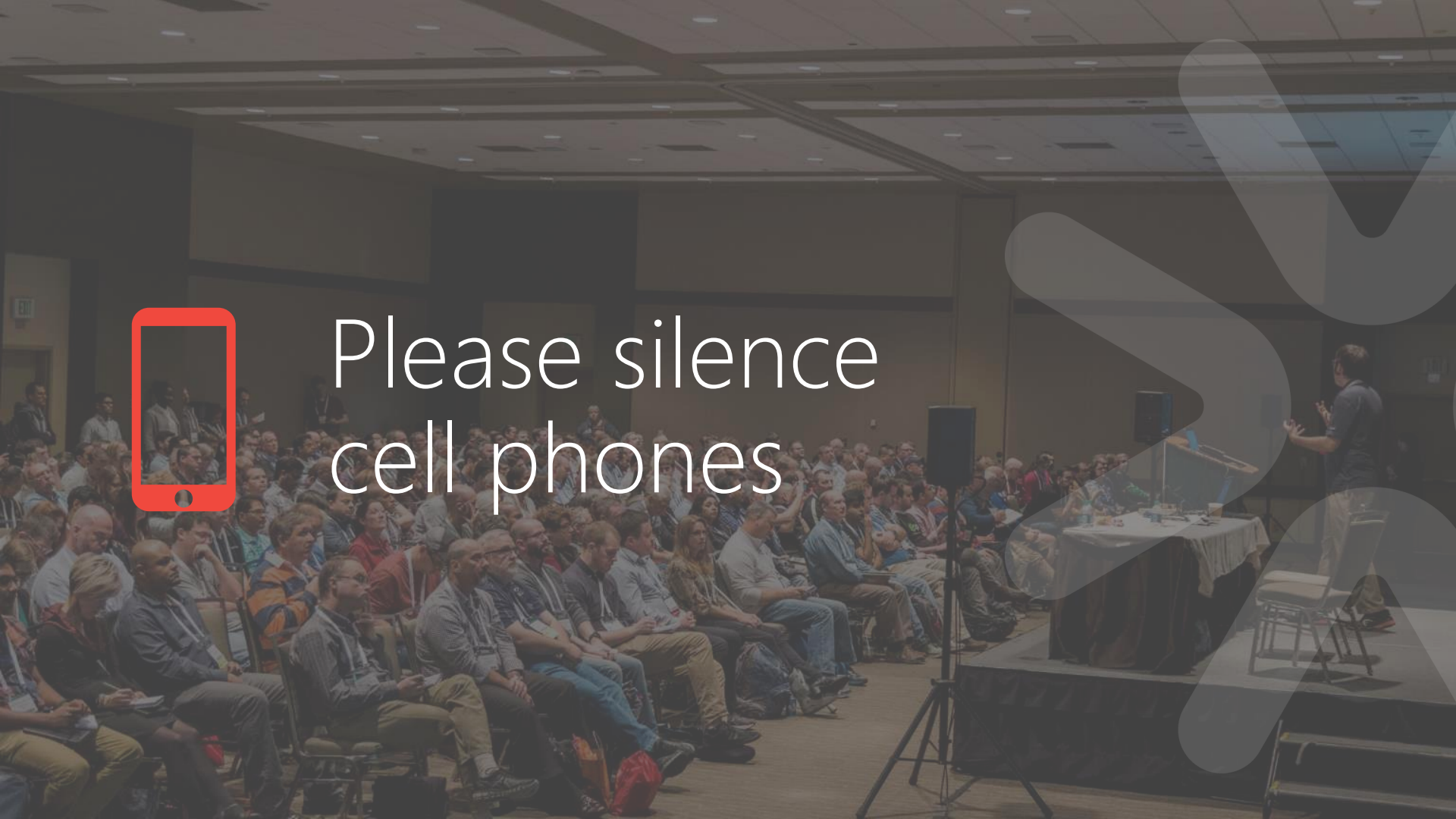
PASS SUMMIT2018

# PowerShell Tips & Tricks for the DBA

## SQL Server with PowerShell

Ben Miller, Sr. Database Architect, MaritzCX


Please silence
cell phones

# Explore everything PASS has to offer

**Free Online Resources**

**Newsletters**

**PASS.org**

24HOURS OF PASS

Free online webinar events

PASS LOCAL GROUPS

Local user groups around the world

PASS SQLSATURDAY

Free 1-day local training events

PASS VIRTUAL GROUPS

Online special interest user groups

PASS MARATHON

Business analytics training

PASS VOLUNTEERS

Get involved

# Session evaluations

Your feedback is important and valuable.

**Submit by 5pm Friday, November 16th to win prizes.**

3 Ways to Access:

**Go to passSummit.com**

**Download the GuideBook App**
and search: PASS Summit 2018

**Follow the QR code link** displayed on session signage throughout the conference venue and in the program guide

PASS

# Ben Miller (DBAduck)

## Sr. Database Architect, MaritzCX

/dbaduck

@DBAduck

DBAduck

DBADuck
ben miller

### SQL Server Certified Master

The first MCM in Utah. Has been working with SQL Server since 4.2, even on O/S2.

### Microsoft Data Platform MVP

Awardee since 2009 for SQL Server. Contributes to Experts-Exchange and SQLServerCentral forums. Speaker at many SQL Saturdays around the country. I have led local PASS Chapters as well as founded the Virtual PowerShell Group.

### PowerShell DBA to the Core!

I have always loved automation and have used PowerShell since v1.0 I have spent many years using SMO and love to automate anything that I can to get the computer to do my work. I even wrote a chapter for the PowerShell Deep Dives book. On the Quest to become a PowerShell DBA.

# What you will learn about?

## PowerShell Tips

Array vs. ArrayList
Get Assemblies in your session
Strings vs. Stringbuilder
1..X iteration
.ForEach() usage
Splatting (what is this?)
-eq vs. –ceq
Pipelining
Object Creation tips

## SQL Server Tips

SMO
Refresh
Test-Path in the SQL Provider
SQL Authentication and the Provider
SetDefaultInitFields
Invoke-SqlCmd
Write-SqlTableData
Using SQL Provider Context

# ArrayList vs. Array

- [System.Collections.ArrayList]
- .NET type
- .Add(object) to add to the array list
- Memory Buffer

- Array (native to PowerShell)
  - @( ) is a blank Array
  - @(1,2,3)
  - 1,2,3
- $ary += $obj to add to the array
- Immutable – cannot be changed

# Get Assemblies in your Session

- Assemblies get loaded by PowerShell
- Assemblies can be loaded by Modules
- Assemblies can be loaded by You

- PowerShell is on top of .NET (Core or Full)
- Assemblies have functionality you don’t have to write

- [AppDomain]::CurrentDomain.GetAssemblies()

# String vs. StringBuilder

- $obj = “String”
- $obj = “String” + “ “ + “Builder”
- $obj = “$obj Builder”
- Strings are Immutable – cannot be changed

- [System.Text.StringBuilder]

- $obj = new-object –typename System.Text.StringBuilder –Args 4096
- $obj.Append(“ “) or $obj.AppendLine(“ “)
- $obj.AppendFormat(“{0} {1}”, “one”, “two”)

- Returns an object

# 1..X shortcut

- 1..5 produces an array of 1 – 5 one at a time
- Can be used with Foreach()
- Increments by 1 so you can start at any number
- Can be done in reverse order
- 5..1 and it will produce 5,4,3,2,1

- Useful and kind of like GO 5 except that it does not always have to go in reverse, and you can use the number in PowerShell

- 1..5 | Foreach { $_ }

# .ForEach() on each object

- Dynamic Properties and Methods
- $obj.Foreach( { code block; } )
- Iteration for each item in the collection in $obj

- * Bonus: You can add your own as well.

# Splatting in PowerShell

- Splatting is all about a Variable satisfying Parameters
- Variables look like this $obj

- Splatting uses @obj

- Basically a Hash Table with multiple Keys/Values passed into a function/cmdlet

- Rules: You can splat with a variable that has as many or fewer parameter satisfying keys/values but NOT more.

# Case Sensitive Comparisons

- PowerShell is Case Insensitive
- Comparing values (mostly Strings) is the same
  - Ben = ben

- Comparison Operators
  - -eq -lt -gt etc.

- Case Sensitive Operators
  - -ceq -clt -cgt etc.

- Great for password comparison and others that require Case Sensitivity

# Pipelining

- Using the Pipeline character | (pipe character)

- Objects are passed By Value not By Reference on the pipeline
- That means Copies are made of the object
- Keep the objects getting smaller on the left and getting smaller going to the right

- Powerful Tip to pass a set of objects over a pipe to a Cmdlet that handles a set of objects

# Object Creation Tips

- You will typically need objects to store data in during automation

- 1. $obj = New-Object –TypeName PSObject
  - Add-Member
- 2. $obj = “” | Select Name, ID, Description
- 3. $obj = @{ Name = Value; ID=6; Description = “Desc” }

- The only one I would stay away from on custom objects is #1

# SMO (Shared Management Objects)

- SMO is a great tool and totally accessible from PowerShell

- Knowing when to use it is the key
- If you need to manipulate the object (Table, Database, etc) use SMO
- If you need information you can use SMO
- It you need information from a set of objects, testing is needed

- Some objects can be easily pulled and manipulated using
  - Server.SetDefaultInitFields

# Refresh() method on SMO objects

- Objects are powerful in PowerShell
- Once properties are pulled from SQL, they are in memory
- Once in Memory they are not asked for again

- Use Refresh() on any object to signal to SMO to retrieve it again

- Normally needed when you are adding objects or if something is going to change like Disk Space or space used on the object

- When using Alter() the property is already refreshed because you set it
- Must be done on each object
  - $collection.Foreach( { $_.Refresh() } )

# Test-Path in the SQL Provider

- SQLPS (deprecated) and SQL Server module load a Provider
- Providers represent a service in a Path-like Structure
- Providers manifest themselves as Drives (PSDrives)

- SQLSERVER: is the drive that gets loaded by the SqlServer module

- Path to a Database
- SQLSERVER:\sql\servername\instancename\Databases\master
  - If you are using a Default Instance the instance name is DEFAULT

- Now you can test for the existence of a database called Ben
- Test-Path SQLSERVER:\sql\localhost\default\Databases\Ben
  - If it exists, you will get back True, if not then False

# SQL Authentication in SQL Provider

- By Default, SQLSERVER: drive uses Windows Authentication
- Because it is drive representation, you can use a Credential

- Get-PSDrive to see which drives are present
- New-PSDrive to create a new drive

Invoke-SqlCmd

- SQL Provider recognizes the -Credential parameter for SQL Auth

- When you see SQLSERVER:\sql\path……+sa
  - The + indicates that you are using SQL authentication
- Need the Root at least to the Instance level to use SQL Authentication but you can go further, even with Windows Authentication

# Invoke-Sqlcmd

- Using TSQL in PowerShell can be useful
- $query = “select * from sys.databases”
- Use Invoke-SqlCmd to get the data
  - ServerInstance servername
  - Database databasename
  - Query $query
- Special clauses in the SqlServer module now
  - -OutputAs
  - Values (DataRows (default), DataTable, DataSet)
- Use QueryTimeout 0 if you intend it to take longer than 10 minutes

# SetDefaultInitFields

- Method is on the Server object
- Parameters are Object Type and String Collection of fields

- Types are like this
  - [Microsoft.SqlServer.Management.Smo.Database]

- String collections are like this
- $coll = new-object –typename System.Collections.StringCollection
- $coll.Add(“ID”)

- `$server.SetDefaultInitFields([Microsoft.SqlServer.Management.Smo.Database, $coll)`

# Write-SqlTableData

- This one is in the SqlServer module (NOT SQLPS)
- One like this is in the DBAtools module as well (Write-DbaDataTable)

- Basically you can write data to a table in SQL from an object
- -Force can be used to create the table
  - Strings become nvarchar(max) so not amazingly useful

- Super fast, uses SqlBulkCopy to get the data in
- Automaps by position of the column, not by column name

# Using SQL Provider Context

- Commands in the SqlServer module understand context
- Inside SQLSERVER:\sql\server\instance\Databases you can use a command that understands where you are.

- Invoke-SqlCmd will use the context of the server you are in

- Backup-SqlDatabase will also use context and will also use settings in the instance


Demo Time
Tips and Tricks

# Session evaluations

Your feedback is important and valuable.

**Submit by 5pm Friday, November 16th to win prizes.**

3 Ways to Access:

**Go to passSummit.com**

**Download the GuideBook App**
and search: PASS Summit 2018

**Follow the QR code link** displayed on session signage throughout the conference venue and in the program guide

PASS


# Thank You

## Learn more from Ben Miller

@DBAduck

ben@benmiller.net