

# Powershell for the SQL SERVER DBA



Protect, unlock and optimize your data's value



[www.mikefal.net](http://www.mikefal.net)



Mike Fal

I D E R A  
ACE

**Microsoft**  
**CERTIFIED**  
Solutions Expert

---

Data Platform

```
        $trns = Get-ChildItem $dir -recurse | Where-Object {$_.Name -like "*.trn"} | Sort-Object LastWriteTime
    }
    else{
        $full = Get-ChildItem $dir | Where-Object {$_.Name -like "*.bak"} | Sort-Object LastWriteTime -desc | Select-Object -first 1
        $diff = Get-ChildItem $dir | Where-Object {$_.Name -like "*.df*" } | Sort-Object LastWriteTime -desc | Select-Object -first 1
        $trns = Get-ChildItem $dir | Where-Object {$_.Name -like "*.trn"} | Sort-Object LastWriteTime
    }

    #initialize and process full backup
    $outputfile = Join-Path -Path $outputdir -ChildPath "restore_$database.sql"
    $restore = Get-RestoreObject $database $full
    $hfull = Get-Header $restore $smosrv
    if($database.Length -eq 0)
    {
        $database = $hfull.DatabaseName
        $restore.Database=$database
    }

    $LSNCheck = $hfull.CheckpointLSN
    $files = $restore.ReadFileList($smosrv)
    foreach($file in $files){
        $pfile = $file.PhysicalName
        if($newdata.Length -gt 0 -and $file.Type -eq "D"){
            $pfile=$newdata + $pfile.Substring($pfile.LastIndexOf("\\"))

        }

        if($newdata.Length -gt 0 -and $file.Type -eq "L"){
            $pfile=$newlog + $pfile.Substring($pfile.LastIndexOf("\\"))

        }

        $newfile = New-Object("Microsoft.SqlServer.Management.Smo.RelocateFile") ($file.LogicalName,$pfile)
        $restore.RelocateFiles.Add($newfile) | Out-Null
    }

    $sqlout += "/*****"
    $sqlout += "Restore Database Script Generated $(Get-Date)"
    $sqlout += "Database: '$database'"
    $sqlout += "*****"
    $sqlout += "--FULL RESTORE"
    If($owner){$sqlout += "EXECUTE AS LOGIN = '$owner';"}
    $sqlout += $restore.Script($smosrv)

    #process differential backups
    if($diff -ne $null){
        $restore = Get-RestoreObject $database $diff
        $hdiff = Get-Header $restore $smosrv

        if($hdiff.DatabaseBackupLSN -eq $LSNCheck){
            $sqlout += "--DIFF RESTORE"
            $sqlout += $restore.Script($smosrv)
            $LSNCheck = $hdiff.LastLSN
        }
    else{ ... -- --
```

Don't focus on the code,  
focus on the  
concepts.

Ask questions!

Powershell Basics

Powershell and SQL Server

Re-using Powershell

Practical Examples

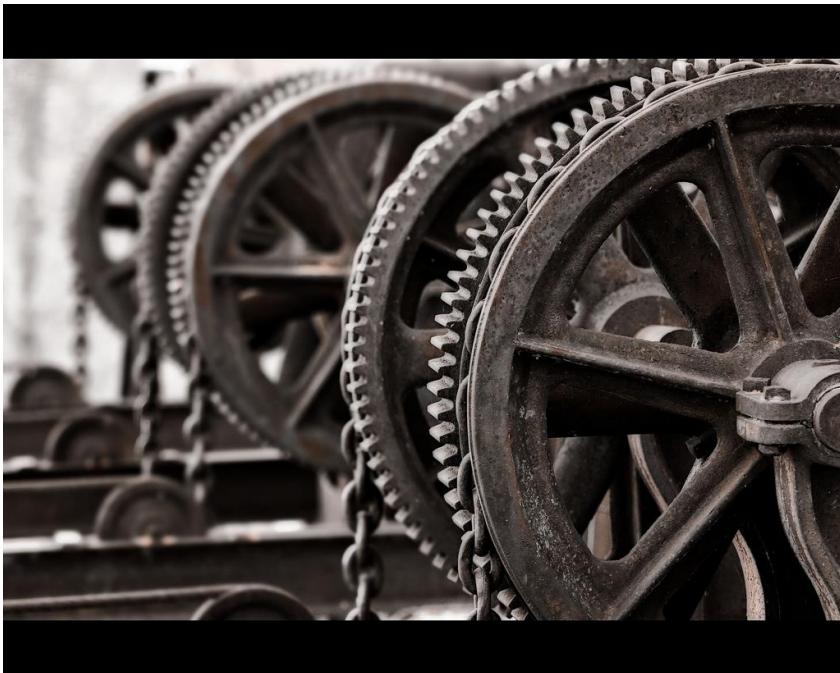
Working With Azure

And then what?

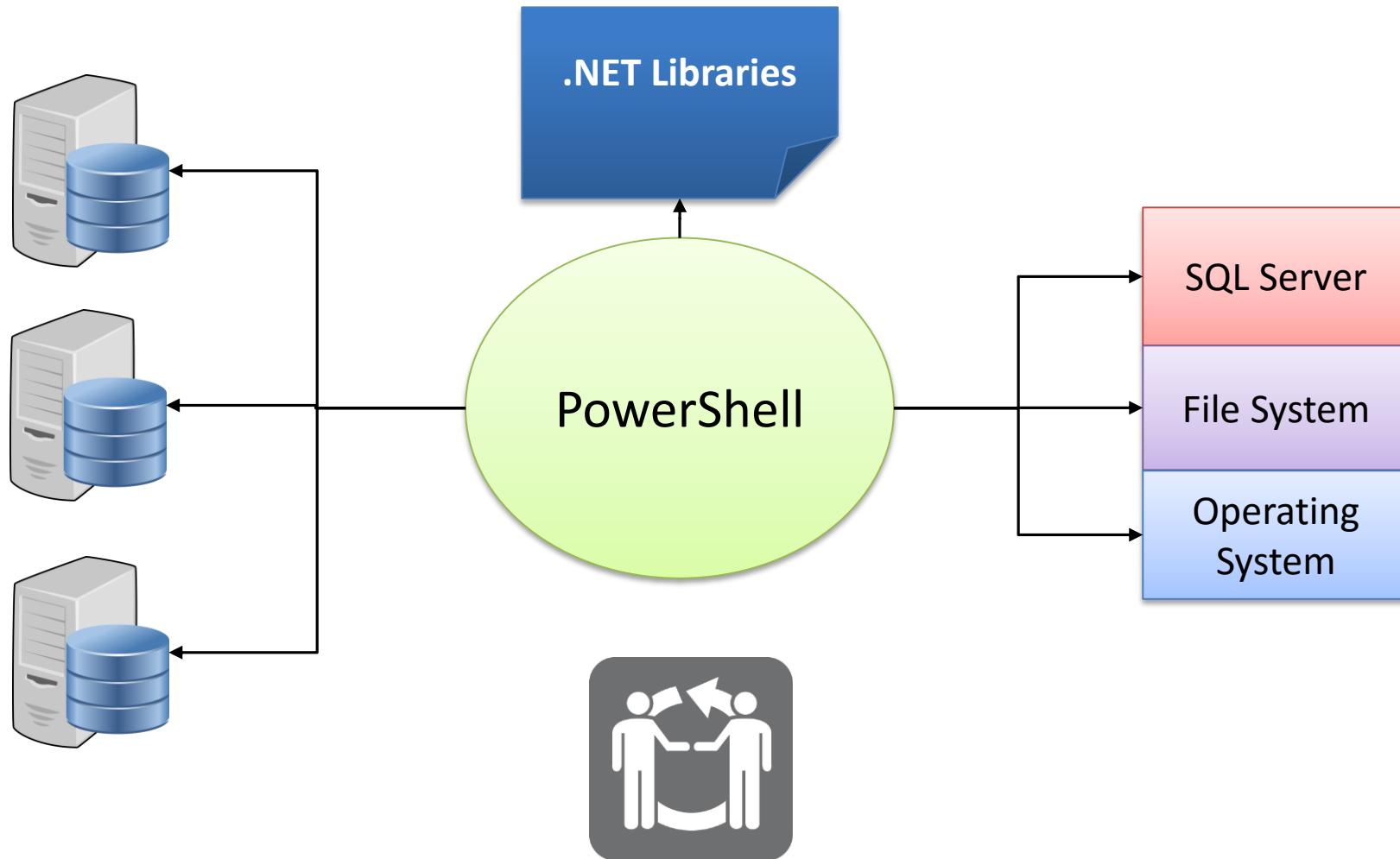
## What is Powershell?

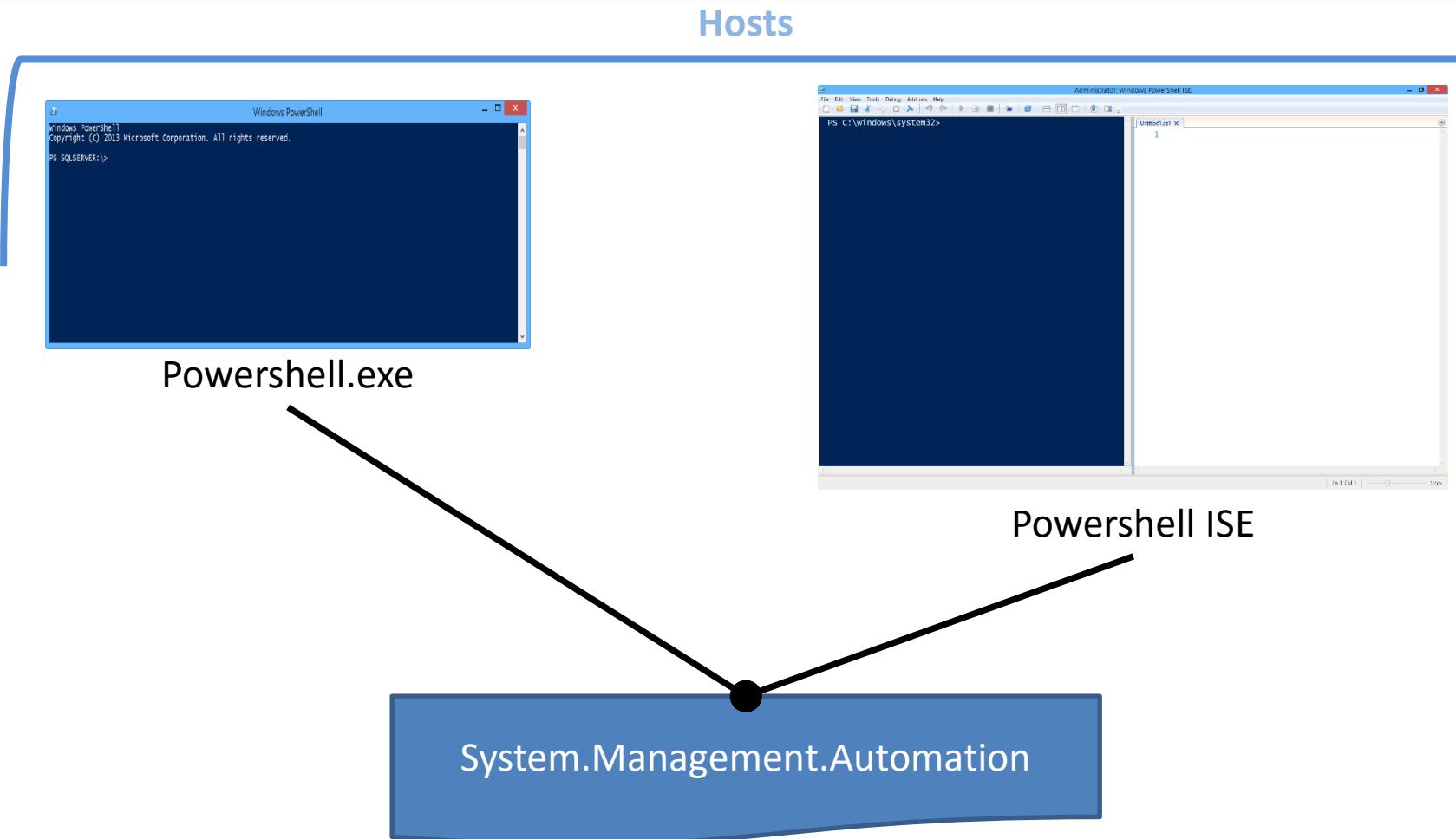


But for what?



- Envisioned by Jeffery Snover – 2002
  - The Monad Manifesto
- Released as Powershell RC 1 – April 2006
  - Originally called Project “Monad”
- Current available version: 5.0
  - Just released, 4.0 has been out for a bit





## Demo – The ISE

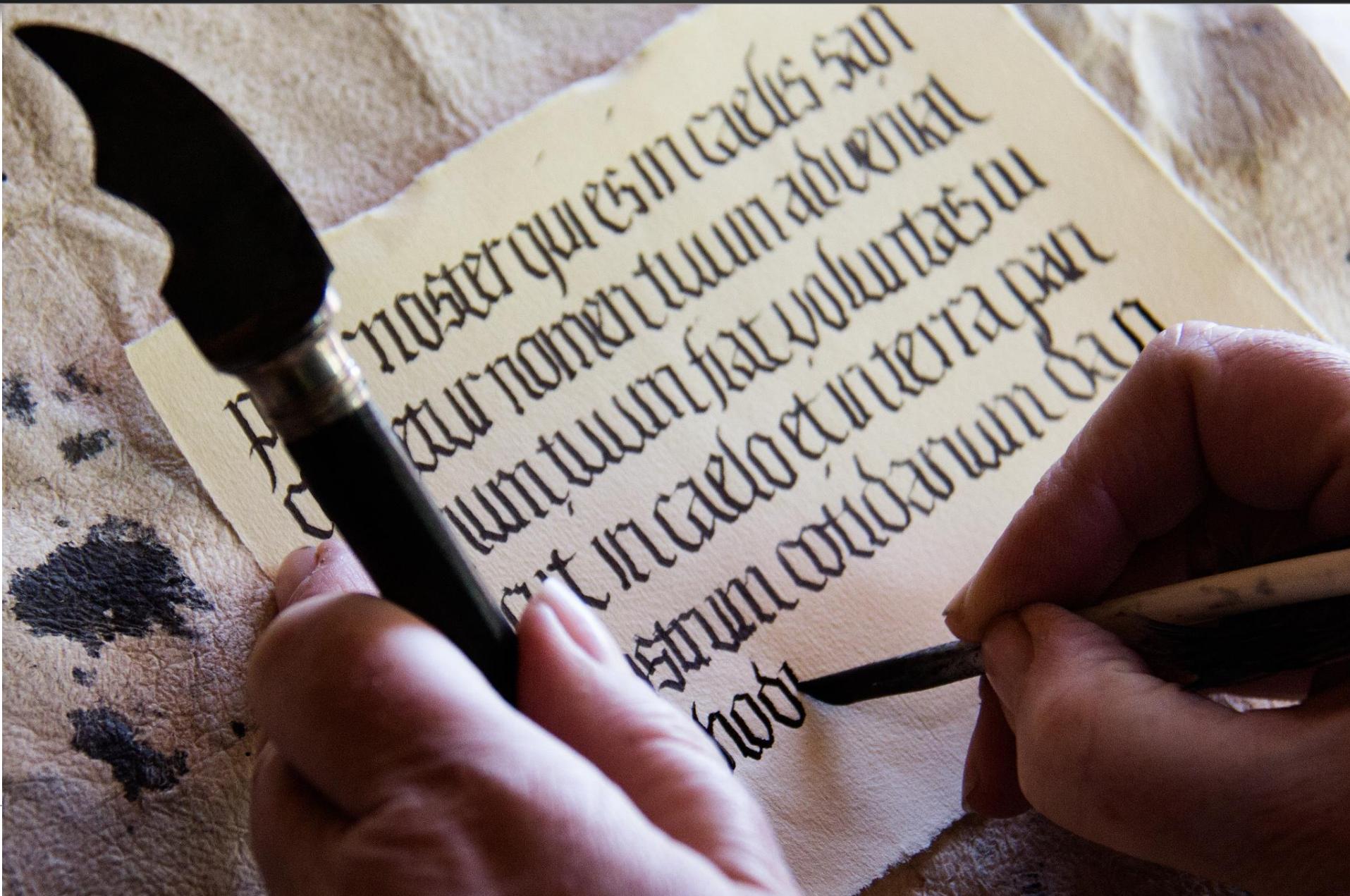
Administrator: Windows PowerShell ISE

```
File Edit View Tools Debug Add-ons Help
BuildVMsScratch.ps1 BuildAGScratch2.ps1 Untitled1.ps1*(Recovered) Build-ClusterScratch.ps1*(Recovered) Build_HVLab.ps1 Powershell_tips_tricks.ps1*(Recovered) SQLBenchmark.psm1

1 function New-LabVM{
2     param([parameter(Mandatory=$true)][string]$VMName
3           ,[ValidateSet('Full','Core')][string]$InstallType
4           )
5
6     $SourceVHD = switch($InstallType){
7         'Full' {'C:\vms\vhds\QM-Server2012Full.vhdx'}
8         'Core' {'C:\vms\vhds\QM-Server2012R2Core.vhdx'}
9     }
10    new-VM -Name $VMName -BootDevice CD -SwitchName 'VMswitch'
11
12    Copy-Item $SourceVHD "C:\vms\vhds\$VMname.vhdx"
13    Add-VMHardDiskDrive -VMName $VMName -Path "C:\vms\vhds\$VMname.vhdx" -ControllerNumber 0 -ControllerLocation 0
14
15    Set-VMVhdDrive -vmmname $VMName -Path C:\ISOs\en_windows_server_2012_r2_with_update_x64_dvd_6052708.iso -ToControllerNumber 1 -ToControllerLocation 0
16    Set-VMMemory -VMName $VMName -DynamicMemoryEnabled $true
17    Rename-VMNetworkAdapter -VMName $VMName -Name 'Network Adapter' -NewName 'LocalNetwork'
18
19    Get-VM -Name $VMName | Start-VM
20
21
22 New-LabVM -VMName 'AD-Full' -InstallType Full
23 New-LabVM -VMName 'P1Card-Core' -InstallType Core
24 New-LabVM -VMName 'Riker-Core' -InstallType Core
25
26 $pw='vanhDuten142' | ConvertTo-SecureString -AsPlainText -force
27 $cred = New-Object System.Management.Automation.PSCredential ('$OF\Administrator', $pw)
28 Add-Computer -DomainName 'SOF.local' -Credential $cred -NewName RIKER
29
30 $new='vanhDuten142' | ConvertTo-SecureString -AsPlainText -force
31
```

PS C:\windows\system32>

Ln 1 Col 1 90%



# **Cmdlets**

# **Variables**

# **Control Logic and Flow**

# **Errors and Error Handling**



Fundamental unit of “getting stuff done”

# Verb-Noun

Limited by  
Microsoft

Unlimited values,  
Should be  
descriptive

**Get-Help**

**New-Item**

**Remove-Module**

**Get-Command**

List Commands:

- \*New\*
- Module SQLPS

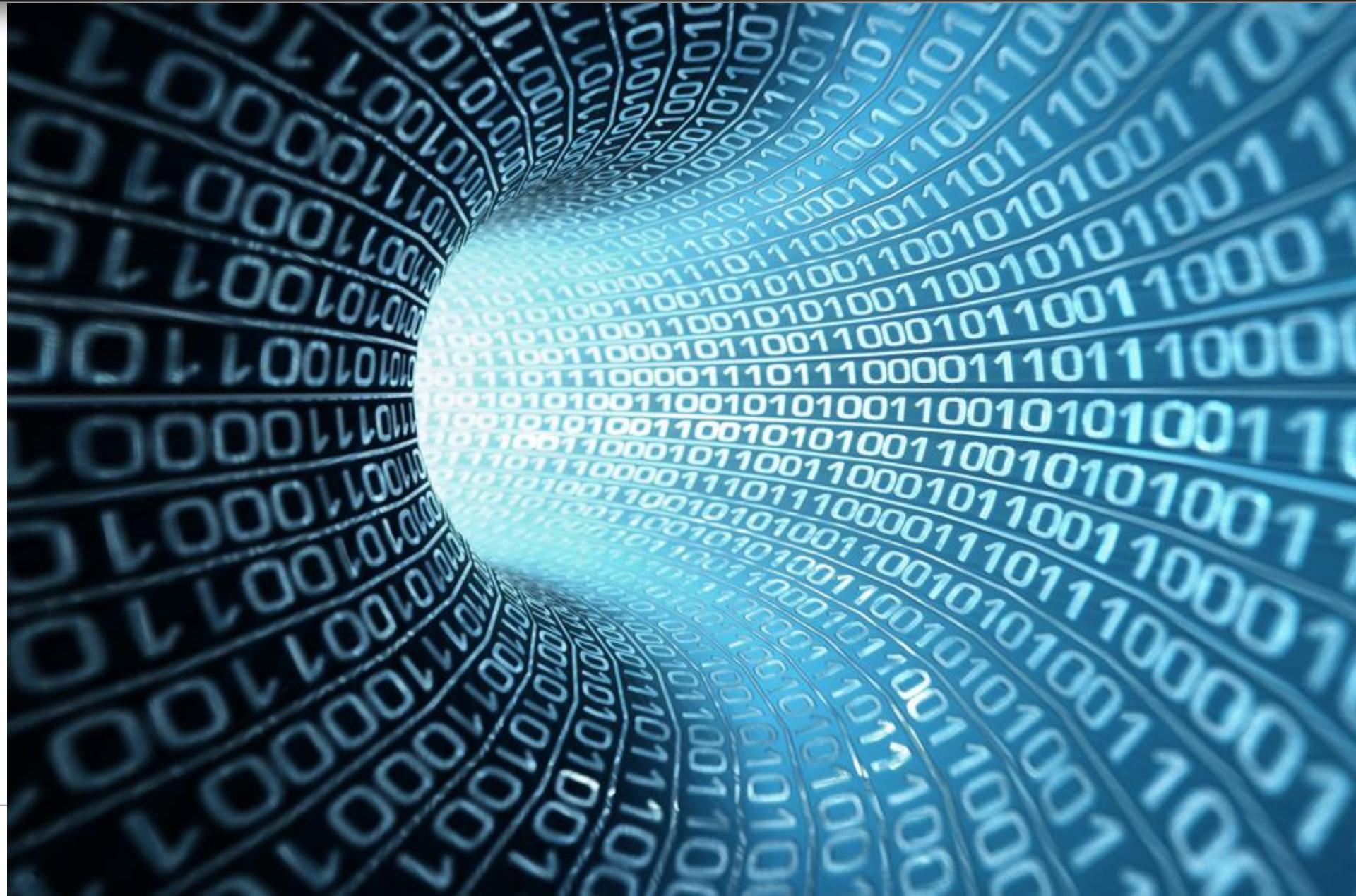
**Get-Help**

Show help info:

- man, help
- ShowWindow

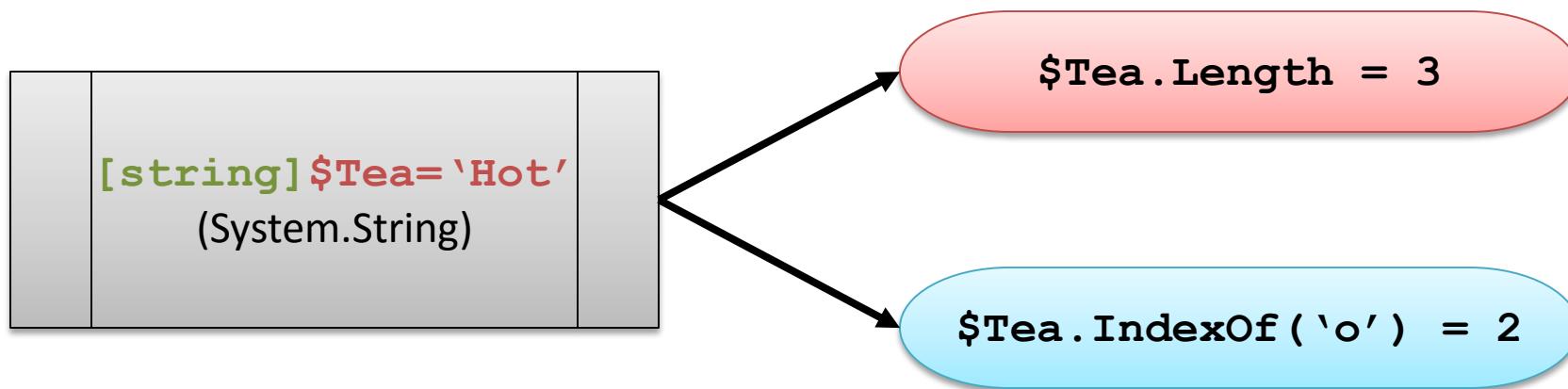
**Get-Member**

Methods and properties



# Everything is a .Net object!

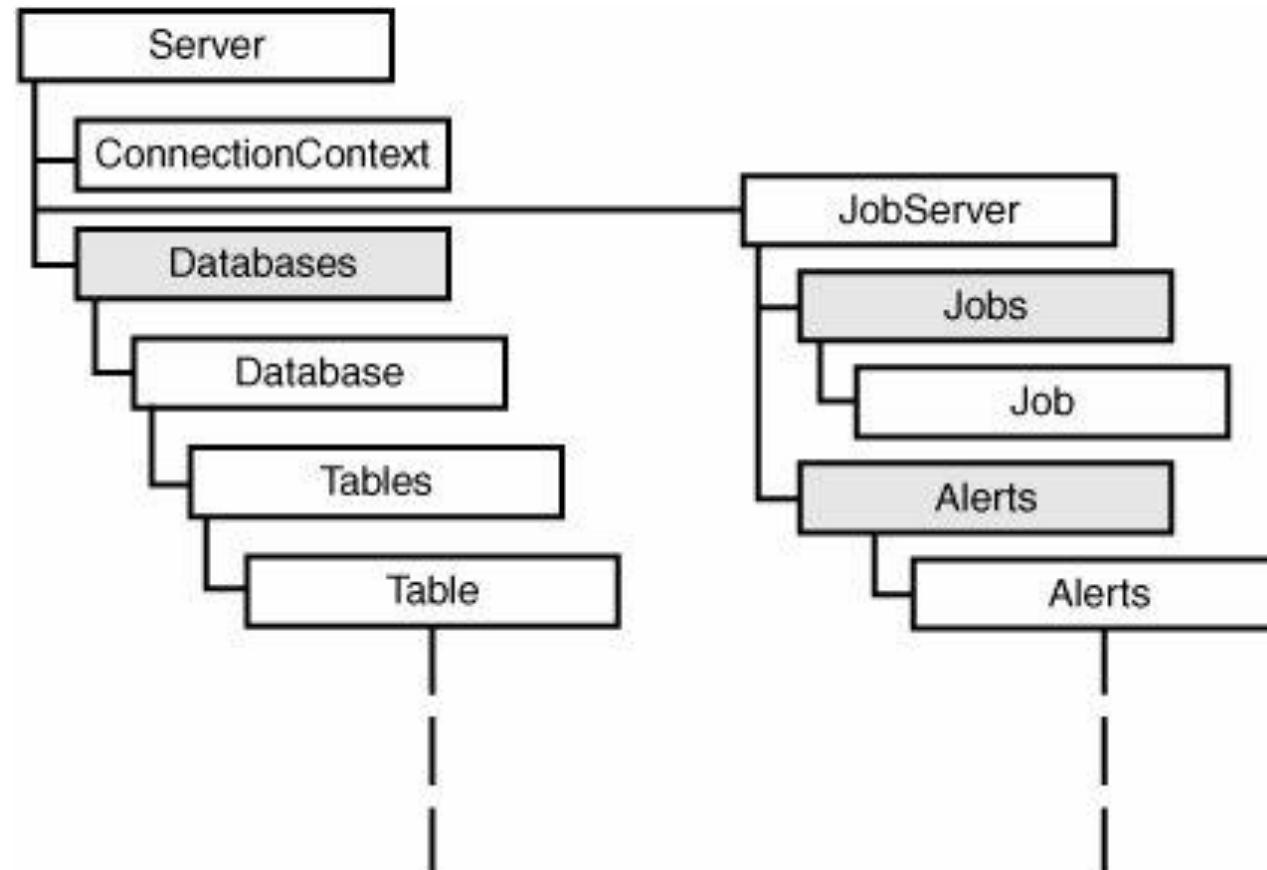
- Properties (attributes)
- Methods (functions, do things)



**'\$' indicates a variable**

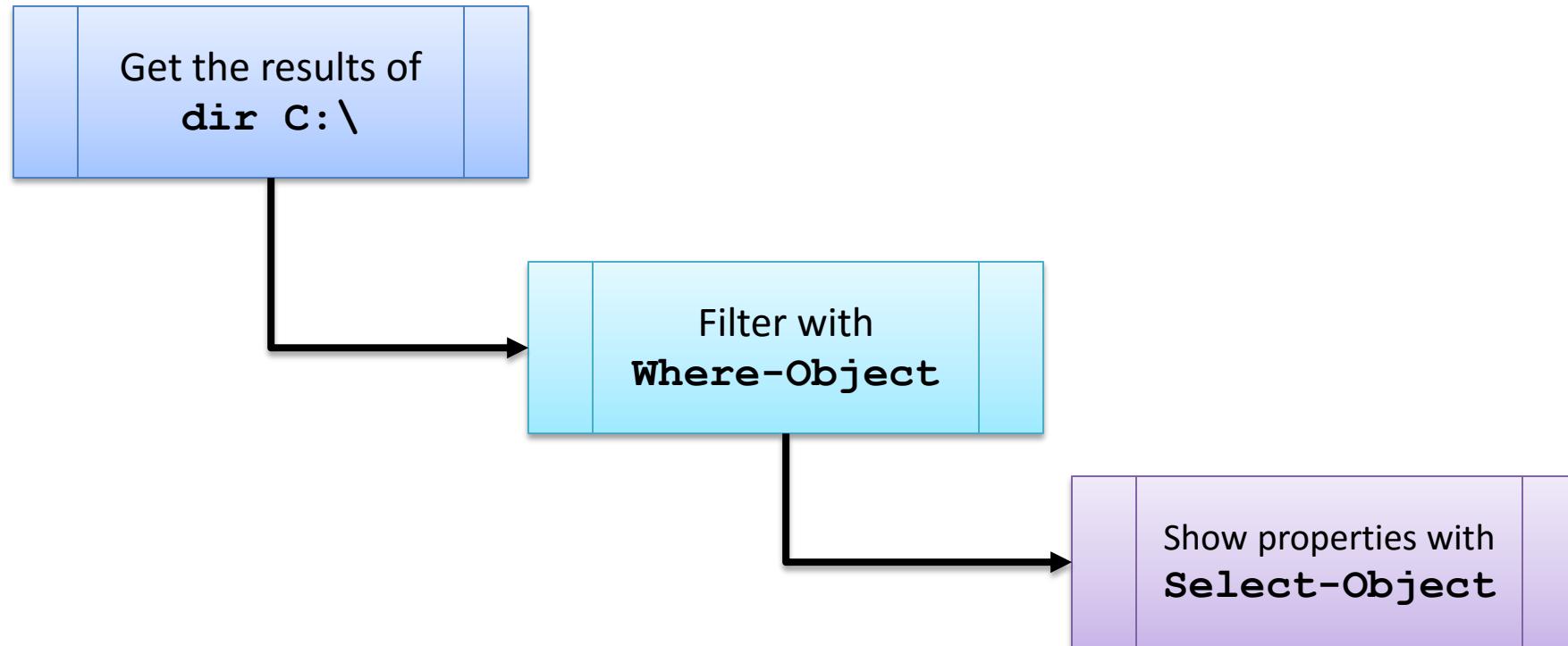
```
[string]$Tea = 'Hot'  
$Tea = 'Hot'  
$TeaTime = Get-Date
```

# Demo – Objects, and Variables





```
dir C:\ | Where-Object {$_._PsIsContainer -eq $true} |  
Select-Object name
```



## Cmdlet Support

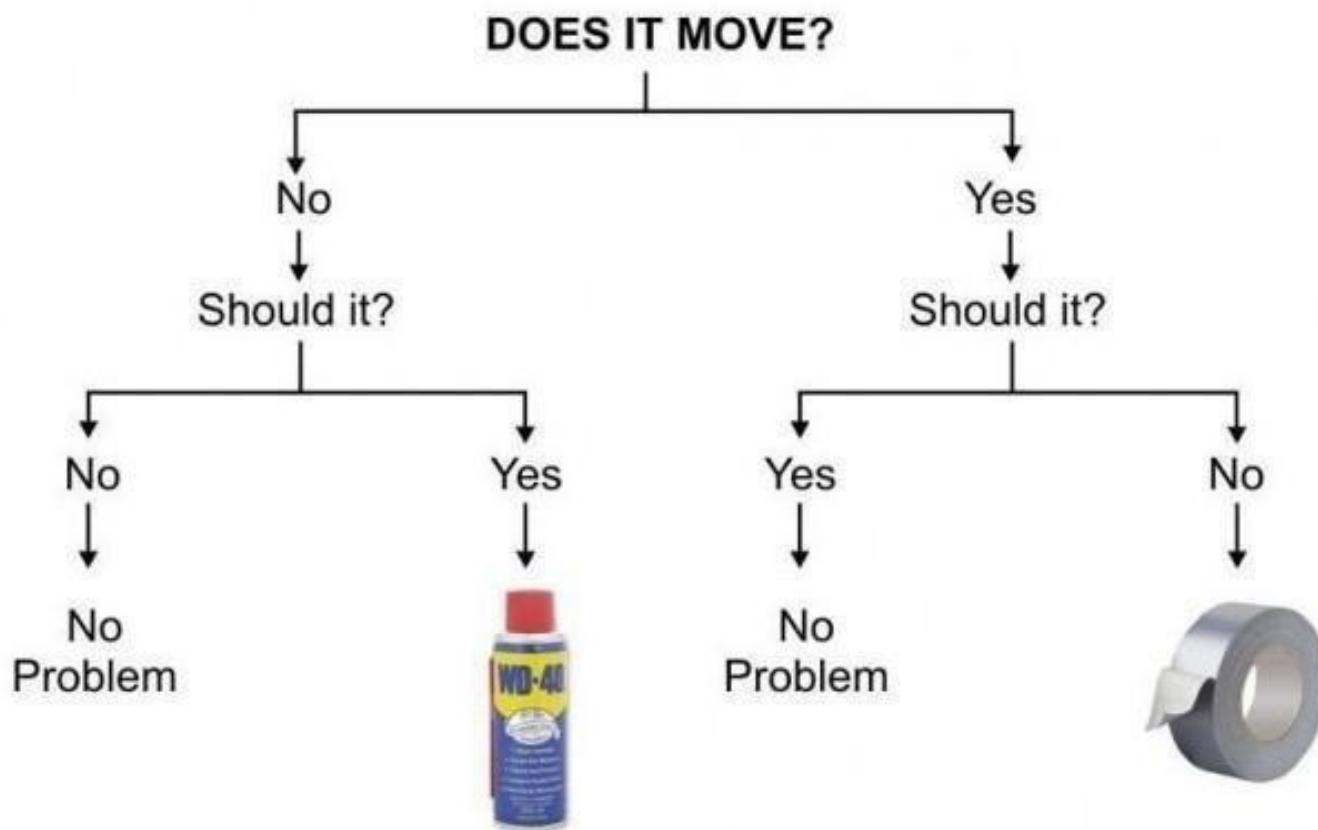
- `-InputObject`

## Looping

- Each Object in the Pipeline is touched
- `$_` is a special variable to represent current object



# Engineering Flowchart



## Shell Syntax

- >**: Redirect output, not greater than
- <**: Take input, not less than
- =**: Set value, not equality

## Powershell Operators

- eq, -ne**: equality check
- lt, -le, -gt, -ge**: Less than and Greater than
- and, -or**: Combine conditions
- like, -notlike**: Wild card comparisons

```
If (<condition>) {  
    <script block>  
}  
  
Else {  
    <script block>  
}  
  
Switch (<item>) {  
    <value> { <script block> }  
    <value> { <script block> }  
    ...  
}
```

```
For (<start>;<check>;<iteration>) {  
    <script block>  
}
```

```
Do{  
    <script block>  
} While/Until (<condition>)
```

```
ForEach(<object> in <collection>) {  
    <script block>  
}
```

**Break** – Exit Loop

---

**\$Error** – System error collection

**\$Error[0]** – Always the last error

```
PS C:\windows\system32> 1/0
Attempted to divide by zero.
At line:1 char:1
+ 1/0
+ ~~~
+ CategoryInfo          : NotSpecified: () [], RuntimeException
+ FullyQualifiedErrorId : RuntimeException
```

```
Try {  
    <script block>  
}  
Catch {  
    <script block>  
}  
Finally {  
    <script block>  
}
```

**Throw** – Create an exception



What are the three key cmdlets for discovery in Powershell?

**Get-Command, Get-Help, Get-Member**

If you want to use a cmdlet to create something (file, directory, object), what verb will it use?

**New**

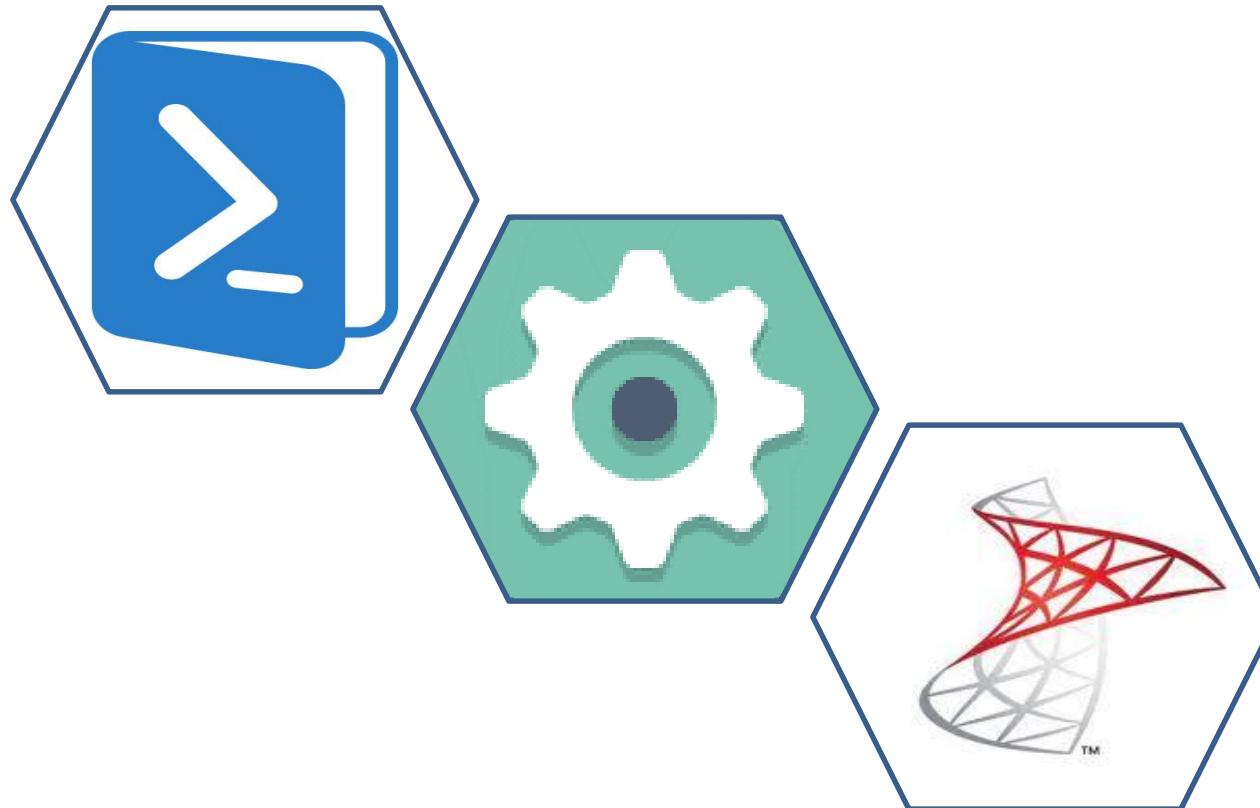
How can you extend the functionality of a command in one line?

**Use the pipeline**

What would you use to loop through a collection of objects?

**ForEach(){ }**





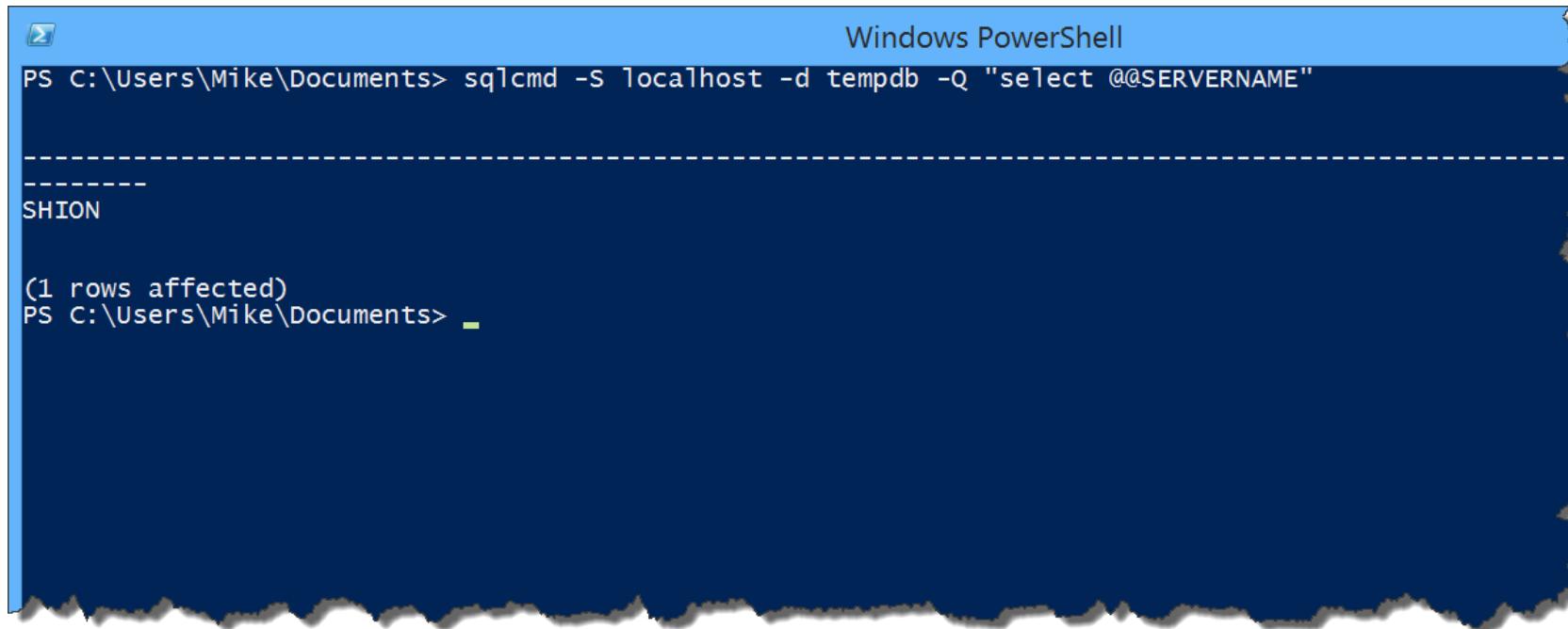
**Command line (sqlcmd)**

**The SQL Powershell Module  
(SQLPS)**

**Server Management Objects  
(SMO)**

**Powershell in the SQL Agent**





Windows PowerShell

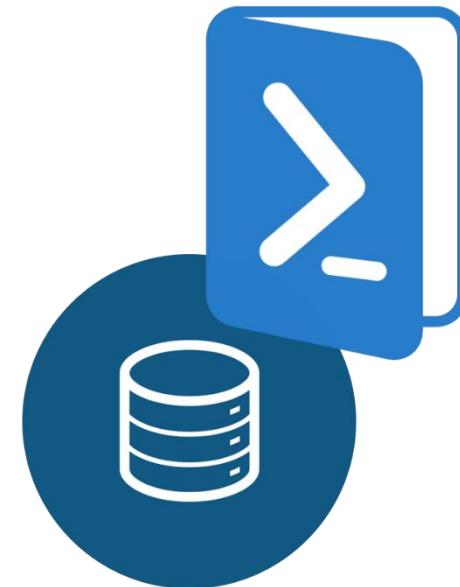
```
PS C:\Users\Mike\Documents> sqlcmd -s localhost -d tempdb -Q "select @@SERVERNAME"
-----
SHION

(1 rows affected)
PS C:\Users\Mike\Documents>
```

# SQLPS

- Powershell module for SQL Server
- Includes cmdlets and the provider
- SQL Server 2012 or greater
- Powershell 2.0 or greater

`Import-Module SQLPS`



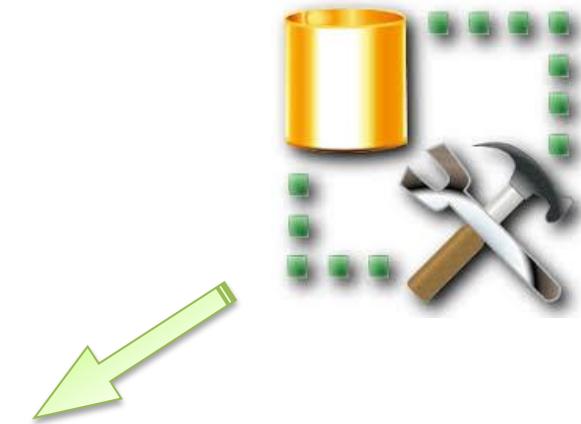
# SqlServer

- Introduced with SSMS July 2016 update!
- Includes everything from SQLPS along with fixes
- Not compatible with SQLPS!

`Import-Module SqlServer`



- Not installed specifically
- Included anytime you install ANY SQL Server component
- **SQLPS Installation Path:**  
C:\Program Files (x86)  
  \Microsoft SQL Server  
  \130\Tools\PowerShell\Modules
- **SqlServer Installation Path**  
C:\Program Files\WindowsPowerShell\Modules



.Net Library  
*requires .Net 2.0*

## SMO ASSEMBLIES

|                                                  |                                              |
|--------------------------------------------------|----------------------------------------------|
| Microsoft.SqlServer.Management.Common            | Microsoft.SqlServer.SqlEnum                  |
| Microsoft.SqlServer.Smo                          | Microsoft.SqlServer.RegSvrEnum               |
| Microsoft.SqlServer.Dmf                          | Microsoft.SqlServer.WmiEnum                  |
| Microsoft.SqlServer.Instapi                      | Microsoft.SqlServer.ServiceBrokerEnum        |
| Microsoft.SqlServer.SqlWmiManagement             | Microsoft.SqlServer.ConnectionInfoExtended   |
| Microsoft.SqlServer.ConnectionInfo               | Microsoft.SqlServer.Management.Collector     |
| Microsoft.SqlServer.SmoExtended                  | Microsoft.SqlServer.Management.CollectorEnum |
| Microsoft.SqlServer.SqlTDiagM                    | Microsoft.SqlServer.Management.Dac           |
| Microsoft.SqlServer.SString                      | Microsoft.SqlServer.Management.DacEnum       |
| Microsoft.SqlServer.Management.RegisteredServers | Microsoft.SqlServer.Management.Utility       |
| Microsoft.SqlServer.Management.Sdk.Sfc           |                                              |

PhotoExtremist.com

DEMO!





| Name     | Provider                                  | Root               | CurrentLocation     |
|----------|-------------------------------------------|--------------------|---------------------|
| A        | Microsoft.PowerShell.Core\FileSystem      | A:\                |                     |
| Alias    | Microsoft.PowerShell.Core\Alias           |                    |                     |
| C        | Microsoft.PowerShell.Core\FileSystem      | C:\                | Users\Administrator |
| Cert     | Microsoft.PowerShell.Security\Certificate | \                  |                     |
| D        | Microsoft.PowerShell.Core\FileSystem      | D:\                |                     |
| Env      | Microsoft.PowerShell.Core\Environment     |                    |                     |
| Function | Microsoft.PowerShell.Core\Function        |                    |                     |
| HKCU     | Microsoft.PowerShell.Core\Registry        | HKEY_CURRENT_USER  |                     |
| HKLM     | Microsoft.PowerShell.Core\Registry        | HKEY_LOCAL_MACHINE |                     |
| Variable | Microsoft.PowerShell.Core\Variable        |                    |                     |
| WSMan    | Microsoft.WSMan.Management\WSMan          |                    |                     |

## Windows Components as Drives

### Get-PSDrive

### Explore/Use like a FileSystem

Windows PowerShell

```
PS SQLSERVER:\> dir
```

| Name            | Root                       | Description                                |
|-----------------|----------------------------|--------------------------------------------|
| ---             | ---                        | -----                                      |
| DAC             | SQLSERVER:\DAC             | SQL Server Data-Tier Application Component |
| DataCollection  | SQLSERVER:\DataCollection  | SQL Server Data Collection                 |
| SQLPolicy       | SQLSERVER:\SQLPolicy       | SQL Server Policy Management               |
| Utility         | SQLSERVER:\Utility         | SQL Server Utility                         |
| SQLRegistration | SQLSERVER:\SQLRegistration | SQL Server Registrations                   |
| SQL             | SQLSERVER:\SQL             | SQL Server Database Engine                 |
| SSIS            | SQLSERVER:\SSIS            | SQL Server Integration Services            |
| XEvent          | SQLSERVER:\XEvent          | SQL Server Extended Events                 |
| DatabaseXEvent  | SQLSERVER:\DatabaseXEvent  | SQL Server Extended Events                 |
| SQLAS           | SQLSERVER:\SQLAS           | SQL Server Analysis Services               |

```
PS SQLSERVER:\sql\localhost\default
```



Provider

Folder

Host/Server

Instance

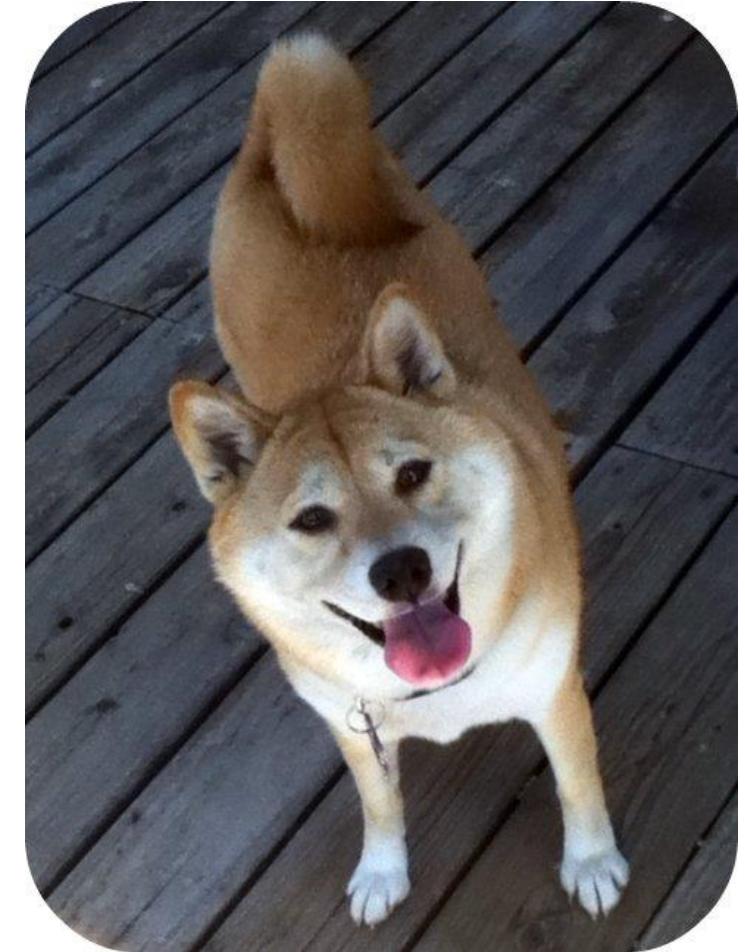
DEMO!



# Timeouts!

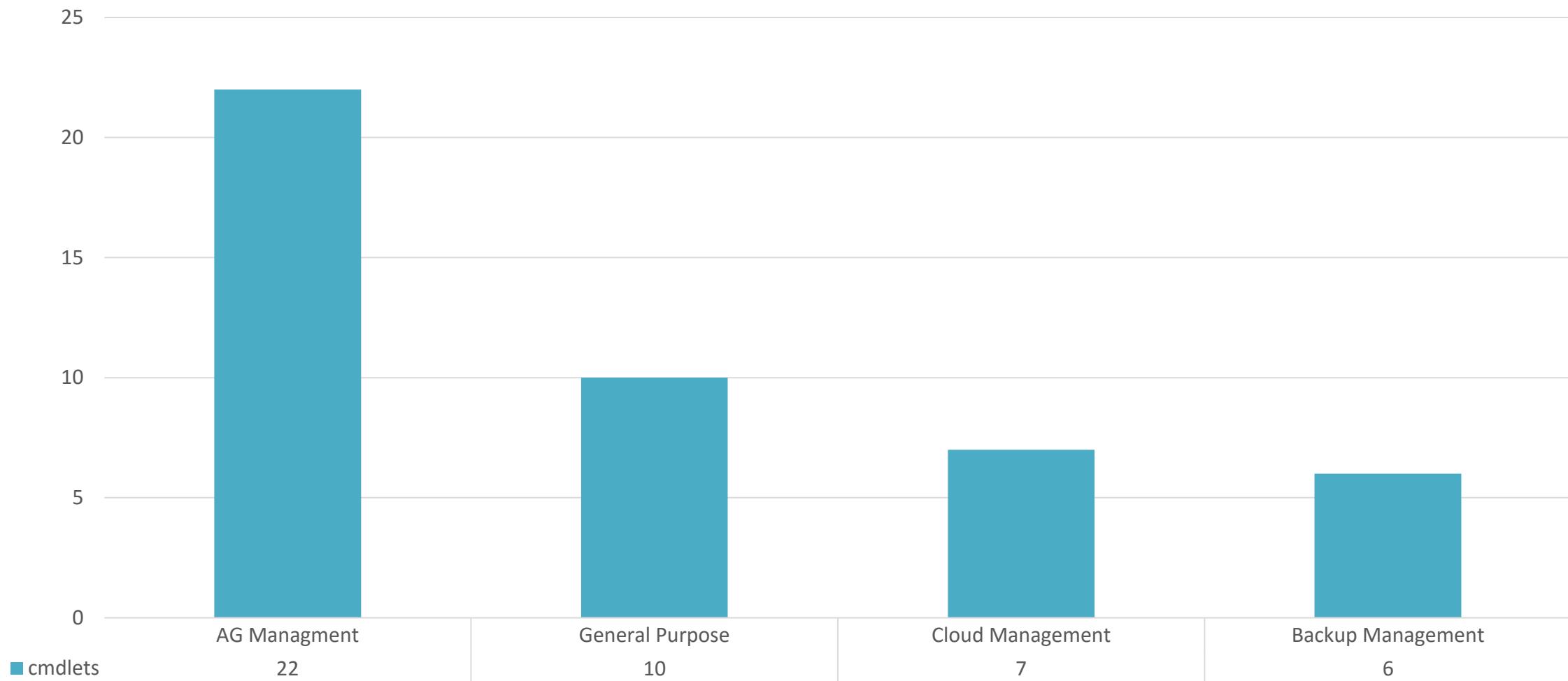
Tab completion and Intellisense

# Contextual behavior



# Get-Command

## 45 cmdlets total



|                         |
|-------------------------|
| Convert-UrnToPath       |
| Decode-SqlName          |
| Encode-SqlName          |
| Get-SqlCredential       |
| Get-SqlDatabase         |
| Invoke-PolicyEvaluation |
| Invoke-Sqlcmd           |
| New-SqlCredential       |
| Remove-SqlCredential    |
| Set-SqlCredential       |

Do not match Powershell  
naming standards

Useful for checking Policy  
Based Management

“Replacement” for sqlcmd,  
heavily used

`Backup-SqlDatabase`

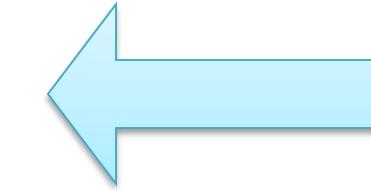
`Restore-SqlDatabase`

`New-SqlBackupEncryptionOption`

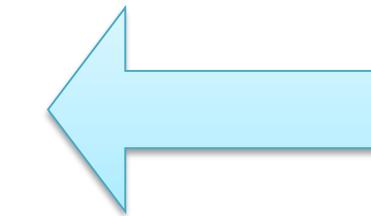
`Get-SqlSmartAdmin`

`Set-SqlSmartAdmin`

`Test-SqlSmartAdmin`



Standard  
Backup and Restore  
operations



SqlSmartAdmin cmdlets  
for Azure Managed Backups

Add-SqlFirewallRule

Remove-SqlFirewallRule

Get-SqlInstance

Set-SqlAuthenticationMode

Set-SqlNetworkConfiguration

Start-SqlInstance

Stop-SqlInstance

These cmdlets all require the  
Cloud Adapter Service

Careful, this doesn't return  
what you think it does...

These do the same as Start-  
Service and Stop-Service

|                                          |                                  |
|------------------------------------------|----------------------------------|
| Add-SqlAvailabilityDatabase              | Remove-SqlAvailabilityReplica    |
| Add-SqlAvailabilityGroupListenerStaticIp | Resume-SqlAvailabilityDatabase   |
| Disable-SqlAlwaysOn                      | Set-SqlAvailabilityGroup         |
| Enable-SqlAlwaysOn                       | Set-SqlAvailabilityGroupListener |
| Join-SqlAvailabilityGroup                | Set-SqlAvailabilityReplica       |
| New-SqlAvailabilityGroup                 | Set-SqlHADREndpoint              |
| New-SqlAvailabilityGroupListener         | Suspend-SqlAvailabilityDatabase  |
| New-SqlAvailabilityReplica               | Switch-SqlAvailabilityGroup      |
| New-SqlHADREndpoint                      | Test-SqlAvailabilityGroup        |
| Remove-SqlAvailabilityDatabase           | Test-SqlAvailabilityReplica      |
| Remove-SqlAvailabilityGroup              | Test-SqlDatabaseReplicaState     |

## Advanced Topic

## Always Encrypted 18 cmdlets

|                                                |                                          |
|------------------------------------------------|------------------------------------------|
| Add-SqlAzureAuthenticationContext              | New-SqlColumnEncryptionKey               |
| Add-SqlColumnEncryptionKeyValue                | New-SqlColumnEncryptionKeyEncryptedValue |
| Complete-SqlColumnMasterKeyRotation            | New-SqlColumnEncryptionSettings          |
| Get-SqlColumnEncryptionKey                     | New-SqlColumnMasterKey                   |
| Get-SqlColumnMasterKey                         | New-SqlCspColumnMasterKeySettings        |
| Invoke-SqlColumnMasterKeyRotation              | Remove-SqlColumnEncryptionKey            |
| New-SqlAzureKeyVaultColumnMasterKeySettings    | Remove-SqlColumnEncryptionKeyValue       |
| New-SqlCertificateStoreColumnMasterKeySettings | Remove-SqlColumnMasterKey                |
| New-SqlCngColumnMasterKeySettings              | Set-SqlColumnEncryption                  |

## SQL Agent 6 cmdlets

|                         |
|-------------------------|
| Get-SqlAgent            |
| Get-SqlAgentJob         |
| Get-SqlAgentJobHistory  |
| Get-SqlAgentJobSchedule |
| Get-SqlAgentJobStep     |
| Get-SqlAgentSchedule    |

## SQL Error Log 2 cmdlets

|                 |
|-----------------|
| Get-SqlErrorLog |
| Set-SqlErrorLog |

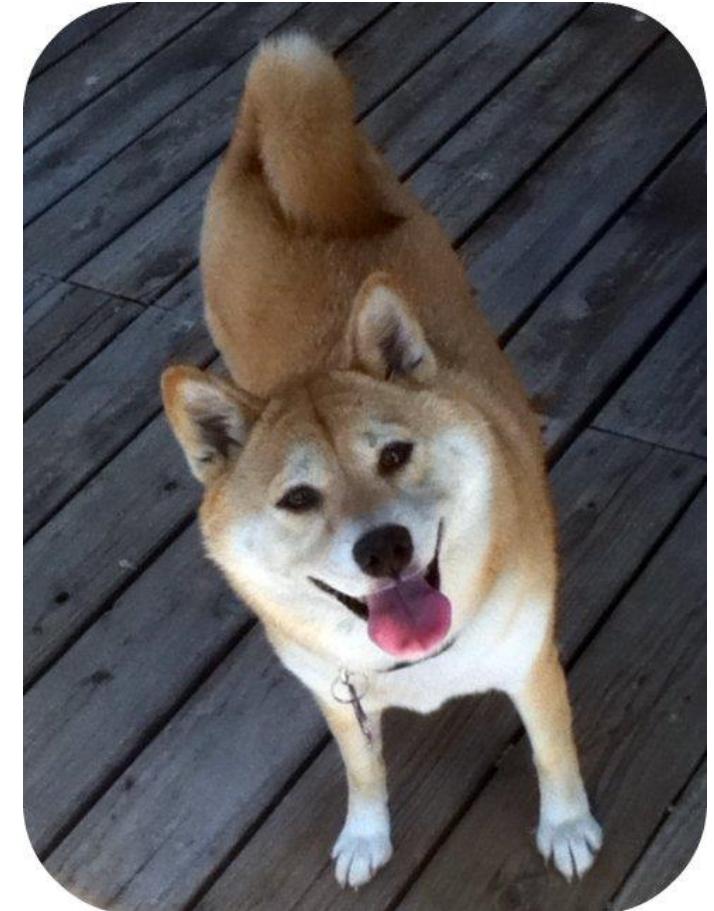


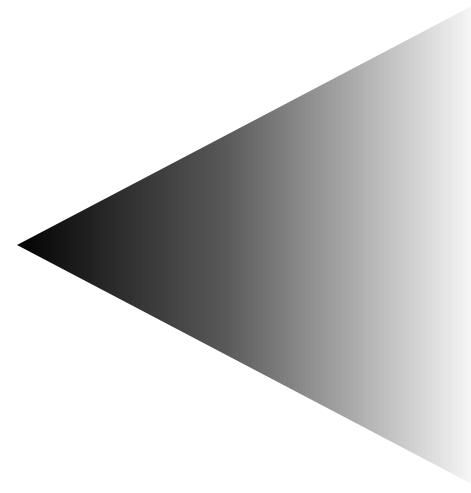
DEMO!



Poorly Documented  
*(Changing!)*  
Unexpected behavior

Lack of support  
*(Changing!)*





C#

Visual Basic

PowerShell

*Requires .NET framework 2.0*

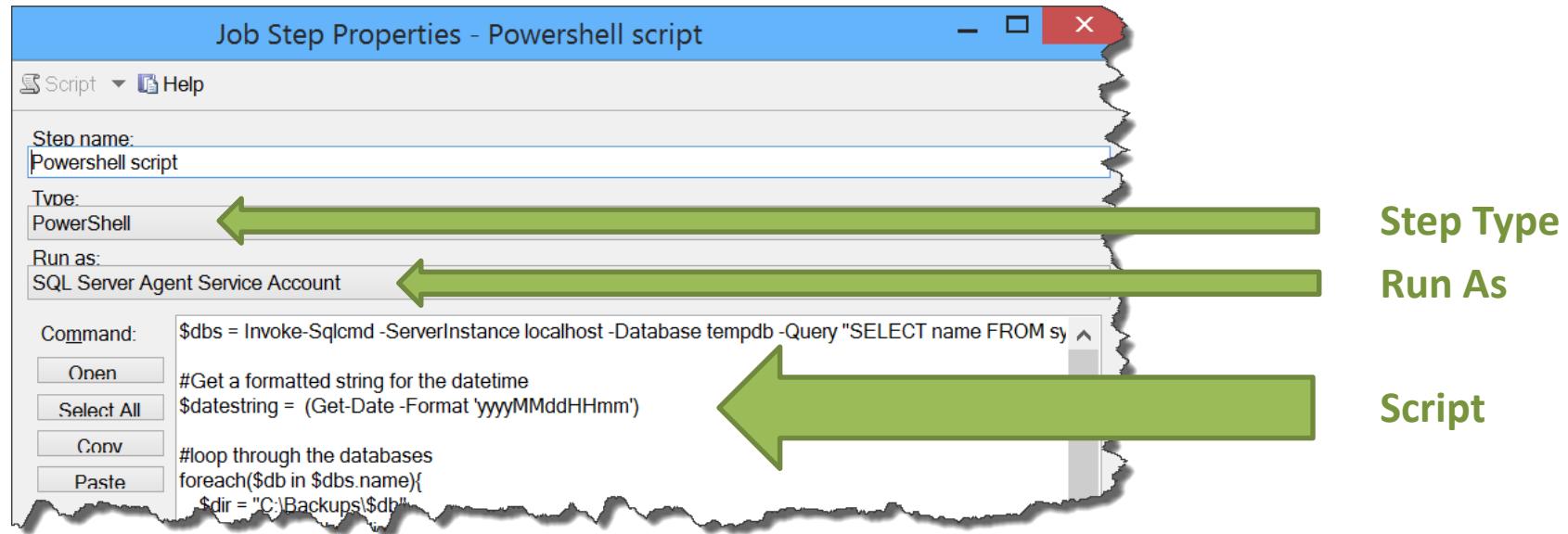
- Robust, object aware feature set
  - Flexibility
  - Durability
- Increased complexity, but also increased re-usability







**Use SQL Agent to mange jobs  
Easily integrate Powershell with your tasks**



Uses SQL Server's “mini” shell  
Powershell 2.0 (booooo)



Once you've loaded the SQLPS, how would you start working with the provider interactively in Powershell?

`cd SQLSERVER:\`

By default, what security account will a SQL Server Agent job run a Powershell script?

**SQL Server Agent service account**

Can you use Invoke-SqlCmd interactively?

**No**

To create a new SMO server object, what cmdlet will you use?

**New-Object**



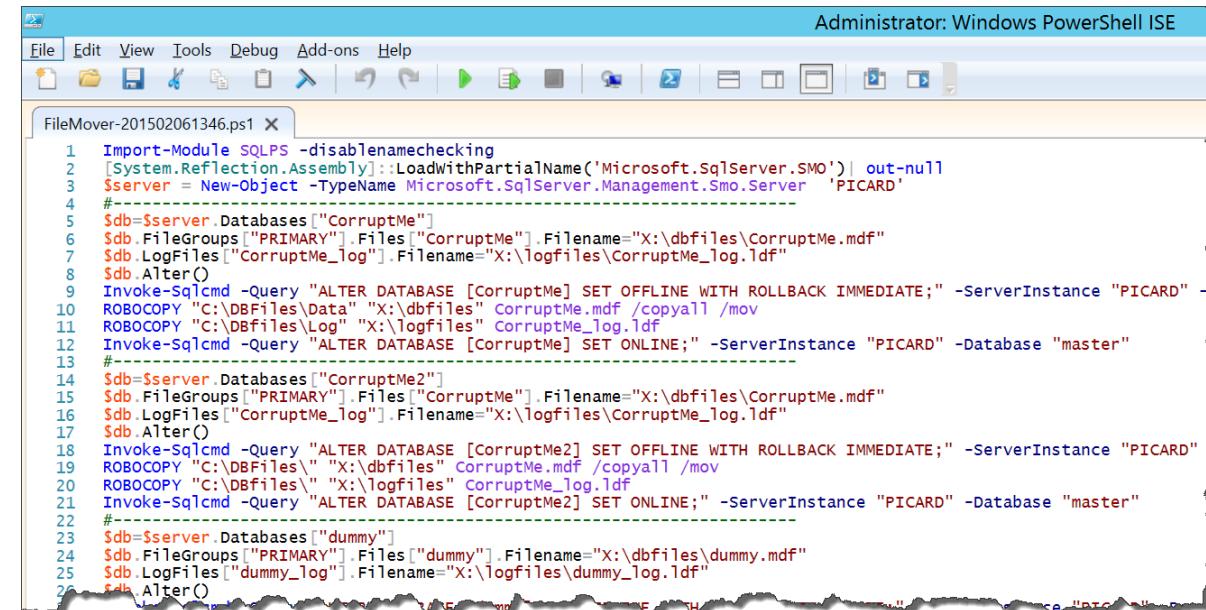


**Writing Scripts  
Re-using Scripts  
Your Powershell Profile  
Functions and Modules  
Best Practices and Documenting your scripts**

Scripts are saved as .ps1 files

Can be run by calling them at the command line

Also can leverage dot sourcing



```
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
FileMover-201502061346.ps1 X
1 Import-Module SQLPS -disablenamechecking
2 [System.Reflection.Assembly]::LoadWithPartialName('Microsoft.SqlServer.SMO') | out-null
3 $server = New-Object -TypeName Microsoft.SqlServer.Management.Smo.Server "PICARD"
4 #---
5 $db=$server.Databases["CorruptMe"]
6 $db.FileGroups["PRIMARY"].Files["CorruptMe"].Filename="X:\dbfiles\CorruptMe.mdf"
7 $db.LogFiles["CorruptMe_log"].Filename="X:\logfiles\CorruptMe_log.ldf"
8 $db.Alter()
9 Invoke-Sqlcmd -Query "ALTER DATABASE [CorruptMe] SET OFFLINE WITH ROLLBACK IMMEDIATE;" -ServerInstance "PICARD" -P
10 ROBOCOPY "C:\DBFiles\Data" "X:\dbfiles" CorruptMe.mdf /copyall /mov
11 ROBOCOPY "C:\DBfiles\Log" "X:\logfiles" CorruptMe_log.ldf
12 Invoke-Sqlcmd -Query "ALTER DATABASE [CorruptMe] SET ONLINE;" -ServerInstance "PICARD" -Database "master"
13 #---
14 $db=$server.Databases["CorruptMe2"]
15 $db.FileGroups["PRIMARY"].Files["CorruptMe"].Filename="X:\dbfiles\CorruptMe.mdf"
16 $db.LogFiles["CorruptMe_log"].Filename="X:\logfiles\CorruptMe_log.ldf"
17 $db.Alter()
18 Invoke-Sqlcmd -Query "ALTER DATABASE [CorruptMe2] SET OFFLINE WITH ROLLBACK IMMEDIATE;" -ServerInstance "PICARD" -P
19 ROBOCOPY "C:\DBFiles" "X:\dbfiles" CorruptMe.mdf /copyall /mov
20 ROBOCOPY "C:\DBfiles" "X:\logfiles" CorruptMe_log.ldf
21 Invoke-Sqlcmd -Query "ALTER DATABASE [CorruptMe2] SET ONLINE;" -ServerInstance "PICARD" -Database "master"
22 #---
23 $db=$server.Databases["dummy"]
24 $db.FileGroups["PRIMARY"].Files["dummy"].Filename="X:\dbfiles\dummy.mdf"
25 $db.LogFiles["dummy_log"].Filename="X:\logfiles\dummy_log.ldf"
26 $db.Alter()
```

Powershell Scripts can be parameterized

```
param($VariableName)
```

Parameters are referenced

- Positionally
- Variable name becomes parameter name

## Script file that loads whenever you start Different profiles

- For you in the current host
- For you in all hosts
- For everyone in the current host
- For everyone in all hosts

`$Profile + $PSHome`

`Get-Help about_Profiles`

You can package code into functions

```
function Function-Name{ }
```

Functions can be parameterized like scripts

They can also be called within a script

[Get-Help about\\_Functions](#)

---

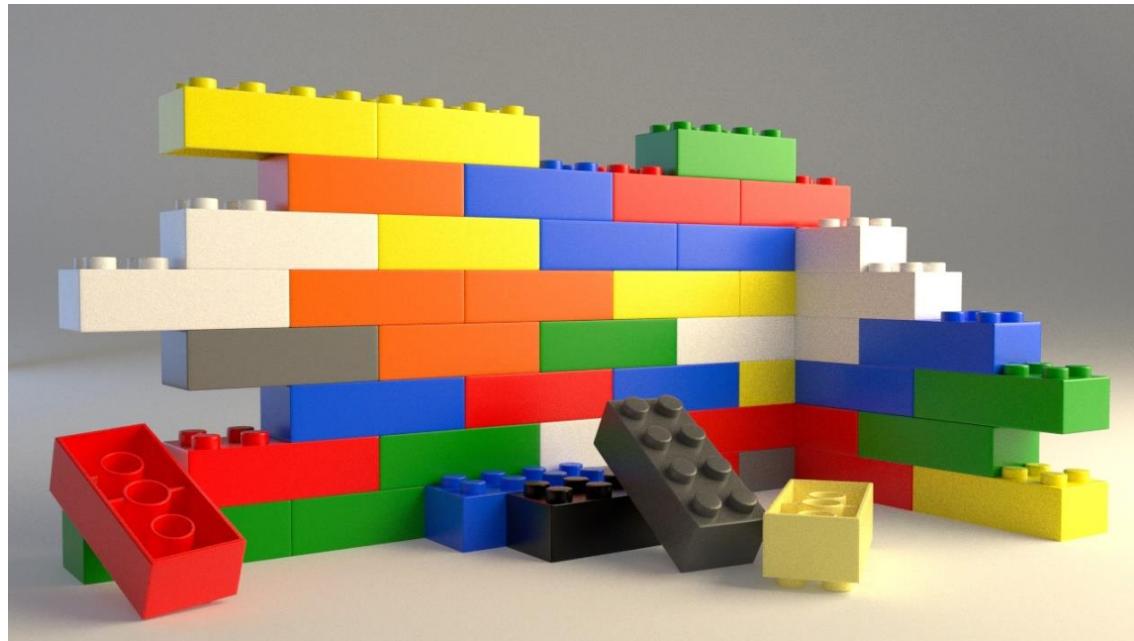
You can make your own modules to use  
No special syntax, just save as a .psm1  
Allows us to use custom functions direct from the  
command line.

Install to either:

C:\Users\<user>\Documents\WindowsPowerShell  
C:\Program Files\WindowsPowerShell

**Get-Help about\_Modules**





- Avoid “Swiss Army knives”! A function should do one thing very well.
- Everything is an object! If you need to return info, return it as an object.
- When building logging, use functions like Write-Verbose and Write-Warning. Write-Host only if necessary.

<#

## .SYNOPSIS

By using a specific formation, you can write  
your own Get-Help information

#>

[Get-Help about\\_comment\\_based\\_help](#)

---

What would you use to load a standard set of Powershell modules every time you opened a session?

Your Powershell profile

What command should you use to load a custom module?

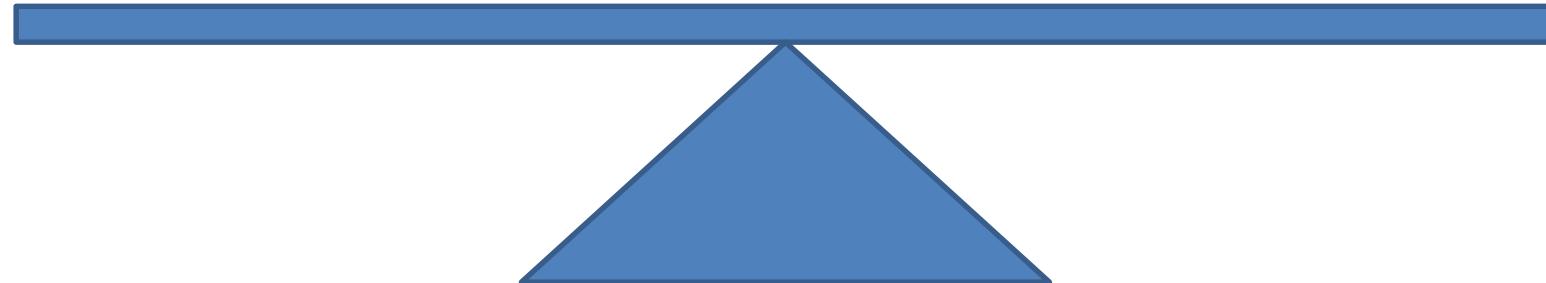
Import-Module

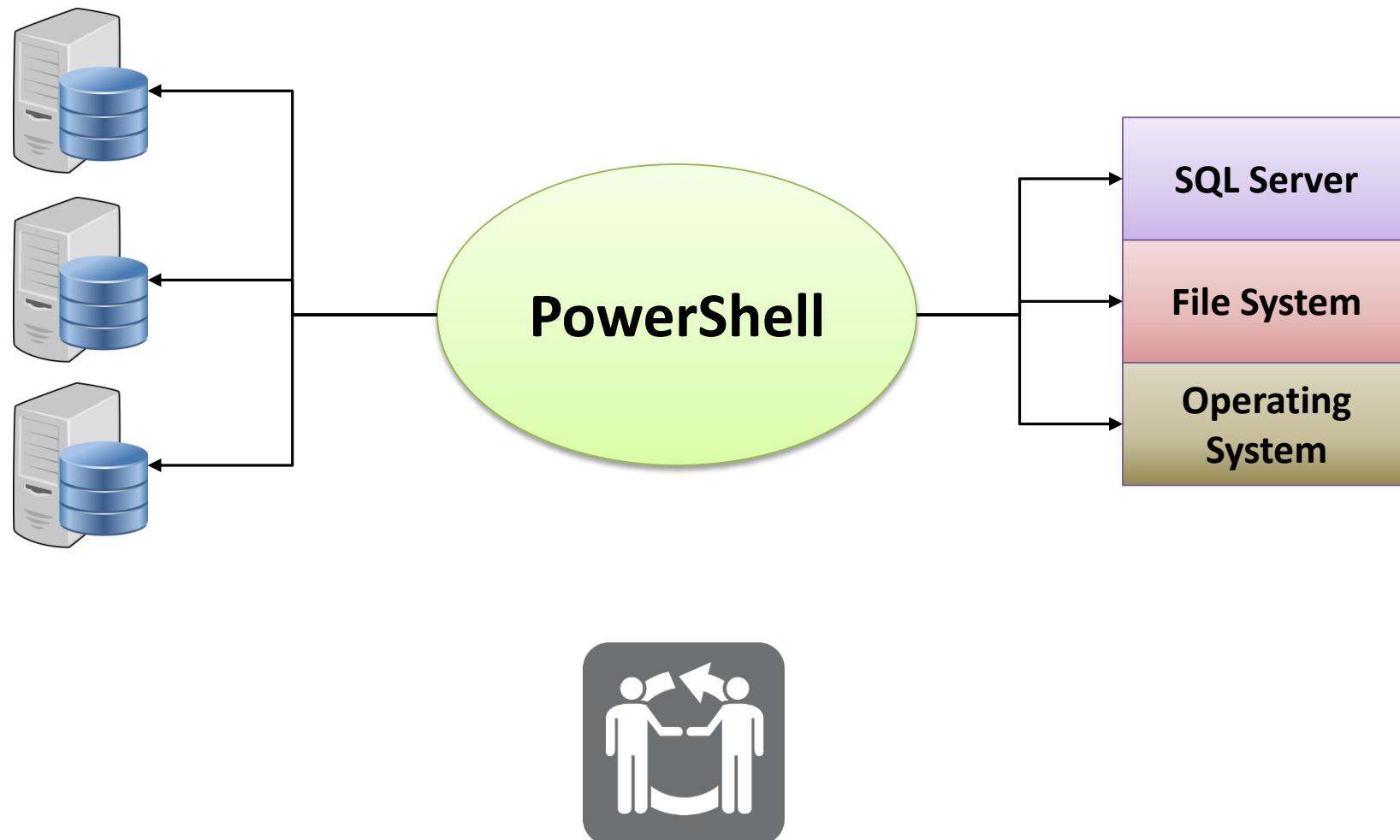
Where does the **param** keyword go when parameterizing a function or a script?

It must go at the beginning (can be after a comment block)

---













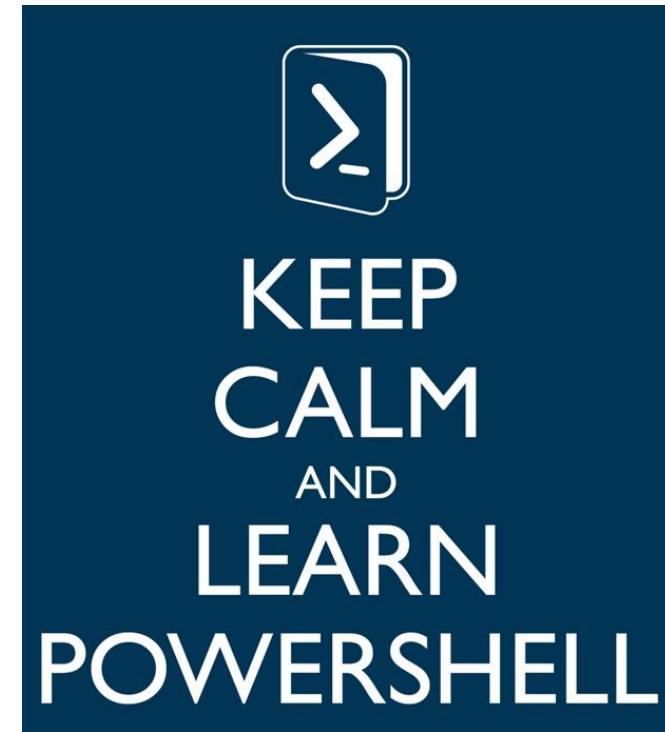
Windows Server...  
...with no GUI!

***Lean*** – Smaller footprint and install

***Stable*** – Fewer applications and services

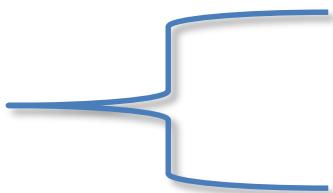
***Secure*** – Less security loopholes

# SCONFIG





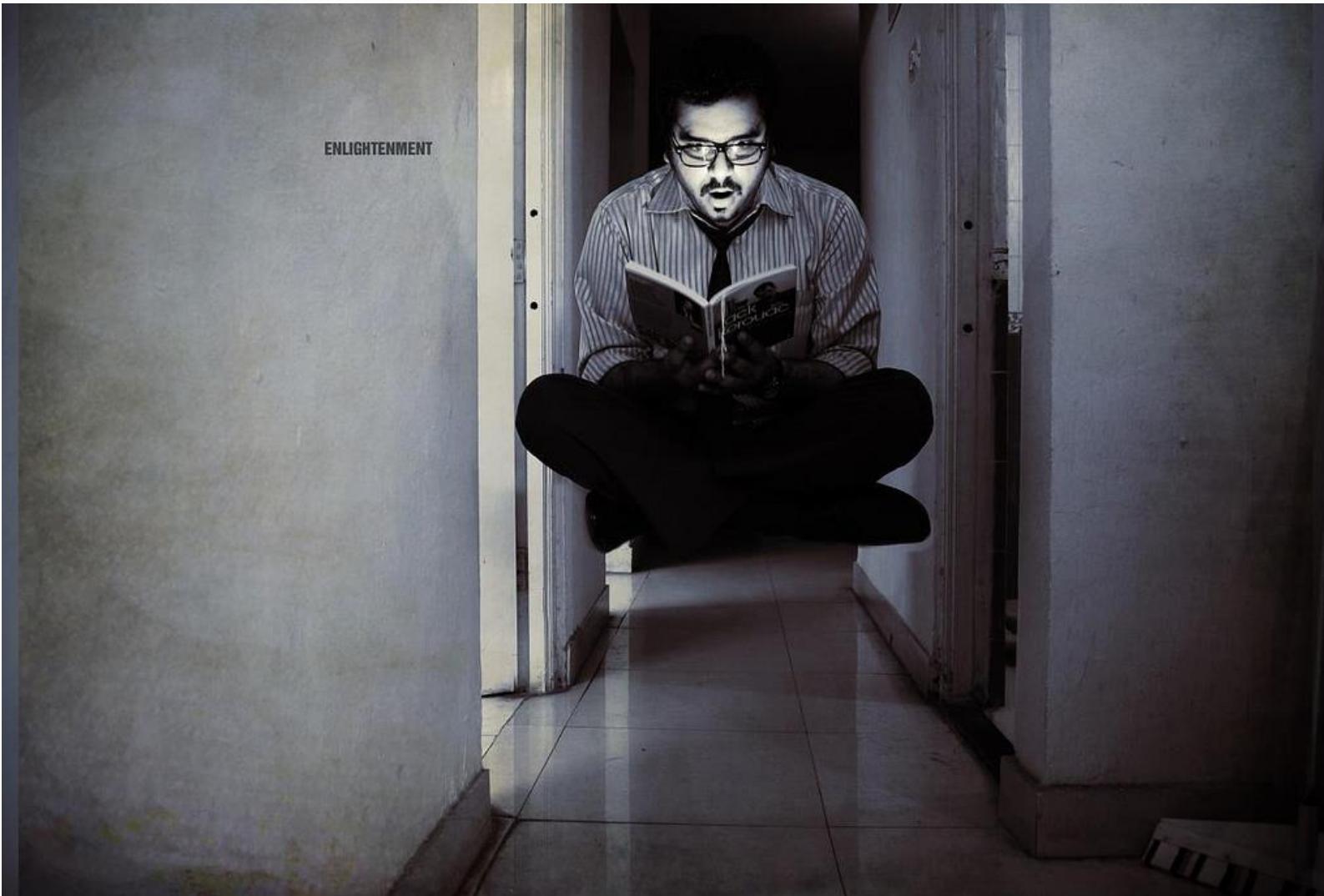
| Feature                             | Supported      |
|-------------------------------------|----------------|
| Database Engine Services            | Yes            |
| SQL Server Replication              | Yes            |
| Full Text Search                    | Yes            |
| Analysis Services                   | Yes            |
| Reporting Services                  | No             |
| SQL Server Data Tools (SSDT)        | No             |
| Client Tools Connectivity           | Yes            |
| Integration Services Server[1]      | Yes            |
| Client Tools Backward Compatibility | No             |
| Client Tools SDK                    | No             |
| SQL Server Books Online             | No             |
| Management Tools - Basic            | Remote Only[2] |
| Management Tools – Complete         | Remote Only[2] |
| Distributed Replay Controller       | No             |
| Distributed Replay Client           | Remote Only[2] |
| SQL Client Connectivity SDK         | No             |
| Microsoft Sync Framework            | Yes[3]         |
| Master Data Services                | No             |
| Data Quality Services               | No             |



| Requirement                          | How to install                                          |
|--------------------------------------|---------------------------------------------------------|
| .NET Framework 2.0 SP2               | Not included in Windows Server 2012, must be installed. |
| .NET Framework 3.5 SP1 Full Profile  | Not included in Windows Server 2012, must be installed. |
| .NET Framework 4 Server Core Profile | Shipped with Windows Server 2012.                       |
| Windows Installer 4.5                | Shipped with Windows Server 2012                        |
| Windows PowerShell 2.0               | Powershell 3.0 shipped with Windows Server 2012.        |

Well established tool  
Allows for consistent SQL  
builds  
Infrastructure as code





# **Getting Connected to Azure**

## **Creating an Azure VM with SQL Server**

### **Setting up and using Azure SQL DB**



## Powershell 5:

`Install-Module Azure`  
`Install-Module AzureRM`

## Powershell 4 and earlier:

`Download Web Installer`

<https://azure.microsoft.com/en-us/documentation/articles/powershell-install-configure/>

## Classic Deployments

- The “old” way
- Supports managing and creating

## Resource Manager Deployments

- The “new” way
- Mostly only creating
- Gives you better control

`Add-AzureAccount`

`Add-AzureRMAccount`

You need both!



## Traditional “Box Version” of SQL Server

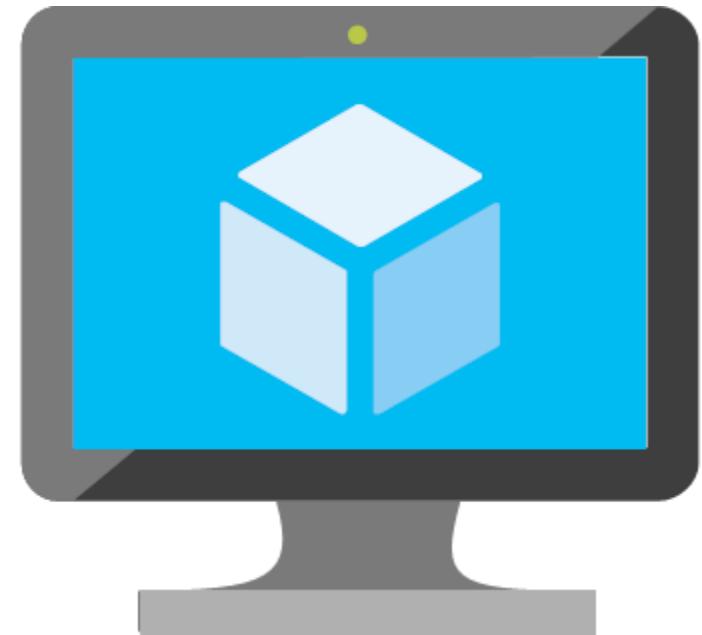
Just like normal SQL Server  
VM hosted in Azure

### Choose your class

Select a VM size based on your resources  
VM size defines how many CPU cores and RAM you get  
You can resize the machine later, but you can not adjust  
the CPU or the RAM explicitly

### Templates

Microsoft offers VM templates to create machines from  
Templates are based on SQL Server version



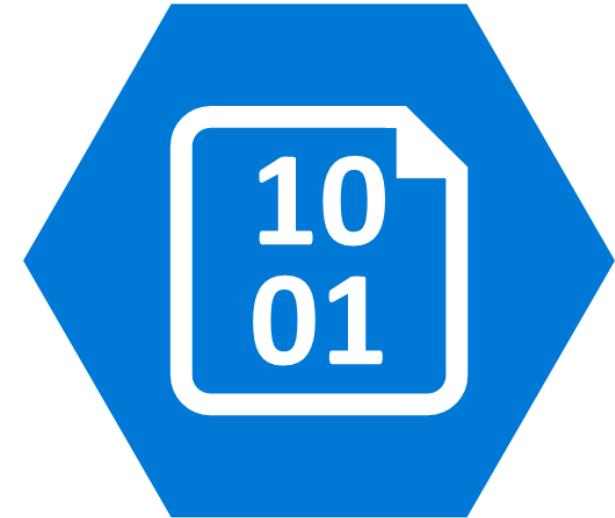
## Storage

Just as with your on-premises server, you need to determine what storage you need.

Azure VMs come with a default C:\ for OS and D:\ for “scratch”. These should NOT be used for database files.

We need to create Azure Data Disks for our database files, which live in an Azure storage account.

The storage account defines what type of storage we use (Standard or Premium) and must be created separately.



**Step 1:** Decide on your SQL Server version, size (CPU and RAM), and storage requirements.

**Step 2:** Create a storage account for the type of Azure storage you intend to use (Standard or Premium).

**Step 3:** Create the VM with the appropriate settings



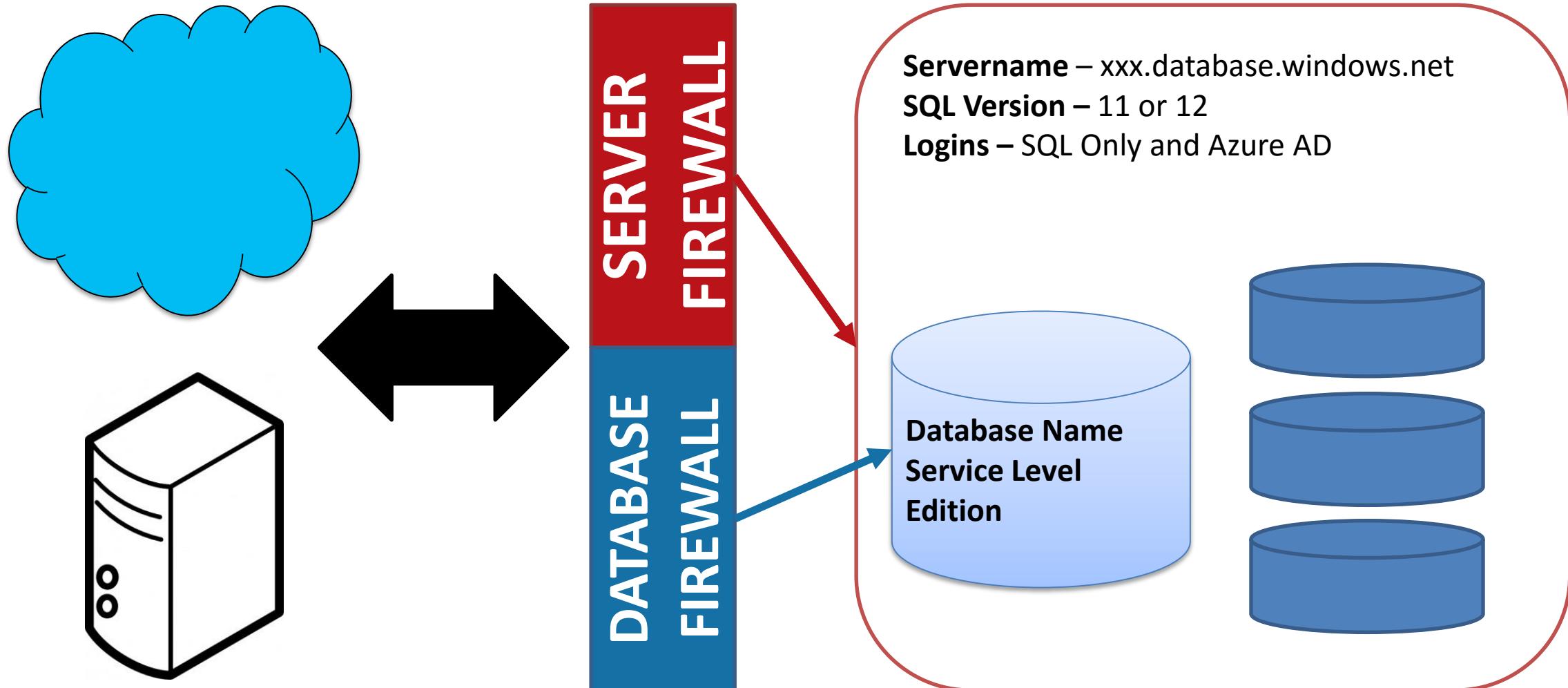


## Platform-as-a-Service

Its own version, does not directly relate to the “box” versions of SQL Server.

You don't get some server features, but that's OK.

What do you get?



|                                   | Basic                                    | Standard                                    |     |       |       | Premium                                                             |       |       |        |        |
|-----------------------------------|------------------------------------------|---------------------------------------------|-----|-------|-------|---------------------------------------------------------------------|-------|-------|--------|--------|
|                                   |                                          | S0                                          | S1  | S2    | S3    | P1                                                                  | P2    | P4    | P6/P3  | P11    |
| <b>Maximum database size</b>      | 2 GB                                     | 250 GB                                      |     |       |       | 500 GB                                                              |       |       |        | 1 TB   |
| DTUs                              | 5                                        | 10                                          | 20  | 50    | 100   | 125                                                                 | 250   | 500   | 1,000  | 1,750  |
| <b>Point-in-time restore</b>      | Any point last 7 days                    | Any point last 14 days                      |     |       |       | Any point last 35 days                                              |       |       |        |        |
| <b>Disaster recovery</b>          | Geo-Restore, restore to any Azure region | Standard Geo-Replication, offline secondary |     |       |       | Active Geo-Replication, up to 4 online (readable) secondary backups |       |       |        |        |
| <b>Max In-Memory OLTP storage</b> | NA                                       | NA                                          | NA  | NA    | NA    | 1 GB                                                                | 2 GB  | 3 GB* | 8 GB   | 10 GB* |
| <b>Max concurrent requests</b>    | 30                                       | 60                                          | 90  | 120   | 200   | 200                                                                 | 400   | 800   | 1,600  | 2,400  |
| <b>Max concurrent logins</b>      | 30                                       | 60                                          | 90  | 120   | 200   | 200                                                                 | 400   | 800   | 1,600  | 2,400  |
| <b>Max sessions</b>               | 300                                      | 600                                         | 900 | 1,200 | 2,400 | 2,400                                                               | 4,800 | 9,600 | 19,200 | 32,000 |

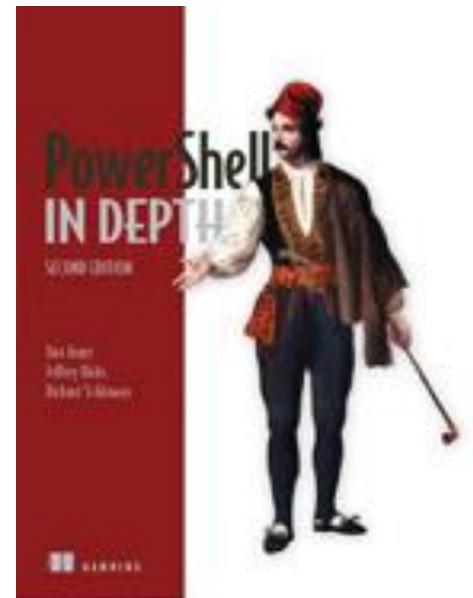
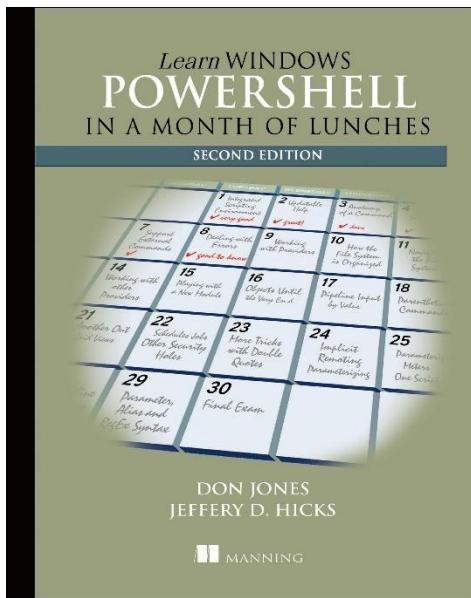
\* In-Memory OLTP storage limits will soon adjust to 4 for P4 and 14 for P11.



So now what?



- [Powershell In A Month of Lunches](#)
- [Powershell in Depth, 2<sup>nd</sup> Edition](#)



- [The Scripting Guys](http://blogs.technet.com/b/heyscriptingguy/)  
[\(http://blogs.technet.com/b/heyscriptingguy/\)](http://blogs.technet.com/b/heyscriptingguy/)
- [Jeff Hicks](http://jdhitsolutions.com/blog/)  
[\(http://jdhitsolutions.com/blog/\)](http://jdhitsolutions.com/blog/)
- [Powershell.org](http://powershell.org/)  
[\(http://powershell.org/\)](http://powershell.org/)

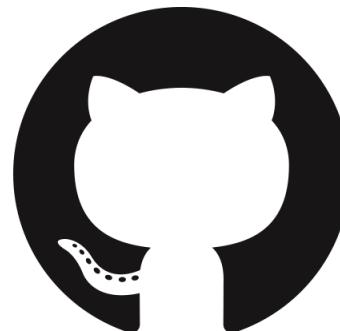
- [Ben Miller](http://www.dbaduck.com/)  
[\(http://www.dbaduck.com/\)](http://www.dbaduck.com/)
- [Allen White](http://sqlblog.com/blogs/allen_white/)  
[\(http://sqlblog.com/blogs/allen white/\)](http://sqlblog.com/blogs/allen_white/)
- [Kendal Van Dyke](http://www.kendalvandyke.com/)  
[\(http://www.kendalvandyke.com/\)](http://www.kendalvandyke.com/)
- [Laerte Junior](https://www.simple-talk.com/author/laerte-junior/)  
[\(https://www.simple-talk.com/author/laerte-junior/\)](https://www.simple-talk.com/author/laerte-junior/)
- [Chrissy LeMaire](https://blog.netnerds.net/author/chrissy/)  
[\(https://blog.netnerds.net/author/chrissy/ \)](https://blog.netnerds.net/author/chrissy/)

## Microsoft Virtual Academy PluralSight (\$\$)



<https://github.com/MikeFal>

- Powershell repository (all my scripts, including WIP)
- Intro To Powershell repository (all the scripts from THIS class)



Find tasks to automate

Manage the file system ONLY through Powershell

Rewrite a T-SQL or other task using Powershell

And so on...



[mike@mikefal.net](mailto:mike@mikefal.net)



[www.mikefal.net](http://www.mikefal.net)



[@Mike Fal](https://twitter.com/@Mike_Fal)



<https://upsearch.com/connect-with-us/>



<https://github.com/MikeFal>