POWERSHELL FOR
THE SQL SERVER PROFESSIONALS

Mike Fal - www.mikefal.net

Microsoft® SQL Server®

www.mikefal.net

MVP

Microsoft CERTIFIED Solutions Expert

Data Platform

Mike Fal

Don’t Panic!

```
    $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 "*.dff"} | 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{
```

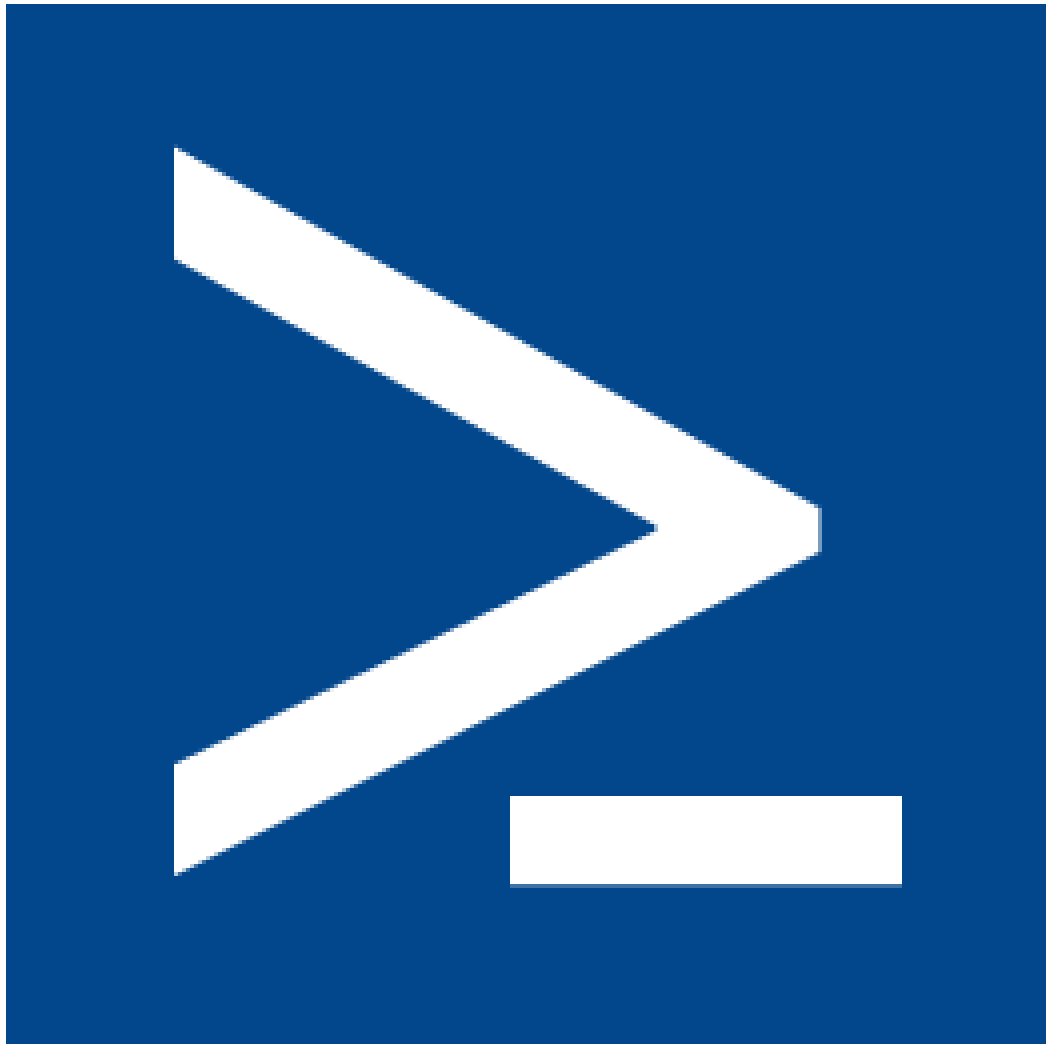
Don’t focus on the code,
focus on the concepts.

Ask questions!

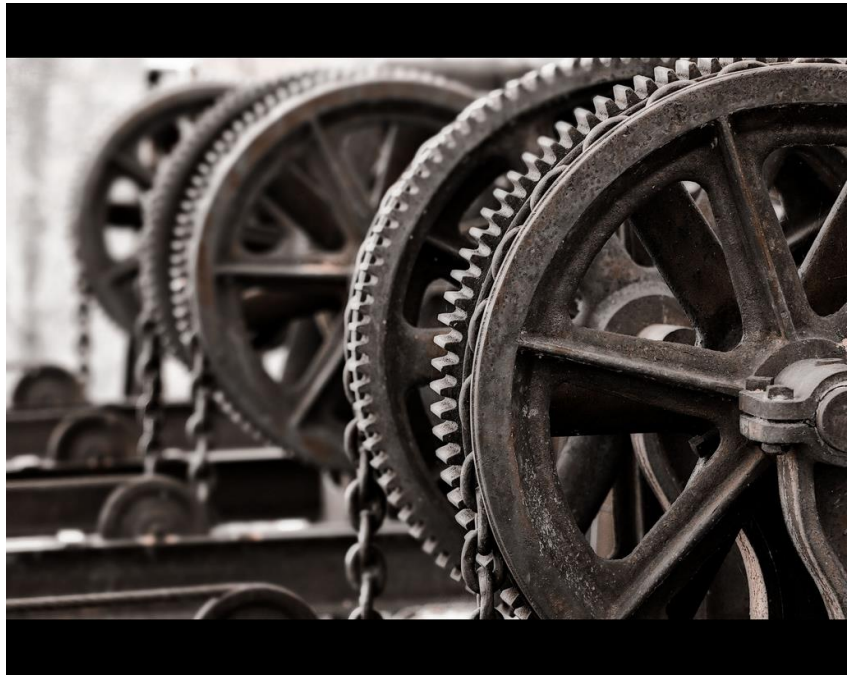
Get-Agenda

- Powershell Basics
- Powershell and SQL Server
- Re-using Powershell
- Practical Examples
- Working With Azure
- And then what?

What is Powershell?



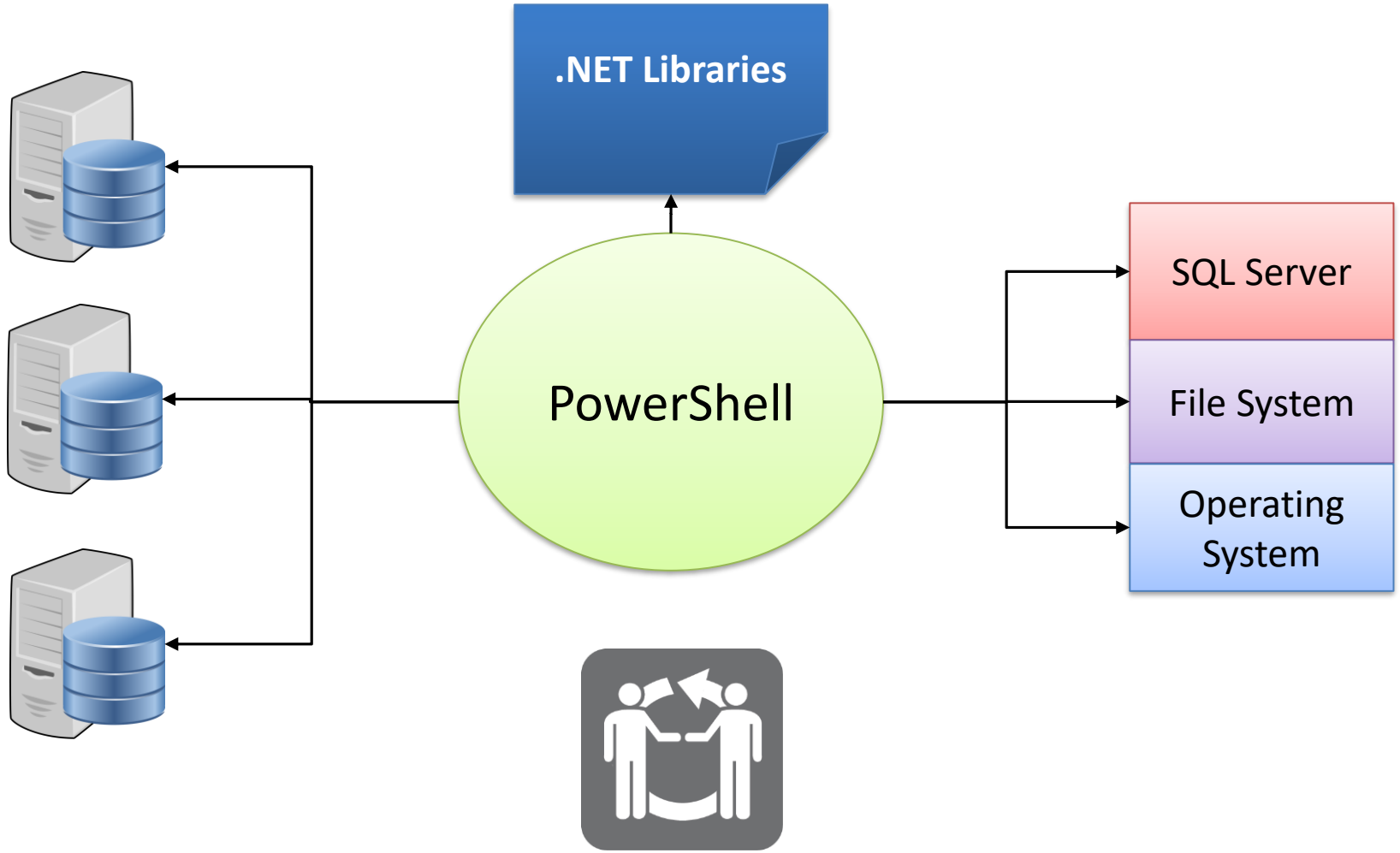
But for what?



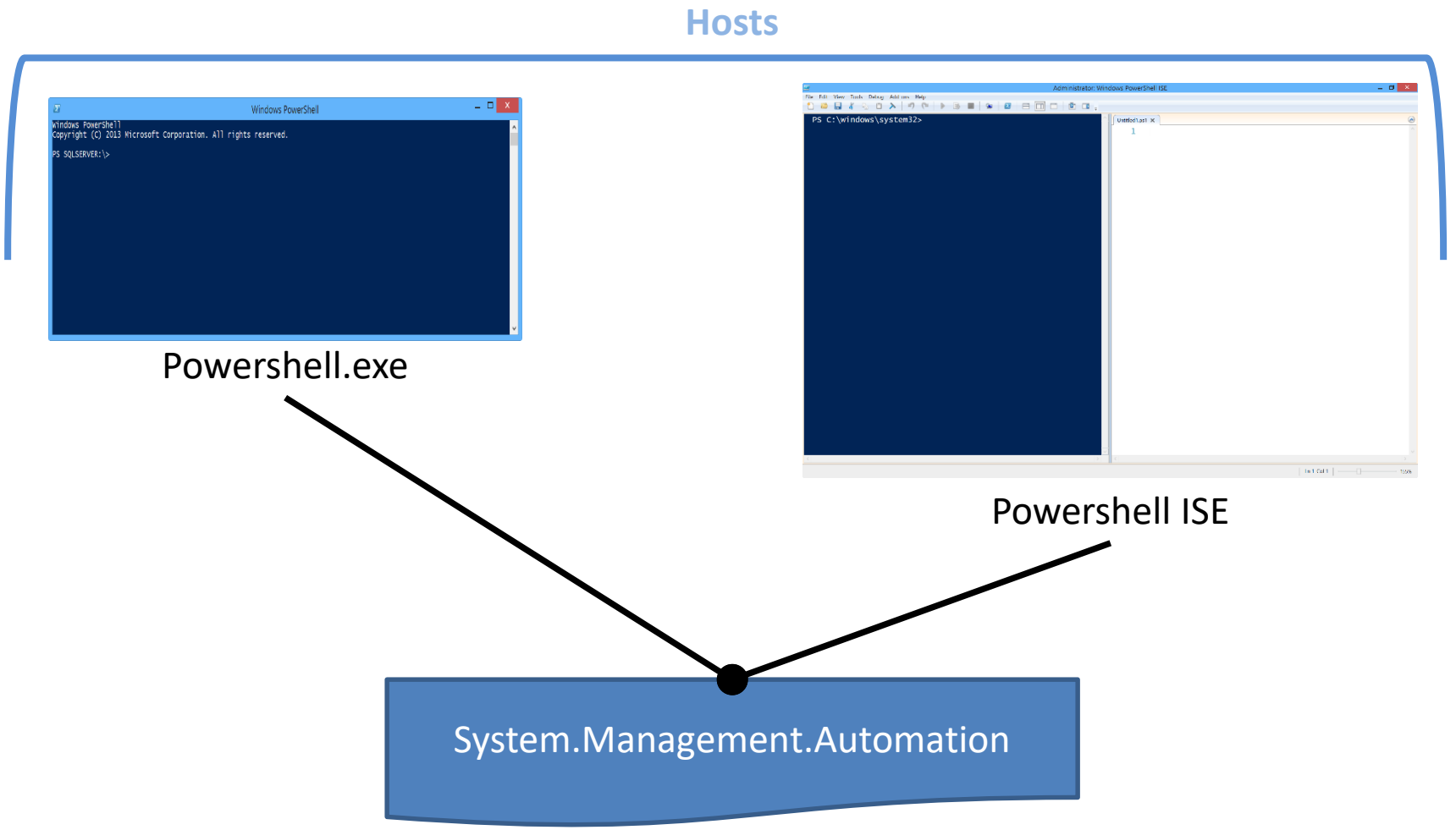
The Facts

- Envisioned by Jeffery Snover – 2002
  - The Monad Manifesto
- Released as Powershell RC 1 – April 2006
  - Originally called Project “Monad”
- Current available version: 5.1
  - Just released, 5.0 has been out for a bit

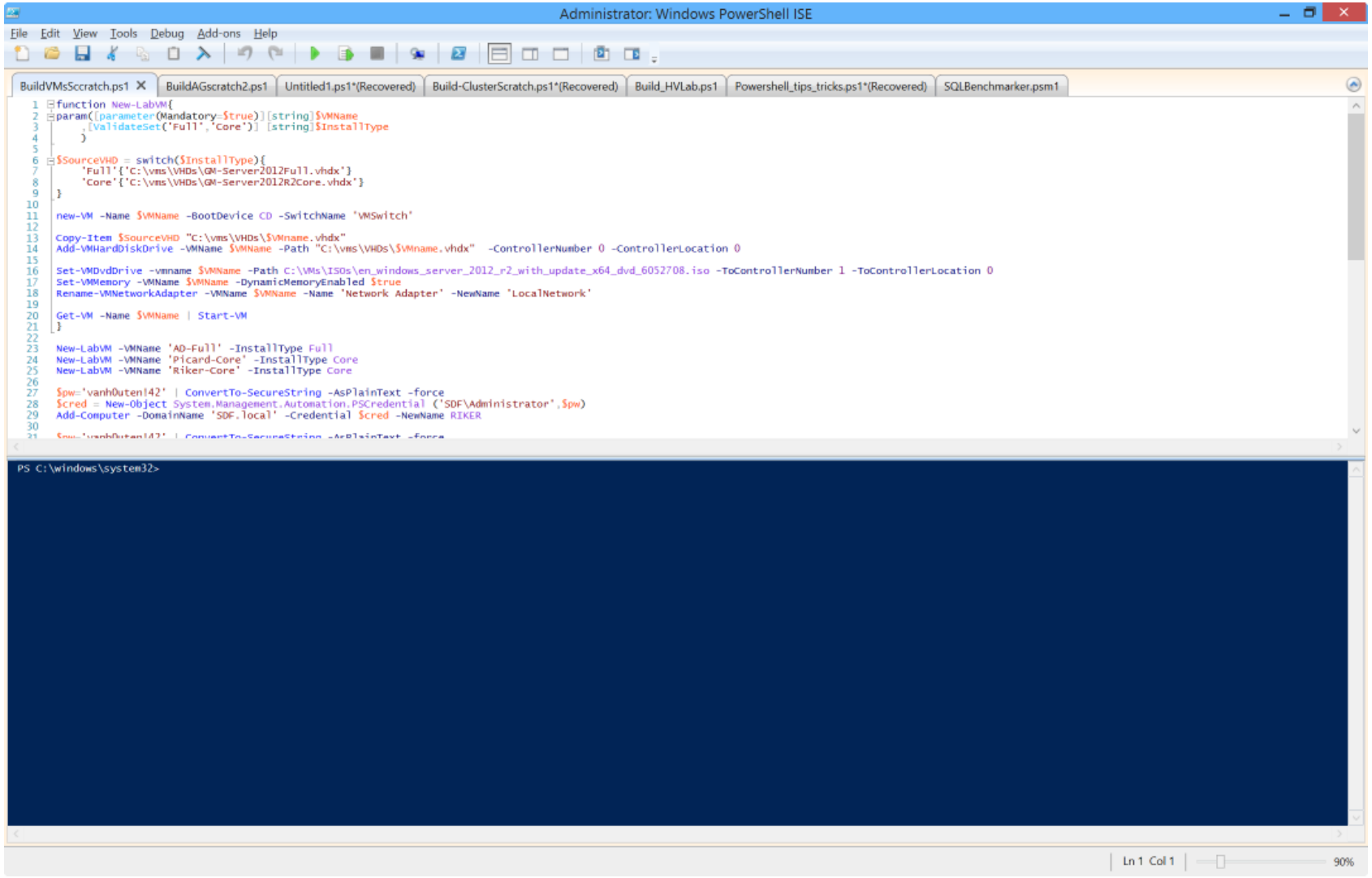
Why PowerShell?



Pieces and Parts



Demo – The ISE



Mike Fal - www.mikefal.net

Writing Powershell

The Lighting Tour of the Language

Cmdlets

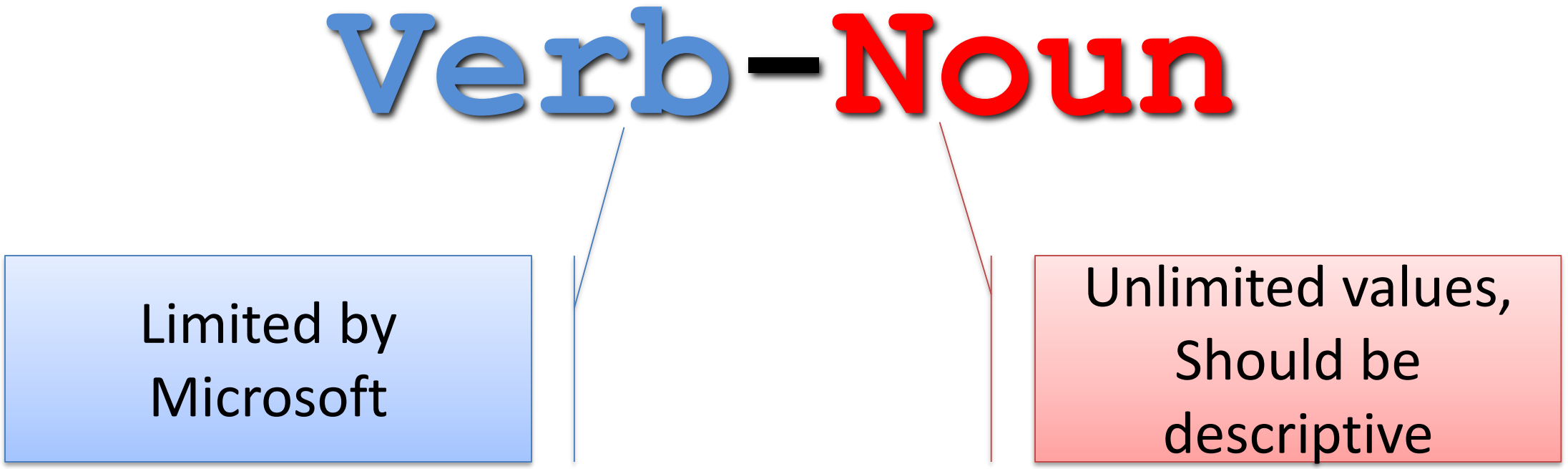
Variables

Control Logic and Flow

Errors and Error Handling

cmdlets

Fundamental unit of “getting stuff done”



cmdlet examples

```
Get-Help
New-Item
Remove-Module
```

Learn Within Powershell

Get-Command

List Commands:
New
-Module SQLPS

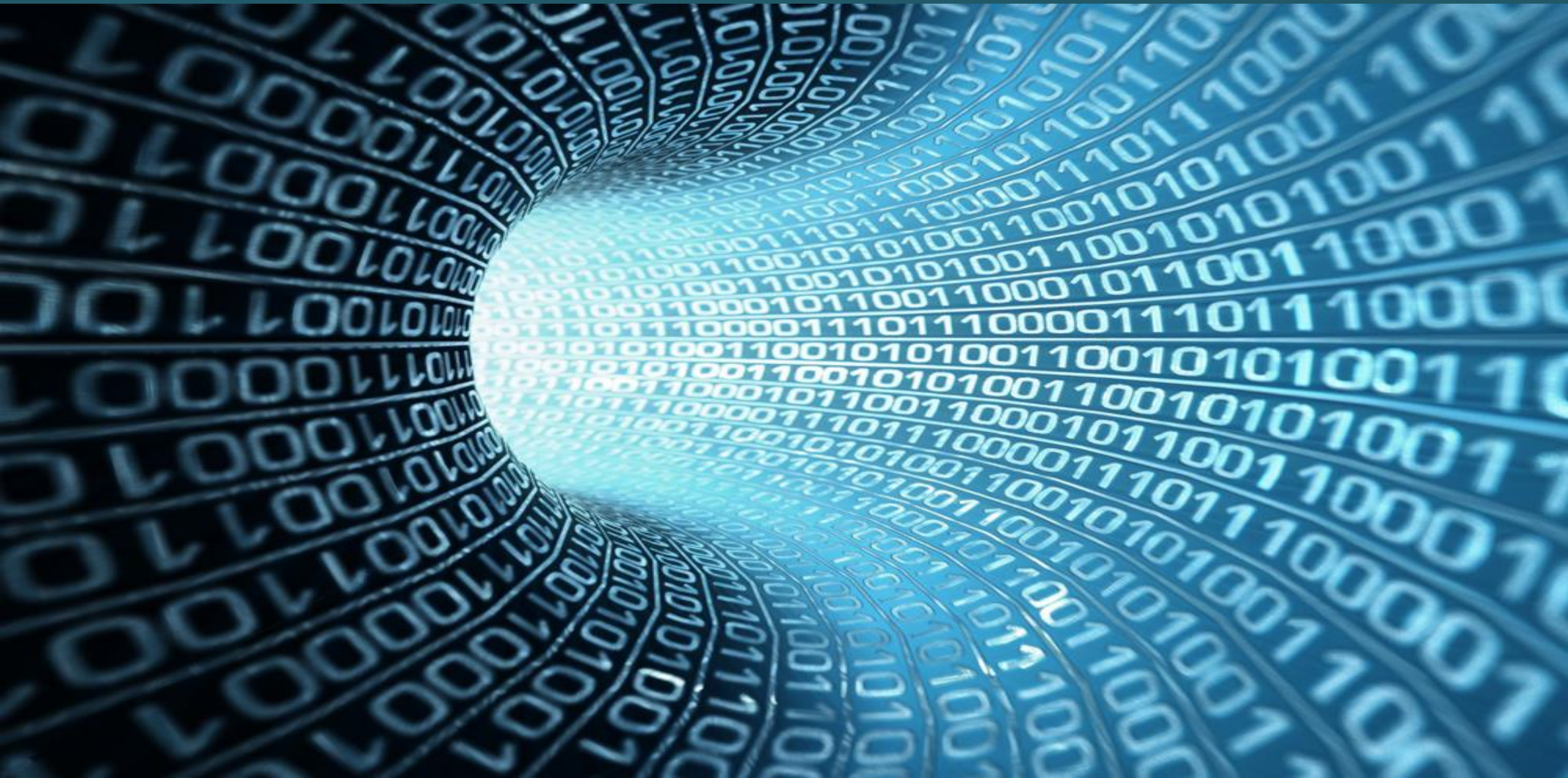
Get-Help

Show help info:
man, help
-ShowWindow

Get-Member

Methods and properties

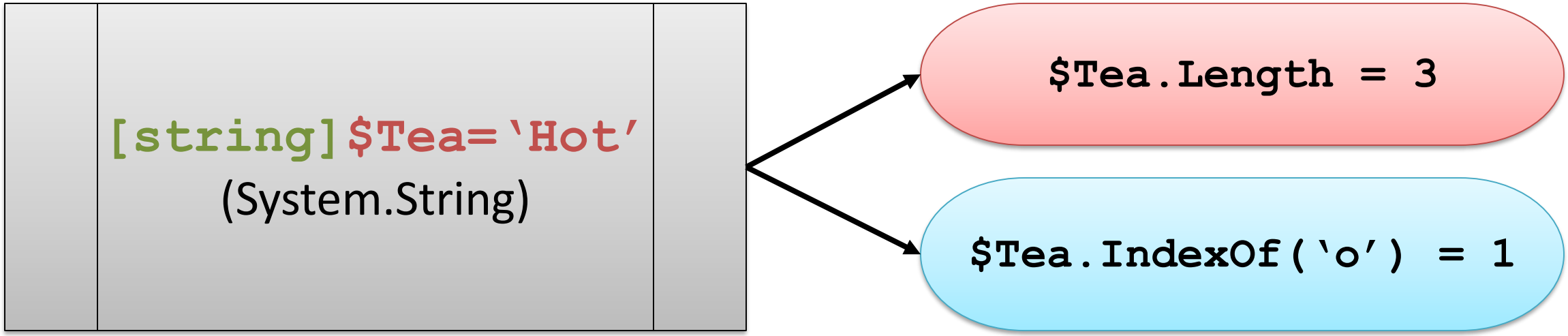
Demo - CMDLETS



Object Oriented Thinking

Everything is a .Net object!

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

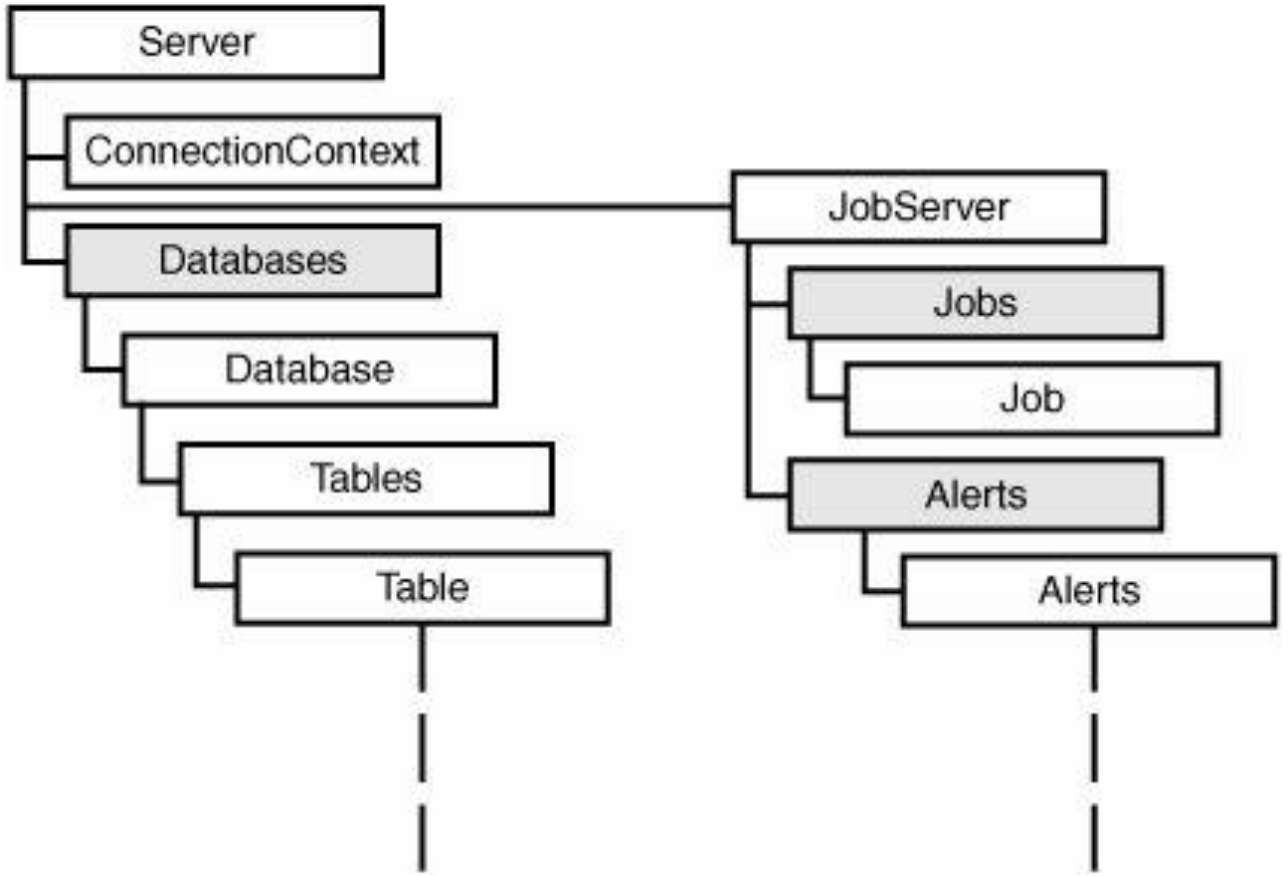


Variables

‘$’ indicates a variable

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

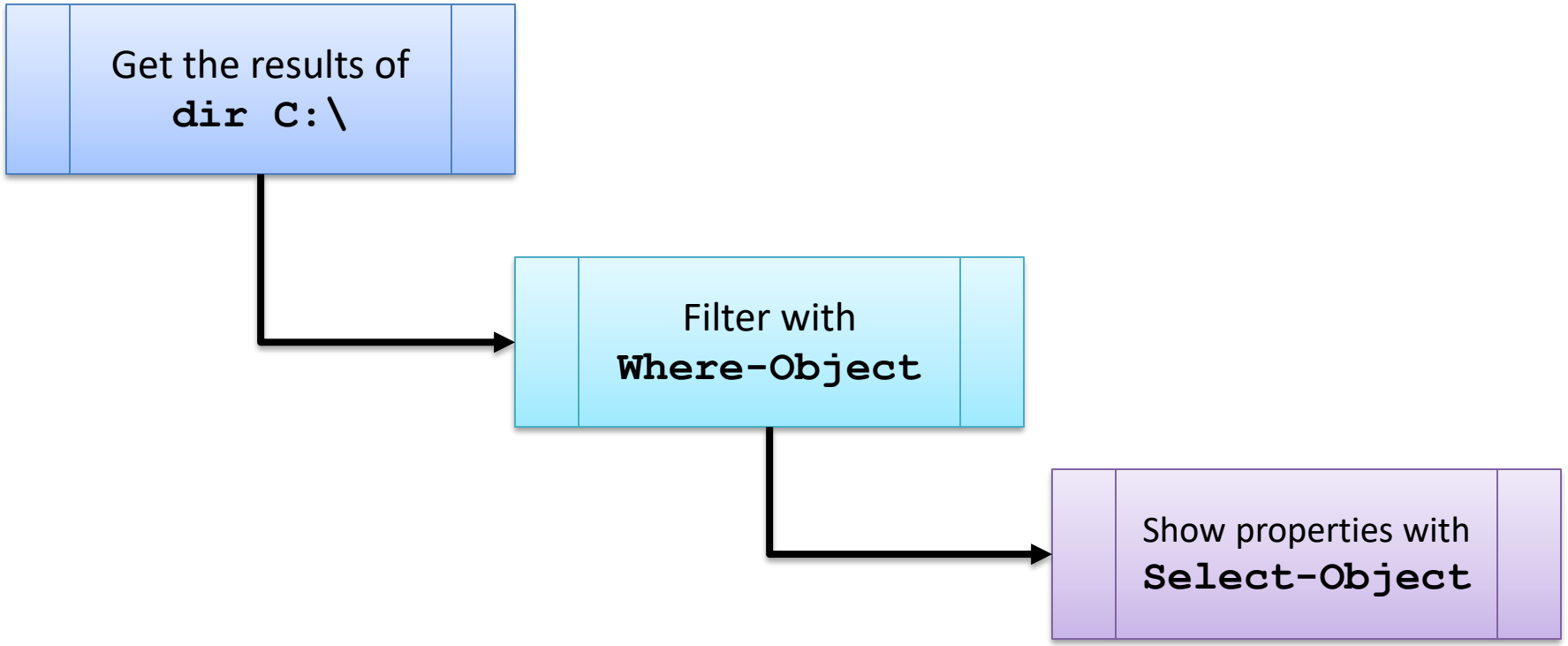
Demo – Objects, and Variables



Pipeline

How does it work?

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



Components of the Pipeline

Cmdlet Support

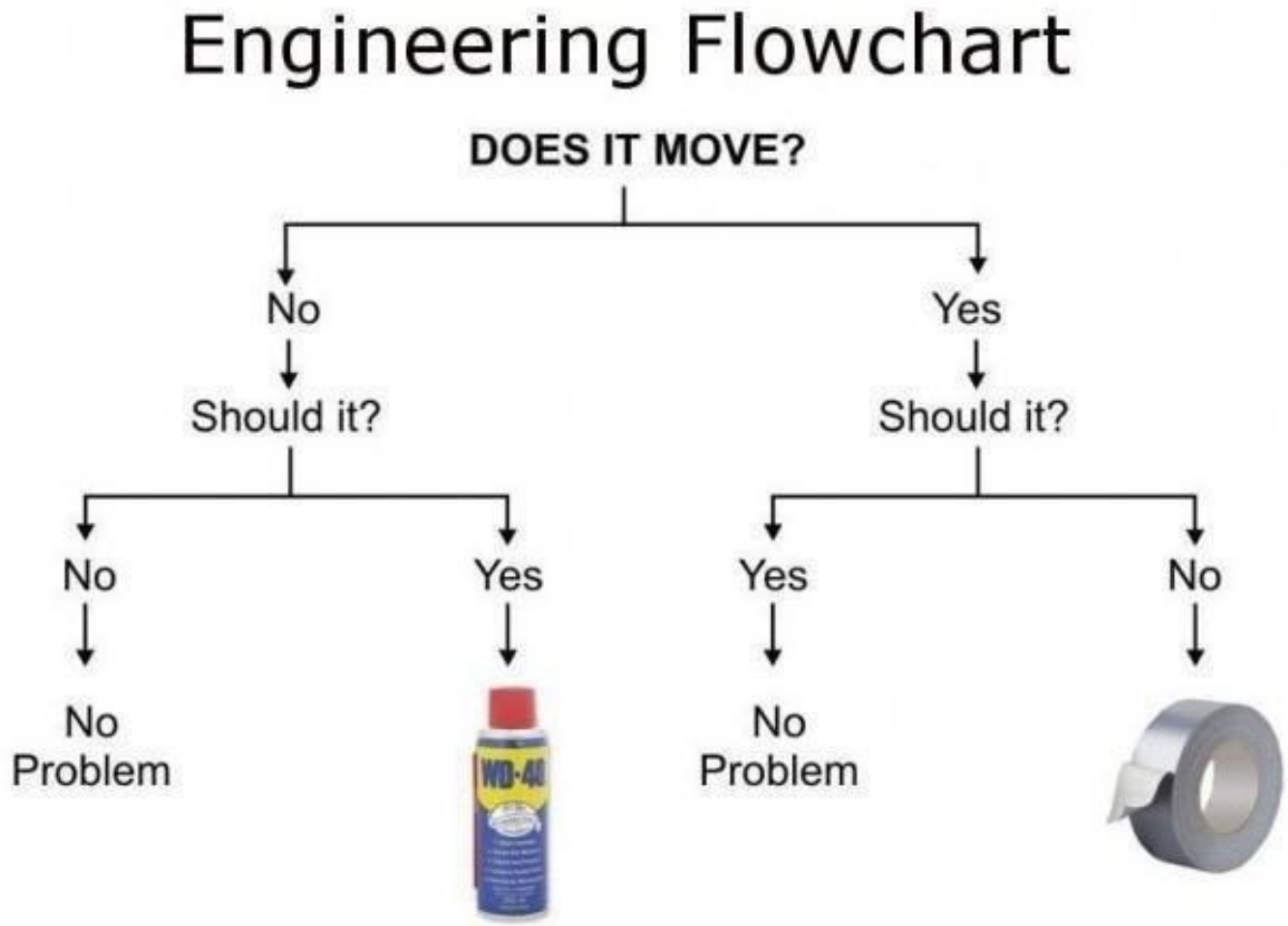
- Accepts Pipeline Input (check help file)
- Usually -InputObject

Looping

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

Demo - Pipeline

Control Logic



Operators

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

Control Flow - Conditionals

```
If(<condition>){
     <script block>
}
Else {
     <script block>
}

Switch(<item>){
     <value> { <script block> }
     <value> { <script block> }
     …
}
```

Control Flow - Loops

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

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

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

Break – Exit Loop

Error Handling

`$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
```

Error Handling

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

Throw – Create an exception

Demo – Control Flow/Error Handling

Review

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?

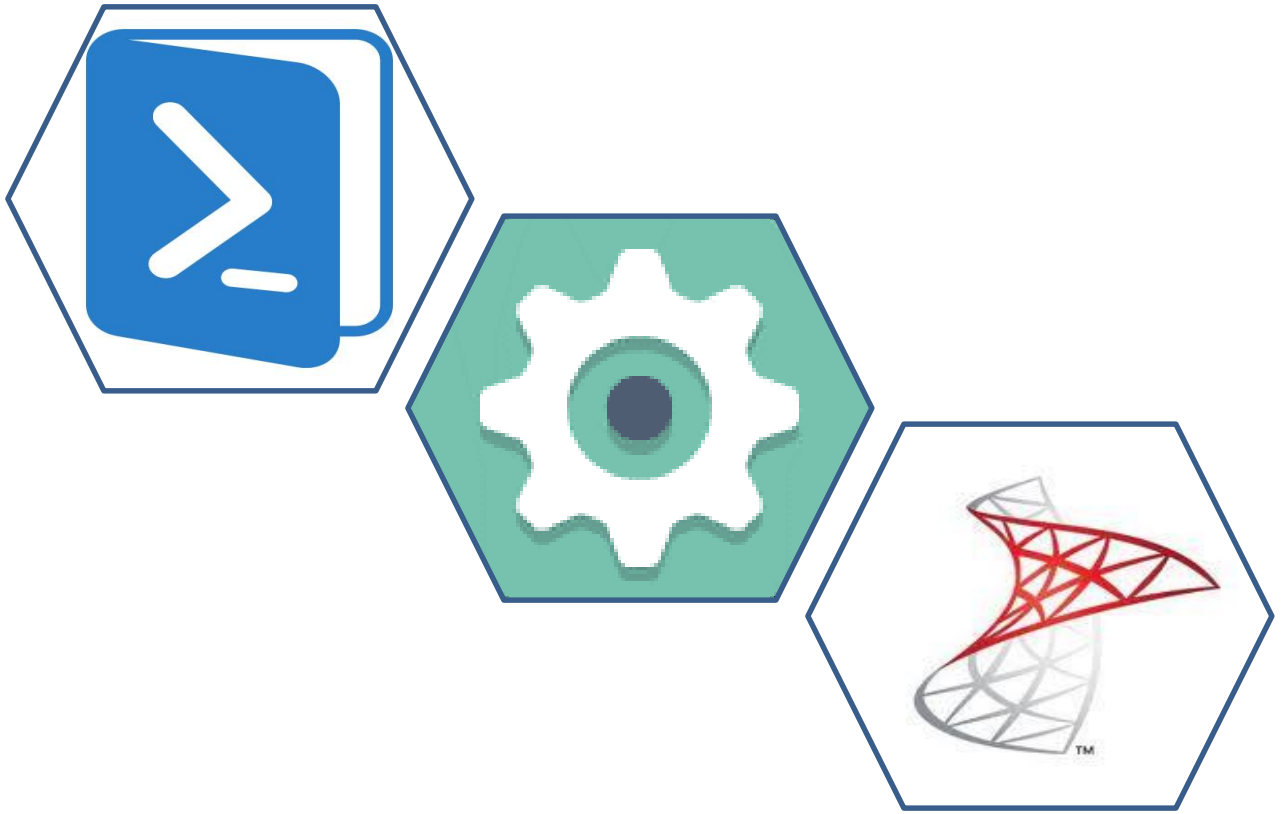
Use the pipeline

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

ForEach(){ }

Practical Use

Powershell And SQL Server



Ways to work with SQL Server

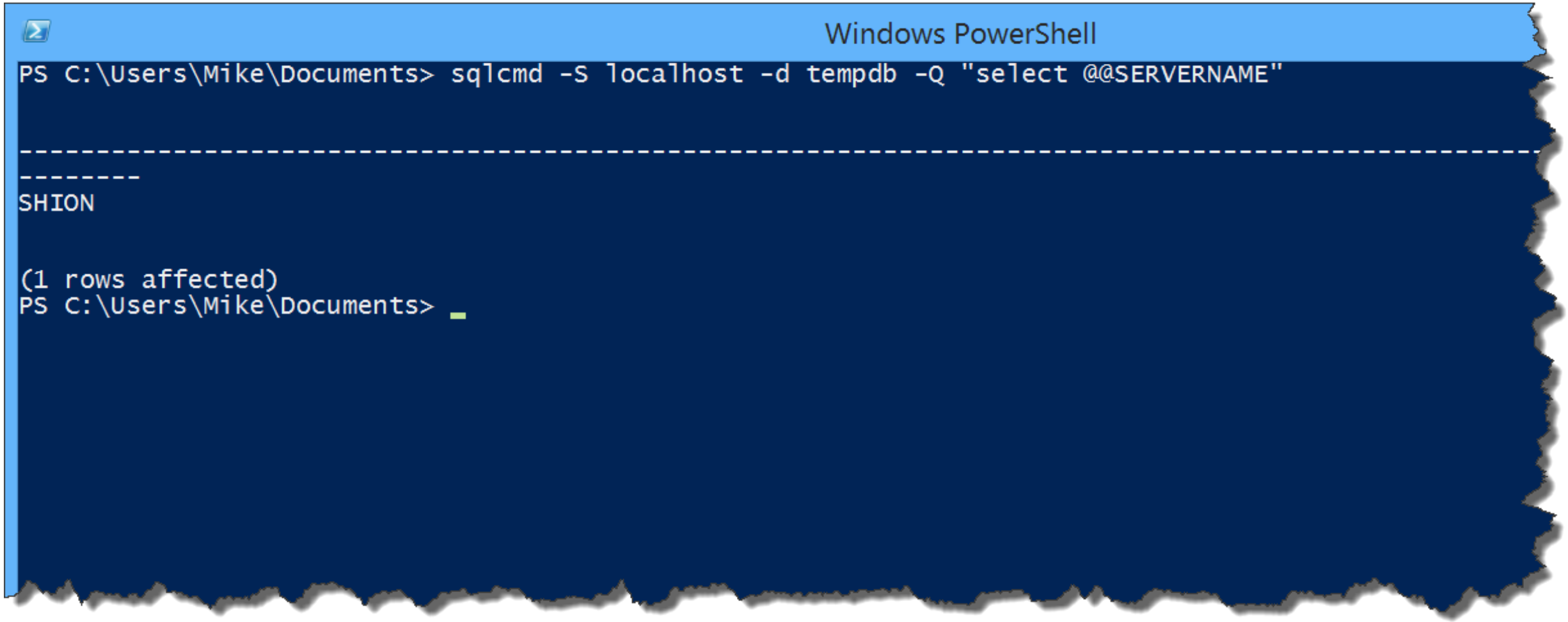
Command line (sqlcmd)

The SQL Powershell Module (SQLPS)

Server Management Objects (SMO)

Powershell in the SQL Agent

sqlcmd


Windows PowerShell
PS C:\Users\Mike\Documents> sqlcmd -S localhost -d tempdb -Q "select @@SERVERNAME"

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

The Basics

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

SqlServer

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

```
Import-Module SqlServer
```

Installation

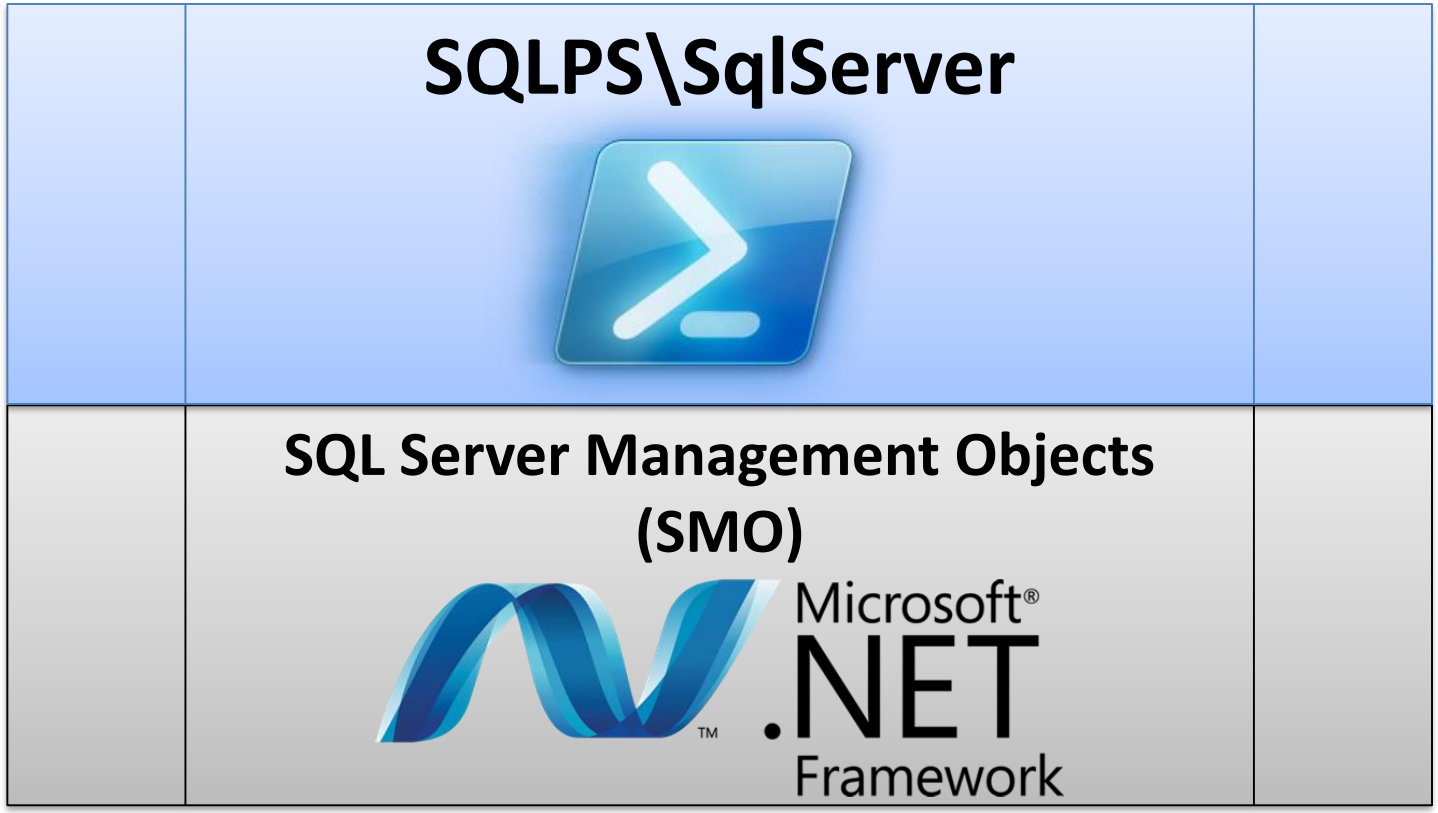
- Not installed specifically
- Included anytime you install ANY SQL Server component
- SQLPS Installation Path:

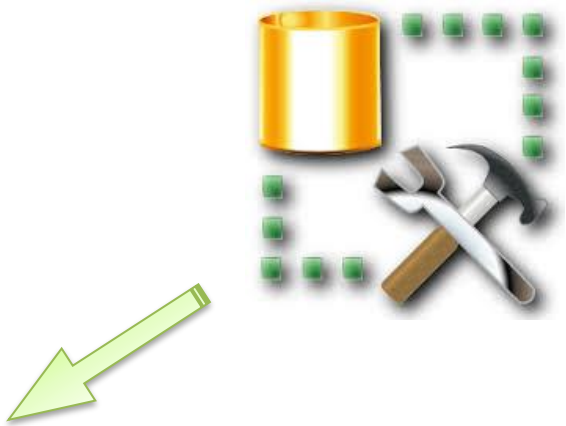
```
C:\Program Files (x86)
   \Microsoft SQL Server
   \130\Tools\PowerShell\Modules
```

- SqlServer Installation Path

```
C:\Program Files\WindowsPowerShell\Modules
```

Built on the SMO





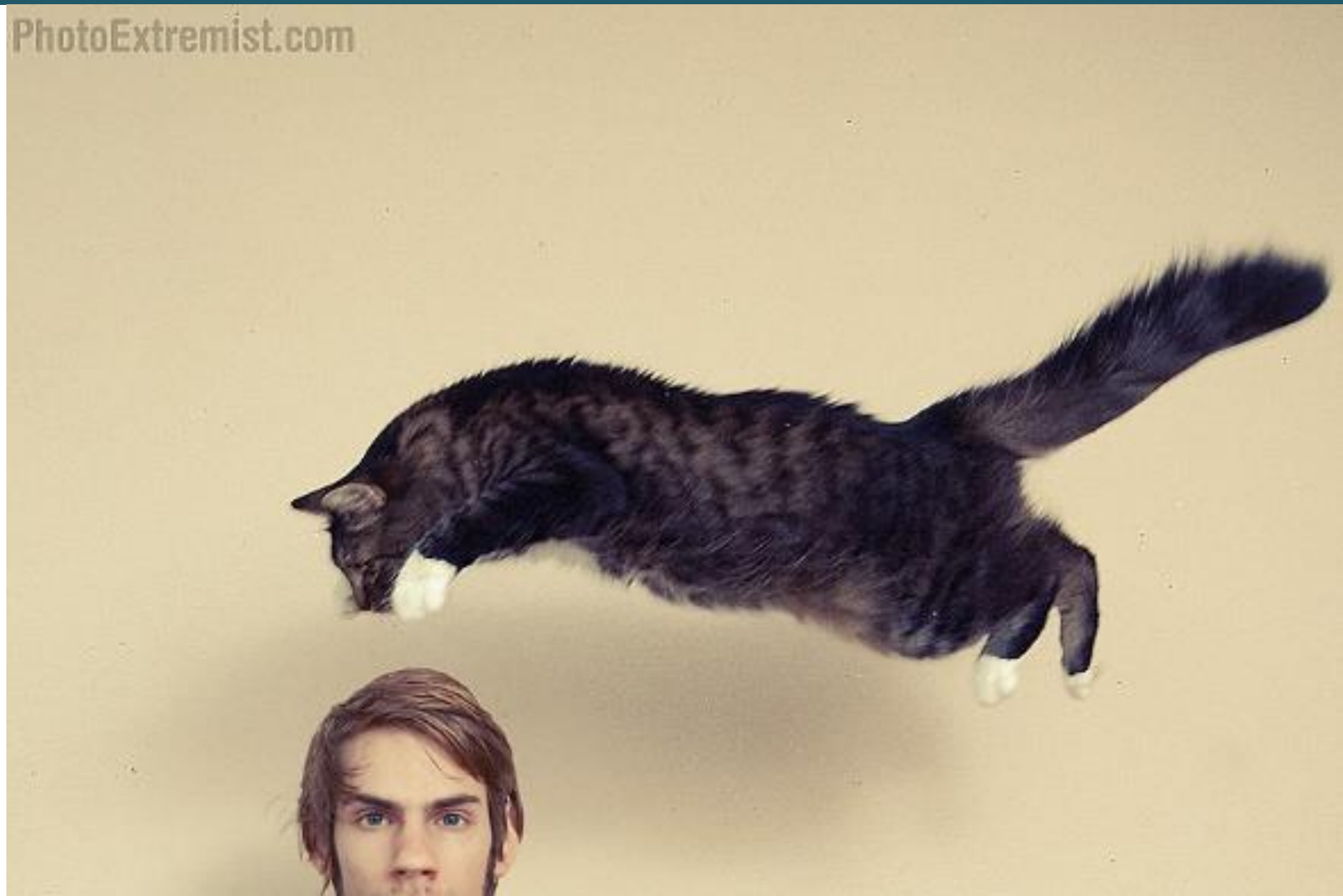
.Net Library

requires .Net 2.0

What Makes It Work?

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	

DEMO!



SQL Server and the File System

Providers

```
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

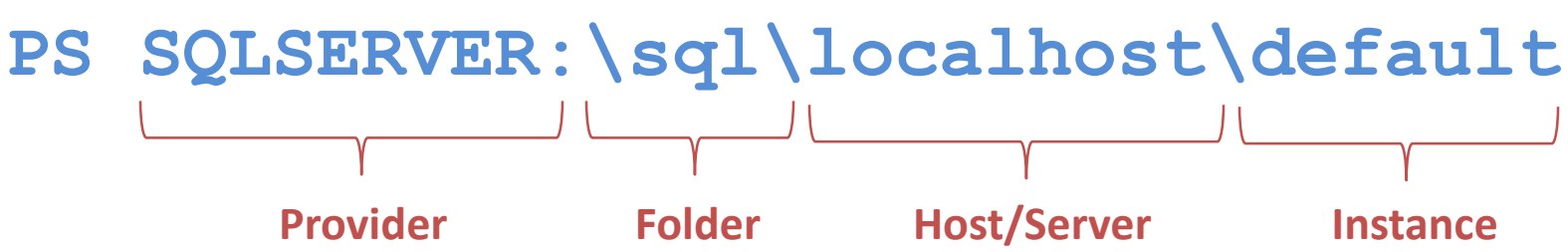
Explore/Use like a FileSystem

SQL Server Provider

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
```



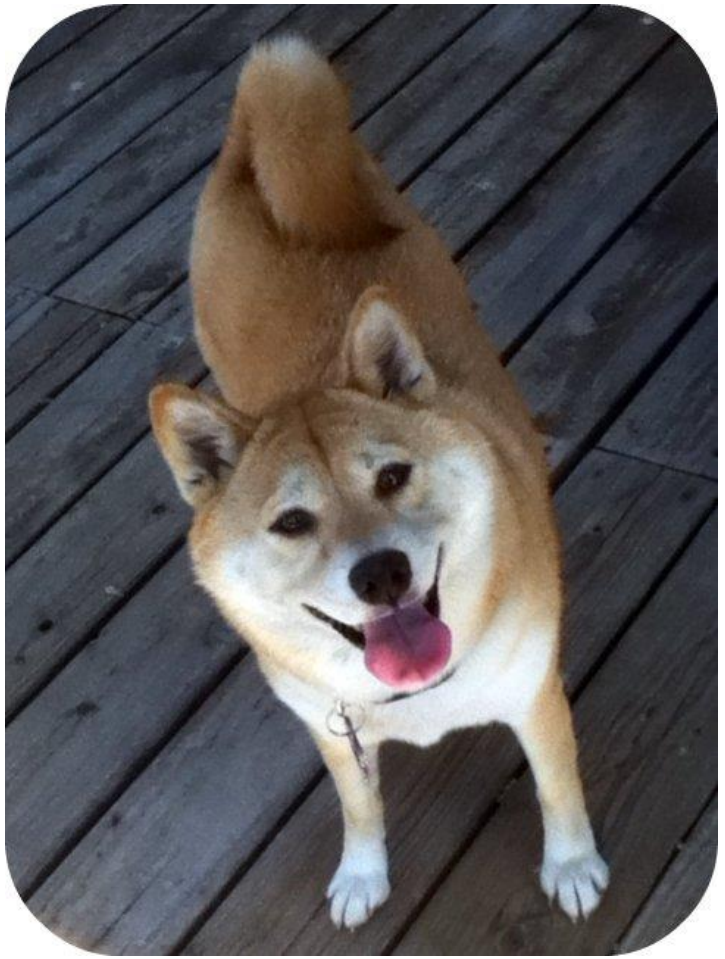
DEMO!

Gotchas

Timeouts!

Tab completion and Intellisense

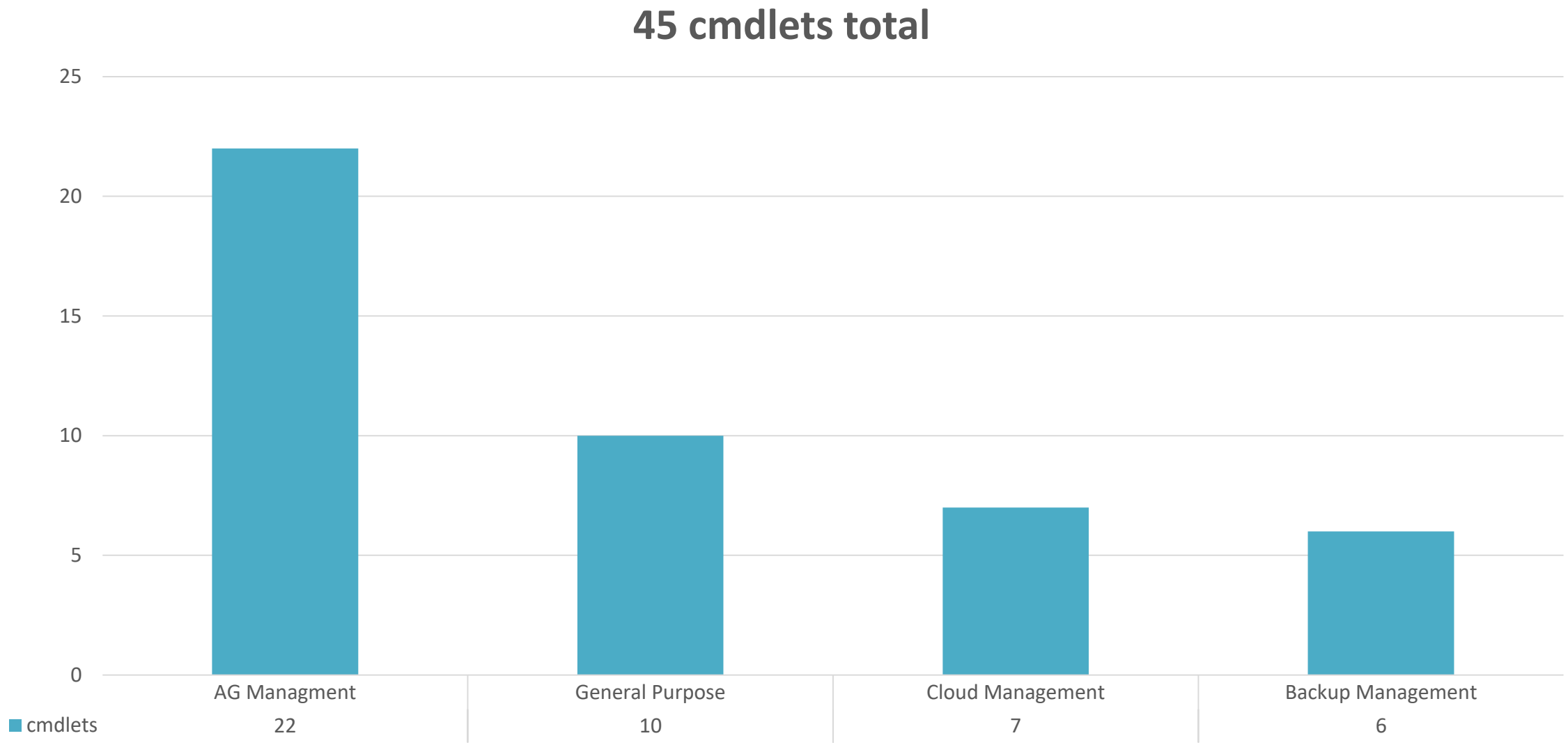
Contextual behavior



SQLPS Cmdlets

Get-Command

Cmdlet Overview



General Purpose

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 and Restore

Backup-SqlDatabase
Restore-SqlDatabase
New-SqlBackupEncryptionOption
Get-SqlSmartAdmin
Set-SqlSmartAdmin
Test-SqlSmartAdmin

Standard
Backup and Restore
operations

SqlSmartAdmin cmdlets
for Azure Managed Backups

Cloud Management

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

Availability Group Management

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

With SSMS July 2016

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

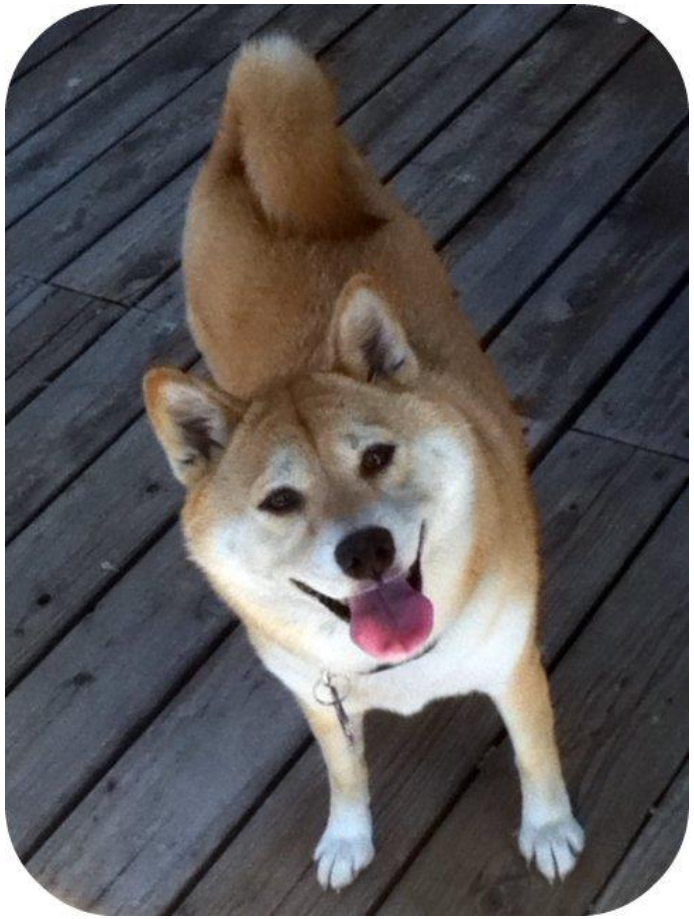
Get-SqlErrorLog
Set-SqlErrorLog

DEMO!

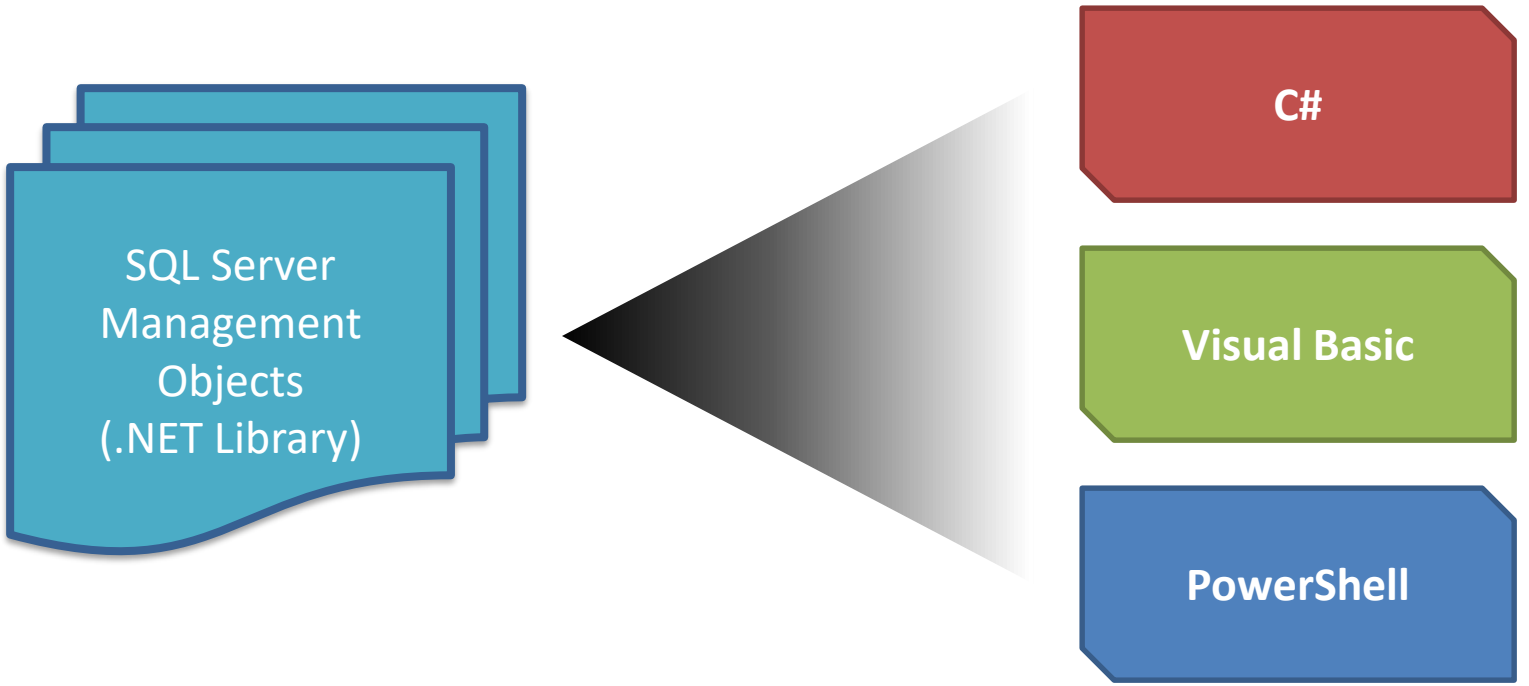
Gotchas

Poorly Documented
(Changing!)
Unexpected behavior

Lack of support
(Changing!)



SMO



Requires .NET framework 2.0

What does it give us?

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

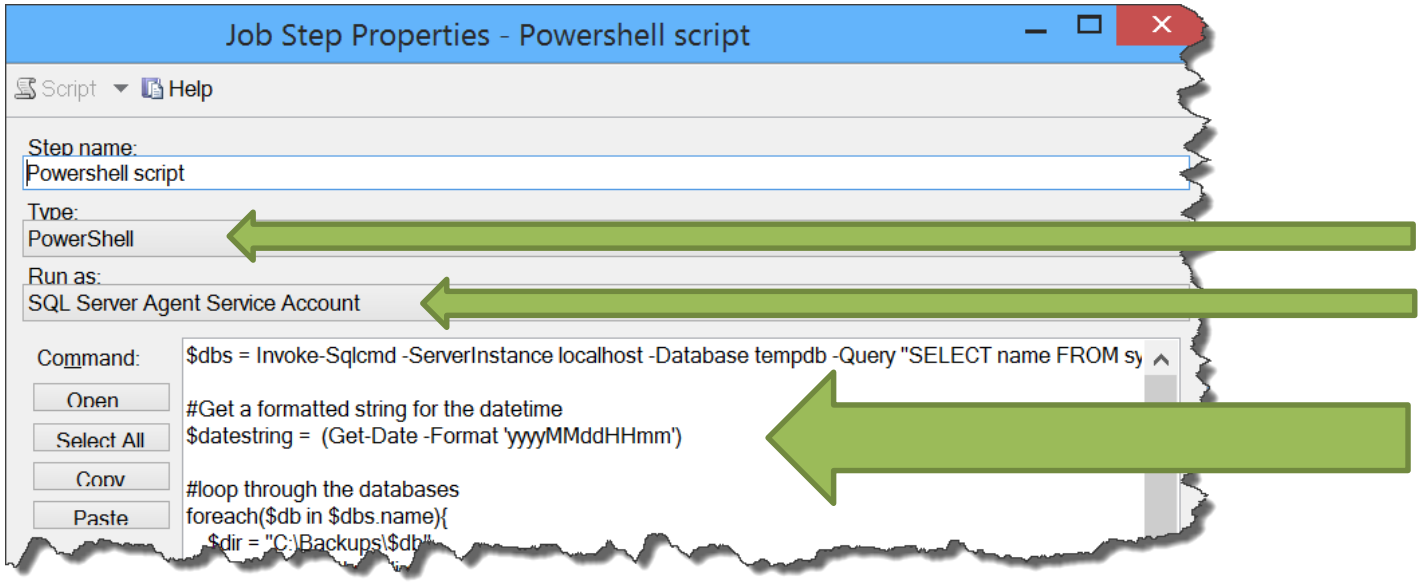
Demos – Using the SMO

Powershell in SQL Agent Jobs

Use SQL Agent to mange jobs

Easily integrate Powershell with your tasks

What does it look like?



Step Type

Run As

Script

Uses SQL Server's "mini" shell

Powershell 2.0 (boooo)

Demo – Powershell and SQL Agent

Review

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

Practical Use

Script re-use

Rinse and Repeat

Writing Scripts

Re-using Scripts

Your Powershell Profile

Functions and Modules

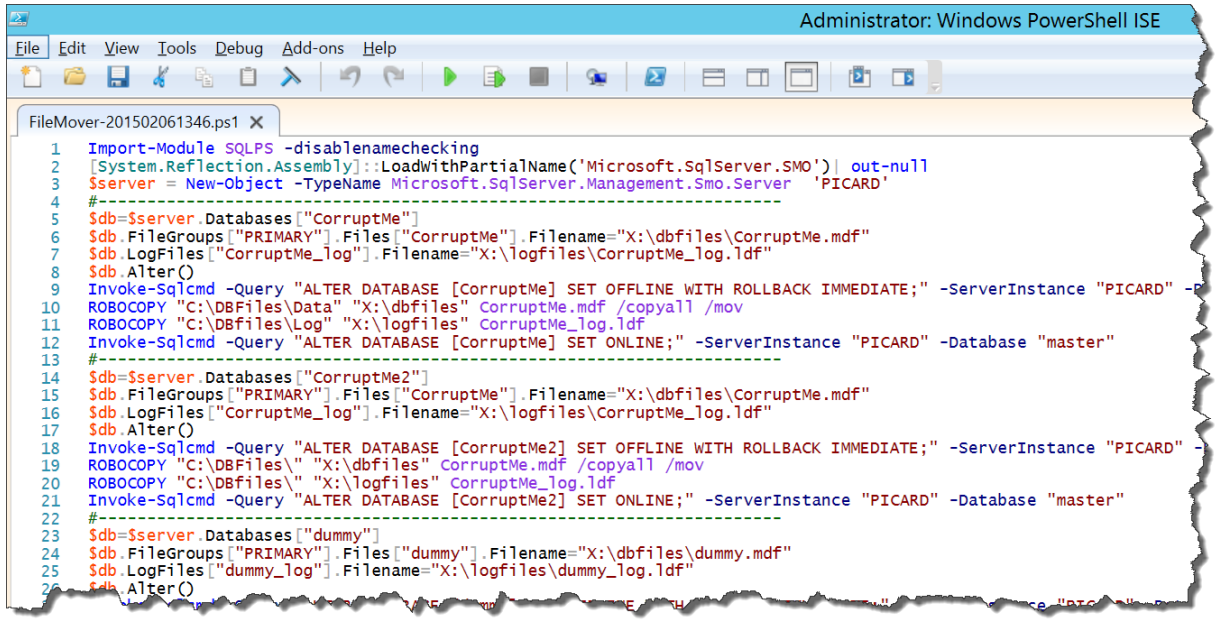
Best Practices and Documenting your scripts

Saving Scripts

Scripts are saved as .ps1 files

Can be run by calling them at the command line

Also can leverage dot sourcing



Code Re-use: Scripts

Powershell Scripts can be parameterized

```
param($VariableName)
```

Parameters are referenced

- Positionally
- Variable name becomes parameter name

Code Re-Use: Your Profile

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
```

Code Re-use: Functions

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
```

Code Re-use: Module

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:

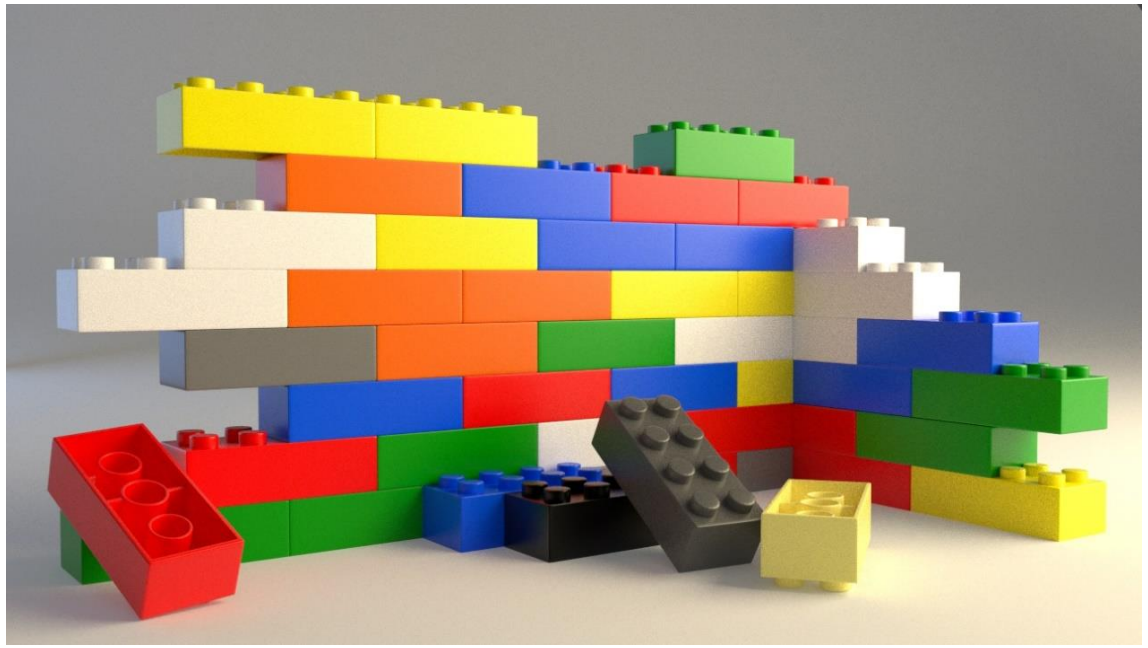
C:\Users\<user>\Documents\WindowsPowerShell

C:\Program Files\WindowsPowerShell

```
Get-Help about_Modules
```

Demo – Functions, Modules, Profile

Best Practices



- 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.

Comment Based Help

```
<#
    .SYNOPSIS
    By using a specific formation, you can write your
    own Get-Help information
#>
```

```
Get-Help about_comment_based_help
```

Review

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?

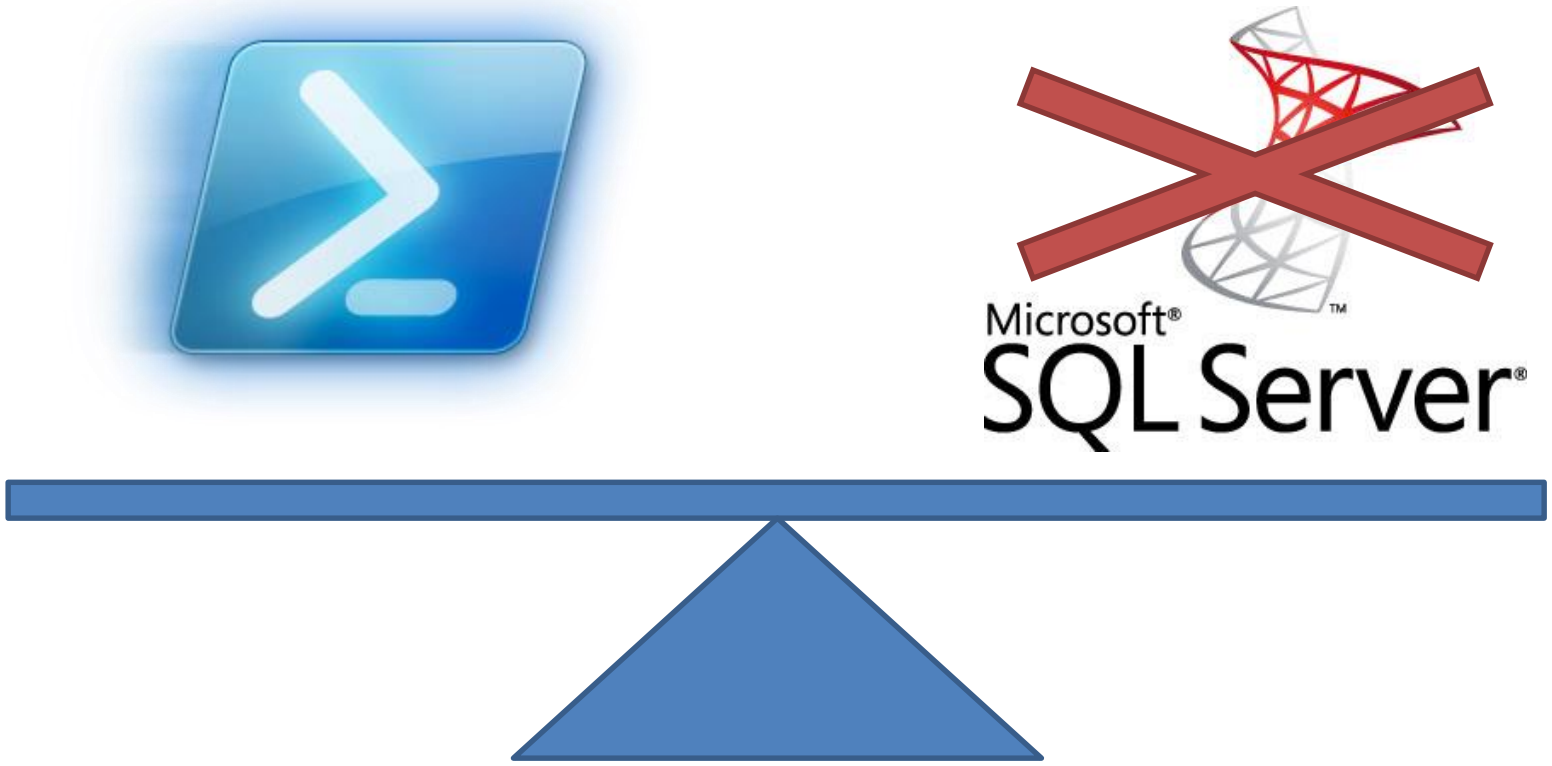
Import-Module

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

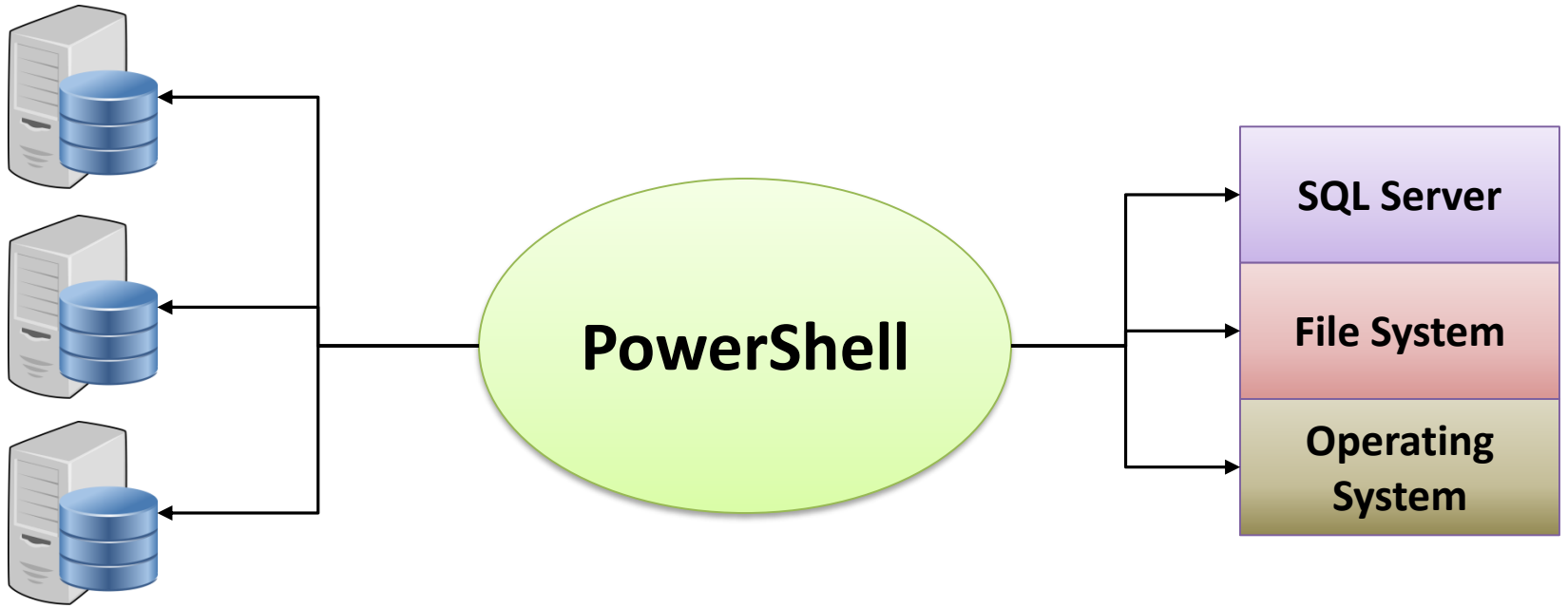
It must go at the at the beginning (can be after a comment block)

Practical Use

The Approach



What to Consider



Demo – Practical Scripts

Powershell and Server Core

What Server Core IS

Windows Server…
…with no GUI!

Lean – Smaller footprint and install
Stable – Fewer applications and services
Secure – Less security loopholes

Tools to Start With

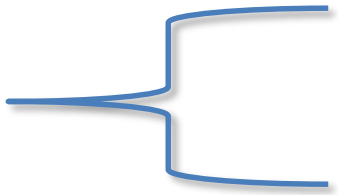
SCONFIG

First Look

Server Core Support

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

SQL Server Requirements



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.

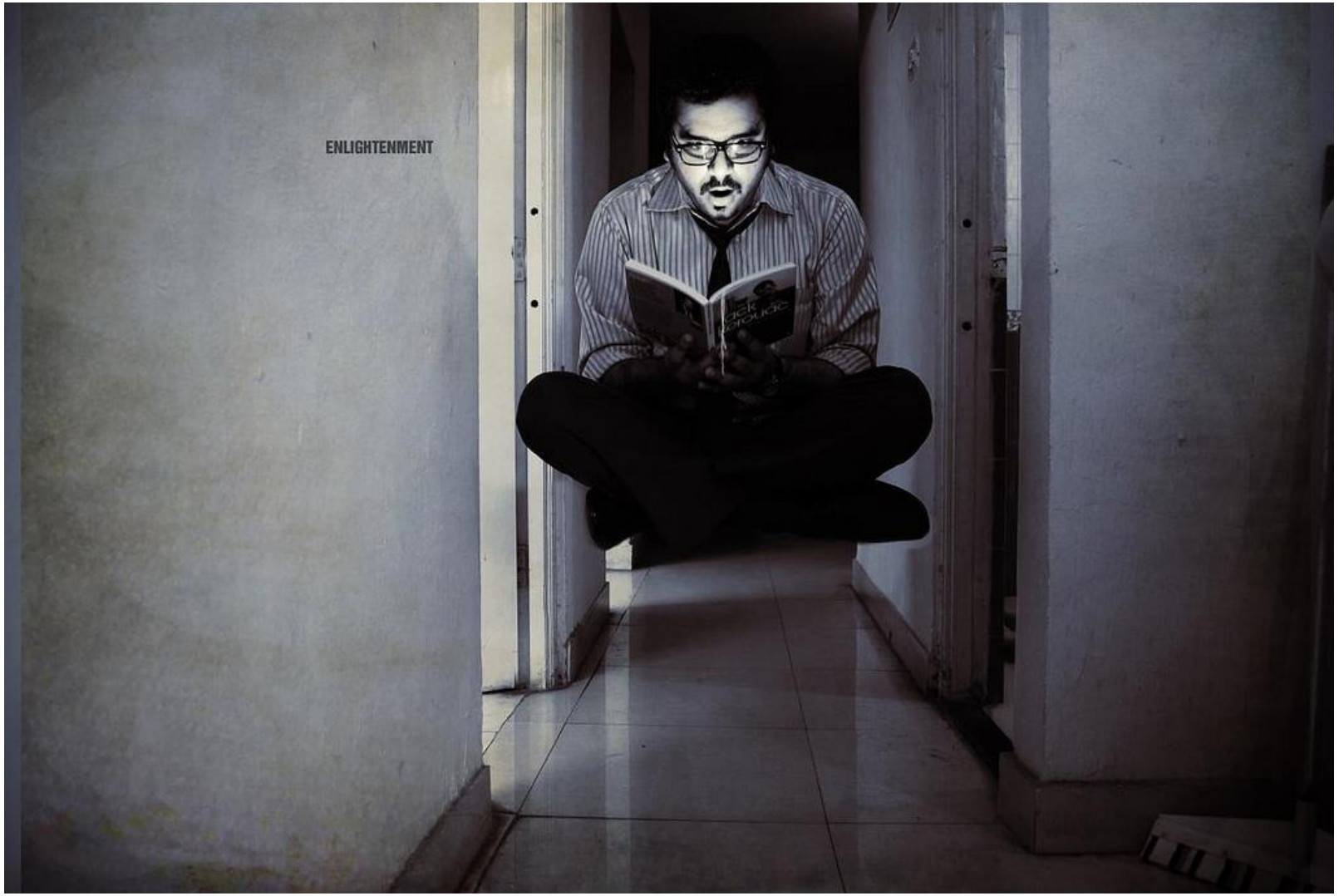
Unattended Install

Well established tool

Allows for consistent SQL builds

Infrastructure as code

Demo - SQL Install



Powershell, Azure, and SQL Server

Getting Connected to Azure

Creating an Azure VM with SQL Server

Setting up and using Azure SQL DB

Microsoft Azure

Azure Powershell Module

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/

Azure Deployments

Classic Deployments

- The “old” way
- Supports managing and creating

```
Add-AzureAccount
```

Resource Manager Deployments

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

```
Add-AzureRMAccount
```

You need both!

Demo – Connecting to Azure

Azure VMs with SQL

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

What do we need?

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.

Creating an Azure VM with SQL Server

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

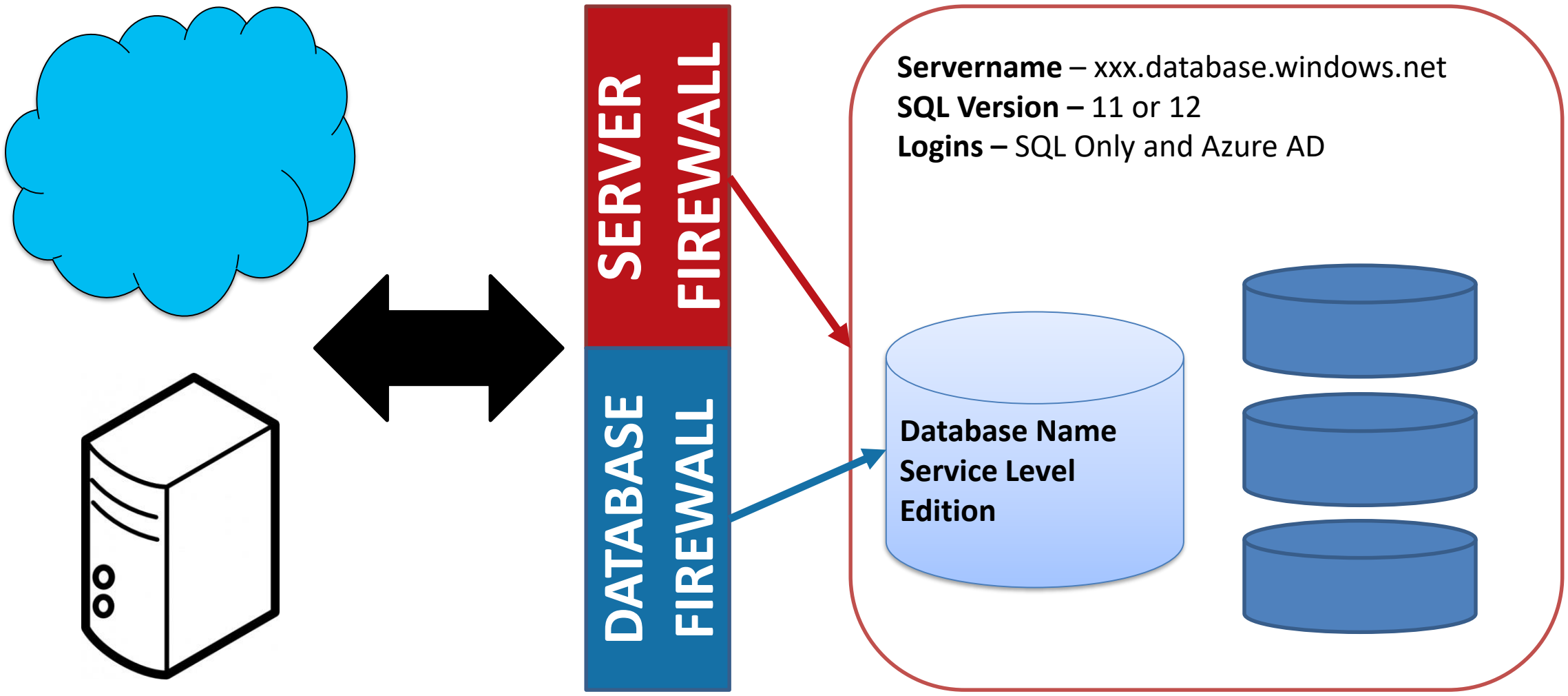
Azure SQL Database

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?



Service Levels

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

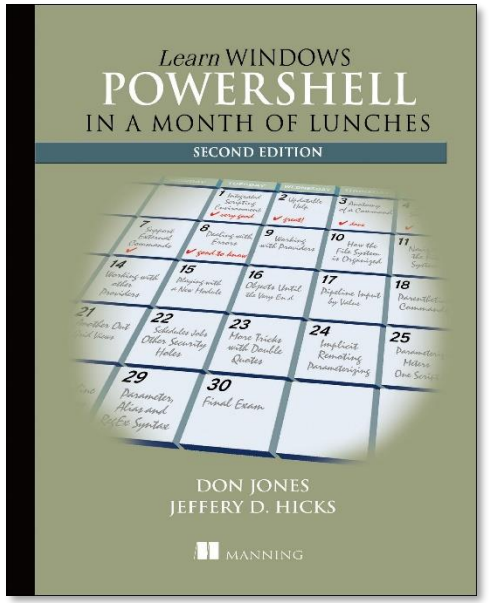
Demo – Creating an Azure SQL Database

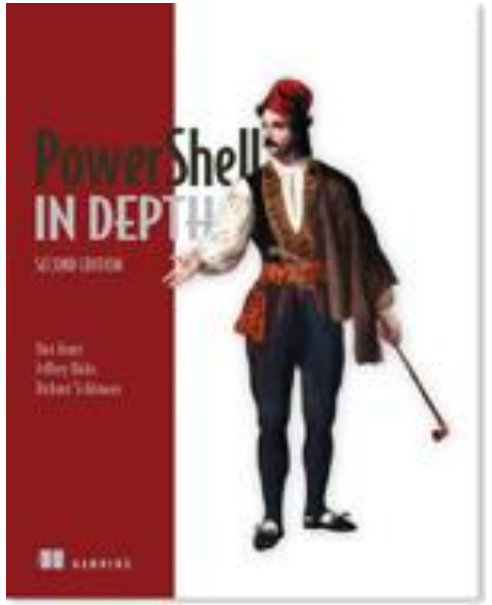
So now what?

Mike Fal - www.mikefal.net

Books

- Powershell In A Month of Lunches
- Powershell in Depth, 2nd Edition





General Powershell

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

Bloggers

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

Online

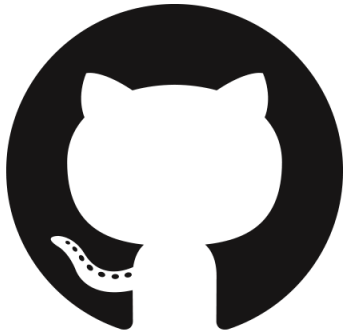
[Microsoft Virtual Academy]

[PluralSight ($$)]

Other scripts

https://github.com/MikeFal

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



Most of all, USE IT

Find tasks to automate

Manage the file system ONLY through Powershell

Rewrite a T-SQL or other task using Powershell

And so on…

Questions

mike@mikefal.net

www.mikefal.net

@Mike_Fal

https://github.com/MikeFal