PowerShell: A CHEAT SHEET

TechRepublic

TABLE OF CONTENTS

PowerShell: A cheat sheet

This comprehensive guide covers essential PowerShell information, including features, system requirements, and how Microsoft's framework extends to task automation and management.

BY JESUS VIGO

PowerShell was developed more than 10 years ago by Microsoft to expand the power of its command line interface (CLI) by coupling it with a management framework that is used to manage local and remote Windows, macOS, and Linux systems. By making use of the Component Object Model (COM), Windows Management Instrumentation (WMI), and Common Information Model (CIM) interface standards, PowerShell allows for managed elements of computing objects to be administered independent of the manufacturer or provider.

This cheat sheet will be updated when Microsoft releases new information throughout PowerShell's development lifecycle.

ILLUSTRATION: LISA HORNUNG

EXECUTIVE SUMMARY

- **What is PowerShell?** Microsoft's PowerShell is a management framework that combines a command-line shell and scripting language that is built upon the .NET framework for native Windows support or the .NET Core framework (which is open source), providing cross-platform support for Windows, macOS, and Linux.
- **Why does PowerShell matter?** PowerShell is capable of automating management tasks and functioning as a dedicated scripting language for both Microsoft- and non-Microsoft-based software. Also, PowerShell includes commands called cmdlets that add functionality to the core foundation, while providing a means to upgrade/update cmdlets to further augment functionality in the future versions.

- **Who does PowerShell affect?** Companies relying on Microsoft, Apple, and Linux services to empower their business functions, and the IT professionals who are responsible for managing this infrastructure.
- **When is PowerShell available?** PowerShell 5.1 (.NET) is closed source, the most recent version available, and supported for Windows-based systems. PowerShell 7 (.NET Core) is open source and the most recent version available, supporting Windows, macOS, and Linux operating systems. On Windows systems only, both versions may exist side by side without conflict. nets, on the other hand, are typically operated for nefarious purposes, and computers become nodes not by installing a program, but by being hijacked directly by hackers or through the installation of malware.
- **How can I get PowerShell?** PowerShell 5.1 (.NET) is a natively installed application that is part of all Windows client and server OSes; by default, the application can be updated directly from Microsoft's downloads website or through Microsoft Updates. You can get the latest version, PowerShell 7 (.NET Core), by visiting Microsoft's GitHub website for PowerShell and downloading the version that supports your operating system; this version of PowerShell may also be downloaded and updated via the native CLI of the operating system.

WHAT IS PowerShell?

Released as PowerShell 1.0 on Nov. 14, 2006, Microsoft developed PowerShell to address the shortcomings of its DOS-based CLI, particularly when managing objects using complex scripting languages.

By creating a new shell from the ground up, Microsoft effectively developed an extensible environment that would be powerful and flexible--it's capable of automating management tasks and functioning as a dedicated scripting language for Microsoft-based software.

Through various revisions, PowerShell has added modules to extend functionality to new objects as well as introduce new cmdlets for managing more resources, including Active Directory and Exchange Server. On Aug. 18, 2016, Microsoft announced that PowerShell was going open source and provided its source code to the public, adding support to Unix-based OSes, including Linux distros and OS X.

PowerShell 7 is the newest version of PowerShell and serves as a replacement console to both the previous versions of PowerShell Core 6.x and the Windows-only PowerShell 5.1. The latter serving as the last supported version of the Windows-only version of PowerShell, with the development team's aim being to condense all previous versions of PowerShell into one beginning with 7.0. This move, which is currently underway, will slowly bring PowerShell 7 into compatibility with previously unsupported cmdlets, further bringing it closer to parity for all supported operating system versions.

PowerShell includes a number of cmdlets with which to manage any number of system attributes, resources, and

objects--far beyond the scope of this guide. The following are some of the most notable features, modules, and cmdlets.

- **Active Directory (module)**: This module is used by PowerShell to extend management capabilities to Active Directory objects, including computers, users, and groups and attributes stored within accounts.
- **Exchange Server (module)**: This module is used by PowerShell to enable full administration of Exchange Servers. Included within the module are additional cmdlets that fully support all aspects of your Exchange email server.
- **Get-Help (cmdlet)**: This built-in cmdlet within PowerShell core provides helpful information, including syntax use and examples of commands and what they accomplish.
- **Get-Command (cmdlet)**: When executed, this built-in cmdlet within PowerShell core provides a list of commands that are available. It's useful in identifying which commands are available for each module.
- **Set-Variable (cmdlet)**: This built-in cmdlet within PowerShell allows the user to create variables used to store data, such as file paths, multiple objects, or snippets of code you wish to reuse.
- **Invoke-Command (cmdlet)**: This built-in cmdlet within PowerShell calls upon another cmdlet, usually run from a local computer, to execute the invoked command on remote computers.
- **Pipeline (|)**: One of the features of PowerShell is the ability to chain commands together by means of the pipe character. Piping commands causes PowerShell to run the first part of the command and then output the results for use by the second command and so on until the entire sequence is run. It is useful when performing a multiple-step task, such as creating a username, adding the username to a security group, and resetting the default password.
- **Function ({ })**: Similar to the pipeline feature in that cmdlets may be linked together, functions allow for greater control over the scripting process. By wrapping cmdlets in braces, a function is created that serves to run the sequence one or more times.
- **Out-File (cmdlet)**: This built-in cmdlet within PowerShell allows a command's output to be exported to a file. Typically used with the pipe feature, a user can get a list of user accounts that are disabled in Active Directory, for example, and export that list to a text file for future use.
- **Import-Module (cmdlet)**: This built-in cmdlet within PowerShell imports one or more modules into PowerShell to further its feature set, cmdlets, and functionality.
- **Third-party Modules**: Software developers can program code to group multiple cmdlets together as Third-party modules that are imported into PowerShell to extend functionality and support for specific applications. Notable third-party modules exist from VMware (virtualization), Dell (PowerEdge servers), and PowerSploit (Security/Pentesting).

CURRENT MINIMUM SYSTEM REQUIREMENTS FOR PowerShell 7

- x64-based processor and operating system
- Ubuntu LTS 16.04 or 18.04
- CentOS 7 or 8
- Arch Linux
- Kali Linux
- Alpine Linux 3.8
- Fedora 30
- Debian 9 or 10
- openSUSE 42.3
- macOS 10.1
- Windows 8.1 or 10
- Windows (ARM)
- Raspbian (ARM)
- Docker
- **Internet:** Broadband access (optional)

WHY DOES PowerShell MATTER?

Until the release of Windows 95, Microsoft chose to run Windows over DOS since it was the de facto OS in use on IBM-compatible computers. From Windows 95 on, Windows kept MS-DOS since some legacy applications still relied upon it.

MS-DOS also served as the means of administering devices through remote methods and by way of scripts that would be coded to automatically run tasks that were deemed repetitive and time-consuming to manage Windows computers.

MS-DOS was released in 1981, and Microsoft did not evolve its CLI (unlike its Unix-based competitors) until the initial development of PowerShell in 2006. In making this 25-year leap, PowerShell was designed as more than just a CLI replacement.

Microsoft created PowerShell as a management framework that combines both a command-line shell and scripting

language that is built upon .NET and the .NET Core and used as a software framework to standardize code, develop powerful applications, and cross-platform management of systems in heterogenous networks.

This results in PowerShell being used to manage hardware, software, and network objects at the command line, while also allowing programmers to use its scripting capabilities to interface with any manageable attributes to share data between them--including outputting code to develop applications to scale--from one personal computer through large enterprises that span the globe.

Open sourcing PowerShell allows for a cross-pollination of system administrators to manage multiple types of server OSes from just about any system--for example, managing Windows servers from macOS or maintaining Linux servers from Windows client machines. This level of flexibility is unprecedented and will be useful with standardizing management of different platforms across industries, especially when it comes to automating system management processes, as PowerShell 7 scripts created on Linux systems will work identically on macOS and Windows systems, easing administrative overhead.

WHO DOES PowerShell AFFECT?

PowerShell affects all types of users, from end users looking to be more productive to administrators seeking a simpler, more powerful solution to manage devices locally and remotely to developers writing their own applications to interface between hardware and software layers. PowerShell is the next step in Microsoft's CLI evolution, but also presents a major step toward unifying management processes across platforms that were previously quite disparate from one another.

PowerShell does require learning new commands, new syntax, and logic in order to reach its maximum potential. And yet, Microsoft is already bringing this point home through the use of PowerShell modules that serve to integrate with enterprise applications such as Exchange, SQL, and Windows Server to extend functionality and manageability.

Adding to this is the explosive growth of cloud-based services, such as Azure from Microsoft. With many of the management features available via GUI from the web-based interface, PowerShell 7 adds management support through Azure modules, further allowing administration of cloud-based infrastructure, including Azure Active Directory, among many other resources through the CLI.

Prior to the shift to open source, PowerShell affected only Windows administrators and those using the Microsoft family of products; however, now Linux and macOS administrators may option the ability to leverage PowerShell's open-source capabilities alongside Microsoft administrators to simplify management.

WHEN IS PowerShell AVAILABLE?

PowerShell has been available for use on Windows computers since 2006. Beginning with version 1.0, PowerShell

was made available to Windows XP SP2, Windows Vista, Windows Server 2003, and 2008.

PowerShell 2.0 was an upgrade to Windows XP SP3, Windows Vista SP1, and Windows Server 2003 SP2. It came integrated with Windows 7 and Windows Server 2008 R2.

PowerShell 3.0 was an upgrade to Windows 7 SP1 and Windows Server 2008 SP1 and 2008 R2 SP1. It came integrated with Windows 8 and Windows Server 2012.

PowerShell 4.0 was an upgrade to Windows 8, Windows 7 SP1, Windows Server 2008 R2 SP1, and 2012. It came integrated with Windows 8.1 and Windows Server 2012 R2.

PowerShell 5.0 is an upgrade to Windows 8.1, Windows 7 SP1, Windows Server 2008 R2 SP1, and 2012 R2. It comes integrated with Windows 10 and Windows Server 2016.

PowerShell 5.1 (.NET) is currently only available via the Windows 10 Anniversary Update.

PowerShell 7 is the most recent version available, based on the .NET Core framework. It was made available to the public on March 4, 2020, and currently supports Windows (x86/x64), Ubuntu 16.04/18.04, Debian 9/10, CentOS 7/8, RHEL 7, openSUSE 42.3, Fedora 30, macOS 10.13+, and Docker.

PowerShell 7.0.3 is the current, stable version; it was released on July 6, 2020.

WHAT ARE ALTERNATIVES TO PowerShell?

These are some of the alternatives to PowerShell.

- Linux SSH
- PuTTY
- OpenSSH
- Cygwin
- Windows command prompt
- GNOME Terminal
- Cmder
- Pash

HOW CAN I GET POWERSHELL?

PowerShell is integrated on all versions of Windows going as far back as Windows 7. It is also integrated on all

versions of Windows Server going as far back as Windows Server 2008 R2.

Previous versions of Windows can run PowerShell, though it is available as an optional update and not integrated with the operating system as is the case in more recent versions.

To install or upgrade to the 5.1 version of PowerShell on Windows computers, the latest Windows Management Framework (WMF) installer must be downloaded. The version number on the WMF installer directly matches the version of PowerShell that will be installed. Currently, WMF 5.1 exists and may be downloaded from Microsoft's website free of charge for Windows users.

To install or upgrade to the open source version of PowerShell 7 on supported systems, such as macOS and Linux, visit the PowerShell GitHub repository to obtain the appropriate package that matches the target operating system. Additionally, the latest versions of PowerShell 7 may be installed using the system's native CLI by entering the respective system's command with administrative credentials.

HOW DO I USE PowerShell?

TechRepublic has published a number of tutorials on how to get the most out of PowerShell. Check out these tips, as well as some PowerShell basics from Microsoft.

- How to customize PowerShell settings using profiles
- How to uninstall the Edge browser in Windows 10 using PowerShell
- How to use PowerShell to manage Microsoft updates on Windows
- How to install PowerShell on Ubuntu Linux
- How to use PowerShell to investigate Windows Defender's malware signature definitions database
- How to enable PowerShell Remoting via Group Policy
- 10 fundamental concepts for PowerShell scripting
- 10 PowerShell commands to make remote management easier
- 10 cool things you can do with Windows PowerShell
- Basic Windows PowerShell commands you should already know
- 10 PowerShell commands every Windows admin should know

CREDITS

Editor In Chief
Bill Detwiler

Editor In Chief, UK
Steve Ranger

Associate Managing Editors
Teena Maddox
Mary Weilage

Editor, Australia
Chris Duckett

Senior Writer
Veronica Combs

Senior Writer, UK
Owen Hughes

Editor
Melanie Wolkoff Wachsman

Staff Writer
R. Dallon Adams

Multimedia Producer
Derek Poore

Staff Reporter
Karen Roby

TechRepublic

ABOUT TECHREPUBLIC

TechRepublic is a digital publication and online community that empowers the people of business and technology. It provides analysis, tips, best practices, and case studies aimed at helping leaders make better decisions about technology.

DISCLAIMER

The information contained herein has been obtained from sources believed to be reliable. CBS Interactive Inc. disclaims all warranties as to the accuracy, completeness, or adequacy of such information. CBS Interactive Inc. shall have no liability for errors, omissions, or inadequacies in the information contained herein or for the interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice.

Cover Image: iStockphoto/gorodenkoff