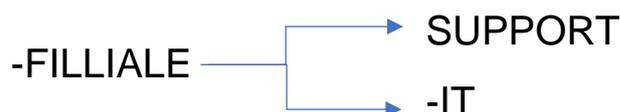




## INTRODUCTION

Nous devons mettre en place plusieurs script PowerShell permettant d'automatiser plusieurs paramètres comme ; attribuer une adresse IP statique, attribué un Hostname, créé un AD et une forêt, Création de OUs ; création d'utilisateur à partir d'un fichier .CSV qui doivent être déplacé dans les OUs selon leurs ID présent dans le .CSV

### Arborescence des OUs ;



## PRE-REQUIS

- ✚ Un Editeur de texte
- ✚ Oracle VirtualBox
- ❖ 1 VM Windows server 2016 >
  - ≥ Allocation mémoire : 4gb
  - ≥ Stockage : 50 go
  - ≥ Iso : Windows Server (domaine : Ayoub.local)
  - ≥ 2nd Carte Réseau « Interne » :
  - ≥ Télécharger mon dépôt sur [GitHub](#)



## TUTORIEL

[Lien vers mon GitHub](#)

(J'ai intentionnellement pousser le tp plus loin)

### Script 1

- Je declare mes variable, le dns nous servira temporairement pour installer des add-on dans le Script 2 il sera par la suite remplacer par 127.0.0.1
- ***\$adapter = Get-NetAdapter).ifIndex*** nous permet de d'obtenir les propriétés basiques de l'adaptateurs réseau ***index*** ainsi il sera encapsulé dans notre variable ***\$adapter***
- ***\$prefix = "24"*** correspond à notre CIDR
- ***[System.Net.Dns]::GetHostName()*** nous permet de d'obtenir le Nom d'hôte de la machine il sera encapsulé dans la variable ***\$myhost***
- ***Read-Host -Prompt*** nous permet de inviter un utilisateur à entrer du contenu décrite par ***'saisissez le nom du pc ? o/n'*** le choix saisie et encapsulé dans la variable ***\$demande***
- ***\$defauthostn = 'AyoubAD'*** nous permet de définir nom d'hôte par défaut il nous servira plus tard

```
$ip = "192.168.2.2"
$prefix = "24"
$GW = "192.168.100.1"
$DNS = "8.8.8.8"
$adapter = (Get-NetAdapter).ifIndex
$myhost =
[System.Net.Dns]::GetHostName()
$demande = Read-Host -Prompt '
saisissez le nom du pc ? o/n '
$defauthostn = 'AyoubAD'
```

- ***New-NetIPAddress -IPAddress*** nous permet de attribuer une adresse ip ***-PrefixLength*** correspond à notre CIDR
- ***-InterfaceIndex*** correspond aux propriétés basiques de l'adaptateurs réseau ***index***
- ***-DefaultGateway*** correspond à notre passerelle
- ***Set-DNSClientServerAddress -InterfaceIndex (Get-NetAdapter).InterfaceIndex -ServerAddresses \$DNS*** nous permet de attribuer un DNS à notre Windows server 2 il sera par la suite remplacer par 127.0.0.1 (lors du lancement du script 2)

```
New-NetIPAddress -IPAddress $ip -PrefixLength $prefix `
-InterfaceIndex $adapter -DefaultGateway $GW
Set-DNSClientServerAddress -InterfaceIndex (Get-
NetAdapter).InterfaceIndex -ServerAddresses $DNS
```



J'ai poussé le TP plus loin en ajoutant un switch et des boucles Else if pour pas que la machine redémarre inutilement

```
switch($demande){
o {
    $newhostnm = Read-Host -Prompt 'saisir un Nom d'hôte pour votre machine'
    if ($myhost -eq $newhostnm) {
        Write-Host -ForegroundColor Yellow " $newhostnm est déjà le nom de la machine "
    }
    else {
        Write-Host -ForegroundColor Green " le nom de la machine sera $newhostnm après redémarrage"
        rename-computer -NewName $newhostnm -Force
        Write-Host -ForegroundColor Yellow " l'ordinateur redémarre tout seul dans 15s"
        Start-Sleep -s 15
        Restart-Computer -Force
    }
}
```

- **Switch** vous permet de fournir une variable et une liste de valeurs possibles, ici 'O' pour oui et 'N' pour non. Si la valeur correspond à la variable, le scriptblock est exécuté, la variable **\$demande** précédemment expliquer est implémenter dans notre **Switch**
- **Read-Host -Prompt** nous permet pour inviter un utilisateur à entrer du contenu décrite par '**saisir un Nom d'hôte pour votre machine**' le choix saisie et encapsulé dans la variable **\$newhostnm**
- **if (\$myhost -eq \$newhostnm)** et une boucle qui nous permet de vérifier si le nom d'hôte saisie précédemment est égale « (**\$myhost -eq \$newhostnm**) » nom d'hôte de la machine
- **Write-Host** est une applette qui permet d'afficher un message ici "**\$newhostnm est déjà le nom de la machine**" **\$newhostnm** correspond nom d'hôte saisie
- **Else** = Sinon le script affiche le nouveau nom d'hôte et prévient que la machine va redémarrer dans 15 secondes
- **rename-computer -NewName \$newhostnm -Force** nous permet de renommer la machine avec le nom d'hôte saisie précédemment
- **Start-Sleep -s 15** permet de réaliser une pose de 15 secondes et **Restart-Computer -Force** permet de redémarrez la machine



```
n{
  if ($myhost -eq $defauthostn) {
    Write-Host -ForegroundColor Yellow " $defauthostn est déjà le nom de la
machine "
  }

  if ($myhost -ne $defauthostn ) {
    Write-Host -ForegroundColor Green " le nom de la machine sera $defauthostn
après redémarrage"

    rename-computer -NewName $defauthostn -Force

    Write-Host -ForegroundColor Yellow " l'ordinateur redémarre tout seul dans
30s"

    Start-Sleep -s 30

    Restart-Computer -Force
  }
}
#pour les saisie invalide
default{write-warning "Saisie invalide"
}
}
```

- La continuité du Switch pour la saisie n=NON, le principe est le même que décrit lors de la dernière page sauf que la boucle vérifie si la variable ***\$defauthostn = 'AyoubAD'*** correspondant au nom d'hôte par défaut et égale au nom d'hôte de la machine « ***(\$myhost -ne \$defauthostn)*** »
- Enfin avant de fermer le Switch (avec une accolade fermante) nous devons mettre en place un paramètre qui permet d'identifier les saisie invalide pour se faire ***default{write-warning "Saisie invalide"***

Ayoub Belbachir



### Script 2

J'ai poussé le TP plus loin en installant un gestionnaire de paquets pour Windows [CHOCO](#) et un module complémentaire telle que [BurntToast](#) pour afficher des notification popup afin personnaliser les scripts

```
$monmdp = ConvertTo-SecureString -String "Btssio92" -AsPlainText -Force  
  
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))  
  
choco install BurntToast-psmodule -y
```

- L'applet de commande **ConvertTo-SecureString** convertit des chaînes standard chiffrées en chaînes sécurisées.
- Pour installer le gestionnaire de paquet nous avons besoin de bypass la sécurité de Windows server **Set-ExecutionPolicy Bypass -Scope Process -Force;**  
**[System.Net.ServicePointManager]::SecurityProtocol =**  
**[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object**  
**System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))**
- Nous les installons BurnToast pour personnaliser nos notifications grâce à notre nouveau gestionnaire de paquets avec la commande **choco install BurntToast-psmodule -y**



```

Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
Install-WindowsFeature DNS -IncludeManagementTools
Install-ADDSForest `
  -CreateDnsDelegation:$false `
  -DatabasePath "C:\Windows\NTDS" `
  -DomainMode 7 `
  -ForestMode 7 `
  -DomainName "AYOUB.local" `
  -InstallDns:$true `
  -SafeModeAdministratorPassword $monmdpap `
  -NoRebootOnCompletion:$true `
  -SysvolPath "C:\Windows\SYSVOL" `
  -LogPath "C:\Windows\NTDS" `
  -Force:$true
Install-WindowsFeature NPAS -IncludeManagementTools
  
```

- **Install-WindowsFeature** permet d'installer des rôles tels que le contrôleur de domaine et le DNS, **-IncludeManagementTools** permet d'installer automatiquement tous les services de rôle ou les fonctionnalités enfants.
- **Install-ADDSForest** nous installons ensuite une nouvelle forêt avec les paramètres souhaiter sans délégation (les paramètres peuvent changer selon la configuration souhaiter)
- **New-BurntToastNotification** nous permet de créer une notification avec un texte et un logo personnalisé
- **Start-Sleep -s 40** permet de réaliser une pause de 40 secondes et **Restart-Computer -Force** permet de redémarrer la machine

Ayoub Belbachir



## Script 3

J'ai poussé le TP plus loin pour le script 3 le script vérifie si les OUs existe avant de les créés grâce à des boucles if Else.

- Je déclare mes variables **\$IT**, **\$parentOU** et **\$SUPPORT** correspond au chemin respectif des OUs

```
$parentOU = 'OU=FILLIALE,DC=AYOUB,DC=local'
$IT      = 'OU=IT,OU=FILLIALE,DC=AYOUB,DC=local'
$SUPPORT = 'OU=SUPPORT,OU=FILLIALE,DC=AYOUB,DC=local'
```

```
if([ADSI]::Exists("LDAP://$parentOU")) {
    Write-Host -ForegroundColor GREEN "Filliale existe"
}
else {
    Write-Host -ForegroundColor CYAN "FILLIAL n'existe pas, ne t'inquiète pas je m'occupe de les créer pour toi."
    New-ADOrganizationalUnit -Name FILLIALE -Path "DC=AYOUB,DC=local"
}
if([ADSI]::Exists("LDAP://$IT")) {
    Write-Host -ForegroundColor GREEN "IT exists "
}
else {
    Write-Host -ForegroundColor CYAN "IT n'existe pas, ne t'inquiète pas je m'occupe de les créer pour toi."
    New-ADOrganizationalUnit -Name IT -Path "OU=FILLIALE,DC=AYOUB,DC=local"
}
```

- **[ADSI]** (ADSI=Active Directory Service Interfaces) , qui est une série d'interfaces développées à partir du modèle COM (« Component Object Model ») permettant d'accéder à la base Active Director. **::Exists** nous permet de vérifier si les OUs existe sinon nous les créons

```
if([ADSI]::Exists("LDAP://$SUPPORT")) {
    Write-Host -ForegroundColor GREEN "IT existe "
}
else {
    Write-Host -ForegroundColor CYAN "SUPPORT n'existe pas, ne t'inquiète pas je m'occupe de les créer pour toi."
    New-ADOrganizationalUnit -Name SUPPORT -Path "OU=FILLIALE,DC=AYOUB,DC=local"
}
New-BurntToastNotification -Text "Script 3 ", "execution du script 3 terminer" -AppLogo
C:\Windows\SCRIPT_AYOUB_BELBACHIR\tmp.png
```

Ayoub Belbachir



## Script 4

Le script 4 permet de créer des utilisateurs depuis un fichier csv et de leur attribuer un ID pour pouvoir les trier dans les OUs

- Je déclare mes variables, **\$ecchi** correspond au chemin du fichier **CSV**

```
$ecchi = Import-csv -Delimiter ";" -Path
C:\Windows\SCRIPT_AYOUB_BELBACHIR\yohoho.csv

$OUIT = 'OU=IT,OU=FILLIALE,DC=AYOUB,DC=local'

$SUPPORT = 'OU=SUPPORT,OU=FILLIALE,DC=AYOUB,DC=local'
```

```
foreach ($User in $ecchi)
{
    $Username = $User.Username
    $Password = $User.password
    $Prenom = $User.Prenom
    $ID = $User.ID
    $Nom = $User.Nom
    $Chemin = $User.Chemin

    if ( Get-ADUser -F { SamAccountName -eq $Username } ) {
        Write-Host -ForegroundColor Green "OHoho! $Username est déjà présent dans l'AD"
    }

    else {

        New-ADUser -SamAccountName $Username -UserPrincipalName "$Username@AYOUB.local" -Name
"$Prenom $Nom" -GivenName $Prenom -Surname $Nom -Enabled $True -DisplayName "$Nom, $Prenom" -
Path $Chemin -AccountPassword (convertto-securestring $Password -AsPlainText -Force)

        Write-Host -ForegroundColor Green "$Username il a été créé"
    }
}
```

- La boucle foreach la boucle **foreach** va automatiquement traiter toutes les lignes de notre fichier CSV implémenter avec la variable correspondant au chemin de fichier CSV **\$ecchi** encapsuler dans la variable **\$User**
- Nous encapsulons ensuite la variable correspondante a notre fichier CSV dans des nouvelles variables distinctes
- **New-ADUser** permet de créer les utilisateurs avec les paramètres présents dans notre csv précédemment implémenter dans des nouvelles

Ayoub Belbachir



Cette partie du script 4 vas nous permettre de triée les nouveaux utilisateurs .

```
$Usercsv = (Get-ADUser $Username).distinguishedName
If ( $ID -le 500 ) {
    Move-ADObject -Identity $Usercsv -TargetPath $OUIT
    Add-ADGroupMember -Identity portalcaptif -Members $Username
}
If ( $ID -ge 501 ) {
    Move-ADObject -Identity $Usercsv -TargetPath $SUPPORT
}
}
New-BurrtToastNotification -Text "Script 3 ", "execution du script 3 terminer" -AppLogo
C:\Windows\SCRIPT_AYOUB_BELBACHIR\tmp.png
```

- La variable \$Usercsv récupère les SamAccountName directement sous format cn=Username,ou=informatique,dc=it-connect,dc=local sans ça on obtient une erreur lors du déplacement de l'objet \$Username vers un des OUs
- La boucle if trie les utilisateurs ayant un ID inférieur ou égale (-le) à 500 dans l'OU IT
- La boucle if trie les utilisateurs ayant un ID supérieur ou égale (-le) à 501 dans l'OU SUPPORT





```
switch($choix)
{
    1{
        Write-Host "Mettre l'adresse IP en statique et modifier le nom de la machine"
        $PlayWav.playsync()
        C:\Windows\SCRIPT_AYOUB_BELBACHIR\Script1.ps1
        Read-Host -Prompt "Appuyer sur une touche une fois la lecture terminée"
        clear
        PowerShell -NoExit $menups1
    }
    2{
        Write-Host "mise en route du scripte creation de l'ad"
        $PlayWav.playsync()
        C:\Windows\SCRIPT_AYOUB_BELBACHIR\Script2.3.ps1
        Read-Host -Prompt "Appuyer sur une touche une fois la lecture terminée"
        clear
        PowerShell -NoExit $menups1
    }
    3{
        Write-Host "Création Des OUs"
        $PlayWav.playsync()
        C:\Windows\SCRIPT_AYOUB_BELBACHIR\Script3.ps1
        Read-Host -Prompt "Appuyer sur une touche une fois la lecture terminée"
        Clear
        PowerShell -NoExit $menups1
    }
    4{
        Write-Host "Création des utilisateurs et déplacements de de ceux-là selon leurs
ID"
        $PlayWav.playsync()
        C:\Windows\SCRIPT_AYOUB_BELBACHIR\Script4.ps1
        https://gitlab.com/digi-boy1/SCRIPT/-
/blob/d81b025c208c6202334182aee7dafa8744ee3a50/readme.md
        Clear
        PowerShell -NoExit $menups1
    }
}
```

Ayoub Belbachir



```
0{  
    $PlayWav.playsync()  
    Stop-Process -Name PowerShell  
    }  
  
    default{ Write-Host -ForegroundColor RED "$choix ne correspond pas au choix  
possible"  
    Read-Host -Prompt "appuyez sur n'importe qu'elle touche pour réessayer "  
    Clear  
    PowerShell -NoExit $menups1  
    }  
}
```

Résulta (disponible aussi sur mon [GitHub](#))