Monitoring Errors from the SQL Server Error Log:

-Open Management Studio

-Ensure SQL Server Agent is running

Sqlps.exe

$svr = New-Object(“Microsoft.SqlServer.Management.SMO.Server”) “localhost”

$err = $svr.ReadErrorLog()

$err | Select-String –InputObject {$\_.text} –Pattern ‘Error: ‘ –Context 0,1

Advanced Monitoring Errors from the SQL Error Log:

$sqlServer = New-Object (“Microsoft.SqlServer.Management.Smo.Server”)

“localhost”

$sqlServer.ReadErrorLog() | Where {$\_.Text –like “Error\*” –or $\_.Text –like “\*error\*” –or $\_.Text –like “\*Fail\*” –and {$\_.Text –notlike “\*without errors\*”}}

# Search Error log wrap text and search only for

$s = New-Object (“Microsoft.SqlServer.Management.smo.Server”) “localhost”

$dt = $s.ReadErrorLog()

$dt | ? {$\_.text –match $filter –and $\_.logdate –ge (get-date).AddHours (-1)} |

Select LogDate, ProcessorInfo, Text | formtal-table –Auto –Wrap

Obtaining Instance Information on Multiple Instances:

SQL Server PowerShell

$Servers = “localhost”

$Query = “SELECT SERVERSPROPERTY(‘ServerName’) AS ServerName, SERVERPROPERTY(‘ProductVersion’) AS Version, SERVERPROPERTY(‘Edition’) AS Edition

# Each server within the location SQLSERVER:\SQL\$server

$Servers | ForEach-Object {$server = “$\_”;  
Set-Location SQLSERVER:\SQL\$server

# Invoke the SQL Query in each Server located in SQLSERVER:\SQL\$server

Invoke-Sqlcmd –Query $Query –ServerInstance $Server}

Enabling PS remoting on SQL instances:

# To check PS remoting Service & Start the Service

Get-Service winrm | Start-Service

# Show all Providers

Get-PSProvider

# Set the item as trustedhosts for all Providers

Set-Item trustedhosts \*

Error Handling:

SQL Server PowerShell.exe

Set-Location SQLSERVER:\SQL\localhost\DEFAULT\DATABASES\MissingDB

-ErrorAction Stop Write-Host –ForegroundColor Red “Ops”

Set-Location SQLSERVER:\SQL\localhost\DEFAULT\DATABASES\MissingDB

-ErrorAction SilentlyContinue

$?

Write-Host –ForegroundColor Red “Ops”

$?

Set-Location SQLSERVER:\SQL\localhost\DEFAULT\DATABASES\MissingDB

-ea SilentlyContinue

$Error[0] | Get-Member –MemberType Property

Set-Location SQLSERVER:\SQL\localhost\DEFAULT\DATABASES\MissingDB

If (!$?) {Write-Host “ExceptionMessage:” $error[0].Exception.Message

Write-Host “Target Object: “ $error[0].TargetObject Write-Host “Category Info: “error[0].CategoryInfo Write-Host “ErrorID: “ $Error[0].FullyQualifiedErrorId}

Set-Location SQLSERVER:\SQL\localhost\DEFAULT\DATABASES\MissingDB

-ErrorVariable err1 –ErrorAction SilentlyContinue

$err1

Restoring All Databases from Most Recent Backup:

SQLQuery2.sql – SQL093.master (ADVENTUREBIKE\SQL Admin (60))

Microsoft SQL Server Management Studio (Administrator)

SQL093 (SQL Server 11.0.2100 – ADVENTUREBIKE\SQLAdmin

Database

System Database

SQLQuery2.sql

SELECT TOP1

Bs.Server\_Name,

Db.name As DBName,

Bs.backup\_finish\_date,

Mf.physical\_device\_name

FROM [master].[dbo].[sysdatabases] db

LEFT OUTER JOIN [msdb].[dbo].[backupset] bs

ON bs.database\_name = db.name

AND bs.type = ‘D’

JOIN msdb.dbo.backupmediaset ms

On ms.media\_set\_id = bs.media\_set\_id

JOIN msdb.dbo.backupmediafamily mf

On mf.media\_set\_id = ms.media\_set\_id

WHERE db.name = ‘$(Database.name)’

AND mf.physical\_device\_name LIKE ‘\\%’

ORDER BY bs.backup\_finish\_date desc

# PowerShell.exe must define SMO

[system.Reflection.Assembly]::LoadWithPartialName

(“Microsoft.SQLServer.Smo”)

$ServerName = “localhost”

$server = New-Object Microsoft.SqlServer.Management.Smo.Server

$ServerName

ForEach ($Database in $Server.Databases) {

“Processing database $($Database.Name): “

$CMD = “

SELECT TOP1

Bs.Server\_Name,

Db.name AS DBName,

Bs.backup\_finish\_date,

Mf.physical\_device\_name

FROM [master].[dbo].[sysdatabase] db

LEFT OUTER JOIN [msdb].[dbo].[backupset] bs

ON bs.database\_name = db.name

AND bs.type = ‘D’

JOIN msdb.dbo.backupmediaset ms

ON ms.media\_set\_id = bs.media\_set\_id

JOIN msdb.dbo.backupmediafamily mf

ON mf.media\_set\_id = ms.media\_set\_id

WHERE db.name = ‘$($Database.Name)’

AND mf.physical\_device\_name LIKE ‘\\%’

ORDER BY bs.backup\_finish\_date desc”

$Backups = ($Server.ConnectionContext.ExecuteWithResults

($CMD)).Tables[0]

Foreach ($Backup in $Backups) {

$CMD = “RESTORE FILELISTONLY FROM DISK = `$($Backup.physical\_device\_name)’”

$DBFiles = ($Server.ConnectionContext.ExecuteWithResults

($CMD)).Tables[0]

$DBFiles | ft –Auto

}

}

Finding Disk Space for SQL Files:

PS.exe

$Filter = @{Expression = {$\_.Name};Label=”DiskName”},

@{Expression={$\_.Label};Label=”Label”},

@{Expression={$\_.FileSystem};Label=”FileSystem”},

@{Expression={[int]$($\_.BlockSize/1KB)};Label=”BlockSizeKB”},

@{Expression={[int]$($\_.Capacity/1GB)};Label=”CapacityGB”},

@{Expression={[int]$($\_.Freespace/1GB)};Label=”FreeSpaceGB”}

Get-WmiObject Win32\_Volume | Format-Table $Filter –AutoSize

Get-ChildItem C:\Backups | Sort-Object –Property Name –Descending

Dir C:\Backups –Recurse

Dir C:\ -include \*.ldf | Select Name, Length –Last 8

Dir C:\ -include \*.mdf | Select Name, Length –Last 8

Displaying Results in Grid View:

Import-Module ServerManager

Add-WindowsFeature PowerShell-ISE

Get-Service | Out-GridView

Get-Service | Export-Clixml C:\Temp\Test.xml

Import-Clixml C:\Temp\Test.xml

Sqlps Utility:

SQL Server PowerShell.exe

[appdomain]::CurrentDomain.GetAssemblies() | Select \*

[appdomain]::CurrentDomain.GetAssemblies() | Where {$\_.FullName –like “\*sql\*”}

PowerShell.exe

Sqlps

[appdomain]::CurrentDomain.GetAssemblies() | Where {$\_.FullName –Like “\*sql\*”}

Creating SQL Audits:

SQL Server PowerShell.exe

# Create SMO on local server

$server = New-Object Microsoft.SqlServer.Management.SMO.Server “localhost”

$audit = New-Object Microsoft.SqlServer.Management.SMO.Audit($server, “Test Audit”)

# To specify Audit Destination to a File

$audit.DestinationType = [Microsoft.SqlServer.Management.SMO.AuditDestinationType] ‘File’

# To specify Audit Folder Path

$audit.FilePath = “C:\Audit”

# To create Audit folder

$audit.Create()

SQLSERVER Drive and Invoke Sqlcmd cmdlets:

SQL Server PowerShell

Invoke-Sqlcmd –Query “SELECT @@VERSION” –QueryTimeout 1

Invoke-Sqlcmd –Query “WAITFOR DELAY ’10:00’ SELECT @@VERSION ”