TEMA 04 – WINDOWS. ADMINISTRACIÓN Y CONFIGURACIÓN – SCRIPTS

Introducción

- Órdenes de Powershell: cmdlets (command-let)
- Al principio: 129 cmdlets básicos
- Pueden incluirse conjuntos específicos para trabajar con Active Directory, Exchange u otros roles de servidor
- 2 opciones:
    - PowerShell ISE (Integrated Scripting Environment)
    - Consola por línea de commandos
- Version actual: 5.1

Commandos y cmdlets

- Fáciles de recorder
- Convención de nomenclatura “verbo-sustantivo”
- Suele indicar claramente su función

Redirecciones y tuberías

- Obtener la salida de una acción y dirigirla a un lugar diferente del predeterminado
- Orientadas a la obtención de un archivo de texto con la salida que ofrece un cmdlet
    - >: nuevo archivo y deposita la salida. Si existe, borra el contenido previo
    - >>: lo añade al final del documento. Si no existe, lo crea
    - |: conecta la salida de un cmdlet con la entrada de otro (n veces)

Primeros pasos con PowerShell

- Niveles de permisibilidad:
    - Restricted: no se permite la ejecución de scripts
    - AllSigned: deberán estar autenticados antes de ejecutarlos, es la más restrictiva
    - RemoteSigned: deberán estar autenticados los scripts que procedan de una ubi remota (p.e.: descargados)
    - Unrestricted: se ejecutará cualquier script
- Con “Get-ExecutionPolicy”
- Para establecerla: “Set-ExecutionPolicy”
- Ejecutar script: “.\script.ps1” o “ruta-absoluta\script.ps1

Variables en PowerShell

- Solamente hay que llamarla
- Restricciones:
    - Primer carácter: $
    - Letras, números o símbolos
    - Espacios en blancos: hay qe meter el nombre entre llaves {}
- También se puede con New-Variable Variable (no hace falta el $)
    - -Name (por si metemos más opciones, también se puede a secas si solo este)

    - -Value
    - -Opcion: readonly…
- Get-Variable: lista completa de las variables que se hayan definido hasta el momento
- Pedir información: $Nombre = Read-Host “¿Cómo te llamas?”
- Mostrar información: Write-Host “Bienvenido” / Write-Host $Nombre
- Tipos de datos: [string] [char] [long] [double] [decima] [datetime] [bool] {array]
- Tipo de dato de una variable: Write-Host $Nombre.GetType().Name
- Rango de valores
    - [decimal]::MinValue
    - [decimal]::MaxValue
- Establecer tipo de dato: [float] $precio = 4.99 / $precio = [float] 4.99
    - Write-Host ([int]$precio)

Observaciones básicas con variables en PowerShell

- Operar con variables de tipo texto
    - $mensaje = $saludo + “ “ + $nombre / $mensaje = “$saludo $nombre”
- Operadores aritméticos

Operadores aritméticos		
Operador	Función	Ejemplo
+	suma	$resultado = $x + 3
-	resta	$resultado = $x - 3
*	multiplicación	$resultado = $x * 3
/	división	$resultado = $x / 3
%	resto	$resultado = $x % 3

Operadores aritméticos especiales			
Operador	Significado	Ejemplo	Equivalencia
+=	Incrementa el valor de la variable	$x += 3	$x = $x + 3
-=	Reduce el valor de la variable	$x -= 3	$x = $x - 3
*=	Multiplica el valor de la variable	$x *= 3	$x = $x * 3
/=	Divide el valor de la variable	$x /= 3	$x = $x / 3

Operadores aritméticos especiales			
Operador	Significado	Ejemplo	Equivalencia
++	Incrementa en 1 el valor de la variable	$x++	$x = $x + 1
--	Reduce 1 del valor de la variable	$x--	$x = $x - 1

- **Operadores de comparación**

Operadores de comparación		
Operador de comparación	Significado	Ejemplo ($true)
-eq	Igual	1 -eq 1
-ieq	Igual Con valores de tipo texto, no tendrá en cuenta la diferencia entre mayúsculas y minúsculas	"Hola" -ieq "HOLA"
-ceq	Igual Con valores de tipo texto, tendrá en cuenta la diferencia entre mayúsculas y minúsculas	"HOLA" -ceq "HOLA"
-ne	Diferente	3 -ne 5
-lt	Menor	3 -lt 5
-le	Menor o igual	5 -le 5 3 -le 5
-gt	Mayor	5 -gt 3
-ge	Mayor o igual	5 -ge 5 5 -ge 3

Operadores de comparación		
Operador de comparación	Significado	Ejemplo ($true)
-like	Es como	"Fermín" -like "Fer*"
-notlike	No es como	"Fermín" -notlike "Fern*"
-contains	Contiene	9,5,2 -contains 5
-notcontains	No contiene	9,5,2 -notcontains 1

- **Operadores lógicos**

Operadores lógicos		
Operador de comparación	Descripción	Ejemplo ($true)
-and	Devuelve $true cuando las dos expresiones que intervienen ofrecen el valor $true	(5 -ge 3) -and (5 -le 5)
-or	Devuelve $true cuando alguna de las expresiones que intervienen ofrecen el valor $true.	(5 -gt 3) -or (5 -lt 5)
-xor	Devuelve $true cuando sólo una de las expresiones que intervienen ofrecen el valor $true.	(5 -gt 3) -xor (8 -le 5)
-not !	Devuelve $true cuando la expresión sobre la que actúa ofrecen el valor $false.	-not (5 -lt 3) !(5 -lt 3)

Operadores lógicos		
Operador de comparación	Descripción	Ejemplo ($true)
-is	Devuelve $true cuando el dato se corresponde con el tipo indicado	"3/11/2009" -is [string]
-isnot	Devuelve $true cuando el dato no se corresponde con el tipo indicado	"3/11/2009" -isnot [DateTime]

- Variables con objetos
    - PowerShell está basado en .NET Framework
        - Nuestros scripts tendrán acceso a todo el modelo de objetos subyacentes en .NET Framework
        - Cada variable es realmente un objeto de .NET Framework
        - Tendrá asociado una serie de métodos
            - Posición: var.indexOf("o")
            - A minúsculas: var.toLower()

Estructuras condicionales en PowerShell

- If

```
[int]$numero = [int] (Read-Host "Escribe un número")
[bool] $positivo = ($numero -ge 0)
if ($positivo) {Write-Host "El número es positivo"}
```

- If else

```
[int]$numero = [int] (Read-Host "Escribe un número")

if ($numero -ge 0) {Write-Host "El número es positivo"}
else {Write-Host "El número es negativo"}
```

- If elseif

```
$respuesta = Read-Host "¿Te gusta el Sushi?"

if (($respuesta -ceq "si") -or
    ($respuesta -ceq "Si") -or
    ($respuesta -ceq "s") -or
    ($respuesta -ceq "S")){
    Write-Host "Tu respuesta ha sido afirmativa"
}
elseif (($respuesta -ceq "no") -or
    ($respuesta -ceq "No") -or
    ($respuesta -ceq "n") -or
    ($respuesta -ceq "N")){
    Write-Host "Tu respuesta ha sido negativa"
}
else{
    Write-Host "No entiendo la respuesta"
}
```

- Switch

```
$valor = 2

switch ($valor){
    1 {Write-Host "Es el uno"}
    2 {Write-Host "Es el dos"}
    3 {Write-Host "Es el tres"}
    4 {Write-Host "Es el cuatro"}

    default {Write-Host "El número no está entre 1 y 4"}
}
```

Estructuras repetitivas en PowerShell

- While

```
$opciones = "si Si sí Sí s no No n "

$respuesta = Read-Host "¿Te gusta el Sushi?"

while ($opciones.IndexOf($respuesta + " ") -lt 0){
    Write-Host "Tu respuesta es errónea"
    $respuesta = Read-Host "¿Te gusta el Sushi?"
}
```

- Do while

```
$opciones = "si Si sí Sí s no No n "

do {
    $respuesta = Read-Host "¿Te gusta el Sushi?"
} while ($opciones.IndexOf($respuesta + " ") -lt 0)
```

- Do until

```
$opciones = "si Si sí Sí s no No n "

do {
    $respuesta = Read-Host "¿Te gusta el Sushi?"

    if ($opciones.IndexOf($respuesta + " ") -lt 0) {
        Write-Host "Tu respuesta es errónea"
    }
} until ($opciones.IndexOf($respuesta + " ") -ge 0)
```

- For

```
for ($i=1; $i -le 5; $i++){
    Write-Host $i
}
```

- Foreach

```
foreach ($elem in 1,2,3,4,5){
    Write-Host $elem
}
```

- Alterar el proceso de las estructuras repetitivas
    - Break

```
for($i=1; $i -le 100; $i++){
    Write-Host $i
    $resp = Read-Host "¿continuamos?"

    if ("no No n ".IndexOf($resp + " ") -ge 0) {
        break
    }
}
```

    - Continue

```
for($i=1; $i -le 10; $i++){
    [int]$num = Read-Host "Escribe un número"

    if ($num %2 -eq 1) { continue }

    Write-Host "El número $num es par"
}
```

LISTADO CMDLETS

Gestión de archivos y directorios

- Get-ChildItem: lista los archivos y directorios (ls) (Get-ChildItem C:\Documents)
- Set-Location: cambia el Directorio actual (cd)
- Copy-Item: copia archivos o directorios (cp) (Copy-Item C:\example.txt – Destination D:)
- Move-Item: mueve archivos o directorios (mv) (Copy-Item C:\example.txt – Destination D:)
- Remove-Item: elimina archivos o directorios (Remove-Item C:\example.txt)
- New-Item: crea un nuevo archivo o directorio (New-Item C:\newfolder – ItemType Directory)

Obtener información y contenido

- Get-Content: lee el contenido de un archivo (cat) (Get-Content C:\example.txt)
- Get-History: muestra el historial de comandos
- Get-Process: lista todos los procesos en ejecución
- Get-Service: lista todos los servicios del sistema

Administración y control del sistema

- Start-Service: inicia un servidor (Start-Service -Name “wuauserv”)
- Stop-Service: detiene un servicio (Stop-Service -Name “wuauserv”)
- Restart-Service: reinicia un servidor (Restart-Service -Name “wuauserv”)
- Start-Process: inicia un proceso o aplicación (Start-Process Notepad)
- Stop-Process: termina un proceso (Stop-Process -Name “notepad”)
- Restart-Computer: reinicia la computadora
- Shutdown-Computer: apaga la computadora

Trabajo con redes

- Test-Connection: prueba la conectividad a un host (Test-Connection google.com)
- Get-NetIPAddress: muestra la información sobre las configuraciones de IP
- Resolve-DnsName: resuelve un nombre a IP (Resolve-DnsName google.com)

Automatización y Scripting

- ForEach-Object: ejecuta acciones en múltiples objetos (Get-Process | FirEach-Object {$_.ProcessName})
- Where-Object: filtra objetos (Get-Process | Where-Object {$_.CPU -gt 10})
- Invoke-Command: ejecuta commandos en máquinas remotas (Invoke-Command -ComputerName Server01 -ScriptBlock {Get-Childre C:\})

Ayuda y descubrimiento

- Get-Help: muestra ayuda (Get-Help Get-Process)
    - -Online: busca la info en la página de Microsoft
- Get-Command: lista cmdlets y funciones
- Show-Command: muestra una interfaz gráfica para cmdlets
- Update-Help: instala los archivos de ayuda de forma local

Variedades y utilidades

- Set-Variable: establece una variable (Set-Variable -Name “x” -Value 10)
- Get-Variable: muestra variables y valores
- ConvertTo-Json: convierte a formato JSON (Get-Process | ConvertTo-Json)