Pràctica RA3.2 – Guia de cmdlets

Aquest document és una **introducció pràctica** als cmdlets que necessitareu per resoldre les 4 activitats de la pràctica: Es una guia de pistes amb exemples que podeu copiar, modificar i adaptar.

Continguts

>_ 1. Què és un cmdlet?

Un **cmdlet** (es pronuncia *"command-let"*) és el bloc bàsic de PowerShell. És una mini-funció pensada per fer *una sola cosa*. Tots segueixen el mateix patró de nom:

```
Verb-Nom
```

El verb indica **què fa** (Get, Set, New, Remove, Start, Stop...) i el nom indica **sobre què**. Així, només llegint elnom ja saps què fa:

Cmdlet	Què fa
`Get-Service`	Obté informació dels serveis
`Start-Service`	Inicia un servei
`Stop-Service`	Atura un servei
`Get-Date`	Obté la data actual
`New-Item`	Crea un fitxer o carpeta
`Remove-Item`	Esborra un fitxer o carpeta
`Test-Path`	Comprova si una ruta existeix

Els paràmetres dels cmdlets

Els cmdlets reben paràmetres amb el format `-Nom valor`. Per exemple:

```
PS> New-Item -ItemType "Directory" -Path "C:\Empresa\Logs"
```

L'ordre dels paràmetres **no importa** perquè cadascun va precedit del seu nom. Això fa que el codi es llegeixi gairebé com una frase en anglès.

> **Truc útil:** si un cmdlet no t'és familiar, escriu `Get-Help Nom-Cmdlet` a la consola — és l'equivalent al `man` de Linux. Per veure exemples concrets: `Get-Help Nom-Cmdlet -Examples`.

>_ 2. Ordres clàssiques (estil .bat)

PowerShell és **compatible amb les ordres de tota la vida** del CMD. Pots utilitzar-les igual que faries a un fitxer `.bat`. Són ideals per a tasques simples on no cal la potència dels cmdlets.

> **`whoami`**
>
> *Mostra l'usuari connectat al sistema.*
>
> ```
> PS> whoami
> empresa\jgarcia
> ```

`hostname`

Mostra el nom de l'equip.

```
PS> hostname
P-CICLES-31
```

`date`

Mostra la data i hora actual.

```
PS> date
dilluns, 04 de maig de 2026 09:00:03
```

`echo`

Mostra un text per pantalla.

```
PS> echo "Hola món"
Hola món
```

Important: totes aquestes ordres tornen el resultat al "pipeline" de PowerShell. Això vol dir que pots **redirigir-les a un fitxer** o combinar-les amb altres ordres exactament com faries a Bash.

>_ 3. Redirecció i operadors

Els operadors de redirecció funcionen igual que a Bash: agafen la sortida d'una ordre i la dirigeixen cap a un fitxer en lloc de mostrar-la per pantalla.

Operador	Què fa	Si el fitxer ja existeix...
`>`	Redirigeix la sortida a un fitxer	**El sobreescriu**
`>>`	Redirigeix afegint al final	**Manté el contingut**

Exemples

```
# Sobreescriu el fitxer cada vegada (per a inventaris diaris, per exemple)
PS> whoami > "C:\Empresa\Logs\usuari.txt"

# Afegeix al final (per a logs que s'acumulen)
PS> whoami >> "C:\Empresa\Logs\historial.log"
```

Combinant ordres en un script

```
# registre.ps1 — afegeix data, equip i usuari al fitxer
echo "--- $(date) ---" >> C:\Empresa\Logs\registre.log
hostname                >> C:\Empresa\Logs\registre.log
whoami                  >> C:\Empresa\Logs\registre.log
echo ""                 >> C:\Empresa\Logs\registre.log
```

> **Observació:** el patró `$(ordre)` executa l'ordre i n'incrusta el resultat dins d'una cadena. És equivalent a `$(...)` de Bash.

>_ 4. Treball amb fitxers i carpetes

Test-Path

Comprova si una ruta (fitxer o carpeta) existeix. Retorna `True` o `False`.

```
PS> Test-Path "C:\Empresa\Logs"
False

# Ús típic dins d'un if:
if (-not (Test-Path "C:\Empresa\Logs")) {
    # Crear la carpeta perquè no existeix
}
```

New-Item

Crea un nou fitxer o carpeta.

```
# Crear una carpeta
PS> New-Item -ItemType "Directory" -Path "C:\Empresa\Logs" -Force

# Crear un fitxer buit
PS> New-Item -ItemType "File" -Path "C:\Empresa\Logs\buit.txt"
```

`-Force` evita errors si la carpeta ja existeix. `| Out-Null` al final amaga la sortida.

Out-File

Versió "cmdlet" de la redirecció `>` i `>>`, amb més opcions.

```
# Sobreescriure
PS> whoami | Out-File -FilePath "C:\Empresa\Logs\usuari.txt"

# Afegir al final (com a >>)
PS> whoami | Out-File -FilePath "C:\Empresa\Logs\hist.log" -Append

# Especificar la codificació (recomanat per a accents catalans)
PS> whoami | Out-File -FilePath "file.txt" -Encoding UTF8
```

Remove-Item

Esborra un fitxer o carpeta. Compte amb el `-Recurse`!

```
# Esborrar un fitxer
PS> Remove-Item "C:\tmp\fitxer.log"

# Esborrar una carpeta i tot el seu contingut
PS> Remove-Item "C:\tmp\vell" -Recurse -Force
```

>_ 5. Comprimir i arxivar

Compress-Archive

Crea un fitxer ZIP a partir d'un o més fitxers/carpetes.

```
# Comprimir una carpeta sencera
PS> Compress-Archive -Path "C:\Empresa\Documents\*" -
DestinationPath "D:\Backups\docs.zip"

# Sobreescriure si ja existeix el ZIP
PS> Compress-Archive -Path "C:\dades\*" -DestinationPath "D:
\backup.zip" -Force
```

Cas pràctic — backup amb data al nom: per a un nom de fitxer únic per a cada backup, combina `Get-Date` dins de la cadena del nom:

```
$data = Get-Date -Format "yyyyMMdd"
Compress-Archive -Path "C:\Empresa\Documents\*" -DestinationPath "D:
\Backups\backup_$data.zip"
```

>_ 6. Treball amb dates

Get-Date

Obté la data i hora actual. Amb `-Format` en pots controlar el format de sortida.

```
PS> Get-Date
dilluns, 04 de maig de 2026 09:00:03

PS> Get-Date -Format "yyyy-MM-dd"
2026-05-04

PS> Get-Date -Format "yyyy-MM-dd HH:mm"
2026-05-04 09:00

PS> Get-Date -Format "yyyyMMdd_HHmm"
20260504_0900
```

Formats útils per a noms de fitxer

Format	Resultat	Útil per a...
`yyyyMMdd`	`20260504`	Backups diaris
`yyyy-MM-dd`	`2026-05-04`	Logs llegibles
`yyyy-MM-dd_HHmm`	`2026-05-04_0900`	Logs amb hora
`yyyyMM`	`202605`	Backups mensuals

Calcular dates passades: per obtenir la data de fa N dies, fes servir `(Get-Date).AddDays(-N)`. Per exemple, fa 30 dies: `(Get-Date).AddDays(-30)`.

>_ 7. Llistar i filtrar fitxers

Get-ChildItem (àlies: ls, dir, gci)

Llista el contingut d'una carpeta. És l'equivalent a `ls` de Linux o `dir` de CMD.

```
# Llistar contingut
PS> Get-ChildItem "C:\Windows\Temp"

# Llistar recursivament (inclou subcarpetes)
PS> Get-ChildItem "C:\Windows\Temp" -Recurse

# Filtrar només fitxers (no carpetes)
PS> Get-ChildItem "C:\Windows\Temp" -File
```

Where-Object (àlies: where, ?)

Filtra els elements segons una condició. És el cmdlet equivalent a un `if` dins d'un pipeline.

```
# Filtrar fitxers més antics de 30 dies
PS> Get-ChildItem "C:\Windows\Temp" |
      Where-Object { $_.LastWriteTime -lt (Get-Date).AddDays(-30) }
```

`$_` representa l'element actual del pipeline. `LastWriteTime` és la data de l'última modificació del fitxer. `-lt` significa "less than".

Operadors de comparació

Operador	Significat
`-eq`	igual a (equal)
`-ne`	diferent de (not equal)
`-lt`	menor que (less than)
`-gt`	major que (greater than)
`-le`	menor o igual (less or equal)
`-ge`	major o igual (greater or equal)

> **Per què no < i >?** Perquè `<` i `>` ja són operadors de redirecció a PowerShell. Per evitar confusions, els operadors de comparació es prefixen amb un guió i les inicials angleses.

Cas pràctic: esborrar fitxers antics

```
# Esborra tot el que té més de 15 dies a C:\Windows\Temp
Get-ChildItem "C:\Windows\Temp" -File -Recurse |
    Where-Object { $_.LastWriteTime -lt (Get-Date).AddDays(-15) } |
    Remove-Item -Force
```

El "pipeline" (la barra `|`) connecta cmdlets: la sortida d'un és l'entrada del següent. Llegit de dalt a baix: *"llista els fitxers, filtra els antics, i esborra'ls"*. Tres línies, sense escriure cap bucle.

>_ 8. Gestió de serveis

Get-Service

Obté informació sobre un o més serveis del sistema.

```
# Veure l'estat d'un servei concret
PS> Get-Service -Name "Spooler"

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

# Veure només l'estat (text)
PS> (Get-Service -Name "Spooler").Status
Running
```

Start-Service

Inicia un servei aturat. `Stop-Service` *fa l'invers.*

```
PS> Start-Service -Name "Spooler"

# Comprovar l'estat i arrencar-lo si està aturat
if ((Get-Service "Spooler").Status -ne "Running") {
    Start-Service "Spooler"
}
```

Estats possibles d'un servei

Estat	Significat
`Running`	El servei està en execució
`Stopped`	El servei està aturat
`Paused`	El servei està pausat
`StartPending`	S'està iniciant
`StopPending`	S'està aturant

>_ Consells finals

1. Prova primer a la consola. Abans de posar les ordres al script, prova-les una per una a la consola de PowerShell ISE o de VS Code. Així veus què retorna cada cmdlet i pots ajustar paràmetres.

2. Get-Help és el teu amic. `Get-Help Compress-Archive -Examples` et mostrarà exemples reals d'ús del cmdlet. `Get-Help Cmdlet -Full` per a la documentació completa.

3. Comenta el codi. Cada script ha de començar amb un bloc de comentaris que indiqui què fa, qui l'ha escrit i quan. Així d'aquí 6 mesos sabreu què fa cada línia.

Política d'execució: abans de poder executar scripts `.ps1` per primera vegada, segurament caldrà canviar la política d'execució amb:

```
PS> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope
CurrentUser
```