

Déploiement et configuration d'un AD rapidement

Premier AD

1^{er} : Prérequis avant l'installation et la configuration

Renommer la machine :

```
##Renommer l'AD  
Rename-Computer -NewName "AD1" -DomainCredential Administrateur -Restart
```

Spécifications de l'appareil

Nom de l'appareil	AD1
Processeur	12th Gen Intel(R) Core(TM) i5-1235U 2.50 GHz
Mémoire RAM installée	3,91 Go
ID de périphérique	15F9A0AC- A082-4D4B-8E09-1184772AD7B6
ID de produit	00454-40000-00001-AA939
Type du système	Système d'exploitation 64 bits, processeur x64
Stylet et fonction tactile	La fonctionnalité d'entrée tactile ou avec un stylet n'est pas disponible sur cet écran

Réaliser séparément l'étape 1 des autres. Il faut changer le numéro de l'interface Index selon ce qu'on obtient avec l'étape 1. Sa correspond au port Ethernet allumé sur la machine. (Voir capture ci-dessous)

On aurait pu rajouter une ligne de code dans le script complet, pour que cela soit automatique, mais si la machine a plusieurs ports allumés sa peut créer des problèmes

```
PS C:\Users\Administrateur> ##Etape 1 : Savoir le port Ethernet  
Get-NetAdapter
```

Name	InterfaceDescription	ifIndex	Status
Ethernet	Intel(R) PRO/1000 MT Desktop Adapter	4	Up

Script pour obtenir le port Ethernet

##Étape 1 : Savoir le port Ethernet Get-NetAdapter

Script complet pour la configuration de l'adresse IP et DNS

```
##Étape 1 : Vérifier si l'adresse IP est déjà attribuée
$existingIP = Get-NetIPAddress -InterfaceIndex 4 -AddressFamily IPv4 | Where-Object { $_.IPAddress -eq "192.168.56.21" }
if (-not $existingIP) {
    Write-Host "Ajout de l'adresse IP 192.168.56.21 à l'interface 3."
    New-NetIPAddress -InterfaceIndex 4 -IPAddress 192.168.56.21 -PrefixLength 24
} else {
    Write-Host "L'adresse IP 192.168.56.21 est déjà attribuée."
}

##Étape 2 : Vérifier si l'adresse DNS est déjà configurée
$dnsServers = Get-DnsClientServerAddress -InterfaceIndex 4 | Select-Object -ExpandProperty ServerAddresses
if ($dnsServers -notcontains "192.168.56.21") {
    Write-Host "Configuration du serveur DNS 192.168.56.21..."
    Set-DnsClientServerAddress -InterfaceIndex 4 -ServerAddresses 192.168.56.21
} else {
    Write-Host "Le serveur DNS 192.168.56.21 est déjà configuré."
}

# Étape 3 : Vérifier la configuration DNS
Write-Host "Configuration DNS actuelle :"
Get-DnsClientServerAddress -InterfaceIndex 4
```

Résultat : Les adresses IP et DNS ont bien été configurer.

```
IPAddress      : 192.168.56.21
InterfaceIndex : 3
InterfaceAlias  : Ethernet
AddressFamily   : IPv4
Type            : Unicast
PrefixLength    : 24
PrefixOrigin    : Manual
SuffixOrigin     : Manual
AddressState    : Tentative
ValidLifetime   : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource    : False
PolicyStore     : ActiveStore

IPAddress      : 192.168.56.21
InterfaceIndex : 3
InterfaceAlias  : Ethernet
AddressFamily   : IPv4
Type            : Unicast
PrefixLength    : 24
PrefixOrigin    : Manual
SuffixOrigin     : Manual
AddressState    : Invalid
ValidLifetime   : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource    : False
PolicyStore     : PersistentStore
```

```
L'adresse IP 192.168.56.21 est déjà attribuée.
Le serveur DNS 192.168.56.21 est déjà configuré.
Configuration DNS actuelle :

InterfaceAlias      Interface Address ServerAddresses
-----
Ethernet            4 IPv4    {192.168.56.21}
Ethernet            4 IPv6    {::1}
```

PS C:\Users\Administrateur> |

2^{ème} : Configuration et Installation du 1^{er} AD.

Script pour l'installation et la configuration de l'AD

```

$FeatureList = @("AD-Domain-Services", "DNS")

Foreach($Feature in $FeatureList){
    $FeatureState = (Get-WindowsFeature -Name $Feature).InstallState

    if($FeatureState -eq "Available"){
        # Si la fonctionnalité est disponible mais pas installée, on tente de l'installer
        Write-Output "La fonctionnalité $Feature va être installée. "

        Try{
            Add-WindowsFeature -Name $Feature -IncludeManagementTools -IncludeAllSubFeature
            Write-Output "Installation réussie "
        } Catch{
            Write-Output "Erreur lors de l'installation. "
        }
    }
    elseif($FeatureState -eq "Installed"){
        # Si la fonctionnalité est déjà installée
        Write-Output "La fonctionnalité $Feature est déjà installée."
    }
    else {
        Write-Output "L'état de la fonctionnalité $Feature est inconnu."
    }
}

# Importer le module ADDSDeployment
Import-Module ADDSDeployment

# Vérifier si Active Directory est déjà installé
try {
    $domain = Get-ADDomain -ErrorAction Stop
    Write-Host "Le domaine $($domain.DNSRoot) existe déjà."
    exit 0
} catch {
    Write-Host "Aucun domaine Active Directory trouvé. Installation de l'active directory. "
}

# Vérifier la présence des fichiers AD (NTDS et SYSVOL)
$ntdsPath = "C:\Windows\NTDS"
$sysvolPath = "C:\Windows\SYSVOL"

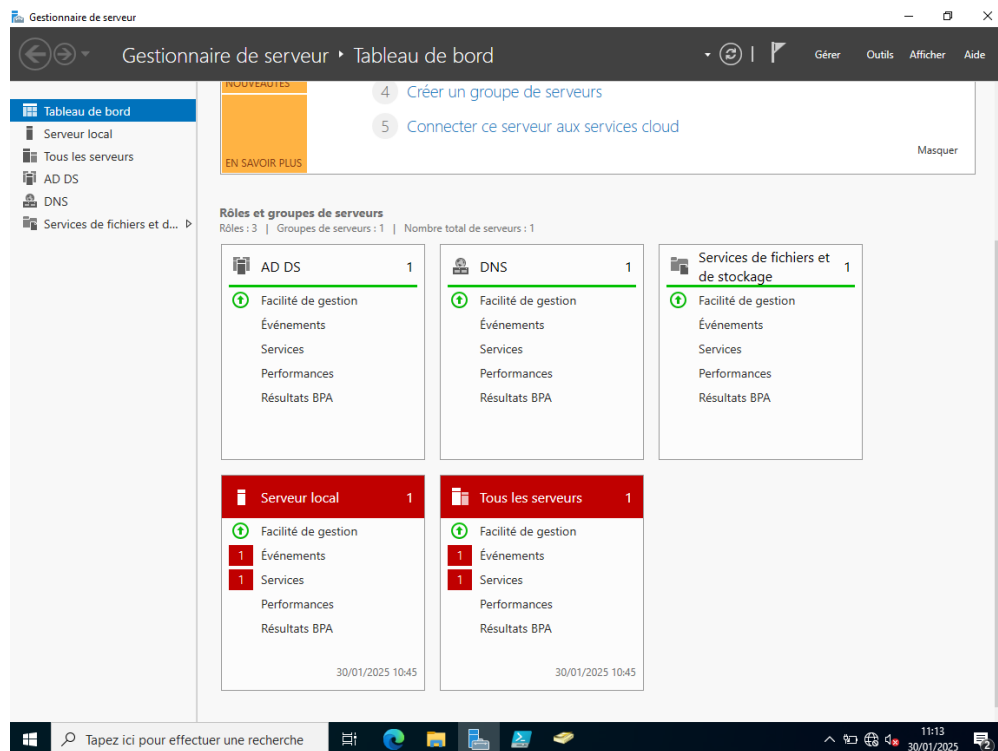
if (Test-Path $ntdsPath -and Test-Path $sysvolPath) {
    Write-Host "Les fichiers AD existent déjà ($ntdsPath et $sysvolPath), aucune installation requise. "
    exit 0
}

# Exécuter l'installation uniquement si AD n'est pas installé et que les fichiers n'existent pas
Install-ADDSForest `
    -CreateDnsDelegation:$false `
    -DatabasePath $ntdsPath `
    -DomainMode "winThreshold" `
    -DomainName "remoteworks.local" `
    -DomainNetbiosName "REMOTEWORKS" `
    -ForestMode "winThreshold" `
    -InstallDns:$true `
    -LogPath $ntdsPath `
    -NoRebootOnCompletion:$false `
    -SysvolPath $sysvolPath `
    -Force:$true

Write-Host "Installation de Active Directory terminée avec succès."

```

Résultat : AD installer et configurer.



```

32 | # Vérifier la présence des fichiers AD (NTDS et SYSVOL)
33 | exit 0
34 | } catch {
35 |     Write-Host "Aucun domaine Active Directory trouvé. Installation de l'active directory. "
36 | }
37 |
38 | # Vérifier la présence des fichiers AD (NTDS et SYSVOL)
39 | $ntdsPath = "C:\windows\NTDS"
40 | $sysvolPath = "C:\Windows\SYSVOL"
41 |
42 | if (Test-Path $ntdsPath -and Test-Path $sysvolPath) {
43 |     Write-Host "Les fichiers AD existent déjà ($ntdsPath et $sysvolPath), aucune installation requise. "
44 |     exit 0
45 | }
46 |
47 | # Exécuter l'installation uniquement si AD n'est pas installé et que les fichiers n'existent pas
48 | Install-ADDSForest `
49 |     -CreateDnsDelegation:$false `
50 |     -DatabasePath $ntdsPath `

```

```

# Exécuter l'installation uniquement si AD n'est pas installé et que les fichiers n'existent pas
Install-ADDSForest `
    -CreateDnsDelegation:$false `
    -DatabasePath $ntdsPath `
    -DomainMode "WinThreshold" `
    -DomainName "remoteworks.local" `
    -DomainNetbiosName "REMOTETWORKS" `
    -ForestMode "WinThreshold" `
    -InstallDns:$true `
    -LogPath $ntdsPath `
    -NoRebootOnCompletion:$false `
    -SysvolPath $sysvolPath `
    -Force:$true

Write-Host "Installation de Active Directory terminée avec succès."

La fonctionnalité AD-Domain-Services est déjà installée.
La fonctionnalité DNS est déjà installée.
Le domaine remoteworks.local existe déjà.

```

Deuxième AD

Avant tout chose si on clone la machine de l'AD1 on doit exécuter un sysprep, pour ne pas avoir le même numéro de machine

Commande pour le sysprep :

```
cd C:\Windows\System32\Sysprep
```

```
sysprep.exe
```

1^{er} : Prérequis avant l'installation

Renommer la machine :

```
##Renommer l'AD  
Rename-Computer -NewName "AD2" -DomainCredential Administrateur -Restart
```

Réaliser séparément l'étape 1 des autres. Il faut changer le numéro de l'interface Index selon ce qu'on obtient avec l'étape 1. Sa correspond au port Ethernet allumé sur la machine (ifIndex). Voir capture ci-dessous,

```
PS C:\Users\Administrateur.REMOTEWORKS> ##Etape 1 : Savoir le port Ethernet  
Get-NetAdapter
```

Name	InterfaceDescription	ifIndex	Status	MacAddress	LinkSpeed
Ethernet	Intel(R) PRO/1000 MT Desktop Adapter	15	Up	08-00-27-CA-10-08	1 Gbps

On peut modifier les InterfaceIndex selon l'interface Ethernet à laquelle on veut ajouter les adresses.

Script pour obtenir le port Ethernet

```
##Etape 1 : Savoir le port Ethernet  
Get-NetAdapter
```

Script complet pour la configuration de l'adresse IP et DNS

```
##Etape 2 : Vérifier si l'adresse IP est déjà attribuée
$existingIP = Get-NetIPAddress -InterfaceIndex 15 -AddressFamily IPv4 | Where-Object { $_.IPAddress -eq "192.168.56.22" }
if (-not $existingIP) {
    Write-Host "Ajout de l'adresse IP 192.168.56.22 à l'interface 15."
    New-NetIPAddress -InterfaceIndex 15 -IPAddress 192.168.56.22 -PrefixLength 24
} else {
    Write-Host "L'adresse IP 192.168.56.22 est déjà attribuée."
}

##Etape 3 : Vérifier si l'adresse DNS est déjà configurée
$dnsServers = Get-DnsClientServerAddress -InterfaceIndex 15 | Select-Object -ExpandProperty ServerAddresses
if ($dnsServers -notcontains "192.168.56.21") {
    Write-Host "Configuration du serveur DNS 192.168.56.21..."
    Set-DnsClientServerAddress -InterfaceIndex 15 -ServerAddresses 192.168.56.21
} else {
    Write-Host "Le serveur DNS 192.168.56.21 est déjà configuré."
}

# Etape 4 : Vérifier la configuration DNS
Write-Host "Configuration DNS actuelle :"
Get-DnsClientServerAddress -InterfaceIndex 15
```

L'adresse DNS est l'adresse IP du premier AD pour qu'ils arrivent à communiquer entre eux.

Résultat : Les adresses IP et DNS sont configurées.

```
L'adresse IP 192.168.56.22 est déjà attribuée.
Le serveur DNS 192.168.56.21 est déjà configuré.
Configuration DNS actuelle :
```

InterfaceAlias	Interface Index	Address Family	ServerAddresses
Ethernet	15	IPv4	{192.168.56.21, 127.0.0.1}
Ethernet	15	IPv6	:::1

```
PS C:\Users\Administrateur.REMOTEWORKS>
```

2^{ème} Ajouter le deuxième AD comme contrôleur de domaine de la forêt du premier AD.

```
##Ajout de l'AD comme contrôleur de domaine
if (-not (Get-ADDomainController -Filter {Domain -eq "remoteworks.local"})) {
    Import-Module ADDSDeployment
    Install-ADDSDomainController `
        -NoGlobalCatalog:$true `
        -CreateDnsDelegation:$false `
        -Credential (Get-Credential) `
        -CriticalReplicationOnly:$false `
        -DatabasePath "C:\Windows\NTDS" `
        -DomainName "remoteworks.local" `
        -InstallDns:$true `
        -LogPath "C:\Windows\NTDS" `
        -NoRebootOnCompletion:$false `
        -SiteName "Default-First-Site-Name" `
        -SysvolPath "C:\Windows\SYSVOL" `
        -Force:$true

    Write-Host "Le contrôleur de domaine pour 'remoteworks.local' a été installé avec succès."
} else {
    Write-Host "Le contrôleur de domaine pour 'remoteworks.local' existe déjà."
}
```


Ajouter un Windows 11 au domaine

1^{er} Configuration requise avant de l'ajouter

Pour ne pas avoir de problème avec les scripts, ouvrir le PowerShell avec les droits administrateurs, vu que l'ordinateur n'est pas encore ajouté au domaine il faut utiliser l'administrateur local du pc

Renommer l'ordinateur

```
##Renommer le PC  
Rename-Computer -NewName "TEST" -DomainCredential Administrateur -Restart
```

On modifie son adresse IP et DNS en mettant comme DNS l'adresse IP du premier AD. Comme pour les AD on réalise l'étape 1 avant et on modifie le numéro de l'interface Ethernet dans le script selon celle à qui on veut ajouter les adresses.

Script complet pour l'adressage IP et DNS

Pour l'adresse DNS on met l'adresse IP de l'AD1 pour que la machine puisse communiquer avec l'AD1, et donc plus tard sa nous permettra de l'ajouter au domaine.

Script pour obtenir l'interface réseaux

```
##Mise en place adresse ip  
##Etape 1 :  
Get-NetAdapter
```

```
PS C:\Users\Albert> ##Mise en place adresse ip  
##Etape 1 :  
Get-NetAdapter
```

Name	InterfaceDescription	ifIndex	Status	MacAddress	LinkSpeed
Ethernet	Intel(R) PRO/1000 MT Desktop Adapter	10	Up	08-00-27-8C-E1-BA	1 Gbps

Script complet pour la configuration de l'adresse IP et DNS

```
##Etape 1B
$existingIP = Get-NetIPAddress -InterfaceIndex 10 -AddressFamily IPv4 | Where-Object { $_.IPAddress -eq "192.168.56.20" }

if (-not $existingIP) {
    Write-Host "Ajout de l'adresse IP 192.168.56.20 à l'interface 10."
    New-NetIPAddress -InterfaceIndex 10 -IPAddress 192.168.56.20 -PrefixLength 24
} else {
    Write-Host "L'adresse IP 192.168.56.20 est déjà attribuée."
}

##Etape 2 : Mise en place du dns, rappel le dns est l'adresse de l'AD1
$dnsServers = Get-DnsClientServerAddress -InterfaceIndex 10 | Select-Object -ExpandProperty ServerAddresses

if ($dnsServers -notcontains "192.168.56.21") {
    Write-Host "Configuration du serveur DNS 192.168.56.21..."
    Set-DnsClientServerAddress -InterfaceIndex 10 -ServerAddresses 192.168.56.21
} else {
    Write-Host "Le serveur DNS 192.168.56.21 est déjà configuré."
}

##Etape 3 : Vérifier la configuration DNS
Write-Host "Configuration DNS actuelle :"
Get-DnsClientServerAddress -InterfaceIndex 10
```

```
L'adresse IP 192.168.56.20 est déjà attribuée.
Le serveur DNS 192.168.56.21 est déjà configuré.
Configuration DNS actuelle :
```

InterfaceAlias	Interface Index	Address Family	ServerAddresses
Ethernet	10	IPv4	{192.168.56.21}
Ethernet	10	IPv6	{}

```
PS C:\Users\Albert> |
```

2^{ème} Ajouter l'ordinateur au domaine

Script complet pour ajouter l'ordinateur au domaine

```
$domain = "remoteworks.local"
$domainExists = (Get-WmiObject -Class Win32_ComputerSystem).Domain

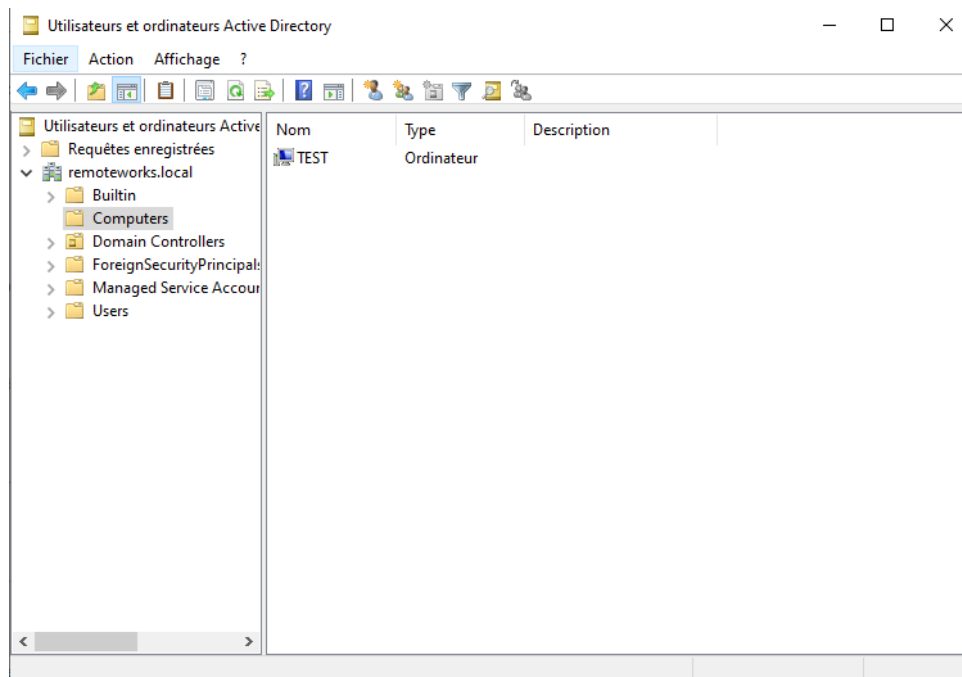
# Vérifiez si l'ordinateur est déjà membre du domaine
if ($domainExists -ne $domain) {
    Add-Computer -DomainName $domain -Credential Administrateur
} else {
    Write-Host "L'ordinateur est déjà membre du domaine $domain."
}
```

Résultat : L'ordinateur apparaît bien dans le groupe « Computer » du domaine.

```
PS C:\Users\Albert> $domain = "remoteworks.local"
$domainExists = (Get-WmiObject -Class Win32_ComputerSystem).Domain

# Vérifiez si l'ordinateur est déjà membre du domaine
if ($domainExists -ne $domain) {
    Add-Computer -DomainName $domain -Credential Administrateur
} else {
    Write-Host "L'ordinateur est déjà membre du domaine $domain."
}

L'ordinateur est déjà membre du domaine remoteworks.local.
PS C:\Users\Albert> |
```



Créer un utilisateur sur le domaine ou le supprimer.

1^{er} Créer un utilisateur

Script pour crée un Utilisateur

```
# Définition des variables
$UserSamAccountName = "hzidane"
$UserUPN = "hzidane@remoteworks.local"
$UserGivenName = "Henri"
$UserSurname = "Zidane"
$UserDisplayName = "$UserGivenName $UserSurname"
$UserPassword = "MotDePasse123!"

# Vérifier si l'utilisateur existe déjà
$ExistingUser = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue

if ($ExistingUser) {
    Write-Output "L'utilisateur '$UserSamAccountName' existe déjà."
} else {
    Write-Output "Création de l'utilisateur '$UserSamAccountName'..."

    New-ADUser -Name $UserDisplayName `
        -GivenName $UserGivenName `
        -Surname $UserSurname `
        -SamAccountName $UserSamAccountName `
        -UserPrincipalName $UserUPN `
        -AccountPassword (ConvertTo-SecureString $UserPassword -AsPlainText -Force) `
        -Enabled $true `
        -ChangePasswordAtLogon $true

    Write-Output "L'utilisateur '$UserSamAccountName' a été créé avec succès."
}
```

On peut le refaire autant de fois qu'on veut il faut juste modifier les variables.

Résultat : Les utilisateurs ont bien été créés.

```

2 $UserSamAccountName = "hzidane"
3 $UserUPN = "hzidane@remoteworks.local"
4 $UserGivenName = "Henri"
5 $UserSurname = "Zidane"
6 $UserDisplayName = "$UserGivenName $UserSurname"
7 $UserPassword = "MotDePasse123!"
8
9 # Vérifier si l'utilisateur existe déjà
10 $ExistingUser = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue
11
12 if ($ExistingUser) {
13     Write-Output "L'utilisateur '$UserSamAccountName' existe déjà."
14 } else {
15     Write-Output "Création de l'utilisateur '$UserSamAccountName'..."
16
17     New-ADUser -Name $UserDisplayName `
18         -GivenName $UserGivenName `
19         -Surname $UserSurname `
20         -SamAccountName $UserSamAccountName `
21         -UserPrincipalName $UserUPN `
22         -AccountPassword (ConvertTo-SecureString $UserPassword -AsPlainText -Force) `
23         -Enabled $true `
24         -ChangePasswordAtLogon $true
25
26     Write-Output "L'utilisateur '$UserSamAccountName' a été créé avec succès."
27 }
28
$UserGivenName = "Henri"
$UserSurname = "Zidane"
$UserDisplayName = "$UserGivenName $UserSurname"
$UserPassword = "MotDePasse123!"
# Vérifier si l'utilisateur existe déjà
$ExistingUser = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue
if ($ExistingUser) {
    Write-Output "L'utilisateur '$UserSamAccountName' existe déjà."
} else {
    Write-Output "Création de l'utilisateur '$UserSamAccountName'..."
    New-ADUser -Name $UserDisplayName `
        -GivenName $UserGivenName `
        -Surname $UserSurname `
        -SamAccountName $UserSamAccountName `
        -UserPrincipalName $UserUPN `
        -AccountPassword (ConvertTo-SecureString $UserPassword -AsPlainText -Force) `
        -Enabled $true `
        -ChangePasswordAtLogon $true
    Write-Output "L'utilisateur '$UserSamAccountName' a été créé avec succès."
}
Création de l'utilisateur 'hzidane'...
L'utilisateur 'hzidane' a été créé avec succès.
PS C:\Users\Administrateur>
  
```

Nom	Type	Description
Administrat...	Utilisateur	Compte d'utilisateur d'a...
Albert Zidane	Utilisateur	
Henri Zidane	Utilisateur	
Invité	Utilisateur	Compte d'utilisateur inv...
Thomas TB. ...	Utilisateur	
Vladimir Vitr...	Utilisateur	
Administrat...	Groupe de séc...	Les membres de ce grou...
Administrat...	Groupe de séc...	Administrateurs désigné...
Administrat...	Groupe de séc...	Administrateurs désigné...
Contrôleurs ...	Groupe de séc...	Les membres de ce grou...
Administrat...	Groupe de séc...	Les membres de ce grou...
Admins du ...	Groupe de séc...	Administrateurs désigné...
Contrôleurs ...	Groupe de séc...	Tous les contrôleurs de ...
Contrôleurs ...	Groupe de séc...	Les membres de ce grou...
Contrôleurs ...	Groupe de séc...	Les membres de ce grou...
DnsUpdateP...	Groupe de séc...	Les clients DNS qui sont ...
Invités du d...	Groupe de séc...	Tous les invités du doma...
Ordinateurs ...	Groupe de séc...	Toutes les stations de tra...
Propriétaires...	Groupe de séc...	Les membres de ce grou...
Protected Us...	Groupe de séc...	Les membres de ce grou...
Utilisateurs ...	Groupe de séc...	Tous les utilisateurs du d...
DnsAdmins	Groupe de séc...	Groupe des administrate...
Éditeurs de c...	Groupe de séc...	Les membres de ce grou...
Groupe de r...	Groupe de séc...	Les mots de passe des ...
Groupe de r...	Groupe de séc...	Les mots de passe des ...
Serveurs RA...	Groupe de séc...	Les serveurs de ce group...

2^{ème} Supprimer un utilisateur

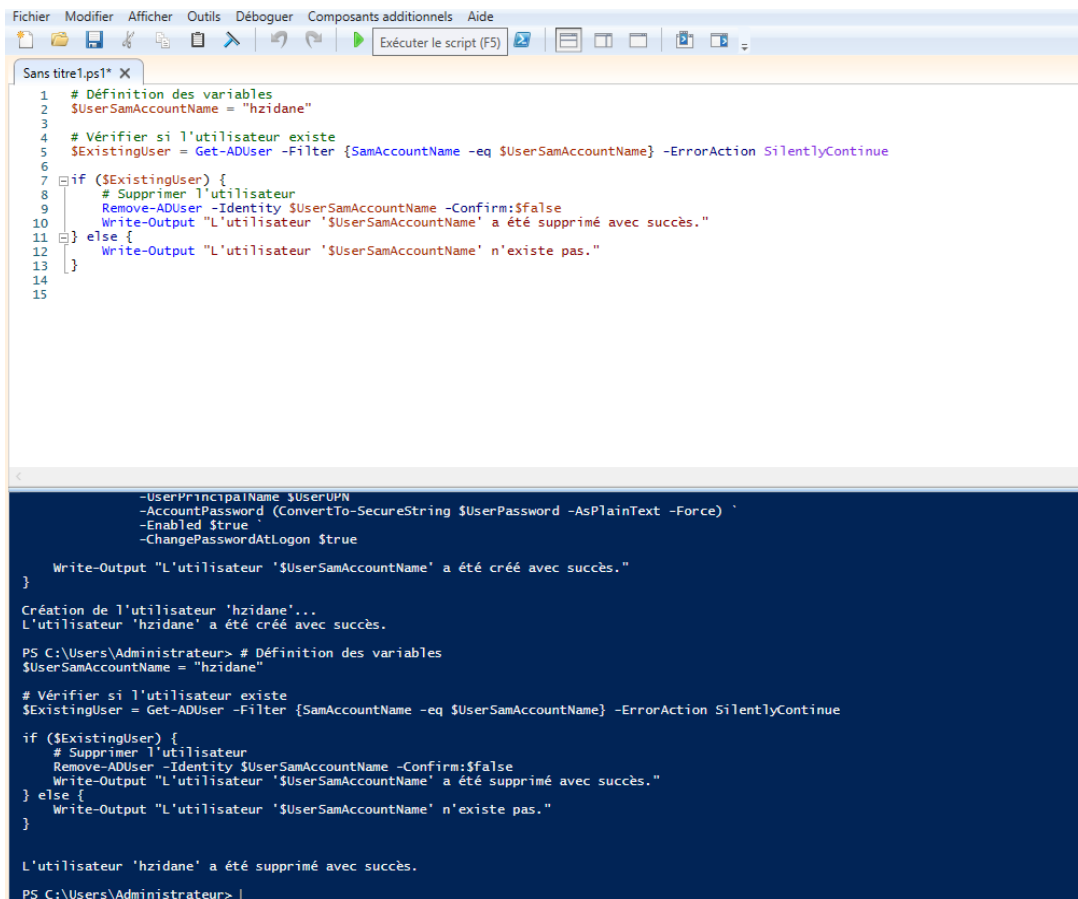
Script pour supprimer l'utilisateur

```
# Définition des variables
$UserSamAccountName = "hzidane"

# Vérifier si l'utilisateur existe
$ExistingUser = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue

if ($ExistingUser) {
    # Supprimer l'utilisateur
    Remove-ADUser -Identity $UserSamAccountName -Confirm:$false
    Write-Output "L'utilisateur '$UserSamAccountName' a été supprimé avec succès."
} else {
    Write-Output "L'utilisateur '$UserSamAccountName' n'existe pas."
}
```

Résultat : L'utilisateur a bien été supprimé



The screenshot shows a PowerShell script editor window titled 'Sans titre1.ps1' with a menu bar (Fichier, Modifier, Afficher, Outils, Déboguer, Composants additionnels, Aide) and a toolbar. The script content is as follows:

```
1 # Définition des variables
2 $UserSamAccountName = "hzidane"
3
4 # Vérifier si l'utilisateur existe
5 $ExistingUser = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue
6
7 if ($ExistingUser) {
8     # Supprimer l'utilisateur
9     Remove-ADUser -Identity $UserSamAccountName -Confirm:$false
10    Write-Output "L'utilisateur '$UserSamAccountName' a été supprimé avec succès."
11 } else {
12    Write-Output "L'utilisateur '$UserSamAccountName' n'existe pas."
13 }
14
15
```

Below the script editor, the execution output is displayed in a dark blue console window. It shows the script being executed from the command prompt, with the following output:

```
-UserPrincipalName $UserUPN
-AccountPassword (ConvertTo-SecureString $UserPassword -AsPlainText -Force) `
-Enabled $true `
-ChangePasswordAtLogon $true

Write-Output "L'utilisateur '$UserSamAccountName' a été créé avec succès."
}

Création de l'utilisateur 'hzidane'...
L'utilisateur 'hzidane' a été créé avec succès.

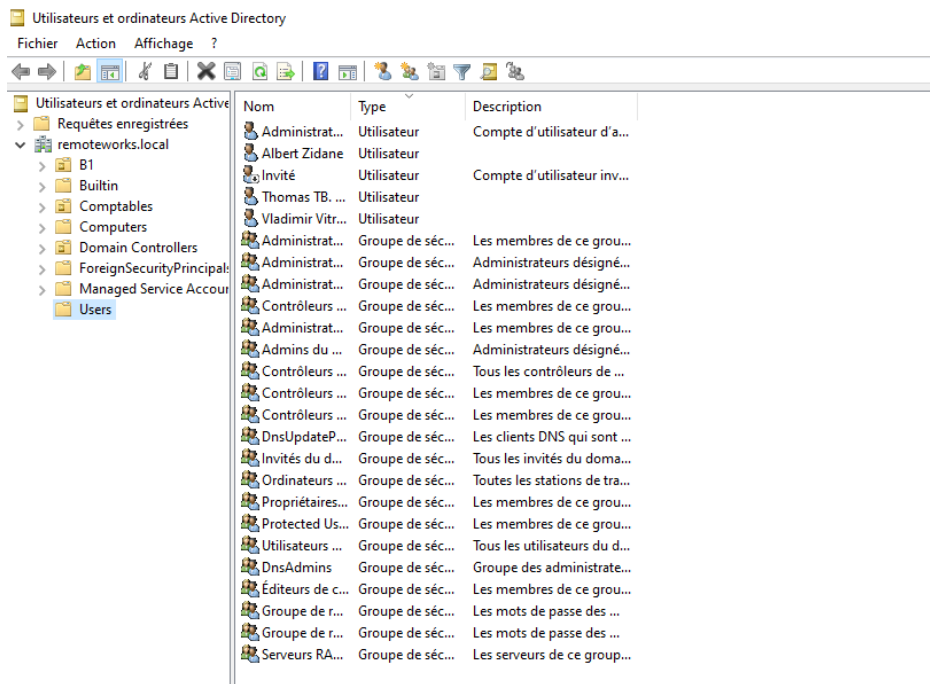
PS C:\Users\Administrateur> # Définition des variables
$UserSamAccountName = "hzidane"

# Vérifier si l'utilisateur existe
$ExistingUser = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue

if ($ExistingUser) {
    # Supprimer l'utilisateur
    Remove-ADUser -Identity $UserSamAccountName -Confirm:$false
    Write-Output "L'utilisateur '$UserSamAccountName' a été supprimé avec succès."
} else {
    Write-Output "L'utilisateur '$UserSamAccountName' n'existe pas."
}

L'utilisateur 'hzidane' a été supprimé avec succès.

PS C:\Users\Administrateur> |
```



Création d'une unité d'organisation ou OU et ajouter ou supprimer un utilisateur de celle-ci.

1^{er} Création de l'OU

Script complet

```
# Demande du nom de l'OU à l'utilisateur
$NomOU = Read-Host "Entrez le nom de l'OU"
$Domaine = (Get-ADDomain).DistinguishedName # Récupère le domaine automatiquement

# Vérifie si l'OU existe déjà
if (Get-ADOrganizationalUnit -Filter "Name -eq '$NomOU'" -ErrorAction SilentlyContinue) {
    Write-Host "L'OU '$NomOU' existe déjà dans le domaine." -ForegroundColor Yellow
} else {
    # Création de l'OU
    New-ADOrganizationalUnit -Name $NomOU -Path $Domaine -ProtectedFromAccidentalDeletion $true
    Write-Host "L'OU '$NomOU' a été créée avec succès !" -ForegroundColor Green
}
```

Résultat : L'OU est bien créée

```
PS C:\Users\Administrateur> # Demande du nom de l'OU à l'utilisateur
$NomOU = Read-Host "Entrez le nom de l'OU"
$Domaine = (Get-ADDomain).DistinguishedName # Récupère le domaine automatiquement

# Vérifie si l'OU existe déjà
if (Get-ADOrganizationalUnit -Filter "Name -eq '$NomOU'" -ErrorAction SilentlyContinue) {
    Write-Host "L'OU '$NomOU' existe déjà dans le domaine." -ForegroundColor Yellow
} else {
    # Création de l'OU
    New-ADOrganizationalUnit -Name $NomOU -Path $Domaine -ProtectedFromAccidentalDeletion $true
    Write-Host "L'OU '$NomOU' a été créée avec succès !" -ForegroundColor Green
}
Entrez le nom de l'OU : B1
L'OU 'B1' existe déjà dans le domaine.

PS C:\Users\Administrateur> |
```

2^{ème} Ajouter un Utilisateur

Script complet

```
# Définition des variables
$UserSamAccountName = "Albert"
$OUPath = "OU=B1,DC=remoteworks,DC=local"

# Vérifier si l'utilisateur existe
$User = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue

if ($User) {
    # Vérifier si l'utilisateur est déjà dans la bonne OU
    if ($User.DistinguishedName -match $OUPath) {
        Write-Output "L'utilisateur '$UserSamAccountName' est déjà dans l'OU spécifiée."
    } else {
        # Déplacer l'utilisateur dans l'OU cible
        Move-ADObject -Identity $User.DistinguishedName -TargetPath $OUPath
        Write-Output "L'utilisateur '$UserSamAccountName' a été déplacé dans l'OU '$OUPath'."
    }
} else {
    Write-Output "L'utilisateur '$UserSamAccountName' n'existe pas. Vérifiez le nom d'utilisateur."
}
```

Résultat : L'utilisateur a bien été déplacé vers l'OU ciblé

```
PS C:\Users\Administrateur> # Définition des variables
$UserSamAccountName = "Albert"
$OUPath = "OU=B1,DC=remoteworks,DC=local"

# Vérifier si l'utilisateur existe
$User = Get-ADUser -Filter {SamAccountName -eq $UserSamAccountName} -ErrorAction SilentlyContinue

if ($User) {
    # Vérifier si l'utilisateur est déjà dans la bonne OU
    if ($User.DistinguishedName -match $OUPath) {
        Write-Output "L'utilisateur '$UserSamAccountName' est déjà dans l'OU spécifiée."
    } else {
        # Déplacer l'utilisateur dans l'OU cible
        Move-ADObject -Identity $User.DistinguishedName -TargetPath $OUPath
        Write-Output "L'utilisateur '$UserSamAccountName' a été déplacé dans l'OU '$OUPath'."
    }
} else {
    Write-Output "L'utilisateur '$UserSamAccountName' n'existe pas. Vérifiez le nom d'utilisateur."
}

L'utilisateur 'Albert' a été déplacé dans l'OU 'OU=B1,DC=remoteworks,DC=local'.
```

Utilisateurs et ordinateurs Active		
> Requetes enregistrées		
▼ remoteworks.local		
B1		
> Builtin		
> Comptables		
> Computers		
> Domain Controllers		
> ForeignSecurityPrincipal:		
> Managed Service Accour		
Users		
	Nom	Type
	Albert Zidane	Utilisateur
	Linux	Groupe de séc...
	Python	Groupe de séc...
	Réseaux	Groupe de séc...
	Description	

3^{ème} : Supprimer l'utilisateur d'une OU et le déplacer dans le container

« Users »

On veut supprimer l'utilisateur de l'OU et le déplacer dans le container

« Users ». Sinon on peut aussi le supprimer sans le déplacer mais dans ce cas-là il est aussi supprimé du domaine.

Script complet pour déplacer l'utilisateur dans le container « Users »

```
# Définir le nom de l'utilisateur, l'OU actuelle et le conteneur "Users"
$UserName = "Albert"
$OUPath = "OU=B1,DC=remoteworks,DC=local"
$DefaultContainer = "CN=Users,DC=remoteworks,DC=local" # Conteneur par défaut

# Vérifier si l'utilisateur existe dans l'OU
$User = Get-ADUser -Filter {SamAccountName -eq $UserName} -SearchBase $OUPath -ErrorAction SilentlyContinue

if ($User) {
    Write-Host "L'utilisateur '$UserName' existe dans l'OU '$OUPath'. Déplacement vers '$DefaultContainer'..."

    # Déplacer l'utilisateur vers le conteneur "Users"
    Move-ADObject -Identity $User.DistinguishedName -TargetPath $DefaultContainer

    Write-Host "L'utilisateur '$UserName' a été déplacé avec succès vers '$DefaultContainer'."
} else {
    Write-Host "L'utilisateur '$UserName' n'existe pas dans l'OU '$OUPath'."
}
```

Résultat : L'utilisateur est bien retourné dans le container « Users »

```
PS C:\Users\Administrateur> # Définir le nom de l'utilisateur, l'OU actuelle et le conteneur "Users"
$UserName = "Albert"
$OUPath = "OU=B1,DC=remoteworks,DC=local"
$DefaultContainer = "CN=Users,DC=remoteworks,DC=local" # Conteneur par défaut

# Vérifier si l'utilisateur existe dans l'OU
$User = Get-ADUser -Filter {SamAccountName -eq $UserName} -SearchBase $OUPath -ErrorAction SilentlyContinue

if ($User) {
    Write-Host "L'utilisateur '$UserName' existe dans l'OU '$OUPath'. Déplacement vers '$DefaultContainer'..."

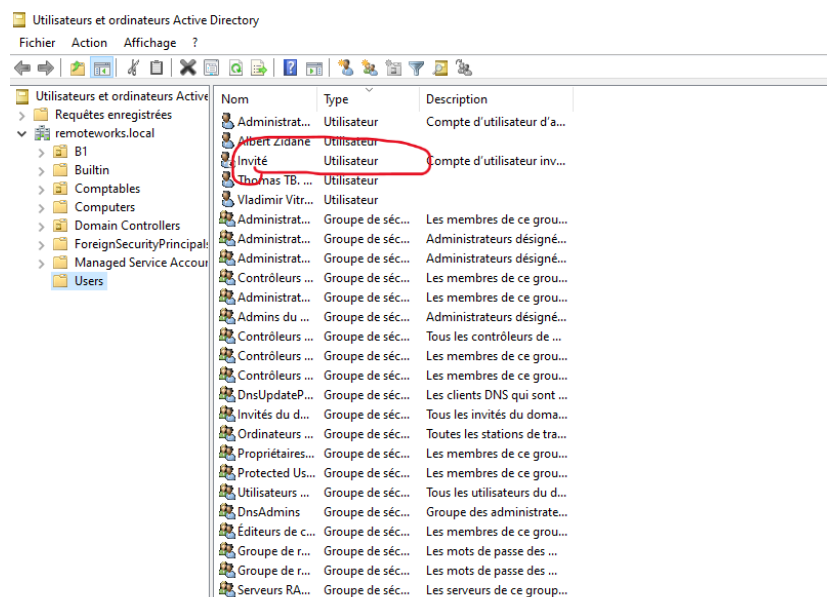
    # Déplacer l'utilisateur vers le conteneur "Users"
    Move-ADObject -Identity $User.DistinguishedName -TargetPath $DefaultContainer

    Write-Host "L'utilisateur '$UserName' a été déplacé avec succès vers '$DefaultContainer'."
} else {
    Write-Host "L'utilisateur '$UserName' n'existe pas dans l'OU '$OUPath'."
}

L'utilisateur 'Albert' existe dans l'OU 'OU=B1,DC=remoteworks,DC=local'. Déplacement vers 'CN=Users,DC=remoteworks,DC=local'...
L'utilisateur 'Albert' a été déplacé avec succès vers 'CN=Users,DC=remoteworks,DC=local'.
```

Utilisateurs et ordinateurs Active Directory

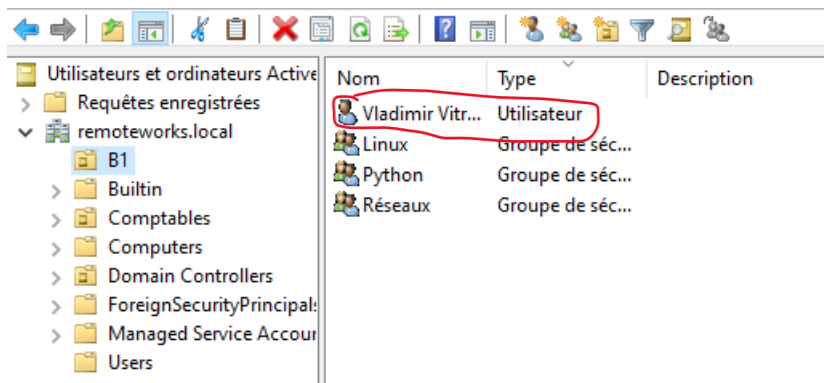
Fichier Action Affichage ?



Nom	Type	Description
Administrat...	Utilisateur	Compte d'utilisateur d'a...
Albert Zidane	Utilisateur	
Invité	Utilisateur	Compte d'utilisateur inv...
Thomas TB. ...	Utilisateur	
Vladimir Vitr...	Utilisateur	
Administrat...	Groupe de séc...	Les membres de ce grou...
Administrat...	Groupe de séc...	Administrateurs désigné...
Administrat...	Groupe de séc...	Administrateurs désigné...
Contrôleurs ...	Groupe de séc...	Les membres de ce grou...
Administrat...	Groupe de séc...	Les membres de ce grou...
Admins du ...	Groupe de séc...	Administrateurs désigné...
Contrôleurs ...	Groupe de séc...	Tous les contrôleurs de ...
Contrôleurs ...	Groupe de séc...	Les membres de ce grou...
Contrôleurs ...	Groupe de séc...	Les membres de ce grou...
DnsUpdateP...	Groupe de séc...	Les clients DNS qui sont ...
Invités du d...	Groupe de séc...	Tous les invités du doma...
Ordinateurs ...	Groupe de séc...	Toutes les stations de tra...
Propriétaires...	Groupe de séc...	Les membres de ce grou...
Protected Us...	Groupe de séc...	Les membres de ce grou...
Utilisateurs ...	Groupe de séc...	Tous les utilisateurs du d...
DnsAdmins	Groupe de séc...	Groupe des administrate...
Éditeurs de c...	Groupe de séc...	Les membres de ce grou...
Groupe de r...	Groupe de séc...	Les mots de passe des ...
Groupe de r...	Groupe de séc...	Les mots de passe des ...
Serveurs RA...	Groupe de séc...	Les serveurs de ce grou...

Si on veut le supprimer, sans le déplacer de l'OU et du domaine.

Preuve que notre utilisateur qu'on veut supprimer est dans l'OU



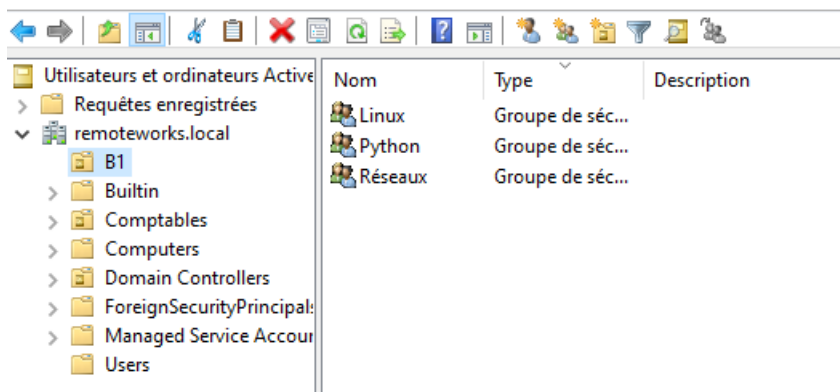
Script pour supprimer définitivement l'utilisateur du domaine

```
# Définir le nom de l'utilisateur et l'OU
$UserName = "VVitrou"
$OUPath = "OU=B1,DC=remoteworks,DC=local"

# Vérifier si l'utilisateur existe dans l'OU
$User = Get-ADUser -Filter {SamAccountName -eq $UserName} -SearchBase $OUPath -ErrorAction SilentlyContinue

if ($User) {
    # Supprimer l'utilisateur
    Write-Host "L'utilisateur '$UserName' existe. Suppression de l'utilisateur..."
    Remove-ADUser -Identity $User -Confirm:$false
} else {
    Write-Host "L'utilisateur '$UserName' n'existe pas dans l'OU '$OUPath'."
}
```

Résultat : L'utilisateur a été supprimé définitivement du domaine.



Pour voir s'il est supprimé du domaine, on exécute le script suivant :

```
$UserName = "VVitrou"

# Tenter de récupérer l'utilisateur
$User = Get-ADUser -Filter {SamAccountName -eq $UserName} -ErrorAction SilentlyContinue

if ($User) {
    Write-Host "L'utilisateur '$UserName' existe."
} else {
    Write-Host "L'utilisateur '$UserName' n'existe pas."
}
```


Résultat : L'utilisateur est totalement supprimé

```
PS C:\Users\Administrateur> $UserName = "VWitrou"
# Tenter de récupérer l'utilisateur
$User = Get-ADUser -Filter {SamAccountName -eq $UserName} -ErrorAction SilentlyContinue
if ($User) {
    Write-Host "L'utilisateur '$UserName' existe."
} else {
    Write-Host "L'utilisateur '$UserName' n'existe pas."
}
L'utilisateur 'VWitrou' n'existe pas.
PS C:\Users\Administrateur>
```

Sinon on peut aussi simplement entre la commande Get-ADUser <nom du compte> et on doit recevoir une erreur, qui précise qu'elle ne trouve pas l'objet <nom du compte>

```
PS C:\Users\Administrateur> Get-ADUser VWitrou
Get-ADUser : Impossible de trouver un objet avec l'identité « VWitrou » sous : « DC=remoteworks,DC=local ».
```

Création d'un groupe et ajouter ou supprimer un utilisateur.

1^{er} Création d'un groupe

Script : Pour le nom ne pas mettre d'espace, sa pose un problème pour ajouter un utilisateur au groupe.

```
$GroupName = "WindowsServer"

if (-not (Get-ADGroup -Filter {Name -eq $GroupName})) {
    New-ADGroup -Name $GroupName -Path "OU=B1,DC=REMOTEWORKS,DC=LOCAL" -GroupScope Global
    Write-Host "Le groupe '$GroupName' a été créé avec succès."
} else {
    Write-Host "Le groupe '$GroupName' existe déjà."
}
```

Résultat : Le groupe a bien été créé

```
PS C:\Users\Administrateur> $GroupName = "WindowsServer"

if (-not (Get-ADGroup -Filter {Name -eq $GroupName})) {
    New-ADGroup -Name $GroupName -Path "OU=B1,DC=REMOTEWORKS,DC=LOCAL" -GroupScope Global
    Write-Host "Le groupe '$GroupName' a été créé avec succès."
} else {
    Write-Host "Le groupe '$GroupName' existe déjà."
}

Le groupe 'WindowsServer' a été créé avec succès.
```

Ajouter un utilisateur au groupe

Script

```
# Définir la variable pour le nom de l'utilisateur
$UserName = "Albert"

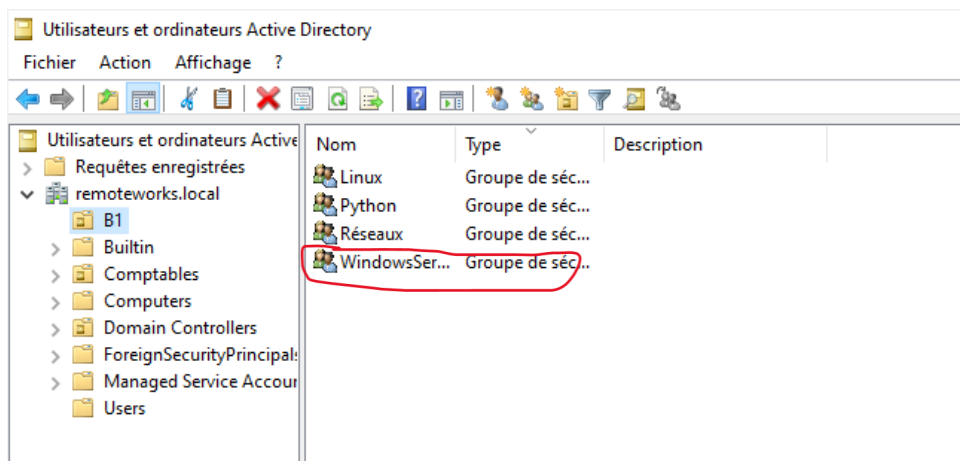
if (-not (Get-AdGroupMember -Identity WindowsServer | Where-Object {$_.SamAccountName -eq $UserName})) {
    Add-AdGroupMember -Identity WindowsServer -Members $UserName
    Write-Host "L'utilisateur '$UserName' a été ajouté au groupe WindowsServer."
} else {
    Write-Host "L'utilisateur '$UserName' est déjà membre du groupe WindowsServer."
}
```

Résultat : L'utilisateur a été ajouté au groupe

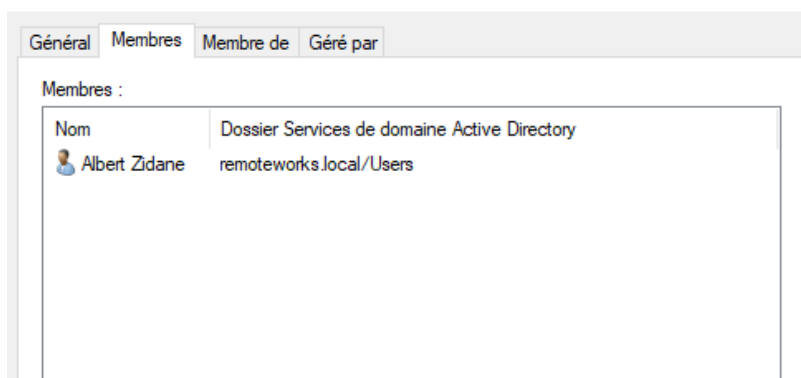
```
PS C:\Users\Administrateur> # Définir la variable pour le nom de l'utilisateur
$UserName = "Albert"

if (-not (Get-AdGroupMember -Identity WindowsServer | Where-Object {$_.SamAccountName -eq $UserName})) {
    Add-AdGroupMember -Identity Linux -Members $UserName
    Write-Host "L'utilisateur '$UserName' a été ajouté au groupe WindowsServer."
} else {
    Write-Host "L'utilisateur '$UserName' est déjà membre du groupe WindowsServer."
}

L'utilisateur 'Albert' a été ajouté au groupe WindowsServer.
```



Propriétés de : WindowsServer



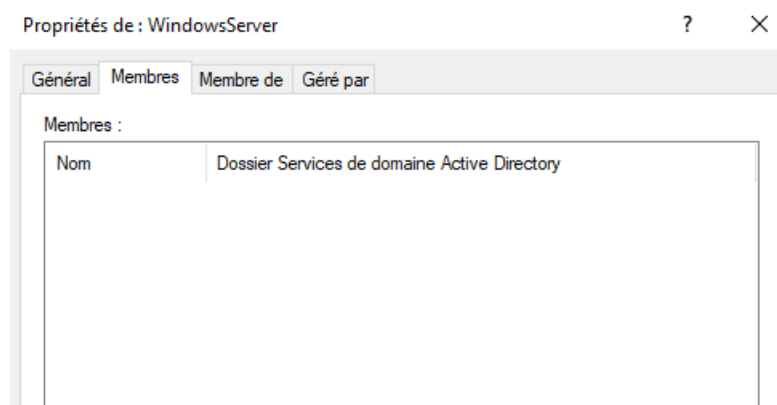
Retirer l'utilisateur du groupe

Script

```
# Définir la variable pour le nom de l'utilisateur
$username = "Albert"

if (Get-AdGroupMember -Identity WindowsServer | Where-Object {$_.SamAccountName -eq $username}) {
    Remove-AdGroupMember -Identity WindowsServer -Members $username -Confirm:$false
    Write-Host "L'utilisateur '$username' a été supprimé du groupe WindowsServer."
} else {
    Write-Host "L'utilisateur '$username' n'est pas membre du groupe WindowsServer."
}
```

Résultat : L'utilisateur a été supprimé du groupe



```
PS C:\Users\Administrateur> # Définir la variable pour le nom de l'utilisateur
$username = "Albert"

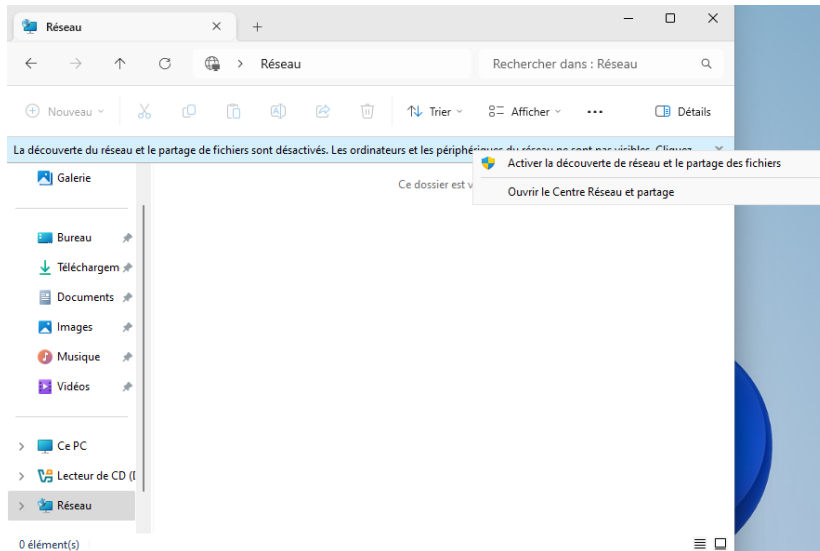
if (Get-AdGroupMember -Identity WindowsServer | Where-Object {$_.SamAccountName -eq $username}) {
    Remove-AdGroupMember -Identity WindowsServer -Members $username -Confirm:$false
    Write-Host "L'utilisateur '$username' a été supprimé du groupe WindowsServer."
} else {
    Write-Host "L'utilisateur '$username' n'est pas membre du groupe WindowsServer."
}

L'utilisateur 'Albert' a été supprimé du groupe WindowsServer.
```

Création d'un partage de données et le supprimer

1^{er} Création du partage de données

Avant tout activer sur les utilisateurs et les AD le partage de données pour avoir accès à AD1



Script complet :

On peut modifier le path pour modifier les dossiers compris dans le partage.

```
$shareName = "Windows"
$sharePath = "C:\Partage"

# Vérifier si le dossier existe, sinon le créer
if (-Not (Test-Path $sharePath)) {
    New-Item -ItemType Directory -Path $sharePath -Force | Out-Null
    Write-Host "Le dossier '$sharePath' a été créé."
}

# Vérifier si le partage existe déjà
$existingShare = Get-SmbShare -Name $shareName -ErrorAction SilentlyContinue

if ($existingShare -eq $null) {
    # Créer le partage s'il n'existe pas
    New-SmbShare -Name $shareName -Path $sharePath -FullAccess "Tout le monde"
    Write-Host "Le partage '$shareName' a été créé avec succès."
} else {
    Write-Host "Le partage '$shareName' existe déjà."
}
```

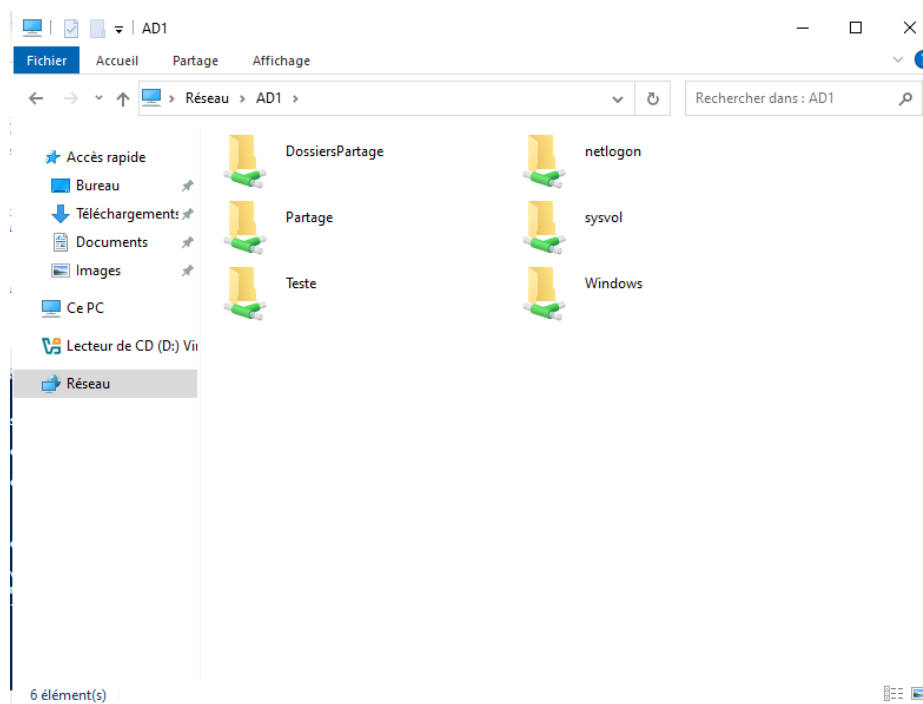
Résultat : Le partage de dossiers Windows a été créé.

```
$existingShare = Get-SmbShare -Name $shareName -ErrorAction SilentlyContinue

if ($existingShare -eq $null) {
    # Créer le partage s'il n'existe pas
    New-SmbShare -Name $shareName -Path $sharePath -FullAccess "Tout le monde"
    Write-Host "Le partage '$shareName' a été créé avec succès."
} else {
    Write-Host "Le partage '$shareName' existe déjà."
}

Le dossier 'C:\Partage' a été créé.
Le partage 'Windows' a été créé avec succès.
Name      ScopeName Path      Description
----      -
Windows *   C:\Partage
```

PS C:\Users\Administrateur>



2^{ème} Supprimer un partage de données

Attention avant de supprimer un partage donné, vous devez être sûr de ce que vous voulez faire, car le script ne demande pas de confirmation. Il supprime directement le partage de données.

Script complet :

```
$shareName = "windows"
$sharePath = "C:\Partage"

# Vérifier si le partage existe
$existingShare = Get-SmbShare -Name $shareName -ErrorAction SilentlyContinue

if ($existingShare -ne $null) {
    # Supprimer le partage
    Remove-SmbShare -Name $shareName -Force
    Write-Host "Le partage '$shareName' a été supprimé avec succès."
} else {
    Write-Host "Le partage '$shareName' n'existe pas."
}

# Vérifier si le dossier existe et le supprimer
if (Test-Path $sharePath) {
    Remove-Item -Path $sharePath -Recurse -Force
    Write-Host "Le dossier '$sharePath' a été supprimé."
} else {
    Write-Host "Le dossier '$sharePath' n'existe pas."
}
```

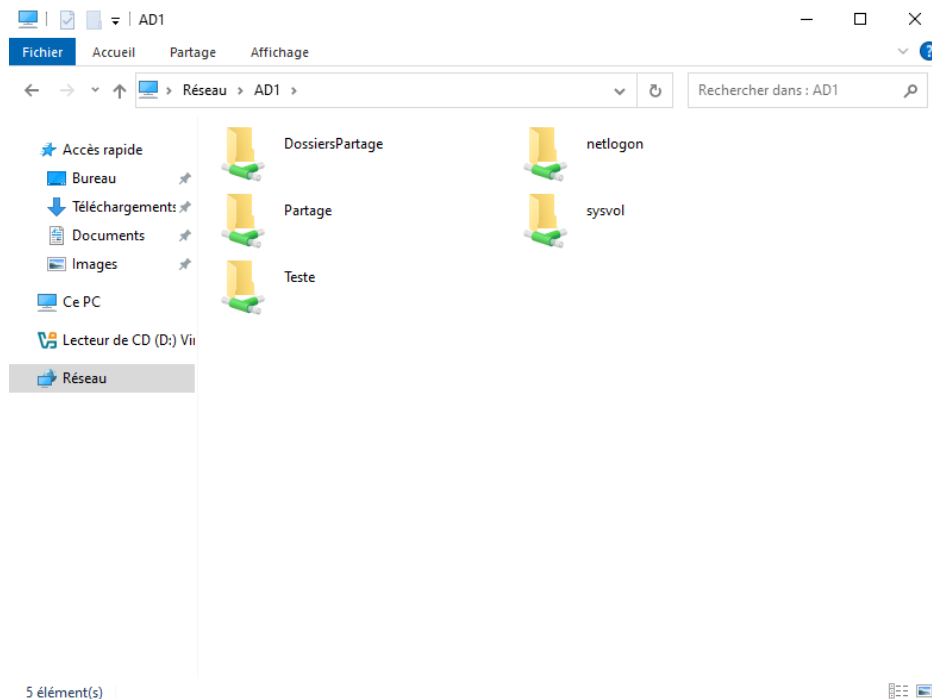
Résultat : Le partage de données Windows a été supprimé.

```
if ($existingShare -ne $null) {
    # Supprimer le partage
    Remove-SmbShare -Name $shareName -Force
    Write-Host "Le partage '$shareName' a été supprimé avec succès."
} else {
    Write-Host "Le partage '$shareName' n'existe pas."
}

# Vérifier si le dossier existe et le supprimer
if (Test-Path $sharePath) {
    Remove-Item -Path $sharePath -Recurse -Force
    Write-Host "Le dossier '$sharePath' a été supprimé."
} else {
    Write-Host "Le dossier '$sharePath' n'existe pas."
}

Le partage 'Windows' a été supprimé avec succès.
Le dossier 'C:\Partage' a été supprimé.

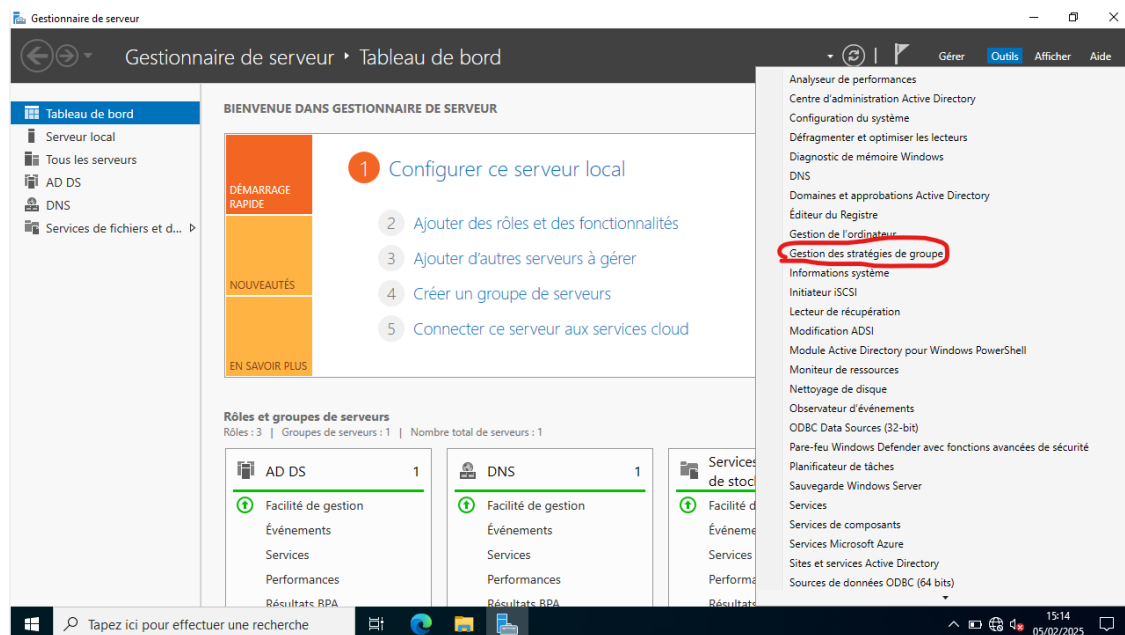
PS C:\Users\Administrateur> |
```



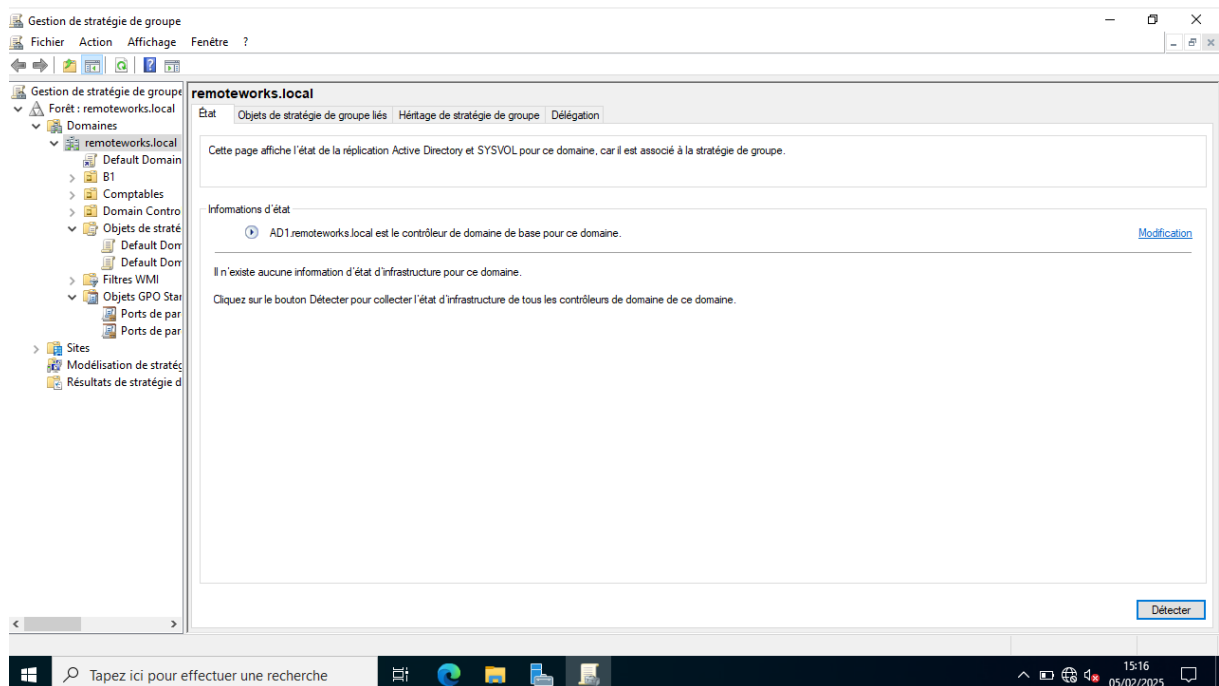
Création d'une GPO bloquant l'accès au CMD pour les utilisateurs du domaine et la sauvegarder.

1^{er} Création d'une GPO

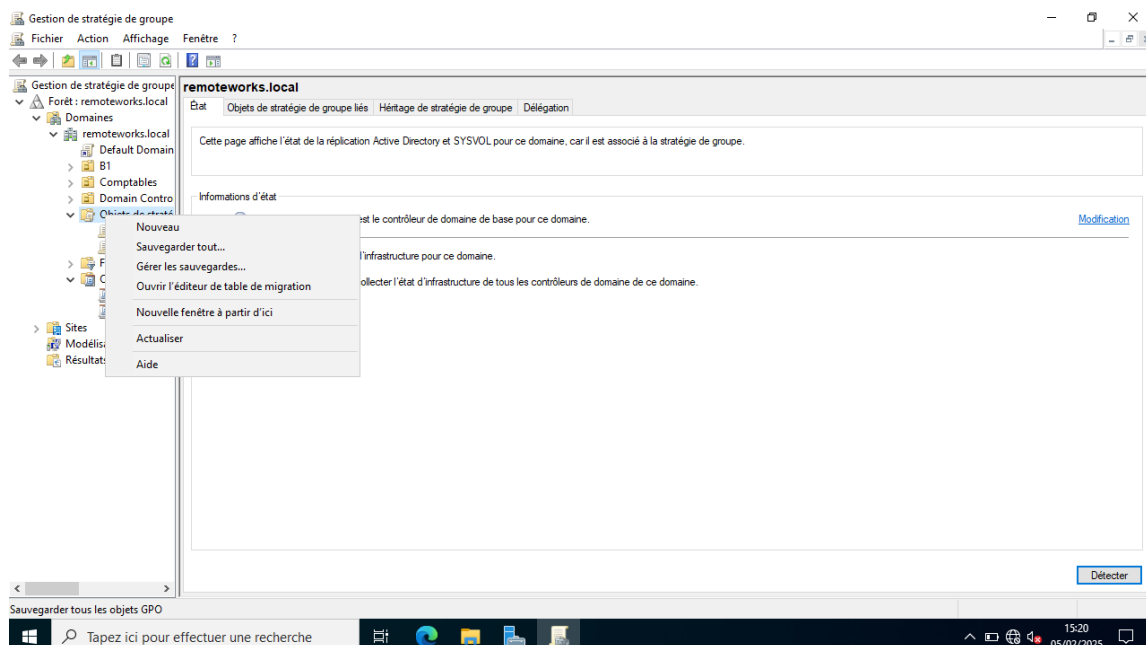
Dans gestionnaire de serveur, aller dans outil et sélectionner gestion de stratégies de groupes.



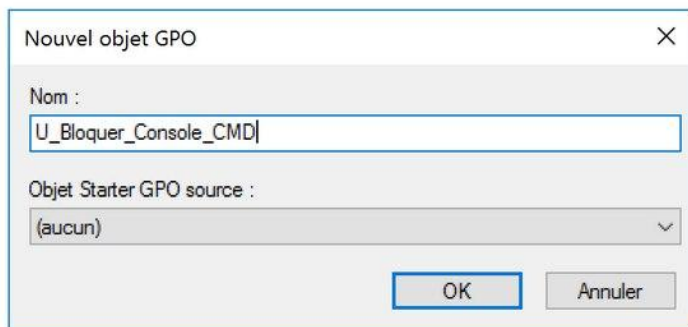
Une fois la fenêtre ouverte dans le domaine, défiler les documents comme la capture ci-dessous pour que sa soit plus simple pour naviguer entre eux.



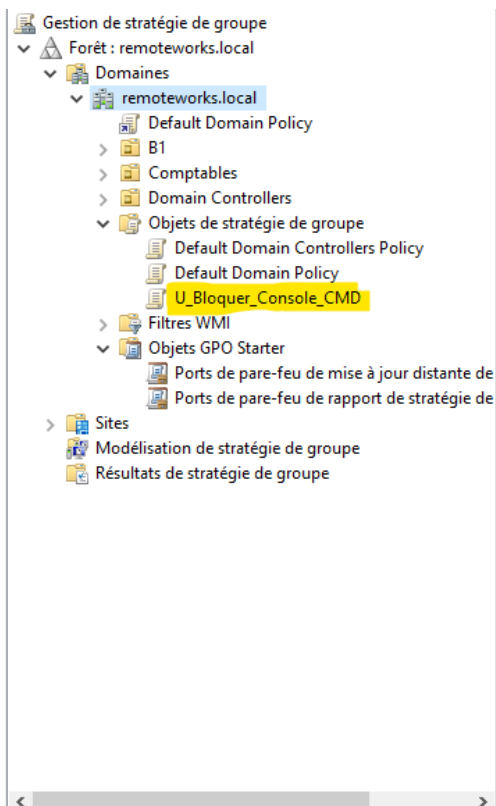
Sur "Objets de stratégie de groupe", effectuez un clic droit et cliquez sur "Nouveau"



Indiquez un nom pour cette GPO, par exemple "*U_Bloquer_Console_CMD*" mais vous pouvez mettre ce que vous voulez. Le "U" étant là en préfixe pour indiquer qu'il s'agit d'une GPO qui va agir au niveau Utilisateur. Cliquez sur "OK" pour valider



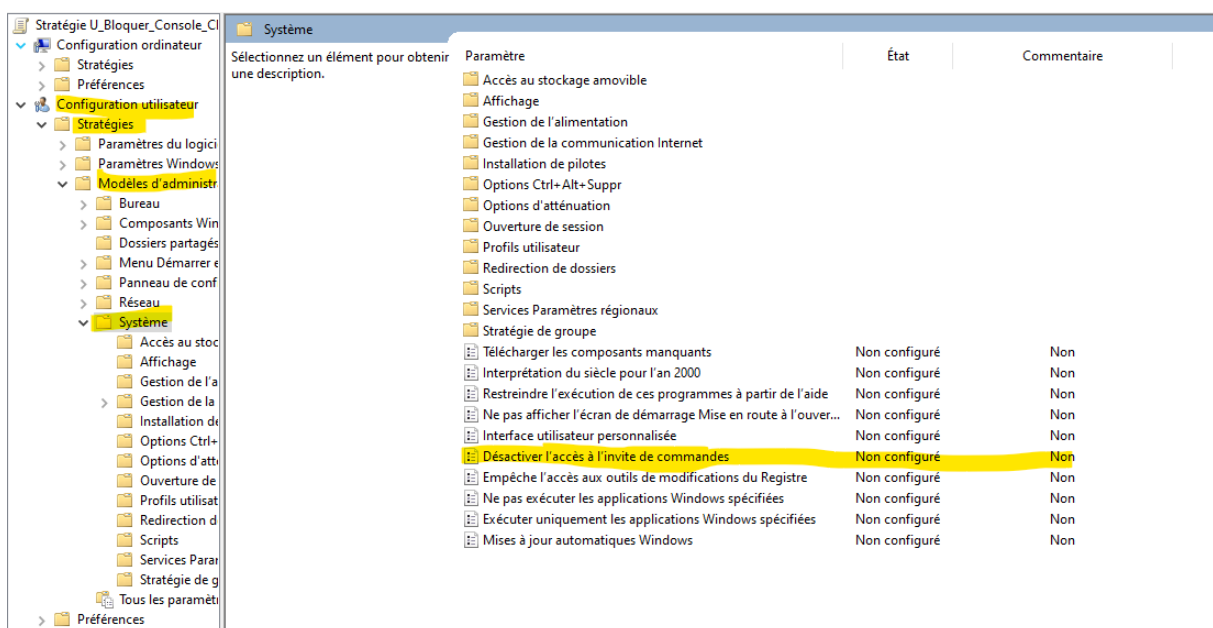
La GPO va s'afficher dans la liste, effectuez un clic droit dessus pour "*Modifier*".



Une fenêtre "Éditeur de gestion des stratégies de groupe" va s'ouvrir, cela permet de configurer la GPO. C'est ici que l'on va activer ou configurer certains paramètres à appliquer sur les utilisateurs (ou les ordinateurs).

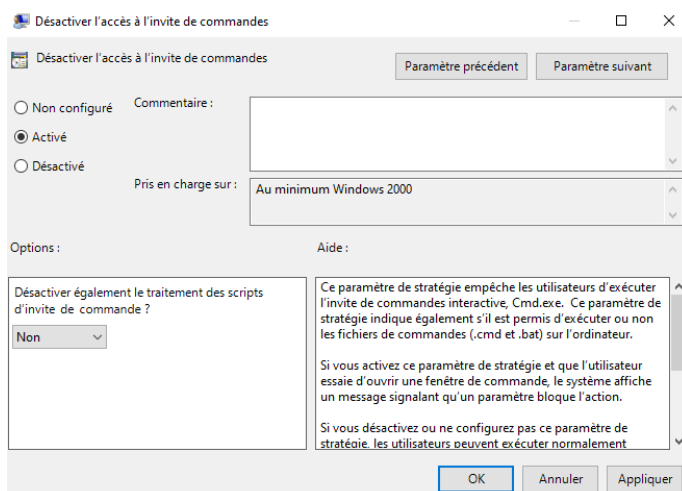
L'objectif maintenant va être de trouver le paramètre qui permet de désactiver l'accès à l'invite de commandes.

Voici le chemin vers notre fameux paramètre : Configuration utilisateur > Stratégies > Modèles d'administration > Système > Désactiver l'accès à l'invite de commandes. Double-cliquez dessus.

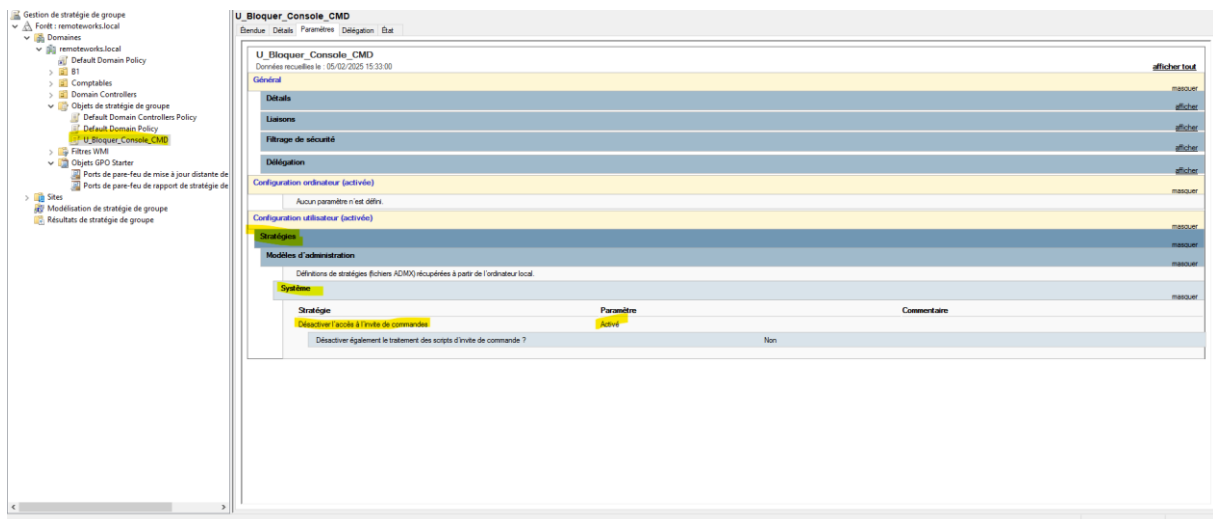


Sélectionner Activer comme la capture ci-dessous

Vous pouvez fermer ensuite la console de modification de cette GPO

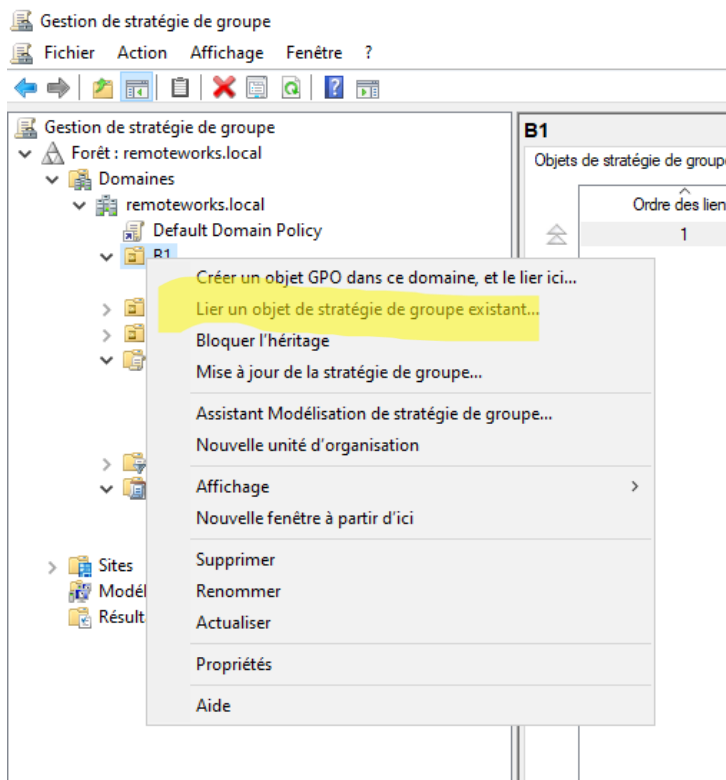


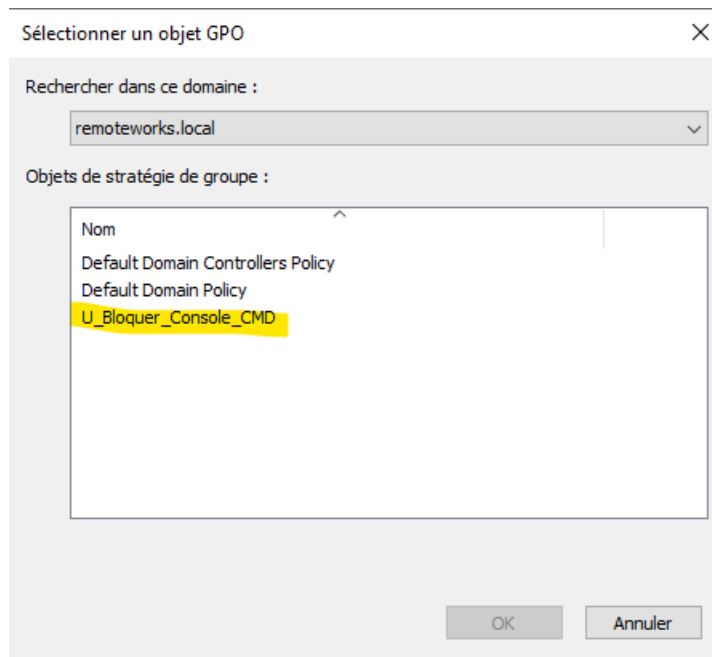
Dans stratégie de groupes sélectionner note GPO et aller dans paramètres > configuration utilisateur > Stratégies > Système



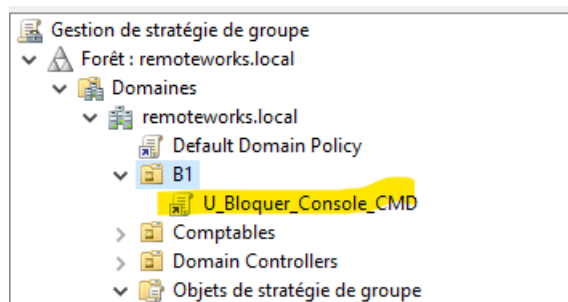
Maintenant on doit créer une liaison entre notre GPO et une OU spécifique ou alors directement appliquer au domaine entier mais sa bloquera aussi le CMD pour les administrateurs ce qui n'est pas top.

Pour créer une liaison entre une GPO et une unité d'organisation, effectuez un clic droit sur l'OU et cliquez sur "Lier un objet de stratégie de groupe existant".





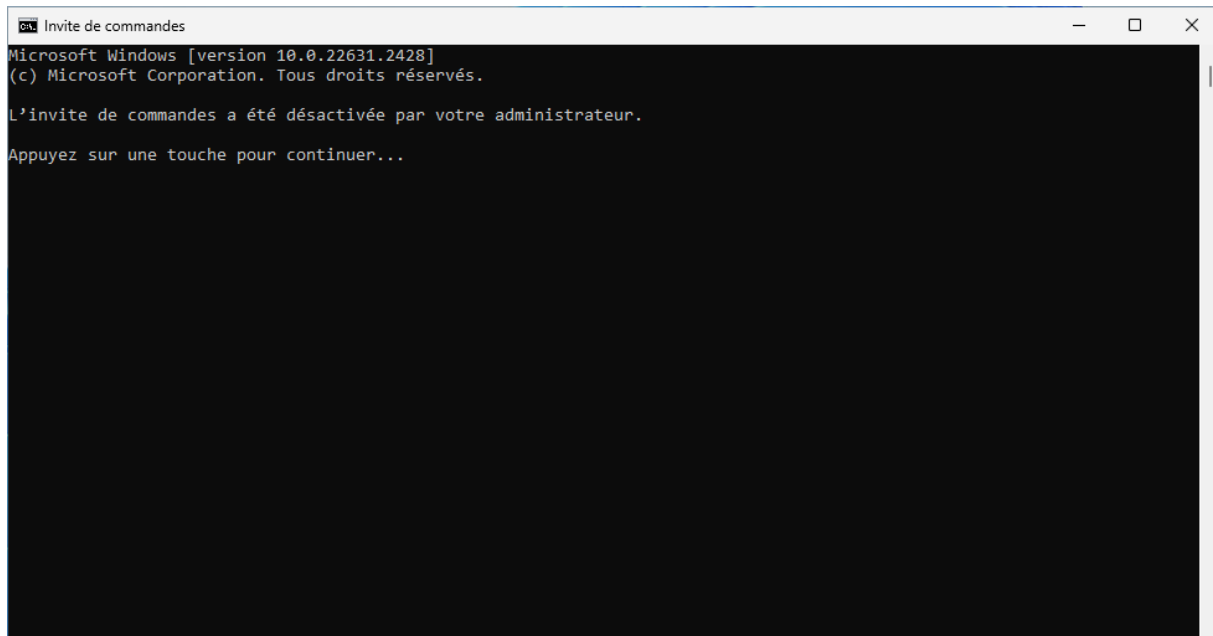
La liaison étant créée on remarque qu'un raccourci s'est ajouté juste sous l'OU "B1".



Maintenant on peut tester la GPO en se connectant sur notre ordinateur lié à notre domaine avec un compte utilisateur faisant partie de l'OU B1

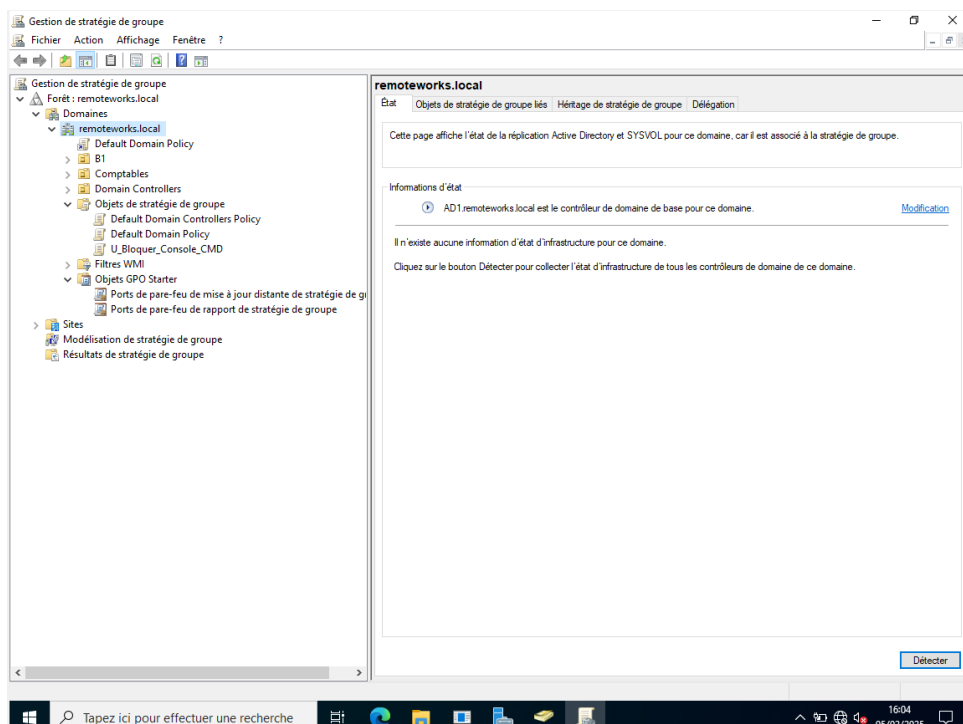
Dans le cas où la session existe déjà, il se peut que la GPO ne s'applique pas immédiatement. Il existe un temps de rafraîchissement pour les stratégies de groupe, ce qui est d'autant plus vrai pour les stratégies d'ordinateurs qui s'appliquent généralement au démarrage de la machine.

Pour forcer l'actualisation des stratégies de groupe sur un poste, que ce soit pour les paramètres ordinateurs ou utilisateurs, il y a une commande magique et qu'il est indispensable de connaître : **gpupdate /force**.

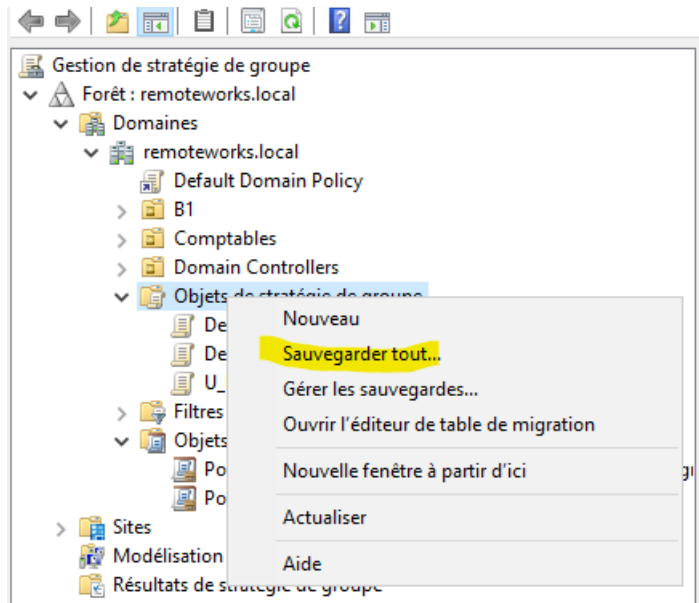


2^{ème} Sauvegarder des GPO

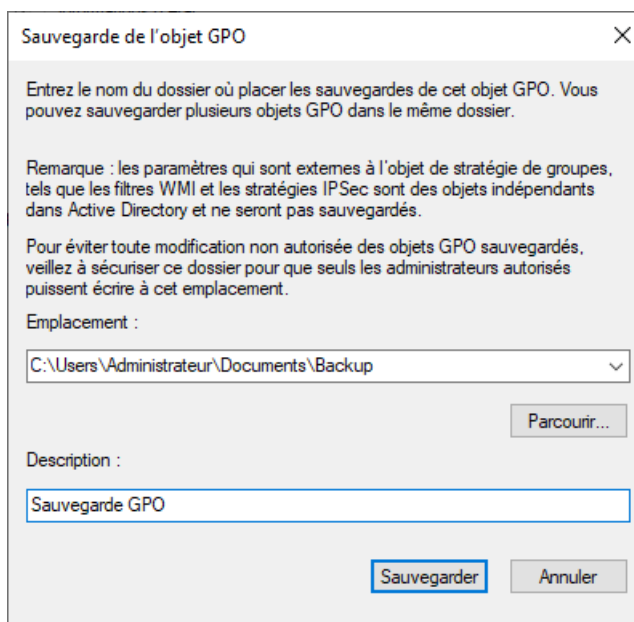
Retourner des gestions des stratégies de groupes comme la capture ci-dessous.



Après on fait un clic droit sur Objet de stratégie de groupes et on sélectionne sauvegarder tout



Après on choisit l'emplacement de nos sauvegardes.



Toute les GPO ont bien été sauvegarder.

