# OPCUG Share PowerShell Overview

STEPHANE RICHARD

25 JAN 2023

# A few definitions first

- In computing, a **shell** is a computer program that exposes an operating system's services to a human user or other programs.
- A command-line interpreter or command-line processor uses a **command-line interface (CLI)** to receive commands from a user in the form of lines of text.
- Many users rely upon **Graphical User Interfaces (GUI)** and menu-driven interactions. However, some programming and maintenance tasks may not have a graphical user interface and use a command line.
- Programs with command-line interfaces are generally easier to automate via **scripting**.
- The Windows command line is an example of a shell.

# PowerShell background

- Way back in 2001, Microsoft started working on something to replace the aging DOS command prompt shell and batch file (i.e. script) and to have a new way to develop scripts.
- PowerShell is a task automation and configuration management program from Microsoft, consisting of a command line shell and the associated scripting language.
- In 2006 Microsoft released version 1 of the PowerShell programming language.
- In 2016, it was made open-source and cross-platform for Windows, MacOS, and some distribution of Linux.
- Cmdlets (pronounced command-lets) are specialized commands in the PowerShell environment that implement specific functions.
- PowerShell scripts are saved in text file with an extension of .ps1.

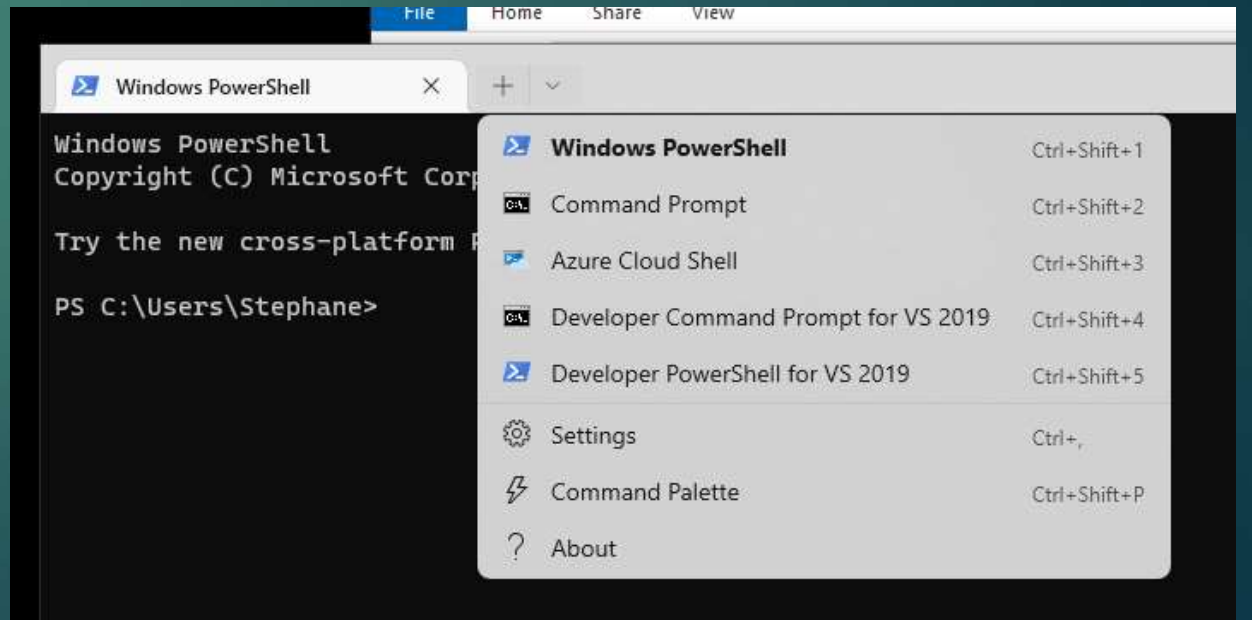
# Current PowerShell Version

## Windows 10

- Shipped with PowerShell Version 5.1 and Windows PowerShell Integrated Scripting Environment (ISE).
- PowerShell Version 7.1 is available to be installed.
- Windows PowerShell ISE:
  - can be used to create, run, and debug commands and scripts.
  - is no longer in active feature development, but "no plans to remove the ISE from Windows".
  - Suggest to use Visual Studio Code with the PowerShell Extension when asked to install.

## Windows 11

- As of Jul 2022, Windows 11 Build 22622.436 sets Windows Terminal as the default terminal app on Windows. That means scripts, applications, i.e. PowerShell and the old Command Prompt, opens in Terminal which is also available in Windows 10.



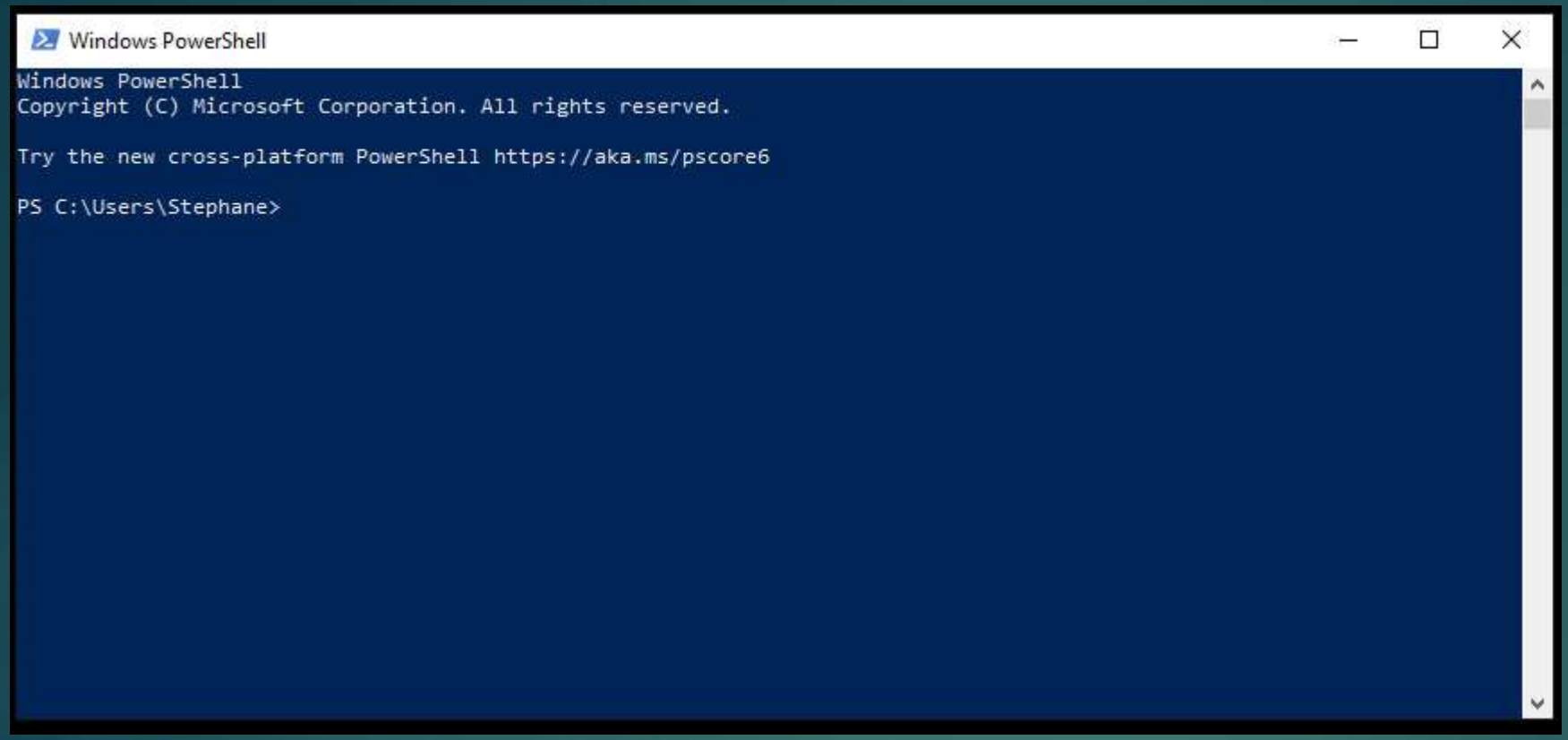
# PowerShell programing components

- Cmdlet and alias
- Variables (integers, floating point, strings, Booleans, datetime and object)
- Conditional Statements (if then else, switch)
- Loops (For, While, Do While, ForEach) NO GOTO
- Arrays and Hash table
- Define custom function
- Create custom object (aka class) and cmdlet
- Customising the console
- User input
- Cursor control
- Background processing
- Networking
- Files processing (text, xml, etc.)
- Sound
- Regular expression
- Error and exception handling
- Data parsing
- and much more...

# Cmdlet

- PowerShell uses a **verb-and-noun** name pair to name cmdlets.
- For example, the Get-Command cmdlet included in PowerShell is used to get all the cmdlets that are registered.
- You can add custom cmdlet with associated help information.
- By using the pipe symbol " | " you can send the output of one cmdlet to another one.
- On the next slide, an example of PowerShell Get-Command and Measure-Object cmdlets to get how many cmdlets are installed and the list of all cmdlets.

# Example to count number of Cmdlets




PowerShell automatically change the color of Cmdlet to yellow

PowerShell automatically change the color of Cmdlet's parameter to light grey


MS-DOS has 59 commands

# Cmdlet approved verbs (alias) list

- To get the list of verbs type: Get-Verb.  You will get a list with 98 rows:
  - Add (a), Approve (ap), Assert (as), Backup (ba), Block (bl), Build (bd), Checkpoint (ch), Clear (cl), Close (cs), Compare (cr), Complete (cp), Compress (cm), Confirm (cn), Connect (cc), Convert (cv), ConvertFrom (cf), ConvertTo (ct), Copy (cp), Debug (db), Deny (dn), Deploy (dp), Disable (d), Disconnect (dc), Dismount (dm), Edit (ed), Enable (e), Enter (et), Exit (ex), Expand (en), Export (ep), Find (fd), Format (f), Get (g), Grant (gr), Group (gp), Hide (h), Import (ip), Initialize (in), Install (is), Invoke (i), Join (j), Limit (l), Lock (lk), Measure (ms), Merge (mg), Mount (mt), Move (m), New (n), Open (op), Optimize (om), Out (o), Ping (pi), Pop (pop), Protect (pt), Publish (pb), Push (pu), Read (rd), Receive (rc), Redo (re), Register (rg), Remove (r), Rename (rn), Repair (rp), Request (rq), Reset (rs), Resize (rz), Resolve (rv), Restart (rt), Restore (rr), Resume (ru), Revoke (rk), Save (sv), Search (sr), Select (sc), Send (sd), Set (s), Show (sh), Skip (sk), Split (sl), Start (sa), Step (st), Stop (sp), Submit (sb), Suspend (ss), Switch (sw), Sync (sy), Test (t), Trace (tr), Unblock (ul), Undo (un), Uninstall (us), Unlock (uk), Unprotect (up), Unpublish (ub), Unregister (ur), Update (ud), Use (u), Wait (w), Watch (wc), Write (wr)
- To get all the nouns associated with the verb “Get”:  Get-Help -verb get
  - You get a list with 477 rows…
  - There is a lot of stuff that you can get!

# Resources

- PowerShell Get-Help cmdlet
- Google
- ChatGPT (according to Alan, this works well)
- The book PowerShell for Beginners by Ian Walters:
  - Highly recommended, no programming knowledge required
  - From the introduction "Throughout this book, you are going to start with the very basics and learn what tools you need and how scripts can be structured and how you can write code to perform the tasks you need and build up your own useful set of PowerShell tools."
- Many other books are unrelated topics arranged in a book. For example, the book Learning PowerShell by Stack Overflow Contributors after a short introduction, jump to Active Directory and getting Amazon Web Services, which may not be the first thing you need to learn, but if you have a specific need, there is probably a chapter in a book that covers it.

# My personal experience

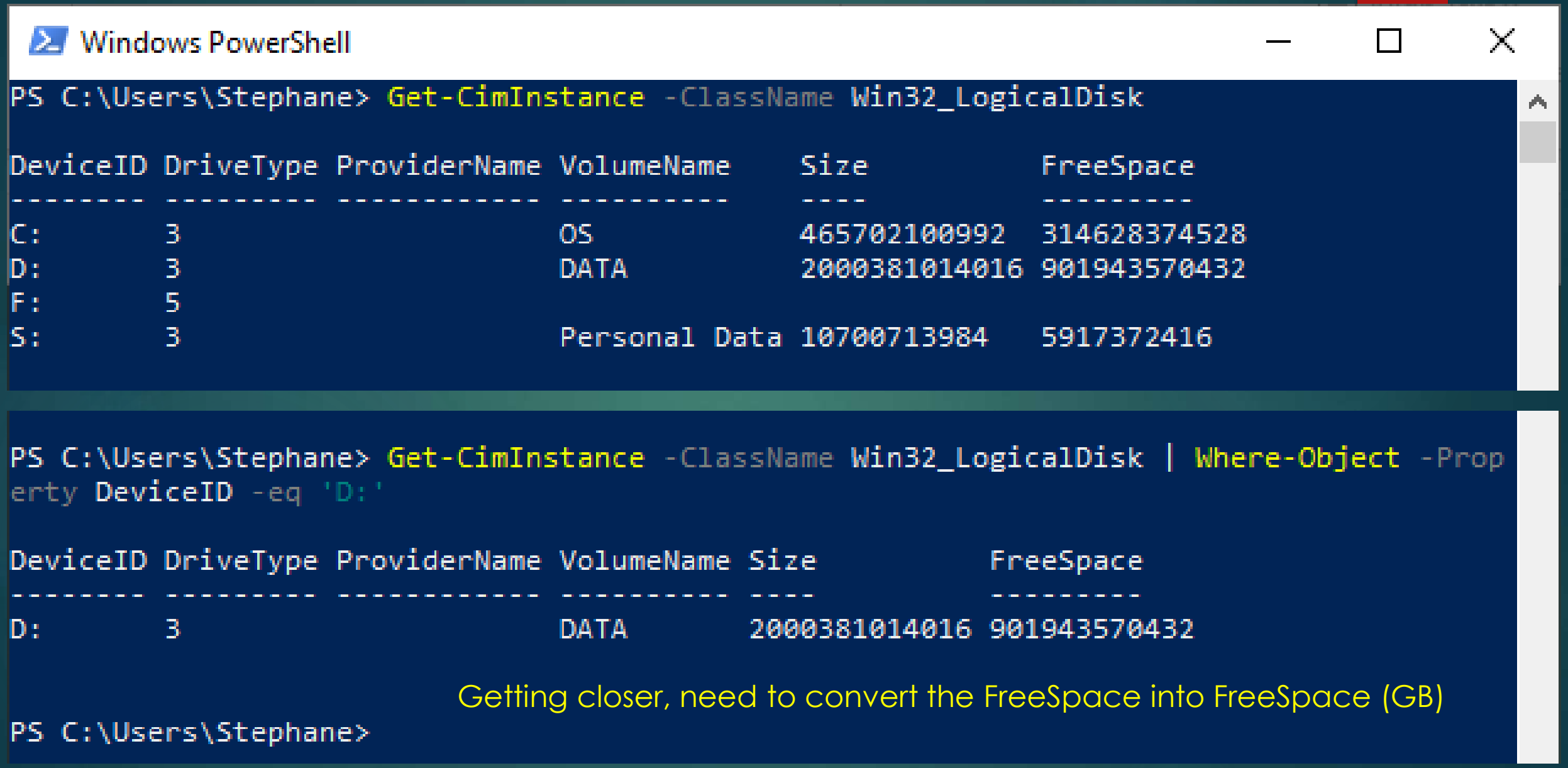
- I have been using MS-DOS and Windows Command Line for 37 years.
- I am not a system administrator, so I do not use it everyday, so I still have to lookup the proper syntax for many commands.
- Debugging MS-DOS batch file is a nightmare as I need to comment out instructions until I find the offending one.
- For my backup batch file, I wanted to add the disk free space to the log.  I found a way, but it is not pretty:
  - “ 2 Dir(s)  2,771,132,702,720 bytes free”
  - I just gave up trying to make it look the way I wanted.
  - With PowerShell, I have much better access to the data and formatting the data to get the results I wanted: “2.58 TB free” and I could test the value and easily display a different message.

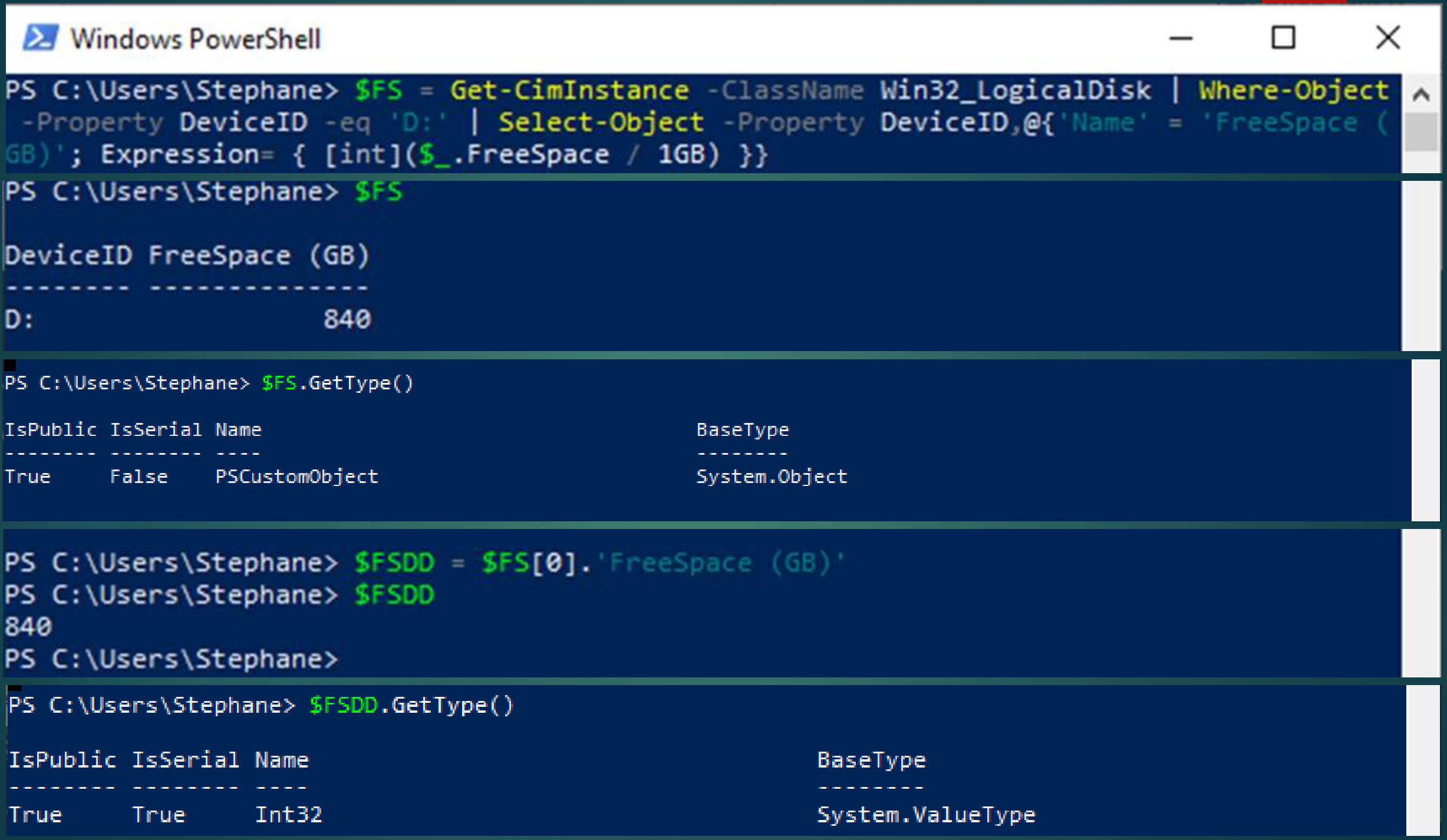
# So, I am making the switch to PowerShell

- My next automation requirement will be done using PowerShell.
- I have been reading many books on PowerShell and learning how powerful are the data manipulation and obtaining data from Windows settings/status.
- The piping of data, i.e. using the output of one command and send it to another, is taken to another level in PowerShell.
- The filtering of the output of a cmdlet to get only the data you are looking for is very powerful.
- The ability to document and create standard help information in any new cmdlet is very useful.
- The learning curve for me is to learn (and remember) what cmdlet does what especially than most of then have parameters.

# Example: Get Free space for a specific drive

- I want the free space on drive D: expressed in GB in a variable so that I can test it and display a different message based on the result of the test.
- Like in any programming language, there is always many ways to do something.
- In the next slides, I will walk you through my process to get that information.


Windows PowerShell
PS C:\Users\Stephane> Get-CimInstance -ClassName Win32_LogicalDisk
DeviceID DriveType ProviderName VolumeName Size FreeSpace
C: 3 OS 465702100992 314628374528
D: 3 DATA 2000381014016 901943570432
F: 5
S: 3 Personal Data 10700713984 5917372416
PS C:\Users\Stephane> Get-CimInstance -ClassName Win32_LogicalDisk | Where-Object -Property DeviceID -eq 'D:'
DeviceID DriveType ProviderName VolumeName Size FreeSpace
D: 3 DATA 2000381014016 901943570432
Getting closer, need to convert the FreeSpace into FreeSpace (GB)
PS C:\Users\Stephane>


Windows PowerShell
PS C:\Users\Stephane> $FS = Get-CimInstance -ClassName Win32_LogicalDisk | Where-Object -Property DeviceID -eq 'D:' | Select-Object -Property DeviceID,@{'Name' = 'FreeSpace (GB)'; Expression= { [int]($_.FreeSpace / 1GB) }}
PS C:\Users\Stephane> $FS
DeviceID FreeSpace (GB)
-------- --------------
D: 840
PS C:\Users\Stephane> $FS.GetType()
IsPublic IsSerial Name BaseType
-------- -------- ---- --------
True False PSCustomObject System.Object
PS C:\Users\Stephane> $FSDD = $FS[0].'FreeSpace (GB)'
PS C:\Users\Stephane> $FSDD
840
PS C:\Users\Stephane>
PS C:\Users\Stephane> $FSDD.GetType()
IsPublic IsSerial Name BaseType
-------- -------- ---- --------
True True Int32 System.ValueType


Any
Questions

# Next steps

- If there is enough interest, a PowerShell Special Topic Group recuring zoom meeting could be setup which will provide a firm grounding in using PowerShell.
- The topics and frequency of the meetings would be discussed amongst those indicating an interest.
- To reduce the work load, and if there is available expertise, volunteers could present a PowerShell topic to the rest of the group.
- If you are interested in participating in the PowerShell Special Topic Group, forward your name to the OPCUG Suggestion Box email with the subject "PowerShell Special Topic Group" by February 1st.