BUILDING POWERSHELL MODULES

An in-depth guide
from design to production

BY BRANDON OLIN

Everything you ever wanted to know about creating useful, engaging, and high-quality PowerShell modules for the community

Building PowerShell Modules

An in-depth guide from design to production

Brandon Olin

This book is for sale at http://leanpub.com/building-powershell-modules

This version was published on 2021-09-12

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.

Tweet This Book!

Please help Brandon Olin by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just purchased ”Building PowerShell Modules” by @devblackops on @leanpub! https://leanpub.com/building-powershell-modules. Check it out if you want to up your #PowerShell game!

Also By Brandon Olin

ChatOps the Easy Way

Contents

1. About the Author

Brandon is a Site Reliability Engineer, Cloud Architect, veteran Systems Engineer, speaker, blogger, freelance writer, and open source contributor. He is a Microsoft MVP in Cloud and Datacenter Management and has a penchant for PowerShell and DevOps processes. He spends much of his time exploring new technologies to drive the business forward and loves to apply ideas pioneered in the DevOps community to solving real-world business problems. Brandon is also active in the PowerShell community and loves to give back with many projects published to the PowerShell Gallery[1].

You can follow his code at GitHub[2], his blog at devblackops.io[3], or reach him on Twitter at @devblackops[4].

Some of his other projects include:

Book: ChatOps the Easy Way
: An in-depth guide to implementing ChatOps using PoshBot.
https://leanpub.com/chatops-the-easy-way

Book: The PowerShell Conference Book
: Contributing author to the PowerShell "conference in a book" project. All proceeds go to the OnRamp scholarship program[5].
https://leanpub.com/powershell-conference-book

PoshBot
: A Powershell-based bot framework.
https://github.com/poshbotio/PoshBot

psake
: A build automation tool.
https://github.com/psake/psake

PowerShellBuild
: Common build tasks for psake and Invoke-Build that build and test PowerShell modules.
https://github.com/psake/PowerShellBuild

[1] https://www.powershellgallery.com/profiles/devblackops/
[2] https://github.com/devblackops
[3] https://devblackops.io
[4] https://twitter.com/devblackops
[5] https://powershell.org/summit-old/summit-onramp/

OperationValidation

A framework for using Pester to test operational validation.
https://github.com/PowerShell/Operation-Validation-Framework

Stucco

An opinionated Plaster template for high-quality PowerShell modules.
https://github.com/devblackops/Stucco

Terminal-Icons

A PowerShell module to show file and folder icons in the terminal.
https://github.com/devblackops/Terminal-Icons

2. About This Book

This book is intended for anyone wishing to increase their knowledge of PowerShell module development, or people already developing modules but want to take their skills to the next level. Whether you are creating internal modules for your organization to use or open-source modules for the community to enjoy, this book is for you.

This is not a "how to learn PowerShell" book. It is expected you already have the basics of PowerShell down, so things like PowerShell semantics and language features will not be the topic of this book. There are already some excellent books out there that cover the PowerShell language itself, such as Learn PowerShell in a Month of Lunches[1] or The PowerShell Scripting and Toolmaking Book[2] by Don Jones and Jeffery Hicks. I'd encourage you to take a look at these books first if you're interested in learning more about PowerShell *as a language*.

Consider this book a follow-up to them where you will learn how to create useful, high-quality, and engaging PowerShell modules with real-world guidance, examples, and concept discussion.

I promise you will learn something valuable from reading this book, and be able to apply that knowledge to your own use-cases.

Throughout the book, you will learn *"good practices"* on how to use and build high-quality PowerShell modules. This includes the basics of module management and the practices recommended to produce the best module you can, and foster an open and engaging community along the way.

—

If you've purchased this book, I sincerely thank you. Creating a book of this type takes considerable time and effort. We're talking hundreds of hours of planning, research, writing, coding, and editing. Treat this as your own **personal copy**. Leanpub provides DRM-free versions of the book for you to use in any way that is convenient for you, but it is not meant for distribution or resale. I appreciate you respecting my rights and not making unauthorized copies.

If you've found this book for free somewhere, know that it is an **unauthorized** copy. I hope you've gotten value from the book, and would greatly appreciate you purchasing a legitimate

[1] https://www.manning.com/books/learn-windows-powershell-in-a-month-of-lunches-third-edition
[2] https://leanpub.com/powershell-scripting-toolmaking

copy from Leanpub.com[3]. Doing this lets me know you've found the content useful, and encourages me to continue writing and improving the book.

2.1 PowerShell version

The code examples in this book are using PowerShell Core v6.2.3 running on Windows. Still, nearly everything shown will also work with Windows PowerShell 5.1 and probably earlier. Code examples should run just fine on Linux or macOS as well (provided platform-specific things like file paths are modified accordingly). If any code example specifically requires Windows PowerShell, it will be noted.

2.2 Roadmap

The book is divided into four separate *parts* to cover high-level concepts. Each part will consist of dedicated topic chapters relevant to that concept.

2.2.1 Part 1

Part 1 is a modules primer and will cover the basic building blocks for creating and working with modules.

Chapter 1 starts with covering the basics of PowerShell modules, including what they are and why you would want to build them. We will discuss some of the benefits of using modules such as code sharing, code reuse, and versioning. We'll also define common module terminology that will be used throughout the book. Finally, we'll discuss the different module types, including binary, script, and manifest modules.

In Chapter 2, we'll explore working with modules, including where they typically live on your system. We'll also learn about some capabilities contained in the PowerShellGet module, such as finding, installing, and uninstalling modules from the PowerShell Gallery. This chapter will also describe how to explore various aspects of PowerShell modules such as the commands they export, dependencies to other modules, and discovering module metadata. We'll also cover how module auto-loading works.

In Chapter 3, we cover the basic structure of script modules and show how to turn a regular PowerShell script into a script module. We'll also discuss in detail the various components of a module manifest. How to update a manifest, and test that it is valid will also be covered.

[3] Leanpub.com

In Chapter 4, we'll talk about dealing with module dependencies, including what they are and the two different types. We're also going to learn how module dependencies can be leveraged to save us time, and how to define dependencies in our own modules. We'll end with how dependent modules are installed from the PowerShell Gallery.

Chapter 5 is the last in part 1 of the book and will explain what PowerShell repositories are and how to create a local one. It will also show how to publish a module to the local repository, and give a tour of the PowerShell Gallery, the central location for community modules.

2.2.2 Part 2

Part 2 will discuss various aspects of project and module design, including the design decisions and project elements that make for a high-quality product.

In Chapter 6, we cover a foundational aspect of module design - the layout of the module and discuss a few options we have to choose from. We'll also explain why you should pick one of these options over the others.

Chapter 7 discusses the difference between public and private functions, and how, using the dot-sourced function pattern, we can easily control the export of the correct ones. We will also walk through how to determine if a function should be private or not. Lastly, we'll discuss how to export variables and aliases from a module.

Chapter 8 gives an overview of how to use PowerShell classes in a module, including the nuances of exporting them from a module. The differences between importing a module with `Import-module` and the new `using module` statement will also be shown.

Other chapters to be written.

2.2.3 Part 3

Part 3 goes over creating an efficient build and test development loop. Chapters in this part cover source-control basics, software testing, and Continuous Integration/Continuous Delivery (CD/CD) topics.

Chapters to be written.

2.2.4 Part 4

Part 4 puts everything discussed so far into practice by walking through creating a high-quality PowerShell module as an open-source project.

Chapters to be written.

2.2.5 Bonus

Bonus chapters covering topics I feel should be explained but aren't directly related to building PowerShell module. Rather than mess with the flow of the book, these bonus topics will be covered here.

3. Feedback

This is an agile-published book that is offered on Leanpub.com[1]. Updates are published as they are written. If you've purchased a copy early on in the writing process, first off, I sincerely thank you. Second, rest assured that you will automatically receive free updates as they become available. Also, as PowerShell module practices evolve, so too will this book.

This book contains a fair amount of code samples, a large amount of collective "good practices", and a decent amount of opinions!

If you notice an error in code, believe I've stated something incorrectly, or flat out think I'm dead wrong about *this* or *that*, please let me know! Mistakes happen, and I want this book to be the best it can be. In order to do that, your feedback is vital. Please post an issue on the Building PowerShell Modules - Feedback [2] GitHub repository if you think something should be corrected.

[1] https://leanpub.com/
[2] https://github.com/devblackops/building-powershell-modules-feedback

4. Typographic Conventions

Every effort will be made to keep code from wrapping to a new line, but PowerShell is a verbose language, and best practice PowerShell conventions will be followed in this book wherever possible, so the use of aliases or shortened parameter names will not be used.

In code samples or console output, *slight* edits may be made in order to get the code to look the best it can be on the page. Any edits done will not have a material impact on the code.

Commands that will have more than three parameters will use `hashtables` and splatting instead of specifying the parameters inline. For instance, the following example is a long line of PowerShell that wraps to the next line and doesn't look that great.

```
Find-Module -Name Pester -RequiredVersion 4.9.0 -Repository PSGallery | Install-Module -S\
cope CurrentUser
```

You can expect to see the example above, instead, be written like this:

```
$findParams = @{
    Name            = 'Pester'
    RequiredVersion = '4.9.0'
    Repository      = 'PSGallery'
}
$pester = Find-Module @findParams
$pester | Install-Module -Scope CurrentUser
```

Every effort will be made to make the code as readable as possible while also following PowerShell best-practices.

Links to relevant resources and websites will be added inline, like this example to the PowerShell[1] repository on GitHub. You can click on these links to view them directly in the browser when reading the eBook and PDF formats of this book, or follow the URLs below in the footnote section when reading the hard copy version.

[1] https://github.com/powershell/powershell

4.1 Asides

Occasionally, asides are used in this book to highlight certain information.
Some example asides are:

This is an Aside Box

It will be used to emphasize or describe something.

This is a Warning Box

Warnings will be used to call out something dangerous.

This is a Tip Box

Tips will be used to point out something useful.

This is an Error Box

Errors will be used to highlight that something has broken or describing a bug of some type.

This is an Information box

Information boxes will be used for special information.

This is a Question Box

Questions boxes will be used to ask the reader a question or to think about a concept.

This is an Exercise Box

Exercise boxes will be used to call out something for the reader to do.

5. Dedication

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

6. Introduction

PowerShell, first released in 2006, has truly changed the lives of the IT professionals who have chosen to embrace it as a means to automate their work. It has been a force-multiplier. Using PowerShell has allowed them to connect the dots between business processes, automate manual tasks away, and enabled its practitioners to provide more value to their organizations. People who have adopted PowerShell have advanced their careers in the process.

For a long time, PowerShell was Windows-only, though. Having been based on the full .NET framework, it wasn't possible to run it on Linux or macOS. Most organizations have some mix of Windows, Linux, and macOS running in their environment. PowerShell's full potential was blocked by being relegated to only one platform.

It was also closed-source, and Microsoft was entirely in control of its development and feature-set. New releases were incredibly slow to come out, often tied to the Windows release cycle. This meant new features or bug fixes were slow to materialize, and the feedback cycle was too long.

In recent years that has all changed. .NET is cross-platform now, and open-source! With that, PowerShell 6 can run on a variety of platforms, including Linux and macOS. An open-source PowerShell allows people from around the world to contribute to it, making it better for everyone. Releases are now more frequent, with feedback being addressed sooner.

This book isn't about PowerShell *the language*. There are plenty of excellent books about that, and I encourage you to pick one up. Instead, this book is about a specific concept in PowerShell, that of creating PowerShell modules that encapsulate bits of functionality you (and others) can reuse. Creating PowerShell modules is covered in other books. Still, it has not been the primary focus, and some ancillary concepts are not included in detail, if at all.

Modules are the intended way to package and distribute the PowerShell code that we write. Yet to many, *how* to create a good PowerShell module, and *what* precisely makes a module good in the first place, is still a bit blurry. There are certainly answers to these questions online, but the information is scattered and hard to digest as a single concept. This book is meant to help with that.

It is not the intent of this book to dictate *the one true way* of creating modules. Instead, I want to share my own experience with creating them, testing them, publishing them, and engaging with the community to develop better tools for everyone. I consider these *"good*

practices", not *"best practices"*. They can be context-specific and may not make sense in every scenario. As with most things, use your best judgment.

If you are writing PowerShell, then, in my opinion, you are a software developer, even if your job title says otherwise. Throughout this book, you're going to learn about and leverage concepts like basic source-control, software testing, Continuous Integration/Continuous Delivery (CI/CD), and how to create helpful documentation for the end-user. These are not PowerShell-specific concepts. They have been around much longer than PowerShell itself. But they *are* good software-development skills, and knowing how to apply them benefits anyone that is touching code, no matter the language.

If nothing else, in the spirit of continuous improvement, it is my hope that gaining knowledge about these areas encourages you to apply them to other aspects of your work.

Hope you enjoy it,
Brandon

Part 1 - Modules Primer

Before we start putting together all the building blocks for what it takes to create high-quality PowerShell modules, we first need a foundation from which to build upon. This part of the book covers those foundations, including what exactly PowerShell modules are, what components they are comprised of, and how to interact with them. We will also show how to get modules from external sources and build a basic module. We'll end with how to manage dependencies in our modules and talk about what PowerShell repositories are, how to create a local one, and publish to it.

Consider this part a warm-up and perhaps a refresher before we dive deep into making modules the best they can be.

7. Module Basics

You will learn

- What modules are
- What role modules play in PowerShell
- Why building and using modules is great
- The various types of modules

Perhaps up until this point in your PowerShell career, you have been mostly using or writing PowerShell scripts (.ps1) files to get things done. For any new task or problem you face, you write a script that solves it, and you save this handy piece of code that just saved your bacon and impressed your boss into a folder called "scripts" in your user directory. Whenever you face a new challenge, you think to yourself: "Ah-ha! I think I have a script that does something like that. Now, where did I put it?"

This is great when first learning PowerShell. You're creating a collection of tools that you can rely upon and call into action when needed. Everyone thinks you're a hero when you quickly rummage through your bag of tricks, pull something out, and save the day.

The trouble is you've now started to accumulate a large collection of scripts with duplicate code in them. Every time you've been asked to pull a user report from your ERP or sales system, you copy an existing script that does something similar, modify whatever logic you need to satisfy the new requirements, and run it.

The problem is pulling the information from the backend system happens to involve a decent amount of complex logic. It requires a few SQL queries, or REST API calls. Maybe some special filtering, etc. Every time you copy a script that contains this code, you're duplicating business logic. What happens if you ever need to change how you pull information from the backend system? It's not in a central location but instead spread across many different scripts. In short, it's starting to become a maintenance nightmare!

If only there was a way to centralize this business logic. If only you could reuse this code in different scenarios. If only this code didn't **just** live on your system where your coworkers

can't access and use as they see fit. If only there was a mechanism to package and share this code with others. We could name this fascinating piece of technology a *"module"*, sell it, and make millions! If only!

7.1 What is a module

Since we're going to be talking quite a bit about modules, heck, this is a whole book about them, let's start off with the basics and build up from there. If you're already comfortable with modules, feel free to skim through this chapter if you'd like, but it's a good idea to level set on terms, so we have a solid foundation to build upon. For those where PowerShell modules are a new concept, let's dive in!

Think of a module as a package or library of related PowerShell code that you can work with as a unit, and re-use readily. Instead of a loose collection of script files you have tucked away in some folder on your desktop, a PowerShell module is treated as a single thing, a single unit that you install, load, and use as needed.

Modules export their capabilities to the PowerShell session that imported the module. What a module chooses to export becomes the public entry points we have to access the module. These entry points could be commands, variables, aliases, or DSC resources. In computer science terminology, this would be like an interface[1]. It is the public boundary we have with the module. How the module operates internally is an implementation detail that doesn't concern us.

Modules usually contain many separate *but related* commands that operate against a single system or technology domain. Since they are all related, it's convenient to package them up as a single unit. When packaged in a module, it's also easy to *find* existing commands for working with technologies like Active Directory as an example, so you don't have to reinvent the wheel. We all like coding, but why do the work when someone else has already done the heavy lifting for you? There are literally *thousands* of modules on PowerShellGallery.com[2] (The PowerShell community's primary repository for modules). We'll explore this great resource in the next chapter. For now, just know that there is a high likelihood that the functionality you desire is probably already out there.

[1] https://en.wikipedia.org/wiki/Interface_(computing)
[2] https://powershellgallery.com

You are already using modules

You may not realize it, but every built-in command shipped with PowerShell is packaged in, you guess it, a module! Thanks to features like module auto-loading (which we'll discuss in chapter 2), these commands are seamlessly imported when needed. For instance, the `Install-WindowsFeature` command comes from the `ServerManager` module. We'll discuss how all this works in the next chapter.

7.2 Why build modules

Now that you got a brief explanation of *what* a PowerShell module is, the next question is: "Why build them?"

7.2.1 Organizing related functionality

Modules are PowerShell's way of bundling similar commands together. In Microsoft's own documentation about modules[3], they describe them as:

> A module is a package that contains PowerShell commands, such as cmdlets, providers, functions, workflows, variables, and aliases.
>
> People who write commands can use modules to organize their commands and share them with others. People who receive modules can add the commands in the modules to their PowerShell sessions and use them just like the built-in commands.

7.2.2 Code sharing

Did you notice the blurb from the official definition that modules are also used to share with others? This is what the PowerShell Gallery is all about. It's a central spot to discover, download, and publish modules for the entire community to enjoy. The great thing is it's

[3] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_modules?view=powershell-6

completely free and open for anyone to use and contribute to. Don't worry; in later chapters, we'll dig deeper into the ins and outs of the gallery as well.

But code sharing doesn't just mean the PowerShell Gallery. There are plenty of people that develop modules internal to their organization. These modules are never meant to be publicly available. They probably operate on internal systems, line of business apps, or manage proprietary processes. These modules are sometimes called organizational, internal, or private, and they are just as important as publicly available ones. They support the business, after all. Sharing these amongst your coworkers is also essential, and there are mechanisms to build internal PowerShell repositories that operate much the same as the public PowerShell Gallery. We'll show the steps needed to do this in later chapters as well.

7.2.3 Code reuse

Remember that we don't want to reinvent the wheel. There is little reason to create a custom function that writes text to a file. PowerShell already provides the `Set-Content` cmdlet that does an excellent job with that. If we need to modify the contents of a file, we'll probably leverage commands from the PowerShell module `Microsoft.PowerShell.Management`. That is a module that is bundled automatically with PowerShell. You can see all `Content` related commands by running:

Get all commands that have -Content in the name

```
Get-Command -Name *-Content*
```

```
PS>Get-Command -Name *-Content*

CommandType Name           Version Source
----------- ----           ------- ------
Cmdlet      Add-Content    7.0.0.0 Microsoft.PowerShell.Management
Cmdlet      Clear-Content  7.0.0.0 Microsoft.PowerShell.Management
Cmdlet      Get-Content    7.0.0.0 Microsoft.PowerShell.Management
Cmdlet      Set-Content    7.0.0.0 Microsoft.PowerShell.Management
```

There is a principle of software development called **Don't Repeat Yourself** or **DRY**. Essentially this means that if you find you are repeating a bit of logic in your code, you should *centralize* this logic and reference this single copy in the rest of your program. This avoids redundancy and makes it easier to modify the logic later on.

We already practice this when writing functions in our scripts. If you are working with a REST API that requires some custom authentication logic, you won't re-implement this logic every time you need to access the REST API. No, you'd create a custom function, stick all the logic in there, then call that function elsewhere in your script. A module serves a similar purpose but at a different layer. If we had multiple scripts that needed this custom authentication logic, we wouldn't copy/paste this handy function in each script. What if we needed to change it? Instead, we can create a module to store this function *once*. Any script can then *import* this module and gain access to the function. If we ever needed to update the REST API logic, we can create a new version of the module with the new logic.

7.2.4 Versioning

We experience software versioning all the time, usually on the consuming side of things. Anytime we install a piece of software, that software includes some defined set of functionality available at the time the software was released. If the authors of that software want to introduce new features or bug fixes, they release a new version of the software for us to install. Sometimes that new version may change existing functionality, and the new behavior is incompatible with how it behaved before. If this new behavior is important to us, we need to make a decision on whether to use the latest version.

When writing software, like we are with developing PowerShell modules, we must think from the producing side, as well as the consuming side. What if we had to change how we interact with the REST API drastically? Perhaps we now need to provide different parameters to the function. We may have many different production scripts using this module, and we don't want to break them suddenly. To solve this, we can introduce a new version of the module that includes the new functionality. Existing scripts can still use the old version. We can modify our existing scripts when we're ready to have them start using the newer version. This allows us to introduce changes slowly without breaking the world. If we didn't have a way to version things, this would be impossible to do.

To list the available versions of a module we have installed on a system, we can use the `Get-Module` cmdlet with the `-ListAvailable` switch.

In the following example, we're listing all the locally installed versions of the Pester[4] module. This module is installed by default in Windows 10, Windows Server 2016 and above, and PowerShell Core on Linux/macOS.

[4] https://github.com/pester/Pester

Listing the available versions of the Pester module

```
Get-Module -Name Pester -ListAvailable
```

```
PS> Get-Module -Name Pester -ListAvailable


    Directory: C:\Users\Brandon\Documents\PowerShell\Modules

ModuleType Version Name   PSEdition ExportedCommands
---------- ------- ----   --------- ----------------
Script     4.9.0   Pester Desk      {Describe, Context, It, Should…}
Script     4.8.1   Pester Desk      {Describe, Context, It, Should…}
Script     4.7.3   Pester Desk      {Describe, Context, It, Should…}
Script     4.7.2   Pester Desk      {Describe, Context, It, Should…}
Script     4.7.1   Pester Desk      {Describe, Context, It, Should…}
Script     4.6.0   Pester Desk      {Describe, Context, It, Should…}
Script     4.4.1   Pester Desk      {Describe, Context, It, Should…}
```

Notice that I have many different versions of the Pester[5] module installed on my system. The commands in this module may behave differently between versions. If I'm a consumer of this module, I may have code that assumes Pester will operate in a certain way. To protect my code from accidentally breaking, I can *pin* my code to a specific version of Pester and safely install a *new* version of the module without actually using it. I can start using that newer version and take advantage of new features or bug fixes at my own discretion.

7.3 Terminology

We're going to be using a few different terms related to modules, so let's go over them quickly before we get in too deep. Modules are comprised of a few different components:

Root module
: The main module file loaded when the module is imported. This could be a script or binary module file.

Module manifest
: A PowerShell data file with various metadata describing the module.

[5] https://github.com/pester/Pester

Module member
: Functions, variables, or aliases defined inside the module. Using the `Export-ModuleMember` cmdlet, you can control which members are exposed outside the module.

Module type
: The type of module. This can be script, binary, or manifest.

Nested module
: Additional modules that this module will import. Nested modules are not visible outside the primary module.

Exported member
: Any module member (function, cmdlet, variable, or alias) that has been marked for export. These are the members that are visible outside the module.

Imported member
: Any module member such as a function, cmdlet, variable, or alias that has been imported from another module. These would be exported members from that other module.

7.4 Module types

So now that we know what modules are and why we should use them, let's talk about the various *types* of modules.

7.4.1 Script modules

The first type of module we'll explore is called a *script* module. As the name implies, this type of module is written in the PowerShell scripting language itself. Instead of a file extension of `.ps1` like standard PowerShell scripts, script modules have a file extension of `.psm1`. In general, any PowerShell script can be converted into a script module just by changing the file extension to `.psm1`.

We're going to focus on script modules for the remainder of this book because they are by far the most popular type of modules that people write.

7.4.2 Binary modules

The second type of module we'll explore is called a *binary* module. These modules contain code that has been compiled into a binary format. With a binary module, classes are written in the C# language and compiled into a .NET assembly (DLL) using PowerShell C# libraries.

Nearly all of the built-in cmdlets that PowerShell provides "out of the box" are packaged in binary modules.

Because binary modules are written in C#, they are typically faster and more efficient than the script-based modules that are the primary focus of this book. Developing binary modules requires knowledge of the C# programming language, but don't let that scare you. Experience with PowerShell is an excellent foundation for entering the world of traditional programming languages like C#.

7.4.3 Manifest

Manifest modules include only metadata about the module. They do not contain executable code like script or binary modules, but they can define dependencies to other modules that do contain code. When the manifest module is imported, any dependencies defined are imported as well. Manifest modules used in this way can act as a wrapper to bundle many modules together and install or import them at the same time.

7.4.4 What is a manifest

A manifest is a file containing metadata about a module. This can include the name of the module, a unique identifier (GUID), the `root module` to load when the module is imported, amongst other things. We can think of the manifest much the same as a shipping manifest describing the contents of a crate of cargo. It tells us what is inside the crate without us having to open it.

The format of the manifest is a PowerShell hashtable in a file with a `.psd1` extension. The name of the manifest file should be the same as the module it defines.

To create a basic manifest, you use the `New-ModuleManifest` cmdlet like so:

Creating a new module manifest

```
1 New-ModuleManifest -Path .\MyModule.psd1
```

Running the previous command will produce a file with content similar to the following snippet. Only a portion is shown here to give you a sense of the contents:

A portion of a module manifest

```
@{

# Script module or binary module file associated with this manifest.
# RootModule = ''

# Version number of this module.
ModuleVersion = '0.0.1'

# Supported PSEditions
# CompatiblePSEditions = @()

# ID used to uniquely identify this module
GUID = '33f32d39-2bfb-4263-97d8-84a1af20bce9'

# Author of this module
Author = 'brandon'

# Company or vendor of this module
CompanyName = 'Unknown'

# Copyright statement for this module
Copyright = '(c) brandon. All rights reserved.'

# Description of the functionality provided by this module
# Description = ''

# Functions to export from this module, for best performance, do not use wildcards and
# do not delete the entry, use an empty array if there are no functions to export.
FunctionsToExport = @()

# Cmdlets to export from this module, for best performance, do not use wildcards and
# do not delete the entry, use an empty array if there are no cmdlets to export.
CmdletsToExport = @()

# Variables to export from this module
VariablesToExport = '*'
...
}
```

There are many more components to a manifest, and we'll go over them in detail in chapter 3. The `New-ModuleManifest` command has parameters that match each property of the

manifest so we can choose to supply them and have the manifest populated automatically, or we can manually modify it after the fact.

While not *technically* required to create a PowerShell module, we shouldn't really call a module complete without one. Don't worry; this will become clear in later chapters when we create our first module. For now, just know that module manifests contain important information about a module and affect what happens when a module is imported into a PowerShell session.

7.5 Summary

Key Points

- Modules are PowerShell's way of packaging up commands (cmdlets/functions) into a single unit.
- Modules contain commands that are related to each other.
- Using modules, we can easily share our code with the community, or internally in our organizations.
- We can easily reuse commands by centralizing them inside modules.
- We can have multiple versions of a module.
- Modules contain several different components like a module manifest, root module, exported members, etc.
- There are three different types of modules (binary, script, and manifest).
- Manifests are an important component of creating high-quality modules.

Now that we've explored what modules are and the benefits they provide, how do we actually work with them in real life? In chapter 2, we'll learn just that as we explore interacting with and managing modules.

8. Working with Modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.1 Module locations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.1.1 $env:PSModulePath

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.1.1.1 Adding directories to $env:PSModulePath

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.2 PowerShellGet

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.2.1 Finding modules online

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.2.2 Installing a module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.2.3 Updating modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.2.4 Uninstalling modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.3 Discovering module information

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.3.1 Listing installed modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.3.2 Getting module commands

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.3.3 Exploring a module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.4 Installing modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.4.1 Creating a hello world module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.4.2 Manual installation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.4.3 PowerShellGet

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.4.3.1 Installation scopes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.4.3.2 Clobbering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.5 Saving modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.6 Importing modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.6.1 Modules are single-instance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.6.2 Module prefixes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.7 Being specific with module specifications

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.8 Module auto-loading

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.8.1 Controlling auto-loading behavior

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.9 Removing modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.9.1 Removing a module by name

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.9.2 Removing a module by using a module specification

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.9.3 Removing a module by PSModuleInfo

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.9.4 Caveats

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

8.10 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9. Authoring a Module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.1 Basic structure of a module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.1.1 Rules and conventions for module files

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.1.2 Exceptions to the rules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.2 Creating a script module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.2.1 Creating a basic script

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.2.2 Turning a script into a module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3 Manifests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.1 Manifest structure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.2 Creating a module manifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.3 Elements of a manifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.3.1 Identity properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.3.2 Runtime properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.3.3 Content properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.3.4 Private data properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.4 Updating a manifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.4.1 Programmatically updating a manifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.5 Testing a manifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.3.5.1 Limitations of Test-ModuleManifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

9.4 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10. Dealing with Module Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1 Module dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.1 Implicit dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.2 Explicit dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.2.1 Module pinning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.2.2 Testing with ease

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.3 Focusing our efforts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.4 Defining dependencies to other modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.1.5 Installing dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

10.2 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11. Distributing Modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1 PowerShell repositories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1.1 The default repository

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1.2 Trusting repositories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1.3 Repository types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1.3.1 NuGet

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1.3.2 File share

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.1.3.3 Third-party NuGet repositories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.2 Creating a local repository

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.2.1 Registering a repository

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.3 Publishing a module locally

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.3.1 Publishing a module by name

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.3.2 Publishing a module by file path

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.4 The PowerShell Gallery, a tour

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.4.1 Navigating the Gallery

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.4.2 Registering on the Gallery

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.4.3 Managing API keys

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

11.5 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

Part 2 - Project and Module Design

Creating high-quality PowerShell modules requires up-front thought and good design choices. For a functional module, we need a good design, implementation of that design, and to be always thinking about the end-user experience. Without answers to these, even the best code in the world can become unwieldy, hard to use, and ultimately not be as successful as it could be.

In part 2, we will discuss the elements of practical module and project design, and why you would choose some decisions over others. These decisions matter, regardless of what our module is trying to accomplish.

This part of the book will be a mix of theory, practical advice, and a healthy dose of useful code. In parts 3 and 4, we'll put these concepts to good use.

12. Choosing a Module Layout

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.1 Monolithic PSM1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.1.1 Module structure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.2 Category submodules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.2.1 Grouping by domain

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.3 Dot-sourced functions from PSM1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.3.1 What happens during import

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.3.2 Advantages and disadvantages

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.4 Final verdict

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

12.5 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

13. Keeping Some Module Contents Private

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

13.1 Determining the public functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

13.1.1 The role of private functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

13.1.2 Candidates for private functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

13.1.3 Controlling the visibility of variables and aliases

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

13.2 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14. Using Classes in a Module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.1 Importing classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.1.1 The *using* statement

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.1.2 Differences from Import-Module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.1.2.1 `using module` and versions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.1.3 Differences from #requires

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.2 Using classes internally in a module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.2.1 Trying to dot-source classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.3 Ordering classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.4 Extending classes from other modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.4.1 Having a module reference classes from another module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.4.1.1 Downsides of `using module` to access shared classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.5 The user experience of classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.5.1 Providing functions to create class instances

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

14.6 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15. Building a Module From Many Files

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.1 Benefits of a monolithic PSM1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.2 Benefits of dot-sourced functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.3 Creating a dot-sourced module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.4 Building a module - E pluribus enum (out of many, one)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.4.1 Creating a project directory

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.4.2 Creating a build script

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.4.2.1 Creating an output directory

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.4.3 Running the build script

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.4.4 Exporting functions in the manifest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

15.5 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16. Versioning a Module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.1 What is a software version

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.1.1 Examples of versioning schemes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.1.1.1 Date-based versioning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.1.1.2 Change significance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.2 Semantic Versioning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.2.1 SemVer rules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.2.1.1 Version 0 is “special”

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.2.2 Why SemVer is useful

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.3 Automating module versioning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.3.1 Avoid auto-incrementing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.3.2 Incrementing a module version

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.4 Pre-release versioning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.4.1 Adding a pre-release moniker

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.4.2 Publishing a pre-release module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.4.3 Finding and installing a pre-release module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.4.3.1 Updating to a newer pre-release version

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.5 Tips for avoiding breaking changes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.6 Handling deprecation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.6.1 Communication avenues

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.6.1.1 GitHub issue

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.6.1.2 Social media

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.6.1.3 Notifying in the module

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.7 Specifying compatible operating systems and PowerShell editions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.7.1 The CompatiblePSEditions module manifest field

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

16.8 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17. Building Better Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.1 Plan your dive, dive your plan

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.1.1 Planning your functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.1.2 Think about the experience

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.2 Functions should do one thing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.3 Creating testable functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.4 Self-contained functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5 Writing defensive functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.1 Enforce mandatory parameters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.2 Explicitly define parameter types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3 Use validation attributes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3.1 `[ValidateCount()]`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3.2 `[ValidateLength()]`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3.3 `[ValidatePattern()]`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3.4 `[ValidateRange()]`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3.5 `[ValidateScript()]`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.5.3.6 `[ValidateSet()]`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.6 Document your functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.7 Adhere to approved verbs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.8 Use proper parameter names

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.9 Use advanced functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.10 Support the pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.11 Create safe functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.11.1 Adding -WhatIf and `-Confirm` to your functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.11.1.1 Required confirmation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.11.2 Using the -Force

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.12 Sensible error handling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.12.1 Creating terminating errors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.12.1.1 Using the ThrowTerminatingError() method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.12.2 Creating non-terminating errors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.13 Write with cross-platform in mind

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.13.1 Working with the PATH environment variable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.13.2 File paths and delimiters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.13.2.1 Using Join-Path

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.13.2.2 Dipping into .NET

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

17.14 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18. Creating a Quality GitHub Project

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.1 The README

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.1.1 Elements of the README

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.1.2 Jazzing up the README with badges

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.2 Choosing a license

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.3 Keeping track of what changed

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.4 The Code of Conduct

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.5 Tell folks how to contribute

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.6 Issue and pull request templates

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.6.1 Template front matter and Markdown

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.6.2 Pull request template

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.7 Divvying up the work with CODEOWNERS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.7.1 CODEOWNERS Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.8 Project sponsors and funding

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

18.9 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

19. Documenting a Project

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

Part 3 - The Build and Test Loop

In part 3, we discuss how to create an efficient build and test development loop. The basics of source control, and how to create a Continuous Integration (CI) workflow will be the focus. We'll talk about the importance of testing, and the steps needed to package and publish a module with Continuous Delivery (CD).

Part 4 - Creating a Quality Community Project

In part 4, we put everything we've learned thus far into practice. We're going to create a high-quality PowerShell module, using all the concepts we've talked about, and go into depth about why everything is the way it is. We've reached the culmination of the book. After this, you have everything you need to create quality, engaging modules of your own!

Bonus Chapters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20. PowerShell Classes Explained

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.1 What is a PowerShell Class

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.2 Why use classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.3 Defining and creating a class

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.4 Adding class properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.5 Defining class methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.6 Initializing class properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.7 Validating properties by using attributes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.8 Accessing the class members with $this

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.9 Passing input to methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.10 Method overloading and signatures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.10.0.1 Finding method signatures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.10.0.2 Method parameter ordering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.11 Making properties and methods static

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.11.1 Accessing static properties

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.11.2 Calling static methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.12 Initializing a class instance with constructors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.13 Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.13.1 Creating a derived class

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.13.2 Overriding methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.13.3 Calling base methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.13.4 Calling constructors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.14 Hiding properties and methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.15 Enumerations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.15.1 Using an enumeration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.15.2 Using enumerations in parameters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.15.3 Creating an enumeration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.15.4 Flag enumerations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.

20.16 Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/building-powershell-modules.