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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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 Server2008 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

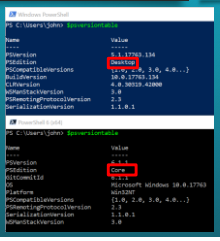
- Included in Windows Server 2016 and Windows 10
- Also part of WMF 5.1
- Available for Windows 7, Windows 2008 R2 and above
- Requires .NET Framework 4.5
- Minor changes
- Available at https://www.microsoft.com/en-us/download/details.aspx?id=54616

© 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



© 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/downloads]
- 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!

© 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

© 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`

© 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

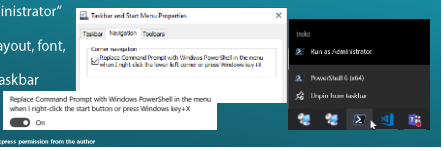
© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

18

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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



© 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?
- Intrisic 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
https://savilltech.com @NTFAQGuy

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

Tab Support

- Can use tab character for:
  - Cmdlet 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

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

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

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

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`

© 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

Copyright 2019 John Savill, Do not redistribute

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
```

© 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-clixml c:\stuff\proc.xml
```

- When importing back keeps the objects(kind of!)

```
Import-Csv C:\stuff\proc.csv | where {$_.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

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
```

© 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

© 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

© 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 –passthruwith some commands to allow objects to continue down the pipeline

```
get-aduser bruce | Disable-ADAccount -PassThru
```

© 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
https://savilltech.com @NTFAQGuy

-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 setting user will be prompted for confirmation
  - $confirmpreference
- To avoid confirmation when not desired add –Confirm:$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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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 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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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 theWinRM 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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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 two
- 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

Copyright 2019 John Savill, Do not redistribute

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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Using Branches

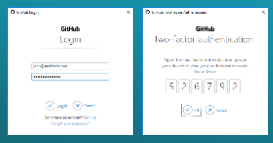
- Any repository has at least one primary branch, typically master
- Additional branches can be created, often as part of development efforts which can later me merged back into the master when complete
- Using branches keeps the master clean and enables development streams to be kept separate
- It is 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 put 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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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

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

87

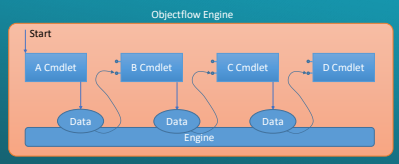
Write-Out vs Write-Host

- 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



© 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

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

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

Advanced Scripting Techniques 5

Passing Data to Scripts
Returning Data
Enabling Script Reuse
Debugging

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

98

Lazy with Parameters

- You do not actually have to define parameters
- Arguments past to a script are captured into default variables
- Prefer to use the named parameters

```
Write-Host 'Number of arguments was :'
($args.Length)
Write-Output 'and they were:'
foreach($arg in $args)
{
    Write-Output $arg
}
```

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

99

Types of Parameter

- There are really 3 types of parameter with PowerShell

Formal Name	Type	Example
Switch Parameter	Switches	Get-ChildItem–Recurse
Parameter	Option	Get-ChildItem–Filter *.txt
Positional Parameter	Arguments	Get-ChildItem*.txt

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

100

Using Multiple Parameters

- Simply add multiple parameters and the variable names are the parameter names
- Implicit positions are possible but have to be used in the correct order
- Alternatively use the parameter names

```
Param([String]$Greeting,[String]$Name)
Write-Host $Greeting $Name
```

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

101

Can Explicitly Define Position

- This means the order defined does not have to match that in the parameters
- Note once you use explicit positional any without a position must be entered by name

```
Param([Parameter(Mandatory=$True,Position=2)][String]$Name,
 [Parameter(Mandatory=$True,Position=1)][String]$Greeting)
Write-Host $Greeting $Name
```

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

102

Use a Different Name

- Possible to alias parameters to use alternate names
- Can still use original name

```
Param([Parameter(Mandatory=$True,Position=2)]
 [Alias("Friend")][String]$Name,
 [Parameter(Mandatory=$True,Position=1)][String]$Greeting)
Write-Host $Greeting $Name
```

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

103

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Switches

- Switches are set to true or false with no additional value
- If only the switch is present it means its value is true
- If switch is not passed its value is false

```
Param(
[Parameter(Mandatory=$true)][string]$computername,[switch]$showlogprocs)
if($showlogprocs)
{
    Get-CimInstance -class win32_computersystem -ComputerName $computername `
    | fl NumberOfLogicalProcessors,totalphysicalmemory
}
else
{
    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

104

Accepting Pipeline Input

```
[cmdletbinding()]
Param(
[Parameter(ValuefromPipeline=$true,Mandatory=$true)][string[]]
$computers)
```

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

105

Enabling Help

```
<#
.SYNOPSIS
Gets information about passed servers
.DESCRIPTION
Gets information about passed servers using WMI
.PARAMETER computer
Names of computers to scan
.EXAMPLE
CompInfo.ps1 host1, host2
Not very interesting
#>
Param(
[Parameter(Mandatory=$true)][string[]]$computers)
foreach ($computername in $computers)
{
```

```
Get-Help .\CompInfo.ps1
```

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

106

Troubleshooting

- Write-Verbose
  - Shown if add –verbose to command (providing have the [CmdletBinding()])
- Write-Debug
  - Shown if add –debug to command
  - Can also suspend, look at the current state then resume by typing 'exit'

```
Write-Verbose "Querying $computername"
$win32CSOut = Get-CimInstance -class win32_computersystem -ComputerName $computername
$win32OSOut = Get-CimInstance -class win32_operatingsystem -ComputerName $computername
Write-Debug "Finished querying $computername"
```

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

107

Try and Catch to catch error

```
$lookinggood = $true
try
{
    $win32CSOut = Get-WmiObject -class win32_computersystem `
        -ComputerName $computername -ErrorAction Stop
}
catch
{
    "Something bad: $_"
    $lookinggood = $false
}
if($lookinggood)
{
```

The $lookinggood is just so we don't try the next command if we know we have an error!

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

108

Creating your own modules

- Saving a file as PSM1 instead of PS1 makes it a PowerShell Script Module
- If you save in folder My Documents\WindowsPowerShell\Modules\<foldername same as PSM1 name\<file>.psm1 it will be available, e.g.
- My Documents\WindowsPowerShell\Modules\CompInfo\CompInfo.psm1
- PSModulePath set when PowerShell starts with your My Documents sub folder added automatically

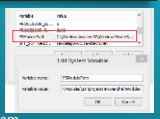
© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

109

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

PSModulePath

- System Environment Variable can be changed
- Look at variable will see My Documents folder added
- Can add your own static paths. For example I have added d:\projects\PowerShell\Module to my system
- These are different for PowerShell Core which does not inherit currently from Windows PowerShell



```
C:\Users\john\Documents\WindowsPowerShell\Modules;
```

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

110

More on Functions

- Functions are units of code that can be called elsewhere in the code or the PowerShell session
- Functions can accept input and output data
- Similar fashion to the parameters in the script file and can use $args or named parameters
- There is also a default $input for all data sent to it
- Anything sent to output is returned from function (NOT write-host)

```
function first3 {$input | Select-Object -First 3}
get-process | first3
```

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

111

Making it a function

- Add function declaration to start of the script with name of the command
- Embed all code in { }

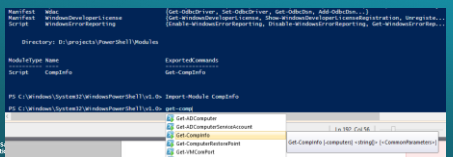
```
function Get-CompInfo
{
    <#
    .SYNOPSIS
    Gets information about passed servers


    }
}
```

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

112

Importing your module

- If saved correctly will now be visible with get-module –listavailable
- Import as normal
- Use your command. Autocomplete even works!



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

113

Final Script .PS1

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

114

Final Script .PSM1

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

115

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Signing your Script

- Require a code signing certificate
  - Use Internal CA
  - Public trusted CA
  - Makecert (although like in all cases not useful outside local testing)
- Use PowerShell to then sign your script

```
$cert = @(gci cert:\currentuser\my -codesigning)[0]
Set-AuthenticodeSignature signme.ps1 $cert
```

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

116

Coming up Next

Parsing Data and Working with Objects

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

117

Parsing Data and Working with Objects 6

Working With Data
Formatting Data

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

118

Storing Credentials

- Many times a credential is needed
- This can be gained online using
- $cred=Get-Credential
- You could also create via code but means plain text!

```
$user = "administrator"
$password = 'Pa55word'
$securePassword = ConvertTo-SecureString $password `
    -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ($user,
$securePassword)
```

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

119

Storing an encrypted version

- Create a string

```
$encryptedPassword = ConvertFrom-SecureString (ConvertTo-
SecureString -AsPlainText -Force "Password123")
```

- Make a note of the output and then use that in the script

```
$securepassword = ConvertTo-SecureString "<the huge value from
previous command>"
```

- This is still easy to reverse but is harder to quickly write down!

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

120

Store in another file

- Create the file

```
$credpath = "c:\temp\MyCredential.xml"
New-Object
System.Management.Automation.PSCredential("john@savilltech.com",
(ConvertTo-SecureString -AsPlainText -Force "Password123")) |
Export-CliXml $credpath
```

- Then in your script

```
$cred = import-clixml -path $credpath
```

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

121

Azure Key Vault

- Azure makes PowerShell better and vice versa
- Any of the previous options run the risk of your secrets being in the code which then finds its way on some kind of repository. Very bad day!
- Azure Key Vault can store various types of data including secrets which can be protected with ACLs
- Instead of stored credentials anyway in files, store them in Azure Key Vault

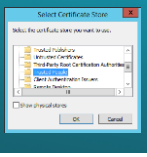
```
Get-Credential <cred> | Save-AzPSCredential `
    -ResourceGroupName <RG> -VaultName <vault>
$cred = Get-AzPSCredential -VaultName <vault> -Name <cred>
```

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

122

Certificate Authentication

- Certificates can be used for authentication
- A User certificate is required for the account which is then exported
- The public key is imported into the target server local computers store under Trusted People
- Create a mapping between the cert and a local user
- Pass certificate thumbprint when connecting



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

123

Variable types

- There are many types of variables
- Common ones:
  - [string] - Unicode string of characters
  - [char] – Single 16-bit Unicode character
  - [byte] – 8-bit character
  - [int] – 32-bit int
  - [long] – 64-bit int
  - [decimal] – 128-bit decimal value
  - [bool] – true or false ($true, $false)
  - [DateTime] – date and time

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

124

Casting and Checking

- Variable types can be automatically set based on the value set

```
$number=42
$boolset=$true
$stringval="hello"
$charval='a' (not what we want!)
```

- Can force a variable to be a certain type

```
[char]$newchar= 'a'
```

- View the type of a variable with $var.GetType()
- Can also test:

```
42 -is [int]
```

- Conversions:

```
$number = [int]
$number.ToString()
```

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

125

String Fun

- Does a string contain another string
- You do NOT want –contains (this checks if a collection of objects includes an object)
- Use –Like

```
$string1 = "the quick brown fox jumped over the lazy dog"
$string1 -like "*fox*"
$string2 = $string1 + " who was not amused"
```

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

126

Fun with Time

- `$today=Get-Date`
- `$today | Select-Object -ExpandProperty DayOfWeek`
- `[DateTime]::ParseExact("02-25-2011","MM-dd-yyyy",[System.Globalization.CultureInfo]::InvariantCulture)`
- `$christmas=[system.datetime]"25 December 2019"`
- `($christmas - $today).Days`
- `$today.AddDays(-60)`
- `$a = new-object system.globalization.datetimeformatinfo`
- `$a.DayNames`

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

127

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Variable Scope

- Variables can have a scope
- By default a variable has local scope and is visible in the scope its created and any child scopes
- Possible to specify alternate scopes

Scope	Meaning
Local	Current scope and child scopes
Global	Accessible in all scopes
Script	Available within the script scope only
Private	Cannot be seen outside the current scope (not children)

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

128

```
function testscope()
{
    write-output $defvar
    write-output $global:globvar
    write-output $script:scripvar
    write-output $private:privvar
    $funcvar = "function"
    $private:funcpriv = "funcpriv"
    $global:funcglobal = "globfunc"
}

$defvar = "default/local" #default
get-variable defvar -scope local
$global:globvar = "global"
$script:scripvar = "script"
$private:privvar = "private"
testscope
$funcvar
$funcglobal #this should be visible
```

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

129

Variables with Invoke-Command

- Something interesting happens using variables and Invoke-Command

```
$message = "Message to John"
Invoke-Command -ComputerName savdalfs01 -ScriptBlock {write-host $message}
```

- The Invoke-Command creates a new session that does not have the variables. Instead the variables must be passed

```
$ScriptBlockContent = {
    param ($MessageToWrite)
    Write-Host $MessageToWrite }
Invoke-Command -ComputerName savdalfs01 -ScriptBlock $ScriptBlockContent -ArgumentList $message
#or
Invoke-Command -ComputerName savdalfs01 -ScriptBlock {Write-Output $args} -ArgumentList $message
```

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

130

Using $using

- PowerShell 3 introduces a new way to utilize local scope variables with Invoke-Command; $using

```
Invoke-Command -ComputerName savazuusscdc01 `
  -ScriptBlock {Write-Output $using:message}
```

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

131

Hash Tables

- Useful to store name/value pairs

```
$favthings =
@{"Julie"="Sushi";"Ben"="Trains";"Abby"="Princess";"Kevin"="Minecraft"}

$favthings.Add("John","Crab Cakes")

$favthings.Set_Item("John","Steak")

$favthings.Get_Item("Abby")
```

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

132

Using PSObject

- PSObject enables you to create a custom object
- There is also System.Object but custom properties tend to get lost with .Net functions which is bad
- With a PSObject add new properties and a value for the new property
- Can add as many properties and values as desired
- Useful way to return multiple pieces of data when only a single object can be returned

```
$cusobj = New-Object PSObject
Add-Member -InputObject $cusobj -MemberType NoteProperty `
  -Name greeting -Value "Hello"
```

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

133

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Create PSObject from Hash

```
$favthings =
@{"Julie"="Sushi";"Ben"="Trains";"Abby"="Princess";"Kevin"="Min
ecraft"}
$favobj = New-Object PSObject -Property $favthings

#In PowerShell v3 can skip a step
$favobj2 =
[PSCustomObject]@{"Julie"="Sushi";"Ben"="Trains";"Abby"="Prince
ss";"Kevin"="Minecraft"}
```

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

134

Foreach-Object

- Sometimes the limitations of $_ and the pipeline (even with ;) means we need something more
- Typical to save results to an array and then examine each element in the array

```
$names = @("Julie","Abby","Ben","Kevin")

$names | ForEach-Object -Process { Write-Output $_}

$names | ForEach -Process { Write-Output $_}

$names | ForEach { Write-Output $_}

$names | % { Write-Output $_}
```

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

135

ForEach ≠ ForEach

- ForEach is an alias for ForEach-Object when data is pipelined to it
- When data is not pipelined to it then it's a PowerShell statement that behaves differently
- ForEach loads all objects into a collection first so uses more memory but may enable better optimization of data loading and execute faster
- ForEach-Object continues data down the pipeline while ForEach does not

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

136

Getting Property Values

- <object>.<property> typically gives the value you need
  - $town.name
  - $_.id
- Sometimes this does not work as expected

```
$samacctname = "John"
Get-ADUser $samacctname  -Properties mail
Get-ADUser $samacctname  -Properties mail | select-object mail
Get-ADUser $samacctname  -Properties mail | select-object mail | get-member
Get-ADUser $samacctname  -Properties mail | select-object -ExpandProperty mail | get-
member
Get-ADUser $samacctname  -Properties mail | select-object -ExpandProperty mail
```

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

137

Coming up Next

Desired State Configuration

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

138

Desired State Configuration 7

Understanding DSC
Basics of DSC
Authoring DSC Scripts
Using DSC

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

139

Why is DSC needed?

- Applying configuration to an OS can be complex
  - OS configurations
  - Roles and features to install
  - Role/feature configuration
  - Applications to install and configure
  - Files, registry etc requirements
- We don't want specific OS images with app configurations
- Vanilla OS then apply configuration post creation
- Possible to achieve this through a PowerShell script during setup but what if someone changes it
- How to detect drift?

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

140

DSC

- PowerShell v4 brings Desired Configuration Management completing the PowerShell initial vision
- Initially 12 providers based around objects such as registry, files, environment, scripts, users, groups and so on
- Many additional providers available
- Declarative rather than imperative
- Not currently available in PowerShell Core

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

141

Regular Imperative PowerShell

- You have to handle all error-handling, checking if may have already run parts of it

```
Import-Module ServerManager

#Check and install Web Server Role if not installed
If (-not (Get-WindowsFeature "Web-Server").Installed)
{
    try {
        Add-WindowsFeature Web-Server
    }
    catch {
        Write-Error $_
    }
}
```

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

142

Declarative PowerShell DSC

- Only focus on the desired final state

```
# Install the IIS role
WindowsFeature IIS
{
    Ensure          = "Present"
    Name            = "Web-Server"
}
```

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

143

Phases of using DSC

- Authoring
  - Creating the DSC file
  - Compile to a MOF file which is DMTF standard
- Staging
  - The DSC is staged on the target using one of the two models (push or pull)
- "Make it so"
  - Configuration is applied to the local configuration store which contains all DSC
  - The configuration is then parsed and the relevant WMI providers implement the required changes
  - Idempotent so can be repeatedly applied without causing damage

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

144

Models of using DSC

- DSC application can be used in one of two modes
- Push
  - Configurations are applied to a servers via Start-DscConfiguration immediately
- Pull
  - Configurations are centrally stored
  - Special service, DSC-Service, runs on Windows Server 2012 R2
  - Servers are configured for pull mode which consists of a unique configuration ID (GUID) and the DSC server to pull from
  - Servers poll and pull their configuration from the central DSC service

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

145

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Configuration Resources

- A number of built-in configuration resources including
  - Archive Resource
  - Environment Resource
  - File Resource
  - Group Resource
  - Log Resource
  - Package Resource
  - Registry Resource
  - Script Resource
  - Service Resource
  - User Resource
  - Windows Feature Resource
  - Windows Process Resource

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

146

DSC Resource Kit

- Additional configuration resources are constantly being added
- These were released as DSC Resource Kits
- Now available via PowerShell Gallery, https://www.powershellgallery.com/packages
- Large number resources including SQL Server, Exchange, Remote Desktop Service, Hyper-V, System Center and more!
- Search for tag DSCResourceKit

```
Find-Module -tag dscresourcekit | Install-Module
Get-DscResource
```

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

147

DSC Configuration Format

- Outermost Configuration block can contain passed parameters then
  - Zero or more Node blocks that define:
    - One or more Resource blocks

```
Configuration WebConfig
{
   param([string[]]$computerName="localhost") #optional parameters

   Node $computerName #zero or more node blocks
   {
      WindowsFeature WebServer #one or more resource blocks
      {
          Ensure = "Present" # To uninstall the role, set Ensure to "Absent"
          Name = "Web-Server"
      }
   }
}
```

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

148

Basic DSC Usage

- A DSC file must be compiled to MOF format by executing the name of the configuration
- The configuration is then applied (in the push model) using
  - `Start-DscConfiguration -Wait -Verbose ` `-Path .\MyTestFileConfig`
- The configuration can be checked using `Get-DscConfiguration`
- To look for drift use `Test-DscConfiguration`
- To see the actual drift add `-Detailed` to `Test-DscConfiguration`. Needs WMF 5

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

149

Compliance of DSC

- Often you will need to understand the deployment status of DSC
- Also the configuration drift knowledge centrally stored is very valuable
- A solution is being created that will be available as open source on GitHub and the PowerShell Gallery
  - DSC Environment Analyzer (DSC-EA)

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

150

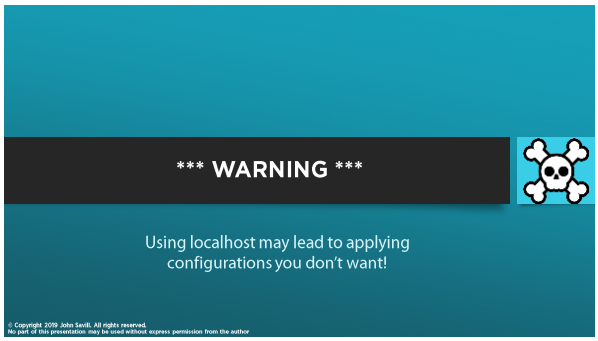
DSC for any Machine

- Typically a DSC file specifies a specific target node for configuration
  - `Node websrv1`
    `{ ... }`
- For configuration to apply to any server use localhost
  - `Node localhost`
    `{ ... }`
- Alternatively use a parameter to pass in the node name that is set at compilation time
  - `param ([string[]]$NodeName = 'localhost' )`
    `Node $NodeName`
    `{ … }`

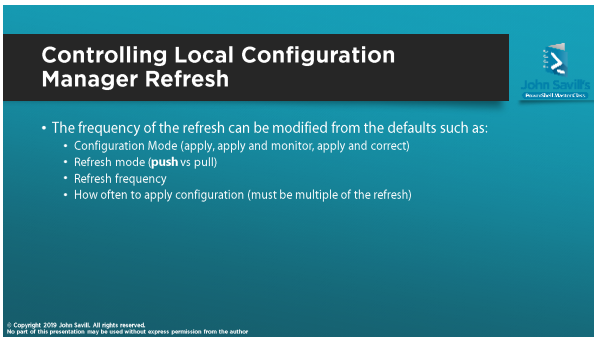
© Copyright 2019 John Savill. All rights reserved.
No part of this presentation may be used without express permission from the author

151

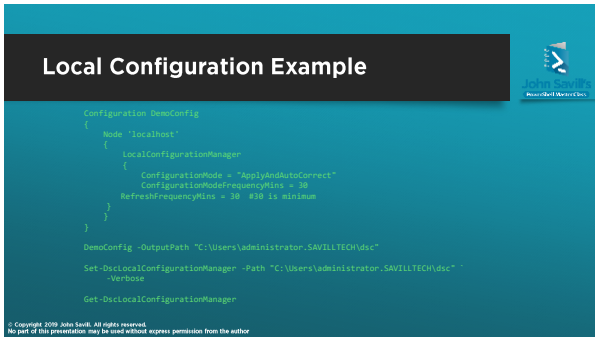
Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy



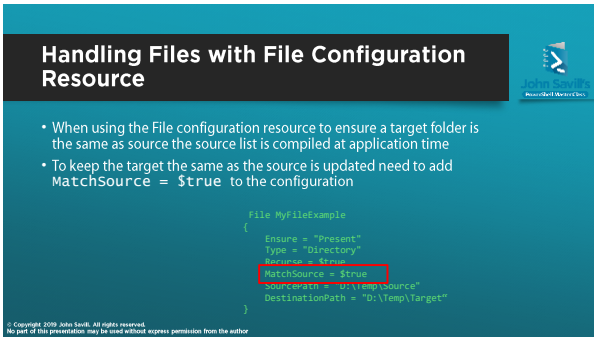
152



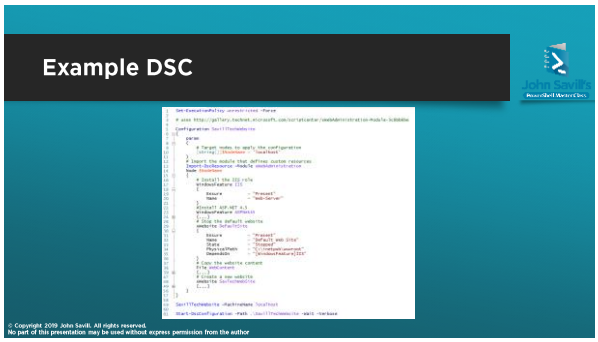
153



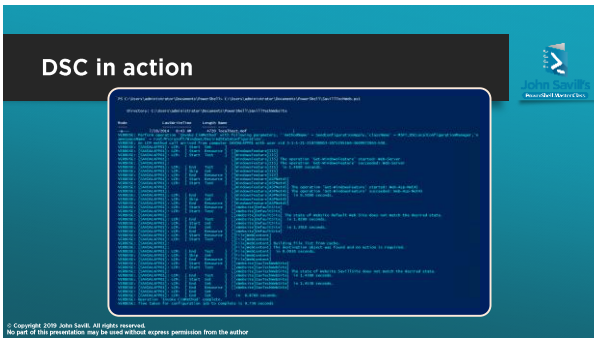
154



155



156



157

Copyright 2019 John Savill, Do not redistribute

PowerShell DSC use in Azure

- Can use DSC with an Azure VM like any other OS instance
- Can also be used when provisioning a VM
- Can use VM extensions to trigger via PowerShell for a running VM

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

158

Tweaked to call from Azure

```
Write-Verbose "Starting the node configuration"

#Set-ExecutionPolicy unrestricted -Force #This is set automatically when called by Azure script extension

Write-Verbose "Enabling Remoting"

Enable-PSRemoting -Force

# uses http://gallery.technet.microsoft.com/scriptcenter/xWebAdministration-Module-3c8bb6be

Write-Verbose "Copying content from Azure Files to PowerShell Modules path"

#Copy the modules to the folder
$username = "savillmasterazurestore"
$password = convertto-securestring -String "R6yw==" -AsPlainText -Force
$cred = new-object -typename System.Management.Automation.PSCredential `
         -argumentlist $username, $password

New-PSDrive -Name T -PSProvider FileSystem -Root "\\savillmasterazurestore.file.core.windows.net\tools" -Credential $cred
Copy-Item -Path "T:\DSC\xWebAdministration" -Destination $env:ProgramFiles\WindowsPowerShell\Modules -Recurse
Remove-PSDrive -Name T

Write-Verbose "Applying DSC Configuration"

$ConfigString=...
Invoke-Expression $ConfigString


SavillTechWebsite -MachineName localhost

Start-DscConfiguration -Path .\SavillTechWebsite -Wait -Verbose
```

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

159

Troubleshooting

- Make sure you enable remoting in your script
- Do not require any interaction
- Write-Verbose in your script to aid troubleshooting
- Give it time. The script is run asynchronously so the VM may show as provisioned but your script may still be running in the background even after you connect to it
- View logs within the VM at C:\WindowsAzure\Logs\Plugins\Microsoft.Compute.CustomScriptExtension\<version, e.g. 1.1>
  - CustomScriptHandler.log – Your file execution

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

160

One Configuration per Machine

- Mostly
- PowerShell v5 introduces multiple partial configurations that can be combined to a single MOF prior to application
- Outside of this you can only have one configuration per machine

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

161

Coming up Next

Automation Technologies

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

162

Automation Technologies 8

Understanding Workflows
Using Workflows
Utilizing Azure Automation
Utilizing Azure Functions

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

163

Problems With Long Running Scripts

- A script that finishes in a second is unlikely to get interrupted
- A script that take 5, 10, 60 minutes has a far greater chance to be interrupted during execution
- If that happens how does it resume
- Can you just rerun the whole script?
  - Most actions are not idempotent
  - Would need checks throughout the script to check if an action has already been done
  - This is painful and time consuming

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

164

Workflows

- Allow complex sequences of actions across multiple machines that may take a long time
- Can be paused and resumed (providing has checkpoints configured or state may be used if –force used)
- Additional parameters for control
  - List of machines
  - Credentials to run workflow
  - Quotas related to run time
  - Retry count if connection problems

```
Workflow MyWorkflow {Write-Output "Hello from Workflow!"}
MyWorkflow
```

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

165

Checkpoints

- Enable the saving of state and data at specific points
  - CheckPoint-Workflow
- Contains the current values of variables, output and the point in the workflow
- Latest checkpoint is used to recover the workflow job if execution is interrupted
- Can add as many checkpoints as desired
- Place at key points but consider each checkpoint uses additional resources
- Is also a workflow level checkpoint in addition to activity level checkpoints

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

166

Example Workflow with Checkpoint

```
Workflow LongWorkflow
{
  Write-Output -InputObject "Loading some information..."
  Start-Sleep -Seconds 10
  CheckPoint-Workflow
  Write-Output -InputObject "Performing process list..."
  Get-process -PSPersist $true #this adds checkpoint
  Start-Sleep -Seconds 10
  CheckPoint-Workflow
  Write-Output -InputObject "Cleaning up..."
  Start-Sleep -Seconds 10
}
LongWorkflow –AsJob –JobName LongWF –PSPersist $true
```

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

167

Workflow and Parallel tasks

- Workflows introduce the ability to run activities in parallel on one or more machines
- New keywords specific to workflow and parallel activities
  - Parallel
  - Foreach -parallel

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

168

Example of Parallel

- Terrible output but proves the point!

```
workflow parralleltest
{
    parallel
    {
        get-process -Name w*
        get-process -Name s*
        get-service -name x*
        get-eventlog -LogName Application -newest 10
    }
}
parralleltest
```

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

169

Running Concurrent Actions Example

- Normally actions will run in series but there are ways to force parallel execution with workflow

```
Workflow Invoke-ParallelLiveMigrate
{
    $VMLIST = get-vm blank1,blank2,blank3,blank4,blank5,blank6
    ForEach -Parallel ($VM in $VMLIST)
    {
        Move-VM -Name $VM.Name -DestinationHost savdalhv02
    }
}

Invoke-ParallelLiveMigrate
```

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

170

Parameters Can Be Passed to Workflow

- Just like a regular function or script

```
workflow compparam
{
   param([string[]]$computers)
   foreach -parallel ($computer in $computers)
   {
        Get-CimInstance -Class Win32_OperatingSystem -PSComputerName $computer
        Get-CimInstance -Class win32_ComputerSystem -PSComputerName $computer
   }
}
compparam -computers savdalfs01,savdalsql01,savdalcm01
```

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

171

Forcing Actions to Sequence

- If there are activities within a parallel block that must run in sequence use the Sequence {} block

```
workflow compparamseq
{
   param([string[]]$computers)
   foreach -parallel ($computer in $computers)
   {
        sequence
        {
            Get-WmiObject -Class Win32_OperatingSystem -PSComputerName $computer
            Get-WmiObject -Class win32_ComputerSystem -PSComputerName $computer
        }
   }
}
compparamseq -computers savdalfs01,savdalsql01,savdalcm01
```

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

172

Better Example of Sequence

```
workflow parallelseqtest
{
    parallel
    {
        sequence
        {
            get-process -Name w*
            get-process -Name s*
        }
        get-service -name x*
        get-eventlog -LogName Application -newest 10
    }
}
parallelseqtest
```

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

173

Loading an Assembly in a Workflow

- Cannot use regular methods such as
  - Add-Type
  - [Reflection.Assembly]::LoadWithPartialName(<assembley>)
- Instead this needs to be done using an InlineScript

```
$NewPassword = InlineScript {
  Add-Type -AssemblyName System.Web
  [System.Web.Security.Membership]::GeneratePassword(16,2)
}
```

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

174

Workflow Restrictions

- Number of other restrictions where InlineScript will save you sometimes
  - Cannot use method invocation ($string.ToLower())
    - Use InlineScript or convert prior to the work
  - Cannot use a case insensitive switch statement
    - Use InlineScript or convert to one case first and then use -CaseSensitive(which will need InlineScript for the case conversion!))
  - No $global scope
  - Some cmdlets do not function (especially ones requiring interactive environment)
    - Use InlineScript

```
$msgtest = InlineScript {($using:msgtest).ToUpper()}
```

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

175

Copyright 2019 John Savill, Do not redistribute
https://savilltech.com @NTFAQGuy

Azure Automation

- The ability to run PowerShell workflows and scripts in Azure (runbooks)
- Enables a schedule to be defined when the automations should run
- Includes an output windows to view execution if required
- Not limited to Azure services. Can connect to any service that has an Internet endpoint
- Do not use cmdlets that are based on interaction, e.g. FT
- Textual and Graphical Modes
- Can create assets including credentials
- Consider use cases
  - Stop/start VMs at a certain time
  - Create new VMs based on scenarios
  - Patching VMs without downtime

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

177

Azure Automation to On-premises

- You could open up firewall ports to machines from Azure but not ideal
- One option is to create a single VM on-premises that is exposed to Azure Automation and through that VM all on-premises actions are performed via
- Better option is to leverage the Azure Automation Hybrid Runbook Worker
  - OS instances run the OMS Microsoft Management Agent
  - Hybrid Runbook Worker will be pulled down
  - Active via the Add-HybridRunbookWorker cmdlet

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

179

Azure Functions

- One of the Azure serverless compute offerings
- Supports a number of runtimes including .NET and JavaScript
- Can run PowerShell along with code
- Supports a number of trigger types including a schedule and a REST trigger

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

180

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

181