**Email:** meghanabadgujar07@gmail.com

**Name:** Meghana Samadhan Badgujar  
**Batch Name:** WiproNGA\_DWS\_B5\_25VID2550

**User ID:** 34737

**Batch ID:** 25VID2550

**Date:** 8th August 2025

**Assignment Topic:**

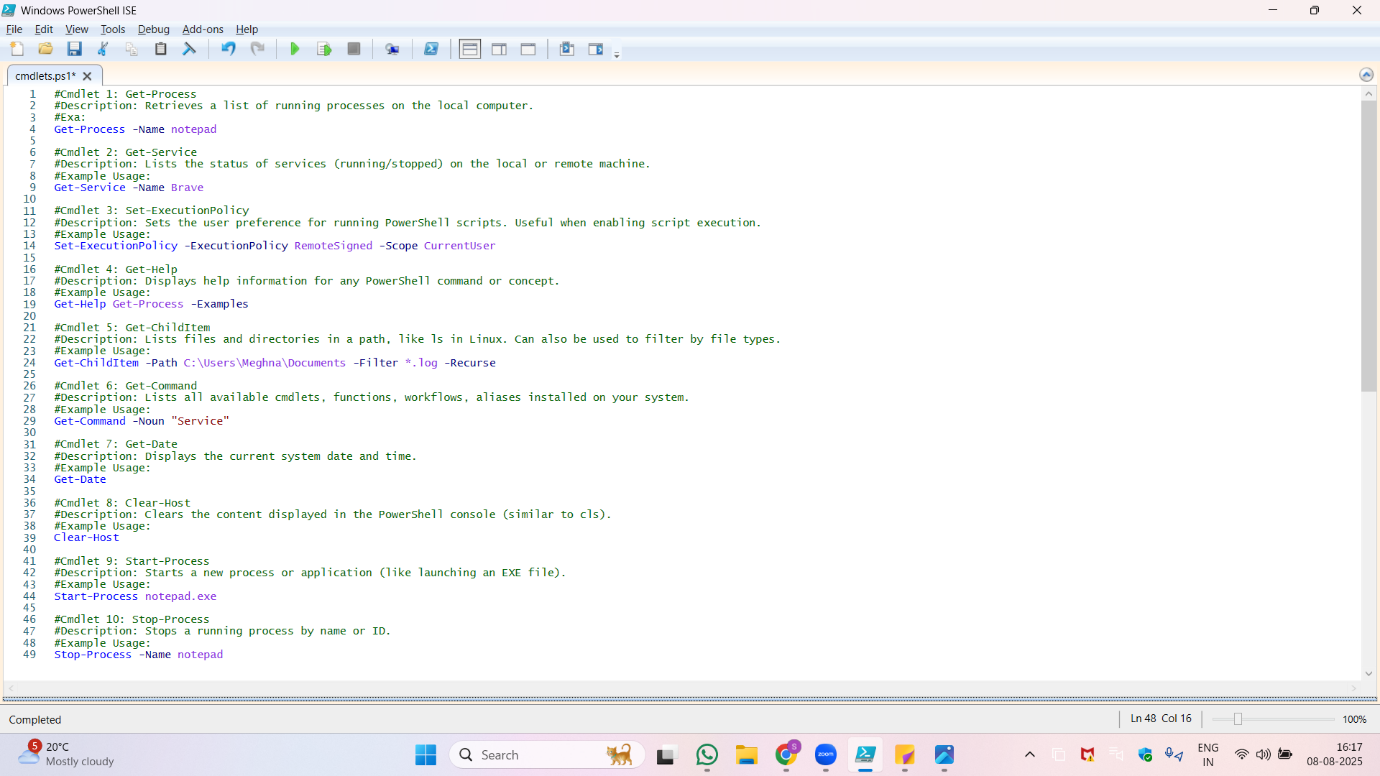
* **PowerShell Fundamentals**

**1. Introduction to Cmdlets:**

**Cmdlets** (pronounced ***command-lets***) are the building blocks of PowerShell scripting. They are small, task-specific commands that follow the Verb-Noun naming pattern (e.g., Get-Process, Set-Date) and are used to automate system administration tasks.

* **Key Characteristics:**
* **Structured Naming**: Verb-Noun format (e.g., Get-Service)
* **Pipeline-Compatible**: Accept input and pass output through the pipeline
* **Object-Oriented**: Work with .NET objects instead of plain text
* **Built-in & Custom**: You can use predefined cmdlets or write your own
* **Non-Executable**: Cmdlets are not standalone .exe files—they are part of PowerShell
* **Real-World Impact:**

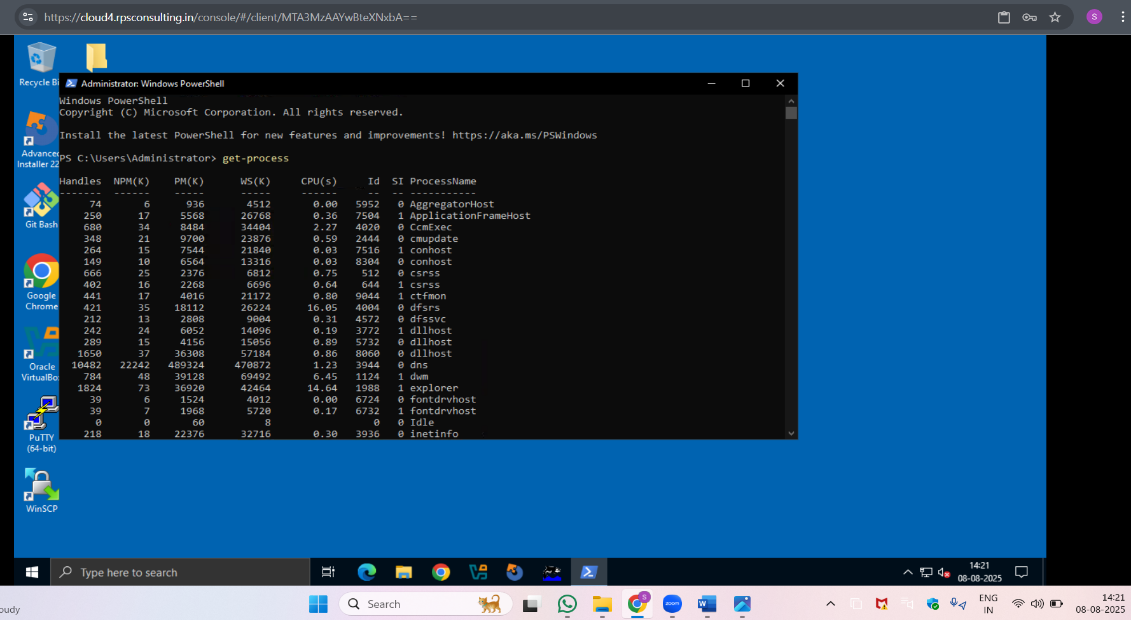
PowerShell cmdlets streamline server management, user provisioning, cloud automation, and routine tasks across industries (e.g., inventory checks, VM management).

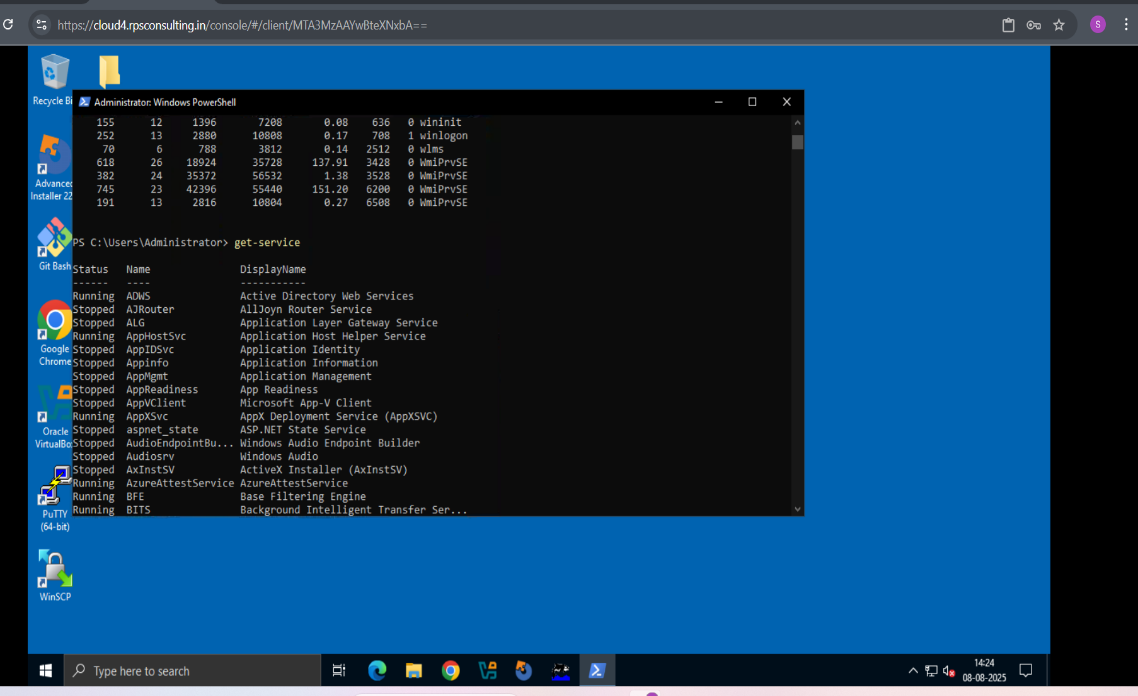


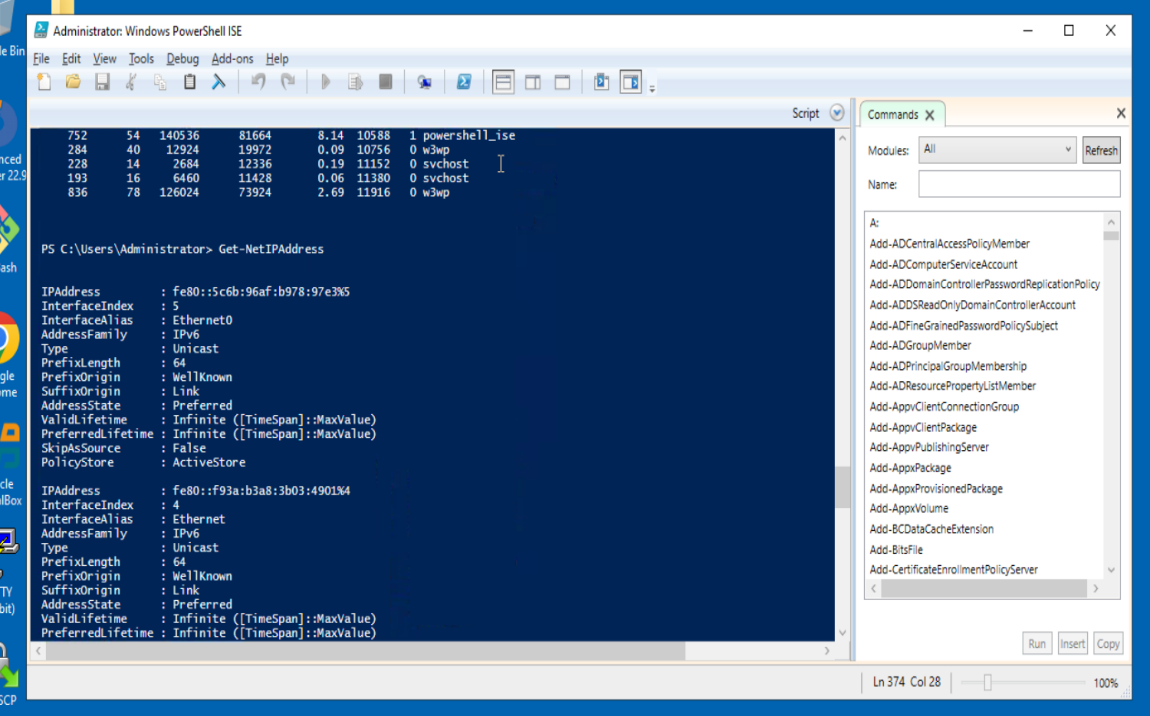
**2. Key Cmdlets to Know**

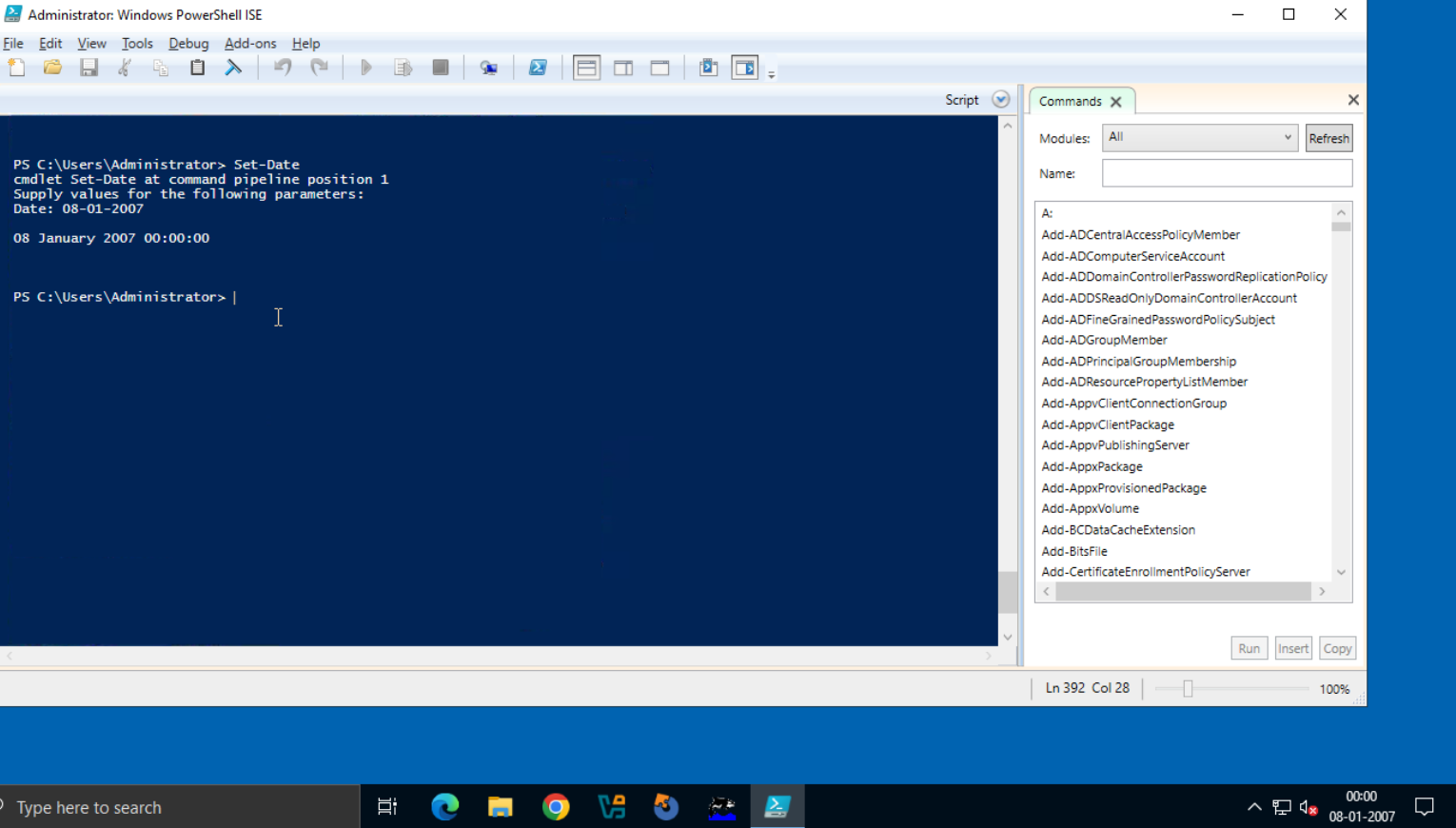
Here are some commonly used cmdlets:

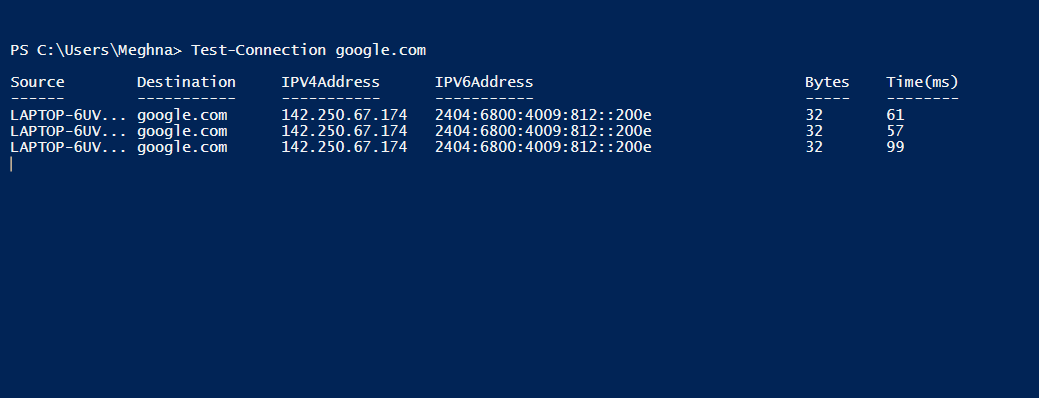
|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Example Usage** |
| Get-Process | Lists running processes | Get-Process |
| Get-Service | Retrieves service statuses | Get-Service -Name wuauserv |
| Stop-Service | Stops a service | Stop-Service -Name spooler |
| Get-ChildItem | Lists files/folders (like dir) | Get-ChildItem C:\Temp |
| Get-EventLog | Displays recent event logs | Get-EventLog -LogName System |
| Set-ExecutionPolicy | Sets script execution policy | Set-ExecutionPolicy RemoteSigned |











**3. The PowerShell Pipeline**

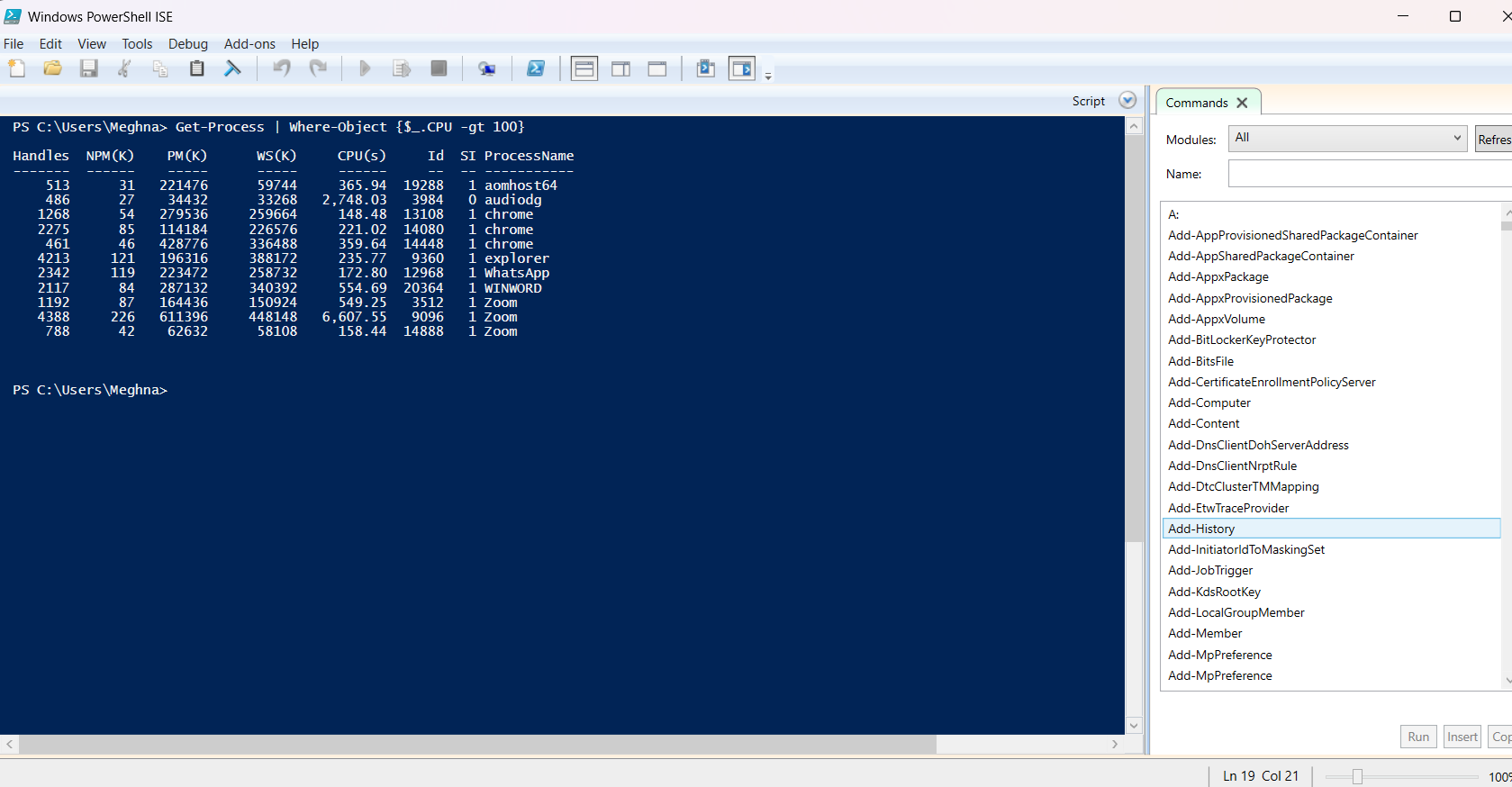
The **pipeline (|)** in PowerShell allows **chaining multiple cmdlets** together. The output of one cmdlet becomes the input for the next.

**Example:**

Get-Process | Where-Object {$\_.CPU -gt 100}

This command:

* Retrieves all running processes
* Filters those using **more than 100 units of CPU**



* **Pipeline Advantage:**
* Saves time and reduces manual effort
* Enables complex operations with simple syntax
* Makes scripts modular and readable

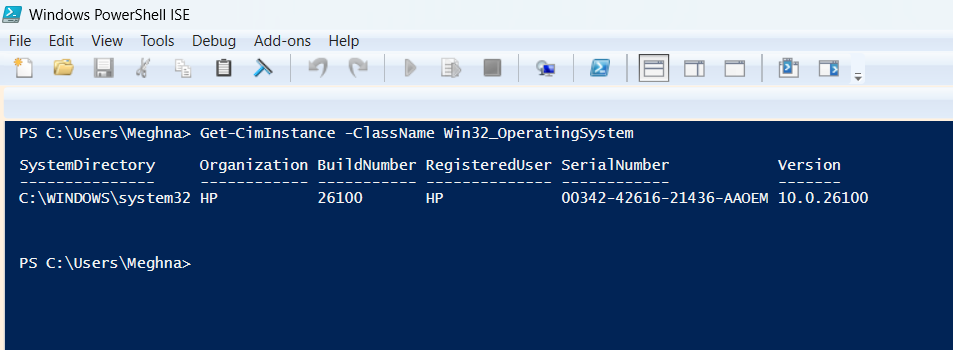
**4. WMI & PowerShell Integration**

**WMI (Windows Management Instrumentation)** provides low-level access to system information. PowerShell interacts with WMI to fetch and manage system data like hardware, OS details, services, and more.

* **Cmdlets to Use:**
* Get-WmiObject (older method)
* Get-CimInstance (modern, preferred method)
* **Example:**

Get-CimInstance -ClassName Win32\_OperatingSystem

- This returns details about the OS like version, install date, memory, etc.

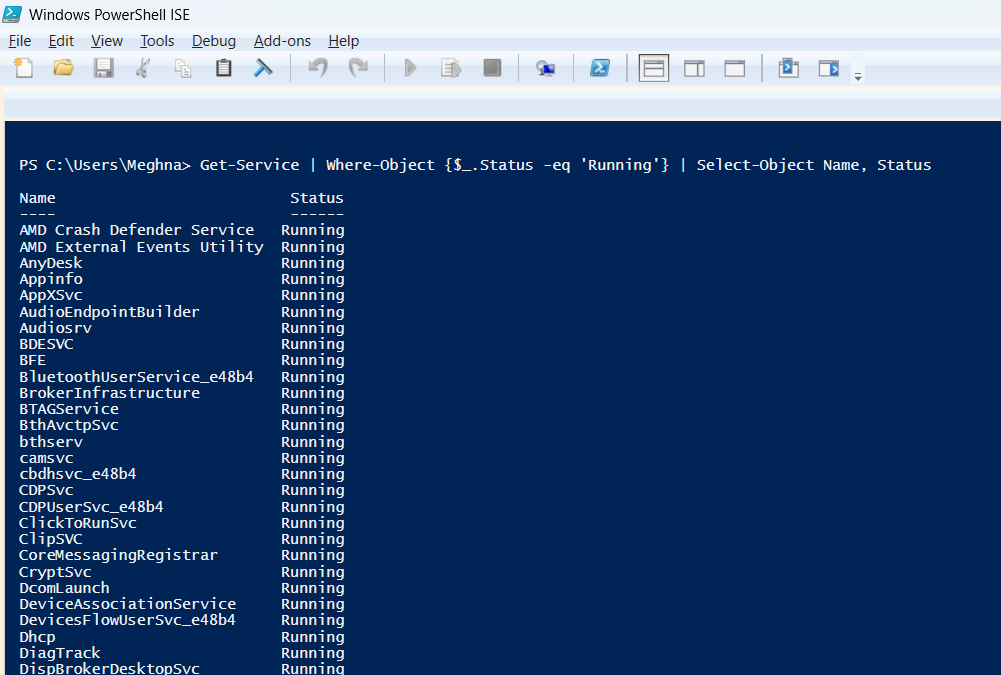


**5. Pipeline Filtering & Operators**

PowerShell offers filtering and logical operators to refine results in scripts.

* **Common Filters:**
* Where-Object – Filter items based on conditions
* Select-Object – Select specific properties
* **Logical Operators:**
* -eq (equals), -ne (not equal)
* -gt, -lt (greater than, less than)
* -like, -match (pattern matching)
* -and, -or, -not (logical combinations)
* **Example:**

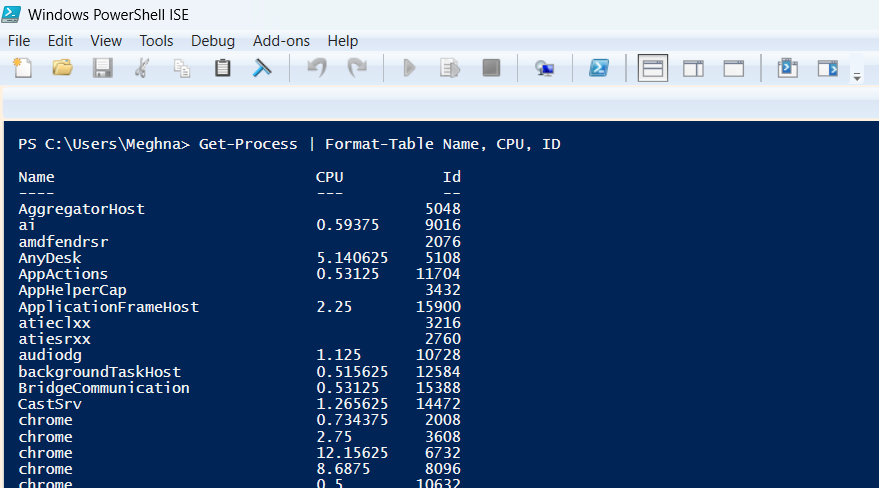
Get-Service | Where-Object {$\_.Status -eq 'Running'} | Select-Object Name, Status



**6. Input, Output & Formatting**

* **Input:**
* **User Input**: via Read-Host
* **File Input**: via Get-Content
* **Output:**
* PowerShell outputs objects by default
* You can **redirect output** to:
  + Files (Out-File)
  + CSVs (Export-Csv)
  + JSON (ConvertTo-Json)
* **Formatting:**
* Format-Table – tabular view
* Format-List – detailed list view
* **Example:**

Get-Process | Format-Table Name, CPU, ID



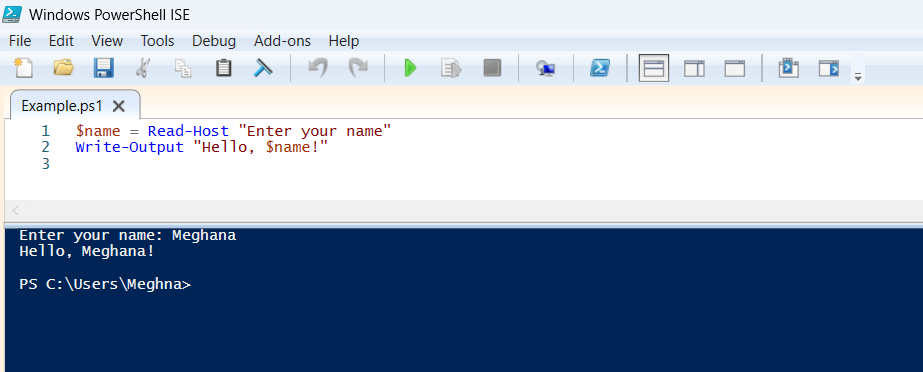
**7. Scripting Overview**

PowerShell scripts are saved with the .ps1 extension. A script can include variables, conditions, loops, and functions.

* **Sample Script:**

$name = Read-Host "Enter your name"

Write-Output "Hello, $name!"

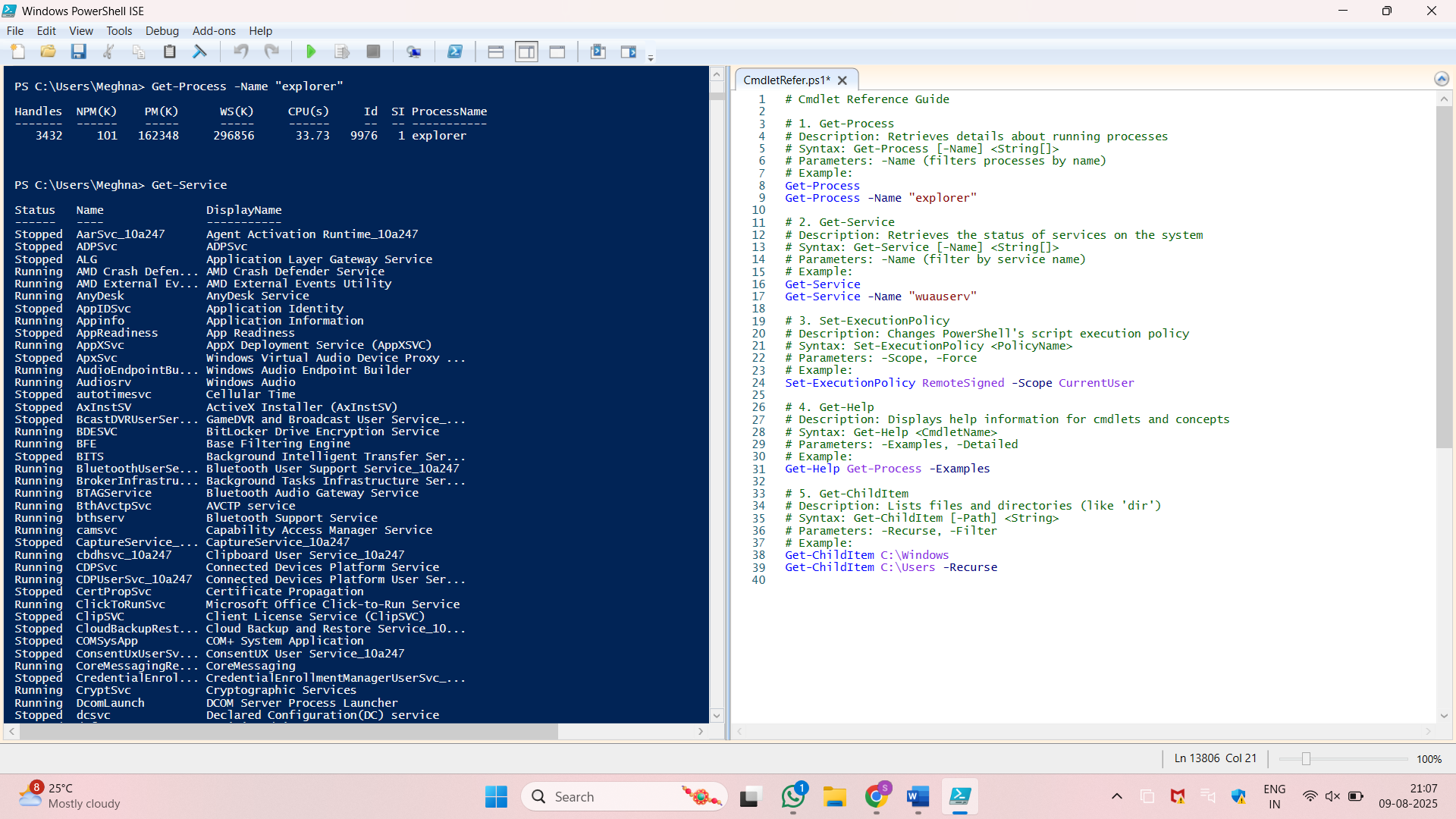


* **Scripting Benefits:**
* Automate daily tasks
* Schedule scripts using Task Scheduler
* Reuse code across machines and teams
* **Project 1: Exploring Cmdlet Syntax**

**Objective:**  
**To understand the structure, purpose, and parameters of commonly used PowerShell cmdlets by creating a simple reference guide.**

* **Steps to Perform:**

1. **Open PowerShell ISE or Visual Studio Code**
   * PowerShell ISE (Integrated Scripting Environment) comes pre-installed on most Windows systems.
   * You can also use VS Code with the PowerShell extension for a better experience.
2. **Create a New Script File**
   * Name the file: CmdletRefer.ps1
3. **For this project, we’ll explore:**
   * Get-Process
   * Get-Service
   * Set-ExecutionPolicy
   * Get-Help
   * Get-ChildItem



* **Project 2: Automate a Task with a Cmdlet Script**

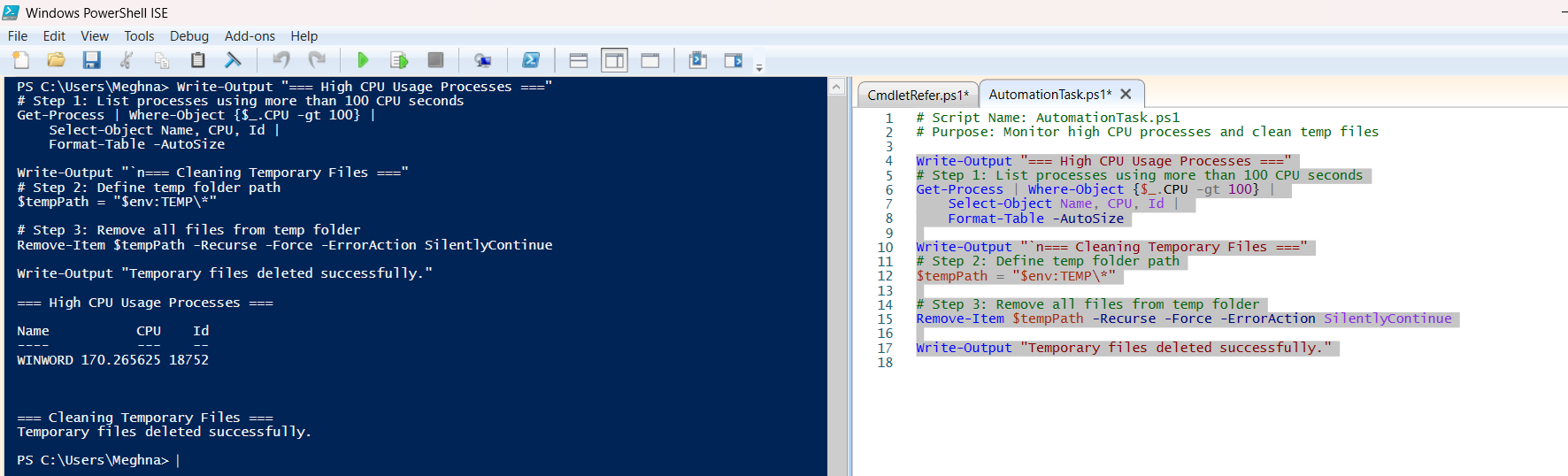
**Objective:**  
To use PowerShell cmdlets in combination to automate a routine system task, demonstrating the ability to chain commands and create an efficient script.

**Task Chosen:**

**Monitoring High CPU Usage Processes and Cleaning Temporary Files**

**Steps to Perform:**

1. **Identify the Routine Task**
   * System administrators often need to monitor resource usage and clear temporary files to improve performance.
   * We’ll create a script that:
     1. Finds processes using high CPU.
     2. Deletes temporary files from the user’s temp directory.
2. **Open PowerShell ISE or VS Code**
   * Create a new script file named: AutomationTask.ps1
3. **Write the Script:**



**How It Works:**

* **Get-Process** retrieves all running processes.
* **Where-Object {$\_.CPU -gt 100}** filters processes with CPU usage above 100 seconds.
* **Select-Object** and **Format-Table** display the results neatly.
* **Remove-Item** deletes all temporary files in the current user’s temp directory.
* **-Recurse -Force** ensures all files and folders are deleted without confirmation prompts.
* **Project 3: Create a PowerShell Cmdlet Cheat Sheet**

**Objective:**  
To compile a quick-reference guide of commonly used PowerShell cmdlets categorized by functionality, along with brief descriptions and example usages.

**1. Basic Cmdlets – General Information and Help**

|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Example Usage** |
| Get-Help | Shows help information about commands | Get-Help Get-Process |
| Get-Command | Lists all available commands | Get-Command |
| Get-Content | Displays the content of a file | Get-Content C:\example.txt |
| Set-Content | Writes or replaces content in a file | Set-Content C:\example.txt "Hi" |
| Select-String | Searches text/files for specific patterns | Select-String -Pattern "error" log.txt |
| Where-Object | Filters objects based on a condition | Get-Process | Where-Object {$\_.CPU -gt 100} |
| Sort-Object | Sorts objects by property | Get-Process | Sort-Object CPU |
| Out-File | Sends output to a file | Get-Process | Out-File processes.txt |
| Get-Variable | Gets variables in the current session | Get-Variable |
| Set-Variable | Creates or changes variables | Set-Variable -Name MyVar -Value 1 |
| Clear-Host | Clears the screen | Clear-Host |
| Measure-Object | Measures properties of objects | Get-ChildItem | Measure-Object |
| Write-Output | Sends output to the pipeline or console | Write-Output "Hello, world!" |
| Write-Host | Displays colored output on console | Write-Host "Success" -ForegroundColor Green |
| Get-History | Shows command history | Get-History |
| Invoke-History | Runs a command from history | Invoke-History 3 |
| Get-Alias | Gets command aliases | Get-Alias |
| Set-Alias | Creates an alias for commands | Set-Alias ll Get-ChildItem |
| ForEach-Object | Performs an operation on each item in a collection | Get-Process | ForEach-Object { $\_.Name } |
| Get-Location | Shows the current directory | Get-Location |

**2. File System Cmdlets – File and Folder Operations**

|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Example Usage** |
| New-Item | Creates new files or directories | New-Item -Path "C:\Test" -ItemType Directory |
| Copy-Item | Copies files or folders | Copy-Item C:\file.txt D:\Backup\file.txt |
| Move-Item | Moves files or folders | Move-Item C:\file.txt D:\Archive\file.txt |
| Remove-Item | Deletes files or folders | Remove-Item C:\Temp\\* -Recurse |
| Rename-Item | Renames a file or folder | Rename-Item "old.txt" "new.txt" |
| Get-ChildItem | Lists files and folders in a directory | Get-ChildItem C:\Temp |
| Test-Path | Checks if a path exists | Test-Path C:\Temp |
| Get-Item | Gets a specific file or folder | Get-Item C:\file.txt |
| Set-Item | Changes the value or content of an item | Set-Item -Path Env:Path -Value $newPath |
| Join-Path | Combines strings into a path | Join-Path C:\Temp "file.txt" |
| Split-Path | Gets part of a path | Split-Path C:\Temp\file.txt -Parent |
| Clear-Content | Clears the content of a file | Clear-Content C:\log.txt |
| Get-ItemProperty | Gets properties of a file or folder | Get-ItemProperty C:\file.txt |
| Set-ItemProperty | Sets properties of a file or folder | Set-ItemProperty C:\file.txt -Name IsReadOnly -Value $true |
| Copy-ItemProperty | Copies properties from one item to another | Copy-ItemProperty C:\file1.txt C:\file2.txt |
| Get-PSDrive | Lists drives available (file system, registry) | Get-PSDrive |
| New-PSDrive | Creates a new drive | New-PSDrive -Name X -PSProvider FileSystem -Root C:\ |
| Remove-ItemProperty | Deletes property of an item | Remove-ItemProperty -Path C:\file.txt -Name IsReadOnly |
| Get-ChildItem -Recurse | Lists all items in directory and subdirectories | Get-ChildItem -Recurse C:\Temp |
| Format-Table | Formats output as a table | Get-ChildItem | Format-Table |

**3. Network Cmdlets – Network Testing and Info Retrieval**

|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Example Usage** |
| Test-Connection | Checks network connectivity (like ping) | Test-Connection google.com |
| Get-NetIPAddress | Shows IP address information | Get-NetIPAddress |
| Get-NetIPConfiguration | Displays IP configuration details | Get-NetIPConfiguration |
| Get-NetAdapter | Gets network adapter information | Get-NetAdapter |
| Enable-NetAdapter | Enables a network adapter | Enable-NetAdapter -Name "Ethernet" |
| Disable-NetAdapter | Disables a network adapter | Disable-NetAdapter -Name "Wi-Fi" |
| Set-DnsClientServerAddress | Sets DNS servers for a network adapter | Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses ("8.8.8.8") |
| Get-DnsClientServerAddress | Gets DNS server addresses for adapters | Get-DnsClientServerAddress |
| Test-NetConnection | Tests network connection with detailed info | Test-NetConnection google.com -InformationLevel Detailed |
| Resolve-DnsName | Performs DNS query | Resolve-DnsName google.com |
| Get-NetRoute | Gets IP route information | Get-NetRoute |
| New-NetIPAddress | Adds a new IP address to an interface | New-NetIPAddress -InterfaceAlias "Ethernet" -IPAddress 192.168.1.10 -PrefixLength 24 |
| Remove-NetIPAddress | Removes IP address from interface | Remove-NetIPAddress -IPAddress 192.168.1.10 |
| Get-NetFirewallRule | Lists firewall rules | Get-NetFirewallRule |
| Enable-NetFirewallRule | Enables a firewall rule | Enable-NetFirewallRule -Name "FPS-Rule" |
| Disable-NetFirewallRule | Disables a firewall rule | Disable-NetFirewallRule -Name "FPS-Rule" |
| Show-NetConnectionProfile | Displays network connection profile | Get-NetConnectionProfile |
| Get-NetIPAddress -AddressFamily IPv6 | Shows IPv6 addresses | Get-NetIPAddress -AddressFamily IPv6 |
| Restart-NetAdapter | Restarts a network adapter | Restart-NetAdapter -Name "Wi-Fi" |
| Get-NetNeighbor | Displays neighbor cache entries | Get-NetNeighbor |

**4. Process & Service Cmdlets – Managing Running Processes and Services**

|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Example** |
| Get-Process | Lists all running processes | Get-Process chrome |
| Stop-Process | Stops a running process | Stop-Process -Name notepad |
| Start-Process | Starts a new process | Start-Process notepad.exe |
| Get-Service | Lists all services on the system | Get-Service |
| Start-Service | Starts a service | Start-Service spooler |
| Stop-Service | Stops a service | Stop-Service spooler |
| Restart-Service | Restarts a service | Restart-Service spooler |
| Set-Service | Changes a service’s properties (e.g., startup type) | Set-Service -Name spooler -StartupType Automatic |

**5. System Information Cmdlets – System and OS Details**

|  |  |  |
| --- | --- | --- |
| **Cmdlet** | **Description** | **Example** |
| Get-ComputerInfo | Displays detailed system info | Get-ComputerInfo |
| Get-Date | Displays or sets the system date/time | Get-Date |
| Set-Date | Sets the system date/time | Set-Date -Date "2025-08-09 14:00:00" |
| Get-EventLog | Retrieves Windows Event Logs | Get-EventLog -LogName System -Newest 10 |
| Get-WmiObject | Retrieves system info via WMI (legacy) | Get-WmiObject Win32\_OperatingSystem |
| Get-CimInstance | Retrieves system info (modern alternative to WMI) | Get-CimInstance Win32\_Processor |
| Get-HotFix | Lists installed Windows updates | |  | | --- | | Get-HotFix |  |  | | --- | |  | |
| Get-Uptime *(custom alias)* | Shows system uptime | `Get-CimInstance Win32\_OperatingSystem |