Windows PowerShell CMD __ ocTony

Powershell alias –shortkey : get-alias | Out-File -FilePath "c:\temp\alias.txt"

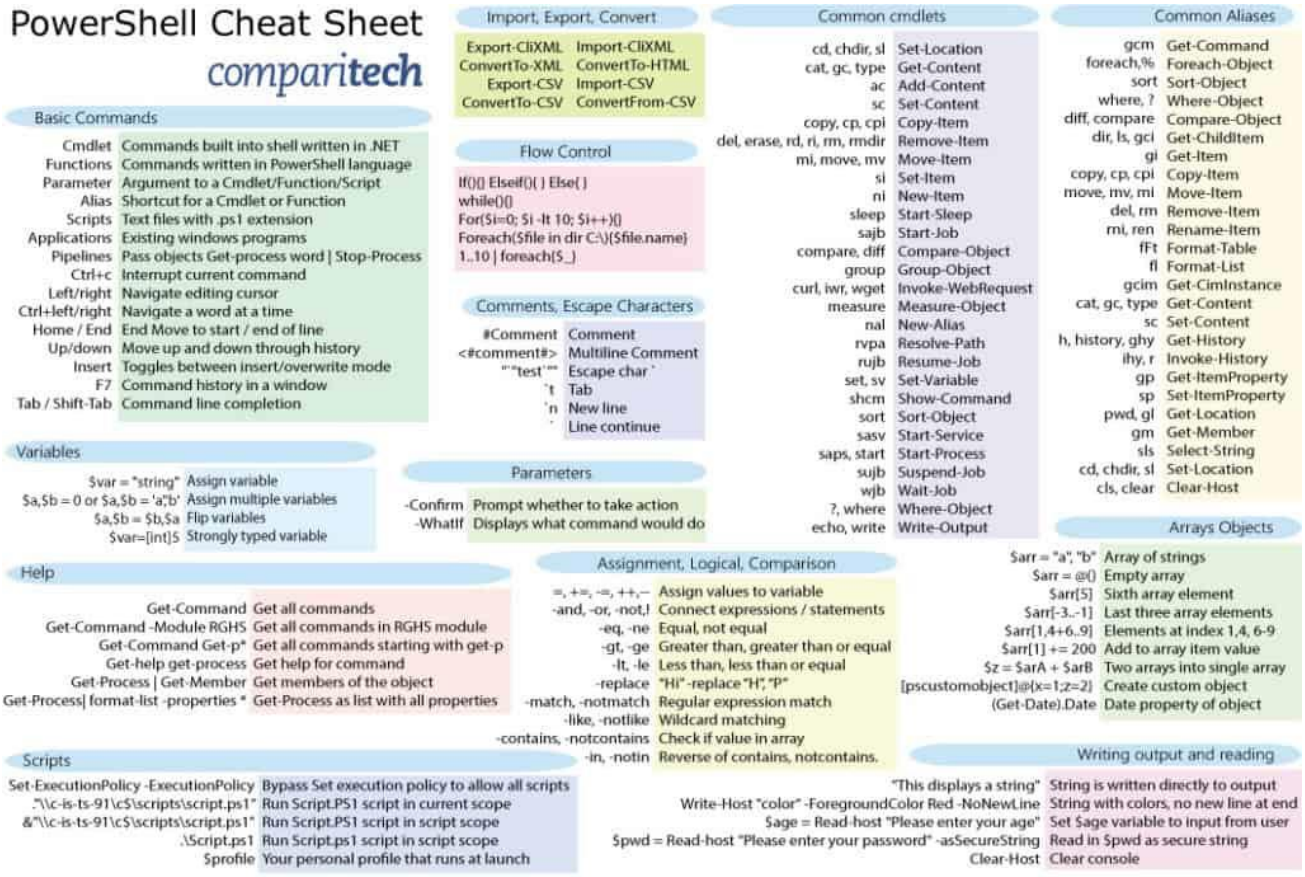
PowerShell is an interactive **Command-Line Interface** (**CLI**) and automation engine designed by Microsoft to help design system configurations and automate administrative tasks.

--write output filename:

Get-Process | Where-Object { $_.WorkingSet -gt 100MB } | Sort-Object CPU -Descending | Out-File -FilePath "c:\temp\processs.txt"

Get-Process -ComputerName 192.168.29.117 | Where-Object { $_.WorkingSet -gt 100MB } | Sort-Object CPU -Descending

PowerShell is a cross-platform task automation solution made up of a command-line shell, a scripting language, and a configuration management framework. PowerShell runs on Windows, Linux, and macOS. – Microsoft Learn



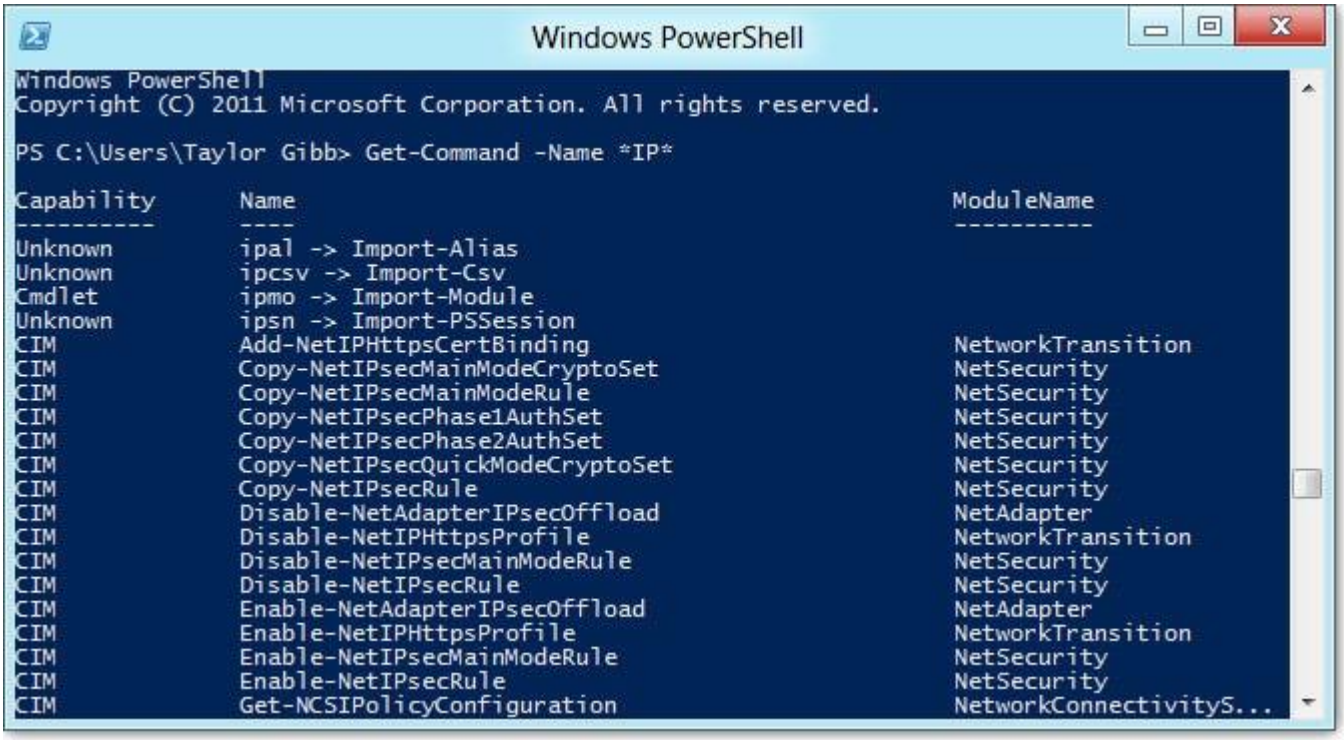
The **latest version is 7.3** (7.4 is available in preview mode). If you are a Windows 10

What are the different ways I can run PowerShell as an Administrator?

1. Press WIN + R, type in **powershell**, press Ctrl+Shift+Enter. Click **OK** to run as Administrator.
2. Type powershell into the Taskbar search field. Select **Run as Administrator** from the list of options in the right panel of the results list.
3. Open the Command Prompt, type **powershell,** and press Enter. Type **start-process PowerShell -verb runas** and press Enter.
4. *Also in the Command Prompt, type **runas /netonly /user:RemoteDomain\Administrator powershell** (substitute a URL or hostname for RemoteDomain)

How to Run cmdlets

In a nutshell, a cmdlet is a single-function command. You input cmdlets into the command line just as you would with a traditional command or utility. Cmdlets are the main way to interact with the CLI.



In PowerShell, most cmdlets are written in C# and comprised of instructions designed to perform a function that returns a .NET object.

Over 200 cmdlets can be used in PowerShell. Windows PowerShell command prompt isn't case-sensitive, so these commands can be typed in either upper or lower case. The main cmdlets are listed below:

- **Get-Location** – Get the current directory
- **Set-Location** – Get the current directory
- **Move-item** – Move a file to a new location
- **Copy-item** – Copy a file to a new location
- **Rename** – item Rename an existing file
- **New-item** – Create a new file

For a full list of commands available to you, use the `Get-Command` cmdlet. In the command line you would enter the following:

```
PS C:\> Get-Command
```

It is important to note that Microsoft restricts users from using custom PowerShell cmdlets in its default settings. In order to use PowerShell cmdlets, you need to change the **ExecutionPolicy** from **Restricted** to **RemoteSigned**. **Remote Signed** will allow you to run your own scripts but will stop unsigned scripts from other users.

To change your Execution policy, type in the following PowerShell command:

```
PS C:\>   Set-ExecutionPolicy
```

To change to **RemoteSigned**, type the following command:

```
PS C:\> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

To run a script, enter its folder and filename into the PowerShell window:

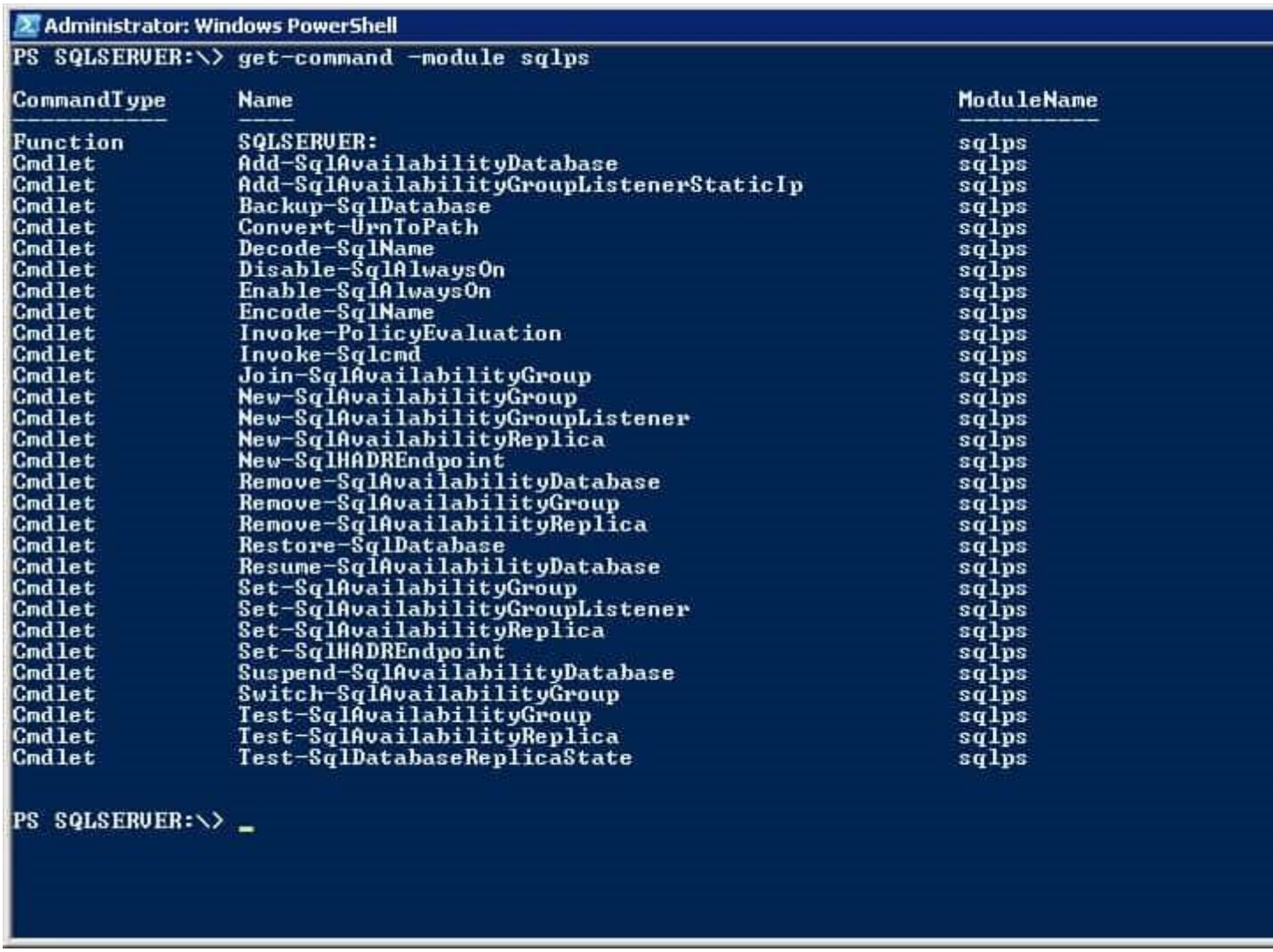
```
PS c:\powershell\mynewscript.ps1
```

Here are some commands that are common to PowerShell and Windows:

- `cd`: Change Directory. This command is used to change the current working directory. In PowerShell, `Set-Location` can be used as well.
- `cls`: Clear Screen. This command clears the screen of the console. In PowerShell, `Clear-Host` or its alias `cls` can be used.
- `dir`: Directory. This command lists the files and subdirectories in the directory. In PowerShell, `Get-ChildItem` can be used as well.
- `echo`: This command prints text to the console. In PowerShell, `Write-Output` can be used as well.

- `copy`: This command copies files. In PowerShell, `Copy-Item` can be used as well.
- `del`: Delete. This command deletes one or more files. In PowerShell, `Remove-Item` can be used as well.
- `move`: This command moves files from one location to another. In PowerShell, `Move-Item` can be used as well.
- `type`: This command displays the contents of a text file. In PowerShell, `Get-Content` can be used as well.
- `find`: This command searches for a text string in a file. In PowerShell, `Select-String` can be used as well.
- `exit`: This command closes the command prompt or terminal window. It works the same in both Command Prompt and PowerShell.

Backing Up an SQL Database with PowerShell



Many people use PowerShell to back up SQL databases. The command-line interface can conduct full database backups, file backups, and transaction log backups. There are many ways to backup a database in PowerShell, but one of the simplest is to use the `Backup-SqlDatabase` command. For example:

```
PS C:\> Backup-SqlDatabase -ServerINstance “Computer\Instance” -Database “Databasecentral”
```

This will create a database backup of a database with the name ‘Databasecentral’ (or the name of your chosen database’.

To back up a transaction log, you would input:

```
PS C:\> Backup-SqlDatabase -ServerInstance "Computer\Instance"  -Database
"Databasecentral" -BackupAction Log
```

This will create a transaction log of the selected database.

The Essential PowerShell Commands

Using aliases will only get you so far on PowerShell, so it's important to commit to learning everything you can about PowerShell's native commands. We touched on some of these above, but we're going to break down the main ones in much more detail below.

Get-Help

This command should be at the very top of any new user's list when it comes to PowerShell. The Get-Help command can be used to literally get help with any other PowerShell command. For example, if you know the name of a command, but you don't know what it does or how to use it, the Get-Help command provides the full command syntax.

For example, **if you wanted to see how Get-Process works, you would type:**

```
PS C:\> Get-Help -Name Get-Process
PS C:\> Set-ExecutionPolicy
```

As touched on earlier in this guide, Microsoft has a restricted execution policy that prevents scripting on PowerShell unless you change it. When setting the execution policy, you have four options to choose from:

- **Restricted** – The default execution policy that stops scripts from running.
- **All Signed** – Will run scripts if they are signed by a trusted publisher
- **Remote Signed** – Allows scripts to run which have been created locally
- **Unrestricted** – A policy with no restrictions on running scripts

```
PS C:\> Get-ExecutionPolicy
```

If you're using PowerShell, you may not always work on a server that you're familiar with. Running the command **Get-Execution Policy** will allow you to see which policy is active on the server before running a new script. If you then see the server in question operating under a restricted policy, you can then implement the **Set-ExecutionPolicy** command to change it.

Get-Service

One of the most important commands is `Get-Service`, which provides the user with a list of all services installed on the system, both running and stopped. This cmdlet can be directed by using specific service names or objects.

For example, if you were to type `PS C:\> Get-Service`, you would be shown a list of all services on your computer, their statuses, and display names.

To use this command to retrieve specific services, type: `PS C:\ Get-Service "WMI*"` to retrieve all services that begin with WMI.

If you wanted to restrict output to active services on your computer, input the following command:

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

ConvertTo-HTML

When using PowerShell, you might want to generate a report about the information you've seen. One of the best ways to do this is by using the **ConvertTo-HTML** command. This cmdlet allows you to build reports with tables and color, which can help to visualize complex data. Simply choose an object and add it to the command.

For example, you could type:

```
Get-PSDrive | ConvertTo-Html
```

This returns a mass of information, so it's a good idea to limit it to a file with the Out-File command. A better alternative command is:

```
Get-PSD Drive | ConvertTo-Html | Out-File -FilePath PSDrives.html
```

This will then generate an HTML file in table form. For example:

Last Updated	Created on	Folder Name	Owner	Size
1/11/2013 7:50:05 PM	9/12/2012 7:36:35 AM	C:\utils	BUILTIN\Administrators	3.31 M
12/31/2012 8:26:54 PM	12/31/2012 8:26:54 PM	C:\utils\empty	BUILTIN\Administrators	0.00 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\Move-PicturesToDropbox	BUILTIN\Administrators	0.47 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp	BUILTIN\Administrators	0.89 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp\docProps	BUILTIN\Administrators	0.00 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp\word	BUILTIN\Administrators	0.89 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp_rels	BUILTIN\Administrators	0.00 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp\word\media	BUILTIN\Administrators	0.84 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp\word\theme	BUILTIN\Administrators	0.01 M
1/11/2013 7:50:05 PM	1/11/2013 7:50:05 PM	C:\utils\temp\word_rels	BUILTIN\Administrators	0.00 M

You can then add your own colors and borders to refine its presentation.

Export-CSV (and Get-Service)

No less important for increasing visibility is the Export-CSV command. It allows you to export PowerShell data into a CSV file. Essentially, this command creates a CSV file compiling all of the objects you've selected in PowerShell. Every object has its own line or row within the CSV file. This command is primarily used to create spreadsheets and share data with external programs.

To use this command, you would type:

```
PS C:\> Get-Service | Export-CSV c:\service.csv
```

It's important to remember not to format objects before running the Export-CSV command. This is because formatting objects results in only the formatted properties being placed into the CSV file rather than the original objects themselves. In the event that you want to send specific properties of an object to a CSV file, you would use the **Select-Object** cmdlet.

To use the **Select-Object** cmdlet, type:

```
PS C:\> Get-Service | Select-Object Name, Status | Export-CSV
c:\Service.csv
```

Get-Process

If you want to view all processes currently running on your system, the **Get-Process** command is very important. To get a list of all active processes on your computer, type:

```
PS C:\ Get-Process
```

Notice that if you don't specify any parameters, you'll get a breakdown of every active process on your computer. To pick a specific process, narrow the results down by process name or process ID and combine that with the **Format-List** cmdlet, which displays all available properties. For example:

```
PS C:\ Get-Process windowrd, explorer | Format-List *
```

This provides you with comprehensive oversight of all active processes.

Get-EventLog

If you ever want to access your computer's event logs (or logs on remote computers) while using PowerShell, then you're going to need the **Get-EventLog** command. This cmdlet only works on classic event logs, so you'll need the **Get-WinEvent** command for logs later than Windows Vista.

To **run the event log command**, type:

```
PS C:\> Get-EventLog -List
```

This will show all event logs on your computer.

One of the most common reasons users look at event logs is to see errors. If you want to **see error events in your log**, simply type:

```
PS C:\> Get-EventLog -LogName System -EntryType Error
```

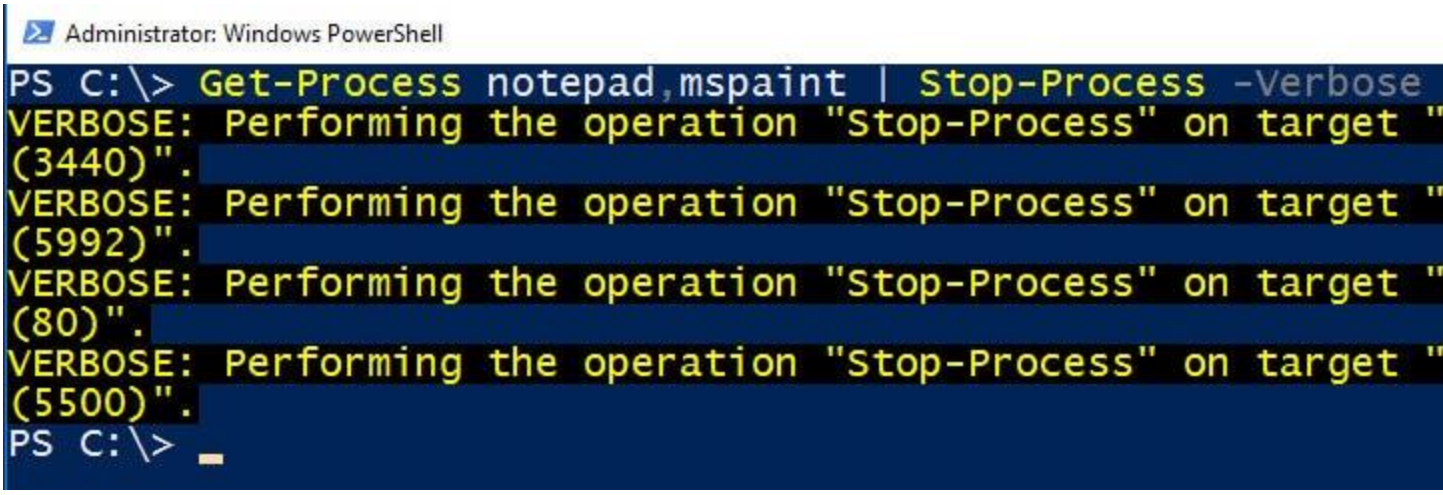
If you want to **get event logs from multiple computers**, specify which devices you want to view (listed below as “Server1” and “Server2”). For example:

```
PS C:\> Get-EventLog - LogName “Windows PowerShell” -ComputerName “local computer”, “Server1”, “Server2”.
```

Parameters you can use to search event logs include:	
After	User specifies a date and time and the cmdlet will locate events that occurred after
AsBaseObject	Provides a System.Diagnostics.EventLogEntry for each event
AsString	Returns the output as strings
Before	User specifies a date and time and the cmdlet will locate events that occurred before
ComputerName	Used to refer to a remote computer

Parameters you can use to search event logs include:	
EntryType	Specifies the entry type of events (Error, Failure Audit, Success Audit, Information, Warning
Index	Specifies index values the cmdlet finds events from
List	Provides a list of event logs
UserName	Specifies usernames associated with a given event

Stop-Process



When using PowerShell, it's not uncommon to experience a process freezing up. Whenever this happens, you can use **Get-Process** to retrieve the name of the process experiencing difficulties and then stop it with the **Stop-Process** command.

Generally, you terminate a process by its name. For example:

```
PS C:\> Stop-Process -Name "notepad"
```

In this example, the user has terminated Notepad by using the `Stop-Process` command.

PowerShell Commands List

Here are 25 basic PowerShell commands:

Command name	Alias	Description
Set-Location	cd, chdir, sl	Sets the current working location to a specified location.
Get-Content	cat, gc, type	Gets the content of the item at the specified location.
Add-Content	ac	Adds content to the specified items, such as adding words to a file.
Set-Content	sc	Writes or replaces the content in an item with new content.
Copy-Item	copy, cp, cpi	Copies an item from one location to another.
Remove-Item	del, erase, rd, ri, rm, rmdir	Deletes the specified items.

Command name	Alias	Description
Move-Item	mi, move, mv	Moves an item from one location to another.
Set-Item	si	Changes the value of an item to the value specified in the command.
New-Item	ni	Creates a new item.
Start-Job	sajb	Starts a Windows PowerShell background job.
Compare-Object	compare, dif	Compares two sets of objects.
Group-Object	group	Groups objects that contain the same value for specified properties.
Invoke-WebRequest	curl, iwr, wget	Gets content from a web page on the Internet.
Measure-Object	measure	Calculates the numeric properties of objects, and the characters, words, objects, such as files …

Command name	Alias	Description
Resolve-Path	rvpa	Resolves the wildcard characters in a path, and displays the path conten
Resume-Job	rujb	Restarts a suspended job
Set-Variable	set, sv	Sets the value of a variable. Creates the variable if one with the requeste exist.
Show-Command	shcm	Creates Windows PowerShell commands in a graphical command wind
Sort-Object	sort	Sorts objects by property values.
Start-Service	sasv	Starts one or more stopped services.
Start-Process	saps, start	Starts one or more processes on the local computer.
Suspend-Job	sujb	Temporarily stops workflow jobs.

Command name	Alias	Description
Wait-Job	wjb	Suppresses the command prompt until one or all of the Windows Power jobs running in the session are …
Where-Object	?, where	Selects objects from a collection based on their property values.
Write-Output	echo, write	Sends the specified objects to the next command in the pipeline. If the c last command in the pipeline,…

PowerShell: A powerful command-line interface

Although making the transition to PowerShell can seem quite complex, it's command-line interface operates much the same as any other. It may have its own unique cmdlets, but a wealth of online resources can help you with any administrative task you can think of. To get the most out of PowerShell, you simply need to get used to the multitude of commands available to you.

As a new user, it is easy to become daunted by PowerShell's 200-plus cmdlets. Make sure you start out with the command line interface before graduating to the full-blown GUI. Regardless of whether you're new to PowerShell or command-line interfaces, more than enough information is available online to help you make the most of this powerful tool.

PowerShell Community Resources

- PowerShell Slack
- PowerShell Discord Server

PowerShell Commands FAQs

How do I navigate in Windows PowerShell?

The most important navigation actions you need to know for PowerShell is how to get into it and how to get out again. The easiest way to access the PowerShell environment is to type **PowerShell** in the search field of your taskbar. PowerShell runs in its own window, so you can close it down just by clicking on the **X** in the top right corner of the window's frame. The proper way to close the window is to type **exit** and the command prompt. The standard navigation commands of the Command Prompt work in PowerShell so use **cd** to change directory. Enter a drive letter followed by a colon (eg. **D:**) to switch to another drive.

Is Windows PowerShell the same as Command Prompt?

PowerShell is an advancement on Command Prompt because its shell scripting capabilities include better programming constructs than those available for batch jobs in Command Prompt. All of the Command Prompt commands are available in PowerShell but then PowerShell has extra commands and utilities, called cmdlets. Think of PowerShell as Command Prompt +.

How do I learn bash scripting?

Bash scripting is a Unix shell script. As Linux is an adaptation of Unix, a shell script written for Linux is often called a Bash script. There are a lot of online tutorials on how to create a Bash script. In order to avoid confusion, try not to refer to a PowerShell script as a Bash script.

How can I make Command Prompt default instead of PowerShell?

When you press WIN + X, you now get a PowerShell window instead of the old Command Prompt. To stick with Command Prompt, go to the Start menu and click on **Settings**. In the Settings menu, select **Personalization**. Select **Taskbar** in the left-hand menu of the Personalization Settings Window. In the main panel of that window, look for **Replace Command Prompt with Windows PowerShell in the**

menu when I right-click the Start button or press Windows key+X. Set that to **Off**.

What are the different ways I can run PowerShell as an Administrator?

1. Press WIN + R, type in **powershell**, press Ctrl+Shift+Enter. Click **OK** to run as Administrator.
2. Type powershell into the Taskbar search field. Select **Run as Administrator** from the list of options in the right panel of the results list.
3. Open the Command Prompt, type **powershell,** and press Enter. Type **start-process PowerShell -verb runas** and press Enter.

How do I run PowerShell commands?

You can run PowerShell commands from a Command Prompt window by using the format: **powershell -command ” <PowerShellCode> “** but put your PowerShell command inside the quotes instead of **<PowerShellCode>**. If your PowerShell command requires a value in quotes, use single quotes in there instead of double-quotes. The surrounding quotes in the execution example here should remain as double-quotes.

```
# Define the username of the account you want to disable

$username = "Username"

------------------------------------------------------

#Create a local account

New-LocalUser -Name "Adminpro8" -Password (ConvertTo-SecureString "Naxxtiox11.." -AsPlainText -Force) -PasswordNeverExpires



# Calculate the date six months from now

$expirationDate = (Get-Date).AddMonths(6)



# Disable the account by setting the expiration date
```

Set-LocalUser -Name "Adminpro2" -AccountExpires $expirationDate

---

Remove the user from the administrators group

Remove-LocalGroupMember -Group "Administrators" -Member "Adminpro2"

----Create local Adminpro account & belong AdminGroup - After RDP

#Create a local account

New-LocalUser -Name "Adminpro2" -Password (ConvertTo-SecureString "Naxxtiox11.." -AsPlainText -Force) -PasswordNeverExpires

ADD the user from the administrators group

add-LocalGroupMember -Group "Administrators" -Member "Adminpro2"

#Create a local account

New-LocalUser -Name "Adminpro2" -Password (ConvertTo-SecureString "Naxxtiox11.." -AsPlainText -Force) -PasswordNeverExpires

ADD the user from the administrators group

add-LocalGroupMember -Group "Administrators" -Member "Adminpro2"

#Enable ping command

New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4 -IcmpType 8 -Enabled True -Action Allow

#list port

Get-NetTCPConnection -ComputerName "RemoteServerName" -State Established | Select-Object LocalAddress, LocalPort, RemoteAddress, RemotePort

-------check PS version

$remoteServer = "<RemoteServerName>"

```
try {
   # Run a command on the remote server to get its PowerShell version
   $psVersion = Invoke-Command -ComputerName $remoteServer -ScriptBlock {
      $PSVersionTable.PSVersion
   }
   Write-Host "PowerShell version on $remoteServer: $($psVersion.Major).$($psVersion.Minor)"
} catch {
   Write-Host "Failed to retrieve PowerShell version on $remoteServer. Error: $_"
}
```

---------- add account in remote servers

Connect to the remote server using PowerShell Remoting

Enter-PSSession -ComputerName 4.242.219.22 -Credential (Get-Credential)

Enter your credentials when prompted

Create a new local user account

New-LocalUser -Name adminpro3 -Password (ConvertTo-SecureString -AsPlainText "Naxxtiox11.." -Force)

Optionally, add the user to any necessary groups:

Add-LocalGroupMember -Group Administrator -Member adminpro3

Exit the remote session

Exit-PSSession

COMMAND Scan open ports:

$server = "RemoteServerName" # Replace "RemoteServerName" with the actual server name or IP address

Define a range of ports to test

```
$startPort = 1
$endPort = 65535

# Loop through the range of ports and test each one
for ($port = $startPort; $port -le $endPort; $port++) {
   $result = Test-NetConnection -ComputerName $server -Port $port -InformationLevel Quiet

   # Check if the port is open
   if ($result.TcpTestSucceeded) {
      Write-Output "Port $port is open on $server."
   }
}
------------------------
$server = "192.168.216.133"  # Replace "RemoteServerName" with the actual server name or IP address

# Define a range of ports to test
$startPort = 1
$endPort = 65535

# Loop through the range of ports and test each one
for ($port = $startPort; $port -le $endPort; $port++) {
   $result = Test-NetConnection -ComputerName $server -Port $port -InformationLevel Quiet

   # Check if the port is open
   if ($result.TcpTestSucceeded) {
      Write-Output "Port $port is open on $server."
   }
}
------------------------
```

DISABLE ACCOUNT in 6mos

```
# Calculate the date six months from now
$expirationDate = (Get-Date).AddMonths(6)

# Disable the account by setting the expiration date
Set-LocalUser -Name "Adminpro2" -AccountExpires $expirationDate
```

```
net user NewAdmin NewPassword /add
net localgroup Administrators NewAdmin /add

# Import the Active Directory module
Import-Module ActiveDirectory

# Retrieve a specific Active Directory object by specifying its distinguished name
Get-ADObject -Identity "CN=John Doe,OU=Users,DC=example,DC=com"

# Retrieve all objects of a specific object class (e.g., user objects)
Get-ADObject -Filter {ObjectClass -eq "user"}

# Retrieve all objects within a specific Organizational Unit (OU)
Get-ADObject -SearchBase "OU=Users,DC=example,DC=com"

# Retrieve all objects with a specific attribute value
Get-ADObject -Filter {EmailAddress -eq "john.doe@example.com"}
```