

Learn **WINDOWS**
POWERSHELL
IN A MONTH OF LUNCHESES

THIRD EDITION

MONDAY	TUESDAY	WEDNESDAY	THURSDAY
	1 Integrated Scripting Environment <i>✓ very good</i>	2 Updatable Help <i>✓ great!</i>	3 Anatomy of a Command <i>✓ done</i>
7 Support External Commands <i>✓</i>	8 Dealing with Errors <i>✓ good to know</i>	9 Working with Providers	10 How the File System is Organized
14 Working with other Providers	15 Playing with a New Module	16 Objects Until the Very End	11 Navigating the File System
21 Another Out Grid Views	22 Schedules Jobs Other Security Holes	23 More Tricks with Double Quotes	17 Pipeline Input by Value
	29 Parameter, Alias and RegEx Syntax	24 Implicit Remoting Parameterizing	18 Parenthetical Commands
	30 Final Exam		25 Parameterizing Meters One Script

DON JONES
JEFFERY D. HICKS

 MANNING

Learn Windows PowerShell in a Month of Lunches
Third Edition

*Learn Windows PowerShell
in a Month of Lunches*

THIRD EDITION

DON JONES
JEFFERY HICKS



MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact


Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

©2017 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

© Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

 Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editor: Helen Stergius
Project editor: Janet Vail
Copyeditor: Sharon Wilkey
Proofreader: Alyson Brener
Technical proofreader: James Berkenbile
Typesetter: Dottie Marsico
Cover designer: Leslie Haines

ISBN 9781617294167

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 – EBM – 21 20 19 18 17 16

brief contents

- 1 ■ Before you begin 1
- 2 ■ Meet PowerShell 9
- 3 ■ Using the help system 20
- 4 ■ Running commands 37
- 5 ■ Working with providers 51
- 6 ■ The pipeline: connecting commands 63
- 7 ■ Adding commands 76
- 8 ■ Objects: data by another name 89
- 9 ■ The pipeline, deeper 101
- 10 ■ Formatting—and why it’s done on the right 123
- 11 ■ Filtering and comparisons 139
- 12 ■ A practical interlude 148
- 13 ■ Remote control: one-to-one, and one-to-many 153
- 14 ■ Using Windows Management Instrumentation and CIM 171
- 15 ■ Multitasking with background jobs 185
- 16 ■ Working with many objects, one at a time 200
- 17 ■ Security alert! 216
- 18 ■ Variables: a place to store your stuff 228
- 19 ■ Input and output 244
- 20 ■ Sessions: remote control with less work 253
- 21 ■ You call this scripting? 263
- 22 ■ Improving your parameterized script 276

- 23 ■ Advanced remoting configuration 286
- 24 ■ Using regular expressions to parse text files 296
- 25 ■ Additional random tips, tricks, and techniques 303
- 26 ■ Using someone else's script 316
- 27 ■ Never the end 325
- 28 ■ PowerShell cheat sheet 328

contents

preface xvii
acknowledgments xix
about this book xx
about the authors xxii

1 *Before you begin* 1

- 1.1 Why you can't afford to ignore PowerShell 1
 - Life without PowerShell* 2 ▪ *Life with PowerShell* 2
- 1.2 And now, it's just "PowerShell" 3
- 1.3 Is this book for you? 3
- 1.4 How to use this book 4
 - The main chapters* 4 ▪ *Hands-on labs* 5 ▪ *Code samples* 5 ▪ *Supplementary materials* 5 ▪ *Further exploration* 5 ▪ *Above and beyond* 5
- 1.5 Setting up your lab environment 6
- 1.6 Installing Windows PowerShell 7
- 1.7 Contacting us 8
- 1.8 Being immediately effective with PowerShell 8

2 *Meet PowerShell* 9

- 2.1 Choose your weapon 9
 - The console window* 11 ▪ *The Integrated Scripting Environment* 13

- 2.2 It's typing class all over again 15
- 2.3 Common points of confusion 17
- 2.4 What version is this? 17
- 2.5 Lab 18

3 *Using the help system* 20

- 3.1 The help system: how you discover commands 20
- 3.2 Updatable help 22
- 3.3 Asking for help 23
- 3.4 Using help to find commands 24
- 3.5 Interpreting the help 26
 - Parameter sets and common parameters* 26
 - Optional and mandatory parameters* 28
 - Positional parameters* 28
 - Parameter values* 30
 - Finding command examples* 33
- 3.6 Accessing “about” topics 33
- 3.7 Accessing online help 34
- 3.8 Lab 34
- 3.9 Lab answers 36

4 *Running commands* 37

- 4.1 Not scripting, but running commands 37
- 4.2 The anatomy of a command 38
- 4.3 The cmdlet naming convention 39
- 4.4 Aliases: nicknames for commands 40
- 4.5 Taking shortcuts 42
 - Truncating parameter names* 42
 - Using parameter name aliases* 42
 - Using positional parameters* 42
- 4.6 Cheating a bit: Show-Command 44
- 4.7 Support for external commands 44
- 4.8 Dealing with errors 48
- 4.9 Common points of confusion 49
 - Typing cmdlet names* 49
 - Typing parameters* 49
- 4.10 Lab 50

5 *Working with providers* 51

- 5.1 What are providers? 51

- 5.2 Understanding how the filesystem is organized 53
- 5.3 Understanding how the filesystem is like other data stores 55
- 5.4 Navigating the filesystem 55
- 5.5 Using wildcards and literal paths 57
- 5.6 Working with other providers 58
- 5.7 Lab 61
- 5.8 Further exploration 62
- 5.9 Lab answers 62

6 *The pipeline: connecting commands* 63

- 6.1 Connecting one command to another: less work for you 63
- 6.2 Exporting to a CSV or an XML file 64
 - Exporting to CSV* 65
 - Exporting to XML* 66
 - Comparing files* 67
- 6.3 Piping to a file or a printer 69
- 6.4 Converting to HTML 70
- 6.5 Using cmdlets that modify the system: killing processes and stopping services 71
- 6.6 Common points of confusion 72
- 6.7 Lab 74
- 6.8 Lab answers 75

7 *Adding commands* 76

- 7.1 How one shell can do everything 76
- 7.2 About product-specific “management shells” 77
- 7.3 Extensions: finding and adding snap-ins 78
- 7.4 Extensions: finding and adding modules 80
- 7.5 Command conflicts and removing extensions 82
- 7.6 On non-Windows operating systems 83
- 7.7 Playing with a new module 83
- 7.8 Profile scripts: preloading extensions when the shell starts 85
- 7.9 Getting modules from the internet 86
- 7.10 Common points of confusion 87

- 7.11 Lab 87
- 7.12 Lab answers 88

8 *Objects: data by another name* 89

- 8.1 What are objects? 89
- 8.2 Understanding why PowerShell uses objects 90
- 8.3 Discovering objects: Get-Member 92
- 8.4 Using object attributes, or properties 94
- 8.5 Using object actions, or methods 94
- 8.6 Sorting objects 95
- 8.7 Selecting the properties you want 96
- 8.8 Objects until the end 97
- 8.9 Common points of confusion 99
- 8.10 Lab 99
- 8.11 Lab answers 100

9 *The pipeline, deeper* 101

- 9.1 The pipeline: enabling power with less typing 101
- 9.2 How PowerShell passes data down the pipeline 101
- 9.3 Plan A: pipeline input ByValue 102
- 9.4 Plan B: pipeline input ByPropertyName 106
- 9.5 When things don't line up: custom properties 111
- 9.6 Parenthetical commands 114
- 9.7 Extracting the value from a single property 115
- 9.8 Lab 121
- 9.9 Further exploration 122
- 9.10 Lab answers 122

10 *Formatting—and why it's done on the right* 123

- 10.1 Formatting: making what you see prettier 123
- 10.2 Working with the default formatting 124
- 10.3 Formatting tables 127
- 10.4 Formatting lists 128
- 10.5 Formatting wide lists 129
- 10.6 Creating custom columns and list entries 130

- 10.7 Going out: to a file, a printer, or the host 133
- 10.8 Another out: GridViews 133
- 10.9 Common points of confusion 133
 - Always format right 133* ▪ *One type of object at a time, please 135*
- 10.10 Lab 137
- 10.11 Further exploration 137
- 10.12 Lab answers 138

11 *Filtering and comparisons 139*

- 11.1 Making the shell give you just what you need 139
- 11.2 Filtering left 140
- 11.3 Using comparison operators 140
- 11.4 Filtering objects out of the pipeline 142
- 11.5 Using the iterative command-line model 144
- 11.6 Common points of confusion 145
 - Filter left, please 145* ▪ *When \$_ is allowed 146*
- 11.7 Lab 146
- 11.8 Further exploration 147
- 11.9 Lab answers 147

12 *A practical interlude 148*

- 12.1 Defining the task 148
- 12.2 Finding the commands 148
- 12.3 Learning to use the commands 150
- 12.4 Tips for teaching yourself 151
- 12.5 Lab 152
- 12.6 Lab answer 152

13 *Remote control: one-to-one, and one-to-many 153*

- 13.1 The idea behind remote PowerShell 154
- 13.2 WinRM overview 155
- 13.3 Using Enter-PSSession and Exit-PSSession for one-to-one remoting 159
- 13.4 Using Invoke-Command for one-to-many remoting 161

- 13.5 Differences between remote and local commands 163
 - Invoke-Command vs. -computerName* 164
 - Local vs. remote processing* 165
 - Serialized objects* 166
- 13.6 But wait, there's more 167
- 13.7 Remote options 168
- 13.8 Common points of confusion 168
- 13.9 Lab 169
- 13.10 Further exploration 170
- 13.11 Lab answers 170

14 *Using Windows Management Instrumentation and CIM* 171

- 14.1 WMI essentials 172
- 14.2 The bad news about WMI 173
- 14.3 Exploring WMI 174
- 14.4 Choose your weapon: WMI or CIM 177
- 14.5 Using Get-WmiObject 178
- 14.6 Using Get-CimInstance 182
- 14.7 WMI documentation 182
- 14.8 Common points of confusion 182
- 14.9 Lab 183
- 14.10 Further exploration 184
- 14.11 Lab answers 184

15 *Multitasking with background jobs* 185

- 15.1 Making PowerShell do multiple things at the same time 185
- 15.2 Synchronous vs. asynchronous 186
- 15.3 Creating a local job 187
- 15.4 WMI, as a job 188
- 15.5 Remoting, as a job 189
- 15.6 Getting job results 189
- 15.7 Working with child jobs 192
- 15.8 Commands for managing jobs 194
- 15.9 Scheduled jobs 196
- 15.10 Common points of confusion 197

- 15.11 Lab 198
- 15.12 Lab answers 199

16 *Working with many objects, one at a time* 200

- 16.1 Automation for mass management 200
- 16.2 The preferred way: “batch” cmdlets 201
- 16.3 The CIM/WMI way: invoking methods 202
- 16.4 The backup plan: enumerating objects 206
- 16.5 Common points of confusion 211
 - Which way is the right way?* 211
 - *WMI methods vs. cmdlets* 212
 - *Method documentation* 213
 - *ForEach-Object confusion* 213
- 16.6 Lab 214
- 16.7 Lab answers 214

17 *Security alert!* 216

- 17.1 Keeping the shell secure 216
- 17.2 Windows PowerShell security goals 217
- 17.3 Execution policy and code signing 218
 - Execution policy settings* 218
 - *Digital code signing* 222
- 17.4 Other security measures 225
- 17.5 Other security holes? 225
- 17.6 Security recommendations 226
- 17.7 Lab 227

18 *Variables: a place to store your stuff* 228

- 18.1 Introduction to variables 228
- 18.2 Storing values in variables 229
- 18.3 Using variables: fun tricks with quotes 231
- 18.4 Storing many objects in a variable 233
 - Working with single objects in a variable* 234
 - *Working with multiple objects in a variable* 235
 - *Other ways to work with multiple objects* 236
 - *Unrolling properties and methods in PowerShell v3* 237
- 18.5 More tricks with double quotes 237
- 18.6 Declaring a variable’s type 239

- 18.7 Commands for working with variables 241
- 18.8 Variable best practices 242
- 18.9 Common points of confusion 242
- 18.10 Lab 242
- 18.11 Further exploration 243
- 18.12 Lab answer 243

19 *Input and output* 244

- 19.1 Prompting for, and displaying, information 244
- 19.2 Read-Host 245
- 19.3 Write-Host 248
- 19.4 Write-Output 249
- 19.5 Other ways to write 251
- 19.6 Lab 252
- 19.7 Further exploration 252
- 19.8 Lab answers 252

20 *Sessions: remote control with less work* 253

- 20.1 Making PowerShell remoting a bit easier 253
- 20.2 Creating and using reusable sessions 254
- 20.3 Using sessions with Enter-PSSession 255
- 20.4 Using sessions with Invoke-Command 257
- 20.5 Implicit remoting: importing a session 258
- 20.6 Using disconnected sessions 260
- 20.7 Lab 261
- 20.8 Further exploration 262
- 20.9 Lab answers 262

21 *You call this scripting?* 263

- 21.1 Not programming, more like batch files 263
- 21.2 Making commands repeatable 264
- 21.3 Parameterizing commands 265
- 21.4 Creating a parameterized script 267
- 21.5 Documenting your script 268
- 21.6 One script, one pipeline 270

- 21.7 A quick look at scope 273
- 21.8 Lab 274
- 21.9 Lab answer 275

22 *Improving your parameterized script* 276

- 22.1 Starting point 276
- 22.2 Getting PowerShell to do the hard work 277
- 22.3 Making parameters mandatory 278
- 22.4 Adding parameter aliases 280
- 22.5 Validating parameter input 281
- 22.6 Adding the warm and fuzzies with verbose output 282
- 22.7 Lab 284
- 22.8 Lab answer 284

23 *Advanced remoting configuration* 286

- 23.1 Using other endpoints 286
- 23.2 Creating custom endpoints 287
 - Creating the session configuration* 288
 - Registering the session* 289
- 23.3 Enabling multihop remoting 291
- 23.4 Digging deeper into remoting authentication 292
 - Defaults for mutual authentication* 292
 - Mutual authentication via SSL* 293
 - Mutual authentication via TrustedHosts* 293
- 23.5 Lab 294
- 23.6 Lab answer 295

24 *Using regular expressions to parse text files* 296

- 24.1 The purpose of regular expressions 297
- 24.2 A regex syntax primer 297
- 24.3 Using regex with -Match 299
- 24.4 Using regex with Select-String 299
- 24.5 Lab 301
- 24.6 Further exploration 301
- 24.7 Lab answers 302

- 25 *Additional random tips, tricks, and techniques* 303**
- 25.1 Profiles, prompts, and colors: customizing the shell 303
 - PowerShell profiles* 303
 - Customizing the prompt* 305
 - Tweaking colors* 306
 - 25.2 Operators: -as, -is, -replace, -join, -split, -in, -contains 307
 - as and -is* 307
 - replace* 308
 - join and -split* 308
 - contains and -in* 309
 - 25.3 String manipulation 310
 - 25.4 Date manipulation 311
 - 25.5 Dealing with WMI dates 312
 - 25.6 Setting default parameter values 313
 - 25.7 Playing with script blocks 315
 - 25.8 More tips, tricks, and techniques 315
- 26 *Using someone else's script* 316**
- 26.1 The script 317
 - 26.2 It's a line-by-line examination 321
 - 26.3 Lab 321
 - 26.4 Lab answer 323
- 27 *Never the end* 325**
- 27.1 Ideas for further exploration 325
 - 27.2 "Now that I've read the book, where do I start?" 326
 - 27.3 Other resources you'll grow to love 327
- 28 *PowerShell cheat sheet* 328**
- 28.1 Punctuation 328
 - 28.2 Help file 331
 - 28.3 Operators 332
 - 28.4 Custom property and column syntax 332
 - 28.5 Pipeline parameter input 333
 - 28.6 When to use \$_ 334
- appendix* *Review labs* 335
- index* 347

preface

We've been teaching and writing about Windows PowerShell for a long time. When Don began contemplating the first edition of this book, he realized that most PowerShell writers and teachers—including himself—were forcing our students to approach the shell as a kind of programming language. Most PowerShell books are into “scripting” by the third or fourth chapter, yet more and more PowerShell students were backing away from that programming-oriented approach. Those students wanted to use the shell as a shell, at least at first, and we weren't delivering a learning experience that matched that desire.

So he decided to take a swing at it. A blog post on the Windows IT Pro website proposed a table of contents for this book, and ample feedback from the blog's readers fine-tuned it into the book you're about to read. He wanted to keep each chapter short, focused, and easy to cover in a short period of time—because we know administrators don't have a lot of free time and often have to learn on the fly. When PowerShell v3 came out, it was obviously a good time to update the book, and Don turned to Jeffery Hicks, a long-time collaborator and fellow MVP, to help out.

We both wanted a book that would focus on PowerShell itself, and not on the myriad technologies that PowerShell touches, like Exchange Server, SQL Server, System Center, and so on. We feel that by learning to use the shell properly, you can teach yourself to administer all of those “PowerShell-ed” server products. So this book focuses on the core of using PowerShell. Even if you're also using a “cookbook” style of book that provides ready-to-use answers for specific administrative tasks, this book will help you understand what those examples are doing. That understanding will make it easier to modify those examples for other purposes, and eventually to construct your own commands and scripts from scratch.

We hope this book won't be the only PowerShell education that you pursue. We've also co-authored *Learn PowerShell Toolmaking in a Month of Lunches*, which offers the same day-at-a-time approach to learning PowerShell's scripting and tool-creation capabilities. You can also find videos we've produced on YouTube and read articles we've authored for sites such as the Petri IT Knowledgebase and Windows IT Pro, not to mention take courses from Pluralsight.

If you need any further help, we encourage you to log on to www.PowerShell.org. We both answer questions in several of the discussion forums there, and we'd be happy to try to get you out of whatever you're stuck on. The site is also a great portal into the robust and active PowerShell community; you can learn about free e-books, the in-person PowerShell and DevOps Summit, and all of the regional and local user groups and PowerShell-related events that happen throughout the year. Get involved—it's a great way to make PowerShell a more powerful part of your career.

Enjoy—and good luck with the shell.

acknowledgments

Books don't write, edit, and publish themselves. Don would like to thank everyone at Manning Publications who decided to take a chance on a different kind of book for Windows PowerShell, and who worked so hard to make the first edition of this book happen. Jeff would like to thank Don for inviting him along for the ride, and the PowerShell community for their enthusiasm and support. Don and Jeff are both grateful to Manning for allowing them to continue the "Month of Lunches" series with this third edition.

Thanks also to the following peer reviewers who read the manuscript during its development and provided feedback: Bennett Scharf, Dave Pawson, David Moravec, Keith Hill, and Rajesh Attaluri. In addition, Erika Bricker, Gerald Mack, Henry Phillips, Hugo Durana, Joseph Tingsanchali, Noreen Dertinger, Olivier Deveault, Stefan Hellweger, Steven Presley, and Tiklu Ganguly provided valuable comments.

Finally, thanks also to James Berkenbile and Trent Whiteley for their technical review of the manuscript and code during production.

about this book

Most of what you need to know about this book is covered in chapter 1, but there are a few things that we should mention up front.

First of all, if you plan to follow along with our examples and complete the hands-on exercises, you'll need a virtual machine or computer running Windows 8.1 or Windows Server 2012, or later. We cover that in more detail in chapter 1. You can get by with Windows 7, but you'll miss out on a few of the hands-on labs.

Second, be prepared to read this book from start to finish, covering each chapter in order. Again, this is something we explain in more detail in chapter 1, but the idea is that each chapter introduces a few new things that you'll need in subsequent chapters. You shouldn't try to push through the whole book—stick with the one chapter per day approach. The human brain can absorb only so much information at once, and by taking on PowerShell in small chunks, you'll learn it a lot faster and more thoroughly.

Third, this book contains a lot of code snippets. Most of them are short, so you should be able to type them easily. In fact, we recommend that you do type them, because doing so will help reinforce an essential PowerShell skill: accurate typing! Longer code snippets are given in listings and are available for download from the book's page on the publisher's website at <https://www.manning.com/books/learn-windows-powershell-in-a-month-of-lunches-third-edition>.

That said, you should be aware of a few conventions. Code always appears in a special font, just as in this example:

```
Get-WmiObject -class Win32_OperatingSystem  
-computerName SERVER-R2
```

That example also illustrates the line-continuation character used in this book. It indicates that those two lines should be typed as a single line in PowerShell. In other

words, don't hit Enter or Return after `Win32_OperatingSystem`—keep right on typing. PowerShell allows for long lines, but the pages of this book can hold only so much.

Sometimes you'll also see that code font within the text itself, such as when we write `Get-Command`. That just lets you know that you're looking at a command, parameter, or other element that you would type within the shell.

Fourth is a tricky topic that we'll bring up again in several chapters: the backtick character (```). Here's an example:

```
Invoke-Command -scriptblock { Dir } `
-computerName SERVER-R2,localhost
```

The character at the end of the first line isn't a stray bit of ink—it's a real character that you would type. On a U.S. keyboard, the backtick (or grave accent) is usually near the upper left, under the Esc key, on the same key as the tilde character (`~`). When you see the backtick in a code listing, type it exactly as is. Furthermore, when it appears at the end of a line—as in the preceding example—make sure that it's the last character on that line. If you allow any spaces or tabs to appear after it, the backtick won't work correctly, and neither will the code example.

Finally, we'll occasionally direct you to internet resources. Where those URLs are particularly long and difficult to type, we've replaced them with Manning-based shortened URLs that look like <http://mng.bz/S085> (you'll see that one in chapter 1).

Author Online

The purchase of *Learn Windows PowerShell in a Month of Lunches, Third Edition* includes access to a private forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the authors and other users. To access and subscribe to the forum, point your browser to <https://www.manning.com/books/learn-windows-powershell-in-a-month-of-lunches-third-edition> and click the Author Online link. This page provides information on how to get on the forum after you're registered, the kind of help that's available, and the rules of conduct in the forum.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the authors can take place. It's not a commitment to any specific amount of participation on the part of the authors, whose contribution to the book's forum remains voluntary (and unpaid). We suggest you try asking the authors some challenging questions, lest their interest stray!

The Author Online forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

about the authors

DON JONES is a multiple-year recipient of Microsoft’s prestigious Most Valuable Professional (MVP) Award for his work with Windows PowerShell. For five years he wrote the Windows PowerShell column for *Microsoft TechNet Magazine*. He currently blogs at <http://PowerShell.org> and authors the “Decision Maker” column and blog for *Redmond Magazine*. Don is a prolific technology author and has published more than a dozen print books since 2001. He’s now a curriculum director for IT Ops content at Pluralsight, an online video training platform. Don’s first Windows scripting language was KiXtart, going back all the way to the mid-1990s. He quickly graduated to VBScript in 1995 and was one of the first IT pros to start using early releases of a new Microsoft product code-named Monad—which later became Windows PowerShell. Don lives in Las Vegas and, when it gets too hot there, near Duck Creek Village in Utah.

JEFFERY HICKS is an IT veteran with more than 25 years of experience, much of it spent as an IT infrastructure consultant specializing in Microsoft server technologies with an emphasis in automation and efficiency. He is a multiyear recipient of the Microsoft MVP Award in Windows PowerShell. He works today as an independent author, trainer, and consultant. He has taught and presented on PowerShell and the benefits of automation to IT pros all over the world. Jeff has written for numerous online sites and print publications, is a contributing editor at Petri IT Knowledgebase, a Pluralsight author, and a frequent speaker at technology conferences and user groups. You can keep up with Jeff at his blog, <http://jdhitsolutions.com/blog>, and on Twitter (@JeffHicks).

Before you begin

We've been teaching Windows PowerShell since version 1 was released in 2006. Back then, most of the folks using the shell were experienced VBScript users, and they were eager to apply their VBScript skills to learning PowerShell. As a result, we and the other folks who taught the shell, wrote books and articles, and so forth, all adopted a teaching style that takes advantage of prior programming or scripting skills.

But since late 2009, a shift has occurred. More and more administrators who *don't* have prior VBScript experience have started trying to learn the shell. All of a sudden, our old teaching patterns didn't work as well, because we had focused on scripting and programming. That's when we realized that PowerShell isn't a scripting language. It's a command-line shell where you run command-line utilities. Like all good shells, it has scripting capabilities, but you don't have to use them, and you certainly don't have to *start* with them. We started changing our teaching patterns, beginning with the many conferences we speak at each year. Don also implemented these changes into his instructor-led training courseware.

This book is the result of that process, and it's the best that we've yet devised to teach PowerShell to someone who might not have a scripting background (although it certainly doesn't hurt if you do). But before we jump into the instruction, let's set the stage for you.

1.1 Why you can't afford to ignore PowerShell

Batch. KiXtart. VBScript. Let's face it, Windows PowerShell isn't exactly Microsoft's (or anyone else's) first effort at providing automation capabilities to Windows administrators. We think it's valuable to understand why you should care about PowerShell, because when you do, you'll feel comfortable that the time you commit

to learning PowerShell will pay off. Let's start by considering what life was like before PowerShell came along, and look at some of the advantages of using this shell.

1.1.1 *Life without PowerShell*

Windows administrators have always been happy to click around in the graphical user interface (GUI) to accomplish their chores. After all, the GUI is largely the whole point of Windows—the operating system isn't called *Text*, after all. GUIs are great because they enable you to discover what you can do. Don remembers the first time he opened Active Directory Users and Computers. He hovered over icons and read tooltips, pulled down menus, and right-clicked things, all to see what was available. GUIs make learning a tool easier. Unfortunately, GUIs have zero return on that investment. If it takes you five minutes to create a new user in Active Directory (and assuming you're filling in a lot of the fields, that's a reasonable estimate), you'll never get any faster than that. One hundred users will take five hundred minutes—there's no way, short of learning to type and click faster, to make the process go any quicker.

Microsoft has tried to deal with that problem a bit haphazardly, and VBScript was probably its most successful attempt. It might have taken you an hour to write a VBScript that could import new users from a CSV file, but after you'd invested that hour, creating users in the future would take only a few seconds. The problem with VBScript is that Microsoft didn't make a wholehearted effort in supporting it. Microsoft had to remember to make things VBScript accessible, and when developers forgot (or didn't have time), you were stuck. Want to change the IP address of a network adapter by using VBScript? OK, you can. Want to check its link speed? You can't, because nobody remembered to hook that up in a way that VBScript could get to. Sorry. Jeffrey Snover, the architect of Windows PowerShell, calls this *the last mile*. You can do a lot with VBScript (and other, similar technologies), but it tends to let you down at some point, never getting you through that last mile to the finish line.

Windows PowerShell is an express attempt on Microsoft's part to do a better job and to get you through the last mile. And it's been a successful attempt so far. Dozens of product groups within Microsoft have adopted PowerShell, an extensive ecosystem of third parties depend on it, and a global community of experts and enthusiasts are pushing the PowerShell envelope every day.

1.1.2 *Life with PowerShell*

Microsoft's goal for Windows PowerShell is to build 100% of a product's administrative functionality in the shell. Microsoft continues to build GUI consoles, but those consoles are executing PowerShell commands behind the scenes. That approach forces the company to make sure that every possible thing you can do with the product is accessible through the shell. If you need to automate a repetitive task or create a process that the GUI doesn't enable well, you can drop into the shell and take full control for yourself.

Several Microsoft products have already adopted this approach, including Exchange Server 2007 and beyond, SharePoint Server 2010 and later, many of the System Center products, Office 365, and many components of Windows itself. Going forward, more and more products and Windows components will follow this pattern. Windows Server 2012, which was where PowerShell v3 was introduced, is almost completely managed from PowerShell—or by a GUI sitting atop PowerShell. That’s why you can’t afford to ignore PowerShell: Over the next few years, it’ll become the basis for more and more administration. It’s already become the foundation for numerous higher-level technologies, including Desired State Configuration (DSC), PowerShell Workflow, and much more. PowerShell is everywhere!

Ask yourself this question: If you were in charge of a team of IT administrators (and perhaps you are), who would you want in your senior, higher-paying positions? Administrators who need several minutes to click their way through a GUI each time they need to perform a task, or ones who can perform tasks in a few seconds after automating them? We already know the answer from almost every other part of the IT world. Ask a Cisco administrator, or an AS/400 operator, or a UNIX administrator. The answer is, “I’d rather have the person who can run things more efficiently from the command line.” Going forward, the Windows world will start to split into two groups: administrators who can use PowerShell, and those who can’t. As Don famously said at Microsoft’s TechEd 2010 conference, “Your choice is *learn PowerShell*, or *would you like fries with that?*”

We’re glad *you’ve* decided to learn PowerShell.

1.2 And now, it’s just “PowerShell”

In mid-2016, Microsoft took the previously unthinkable step of open sourcing all of Windows PowerShell. At the same time, it released versions of PowerShell—without the *Windows* attached—for macOS and numerous Linux builds. Amazing! Now, the same object-centric shell is available on many operating systems, and can be evolved and improved by a worldwide community. So for this edition of the book, we decided to make sure we addressed PowerShell on something other than Windows. We still feel that PowerShell’s biggest audience will be Windows users, but we also want to make sure you understand how it works on other operating systems.

1.3 Is this book for you?

This book doesn’t try to be all things to all people. Microsoft’s PowerShell team loosely defines three audiences who use PowerShell:

- Administrators who primarily run commands and consume tools written by others
- Administrators who combine commands and tools into more-complex processes, and perhaps package those as tools that less-experienced administrators can use
- Administrators and developers who create reusable tools and applications

This book is designed primarily for the first audience. We think it's valuable for anyone, even a developer, to understand how the shell is used to run commands. After all, if you're going to create your own tools and commands, you should know the patterns that the shell uses, as they allow you to make tools and commands that work as well as they can within the shell.

If you're interested in creating scripts to automate complex processes, such as new user provisioning, then you'll see how to do that by the end of this book. You'll even see how to get started on creating your own commands that other administrators can use. But this book won't probe the depths of everything that PowerShell can possibly do. Our goal is to get you using the shell and being effective with it in a production environment.

We'll also show you a couple of ways to use PowerShell to connect to external management technologies; Windows Management Instrumentation (WMI) and regular expressions are the two examples that come quickly to mind. For the most part, we're going to introduce only those technologies and focus on how PowerShell connects to them. Those topics deserve their own books (and have them—we'll provide recommendations when we get there), so we concentrate solely on the PowerShell side of things. We'll provide suggestions for further exploration if you'd like to pursue those technologies on your own. In short, this book isn't meant to be the last thing you use to learn about PowerShell, but instead is designed to be a great first step.

1.4 *How to use this book*

The idea behind this book is that you'll read one chapter each day. You don't have to read it during lunch, but each chapter should take you only about 40 minutes to read, giving you an extra 20 minutes to gobble down the rest of your sandwich and practice what the chapter showed you.

1.4.1 *The main chapters*

Of the chapters in this book, chapters 2 through 25 contain the main content, giving you 24 days' worth of lunches to look forward to. You can expect to complete the main content of the book in about a month. Try to stick with that schedule as much as possible, and don't feel the need to read extra chapters in a given day. It's more important that you spend some time practicing what each chapter shows you, because using the shell will help cement what you've learned. Not every chapter requires a full hour, so sometimes you'll be able to spend additional time practicing (and eating lunch) before you have to get back to work. We find that a lot of people learn more quickly when they stick with just one chapter a day, because it gives your brain time to mull over the new ideas, and gives you time to practice them on your own. Don't rush it, and you may find yourself moving more quickly than you thought possible.

1.4.2 Hands-on labs

Most of the main content chapters include a short lab for you to complete. You'll be given instructions, and perhaps a hint or two. The answers for these labs appear at the end of each chapter. But try your best to complete each lab without looking at the answers.

1.4.3 Code samples

Throughout the book, you'll encounter code listings. These are longer PowerShell examples. But don't feel you need to copy them. If you head to www.manning.com and find the page for this book, you'll see a link to download all of the code listings.

1.4.4 Supplementary materials

Don's YouTube channel, [YouTube.com/PowerShellDon](https://www.youtube.com/PowerShellDon), contains a bunch of free videos that he made for the original edition of this book—and they're all still 100% applicable. They're a great way to get some short, quick demos. He also hosts videos from recorded conference workshops and more, and they're all worth a look. We also suggest the [PowerShell.org](https://www.youtube.com/powershellorg) channel, [YouTube.com/powershellorg](https://www.youtube.com/powershellorg), which contains a ton of video content. You'll find recorded sessions from the PowerShell + DevOps Global Summit events, online community webinars, and a lot more. All free!

Jeff does a lot of writing for the Petri IT Knowledgebase (www.petri.com), where you'll find a huge collection of content covering all sorts of PowerShell topics. You might also see whether Jeff has anything new on his YouTube channel, <http://YouTube.com/jdhitsolutions>.

1.4.5 Further exploration

A few chapters in this book only skim the surface of some cool technologies, and we end those chapters with suggestions for exploring those technologies on your own. We point out additional resources, including free stuff that you can use to expand your skill set as the need arises.

1.4.6 Above and beyond

As we learned PowerShell, we often wanted to go off on a tangent and explore why something worked the way it did. We didn't learn a lot of extra practical skills that way, but we did gain a deeper understanding of what the shell is and how it works. We've included some of that tangential information throughout the book in sections labeled "Above and beyond." None of those will take you more than a couple of minutes or so to read, but if you're the type of person who likes to know why something works the way it does, they can provide some fun additional facts. If you feel those sections might distract you from the practical stuff, ignore them on your first read-through. You can always come back and explore them later, after you've mastered the chapter's main material.

1.5 **Setting up your lab environment**

You're going to be doing a lot of practicing in Windows PowerShell throughout this book, and you'll want to have a lab environment to work in; please don't practice in your company's production environment.

All you'll need to run most of the examples in this book—and to complete all of the labs—is a copy of Windows that has PowerShell v3 or later installed. We suggest Windows 8.1 or later, or Windows Server 2012 R2 or later, which both come with PowerShell v4. Note that PowerShell might not exist on certain editions of Windows, such as Starter editions. If you're going to play with PowerShell, you'll have to invest in a version of Windows that has it. Also note that some of the labs rely on functionality that was new in Windows 8 and Windows Server 2012, so if you're using something older, things might work differently. At the start of each lab, we tell you what operating system you need in order to complete the lab.

Keep in mind that, throughout this book, we're assuming you'll be working on a 64-bit operating system, also referred to as an *x64* operating system. As such, it comes with two copies of Windows PowerShell and the graphically-oriented Windows PowerShell Integrated Scripting Environment (ISE). In the Start menu (or, in Windows 8, the Start screen), the 64-bit versions of these are listed as *Windows PowerShell* and *Windows PowerShell ISE*. The 32-bit versions are identified by an *(x86)* in the shortcut name, and you'll also see *(x86)* in the window's title bar when running those versions. If you're on a 32-bit operating system, you'll have only the 32-bit version of PowerShell, and it won't specifically say *(x86)*.

The examples in this book are based on the 64-bit versions of PowerShell and the ISE. If you're not using those, you may sometimes get slightly different results than ours when running examples, and a few of the labs might not work properly. The 32-bit versions are primarily provided for backward compatibility. For example, some shell extensions are available only in 32-bit flavors and can be loaded into only the 32-bit (or *x86*) shell. Unless you need to use such an extension, we recommend using the 64-bit shell when you're on a 64-bit operating system. Microsoft's investments going forward are primarily in 64-bit; if you're stuck with a 32-bit operating system, unfortunately that's going to hold you back.

TIP You should be able to accomplish everything in this book with a single computer running PowerShell, although some stuff gets more interesting if you have two or three computers, all in the same domain, to play with. We've used CloudShare (www.cloudshare.com) as an inexpensive way to spin up several virtual machines in the cloud. If such a scenario interests you, look into that service or something like it. Note that CloudShare isn't available in all countries. Another possibility if you're running Windows 8 or later is to use the Hyper-V feature and run a few virtual machines there.

If you're using a non-Windows build of PowerShell, you'll have fewer options to worry about. Just get the right build for your version of macOS or Linux (or whatever) from

<http://github.com/PowerShell/PowerShell>, and you should be good to go. Keep in mind, however, that a lot of the *functionality* we'll be using in our examples is unique to Windows. For example, you can't get a list of services on Linux, because Linux doesn't have services (it has daemons, which are similar, but different).

1.6 Installing Windows PowerShell

Windows PowerShell v3 has been available for most versions of Windows since the release of Windows Server 2008, Windows Server 2008 R2, Windows 7, and later versions. Windows Vista isn't supported, but it can still run v2. The shell is preinstalled only on the most recent versions of Windows; it must be manually installed on older versions. PowerShell v4 is available for Windows 7 and later and Windows Server 2008 R2 or later, although those versions of Windows don't have as many components that are "hooked up" to PowerShell, which is why we recommend Windows 8 or Windows Server 2012 as minimum versions. And although PowerShell v4 isn't the latest version of the shell, that or anything later will suffice for this book's content.

TIP You should check your version of PowerShell: Open the PowerShell console, type `$PSVersionTable`, and hit Enter. If you get an error, or if the output doesn't indicate PSVersion 4.0, then you don't have PowerShell v4.

If you want to check the latest available version of PowerShell or download it, go to <http://msdn.microsoft.com/powershell>. This official PowerShell home page has links to the latest Windows Management Framework (WMF) installer, which is what installs PowerShell and its related technologies. Again, because this book is covering entry-level stuff, you'll find that not much has changed from v3, but it's always fun to have the latest version to play with.

PowerShell has two application components: the standard, text-based console host (PowerShell.exe) and the more visual ISE (PowerShell_ISE.exe). We use the text-based console most of the time, but you're welcome to use the ISE if you prefer.

NOTE The PowerShell ISE isn't preinstalled on server operating systems. If you want to use it, you'll need to go into Windows Features (using Server Manager) and manually add the ISE feature (you can also open the PowerShell console and run `Add-WindowsFeature powershell-ise`). The ISE isn't available at all on server installations that don't have the full GUI (for example, Server Core or Nano Server).

Before you go any further, take a few minutes to customize the shell. If you're using the text-based console host, we strongly recommend that you change the default console font to the Lucida fixed-width font. The default font makes it difficult to distinguish some of the special punctuation characters that PowerShell uses. Follow these steps to customize the font:

- 1 Click the control box (that's the PowerShell icon in the upper left of the console window) and select Properties from the menu.

- 2 In the dialog box that appears, browse through the various tabs to change the font, window colors, window size and position, and so forth.

TIP We strongly recommend you make sure that both the Window Size and Screen Buffer have the same Width values.

Your changes will apply to the default console, meaning they'll stick around when you open new windows. Of course, all of this applies only to Windows: On non-Windows operating systems, you'll usually install PowerShell, open your operating system's command-line (for example, a Bash shell), and run `powershell`. Your console window will determine your colors, screen layout, and so on, so adjust to suit your preferences.

1.7 **Contacting us**

We're passionate about helping folks like you learn Windows PowerShell, and we try to provide as many resources as we can. We also appreciate your feedback, because that helps us come up with ideas for new resources that we can add to the site, and ways to improve future editions of this book. You can reach Don on Twitter @concentratedDon, or Jeff @JeffHicks. We also both hang out in the forums of <http://PowerShell.org> if you have PowerShell questions. <http://PowerShell.org> is also a wonderful place for more resources, including free e-books, an in-person annual conference, free webinars, and tons more. We help run the organization, and we can't recommend it highly enough as a place to continue your PowerShell education after you've finished this book.

1.8 **Being immediately effective with PowerShell**

Immediately effective is a phrase we've made our primary goal for this entire book. As much as possible, each chapter focuses on something that you could use in a real production environment, right away. That means we sometimes gloss over some details in the beginning, but when necessary we promise to circle back and cover those details at the right time. In many cases, we had to choose between hitting you with 20 pages of theory first, or diving right in and accomplishing something without explaining all the nuances, caveats, and details. When those choices came along, we almost always chose to dive right in, with the goal of making you *immediately effective*. But all of those important details and nuances are still explained later in the book.

OK, that's enough background. It's time to start being immediately effective. Your first lunch lesson awaits.

Meet PowerShell

This chapter is all about getting you situated and helping you to decide which PowerShell interface you'll use (yes, you have a choice). If you've used PowerShell before, this material might seem redundant, so feel free to *skim* this chapter—you might still find some tidbits here and there that'll help you down the line.

Also, this chapter applies exclusively to PowerShell on Windows. Non-Windows versions don't come in as many options or flavors, so if that's your situation, you can skip this chapter.

2.1 Choose your weapon

On Windows, Microsoft provides two ways (four, if you're being picky) for you to work with PowerShell. Figure 2.1 shows the Start screen's Apps page, with four PowerShell icons. We've highlighted them to help you spot them more easily.

TIP On older versions of Windows, these icons are on your Start menu. You point to All Programs > Accessories > Windows PowerShell to find the icons. You can also select Run from the Start menu, type `PowerShell.exe`, and hit Enter to open the PowerShell console application. On Windows 8 and Windows Server 2012 or later, hold the Windows key on your keyboard and press R to get the Run dialog box. Or press and release the Windows key, and start typing `powershell` to quickly get to the PowerShell icons.

On a 32-bit operating system, you have only two (at most) PowerShell icons; on a 64-bit system, you have up to four. These include