GESTION DE PROCESOS Y SERVICIOS EN WINDOWS

PROCESOS

Prioridad de un proceso y sus hebras

• Un proceso pertenece a una clase de prioridad:
– Desocupado (4)
– Normal (7 o 9)
– Alta (13)
– Tiempo Real (24)

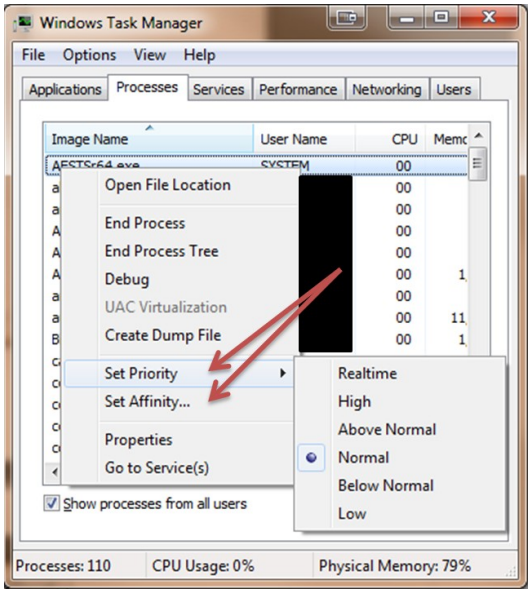
• Una hebra hereda el valor de prioridad de la clase a la que pertenece el proceso pero después este valor puede variar si tiene prioridad dinámica

Las hebras pueden tener 32 niveles de prioridad divididos en dos grupos:

- Prioridades de tiempo real [16..31] con privilegios de administrador. Son fijas.
- Prioridades dinámicas [1,,15] aplicaciones típicas. Son variables.

Utilidades gestión de procesos en Windows

- Process Explorer de Sysinternals
- Comando tasklist del cmd
- Comando start del cmd
- Administrador de tareas (taskmgr)
- cmdlet GET-PROCESS de Powershell



Comando start (CMD)

Inicia una ventana aparte para ejecutar un programa o un comando especificado.

START ["título"] [/Druta] [/I] [/MIN] [/MAX] [/SEPARATE | /SHARED]
[/LOW | /NORMAL | /HIGH | /REALTIME | /ABOVENORMAL | /BELOWNORMAL]
[/WAIT] [/B] [comando o programa]
[parámetros]

"título" Texto que se mostrará en la barra de t¡tulo de la ventana.
ruta Directorio de inicio.
B Inicia la aplicaci¢n sin crear una ventana. La aplicaci¢n no controla la funcionalidad de ^. A menos que la aplicaci¢n habilite el procesamiento de ^C processing, ^Pausa es la £nica manera de interrumpir la aplicaci¢n.
I El nuevo entorno será el entorno original pasado a cmd.exe y no el actual.
MIN Abre la ventana minimizándola.

MAX Abre la ventana minimizándola.
SEPARATE Inicia el programa de Windows de en un espacio de memoria separado.
SHARED Inicia el programa de Windows de en el espacio de memoria compartida.
LOW Inicia la aplicaci¢n en la clase de prioridad IDLE.
NORMAL Inicia la aplicaci¢n en la clase de prioridad NORMAL.
HIGH Inicia la aplicaci¢n en la clase de prioridad HIGH.
REALTIME Inicia la aplicaci¢n en la clase de prioridad REALTIME.
ABOVENORMAL Inicia la aplicaci¢n en la clase de prioridad ABOVENORMAL.
BELOWNORMAL Inicia la aplicaci¢n en la clase de prioridad BELOWNORMAL.
WAIT Inicia la aplicaci¢n y espera a que esta finalice.
Comando o programa

Si es un comando interno de cmd o un archivo por lotes el comando se ejecuta con el modificador /K en cmd.exe.
Esto significa que la ventana continuará abierta una vez que el comando se haya ejecutado.

Si no es un comando interno de cmd o un archivo por lotes entonces es un programa y se ejecutará como una aplicación con ventanas o como una aplicaci¢n de consola.

Comando tasklist (CMD)

```
TASKLIST [/S sistema [/U usuario [/P contraseña]]]
      [/M [módulo] | /SVC | /V] [/FI filtro] [/FO formato] [/NH]

Descripci¢n:
   Esta herramienta de la l¡nea de comandos muestra una lista de
   aplicaciones y las tareas o procesos asociados que se ejecutan
   en un sistema local o remoto

Lista de parámetros:
  /S    sistema           Especifica el sistema remoto para conectarse.

  /U    [dominio\]usuario  Especifica el contexto de usuario en el que
                           que el comando debe ejecutarse.

  /P    contrase¤a         Especifica la contrase¤a para el contexto de
                           usuario dado.

  /M    [m¢dulo]           Muestra todas las tareas que tienen cargados
                           m¢dulos de biblioteca DLL que coincidan con el
                           nombre est ndar dado.
                           Si no se especifica el nombre del m¢dulo, se
                           mostrar n todos los m¢dulos cargados por cada
                           tarea.

  /SVC                     Muestra los servicios en cada proceso.

  /V                       Especifica que la informaci¢n
                           sea mostrada.
  /FI   filtro             Muestra un conjunto de tareas que coinciden
                           con el criterio especificado por el filtro.

  /FO  formato             Especifica el formato de salida.
                           Valores v lidos: "TABLE", "LIST", "CSV".

  /NH                      Especifica que el "encabezado de columna" no
                           no debe mostrarse en la salida.
                           V lido s¢lo para formatos "TABLE" y "CSV".

  /?                       Muestra el uso/ayuda.

Filtros:
   Nombre filtro  Operadores v lidos          Valores v lidos
   -------------  ------------------          ---------------
   STATUS         eq, ne                      RUNNING | NOT RESPONDING
   IMAGENAME      eq, ne                      Nombre de imagen
   PID            eq, ne, gt, lt, ge, le      Valor del PID
   SESSION        eq, ne, gt, lt, ge, le      N£mero de sesi¢n
```

```
SESSIONNAME     eq, ne                    Nombre de sesi¢n
CPUTIME         eq, ne, gt, lt, ge, le    Tiempo de la CPU en el formato
                                          hh:mm:ss.
                                          hh - n£mero de horas,
                                          mm - minutos, ss - segundos
MEMUSAGE        eq, ne, gt, lt, ge, le    Uso de memoria en KB
USERNAME        eq, ne                    Nombre de usuario en formato
                                          [dominio\]usuario
SERVICES        eq, ne                    Nombre de servicio
WINDOWTITLE     eq, ne                    T¡tulo de ventana
MODULES         eq, ne                    Nombre DLL
```

Ejemplos:

```
TASKLIST
TASKLIST /M
TASKLIST /V
TASKLIST /SVC
TASKLIST /M wbem*
TASKLIST /S sistema /FO LIST
TASKLIST /S sistema /U dominio nombreusuario /FO CSV /NH
TASKLIST /S sistema /U nombreusuario /P contrase¤a /FO TABLE /NH
TASKLIST /FI "USERNAME ne NT AUTHORITY\SYSTEM" /FI "STATUS eq running"
```

NOTA: Es especialmente útil TASKLIST /SVC para obtener los servicios que presta un proceso (svchost).

Get-Process (PowerShell)

Obtener procesos (Get-Process)

Para obtener los procesos que se están ejecutando en el equipo local, ejecute **Get-Process** sin parámetros.

Puede obtener determinados procesos especificando sus nombres de proceso o identificadores de proceso. El siguiente comando obtiene el proceso inactivo:

```
PS> Get-Process -id 0

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
      0       0        0         16     0                0 Idle
```

Aunque es normal que los cmdlets no devuelvan datos en algunas situaciones, cuando se especifica un proceso por su id. de proceso **Get-Process**, genera un error si no encuentra ninguna coincidencia, porque la intención habitual consiste en recuperar un proceso en ejecución conocido. Si no hay ningún proceso con ese identificador, es probable que el identificador sea incorrecto o que el proceso de interés haya terminado:

```
PS> Get-Process -Id 99

Get-Process : No process with process ID 99 was found.
At line:1 char:12
+ Get-Process  <<<< -Id 99
```

Puede usar el parámetro Name del cmdlet Get-Process para especificar un subconjunto de procesos basado en el nombre del proceso. El parámetro Name puede tomar varios nombres de una lista de nombres separados por comas y admite el uso de caracteres comodín, para que pueda escribir patrones de nombre.

Por ejemplo, el siguiente comando obtiene el proceso cuyos nombres comienzan por "ex".

```
PS> Get-Process -Name ex*

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    234       7     5572      12484   134     2.98   1684 EXCEL
    555      15    34500      12384   134   105.25    728 explorer
```

Dado que la clase System.Diagnostics.Process de .NET es la base de los procesos de Windows PowerShell, sigue algunas de las convenciones usadas por System.Diagnostics.Process. Una de estas convenciones es que el nombre de proceso de un archivo ejecutable nunca incluya ".exe" al final del nombre del ejecutable.

Get-Process también acepta varios valores para el parámetro Name.

```
PS> Get-Process -Name exp*,power*

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    540      15    35172      48148   141    88.44    408 explorer
    605       9    30668      29800   155     7.11   3052 powershell
```

Puede usar el parámetro ComputerName de Get-Process para obtener procesos en equipos remotos. Por ejemplo, el comando siguiente obtiene los procesos de PowerShell en el equipo local (representado por "localhost") y en dos equipos remotos.

```
PS> Get-Process -Name PowerShell -ComputerName localhost, Server01, Server02

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    258       8    29772      38636   130             3700 powershell
    398      24    75988      76800   572             5816 powershell
    605       9    30668      29800   155     7.11    3052 powershell
```

Los nombres de equipo no son evidentes en esta presentación, pero se almacenan en la propiedad MachineName de los objetos de proceso que devuelve Get-Process. El siguiente comando usa el cmdlet Format-Table para mostrar el identificador de proceso y las propiedades ProcessName y MachineName (ComputerName) de los objetos de proceso.

```
PS> Get-Process -Name PowerShell -ComputerName localhost, Server01, Server01 | Format-Table
-Property ID, ProcessName, MachineName

  Id ProcessName MachineName
  -- ----------- -----------
3700 powershell  Server01
3052 powershell  Server02
5816 powershell  localhost
```

Este comando más complejo agrega la propiedad MachineName a la presentación estándar de Get-Process.

```
PS> Get-Process powershell -ComputerName localhost, Server01, Server02 |
    Format-Table -Property Handles,
        @{Label="NPM(K)";Expression={[int]($_.NPM/1024)}},
        @{Label="PM(K)";Expression={[int]($_.PM/1024)}},
        @{Label="WS(K)";Expression={[int]($_.WS/1024)}},
        @{Label="VM(M)";Expression={[int]($_.VM/1MB)}},
        @{Label="CPU(s)";Expression={if ($_.CPU -ne $()){$_.CPU.ToString("N")}}},
        Id, ProcessName, MachineName -auto

Handles  NPM(K)  PM(K) WS(K) VM(M) CPU(s)  Id ProcessName  MachineName
-------  ------  ----- ----- ----- ------  -- -----------  -----------
    258       8  29772 38636   130         3700 powershell Server01
    398      24  75988 76800   572         5816 powershell localhost
    605       9  30668 29800   155 7.11    3052 powershell Server02
```

Detener procesos (Stop-Process)

Windows PowerShell ofrece flexibilidad para enumerar procesos, pero ¿qué hay de detenerlos?

El cmdlet **Stop-Process** toma un nombre o un identificador para especificar un proceso que quiere detener. Su capacidad para detener procesos dependerá de sus permisos. Algunos procesos no se pueden detener. Por ejemplo, si intenta detener el proceso inactivo, obtendrá un error:

```
PS> Stop-Process -Name Idle
Stop-Process : Process 'Idle (0)' cannot be stopped due to the following error:
 Access is denied
At line:1 char:13
+ Stop-Process  <<<< -Name Idle
```

También puede forzar la solicitud con el parámetro **Confirm**. Este parámetro es especialmente útil si se usa un carácter comodín al especificar el nombre del proceso, ya que accidentalmente podría coincidir con algunos procesos que no quiere detener:

```
PS> Stop-Process -Name t*,e* -Confirm
Confirm
Are you sure you want to perform this action?
Performing operation "Stop-Process" on Target "explorer (408)".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help
(default is "Y"):n
Confirm
Are you sure you want to perform this action?
Performing operation "Stop-Process" on Target "taskmgr (4072)".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help
(default is "Y"):n
```

La manipulación de procesos complejos es posible si se usan algunos cmdlets de filtrado de objetos. Dado que un objeto Process tiene una propiedad Responding que es true cuando ya no responde, puede detener todas las aplicaciones que no respondan con el comando siguiente:

```
Get-Process | Where-Object -FilterScript {$_.Responding -eq $false} | Stop-Process
```

Puede usar el mismo enfoque en otras situaciones. Por ejemplo, supongamos que una aplicación de área de notificaciones secundaria se ejecuta automáticamente cuando los usuarios inician otra aplicación. Es posible que esto no funcione correctamente en las sesiones de Terminal Services, pero lo quiera seguir manteniendo en las sesiones que se ejecutan en la consola del equipo físico. Las sesiones conectadas en el escritorio del equipo físico siempre tienen un identificador de la sesión 0, por lo que puede detener todas las instancias del proceso que están en otras sesiones mediante **Where-Object** y el proceso,

SessionId:PowerShell

```
Get-Process -Name BadApp | Where-Object -FilterScript {$_.SessionId -neq 0} | Stop-Process
```

El cmdlet Stop-Process no tiene un parámetro ComputerName. Por lo tanto, para ejecutar un comando para detener un proceso en un equipo remoto, debe usar el cmdlet Invoke-Command. Por ejemplo, para detener el proceso de PowerShell en el equipo remoto Server01, escriba:PowerShell

```
Invoke-Command -ComputerName Server01 {Stop-Process Powershell}
```

SERVICIOS

¿Qué es un servicio?

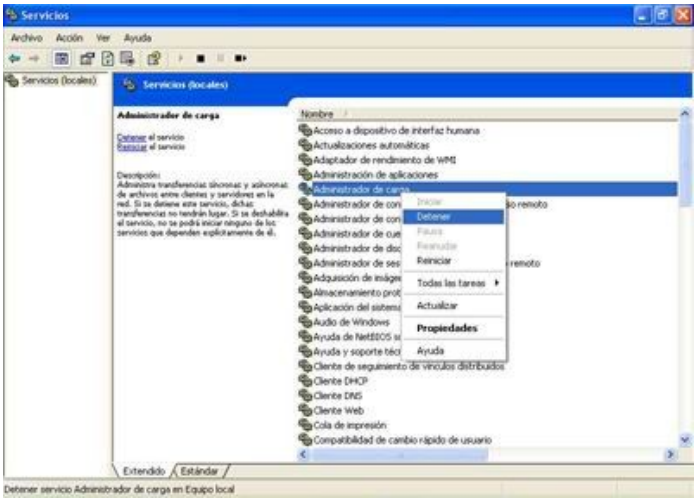
Los servicios no son nada más ni nada menos que programas o aplicaciones cargadas por el propio sistema operativo. Estas aplicaciones tienen la particularidad que se encuentran corriendo en segundo plano (Background).

Por defecto, con la instalación, se instalan y ejecutan una cierta cantidad de servicios. De más está decir, que dependiendo de nuestras necesidades, podemos necesitarlos a todos o no.
Como sabemos, mientras más aplicaciones tengamos ejecutándose consumimos más recursos, por lo tanto, vamos a tratar de deshabilitar lo que no utilizamos.

¿Dónde veo los servicios?

Para visualizar los servicios, o para cambiar algunas de sus opciones y/o estados, debemos abrir la consola de Microsoft.
Esto lo podemos hacer yendo a:
- Inicio / Panel de control / Rendimiento y mantenimiento / Herramientas Administrativas / Servicios o - Inicio / Panel de control / Herramientas Administrativas / Servicios dependiendo de cómo tengamos configurada la vista de Panel de Control.

Estos pasos pueden ser reemplazados por lo siguiente:
Nos dirigimos a Inicio, Ejecutar, escribimos **services.msc** y presionamos Enter.



¿Cómo inicio o detengo un servicio? (CMD)

Una vez en la consola, nos posicionamos arriba del servicio que queremos iniciar o detener y haciendo click con el boton derecho vamos a ver las acciones correspondientes.

Otras maneras de iniciar o detener un servicio
Desde la consola, podemos hacerlo utilizando los comandos NET START y NET STOP. Para iniciar y detener un servicio, respectivamente.

El modo de uso es: NET START/STOP NombreDelServicio

Dónde NombreDelServicio es el nombre del servicio completo (entre "" comillas si contiene espacios) o el nombre abreviado.
Ejemplos:

net start BITS
net start "Background Intelligent Transfer Service"
net stop "Automatic Updates"

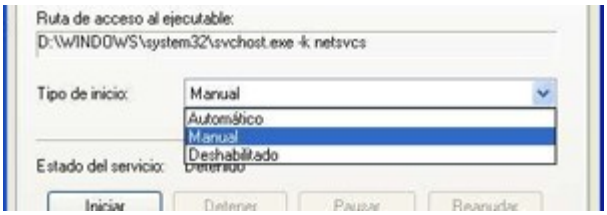
Recuperando un servicio

Supongamos que necesitamos recuperar la manera de inicio de alguno de los servicios, pero por alguna razón, no podemos iniciar la consola. ¿Qué podemos hacer?
No dirigimos a Inicio / Ejecutar
Escribimos regedit y presionamos Enter
Expandimos la siguiente clave:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services] y buscamos el servicio en cuestion. Luego seleccionamos la clave Start y con click derecho elegimos modificar.
Las opciones que tenemos son:
2 – Automático
3 – Manual
4 – Deshabilitado
Por último, aceptamos y reiniciamos la computadora.

¿Diferentes estados?

Los servicios pueden encontrarse en dos estados posibles. Pueden estar iniciados, es decir, se encuentra ejecutándose/corriendo o puede estar detenido.
Y tenemos tres opciones posibles de inicio:
- Automático: Se inician junto con el sistema operativo.
- Manual: Podemos iniciarlo y detenerlo manualmente cuando queramos u otro servicio puede hacerlo automáticamente. En un principio estaría detenido.
- Deshabilitado: No se puede iniciar manualmente ni otro servicio puede hacerlo.

Para cambiar la manera en que se inicia un servicio, debemos dirigirnos a la consola. Una vez ahí elegimos el servicio con el cual vamos a trabajar, hacemos click con el botón derecho del mouse y elegimos propiedades.



Creación de un servicio (CMD)

Puede utilizar **Sc.exe** como ayuda para desarrollar servicios para Windows. Sc.exe, que se incluye en el Kit de recursos, implementa llamadas a todas las funciones de la interfaz de programación de aplicaciones (API) de control de servicios de Windows. Puede establecer los parámetros de esta función cualquiera de las funciones de API de control de servicios y modificar cualquier parámetro desde la línea de comandos. La ventaja de ello es que ofrece una forma cómoda de crear o configurar la información si los especifica en la línea de comandos. Sc.exe también muestra el estado de los servicios y recupera los valores almacenados en los campos de estructura de estado. Asimismo, la herramienta le permite especificar el nombre de un equipo remoto de forma que pueda llamar a las funciones de API de los servicios o ver las estructuras de estado de servicio en el equipo remoto.

Sc.exe utiliza la sintaxis siguiente:

Sintaxis1 (utilice Sintaxis1 para ejecutar Sc.exe)

sc [nombreDeServidor] Comando nombreDeServicio [nombreDeOpción = valorDeOpción...]

Sintaxis2 (utilice Sintaxis2 para mostrar información de Ayuda, excepto para el comando **query**)

sc [Comando]

Get-Service (PowerShell)

Dado que no siempre es evidente cuál es el nombre real del servicio, es posible que necesite buscar servicios por el nombre para mostrar. Puede hacerlo por el nombre específico, mediante caracteres comodín o con una lista de nombres para mostrar: PowerShell

```
PS> Get-Service -DisplayName se*

Status   Name               DisplayName
------   ----               -----------
Running  lanmanserver       Server
Running  SamSs              Security Accounts Manager
Running  seclogon           Secondary Logon
Stopped  ServiceLayer       ServiceLayer
Running  wscsvc             Security Center

PS> Get-Service -DisplayName ServiceLayer,Server

Status   Name               DisplayName
------   ----               -----------
Running  lanmanserver       Server
Stopped  ServiceLayer       ServiceLayer
```

Puede usar el parámetro ComputerName del cmdlet Get-Service para obtener los servicios en equipos remotos. El parámetro ComputerName acepta varios valores y caracteres comodín, por lo que puede obtener los servicios en varios equipos con un solo comando. Por ejemplo, el siguiente comando obtiene los servicios en el equipo remoto Server01.

```
Get-Service -ComputerName Server01
```

Obtener servicios necesarios y dependientes

El cmdlet Get-Service tiene dos parámetros que son muy útiles para la administración de servicios. El parámetro DependentServices obtiene servicios que dependen del servicio. El parámetro RequiredServices obtiene servicios de los que depende este servicio.

Estos parámetros solo muestran los valores de las propiedades DependentServices y ServicesDependedOn (alias=RequiredServices) del objeto System.ServiceProcess.ServiceController que devuelve Get-Service, pero simplifican los comandos y hacen que resulte mucho más fácil obtener esta información.

El comando siguiente obtiene los servicios que el servicio LanmanWorkstation requiere.

```
PS> Get-Service -Name LanmanWorkstation -RequiredServices

Status   Name               DisplayName
------   ----               -----------
Running  MRxSmb20           SMB 2.0 MiniRedirector
Running  bowser             Bowser
Running  MRxSmb10           SMB 1.x MiniRedirector
Running  NSI                Network Store Interface Service
```

El comando siguiente obtiene los servicios que requieren el servicio LanmanWorkstation.

```
PS> Get-Service -Name LanmanWorkstation -DependentServices
```

```
Status   Name               DisplayName
------   ----               -----------
Running  SessionEnv         Terminal Services Configuration
Running  Netlogon           Netlogon
Stopped  Browser            Computer Browser
Running  BITS               Background Intelligent Transfer Ser...
```

También puede obtener todos los servicios que tengan dependencias. El comando siguiente hace justamente eso y, después, usa el cmdlet Format-Table para mostrar las propiedades Status, Name, RequiredServices y DependentServices de los servicios en el equipo.

Get-Service -Name * | Where-Object {$_.RequiredServices -or $_.DependentServices} | Format-Table -Property Status, Name, RequiredServices, DependentServices -auto

Detener, iniciar, suspender y reiniciar los servicios

Todos los cmdlets Service tienen el mismo formato general. Los servicios pueden especificarse por el nombre común o el nombre para mostrar, y toman listas y caracteres comodín como valores. Para detener el administrador de trabajos de impresión, use:

Stop-Service -Name spooler

Para iniciar el administrador de trabajos de impresión una vez detenido, use:

Start-Service -Name spooler

Para suspender el administrador de trabajos de impresión, use:

Suspend-Service -Name spooler

El cmdlet Restart-Service funciona de la misma manera que los otros cmdlets Service, pero mostraremos algunos ejemplos más complejos. En el uso más simple, debe especificar el nombre del servicio:

PS> Restart-Service -Name spooler

WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...
WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...
PS>

Observará que obtiene un mensaje de advertencia repetido sobre el inicio del administrador de trabajos de impresión. Si realiza una operación de servicio que tarda bastante tiempo, Windows PowerShell le notificará que todavía está intentando realizar la tarea.

Si desea reiniciar varios servicios, puede obtener una lista de estos, filtrarlos y, después, ejecutar el reinicio:

PS> Get-Service | Where-Object -FilterScript {$_.CanStop} | Restart-Service

WARNING: Waiting for service 'Computer Browser (Browser)' to finish stopping...
WARNING: Waiting for service 'Computer Browser (Browser)' to finish stopping...
Restart-Service : Cannot stop service 'Logical Disk Manager (dmserver)' because
 it has dependent services. It can only be stopped if the Force flag is set.
At line:1 char:57
+ Get-Service | Where-Object -FilterScript {$_.CanStop} | Restart-Service <<<<
WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...
WARNING: Waiting for service 'Print Spooler (Spooler)' to finish starting...

Estos cmdlets Service no tienen un parámetro ComputerName, pero se pueden ejecutar en un equipo remoto mediante el cmdlet Invoke-Command. Por ejemplo, el siguiente comando reinicia el servicio de administrador de trabajos en cola en el equipo remoto Server01.

Invoke-Command -ComputerName Server01 {Restart-Service Spooler}