



This lab is designed to help you gain hands-on experience using PowerShell for system administration tasks.

You will practice working with commands to manage files, services, processes, event logs, and system monitoring.

Lab 2 - PowerShell Basics & System Administration

420-636-AB-Network
Installation and Administration
II

Teacher: Antoine Tohme
Student: Monica Perez Mata
Student id : 2498056

1	Task 1: Exploring PowerShell Commands.....	4
1.1	Identify available commands.....	4
1.1.1	List all commands	4
1.1.2	Filtering Commands	4
1.1.2.1	List only cmdlets:.....	4
1.1.2.2	Find commands related to networking:	5
1.1.2.3	Discover commands for remote sessions	6
1.2	Retrieve detailed help for a specific command.	6
1.2.1	Get basic syntax	6
1.2.2	Retrieve detailed help for a specific PowerShell command	6
1.2.3	Get examples.....	7
1.2.4	Get full documentation, including parameters and descriptions:	8
1.2.5	Get online detailed version.....	8
1.3	List all properties and methods of an object.	9
1.3.1	Basic Syntax	9
1.3.2	Inspect a Specific Object	10
1.3.3	Filtering for Only Properties or Methods.....	10
1.3.3.1	List only properties:.....	10
1.3.3.2	List only methods:.....	11
2	Task 2: Working with Objects	11
2.1	Display process information in table and list formats.	11
2.1.1	Table Format (Structured View)	11
2.1.2	List Format (Detailed View)	12
2.1.3	Custom Table View	13
2.2	Sort processes based on CPU usage.....	14
2.2.1	Additional Customization.....	15
2.3	Select and filter objects based on specific conditions.....	17
2.3.1	Select Specific Properties	17
2.3.2	Filter Objects Based on a Condition	18

2.3.3	Filter Objects Using Multiple Conditions.....	18
2.3.4	Select Top N Results	18
2.4	Loop through system objects and extract specific information.	19
2.4.1	Using `ForEach-Object` (Pipeline).....	19
2.4.2	Using a `foreach` Loop	19
2.4.3	Loop Through Services and Extract Specific Details	20
3	Task 3: Managing the File System.....	21
3.1	Navigate through directories and list files.	21
3.1.1	Change Directory (`cd` or `Set-Location`).....	21
3.1.2	List Files in a Directory (` Get-ChildItem`)	22
3.1.3	Show Only Files (Exclude Folders)	23
3.1.4	Show Only Folders (Exclude Files)	23
3.1.5	List Files with Specific Extensions	24
3.1.6	Navigate Back to Previous Directory	24
3.2	Create and delete files and folders	25
3.2.1	Create a New File.....	25
3.2.2	Create a New Folder	25
3.2.3	Delete a File.....	26
3.2.4	Delete a Folder	26
3.2.4.1	To remove an empty folder:.....	26
3.2.4.2	To delete a folder with contents, use:	27
3.3	Copy and move files between locations.....	27
3.3.1	Prepare.....	27
3.3.2	Copy a File.....	29
3.3.3	Copy an Entire Folder.....	29
3.3.4	Move a File.....	29
3.3.5	Move an Entire Folder.....	30
3.3.6	Restore.....	30
3.4	Check available disk space.	32

4	Task 4: Managing System Services and Processes.....	32
4.1	List all running services on the system.....	32
4.2	Start and stop specific services.....	33
4.3	Retrieve information about active processes.	34
4.4	Terminate a process	35
5	Task 5: Monitoring Event Logs and System Information.....	35
5.1	View the latest system event logs.	35
5.2	Retrieve security event logs.	36
5.3	Extract operating system details using PowerShell.	37

1 Task 1: Exploring PowerShell Commands

1.1 Identify available commands.

1.1.1 List all commands

Get-command

CommandType	Name	Version	Source
Alias	Add-AppPackage	2.0.1.0	Appx
Alias	Add-AppPackageVolume	2.0.1.0	Appx
Alias	Add-AppProvisionedPackage	3.0	Dism
Alias	Add-ProvisionedAppPackage	3.0	Dism
Alias	Add-ProvisionedAppxPackage	3.0	Dism
Alias	Add-WindowsFeature	2.0.0.0	ServerManager
Alias	Apply-WindowsUnattend	3.0	Dism
Alias	Disable-PhysicalDiskIndication	2.0.0.0	Storage
Alias	Disable-PhysicalDiskIndication	1.0.0.0	VMDirectStorage
Alias	Disable-StorageDiagnosticLog	2.0.0.0	Storage
Alias	Disable-StorageDiagnosticLog	1.0.0.0	VMDirectStorage
Alias	Dismount-AppPackageVolume	2.0.1.0	Appx
Alias	Enable-PhysicalDiskIndication	2.0.0.0	Storage
Alias	Enable-PhysicalDiskIndication	1.0.0.0	VMDirectStorage
Alias	Enable-StorageDiagnosticLog	2.0.0.0	Storage
Alias	Enable-StorageDiagnosticLog	1.0.0.0	VMDirectStorage
Alias	Expand-IscsiVirtualDisk	2.0.0.0	IscsiTarget
Alias	Flush-Volume	2.0.0.0	Storage
Alias	Flush-Volume	1.0.0.0	VMDirectStorage
Alias	Get-AppPackage	2.0.1.0	Appx
Alias	Get-AppPackageDefaultVolume	2.0.1.0	Appx
Alias	Get-AppPackageLastError	2.0.1.0	Appx
Alias	Get-AppPackageLog	2.0.1.0	Appx
Alias	Get-AppPackageManifest	2.0.1.0	Appx
Alias	Get-AppPackageVolume	2.0.1.0	Appx
Alias	Get-AppProvisionedPackage	3.0	Dism
Alias	Get-DiskSNV	2.0.0.0	Storage
Alias	Get-DiskSNV	1.0.0.0	VMDirectStorage
Alias	Get-PhysicalDiskSNV	2.0.0.0	Storage
Alias	Get-PhysicalDiskSNV	1.0.0.0	VMDirectStorage
Alias	Get-ProvisionedAppPackage	3.0	Dism
Alias	Get-ProvisionedAppxPackage	3.0	Dism
Alias	Get-StorageEnclosureSNV	2.0.0.0	Storage
Alias	Get-StorageEnclosureSNV	1.0.0.0	VMDirectStorage

1.1.2 Filtering Commands

1.1.2.1 List only cmdlets:

Get-Command - CommandType Cmdlet

```

PS C:\Users\Administrator> ### Filtering Commands
PS C:\Users\Administrator> ##List only cmdlets:
PS C:\Users\Administrator> Get-Command - CommandType Cmdlet

CommandType      Name                           Version   Source
----           ----
Cmdlet          Add-AppvClientConnectionGroup    1.0.0.0   AppvClient
Cmdlet          Add-AppvClientPackage            1.0.0.0   AppvClient
Cmdlet          Add-AppvPublishingServer         1.0.0.0   AppvClient
Cmdlet          Add-AppxPackage                 2.0.1.0   Appx
Cmdlet          Add-AppxProvisionedPackage       3.0       Dism
Cmdlet          Add-AppxVolume                 2.0.1.0   Appx
Cmdlet          Add-BitsFile                  2.0.0.0   BitsTransfer
Cmdlet          Add-CertificateEnrollmentPolicyServer 1.0.0.0   PKI
Cmdlet          Add-ClusterIsCSITargetServerRole 2.0.0.0   IscsiTarget
Cmdlet          Add-Computer                  3.1.0.0   Microsoft.PowerShell.Management
Cmdlet          Add-Content                   3.1.0.0   Microsoft.PowerShell.Management
Cmdlet          Add-History                  3.0       Microsoft.PowerShell.Core
Cmdlet          Add-IscsiVirtualDiskTargetMapping 2.0.0.0   IscsiTarget
Cmdlet          Add-JobTrigger                1.1.0.0   PSScheduledJob
Cmdlet          Add-KdsRootKey               1.0.0.0   Kds
Cmdlet          Add-LocalGroupMember          1.0.0.0   Microsoft.PowerShell.LocalAccounts
Cmdlet          Add-Member                   3.1.0.0   Microsoft.PowerShell.Utility
Cmdlet          Add-PSSnapin                3.0.0.0   Microsoft.PowerShell.Core
Cmdlet          Add-SignerRule              1.0       ConfigCI
Cmdlet          Add-Type                    3.1.0.0   Microsoft.PowerShell.Utility
Cmdlet          Add-WindowsCapability        3.0       Dism
Cmdlet          Add-WindowsDriver             3.0       Dism
Cmdlet          Add-WindowsImage              3.0       Dism
Cmdlet          Add-WindowsPackage            3.0       Dism
Cmdlet          Backup-AuditPolicy           1.0.0.0   SecurityCmdlets
Cmdlet          Backup-SecurityPolicy        1.0.0.0   SecurityCmdlets
Cmdlet          Checkpoint-Computer          3.1.0.0   Microsoft.PowerShell.Management
Cmdlet          Checkpoint-IscsiVirtualDisk   2.0.0.0   IscsiTarget
Cmdlet          Clear-Content                3.1.0.0   Microsoft.PowerShell.Management
Cmdlet          Clear-EventLog                3.1.0.0   Microsoft.PowerShell.Management
Cmdlet          Clear-History                3.0.0.0   Microsoft.PowerShell.Core
Cmdlet          Clear-Item                  3.1.0.0   Microsoft.PowerShell.Management

```

1.1.2.2 Find commands related to networking:

Get-Command -Name *Net*

```

Administrator: Windows PowerShell
PS C:\Users\Administrator> #Find commands related to networking:
PS C:\Users\Administrator> Get-Command -Name *Net*

CommandType      Name                           Version   Source
----           ----
Function         Add-NetEventNetworkAdapter     1.0.0.0   NetEventPacketCapture
Function         Add-NetEventPacketCaptureProvider 1.0.0.0   NetEventPacketCapture
Function         Add-NetEventProvider            1.0.0.0   NetEventPacketCapture
Function         Add-NetEventVFPProvider         1.0.0.0   NetEventPacketCapture
Function         Add-NetEventVmNetworkAdapter    1.0.0.0   NetEventPacketCapture
Function         Add-NetEventVmSwitch           1.0.0.0   NetEventPacketCapture
Function         Add-NetEventVmSwitchProvider     1.0.0.0   NetEventPacketCapture
Function         Add-NetEventWPCaptureProvider   1.0.0.0   NetEventPacketCapture
Function         Add-NetIPHttpsCertBinding       1.0.0.0   NetworkTransition
Function         Add-NetLbfoTeamMember          2.0.0.0   NetLbfo
Function         Add-NetLbfoTeamNic            2.0.0.0   NetLbfo
Function         Add-NetNatExternalAddress      1.0.0.0   NetNat
Function         Add-NetNatStaticMapping        1.0.0.0   NetNat
Function         Add-NetSwitchTeamMember        1.0.0.0   NetSwitchTeam
Function         Add-VpnConnectionTriggerTrustedNetwork 2.0.0.0   VpnClient
Function         Copy-NetFirewallRule           2.0.0.0   NetSecurity
Function         Copy-NetIPsecMainModeCryptoSet 2.0.0.0   NetSecurity
Function         Copy-NetIPsecMainModeRule       2.0.0.0   NetSecurity
Function         Copy-NetIPsecPhase1AuthSet     2.0.0.0   NetSecurity
Function         Copy-NetIPsecPhase2AuthSet     2.0.0.0   NetSecurity
Function         Copy-NetIPsecQuickModeCryptoSet 2.0.0.0   NetSecurity
Function         Copy-NetIPsecRule              2.0.0.0   NetSecurity
Function         Disable-NetAdapter            2.0.0.0   NetAdapter
Function         Disable-NetAdapterBinding      2.0.0.0   NetAdapter
Function         Disable-NetAdapterChecksumOffload 2.0.0.0   NetAdapter
Function         Disable-NetAdapterEncapsulatedPacketTaskOffload 2.0.0.0   NetAdapter
Function         Disable-NetAdapterIPsecOffload    2.0.0.0   NetAdapter
Function         Disable-NetAdapterLso            2.0.0.0   NetAdapter
Function         Disable-NetAdapterPacketDirect 2.0.0.0   NetAdapter
Function         Disable-NetAdapterPowerManagement 2.0.0.0   NetAdapter
Function         Disable-NetAdapterQos            2.0.0.0   NetAdapter
Function         Disable-NetAdapterRdma           2.0.0.0   NetAdapter
Function         Disable-NetAdapterRsc            2.0.0.0   NetAdapter

```

1.1.2.3 Discover commands for remote sessions

Get-Command -Name *PSSession*

```
PS C:\Users\Administrator> #Discover commands for remote sessions:  
PS C:\Users\Administrator> Get-Command -Name *PSSession*  


| CommandType | Name                              | Version | Source                       |
|-------------|-----------------------------------|---------|------------------------------|
| Cmdlet      | Connect-PSSession                 | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Disable-PSSessionConfiguration    | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Disconnect-PSSession              | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Enable-PSSessionConfiguration     | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Enter-PSSession                   | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Exit-PSSession                    | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Export-PSSession                  | 3.1.0.0 | Microsoft.PowerShell.Utility |
| Cmdlet      | Get-PSSession                     | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Get-PSSessionCapability           | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Get-PSSessionConfiguration        | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Import-PSSession                  | 3.1.0.0 | Microsoft.PowerShell.Utility |
| Cmdlet      | New-PSSession                     | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | New-PSSessionConfigurationFile    | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | New-PSSessionOption               | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Receive-PSSession                 | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Register-PSSessionConfiguration   | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Remove-PSSession                  | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Set-PSSessionConfiguration        | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Test-PSSessionConfigurationFile   | 3.0.0.0 | Microsoft.PowerShell.Core    |
| Cmdlet      | Unregister-PSSessionConfiguration | 3.0.0.0 | Microsoft.PowerShell.Core    |

  
PS C:\Users\Administrator>
```

1.2 Retrieve detailed help for a specific command.

1.2.1 Get basic syntax

Get-Help <Cmdlet-Name> -Detailed

1.2.2 Retrieve detailed help for a specific PowerShell command

Get-Help Get-NetAdapter -Detailed

Note - Update help will be loaded when running for first time

```
PS Select Administrator: Windows PowerShell
PS C:\Users\Administrator> ### Get Help on a Specific Command
PS C:\Users\Administrator> # Get-Help <cmdlet-Name> -Detailed
PS C:\Users\Administrator> Get-Help Get-NetAdapter -Detailed

Do you want to run Update-Help?
The Update-Help cmdlet downloads the most current Help files for Windows PowerShell modules, and installs them on your computer. For more information about the Update-Help cmdlet, see https://go.microsoft.com/fwlink/?LinkID=210614.
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"); Y

NAME
Get-NetAdapter

SYNOPSIS
Gets the basic network adapter properties.

SYNTAX
Get-NetAdapter [-AsJob] [-CimSession <CimSession[]>] [-IncludeHidden] -InterfaceDescription <String[]> [-Physical] [-ThrottleLimit <Int32>] [<CommonParameters>]
Get-NetAdapter [-AsJob] [-CimSession <CimSession[]>] [-IncludeHidden] -InterfaceIndex <UInt32[]> [-Physical] [-ThrottleLimit <Int32>] [<CommonParameters>]
Get-NetAdapter [[-Name] <String[]>] [-AsJob] [-CimSession <CimSession[]>] [-IncludeHidden] [-Physical] [-ThrottleLimit <Int32>] [<CommonParameters>]

DESCRIPTION
The Get-NetAdapter cmdlet gets the basic network adapter properties. By default only visible adapters are returned. To see the common network adapter properties, pipe the output into the Format-List cmdlet. To see all the properties, pipe the output to the Format-Table cmdlet with the Property parameter specified as the wildcard character "*". This cmdlet supports multiple views. The default view is as a table. To see more information regarding various network adapter identifiers use the names view using the Format-Table cmdlet with the View parameter specified as name . To see more information regarding the miniport, device driver, such as driver date or version use the driven view using the Format-Table cmdlet with the View parameter specified as driver .

PARAMETERS
-AsJob [<SwitchParameter>]
Runs the cmdlet as a background job. Use this parameter to run commands that take a long time to complete. The cmdlet immediately returns an object that represents the job and then displays the command prompt. You can continue to work in the session while the job completes. To manage the job, use the *-Job cmdlets. To get the job results, use the Receive-Job (https://go.microsoft.com/fwlink/?LinkID=113372) cmdlet. For more information about Windows PowerShell® background jobs, see about_Jobs

7:32 PM
```

1.2.3 Get examples

Get-Help Get-NetAdapter -Examples

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> # See examples, use:
PS C:\Users\Administrator> Get-Help Get-NetAdapter -Examples

NAME
Get-NetAdapter

SYNOPSIS
Gets the basic network adapter properties.

----- Example 1: Get all visible network adapters -----

PS C:\> Get-NetAdapter -Name *

This command gets all of the visible network adapters.
---- Example 2: Get all visible and hidden network adapters ----

PS C:\> Get-NetAdapter -Name * -IncludeHidden

This command gets all of the network adapters.
----- Example 3: Get all physical network adapters ----

PS C:\> Get-NetAdapter -Name * -Physical

This command gets all of the physical network adapters.
---- Example 4: Get a network adapter by the specified name ----

PS C:\> Get-NetAdapter -Name "Ethernet 2"
Example 12: Gets visible network adapters and formats the output
```

1.2.4 Get full documentation, including parameters and descriptions:

Get-Help Get-NetAdapter -Full

```
PS C:\Users\Administrator> #Get full documentation, including parameters and descriptions:
PS C:\Users\Administrator>
PS C:\Users\Administrator> Get-Help Get-NetAdapter -Full

NAME
  Get-NetAdapter

SYNOPSIS
  Gets the basic network adapter properties.

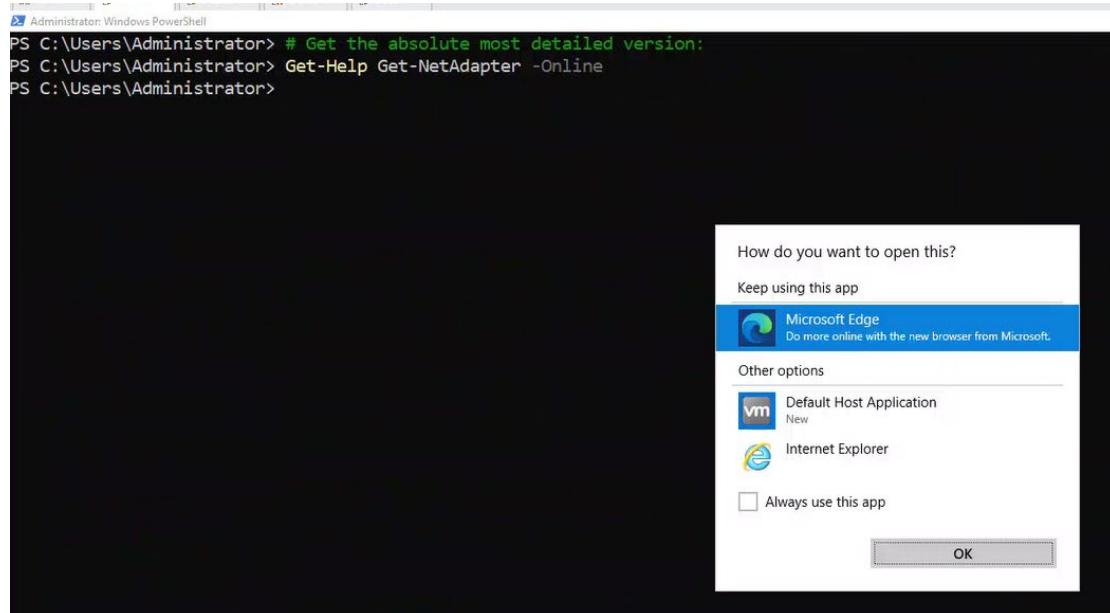
SYNTAX
  Get-NetAdapter [-AsJob] [-CimSession <CimSession[]>] [-IncludeHidden] -InterfaceDescription <String[]> [-Physical] [-ThrottleLimit <Int32>] [<CommonParameters>]
  Get-NetAdapter [-AsJob] [-CimSession <CimSession[]>] [-IncludeHidden] -InterfaceIndex <UInt32[]> [-Physical] [-ThrottleLimit <Int32>] [<CommonParameters>]
  Get-NetAdapter [[-Name] <String[]>] [-AsJob] [-CimSession <CimSession[]>] [-IncludeHidden] [-Physical] [-ThrottleLimit <Int32>] [<CommonParameters>]

DESCRIPTION
  The Get-NetAdapter cmdlet gets the basic network adapter properties. By default only visible adapters are returned. To see the common network adapter properties, pipe to the Format-List cmdlet. To see all the properties, pipe the output to the Format-List cmdlet with the Property parameter specified as the wildcard character "*". This cmdlet supports multiple views. The default view is as a table. To see more information regarding various network adapter identifiers use the names view using the Format-Table cmdlet with the View parameter specified as name . To see more information regarding the miniport, device driver, such as driver date or version use the driver view using the Format-Table cmdlet with the View parameter specified as driver .

PARAMETERS
  -AsJob [<SwitchParameter>]
    Runs the cmdlet as a background job. Use this parameter to run commands that take a long time to complete. The cmdlet immediately returns an object that represents the command prompt. You can continue to work in the session while the job completes. To manage the job, use the *-Job cmdlets. To get the job result, receive-Job (https://go.microsoft.com/fwlink/?LinkID=113372) cmdlet. For more information about Windows PowerShell® background jobs, see about_Jobs (https://go.microsoft.com/fwlink/?LinkID=113251).
  -Required? false
  Activate Windows
  Go to Settings to activate Windows.
```

1.2.5 Get online detailed version

Get-Help Get-NetAdapter -Online



1.3 List all properties and methods of an object.

1.3.1 Basic Syntax

To list all properties and methods of an object in , use the `Get-Member` cmdlet:

Get-Service | Get-Member

This lists all methods and properties of objects returned by `Get-Service` .

```
PS C:\Users\Administrator> Get-Service | Get-Member

TypeName: System.ServiceProcess.ServiceController
Name          MemberType   Definition
----          -----
Name          AliasProperty Name = ServiceName
RequiredServices AliasProperty RequiredServices = ServicesDependedOn
Disposed       Event        System.EventHandler Disposed(System.Object, System.EventArgs)
Close         Method      void Close()
Continue     Method      void Continue()
CreateObjRef Method      System.Runtime.Remoting.ObjRef CreateObjRef(type requestedType)
Dispose       Method      void Dispose(), void IDisposable.Dispose()
Equals        Method      bool Equals(System.Object obj)
ExecuteCommand Method      void ExecuteCommand(int command)
GetHashCode   Method      int GetHashCode()
GetLifetimeService Method      System.Object GetLifetimeService()
GetType       Method      type GetType()
InitializeLifetimeService Method      System.Object InitializeLifetimeService()
Pause         Method      void Pause()
Refresh       Method      void Refresh()
Start         Method      void Start(), void Start(string[] args)
Stop          Method      void Stop()
WaitForStatus Method      void WaitForStatus(System.ServiceProcess.ServiceControllerStatus desiredStatus), void WaitForStatus(System.ServiceProcess.ServiceControllerStatus ...)

CanPauseAndContinue  Property   bool CanPauseAndContinue {get;}
CanShutdown        Property   bool CanShutdown {get;}
CanStop           Property   bool CanStop {get;}
Container          Property   System.ComponentModel.IContainer Container {get;}
DependentServices  Property   System.ServiceProcess.ServiceController[] DependentServices {get;}
DisplayName        Property   string DisplayName {get;set;}
MachineName        Property   string MachineName {get;set;}
ServiceHandle      Property   System.Runtime.InteropServices.SafeHandle ServiceHandle {get;}
ServiceName        Property   string ServiceName {get;set;}
ServicesDependedOn Property   System.ServiceProcess.ServiceController[] ServicesDependedOn {get;}
ServiceType        Property   System.ServiceProcess.ServiceType ServiceType {get;}
Site              Property   System.ComponentModel.ISite Site {get;set;}
```

1.3.2 Inspect a Specific Object

To check properties and methods for a specific object, use:

\$object = Get-NetAdapter

\$object | Get-Member

```
PS C:\Users\Administrator> $object = Get-NetAdapter
PS C:\Users\Administrator> $object | Get-Member

TypeName: Microsoft.Management.Infrastructure.CimInstance#ROOT/StandardCimv2/MSFT_NetAdapter
Name          MemberType   Definition
----          ----
DriveVersion  AliasProperty DriveVersion = DriveVersionString
ifAlias       AliasProperty ifAlias = Name
ifDesc        AliasProperty ifDesc = InterfaceDescription
ifIndex       AliasProperty ifIndex = InterfaceIndex
ifName        AliasProperty ifName = InterfaceName
InterfaceAlias AliasProperty InterfaceAlias = Name
LinkLayerAddress
Clone        Method     System.Object Clone()
Dispose      Method     void Dispose(), void IDisposable.Dispose()
Equals       Method     bool Equals(System.Object obj)
GetCimSessionComputerName
GetCimSessionInstanceId
GetHashCode   Method     guid GetCimSessionInstanceId()
GetObjectData
GetType      Method     type GetType()
ToString    Method     string ToString()
ActiveMaximumTransmissionUnit
AdditionalAvailability
AdminLocked
AutoSense
Availability
AvailableRequestedStates
Caption
CommunicationStatus
ComponentID
ConnectorPresent
CreationClassName
Description
DetailedStatus
DeviceID
DeviceName
DeviceWakeUpEnable
DriverDate
DriverDateData
DriverDescription
DriverMajorNdisVersion
DeviceVersionString
ElementName
EnabledDefault
EnabledState
EndPointInterface
ErrorCleared
ErrorMessage
FullDuplex

Name          MemberType   Definition
----          ----
DriveVersion  Property   uint64 ActiveMaximumTransmissionUnit {get;set;}
ifAlias       Property   uint16[] AdditionalAvailability {get;set;}
ifDesc        Property   bool AdminLocked {get;}
AutoSense      Property   bool AutoSense {get;set;}
Availability   Property   uint16 Availability {get;set;}
AvailableRequestedStates
Caption        Property   string Caption {get;set;}
CommunicationStatus
ComponentID   Property   string ComponentID {get;}
ConnectorPresent
CreationClassName
Description     Property   string CreationClassName {get;set;}
DetailedStatus
DeviceID       Property   uint16 Detailedstatus {get;set;}
DeviceName     Property   string DeviceID {get;set;}
DeviceWakeUpEnable
DriverDate     Property   string DeviceName {get;}
DriverDateData
DriverDescription
DriverMajorNdisVersion
DriverMinorNdisVersion
DriverName      Property   string DriverDateData {get;}
DriverProvider
DriverVersionString
ElementName    Property   string DriverDescription {get;}
EnabledDefault
EnabledState
EndPointInterface
ErrorCleared
ErrorMessage
FullDuplex

Name          MemberType   Definition
----          ----
DriveVersion  Property   byte ActiveMaximumTransmissionUnit {get;set;}
ifAlias       Property   byte AdditionalAvailability {get;set;}
ifDesc        Property   bool AdminLocked {get;}
AutoSense      Property   bool AutoSense {get;set;}
Availability   Property   byte Availability {get;set;}
AvailableRequestedStates
Caption        Property   string Caption {get;set;}
CommunicationStatus
ComponentID   Property   string ComponentID {get;}
ConnectorPresent
CreationClassName
Description     Property   string CreationClassName {get;set;}
DetailedStatus
DeviceID       Property   byte Detailedstatus {get;set;}
DeviceName     Property   string DeviceID {get;set;}
DeviceWakeUpEnable
DriverDate     Property   string DeviceName {get;}
DriverDateData
DriverDescription
DriverMajorNdisVersion
DriverMinorNdisVersion
DriverName      Property   string DriverDateData {get;}
DriverProvider
DriverVersionString
ElementName    Property   string DriverDescription {get;}
EnabledDefault
EnabledState
EndPointInterface
ErrorCleared
ErrorMessage
FullDuplex

Name          MemberType   Definition
----          ----
DriveVersion  Property   string DriveVersionString {get;set;}
ifAlias       Property   string ElementName {get;set;}
ifDesc        Property   uint16 EnabledDefault {get;set;}
ifIndex       Property   uint16 EnabledState {get;set;}
ifName        Property   bool EndpointInterface {get;}
ifName        Property   bool ErrorCleared {get;set;}
ifName        Property   string ErrorMessage {get;set;}
ifName        Property   bool FullDuplex {get;set;}
```

1.3.3 Filtering for Only Properties or Methods

1.3.3.1 List only properties:

\$object | Get-Member -MemberType Property

```
PS C:\Users\Administrator> $object | Get-Member -MemberType Property

TypeName: Microsoft.Management.Infrastructure.CimInstance#ROOT/StandardCimv2/MSFT_NetAdapter
Name          MemberType   Definition
----          ----
DriveVersion  Property   uint64 ActiveMaximumTransmissionUnit {get;set;}
ifAlias       Property   uint16[] AdditionalAvailability {get;set;}
ifDesc        Property   bool AdminLocked {get;}
AutoSense      Property   bool AutoSense {get;set;}
Availability   Property   uint16 Availability {get;set;}
AvailableRequestedStates
Caption        Property   string Caption {get;set;}
CommunicationStatus
ComponentID   Property   string ComponentID {get;}
ConnectorPresent
CreationClassName
Description     Property   string CreationClassName {get;set;}
DetailedStatus
DeviceID       Property   uint16 Detailedstatus {get;set;}
DeviceName     Property   string DeviceID {get;set;}
DeviceWakeUpEnable
DriverDate     Property   string DeviceName {get;}
DriverDateData
DriverDescription
DriverMajorNdisVersion
DriverMinorNdisVersion
DriverName      Property   string DriverDateData {get;}
DriverProvider
DriverVersionString
ElementName    Property   string DriverDescription {get;}
EnabledDefault
EnabledState
EndPointInterface
ErrorCleared
ErrorMessage
FullDuplex

Name          MemberType   Definition
----          ----
DriveVersion  Property   byte ActiveMaximumTransmissionUnit {get;set;}
ifAlias       Property   byte AdditionalAvailability {get;set;}
ifDesc        Property   bool AdminLocked {get;}
AutoSense      Property   bool AutoSense {get;set;}
Availability   Property   byte Availability {get;set;}
AvailableRequestedStates
Caption        Property   string Caption {get;set;}
CommunicationStatus
ComponentID   Property   string ComponentID {get;}
ConnectorPresent
CreationClassName
Description     Property   string CreationClassName {get;set;}
DetailedStatus
DeviceID       Property   byte Detailedstatus {get;set;}
DeviceName     Property   string DeviceID {get;set;}
DeviceWakeUpEnable
DriverDate     Property   string DeviceName {get;}
DriverDateData
DriverDescription
DriverMajorNdisVersion
DriverMinorNdisVersion
DriverName      Property   string DriverDateData {get;}
DriverProvider
DriverVersionString
ElementName    Property   string DriverDescription {get;}
EnabledDefault
EnabledState
EndPointInterface
ErrorCleared
ErrorMessage
FullDuplex
```

1.3.3.2 List only methods:

\$object | Get-Member -MemberType Method

```
PS C:\Users\Administrator> $object | Get-Member -MemberType Method

TypeName: Microsoft.Management.Infrastructure.CimInstance#ROOT/StandardCimv2/MSFT_NetAdapter

Name           MemberType Definition
---           ----   ~~~~~
Clone          Method   System.Object ICloneable.Clone()
Dispose        Method   void Dispose(), void IDisposable.Dispose()
Equals         Method   bool Equals(System.Object obj)
GetCimSessionComputerName Method string GetCimSessionComputerName()
GetCimSessionInstanceId Method guid GetCimSessionInstanceId()
GetHashCode     Method   int GetHashCode()
GetObjectData  Method   void GetObjectData(System.Runtime.Serialization.SerializationInfo info, System.Runtime.Serialization.StreamingContext context), void ISerializable.GetObjectData(System.Runtime.Serialization.SerializationInfo info, System.Runtime.Serialization.StreamingContext context)
GetType        Method   type GetType()
ToString       Method   string ToString()
```

2 Task 2: Working with Objects

2.1 Display process information in table and list formats.

2.1.1 Table Format (Structured View)

Use the `Format-Table` cmdlet for a neatly organized table:

Get-Process | Format-Table -AutoSize

`-AutoSize` ensures columns adjust for readability.

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
92	6	904	4712	0.02	3316	0	AggregatorHost
258	15	12664	31500	16.64	5920	1	conhost
516	22	2252	6516	2.09	472	0	csrss
365	20	2296	7024	32.00	592	1	csrss
410	15	3772	20692	0.44	5068	1	ctfmon
237	18	3588	12852	0.17	2072	1	dllhost
310	15	4092	14956	0.06	3424	0	dllhost
865	40	32840	68920	82.89	456	1	dwm
1663	65	29644	108376	10.97	1624	1	explorer
40	9	3604	8424	0.16	880	1	fontdrvhost
40	7	1368	4184	0.00	884	0	fontdrvhost
0	0	60	8	0	0	0	Idle
1209	26	6956	20220	4.17	740	0	lsass
215	14	2000	5368	0.08	4832	0	MicrosoftEdgeUpdate
936	39	51884	21764	19.42	3944	1	mmc
204	13	3536	14216	0.08	5104	0	MoUsoCoreWorker
263	14	2940	11188	0.06	3632	0	msdtc
703	224	250444	205028	6.05	2964	0	MsMpEng
199	39	3720	11336	0.00	2348	0	NisSrv
1463	69	253520	290072	19.11	4980	1	powershell
0	8	2096	62368	0.61	124	0	Registry
217	12	2200	13384	0.08	376	1	RuntimeBroker
190	11	2452	16768	0.06	584	1	RuntimeBroker
423	19	6480	27780	0.92	764	1	RuntimeBroker
258	15	3440	21276	0.27	3336	1	RuntimeBroker
1228	80	89220	151108	1.86	1500	1	SearchApp
218	12	2360	15872	0.08	736	0	SecurityHealthService
768	82	112300	137780	36.02	4448	1	ServerManager
557	14	5532	10240	8.17	712	0	services
622	28	11084	50132	0.22	4400	1	ShellExperienceHost
537	17	5144	27928	1.47	4712	1	sihost
57	4	1096	1260	0.08	344	0	smss
490	24	6512	20076	0.30	2476	0	spoolsv
571	29	17376	61108	0.88	2080	1	StartMenuExperienceHost

2.1.2 List Format (Detailed View)

Use `Format-List` to display processes in a detailed list format:

Get-Process | Format-List

```
PS C:\Users\Administrator> Get-Process | Format-List

Id      : 3316
Handles : 92
CPU     : 0.015625
SI      : 0
Name    : AggregatorHost

Id      : 5920
Handles : 258
CPU     : 16.75
SI      : 1
Name    : conhost

Id      : 472
Handles : 516
CPU     : 2.09375
SI      : 0
Name    : csrss

Id      : 592
Handles : 365
CPU     : 32.03125
SI      : 1
Name    : csrss

Id      : 5068
Handles : 410
CPU     : 0.4375
SI      : 1
Name    : ctfmon

Id      : 2072
Handles : 237
CPU     : 0.171875
SI      : 1
Name    : dllhost
```

2.1.3 Custom Table View

If you only need specific details (like process name & ID), format the table like this:

Get-Process | Format-Table Name, Id, CPU, MemoryUsage -AutoSize

Name	Id	CPU	MemoryUsage
AggregatorHost	3316	0.015625	
conhost	5920	17.5625	
csrss	472	2.09375	
csrss	592	32.125	
ctfmon	5068	0.4375	
dllhost	2072	0.171875	
dllhost	3424	0.0625	
dwm	456	83.71875	
explorer	1624	10.96875	
fontdrvhost	880	0.15625	
fontdrvhost	884	0	
Idle	0		
lsass	740	4.171875	
MicrosoftEdgeUpdate	4832	0.078125	
mmc	3944	19.4375	
MoUsaCoreWorker	5104	0.078125	
msdtc	3632	0.0625	
MsMpEng	2964	6.046875	
NisSrv	2348	0	
powershell	4980	19.25	
Registry	124	0.609375	
RuntimeBroker	376	0.078125	
RuntimeBroker	584	0.0625	
RuntimeBroker	764	0.921875	
RuntimeBroker	3336	0.265625	
SearchApp	1500	1.859375	
SecurityHealthService	736	0.078125	
ServerManager	4448	36.015625	
services	712	8.203125	
ShellExperienceHost	4400	0.21875	
sihost	4712	1.46875	
smss	344	0.078125	
spoolsv	2476	0.296875	
StartMenuExperienceHost	2080	0.875	

2.2 Sort processes based on CPU usage.

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

`Sort-Object -Descending` ensures processes with the highest CPU usage appear at the top.

PS C:\Users\Administrator> Get-Process Sort-Object -Property CPU -Descending							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
2322	0	40	128	166.50	4	0	System
427	20	13356	23468	147.84	2748	0	svchost
865	40	33036	68928	84.20	456	1	dwm
418	21	15456	30104	75.13	3568	0	WmiPrvSE
765	81	112116	137692	36.02	4448	1	ServerManager
366	20	2232	7008	32.20	592	1	csrss
312	20	15660	19796	29.06	6076	0	svchost
404	25	10672	24372	21.19	2684	0	vmtoolsd
393	16	16664	21136	20.59	1120	0	svchost
946	40	51984	21916	19.44	3944	1	mmc
1214	69	218052	254788	19.34	4980	1	powershell
258	15	12664	31500	17.72	5920	1	conhost
1676	66	29776	108164	11.03	1624	1	explorer
539	14	5340	10156	8.20	712	0	services
462	28	20840	41332	6.27	5804	1	vmtoolsd
699	222	249904	203732	6.05	2964	0	MsMpEng
947	18	8656	15516	5.77	964	0	svchost
702	29	12936	27564	4.88	5740	0	svchost
1202	26	6872	20172	4.17	740	0	lsass
1005	21	7252	23856	3.94	852	0	svchost
253	15	3124	8984	3.42	1860	0	svchost
265	14	2660	12316	2.84	672	1	winlogon
409	33	8336	18192	2.81	2100	0	svchost
273	18	2992	14284	2.72	5660	0	svchost
233	11	2392	8260	2.19	1320	0	svchost
502	21	2188	6488	2.09	472	0	csrss
391	19	5172	16348	1.98	1584	0	svchost
555	25	17156	34744	1.89	2548	0	svchost
1228	80	89220	151108	1.86	1500	1	SearchApp
528	23	10176	44224	1.78	2056	1	TextInputHost
420	14	2960	11228	1.75	1888	0	svchost
254	14	3340	13508	1.69	2760	0	svchost
524	16	5052	27872	1.47	4712	1	sihost
359	13	2916	10800	1.34	1008	0	svchost

2.2.1 Additional Customization

If you want a table view for better readability:

Get-Process | Sort-Object -Property CPU -Descending | Format-Table -AutoSize

```
PS C:\Users\Administrator> Get-Process | Sort-Object -Property CPU -Descending | Format-Table -AutoSize
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
2340	0	40	128	166.77	4	0	System
436	20	13308	23692	148.38	2748	0	svchost
871	41	33076	69284	85.64	456	1	dwm
421	21	19920	34628	75.36	3568	0	WmiPrvSE
754	81	116392	141156	36.17	4448	1	ServerManager
366	20	2232	7008	32.30	592	1	csrss
310	20	15832	19916	29.23	6076	0	svchost
404	25	10672	24372	21.19	2684	0	vmtoolsd
393	16	16836	21272	20.77	1120	0	svchost
959	40	52984	22080	19.44	3944	1	mmc
1247	69	208400	245224	19.41	4980	1	powershell
258	15	12912	31504	17.97	5920	1	conhost
1690	66	30180	108612	11.16	1624	1	explorer
537	13	5288	10156	8.27	712	0	services
460	28	20876	41352	6.30	5804	1	vmtoolsd
698	222	249864	204404	6.19	2964	0	MsMpEng
953	18	8752	15616	5.80	964	0	svchost
700	28	12888	27548	4.88	5740	0	svchost
1198	26	6856	20148	4.17	740	0	lsass
1007	21	7380	23900	3.95	852	0	svchost
256	15	3044	8964	3.42	1860	0	svchost
265	13	2660	12316	2.88	672	1	winlogon
413	34	8404	18212	2.81	2100	0	svchost
273	18	2992	14284	2.72	5660	0	svchost
234	11	2328	8240	2.19	1320	0	svchost
504	21	2188	6488	2.09	472	0	csrss
393	19	5220	16356	2.00	1584	0	svchost
555	25	17164	34752	1.89	2548	0	svchost
1228	80	89220	150384	1.86	1500	1	SearchApp
528	23	10176	44224	1.81	2056	1	TextInputHost
420	14	2888	11208	1.75	1888	0	svchost
254	14	3340	13508	1.69	2760	0	svchost
524	16	5048	27868	1.47	4712	1	sihost
358	13	2916	10800	1.34	1008	0	svchost

To show only specific properties (like Name, ID, and CPU usage):

Get-Process | Sort-Object -Property CPU -Descending | Select-Object Name, Id, CPU | Format-Table -AutoSize

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> Get-Process | Sort-Object -Property CPU -Descending | Select-Object Name, Id, CPU | Format-Table -AutoSize
```

Name	Id	CPU
System	4	166.765625
svchost	2748	148.640625
dwm	456	86.3125
WmiPrvSE	3568	75.40625
ServerManager	4448	36.171875
csrss	592	32.375
svchost	6076	29.234375
vmtoolsd	2684	21.1875
svchost	1120	20.765625
powershell	4980	19.515625
mmc	3944	19.4375
conhost	5920	18.15625
explorer	1624	11.15625
services	712	8.265625
vmtoolsd	5804	6.296875
MsMpEng	2964	6.203125
svchost	964	5.796875
svchost	5740	4.875
lsass	740	4.171875
svchost	852	3.953125
svchost	1860	3.4375
winlogon	672	2.875
svchost	2100	2.8125
svchost	5660	2.71875
svchost	1320	2.1875
csrss	472	2.09375
svchost	1584	2
svchost	2548	1.90625
SearchApp	1500	1.859375
TextInputHost	2056	1.8125
svchost	1888	1.75
svchost	2760	1.6875
sihost	4712	1.46875
svchost	1008	1.34375

2.3 Select and filter objects based on specific conditions.

2.3.1 Select Specific Properties

If you want to display only certain attributes of an object, use:

```
Get-Process | Select-Object Name, Id, CPU
```

This will show only the Name, ID, and CPU usage of each process.

PS C:\Users\Administrator> Get-Process Select-Object Name, Id, CPU		
Name	Id	CPU
AggregatorHost	3316	0.015625
conhost	5920	18.328125
csrss	472	2.09375
csrss	592	32.4375
ctfmon	5068	0.4375
dllhost	2072	0.171875
dllhost	3424	0.0625
dwm	456	87.203125
explorer	1624	11.15625
fontdrvhost	880	0.15625
fontdrvhost	884	0
Idle	0	
lsass	740	4.171875
MicrosoftEdgeUpdate	4832	0.078125
mmc	3944	19.5
MoUsCoreWorker	5104	0.078125
msdtc	3632	0.0625
MsMpEng	2964	6.3125
NisSrv	2348	0
powershell	4980	19.5625
Registry	124	0.609375
RuntimeBroker	376	0.078125
RuntimeBroker	584	0.0625
RuntimeBroker	764	0.921875
RuntimeBroker	3336	0.265625
SearchApp	1500	1.859375
SecurityHealthService	736	0.078125
ServerManager	4448	36.234375
services	712	8.265625
ShellExperienceHost	4400	0.21875
sihost	4712	1.46875
smss	344	0.078125
spoolsv	2476	0.296875
StartMenuExperienceHost	2080	0.875

2.3.2 Filter Objects Based on a Condition

If you need to filter objects, use `Where-Object` . For example, to list processes consuming more than 50% CPU, run:

```
Get-Process | Where-Object { $_.CPU -gt 50 }
```

```
PS C:\Users\Administrator> Get-Process | Where-Object { $_.CPU -gt 50 }

Handles  NPM(K)    PM(K)    WS(K)    CPU(s)   Id  SI ProcessName
-----  -----    -----    -----    -----  --  --  -----
    871      40     33444    69264    87.81  456  1  dwm
    409      19     13760    24156   151.19  2748  0  svchost
   2313       0      40      128    167.83    4  0  System
    459      24     21908    37720    76.39  3568  0  WmiPrvSE

PS C:\Users\Administrator>
```

2.3.3 Filter Objects Using Multiple Conditions

To filter objects using multiple conditions, for example, selecting processes with CPU > 50 AND Memory usage > 100MB, use:

```
Get-Process | Where-Object { $_.CPU -gt 50 -and $_.WS -gt 100 }
```

```
PS C:\Users\Administrator> Get-Process | Where-Object { $_.CPU -gt 50 -and $_.WS -gt 100 }

Handles  NPM(K)    PM(K)    WS(K)    CPU(s)   Id  SI ProcessName
-----  -----    -----    -----    -----  --  --  -----
    865      41     32812    68920    88.23  456  1  dwm
    411      19     13584    23896   151.28  2748  0  svchost
   2313       0      40      128    167.88    4  0  System
    421      21     20496    34160    76.50  3568  0  WmiPrvSE

PS C:\Users\Administrator>
```

2.3.4 Select Top N Results

If you want only the top 5 highest CPU-consuming processes, use:

```
Get-Process | Sort-Object -Property CPU -Descending | Select-Object -First 5
```

```
PS C:\Users\Administrator> Get-Process | Sort-Object -Property CPU -Descending | Select-Object -First 5

Handles  NPM(K)    PM(K)    WS(K)    CPU(s)   Id  SI ProcessName
-----  -----    -----    -----    -----  --  --  -----
  2340       0      40      128    168.06    4  0  System
   428      20     13484    23588   151.53  2748  0  svchost
   867      41     32892    69000    89.13  456  1  dwm
   420      21     20344    33976    76.73  3568  0  WmiPrvSE
   765      81    117548   142136   36.47  4448  1  ServerManager
```

2.4 Loop through system objects and extract specific information.

To loop through system objects and extract specific information in PowerShell, use a `ForEach-Object` loop or a `foreach` statement.

2.4.1 Using `ForEach-Object` (Pipeline)

You can iterate over objects in a pipeline and extract specific properties. For example, listing all running processes with their name and CPU usage:

```
Get-Process | ForEach-Object { "$($_.Name) - CPU: $($_.CPU)" }
```

```
PS C:\Users\Administrator> Get-Process | ForEach-Object { "$($_.Name) - CPU: $($_.CPU)" }
AggregatorHost - CPU: 0.015625
conhost - CPU: 18.859375
csrss - CPU: 2.140625
csrss - CPU: 32.828125
ctfmon - CPU: 0.4375
dllhost - CPU: 0.171875
dllhost - CPU: 0.0625
dwm - CPU: 89.78125
explorer - CPU: 11.25
fontdrvhost - CPU: 0.15625
fontdrvhost - CPU: 0
Idle - CPU:
lsass - CPU: 4.21875
MicrosoftEdgeUpdate - CPU: 0.078125
mmc - CPU: 19.625
MoUsoCoreWorker - CPU: 0.078125
msdtc - CPU: 0.0625
MsMpEng - CPU: 6.53125
NisSrv - CPU: 0
powershell - CPU: 19.890625
Registry - CPU: 0.609375
RuntimeBroker - CPU: 0.078125
RuntimeBroker - CPU: 0.0625
RuntimeBroker - CPU: 0.921875
RuntimeBroker - CPU: 0.265625
SearchApp - CPU: 1.859375
SecurityHealthService - CPU: 0.078125
ServerManager - CPU: 36.578125
services - CPU: 8.40625
ShellExperienceHost - CPU: 0.21875
sihost - CPU: 1.46875
smss - CPU: 0.078125
spoolsv - CPU: 0.296875
StartMenuExperienceHost - CPU: 0.875
svchost - CPU: 3.953125
svchost - CPU: 5.84375
svchost - CPU: 1.03125
```

2.4.2 Using a `foreach` Loop

For more control, store the objects in a variable and loop through them:

```
$processes = Get-Process
```

```
foreach ($process in $processes) {
```

```
        Write-Host "Process: $($process.Name) - ID: $($process.Id) - CPU: $($process.CPU)"  
    }  
  
}
```

```
PS C:\Users\Administrator> $processes = Get-Process  
PS C:\Users\Administrator> foreach ($process in $processes) {  
>>     Write-Host "Process: $($process.Name) - ID: $($process.Id) - CPU: $($process.CPU)"  
>> }  
Process: AggregatorHost - ID: 3316 - CPU: 0.015625  
Process: conhost - ID: 5920 - CPU: 18.984375  
Process: csrss - ID: 472 - CPU: 2.140625  
Process: csrss - ID: 592 - CPU: 32.828125  
Process: ctfmon - ID: 5068 - CPU: 0.46875  
Process: dllhost - ID: 2072 - CPU: 0.171875  
Process: dllhost - ID: 3424 - CPU: 0.0625  
Process: dwm - ID: 456 - CPU: 89.984375  
Process: explorer - ID: 1624 - CPU: 11.265625  
Process: fontdrvhost - ID: 880 - CPU: 0.15625  
Process: fontdrvhost - ID: 884 - CPU: 0  
Process: Idle - ID: 0 - CPU:  
Process: lsass - ID: 740 - CPU: 4.234375  
Process: MicrosoftEdgeUpdate - ID: 4832 - CPU: 0.078125  
Process: mmc - ID: 3944 - CPU: 19.625  
Process: MoUsaCoreWorker - ID: 5104 - CPU: 0.078125  
Process: msdtc - ID: 3632 - CPU: 0.0625  
Process: MsMpEng - ID: 2964 - CPU: 6.59375  
Process: NisSrv - ID: 2348 - CPU: 0  
Process: powershell - ID: 4980 - CPU: 19.9375  
Process: Registry - ID: 124 - CPU: 0.609375  
Process: RuntimeBroker - ID: 376 - CPU: 0.078125  
Process: RuntimeBroker - ID: 584 - CPU: 0.0625  
Process: RuntimeBroker - ID: 764 - CPU: 0.921875  
Process: RuntimeBroker - ID: 3336 - CPU: 0.265625  
Process: SearchApp - ID: 1500 - CPU: 1.859375  
Process: SecurityHealthService - ID: 736 - CPU: 0.078125  
Process: ServerManager - ID: 4448 - CPU: 36.578125  
Process: services - ID: 712 - CPU: 8.40625  
Process: ShellExperienceHost - ID: 4400 - CPU: 0.21875  
Process: sihost - ID: 4712 - CPU: 1.46875  
Process: smss - ID: 344 - CPU: 0.078125  
Process: spoolsv - ID: 2476 - CPU: 0.296875  
Process: StartMenuExperienceHost - ID: 2080 - CPU: 0.875
```

2.4.3 Loop Through Services and Extract Specific Details

To list services that are running, with their status and display name:

```
Get-Service | Where-Object { $_.Status -eq "Running" } | ForEach-Object {  
    "$($_.DisplayName) - Status: $($_.Status)" }
```

```
PS C:\Users\Administrator> Get-Service | Where-Object { $_.Status -eq "Running" } | ForEach-Object { "$($_.DisplayName) - Status: $($_.Status)" }
Base Filtering Engine - Status: Running
Background Tasks Infrastructure Service - Status: Running
Capability Access Manager Service - Status: Running
Clipboard User Service_45038 - Status: Running
Connected Devices Platform Service - Status: Running
Connected Devices Platform User Service_45038 - Status: Running
Certificate Propagation - Status: Running
COM+ System Application - Status: Running
CoreMessaging - Status: Running
Cryptographic Services - Status: Running
DCOM Server Process Launcher - Status: Running
DHCP Client - Status: Running
Connected User Experiences and Telemetry - Status: Running
Display Policy Service - Status: Running
DNS Client - Status: Running
Diagnostic Policy Service - Status: Running
Data Sharing Service - Status: Running
Windows Event Log - Status: Running
COM+ Event System - Status: Running
Windows Font Cache Service - Status: Running
Group Policy Client - Status: Running
IP Helper - Status: Running
CNG Key Isolation - Status: Running
Server - Status: Running
Workstation - Status: Running
Windows License Manager Service - Status: Running
TCP/IP NetBIOS Helper - Status: Running
Local Session Manager - Status: Running
Windows Defender Firewall - Status: Running
Distributed Transaction Coordinator - Status: Running
Network Connection Broker - Status: Running
Network Connections - Status: Running
Network List Service - Status: Running
Network Setup Service - Status: Running
Network Location Awareness - Status: Running
Network Store Interface Service - Status: Running
Program Compatibility Assistant Service - Status: Running
```

3 Task 3: Managing the File System

3.1 Navigate through directories and list files.

You can navigate through directories and list files in PowerShell using these commands:

3.1.1 Change Directory (`cd` or `Set-Location`)

To move into a different directory:

Set-Location C:\Users\Administrator\Documents

or simply:

cd C:\Users\Administrator\Documents

```
PS C:\Users\Administrator> Set-Location C:\Users\Administrator\Documents
PS C:\Users\Administrator\Documents> cd C:\Users\Administrator\
PS C:\Users\Administrator> cd C:\Users\Administrator\Documents\
PS C:\Users\Administrator\Documents> ■
```

3.1.2 List Files in a Directory (`Get-ChildItem`)

To display files and folders in the current directory:

Get-ChildItem

```
PS C:\Users\Administrator> Get-ChildItem

Directory: C:\Users\Administrator

Mode                LastWriteTime      Length Name
----              <-----           ----- 
d-r---        4/29/2025 9:18 PM          3D Objects
d-r---        4/29/2025 9:18 PM          Contacts
d-r---        4/30/2025 7:51 PM          Desktop
d-r---        4/29/2025 9:18 PM          Documents
d-r---        4/29/2025 9:18 PM          Downloads
d-r---        4/29/2025 9:18 PM          Favorites
d-r---        4/29/2025 9:18 PM          Links
d-r---        4/29/2025 9:18 PM          Music
d-r---        4/29/2025 9:18 PM          Pictures
d-r---        4/29/2025 9:18 PM          Saved Games
d-r---        4/29/2025 9:18 PM          Searches
d-r---        4/29/2025 9:18 PM          Videos

PS C:\Users\Administrator>
```

or specify a directory:

```
Get-ChildItem C:\Users\Administrator\
```

```

PS C:\Users\Administrator\Documents> cd C:\Users\Administrator
PS C:\Users\Administrator> Get-ChildItem C:\Users\Administrator

Directory: C:\Users\Administrator

Mode                LastWriteTime      Length Name
----              -----          ----
d-r---        4/29/2025 9:18 PM           3D Objects
d-r---        4/29/2025 9:18 PM           Contacts
d-r---        4/30/2025 7:51 PM           Desktop
d-r---        4/29/2025 9:18 PM           Documents
d-r---        4/29/2025 9:18 PM           Downloads
d-r---        4/29/2025 9:18 PM           Favorites
d-r---        4/29/2025 9:18 PM           Links
d-r---        4/29/2025 9:18 PM           Music
d-r---        4/29/2025 9:18 PM           Pictures
d-r---        4/29/2025 9:18 PM           Saved Games
d-r---        4/29/2025 9:18 PM           Searches
d-r---        4/29/2025 9:18 PM           Videos

```

3.1.3 Show Only Files (Exclude Folders)

To list only files, excluding directories:

Get-ChildItem -File

```

PS C:\Program Files\Common Files\System> cd 'C:\Program Files\Common Files\System\' 
PS C:\Program Files\Common Files\System>
PS C:\Program Files\Common Files\System>
PS C:\Program Files\Common Files\System> Get-ChildItem -File

```

```

Directory: C:\Program Files\Common Files\System

Mode                LastWriteTime      Length Name
----              -----          ----
-a---        7/7/2023  5:22 PM       958464 wab32.dll
-a---        7/7/2023  5:22 PM       974848 wab32res.dll

```

3.1.4 Show Only Folders (Exclude Files)

To list only directories, excluding files:

Get-ChildItem -Directory

```
PS C:\Program Files\Common Files\System> Get-ChildItem -Directory

Directory: C:\Program Files\Common Files\System

Mode                LastWriteTime      Length Name
----              LastWriteTime      Length Name
---
d-----        5/8/2021    5:36 AM
d-----        5/8/2021    5:36 AM
d-----        5/8/2021    5:36 AM
d-----        7/7/2023    5:27 PM

PS C:\Program Files\Common Files\System>
```

3.1.5 List Files with Specific Extensions

To find only *.dll files, for example:

Get-ChildItem -Path C:\Users\Administrator\Documents -Filter *.dll

```
PS C:\Program Files\Common Files\System> Get-ChildItem -Path 'C:\Program Files\Common Files\System\' -Filter *.dll

Directory: C:\Program Files\Common Files\System

Mode                LastWriteTime      Length Name
----              LastWriteTime      Length Name
---
-a----        7/7/2023    5:22 PM     958464 wab32.dll
-a----        7/7/2023    5:22 PM     974848 wab32res.dll

PS C:\Program Files\Common Files\System> ■
```

3.1.6 Navigate Back to Previous Directory

To return to the previous directory:

cd ..

or

Set-Location ..

```
PS C:\Program Files\Common Files\System> cd ..  
PS C:\Program Files\Common Files> Set-Location ..  
PS C:\Program Files>
```

3.2 Create and delete files and folders

You can create and delete files and folders using commands.

3.2.1 Create a New File

To create a new file, use:

New-Item -Path "C:\Users\Administrator\Documents\example.txt" -ItemType File

```
PS C:\Program Files>  
PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\example.txt" -ItemType File
```

```
Directory: C:\Users\Administrator\Documents
```

Mode	LastWriteTime	Length	Name
-a---	5/1/2025 12:48 AM	0	example.txt

```
PS C:\Program Files>
```

3.2.2 Create a New Folder

To create a directory:

New-Item -Path "C:\Users\Administrator\Documents\NewFolder" -ItemType Directory

This creates "NewFolder" inside Documents .

```
PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\NewFolder" -ItemType Directory
```

```
Directory: C:\Users\Administrator\Documents
```

Mode	LastWriteTime	Length	Name
d----	5/1/2025 12:48 AM		NewFolder

```
PS C:\Program Files>
```

3.2.3 Delete a File

To remove a file:

```
Remove-Item -Path "C:\Users\Administrator\Documents\example.txt"
```

Add ` -Force` to bypass confirmation if needed.

```
PS C:\Program Files> Remove-Item -Path "C:\Users\Administrator\Documents\example.txt"
PS C:\Program Files> Get-ChildItem -Path C:\Users\Administrator\Documents
[...]
Directory: C:\Users\Administrator\Documents

Mode          LastWriteTime      Length Name
----          -----          ---- 
d----  5/1/2025 12:48 AM           NewFolder

PS C:\Program Files> ■
```

3.2.4 Delete a Folder

3.2.4.1 *To remove an empty folder:*

```
Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder"
```

```
PS C:\Program Files> Get-ChildItem -Path C:\Users\Administrator\Documents
[...]
Directory: C:\Users\Administrator\Documents

Mode          LastWriteTime      Length Name
----          -----          ---- 
d----  5/1/2025 12:48 AM           NewFolder

PS C:\Program Files> Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder"
PS C:\Program Files> Get-ChildItem -Path C:\Users\Administrator\Documents
PS C:\Program Files> ■
```

3.2.4.2 To delete a folder with contents, use:

Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder" -Recurse -Force

```
PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\NewFolder" -ItemType Directory

    Directory: C:\Users\Administrator\Documents

Mode           LastWriteTime         Length Name
----           -----          ----- -
d---           5/1/2025 12:52 AM            NewFolder

PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\NewFolder\example2.txt" -ItemType File

    Directory: C:\Users\Administrator\Documents\NewFolder

Mode           LastWriteTime         Length Name
----           -----          ----- -
-a---          5/1/2025 12:54 AM            0 example2.txt

PS C:\Program Files> Get-ChildItem -Path C:\Users\Administrator\Documents\NewFolder\

    Directory: C:\Users\Administrator\Documents\NewFolder

Mode           LastWriteTime         Length Name
----           -----          ----- -
-a---          5/1/2025 12:54 AM            0 example2.txt

PS C:\Program Files> Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder" -Recurse -Force ←
PS C:\Program Files> Get-ChildItem -Path C:\Users\Administrator\Documents
PS C:\Program Files>
```

3.3 Copy and move files between locations

3.3.1 Prepare

1. Create a new folder called "SourceFolder" inside
C:\Users\Administrator\Documents

New-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -ItemType Directory

This ensures the directory exists before copying or moving files.

2. Create a New File
Create a new text file inside SourceFolder

New-Item -Path

"C:\Users\Administrator\Documents\SourceFolder\example.txt" - ItemType File

This creates an empty file that can later be edited or copied.

```
PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -ItemType Directory

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime        Length Name
----                -----          ----- 
d-----      5/1/2025  1:17 AM           SourceFolder

PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -ItemType File

Directory: C:\Users\Administrator\Documents\SourceFolder

Mode                LastWriteTime        Length Name
----                -----          ----- 
-a----      5/1/2025  1:17 AM            0 example.txt

PS C:\Program Files> ■
```

3. This creates an empty file that can later be edited or copied.

create destination Folder

New-Item -Path "C:\Users\Administrator\Documents\DestinationFolder" - ItemType Directory

```
PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\DestinationFolder" -ItemType Directory

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime        Length Name
----                -----          ----- 
d-----      5/1/2025  1:21 AM           DestinationFolder

PS C:\Program Files> ■
```

4. This creates an empty folder that can later be copied or moved

New-Item -Path "C:\Users\Administrator\Documents\OldFolder"

```
Administrator: Windows PowerShell
PS C:\Program Files> New-Item -Path "C:\Users\Administrator\Documents\oldFolder"

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime      Length Name
----                -----          ---- 
-a----  5/1/2025 1:35 AM           0 OldFolder

PS C:\Program Files>
```

3.3.2 Copy a File

Copy example.txt from SourceFolder to DestinationFolder

Copy-Item -Path

```
"C:\Users\Administrator\Documents\SourceFolder\example.txt" -Destination
"C:\Users\Administrator\Documents\DestinationFolder\example.txt"
```

The file remains in SourceFolder while being duplicated in DestinationFolder.

```
Administrator: Windows PowerShell
PS C:\Program Files> Copy-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -Destination "C:\Users\Administrator\Documents\DestinationFolder\example.txt"
PS C:\Program Files>
```

3.3.3 Copy an Entire Folder

Copy SourceFolder and all its contents to BackupFolder

Copy-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -
Destination "C:\Users\Administrator\Documents\BackupFolder" -Recurse

The ` -Recurse` flag ensures that all files and subfolders are copied.

```
Administrator: Windows PowerShell
PS C:\Program Files>
PS C:\Program Files> Copy-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -Destination "C:\Users\Administrator\Documents\BackupFolder" -Recurse
PS C:\Program Files>
```

3.3.4 Move a File

Move example.txt from SourceFolder to DestinationFolder

Move-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -
Destination "C:\Users\Administrator\Documents\DestinationFolder\example.txt" -
Force

Unlike copying, this removes the file from SourceFolder and places it in DestinationFolder.

```
PS C:\Program Files> Move-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -Destination "C:\Users\Administrator\Documents\DestinationFolder\example.txt" -Force
$ C:\Program Files>
```

3.3.5 Move an Entire Folder

Move OldFolder and all contents into NewFolder

Move-Item -Path "C:\Users\Administrator\Documents\OldFolder" -Destination "C:\Users\Administrator\Documents\NewFolder"

```
PS C:\Program Files> Move-Item -Path "C:\Users\Administrator\Documents\OldFolder" -Destination "C:\Users\Administrator\Documents\NewFolder"
PS C:\Program Files> -
```

The folder and its files will be transferred, leaving OldFolder empty or deleted.

```
PS C:\Users\Administrator\Documents> cd C:\Users\Administrator\Documents\
PS C:\Users\Administrator\Documents> Get-ChildItem

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime      Length Name
----                -----          ---- 
d-----        5/1/2025  1:23 AM           BackupFolder
d-----        5/1/2025  1:28 AM           DestinationFolder
d-----        5/1/2025  1:17 AM           SourceFolder
-a----        5/1/2025  1:35 AM            0 NewFolder

PS C:\Users\Administrator\Documents>
```

3.3.6 Restore

1. Delete a File

Delete example.txt from DestinationFolder

**Remove-Item -Path
"C:\Users\Administrator\Documents\DestinationFolder\example.txt"**

Use ` -Force` if you want to delete without confirmation.

2. Delete a Folder (With Contents)

Delete BackupFolder and all its files

Remove-Item -Path

"C:\Users\Administrator\Documents\DestinationFolder\example.txt"

Remove-Item -Path "C:\Users\Administrator\Documents\BackupFolder" -Recurse -Force

Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder" -Recurse -Force

Remove-Item -Path "C:\Users\Administrator\Documents\DestinationFolder" -Recurse -Force

Remove-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -Recurse -Force

This permanently removes the folder and its contents.

```
PS C:\Users\Administrator\Documents>
PS C:\Users\Administrator\Documents> cd C:\Users\Administrator\Documents\
PS C:\Users\Administrator\Documents> Get-ChildItem

    Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime      Length Name
----              -----          ---- 
d----       5/1/2025  1:23 AM           BackupFolder
d----       5/1/2025  1:28 AM           DestinationFolder
d----       5/1/2025  1:17 AM           SourceFolder
-a---       5/1/2025  1:35 AM            0 NewFolder

PS C:\Users\Administrator\Documents> Remove-Item -Path "C:\Users\Administrator\Documents\DestinationFolder\example.txt"
PS C:\Users\Administrator\Documents> Remove-Item -Path "C:\Users\Administrator\Documents\BackupFolder" -Recurse -Force
PS C:\Users\Administrator\Documents> Get-ChildItem

    Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime      Length Name
----              -----          ---- 
d----       5/1/2025  1:28 AM           DestinationFolder
d----       5/1/2025  1:17 AM           SourceFolder
-a---       5/1/2025  1:35 AM            0 NewFolder

PS C:\Users\Administrator\Documents> Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder" -Recurse -Force
PS C:\Users\Administrator\Documents> Remove-Item -Path "C:\Users\Administrator\Documents\DestinationFolder" -Recurse -Force
PS C:\Users\Administrator\Documents> Remove-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -Recurse -Force
PS C:\Users\Administrator\Documents> Get-ChildItem
PS C:\Users\Administrator\Documents>
```

3.4 Check available disk space.

1. Get available disk space for the C: drive

Get-PSDrive C

2. Retrieve detailed volume information, including free space

Get-Volume | Select-Object DriveLetter, SizeRemaining, Size

```
PS C:\Users\Administrator\Documents>
PS C:\Users\Administrator\Documents> Get-PSDrive C

Name          Used (GB)    Free (GB) Provider      Root
----          -----        -----       FileSystem   C:\

PS C:\Users\Administrator\Documents> Get-Volume | Select-Object DriveLetter, SizeRemaining, Size

DriveLetter  SizeRemaining      Size
-----        -----
C           93324333056 107028148224
                  175202304  205520896
D           0                0
A           0                0

PS C:\Users\Administrator\Documents>
```

4 Task 4: Managing System Services and Processes

4.1 List all running services on the system.

Get-Service | Where-Object { \$_.Status -eq "Running" }

```
PS C:\Users\Administrator\Documents> Get-Service | Where-Object { $_.Status -eq "Running" }

Status    Name          DisplayName
----      --          -----
Running   BFE          Base Filtering Engine
Running   BrokerInfrastru... Background Tasks Infrastructure Ser...
Running   camsvc        Capability Access Manager Service
Running   cbdhsvc_45038 Clipboard User Service_45038
Running   CDPSvc        Connected Devices Platform Service
Running   CDPUUserSvc_45038 Connected Devices Platform User Ser...
Running   CertPropSvc   Certificate Propagation
Running   COMSysApp     COM+ System Application
Running   CoreMessagingRe... CoreMessaging
Running   CryptSvc       Cryptographic Services
Running   DcomLaunch    DCOM Server Process Launcher
Running   Dhcp          DHCP Client
Running   DiagTrack     Connected User Experiences and Tele...
Running   DispBrokerDeskt... Display Policy Service
Running   Dnscache      DNS Client
Running   DPS           Diagnostic Policy Service
Running   DsSvc          Data Sharing Service
Running   EventLog       Windows Event Log
Running   EventSystem    COM+ Event System
Running   FontCache     Windows Font Cache Service
Running   gpsvc         Group Policy Client
Running   iphlpsvc     IP Helper
Running   KeyIso        CNG Key Isolation
Running   LanmanServer   Server
Running   LanmanWorkstation Workstation
Running   LicenseManager Windows License Manager Service
Running   lmhosts        TCP/IP NetBIOS Helper
Running   LSM            Local Session Manager
Running   mpssvc        Windows Defender Firewall
Running   MSDTC          Distributed Transaction Coordinator
Running   NcbService    Network Connection Broker
Running   Netman         Network Connections
Running   netprofm      Network List Service
Running   NlaSvc        Network Location Awareness
```

4.2 Start and stop specific services.

1 Look for service spooler

Get-Service -Name "Spooler"

2 Stop a specific service (Spooler)

Stop-Service -Name "Spooler"

3 Start a specific service (Spooler)

Start-Service -Name "Spooler"

4 Stop a specific service (Spooler)

Stop-Service -Name "Spooler"

5 Restore and Retrieve information about the Spooler service

Start-Service -Name "Spooler"

Get-Service -Name "Spooler"

```
PS C:\Users\Administrator\Documents> Get-Service -Name "Spooler"

Status    Name          DisplayName
----      --          -----
Running   Spooler      Print Spooler


PS C:\Users\Administrator\Documents> Stop-Service -Name "Spooler"
PS C:\Users\Administrator\Documents> Start-Service -Name "Spooler"
PS C:\Users\Administrator\Documents> Stop-Service -Name "Spooler"
PS C:\Users\Administrator\Documents> Start-Service -Name "Spooler"
PS C:\Users\Administrator\Documents> Get-Service -Name "Spooler"

Status    Name          DisplayName
----      --          -----
Running   Spooler      Print Spooler


PS C:\Users\Administrator\Documents>
```

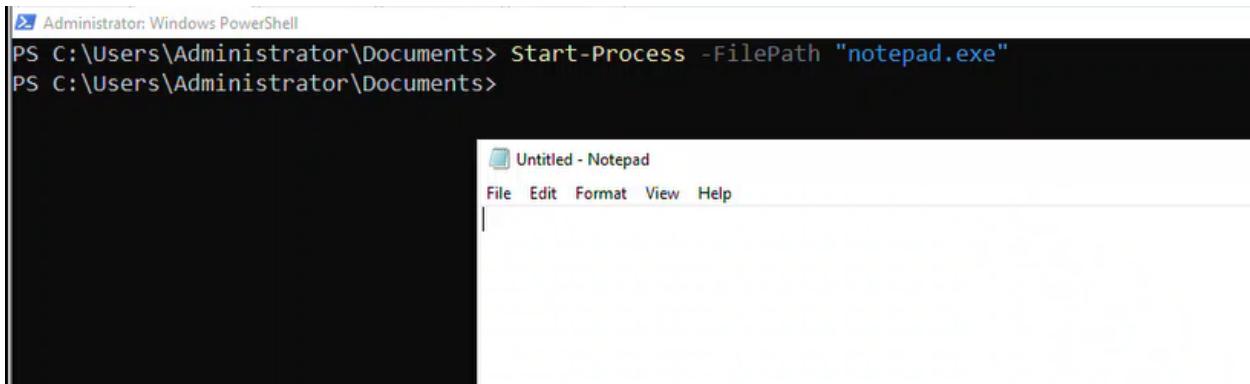
4.3 Retrieve information about active processes.

Get-Process

Get-Process							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
92	6	904	4744	0.02	3316	0	AggregatorHost
258	15	13032	31692	22.48	5920	1	conhost
511	22	2268	6560	2.38	472	0	csrss
379	21	2312	7072	37.64	592	1	csrss
410	16	3816	20772	0.55	5068	1	ctfmon
237	18	3604	12912	0.19	2072	1	dllhost
310	15	4092	14980	0.06	3424	0	dllhost
857	38	32860	76360	116.22	456	1	dwm
1673	65	30208	108728	12.81	1624	1	explorer
40	9	3604	8440	0.16	880	1	fontdrvhost
40	7	1376	4204	0.00	884	0	fontdrvhost
0	0	60	8		0	0	Idle
1204	26	6836	20168	4.44	740	0	lsass
215	14	2068	5460	0.08	4832	0	MicrosoftEdgeUpdate
653	40	51984	9340	20.13	3944	1	mmc
204	14	3536	14228	0.08	5104	0	MoUsoCoreWorker
263	14	2940	11200	0.06	3632	0	msdtc
614	224	259212	216560	11.20	2964	0	MsMpEng
199	39	3728	11380	0.02	2348	0	NisSrv
1450	72	280968	320180	22.91	4980	1	powershell
0	8	2172	67004	0.63	124	0	Registry
215	12	2200	13428	0.08	376	1	RuntimeBroker
190	12	2520	16812	0.06	584	1	RuntimeBroker
423	19	6588	27752	0.97	764	1	RuntimeBroker
254	14	3232	21272	0.27	3336	1	RuntimeBroker
1274	80	90272	149404	1.91	1500	1	SearchApp
218	12	2360	15896	0.08	736	0	SecurityHealthService
760	81	119524	144132	38.38	4448	1	ServerManager
548	14	5504	10276	9.31	712	0	services
622	28	11084	50132	0.22	4400	1	ShellExperienceHost
526	17	5080	27952	1.50	4712	1	sihost
57	4	1096	1260	0.08	344	0	sms
480	23	6060	16928	0.11	6660	0	spoolsv
566	29	17524	61448	0.94	2080	1	StartMenuExperienceHost

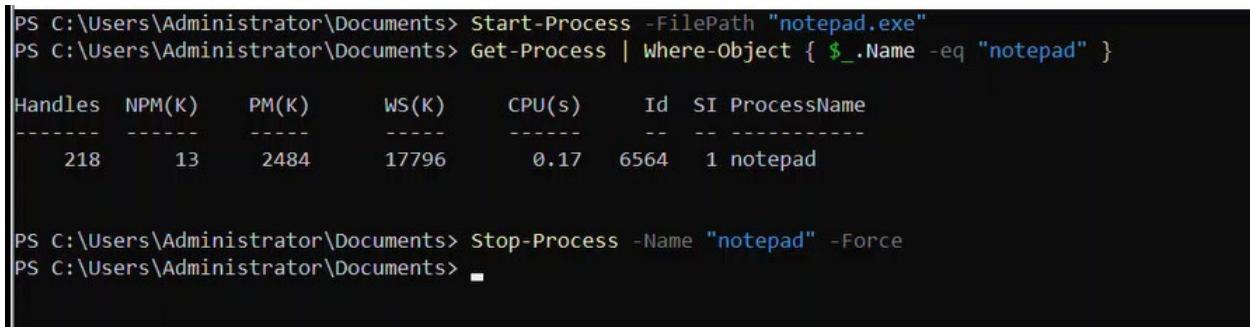
4.4 Terminate a process

```
Start-Process -FilePath "notepad.exe"  
Get-Process | Where-Object { $_.Name -eq "notepad" }  
Stop-Process -Name "notepad" -Force
```



```
PS C:\Users\Administrator\Documents> Start-Process -FilePath "notepad.exe"  
PS C:\Users\Administrator\Documents>
```

A screenshot of a Windows PowerShell window. The command `Start-Process -FilePath "notepad.exe"` is run, followed by `Get-Process | Where-Object { \$_.Name -eq "notepad" }` to find the process, and finally `Stop-Process -Name "notepad" -Force` to terminate it. In the background, a Notepad window titled "Untitled - Notepad" is open.



```
PS C:\Users\Administrator\Documents> Start-Process -FilePath "notepad.exe"  
PS C:\Users\Administrator\Documents> Get-Process | Where-Object { $_.Name -eq "notepad" }  

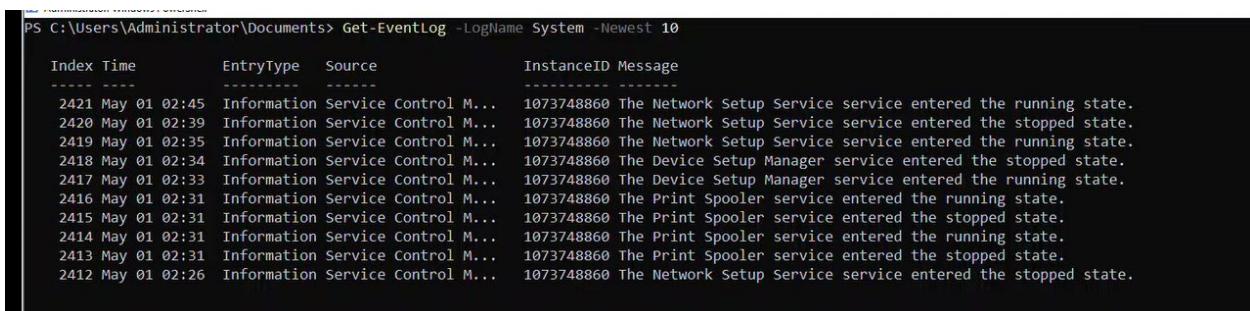

| Handles | NPM(K) | PM(K) | WS(K) | CPU(s) | Id   | SI | ProcessName |
|---------|--------|-------|-------|--------|------|----|-------------|
| 218     | 13     | 2484  | 17796 | 0.17   | 6564 | 1  | notepad     |

  
PS C:\Users\Administrator\Documents> Stop-Process -Name "notepad" -Force  
PS C:\Users\Administrator\Documents>
```

5 Task 5: Monitoring Event Logs and System Information

5.1 View the latest system event logs.

Get-EventLog -LogName System -Newest 10



```
PS C:\Users\Administrator\Documents> Get-EventLog -LogName System -Newest 10
```

Index	Time	EntryType	Source	InstanceID	Message
2421	May 01 02:45	Information	Service Control Manager	1073748860	The Network Setup Service service entered the running state.
2420	May 01 02:39	Information	Service Control Manager	1073748860	The Network Setup Service service entered the stopped state.
2419	May 01 02:35	Information	Service Control Manager	1073748860	The Network Setup Service service entered the running state.
2418	May 01 02:34	Information	Service Control Manager	1073748860	The Device Setup Manager service entered the stopped state.
2417	May 01 02:33	Information	Service Control Manager	1073748860	The Device Setup Manager service entered the running state.
2416	May 01 02:31	Information	Service Control Manager	1073748860	The Print Spooler service entered the running state.
2415	May 01 02:31	Information	Service Control Manager	1073748860	The Print Spooler service entered the stopped state.
2414	May 01 02:31	Information	Service Control Manager	1073748860	The Print Spooler service entered the running state.
2413	May 01 02:31	Information	Service Control Manager	1073748860	The Print Spooler service entered the stopped state.
2412	May 01 02:26	Information	Service Control Manager	1073748860	The Network Setup Service service entered the stopped state.

Get-EventLog -LogName System | Select-Object -First 20 | Format-List *

```
PS C:\Users\Administrator\Documents> Get-EventLog -LogName System | Select-Object -First 20 | Format-List *

EventID      : 7036
MachineName  : DC101
Data         : {78, 0, 101, 0...}
Index        : 2422
Category     : (0)
CategoryNumber : 0
EntryType    : Information
Message      : The Network Setup Service service entered the stopped state.
Source       : Service Control Manager
Replacementstrings : {Network Setup Service, stopped}
InstanceId   : 1073748860
TimeGenerated : 5/1/2025 2:46:42 AM
TimeWritten   : 5/1/2025 2:46:42 AM
UserName     :
Site         :
Container    :

EventID      : 7036
MachineName  : DC101
Data         : {78, 0, 101, 0...}
Index        : 2421
Category     : (0)
CategoryNumber : 0
EntryType    : Information
Message      : The Network Setup Service service entered the running state.
Source       : Service Control Manager
Replacementstrings : {Network Setup Service, running}
InstanceId   : 1073748860
TimeGenerated : 5/1/2025 2:45:09 AM
TimeWritten   : 5/1/2025 2:45:09 AM
UserName     :
```

5.2 Retrieve security event logs.

Get-EventLog -LogName Security -Newest 10

```
Get-EventLog -LogName Security | Where-Object { $_.EventID -eq 4625 }
```

** Event ID **4625** corresponds to **failed login attempts**

```

PS C:\Users\Administrator\Documents> Get-EventLog -LogName Security -Newest 10
Index Time          EntryType   Source           InstanceID Message
---- --          -----   -----           -----   -----
1737 May 01 02:45 SuccessA... Microsoft-Windows...        4672 Special privileges assigned to new logon....
1736 May 01 02:45 SuccessA... Microsoft-Windows...        4624 An account was successfully logged on....
1735 May 01 02:35 SuccessA... Microsoft-Windows...        4672 Special privileges assigned to new logon....
1734 May 01 02:35 SuccessA... Microsoft-Windows...        4624 An account was successfully logged on....
1733 May 01 02:33 SuccessA... Microsoft-Windows...        4672 Special privileges assigned to new logon....
1732 May 01 02:33 SuccessA... Microsoft-Windows...        4624 An account was successfully logged on....
1731 May 01 02:31 SuccessA... Microsoft-Windows...        4672 Special privileges assigned to new logon....
1730 May 01 02:31 SuccessA... Microsoft-Windows...        4624 An account was successfully logged on....
1729 May 01 02:31 SuccessA... Microsoft-Windows...        4672 Special privileges assigned to new logon....
1728 May 01 02:31 SuccessA... Microsoft-Windows...        4624 An account was successfully logged on....
```

```

PS C:\Users\Administrator\Documents> Get-EventLog -LogName Security | Where-Object { $_.EventID -eq 4625 }
Index Time          EntryType   Source           InstanceID Message
---- --          -----   -----           -----   -----
1379 Apr 30 12:18 FailureA... Microsoft-Windows...        4625 An account failed to log on....
```

```

PS C:\Users\Administrator\Documents>
```

5.3 Extract operating system details using PowerShell.

This retrieves your **OS name, version, and architecture**

Get-ComputerInfo | Select-Object WindowsProductName, WindowsVersion, OsArchitecture

```

PS C:\Users\Administrator\Documents> Get-ComputerInfo | Select-Object WindowsProductName, WindowsVersion, OsArchitecture
WindowsProductName      WindowsVersion OsArchitecture
-----                  -----       -----
Windows Server 2022 Datacenter 2009      64-bit
```

```

PS C:\Users\Administrator\Documents>
```

This shows **when your system was last restarted**

(Get-CimInstance Win32_OperatingSystem).LastBootUpTime

```

PS C:\Users\Administrator\Documents> (Get-CimInstance Win32_OperatingSystem).LastBootUpTime
Tuesday, April 29, 2025 9:33:54 PM
```

```

PS C:\Users\Administrator\Documents>
```

```
#####
## Task 1: Exploring PowerShell Commands
#####

### 1 Identify available PowerShell commands.
#####

# Identify available commands.
Get-Command

# Filtering Commands
#List only cmdlets:
Get-Command - CommandType Cmdlet
#Find commands related to networking:
Get-Command -Name *Net*

#Discover commands for remote sessions:
Get-Command -Name *PSSession*

### 2. Retrieve detailed help for a specific command.
#####

### Get Help on a Specific Command
# Get-Help <Cmdlet-Name> -Detailed
Get-Help Get-NetAdapter -Detailed

# See examples, use:
Get-Help Get-NetAdapter -Examples

#Get full documentation, including parameters and descriptions:
Get-Help Get-NetAdapter -Full

# Get the absolute most detailed version:
Get-Help Get-NetAdapter -Online

## 3. Retrieve detailed help for a specific command.
#####

## To list all properties and methods of an object in , use the `Get-Member` cmdlet:

### Basic Syntax
Get-Service | Get-Member

# This lists all methods and properties of objects returned by `Get-Service`.

### Inspect a Specific Object
# To check properties and methods for a specific object, use:
$object = Get-NetAdapter
$object | Get-Member

### Filtering for Only Properties or Methods
# List only properties:
$object | Get-Member -MemberType Property

# List only methods:
$object | Get-Member -MemberType Method

#####

## Task 2: Working with Objects
#####

### 1. Display process information in table and list formats.
#####

### Table Format (Structured View)
```

```
### Use the `Format-Table` cmdlet for a neatly organized table:  
Get-Process | Format-Table -AutoSize  
  
##`-AutoSize` ensures columns adjust for readability.  
  
### List Format (Detailed View)  
## Use `Format-List` to display processes in a detailed list format:  
Get-Process | Format-List  
  
### Custom Table View  
## If you only need specific details (like process name & ID), format the table like this:  
Get-Process | Format-Table Name, Id, CPU, MemoryUsage -AutoSize  
  
### Custom List View  
## For a refined list format with selected properties:  
Get-Process | Format-List Name, Id, CPU, MemoryUsage  
  
### 2. Sort processes based on CPU usage.  
#####
# To sort processes by CPU usage in PowerShell, use the following command:  
Get-Process | Sort-Object -Property CPU -Descending  
  
`Sort-Object -Descending`** ensures processes with the highest CPU usage appear at the top.  
  
### Additional Customization  
# If you want a table view for better readability:  
Get-Process | Sort-Object -Property CPU -Descending | Format-Table -AutoSize  
  
# To show **only specific properties** (like Name, ID, and CPU usage):  
Get-Process | Sort-Object -Property CPU -Descending | Select-Object Name, Id, CPU |  
Format-Table -AutoSize  
  
## 3. Select and filter objects based on specific conditions.  
#####
### Select Specific Properties  
## If you want to display only certain attributes of an object, use:  
Get-Process | Select-Object Name, Id, CPU  
  
# This will show only the Name, ID, and CPU usage of each process.  
  
### Filter Objects Based on a Condition  
# If you need to filter objects, use `Where-Object`. For example, to list processes consuming  
**more than 50% CPU, run:  
Get-Process | Where-Object { $_.CPU -gt 50 }  
  
### Filter Objects Using Multiple Conditions  
## To filter objects using multiple conditions, for example, selecting processes with CPU > 50  
AND Memory usage > 100MB, use:  
Get-Process | Where-Object { $_.CPU -gt 50 -and $_.WS -gt 100 }  
  
### Select Top N Results  
## If you want only the top 5 highest CPU-consuming processes, use:  
Get-Process | Sort-Object -Property CPU -Descending | Select-Object -First 5  
  
## 4. Loop through system objects and extract specific information.  
#####
# To loop through system objects and extract specific information in PowerShell, use a  
`ForEach-Object` loop or a `foreach` statement.  
  
### Using `ForEach-Object` (Pipeline)  
# You can iterate over objects in a pipeline and extract specific properties. For example,
```

```
listing all **running processes** with their **name and CPU usage**:  
Get-Process | ForEach-Object { "$($_.Name) - CPU: $($_.CPU)" }  
  
### Using a `foreach` Loop  
For more control, store the objects in a variable and loop through them:  
$processes = Get-Process  
foreach ($process in $processes) {  
    Write-Host "Process: $($process.Name) - ID: $($process.Id) - CPU: $($process.CPU)"  
}  
  
### Loop Through Services and Extract Specific Details  
# To list services that are **running**, with their **status and display name**:  
Get-Service | Where-Object { $_.Status -eq "Running" } | ForEach-Object { "$($_.DisplayName) - Status: $($_.Status)" }  
  
#####  
## Task 3: Managing the File System  
#####  
  
## 1. Navigate through directories and list files  
#####  
  
### Change Directory (`cd` or `Set-Location`)  
  
# To move into a different directory:  
Set-Location C:\Users\Administrator\Documents  
  
# or simply:  
  
cd C:\Users\Administrator\Documents  
  
### 2 List Files in a Directory (`Get-ChildItem`)  
# To display files and folders in the current directory:  
Get-ChildItem  
# or specify a directory:  
Get-ChildItem C:\Users\Administrator\  
  
### 3 Show Only Files (Exclude Folders)  
# To list only files, excluding directories:  
Get-ChildItem -File  
  
### 4 Show Only Folders (Exclude Files)  
# To list only directories, excluding files:  
Get-ChildItem -Directory  
  
### 5 List Files with Specific Extensions  
# To find only *.dll files, for example:  
Get-ChildItem -Path 'C:\Program Files\Common Files\System\' -Filter *.dll  
  
### 6. Navigate Back to Previous Directory  
# To return to the previous directory:  
  
cd ..  
# or  
Set-Location ..  
  
## 2 Create and delete files and folders  
#####  
  
### Create a New File  
New-Item -Path "C:\Users\Administrator\Documents\example.txt" -ItemType File  
  
### Create a New Folder
```

```
New-Item -Path "C:\Users\Administrator\Documents\NewFolder" -ItemType Directory

### Delete a File
## To remove a file:
Remove-Item -Path "C:\Users\Administrator\Documents\example.txt"
## Add `‐Force` to bypass confirmation if needed.

### Delete a Folder
## To remove an **empty** folder:
Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder"

## To delete a folder with contents, use:
Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder" -Recurse -Force

## 3 Copy and move files between locations.
#####
#3.3.1 Prepare
#1. Create a new folder called "SourceFolder" inside C:\Users\Administrator\Documents
New-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -ItemType Directory
# This ensures the directory exists before copying or moving files.
#2. Create a New File
#Create a new text file inside SourceFolder
New-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -ItemType File
#This creates an empty file that can later be edited or copied.

#3. This creates an empty file that can later be edited or copied.
# create destination Folder
New-Item -Path "C:\Users\Administrator\Documents\DestinationFolder" -ItemType Directory

#4. This creates an empty folder that can later be copied or moved
New-Item -Path "C:\Users\Administrator\Documents\OldFolder"

##
## Copy
##
#3.3.2 Copy a File
#Copy example.txt from SourceFolder to DestinationFolder
Copy-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -Destination
"C:\Users\Administrator\Documents\DestinationFolder\example.txt"
#The file remains in SourceFolder while being duplicated in DestinationFolder.

#3.3.3 Copy an Entire Folder
#Copy SourceFolder and all its contents to BackupFolder
Copy-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -Destination
"C:\Users\Administrator\Documents\BackupFolder" -Recurse
#The `‐Recurse` flag ensures that all files and subfolders are copied.

##
## Move
##
#3.3.4 Move a File
#Move example.txt from SourceFolder to DestinationFolder
Move-Item -Path "C:\Users\Administrator\Documents\SourceFolder\example.txt" -Destination
"C:\Users\Administrator\Documents\DestinationFolder\example.txt" -Force
# Unlike copying, this removes the file from SourceFolder and places it in DestinationFolder.

#3.3.5 Move an Entire Folder
#Move OldFolder and all contents into NewFolder
Move-Item -Path "C:\Users\Administrator\Documents\OldFolder" -Destination
"C:\Users\Administrator\Documents\NewFolder"
```

```
#The folder and its files will be transferred, leaving OldFolder empty or deleted.
```

#3.3.6 Restore

#1. Delete a File

```
#Delete example.txt from DestinationFolder
```

```
Remove-Item -Path "C:\Users\Administrator\Documents\DestinationFolder\example.txt"
```

```
# Use `‐Force` if you want to delete without confirmation.
```

#2. Delete a Folder (With Contents)

```
# Delete BackupFolder and all its files
```

```
Remove-Item -Path "C:\Users\Administrator\Documents\DestinationFolder\example.txt"
```

```
Remove-Item -Path "C:\Users\Administrator\Documents\BackupFolder" -Recurse -Force
```

```
Remove-Item -Path "C:\Users\Administrator\Documents\NewFolder" -Recurse -Force
```

```
Remove-Item -Path "C:\Users\Administrator\Documents\DestinationFolder" -Recurse -Force
```

```
Remove-Item -Path "C:\Users\Administrator\Documents\SourceFolder" -Recurse -Force
```

```
#This permanently removes used files and folders
```

```
## 4 Check available disk space.
```

```
#####
# Get available disk space for the C: drive
```

```
Get-PSDrive C
```

```
# Retrieve detailed volume information, including free space
```

```
Get-Volume | Select-Object DriveLetter, SizeRemaining, Size
```

```
#####
##Task 4: Managing System Services and Processes
```

```
#####
#1. List all running services on the system.
```

```
Get-Service | Where-Object { $_.Status -eq "Running" }
```

```
# 2. Start and stop specific services.
```

```
# Look for service spooler
```

```
Get-Service -Name "Spooler"
```

```
# Stop a specific service (Spooler)
```

```
Stop-Service -Name "Spooler"
```

```
# Start a specific service (Spooler)
```

```
Start-Service -Name "Spooler"
```

```
# Stop a specific service (Spooler)
```

```
Stop-Service -Name "Spooler"
```

```
# Restore and Retrieve information about the Spooler service
```

```
Start-Service -Name "Spooler"
```

```
Get-Service -Name "Spooler"
```

```
#3. Retrieve information about active processes.
```

```
Get-Service -Name "Spooler"
```

```
Get-Process | Select-Object Name, Id, CPU, WorkingSet
```

```
#4. Terminate a process.
```

```
Start-Process -FilePath "notepad.exe"
```

```
Get-Process | Where-Object { $_.Name -eq "notepad" }
```

```
Stop-Process -Name "notepad" -Force
```

```
#####
##Task 5: Monitoring Event Logs and System Information
```

```
#####
# 1 View the latest system event logs.
```

```
Get-EventLog -LogName System -Newest 10
```

```
Get-EventLog -LogName System | Select-Object -First 20 | Format-List *
```

```
#2 Retrieve security event logs.
```

```
Get-EventLog -LogName Security -Newest 10
```

```
Get-EventLog -LogName Security | Where-Object { $_.EventID -eq 4625 }
## Event ID 4625 corresponds to failed login attempts

## 3 Extract operating system details using PowerShell.

# This retrieves your OS name, version, and architecture

Get-ComputerInfo | Select-Object WindowsProductName, WindowsVersion, OsArchitecture

# This shows when your system was last restarted
(Get-CimInstance Win32_OperatingSystem).LastBootUpTime
```