT, rjmholt}
PublishedDate  : 29/09/2022 20:14:59
InstalledDate  :
UpdatedDate    :
LicenseUri     : https://github.com/PowerShell/PSScriptAnalyzer/blob/master/LICENSE
ProjectUri     : https://github.com/PowerShell/PSScriptAnalyzer
IconUri        : https://raw.githubusercontent.com/powershell/psscriptanalyzer/master/logo.png
```

```
PS C:\Foo> # 3. Discovering the commands in the Script Analyzer module
PS C:\Foo> Get-Command -Module PSScriptAnalyzer
```

CommandType	Name	Version	Source
Cmdlet	Get-ScriptAnalyzerRule	1.21.0	PSScriptAnalyzer
Cmdlet	Invoke-Formatter	1.21.0	PSScriptAnalyzer
Cmdlet	Invoke-ScriptAnalyzer	1.21.0	PSScriptAnalyzer

```
PS C:\Foo> # 4. Discovering analyzer rules
PS C:\Foo> Get-ScriptAnalyzerRule |
    Group-Object -Property Severity |
    Sort-Object -Property Count -Descending
```

Count	Name	Group
50	Warning	{PSAlignAssignmentStatement, PSAvoidUsingCmdletAliases, PSAvoidAssignmentT...
11	Information	{PSAvoidUsingPositionalParameters, PSAvoidTrailingWhitespace, PSAvoidUsing...
7	Error	{PSAvoidUsingUsernameAndPasswordParams, PSAvoidUsingComputerNameHardcoded,...

```
PS C:\Foo> # 5. Examining a rule
PS C:\Foo> Get-ScriptAnalyzerRule |
    Select-Object -First 1 |
    Format-List
```

```
Name          :
Severity       : Warning
Description    : Line up assignment statements such that the assignment operator are aligned.
SourceName     : PS
```

```
PS C:\Foo> # 7. Checking the newly created script file
PS C:\Foo> Get-ChildItem C:\Foo\Bad.ps1
```

Directory: C:\Foo

Mode	LastWriteTime	Length	Name
-a---	26/10/2022 19:38	567	Bad.ps1



```
PS C:\Foo> PS C:\Foo> # 8. Analyzing the script file
PS C:\Foo> Invoke-ScriptAnalyzer -Path C:\Foo\Bad.ps1 |
Sort-Object -Property Line
```

RuleName	Severity	ScriptName	Line	Message
PSAvoidUsingCmdletAliases	Warning	Bad.ps1	5	'gps' is an alias of 'Get-Process'. Alias can introduce possible problems and make scripts hard to maintain. Please consider changing alias to its full content.
PSUseDeclaredVarsMoreThanAssignments	Warning	Bad.ps1	5	The variable 'Procs' is assigned but never used.
PSUseDeclaredVarsMoreThanAssignments	Warning	Bad.ps1	7	The variable 'Services' is assigned but never used.
PSAvoidOverwritingBuiltInCmdlets	Warning	Bad.ps1	11	'Get-ChildItem' is a cmdlet that is included with PowerShell (version core-6.1.0-windows) whose definition should not be overridden
PSAvoidUsingComputerNameHardcoded	Error	Bad.ps1	13	The ComputerName parameter of cmdlet 'Test-Connection' is hardcoded. This will expose sensitive information about the system if the script is shared.
PSAvoidTrailingWhitespace	Information	Bad.ps1	15	Line has trailing whitespace
PSUseDeclaredVarsMoreThanAssignments	Warning	Bad.ps1	15	The variable 'foobar' is assigned but never used.
PSAvoidGlobalVars	Warning	Bad.ps1	17	Found global variable 'Global:foo'.

```
PS C:\Foo> # 11. Invoking formatter
PS C:\Foo> Invoke-Formatter -ScriptDefinition $Script1 -Settings $Settings
function foo {
    "hello!"
    Get-ChildItem -Path C:\F00
}
```

```
PS C:\Foo> # 12. Changing settings and reformatting
PS C:\Foo> $Settings.Rules.PSPlaceOpenBrace.OnSameLine = $False
PS C:\Foo> Invoke-Formatter -ScriptDefinition $Script1 -Settings $Settings
function foo
{
    "hello!"
    Get-ChildItem -Path C:\F00
}
```

```
PS C:\Foo> # 1. Getting and displaying the DNS name of this host
PS C:\Foo> $DNSDomain = $Env:USERDNSDOMAIN
PS C:\Foo> $FQDN      = "$Env:COMPUTERNAME.$DNSDomain"
PS C:\Foo> "Host FQDN: $FQDN"
Host FQDN: SRV1.RESKIT.ORG
```

```
PS C:\Foo> # 2. Getting DNS server address
PS C:\Foo> $DNSHT = @{
    InterfaceAlias = "Ethernet"
    AddressFamily  = 'IPv4'
}
PS C:\Foo> $DNSServers = (Get-DnsClientServerAddress @DNSHT).ServerAddresses
PS C:\Foo> $DNSServers
10.10.10.10
10.10.10.11
```

```

PS C:\Foo> # 3. Checking if the DNS servers are online
PS C:\Foo> Foreach ($DNSServer in $DNSServers) {
    $TestDNS = Test-NetConnection -Port 53 -ComputerName $DNSServer
    $Result = $TestDNS ? "Available" : ' Not reachable'
    "DNS Server [$DNSServer] is $Result"
}
DNS Server [10.10.10.10] is Available
DNS Server [10.10.10.11] is Available

```

```

PS C:\Foo> # 4. Defining a search for DCs in our domain
PS C:\Foo> $DNSRRName = "_ldap._tcp." + $DNSDomain
PS C:\Foo> $DNSRRName
_ldap._tcp.RESKIT.ORG

```

```

PS C:\Foo> # 5. Getting the DC SRV records
PS C:\Foo> $DCRRS = Resolve-DnsName -Name $DNSRRName -Type all |
    Where-Object IP4address -ne $null
PS C:\Foo> $DCRRS

```

Name	Type	TTL	Section	IPAddress
dc2.reskit.org	A	3600	Additional	10.10.10.11
dc1.reskit.org	A	3600	Additional	10.10.10.10

```

PS C:\Foo> # 6. Testing each DC for availability over LDAP
PS C:\Foo> ForEach ($DNSRR in $DCRRS){
    $TestDC = Test-NetConnection -Port 389 -ComputerName $DNSRR.IPAddress
    $Result = $TestDC ? 'DC Available' : 'DC Not reachable'
    "DC [$($DNSRR.Name)] at [$($DNSRR.IPAddress)] $Result for LDAP"
}
DC [dc2.reskit.org] at [10.10.10.11] DC Available for LDAP
DC [dc1.reskit.org] at [10.10.10.10] DC Available for LDAP

```

```

PS C:\Foo> # 7. Testing DC availability for SMB
PS C:\Foo> ForEach ($DNSRR in $DCRRS){
    $TestDC =
    Test-NetConnection -Port 445 -ComputerName $DNSRR.IPAddress
    $Result = $TestDC ? 'DC Available' : 'DC Not reachable'
    "DC [$($DNSRR.Name)] at [$($DNSRR.IPAddress)] $Result for SMB"
}
DC [dc2.reskit.org] at [10.10.10.11] DC Available for SMB
DC [dc1.reskit.org] at [10.10.10.10] DC Available for SMB

```

```

PS C:\Foo> # 8. Testing default gateway
PS C:\Foo> $NIC = Get-NetIPConfiguration -InterfaceAlias Ethernet
PS C:\Foo> $DGW = $NIC.IPv4DefaultGateway.NextHop
PS C:\Foo> $TestDG = Test-NetConnection $DGW
WARNING: Ping to 10.10.10.254 failed with status: DestinationHostUnreachable
PS C:\Foo> $Result = $TestDG.PingSucceeded ? "Reachable" : ' NOT Reachable'
PS C:\Foo> "Default Gateway for [$($NIC.InterfaceAlias) is [$DGW] - $Result"
Default Gateway for [Ethernet is [10.10.10.254] - NOT Reachable

```

```

PS C:\Foo> # 9. Testing a remote web site using ICMP
PS C:\Foo> $Site = "WWW.Packt.Com"
PS C:\Foo> $TestIP = Test-NetConnection -ComputerName $Site
PS C:\Foo> $ResultIP = $TestIP ? "Ping OK" : "Ping FAILED"
PS C:\Foo> "ICMP to $Site - $ResultIP"
ICMP to WWW.Packt.Com - Ping OK

```

```

PS C:\Foo> # 10. Testing a remote web site using port 80
PS C:\Foo> $TestPort80 = Test-Connection -ComputerName $Site -TcpPort 80
PS C:\Foo> $Result80 = $TestPort80 ? 'Site Reachable' : 'Site NOT reachable'
PS C:\Foo> "$Site over port 80 : $Result80"
WWW.Packt.Com over port 80 : Site Reachable

```

```

PS C:\Foo> # 11. Testing a remote web site using port 443
PS C:\Foo> $TestPort443 = Test-Connection -ComputerName $Site -TcpPort 443
PS C:\Foo> $Result443 = $TestPort443 ? 'Site Reachable' : 'Site NOT reachable'
PS C:\Foo> "$Site over port 443 : $Result443"
WWW.Packt.Com over port 443 : Site Reachable

```

```

PS C:\Foo> # 1. Finding the Get-NetView module on the PS Gallery
PS C:\Foo> Find-Module -Name Get-NetView

```

Version	Name	Repository	Description
2022.10.17.222	Get-NetView	PSGallery	Get-NetView is a tool used to simplify the collection of network configuration information for diagnosis of networking issues on Windows

```

PS C:\Foo> # 3. Checking installed version of Get-NetView
PS C:\Foo> Get-Module -Name Get-NetView -ListAvailable

```

Directory: C:\Users\administrator.RESKIT\Documents\PowerShell\Modules

ModuleType	Version	PreRelease	Name	PSEdition	ExportedCommands
Script	2022.10.17.222		Get-NetView	Core,Desk	Get-NetView



```

PS C:\Foo> # 6. Running Get-View
PS C:\Foo> Get-NetView -OutputDirectory $FolderName
Transcript started, output file is C:\NetViewOutput\msdbg.SRV1\Get-NetView.log
( 988 ms) Get-Service "*" | Sort-Object Name | Format-Table -AutoSize
( 1,181 ms) Get-Service "*" | Sort-Object Name | Format-Table -Property * -AutoSize
( 2,658 ms) Get-ChildItem HKLM:\SYSTEM\CurrentControlSet\Services\vmssmp -Recurse
Get-VMSSwitch: The Hyper-V Management Tools could not access an expected WMI class on computer 'SRV1'.
This may indicate that the Hyper-V Platform is not installed on the computer or that the version
of the Hyper-V Platform is incompatible with these management tools.

( 45 ms) Get-NetAdapterQos
( 49 ms) Get-NetAdapterQos -IncludeHidden
( 26 ms) Get-NetAdapterQos -IncludeHidden | Format-List -Property *
HNSDetail: hns service not found, skipping.
( 0 ms) [Unavailable] Get-NetIntent -ClusterName Get-Cluster
( 0 ms) [Unavailable] Get-NetIntentStatus -ClusterName Get-Cluster
( 0 ms) [Unavailable] Get-NetIntentAllGoalStates -ClusterName Get-Cluster | ConvertTo-Json -Depth 10
( 54 ms) pktmon status
( 107 ms) pktmon stop
( 16 ms) pktmon filter list
( 82 ms) pktmon list
( 59 ms) pktmon list --all
( 50 ms) pktmon list --all --include-hidden
( 46 ms) pktmon start --capture --counters-only --comp all
... remainder of output snipped for brevity

```

```

PS C:\NetViewOutput> # 7. Viewing the output folder using Get-ChildItem
PS C:\NetViewOutput> $OutputDetails = Get-ChildItem $FolderName
PS C:\NetViewOutput> $OutputDetails

```

Directory: C:\NetViewOutput

Mode	LastWriteTime	Length	Name
d----	03/11/2022 13:44		msdbg.SRV1
-a---	03/11/2022 13:44	76218062	msdbg.SRV1-2022.11.03_01.43.12.zip



```

PS C:\Foo> # 8. Viewing the output folder contents using Get-ChildItem
PS C:\Foo> $Results = $OutputDetails | Select-Object -First 1
PS C:\Foo> Get-ChildItem -Path $Results

```

Directory: C:\NetViewOutput\msdbg.SRV1

Mode	LastWriteTime	Length	Name
d----	03/11/2022 13:44		_Localhost
d----	03/11/2022 13:44		_Logs
d----	03/11/2022 13:44		802.1X
d----	03/11/2022 13:43		ATC
d----	03/11/2022 13:43		Counters
d----	03/11/2022 13:44		NetIp
d----	03/11/2022 13:44		NetNat
d----	03/11/2022 13:44		NetQoS
d----	03/11/2022 13:43		NetSetup
d----	03/11/2022 13:44		Netsh
d----	03/11/2022 13:43		NIC.5.Ethernet 2.Microsoft Hyper-V Network Adapter #2
d----	03/11/2022 13:43		NIC.7.Ethernet.Microsoft Hyper-V Network Adapter
d----	03/11/2022 13:43		NIC.Hidden
d----	03/11/2022 13:43		Pktmon
d----	03/11/2022 13:44		SMB
d----	03/11/2022 13:43		VMSwitch.Detail
-a---	03/11/2022 13:43	2233	_advfirewall.txt
-a---	03/11/2022 13:43	776	_arp.txt
-a---	03/11/2022 13:43	1783	_ipconfig.txt
-a---	03/11/2022 13:43	498394	_netcfg.txt
-a---	03/11/2022 13:43	10717	_netstat.txt
-a---	03/11/2022 13:43	69	_nmbind.txt
-a---	03/11/2022 13:43	6897	Environment.txt
-a---	03/11/2022 13:43	12811	Get-ComputerInfo.txt
-a---	03/11/2022 13:43	19460	Get-NetAdapter.txt
-a---	03/11/2022 13:43	3747	Get-NetAdapterStatistics.txt
-a---	03/11/2022 13:43	6574	Get-NetIpAddress.txt
-a---	03/11/2022 13:43	254	Get-NetLbfoTeam.txt
-a---	03/11/2022 13:43	1164	Get-NetOffloadGlobalSetting.txt
-a---	03/11/2022 13:43	894	Get-VMNetworkAdapter.txt
-a---	03/11/2022 13:43	782	Get-VMSwitch.txt
-a---	03/11/2022 13:43	354	Powercfg.txt
-a---	03/11/2022 13:43	1312	Verifier.txt

```
PS C:\NetViewOutput> # 9. Viewing IP configuration
PS C:\NetViewOutput> Get-Content -Path $Results\_ipconfig.txt
administrator @ SRV1:
PS C:\NetViewOutput> ipconfig
```

Windows IP Configuration

Ethernet adapter Ethernet:

```
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::a125:4dcc:684b:bfa2%7
IPv4 Address. . . . . : 10.10.10.50
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
```

administrator @ SRV1:

```
PS C:\NetViewOutput> ipconfig /allcompartments /all
```

Windows IP Configuration

```
=====
Network Information for Compartment 1 (ACTIVE)
=====
```

```
Host Name . . . . . : SRV1
Primary Dns Suffix . . . . . : Reskit.Org
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : Reskit.Org
```

Ethernet adapter Ethernet:

```
Connection-specific DNS Suffix . :
Description . . . . . : Microsoft Hyper-V Network Adapter
Physical Address. . . . . : 00-15-5D-01-2A-2E
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::a125:4dcc:684b:bfa2%7(Preferred)
IPv4 Address. . . . . : 10.10.10.50(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 100668765
DHCPv6 Client DUID. . . . . : 00-01-00-01-2A-BE-82-5D-00-15-5D-01-2A-2E
DNS Servers . . . . . : 10.10.10.10
                        10.10.10.11
NetBIOS over Tcpip. . . . . : Enabled
```

```
PS C:\Foo> # 2. Discovering the BPA module on DC1
PS C:\Foo> $ScriptBlock1 = {
    Get-Module -Name BestPractices -List |
    Format-Table -AutoSize
}
PS C:\Foo> Invoke-Command -Session $BPASession -ScriptBlock $ScriptBlock1
```

Directory: C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules

ModuleType	Version	Name	ExportedCommands
Manifest	1.0	BestPractices	{Get-BpaModel, Get-BpaResult, Invoke-BpaModel, Set-BpaResult}

```

PS C:\Foo> # 3. Discovering the commands in the BPA module
PS C:\Foo> $ScriptBlock2 = {
    Get-Command -Module BestPractices |
    Format-Table -AutoSize
}
PS C:\Foo> Invoke-Command -Session $BPASession -ScriptBlock $ScriptBlock2

```

CommandType	Name	Version	Source
Cmdlet	Get-BpaModel	1.0	BestPractices
Cmdlet	Get-BpaResult	1.0	BestPractices
Cmdlet	Invoke-BpaModel	1.0	BestPractices
Cmdlet	Set-BpaResult	1.0	BestPractices

```

PS C:\Foo> # 4. Discovering all available BPA models on DC1
PS C:\Foo> $ScriptBlock3 = {
    Get-BpaModel |
    Format-Table -Property Name,Id, LastScanTime -AutoSize
}
PS C:\Foo> Invoke-Command -Session $BPASession -ScriptBlock $ScriptBlock3

```

Name	Id	LastScanTime
RightsManagementServices	Microsoft/Windows/AD RMS	Never
CertificateServices	Microsoft/Windows/CertificateServices	Never
Microsoft DHCP Server Configuration Analysis Model	Microsoft/Windows/DHCP Server	Never
DirectoryServices	Microsoft/Windows/DirectoryServices	Never
Microsoft DNS Server Configuration Analysis Model	Microsoft/Windows/DNSServer	Never
File Services	Microsoft/Windows/FileServices	Never
Hyper-V	Microsoft/Windows/Hyper-V	Never
LightweightDirectoryServices	Microsoft/Windows/LightweightDirectoryServices	Never
Network Policy and Access Services (NPAS)	Microsoft/Windows/NPAS	Never
Microsoft Remote Access Server Configuration Analysis Model	Microsoft/Windows/RemoteAccessServer	Never
TerminalServices	Microsoft/Windows/TerminalServices	Never
Windows Server Update Services	Microsoft/Windows/UpdateServices	Never
Microsoft Volume Activation Configuration Analysis Model	Microsoft/Windows/VolumeActivation	Never
WebServer	Microsoft/Windows/WebServer	Never

```

PS C:\Foo> # 5. Running the BPA DirectoryServices model on DC1
PS C:\Foo> $ScriptBlock4 = {
    Invoke-BpaModel -ModelID Microsoft/Windows/DirectoryServices -Mode ALL |
    Format-Table -AutoSize
}
PS C:\Foo> Invoke-Command -Session $BPASession -ScriptBlock $ScriptBlock4

```

ModelId	SubModelId	Success	ScanTime	ScanTimeUtcOffset	Detail
Microsoft/Windows/DirectoryServices		True	04/11/2022 11:41:35	01:00:00	{DC1, DC1}



```
PS C:\Foo> # 6. Getting BPA results from DC1
PS C:\Foo> $ScriptBlock5 = {
    Get-BpaResult -ModelID Microsoft/Windows/DirectoryServices |
        Where-Object Resolution -ne $null |
        Format-List -Property Problem, Resolution
}
PS C:\Foo> Invoke-Command -Session $BPASession -ScriptBlock $ScriptBlock5

Problem      : The primary domain controller (PDC) emulator operations master in this
                forest is not configured to correctly synchronize time from a valid time
                source.
Resolution    : Set the PDC emulator master in this forest to synchronize time with a
                reliable external time source. If you have not configured a reliable time
                server (GTIMESERV) in the forest root domain, set the PDC emulator master
                days.
Resolution    : To ensure that recent system state backups are available to recover Active
                Directory data that was recently added, deleted, or modified, perform
                daily backups of all directory partitions in your forest or keep the time
                between Active Directory backups to a maximum of 8 days.

Problem      : The directory partition DC=DomainDnsZones,DC=Reskit,DC=Org on the domain
                controller DC1.Reskit.Org has not been backed up within the last 8 days.
Resolution    : To ensure that recent system state backups are available to recover Active
                Directory data that was recently added, deleted, or modified, perform
                daily backups of all directory partitions in your forest or keep the time
                between Active Directory backups to a maximum of 8 days.

Problem      : The directory partition DC=ForestDnsZones,DC=Reskit,DC=Org on the domain
                controller DC1.Reskit.Org has not been backed up within the last 8 days.
Resolution    : To ensure that recent system state backups are available to recover Active
                Directory data that was recently added, deleted, or modified, perform
                daily backups of all directory partitions in your forest or keep the time
                between Active Directory backups to a maximum of 8 days.

Problem      : The Active Directory Domain Services (AD DS) server role on the domain
                controller DC1.Reskit.Org is installed on a virtual machine (VM).
Resolution    : Make sure that the domain controller DC1.Reskit.Org complies with the best
                practice guidelines that are described in the Help to avoid performance
                issues and replication and security failures in the Active Directory
                environment.
```

```
PS C:\Foo> # 3. Executing the script
PS C:\Foo> & $FileName
In Breakpoint.ps1
Calculating Bar as [42]
```



```
PS C:\Foo> # 7. Viewing the breakpoints set in this session
PS C:\Foo>
PS C:\Foo> Get-PSBreakpoint | Format-Table -AutoSize
```

ID	Script	Line	Command	Variable	Action
1	Breakpoint.ps1	4			
2	Breakpoint.ps1		Get-CimInstance		
3	Breakpoint.ps1			M	

```
PS C:\Foo> # 8. Running the script – until the first breakpoint is hit
PS C:\Foo> & $FileName
```

In Breakpoint.ps1

Entering debug mode. Use h or ? for help.

Hit Line breakpoint on 'C:\Foo\Breakpoint.ps1:4'

At C:\Foo\Breakpoint.ps1:4 char:3

+ \$k = \$J\*2 # NB: line 4

+ ~~~~~

[DBG]: PS C:\Foo>>

```
[DBG]: PS C:\Foo>> # 9. Viewing the value of $J from the debug console
[DBG]: PS C:\Foo>> $J
```

21

```
[DBG]: PS C:\Foo>> # 10. Viewing the value of $k from the debug console
```

```
[DBG]: PS C:\Foo>> $k
```

```
[DBG]: PS C:\Foo>>
```

```
[DBG]: PS C:\Foo>> # 11. Continuing script execution from the DBG prompt until the next breakpoint
```

```
[DBG]: PS C:\Foo>> continue
```

Hit Variable breakpoint on 'C:\Foo\Breakpoint.ps1:\$M' (Write access)

At C:\Foo\Breakpoint.ps1:5 char:3

+ \$M = \$k # NB: \$M written to

+ ~~~~~

[DBG]: PS C:\Foo>>

```
[DBG]: PS C:\Foo>> # 12. Continue script execution until the execution of Get-CimInstance
```

```
[DBG]: PS C:\Foo>> continue
```

Hit Command breakpoint on 'C:\Foo\Breakpoint.ps1:Get-CimInstance'

At C:\Foo\Breakpoint.ps1:7 char:3

+ \$BIOS = Get-CimInstance -Class Win32\_Bios

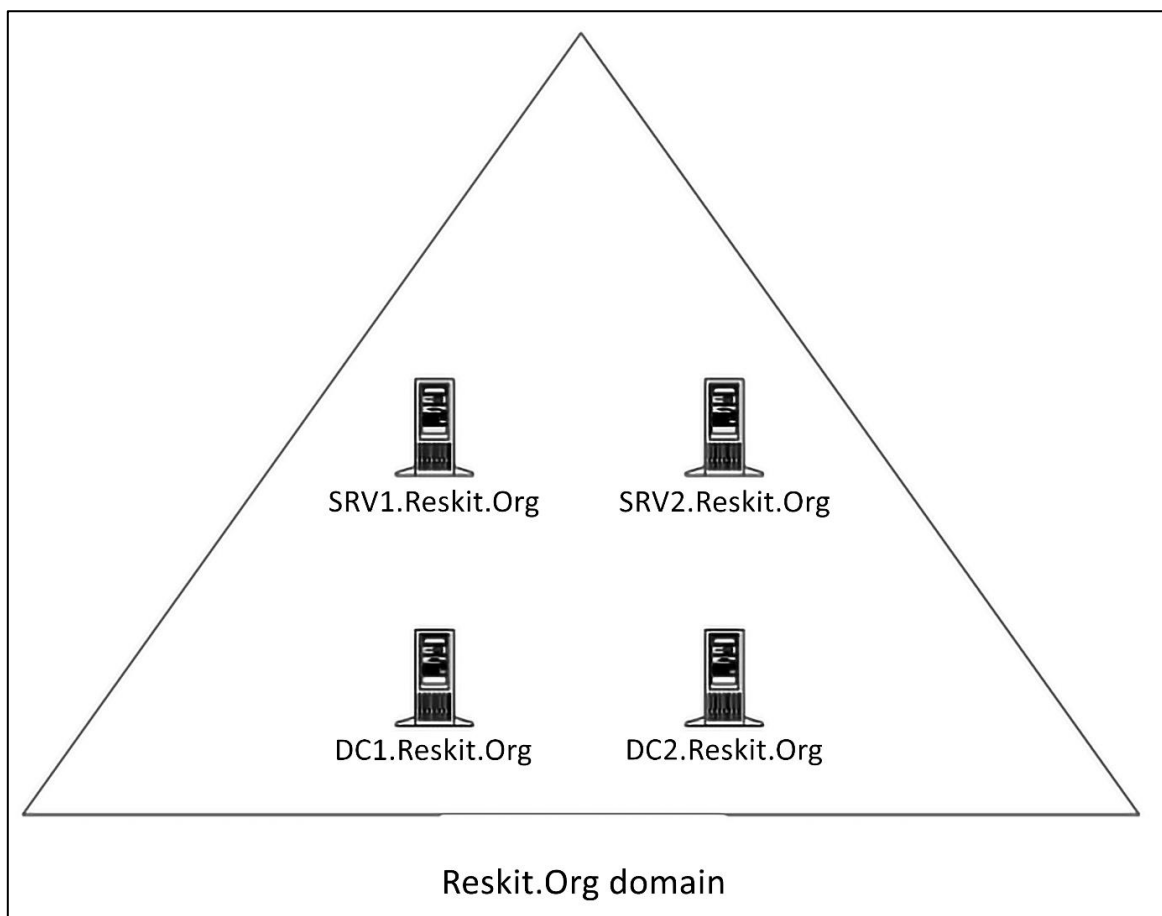
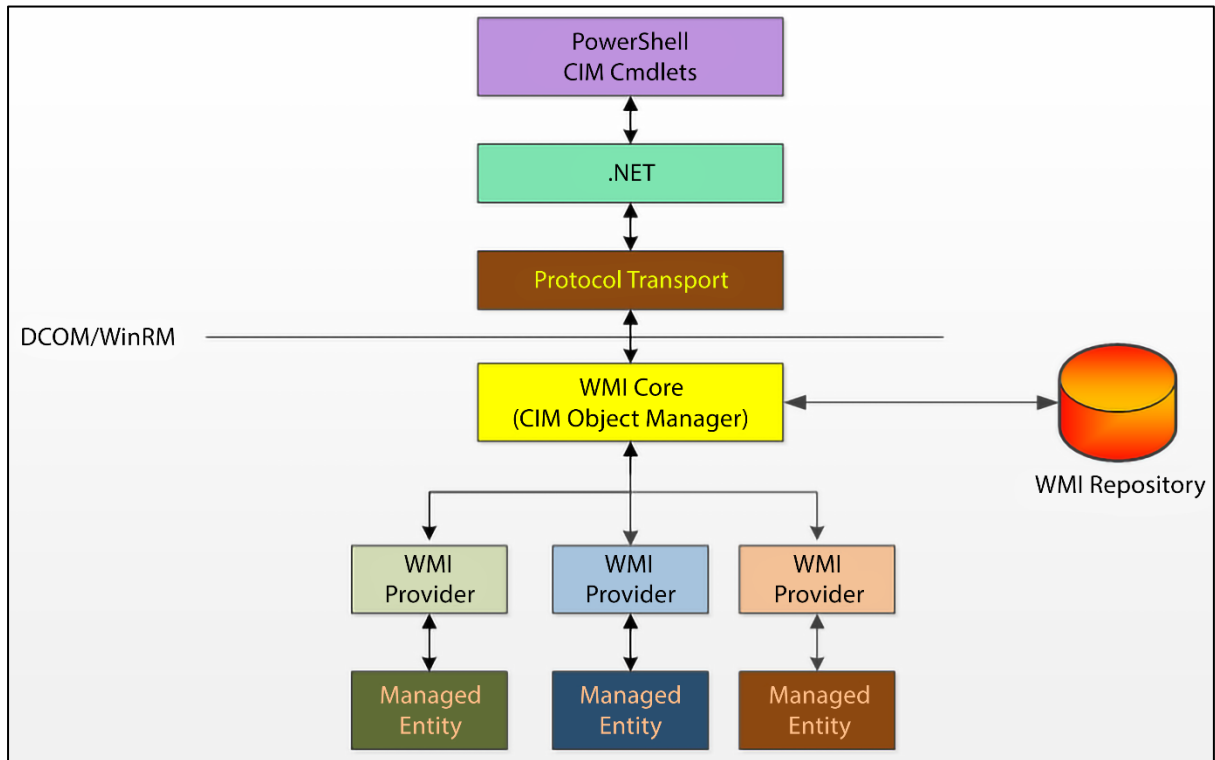
+ ~~~~~

```
[DBG]: PS C:\Foo>> # 13. Continuing script execution from until the end of the script
```

```
[DBG]: PS C:\Foo>> continue
```

Calculating Bar as [42]

## Chapter 13: Managing Window Server with Window Management Instrumentation (WMI)



```

PS C:\Foo> # 1. Viewing the WBEM folder
PS C:\Foo> $WBEMFOLDER = "$Env:windir\system32\wbem"
PS C:\Foo> Get-ChildItem -Path $WBEMFOLDER |
    Select-Object -First 20

```

Directory: C:\Windows\System32\wbem

Mode	LastWriteTime	Length	Name
d----	23/09/2022 16:35		AutoRecover
d----	08/05/2021 10:41		en
d----	22/09/2022 16:24		en-US
d----	08/05/2021 09:20		Logs
d----	22/09/2022 13:49		MOF
d----	26/10/2022 17:23		Performance
d----	24/09/2022 16:09		Repository
d----	08/05/2021 09:20		tmf
d----	08/05/2021 09:20		xml
-a---	08/05/2021 09:14	2852	aeinv.mof
-a---	08/05/2021 10:41	17510	AgentWmi.mof
-a---	08/05/2021 09:15	693	AgentWmiUninstall.mof
-a---	08/05/2021 09:14	852	appbackgroundtask_uninstall.mof
-a---	08/05/2021 09:14	65536	appbackgroundtask.dll
-a---	08/05/2021 09:14	2902	appbackgroundtask.mof
-a---	08/05/2021 09:15	1112	AttestationWmiProvider_Uninstall.mof
-a---	08/05/2021 09:15	3376	AttestationWmiProvider.mof
-a---	08/05/2021 09:14	1724	AuditRsop.mof
-a---	08/05/2021 09:14	1092	authfwcfg.mof
-a---	08/05/2021 09:14	12120	bcd.mof

```

PS C:\Foo> # 2. Viewing the WMI repository folder
PS C:\Foo> Get-ChildItem -Path $WBEMFOLDER\Repository

```

Directory: C:\Windows\System32\wbem\Repository

Mode	LastWriteTime	Length	Name
-a---	05/11/2022 22:20	5750784	INDEX.BTR
-a---	05/11/2022 19:32	105640	MAPPING1.MAP
-a---	05/11/2022 22:20	105640	MAPPING2.MAP
-a---	05/11/2022 06:52	105640	MAPPING3.MAP
-a---	05/11/2022 22:20	31899648	OBJECTS.DATA

```
PS C:\Foo> # 3. Viewing the WMI service details
PS C:\Foo> Get-Service -Name Winmgmt |
Format-List -Property *
```

```
UserName          : localSystem
Description        : Provides a common interface and object model to access management
                    : information about operating system, devices, applications and
                    : services. If this service is stopped, most Windows-based software will
                    : not function properly. If this service is disabled, any services that
                    : explicitly depend on it will fail to start.
DelayedAutoStart   : False
BinaryPathName     : C:\WINDOWS\system32\svchost.exe -k netsvcs -p
StartupType        : Automatic
Name               : Winmgmt
RequiredServices   : {RPCSS}
CanPauseAndContinue : True
CanShutdown        : True
CanStop            : True
DisplayName         : Windows Management Instrumentation
DependentServices   : {UALSVC, HgClientService}
MachineName        : .
ServiceName         : Winmgmt
ServicesDependedOn : {RPCSS}
StartType          : Automatic
ServiceHandle       : Microsoft.Win32.SafeHandles.SafeServiceHandle
Status             : Running
ServiceType         : Win32OwnProcess, Win32ShareProcess
Site               :
Container          :
```

```
PS C:\Foo> # 4. Getting process details
PS C:\Foo> $Service = tasklist.exe /svc /fi "SERVICES eq winmgmt" |
Select-Object -Last 1
PS C:\Foo> $Process = [int] ($Service.Substring(30,4))
PS C:\Foo> Get-Process -Id $Process
```

NPM(K)	PM(M)	WS(M)	CPU(s)	Id	SI	ProcessName
17	16.25	36.81	90.09	3392	0	svchost

```
PS C:\Foo> # 5. Examining DLLs loaded by the WMI service process
PS C:\Foo> Get-Process -Id $Process |
Select-Object -ExpandProperty Modules |
Where-Object ModuleName -match 'wmi' |
Format-Table -Property FileName, Description, FileVersion
```

FileName	Description	FileVersion
c:\windows\system32\wbem\wmisvc.dll	WMI	10.0.20348.1 (WinBuild.160101.0800)
C:\WINDOWS\SYSTEM32\WMICLNT.dll	WMI Client API	10.0.20348.1 (WinBuild.160101.0800)
C:\WINDOWS\system32\wbem\wmiutils.dll	WMI	10.0.20348.1 (WinBuild.160101.0800)
C:\WINDOWS\system32\wbem\wmiprvsd.dll	WMI	10.0.20348.1 (WinBuild.160101.0800)



DhcpServerPsProvider.dll	DHCP WMIv2 Provider	10.0.20348.1
DMWmiBridgeProv.dll	DM WMI Bridge Provider	10.0.20348.1
DMWmiBridgeProv1.dll	DM WMI Bridge Provider	10.0.20348.1
DnsClientPsProvider.dll	DNS Client WMIv2 Provider	10.0.20348.1
DnsServerPsProvider.dll	DNS WMIv2 Provider	10.0.20348.1
dsprov.dll	WMI DS Provider	10.0.20348.1
"EventTracingManagement.dll"	WMI Provider for ETW	10.0.20348.1
ipmiprr.dll	IPMI Provider Resource	10.0.20348.1
IPMIIPRV.dll	WMI IPMI PROVIDER	10.0.20348.1
MDMAppProv.dll	MDM Application Provider	10.0.20348.1
MDMSettingsProv.dll	MDM Settings Provider	10.0.20348.1
mgmtprovider.dll	Server Manager Managment Provider	10.0.20348.1
mistreamprov.dll	SIL Composable Streams Provider	10.0.20348.1
DtcCIMProvider.dll	DTC WMIv2 Provider	10.0.20348.1
msiprov.dll	WMI MSI Provider	10.0.20348.1
MspProv.exe	MspProv	10.0.20348.1
mttprov.dll	Management Tools Task Manager CIM Provider	10.0.20348.1
NCObjAPI	Non-COM WMI Event Provision APIs	10.0.20348.1
ndisimplatwmi.DLL	NDIS IM Platform WMI Provider	10.0.20348.1
NetAdapter.dll	Network Adapter WMI Provider	10.0.20348.1
netdacim.dll	Microsoft Direct Access WMI Provider	10.0.20348.1
NetEventPacketCapture.dll	NetEvent Packet Capture Provider	10.0.20348.1
NetNat.dll	Windows NAT WMI Provider	10.0.20348.1
netnccim.dll	DA Network Connectivity WMI Provider	10.0.20348.1
NetPeerDistCim.dll	BranchCache WMI Provider	10.0.20348.1
netswitchteamcim.DLL	VM Switch Teaming WMI Provider	10.0.20348.1
NetTCPIP.dll	TCPIP WMI Provider	10.0.20348.1
nlnccim.dll	Network Connection Profiles WMI Provider	10.0.20348.1
ntevt.dll	WMI Event Log Provider	10.0.20348.1
PLATID.DLL	WMIv2 Provider to Retrieve Platform Identifiers	10.0.20348.1
PrintManagementProvider.DLL	Print WMI Provider	10.0.20348.1
RacWmiProv.dll	Reliability Metrics WMI Provider	10.0.20348.1
RAMgmtPSPProvider.dll	RAMgmtPSPProvider	10.0.20348.1
regprov.dll	Registry Provider	10.0.20348.1
schedprov.dll	Task Scheduler WMIv2 Provider	10.0.20348.1
SDNDiagnosticsProvider.dll	SDNDiagnosticsProvider	10.0.20348.1
servercompprov.dll	Windows Server Feature WMI Provider	10.0.20348.1
sllprovider.dll	Server License Logging CIM Provider	10.0.20348.1
vdswmi.dll	WMI Provider for VDS	10.0.20348.1
viewprov.dll	WMI View Provider	10.0.20348.1
VpnClientPsProvider.dll	VPN Client WMIv2 Provider	10.0.20348.1
VSSPROV.DLL	WMI Provider for VSS	10.0.20348.1
WdacWmiProv.dll	WDAC WMI Providers	10.0.20348.1
Win32_EncryptableVolume.DLL	Win32_Encryptable Volume Provider	10.0.20348.143
Win32_Tpm.DLL	TPM WMI Provider	10.0.20348.1
WMIPCIIMA.dll	WMI Win32Ex Provider	10.0.20348.1
wmipdfs.dll	WMI DFS Provider	10.0.20348.1
WMIPDskQ.dll	WMI Provider for Disk Quota Information	10.0.20348.1
WbemPerfClass.dll	WbemPerf V2 Class Provider	10.0.20348.1
WbemPerfInst.dll	WbemPerf V2 Instance Provider	10.0.20348.1
wmipicmp.dll	WMI ICMP Echo Provider	10.0.20348.1
WMIPIPRT.dll	WBEM Provider for IP4 Routes	10.0.20348.1
wmipjobj.dll	WMI Windows Job Object Provider	10.0.20348.1
WMIPSess.dll	WMI Provider for Sessions and Connections	10.0.20348.1

```
PS C:\Foo> # 7. Examining the WmiPrvSE process
PS C:\Foo> Get-CimInstance -ClassName Win32_Bios | Out-Null
PS C:\Foo> Get-Process -Name WmiPrvSE
```

NPM(k)	PM(M)	WS(M)	CPU(s)	Id	SI	ProcessName
11	2.75	9.10	0.03	10012	0	WmiPrvSE

```
PS C:\Foo> # 8. Finding the WMI event log
PS C:\Foo> $Log = Get-WinEvent -ListLog *wmi*
PS C:\Foo> $Log
```

LogMode	MaximumSizeInBytes	RecordCount	LogName
Circular	1052672	1457	Microsoft-Windows-WMI-Activity/Operational

```
PS C:\Foo> # 9. Looking at the Event Types in the WMI log
PS C:\Foo> $Events = Get-WinEvent -LogName $Log.LogName
PS C:\Foo> $Events | Group-Object -Property LevelDisplayName
```

Count	Name	Group
728	Error	{System.Diagnostics.Eventing.Reader.EventLogRecord, System.Diagnostics.Eventing.Reader.EventLogRecord...}
729	Information	{System.Diagnostics.Eventing.Reader.EventLogRecord, System.Diagnostics.Eventing.Reader.EventLogRecord...}

```
PS C:\Foo> # 10. Examining WMI event log entries
PS C:\Foo> $Events |
    Select-Object -First 5 |
    Format-Table -Wrap
```

```
ProviderName: Microsoft-Windows-WMI-Activity
```

TimeCreated	Id	LevelDisplayName	Message
06/11/2022 20:51:39	5857	Information	CIMWin32 provider started with result code 0x0. HostProcess = wmioprse.exe; ProcessID = 10012; ProviderPath = %systemroot%\system32\wbem\cimwin32.dll
06/11/2022 20:50:53	5857	Information	WMIProv provider started with result code 0x0. HostProcess = wmioprse.exe; ProcessID = 8304; ProviderPath = %systemroot%\system32\wbem\wmiprov.dll
06/11/2022 20:50:52	5860	Information	Namespace = ROOT\CIMV2; NotificationQuery = SELECT * FROM Win32_ProcessStartTrace WHERE ProcessName = 'wsmprovhost.exe'; UserName = NT AUTHORITY\SYSTEM; ClientProcessID = 6780; ClientMachine = SRV1; PossibleCause = Temporary
06/11/2022 20:50:52	5857	Information	WMI Kernel Trace Event Provider provider started with result code 0x0. HostProcess = wmioprse.exe; ProcessID = 3392; ProviderPath = C:\Windows\System32\wbem\knlprov.dll
06/11/2022 20:50:51	5857	Information	WmiPerfClass provider started with result code 0x0. HostProcess = wmioprse.exe; ProcessID = 8304; ProviderPath = C:\Windows\System32\wbem\WmiPerfClass.dll

```
PS C:\Foo> # 11. Viewing executable programs in WBEM folder
PS C:\Foo> $Files = Get-ChildItem -Path $WBEMFOLDER\*.exe
PS C:\Foo> "{0,15} {1,-40}" -f 'File Name','Description'
PS C:\Foo> Foreach ($File in $Files){
    $Name = $File.Name
    $Desc = ($File |
        Select-Object -ExpandProperty VersionInfo).FileDescription
    "{0,15} {1,-40}" -f $Name,$Desc
}
```

File Name	Description
mofcomp.exe	The Managed Object Format (MOF) Compiler
scrcons.exe	WMI Standard Event Consumer - scripting
unsecapp.exe	Sink to receive asynchronous callbacks for WMI client application
wbemtest.exe	WMI Test Tool
WinMgmt.exe	WMI Service Control Utility
WMIADAP.exe	WMI Reverse Performance Adapter Maintenance Utility
WmiApSrv.exe	WMI Performance Reverse Adapter
WMIC.exe	WMI Commandline Utility
WmiPrvSE.exe	WMI Provider Host



```
PS C:\Foo> # 12. Examining the CimCmdlets module
PS C:\Foo> Get-Module -Name CimCmdlets |
Select-Object -ExcludeProperty Exported*
```

```
LogPipelineExecutionDetails : False
Name : CimCmdlets
Path : C:\Program Files\PowerShell\7\Microsoft.Management.Infrastructure.CimCmdlets.dll
ImplementingAssembly : Microsoft.Management.Infrastructure.CimCmdlets, Version=7.2.6.500, Culture=neutral,
PublicKeyToken=31bf3856ad364e35
```

```
PS C:\Foo> # 13. Finding cmdlets in the CimCmdlets module
PS C:\Foo> Get-Command -Module CimCmdlets
```

CommandType	Name	Version	Source
Cmdlet	Get-CimAssociatedInstance	7.0.0.0	CimCmdlets
Cmdlet	Get-CimClass	7.0.0.0	CimCmdlets
Cmdlet	Get-CimInstance	7.0.0.0	CimCmdlets
Cmdlet	Get-CimSession	7.0.0.0	CimCmdlets
Cmdlet	Invoke-CimMethod	7.0.0.0	CimCmdlets
Cmdlet	New-CimInstance	7.0.0.0	CimCmdlets
Cmdlet	New-CimSession	7.0.0.0	CimCmdlets
Cmdlet	New-CimSessionOption	7.0.0.0	CimCmdlets
Cmdlet	Register-CimIndicationEvent	7.0.0.0	CimCmdlets
Cmdlet	Remove-CimInstance	7.0.0.0	CimCmdlets
Cmdlet	Remove-CimSession	7.0.0.0	CimCmdlets
Cmdlet	Set-CimInstance	7.0.0.0	CimCmdlets

```
PS C:\Foo> # 14. Examining the .NET type returned from Get-CimInstance
PS C:\Foo> Get-CimInstance -ClassName Win32_Share | Get-Member
```

TypeName: Microsoft.Management.Infrastructure.CimInstance#root/cimv2/Win32\_Share

Name	MemberType	Definition
Dispose	Method	void Dispose(), void IDisposable.Dispose()
Equals	Method	bool Equals(System.Object obj)
GetCimSessionComputerName	Method	string GetCimSessionComputerName()
GetCimSessionInstanceId	Method	guid GetCimSessionInstanceId()
GetHashCode	Method	int GetHashCode()
GetType	Method	type GetType()
ToString	Method	string ToString()
AccessMask	Property	uint AccessMask {get;}
AllowMaximum	Property	bool AllowMaximum {get;}
Caption	Property	string Caption {get;}
Description	Property	string Description {get;}
InstallDate	Property	CimInstance#DateTime InstallDate {get;}
MaximumAllowed	Property	uint MaximumAllowed {get;}
Name	Property	string Name {get;}
Path	Property	string Path {get;}
PSComputerName	Property	string PSComputerName {get;}
Status	Property	string Status {get;}
Type	Property	uint Type {get;}
PSStatus	PropertySet	PSStatus {Status, Type, Name}

```
PS C:\Foo> # 1. Viewing WMI classes in the root namespace
PS C:\Foo> Get-CimClass -Namespace 'ROOT' |
Select-Object -First 10
```

Namespace: ROOT

CimClassName	CimClassMethods	CimClassProperties
__SystemClass	{}	{}
__thisNAMESPACE	{}	{SECURITY_DESCRIPTOR}
__CacheControl	{}	{}
__EventConsumerProviderCacheControl	{}	{ClearAfter}
__EventProviderCacheControl	{}	{ClearAfter}
__EventSinkCacheControl	{}	{ClearAfter}
__ObjectProviderCacheControl	{}	{ClearAfter}
__PropertyProviderCacheControl	{}	{ClearAfter}
__NAMESPACE	{}	{Name}
__ArbitratorConfiguration	{}	{OutstandingTasksPerUser, OutstandingTasksTotal...

```
PS C:\Foo> # 2. Viewing the __NAMESPACE class in ROOT
PS C:\Foo> Get-CimInstance -Namespace 'ROOT' -ClassName __NAMESPACE |
Sort-Object -Property Name
```

Name	PSComputerName
AccessLogging	
Appv	
CIMV2	
Cli	
DEFAULT	
directory	
Hardware	
Interop	
InventoryLogging	
Microsoft	
msdtc	
MSPS	
PEH	
Policy	
RSOP	
SECURITY	
ServiceModel	
StandardCimv2	
subscription	
WMI	

```
PS C:\Foo> # 3. Getting and counting classes in ROOT\CIMV2
PS C:\Foo> $Classes = Get-CimClass -Namespace 'ROOT\CIMV2'
PS C:\Foo> "There are $($Classes.Count) classes in ROOT\CIMV2"
There are 1196 classes in ROOT\CIMV2
```



```

PS C:\Foo> # 4. Discovering all the namespaces on SRV1
PS C:\Foo> $EAHT = @{ErrorAction = 'SilentlyContinue'}
PS C:\Foo> Function Get-WMINamespaceEnum {
    [CmdletBinding()]
    Param($NS)
    Write-Output $NS
    Get-CimInstance "__Namespace" -Namespace $NS @EAHT |
    ForEach-Object { Get-WMINamespaceEnum "$ns\$($_.name)" }
} # End of function
PS C:\Foo> $Namespaces = Get-WMINamespaceEnum 'ROOT' |
    Sort-Object
PS C:\Foo> "There are $($Namespaces.Count) WMI namespaces on SRV1"

```

There are 126 WMI namespaces on SRV1

```

PS C:\Foo> # 5. Viewing first 25 namespaces on SRV1
PS C:\Foo> $Namespaces |
    Select-Object -First 25

```

```

ROOT
ROOT\AccessLogging
ROOT\Appv
ROOT\CIMV2
ROOT\CIMV2\mdm
ROOT\CIMV2\mdm\dmmap
ROOT\CIMV2\mdm\MS_409
ROOT\CIMV2\ms_409
ROOT\CIMV2\power
ROOT\CIMV2\power\ms_409
ROOT\CIMV2\Security
ROOT\CIMV2\Security\MicrosoftTpm
ROOT\CIMV2\Security\MicrosoftVolumeEncryption
ROOT\CIMV2\TerminalServices
ROOT\CIMV2\TerminalServices\ms_409
ROOT\Cli
ROOT\Cli\MS_409
ROOT\DEFAULT
ROOT\DEFAULT\ms_409
ROOT\directory
ROOT\directory\LDAP
ROOT\directory\LDAP\ms_409
ROOT\Hardware
ROOT\Hardware\ms_409
ROOT\Interop

```

```
PS C:\Foo> # 7. Running the script block locally on SRV1
PS C:\Foo> Invoke-Command -ComputerName SRV1 -ScriptBlock $ScriptBlock
There are 126 WMI namespaces on SRV1
There are 17213 classes on SRV1
```

```
PS C:\Foo> # 8. Running the script block on SRV2
PS C:\Foo> Invoke-Command -ComputerName SRV2 -ScriptBlock $ScriptBlock
There are 102 WMI namespaces on SRV2
There are 14150 classes on SRV2
```

```
PS C:\Foo> # 9. Running the script block on DC1
PS C:\Foo> Invoke-Command -ComputerName DC1 -ScriptBlock $$ScriptBlock
There are 114 WMI namespaces on DC1
There are 16492 classes on DC1
```

```
PS C:\Foo> # 1. Viewing Win32_Share class
PS C:\Foo> Get-CimClass -ClassName Win32_Share
```

Namespace: ROOT/cimv2

CimClassName	CimClassMethods	CimClassProperties
Win32_Share	{Create, SetShareInfo, GetAccessMask, Delete}	{Caption, Description, InstallDate, Name, Status, AccessMask, AllowMaximum, MaximumAllowed, Path, Type}

```
PS C:\Foo> # 2. Viewing Win32_Share class properties
PS C:\Foo> Get-CimClass -ClassName Win32_Share |
    Select-Object -ExpandProperty CimClassProperties |
    Sort-Object -Property Name |
    Format-Table -Property Name, CimType
```

Name	CimType
AccessMask	UInt32
AllowMaximum	Boolean
Caption	String
Description	String
InstallDate	DateTime
MaximumAllowed	UInt32
Name	String
Path	String
Status	String
Type	UInt32

```
PS C:\Foo> # 3. Getting methods of Win32_Share class
PS C:\Foo> Get-CimClass -ClassName Win32_Share |
    Select-Object -ExpandProperty CimClassMethods
```

Name	ReturnType	Parameters	Qualifiers
Create	UInt32	{Access, Description, MaximumAllowed, Name, Password, Path, Type}	{Constructor, Implemented, MappingStrings, Static}
SetShareInfo	UInt32	{Access, Description, MaximumAllowed}	{Implemented, MappingStrings}
GetAccessMask	UInt32	{}	{Implemented, MappingStrings}
Delete	UInt32	{}	{Destructor, Implemented, MappingStrings}

```
PS C:\Foo> # 4. Getting classes in a non-default namespace
PS C:\Foo> Get-CimClass -Namespace root\directory\LDAP |
Where-Object CimClassName -match '^ds_group'
```

Namespace: ROOT/directory/LDAP

CimClassName	CimClassMethods	CimClassProperties
ds_groupofuniquenames	{}	{ADSIPath, DS_adminDescription, DS_adminDisplayName...
ds_groupofnames	{}	{ADSIPath, DS_adminDescription, DS_adminDisplayName...
ds_group	{}	{ADSIPath, DS_adminDescription, DS_adminDisplayName...
ds_grouppolicycontainer	{}	{ADSIPath, DS_adminDescription, DS_adminDisplayName...

```
PS C:\Foo> # 5. Viewing the instances of the ds_group class
PS C:\Foo> Get-CimInstance -Namespace root\directory\LDAP -ClassName 'DS_Group' |
Select-Object -First 10 |
Format-Table -Property DS_name, DS_Member
```

DS_name	DS_Member
Administrators	{CN=Domain Admins,CN=Users,DC=Reskit,DC=Org, CN=Enterprise Admins,CN=Users,DC=Reskit,DC=Org, CN=Administrator,CN=Users,DC=Reskit,DC=Org}
Users	{CN=Domain Users,CN=Users,DC=Reskit,DC=Org, CN=S-1-5-11,CN=ForeignSecurityPrincipals,DC=Reskit,DC=Org, CN=S-1-5-4,CN=ForeignSecurityPrincipals,DC=Reskit,DC=Org}
Guests	{CN=Domain Guests,CN=Users,DC=Reskit,DC=Org, CN=Guest,CN=Users,DC=Reskit,DC=Org}
Print Operators	
Backup Operators	
Replicator	
Remote Desktop Users	
Network Configuration Operators	
Performance Monitor Users	
Performance Log Users	

```
PS C:\Foo> # 1. Using Get-CimInstance in the default namespace
PS C:\Foo> Get-CimInstance -ClassName Win32_Share
```

Name	Path	Description
ADMIN\$	C:\WINDOWS	Remote Admin
C\$	C:\	Default share
IPC\$		Remote IPC

```
PS C:\Foo> # 2. Getting WMI objects from a non-default namespace
PS C:\Foo> $Instance1 = @{
    Namespace = 'ROOT\directory\LDAP'
    ClassName = 'ds_group'
}
PS C:\Foo> Get-CimInstance @Instance1 |
Sort-Object -Property Name |
Select-Object -First 10 |
Format-Table -Property DS_name, DS_distinguishedName
```

DS_name	DS_distinguishedName
Administrators	CN=Administrators,CN=Builtin,DC=Reskit,DC=Org
Group Policy Creator Owners	CN=Group Policy Creator Owners,CN=Users,DC=Reskit,DC=Org
Pre-Windows 2000 Compatible Access	CN=Pre-Windows 2000 Compatible Access,CN=Builtin,DC=Reskit,DC=Org
Windows Authorization Access Group	CN=Windows Authorization Access Group,CN=Builtin,DC=Reskit,DC=Org
Allowed RODC Password Replication Group	CN=Allowed RODC Password Replication Group,CN=Users,DC=Reskit,DC=Org
Denied RODC Password Replication Group	CN=Denied RODC Password Replication Group,CN=Users,DC=Reskit,DC=Org
Enterprise Read-only Domain Controllers	CN=Enterprise Read-only Domain Controllers,CN=Users,DC=Reskit,DC=Org
Cloneable Domain Controllers	CN=Cloneable Domain Controllers,CN=Users,DC=Reskit,DC=Org
Protected Users	CN=Protected Users,CN=Users,DC=Reskit,DC=Org
Key Admins	CN=Key Admins,CN=Users,DC=Reskit,DC=Org



```

PS C:\Foo> # 3. Using a WMI filter
PS C:\Foo> $Filter = "ds_Name LIKE '%operator%' "
PS C:\Foo> Get-CimInstance @Instance1 -Filter $Filter |
Format-Table -Property DS_Name

```

DS\_Name

```

-----
Network Configuration Operators
Cryptographic Operators
Access Control Assistance Operators
Print Operators
Account Operators
Server Operators
Backup Operators

```

```

PS C:\Foo> # 4. Using a WMI query
PS C:\Foo> $Query = @"
SELECT * from ds_group
WHERE ds_Name like '%operator%'
"@
PS C:\Foo> Get-CimInstance -Query $Query -Namespace 'root\directory\LDAP' |
Format-Table DS_Name

```

DS\_Name

```

-----
Network Configuration Operators
Cryptographic Operators
Access Control Assistance Operators
Print Operators
Account Operators
Server Operators
Backup Operator

```

```

PS C:\Foo> # 5. Getting a WMI object from a remote system (DC1)
PS C:\Foo> Get-CimInstance -CimSession DC1 -ClassName Win32_ComputerSystem
Format-Table -AutoSize

```

Name	PrimaryOwnerName	Domain	TotalPhysicalMemory	Model	Manufacturer	PSComputerName
DC1	Book Readers	Reskit.Org	4331601920	Virtual Machine	Microsoft Corporation	DC1

```

PS C:\Foo> # 1. Reviewing methods of Win32_Share class on SRV1
PS C:\Foo> Get-CimClass -ClassName Win32_Share |
Select-Object -ExpandProperty CimClassMethods

```

Name	ReturnType	Parameters	Qualifiers
Create	UInt32	{Access, Description, MaximumAllowed, Name, Password, Path, Type}	{Constructor, Implemented, MappingStrings, Static}
SetShareInfo	UInt32	{Access, Description, MaximumAllowed}	{Implemented, MappingStrings}
GetAccessMask	UInt32	{}	{Implemented, MappingStrings}
Delete	UInt32	{}	{Destructor, Implemented, MappingStrings}



```

PS C:\Foo> # 2. Reviewing properties of Win32_Share class
PS C:\Foo> Get-CimClass -ClassName Win32_Share |
    Select-Object -ExpandProperty CimClassProperties |
    Format-Table -Property Name, CimType

```

Name	CimType
-----	-----
Caption	String
Description	String
InstallDate	DateTime
Name	String
Status	String
AccessMask	UInt32
AllowMaximum	Boolean
MaximumAllowed	UInt32
Path	String
Type	UInt32

```

PS C:\Foo> # 3. Creating a new SMB share using the Create() static method
PS C:\Foo> $NewShareDetails = @{
    Name       = 'TestShare1'
    Path       = 'C:\Foo'
    Description = 'Test Share'
    Type       = [uint32] 0 # disk
}
PS C:\Foo> $CimShareHT = @{
    ClassName = 'Win32_Share'
    MethodName = 'Create'
    Arguments = $NewShareDetails
}
PS C:\Foo> Invoke-CimMethod @CimShareHT

```

```

ReturnValue PSComputerName
-----
0

```

```

PS C:\Foo> # 4. Viewing the new SMB share
PS C:\Foo> Get-SMBShare -Name 'TestShare1'

```

Name	ScopeName	Path	Description
-----	-----	-----	-----
TestShare1	*	C:\Foo	Test Share

```
PS C:\Foo> # 5. Viewing the new SMB share using Get-CimInstance
PS C:\Foo> Get-CimInstance -Class Win32_Share -Filter "Name = 'TestShare1'"
```

Name	Path	Description
TestShare1	C:\Foo	Test Share

```
PS C:\Foo> # 6. Removing the share
PS C:\Foo> Get-CimInstance -Class Win32_Share -Filter "Name = 'TestShare1'" |
Invoke-CimMethod -MethodName Delete
```

```
ReturnValue PSComputerName
-----
0
```

```
PS C:\Foo> # 4. Displaying event details
PS C:\Foo> $NotepadEvent.SourceEventArgs.NewEvent.TargetInstance
```

ProcessId	Name	HandleCount	WorkingSetSize	VirtualSize
10184	notepad.exe	214	18427904	22034896

```
PS C:\Foo> # 6. Registering an event query based on the registry provider
PS C:\Foo> New-Item -Path 'HKLM:\SOFTWARE\Packt' | Out-Null
PS C:\Foo> $Query2 = "SELECT * FROM RegistryValueChangeEvent
WHERE Hive='HKEY_LOCAL_MACHINE'
AND KeyPath='SOFTWARE\Packt' AND ValueName='MOLTUAE'"
PS C:\Foo> $Action2 = {
Write-Host -Object "Registry Value Change Event Occurred"
$Global:RegEvent = $Event
}
PS C:\Foo> $RegisterHT = @{
Query = $Query2
Action = $Action2
Source = 'RegChange'
}
PS C:\Foo> Register-CimIndicationEvent @RegisterHT
```

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
1	RegChange		NotStarted	False		...

```

PS C:\Foo> # 7. Creating a new registry key and setting a value entry
PS C:\Foo> $Query2HT = [ordered] @{
    Type = 'DWord'
    Name = 'MOLTUAE'
    Path = 'HKLM:\Software\Packt'
    Value = 42
}
PS C:\Foo> Set-ItemProperty @Query2HT

Registry Value Change Event Occurred
PS C:\Foo> Get-ItemProperty -Path HKLM:\SOFTWARE\Packt

MOLTUAE      : 42
PSPath       : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Packt
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\SOFTWARE
PSChildName  : Packt
PSDrive      : HKLM
PSProvider    : Microsoft.PowerShell.Core\Registry

```

```

PS C:\Foo> # 9. Examining event details
PS C:\Foo> $RegEvent.SourceEventArgs.NewEvent

```

```

SECURITY_DESCRIPTOR :
TIME_CREATED        : 133136814589107181
Hive                 : HKEY_LOCAL_MACHINE
keyPath              : SOFTWARE\Packt
ValueName            : MOLTUAE
PSComputerName       :

```

```

PS C:\Foo> # 11. Creating a temporary WMI event registration
PS C:\Foo> $EventHT= @{
    Namespace = 'ROOT\directory\LDAP'
    SourceID = 'DSGroupChange'
    Query = $Query1
    Action = {
        $Global:ADEvent = $Event
        Write-Host 'We have a group change'
    }
}
PS C:\Foo> Register-CimIndicationEvent @EventHT

```

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
2	DSGroupChange		NotStarted	False		...

```

PS C:\Foo> # 12. Adding a user to the Enterprise Admins group
PS C:\Foo> Add-ADGroupMember -Identity 'Enterprise Admins' -Members Malcolm
PS C:\Foo> We have a group change

```

```
PS C:\Foo> # 13. Viewing the newly added user within the group
PS C:\Foo> $ADEvent.SourceEventArgs.NewEvent.SourceInstance |
Format-Table -Property DS_sAMAccountName,DS_Member
```

```
DS_sAMAccountName DS_Member
-----
Enterprise Admins {CN=Malcolm,OU=IT,DC=Reskit,DC=Org,
                  CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org,
                  CN=Administrator,CN=Users,DC=Reskit,DC=Org}
```

```
PS C:\Foo> # 8. Viewing the event registration details
PS C:\Foo> Get-WMIPE
```

\*\*\* Event Filters Defined \*\*\*

Name	Query
EventFilter1	SELECT * From __InstanceModificationEvent Within 10 WHERE TargetInstance ISA 'ds_group' AND TargetInstance.ds_name = 'Enterprise Admins'

\*\*\*Consumer Defined \*\*\*

Name	Commandlinetemplate
EventConsumer1	Pwsh.exe -File C:\Foo\Monitor.ps1

\*\*\*Bindings Defined \*\*\*

Filter	Consumer
__EventFilter (Name = "EventFilter1")	CommandLineEventConsumer (Name = "EventConsumer1")

```
PS C:\Foo> # 10. Viewing Grouplog.txt file
PS C:\Foo> Get-Content -Path C:\Foo\Grouplog.txt
```

On: [11/25/2022 15:44:54] Group [Enterprise Admins] was changed

Name	DistinguishedName
Malcolm	CN=Malcolm,OU=IT,DC=Reskit,DC=Org
Jerry Garcia	CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org
Administrator	CN=Administrator,CN=Users,DC=Reskit,DC=Org

Unauthorized user [Malcolm] added to Enterprise Admins

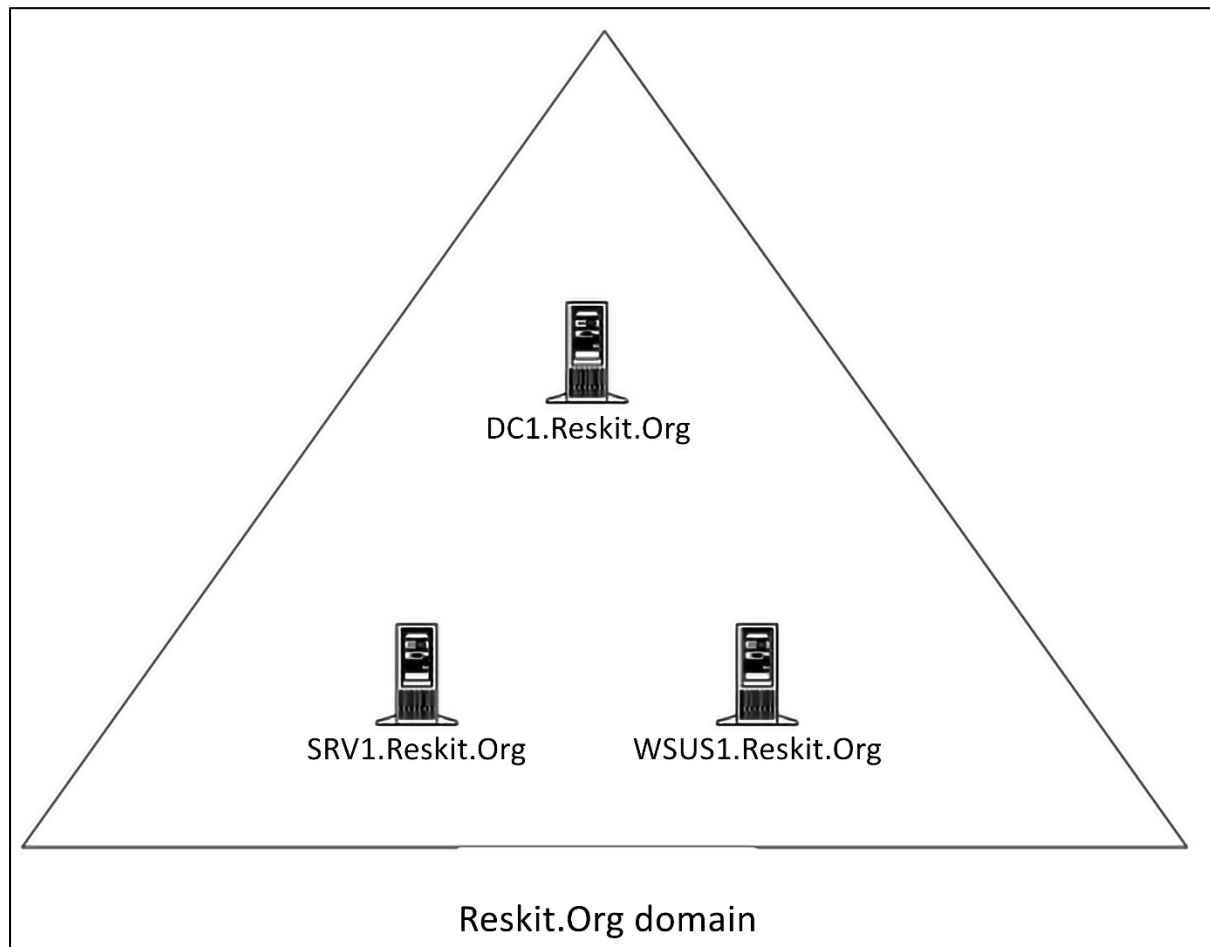
\*\*\*\*\*



```
PS C:\Foo> # 11. Tidying up
PS C:\Foo> Remove-WMIPE # invoke this function you defined above
PS C:\Foo> $RGMHT = @{
    Identity = 'Enterprise admins'
    Member   = 'Malcolm'
    Confirm  = $false
}
PS C:\Foo> Remove-ADGroupMember @RGMHT
PS C:\Foo> Get-WMIPE # ensure you have removed the event handling

*** Event Filters Defined ***
***Consumer Defined ***
***Bindings Defined ***
```

## Chapter 14: Managing Windows Update Services



```
PS C:\Foo> # 2. Installing WSUS on WSUS1
PS C:\Foo> $ScriptBlock1 = {
    $InstallHT = @{
        Name = 'UpdateServices'
        IncludeManagementTools = $true
    }
    Install-WindowsFeature @InstallHT |
    Format-Table
}
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock $ScriptBlock1
```

Success	Restart Needed	Exit Code	Feature	Result
---------	----------------	-----------	---------	--------

True	No	Success	{ASP.NET 4.8, HTTP Activation, Remote Server Administration Tools, Role Administration Tools...}	
------	----	---------	--	--

WARNING: Additional configuration may be required. Review the article [Managing WSUS Using PowerShell at TechNet Library \(http://go.microsoft.com/fwlink/?LinkId=235499\)](http://go.microsoft.com/fwlink/?LinkId=235499) for more information on the recommended steps to perform WSUS installation using PowerShell.

```
PS C:\Foo> # 3. Determining features installed on WSUS1
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WindowsFeature |
        Where-Object Installed |
        Format-Table
}
```

Display Name	Name	Install Sta
[X] File and Storage Services	FileAndStorage-Services	Installed
[X] Storage Services	Storage-Services	Installed
[X] Web Server (IIS)	Web-Server	Installed
[X] Web Server	Web-WebServer	Installed
[X] Common HTTP Features	Web-Common-Http	Installed
[X] Default Document	Web-Default-Doc	Installed
[X] Static Content	Web-Static-Content	Installed
[X] Performance	Web-Performance	Installed
[X] Dynamic Content Compression	Web-Dyn-Compression	Installed
[X] Security	Web-Security	Installed
[X] Request Filtering	Web-Filtering	Installed
[X] Windows Authentication	Web-Windows-Auth	Installed
[X] Application Development	Web-App-Dev	Installed
[X] .NET Extensibility 4.8	Web-Net-Ext45	Installed
[X] ASP.NET 4.8	Web-Asp-Net45	Installed
[X] ISAPI Extensions	Web-ISAPI-Ext	Installed
[X] ISAPI Filters	Web-ISAPI-Filter	Installed
[X] Management Tools	Web-Mgmt-Tools	Installed
[X] IIS Management Console	Web-Mgmt-Console	Installed
[X] IIS 6 Management Compatibility	Web-Mgmt-Compat	Installed
[X] IIS 6 Metabase Compatibility	Web-Metabase	Installed
[X] Windows Server Update Services	UpdateServices	Installed
[X] WID Connectivity	UpdateServices-WidDB	Installed
[X] WSUS Services	UpdateServices-Services	Installed
[X] .NET Framework 4.8 Features	NET-Framework-45-Featu...	Installed
[X] .NET Framework 4.8	NET-Framework-45-Core	Installed
[X] ASP.NET 4.8	NET-Framework-45-ASPNET	Installed
[X] WCF Services	NET-WCF-Services45	Installed
[X] HTTP Activation	NET-WCF-HTTP-Activatio...	Installed
[X] TCP Port Sharing	NET-WCF-TCP-PortSharin...	Installed
[X] Microsoft Defender Antivirus	Windows-Defender	Installed
[X] Remote Server Administration Tools	RSAT	Installed
[X] Role Administration Tools	RSAT-Role-Tools	Installed
[X] Windows Server Update Services Tools	UpdateServices-RSAT	Installed
[X] API and PowerShell cmdlets	UpdateServices-API	Installed
[X] User Interface Management Console	UpdateServices-UI	Installed
[X] System Data Archiver	System-DataArchiver	Installed
[X] Windows Internal Database	Windows-Internal-Datab...	Installed
[X] Windows PowerShell	PowerShellRoot	Installed
[X] Windows PowerShell 5.1	PowerShell	Installed
[X] Windows Process Activation Service	WAS	Installed
[X] Process Model	WAS-Process-Model	Installed
[X] Configuration APIs	WAS-Config-APIs	Installed
[X] WoW64 Support	Wow64-Support	Installed
[X] XPS Viewer	XPS-Viewer	Installed

```
PS C:\Foo> # 5. Performing post-installation configuration using WsusUtil.exe
PS C:\Foo> $ScriptBlock3 = {
    $WSUSDir = 'C:\WSUS'
    $Child = 'Update Services\Tools\wsusutil.exe'
    $CMD = Join-Path -Path "$env:ProgramFiles\" -ChildPath $Child
    & $CMD --% Postinstall CONTENT_DIR=$WSUSDir
}
```

```
PS C:\Foo> Invoke-Command -ComputerName WSUS1 -ScriptBlock $ScriptBlock3
Log file is located at C:\Users\Administrator\AppData\Local\Temp\WSUS Postinstall_20221202T165613.10g
Post install is starting
Post install has successfully completed
```

```
PS C:\Foo> # 6. Viewing the WSUS website on WSUS1
PS C:\Foo> Invoke-Command -ComputerName WSUS1 -ScriptBlock {
    Get-Website -Name ws* | Format-Table -AutoSize
}
```

Name	ID	State	Physical Path	Bindings
WSUS Administration	624931287	Started	C:\Program Files\Update Services\WebServices\Root\	http :8530: https :8531: sslFlags=0

```
PS C:\Foo> # 7. Viewing the cmdlets in the UpdateServices module
PS C:\Foo> Invoke-Command -ComputerName WSUS1 -ScriptBlock {
    Get-Command -Module UpdateServices |
    Format-Table -AutoSize
}
```

CommandType	Name	Version	Source
Cmdlet	Add-WSusComputer	2.0.0.0	UpdateServices
Cmdlet	Add-WSusDynamicCategory	2.0.0.0	UpdateServices
Cmdlet	Approve-WSusUpdate	2.0.0.0	UpdateServices
Cmdlet	Deny-WSusUpdate	2.0.0.0	UpdateServices
Cmdlet	Get-WSusClassification	2.0.0.0	UpdateServices
Cmdlet	Get-WSusComputer	2.0.0.0	UpdateServices
Cmdlet	Get-WSusDynamicCategory	2.0.0.0	UpdateServices
Cmdlet	Get-WSusProduct	2.0.0.0	UpdateServices
Cmdlet	Get-WSusServer	2.0.0.0	UpdateServices
Cmdlet	Get-WSusUpdate	2.0.0.0	UpdateServices
Cmdlet	Invoke-WSusServerCleanup	2.0.0.0	UpdateServices
Cmdlet	Remove-WSusDynamicCategory	2.0.0.0	UpdateServices
Cmdlet	Set-WSusClassification	2.0.0.0	UpdateServices
Cmdlet	Set-WSusDynamicCategory	2.0.0.0	UpdateServices
Cmdlet	Set-WSusProduct	2.0.0.0	UpdateServices
Cmdlet	Set-WSusServerSynchronization	2.0.0.0	UpdateServices



```
PS C:\Foo> # 8. Inspecting properties of the object created with Get-WsusServer
```

```
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
```

```
    $WSUSServer = Get-WsusServer
```

```
    $WSUSServer.GetType().FullName
```

```
    $WSUSServer | Select-Object -Property *
```

```
}
```

```
    TypeName: Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
```

```
WebServiceUrl           : http://WSUS1:8530/ApiRemoting30/WebService.asmx
```

```
BypassApiRemoting       : False
```

```
IsServerLocal           : True
```

```
Name                    : WSUS1
```

```
Version                  : 10.0.20348.143
```

```
IsConnectionSecureForApiRemoting : True
```

```
PortNumber               : 8530
```

```
PreferredCulture         : en
```

```
ServerName               : WSUS1
```

```
UseSecureConnection      : False
```

```
ServerProtocolVersion    : 1.20
```

```
PSComputerName           : WSUS1
```

```
RunspaceId              : 80977ba5-ba38-4d97-95d5-a5057610dbb1
```

```
PS C:\Foo> # 9. Viewing details of the WSUS Server object
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
($WSUSServer | Get-Member -MemberType Method).count
$WSUSServer | Get-Member -MemberType Method
}
```

TypeName: Microsoft.UpdateServices.Internal.BaseApi.UpdateServer

Name	MemberType	Definition
AddDynamicCategories	Method	void AddDynamicCategories(Sys...
AddDynamicCategory	Method	void AddDynamicCategory(Micro...
CancelAllDownloads	Method	void CancelAllDownloads(), vo...
CreateComputerTargetGroup	Method	Microsoft.UpdateServices.Admi...
CreateDynamicCategory	Method	Microsoft.UpdateServices.Admi...
CreateInstallApprovalRule	Method	Microsoft.UpdateServices.Admi...
CreateObjRef	Method	System.Runtime.Remoting.ObjRe...
DeleteDynamicCategory	Method	void DeleteDynamicCategory(st...
DeleteUpdate	Method	void DeleteUpdate(guid update...
Equals	Method	bool Equals(System.Object obj...
ExpirePackage	Method	void ExpirePackage(Microsoft....
ExportPackageMetadata	Method	void ExportPackageMetadata(Mi...
ExportUpdates	Method	void ExportUpdates(string pac...
GetChildServers	Method	Microsoft.UpdateServices.Admi...
GetCleanupManager	Method	Microsoft.UpdateServices.Admi...
GetComponentsWithErrors	Method	System.Collections.Specialize...
GetComputersNotContactedSinceCount	Method	int GetComputersNotContactedS...
GetComputersWithRecentNameChange	Method	System.Collections.Specialize...
GetComputerTarget	Method	Microsoft.UpdateServices.Admi...
GetComputerTargetByName	Method	Microsoft.UpdateServices.Admi...
GetComputerTargetCount	Method	int GetComputerTargetCount(),...
GetComputerTargetGroup	Method	Microsoft.UpdateServices.Admi...
GetComputerTargets	Method	Microsoft.UpdateServices.Admi...
GetContentDownloadProgress	Method	Microsoft.UpdateServices.Admi...
GetCurrentUserRole	Method	Microsoft.UpdateServices.Admi...
GetDatabaseConfiguration	Method	Microsoft.UpdateServices.Admi...
GetDownstreamServer	Method	Microsoft.UpdateServices.Admi...
GetDownstreamServers	Method	Microsoft.UpdateServices.Admi...
GetDynamicCategories	Method	System.Collections.Generic.IE...
GetDynamicCategory	Method	Microsoft.UpdateServices.Admi...
GetEmailNotificationConfiguration	Method	Microsoft.UpdateServices.Admi...
GetFailedToDownloadUpdatesCount	Method	int GetFailedToDownloadUpdate...
GetFrontEndServers	Method	System.Collections.ObjectModel...
GetHashCode	Method	int GetHashCode()

```

PS C:\Foo> # 10. Viewing WSUS server configuration
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSServer.GetConfiguration() |
    Select-Object -Property SyncFromMicrosoftUpdate,LogFilePath
}

```

```

SyncFromMicrosoftUpdate : True
LogFilepath              : C:\Program Files\Update Services\LogFiles\SoftwareDistribution.log
PSComputerName           : WSUS1
RunspaceId               : 262cd894-b585-4b58-b7c4-6bddebfb635

```

```

PS C:\Foo> # 11. Viewing product categories after initial install
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSProducts = Get-WSusProduct -UpdateServer $WSUSServer
    "{0} WSUS Products discovered" -f $WSUSProducts.Count
    $WSUSProducts |
    Select-Object -ExpandProperty Product |
    Format-Table -Property Title,
    Description
}

```

17 WSUS Products discovered

Title	Description
Exchange 2000 Server	For Exchange 2000 Products
Exchange Server 2003	For Exchange 2003 Products
Exchange	Exchange
Local Publisher	The local publisher (not Microsoft Update) of patches and applications.
Locally published packages	Patches and applications that are published locally, not synchronized from Microsoft Update.
Microsoft Corporation	Microsoft Corporation
Office 2003	Office 2003
Office XP	Office XP
Office	Office
SQL Server	SQL Server Category Description
SQL	SQL
Windows 2000 family	Windows 2000 family
Windows Server 2003 family	Windows Server 2003 family
Windows Server 2003, Datacenter Edition	Windows Server 2003, Datacenter Edition
Windows XP 64-Bit Edition Version 2003	Windows XP 64-Bit Edition Version 2003
Windows XP family	Windows XP family
Windows	Windows

```

PS C:\Foo> # 12. Displaying subscription information
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSSubscription = $WSUSServer.GetSubscription()
    $WSUSSubscription |
    Select-Object -Property * |
    Format-List
}

```

```

UpdateServer              : Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
SynchronizeAutomatically  : False
SynchronizeAutomaticallyTimeOfDay : 09:15:52
LastModifiedTime          : 03/12/2022 11:31:41
LastModifiedBy            : RESKIT\Administrator
LastSynchronizationTime   : 01/01/0001 00:00:00
Anchor                    : 0,2000-01-01 00:00:01.000
DeploymentAnchor           :
NumberOfSynchronizationsPerDay : 1
IsCategoryOnlySync        : False

```



```

PS C:\Foo> # 13. Getting latest categories of products available from Microsoft Update
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSSubscription.StartSynchronization()
    Do {
        Write-Output $WSUSSubscription.GetSynchronizationProgress()
        Start-Sleep -Seconds 30
    }
    While ($WSUSSubscription.GetSynchronizationStatus() -ne
        'NotProcessing')
}

```

```

PSComputerName : WSUS1
RunspaceId      : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems      : 0
ProcessedItems  : 0
Phase           : NotProcessing

```

```

PSComputerName : WSUS1
RunspaceId      : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems      : 3954
ProcessedItems  : 0
Phase           : Categories

```

... snipped for brevity

```

PSComputerName : WSUS1
RunspaceId      : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems      : 103642
ProcessedItems  : 5451
Phase           : Updates

```

```

PSComputerName : WSUS1
RunspaceId      : 262cd894-b585-4b58-b7c4-6bdedebfb635
TotalItems      : 103642
ProcessedItems  : 104202
Phase           : Updates

```

```

PS C:\Foo> # 14. Checking the results of the synchronization
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSSubscription.GetLastSynchronizationInfo()
}

```

```

PSComputerName : WSUS1
RunspaceId      : 77ec8cff-d1c6-4c90-a11d-4cef89201968
Id              : f2db6a4e-da3a-41b5-9330-2630cfb7d724
StartTime       : 04/12/2022 11:36:04
EndTime         : 04/12/2022 11:37:33
StartedManually : True
Result          : Succeeded
Error           : NotApplicable
ErrorText       :
UpdateErrors    : {}

```



```

PS C:\Foo> # 15.Reviewing the categories of the products available after synchronzation
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSProducts = Get-WsusProduct -UpdateServer $WSUSServer
    "{0} Product found on WSUS1" -f $WSUSProducts.Count
    $WSUSProducts |
        Select-Object -ExpandProperty Product -First 25 |
        Format-Table -Property Title,
                    Description
}

```

391 Product found on WSUS1

Title	Description
.NET 5.0	.NET 5.0
.NET 6.0	.NET 6.0
.NET Core 2.1	.NET Core 2.1
.NET Core 3.1	.NET Core 3.1
Active Directory Rights Management Services Client 2.0	Active Directory Rights Management Services Client 2.0 (AD...
Active Directory	Active Directory Product Family Category
Antigen for Exchange/SMTP	Defines the category for Antigen Updates. This will make s...
Antigen	Antigen Product Family Category
ASP.NET Web and Data Frameworks	ASP.NET Web and Data Frameworks product family
ASP.NET Web Frameworks	ASP.NET Web Framework
Azure Connected Machine Agent 3	Updates for the Azure Connected Machine Agent
Azure Connected Machine Agent	Product Family for Azure Connected Machine Agent
Azure File Sync agent updates for Windows Server 2012 R2	Azure File Sync agent updates for Windows Server 2012 R2
Azure File Sync agent updates for Windows Server 2016	Azure File Sync agent updates for Windows Server 2016
Azure File Sync agent updates for Windows Server 2019	Azure File Sync agent updates for Windows Server 2019
Azure File Sync agent updates for Windows Server 2022	Azure File Sync agent updates for Windows Server 2022
Azure File Sync	Azure File Sync
Azure IoT Edge for Linux on Windows Category	Updates for Azure IoT Edge for Linux on Windows Category. ...
Azure IoT Edge for Linux on Windows	Product Family for Azure IoT Edge for Linux on Windows
Azure Stack HCI	Azure Stack HCI and above
Bing Bar	Get quick access to Bing and MSN, as well as handy tools f...
Bing Growth	Product Family for Bing Growth
Bing Service v2.0	Bing Service 2.0 update
Bing	Live Search Product Family Category
BizTalk Server 2002	Category for BizTalk 2002. It requires SP1 as the minimum ...

```

PS C:\Foo> # 2. Locating versions of Windows Server supported by Windows Update
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WsusProduct |
        Where-Object -FilterScript {$_.product.title -match
                                '^Windows Server'} |
        Select-Object -ExpandProperty Product |
        Format-Table Title, UpdateSource
}

```

Title	UpdateSource
Windows Server 2003, Datacenter Edition	MicrosoftUpdate
Windows Server 2003	MicrosoftUpdate
Windows Server 2008 R2	MicrosoftUpdate
Windows Server 2008 Server Manager Dynamic Installer	MicrosoftUpdate
Windows Server 2008	MicrosoftUpdate
Windows Server 2012 Language Packs	MicrosoftUpdate
Windows Server 2012 R2 and later drivers	MicrosoftUpdate
Windows Server 2012 R2 Drivers	MicrosoftUpdate
Windows Server 2012 R2 Language Packs	MicrosoftUpdate
Windows Server 2012 R2	MicrosoftUpdate
Windows Server 2012	MicrosoftUpdate
Windows Server 2016 and Later Servicing Drivers	MicrosoftUpdate
Windows Server 2016 for RS4	MicrosoftUpdate
Windows Server 2016	MicrosoftUpdate
Windows Server 2016	MicrosoftUpdate
Windows Server 2019 and later, Servicing Drivers	MicrosoftUpdate
Windows Server 2019 and later, Upgrade & Servicing Drivers	MicrosoftUpdate
Windows Server 2019 Datacenter: Azure Edition Hotpatch	MicrosoftUpdate
Windows Server 2019	MicrosoftUpdate
Windows Server 2019	MicrosoftUpdate
Windows Server Drivers	MicrosoftUpdate
Windows Server Manager - Windows Server Update Services (WSUS) Dynamic Installer	MicrosoftUpdate
Windows Server Solutions Best Practices Analyzer 1.0	MicrosoftUpdate
Windows Server Technical Preview Language Packs	MicrosoftUpdate
Windows Server, version 1903 and later	MicrosoftUpdate
Windows Server, version 1903 and later	MicrosoftUpdate

```

PS C:\Foo> # 3. Discovering updates for for Windows 11
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WsusProduct -TitleIncludes 'Windows 11' |
    Select-Object -ExpandProperty Product |
    Format-Table -Property Title
}

```

Title

```

-----
Windows 11 Client S, version 22H2 and later, Servicing Drivers
Windows 11 Client S, version 22H2 and later, Upgrade & Servicing Drivers
Windows 11 Client, version 22H2 and later, Servicing Drivers
Windows 11 Client, version 22H2 and later, Upgrade & Servicing Drivers
Windows 11 Dynamic Update
Windows 11 GDR-DU
Windows 11

```

```

PS C:\Foo> # 4. Create and view a list of software product titles to include
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $Products =
    (Get-WsusProduct |
     Where-Object -FilterScript {$_.product.title -match
                                '^Windows Server'}).Product.Title
    $Products += @('Microsoft SQL Server 2016','Windows 11')
    $Products
}

```

```

Windows Server 2003, Datacenter Edition
Windows Server 2003
Windows Server 2008 R2
Windows Server 2008 Server Manager Dynamic Installer
Windows Server 2008
Windows Server 2012 Language Packs
Windows Server 2012 R2 and later drivers
Windows Server 2012 R2 Drivers
Windows Server 2012 R2 Language Packs
Windows Server 2012 R2
Windows Server 2012
Windows Server 2016 and Later Servicing Drivers
Windows Server 2016 for RS4
Windows Server 2016
Windows Server 2016
Windows Server 2019 and later, Servicing Drivers
Windows Server 2019 and later, Upgrade & Servicing Drivers
Windows Server 2019 Datacenter: Azure Edition Hotpatch
Windows Server 2019
Windows Server 2019
Windows Server Drivers
Windows Server Manager - Windows Server Update Services (WSUS) Dynamic Installer
Windows Server Solutions Best Practices Analyzer 1.0
Windows Server Technical Preview Language Packs
Windows Server, version 1903 and later
Windows Server, version 1903 and later
Microsoft SQL Server 2016
Windows 11

```

```

PS C:\Foo> # 6. Getting WSUS classification
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WsusClassification |
        Select-Object -ExpandProperty Classification |
        Format-Table -Property Title, Description -Wrap
    }

```

Title	Description
Applications	Products or line of business applications.
Critical Updates	A broadly released fix for a specific problem addressing a critical, non-security related bug.
Definition Updates	A broadly-released and frequent software update containing additions to a product's definition database. Definition databases are often used to detect objects with specific attributes, such as malicious code, phishing Web sites, or junk e-mail.
Driver Sets	A package of software modules that is designed to support the hardware of a specific model of computing device.
Drivers	A software component necessary to control or regulate another device.
Feature Packs	New product functionality that is first distributed outside the context of a product release, and usually included in the next full product release.
Security Updates	A broadly released fix for a product-specific security-related vulnerability. Security vulnerabilities are rated based on their severity which is indicated in the Microsoft® security bulletin as critical, important, moderate, or low.
Service Packs	A tested, cumulative set of all hotfixes, security updates, critical updates and updates, as well as additional fixes for problems found internally since the release of the product. Service packs may also contain a limited number of customer-requested design changes or features.
Tools	A utility or feature that aids in accomplishing a task or set of tasks.
Update Rollups	A tested, cumulative set of hotfixes, security updates, critical updates, and updates packaged together for easy deployment. A rollup generally targets a specific area, such as security, or a component of a product, such as Internet Information Services "IIS".
Updates	A broadly released fix for a specific problem addressing a noncritical, non-security-related bug.
Upgrades	A new product release bringing a device to the next version, containing bug fixes, design changes and new features.

```

PS C:\Foo> # 12. Synchronizing the updates which can take a long while to complete
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $IntervalSeconds = 15
    $NP = 'NotProcessing'
    Do {
        $WSUSSubscription.GetSynchronizationProgress()
        Start-Sleep -Seconds $IntervalSeconds
    } While ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)
}

```

```

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
TotalItems     : 31327
ProcessedItems : 420
Phase          : Updates

```

```

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
TotalItems     : 31327
ProcessedItems : 730
Phase          : Updates
PS C:\Foo>

```



```

PS C:\Foo> # 13. Synchronize the updates which can take a long while to complete.
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $IntervalSeconds = 15
    $NP = 'NotProessing'
    # Wait for synchronizing to start
    Do {
        Write-Output $WSUSSubscription.GetSynchronizationProgress()
        Start-Sleep -Seconds $IntervalSeconds
    }
    While ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)
    # Wait for all phases of process to end
    Do {
        Write-Output $WSUSSubscription.GetSynchronizationProgress()
        Start-Sleep -Seconds $IntervalSeconds
    } Until ($WSUSSubscription.GetSynchronizationStatus() -eq $NP)
}

```

```

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
TotalItems     : 31327
ProcessedItems : 3576
Phase          : Updates

```

```

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
TotalItems     : 31327
ProcessedItems : 24204
Phase          : Updates
PS C:\Foo>

```

```

PS C:\Foo> # 14. Checking the results of the synchronization
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSSubscription.GetLastSynchronizationInfo()
}

```

```

PSComputerName : WSUS1
RunspaceId     : 77ec8cff-d1f6-4c90-a11d-4cef89201968
Id             : 10aee80a-2746-49af-b78e-27fd12c6ec09
StartTime      : 04/12/2022 12:12:28
EndTime        : 04/12/2022 14:38:41
StartedManually : True
Result         : Succeeded
Error          : NotApplicable
ErrorText      :
UpdateErrors   : {}

```



```
PS C:\Foo> # 2. Creating a new policy and linking it to the domain
PS C:\Foo> $PolicyName = 'Reskit WSUS Policy'
PS C:\Foo> New-GPO -Name $PolicyName
```

```
DisplayName      : Reskit WSUS Policy
DomainName       : Reskit.Org
Owner            : RESKIT\Domain Admins
Id               : 9c6e0c02-75ef-43c7-8148-366e23da186e
GpoStatus        : AllSettingsEnabled
Description      :
CreationTime     : 08/12/2022 17:08:43
ModificationTime : 08/12/2022 17:08:43
UserVersion      :
ComputerVersion  :
WmiFilter        :
```

Reskit WSUS Policy  
Data collected on: OS:12:2022 17:27:15 [show all](#)

**General** [hide](#)

**Detail** [show](#)

**Link** [show](#)

**Security Filtering** [show](#)

**Delegation** [show](#)

**Computer Configuration (Enabled)** [hide](#)

**Policies** [hide](#)

**Administrative Templates** [hide](#)

Policy definitions (ADMX files) retrieved from the local computer.

**Windows Components/Windows Update** [hide](#)

Policy	Setting	Comment
<b>Configure Automatic Updates</b>	Enabled	2 - Notify for download and auto install
Configure automatic updating: The following settings are only required and applicable if 4 is selected. Install during automatic maintenance Scheduled install day: Scheduled install time: If you have selected "4 — Auto download and schedule the install" for your scheduled install day and specified a schedule, you also have the option to limit updating to a weekly bi-weekly or monthly occurrence, using the options below: Every week First week of the month Second week of the month Third week of the month Fourth week of the month Install updates for other Microsoft products		
<b>Specify intranet Microsoft update service location</b>	Enabled	
Set the intranet update service for detecting updates: http://WSUS1-8530 Set the intranet statistics server. http://WSUS1-8530 Set the alternate download server. (example: https://IntranetUpd01) Download files with no Url in the metadata if alternate download server is set. Do not enforce TLS certificate pinning for Windows Update client for detecting updates. Select the proxy behavior for Windows Update client for detecting updates:		

```
PS C:\Foo> # 2. Creating a WSUS computer target group for servers
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSServer = Get-WsusServer -Name WSUS1 -port 8530
    $WSUSServer.CreateComputerTargetGroup('Domain Servers')
}
```

```
PSComputerName    WSUS1
RunspaceId        c174bdae-e697-4fb4-8be2-3c9da9701920
UpdateServer      Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
Id                3a409534-50c7-4159-af22-caea674d5ff1
Name               Domain Servers ←
```

```
PS C:\Foo> # 3. Viewing all computer target groups on WSUS1
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSServer.GetComputerTargetGroups() |
    Format-Table -Property Name
}
```

Name

```
-----
All Computers
Domain Servers ←
Unassigned Computers
```

```
PS C:\Foo> # 4. Finding the Servers whose name includes SRV
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WsusComputer -NameIncludes SRV |
    Format-Table -Property FullDomainName, OSDescription
}
```

FullDomainName OSDescription

```
-----
srv1.reskit.org Windows Server 2022 Datacenter
srv2.reskit.org Windows Server 2022 Datacenter
psrv.reskit.org Windows Server 2022 Datacenter
```

```
PS C:\Foo> # 7. Finding the computers in the group:
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    Get-WsusComputer |
    Where-Object ComputerTargetGroupIDs -Contains $SRVGroup.id |
    Sort-Object -Property FullDomainName |
    Format-Table -Property FullDomainName, ClientVersion,
    LastSyncTime
}
```

FullDomainName ClientVersion LastSyncTime

```
-----
srv1.reskit.org 10.0.20348.1070 09/12/2022 10:03:01
srv1.reskit.org 10.0.20348.1070 09/12/2022 08:49:28
```

```

PS C:\Foo> # 7. Getting a list of approval rules
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock{
    $WSUSServer.GetInstallApprovalRules() |
    Format-Table -Property Name, Enabled, Action
}

```

Name	Enabled	Action
Default Automatic Approval Rule	False	Install
Critical Updates	True	Install

```

PS C:\Foo> # 2. Viewing the status of WSUS1
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSServer = Get-WSUSServer
    $WSUSServer.GetStatus()
}

```

```

PSComputerName           : WSUS1
RunspaceId               : aaeaa23c-8aff-4460-abc8-5580828dfb01
UpdateCount              : 26471
DeclinedUpdateCount      : 539
ApprovedUpdateCount      : 18
NotApprovedUpdateCount   : 25914
UpdatesWithStaleUpdateApprovalsCount : 0
ExpiredUpdateCount       : 0
CriticalOrSecurityUpdatesNotApprovedForInstallCount : 15474
WsusInfrastructureUpdatesNotApprovedForInstallCount : 0
UpdatesWithClientErrorsCount : 0
UpdatesWithServerErrorsCount : 0
UpdatesNeedingFilesCount : 0
UpdatesNeededByComputersCount : 46
UpdatesUpToDateCount     : 25886
CustomComputerTargetGroupCount : 1
ComputerTargetCount      : 13
ComputerTargetsNeedingUpdatesCount : 12
ComputerTargetsWithUpdateErrorsCount : 0
ComputersUpToDateCount   : 0
UnrecognizedClientRequestedTargetGroupNames : {}
ShouldDeleteUnneededRevisions : False

```

```

PS C:\Foo> # 3. Viewing computer targets
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $WSUSServer.GetComputerTargets() |
    Sort-Object -Property FullDomainName |
    Format-Table -Property FullDomainName, IPAddress, Last*
}

```

FullDomainName	IPAddress	LastSyncTime	LastSyncResult	LastReportedStatusTime	LastReportedInventoryTime
ch1.reskit.org	10.10.10.221	11/12/2022 13:58:36	Succeeded	11/12/2022 14:06:54	01/01/0001 00:00:00
dc1.reskit.org	10.10.10.10	11/12/2022 15:56:15	Succeeded	11/12/2022 16:04:22	01/01/0001 00:00:00
dc2.reskit.org	10.10.10.11	11/12/2022 14:28:41	Succeeded	11/12/2022 14:36:46	01/01/0001 00:00:00
fs1.reskit.org	10.10.10.101	11/12/2022 13:58:50	Succeeded	11/12/2022 14:06:59	01/01/0001 00:00:00
fs2.reskit.org	10.10.10.102	11/12/2022 13:58:26	Succeeded	11/12/2022 14:06:36	01/01/0001 00:00:00
hvl.reskit.org	10.10.10.201	11/12/2022 13:58:49	Succeeded	11/12/2022 14:06:59	01/01/0001 00:00:00
hvl2.reskit.org	10.10.10.202	11/12/2022 13:58:51	Succeeded	11/12/2022 14:07:00	01/01/0001 00:00:00
psrv.reskit.org	10.10.10.60	11/12/2022 14:59:07	Succeeded	11/12/2022 15:07:19	01/01/0001 00:00:00
smtp.reskit.org	10.10.10.49	11/12/2022 10:54:59	Succeeded	11/12/2022 10:57:01	01/01/0001 00:00:00
srv1.reskit.org	10.10.10.50	11/12/2022 16:23:24	Succeeded	11/12/2022 13:06:36	01/01/0001 00:00:00
srv2.reskit.org	10.10.10.52	11/12/2022 11:52:24	Succeeded	11/12/2022 12:00:32	01/01/0001 00:00:00
ssl.reskit.org	10.10.10.111	11/12/2022 13:58:51	Succeeded	11/12/2022 14:06:59	01/01/0001 00:00:00
wsus1.reskit.org	fe80::8672:8cc3:1e91:da0f%3	11/12/2022 06:00:52	Succeeded	11/12/2022 06:09:01	01/01/0001 00:00:00



Title	Description
2021-11 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5007386)	Install this update to resol...
2021-12 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5008286)	Install this update to resol...
2022-02 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5010456)	Install this update to resol...
2022-03 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5011580)	Install this update to resol...
2022-06 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5014677)	Install this update to resol...
2022-09 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5017392)	Install this update to resol...
2022-11 Cumulative security Hotpatch for Azure Stack HCI, version 21H2 and Windows Server 2022 Datacenter: Azure Edition for x64-based Systems (KB5019080)	Install this update to resol...

```
PS C:\Foo> # 8. Approving the update for installation in the target group
PS C:\Foo> Invoke-Command -Session $Session -ScriptBlock {
    $SelectedUpdate.Approve('Install',$SRVTargetGroup)
}
```

```
PSComputerName      : WSUS1
RunspaceId          : aaeaa23c-8aff-4460-abc8-5580828dfb01
UpdateServer        : Microsoft.UpdateServices.Internal.BaseApi.UpdateServer
Id                  : 678c6d2f-f575-4ade-95b1-9521942a8df6
CreationDate        : 11/12/2022 19:26:26
Action              : Install
GoLiveTime          : 11/12/2022 19:26:26
Deadline            : 31/12/9999 23:59:59
IsOptional          : False
State               : Pending
AdministratorName   : RESKIT\Administrator
UpdateId            : Microsoft.UpdateServices.Administration.UpdateRevisionId
ComputerTargetGroupId : 3a409534-50c7-4159-af22-caea674d5ff1
IsAssigned          : True
```