Rendu Exercices La Plateforme

Gaël GREVSBO

Exercice 01 : Packet Tracer

Lien Github : https://github.com/GaelGre/La-Plateforme/tree/main/Ex%201

Process détaillé :

Chaque bureau sera composé de :

- 1 switch
- 1 point d'accès Wi-Fi
- 1 PC portable
- 2 PC fixes
- 1 téléphone IP

On va donc faire 3 bureaux car on dispose de 3 switchs, 3 points d'accès, 3 PC Portable, ...

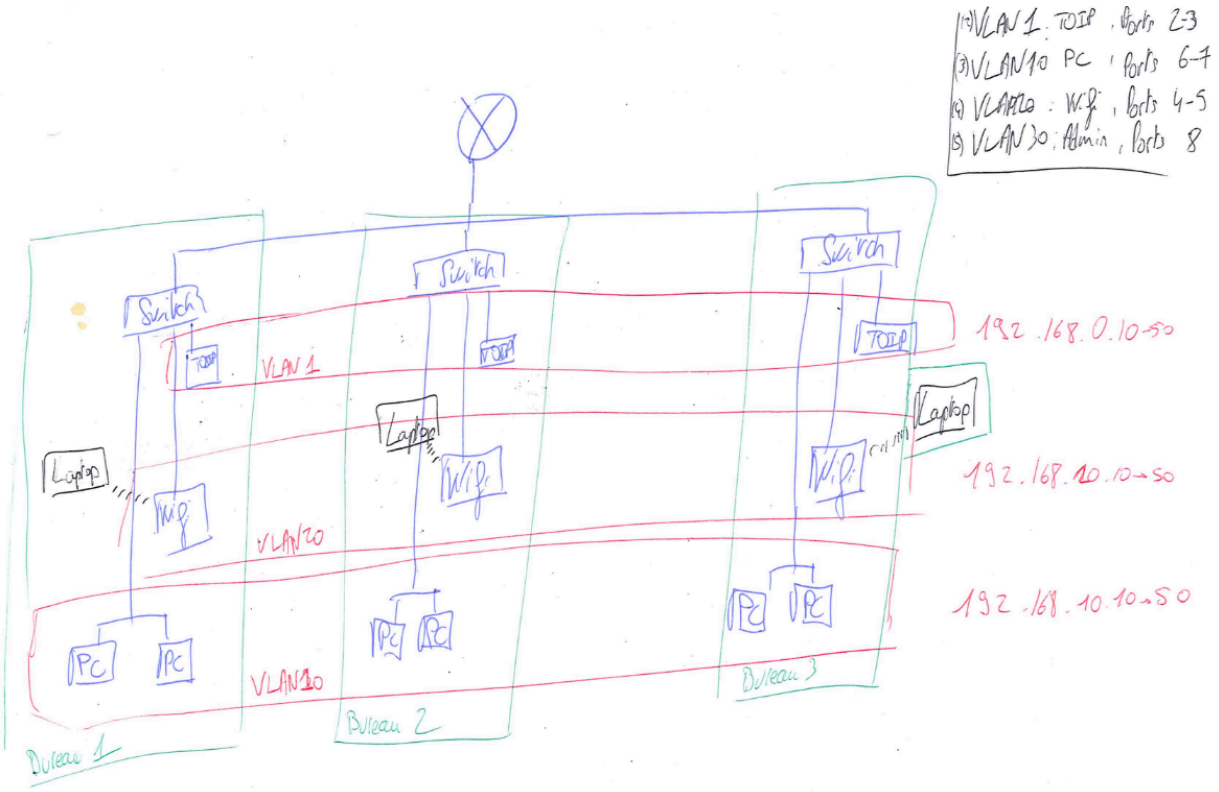
On va également faire 4 VLAN :

- 1 pour les téléphones IP (VLAN 1)
- 1 pour les points d'accès Wifi (VLAN 20)

- 1 pour les PC Fixes (VLAN 10)
- 1 pour l'administration (VLAN 30)

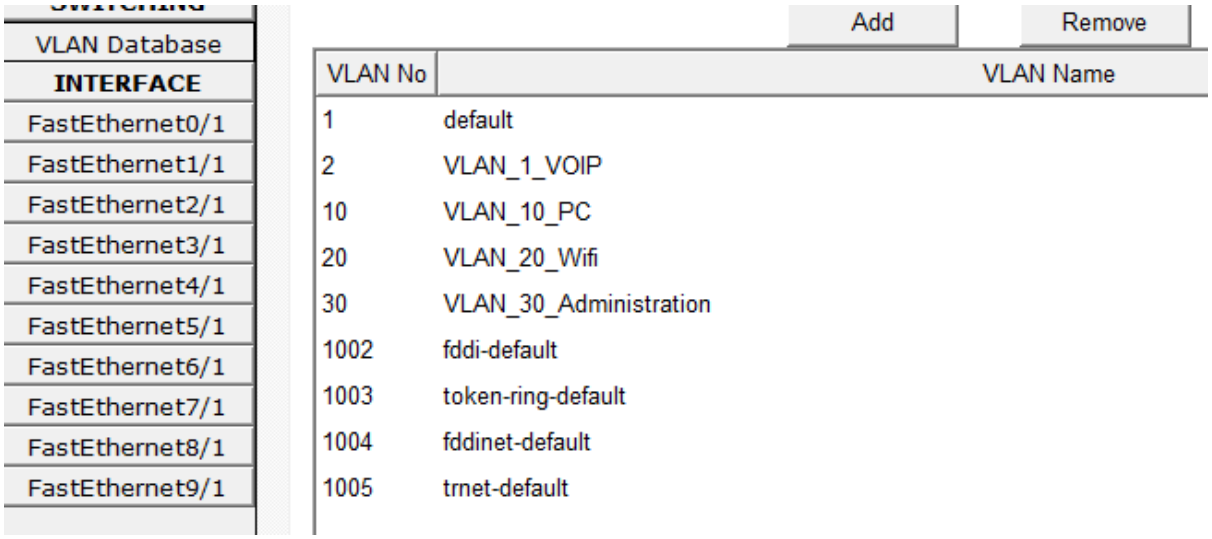
Il y a une petite coquille dans l'énoncé sur la numérotation des VLAN, j'adopte donc la numérotation mentionnée ci-dessus.

Pour résumer, on fait un schéma :



On commence par créer un premier bureau avec 1 switch, 2 PC fixes, 1 téléphone à IP, 1 access point et 1 laptop.

On crée les 4 VLAN dans le switch avec la table de VLAN :



Et on vient ensuite attribuer le bon VLAN à chaque port comme défini dans l'énoncé (en forçant bien les ports 1 et 9 en mode TRUNK) :

Port 8 : Administration, VLAN 30

Ports 6-7 : PC fixes, VLAN 20

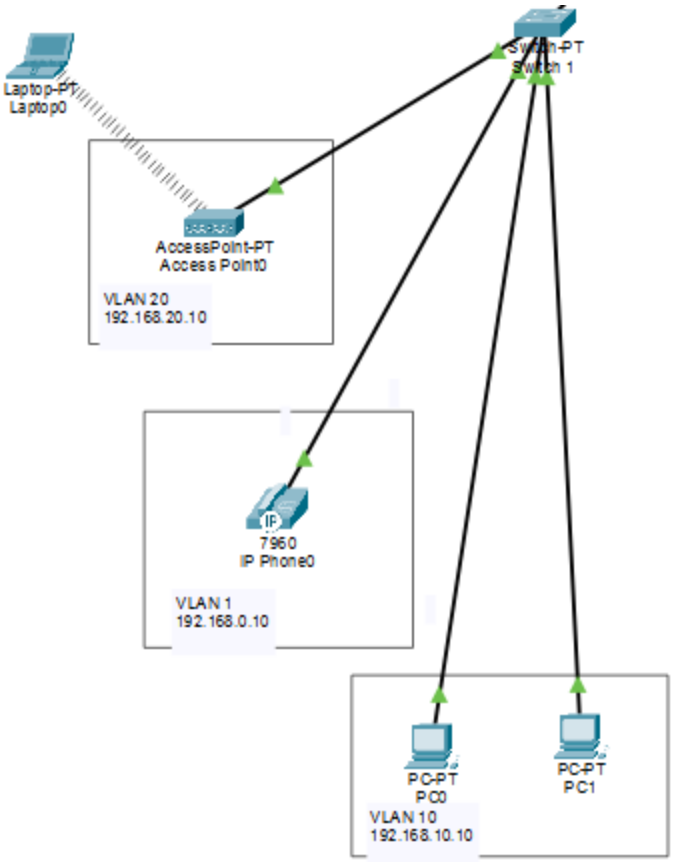
Ports 4-5 : Points d'accès Wi-Fi, VLAN 10

Ports 2-3 : Téléphones IP (VoIP), VLAN 1

Ports 1 et 9 : Uplink (Ethernet/Fibre), TRUNK

On connecte les différents périphériques à leur port respectif et on connecte également le laptop au point d'accès wifi (en lui ajoutant le module wifi WPC300N dans l'onglet "Physical").

A ce stade, nous en sommes là :



On ajoute un routeur qu'on relie au switch et on active l'interface du routeur qui est reliée au switch (Gigabit Ethernet 0/0) avec no shutdown.

On va ensuite attribuer des adresses IP (statiques pour l'instant) à nos ordinateurs :

192.168.10.10 et 192.168.10.11 pour les PC1 et PC2 puis 192.168.20.10 pour le laptop1. Le masque de sous-réseau /24 est automatiquement attribué pour chacun. Pour l'instant les 2 PC fixes peuvent communiquer entre eux car ils sont dans le même VLAN mais pas un PC et le laptop. On peut vérifier cela avec une commande ping depuis l'une des deux machines, en appelant l'autre :

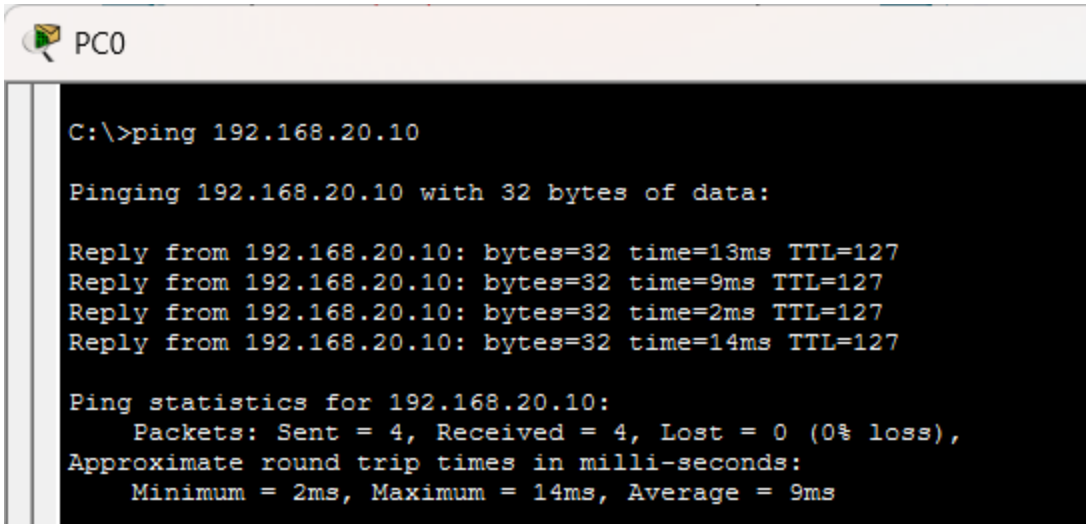
```
C:\>ping 192.168.20.10

Pinging 192.168.20.10 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.20.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Nous allons ensuite créer les 4 sous-interfaces pour les 4 VLAN au niveau du routeur, avec la commande encapsulation dot1Q. Exemple ici pour les VLAN 10 et 20 :

```
Router(config)#int
Router(config)#interface gi
Router(config)#interface gigabitEthernet 0/0.10
Router(config-subif)#en
Router(config-subif)#encapsulation do
Router(config-subif)#encapsulation dot1Q 10
Router(config-subif)#ip adD
Router(config-subif)#ip add
Router(config-subif)#ip address 192.168.10.50 255.255.255.0
Router(config-subif)#exit
Router(config)#interface gigabitEthernet 0/0.20
Router(config-subif)#encapsulation dot1Q 20
Router(config-subif)#ip address 192.168.20.50 255.255.255.0
Router(config-subif)#exit
```

Le routage inter VLAN a bien été configuré. On ajoute les default Gateway pour chaque ordinateur.

On peut désormais re-tester la connectivité entre 2 machines de 2 VLAN différents. Exemple ici avec l'appel du laptop depuis le PC Fixe 0 :



On va maintenant configurer le DHCP pour chaque VLAN.

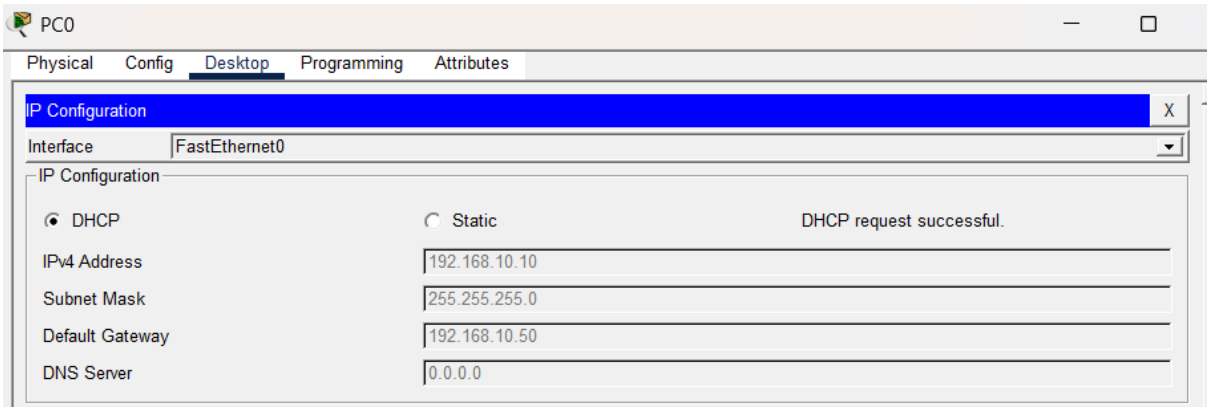
On va créer un pool d'adresse pour chaque VLAN et déclarer la passerelle par défaut. Exemple pour VLAN10 et VLAN20 :

```
Router(config)#ip
Router(config)#ip dh
Router(config)#ip dhcp p
Router(config)#ip dhcp pool VL
Router(config)#ip dhcp pool VLAN10
Router(dhcp-config)#net
Router(dhcp-config)#network 192.168.10.10 255.255.255.0
Router(dhcp-config)#DEF
Router(dhcp-config)#def
Router(dhcp-config)#default-router 192.168.10.50
Router(dhcp-config)#EX
Router(config)#ip dh
Router(config)#ip dhcp p
Router(config)#ip dhcp pool VLAN20
Router(dhcp-config)#net
Router(dhcp-config)#network 192.168.20.10 255.255.255.0
Router(dhcp-config)#def
Router(dhcp-config)#default-router 192.168.20.50
Router(dhcp-config)#ex
Router(dhcp-config)#exit
```

On exclut ensuite les adresses que l'on ne veut pas pour garder seulement celles comprises entre .10 et .50. Donc dans notre cas on exclut celles entre .1 et .9 puis celles entre .51 et .254 :

```
Router(config)#ip dhcp excluded-address 192.168.0.1 192.168.0.9
Router(config)#ip dhcp excluded-address 192.168.10.1 192.168.10.9
Router(config)#ip dhcp excluded-address 192.168.20.1 192.168.20.9
Router(config)#ip dhcp excluded-address 192.168.30.1 192.168.30.9

Router(config)#ip dhcp excluded-address 192.168.10.51 192.168.10.254
Router(config)#ip dhcp excluded-address 192.168.20.51 192.168.20.254
Router(config)#ip dhcp excluded-address 192.168.30.51 192.168.30.254
Router(config)#ip dhcp excluded-address 192.168.0.51 192.168.0.254
```

On peut ensuite aller sur nos différentes machines vérifier que la configuration de notre DHCP a bien été effectuée. On clique sur “DHCP” dans l’onglet “Desktop”. Exemple ici pour PC0 :



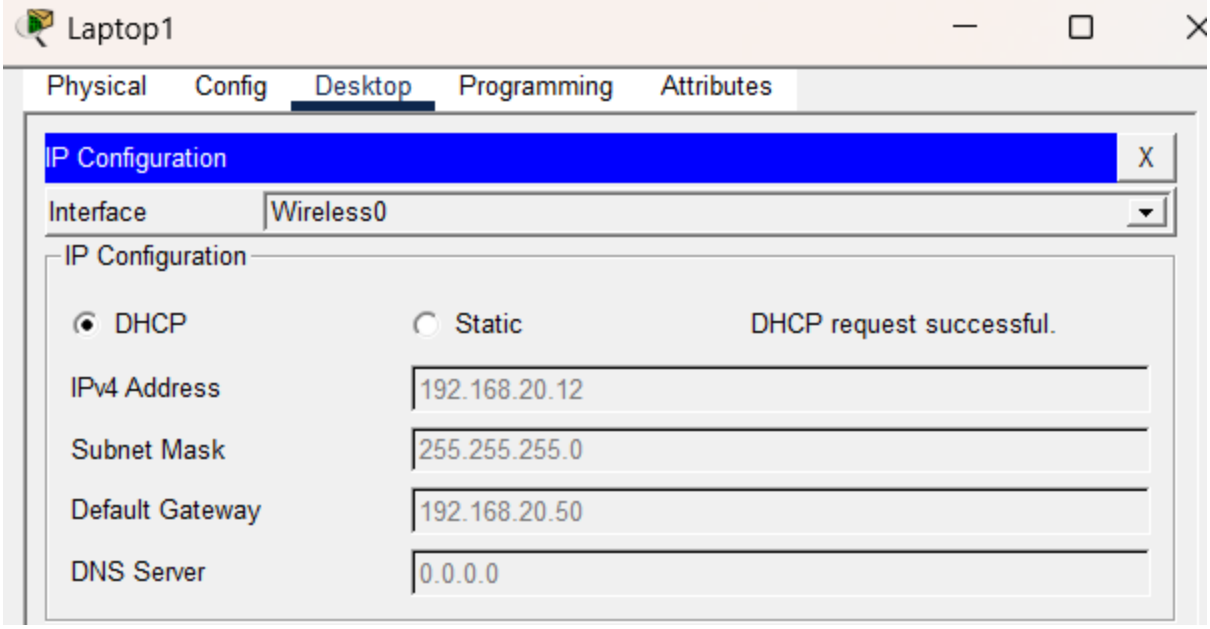
Une adresse IP correspondante à celle que l’on voulait pour cette VLAN a bien été générée.

On vérifie avec un Ping que la connexion entre VLAN est toujours effective et on continue.

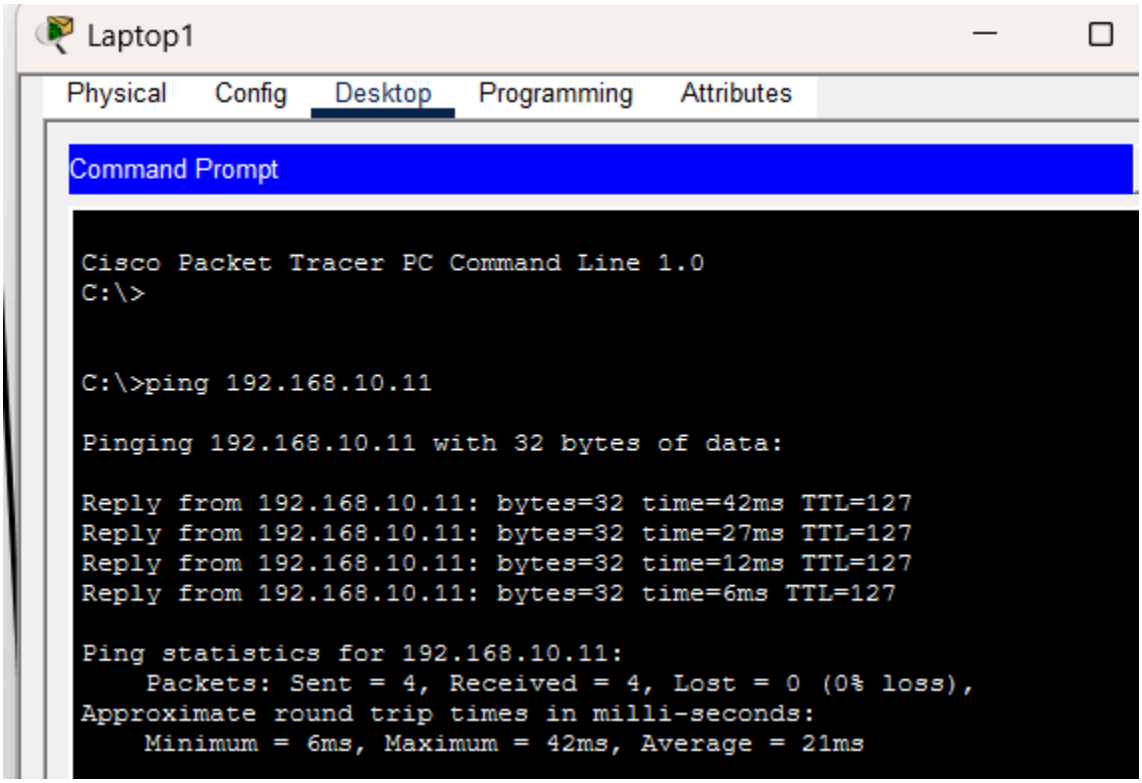
On crée notre deuxième bureau : on duplique notre premier switch et on rajoute les autres éléments (2 PC fixes, 1 téléphone IP, …). On connecte ce switch à celui du premier bureau via un câble croisé sur les bons ports (0 ou 9) puis on connecte les différents éléments au switch, toujours sur les bons ports définis au début. On rajoute la carte réseau sur le pc portable et on se

connecte à l'access point du bureau correspondant.

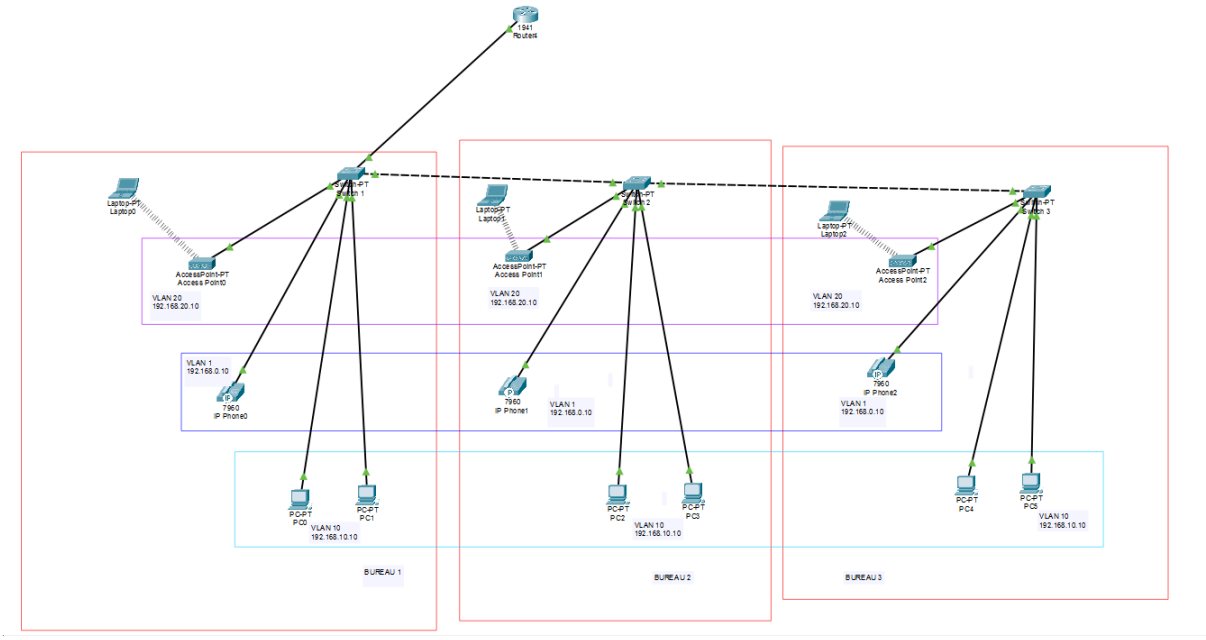
On vérifie sur les différentes machines que les adresses IP se sont bien générées.



On peut également vérifier la connectivité entre VLAN avec un Ping :



On met au propre notre schéma en faisant apparaître les différents VLAN et les 3 bureaux :



Exercice 02 : Active Directory

Lien Github : https://github.com/GaelGre/La-Plateforme/tree/main/Ex%202

Process détaillé :

On suppose que sur le windows server, la machine a été renommée, la configuration réseau et la définition du serveur DNS ont déjà été faite.

On commence donc par ajouter les rôles ADDS et DNS à notre windows server.

```powershell
# Installation du rôle AD DS
Add-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools -IncludeAllSubFeature
Write-Output "Installation ADDS réussie"

# Installation du rôle DNS
Add-WindowsFeature -Name DNS -IncludeManagementTools -IncludeAllSubFeature
Write-Output "Installation DNS réussie"
```

Ensuite, pour créer le domaine, je vais créer une hastable qui va contenir tous les paramètres de configuration ($ForestConfiguration). C'est plus lisible et ça permet de modifier plus facilement les paramètres si besoin.

```powershell
$DNSName = "laplateforme.io"
$NetbiosName = "LAPLATEFORME"
$AdminPassword = ConvertTo-SecureString "AdminPwd" -AsPlainText -Force

$ForestConfiguration = @{
'-DomainName' = $DNSName;
'-DomainNetbiosName' = $NetbiosName;
'-DomainMode' = 'Default';
'-LogPath' = 'C:\Windows\NTDS';
'-SysvolPath' = 'C:\Windows\SYSVOL';
'-DatabasePath'= 'C:\Windows\NTDS';
'-ForestMode' = 'Default';
'-InstallDns' = $true;
'-NoRebootOnCompletion' = $false;
'-Force' = $true;
'-CreateDnsDelegation' = $false
'-SafeModeAdministratorPassword' = $AdminPassword  }
```

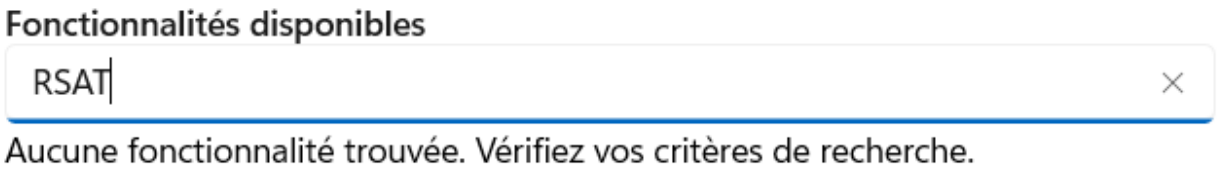
Enfin, on lance la création du domaine

```
32  # Lancement de la création du domaine
33  Import-Module ADDSDeployment
34  Install-ADDSForest @ForestConfiguration
```

Pour le deuxième script qui doit peupler l'AD, je précise que je n'ai pas réussi à installer les outils d'administration à distance sur mon PC. J'ai essayé aussi bien depuis Powershell avec la commande ci-dessous mais le statut est toujours resté en "running".

Commande : *Add-WindowsCapability -Online -Name "Rsat.ServerManager.Tools~~~~0.0.1.0"*

J'ai également essayé depuis mes paramètres systèmes mais impossible d'ajouter les fonctionnalités RSAT :



Je n'ai donc pas pu importer le module ActiveDirectory sur ma machine personnelle et vérifier que le script fonctionnait bien.

Pour créer ce deuxième script qui doit peupler l'AD, on va commencer par créer notre script qui crée chaque user à partir du CSV et ensuite on

s'intéressera aux groupes.

On commence par importer le module ActiveDirectory et créer nos OU Utilisateurs et Groupes.

```
Import-Module ActiveDirectory

#Création des OU
New-ADOrganizationalUnit -Name "Utilisateurs" -Path "OU=DC=LAPLATEFORME, DC=IO"
New-ADOrganizationalUnit -Name "Groupes" -Path "OU=DC=LAPLATEFORME, DC=IO"
```

On peut ensuite importer les données du CSV :

```
# Importer les données
$CSVFile = "C:\Users\bigbo\Documents\La Plateforme\Fichier CSV.csv"
$CSVData = Import-Csv -Path $CSVFile -Delimiter "," -Encoding Default
Write-Output $CSVData
```

On vérifie que cela marche bien avec le Write-Ouput :

```
nom       : ALEXANDRE
prénom    : MARCELLINE
groupe1   : Animation
groupe2   :
groupe3   :
groupe4   :
groupe5   :
groupe6   :

nom       : ARAGON
prénom    : ISABELLE
groupe1   : Animation
groupe2   :
groupe3   :
groupe4   :
groupe5   :
groupe6   :

nom       : AVARO
prénom    : MARINA
groupe1   : As
groupe2   : Médical
groupe3   :
groupe4   :
groupe5   :
groupe6   :

nom       : BERNARD
prénom    : ISABELLE
groupe1   : As
groupe2   : Médical
```

On crée ensuite notre boucle Foreach qui va parcourir le CSV. Pour chaque user on va créer différentes variables qui reprennent les éléments de chacun : nom, prénom, login (première lettre du prénom et nom, le tout en minuscule) et email.

```
Foreach($User in $CSVData){
    $NomUser = $User.nom
    $PrenomUser = $User.prénom
    $LoginUser = ($User.prénom.Substring(0,1) + $User.nom).ToLower()
    $EmailUser = "$LoginUser@laplateforme.fr"
```

On peut ensuite créer l'user, en vérifiant au préalable :

- que le SamAccountName ne dépasse pas 20 caractères
- que l'user n'existe pas déjà avec un if qui filtre sur le SamAccountName.

Ce qui nous donne en powershell :

```
    #Vérification de la longueur du SamAccountName
    if ($LoginUser.Length -gt 20)
    {
        $LoginUser = $LoginUser.Substring(0,20)
    }

    #Vérification de la présence de l'user dans l'AD
    if (Get-ADUser -Filter {SamAccountName -eq $LoginUser})
    {
        Write-Warning "L'utilisateur $LoginUser existe déjà dans l'AD"
    }

    else
    {
        New-ADUser `
        -Name "$NommUser $PrenomUser" `
        -DisplayName "$NommUser $PrenomUser" `
        -GivenName $PrenomUser `
        -Surname $NomUser `
        -SamAccountName $LoginUser `
        -UserPrincipalName $EmailUser `
        -EmailAddress $EmailUser `
        -Path "OU=Utiliasteurs, DC=LAPLATEFORME, DC=IO" `
        -AccountPassword $PasswordUser `
        -Enabled $true `
        -ChangePasswordAtLogon $true
    }
```

On a bien rajouté le ChangePasswordAtLogon en true comme demandé dans l'exercice.

On peut maintenant se concentrer sur les groupes. On va commencer par créer la variables groupes qui va contenir tous les groupes possibles non vides et sans doublon grâce au Sort-Object -Unique.

```
$groupes = $CSVData | ForEach-Object {
    $_.groupe1, $_.groupe2, $_.groupe3, $_.groupe4, $_.groupe5, $_.groupe6} | Where-Object {$_ -and $_ -ne "" } | Sort-Object -Unique
```

On peut ensuite les créer :

```
# Création des groupes
foreach ($groupe in $groupes) {
    New-ADGroup `
        -Name $groupe `
        -GroupScope Global `
        -GroupCategory Security `
        -Path "OU=Groupes, DC=LAPLATEFORME, DC=IO"
}
```

Enfin, on peut rajouter dans notre boucle Foreach initiale, l'ajout à chaque groupe auquel un utilisateur appartient :

```
    Foreach($groupe in @($User.groupe1,$User.groupe2,$User.groupe3,$User.groupe4,$User.groupe5,$User.groupe6)) {
        if ($groupe -and $groupe -ne "")
        {
            Add-ADGroupMember -Identity $groupe -Members $LoginUser
        }

    }
```

Exercice 03 : Docker

Lien Github : https://github.com/GaelGre/La-Plateforme/tree/main/Docker%20Local%20Wordpress

Process détaillé :

On crée notre fichier racine et on crée notre fichier docker-compose.

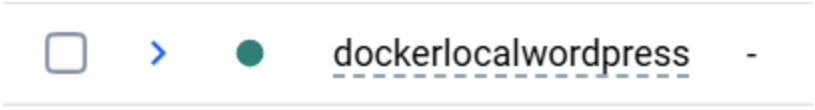
On commence par rajouter le service nginx (j'ai pris la dernière version en alpine car elle est plus légère).

```
services:
  ▷Run Service
  web:
    container_name: wp_nginx
    image: nginx:1.29.4-alpine
    ports:
      - 8080:80
    restart: unless-stopped
    env_file:
      - .env
    volumes:
      - ./nginx:/etc/nginx/conf.d:rw
```

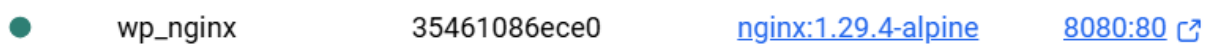
On crée ensuite le fichier default.conf et on configure la gestion PHP.

```
server {
    listen 80 default_server;

    root /var/www/html;

    location / {
        index index.php index.html;
    }
}
```

On peut ensuite lancer un docker compose up dans le terminal puis vérifier dans Docker que notre container s'est bien créé :



On regarde également que le container nginx s'est bien créé, en version 1.29.4.



On peut ensuite se rendre sur notre local host:8080 :

On a une erreur 404, ce qui est normal pour le moment car il cherche un fichier index.php alors que nous n'avons pas encore ajouté le container php, comme on peut le voir dans les logs :

```
[error] 29#29: *1 "/var/www/html/index.php" is not found (2: No such file or directory),
```

On va ensuite créer notre base de données MariaDB.

On commence par créer un fichier .env qui contiendra nos variables. Cela permet que nos variables soient dans un fichier séparé de notre docker-compose.

.env

```
MARIA_DATABASE=wordpress
MARIA_USER=wp_user
MARIA_PASSWORD=wp_password
MARIA_ROOT_PASSWORD=root_password
```

Le fichier .env n'est pas sur le git mais correspond juste à ces 4 lignes.

On prend la dernière version de MariaDB et on pointe vers le port 3306 qui est le port par défaut de MySQL.
Comme on veut que les deux images (web et BDD) communiquent entre elles, on va rajouter un network commun à chacun que l'on définit en bas de notre fichier. On ajoutera également ce network à chaque nouvelle image.
On crée enfin un volume pour notre base de données(db_data). On va donc rajouter la section volumes dans notre container database et on va le mapper à db_data.
Ce qui nous donne :

```yaml
database:
  container_name: wp_database
  image: mariadb:latest
  restart: unless-stopped
  ports:
    - 3306:3306
  env_file: .env
  environment:
    MARIADB_ROOT_PASSWORD: '${MARIA_ROOT_PASSWORD}'
    MARIADB_DATABASE: '${MARIA_DATABASE}'
    MARIADB_USER: '${MARIA_USER}'
    MARIADB_PASSWORD: '${MARIA_PASSWORD}'
  volumes:
    - db_data:/var/lib/mariadb
  networks:
    - wordpress-network
```

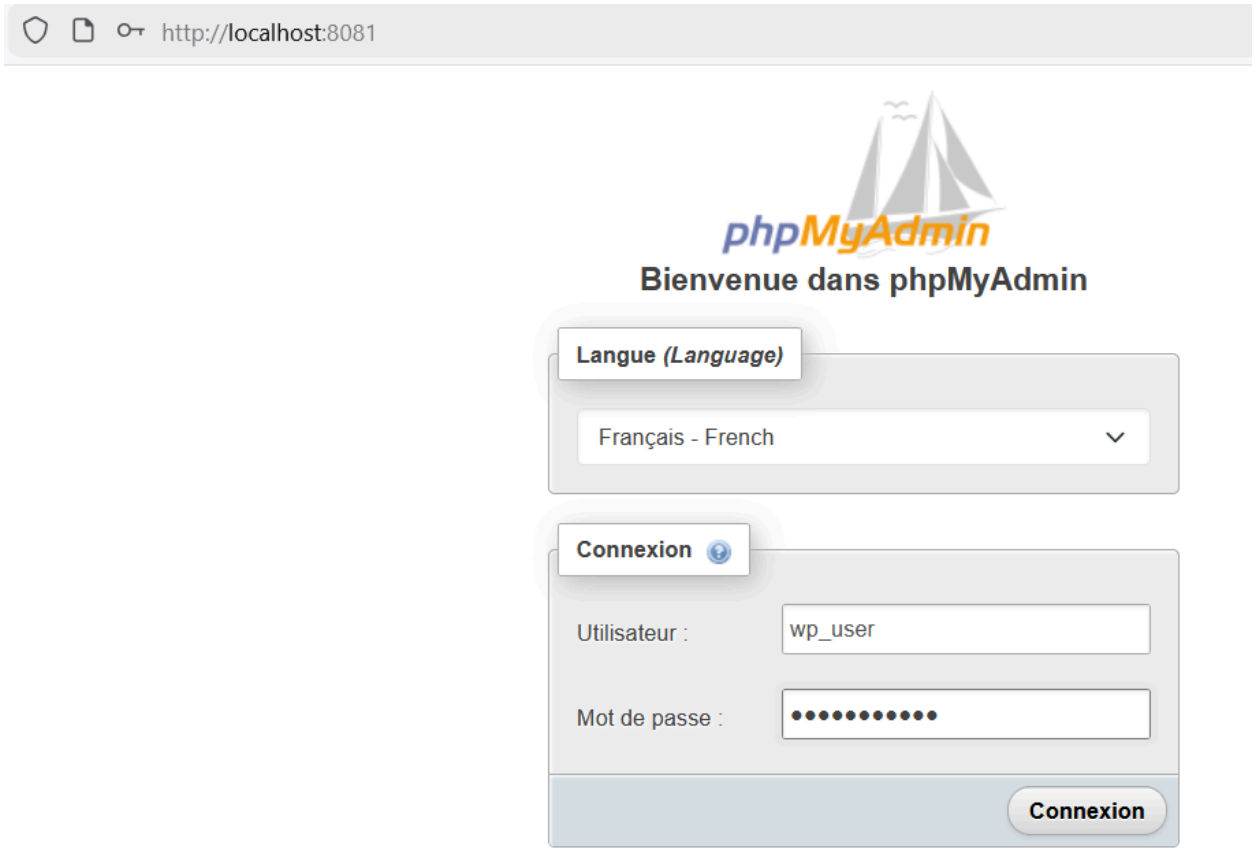
Enfin on ajoute notre container phpmyadmin, monté de la même manière :

```yaml
phpmyadmin:
  container_name: wp_php
  depends_on:
    - database
  image: phpmyadmin/phpmyadmin
  restart: unless-stopped
  ports:
    - 8081:80
  env_file: .env
  environment:
    PMA_HOST: database
    PMA_PORT: 3306
    MARIA_ROOT_PASSWORD: '${MARIA_ROOT_PASSWORD}'
  networks:
    - wordpress-network
```

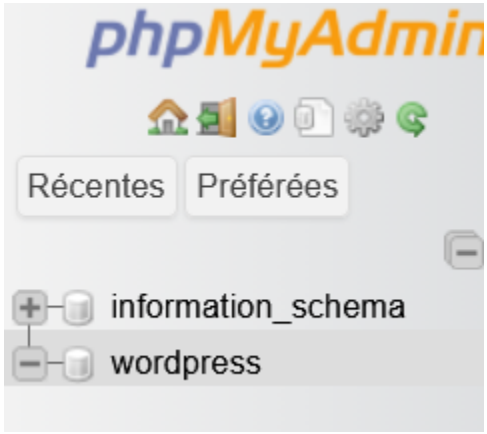
On peut ensuite refaire un docker compose up pour créer nos containers et on vérifie qu'ils sont bien créés dans Docker.

	Name	Container ID	Image	Port(s)
	dockerlocalwordpress	-	-	-
	wp_php	ebb2c9e2870f	phpmyadmin/phpmyadn	8081:80
	wp_nginx	e947c39774bf	nginx:1.29.4-alpine	8080:80
	wp_database	8ed9d1b0e310	mariadb:latest	3306:3306

On peut aller sur notre localhost:8081 qui correspond à phpmyadmin et se connecter avec les identifiants qu'on a mis dans notre fichier .env.



On peut notamment vérifier qu'on a bien notre base de données "wordpress" qui a été créée :



On ajoute enfin le container Wordpress, toujours monté de la même manière. On prend la version fpm pour la connecter plus facilement à nginx et configurer la prise en charge de php. On a aussi créé un volume wordpress

que l'on va également rajouté dans notre container nginx.

```yaml
wordpress:
  depends_on:
    - database
  image: wordpress:6.9.0-fpm-alpine
  restart: unless-stopped
  env_file: .env
  environment:
    WORDPRESS_DB_HOST: database:3306
    WORDPRESS_DB_NAME: '${MARIA_DATABASE}'
    WORDPRESS_DB_USER: '${MARIA_USER}'
    WORDPRESS_DB_PASSWORD: '${MARIA_PASSWORD}'
  volumes:
    - wordpress:./:var/www/html
  networks:
    - wordpress-network
```

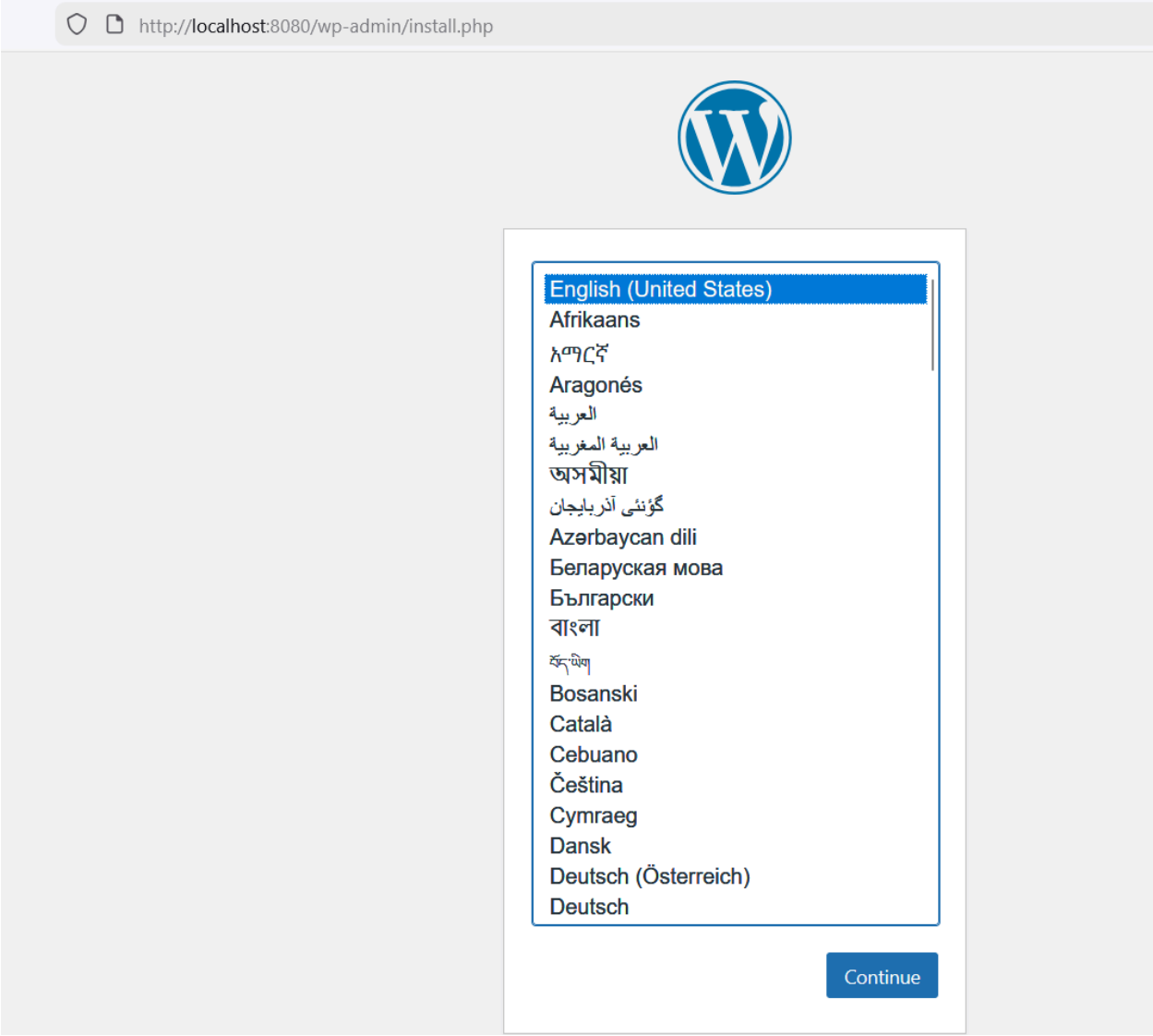
On va maintenant pouvoir revenir sur notre fichier de configuration NGINX pour configurer la gestion PHP avec FastCGI, qui est intégré à l'image FPM. On met bien le port 9000 qui est le port par défaut de FPM.

```
location ~ \.php$ {
    include fastcgi_params;
    fastcgi_pass wordpress:9000;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
}
```

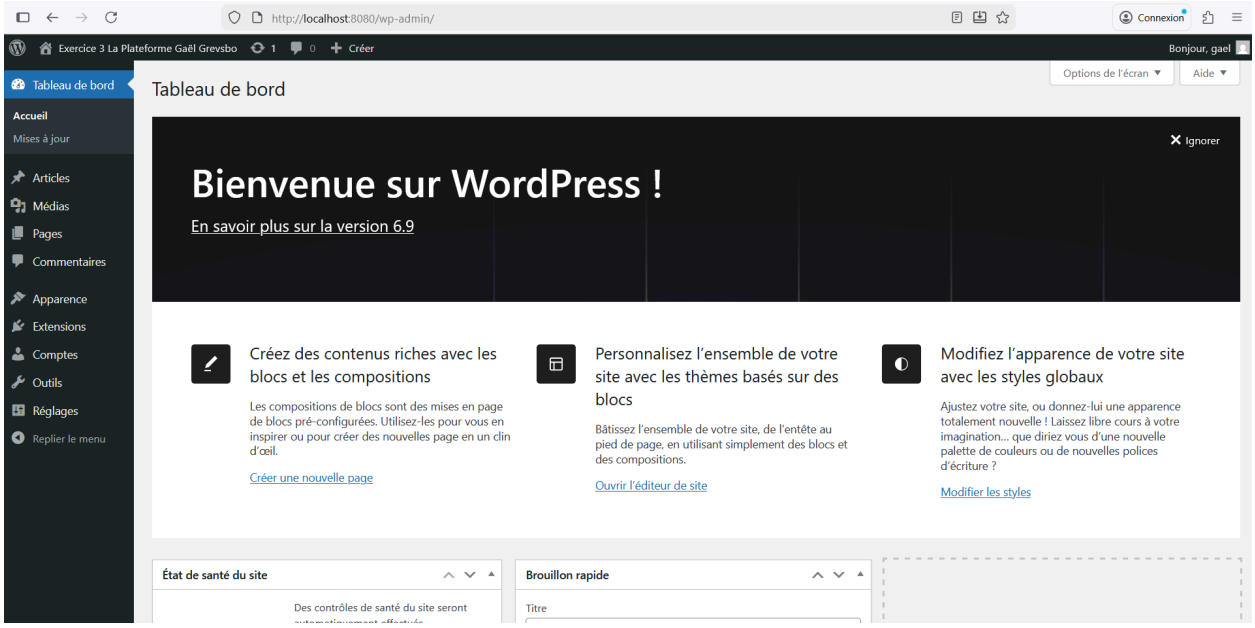
On peut désormais faire un dernier docker compose up et vérifier dans Docker que tous nos containers sont bien en running :

	Name	Container ID	Image	Port(s)
	dockerlocalwordpress	-	-	-
	wp_php	ebb2c9e2870f	phpmyadmin/phpmyadn	8081:80
	wp_database	8ed9d1b0e310	mariadb:latest	3306:3306
	e947c39774bf_wp_ng	717c46c77385	nginx:1.29.4-alpine	8080:80
	wordpress-1	6f0fe011641b	wordpress:6.9.0-fpm-alp	

On peut ensuite se rendre sur notre localhost:8080 et vérifier qu'on a bien Wordpress :



On a bien installé Wordpress !



En relisant l'énoncé je m'aperçois que je n'ai pas pris en compte le fait d'utiliser un volume commun.

Je supprime donc le volume db_data à la fin de mon fichier :

```
volumes:
  wordpress:
```

Et dans mon container database, je remplace le volume db_data par l'unique volume wordpress :

```
volumes:
  - wordpress:/var/lib/mariadb
```