Compte Rendu des Activités de stage

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

1. Création et amélioration de scripts PowerShell

Attribution de Signature de mail sur Outlook 2016

L'association où j'ai effectué mon stage utilise une signature de mail pour les employés, laquelle s'appliquait automatiquement. Si les employés voulaient utiliser une signature, ils devaient soit copier celle d'un collègue, soit la recréer eux-mêmes, ce qui pouvait créer un manque d'uniformité. J'ai donc créé un script PowerShell permettant d'attribuer une signature personnalisée à chaque employé en utilisant les informations présentes dans leur compte Active Directory, en se basant sur un modèle au format HTML.

Explication des étapes du script :

- Création d'un fichier de log dans le dossier : .\Log\

```
# Création fichier de logs
$DateLog = Get-Date -Format "dd_MM_yyyy_HH_mm_ss"
$NomFichier = "Signature_Outlook-$DateLog.txt"
$Log = ".\Logs\$NomFichier"
New-Item -path $Log -ItemType File -Force
```

- Récupération de la liste des « homedir »

Les **homedir** sont nommés à partir du nom d'utilisateur donc récupérer leur nom permet de récupérer les noms d'utilisateurs.

*Le fait de se basé sur le nom des **homedir** permet d'éviter que le script tente d'appliquer la signature à un compte qui ne possède pas de **homedir** comme un compte de service, cela limite donc les tentatives inutiles.*

```
# Définir le chemin du dossier contenant les homedirs
$cheminHomedirs = "\\SRV-FICHIERS\homedir$\"

# Obtenir la liste des dossiers dans le chemin spécifié
$homedirs = Get-ChildItem $cheminHomedirs -Directory
```

- Récupération des infos de chaque utilisateur et vérification de leur présence dans l'AD

```
foreach ($homedir in $homedirs) {

    $Login = $homedir.Name
    # Récupérer les informations du compte AD de l'utilisateur
    $Utilisateur = Get-ADUser -Identity $Login -Properties DisplayName, HomeDirectory, OfficePhone, StreetAddress, Department
    $cheminSignature = "\\SRV-FICHIERS\homedir$\$Login\AppData\Roaming\Microsoft\Signatures"

    # Vérifier si l'utilisateur existe dans AD et s'il a un homedir défini
    if ((Test-Path $cheminSignature) -and $null -ne $Utilisateur) {
```

- Récupération des infos des utilisateurs

```
        $Nom = $Utilisateur.DisplayName
        $Telephone = $Utilisateur.OfficePhone
        $Adresse = $Utilisateur.StreetAddress
        $Mail = $Utilisateur.Mail
        $Etablissement = $Utilisateur.Department
        $Fonction = $Utilisateur.Title
        $CodePostale = $Utilisateur.PostalCode
        $Ville = $Utilisateur.City
```

- Le chemin vers le fichier HTML modèle et celui vers l'image à intégrer dans la signature sont nécessaires. L'image est convertie en Base 64, ce qui permet son intégration directe dans le code HTML du mail. Cette méthode évite la nécessité de placer l'image sur un serveur et d'insérer un lien vers celle-ci.

```powershell
# Chemin vers le modele de signature
$modelePath = ".\Ressources\Signature_Outlook-2016.html"

$CheminImage1 = ".\Ressources\Images\$Image1"
$ConvertionImage1 = [Convert]::ToBase64String((Get-Content -Path $CheminImage1 -Encoding Byte))
```

- Récupération du contenu du fichier HTML et remplacement des valeurs {Nom}, {Fonction}, {Etablissement}... par les valeurs correspondantes

```powershell
$modele = Get-Content $modelePath -Encoding UTF8

$signature = $modele -replace '\{Nom\}', $Nom -replace '\{Tel\}', $Telephone -replace '\{Mail\}', $
```

- Création du fichier de signature

Le contenu du fichier HTML modifié est intégré dans un fichier au format .htm se trouvant dans le dossier :

\\srv-fichiers\homedir$**<homedir>**\AppData\Roaming\Microsoft\Signatures\

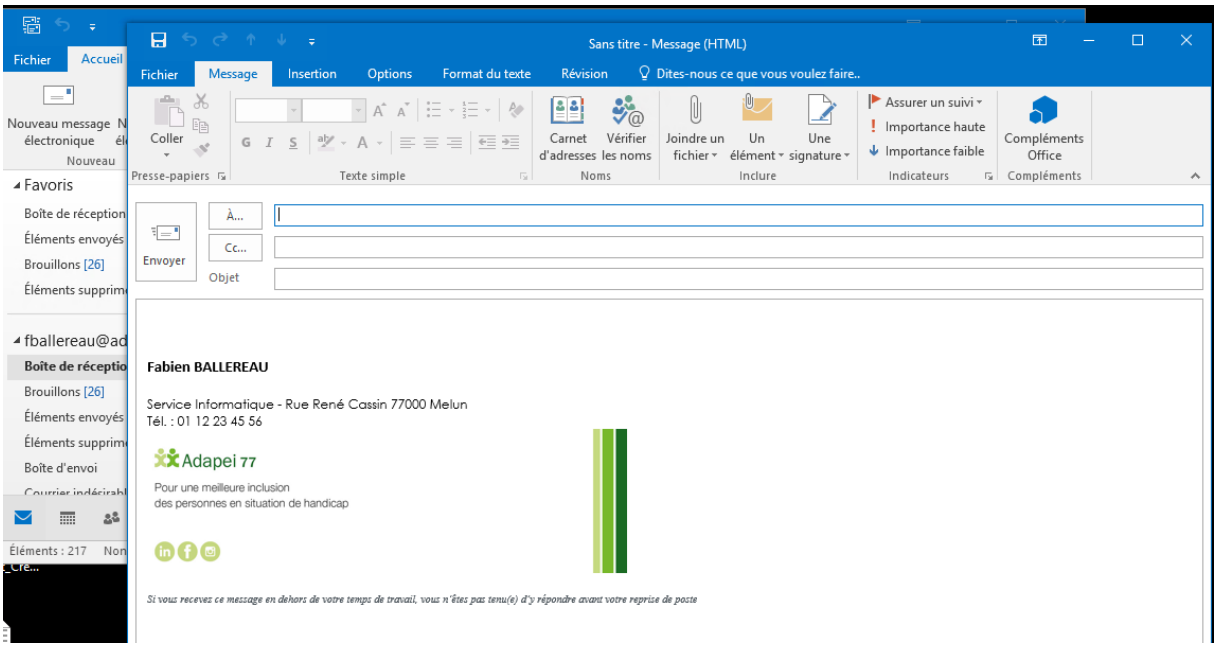
```powershell
$signaturePath = Join-Path $cheminSignature "Signature_$Nom.htm"
$signature | Out-File -FilePath $signaturePath -Encoding UTF8
```

Modèle HTML utilisé :

```html
        .droit-deconnexion {
            font-size: 7.5pt;
            font-style: italic;
            color: rgb(71, 80, 87);
        }

        p {
            margin: 0;
            line-height: 1.2;
        }
    </style>
</head>
<body>
    <p class="calibri"><strong>{Nom}</strong></p>
    <br>
    <p class="century-gothic">{Fonction}</p>
    <p class="century-gothic">{Etablissement} - {Adresse} {CodePostale} {Ville}</p>
    <p class="century-gothic">Tél. : {Tel}</p>
    <p><a href="http://www.adapei77.org" style="text-decoration: none;"><img src="Image1"></a></p>
    <br>
    <p class="droit-deconnexion">Si vous recevez ce message en dehors de votre temps de travail, vous n'êtes pas tenu(e) d'
</body>
</html>
```

Résultat de l'exécution du script :



Attribution de Signature de mail sur Outlook Web App

L'Adapei77 utilise également Outlook Web App de Exchange comme client de messagerie. J'ai donc créé un second script PowerShell permettant d'y attribuer une signature personnalisée. Il était essentiel que cette signature soit la plus proche possible de celle d'Outlook 2016 pour garantir une bonne uniformité.

Explication des étapes du script :

- Création d'un fichier de log

```
$DateLog = Get-Date -Format "dd_MM_yyyy_HH_mm_ss"
$NomFichier = "Signature_Webmail-$DateLog.txt"
$Log = ".\Logs\$NomFichier"
New-Item -path $Log -ItemType File -Force
```

- Informations de connexion à Exchange

```
# Infos de connexion au serveur Exchange
$Admin_AD = "adapei77\Administrateur"
$MDPAdmin_AD = ConvertTo-SecureString "[redacted]" -AsPlainText -Force

# Creation d'une session Remote Powershell vers le Exchange Management Shell
$credentials = New-Object System.Management.Automation.PSCredential($Admin_AD, $MDPAdmin_AD)
$Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://[redacted]/PowerShell/
```

- Récupération du contenu de la liste des employés

```
$CSVFile = "\\[redacted]\Export_Sal_Octime.csv"

$CSVData = Import-CSV -Path $CSVFile -Delimiter ";" -Encoding Default
```

- Conversion de l'image en Base 64

L'image est convertie en Base 64, ce qui permet son intégration directe dans le code HTML du mail. Cette méthode évite la nécessité de placer l'image sur un serveur et d'insérer un lien vers celle-ci.

```
$CheminImage1 = ".\Ressources\Images\$Image1"
$ConversionImage1 = [Convert]::ToBase64String((Get-Content -Path $CheminImage1 -Encoding Byte))
```

- Analyse du fichier ligne par ligne afin d'intégrer les lignes des employés possédant un identifiant dans une variable

```
    foreach ($Employe in $CSVData) {
        $Login = $Employe.Identifiant

        # Verifie que l'employe possede un identifiant
        if ($null -ne $Login) {
                $Utilisateur = Get-ADUser -Identity $Login -Properties officephone, streetaddress, d

                $Tableau += $Utilisateur
        }
    }
```

- Ouverture de la session PowerShell à distance vers le serveur Exchange

```
Import-PSSession $Session -DisableNameChecking
```

- Analyse de la variable et récupération des informations AD des employés

```
    foreach ($Ligne in $Tableau) {
        $Nom = $Ligne.DisplayName
        $Telephone = $Ligne.OfficePhone
        $Adresse = $Ligne.StreetAddress
        $Fonction = $Ligne.Title
        $Mail = $Ligne.Mail
        $Etablissement = $Ligne.Department
        $CodePostale = $Ligne.PostalCode
        $Ville = $Ligne.City
```

- Récupération du contenu du fichier HTML et remplacement des valeurs {Nom}, {Fonction}, {Etablissement}… par les valeurs correspondantes

```
        $modele = Get-Content $modelePath -Encoding UTF8

        # Ajoute les informations de l'utilisateur dans le fichier HTML de la signature
        $signature = $modele -replace '\{Nom\}', $Nom -replace '\{Tel\}', $Telephone -replace '\{Mail\}', $Mail -replac
```

- Application de la signature

```
Set-MailboxMessageConfiguration -Identity "$Mail" -SignatureHTML "$signature"
-AutoAddSignature $true
```

La signature est appliquée, l'utilisateur n'a plus qu'à créer un nouveau mail sur le Webmail pour voir apparaitre la nouvelle signature.

Difficulté rencontrée :

L'une des principales difficultés que j'ai rencontrées était que Exchange ne prend pas en charge le HTML de la même manière qu'Outlook 2016 ; il utilise une mise en forme plus "classique". Pour comprendre le fonctionnement de la gestion du HTML sur Exchange, j'ai utilisé une signature attribuée par défaut sur mon profil sur OWA, puis je me suis connecté au serveur Exchange afin d'exporter cette signature au format HTML en utilisant la commande PowerShell suivante :

```
$Utilisateur = "<Login>"
$BoiteAuxLettres = Get-Mailbox -Identity $Utilisateur
$SignatureHTML = $BoiteAuxLettres | Get-MailboxMessageConfiguration | Select-
Object -ExpandProperty SignatureHTML
```

Voici une partie du code HTML :

```
<p class="MsoNormal" style="color:rgb(0,0,0); font-family:Calibri,Helvetica,sans-serif; font-size:12pt">
<span style="font-size:9.0pt; font-family:&quot;Century Gothic&quot;,sans-serif; color:black">{Fonction}</span></p>
<p class="MsoNormal" style="color:rgb(0,0,0); font-family:Calibri,Helvetica,sans-serif; font-size:12pt">
<span style="font-size:9.0pt; font-family:&quot;Century Gothic&quot;,sans-serif; color:black">{Etablissement} - {Adresse} {CodePostale} {Ville}</span></p>
<p class="MsoNormal" style="color:rgb(0,0,0); font-family:Calibri,Helvetica,sans-serif; font-size:12pt">
<span style="font-size:9.0pt; font-family:&quot;Century Gothic&quot;,sans-serif; color:black">Tél. : {Tel}</span></p>

<p class="MsoNormal" style="color:rgb(0,0,0); font-family:Calibri,Helvetica,sans-serif; font-size:12pt">
<i><span style=""></span></i></p>

<div>
<p class=""x_MsoNormal"" style=""margin:0cm 0cm 0.0001pt; font-size:11pt; font-family:Calibri,sans-serif; color:rgb(33,33,33)"">
<a href="http://www.adapei77.org" class="OWAAutoLink">
<span style=""font-family:&quot;Century Gothic&quot;,sans-serif,serif,EmojiFont; color:blue""><img originalsrc=""cid:image002.jpg@01DA52C2.9CABF8A0"" src="""" bor
<p class=""x_MsoNormal"" style=""margin:0cm 0cm 0.0001pt; font-size:11pt; font-family:Calibri,sans-serif; color:rgb(33,33,33)"">
<b><span style=""font-size:8pt; font-family:&quot;Arial Narrow&quot;,sans-serif,serif,EmojiFont; color:rgb(0,176,80)""></span></b></p>
</div>

<br>

<p class="MsoNormal" style="color:rgb(0,0,0); font-family:Calibri,Helvetica,sans-serif; font-size:12pt">
<span style="font-size:7.5pt; color:rgb(71,80,87)"><i>Si vous recevez ce message en dehors de votre temps de travail, vous n'êtes pas tenu(e) d'y répondre avant
<p style="color:rgb(0,0,0); font-family:Calibri,Helvetica,sans-serif; font-size:12pt">
</p>
```

On peut voir que des « class » sont utilisée dans de nombreuses balises, celle-ci sont en fait utilisé directement par Exchange afin d'attribuer une mise en forme précise à la signature.
Par exemple, pour que l'image possède un lien HyperText il a fallu utiliser la class « OWAAutoLink ».

Amélioration du script de création de Comptes utilisateur

Lors de mon premier stage au sein de l'Adapei77, j'ai eu l'occasion de créer un script PowerShell permettant d'automatiser la création de comptes utilisateurs pour les employés. J'ai donc profité de mon second stage pour grandement améliorer l'efficacité du script, tout en réduisant les tâches nécessaires à son exécution, au point qu'il puisse être automatisé et exécuté de manière régulière.

Pour fonctionner, le script a besoin de la liste des employés remplissant les conditions pour avoir un compte utilisateur. Auparavant, cette liste était triée à l'aide d'un script Excel VBA, ce qui nécessitait de convertir le fichier de départ de CSV à XLSX, puis de récupérer la liste triée pour la reconvertir en CSV afin que le script puisse l'utiliser. Pour réduire le nombre d'étapes nécessaires à l'exécution du script PowerShell, j'ai décidé d'effectuer le tri via un second script PowerShell, qui envoie ensuite les informations au script de création de comptes.

Explication des étapes des scripts :

Partie A : Script de Tri

I. Vérification de la présence de la liste des employés

```powershell
Tri_Compte_a_Creer.ps1
$CSVFile = "\\[redacted]\Export_Sal_Octime.csv"

$Date = Get-Date -Format "dd_MM_yyyy"
$NomFichier = "CreationCompte-$Date.txt"
$Log = "\\SRV-FICHIERS\Informatique\04 - Projets\02 - Projets en
New-Item -path $Log -ItemType File -Force

if (Test-Path $CSVFile) {
```

II. Parcours le fichier CSV ligne par ligne pour créer un compte utilisateur aux employés concernés

Conditions à remplir pour la création d'un compte :

- Ne doit pas avoir de mail professionnel (example@adapei77.fr]
- La fonction ne doit pas être « OUVRIER EN ESAT » (la case ne peut pas être vide non plus)
- Il ne doit pas y avoir de date de fin de contrat ou alors elle doit être dans au moins 30 jours

```powershell
    foreach ($Employe in $CSVData) {
        $VerifMail = $Employe."EMAIL_PROF"
        $VerifEmploi = $Employe."Libellé qualification non conventionnelle"
        $VerifDate = $Employe."Date fin contrat"

        if (-not $VerifMail -and $VerifEmploi -notlike "*OUVRIER EN ESAT*" -and $VerifEmploi -
            if (-not ($Tableau."Matricule Salarié" -contains $Employe."Matricule Salarié")) {
                # Ajoute la ligne au tableau
                $Tableau += $Employe
            }
        }
    }
```

Si les conditions sont remplies, les informations sont ajoutées dans une variable de type tableau :

```powershell
        if (-not $VerifMail -and $VerifEmploi -notlike "*OUVRIER EN ESAT*" -and $VerifEmploi -
            if (-not ($Tableau."Matricule Salarié" -contains $Employe."Matricule Salarié")) {
                # Ajoute la ligne au tableau
                $Tableau += $Employe
            }
        }
```

III. Parcours la variable Tableau afin d'envoyer les informations au script de création de comptes

```
38      foreach ($Ligne in $Tableau) {
```

Récupération des informations nécessaires :

```
43          $Etablissement = $Ligne."Libellé Dossier"
44          $UtilisateurFonction = $Ligne."Libellé qualification non conventionnelle"
45          $Prenom = $Ligne."Prénom du salarié"
46          $Prenom = $Prenom.Substring(0,1).ToUpper() + $Prenom.Substring(1).ToLower()
47
48          $Nom = $Ligne."Nom du salarié"
49          $DateDeNaissance = $Ligne."Date de naissance"
50          $Contrat = $Ligne."Code Type de contrat"
51          $Expiration = $Ligne."Date fin contrat"
```

Vérification de la fonction (attribut une base de données Exchange et l'Unité d'Organisation adaptée)

```
55      if($UtilisateurFonction -eq "A.M.P.  NUIT MAS" -or $UtilisateurFonction -eq "accompagnant educatif et social Internat" -o
56      {
57          $UO_AD = "Educateurs"
58          $BDDExchange = "Accompagnement"
59      }
60      elseif($UtilisateurFonction -eq "PSYCHOLOGUE" -or $UtilisateurFonction -eq "PSYCHOMOTRICIEN(NE)" -or $UtilisateurFonction
61      {
62          $UO_AD = "Paramedicale"
63          $BDDExchange = "Accompagnement"
64      }
65      elseif($UtilisateurFonction -eq "AIDE SOIGNANT(E)   -  INTERNAT" -or $UtilisateurFonction -eq "AIDE SOIGNANT(E) - EXTERNA
66      {
67          $UO_AD = "Soins"
68          $BDDExchange = "Accompagnement"
69      }
70      elseif($UtilisateurFonction -eq "AGENT DE SERVICE INTERIEUR - EXT" -or $UtilisateurFonction -eq "AGENT DE SERVICE INTERIE
71      {
72          $UO_AD = "Services Généraux"
73          $BDDExchange = "Accompagnement"
74      }
75      elseif($UtilisateurFonction -eq "ASSISTANTE ADMINISTRATIVE" -or $UtilisateurFonction -eq "ASSISTANTE DE SERVICE SOCIAL")
76      {
77          $UO_AD = "Administratif"
78          $BDDExchange = "Administratif"
79      }
```

Vérification de l'Etablissement de l'employé

Utilisation de la condition « switch », cette condition permet de vérifier si la variable correspond à l'une des valeurs indiquées et si elle correspond alors le script attribuera la valeur désirée à la variable entre « { } », exemple :

```
107             "FRAIS DE SIEGE" {$UtilisateurLieu = "Siege"}
108             "ROZAY" {$UtilisateurLieu = "Rozay En Brie" ; $Territoire = "Territ
109             "FOYER LE CHENE ROUVRE" {$TelephoneEtablissement = "01 64 03 93 98"
110             "LE GINKGO BILOBA" {$TelephoneEtablissement = "01 60 22 20 14"; $Ut
111             "FOYER DE VIE LA MARGUETTE" {$TelephoneEtablissement = "01 64 36 23
112             "FOYER LES TILLEULS" {$TelephoneEtablissement = "01 64 63 55 30"; $
```

- Le nom de l'établissement est corrigé afin de correspondre au nom dans l'Active Directory

```
107                 "FRAIS DE SIEGE" {$UtilisateurLieu = "Siege"}
```

- Ajout du numéro de téléphone en fonction de l'établissement

```
109                 "FOYER LE CHENE ROUVRE" {$TelephoneEtablissement = "01 64 03 93 98";
```

- Ajout du territoire et du mail du Directeur d'Etablissement correspondant

```
$Territoire = "Territoire Marne et Morin"; $MailDA = "mduval@adapei77.fr"}
```

IV. Exécution du script de création de comptes

Si la fonction de l'employé a été reconnue alors le script de création de comptes est exécuté avec toutes les informations collectées en tant que paramètres.

```
131             $command ="& `"$ScriptCreationComptes`" -Prenom `"$Prenom`" -Nom `"$Nom`" -UtilisateurFonction
132             Write-Output "Utilisateur : $Prenom $Nom" | Out-File -FilePath "$Log" -Append
133             $output = Invoke-Expression $command
```

Partie B : Script de Création

Récupération des informations via les paramètres d'exécution (tous les paramètres sont obligatoires)

Creation_Comptes_Utilisateurs.ps1

```
1   param(
2       [Parameter(Mandatory=$true)]
3       [string]$Prenom,
4
5       [Parameter(Mandatory=$true)]
6       [string]$Nom,
7
8       [Parameter(Mandatory=$true)]
9       [string]$UtilisateurFonction,
```

Vérification de la présence d'un compte déjà existant

```
61  if (Get-ADUser -Filter {GivenName -eq $Prenom -and Surname -eq $Nom})
62  {
63      Write-Output "Attention : $Prenom $Nom possède déjà un compte sur l'AD."
64      $VerifUtilisateur = "True"
```

Si aucun compte n'est présent pour la personne en question mais qu'un compte possédant un nom d'utilisateur correspondant est trouvé, alors le script rajoute un point entre la première lettre du prénom et le nom de famille.

```
66  elseif (Get-ADUser -Filter {SamAccountName -eq $UtilisateurLogin})
67  {
68      Write-Output "Attention : L'identifiant $UtilisateurLogin existe déjà dans l'AD."
69      $UtilisateurLogin = ($Prenom.Substring(0, 1)) + "." + $Nom
```

Certains établissements ne sont pas organisés de la même manière dans le serveur Active Directory, certains ne possède pas d'UO « Educateurs », « Paramédicales »... et d'autres, leur UO ne se trouve pas dans une UO de Territoire.
Le script s'adapte donc à ces établissements :

```
88      $CheminUO = "OU=$UO_AD,OU=$UtilisateurLieu,OU=$Territoire"
89      switch($UtilisateurLieu) {
90      "Siege" {$CheminUO = "OU=$UtilisateurLieu"}
91      "Rozay En Brie" {$CheminUO = "OU=$UtilisateurLieu,OU=$UtilisateurTerritoire"}
92      "STR" {$CheminUO = "OU=$UtilisateurLieu,OU=$UtilisateurTerritoire"}
93      }
```

Ajoute de la date création du compte dans l'attribut « extensionAttributes1 » (évite le compte d'être supprimé par le script de suppression de comptes)

```
91          # Permet au compte de ne pas être supprimé tout de suite par le script de suppression
92          $DateC = "C=$date"
93          Set-ADUser -Identity $UtilisateurLogin -Add @{extensionAttribute1=$DateC}
```

Création du compte Active Directory

```
101     $ADUserParams = @{
102         'Name' = "$Nom $Prenom"
103         'DisplayName' = "$Nom $Prenom"
104         'GivenName' = $Prenom
105         'Surname' = $Nom
106         'SamAccountName' = $UtilisateurLogin
107         'UserPrincipalName' = $UtilisateurEmail
108         'EmailAddress' = $UtilisateurEmail
109         'Title' = $UtilisateurFonction
110         'Path' = "$CheminUO,OU=Utilisateurs Generale,OU=Adapei,DC=adapei77,DC=fr"
111         'AccountPassword' = (ConvertTo-SecureString $UtilisateurMotDePasse -AsPlainText -Force)
112         'ChangePasswordAtLogon' = $true
113         'Enabled' = $true
114         'Description' = $UtilisateurFonction
115     }
116
117     New-ADUser @ADUserParams
```

Le script ajoute les groupes correspondant à la fonction et à l'établissement de l'employé. Pour cela il récupère les groupes attribués à un compte Active Directory de référence.

```powershell
$UtilisateurReference = ($UtilisateurLieu + "_" + $UO_AD)

$GroupeReferences = Get-ADUser -Filter "UserPrincipalName -eq '$UtilisateurReference@adapei77.fr'" -Properties MemberOf | S

foreach ($groupe in $GroupeReferences) {
    Add-ADGroupMember -Identity $groupe -Members $UtilisateurLogin
}
```

Création d'un homedir$ et attribution des droits sur celui-ci

```powershell
    # Creer un répertoire du nom du login
    Write-Host "Création d'un répertoire..."
    New-Item -ItemType Directory -Path $dirPath | Out-Null
    $acl = Get-Acl -Path $dirPath
    $rule = New-Object System.Security.AccessControl.FileSystemAccessRule($UtilisateurLogin, "Ful
    $acl.AddAccessRule($rule) # Valide la nouvelle regle
    Set-Acl -Path $dirPath -AclObject $acl # Applique la regle
```

Création du Compte LeSpot via l'API de la plateforme

```powershell
    $uri = "https://lespot.adapei77.fr/api/api.php?o=user&f=create"` +
    "&Mail=$UtilisateurEmail"` +  # Ceci est la première option permetta
    "&Password=$UtilLeSpotMotDePasse"` +  # Attribution d'un mot de pass
    "&Language=fr"` +  #  Attribution de la langue principale du compte
    "&Pseudo=$Prenom $Nom"` +  # Ajout du pseudo du compte sous la forme
    "&Country=fr"` +  #  Attribution du pays
    "&Role=User"` +  # Attribution du type de compte (Administrateur / U
    "&Firstname=$Prenom"` +  # Ajout du prenom
    "&Lastname=$Nom"` +  # Ajout du nom de famille
    [redacted]
    "&Function=$UtilisateurFonction"` +  # Ajout de l'emploi occupé par
    "&tag_3=$Territoire"` +  # Ajout du territoire où se trouve le propr
    "&Company=$UtilisateurLieu"` +  # Ajout du nom de l'etablissement où
    "&PhoneNumber=$TelephoneEtablissement"` +  # Ajout du numéro de cet
    "&autoPublish=1"` +
    "&welcomeMail=1"` +  # Permet l'envoi d'un mail lors de la création
    "&v=2.0"  # Definit la version de l'API utilisee (cette version perm
```

Création d'une boîte mail via Exchange

(Le script ouvre une session PowerShell à distance vers le Exchange Management Shell pour exécuter des commandes Exchange)

```powershell
    $credentials = New-Object System.Management.Automation.PSCredential($Admin_AD, $MDPAdmin_AD)
    $Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://srv-mess.adapei77.fr/PowerShell/

    Write-Output "Création du compte Exchange..."
    Import-PSSession $Session -DisableNameChecking

    Enable-Mailbox -Identity $UtilisateurEmail -Database $BDDExchange

    Remove-PSSession $Session
```

Envoi d'un mail d'annonce de création du compte au Directeur d'Etablissement concerné et à l'Assistance Informatique

```powershell
        $smtpServer = "webmail.adapei77.fr"
        $To = $MailDA
        $CC = "assistance-informatique@adapei77.fr"
        $From = "assistance-informatique@adapei77.fr"
        $Subject = "Création du compte de $Prenom $Nom"
        $Body = "Bonjour, le compte de $Prenom $Nom a été créé.<br> Ces identifia

        $smtp = New-Object Net.Mail.SmtpClient($smtpServer)
        $msg = New-Object Net.Mail.MailMessage($From,$To,$Subject,$Body)
        $msg.CC.Add($cc)
        $msg.IsBodyHtml= $true
        $smtp.Send($msg)
```

Logs :

Le script créer un fichier de log se trouvant dans le chemin relatif par rapport au script : .\Logs\

```powershell
$DateLog = Get-Date -Format "dd_MM_yyyy_HH_mm_ss"
$NomFichier = "CreationCompte-$DateLog.txt"
$Log = ".\Logs\$NomFichier"
New-Item -path $Log -ItemType File -Force
```

2. Création d'une documentation pour chacun des scripts PowerShell

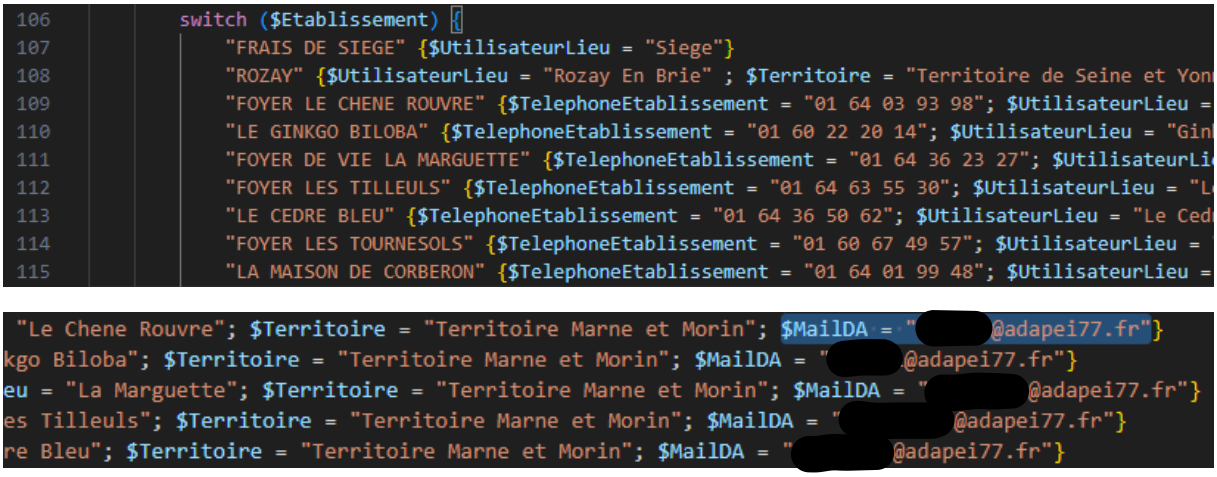
Les scripts que j'ai créés durant mes deux stages à l'Adapei77 comportent de nombreuses étapes, ce qui peut rendre leur compréhension complexe. Cependant, cela peut aussi les rendre plus sensibles aux éventuels changements dans l'infrastructure des serveurs.
J'ai donc rédigé une documentation pour chacun des scripts, expliquant chaque étape, comment les modifier en cas de changement dans l'infrastructure, et quelles erreurs peuvent être rencontrées ainsi que comment y réagir.

La partie expliquant les étapes du script a été reprise pour rédiger les explications sur les scripts dans la première partie.

Voici un exemple de la partie « Modification du script » d'une des documentations :

- Nouveau Directeur d'établissement

Il faut modifier le mail du DA dans la ligne correspondant à l'établissement et dans la variable « $MailDA »



- Nouvel Etablissement

Au même endroit que pour le changement de Directeur d'Etablissement, il faut ajouter une ligne correspondant à l'établissement dans la condition "switch". Dans le fichier contenant la liste des employés, les noms des établissements ne sont pas écrits de la manière dont le script en a besoin (c'est-à-dire la manière dont le nom est écrit dans l'AD).

Example :

"MOISSY" {$TelephoneEtablissement = ""; $UtilisateurLieu = "Moissy"; $Territoire = ""; $MailDA = ""}

- Nouvelle Fonction d'employé

Il faut ajouter une condition correspondant au nom exact de la fonction dans la partie ci-dessous (il faut savoir à quelle UO et a quelle base de données Exchange correspond la fonction afin de savoir ou ajouter la condition) :

```
55          if($UtilisateurFonction -eq "A.M.P.  NUIT MAS" -or $UtilisateurFonction -eq "accompagnant educatif et social Internat" -or $Util
56          {
57              $UO_AD = "Educateurs"
58              $BDDExchange = "Accompagnement"
59          }
60          elseif($UtilisateurFonction -eq "PSYCHOLOGUE" -or $UtilisateurFonction -eq "PSYCHOMOTRICIEN(NE)" -or $UtilisateurFonction -eq "
61          {
62              $UO_AD = "Paramedicale"
63              $BDDExchange = "Accompagnement"
64          }
65          elseif($UtilisateurFonction -eq "AIDE SOIGNANT(E)   -  INTERNAT" -or $UtilisateurFonction -eq "AIDE SOIGNANT(E) - EXTERNAT" -or
66          {
67              $UO_AD = "Soins"
68              $BDDExchange = "Accompagnement"
69          }
70          elseif($UtilisateurFonction -eq "AGENT DE SERVICE INTERIEUR - EXT" -or $UtilisateurFonction -eq "AGENT DE SERVICE INTERIEUR - IN
71          {
72              $UO_AD = "Services Généraux"
73              $BDDExchange = "Accompagnement"
```

Voici ce qui doit être rajouté :

```
if($UtilisateurFonction -eq "A.M.P.  NUIT MAS" -or $UtilisateurFonction -eq "accompagnant educatif et social Internat" -or $Utilisate
```

(Remplacer « accompagnant educatif et social Internat » par le nom de la fonction en question)

Exemple de la partie sur les potentiels erreurs :

001 – Le fichier CSV « Export_Sal_Octime.csv » n'a pas été trouvé.
- Vérifiez la présence du fichier dans le chemin : \\[redacted]\Export_Sal_Octime.csv
- Vérifiez si le fichier utilise « ; » comme délimiteur

002 – La fonction de l'employé n'a pas été reconnue.
- Vous pouvez ajouter la fonction dans la condition du script indiquée plus haut
- Vous pouvez modifier le nom de la fonction de l'employé dans le fichier CSV en indiquant un nom de fonction pris en charge par le script puis vous pourrez modifier le nom de la fonction dans les propriétés du compte Active Directory

003 – Il existe déjà un compte dont le nom utilisateur correspond à celui que le script tente d'attribuer au compte de l'employer mais il y a également un autre compte dont le nom d'utilisateur est le même mais avec un point qui sépare la première lettre du prénom et le nom. (ex : padider ; p.adider)
- Le compte ne peut pas être créé via le script

004 – La création du compte Active Directory a échouée.
- Vérifiez que le script ait bien été exécuté depuis un serveur Active Directory

Erreurs LeSpot :

Erreur de l'API de LeSpot – La création du compte LeSpot a échouée
- Le compte existe peut-être déjà sur LeSpot
- Il manque peut-être des informations dans l'URL de la requête

Erreur Powershell (impossible d'établir une connexion sécurisée SSL/TLS) : PowerShell n'est pas parvenu à envoyer une requête à LeSpot
- Le script doit être exécuté depuis un Windows Server 2019 au minimum

Erreurs Exchange :

New-PSSession : [[redacted].adapei77.fr] La connexion au serveur distant [redacted].adapei77.fr a échoué avec le message d'erreur suivant : Le certificat serveur de l'ordinateur de destination ([redacted].adapei77.fr:443) comporte les erreurs suivantes :

Le certificat SSL contient un nom commun qui ne correspond pas au nom d'hôte. Pour plus d'informations, voir la rubrique d'aide about_Remote_Troubleshooting.

- Aucun Certificat TLS n'a été trouvé

Vérifiez que le lien PowerShell vers le serveur Exchange commence bien par : http://

New-PSSession : [[redacted].adapei77.fr] La connexion au serveur distant srrv-mess.adapei77.fr a échoué avec le message d'erreur suivant : WinRM ne peut pas traiter la demande.

L'erreur suivante s'est produite lors de l'utilisation de l'authentification Kerberos: l'ordinateur [redacted].adapei77.fr est inconnu à Kerberos. Vérifiez que l'ordinateur existe sur le réseau, que le nom spécifié est correctement orthographié et que la configuration Kerberos pour l'accès à l'ordinateur est correcte. Le problème de configuration Kerberos le plus courant est qu'aucun SPN au format HTTP/[redacted].adapei77.fr n'est configuré pour la cible. Si Kerberos n'est pas nécessaire, spécifiez le mécanisme d'authentification Negotiate et soumettez à nouveau l'opération. Pour plus d'informations, voir la rubrique d'aide about_Remote_Troubleshooting.

- La connexion au serveur Exchange a échoué

Vérifiez que le lien http vers le serveur Exchange n'a pas changé (il est trouvable dans le Centre D'administration d'Exchange)

Vérifiez que la méthode d'authentification utilisé est Kerberos (et que le poste d'où est exécuté le script est bien dans le domaine)

$Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://[redacted].adapei77.fr/PowerShell/ -Authentication Kerberos -Credential $credentials

3. Brassage d'une baie informatique

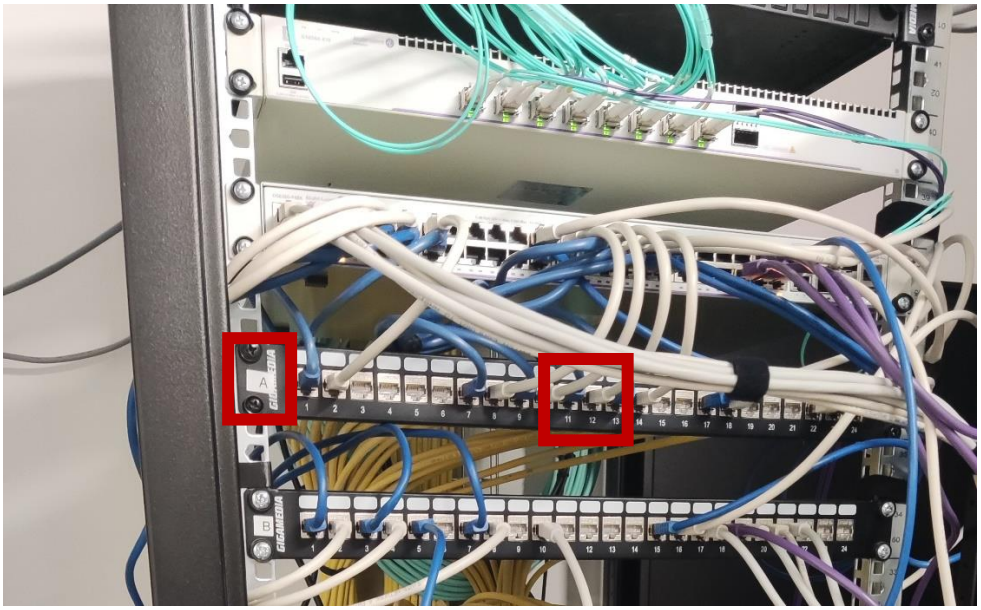
Durant mon stage à l'Adapei77, j'ai eu l'occasion de participer au brassage d'une baie informatique dans un nouvel établissement en fin de construction, j'ai également pu effectuer du « cable management » sur baie secondaire.

Brassage :

Pour effectuer le brassage nous devions tout d'abord repérer les numéros des prises RJ45 murales des salles du bâtiments comme on peut le voir ci-dessous :

Un fois que nous avions repéré le numéro de chaque prise (ici A12 et A11), nous repérions le port correspondant dans la baie que nous pouvons voir ci-dessous :

(Le cable management n'avait pas été effectué car des prestataires effectuaient des tests et faisaient du câblage également)



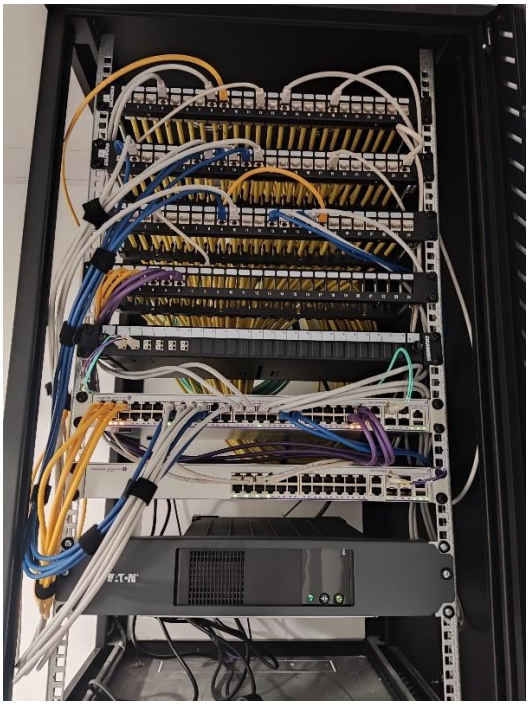
Une fois que nous avions repéré le port sur la baie, nous le relions au switch le plus proche en utilisant un câble RJ45 gris pour les ordinateurs et un câble RJ45 bleu pour les téléphones. Les ordinateurs étaient reliés aux ports RJ45 avec un numéro pair, tandis que les téléphones étaient reliés aux ports avec un numéro impair.

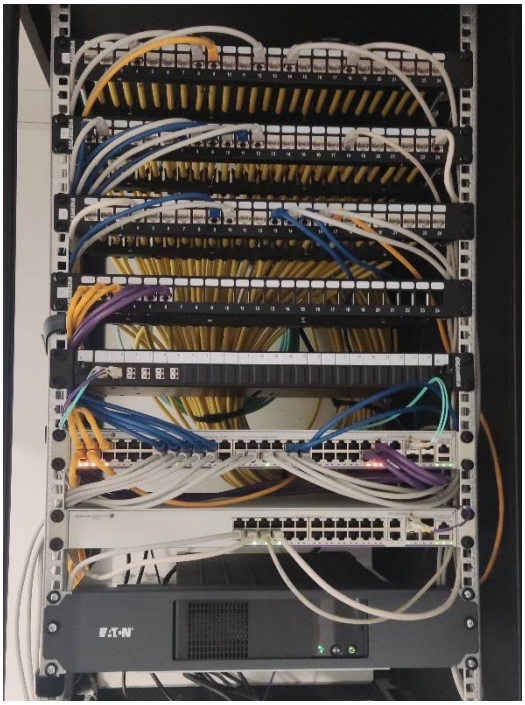
L’utilisation des câbles de couleurs et de la logique de pair et impair permettra à l’avenir de simplifier la maintenance de la baie.

Cable managament :

Comme dit précédemment, j’ai eu l’occasion d’effectuer du cable management sur un baie secondaire avec l’aide d’un second stagiaire.

Avant et après le cable management :



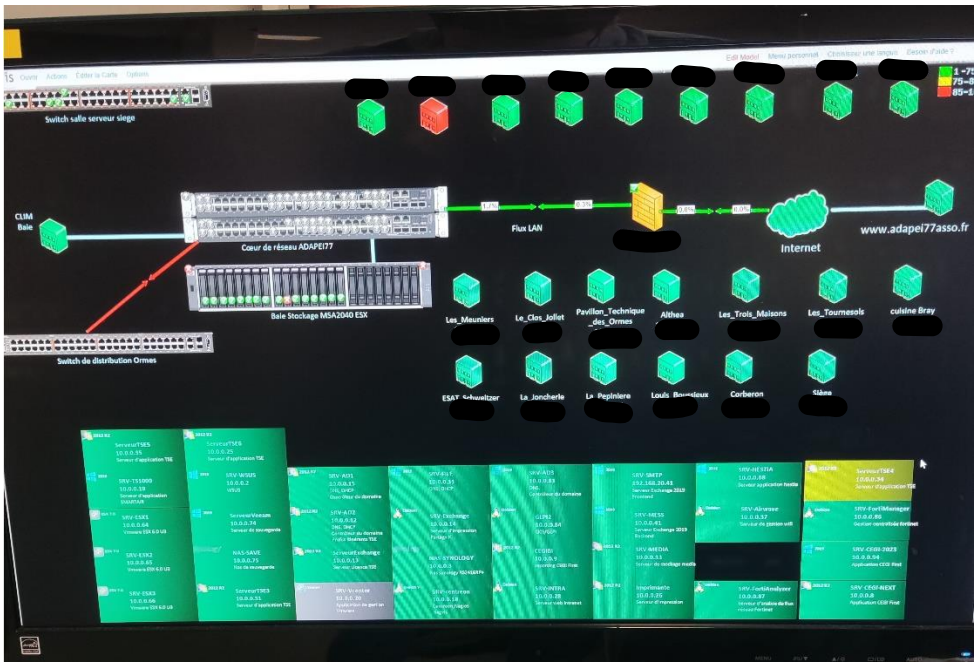


Vue de côté après le cable management :

Enfin, nous avons également installer des postes au niveau des prises qui ont été brassées.

4. Mise en place d'un serveur de supervision (Check_MK)

L'Adapei77 utilisait un serveur de supervision sous Centreon qui était vieillissant avec une interface web permettant de voir en temps réel l'état des différents serveurs et établissements comme nous pouvons le voir ci-dessous :



Cette interface est en fait un png avec des icones qui changent de couleur en fonction de l'état du serveur ou de l'établissement mais ce système rend toute modification complexe.

J'ai donc eu la mission de trouver un nouveau serveur qui devait remplacer le Centreon, celui-ci devait respecter certaines conditions :
- Gratuit
- Documentation complète
- Mise à jour simple et régulière
- Nécessitant le moins de ressources possible (Basé sur Linux)
- Interface moderne (si possible)

A la suite de nombreuses recherches et de comparaison entre les différents services existant je me suis donc tourné vers **Check_MK** qui est un service de supervision basé sur Nagios et qui respecte toutes les conditions indiquées précédemment, **il possède une offre gratuite très complète, les mises à jour sont simples à effectuer, la dernière version est utilisable sur Debian 12, l'interface est moderne et enfin, Check_MK possède une documentation très complète comprenant des vidéos explicatives sur son utilisation.**

<u>Mise en place de Check_MK :</u>

- Installation d'une VM Debian 12 sur VMWare
- Suivi des quelques étapes d'installation de la documentation :

Télécharger le paquet

wget https://download.checkmk.com/checkmk/2.2.0p23/check-mk-raw-2.2.0p23_0.bookworm_amd64.deb

Lancer l'installation

apt install ./check-mk-raw-2.2.0p23_0.bookworm_amd64.deb

Créer l'interface web de Check_MK

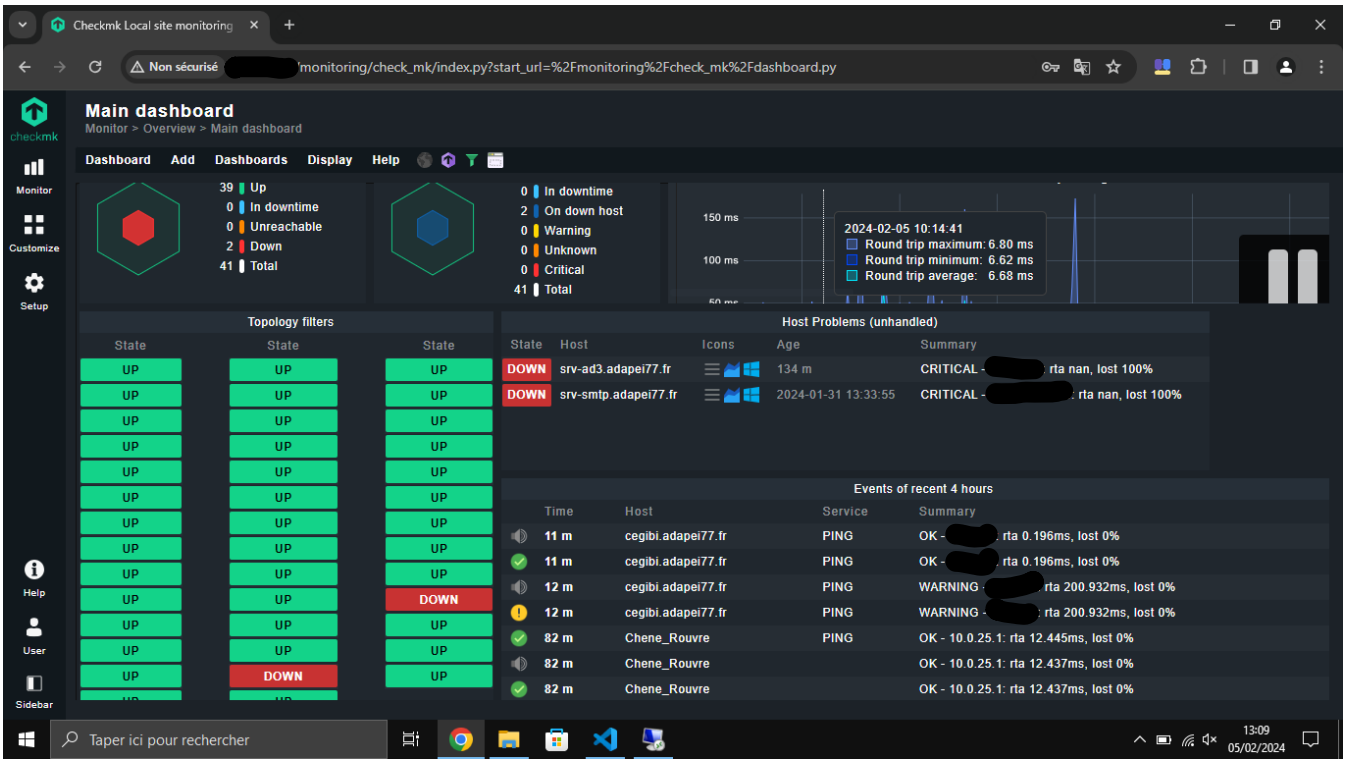
omd create monitoring

Lancer le service

omd start monitoring

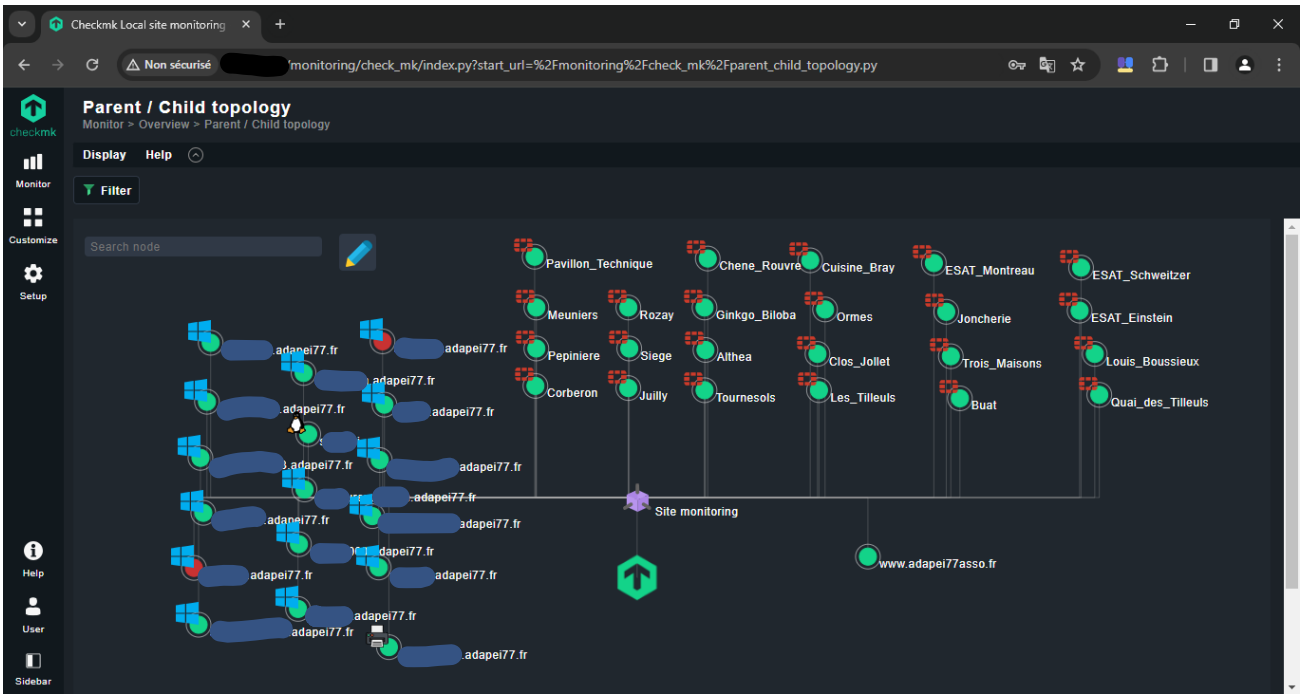
L'installation est terminée.

Dashboard de Check_MK



Le « Dashboard » est la page principale qui peut être entièrement personnalisée, on peut ajouter des graphiques sur l'utilisation de différents services ou encore une fenêtre indiquant l'état de toutes les machines.

Topologie du réseau



La fenêtre de topologie ci-dessus à pour but de remplacer celle de Centreon, son interface est plus sobre et elle est beaucoup simple à modifier.

5. Création d'un fichier recensant les actions de chaque GPO de l'infrastructure

L'Adapei77 utilise principalement Windows Server dans son infrastructure, ainsi que des sessions RDP pour les employés, au sein desquelles diverses actions sont effectuées via des GPO (plus de 50 au total). Cependant, bon nombre de ces GPO ont été créées il y a plusieurs années et effectuent plusieurs actions à la fois qui n'étaient pas référencées. **J'ai donc pris l'initiative de créer un fichier recensant chaque GPO et toutes les actions qu'elles effectuent.**

Ce recensement a permis de déterminer que certaines GPO n'étaient plus utiles, ce qui a facilité leur suppression. De plus, cela permettra également de régler plus facilement les divers problèmes pouvant survenir lors de l'utilisation de ces GPO.

Voici des exemples de description :

Restriction Powershell

- Interdit l'exécution des fichiers : pwsh.exe, powershell.exe, powershell_ise.exe pour les utilisateurs standards (session TSE)

SYS : Désactiver le gestionnaire de serveur

- Désactive l'affichage automatique du gestionnaire de serveur
(Le gestionnaire de serveur n'apparait pas de base sur les sessions TSE)

UO : Machines

SYS : NTP Clients

- Active le client NTP de Windows et le configure :

NtpServer	adapei77.fr
Type	NTP
CrossSiteSyncFlags	2
ResolvePeerBackoffMinutes	15
ResolvePeerBackoffMaxTimes	7
SpecialPollInterval	3600
EventLogFlags	0

UO : Computers

SYS : WSUS Postes

- Active l'utilisation de WSUS vers le lien : http://wsus.adapei77.fr:8530

Les parties surligner en jaune indique des GPO potentiellement inutiles par rapport à l'infrastructure actuelle de l'Adapei77 (les postes des utilisateurs ne sont plus dans le domaine donc une GPO pour WSUS n'est plus utile). Le fichier indique également dans quel UO s'applique la GPO