**Lab 1 Introduction the Powershell**

1. Select fonts and colors that work for you.
2. Make sure the console application has no horizontal scroll bar at the bottom.
3. In the ISE, get the Console Pane maximized; remove or leave the Commands Explorer at your discretion.
4. In both applications, type a single quote, ', and a backtick, `, and make sure you can easily tell the difference. On a U.S. keyboard (at least), a backtick is on one of the upper-left keys, under the Escape key, on the same key as the tilde (~) character.
5. Also type (parentheses), [square brackets], <angle brackets>, and {curly brackets} to make sure the font and size you’ve selected display well, so that all of these symbols are immediately distinguishable. If there’s some visual confusion about which is which, changes fonts or select a bigger font size.

**Lab 2 Introduction to Help System**

1. First, run Update-Help and ensure it completes without errors. That will get a copy of the help on your local computer. You’ll need an internet connection, and the shell needs to run under elevated privileges (which mean it must say “Administrator” in the shell window’s title bar).
2. Can you find any cmdlets capable of converting other cmdlets’ output into HTML?
3. Are there any cmdlets that can redirect output into a file, or to a printer?
4. How many cmdlets are available for working with processes? (Hint: remember that cmdlets all use a singular noun.)
5. What cmdlet might you use to write to an event log?
6. You’ve learned that aliases are nicknames for cmdlets; what cmdlets are available to create, modify, export, or import aliases?
7. Is there a way to keep a transcript of everything you type in the shell, and save that transcript to a text file?
8. It can take a long time to retrieve all of the entries from the Security event log. How can you get only the 100 most recent entries?
9. Is there a way to retrieve a list of the services that are installed on a remote computer?
10. Is there a way to see what processes are running on a remote computer?
11. Examine the help file for the Out-File cmdlet. The files created by this Cmdlet default to a width of how many characters? Is there a parameter that would enable you to change that width?
12. By default, Out-File will overwrite any existing file that has the same filename as what you specify. Is there a parameter that would prevent the cmdlet from overwriting an existing file?
13. How could you see a list of all aliases defined in PowerShell?
14. Using both an alias and abbreviated parameter names, what is the shortest command line you could type to retrieve a list of running processes from a computer named Server1?
15. How many cmdlets are available that can deal with generic objects? (Hint: remember to use a singular noun like “object” rather than a plural one like “objects”.)

**Lab 3: Commands Execution**

1. Display a list of running processes.
2. Display the 100 most recent entries from the Application event log
3. Display a list of all commands that are of the “cmdlet” type (this is tricky
4. Display a list of all aliases.
5. Make a new alias, so you can run d to get a directory listing.
6. Display a list of services that begin with the letter M. Again, read the help for the necessary command—and don’t forget that the asterisk (\*) is a near-universal wildcard in PowerShell.
7. Display a list of all Windows Firewall rules. You’ll need to use Help or Get-Command to discover the necessary cmdlet. 8 Display a list only of inbound Windows Firewall rules. You can use the same cmdlet as in the previous task, but you’ll need to read its help to discover the necessary parameter and its allowable values.

**Lab 4: Powershell Providers**

1. In the registry, go to HKEY\_CURRENT\_USER\software\microsoft\Windows\currentversion\explorer. Locate the Advanced key, and set its DontPrettyPath property to 1.
2. Create a zero-length file named C:\Test.txt (use New-Item).
3. Is it possible to use Set-Item to change the contents of C:\Test.txt to TESTING? Or do you get an error? If you get an error, why?
4. What are the differences between the -Filter, -Include, and -Exclude parameters of Get-ChildItem?
5. Initial Server configuration.
   1. Assign IP address
   2. Set Time zone
   3. Rename Computer
   4. Enable RDP
   5. Join Domain
   6. Disable the ServerManager Auto start
   7. Software inventory

**Lab 5: Concept of Pipeline**

1. What happens if you run Get-Service | Export-CSV services.csv | Out-File from the console? Why does that happen?
2. 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?
3. What if you want to create a pipe-delimited file instead of a comma-separated (CSV) file? You would still use the Export-CSV command, but what parameters would you specify?
4. 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 want to omit that from a particular file?
5. 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?
6. Windows maintains several regional settings, which include a default list separator. On U.S. Systems, that separator are a comma. How can you tell Export-CSV to use the system’s default separator, rather than a comma?

**Lab 6: Extending Shell**

1. Check to see if you have any PS Snapin is installed on your machine. If Yes what is its name and how to find commands associated with that.
2. List the modules available on machine.
3. Load the Server Manager Module and check if the commands associated with it are added in to the shell.
4. Explore the help file associated with ServerManager Modules and examples.
5. Pull the List of server role or features Installed on your Local machine.
6. Import module for networking on local computer and explore it.
7. Read help file and identify the tasks that could be achieved with it. Check if there is any command for modifying the advanced property of Ethernet network adapter? How?
8. How to check the IP address of Net adapter.

**Lab 7: Object by another Name**

1. Identify a cmdlet that will produce a random number.
2. Identify a cmdlet that will display the current date and time.
3. What type of object does the cmdlet from task #2 produce? (What is the type name of the object produced by the cmdlet?)
4. Using the cmdlet from task #2 and Select-Object, display only the current day of the week in a table like the following (caution: the output will right-align, so make sure your PowerShell window doesn’t have a horizontal scroll bar):

DayOfWeek

---------

Monday

1. Identify a Cmdlet that will display information about installed hotfixes.
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. Remember that the column headers shown in a command’s default output aren’t necessarily the real property names—you’ll need to look up the real property names to be sure.
3. Get the list of process and modify the Headers of columns, also display the values in MB rather than KB. Hint: You will have to use Hashtable.

**Lab 8: Pipeline in Details**

1. Would the following command work to retrieve a list of installed hotfixes from all computers in the specified domain? Why or why not? Write out an explanation. Get-HotFix -computername (get-adcomputer -filter \* | Select-Object -expand name)
2. Would this alternative command work to retrieve the list of hotfixes from the same computers? Why or why not? Write out an explanation

Get-adcomputer -filter \* | Get-HotFix or Get-content C:\temp\servers.txt | get-hotfix

1. Would this third version of the command work to retrieve the list of hotfixes from the domain computers? Why or why not? Write out an explanation.

Get-adcomputer -filter \* | Select-Object @{l='computername';e={$\_.name}} | Get-HotFix

1. Write a command that uses pipeline parameter binding to retrieve a list of running processes from every computer in an Active Directory (AD) domain. Don’t use parentheses.
2. Write a command that retrieves a list of installed services from every computer in an AD domain. Don’t use pipeline input; instead use a parenthetical command

(a command in parentheses).

1. Sometimes Microsoft forgets to add a pipeline parameter binding to a cmdlet. For example, would the following command work to retrieve information from every computer in the domain? Write out an explanation.

get-adcomputer -filter \* | Select-Object @{l='computername';e={$\_.name}} | Get-WmiObject -class Win32\_BIOS

**Lab 9: Formatting**

1. Display a table of processes that includes only the process names, IDs, and whether or not they’re responding to Windows (the Responding property has that information). Have the table takes up as little horizontal room as possible, but don’t allow any information to be truncated.
2. Display a table of processes that includes the process names and IDs. Also include columns for virtual and physical memory usage, expressing those values in megabytes (MB).
3. Use Get-EventLog to display a list of available event logs. (Hint: you’ll need to read the help to learn the correct parameter to accomplish that.) Format the output as a table that includes, in this order, the log display name and the retention period. The column headers must be “LogName” and “RetDays.”
4. Display a list of services so that a separate table is displayed for services that are started and services that are stopped. Services that are started should be displayed first. (Hint: you’ll use a -groupBy parameter).

**Lab 10: Filtering and comparisons**

1. Import the NetAdapter module (available in the latest version of Windows, both client and server). Using the Get-NetAdapter cmdlet, display a list of nonvirtual network adapters (that is, adapters whose Virtual property is False, which PowerShell represents with the special $False constant).
2. Import the DnsClient module (available in the latest version of Windows, both client and server). Using the Get-DnsClientCache cmdlet, display a list of A and AAAA records from the cache. Hint: if your cache comes up empty, try visiting a few web pages first to force some items into the cache.
3. Display a list of hotfixes that are security updates.
4. Using Get-Service, is it possible to display a list of services that have a start type of Automatic, but that aren’t currently started? Answer “Yes” or “No” to this question. You don’t need to write a command to accomplish this.
5. Display a list of hotfixes that were installed by the Administrator, and which are updates. Note that some hotfixes won’t have an “installed by” value—that’s OK.
6. Display a list of all processes running with either the name “Conhost” or the name “Svchost”.

**Lab 11: Powershell Remoting**

1. Make a one-to-one connection with a remote computer (or with “localhost” if you only have one computer). Launch Notepad.exe. What happens?
2. Using Invoke-Command, retrieve a list of services that aren’t started from one or two remote computers (it’s OK to use “localhost” twice if you only have one computer). Format the results as a wide list. (Hint: it’s OK to retrieve results and have the formatting occur on your computer—don’t include the Format cmdlets in the commands that are invoked remotely.)
3. Use Invoke-Command to get a list of the top ten processes for virtual memory (VM) usage. Target one or two remote computers, if you can; if you only have one computer, target “localhost” twice.
4. Create a text file that contains three computer names, with one name per line. It’s OK to use the same computer name, or “localhost,” three times if you only have access to one remote computer. Then use Invoke-Command to retrieve the 100 newest Application event log entries from the computer names listed in that file.

**Lab 12: Windows Management Instrumentation**

1. What class can you use to view the current IP address of a network adapter? Does the class have any methods that you could use to release a DHCP lease? (Hint: network is a good keyword here.)
2. Create a table that shows a computer name, operating system build number, operating system description (caption), and BIOS serial number. (Hint: you’ve seen this technique, but you’ll need to reverse it a bit and query the OS class first, then query the BIOS second.)
3. Query a list of hotfixes using WMI. (Hint: Microsoft formally refers to these as quick fix engineering.) Is the list different from that returned by the Get-Hotfix Cmdlet?
4. Display a list of services, including their current statuses, their start modes, and the accounts they use to log on.
5. Can you find a class that will display a list of installed software products? Do you consider the resulting list to be complete?

**Lab 13: Multitasking with Background Jobs**

1. Create a one-time background job to find all the PowerShell scripts on the C: drive. Any task that might take a long time to complete is a great candidate for a job.
2. You realize it would be helpful to identify all PowerShell scripts on some of your servers. How would you run the same command from task 1 on a group of remote computers?
3. Create a background job that will get the latest 25 errors from the system event log on your computer and export them to a CliXML file. You want this job to run every day, Monday through Friday at 6:00 a.m., in order for it to be ready for you to look at when you come in to work.
4. What cmdlet would you use to get the results of a job, and how would you save the results in the job queue?

**Lab 14: Working with Many objects**

1. What method of a ServiceController object (produced by Get-Service) will pause the service without stopping it completely?
2. What method of a Process object (produced by Get-Process) would terminate a given process?
3. What method of a WMI Win32\_Process object would terminate a given process?
4. Write four different commands that could be used to terminate all processes named “Notepad”, assuming that multiple processes might be running under that same name.

**Lab 15: Security in Powershell**

1. Define all 4 Execution policies and set them in console.
2. Create the Services.ps1 file that will query bits service on Local computer as well as one remote computer and Display the status on Screen.
3. Try Executing script created in step# 2 with each Execution policy and identify the error you encounter each time.
4. Generate a self signed certificate and Sign you script with digital signature.
5. Check the difference between signed and unsigned script.
6. Try modifying signed script and save it? Would it execute? Yes or No? Why?
7. Try modifying unsigned script and save it? Would it execute? Yes or No? Why?

**Lab 16: Variables**

1. Create a background job that queries the Win32\_BIOS information from two computers (use “localhost” twice if you only have one computer to experiment with).
2. When the job finishes running, receive the results of the job into a variable.
3. Display the contents of that variable.
4. Export the variable’s contents to a CliXML file.

**Lab 17: Input and Output**

1. Use Write-Output to display the result of 100 multiplied by 10.
2. Use Write-Host to display the result of 100 multiplied by 10.
3. Prompt the user to enter a name, and then display that name in yellow text.
4. Prompt the user to input the input the password and the password must be save in encrypted format.
5. Prompt the user to enter a name, and then display that name only if it’s longer than five characters. Do this all in a single line—don’t use a variable.

**Lab 18:**

1. Close all open sessions in your shell.
2. Establish a session to a remote computer. Save the session in a variable named $session.
3. Use the $session variable to establish a one-to-one remote shell session with the remote computer. Display a list of processes, and then exit.
4. Use the $session variable with Invoke-Command to get a list of services from the remote computer.
5. Use Get-PSSession and Invoke-Command to get a list of the 20 most recent Security event log entries from the remote computer.
6. Use Invoke-Command and your $session variable to load the ServerManager module on the remote computer.
7. Import the ServerManager module’s commands from the remote computer to your computer. Add the prefix “rem” to the imported commands’ nouns. 8 Run the imported Get-WindowsFeature command.
8. Close the session that’s in your $session variable.