

Compte Rendu des Activités de stage

Au cours de ma période de stage, j'ai travaillé sur divers projets et tâches liés au domaine informatique que je vous présenterai dans l'ordre suivant :

- 1. Script Bash de diagnostique réseau (p.2 – 3)**
- 2. Préparation de PC fixe avec Clonezilla (p.4 – 6)**
- 3. Amélioration d'un script Batch pour la préparation de PC portables (p.7 – 8)**
- 4. Participation à la création d'un script Excel pour trier une liste d'employés (p.9 – 10)**
- 5. Participation à la création d'un script PowerShell (p.11 – 13)**
- 6. Participation aux interventions sur site en Seine et Marne (p.14)**

1. Création d'un script Bash de diagnostic réseau

J'ai développé un script Bash qui permet aux employés de l'association de diagnostiquer les problèmes réseau auxquels ils sont confrontés. Ce script fournit des informations utiles pour résoudre les problèmes courants, ce qui facilite le travail de l'équipe informatique.

```
ip_valide=$(ip -4 -o a show | awk '/inet / {print $4}' | grep 10.0 | cut -d'/' -f1 | cut -d'.' -f1,2,3)

ip_de_depart=${ip_valide}

ip_du_poste=$(ip -4 -o a show | awk '/inet / {print $4}' | sed '1d' | cut -d'/' -f1)
```

```
verif_ip () {
    echo "Vérification de la présence d'une adresse IP : \c"
    if [ -n "$ip_du_poste" ]; then
        if [ -z "$ip_valide" ]; then
            # Cette boucle permet de supprimer les IP du poste dans le cas où elle ne serait pas du type 10.0.X.X
            for suppr_ip in $ip_du_poste; do
                echo "Suppression de l'IP : ${suppr_ip}"
                ip a del ${suppr_ip} dev ${carte_reseau} >/dev/null 2>&1
            done
            result_test=1
        else
            result_test=0
            echo "OK"
        fi
    else
        # Tente de demander une IP au serveur DHCP si jamais aucune IP n'a été trouvée sur le poste
        timeout 4 dhclient 2>&1
        if [ -z "$(ip -4 -o a show | awk '/inet / {print $4}' | grep 10.0)" ]; then
            result_test=1
            echo "Erreur"
        else
            result_test=0
            echo "OK"
        fi
    fi
}
```

Le script est composé d'une succession de fonctions comme celle-ci-dessus qui permet de vérifier si la machine possède une adresse IP et également si le format de celle-ci correspondant au format utilisé par l'association, c'est-à-dire une adresse IP de classe A avec un CIDR 16 et si jamais le poste ne possède pas d'IP alors le script lance un dhclient permettant de faire une requête d'adresse IP au serveur DHCP.

```

# Récupère le nom de la première carte réseau physique
carte_reseau=$(ip a | awk '/2: / {print $2}')
if [ -z "$carte_reseau" ]; then
    echo "Erreur: Aucune carte réseau trouvée...."
    exit 1
else
    test_loopback
    etat_loopback=0
    if [ $result_test -eq 1 ]; then
        echo "Erreur au niveau de la carte réseau."
    else
        verif_ip
        if [ $result_test -eq 1 ]; then
            test_toutes_les_passerelles
            if [ $result_test -eq 1 ]; then
                echo "Vérifier si le câble Ethernet (RJ45) est bien branché à l'ordinateur"
                exit 1
            else
                if [ -n "$ip_du_poste" ]; then
                    echo "Votre ancienne IP : ${ip_du_poste}"
                    echo "Votre nouvelle IP : $(ip -4 -o a show | awk '/inet / {print $4}' | sed '1d' | cut -d '/' -f1)"
                fi
            fi
        else
            test_passerelle_locale
            if [ $result_test -eq 1 ]; then
                # Cette boucle permet de supprimer toutes les IP du poste si le test de ping à la passerelle n'a pas fonctionné
                for suppr_ip in $ip_du_poste; do
                    echo "Suppression de l'IP : ${suppr_ip}"
                    ip a del ${suppr_ip} dev ${carte_reseau} >/dev/null 2>&1
                done
            fi
            test_toutes_les_passerelles
            if [ $result_test -eq 1 ]; then
                passerelle=$(ip route | awk '/default / {print $3}')
                if [ -z "$passerelle" ]; then
                    echo "Aucune passerelle configurée sur le poste."
                    echo "Vérifier si le câble Ethernet (RJ45) est bien branché à l'ordinateur"
                else
                    echo "Une adresse de passerelle est présente mais elle n'est pas joignable : $passerelle"
                fi
            fi
            exit 1
        fi
    fi
fi

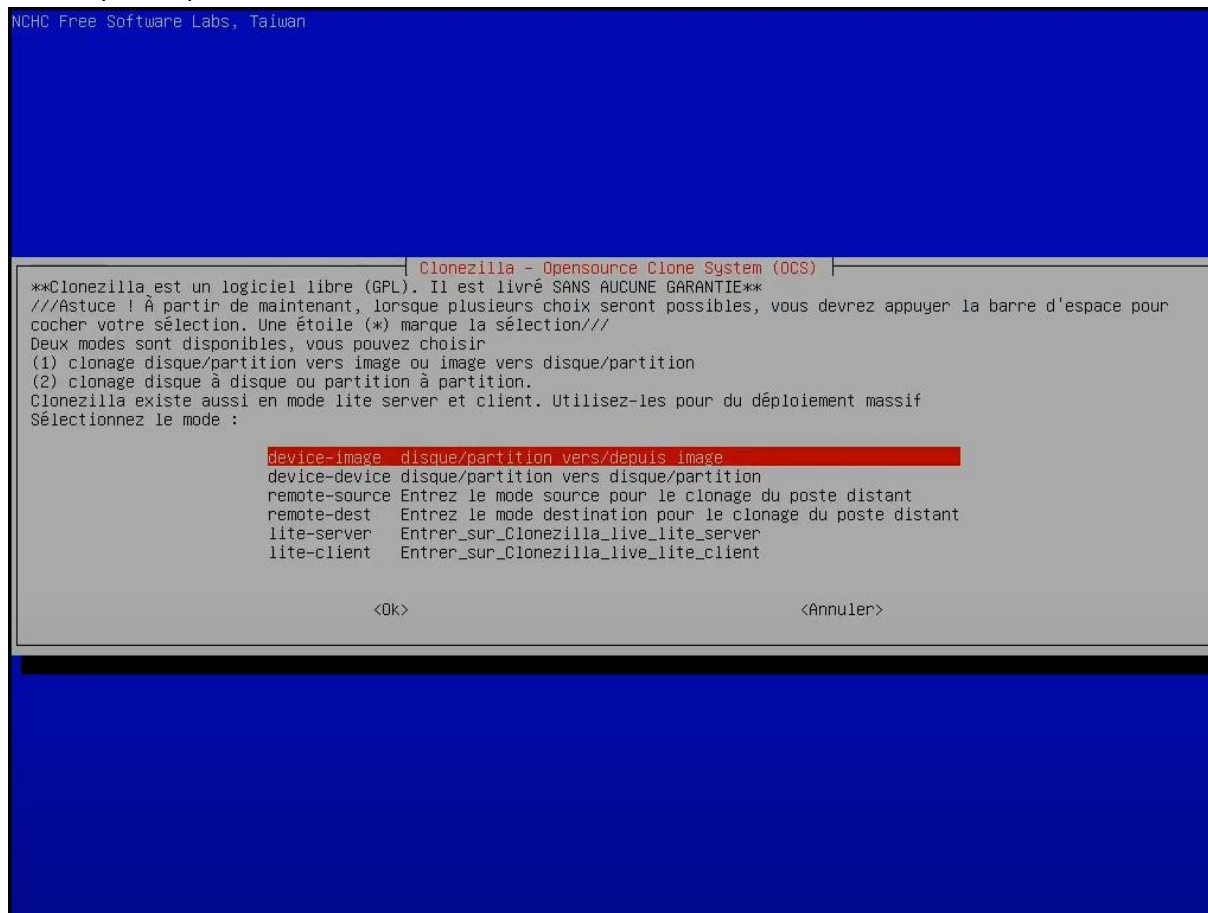
```

L'extrait de code ci-dessus montre la manière dont les fonctions sont appelées et également l'impacte que le résultat de ces fonctions a sur la suite du script comme le fait que la première condition arrête le script si jamais aucune carte réseau n'est détecté sur le PC tout en affichant un message indiquant d'où provient l'erreur.

2. Préparation de PC fixes avec Clonezilla

J'ai participé à la préparation des PC fixes en utilisant l'outil Clonezilla. Cette tâche consistait à cloner l'image système d'un PC configuré correctement vers d'autres machines, permettant ainsi une installation rapide et uniforme du système d'exploitation et des logiciels nécessaires pour ensuite faire la remontée des postes en question sur GLPI.

Les étapes importantes à effectuer :



« device-image » permet de récupérer une image contenu dans une clé USB pour la cloner dans le système de stockage de notre choix

;

```

Clonzilla - Opensource Clone System (OCS) | Mode: disk_to_local_disk
Choix du disque local source.
Le nom du disque est le nom du périphérique sous GNU/Linux. Le premier disque du système se nomme "hda" ou "sda", le 2è est "hdb" ou "sdb", etc.

sda 64.4GB_VMware_Virtual_S_No_disk_serial_no
sdb 64.4GB_VMware_Virtual_S_No_disk_serial_no
sdc 15.4GB_Ultra__SanDisk_Ultra_4C530001240614105092-0:0

<Ok>                                <Annuler>

```

A cette étape on doit choisir le stockage qui contient l'image.

;

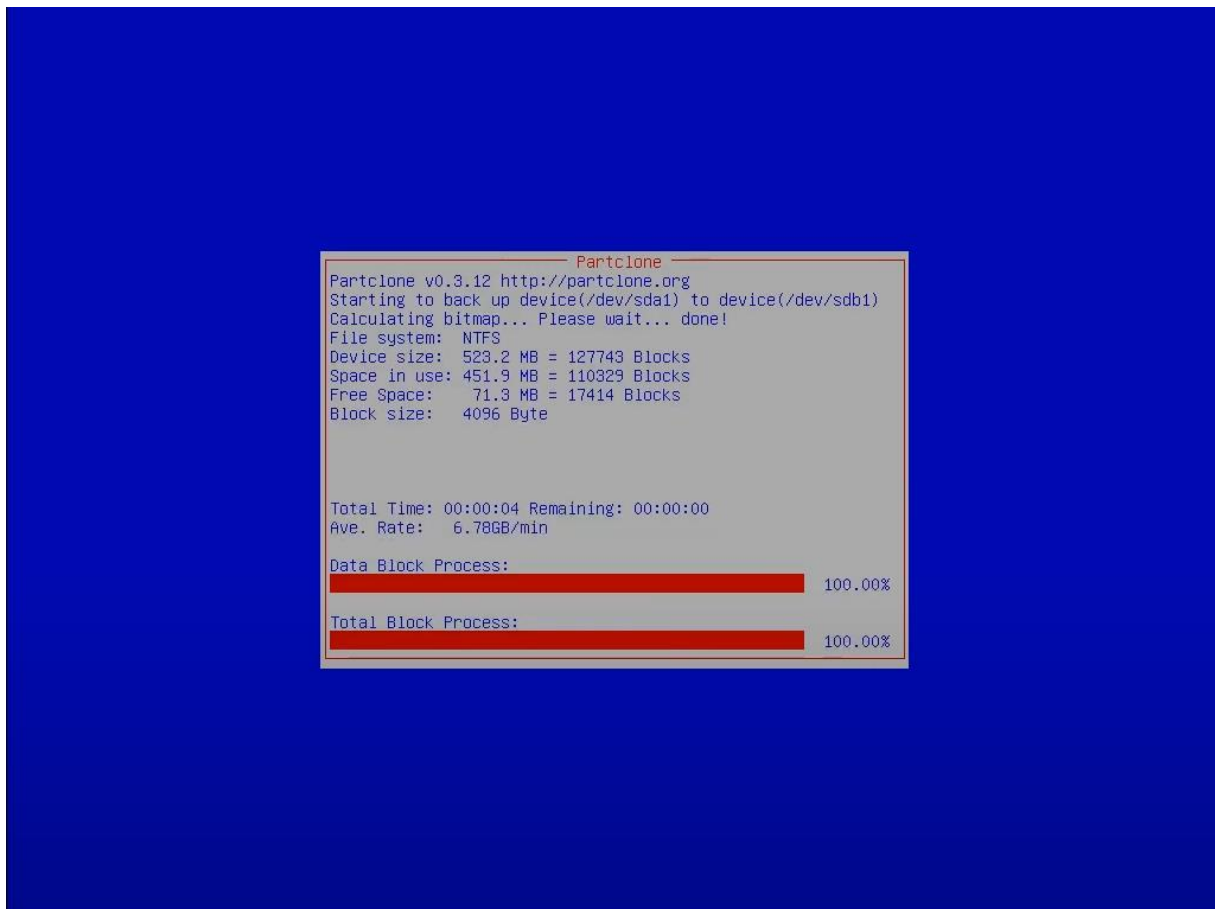
```

PS. La prochaine fois vous pourrez exécuter cette commande directement :
/usr/sbin/ocs-onthefly -g auto -e1 auto -e2 -r -j2 -sfsck -pa reboot -f sda -t sdb
Cette commande a été enregistrée sous le nom suivant pour usage ultérieur si nécessaire: /tmp/ocs-onthefly-2019-07-10-16-42
*****
Appuyez sur "Entrée" pour continuer...
*****
Searching for data partition(s)...
Excluding busy partition or disk...
Unmounted partitions (including extended or swap): sdb1
Collecting info.. done!
Getting /dev/sdb1 info...
ATTENTION!!! ATTENTION!!! ATTENTION!!!
ATTENTION! LES DONNÉES EXISTANTES SUR LE DISQUE OU LA PARTITION VONT ÊTRE ÉCRASÉES ! TOUTES LES DONNÉES EXISTANTES SERONT PERDUE
S: sdb
*****
Machine: VMWare7,1
sdb (64.4GB_VMware_Virtual_S_No_disk_serial_no)
sdb1 (16M_MS_Reserved_Partition(In_VMware_Virtual_S)_No_disk_serial_no)
*****
Etes-vous sûr de vouloir continuer? (y/n) y
OK, c'est parti !!
*****
Alors je vous le redemande :.
ATTENTION!!! ATTENTION!!! ATTENTION!!!
ATTENTION! LES DONNÉES EXISTANTES SUR LE DISQUE OU LA PARTITION VONT ÊTRE ÉCRASÉES ! TOUTES LES DONNÉES EXISTANTES SERONT PERDUE
S: sdb
Etes-vous sûr de vouloir continuer? (y/n) y
OK, c'est parti !!
*****
Saving the GPT of /dev/sda as /tmp/ocs_onthefly_local.WEqpiB/src-gpt.gdisk by sgdisk...
The operation has completed successfully.
Saving the primary GPT of /dev/sda as /tmp/ocs_onthefly_local.WEqpiB/src-gpt-1st.img by dd...
34+0 records in
34+0 records out
17408 bytes (17 kB, 17 KiB) copied, 0.00153987 s, 11.3 MB/s
Saving the secondary GPT of /dev/sda as /tmp/ocs_onthefly_local.WEqpiB/src-gpt-2nd.img by dd...
32+0 records in
32+0 records out
16384 bytes (16 kB, 16 KiB) copied, 0.0018225 s, 9.0 MB/s
*****

```

A cette étape on doit simplement répondre par « y » pour indiquer que l'on a conscience des risques et que l'on souhaite débiter le clonage.

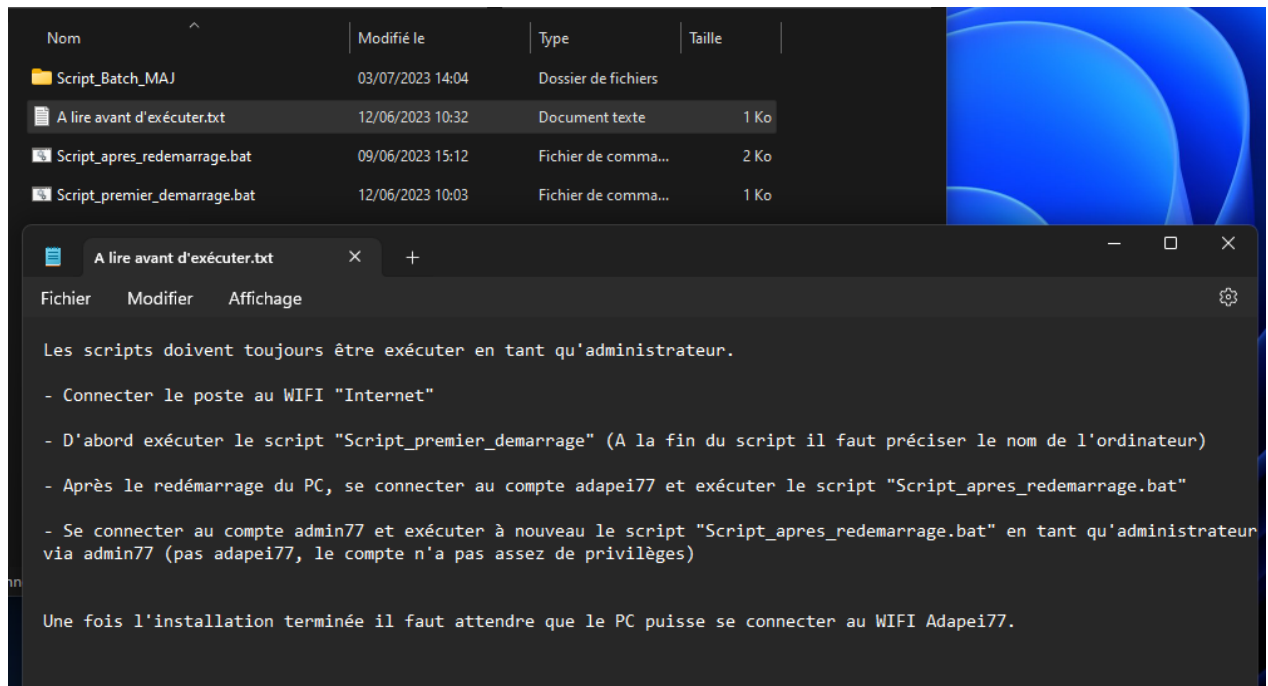
;



Le clonage est en cours.

3. Amélioration d'un script batch pour l'automatisation de la préparation des PC portables

J'ai contribué à l'amélioration d'un script batch existant qui avait pour but d'automatiser la préparation des PC portables. J'ai ajouté des fonctionnalités supplémentaires telles que l'installation de logiciels spécifiques et la création de profils utilisateurs, ce qui a permis de gagner du temps lors du déploiement des nouveaux ordinateurs.



La capture d'écran ci-dessus permet de voir la manière dont est agencé le script batch, c'est-à-dire qu'il est séparé en deux. Le premier est à exécuter au premier démarrage de l'ordinateur et le second est à exécuter après le redémarrage du PC.

Les deux scripts récupèrent tous les fichiers nécessaires dans le dossier « Script_Batch_MAJ ».

```
xcopy "%currentpath%\FortiClientVPN.msi" C:\Source
xcopy "%currentpath%\acrobat.exe" C:\Source
xcopy "%currentpath%\rename.ps1" C:\Source
xcopy "%currentpath%\GLPI-Agent-1.4-x64.msi" C:\Source
xcopy "%currentpath%\CitrixWorkspaceApp.exe" C:\Source
xcopy "%currentpath%\SentinelInstaller_windows_64bit_v22_3_5_887.msi" C:\Source
xcopy "%currentpath%\install\usb-ports-disabler-setup.exe" C:\Source
xcopy "%currentpath%\ChromeStandaloneSetup64.exe" C:\Source
if %TypeConfig%==1 (
    xcopy "%currentpath%\TS1000.exe" C:\Source
)
rem xcopy "%currentpath%\ccsetup561.exe" C:\Users\%username%\Desktop\install
REM xcopy "%currentpath%\webexplugin.msi" C:\Users\%username%\Desktop\install
REM xcopy "%currentpath%\webexapp.msi" C:\Users\%username%\Desktop\install
xcopy "%currentpath%\icone\*" C:\Windows\System32

echo ===== Installation des logiciels =====

echo Installation de FortiClient...
msiexec /i C:\Source\FortiClientVPN.msi /quiet /norestart
"C:\Program Files\Fortinet\FortiClient\FCConfig.exe" -m vpn -f "%currentpath%\Config_Forti.conf" -o importvpn -i 1

echo Installation de Chrome...
start /w C:\Source\ChromeStandaloneSetup64.exe

rem start /w C:\Users\%username%\Desktop\install\ccsetup561.exe /S

if %TypeConfig%==1 (
    echo Installation de TS1000...
    start /w C:\Source\TS1000.exe /S
)

REM start C:\Users\%username%\Desktop\install\webexapp.msi
REM PAUSE

REM start C:\Users\%username%\Desktop\install\webexplugin.msi
REM PAUSE

echo Installation de Acrobat...
start /w C:\Source\acrobat.exe /sAll

start /w C:\Source\CitrixWorkspaceApp.exe /silent /norestart

setlocal
set SITE_TOKEN=eyJ1
set MSI_PATH=C:\Source\SentinelInstaller_windows_64bit_v22_3_5_887.msi
msiexec /i "%MSI_PATH%" /quiet /norestart SITE_TOKEN="%SITE_TOKEN%"
```

L'extrait du script ci-dessus a pour but de copier les fichiers d'installation de tous les logiciels nécessaires dans un dossier « Source » créé à la racine de l'ordinateur et étant présent pour récupérer les icones ou les fichiers d'installation en cas de problème. Ensuite le script lance l'installation de tous les logiciels de manière silencieuse tout en faisant leur configuration si nécessaire.



```
echo ===== Creation d'un compte admin77 local et création du compte adapei77 puis suppression du compte ASF

net user %UserName% adapei77 /add /passwordchg:no
net localgroup "Utilisateurs avec pouvoir" %UserName% /add
net localgroup Utilisateurs %UserName% /delete
wmic useraccount where Name='adapei77' set PasswordExpires=FALSE
net user ASF /delete
```

L'extrait ci-dessus permet de créer un compte utilisateur faisant partie du groupe de privilèges « Utilisateurs avec pouvoir » qui ne peut pas changer son mot de passe et dont le mot de passe n'expire pas (en sachant qu'un compte administrateur est créé précédemment dans le script).

4. Participation à la création d'un script Excel permettant de trier une liste d'employés

J'ai pris part à la création d'un script Excel avec Jules permettant de trier un fichier contenant la liste des employés de l'association. Ce script nous a permis de filtrer les informations et de ne conserver que les employés nécessitant la création d'un compte utilisateur, simplifiant ainsi le processus de gestion des comptes.

 Export_Sal_Octime.xlsx	26/06/2023 19:52	Feuille de calcul ...	195 Ko
 Script_Tri_Comptes.xlsm	26/06/2023 21:31	Feuille de calcul ...	24 Ko

Comme le montre la capture d'écran ci-dessus, le script est organisé en deux fichiers, le premier « Export_Sal_Octime » contient la liste de tous les employés de l'association avec leurs informations, le second fichier « Script_Tri_Comptes.xlsm » est un fichier Excel contenant uniquement le script.

```
' Spécifiez le chemin d'accès complet du fichier source XLSX
Set fichierSource = Workbooks.Open(ThisWorkbook.Path & "\Export_Sal_Octime.xlsx")

' Spécifiez le nom de la feuille source
Set feuilleSource = fichierSource.Sheets("Export_Sal_Octime")
```

L'extrait de script précédent ci-dessus permet d'indiquer au script qu'il doit récupérer les informations du premier fichier « Export_Sal_Octime ».

```
' Copie la première ligne vers la feuille de destination
feuilleSource.Rows(1).Copy Destination:=feuilleDestination.Rows(1)
k = 2




' Parcours les lignes de la feuille source
For i = 2 To derniereligne
    ' Vérifie si la colonne N est vide, la colonne X ne contient pas "OUVRIERS EN ESAT",
    ' et si la date dans la colonne P n'est pas déjà passée (si la cellule n'est pas vide)
    If feuilleSource.Cells(i, 14).Value = "" And InStr(1, feuilleSource.Cells(i, 24).Value, "OUVRIER EN ESAT") = 0 _
    And (feuilleSource.Cells(i, 16).Value = "" Or feuilleSource.Cells(i, 16).Value >= Date + 7) Then
        ' Copie la ligne vers la feuille de destination
        feuilleSource.Rows(i).Copy Destination:=feuilleDestination.Rows(k)
        k = k + 1
    End If
Next i

' Supprime les lignes contenant "OUVRIERS EN ESAT" dans la colonne X
derniereligne = feuilleDestination.Cells(Rows.Count, 1).End(xlUp).Row
For j = derniereligne To 2 Step -1
    If InStr(1, feuilleDestination.Cells(j, 24).Value, "OUVRIER EN ESAT") > 0 Then
        feuilleDestination.Rows(j).Delete
    End If
Next j

' Supprime les doublons dans la colonne A de la feuille de destination
derniereligne = feuilleDestination.Cells(Rows.Count, 1).End(xlUp).Row
Set listeDoublons = CreateObject("Scripting.Dictionary")
For j = derniereligne To 2 Step -1
    valeur = feuilleDestination.Cells(j, 1).Value
    If Not listeDoublons.exists(valeur) Then
        listeDoublons.Add valeur, True
    Else
        feuilleDestination.Rows(j).Delete
    End If
Next j
```

Cet extrait permet de trier les informations du fichier source, tout d'abord il copie le contenu de la première ligne (qui correspond au nom des colonnes), ensuite il vérifie si la case de chaque ligne se trouvant à la colonne X est vide (cette colonne contient l'adresse mail professionnelle des employés donc si l'employé possède déjà une adresse mail cela signifie qu'il possède déjà un compte utilisateur).

Une fois les informations triées, le script les copie dans un nouveau fichier au format csv nommé « Tri_Compte_a_Creer.csv » qui sera ensuite réutiliser dans un script PowerShell permettant de créer les comptes utilisateur.

 Export_Sal_Octime.xlsx	26/06/2023 19:52	Feuille de calcul ...	195 Ko
 Script_Tri_Comptes.xlsm	26/06/2023 21:31	Feuille de calcul ...	24 Ko
 Tri_Compte_a_Creer.csv	10/07/2023 14:29	Fichier CSV Micro...	4 Ko

5. Participation à la création d'un script PowerShell pour la création de comptes utilisateurs

J'ai également contribué à la création d'un script PowerShell qui permet de créer des comptes utilisateurs sur un serveur Active Directory ainsi que sur le Jamespot de l'association via un API. Ce script a automatisé la création des comptes et a facilité la gestion des utilisateurs au sein de l'association.

```
# Importer le module PowerShell pour les requêtes Web
Import-Module -Name Microsoft.PowerShell.Utility

[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

$CSVFile = "\\SRV-FICHIERS\homedir$\administrateur\Bureau\Test_PS_Fonctionnel\Add_New_AD_Users.csv"
$CSVData = Import-CSV -Path $CSVFile -Delimiter "," -Encoding UTF8

# Integre le nom du module et sa clé dans le Header de l'URL
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("X-Com-Jamespot-Module-Name", "EXT-")
$headers.Add("X-Com-Jamespot-Module-Key", " ")
```

L'extrait ci-dessus permet de récupérer les informations du fichier CSV créé précédemment via le script Excell et de créer une variable contenant les informations de connexions nécessaires pour l'utilisation de l'API de Jamespot.

```
foreach ($Utilisateur in $CSVData) {
    $UtilisateurPrenom = $Utilisateur."Prenom du salarie"
    $UtilisateurNom = $Utilisateur."Nom du salarie"

    $UtilisateurLogin = ($UtilisateurPrenom.Substring(0, 1)) + $UtilisateurNom
    $UtilisateurMotDePasse = " "
    $UtilisateurSpotMotDePasse = " "

    $UtilisateurFonction = $Utilisateur."Libelle qualification non conventionnelle"
    $UtilisateurLieu = $Utilisateur."Libelle Dossier"
    $UtilisateurTerritoire = $Utilisateur."Territoire"
    $UtilisateurDateNaissance = $Utilisateur."Date de Naissance"
```

L'extrait ci-dessus est une boucle permettant de récupérer les informations importantes de chaque ligne du fichier CSV pour les ajouter dans des variables qui seront ensuite réutilisées lors de la création des comptes Active Directory comme nous pouvons le voir ci-dessous :

```
if($CreationCompte -eq "True") {
    $UtilisateurEmail = "$UtilisateurLogin@adapei77.fr"
    Write-Host "Création de l'utilisateur : $UtilisateurLogin ($UtilisateurNom $UtilisateurPrenom)"
    $GroupeUtilisateur = ($Groupe_Privileges + "_" + $UtilisateurLieu)
    New-ADUser -Name "$UtilisateurNom $UtilisateurPrenom" `
        -DisplayName "$UtilisateurNom $UtilisateurPrenom" `
        -GivenName $UtilisateurPrenom `
        -Surname $UtilisateurNom `
        -SamAccountName $UtilisateurLogin `
        -UserPrincipalName "$UtilisateurLogin@adapei77.fr" `
        -EmailAddress $UtilisateurEmail `
        -Title $UtilisateurFonction `
        -Path "OU=$OU_AD,OU=$UtilisateurLieu,OU=Test,OU=Adapei,DC=adapei77,DC=fr" `
        -AccountPassword(ConvertTo-SecureString $UtilisateurMotDePasse -AsPlainText -Force) `
        -ChangePasswordAtLogon $true `
        -Enabled $true `
        -Description $UtilisateurFonction
    Add-ADGroupMember -Identity $GroupeUtilisateur -Members "$UtilisateurLogin"
    $dirPath = "\\srv-fichiers\homedir$\$UtilisateurLogin"
    $dirPath = $dirPath.ToLower()
```

```

if ($CompteDansAD -eq "True") {
    Write-Host "Création du compte Le Spot de $UtilisateurPrenom $UtilisateurNom..."

    # Url permettant de faire la requête à l'API
    $Uri = "https://lespot.adapei77.fr/api/api.php?o=user&f=create" + # L'option "o" permet de choisir dans quelle catégorie sera effectuée
    "&Mail=$UtilisateurEmail" + # Ceci est la première option permettant d'ajouter une adresse mail à l'utilisateur en cours de création
    "&Password=$UtilisateurMotDePasse" + # Attribution d'un mot de passe pour le compte
    "&Language=fr" + # Attribution de la langue principale du compte
    "&Pseudo=$UtilisateurPrenom $UtilisateurNom" + # Ajout du pseudo du compte sous la forme : Prenom NOM
    "&Country=fr" + # Attribution du pays
    "&Role=User" + # Attribution du type de compte (Administrateur / Utilisateur)
    "&Firstname=$UtilisateurPrenom" + # Ajout du prénom
    "&Lastname=$UtilisateurNom" + # Ajout du nom de famille
    "&dateOfBirth=$UtilisateurDateNaissance" + # Ajout de la date de naissance
    "&Function=$UtilisateurFonction" + # Ajout de l'emploi occupé par le propriétaire du compte (ex : Surveillant de Nuit)
    "&tag_3=$UtilisateurTerritoire" + # Ajout du territoire où se trouve le propriétaire du compte
    "&Company=$UtilisateurLieu" + # Ajout du nom de l'établissement où se trouve le propriétaire du compte
    "&PhoneNumber=$TelephoneEtablissement" + # Ajout du numéro de cet établissement
    "&autoPublish=1" +
    "&welcomeMail=1" + # Permet l'envoi d'un mail lors de la création du compte pour initialiser le mot de passe
    "&v=2.0" # Définit la version de l'API utilisée (cette version permet d'utiliser les mêmes nom de paramètres que dans le fichier CSV

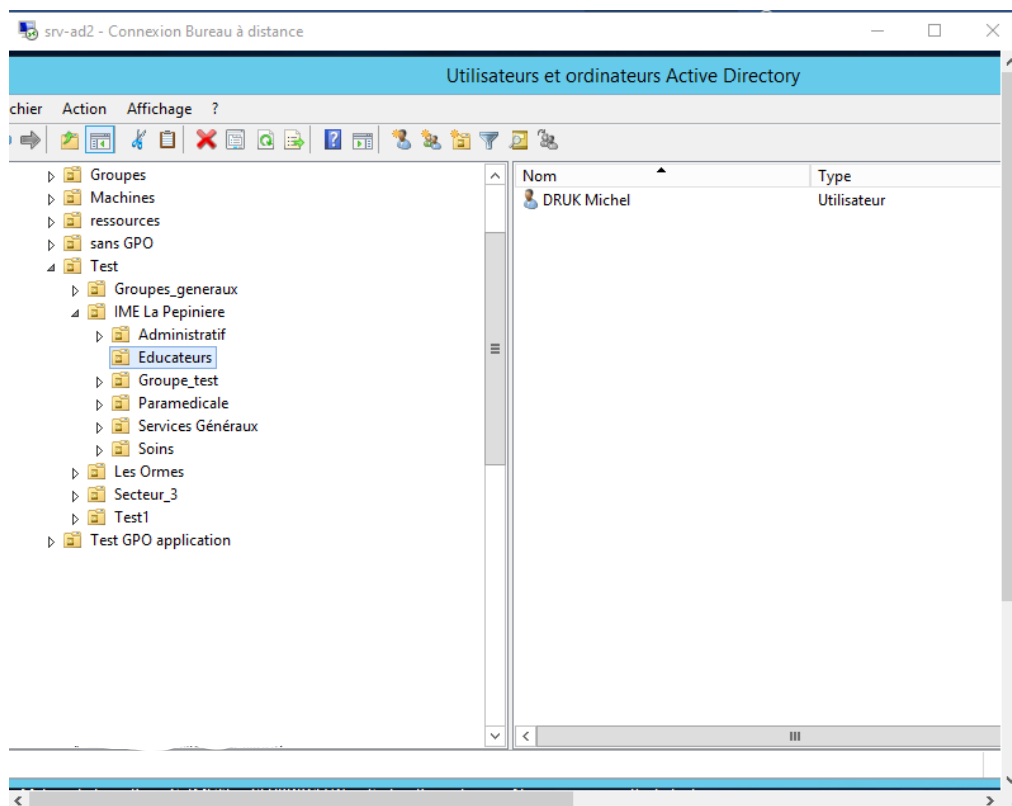
    # Fait la requête à l'API
    Invoke-RestMethod -Uri $Uri -Headers $headers -ContentType "application/json"
}

```

Enfin, nous pouvons ci-dessus la manière dont les requêtes de création de compte sont envoyées à l'API de Jamespot.

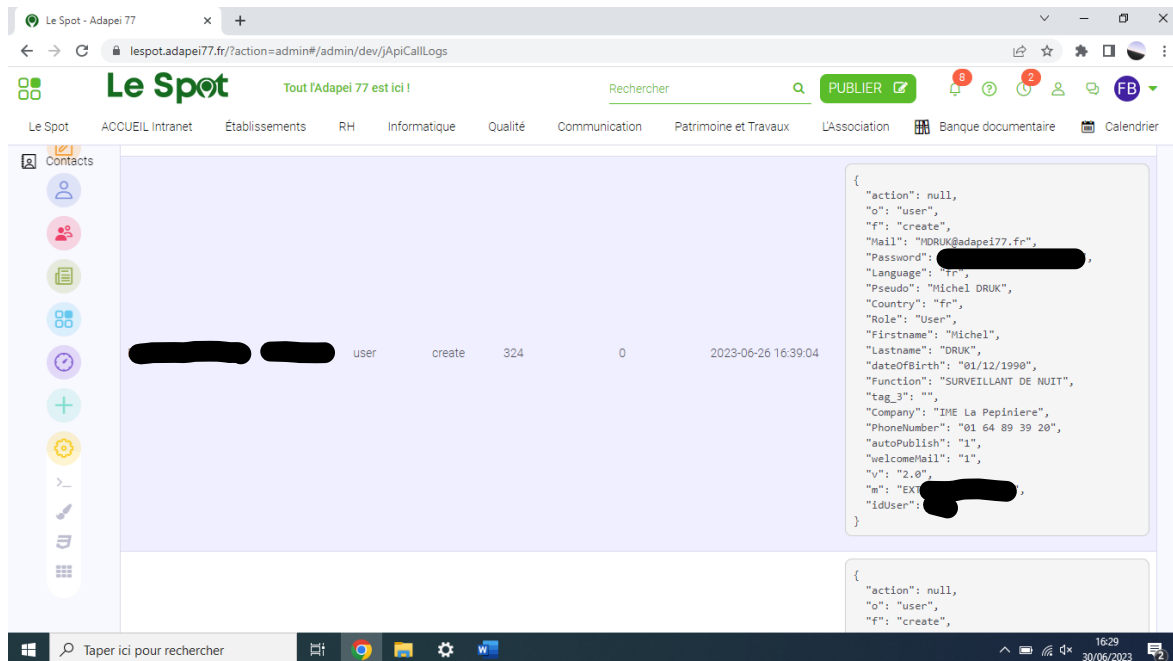
Tout d'abord, une variable contenant l'URL de l'API ainsi que toutes les options de créations de compte est créée pour ensuite être réutilisée dans la commande « Invoke-RestMethod » tout en y ajoutant le contenu de la variable créée au début du script contenant les informations de connexion à l'API ajoutées dans l'en-tête de la requête HTTPS.

Environnement de test pour la création des comptes AD :

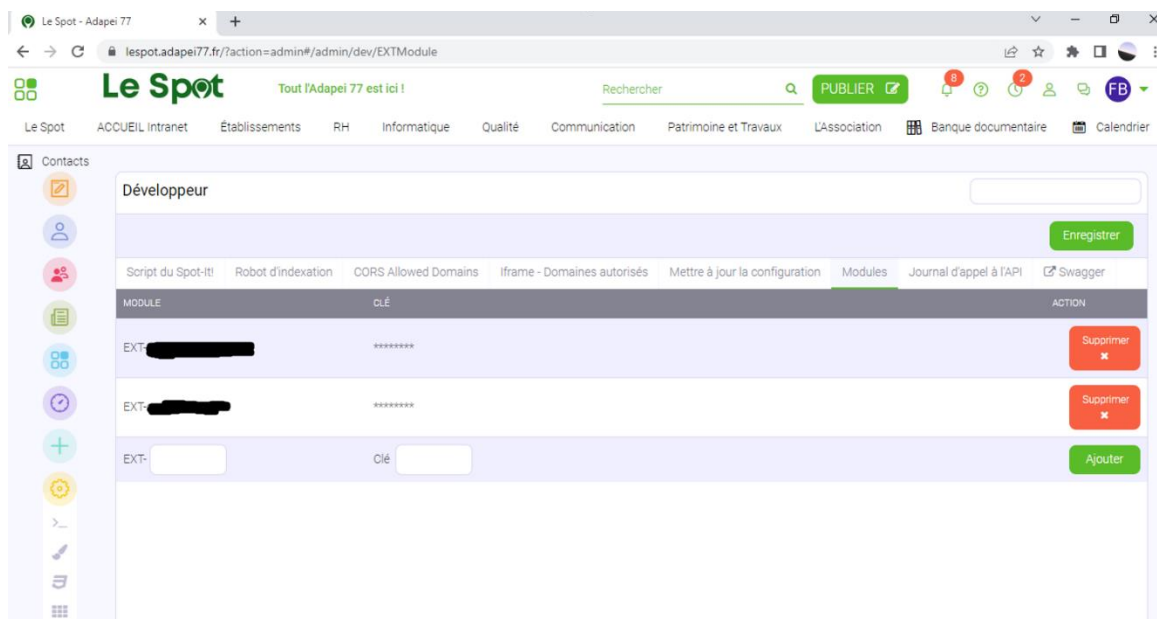


Un environnement a été créé sur le serveur AD de l'association pour tester le script de création de compte utilisateur AD. Cet environnement contient des Unités d'Organisation étant organisées d'une manière similaire à l'environnement de production de l'Adapei77.

API de Jamespot :



Ci-dessus nous pouvons voir un exemple de requête de création de compte reçu par l'API de Jamespot



L'image ci-dessus correspond à la page où se trouve les modules permettant de se connecter à l'API, pour la création des comptes j'ai donc dû créer un nouveau module qui est le deuxième visible pour lequel j'ai dû également créer une clé permettant d'y faire des requêtes. (Cela correspond donc aux informations ajouter dans la variable « header » au début du script PowerShell).

6. Participation aux interventions sur site en Seine et Marne

En plus des tâches de développement de scripts, j'ai eu l'opportunité de participer à des interventions sur site dans les établissements de l'association situés en Seine et Marne.