2

**Introducing PowerShell 7**

This chapter covers the following recipes:

* Exploring new Operators
* Exploring Parallel processing
* Explore Performance improvements
* Using Test-Connection
* Using Select-String
* Exploring Error view and Get-Error

# Introduction

In Chapter 1, you installed and configured PowerShell 7, along with VS code and a new font. The recipes enabled you to get PowerShell 7 installed and configured. In this chapter, we look at PowerShell 7 and how it differs from Windows PowerShell. The recipes in this chapter illustrate some of the important new features which come with PowerShell 7.

PowerShell 7 is cross-platform, and you can install it into Linux and macOS. With PowerShell now being cross-platform, there is a new audience to PowerShell, one with a background in Linux shells such as Bash. As a result, the PowerShell team added several new operators.

In Windows PowerShell, the Foreach syntax item and the Foreach-Object command allowed you to process collections of objects. With Windows PowerShell, each iteration through a collection was serial, which could result in a very long script run times PowerShell 7 introduces an improvement in the Foreach‑Object command that enables you to run iterations in parallel.

With the move to open source, the PowerShell code was open to inspection by the community. Many talented developers were able to make improvements to the performance, particularly in how the Foreach syntax element works.

Another significant improvement in PowerShell 7 is the update to Test-Connection, a command you use to test a network connection with a remote system. Test-Connection, in PowerShell 7 not only does more but is significantly faster than with Windows PowerShell.

Error reporting in Windows PowerShell was excellent. Clear and generally actionable error messages with details of exactly where the error occurred.

# Exploring New Operators

Operators are symbols or combinations of keystrokes which PowerShell recognizes and assigns some meaning. PowerShell uses the ‘+’ operator to mean addition, either arithmetic addition or string addition/concatenation. Most of the PowerShell operators were defined with Windows PowerShell V1.

PowerShell 7 now implements some new operators, including:

* Pipeline Chain
* Null-coalescing operator ??
* Null conditional assignment operator ??=
* Experimental Null conditional member access operators ?. and ?[]
* Background Processing Operator &
* Culture Invariant

## Getting Ready

This recipe uses SRV1, a Windows Server 2020 workgroup host. You have installed and configured PowerShell 7 and VS Code.

## How to do it...

1. Set Execution Policy for Windows PowerShell

Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force

1. Install the latest versions of Nuget and PowerShellGet

Install-PackageProvider Nuget -MinimumVersion 2.8.5.201 -Force |

  Out-Null

Install-Module -Name PowerShellGet -Force -AllowClobber

1. Ensure the C:\Foo Folder exists

$LFHT = @{

  ItemType    = 'Directory'

  ErrorAction = 'SilentlyContinue' # should it already exist

}

New-Item -Path C:\Foo @LFHT | Out-Null

1. Download PowerShell 7 installation script

Set-Location C:\Foo

$URI = ‘https://aka.ms/install-powershell.ps1’

Invoke-RestMethod -Uri $URI |

  Out-File -FilePath C:\Foo\Install-PowerShell.ps1

1. View Installation Script Help

Get-Help -Name C:\Foo\Install-PowerShell.ps1

1. Install PowerShell 7

$EXTHT = @{

  UseMSI                 = $true

  Quiet                  = $true

  AddExplorerContextMenu = $true

  EnablePSRemoting       = $true

}

C:\Foo\Install-PowerShell.ps1 @EXTHT | Out-Null

1. For the Adventurous - install the preview and daily builds as well

C:\Foo\Install-PowerShell.ps1 -Preview -Destination C:\PWSHPreview |

Out-Null

C:\Foo\Install-PowerShell.ps1 -Daily   -Destination C:\PWSHDailBuild |

  Out-Null

1. Create Windows PowerShell default Profiles

$URI = 'https://raw.githubusercontent.com/doctordns/Wiley20/master/' +

       'Goodies/Microsoft.PowerShell\_Profile.ps1'

$ProfileFile = $Profile.CurrentUserCurrentHost

New-Item $ProfileFile -Force -WarningAction SilentlyContinue |

   Out-Null

(Invoke-WebRequest -Uri $uri -UseBasicParsing).Content |

  Out-File -FilePath  $ProfileFile

$ProfilePath = Split-Path -Path $ProfileFile

$ConsoleProfile = Join-Path -Path $ProfilePath -ChildPath 'Microsoft.PowerShell\_profile.ps1'

(Invoke-WebRequest -Uri $URI -UseBasicParsing).Content |

  Out-File -FilePath  $ConsoleProfile

1. Check versions of PowerShell 7 loaded

Get-ChildItem -Path C:\pwsh.exe -ErrorAction SilentlyContinue

## How it works...

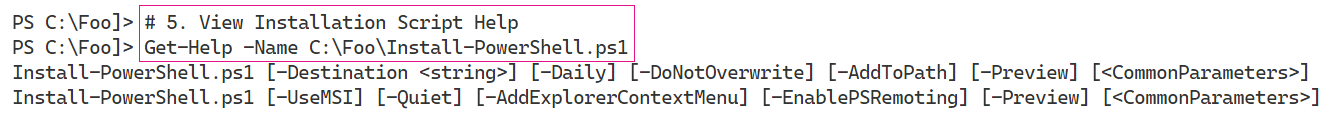
In step 1, you set the PowerShell execution policy to unrestricted which simplifies using scripts to configure hosts. In production, you may wish to set PowerShell’s execution policy to be more restrictive.

The PowerShell Gallery is a repository of PowerShell modules and scripts and is an essential resource for the IT Pro. This book makes use of several modules from the PS Gallery. In step 2, you update both the Nuget package provider (to version 2.8.5.201 or later) and an updated version of the PowerShellGet module.

Throughout this book, you use the C:\Foo folder to hold various files which you use in conjunction with the recipes. In step 3, you ensure the folder exists.

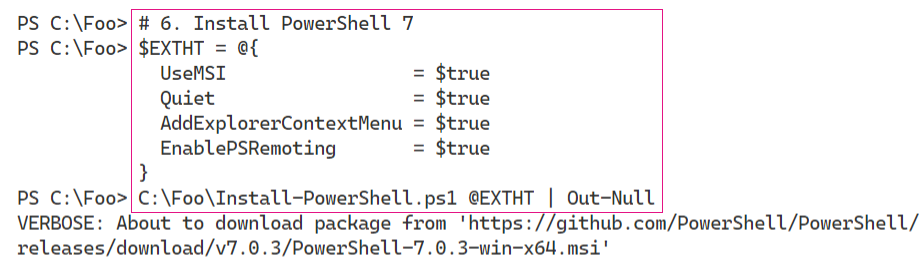
PowerShell 7 is not installed, at present, in Windows (or on in macOS or Linux), although this could change. To enable you to install PowerShell 7 in Windows, you retrieve an installation script from GitHub and store that in the C:\Foo folder. In step 4, you use a shortcut URL which points to GitHub and then use Invoke-RestMethod to download the file.

In step 5, you view the help information contained in the help file, which produces the following output:



Insert image B42024\_01\_01.png

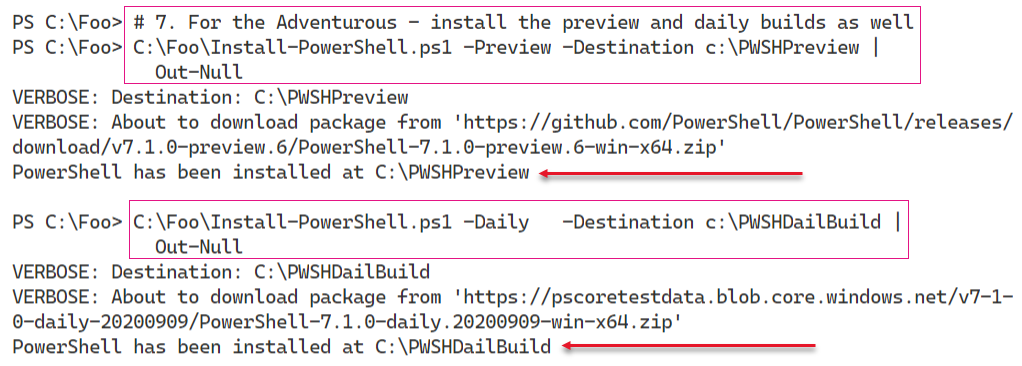
In step 6, you use the installation script and install PowerShell 7. The commands use an MSI which you then install silently. The MSI updates the system execution path to add the PowerShell 7 installation folder. The code retrieves the latest supported version of PowerShell 7, and you can view the actual file name in the following output:



Insert image B42024\_01\_02.png

PowerShell 7 is a work in progress. Every day, the PowerShell team builds updated versions of PowerShell and releases previews of the next major release. The preview builds are mostly stable and allow you to try out new features that are coming in the next major release. The daily build allows you to view progress on a specific bug or feature. You may find it useful to install both of these (and ensure you keep them up to date as time goes by).

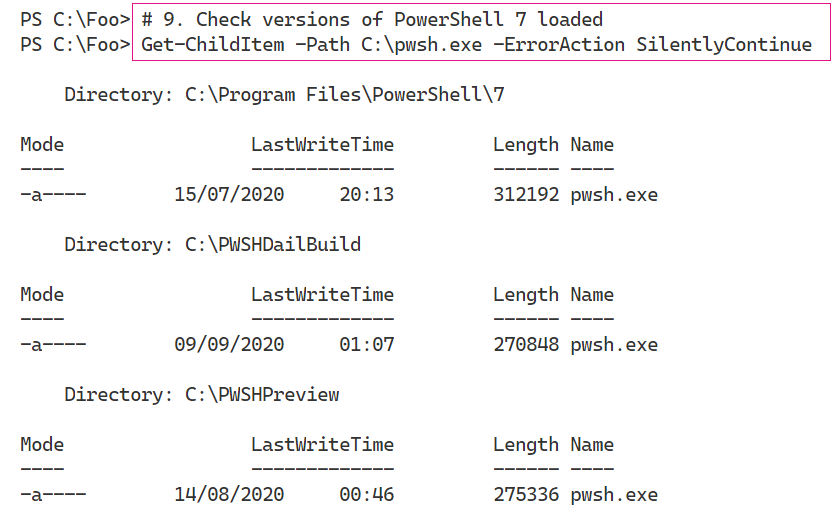
In step 7, you install the daily build and the latest Preview build, which looks like this:



Insert image B42024\_01\_03.png

PowerShell uses Profile files to enable you to configure PowerShell each time you run PowerShell (whether in the PowerShell console or as part of VS Code or the ISE). In step 8, you download a sample PowerShell profile script and save it locally. Note that the profile file you create in step 8 is for Windows PowerShell only.

The Executable name for PowerShell 8 is pwsh.exe. In step 9, you view the versions of this file as follows:



Insert image B42024\_01\_04.png

As you can see, there are 3 versions of PowerShell 7 installed on SRV1: the latest full release, the latest preview, and the build of the day.

## There's more...

In step 4, you use a shortened URL to download the Install-PowerShell.ps1 script. When you use Invoke-RestMethod, PowerShell discovers the underlying target URL for the script. The short URL allows Microsoft and the PowerShell team to publish a well-known URL and then have the flexibility to move the target location should that be necessary. The target URL, at the time of writing, is https://raw.githubusercontent.com/PowerShell/PowerShell/master/tools/install-powershell.ps1.

In step 7, you install both the latest daily build and the latest preview versions. The specific file versions you see are going to be different from the output shown here as at least for the preview versions!

# Exploring Parallel Processing

With PowerShell 7, the name of the PowerShell executable is now pwsh.exe, as you saw in the previous recipe. After installing PowerShell 7 in Windows, you can start the PowerShell 7 console by clicking start and typing pwsh.exe, then hitting return. The PowerShell MSI installer does not create a start panel or taskbar shortcut.

## Getting Ready

You run this recipe on SRV1 after you have installed PowerShell 7.

## How to do it...

1. Run the PowerShell 7 console.

From the Windows desktop in SRV1, click on the Windows key, then type pwsh, followed by the enter key.

1. View the PowerShell Version

$PSVersionTable

1. View the $Host variable

$Host

1. Look at the PowerShell process

Get-Process -Id $Pid |

  Format-Custom MainModule -Depth 1

1. Look at resource usage statistics

Get-Process -Id $Pid |

  Format-List CPU,\*Memory\*

1. Update PowerShell Help

$Before = Get-Help -Name about\_\*

Update-Help -Force | Out-Null

$After = Get-Help -Name about\_\*

$Delta = $After.Count - $Before.Count

"{0} Conceptual Help Files Added" -f $Delta

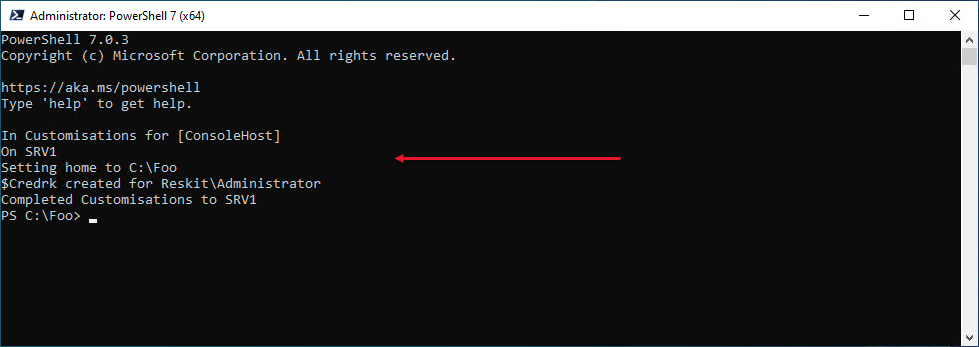
1. How many commands are available?

Get-Command |

  Group-Object -Property CommandType

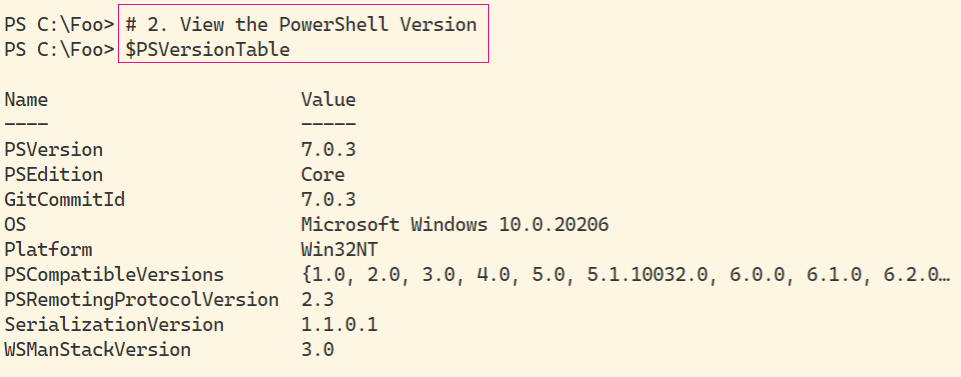
## How it works...

In step 1, you start the PowerShell 7 console on SRV1. The console should look like this:



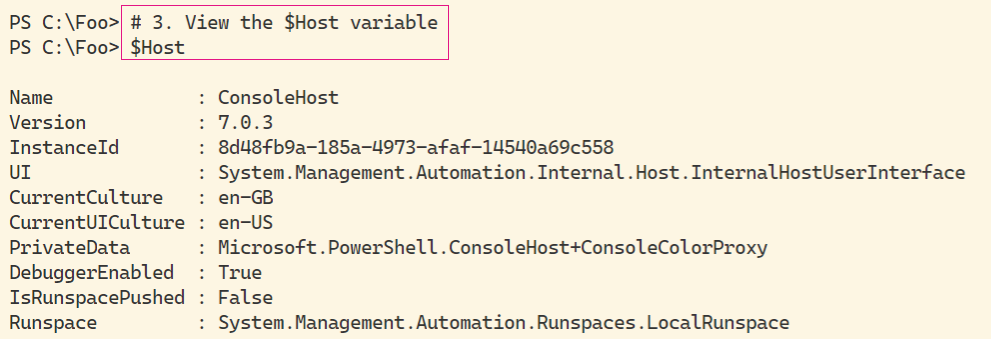
Insert image B42024\_01\_05.png

1. In step 2, you view the specific version of PowerShell by viewing the built-in variable $PSVersionTable, which looks like this:



Insert image B42024\_01\_06.png

In step 3, you examine the $Host variable to determine details about the PowerShell 7 host (the PowerShell console), which looks like this:



Insert image B42024\_01\_07.png

As you can see, in this case, you can see, the current culture is EN-GB. You may see a different value depending on which specific version of Windows Server you are using.

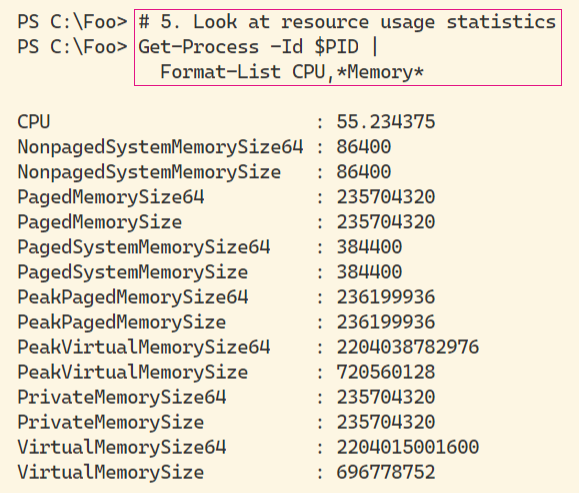
In step 4, you use Get-Process to look at the details of the PowerShell process, which looks like this:



Insert image B42024\_01\_08.png

In this figure, you can see the path to the PowerShell 7 executable. This value changes if you are running the release version or the daily/preview releases.

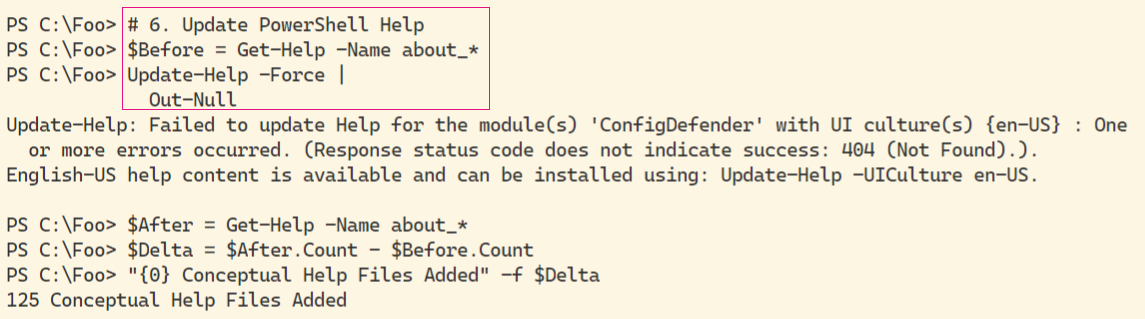
You can see, in step 5, details of resource usage of the pwsh.exe process running on the SRV1 host.



Insert image B42024\_01\_09.png

The values of each of the performance counters are likely to. You may see different values.

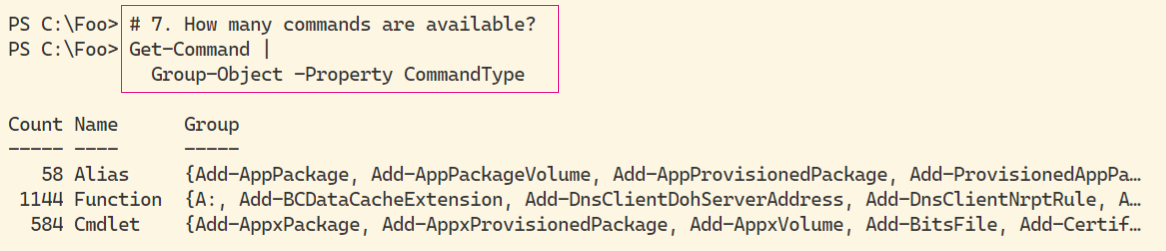
By default, Powershell 7, like Windows PowerShell, ships with minimum help files. You can, as you can see in step 6, use the Update-Help command to download updated PowerShell help content, like this:



Insert image B42024\_01\_10.png

As you can see from the output, not all help files were updated. In this case, the ConfigDefender module does not, at present, have updated help information. Also note that although the UK English versions of help details may be missing, there are US English versions that you can install which may be useful.

Commands, in PowerShell, include functions, cmdlets, and aliases. In step 7, you examine how many of each time of command is available by default, like this:



Insert image B42024\_01\_11.png

## There's more...

In step 1, you open the PowerShell console for the version of PowerShell you installed in “Installing PowerShell 7”. At the release of PowerShell 7.1, the version number you would see is 7.1.0. By the time you read this book, that version number may have advanced. To ensure you have the latest released version of PowerShell 7, re-run the Install-PowerShell.ps1 script you downloaded in “Installing PowerShell 7” In this step, you can also see the output generated by the Write-Host statements in the profile file you set up in “Introducing PowerShell 7”.

In step 4, you use the variable $PID, which contains the Windows process identifier of the PowerShell 7 console process. The actual value of $PID changes each time you run PowerShell, but the value always contains the process id of the current console process.

In step 7, you saw that you had 1786 commands available. This number changes as you add more features (and their accompanying modules) or download and install modules from repositories such as the PowerShell Gallery.

# Exploring Performance Improvements

In PowerShell 7, certain objects added by the PowerShell 7 installer (and PowerShell 7) differ from those used by Windows PowerShell.

## Getting Ready

This recipe uses SRV1 after you have installed PowerShell 7. In this recipe, you use the PowerShell 7 console to run the steps.

## How to do it...

1. Check the version table for PowerShell 7 console

$PSVersionTable

1. Examine the PowerShell 7 installation folder

Get-Childitem -Path $env:ProgramFiles\PowerShell\7 -Recurse |

  Measure-Object -Property Length -Sum

1. Look at PowerShell Configuration JSON file

Get-ChildItem -Path $env:ProgramFiles\PowerShell\7\powershell\*.json |

  Get-Content

1. Check Execution Policy for PowerShell 7

Get-ExecutionPolicy

1. View Module folders

$I = 0

$ModPath = $env:PSModulePath -split ';'

$ModPath |

  Foreach-Object {

    "[{0:N0}]   {1}" -f $I++, $\_}

1. View the Modules

$TotalCommands = 0

$TotalModules  = 0

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)"

  }

  Foreach ($Module in $Modules) {

    # "Module [$($module.name)] - Commands: [$($Cmds.Count)]"

    $TotalCommands += $Cmds.Count

    $TotalModules ++

  }

  ""

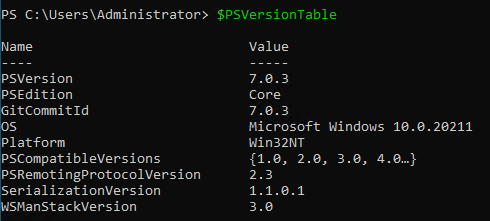
}

1. View totals of command and modules

"{0} commands in {1} modules" -f $TotalCommands, $TotalModules

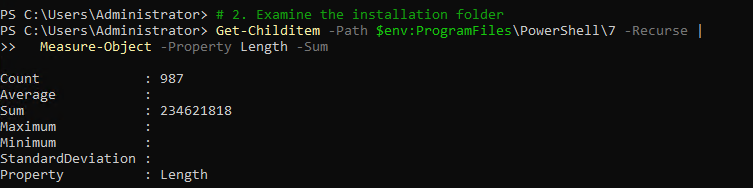
## How it works...

In step 1, you examine the $PSVersionTable variable to view the version information for PowerShell 7, which looks like this



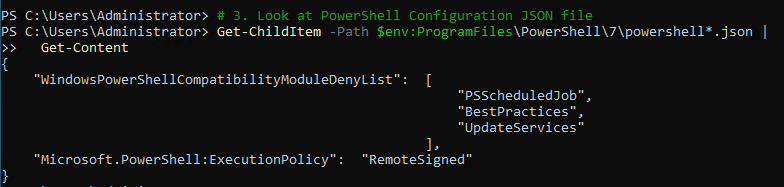
Insert image B42024\_01\_12.png

The PowerShell 7 installation program installs PowerShell 7 into a different folder (by default) from that used by Windows PowerShell. In step 2, you see a summary of the files installed into the PowerShell 7 installation folder as follows:



Insert image B42024\_01\_13.png

PowerShell 7 stores configuration values in a JSON file in the PowerShell 7 installation folder. In step 3, you view the contents of this file:



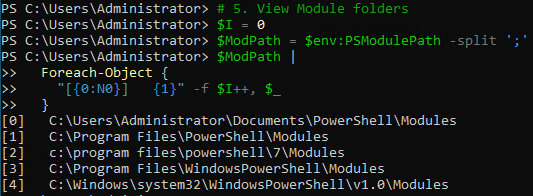
Insert image B42024\_01\_14.png

In step 4, you view the execution policy for PowerShell 7, as follows:



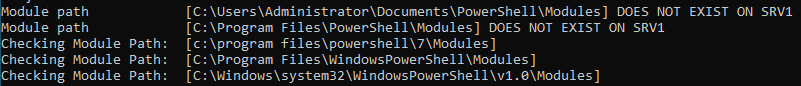
Insert image B42024\_01\_15.png

As with Windows PowerShell, PowerShell 7 loads commands from modules. PowerShell uses the $PSModulePath to determine which file store folders PowerShell 7 should use to find these modules. Viewing the contents of this variable, and discovering the folders, in step 5, looks like this:



Insert image B42024\_01\_16.png

With those module folders defined (by default), you can check how many commands exist in each folder, in step 6, the output of which looks like this:



Insert image B42024\_01\_17.png

In step 7, you can view the results to see how many commands exist in each of the modules in each module path. The output looks like this:



Insert image B42024\_01\_18.png

## There's more...

In step 1, you viewed the PowerShell version table. Depending on when you read this book, the version numbers you see may be later than shown here.

In step 4, you viewed PowerShell 7’s execution policy. Each time PowerShell 7 starts up, it reads the JSON file to obtain the value of Execution policy. You can use Set-Execution policy to re-set the policy immediately, or change the value in the JSON file and restart the PowerShell 7 console.

In step 5, you viewed the default folders which PowerShell 7 uses to search for a module (by default). The first folder is your personal modules, followed by PowerShell 7 and then Windows PowerShell modules. We cover the Windows PowerShell modules and Windows PowerShell compatibility in more detail in Chapter 2.

# Using Test-Connection

In Windows PowerShell and PowerShell 7, profile files contain code which PowerShell runs each time you start a new PowerShell console or a new instance of the ISE (or VS Code). These files enable you to pre-configure a PowerShell 7. You can add variables, PowerShell PS drives, functions and more using Profiles. As part of this book, there are PowerShell (and VS Code) profile file samples which you can download from GitHub.

This recipe download and installs the profile files for the PowerShell 7 console.

## Getting Ready

You run this recipe on SRV1 after you have installed PowerShell 7.

## How to do it...

1. Discover the profile file names

$ProfileFiles = $Profile |  Get-Member -MemberType NoteProperty

$ProfileFiles | Format-Table -Property Name, Definition

1. Check for Existence of each PowerShell Profile Files

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"

  }

  ""

}

1. Create Current User Current Host Profile

$CUCHProfile = $profile.CurrentUserCurrentHost

'Current user, current host profile path: '

"   [$CUCHPROFILE]"

1. Create Current user/Current host profile

$URI = 'https://raw.githubusercontent.com/doctordns/PACKT-PS7/master/' +

       'scripts/goodies/Microsoft.PowerShell\_Profile.ps1'

New-Item $CUCHProfile -Force -WarningAction SilentlyContinue |

   Out-Null

(Invoke-WebRequest -Uri $URI -UseBasicParsing).Content |

  Out-File -FilePath  $CUCHProfile

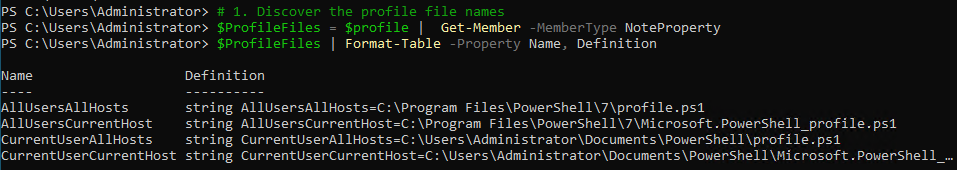
1. Existing from PowerShell 7 console

Exit-PSHostProcess

1. Restart the PowerShell 7 console and view the profile output at startup
2. Get-ChildItem -Path $Profile

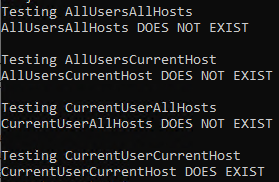
## How it works...

In step 1, you discover the names of each of the 4 profile files (for the PowerShell 7 console) then view their name and location (that is, the definition) which looks like this:



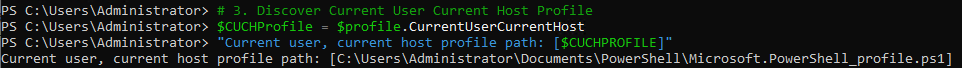
Insert image B42024\_01\_19.png

In step 2, you check to see which, if any, of the profile files exist, which looks like this:



Insert image B42024\_01\_20.png

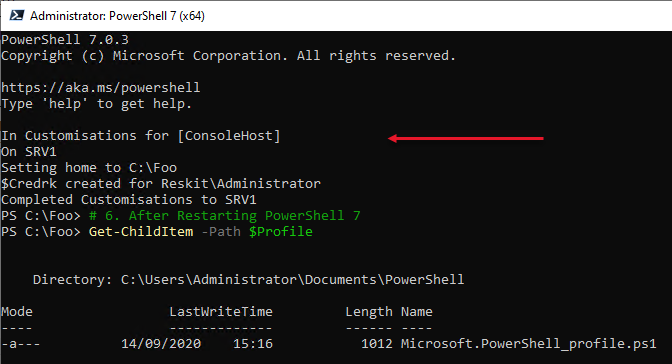
In step 3, you obtain and display the filename of the current user/current host profile file, which looks like this.



Insert image B42024\_01\_21.png

In step 4, you create an initial current user/current host profile file. This file is part of the GitHub repository which supports this book. In step 5, you exist the current PowerShell 7 console host. These two steps create no output.

In step 6, you start a new PowerShell profile. This time, as you can see here, the profile file exists, runs, and customizes the console:



Insert image B42024\_01\_22.png

## There's more...

In step 4, you download a sample profile file. This file contains some customizations for the PowerShell console configuration, including changing the default starting folder (to C:\Foo), creating some aliases, PowerShell drives and a credential object. These represent sample content you might consider for including in your console profile file. Note that VIS Code, which you install in the next recipe, uses a separate current user/current host profile file which means you can customize PowerShell at the console and in VS code differently.

# Using Select-String

The Windows PowerShell ISE was a great tool that Microsoft first introduced with Windows PowerShell v2 (and vastly improved with v3). This tool has reached feature completeness, and Microsoft has no plans for further development.

In its place, however, is Visual Studio Code, or VS Code. This open-source tool provides an extensive range of features for IT pros and others. For IT Pros, this should be your editor of choice. Whilst there is a learning curve (as for any new product), VS Code contains all the features you found in the ISE and more.

VS Code, and the available extensions, are works in progress. Each new release brings additional features which can be highly valuable. A recent addition, for example, is the ability to create diagrams directly in VS Code, For more details on VS Code, see https://code.visualstudio.com/. For details of VS Code extensions and the VS Code Extension Market Place, see https://code.visualstudio.com/docs/editor/extension-gallery#:~:text=You%20can%20browse%20and%20install,on%20the%20VS%20Code%20Marketplace.

## Getting Ready

You run this recipe on SRV1 after you have installed PowerShell 7 and have created a console profile file.

## How to do it...

1. Download the VS Code installation script from PS Gallery

$VSCPATH = 'C:\Foo'

Save-Script -Name Install-VSCode -Path $VSCPATH

Set-Location -Path $VSCPATH

1. Run the installation script and add in some popular extensions

$Extensions =  'Streetsidesoftware.code-spell-checker',

               'yzhang.markdown-all-in-one',

               'hediet.vscode-drawio'

$InstallHT = @{

  BuildEdition         = 'Stable-System'

  AdditionalExtensions = $Extensions

  LaunchWhenDone       = $true

}

.\Install-VSCode.ps1 @InstallHT

1. Exit VS Code.
2. Restart VS Code by clicking on the Windows Key and typing code to bring up the VS Code tile in the Windows Start Panel. Then right-click the VS Code tile and select Run as Administrator to start VS Code as an administrator
3. Inside, VS Code, open a VS Code Terminal by typing Ctrl+Shift+’. From the terminal, type pwsh and return.
4. Using VS Code, create a Sample Profile File for VS Code

$SAMPLE =

  'https://raw.githubusercontent.com/doctordns/PACKT-PS7/master/' +

  'scripts/goodies/Microsoft.VSCode\_profile.ps1'

(Invoke-WebRequest -Uri $Sample).Content |

  Out-File $Profile

1. Update Local User settings for VS Code
2. $JSON = @'
3. {
4. "workbench.colorTheme": "PowerShell ISE",
5. "powershell.codeFormatting.useCorrectCasing": true,
6. "files.autoSave": "onWindowChange",
7. "files.defaultLanguage": "powershell",
8. "editor.fontFamily": "'Cascadia Code',Consolas,'Courier New'",
9. "workbench.editor.highlightModifiedTabs": true,
10. "window.zoomLevel": 1
11. }
12. '@
13. $JHT = ConvertFrom-Json -InputObject $JSON -AsHashtable
14. $PWSH = "C:\\Program Files\\PowerShell\\7\\pwsh.exe"
15. $JHT += @{
16. "terminal.integrated.shell.windows" = "$PWSH"
17. }
18. $Path = $Env:APPDATA
19. $CP   = '\Code\User\Settings.json'
20. $Settings = Join-Path  $Path -ChildPath $CP
21. $JHT |
22. ConvertTo-Json  |
23. Out-File -FilePath $Settings
24. Create a short cut to VSCode

$SourceFileLocation  = "$env:ProgramFiles\Microsoft VS Code\Code.exe"

$ShortcutLocation    = "C:\foo\vscode.lnk"

# Create a  new wscript.shell object

$WScriptShell        = New-Object -ComObject WScript.Shell

$Shortcut            = $WScriptShell.CreateShortcut($ShortcutLocation)

$Shortcut.TargetPath = $SourceFileLocation

#Save the Shortcut to the TargetPath

$Shortcut.Save()

1. Create a short cut to PowerShell 7

$SourceFileLocation  = "$env:ProgramFiles\PowerShell\7\pwsh.exe"

$ShortcutLocation    = 'C:\Foo\pwsh.lnk'

# Create a  new wscript.shell object

$WScriptShell        = New-Object -ComObject WScript.Shell

$Shortcut            = $WScriptShell.CreateShortcut($ShortcutLocation)

$Shortcut.TargetPath = $SourceFileLocation

#Save the Shortcut to the TargetPath

$Shortcut.Save()

1. Build Updated Layout XML

$XML = @'

<?xml version="1.0" encoding="utf-8"?>

<LayoutModificationTemplate

  xmlns="http://schemas.microsoft.com/Start/2014/LayoutModification"

  xmlns:defaultlayout=

    "http://schemas.microsoft.com/Start/2014/FullDefaultLayout"

  xmlns:start="http://schemas.microsoft.com/Start/2014/StartLayout"

  xmlns:taskbar="http://schemas.microsoft.com/Start/2014/TaskbarLayout"

  Version="1">

<CustomTaskbarLayoutCollection>

<defaultlayout:TaskbarLayout>

<taskbar:TaskbarPinList>

 <taskbar:DesktopApp DesktopApplicationLinkPath="C:\Foo\vscode.lnk" />

 <taskbar:DesktopApp DesktopApplicationLinkPath="C:\Foo\pwsh.lnk" />

</taskbar:TaskbarPinList>

</defaultlayout:TaskbarLayout>

</CustomTaskbarLayoutCollection>

</LayoutModificationTemplate>

'@

$XML | Out-File -FilePath C:\Foo\Layout.Xml

1. Import the  start layout XML file

Import-StartLayout -LayoutPath C:\Foo\Layout.Xml -MountPath C:\

1. Create VSCode Profile for PowerShell 7

$CUCHProfile   = $profile.CurrentUserCurrentHost

$ProfileFolder = Split-Path -Path $CUCHProfile

$ProfileFile   = 'Microsoft.VSCode\_profile.ps1'

$VSProfile     = Join-Path -Path $ProfileFolder -ChildPath $ProfileFile

$URI = 'https://raw.githubusercontent.com/doctordns/PACKT-PS7/master/' +

       "scripts/goodies/$ProfileFile"

New-Item $VSProfile -Force -WarningAction SilentlyContinue |

   Out-Null

(Invoke-WebRequest -Uri $URI -UseBasicParsing).Content |

  Out-File -FilePath  $VSProfile

1. Logoff

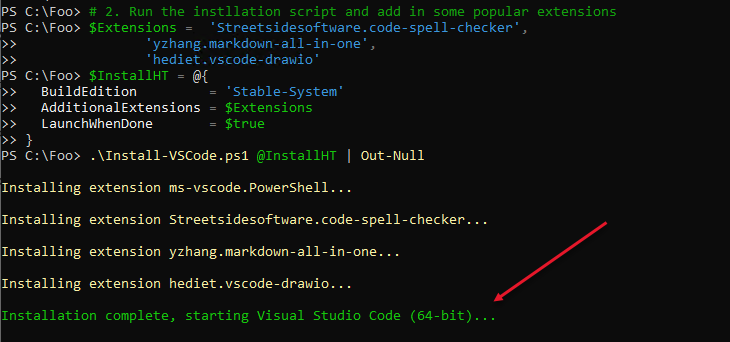
logoff.exe

1. Log back into Windows  and observe the taskbar
2. Run PowerShell console from the short cut
3. Run VS Code from the new taskbar shortcut and observe the profile file running.

## How it works...

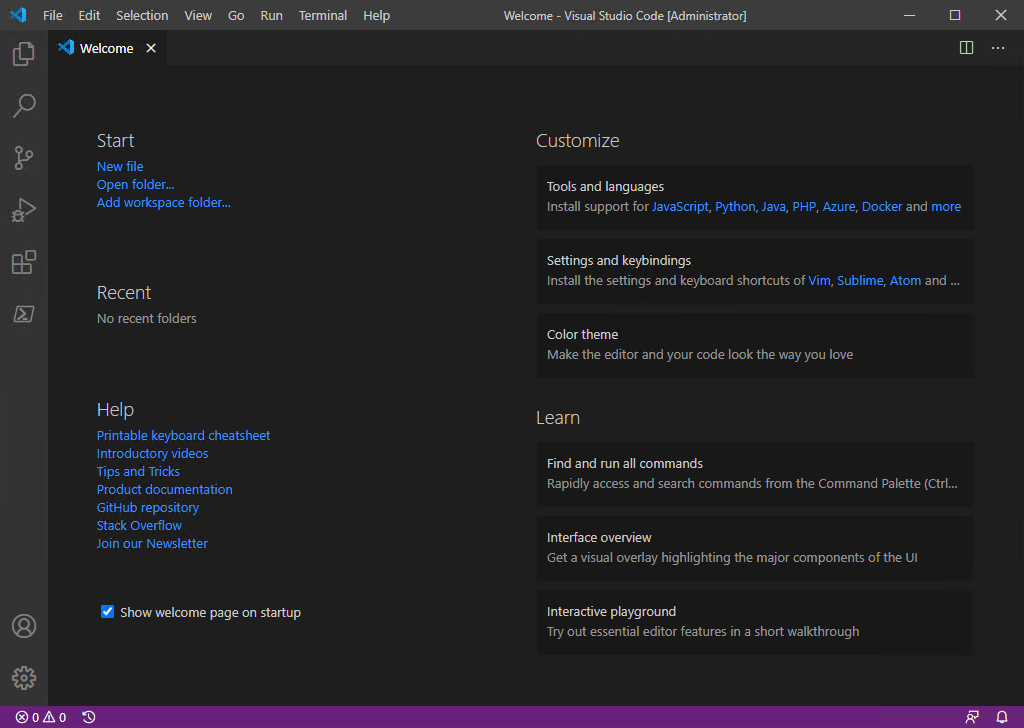
In step 1, you download the VS Code installation script from the PowerShell Gallery. This step produces no output.

Then, in step 2, you run the installation script and add in three specific extensions. Running this step in the PowerShell 7 console looks like this:



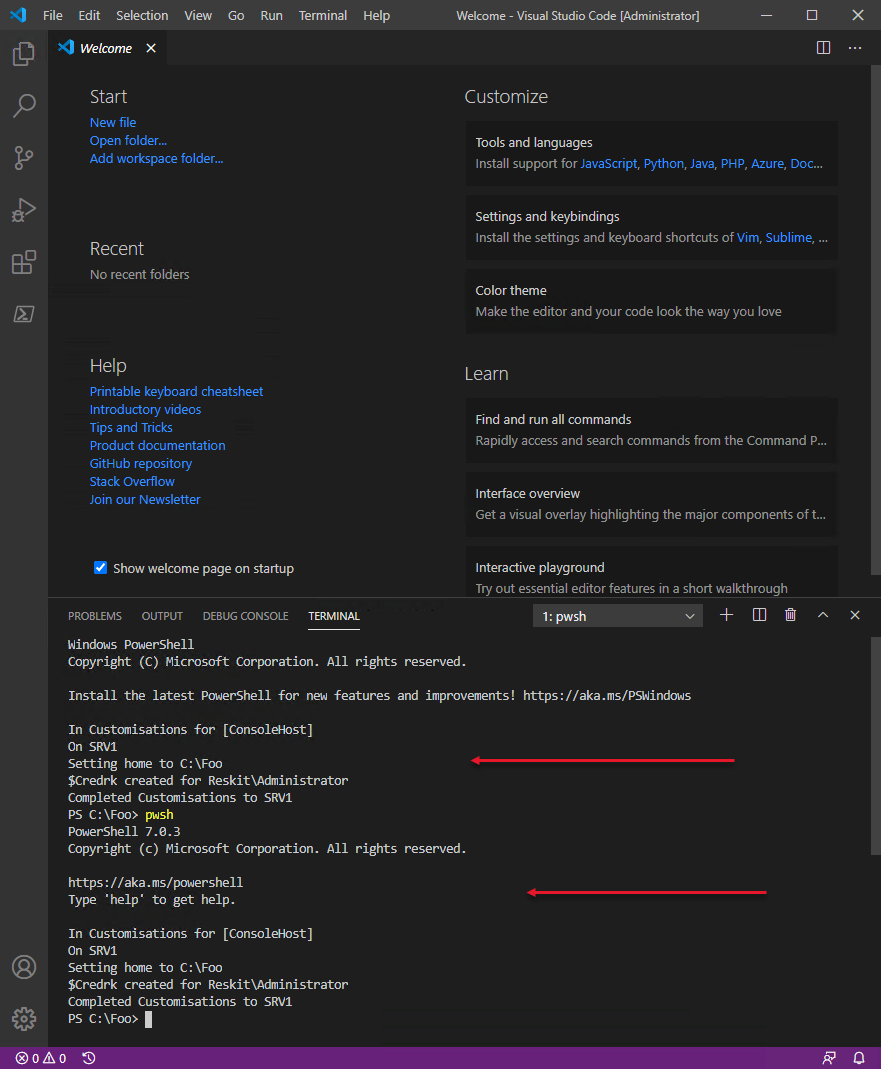
Insert image B42024\_01\_23.png

Once VS Studio has started, you can see the opening window, like this:



Insert image B42024\_01\_24.png

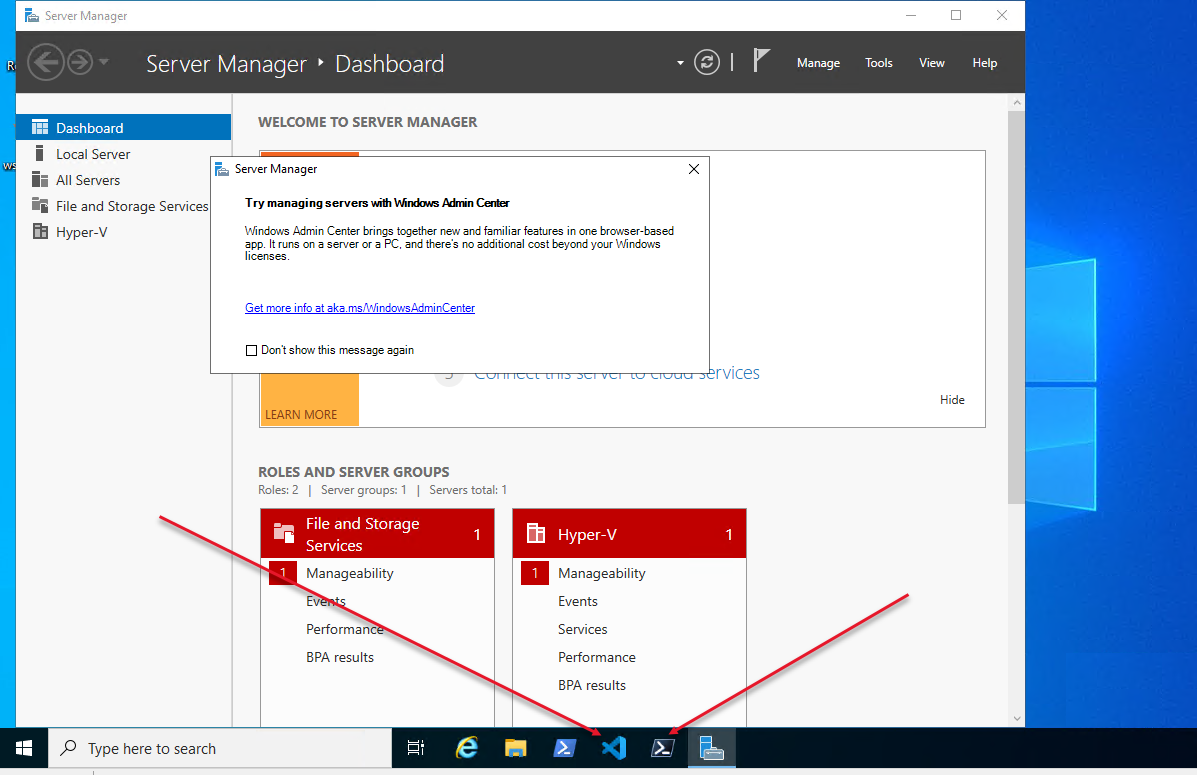
You then, in step 3, you close this window. You run VS Code as Administrator in step 4. In step 5, you open a new VS Code terminal and run PowerShell 7. which now looks like this:



Insert image B42024\_01\_25.png

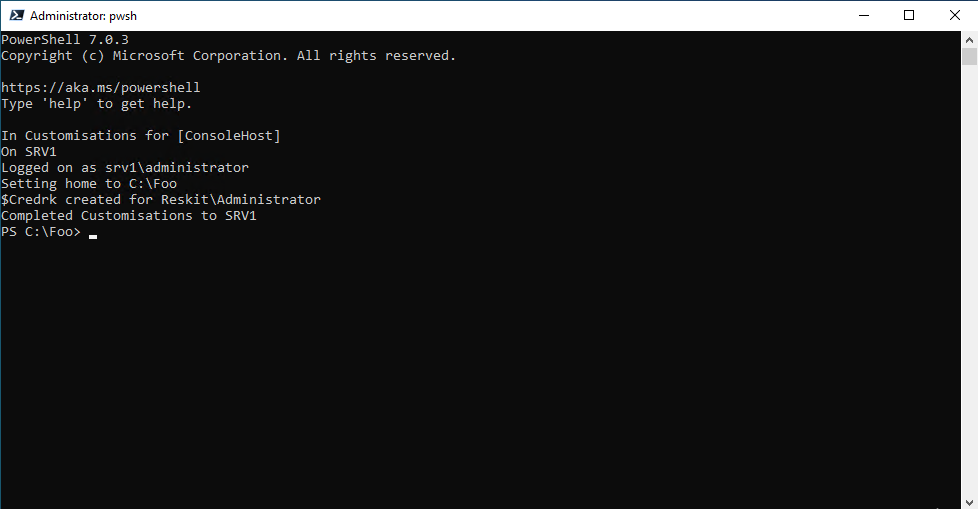
In step 6, you create a VS Code sample profile file. This step generates no output. In step 7, you update several VS Code runtime options. In step 8, you create a short cut to VS Code, and in step 9, you create a short cut to the PowerShell 7 console. In step 10, you update the XML that describes the Window task pane to add the shortcuts to VS Code ane the PowerShell console. In step 11, you import the updated task pane description back into Windows. In step 12, you create a VS Code profile for PowerShell 7. These steps produce no output as such.

Next, in step 13, you log off from Windows. In step 14, you re-login and note the updated taskbar, as you can see here:



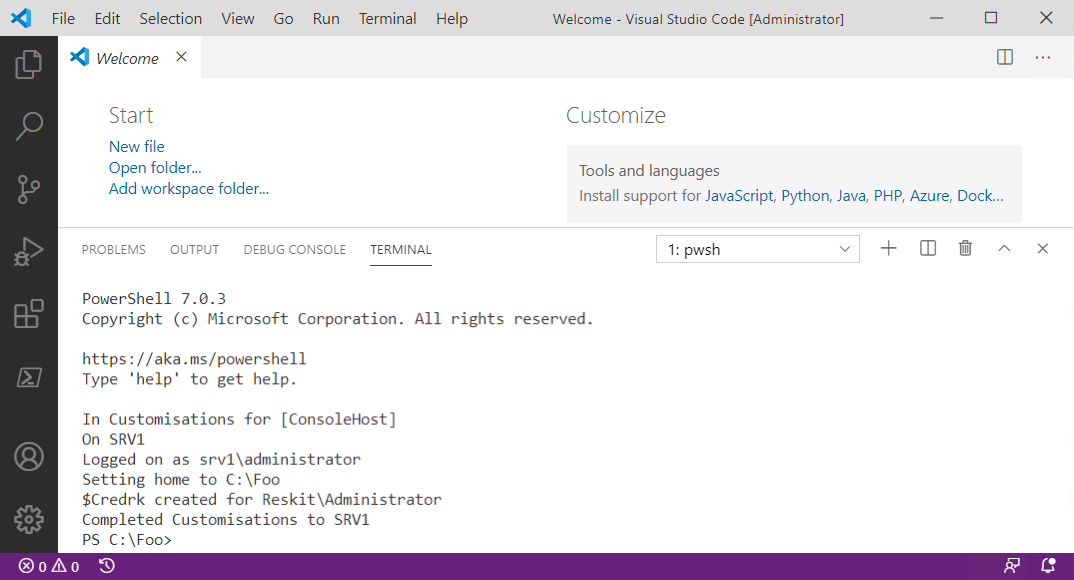
Insert image B42024\_01\_26.png

In step 15, you open a PowerShell 7 console which looks like this:



Insert image B42024\_01\_27.png

In step 16, you open VS Code, which looks like this:



Insert image B42024\_01\_28.png

## There's more...

In step 2, you install vs Code and three additional VS Code extensions. The Streetsidesoftware.code-spell-checker extension provides spell-checking for your scripts and other files. The yzhang.markdown-all-in-one extension supports the use of Markdown. This extension is useful if, for example, you are writing documentation in Github or updating the existing public PowerShell 7 help information. The hediet.vscode-drawio extension enables you to create rich diagrams directly in VS Code. Visit the VS Code market place for details on these and other extensions.

In step 4, you ensure you are running VS Code as Administrator. Some of the code requires this and fails if you are not running PowerShell (inside VS Code) as an admin.

In step 5, you open a terminal inside VS Code. IN VS Code the ‘terminal’ is initially a Windows PowerShell console. You can see in the output the results of running the Profile file for Windows PowerShell. This terminal is the one you see inside VS code by default.

In step 7, you update and save some updates to VS Code settings. Note that in this step, you tell VS Code where to find the version of PowerShell you wish to run. You can, should you chose, change this to run a preview version of PowerShell or even the daily build.

In step 8 and step 9, you create shortcuts to VS Code and the PowerShell 7 console. In step 110 and you update the layout of the Windows Task Bar to include the two short cuts. Unfortunately, you have to logoff (as you do in step 13) before logging back into Windows where you can observe and use the two shortcuts.

# Exploring Error View and Get-Error

As part of the launch of Visual Studio Code, Microsoft also created a new and free type font which you can download and use both at the PowerShell 7 console and inside VS Code. This recipe shows how you can download the font, install it, and set to be the default in VS Code.

## Getting Ready

You run this recipe on SRV1 after you have installed both PowerShell 7 and VS Code.

## How to do it...

1. Get Download locations for the Cascadia Code font

## How it works...

## There's more...