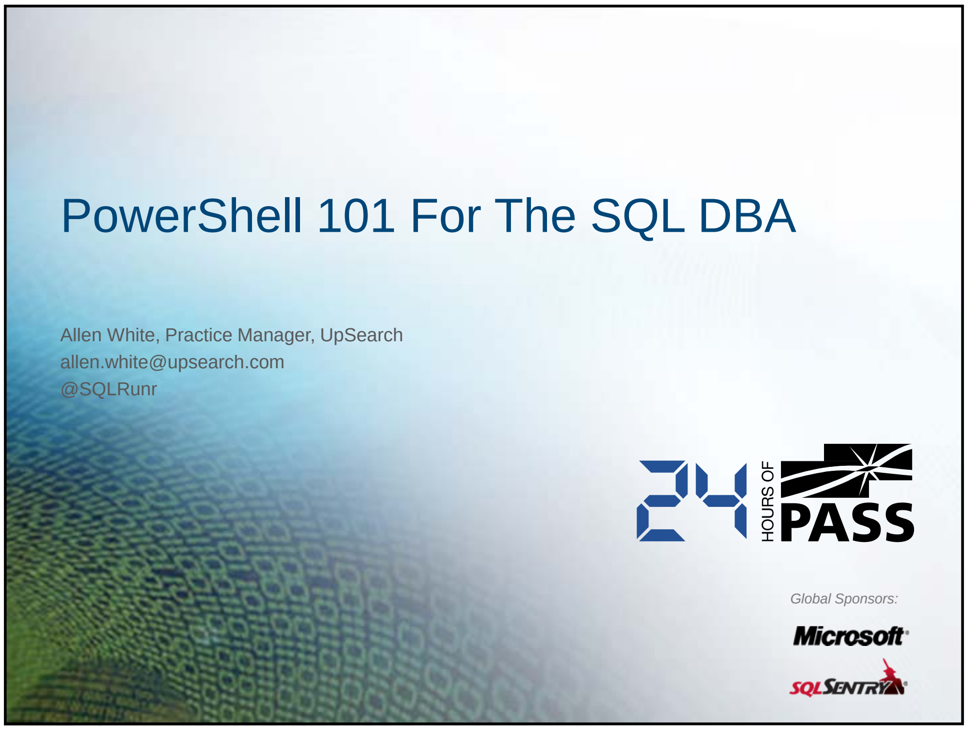


PowerShell 101 For The SQL DBA

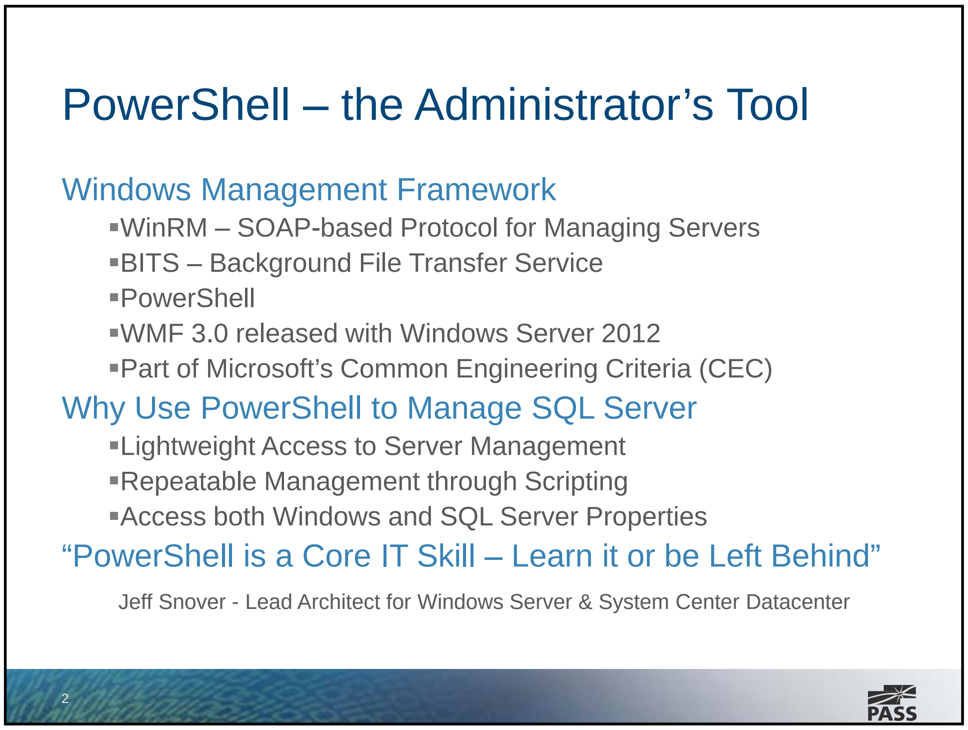
Allen White, Practice Manager, UpSearch
allen.white@upsearch.com
@SQLRunr

24 HOURS OF PASS

Global Sponsors:

Microsoft

SQLSentry



PowerShell – the Administrator's Tool

Windows Management Framework

- WinRM – SOAP-based Protocol for Managing Servers
- BITS – Background File Transfer Service
- PowerShell
- WMF 3.0 released with Windows Server 2012
- Part of Microsoft's Common Engineering Criteria (CEC)

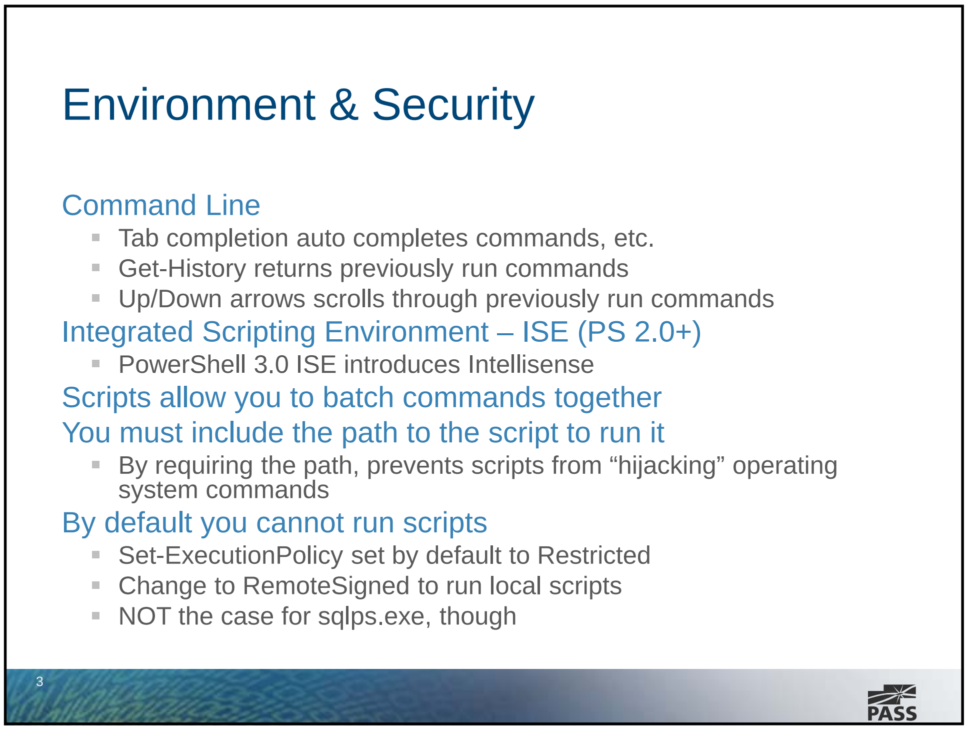
Why Use PowerShell to Manage SQL Server

- Lightweight Access to Server Management
- Repeatable Management through Scripting
- Access both Windows and SQL Server Properties

"PowerShell is a Core IT Skill – Learn it or be Left Behind"

Jeff Snover - Lead Architect for Windows Server & System Center Datacenter

PASS



Environment & Security

Command Line
- Tab completion auto completes commands, etc.
- Get-History returns previously run commands
- Up/Down arrows scrolls through previously run commands

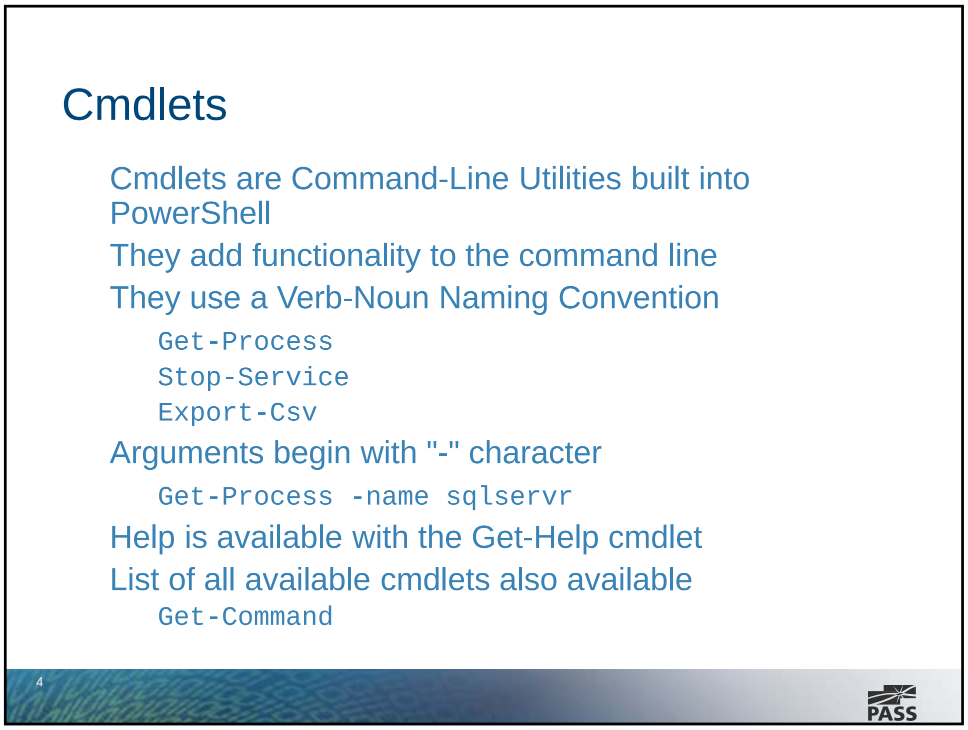
Integrated Scripting Environment – ISE (PS 2.0+)
- PowerShell 3.0 ISE introduces Intellisense

Scripts allow you to batch commands together

You must include the path to the script to run it
- By requiring the path, prevents scripts from “hijacking” operating system commands

By default you cannot run scripts
- Set-ExecutionPolicy set by default to Restricted
- Change to RemoteSigned to run local scripts
- NOT the case for sqlps.exe, though

3



Cmdlets

Cmdlets are Command-Line Utilities built into PowerShell

They add functionality to the command line

They use a Verb-Noun Naming Convention

```
Get-Process
Stop-Service
Export-Csv
```

Arguments begin with "-" character

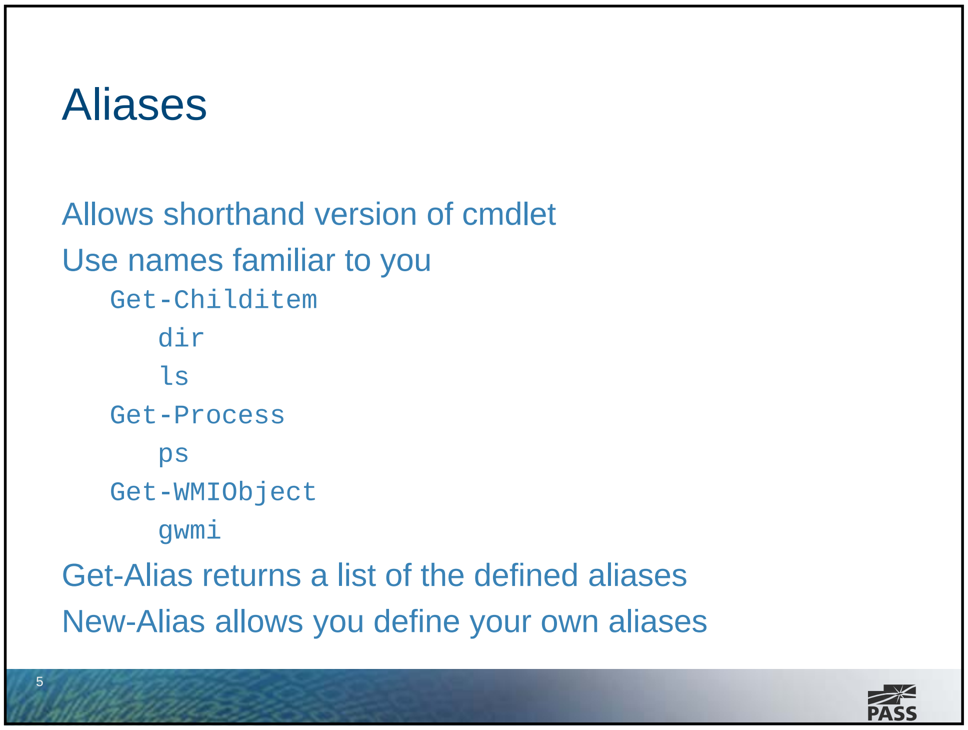
```
Get-Process -name sqlservr
```

Help is available with the Get-Help cmdlet

List of all available cmdlets also available

```
Get-Command
```

4



Aliases

Allows shorthand version of cmdlet

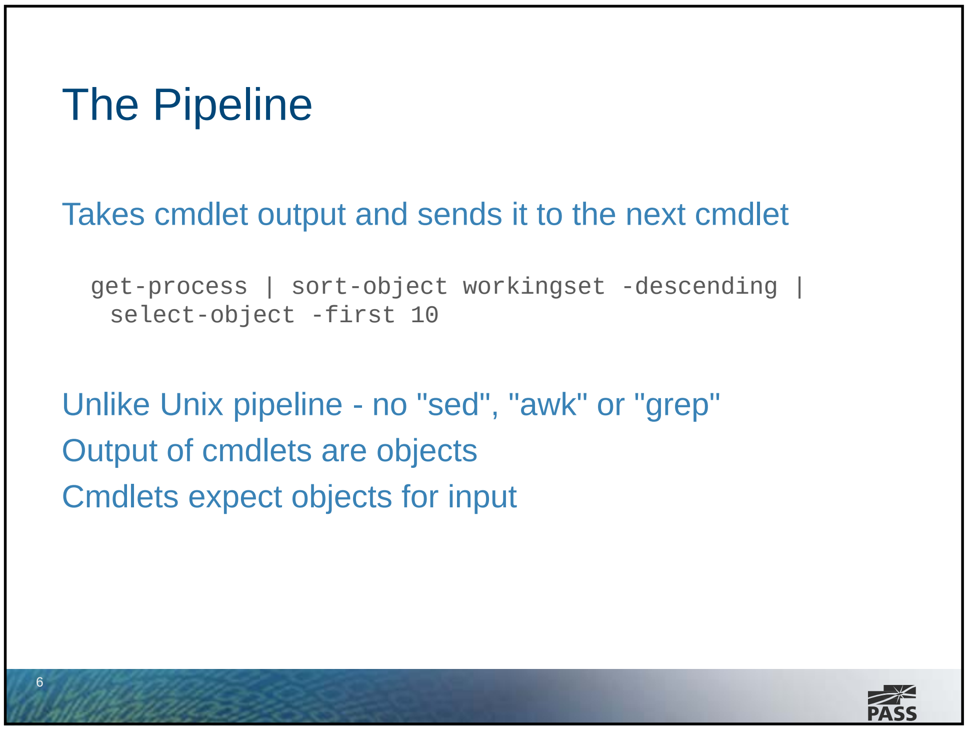
Use names familiar to you

```
Get-Childitem
   dir
   ls
Get-Process
   ps
Get-WMIObject
   gwmi
```

Get-Alias returns a list of the defined aliases

New-Alias allows you define your own aliases

5



The Pipeline

Takes cmdlet output and sends it to the next cmdlet

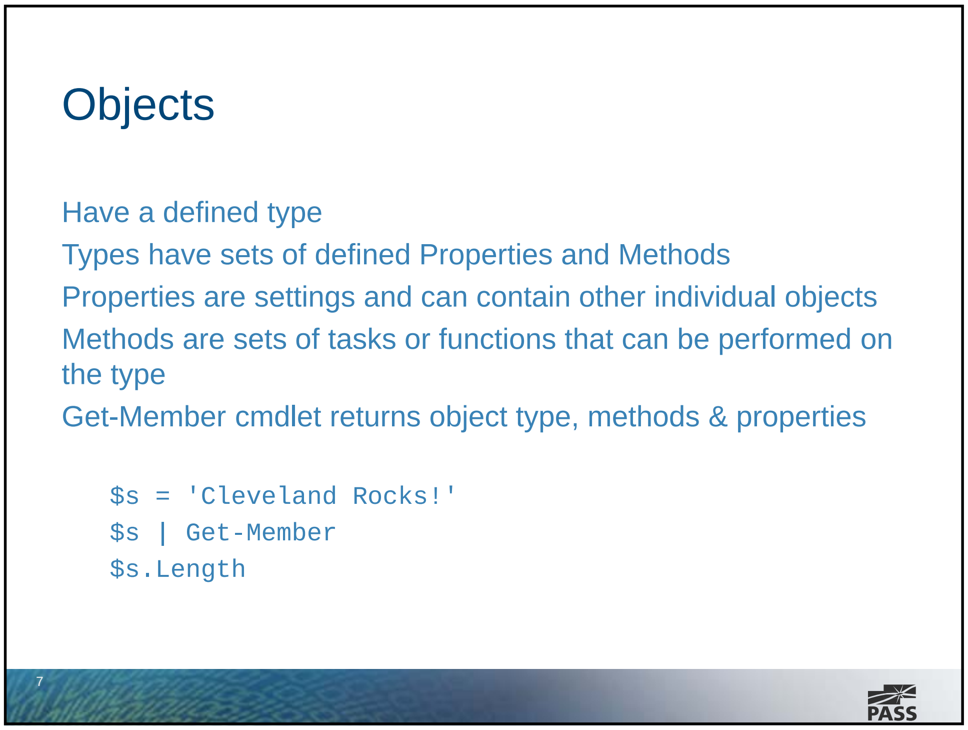
```
get-process | sort-object workingset -descending |
 select-object -first 10
```

Unlike Unix pipeline - no "sed", "awk" or "grep"

Output of cmdlets are objects

Cmdlets expect objects for input

6



Objects

Have a defined type

Types have sets of defined Properties and Methods

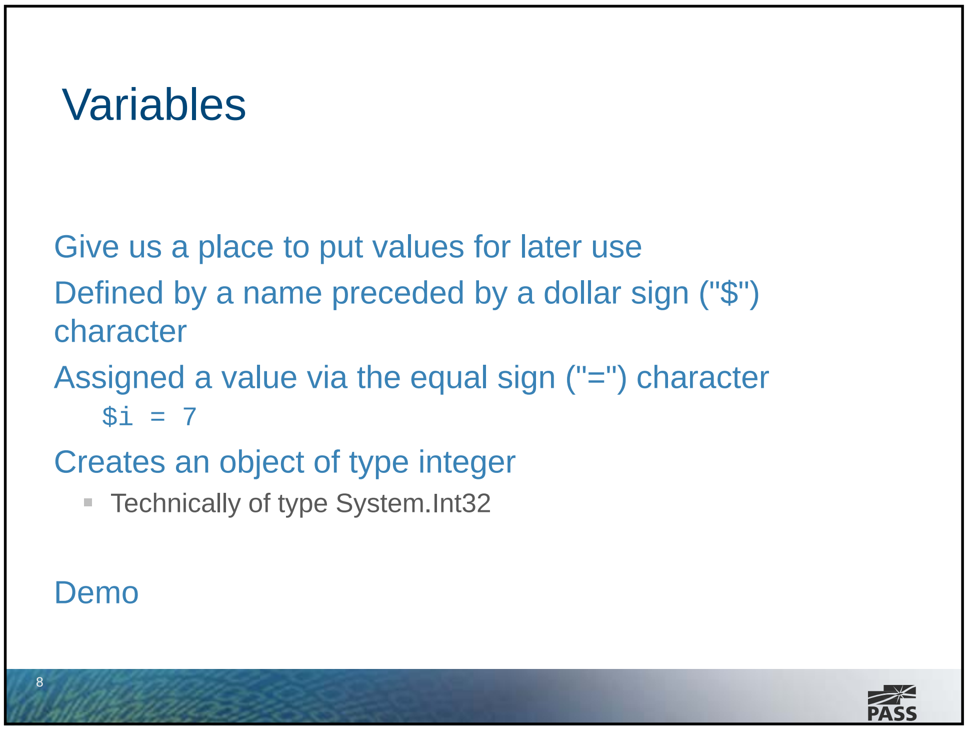
Properties are settings and can contain other individual objects

Methods are sets of tasks or functions that can be performed on the type

Get-Member cmdlet returns object type, methods & properties

```
$s = 'Cleveland Rocks!'
$s | Get-Member
$s.Length
```

7



Variables

Give us a place to put values for later use

Defined by a name preceded by a dollar sign ("$") character

Assigned a value via the equal sign ("=") character

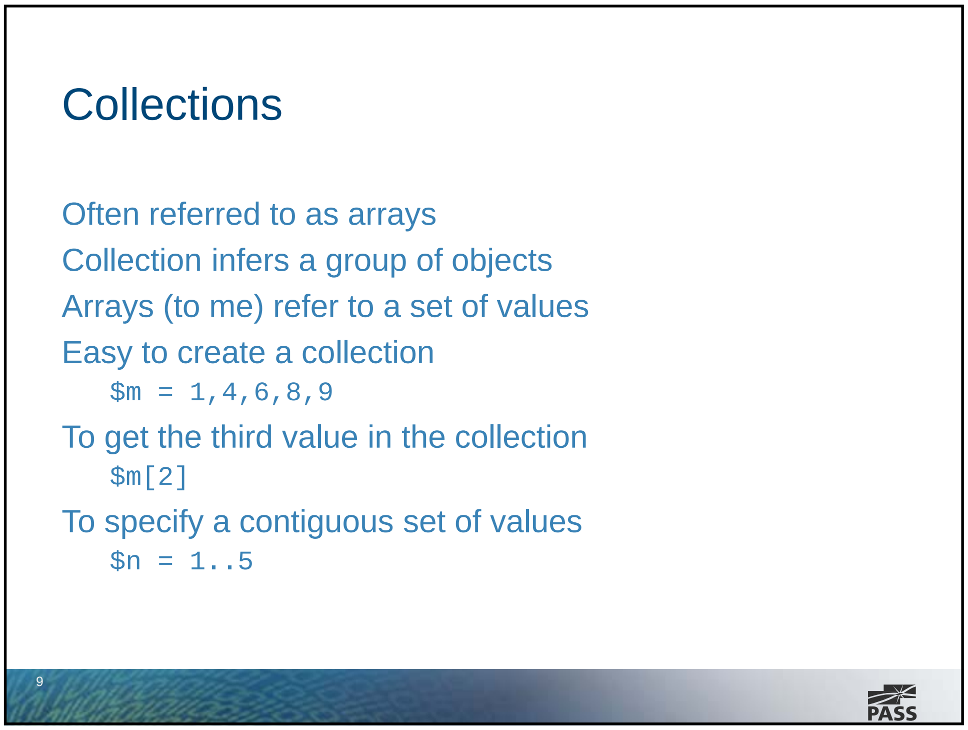
```
$i = 7
```

Creates an object of type integer

- Technically of type System.Int32

Demo

8



Collections

Often referred to as arrays

Collection infers a group of objects

Arrays (to me) refer to a set of values

Easy to create a collection

```
$m = 1,4,6,8,9
```

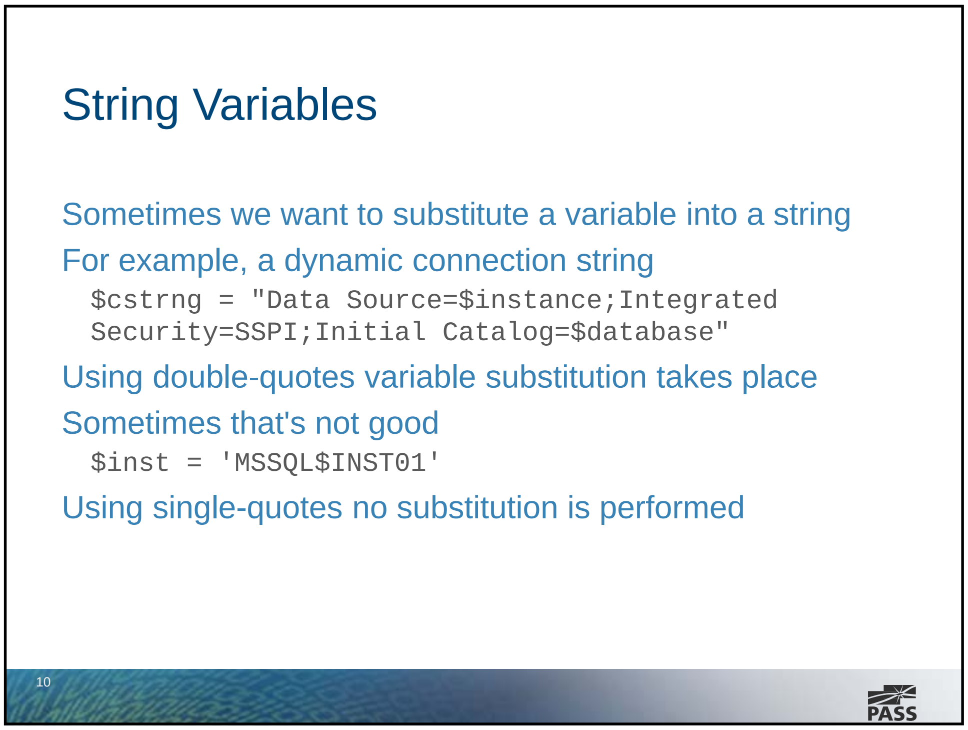
To get the third value in the collection

```
$m[2]
```

To specify a contiguous set of values

```
$n = 1..5
```

9



String Variables

Sometimes we want to substitute a variable into a string

For example, a dynamic connection string

```
$cstrng = "Data Source=$instance;Integrated
Security=SSPI;Initial Catalog=$database"
```

Using double-quotes variable substitution takes place

Sometimes that's not good

```
$inst = 'MSSQL$INST01'
```

Using single-quotes no substitution is performed

10

String Variables Part II

When you want to build a long string

```
$q = "SELECT TOP 25 [ContactID]"
$q = $q + "      ,[FirstName]"
$q = $q + "      ,[LastName]"
$q = $q + "      ,[EmailAddress]"
$q = $q + "      ,[Phone]"
$q = $q + "  FROM [AdventureWorks].[Person].[Contact]“
```

Or you can use a "here-string“

```
$q = @"
SELECT TOP 25 [ContactID]
      ,[FirstName]
      ,[LastName]
      ,[EmailAddress]
      ,[Phone]
  FROM [AdventureWorks].[Person].[Contact]
"@
```

Demo

SQLPS.exe – The SQL Server Mini-Shell

SQL Server 2008 & 2008 R2 include the “mini-shell”

- Included the full PowerShell version 1.0 executable
- Included the SQL Server snap-ins and Provider
- Installed five new cmdlets and the SQLServer PSDrive
  - Allows you to navigate SQL Server like a File System

Start the mini-shell from SQL Server Management Studio

- Right-click on any object and select Start PowerShell
- Mini-shell Execution Policy is RemoteSigned by default

SQL Server 2012 delivers SQLPS as a Module

- PowerShell 2.0+ required to install SQL Server 2012

SQL Server Cmdlets

Invoke-Sqlcmd takes a query in the form of a string and executes it on the server specified

Invoke-PolicyEvaluation allows you to test one of the policies created with SQL Server 2008 Policy-based Management

Encode-SqlName and **Decode-SqlName** allow you to convert names that are acceptable to SQL Server but not PowerShell

- Encode-SqlName will convert an instance name like SQLTBWS\INST01 to SQLTBWS%5CINST01
- Decode-SqlName will convert it back

Convert-UrnToPath converts the Universal Resource Name to a path name

- Used internally by SQL Server to reference an object
- Path Name is useful with the SQL Server Provider

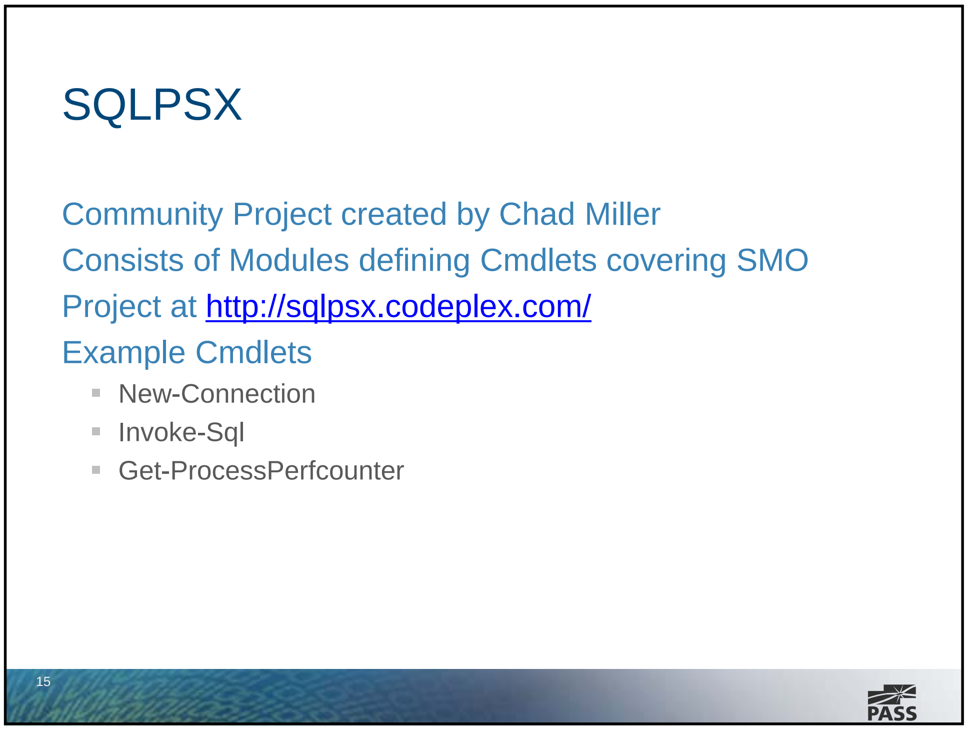
The SQL Server Provider

PSDrive allows you to browse SQL Server like a file system

Major folders under the SQLSERVER: drive

- SQL
  - Access Database Engine, Agent, Database Mail and Service Broker
- SQLPolicy
  - Access Policy-Based Management Objects
- SQLRegistration
  - Access Registered Servers and Central Management Server
- DataCollection
  - Access Data Collection feature from Management Data Warehouse
- Utility (SQL 2008 R2)
  - Managed Objects, like instances of the Database Engine
- DAC (SQL 2008 R2)
  - Data Application Objects

Demo



SQLPSX

Community Project created by Chad Miller

Consists of Modules defining Cmdlets covering SMO

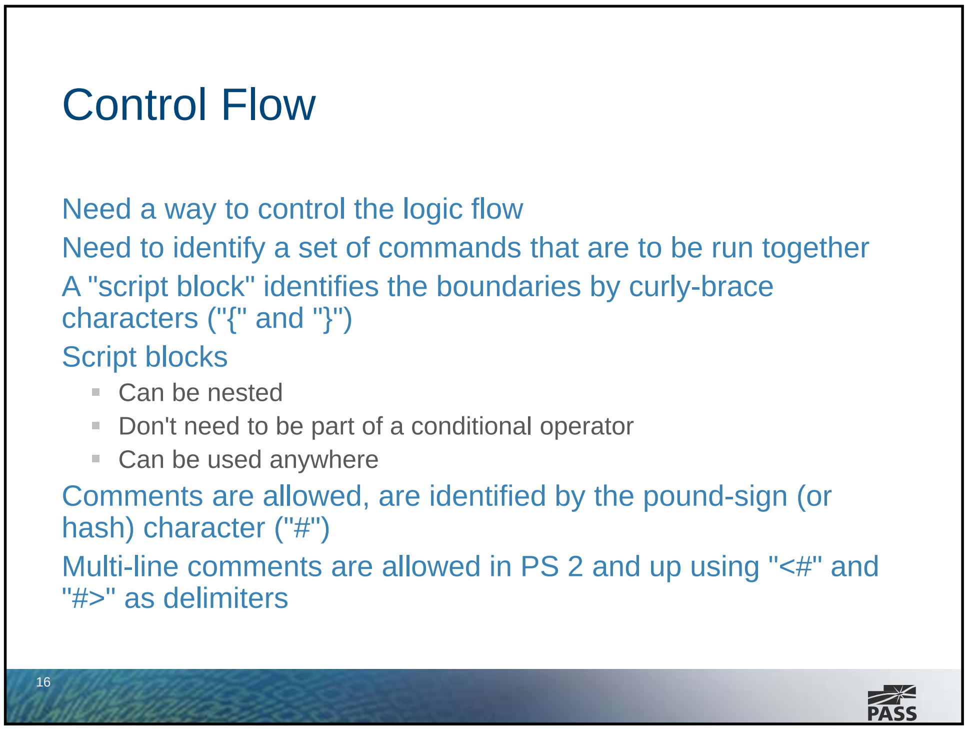
Project at http://sqlpsx.codeplex.com/

Example Cmdlets

- New-Connection
- Invoke-Sql
- Get-ProcessPerfcounter

15

PASS



Control Flow

Need a way to control the logic flow

Need to identify a set of commands that are to be run together

A "script block" identifies the boundaries by curly-brace characters ("{" and "}")

Script blocks

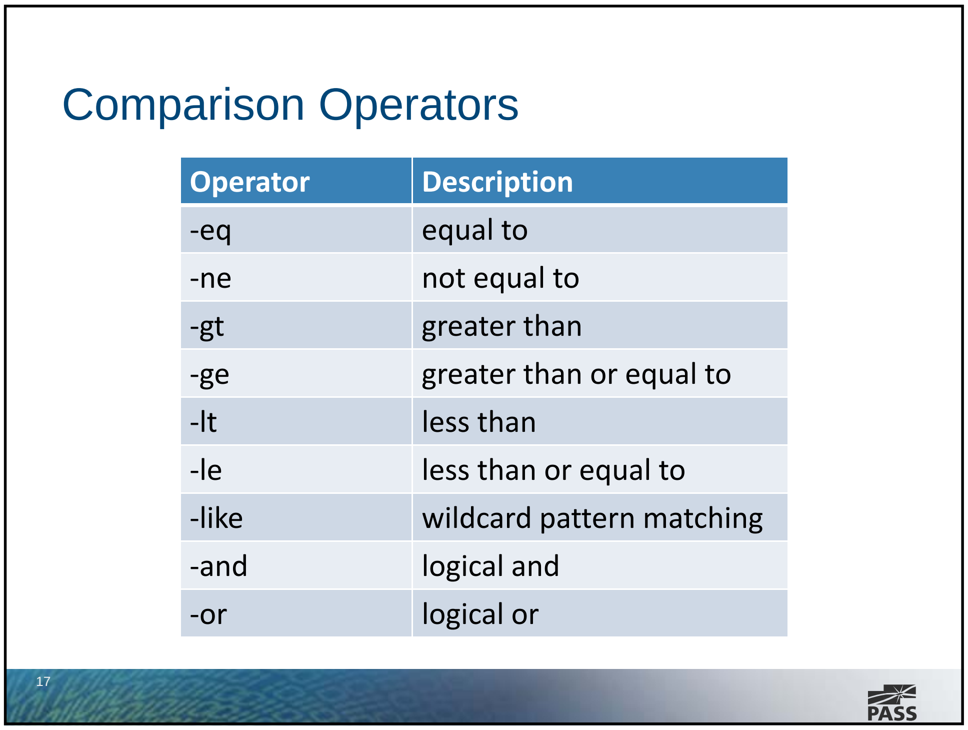
- Can be nested
- Don't need to be part of a conditional operator
- Can be used anywhere

Comments are allowed, are identified by the pound-sign (or hash) character ("#")

Multi-line comments are allowed in PS 2 and up using "<#" and "#>" as delimiters

16

PASS

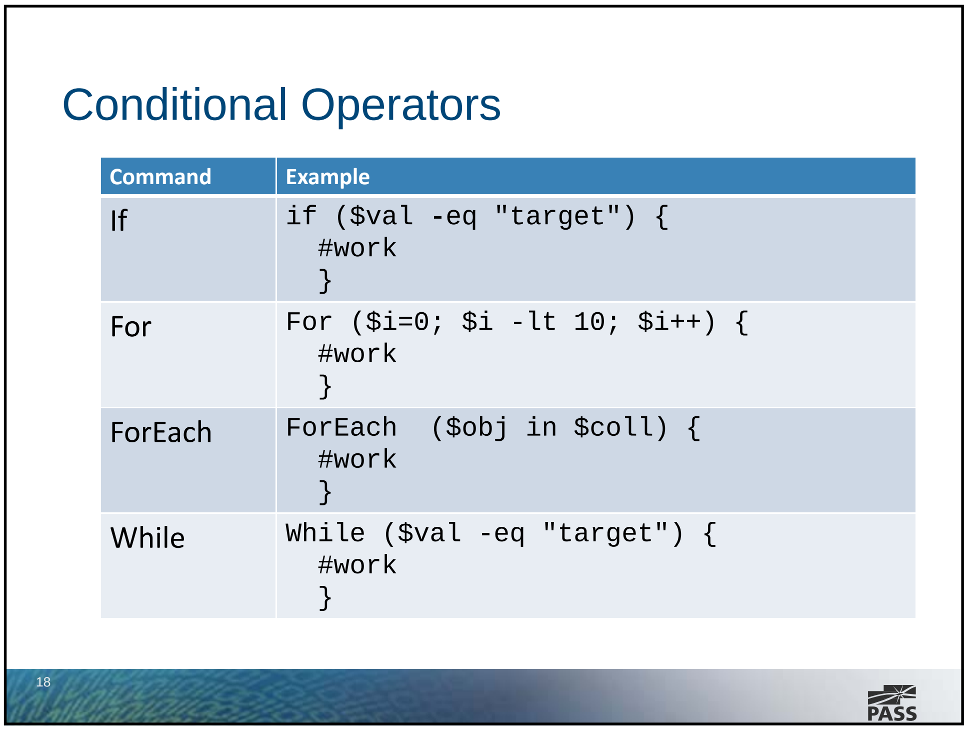


Comparison Operators

Operator	Description
-eq	equal to
-ne	not equal to
-gt	greater than
-ge	greater than or equal to
-lt	less than
-le	less than or equal to
-like	wildcard pattern matching
-and	logical and
-or	logical or

17

PASS



Conditional Operators

Command	Example

If	`if ($val -eq "target") {`
	`  #work`
	`  }`

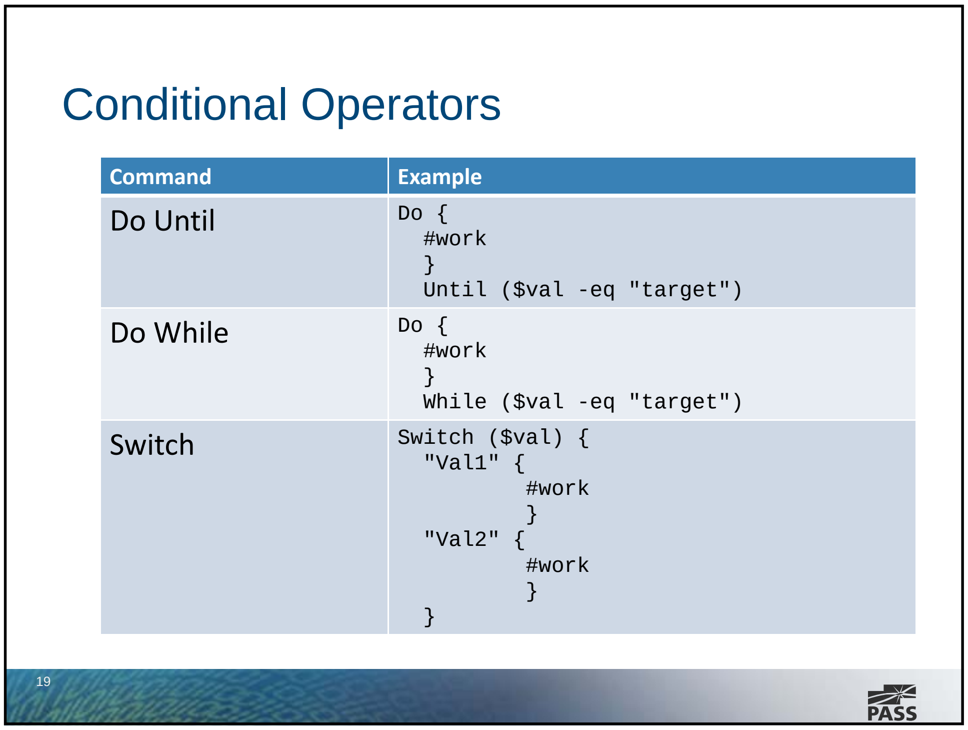
For	`For ($i=0; $i -lt 10; $i++) {`
	`  #work`
	`  }`

ForEach	`ForEach  ($obj in $coll) {`
	`  #work`
	`  }`

While	`While ($val -eq "target") {`
	`  #work`
	`  }`

18

PASS



Conditional Operators

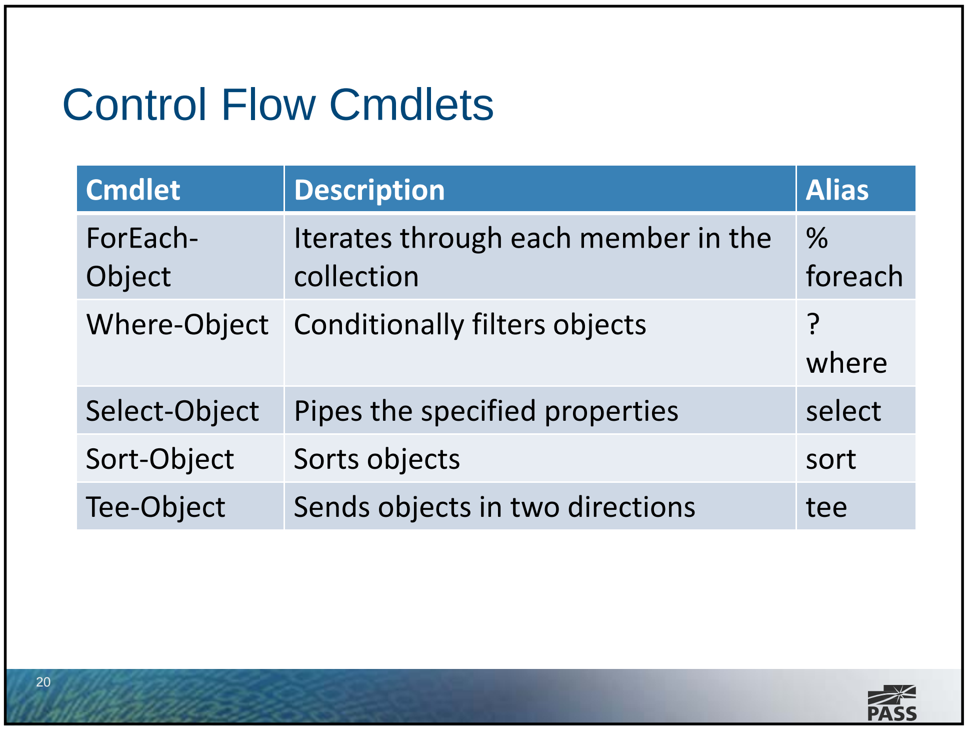
Command	Example

Do Until	`Do {`
	`  #work`
	`  }`
	`  Until ($val -eq "target")`

Do While	`Do {`
	`  #work`
	`  }`
	`  While ($val -eq "target")`

Switch	`Switch ($val) {`
	`  "Val1" {`
	`          #work`
	`          }`
	`  "Val2" {`
	`          #work`
	`          }`
	`  }`

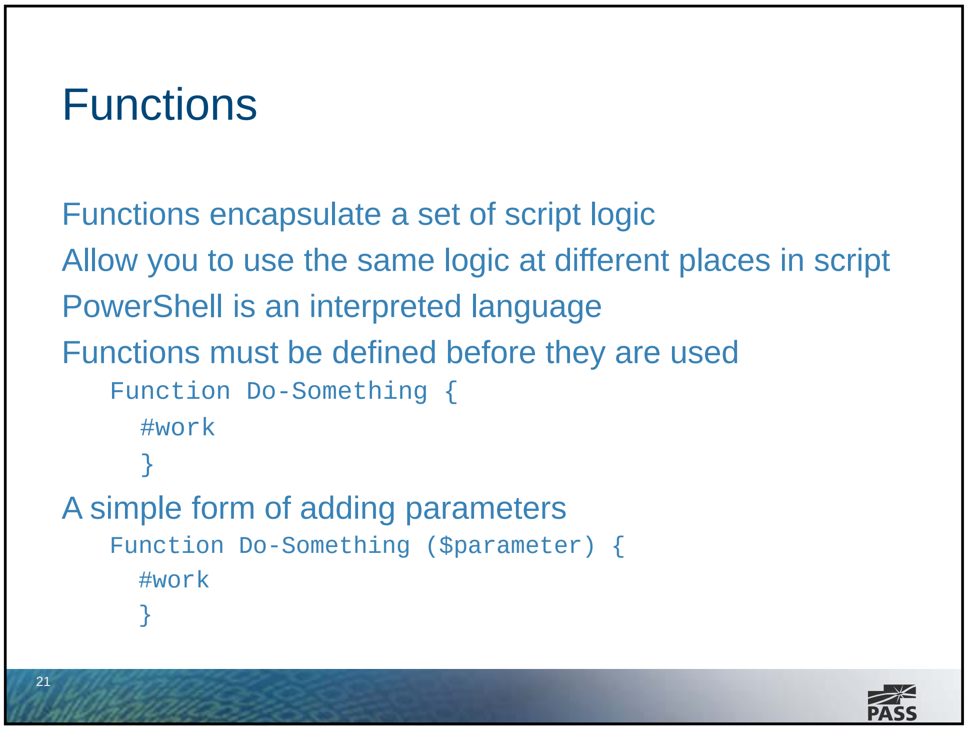
19



Control Flow Cmdlets

Cmdlet	Description	Alias
ForEach-Object	Iterates through each member in the collection	% foreach
Where-Object	Conditionally filters objects	? where
Select-Object	Pipes the specified properties	select
Sort-Object	Sorts objects	sort
Tee-Object	Sends objects in two directions	tee

20



Functions

Functions encapsulate a set of script logic

Allow you to use the same logic at different places in script

PowerShell is an interpreted language

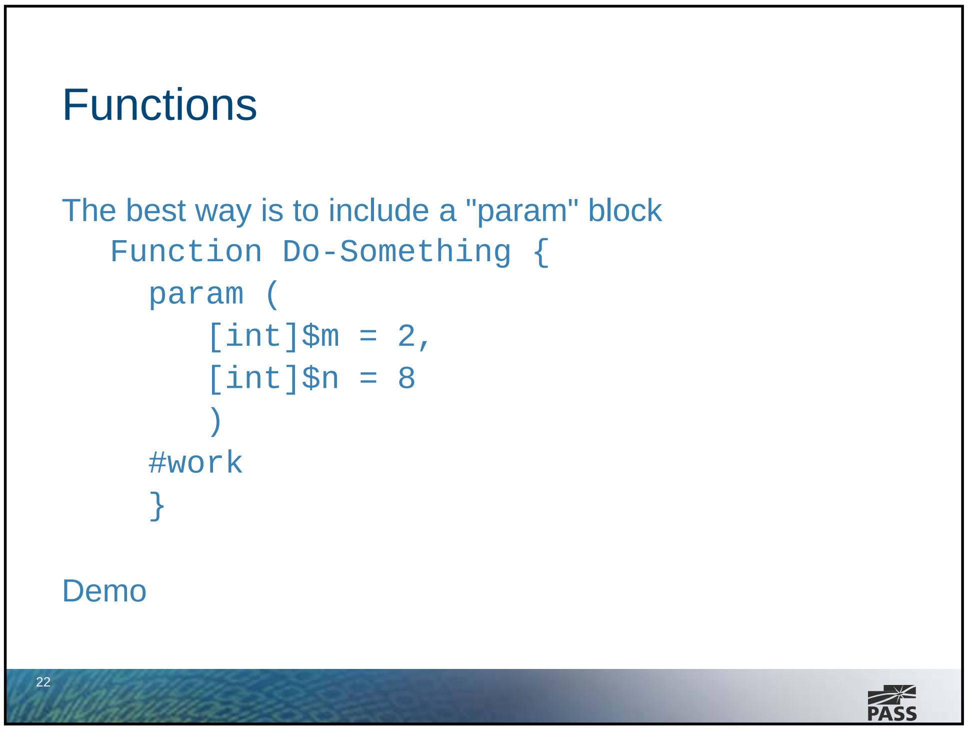
Functions must be defined before they are used

```
Function Do-Something {
  #work
  }
```

A simple form of adding parameters

```
Function Do-Something ($parameter) {
  #work
  }
```

21



Functions

The best way is to include a "param" block

```
Function Do-Something {
  param (
     [int]$m = 2,
     [int]$n = 8
     )
  #work
  }
```

Demo

22

Modules

Introduced in PowerShell 2.0
Designed to replace "snap-ins"
Allow you to organize code into reusable units
First place related functions into a script file
Rename the script file extension as .psm1
Use the Import-Module cmdlet to load the script
Use the Remove-Module cmdlet to unload the functions
Get-Module allows you to see what has been loaded
Export-ModuleMember exposes the module functions

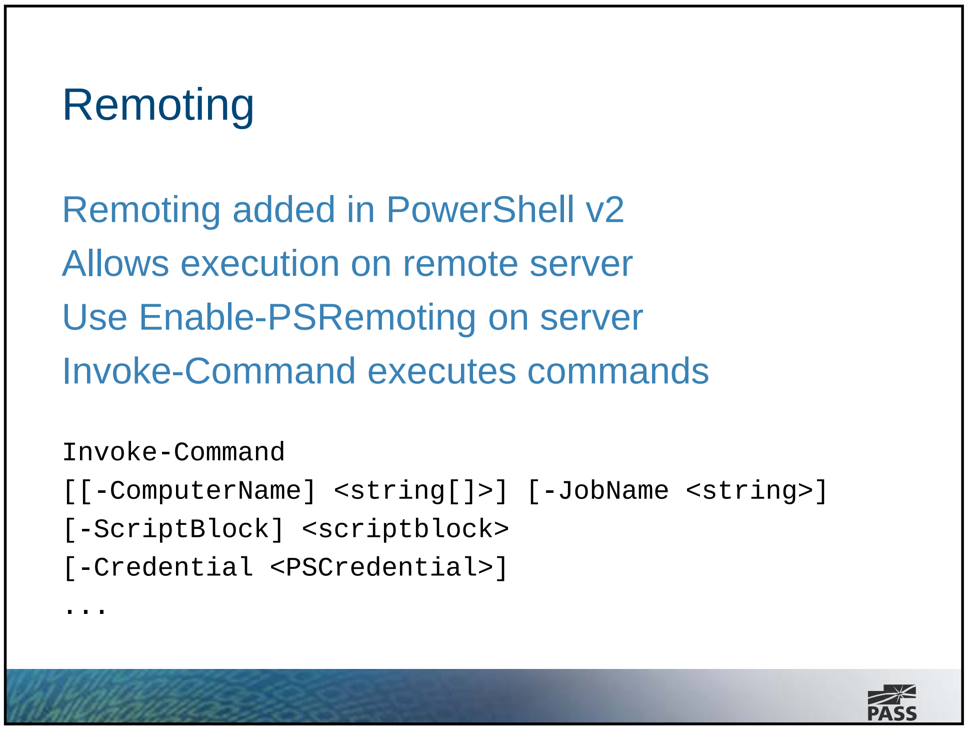
PASS

Create a Module

First place related functions into a script file
Rename the script file extension as .psm1
Move the file into the $ENV:PSModulePath directory
Use the Import-Module cmdlet to load the script
Use the Remove-Module cmdlet to unload the functions
Export-ModuleMember exposes the module functions

- By default all objects exposed
- Using Export-ModuleMember exposes only those objects listed as arguments

PASS



Remoting

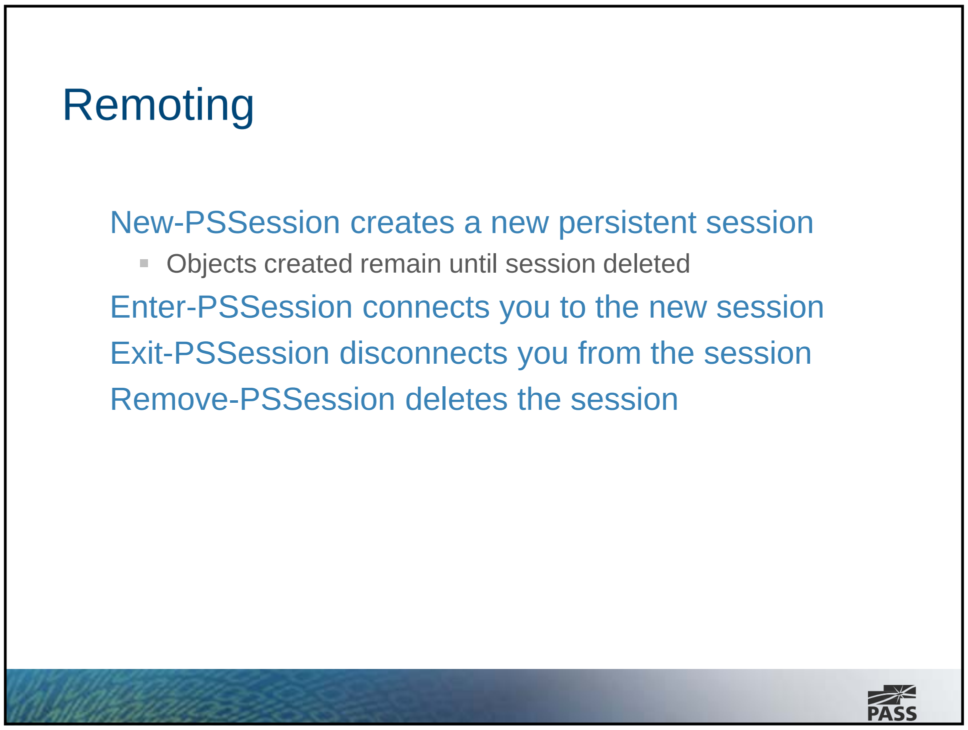
Remoting added in PowerShell v2

Allows execution on remote server

Use Enable-PSRemoting on server

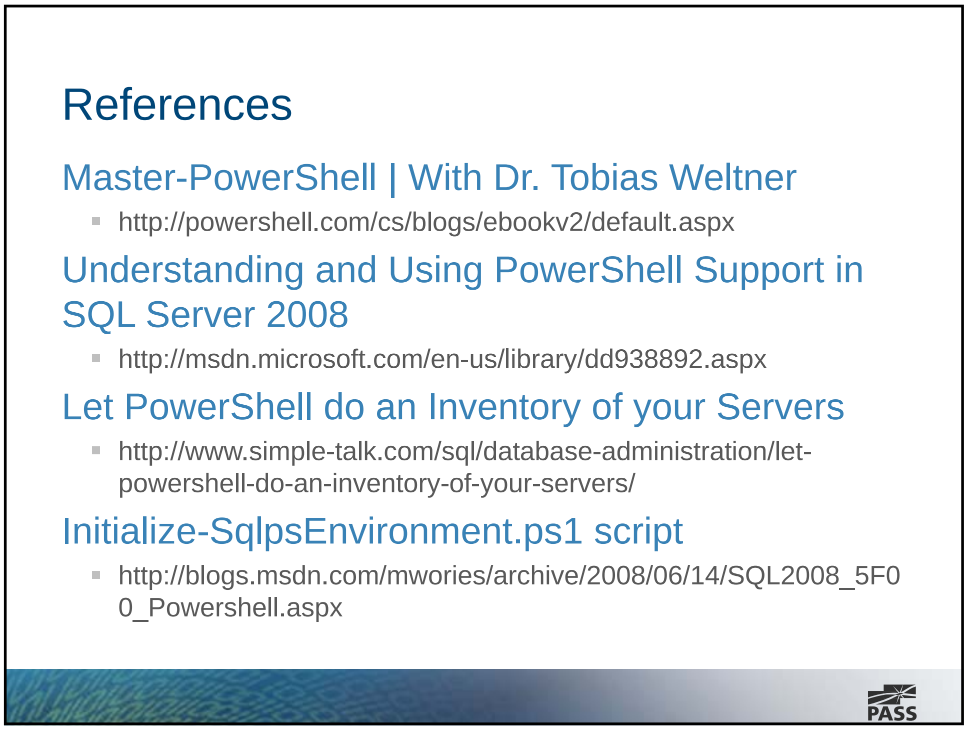
Invoke-Command executes commands

```
Invoke-Command
[[-ComputerName] <string[]>] [-JobName <string>]
[-ScriptBlock] <scriptblock>
[-Credential <PSCredential>]
...
```



Remoting

- New-PSSession creates a new persistent session
  - Objects created remain until session deleted
- Enter-PSSession connects you to the new session
- Exit-PSSession disconnects you from the session
- Remove-PSSession deletes the session



References

Master-PowerShell | With Dr. Tobias Weltner

- http://powershell.com/cs/blogs/ebookv2/default.aspx

Understanding and Using PowerShell Support in SQL Server 2008

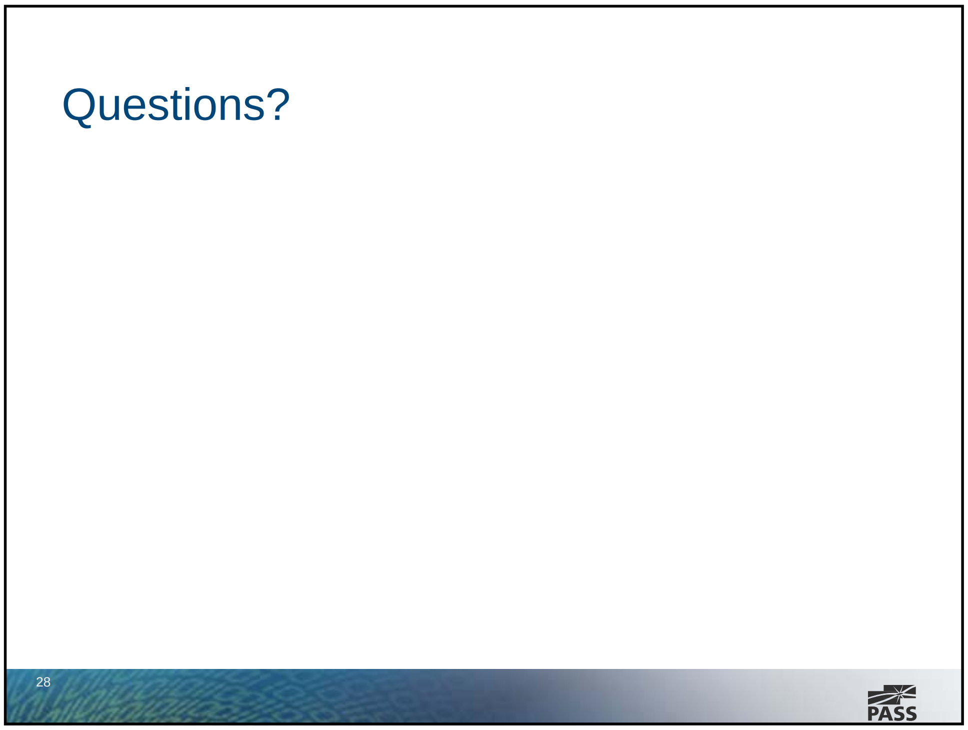
- http://msdn.microsoft.com/en-us/library/dd938892.aspx

Let PowerShell do an Inventory of your Servers

- http://www.simple-talk.com/sql/database-administration/let-powershell-do-an-inventory-of-your-servers/

Initialize-SqlpsEnvironment.ps1 script

- http://blogs.msdn.com/mwories/archive/2008/06/14/SQL2008_5F00_Powershell.aspx



Questions?

28