Installing and Running PowerShell

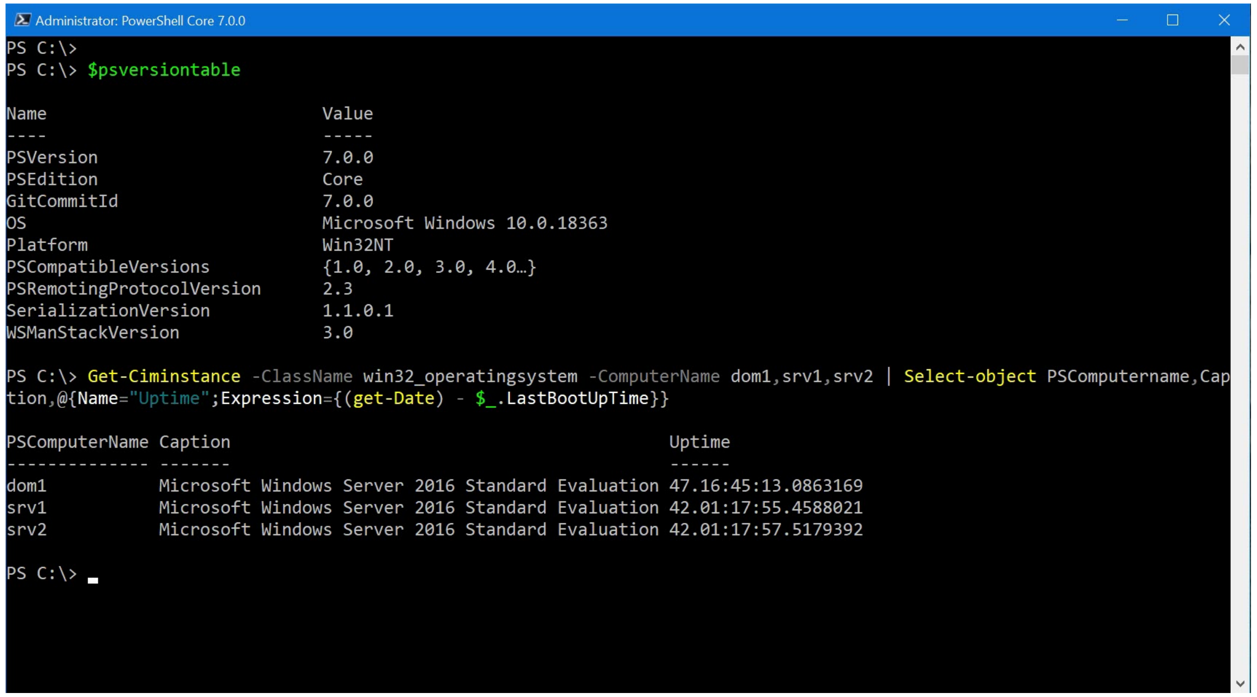
UNDERSTANDING THE ROLE OF POWERSHELL

Jeff Hicks
AUTHOR/TEACHER

@jeffhicks | https://jdhitsolutions.com

Today

PowerShell In Action

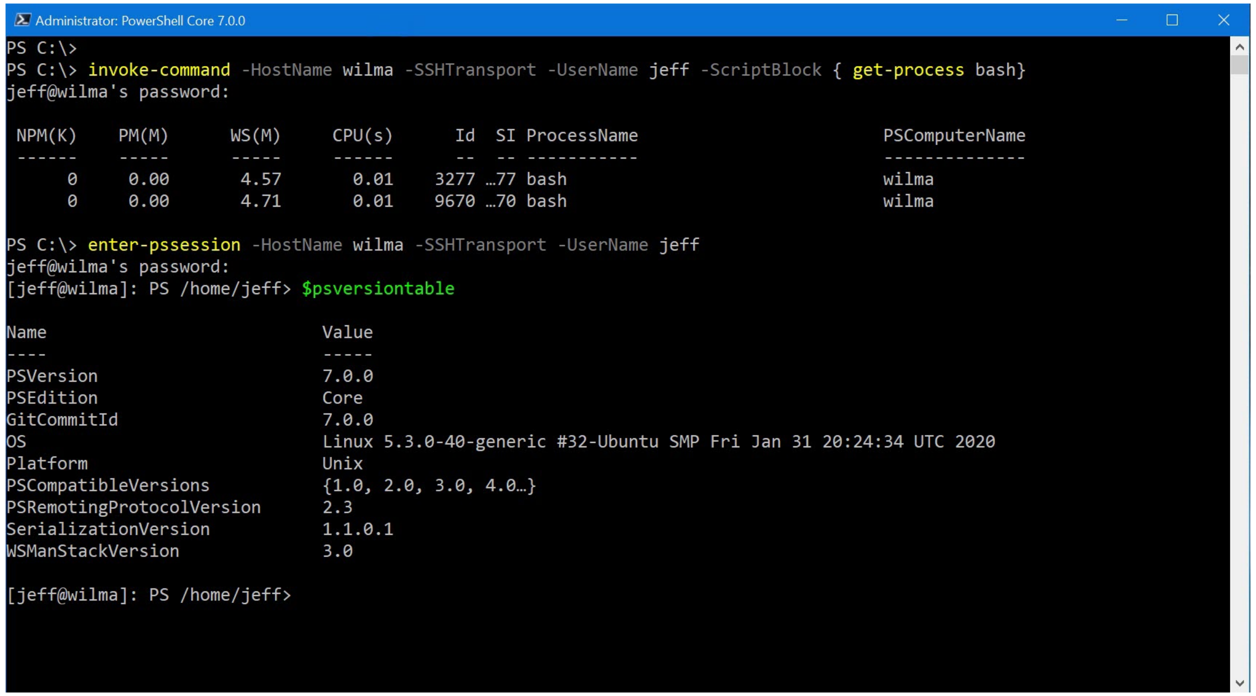

Administrator: PowerShell Core 7.0.0

```
PS C:\>
PS C:\> $psversiontable

Name                           Value
----                           -----
PSVersion                      7.0.0
PSEdition                      Core
GitCommitId                    7.0.0
OS                             Microsoft Windows 10.0.18363
Platform                       Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0

PS C:\> Get-Ciminstance -ClassName win32_operatingsystem -ComputerName dom1,srv1,srv2 | Select-object PSComputername,Cap
tion,@{Name="Uptime";Expression={(get-Date) - $_.LastBootUpTime}}

PSComputerName Caption                                          Uptime
-------------- -------                                          ------
dom1           Microsoft Windows Server 2016 Standard Evaluation 47.16:45:13.0863169
srv1           Microsoft Windows Server 2016 Standard Evaluation 42.01:17:55.4588021
srv2           Microsoft Windows Server 2016 Standard Evaluation 42.01:17:57.5179392

PS C:\>
```

Today

PowerShell In Action

“Not Your Father’s PowerShell”



```
PS C:\>
PS C:\> invoke-command -HostName wilma -SSHTransport -UserName jeff -ScriptBlock { get-process bash}
jeff@wilma's password:

 NPM(K)    PM(M)      WS(M)     CPU(s)      Id  SI ProcessName                       PSComputerName
 ------    -----      -----     ------      --  -- -----------                       --------------
      0     0.00       4.57       0.01    3277 …77 bash                              wilma
      0     0.00       4.71       0.01    9670 …70 bash                              wilma

PS C:\> enter-pssession -HostName wilma -SSHTransport -UserName jeff
jeff@wilma's password:
[jeff@wilma]: PS /home/jeff> $psversiontable

Name                           Value
----                           -----
PSVersion                      7.0.0
PSEdition                      Core
GitCommitId                    7.0.0
OS                             Linux 5.3.0-40-generic #32-Ubuntu SMP Fri Jan 31 20:24:34 UTC 2020
Platform                       Unix
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0

[jeff@wilma]: PS /home/jeff>
```

It feels like only yesterday...

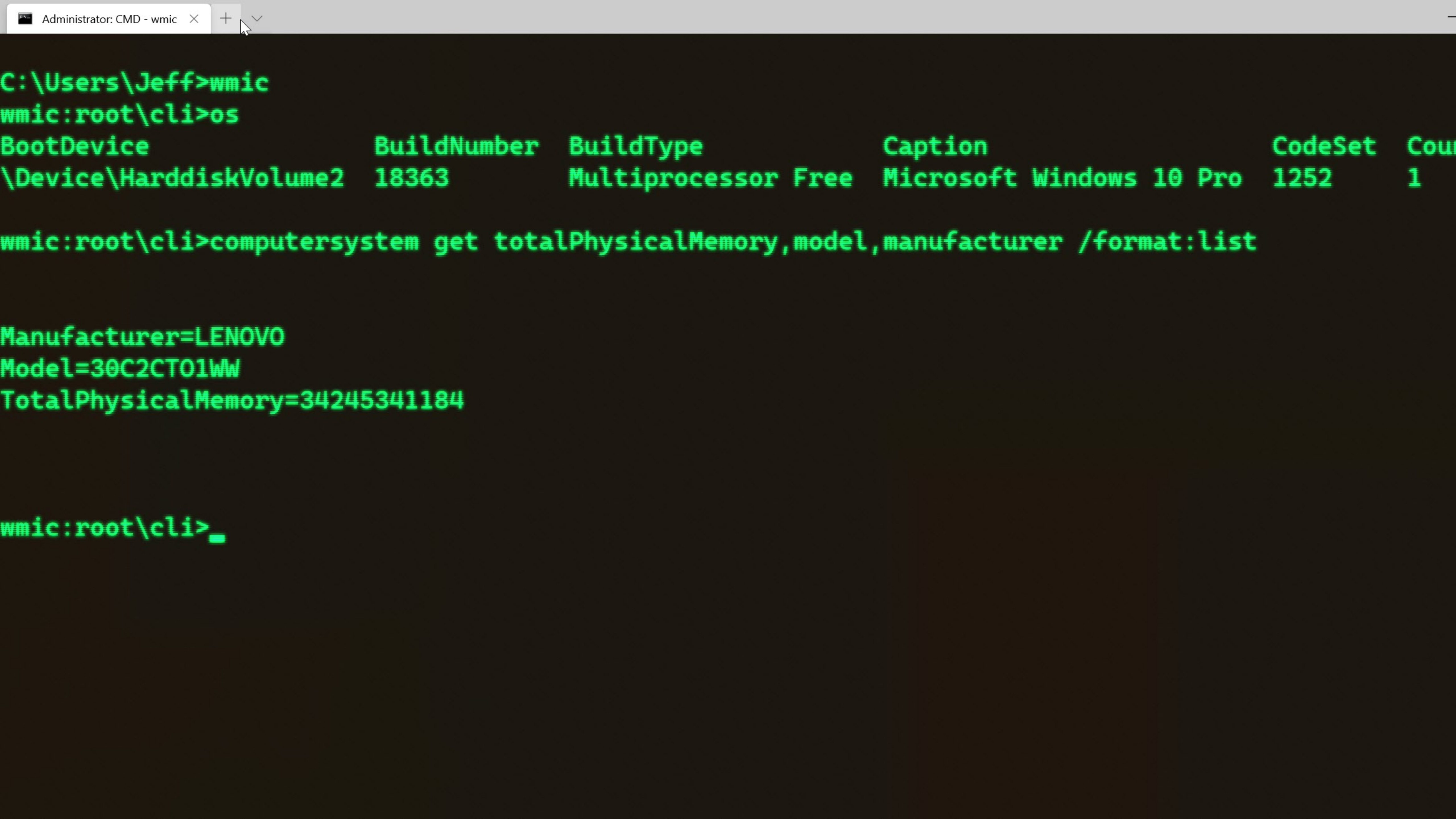
- DOS Batch files
- Resource Kit Tools
- Microsoft Management Console
- VBScript
- Vendor Tools

Graphical tools don’t scale

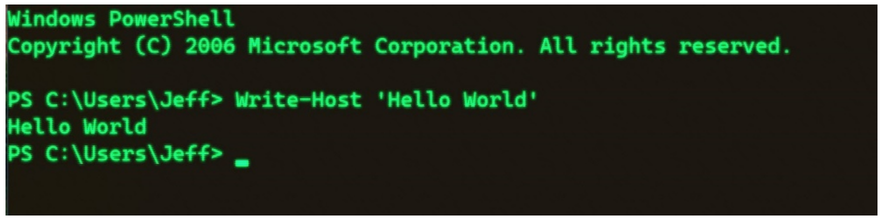
Automation becoming critical

DevOps spreading

True enterprise management is console-based


Administrator: CMD - wmic

```
C:\Users\Jeff>wmic
wmic:root\cli>os
BootDevice                 BuildNumber  BuildType              Caption                       CodeSet   Cou
\Device\HarddiskVolume2    18363        Multiprocessor Free    Microsoft Windows 10 Pro      1252      1

wmic:root\cli>computersystem get totalPhysicalMemory,model,manufacturer /format:list


Manufacturer=LENOVO
Model=30C2CTO1WW
TotalPhysicalMemory=34245341184




wmic:root\cli>_
```



```
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. All rights reserved.

PS C:\Users\Jeff> Write-Host 'Hello World'
Hello World
PS C:\Users\Jeff> _
```

PowerShell is Born

- Code named Monad
- Built on the .NET Framework
- Object-centered
- Interactive management via console
- Easy to learn scripting language

A Brief History of PowerShell

NT 4
- Management Console
- Resource Kit Tools

Windows XP/2000
- VBScript
- Resource Kit Tools

2006
- PowerShell 1.0

2009
- PowerShell 2.0
- Remoting over WsMan

2012
- PowerShell 3.0
- Scripting Improvements

A Brief History of PowerShell

2013
- PowerShell 4
- Windows 8.1/Windows 7 SP1
- Desired State Configuration

Early 2016
- PowerShell 5
- Windows 10

2016
- PowerShell 5.1
- Last Windows Version

2016
- PowerShell Core
- Open source
- First cross-platform

2020
- PowerShell 7.0

The PowerShell Paradigm

No text parsing

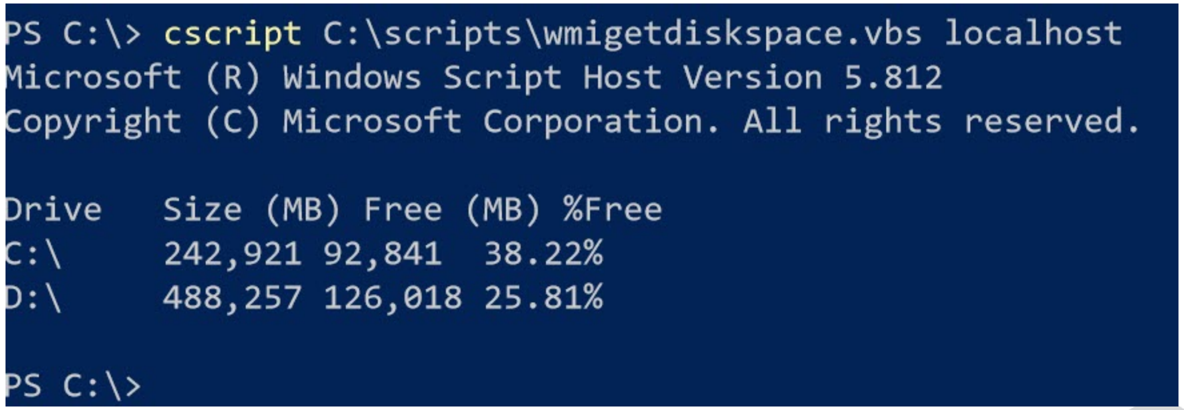
Manipulate objects

In a pipeline

If you can type it at a prompt you can script it

```
Dim objSet, wshell
On Error Resume Next

Set wshell=CreateObject("Wscript.Shell")
strSrv=Trim(wscript.arguments(0))

strQuery = "Select * from win32_logicaldisk where drivetype=3"
Set objSet=GetObject("winmgmts:\\" & strSrv).ExecQuery(strQuery)
if err.number<>0 then
 wshell.popup "Oops! Error connecting to " & UCase(strSrv) &
vbCrlf & "make sure you are using valid " & _
 "credentials." & vbCrlf & "Error: " & err.number & " -
 " & err.description,5,"Disk Check Error",0+48
 wscript.quit
end if

For Each item In objSet
   PerFree=FormatPercent(item.FreeSpace/item.Size,2)
   o=o & item.DeviceID & "\" & VBTAB
   o=o & FormatNumber(item.Size/1048576,0) & Vbtab &
FormatNumber(item.FreeSpace/1048576,0) & Vbtab & PerFree &
Vbcrlf
Next

WScript.Echo "Drive" & Vbtab & "Size (MB) Free (MB) %Free" &
VbCrLf & o

set objSet=Nothing
set wshell=Nothing

wscript.quit
```

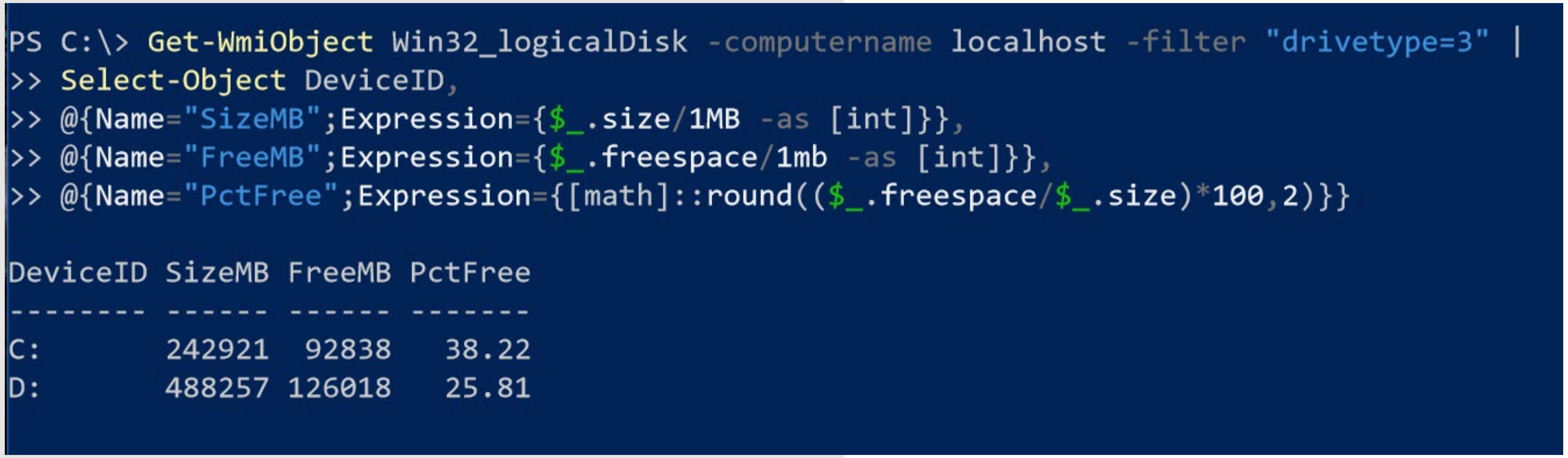
- Old school VBScript
- 20 lines of arcane code
- Parse text output
- Create a script
- Then execute



```
PS C:\> cscript C:\scripts\wmigetdiskspace.vbs localhost
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

Drive   Size (MB) Free (MB) %Free
C:\     242,921 92,841  38.22%
D:\     488,257 126,018 25.81%

PS C:\>
```

```
Get-WmiObject Win32_logicalDisk -computername localhost
-filter "drivetype=3" | Select-Object DeviceID,
@{Name="SizeMB";Expression={$_.size/1MB -as [int]}},
@{Name="FreeMB";Expression={$_.freespace/1mb -as [int]}},
@{Name="PctFree";Expression={[math]::round(($_.freespace/$
_.size)*100,2)}}
```

- A one-line PowerShell command
- No scripting
- Easy to understand
- Could be put into a script file or a function
- Can be run interactively
- Output easily converted or exported



```
PS C:\> Get-WmiObject Win32_logicalDisk -computername localhost -filter "drivetype=3" |
>> Select-Object DeviceID,
>> @{Name="SizeMB";Expression={$_.size/1MB -as [int]}},
>> @{Name="FreeMB";Expression={$_.freespace/1mb -as [int]}},
>> @{Name="PctFree";Expression={[math]::round(($_.freespace/$_.size)*100,2)}}

DeviceID SizeMB FreeMB PctFree
-------- ------ ------ -------
C:       242921  92838   38.22
D:       488257 126018   25.81
```

PowerShell is a management engine

Manage anything from anywhere from any platform

Manage local systems and services

Manage the cloud

Do it for 1 or 100 or 1000

Interactive or Script

PowerShell is a preferred automation language

Scripting provides consistency

Scripting provides documentation

Scripting provides efficiency

```
C:\> for /F %i in (c:\work\clist.txt) do sc %i query bits
```

Old School

- **Using legacy command line tools**
- **Query 100 servers**
- **10 minutes**

```
PS C:\> $list = get-content c:\work\dlist.txt

PS C:\> invoke-command { Get-Service -Name bits }
-computername $list
```

The PowerShell Way

- **27 seconds**
- **One of many PowerShell solutions**
- **More options available to easily convert, export, format or save**

PowerShell is the language of the cloud and the modern datacenter

Jeff's Modern Management Paradigm

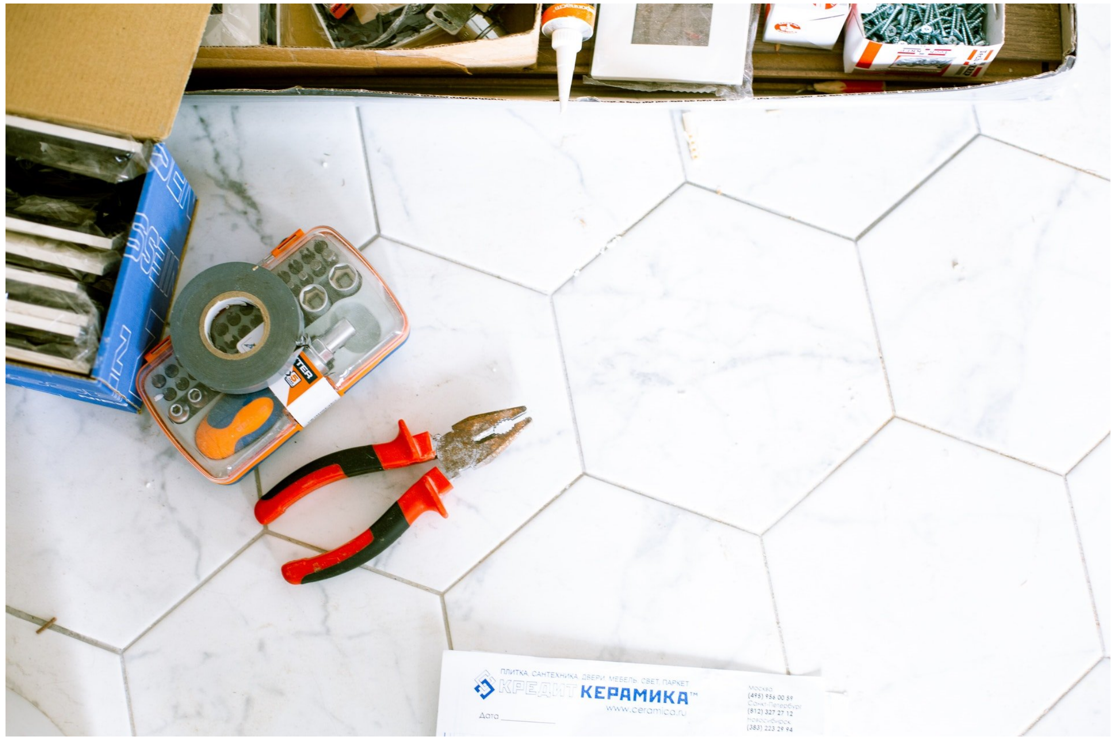
Learn the manual process

Understand the technology

Use PowerShell tools

Automate!

Learn and Leverage One Tool



PowerShell is an enabling tool

Scripts & Modules

Workflows

Desired State Configuration

Just Enough Administration

Windows PowerShell 5.1	PowerShell 7
Proprietary	Open Source
Windows platforms only	Windows, Linux and MacOS
Ships with Windows	Manual install
PowerShell 5.1 is feature complete	Active development
PowerShell ISE	Visual Studio Code
powershell.exe	pwsh.exe

Summary

PowerShell is a primary management tool you should learn

Learn it once and apply it everywhere

The future is cross-platform and PowerShell 7

Windows PowerShell 5.1 remains for legacy or compatibility requirements