Module 2: Working with pipeline

1. Create two similar, but different, text files. Try comparing them using Diff. To do so, run something like this: Diff -reference (Get-Content File1.txt) -difference (Get-Content File2.txt). If the files have only one line of text that’s different, the command should work.

Usually to compare two set of objects we use the **Compare-Object** cmdlet.

There are two alias to the Compare-Object cmdlet. They are: “**compare**” and “**diff**”

The two objects which we compare one would be the reference object and one would be the difference object.

PS C:\users\ve40013372> "My name is V.Sai Ganesh" | Out-file test1.txt

PS C:\users\ve40013372> "I am interested in learning PowerShell Scripting" | Out-File test2.txt

PS C:\users\ve40013372> $a=get-content .\test1.txt

PS C:\users\ve40013372> $b=get-content .\test2.txt

PS C:\users\ve40013372> compare-object -ReferenceObject $a -DifferenceObject $b

InputObject SideIndicator

----------- -------------

I am interested in learning PowerShell Scripting =>

My name is V.Sai Ganesh <=

1. What happens if you run Get-Service | Export-CSV services.csv | Out-File from the console? Why does that happen?  
     
   If you don’t specify a file name with Out-File you’ll get an error.

But even if you do Out-File won’t really do anything because the file is actually created by Export-CSV.

We observe that there will be no data in that file.

1. Apart from getting one or more services and piping them to Stop-Service, what other means does Stop-Service provide for you to specify the service or services you want to stop? Is it possible to stop a service without using Get-Service at all?  
     
   We will be able to stop one or more services individually by using the parameter values for the parameter

“-Name“. Example to stop a specific service like Print Spooler. We can use the below cmdlet:

“**Stop-Service –Name Spooler”**

1. What if you wanted to create a pipe-delimited file instead of a comma-separated file? You would still use the Export-CSV command, but what parameters would you specify?  
     
   We can use the below cmdlet to create a pipe-delimited file instead of a comma-separated file still using the export-csv command by using the parameter “**–Delimiter**” the below cmdlet:

**Get-Service | Export-CSV services.csv –Delimiter “|”**

1. Is there a way to eliminate the # comment line from the top of an exported CSV file? That line normally contains type information, but what if you wanted to omit that from a particular file?

We have a parameter named **“–NoTypeInformation”** for omitting the #comment line in the output of an exported CSV file.

Ex: **Get-service | export-csv services.csv –NoTypeInformation**

1. Export-CliXML and Export-CSV both modify the system, because they can create and overwrite files. What parameter would prevent them from overwriting an existing file? What parameter would ask you if you were sure before proceeding to write the output file?  
     
   In order to prevent the over-writing of any of the file using the Export-Clixml/CSV we can use the parameter –NoClobber. By using this parameter we get an error that a file already exists.

Ex: **get-service | Export-Csv services.csv –noclobber**

The “-confirm” parameter is used to make us sure before proceeding to write the output file

Ex: **get-service | Export-Csv services.csv -confirm**

1. Windows maintains several regional settings, which include a default list separator. On U.S. systems, that separator is a comma. How can you tell Export-CSV to use the system’s default separator, rather than a comma?  
     
   We can tell the Export-CSV to use the system default separator rather than a comma using the parameter “–UseCulture”.

**Get-service | Export-Csv services.csv –UseCulture**

1. Using the cmdlet from task #2 and Select-Object, display only the current day of the week in a table like this (caution: The output will right-align, so make sure your PowerShell window doesn’t have a horizontal scroll bar):  
     
   In order to display only the current day of the week in a table, using the **select-object** we can use the following cmdlet:

**Get-Date | select-object DayofWeek**

**PS C:\users\ve40013372> Get-Date | select DayOfWeek**

**DayOfWeek**

**---------**

**Friday**

1. Identify a cmdlet that will display information about installed hotfixes.  
     
   We can get the info of the installed hotfixes using the command “**Get-HotFix”**
2. Using the cmdlet from task #5, display a list of installed hotfixes. Sort the list by the installation date, and display only the installation date, the user who installed the hotfix, and the hotfix ID.  
     
   To display the list of installed hotfixes and only the installation date, the user and the hotfix ID we can use the below cmdlet:

“**Get-HotFix | Sort InstalledOn | Select InstalledOn,InstalledBy,HotFixID”**

1. Repeat task #6, but this time sort the results by the hotfix description, and include the description, the hotfix ID, and the installation date. Put the results into an HTML file.  
     
   Get-HotFix | Sort Description | Select Description,InstalledOn,InstalledBy,HotFixID | ConvertTo-Html -Title "HotFix Report" | Out-File HotFixReport.html

PS C:\users\ve40013372> get-hotfix |sort Description |select Description, hotfixID, InstalledOn, Installedby |ConvertTo-html -Title Report | out-file 1.html

PS C:\users\ve40013372> .\1.html

PS C:\users\ve40013372>

1. Display a list of the 50 newest entries from the Security event log (you can use a different log, such as System or Application, if your Security log is empty). Sort the list so that the oldest entries appear first, and so that entries made at the same time are sorted by their index. Display the index, time, and source for each entry. Put this information into a text file (not an HTML file, just a plain text file). You may be tempted to use Select-Object and its –first or –last parameters to achieve this; don’t. There’s a better way. Also, avoid using Get-WinEvent for now – there’s a better cmdlet to work with for this particular task.

PS C:\Users\ve40013372> Get-EventLog -LogName Application -Newest 50 | Sort TimeGenerated,Index | Select Index,TimeGenerated,Source | Out-File applicationlogs.txt

PS C:\Users\ve40013372> .\applicationlogs.txt

PS C:\Users\ve40013372>  
  
**Get-EventLog -LogName Application -Newest 50 | Sort TimeGenerated,Index | Select Index,TimeGenerated,Source | Out-File applicationlogs.txt**