Gather SQL Server Performance Data with PowerShell

UpSearch

Allen White
UpSearch Technology Services
SQL Server MVP

UpSearch

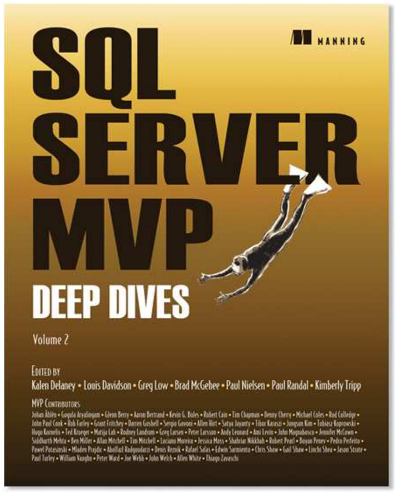
About Me

- SQL Server Consultant with UpSearch Technology Services
- Over 35 years in IT
- Career covered multiple disciplines – operations, development, telecommunications, network design/administration and database design and administration
- Started using Sybase in 1992, MS SQL Server in 1995
- Microsoft Certified IT Professional: Database Administrator and Database Developer, Microsoft Certified Trainer (MCT)
- Awarded Microsoft MVP Award for SQL Server for last 5 years

MVP Microsoft Most Valuable Professional

UPSEARCH

SQL Server MVP Deep Dives, Volume 2


SQL SERVER MVP
DEEP DIVES
Volume 2
MANNING
Edited by
Kalen Delaney • Louis Davidson • Greg Low • Brad McGehee • Paul Nielsen • Paul Randal • Kimberly Tripp

- **www.operationsmile.org**
- **www.manning.com/delaney**

3

UPSEARCH

Agenda

- Brief Introduction to PowerShell
- Performance Counters
- Capture Options
- PowerShell Script
- Performance Analysis Report

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+)
- 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

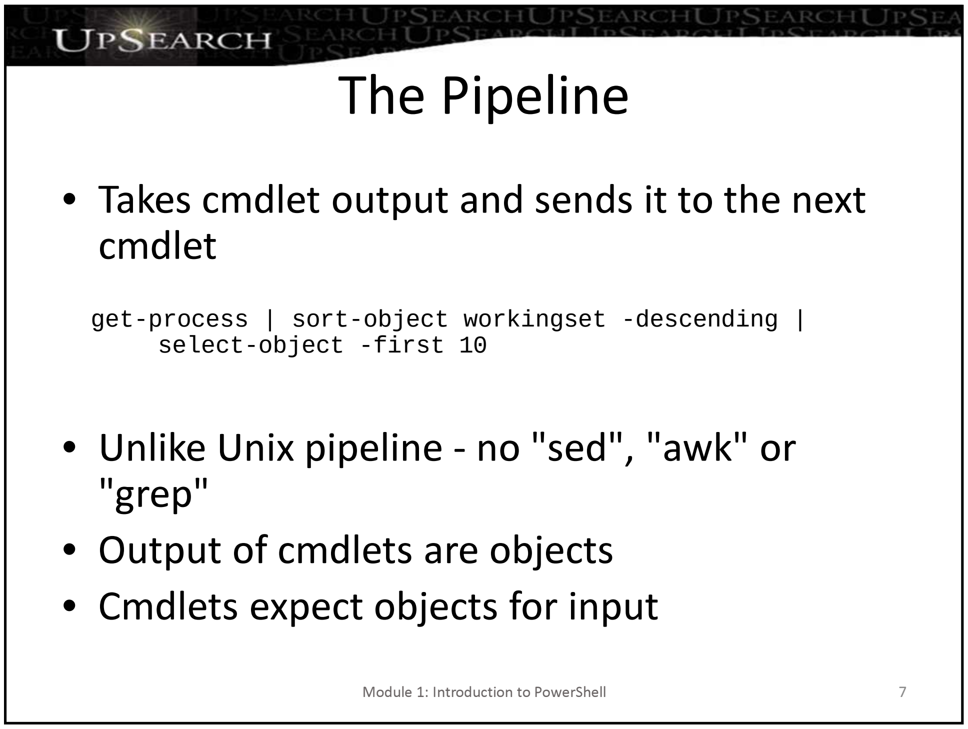
Cmdlets

- 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
```

- Three Most Important cmdlets

```
Get-Help
Get-Command
Get-Member
```

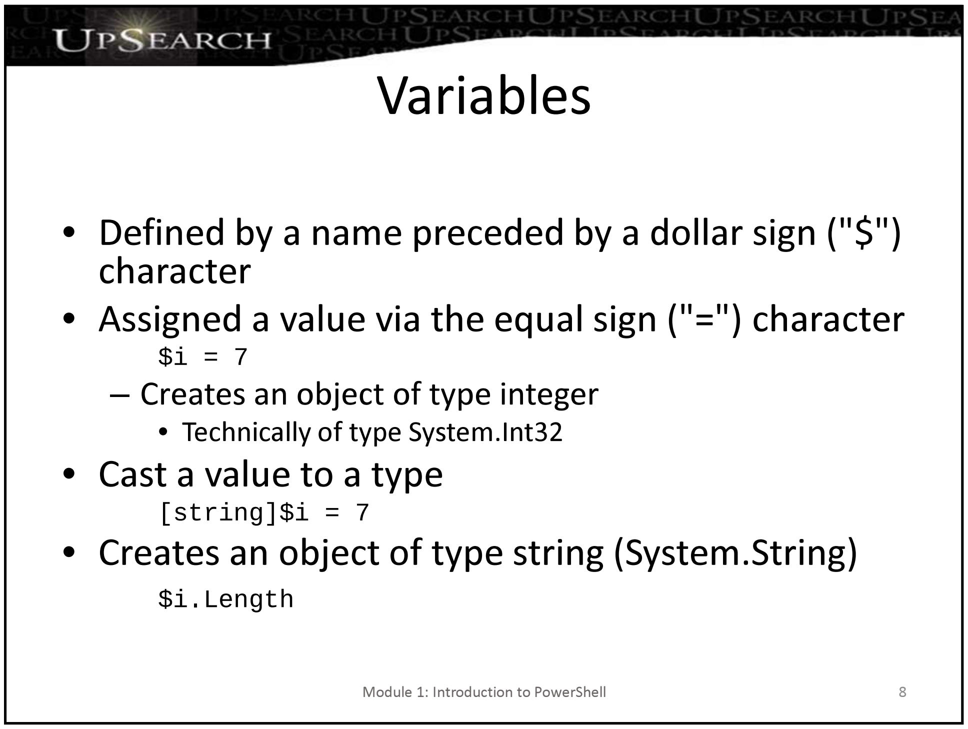


The Pipeline

- 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



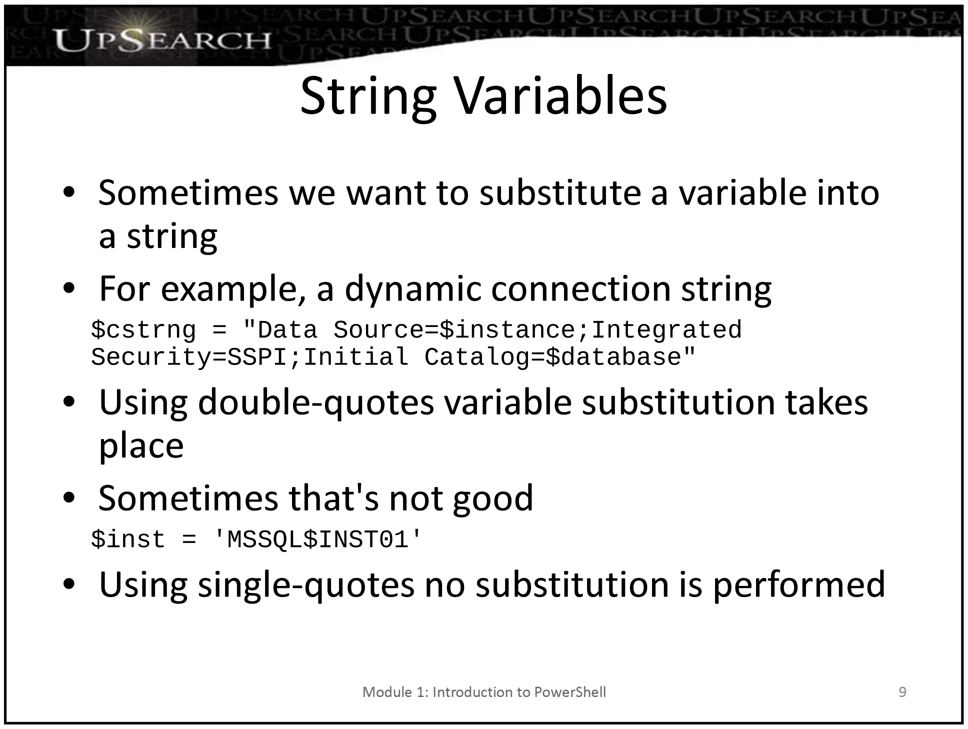
Variables

- 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
- Cast a value to a type

  ```
  [string]$i = 7
  ```
- Creates an object of type string (System.String)

  ```
  $i.Length
  ```



String Variables

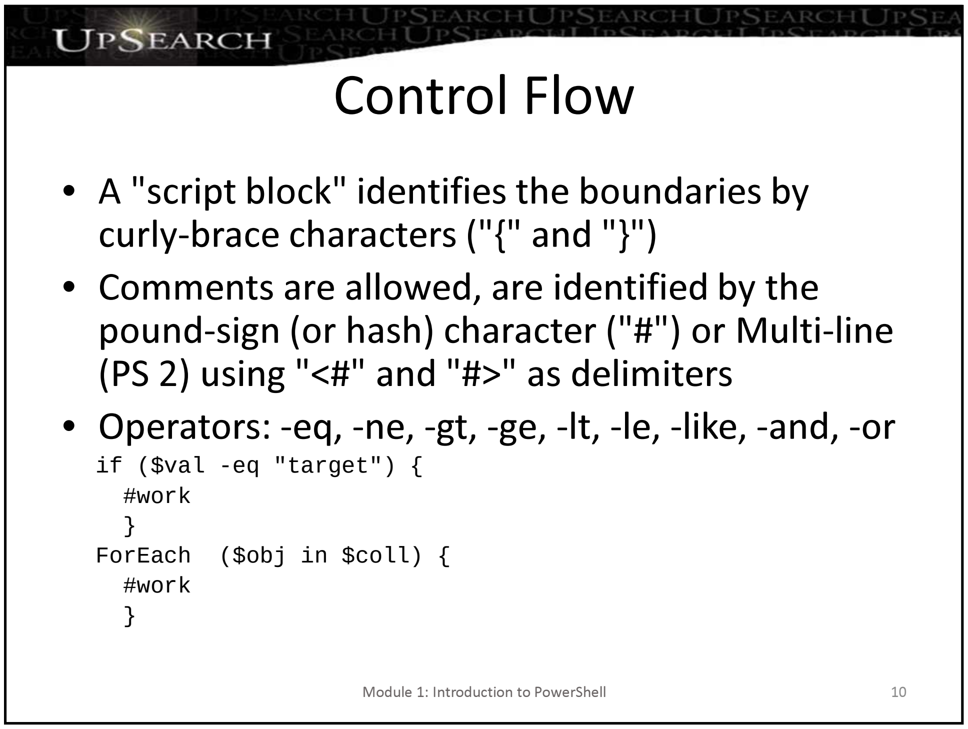
- Sometimes we want to substitute a variable into a string
- For example, a dynamic connection string

```
$cstrng = "Data Source=$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



Control Flow

- A "script block" identifies the boundaries by curly-brace characters ("{" and "}")
- Comments are allowed, are identified by the pound-sign (or hash) character ("#") or Multi-line (PS 2) using "<#" and "#>" as delimiters
- Operators: -eq, -ne, -gt, -ge, -lt, -le, -like, -and, -or

```
if ($val -eq "target") {
  #work
  }
ForEach  ($obj in $coll) {
  #work
  }
```

Demo

- PowerShell Introduction

Goal - Capture Performance Baseline

- Baseline shows normal performance
- Deviations from Baseline require investigation
- Problem
  - Data comes from disparate sources
  - Coordination of multiple gathering tools
  - Synchronizing data for true baseline analysis
- Solution
  - PowerShell

Key Performance Indicators

- Which counters show us system health
- There's no "right" answer
- These are my choices

Operating System Counters

Object	Counter	Look For
Processor	% Processor Time	<= 80%
Memory	Available MBytes	Low Memory, Server Paging
Paging File(_Total)	% Usage	Should be < 70%
PhysicalDisk(*)	Avg. Disk Sec/Read	Latency. Avg time to read data (<.02)
PhysicalDisk(*)	Avg. Disk Sec/Write	Latency. Avg time to write data (<.02)
System	Processor Queue Length	> 10 threads/proc and CPU > 80%

SQL Server Counters

Object	Counter	Look For
Access Methods	Forwarded Records/sec	< 10 per 100 batch requests/sec
Access Methods	Page Splits/sec	<20 per 100 batch requests/sec
Buffer Manager	Buffer cache hit ratio	below 90% is bad
Buffer Manager	Page life expectancy	>= (DataCacheSize/4*300)
General Statistics	Processes blocked	Baseline, check for changes
SQL Statistics	Batch Requests/sec	> 1000 is busy system
SQL Statistics	SQL Compilations/sec	<10% of batch requests/sec
SQL Statistics	SQL Re-Compilations/sec	<10% of compilations/sec

Sources for Performance Data

- Perfmon
  - Save data to .CSV
  - Use SSIS or PowerShell to import results
- DMVs
  - Great source of SQL Server data
    - sys.dm_os_performance_counters
  - Only returns SQL Server current instance counters
- WMI
  - Allows access to all aspects of server
  - Crunching the numbers can be tricky

Performance Data in PowerShell

- Get-Counter cmdlet (PowerShell 2.0)
  - Invocation sets own interval handler
- System.Diagnostics.PerformanceCounter
  - Support directly within .NET
  - Results directly match Perfmon values
  - Accessible from PowerShell
- Demo

Performance Database

- Every Admin should have one
- Store Baseline Data
- Store Server Side Trace info
- Store Server and Instance info
- Keep all management info in one place

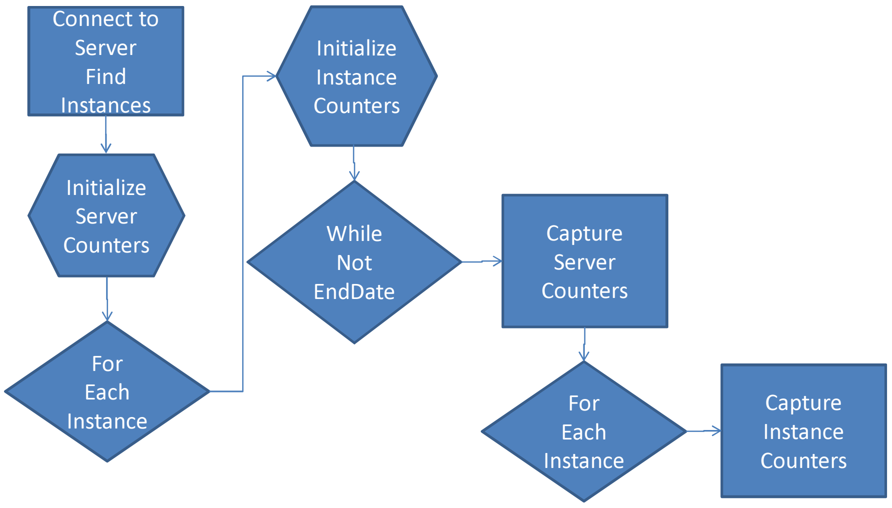
Scripting the Data Capture

- Capture the counter data

```
# Initialize Perfcounters
$ppt = new-object System.Diagnostics.PerformanceCounter
$ppt.CategoryName = 'Processor'
$ppt.CounterName = '% Processor Time'
$ppt.InstanceName = '_Total'
$pptv = $ppt.nextvalue()
```

- Insert into Performance Database
- Wait defined interval and do it again

Logic Flow for getperf.ps1


Connect to Server Find Instances
Initialize Server Counters
For Each Instance
Initialize Instance Counters
While Not EndDate
Capture Server Counters
For Each Instance
Capture Instance Counters

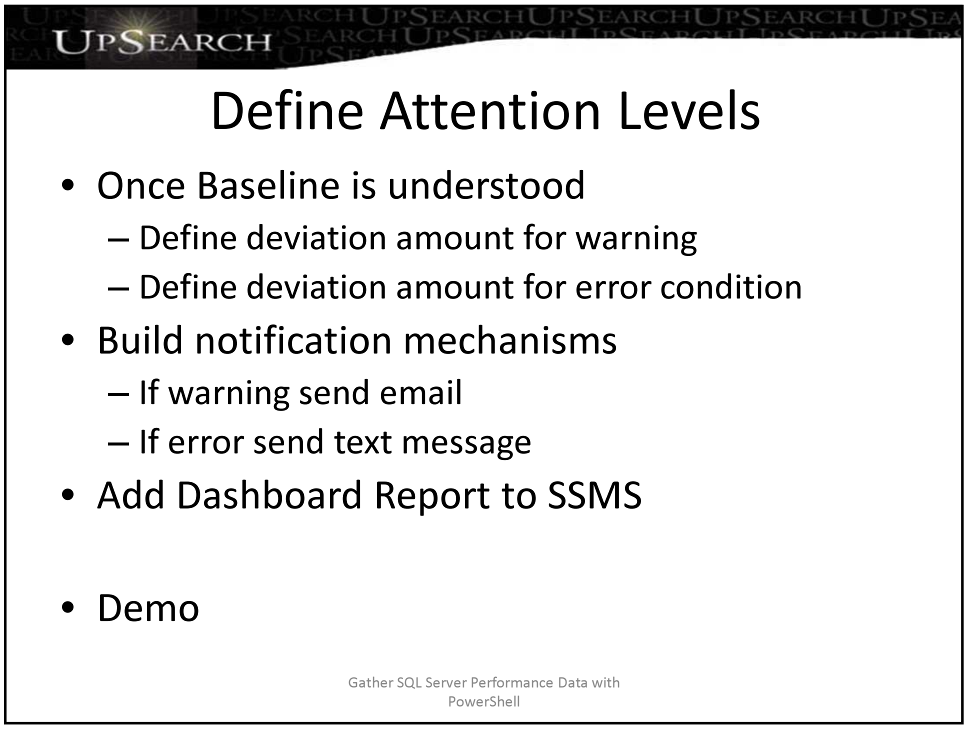
Demo

- getperf.ps1

Creating the Analysis Reports

- Create Basic Report
- Add Table for Counter Data
- Add Graphs to see Trends

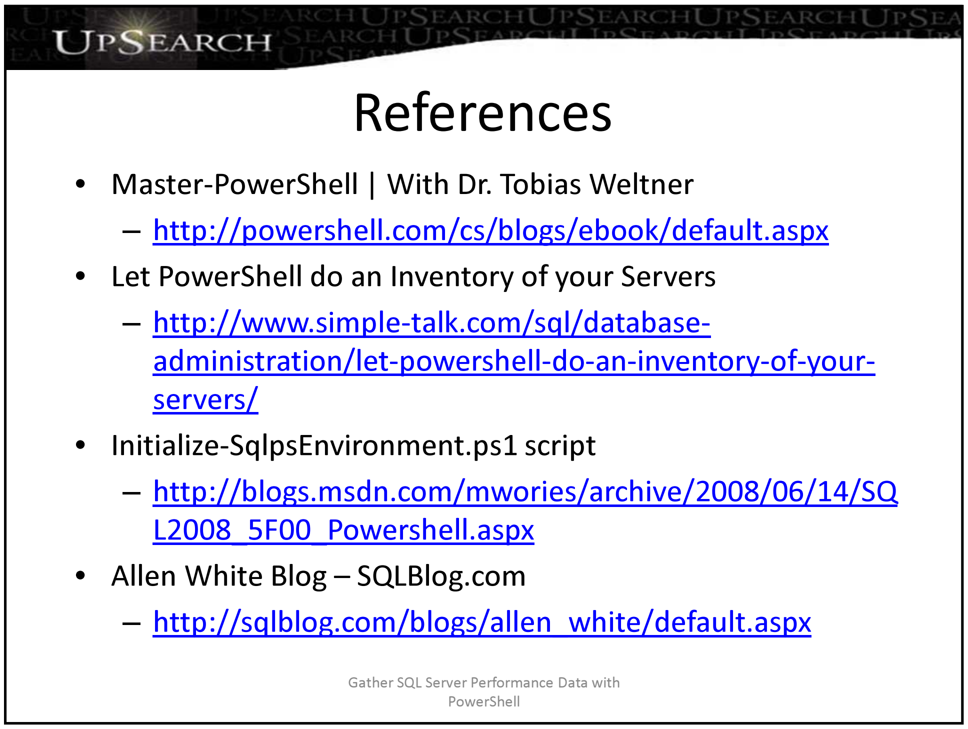
- Demo



Define Attention Levels

- Once Baseline is understood
  - Define deviation amount for warning
  - Define deviation amount for error condition
- Build notification mechanisms
  - If warning send email
  - If error send text message
- Add Dashboard Report to SSMS

- Demo



References

- Master-PowerShell | With Dr. Tobias Weltner
  - http://powershell.com/cs/blogs/ebook/default.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
- Allen White Blog – SQLBlog.com
  - http://sqlblog.com/blogs/allen_white/default.aspx

UpSearch

Thank you