

Scripts : Introduction à PowerShell – Module Active Directory

1. Introduction.....	3
1.1. Notion de scripts.....	3
1.2. Pourquoi utiliser les scripts.....	3
1.3. Microsoft et l'arrivée de Powershell.....	3
2. Prise en main.....	4
2.1. Découverte de la console ligne de commandes.....	4
2.2. L'environnement d'écriture de scripts intégré (interface ISE).....	5
2.3. Transition avec le passé.....	7
3. Les commandes de base.....	9
3.1. Constitution des commandes.....	9
3.2. Get-Command.....	10
3.3. Get-Help.....	12
3.4. Get-Member et le concept des objets.....	12
4. Navigation dans les répertoires et les fichiers.....	14
4.1. Les nouvelles commandes.....	14
4.2. Get-ChildItem (Alias : gci, ls, dir).....	14
4.3. Set-Location (Alias : sl, cd, chdir).....	19
4.4. Get-Location (Alias : gl, pwd).....	20
4.5. New-Item (Alias : ni, md).....	21
4.5.1. Création d'un répertoire.....	21
4.5.2. Création d'un fichier.....	22
4.6. Remove-Item (Alias : ri, rm, rmdir, rd, erase, del).....	22
4.7. Move-Item (Alias : mi, move, mv).....	23
4.7.1. Déplacement de fichiers.....	24
4.7.2. Déplacement d'un répertoire.....	24
4.8. Rename-Item (Alias : ren, rni).....	25
4.8.1. Renommer un fichier.....	25
4.8.2. Renommer un dossier.....	25
4.9. Copy-Item (Alias : cpi, cp, copy).....	26
5. La boucle Foreach.....	27
6. Module Active Directory.....	28
6.1. Mise en route du module.....	28
6.2. Gestion des utilisateurs.....	29
6.2.1. Créer des utilisateurs.....	30
6.2.2. Affecter un mot de passe à la création.....	31
6.2.3. Activer un compte à la création.....	31
6.3. Gestion des groupes.....	32

6.3.1. Créer des groupes.....	32
6.3.2. Ajouter des membres à un groupe.....	33
7. TP : Création des comptes utilisateurs par lots.....	34

1. Introduction.

1.1. Notion de scripts

Un script est un simple fichier texte ASCII dans lequel s'enchaînent toutes les instructions qui le composent. La différence entre un langage de script et un langage de programmation tient au fait qu'un script n'est pas compilé. Il n'est pas transformé en un binaire directement exécutable par la machine. Il faut obligatoirement un interpréteur de commandes appelé « shell » pour exécuter le script.

1.2. Pourquoi utiliser les scripts.

Les administrateurs systèmes utilisent des scripts pour automatiser la réalisation des tâches fastidieuses. On peut citer, à titre d'exemple la configuration d'un serveur (nom, paramètres IP, ajout de rôles et leur configuration) ou la création manuelle d'une centaine de comptes utilisateurs ou plus. Outre le temps passé nécessaire à la réalisation de cette dernière opération, on peut aisément imaginer le risque d'erreurs qu'elle comporte (nom mal orthographié, oubli de créer l'espace home directory, permissions mal positionnées, etc.).

1.3. Microsoft et l'arrivée de Powershell

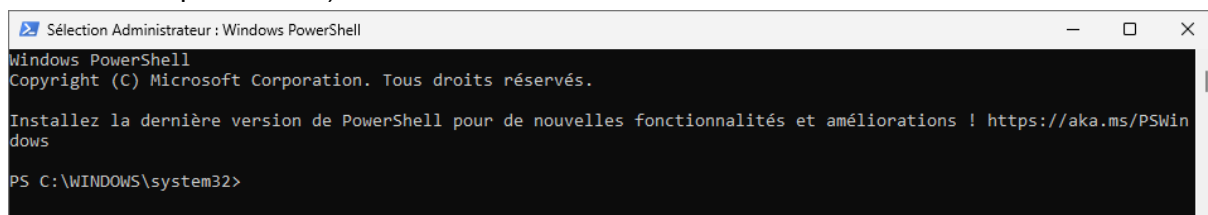
PowerShell est à la fois un interpréteur de commandes et un puissant langage de scripts. Microsoft, aux alentours de 2004-2005, prit conscience des limites de son système d'exploitation basé presque exclusivement sur l'interface graphique et des difficultés éprouvées par les utilisateurs pour la réalisation de scripts contrairement aux utilisateurs Linux (cmd.exe depuis 1993 avec la sortie de Windows NT qui remplaça les OS basés sur MS-DOS). Aussi, Microsoft livra la version 1.0 de Powershell au public en 2006. Windows PowerShell 7, la dernière version de la version de PowerShell réservée à Windows, est publiée en 2020 (version 7.3.4 en avril 2023). La nouvelle stratégie de Microsoft pour l'administration système de ses produits est maintenant clairement orientée ligne de commandes. Certes les interfaces graphiques des outils d'administration, qui ont contribué au succès de Windows, demeureront mais celles-ci seront dorénavant construites au-dessus de PowerShell. Aujourd'hui, PowerShell est profondément ancré dans les systèmes Windows Server, si bien que depuis Windows Server 2008 R2 jusqu'à Windows Server 2022, chaque nouveau rôle installé apporte avec lui des nouvelles commandes. On dispose ainsi, à présent, de commandes pour gérer l'Active Directory, les stratégies de groupes, IIS, la gestion des rôles et fonctionnalités, etc. Mais PowerShell est également au cœur de nombreux produits Microsoft : Exchange Server, SQL Server, Microsoft Deployment Toolkit, etc. Ainsi, dès que l'on veut installer un nouveau rôle ou une application sur un serveur Windows, il y a un jeu de commandes associé.

2. Prise en main.

Depuis la version 2.0 il existe un très pratique éditeur de scripts PowerShell en mode graphique, mais dans un premier temps, intéressons-nous à la console « classique ».

2.1. Découverte de la console ligne de commandes

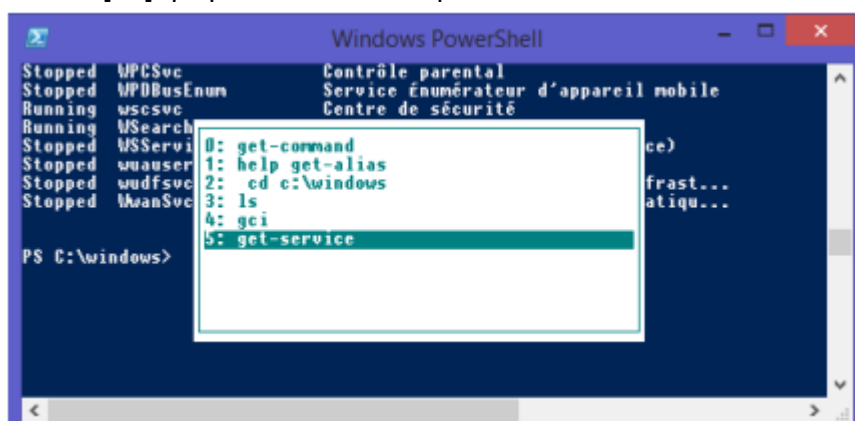
Au premier coup d'œil, rien ne permet de distinguer une fenêtre « console PowerShell » d'une fenêtre « invite de commande CMD.exe », si ce n'est la couleur de fond (bleue pour l'une et noire pour l'autre).



Touches qui permettent de « naviguer » dans la console :

Touche	Description
[Tab]	Réalise la complétion automatique d'un chemin, nom de commande, paramètre ou une valeur de paramètre (si le type est une valeur Enum).
[Flèche haut]/[Flèche bas]	Fait défiler l'historique des commandes déjà frappées.
[F7]	Affiche une boîte contenant l'historique des commandes. La sélection s'effectue à l'aide des flèches.
[F8]	Fait défiler l'historique sur la ligne de commande.

Touche [F7] qui permet en un coup d'œil de retrouver une commande dans l'historique :



- Pour savoir quelle version est installée sur votre poste de travail ou serveur, nous tapons la commande `$PSVersionTable` dans la console :

```
PS C:\WINDOWS\system32> $PSVersionTable

Name                           Value
----                           -
PSVersion                      5.1.22621.4391
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.22621.4391
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1

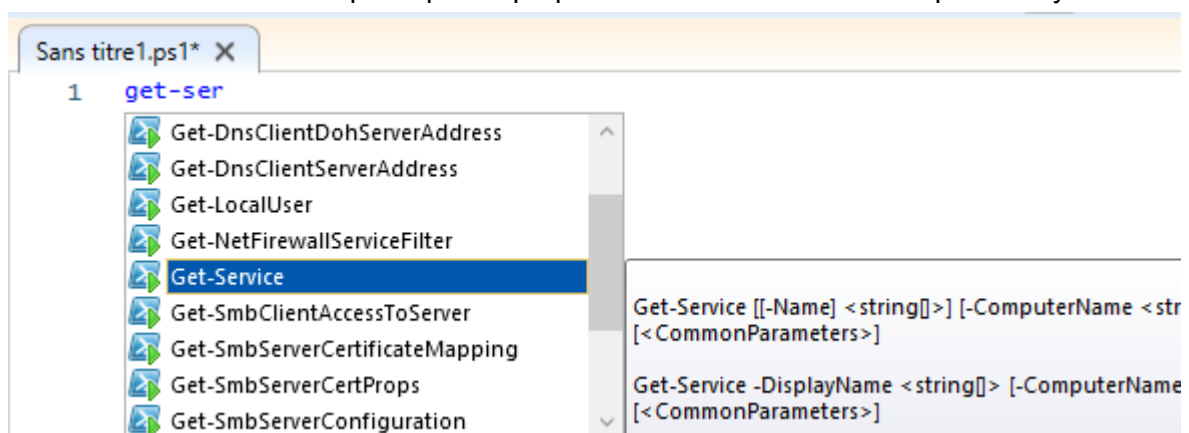
PS C:\WINDOWS\system32>
```

2.2. L'environnement d'écriture de scripts intégré (interface ISE)

Un éditeur de scripts est disponible depuis PowerShell v2 avec la coloration syntaxique, l'affichage des numéros de lignes, le débogueur intégré, l'aide en ligne en mode graphique, etc. On a également la possibilité d'ouvrir une console PowerShell sur une machine à distance directement dans l'éditeur.

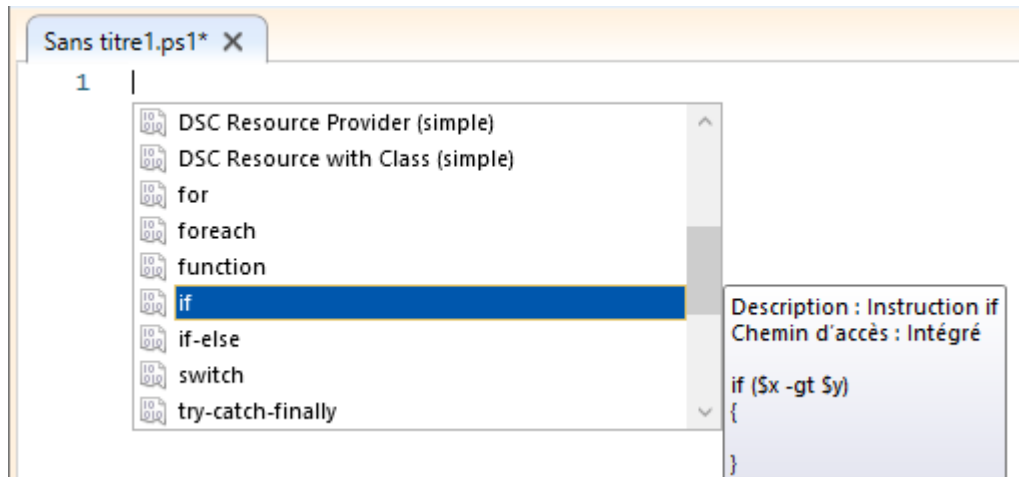
Dans le volet supérieur gauche se trouve l'éditeur de script dans lequel vont se logger des onglets. Cela permet de travailler sur plusieurs scripts à la fois. Celui du dessous quant à lui permet de saisir directement des commandes interactives, comme dans la console bleue originelle.

- Avec la fonctionnalité « IntelliSense », il suffit à présent de commencer à frapper au clavier le début d'une commande pour qu'une proposition soit faite tout en indiquant la syntaxe :

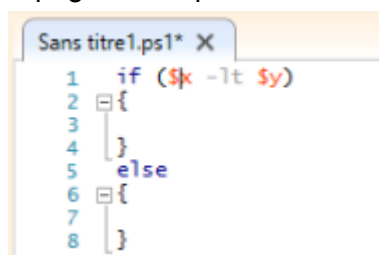


La fonctionnalité IntelliSense permet de gérer un ensemble de propositions qui sont déclenchées lorsque l'on utilise notamment les caractères suivants : - Tirés - derrière le verbe d'une commande comme Get- ou avant de saisir le nom d'un paramètre de commande, comme dans Get-Command - ; - Point . après un nom de variable, exemple \$profile. Notez que dans le cas où vous refuseriez l'aide apportée par l'IntelliSense en appuyant sur [Echap], il vous est possible de la faire réapparaître avec la combinaison de touches [Ctrl][Espace].

- Nous saisissons la combinaison [Ctrl] J et une liste déroulante apparaît pour choisir un élément de code :



Une fois sélectionné avec la touche [Entrée], le code PowerShell s'inscrit directement dans la page de script :



2.3. Transition avec le passé

Toutes les commandes qui étaient incluses dans CMD le sont aussi dans PowerShell. Certaines le sont sous forme d'alias ou de fonctions.

- Saisie par exemple la commande dir :

```
PS C:\WINDOWS\system32> dir

Répertoire : C:\WINDOWS\system32

Mode                LastWriteTime         Length Name
----                -
d-----          07/05/2022   12:25             0409
d-----          07/05/2022    07:24      AdvancedInstallers
d-----          07/05/2022    07:24      AppLocker
d-----          21/03/2025   12:52      appraiser
d---s-          24/02/2025   17:56      AppV
```

- De la même manière, saisissons la commande dir dans l'invite de commande CMD :

```
PS C:\WINDOWS\system32> dir

Répertoire : C:\WINDOWS\system32

Mode                LastWriteTime         Length Name
----                -
d-----          07/05/2022   12:25             0409
d-----          07/05/2022    07:24      AdvancedInstallers
d-----          07/05/2022    07:24      AppLocker
d-----          21/03/2025   12:52      appraiser
d---s-          24/02/2025   17:56      AppV
d-----          24/02/2025   17:56      ar-SA
d-----          24/02/2025   17:56      bg-BG
d-----          21/03/2025   12:52      Boot
d-----          07/05/2022    07:24      Bthprops
d-----          24/02/2025   17:56      ca-ES
d-----          03/03/2025   11:14      CatRoot
```

On peut constater qu'au premier abord la différence n'est pas flagrante. Pour les tâches courantes, telles que la navigation dans les répertoires et les fichiers, nous n'avons donc pas vraiment besoin de connaître les vraies commandes PowerShell qui se cachent derrière les alias. Exemples d'anciennes commandes que l'on peut réutiliser dans PowerShell : dir, md, cd, rd, move, ren, copy. Les Unixiens ne sont pas perdus non plus car la plupart des commandes Unix de base fonctionnent grâce aux alias PowerShell, tels que : ls, mkdir, cp, mv, pwd, cat, mount, ps, etc.

- Pour connaître la liste complète des alias disponibles, nous tapons la commande suivante : Get-Alias.

```
PS C:\WINDOWS\system32> Get-Alias
```

CommandType	Name	Version	Source
Alias	% -> ForEach-Object		
Alias	? -> Where-Object		
Alias	ac -> Add-Content		
Alias	asnp -> Add-PSSnapin		
Alias	cat -> Get-Content		
Alias	cd -> Set-Location		
Alias	CFS -> ConvertFrom-String	3.1.0.0	Microsoft.PowerShell.Utility
Alias	chdir -> Set-Location		
Alias	clic -> Clear-Content		

- Pour les fonctions, nous tapons la commande suivante : Get-Command -CommandType function.

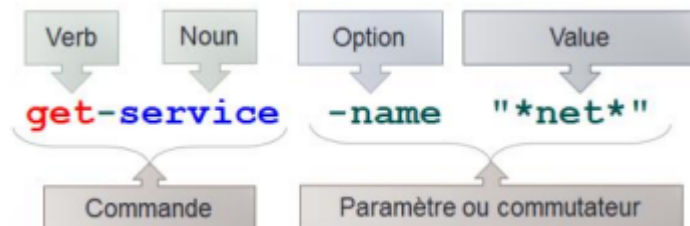
```
PS C:\WINDOWS\system32> get-command -CommandType Function
```

CommandType	Name	Version	Source
Function	A:		
Function	Add-BCDataCacheExtension	1.0.0.0	BranchCache
Function	Add-BitLockerKeyProtector	1.0.0.0	BitLocker
Function	Add-DnsClientDohServerAddress	1.0.0.0	DnsClient
Function	Add-DnsClientNrptRule	1.0.0.0	DnsClient
Function	Add-DtcClusterTMMapping	1.0.0.0	MsDtc
Function	Add-EtwTraceProvider	1.0.0.0	EventTracingManagement
Function	Add-InitiatorIdToMaskingSet	2.0.0.0	Storage
Function	Add-MpPreference	1.0	ConfigDefender
Function	Add-MpPreference	1.0	Defender
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-NetEventVFPPProvider	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-NetEventWFPCaptureProvider	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

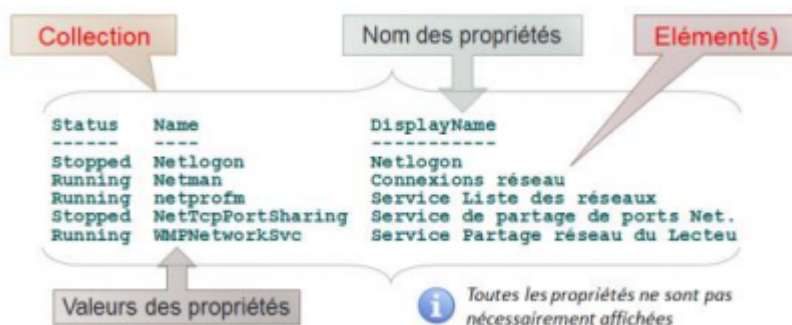
3. Les commandes de base.

3.1. Constitution des commandes

Les commandes de PowerShell sont appelées « cmdlets » (pour command-applets) ou commandelettes en français. Elles sont constituées de la manière suivante : un verbe et un nom séparés par un tiret : verbe-nom. Le verbe décrit l'action à appliquer sur le nom. La commande `Get-Command` permet par exemple d'obtenir (Get) les commandes (Command). Autre exemple :



Le résultat renvoie une « collection d'objets ». Considérez que c'est une sorte de tableau avec des en-têtes de colonnes (les propriétés), où chaque ligne représente un élément (un objet) :



Avec PowerShell on trouve de nombreux verbes génériques tels que `Get`, `Set`, `Add`, `Remove`, etc. qui se combinent avec différents noms comme `Path`, `Variable`, `Item`, `Object`, `Computer`, etc. Les commandes, ainsi que leurs paramètres associés, peuvent s'écrire indifféremment en majuscules ou en minuscules, l'analyseur de syntaxe PowerShell n'étant pas sensible à la casse.

- Saisie de la commande `Get-Verb` permettant de retrouver la liste complète des verbes officiels et leurs groupes d'applications.

```
PS C:\WINDOWS\system32> Get-Verb

Verb      Group
----
Add       Common
Clear     Common
Close    Common
Copy      Common
Enter     Common
Exit      Common
Find      Common
Format    Common
```

3.2. Get-Command

Cette commande permet de découvrir toutes les commandes PowerShell. Sans préciser de paramètre, Get-Command retourne également les alias et les fonctions.

- Pour l'instant, intéressez-vous uniquement aux commandelettes, et pour cela, nous ajoutons le paramètre -CommandType suivi du type de commandes choisi, à savoir cmdlet :

```
PS C:\WINDOWS\system32> Get-command -CommandType cmdlet
```

CommandType	Name	Version	Source
Cmdlet	Add-AppProvisionedSharedPackageContainer	3.0	Dism
Cmdlet	Add-AppSharedPackageContainer	2.0.1.0	Appx
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

- Nous saisissons la commande suivante :

```
PS C:\WINDOWS\system32> (Get-command -CommandType cmdlet).count
682
PS C:\WINDOWS\system32>
```

Ce résultat est celui obtenu sur un système Windows 11.

- Get-Command possède le paramètre -verb et ce dernier permet de retourner toutes les commandes commençant par un verbe donné. Nous obtenons par exemple toutes les commandes dont le verbe commence par Write :

```
PS C:\WINDOWS\system32> get-command -verb write
```

CommandType	Name	Version	Source
Alias	Write-FileSystemCache	2.0.0.0	Storage
Alias	Write-FileSystemCache	1.0.0.0	VMDirectStorage
Function	Write-DtcTransactionsTraceSession	1.0.0.0	MsDtc
Function	Write-PrinterNfcTag	1.1	PrintManagement
Function	Write-VolumeCache	2.0.0.0	Storage
Cmdlet	Write-Debug	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Error	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-EventLog	3.1.0.0	Microsoft.PowerShell.Management
Cmdlet	Write-Host	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Information	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Output	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Progress	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Verbose	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Write-Warning	3.1.0.0	Microsoft.PowerShell.Utility

- De manière similaire, avec cette fois-ci le paramètre -noun, nous affichons les commandes qui s'appliquent, par exemple, aux objets, c'est-à-dire celles dont la partie nom est object :

```
PS C:\WINDOWS\system32> get-command -noun object
```

CommandType	Name	Version	Source
Cmdlet	Compare-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	ForEach-Object	3.0.0.0	Microsoft.PowerShell.Core
Cmdlet	Group-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Measure-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	New-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Select-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Sort-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Tee-Object	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Where-Object	3.0.0.0	Microsoft.PowerShell.Core

- Affichage des commandes de type alias :

```
PS C:\WINDOWS\system32> get-command -CommandType alias
```

CommandType	Name	Version	Source
Alias	% -> ForEach-Object		
Alias	? -> Where-Object		
Alias	ac -> Add-Content		
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-ProvisionedAppSharedPackageContainer	3.0	Dism
Alias	Add-ProvisionedAppxPackage	3.0	Dism
Alias	Add-ProvisioningPackage	3.0	Provisioning
Alias	Add-TrustedProvisioningCertificate	3.0	Provisioning
Alias	algm ->	1.0.0.0	Microsoft.PowerShell.LocalAccounts
Alias	Apply-WindowsUnattend	3.0	Dism
Alias	asnp -> Add-PSSnapin		
Alias	blsmba ->	2.0.0.0	SmbShare
Alias	blsmbcl as ->	2.0.0.0	SmbShare
Alias	cat -> Get-Content		
Alias	cd -> Set-Location		
Alias	CFS -> ConvertFrom-String	3.1.0.0	Microsoft.PowerShell.Utility
Alias	chdir -> Set-Location		

- Nous recherchons une commande dont on ignore le nom mais sachant qu'elle commence par le verbe Get :

```
PS C:\WINDOWS\system32> get-command get-*
```

CommandType	Name	Version	Source
Alias	Get-AppPackage	2.0.1.0	Appx
Alias	Get-AppPackageAutoUpdateSettings	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-Language	1.0	LanguagePackManagement
Alias	Get-PhysicalDiskSNV	2.0.0.0	Storage

3.3. Get-Help

Cette commande de base permet, comme son nom l'indique, d'obtenir de l'aide sur n'importe quelle commande. Pour demander de l'aide sur une commande, On peut le faire de différentes façons : - Get-Help maCommande ; - Help maCommande ; - maCommande -?. Get-Help maCommande affiche l'aide standard, c'est-à-dire l'aide minimale. Avec PowerShell, il existe trois niveaux d'aide : l'aide standard, l'aide détaillée et l'aide complète. Pour avoir accès à l'aide détaillée, ajoutez le paramètre -Detailed, soit Get-Help maCommande - detailed. Et pour obtenir l'aide complète, spécifiez le paramètre -Full, soit Get-Help maCommande -full.

3.4. Get-Member et le concept des objets

Get-Member est probablement la commande la plus intéressante de toutes car elle retourne toutes les propriétés et méthodes d'un objet ainsi que son type. Avant de parler de la commande Get-Member, on doit présenter le concept des objets. Sous Powershell, tout est objet. Une simple chaine de caractère est par exemple un objet de type [string] doté de propriétés et de méthodes.

Vocabulaire :

Terme	Explication
Classe	C'est un peu comme le "Plan de construction" aussi appelé "Type" d'objets
Instance	Objet existant (élément) construit selon une classe
Collection	Désigne un ensemble d'instances (collection d'objets) généralement d'un même type
Membres	Ce sont les caractéristiques (Propriétés) et les capacités (Méthodes) d'un objet.

Illustration :



Une classe de type "Boite", va nous permettre de construire une ou plusieurs boites similaires (instance ou objet). Nous pourrons ensuite consulter (voir modifier) les caractéristiques (Propriétés) telles que la taille, la couleur, etc. De la même manière, nous pouvons agir sur une boite et ses capacités (Méthodes) telles que l'ouvrir, la fermer, etc... Sur le plan syntaxique, les membres (propriétés ou méthodes) sont liés à l'objet (l'instance) par un point : "Instance.Membre".

- Création de la variable \$maVariable et affectez-lui une valeur de type chaîne (String).

```
PS C:\WINDOWS\system32> $maVariable = 'Bonjour tout le monde !'
PS C:\WINDOWS\system32>
```

Nous n'avons pas besoin de déclarer la variable car PowerShell affecte automatiquement un type en fonction du contenu. Une variable commence toujours par le caractère dollar. Nous souhaitons faire des actions dessus, comme par exemple la convertir en majuscules ou bien compter le nombre de caractères qu'elle contient.

- Nous tapons la commande \$maVariable | get-member :

```
PS C:\WINDOWS\system32> $maVariable | gm

TypeName : System.String

Name      MemberType      Definition
-----
Clone      Method          System.Object Clone(), System.Object ICloneable.Clone()
CompareTo  Method          int CompareTo(System.Object value), int CompareTo(string strB), int IComparable.Co...
Contains   Method          bool Contains(string value)
CopyTo     Method          void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int count)
EndsWith   Method          bool EndsWith(string value), bool EndsWith(string value, System.StringComparison c...
Equals     Method          bool Equals(System.Object obj), bool Equals(string value), bool Equals(string valu...
GetEnumerator Method          System.CharEnumerator GetEnumerator(), System.Collections.IEnumerator IEnumerable...
GetHashCode Method          int GetHashCode()
GetType    Method          type GetType()
GetTypeCode Method          System.TypeCode GetTypeCode(), System.TypeCode IConvertible.GetTypeCode()
IndexOf    Method          int IndexOf(char value), int IndexOf(char value, int startIndex), int IndexOf(stri...
IndexOfAny Method          int IndexOfAny(char[] anyOf), int IndexOfAny(char[] anyOf, int startIndex), int In...
Insert     Method          string Insert(int startIndex, string value)
IsNormalized Method          bool IsNormalized(), bool IsNormalized(System.Text.NormalizationForm normalization...
LastIndexOf Method          int LastIndexOf(char value), int LastIndexOf(char value, int startIndex), int Last...
```

Nous voyons apparaître plusieurs éléments : - Le champ TypeName indique le type de la variable (String). - Une liste de noms de méthodes, de propriétés, et leur définition associée.

- Nous utilisons la méthode ToUpper afin de retourner la chaîne contenue dans \$maVariable en majuscules.

```
PS C:\WINDOWS\system32> $maVariable.ToUpper()
BONJOUR TOUT LE MONDE !
PS C:\WINDOWS\system32>
```

Note : pour invoquer une méthode (sans paramètres), vous devez ajouter les parenthèses à la fin “.Method()”

- De la même façon, nous utilisons la propriété Length afin d'obtenir le nombre de caractères contenus dans notre chaîne.

```
PS C:\WINDOWS\system32> $maVariable.Length
23
PS C:\WINDOWS\system32> |
```

4. Navigation dans les répertoires et les fichiers.

4.1. Les nouvelles commandes

Nous avons vu que nous pouvions toujours utiliser les vieilles commandes DOS/CMD afin de nous déplacer dans une hiérarchie de dossiers. En effet, lorsque vous tapez la commande `dir` dans la console PowerShell, vous faites en réalité appel à un alias. L'alias vous fait exécuter la commande `Get-ChildItem`.

- Pour le vérifier, nous tapons la commande suivante :

```
PS C:\WINDOWS\system32> Get-Alias dir

CommandType      Name                               Version        Source
-----
Alias             dir -> Get-ChildItem
```

Voici un tableau récapitulatif des principales commandes CMD et de leurs équivalents en PowerShell :

DOS/CMD	Alias PS	Commandelette PS	Description
DIR	DIR	Get-ChildItem	Liste le contenu d'un répertoire
CD	CD	Set-Location	Change de répertoire courant
MD	MD	New-Item	Crée un fichier/ répertoire
RD	RD	Remove-Item	Supprime un fichier/répertoire
MOVE	MOVE	Move-Item	Déplace un fichier/répertoire
REN	REN	Rename-Item	Renomme un fichier/répertoire
COPY	COPY	Copy-Item	Copie un fichier/ répertoire

4.2. Get-ChildItem (Alias : gci, ls, dir)

Cette commandelette permet d'obtenir les fichiers et dossiers présents dans le système de fichiers.

- Par exemple, nous observons le résultat de la commande suivante :

```
PS C:\WINDOWS\system32> gci c:\

Répertoire : C:\

Mode                LastWriteTime         Length Name
----                -
d-----          07/05/2022    07:24             PerfLogs
d-r---          28/04/2025    15:53       Program Files
d-r---          03/03/2025    15:46 Program Files (x86)
d-r---          24/02/2025    17:59           Users
d-----          02/05/2025    14:45          Windows
```

- Nous observons la colonne Mode.

```
PS C:\WINDOWS\system32> gci c:\

Répertoire : C:\

Mode                LastWriteTime         Length Name
----                -
d-----          07/05/2022    07:24             PerfLogs
d-r---          28/04/2025    15:53       Program Files
d-r---          03/03/2025    15:46 Program Files (x86)
d-r---          24/02/2025    17:59           Users
d-----          02/05/2025    14:45          Windows
```

Celle-ci indique la nature des objets à l'intérieur du système de fichiers, voici les valeurs possibles : - d : pour un répertoire. - a : pour archive. - r : pour un objet en lecture seule. - h : pour un objet caché. - s : pour un objet système.

- Nous ajoutons à la commande Get-Childitem le paramètre -Force, afin d'afficher les fichiers cachés :

```
PS C:\WINDOWS\system32> gci c:\ -Force

Répertoire : C:\

Mode                LastWriteTime         Length Name
----                -
d- -hs-           24/02/2025    10:35          $Recycle.Bin
d- -h--           22/04/2025    09:39          $WINDOWS.~BT
d- -hs-           25/04/2025    13:55          Config.Msi
d- -hsl           08/01/2025    11:29      Documents and Settings
d- -         07/05/2022    07:24          PerfLogs
d- r---           28/04/2025    15:53      Program Files
d- -         03/03/2025    15:46      Program Files (x86)
d- -h-           04/04/2025    09:35      ProgramData
d- -hs-           27/02/2025    10:32          Recovery
d- -hs-           15/05/2025    10:30      System Volume Information
d- -         24/02/2025    17:59          Users
d- -         02/05/2025    14:45          Windows
-a- -hs-           12/05/2025    10:35      12288 DumpStack.log.tmp
-a- -hs-           15/05/2025    09:43      27451232256 hiberfil.sys
-a- -hs-           12/05/2025    10:35      10200547328 pagefile.sys
-a- -hs-           12/05/2025    10:35      16777216 swapfile.sys
```

Le nom des colonnes indique le nom d'une propriété de l'objet fichier ou répertoire (la commandelette gci renvoie une collection d'objets).

Quelques exemples :

- Affichage de tous les fichiers ayant l'extension .log contenus à l'intérieur d'une arborescence :

```
PS C:\WINDOWS\system32> gci C:\users\* -Include *.log -Recurse

Répertoire : C:\users\afrancomme\VirtualBox

Mode                LastWriteTime         Length Name
----                -
-a-----         07/02/2025         09:37           4539 selectorwindow.log
-a-----         07/02/2025         09:37          81419 VBoxSVC.log

Répertoire : C:\users\llopez\VirtualBox

Mode                LastWriteTime         Length Name
----                -
-a-----         25/03/2025         09:31          1859 HostInterfaceNetworking-VirtualBox Host-Only Ethernet
Adapter-Dhcpd.log
-a-----         12/05/2025         12:52           140 selectorwindow.log
-a-----         12/05/2025         12:52          88798 VBoxSVC.log
```

- Affichage du nom des fichiers dont la taille est supérieure à 32 Ko :

```
PS C:\WINDOWS\system32> Get-ChildItem | Where-Object {$_.Length -gt 32KB}

Répertoire : C:\WINDOWS\system32

Mode                LastWriteTime         Length Name
----                -
-a-----         24/02/2025         17:54          212411 @facial-recognition-windows-hello-rejuv.gif
-a-----         07/05/2022         07:20          243815 @facial-recognition-windows-hello.gif
-a-----         20/03/2025         08:57          446464 aadauthhelper.dll
-a-----         20/03/2025         08:57          950272 aadcloudap.dll
-a-----         27/02/2025         10:35           94208 aadjcsp.dll
-a-----         27/02/2025         10:34          1482240 aadtb.dll
-a-----         27/02/2025         10:34          226760 aadWamExtension.dll
-a-----         27/02/2025         10:34          716800 AarSvc.dll
-a-----         20/03/2025         08:57          824736 AboutSettingsHandlers.dll
-a-----         24/02/2025         17:53          430080 AboveLockAppHost.dll
-a-----         27/02/2025         10:35          294912 accessibilitycpl.dll
-a-----         24/02/2025         17:54          303104 accountaccessor.dll
-a-----         24/02/2025         17:54          491520 AccountsRt.dll
```

- Affichage des fichiers dont la date de dernier enregistrement est postérieure au 01/04/2025:

```
PS C:\WINDOWS\system32> Get-ChildItem | Where-Object {$_.LastWriteTime -gt '04/01/2025'}
```

Répertoire : C:\WINDOWS\system32

Mode	LastWriteTime	Length	Name
d----	02/05/2025 16:49		catroot2
d----	02/05/2025 15:03		config
d----	28/04/2025 15:54		drivers
d----	28/04/2025 15:54		DriverStore
d----	25/04/2025 16:16		LogFiles
d----	28/04/2025 15:54		Npcap
d----	22/04/2025 14:01		SecurityHealth
d----	15/05/2025 09:44		SleepStudy
d----	15/05/2025 10:45		sru
d----	14/05/2025 12:30		Tasks
d----	02/05/2025 14:54		WebThreatDefSvc
-a----	12/05/2025 10:35	845256	AsusUpdateCheck.exe
-a----	15/05/2025 10:49	42022	Get-ChildItem
-a----	12/05/2025 10:42	129254	perfc009.dat
-a----	12/05/2025 10:42	149946	perfc00C.dat
-a----	12/05/2025 10:42	675316	perfh009.dat
-a----	12/05/2025 10:42	774292	perfh00C.dat
-a----	12/05/2025 10:42	1721728	PerfStringBackup.INI
-a----	12/05/2025 10:35	901328	wpbbin.exe

Attention : la date est toujours au format américain, soit MM/JJ/AAAA. Le pipe « | » permet de passer un ou plusieurs objets à la commande qui suit. Dans nos exemples, nous passons chaque objet à la commandelette Where-Object (appelée aussi clause, ou encore filtre). Cette dernière va analyser les propriétés Length ou LastWriteTime et retourner les objets correspondants. Le « \$_ » indique qu'on traite l'objet courant.

- Affichage de tous les fichiers ou répertoires cachés à la racine de la partition système :

```
PS C:\WINDOWS\system32> Get-ChildItem c:\ -Attributes Hidden
```

Répertoire : C:\

Mode	LastWriteTime	Length	Name
d--hs-	24/02/2025 10:35		\$Recycle.Bin
d--h--	22/04/2025 09:39		\$WINDOWS.~BT
d--hs-	25/04/2025 13:55		Config.Msi
d--hsl	08/01/2025 11:29		Documents and Settings
d--h--	04/04/2025 09:35		ProgramData
d--hs-	27/02/2025 10:32		Recovery
d--hs-	15/05/2025 10:30		System Volume Information
-a-hs-	12/05/2025 10:35	12288	DumpStack.log.tmp
-a-hs-	15/05/2025 09:43	27451232256	hiberfil.sys
-a-hs-	12/05/2025 10:35	10200547328	pagefile.sys
-a-hs-	12/05/2025 10:35	16777216	swapfile.sys

On peut associer des combinaisons d'attributs via les opérateurs suivants : - + : pour signifier un ET logique. - , : pour signifier un OU logique. - ! : pour signifier une négation.

- Par exemple, nous filtrons uniquement les fichiers cachés et non les répertoires en combinant les attributs « caché » et « n'est pas un répertoire ».

```
PS C:\WINDOWS\system32> Get-ChildItem c:\ -Attributes Hidden+!Directory
```

Répertoire : C:\

Mode	LastWriteTime		Length	Name
----	-----		-----	----
-a-hs-	12/05/2025	10:35	12288	DumpStack.log.tmp
-a-hs-	15/05/2025	09:43	27451232256	hiberfil.sys
-a-hs-	12/05/2025	10:35	10200547328	pagefile.sys
-a-hs-	12/05/2025	10:35	16777216	swapfile.sys

4.3. Set-Location (Alias : sl, cd, chdir)

Elle nous permet de nous déplacer dans une arborescence de dossiers.

- Saisie, par exemple :

```
PS C:\WINDOWS\system32> Set-Location D:\
PS D:\>
```

Comme dans CMD, on peut utiliser des chemins relatifs ainsi que les raccourcis « .. » et « \ », pour désigner respectivement le répertoire parent et le répertoire racine du disque en cours

4.4. Get-Location (Alias : gl, pwd)

Cette commande retourne l'emplacement actuel à l'intérieur d'une arborescence.

- Saisie de la commande Get-Location :

```
PS C:\WINDOWS\system32> get-location

Path
----
C:\WINDOWS\system32
```

- Affichage des propriétés et méthodes :

```
PS C:\WINDOWS\system32> Get-Location | Get-Member

    TypeName : System.Management.Automation.PathInfo

Name      MemberType Definition
-----
Equals    Method      bool Equals(System.Object obj)
GetHashCode Method      int GetHashCode()
GetType   Method      type GetType()
ToString  Method      string ToString()
Drive     Property    System.Management.Automation.PSDriveInfo Drive {get;}
Path      Property    string Path {get;}
Provider  Property    System.Management.Automation.ProviderInfo Provider {get;}
ProviderPath Property    string ProviderPath {get;}
```

- Nous récupérons dans une variable le chemin (path) de l'emplacement courant en une seule ligne de commande.

```
PS C:\WINDOWS\system32> $CheminCourant = (Get-Location).Path
PS C:\WINDOWS\system32> $CheminCourant
```

- Nous venons de stocker la valeur du chemin courant, en l'occurrence C:\users, dans la variable \$CheminCourant.

Nous l'affichons dans la console :

```
PS C:\WINDOWS\system32> $CheminCourant
C:\WINDOWS\system32
```

4.5. New-Item (Alias : ni, md)

Cette commande permet de créer des répertoires, à l'instar de la commande md de CMD, mais aussi des fichiers.

Examinons de plus près quelques-uns de ses paramètres :

Paramètre	Description
Path	Chemin d'accès de l'élément à créer (ex : C:\Temp).
Itemtype	Type d'élément à créer : file pour un fichier, directory pour un dossier.
Name	Nom du nouvel élément à créer.
Value	Contenu de l'élément à créer (ex : "bonjour !" dans le cas d'un fichier texte).

4.5.1. Création d'un répertoire

- Création du répertoire Temp :

```
PS C:\WINDOWS\system32> New-Item -ItemType directory -Name Temp
```

Répertoire : C:\WINDOWS\system32

Mode	LastWriteTime	Length	Name
d----	15/05/2025 10:56		Temp

- Si le dossier avait contenu un espace, nous aurions dû le mettre entre guillemets. Par exemple :

```
PS C:\WINDOWS\system32> New-Item -Name 'Dossier Test' -ItemType directory
```

Répertoire : C:\WINDOWS\system32

Mode	LastWriteTime	Length	Name
d----	15/05/2025 10:57		Dossier Test

4.5.2. Création d'un fichier

- Création d'un fichier nommé monFichier.txt qui contient la phrase « Madame Pasquier for ever ! » :

```
PS C:\WINDOWS\system32> ni -Name monFichier.txt -ItemType file -Value 'Madame Pasquier for ever!'
```

Répertoire : C:\WINDOWS\system32

Mode	LastWriteTime	Length	Name
-a----	15/05/2025 10:59	25	monFichier.txt

Sachez que les opérateurs de redirection « > » et « >> » fonctionnent aussi très bien avec PowerShell.

4.6. Remove-Item (Alias : ri, rm, rmdir, rd, erase, del)

Cette commandelette, comme son nom l'indique, permet de supprimer des fichiers ou des dossiers.

- Suppression de tous les fichiers .log contenus dans le répertoire C:\Temp :

```
PS C:\WINDOWS\system32> Remove-Item C:\Temp\*.log
PS C:\WINDOWS\system32>
```

- Suppression sélective de tous les fichiers contenus dans une arborescence de dossiers dont l'extension est .txt :

```
PS C:\WINDOWS\system32> Get-ChildItem C:\Temp\* -Include *.txt -Recurse | Remove-Item
PS C:\WINDOWS\system32>
```

On liste d'abord les objets (fichiers à supprimer), puis on les passe via le pipe à la commande Remove-item. Pour supprimer un fichier système, masqué ou en lecture seule, il suffit tout simplement d'utiliser le paramètre -Force, comme dans l'exemple ci-dessous :

```
PS C:\WINDOWS\system32> ri monFichier.txt -Force
PS C:\WINDOWS\system32> _
```

4.7. Move-Item (Alias : mi, move, mv)

Cette commande permet de déplacer un fichier ou un répertoire d'un emplacement vers un autre emplacement. Dans le cas d'un répertoire, le contenu est également déplacé. Move-Item permet en outre d'effectuer le renommage de l'objet manipulé.

4.7.1. Déplacement de fichiers

Exemple : déplacement des fichiers *.jpg du répertoire courant vers le dossier « mes photos ».

```
PS C:\WINDOWS\system32> Move-Item -Path *.jpg -destination 'Mes photos'
PS C:\WINDOWS\system32>
```

Ou

```
PS C:\WINDOWS\system32> Move-Item *.jpg 'Mes photos'
PS C:\WINDOWS\system32>
```

Pour que l'exemple ci-dessus fonctionne, il faudrait qu'au préalable nous ayons créé le dossier mes photos. Ceci étant, il est possible de forcer la création du dossier de destination en utilisant le paramètre -force.

4.7.2. Déplacement d'un répertoire

Le déplacement d'un répertoire est similaire au déplacement de fichiers. Prenons l'exemple suivant :

```
PS C:\WINDOWS\system32> New-Item -ItemType directory -Name Rep1

Répertoire : C:\WINDOWS\system32

Mode                LastWriteTime         Length Name
----                -
d-----          15/05/2025   11:09                Rep1

PS C:\WINDOWS\system32> Move-Item 'Rep1' 'Rep2'
PS C:\WINDOWS\system32>
```

Nous venons de déplacer l'intégralité du répertoire Rep1 dans le répertoire Rep2. Comment ferons nous maintenant pour renommer le dossier Rep1 en Rep3 ? Réponse :

```
PS C:\WINDOWS\system32\Rep2> New-Item -ItemType directory -Name Rep1

Répertoire : C:\WINDOWS\system32\Rep2

Mode                LastWriteTime         Length Name
----                -
d-----          15/05/2025   11:11                Rep1

PS C:\WINDOWS\system32\Rep2> Move-Item 'Rep1' 'Rep3'
PS C:\WINDOWS\system32\Rep2> ls

Répertoire : C:\WINDOWS\system32\Rep2

Mode                LastWriteTime         Length Name
----                -
d-----          15/05/2025   11:11                Rep3

PS C:\WINDOWS\system32\Rep2>
```


4.8. Rename-Item (Alias : ren, rni)

L'objectif de cette commande est de renommer un fichier ou un dossier. Celle-ci n'est que moyennement utile dans la mesure où elle fait double emploi avec la commande Move-Item. Elle peut éviter de confondre renommage et déplacement comme dans l'exemple précédent.

4.8.1. Renommer un fichier

Exemple : renommage du fichier monFichierDeLog.txt en ficlog.txt.

```
PS C:\WINDOWS\system32> Rename-Item -Path C:\WINDOWS\system32\monFichierDeLog.txt -Newname ficlog.txt
PS C:\WINDOWS\system32> _
```

ou

```
PS C:\WINDOWS\system32> Rename-Item -Path C:\WINDOWS\system32\ficlog.txt ficlog1.txt
PS C:\WINDOWS\system32> _
```

4.8.2. Renommer un dossier

Exemple : renommage du répertoire Dossier1 en Dossier2.

```
PS C:\WINDOWS\system32> Rename-Item -Path C:\WINDOWS\system32\Dossier1 -Newname Dossier2
PS C:\WINDOWS\system32> _
```

Ou

```
PS C:\WINDOWS\system32> Rename-Item -Path C:\WINDOWS\system32\Dossier2 Dossier1
PS C:\WINDOWS\system32>
```

4.9. Copy-Item (Alias : cpi, cp, copy)

Grâce à cette commande, nous pouvons copier des fichiers ou des répertoires, voire les deux à la fois

.

Quelques exemples :

- Copie un fichier d'un répertoire source vers un répertoire destination.

```
PS C:\WINDOWS\system32> Copy-Item -Path C:\windows\system32\ficLog1.txt -destination D:\Logs
PS C:\WINDOWS\system32> _
```

Ou

```
PS D:\LogsTest> Copy-Item d:\LogsTest\ficLog2.txt c:\windows\system32
PS D:\LogsTest>
```

- Copie d'une arborescence de répertoires (c'est-à-dire avec tous les sous-dossiers et fichiers).

```
PS D:\LogsTest> New-Item -ItemType directory -Name RepSource
```

Répertoire : D:\LogsTest

Mode	LastWriteTime	Length	Name
d----	15/05/2025 11:32		RepSource

```
PS D:\LogsTest> New-Item -ItemType directory -Name RepDest
```

Répertoire : D:\LogsTest

Mode	LastWriteTime	Length	Name
d----	15/05/2025 11:32		RepDest

```
PS D:\LogsTest> Copy-Item -Path RepSource -Destination RepDest -Recurse
```

```
PS D:\LogsTest>
```

Copy-Item crée automatiquement le répertoire de destination s'il n'existe pas.

5. La boucle Foreach.

Une boucle est une structure répétitive qui permet d'exécuter plusieurs fois les instructions qui se trouvent à l'intérieur du bloc d'instruction. Ce type de traitement est de loin le plus pratique pour parcourir une collection de données. Contrairement à la boucle For, il n'est pas nécessaire de déterminer à l'avance le nombre d'éléments contenus dans la collection. Bien qu'utilisable en ligne de commandes, c'est-à-dire directement dans la console, on trouve plus fréquemment cette forme dans les scripts. Sa syntaxe est la suivante : Foreach (<élément> in) { #bloc d'instructions }

6. Module Active Directory.

Ce module a fait son apparition dans Windows Server 2008 R2 et a été grandement enrichi avec l'arrivée de Windows Server 2012. Il permet d'administrer en ligne de commandes PowerShell le rôle « Active Directory Domain Services (AD DS) ». Il est installé en même temps que le rôle Active Directory.

6.1. Mise en route du module

Contrairement à la version 2.0 de PowerShell, par défaut, il n'est plus nécessaire d'importer le module Active Directory afin de pouvoir s'en servir. La seule chose que vous verrez lors de la première utilisation de l'une des commandes dans votre session PowerShell est l'apparition d'une barre de progression durant le chargement du module Active Directory.

Dans le cas où vous avez désactivé la fonctionnalité d'importation automatique des modules, il vous faudra alors l'importer manuellement de la façon suivante :

Contrairement à la version 2.0 de PowerShell, par défaut, il n'est plus nécessaire d'importer le module Active Directory afin de pouvoir s'en servir. La seule chose que vous verrez lors de la première utilisation de l'une des commandes dans votre session PowerShell est l'apparition d'une barre de progression durant le chargement du module Active Directory. Dans le cas où vous avez désactivé la fonctionnalité d'importation automatique des modules, il vous faudra alors l'importer manuellement de la façon suivante : PS > Import-Module ActiveDirectory Vous pouvez obtenir l'ensemble des commandes apportées par le module Active Directory en tapant ceci : PS > Get-Command -Module ActiveDirectory

6.2. Gestion des utilisateurs

Pour la gestion des utilisateurs, nous disposons d'un jeu de quelques commandes que nous pouvons obtenir de la manière suivante :

```
PS C:\Users\Administrateur> get-command -module ActiveDirectory -name *user*

CommandType      Name                                     Version      Source
-----
Cmdlet           Get-ADUser                             1.0.1.0      ActiveDirectory
Cmdlet           Get-ADUserResultantPasswordPolicy      1.0.1.0      ActiveDirectory
Cmdlet           New-ADUser                             1.0.1.0      ActiveDirectory
Cmdlet           Remove-ADUser                          1.0.1.0      ActiveDirectory
Cmdlet           Set-ADUser                             1.0.1.0      ActiveDirectory

PS C:\Users\Administrateur>
```

Commande	Description
Get-ADUser	Obtient un ou plusieurs utilisateurs Active Directory.
Set-ADUser	Modifie un utilisateur Active Directory.
New-ADUser	Crée un utilisateur Active Directory.
Remove-ADUser	Supprime un utilisateur Active Directory.
Get-ADUserResultantPasswordPolicy	Obtient la stratégie de mot de passe résultante pour un utilisateur Active Directory.

6.2.1. Créer des utilisateurs

La commande New-ADUser possède de très nombreux paramètres. Les plus couramment utilisés sont les suivantes :

Paramètre	Description
SamAccountName <String>	Spécifie le nom du compte de sécurité SAM (nom de groupe antérieur à Windows 2000).
Name <String>	Spécifie le nom de l'objet.
Surname <String>	Spécifie le nom de famille de l'utilisateur.
DisplayName <String>	Spécifie le nom d'affichage de l'objet.
GivenName <String>	Spécifie le prénom de l'utilisateur.
Description <String>	Spécifie une description de l'objet.
EmailAddress <String>	Spécifie l'adresse de messagerie de l'utilisateur.
Enabled { \$true \$false }	Spécifie si le compte doit être activé. Un compte activé requiert un mot de passe.
AccountPassword <SecureString>	Spécifie une nouvelle valeur de mot de passe pour un compte. Cette valeur est stockée sous forme d'une chaîne sécurisée.
CannotChangePassword { \$true \$false }	Spécifie si le mot de passe du compte peut être modifié.
ChangePasswordAtLogon { \$true \$false }	Spécifie si un mot de passe doit être modifié lors de la prochaine tentative d'ouverture de session.
PasswordNeverExpires { \$true \$false }	Spécifie si le mot de passe du compte peut expirer.
PasswordNotRequired { \$true \$false }	Spécifie si le compte requiert un mot de passe. Un mot de passe n'est pas nécessaire pour un nouveau compte.
HomeDirectory <String>	Spécifie le répertoire de base d'un utilisateur.
HomeDrive <String>	Spécifie un lecteur associé au chemin UNC défini par la propriété HomeDirectory.
ProfilePath <String>	Spécifie un chemin d'accès au profil de l'utilisateur.
Path <String>	Spécifie le chemin d'accès de l'unité d'organisation ou du conteneur où le nouvel objet est créé. Si cette valeur n'est pas précisée, avec AD DS les objets utilisateurs seront rangés dans l'unité d'organisation Users.
ScriptPath <String>	Spécifie un chemin d'accès au script d'ouverture de session de l'utilisateur.
Instance <ADUser>	Spécifie une instance d'un objet utilisateur à utiliser comme modèle pour un nouvel objet utilisateur.

Paramètre	Description
Credential <PSCredential>	Spécifie les informations d'identification de compte d'utilisateur à utiliser pour effectuer cette tâche.

Pour créer un compte utilisateur, il faut au minimum préciser la propriété Name. PS >
New-ADUser -Name Winnie Cette ligne de commandes crée un utilisateur dans le conteneur Users (car nous n'avons rien précisé) qui apparaîtra sous le nom de « Winnie ». Il aura également par défaut un attribut SamAccountName qui prendra la valeur « Winnie ». Si maintenant nous voulons créer un compte avec davantage d'attributs et que nous voulons le placer dans l'unité d'organisation « BTSSIO1 », nous pouvons le faire ainsi : PS >
New-ADUser -SamAccountName johnny -Name 'Jean-Philippe Smet' -GivenName Jean-Philippe -Surname Smet -DisplayName 'Jean-Philippe Smet' -description 'Etudiant sio' -HomeDrive 'H:' -HomeDirectory '\\AD\Home' -Path 'OU=BTSSIO1,OU=Elevés,DC=sio-exupery,DC=local'

6.2.2. Affecter un mot de passe à la création

Nous pouvons affecter un mot de passe à la création d'un compte, avec la commande New-ADUser. Comme la valeur attendue pour le paramètre -AccountPassword est de type chaîne sécurisée (SecureString), nous devons effectuer quelques petites manipulations supplémentaires : PS > \$passwd = '123AZEqsdl!' PS > \$passwd = ConvertTo-SecureString \$passwd -AsPlainText -Force PS > New-ADUser -SamAccountName Johnny -Name 'Jean-Philippe Smet' -AccountPassword \$passwd Au lieu de stocker le mot de passe en clair, ce qui n'est pas idéal du point de vue de la sécurité, nous aurions pu utiliser la commande Read-Host associée au paramètre -AsSecureString : PS > \$secure_passwd = read-host "Entrer un mot de passe :" -assecurestring

6.2.3. Activer un compte à la création

Pour activer un compte à la création, il est nécessaire de lui affecter un mot de passe et d'utiliser le paramètre Enabled. PS > \$passwd = 'Passw0rd123*!' PS > \$passwd = ConvertTo-SecureString \$passwd -AsPlainText -Force PS > New-ADUser -SamAccountName Johnny -Name 'Jean-Philippe Smet' -GivenName Jean-Philippe -Surname Smet -DisplayName 'Jean-Philippe Smet' -description 'Etudiant sio' -HomeDrive 'H:' -HomeDirectory '\\AD\Home' -Path 'OU=BTSSIO1,OU=Elevés,DC=sio-exupery,DC=local' -AccountPassword \$passwd -Enabled \$true

6.3. Gestion des groupes

Commande	Description
Add-ADGroupMember	Ajoute un ou plusieurs membres à un groupe Active Directory.
Add-ADPrincipalGroupMembership	Ajoute un membre à un ou plusieurs groupes Active Directory.
Get-ADAccountAuthorizationGroup	Obtient les groupes de sécurité par l'utilisateur, l'ordinateur ou le jeton de comptes de service spécifié.
Get-ADGroup	Obtient un ou plusieurs groupes Active Directory.
Get-ADGroupMember	Obtient les membres d'un groupe Active Directory.
Get-ADPrincipalGroupMembership	Obtient les groupes Active Directory qui possèdent un utilisateur, un ordinateur, un groupe ou un compte de service spécifié.
New-ADGroup	Crée un groupe Active Directory.
Remove-ADGroup	Supprime un groupe Active Directory.
Remove-ADGroupMember	Supprime un ou plusieurs membres d'un groupe Active Directory.
Remove-ADPrincipalGroupMembership	Supprime un membre d'un ou plusieurs groupes Active Directory.
Set-ADGroup	Modifie un groupe Active Directory.

6.3.1. Créer des groupes

La création de groupes s'effectue à l'aide de la commande New-ADGroup. Celle-ci comprend les paramètres suivants :

Paramètre	Description
Name <String>	Spécifie le nom de l'objet.
Description <String>	Spécifie une description de l'objet.
DisplayName <String>	Spécifie le nom d'affichage de l'objet.
GroupCategory {Distribution Security}	Spécifie la catégorie du groupe.
GroupScope {DomainLocal Global Universal}	Spécifie l'étendue de groupe du groupe.
SamAccountName <String>	Spécifie le nom du compte de sécurité SAM (nom de groupe antérieur à Windows 2000).
Instance <ADGroup>	Spécifie une instance d'un objet groupe à utiliser comme modèle pour un nouvel objet groupe.
ManagedBy <ADPrincipal>	Spécifie l'utilisateur ou le groupe qui gère l'objet.
Path <String>	Spécifie le chemin d'accès de l'unité d'organisation ou du conteneur où le nouvel objet est créé.
Credential <PSCredential>	Spécifie les informations d'identification de compte d'utilisateur à utiliser pour effectuer cette tâche.

Pour créer un groupe, il faut au minimum lui donner un nom (propriété Name) et une étendue (GroupScope). Si aucune catégorie (GroupCategory) n'est spécifiée alors par défaut il sera créé en tant que groupe de sécurité. Si vous ne spécifiez pas de SamAccountName alors cette propriété prendra le même nom que la propriété Name du groupe.

Exemple : création d'un groupe global de sécurité :

```
PS > New-ADGroup -Name BTSSIO1 -GroupScope Global -GroupCategory Security
```


6.3.2. Ajouter des membres à un groupe

L'ajout de membres à un groupe s'effectue avec la commande `Add-ADGroupMember`. Les paramètres disponibles sont les suivants :

Paramètre	Description
<code>Identity <ADGroup></code>	Spécifie un objet groupe en fournissant l'une de ses valeurs de propriété permettant de l'identifier.
<code>Members <ADPrincipal[]></code>	Spécifie un jeu d'objets (utilisateur, groupe et ordinateur), dans une liste séparée par des virgules, à ajouter à un groupe.

Exemple : ajout de plusieurs utilisateurs à un groupe.

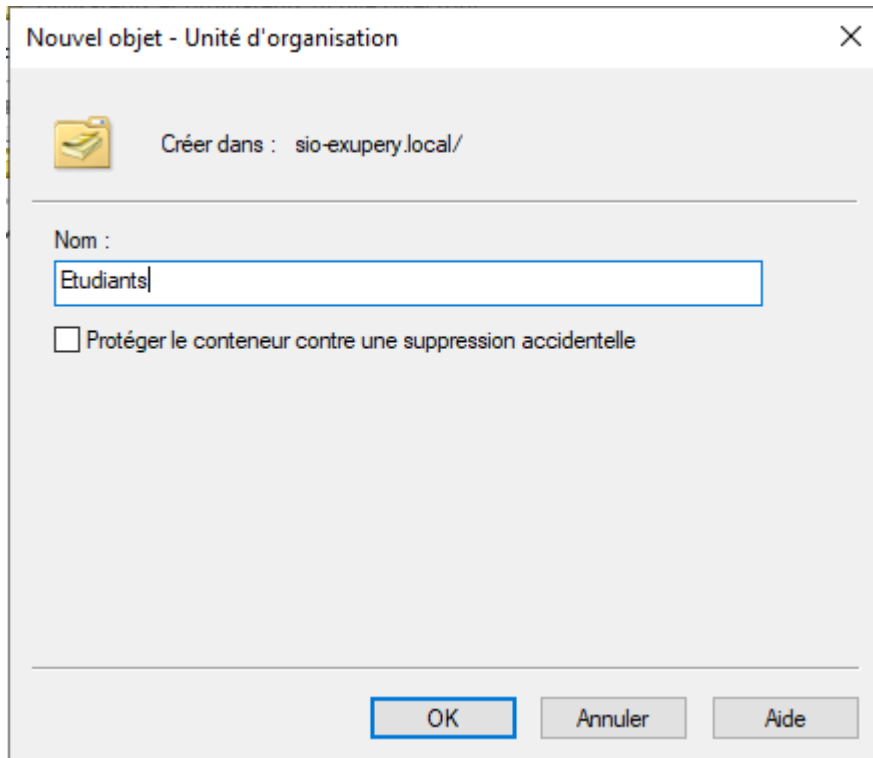
```
PS > Add-ADGroupMember -Identity BTSSIO1 -Members Eleve1,Eleve2
```

7. TP : Création des comptes utilisateurs par lots.

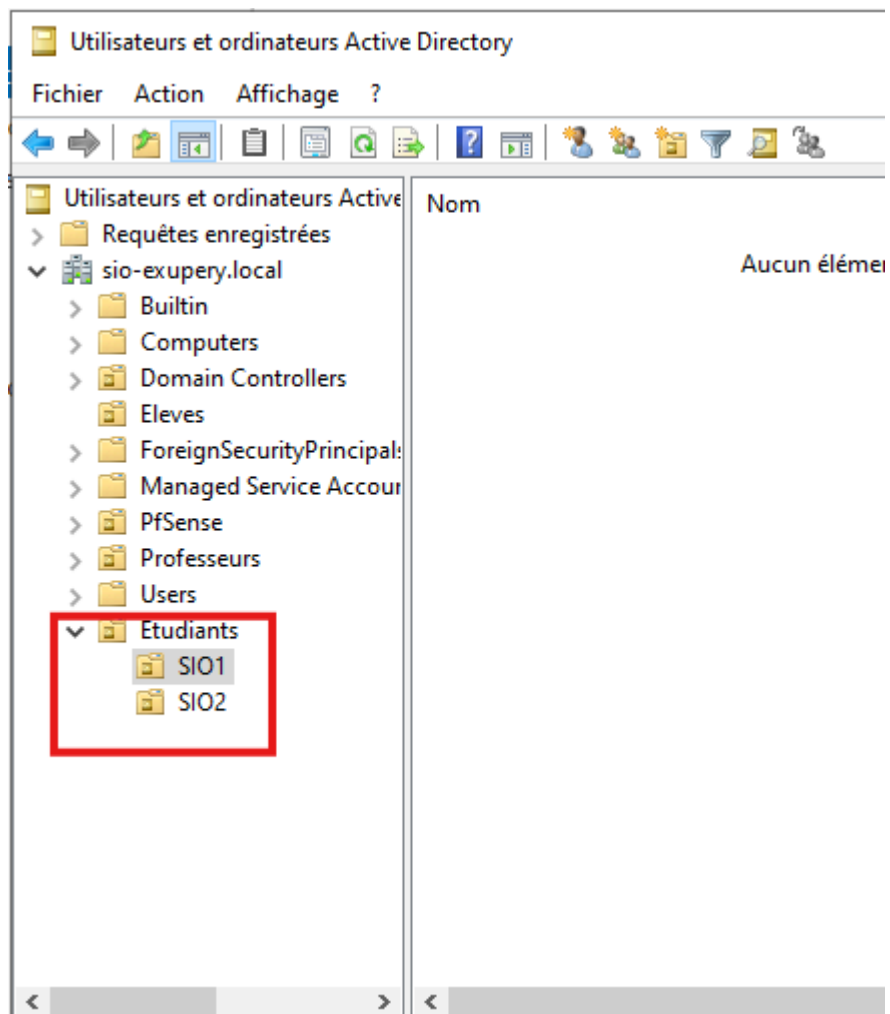
Comme chaque année, vous avez plusieurs dizaines de comptes utilisateurs à créer pour les étudiants. Il s'agit d'automatiser cette tâche à l'aide d'un script que vous nommerez Add-Users.ps1 et d'un fichier comptes.csv. Les comptes étudiants doivent être affectés aux groupes correspondant à leurs classes.

Solution :

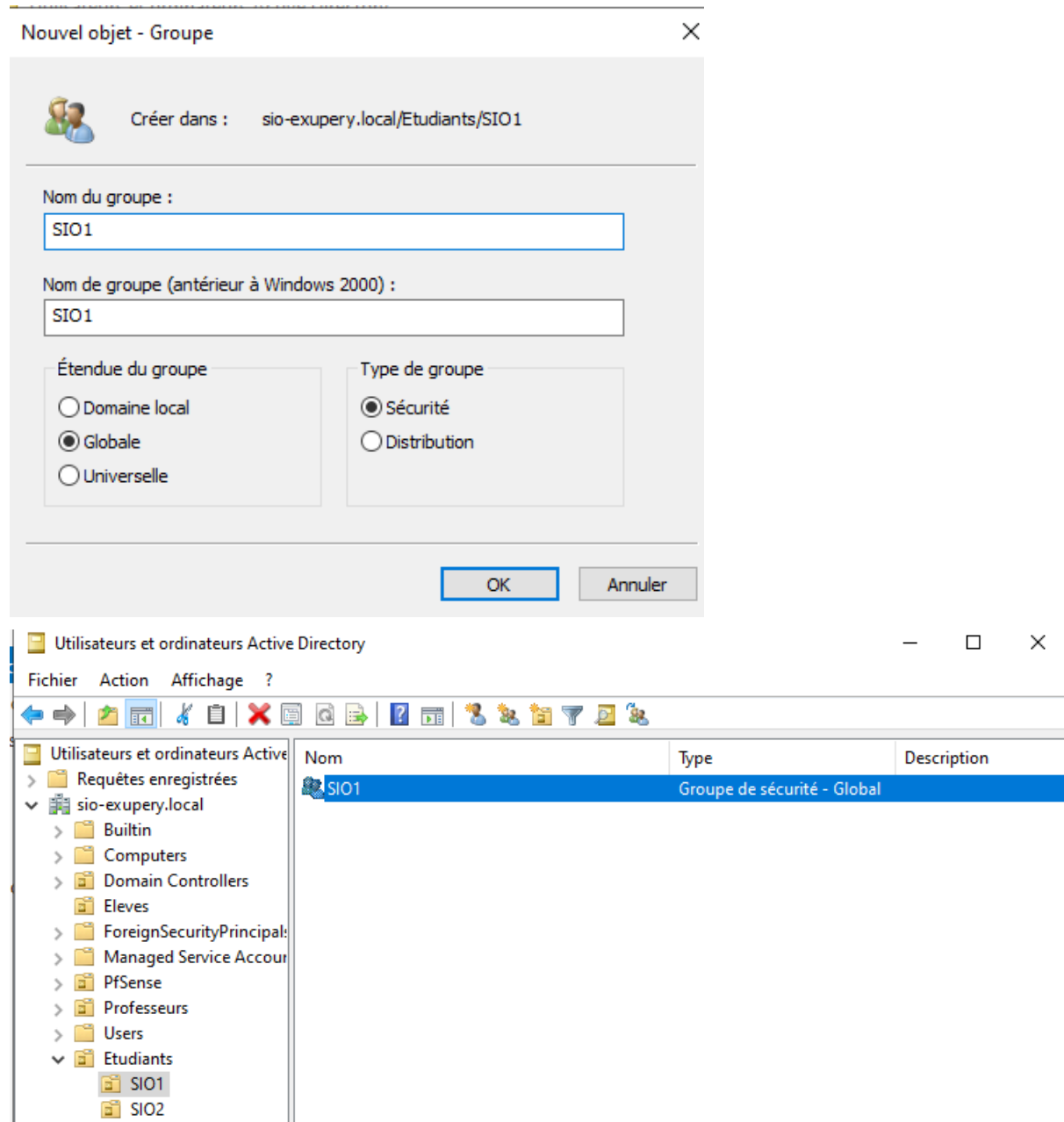
- Création de l'UO Etudiants puis, à l'intérieur de celle-ci, les UO BTSSIO1 et BTSSIO2.



Nous avons créé les UO SIO1 et SIO2



- Création des groupes de sécurité BTSSIO1 et BTSSIO2 respectivement dans les UO BTSSIO1 et BTSSIO2.



Nouvel objet - Groupe

Créer dans : sio-exupery.local/Etudiants/SIO2

Nom du groupe :
SIO2

Nom de groupe (antérieur à Windows 2000) :
SIO2

Étendue du groupe

☐ Domaine local

☒ Globale

☐ Universelle

Type de groupe

☒ Sécurité

☐ Distribution

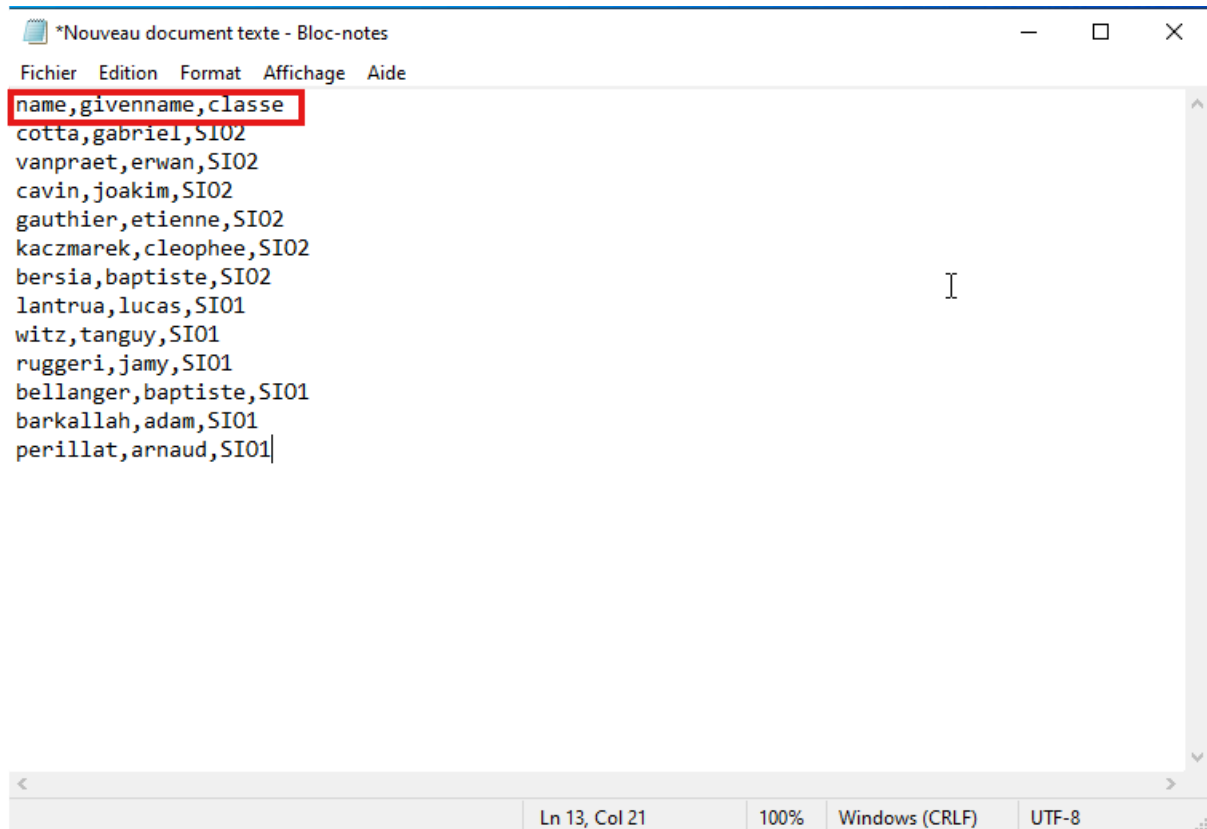
OK Annuler

Utilisateurs et ordinateurs Active Directory

Fichier Action Affichage ?

	Nom	Type	Description
Utilisateurs et ordinateurs Active Directory			
Requêtes enregistrées			
sio-exupery.local			
Builtin			
Computers			
Domain Controllers			
Eleves			
ForeignSecurityPrincipals			
Managed Service Accounts			
PfSense			
Professeurs			
Users			
Etudiants			
SIO1	SIO2	Groupe de sécurité - Global	
SIO2			

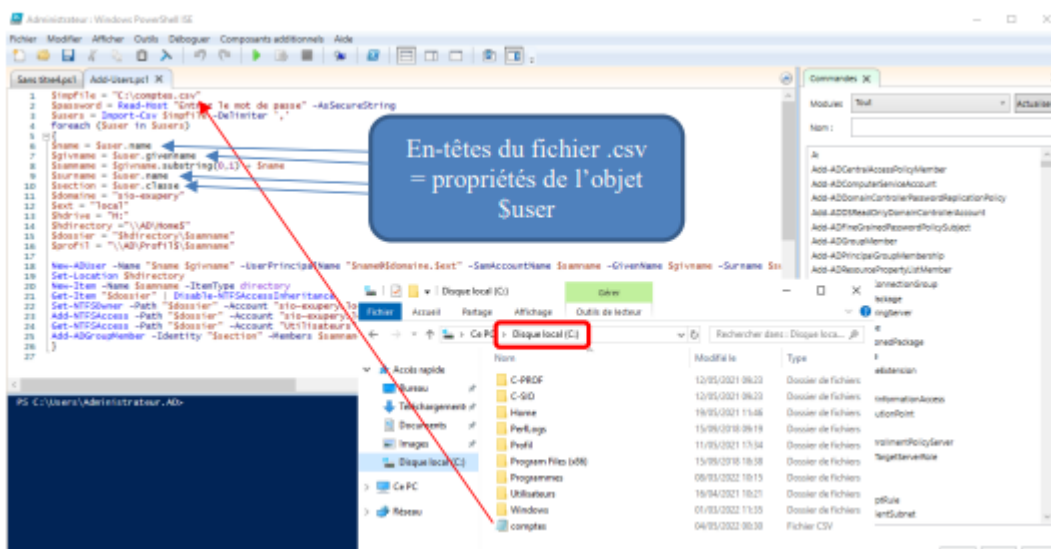
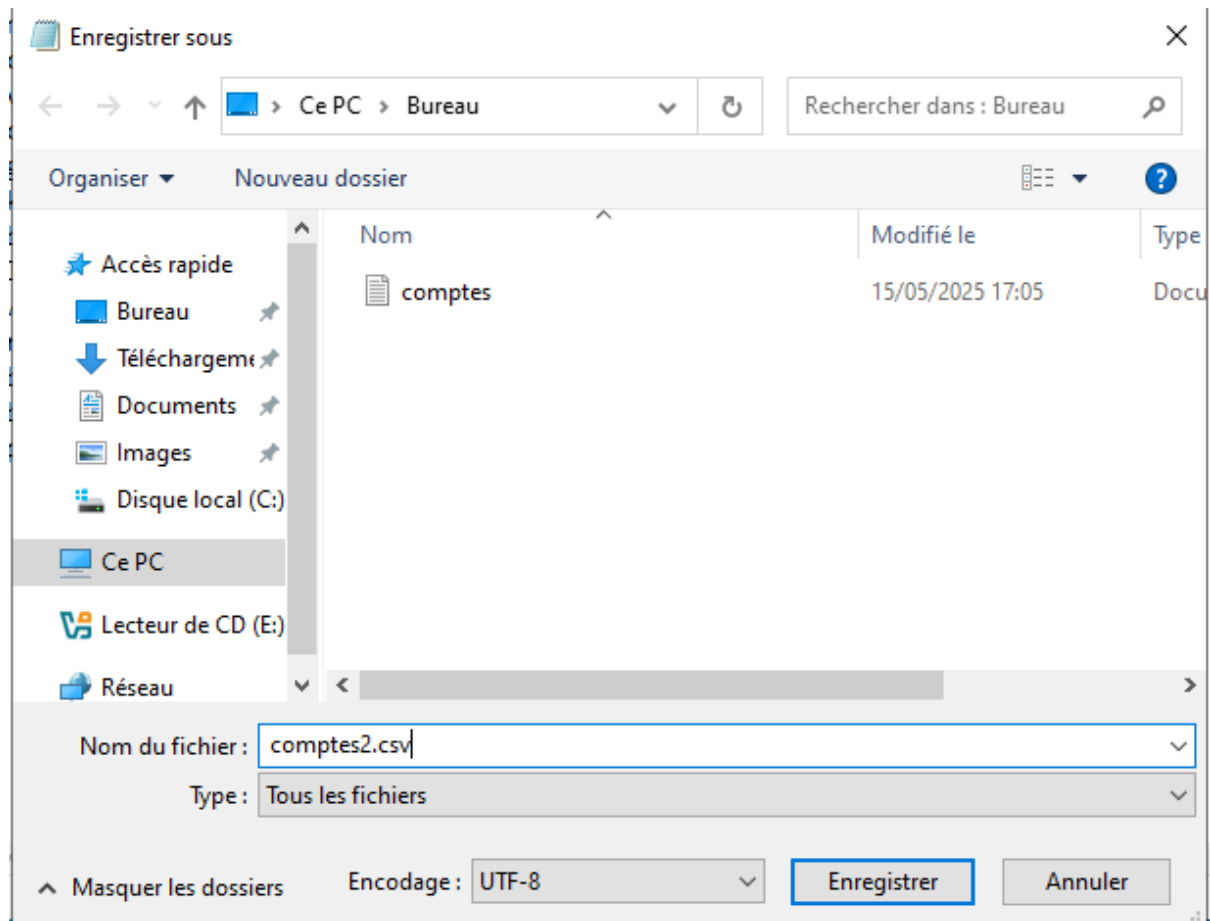
Création, sur votre serveur Windows, d'un fichier .csv où chaque ligne correspondra à la description d'un utilisateur



The screenshot shows a Windows Notepad application window titled "*Nouveau document texte - Bloc-notes". The menu bar includes "Fichier", "Edition", "Format", "Affichage", and "Aide". The text content is a CSV file with the following lines:

```
name,givenname,classe
cotta,gabriel,SI02
vanpraet,erwan,SI02
cavin,joakim,SI02
gauthier,etienne,SI02
kaczmarek,cleophee,SI02
bersia,baptiste,SI02
lantrua,lucas,SI01
witz,tanguy,SI01
ruggeri,jamy,SI01
bellanger,baptiste,SI01
barkallah,adam,SI01
perillat,arnaud,SI01
```

The first line, "name,givenname,classe", is highlighted with a red rectangular box. The status bar at the bottom indicates "Ln 13, Col 21", "100%", "Windows (CRLF)", and "UTF-8".



Dans le script Powershell (cf. ci-après), vous prévoyez l'importation de ce fichier dans PowerShell grâce à la commande Import-CSV, puis vous passerez la collection d'objets résultants à la commande New-ADUser. Il restera ensuite à ajouter les utilisateurs aux groupes BTSSIO1 et BTSSIO2 avec la commande Add-ADGroupMember.

- Nous avons installé les commandes/modules NTFS à l'aide de la commande Install-Module -Name NTFSSecurity.
- Les paramètres (attributs de l'objet utilisateur) figurant ci-dessous seront utilisés par la commande New-ADUser :
 - Name : nom de l'objet (il s'agit du nom visible dans la console de gestion de l'Active Directory).
 - UserPrincipalName (UPN) : identifiant de l'utilisateur suivi du nom du domaine. La partie identifiant du SAMAccountName et de l'UPN peut être différente. L'attribut UserPrincipalName peut être identique à l'adresse e-mail de l'utilisateur.
 - SAMAccountName : nom de login.
 - Surname : nom de l'utilisateur.
 - GivenName : prénom de l'utilisateur.
 - Path : unité d'organisation de l'utilisateur.
 - ProfilePath : chemin à spécifier en cas de profils itinérants.
 - HomeDrive : lettre de connexion au home directory.
 - HomeDirectory : chemin réseau (au format UNC) vers un partage sur un serveur.
- Affichage des commandes lets contenant NTFS :

```
PS C:\Users\Administrateur> get-command *ntfs*
```

CommandType	Name	Version	Source
-----	----	-----	-----
Cmdlet	Add-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Add-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Clear-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Clear-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Disable-NTFSAccessInheritance	4.2.6	NTFSSecurity
Cmdlet	Disable-NTFSAuditInheritance	4.2.6	NTFSSecurity
Cmdlet	Enable-NTFSAccessInheritance	4.2.6	NTFSSecurity
Cmdlet	Enable-NTFSAuditInheritance	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSEffectiveAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSHardLink	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSInheritance	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOrphanedAccess	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOrphanedAudit	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSOwner	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSSecurityDescriptor	4.2.6	NTFSSecurity
Cmdlet	Get-NTFSSimpleAccess	4.2.6	NTFSSecurity
Cmdlet	New-NTFSHardLink	4.2.6	NTFSSecurity
Cmdlet	New-NTFSSymbolicLink	4.2.6	NTFSSecurity
Cmdlet	Remove-NTFSAccess	4.2.6	NTFSSecurity
Cmdlet	Remove-NTFSAudit	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSInheritance	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSOwner	4.2.6	NTFSSecurity
Cmdlet	Set-NTFSSecurityDescriptor	4.2.6	NTFSSecurity
Cmdlet	Show-NTFSSimpleAccess	4.2.6	NTFSSecurity
Application	chkntfs.exe	10.0.20...	C:\Windows\system32\chkntfs.exe

▪ Ecriture du script dans PowerShell ISE :

```

1 $impfile = "C:\comptes2.csv"
2 $password = ReadHost "Entrez le mot de passe" -asSecureString
3 $users = Import-Csv $impfile -delimiter ';'
4 foreach ($user in $users)
5 {
6     $name = $user.name
7     $givenname = $user.givenname
8     $samname = $givenname.substring(0,1) + $name
9     $surname = $user.name
10    $section = $user.classe
11    $domaine = "sio-exupery"
12    $ext = "local"
13    $hdrive = "H:"
14    $hdirectory = "\\AD\Home$"
15    $dossier = "$hdirectory\$samname"
16    $profil = "\\AD\Profil\$samname"
17
18    New-ADUser -Name "$name $givenname" -UserPrincipalName "$name@$domaine.$ext" -SamAccountName $samname -GivenName $givenname -Surname $surname -Password $password -ChangePasswordAtExpiration $true -ChangePasswordOnLogon $true -Enabled $true
19    Set-Location $hdirectory
20    New-Item -Name $samname -ItemType directory
21    Get-Item "$dossier" | Disable-NTFSAccessInheritance
22    Set-NTFSOwner -Path "$dossier" -Account "sio-exupery.local\$samname"
23    Add-NTFSAccess -Path $dossier -Account "sio-exupery.local\$samname" -AccessRights FullControl
24    Get-NTFSAccess -Path "$dossier" -Account "Utilisateurs" -ExcludeInherited | Remove-NTFSAccess
25    Add-ADGroupMember -Identity "$section" -Members $samname
26 }

```

Le script génère, pour chaque utilisateur, un répertoire personnel au nom de l'utilisateur ainsi que la suppression de l'héritage des permissions. Les cmdlets du module NTFSSecurity positionnent ensuite des autorisations NTFS (ajout de l'autorisation Contrôle total pour l'utilisateur concerné), définissent comme propriétaire l'utilisateur concerné et enfin, retirent les droits du groupe Utilisateurs sur le dossier personnel de l'utilisateur.

Removing access

Removing access is similar to adding permissions. The command **Remove-NTFSAccess** takes the same parameters as **Add-NTFSAccess**.

To remove a user from the ACL, provide the path, the account name, and the permissions you want to remove, for example:

```
Remove-NTFSAccess D:\Data -Account RAANDREE0\randr_000 -AccessRights Read -PassThru
```

If the user has different permissions than those you want to remove, nothing happens. There needs to be an exact match.

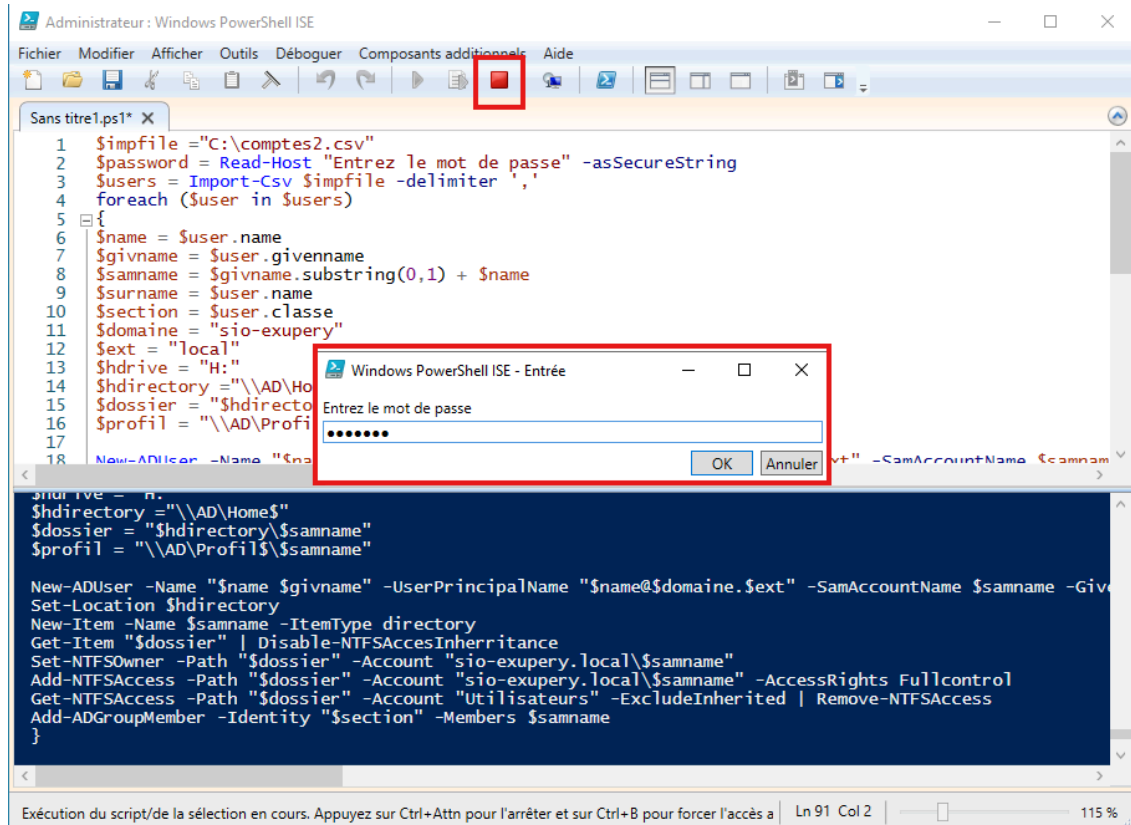
Note You cannot remove inherited permissions. **Get-NTFSAccess** informs about the source of the inherited permissions where the respective ACE can be changed or removed.

Remove-NTFSAccess accepts pipeline input. If you want to remove all permissions for a certain user account, you can read the permissions first and then pipe the results to **Remove-NTFSAccess**. This operation can also run recursively:

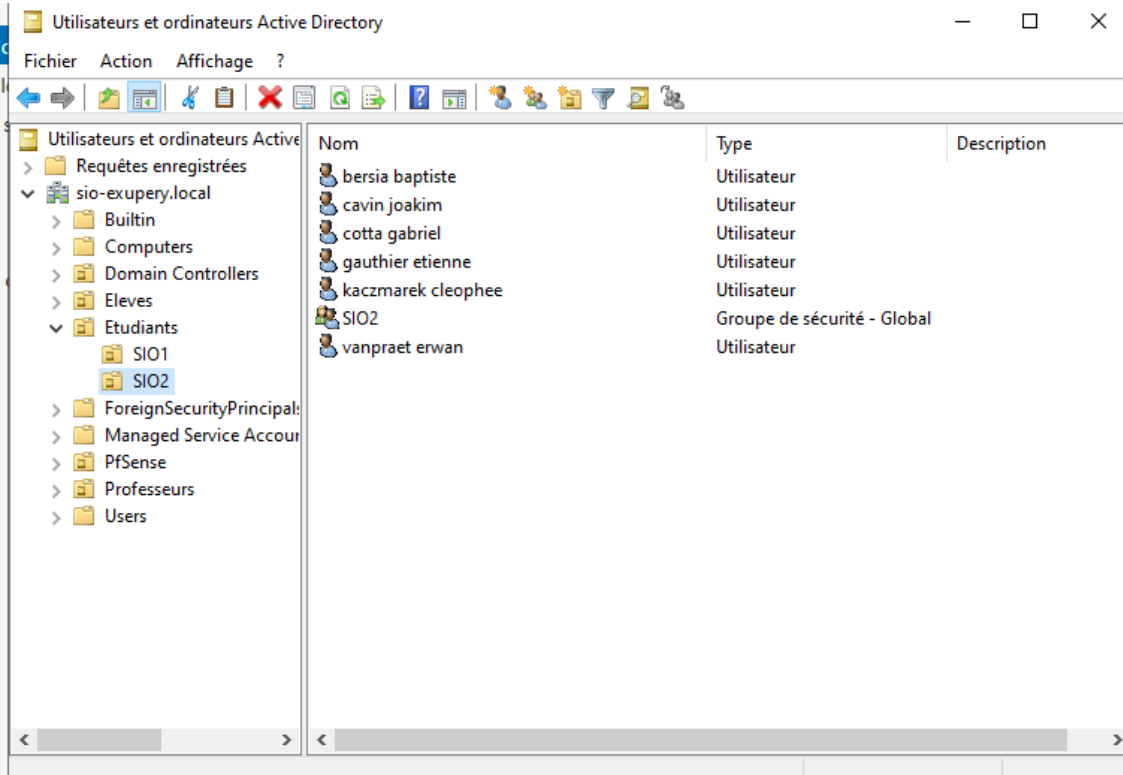
```
Get-ChildItem -Path d:\ -Recurse |
Get-NTFSAccess -Account raandree0\randr_000 -ExcludeInherited |
Remove-NTFSAccess
```

Note The cmdlets in the NTFSSecurity module do not provide a way to process files and folders recursively. You have to use **Get-ChildItem** or **Get-ChildItem2** with the **Recurse** switch. (The **Get-ChildItem2** cmdlet is part of the NTFSSecurity module, and it will be discussed in a future post.)

- Exécution du script :



- Nous constatons la création des comptes au sein de chaque UO et l'affectation aux groupes :



Propriétés de : barkallah adam

Environnement Sessions Contrôle à distance Profil des services Bureau à distance COM+
Général Adresse Compte Profil Téléphones Organisation Membre de Appel entrant

Nom d'ouverture de session de l'utilisateur :
barkallah @sio-exupery.local

Nom d'ouverture de session de l'utilisateur (antérieur à Windows 2000) :
SIO-EXUPERY\ abarkallah

Horaires d'accès... Se connecter à...

☐ Déverrouiller le compte

Options de compte :

- ☒ L'utilisateur devra changer le mot de passe
- ☐ L'utilisateur ne peut pas changer de mot de passe
- ☐ Le mot de passe n'expire jamais
- ☐ Enregistrer le mot de passe en utilisant un chiffrement réversible

Date d'expiration du compte

☒ Jamais

☐ Fin de : samedi 14 juin 2025

OK Annuler Appliquer Aide

Propriétés de : barkallah adam

Environnement Sessions Contrôle à distance Profil des services Bureau à distance COM+
Général Adresse Compte Profil Téléphones Organisation Membre de Appel entrant

Profil utilisateur

Chemin du profil : \\AD\Profil\$\abarkallah

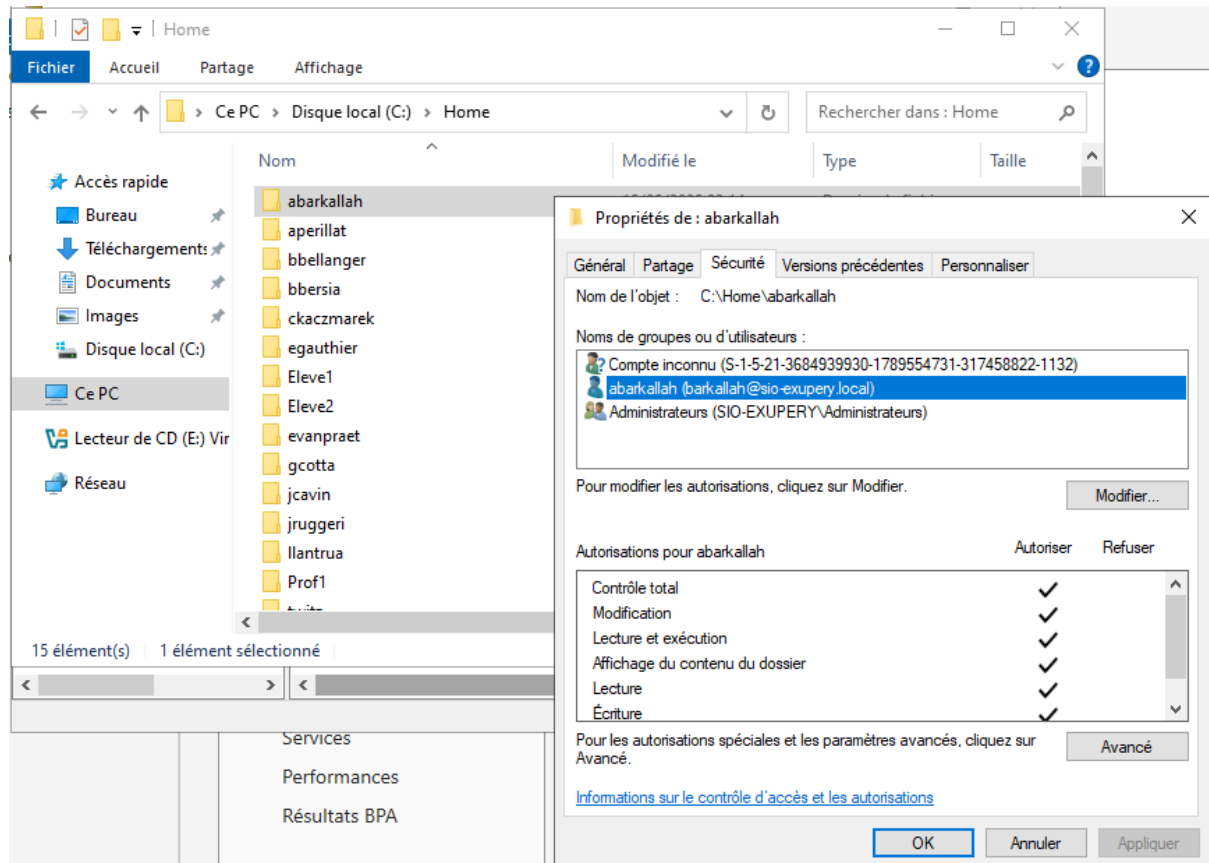
Script d'ouverture de session :

Dossier de base

☐ Chemin d'accès local :

☒ Connecter : H: à : \\AD\Home\$\abarkallah

- Nous vérifions les autorisations NTFS définies sur les répertoires personnels :



Le groupe Utilisateurs ne dispose plus d'autorisations NTFS sur les répertoires personnels.