

Introducción a PowerShell

Qué es?

- Un entorno interactivo orientado a objetos que usa programas llamados cmdlets para tareas de configuración y administración.

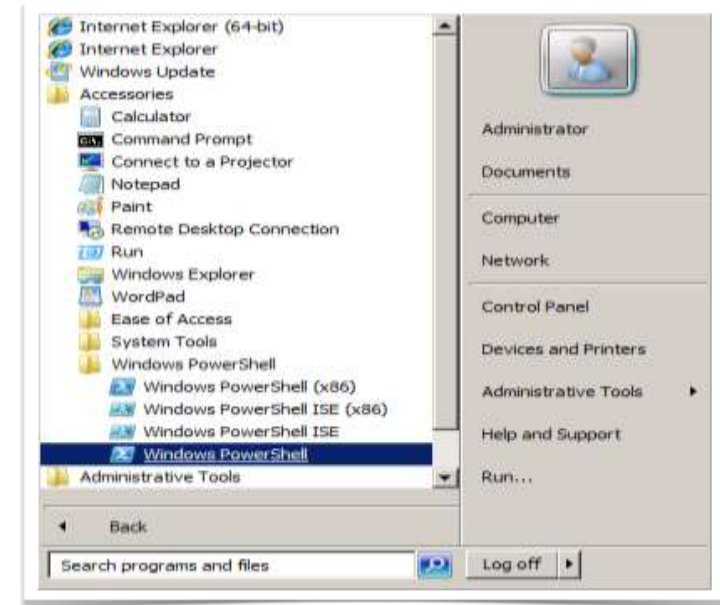
Qué me permite hacer?

- Mejora de la gestión y automatización
- Gestión en tiempo real
- Gestión a gran escala

- Protegido por defecto
- Evita errores de administradores y usuarios no intencionados
- Sin ejecución de script
- .ps1 extensión asociada a notepad
- Debe escribir la ruta para ejecutar un script

Versiones

- Instalar Windows PowerShell 5.0
- • Windows 10 or Windows Server 2016
- • Parte de Windows Management Framework (WMF) 5.0 incluido en Windows



INTRODUCCIÓN – CONCEPTOS BÁSICOS

- Familiarizándonos con la consola
- Cmdlets : Verbo – Sustantivo
- ¡Los comandos nativos funcionan!
- – - Ping, IPConfig, calc, notepad, mspaint
- cls - Clear-Host
- cd - Set-Location
- dir, ls - Get-Childitem
- type, cat - Get-Content
- Copy, cp - Copy-item

Ayuda

- Get-Help, help y man
- `? Help <cmdlet>`
- `? Help *parcial*`
- `? Help <cmdlet> -Full`
- `? Help <cmdlet> -Online`
- `? Help <cmdlet> -ShowWindow`
- `? Help <cmdlet> -Examples`
- `? Get-Help About_*`

- Get-Help Get-Help -Full

```
-Full  
  
Required?                false  
Position?                Named  
Accept pipeline input?   false  
Parameter set name       AllUsersView  
Aliases                  None  
Dynamic?                 false  
Accept wildcard characters? false
```

- Esta descripción te informa sobre la utilización del parámetro. Así, te indica que no es un parámetro requerido, que es independiente de la posición, que no se puede utilizar encadenado a otros comandos, es decir, no se puede utilizar con *pipes* o tuberías.

GET-COMMAND

Get-Member (gm)

❓ TypeName es un nombre único asignado por Windows

❓ Muestra las propiedades y métodos de un objeto

❓ Las propiedades son columnas de información del objeto

❓ Los métodos son las acciones que puede realizar el objeto

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

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Javier> get-command *
```

CommandType	Name	Version	Source
-----	----	-----	-----
Alias	% -> ForEach-Object		
Alias	? -> Where-Object		
Alias	ac -> Add-Content		
Alias	Add-AppPackage	2.0.1.0	Appx
Alias	Add-AppPackageVolume	2.0.1.0	Appx
Alias	Add-AppProvisionedPackage	3.0	Dism
Alias	Add-ProvisionedAppPackage	3.0	Dism
Alias	Add-ProvisionedAppxPackage	3.0	Dism
Alias	Add-ProvisioningPackage	3.0	Provisioning
Alias	Add-TrustedProvisioningCertificate	3.0	Provisioning
Alias	algm ->	1.0.0.0	Microsoft.PowerShell.LocalAccounts
Alias	Apply-WindowsUnattend	3.0	Dism
Alias	asnp -> Add-PSSnapin		
Alias	blsmba ->	2.0.0.0	SmbShare
Alias	cat -> Get-Content		
Alias	cd -> Set-Location		
Alias	CFS -> ConvertFrom-String	3.1.0.0	Microsoft.PowerShell.Utility
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	cmpcfg ->	1.1	PSDesiredStateConfiguration
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	cssmbo ->	2.0.0.0	SmbShare

- ¿PUEDO ENCADENAR CUALQUIER COMANDO?
- No. No todos los comandos se pueden encadenar. Y, ¿como puedo saber si se puede encadenar un comando?.
- Para esto tienes que utilizar la ayuda. En concreto tienes que utilizar la opción -Full y fijarte en las secciones INPUTS y OUTPUTS donde te indica si admite la posibilidad de encadenar y que tipos de datos admite.

EL SISTEMA DE TUBERÍAS

El carácter para las tuberías es el AltGr+1

|

Conecta cmdlets para conseguir resultados más útiles



```
PS C:\> Get-Service | select-object name, status | sort-object name
```

Puede ser dividido en varias líneas para facilitar su lectura

```
PS C:\> Get-Service |  
>> select-object name, status |  
>> sort-object name
```

```
PS C:\Users\Javier> ls | format-list
```

```
Directorio: C:\Users\Javier

Name           : .dbus-keyrings
CreationTime    : 21/11/2022 23:09:32
LastWriteTime   : 08/01/2023 0:24:04
LastAccessTime  : 08/01/2023 0:24:04
Mode            : d-----
LinkType        : 
Target          : {}
```

```
Name           : .ms-ad
CreationTime    : 08/09/2022 11:42:16
LastWriteTime   : 08/09/2022 11:42:16
LastAccessTime  : 22/12/2022 21:55:34
Mode            : d-----
LinkType        : 
Target          : {}
```

```
PS C:\Users\Javier> ls | format-table
```

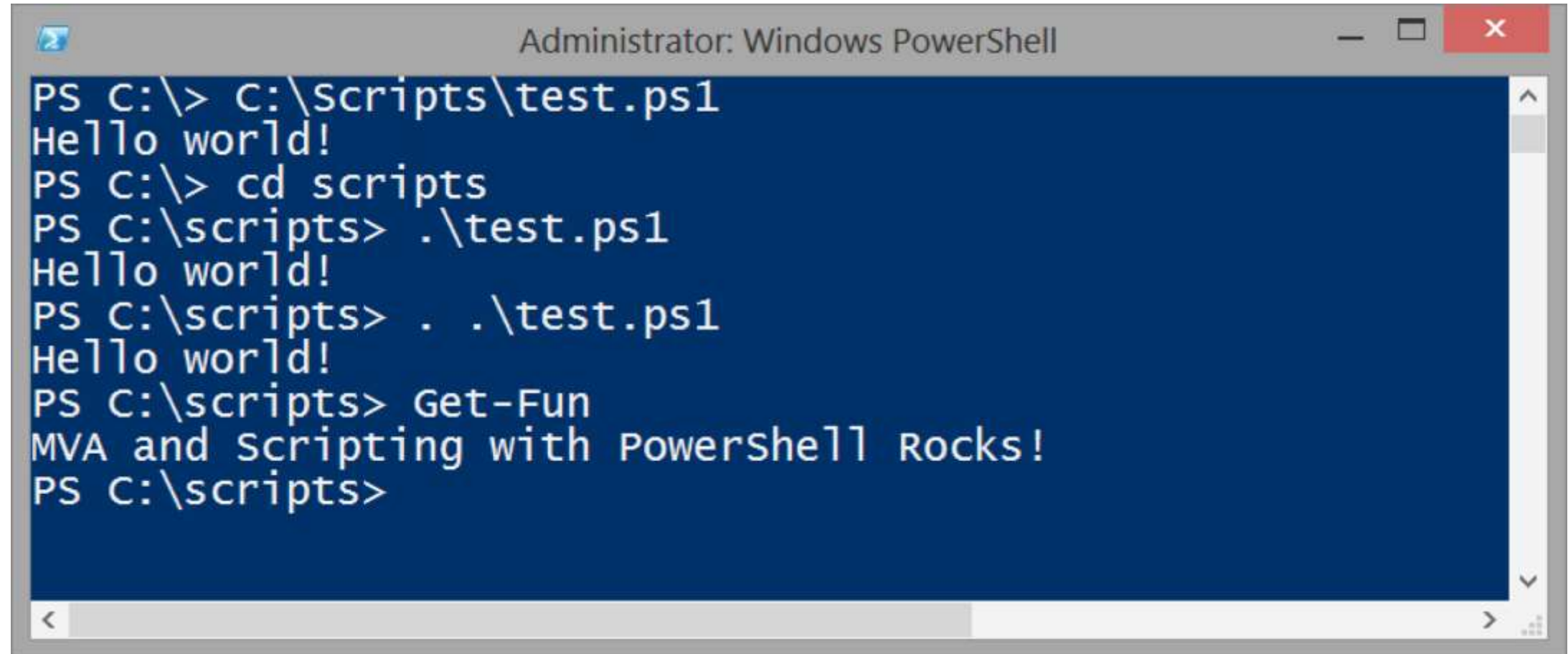
Directorio: C:\Users\Javier

Mode		LastWriteTime	Length	Name
----		-----	-----	----
d-----	08/01/2023	0:24		.dbus-keyrings
d-----	08/09/2022	11:42		.ms-ad
d-----	05/11/2022	11:28		.VirtualBox
d-----	23/10/2022	18:18		.vscode
d-r---	08/09/2022	11:37		3D Objects
d-----	11/12/2022	14:04		Calibre Portable
d-r---	08/09/2022	11:37		Contacts
d-r---	08/01/2023	0:23		Desktop
d-r---	05/01/2023	10:20		Documents
d-r---	08/01/2023	17:41		Downloads
d-r---	08/09/2022	11:37		Favorites
d-----	07/01/2023	23:55		Intel
d-r---	08/09/2022	11:37		Links
d-r---	21/09/2022	23:03		Music

Errores

```
PS C:\Users\Javier> ls format-list
ls : No se encuentra la ruta de acceso 'C:\Users\Javier\format-list' porque no existe.
En línea: 1 Carácter: 1
+ ls format-list
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\Users\Javier\format-list:String) [Get-ChildItem]
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand
```

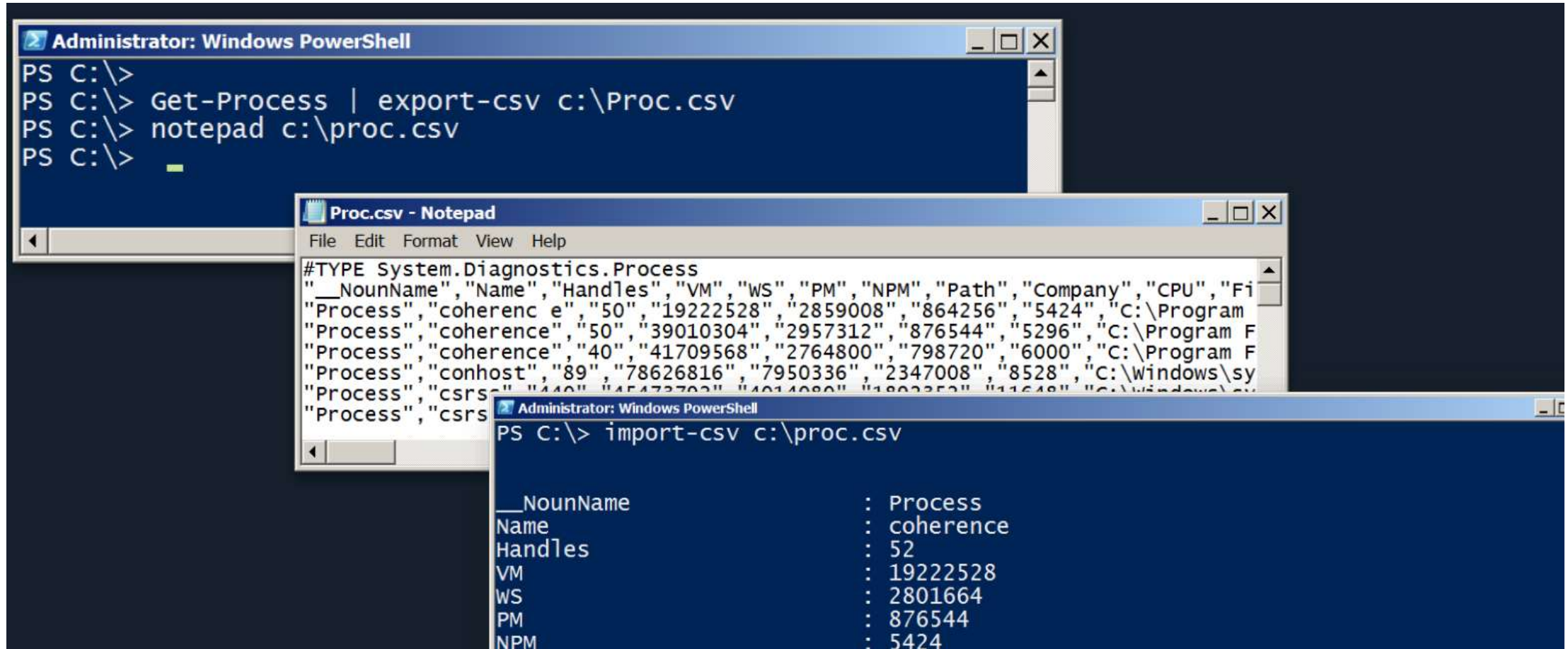
Ejecutar scripts

A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window has a blue background and a white border. The command prompt shows the following sequence of commands and outputs:

```
PS C:\> C:\Scripts\test.ps1
Hello world!
PS C:\> cd scripts
PS C:\scripts> .\test.ps1
Hello world!
PS C:\scripts> . .\test.ps1
Hello world!
PS C:\scripts> Get-Fun
MVA and Scripting with PowerShell Rocks!
PS C:\scripts>
```

The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side. The command prompt is currently at the "PS C:\scripts>" prompt.

Exportar csv



The image shows a Windows PowerShell window and a Notepad window. The PowerShell window has executed the following commands:

```
PS C:\>
PS C:\> Get-Process | export-csv c:\Proc.csv
PS C:\> notepad c:\proc.csv
PS C:\> _
```

The Notepad window, titled "Proc.csv - Notepad", displays the following CSV data:

```
#TYPE System.Diagnostics.Process
"__NounName","Name","Handles","VM","WS","PM","NPM","Path","Company","CPU","Fi
"Process","coherenc e","50","19222528","2859008","864256","5424","C:\Program
"Process","coherence","50","39010304","2957312","876544","5296","C:\Program F
"Process","coherence","40","41709568","2764800","798720","6000","C:\Program F
"Process","conhost","89","78626816","7950336","2347008","8528","C:\windows\sy
"Process","csrss","1440","15473702","1014080","1802252","11648","C:\windows\sy
"Process","csrss"
```

Below the Notepad window, the PowerShell window shows the command to import the CSV file:

```
PS C:\> import-csv c:\proc.csv
```

The output of the import command is displayed as a table:

__NounName	: Process
Name	: coherence
Handles	: 52
VM	: 19222528
WS	: 2801664
PM	: 876544
NPM	: 5424

Otros ficheros y salidas

```
Administrator: Windows PowerShell
PS C:\>
PS C:\> Get-Service > C:\serv.txt
PS C:\> Get-Service | Out-File c:\serv2.txt
PS C:\> Get-Service | Out-Printer
PS C:\> Notepad c:\serv.txt
PS C:\> Notepad c:\serv2.txt
PS C:\>
```

serv.txt - Notepad

Status	Name	DisplayName
Running	ADWS	Active Directory Web Services
Stopped	AeLookupSvc	Application Experience
Stopped	ALG	Application Layer Gateway Service

serv2.txt - Notepad

Status	Name	DisplayName
--------	------	-------------

cmdlets

- Son el corazón del funcionamiento de PowerShell
- Se pueden importar nuevos cmdlets con el uso de módulos
- Normalmente se pueden encadenar en tuberías
- Su nomenclatura suele ser “VerboNombre”
- La lista de verbos recomendados se puede consultar con “Get-Verb”

Cmdlets interessantes

- Get-ChildItem (ls)
- Show-Command (v3)
- Get-Command
- Get-Process
- Get-Member
- \$_
- Where-Object
- Get-Service
- Group-Object
- Get-Eventlog

```
Administrator: Windows PowerShell
PS C:\Windows\system32> get-command -name *set-vm*

CommandType      Name
-----
Cmdlet            Reset-VMRepli
Cmdlet            Reset-VMResou
Cmdlet            Set-VM
Cmdlet            Set-VMBios
Cmdlet            Set-VMComPort
```

```
Administrator: Windows PowerShell
PS C:\Windows\system32> Get-Process | Where-Object {$_.Name -like '*vm*'}

Handles  NPM(K)  PM(K)  WS(K) VM(M)  CPU(s)  Id
-----
1,30     492
1,44     1316
75,75    2032
632,80   2540
0,80     2652
3,52     3768
1,78     4684
```

```
Administrator: Windows PowerShell
PS C:\Windows\system32> Get-Service | Group-Object status

Count Name
-----
96 Running
107 Stopped

Group
-----
{AdobeARMservice, AlienFusionService, Appinfo,
{AeLookupSvc, ALG, AppIDSvc, AppMgmt...}
```

```
Administrator: Windows PowerShell
PS C:\Windows\system32> Get-EventLog system

Index Time      EntryType Source
-----
-----
```

COMMANDLETS PELIGROSOS...

... y sus salvavidas correspondientes

- Stop-Process | kill
- Stop-service
- Remove-Item
- -Confirm
- -Whatif
- \$WhatIfPreference

```
PS C:\> Remove-Item c:\serv.htm -Whatif
What if: Performing operation "Remove File" on Target "C:\serv.htm".
```

```
PS C:\> Remove-Item c:\serv.htm -Confirm
Confirm
Are you sure you want to perform this action?
Performing operation "Remove File" on Target "C:\serv.htm".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help
(default is "Y"):_
```

OBJETOS

- Todo en PowerShell son objetos
- Los objetos pueden tener propiedades y/o métodos
- Los resultados de un cmdlet pueden moverse entre cmdlets
- Generalmente se usa la técnica de inclusion (Un objeto tiene otros objetos) para representar datos más complejos

- Podemos incluirlos en tuberías

```
PS C:\> Get-Service | select-Object name, status
```



Status	Name	DisplayName
Running	AdobeARMservice	Adobe Acrobat Update Service
Running	ADWS	Active Directory Web Services
Stopped	AeLookupSvc	Application Experience
Stopped	ALG	Application Layer Gateway Service
Stopped	AllUserInstallA...	Windows All-User Install Agent

Name	Status
AdobeARMservice	Running
ADWS	Running
AeLookupSvc	Stopped
ALG	Stopped
AllUserInstallAgent	Stopped
AppHostSvc	Running
AppIDSvc	Stopped

- Get-Member (gm)
- TypeName es un nombre único asignado por Windows
- Muestra las propiedades y métodos de un objeto
- Las propiedades son columnas de información del objeto
- Los métodos son las acciones que puede realizar el objeto

```
PS C:\Users\Javier> Get-process | get-member

TypeName: System.Diagnostics.Process

Name      MemberType Definition
-----
Handles   AliasProperty Handles = Handlecount
Name       AliasProperty Name = ProcessName
NPM        AliasProperty NPM = NonpagedSystemMemorySize64
PM         AliasProperty PM = PagedMemorySize64
SI         AliasProperty SI = SessionId
VM         AliasProperty VM = VirtualMemorySize64
WS         AliasProperty WS = WorkingSet64
Disposed  Event      System.EventHandler Disposed(System.Object, System.EventArgs)
ErrorDataReceived Event      System.Diagnostics.DataReceivedEventHandler ErrorDataReceived(System.
Exited     Event      System.EventHandler Exited(System.Object, System.EventArgs)
OutputDataReceived Event      System.Diagnostics.DataReceivedEventHandler OutputDataReceived(System
BeginErrorReadLine Method      void BeginErrorReadLine()
BeginOutputReadLine Method      void BeginOutputReadLine()
CancelErrorRead Method      void CancelErrorRead()
CancelOutputRead Method      void CancelOutputRead()
Close      Method      void Close()
CloseMainWindow Method      bool CloseMainWindow()
CreateObjRef Method      System.Runtime.Remoting.ObjRef CreateObjRef(type requestedType)
Dispose    Method      void Dispose(), void IDisposable.Dispose()
Equals     Method      bool Equals(System.Object obj)
GetHashCode Method      int GetHashCode()
```

- Seleccionar objetos:
- Select-Object selecciona propiedades
- Usamos Get-Member para ver que propiedades podemos seleccionar
- -first y -last restringe el número de filas mostradas

- **GET-MEMBER**

- cmdlet **New-Item** que te permite entre otras cosas crear archivos. Así, lo primero es crear un archivo que se va a llamar ejemplo.txt. Para esto, ejecuta la siguiente instrucción,
- **New-Item -Name ejemplo.txt -ItemType File**
- puedes confirmar su existencia, verificando la propiedad Exists,
- **(Get-Item ejemplo.txt).Exists**
- Si quisieras ver todas las propiedades y métodos de tu nuevo objeto archivo llamado ejemplo.txt. Tienes que ejecutar la siguiente instrucción,
- **Get-Item ejemplo.txt | Get-Member**

- PROPIEDADES
- Si quieres mostrar solo las propiedades:
- **Get-Item ejemplo.txt | Get-Member -MemberType Property**

Métodos

- Ahora si lo que quisieras ver son los métodos que tiene tu archivo ejemplo.txt, lo que tienes que hacer es ejecutar la siguiente instrucción,
- **Get-Item ejemplo.txt | Get-Member -MemberType Method**

Ejemplos

- `(Get-Item ejemplo.txt).CopyTo("ejemplo2.txt")` copia tu primer archivo creado con PowerShell, `ejemplo.txt`, en tu segundo archivo `ejemplo2.txt`.
- `(Get-Item ejemplo.txt).Delete()` borra el archivo `ejemplo.txt`.
- `(Get-Item ejemplo.txt).GetHashCode()` te devuelve el código hash del archivo `ejemplo.txt`

VARIABLES EN POWERSHELL

- Una variable, es un espacio en memoria donde guardar información. Un espacio con un nombre. Dado que es necesario tener identificado a ese espacio para poder guardar o sacar esa información.
- En el caso de PowerShell la forma de identificar esos espacios en memoria es mediante una cadena de texto precedida por \$. Por ejemplo, \$variable.

- Para definir una variable es tan sencillo como hacer lo siguiente,
- `$variable = 1`
- De la misma manera puedes crear una variable que contenga una cadena de texto,
- `$variable = "Hola mundo"`
- O incluso que contenga un vector de enteros.
- `$variable = 1, 2, 3`

- puedes guardar el resultado de la ejecución de un cmdlet en una variable para utilizarlo posteriormente. Esto es especialmente cómodo cuando el resultado es un objeto y quieres utilizar alguno de los métodos del objeto. Por ejemplo, puedes guardar todos los procesos que están corriendo en tu sistema utilizando la siguiente instrucción,

- **\$procesos = Get-Process**

- Si quieres obtener el primero de los procesos, dado que se trata de un vector de procesos, tan solo tienes que ejecutar

- **Write-Output \$procesos[0]**

- si quisieras saber cuando comenzó ese proceso, tan solo tienes que ejecutar,

- **echo \$procesos[0].StartTime**

OPERACIONES CON VARIABLES EN POWERSHELL

- Eliminar contenido:
- **Clear-Variable -Name procesos**
- O bien directamente asignado \$null a la variable
- **\$procesos = \$null**

- **Eliminar variable**
- **Remove-Variable -Name cupsd**

TIPOS DE VARIABLES

- PS no es tipado.
- Se pueden crear tipos:
- tienes que preceder al nombre de la variable del tipo que le quieres asignar entre corchetes. Por ejemplo,
- `[double]$variable = 3.1415;`
- Eso si, la variable `$variable` no debe estar definida, o debe estar a `$null`

Comillas

- PS> Write-Output "La variable contiene \$variable"
 - La variable contiene 3.141592
 - PS> Write-Output 'La variable contiene \$variable'
 - La variable contiene \$variable
-
- PS> \$variable = 3.141592
 - PS> Write-Output "La variable `\$variable contiene \$variable"
 - La variable \$variable contiene 3.141592

ARRAY

- `$a = @()`
- los arrays tienen un tamaño fijo. (Para incremental usar `ARRAYLIST`)
- Puedes conocer el número de elementos de un array, utilizando la propiedad `Count`, como en el siguiente ejemplo,
- Otra forma
- `PS> $array = 1..3`
- `PS> Write-Output $array`
- 1
- 2
- 3
- `Write-Output $a.Count`

Acciones ARRAY

- ACCEDIENDO A LOS ELEMENTOS DEL ARRAY
- PS> \$a = 1, 2, 3, 4
- PS> \$a[0]
- 1
- El primer elemento de un array es el elemento 0.

- **PS> \$a = 'uno', 'dos', 'tres', 'cuatro', 'cinco'**

- **PS> Write-Output \$a[0, 1, 2]**

- uno

- dos

- tres

- **PS> Write-Output \$a[0..2]**

- uno

- dos

- Tres

- Pero no solo puedes obtener los valores en sentido directo, sino que también los puedes obtener en **sentido inverso** ¿como?

- **PS> Write-Output \$a[2..0]**

- tres

- dos

- Uno

- Ultimo elemento

- **PS> Write-Output \$a[-1]**

- cinco

- array, tiene algunos métodos muy interesantes, como son,
- Clear para eliminar el contenido, como por ejemplo **\$a.Clear()**
- Contains para saber si un elemento pertenece al array, **\$a.Contains('cinco')**.
- GetLowerBound permite obtener el índice mas bajo del array, como por ejemplo, `$a.GetLowerBound(0)`, que nos devolverá en este caso 0.
- GetUpperBound igual que el anterior pero para obtener el índice mayor. En nuestro ejemplo sería `$a.GetUpperBound(0)` y el resultado sería 4.
- IndexOf, te permite obtener el índice de un elemento, por ejemplo, `$a.IndexOf('cuatro')` devolverá 3.

Movernos : **FOREACH**

- PS> \$a.ForEach({Write-Output \$PSItem})
 - uno
 - dos
 - tres
 - cuatro
 - cinco
 - Evidentemente para esto no tiene mucho sentido, pero ¿que tal esto?
-
- PS> \$a = 0, 1, 2
 - PS> \$a.ForEach({\$PSItem * 3})
 - 0
 - 3
 - 6

WHERE

- Este método te permite filtrar los elementos del array y obtener un nuevo array, con solo aquellos elementos que cumplen la condición que hayas establecido. Por ejemplo,
- PS> \$a = 0, 1, 2, 3, 4, 5, 6
- PS> \$a.Where({\$PSItem > 4})
- 5
- 6

OPERADORES

Operador de comparación	Significado	Ejemplo (devuelve el valor True)
-eq	Es igual a	1 -eq 1
-ne	Es distinto de	1 -ne 2
-lt	Es menor que	1 -lt 2
-le	Es menor o igual que	1 -le 2
-gt	Es mayor que	2 -gt 1
-ge	Es mayor o igual que	2 -ge 1
-like	Es como (comparación de caracteres comodín para texto)	"Negu" -like "N*"
-notlike	No es como (comparación de caracteres comodín para texto)	"Negu" -notlike "P*"
-contains	Contiene	1,2,3 -contains 1
-notcontains	No contiene	1,2,3 -notcontains 4

Aritméticos

Operador	Significado	Ejemplo
*	Multiplicación	$a * 5$
/	Division	$a / 5$
+	Suma	$a + 5$
-	Menos	$a - 5$
%	Resto	$a \% 5$

Lógicos

Operador	Descripción	Ejemplo (devuelve el valor True)
-And	Todas las partes de la expresión tienen que ser True	(5 -gt 1) -And (5 -lt 10)
-Or	Alguna de las partes de la expresión tiene que ser True	(5 -gt 1) -Or (5 -lt 1)
-Xor	Exclusión lógica. Es True cuando una parte es True y otra False	(5 -gt 1) -Xor (5 -lt 1)
-Not (!)	Negación	-Not (5 -lt 1)

Operador	Significado	Ejemplo
=	Asigna un valor	\$a = 5
+=	Suma el valor indicado al ya existente	\$a += 5
-=	Resta el valor indicado al ya existente	\$a -= 5
*=	Multiplica el valor indicado al ya existente	\$a *= 5
/=	Divide por el valor indicado el ya existnte	\$a /= 5
++	Incrementa el valor en 1	\$a++
—	Decrementa el valor en 1	\$a—

IF

- \$valor = "Hola"
- if (\$valor -eq "Hola"){
- Write-Output "Son iguales"
- }else{
- Write-Output "Son distintos"
- }

- \$valor = "Hola"
- if (\$valor -eq "Hola"){
- Write-Output "es igual a Hola"
- }elseif(\$valor -eq "Adios"){
- Write-Output "es igual a Adios"
- }else{
- Write-Output "Ni es igual a Hola ni a Adios"
- }

Switch

- \$valor = "Hola"
- switch(\$valor){
- "Hola"{Write-Output "Es igual a Hola"}
- "Adios"{Write-Output "Es igual a Adios"}
- default{Write-Output "No se parece a nada
- }

Bucles

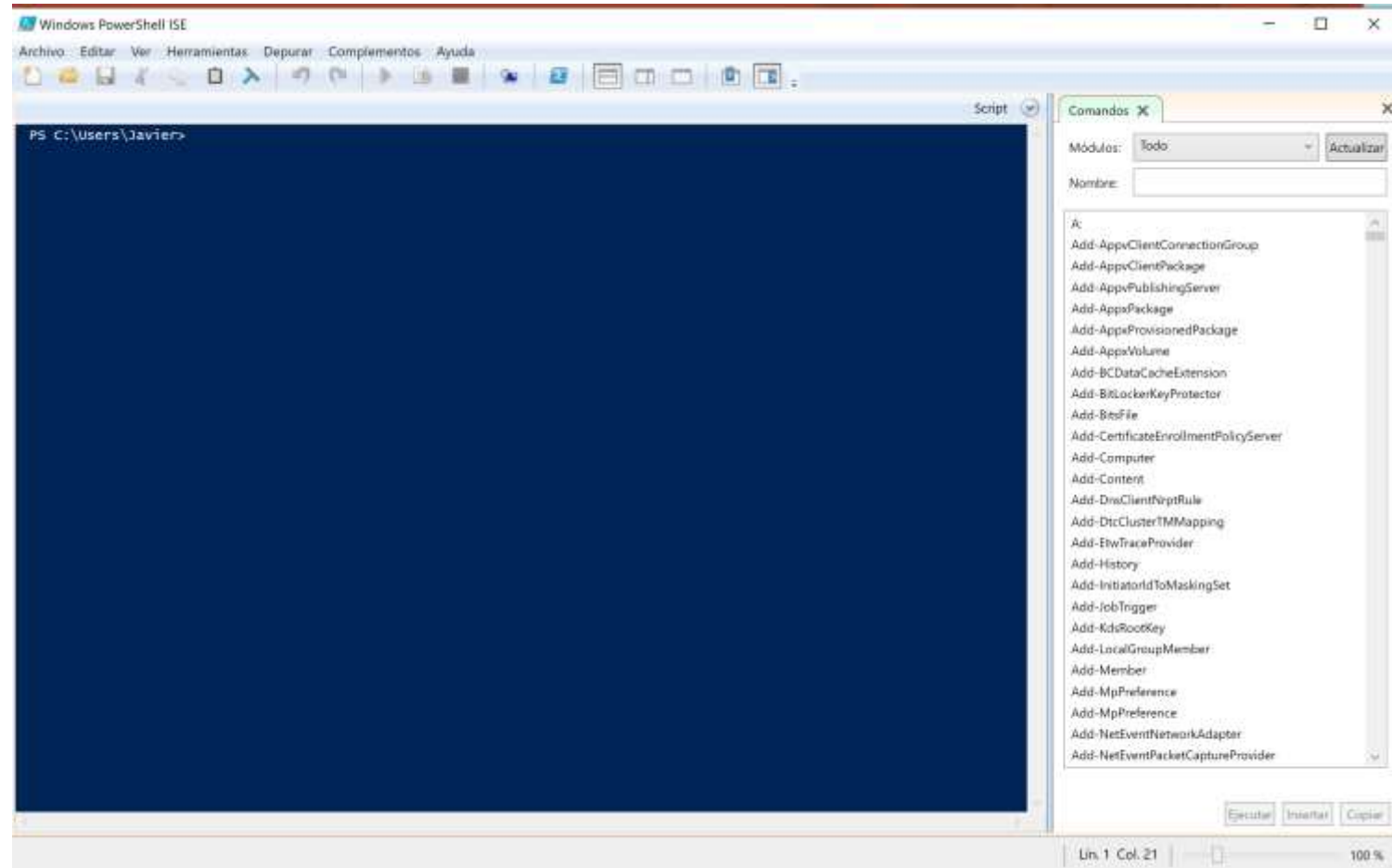
- Get-Process, si quieres conocer el nombre de cada uno de los procesos que están corriendo en tu sistema, lo podrías hacer entre otras formas de la siguiente,
- \$ps = Get-Process
- for(\$i=0; \$i -lt \$ps.Length; \$i++){
- Write-Output \$ps[\$i].Name
- }

Ejecución de scripts

- A partir de PowerShell 3.0, puede ejecutar scripts desde Explorador de archivos.
- Para usar la característica "Ejecutar con PowerShell":
- Ejecute Explorador de archivos, haga clic con el botón derecho en el nombre de archivo del script y seleccione "Ejecutar con PowerShell".

Powershell ISE

- powershell ise



- Para escribir un script, abra un nuevo archivo en un editor de texto, escriba los comandos y guárdelos en un archivo con un nombre de archivo válido con la extensión de .ps1 archivo.
- El ejemplo siguiente es un script sencillo que obtiene los servicios que se ejecutan en el sistema actual y los guarda en un archivo de registro. El nombre de archivo de registro se crea a partir de la fecha actual.
- `$date = (get-date).dayofyear`
- `get-service | out-file "$date.log"`

Ejecución de scripts en otros equipos

- se el parámetro FilePath del Invoke-Command cmdlet .
- Escriba la ruta de acceso y el nombre de archivo del script como valor del parámetro FilePath . El script debe encontrarse en el equipo local o en un directorio al que el equipo local pueda acceder.
- El siguiente comando ejecuta el Get-ServiceLog.ps1 script en los equipos remotos denominados Server01 y Server02.
- Invoke-Command -ComputerName Server01,Server02 -FilePath `
- C:\Scripts\Get-ServiceLog.ps1

Errores

- Políticas de seguridad:
- Get-ExecutionPolicy,

```
PS C:\Users\Javier> Get-ExecutionPolicy -list
```

Scope	ExecutionPolicy
-----	-----
MachinePolicy	Undefined
UserPolicy	Undefined
Process	Undefined
CurrentUser	Undefined
LocalMachine	Undefined

- Para establecer nuevas reglas en este campo, se debe usar la siguiente comando: (Es recomendable utilizar el principio de mínimo privilegio)
- Set-ExecutionPolicy RemoteSigned
- o
- Set-ExecutionPolicy Unrestricted
- y ya con este método se soluciona.

```
PS C:\Users\Javier> Set-ExecutionPolicy RemoteSigned
```

Cambio de directiva de ejecución

La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha directiva, podrías exponerte a los riesgos de seguridad descritos en el tema de la Ayuda `about_Execution_Policies` en <https://go.microsoft.com/fwlink/?LinkID=135170>. ¿Quieres cambiar la directiva de ejecución?

[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda (el valor predeterminado es "N"): S

Set-ExecutionPolicy : Se denegó el acceso a la clave de Registro

'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell'. Para cambiar la directiva de ejecución para el ámbito (LocalMachine) predeterminado, inicie Windows PowerShell con la opción "Ejecutar como administrador". Para cambiar la directiva de ejecución para el usuario actual, ejecute "Set-ExecutionPolicy -Scope CurrentUser".

En línea: 1 Carácter: 1

+ Set-ExecutionPolicy RemoteSigned

+ ~~~~~

+ CategoryInfo : PermissionDenied: (:) [Set-ExecutionPolicy], UnauthorizedAccessException

+ FullyQualifiedErrorId : System.UnauthorizedAccessException,Microsoft.PowerShell.Commands.SetExecutionPolicyCommand

Administrador: Windows PowerShell

Windows PowerShell

Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma <https://aka.ms/pscore6>

PS C:\windows\system32> Set-ExecutionPolicy RemoteSigned

Cambio de directiva de ejecución

La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha directiva, podrías exponerte a los riesgos de seguridad descritos en el tema de la Ayuda `about_Execution_Policies` en <https://go.microsoft.com/fwlink/?LinkID=135170>. ¿Quieres cambiar la directiva de ejecución?

[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda (el valor predeterminado es "N"): S

PS C:\windows\system32>