

Behind the PowerShell Pipeline

*Discovering What You
Didn't Know You Needed to
Know*

by Jeff Hicks

Behind the PowerShell Pipeline

Discovering What You Didn't Know You Needed to Know

Jeff Hicks

This book is available at <https://leanpub.com/behind-the-pspipeline>

This version was published on 2025-04-26



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2025 Jeffery D. Hicks - All Rights Reserved

Also By **Jeff Hicks**

[#PS7Now](#)

[The PowerShell Practice Primer](#)

[The PowerShell Scripting and Toolmaking Book](#)

[The PowerShell Conference Book](#)

Contents

About This Book	i
A Note on Code Listings	ii
Foreword	iii
Introduction	iv
 Part I The PowerShell Consumer	 1
1 Learning How to Learn	2
2 Proper PowerShell Practice	7
3 Hosting Help	11
4 PowerShell Profile Prowess	12
5 Synchronized PowerShell Profiles	14
6 PowerShell Profile Logging	15
7 Richer PowerShell Logging	16
8 The Value of Objects	17
9 Objectifying PowerShell	18
10 Optimizing PowerShell Objects	20

CONTENTS

11 Are You My Type?	21
12 Pretty PowerShell is Better PowerShell	23
13 Special Common Variables	26
14 Buffering Up the Pipeline	28

Part II The PowerShell Creator **29**

15 Automation Decisions	30
16 What's in a Name	31
17 Planning for PowerShell Parameters	32
18 Creating Better PowerShell Output	34
19 Better Function Design	35
20 Information is Power	36
21 Clean Code is Secure Code	37
22 Data Files as Script Files	39
23 Better Code Comments	40
24 PowerShell the PowerShell Way	42
25 Creating User-Friendly Code	43
26 The Zen of PowerShell Code	44
27 Code Globally, Think Locally	45

Part III Creating PowerShell Functions . . **47**

28 Creating PowerShell Functions 101	48
---	-----------

29 Creating PowerShell Functions 201 58

30 Creating PowerShell Functions 301 59

31 Creating PowerShell Functions 401 60

Part IV The PowerShell Community 62

32 The PowerShell Learning Experience 63

33 Anyone Can Teach 65

34 Presenting PowerShell 68

TL;DR 70

About the Author 71

Release Notes 72

About This Book

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Dedication

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Acknowledgements

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

A Note on Code Listings

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Foreword

There are few people in the PowerShell community who have shaped the way we think about automation and scripting more than Jeff Hicks. I count myself among the many who have been profoundly influenced by his teaching, guidance, and example. In fact, I can say without hesitation that I wouldn't be where I am today—a Product Manager on the PowerShell team at Microsoft—without Jeff's impact on my journey.

Jeff didn't just teach me PowerShell; he taught me how to think in PowerShell. He taught me that scripting isn't just about getting something to work, but about understanding *why* it works. His ability to break down complex concepts into digestible, practical lessons helped me and countless others—go from casual users to proficient, confident PowerShell professionals.

This book, **Behind the PowerShell Pipeline**, is another testament to Jeff's expertise and generosity as an educator. It goes beyond syntax and commands, diving into the *philosophy* of PowerShell—how to craft effective, maintainable, and efficient code that works with the pipeline instead of against it. From learning how to structure functions properly to optimizing object handling, Jeff distills two decades of experience into insights that make you a better scripter, toolmaker, and automation engineer.

Jeff Hicks has given so much to the PowerShell community. His work is more than just books and blog posts—it's a foundation upon which many of us have built our careers. I'm deeply honored to write this foreword for my friend and mentor.

Jason C. Helmick

Sr Product Manager PowerShell Team Microsoft

Introduction

Hello, kind reader! This book is the culmination of a nearly two-decade PowerShell career. In that time, I realized there is more to PowerShell than stringing some commands together in a pipeline or writing a PowerShell script. For years, I focused on mechanics, syntax, and language, such as how to write a script, use a cmdlet, and use the pipeline. But I have come to realize that there is more to PowerShell than mere mechanics. There is a philosophy behind PowerShell that is just as important as the mechanics. This book is about that philosophy.

These are my ideas and principles based on nearly two decades of teaching, presenting, and writing about PowerShell. These are ideas that aren't necessarily documented in Microsoft documentation. I'm not even sure my ideas were even considered by the original PowerShell team at Microsoft. I think of my PowerShell philosophy as *emergent*¹. They arise because PowerShell exists as a complex system. But, to keep things simple and in perspective, I think of these concepts as PowerShell's "squishy" bits. The parts of PowerShell that are difficult to quantify but are just as important as the mechanics.

Learning the PowerShell language and syntax is easy. Generally, you are writing commands for someone to run. Maybe you. Maybe the help desk. You can always ask ChatGPT or CoPilot to write a PowerShell command. However, what do you need to know to create a PowerShell command that is fluid, easy to use, and easy to understand? What is the context in which someone will run your command? How can you create something that is as frictionless as possible for the user? AI-generated code may be a good starting point but should not be the end.



This book is not intended to teach you the PowerShell language, but rather *how* and *why* you should craft your PowerShell projects.

¹<https://en.wikipedia.org/wiki/Emergence>

I've long wanted to share my thoughts on these “squishy” bits and document the parts of PowerShell that people rarely consider when creating PowerShell tools.

Requirements

Unless stated otherwise, when I refer to PowerShell, I am referring to PowerShell 7. I use PowerShell 7.5 on a Windows 11 desktop, although much of the content in the book was originally developed on PowerShell 7.2 and later. Many of the book's concepts are applicable to PowerShell 5.1, but the examples are written for PowerShell 7, with no guarantee that they will work in Windows PowerShell 5.1.

I assume you are not a complete PowerShell novice and can, at minimum, run PowerShell commands from the console. I trust you know about cmdlets, the pipeline, and how to use the PowerShell help system. Ideally, you will have started scripting and maybe even written a few functions.

Other than a PowerShell 7 environment, you shouldn't need much to try the code samples yourself. I assume most readers will also use VS Code as their editor, but you can use any editor you like. However, I would advise against using the PowerShell ISE for code examples that rely on PowerShell 7 features like `PSStyle`.

Finally, my PowerShell console environment is Windows Terminal. Again, I expect most readers will be using this environment and the most current version of the `PSReadLine` module.

Book Parts

I have organized the book into four parts. The first part focuses on using PowerShell interactively from the console and command prompt. If you can't run PowerShell commands interactively, you will struggle to write scripts and functions. Part I will also offer suggestions on optimizing your PowerShell work environment.

Part II is the core of the book. The focus is on writing scripts and functions but taking the “squishy” bits into account. What unknown unknowns do you need to consider to write better scripts and functions? What are the things that you need to know to make your scripts and functions more user-friendly, secure, and effective?

I debated on how much technical content to include but decided to err on the side of education. Part III is a set of chapters that guide you through creating PowerShell functions. I know many readers probably already know how to write a PowerShell function. However, there are learning opportunities, even in what you might consider basic knowledge. The chapters in this part will guide you in the process of creating effective PowerShell functions. Hopefully, the chapters will have at least a few nuggets of new and helpful information for more experienced PowerShell users.

Finally, Part IV addresses the PowerShell community and you. One of the reasons that PowerShell has flourished is the fantastic community that has grown around it. I’m betting that more than a few readers wouldn’t be where they are today without the help of the PowerShell community. This section covers some of the ways you can give back to the community and help cultivate the next generation of PowerShell professionals.

Even though I have a plan for organizing chapters, you may find some chapters with content that spans multiple parts or doesn’t neatly fit into a specific part. Don’t let that distract you. You will also find me repeating some ideas in different chapters. When you see this, it is a hint that I consider the concept or topic important, so maybe you should too.

Code Samples

It wouldn’t be a book about PowerShell without code samples. Except for exceptionally short examples, don’t waste your time trying to retype code from the book. You can download an “Extras” ZIP file of all the code samples and listings with your book purchase from Leanpub. Open the files in Visual Studio Code or your favorite editor, and run the code samples yourself. You are welcome to use any of my code samples in your projects.

All PowerShell code samples are provided as-is and without warranty. Nothing is to be considered production-ready. All code samples and listings should be considered educational material.

Behind the PowerShell Pipeline

For the last few years, I have published a premium newsletter called [Behind the PowerShell Pipeline](https://buttondown.com/behind-the-powershell-pipeline)². I've used this newsletter to share my thoughts and examples regarding the hidden parts of PowerShell, or as I've been referring to them, the “squishy bits,” things that can make you a better scripter and toolmaker. This book is a collection of curated newsletter content from the last several years. However, this book is not merely a collection of reprints. In most chapters, I have revised or expanded the content from the original newsletter. Even if you read the original source material, I encourage you to read the chapter.

Premium newsletter subscribers have complete access to the [full archive](https://buttondown.com/behind-the-powershell-pipeline/archive/)³ of material going back to 2022. There is a lot of content not included in this book. Monthly and yearly subscriptions are available, and you can cancel anytime. I hope you'll consider subscribing and supporting my work. Thank you in advance.

²<https://buttondown.com/behind-the-powershell-pipeline>

³<https://buttondown.com/behind-the-powershell-pipeline/archive/>

Part I: The PowerShell Consumer

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

1 Learning How to Learn

I have always valued the importance of learning how to learn or how to teach yourself. This concept is much more valuable than learning a specific skill by rote memorization. When I teach a PowerShell class, before I finish, I must ensure you know how to teach yourself. First, it is impossible to teach you everything about PowerShell in a single class, nor can you memorize everything PowerShell can do. This might have been possible in PowerShell v1.0, but not today. You need to understand *how* PowerShell works and *how* to teach yourself.

As IT Pros, we are often asked to learn new tools and technologies. These tools might have a PowerShell angle, like Azure or AWS. But I want to caution you against jumping directly to the PowerShell equivalent. I think there is a better approach to learning a new technology. You can apply my ideas to just about anything new you need to learn.



Everything I am discussing in this chapter is my opinion. However, it is an opinion based on 25 years of teaching IT Pros and witnessing how people learn, especially adult learners.

Authoritative Sources

My first step is to identify authoritative sources. Who is going to teach me? This step might take some time, especially for a topic that is entirely new to you. You must research the topic and discover who or what is respected and what high-quality learning resources are. Don't simply jump to YouTube and hope to find something useful. I'm not saying you won't find helpful information on YouTube, but you need to ensure you are investing your time with a reputable

and high-quality source. It is easy to talk to a camera, but not everyone can be an experienced *teacher*.

I can't stress the quality idea enough – not only in the quality of the content but also in how it is presented. I'm not going to suggest you select one learning modality. If you learn better through video, go that route. However, I'd suggest supplementing your learning material with books at some point. This also goes the other way. You will reinforce your learning by combining modalities.

There is a reason learning resources like books or video offerings like Pluralsight cost money. Yes, content publishers are trying to earn a profit. But that means they have an incentive to provide high-quality content that can compete in the marketplace. They typically engage with respected and experienced creators, authors, and teachers. They also offer added value, such as editing and production support. These things are worth the money. If I am reading a self-published book full of grammar errors and spelling mistakes, that takes me out of learning mode. You should also assess content publishers to ensure you invest time and money into high-quality content. If it is possible, don't let price alone guide you.

You will need to identify your authoritative sources because you will use them for the learning methodology's next steps. Oh, and don't shy away from content you think might be repetitive. Repetition is a valid learning technique, and authors might present the same general idea differently. One way might click more with you. Or you might pick up a new detail. This is also why I encourage a mix of video and printed material, not to mention live events such as conferences or user groups. How the material is presented will impact how you learn it.

Terminology and Vocabulary

Learning anything new, whether a new tool like Python or a technology like Kubernetes, will expose you to strange and new terms. You don't need to learn the details of every term or new word. But you need to know enough to recognize what it is and be able to provide a rough definition. Over time, your mastery of terms and vocabulary will improve.

Learning the necessary **fundamental** terms is your first step. This is why you need to identify reputable, authoritative sources. You need to understand what you need to know to get started. Your focus should be simple understanding and recognition. For someone new to PowerShell, I would teach them terms like *cmdlet* and *pipeline*. You don't need to know how to use them, but you should be able to define them, even if you don't fully understand all parts of the definition. Deeper comprehension of terms and vocabulary will come with time, repetition, and the following steps.

Relationships and Patterns

Next, and I'm being a little abstract here, you need to be able to *identify* relationships and patterns. No matter what you are learning, the **thing** will have some sort of structure. There is an organizational structure or framework, whether you are learning C# or French. You can think of it as syntax. When you learned how to use a PowerShell cmdlet, you had to learn the syntax of parameters. Some parameters are required. Some are positional. Parameter values are typed objects. Some parameters have default values. There are many ways to pass parameter values to a cmdlet. All of this is syntax. It takes time to learn and absorb. Don't rush this step.

For example, in Kubernetes, there are a variety of technical terms, such as containers and orchestration. You need to learn the relationships between the technical concepts. Don't feel you need to know *how* the relationship works. That is the easy part since you have authoritative sources to teach you the technical details. But you need to recognize the pattern or relationship to know what to look for to **learn** it.

Walk Before You Run

I can't stress this point enough, but when learning something new, focus on the fundamentals. Learning something new is no different than constructing a house. If you don't have a solid foundation, the building will not last. Everyone

learns at a different pace, so don't compare your learning journey with that of others. Take the time you need to understand the bigger picture. Just as it is in learning a foreign language, you start with some words, learn to put them together in sentences, and, with practice, eventually hold a conversation. You will be frustrated and unsuccessful if you jump in and attempt the conversation phase without the fundamentals.

The steps I've mentioned so far are iterative. You start with basic terms and patterns, then move to intermediate-level understanding, and finally, advanced or expert level. Over time, with repeated work and exposure, you will become fluent in the new thing, and it will simply be part of you.

I am fluent in PowerShell and don't have to work as hard because I instinctively know what I want to do and how to achieve it. However, this is only possible because I have been learning, using, and teaching myself PowerShell and related technologies like .NET and C# for almost 20 years.

Have a Goal

Another recommendation when learning something new is to have a **realistic** goal. Why are you learning a new thing? What do you want to accomplish? Again, keep it simple. I remember, back in my VBScript days, encountering someone in a forum looking for help. He admitted to being an absolute beginner but sought help writing a script that would use WMI to set file permissions. That's a task even I would have wanted to avoid attempting. I would classify this as an unrealistic goal for a beginner.

Think about a task or goal that you would like to achieve. Maybe you want to be able to automate the creation of an Azure virtual machine or order dinner in Italian. Knowing *why* you are learning something helps put your learning in context and perspective. Having a goal adds *meaning* to your knowledge.

GUI First

Before I wrap this up, I want to briefly suggest a few tips for learning something new that is PowerShell-related. Let's say your company has decided to migrate to the cloud and wants to run workloads in Azure. You know you can manage Azure with PowerShell, so your initial thought is to load the Azure modules and start banging away.

This step would be OK **if you already know Azure**. Without a doubt, PowerShell can make it easier to manage and automate a technology like Azure. But if you don't understand how Azure works, you are doubling up on the learning pain. You are most likely trying to understand Azure **and** make sense of the Azure cmdlets.

In this example, I recommend following the steps I suggested earlier and learning Azure first. Take advantage of the graphical tools to use the new technology manually. A graphical interface can often help you build internal patterns and relationship maps. In my Azure scenario, you should learn the graphical and manual steps to create a virtual machine. Try to visualize what Azure is doing with each button click. Document the workflow.

PowerShell It

Once you understand how the technology works manually, you can turn to the PowerShell cmdlets. Since you know what steps you manually perform to achieve a task in Azure, you can search the available cmdlets for the tools that will achieve the same result in PowerShell. In reality, this step is something new to learn, so apply the steps I've been talking about: identify resources, learn terms, build patterns, etc.

Learning is an iterative process that never ends.

I love the thrill of learning something new and applying it. The last step is critical. If you don't use what you learn, you won't retain the knowledge, and you'll never truly know it. We learn by doing, which is especially true in IT.

2 Proper PowerShell Practice

I'm assuming most of you have been using PowerShell for a while, and most likely consider yourselves adept or at least capable PowerShell scripters. You've achieved a reasonable degree of mastery over the PowerShell language. This chapter focuses on *where* you apply your PowerShell skills and *how*.

Location, Location, Location

Much like the real estate adage, **where** you apply your PowerShell skills is just as crucial as the commands you write. I believe there is a difference in how you want to use PowerShell. I don't feel you write the same PowerShell expression in all situations.

When using PowerShell **interactively** at a console prompt, the emphasis should be on speed and efficiency. As the name implies, PowerShell is as much a "shell" as anything else. You are typing commands to achieve a specific result. In these situations, I'm always telling my students to get in the habit of using tab-completion and PSReadline to write out the desired PowerShell expression quickly.

The console prompt is also the location where the use of PowerShell aliases is proper and effective. It is simple to build PowerShell muscle memory to write an expression like this quickly:

```
gsv | ? status -eq running | ogv -t 'Running
```

You know what you wrote and could quickly type out the command. There is nothing wrong with using aliases in this context. I constantly use shortcuts and aliases in PowerShell when running commands interactively. Commands you type at a PowerShell prompt are essentially ephemeral.

Where this becomes a potential issue is with PowerShell scripting. The PowerShell script commands are no different from what you can run interactively. PowerShell won't complain if you put the line above in a script file. However, I firmly believe you need to take a different perspective. When you put code in a PowerShell script file, you are committing. You are creating a permanent artifact that can be reused indefinitely.

The goal isn't how quickly you can write the PowerShell expression. A good scripting editor like VSCode can assist with that. Instead, the goal should be clarity. PowerShell can be wordy, but it is meaningful. There is a reason cmdlet names follow a Verb-Noun naming convention, and parameters are generally consistent. These features make it easier to look at a line of PowerShell code and understand what it will do.

I always tell people, "Write your PowerShell scripts for the next person. It might be you!" In other words, you may write a "quick and dirty" PowerShell script only to edit it nine months later, and you need to take some time to recall how the code works. I have had more than one IT pro admit to this very problem. This confusion is why there is a community recommendation to avoid PowerShell aliases in your script files. Instead, take the time to write an expression like this:

```
Get-Service | Where-Object {$_.Status -eq 'running'} |  
Select-Object -property Name,Status,StartType,DisplayName |  
Out-GridView -title "Running"
```

I recommend using full cmdlet names and parameter names. However, I'm cheating a bit here with `Where-Object`. There is no confusion or misunderstanding about what this expression will do. Even if you knew very little PowerShell, you could make a solid, educated guess about the result.

There are a few other reasons for the full-name recommendation. First, your code may be executed on non-Windows machines because PowerShell is cross-platform. For example, you might not get the expected result on Linux if you run a PowerShell script that uses the alias `Sort` for `Sort-Object` because there is no `sort` alias. Instead, PowerShell will run the native `sort` command.

Another reason for using full parameter names, although I'll admit the risk is probably low, is that commands might change. You might write a PowerShell expression using positional parameters. However, in a future release, Microsoft may revise the command and insert a new positional parameter, or the positional parameter will become a named parameter. These changes will break your script unless you use complete parameter names.

One other “location” to consider relates to PowerShell scheduled jobs. This is a bit of a hybrid situation. Typically, you are creating a PowerShell scheduled job interactively. On the one hand, you want to get it done quickly. But, there is also an element of persistence. You may return to your scheduled job in a year and look at the command. Will you understand what you set up? Or perhaps your organization uses a centralized “jump box” to run your scheduled jobs. In this situation, I would prioritize clarity. On a side note, I'd recommend defining PowerShell scheduled jobs as script files, not script blocks. If you need to update the PowerShell commands, you can edit the script file and leave the scheduled job alone. Of course, your script file should follow my recommendations.

Who Are You Scripting For?

Ultimately, all of this comes down to a simple question, “Who is your PowerShell for?” If you are using PowerShell in an interactive console session, you can follow one set of guidelines. If you are writing a PowerShell module, there are different community-accepted standards. That is one of the reasons we have tools like the PowerShell Script Analyzer.

I will take this a step further with PowerShell functions and tools. When you write a PowerShell function, you must consider who will use it and their expectations. Maybe you are writing the command for yourself. Or perhaps it is for the intern on your team. Will the person executing your command need to enter many parameters? Parameters are a great way to write flexible and reusable code. Can your command be simplified by setting default values? Do you need a parameter alias that is more meaningful to the user?

Will the user typically pipe the output of a PowerShell command to *your* command, such as importing data from a CSV file? Will they pipe output from your command so another PowerShell command? In other words, how do you think someone will use your command in the PowerShell ecosystem?

Learning the mechanics of PowerShell scripting is the easy part. Deciding what to script is where the art comes in.

3 Hosting Help

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Writing Output vs. Host

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

4 PowerShell Profile Prowess

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Understanding PowerShell Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Profile Processing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Profiles and Scheduled Jobs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Remote PowerShell Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

RemoteProfile.ps1

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

New-PSConnection

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Managing Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Go Configure

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Get Profile Status

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Cascading Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Supporting Cross-Version Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Profile Script Backup

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

5 Synchronized PowerShell Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Central Store

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Linking CurrentUser Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Linking AllUser Profiles

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Caveats

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

6 PowerShell Profile Logging

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

LogPSStart

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Profile Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Logging Profile Execution

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

7 Richer PowerShell Logging

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Why Are You Logging?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Objectify It

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Accessing the Function

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Object Data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Managing the Logged Data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

8 The Value of Objects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Objects Are Easy

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

You Control the Object

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Functions Write Objects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

9 Objectifying PowerShell

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PSBase

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PSAdapted

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PSExtended

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PSObject

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Custom Objects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Select-Object Alternative

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

ConvertTo-PSClass

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

10 Optimizing PowerShell Objects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Being Pretty Matters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Abstraction Layers

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Reformatting the Object

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Optimize-Object

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

11 Are You My Type?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Meet the ETS

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Add-Member

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

New-ManagedComputer

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Update-TypeData

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Defining New TypeData Members

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Defining a ScriptMethod

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Property Sets

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

12 Pretty PowerShell is Better PowerShell

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

ANSI Escape Plan

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Applying ANSI Sequences

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

ANSI Discovery

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Simple Examples

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

PSStyle

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

PSStyle Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

PSStyle and Host Private Data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

PSStyle in Action

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

PSStyle Foreground/Background

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Scripting Integration

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

13 Special Common Variables

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

OutVariable

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Appending

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Pipeline Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Pipeline Processing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Pipeline Execution

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PipelineVariable

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Combining Common Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Pipeline Trade-Offs

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

14 Buffering Up the Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

OutBuffer

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Buffering with Lists

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Part II: The PowerShell Creator

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

15 Automation Decisions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Is It Worth the Investment?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Can It Be Scripted?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Are There Better Alternatives?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

PowerShell Isn't Always the Answer

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

You Decide

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

16 What's in a Name

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Consistency

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Noun Naming

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameter Naming

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

The Name Name

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

17 Planning for PowerShell Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Why Parameters?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

What's in a Name?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Pipeline Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Parameter Aliases

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Are You My Type?

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Parameter Attributes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Setting Defaults

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Validation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Custom Validation Error Messages

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Putting It All Together

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

18 Creating Better PowerShell Output

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Return vs. Write

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Write Objects to the Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Structure Matters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

19 Better Function Design

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Don't Mix Your Output

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Write-Information

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Complex Objects

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Write a Single Object Type

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Simple and Specific

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

20 Information is Power

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Write-Information and Write-Host

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Doing More with Information

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Stream Redirection

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Logging Information

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

21 Clean Code is Secure Code

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Data is Not Code

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Credentials

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Setting Defaults

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PSDefaultParameterValues

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

SecretManagement

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Configuration Data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

22 Data Files as Script Files

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Data Strings

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Data Script Blocks

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Using External Files

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

One Command to Run Them All

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

23 Better Code Comments

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

PowerShell is Self-Documenting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Get Verbose

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Variable Names

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Errors and Warnings

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Verbose Progress

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Documentation Overload

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Use Regions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Cozy Comments

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Revisionist History

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

24 PowerShell the PowerShell Way

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Think Objects, Not Text

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Better Pipeline Output

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Switch to Switch

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Think Local, Script Remotely

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

25 Creating User-Friendly Code

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Don't Force Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Don't Force the User to Convert

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Anticipate Usage

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

26 The Zen of PowerShell Code

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Flowing Code

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Write Code Once

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Comment Concisely

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Code Layout and Formatting

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Separate Data from Code

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

27 Code Globally, Think Locally

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

String Data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Revise the Function

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Import-LocalizedData

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Using Culture

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Localization Limitation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Building String Data

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Revising the Function

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Module Localization

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Part III: Creating PowerShell Functions

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

28 Creating PowerShell Functions 101

I write about functions and the scripting process all the time, but new people are always coming into PowerShell, so let's spend some time reviewing how to create a proper and effective PowerShell function. Even if you think you know how to write a function, repetition is always good when it comes to learning. So let's look anew at PowerShell functions.

I will begin with some general guidelines:

- A PowerShell function is a unit of code execution.
- It is a single-purpose command.
- Parameters can customize their behavior.
- It writes one type of object to the pipeline.

At its core, a PowerShell function is nothing more than a named script block. This makes it easier to invoke. The function's code is generally the same code you would use interactively at a PowerShell prompt. However, as you'll learn, since we're taking the time to commit the commands to a file, we often take advantage of PowerShell's scripting language.

Let's start with a simple task.

```
PS C:\> $os = Get-CimInstance -ClassName Win32_OperatingSystem
PS C:\> New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)
```

```
Days           : 6
Hours          : 4
Minutes        : 26
Seconds        : 25
Milliseconds    : 155
Ticks          : 5343851556102
TotalDays      : 6.18501337511806
TotalHours     : 148.440321002833
TotalMinutes   : 8906.41926017
TotalSeconds   : 534385.1556102
TotalMilliseconds : 534385155.6102
```

If this is a task we want to do repeatedly, we can put these commands into a simple function.

```
function Get-Uptime {
    $os = Get-CimInstance -ClassName Win32_OperatingSystem
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)
}
```

You can type this in the console, or put the code into a .ps1 file. Scoping requires that you dot-source the script file.

```
PS C:\> . c:\scripts\getuptime.ps1
```



I know that PowerShell 7 has a similar `Get-Uptime` command. Don't get distracted. Instead, focus on the *process* of creating a function.

Now, I can run the command anytime in my current PowerShell session.

```
PS C:\> Get-Uptime
```

```
Days           : 6
Hours          : 4
Minutes        : 29
Seconds        : 33
Milliseconds    : 961
Ticks          : 5345739616066
TotalDays      : 6.18719862970602
TotalHours     : 148.492767112944
TotalMinutes   : 8909.56602677667
TotalSeconds   : 534573.9616066
TotalMilliseconds : 534573961.6066
```

The function does one thing and writes one type of object to the pipeline. If you always wanted to have this command, put the dot-sourcing command in your PowerShell profile script.

Before we get any further, let's explore function names.

Naming Standards

The accepted best practice for naming your function is to follow the Verb-Noun naming convention. There's nothing preventing me from using code like this:

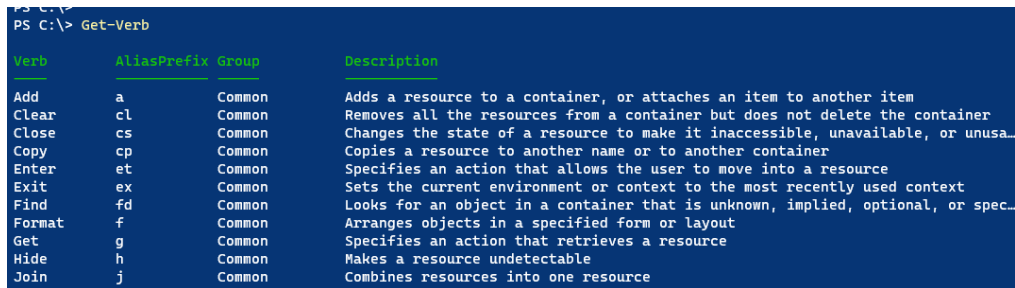
```
function gup {
    $os = Get-CimInstance -ClassName Win32_OperatingSystem
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)
}
```

But this function is going to be difficult to discover. Any PowerShell function exposed to the user, something the user can run interactively, should follow the Verb-Noun naming convention.



*When you get to writing modules where you might have helper functions, then using a non-standard name is permissible, if not preferable. A helper function is a unit of code called from another function. It is **not** exposed to the user. In these situations, I intentionally use a non-standard name so that I can identify bugs where the private functions are being made public.*

The verb should be something you find in the output from `Get-Verb`. I like the format introduced with PowerShell 7.



```
PS C:\> Get-Verb
```

Verb	AliasPrefix	Group	Description
Add	a	Common	Adds a resource to a container, or attaches an item to another item
Clear	cl	Common	Removes all the resources from a container but does not delete the container
Close	cs	Common	Changes the state of a resource to make it inaccessible, unavailable, or unusable
Copy	cp	Common	Copies a resource to another name or to another container
Enter	et	Common	Specifies an action that allows the user to move into a resource
Exit	ex	Common	Sets the current environment or context to the most recently used context
Find	fd	Common	Looks for an object in a container that is unknown, implied, optional, or specified
Format	f	Common	Arranges objects in a specified form or layout
Get	g	Common	Specifies an action that retrieves a resource
Hide	h	Common	Makes a resource undetectable
Join	j	Common	Combines resources into one resource

Figure 28.1. Get-Verb

There should be something in the list that will fit your function. Remember, your function is doing one thing. How would you describe your function's purpose in a single sentence?

Get the computer's uptime.

In most cases, the verb will be self-evident.

The noun portion should be a singular version of the “thing” you are working with. In my example, I'm getting uptime. However, you should be as specific and unique as possible. I have options. I could use the noun “runtime.” Or maybe your organization has a term that reflects how long a computer has been running.

The noun portion of the name is where you can avoid naming conflicts. `Get-Uptime` is rather generic, and could conflict with a command from some future module. The accepted recommendation is to use a prefix with the noun. The prefix could be your initials. It could be a set of initials that represents your company. When you create modules, the prefix might tie all the module commands together.



Be careful with the potentially popular prefixes My and PS.

In reviewing “Uptime” as a noun, I realize this could be more specific. A function called `Get-Uptime` could mean getting how long a process or service has been running. I’m going to revise my basic function.

```
function Get-ComputerUptime {  
    $os = Get-CimInstance -ClassName Win32_OperatingSystem  
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)  
}
```

Now, there’s no mystery about the function’s purpose.

Customize with Parameters

Another important scripting guideline is that you want your function to be flexible and reusable. You should avoid making hard-coded or static references in your code to items like these:

- Domain name
- An organizational unit name
- A path
- A user name

And while it should be obvious, let me go on the record and state your code should **never** include plain-text passwords, API tokens, keys, or other secrets.

Once your function is written, you shouldn’t have to edit it to change its behavior. That’s the role of a parameter.

You should look at your code and consider what parts are variable. In other words, what parts could a user want to change? These items will become your parameters. In my basic example, all I can do with the current version is get the uptime for the local computer. That seems a bit silly when I can easily get it for a remote computer. I’ll add a parameter to the basic function.

```
function Get-ComputerUptime {  
    Param($ComputerName)  
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `   
    -ComputerName $ComputerName  
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)  
}
```

The parameter name becomes the variable you use in your code. As with function names, there are some parameter naming guidelines.

- It should contain alphabetical characters only—no numbers, spaces, or special characters. You also don't need to use Hungarian notation like `$strComputerName`.
- If there is an existing parameter from another command that fits your need, use it. Don't reinvent the wheel or try to get clever.
- I recommend using proper casing, including camel case, with your parameter and being consistent throughout your code. It will make your code easier to read.

By the way, you will come across code written like this.

```
function Get-ComputerUptime ($ComputerName) {  
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `   
    -ComputerName $ComputerName  
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)  
}
```

This version of the function will run the same way. But I don't recommend it. This is a style drawn from C# and the .NET Framework. The challenge is that, eventually, you will do much more with parameters, such as adding parameter aliases. Defining parameters *inside* the function will make it easier to write and neater.

I suggest parameterizing anything that could be customized. If you think the user will use a value of 'foo' nine out of ten times, then make that the default.

```
function Get-ComputerUptime {  
    Param($ComputerName = $env:computername)  
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `   
    -ComputerName $ComputerName  
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)  
}
```

Rich Output

The last part of the basic function I want to cover is output. As I mentioned earlier, your function should only write one **type** of object to the pipeline. As my function now stands, all I get is a TimeSpan object. If I process a list of computers in a ForEach-Object loop, I can't tell what uptime goes with which computer.

You **do not** want to write a function like this:

```
function Get-ComputerUptime {  
    Param($ComputerName = $env:computername)  
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `   
    -ComputerName $ComputerName  
    $Computername  
    New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)  
}
```

Yes, it works.

```
PS C:\> Get-ComputerUptime -ComputerName dom1
dom1
```

```
Days           : 31
Hours          : 5
Minutes        : 28
Seconds        : 43
Milliseconds    : 66
Ticks          : 26981230668450
TotalDays      : 31.2282762366319
TotalHours     : 749.478629679167
TotalMinutes   : 44968.71778075
TotalSeconds   : 2698123.066845
TotalMilliseconds : 2698123066.845
```

But if I try to sort the output from multiple computers, I'll get odd results since the function is writing a string **and** TimeSpan objects to the pipeline.

The better approach for your function is to write a single, rich object to the pipeline. You might use `Select-Object` in a pipelined expression. Or something like this:

```
function Get-ComputerUptime {
    Param($ComputerName = $env:computername)
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `
        -ComputerName $ComputerName
    $up = New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)
    [PSCustomObject]@{
        ComputerName = $os.CSName
        LastBoot      = $os.LastBootUpTime
        Uptime        = $up
    }
}
```

Include as much information as you think the user might want to see. Don't worry about making it pretty. That's where custom format files come into the picture, which is outside the scope of this chapter.

You should never include formatting commands in your function. Here's another **bad** example.


```
function Get-ComputerUptime {
    Param($ComputerName = $env:computername)
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `
        -ComputerName $ComputerName
    $up = New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)
    $out = [PSCustomObject]@{
        ComputerName = $os.CSName
        LastBoot      = $os.LastBootUpTime
        Uptime        = $up
    }
    #DON'T DO THIS
    $out | Format-Table
}
```

All I can ever do with this function is look at the results in the console, or pipe to `Out-File`. There is no way to sort, filter, convert, or export the output. Don't assume you can think of every possible way a user might want to use your function. Write a rich object to the pipeline that users can consume however they choose.

This also means that you **do not** use `Write-Host` to output results.

```
function Get-ComputerUptime {
    Param($ComputerName = $env:computername)
    $os = Get-CimInstance -ClassName Win32_OperatingSystem `
        -ComputerName $ComputerName
    $up = New-TimeSpan -Start $os.LastBootUpTime -End (Get-Date)
    $out = [PSCustomObject]@{
        ComputerName = $os.CSName
        LastBoot      = $os.LastBootUpTime
        Uptime        = $up
    }
    #NO!!!
    Write-Host $out
}
```

Output from `Write-Host` cannot be consumed by other PowerShell commands.

All you need to do is use PowerShell code that writes to the pipeline. Do not be tempted to use the return keyword. This is a developer-oriented concept

that doesn't usually apply to PowerShell scripting. When PowerShell encounters `return`, it writes the return value to the pipeline and then **terminates the pipeline**. If your command has more output, you won't get it. While you *can* use `return`, for me, this is a conceptual issue; don't think of a PowerShell function *returning* a value. Think of it as *writing* objects to the pipeline. This mental model will help you write better PowerShell functions.

29 Creating PowerShell Functions 201

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Write Once

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

CmdletBinding

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameter Type

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Command Aliases

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

OutputType

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

30 Creating PowerShell Functions 301

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Defining Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameter Aliases

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameter Validation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Pipeline Processing

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

31 Creating PowerShell Functions 401

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Array Values

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Process Individually

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Error Handling

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Best of Both Worlds

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameter Sets

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Parameter Set Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Multiple Parameter Combinations

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Get-ComputerUptime

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Part IV: The PowerShell Community

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

32 The PowerShell Learning Experience

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Focus on Concepts

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Finding a Learning Resource

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Read, Do, Refine, Repeat

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Teaching Not Training

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

Learning Never Ends

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

33 Anyone Can Teach

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Teaching vs. Training

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Yes, You Can Teach

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Avoid the Black Hole

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Command Shell First

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Build Slowly

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Essential Cmdlets

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

How to Teach

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Patience is a Virtue

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Visualization

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Be Open to Learning

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Just Do It

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Read Everything

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Practice PowerShell Daily

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Learning Resources

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

34 Presenting PowerShell

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Start Small

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Proposals

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Presentation Materials

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

The Presentation

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Demos

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

Get Ready for the Next One

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

TL;DR

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

About the Author



Jeff Hicks is a veteran IT Professional with 35 years of experience, much of it spent as an IT infrastructure consultant specializing in Microsoft server technologies with an emphasis on automation and efficiency. He is a multi-year recipient of the Microsoft MVP Award for his PowerShell-related contributions. He is an independent author, teacher, and consultant. Jeff has written, taught, and presented on PowerShell and the benefits of automation to IT professionals worldwide for over 30 years. He has authored, and co-authored several books, writes for numerous online sites, is a [Pluralsight](https://pluralsight.com/profile/author/jeff-hicks)¹ author, and is a frequent speaker at technology conferences and user groups.

You can keep up with Jeff at <https://jdhitsolutions.github.io>

¹<https://app.pluralsight.com/profile/author/jeff-hicks>

Release Notes

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

25 April 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

21 March 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

06 Mar 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

28 Feb 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

14 Feb 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-ppipeline>.

8 Feb 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.

5 Feb 2025

This content is not available in the sample book. The book can be purchased on Leanpub at <https://leanpub.com/behind-the-pspipeline>.