Lab 8b – Working from Command Line/PowerShell

Part 1 - Configuring Windows Server 2019 Data Center (without GUI)

You will work on this part of the lab in **Server 3.** Please note that Windows Server 2019 Data Center (Without GUI) is installed in Server 3 VM.

Use the information from the table in the PreLab Chart to perform the steps below, substituting the appropriate information. You will need the computer name and the IP address for Server3.

Boot Server 3 and login. You will only see a Command Prompt on Server 3. Perform the following steps to complete the Initial Configuration of your server

1.0 Change the Computer's Name

Your computer will have a randomly generated name. Change that name with the following command:

netdom renamecomputer %computername% /newname:*newComputerName*

Where *newComputerName* is the computer name for Server3 from the chart in the Prelab. It should be **SenecaID-S3** (please replace **SenecaID** with your seneca userid)

Now, you need to reboot the computer to see the computer name change in effect. You can reboot with the following command:

shutdown /r /f /t 0

2.0 Set Static IP Address, Subnet Mask, and Default Gateway

To set a static IP address and configure the default gateway for your server, use this command, substituting the *interfaceName*, *ipAddress*, *subnetMask* and *defaultGateway* with the appropriate values from your PreLab chart

netsh interface ipv4 set address *interfaceName* static *ipAddress subnetMask defaultGateway*

For most servers the value for *interfaceName* will be set to "Ethernet." You can verify that name first with this command:

netsh interface show interface

**Note the defaultGateway value may be set to 0.0.0.0 because we do not have a router.

To prove you have completed this lab:

Type in ipconfig /all to display all of your settings. You can take a screen shot of the ipconfig output and save to a file called Lab8Part1.jpg. Upload the file into Blackboard.

Part 2 - Introduction to PowerShell

Windows PowerShell is an updated, more powerful version of the command line – it is like a combination of a Windows Command Line and scripting environment rolled into one. It can help with repetitive tasks; processes acting on many files at once; automating and scheduling tasks; and configuring Windows components and services. While PowerShell is a command line interface it is more powerful and flexible than that. It supports complex decision making, connecting to a wide variety of data sources, and can even be used when building graphical user interfaces.

PowerShell is now an essential skill for IT and server administrators. Windows IT professionals who only use a GUI interface for server and network management will need to know how to use PowerShell. It is constantly being developed and there are hundreds of commands.

PowerShell uses commands called CMDLETS (pronounced command-lets). They are very english-like commands and use a VERB-NOUN format.

It is easy for Linux and Windows Command Line users to get a quick start with PowerShell because of built-in aliases. However, it is best to learn PowerShell's proper syntax quickly in order to make the most of the environment.

1.0. Customize a shortcut to point to a home directory (Server 1)

In this exercise we configure the PowerShell Console to open in a directory called "MyScripts." To open Powershell, type Powershell from the Run command, or click the Powershell icon on the task bar. Do this lab on **Server1**.

1. Create the directory – **C:\MyScripts** using the following command:
   New-Item C:\MyScripts -type directory
2. Also, create 3 files in the **MyScripts** directory (***file1.txt, file2.txt*** and ***file3.txt***) using the following syntax:
   New-Item C:\MyScripts\file1.txt -type file
3. Create a PowerShell shortcut on your desktop.
   (To create a shortcut for Powershell, right-click on the *Powershell* icon on your taskbar, then right-click *Windows Powershell* and select *Properties*. Copy the text that points to where Powershell can be found on the hard drive. Then right-click on the desktop, select *New* and *Shortcut*. Paste the Powershell path into the shortcut field.)
4. Right-Click on the shortcut and choose **Properties.**
5. Under the Shortcut tab locate the **Start in:** text box. Type in the path to the folder created in step 1. ***C:\MyScripts***
6. Click OK.
7. Launch PowerShell from the desktop shortcut. PowerShell starts in the directory path we've chosen.

8. Customize your PowerShell window by clicking the PS symbol in the upper left corner and selecting Properties from the menu. Cusomize your Font, Layout and Colours to suit you ensure you can see the entire PS window without scrollbars. (You must make some change.) Click OK.

*****Take a screen shot of the PowerShell console with your custom directory. Name this file *PS-custom-dir.jpg***

(To take a screen shot, press *Shift+PrintScrn* keys. This captures the entire screen. (Or *Alt- PrintScrn* which captures the active window.) Then open the **Paint** program and press *Ctrl+V* to paste the screen capture into paint. Save the file with the appropriate name. You can also use the Snipping Tool from Windows 10.)

2.0 Using PowerShell cmdlets

PowerShell commands have been standardized using a “verb-noun” naming convention know as a cmdlet. This standard simplifies the learning curve and provides a description of what the cmdlet does.

1. To see a list of cmdlets available in PowerShell type the following cmdlet from PowerShell:

   get-command

   A list of PowerShell commands will be displayed. Look in the “Name” column to see the “verb-noun” naming convention.

2. To list all the commands that use a specific verb, such as **Get,** use the following command:

   get-command -Verb Get

3. List a few of the Get commands here:

4. To find out how many cmdlets are available in the default installation of PowerShell on your server, use the following command:

   Get-Module -ListAvailable | Import-Module ; gcm -co cmdlet | measure

5. How many cmdlets are available on your server? ____________________

2.1 Getting help

In learning new technologies, it is important to find information quickly and easily. Get-Help cmdlet has been designed for that purpose; this will be the most utilized cmdlet until you become more proficient.

1. To get information about the Get-Help cmdlet, including description, syntax, and remarks, type:

   Get-Help

   The **Update-Help** cmdlet downloads the newest help files for PowerShell modules and installs them on your computer. You don't need to restart PowerShell to make the change effective. You can use the Get-Help cmdlet to view the new help files immediately.

2. Type the following command to update the help files on your computer:

   Update-Help

3. To get information about a specific cmdlet, in this case Get-ChildItem, type:

   Get-Help Get-ChildItem

4. You can use Get-Help with an alias. Type:

   Get-Help dir

   Notice that you are actually getting help for Get-ChildItem.

5. To get more detailed help, type:

   Get-Help dir -Full

6. To get some examples of the command, type:

   Get-Help dir -Examples

7. To get the most up to date, easily readable help for a command, use the -Online option:

   Get-Help dir -Online

8. In the next section, you will be using some cmdlets to manage services on your server. Use Get-Help to learn about the following commands before moving on to section 3.0:

   Get-Help Get-Service

   Get- Help Start-Service

   Get-Help Stop-Service

**Take a screen shot of the PowerShell console showing the first part of the output from the Get-Help Stop-Service command. Name this file *PS-gethelp.jpg*

3.0 Starting and Stopping a Service Using cmdlets

The cmdlets associated with the noun **Service** are as follows: Get-Service, New-Service, Restart-Service, Set-Service, Start-Service, Stop-Service, and Suspend-Service.

1. To find out what services are on the computer? Type the following cmdlet:

   get-service

   You should now have a list of services on the computer as well as their status. The Services MMC (Microsoft Management Console) in the Graphical User Interface would display the same information.

3.1 Stopping and Starting a Service.

For this exercise we will be using both the **Stop-Service** and **Start-Service** cmdlets to change the status of the "Print Spooler" service. The current status of the Print Spooler service is "Started" and the "Status Type" is "Automatic." For this exercise it is important that you choose a service that has a status type of either automatic or manual. If you choose a service with a status type of disabled, this exercise will not work.

1. To begin show the status of the Spooler Service, type:

   Get-Service -name Spooler

   You should see that the status of the the spooler service is "running".

2. Then Stop it using the Stop-Service cmdlet:

   Stop-Service -name Spooler

   Check the status of the Spooler service again. It should display "stopped".

   Get-Service -name Spooler

3. To start the Spool service again, using the following command:

   Start-Service -name Spooler

   Check the status of the service again, using:

   Get-Service -name Spooler

   Check the GUI, the status of Spooler should be "Started"

**Take a screen shot of the PowerShell console showing your stopped and started service. Name this file *PS-*

service.jpg

4.0 Aliases

Aliases have been built-in to PowerShell so that Windows Command Line and Linux administrators can start using it quickly. Common commands have been created as aliases that run PowerShell commands in the background.

1. Type the following cmdlet to get a list of PowerShell Aliases available on your server:

   Get-Alias

2. You could look through the list to see how many aliases there are for the **Get-ChildItem** cmdlet, or run the following command to see only alias for **Get-ChildItem**:

   Get-Alias -Definition Get-ChildItem

   Record the aliases here:

3. Use each of the aliases and the **Get-ChildItem** command and compare the output.

   What do you notice?

You should see the exact same output 4 times because it is really the Get-ChildItem command running each time. Note: (If you didn't create the 3 files in My Scripts, you will not see any output as in the examples below because your directory is empty. If you would like to see some files listed, create the 3 sample files before executing the commands.)

4.1 User-Defined PowerShell Aliases

Users can create their own commands by defining aliases. User-defined aliases only last while the PowerShell session is active. As soon as you quit the session, your alias is deleted. You can define user aliases in a PowerShell profile to make it available for every session.

1. The syntax for creating a User-Defined alias is very simple. To create an alias called "gs" for the Get-Service cmdlet, type:

   Set-Alias gs Get-Service

2. Test your new alias by typing ***gs*** at the prompt. You should see the same output as you do when you ran the **Get-Service** command earlier.

3. To prove that your alias is not saved after you close your PowerShell session, close and re- launch Powershell

from your Shortcut. Attempt to use the “gs” alias again and you will receive the following error: ***“The term ‘gs’ is not recognized as a cmdlet, function, operable program, or script file. Verify the term and try again.”***

4. Create the alias again:

   Set-Alias gs Get-Service

5. Verify that the alias works by running the gs command. Then type the following command:

   Get-Alias -Name gs

**Take a screen shot of the PowerShell console displaying your gs alias. Name this file *PS-alias.jpg*

5.0 Creating a Profile

You can create a PowerShell profile to customize your environment and to add session- specific elements to every PowerShell session that you start.

A PowerShell profile is a script that runs when PowerShell starts. You can use the profile as a logon script to customize the environment. You can add commands, aliases, functions, variables, snap-ins, modules, and PowerShell drives.

1. $PROFILE is a variable that stores the paths to the PowerShell profiles in the current session. At the prompt, type ***$Profile*** to see the path to your current profile.
2. Try to edit the profile, by typing:

   Notepad $Profile

   You should get a message “System Cannot Find the File Specified”. This is because there is no current profile created.

3. To create the profile file, type the following command:

   New-Item -Itemtype File -Path $Profile -Force

4. Now edit the Profile by typing:

   Notepad $Profile

   Notepad will open an empty file.

5. Type your alias command from the previous section into this file:

Set-Alias gs Get-Service

Save and close the file.

6. Close and open PowerShell and then type **gs** at the prompt. If your alias worked, then your Profile is set up correctly.

**Take a screen shot of the PowerShell console displaying your gs alias. Name this file *PS-profile.jpg*

To prove you have completed this lab:

- Create a Microsoft Word document (or use Google docs), with a name of *YourSenecaID*-Lab8.docx.
- Paste each screen shot from the lab into the MSWord document, and label them clearly. (PS-custom-dir.jpg, PS-help.jpg, PS-service.jpg, PS-alias.jpg and PS-profile.jpg)
- Save the document as a PDF file (or Print as PDF) using the same name as the document file, and upload it to MySeneca.