

UD7: POWERSHELL

INTRODUCCIÓN

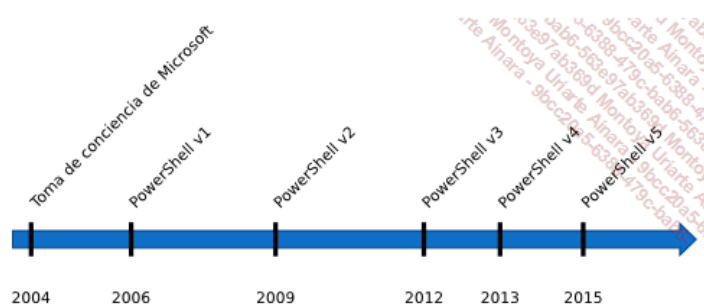
PowerShell es a la vez una interfaz de comandos (CLI) y un potente lenguaje de scripts, que trabaja con objetos.

Esta interfaz está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada.

Los comandos tienen nombres fáciles de retener, y es fácil adivinar los nombres de comandos que no conocemos. Ahora los comandos se llaman **cmdlet**

Además, existe una ayuda en línea integrada en la consola que puede consultar en cualquier momento y que es muy completa.

HISTÓRICO DE VERSIONES



Versiones soportadas	Windows 7	Windows 8	Windows 8.1	Windows 10
V2	Nativo			
V3	Necesita el SP1	Nativo		
V4	Necesita el SP1		Nativo	
V5			A instalar	Nativo

Versiones soportadas	Windows Server 2008 R2	Windows Server 2012	Windows Server 2012 R2	Windows Server 2016
V2	Nativo			
V3	Necesita el SP1	Nativo		
V4	Necesita el SP1	A instalar	Nativo	
V5		A instalar	A instalar	Nativo

Cualquiera que sea la versión de PowerShell, esta se instala siempre en la ruta **C:\Windows\System32\WindowsPowerShell\v1.0**.

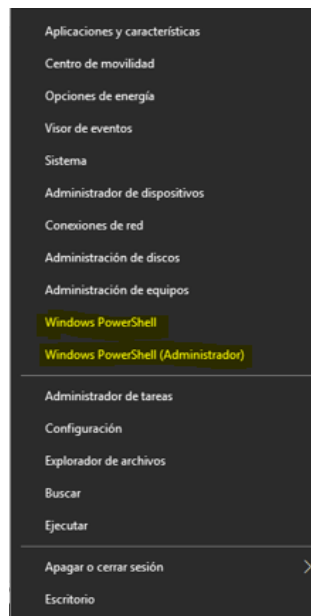
Desde la versión 2.0 existe un excelente editor de scripts PowerShell en modo gráfico llamado **PowerShell ISE**

COMENZANDO CON POWERSHELL

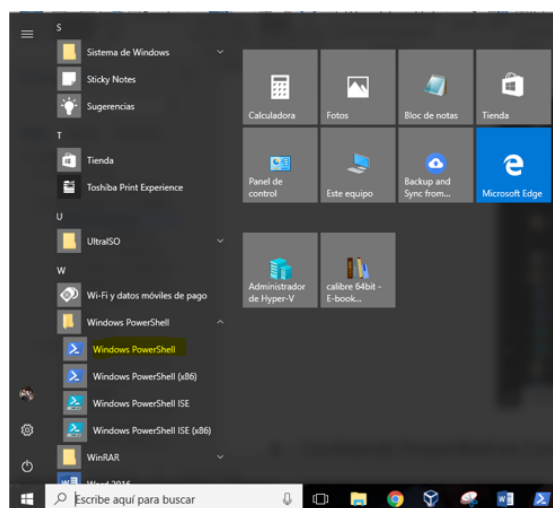
ABRIR POWERSHELL

Para abrir PowerShell:

- Botón derecho en el botón de Inicio

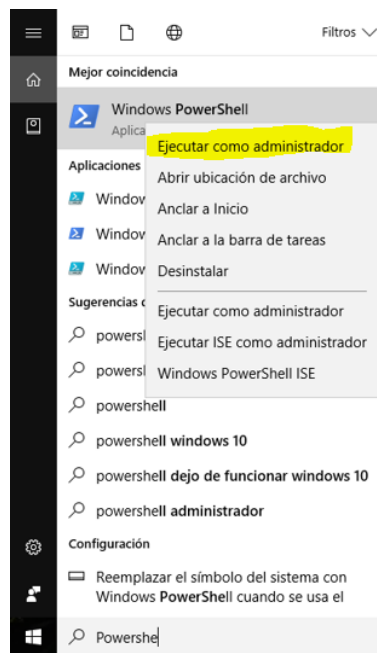


- Inicio → PowerShell

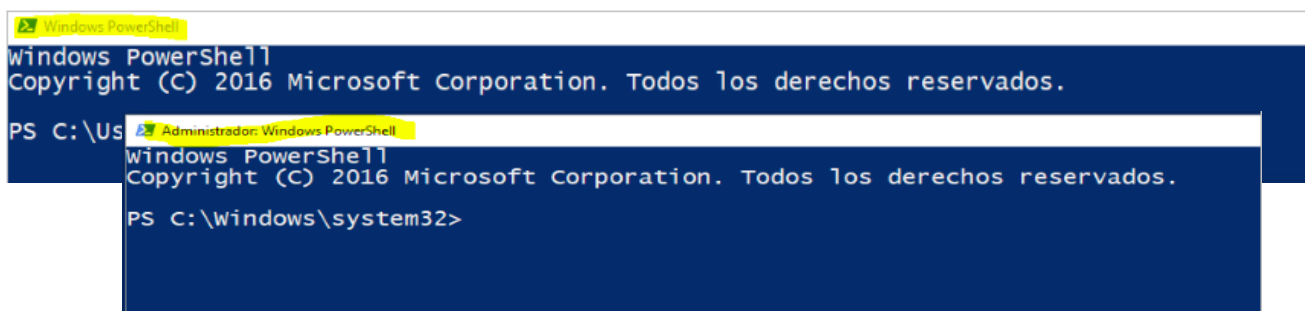


- Escribiendo PowerShell en Cortana

Cuando abrimos la consola PowerShell, esta se ejecuta con permisos simples de usuario y por lo tanto limitados (aunque estemos con un usuario administrador). Si queremos permisos de administrador sobre el icono de PowerShell pinchamos con el botón derecho y elegimos la opción ejecutar como administrador.



Verás la diferencia entre las consolas PowerShell abiertas como administrador y las que no lo son observando el título de la ventana arriba a la izquierda de estas, como se muestra aquí:



CONSOLA POWERSHELL

Tecla	Descripción
[Tab] / [Mayús]+[Tab]	Realiza el autocompletado de una ruta, nombre de comando, parámetro o valor del parámetro.
[Esc]	Borra la línea de comando actual.
[Flecha arriba] / [Flecha abajo]	Permite ver el histórico de comandos ejecutados.
[Flecha derecha] / [Flecha izquierda]	Desplaza el cursor sobre la línea de comandos actual.
[Ctrl] + [Flecha derecha]	Desplaza el cursor hacia la derecha saltando de palabra en palabra en la línea de comandos.
[Ctrl] + [Flecha izquierda]	Desplaza el cursor hacia la izquierda saltando de palabra en palabra en la línea de comandos.
[Inicio]	Devuelve el cursor al inicio de la línea de comandos.
[Fin]	Devuelve el cursor al final de la línea de comandos.
[Ctrl] + C	Pone fin a la ejecución del comando actual.

VERSIÓN DE POWERSHELL

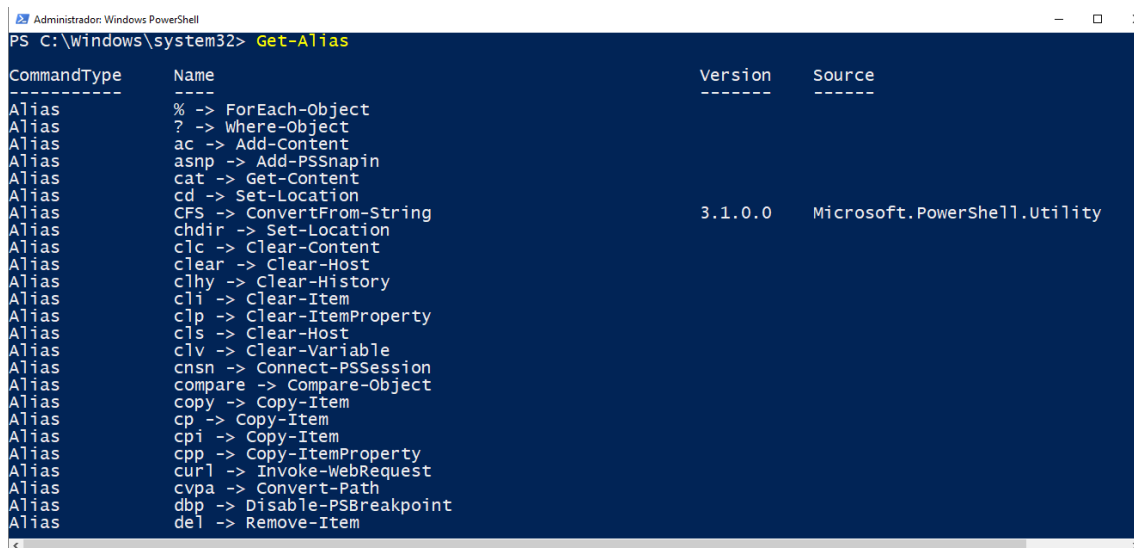
Para saber qué versión de PowerShell que se está ejecutando podemos usar el comando **Get-Host**

```
Name           : ConsoleHost
Version        : 5.1.15063.632
InstanceId     : 9b83dea6-666a-49b9-b23c-3779d5c0508a
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
```

UNA TRANSICIÓN SUAVE

Prácticamente todos los comandos que se encontraban en CMD lo están también en PowerShell; mediante alias y también algunos de la Shell de Linux

Para conocer la lista completa de alias disponibles, puedes usar el comando **Get-Alias**.



```
Administrador: Windows PowerShell
PS C:\Windows\system32> Get-Alias

CommandType      Name                                Version      Source
-----
Alias            % -> ForEach-Object
Alias            ? -> Where-Object
Alias            ac -> Add-Content
Alias            asnp -> Add-PSSnapin
Alias            cat -> Get-Content
Alias            cd -> Set-Location
Alias            CFS -> ConvertFrom-String
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
Alias            cpi -> Copy-Item
Alias            cpp -> Copy-ItemProperty
Alias            curl -> Invoke-WebRequest
Alias            cvpa -> Convert-Path
Alias            dbp -> Disable-PSBreakpoint
Alias            del -> Remove-Item
```

AYUDA

Para obtener ayuda tenemos el comando **Get-Help**, para asegurarnos de que tenemos la última versión de la ayuda podemos actualizarla con el comando **Update-Help**

GET-HELP

El comando **Get-Help** es para visualizar la ayuda de Powershell

PARÁMETRO	DESCRIPCIÓN
-Name	Muestra ayuda sobre un tema conceptual o cmdlet que le especifiquemos. Se permite el uso de caracteres comodín. Se puede omitir el nombre del parámetro (-Name). Get-Help Get-Command → muestra información sobre el commando Get-Command Get-Help -Name Get-Command → muestra información sobre el comando Get-Command Get-Help -Name about_Arithmetic_Operators → muestra información sobre los operadores aritméticos. Get-Help about_Arithmetic_Operators → muestra información sobre los operadores aritméticos.
-Detailed	Muestra información adicional.
-Examples	Muestra únicamente el apartado de ejemplos de la ayuda.
-Full	Muestra la ayuda completa.

CARACTERES COMODÍN

CARÁCTER	DESCRIPCIÓN
*	Sustituye a cualquier cadena de caracteres
?	Sustituye a un único caracter

REDIRECCIÓN

Por defecto Powershell dirige la salida de comandos a la pantalla, pero podemos modificar este comportamiento dirigiendo la salida a un archivo de texto. Para hacer esto, tenemos que utilizar el símbolo "mayor que" (>) y el nombre del archivo al que vamos a redirigir la salida.

Get-Help -Detailed Get-Command > C:\get-command.txt → este comando crea un fichero de texto en la ruta especificada cuyo contenido es la ayuda (detallada) del comando Get-Command.

Al redireccionar la salida a un fichero de texto si este no existe lo crea y en caso de que exista elimina su contenido para introducir el nuevo texto que le enviamos. Si no queremos hacer esto y lo que queremos es añadir nuevos datos al fichero, utilizaremos el símbolo ">>".

TUBERÍAS

Para encadenar la salida de un comando con la entrada de otro comando tenemos las tuberías "|". Sobre el objeto u objetos que devuelve el primer parámetro después de ejecutarse, se aplicará el segundo comando y así sucesivamente si encadenáramos más de dos comandos.

Get-Command -Verb Set | Get-Help -Examples → el primer comando nos devuelve una lista de todos los comandos cuyo verbo es Set. Le pasamos esta lista al segundo comando y nos devolverá el apartado Examples de la ayuda de cada uno de los comandos cuyo verbo es Set.

RUTAS

Definir una ruta es indicar los directorios que se deben recorrer hasta llegar al elemento deseado, ya sea este un archivo o un directorio. Para ello tenemos que especificar el nombre de la unidad así como los diferentes directorios (en orden secuencial) hasta acceder al archivo destino. Para separar y diferenciar directorios y archivos, utilizamos el carácter \.

Las rutas pueden ser de dos tipos:

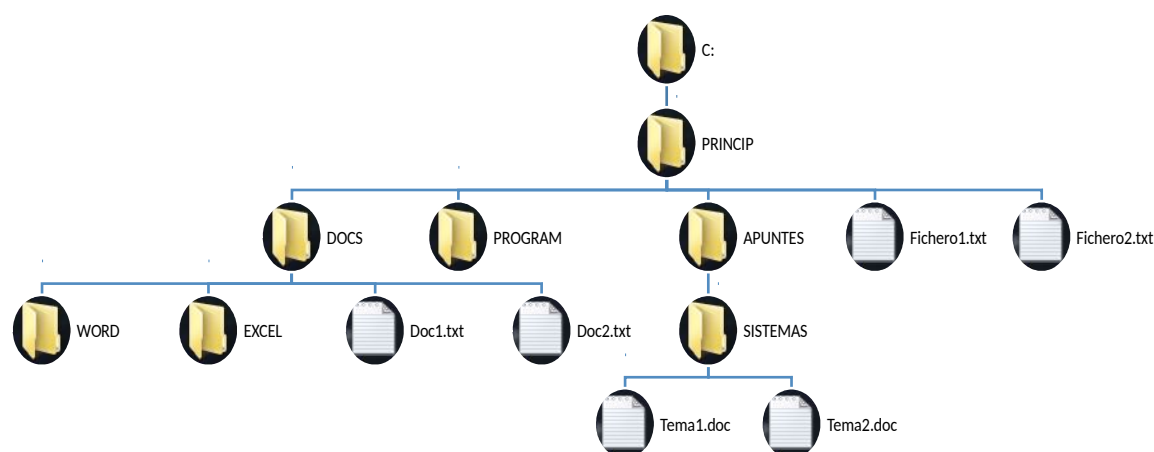
- **Rutas absolutas:** parten de la unidad independientemente del directorio activo (en el directorio en el que este el usuario).
- **Rutas relativas:** parten del directorio activo.

Al crear un directorio, automáticamente se generan dentro de él dos subdirectorios:

- . (hace referencia al contenido del propio directorio)
- .. (hace referencia al directorio padre)

Ejemplo:

Suponiendo que tenemos la siguiente estructura de directorios y que estamos en el directorio EXCEL indica la trayectoria absoluta y relativa para llegar al directorio SISTEMAS



- Ruta absoluta: C:\PRINCIP\APUNTES\SISTEMAS
- Ruta relativa: ..\..\APUNTES\SISTEMAS

COMANDOS BÁSICOS

ESTRUCTURA DE LOS COMANDOS

Los comandos de PowerShell con llamado **cmdlets**. Su formato es un verbo y un sustantivo separados por un guion (**verbo-sustantivo**). Por ejemplo: **Get-Command**.

El verbo describe la acción a realizar sobre el nombre. En el anterior ejemplo, recuperamos (**Get**) los comandos (**Command**).

Los nombres que constituyen los comandos están siempre en singular; y esto es válido también para los parámetros. Además, los comandos, así como sus parámetros asociados, pueden escribirse indistintamente en mayúsculas o en minúsculas

GET-COMMAND

El comando **Get-Command** obtiene información básica acerca de los comandos y otros elementos de Windows Powershell (alias, funciones...)

PARÁMETRO	DESCRIPCIÓN
-Name	Obtiene información únicamente de los comandos o elementos de comando con el nombre especificado. Se permite el uso de caracteres comodín. Se puede omitir el nombre del parámetro (-Name).
-Verb	Obtiene información básica acerca de los comandos con cuyo verbo sea el especificado.
-Type	Obtiene información solo de objetos especificados: function, alias, cmdlet, application...

Ejemplos:

Get-Command → muestra una lista de todos los comandos que tenemos disponibles en Windows Powershell. Los ordena en varias columnas.

Get-Command -Verb Set → muestra los comandos con el verbo "set".

Get-Command -Name *Item* → muestra todos los comandos que tienen la palabra Item en su nombre

SET-LOCATION (SL, CD o CHDIR)

Este comando nos permite cambiar de ubicación dentro de nuestra jerarquía de directorios

Set-Location D: → nos posiciona en la unidad D

Podemos utilizar rutas relativas o absolutas.

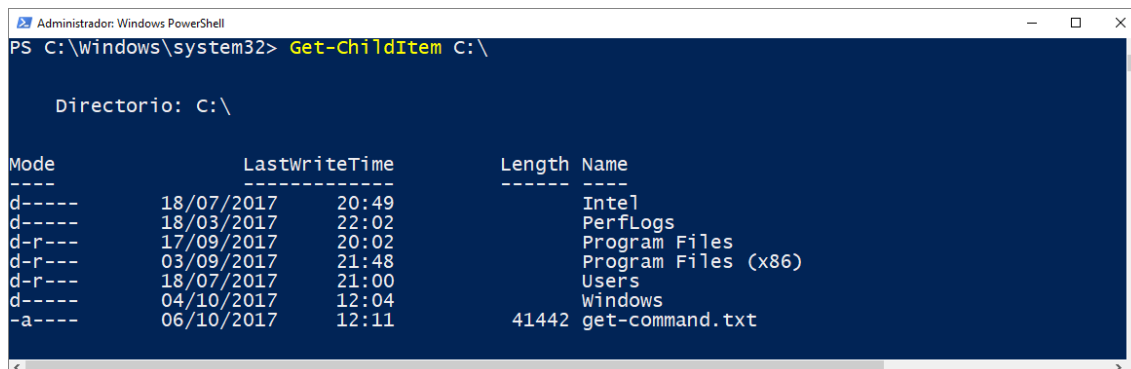
GET-LOCATION (GL o PWD)

Este comando muestra la ruta completa del directorio en el que estamos.

```
PS C:\users\Administrador> Get-Location
Path
----
C:\users\Administrador
```

GET-CHILDITEM (GCI, LS o DIR)

Este comando permite obtener o buscar los archivos y carpetas presentes en el sistema de archivos de una o varias ubicaciones especificadas.



```
Administrador: Windows PowerShell
PS C:\windows\system32> Get-Childitem C:\

Directorio: C:\

Mode                LastWriteTime         Length Name
----                -
d-----          18/07/2017    20:49             Intel
d-----          18/03/2017    22:02             PerfLogs
d-r-----        17/09/2017    20:02             Program Files
d-r-----        03/09/2017    21:48             Program Files (x86)
d-r-----        18/07/2017    21:00             Users
d-----         04/10/2017    12:04             windows
-a-----         06/10/2017    12:11         41442 get-command.txt
```

En la columna Mode se indica la naturaleza de los objetos en el interior del sistema de archivos:

- **d:** para un directorio.
- **a:** para un archivo.
- **r:** para un objeto de solo lectura.
- **h:** para un objeto oculto.
- **s:** para un objeto de sistema.

PARÁMETRO	DESCRIPCIÓN
-Path	Especifica una o más rutas de ubicaciones. Se permite el uso de caracteres comodín. La ubicación predeterminada es el directorio actual (.). Se puede omitir el nombre del parámetro (-Path)..
-Include	Recupera únicamente los elementos especificados. Se puede escribir en este apartado un patrón como "*.exe".
-Exclude	Omite los elementos especificados. Se puede usar un patrón.
-Depth	Indica el número de niveles (hijos) a recorrer
-Name	Recupera únicamente los nombres de los elementos.
-Recurse	Obtiene los elementos y elementos hijos de las ubicaciones especificadas.
-Force	Muestra también los archivos ocultos.
-Attribute	Muestra archivos o carpetas en función de la naturaleza de los objetos buscados. Los posibles valores son: <ul style="list-style-type: none">• Directory: para una carpeta.• Archive: para un archivo.• Hidden: para un objeto oculto.• ReadOnly: para un objeto de solo lectura.• System: para un objeto de sistema. Podemos asociar combinaciones de atributos con operadores: <ul style="list-style-type: none">• +: para representar un Y lógico.• ,: para representar un O lógico.• !: para representar un NO lógico.

Ejemplos:

Get-ChildItem → muestra un listado del contenido de la ubicación actual.

Get-ChildItem -Recurse → Recupera los elementos del directorio actual y de sus subdirectorios.

Get-ChildItem -Include *.txt -Recurse → Igual que el ejemplo anterior, pero solo mostrara los ficheros con extensión txt.

Get-ChildItem -Path C:\Windows -Include *.txt -Recurse → Nos muestra un listado de los ficheros con extensión txt, que hay en el directorio C:\Windows y en sus subdirectorios.

Get-ChildItem -Path C:\, D:\ -Include *.txt -Recurse > C:\listado.txt → Hace un listado de los ficheros con extensión txt que hay en los volúmenes C y D (y en sus subdirectorios) y crea un fichero listado.txt con ese contenido.

Get-ChildItem | Where-Object {\$_.Length -gt 32KB} → Muestra los archivos cuyo tamaño es superior a 32 KB

Get-ChildItem C:\ -Attributes Hidden+!Directory → muestra los archivos ocultos y aquellos que no son carpetas, basta con combinar los atributos «oculto» y «no es una carpeta».

NEW-ITEM (NI o MD)

Este comando permite crear carpetas y archivos.

PARÁMETRO	DESCRIPCIÓN
-Path	Ruta de acceso del elemento a crear
-ItemType	Tipo de elemento a crear: File o Directory .
-Name	Nombre del nuevo elemento.
-Value	Contenido del elemento a crear (p.e.: "¡hola!" en el caso de un archivo de texto).

Ejemplos:

New-Item -ItemType Directory -Name Temp → Crea el directorio Temp en el directorio actual.

New-Item -Name miArchivo.txt -ItemType File → Crea el fichero miArchivo.txt en la ubicación actual.

New-Item -Name miArchivo.txt -ItemType File -Path "C:/" -Value "Mi contenido" → Crea el fichero miArchivo.txt en C:/ con el contenido "Mi contenido".

REMOVE-ITEM (RI, RM, RMDIR, RD, ERASE o DEL)

Este comando permite eliminar archivos o carpetas.

Ejemplos:

Remove-Item C:\Temp*.log → borra los ficheros con extensión log del directorio C:\Temp

Get-ChildItem C:\Temp* -Include *.txt -Recurse | Remove-Item → elimina todos los archivos con extensión txt de C:\Temp y de sus subdirectorios Esta sintaxis es muy práctica ya que podemos construirla poco a poco; primero enumeramos los archivos que queremos suprimir, para después pasarlos mediante una tubería al comando que los va a borrar.

Remove-Item posee también el parámetro **-WhatIf**; este indica lo que hará el comando, pero sin ejecutarlo (modo de simulación). Otro parámetro interesante es **-Confirm**, el cual pide confirmación para cada archivo a suprimir.

MOVE-ITEM (MI, MOVE o MV)

Este comando permite mover un archivo o una carpeta de un sitio a otro y renombrarlo.

Ejemplos:

Move-Item -Path *.jpg -Destination "Mis fotos" → Mueve los archivos de extensión jpg de la carpeta actual a la carpeta "Mis fotos" (tiene que existir).

Move-Item "Mis fotos" "Mis nuevas fotos" → Si la carpeta "Mis nuevas fotos" existe moverá el contenido de la carpeta "Mis fotos" a la carpeta "Mis nuevas fotos"; si la carpeta "Mis nuevas fotos" no existe renombra la carpeta "Mis fotos" a "Mis nuevas fotos".

RENAME-ITEM (REN o RNI)

Este comando renombra un archivo o una carpeta.

Ejemplos:

Rename-Item -Path C:\Temp\miArchivoDeLog.txt -NewName archlog.txt → Renombra el archivo miArchivoDeLog.txt a archlog.txt

Rename-Item -Path C:\Temp\MiCarpeta1 -Newname MiCarpeta2 → renombra la carpeta MiCarpeta1 a MiCarpeta2

COPY-ITEM (CPI, CP o COPY)

Este comando copia archivos y carpetas.

Ejemplos:

Copy-Item -Path C:\Temp\ficLog.txt -Destination D:\Logs → Copia el fichero ficLog.txt de la carpeta C:\Temp a la carpeta D:\Logs

Copy-Item C:\Temp\ficLog.txt D:\Logs → Hace lo mismo que el comando anterior.

Copy-Item -Path DirSource -Destination DirDest -Recurse → Copia el directorio DirSource y todo su contenido con el nombre DirDest

FORMATEANDO LA SALIDA

Vamos a ver unos comandos que nos van a servir para formatear la salida de otros comandos.

FORMAT-LIST (FL)

Este comando aplica a la salida el formato de una lista de propiedades en la que cada propiedad aparece en una línea diferente.

PARÁMETRO	DESCRIPCIÓN
-Property	Especifica las propiedades del objeto que se van a mostrar y el orden en el que van a aparecer.
-View	Especifica el nombre de una vista o un formato de lista alternativo. Los parámetros -Property y View no se pueden utilizar en el mismo comando

Ejemplos:

Get-Childitem

```
Seleccionar Administrador: Windows PowerShell
PS C:\Temp> Get-ChildItem

Directorio: C:\Temp

Mode                LastWriteTime         Length Name
----                -
d-----          07/10/2017    0:08           DirDest
d-----          07/10/2017    0:08         DirDest1
d-----          07/10/2017    0:03       Nueva carpeta
-a-----          06/10/2017   17:41       628470 Logs
-a-----          06/10/2017   18:27         12 miArchivo - copia.txt
-a-----          06/10/2017   18:27         12 miArchivo.txt
```

Get-Childitem | Format-List

```
Seleccionar Administrador: Windows PowerShell
PS C:\Temp> Get-ChildItem | Format-List

Directorio: C:\Temp

Name                : DirDest
CreationTime        : 07/10/2017 0:08:37
LastWriteTime       : 07/10/2017 0:08:37
LastAccessTime      : 07/10/2017 0:08:37
Mode                : d-----
LinkType            :
Target              : {}

Name                : DirDest1
CreationTime        : 07/10/2017 0:08:55
LastWriteTime       : 07/10/2017 0:08:55
LastAccessTime      : 07/10/2017 0:08:55
Mode                : d-----
LinkType            :
Target              : {}

Name                : Nueva carpeta
CreationTime        : 06/10/2017 23:32:17
LastWriteTime       : 07/10/2017 0:03:00
```

Get-Childitem | Format-List -Property Name, Length → el resultado es un listado formateado en el que se muestran las propiedades Name y Length.

```
Administrador: Windows PowerShell

PS C:\Temp> Get-Childitem | Format-List -Property Name, Length

Name : DirDest
Name : DirDest1
Name : Nueva carpeta
Name : Logs
Length : 628470
Name : miArchivo - copia.txt
Length : 12
Name : miArchivo.txt
Length : 12
```

FORMAT-WIDE (FW)

Este comando aplica a los objetos el formato de una tabla ancha en la que se muestra únicamente una propiedad de cada objeto. Podemos utilizar el parámetro “property” para determinar que propiedad se va a mostrar.

PARÁMETROS	DESCRIPCIÓN
-Property	Especifica la propiedad del objeto que se va a mostrar
-Autosize	Ajusta el tamaño de columna en función del ancho de los datos.
-Column	Los parámetros -Autosize y -Column son incompatibles. Especifica el número de columnas de la presentación.

Ejemplos:

Get-Childitem | Format-Wide

```
Administrador: Windows PowerShell

PS C:\Temp> Get-Childitem | Format-Wide

Directorio: C:\Temp

DirDest                               DirDest1
Nueva carpeta                         Logs
miArchivo - copia.txt                 miArchivo.txt
```

Get-Childitem | Format-Wide -Column 3 → en este caso le indicamos que la salida la formatee en 3 columnas.

```

Administrador: Windows PowerShell

PS C:\Temp> Get-Childitem | Format-Wide -Column 3

Directorio: C:\Temp

DirDest          DirDest1          Nueva carpeta
Logs             miArchivo - copia.txt  miArchivo.txt

```

FORMAT-TABLE (FT)

Este comando aplica a la salida de un comando el formato de una tabla con las propiedades seleccionadas del objeto en cada columna.

PARÁMETROS	DESCRIPCIÓN
-Property	Especifica las propiedades del objeto que se van a mostrar y el orden en el que van a aparecer.
-AutoSize	Ajusta el tamaño de columna y el número de columnas en función del ancho de los datos.

Ejemplos:

Get-Childitem | Format-Table → en este caso podemos ver que la salida es la misma que si no hubiésemos utilizado Format-Table.

```

Administrador: Windows PowerShell

PS C:\Temp> Get-Childitem | Format-Table

Directorio: C:\Temp

Mode                LastWriteTime         Length Name
-----
d-----          07/10/2017         0:08      DirDest
d-----          07/10/2017         0:08      DirDest1
d-----          07/10/2017         0:03      Nueva carpeta
-a-----          06/10/2017        17:41    628470      Logs
-a-----          06/10/2017        18:27         12      miArchivo - copia.txt
-a-----          06/10/2017        18:27         12      miArchivo.txt

```

Get-Childitem | Format-Table -Property Name, Length → muestra una table con dos columnas una para la propiedad Name y otra para la propiedad Length.

```

Administrador: Windows PowerShell

PS C:\Temp> Get-Childitem | Format-Table -Property Name, Length

Name                Length
-----
DirDest
DirDest1
Nueva carpeta
Logs                628470
miArchivo - copia.txt 12
miArchivo.txt        12

```

MÓDULOS

Un módulo es una especie de contenedor (package) que agrupa comandos, pero también scripts, variables, alias y funciones. Un módulo es fácilmente transportable (basta con una simple copia del archivo) y por lo tanto también fácil de compartir para que otros usuarios puedan disfrutarlo. La idea del equipo de PowerShell de Microsoft es crear una gran comunidad de usuarios y hacer que esta pueda intercambiar o compartir sus módulos.

INSTALAR UN MÓDULO

Con Windows Server, con cada rol que instalamos se instala automáticamente su módulo asociado. En caso de querer instalar un módulo de terceros, simplemente hay que copiarlo en el directorio Modules de PowerShell: ()

Ruta	Utilización
C:\Users\nombre_usuario\Documents\WindowsPowerShell\Modules	Contiene los módulos «de usuario»
C:\Windows\System32\WindowsPowerShell\v1.0\Modules	Contiene todos los módulos «de sistema». Es aquí donde encontraremos los módulos por defecto o los instalados después de habilitar un rol en una versión Windows Server.

ENUMERAR E IMPORTAR LOS MÓDULOS

Para conocer los módulos ya importados, PowerShell dispone del comando **Get-Module**.

Parámetro	Descripción
-All	Sin este parámetro, se enumera un único objeto por módulo (es decir sin las dependencias).
-ListAvailable	Permite obtener todos los módulos instalados, pero no importados en la sesión.
-Name	Permite obtener únicamente el o los módulos especificados.

```

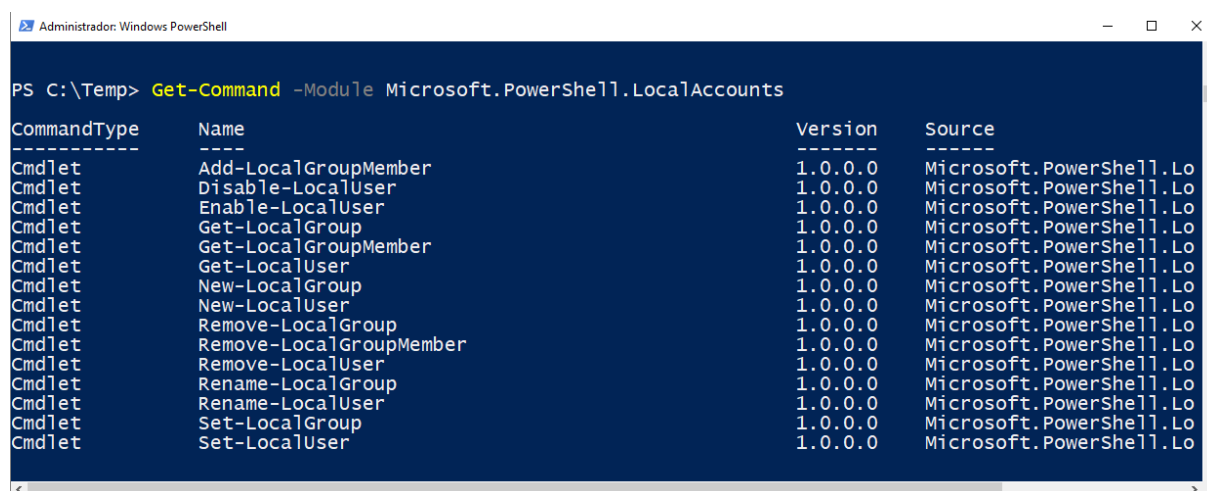
Administrador: Windows PowerShell
PS C:\Temp> Get-Module

ModuleType Version      Name                                     ExportedCommands
-----
Manifest 1.0.0.0      AppBackgroundTask                      {Disable-AppBackgroundTaskDiagnosticLog, En
Manifest 2.0.0.0      AppLocker                              {Get-AppLockerFileInformation, Get-AppLocke
Manifest 1.0.0.0      AppvClient                             {Add-AppvClientConnectionGroup, Add-AppvCli
Binary 2.0.0.0      Appx                                    {Add-AppxPackage, Add-AppxVolume, Dismount-
Script 1.0.0.0      AssignedAccess                         {Clear-AssignedAccess, Get-AssignedAccess,
Manifest 2.0.0.0      BitsTransfer                           {Add-BitsFile, Complete-BitsTransfer, Get-B
Manifest 1.0.0.0      BranchCache                            {Add-BCDataCacheExtension, Clear-BCCache, D
Binary 1.0.0.0      CimCmdlets                             {Export-BinaryMiLog, Get-CimAssociatedInsta
Binary 1.0      ConfigCI                               {Add-SignerRule, ConvertFrom-CIPolicy, Edit
Manifest 1.0      Defender                                {Add-MpPreference, Get-MpComputerStatus, Ge
Manifest 1.0.0.0      DirectAccessClientComponents           {Disable-DAManualEntryPointSelection, Enabl
Script 3.0          Dism                                    {Add-AppxProvisionedPackage, Add-WindowsCap
Manifest 1.0.0.0      DnsClient                             {Resolve-DnsName, Add-DnsClientNrptRule, Cl
Manifest 1.0.0.0      EventTracingManagement                {Add-EtwTraceProvider, Get-AutoLoggerConfig
Manifest 1.0.0.0      HgsClient                             {Get-HgsAttestationBaselinePolicy, ConvertT
Binary 2.0.0.0      Hyper-V                               {Add-VMAssignableDevice, Add-VMHvDrive, Ad
Manifest 2.0.0.0      International                          {Get-WinAcceptLanguageFromLanguageListOptou
Manifest 1.0.0.0      iSCSI                                  {Connect-IscsiTarget, Disconnect-IscsiTarge
Binary 1.0.0.0      Kds                                    {Add-KdsRootKey, Clear-KdsCache, Get-KdsCon
Binary 1.0.0.0      Microsoft.PowerShell.LocalAccounts     {Add-LocalGroupMember, Disable-LocalUser, E
  
```

Por defecto si se usa algún comando de un módulo este se carga automáticamente, aunque también existe el comando **Import-Module** <nombre módulo> para importar un módulo manualmente.

ENUMERAR LOS COMANDOS DE UN MÓDULO

El comando **Get-Command** y el parámetro **-Module** nos permite ver los elementos (comandos, métodos, propiedades...) de un módulo.



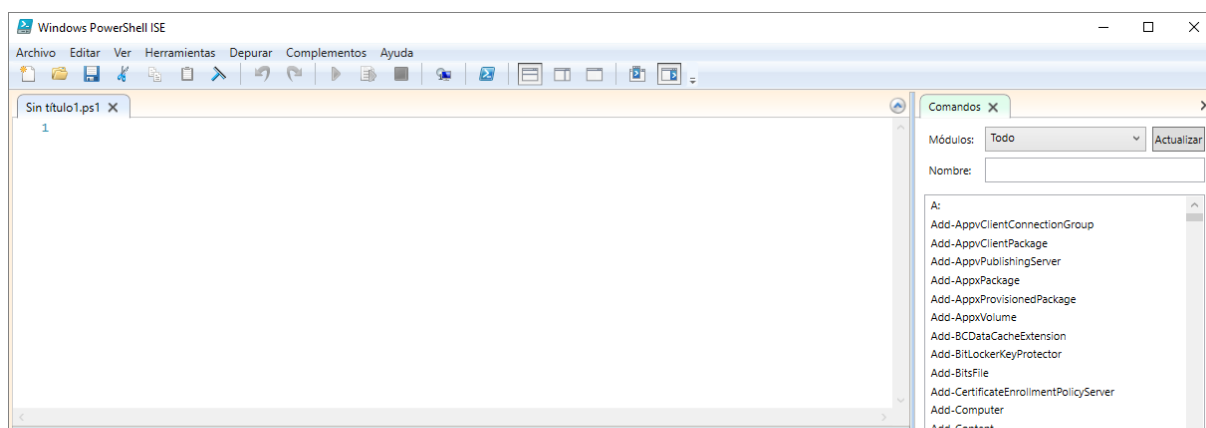
```
PS C:\Temp> Get-Command -Module Microsoft.PowerShell.LocalAccounts
```

CommandType	Name	Version	Source
Cmdlet	Add-LocalGroupMember	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Disable-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Enable-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Get-LocalGroup	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Get-LocalGroupMember	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Get-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	New-LocalGroup	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	New-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Remove-LocalGroup	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Remove-LocalGroupMember	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Remove-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Rename-LocalGroup	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Rename-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Set-LocalGroup	1.0.0.0	Microsoft.PowerShell.Lo
Cmdlet	Set-LocalUser	1.0.0.0	Microsoft.PowerShell.Lo

SCRIPTS

EL ENTORNO INTEGRADO DE ESCRITURA DE SCRIPTS (ISE)

Existe un editor de scripts en modo gráfico: PowerShell ISE (se puede acceder a él desde el menú Inicio o desde Cortana)

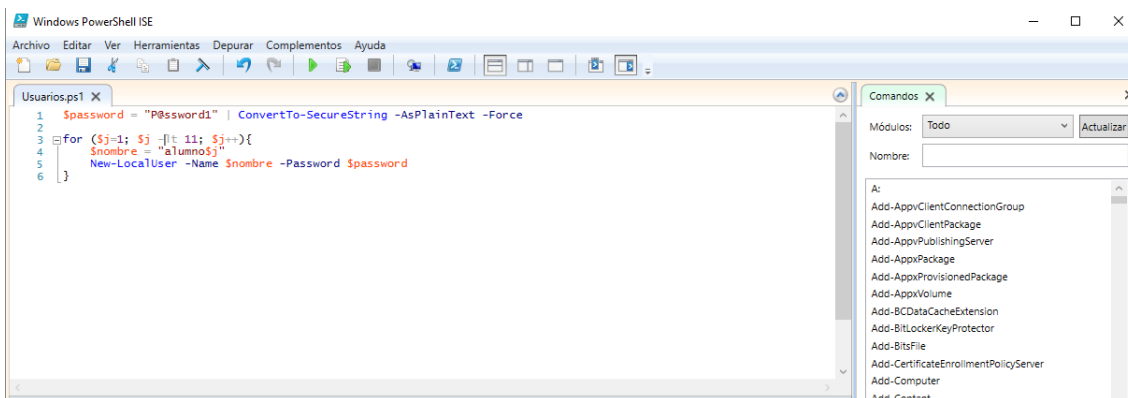


Ejemplo:

Antes de poder ejecutar un script creado por nosotros o por terceros y hay activar una directiva que lo permita:

C:\Users\usuario\Set-ExecutionPolicy Unrestricted

Vamos a crear un script que cree los usuarios: alumno1, ..., alumno10 con contraseña P@ssword1 para todos ellos.



Guardamos el script como usuarios.ps1, los scripts de PowerShell se guardan con la extensión ps1. Por último, nos queda ejecutar el script.



GESTIÓN DE CUENTAS DE USUARIO Y GRUPOS LOCALES DESDE POWERSHELL

USUARIOS

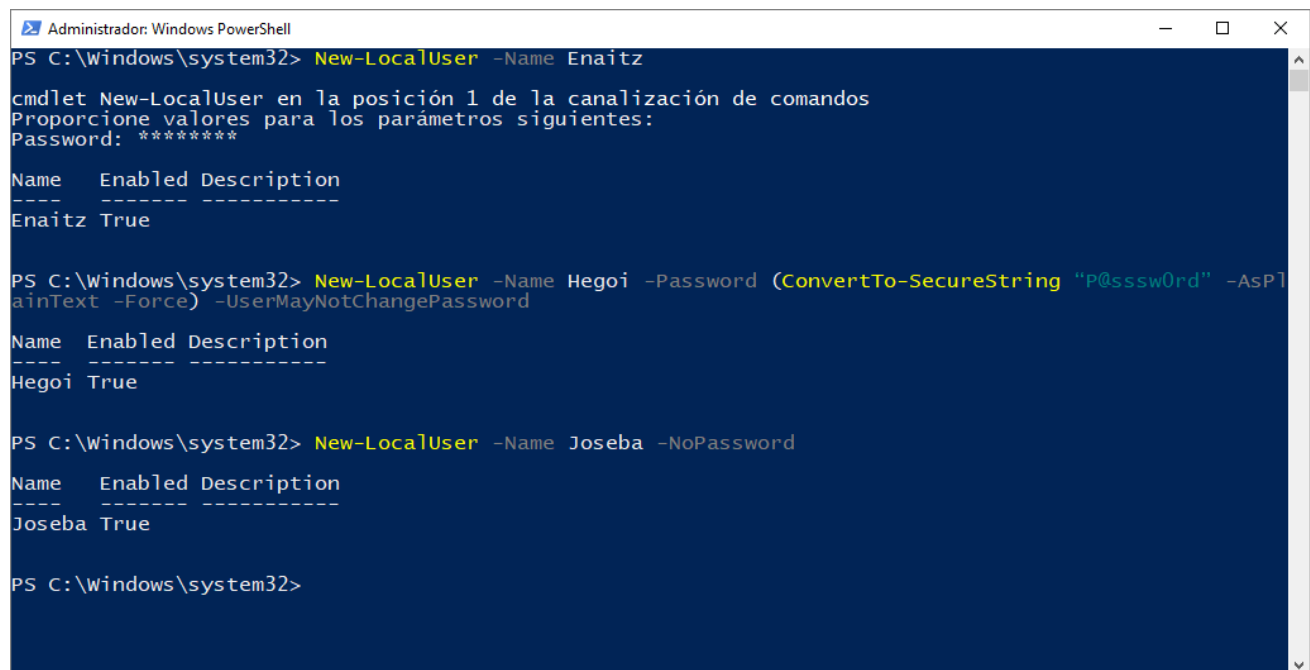
El comando *New-LocalUser* nos permite crear una cuenta de usuario local:

New-LocalUser

- [-Name] <String>*
- [-AccountExpires <DateTime>]*
- [-AccountNeverExpires]*
- [-Description <String>]*
- [-Disabled]*
- [-FullName <String>]*
- [-Password <SecureString>]*
- [-NoPassword]*
- [-PasswordNeverExpires]*
- [-UserMayNotChangePassword]*

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

Ejemplos:



```
PS C:\Windows\system32> New-LocalUser -Name Enaitz

cmdlet New-LocalUser en la posición 1 de la canalización de comandos
Proporcione valores para los parámetros siguientes:
Password: *****

Name      Enabled Description
----      -
Enaitz    True

PS C:\Windows\system32> New-LocalUser -Name Hegoi -Password (ConvertTo-SecureString "P@ssw0rd" -AsPlainText -Force) -UserMayNotChangePassword

Name      Enabled Description
----      -
Hegoi     True

PS C:\Windows\system32> New-LocalUser -Name Joseba -NoPassword

Name      Enabled Description
----      -
Joseba    True

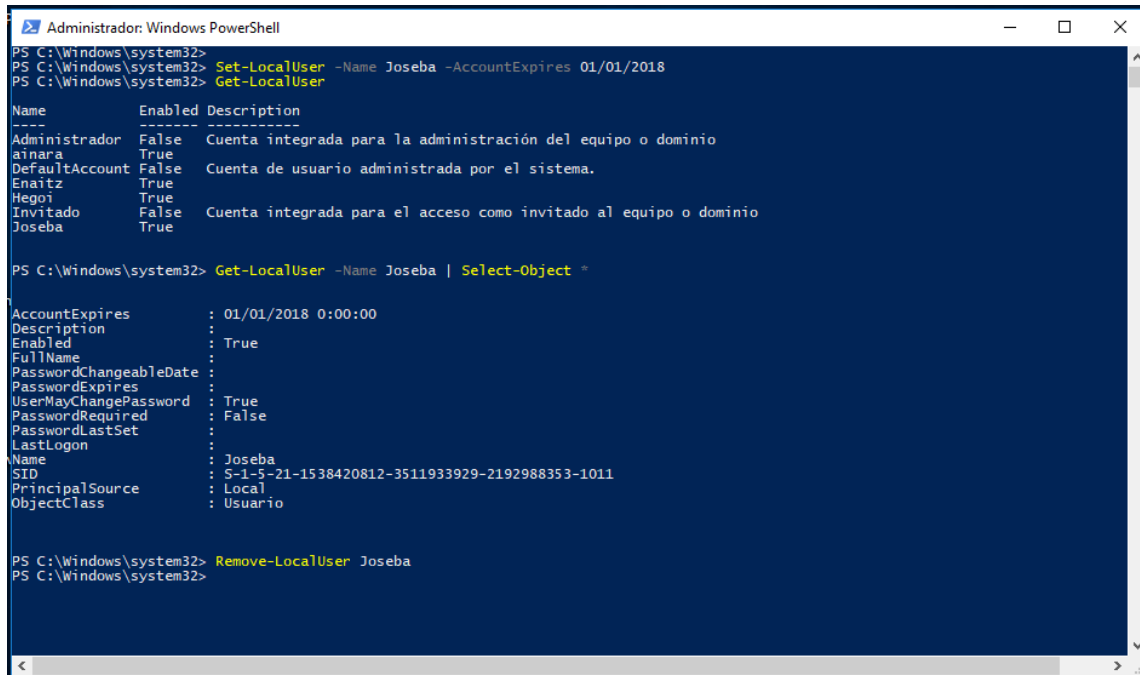
PS C:\Windows\system32>
```

Otros comandos para gestionar los usuarios locales son:

- *Remove-LocalUser*: para eliminar un usuario local
- *Get-LocalUser*: lista los usuarios locales
- *Set-LocalUser*: para modificar un usuario local
- *Disable-LocalUser*: deshabilita un usuario local
- *Enable-LocalUser*: habilita un usuario local
- *Rename-LocalUser*: cambia el nombre de un usuario local

<https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/?view=powershell-5.1>

Ejemplos:



```
Administrador: Windows PowerShell
PS C:\Windows\system32>
PS C:\Windows\system32> Set-LocalUser -Name Joseba -AccountExpires 01/01/2018
PS C:\Windows\system32> Get-LocalUser

Name      Enabled Description
----      -
Administrador False  Cuenta integrada para la administración del equipo o dominio
ainara     True   Cuenta de usuario administrada por el sistema.
DefaultAccount False  Cuenta de usuario administrada por el sistema.
Enaitz     True   Cuenta de usuario administrada por el sistema.
Hegoi      True   Cuenta de usuario administrada por el sistema.
Invitado   False  Cuenta integrada para el acceso como invitado al equipo o dominio
Joseba     True   Cuenta integrada para el acceso como invitado al equipo o dominio

PS C:\Windows\system32> Get-LocalUser -Name Joseba | Select-Object *

AccountExpires      : 01/01/2018 0:00:00
Description         : 
Enabled             : True
FullName            : 
PasswordChangeableDate : 
PasswordExpires     : 
UserMayChangePassword : True
PasswordRequired    : False
PasswordLastSet     : 
LastLogon           : 
Name                : Joseba
SID                 : S-1-5-21-1538420812-3511933929-2192988353-1011
PrincipalSource     : Local
ObjectClass         : Usuario

PS C:\Windows\system32> Remove-LocalUser Joseba
PS C:\Windows\system32>
```

GRUPOS

Para gestionar los grupos tenemos los comandos:

- *New-LocalGroup*: crea un grupo local
- *Set-LocalGroup*: modifica un grupo local
- *Remove-LocalGroup*: elimina un grupo local
- *Get-LocalGroup*: lista los grupos locales
- *Rename-LocalGroup*: cambia de nombre a un grupo local
- *Add-LocalGroupMember*: añade usuarios a un grupo local
- *Get-LocalGroupMember*: lista los usuarios de un grupo local
- *Remove-localGroupMember*: elimina usuario de un grupo local

INSTALACIÓN Y CONFIGURACIÓN DE ACTIVE DIRECTORY

Vamos crear un dominio con Powershell

CAMBIAMOS EL NOMBRE DEL EQUIPO

Para cambiar el nombre de un equipo hay que usar el comando

Rename-Computer Rename-Computer -NewName WS2012

CONFIGURAMOS LA RED

Para establecer una IP fija usamos el comando **New-NetIPAddress** y **Set-DNSClientServerAddress**
Primero vamos a mirar la configuración actual con el comando **Get-NetIPAddress**:

New-NetIPAddress -InterfaceIndex 12 -IPAddress 192.168.1.1 -PrefixLength 24 -DefaultGateway 192.168.1.1

- InterfaceIndex es el número de la interfaz de red a la que queremos modificar: Por ejemplo 12
- IPAddress es la dirección IP estática que piensa establecer.
- PrefixLength es la longitud del prefijo (una forma de indicar la máscara de red).
- DefaultGateway es la puerta de enlace predeterminada.

Set-DNSClientServerAddress -InterfaceIndex 12 -ServerAddresses 192.168.1.1

- *InterfaceIndex es el número de la interfaz de red que queremos modificar.
- *ServerAddresses se indican los servidores DNS separados por comas.

INSTALAMOS EL ROL ACTIVE DIRECTORY

Para instalar roles se usa el comando `Install-WindowsFeature`

Install-WindowsFeature -Name AD-Domain-Services Install-WindowsFeature RSAT-AD-Tools

CREAMOS EL DOMINIO

En este caso tenemos que crear un nuevo dominio en un nuevo bosque para lo que usaremos el comando `Install-AddForest`

Install-ADDSForest -DomainName midminiops.com -InstallDns

Si nuestro dominio no fuera un nuevo dominio en un nuevo bosque tendríamos otros comandos a la hora de promocionar nuestro servidor a controlador de dominio

COMANDO	DESCRIPCIÓN
Add- ADDSReadOnlyDomainControllerAccount	Añade un controlador de dominio de sólo lectura
Install-ADDSDomain	Crea un dominio nuevo en un arbol ya existente
Install-ADDSDomainController	Crea un controlador de dominio adicional
Install-ADDSTForest	Crea un dominio nuevo en un árbol nuevo

ANADIMOS UN EQUIPO AL DOMINIO

En el equipo que queremos añadir al dominio debemos ejecutar el siguiente comando:

Add-Computer -DomainName midminiops.com

GESTIONAR USUARIOS DE DOMINIO DESDE POWERSHELL

Se pueden gestionar las cuentas de usuario con Power Shell usando los siguientes comandos:

- **New-ADUser**: crea una cuenta de usuario.
- **Set-ADUser**: modifica una cuenta de usuario.
- **Get-ADUser**: muestra la configuración de una cuenta de usuario
- **Remove-ADUser**: elimina una cuenta de usuario.
- **Set-ADAccountPassword**: modifica una contraseña.
- **Set-ADAccountExpiration**: modifica la fecha de expiración.
- **Unlock-ADAccount**: desbloquea una cuenta de usuario.
- **Enable-ADAccount**: activa una cuenta de usuario.
- **Disable-ADAccount**: desactiva una cuenta de usuario.
- ...

Estos comandos tienen a su vez diferentes parámetros. Por ejemplo, el comando **New-ADUser** tiene muchos parámetros entre los cuales podemos encontrar los siguientes:

- **Name**: indica el nombre de la cuenta de usuario
- **AccountExpirationDate**: permite definir la fecha de expiración de una cuenta de usuario.
- **AccountPassword**: proporciona la contraseña para el usuario (si creamos un usuario sin contraseña, la cuenta estará deshabilitada).
- **ChangePasswordAtLogon**: activa la opción que obliga el cambio de contraseña tras el primer inicio de sesión.
- **Enabled**: permite definir si la cuenta está activa o no.
- **HomeDirectory**: define la ruta a la carpeta personal del usuario.
- **Path**: ruta donde se crea el usuario (unidad organizativa y dominio).
- **ProfilePath**: ruta para el perfil del usuario

Ejemplos:

- **New-ADUser -Name "Enaitz" -ChangePasswordAtLogon \$True -Enabled \$True -Path "CN=Users,DC=midominio,DC=com" -AccountPassword (ConvertTo-SecureString "P@ssw0rd" -AsPlainText -Force)**
- **New-ADUser -Name "Hegoi" -ChangePasswordAtLogon \$False -Enabled \$True -Path "OU=sucursal1,DC=midominio,DC=com" -AccountPassword (ConvertTo-SecureString "P@ssw0rd" -AsPlainText -Force)**
- **Set-ADUser -Identity "Enaitz" -ProfilePath "\\S2012\Perfiles\Enaitz"**

En Power Shell con el comando **Get-Command -Module ActiveDirectory** podemos ver todos los comandos para gestionar ActiveDirectory (entre ellos estarán los que acabamos de ver New-ADUser, Set-ADUser...)

Y para ver los atributos de cada comando podemos usar el comando **Get-Help** comando. Por ejemplo, **Get-Help New-ADUser**

GESTIONAR GRUPOS DESDE POWER SHELL

Se pueden gestionar los grupos con Power Shell usando los siguientes comandos:

- **New-ADGroup**: crea un nuevo grupo.
- **Set-ADGroup**: modifica las propiedades de un grupo.
- **Get-ADGroup**: permite ver las propiedades de un grupo.
- **Remove-ADGroup**: elimina un grupo:
- **Add-ADGroupMember**: agrega un miembro al grupo.
- **Get-ADGroupMember**: muestra los miembros de un grupo.
- **Remove-ADGroupMember**: elimina un usuario de un grupo.

Estos comandos tienen a su vez diferentes parámetros. Por ejemplo, el comando **New-ADGroup** tiene muchos parámetros entre los cuales podemos encontrar los siguientes:

- **Name**: indica el nombre del grupo.
- **GroupScope**: define el ámbito del grupo (Dominio Local, Global o Universal). Este parámetro es obligatorio.
- **GroupCategory**: especifica si el grupo es de tipo seguridad o distribución. Si no se especifica este parámetro, se crea un grupo de seguridad.
- **Path**: proporciona el contenedor que va a almacenar al objeto (unidad organizativa y dominio).

Vamos a crear el grupo Oficina dentro carpeta Users y luego vamos a añadir el usuario Enaitz a este grupo

New-ADGroup -Name Oficina -Path "CN=Users,DC=si1,DC=com" -GroupScope Universal

Add-ADGroupMember Oficina -Members "CN=Enaitz,CN=Users,DC=si1,DC=com"

En Power Shell con el comando **Get-Command -Module ActiveDirectory** podemos ver todos los comandos para gestionar ActiveDirectory (entre ellos estarán los que acabamos de ver New-ADGroup, Set-ADGroup...)

Y para ver los atributos de cada comando podemos usar el comando **Get-Help** comando. Por ejemplo, **Get-Help New-ADGroup**

GESTIONAR UNIDADES ORGANIZATIVAS DESDE POWER SHELL

Se pueden gestionar las unidades organizativas con Power Shell usando los siguientes comandos:

- **New-ADOrganizationalUnit**: realiza la creación de una unidad organizativa.
- **Set-ADOrganizationalUnit**: modifica las diferentes propiedades que tiene la unidad organizativa.
- **Get-ADOrganizationalUnit**: muestra las propiedades de la unidad organizativa.
- **Remove-ADOrganizationalUnit**: permite eliminar una unidad organizativa.

Estos comandos tienen a su vez diferentes parámetros. Por ejemplo, el comando **New-ADOrganizationalUnit** tiene muchos parámetros entre los cuales podemos encontrar los siguientes:

- **Name**: define el nombre a utilizar.
- **Path**: ruta donde se almacena el nuevo objeto.
- **ProtectedFromAccidentalDeletion**: define el estado del atributo de protección contra la eliminación.

Vamos a crear la unidad organizativa Administracion:

New-ADOrganizationalUnit -Name Administracion -Path "DC=si1,DC=com" -ProtectedFromAccidentalDeletion \$True

