TP4 : PowerShell Scripts

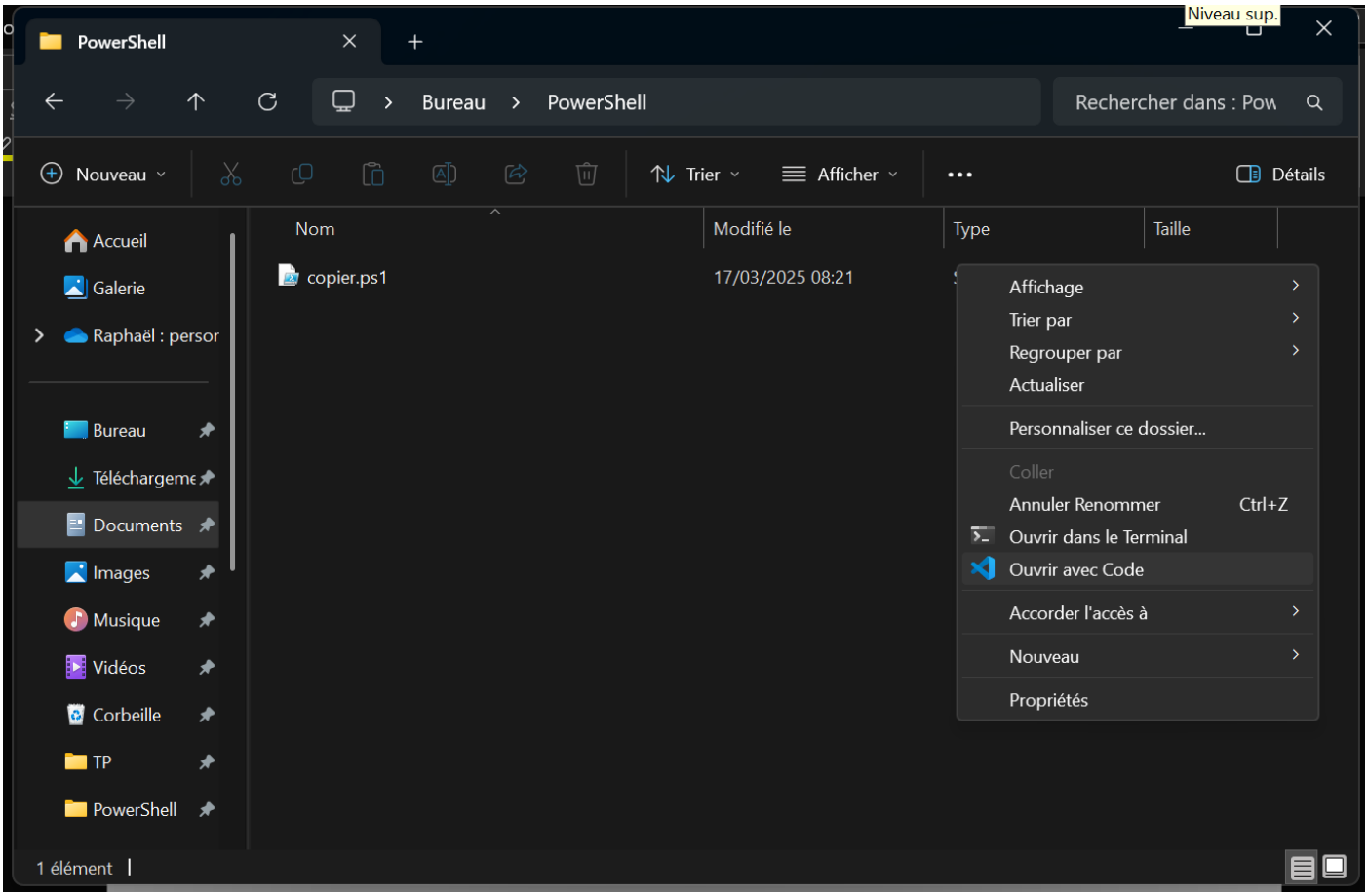
Sommaire

Table des matières

1. Crée un script copiant tous les fichiers de « Mes documents » vers le disque D:\Backup 2
2. Commentez les actions faites par ce script à chaque étape 5
3. Par déduction quelle serai la commande à exécuter afin d'avoir des informations sur un compte « Nom_utilisateur » 5
4. Commentez ligne par ligne des actions faites par ce script 6
5. Commentez ligne par ligne des actions faite par ce script 6
6. Créez un script faisant les actions suivantes 7
7. Comment appeler de l'aide sur une commande en PowerShell ? 11
8. Quelle commande pouvez-vous lancer pour obtenir la liste de tous les sous-dossiers de « C:\Windows » 11
9. Crée une modification du script de la question 2 avec la création d'un compte jean et de test si le compte existe avec un message d'erreur comme quoi le compte existe déjà 12

1. Crée un script copiant tous les fichiers de « Mes documents » vers le disque D:\Backup

Pour créer un script premièrement, il est important de créer un fichier .ps1. Ici nous allons l'appeler « copier.ps1 » et nous allons l'éditer avec Visual Studio Code.



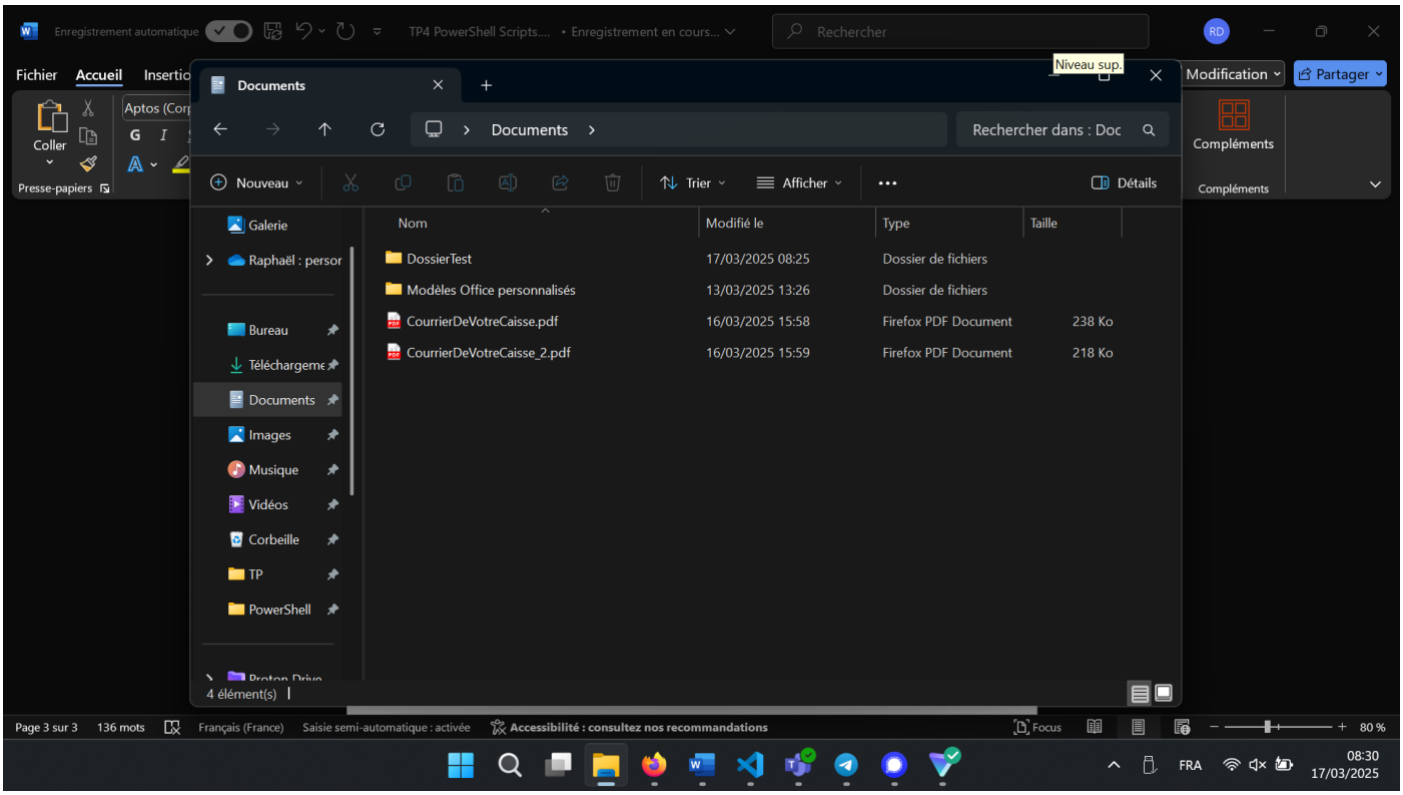
Dans notre fichier, nous aurons besoin de l'applet Copy-Item permettant de copier des éléments en PowerShell. Voici un script créer permettant de réaliser la copie de tous les fichiers de Mes Documents vers D:\Backup

```
1  Copy-Item C:\Users\$env:USERNAME\Documents\* D:\Backup
```

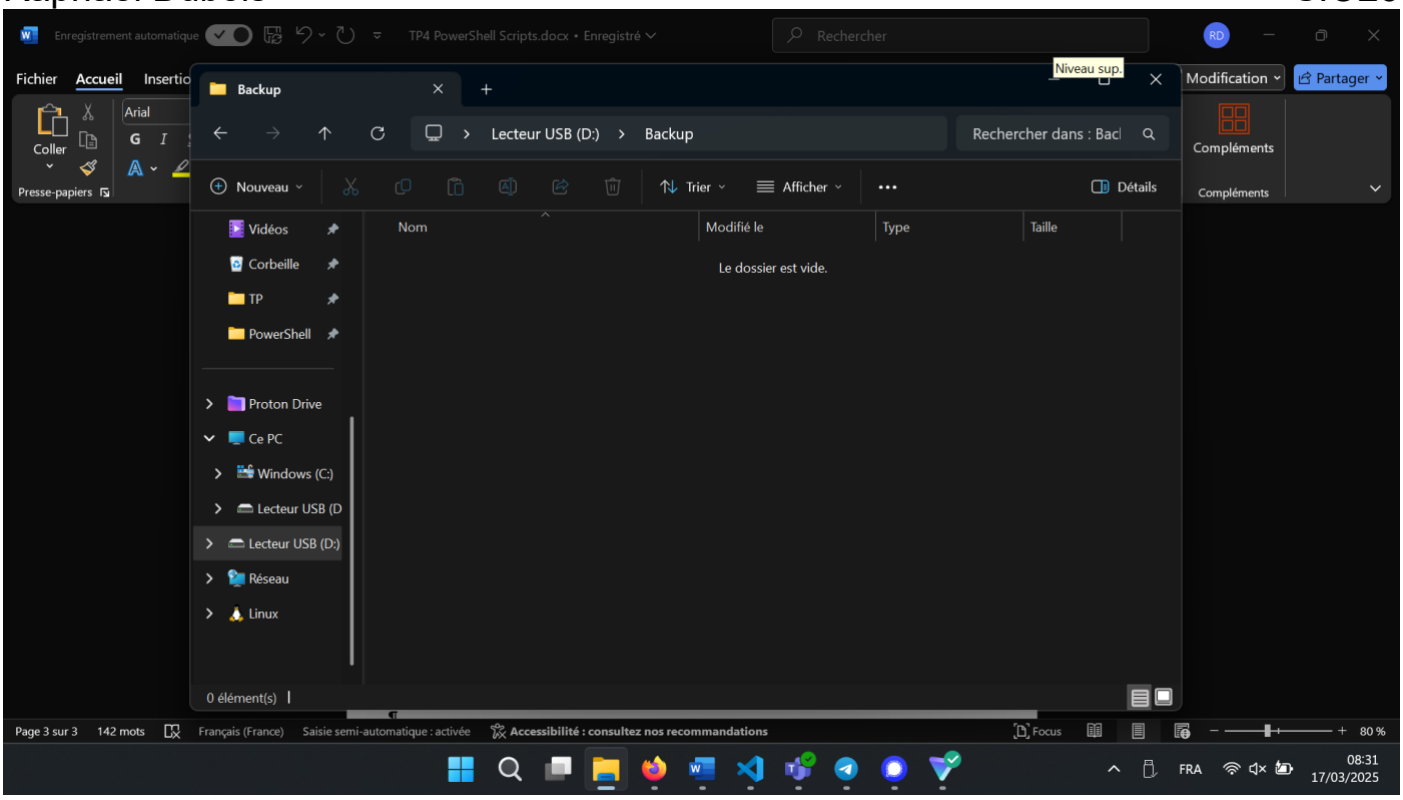
Ce script copie le contenue de C:\Users\(Nom d'utilisateur obtenu automatiquement via la variable d'environnement « USERNAME »)\Documents* D:\Backup

L'étoile à la fin du chemin à copier signifie que l'on souhaite copier tous les fichiers et dossiers présents dans ce répertoire.

Contenu de C:\Users\rdubois\Documents avant exécution :



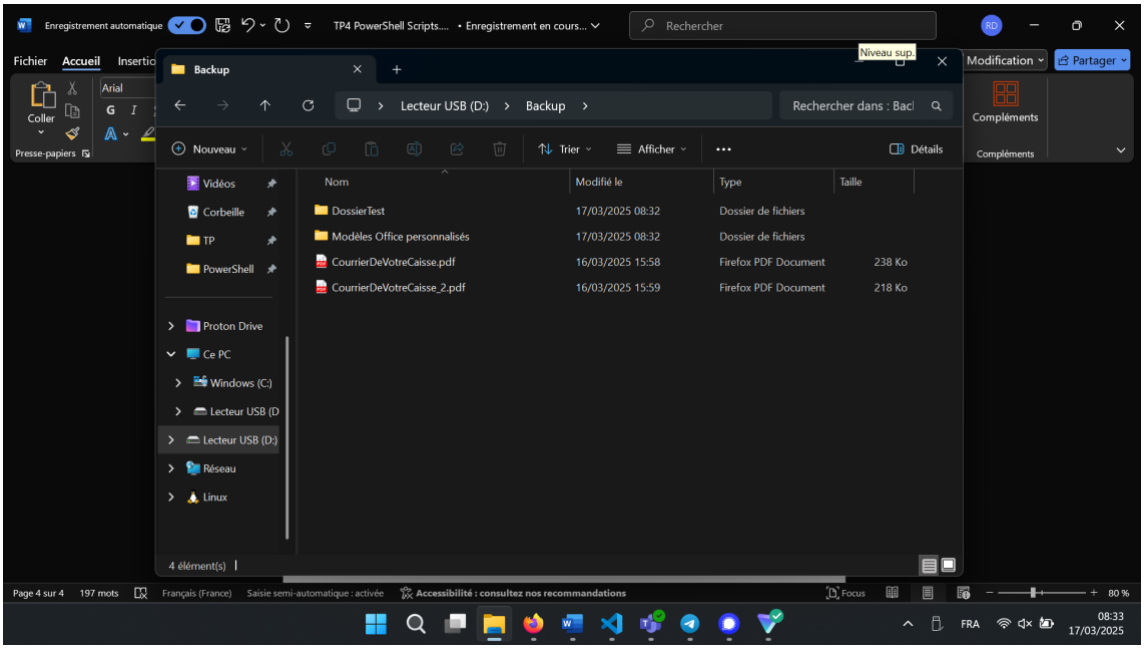
Contenu de D:\Backup avant exécution :



Maintenant nous allons exécuter le script via Visual Studio Code :

```
PS C:\Users\rdubois\Desktop\PowerShell> . 'C:\Users\rdubois\Desktop\PowerShell\copier.ps1'
PS C:\Users\rdubois\Desktop\PowerShell>
```

A première vu rien ne s'est passé mais cela est normal, la tâche c'est réalisé directement au lancement du script, si l'on vérifie le contenu de D:\Backup cette fois ci, on pourra remarquer que tous les dossiers et fichiers ont bien été copiés :



2. Commentez les actions faites par ce script à chaque étape

Création d'un utilisateur « Nom _Utilisateur » avec comme description « Description_utilisateur » avec comme mot de passe chiffré « Mot_de_passe ».

```
New-LocalUser -Name "Nom_utilisateur" -Description "Description_utilisateur" -Password
(ConvertTo-SecureString "Mot_de_passe" -AsPlainText -Force)
```

Modification de la description de l'utilisateur local en « Nouvelle_description ».

```
Set-LocalUser -Name "Nom_utilisateur" -Description "Nouvelle_description"
```

Suppression de l'utilisateur « Nom_utilisateur ».

```
Remove-LocalUser -Name "Nom_utilisateur"
```

3. Par déduction quelle serai la commande à exécuter afin d'avoir des informations sur un compte « Nom_utilisateur »

Par déduction, la commande à exécuter afin d'avoir les informations sur le compte « Nom_utilisateur » serait la suivante :



4. Commentez ligne par ligne des actions faites par ce script

Voici les différentes actions de ce script commenté à chaque étape :

```powershell
# Affiche les différents processus en cours d'exécution, avec leur ID, nom, CPU et mémoire utilisée sous forme de tableau
Get-Process | Format-Table Id, ProcessName, CPU, Memory -AutoSize

# Créer une variable $processId qui demande à l'utilisateur d'entrer l'ID du processus qu'il souhaite terminer
# La valeur de l'ID est stockée dans la variable $processId
$processId = Read-Host "Entrez l'ID du processus que vous souhaitez terminer"

# Arrête le processus en utilisant la commande Stop-Process avec l'ID du processus et l'option -Force pour forcer l'arrêt
Stop-Process -Id $processId -Force
```

5. Commentez ligne par ligne des actions faite par ce script

```powershell
# Défini la variable $threshold à 80 (seuil en Anglais)
$threshold = 80
# Variable ayant pour valeur les disques du système
$disks = Get-PSDrive -PSProvider FileSystem | Where-Object {$_.Free -ne $null}

# Création d'une boucle foreach
foreach ($disk in $disks) {
    # Calcul de l'utilisation du disque en pourcentage
    $usage = ($disk.Used / $disk.Free) * 100
    # Condition vérifiant si l'utilisation du disque est supérieure au seuil
    if ($usage -gt $threshold) {
        # Création d'une variable $subject qui aura pour rôle l'envoie d'un email avec cet objet
        $subject = "Alerte d'utilisation du disque"
        # Création d'une variable $body qui aura pour rôle l'envoie d'un email avec ce message
        $body = "L'utilisation du disque $($disk.Name) a dépassé le seuil de $threshold%."
        # Envoie d'un email avec toutes les informations précèdentes regroupés dans les variables $subject et $body à admin@example.com
        # En passant par le serveur SMTP suivant : smtp.example.com
        Send-MailMessage -To "admin@example.com" -From "alert@example.com" -Subject $subject -Body $body -SmtpServer "smtp.example.com"
    }

    # Le nom du disque est stocké dans la variable $name
    $name = $disk.Name
    # La taille du disque est stockée dans la variable $size
    $size = [math]::Round($disk.Used / 1GB, 2)
    # L'espace libre sur le disque est stocké dans la variable $free
    $free = [math]::Round($disk.Free / 1GB, 2)
    # L'espace utilisé sur le disque est stocké dans la variable $used
    $used = [math]::Round($disk.Used / 1GB, 2)
    # L'utilisation du disque en pourcentage est stockée dans la variable $usage
    $usage = [math]::Round($usage, 2)
    # Les informations sur le disque sont ensuite affichés sur le terminal via les variables $name, $size, $free, $used et $usage
    Write-Host "$name - Taille: $size Go, Libre: $free Go, Utilisé: $used Go, Utilisation: $usage%"
}
```

Le but principal de ce script est d'envoyer un mail dès lors qu'un disque sur le système atteint 80% d'espace libre. Ce script réalise plusieurs opérations, créer différentes variables et envoie les informations par mail via le protocole SMTP.

6. Créez un script faisant les actions suivantes

- Crée un dossier dans C: nommé « Apps »
- Dans ce dossier créez un fichier Texte « info.txt »
- Écrivez dans ce fichier texte :
- En ligne 1 la date du jour
- En ligne 2 le nom de l'ordinateur qui exécute ce script
- En ligne 3 l'espace du disque « C: » de cet ordinateur
- En ligne 4 la RAM totale de cet ordinateur
- En ligne 5 le CPU de cet ordinateur
- En ligne 6 le compte utilisateur actuel de cet ordinateur

Ressources utilisées pour créer le script :

- https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/add-content?view=powershell-7.5 (Commande Add-Content)
- https://www.it-connect.fr/comment-obtenir-lespace-disque-restant-en-powershell/ (Obtenir espace disque)
- https://www.lecoindunet.com/details-ram-windows (Obtenir RAM)
- https://powershellcommands.com/powershell-get-cpu-info (Obtenir info CPU)
- https://ciksiti.com/fr/chapters/16835-how-to-get-the-current-username-in-windows-powershell (Obtenir nom d'utilisateur)

Voici le script créer effectuant toutes les tâches demandées :

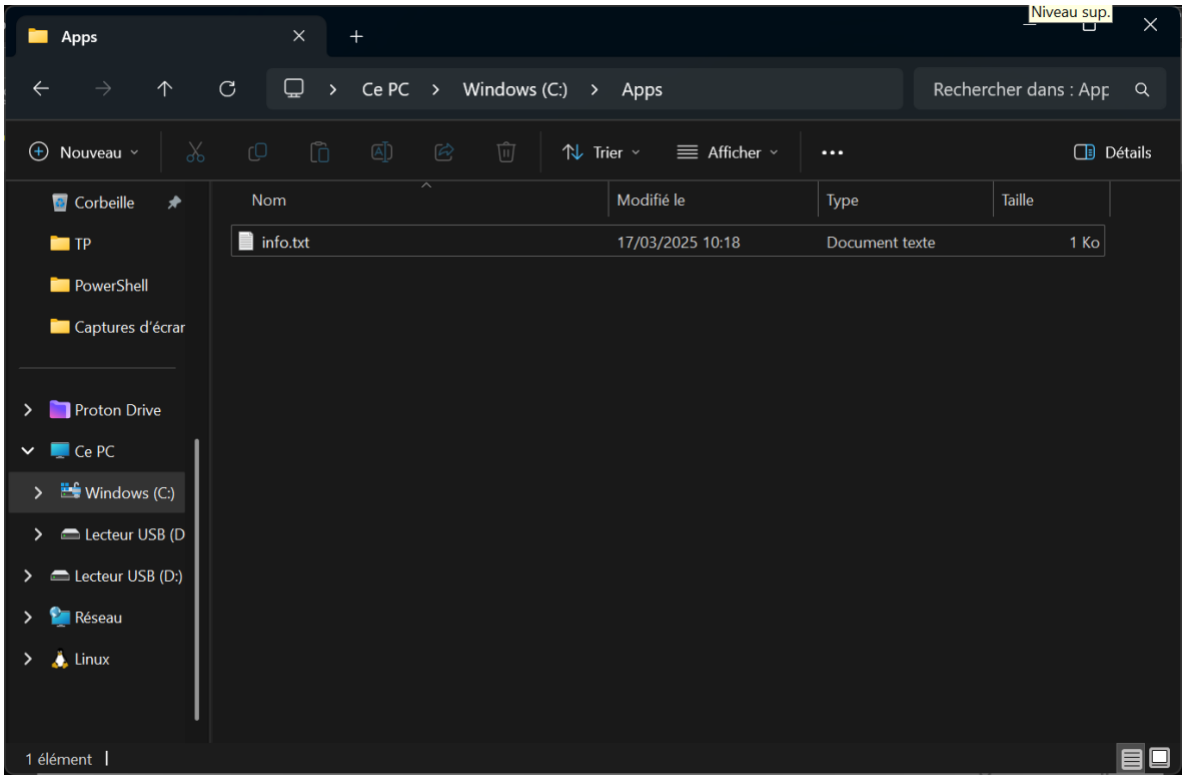
```powershell
# Variable contenant le nom du dossier et du fichier
$folder = "C:\Apps"
$file = "C:\Apps\info.txt"

# Création du dossier et fichier
New-Item -ItemType Directory -Path $folder
New-Item -ItemType File -Path $file

# Ajout de la date au format yyy-MM-dd
$date = Get-Date -Format "yyyy-MM-dd"
Add-Content -Path $file -Value $date

# Ajout du nom de la machine
$hostname = hostname
Add-Content -Path $file -Value $hostname

# Ajout de l'espace du disque C:\ en Giga
$diskSize = (Get-WmiObject -Class Win32_LogicalDisk -Filter "DeviceID='C:'").Size / 1GB
Add-Content -Path $file -Value $diskSize

# Ajout de la capacité de la RAM
$memoryCapacity = (Get-CimInstance -ClassName Win32_PhysicalMemory).Capacity / 1GB
Add-Content -Path $file -Value $memoryCapacity

# Ajout du nom du CPU
$cpuName = (Get-WmiObject -Class Win32_Processor).Name
Add-Content -Path $file -Value $cpuName

# Ajout du nom d'utilisateur
$username = (whoami).Split('\')[1]
Add-Content -Path $file -Value $username
```

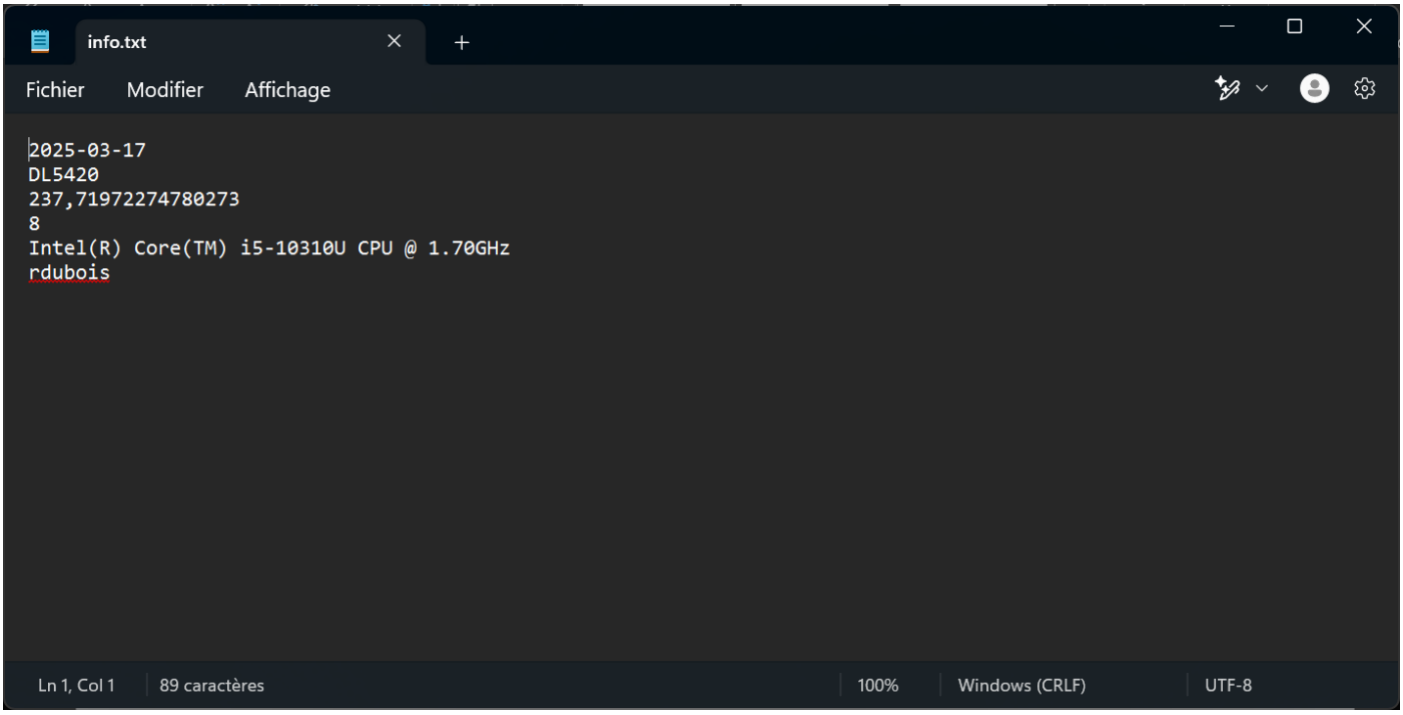
Ce dernier va tout d'abord créer un dossier puis un fichier C:\Apps\info.txt et va ensuite ajouter les informations nécessaires au fur et à mesure de son exécution, voici ce que nous avons en sorti :

```
PS C:\Users\rdubois\Desktop\PowerShell>
PS C:\Users\rdubois\Desktop\PowerShell> . 'C:\Users\rdubois\Desktop\PowerShell\copier.ps1'

    Directory: C:\

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          17/03/2025    10:18                Apps

    Directory: C:\Apps

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          17/03/2025    10:18              0 info.txt

PS C:\Users\rdubois\Desktop\PowerShell>
```

A première vue, nous n'avons rien de spécialement intéressant mis à part le dossier et fichier qui a été bien été créer, mais si l'on essaie d'ouvrir le fichier avec un Get-Content ou un éditeur de texte, on peut observer que les informations ont bien été enregistré dans le fichier :

```
PS C:\Users\rdubois\Desktop\PowerShell> Get-Content C:\Apps\info.txt
2025-03-17
DL5420
237,71972274780273
8
Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz
rdubois
PS C:\Users\rdubois\Desktop\PowerShell>
```

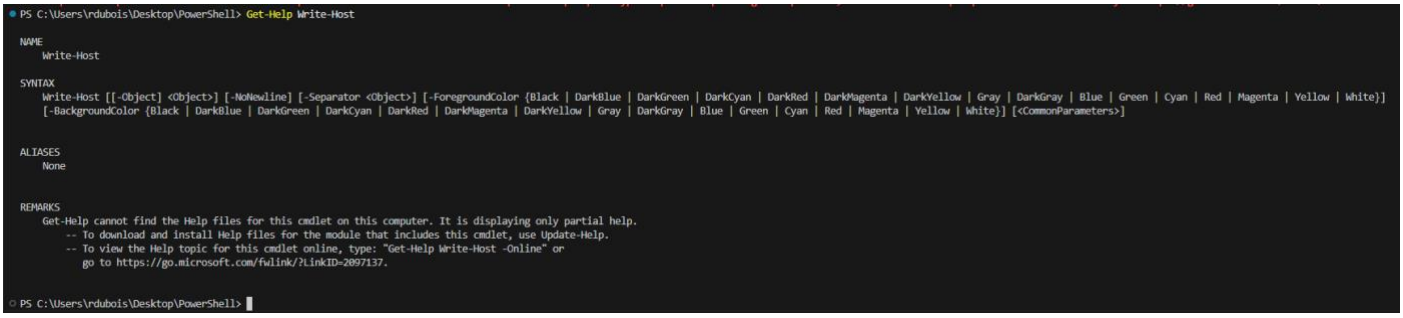




Les informations sur le stockage de C:\ et de la capacité de la RAM sont exprimés en GB, grâce au / 1GB indiqué à la fin de ces deux commandes.

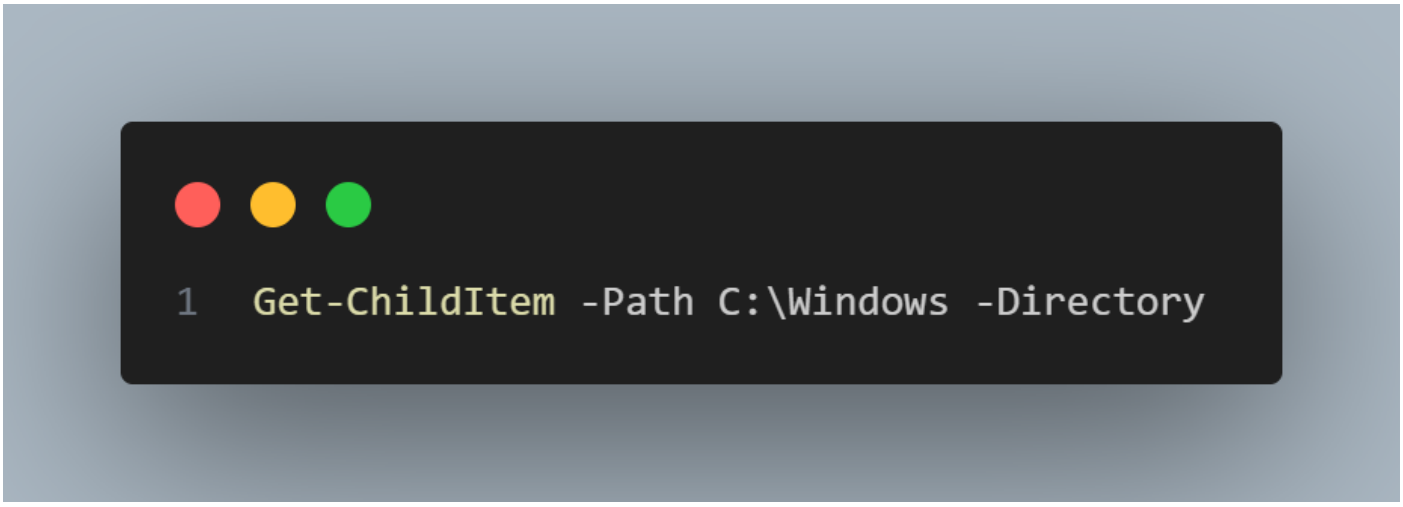
7. Comment appeler de l'aide sur une commande en PowerShell ?

La commande Get-Help permet d'obtenir de l'aide sur une commande en PowerShell. Par exemple, la commande « Get-Help Write-Host » permet d'avoir de l'aide sur la commande « Write-Host » :



8. Quelle commande pouvez-vous lancer pour obtenir la liste de tous les sous-dossiers de « C:\Windows »

Pour lister les sous dossiers de C:\Windows, on peut utiliser la commande suivante :



- **Get-ChildItem** : Cette commande permet d'afficher le contenu d'un dossier (similaire à dir en cmd).
- **-Path C:\Windows** : Spécifie le dossier dont on veut lister le contenu. En l'occurrence : C:\Windows.

- **-Directory** : Filtre pour afficher uniquement les dossiers (et non les fichiers).

9. Crée une modification du script de la question 2 avec la création d'un compte jean et de test si le compte existe avec un message d'erreur comme quoi le compte existe déjà

Ressources utilisées pour créer le script :

- https://stackoverflow.com/questions/49595003/checking-if-a-local-user-account-group-exists-or-not-with-powershell (Vérification si l'utilisateur existe ou non)
- https://stackoverflow.com/questions/64633727/how-to-fix-running-scripts-is-disabled-on-this-system (Activation de l'exécution des scripts en tant qu'Administrateur)

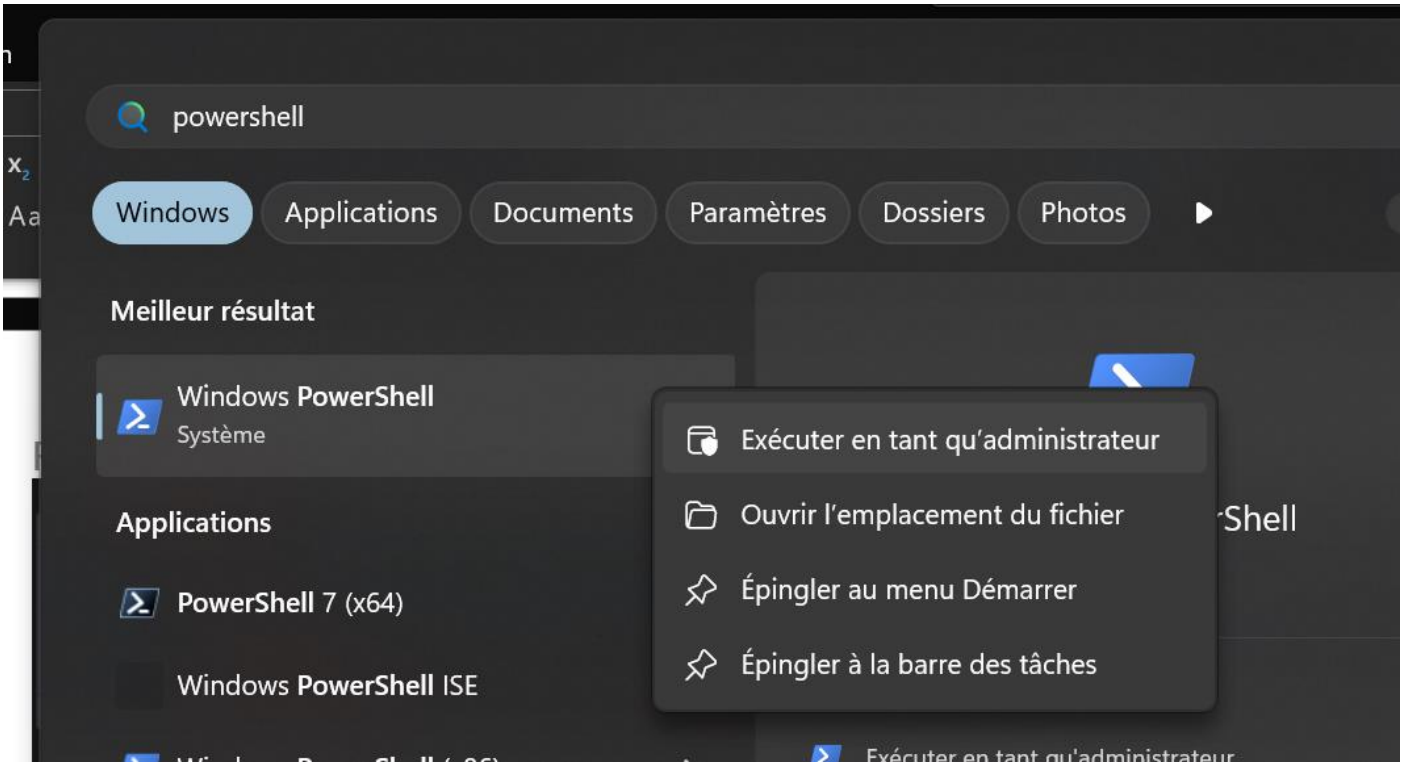
Ce script aura pour rôle de créer deux comptes, un compte jean et un compte test en PowerShell. Voici le code qui a été réalisé pour vérifier si l'utilisateur existe avec des conditions (if, else) et la création de ces comptes. Pour cela nous allons nous aider de la commande Get-LocalUser nous permettant d'avoir des informations sur les utilisateurs présents sur la machine et notamment s'ils sont présents ou non :

```powershell
# Compte de Jean
$op1 = Get-LocalUser | where-Object Name -eq "jean" | Measure
if ($op1.Count -eq 0) {
    New-LocalUser -Name "jean" -Description "Compte de Jean" -Password (ConvertTo-SecureString "Mot_de_passe" -AsPlainText -Force)
    Write-Host "Le compte a été créé : jean"
} else {
    Write-Host "Le compte existe déjà"
}

# Compte de Test
$op2 = Get-LocalUser | where-Object Name -eq "test" | Measure
if ($op2.Count -eq 0) {
    New-LocalUser -Name "test" -Description "Compte de Test" -Password (ConvertTo-SecureString "Mot_de_passe" -AsPlainText -Force)
    Write-Host "Le compte a été créé : test"
} else {
    Write-Host "Le compte existe déjà"
}
```

Si l'on tente d'exécuter le script directement en tant qu'utilisateur, PowerShell nous répondra « Accès refusé ». Ce qui est normal étant donné que nous sommes qu'utilisateur et non administrateur :

```
PS C:\Users\rdubois\Desktop\PowerShell> . 'C:\Users\rdubois\Desktop\PowerShell\powershell.ps1'
New-LocalUser: C:\Users\rdubois\Desktop\PowerShell\powershell.ps1:3:5
Line |
   3 |      New-LocalUser -Name "jean" -Description "Compte de Jean" -Passwor …
     |      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     | Accès refusé.
Le compte a été créé
PS C:\Users\rdubois\Desktop\PowerShell>
```

Pour corriger cela, nous allons lancer un PowerShell en tant qu'administrateur en faisant clic droit sur l'icône de PowerShell puis exécuter en tant qu'administrateur :



Puis nous allons nous rendre dans le répertoire du script avec un cd :

```
PS C:\Users\rdubois\Desktop> cd .\PowerShell\
PS C:\Users\rdubois\Desktop\PowerShell> ls


    Répertoire : C:\Users\rdubois\Desktop\PowerShell


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         17/03/2025     10:18           1011 copier.ps1
-a----         17/03/2025     10:43            636 powershell.ps1
```

Si l'on essaie d'exécuter ensuite notre script en faisant un .\powershell.ps1, le terminal nous répondra avec une erreur indiquant que nous ne sommes pas autorisées à exécuter des scripts :

```
PS C:\Users\rdubois\Desktop\PowerShell> .\powershell.ps1
.\powershell.ps1 : Impossible de charger le fichier C:\Users\rdubois\Desktop\PowerShell\powershell.ps1, car
l'exécution de scripts est désactivée sur ce système. Pour plus d'informations, consultez
about_Execution_Policies à l'adresse https://go.microsoft.com/fwlink/?LinkID=135170.
Au caractère Ligne:1 : 1
+ .\powershell.ps1
+ ~~~~~~~~~~~~~~~~
    + CategoryInfo          : Erreur de sécurité : (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\rdubois\Desktop\PowerShell>
```

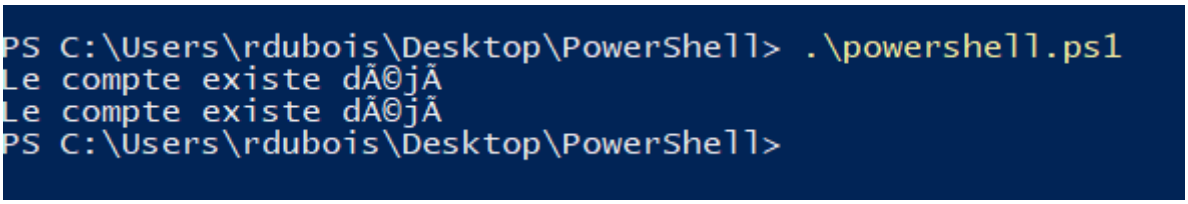
Pour corriger cela, il suffit de lancer la commande « Set-ExecutionPolicy -ExecutionPolicy RemoteSigned » et puis de faire « Oui » :

```
Modification de la stratégie d'exécution
La stratégie d'exécution permet de vous prémunir contre les scripts que vous jugez non fiables. En modifiant la stratégie d'exécution, vous vous exposez aux risques de sécurité décrits dans
la rubrique d'aide about_Execution_Policies à l'adresse https://go.microsoft.com/fwlink/?LinkID=135170. Voulez-vous modifier la stratégie d'exécution ?
[O] Oui  [T] Oui pour tout  [N] Non  [U] Non pour tout  [S] Suspendre  [?] Aide (la valeur par défaut est « N ») : O
```

Une fois ceci fait, nous pouvons exécuter notre script et observer que celui-ci fonctionne :

```
PS C:\Users\rdubois\Desktop\PowerShell> .\powershell.ps1

Le compte a Ã©tÃ© crÃ©Ã© : jean
Le compte a Ã©tÃ© crÃ©Ã© : test
Name Enabled Description
---- ------- -----------
jean True    Compte de Jean
test True    Compte de Test
```

Si l'on tente d'exécuter à nouveau le script, on se rend compte qu'il y a une erreur, ce qui est normal étant donné que nous avons ajouté des conditions pour que le script ne créer pas de nouveau utilisateur si ce dernier existe déjà :


PS C:\Users\rdubois\Desktop\PowerShell> .\powershell.ps1
Le compte existe dÃ©jÃ
Le compte existe dÃ©jÃ
PS C:\Users\rdubois\Desktop\PowerShell>