Using PowerShell to Monitor SQL Server

Over the past few years, I have used a number of approaches to monitor SQL Server. For example, I have configured jobs to e-mail the DBA team if they fail. I have written programs to check that everything is OK. I have used a number of third-party tools to monitor my instances. Each approach has its own strengths and weaknesses. If the SQL Server Agent is not running, the jobs don’t run and I don’t get any messages telling me that they have failed. To stop this happening, I have used server monitoring tools for example Big Brother http://www.bb4.org/ which can warn if a service has stopped. This sort of tool is pretty good, but you need to invest time and effort setting it up. The same applies to monitoring tools like Ideas Diagnostic Manager or Red Gates SQL Monitor. Also, these tools can cost a lot of money and can themselves affect performance of the server.

 So I want to find a tool to monitor SQL that is independent of SQL Server, free and easy to use. PowerShell fits the first two requirements, and it is not too difficult to learn. I’ll suggest an approach to monitor your backups that took me a few hours to implement, is independent of SQL server, takes few resources and costs nothing apart from your time.

 I have a production database in full recovery. A scheduled SQL Server Agent backs up the log on the hour between 07:00 and 19:00. If the job to backup the log fails, it will e-mail the SQL Server DBA group. However, if the SQL Agent fails there is a danger that there will be no backup notification. To address this issue, I have written a PowerShell job that runs on a monitor server (Windows 2003) and checks that the log file is in the backup directory. If the file is not present, the PowerShell script e-mails the DBA team.

 The first thing that I have written is a PowerShell function library. Create a directory called C:\PSScripts. Create this a file in this directory called CalcFunctions.ps1 and paste the following code into this file:

function SendMail($filename){$emailFrom = "SQLMonitor@myco.com"$emailTo = "SQLDBAs@myco.com "$subject = "Missing backup file - $filename does not exist"$body = "Missing backup file - $filename does not exist"$smtpServer = "mail1"$smtp = new-object Net.Mail.SmtpClient($smtpServer)$smtp.Send($emailFrom, $emailTo, $subject, $body)} function CheckBackupFile($filename){$bck = get-childitem $filename | where {((get-date) - $\_.LastWriteTime).minutes -lt 9}if (!$bck) { SendMail($filename)}}

There are two functions

The first function – SendMail – will send an e-mail to whoever you configure in your code. I am sending the email to a mail group called SQLDBAs. The function accepts a parameter – filename – which indicates the name of the missing log file, which forms part of the message.

The second function – CheckBAckupFile – is more interesting. It accepts as a parameter the name of a backup file. It then loads a list of files matching the parameter $filename (get-childitem $filename) which were written to in the last 9 minutes (where {((get-date) - $\_.LastWriteTime).minutes -lt 9}). If this list is empty the SendMail function is invoked with the $filename as a parameter. The 9 minutes is used because the job will be scheduled to run at 7 minutes past the hour, and there should always be a log file for the production database available at this time.

 Now create a script to use these functions – create a file called ChackLogBackups.ps1 and paste the following code into it:

param([string]$logFilename = "") . c:\psscripts\CalcFunctions.ps1checkbackupfile($logFilename)

In this script, note that there is a space between the dot and the c: (. \) – originally I missed the space and it took me a long time to identify the problem.

 Next, create a batch file RunPS1.bat as follows:

powershell.exe -command "& 'c:\psscripts\checklogbackups.ps1'" “\\sql\r$\sqlbackup\log\_prod\*.sqb”

In the above file, the script file checkLogBAckups is run with a parameter pointing to the location of the backup log file. This is on disk at \\sql1\r$\sqlbackup. The wildcard (‘\*’) is used because the backup files are named using a datetime stamp – for example log\_prod20110801\_09:00.sqb.

 Finally, create a scheduled job with the windows scheduler. This job must run at 7 minutes past the hour, and run the batch file RunPS1.

 This is a simple example. It can be extended to check for other log files on other servers, or overnight full backups. By creating a file of files to be checked, it will be possible to be certain that your production databases are being backed up as expected with little cost, effort or expense