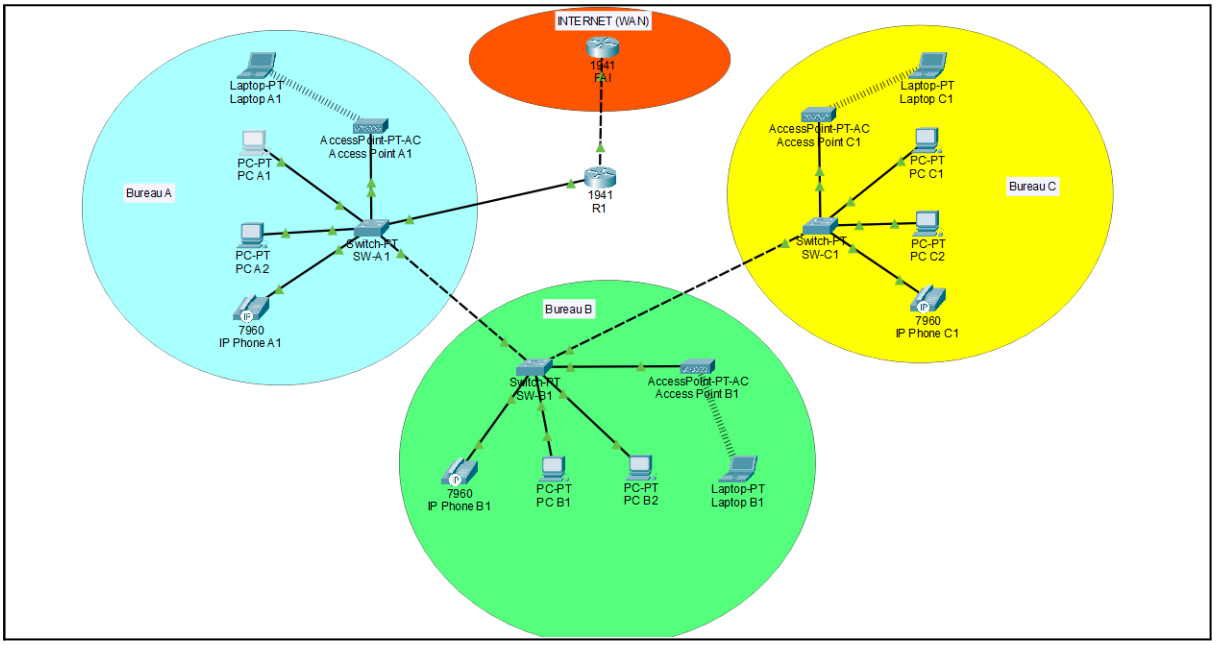
Test d’admission Msc Cyber

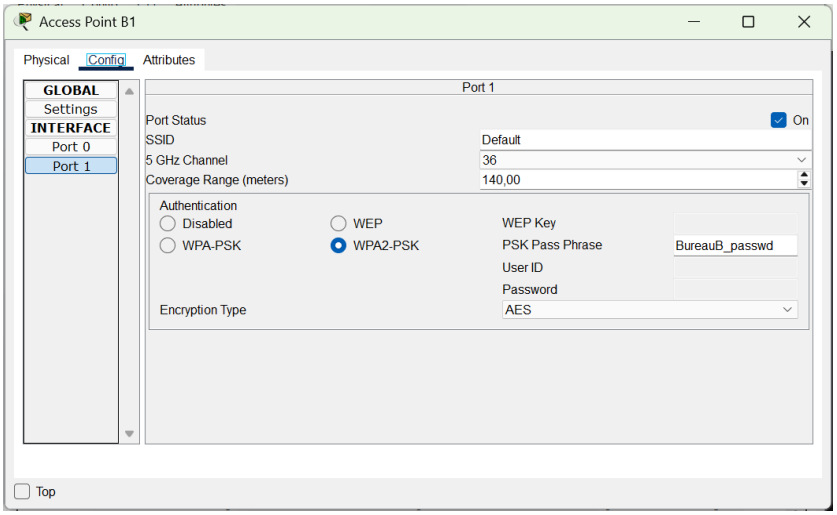
Travail réalisé par Paul Vanderhaegen le 10/12/2025

Exercice 1 : Packet Tracer - [lien Github de l’exercice]

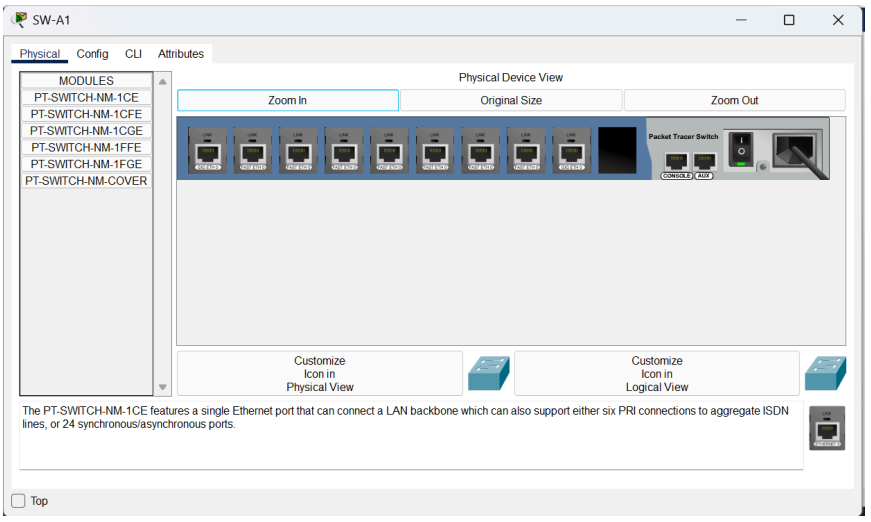


Pour cet exercice, j’ai commencé par installer tous les éléments physiques dans un fichier pkt vierge en définissant clairement les différents bureaux et j’ai relié les différents appareils pour former la topologie de l’infrastructure.

On sécurise nos points d'accès wi-fi en y ajoutant une pass phrase unique pour chaque bureau



J'ai ensuite ajouté différents modules sur les switchs afin d'obtenir 9 ports physique comme demandé en veillant à insérer des liaisons à 1gbps pour les connexions uplink vers le routeur.



Une fois les éléments disposés et reliés entre eux, j'ai commencé par appliquer une configuration de base à l'un des switchs en veillant à sécuriser les accès par un mot de passe chiffré

Configuration de base des switchs

```
en
conf t
hostname SW-XX
```

```
service password-encryption
enable secret cisco
no ip domain-lookup

line console 0
password cisco
login
end

line vty 0 15
password cisco
login
end

wr
```

Je crée ensuite les différents VLAN sur les switchs. Afin de respecter les bonnes pratiques et d'éviter de potentiels problèmes réseaux sur notre infrastructure, j'ai fait le choix d'utiliser le numéro 40 pour le VLAN VOIP. Dans un souci de cohérence, le réseau utilisé pour le VLAN VOIP sera le 192.168.40.0 /24 et sa plage DHCP ira de 192.168.40.10 à 192.168.40.50.

Création des VLANS

```
conf t
vlan 10
name PC_fixes
end

vlan 20
name Wi-Fi
end

vlan 30
name Administration
end

vlan 40
name VOIP
end
```

J’affecte ensuite les ports sur le switch aux différents VLAN. On notera que l’interface VLAN 1 est désormais désactivée pour des raisons de sécurité. Les interfaces du switch sont configurées en mode *access (*et en mode *voice* pour les interfaces reliées à un téléphone IP) à l’exception des interfaces 1/1 et 9/1 qui sont configurés en mode *trunk* et chargés de remonter les différentes trames des VLAN vers le routeur.

Configuration des interfaces

```
conf t

interface GigabitEthernet1/1
 description Liaison_uplink
 switchport mode trunk

interface FastEthernet2/1
 description VOIP
 switchport access vlan 40
 switchport voice vlan 40
 switchport mode access

interface FastEthernet3/1
 description VOIP
 switchport access vlan 40
 switchport voice vlan 40
 switchport mode access

interface FastEthernet4/1
 description Wi-Fi
 switchport access vlan 20
 switchport mode access

interface FastEthernet5/1
 description Wi-Fi
 switchport access vlan 20
 switchport mode access

interface FastEthernet6/1
 description PC_fixes
 switchport access vlan 10
 switchport mode access

interface FastEthernet7/1
 description PC_fixes
 switchport access vlan 10
 switchport mode access

interface FastEthernet8/1
 description Administration
```

```
 switchport access vlan 30
 switchport mode access

interface GigabitEthernet9/1
 description Liaison_uplink
 switchport mode trunk


interface Vlan1
 no ip address
 shutdown
```

On copie ensuite cette configuration sur les deux autres switchs.

Les switchs étant configurés, nous allons désormais configurer notre routeur en commençant par lui appliquer une configuration de base comme ceci.

Configuration de base du routeur

```
en
conf t
hostname R1
enable secret cisco
line console 0
password cisco
login
line vty 0 4
password cisco
login
transport input all
exit
service password-encryption
no ip domain-lookup
end
wr
```

Notre routeur possède ici deux interfaces: une interface LAN dédiée au réseau local et une interface WAN pour pouvoir naviguer sur Internet. Tous les VLAN étant amenés au routeur sur la seule interface LAN existante (Router-on-a-Stick), nous devons donc configurer des sous-interfaces pour chaque VLAN sur cet unique lien et renseigner à chaque fois l'adresse ip qui servira de passerelle pour ces VLAN.

Configuration du Routeur-on-a-Stick

```
interface GigabitEthernet0/1
no ip address
duplex auto
speed auto
!
interface GigabitEthernet0/1.10
description Passerelle_VLAN_PC_fixes
encapsulation dot1Q 10
ip address 192.168.10.254 255.255.255.0
ip nat inside
no shutdown
!
interface GigabitEthernet0/1.20
description Passerelle_VLAN_Wi-Fi
encapsulation dot1Q 20
ip address 192.168.20.254 255.255.255.0
ip nat inside
no shutdown
!
interface GigabitEthernet0/1.30
description Passerelle_VLAN_Administration
encapsulation dot1Q 30
ip address 192.168.30.254 255.255.255.0
ip nat inside
no shutdown
!
interface GigabitEthernet0/1.40
description Passerelle_VLAN_VOIP
encapsulation dot1Q 40
ip address 192.168.40.254 255.255.255.0
ip nat inside
no shutdown
!
interface Vlan1
no ip address
shutdown
```

On configure ensuite notre routeur pour agir en tant que serveur DHCP. On doit alors exclure les adresses ip qui ne seront pas incluses dans les différentes plages DHCP avant de créer nos différents “pools”

Configuration DHCP

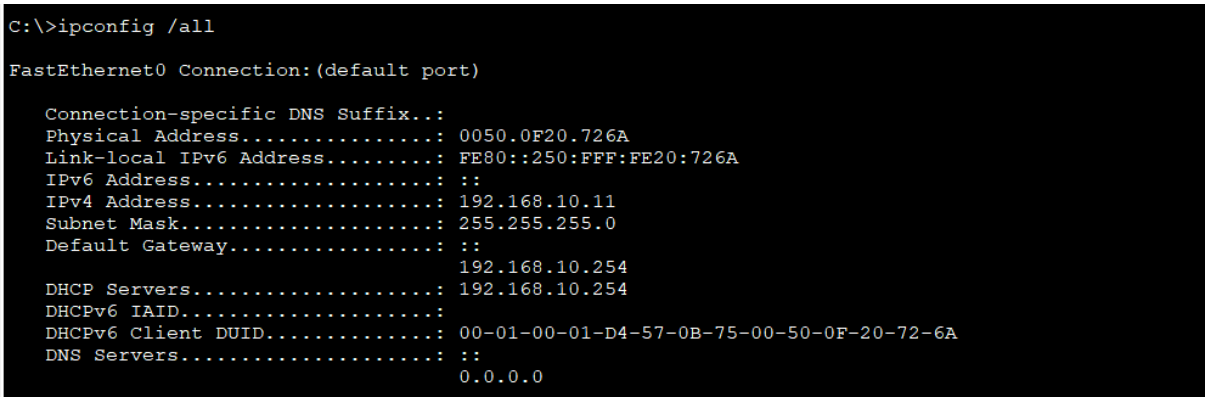
Exclusion des adresses
ip dhcp excluded-address 192.168.10.1 192.168.10.9
ip dhcp excluded-address 192.168.10.51 192.168.10.254
ip dhcp excluded-address 192.168.20.1 192.168.20.9
ip dhcp excluded-address 192.168.30.1 192.168.30.9
ip dhcp excluded-address 192.168.40.1 192.168.40.9
ip dhcp excluded-address 192.168.40.51 192.168.40.254
ip dhcp excluded-address 192.168.30.51 192.168.30.254
ip dhcp excluded-address 192.168.20.51 192.168.20.254

Création du pool DHCP
ip dhcp pool vlan_pc_fixes
network 192.168.10.0 255.255.255.0
default-router 192.168.10.254
ip dhcp pool vlan_wi-fi
network 192.168.20.0 255.255.255.0
default-router 192.168.20.254
ip dhcp pool vlan_administration
network 192.168.30.0 255.255.255.0
default-router 192.168.30.254
ip dhcp pool vlan_voip
network 192.168.40.0 255.255.255.0
default-router 192.168.40.254

On vérifie le bon fonctionnement de notre DHCP avec la commande *show ip dhcp pool*

```
R1#sh ip dhcp pool

Pool vlan_pc_fixes :
 Utilization mark (high/low)    : 100 / 0
 Subnet size (first/next)       : 0 / 0
 Total addresses                : 254
 Leased addresses               : 6
 Excluded addresses             : 8
 Pending event                  : none

 1 subnet is currently in the pool
 Current index        IP address range                    Leased/Excluded/Total
 192.168.10.1         192.168.10.1     - 192.168.10.254    6    / 8     / 254

Pool vlan_wi-fi :
 Utilization mark (high/low)    : 100 / 0
 Subnet size (first/next)       : 0 / 0
 Total addresses                : 254
 Leased addresses               : 3
 Excluded addresses             : 8
 Pending event                  : none

 1 subnet is currently in the pool
 Current index        IP address range                    Leased/Excluded/Total
 192.168.20.1         192.168.20.1     - 192.168.20.254    3    / 8     / 254
```

On vérifie directement sur les terminaux qu'ils obtiennent une adresse ip avec la commande *ip config /all*



Il nous reste à configurer le PAT (Port Adress Translation) pour que nos équipements puissent naviguer sur Internet via l'adresse publique de notre routeur que l'on va désormais configurer.

On attribue en premier lieu une adresse IPV4 publique à l'interface WAN qui sera l'adresse IP de notre NAT.

On crée ensuite une access-list (ACL) pour identifier le trafic interne en provenance de nos VLAN qui sera traduit par le NAT avant d'activer le PAT avec la mention *overload.*

On définit enfin une route par défaut pour diriger notre trafic vers l'extérieur.
Dans le cas présent, j'ai ajouté un routeur d'un FAI dont nous utiliserons l'adresse IP reliée à notre interface WAN comme prochain saut de notre routeur pour accéder à Internet.

Configuration de l'interface WAN

R1# interface GigabitEthernet0/0
R1# ip address 203.0.113.2 255.255.255.0
R1# ip nat outside

R1# access-list 1 permit 192.168.0.0 0.0.255.255

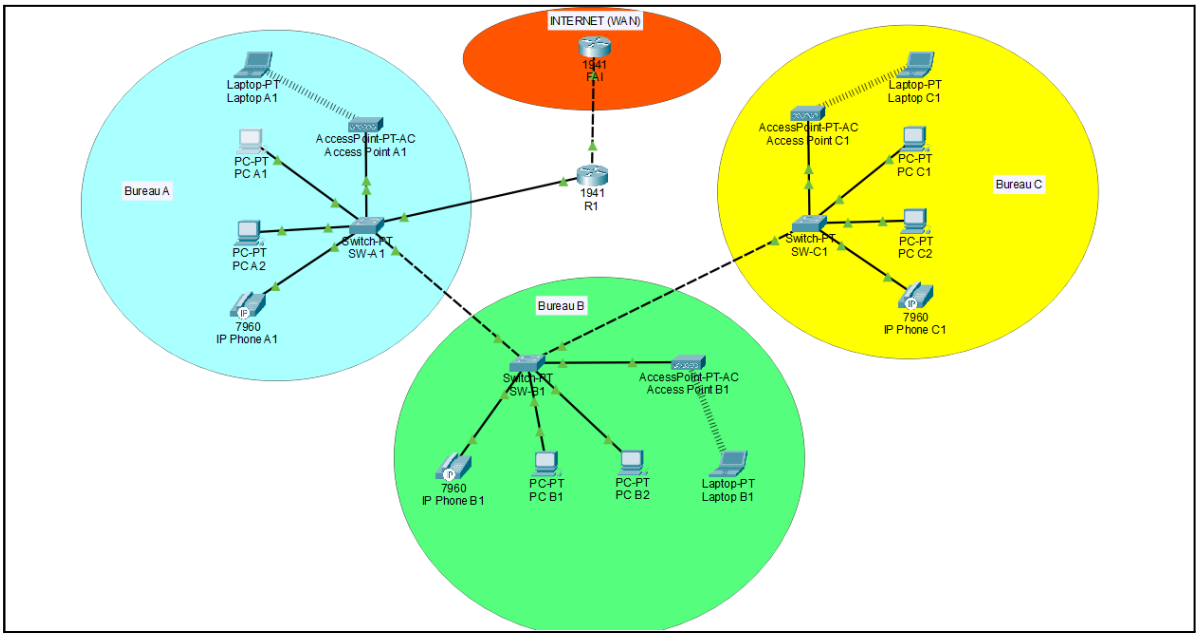
R1# ip nat inside source list 1 interface GigabitEthernet0/0 overload

R1# ip route 0.0.0.0 0.0.0.0 203.0.113.1 (adresse interface FAI)

La création de mon infrastructure étant achevée, j'ai alors effectué quelques tests de connectivité inter-VLAN et vers le WAN afin de m'assurer du bon fonctionnement de mon réseau.

```
C:\>ipconfig

FastEthernet0 Connection:(default port)

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.....: FE80::250:FFF:FE20:726A
   IPv6 Address.....: ::
   IPv4 Address.....: 192.168.10.11
   Subnet Mask.....: 255.255.255.0
   Default Gateway.....: ::
                                 192.168.10.254

Bluetooth Connection:

   Connection-specific DNS Suffix..:
   Link-local IPv6 Address.....: ::
   IPv6 Address.....: ::
   IPv4 Address.....: 0.0.0.0
   Subnet Mask.....: 0.0.0.0
   Default Gateway.....: ::
                                 0.0.0.0

C:\>ping 192.168.20.12

Pinging 192.168.20.12 with 32 bytes of data:

Reply from 192.168.20.12: bytes=32 time=25ms TTL=127
Reply from 192.168.20.12: bytes=32 time=23ms TTL=127
Reply from 192.168.20.12: bytes=32 time=22ms TTL=127
Reply from 192.168.20.12: bytes=32 time=22ms TTL=127

Ping statistics for 192.168.20.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 22ms, Maximum = 25ms, Average = 23ms

C:\>ping 203.0.113.1

Pinging 203.0.113.1 with 32 bytes of data:

Request timed out.
Reply from 203.0.113.1: bytes=32 time<1ms TTL=254
Reply from 203.0.113.1: bytes=32 time<1ms TTL=254
Reply from 203.0.113.1: bytes=32 time<1ms TTL=254
```

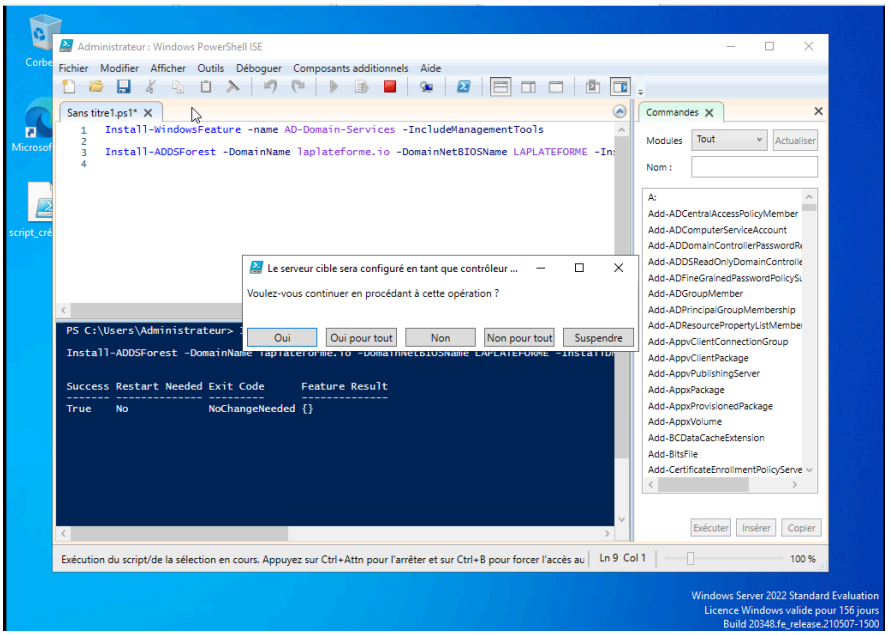
Vous trouverez ci-dessous le schéma complet de mon infrastructure



Exercice 2 : Active Directory - [lien Github de l'exercice]

J'ai commencé par installer une machine virtuelle Windows Server 2022 sur VMware Workstation à l'aide d'un fichier iso. Une fois mon Windows Server prêt à l'emploi, j'ai créé le domaine Active Directory laplateforme.io à l'aide d'un premier script en Powershell.

```
Install-WindowsFeature -name AD-Domain-Services -IncludeManagementTools

Install-ADDSForest -DomainName laplateforme.io -DomainNetBIOSName
LAPLATEFORME -InstallDNS:$true -SafeModeAdministratorPassword
(ConvertTo-SecureString "Azerty_2025!" -AsPlainText -Force)
```



Après avoir créé mon domaine et configuré le serveur en tant que contrôleur de domaine, on vient créer un fichier csv dénommé *AD_Users_LP.csv* contenant l'ensemble des utilisateurs ainsi que leurs groupes.

```
Nom,Prenom,Groupe1,Groupe2,Groupe3,Groupe4,Groupe5,Groupe6
ALEXANDRE,MARCELLINE,Animation,,,,,
ARAGON,ISABELLE,Animation,,,,,
AVARO,MARINA,As,Medical,,,,
BERNARD,ISABELLE,As,Medical,,,,
BOUFFIER,STEPHANE,ASH,Hebergement,,,,
BOUZIANE,FATIHA,cadre de sante,Cadres,Medical,,,
CARLIER,CHANTAL,Comptable,Administratif,,,,
THILLOT,MARC,Directeur,Cadres,Hebergement,Technique,Administratif,Animation
GALLIEN,CAROLE,Maitresse de Maison,Cadres,Hebergement,,,
GRIVEAUX,PATRICIA,Medecin,Cadres,Medical,,,
LARGUIER,SILVANIA,Psychologue,Cadres,Medical,,,
MALAURE,OPHYLANADRA,ASH,Hebergement,,,,
PRATABUY,MYRIAM,Secretaire,Administratif,,,,
SAHTIT,OIFAA,IDE,Medical,,,,
SALVADOR,GLADYS,IDE,Medical,,,,
SCHNEIDER,EMILE,Technique,Cadres,,,,
VIGNOLO,VERONIQUE,Animation,Cadres,Hebergement,Administratif,,
```

Je conçois alors un second script qui commence par charger les données du fichier *AD_Users_LP.csv* et inscrit dans une variable le mot de passe qu'auront par défaut tous les utilisateurs lors de leur première connexion ("Azerty_2025!").

Il collecte ensuite tous les noms de groupes uniques du fichier csv et vérifie leur existence dans l'AD avant de créer les groupes manquants.

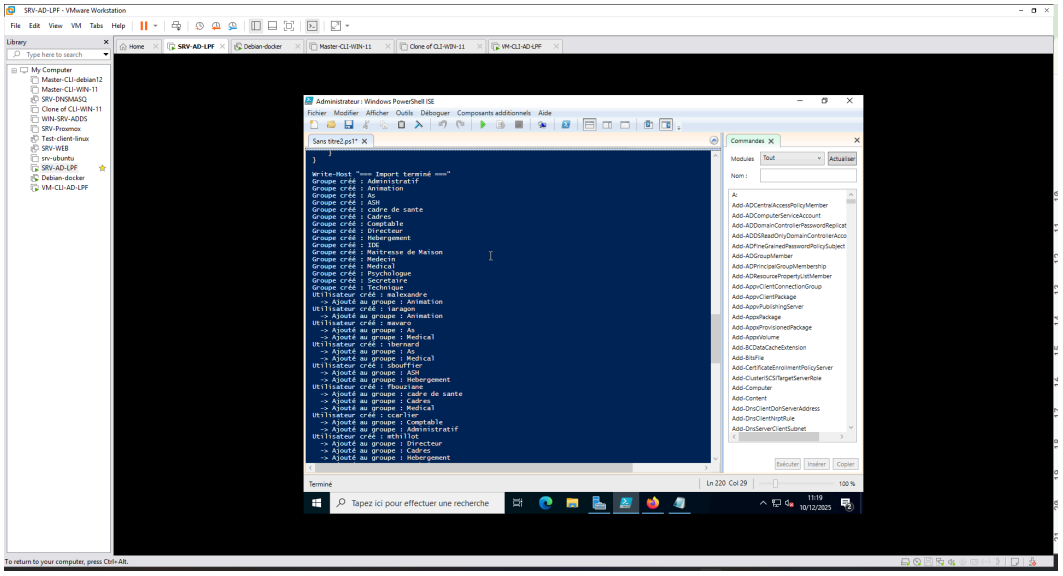
Il génère ensuite une boucle pour chaque utilisateur qu'il se charge de créer avec la commande *New-ADUser* avant de les doter de différents attributs tels que *"-ChangePasswordAtLogon $true"* afin de forcer le changement de mot de passe à la première connexion.

Il ajoute ensuite chaque utilisateur aux groupes auxquels il est rattaché.

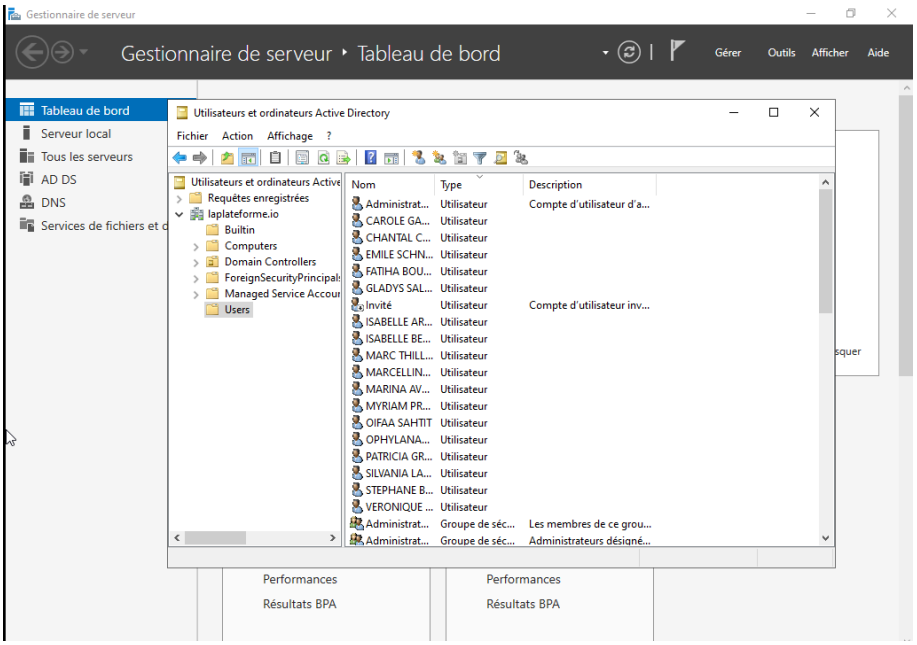
Enfin, la commande Write-Host nous permet de visualiser l'avancement des tâches et le bon fonctionnement du script.

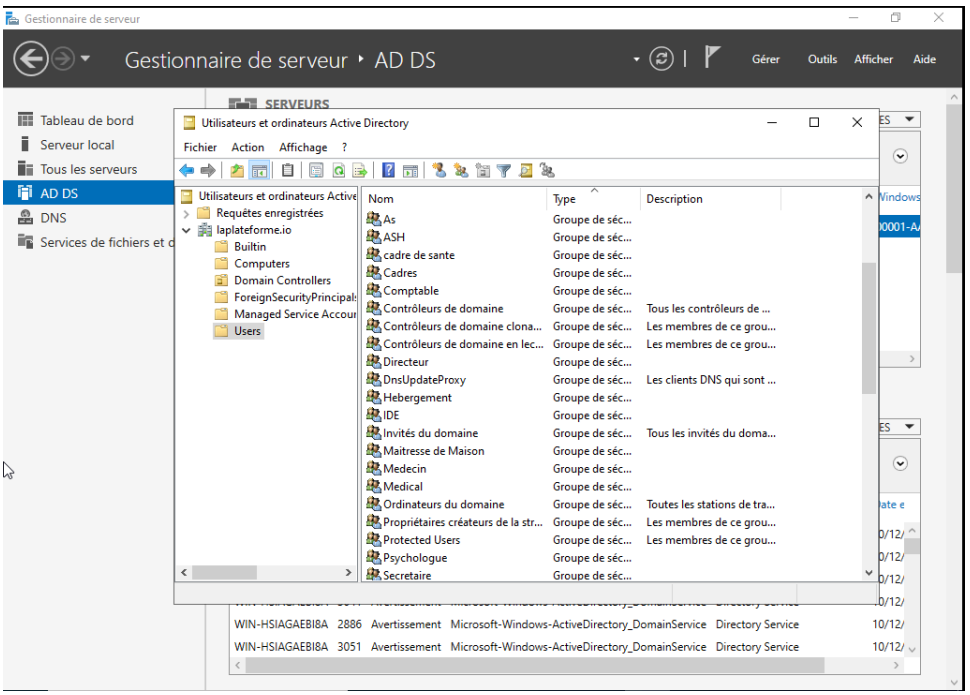
```
Import-Module ActiveDirectory

# Chargement du CSV
$users = Import-Csv -Path ".\AD_Users_LP.csv"

# Mot de passe par défaut pour tous
$password = ConvertTo-SecureString "Azerty_2025!" -AsPlainText -Force

# --- Création des groupes ---
$allGroups = @()

foreach ($u in $users) {
    $allGroups += $u.Groupe1, $u.Groupe2, $u.Groupe3, $u.Groupe4, $u.Groupe5,
$u.Groupe6
}

$allGroups = $allGroups | Where-Object { $_ -and $_ -ne "" } | Sort-Object -Unique

foreach ($g in $allGroups) {
    if (-not (Get-ADGroup -Filter "Name -eq '$g'" -ErrorAction SilentlyContinue)) {
        New-ADGroup -Name $g -GroupScope Global -Path
"CN=Users,DC=laplateforme,DC=io"
        Write-Host "Groupe créé : $g"
    }
}

# --- Création des utilisateurs + affectation aux groupes ---
foreach ($u in $users) {
```

```
    # Création du SamAccountName : 1ère lettre prénom + nom
    $sam = ($u.Prenom.Substring(0,1) + $u.Nom).ToLower()

    # Création de l'utilisateur si inexistant
    if (-not (Get-ADUser -Filter "SamAccountName -eq '$sam'" -ErrorAction
SilentlyContinue)) {

        New-ADUser `
            -Name "$($u.Prenom) $($u.Nom)" `
            -GivenName $u.Prenom `
            -Surname $u.Nom `
            -SamAccountName $sam `
            -UserPrincipalName "$sam@laplateforme.io" `
            -AccountPassword $password `
            -Enabled $true `
            -ChangePasswordAtLogon $true `
            -Path "CN=Users,DC=laplateforme,DC=io"

        Write-Host "Utilisateur créé : $sam"
    }

    # Récupération des groupes de l'utilisateur
    $groups = @(
        $u.Groupe1,
        $u.Groupe2,
        $u.Groupe3,
        $u.Groupe4,
        $u.Groupe5,
        $u.Groupe6
    ) | Where-Object { $_ -and $_ -ne "" }

    # Ajout aux groupes
    foreach ($g in $groups) {
        Add-ADGroupMember -Identity $g -Members $sam -ErrorAction SilentlyContinue
        Write-Host "  -> Ajouté au groupe : $g"
    }
}

Write-Host "=== Import terminé ==="
```

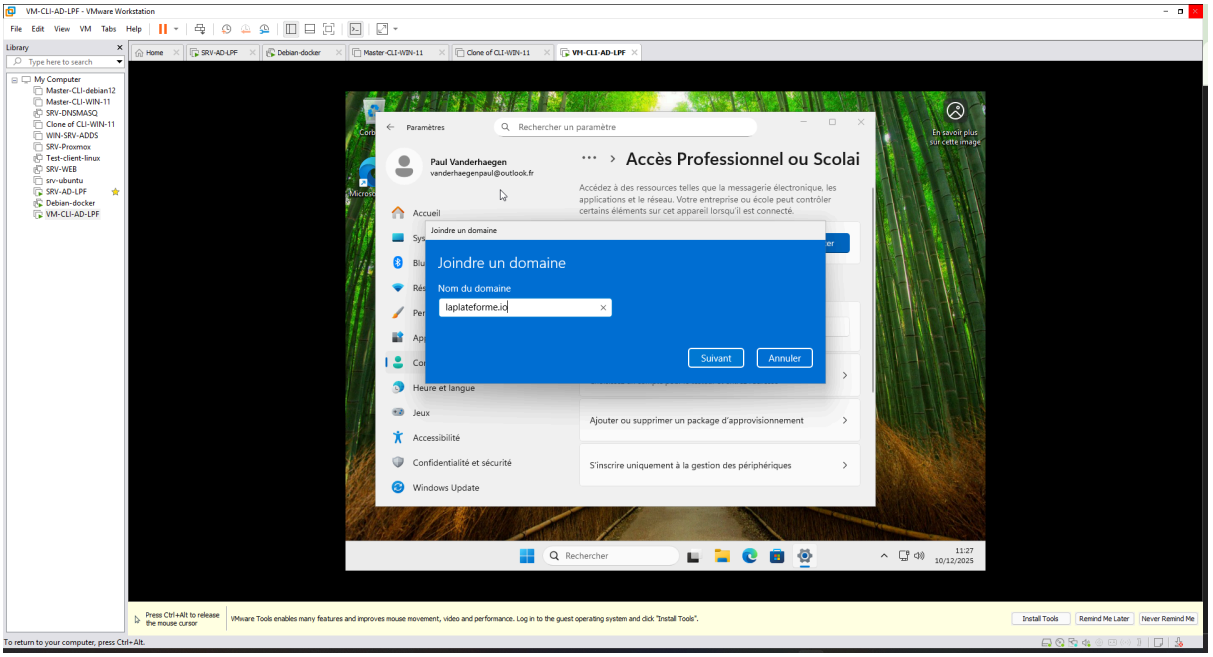


Après avoir exécuté le script, je vérifie sur l'interface graphique du serveur que mes groupes et utilisateurs sont bien présents dans le domaine laplateforme.io

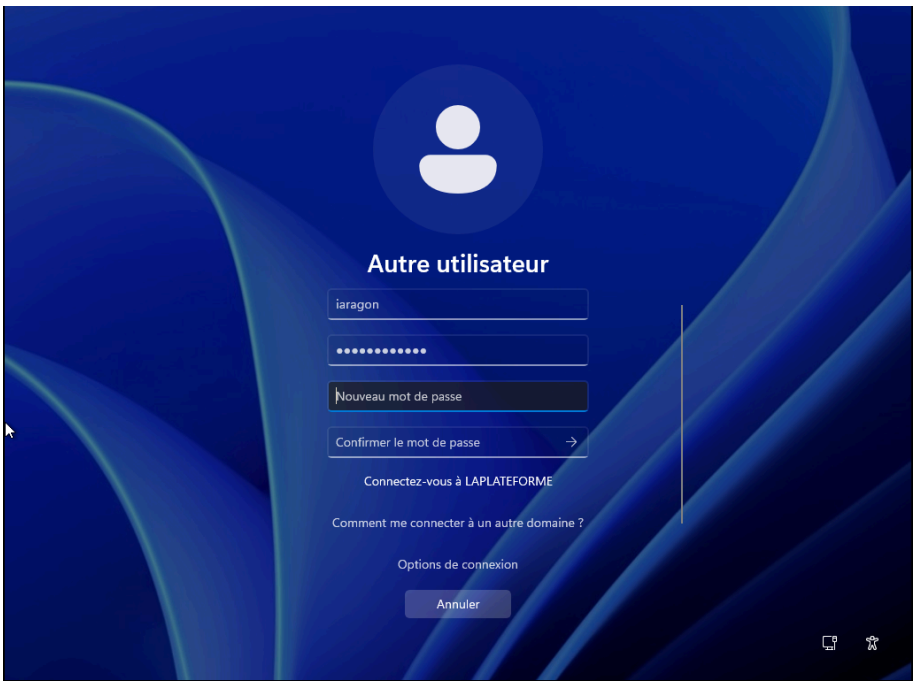




Je crée ensuite une machine cliente sous Windows 11 que je viens rattacher au domaine laplateforme.io

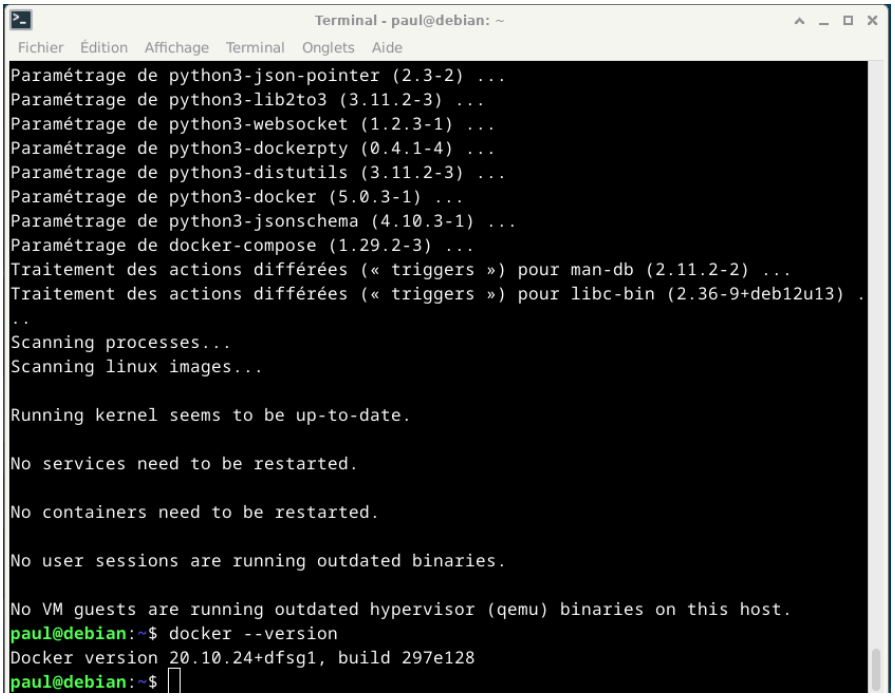


Je tente de me connecter avec les identifiants de l'utilisateur Isabelle Aragon et on me demande alors de modifier le mot de passe me confirmant ainsi le bon fonctionnement du script.



Exercice 3 : Docker - [lien Github de l'exercice]

Après avoir créé une machine virtuelle sous Debian 12 dans VMware Workstation, je commence par installer Docker et ses dépendances



Je crée ensuite un dossier visant à héberger mon projet. A l'intérieur de ce dossier, je crée un répertoire nommé *secrets* où l'on pourra trouver les fichiers contenant les mots de passe des utilisateurs dans le but d'éviter leur exposition.

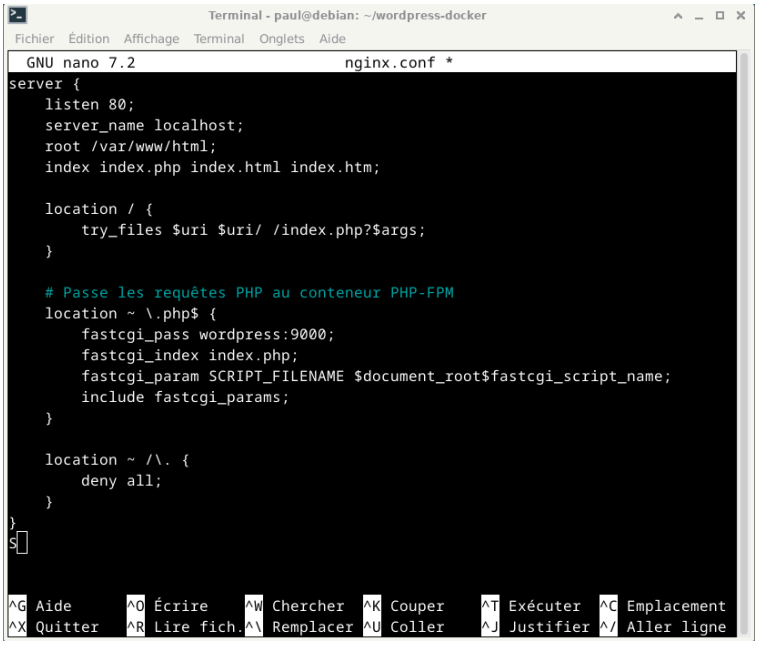
Je crée alors un fichier docker-compose.yml qui va définir trois services qui vont me permettre de déployer mon site Wordpress:

- la base de données (MariaDB) qui stocke les données de WordPress
- L'application Wordpress qui exécute le code PHP et interroge la base de données
- Le serveur web Nginx qui reçoit les requêtes des utilisateurs, sert les fichiers statiques, et transmet les requêtes PHP à Wordpress

```
services:
  # 1. Base de Données (MariaDB)
  db:
    image: mariadb:10.9
    container_name: wordpress_db
    restart: always
    environment:
      # Les mots de passe sont lus depuis les secrets montés dans le
conteneur
      MYSQL_ROOT_PASSWORD_FILE: /run/secrets/db_root_password
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wp_user
      MYSQL_PASSWORD_FILE: /run/secrets/db_user_password
    volumes:
      - db_data:/var/lib/mysql
    secrets:
      - db_root_password
      - db_user_password
    networks:
      - wp-network

  # 2. Moteur PHP-FPM (Wordpress Core)
  wordpress:
    image: wordpress:fpm
    container_name: wordpress_php
    restart: always
    environment:
      # Connexion à la base de données via le nom du service "db"
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wp_user
      WORDPRESS_DB_PASSWORD_FILE: /run/secrets/db_user_password
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wp_html:/var/www/html # Volume partagé (code et données WP)
    depends_on:
      - db
    secrets:
      - db_user_password

    # COMMANDE POUR FIXER LES PERMISSIONS :
    # Assure que l'utilisateur PHP (www-data) a les droits d'écriture
```

```
sur le volume.
    command: >
      bash -c "chown -R www-data:www-data /var/www/html && exec
docker-entrypoint.sh php-fpm"

    networks:
      - wp-network

  # 3. Serveur Web (Nginx)
  web:
    image: nginx:stable-alpine
    container_name: wordpress_nginx
    restart: always
    ports:
      - "80:80" # Expose le port 80 de l'hôte
    volumes:
      - wp_html:/var/www/html:ro # Le même volume, en lecture seule
(ro) pour la sécurité
      - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro #
Configuration Nginx locale
    depends_on:
      - wordpress
    networks:
      - wp-network

# Définition des Volumes pour la Persistance des Données
volumes:
  db_data:
  wp_html:

# Définition du Réseau Interne
networks:
  wp-network:
    driver: bridge

# Définition des Secrets (Chemins vers les fichiers locaux)
secrets:
  db_root_password:
    file: ./secrets/db_root_password.txt
  db_user_password:
    file: ./secrets/db_user_password.txt
```

Je crée ensuite mon fichier de configuration pour Nginx afin que mon serveur web écoute le trafic sur le port 80 et transmet les requêtes PHP à Wordpress pour qu'il exécute le code.



Je démarre ensuite l'application avec la commande *docker compose up -d* et vérifie le fonctionnement de mes conteneurs avec *docker ps*

J'ouvre un navigateur et effectue une requête http sur le port 80 de ma machine afin de vérifier le bon déploiement de mon site Wordpress