From Scripting to Toolmaking: Taking the Next Step with PowerShell

SPTCEWORLD2019

Joe Houghes

Veeam Software

SPTCEWORLD2019

About Me

Austin, Texas native; Solutions Architect @ Veeam
Co-leader - Austin VMUG & PowerShell User Group
Green hat, big grin and the loud "Howdy Y'all."

Automate yourself out of a basic IT job
Free yourself to perform higher-level tasks

Push yourself outside of your comfort zone
Learn more about any topic of interest

Stop letting fear keep you from sharing with others
Share your knowledge (Perspective is important)
Teach whatever you can (Be The Master)

Agenda

Start with a Plan

Concepts of Functions & Modules

Well-Defined Parameters

Pipeline Input & Error Handling

Tips to Cheat and Steal (Really, Borrow – and Give Credit)

What Comes Next - After You Ship

Other Resources and Advanced Topics to Learn

Takeaways: Basic concepts, sample code, other resources

The chance to hang out with me

Why We're Here

SPICEWORLD2019

Who's in the audience?

Who in the room is an IT Pro? How about a Developer?

Who uses VSCode for their scripting/toolmaking? PowerShell ISE? A different IDE, like PowerShell Studio? Notepad?

Who uses source control for their code?

Who uses unit testing (Pester) for their code?

Who is running, or has started testing, PowerShell Core? PowerShell 7?

How many PowerShell Experts in the room?

We're all learning, so be sure to write it down and share with others.

Challenges Faced by Customers

SMB & SLED – Coverage, budget, time, training

Commercial – Coverage, budget, time, app lifecycle, infrastructure

FED – Auditing, compliance, time, access restriction

Enterprise – Scalability, repeatability, time, auditing

MSP – Scalability, repeatability, time, auditing, customer capability

Anyone in IT is always short of time, not tasks

Maintaining Control with Tools

How do you:

- Deploy/Change/Backup new systems in your environment?
- Make changes to your environment?
- Handle advanced configurations & deployments?
- Test/confirm/audit changes?

Is all of this a manual or automated process?

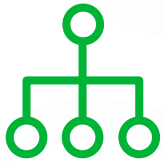
The Planning Stage

SPICEWORLD2019

Start With A Plan

Define a goal. Think tasks & measurable results.

Speed/performance = advantage, not benefit.



Focus on standardization, validation and efficiencies gained.

Use small, incremental changes that can be verified.

Basics of a Toolmaking Plan

Determine your MVP (minimum viable product)

Document requirements & assumptions

Define your process in small tasks

Avoid unnecessary functionality

Try to keep code simple & readable

Don't forget the PowerShell Basics

Get-Help is your friend, make your life easier.

PowerShell - built for pipeline, embrace & utilize it

Stack cmdlets and results like Legos

(Buy a set if you are outdated on the concept)

Forget what your teachers told you:

Cheat/look up the answer whenever possible.

Starting Some Code

Script or Function – Why?

When to Script, When to Function

- Scripts – All-in-one, self container, hardly updated
- Functions – Tool-oriented, modular, easier to update for new functionality

Advantages of Functions

- Module auto-loading
- Public/internal repository
- Easier to share

When to Use Both

- Controller scripts – use to drive your functions/modules
- Best of both worlds – customization for your environment, flexibility elsewhere

Proper Function Naming & Conventions

Use proper/consistent naming for functions

Use proper/consistent naming for parameters (also variables)

Use approved verb & singular noun

Add prefix if necessary

Follow community best practices, use PSScriptAnalyzer

Advanced Functions Best Practices

Accomplished by [CmdletBinding()]

Add comment-based help (also MAML if online)

Perform only a specific task

- Get data, process data, or output data (as objects)

Error handling – try/catch; throw, return, ErrorAction

Do NOT format, that is a separate function

Recommendations for Functions

Be careful with native cmdlets & functionality – Use #Requires

Verbose output with Write-Verbose

SupportsShouldProcess = WhatIf/Confirm

Use well-defined parameters

- Type/Mandatory → Alias → Validation → Parameter Set → Pipeline Input

Understand the Begin/Process/End blocks, especially for pipeline

Getting More Advanced

SPICEWORLD2019

Controller Scripts

This is where you import modules & automate your specific environment

Can contain environment specific data/variables

Avoid hard-coding usernames/passwords

Can truly build your outside-the-box Lego style customization

Menus, GUIs and JEA – oh my!

Next Steps for Functions

Create snippets to save time

Save your functions in modules with proper manifests

Leverage proxy functions to add/remove from native cmdlets

If using PowerShell v5+, consider using classes

Get your code into Git

Next Steps for Functions (cont.)

Create format.ps1xml files

Leverage type data

Look into Plaster for “scaffolding” of modules, Pester tests, DSC configs, etc.

Leverage Pester for unit testing of your code

Publish to internal or PowerShell Gallery

Community Resources

SPICEWORLD2019

Additional Resources

PowerShell.org YouTube Channel:

- Functions/Toolmaking - Don Jones Toolmaking
- Proxy Functions - Jeff Hicks - Accelerated Toolmaking
- Optimizing Performance – Joshua King - Whip Your Scripts into Shape

The PowerShell Scripting & Toolmaking Book – Don Jones/Jeff Hicks

ATXPowerShell YT Channel - Debugging PowerShell in VSCode w/Josh Duffney

PSKoans – Learn PowerShell through Pester

Plaster – "Scaffolding" of modules, Pester tests, DSC configs, etc.

PlatyPS – Write PowerShell External Help in Markdown

Questions?

SPICEWORLD2019

Thank you.

SPTCEWORLD2019