

TP PowerShell : Automatisation de la configuration d'un domaine Active Directory

Je dois automatiser entièrement la configuration d'un domaine Active Directory que j'ai déjà configuré manuellement. Pour cela, je vais créer un script PowerShell complet et commenté. Ce script devra :

- définir une adresse IP statique,
- renommer l'ordinateur,
- installer les rôles nécessaires (AD DS, DNS),
- promouvoir le serveur en contrôleur de domaine d'une nouvelle forêt,
- créer des unités organisationnelles (OU),
- créer des utilisateurs et des groupes,
- créer un dossier partagé,
- configurer des stratégies de groupe (GPO),
- configurer ou désactiver le pare-feu Windows pour faciliter les tests.

Je dois aussi rendre mon script **réutilisable** en utilisant des variables à la place de valeurs codées en dur. En bonus, je pourrai ajouter une fonctionnalité qui crée automatiquement des utilisateurs à partir d'un fichier CSV.

I. Recherche des cmdlets PowerShell utiles

Pour commencer, je recherche toutes les commandes PowerShell disponibles dans le module ActiveDirectory avec la commande :

```
Get-Command -Module ActiveDirectory
```

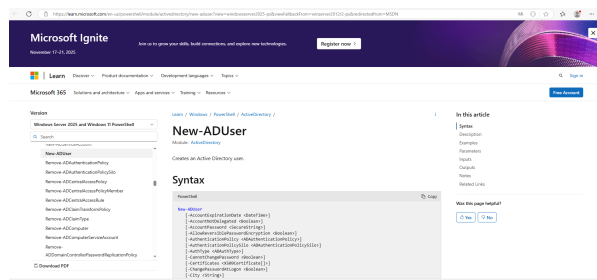
```
PS C:\Users\Administrateur.SRV-AD-TSSR-B1> Get-Command -Module ActiveDirectory
CommandType      Name                                     Version      Source
-----
Cmdlet           Add-ADCentralAccessPolicyMember        1.0.1.0      ActiveDirectory
Cmdlet           Add-ADComputerServiceAccount           1.0.1.0      ActiveDirectory
Cmdlet           Add-ADDomainControllerPasswordReplicationPolicy 1.0.1.0      ActiveDirectory
Cmdlet           Add-ADFineGrainedPasswordPolicySubject 1.0.1.0      ActiveDirectory
Cmdlet           Add-ADGroupMember                      1.0.1.0      ActiveDirectory
Cmdlet           Add-ADPrincipalGroupMembership         1.0.1.0      ActiveDirectory
Cmdlet           Add-ADResourcePropertyListMember       1.0.1.0      ActiveDirectory
Cmdlet           Clear-ADAccountExpiration               1.0.1.0      ActiveDirectory
Cmdlet           Clear-ADClaimTransfomLink              1.0.1.0      ActiveDirectory
Cmdlet           Disable-ADAccount                      1.0.1.0      ActiveDirectory
Cmdlet           Disable-ADOptionalFeature              1.0.1.0      ActiveDirectory
Cmdlet           Enable-ADAccount                       1.0.1.0      ActiveDirectory
Cmdlet           Enable-ADOptionalFeature                1.0.1.0      ActiveDirectory
Cmdlet           Get-ADAccountAuthorizationGroup         1.0.1.0      ActiveDirectory
Cmdlet           Get-ADAccountResultantPasswordReplicationPolicy 1.0.1.0      ActiveDirectory
Cmdlet           Get-ADAuthenticationPolicy             1.0.1.0      ActiveDirectory
Cmdlet           Get-ADAuthenticationPolicySilo         1.0.1.0      ActiveDirectory
Cmdlet           Get-ADCentralAccessPolicy              1.0.1.0      ActiveDirectory
Cmdlet           Get-ADCentralAccessRule                1.0.1.0      ActiveDirectory
Cmdlet           Get-ADClaimTransfomPolicy              1.0.1.0      ActiveDirectory
Cmdlet           Get-ADClaimType                        1.0.1.0      ActiveDirectory
Cmdlet           Get-ADComputer                         1.0.1.0      ActiveDirectory
Cmdlet           Get-ADComputerServiceAccount            1.0.1.0      ActiveDirectory
Cmdlet           Get-ADCloningExcludedApplicationList   1.0.1.0      ActiveDirectory
Cmdlet           Get-ADDefaultDomainPasswordPolicy      1.0.1.0      ActiveDirectory
Cmdlet           Get-ADDomain                           1.0.1.0      ActiveDirectory
Cmdlet           Get-ADDomainController                 1.0.1.0      ActiveDirectory
Cmdlet           Get-ADDomainControllerPasswordReplicationPolicy 1.0.1.0      ActiveDirectory
Cmdlet           Get-ADDomainControllerPasswordReplicationPolicy... 1.0.1.0      ActiveDirectory
Cmdlet           Get-ADFineGrainedPasswordPolicy        1.0.1.0      ActiveDirectory
Cmdlet           Get-ADFineGrainedPasswordPolicySubject 1.0.1.0      ActiveDirectory
Cmdlet           Get-ADForest                           1.0.1.0      ActiveDirectory
Cmdlet           Get-ADGroup                            1.0.1.0      ActiveDirectory
```

Cela me permettra de lister les cmdlets que je pourrai utiliser, comme New-ADUser, New-ADOrganizationalUnit, Install-WindowsFeature, etc.

Pour avoir de l'aide sur une commande spécifique, par exemple New-ADUser, je peux afficher la documentation complète avec :

```
Get-Help New-ADUser -Online
```

Cela me renvoie sur un page internet appropriée à mon besoin :



II. Étapes détaillées du script PowerShell

Définir une adresse IP statique

Je commence par définir l'adresse IP statique sur la carte réseau. Je dois connaître le nom de ma carte réseau, que je peux récupérer avec :

Get-NetAdapter

```
PS C:\Users\Administrator> netsh interface ipv4 show address
```

Name	InterfaceDescription	IfIndex	Status	MacAddress	LinkSpeed
Ethernet	Intel(R) PRO/1000 MT Desktop Adapter	6	Up	88-00-27-3F-FB-5F	1 Gbps

```
PS C:\Users\Administrator>
```

Dans mon cas le nom de ma carte réseau est Ethernet

Ensuite, j'utilise `New-NetIPAddress` ou `Set-NetIPAddress` pour configurer l'IP, le masque et la passerelle.

Exemple :

```
$InterfaceAlias = "Ethernet" # Adapté à ma configuration
$IPAddress = "192.168.1.10"
$PrefixLength = 24 # équivalent masque 255.255.255.0
$DefaultGateway = "192.168.1.1"
$DNSServer = "192.168.1.1"

# Suppression de toute IP existante sur l'interface
Get-NetIPAddress -InterfaceAlias $InterfaceAlias | Remove-NetIPAddress -
Confirm:$false

# Ajout de la nouvelle IP statique
New-NetIPAddress -InterfaceAlias $InterfaceAlias -IPAddress $IPAddress -
PrefixLength $PrefixLength -DefaultGateway $DefaultGateway

# Configuration du serveur DNS
Set-DnsClientServerAddress -InterfaceAlias $InterfaceAlias -ServerAddresses
$DNSServer
```

J'ai défini les paramètres réseau adaptés à ma configuration, comme le nom de ma carte réseau, mon adresse IP statique, le masque de sous-réseau, la passerelle par défaut et le serveur DNS.

Ensuite, j'ai supprimé toutes les adresses IP déjà configurées sur cette interface réseau pour éviter les conflits.

Puis, j'ai ajouté une nouvelle adresse IP statique avec les paramètres que j'ai définis, en précisant aussi la passerelle par défaut.

Enfin, j'ai configuré le serveur DNS pour que mon ordinateur sache où envoyer les requêtes de noms de domaine.

Renommer l'ordinateur

Pour renommer l'ordinateur, j'utilise la cmdlet Rename-Computer :

```
$NewComputerName = "DC02"  
Rename-Computer -NewName $NewComputerName -Restart
```

Attention : Le redémarrage sera nécessaire pour appliquer le nouveau nom.

Installer les rôles AD DS et DNS

J'installe les rôles nécessaires avec Install-WindowsFeature :

```
Install-WindowsFeature -Name AD-Domain-Services, DNS -IncludeManagementTools
```

Promouvoir le serveur en contrôleur de domaine

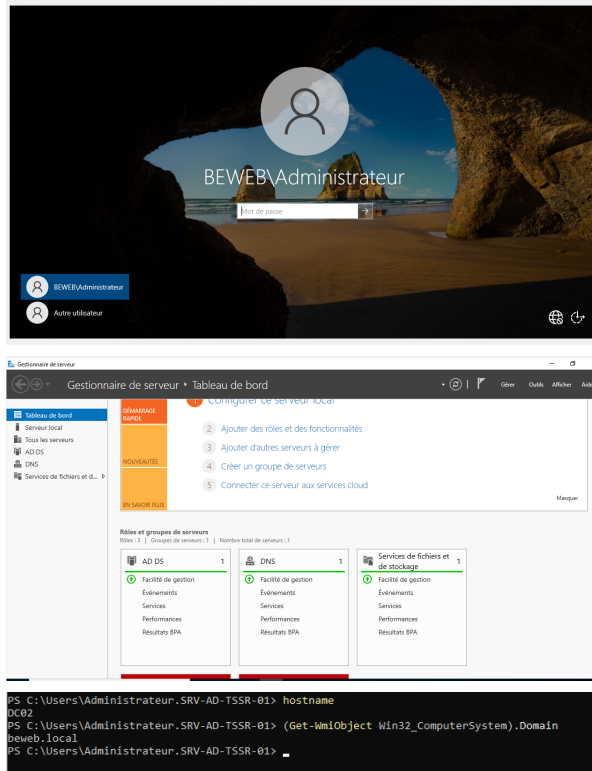
J'utilise la cmdlet Install-ADDSForest pour créer une nouvelle forêt :

```
$DomainName = "beweb.local"  
$SafeModePassword = Read-Host -Prompt "Entrez le mot de passe DSRM" -  
AsSecureString  
  
Install-ADDSForest -DomainName $DomainName -SafeModeAdministratorPassword  
$SafeModePassword -InstallDNS -Force -NoRebootOnCompletion
```

Remarque : la ligne \$SafeModePassword cache la saisie du mot de passe : quand je tape, rien ne s'affiche à l'écran (ni caractères ni astérisques). C'est la meilleure méthode native pour garder le mot de passe secret pendant la saisie à mon niveau.

À la fin, je redémarre la machine pour appliquer les changements :

```
Restart-Computer
```



Créer des Unités Organisationnelles (OU)

Je peux créer des OU avec New-ADOrganizationalUnit :

```
# Définition de mon domaine
$DomainName = "beweb.local"

# Découper le nom de domaine en parties et créer un tableau DC=...
$dcParts = foreach ($part in $DomainName -split '\.') { "DC=$part" }

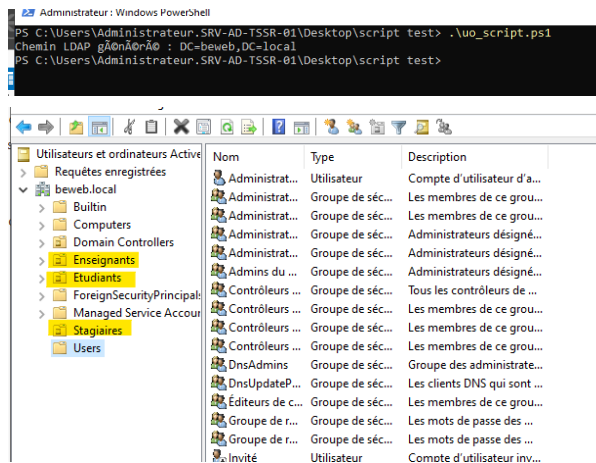
# Joindre le tout avec une virgule pour obtenir le chemin LDAP
$LDAPPath = $dcParts -join ','

# Afficher pour vérifier
Write-Host "Chemin LDAP généré : $LDAPPath"

# Création des OU avec le chemin LDAP généré
New-ADOrganizationalUnit -Name "Enseignants" -Path $LDAPPath
New-ADOrganizationalUnit -Name "Etudiants" -Path $LDAPPath
New-ADOrganizationalUnit -Name "Stagiaires" -Path $LDAPPath
```

Je définis mon domaine dans \$DomainName. Ensuite, je crée automatiquement le chemin LDAP (\$LDAPPath) qui correspond à mon domaine, pour ne pas avoir à l'écrire en dur plusieurs fois. Puis, je crée mes unités organisationnelles (OU) "Enseignants", "Etudiants" et "Stagiaires" dans ce domaine, en utilisant \$LDAPPath.

Comme ça, si je change de domaine plus tard, je modifie juste \$DomainName en haut, et tout le script s'adapte automatiquement.



Créer des utilisateurs et des groupes

```
# Domaine racine
$domainDN = "DC=beweb,DC=local"

# Listes des OU à créer
$ouList = @("Utilisateurs", "Groupes")

# Fonction pour vérifier ou créer une OU
function Ensure-OU {
    param (
        [string]$OUName
    )
    $ouPath = "OU=$OUName,$domainDN"
    $ouExists = Get-ADOrganizationalUnit -Filter "DistinguishedName -eq '$ouPath'"
    -ErrorAction SilentlyContinue

    if (-not $ouExists) {
        Write-Host "Création de l'OU $OUName..."
        New-ADOrganizationalUnit -Name $OUName -Path $domainDN
    }
    else {
        Write-Host "OU $OUName existe déjà."
    }
    return $ouPath
}

# Créer les OU nécessaires
$ouPaths = @{}
foreach ($ou in $ouList) {
    $ouPaths[$ou] = Ensure-OU -OUName $ou
}

# Définition des utilisateurs à créer (tableau d'objets)
$users = @(
    @{
        Name = "Marie Dupont"
        GivenName = "Marie"
        Surname = "Dupont"
    }
)
```

```

        SamAccountName = "mdupont"
        UserPrincipalName = "mdupont@beweb.local"
        Password = "P@ssw0rd"
        OU = "Utilisateurs"
    },
    @{
        Name = "Jean Martin"
        GivenName = "Jean"
        Surname = "Martin"
        SamAccountName = "jmartin"
        UserPrincipalName = "jmartin@beweb.local"
        Password = "P@ssw0rd123"
        OU = "Utilisateurs"
    }
)

# Définition des groupes à créer (tableau d'objets)
$groups = @(
    @{
        Name = "Admins"
        GroupScope = "Global"
        OU = "Groupes"
    },
    @{
        Name = "UtilisateursNormaux"
        GroupScope = "Global"
        OU = "Groupes"
    }
)

# Création des utilisateurs
foreach ($user in $users) {
    $userOUPath = $ouPaths[$user.OU]
    Write-Host "Création de l'utilisateur $($user.Name)..."
    New-ADUser `
        -Name $user.Name `
        -GivenName $user.GivenName `
        -Surname $user.Surname `
        -SamAccountName $user.SamAccountName `
        -UserPrincipalName $user.UserPrincipalName `
        -AccountPassword (ConvertTo-SecureString $user.Password -AsPlainText -
Force) `
        -Enabled $true `
        -Path $userOUPath `
        -ChangePasswordAtLogon $true
}

# Création des groupes
foreach ($group in $groups) {
    $groupOUPath = $ouPaths[$group.OU]
    Write-Host "Création du groupe $($group.Name)..."
    New-ADGroup `
        -Name $group.Name `
        -GroupScope $group.GroupScope `

```

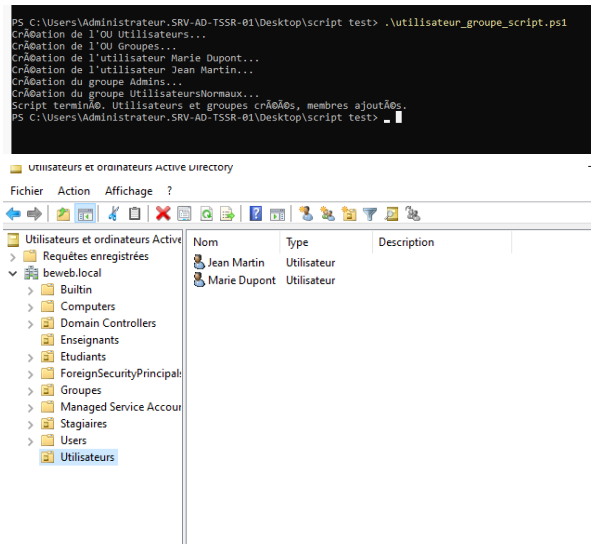
```

-Path $groupOUPath
}

# Association utilisateurs -> groupes (exemple simple)
# Ici on ajoute Marie Dupont et Jean Martin dans le groupe Admins pour l'exemple
Add-ADGroupMember -Identity "Admins" -Members @("mdupont", "jmartin")

Write-Host "Script terminé. Utilisateurs et groupes créés, membres ajoutés."

```



Créer un dossier partagé

Je crée un dossier et je le partage avec les permissions nécessaires :

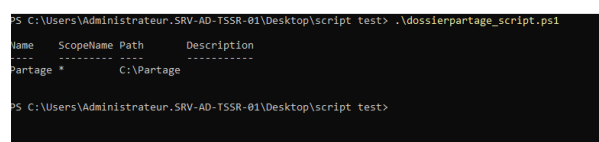
```

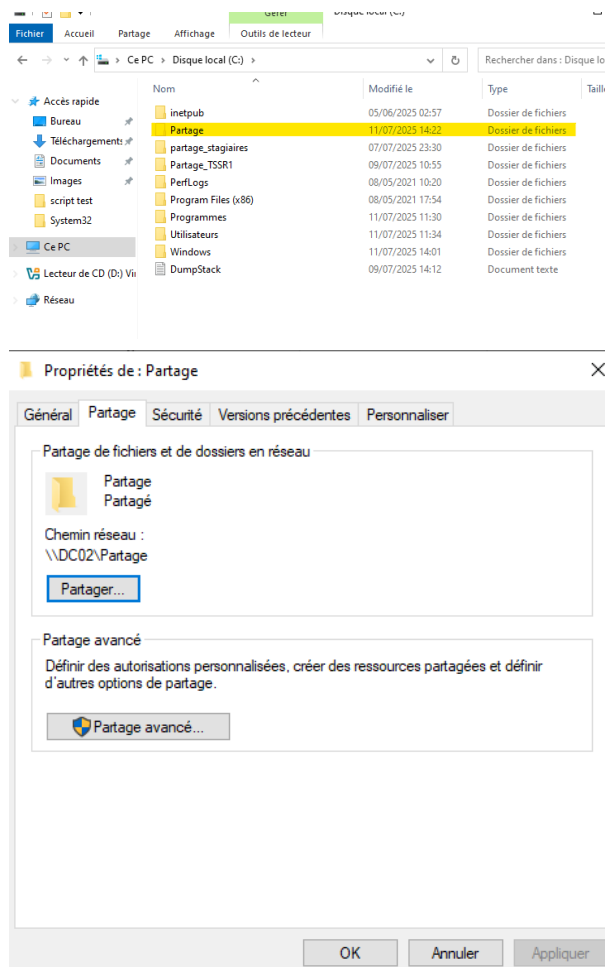
$FolderPath = "C:\Partage"

# Création du dossier si nécessaire
if (-Not (Test-Path $FolderPath)) {
    New-Item -ItemType Directory -Path $FolderPath
}

# Création du partage SMB avec droits d'accès pour les administrateurs locaux
New-SmbShare -Name "Partage" -Path $FolderPath -FullAccess
"BUILTIN\Administrateurs"

```





Configurer des GPO locales ou de domaine

Je peux configurer une GPO via les cmdlets New-GPO, Set-GPRegistryValue, New-GPLink. Exemple pour créer une GPO :

```
# Nom de la GPO et DN du domaine
$gpoName = "Bloquer USB"
$domainDN = "DC=beweb,DC=local"

# Essayer de récupérer la GPO existante
$Gpo = Get-GPO -Name $gpoName -ErrorAction SilentlyContinue

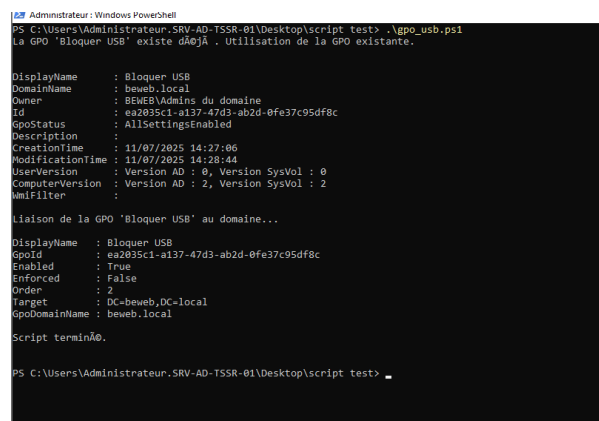
# Si elle n'existe pas, la créer
if (-not $Gpo) {
    Write-Host "Création de la GPO '$gpoName'..."
    $Gpo = New-GPO -Name $gpoName -Comment "Désactivation des périphériques USB
via la clé de registre USBSTOR"
} else {
    Write-Host "La GPO '$gpoName' existe déjà. Utilisation de la GPO existante."
}

# Modifier la valeur de registre dans la GPO pour bloquer les périphériques USB
Set-GPRegistryValue -Name $Gpo.DisplayName `
    -Key "HKLM\SYSTEM\CurrentControlSet\Services\USBSTOR" `
    -ValueName "Start" `
    -Type DWord `
```


-Value 4

```
# Lier la GPO au domaine si ce n'est pas déjà fait
$gpoLinks = Get-GPInheritance -Target $domainDN | Select-Object -ExpandProperty GpoLinks
if (-not ($gpoLinks | Where-Object { $_.DisplayName -eq $gpoName })) {
    Write-Host "Liaison de la GPO '$gpoName' au domaine..."
    New-GPLink -Name $Gpo.DisplayName -Target $domainDN
} else {
    Write-Host "La GPO '$gpoName' est déjà liée au domaine."
}

Write-Host "Script terminé."
```



```
PS C:\Users\Administrateur.SRV-AD-TSSR-01\Desktop\script test> .\gpo_usb.ps1
La GPO 'Bloquer USB' existe déjà. Utilisation de la GPO existante.

DisplayName : Bloquer USB
DomainName  : beweb.local
Owner       : BSEC\Admins du domaine
Id          : ea2035c1-a137-47d3-ab2d-0fe37c95df8c
GpoStatus   : AllSettingsEnabled
Description :
CreationTime : 11/07/2025 14:27:06
ModificationTime : 11/07/2025 14:28:44
UserVersion  : Version AD : 0, Version SysVol : 0
ComputerVersion : Version AD : 2, Version SysVol : 2
WmiFilter    :

Liaison de la GPO 'Bloquer USB' au domaine...
DisplayName : Bloquer USB
GpoId       : ea2035c1-a137-47d3-ab2d-0fe37c95df8c
Enabled     : True
Enforced    : False
Order       : 2
Target      : DC=beweb,DC=local
GpoDomainName : beweb.local

Script terminé.

PS C:\Users\Administrateur.SRV-AD-TSSR-01\Desktop\script test> _
```

Configurer ou désactiver le pare-feu Windows

Pour désactiver temporairement le pare-feu (utile pour les tests) :

```
param(
    [ValidateSet("Enable", "Disable")]
    [string]$Action
)

if ($Action -eq "Disable") {
    Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
    Write-Host "Firewall désactivé."
} elseif ($Action -eq "Enable") {
    Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True
    Write-Host "Firewall activé."
} else {
    Write-Host "Action non reconnue. Utilisez Enable ou Disable."
}
```

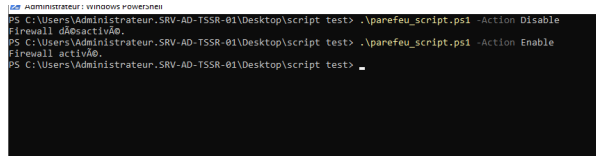
Je peux activer ou désactiver le pare-feu :

Pour désactiver le firewall :

```
.\parefeu_script.ps1 -Action Disable
```

Pour l'activer :

```
.\parefeu_script.ps1 -Action Enable
```



```
PS C:\Users\Administrateur.SRV-AD-TSSR-01\Desktop> script test> .\parefeu_script.ps1 -Action Disable
Firewall désactivée.
PS C:\Users\Administrateur.SRV-AD-TSSR-01\Desktop> script test> .\parefeu_script.ps1 -Action Enable
Firewall activée.
PS C:\Users\Administrateur.SRV-AD-TSSR-01\Desktop> script test> .
```

Rendre le script réutilisable

J'utilise des variables en début de script pour stocker les paramètres (nom du domaine, IP, nom ordinateur, chemins, mots de passe...) afin de faciliter la réutilisation sans modifier le script dans le corps.

Structure complète du script Setup-ADServer.ps1

Je l'adapte pour le différencier de mes tests précédents :

```
param(
    [Parameter(Mandatory = $true)]
    [ValidateSet("1", "2", "3", "4", "5", "6", "7")]
    [string]$Etape
)

# Variables globales communes
$InterfaceAlias = "Ethernet"
$IPAddress = "192.168.1.10"
$PrefixLength = 24
$DefaultGateway = "192.168.1.1"
$DNSServer = "192.168.1.1"
$NewComputerName = "DC03"
$DomainName = "beweb2.local"
$DomainDN = "DC=beweb2,DC=local"

switch ($Etape) {

    "1" {
        # Etape 1 : Configuration IP + Changement de nom + redémarrage
        Write-Host "Étape 1 : Configuration réseau et renommage de la machine."

        Get-NetIPAddress -InterfaceAlias $InterfaceAlias | Remove-NetIPAddress -
Confirm:$false

        New-NetIPAddress -InterfaceAlias $InterfaceAlias -IPAddress $IPAddress -
PrefixLength $PrefixLength -DefaultGateway $DefaultGateway
```

```

Set-DnsClientServerAddress -InterfaceAlias $InterfaceAlias -
ServerAddresses $DNSServer

    Rename-Computer -NewName $NewComputerName -Restart
}

"2" {
    # Etape 2 : Installation des rôles AD + DNS + Promotion du contrôleur
    Write-Host "Étape 2 : Installation AD-DS et DNS."

    Install-WindowsFeature -Name AD-Domain-Services, DNS -
IncludeManagementTools

    $SafeModePassword = Read-Host -Prompt "Entrez le mot de passe DSRM" -
AsSecureString

    Install-ADDSForest -DomainName $DomainName -SafeModeAdministratorPassword
$SafeModePassword -InstallDNS -Force -NoRebootOnCompletion

    Restart-Computer
}

"3" {
    # Etape 3 : Création des OU principales
    Write-Host "Étape 3 : Création des OU principales."

    $dcParts = foreach ($part in $DomainName -split '\.'){ "DC=$part" }
    $LDAPPath = $dcParts -join ','

    Write-Host "Chemin LDAP généré : $LDAPPath"

    New-ADOrganizationalUnit -Name "Enseignants" -Path $LDAPPath
    New-ADOrganizationalUnit -Name "Etudiants" -Path $LDAPPath
    New-ADOrganizationalUnit -Name "Stagiaires" -Path $LDAPPath
}

"4" {
    # Etape 4 : Création des utilisateurs et des groupes
    Write-Host "Étape 4 : Création des utilisateurs et groupes."

    function Ensure-OU {
        param([string]$Ouname)
        $ouPath = "OU=$Ouname,$DomainDN"
        if (-not (Get-ADOrganizationalUnit -Filter "DistinguishedName -eq
'$ouPath'" -ErrorAction SilentlyContinue)) {
            New-ADOrganizationalUnit -Name $Ouname -Path $DomainDN
        }
        return $ouPath
    }

    $ouPaths = @{
        "Utilisateurs" = Ensure-OU -Ouname "Utilisateurs"
        "Groupes" = Ensure-OU -Ouname "Groupes"
    }
}

```

```

    $users = @(
        @{Name="Marie Dupont"; GivenName="Marie"; Surname="Dupont";
        SamAccountName="mdupont"; UserPrincipalName="mdupont@beweb.local";
        Password="P@ssw0rd"; OU="Utilisateurs"},
        @{Name="Jean Martin"; GivenName="Jean"; Surname="Martin";
        SamAccountName="jmartin"; UserPrincipalName="jmartin@beweb.local";
        Password="P@ssw0rd123"; OU="Utilisateurs"}
    )

    $groups = @(
        @{Name="Admins"; GroupScope="Global"; OU="Groupes"},
        @{Name="UtilisateursNormaux"; GroupScope="Global"; OU="Groupes"}
    )

    foreach ($user in $users) {
        New-ADUser -Name $user.Name -GivenName $user.GivenName -Surname
        $user.Surname -SamAccountName $user.SamAccountName -UserPrincipalName
        $user.UserPrincipalName -AccountPassword (ConvertTo-SecureString $user.Password -
        AsPlainText -Force) -Enabled $true -Path $ouPaths[$user.OU] -ChangePasswordAtLogon
        $true
    }

    foreach ($group in $groups) {
        New-ADGroup -Name $group.Name -GroupScope $group.GroupScope -Path
        $ouPaths[$group.OU]
    }

    Add-ADGroupMember -Identity "Admins" -Members @("mdupont", "jmartin")
}

"5" {
    # Etape 5 : Création d'un partage SMB
    Write-Host "Étape 5 : Création d'un partage SMB."

    $FolderPath = "C:\Partage"
    if (-Not (Test-Path $FolderPath)) {
        New-Item -ItemType Directory -Path $FolderPath
    }

    New-SmbShare -Name "Partage" -Path $FolderPath -FullAccess
    "BUILTIN\Administrateurs"
}

"6" {
    # Etape 6 : Création et liaison d'une GPO
    Write-Host "Étape 6 : Configuration de la GPO 'Bloquer USB'."

    $gpoName = "Bloquer USB"

    $Gpo = Get-GPO -Name $gpoName -ErrorAction SilentlyContinue
    if (-not $Gpo) {
        $Gpo = New-GPO -Name $gpoName -Comment "Désactivation des
        périphériques USB"
    }
}

```

```

    }

    Set-GPRegistryValue -Name $Gpo.DisplayName -Key
    "HKLM\SYSTEM\CurrentControlSet\Services\USBSTOR" -ValueName "Start" -Type DWord -
    Value 4

    $gpoLinks = Get-GPInheritance -Target $DomainDN | Select-Object -
    ExpandProperty GpoLinks
    if (-not ($gpoLinks | Where-Object { $_.DisplayName -eq $gpoName })) {
        New-GPLink -Name $Gpo.DisplayName -Target $DomainDN
    }
}

"7" {
    # Etape 7 : Activer ou désactiver le pare-feu
    Write-Host "Étape 7 : Activation/désactivation du pare-feu."

    $Action = Read-Host "Entrez 'Enable' ou 'Disable' pour activer/désactiver
    le pare-feu"
    if ($Action -eq "Disable") {
        Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
        Write-Host "Firewall désactivé."
    } elseif ($Action -eq "Enable") {
        Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True
        Write-Host "Firewall activé."
    } else {
        Write-Host "Action non reconnue. Utilisez Enable ou Disable."
    }
}

Default {
    Write-Host "Étape non reconnue."
}
}

```

J'ai organisé le script en plusieurs étapes pour faciliter l'exécution et la gestion :

Certaines actions nécessitent un redémarrage (comme le renommage de l'ordinateur ou la promotion en contrôleur de domaine).

En divisant le script, je peux reprendre la suite sans tout refaire ni perdre mes configurations déjà appliquées.

Cela évite aussi les erreurs liées à l'exécution prématurée d'étapes dépendantes d'une précédente (ex. création d'OU après la promotion AD).

Enfin, ce découpage me permet de contrôler précisément l'avancement et d'intervenir manuellement si besoin.

Cette méthode est pratique et sécurise la procédure d'installation étape par étape.

UTILISATION :

Étape 1 (config IP + renommage) :

```
.\Setup-ADServer.ps1 -Etape 1
```

Étape 2 (installation AD) après redémarrage :

```
.\Setup-ADServer.ps1 -Etape 2
```

Étape 3 (création des OU) :

```
.\Setup-ADServer.ps1 -Etape 3
```

Et ainsi de suite.

```
PS C:\Users\Administrateur.DC02\Desktop> .\Setup-ADServer.ps1
Applet de commande Setup-ADServer.ps1 à la position 1 du pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
Etape:
```

Ainsi je choisis l'étape

Suite à l'étape 1 :

```
PS C:\Users\Administrateur.DC02> hostname
DC03
PS C:\Users\Administrateur.DC02> ipconfig /all

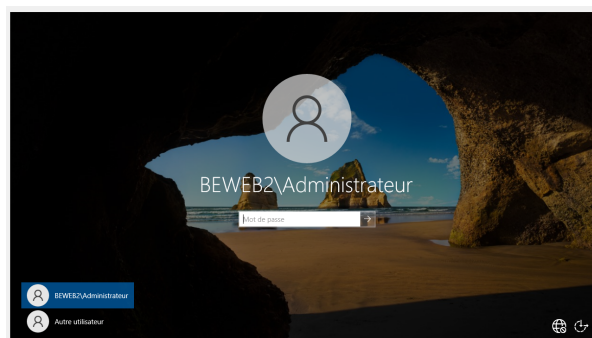
Configuration IP de Windows

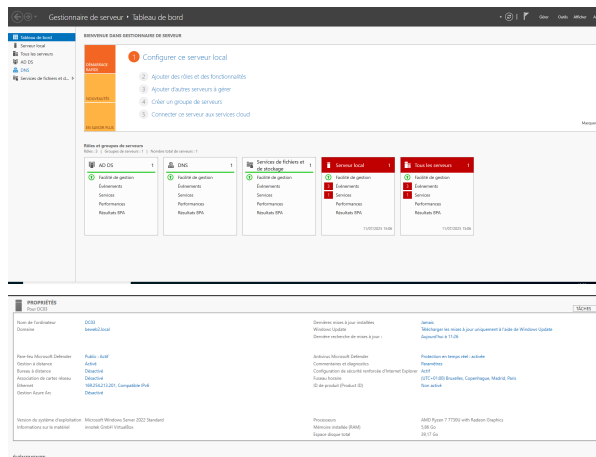
Nom de l'hôte . . . . . : DC03
Suffixe DNS principal . . . . . : beweb.local
Type de noeud . . . . . : Hybride
Routage IP activé . . . . . : Oui
Proxy WINS activé . . . . . : Non
Liste de recherche du suffixe DNS.: beweb.local

Carte Ethernet Ethernet :

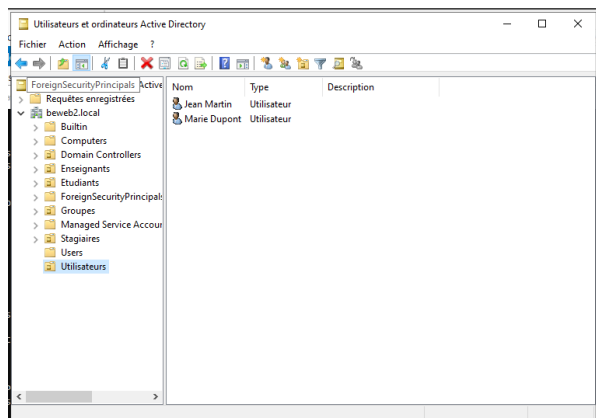
Suffixe DNS propre à la connexion. . . :
Description. . . . . : Intel(R) PRO/1000 MT Desktop Adapter
Adresse physique . . . . . : 08-00-27-33-F8-5F
DHCP activé. . . . . : Non
Configuration automatique activée. . . : Oui
Adresse IPv6. . . . . : fd17:625c:f037:2:a9e2:aad:721:8a93(préfére)
Adresse IPv6 de liaison locale. . . . : fe80::66cd:8d0:d3e8:5de5%6(préfére)
Adresse d'autoconfiguration IPv4 . . . : 169.254.213.201(préfére)
Masque de sous-réseau. . . . . : 255.255.0.0
Passerelle par défaut. . . . . : fe80::2%6
192.168.1.1
IAID DHCPv6 . . . . . : 101187623
DUID de client DHCPv6. . . . . : 00-01-00-01-2F-FD-CD-A4-08-00-27-33-F8-5F
Serveurs DNS. . . . . : ::1
192.168.1.1
NetBIOS sur Tcpip. . . . . : Activé
PS C:\Users\Administrateur.DC02>
```

Suite à l'étape 2 :

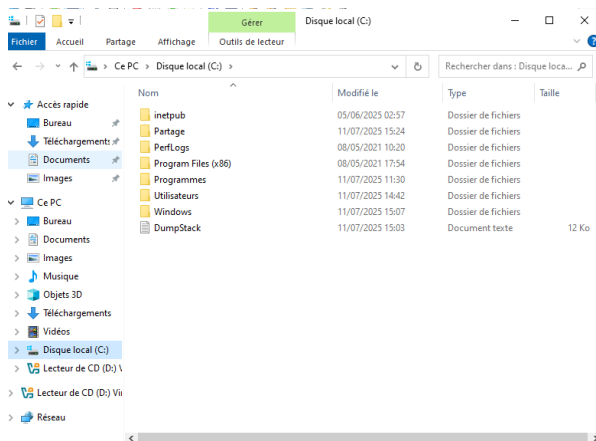




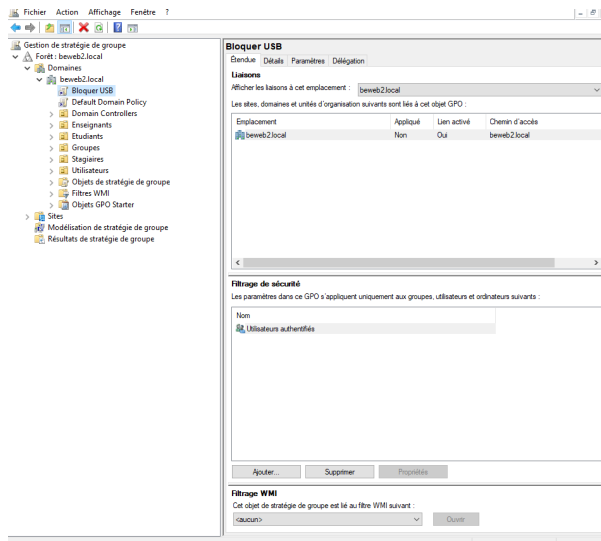
Suite à l'étape 3 :



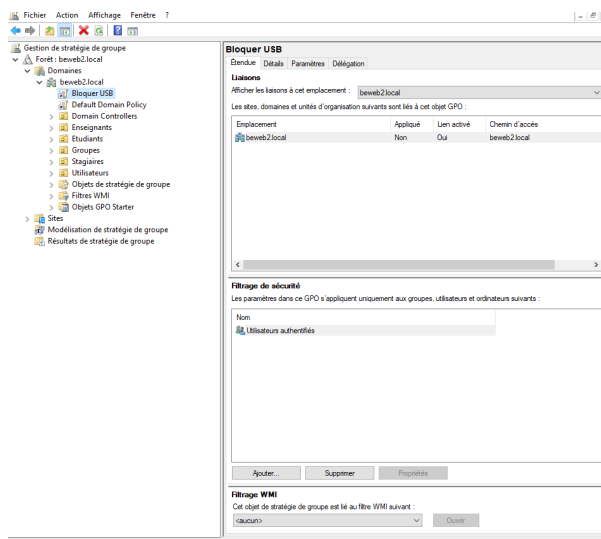
Suite à l'étape 4 :



Suite à l'étape 5 :



Suite à l'étape 6 :



Suite à l'étape 7 (falcultative si l'on veut activer ou non le pare-feu) :

```
P5 C:\Users\Administrateur.DC02\Desktop> .\Setup-ADServer.ps1
applet de commande Setup-ADServer.ps1 à la position 1 du pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
Etape: 7
Etape 7 : Activation/d activation du pare-feu.
Entrez 'Enable' ou 'Disable' pour activer/d activer le pare-feu: disable
Firewall d activ  .
P5 C:\Users\Administrateur.DC02\Desktop>
```

Bonus : Cr ation automatique d'utilisateurs depuis un fichier CSV

Je peux pr parer un fichier CSV avec les colonnes : Nom, Prenom, SamAccountName, MotDePasse, par exemple.

Fichier users.csv :

```
Name,GivenName,Surname,SamAccountName,UserPrincipalName>Password,OU
Alice,Durand,Alice,Durand,adurand,beweb2.local,Azerty1234,Utilisateurs
Bob,Petit,Bob,Petit,bpetit,beweb2.local,Azerty1234,Utilisateurs
```

Puis dans le script :


```

"8" {
    Write-Host "Étape 8 : Importation d'utilisateurs depuis un fichier CSV."

    $CsvPath = "$env:USERPROFILE\Desktop\users.csv"
    if (-Not (Test-Path $CsvPath)) {
        Write-Host "Erreur : Fichier CSV non trouvé à l'emplacement $CsvPath."
        break
    }

    $importedUsers = Import-Csv -Path $CsvPath

    foreach ($user in $importedUsers) {
        Write-Host "Création de l'utilisateur $($user.Name)..."

        $ouPath = "OU=$($user.OU),$DomainDN"
        if (-not (Get-ADOrganizationalUnit -Filter "Name -eq '$($user.OU)'" -
SearchBase $DomainDN -ErrorAction SilentlyContinue)) {
            New-ADOrganizationalUnit -Name $user.OU -Path $DomainDN
        }

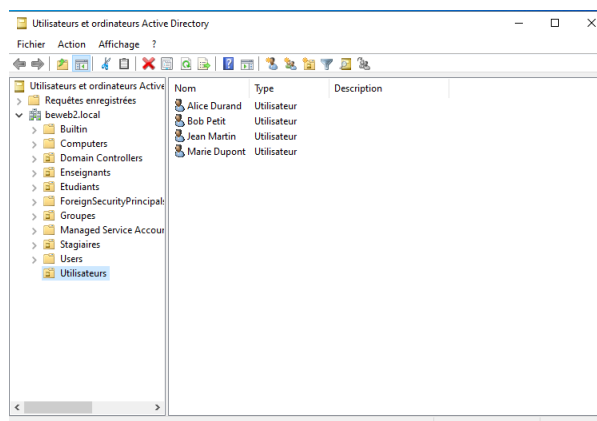
        New-ADUser -Name $user.Name `
            -GivenName $user.GivenName `
            -Surname $user.Surname `
            -SamAccountName $user.SamAccountName `
            -UserPrincipalName $user.UserPrincipalName `
            -AccountPassword (ConvertTo-SecureString $user.Password -
AsPlainText -Force) `
            -Enabled $true `
            -Path $ouPath `
            -ChangePasswordAtLogon $true

        Write-Host "Utilisateur $($user.Name) créé."
    }
}

```

Je rajoute cette partie à mon script d'installation automatisée en étape 8.

Suite à l'étape 8 :



Conclusion

Avec ce script, je peux automatiser rapidement et efficacement la mise en place complète d'un domaine Active Directory, tout en facilitant les modifications futures grâce à l'usage de variables. Le script peut aussi être adapté pour créer des utilisateurs à la volée via un fichier CSV, ce qui est très pratique en entreprise.

Optimisations :

Centralisation des paramètres

J'ai regroupé toutes les valeurs modifiables dans une seule variable globale `$Config`. Cela me permet de modifier rapidement une IP, un nom de domaine ou un chemin de partage sans avoir à parcourir tout le script.

Modularité par étapes

J'ai gardé l'approche étape par étape (paramètre `-Etape`), pour que je puisse exécuter uniquement ce qui m'intéresse, sans relancer tout le script à chaque test. Ça facilite aussi le débogage et l'automatisation.

Création de fonctions réutilisables

Pour éviter les copier-coller inutiles, j'ai créé :

`Restart-IfConfirmed` → Pour gérer les redémarrages en fin d'étape, avec confirmation utilisateur.

`Ensure-OUExists` → Pour vérifier si une OU existe déjà avant de la créer (évite les erreurs si le script est relancé plusieurs fois).

Gestion des utilisateurs et groupes simplifiée

J'ai transformé la création d'utilisateurs et de groupes en tableaux d'objets, ce qui me permet de les parcourir en boucle (`foreach`), rendant le script plus lisible et plus court. Si je veux ajouter un nouvel utilisateur, il me suffit d'ajouter une ligne dans la liste `$Users`.

Contrôle d'erreurs basique intégré

J'ai mis en place des vérifications (`-ErrorAction SilentlyContinue`, `Test-Path`, etc.) pour éviter que le script échoue bêtement si un élément existe déjà ou si un fichier est manquant.

Standardisation des chemins et noms

Tous les chemins (exemple : le DN du domaine ou les chemins de partage) sont automatiquement générés à partir des variables, ce qui évite les erreurs de frappe.

Encodage et affichage propre

J'ai forcé l'encodage UTF-8 en console pour que les caractères accentués s'affichent correctement quand je travaille sous Windows Server.

Pourquoi ces choix ?

- Parce que je voulais un script lisible, pédagogique et fiable, même pour une débutante comme moi qui doit m'assurer que ça fonctionne du premier coup.
- Je pourrai le réutiliser et l'adapter facilement pour d'autres projets AD.
- Ces bonnes pratiques sont aussi celles que je compte appliquer dans mes futurs environnements pro (Git, CI/CD, scripts PowerShell d'entreprise...).

Script final :

Script PowerShell - Automatisation Active Directory (Optimisé)

```
# Script PowerShell - Automatisation Active Directory (Optimisé)

# === Paramètres d'exécution ===
param(
    [Parameter(Mandatory = $true)]
    [ValidateSet("1", "2", "3", "4", "5", "6", "7", "8")]
    [string]$Etape
)

# === Variables globales ===
$Config = @{
    InterfaceAlias = "Ethernet"
    IPAddress      = "192.168.1.10"
    PrefixLength   = 24
    DefaultGateway = "192.168.1.1"
    DNSServer      = "192.168.1.1"
    ComputerName   = "DC03"
    DomainName      = "beweb2.local"
    DomainDN       = "DC=beweb2,DC=local"
    SharePath      = "C:\Partage"
    CsvPath        = "$env:USERPROFILE\Desktop\users.csv"
}

# === Fonctions réutilisables ===
function Restart-IfConfirmed {
    if ((Read-Host "Redémarrer maintenant ? (Y/N)") -eq "Y") {
        Restart-Computer
    }
}

function Ensure-OUExists {
    param([string]$OUName)

    $ouDN = "OU=$OUName,$($Config.DomainDN)"
    # Vérifie si l'OU existe déjà
    $existingOU = Get-ADOrganizationalUnit -Filter "DistinguishedName -eq '$ouDN'"
    -ErrorAction SilentlyContinue
}
```

```

if (-not $existingOU) {
    Write-Host "Création de l'OU $OUname"
    New-ADOrganizationalUnit -Name $OUname -Path $Config.DomainDN
}
else {
    Write-Host "OU $OUname existe déjà"
}
return $ouDN
}

# === Étapes ===
switch ($Etape) {

    "1" {
        Write-Host "Étape 1 : Configuration réseau et renommage."

        Get-NetIPAddress -InterfaceAlias $Config.InterfaceAlias | Remove-
NetIPAddress -Confirm:$false
        New-NetIPAddress -InterfaceAlias $Config.InterfaceAlias -IPAddress
$Config.IPAddress -PrefixLength $Config.PrefixLength -DefaultGateway
$Config.DefaultGateway
        Set-DnsClientServerAddress -InterfaceAlias $Config.InterfaceAlias -
ServerAddresses $Config.DNSServer

        Rename-Computer -NewName $Config.ComputerName
        Restart-IfConfirmed
    }

    "2" {
        Write-Host "Étape 2 : Installation AD-DS et DNS."

        Install-WindowsFeature -Name AD-Domain-Services, DNS -
IncludeManagementTools
        $SafeModePassword = Read-Host -Prompt "Mot de passe DSRM" -AsSecureString

        Install-ADDSForest -DomainName $Config.DomainName -
SafeModeAdministratorPassword $SafeModePassword -InstallDNS -Force -
NoRebootOnCompletion
        Restart-Computer
    }

    "3" {
        Write-Host "Étape 3 : Création des OU principales."
        "Enseignants", "Etudiants", "Stagiaires" | ForEach-Object { Ensure-
OUExists $_ }
    }

    "4" {
        Write-Host "Étape 4 : Création des utilisateurs et groupes."

        $ouPaths = @{
            Utilisateurs = Ensure-OUExists "Utilisateurs"
            Groupes      = Ensure-OUExists "Groupes"
        }
    }
}

```

```

$Users = @(
    @{Name="mdupont"; GivenName="Marie"; Surname="Dupont";
    SamAccountName="mdupont"; UserPrincipalName="mdupont@ $($Config.DomainName)";
    Password="P@ssw0rd"; OU="Utilisateurs"},
    @{Name="jmartin"; GivenName="Jean"; Surname="Martin";
    SamAccountName="jmartin"; UserPrincipalName="jmartin@ $($Config.DomainName)";
    Password="P@ssw0rd123"; OU="Utilisateurs"}
)

foreach ($user in $Users) {
    # Vérifie si l'utilisateur existe déjà
    if (-not (Get-ADUser -Filter "SamAccountName -eq
'$($user.SamAccountName)'" -ErrorAction SilentlyContinue)) {
        Write-Host "Création de l'utilisateur $($user.Name)"
        New-ADUser -Name $user.Name `
            -GivenName $user.GivenName `
            -Surname $user.Surname `
            -SamAccountName $user.SamAccountName `
            -UserPrincipalName $user.UserPrincipalName `
            -AccountPassword (ConvertTo-SecureString $user.Password
-AsPlainText -Force) `
            -Enabled $true `
            -Path $ouPaths[$user.OU] `
            -ChangePasswordAtLogon $true
    }
    else {
        Write-Host "Utilisateur $($user.Name) existe déjà"
    }
}

$Groups = @(
    @{Name="Admins"; GroupScope="Global"; OU="Groupes"},
    @{Name="UtilisateursNormaux"; GroupScope="Global"; OU="Groupes"}
)

foreach ($group in $Groups) {
    # Vérifie si le groupe existe déjà
    if (-not (Get-ADGroup -Filter "Name -eq '$($group.Name)'" -ErrorAction
SilentlyContinue)) {
        Write-Host "Création du groupe $($group.Name)"
        New-ADGroup -Name $group.Name -GroupScope $group.GroupScope -Path
$souPaths[$group.OU]
    }
    else {
        Write-Host "Groupe $($group.Name) existe déjà"
    }
}

# Ajout des membres au groupe Admins
Write-Host "Ajout des utilisateurs aux groupes"
Add-ADGroupMember -Identity "Admins" -Members @("mdupont", "jmartin")
}

```

```

"5" {
    Write-Host "Étape 5 : Création d'un partage SMB."

    Write-Host "DEBUG : Chemin partagé = '$($Config.SharePath)'"

    if (-not (Test-Path $Config.SharePath)) {
        Write-Host "Création du dossier $($Config.SharePath)..."
        New-Item -ItemType Directory -Path $Config.SharePath | Out-Null
    }
    else {
        Write-Host "Le dossier $($Config.SharePath) existe déjà."
    }

    Write-Host "Création du partage SMB..."

    if (Get-SmbShare -Name "Partage" -ErrorAction SilentlyContinue) {
        Remove-SmbShare -Name "Partage" -Force
        Write-Host "Ancien partage 'Partage' supprimé."
    }

    New-SmbShare -Name "Partage" -Path $Config.SharePath -FullAccess
    "BUILTIN\Administrateurs"
    Write-Host "Partage SMB créé avec succès."
}

"6" {
    Write-Host "Étape 6 : Configuration GPO - Blocage USB."
    $GpoName = "Bloquer USB"

    $Gpo = Get-GPO -Name $GpoName -ErrorAction SilentlyContinue
    if (-not $Gpo) { $Gpo = New-GPO -Name $GpoName -Comment "Désactivation
USB" }

    Set-GPRegistryValue -Name $Gpo.DisplayName -Key
    "HKLM\SYSTEM\CurrentControlSet\Services\USBSTOR" -ValueName "Start" -Type DWord -
    Value 4
    New-GPLink -Name $Gpo.DisplayName -Target $Config.DomainDN
}

"7" {
    Write-Host "Étape 7 : Activation/désactivation du pare-feu."
    $Action = Read-Host "Activer ou désactiver le pare-feu (Enable/Disable)"
    switch ($Action.ToLower()) {
        "enable" { Set-NetFirewallProfile -Profile Domain,Public,Private -
Enabled True; Write-Host "Pare-feu activé." }
        "disable" { Set-NetFirewallProfile -Profile Domain,Public,Private -
Enabled False; Write-Host "Pare-feu désactivé." }
        default { Write-Host "Action non reconnue." }
    }
}

"8" {
    Write-Host "Étape 8 : Importation d'utilisateurs depuis un fichier CSV."

```

```

        if (-not (Test-Path $Config.CsvPath)) {
            Write-Host "Erreur : Fichier CSV non trouvé à l'emplacement
 $($Config.CsvPath)."
            break
        }

        Import-Csv -Path $Config.CsvPath | ForEach-Object {
            $ouPath = Ensure-OUExists $_.OU

            New-ADUser -Name $_.Name `
                -GivenName $_.GivenName `
                -Surname $_.Surname `
                -SamAccountName $_.SamAccountName `
                -UserPrincipalName $_.UserPrincipalName `
                -AccountPassword (ConvertTo-SecureString $_.Password -
AsPlainText -Force) `
                -Enabled $true `
                -Path $ouPath `
                -ChangePasswordAtLogon $true

            Write-Host "Utilisateur $($_.Name) créé."
        }
    }

    Default {
        Write-Host "Étape non reconnue."
    }
}

```