**Lista de comandos CMD en Windows (MS-DOS)**

| ***Comando*** | ***Funcionalidad*** |
|---|---|
| ASSOC * | Muestra o modifica las asociaciones de las extensiones de archivos. |
| AT | Planifica comandos y programas para ejecutarse en un equipo. Está en desuso, en su lugar se recomienda utilizar **schtask.exe**. |
| ATTRIB | Muestra o cambia los atributos del archivo. |
| BREAK * | Establece o elimina la comprobación extendida de *Ctrl+C* en los sistemas *MS-DOS*. Este comando ya no está en uso. Se incluye para preservar la compatibilidad con archivos de *MS-DOS* ya existentes, pero no tiene ningún efecto en la línea de comandos porque la funcionalidad es automática. |
| CALCS | Muestra o modifica las listas de control de acceso (ACLs) de archivos. |
| CALL * | Llama a un programa por lotes desde otro. |
| **CD** * | Muestra el nombre del directorio actual o cambia a otro directorio. |
| CHCP | Muestra o establece el número de página de códigos activa. |
| **CHDIR** * | Muestra el nombre del directorio actual o cambia a otro directorio. |
| CHKDSK | Comprueba un disco y muestra un informe de su estado. |
| CHKNTFS | Muestra o modifica la comprobación de disco al iniciar (arrancar). |
| **CLS** * | Borra la pantalla. |
| CMD | Inicia una nueva instancia del intérprete de comandos de Windows. |

**Lista de comandos CMD en Windows (MS-DOS)**

| ***Comando*** | ***Funcionalidad*** |
|---|---|
| **COLOR** * | Establece los colores del primer plano y fondo predeterminados de la consola. |
| COMP | Compara el contenido de dos archivos o un conjunto de archivos. |
| COMPACT | Muestra o cambia el estado de compresión de archivos en particiones NTFS. |
| CONVERT | Convierte volúmenes FAT a volúmenes NTFS. No puede convertir la unidad actual. |
| **COPY** * | Copia uno o más archivos a otro lugar (en otra ubicación). |
| **DATE** * | Muestra o establece la fecha. |
| DEL * | Elimina uno o más archivos. |
| DIR * | Muestra una lista de archivos y subdirectorios en un directorio. |
| DISKPART | Muestra o configura las propiedades de partición de disco. |
| DISKCOMP | Compara el contenido de dos disquetes. |
| DOSKEY | Edita líneas de comando, recupera comandos de Windows y crea macros. |
| DRIVERQUERY | Muestra el estado y las propiedades actuales del controlador de dispositivo. |
| ECHO * | Muestra mensajes, o activa y desactiva el eco. |
| ENDLOCAL * | Termina la búsqueda de variables de entorno del archivo por lotes. |

**Lista de comandos CMD en Windows (MS-DOS)**

| ***Comando*** | ***Funcionalidad*** |
|---|---|
| ERASE * | Elimina uno o más archivos. |
| EXIT * | Sale del programa **cmd.exe** (intérprete de comandos). |
| FC | Compara dos archivos o conjunto de archivos y muestra las diferencias entre ellos. |
| FIND | Busca una cadena de texto en uno o más archivos. |
| FINDSTR | Busca cadenas de texto en archivos. |
| FOR * | Ejecuta un comando para cada archivo en un conjunto de archivos. |
| FORMAT | Formatea un disco para usarse con Windows. |
| FSUTIL | Muestra o configura las propiedades del sistema de archivos. |
| FTYPE * | Muestra o modifica los tipos de archivo usados en asociaciones de extensión de archivo. |
| GOTO * | Direcciona el intérprete de comandos de Windows a una línea con etiqueta. |
| GPRESULT | Muestra información de directiva de grupo por equipo o usuario. |
| GRAFTABL | Permite a Windows mostrar un conjunto de caracteres extendidos en modo gráfico. |
| **HELP** | Proporciona información de ayuda para los comandos Windows. |
| ICACLS | Muestra, modifica, hace copias de seguridad o restaura listas de control de acceso (ACL) para archivos y directorios. |

**Lista de comandos CMD en Windows (MS-DOS)**

| *Comando* | *Funcionalidad* |
|---|---|
| IF * | Ejecuta procesos condicionales en programas por lotes. |
| LABEL | Crea, cambia o elimina la etiqueta del volumen de un disco. |
| MD * | Crea un directorio. |
| MKDIR * | Crea un directorio. |
| MKLINK * | Crea vínculos simbólicos y vínculos físicos. |
| MODE | Configura un dispositivo de sistema. |
| MORE | Muestra la información pantalla por pantalla. |
| MOVE * | Mueve uno o más archivos de un directorio a otro. |
| OPENFILES | Muestra archivos compartidos abiertos por usuarios remotos como recurso compartido de archivos. |
| PATH * | Muestra o establece una ruta de búsqueda para archivos ejecutables. |
| PAUSE * | Suspende el proceso de un archivo por lotes y muestra un mensaje. |
| POPD * | Restaura el valor anterior del directorio actual guardado en PUSHD. |
| PRINT | Imprime un archivo de texto. |
| PROMPT * | Cambia el símbolo de comandos de Windows. |
| PUSHD * | Guarda el directorio actual y después lo cambia. |

**Lista de comandos CMD en Windows (MS-DOS)**

| ***Comando*** | ***Funcionalidad*** |
|---|---|
| **RD** * | Elimina un directorio. |
| RECOVER | Recupera la información legible de un disco dañado o defectuoso. |
| REM * | Registra comentarios (notas) en archivos por lotes o CONFIG.SYS. |
| **REN** * | Cambia el nombre de uno o más archivos. |
| **RENAME** * | Cambia el nombre de uno o más archivos. |
| REPLACE | Reemplaza archivos. |
| **RMDIR** * | Elimina (quita) un directorio. |
| ROBOCOPY | Utilidad avanzada para copiar archivos y árboles de directorios. |
| SC | Muestra o configura servicios (procesos en segundo plano). |
| SCHTASKS | Programa comandos y programa para ejecutarse en un equipo |
| SET * | Muestra, establece o elimina (quita) variables de entorno en Windows. |
| SETLOCAL * | Inicia la localización de los cambios de entorno en un archivo por lotes. |
| SHIFT * | Cambia la posición de parámetros reemplazables en archivos por lotes. |
| SHUTDOWN | Permite el apagado local o remoto de un equipo. |
| SORT | Ordena la salida. |

**Lista de comandos CMD en Windows (MS-DOS)**

| ***Comando*** | ***Funcionalidad*** |
|---|---|
| START * | Inicia otra ventana para ejecutar un programa o comando especificado. |
| SUBST | Asocia una ruta de acceso con una letra de unidad. |
| SYSTEMINFO | Muestra las propiedades y la configuración específicas del equipo. |
| TASKKILL | Termina o interrumpe un proceso o aplicación que se está ejecutando. |
| TASKLIST | Muestra todas las tareas en ejecución, incluidos los servicios. |
| **TIME** * | Muestra o establece la hora del sistema. |
| **TITLE** * | Establece el título de la ventana de una sesión de **cmd.exe**. |
| TREE | Muestra gráficamente la estructura de directorios de una unidad o ruta de acceso. |
| TYPE * | Muestra el contenido de uno o más archivos de texto (*.bat*, *.txt*,...). |
| **VER** * | Muestra la versión de Windows. |
| VERIFY * | Comunica a Windows si debe comprobar que los archivos se escriben de forma correcta en un disco. |
| **VOL** * | Muestra la etiqueta del volumen y el número de serie del disco. |
| WMIC | Muestra información de WMI en el shell de comandos interactivo. |
| XCOPY | Copia archivos y árboles de directorios. |

# Comandos PowerShell

En PowerShell, los comandos reciben el nombre de "cmdlet", y se puede utilizar la tecla de tabulación para que Powershell autocomplete el nombre del cmdlet que se quiere usar.

**Windows PowerShell** fue creado teniendo en cuenta su compatibilidad con versiones anteriores, por lo que es un recurso que funciona bien con los mismos comandos que utiliza el CMD. Sabiendo esto, se pueden utilizar los mismos comandos que se usaban en el Símbolo del Sistema, pero en una interfaz más avanzada y con muchos más comandos.

Aquí hemos reunido una cantidad importante de **cmdlets útiles que se pueden usar en Powershell** y detallamos la sintaxis de cada uno, así como la función específica de cada uno.

Para comenzar con lo básico, y dar una mirada rápida a los cmdlets que nos ofrece PowerShell, podemos ejecutar el comando "Show-Command", mediante el cual se va a abrir una ventana mostrándonos una lista extensa y completa de todos los comandos disponibles.

## Get-Command

En caso de que quieras conocer todos los cmdlets que ofrece PowerShell, lo podrás hacer escribiendo este comando en la consola.

**Windows PowerShell** permite, a través de este comando, conocer todas las **funciones y características que incluyen sus cmdlets**, presentados en forma de lista en la que se describen las funciones de cada uno, así como sus parámetros y opciones especiales.

Para obtener esta lista de comandos, es necesario escribir "Get-Command" seguido de un parámetro específico, con el que se obtendrá información del cmdlet en cuestión.

Por ejemplo, si escribimos en Powershell "Get-Command *-help*", vamos a ver una serie de comandos que aceptan el parámetro **"-help"**.

Si le añades, como hemos hecho en el ejemplo, un asterisco a cada lado del parámetro, obtendrás todas las posibles combinaciones que utiliza el cmdlet Get-Command cuando va acompañado de "-help".

Escribiendo en la consola “Get-Command -Name <nombre>” obtenemos un conjunto de comandos que incluyen ese nombre en específico. Puede suceder que no recuerdes o no sepas cuál es el nombre correcto de un cmdlet. Ante esta situación, puedes incluir los dos asteriscos a cada lado del nombre como mencionamos anteriormente, por ejemplo, “Get-Command -Name *set*”, con lo que podrías ver una lista de los cmdlets que incluyen el término “set” en su nombre.

## Get-Host

Con la ejecución de este comando se obtiene la versión de Windows PowerShell que está usando el sistema.

## Get-History

Con este comando se obtiene un historial de todos los comandos que se ejecutaron bajo una sesión de PowerShell y que actualmente se encuentran ejecutándose.

## Get-Random

Ejecutando este comando se obtiene un número aleatorio entre 0 y 2.147.483.646.

## Get-Service

En ciertas ocasiones, será necesario saber qué servicios se instalaron en el sistema, para lo que se puede usar el comando Get-Service, que brindará información acerca de los servicios que se están ejecutando y los que ya fueron detenidos.

Para usar este cmdlet, hay que ingresar “Get-Service” en la consola, usando al mismo tiempo alguna de las parámetros adicionales, en una sintaxis similar al siguiente ejemplo:

```
Get-Service | Where-Object {$_.Status -eq "Running"}
```

Con esto se logra que se ejecuten los servicios en el sistema. En caso de que se ejecute este comando sin ningún parámetro, se presentará una lista de todos los servicios con sus respectivos estados (“Ejecutándose o “Detenido”, por ejemplo).

Si ya se sabe con exactitud sobre qué comando se desea obtener información, usar Get-Service es bastante más práctico que dirigirse al Panel de Control de Windows y trabajar desde la GUI (interfaz gráfica de usuario) de Windows.

## Get-Help

Especialmente muy útil para usuarios novatos en el uso de Powershell, este comando presenta una ayuda básica para conocer más acerca de los cmdlets y sus funciones.

En caso de que estés usando PowerShell desde hace poco tiempo, es muy factible que te encuentres desorientado y con algunas dificultades; en dichas circunstancias, Get-Help va a convertirse en tu guía, puesto que este comando aporta la documentación imprescindible acerca de cmdlets, funciones, comandos y scripts.

De igual forma, su uso no es nada complicado: tan solo hay que escribir "Get-Help" acompañado del cmdlet del que se desean conocer más detalles. Para ejemplificar su uso, podríamos estar buscando más información del cmdlet "Get-Process", en cuyo caso sería suficiente con escribir "Get-Help Get-Process".

Para tener una idea más clara acerca del funcionamiento de Get-Help en Windows PowerShell, con solo ejecutar este comando vamos a ver una descripción junto a una breve explicación sobre cómo utilizarlo.

## Get-Date

Para saber de una forma rápida qué día fue en una determinada fecha del pasado, usando este comando se obtendrá el día exacto. Por ejemplo, para saber qué día fue el 20 de mayo de 2009, habría que escribir en Powershell:

"Get-Date 20.05.2009", ingresando la fecha en formato "dd.mm.aa". En caso de ejecutar Get-Date solo, nos brindará la fecha y hora actuales.

```
PS C:\Users\MiguePR> Get-Date
```

```
sábado, 27 de julio de 2019 12:00:40
```

## Copy-Item

Con este comando se pueden copiar carpetas o archivos.

Si estás buscando hacer una copia de archivos y directorios en tu unidad de almacenamiento, o si necesitas copiar claves o entradas del registro, Copy-Item es el cmdlet indicado. Tiene un funcionamiento muy parecido al comando “cp” que se incluye en el Símbolo del Sistema, aunque es bastante mejor.

Para esto, se debe usar el comando Copy-Item para copiar y modificar el nombre de elementos usando el mismo comando, con el que se puede establecer un nuevo nombre para dicho elemento. En el caso de que quieras copiar y renombrar el archivo “ProfesionalReview.htm” a “Proyectitosbuenos.txt”, escribe:

```
Copy-Item "C:\Proyectos.htm" -Destination "C:\MyData\Proyectos.txt".
```

## Invoke-Command

En el momento en que quieras ejecutar un script o un comando PowerShell (de forma local o remota, en uno o varios ordenadores), “Invoke-Command” va a ser tu mejor opción. Es simple de utilizar y te ayudará a gestionar ordenadores por lotes.

Es necesario tipear Invoke-Command junto al script o comando con su localización exacta.

## Invoke-Expression

Con Invoke-Expression se ejecuta otra expresión o comando. Si te encuentras ingresando una cadena de entrada o una expresión, en primer lugar este comando la va a analizar y a continuación la ejecutará. Sin este comando, la cadena no devuelve ninguna acción. Invoke-Expression solo trabaja a nivel local, a diferencia de Invoke-Command.

Para usar este comando, se debe escribir **Invoke-Expression junto con una expresión o comando**. Por ejemplo, se podría fijar una variable "$Command" con una orden que señale el cmdlet "Get-Process". Mediante la ejecución del comando "Invoke-Expression $Command", "Get-Process" va a actuar del mismo modo que un cmdlet en el equipo local.
Del mismo modo, se puede ejecutar una función en un script con el uso de una variable, lo que resulta muy útil si se trabaja con scripts dinámicos.

## Invoke-WebRequest

A través de este cmdlet, similar a cURL en Linux, se puede hacer un inicio de sesión, un scraping y la descarga de información relacionada a servicios y páginas web, mientras se trabaja desde la interfaz de PowerShell haciendo el monitoreo de algún sitio web del que se desee obtener esta información.

Para llevar a cabo estas tareas, hay que utilizarlo como Invoke-WebRequest junto a sus parámetros. Con esto, es posible conseguir los enlaces que tiene un sitio web específico con la siguiente sintaxis de ejemplo:

```
(Invoke-WebRequest –Uri 'https://wwww.ebay.com').Links
```

En este caso, se obtendrían los enlaces del sitio eBay.

## Set-ExecutionPolicy

Si bien podemos crear e iniciar scripts (.ps1) desde PowerShell, vamos a encontrarnos limitados debido a cuestiones de seguridad. Sin embargo, esto puede ser modificado a través de la categoría de seguridad empleando el cmdlet Set-ExecutionPolicy.

Solo es necesario tipear Set-ExecutionPolicy junto a una de las cuatro opciones de seguridad para hacer los cambios que se requieren:

- Restricted
- All Signed

- Remote Signed
- Unrestricted

Por ejemplo, si queremos establecer el nivel de seguridad restringida, tendríamos que usar:

```
Set-ExecutionPolicy -ExecutionPolicy Restricted
```

## Get-Item

En caso de que estés buscando información acerca de un elemento con una ubicación concreta, como podría ser un directorio en el disco duro, el comando Get-Item resulta el indicado para esta tarea.

Hay que aclarar que no se obtiene el contenido mismo del elemento, tal como subdirectorios y archivos en una carpeta específica, a no ser que lo solicites de manera explícita.

Del otro lado a Get-Item nos encontramos con el cmdlet Remove-Item, que permite eliminar el elemento especificado.

## Remove-Item

En caso de que desees borrar elementos como carpetas, archivos, funciones y variables y claves del registro, Remove-Item será el mejor cmdlet. Lo importante es que ofrece parámetros para introducir y expulsar elementos.

Con el cmdlet Remove-Item puedes remover elementos de localizaciones específicas con el uso de ciertos parámetros. A modo de ejemplo, es posible remover el archivo “Finanzas.txt” empleando el comando siguiente:

```
Remove-Item "C:\MyData\Finanzas.txt"
```

## Get-Content

Cuando necesites todo lo que incluye en cuanto a contenido un archivo de texto en una ruta concreta, ábrelo y léelo utilizando un editor de textos como el Bloc de Notas. Mediante Windows PowerShell se puede utilizar el comando Get-Content para examinar lo que contiene un archivo sin necesidad de abrirlo.

Por ejemplo, es posible obtener 20 líneas de texto incluidas en el archivo "Proyectos.htm", para lo que puedes escribir:

```
Get-Content "C:\Proyectos.htm" -TotalCount 20
```

Este cmdlet es similar al cmdlet Get-Item anterior, pero con el cual podemos obtener lo que incluye el archivo que has indicado. Si ejecutas este comando para un archivo de extensión txt, te revelará íntegramente el texto que incluye dicho archivo. Si lo utilizas en un archivo de imagen png, vas a obtener gran cantidad de datos binarios ilegibles y sin sentido.

Si se utiliza solo, Get-Content no ofrece mucha utilidad. Pero se puede mezclar con cmdlets más específicos con el objetivo de obtener resultados más precisos.

## Set-Content

Con este cmdlet es posible almacenar texto en un archivo, algo parecido a lo que se puede hacer con "echo" en el Bash. Si se usa en combinación con el cmdlet Get-Content, se puede ver primero qué es lo que contiene un determinado archivo para posteriormente hacer la copia a otro archivo a través de Set-Content.

Por ejemplo, se puede usar el cmdlet Set-Content para añadir o sustituir lo que contiene un archivo por otro contenido. Por último, se puede combinar con el comando antes mencionado para guardarlo con un nuevo nombre (ejemplo.txt) de la siguiente manera:

```
Get-Content "C:\Proyectos.htm" -TotalCount 30 | Set-Content "Ejemplo.txt"
```

## Get-Variable

Si estás en PowerShell tratando de utilizar variables, esto podrá ser hecho con el cmdlet Get-Variable, con el que vas a poder visualizar dichos valores. Este comando muestra los valores en una tabla, desde donde se pueden utilizar, incluir y excluir comodines.

Para utilizarlo solo debes escribir “Get-Variable” acompañado de sus parámetros y demás opciones. Por ejemplo, si te gustaría conocer el valor de la variable “descuento” escribe lo siguiente:

```
Get-Variable -Name "descuento"
```

## Set-Variable

El valor de una variable puede ser establecido, modificado o reinicializado con este cmdlet. Para fijar el valor de la variable del caso anterior, habría que escribir lo siguiente:

```
Set-Variable -Name "descuento" -Value "Aquí se fija el valor"
```

## Get-Process

A menudo, utilizamos el Administrador de Tareas con el fin de descubrir exactamente qué procesos se están ejecutando en nuestro PC. En PowerShell, cualquier usuario puede saber esto ejecutando este cmdlet, con el que obtendrá la lista de procesos activos en ese momento.

El cmdlet Get-Process tiene cierta semejanza con Get-Service, aunque en este caso ofrece información acerca de los procesos.

## Start-Process

Con este cmdlet, Windows PowerShell hace que sea mucho más fácil ejecutar procesos en el equipo.

Por ejemplo, si necesitas usar la calculadora, la podrás abrir de forma rápida y sencilla tipeando lo siguiente:

Start-Process -FilePath “calc” –Verb

## Stop-Process

Con este cmdlet puedes detener un proceso, ya sea que haya sido iniciado por ti o por otro usuario.

Siguiendo con el ejemplo de la Calculadora, si deseas interrumpir íntegramente sus procesos en ejecución, escribe lo indicado abajo en PowerShell:

Stop-Process -Name “calc”

## Start-Service

Si necesitas comenzar un servicio en el PC, el cmdlet Start-Service es el indicado en este caso, sirviendo de igual modo aunque dicho servicio esté deshabilitado en el PC.