# PowerShell in 7 Days

*Learn essential skills in scripting and automation using PowerShell*

**Liam Cleary**

www.bpbonline.com

First Edition 2024



ISBN: 978-93-55518-910



## LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

## Dedicated to

*My beloved wife:*
***Lisa***
*and*
*My Children* ***Bethan***, ***Joel***, ***Aidan*** *and* ***Noah***

# About the Author

**Liam Cleary** has been deeply immersed in computer training, programming, and cybersecurity, discovering his passion for these fields early in his career. His professional journey led him to work extensively within core infrastructure and security services. He founded SharePlicity, a consultancy specializing in Microsoft 365 and Azure technologies, where he spearheads efforts to enhance collaboration, document management, and business process automation and implements robust security controls. A Microsoft MVP Alumni with 17 years of recognition and a Microsoft Certified Trainer, Liam excels in architecture, security, and bridging the gap to software development. His recent focus has been on security within Microsoft 365, Azure, and related platforms. As an educator, Liam develops online courses for platforms like Pluralsight, LinkedIn Learning, Cloud Academy, and Cybrary IT and teaches Microsoft Certification courses for Opsgility and Microsoft. Beyond professional pursuits, Liam is actively involved in community events, from user groups to conferences, sharing knowledge, coding with his kids, and engaging in various outdoor adventures.

# About the Reviewer

**Carole McNally** is a tech industry veteran with an extensive 20-year journey. Rooted in the principles of neurodiversity and a staunch advocate for lifelong learning, Carole has dedicated her career to making technology accessible to all.

As an Integration Specialist, Carole excels in crafting seamless connections through a decade of expertise in multilanguage ETL. Her passion lies in innovating solutions that bridge the gap between technology and users.

Carole's proficiency shines in developing robust integration solutions, utilizing in-house products, APIs, and third-party tools. With a dynamic approach, she excels in configuring front-office products, expanding the Microsoft Bot Framework, and implementing Adaptive Cards to optimize user experiences and operational efficiency.

Within Techwitch Ltd, Carole has provided invaluable technical consultancy, achieving Cyber Essentials certification, troubleshooting complex issues, and overseeing successful migrations. Certified as a Conversational Designer, her skills extend to Azure Cognitive Services and API integrations, showcasing a commitment to staying at the forefront of industry advancements.

# Acknowledgement

I want to express my deepest gratitude to my family and friends, especially my wife, Lisa, and my children, Bethan, Joel, Aidan, and Noah, for their unwavering support and encouragement throughout this book's writing.

I am also grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. Writing this book was a long journey, with the valuable participation and collaboration of reviewers, technical experts, and editors.

I also want to acknowledge the valuable contributions of my colleagues and co-workers during many years working in the tech industry, who have taught me so much and provided valuable feedback on my work.

Finally, I thank all the readers who have taken an interest in my book and for their support in making it a reality. Your encouragement has been invaluable.

# Preface

"PowerShell in 7 Days" is a comprehensive guide that aims to demystify the world of PowerShell scripting and automation. PowerShell is a must-have skill set for IT professionals and system administrators. This book guides you through PowerShell, making it an indispensable resource for newcomers and seasoned professionals aiming to refine their automation skills.

Throughout the book, you will embark on a structured journey that transforms you from a PowerShell novice to a proficient scripter. Starting from the very foundations of PowerShell, the book unravels its syntax and core functionalities. As you progress, you will delve into more advanced topics such as creating complex scripts, managing data locally and remotely, and effectively using PowerShell for problem-solving and system troubleshooting.

The book is for beginners and IT veterans looking to integrate PowerShell into their workflow. Through fundamental concepts and hands-on examples, you will learn to leverage PowerShell's capabilities to streamline your daily tasks, enhance system performance, and automate repetitive and complex operations. The focus is on the 'how' and the 'why,' providing a deep understanding of PowerShell's potential in various job roles.

By the end of "PowerShell in 7 Days," you will have gained a comprehensive understanding of PowerShell's capabilities and how to harness them effectively. The book is a learning tool and a reference guide that helps you in your professional journey. Whether automating a small task or tackling a large-scale enterprise operation, the insights and skills acquired here will be valuable to any professional toolkit. Welcome to the world of PowerShell – let's embark on this journey together.

**Chapter 1: Introducing PowerShell –** This chapter provides a comprehensive introduction for readers to understand and leverage PowerShell, Microsoft's task automation and configuration management framework. It includes essential information on the initial setup, guidance on the tools required for writing and executing scripts, and an explanation of PowerShell's powerful capabilities for automating a wide range of administrative tasks. The chapter also presents a historical perspective of PowerShell's evolution, its various versions, and the current landscape, setting the stage for why learning PowerShell is invaluable for IT professionals. Readers will also discover the diverse contexts in which

PowerShell can be applied, from basic system administration to complex automation scenarios.

**Chapter 2: Setting Up PowerShell –** This chapter provides a detailed guide to help readers install PowerShell on different platforms. It offers specific instructions for each installation method, including using the PowerShell Gallery for module management and direct installable packages for Windows users. The chapter also explains how to set up PowerShell on macOS and Linux. Additionally, the chapter includes a comparative overview of the various installation methods available. It helps readers make informed choices that suit their specific requirements and environment.

**Chapter 3: Getting Started with Modules and Providers –** This chapter provides a comprehensive understanding of the PowerShell environment, focusing on the critical roles of modules and providers. In this chapter, readers will learn how to use PowerShell modules in detail, including their purpose and how to leverage both built-in and external modules to enhance the shell's capabilities. The various types of providers and how to interact with different data stores within PowerShell are explained. The chapter also simplifies the help system, a valuable resource for mastering command syntax and functionality. Moreover, the chapter discusses the integration and use of WMI and CIM, demonstrating how to use these powerful tools for system management and information retrieval.

**Chapter 4: Executing PowerShell Commands –** This chapter provides readers with the necessary knowledge to execute PowerShell commands effectively in various environments. It explains the different types of cmdlets available and the appropriate way to use them within a Windows setting. The chapter also focuses on specific modules, teaching readers how to use specialized commands tailor-made for their scripting needs. Additionally, it emphasizes the importance of command outputs, allowing readers to interpret and use the data returned from their scripts. Furthermore, the chapter thoroughly explores the integration of PowerShell with Visual Studio Code, demonstrating how this powerful editor can enhance scripting efficiency. By the end of this chapter, readers will have acquired the skills to write and debug PowerShell scripts within Visual Studio Code, laying a solid foundation for advanced scripting and automation.

**Chapter 5: Working with Variables and Pipelines –** This chapter is about using variables and pipelines to manage data flow and storage in scripting. In this chapter, you will learn about the pipeline operator, a fundamental feature of PowerShell that allows for seamless output transfer between commands. The chapter covers the creation, manipulation, and management of variables, from basic data types to complex objects, giving readers the skills to handle data dynamically within their scripts. Additionally, it discusses the

nuances of declaring and casting variable types to ensure data integrity and type safety. The chapter also covers advanced techniques for passing variables between commands, which enhances script modularity and reusability. Finally, readers will learn sophisticated strategies for filtering data within pipelines, allowing for refined and precise data operations. With practical examples and clear explanations, this chapter will give you a robust understanding of these core PowerShell concepts, ready to implement them in real-world scenarios.

**Chapter 6: Deep Diving PowerShell Objects –** This chapter provides detailed insights into PowerShell objects and their usage in scripting. It begins with the basics of PowerShell objects, which are crucial for utilizing the scripting language to its full potential. The chapter covers managing and creating string arrays, a fundamental skill for efficient data organization and handling. Further, it delves into the intricacies of object properties and methods, enabling the readers to manipulate custom data structures adeptly. The chapter also introduces the [PSCustomObject] data type, which offers a dynamic approach to object creation. Precisely defining data types within PowerShell variables is also emphasized as a critical practice for script accuracy. The chapter explores export and import cmdlets, which provide a pathway for maintaining data persistence. Lastly, the chapter concludes by demonstrating how PowerShell can be connected with .NET objects, expanding the reader's automation toolkit with the vast capabilities of the .NET framework.

**Chapter 7: Using Functions and Parameters –** This chapter aims to help readers learn how to create and use PowerShell functions, which will transform how they script and automate tasks. The chapter starts with the basics of crafting PowerShell functions, which will teach you how to encapsulate and modularize code for reusability and maintainability. After that, we move on to the critical aspect of determining the output of functions, which ensures that each function serves its intended purpose effectively. The journey continues by incorporating parameters into functions, significantly enhancing flexibility and adaptability for various use cases. The chapter further guides you through setting default values and defining data types for parameters, which is pivotal in maintaining data integrity and script reliability. The chapter also covers advanced features like parameter validation and input masks, offering methods to safeguard input and streamline function execution. Finally, the chapter culminates by demonstrating the integration of these elements into complex scripts, showcasing the power of tasks in creating sophisticated and efficient PowerShell tools.

**Chapter 8: Flow Control, Looping, and Error Handling –** This chapter provides comprehensive insights into the techniques of controlling script execution using loops and error handling, which are crucial for robust PowerShell scripting. The chapter starts

with an overview of looping constructs within PowerShell, followed by an in-depth exploration of the ForEach-Object command and foreach loops, which are vital for iterating over collections. The chapter then delves into the Switch command, which streamlines decision-making in scripts based on conditional logic. The latter part of the chapter is dedicated to the foundational concepts of error handling in PowerShell, emphasizing its critical importance in scripting for preventing and managing runtime exceptions. Readers will learn to implement error-handling mechanisms that ensure scripts execute gracefully, even when encountering unforeseen issues. By the end of this chapter, readers will be equipped with the knowledge to manage the flow of their PowerShell scripts effectively, confidently address errors, and maintain control over script outcomes in various scenarios.

**Chapter 9: Scripts for Multiple Output Paths –** This chapter simplifies the process of managing the output of PowerShell scripts across different destinations. This chapter serves as a foundation for understanding PowerShell's output redirection, a valuable feature that allows you to route script results to the appropriate endpoints. It teaches you how to use output redirection operators precisely, directing outputs to files, consoles, or other processes. The chapter then covers the creation of commands capable of generating multiple output streams, enhancing script versatility. It also discusses techniques for piping output to various targets to ensure effective data distribution where needed. Additionally, the chapter presents the skill of splitting output into separate files, along with methods for appending data to existing files without overwriting valuable information. Finally, the chapter culminates with strategies for dynamically customizing output destinations using conditional logic and loops, tailoring the data handling to the script's context. The author provides practical examples to illustrate how these techniques come together, providing a template for scripts requiring complex output management.

**Chapter 10: PowerShell Remoting, WinRM, and the Invoke-Comma –** This chapter discusses using PowerShell Remoting, WinRM, and Invoke-Command for managing and scripting on remote computers using PowerShell. The chapter starts by introducing PowerShell remoting, which is an essential tool for remote management and automation. It explains Windows Remote Management (WinRM) and its role as the backbone of PowerShell remoting. The chapter also covers the different configurations, functionalities, and operational scope of WinRM. Furthermore, the chapter addresses the critical security aspect and highlights the best practices to ensure safe and secure remote operations. The readers are guided through the detailed steps to configure remoting in Windows and Linux environments, which will help them extend their administrative reach across different platforms. The chapter also explores the New-PSSession and Enter-PSSession cmdlets, allowing seamless remote connections to Windows and Linux systems. Finally, the chapter thoroughly examines the Invoke-Command cmdlet, which effectively executes commands

on remote computers. By the end of this chapter, readers will have a comprehensive understanding of how to use PowerShell for remote management and automation tasks, regardless of the operating system.

**Chapter 11: Managing On-premises Services –** The chapter covers many topics, including an introduction to specialized PowerShell commands designed for on-premises administrative tasks. It then provides a detailed guide on configuring and managing Active Directory, simplifying the complexities involved in user and group management and highlighting the effectiveness of PowerShell in these critical areas. The chapter also delves into the management of domain controllers, providing insights into the nuances of this crucial aspect of network infrastructure. It covers essential network services such as the **Domain Name Service (DNS)** and **Dynamic Host Configuration Protocol (DHCP)**, demonstrating how PowerShell can be a powerful tool for managing them. Additionally, the chapter explains how to create and manage file shares, which is a crucial task in maintaining an organization's data accessibility and security. The chapter also explores managing certificates within Windows Server to ensure a comprehensive understanding of this vital security component. Finally, the chapter concludes by providing insights into managing various server roles and features, showcasing PowerShell's versatility in handling multiple server management tasks. This chapter aims to empower the reader with the knowledge and skills necessary to manage and optimize on-premises services using PowerShell, turning routine administrative tasks into streamlined processes.

**Chapter 12: Troubleshooting Windows and Performance Optimization –** This chapter starts by delving into system resource analysis, teaching readers how to scrutinize CPU, memory, disk, and network usage using PowerShell commands. This foundational knowledge is vital for identifying and addressing performance issues hindering system responsiveness and reliability. The chapter then transitions into the nuances of diagnosing and resolving performance bottlenecks, offering practical techniques to pinpoint and alleviate these critical issues. It teaches readers with PowerShell command expertise, such as utilizing ping, traceroute, DNS lookups, and firewall configurations, to diagnose and resolve network problems effectively. In the security realm, the chapter provides in-depth insights on identifying and remediating security vulnerabilities. Readers will learn to employ PowerShell for essential security tasks, including malware scanning, updating Windows Defender, and applying necessary security patches. The chapter also provides readers with the skills to analyze and filter event logs. By mastering these techniques, readers can extract critical information about errors, warnings, and events that could signify underlying system issues. Lastly, the chapter addresses common pitfalls and challenges of using PowerShell for Windows troubleshooting. It guides handling errors,

exceptions, and permissions, ensuring readers are well-prepared to tackle real-world troubleshooting scenarios with confidence and expertise.

**Chapter 13: Miscellaneous PowerShell Capabilities –** The final chapter covers PowerShell security. It explains how to secure PowerShell environments, regulate script execution, and ensure a secure scripting environment. The chapter also discusses methods for maintaining computer and data security while using PowerShell and implementing AppLocker policies to turn off unauthorized PowerShell scripts. Additionally, it explains the importance of signing PowerShell scripts for script integrity and authenticity. Finally, the chapter offers valuable insights and guidance for IT administrators on the next steps in their journey with PowerShell.

# Code Bundle and Coloured Images

Please follow the link to download the
***Code Bundle*** and the ***Coloured Images*** of the book:

## https://rebrand.ly/9faa44

The code bundle for the book is also hosted on GitHub at
**https://github.com/bpbpublications/PowerShell-in-7-Days**.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

# Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

**errata@bpbonline.com**

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

## Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

## If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

## Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

# Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

**https://discord.bpbonline.com**

# Table of Contents

# Chapter 1
# Introducing PowerShell

## Introduction

This chapter will introduce you to PowerShell. We will start by reviewing the history of PowerShell, discussing versions and the current state of PowerShell, as well as providing an understanding of what PowerShell is and why you should consider using it.

## Structure

In this chapter, we will cover the following topics:

- Introduction to PowerShell
- PowerShell versions and current state
- What is PowerShell?
- Who is PowerShell for?
- When should you use PowerShell?

## Objectives

By the end of this chapter, you will understand PowerShell, its history, versions, current state, and rationale for usage.

# Introduction to PowerShell

In November 2006, Microsoft released the first version of PowerShell, the initial release for Windows only, aiming to automate and simplify administrative tasks in Windows environments. Microsoft created PowerShell as a powerful scripting language and task automation framework to surpass the capabilities of the traditional command prompt.

In 2016 Microsoft released PowerShell Core, expanding the reach of the PowerShell scripting beyond Windows to include macOS and Linux platforms. It has evolved through multiple versions, with each iteration bringing new features, improvements, and cross-platform compatibility.

Today, PowerShell is an indispensable tool for system administrators and developers, offering a robust and flexible solution for managing and automating various tasks across diverse environments.

# PowerShell versions and current state

The initial version of PowerShell was a Windows component only, known as Windows PowerShell. It only worked within Windows operating systems and provided a limited subset of capabilities for management. In August 2016, with the introduction of PowerShell Core, Microsoft made it open-source and cross-platform. The last version of Windows PowerShell is Windows PowerShell 5.1, and PowerShell 7+ is the successor to PowerShell Core 6+. The following table outlines the Windows PowerShell, PowerShell Core, and PowerShell versions and their respective release dates:

| Type | Version | Release Date |
|---|---|---|
| Windows PowerShell | 1.0 | November 2006 |
| Windows PowerShell | 2.0 | July 2009 |
| Windows PowerShell | 3.0 | October 2012 |
| Windows PowerShell | 4.0 | October 2013 |
| Windows PowerShell | 5.0 | February 2016 |
| Windows PowerShell | 5.1 | August 2016 |
| PowerShell Core | 6.0 | January 2018 |
| PowerShell Core | 6.1 | September 2018 |
| PowerShell Core | 6.2 | March 2019 |
| PowerShell | 7.0 | March 2020 |

| Type | Version | Release Date |
|---|---|---|
| PowerShell | 7.1 | November 2020 |
| PowerShell | 7.2 | November 2021 |
| PowerShell | 7.3 | November 2022 |

***Table 1.1:*** *PowerShell versions and releases dates*

PowerShell 6+ is installed side-by-side with earlier PowerShell releases. There are two editions of PowerShell 6+, the desktop and core versions. The desktop version runs on the .NET Framework, and the core version runs on .NET Core. PowerShell 7+, built on .NET Core 3.1, adds significant functionality compared to Windows PowerShell and aims to maintain full backward compatibility with Windows PowerShell.

The .NET Foundation and Microsoft develop and maintain PowerShell versions. They shifted the development focus to improve its capabilities, performance, cross-platform compatibility, and operability with different cloud platforms and services. The latest release supports Windows x64 / x86, Ubuntu, Debian, CentOS, Red Hat Enterprise Linux, OpenSUSE, Fedora, and macOS and supports multiple processor types. PowerShell 7+ is available for anyone to access and contribute, and it is available within the GitHub platform: **https://github.com/PowerShell**.

# What is PowerShell?

PowerShell is a task-based command-line shell and scripting language built on the .NET and .NET Core Frameworks. Microsoft designed this tool to enable IT professionals to control and automate Windows administration. In contrast to traditional command-line interfaces, PowerShell revolves around objects. An object is a structure containing data and operations you can perform on this data.

PowerShell supports providers allowing access to data stores, such as the registry, certificate store, and file system. Commands, called cmdlets (pronounced "command-lets"), are single-function commands that manipulate objects in PowerShell. They are the native commands in the PowerShell stack. In cmdlet names, the verb indicates the action it performs, while the noun specifies the target of the action. A few example cmdlets are:

- This command lists the items, such as files or folders, in the current or specified directory:

  ```
  Get-ChildItem
  ```

- This command creates a new user in the active directory:

  ```
  New-ADUser
  ```

- This command removes specified capabilities from a Windows operating system image:

  ```
  Remove-WindowsCapability
  ```

- This command directly puts text or objects you specify into the clipboard:

  ```
  Set-Clipboard
  ```

The pipeline is a powerful feature in PowerShell, allowing you to pass objects, not just text, from one cmdlet to another. It enables you to perform complex multiple operations with a single line of code.

Lastly, PowerShell allows you to write scripts and functions to automate tasks. A script is a plain text file containing one or more PowerShell commands, while a function is a list of statements with a name you assign. IT professionals can also distribute written scripts, making it an excellent solution for consistently managing deployments and configuration.

# Who is PowerShell for?

Primarily, PowerShell serves IT professionals, system administrators, and developers who manage Windows-based systems, but its reach is much more comprehensive. Network engineers frequently use PowerShell for network-related tasks, from configuration to automation. **Database Administrators** (**DBAs**) often employ it in SQL Server environments, benefiting from its seamless database management capabilities. In the era of DevOps, the tool's scripting and automation proficiency prove indispensable for DevOps teams. Security experts also leverage PowerShell's robust features for system hardening and security audits. Cloud professionals, particularly those engaged with Azure, find its cmdlets helpful in managing cloud resources. PowerShell is widely used in various IT roles for its scripting, automation, and administrative capabilities, particularly in Microsoft environments.

# When should you use PowerShell?

PowerShell is a powerful tool for automating administrative tasks and managing systems, whether those systems are in the cloud or on-premises. Some of the main reasons to use PowerShell are:

- **Automation:** PowerShell automates repetitive tasks, reducing manual effort and errors, particularly in large setups.
- **Consistency:** Scripting ensures tasks consistently perform similarly, minimizing errors.
- **Efficiency**: Automation allows swift completion of complex tasks, freeing time for other duties.

- **Compatibility:** Being a Microsoft product, PowerShell works well with other Microsoft products like Windows Server, Exchange Server, and more.
- **Cross-platform:** PowerShell Core (version 6+) can run on Linux, macOS, and Windows, broadening its utility.
- **Object-based:** PowerShell works with complex data structures, not just text, making it more powerful than many scripting languages.
- **Community support:** With a large active community, finding scripts, solutions, and best practices becomes easy.
- **Built-in security:** PowerShell incorporates features like execution policy to run scripts securely.
- **Integration with .NET:** Being built on .NET, PowerShell can access .NET classes and objects, opening additional functionality.

# Conclusion

In conclusion, PowerShell has emerged as a crucial tool for automating tasks and managing systems. Since its inception in 2006, PowerShell has made significant strides in evolving its capabilities and extending its reach to various platforms. From being a Windows-specific component, it has transitioned to being cross-platform and open-source, encapsulating enhancements in its functionality and performance. The strength of PowerShell lies in its object-based operations, integration with .NET, compatibility with Microsoft products, and vast community support. With a range of features such as automation, efficiency, and built-in security, PowerShell is an indispensable tool for IT professionals in managing administrative tasks. As we continue this book, we will delve deeper into its functionalities, exploring PowerShell's immense potential in transforming system administration and task automation.

In the next chapter, we will review the installation of PowerShell, focusing on the various installation approaches and installing it in a non-Windows environment.

## Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

**https://discord.bpbonline.com**

# Chapter 2
# Setting Up PowerShell

## Introduction

This chapter guides you through installing PowerShell from various options like the PowerShell Gallery, installable packages, and on non-Windows platforms such as macOS and Linux. It provides an overview of these options, detailed instructions for each method, and considerations for non-Windows installations.

## Structure

In this chapter, we will cover the following topics:

- Overview of the PowerShell installation options
- Using the PowerShell Gallery
- Installing using an installable package
- Installing on a non-Windows platform

## Objectives

By the end of this chapter, you will understand the various options for installing PowerShell, including the PowerShell Gallery, installable packages, and alternatives for

non-Windows platforms like macOS and Linux. You will follow instructions for each method and consider the specific needs of non-Windows installations. This knowledge will enable you to install and use PowerShell in your chosen environment.

# Overview of the PowerShell installation options

Several options exist for installing PowerShell, depending on the platform and version you wish to install:

- **Windows PowerShell:** Windows PowerShell 5.1 comes pre-installed on most modern Windows operating systems, including Windows 10 and Windows Server 2016/2019.
- **PowerShell 7+ (Latest version):** PowerShell 7+, the successor to PowerShell Core 6+, is cross-platform and works on Windows, Linux, and macOS.
- **Windows Package Manager (Winget):** For newer Windows 10 and 11 versions, you can install the Windows Package Manager (Winget) to install PowerShell.
- **Docker:** If you want a completely isolated environment, you can use Docker to run PowerShell in a container.
- **Azure Cloud Shell:** If you are working with Azure, PowerShell is available in the Azure Cloud Shell. No installation is required.
- **Direct installation from a Linux terminal:** For Linux distributions, PowerShell can be installed directly from the terminal using specific commands.

Understanding what gets installed is critical to PowerShell installation. The primary component is the Windows PowerShell engine, and this core component enables the execution of PowerShell scripts and commands. It is responsible for various fundamental aspects of PowerShell's functionality, such as parsing commands, managing objects, controlling the pipeline, and invoking cmdlets.

The Windows PowerShell Engine has crucial components that make it work. The Command Parser interprets command inputs and prepares them for execution, working with cmdlets, functions, script blocks, and control structures. Unlike other shells that only handle text, PowerShell uses an object pipeline to pass objects between cmdlets quickly. Based on the parsed instructions, the engine activates these cmdlets, whether compiled commands, PowerShell functions, or scripts. This process occurs within the scripting host and interacts with the session state, including variables, functions, aliases, and more. Providers, or groups of cmdlets, provide a consistent interface to manage diverse data stores, with the engine facilitating access to data and storage types such as the registry, certificate store, and file system.

Microsoft Windows has pre-installed Windows PowerShell in every version since Windows 7 and Server 2008. The installed version of PowerShell is different depending on the operating system. The list below displays which versions of PowerShell are compatible with various Windows operating systems:

- **Windows Vista:** PowerShell 1.0
- **Windows 7:** PowerShell 2.0
- **Windows 8:** PowerShell 3.0
- **Windows 8.1:** PowerShell 4.0
- **Windows Server 2008/2008 R2:** PowerShell 2.0
- **Windows Server 2012:** PowerShell 4.0
- **Windows Server 2012 R2:** PowerShell 5.0
- **Windows Server 2016:** PowerShell 5.1
- **Windows Server 2019:** PowerShell 5.1
- **Windows 10:** PowerShell 5.1
- **Windows 11:** PowerShell 5.1

PowerShell 6+, or PowerShell Core, is cross-platform, enabling installation on various operating systems, and is the predecessor to PowerShell 7. However, unlike Windows PowerShell 5.1 and earlier versions, it does not come pre-installed with any operating system. Even the latest versions of Windows do not include the latest PowerShell; instead, they come with PowerShell 5.1.

**Note: Older versions of Windows operating systems (2008 R2 SP1, 2012, and 2012 R2) must have the Windows Management Framework 5.1 installed to support Windows PowerShell 5.**

Before installing PowerShell, it is essential to have specific prerequisites in place for your operating systems, such as Windows, Linux, macOS, Arm, or Docker. The list below outlines these essential requirements:

- **Windows**: When installing PowerShell on Windows operating systems, Winget is recommended for Windows clients, while an MSI package is best for Windows Servers and enterprise deployment. Despite its limitations, the Microsoft Store package is a more straightforward installation option, especially for casual PowerShell users. To install multiple versions or 'side load,' use the ZIP package method, which also works for Windows Nano Server, Windows IoT, and Arm-based systems. .NET developers may prefer the .NET Global tool as another viable option.

- **Linux**: Different Linux distributions support PowerShell installation. Most Linux platforms and distributions release a yearly update, including a package manager for installing PowerShell. Each version of Linux may require a different command for installation. There are alternate ways of installing PowerShell using the Snap Package manager, binary archives, or the .NET Global tool.
- **macOS**: Installing PowerShell on macOS requires 10.13 or higher. The preferred package manager for macOS, Homebrew, offers one such method. Alternatively, you can directly download PowerShell or install it from binary archives. Choose the method that best suits your needs.
- **Arm**: PowerShell's support for Arm processors aligns with the support policy of its underlying .NET version. Although .NET supports a wide range of operating systems, PowerShell's Arm support remains limited.
- **Docker**: Microsoft provides Docker images with PowerShell pre-installed. You need Docker 17.05 or later to run the released images. Additionally, you should be able to operate Docker without needing sudo privileges or local administrative rights.

You must choose the correct installation option for your operating system selected.

# Installing using an installable package

To install PowerShell on Windows, you have two options available for download from GitHub: **https://github.com/PowerShell/PowerShell/releases**:

- **MSI Package** is the easiest method and involves downloading a file with a name like **PowerShell-x.x.x-win-x64.msi** (for 64-bit systems) or **PowerShell-x.x.x-win-x86.msi** (for 32-bit systems), where **x.x.x** is the version number. Once downloaded, simply double-click the file to launch the Windows installation wizard.
- **ZIP Package** is useful if you prefer a portable version or don't have administrative rights to install software. The ZIP file will have a name like the MSI but with .zip at the end. After downloading, extract the files to a folder of your choice and run PowerShell directly from there.

There are four steps to install PowerShell:

1. **Download the MSI file**: Navigate to the releases page and download the appropriate MSI file based on your system architecture (64-bit or 32-bit).