MODULO 2 – Power Shell

PowerShell es un lenguaje de secuencias de comandos y una shell de línea de comandos desarrollada por Microsoft, diseñada para automatizar tareas de administración del sistema y la configuración, entre otras funciones. PowerShell está basado en el .NET Framework y es más avanzado que la tradicional Command Prompt (CMD), lo que lo convierte en una herramienta poderosa para administradores de sistemas y desarrolladores.

Diferencias entre PowerShell y CMD

1. **Lenguaje de Scripting**:
    - **PowerShell**: Utiliza un lenguaje de scripting completo y avanzado basado en .NET, lo que permite realizar tareas complejas como la manipulación de objetos, automatización de procesos, acceso a APIs y más.
    - **CMD**: Es una interfaz de línea de comandos mucho más simple y limitada que solo permite ejecutar comandos y scripts básicos, sin acceso a objetos complejos.
2. **Objetos vs. Texto**:
    - **PowerShell**: Maneja objetos en lugar de texto plano. Esto significa que puedes pasar objetos complejos (como archivos, servicios, procesos) entre comandos sin perder estructura o detalles.
    - **CMD**: Trabaja solo con texto, lo que puede hacer que el procesamiento y la manipulación de datos sea menos eficiente y flexible.
3. **Interoperabilidad**:
    - **PowerShell**: Permite interactuar con herramientas de administración de Windows, como WMI (Windows Management Instrumentation), COM (Component Object Model), y más. También puede interactuar con servicios RESTful y otras tecnologías a través de módulos.
    - **CMD**: Tiene acceso limitado a las herramientas y tecnologías avanzadas de Windows, y no está diseñado para integrar fácilmente otras plataformas.
4. **Comandos**:
    - **PowerShell**: Utiliza cmdlets, que son comandos específicos de PowerShell, como `Get-Process`, `Set-Item`, etc. Estos cmdlets están diseñados para trabajar de manera eficiente con objetos.
    - **CMD**: Usa comandos básicos como `dir`, `copy`, `del`, etc., que están más orientados a manipular archivos y directorios en el sistema de archivos.
5. **Soporte de Pipelining**:
    - **PowerShell**: Ofrece una potente funcionalidad de pipeline (tubería), lo que permite encadenar múltiples cmdlets para realizar tareas complejas con facilidad. Los resultados de un cmdlet pueden ser pasados directamente a otro.
    - **CMD**: También tiene pipelining, pero está limitado a pasar texto entre comandos, lo que lo hace menos flexible.

Características de PowerShell

- **Automatización**: PowerShell se usa principalmente para automatizar tareas repetitivas en sistemas Windows. Puedes escribir scripts para crear, configurar y administrar usuarios, archivos, redes, servicios, y más.
- **Extensibilidad**: Tiene un sistema de módulos que permite ampliar sus capacidades, permitiendo la integración con nuevas herramientas o servicios.
- **Soporte para Remotización**: PowerShell permite ejecutar comandos y scripts en sistemas remotos usando PowerShell Remoting.
- **Manejo de Objetos**: A diferencia de CMD, PowerShell trata con objetos, lo que facilita la manipulación de datos complejos y la exportación a formatos como CSV, JSON o XML.
- **Compatibilidad Multiplataforma**: Con PowerShell Core (basado en .NET Core), PowerShell ahora es compatible con Linux y macOS además de Windows.

Arquitectura de PowerShell

PowerShell está basado en .NET, y su arquitectura consta de varios componentes clave:

1. **Motor de ejecución (Execution Engine)**: Es responsable de interpretar y ejecutar los scripts de PowerShell. Controla el flujo de ejecución, maneja los comandos y gestiona el contexto de ejecución.
2. **Cmdlets**: Son comandos específicos de PowerShell que realizan tareas básicas o complejas. Estos cmdlets son pequeñas aplicaciones que realizan una única función dentro del entorno de PowerShell.
3. **Pipelines**: PowerShell permite que los resultados de un comando se pasen a otros comandos mediante un pipeline. Los datos no se transfieren como texto, sino como objetos.
4. **Hosting API**: Esta interfaz permite que otros programas o aplicaciones inviten o embeban el motor de PowerShell para que puedan ejecutar comandos de PowerShell dentro de sus propios procesos.
5. **Integración con .NET Framework**: PowerShell tiene acceso a todas las bibliotecas y clases del .NET Framework, lo que le da un alto nivel de flexibilidad y potencia para la administración del sistema.

Cmdlet	Descripción
`Get-Process`	Obtiene información sobre los procesos en ejecución en el sistema.
`Get-Service`	Muestra información sobre los servicios de Windows en el sistema.
`Get-EventLog`	Recupera los registros de eventos de un equipo local o remoto.
`Get-WmiObject`	Extrae información del sistema utilizando WMI, como procesos, discos, adaptadores de red, etc.
`Get-WmiObject -Class`	Recupera información de una clase WMI específica, como configuraciones de hardware, software, etc.
`Get-WmiObject -Query`	Ejecuta una consulta WMI personalizada para obtener información específica.
`Get-ComputerInfo`	Proporciona información detallada sobre el hardware y

Cmdlet	Descripción
	software del sistema.
`Get-PhysicalDisk`	Obtiene información sobre los discos físicos del sistema.
`Get-NetAdapter`	Muestra información sobre los adaptadores de red del sistema.
`Get-WmiObject -Class Win32_Processor`	Recupera información sobre los procesadores del sistema utilizando WMI.
`Get-WmiObject -Class Win32_OperatingSystem`	Extrae información sobre el sistema operativo instalado.
`Get-WmiObject -Class Win32_NetworkAdapterConfiguration`	Obtiene detalles sobre la configuración de los adaptadores de red.
`Get-WmiObject -Class Win32_LogicalDisk`	Muestra información sobre los discos lógicos del sistema, como las particiones y su uso.

Ejemplos detallados:

1. **Obtener información del sistema operativo**:

   ```
   Get-WmiObject -Class Win32_OperatingSystem
   ```

   Este comando extrae información como la versión del sistema operativo, nombre del producto, arquitectura, etc.

2. **Obtener detalles de los discos lógicos**:

   ```
   Get-WmiObject -Class Win32_LogicalDisk
   ```

   Devuelve información sobre los discos lógicos, como el tamaño, el espacio libre y el tipo de archivo.

3. **Obtener los servicios en ejecución**:

   ```
   Get-Service
   ```

   Muestra todos los servicios activos en el sistema.

4. **Consultar procesos específicos**:

   ```
   Get-Process -Name "notepad"
   ```

   Este comando devuelve información sobre el proceso de Notepad si está en ejecución.

5. **Consultar la configuración de los adaptadores de red**:

   ```
   Get-WmiObject -Class Win32_NetworkAdapterConfiguration
   ```

   Muestra detalles sobre los adaptadores de red, como su configuración de IP.