1

Installing and Configuring PowerShell 7

This chapter covers the following recipes:

* Installing PowerShell 7
* Installing PowerShell 7 using Chocolately
* Using the PowerShell 7 Console
* Building PowerShell 7 profile files
* Exploring Installation artifacts
* Installing VS Code
* Installing the Cascadia Code font
* Exploring PSReadLine

# Introduction

Since moving to Open Source, PowerShell Core or PowerShell 7 (or just plain PowerShell), has been a continuing work in progress. Windows PowerShell, first introduced to the public in 2003, was first released formally, as Windows PowerShell v1, in 2006. Over the next decade, Microsoft released multiple versions of Windows PowerShell, ending with PowerShell 5.1.

Initially, Windows PowerShell was an add-in but Microsoft wisely made Windows PowerShell to an integrated feature of Windows. Microsoft plans to support Windows PowerShell 5.1 for a long time, but no new features are likely.

In 2016, the PowerShell development team began working on an open-source version of PowerShell based on the open-source version of .NET Core. You can read the announcement by Jeffrey Snover here: https://azure.microsoft.com/en-us/blog/powershell-is-open-sourced-and-is-available-on-linux/.

The initial versions, PowerShell Core, Versions 6.0, 6.1 and 6.2, represented, in effect, a proof of concept – you could run the core functions and features of PowerShell across the Windows, Mac, and Linux platforms. Those early versions also enabled the development team to implement all the necessary tooling to allow future development. But they were quite limited in supporting the rich needs of the IT professional community.

With the release of PowerShell 7.0 came improved parity with Windows PowerShell. A few modules did not work with PowerShell 7, and a few more operated via a compatibility mechanism. PowerShell 7.0 shipped in 2019 and was followed by version 7.1 and Version 7.2 (released in late 2021). This book uses the term “PowerShell 7” to include both PowerShell 7.0, 7.1, and 7.2. If there are version-specific issues, the chapters call those out specifically.

Because Microsoft does not include PowerShell 7 in Windows, you have to install it on each system. And as ever, you have options including direct from GitHub and via other installers such as Chocolatey.

Once you have installed PowerShell 7, you can use it just as you used the Windows PowerShell console to run commands or scripts. You can run it from a shortcut on the desktop, from the start panel, from a shortcut on the taskbar, or just run the executable. The name of the executable for PowerShell 7 is pwsh.exe (versus powershell.exe for Windows PowerShell).

Another important difference is that PowerShell 7 uses different profile file locations from Windows PowerShell. This feature allows you to customize your profiles to use the new PowerShell 7 features. And that, in turn, enables you to run both Windows PowerShell and PowerShell 7 side by side without interference.

Most IT Pros who have used Windows PowerShell are familiar with the Integrated Scripting Environment (ISE). The ISE was a great tool you used with Windows PowerShell to help you to develop production scripts. However, the ISE does not support PowerShell 7 and (probably!) never will. A very worth successor to the ISE is Visual Studio Code (VS Code), an open-source editing project that provides all the features of the ISE and a great deal more. Installation of VS Code is optional but relatively straightforward. It is also easy for you to customize VS Code.

Microsoft also developed a new font, Cascadia Code, to coincide with the launch of VS Code. This font is a nice improvement over Courier or other mono-width fonts. All screenshots of working code in this book use this new font.

PSReadLine is a PowerShell module designed to provide color-coding of PowerShell scripts in the PowerShell 7 console. The module, included with PowerShell 7 by default, makes editing at the command line easier and more on par with the features available in Linux shells. You can also use the later versions of PS Readline with Windows PowerShell.

## System used in the chapter

This chapter is all about getting you started with PowerShell 7 – installing and configuring your environment to make the most out of PowerShell 7. In this chapter, you use a single host, SRV1, as follows:



Figure 1.1: Host in use for this chapter

Insert image B18878\_01\_01.png

In later chapters, you will use additional servers and will promote SRV1 to be a domain-based server rather than being in a workgroup.

# Installing PowerShell 7

As mentioned, PowerShell 7 is not installed in Windows by default, at least not at the time of writing. The PowerShell team made PowerShell 7.1 available from the Microsoft Store, which is useful to install PowerShell 7.1 or later on Windows 10/11 systems. Windows Server does not6 support the Microsoft store.

You have other methods of installing PowerShell 7 on your systems. The first option is to use the Install-PowerShell.ps1, which you download from GitHub, as shown in this recipe. You can also use this recipe on Windows 10 hosts. This approach has the advantage of being the most up-to-date source of the latest versions of PowerShell.

## Getting ready

This recipe uses SRV1, a Windows Server workgroup host. There are no features of applications loaded on this server (yet).

You can use either the Windows PowerShell console or the ISE for this recipe.

## How to do it...

1. Setting an execution policy for Windows PowerShell

Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force

1. Updating help text for Windows PowerShell

Update-Help -Force |

  Out-Null

1. Ensuring the C:\Foo Folder exists

$LFHT = @{

  ItemType    = 'Directory'

  ErrorAction = 'SilentlyContinue' # should it already exist

}

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

1. Downloading PowerShell 7 installation script from GitHub

Set-Location -Path C:\Foo

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

Invoke-RestMethod -Uri $URI |

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

1. Viewing Installation Script Help

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

1. Installing PowerShell 7.2

$EXTHT = @{

  UseMSI                 = $true

  Quiet                  = $true

  AddExplorerContextMenu = $true

  EnablePSRemoting       = $true

}

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

1. Installing the preview and daily builds (for the adventurous)

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

  Out-Null

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

  Out-Null

1. Creating Windows PowerShell default profiles

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

       '/scripts/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

$ChildPath      = 'Microsoft.PowerShell\_profile.ps1'

$ConsoleProfile = Join-Path -Path $ProfilePath -ChildPath $ChildPath

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

  Out-File -FilePath  $ConsoleProfile

1. Checking versions of PowerShell 7 loaded

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

## How it works...

In step 1, you set the execution policy for Windows PowerShell to Unrestricted. This step, which produces no output, simplifies the installation and setup of PowerShell. In production, you may wish to set PowerShell’s execution policy to be more restrictive.

In step 2, you update the help text files for Windows PowerShell, which produces output like this:

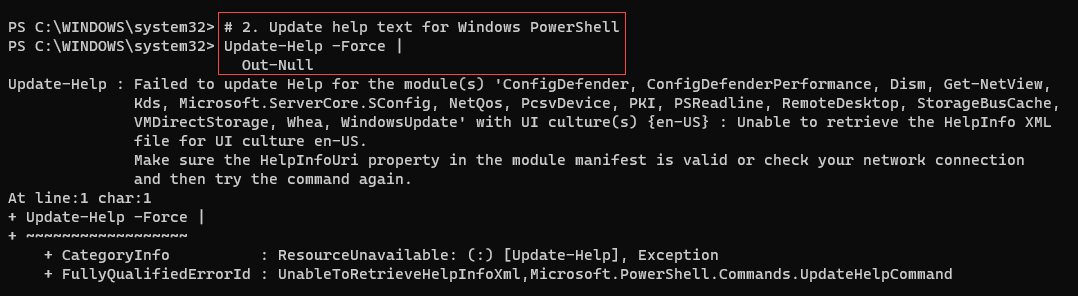


Figure 1.2: Updating Help Files

Insert image B18878\_01\_02.png

Note that after installing PowerShell 7, PowerShell prompts you to download help text (not shown in this figure) the first time you use Get-Help.

In step 3, you create a folder, C:\Foo. This book uses this folder as a place to put files used by the book’s recipes. For example, this recipe stores the PowerShell installation file in this folder from which you execute the script to install PowerShell 7.

With step 4, you download the PowerShell installation script from GitHub. Although you can look in C:\Foo to examine the script, this step produces no output.

The installation script is a PowerShell script. In step 5, you use Get-Help to get details on the script, as shown here:

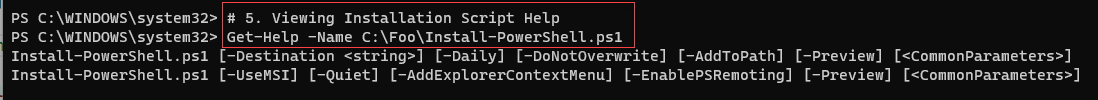


Figure 1.3: Getting help information from the installation script6

Insert image B18878\_01\_03.png

In step 6, you use the installation script to install PowerShell 7 on SRV1, with output like this:

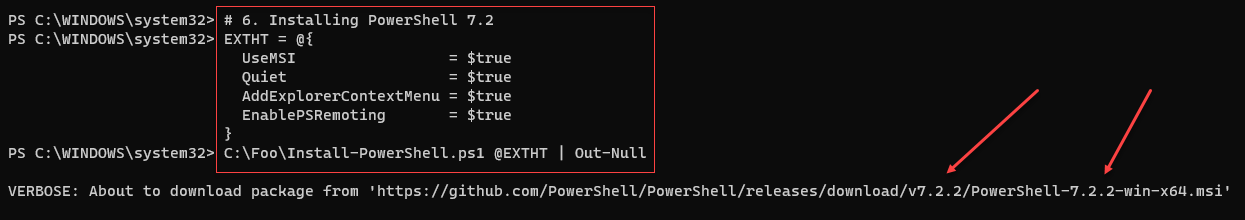


Figure 1.4: Installing PowerShell 7

Insert image B18878\_01\_04.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 which may be in the next major release. The daily build enables 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:

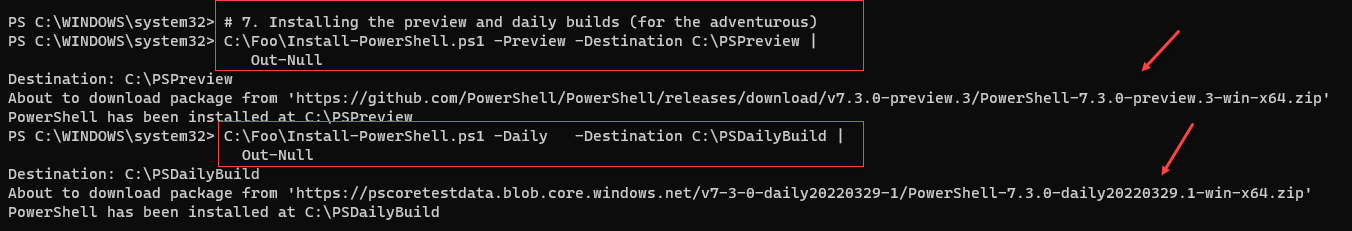


Figure 1.5: Installing the preview and daily builds

Insert image B18878\_01\_05.png

PowerShell, like Windows PowerShell, uses profile files to enable you to configure PowerShell each time you run it (whether in the PowerShell console or as part of VS Code. In step 8, you download a sample PowerShell profile script and save it locally. Note that the profile files you create in step 8 are for Windows PowerShell only. This step produces no output

The executable name for PowerShell 7 is pwsh.exe. In step 9, you view the versions of this file as follows:

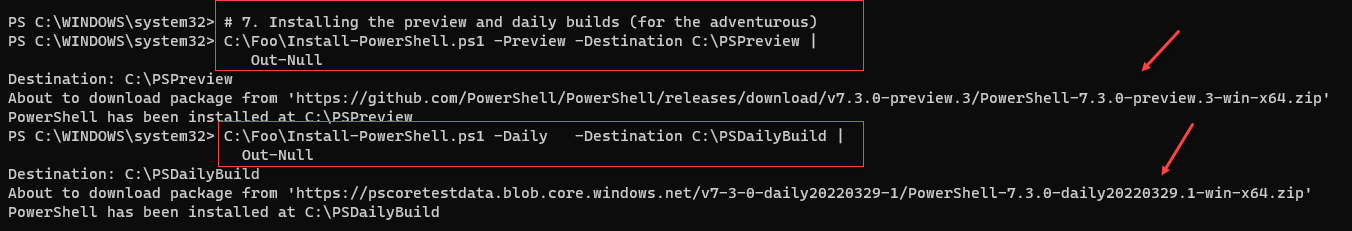


Figure 1.6: Checking PowerShell 7 versions loaded

Insert image B18878\_01\_06.png

As you can see, there are three 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 1, you update the execution policy for Windows PowerShell. While this simplifies the installation and configuration of hosts, it may be unduly permissive for your environment, and you can change it as needed. Don’t forget, though. PowerShell’s execution policy is not truly a security mechanism – it just slows down an inexperienced administrator. For a good explanation of PowerShell’s Security Guiding Principles, see <https://devblogs.microsoft.com/powershell/powershells-security-guiding-principles/>.

In step 2, you updated the help files for Windows PowerShell. This step is optional, but later steps can prompt you to update your help files if you skip it. Installing the most up-to-date help files also add many conceptual help topics to help you get more out of PowerShell.

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 6, you use the installation script and install PowerShell 7. This step installs PowerShell 7.2.2, as you can see, using an MSI. The MSI, which you install silently without any user notification) updates the system execution path to add the PowerShell 7 installation folder. At the time of writing, the latest released version of PowerShell is 7.2.2. The code here retrieves the latest supported version of PowerShell 7, which may be later than you see in the output here.

In step 7, you can see that you have installed PowerShell 7 (into C:\Program Files) and the latest daily build and the latest preview versions. The specific file versions you see may differ from the output shown here, reflecting the relentless progress of the PowerShell team.

**The rest of this chapter is just a placeholder holding the sections to be filled in later.**

# Installing PowerShell 7 Using Chocolatey

Chocolatey is a third party package management tool for Windows. You can read more about ther company and its products at https://chocolatey.org/. Chocolatey has a very large online registry of Windows packagres which you can easily install. This can simplify deployment of applications in your environment.

## Getting ready

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

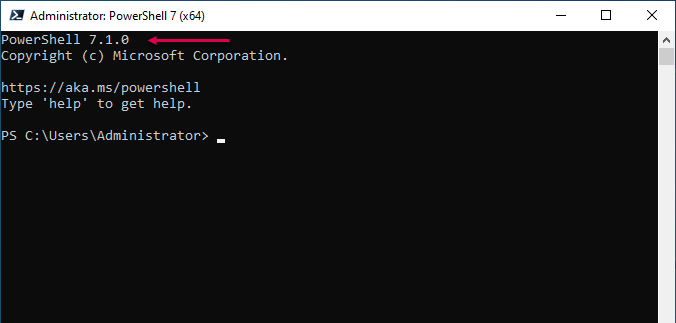


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator

# Using the PowerShell 7 console

## Getting ready

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

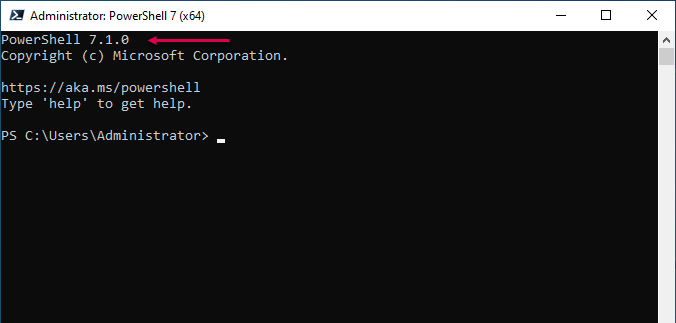


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator

# Building PowerShell 7 profile files

## Getting ready

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

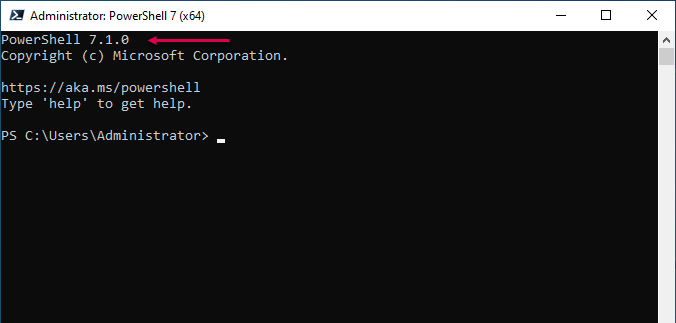


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator

# Exjploring Installation artifacts

## Getting ready

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

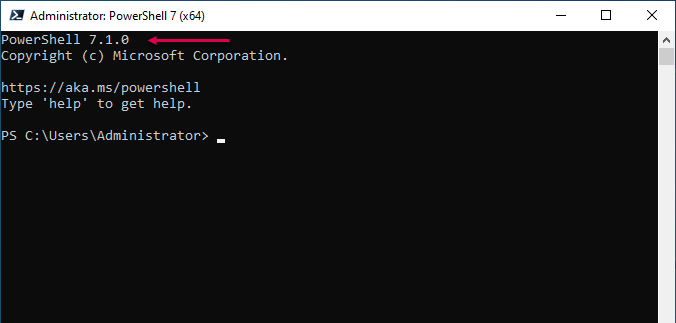


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator

# Installing VS Code

## Getting ready

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

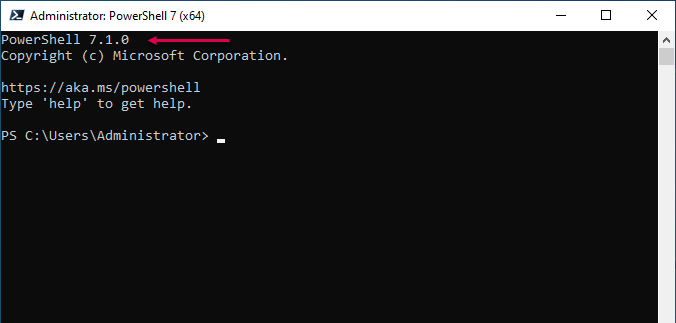


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator

# Installing the Cascadia Code font

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

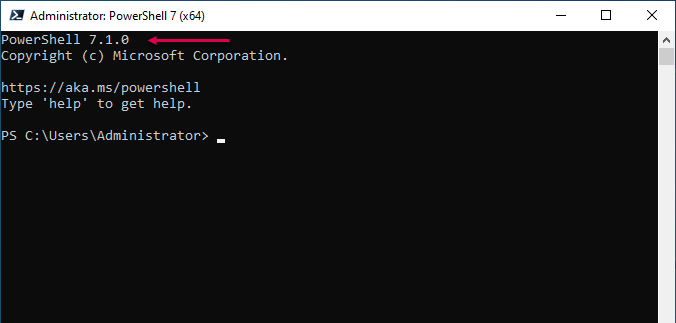


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator

# Exploring PSReadline

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

## How to do it...

1. Running the PowerShell 7 console

## How it works...

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

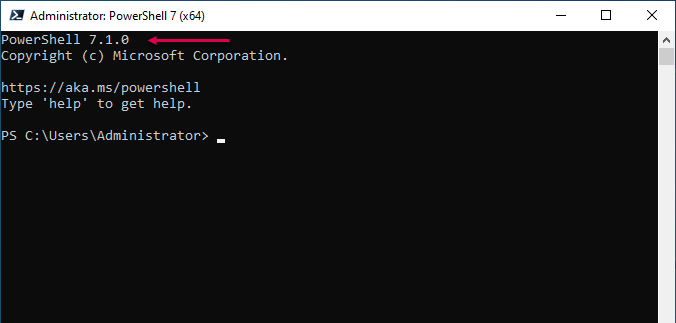


Figure 1.5: The PowerShell 7 console

Insert image B42024\_01\_05.png

In

## There's more...

In step 1, you open a new Windows PowerShell console. Make sure you run the console as the local administrator