

PowerShell (PS).....	1
Contenidos:.....	1
Línea de Comandos de Windows .....	2
Introducción: .....	2
La línea de comandos de Windows .....	2
¿Qué es la PS?.....	2
Abrir la PS .....	2
La ayuda de PS.....	3
Introducción .....	3
Concepto de CmdLet .....	4
Concepto de Módulo .....	4
Obtener ayuda de un cmdlet.....	5
Atajos.....	6
Alias .....	6
Gestión de Archivos y Carpetas, Comandos fundamentales: .....	7
Get-ChildItem .....	7
Comandos para crear archivos y directorios New-Item (ni).....	7
Eliminar archivos y carpetas con Remove-Item (rm) .....	7
Otros:.....	7
Tuberías y redireccionamiento.....	7
Tuberías: .....	7
Redireccionamiento: .....	9

## PowerShell (PS)

### Contenidos:

La línea de comandos de Windows. PowerShell y PowerShell ISE

La ayuda del PS

Comandos para Gestión de carpetas y archivos

Tuberías y redireccionamiento:

Iniciación a los scripts (ISE)

Scripts – variables

Scripts -Estructuras de control

## Línea de Comandos de Windows

### Introducción:

Interfaz de usuario:

- Interfaz gráfica (GUI): Visual, para manejar el S.O.
- Interfaz de línea de comandos (CLI): Línea de texto, se usa para automatizar tareas

### La línea de comandos de Windows

Tenemos dos interfaces:

- CMD o símbolo del sistema, se usa cada vez menos
- PowerShell. Herramienta más potente

### ¿Qué es la PS?

Es la nueva línea de comandos

Para automatizar tareas.

PowerShell no solo permite interactuar con el sistema operativo, sino también con programas de Microsoft como SQL Server, Exchange o IIS.

La característica distintiva de PowerShell, es que es un intérprete de comandos orientado a objetos. La información de entrada y de salida en cada etapa del proceso (cmdlet, "comándulo") es un conjunto de instancias de objeto, a diferencia de lo que ocurre con los intérpretes de comandos tradicionales, que solo devuelven y reciben texto.

Es un completo lenguaje de Script.

Actualmente se incluye en todos los S.O. de Microsoft. A lo largo del tiempo han salido versiones:

- V1 en 2006
- V2 en 2009
- V3 en 2012
- V4 en 2013
- V5 en 2017
- V5.1 en 2017
- VCore6.0 en 2018, válida también para Linux y MacOS
- V7 en diciembre de 2019 (en fase de pruebas)

No debemos confundir:

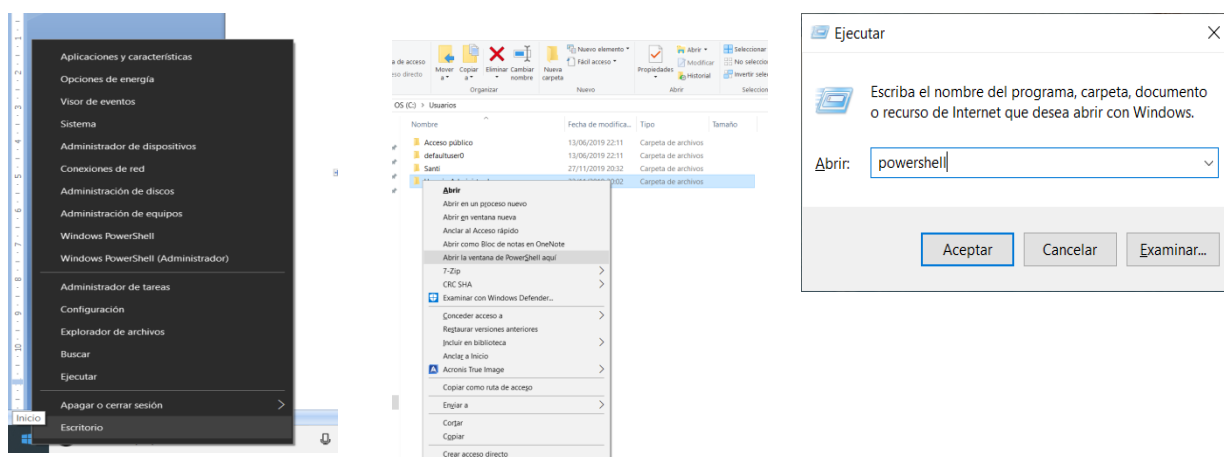
- Windows PowerShell ISE (Integrated Scripting Environment): Para escribir comandos y probar scripts
- Windows PowerShell

### Abrir la PS

Botón derecho en icono Windows y vemos opción de menú.

En el icono de una carpeta shift+Botón derecho y aparece opción

Windows+R y ejecutar el comando: `powershell`



En los tres casos veremos algo similar a:

```

Selecionar Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\Santi>

```

Saber la versión: Get-Host

```

Windows PowerShell
PS C:\Users\Santi> Get-Host

Name           : ConsoleHost
Version        : 5.1.17763.771
InstanceId      : a89d12c4-ca80-47ab-934a-5a5639f27b03
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

```

En la primera de las formas, vemos que existe la opción de ejecutar Windows PowerShell como administrador. Esta forma de ejecución será necesaria cuando los comandos requieran de permisos de administrador.

## La ayuda de PS

### Introducción

La ayuda es muy completa, se necesita internet para descargar la actualización.

Desde PS> Update-Help

```

Windows PowerShell
PS C:\Users\Santi> Update-Help

Actualizando Ayuda para el módulo Microsoft.PowerShell.Management
Buscando contenido de Ayuda...
[oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo]

```

Puede que muestre que ha habido algunos errores (es normal)

Para que se instalen “las ayudas” es recomendable hacerlo desde la consola PS en modo administrador.

Si quieres borrar pantalla puedes teclear el comando `Cls`

### Concepto de CmdLet

Un cmdlet es una combinación de **verbo** y **nombre** separados por un guión.

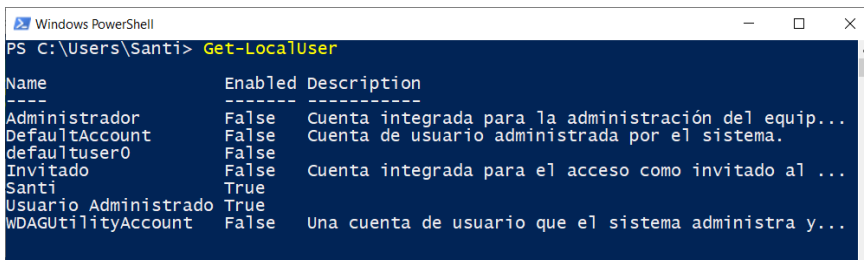
Verbo: describe la acción que se va a realizar

Nombre: Indica el objeto sobre el que se va a realizar la acción

Ejemplos de verbos: `Get`, `Set`, `Remove`, `New`...

Ejemplos de nombres: `LocalUser`, `LocalGroup`, `NetAdapter`, `Partition`...

Ejemplo de CmdLet: `PS> Get-LocalUser`



```
Windows PowerShell
PS C:\Users\Santi> Get-LocalUser

Name                Enabled Description
-----
Administrador       False  Cuenta integrada para la administración del equip...
DefaultAccount      False  Cuenta de usuario administrada por el sistema.
defaultuser0        False
Invitado            False  Cuenta integrada para el acceso como invitado al ...
Santi               True
Usuario Administrado True
WDAGUtilityAccount  False  Una cuenta de usuario que el sistema administra y...
```

Este CmdLet muestra los usuarios locales del equipo

Otros ejemplos: `Get-Date`, `Clear-Host`, `Get-Location`, `Get-ChildItem`, `Get-LocalGroup`, `Get-NetAdapter`,

`PS> Get-Command` → nos dice todos los comandos que existen (alias, funciones y CmdLet)

`PS> Get-Command -verb New` → Muestra los comandos que utilizan el verbo New

`PS> Get-Command -Noun LocalGroup` → Muestra los comandos que tienen que ver con grupos locales

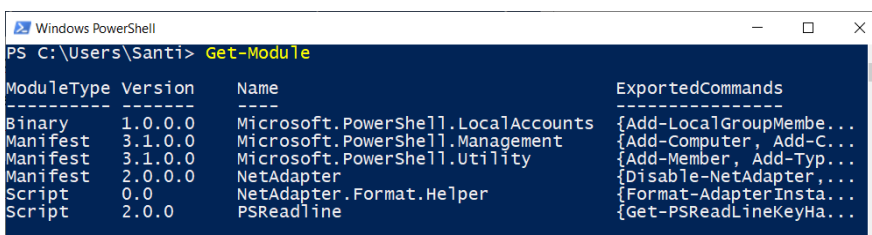
`PS> Get-Command *LocalGroup*` → Muestra todos los comandos que contienen el texto “localgroup”

### Concepto de Módulo

Un **módulo** es un conjunto de CmdLet

Los CmdLet se agrupan en conjuntos que se denominan **módulos**.

Para ver los módulos del sistema: `PS> Get-Module`



```
Windows PowerShell
PS C:\Users\Santi> Get-Module

ModuleType Version Name ExportedCommands
-----
Binary 1.0.0.0 Microsoft.PowerShell.LocalAccounts {Add-LocalGroupMembe...
Manifest 3.1.0.0 Microsoft.PowerShell.Management {Add-Computer, Add-C...
Manifest 3.1.0.0 Microsoft.PowerShell.Utility {Add-Member, Add-Typ...
Manifest 2.0.0.0 NetAdapter {Disable-NetAdapter,...
Script 0.0 NetAdapter.Format.Helper {Format-AdapterInsta...
Script 2.0.0 PSReadline {Get-PSReadLineKeyHa...
```

Para ver todos los comandos de un módulo : `PS> Get-Command -module Microsoft...`

```
Windows PowerShell
PS C:\Users\Santi> Get-Command -Module Microsoft.PowerShell.LocalAccounts
```

CommandType	Name	Version	Source
Cmdlet	Add-LocalGroupMember	1.0.0.0	Mic
Cmdlet	Disable-LocalUser	1.0.0.0	Mic
Cmdlet	Enable-LocalUser	1.0.0.0	Mic
Cmdlet	Get-LocalGroup	1.0.0.0	Mic
Cmdlet	Get-LocalGroupMember	1.0.0.0	Mic
Cmdlet	Get-LocalUser	1.0.0.0	Mic
Cmdlet	New-LocalGroup	1.0.0.0	Mic
Cmdlet	New-LocalUser	1.0.0.0	Mic
Cmdlet	Remove-LocalGroup	1.0.0.0	Mic
Cmdlet	Remove-LocalGroupMember	1.0.0.0	Mic
Cmdlet	Remove-LocalUser	1.0.0.0	Mic
Cmdlet	Rename-LocalGroup	1.0.0.0	Mic
Cmdlet	Rename-LocalUser	1.0.0.0	Mic
Cmdlet	Set-LocalGroup	1.0.0.0	Mic
Cmdlet	Set-LocalUser	1.0.0.0	Mic

No todos los módulos del sistema están disponibles, si necesitamos utilizar uno en concreto debemos importarlo. Por ejemplo: Si queremos tratar con el programa de cifrado BitLocker de Microsoft, haremos lo siguiente:

Para ver los módulos disponibles:

```
Windows PowerShell
PS C:\Users\Santi> Get-Module -ListAvailable
```

ModuleType	Version	Name	ExportedCommands
Script	1.0.1	Microsoft.PowerShell.Operation.V...	{Get-OperationValida...
Binary	1.0.0.1	PackageManagement	{Find-Package, Get-P...
Script	3.4.0	Pester	{Describe, Context, ...}
Script	1.0.0.1	PowerShellGet	{Install-Module, Fin...
Script	2.0.0	PSReadline	{Get-PSReadLineKeyHa...

Directorio: C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules

ModuleType	Version	Name	ExportedCommands
Manifest	1.0.0.0	AppBackgroundTask	{Disable-AppBackgrou...
Manifest	2.0.1.0	Appx	{Add-AppxPackage, Ge...
Manifest	1.0.0.0	BitLocker	{Unlock-BitLocker, S...
Manifest	2.0.0.0	BitsTransfer	{Add-BitsFile, Compl...

Vemos que uno de los módulos es el de BitLocker, lo importaremos:

```
Windows PowerShell
PS C:\Users\Santi> Import-Module BitLocker
ADVERTENCIA: Algunos nombres de comando importados del módulo 'BitLocker' incluyen
verbos no aprobados que podrían dificultar su reconocimiento. Para encontrar los
comandos con verbos no aprobados, vuelva a ejecutar el comando Import-Module con el
parámetro Verbose. Para obtener una lista de verbos aprobados, escriba Get-Verb.
PS C:\Users\Santi> Import-Module BitLocker -Verbose
```

Vemos que existe como módulo con el comando `Get-Module`

El módulo se puede quitar de memoria (sin eliminar) con el comando `PS> Remove-Module BitLocker`

### Obtener ayuda de un cmdlet

`Get-Help Nombre-Comando` → manera estándar

`Get-Help Nombre-Comando -examples`

`Get-Help Nombre-Comando -Details` → Detallada

`Get-Help Nombre-Comando -Full` → Completa

Ejemplo: `Get-Help New-LocalUser`

### Práctica:

Abrir consola PS como administrador

Crear usuario: `New-LocalUser -name "pelaez" -Description "Peláez el disrruptor" -Nopassword`

Comprobar que se ha creado con `Get-LocalUser`

Eliminar al usuario: Remove-LocalUser pelaez

Nota: Si esta práctica la intentamos hacer desde un servidor con AD, no estará disponible el CmdLet **New-LocalUser**. En este caso habrá que utilizar otro. Utiliza el comando Get-Command para encontrar el CmdLet necesario.

Ejemplo Ayuda online: Get-Help GET-HELP -Online

## Atajos

**Tabulador:** Completa el comando (cada palabra)

**Cursores:** Para seleccionar un comando ejecutado anteriormente

**Historial:** Se puede ver el historial de comandos ejecutados con Get-History

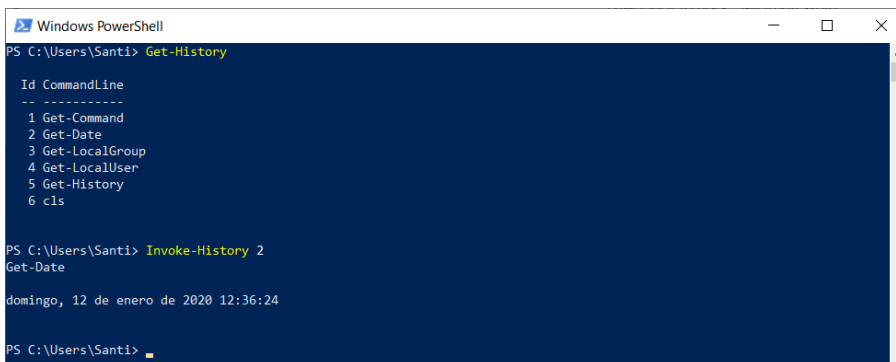


```
Windows PowerShell
PS C:\Users\Santi> Get-History

Id CommandLine
--
1 Get-Command
2 Get-Date
3 Get-LocalGroup
4 Get-LocalUser
5 Get-History
6 cls

PS C:\Users\Santi>
```

PS> Invoke-History n → Ejecuta el comando en el orden n (n debe ser el número de Id)



```
Windows PowerShell
PS C:\Users\Santi> Get-History

Id CommandLine
--
1 Get-Command
2 Get-Date
3 Get-LocalGroup
4 Get-LocalUser
5 Get-History
6 cls

PS C:\Users\Santi> Invoke-History 2
Get-Date

domingo, 12 de enero de 2020 12:36:24

PS C:\Users\Santi>
```

PS> h → Alias de get-History

PS> r 2 → Alias de Invoke-History 2

Para buscar dentro del historial: Ctrl+R **palabra** → encuentra el último CmdLet que coincide con **palabra**. Pulsando Ctrl+R sucesivamente, se ven otros comandos utilizados coincidentes. Ctrl+C cancela

PS> Clear-History → Borra el historial

## Alias

Alias: Es un apodo para referirse a un comando CmdLet

Get-Alias → muestra los alias que trae el sistema

Get-Alias -Definition Get-ChildItem → muestra los alias de ese CmdLet



```
Windows PowerShell
PS C:\Users\Santi> Get-Alias -Definition Get-ChildItem

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

PS C:\Users\Santi>
```

## Gestión de Archivos y Carpetas, Comandos fundamentales:

**Get-Location (pwd)** → Ruta o path donde estamos

**Set-Location (cd)** → cambiar a otro directorio

**New-Item (ni/md)**: Para crear archivos y carpetas.

**Remove-Item (rm)**: Para Borrar archivos y carpetas.

**Move-Item (mv)**: Para mover archivos y carpetas.

**Rename-Item (ren)**: Para renombrar archivos y carpetas.

**Copy-Item (cp)**: Para copiar archivos y carpetas

**Get-ChildItem (ls)** → contenido del directorio donde estamos

### Get-ChildItem

En caso de Get-ChildItem, los elementos tienen en la columna Mode algunas de las 6 letras siguientes:

d → directorios

a → archivos

r → sólo lectura

h → ocultos

s → sistema

l → enlace de otro archivo

Podemos usar **Get-ChildItem -Attributes hidden** → muestra solo los ocultos.

### Comandos para crear archivos y directorios **New-Item (ni)**

**New-Item carpeta01 -ItemType Directory** → md carpeta01

**New-Item .\carpeta01\fichero1.txt , .\carpeta02\fichero2.txt**

### Eliminar archivos y carpetas con **Remove-Item (rm)**

**Remove-Item .\carpeta01\ -recurse**

#### Otros:

**Move-Item \*.jpg .\fotos**

**Rename-Item .\fotos .\imagenes**

**Copy-Item .\imagenes\ copiaImagenes -recurse** (en este caso se crea la carpeta **copiaImagenes** si no existe)

**Get-Content archivo** → muestra el contenido del archivo

Alias: pwd, ls, cd, md, ni, ren, cp, cat, rm

## Tuberías y redireccionamiento.

Queremos conocer todos los archivos mayores de 4 GB y ordenados de mayor a menor. (con tuberías)

Además necesitamos que se almacene en un archivo. (con redireccionamiento).

### Tuberías:

Las tuberías permiten conectar la salida de un CmdLet con la entrada de otro, que lo tratará como su información de inicio.

Se utiliza el carácter | (tubería o pipe) para enlazar los comandos.

Ejemplos:

Get-Command | Measure-Object

```
Windows PowerShell
PS C:\Users\Santi\Desktop> Get-Command | Measure-Object

Count      : 1463
Average    :
Sum        :
Maximum    :
Minimum    :
Property   :
```

Cuenta los objetos que hay en Get-Command

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\Santi> Get-ChildItem -Recurse | Where-Object {$_.Length -gt 2500Mb}
```

Muestra los archivos que hay en es carpeta y todas las subcarpetas (-Recurse), pero únicamente cuando tengan un tamaño mayor a 2500Mb

```
Windows PowerShell
PS C:\Users\Santi> Get-ChildItem -Recurse | Where-Object {$_.Length -gt 2500Mb} | Sort-Object -Descending -Property Length
```

Como el anterior, pero además ordenados de forma descendente según su tamaño.

**Ejemplo: Para averiguar en qué puertos ha habido alguna conexión:**

```
Windows PowerShell
PS C:\Users\Santi> Get-NetTCPConnection | ft -a

LocalAddress LocalPort RemoteAddress RemotePort State      AppliedSetting OwningProcess
-----
::           60040    ::           0           Bound           10780
::           49672    ::           0           Listen          848
::           49668    ::           0           Listen          796
::           49667    ::           0           Listen          3428
::           49666    ::           0           Listen          1824
::           49665    ::           0           Listen          1460
::           49664    ::           0           Listen          684
```

Ft -a → hace un autoajuste para ver bien las columnas

Nos interesan los puertos que en la columna state tengan el valor “Established” (se pueden usar las comillas simples o dobles).

```
Windows PowerShell
PS C:\Users\Santi> Get-NetTCPConnection | Where-Object{$_. State -eq "established"}

LocalAddress      LocalPort RemoteAddress      RemotePort State
-----
192.168.1.34      60496    82.196.7.188       443      Estab...
192.168.1.34      60483    151.101.0.217      443      Estab...
192.168.1.34      60481    35.190.12.249      443      Estab...
192.168.1.34      60047    40.67.254.36       443      Estab...
192.168.1.34      60040    52.114.77.39       443      Estab...
```



### Redireccionamiento:

El redireccionamiento nos permite mandar los resultados a un lugar diferente de la pantalla. Normalmente a un archivo.

Para crear un nuevo archivo (borrando su contenido en caso de existir) se utiliza el símbolo > (mayor que), depositando en dicho archivo la salida del CmdLet

Si utilizamos el doble >> (doble mayor que), la salida del CmdLet se añade al contenido del archivo, conservando el contenido anterior. Si el archivo no existía, se crea en este momento.

Ejemplo:

```
Get-ChildItem > listado.txt
```

```
Get-Date >> listado.txt
```

```
Get-Content .\listado.txt
```

Otro ejemplo:

```
PS C:\Users\Santi> h

Id CommandLine
--
1 Get-ChildItem -Recurse | Where-Object {$_.Length -gt 2500Mb} | Sort-Object -Descending -Property ...
2 Get-ChildItem -Recurse | Where-Object {$_.Length -gt 2500Mb} | Sort-Object -Descending -Property ...
3 cls
4 cls
5 Get-NetTCPConnection
6 Get-NetTCPConnection -a
7 cls
8 Get-NetTCPConnection | ft -a
9 Get-NetTCPConnection | Where-Object{$_. State -eq "established"}
10 Get-NetTCPConnection | Where-Object{$_. State -eq 'established'}
11 cls
12 Get-NetTCPConnection | Where-Object{$_. State -eq "established"}

PS C:\Users\Santi> r 12 >>listado.txt
```