Test SQL Server with Windows PowerShell - Part 1

Instead of using Vbscript, bat files, or sql client implementations, ... we'll show you the power of Windows PowerShell in checking SQL Servers status.

Muthusamy

Network Administration - *This series will introduce you to the methods and procedures to check the current state of the operating system, instances of SQL Server and database, . with Windows PowerShell.*

Instead of using Vbscript, bat files, or sql client implementations, . we'll show you the power of Windows PowerShell in checking SQL Servers status.

Prerequisites

1. Install .Net 2.0

1. On the client computer, you need to install Windows PowerShell 1.0.

1. Your login needs to have permission to create folders and files in the client.

Before going into the actual SQL Server test, we want to introduce a bit of platform and build directories, libraries, .

At the end of this series, there will be a Powershell library with many functions used as a source for any PowerShell script. One, many or all of these functions can be called from any PowerShell script when the library is used as the source.

Step 1

Launch Windows PowerShell by executing the command below (Figure 1.0)

```
% SystemRoot% system32WindowsPowerShellv1.0powershell.exe
```

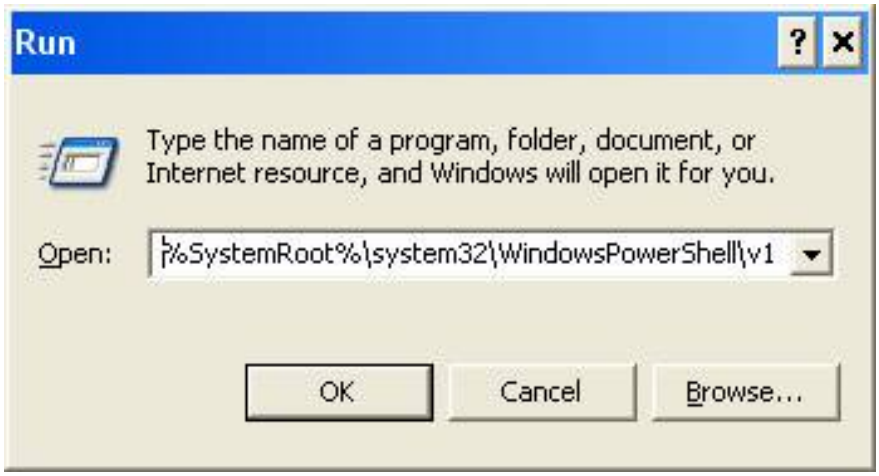

Figure 1.0: Launch PowerShell

Step 2

Create a directory using the PowerShell command below. This directory will be used exclusively for PowerShell scripts, libraries, and functions that are related to testing SQL Server. Execute the command given below (see Figure 1.1)

```
New-Item -Path C: -Name CheckSQLServer -Type directory
```

You can see the result of executing the command below. Basically, the CheckSQLServer folder has been successfully created.

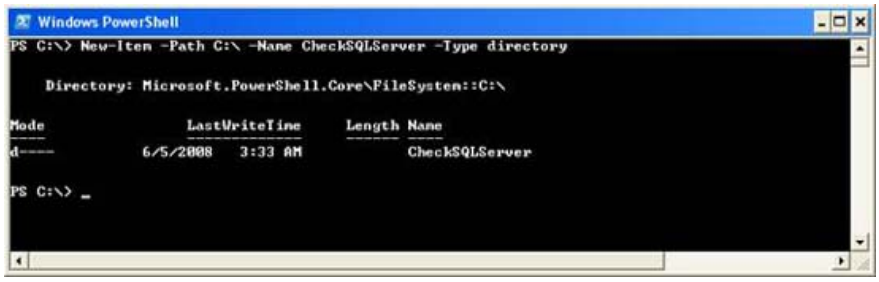

Figure 1.1 Create new folder

Step 3

Navigate to the CheckSQLServer folder and then create other files using the following PowerShell command (see Figure 1.3).

```
set-location C: CheckSQLServer
Notepad CheckSQL_Lib.ps1
Notepad CheckSQLServer.ps1
Notepad Pinghost.ps1
```

You can see that the location is changed to C: CheckSQLServer and also opened three notepad windows to be able to edit the files CheckSQL_Lib.sql, CheckSQLServer.Ps1 and PingHost.ps1.

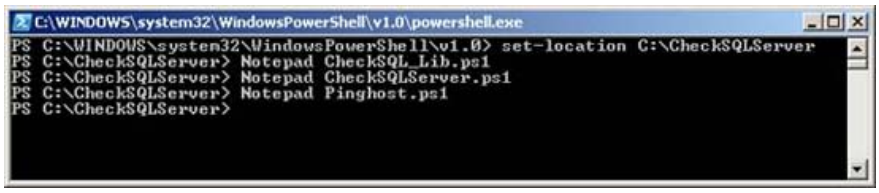

Figure 1.2: Create a PowerShell script and folder

Note: Notepad is an editor that we use for editing PowerShell scripts. You can completely customize other editors for you personally.

Step 4

Type or copy / paste the following code into the notepad editor that opened PingHost.ps1 as shown below (see Figure 1.3)

```
Function Pinghost ([string] $ Hostname)
{
$ status = get-wmiobject win32_pingstatus -Filter "Address = '$ Hostname'" |  S
if ($ status.statuscode -eq 0)
{write-host $ Hostname is REACHABLE -background "GREEN" -foreground "BLACk"}
else
{write-host $ Hostname is NOT reachable -background "RED" -foreground "BLACk"}
}
```

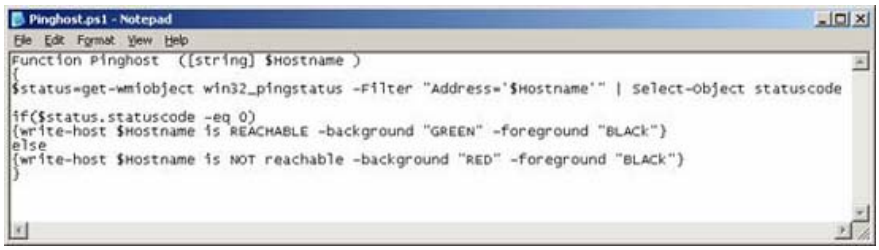

Figure 1.3: PingHost.ps1 script file

Save the Pinghost.ps1 file and exit notepad.

Step 5

Type or copy / paste the following code into the notepad editor that opened CheckSQL_Lib.ps1 as shown below (see Figure 1.4).

```
#Source all the relate to functions CheckSQL
.  ./PingHost.ps1
```

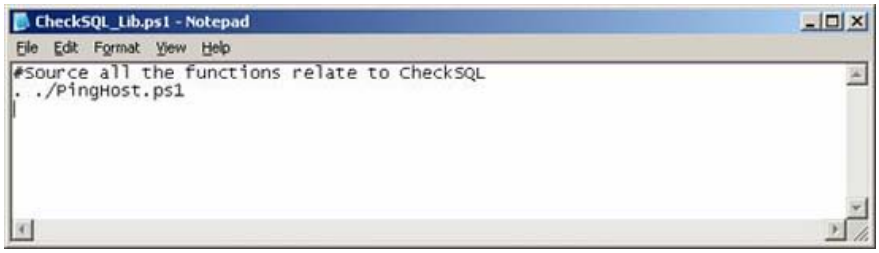

Figure 1.4: CheckSQL_Lib.ps1

Save the file CheckSQL_Lib.ps1 and exit notepad.

Note : This CheckSQL_Lib.ps1 will be updated with sources of new scripts like PingHost.PS1

Basically the source will load the functions listed in the script file and make it available throughout the PowerShell session.

Step 6

Type or copy / paste the following code into the notepad editor that opened CheckSQLServer.ps1 as shown below (see Figure 1.5).

```
#Objective: To check various status of SQL Server
#Host, instances and databases.
#Author: MAK
#Date Written: June 5, 2008
param (
[string] $ Hostname
)
.  ./CheckSQL_Lib.ps1
PingHost $ Hostname
```

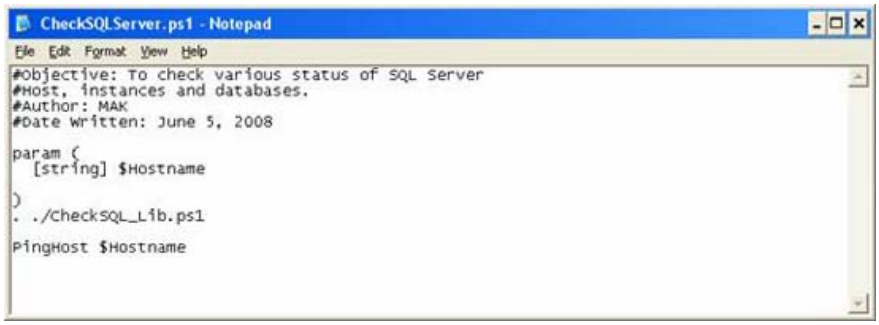

Figure 1.5: CheckSQLServer.ps1

Save the file CheckSQLServer.ps1 and exit notepad

Note : This CheckSQLServer.ps1 file will be updated with new conditions and parameters in later sections of this series.

The source will load the functions listed in the script file and make it available to the entire PowerShell session. In this case, we will source a script that uses multiple sources from other scenarios.

Step 7

Execute CheckSQLServer.ps1 as shown below (see Figure 1.6).

```
./CheckSQLServer.ps1 PowerMachine
./CheckSQLServer.ps1 TestServer
```

We will see the results, based on which you will know if the computer has ping capability. If the machine can be probed by ping, the message will be marked in green and it will be marked in red.

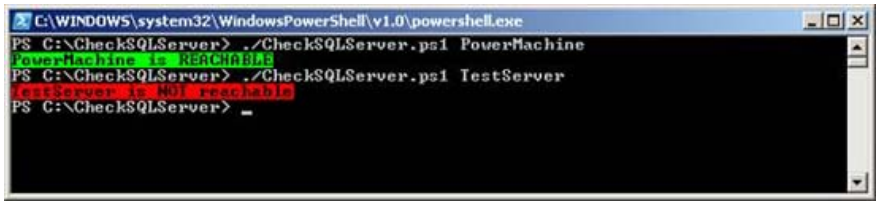

Figure 1.6: Pinging the server

By default, PowerShell scripts cannot be executed on the machine if you use it for the first time. If you encounter the following error message as shown in Figure 1.7, execute the command to enable unrestricted execution of PowerShell script.

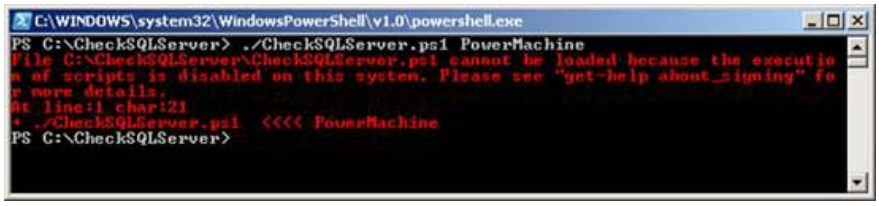

Figure 1.7: Error in executing PowerShell script

```
set-executionpolicy unrestricted
```

Note: You do not have to execute the above command multiple times but only execute it once.
You can check the execution policy of the current PowerShell configuration by executing the following command (shown in Figure 1.8).

```
get-executionpolicy
```

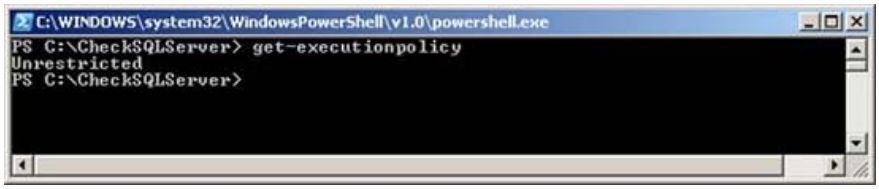

Figure 1.8: Enforcement policy

Conclude

This is the first part of this series. This first part has shown you how to create a PowerShell script to ping a server. It also introduces how to use the PowerShell function source and how to call the function. In the later part of this series, we will continue to dive deeper into other Windows PowerShell features in checking the status of SQL Server.