

# Los comandos básicos

Ante todo, PowerShell no es más que un entorno en línea de comandos al servicio del sistema operativo pero también, y principalmente, al servicio de los usuarios. Por tanto, se presenta con un conjunto de comandos que es muy conveniente conocer. O al menos saber como encontrarlos o recuperarlos...

## 1. Constitución de los comandos


Los comandos de PowerShell se denominan «cmdlets» (por command-applets). Por nuestra parte, al no existir una traducción oficial, nos hemos tomado la licencia de llamarlos «**commandlets**». La mayoría están constituidos de la siguiente forma: un verbo y un nombre separados por un guión (-): **verbo-nombre**. Por ejemplo, **Get-Command**.

El verbo (en inglés por supuesto) describe la acción a aplicar sobre el nombre. En este ejemplo vamos a recuperar (**get**) comandos (**command**).

Con PowerShell encontraremos una serie de verbos genéricos: **Get**, **Set**, **Add**, **Remove**, etc. que se combinan con diferentes nombres como **Path**, **Variable**, **Item**, **Object**, etc.

Los nombres que constituyen los comandos siempre se escriben en singular; y esto es igualmente válido para sus parámetros.

Podemos entonces entrecruzar los verbos con los nombres, obteniendo de este modo un buen número de comandos. Observe que los comandos, así como sus parámetros asociados, pueden escribirse tanto en mayúsculas como en minúsculas. El analizador de sintaxis PowerShell no es sensible a mayúsculas y minúsculas.

 Encontrará la lista completa de commandlets en el anexo Lista de comandos.

## 2. Get-Command

Si usted debiera memorizar un único comando, entonces este sería el indicado: **Get-Command**.

PS > Get-Command -CommandType cmdlet		
CommandType	Name	Definition
-----	----	-----
Cmdlet	Add-Content	Add-Content [-Path] <String[...
Cmdlet	Add-History	Add-History [[-InputObject] ...
Cmdlet	Add-Member	Add-Member [-MemberType] <PS...
Cmdlet	Add-PSSnapin	Add-PSSnapin [-Name] <String...
Cmdlet	Clear-Content	Clear-Content [-Path] <Strin...
Cmdlet	Clear-Item	Clear-Item [-Path] <String[]...
Cmdlet	Clear-ItemProperty	Clear-ItemProperty [-Path] <...
Cmdlet	Clear-Variable	Clear-Variable [-Name] <Stri...
Cmdlet	Compare-Object	Compare-Object [-ReferenceOb...
Cmdlet	ConvertFrom-SecureString	ConvertFrom-SecureString [-S...
Cmdlet	Convert-Path	Convert-Path [-Path] <String...
Cmdlet	ConvertTo-Html	ConvertTo-Html [[-Property] ...
Cmdlet	ConvertTo-SecureString	ConvertTo-SecureString [-Str...
Cmdlet	Copy-Item	Copy-Item [-Path] <String[]>...
Cmdlet	Copy-ItemProperty	Copy-ItemProperty[-Path] <S...
Cmdlet	Export-Alias	Export-Alias [-Path] <String...
...		
...		
...		

Cmdlet	Where-Object	Where-Object [-FilterScript]...
Cmdlet	Write-Debug	Write-Debug [-Message] Stri...
Cmdlet	Write-Error	Write-Error [-Message] <Stri...
Cmdlet	Write-Host	Write-Host [[-Object] <Objec...
Cmdlet	Write-Output	Write-Output [-InputObject] ...
Cmdlet	Write-Progress	Write-Progress [-Activity] <...
Cmdlet	Write-Verbose	Write-Verbose [-Message] <St...
Cmdlet	Write-Warning	Write-Warning [-Message] <St...

**Get-Command** le permite conocer todos los comandos integrados en PowerShell. En la primera versión de PowerShell, los comandos básicos eran 129. Actualmente, en la versión 2, han aumentado hasta los 236. Para verificarlo, puede teclear:

```
PS > Get-Command -CommandType cmdlet | Measure-Object
```

```
Count      : 236
Average    :
Sum        :
Maximum    :
Minimum    :
Property   :
```

Si su resultado difiere, es que probablemente deba instalar los commandlets adicionales, ya sea a través de *Snap-ins*, de *funciones avanzadas*, de *módulos* (volveremos a ellos más tarde en el libro) o bien, añadiendo funciones o funcionalidades si se encuentra en una plataforma Windows Server.

Para conocer más acerca de **Get-Command**, teclee el comando:

```
PS > Get-Help Get-Command -Detailed | more
```

o

```
PS > Help Get-Command
```

### Por ejemplo:

```
PS > Help Get-Command
```

#### NOMBRE

Get-Command

#### SINOPSIS

Obtiene información básica acerca de los cmdlets y otros elementos de comandos de Windows PowerShell.

#### SINTAXIS

```
Get-Command [[-Name] <string[]>] [-CommandType {Alias | Function |
Filter | Cmdlet | ExternalScript | Application | Script | All}]
[[-ArgumentList] <Object[]>] [-Module <string[]>] [-Syntax] [-TotalCount
<int>] [<CommonParameters>]

Get-Command [-Noun <string[]>] [-Verb <string[]>] [[-ArgumentList]
<Object[]>] [-Module <string[]>] [-Syntax] [-TotalCount <int>]
[<CommonParameters>]
```

#### DESCRIPCIÓN

El cmdlet `Get-Command` obtiene información básica sobre cmdlets y otros elementos de comandos de Windows PowerShell en la sesión, como alias, funciones, filtros, scripts y aplicaciones. `Get-Command` obtiene sus datos directamente del código de un cmdlet, función, script o alias, a diferencia de `Get-Help` que obtiene su información de los archivos de temas de Ayuda.

Sin parámetros, "`Get-Command`" obtiene todos los cmdlets y funciones de la sesión actual. "`Get-Command *`" obtiene todos los elementos de Windows PowerShell y todos los archivos que no son de Windows PowerShell en la variable de entorno `Path` (`$env:path`). Agrupa los archivos en el tipo de comando "`Application`".

Puede utilizar el parámetro `Module` de `Get-Command` para buscar los comandos que se agregaron a la sesión agregando un complemento de Windows PowerShell o importando un módulo.

#### VÍNCULOS RELACIONADOS

Online version: <http://go.microsoft.com/fwlink/?LinkID=113309>  
`about_Command_Precedence`  
`Get-Help`  
`Get-PSDrive`  
`Get-Member`  
`Import-PSSession`  
`Export-PSSession`

#### NOTAS

Para ver los ejemplos, escriba: "`get-help Get-Command -examples`".  
Para obtener más información, escriba: "`get-help Get-Command -detailed`".  
Para obtener información técnica, escriba:  
"`get-help Get-Command -full`".

Esta línea de comando nos permite obtener una ayuda detallada sobre la utilización de `Get-Command`. Podemos observar por ejemplo que podemos utilizar el parámetro `-verb`. Veamos que obtendríamos con ello:

```
PS > Get-Command -Verb write
```

CommandType	Name	Definition
-----	----	-----
Cmdlet	Write-Debug	Write-Debug [-Message] ...
Cmdlet	Write-Error	Write-Error [-Message] ...
Cmdlet	Write-EventLog	Write-EventLog [-LogNam...
Cmdlet	Write-Host	Write-Host [[-Object] <...
Cmdlet	Write-Output	Write-Output [-InputObj...
Cmdlet	Write-Progress	Write-Progress [-Activi...
Cmdlet	Write-Verbose	Write-Verbose [-Message...
Cmdlet	Write-Warning	Write-Warning [-Message...

Obtenemos una lista de todos los comandos donde el verbo comienza por `write`. Veremos en la próxima parte cómo están estructurados los comandos PowerShell.

Del mismo modo, podríamos obtener la lista de comandos que se aplican a los objetos:

```
PS > Get-Command -Noun object
```

CommandType	Name	Definition
-------------	------	------------

-----	----	-----
Cmdlet	Compare-Object	Compare-Object [-Refere...
Cmdlet	ForEach-Object	ForEach-Object [-Proces...
Cmdlet	Group-Object	Group-Object [[-Propert...
Cmdlet	Measure-Object	Measure-Object [[-Prope...
Cmdlet	New-Object	New-Object [-TypeName] ...
Cmdlet	Select-Object	Select-Object [[-Proper...
Cmdlet	Sort-Object	Sort-Object [[-Property...
Cmdlet	Tee-Object	Tee-Object [-FilePath] ...
Cmdlet	Where-Object	Where-Object [-FilterSc...

Podemos de igual forma obtener los comandos de un cierto tipo, los más utilizados son: **Alias**, **Function**, **cmdlet**, **externalscript**, **application**.

Ejemplo:

```
PS > Get-Command -Commandtype alias
```

CommandType	Name	Definition
-----	----	-----
Alias	%	ForEach-Object
Alias	?	Where-Object
Alias	ac	Add-Content
Alias	asnp	Add-PSSnapin
Alias	cat	Get-Content
Alias	cd	Set-Location
Alias	chdir	Set-Location
Alias	clc	Clear-Content
Alias	clear	Clear-Host
Alias	cli	Clear-Item
Alias	clp	Clear-ItemProperty
Alias	cls	Clear-Host
Alias	clv	Clear-Variable
Alias	copy	Copy-Item
Alias	cp	Copy-Item
Alias	cpi	Copy-Item
Alias	cpp	Copy-ItemProperty
Alias	cvpa	Convert-Path
...		
...		
...		
Alias	spsv	Stop-Service
Alias	sv	Set-Variable
Alias	tee	Tee-Object
Alias	type	Get-Content
Alias	where	Where-Object
Alias	write	Write-Output

Si está intentando encontrar un comando del cual ignora el nombre, pero sabe que el comando que está buscando debe facilitarnos una determinada información, hay muchas probabilidades que este empiece por **Get**. En esta situación, podrá hacer lo siguiente: `Get-Command Get*` o `Get-Command Get-*`.

```
PS > Get-Command Get-*
```

CommandType	Name	Definition
-----	----	-----

Cmdlet	Get-Acl	Get-Acl [[-Path] <Strin...
Cmdlet	Get-Alias	Get-Alias [[-Name] <Str...
Cmdlet	Get-AuthenticodeSignature	Get-AuthenticodeSignatu...
Cmdlet	Get-ChildItem	Get-ChildItem [[-Path] ...
Cmdlet	Get-Command	Get-Command [[-Argument...
Cmdlet	Get-ComputerRestorePoint	Get-ComputerRestorePoin...
Cmdlet	Get-Content	Get-Content [-Path] <St...
Cmdlet	Get-Counter	Get-Counter [[-Counter]...
Cmdlet	Get-Credential	Get-Credential [-Creden...
Cmdlet	Get-Culture	Get-Culture [-Verbose] ...
Cmdlet	Get-Date	Get-Date [[-Date] <Date...
Cmdlet	Get-Event	Get-Event [[-SourceIden...
Cmdlet	Get-EventLog	Get-EventLog [-LogName]...
Cmdlet	Get-EventSubscriber	Get-EventSubscriber [[-...
Cmdlet	Get-ExecutionPolicy	Get-ExecutionPolicy [[-...

Del mismo modo, si sabe que el comando que está buscando se aplica a « items », podrá intentar esto:

```
PS > Get-Command *-Item
```

CommandType	Name	Definition
-----	----	-----
Cmdlet	Clear-Item	Clear-Item [-Path] <Str...
Cmdlet	Copy-Item	Copy-Item [-Path] <Stri...
Cmdlet	Get-Item	Get-Item [-Path] <Strin...
Cmdlet	Invoke-Item	Invoke-Item [-Path] <St...
Cmdlet	Move-Item	Move-Item [-Path] <Stri...
Cmdlet	New-Item	New-Item [-Path] <Strin...
Cmdlet	Remove-Item	Remove-Item [-Path] <St...
Cmdlet	Rename-Item	Rename-Item [-Path] <St...
Cmdlet	Set-Item	Set-Item [-Path] <Strin...

Como hemos introducido el comando `Get-Help` en uno de los ejemplos anteriores, lo detallaremos a continuación.

### 3. Get-Help

Este comando nos va a permitir, como su nombre indica, obtener la ayuda de cualquier commandlet, io incluso más!

Para acceder a la ayuda de un determinado comando, existen diferentes formas:

- `Get-Help miComando`
- `Help miComando`
- `miComando -?`

`Get-Help miComando` le mostrará la ayuda estándar.

Con PowerShell, tendrá tres niveles de ayuda:

- la ayuda estándar,
- la ayuda detallada,
- la ayuda completa.

Para acceder a la ayuda detallada, añada el parámetro **-Detailed**, o bien `Get-Help miComando -detailed`. Y para la ayuda completa, especifique el parámetro **-Full**, `Get-Help miComando -full`.

Cuando introduzca `miComando -?`, no podrá especificar el nivel de detalle, la ayuda devuelta es entonces la ayuda estándar.



Le recomendamos preferiblemente la utilización del comando `Help miComando`, seguido del nivel de detalle deseado (**-detailed** o **-full**) esto le ofrecerá dos ventajas interesantes: la primera, que esta opción es más corta de teclear, y la segunda, la ayuda se visualizará página por página. `Help` es una función que permite mostrar el contenido de la ayuda página por página. Fundamentalmente, esta función se contenta con llamar `Get-Help` y pasarle su contenido a `more`.



Si teclea simplemente el comando `Help`, tendrá acceso a todas opciones de ayuda que PowerShell puede proponerle. Pruébelo, se sorprenderá.

La ayuda de PowerShell es especialmente rica debido a que permite igualmente el acceso a la ayuda sobre la utilización de tablas, de operadores de comparación, de bucles, de tuberías, de funciones, etc.

Para descubrir todas las opciones posibles teclee: `help about_*`

Esta ayuda será muy valiosa cuando esté desarrollando un script y haya olvidado de llevar consigo la maravillosa obra que tiene ahora en la mano... 😊

```
PS > Help about_*
```

Name	Category	Synopsis
----	-----	-----
about_aliases	HelpFile	Describe cómo usar nombres al...
about_Arithmetic_Operators	HelpFile	Describe los operadores que r...
about_arrays	HelpFile	Estructura de datos compacta ...
about_Assignment_Operators	HelpFile	Describe cómo utilizar operad...
about_Automatic_Variables	HelpFile	Describe las variables que al...
about_Break	HelpFile	Describe una instrucción que ...
about_command_precedence	HelpFile	Describe cómo determina Windo...
about_Command_Syntax	HelpFile	Describe la notación utilizad...
about_Comment-Based_Help	HelpFile	Describe cómo escribir temas ...
about_CommonParameters	HelpFile	Describe los parámetros que s...
...		
...		
...		
about_Special_Characters	HelpFile	Describe los caracteres espec...
about_split	HelpFile	Explica cómo usar el operador...
about_Switch	HelpFile	Explica cómo se utiliza una i...
about_Throw	HelpFile	Describe la palabra clave Thr...
about_transactions	HelpFile	Describe cómo se administran ...
about_trap	HelpFile	Describe una palabra clave qu...
about_try_catch_finally	HelpFile	Describe cómo se usan los blo...
about_types.ps1xml	HelpFile	Explica cómo los archivos Typ...
about_type_operators	HelpFile	Describe los operadores que f...
about_Variables	HelpFile	Describe cómo las variables a...
about_While	HelpFile	Describe una instrucción de l...
about_wildcards	HelpFile	Describe cómo se utilizan los...
about_Windows_PowerShell_2.0	HelpFile	Describe las nuevas caracterí...
about_Windows_PowerShell_ISE	HelpFile	Describe las características ...
about_WMI_Cmdlets	HelpFile	Proporciona información adici...

Básicamente, existe cerca de un centenar de opciones de ayuda. Podrá profundizar ampliamente sus conocimientos en numerosos temas. Le animamos a su lectura ya que es de excelente calidad y además en castellano.

Que mejor que un ejemplo para ver cómo se presenta la ayuda:

```
PS > Help Get-Item
NOMBRE
    Get-Item

SINOPSIS
    Obtiene el elemento de la ubicación especificada.

SINTAXIS
    Get-Item [-LiteralPath] <string[]> [-Credential <PSCredential>]
[-Exclude <string[]>] [-Filter <string>] [-Force] [-Include <string[]>]
[-UseTransaction] [<CommonParameters>]

    Get-Item [-Path] <string[]> [-Credential <PSCredential>] [-Exclude
<string[]>] [-Filter <string>] [-Force] [-Include <string[]>]
[-UseTransaction] [<CommonParameters>]

DESCRIPCIÓN
    El cmdlet Get-Item obtiene el elemento de la ubicación especificada. No
    obtiene el contenido del elemento de la ubicación especificada, a menos
    que se use un carácter comodín (*) para solicitar todo el contenido del
    elemento.

    Los proveedores de Windows PowerShell usan el cmdlet Get-Item para permi
    tir navegar por distintos tipos de almacenes de datos.

VÍNCULOS RELACIONADOS
    Online version: http://go.microsoft.com/fwlink/?LinkID=113319
    about_Providers
    Clear-Item
    Copy-Item
    Invoke-Item
    Move-Item
    Set-Item
    New-Item
    Remove-Item
    Rename-Item

NOTAS
    Para ver los ejemplos, escriba: "get-help Get-Item -examples".
    Para obtener más información, escriba: "get-help Get-Item -detailed".
    Para obtener información técnica, escriba: "get-help Get-Item -full".
```

Una novedad introducida por PowerShell v2 es el enlace hacia la versión « Online » de la ayuda. Un copiar/pegar de la URL situada en la rúbrica de enlaces conexos en su navegador le permitirá beneficiarse de la última versión de la ayuda del comando buscado. Dicho esto, la ayuda en castellano no está siempre disponible en línea.

## 4. Get-Member

Éste es probablemente el comando más interesante de todos ya que permite listar todas las propiedades y métodos de un objeto así como su tipo. Tenga en cuenta que no es necesario conocer este comando cuando empiece a dar sus primeros pasos con PowerShell. En efecto, este comando introduce el concepto de objetos al que nos referiremos un poco más adelante. Puede volver a este comando posteriormente, una vez haya adquirido una buena base.

Gracias a **Get-Member** podrá sorprender a sus compañeros de trabajo debido a que ganará un tiempo considerable al escribir sus scripts.

```
PS > $miVariable = '¡Buenos días a todos!'
```

Acabamos de crear la variable `$miVariable` y le hemos asociado un valor de tipo cadena (**String**). Observará que no hemos tenido la necesidad de declararla ya que PowerShell reconoce automáticamente su tipo en función de su contenido. Una variable empieza siempre con el carácter dólar. Examinaremos en detalle las variables en el capítulo siguiente.

Mientras tanto, supongamos que queremos realizar las acciones siguientes, como por ejemplo, convertirla en mayúsculas o bien contar el número de caracteres de contiene.

Para ello normalmente en todo lenguaje de scripts o de programación debemos hacer referencia a la documentación para conocer los comandos que permiten la manipulación de las cadenas de caracteres. Por supuesto en PowerShell podemos hacer lo mismo, pero es en este momento cuando el comando `Get-Member` toma sentido y ahora entenderá porqué...

■ Teclee:

```
PS > $miVariable | Get-Member
```

```
TypeName: System.String
```

Name	MemberType	Definition
----	-----	-----
Clone	Method	System.Object Clone()
CompareTo	Method	int CompareTo(System.Object valu...
Contains	Method	bool Contains(string value)
CopyTo	Method	System.Void CopyTo(int sourceInd...
EndsWith	Method	bool EndsWith(string value), boo...
Equals	Method	bool Equals(System.Object obj), ...
GetEnumerator	Method	System.CharEnumerator GetEnumera...
GetHashCode	Method	int GetHashCode()
GetType	Method	type GetType()
GetTypeCode	Method	System.TypeCode GetTypeCode()
IndexOf	Method	int IndexOf(char value), int Ind...
IndexOfAny	Method	int IndexOfAny(char[] anyOf), in...
Insert	Method	string Insert(int startIndex, st...
IsNormalized	Method	bool IsNormalized(), bool IsNorm...
LastIndexOf	Method	int LastIndexOf(char value), int...
LastIndexOfAny	Method	int LastIndexOfAny(char[] anyOf)...
Normalize	Method	string Normalize(), string Norma...
PadLeft	Method	string PadLeft(int totalWidth), ...
PadRight	Method	string PadRight(int totalWidth),...
Remove	Method	string Remove(int startIndex, in...
Replace	Method	string Replace(char oldChar, cha...
Split	Method	string[] Split(Params char[] sep...
StartsWith	Method	bool StartsWith(string value), b...
Substring	Method	string Substring(int startIndex)...
ToCharArray	Method	char[] ToCharArray(), char[] ToC...



ToLower	Method	string ToLower(), string ToLower...
ToLowerInvariant	Method	string ToLowerInvariant()
ToString	Method	string ToString(), string ToStri...
ToUpper	Method	string ToUpper(), string ToUpper...
ToUpperInvariant	Method	string ToUpperInvariant()
Trim	Method	string Trim(Params char[] trimCh...
TrimEnd	Method	string TrimEnd(Params char[] tri...
TrimStart	Method	string TrimStart(Params char[] t...
Chars	ParameterizedProperty	char Chars(int index) {get;}
Length	Property	System.Int32 Length {get;}

Vemos aparecer múltiples elementos particularmente interesantes:

- El campo `TypeName` nos indica el tipo de nuestra variable. Siendo como se suponía un tipo `String`.
- Una lista de nombres de métodos, de propiedades y sus definiciones asociadas.

Sin gran esfuerzo podemos imaginar que el método `ToUpper` nos permitirá pasar a mayúsculas la cadena contenida en `$miVariable`.

```
PS > $miVariable.ToUpper()  
  
¡BUENOS DIAS A TODOS!
```

Del mismo modo, podemos decir que la propiedad `Length` nos dará el número de caracteres contenidos en nuestra cadena.

```
PS > $miVariable.Length  
  
21
```

En resumen, este commandlet es verdaderamente indispensable una vez se ha probado...



Un error clásico al iniciarse con PowerShell es olvidar los paréntesis del final cuando se hace una llamada a un método. Por ejemplo, si teclea `$miVariable.ToUpper`, no obtendrá el resultado esperado ya que PowerShell mostrará la definición del método. Preste entonces atención a este punto.



PowerShell v2 aporta al comando `Get-Member` el conmutador **-Force**. Este permite mostrar las propiedades y métodos avanzados de los objetos. No es necesario preocuparse por el momento, volveremos a hablar en el capítulo Control del Shell.