

# PRIMERA ECTS DE FC



## PowerShell y el uso de Git en Github

 POWERSHELL

 GITHUB

 GIT

Nombre: Jose Luis Obiang Ela Nanguan

Profesor: Francisco Fernandez de Vega

Fecha | 27/04/2022

*Copyright :copyright: José Luis Obiang Ela Nanguan*

*Documentación del proyecto:* :point\_right:JLOEN1999

*Documentación oficial de PowerShell:* :point\_right:PowerShell

Version Actual | 27.04.2022

## Índice de contenido

1. Presentación
  - i. Objetivos
2. PowerShell
  - i. Introducción
  - ii. El paso de CMD a PowerShell
  - iii. ¿Qué es PowerShell?
  - iv. La consola de PowerShell
  - v. La ayuda en PowerShell
  - vi. Obtener Comandos con get-command y get-module
  - vii. Obtener ejemplos del uso de Comandos con get-help
  - viii. Atajos y Alias
  - ix. Gestión de archivos y carpetas
  - x. Tuberías y redirecccionamiento
  - xi. Iniciación a los scripts
  - xii. Fundamentos de scripts-I: Comentarios, Variables/Constantes
  - xiii. Fundamentos de scripts-II: Estructuras de control y funciones

## 1. Presentación

Este curso está planteado en lo más práctico posible. Las Prácticas que vamos a realizar son las siguientes:

*En primer lugar empezaremos a trabajar con la PowerShell y la PowerShell ISE.*

*Vamos a buscar Información sobre el uso de algunos comandos.*

*Vamos a utilizar los comandos básicos relacionados con la gestión de archivos y carpetas.*

*Vamos a enlazar la salida de un comando con la entrada de otro y redireccionar la salida.*

*Y vamos a realizar pequeños scripts, vamos a hacer scripts en los que combinaremos estructuras condicionales, repetitivas y redireccionamiento.*

*También haremos uso de la herramienta Git y la subida de nuestros scripts a Github*

## 1.1. Objetivos

*¿Qué vamos a conseguir al finalizar este curso?*

Pues vamos a:

|   |   |                               |   |   |
|---|---|-------------------------------|---|---|
| A manejar tanto la PowerShell como la PowerShell ISE con soltura. | Buscar información en la ayuda de PowerShell. | Conocer los comandos básicos. | Ser capaces de realizar scripts para automatizar determinadas tareas. | Conocer los comandos básicos de Git para hacer un buen seguimiento de nuestros proyectos en Github. |
|---|---|-------------------------------|---|---|

## 2. PowerShell



### 2.1. Introducción

La interfaz de usuario es el medio que utilizamos para comunicarnos con el ordenador.

Interfaz gráfica: GUI(Proporciona un entorno visual).

Interfaz de línea de comandos: CLI(Command Line Interface, nos permite dar instrucciones por medio de una línea de texto).

### ¿Cuál de las dos debemos utilizar?

La respuesta es fácil, depende de lo que queramos hacer, si quieras navegar, trabajar con un procesador de texto, hoja de cálculo, retocar fotografía, etc, tu respuesta es la interfaz gráfica , pero si lo que quieras es automatizar tareas, crear usuarios de forma masiva, comprobar conectividad con servidores pues la respuesta es PowerShell(La linea de comandos).

## 2.2. El paso de CMD a PowerShell

Vamos a ver ahora cómo ha evolucionado la línea de comandos de Windows.

### **CMD o símbolo del sistema:**

Todavía convive con nosotros, pero cada vez se utiliza menos, tal vez para hacer.

ping (Para comprobar si nuestra PC está conectada a la red o para saber si una determinada página e

```
C:\Users\JLOel>ping 8.8.8.8

Haciendo ping a 8.8.8.8 con 32 bytes de datos:
Respuesta desde 8.8.8.8: bytes=32 tiempo=10ms TTL=118
Respuesta desde 8.8.8.8: bytes=32 tiempo=12ms TTL=118
Respuesta desde 8.8.8.8: bytes=32 tiempo=13ms TTL=118

Estadísticas de ping para 8.8.8.8:
    Paquetes: enviados = 3, recibidos = 3, perdidos = 0
                (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 10ms, Máximo = 13ms, Media = 11ms
```

ipconfig (Para ver los adaptadores de red de la PC)

msinfo32 (Para conocer información sobre nuestro sistema)

| Resumen del sistema  |                              |
|----------------------|------------------------------|
| Recursos de hardware | Elemento                     |
| Componentes          | Nombre del SO                |
| Entorno de software  | Versión                      |
|                      | Descripción adicional del SO |
|                      | Fabricante del SO            |
|                      | Nombre del sistema           |
|                      | Fabricante del sistema       |
|                      | Modelo del sistema           |
|                      | Tipo de sistema              |

regedit (Que es una base de datos donde están los ajustes de configuración y opciones en los sistemas)

| Editor del Registro                 |           |                        |  |
|-------------------------------------|-----------|------------------------|--|
| Archivo Edición Ver Favoritos Ayuda |           |                        |  |
| Equipo\HKEY_CURRENT_USER            |           |                        |  |
| Nombre                              | Tipo      | Datos                  |  |
| (Predeterminado)                    | REG_SZ    | (valor no establecido) |  |
| VDIEnvironment                      | REG_DWORD | 0x00000000 (0)         |  |

### PowerShell:

En cambio, la PowerShell se pensó como una herramienta de reemplazo del CMD y con el tiempo se ha convertido en una herramienta poderosa de gestión tanto para usuarios domésticos como administradores.

## 2.3. ¿Qué es PowerShell?

La PowerShell es una nueva línea de comandos, es decir, es una herramienta multiplataforma utilizada principalmente por los administradores de Sistemas Windows para automatizar tareas y tener un mayor control del sistema. Esta herramienta está formada por una shell de comandos, un lenguaje de scripting y un marco de administración de configuración. Trabaja con objetos, acepta y devuelve objetos y no acepta ni devuelve texto como lo hace CMD. PowerShell fue desarrollado por Microsoft el 14 de noviembre de 2006 y está programado en C#.

**¿Dónde podemos encontrar PowerShell por defecto?** En Windows 10 la encontramos, Windows Server, Microsoft Azure, SQL Server, Sercivios de Office 365, se encuentra prácticamente en todos los productos de Microsoft. Se puede instalar en Linux o MacOS.

| Versiones |      |
|-----------|------|
| Versión   | año  |
| V1        | 2006 |
| V2        | 2009 |

| Versiones                   |             |
|-----------------------------|-------------|
| V3                          | 2012        |
| V4                          | 2013        |
| V5                          | 2016        |
| V5.1                        | 2017        |
| V Core 6.0                  | 2018        |
| <b>Versión actual 7.2.2</b> | <b>2022</b> |

No tenemos que confundir:

- Windows PowerShell ISE, es un entorno en el que podemos ejecutar comandos, escribir, probar y depurar script.
- Windows PowerShell es la consola de comandos.

#### *¿Qué requisitos se necesitan para aprender dicha herramienta?*

Como se trata de un curso en el ámbito de iniciación en PowerShell, cualquier persona con conocimientos de informática a nivel de usuario podría hacerlo sin mayor problema, ahora bien hay una parte en la que se habla de variables y estructuras condicionales y entonces aquí si se requiere conocimientos mínimos de programación.

#### *¿Qué máquina necesitamos para trabajar en PowerShell?*

Es suficiente con tener un Windows 10 instalado o bien un Windows Server.

## 2.4. La consola de PowerShell

Hay varias maneras de abrir la consola de comandos en Windows:

- Dando clic derecho sobre el símbolo de Windows y pinchamos donde aparece PowerShell.
- Pulsando **Windows+R** y escribimos **PowerShell** .

Podemos ver la versión que tiene nuestro PowerShell con el comando `get-host` .

 JL0el ➤ get-host

```
Name          : ConsoleHost
Version      : 7.2.2
InstanceId   : cb09674a-7e20-4738-9b25-12147a39a061
UI           : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : es-ES
CurrentUICulture : es-ES
PrivateData    : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
DebuggerEnabled : True
IsRunspacePushed : False
Runspace      : System.Management.Automation.Runspaces.LocalRunspace
```

### Concepto de cmdlet:

Los comandos en PowerShell se llaman cmdlet. Es una combinación de verbo y nombre separados por un guión (-) .

Verbo(Verb): describe la acción que se va a realizar.

Nombre(Noun) es el objeto sobre el que se va a realizar la acción.

NOTA: PowerShell incorpora incluso muchos de los comandos de Linux.

Para entender mejor, ahora abrimos la línea de Comandos PowerShell.

- Ejemplos de Verbos:
  - get (Obtiene una información)
  - set (Cambia una propiedad)
  - remove (Elimina un objeto)
  - new (Crea un objeto)

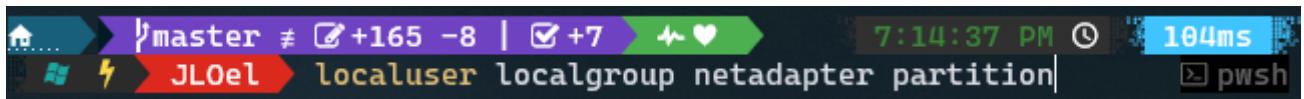
Pero estos verbos no hacen nada si no les aplicamos un nombre

 master # ↗+165 -8 | ↗+7 ➤ ↻ ❤️ 7:14:37 PM ⏱ 104ms ➤ pwsh

```
JL0el ➤ get set remove new
```

- Ejemplos de Nombres:
  - localuser
  - localgroup
  - netadapter
  - partition
  - etc..

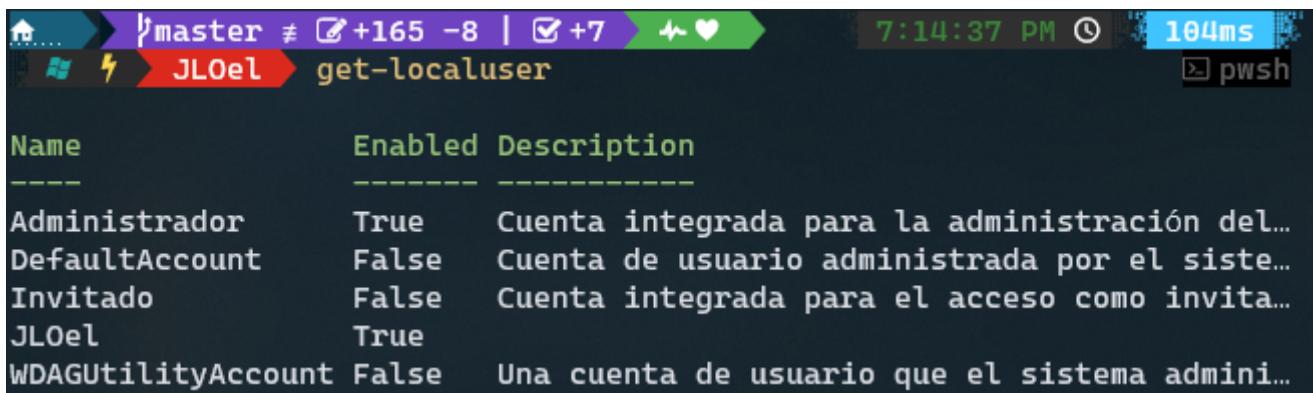
Pero estos nombres tampoco hacen algo si no les aplicamos un verbo.

A screenshot of a Windows taskbar showing the PowerShell icon. The taskbar also displays system icons for signal strength, battery, and volume, along with the current time (7:14:37 PM) and a 104ms CPU usage indicator.

- Parámetros. Los comandos en PowerShell también tienen parámetros y podemos combinarlos:
  - *Path* (Para especificar la ruta del directorio que puede ser absoluta o relativa)
  - *Force* (Para mostrar también los archivos ocultos o para borrar un elemento de manera forzosa)
  - *Recurse* (Para mostrar, copiar o mover absolutamente todos los archivos)
  - *Filter* (Para hacer un filtro a la hora buscar información)
  - *Include* (Hace lo mismo que Filter, pero siempre va precedido del parámetro Recurse)
  - *Exclude ()* (Siempre va precedido del parámetro Recurse y sirve para mostrar solo los elementos que indicamos en el parámetro)
  - etc...
- Campos: La información suele mostrarse por campos, por lo tanto, también podríamos hacer un filtro de lo que queremos mostrar especificando el nombre del campo seguido del elemento a buscar.

Vamos a combinar los verbos y los nombres a ver si ahora conseguimos algo.[Ejemplos:](#)

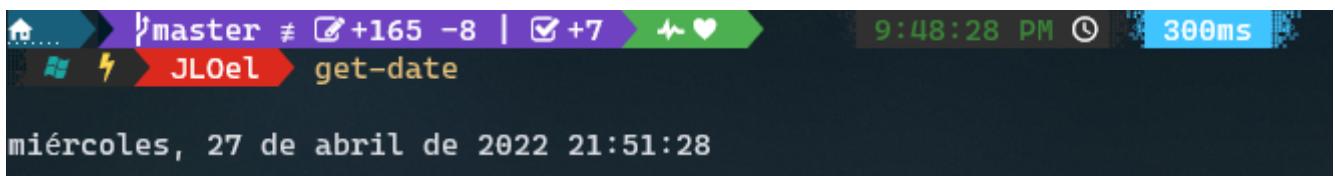
- `Get-localuser` (Muestra todos los usuarios locales del sistema).

A screenshot of a PowerShell window showing the output of the `Get-localuser` command. The output is a table with columns: Name, Enabled, and Description. The table lists several local users with their respective status and descriptions.

| Name               | Enabled | Description                                    |
|--------------------|---------|--|
| Administrador      | True    | Cuenta integrada para la administración del... |
| DefaultAccount     | False   | Cuenta de usuario administrada por el siste... |
| Invitado           | False   | Cuenta integrada para el acceso como invit...  |
| JLOel              | True    |  |
| WDAGUtilityAccount | False   | Una cuenta de usuario que el sistema admini... |

NOTA: Una cosa, PowerShell no es case sensitive, es decir, no distingue mayúsculas de minúsculas, es decir, puedes poner un comando con mayúsculas o minúsculas o puedes acceder a un archivo de igual manera

- `get-date` (Para saber la fecha actual del sistema)

A screenshot of a PowerShell window showing the output of the `get-date` command. The output shows the date and time as "miércoles, 27 de abril de 2022 21:51:28".

- `clear-host` (Para limpiar la pantalla)
- `get-location` (Para saber mi directorio actual)

```
master # +166 -6 | +7 12:14:21 PM 43ms
JLoel get-location

Path
-----
C:\Users\JLoel
```

- `get-childItem` (Para ver los ficheros y carpetas que hay en nuestra ubicación actual o lo que especifiquemos nosotros)

```
master # +166 -6 | +7 12:26:56 PM 3ms
JLoel get-childitem

Directory: C:\Users\JLoel

Mode LastWriteTime Length Name
---- ----- ----- -----
d--- 14/03/2022 0:00 .afirma
d--- 30/03/2020 17:49 .android
d--- 13/09/2021 12:26 .cache
d--- 30/12/2021 1:31 .config
d--- 12/04/2022 22:42 .dotnet
d--- 21/06/2020 17:28 .eclipse
```

- `get-netadapter` (Para visualizar los adaptadores de red)

| Name                         | InterfaceDescription                      | ifIndex |
|------------------------------|---|---------|
| VMware Network Adapter ... 8 | VMware Virtual Ethernet Adapter for VM... | 30      |
| Wi-Fi                        | Intel(R) Centrino(R) Advanced-N 6235      | 29      |
| vEthernet (WSL)              | Hyper-V Virtual Ethernet Adapter #5       | 81      |
| Conexión de red Bluetooth    | Bluetooth Device (Personal Area Network)  | 19      |
| vEthernet (VMware Netw...)   | Hyper-V Virtual Ethernet Adapter #2       | 35      |
| VMware Network Adapter ... 1 | VMware Virtual Ethernet Adapter for VM... | 13      |
| Ethernet                     | Intel(R) 82579LM Gigabit Network Conne... | 10      |
| vEthernet (Ethernet)         | Hyper-V Virtual Ethernet Adapter          | 9       |
| vEthernet (Wi-Fi)            | Hyper-V Virtual Ethernet Adapter #4       | 42      |
| vEthernet (VMware Net ... 2  | Hyper-V Virtual Ethernet Adapter #3       | 80      |

- `get-service` (Para ver cuáles son los procesos en segundo plano que se están ejecutando en nuestra máquina)

| Status  | Name                 | DisplayName                              |
|---------|----------------------|--|
| Stopped | AarSvc_2536caaee     | Agent Activation Runtime_2536caaee       |
| Stopped | AdobeUpdateService   | AdobeUpdateService                       |
| Stopped | AGMService           | Adobe Genuine Monitor Service            |
| Stopped | AGSService           | Adobe Genuine Software Integrity Serv... |
| Stopped | AJRouter             | Servicio de enrutador de AllJoyn         |
| Stopped | ALG                  | Servicio de puerta de enlace de nivel... |
| Stopped | AppIDSvc             | Identidad de aplicación                  |
| Running | Appinfo              | Información de la aplicación             |
| Stopped | AppMgmt              | Administración de aplicaciones           |
| Stopped | AppReadiness         | Preparación de aplicaciones              |
| Stopped | AppVClient           | Microsoft App-V Client                   |
| Running | AppXSvc              | Servicio de implementación de AppX (A... |
| Stopped | AssignedAccessMan... | Servicio AssignedAccessManager           |

Por ejemplo si quisiéramos un servicio en concreto, podríamos hacerlo especificando el nombre del campo como parámetro y a continuación el nombre del servicio.

| Status  | Name    | DisplayName     |
|---------|---------|-----------------|
| Stopped | w32time | Hora de Windows |

Digamos que estos son los comandos que conocemos. Entonces imaginaros que queremos usar un comando y no se cómo se escribe exactamente.

## 2.5. La ayuda en PowerShell

### 2.5.1. Obtener Comandos con get-command y get-module

Si yo pongo el comando `get-command` lo que conseguimos es visualizar todos los comandos que tiene la PowerShell.

Terminal window showing the output of the `get-command` command. The output lists various cmdlets categorized by CommandType (Alias) and their names, along with their versions.

| CommandType | Name                                     | Version |
|-------------|--|---------|
| Alias       | Add-AppPackage                           | 2.0...  |
| Alias       | Add-AppPackageVolume                     | 2.0...  |
| Alias       | Add-AppProvisionedPackage                | 3.0     |
| Alias       | Add-ProvisionedAppPackage                | 3.0     |
| Alias       | Add-ProvisionedAppSharedPackageContainer | 3.0     |
| Alias       | Add-ProvisionedAppxPackage               | 3.0     |
| Alias       | Add-ProvisioningPackage                  | 3.0     |
| Alias       | Add-TrustedProvisioningCertificate       | 3.0     |
| Alias       | Apply-WindowsUnattend                    | 3.0     |

**Ejemplo:** Estoy buscando un comando que tiene que ver con un verbo determinado

`get-command -verb new` (Busca aquellos comandos que tengan que ver con el verbo new)

Terminal window showing the output of the `get-command -verb new` command. The output lists various cmdlets categorized by CommandType (Function) and their names, along with their versions.

| CommandType | Name                      | Version |
|-------------|---------------------------|---------|
| Function    | New-AutologgerConfig      | 1.0...  |
| Function    | New-DAEntryPointTableItem | 1.0...  |
| Function    | New-DscChecksum           | 1.1     |
| Function    | New-EapConfiguration      | 2.0...  |
| Function    | New-EtwTraceSession       | 1.0...  |
| Function    | New-FileShare             | 2.0...  |
| Function    | New-Fixture               | 3.4...  |
| Function    | New-GitPromptSettings     | 1.0...  |
| Function    | New-IscsiTargetPortal     | 1.0...  |

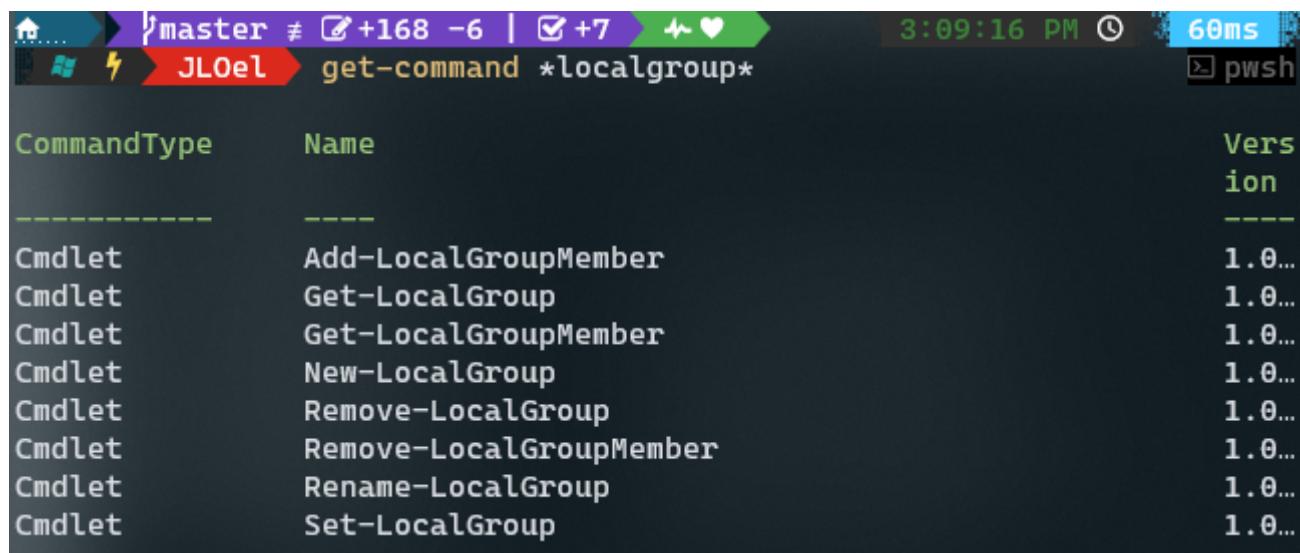
De igual manera podemos buscar los comandos que tienen que ver con un nombre en concreto

`get-command -noun localgroup` (Muestra aquellos comandos que tengan el nombre localgroup)

Terminal window showing the output of the `get-command -noun localgroup` command. The output lists various cmdlets categorized by CommandType (Cmdlet) and their names, along with their versions.

| CommandType | Name              | Version |
|-------------|-------------------|---------|
| Cmdlet      | Get-LocalGroup    | 1.0...  |
| Cmdlet      | New-LocalGroup    | 1.0...  |
| Cmdlet      | Remove-LocalGroup | 1.0...  |
| Cmdlet      | Rename-LocalGroup | 1.0...  |
| Cmdlet      | Set-LocalGroup    | 1.0...  |

Y si quisiera buscar absolutamente todos los comandos que incluyan la palabra localgroup haríamos lo siguiente



A screenshot of a Windows terminal window titled 'pwsh'. The command entered is 'get-command \*localgroup\*'. The output is a table with columns: CommandType, Name, and Version. The table lists eight cmdlets related to local groups.

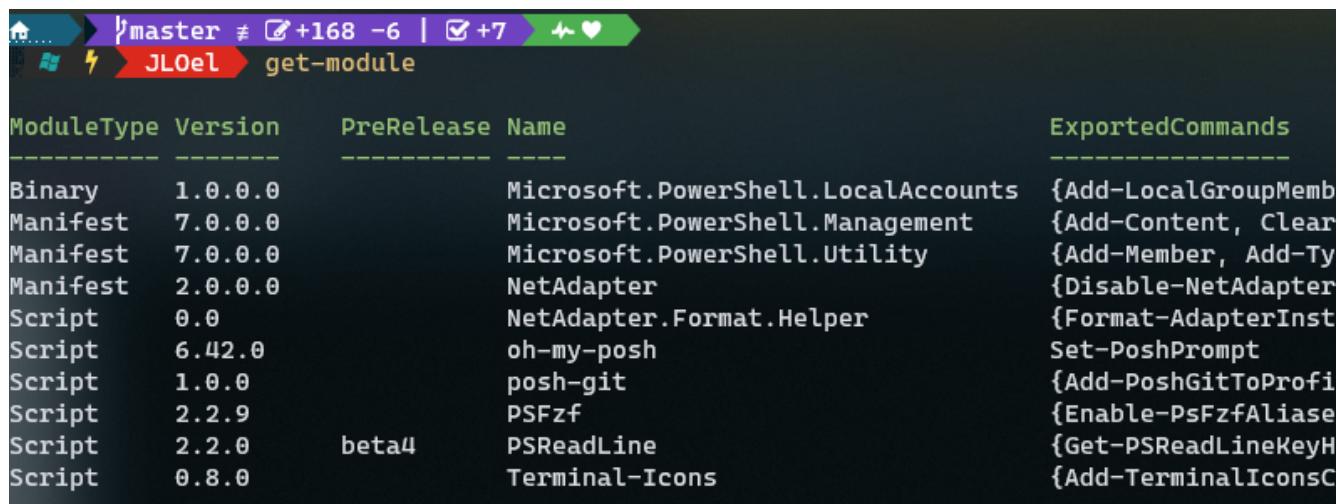
| CommandType | Name                    | Version |
|-------------|-------------------------|---------|
| Cmdlet      | Add-LocalGroupMember    | 1.0...  |
| Cmdlet      | Get-LocalGroup          | 1.0...  |
| Cmdlet      | Get-LocalGroupMember    | 1.0...  |
| Cmdlet      | New-LocalGroup          | 1.0...  |
| Cmdlet      | Remove-LocalGroup       | 1.0...  |
| Cmdlet      | Remove-LocalGroupMember | 1.0...  |
| Cmdlet      | Rename-LocalGroup       | 1.0...  |
| Cmdlet      | Set-LocalGroup          | 1.0...  |

Ahora vamos a ver el concepto de módulo.

### Concepto de Módulo:

Un módulo no es nada más que un conjunto de comandos

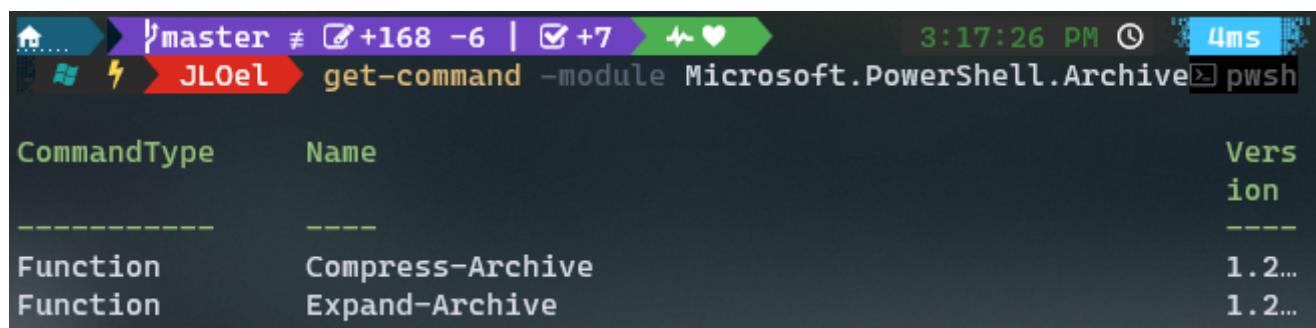
Para ver los módulos que hay en el sistema ejecutamos el siguiente comando `get-module`



A screenshot of a Windows terminal window titled 'pwsh'. The command entered is 'get-module'. The output is a table with columns: ModuleType, Version, PreRelease, Name, and ExportedCommands. The table lists several modules and their associated commands.

| ModuleType | Version | PreRelease | Name                               | ExportedCommands       |
|------------|---------|------------|------------------------------------|------------------------|
| Binary     | 1.0.0.0 |            | Microsoft.PowerShell.LocalAccounts | {Add-LocalGroupMemb... |
| Manifest   | 7.0.0.0 |            | Microsoft.PowerShell.Management    | {Add-Content, Clear... |
| Manifest   | 7.0.0.0 |            | Microsoft.PowerShell.Utility       | {Add-Member, Add-Ty... |
| Manifest   | 2.0.0.0 |            | NetAdapter                         | {Disable-NetAdapter... |
| Script     | 0.0     |            | NetAdapter.Format.Helper           | {Format-AdapterInst... |
| Script     | 6.42.0  |            | oh-my-posh                         | Set-PoshPrompt         |
| Script     | 1.0.0   |            | posh-git                           | {Add-PoshGitToProf...  |
| Script     | 2.2.9   |            | PSFzf                              | {Enable-PsFzfAliase... |
| Script     | 2.2.0   | beta4      | PSReadLine                         | {Get-PSReadLineKeyH... |
| Script     | 0.8.0   |            | Terminal-Icons                     | {Add-TerminalIconsC... |

Si queremos ver los comandos que tiene un determinado módulo pondríamos `get-command -module <nombre_modulo>`



A screenshot of a Windows terminal window titled 'pwsh'. The command entered is 'get-command -module Microsoft.PowerShell.Archive'. The output is a table with columns: CommandType, Name, and Version. The table lists two functions related to archive management.

| CommandType | Name             | Version |
|-------------|------------------|---------|
| Function    | Compress-Archive | 1.2...  |
| Function    | Expand-Archive   | 1.2...  |

Si queremos saber cuáles son los módulos que se encuentran disponibles ejecutamos el comando

```
get-module -ListAvailable
```

The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a header bar with icons for file, edit, and search, followed by the text 'master' and some status indicators. Below the header, the command 'get-module -ListAvailable' is typed. The output lists several modules:

| ModuleType | Version | PreRelease | Name           |
|------------|---------|------------|----------------|
| Script     | 6.42.0  |            | oh-my-posh     |
| Script     | 1.0.0   |            | posh-git       |
| Script     | 2.2.9   |            | PSFzf          |
| Script     | 2.2.0   | beta4      | PSReadLine     |
| Script     | 0.8.0   |            | Terminal-Icons |
| Script     | 1.1.13  |            | z              |

At the bottom of the output, it says 'Directory: C:\Users\JLOel\OneDrive\Documentos\PowerShell\Modules'. The prompt at the bottom of the terminal is 'Directory: C:\program files\powershell\7\Modules'.

Digamos que queremos trabajar con los comandos del módulo BitLocker que no están en memoria, ejecutamos el comando `import-module bitlocker`

```

master # +168 -6 | +7 3:23:52 PM 3ms
Windows PowerShell [JLoel] import-module bitlocker
WARNING: The names of some imported commands from the module 'bitlocker' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the Verbose parameter. For a list of approved verbs, type Get-Verb.

master # +168 -6 | +7 3:23:59 PM 541ms
Windows PowerShell [JLoel] import-module bitlocker -verbose
VERBOSE: Loading module from path 'C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules\bitlocker\bitlocker.psd1'.
VERBOSE: Importing function 'Add-BitLockerKeyProtector'.
VERBOSE: Importing function 'Backup-BitLockerKeyProtector'.
WARNING: The names of some imported commands from the module 'bitlocker' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run the Import-Module command again with the Verbose parameter. For a list of approved verbs, type Get-Verb.

VERBOSE: The 'BackupToAAD-BitLockerKeyProtector' command in the bitlocker module was imported, but because its name does not include an approved verb, it might be difficult to find. For a list of approved verbs, type Get-Verb.

VERBOSE: Importing function 'BackupToAAD-BitLockerKeyProtector'.
VERBOSE: Importing function 'Clear-BitLockerAutoUnlock'.
VERBOSE: Importing function 'Disable-BitLocker'.
VERBOSE: Importing function 'Disable-BitLockerAutoUnlock'.
VERBOSE: Importing function 'Enable-BitLocker'.
VERBOSE: Importing function 'Enable-BitLockerAutoUnlock'.
VERBOSE: Importing function 'Get-BitLockerVolume'.
VERBOSE: Importing function 'Lock-BitLocker'.
VERBOSE: Importing function 'Remove-BitLockerKeyProtector'.
VERBOSE: Importing function 'Resume-BitLocker'.
VERBOSE: Importing function 'Suspend-BitLocker'.
VERBOSE: Importing function 'Unlock-BitLocker'.

```

Ahora comprobamos que el módulo ha sido importado correctamente, ejecutamos nuevamente el comando `get-module`

| ModuleType | Version | PreRelease | Name                               | ExportedCommands   |
|------------|---------|------------|------------------------------------|--|
| Script     | 1.0.0.0 |            | bitlocker                          | {Add-BitLockerKeyProtector, Compress-Archive, Add-LocalGroupMember, Add-Content, Clear-Content, Remove-Content, Set-Content, Add-Member, Add-Type, Remove-Member, Remove-Type, Format-Adapter, Set-PoshPrompt, Get-PSReadLineKey, Add-TerminalIcons} |
| Manifest   | 1.2.5   |            | Microsoft.PowerShell.Archive       |  |
| Binary     | 1.0.0.0 |            | Microsoft.PowerShell.LocalAccounts |  |
| Manifest   | 7.0.0.0 |            | Microsoft.PowerShell.Management    | {Add-Content, Clear-Content, Remove-Content, Set-Content, Add-Member, Add-Type, Remove-Member, Remove-Type, Format-Adapter, Set-PoshPrompt, Get-PSReadLineKey, Add-TerminalIcons}  |
| Manifest   | 7.0.0.0 |            | Microsoft.PowerShell.Utility       |  |
| Manifest   | 2.0.0.0 |            | NetAdapter                         | {Disable-NetAdapter, Format-Adapter, Set-PoshPrompt, Get-PSReadLineKey, Add-TerminalIcons}   |
| Script     | 0.0     |            | NetAdapter.Format.Helper           |  |
| Script     | 6.42.0  |            | oh-my-posh                         |  |
| Script     | 1.0.0   |            | posh-git                           | {Add-PoshGitToProfile, Enable-PsfzfAlias}  |
| Script     | 2.2.9   |            | PSFzf                              |  |
| Script     | 2.2.0   | beta4      | PSReadLine                         | {Get-PSReadLineKey, Add-TerminalIcons}   |
| Script     | 0.8.0   |            | Terminal-Icons                     |  |

El módulo bitlocker ya está disponible en memoria, por lo tanto, ya podemos trabajar con sus comandos. Si quisiéramos eliminar dicho módulo solo cambiamos el verbo get por remove y especificamos el nombre del módulo el cual queremos eliminar

```
remove-module bitlocker
```

Y comprobamos que ya no está con el comando get-module

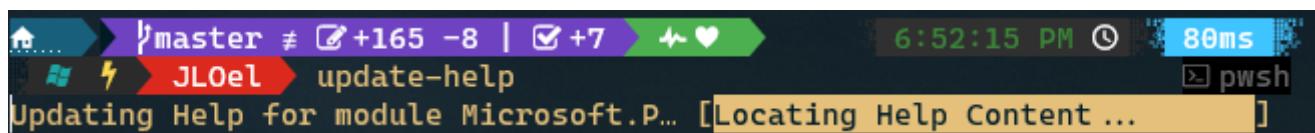
| ModuleType | Version | PreRelease | Name                               | ExportedCommands       |
|------------|---------|------------|------------------------------------|------------------------|
| Binary     | 1.0.0.0 |            | Microsoft.PowerShell.LocalAccounts | {Add-LocalGroupMember} |
| Manifest   | 7.0.0.0 |            | Microsoft.PowerShell.Management    | {Add-Content, Clear    |
| Manifest   | 7.0.0.0 |            | Microsoft.PowerShell.Utility       | {Add-Member, Add-Ty    |
| Manifest   | 2.0.0.0 |            | NetAdapter                         | {Disable-NetAdapter}   |
| Script     | 0.0     |            | NetAdapter.Format.Helper           | {Format-AdapterInst    |
| Script     | 6.42.0  |            | oh-my-posh                         | Set-PoshPrompt         |
| Script     | 1.0.0   |            | posh-git                           | {Add-PoshGitToProfi    |
| Script     | 2.2.9   |            | PSFzf                              | {Enable-PsFzfAliase    |
| Script     | 2.2.0   | beta4      | PSReadLine                         | {Get-PSReadLineKeyH    |
| Script     | 0.8.0   |            | Terminal-Icons                     | {Add-TerminalIconsC    |

## 2.5.2. Obtener ejemplos del uso de Comandos con get-help

¿Y ahora qué pasa si queremos buscar información e incluso ejemplos del uso de un comando?

Entonces necesitamos la ayuda de PowerShell.

La ayuda en PowerShell es muy completa y trae muchos ejemplos, necesitamos acceso a internet para d  
Para actualizar la ayuda de PowerShell necesitamos el comando update-help, para tener incorporado l



Existen tipos de ayuda en PowerShell:

- Ayuda estándar: get-help <nombre\_comando>
- Ayuda con ejemplos: get-help <nombre\_comando> -examples
- Ayuda con ejemplos y más detalles: get-help <nombre\_comando> -detailed
- Ayuda Completa: get-help <nombre\_comando> -full
- Ayuda Online: get-help <nombre\_comando> -online

Vamos a verlo sobre la marcha, por ejemplo necesitamos crear la cuenta de un usuario, pero no sabemos nada sobre la sintaxis de ese comando.

Pues usamos lo siguiente. get-help new-localuser

## New-LocalUser

### SYNOPSIS

Creates a local user account.

### SYNTAX

```
New-LocalUser [-Name] <System.String> [-AccountExpires <System.DateTime>] [-Account  
[-FullName <System.String>] -NoPassword [-UserMayNotChangePassword] [-Confirm] [-Wh
```

```
New-LocalUser [-Name] <System.String> [-AccountExpires <System.DateTime>] [-Account  
[-FullName <System.String>] -Password <System.Security.SecureString> [-PasswordNever  
[<CommonParameters>]
```

### DESCRIPTION

The `New-LocalUser` cmdlet creates a local user account. This cmdlet creates a local user account on a Microsoft account.

> [!NOTE] > The `Microsoft.PowerShell.LocalAccounts` module is not available in 32-bit PowerShell.

### RELATED LINKS

Online Version:

<https://docs.microsoft.com/powershell/module/microsoft.powershell.localaccounts/new-localuser>

Disable-LocalUser

Enable-LocalUser

Get-LocalUser

Remove-LocalUser

Rename-LocalUser

Set-LocalUser

### REMARKS

To see the examples, type: "Get-Help New-LocalUser -Examples"

For more information, type: "Get-Help New-LocalUser -Detailed"

For technical information, type: "Get-Help New-LocalUser -Full"

For online help, type: "Get-Help New-LocalUser -Online"

Digamos que aun no me he enterado bien sobre el uso del comando, voy a hacer que me una ayuda a través de los ejemplos.

```
get-help new-localuser -examples
```

```
master # +168 -6 | +7 JL0el get-help new-localuser -examples

NAME
  New-LocalUser

SYNOPSIS
  Creates a local user account.

----- Example 1: Create a user account -----

PS C:\> New-LocalUser -Name "User02" -Description "Description of this account." -NoPassword
Name   Enabled Description
----   ----- -----
User02 True    Description of this account.

This command creates a local user account and does not specify the AccountExpires or Password p
expire or have a password by default.

----- Example 2: Create a user account that has a password ----

PS C:\> $Password = Read-Host -AsSecureString
PS C:\> New-LocalUser "User03" -Password $Password -FullName "Third User" -Description "Descrip
Name   Enabled Description
----   ----- -----
User03 True    Description of this account.

The first command prompts you for a password by using the 'Read-Host' cmdlet. The command store
'$Password' variable.
```

NOTA: Como crear o eliminar un usuario es una operación que necesita permiso de administrador  
tenemos que abrir nuestra PowerShell como administrador

### 2.5.3. Atajos y Alias

- **Atajos:**

La mayoría de los administradores quieren escribir los comandos lo más rápido posible, para ello usan los tabuladores. ¿Qué hace el tabulador? Nos completa el comando que estamos escribiendo, si hay más de una opción podemos verla con los cursores.

- **Cursor:**

Nos permite seleccionar un comando ejecutado anteriormente.

- **Historial:**

Otra función más interesante es el historial, entonces si yo digo dame el historial de todos los comandos que he ejecutado, usamos el comando `get-history`

```
master ⚡+169 -6 | ✅+7 ➔ JL0el ➔ get-history

Id      Duration CommandLine
--      -----
1       0.043 clear
2       0.027 get-location
3       0.014 pwd
4       0.059 date
5       0.008 clear
6       0.298 get-localgroup
7       0.003 clear
8       0.007 cls
9       0.345 get-children
10      4.083 get-item
11      0.084 get-chilitem
12      0.627 get-childitem
13      0.147 dir
14      0.179 ls
15      0.056 ls clear
16      0.003 clear
```

Por ejemplo si yo ejecuto `get-process` para saber todos los procesos que se están ejecutando ahora

```
master ⚡+169 -6 | ✅+7 ➔ JL0el ➔ get-process

NPM(K)    PM(M)    WS(M)    CPU(s)    Id    SI ProcessName
----      ----      ----      -----    --    -- -----
21        9,31     4,66     0,36     22252  13 AdobeNotificationClient
18        7,57     16,00    10,95    3512   0 AppHelperCap
24        10,99    23,27    6,66     18072  13 ApplicationFrameHost
8         1,55     7,25     0,03     16628  0 AppVShNotify
12        6,54     13,98    1.052,45  21592  0 audiodg
16        4,16     20,78    0,20     17848  13 backgroundTaskHost
16        23,92    20,62    5,75     1124   13 chrome
```

Vemos que sí se ha guardado en el historial

```
master # +169 -6 | +7 ↗ ↘
JLoel > get-history

Id Duration CommandLine
-- -----
1 0.043 clear
2 0.027 get-location
3 0.014 pwd
4 0.059 date
5 0.008 clear
6 0.298 get-localgroup
7 0.003 clear
8 0.007 cls
9 0.345 get-children
10 4.083 get-item
11 0.084 get-chilitem
12 0.627 get-childitem
13 0.147 dir
14 0.179 ls
15 0.056 ls clear
16 0.003 clear
137 0.102 get-history
138 0.010 clear
139 0.202 get-process
```

Si quisieramos ejecutar por ejemplo el comando que está en el historial en la línea 2 lo hacemos con el comando `invoke-history <posición_historial>`

```
master # +169 -6 | +7 ↗ ↘
JLoel > invoke-history 2
get-location

Path
-----
C:\Users\JLoel
```

Pero veo que es escribir mucho, mirar con tan solo escribir `h` nos muestra el historial de comandos

| <b>Id</b> | <b>Duration</b> | <b>CommandLine</b> |
|-----------|-----------------|--------------------|
| 1         | 0.043           | clear              |
| 2         | 0.027           | get-location       |
| 3         | 0.014           | pwd                |
| 4         | 0.059           | date               |
| 5         | 0.008           | clear              |
| 6         | 0.298           | get-localgroup     |
| 7         | 0.003           | clear              |
| 8         | 0.007           | cls                |
| 9         | 0.345           | get-children       |
| 10        | 4.083           | get-item           |
| 11        | 0.084           | get-chilitem       |
| 12        | 0.627           | get-childitem      |
| 13        | 0.147           | dir                |

Por ejemplo si quiero ejecutar nuevamente el comando de la línea 2 del historial de comandos podría hacerlo solo con este comando

| <b>Path</b>    |
|----------------|
| C:\Users\JLOel |

Es una forma muy abreviada y fácil de buscar y ejecutar comandos del historial.

Una utilidad que uso mucho es buscar un comando del historial con **ctrl+R**

Por ejemplo si quisiera buscar un comando relacionado con **new** , y cada vez que pulso **ctrl+R** me van apareciendo los comandos. Es una forma bastante fácil y útil de buscar un comando en el historial de ayuda.

|                                |
|--------------------------------|
| get-help new-localuser -online |
| bck-i-search: new_             |

Si quisieramos eliminar todo el historial de comandos usamos el comando **clear-history**

```

master # +169 -6 | +7
JLoel clear-history
master # +169 -6 | +7
JLoel get-history

Id      Duration CommandLine
--      -----
149    0.021 clear-history

```

Tanto los historiales, los tabuladores y los cursores nos pueden ayudar a escribir los comandos de una forma más rápida.

- **Alias:**

El alias no es nada más que un apodo o un sobrenombre para referirse a un Cmdlet

Vamos a ver los alias que tiene el sistema con el comando `get-alias`

```

master # +169 -6 | +7
JLoel get-alias

 CommandType      Name          Version   Source
-----      ----
 Alias        ? → Where-Object
 Alias        % → ForEach-Object
 Alias        ac → Add-Content
 Alias        algm → Add-LocalGroupMember
 Alias        cat → Get-Content
 Alias        cd → Set-Location
 Alias        chdir → Set-Location
 Alias        clc → Clear-Content
 Alias        clear → Clear-Host
 Alias        clhy → Clear-History
 Alias        cli → Clear-Item
 Alias        clp → Clear-ItemProperty
 Alias        cls → Clear-Host
 Alias        clv → Clear-Variable
 Alias        cnsn → Connect-PSSession
 Alias        compare → Compare-Object
 Alias        copy → Copy-Item
 Alias        cp → Copy-Item

```

Como podemos observar muchos de los alias son iguales que los comandos que se usan la shell de Linux

Digamos que queremos saber si un comando tiene alias solo necesitamos este comando `get-alias -Definition <Cmdlet>`

```

master # +169 -6 | +7
JLoel get-alias -Definition Get-ChildItem

 CommandType      Name
-----      ----
 Alias        dir → Get-ChildItem
 Alias        gci → Get-ChildItem
 Alias        ls → Get-ChildItem

```

Voy a usar el que más se parece a Linux, el `ls`

| Mode  | LastWriteTime    | Length | Name     |
|-------|------------------|--------|----------|
| d---- | 14/03/2022 0:00  |        | .afirma  |
| d---- | 30/03/2020 17:49 |        | .android |
| d---- | 13/09/2021 12:26 |        | .cache   |
| d---- | 30/12/2021 1:31  |        | .config  |

Y así existen un montón de alias.

Por ejemplo voy a crear un archivo y luego quisiera ver su contenido. Con el alias es bastante fácil

```
PS C:\Users\JLOel> echo "Hola Mundo" > fichero.txt
PS C:\Users\JLOel> cat fichero.txt
Hola Mundo
```

Pero sin Alias sería algo así:

```
PS C:\Users\JLOel> new-item fichero.txt

Directory: C:\Users\JLOel\OneDrive\Escritorio\CarpetaPruebaFC

Mode          LastWriteTime        Length Name
----          -----        ---- 
-a--- 28/04/2022 20:06           0  fichero.txt

PS C:\Users\JLOel> Write-Output "Hola Mundo" >.\fichero.txt
PS C:\Users\JLOel> get-content .\fichero.txt
Hola Mundo
```

## 2.6. Gestión de archivos y carpetas

Ahora vamos a ver el tema de gestión de archivos y carpetas.

- Veremos los principales comandos para trabajar con archivos y carpetas.
- También vamos a hacer uso de los alias.

Existen 3 comandos fundamentales que son.

- `Get-Location (pwd)`: nos devuelve la ruta o path en la que nos encontramos.
- `Set-Location (cd)`: es para desplazarnos por la estructura de directorios.

- `Get-ChildItem (ls)`: nos permite mostrar el contenido de un directorio.
  - Cuando mostramos el contenido de un directorio en el campo Mode nos aparecen unas letras.
    - d: directorio
    - a: archivo
    - s: archivo del sistema
    - h: archivo oculto
    - r: lectura
    - w: escritura
    - x: ejecución

```

Path
-----
C:\Users\JLoel\OneDrive\Escritorio

Path
-----
C:\Users\JLoel\OneDrive\Escritorio
JLoel set-location .\CarpetaPruebaFC\
Path
-----
C:\...\CarpetaPruebaFC
JLoel get-childItem

Directory: C:\Users\JLoel\OneDrive\Escritorio\CarpetaPruebaFC

Mode           LastWriteTime        Length Name
----           -----          ----
-a--- 28/04/2022 13:20            9106 Alumnos.xlsx
-a--- 28/04/2022 13:53            1068 Celestina.txt
-a--- 28/04/2022 13:35            3504 CSV.csv
-a--- 21/11/2019 11:06           67246 EjerciciosBD.pdf
-a--- 28/04/2022 20:06             12 fichero.txt
-a--- 22/10/2020 15:26          1068951 SERVIDOR_DHCP.docx
-a--- 28/04/2022 13:25           12037 XML.xlsx
```

Digamos que queremos ver los archivos ocultos que hay en tu sistema, normalmente los archivos ocultos suelen estar en la raíz para que no puedan entrar los virus.

```

    \...\...\CarpetaPruebaFC ➤ master ✘+169 -6 | ✘+7 ↵+♥ ➤
    JL0el ➤ get-childItem c:\ -Attributes hidden

    Directory: C:\


```

| Mode  | LastWriteTime | Length | Name                                    |
|-------|---------------|--------|---|
| d--h- | 23/08/2020    | 22:14  | _acestream_cache_                       |
| d--h- | 24/09/2020    | 11:56  | \$AV_ASW                                |
| d--hs | 26/09/2020    | 10:15  | \$Recycle.Bin                           |
| d--h- | 31/05/2021    | 20:02  | \$SysReset                              |
| d--h- | 04/01/2022    | 16:38  | \$Windows.~WS                           |
| d--h- | 13/04/2022    | 3:03   | \$WinREAgent                            |
| l--hs | 21/09/2019    | 5:59   | Archivos de programa ➔ C:\Program Files |
| d--hs | 28/04/2022    | 2:19   | Config.Msi                              |
| l--hs | 21/09/2019    | 5:59   | Documents and Settings ➔ C:\Users       |
| d--hs | 13/05/2020    | 15:09  | MSOCache                                |
| d--h- | 23/03/2020    | 8:12   | OneDriveTemp                            |
| d--h- | 27/04/2022    | 23:14  | ProgramData                             |
| d--hs | 13/03/2022    | 23:44  | Recovery                                |
| d--hs | 28/04/2022    | 3:47   | System Volume Information               |
| -a-hs | 17/04/2022    | 15:33  | 12288 DumpStack.log.tmp                 |
| -a-hs | 27/04/2022    | 17:11  | 3378712576 hiberfil.sys                 |
| -a-hs | 28/04/2022    | 1:59   | 7938310144 pagefile.sys                 |
| -a-hs | 17/04/2022    | 15:33  | 167777216 swapfile.sys                  |

A algunos a lo mejor les aparezca el gestor de arranque de Windows bootmgr, el pagefile es donde están los archivos de paginación.

- New-Item :
- Nos permite crear archivos y directorios
- Alias:
  - ni :arrow\_right:Archivos
  - md :arrow\_right:Directorios

Vamos a crear un archivo y un directorio

```

    \...\...\CarpetaPruebaFC ➤ master ✘+169 -6 | ✘+7 ↵+♥ ➤
    JL0el ➤ new-item carpeta -itemtype Directory

    Directory: C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC


```

| Mode  | LastWriteTime | Length | Name    |
|-------|---------------|--------|---------|
| d---- | 28/04/2022    | 21:15  | carpeta |

```

    \...\...\CarpetaPruebaFC ➤ master ✘+169 -6 | ✘+7 ↵+♥ ➤
    JL0el ➤ new-item archivo -itemtype File

    Directory: C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC


```

| Mode  | LastWriteTime | Length | Name    |
|-------|---------------|--------|---------|
| -a--- | 28/04/2022    | 21:15  | archivo |

¿Si quisieramos eliminarlos?

- remove-item :
  - Permite eliminar archivos o carpetas
  - Alias:
    - rm:arrow\_right:Archivos y carpetas

```

C:\...\...\CarpetaPruebaFC> remove-item .\carpeta\
C:\...\...\CarpetaPruebaFC> remove-item .\archivo
C:\...\...\CarpetaPruebaFC> remove-item .\carpeta -Recurse
  
```

¿Qué pasaría si el directorio que queremos eliminar tiene archivos dentro?

Solo con el comando el sistema nos preguntaría si queremos borrar la carpeta. Para que no nos pregunte usamos el parámetro Recurse

Hacemos una prueba, creamos archivos dentro de un directorio que crearemos ahora.

- Creamos la carpeta. new-item carpeta -itemType Directory (Alias: md carpeta)
- Nos situamos en la carpeta: set-location carpeta (Alias: cd carpeta)
- Creamos el archivo: new-item archivo -itemType File (Alias: ni archivo)
- Nos desplazamos una posición atrás de la carpeta: set-Location .. (Alias: cd ..)
  - Eliminamos la carpeta sin el parámetro Recurse : remove-item carpeta (rmdir carpeta). Nos salta una alerta de que vamos a eliminar el contenido de la carpeta.
  - Eliminamos la carpeta con el parámetro Recurse : remove-item carpeta -Recurse (rmdir -r carpeta). No nos salta nada.

```

..\..\..\CarpetaPruebaFC > master # +169 -6 | +7 > JL0el new-item carpeta -itemType Directory

Directory: C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC

Mode          LastWriteTime      Length Name
----          -----          ---- 
d--- 28/04/2022    21:35           carpeta

..\..\..\CarpetaPruebaFC > master # +169 -6 | +7 > JL0el set-location carpeta
..\..\..\carpeta > master # +169 -6 | +7 > JL0el new-item archivo -itemType File

Directory: C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\carpeta

Mode          LastWriteTime      Length Name
----          -----          ---- 
-a-- 28/04/2022    21:36           archivo

..\..\..\carpeta > master # +169 -6 | +7 > JL0el remove-item carpeta

Confirm
The item at C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\carpeta has children and the Recurse all children will be removed with the item. Are you sure you want to continue?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
..\..\..\CarpetaPruebaFC > master # +169 -6 | +7 > JL0el remove-item carpeta -Recurse
```

Si intentamos hacer un listado del directorio eliminado nos saltará un error.

```

..\..\..\CarpetaPruebaFC > master # +169 -6 | +7 > JL0el get-childItem carpeta
Get-ChildItem: Cannot find path 'C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\carpeta' because it does not exist.
..\..\..\CarpetaPruebaFC > master # +169 -6 | +7 >
```

Vamos a ver ahora otro comando que sirve para mover elementos, objetos de un sitio a otro

- move-item
  - Permite mover elementos u objetos de un sitio a otro
  - Alias:
    - mv: Archivos y Directorios
    - mi: Archivos y Directorios Ejemplo: crearemos unos cuantos archivos, luego crearemos una carpeta y dichos archivos las moveremos a la carpeta creada.
  - Pasos:
    - Creamos el|los archivos: new-item <nombre\_archivo> (sin especificar el tipo de archivo con el parámetro itemType, por defecto estaríamos creando un archivo)
    - Creamos la carpeta en la cual moveremos dichos archivos: new-item <nombre\_carpeta> -itemType Directory
    - Movemos los archivos a la carpeta: move-item <archivoCreado> <carpetaCreada>
    - Podemos comprobar que los archivos han sido movidos correctamente de tres maneras:
      - a. Pasando al comando get-childItem la ruta relativa de carpeta
      - b. Pasando al comando get-childItem la ruta absoluta de carpeta

c. O bien situándonos primero a la ubicación de la carpeta `set-location <carpetaCreada>` y luego ejecutar el comando `get-childItem` sin especificar ninguna ruta

```

master ✘ +168 -6 | ↴ +7 ➜ 3:54:56 PM ⏱ 2ms pwsh
JLoel new-item foto1.jpg, foto2.jpg

Directory: C:\Users\JLoel

Mode           LastWriteTime      Length Name
----           -----          ---- 
-a---        29/04/2022     15:55      0 foto1.jpg
-a---        29/04/2022     15:55      0 foto2.jpg

master ✘ +170 -6 | ↴ +7 ➜ 3:55:10 PM ⏱ 25ms pwsh
JLoel new-item fotos -itemType Directory

Directory: C:\Users\JLoel

Mode           LastWriteTime      Length Name
----           -----          ---- 
d---        29/04/2022     15:55      0 fotos

master ✘ +170 -6 | ↴ +7 ➜ 3:55:24 PM ⏱ 23ms pwsh
JLoel move-item foto?.jpg fotos
master ✘ +169 -6 | ↴ +7 ➜ 3:56:06 PM ⏱ 45ms pwsh
JLoel get-childItem fotos

Directory: C:\Users\JLoel\fotos

Mode           LastWriteTime      Length Name
----           -----          ---- 
-a---        29/04/2022     15:55      0 foto1.jpg
-a---        29/04/2022     15:55      0 foto2.jpg

```

¿Y ahora en vez de mover un archivo quisiéramos copiarlo?

- **`copy-item`**

- Copia un archivo o carpeta
- Alias:
  - `cp`
  - `copy` Por ejemplo:
- Pasos:
  - Creamos una nueva carpeta en la cual moveremos ellos archivos: `new-item <nombre_carpeta> -itemType Directory`
  - Hacemos la copia:
    - Copiar una carpeta a otra: `copy-item <carpeta_Creada> <nuevaCarpeta> -Recurse` (Parámetro `Recurse` es para que también se copie el contenido de la carpeta origen a la carpeta destino)

```

master ⚡ +169 -6 | ✅ +7 4:00:29 PM 2ms
JL0el copy-item fotos .\Download -Recurse pwsh
master ⚡ +170 -6 | ✅ +7 4:04:11 PM 29ms
JL0el dir .\Download\ pwsh

Directory: C:\Users\JL0el\Download

Mode LastWriteTime Length Name
---- ----- ---- -
d--- 29/04/2022 16:04   fotos

```

NOTA: Si intentamos copiar una carpeta a otra que no existe, esa se creará automáticamente con el contenido de la carpeta copiada

Ejemplo:

```

master ⚡ +173 -6 | ✅ +7 5:32:34 PM 2ms
JL0el copy-item .\recuerdos\ usos -Recurse pwsh
master ⚡ +174 -6 | ✅ +7 5:32:49 PM 22ms
JL0el dir usos pwsh

Directory: C:\Users\JL0el\usos

Mode LastWriteTime Length Name
---- ----- ---- -
-a-- 29/04/2022 15:55 0 foto2.jpg
-a-- 29/04/2022 15:55 0 imagen.png

```

- Copiar un archivo a una carpeta: `copy-item <fichero_Creado> <nuevaCarpeta>`

```

master ⚡ +170 -6 | ✅ +7 4:06:08 PM 5ms
JL0el copy-item .\fotos\foto1.jpg .\pseint\
master ⚡ +170 -6 | ✅ +7 4:06:27 PM 18ms
JL0el dir .\pseint\ pwsh

Directory: C:\Users\JL0el\pseint

Mode LastWriteTime Length Name
---- ----- ---- -
-a-- 21/04/2022 23:28 1991 config
-a-- 29/04/2022 15:55 0 foto1.jpg

```

Ahora pasamos al uso del comando `rename-item`

- `rename-item`**
  - Permite renombrar archivos y carpetas.
  - Alias:
  - `ren` Por ejemplo:

- Renombrar carpetas.

```

/home ➔ ⚡+170 -6 | ✅+7 ↵ ↺ ❤️ 4:07:19 PM ⓘ 2ms
  JL0el ➔ rename-item fotos recuerdos
/home ➔ ⚡+170 -6 | ✅+7 ↵ ↺ ❤️ 4:07:39 PM ⓘ 30ms
  JL0el ➔ dir recuerdos

Directory: C:\Users\JL0el\recuerdos

Mode          LastWriteTime      Length Name
----          -----          ---- 
-a---        29/04/2022       15:55      0 📁 foto1.jpg
-a---        29/04/2022       15:55      0 📁 foto2.jpg

/home ➔ ⚡+170 -6 | ✅+7 ↵ ↺ ❤️ 4:07:47 PM ⓘ 22ms
  JL0el ➔ dir fotos
Get-ChildItem: Cannot find path 'C:\Users\JL0el\fotos' because it does not exist.

```

Como podemos observar la carpeta fotos ya no existe, ya que lo hemos renombrado a recuerdos.

- Renombrar archivos. Si quisiéramos renombrar al archivo foto1.jpg que habíamos creado previamente, lo hacemos de la siguiente manera.

```

/home ➔ ⚡+170 -6 | ✅+7 ↵ ↺ ❤️ 4:10:47 PM ⓘ 3ms
  JL0el ➔ rename-item .\recuerdos\foto1.jpg imagen.png
/home ➔ ⚡+170 -6 | ✅+7 ↵ ↺ ❤️ 4:10:59 PM ⓘ 14ms
  JL0el ➔ dir recuerdos

Directory: C:\Users\JL0el\recuerdos

Mode          LastWriteTime      Length Name
----          -----          ---- 
-a---        29/04/2022       15:55      0 📁 foto2.jpg
-a---        29/04/2022       15:55      0 📁 imagen.png

```

Digamos que queremos editar el contenido de un archivo. Creamos el archivo

```

/home ➔ ⚡+175 -6 | ✅+7 ↵ ↺ ❤️ 6:57:18 PM ⓘ 3.022s
  JL0el ➔ new-item fichero.txt

```

Podemos hacerlo de varias maneras.

- Con el block de notas Notepad: `notepad <nombre_archivo>`
- Desde la consola de PowerShell, para ello tendremos que instalar el editor nano o vim.
  - Instalación:
    - Primero instalamos el paquete de instalación chocolatey, que nos ayudará a instalar los editores: `Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object`

```

System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
PS C:\Users\JLOel> Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org)
Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/1.1.0.
Downloading https://community.chocolatey.org/api/v2/package/chocolatey/1.1.0 to C:\Users\JLOel\AppData\Local\Temp\chocolatey\chocoInstall https://community.chocolatey.org/1.1.0

```

- Ahora procedemos a instalar los editores:

- nano--> choco install nano

```

PS C:\Users\JLOel> choco install nano
Chocolatey v1.1.0
Installing the following packages:
nano
By installing, you accept licenses for the packages.
nano v6.2.16 already installed.
Use --force to reinstall, specify a version to install, or try upgrade.

Chocolatey installed 0/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log)

.
.

Warnings:
- nano - nano v6.2.16 already installed.
Use --force to reinstall, specify a version to install, or try upgrade

```

- vim--> choco install vim

```

master # +173 -6 | +7 6:09:49 PM 5ms
JLOel > choco install vim
Chocolatey v1.1.0
Installing the following packages:
vim
By installing, you accept licenses for the packages.
Progress: Downloading vim 8.2.4827 ... 100%

vim v8.2.4827 [Approved]
vim package files install completed. Performing other installation steps
.

```

**NOTA:** Puede que el editor nano o vim ya vengan por defecto.

- Procedemos a editar un archivo de tres maneras distintas:\

- a. Con el comando: wsl nano|vim <nombre\_archivo> , tendremos que introducir la ruta absoluta del archivo como si lo estuviéramos introduciendo en una distribución de Linux del WSL.

```

master # +175 -6 | +7 6:38:32 PM 20ms
JLOel > wsl nano /mnt/c/Users/JLOel/fichero.txt| pwsh

```

- b. Con el comando: bash -c "nano|vim <nombre\_archivo>"

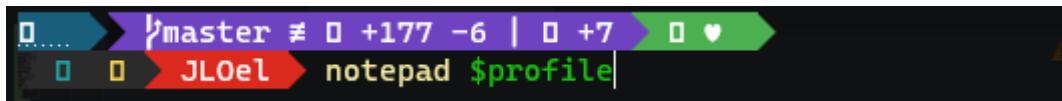
```

master # +175 -6 | +7 6:57:18 PM 3.022s
JLOel > bash -c "nano fichero.txt"| pwsh

```

- c. O simplemente con el comando nano|vim <nombre\_archivo> , pero para ello primero tendremos que seguir los siguientes pasos:

- Primero abrimos el fichero de configuración \$profile de PowerShell, por ejemplo con el notepad.



- Luego añadimos las siguientes líneas:

```
function vi ($File){
    bash -c "vi $File"
}
```

```
function nano ($File){
    bash -c "nano $File"
}
```

A screenshot of Notepad. The title bar says '\*Microsoft.PowerShell\_profile.ps1 - Notepad'. The content of the file is as follows:

```
. $env:USERPROFILE\.config\powershell\user_profile.ps1

function vi ($File){
    bash -c "vi $File"
}

function nano ($File){
    bash -c "nano $File"
}
```

- Ya dentro del archivo procedemos a escribir lo que quisiéramos:

**NOTA: Para entrar en modo edición en vim pulsa la tecla i , para entrar en modo lectura pulsa la tecla ESC .**

- Para guardar y salir de la edición:

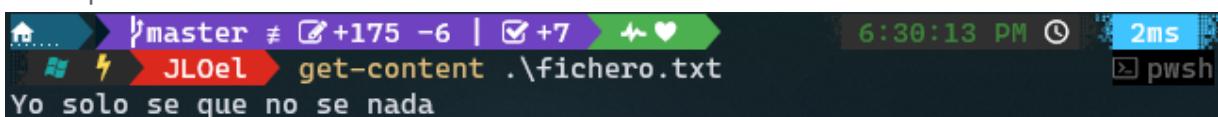
- nano:

- Guardar: `ctrl+O+Enter`
    - Salir: `ctrl+X+Enter`

- vim:

- Primero entramos en modo lectura con la tecla `ESC` .
    - Guardar: `:w`
      - Salir: `:q`

- Ahora procedemos a leer el contenido con normalidad.



Y ahora si quisiéramos leer el contenido del archivo, usamos el comando `get-content` .

- **get-content:**

- Nos permite ver el contenido de un archivo.
  - Alias:

- cat Ejemplo:

```
master # +177 -6 | +7 2:43:28 AM 2ms
JLoel > get-content .\fichero.txt
Yo solo se que no se nada
```

## 2.7. Tuberías y redireccionamiento

Imaginemos que necesitamos conocer todos los archivos que tengan más de 4GB y ordenados de mayor a menor.

¿Y eso cómo lo vamos a hacer? Pues con **tuberías**.

Y digamos que además necesitamos que se almacene en un archivo y eso cómo lo haríamos?, pues con **Redireccionamiento**.

Vamos a empezar con las tuberías.

- Tuberías:
  - Las tuberías nos permiten conectar la salida de un Cmdlet con la entrada de otro, que la tratará como su información de inicio.
  - Utilizaremos el carácter | (tubería o pipe) para enlazar los comandos.
  - Vamos a ver ejemplos:

Primero vamos a ver un comando que ya vimos llamado `get-command` pero ahora no queremos mostrarlos sino contar el total pero para ello usamos comando. `get-command | measure-object`

(El último comando recibe como entrada todos los comandos gracias a la tubería y devuelve el total).

```
master # +177 -6 | +7 3:21:49 AM 18ms
JLoel > get-command | measure-object

Count : 1726
Average :
Sum :
Maximum :
Minimum :
StandardDeviation :
Property :
```

`get-childItem -Recurse | where-object {$_._Length -gt 100Mb}`

El primer comando `Get-ChildItem -Recurse`: devuelve un objeto de archivo o directorio para cada elemento del directorio actual del sistema de archivos. Los objetos de archivo y directorio se pasan por la canalización al segundo comando.

El segundo comando usa `where-object {$_._Length -gt 100Mb}` la propiedad `Length` de todos los objetos del sistema de archivos para seleccionar solo los archivos, que tienen un tamaño mayor de 100Mb.

Vamos a ver si nos sale algo.

```
master # +177 -6 | +7 3:44:15 AM 4ms
JLoel > get-childItem -Recurse | where-object {$_.Length -gt 100Mb}

Directory: C:\Users\JLoel

Mode LastWriteTime Length Name
-- -- -- -- --
-a--- 11/03/2022 4:07 624318381 java_error_in_idea64.hprof
```

Y si también quisieramos ordenarlos en orden descendente por la propiedad longitud haríamos lo siguiente: Fijaros que hemos empleado dos filtros, el primero recoge lo que devuelve el primer comando y el segundo lo que devuelve el segundo comando. Fijaros la utilidad que tienen los filtros.

```
\...\...\CarpetaPruebaFC > master # +179 -6 | +7 6:44:37 AM
JLoel > get-childItem -Recurse | where-object {$_.Length -gt 100Mb} | sort-object -descending -property length

Directory: C:\Users\JLoel\OneDrive\Escritorio\CarpetaPruebaFC\ASIR\Trabajos de SGBD\TEMA 3\TRABAJO DE INVESTIGACIÓN

Mode LastWriteTime Length Name
-- -- -- -- --
-a--- 07/02/2020 22:23 416997774 CONEXIÓN A UN SERVIDOR MYSQL EN UBUNTU.pptx
-a--- 07/02/2020 22:14 412528055 CONEXIÓN A UN SERVIDOR MYSQL EN UBUNTU_JoseLuisObiang.mkv
-a--- 07/02/2020 22:32 177253875 CONEXIÓN A UN SERVIDOR MYSQL EN UBUNTU_JoseLuisObiang&JaimeFernandezNovoa (2).zip
```

Vamos a otro ejemplo que yo creo que es importante.

Imaginaros como administrador necesitáis averiguar aquellos puertos en los que se ha producido una conexión. El comando sería `get-netTCPConnection`

Pero debido a que no se ve bien la información lo ejecutaremos de la siguiente manera `get-netTCPConnection | format-table -autosize`, porque lo que hace ahora es visualizar la información en formato tabla y que se ajuste mejor a la pantalla.

```
\...\...\CarpetaPruebaFC > master # +179 -6 | +7 6:44:37 AM
JLoel > get-netTCPConnection | format-table -autoSize

LocalAddress LocalPort RemoteAddress RemotePort State AppliedSetting OwningProcess
-- -- -- -- -- -- --
:: 54138 :: 0 Bound 3120
:: 53940 :: 0 Bound 12364
:: 53886 :: 0 Bound 12364
:: 53884 :: 0 Bound 12364
:: 53882 :: 0 Bound 12364
:: 49670 :: 0 Listen 796
:: 49669 :: 0 Listen 4288
:: 49668 :: 0 Listen 2836
:: 49667 :: 0 Listen 2600
```

¿Qué pasa si quisieramos ver solo aquellas conexiones establecidas?

Pues solo necesitamos hacer un filtro.

```
□ \..\..\CarpetaPruebaFC ▶ ▷master ≠ □ +179 -6 | □ +7 □ ♥
□ □ □ JL0el ▶ get-netTCPConnection | where-object {$_.State -eq 'Established'} | format-table -autosize
```

| LocalAddress | LocalPort | RemoteAddress | RemotePort | State       | AppliedSetting | OwningProcess |
|--------------|-----------|---------------|------------|-------------|----------------|---------------|
| 192.168.1.93 | 64128     | 20.54.37.64   | 443        | Established | Internet       | 4868          |
| 127.0.0.1    | 63342     | 127.0.0.1     | 54157      | Established | Internet       | 12364         |
| 127.0.0.1    | 63342     | 127.0.0.1     | 54156      | Established | Internet       | 12364         |
| 127.0.0.1    | 63342     | 127.0.0.1     | 54154      | Established | Internet       | 12364         |
| 127.0.0.1    | 63342     | 127.0.0.1     | 54155      | Established | Internet       | 12364         |
| 127.0.0.1    | 63342     | 127.0.0.1     | 54153      | Established | Internet       | 12364         |
| 127.0.0.1    | 63342     | 127.0.0.1     | 54152      | Established | Internet       | 12364         |
| 192.168.1.93 | 54647     | 52.146.136.48 | 443        | Established | Internet       | 10692         |
| 192.168.1.93 | 54176     | 140.82.114.25 | 443        | Established | Internet       | 2732          |

Vamos a ver ahora el tema de Redireccionamiento

- **Redireccionamiento:**

- Las redirecciones nos permiten mandar los resultados a un lugar diferente de la pantalla.  
Normalmente a un archivo.
- > : Crea un nuevo archivo y deposita en él la salida del cmdlet.

```
□ \..\..\CarpetaPruebaFC ▶ ▷master ≠ □ +179 -6 | □ +7 □ ♥
□ □ □ JL0el ▶ get-childItem > informe.txt
□ \..\..\CarpetaPruebaFC ▶ ▷master ≠ □ +179 -6 | □ +7 □ ♥
□ □ □ JL0el ▶ get-content ./informe.txt
```

Directory: C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC

| Mode  | LastWriteTime    | Length  | Name               |
|-------|------------------|---------|--------------------|
| d---  | 29/04/2022 16:04 | 0       | ASIR               |
| -a--- | 28/04/2022 13:53 | 1068    | Celestina.txt      |
| -a--- | 28/04/2022 13:35 | 3504    | CSV.csv            |
| -a--- | 30/04/2022 6:05  | 1649    | Ejemplo_XML1.xml   |
| -a--- | 30/04/2022 6:05  | 1835    | Ejemplo_XML2.xml   |
| -a--- | 21/11/2019 11:06 | 67246   | EjerciciosBD.pdf   |
| -a--- | 30/04/2022 5:36  | 206467  | fichero            |
| -a--- | 30/04/2022 5:41  | 202254  | fichero.csv        |
| -a--- | 30/04/2022 7:11  | 0       | informe.txt        |
| -a--- | 30/04/2022 6:05  | 2535    | relacionados.xml   |
| -a--- | 22/10/2020 15:26 | 1068951 | SERVIDOR_DHCP.docx |
| -a--- | 30/04/2022 6:10  | 1172    | testXML.xml        |

- >> : Añade al contenido del archivo la salida del cmdlet. Solo he añadido la fecha

```

D:\...\...\CarpetaPruebaFC> >master # 0 +179 -6 | 0 +7 0 ♥
  0 0 > JLoel get-date >> .\informe.txt
D:\...\...\CarpetaPruebaFC> >master # 0 +179 -6 | 0 +7 0 ♥
  0 0 > JLoel get-content .\informe.txt

Directory: C:\Users\JLoel\OneDrive\Escritorio\CarpetaPruebaFC

Mode                LastWriteTime          Length Name
----                -----          -----    -----   -----
d---        29/04/2022     16:04              0 ASIR
-a---        28/04/2022     13:53            1068 Celestina.txt
-a---        28/04/2022     13:35            3504 CSV.csv
-a---        30/04/2022      6:05            1649 Ejemplo_XML1.xml
-a---        30/04/2022      6:05            1835 Ejemplo_XML2.xml
-a---        21/11/2019     11:06           67246 EjerciciosBD.pdf
-a---        30/04/2022      5:36           206467 fichero
-a---        30/04/2022      5:41           202254 fichero.csv
-a---        30/04/2022      7:11              0 informe.txt
-a---        30/04/2022      6:05            2535 relacionados.xml
-a---        22/10/2020     15:26          1068951 SERVIDOR_DHCP.docx
-a---        30/04/2022      6:10            1172 testXML.xml

sábado, 30 de abril de 2022 7:13:28

```

Las salidas de los comandos anteriores también podríamos añadirlos al mismo archivo con el signo `>` pero si quisiéramos sobreescribir entonces sería con `>>`, por ejemplo, el comando que nos mostraba las conexiones o puertos abiertos.

```

D:\...\...\CarpetaPruebaFC> >master # 0 +179 -6 | 0 +7 0 ♥
  0 0 > JLoel get-netTCPConnection|where-object {$_.State -eq 'Established'}|format-table -autosize >> .\informe.txt
D:\...\...\CarpetaPruebaFC> >master # 0 +179 -6 | 0 +7 0 ♥
  0 0 > JLoel get-content .\informe.txt

Directory: C:\Users\JLoel\OneDrive\Escritorio\CarpetaPruebaFC

Mode                LastWriteTime          Length Name
----                -----          -----    -----   -----
d---        29/04/2022     16:04              0 ASIR
-a---        28/04/2022     13:53            1068 Celestina.txt
-a---        28/04/2022     13:35            3504 CSV.csv
-a---        30/04/2022      6:05            1649 Ejemplo_XML1.xml
-a---        30/04/2022      6:05            1835 Ejemplo_XML2.xml
-a---        21/11/2019     11:06           67246 EjerciciosBD.pdf
-a---        30/04/2022      5:36           206467 fichero
-a---        30/04/2022      5:41           202254 fichero.csv
-a---        30/04/2022      7:11              0 informe.txt
-a---        30/04/2022      6:05            2535 relacionados.xml
-a---        22/10/2020     15:26          1068951 SERVIDOR_DHCP.docx
-a---        30/04/2022      6:10            1172 testXML.xml

sábado, 30 de abril de 2022 7:13:28

LocalAddress LocalPort RemoteAddress  RemotePort State      AppliedSetting OwningProcess
-----  -----  -----  -----  -----  -----  -----  -----
192.168.1.93 64128    20.54.37.64    443     Established Internet 4868
127.0.0.1    63342    127.0.0.1     54154    Established Internet 12364
127.0.0.1    63342    127.0.0.1     54157    Established Internet 12364
127.0.0.1    63342    127.0.0.1     54153    Established Internet 12364
127.0.0.1    63342    127.0.0.1     54156    Established Internet 12364
127.0.0.1    63342    127.0.0.1     54155    Established Internet 12364
127.0.0.1    63342    127.0.0.1     54152    Established Internet 12364
192.168.1.93 54647    52.146.136.48    443     Established Internet 10692

```

En este último comando como podemos fijarnos hemos utilizado un comando, tuberías y redireccionamiento.

## 2.8. Iniciación a los scripts

## **¿Qué es un script?**

Bueno, los administradores de sistemas utilizan principalmente los scripts para automatizar tareas, pero qué es un script?

pues un script es un archivo de texto plano que contiene una secuencia de órdenes o comandos. Para que un archivo sea tratado como un script tiene que tener una extensión de ps1.

## **¿Para qué se usa un script?**

Podríamos usar un script para automatizar tareas, por ejemplo para:

- Comprobar si un determinado servicio está activo y si se ha detenido lo quiere activar.
- Realizar una copia de seguridad en un viernes a las 3PM.
- Copia de seguridad del sistema por ejemplo a las 6AM.
- Creación de usuarios de manera masiva.

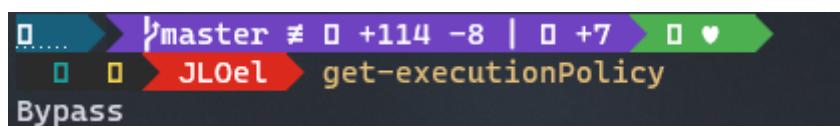
Fijaros la gran utilidad que tiene a la hora de automatizar tareas el uso de los scripts.

Ahora vamos a ver el punto de seguridad en los scripts.

**Seguridad en los scripts:** PowerShell incorpora medidas de seguridad para evitar que se ejecuten sin la autorización del usuario scripts que puedan dañar al equipo y ha establecido 4 niveles de seguridad.

- Niveles de seguridad:
  - Restricted. Es el nivel predeterminado que no permite la ejecución de scripts.
  - AllSigned. Todos los scripts deberán estar autenticados para poder ejecutarlos, es la opción más segura.
  - RemoteSigned. Solo deberán ser autenticados los scripts que proceden de una ubicación remota, es decir, solo funcionarían nuestros scripts y aquellos que se bajen de la red pero que estén autenticados.
  - Unrestricted. Permite la ejecución de cualquier script y por eso es la opción menos segura.

Vamos a abrir nuestro PowerShell como administrador. Ahora procedemos a comprobar la política de seguridad que tenemos establecida con el comando `get-executionPolicy`



```
PS C:\Windows\system32> get-executionPolicy
Bypass
```

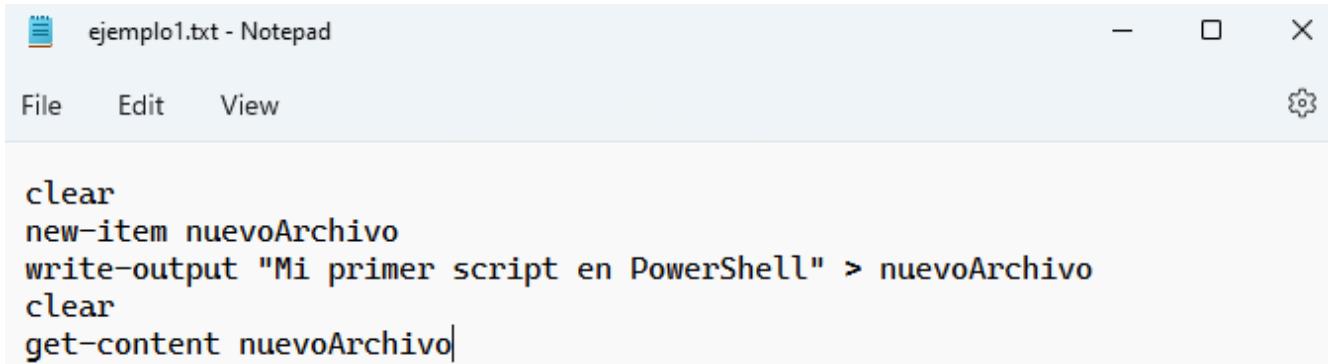
Bypass es parecido a Unrestricted.

Procedemos a cambiar nuestra política de seguridad a RemoteSigned, es decir, nuestros scripts se podrán ejecutar pero los que sean remotos solo se podrán ejecutar si están autenticados.

```
master # +114 -8 | +7
JLoel set-executionPolicy RemoteSigned
master # +114 -8 | +7
JLoel get-executionPolicy
RemoteSigned
```

Vamos a crear nuestro primer Script, primero lo haremos con algún block de notas que tengamos y lo llamaremos ejemplo1.ps1(el ps1 es la extensión de los scripts en PowerShell)

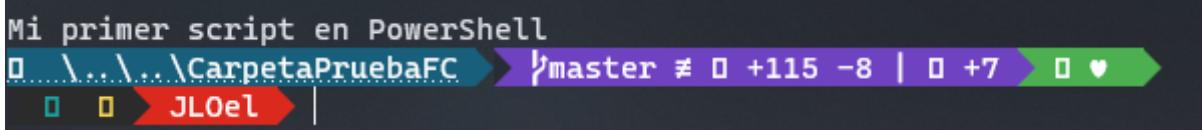
Nuestro primer script primero limpia la pantalla, luego crea un fichero, introduce texto, limpia otra vez la pantalla y por último muestra el contenido del archivo.



```
ejemplo1.txt - Notepad
File Edit View
clear
new-item nuevoArchivo
write-output "Mi primer script en PowerShell" > nuevoArchivo
clear
get-content nuevoArchivo
```

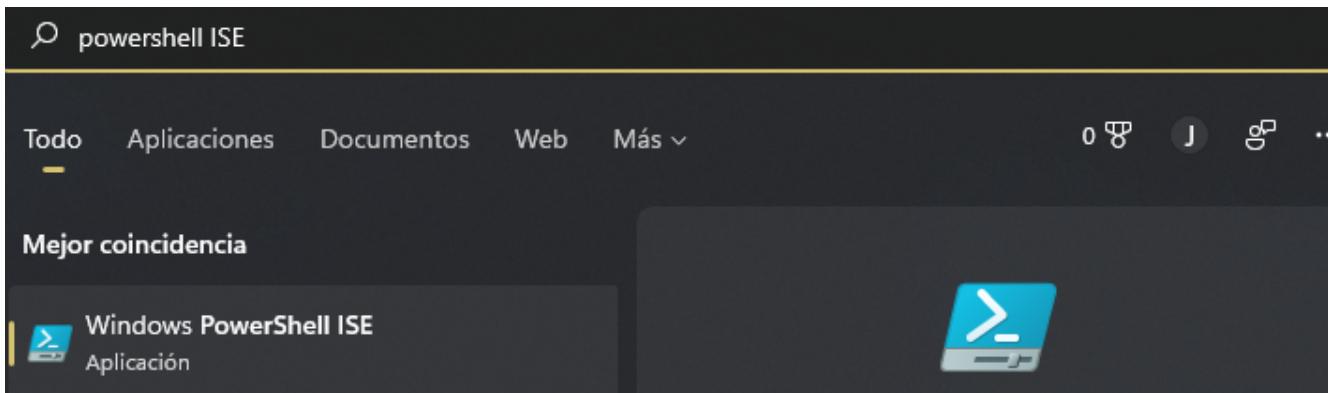
Para ejecutarlo simplemente escribimos la ruta del archivo punto PS1.

```
master # +115 -8 | +7
JLoel cd .\OneDrive\Escritorio\CarpetaPruebaFC\
..\..\CarpetaPruebaFC master # +115 -8 | +7
JLoel .\ejemplo.ps1
```



```
Mi primer script en PowerShell
..\..\CarpetaPruebaFC master # +115 -8 | +7
JLoel |
```

Hemos hecho un script simplemente utilizando un procesador de texto como Notepad pero ahora usaremos el entorno que trae PowerShell para desarrollar script, es lo que se llama Windows PowerShell ISE.



Este entorno tiene dos partes, una parte en la que escribimos el script y otra parte en la que se ejecutan los scripts y también podemos ejecutar comandos. Hay una cosa muy interesante que nos permite depurar los scripts.

## PowerShell ISE

- PowerShell dispone de un entorno gráfico llamado PowerShell ISE(Integrated Scripting Environment)
- Su gran ventaja es que integra las tareas relativas a la escritura, depuración y ejecución de scripts.

Vamos a crear un nuevo script el cual llamaremos ejemplo2.

Podemos ejecutar todo el script.

The screenshot shows the Windows PowerShell ISE interface. At the top, there's a menu bar with Archivo, Editar, Ver, Herramientas, Depurar, Complementos, and Ayuda. Below the menu is a toolbar with various icons. A tab labeled "ejemplo2.ps1" is selected. The code in the editor is:

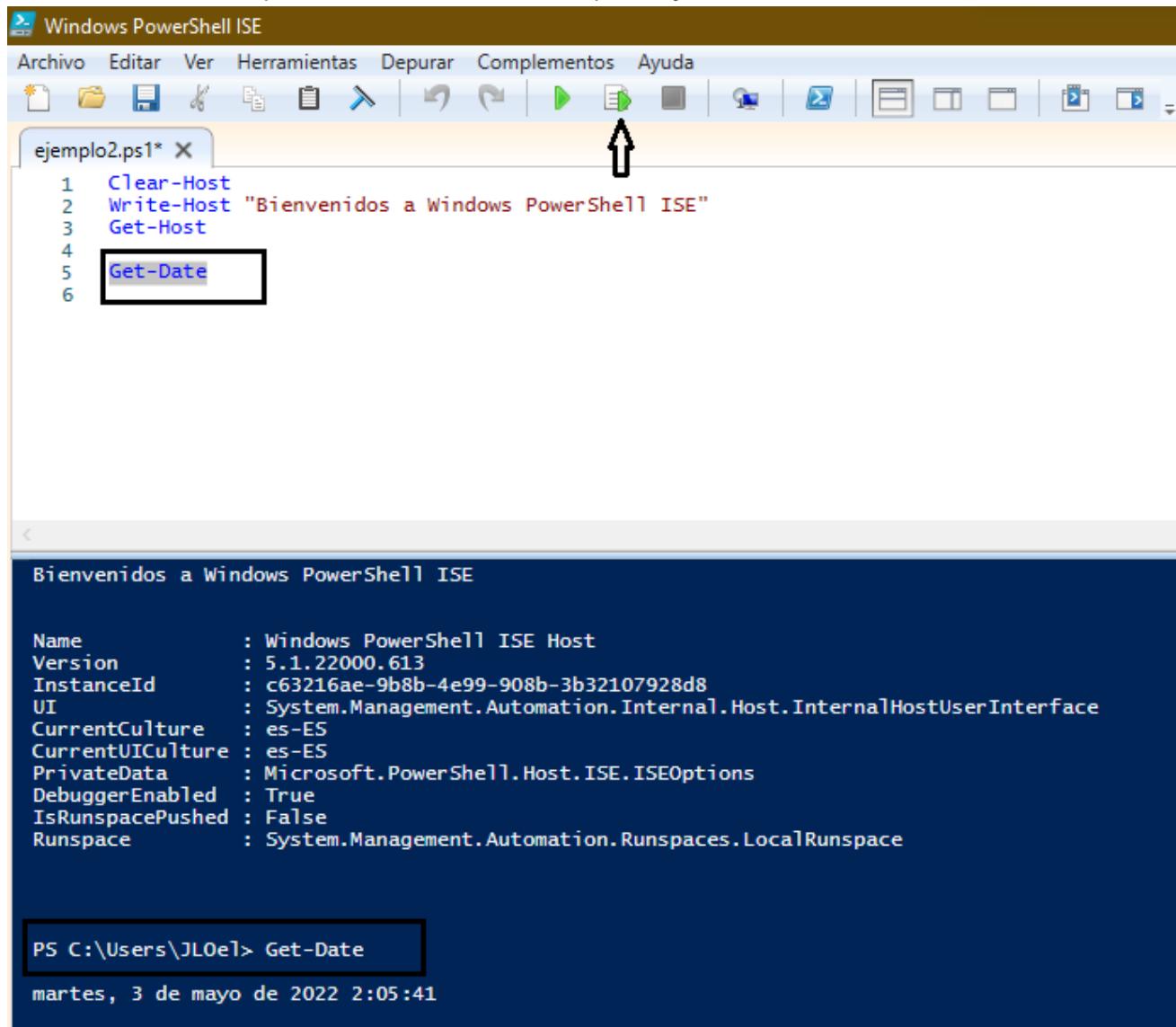
```
1 Clear-Host
2 Write-Host "Bienvenidos a Windows PowerShell ISE"
3 Get-Host
4
```

Below the editor, a preview pane displays the output of the script. It starts with "Bienvenidos a Windows PowerShell ISE" and then lists properties of the host object:

```
Bienvenidos a Windows PowerShell ISE

Name          : Windows PowerShell ISE Host
Version       : 5.1.22000.613
InstanceId    : c63216ae-9b8b-4e99-908b-3b32107928d8
UI           : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : es-ES
CurrentUICulture : es-ES
PrivateData   : Microsoft.PowerShell.Host.ISE.ISEOptions
DebuggerEnabled : True
IsRunspacePushed : False
Runspace      : System.Management.Automation.Runspaces.LocalRunspace
```

O bien solo las líneas que seleccionemos desde la opción ejecutar selección



```
Windows PowerShell ISE

Archivo Editar Ver Herramientas Depurar Complementos Ayuda
ejemplo2.ps1* ↑
1 Clear-Host
2 Write-Host "Bienvenidos a Windows PowerShell ISE"
3 Get-Host
4
5 Get-Date
6

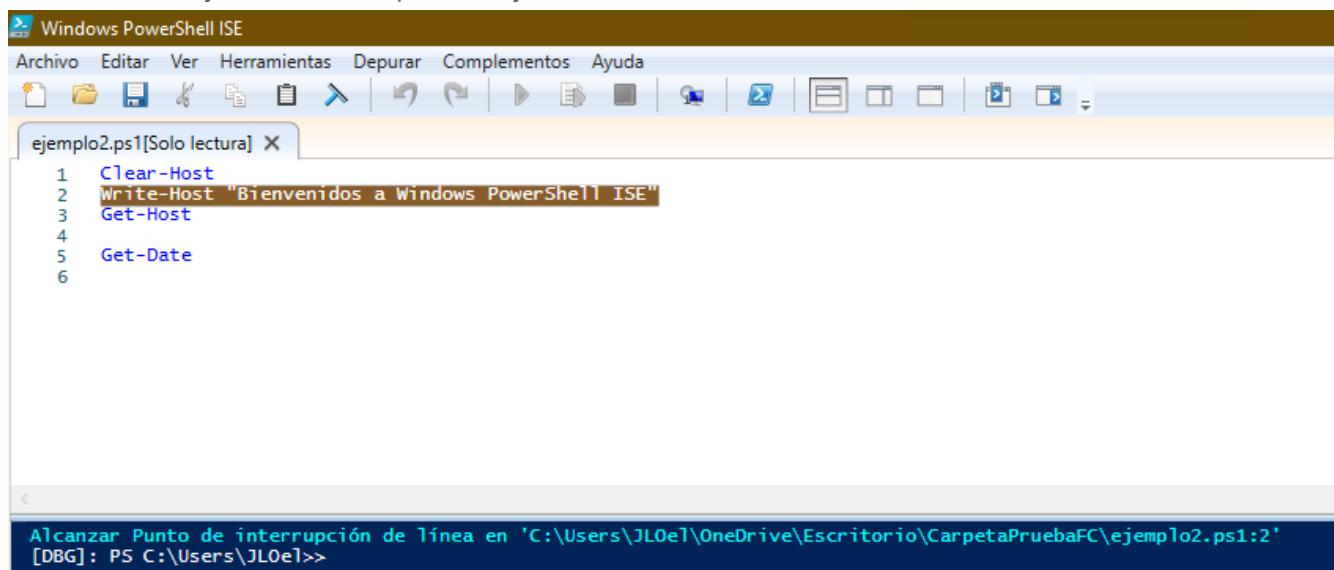
Bienvenidos a Windows PowerShell ISE

Name          : Windows PowerShell ISE Host
Version       : 5.1.22000.613
InstanceId    : c63216ae-9b8b-4e99-908b-3b32107928d8
UI            : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : es-ES
CurrentUICulture : es-ES
PrivateData   : Microsoft.PowerShell.Host.ISE.ISEOptions
DebuggerEnabled : True
IsRunspacePushed : False
Runspace      : System.Management.Automation.Runspaces.LocalRunspace

PS C:\Users\JL0el> Get-Date
martes , 3 de mayo de 2022 2:05:41
```

Si quisieramos añadir un punto de ruptura en alguna línea de nuestro script, digamos en la línea 2.

Primero le damos al botón derecho y seleccionamos la opción alternar puntos de interrupción, entonces cuando le das ejecutar se va a parar la ejecución en dicha línea



```
Windows PowerShell ISE

Archivo Editar Ver Herramientas Depurar Complementos Ayuda
ejemplo2.ps1[Solo lectura] ↑
1 Clear-Host
2 Write-Host "Bienvenidos a Windows PowerShell ISE"
3 Get-Host
4
5 Get-Date
6

Alcanzar Punto de interrupción de Línea en 'C:\Users\JL0el\OneDrive\Escritorio\CarpetapruEBC\ejemplo2.ps1:2'
[DBG]: PS C:\Users\JL0el>>
```

y después si quiero ver lo que va pasando poco a poco pulsaría F11 y continuaría la ejecución en línea a línea a línea cada vez que le doy al F11

The screenshot shows the Windows PowerShell ISE interface. At the top, there's a menu bar with Archivo, Editar, Ver, Herramientas, Depurar, Complementos, and Ayuda. Below the menu is a toolbar with various icons. A tab labeled "ejemplo2.ps1[Solo lectura]" is open. The code in the editor is:

```

1  Clear-Host
2  Write-Host "Bienvenidos a Windows PowerShell ISE"
3  Get-Host
4
5  Get-Date
6

```

Below the editor, the output window shows the results of running the script. It starts with a debugger prompt "[DBG]:" followed by the host information and the output of the Get-Date cmdlet.

```

Alcanzar Punto de interrupción de línea en 'C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\ejemplo2.ps1:2'
[DBG]: PS C:\Users\JL0el>>
Bienvenidos a Windows PowerShell ISE

[DBG]: PS C:\Users\JL0el>>

Name          : Windows PowerShell ISE Host
Version       : 5.1.22000.613
InstanceId    : c63216ae-9b8b-4e99-908b-3b32107928d8
UI           : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : es-ES
CurrentUICulture : es-ES
PrivateData   : Microsoft.PowerShell.Host.ISE.ISEOptions
DebuggerEnabled : True
IsRunspacePushed : False
Runspace      : System.Management.Automation.Runspaces.LocalRunspace

[DBG]: PS C:\Users\JL0el>>
DisplayHint : DateTime
Date        : 03/05/2022 0:00:00
Day         : 3
DayOfWeek   : Tuesday
DayOfYear    : 123
Hour        : 2
Kind         : Local
Millisecond : 686
Minute       : 20
Month        : 5
Second       : 12
Ticks        : 637871412126865466
TimeOfDay    : 02:20:12.6865466
Year         : 2022
DateTime     : martes, 3 de mayo de 2022 2:20:12

```

## 2.9. Fundamentos de scripts-I: Comentarios, Variables/Constantes

- \* Cualquier script debe tener líneas de comentarios, que explique algún detalle en concreto o algo.
- \* Es fundamental saber correctamente las variables.

- **Comentarios:**

- Hay dos formas de comentarios:
  - En línea. Empiezan solo con una almohadilla (#).
  - En bloque:
    - <#
    - Todo lo que vaya
    - Entre etiquetas
    - También se comentarios
    - #>

Creamos un archivo el cual llamaremos comentarios, incluimos comentario en línea, comentario en bloque y algún comando.

```
Windows PowerShell ISE
Archivo Editar Ver Herramientas Depurar Complementos Ayuda
Ejecutar script (F5)

comentarios.ps1 ×
1 #Ejemplos de comentarios
2 <#Es un ejemplo
3 de bloque de
4 comentarios
5 #
6
7 Clear-Host #Limpia la pantalla
8 Get-Date #Muestra la fecha

martes, 3 de mayo de 2022 2:32:49

PS C:\Users\JLoel>
```

Si los comentarios están bien hechos no debería mostrar el texto comentado.

- **Variables:**

- Es un espacio en memoria, que contiene un valor que puede cambiar.
- No es obligatorio declararla, ni inicializarla.
- Es suficiente con utilizar el signo = para asignarle un valor. PowerShell se encargará de crearla y determinar el tipo.
- El primer carácter debe ser siempre un símbolo (\$).

Vamos a hacer una prueba:

```
Windows PowerShell ISE
Archivo Editar Ver Herramientas Depurar Complementos Ayuda
Ejecutar script (F5)

variables.ps1 ×
1 #Ejemplo de variables
2 $edad=22
3 $nombre="Jorge"
4 $edad
5 $nombre

PS C:\Users\JLoel> C:\Users\JLoel\OneDrive\Escritorio\CarpetaPruebaFC\variables.ps1
22
Jorge

PS C:\Users\JLoel>
```

PowerShell en función del dato que pongas es capaz de asignar un tipo a una variable, las cadenas van entre comillas y los números sin comillas. Si quisieramos saber qué tipo de dato es una variable, usamos, el comando \$nombre\_variable.GetType()

```
PS C:\Users\JL0e1> $nombre.GetType()
```

| IsPublic | IsSerial | Name   | BaseType      |
|----------|----------|--------|---------------|
| True     | True     | String | System.Object |

También

podríamos declarar variables con el cmdlet new-variable -name nombre\_variable [-value valor]

The screenshot shows the Windows PowerShell ISE interface. The title bar says "Windows PowerShell ISE". The menu bar includes Archivo, Editar, Ver, Herramientas, Depurar, Complementos, and Ayuda. Below the menu is a toolbar with various icons. A tab labeled "variables.ps1" is selected. The code in the editor is:

```
1 #Ejemplo de variables
2 new-variable -name edad -value 22
3 new-variable -name nombre -value "Jorge"
4 $edad
5 $nombre
```

Below the editor, the command PS C:\Users\JL0e1> C:\Users\JL0e1\OneDrive\Escritorio\CarpetapruебaFC\variables.ps1 is run, followed by its output:

```
22
Jorge
```

- **Constantes:**

- Las constantes son variables cuyos valores no pueden cambiarse
- Se crean con el comando new-variable -name nombre\_variable [-value valor] -option constant

The screenshot shows the Windows PowerShell ISE interface. The title bar says "Windows PowerShell ISE". The menu bar includes Archivo, Editar, Ver, Herramientas, Depurar, Complementos, and Ayuda. Below the menu is a toolbar with various icons. A tab labeled "constantes.ps1" is selected. The code in the editor is:

```
1 #Constantes
2 new-variable -name PI -value 3.14 -option constant
3 $PI
4 $PI.GetType()
```

Below the editor, the command PS C:\Users\JL0e1> #Constantes is run, followed by its output:

```
new-variable -name PI -value 3.14 -option constant
$PI
$PI.GetType()
3.14
```

Finally, the command \$PI.GetType() is run again, resulting in the following table:

| IsPublic | IsSerial | Name   | BaseType         |
|----------|----------|--------|------------------|
| True     | True     | Double | System.ValueType |

Si intentamos cambiar el valor de una constante nos saldrá el siguiente mensaje.

The screenshot shows the Windows PowerShell ISE interface. In the top menu bar, the 'File' (Archivo) option is highlighted. Below the menu bar, there's a toolbar with various icons. A tab labeled 'constantes.ps1' is open. The code in the editor is:

```
1 #Constantes
2 new-variable -name C -value 300000 -option constant
3 $C=2
4 $C
```

In the bottom pane, the output of running the script is displayed:

```
PS C:\Users\JL0el> C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\constantes.ps1
No se puede sobrescribir la variable C porque es de solo lectura o constante.
En C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\constantes.ps1: 3 Carácter: 1
+ $C=2
+ ~~~
+ CategoryInfo          : WriteError: (C:String) [], SessionStateUnauthorizedAccessException
+ FullyQualifiedErrorId : VariableNotWritable
300000
```

#### *Tipos de datos con los que trabaja PowerShell*

| Tipos de datos | Descripción                |
|----------------|----------------------------|
| int            | Entero con signo           |
| Double         | Números decimales          |
| Char           | Un solo carácter           |
| String         | Una cadena de texto        |
| Boolean        | Valor lógico(True o False) |

## 2.10. Fundamentos de scripts-II: Estructuras de control y funciones

Esta es la parte más interesante porque trabajaremos con Estructuras de control, condiciones y funciones.

Como cualquier otro lenguaje, PowerShell también tiene estructuras de control, que permiten modificar el flujo de ejecución de las instrucciones de un programa.

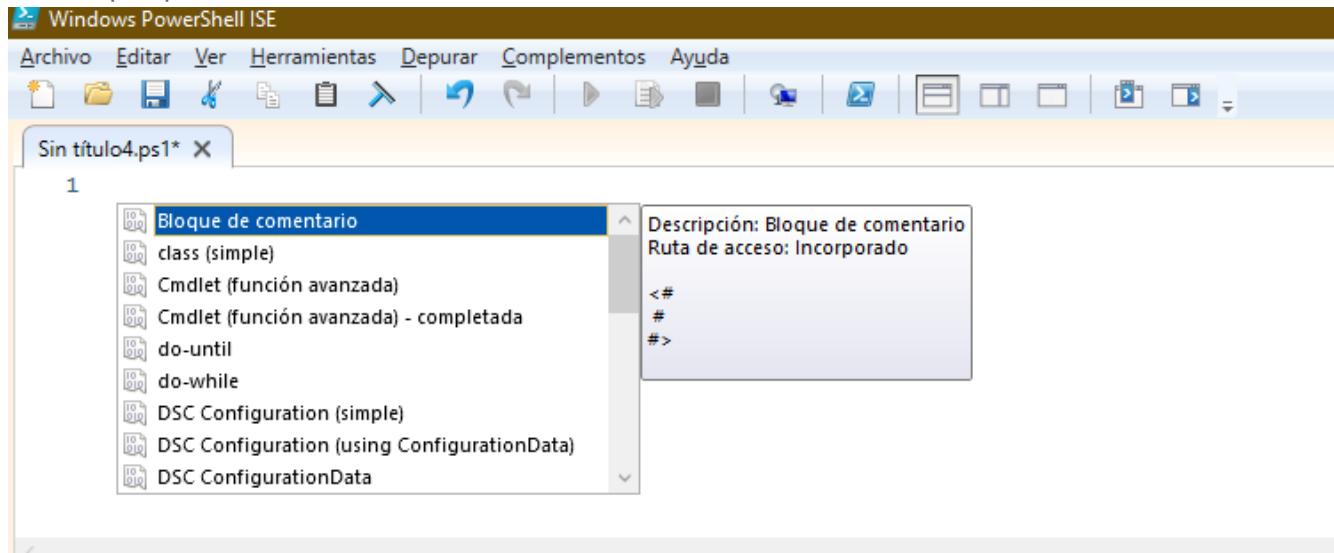
- Por ejemplo dependiendo de una condición se puede ejecutar un grupo u otro de sentencias.
- Dependiendo de la condición se puede ejecutar un grupo determinado de sentencias un número determinado de veces.

PowerShell incorpora una funcionalidad muy útil denominada snippets, que son estructuras de código listas para ser usadas. Vamos a verlo ahora cuando hablemos de las estructuras condicionales.

- **Estructuras condicionales:**

- Las estructuras condicionales ejecutan un grupo de sentencias en función del valor de una condición.

Vamos a verlo con un ejemplo: Con las teclas **ctrl+J** podemos elegir el código base de la estructura de control que queramos



**Vamos a hacer un script que comprueba la conectividad con el servidor pero para ello primero tenemos que abrir la ISE de PowerShell como administrador.**

Pasos:

- Limpiamos la pantalla: `Clear-Host .`
- Mostramos el mensaje Conectividad: `Write-Host "Conectividad" .`
- Pedimos al usuario que introduzca una dirección IP, la cual la almacenamos en una variable:  
`$IP=Read-Host "Introduce una IP "`
- Comprobamos la conectividad, el comando nos devuelve un valor booleano, por lo tanto, lo almacenaremos en una variable: `$conexion=Test-Connection $IP -count 1 -Quiet`
  - `-count`: Indica la cantidad de Ping
  - `-Quiet`: Muestra si hay conexión o no hay.
- Evaluamos una condición según si hay conexión o no con la estructura condicional `if-else`.

- o if: si se cumple nos mostrará el mensaje "\$ip conexión establecida".

The screenshot shows the Windows PowerShell ISE interface. The top menu bar includes Archivo, Editar, Ver, Herramientas, Depurar, Complementos, and Ayuda. Below the menu is a toolbar with various icons. The main code editor window is titled 'servidor.ps1' and contains the following PowerShell script:

```

1 #Conexion con el Servidor
2 Clear-Host
3 Write-Host "Conectividad"
4 $ip=Read-Host "Introduce una IP "
5 $conexion=Test-Connection $ip -count 1 -Quiet
6
7 if($conexion){
8     Write-Host "$ip conexión establecida"
9 }else{
10    Write-Host "$ip Error de conexión"
11 }
12

```

The output window below displays the results of running the script. It shows the prompt 'Conectividad', the user input 'Introduce una IP : 192.168.1.1', and the output '192.168.1.1 conexión establecida'.

- o else: "\$ip Error de conexión". Esta IP no está en nuestra red

The screenshot shows the Windows PowerShell ISE interface. The top menu bar includes Archivo, Editar, Ver, Herramientas, Depurar, Complementos, and Ayuda. Below the menu is a toolbar with various icons. The main code editor window is titled 'servidor.ps1' and contains the same PowerShell script as the previous screenshot:

```

1 #Conexion con el Servidor
2 Clear-Host
3 Write-Host "Conectividad"
4 $ip=Read-Host "Introduce una IP "
5 $conexion=Test-Connection $ip -count 1 -Quiet
6
7 if($conexion){
8     Write-Host "$ip conexión establecida"
9 }else{
10    Write-Host "$ip Error de conexión"
11 }
12

```

The output window below displays the results of running the script. It shows the prompt 'Conectividad', the user input 'Introduce una IP : 192.168.2.1', and the output '192.168.2.1 Error de conexión'.

**¿Para qué puede servir este script?** Imaginarnos que tenemos un fichero con todos los servidores que nos interesa, podemos hacer una conexión con cada uno de esos servidores y elaborar un informe que nos diga qué servidor está disponible y cuál no?

- **Estructuras repetitivas:**

- Nos permiten repetir un bloque de instrucciones.
- Tipos:
  - While. Se repite la condición cero o más veces de manera
  - Do-while. Se repite el bloque de código por lo menos una vez.
  - For. Se repite la condición un número determinado de veces.

- Foreach. Lo usaremos para recorrer una colección de datos.

Vamos a suponer que la IP de los servidores lo tenemos en un fichero de texto, el cual creamos ahora.

En este

fichero tenemos las IP de los servidores.

| GNU nano 4.8                          | servidores.txt | Modified |
|---------------------------------------|----------------|----------|
| 192.168.1.1<br>192.168.2.1<br>8.8.8.8 |                |          |

Ahora como queremos hacer lo mismo que el script anterior pero para muchas Ip de servidores tendríamos que usar una estructura repetitiva. Como ya no necesitamos introducir las IP ya que hemos creado un fichero que contiene las IP. Para obtener las IP del fichero usamos el cmdlet `get-content <path_archivo>` y lo almacenamos en una variable. Luego hacemos un bucle con la estructura repetitiva `foreach` y dentro del bucle hacemos el test y evaluamos la condición.

```

#Conexion con el Servidor
Clear-Host
Write-Host "Conectividad"
$datos=Get-Content C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\servidores.txt
foreach ($ip in $datos)
{
    $conexion=Test-Connection $ip -count 1 -Quiet
    if($conexion){
        Write-Host "$ip conexión establecida"
    }else{
        Write-Host "$ip Error de conexión"
    }
}

```

Conectividad  
192.168.1.1 conexión establecida  
192.168.2.1 Error de conexión  
8.8.8.8 conexión establecida

- **Función:**

- Es un conjunto de instrucciones a las que le damos un nombre y podemos llamarla en cualquier parte del código.

Vamos a definir una función en base al script anterior. Primero definimos la función, la cuál recibirá como parámetro el conjunto de datos que Esta función lo que hará es el `foreach`.

Administrador: Windows PowerShell ISE

Archivo Editar Ver Herramientas Depurar Complementos Ayuda

servidor.ps1 X

```
1 #Conexión con el Servidor
2
3 #Definición de funciones
4 Function Conectividad($conjunto){
5     foreach ($ip in $datos)
6     {
7         $conexión=Test-Connection $ip -count 1 -Quiet
8         if($conexión){
9             Write-Host "$ip conexión establecida"
10        }else{
11            Write-Host "$ip Error de conexión"
12        }
13    }
14
15 }
16
17 #Inicio del programa
18
19 Clear-Host
20 Write-Host "Conektividad"
21 $datos=Get-Content C:\Users\JL0el\OneDrive\Escritorio\CarpetaPruebaFC\servidores.txt
22 Conectividad($datos)
```

Conektividad  
192.168.1.1 conexión establecida  
192.168.2.1 Error de conexión  
8.8.8.8 conexión establecida

Buen pues si hacemos un resumen de lo que se ha visto en el curso. Hemos visto lo más importante

- Obtener ayuda.
- La diferencia entre los cmdlet y los módulos que nos ayudan a obtener información del sistema.
- También hemos visto estructuras de control y repetitivas.
- Iniciación a los scripts, filtros y redirecciónamientos.
- Y hemos visto que cualquier administrador de sistemas operativos Windows tiene que usar la PowerShell sí o sí.