WorkshopPLUS Windows PowerShell Desired State Configuration

Module 3: Pull

Student Lab Manual

Version 1.2

Conditions and Terms of Use

Microsoft Confidential - For Internal Use Only

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

Contents

Lab 3: Pull

Introduction

In this lab, we will discover how to host our DSC configurations and DSC resources on a dedicated DSC pull server. The **DSC-Service** is a built-in Windows feature on Windows Server 2012 R2, which we will enable and configure. We will initially use DSC in push mode to configure the pull server.

Once the pull service is configured, we will host our configuration on the pull server and then configure the target node **LocalConfigurationManager** (LCM) to pull the configuration.

There are three pull server configurations: HTTP, HTTPS and SMB. This lab will focus on the HTTP configuration. HTTPS will be covered in a later module.

Objectives

After completing this lab, you will be able to:

- Enable and configure a DSC pull server
- Configure a target node LCM in pull mode
- Host DSC configurations on the pull server

Prerequisites

If you have not completed the Module 1 lab, then you will need to install WMF 5.0 by running the following two scripts in the order listed:

- C:\PShell\Labs\Lab00\Lab_00_01_01_Stage_Resources.ps1
- C:\PShell\Labs\Lab00\Lab_00_01_02_Install_WMF_5.ps1

Start all VMs provided for the workshop labs.

Logon to the **PULL** server as **Administrator** with the password **PowerShell5!**

Estimated time to complete this lab

90 minutes

For more information

Setting up a DSC web pull server

https://msdn.microsoft.com/en-us/PowerShell/DSC/pullServer

Scenario

Your team is responsible for the infrastructure and operations for a new application that is being deployed within your organization. The application will run in an environment hosted on virtual machines running Windows Server 2012 R2.

Once the virtual machines have been deployed, you need to ensure the configurations of those machines meet the strict security guidelines that have been outlined from the compliance team. As the application updates are being deployed regularly, you need to maintain the settings across the servers several times per week or even several times per day.

In order to perform this task in a scalable and manageable way, you choose to configure a DSC **pull server** so that you can easily host the configurations with the core server settings. The target nodes will periodically check for new settings and apply them.

NOTE: Use the Windows PowerShell Integrated Scripting Environment (ISE) for typing and editing the PowerShell scripts in this course. The automatic Intellisense will be very helpful in the learning process. If you need to reactivate Intellisense, use the **CTRL SPACE** keyboard shortcut.

Each lab has its own files. For example, the lab exercises for Module 3 have a folder at **C:\PShell\Labs\Lab03** on the **PULL** virtual machine. Use these provided files to avoid typing long sections of code.

We recommend that you connect to the virtual machines for these labs through Remote Desktop rather than connecting through the Hyper-V console. When you connect with Remote Desktop, you can copy and paste from one virtual machine to another.

Exercise 3.1: Configure the Pull Server

Objectives

In this exercise, you will:

- Use DSC to configure the node **PULL** with the **DSC-Service**
- Review the settings that are configured on the IIS web server

Scenario

Within the DSC platform a server that can host configurations is known as the pull server. The **DSC-Service** is a built-in feature on Windows Server 2012 R2 and later operating systems. This service is built on top of the **Web-Server** role.

Simply installing the Windows feature does not enable the service. Then it must be configured. The best way to perform the configuration is using DSC.

There is an external DSC resource module named **xPSDesiredStateConfiguration** that contains the **xDSCWebService** resource. It contains all of the appropriate settings to configure the pull server.

Task 3.1.1: Create the configuration for the pull server

1. Open the PowerShell Integrated Scripting (ISE) Environment and begin a new script by pressing **CTRL+N** or from the menu: **File** > **New** or the NEW icon on the taskbar. It is recommended that you type all of the commands below into that new script, each on a new line.

> **NOTE:** You can highlight a section of code and run it with the **F8** key OR execute only the current line simply by pressing the **F8** key (even without highlighting anything, so long as the cursor is on the line).

2. Run the following command to download the latest version of the DSC resource for configuring the pull server. This may take a few minutes.

```
Install-Module -Name xPSDesiredStateConfiguration -Force
```

> **NOTE:** In a later module we will explain more about this download process.

3. If you are prompted to install the **NuGet provider**, select **Yes** to confirm.
4. If you are prompted to install from an **untrusted repository**, select **Yes** to confirm.
5. Run the following command to view the resources in the **xPSDesiredStateConfiguration** module. Look for the **xDSCWebService** resource.

```
Get-DscResource -Module xPSDesiredStateConfiguration
```

6. Run the following command to view the syntax for that resource:

```
Get-DscResource -Name xDSCWebService –Syntax
```

7. The output should look similar to below. These are all of the settings available to configure the Windows server feature called **DSC-Service**.

```
xDSCWebService [String] #ResourceName
{
    CertificateThumbPrint = [string]
    EndpointName = [string]
    UseSecurityBestPractices = [bool]
    [AcceptSelfSignedCertificates = [bool]]
    [ConfigurationPath = [string]]
    [DatabasePath = [string]]
    [DependsOn = [string[]]]
    [DisableSecurityBestPractices = [string[]]{ SecureTLSProtocols }]
    [Ensure = [string]{ Absent | Present }]
    [ModulePath = [string]]
    [PhysicalPath = [string]]
    [Port = [UInt32]]
    [PsDscRunAsCredential = [PSCredential]]
    [RegistrationKeyPath = [string]]
    [State = [string]{ Started | Stopped }]
}
```

8. Many PowerShell resource modules contain a folder of example usage code. Explore the folder where this module was installed:

```
Set-Location (Get-Module -Name xPSDesiredStateConfiguration -List).ModuleBase

dir

cd .\Examples

dir
```

9. Open and review the first sample for **xDscWebService**:

```
psEdit .\Sample_xDscWebServiceRegistration.ps1
```

10. Copy all of the sample configuration **Sample_xDscWebServiceRegistration** from the file that you opened.

11. Create a new PowerShell script tab in the ISE and paste this code for editing.

12. First we can clean up some of the configuration that we will not use today. Remove the entire **Param** block, including the **param (…)**

```
    param
    (
        [string[]]$NodeName = 'localhost',

        [ValidateNotNullOrEmpty()]
        [string] $certificateThumbPrint
    )
```

13. Comment out or delete the entire **RegistrationKeyFile** resource settings

```
#File RegistrationKeyFile
#{
```

```
#    Ensure          = "Present"
#    Type            = 'File'
#      DestinationPath="$env:ProgramFiles\WindowsPowerShell\DscService\RegistrationKeys.txt"
#      Contents        = $RegistrationKey
#}
```

14. Under the PSDSCPullServer resource comment out or delete the **RegistrationKeyPath** and **AcceptSelfSignedSertificates** setting.

```
#RegistrationKeyPath     = "$env:PROGRAMFILES\WindowsPowerShell\DscService"
#AcceptSelfSignedCertificates = $true
```

15. The **WindowsFeature** resource comes from the **PSDesiredStateConfiguration** module, so we must ensure the following line is present above the node block:

```
Import-DSCResource –ModuleName PSDesiredStateConfiguration
```

16. We are going to configure a secure HTTPS pull server in the next module. For now replace the **CertificateThumbprint** with the following value

```
CertificateThumbprint = "AllowUnencryptedTraffic"
```

17. Change the **Node** name to **localhost**:

```
Node localhost
```

18. Under the xDSCResource add the following property setting:

```
UseSecurityBestPractices = $false
```

19. Rename the configuration from Sample_xDscWebServiceRegistration to **PullServerHTTP**.
20. Remove any extra comments at the top if any.
21. Your final pull server configuration should appear as follows (see below):

```
configuration PullServerHTTP
{

    Import-DSCResource -ModuleName xPSDesiredStateConfiguration
    Import-DSCResource -ModuleName PSDesiredStateConfiguration

    Node localhost
    {
        WindowsFeature DSCServiceFeature
        {
            Ensure = "Present"
            Name   = "DSC-Service"
        }

        xDscWebService PSDSCPullServer
        {
            Ensure                  = "Present"
            EndpointName            = "PSDSCPullServer"
            Port                    = 8080
            PhysicalPath            = "$env:SystemDrive\inetpub\PSDSCPullServer"
            CertificateThumbPrint   = "AllowUnencryptedTraffic"
            ModulePath              = "$env:PROGRAMFILES\WindowsPowerShell\DscService\Modules"
            ConfigurationPath       = "$env:PROGRAMFILES\WindowsPowerShell\DscService\Configuration"
            State                   = "Started"
            UseSecurityBestPractices = $false
            DependsOn               = "[WindowsFeature]DSCServiceFeature"
        }
    }
}
```

Question A: How many resources did it take to write the configuration for a pull server?

Question B: Why was the **WindowsFeature** resource required?

Question C: Why was the **xDSCWebService** resource required?

Task 3.1.2: Apply the pull server configuration

1. Highlight and execute the configuration **PullServerHTTP** to load it into memory.
2. Set your location as follows:

```
Set-Location C:\PShell\Labs\Lab03
```

3. Run the configuration as follows:

```
PullServerHTTP
```

4. Apply the configuration. This will take a couple minutes.

```
Start-DscConfiguration -Path .\PullServerHTTP -Verbose –Wait
```

Question D: Read the configuration output. What is the name of the endpoint that was configured?

---

5. Navigate to the URL of the pull server. If you see a prompt that Internet Explorer has blocked the page, simple click **Close**.

```
Start-Process http://PULL:8080/PSDSCPullServer.svc
```

Question E: What is the format of the page output? (See the first line for a clue.)

Question F: Look back at the configuration script used to build the pull server. What is the directory where you will need to place the DSC configurations to be pulled by nodes?

> **NOTE:** Remember this file system location for the next task. You will host the configurations here for the target nodes.

6. Run the following command to open the Internet Information Services (IIS) Manager:

```
& $ENV:Windir\System32\Inetsrv\InetMgr.exe
```

> **NOTE:** If you get the prompt “Get started with the Microsoft Web Platform”, select **Do not show this message**. Then select **No.**

7. In the tree view on the left, expand **PULL**, then expand **Sites**. Click on the **PSDSCPullServer** web site.

8. Right click on the **PSDSCPullServer** web site and select **Edit Bindings**.

   Question G: What is the port that was used? How did this happen?

   _____

9. Do not close the **InetMgr.exe**. We will use it in the next task.

Exercise 3.2: Prepare and Host Node Configurations

Objectives

In this exercise, you will:

- Create a node configuration
- Stage the node configuration on the pull server
- Generate a checksum file for a configuration

Scenario

The target node is running Windows Server Core, which does not have a graphical interface for web administration. By default, IIS web servers do not allow remote administration for security reasons. You need to create a configuration that will both configure the web service and allow remote administration. Once created, the configuration must be hosted for the node to pull.

Task 3.2.1: Create the node configuration

1. If not already open from the previous task, type in the following to navigate to the **IIS Manager**:

```
& $ENV:Windir\System32\Inetsrv\InetMgr.exe
```

2. Select the **File** Menu and select "**Connect to a Server...**"
3. Type in **MS1**, then Click **Next**.
4. Type the user name: **contoso\administrator**
5. Type the password: **PowerShell5!**
6. Click **Next**.

Question A: What happens? Were you able to connect?

_____

_____

> **NOTE:** The servers in our labs are very near to the default settings. We have not configured a web server or remote web management access on **MS1**. We expect the error.

7. Click **OK** and **Cancel**. Open the **Windows PowerShell ISE**.
8. Open the script file with our **BaseConfig** configuration. Type in the following (use TAB completion where possible):

```
psEdit C:\PShell\Labs\Lab03\Lab_03_02_01_Install_BaseConfig.ps1
```

9. This script holds our base server configuration. Review the resources. Note the sequencing with the **DependsOn** property.
10. Press **F5** to run the configuration and create the **.\BaseConfig\ms1.mof** file.

Task 3.2.2: Host the node configuration on the pull server

1. Begin a new script by pressing **CTRL+N** or from the menu **File** > **New** or the NEW icon on the taskbar. It is recommended that you type all of the commands below into the new script, each on a new line.
2. In a pull server scenario we must configure the node **LCM** with a **ConfigurationID**. This is a Globally Unique Identifier (GUID) that matches the file name of the configuration hosted on the pull server. This is how the node knows which configuration to pull.

> **NOTE:** It is possible for several servers to be configured with the same **ConfigurationID**, in which case they would share the same configurations.

3. Run the following command to create a GUID for our configuration:

```
$ms1 = (New-Guid).Guid
$ms1
```

4. Save the GUID to disk for use in **Exercise 3.3** later:

```
Set-Content -Path C:\PShell\Labs\Lab03\MS1guid.txt -Value $ms1 -PassThru
```

5. We must now copy our configuration MOF file to the pull server configuration directory and give it a new name in the format **GUID.mof**. Run the following command to correctly place and name the configuration that we previously created.

```
$destination =
"$env:PROGRAMFILES\WindowsPowerShell\DscService\Configuration\$ms1.mof"

Copy-Item -Path .\BaseConfig\ms1.mof -Destination $destination –PassThru
```

6. Each time we publish or modify a configuration we must generate a checksum file. This is how the node LCM tests to see if there is a new or updated configuration on the pull server. Run the following command:

```
New-DscChecksum -Path $destination -Force –Verbose
```

Question B: What is the extension of the file that was created?

_____

> **NOTE:** It is a good practice to always use the **-Force** switch with **New-DscChecksum**. By default the cmdlet does not overwrite old checksum files.

Exercise 3.3: Configure Target Node for Pull Mode

Objectives

In this exercise, you will:

- Configure a node LCM for pull mode
- Force a node to update
- Use the DSC cmdlets to determine the status of the LCM
- Update the configuration hosted on the pull server

Scenario

When nodes should periodically check for new configurations, using DSC in pull mode scales the best. This requires setting the target node LCM with the URL of the pull server and the node's configuration ID. By default, a node will check for an updated configuration every 30 minutes. During testing you can override this behavior and trigger the node to pull a configuration on demand.

Task 3.3.1: Configure the LCM on the target node

1. Begin a new script by pressing **CTRL+N** or from the menu **File** > **New** or the NEW icon on the taskbar. It is recommended that you type all of the commands below into a new script, each on a new line.
2. Type the following meta configuration. Use **TAB** complete for keywords, properties, and values where possible. This is an enhancement introduce in PowerShell v5.

> **TIP:** Often PowerShell syntax uses opening and closing punctuation (parentheses, quotes, braces, etc.). Type both pieces of punctuation at the same time, and then arrow back inside to fill in the code. This helps reduce syntax errors when scripting.

> **NOTE:** If you are short on time, you may use the supplied solution file: C:\PShell\Labs\Lab03\Lab_03_03_01_Configure_LCM_Pull_Solution.ps1. The intent is that you become familiar with the new Intellisense features of PowerShell v5 ISE and meta configurations.

```
[DscLocalConfigurationManager()]
Configuration MS1Meta
{
Param (
    [String]$Guid = (Get-Content -Path C:\PShell\Labs\Lab03\MS1guid.txt)
)
    Node MS1
    {
        Settings
        {
```

```
            ConfigurationMode = "ApplyAndAutoCorrect"
            ConfigurationID   = $Guid
            RefreshMode       = "Pull"
        }

        ConfigurationRepositoryWeb PULL
        {
            ServerURL = "http://PULL:8080/PSDSCPullServer.svc"
            AllowUnsecureConnection = $True
        }
    }
}
```

3. Select and execute (**F8**) the above code to load it into memory.

4. Ensure you are working from the correct directory, run the following:

```
Set-Location C:\PShell\Labs\Lab03
```

5. Run the meta configuration:

```
MS1Meta
```

Question C: What is the output? How is this different from the previous configurations?

6. Check the current LCM settings on node **MS1**:

```
Get-DscLocalConfigurationManager -CimSession ms1
```

Question D: What is the current **RefreshMode**?

7. Apply the meta configuration to the target node:

```
Set-DscLocalConfigurationManager -Path .\MS1Meta –Verbose
```

8. Check the new LCM settings on that node:

```
Get-DscLocalConfigurationManager -CimSession ms1
```

Question E: What is the **RefreshMode** now?

Question F: What is the current **LCMState**?

Question G: How often does the node check the pull server for a new configuration?

HINT: The **RefreshFrequencyMins** property determines how often the LCM will check for a new checksum on the pull server. When configuring this setting be aware that the minimum value is 30 minutes.

9. Check for MOF files on the server **MS1** with the following command:

```
Invoke-Command -ComputerName ms1 -ScriptBlock {
        Get-ChildItem -Path C:\Windows\System32\Configuration }
```

Question H: Which MOF file contains the LCM meta configuration that you just now set?

---

10. The LCM will eventually poll the pull server for a configuration. However, we want to force it to check immediately. Run the following command to trigger the pull:

```
Update-DscConfiguration -CimSession ms1 -Wait –Verbose
```

11. View the web server to see if it is running

```
Start-Process http://ms1/
```

12. If not already open from the previous task, open the **IIS Manager**:

```
& $ENV:Windir\System32\Inetsrv\InetMgr.exe
```

13. Open the **File** menu and select "**Connect to a Server...**"
14. Type **MS1** and click **Next**.
15. Type the user name: **Contoso\Administrator**
16. Type the password: **PowerShell5!**
17. Click **Next**.
18. Ignore the certificate warning. Click **Connect**.
19. Click **Finish**.
20. Notice that you are now able to connect and remotely administer the **Windows Server Core** web server.
21. Open the **ServerManager** console using the icon or the command below:

```
ServerManager.exe
```

22. Navigate to the node on the left named **IIS**.
23. Click on server **MS1** under the **SERVERS** section.
24. Scroll down to the **SERVICES** section.

    Question I: What is the name of the service that we configured to allow the remote connection via the **IIS Manager**?

25. Right click on that service and select **Stop Services**.
26. Go back to **IIS Manager**. Right click on the **ms1** node. Click Refresh.

    Question J: What is the outcome?

27. Execute the following command to refresh the configuration on MS1

```
Start-DscConfiguration -CimSession ms1 -Wait -Verbose -UseExisting
```

28. Go back to **IIS Manager**. Right click on the **ms1** node. Click Refresh.

    Question K: What is the outcome?

NOTE: Server **MS1** is running an instance of Windows Server 2012 R2 using the default installation option **Server Core**. The server has no Graphical User Interface (GUI). However, as you have seen, we can still easily manage it remotely using **DSC** and also using remote GUIs such as **Servermanager.exe** or **InetMgr.exe**

Task 3.3.2: Update the hosted configuration

1. The application development team has a newer version of the website that they need to update on **MS1**. Make the following changes to the **BaseConfig** configuration that you have already been using. Under the **Archive WebFiles** resource, update the **Path**:

```
Path        = '\\PULL\PShell\Labs\Lab03\Lab_03_03_02_WebSite3_2.zip'
```

2. Run the configuration **BaseConfig** to create the **.\BaseConfig\MS1.mof** file, as below:

```
BaseConfig
```

NOTE: The following steps should be familiar. We are repeating the same steps as when we initially hosted the configuration on the pull server.

3. We must now copy our configuration to the pull server configuration directory and give it the name in the format GUID.mof. Run the following command to correctly place and name the configuration that we had previously run.

```
$GUID = (Get-Content -Path c:\pshell\labs\lab03\ms1guid.txt)

$destination =
"$env:PROGRAMFILES\WindowsPowerShell\DscService\Configuration\$GUID.mof"

Copy-Item -Path .\BaseConfig\MS1.mof -Destination $destination –PassThru
```

4. Each time you modify and publish a configuration document, you must also ensure that the checksum file is recreated as well. Run the following command:

```
New-DscChecksum -Path $destination -Force –Verbose
```

5. Run the following command to trigger the pull on the **MS1** node:

```
Update-DscConfiguration -CimSession ms1 -Wait –Verbose
```

Question L: Scroll up through the verbose output. Many resource say **Skip Set**. Which one resource says **Start Set**? Why?

6. View the web server to see if it displays the update:

```
Start-Process http://ms1/
```

Question M: Was the update successful?

NOTE: The update should succeed. If you do not see the new page, then try clearing the browser cache or refreshing using **Ctrl+F5**.