1. What is PowerShell

PowerShell is a **task automation and configuration management tool** developed by **Microsoft**.
It includes a **command-line shell**, a **scripting language**, and a **framework** built on **.NET**.

It is mainly used to:

- Automate repetitive administrative tasks
- Manage system configurations
- Control cloud services like **Microsoft Azure**, **AWS**, etc.
- Work across **Windows, Linux, and macOS**

PowerShell combines the power of:

- **Command Prompt (cmd)** — for commands
- **Scripting language** — for automation
- **.NET framework** — for system and cloud integration

2. Why Use PowerShell in Cloud Computing

PowerShell is heavily used in **cloud computing** because it allows:

- **Automation** of cloud tasks (creating/deleting VMs, storage accounts, etc.)
- **Resource management** through scripts instead of manual GUI operations
- **Integration with cloud SDKs** (like Azure PowerShell, AWS Tools for PowerShell)
- **Remote management** of servers or services
- **DevOps pipelines** automation (CI/CD tasks)

Example cloud tasks automated by PowerShell:

```
# Create a new Azure Resource Group
New-AzResourceGroup -Name "MyResourceGroup" -Location "EastUS"
```

3. What is Scripting

A **script** is a set of instructions written in a programming or scripting language that automates a task.
In PowerShell, scripts use the **.ps1** file extension.

Example:

```
# Example PowerShell Script
Write-Host "Starting backup..."
Copy-Item "C:\Data" -Destination "D:\Backup" -Recurse
Write-Host "Backup completed successfully!"
```

Scripts are used to **automate cloud deployments**, **resource monitoring**, and **configuration updates**.

4. How to Check PowerShell Version

You can check your PowerShell version using any of these commands:

```
$PSVersionTable
```

or

```
$PSVersionTable.PSVersion
```

This shows details like **major**, **minor**, and **build version**.

Example Output:

```
Major  Minor  Build  Revision
-----  -----  -----  --------
7      4      0      0
```

5. What is PowerShell Administrator

Running PowerShell as **Administrator** gives you **elevated privileges**, meaning:

- You can make system-level changes
- Install or remove modules
- Access protected files or directories
- Manage cloud credentials securely

How to open PowerShell as Administrator:

1. Search "PowerShell" in Start Menu
2. Right-click → "Run as Administrator"

6. What is PowerShell ISE

ISE (Integrated Scripting Environment) is a GUI-based tool to **write, test, and debug** PowerShell scripts easily.

Key features:

- Syntax highlighting
- Auto-completion
- Integrated console
- Multiple script tabs
- Debugging support

How to open:
Search **"Windows PowerShell ISE"** in Start Menu.

Note: In modern Windows versions, **ISE** is being replaced by **Visual Studio Code with PowerShell extension**, which works better for cloud scripting.

7. History of PowerShell

Year	Event
2002	Microsoft began developing PowerShell (Project "Monad")
2006	PowerShell 1.0 released with Windows Server 2003 R2
2009	PowerShell 2.0 – added remoting features
2012	PowerShell 3.0 – improved scripting & workflows
2016	PowerShell 5.1 – last Windows-only version
2018	**PowerShell Core 6.0** released — cross-platform (Windows, Linux, macOS)
2020+	**PowerShell 7+** (based on .NET Core) – used widely in **cloud automation**

Developed by:

- **Microsoft Corporation**
- **Led by Jeffrey Snover** (Chief Architect, Windows Server Division)

Purpose:
To create a **powerful automation tool** that replaces traditional command-line tools and simplifies **system and cloud administration**.

1. What are Cmdlets (Command-lets)

Cmdlets are the **core commands** in PowerShell.
They are small, single-function commands built on the **.NET framework**, used to perform administrative or automation tasks.

Cmdlet Naming Convention

Each cmdlet follows a standard format:

```
Verb-Noun
```

Example:

```
Get-Process
Start-Service
New-Item
Remove-Item
```

Verb → Defines the action (Get, Set, New, Remove, Start, Stop, etc.)
Noun → Defines the target object (Process, Item, Service, etc.)

Examples

```
# Get a list of running processes
Get-Process

# Start a service
Start-Service -Name "wuauserv"

# Create a new folder
New-Item -Path "C:\CloudData" -ItemType Directory
```

2. Cmdlets for Cloud Computing

PowerShell provides **special cloud modules**, for example:

- **Azure PowerShell** module (Az)
- **AWS Tools for PowerShell**

Example: Azure Cmdlets

```
# Login to Azure
Connect-AzAccount

# Create a new resource group
New-AzResourceGroup -Name "MyCloudRG" -Location "EastUS"

# List all virtual machines
Get-AzVM
```

These cmdlets help automate cloud management instead of using a GUI dashboard.

3. What is Alias in PowerShell

An **Alias** is a **shortcut name** or **nickname** for a cmdlet.
It helps you type commands faster.

For example:

Alias	Full Cmdlet Name	Description
`ls`	`Get-ChildItem`	Lists files and folders
`dir`	`Get-ChildItem`	Same as above (Windows style)
`cp`	`Copy-Item`	Copies files or folders
`mv`	`Move-Item`	Moves or renames files/folders
`rm`	`Remove-Item`	Deletes files/folders
`cls`	`Clear-Host`	Clears the PowerShell screen

View All Aliases

```
Get-Alias
```

Find a Cmdlet of an Alias

```
Get-Alias ls
```

Create Your Own Alias

```
Set-Alias mydel Remove-Item
mydel test.txt   # same as Remove-Item test.txt
```

Tip: In cloud automation scripts, use **full cmdlet names** instead of aliases for better readability.

4. What is Pipeline in PowerShell

The **pipeline (|)** allows you to **send the output of one cmdlet** as **input to another cmdlet**.

It helps chain multiple commands together — just like a real pipeline passes data from one step to another.

Basic Example

```
Get-Process | Sort-Object CPU -Descending
```

`Get-Process` outputs all running processes
`Sort-Object` sorts them by CPU usage

5. Advantages of Pipeline

Benefit	Description
Efficiency	Reduces the need for temporary variables
Automation	Combine multiple actions in one command
Readability	Clear logical flow from one command to next
Cloud Power	Ideal for automating large-scale resource management

6. Summary

Concept	Description	Example
Cmdlet	Core PowerShell command performing one task	`Get-Service`, `New-Item`
Alias	Short name for a cmdlet	`ls` = `Get-ChildItem`
Pipeline	Transfers output from one cmdlet to another	`Get-Process	Sort-Object`

1. dir

Purpose: shows list of files and folders in the current directory.

syntax:

```
dir
```

Example:

```
c:\users\admin>dir
```

→ displays all files and folders in the current location.

2. dir | ft

Purpose: filters or formats the directory output for better viewing.

Syntax:

```
dir | ft
```

Explanation:
the pipe symbol `|` sends the output of `dir` to the `ft` (format-table) command for formatted display.

3. cd, cd/, cd..

Purpose: used to change the current working directory.

Syntax and Examples:

`cd foldername` → opens a subfolder.

`cd ..` → moves one step back (to parent folder).

`cd /` → goes to the root directory.

4. xcopy

Purpose: used for copying files and folders (more advanced than copy).

Syntax:

```
xcopy source destination
```

Example:

```
xcopy d:\notes c:\backup
```

→ copies all files from notes to backup folder.

5. move

Purpose: used for moving files or folders to another location.

Syntax:

```
move source destination
```

Example:

```
move d:\file.txt c:\folder
```

→ moves `file.txt` to the folder.

6. md / mkdir

Purpose: used for creating a new directory (folder).

Syntax:

```
md foldername
```

or

```
mkdir foldername
```

Example:

```
md projects
```

→ creates a folder named “projects”.

7. rmdir

Purpose: removes or deletes an empty directory.

Syntax:

```
rmdir foldername
```

Example:

```
rmdir olddata
```

8. del

Purpose: deletes one or more files.

Syntax:

```
del filename
```

Example:

```
del report.txt
```

→ deletes the file named `report.txt`.

9. type nul > adarsh.txt

Purpose: creates an empty text file.

Syntax:

```
type nul > filename.txt
```

Example:

```
type nul > adarsh.txt
```

→ creates a blank text file named `adarsh.txt`.

10. pwd

Purpose: shows the current working directory (mainly used in powershell or linux).

Syntax:

```
pwd
```

Example:

```
c:\users\admin>pwd
```

→ shows your current directory location.

PowerShell Networking Commands – Notes

1. IPCONFIG / Get-NetIPAddress / Get-NetIPConfiguration

- **Purpose:** Displays network configuration details of your system.
- **Example Uses:**
  - `ipconfig` → Shows IPv4, IPv6, subnet mask, default gateway, etc.
  - `Get-NetIPAddress` → Shows all IP addresses assigned to your network adapters.
  - `Get-NetIPConfiguration` → Displays full network configuration (DNS, Gateway, etc.).

2. Tracert / Test-NetConnection -Traceroute

- **Purpose:** Checks the route (path) taken by packets to reach a destination.
- **Example Uses:**
  - `tracert 127.0.0.0` → Displays hops between your computer and the destination.

    - `Test-NetConnection -Traceroute www.google.com` → Shows route information using PowerShell.

3. Get-NetAdapter / wmic nic get adaptertype, name, MACAddress

- **Purpose:** Displays network adapter details.
- **Example Uses:**
    - `Get-NetAdapter` → Lists all network adapters with their status (Up/Down).
    - `wmic nic get adaptertype, name, MACAddress` → Shows NIC (Network Interface Card) info including MAC address.

4. Ping / Test-NetConnection

- **Purpose:** Tests network connectivity between your system and a remote host.
- **Example Uses:**
    - `ping 127.0.0.0` → Tests connection to the given IP.
    - `Test-NetConnection www.google.com` → Tests connection and gives more details (Ping, Port, DNS, etc.).

5. Enable / Disable-NetAdapter

- **Purpose:** Turns a network adapter ON or OFF.
- **Example Uses:**
    - `Enable-NetAdapter -Name "Ethernet"` → Enables Ethernet adapter.
    - `Disable-NetAdapter -Name "Ethernet"` → Disables Ethernet adapter.

6. DNSClient

- **Purpose:** Manages and displays DNS client settings.
- **Example Use:**
    - `Get-DnsClient` → Shows DNS configuration of network interfaces.

7. Resolve-DnsName

- **Purpose:** Resolves a domain name to its IP address (like nslookup).
- **Example Use:**
    - `Resolve-DnsName www.google.com` → Displays the IP address of Google's domain.

9. New-NetIPAddress

- **Purpose:** Assigns a new static IP address to a network adapter.
- **Example Use:**
    - `New-NetIPAddress -IPAddress 127.0.0.0 -PrefixLength 24 -InterfaceAlias "Ethernet"` → Sets a static IP with a subnet prefix.

10. netsh interface ipv4 set address

- **Purpose:** Configures static IP settings using `netsh` (older method).

- **Syntax:**
- `netsh interface ipv4 set address name="Ethernet" static IP SM DG`
    - **IP:** IP Address
    - **SM:** Subnet Mask
    - **DG:** Default Gateway
- **Example:**

```
netsh interface ipv4 set address name="Ethernet" static 192.168.1.10
255.255.255.0 192.168.1.1
```

11. Get-Date

- **Purpose:** Displays the current system date and time.
- **Example Use:**
- `Get-Date`

→ Shows current date, time, and timezone information.

12. Get-TimeZone

- **Purpose:** Displays the current system timezone settings.
- **Example Use:**
- `Get-TimeZone`

→ Shows timezone name, offset from UTC, and daylight saving info.

13. Notepad, Control, Calc, mspaint & winword

- **Purpose:** Opens common Windows applications using PowerShell.
- **Examples:**
    - `notepad` → Opens Notepad.
    - `control` → Opens Control Panel.
    - `calc` → Opens Calculator.
    - `mspaint` → Opens Microsoft Paint.
    - `winword` → Opens Microsoft Word.

14. Set-Content file.txt 'Here is your text'

- **Purpose:** Creates or replaces the content of a file.
- **Example Use:**
- `Set-Content file.txt 'Here is your text'`

→ Creates a file named **file.txt** and writes the given text inside it.

Powershell ise commands for create a file and add text some