Lab 8: PowerShell Scripting

The objective of the lab is to familiarize you with Command Line Interface (CLI) tools used to script and automate various network administration tasks. In this exercise, you will use Microsoft PowerShell, a command line scripting environment to automate two tasks.

This is a team assignment, but you must submit it as individuals, with all submitted work being your own.

Actions

Action 1: From your fellow classmates, form teams of no more than four members. Each team has been assigned a Windows Server 2019 instance.

For Deliverable #1, write your name, the number of your team, the name of your partners, and the date.

Connect to the CCI Virtual Environment

Action 2: You will find instructions for connecting to the CCI virtual environment in Actions 1 – 3 of Lab 1: *Access Virtual Lab Environment*.

As advised in *Lab 1: Access Virtual Lab Environment*, you may want to uncheck various local sharing checkboxes, for security reasons.

This week, you will be connecting to both your Microsoft Windows Server 2019 instance (as a team) and to your respective Windows 10 computer instances (as individuals).

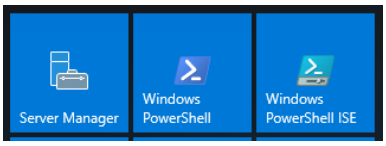
Rather than connecting to your individual Microsoft Windows 10 instances, however, one member of your team will log into your team's Microsoft Windows Server 2019 instance, as you did in previous labs, and the rest of your team will use TightVNC to work as a group

Action 3: Have one team member sign in to your team's Microsoft Windows 2019 Server instance using the *AdminLite* account. The password was given to you in class.

Since this is a Domain account, the full context of the username will be CCI-LAB-DOM\AdminLite

Invoke the Microsoft PowerShell Command Line Interface (CLI)

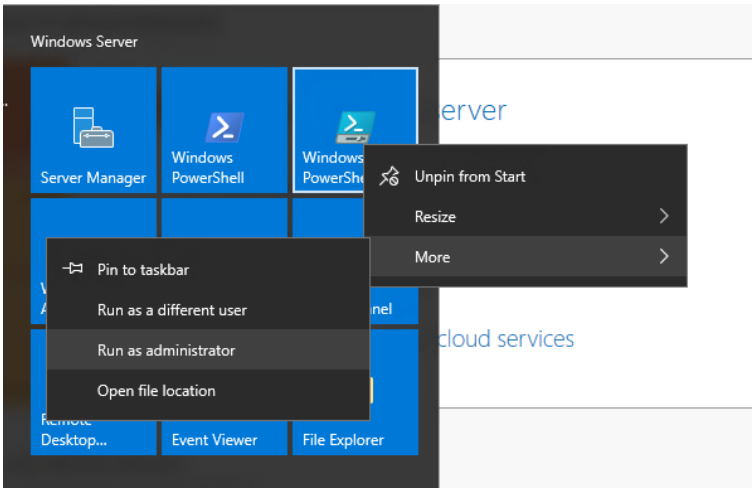
Action 4: Click the *Start* button, and right-click Windows PowerShell ISE.



Microsoft Windows PowerShell ISE is a PowerShell Integrated Scripting Environment that also serves as a graphical editor.

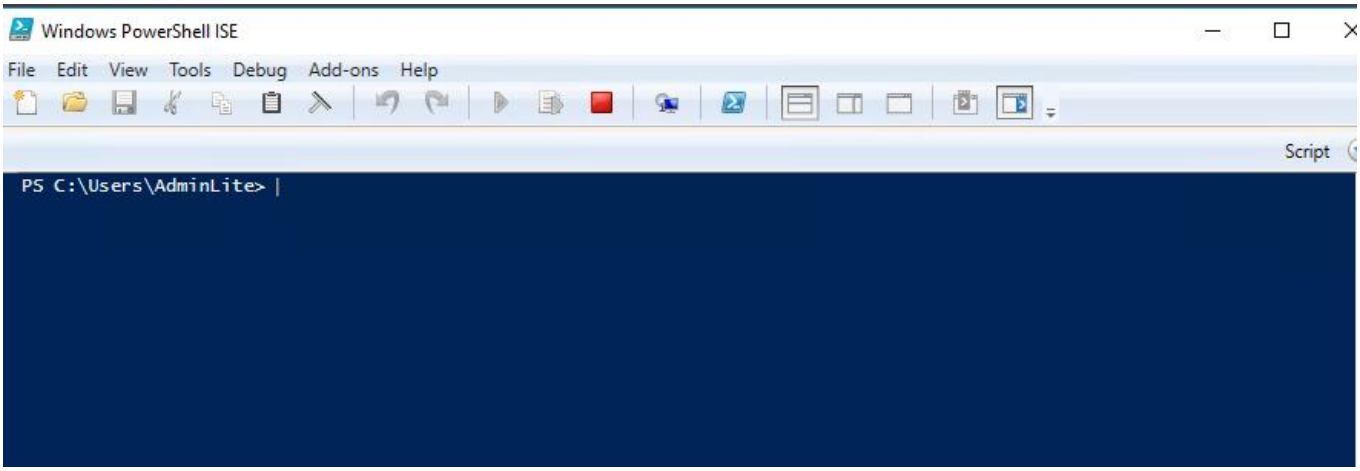
The Microsoft Windows PowerShell ISE
https://docs.microsoft.com/en-us/powershell/scripting/components/ise/introducing-the-windows-powershell-ise?view=powershell-6

For some of the following exercises, you will need to run PowerShell ISE with elevated permissions, as such:



Select *Run as administrator*.

You will see the Windows PowerShell ISE environment as pictured below:



Invoke the Microsoft PowerShell Command Line Interface (CLI) (continued)

If you have worked in a Linux CLI, you may recognize some of the same commands. For example, the following three commands common to Linux, may also be used in PowerShell:

whoami	displays the username of the present system user
pwd	Present Working Directory displays which file directory you are presently in
ps	lists processes currently running

Run each of the three commands above, and answer the questions in the deliverable below.

For Deliverable #2, what is the username of the present user? What is the present working directory? Name three processes currently running on your server.

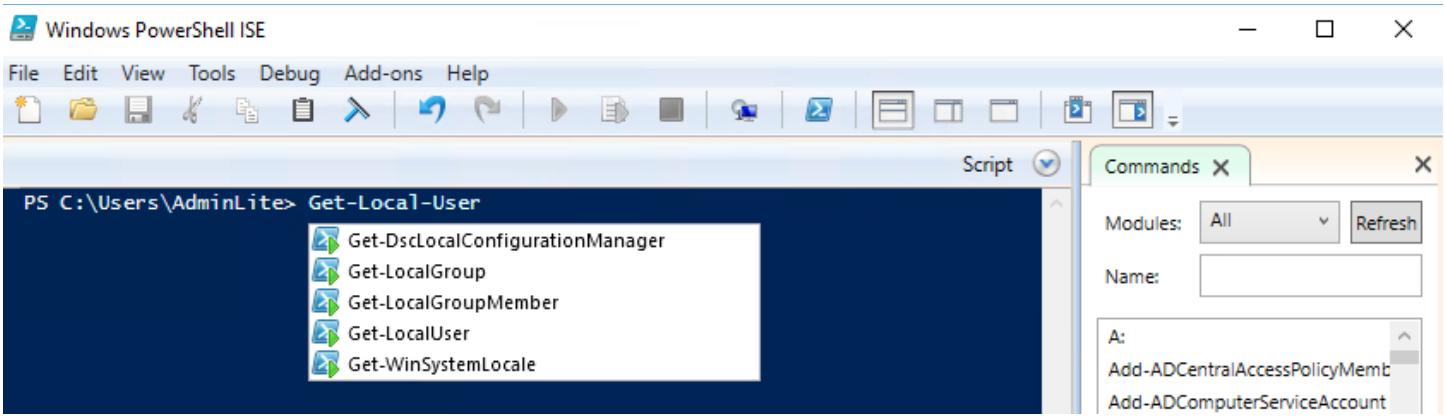
Perform Basic User Management in PowerShell

Action 5: PowerShell allows you to manage local and Active Directory user accounts, both interactively from the Command Line, and from scripts.

First, you will list local users using the Get-LocalUser command:

Get-LocalUser
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localuser?view=powershell-5.1

Enter Get-LocalUser in your PowerShell ISE CLI:



Note that like certain programming interfaces that you may have used in the past, PowerShell ISE has a powerful autocomplete and help feature. As you can see in the example above, the command was in the process of being entered incorrectly with an extraneous dash between Local and User, and the ISE suggested the correct form.

You can also see that there are other possible commands, such as get-LocalGroup, which would list local groups.

For Deliverable #3, list the local users on your server.

You won't be asked to list all the local groups, but you will be asked to count them. The number of users in this exercise was small enough to efficiently count by hand, but the number of default groups alone is likely to be high enough to preclude that.

Perform Basic User Management in PowerShell (continued)

As with most CLI commands, however, you can redirect the output of one command into the input of another. In this exercise, you will use the pipe operator: |

You will find the pipe symbol above the <ENTER> key on your keyboard.

Enter the following chain of commands:

```
Get-LocalGroup | select-string $ | measure-object Line
```

This pipes the output from the first command, which lists all local groups, into the second command, select-string $ (which is a bit like grep in Linux). The output from the second command is piped into the third, which calculates the numeric properties of the second, i.e. the number of lines.

For Deliverable #4, how many local groups exist on your server?

Action 6: In previous labs, you were asked to use the Microsoft Windows Graphical User Interface (GUI) to determine which users and groups were members of specific groups. For example, one action item had you look at your local *Administrators* group to confirm that the *Domain Administrators* group had been joined to it when you joined the computer to the Domain.

You can perform the same task from the command line:

```
Get-LocalGroupMember -Group “Administrators”
```

Enter that command at this time. Remember that you can use the ISE autocomplete to save you from having to type out entire commands.

Get-LocalGroupMember
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/get-localgroupmember?view=powershell-5.1

A command such as the one above could be useful in, say, speeding security audits. More advanced PowerShell users could pipe and argument to and from text files, for example.

```
PS C:\Users\AdminLite> Get-LocalGroupMember -Group "Administrators"

ObjectClass Name                          PrincipalSource
----------- ----                          ---------------
Group       CCI-LAB-DOM\Domain Admins     ActiveDirectory
User        LAB-WIN2016-13\Administrator  Local
```

For Team 13 server (Instructor Team), group CCI/LAB-DOM\Domain Admins (an Active Directory account) and user LAB-WIN2019-13\Administrator happens to be members of the local group named “Administrators”.

The context for an Active Directory object always includes the name of the Domain, which in this case is CCI-LAB-DOM. The context for a local object consists of the name of the server itself, which in this case is LAB-WIN2019-13.

For Deliverable #5, name the user and group objects, using full-context, that have local administrator access to your server.

Perform Basic User Management in PowerShell (continued)

Action 7: You may also create, assuming sufficient permissions, users and groups as well:

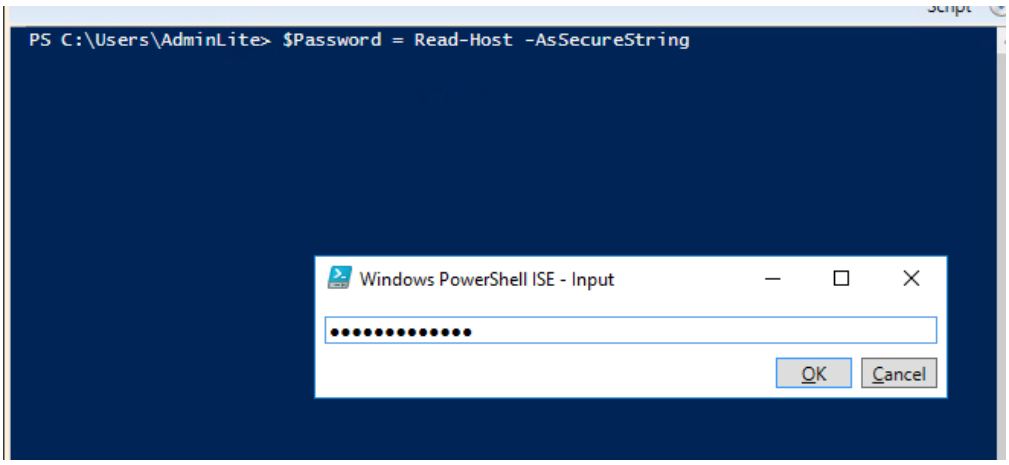
New-LocalUser
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/new-localuser?view=powershell-5.1

You are going to create a user named “FooUser01”, entering a password as a secure string into variable $Password, with a full name of “Test User”, with a brief description.

First, set the $Password variable:

```
$Password = Read-Host -AsSecureString
```

This invokes a commandlet to elicit user input from the console, and place it into a secure string:



WARNING: IN THIS CLASS, NEVER SET PASSWORDS TO ANY PASSWORD THAT YOU USE ON ANY OTHER SYSTEM, AS THIS IS A TRAINING NETWORK, AND IT IS NOT CONSIDERED SECURE!

Now create the local user:

```
New-LocalUser "FooUser01" -Password $Password -FullName "Test User" -Description "PowerShell Demo”
```

```
PS C:\Windows\system32> $Password = Read-Host -AsSecureString

PS C:\Windows\system32> New-LocalUser "FooUser01" -Password $Password -FullName "Test User" -Description "PowerShell Demo"

Name      Enabled Description
----      ------- -----------
FooUser01 True    PowerShell Demo
```

Repeat Get-LocalUser to confirm that the user you created is listed among the other local users.

You may also use PowerShell to create Active Directory users and groups.

Run scripts in PowerShell

The PowerShell scripting language is truly powerful, and may be used in a manner similar to that of *NIX shell scripts, Perl, or Python in automating network administration tasks. PowerShell allows, as you saw in the previous example, variables, as well as loops and conditionals.

The course textbook contains an entire chapter on PowerShell, and other resources abound. This is just one of many:

Getting Started with Microsoft PowerShell
https://mva.microsoft.com/en-us/training-courses/getting-started-with-powershell-3-0-jump-start-8276

Coding scripts from scratch is beyond the scope of this lab. However, you will run a pre-made script to perform a network management task.

Action 8: For this exercise, you will download and execute a PowerShell Script.

The script in question can be found here:

SubNet Scan
https://gallery.technet.microsoft.com/scriptcenter/SubNet-Scan-dad0311f

This script scans a range of IP addresses, producing to the console reports.

There are many methods of downloading files from the Internet. However, using a web browser on a Windows Server console tends not only to be ill-advised, for security reasons, but also tends to be notoriously cumbersome.

Server administrators often perform web browsing and file download tasks from platforms friendlier to the task, such as a Windows 10 computer: downloaded files are simply redirected to a share on the server on which the files are needed. You could use PowerShell to create such a share; in previous labs, you used the Windows GUI to create shares.

Let's take a more direct route.

In the PowerShell ISE CLI, create a directory using the following command:

```
mkdir c:/scripts
```

Those of you familiar with *NIX will recognize mkdir as the "create a directory" command. To do the same in the Windows GUI, approximately five clicks would have been required, with additional file browsing, and the file name *still* would have had to been typed in.

The command below might download the needed file directly to the /scripts folder:

Invoke-WebRequest https://gallery.technet.microsoft.com/scriptcenter/SubNet-Scan-dad0311f/file/44203/5/Get-SubNetItems.ps1 -OutFile c:\scripts\Get-SubNetItems.ps1

However, this is quite a bit of typing. Note that command lines are very handy when lengthy commands already exist in scripts or files – or when copying-and-pasting is available. They aren't so useful when this much typing is involved each time.

Run scripts in PowerShell (continued)

The necessary file also exists in a share folder on the course network:

The UNC is \\CCI-LAB-DC1\labfiles

You may temporarily map local drive L to this share with the following command:

New-PSDrive -Name "L" -PSProvider FileSystem -Root \\CCI-LAB-DC1\labfiles

```
PS C:\Windows\system32> New-PSDrive -Name "L" -PSProvider FileSystem -Root \\CCI-LAB-DC1\labfiles

Name           Used (GB)     Free (GB) Provider      Root                                               CurrentLocation
----           ---------     --------- --------      ----                                               ---------------
L                                      FileSystem    \\CCI-LAB-DC1\labfiles
```

You may now try to run the file with the command

L:\ Get-SubNetItems.ps1

It very likely will produce an error that it can't run because the ps1 file wasn't digitally-signed. PowerShell scripts can do a tremendous amount of damage to a network, especially if the account under which they are run is not configured under the principle of least-access.

About Execution Policies
https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-6

For purposes of **this exercise only**, we will grant an exemption for this script only. Do use extreme caution when issuing exemptions to any unsigned script.

Deliverables

Deliverable 1: From Action 1, write your name, the number of your team, the name of your partners, and the date.

Deliverable 2: From Action 4, what is the username of the present user? What is the present working directory? Name three processes currently running on your server.

Deliverable #3: From Action 5, list the local users on your server.

Deliverable #4: From Action 5, how many local groups exist on your server?

Deliverable #5: From Action 6, name the user and group objects, using full-context, that have local administrator access to your server.