How to deploy SQL Server via PowerShell DSC

Today, TipsMake will guide you on how to deploy SQL Server through PowerShell DSC, enabling you to use the new PowerShell feature to manage your infrastructure on-premises and in the cloud efficiently and easily.

DSC is integrated into PowerShell 4.0 and is part of **the Windows Management Framework** . PowerShell DSC is included in Windows 2012 R2, but is available for Windows 2008 R2 and Windows 2012 users to download and install.

DSC is a declaration. A DSC script doesn't contain logic about how to perform specific installations or uninstallations. Instead, it defines specific server or application settings.

Imagine you've just bought a new apartment, complete with walls, doors, and windows. You don't need to worry about the walls or window placement, as that's the builder's job. All you need to do is describe the apartment you want, and the builder will construct it for you based on that description.

DSC works in a similar way. The specifications are contained in the configuration, and the DSC runtime acts as a builder, ensuring that the resources on the server are set up according to the user's wishes.

DSC is a relatively new tool. Other configuration management tools like Chef and Puppet have been on the market for many years, and these tools manage both Windows and non-Windows systems. However, using these tools requires users to learn another language. DSC scripts utilize newer extensions for PowerShell.

Advantages of DSC

- Standardization : Create scripts containing definitions of different service types within your architecture (such as IIS, database, file server) and then use them for all new deployments. You can be sure that all servers will be set up identically.

- Accelerated Deployment : Quickly and easily apply DSC configuration to servers via the PowerShell runtime.

- Configuration Detection : DSC provides a solution to identify the active server configuration specified in scripts and can report or automatically correct discrepancies.

- Simple: DSC is developed using PowerShell, so users can fine-tune DSC through PowerShell. However, the scripts lack logic and error handling capabilities, but they are easy to read.

- Idempotency: You can apply the same DSC configuration without any problems. And if the configuration is updated, only the different settings change.

Components of DSC

- WinRM (Windows Remote Management): Microsoft's implementations of the standard WS-Management protocol for managing servers using SOAP.

- CIM (Common Information Model): a standard for describing the structure and behavior of managed resources (such as storage, networks, and software components). WMI is implemented from CIM on Windows.

- MOF (Managed Object Format) files: Contain the configuration applied to the target node.

- Resources: These are the building blocks for configuring DSC. DSC includes several built-in resources, such as File and Windows Features, or users can create their own resources.

Function of DSC resources

Each DSC resource includes 3 functions.

- Test-TargetResource: This is the first function called when the DSC configuration is applied. It returns True/False based on whether the source is correct. If true, the DSC runtime does not need to perform any further operations.

- Set-TargetResource: Called when Test-TargetResource returns False. Responsible for ensuring the resource is set according to the specifications stored in the configuration.

- Get-TargetResource: Returns all properties of the resource. Not used in the configuration step but used for reporting.

DSC configuration

The configuration below specifies that a directory named **DSC_Demo** exists in **C:temp** .

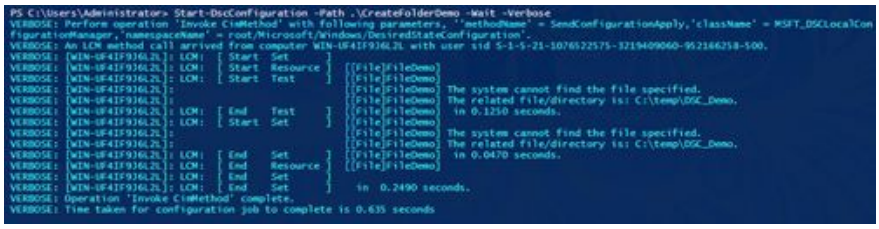
```
Configuration CreateFolderDemo {
    Node localhost {
        File FileDemo {
            DestinationPath="C:\temp\DSC_Demo"
            Type="Directory"
            Ensure="Present"
        }
    }
}

CreateFolderDemo
```

When the PowerShell script above is executed, a MOF file named **localhost.mof** is created by the DSC runtime in the **CreateFolderDemo directory.** If you open the file, you will see a file that looks like this:

```
/*
@TargetNode='localhost'
@GeneratedBy=Administrator
@GenerationDate=04/26/2015 15:00:56
@GenerationHost=WIN-UF4IF9J6L2L
*/

instance of MSFT_FileDirectoryConfiguration as $MSFT_FileDirectoryConfiguration1ref
{
ResourceID = "[File]FileDemo";
 Type = "Directory";
 Ensure = "Present";
 SourceInfo = "C:\\Users\\Administrator\\Desktop\\CreateFolderDemo\\CreateFolderDemo.ps1::3::8::File";
 DestinationPath = "C:\\temp\\DSC_Demo";
 ModuleName = "PSDesiredStateConfiguration";
 ModuleVersion = "1.0";

};

instance of OMI_ConfigurationDocument
{
 Version="1.0.0";
 Author="Administrator";
 GenerationDate="04/26/2015 15:00:57";
 GenerationHost="WIN-UF4IF9J6L2L";
};
```

To apply the configuration **-and** and " **make it so** ", we use the command Start-DscConfiguration:

Start-DscConfiguration -Path .CreateFolderDemo -Wait -Verbose



The directory does not exist before the script is run, so the **Test-TargetResource** function will return False **.** Next, the **Set-TargetResource** function will be called and the directory will be created.

To display idempotency, if the script runs a second time, Test-TargetResource will return True and no further configuration will take place.

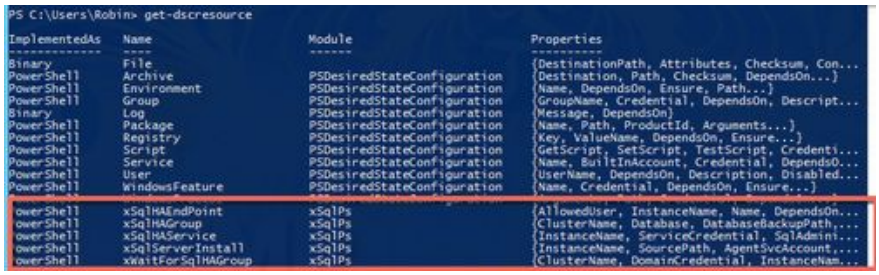
Deploying SQL Server via PowerShell DSC

If you use **the xSqlPs PowerShell module** , you can implement SQL Server through PowerShell DSC.

First, download the module and extract it into the **$env:ProgramFilesWindowsPowerShell folder.**

Download and extract the xSqlPs PowerShell module from here: Download **xSqlPs PowerShell** (insert link)

Open PowerShell and verify the modules are present by running the command **Get-DSCResource:**



Next, create the configuration. In this example, we're installing a SQL 2014 instance named DSCInstance on the local computer:

```
Configuration InstallSqlDemo {

    Import-DscResource -Module XsqlPs

    Node localhost {

        xSqlServerInstall installSqlServer
        {
            InstanceName = "DSCInstance"
            SourcePath = "D:\"
            Features= "SQLEngine,SSMS"
            SqlAdministratorCredential = $credential
        }
    }
}

InstallSqlDemo -credential (Get-Credential -UserName "sa" -Message "Enter password fo
```

When you run the script, it will create a **localhost.mof file in the InstallSQLDemo** directory .

```
PS C:\Users\Administrator> dir .\InstallSqlDemo


    Directory: C:\Users\Administrator\InstallSqlDemo


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         4/26/2015  12:07 PM       1402 localhost.mof


PS C:\Users\Administrator> _
```

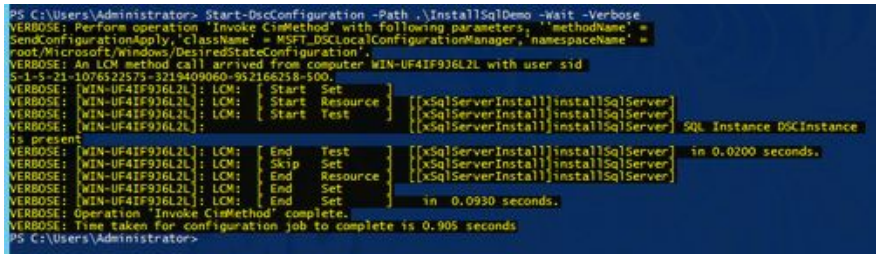
Configure using commands:

Start-DscConfiguration -Path .InstallSqlDemo -Wait -Verbose

The command above will run the SQL Server installation, creating a new DSCInstance database.

```
PS C:\Users\Administrator> get-service mssql$`dscinstance
get-service : Cannot find any service with service name 'mssql$dscinstance'.
At line:1 char:1
+ get-service mssql$`dscinstance
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (mssql$dscinstance:String) [Get-Service], ServiceCommandException
    + FullyQualifiedErrorId : NoServiceFoundForGivenName,Microsoft.PowerShell.Commands.GetServiceCommand

PS C:\Users\Administrator> Start-DscConfiguration -Path .\InstallSqlDemo -Wait -Verbose
VERBOSE: Perform operation 'Invoke CimMethod' with following parameters, ''methodName' =
SendConfigurationApply,'className' = MSFT_DSCLocalConfigurationManager,'namespaceName' =
root/Microsoft/Windows/DesiredStateConfiguration'.
VERBOSE: An LCM method call arrived from computer WIN-UF4IF9J6L2L with user sid
S-1-5-21-1076522575-3219409060-952166258-500.
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ Start  Set      ]
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ Start  Resource ]    [[xSqlServerInstall]installSqlServer]
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ Start  Test     ]    [[xSqlServerInstall]installSqlServer]
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ End    Test     ]    [[xSqlServerInstall]installSqlServer]    in 0.0000 seconds.
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ Start  Set      ]    [[xSqlServerInstall]installSqlServer]
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ End    Set      ]    [[xSqlServerInstall]installSqlServer]    in 189.1950 seconds.
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ End    Resource ]    [[xSqlServerInstall]installSqlServer]
VERBOSE: [WIN-UF4IF9J6L2L]:                                [] A reboot is required to progress further. Please reboot the
system.
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ End    Set      ]
VERBOSE: [WIN-UF4IF9J6L2L]: LCM:   [ End    Set      ]    in  189.5350 seconds.
VERBOSE: Operation 'Invoke CimMethod' complete.
VERBOSE: Time taken for configuration job to complete is 190.34 seconds
PS C:\Users\Administrator> get-service mssql$`dscinstance

Status   Name               DisplayName
------   ----               -----------
Running  mssql$dscinstance  SQL Server (DSCINSTANCE)


PS C:\Users\Administrator> _
```

Similar to the FileDemo above, if you try running Start-DscConfiguration a second time, it will attempt to complete without any errors:



Note : xSqlServerResource is designed for SQL 2012 and other versions, and will report an installation failure error even after the user has completed the installation. This is because the resource searches for the SQL Setup log file in directory 110, but SQL Server versions store the log file in a different location.

The quickest way to fix this is to edit " **C:Program FilesWindowsPowerShellModulesxSqlPsDSCResourcesMSFT_xSqlServerInstallMSFT_xSqlServerInstall.psm1"** and change the directory on **line 154** to one of the lines related to your SQL version (100 for Server 2008 and 120 for 2014).

DSC is a great tool for easier server management and deployment in the future. However, at this time, DSC is still very new and only supports basic operations. The SQL Server module does not display many of the command-line utilities available in SQL Setup.

In summary, this article from TipsMake has guided you on how to deploy SQL Server using PowerShell DSC. Additionally, you can use DSC to deploy Service Packs or Cumulative Updates to a SQL Server instance. Then, if a new Service Pack is released, the DBA will update some configuration files and redeploy, and the SQL Server resource will detect that only one patch needs to be installed, not the entire deployment.

Before making any significant changes to SQL Server, to avoid risks, you should refer to the SQL Server backup and restore guide to understand how to recover SQL Server data.

Taimienphi.vn has also provided detailed instructions on how to install SQL Server 2019 in the article "How to **install SQL Server 2019"** . If you need to install it, please refer to that guide for easier operation.

You finished reading the article "**How to deploy SQL Server via PowerShell DSC**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
