# Batch Information:

* **Batch Start Date:** 2025-08-04
* **Batch Name:** WiproNGA\_DWS\_B5\_25VID2550
* **First Name:** Aayush
* **Last Name:** Kumar
* **User ID:** 34758
* **Batch ID:** 25VID2550
* **Email ID:** theaayushtagore@gmail.com

**Assignment**

**Introduction to PowerShell and Cmdlets**

PowerShell is a powerful task automation and configuration management framework from Microsoft. It’s widely used by system administrators, DevOps engineers, and developers to manage computers, automate repetitive tasks, and handle system configurations. Unlike the traditional Command Prompt (CMD), PowerShell works with objects rather than plain text, which makes it far more powerful and versatile.

At the heart of PowerShell are cmdlets (short for command-lets) — small, single-function commands that do one job very well.

**What are Cmdlets?**

A cmdlet is a lightweight, single-purpose command that is built into PowerShell. These commands generally perform a single, specific task, such as retrieving a list of processes, copying a file, or querying system information.

Key characteristics of cmdlets:

1. Built into PowerShell – They are part of the PowerShell environment, not separate executable files.
2. Named in Verb-Noun format – Every cmdlet name follows this standard, which makes it easy to remember.
   * Examples:
     + Get-Process – retrieve processes running on your machine
     + Start-Service – start a stopped service
     + Stop-Process – terminate a running process
3. Work with objects, not text – Unlike traditional shells, PowerShell cmdlets pass structured objects from one command to another, allowing rich data handling.
4. Composable using pipelines – Cmdlets can be chained together with the pipeline operator (|) to perform complex tasks by processing data step-by-step.
5. Accept and produce output – They take input from parameters or pipelines, process the data, and pass output to the next cmdlet if required.
6. Extensible – You can create your own cmdlets using PowerShell scripts or .NET languages.

**PowerShell Cmdlet Naming Convention**

Every cmdlet name follows the Verb-Noun pattern:

* Verb: Specifies the action (Get, Set, Remove, Start, Stop, etc.)
* Noun: Specifies the object or resource type (Process, Service, Item, etc.)

Examples:

* Get-Date → Retrieves the current date and time
* Set-Date → Changes the system date/time
* Get-EventLog → Reads event log entries
* Clear-Content → Deletes the content of a file (but not the file itself)

**The PowerShell Pipeline**

One of PowerShell’s most powerful features is the pipeline, which is a mechanism for chaining multiple cmdlets so the output of one command becomes the input of the next.

How it works:

* Symbol: |
* Data passed along the pipeline is structured objects, not plain strings.
* This enables very advanced processing without having to manually parse output.

Example:

powershell

Get-Process | Sort-Object CPU

* Get-Process retrieves all running processes.
* The output (a list of Process objects) is sent into Sort-Object CPU.
* Sort-Object sorts them based on CPU usage.

**Advantages of the Pipeline:**

1. Helps build powerful scripts step-by-step.
2. Keeps commands clean and readable.
3. Avoids extra intermediary files or temporary variables.
4. Allows for efficient memory usage (streams data instead of holding everything in memory).

**Popular Cmdlets**

System Information

* Get-ComputerInfo → Displays detailed system specs
* Get-Process → Lists running processes
* Get-Service → Shows system services and their states
* Get-EventLog → Retrieves Windows event logs

File & Folder Management

* Get-Item → Retrieve details of a file or folder
* Get-ChildItem → List files and subfolders in a directory
* Copy-Item → Copy files or folders
* Move-Item → Move files or folders
* Remove-Item → Delete files or folders
* Rename-Item → Rename files or folders

**Text & Content Management**

* Get-Content → Read file contents
* Set-Content → Write/overwrite file contents
* Clear-Content → Remove content from files without deleting them
* Select-String → Search inside files (like grep)

**Active Directory Management *(requires AD module)***

* Get-ADUser → Retrieve details of an Active Directory user
* New-ADUser → Create a new AD user account

Filtering and Sorting

* Where-Object → Filter based on conditions
* Sort-Object → Sort by specific properties

**Windows Management Instrumentation (WMI) in PowerShell**

**What is WMI?**

Windows Management Instrumentation is a framework that lets you access detailed system data and perform management tasks remotely or locally.

PowerShell can query WMI classes using commands like:

powershell

Get-WmiObject -Class Win32\_OperatingSystem

* Shows information about your Windows OS (version, build number, architecture, etc.)

Benefits of using WMI in PowerShell:

1. Read detailed hardware information (CPU, RAM, Disk, Network Adapters).
2. Check disk space remotely.
3. Manage services and processes on remote computers.

**Filtering & Comparison Operators in PowerShell**

When you only want specific data from a large dataset, you use filters.

The most common cmdlet for filtering is Where-Object.

Basic syntax:

powershell

<Command> | Where-Object { $\_.Property -Operator Value }

* $\_ is the current object in the pipeline.
* Property is the object’s property.
* Operator is the comparison type.
* Value is the expected match.

**Common comparison operators:**

* -eq → equal to  
  Example: Status -eq "Running"
* -ne → not equal to  
  Example: Name -ne "Admin"
* -gt → greater than  
  Example: CPU -gt 100
* -lt → less than  
  Example: FileSize -lt 1MB
* -ge → greater than or equal to  
  Example: Age -ge 18
* -le → less than or equal to  
  Example: Score -le 50
* -like → wildcard match  
  Example: Name -like "Test\*"
* -match → regex match  
  Example: Name -match "^A"

**PowerShell Scripting Overview**

Instead of typing commands interactively, PowerShell allows you to save commands in a .ps1 script file and run them anytime.

**Advantages of scripting:**

* Saves time for repetitive work
* Reduces errors
* Automates complex workflows
* Useful for backups, monitoring, cleanup, and more

Basic Steps to Create a Script:

1. Open PowerShell ISE or a code editor.
2. Save the file with .ps1 extension.
3. Write your cmdlet sequence.
4. Run with F5 or:

powershell

.\scriptname.ps1

**Mini Projects for Practice**

Project 1: Exploring Cmdlet Syntax

* Use cmdlets like Get-Process, Get-Service, Set-ExecutionPolicy, Get-Help, and Get-ChildItem.
* Save these commands into a .ps1 file.
* Run the script and view the output.

Project 2: Automating Tasks

* Pick two tasks:
  1. Show processes using CPU > 100.
  2. Clean %temp% folder.
* Implement using:

powershell

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

Remove-Item "$env:TEMP\\*" -Recurse -Force

Project 3: Cmdlet Cheat Sheet

Create your own personal reference of cmdlets grouped into categories.

**Final Thoughts**

* Cmdlets are the building blocks of PowerShell.
* Pipelines empower you to combine commands for more complex workflows.
* Filtering lets you extract just the data you need.
* WMI unlocks deeper system control.
* Scripting turns manual tasks into fully automated processes.