Learning PowerShell

from Script to Module

PowerShell on the River

August 2023

Mike

mikenelsonio

nelmedia

mikenelson-io

- Almost 40 years in tech
- Principal Technical Evangelist @ Pure Storage
- Experience from Helpdesk to Architect
- Scripter, not a coder
- Passion for community, teaching, learning
- Beer, BBQ, & Gadgets

MVP Microsoft Most Valuable Professional

Citrix Technology Advocate CTA

vmware vEXPERT

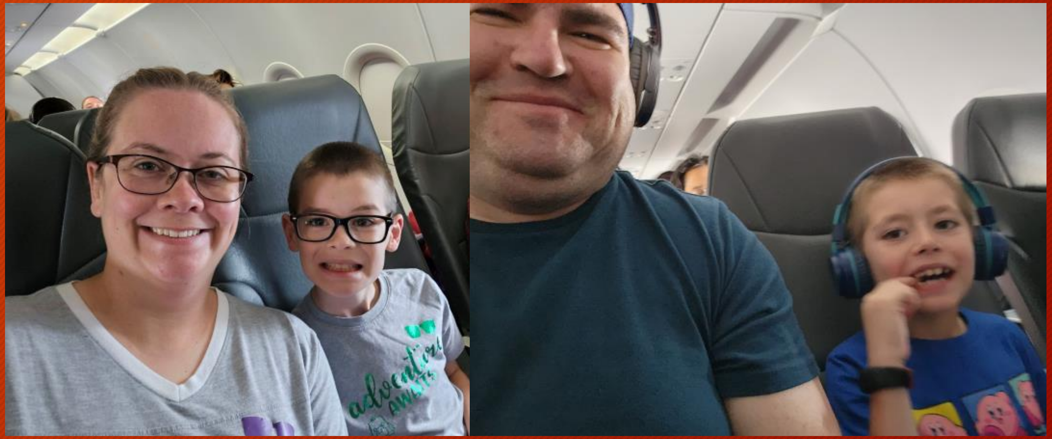
Joe

jhoughes

joehoughes

jhoughes

- 20 years in IT as customer, partner, vendor
- Senior Solutions Architect @ Pure Storage
- •Operations - SP - Architect - Sales Weasel (Vendor)
- Scripter, previously hacky coder
- Serial community collector, learning, sharing
- Texas BBQ, Volunteering, Beer, Geek Stuff



Microsoft® MVP Most Valuable Professional

vmware® vEXPERT ★★★★★★★★

veeAM VANGUARD

Cisco Champion

While we babble on…

Download and install PowerShell latest

    Google “powershell github”

Download and install VSCode

How this should go

- Who we are
- Why we are here
- What we will learn
- How we will learn it

Will we cover everything?

Why & When of PowerShell?

PURESTORAGE

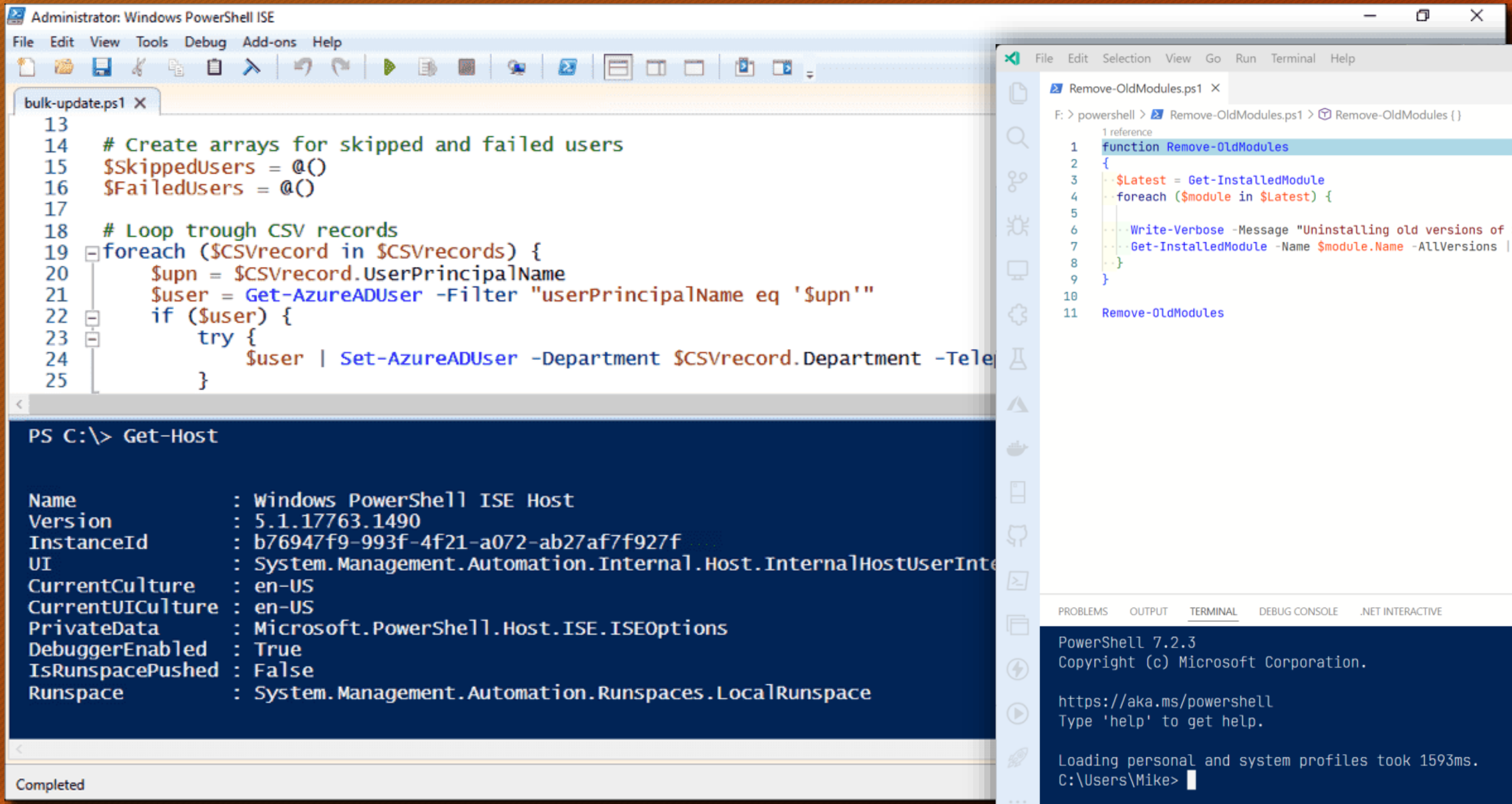
PowerShell

aka PoSH

Started as a scripting framework for automation & evolved into a command line interface (CLI) and a scripting language.

Native executables, cmdlets, scripts, functions, aliases, modules, help, profiles, parameters, and more

Creating / Editing / Running


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
bulk-update.ps1

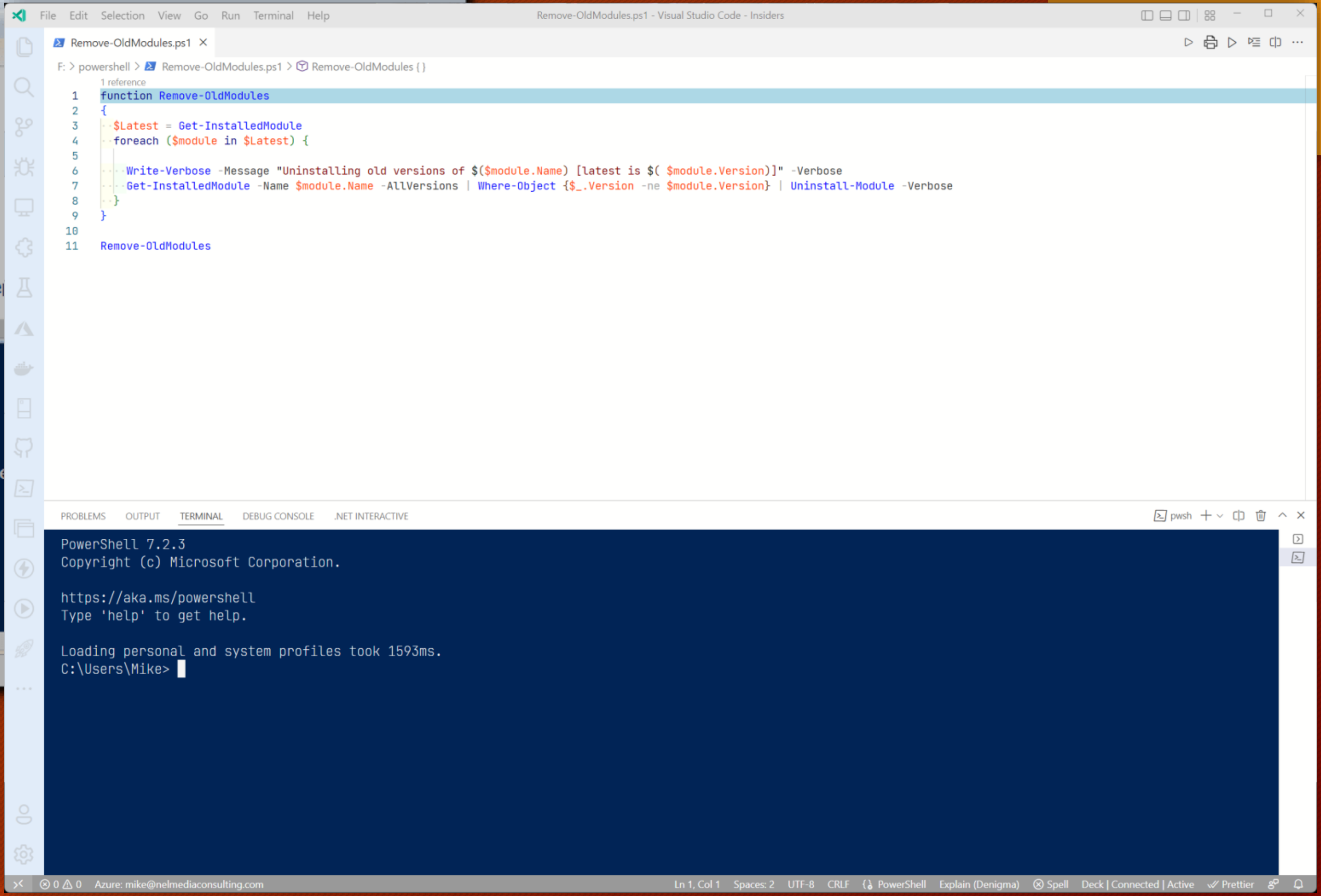
```
# Create arrays for skipped and failed users
$SkippedUsers = @()
$FailedUsers = @()

# Loop trough CSV records
foreach ($CSVrecord in $CSVrecords) {
    $upn = $CSVrecord.UserPrincipalName
    $user = Get-AzureADUser -Filter "userPrincipalName eq '$upn'"
    if ($user) {
        try {
            $user | Set-AzureADUser -Department $CSVrecord.Department -Tele
        }
```

```
PS C:\> Get-Host


Name             : Windows PowerShell ISE Host
Version          : 5.1.17763.1490
InstanceId       : b76947f9-993f-4f21-a072-ab27af7f927f
UI               : System.Management.Automation.Internal.Host.InternalHostUserInte
CurrentCulture   : en-US
CurrentUICulture : en-US
PrivateData      : Microsoft.PowerShell.Host.ISE.ISEOptions
DebuggerEnabled  : True
IsRunspacePushed : False
Runspace         : System.Management.Automation.Runspaces.LocalRunspace
```

Integrated Scripting Environment (ISE)


File Edit Selection View Go Run Terminal Help
Remove-OldModules.ps1 - Visual Studio Code - Insiders
Remove-OldModules.ps1
F: > powershell > Remove-OldModules.ps1 > Remove-OldModules { }
1 reference

```
function Remove-OldModules
{
  $Latest = Get-InstalledModule
  foreach ($module in $Latest) {

    Write-Verbose -Message "Uninstalling old versions of $($module.Name) [latest is $( $module.Version)]" -Verbose
    Get-InstalledModule -Name $module.Name -AllVersions | Where-Object {$_.Version -ne $module.Version} | Uninstall-Module -Verbose
  }
}

Remove-OldModules
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE .NET INTERACTIVE
pwsh

```
PowerShell 7.2.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

Loading personal and system profiles took 1593ms.
C:\Users\Mike>
```

0 0 Azure: mike@nelmediaconsulting.com
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF PowerShell Explain (Denigma) Spell Deck | Connected | Active Prettier

Visual Studio Code (VSCode)

** Use VSCodium for security-minded folks*

Versions

.Net Framework

.NET Core

<=5.1
Windows

~~6.x~~

7.x

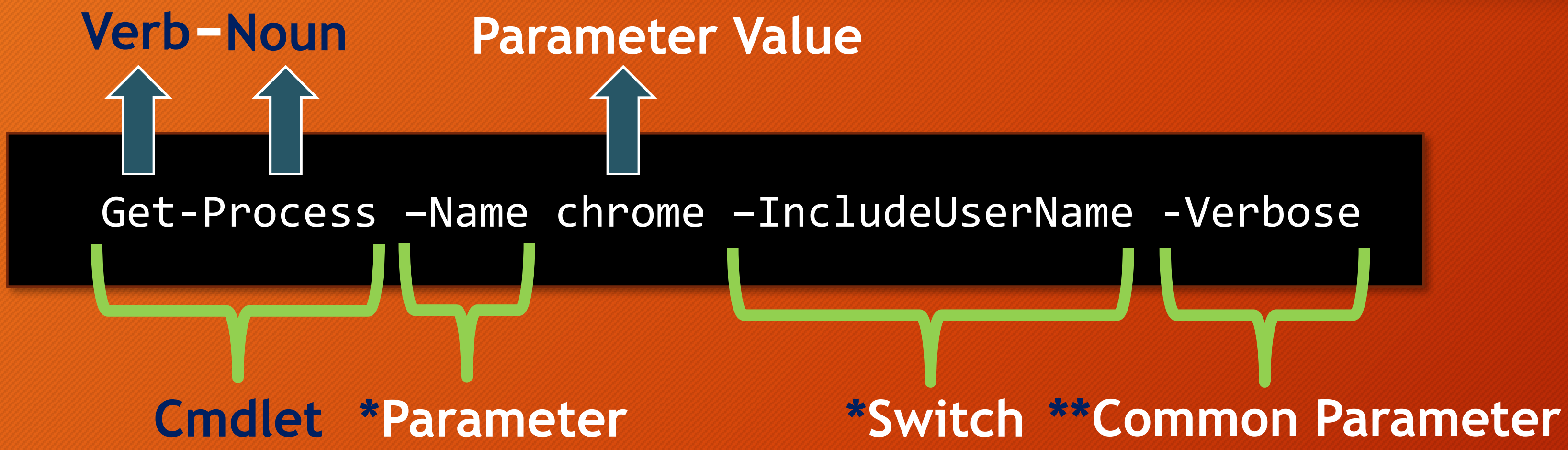
Windows
Linux
MacOS

Cmdlets

“command-lets”

- The “soul” of PowerShell
- A type of command in PowerShell
- Common syntax & options
- Almost always takes objects as input & return objects as output
- Used at the command line or placed in a .ps1 file.

Example Syntax



```
Get-Process -Name chrome -IncludeUserName -Verbose
```

* = Optional or required

** = the cmdlet may or may not support

Get-Verb

Profiles

- The PowerShell profile is a script that runs when a PowerShell session is started (unless the –noprofile switch is used).
- Basically, it is a logon script for PowerShell containing commands, aliases, variables, drives, functions, modules, etc.
- You can have different profiles for different user scopes, and there is no default profile.
- Your current user profile is stored in the $profile variable. To edit your current user profile with VSCode, type code $profile at a PowerShell prompt.

Aliases

- Get-Alias
- Just like it is – an alternative way to call something
- Can be set in the $Profile or New-Alias
- Native & cmdlet default aliases & custom aliases
  - Native default – sleep → Start-Sleep
  - Native module - ?? → copilot
  - Custom – gs → Get-GitStatus
  - Custom – np → notepad.exe

Variables

- A unit of *memory* in which a value is stored
- PowerShell variables are text strings represented by the dollar sign "$"prefix (ex. $a, $my_var, $var1, etc.)
- Although special characters and spaces allowed, variable names should be kept simple
- Types of variables:
  - User – user defined and deleted on exit (add to your PowerShell Profile to sustain)
  - Automatic – defined by Posh & not editable (ex. $PSHOME)
  - Preference – defaults defined & are user editable
- Type `Get-Variable` to show all variables defined in a session

Parameters

- Allow for users to provide input or options
- A pre-hyphen (“-”) is not always necessary (ie. a positional parameter)
- Some parameters have default values (creator decision)
- Parameter sets → different scenarios, different parameters
- Different Types:
  - Named -> default full name of parameter
  - Positional -> typed in a relative order (caution)
  - Dynamic -> only available under special conditions
  - Common -> built-in parameters
  - Sets -> expose different parameters & return different information

Exercise 1

Edit your profile to add an alias

Construct a command line that will:

- Find the ProcessName of 2 process that are running on your machine
- Execute a Get-Process to view those 2 processes
- Include the user name in the output
- Run the command with full output of what it is doing

Exercise 2

Construct a command line that will:

- Return all of the processes on your machine that start with the letter “a”
- Exclude all services that start with the word “com”

Extra Credit

Construct a command that will return the top 5 processes

Pipelines

Pipeline operator

```
PS>Get-Process -Name chrome -IncludeUserName | Stop-Process
```

Object(s) returned by first cmdlet are sent (piped) to the second cmdlet

Not all cmdlets or cmdlet parameters accept pipeline input

Pipelines

“One-liner”

Get all Windows VMs that need updated tools, then update all the tools at once

```
PS> Get-VM -Location 'MyDatacenter' | Where-Object { $_.ExtensionData.Guest.ToolsVersionStatus
 -eq 'guestToolsNeedUpgrade' -and $_.PowerState -like 'PoweredOn' } |
Get-VMGuest | Where-Object { $_.GuestFamily -like 'WindowsGuest'} |
Update-Tools -NoReboot -RunAsync
```

cmd.exe max character limit? 8,191

PowerShell max character limit? 32,764

PowerShell command separator? “||” Ex. Get-Process || Get-Disk

Exercise 3

Construct a command that will:

- Return the object for a “widget” process
- Pipe that object to a cmdlet to stop that process

Extra credit:

- Specify the service named object first and then pipe that to a cmdlet that will return the object and its status.
- Then add a pipeline to stop the service.
- Then reverse it and add a pipeline to start the service
- And do this all “verbose”

Functions

A list of PowerShell statements that run like you had entered them on the command line.

```
function Get-ChromeProcess { Get-Process chrome }

function Get-ChromeProcess {
  $a = Get-Process chrome
    if ($a –eq $null) {
      Write-Host “No Chrome process present”}
    return $a
}

Get-ChromeProcess
```

To run a function, simply “call” it.

Exercise 4

Create a simple function that:

- Returns the PowerShell process and is called Get-PwshProcess

Create an advanced function that:

- That displays all the files in $HOME folder, excludes any directories in that folder, and that have a size smaller than the value of the user-supplied size value.
- Extra Credit: Add a default parameter of 50 for the value of size.

Scripts

- Review: Scripts are .ps1 files.
- Scripts can contain one line or thousands.
- Scripts can accept parameters.
- Scripts should be signed. Set-ExecutionPolicy –AllSigned, -UnRestricted, etc. (global setting - Win only)
- To run, use the full path or ./ (dot-source).
- MacOS & Linux can execute via #! (Sha-Bang or Magic Number), but runs in a new session.

Scripts

Simple:

```
$date = (Get-Date).dayofyear
Get-Service | Out-File "$date.log“
```

With a parameter:

```
param ($ComputerName = $(throw "ComputerName parameter is required."))
function CanIPingIt {
   $a = Test-Connection $computername -erroraction SilentlyContinue
   if (!$?)
       {write-host "Ping failed: $ComputerName."; return $false}
   else
       {write-host "Ping succeeded: $ComputerName"; return $true}
}
```

Reusing Scripts

- Use script(s) in other scripts
- Called “sourcing” scripts
- Four ways to do it:
  - Using “.” operator
  - Using “&” operator
  - Using Invoke-Expression cmdlet
  - Using Start-Process cmdlet

Exercise 5

Create a simple script that:

- Returns all the files in your $HOME folder, including hidden files.

Add a pipeline:

- Use the same command as above but add a pipeline to a cmdlet that will return the total number of files returned.

Extra-Extra Credit:

- Use "splatting" to define a function called Get-MyCommand that will will return either the process information, or the command information, or both, for a named process if a cmdlet switch or switches are used.

Modules

- Modules are .psm1 files, which can contain commands, providers, variables, functions, help context, aliases, workflows, etc., all bundled into a single file.
- A .psd1 file is a module manifest file, which is basically a definition file for a module.
- Modules can be autoloaded by PowerShell
  - Uses the Abstract Syntax Tree (AST) to determine module from cmdlet

Example Manifest & Modules

PureStoragePowerShellToolkit.psd1

PSM1Template

PSQuizMaster.psm1

CustomizeWindows11

Core Commands to Know

- Get-Command
  - Show-Command
- Get-Help
  - ShowWindow
- Get-Member
- Update-Help
- Update-Module

about_Topics

The Lab

Create a:

Manifest and module that contains all of the commands and scripts you created in the exercises.

Extra Credit:

- Create a module that only calls .ps1 files as functions and does not have them natively in the module file.

Thank you!

@ joehoughes @mikenelsonio