PowerShell Cmdlets and Scripting Overview

# Introducing to Cmdlets

Cmdlets (pronounced "command-lets") are lightweight PowerShell commands used in the PowerShell environment.   
They follow a Verb-Noun naming convention such as Get-Process, Set-Item, or Remove-Item. Cmdlets are instances of .NET classes and can perform actions like file operations, service control, and registry modifications.

# The PowerShell Pipeline

The pipeline (|) in PowerShell allows the output of one cmdlet to be used as the input to another cmdlet.   
This enables chaining of commands and efficient processing of objects. For example: Get-Process | Where-Object {$\_.CPU -gt 100}.

# Key Cmdlets

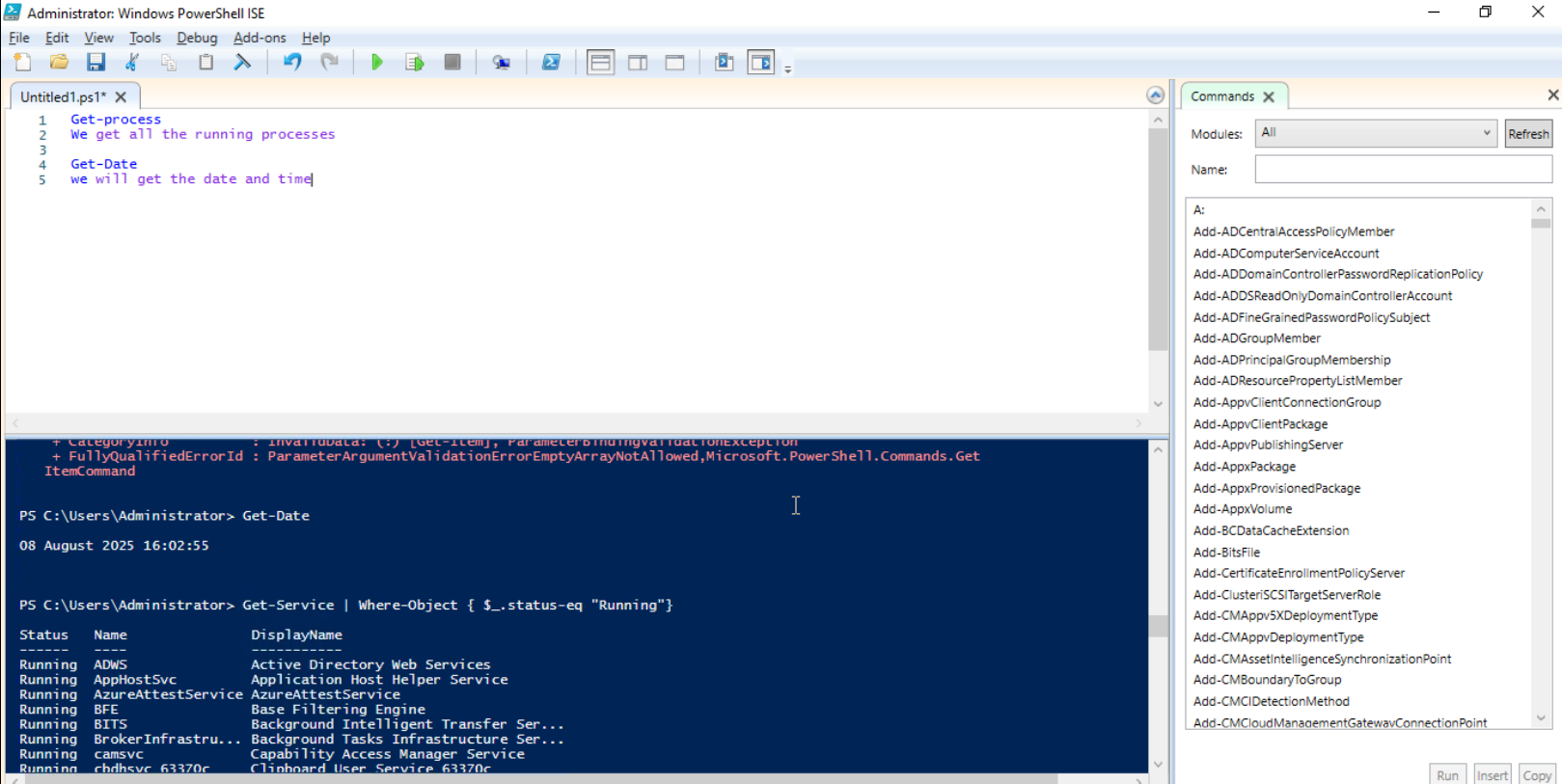
Some of the most commonly used cmdlets include:  
- Get-Help: Displays help information.  
- Get-Command: Lists all available cmdlets.  
- Get-Service: Retrieves the status of services.  
- Get-Process: Lists currently running processes.  
- Set-ExecutionPolicy: Changes the user preference for the PowerShell script execution policy.

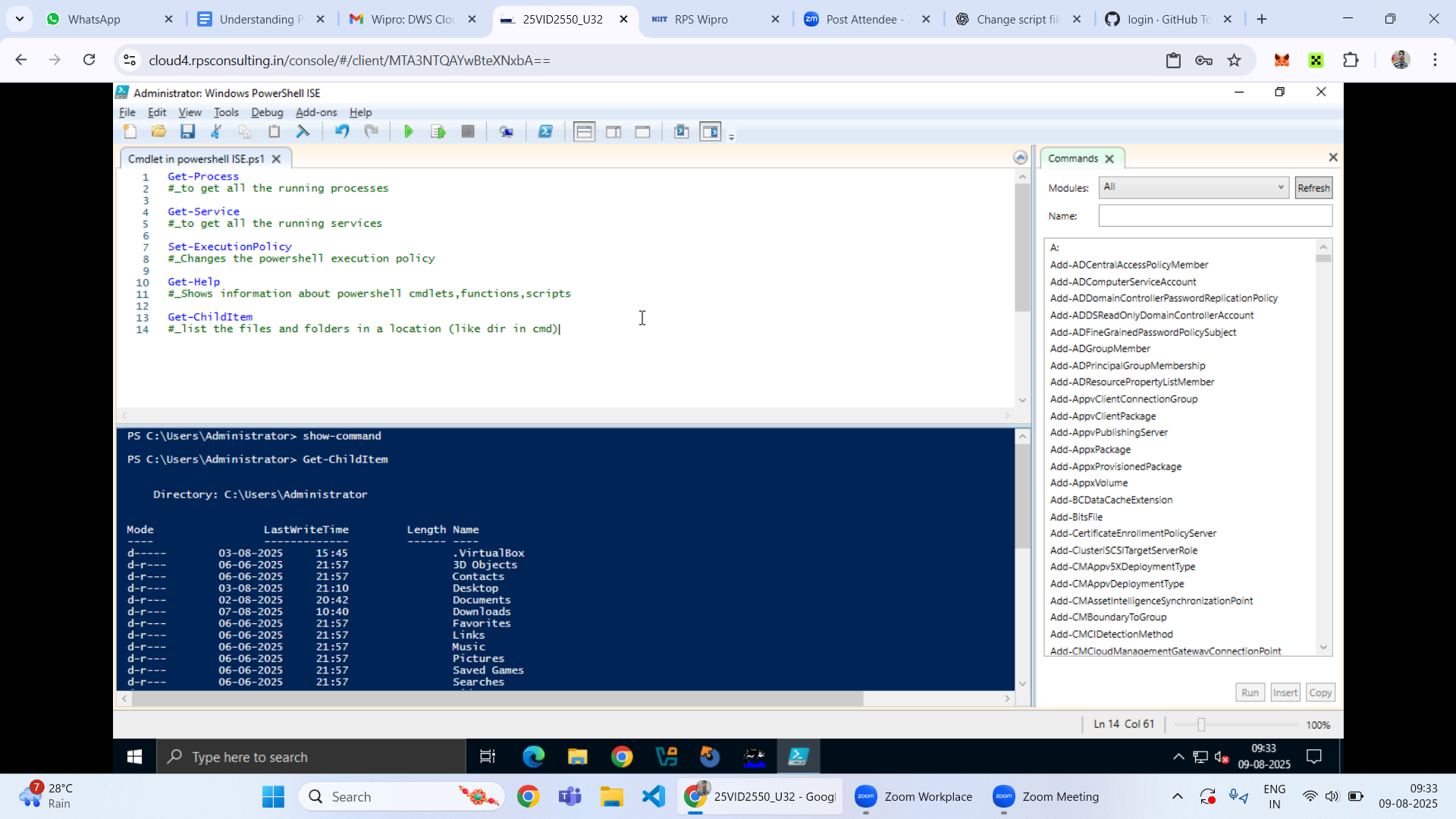
# WMI & PowerShell

WMI (Windows Management Instrumentation) allows management of Windows devices and systems.   
PowerShell can access WMI using cmdlets like Get-WmiObject or Get-CimInstance to retrieve system information like BIOS, disk drives, and network adapters.

# Pipeline Filtering & Operators

PowerShell supports powerful filtering using cmdlets like Where-Object and Select-Object.   
Operators like -eq (equals), -lt (less than), -gt (greater than), and -like (pattern matching) are used to compare and filter objects in the pipeline.



s

# Input, Output & Formatting

PowerShell can accept input from the user, files, or command line, and output data to the console, files, or other commands.   
Formatting cmdlets like Format-Table, Format-List, and Out-File allow for customized display and redirection of output.

# Scripting Overview

PowerShell scripts are text files with a .ps1 extension that contain a sequence of PowerShell commands.   
Scripts support variables, loops, conditionals, functions, and error handling to automate complex tasks. Execution policies and permissions must be managed to run scripts effectively.