



John Savill's
PowerShell Master Class




© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

1

Who am I?




 @NTFAQGuy

- MCSE NT 4 through 2016, Azure, CISSP, ITIL v3
- Author of the Windows FAQ
- Latest book "Mastering Server 2016 Hyper-V"
- Speaker at Tech Ed, Windows Connections and Ignite
- Large number of Pluralsight courses
- <https://savilltech.com/>
- <http://www.youtube.com/NTFAQGuy>

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

2

Agenda




- Because this is online I will be updating it as required
- Key module themes:
 - PowerShell Fundamentals
 - Remote Management with PowerShell
 - Connecting commands together and mastering objects
 - Creating a PowerShell script
 - Advanced scripting techniques
 - Parsing data and working with objects
 - Desired State Configuration
 - PowerShell Workflows, Azure Automation and Azure Functions
 - The best of the rest

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

3

PowerShell Fundamentals




Before PowerShell
PowerShell Interfaces
Structure of PowerShell Cmdlets and Modules

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

4

The World Without PowerShell




- Many administrators are used to the command prompt
- The command prompt has some built-in commands such as dir and can call separate executables such as ipconfig
- Batch files can be created to perform a fixed set of tasks
- Some pipelining (|) is possible but the output from commands is a text string

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

5

WMI, VBScript, JScript




- There are many scripting languages available which provide a richer language than available in batch files
- Each has its own format and different levels of capability
- Different syntax needed between scripting languages and command line so requires learning multiple systems and environments
- Not highly extensible

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

6

Enter PowerShell




- Formally codename Monad a loooong time ago
- Provides an environment for both executing live commands and creating powerful scripts
- Built on the .NET framework (and now .NET Core) and is focused around objects enabling rich sequences to be created which can perform complex tasks
- Uses a common syntax and common verb-noun pair command naming for cmdlets

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

7

Simpler Language and Standards Based




- Thousands of cmdlets so simplicity is important
- PowerShell is focused on "least cognitive distance" between what you want to do and the actual command
- <verb>-<noun>
- New core cmdlets related to new functionality and also changes to existing cmdlets to match simpler syntax and more features (e.g. Get-Childitem)
- PowerShell can manage non-Windows such as Linux as WSMAN and CIM (WMI2) are the default protocols for management
- PowerShell Core utilizes SSH in addition to WSMAN
- Remote Management enabled by default

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

8

PowerShell Availability




- PowerShell v2 is available for
 - Windows Server 2008 SP1 +
 - Windows Vista SP 2+
 - Windows XP SP 2 +
- Pre-installed in Windows Server 2008 R2 and Windows 7
- .NET Framework 2.0 SP1 required
- .NET Framework 3.5 SP1 required for ISE

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

9

PowerShell v3



- Standard in Windows Server 2012 and Windows 8
- Also available as part of Windows Management Framework (WMF) 3.0
 - <http://www.microsoft.com/en-us/download/details.aspx?id=34595>
- Available for Windows Server 2008 SP2, 2008 R2 SP1 and Windows 7 SP1
- Can be installed side-by-side with PowerShell v2
- Consider this the absolute minimum version

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

10

PowerShell v4




- Standard in Windows Server 2012 R2 and Windows 8.1
- Also available as part of Windows Management Framework (WMF) 4.0
 - <http://www.microsoft.com/en-us/download/details.aspx?id=40855>
- Available for Windows Server 2008 R2 SP1, Windows Server 2012 and Windows 7 SP1
- Still have version 2 available

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

11

PowerShell v5



- New cmdlets such as Get-ItemPropertyValue (to look at property values directly).
- Windows PowerShell Integrated Scripting Environment (ISE) updates
- Windows PowerShell Desired State Configuration (DSC) performance and optimizations (and bug fixes)
- Network Switch cmdlets (in the NetworkSwitch module), which allow management of switches that are Windows Server 2012 R2 certified to have management related to global configuration, VLAN configuration, and Layer 2 port configurations performed.
- OneGet, which allows the discovery and installation of packages from around the web, including services such as Chocolatey
- Part of WMF 5.0

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

12

PowerShell v5.1

- Included in Windows Server 2016 and Windows 10
- Also part of WMF 5.1
- Available for Windows 7, Windows 2008 R2 and above
- Requires .NET Framework 4.5
- Minor changes
- Available at <https://www.microsoft.com/en-us/download/details.aspx?id=54616>

John Savill's

PowerShell Network

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

13

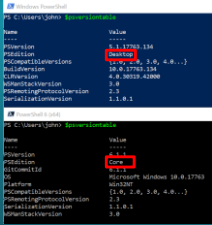
PowerShell Core [6]

- Originally part of Nano Server but now the future of PowerShell
- PowerShell Core runs on .NET Core
- This enables cross-platform usage
- Most existing PowerShell works on PowerShell Core
- Many existing modules updated with Core compatibility
- Can co-exist with Windows PowerShell
- Has it's own module folder

John Savill's

PowerShell Network

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author



14

Installing PowerShell Core (and Visual Studio Code)

- PowerShell Core - <https://github.com/powershell/powershell>
- Visual Studio Code - <https://code.visualstudio.com/Download>
- [Git - <https://git-scm.com/download>]
- VSCode and Git can auto-update
- PowerShell Core can be reinstalled or use Chocolatey to check and update as required
- Don't worry too much about all of this yet!

John Savill's

PowerShell Network

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

15

Windows PowerShell vs PowerShell Core

- Windows PowerShell will continue to ship as part of Windows but is unlikely to get new capabilities
- PowerShell Core is the future of PowerShell innovation
- PowerShell Core is available cross-platform
- Expect more functionality and modules to support PowerShell Core
- Where possible make PowerShell Core your primary environment with development using Visual Studio Code
- Windows PowerShell is still available side-by-side to be used if required

John Savill's

PowerShell Network

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

16

Checking your version of PowerShell

- `Get-Host`
- `$PSVersionTable`
- `$psversiontable.PSVersion`

John Savill's

PowerShell Network

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

17

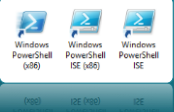
32bit and 64bit PowerShell

- Both 32 and 64-bit versions available
- A 32-bit OS can only have the 32-bit version installed
- A 64-bit OS has both the 64-bit and the 32-bit versions installed
- Use 32-bit when installing a module that requires 32-bit or connecting to a 32-bit computer
- For PowerShell Core you pick the version to install

John Savill's

PowerShell Network

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author



18

Easy Access to PowerShell

- Right click on Start and by default shows Command Prompt and Command Prompt (Admin)
- This can be changed to PowerShell through Taskbar properties
- When using PowerShell sometimes you want to run as Admin
- Should say "Administrator" at top
- Can customize layout, font, opacity (Win 10)
- Can also pin to taskbar

Taskbar and Start Menu Properties

Taskbar: Navigation Taskbars

Current navigation

- Replace Command Prompt with Windows PowerShell in the menu when I right-click the start button or press Windows key+X

Replace Command Prompt with Windows PowerShell in the menu when I right-click the start button or press Windows key+X

On

Taskbar

Run as Administrator

PowerShell (Admin)

Unpin from taskbar

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

19

Using ISE

- The standard PowerShell interface has basic autocomplete
- Integrated Scripting Interface features intellisense, scripting and debugging and help windows
- While learning (and even using) the ISE may be your way to go
- ISE does require WPF so not available on Server Core instances and takes a little longer to start
- Snippets are your friend!
- Only supports Windows PowerShell

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

20

Visual Studio Code

- A free, lightweight, cross-platform editor for many languages
- This is the development environment that is the future and has a great deal of investment
- Add-in available to add full syntax and debug support for PowerShell from <https://marketplace.visualstudio.com/items?itemName=ms-vscode.PowerShell>
- Many extensions available including support multiple for terminals
- Native integration with GIT change control
- Supports Windows PowerShell and PowerShell Core (and a lot more)

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

21

Start with Interactive Shell

- Best way to get started is using PowerShell as an interactive shell
- The shell environment can be customized just like a command prompt window
- Start with navigating round the file system
- Dir works!
- But the switches don't, e.g. `dir /s`, why?
- Intrinsic vs external cmd.exe commands

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

22

Using non-PowerShell in PowerShell

- Difference between commands built into the shell (cmd.exe) and executables
- Search for dir.exe or dir.com, you won't find it. This is part of cmd.exe
- It works in PowerShell but only because PowerShell has an alias that resolves to Get-ChildItem
- Therefore the switches used must be those for Get-ChildItem and NOT dir
- If you call an executable, e.g. IPCONFIG.exe then the existing switches work

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

23

Aliases are your friend

- At least initially
- For most cmd.exe intrinsic commands PowerShell has an alias to the PowerShell cmdlet
- However start to learn the proper PowerShell methods rather than relying on aliases long term
- In scripts use the full commands rather than aliases

Get-Alias

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

24

Copyright 2019 John Savill
Do not redistribute

4

John Savill's
PowerShell Master Class

Navigating with PowerShell

- Regular commands will work but file system is not the only hierarchical storage system
 - Registry
 - Certificate Store
 - Active Directory
 - RDS
 - IIS
- These can also be navigated and configured like a file system

```
Get-PSDrive
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

25

John Savill's
PowerShell Master Class

Tab Support

- Can use tab character for:
 - Cmldet completion
 - Cycling through parameters
 - Cycling through valid values for parameters (where the parameter has a set of valid values)
 - Cycling through files and folders

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

26

John Savill's
PowerShell Master Class

PowerShell Modules

- Modules are loaded which contain PowerShell functionalities such as cmdlets
- `Get-module` to see those loaded
- `Get-module -listavailable` to see all available
- `Import-module <module>` to add into PowerShell instance
- `Get-Command -Module <module>` to list commands in a module

```
Get-Command -Module Hyper-V | Select  
-Unique Noun | Sort Noun
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

27

John Savill's
PowerShell Master Class

Module Versions and Updating

- Modules go through updates
- Check version with
 - `(Get-Module <module name>).Version`
- Closer integration with repositories like GitHub
 - This enables initial installation and update
 - `Install-Module Az`
 - `Update-Module Az`

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

28

John Savill's
PowerShell Master Class

Getting Help

- PowerShell has great help
- `Get-Help <cmdlet>`
- Also available
 - `-full`
 - `-detailed`
 - `-examples`
 - `-online` (to display in web)
 - `-showwindow`
- `Get-Command -Module <module>`
- `Get-Command -Noun <noun>`

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

29

John Savill's
PowerShell Master Class

Updatable Help

- Help for cmdlets is pulled from the Internet and installed into correct location for module (language specific)
 - `Update-Help` (run elevated)
 - `Save-Help` (save to a folder and use this with above Update-Help cmdlet)
- `Get-Help` will automatically pull down latest help
- `Get-Help <cmdlet> -Online` for web based help information
 - `Get-Help New-VHD -Online`
 - `(Get-Command New-VHD).HelpUri`


Updating help for module Microsoft.PowerShell.Management

Downloading help content...

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

30

Coming up Next



Connecting Commands Together and Mastering Objects

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

31

Connecting Commands Together and Mastering Objects


2

PowerShell Pipeline
Passing and Using Objects
Tricks With Objects

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

32

Semicolon




- Do not have to terminate commands
- You can if you want
- If you want to run multiple commands on one line that are not using objects down the pipe use ;
- <command 1>;<command 2>

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

33

Keeping the objects



- PowerShell does not convert output to text but rather maintains the objects
- This enables easy chaining of commands


```
Get-VMNetworkAdapter <VM Name> |
Set-VMNetworkAdapter -AllowTeaming On
```
- Normal commands can be used with PowerShell

```
ipconfig | select-string -pattern 255
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

34

The Pipeline




- PowerShell commands within a session keeps the output as objects
- These objects can be passed along the pipeline to be manipulated by later commands
- Can check the type of an object

```
<var>.GetType()
<var>.GetType().fullname
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

35

Default Pipeline Output



- Typically at the end of the pipeline the data is sent to screen
- There are various different Out verb cmdlets
- The default is **Out-Default**
- This directs to **Out-Host**
- Other targets such as
 - **Out-Printer**
 - **Out-GridView** (this is cool and combine with **-passthru** to select then perform an action)
 - **Out-Null** to suppress

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

36

Command > file.txt

- In a command session the > character can be used to send the output to a file
 - `Tasklist > procs.txt`
- This also works in PowerShell
 - `Get-Process > procs.txt`
- Reality is > is a shortcut taking advantage of the pipeline
 - `Get-Process | Out-File procs.txt`

John Savill's PowerShell Resources

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

37

Export Data

- Simple use of pipeline is exporting of the data
- Initially data may be output to screen with a default formatting
 - `Get-Process | Format-Table/Format-List`
- Exporting out to other formats such as a CSV file and XML file
 - `get-process | Export-csv c:\stuff\proc.csv`
 - `get-process | Export-clsxml c:\stuff\proc.xml`
- When importing back keeps the objects(kind of!)
 - `Import-Csv C:\stuff\proc.csv | where {$_.name -eq "notepad"} | Stop-Process`

John Savill's PowerShell Resources

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

38

Limiting Objects Returned

- Can use `Sort-Object`
- Then combine with `-descending,-ascending`
- Send to `Select-Object` with `-first n,-last n`
- For example to get the newest 5 processes use:
 - `Get-Process | Sort-Object -Descending -Property StartTime | Select-Object -First 5`
- Can use `Measure-Object` to count output
 - `Get-Process | Measure-Object`
 - `Get-Process | Measure-Object WS -Sum`

John Savill's PowerShell Resources

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

39

Comparing Objects

- Often you may wish to compare data
- `Compare-Object` can be used to compare two sets of data
- You can select how the compare of objects works
 - `-Property <name>` - finds differences in instances of objects in the results
 - Without this any change would be displayed

John Savill's PowerShell Resources

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

40

Advanced Outputs

- There are various `ConvertTo` cmdlets for advanced output
 - HTML
 - JSON
 - CSV
 - XML
 - Secure String
- Would combine with `Out-File` to output the formatted output to a file

John Savill's PowerShell Resources

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

41

Doing Things with Objects

- Outputting to file is great but the real power is using cmdlets together and keeping the objects
- Cmdlets need to be able to accept as input objects output from the previous cmdlet in the pipeline
- For example (don't run this!)
 - `Get-Process | Stop-Process`
- The above would be very bad
 - `Get-Process -Name notepad | Stop-Process`
- Can use `-passthru` with some commands to allow objects to continue down the pipeline
 - `get-aduser bruce | Disable-ADAccount -PassThru`

John Savill's PowerShell Resources

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

42

Copyright 2019 John Savill
Do not redistribute

7

-confirm and -whatif

- These are parameters that can be added to other commands
- -confirm will prompt before performing any actions
- -whatif will display what would happen without actually performing any real actions
 - Very useful for testing purposes

```
get-aduser -filter * | Remove-ADUser -whatif
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

43

More -Confirm

- Every cmdlet has an impact level
- If the impact level is equal or higher than the confirmation preference
 - \$confirmPreference
- To avoid confirmation when not desired add -Confirm:\$false

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

44

Find Members of an Object

- **Get-Member!**
- This will list all the properties, events and methods for the object passed
- This is an easy way to find out what you can do with objects

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

45

What is \$_ ???

- Objects are passed from one command to another but sometimes an attribute of a passed object needs to be used
- \$ _ represents the current pipeline object and lets you access a property of a piped in object instead of the entire object

```
Get-Process | Where-Object { $_.name -eq "notepad" } `
| Stop-Process
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

46

Simple Syntax Example

```
Get-Process | where { $_.HandleCount -gt 900 }
Get-Process | where { $psitem.HandleCount -gt 900 }
Get-Process | where HandleCount -gt 900
```

- Can use gps instead of Get-Process (just an alias)
- I prefer to still use \$ _ but I'm just set in my ways

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

47

Advanced \$ _ use


```
$UnattendFile = "C:\stuff\unattend.xml"
$xml = [xml](gc $UnattendFile)
$child = $xml.CreateElement("TimeZone", $xml.unattend.NamespaceURI)
$child.InnerXml = "Central Standard Time"
$null = $xml.unattend.settings.Where({ $_.Pass -eq 'oobeSystem' }).component.appendchild($child)
$xml.Save($UnattendFile)
```

- Or
\$resources = Get-AzResourceProvider -ProviderNamespace Microsoft.Compute
\$resources.ResourceTypes.Where({ \$_.ResourceTypeName -eq 'virtualMachines' }).Locations

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

49

Coming up Next



Remote Management with PowerShell

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

50

Remote Management with PowerShell

3

Remote Connectivity

Connecting to Remote Machines


Executing PowerShell Remotely

Executing PowerShell on Multiple Machines

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

51

Remote Management




- The goal of PowerShell is to provide centralized management of an entire IT environment
- Being able to perform actions on remote machines is critical
- Being able to perform actions on MULTIPLE remote machines is even more critical

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

52

RPC, no thank you!




- Traditionally Remote Procedure Call (RPC) was used for remote interaction
- This uses MANY different ports and is very hard to configure on a firewall
- Normally the firewall is disabled
- Microsoft has moving away from RPC

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

53

Introducing WinRM




- WS-MAN is a standard protocol for remote management using HTTP and HTTPS and WinRM is the Windows implementation
- It does not use port 80 or 443 so does not conflict (well it did with WinRM 1.1 and earlier)
- WinRM uses HTTP port 5985. HTTPS uses 5986 (when used)
- In production HTTPS should be used so content is encrypted (or use IPSEC)

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

54

Enabling WinRM



- WinRM is installed by default on Windows 2008 R2 and above (which had PowerShell v2)
- WinRM is ENABLED by default on Windows 2012 which uses WinRM for basically everything management
- WinRM must be enabled on client operating systems and other server operating systems in elevated PowerShell
 - Enable-PSRemoting
 - Use Group Policy
- Get-PSSessionConfiguration

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

55

Enable-PSRemoting

```
PS C:\WINDOWS\system32> Enable-PSRemoting
winRM has been updated to receive requests.
winRM service type changed successfully.
winRM service started.

winRM has been updated for remote management.
Created a winRM listener on HTTP://* to accept WS-Man requests to any IP on this machine.
winRM firewall exception enabled.
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

56

Invoke-Command

- Can be one-to-many
- Can invoke commands to an existing session
- When used with a computer a session is created then terminated meaning state is not persisted
- Data is serialized into XML when sent back then deserialized back to objects in local session
- These objects are not linked to the original anymore so perform all actions on the remote as needed and optimize what is returned as serialization/deserialization is computationally expensive
- By default only 32 concurrent connections. Can be changed with -ThrottleLimit <new number>

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

57

Sessions and Enter-PSSession

- One-to-one
- Can create sessions in advance then enter the session or just create on demand
- Can import-modules FROM a session and then execute them, known as implicit remoting
- Can add a prefix so can load module on multiple remote sessions

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

58

A Quick Word on Compatibility with PowerShell Core

- PowerShell Core continues to have more cmdlets available however there are still some only available for Windows PowerShell
- The Windows PowerShell Compatibility module works in a similar way to implicit remoting
- Modules are loaded by creating a session to the local machine then creating function definitions that redirect to the local session that uses Windows PowerShell

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

59

Alternate Endpoints

- Get-PSSessionConfiguration shows available endpoints
- JEA leverages this heavily for constrained endpoints which contain subsets of privileges and cmdlets
- Add -ConfigurationName <endpoint configuration> to run against those endpoints

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

60

Example

```
Enable-WSManCredSSP -Role "Server" -Force
New-PSSessionConfigurationFile -ModulesToImport OneTech, ActiveDirectory,
Microsoft.PowerShell.Utility
-VisibleCmdlets ("*OneTech*", "AD*", "format*", "get-help")
-VisibleFunctions ("TabExpansion2") -VisibleAliases ("exit", "ft", "fl") -
LanguageMode ConstrainedLanguage
-VisibleProviders FileSystem
-SessionType 'RestrictedRemoteServer' -Path 'c:\dcmonly.pssc'
Register-PSSessionConfiguration -Name "DCMs" -Path C:\dcmonly.pssc -StartupScript
C:\PSData\DCMProd.ps1
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

61

Robust Connections

- If a session is disconnected PowerShell will try on its own for up to 4 minutes to re-establish the connection
- Can disconnect from sessions then reconnect and the state is maintained on the target

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

62

Authentication

- The account you use to run commands is used on the remote machine even though WinRM runs as local system (which it always should)
- Your credential can be used on the remote machine but cannot be hopped to another machine from that machine
- If hopping is required then CredSSP can be used
- Once enabled you must use –Authentication CredSSP –Credential <cred> when using CredSSP
- Kerberos resource-based delegation is another option that will work if machines are in a trusted domain/forest
- Also certificate authentication may be an option when using SSL

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

63

Connection Outside Trusted Domains

- Mutual authentication works within domains or trusted domains with no extra work
- For connections to machines outside trusted domain, not in a domain or not using the machines AD name (e.g. DNS alias, IP address) mutual authentication is not possible without extra work
- For the above scenarios either:
 - Enable HTTPS connections using CA issued SSL certificates on the remote machine
 - Add the name/IP to the WinRM TrustedHosts list

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

64

Trusted Host

```
cd wsman;  
cd .\localhost\Client  
ls  
Set-Item -Path .\TrustedHosts -Value ''  
ls  
Set-Item -Path .\TrustedHosts -Value '10.7.173.193'  
Set-Item -Path .\TrustedHosts -Value '*.savilltech.net'  
Set-Item -Path .\TrustedHosts -Value '*' #very bad!  
Set-Item -Path .\TrustedHosts -Value ''
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

65

Troubleshooting

- First try a telnet to <node> 5985
- Applications and Service logs – Microsoft – Windows – PowerShell-Operational and will see information about connections
- Windows Remote Management – Operational would see problems about the connections if there were issues

```
Import-module PSDiagnostics  
Get-command -module PSDiagnostics  
Enable-pswsmancombinedtrace  
  
Invoke-command -ComputerName comp1 -ScriptBlock {get-process}  
Disable-PSWSManCombinedTrace
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

66

HTTPS with SSL

- Uses a certificate to confirm the identity of the target server
- A separate SSL encrypted connection and port that must be enabled
- Avoid using makecert for the certificate (as this defeats many of the benefits)
- Use a cert from internal trusted CA or external CA
- Name in certificate must match the name you will connect to the server as

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

67

Connecting using SSL

- Add -UseSSL (and a credential when outside of domain or just want to!)
- -SkipCACheck - Don't worry if SSL is from a trusted CA (would need this if used makecert for example)
- -SkipCNCheck - Don't worry if the certificate does not match the name you are connecting to
- Outside of a lab both of those options are good ways to get in trouble as they defeat mutual authentication

```
$cred=get-credential
Enter-PSsession savdalfs01.savilltech.net `
-credential $cred -UseSSL
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

68


A Quick Word on Non-Windows and SSH

- PowerShell Core embraces cross-platform
- As does Visual Studio Code enabling a complete, consistent experience
- While WinRM is commonly used it is also possible to use SSH (for both Linux and Windows)
- For Windows need OpenSSH installed (part of 1809+) and for Linux/MacOS use the appropriate SSH remoting feature
 - `invoke-command -hostname 10.0.1.40 -Username john -ScriptBlock {get-host}`

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

69

Coming up Next



Creating a PowerShell Script

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

70

Getting Ready for DevOps

- Setting up the DevOps environment
- Basic usage of VS Code
- Using GIT with GitHub/Azure Repos

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

71

Getting Ready

- Make sure you have the following installed
 - PowerShell Core - <https://github.com/powershell/powershell>
 - Visual Studio Code - <https://code.visualstudio.com/Download>
 - Git - <https://git-scm.com/downloads>
- You can accept the defaults for the installs but during the Git installation select to use VS Code for the editor for Git configuration
- Create a taskbar shortcut for VS Code (you can also launch via running 'code')
- Create a GitHub account (it's free) and I recommend enabling MFA

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

72

VS Code Extensions

- Extensions bring additional functionality to VS Code
- Some of these will automatically be recommended based on the type of file you are working with
- The first extension I install is Settings Sync which synchronizes settings to all my VS Code instances across machines via a private GIST (a list hosted in GitHub)
- Now the changes you make will upload and download manually or set to auto-download and auto-upload

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

73

Enabling GIT with VS Code

- VS Code natively supports GIT integration
- You need to tell VS Code where GIT is installed
- Ctrl + Shift + P opens the command palette (you will use this a lot)
- Type >preferences: open settings (JSON)
- In the User Settings area add:
"git.path": "C:\\Program Files\\Git\\bin\\git.exe"
- Setup GIT with your name and email
 - `git config --global user.name "John Savill"`
 - `git config --global user.email "john@savilltech.com"`
 - `git config --global --list`

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

74

Other Key Extensions

- PowerShell
- Shell launcher
- GitLens
- Each have various configurations
- You can see my configurations in the `VSCodeSettingsSample.json` file

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

75

Work in Folders

- While you can open files for integration with change control open a folder and then from there open files you wish to work on
- This will enable the folder to be enabled for change control (local and remote)

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

76

Enabling a Project (Folder) for Git

- Git is a distributed version control system (VCS)
- All of the data related to the repository instance is stored in a .git subfolder
- It can be completely local without any remote repository
- Commonly the local copy is a clone of a remote that can be used in isolation and then commits pushed to the remote origin
- Many people can work together on a project on their own clones of the repository then push changes when ready but often this is frequently done (Continuous Integration)

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

77

Populating a Remote Repository from Local

- One option would be to create the remote repository first, pull the down locally via clone then populate
- Changes can then be pushed
- Other option if you have a local repository you can add the remote as the origin and push

```
git remote add origin <URL>
git remote -v
git push -u origin master
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

78

Working with Git in VS Code

- VS Code can perform most common GIT actions through its Source Control interface
- Includes staging and commits
- Interact with remote repositories
- Can utilize branches
- Merge available via the Command Palette (Git: Merge Branch...)

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

79

Using Branches

- Any repository has at least one primary branch, typically master
- Additional branches can be created, often as part of development efforts which can later be merged back into the master when complete
- Using branches keeps the master clean and enables development streams to be kept separate
- It is easy to switch between branches, for example I may be working on a new feature in a branch and a fix in a different branch
- Once branches are merged into master the branch is often deleted

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

80

Public Repositories

- GitHub and Azure Repos are two major ones
- Both offer private and public repositories
- Both can be utilized via Git
- While you can clone without authentication when performing a pull it will prompt for credentials which may include MFA (recommended)

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

81

Useful Keyboard Shortcuts

Ctrl Shift P	Command Palette
Ctrl `	Terminal
Ctrl F2	Change all occurrences
Ctrl B	Toggle sidebar
Ctrl Alt Up/Down	Multi-line cursor (can also Alt + click with mouse)
F8	Execute selected code
Ctrl F (F3)	Find (Find Next)
F9	Toggle breakpoint
F5	Start/Continue
F11 (Shift F11)	Step Into (Step Out)
Shift F5	Stop

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

82

Creating a PowerShell Script

4

Basics of PowerShell Script
Best Practices for Script Authoring
Interacting With Users

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

83

Region Example

- Regions enable code to be “folded” within ISE and VS Code

```
#region Simplification example
Get-Process | where {$_.HandleCount -gt 900}
Get-Process | where {$psitem.HandleCount -gt 900}
Get-Process | where HandleCount -gt 900
#endregion
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

84

Signed Scripts

- Get-ExecutionPolicy
- By default the environment is restricted
- Set-ExecutionPolicy used to change to
 - RemoteSigned (trusted publisher for external scripts)
 - AllSigned (trusted publisher for any script including your own)
 - Unrestricted (run anything and not recommended!)
- For our testing set to RemoteSigned
- RemoteSigned is the default (instead of Restricted) from 2012 R2 and above

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

85

First Script

- A text file saved with a ps1 extension

```
Write-Output "Hello World"

Write-Host ("Hello " + $env:USERNAME)
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

86

Execute Script

- From PowerShell
 - `& .\script.ps1`
- From cmd.exe
 - `Powershell [-noexit] "& <path>\script.ps1"`
 - `pwsh -command "& .\adddnumbers.ps1"`

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

87

Write-Out vs Write-Host

- Remember the various Out targets and the pipeline
- These two cmdlets are not equal but initially appear to have the same result
- The goal of PowerShell is to support data to pass along the PowerShell Objectflow engine
- `Write-Host` outputs data directly to the host and not along the pipeline
- `Write-Output` has its output continue down the pipeline
- Always use `Write-Output` as `Write-Host` limits the handling of data

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

88

Write-Output Working

```
graph LR
    Start --> A[A Cmdlet]
    A --> B[B Cmdlet]
    B --> C[C Cmdlet]
    C --> D[D Cmdlet]
    D --> Engine[Engine]
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

89

Why have Write-Host

- Because it can do pretty output formatting not possible with `Write-Output`

```
Write-Host "You are looking " -NoNewline
Write-Host "AWESOME" -ForegroundColor Red `
-BackgroundColor Yellow -NoNewline
Write-Host " today"
```

You are looking **AWESOME** today

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

90

Difference between ' and "

- The difference is minimal
- Generally use single quotes
- Variables in double quotes are replaced with their values but not in single quotes
- Double quotes enable delimiting within a string
- Can use escape characters with double quotes

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

91

Prompting the User

- Read-Host is an easy way to get input
- Possible to add -AsSecureString to avoid displaying to screen and storing securely

```
$name = Read-Host "Who are you?"
$pass = Read-Host "What's your password?" -AsSecureString
[Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::SecureStringToBSTR($pass))
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

92

First Script With Input

```
Param(
    [string]$computername='savazuusdc01')
Get-WmiObject -class win32_computersystem `
    -ComputerName $computername |
    fl numberofprocessors,totalphysicalmemory
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

93

Script With Mandatory Input

- If user does not type in the parameter they will be prompted for it!
- No special code needed

```
Param(
    [Parameter(Mandatory=$true)][string]$computername)
Get-CimInstance -class win32_computersystem `
    -ComputerName $computername |
    fl numberofprocessors,totalphysicalmemory
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

94

Script With Multiple Input

```
Param(
    [Parameter(Mandatory=$true)][string[]]$computers)
foreach ($computername in $computers)
{
    Get-CimInstance -class win32_computersystem `
        -ComputerName $computername |
        fl numberofprocessors,totalphysicalmemory
}
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

95

Script With Different Output

```
Param(
    [Parameter(Mandatory=$true)][string[]]$computers)
foreach ($computername in $computers)
{
    $win32CSOut = Get-CimInstance -ClassName win32_computersystem -ComputerName $computername
    $win32OSOut = Get-CimInstance -ClassName win32_operatingsystem -ComputerName $computername

    $paramout = @(
        'ComputerName'=$computername;
        'Memory'=$win32CSOut.totalphysicalmemory;
        'Free Memory'=$win32OSOut.freephysicalmemory;
        'Procs'=$win32CSOut.numberofprocessors;
        'Version'=$win32OSOut.version)

    $outobj = New-Object -TypeName PSObject -Property $paramout
    Write-Output $outobj
}
```

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

96

Coming up Next

Advanced Scripting Techniques

© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

97