

How we made Configurable Pester Tests For SQL Server



Rob Sewell @sqldbawithbeard
<https://sqldbawithAbeard.com>

Speaker Questionnaire

Name : Rob

Occupation : DBA, Automator, Do-er, Trainer

Interests : PowerShell, Automation And SQL (PaaS geddit?)

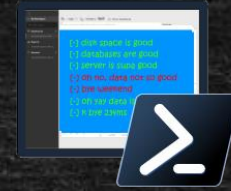
Interesting Fact : Beard. (Still) Plays Cricket, Flies Drone

Speaker : SQL Saturdays, SQL Relay, PowerShell Events

Community : SQL South West , SQL Sat Exeter , PowerShell VG, PowerShell Conference EU Organiser, dbareports, dbatools, dbachecks, MVP

Quick Poll

Contact Me



@sqldbawithbeard



robsewellsqldb



sqldbawithAbeard.com



mrrobsewell@outlook.com



RobSewell.info

Background



dbatools

- *Community module founded by Chrissy*
- *Over 100 contributors*
- *5 billion commands to work with SQL Server (no point putting an actual number in here it will change!!)*
- *Many commands to get information or check best practices*

dbatools

- *Rob needed to validate estates at work*
- *Wrote Pester tests using dbatools*
- *More than a year discussing wondering the best way to enable configuration*

Pester

- *Writing Pester Tests for one SQL instance is easy once you “get” Pester*

```
Describe "SQL Instance Number 1 Config" {  
    It "Should have Max Memory set to 112Gb" {  
        (Get-DbMaxMemory -SqlInstance SQLPROD2).SqlMaxMb | Should -Be 114688  
    }  
}
```


Pester

Writing slightly different Pester Tests for slightly different instances is copy and paste

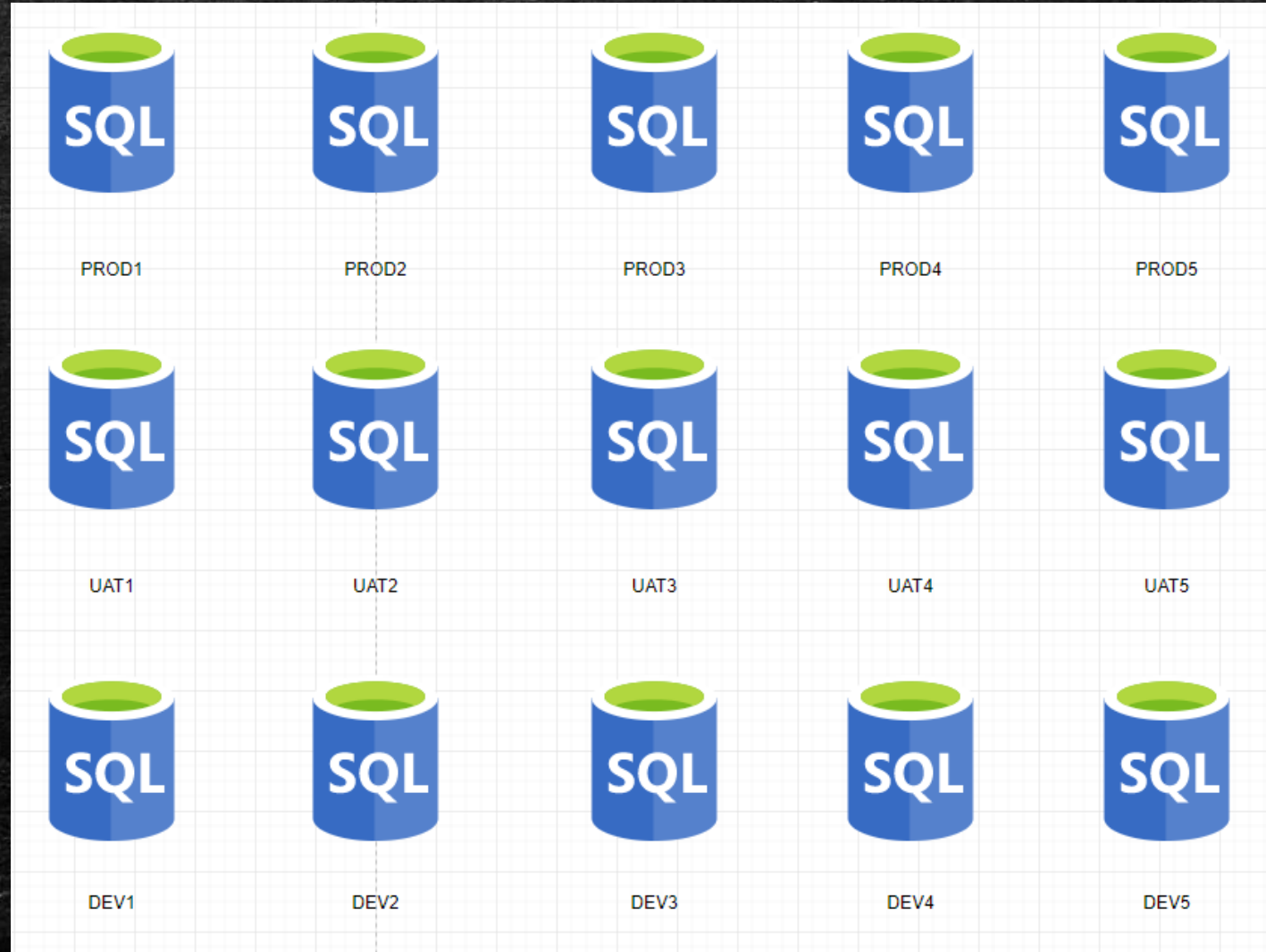
```
Describe "SQL Instance Number 2 Config" {  
    It "Should have Max Memory set to 56Gb" {  
        ([Get-DbMaxMemory -SqlInstance SQLPROD2]).SqlMaxMb | Should -Be 57344  
    }  
}
```

It is possible to parameterize Pester tests (but not so easy to say!)

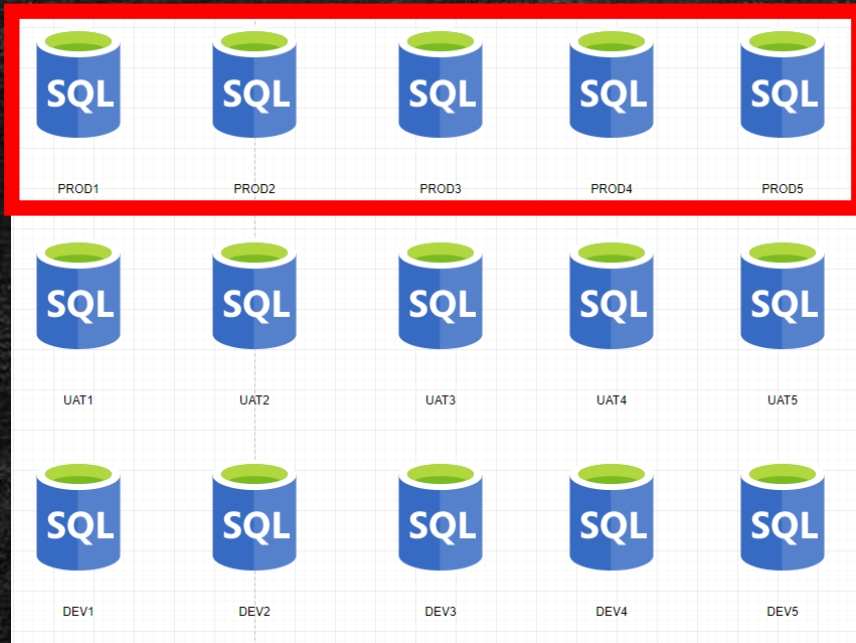
Challenges & Goals



Consider A SQL Environment



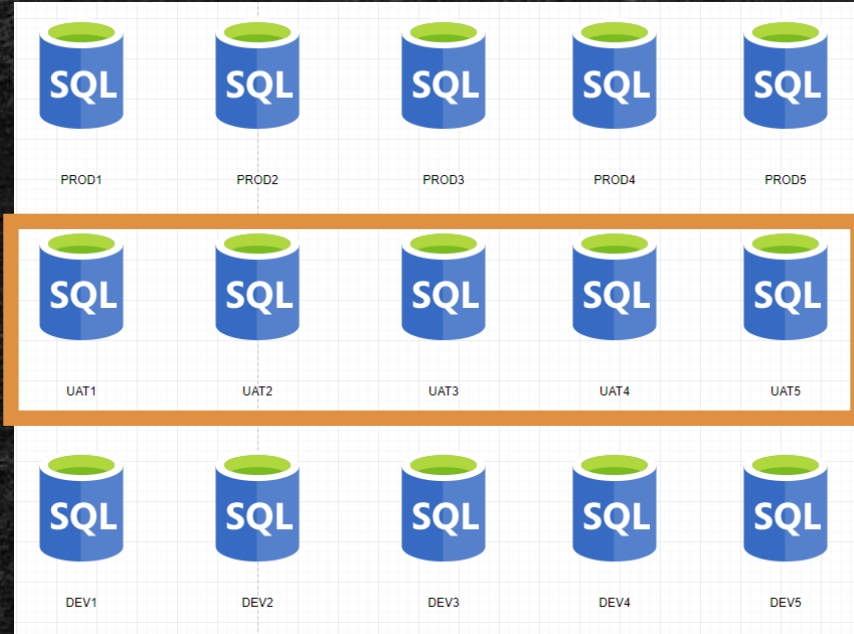
Consider A SQL Environment



PRODUCTION

- No SA account
- Backup Compression
- FULL Recovery
- Secure Backup Location
- Agent Alerts
- Production Agent Jobs
- Only Certain Production AD Groups allowed
- And much more

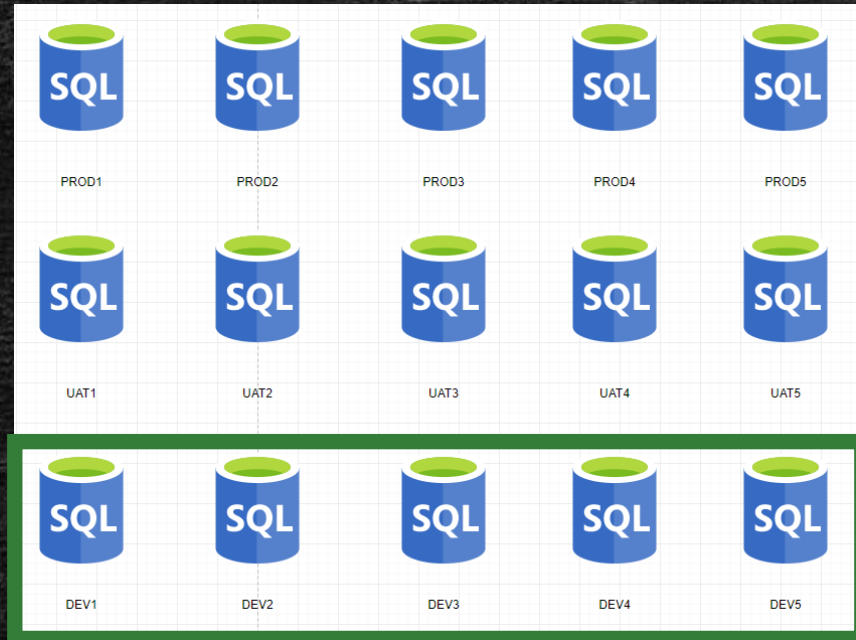
Consider A SQL Environment



TEST

- Different Backup Strategy
- Test Agent Jobs
- Needs the Testers AD Groups
- And much more

Consider A SQL Environment



Development

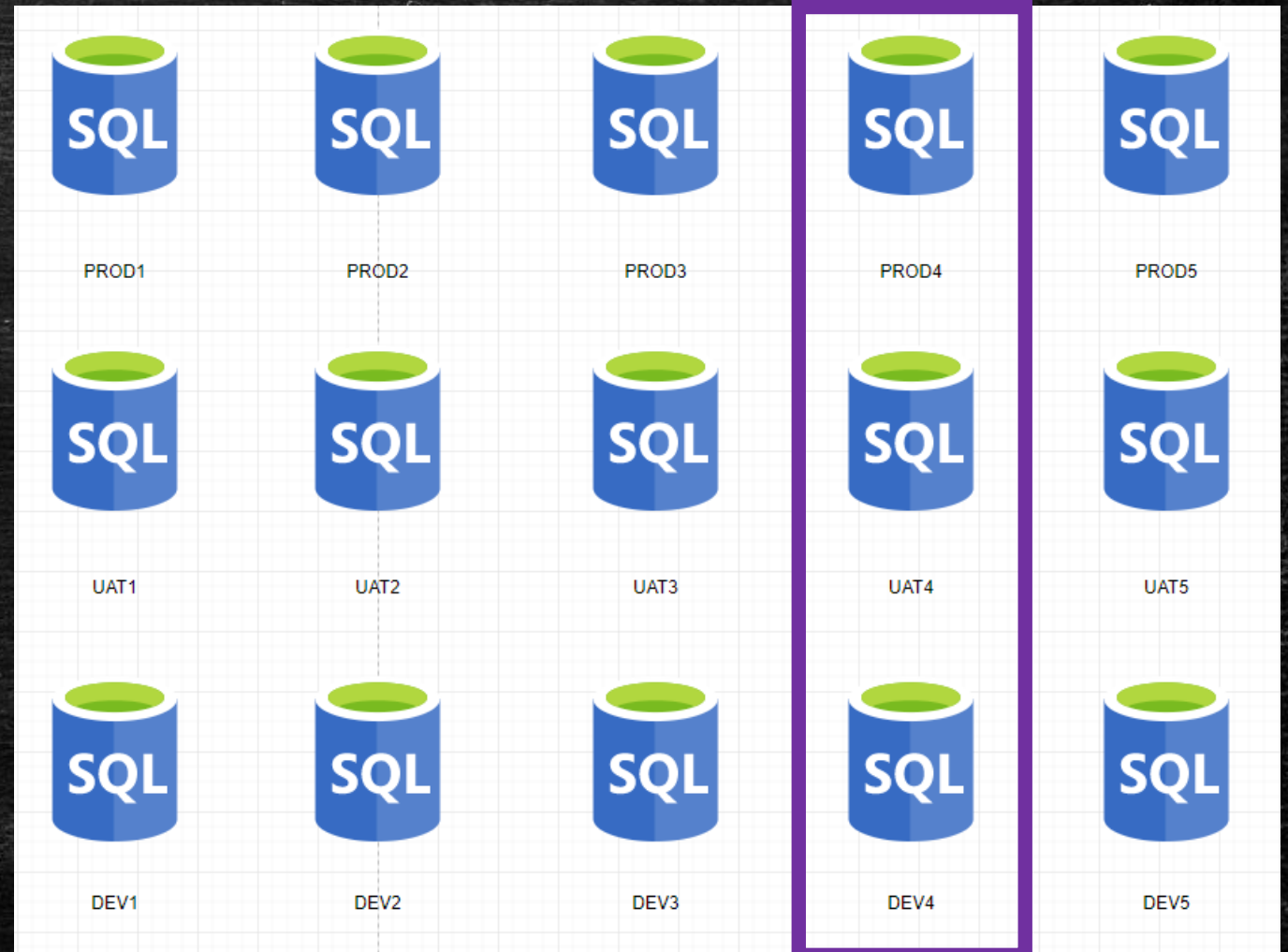
- Simple Recovery
- No Agent Jobs
- Definitely no Production AD Information
- SA Account
- And much more

Consider A SQL Environment

Widget One

Global view of one 'widgets' environments might be needed for

- Project Managers
- Team Leaders
- Release Managers
- Agile Teams

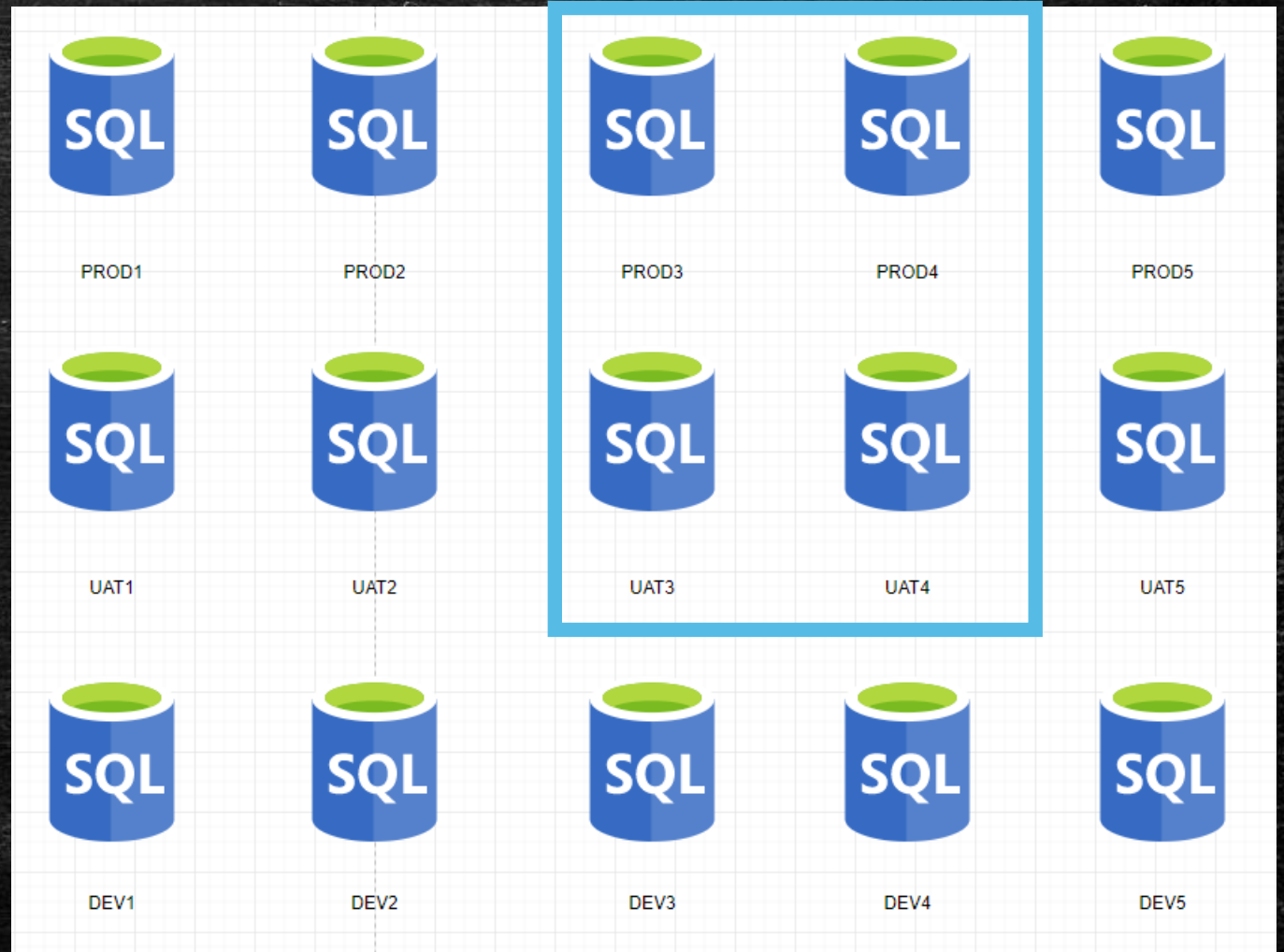


Consider A SQL Environment

Special Snowflakes

For those 'special' SQL Servers that don't fit in the usual boxes

I.E. SharePoint



Challenge - Output

```
dbachecks is awesome > Invoke-DbaCheck -SqlInstance sql0 -Check FailedJob
Executing all tests in 'C:\Users\enterpriseadmin.THEBEARD\Documents\WindowsPowerShell\Modules\dbachecks\1.1.155\checks\Agent.Tests.ps1' with Tags FailedJob
```

```
Executing script C:\Users\enterpriseadmin.THEBEARD\Documents\WindowsPowerShell\Modules\dbachecks\1.1.155\checks\Agent.Tests.ps1
```

Describing Failed Jobs

```
Context Checking for failed enabled jobs on sql0
```

```
[+] CommandLog Cleanup's last run outcome on SQL0 is Succeeded 3.8s
[+] DatabaseBackup - SYSTEM_DATABASES - FULL's last run outcome on SQL0 is Succeeded 103ms
[+] DatabaseBackup - USER_DATABASES - DIFF's last run outcome on SQL0 is Succeeded 25ms
[+] DatabaseBackup - USER_DATABASES - FULL's last run outcome on SQL0 is Succeeded 22ms
[+] DatabaseBackup - USER_DATABASES - LOG's last run outcome on SQL0 is Succeeded 23ms
[+] DatabaseIntegrityCheck - SYSTEM_DATABASES's last run outcome on SQL0 is Succeeded 50ms
[+] DatabaseIntegrityCheck - USER_DATABASES's last run outcome on SQL0 is Succeeded 84ms
[+] IndexOptimize - USER_DATABASES's last run outcome on SQL0 is Succeeded 31ms
[+] Output File Cleanup's last run outcome on SQL0 is Succeeded 34ms
[+] sp_delete_backuphistory's last run outcome on SQL0 is Succeeded 81ms
[+] sp_purge_jobhistory's last run outcome on SQL0 is Succeeded 24ms
[+] syspolicy_purge_history's last run outcome on SQL0 is Succeeded 22ms
```

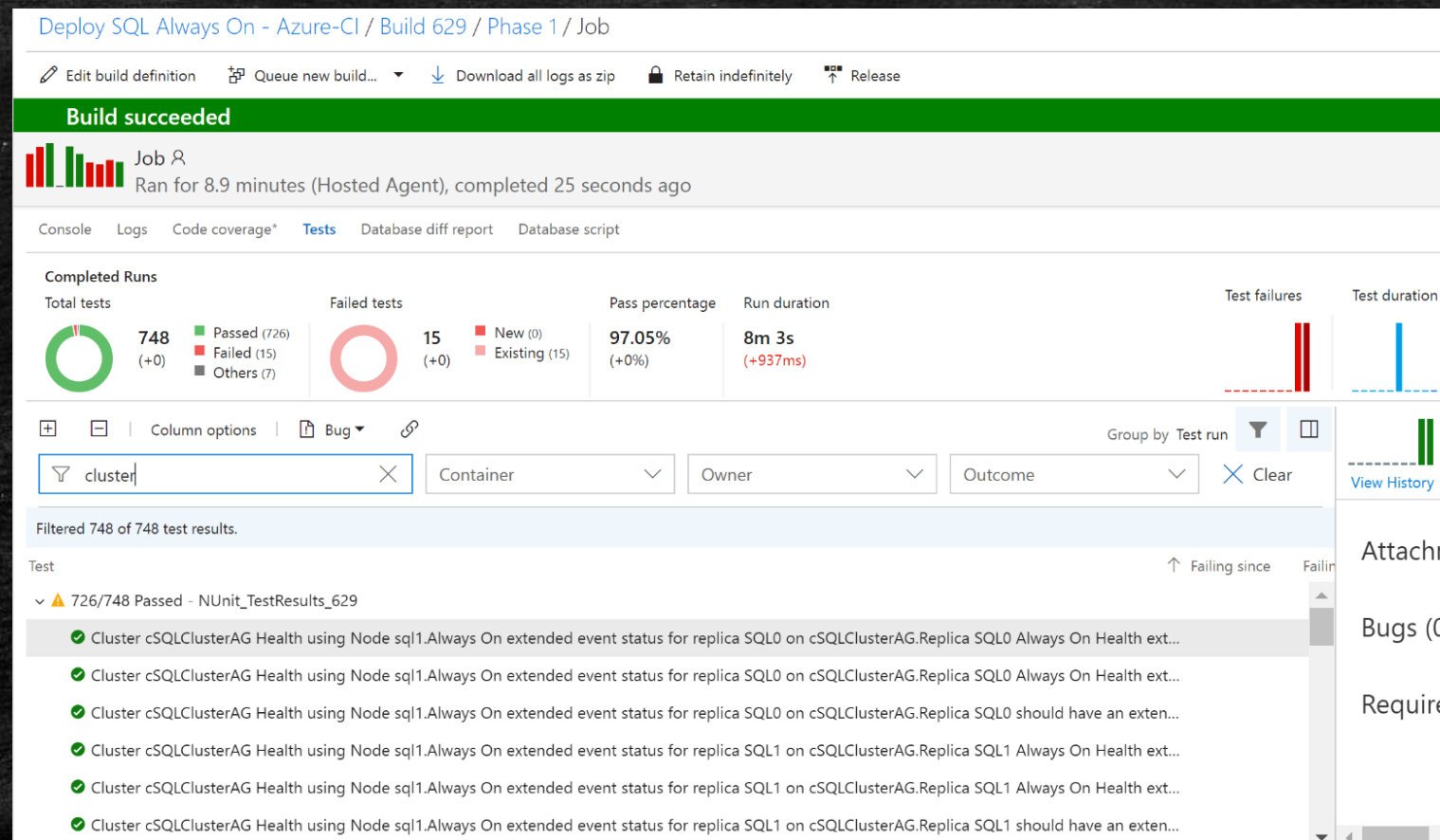
```
Tests completed in 4.31s
```

```
Tests Passed: 12, Failed: 0, Skipped: 0, Pending: 0, Inconclusive: 0
```

DBAs may need output instantly

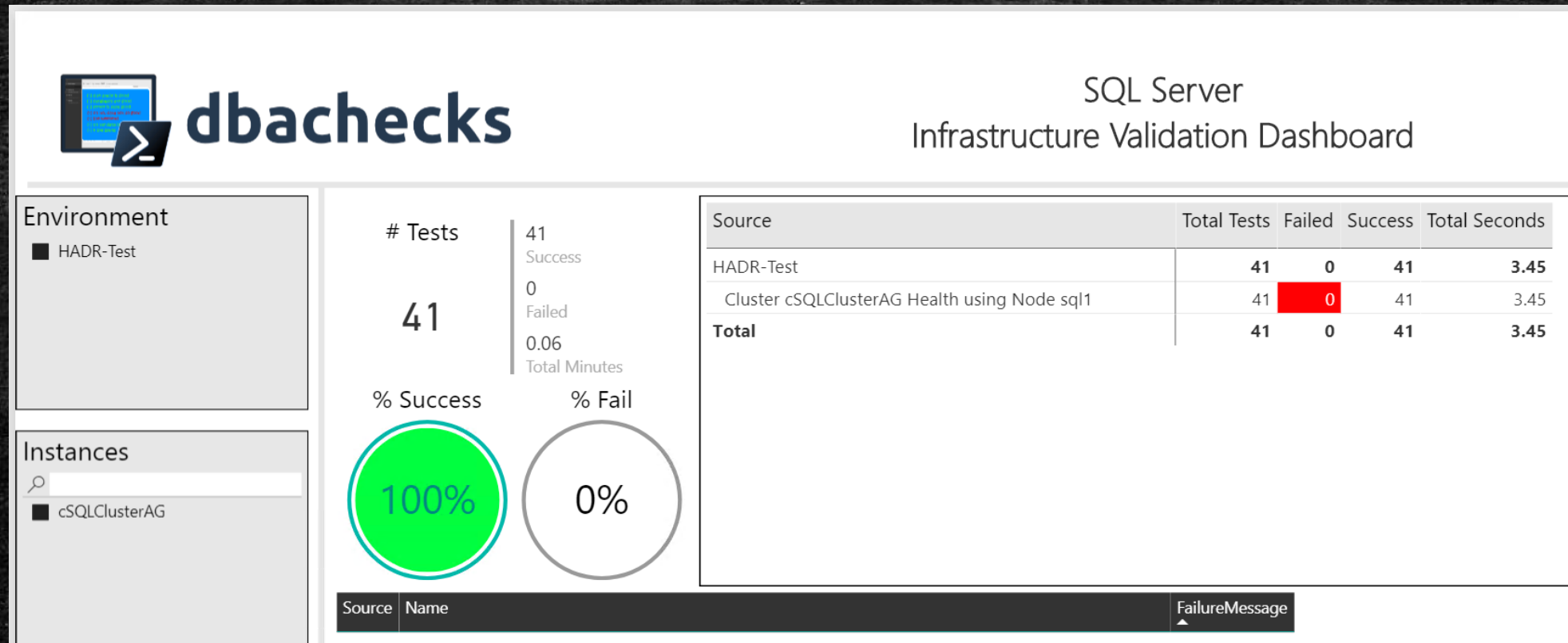
Challenge - Output

- DBAs may want to automate and integrate with other solutions (DevOps, Daily Checks, Incident Response, Maintenance Windows)



Challenge - Output

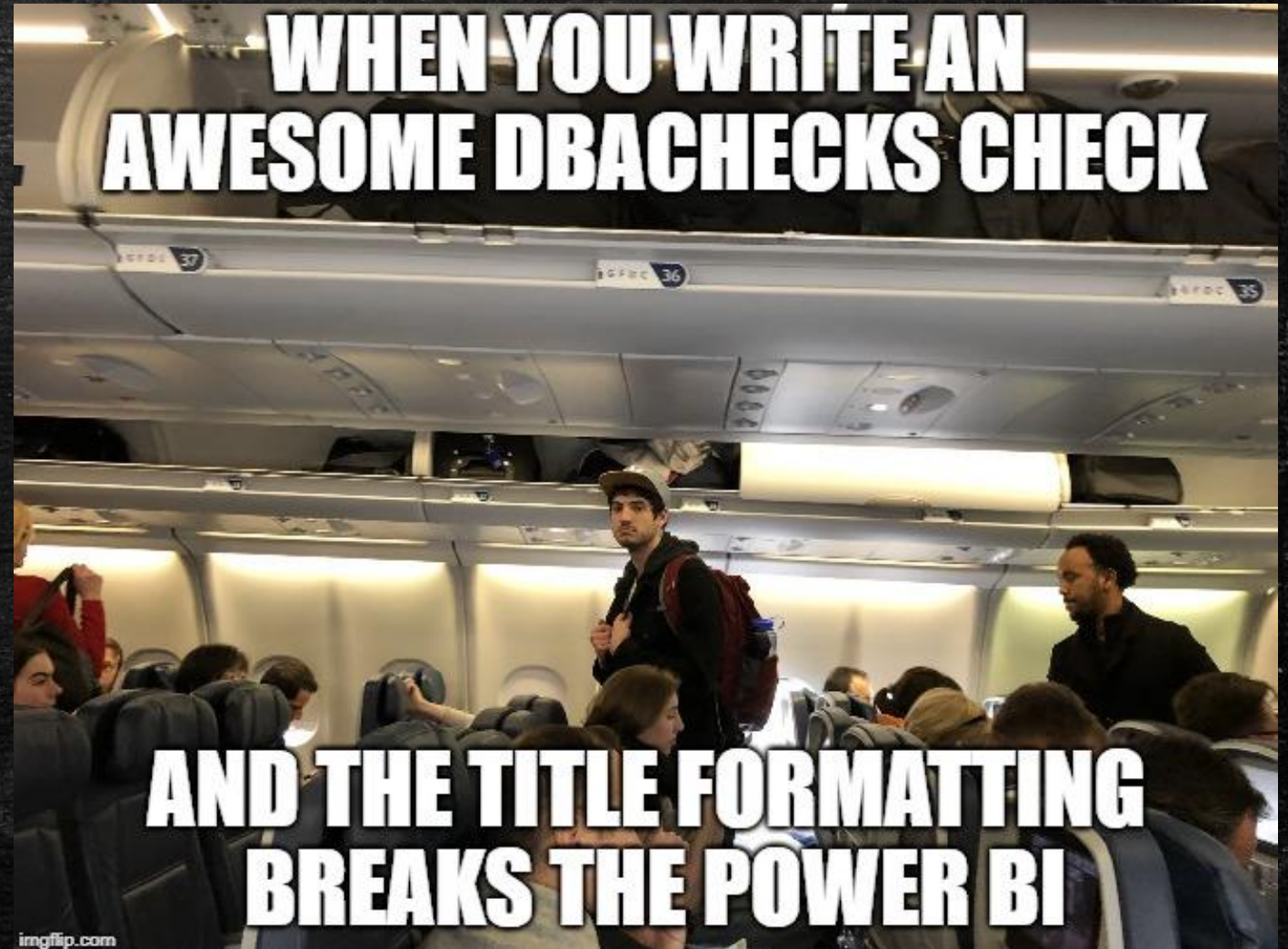
- *Management want output they understand*



– (and with pretty interactive pictures !)

Challenge - Output

- *The Power Bi template requires a specific title configuration*



Challenge – User Simplicity

End Users

- *Need simplicity to enable easy adoption*
- *Need index for Checks*
- *Need index for Configuration*
- *Simplified output options*
- *File system access to work across many differing user environments and permissions*

Solution



Installation is easy

PowerShell Gallery














```
Install-Module dbachecks
```

```
Install-Module dbachecks -Scope CurrentUser
```

** Automatically installs required modules*

dbachecks

▲ DBACHECKS

- ▶  .github
- ▶  bin
- ▶  checks
- ▶  en-us
- ▶  functions
- ▶  internal
- ▶  tests
- ▶  xml
-  .gitattributes
-  .gitignore
-  CODE_OF_CONDUCT.md
-  dbachecks.psd1
-  dbachecks.psm1

Dbachecks – Configuration

- *Using PSFramework to create configuration items*

```
Set-PSFConfig -Module dbachecks -Name app.sqlinstance -Value $null  
-Initialize -Description "List of SQL Server instances"
```

- *Stored in registry*

Dbachecks – Configuration

- *Enabling Users to set configuration*

```
Set-DbcConfig -Name app.sqlinstance -Value sql2016, sql2017
```

```
Set-DbcConfig -Name app.sqlinstance -Value sqlcluster –Append
```


What Checks Are Available? Use Get-DbcCheck

```
PowerShell > Get-DbcCheck
```


What Checks Are Available? Use Get-DbcCheck

```
dbchecks is awesome > Get-DbcCheck
```

Group		Type	UniqueTag	AllTags	Config
-----		----	-----	-----	-----
Agent		Sqlinstance	AgentServiceAccount	AgentServiceAccount, ServiceAccount, Agent	app.sqlinstance
Agent		Sqlinstance	DbasOperator	DbasOperator, Operator, Agent	app.sqlinstance agent.dbaoper...
Agent	↑	Sqlinstance	FailsafeOperator	FailsafeOperator, Operator, Agent	app.sqlinstance agent.failsaf...
Agent		Sqlinstance	DatabaseMailProfile	DatabaseMailProfile, Agent	app.sqlinstance agent.databa...
Agent		Sqlinstance	FailedJob	FailedJob, Agent	app.sqlinstance
Agent		Sqlinstance	ValidJobOwner	ValidJobOwner, Agent	app.sqlinstance agent.validjo...
Agent		Sqlinstance	AgentAlert	AgentAlert, Agent	app.sqlinstance agent.alert.S...
Database		Sqlinstance	DatabaseCollation	DatabaseCollation, Database	app.sqlinstance policy.databa...
Database		Sqlinstance	SuspectPage	SuspectPage, Database	app.sqlinstance
Database		Sqlinstance	TestLastBackup	TestLastBackup, Backup, Database	app.sqlinstance skip.backup.t...
Database		Sqlinstance	TestLastBackupVerifyOnly	TestLastBackupVerifyOnly, Backup, Database	app.sqlinstance policy.backup...
Database		Sqlinstance	ValidDatabaseOwner	ValidDatabaseOwner, Database	app.sqlinstance policy.validd...
Database		Sqlinstance	InvalidDatabaseOwner	InvalidDatabaseOwner, Database	app.sqlinstance policy.invali...
Database		Sqlinstance	LastGoodCheckDb	LastGoodCheckDb, Database	app.sqlinstance policy.dbcc.m...
Database		Sqlinstance	IdentityUsage	IdentityUsage, Database	app.sqlinstance policy.identi...
Database		Sqlinstance	RecoveryModel	RecoveryModel, DISA, Database	app.sqlinstance policy.recover...
Database		Sqlinstance	DuplicateIndex	DuplicateIndex, Database	app.sqlinstance
Database		Sqlinstance	UnusedIndex	UnusedIndex, Database	app.sqlinstance
Database		Sqlinstance	DisabledIndex	DisabledIndex, Database	app.sqlinstance
Database		Sqlinstance	DatabaseGrowthEvent	DatabaseGrowthEvent, Database	app.sqlinstance policy.databa...
Database		Sqlinstance	PageVerify	PageVerify, Database	app.sqlinstance policy.pageve...
Database		Sqlinstance	AutoClose	AutoClose, Database	app.sqlinstance policy.databa...
Database		Sqlinstance	AutoShrink	AutoShrink, Database	app.sqlinstance policy.databa...

What Checks Are Available? Use Get-DbcCheck

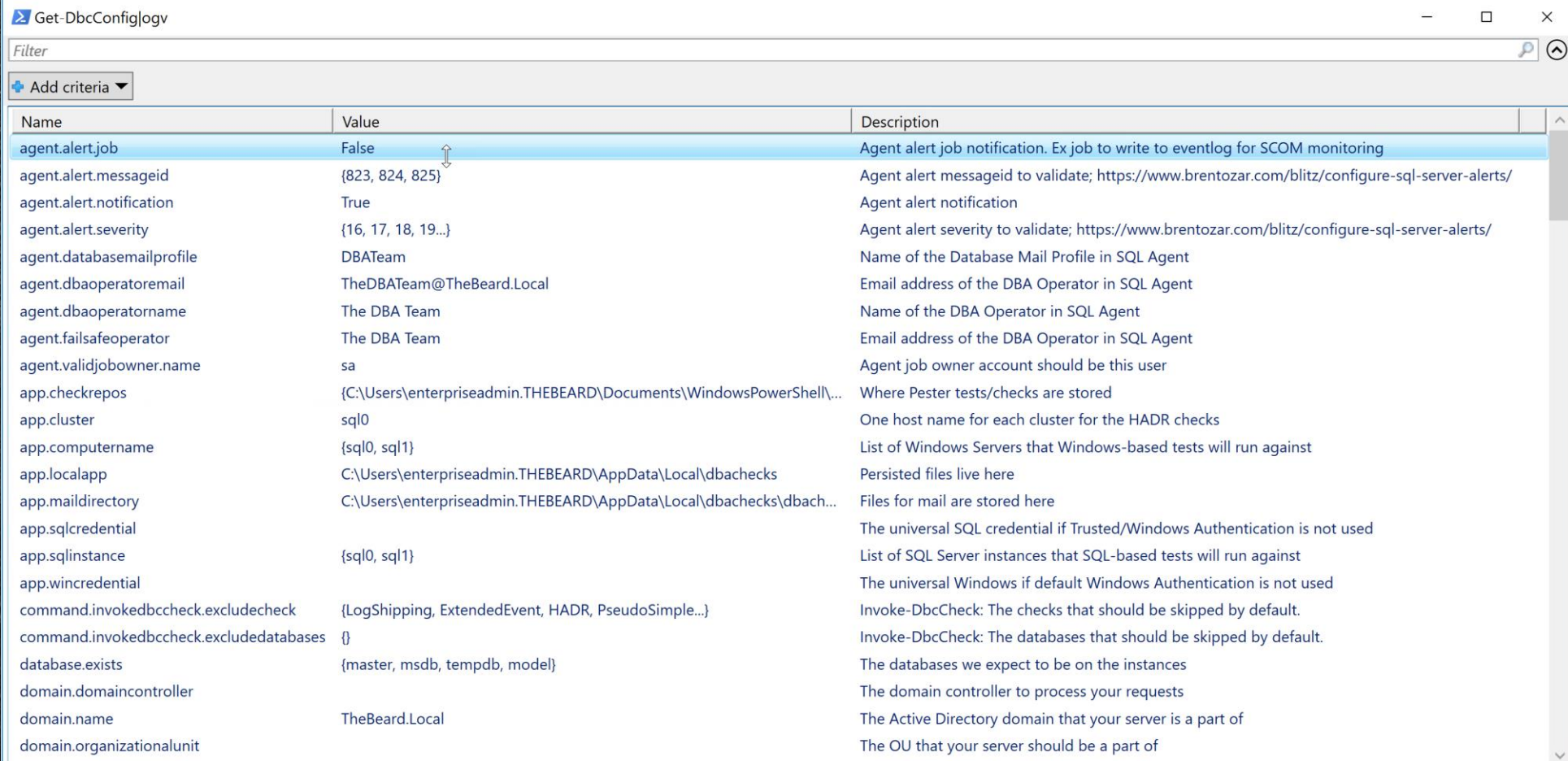
```
dbachecks is awesome > Get-DbcCheck -Pattern checkdb | fl
```

```
Group       : Database
Type        : Sqlinstance
UniqueTag    : LastGoodCheckDb
AllTags      : LastGoodCheckDb, Database
Config       : app.sqlinstance policy.dbcc.maxdays skip.dbcc.datapuritycheck
              policy.backup.newdbgraceperiod
Description : Tests that there was a Last Good DBCC CHECKDB within the specified limit (default
              blank) and if specified that the DATA_PURITY flag is set
```


What Configs Are Available? Use Get-DbcConfig

```
PowerShell > Get-DbcConfig
```


What Configs Are Available? Use Get-DbcConfig



The screenshot shows a window titled "Get-DbcConfig logv" with a search bar and a table of configuration items. The table has three columns: Name, Value, and Description. The first row, "agent.alert.job", is highlighted in blue. A mouse cursor is hovering over the "Value" cell of this row. The table lists various configuration parameters for SQL Agent, including alert settings, database mail profiles, operator information, job ownership, application paths, and domain information.

Name	Value	Description
agent.alert.job	False	Agent alert job notification. Ex job to write to eventlog for SCOM monitoring
agent.alert.messageid	{823, 824, 825}	Agent alert messageid to validate; https://www.brentozar.com/blitz/configure-sql-server-alerts/
agent.alert.notification	True	Agent alert notification
agent.alert.severity	{16, 17, 18, 19...}	Agent alert severity to validate; https://www.brentozar.com/blitz/configure-sql-server-alerts/
agent.databasemailprofile	DBATeam	Name of the Database Mail Profile in SQL Agent
agent.dbaoperatoremail	TheDBATeam@TheBeard.Local	Email address of the DBA Operator in SQL Agent
agent.dbaoperatorname	The DBA Team	Name of the DBA Operator in SQL Agent
agent.failSAFEoperator	The DBA Team	Email address of the DBA Operator in SQL Agent
agent.validjobowner.name	sa	Agent job owner account should be this user
app.checkrepos	{C:\Users\enterpriseadmin.THEBEARD\Documents\WindowsPowerShell\...	Where Pester tests/checks are stored
app.cluster	sql0	One host name for each cluster for the HADR checks
app.computername	{sql0, sql1}	List of Windows Servers that Windows-based tests will run against
app.localapp	C:\Users\enterpriseadmin.THEBEARD\AppData\Local\dbchecks	Persisted files live here
app.maildirectory	C:\Users\enterpriseadmin.THEBEARD\AppData\Local\dbchecks\dbach...	Files for mail are stored here
app.sqlcredential		The universal SQL credential if Trusted/Windows Authentication is not used
app.sqlinstance	{sql0, sql1}	List of SQL Server instances that SQL-based tests will run against
app.wincredential		The universal Windows if default Windows Authentication is not used
command.invokedbccheck.excludecheck	{LogShipping, ExtendedEvent, HADR, PseudoSimple...}	Invoke-DbcCheck: The checks that should be skipped by default.
command.invokedbccheck.excludedatabases	{}	Invoke-DbcCheck: The databases that should be skipped by default.
database.exists	{master, msdb, tempdb, model}	The databases we expect to be on the instances
domain.domaincontroller		The domain controller to process your requests
domain.name	TheBeard.Local	The Active Directory domain that your server is a part of
domain.organizationalunit		The OU that your server should be a part of

What Configs Are Available? Use Get-DbcConfig

```
dbachecks is awesome > Get-DbcConfig -Name policy.backup.fullmaxdays | fl
```



```
Name      : policy.backup.fullmaxdays
```

```
Value     : 7
```

```
Description : Maximum number of days before Full Backups are considered outdated
```


How Do I Run A Check?

```
PowerShell > Invoke-DbcCheck -SqlInstance ROB-  
XPS -Check FailedJob
```


Invoke-DbcCheck – Run A Check

```
dbchecks is awesome > Invoke-DbcCheck -SqlInstance ROB-XPS -Check FailedJob
Executing all tests in 'C:\Program Files\WindowsPowerShell\Modules\dbchecks\1.1.155\checks\Agent.Tests.ps1' with Tags FailedJob

Executing script C:\Program Files\WindowsPowerShell\Modules\dbchecks\1.1.155\checks\Agent.Tests.ps1

Describing Failed Jobs

Context Checking for failed enabled jobs on ROB-XPS
  [+] CommandLog Cleanup's last run outcome on ROB-XPS is Succeeded 1.93s
  [+] DatabaseBackup - SYSTEM_DATABASES - FULL's last run outcome on ROB-XPS is Succeeded 25ms
  [+] DatabaseBackup - SYSTEM_DATABASES - FULL - Big_Beard_NAS's last run outcome on ROB-XPS is Succeeded 25ms
  [+] DatabaseBackup - USER_DATABASES - DIFF's last run outcome on ROB-XPS is Succeeded 21ms
  [+] DatabaseBackup - USER_DATABASES - FULL's last run outcome on ROB-XPS is Succeeded 23ms
  [+] DatabaseBackup - USER_DATABASES - FULL - Big_Beard_NAS's last run outcome on ROB-XPS is Succeeded 23ms
  [+] DatabaseBackup - USER_DATABASES - LOG's last run outcome on ROB-XPS is Succeeded 32ms
  [+] DatabaseIntegrityCheck - SYSTEM_DATABASES's last run outcome on ROB-XPS is Succeeded 24ms
  [+] DatabaseIntegrityCheck - USER_DATABASES's last run outcome on ROB-XPS is Succeeded 49ms
  [+] FreeSpace Beard Collection's last run outcome on ROB-XPS is Succeeded 102ms
  [+] IndexOptimize - USER_DATABASES's last run outcome on ROB-XPS is Succeeded 33ms
  [+] Log SP_Blitz to table's last run outcome on ROB-XPS is Succeeded 35ms
  [!] Log SP_WhoIsActive to Table's last run outcome on ROB-XPS is unknown 20ms
  [+] Output File Cleanup's last run outcome on ROB-XPS is Succeeded 29ms
  [+] Rationalised Database Restore Script for database's last run outcome on ROB-XPS is Succeeded 24ms
  [+] sp_delete_backuphistory's last run outcome on ROB-XPS is Succeeded 73ms
  [+] sp_purge_jobhistory's last run outcome on ROB-XPS is Succeeded 52ms
  [+] SSIS Server Maintenance Job's last run outcome on ROB-XPS is Succeeded 34ms
  [+] syspolicy_purge_history's last run outcome on ROB-XPS is Succeeded 82ms

Tests completed in 2.65s
Tests Passed: 18, Failed: 0, Skipped: 1, Pending: 0, Inconclusive: 0
```


Configurations?

```
> # Set the instances to check  
> Set-DbcConfig -Name app.sqlinstance Rob-XPS  
> Invoke-DbcCheck -Check OlaInstalled
```


Configurations?

```
dbachecks is awesome > Invoke-DbcCheck -Check OlaInstalled
Executing all tests in 'C:\Program Files\WindowsPowerShell\Modules\dbachecks\1.1.155\
Executing script C:\Program Files\WindowsPowerShell\Modules\dbachecks\1.1.155\
Describing Ola maintenance solution installed

Context Checking the CommandLog table on Rob-XPS
[-] The CommandLog table exists in master on Rob-XPS 826ms
Expected 1, because The command log table is required, but got 0.
10:          @($db.tables | Where-Object name -eq "CommandLog")
at <ScriptBlock>, C:\Program Files\WindowsPowerShell\Modules\dbachecks

Context Checking the Ola Stored Procedures on Rob-XPS
[-] The stored procedures exists in master on Rob-XPS 422ms
Expected 4, because The stored procedures are required for Olas jobs to
15:          ($db.StoredProcedures | Where-Object { $psitem.schema -eq 'master' })
ld -Be $OlaSPs.Count -Because 'The stored procedures are required for Olas jobs to
at <ScriptBlock>, C:\Program Files\WindowsPowerShell\Modules\dbachecks

Tests completed in 1.25s
Tests Passed: 0, Failed: 2, Skipped: 0, Pending: 0, Inconclusive: 0
```


Configurations?

I don't have Ola installed in the default database (master)

```
> Set-DbcConfig -Name policy.ola.database -value  
DBA-Admin  
  
> Invoke-DbcCheck -Check OlaInstalled
```


Configurations?

```
dbachecks is awesome > Invoke-DbcCheck -Check OlaInstalled
Executing all tests in 'C:\Program Files\WindowsPowerShell\Modules\dbach
Executing script C:\Program Files\WindowsPowerShell\Modules\dbachecks\1.

Describing Ola maintenance solution installed

Context Checking the CommandLog table on Rob-XPS
    [+] The CommandLog table exists in DBA-Admin on Rob-XPS 363ms

Context Checking the Ola Stored Procedures on Rob-XPS
    [+] The stored procedures exists in DBA-Admin on Rob-XPS 141ms
Tests completed in 505ms
Tests Passed: 2, Failed: 0, Skipped: 0, Pending: 0, Inconclusive: 0
```


Export-DbcConfig/Import-DbcConfig

```
Export-DbcConfig -Path C:\Users\Beard\git\PesterConfigs\Application1_PROD.json
```

```
Export-DbcConfig -Path C:\Users\Hair\git\PesterConfigs\Client1_System2_Quick.json
```

```
Import-DbcConfig -Path Git:\PesterConfigs\Application1_PROD.json
```

```
Invoke-DbcCheck
```

```
Import-DbcConfig -Path Git:\PesterConfigs\Client1_System2_Quick.json
```

```
Invoke-DbcCheck
```


Start-DbcPowerBi

```
Import-DbcConfig -path Git:\PesterConfigs\App1_Dev.json
```

```
Invoke-DbcCheck -Show Summary -PassThru | Update-DbcPowerBiDataSource -Environment Development
```

```
Import-DbcConfig -path Git:\PesterConfigs\App1_Test.json
```

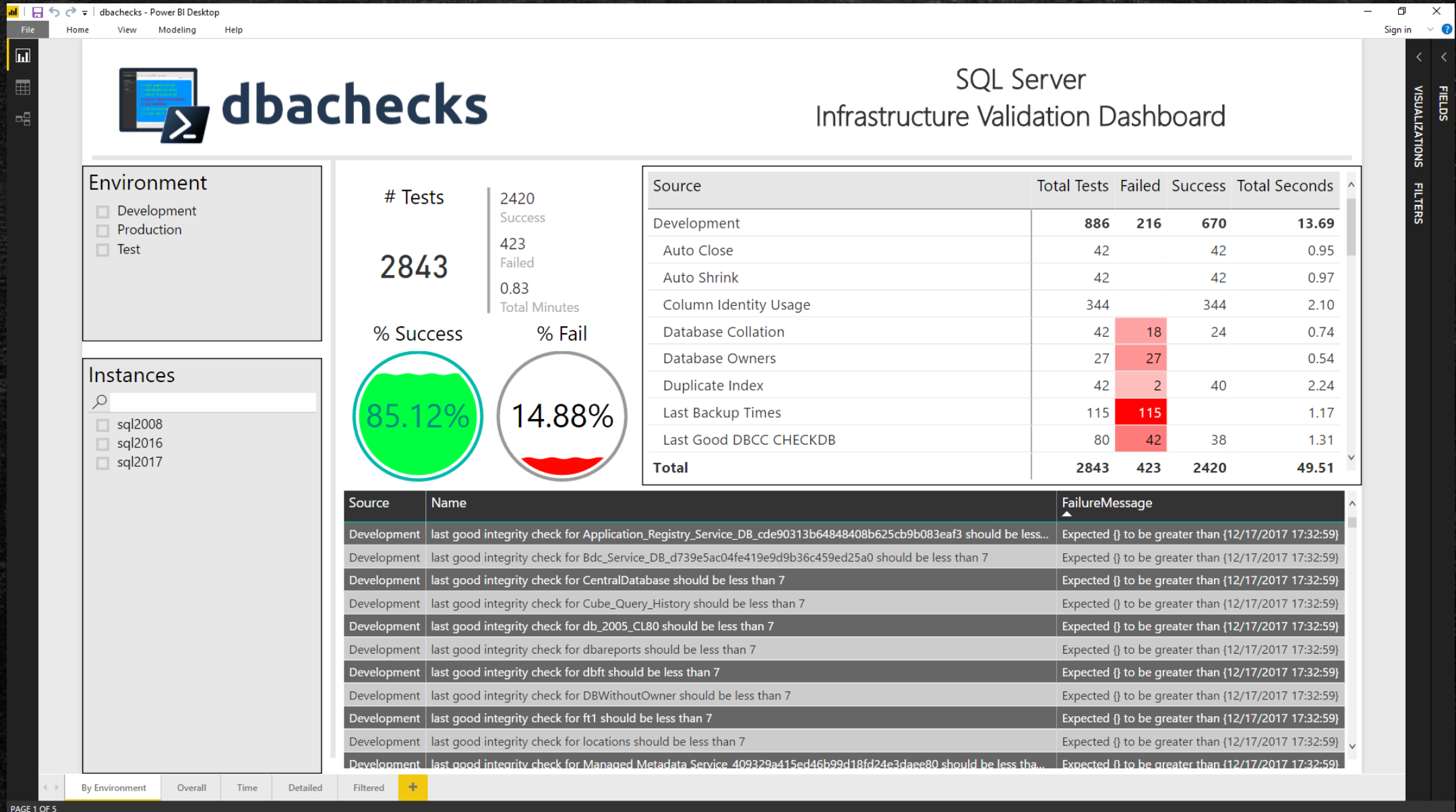
```
Invoke-DbcCheck -Show Summary -PassThru | Update-DbcPowerBiDataSource -Environment Test
```

```
Import-DbcConfig -path Git:\PesterConfigs\App1_PROD.json
```

```
Invoke-DbcCheck -Show Summary -PassThru | Update-DbcPowerBiDataSource -Environment Production
```

```
Start-DbcPowerbi
```



Start-DbcPowerBi





Got your own Checks?


```
Set-DbcConfig -Name app.checkrepos -Value 'Path To PesterTests'
```


▲ PESTERTESTS



 ETL.Tests.ps1

 HADR.Tests.ps1

 Morning.Tests.ps1

 OnCall.Tests.ps1

 Pester.Tests.ps1

  ServiceDesk.Tests.ps1

 UAT.Tests.ps1

Hey Beardy !

***MUST BE TIME
FOR A DEMO***



Questions?

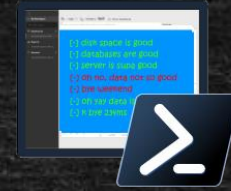


I mustache you a
Question

Rob Sewell @sqldbawithbeard
<https://sqldbawithabeard.com>

Quick Poll

Contact Me



@sqldbawithbeard



robsewellsqldb



sqldbawithAbeard.com



mrrobsewell@outlook.com



RobSewell.info

What is Pester?

- *Test Runner for PowerShell*
- *A Unit Testing Framework*
- *An Infrastructure Testing Framework*
- *<https://github.com/pester>*

