NAME : B S SUDHEENDRA

EMAIL ID : bssudheendra2006@gmail.com

TOPIC - 1 : Introducing to Cmdlets

Cmdlets are the basic, lightweight commands in the Windows PowerShell environment. They are designed to work in conjunction with the pipeline, processing objects as input and outputting objects. Cmdlets are named using a "Verb-Noun" format, where the verb represents the action and the noun represents the object of that action.

Key Features of Cmdlets:

- Cmdlets can pass data to one another using the pipeline.
- PowerShell comes with hundreds of built-in cmdlets.
- You can create custom cmdlets PowerShell or .NET.
- All cmdlets follow the Verb-Noun naming for readability and consistency.

How cmdlets works:

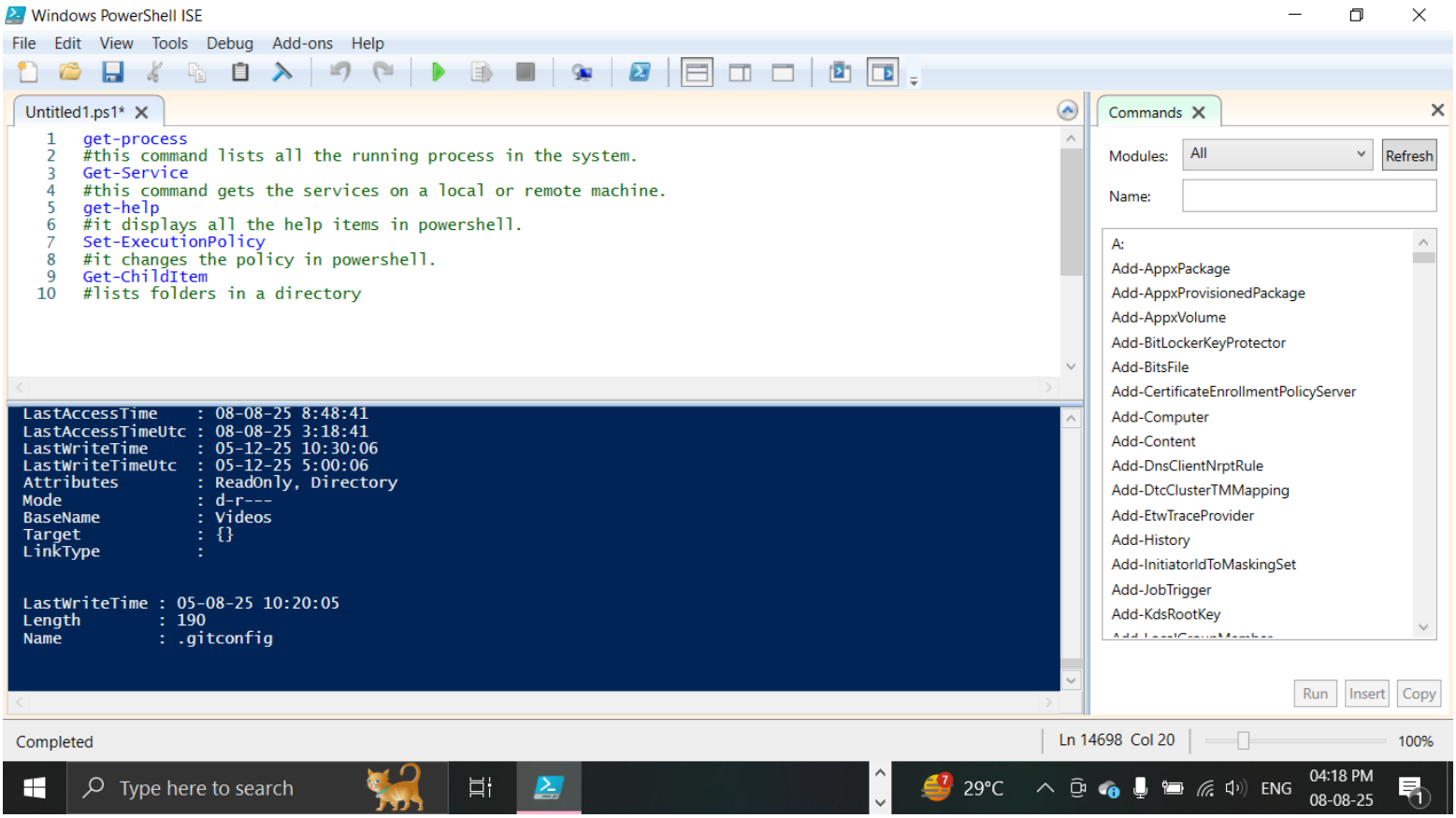
- Input: Takes user input via parameters or pipeline.
  Example- Get-Service -Name spooler.
- Processing: Executes logic using .NET
  Example- Internally uses ProcessRecord ( ).
- Output: Returns .NET objects.
  Example- Results can be piped or saved.

TOPIC - 2 : Key Cmdlets

System information :

- Get-Process : It lists all the running processes in the system.
- Get-Service : It gets all the services on a local or remote machine.

- Get-Help : It displays all the help items in powershell.
- Set-ExecutionPolicy : It changes the policy in powershell.
- Get-ChildItem : Lists folders in a directory.



Here are the some other Key **CMDlets**

- Get-Eventlog
- Start-Services
- Stop-Services
- Set-Location
- Set-Content
- Copy-Item
- Move-Item
- Set-Content
- Get-Help
- Get-ADUser
- Select-ADUser

- Where-Object
- Set-Object
- Set-Date

TOPIC - 3 : WMI AND POWERSHELL

WMI

WMI stands for Windows Management Instrumentation, WMI is a Microsoft technology.

WMI & PowerShell

It is used to access and manage the internal datd of windows-based systems.Think of it as a bridge to get detailed system info like CPU, memory, disk, OS, drivers, etc…

WMI uses a database of classes like:

- Win32_OperatingSystem
- Win32_LogicalDisk
- Win32_Service
- Win32_Process

POWERSHELL

Understanding PowerShell cmdlets is a critical step toward mastering automation and scripting in Windows environments. By diving into hands-on projects, you'll engage more actively with PowerShell, which can greatly improve your comprehension and retention of cmdlet structures and functions. Practical exercises allow you to learn from real-world scenarios, deepen your understanding, and build your confidence.

TOPIC - 4 : Pipeline Filtering & Operators

One of the most powerful features of PowerShell is the ability to use Cmdlets in pipelines. This means you can chain multiple Cmdlets together, passing the output of one as the input to the next.

Example of a Pipeline

Get-Process | Where-Object { $_.CPU -gt 100 } | Sort-Object -Property CPU -Descending

TOPIC - 5 : Input, Output & Formatting

Input :

Input refers to the data or parameters you pass into a cmdlet, script, or function.

Example-Get-Srevice -Name

Output :

Output is the result produced by a cmdlet, function, or script. It could be:

- A text result
- An object
- A list or table

Formatting :

Formatting control how output is displayed. Powershell automatically formats output as:

- Table
- List
- Wide

TOPIC - 5 : Scripting Overview

PowerShell scripting is the process of writing and executing a series as powershell commands in a file to automate tasks.

A PowerShell scripting is a text file containing a sequence of cmdlets, functions, logic and comments. It helps in automating repetitive administrative tasks.

TOPIC - 6 : The PowerShell Pipeline

The PowerShell pipeline (|) allows you to pass the ***output*** of one cmdlet as the ***input*** to another. It is a powerful feature that supports automations and scripting by ***chaining*** commands together.

Sample example- **Get-process | Where -object { $_.CPU -gt 100 } | Sort-object CPU -Descending**