# Powershell Scripting

💡The notion of Windows vs Linux Administration roles is somewhat antiquated.  Tech pro's in today's competitive workforce need to be familiar and multi-lingual in terms of scripting languages used on Linux and Windows platforms.

This week you will take a tour of the Windows side of scripting.

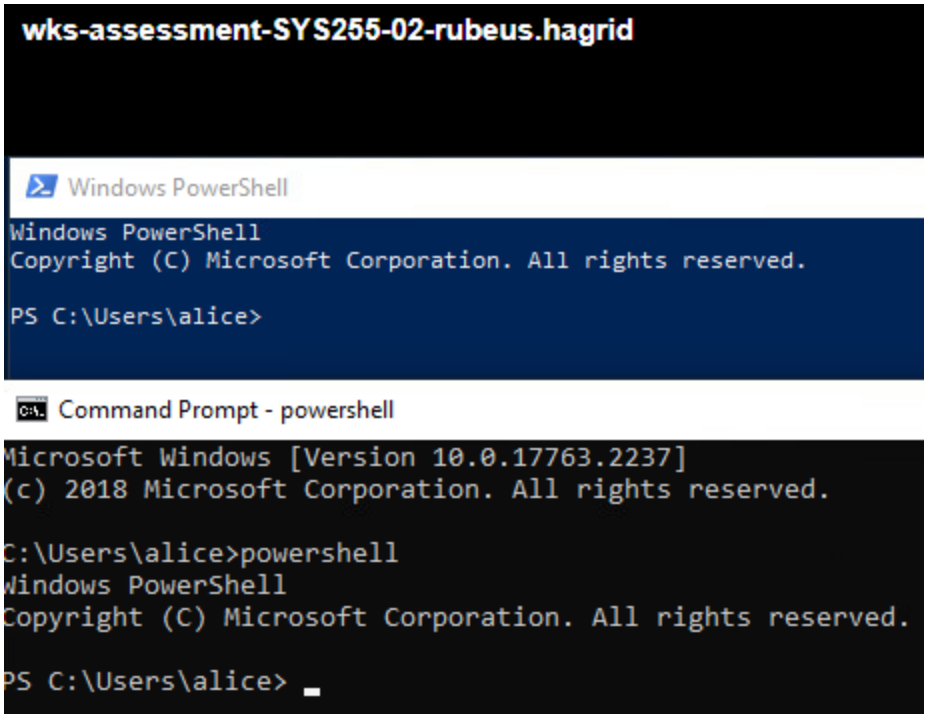
## Prerequisites

We will be using both your AD server, File Server, and WKS.

## Invoking Powershell

The easiest way to start Powershell is to launch it from the start menu.  You can also type *Powershell* at the command prompt.  The following shows starting Powershell as a limited user and also from the command prompt.  Try both.  We will be using an unprivileged account on WKS unless otherwise specified.

💡Note: if you are logged in as an unprivileged user and you attempt to "run as" Administrator, you may be tripped up by a notification that the Administrator has blocked the application.  This is ok.  But if you want to change this behavior, go to the end of the lab and there is a solution there.




## Path, shortcuts, command completion and history

You will find that many of your Linux bash shortcuts and commands will work in Powershell. The Powershell designers have created "aliases" so that commands like ls, pwd, cd ~, history will all work on windows. Powershell is an object-oriented scripting language.  To access the path string from the environment object, note the first command in the illustration.

```
Windows PowerShell
PS C:\Users\alice> Write-Host $env:Path
C:\Windows\system32;C:\Windows;C:\Windows\
OpenSSH\;C:\Users\alice\AppData\Local\Micr
PS C:\Users\alice> cd c:\users ; pwd

Path
----
C:\users


PS C:\users> cd ~ ; pwd

Path
----
C:\Users\alice


PS C:\Users\alice> history

  Id CommandLine
  -- -----------
   1 Write-Host $env:Path
   2 cd C:\Users\ && pwd
   3 clear
   4 Write-Host $env:Path
   5 cd c:\users ; pwd
   6 cd ~ ; pwd


PS C:\Users\alice>
```

## Looping

The following code sequence shows assignment of the $env:Path string to the $mypath variable, followed by the conversion of that path to an array using the split operator. Once we have the array, we loop through it using the Foreach method. Give this a try:

```
Windows PowerShell
PS C:\Users\alice> $mypath = $env:Path
PS C:\Users\alice> Write-Host $mypath
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32
OpenSSH\;C:\Users\alice\AppData\Local\Microsoft\WindowsApps;
PS C:\Users\alice> foreach($item in $mypath.split(';'))
>> {
>> Write-Host $item
>> }
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Users\alice\AppData\Local\Microsoft\WindowsApps

PS C:\Users\alice>
```

# Aliasing and Get-ChildItem

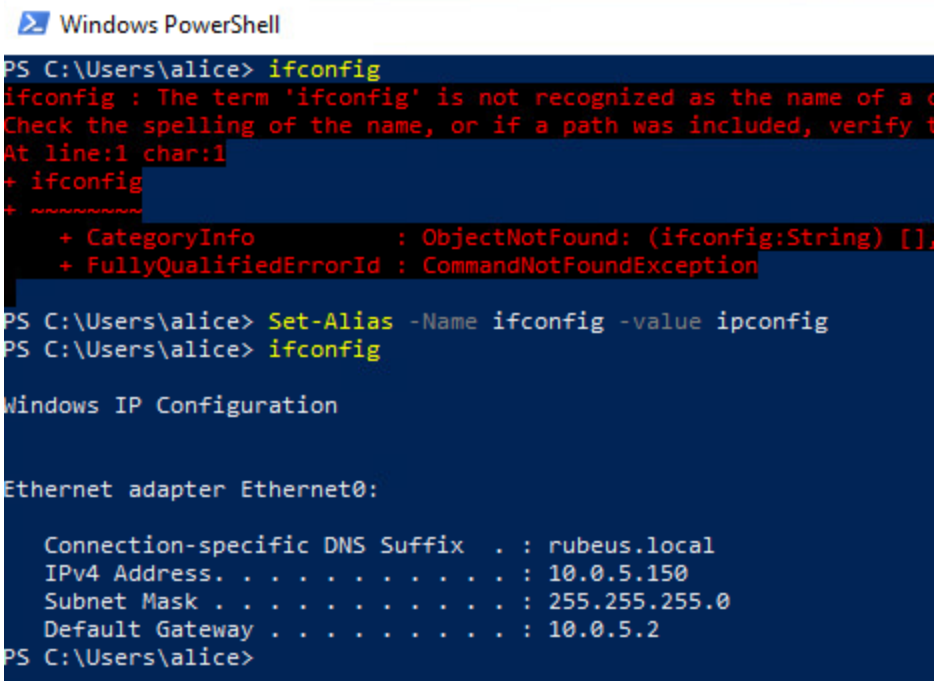
The following screenshot further illustrates the object-oriented nature of Powershell. The legacy *dir* and *ls* commands really point to a Powershell "cmdlet" called Get-ChildItem. If the **object** contains other **objects**, it can be **enumerated**.

Windows PowerShell

```
PS C:\Users\alice> alias dir

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           dir -> Get-ChildItem


PS C:\Users\alice> alias | findstr Get-ChildItem
Alias           dir -> Get-ChildItem
Alias           gci -> Get-ChildItem
Alias           ls -> Get-ChildItem
PS C:\Users\alice> ls Env:

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\alice\AppData\Roaming
CommonProgramFiles             C:\Program Files\Common Files
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
CommonProgramW6432             C:\Program Files\Common Files
COMPUTERNAME                   WKS02-RUBEUS
ComSpec                        C:\Windows\system32\cmd.exe
DriverData                     C:\Windows\System32\Drivers\DriverData
HOMEDRIVE                      C:
HOMEPATH                       \Users\alice
LOCALAPPDATA                   C:\Users\alice\AppData\Local
LOGONSERVER                    \\AD02-RUBEUS
NUMBER_OF_PROCESSORS           2
OS                             Windows_NT
Path                           C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.
PROCESSOR_ARCHITECTURE         AMD64
PROCESSOR_IDENTIFIER           Intel64 Family 6 Model 47 Stepping 2, GenuineIntel
PROCESSOR_LEVEL                6
PROCESSOR_REVISION             2f02
ProgramData                    C:\ProgramData
ProgramFiles                   C:\Program Files
ProgramFiles(x86)              C:\Program Files (x86)
ProgramW6432                   C:\Program Files
PSModulePath                   C:\Users\alice\Documents\WindowsPowerShell\Modules;C:\P
PUBLIC                         C:\Users\Public
SESSIONNAME                    Console
SystemDrive                    C:
SystemRoot                     C:\Windows
TEMP                           C:\Users\alice\AppData\Local\Temp
```

## Create your own Alias

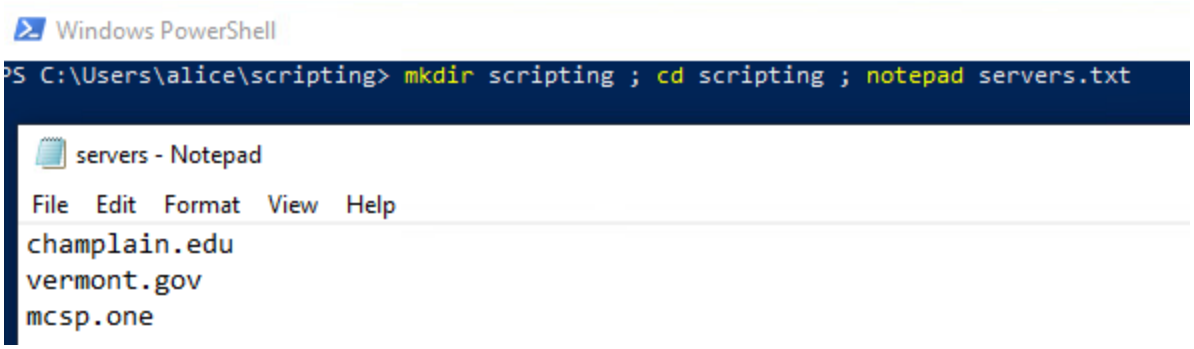
Here's a common alias used in Windows by those who grew up with Linux. Note the error when the Linux admin types in ifconfig. Setting the alias this way only lasts for the current session.  If you want one to be persistent, you need to research Powershell profiles.

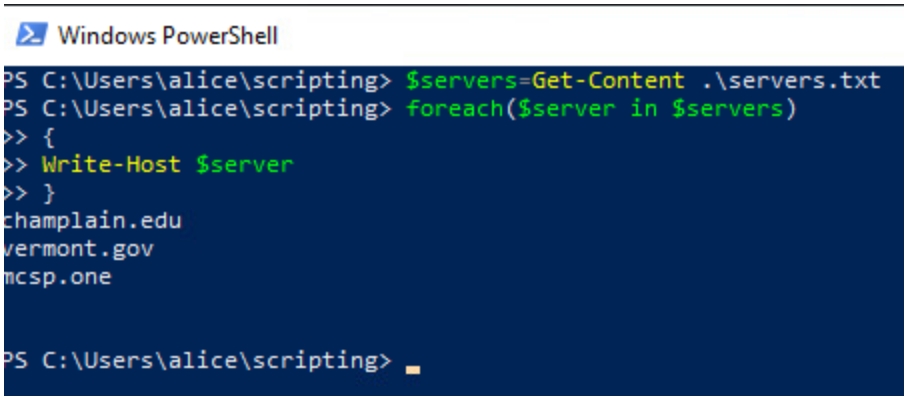



Deliverable 1.  Create and demonstrate your own alias **other than ifconfig.**  Provide a screenshot of the alias creation syntax and execution.

## Creating a Script

Take the list of servers (servers.txt), and Powershell file(servers.ps1), you may wish to have windows explorer show file extensions.You will notice right away when executing the script via ./servers.ps1 that there is an error.  You have to configure windows to allow Powershell scripting.



```
Windows PowerShell
PS C:\Users\alice\scripting> cat .\servers.ps1
$servers = Get-Content .\servers.txt
foreach($server in $servers)
{
Write-Host $server
}
PS C:\Users\alice\scripting> .\servers.ps1
.\servers.ps1 : File C:\Users\alice\scripting\servers.ps1 cannot be loaded because running scripts is disabled on this
system. For more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\servers.ps1
+ ~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\alice\scripting> Set-ExecutionPolicy -Scope CurrentUser RemoteSigned

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): Y
PS C:\Users\alice\scripting> .\servers.ps1
champlain.edu
vermont.gov
duckduckgo.com
mcsp.one
PS C:\Users\alice\scripting>
```

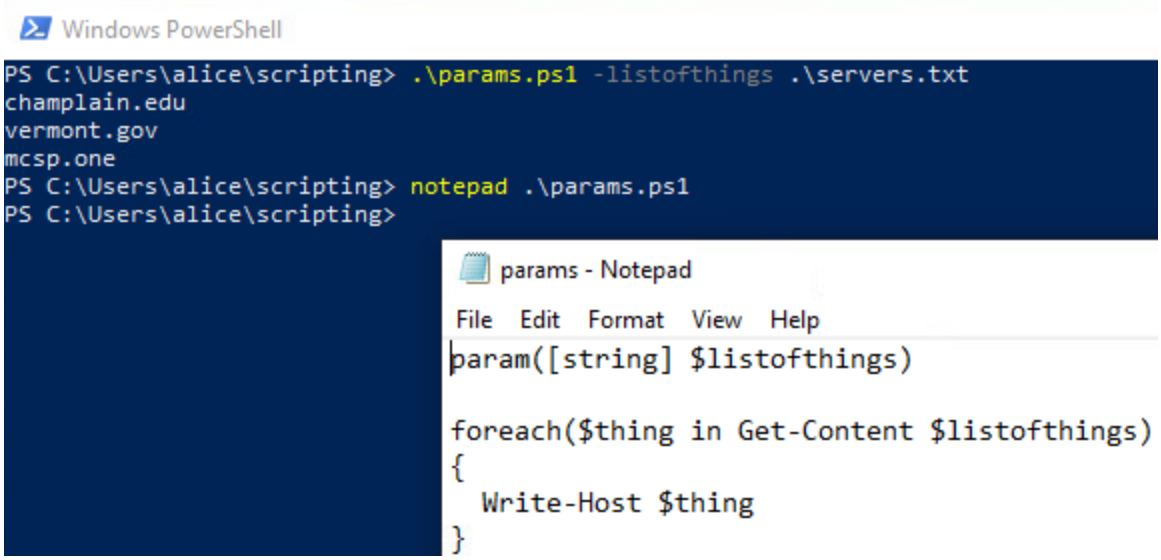
The Set-Execution-Context command shown above allows current users to run local scripts and digitally signed remote scripts.  This is analogous to chmod +x  on Linux for all the user's Powershell scripts.

Deliverable 2. Extend this script to ping each server in the list one time. Provide a screenshot showing the syntax and output of your script.

```
Windows PowerShell
PS C:\Users\alice\scripting> .\servers.ps1

Pinging champlain.edu [208.115.107.132] with 32 bytes of data:
Reply from 208.115.107.132: bytes=32 time=81ms TTL=48

Ping statistics for 208.115.107.132:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 81ms, Maximum = 81ms, Average = 81ms

Pinging vermont.gov [199.107.32.35] with 32 bytes of data:
Reply from 199.107.32.35: bytes=32 time=41ms TTL=233

Ping statistics for 199.107.32.35:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 41ms, Maximum = 41ms, Average = 41ms

Pinging mcsp.one [217.160.0.8] with 32 bytes of data:
Reply from 217.160.0.8: bytes=32 time=106ms TTL=50

Ping statistics for 217.160.0.8:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 106ms, Maximum = 106ms, Average = 106ms
PS C:\Users\alice\scripting>
```

## Parameters

The script, as created, has a flaw in that the path and file name of the servers.txt file are "hard-coded", as opposed to being dynamically passed in as a parameter.  Take a look at the following program, and then extend servers.ps1 to accept a file path as a parameter.




Consider the following set of DNS Resolution Commands:

```
Windows PowerShell
PS C:\Users\alice\scripting> Resolve-DnsName google.com

Name                                           Type   TTL   Section    IPAddress
----                                           ----   ---   -------    ---------
google.com                                     AAAA   259   Answer     2607:f8b0:4006:822::200e
google.com                                     A      259   Answer     142.251.40.206


PS C:\Users\alice\scripting> Resolve-DnsName google.com -Type A | Select-Object Name, IPAddress

Name       IPAddress
----       ---------
google.com 142.251.40.206


PS C:\Users\alice\scripting>
```

Deliverable 3. Write a script that takes two parameters that include the DNS response Type and a file with a list of hosts. Your output should look similar to the following. You may need to conduct some research to see how to pass more than one parameter. Provide a screenshot of your syntax and execution output.

```
wks-assessment1-hermione.granger                    Enforce US Key

Windows PowerShell
PS C:\Users\alice\scripting> .\resolve.ps1 -type "A" -servers .\servers.txt

Name          IPAddress
----          ---------
champlain.edu 184.154.210.190
vermont.gov   206.16.212.90
google.com    172.217.12.174
reddit.com    151.101.193.140
reddit.com    151.101.65.140
reddit.com    151.101.1.140
reddit.com    151.101.129.140
slashdot.org  216.105.38.15


PS C:\Users\alice\scripting>
```

## Remote Powershell

Move over to your AD Server (Windows Server) and open up a Powershell prompt. Though Windows does not natively support SSH for remote access, Powershell can be invoked remotely using **PSSession**. Refer to the following screenshot.

Select Windows PowerShell

```
PS C:\Users\rubeus-adm> Get-ADComputer -Filter * | Select-Object Name

Name
----
AD02-RUBEUS
WKS02-RUBEUS
FS01-RUBEUS
WEB01-RUBEUS


PS C:\Users\rubeus-adm> Enter-PSSession -ComputerName FS01-RUBEUS
[FS01-RUBEUS]: PS C:\Users\rubeus-adm\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::4ce:bf17:4150:732d%5
   IPv4 Address. . . . . . . . . . . : 10.0.5.8
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.5.2
[FS01-RUBEUS]: PS C:\Users\rubeus-adm\Documents>
```

Deliverable 4. Provide a screenshot that shows a remote PS-Session and the command of your choice on FS01.

Deliverable 5. The following command shows how one can just launch a command remotely without having an interactive session. Explore the -ScriptBlock Option and provide a screenshot showing **your own command** launched on FS01.

Windows PowerShell

```
[FS01-RUBEUS]: PS C:\Users\rubeus-adm\Documents> exit
PS C:\Users\rubeus-adm> Invoke-Command -ComputerName FS01-RUBEUS -ScriptBlock { ipconfig }

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::4ce:bf17:4150:732d%5
   IPv4 Address. . . . . . . . . . . : 10.0.5.8
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.5.2
PS C:\Users\rubeus-adm>
```

Deliverable 6. Using Powershell on AD, figure out how to add a single user to Active Directory, and then how to add that user to a domain group that you create. Provide a screenshot that shows the command syntax and execution w/ response.

Deliverable 7: If you try the PS-Remoting commands against your workstation, it may fail due to firewall and other issues. Research the problem and see if you can resolve them. Take a screenshot of remote command invocation from AD to your workstation.

# Optional Fixing Restrictive UAC on a workstation

This tweak is configured by opening the registry editor on the Windows workstation (regedit command), navigating to the registry key below, and setting its value.