TP 4

Administration système

Dardor Rochdi

Intégration dans l'infrastructure

Pour sécuriser et contrôler l'accès au service SSH dans un environnement réseau virtualisé, la configuration du pare-feu iptables a été mise en place dans le fichier /etc/iptables/rules.v4. Cette configuration permet la redirection des connexions SSH entrantes depuis le port externe 2223 vers le serveur interne 192.168.0.3 sur le port standard 22.
Dans la section filter, une règle a été ajoutée pour autoriser le transfert des paquets SSH vers la machine ciblée
Dans la section NAT : **Redirection de port (DNAT)** : Le trafic entrant sur le port 2223 est redirigé vers le port 22 de la machine 192.168.0.3.
Masquage d'adresse (POSTROUTING) : Le trafic sortant depuis le sous-réseau privé 192.168.0.0/24 est masqué via l'interface publique enpos3.


```
root@vbox:~# sudo iptables -t nat -I PREROUTING -p tcp --dport 80 -i enp0s3 -j DNAT --to 192.168.0.4:80
root@vbox:~#
```

Cette configuration permet de sécuriser l'accès SSH en utilisant un port personnalisé tout en assurant la traduction d'adresse pour l'accès des machines internes vers l'extérieur. Il est recommandé d'améliorer cette configuration en restreignant l'accès SSH à des adresses IP spécifiques et en activant la journalisation des tentatives d'accès.

Service SSH

EXERCICE 4.1 :

Pour sécuriser et faciliter la gestion distante du serveur, l'installation du service SSH a été réalisée sur la machine *server.istycorp.fr*. Le serveur *openssh-server* a été installé à l'aide de la commande suivante : **sudo apt install openssh-server -y**

Le service a ensuite été activé et démarré :

```
vboxuser@Server:~$ sudo apt install openssh-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssh-server is already the newest version (1:9.2p1-2+deb12u3).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
vboxuser@Server:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-01-04 17:32:18 +01; 8min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 1507 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS
   Main PID: 1510 (sshd)
      Tasks: 1 (limit: 2284)
     Memory: 3.1M
        CPU: 56ms
     CGroup: /system.slice/ssh.service
             └─1510 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

Enfin, le bon fonctionnement du service a été vérifié par la commande :

sudo systemctl status ssh

Le service SSH est donc fonctionnel et prêt à l'emploi pour la gestion distante du serveur.

EXERCICE 4.2 :

Pour renforcer la sécurité des connexions SSH, l'authentification par clé publique a été mise en place. Une paire de clés SSH (publique et privée) a été générée sur la machine cliente à l'aide de la commande suivante :

ssh-keygen -t rsa -b 4096

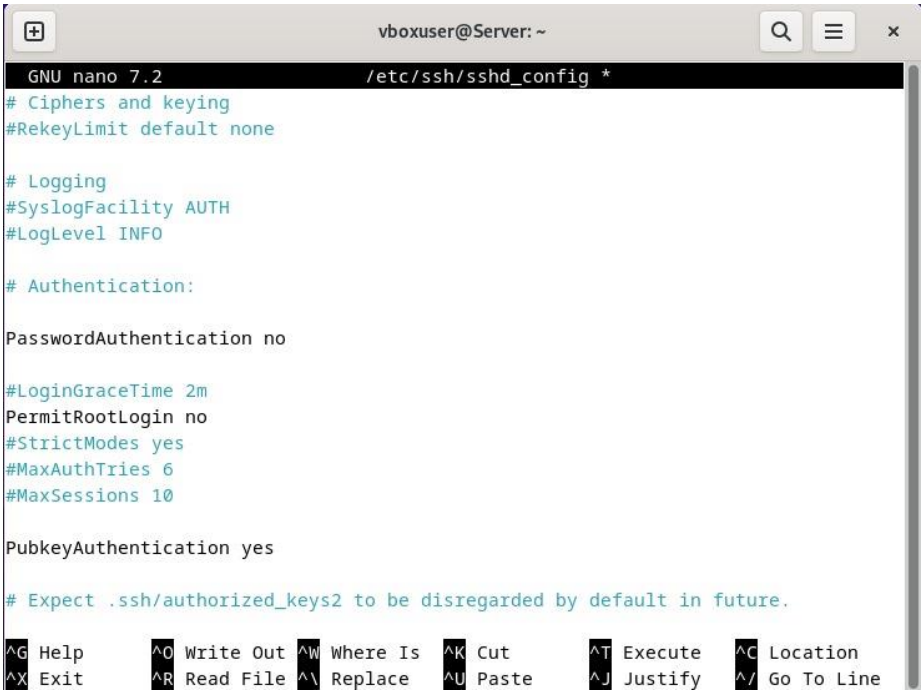
La clé publique a ensuite été copiée sur le serveur via la commande :

```
vboxuser@Client1:~$ ssh-copy-id vboxuser@192.168.0.4
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
vboxuser@192.168.0.4's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'vboxuser@192.168.0.4'"
and check to make sure that only the key(s) you wanted were added.
```

Cette action a permis d'ajouter automatiquement la clé publique au fichier ~/.ssh/authorized_keys du serveur. Un test de connexion a été réalisé avec succès sans nécessiter de mot de passe.

EXERCICE 4.3 :

Pour renforcer la sécurité du serveur SSH, l'accès par mot de passe a été désactivé et la connexion directe au compte root a été interdite. Les modifications ont été effectuées dans le fichier /etc/ssh/sshd_config :



- PasswordAuthentication no pour désactiver l'accès par mot de passe.
- PermitRootLogin no pour interdire les connexions root directes.

Après avoir enregistré ces modifications, le service SSH a été redémarré via :

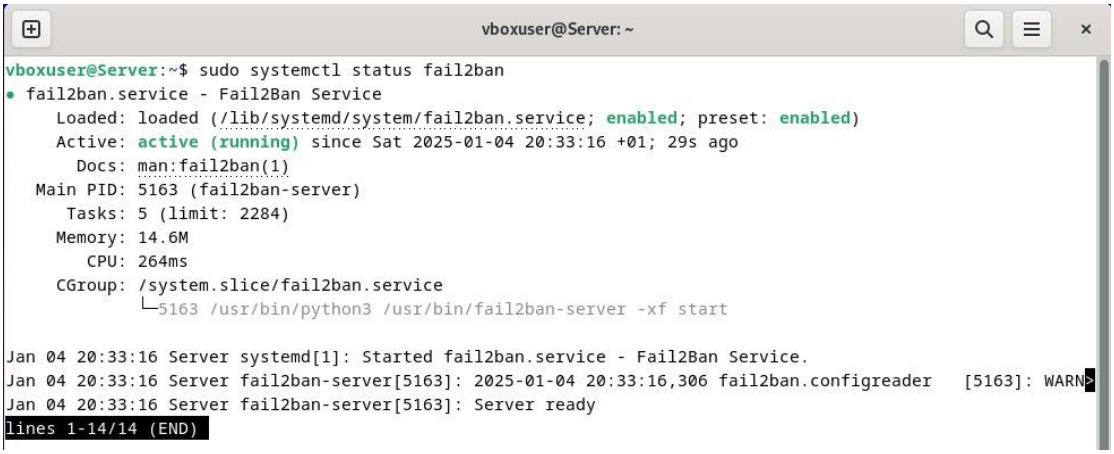
sudo systemctl restart ssh

Les tests effectués ont confirmé que l'accès au serveur est désormais uniquement possible via l'authentification par clé SSH, garantissant un niveau de sécurité accru.

EXERCICE 4.4 :

Pour protéger le serveur contre les tentatives d'accès non autorisées, l'outil *Fail2ban* a été installé et configuré. Après l'installation :

- sudo apt update
- sudo apt install fail2ban



Un fichier de configuration personnalisé /etc/fail2ban/jail.local a été créé et modifié pour activer la protection SSH. J'ai ajouté les lignes suivantes :

enabled = true
port = 22
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 600

Le nombre maximal de tentatives a été limité à trois avant bannissement, avec un temps de bannissement fixé à 10 minutes. Le service a été redémarré et activé :

- sudo systemctl restart fail2ban
- sudo systemctl enable fail2ban

Des tests ont été effectués en réalisant plusieurs tentatives de connexion échouées, confirmant l'efficacité du système avec un bannissement automatique de l'adresse IP source.

```
vboxuser@Client1:~$ ssh -p 22 isty@192.168.0.4
isty@192.168.0.4's password:
Permission denied, please try again.
isty@192.168.0.4's password:
Permission denied, please try again.
isty@192.168.0.4's password:


^C
vboxuser@Client1:~$ ssh -p 22 isty@192.168.0.4
ssh: connect to host 192.168.0.4 port 22: Connection refused
vboxuser@Client1:~$
```

```
vboxuser@Server:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 1
|  |- Total failed:     11
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 2
   |- Total banned:     2
   `- Banned IP list:   192.168.0.4 192.168.0.3
vboxuser@Server:~$
```

EXERCICE 4.5 :

Après avoir provoqué volontairement plusieurs échecs de connexion SSH, l'outil *Fail2ban* a automatiquement ajouté une règle dans le pare-feu *iptables*. Cette règle est visible via la commande : **sudo iptables-save**

```
vboxuser@Server:~$ sudo iptables-save
# Generated by iptables-save v1.8.9 (nf_tables) on Sat Jan  4 21:35:30 2025
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:f2b-sshd - [0:0]
-A INPUT -p tcp -m multiport --dports 22 -j f2b-sshd
-A f2b-sshd -s 192.168.0.3/32 -j REJECT --reject-with icmp-port-unreachable
-A f2b-sshd -j RETURN
COMMIT
# Completed on Sat Jan  4 21:35:30 2025
vboxuser@Server:~$
```

Elle se manifeste sous la forme d'une chaîne f2b-sshd bloquant le trafic provenant de l'adresse IP bannie. Cette mesure préventive empêche toute tentative d'accès ultérieure depuis l'adresse source durant la période de bannissement définie.

EXERCICE 4.6 :

Les fichiers de journalisation `/var/log/auth.log` et `/var/log/fail2ban.log` ont été consultés après plusieurs tentatives de connexion échouées. Le fichier `auth.log` a enregistré chaque tentative de connexion avec l'adresse IP source et l'utilisateur ciblé. Quant au fichier `fail2ban.log`, il a indiqué l'ajout de l'adresse IP fautive dans la chaîne `f2b-sshd` du pare-feu *iptables*, confirmant ainsi le bannissement de l'adresse après le dépassement du nombre maximal de tentatives autorisées.

```
2025-01-04T21:32:43.946344+01:00 Server sudo: pam_unix(sudo:session): session closed for user root
2025-01-04T21:33:06.869891+01:00 Server sshd[5678]: Invalid user isty from 192.168.0.3 port 53484
2025-01-04T21:33:07.791115+01:00 Server sshd[5678]: pam_unix(sshd:auth): check pass; user unknown
2025-01-04T21:33:07.792200+01:00 Server sshd[5678]: pam_unix(sshd:auth): authentication failure; logname= u
id=0 euid=0 tty=ssh ruser= rhost=192.168.0.3
2025-01-04T21:33:10.044319+01:00 Server sshd[5678]: Failed password for invalid user isty from 192.168.0.3
port 53484 ssh2
2025-01-04T21:33:12.844011+01:00 Server sshd[5678]: pam_unix(sshd:auth): check pass; user unknown
2025-01-04T21:33:14.780748+01:00 Server sshd[5678]: Failed password for invalid user isty from 192.168.0.3
port 53484 ssh2
2025-01-04T21:35:30.547467+01:00 Server sudo: vboxuser : TTY=pts/0 ; PWD=/home/vboxuser ; USER=root ; COMMA
ND=/usr/sbin/iptables-save
2025-01-04T21:35:30.553583+01:00 Server sudo: pam_unix(sudo:session): session opened for user root(uid=0) b
y (uid=1000)
2025-01-04T21:35:30.581433+01:00 Server sudo: pam_unix(sudo:session): session closed for user root
2025-01-04T21:40:59.477138+01:00 Server sudo: vboxuser : TTY=pts/0 ; PWD=/home/vboxuser ; USER=root ; COMMA
ND=/usr/bin/cat /var/log/auth.log
2025-01-04T21:40:59.505099+01:00 Server sudo: pam_unix(sudo:session): session opened for user root(uid=0) b
y (uid=1000)
vboxuser@Server:~$
```

```
2025-01-04 21:33:06,936 fail2ban.filter         [5163]: INFO    [sshd] Found 192.168.0.3 - 2025-01-04 21:33
:06
2025-01-04 21:33:10,497 fail2ban.filter         [5163]: INFO    [sshd] Found 192.168.0.3 - 2025-01-04 21:33
:10
2025-01-04 21:33:14,808 fail2ban.filter         [5163]: INFO    [sshd] Found 192.168.0.3 - 2025-01-04 21:33
:14
2025-01-04 21:33:14,893 fail2ban.actions        [5163]: NOTICE  [sshd] Ban 192.168.0.3
vboxuser@Server:~$
```

```
vboxuser@Server:~$ sudo cat /var/log/auth.log
2025-01-04T21:52:16.771233+01:00 Server sudo: pam_unix(sudo:session): session closed for user root
2025-01-04T21:52:23.341063+01:00 Server sudo: vboxuser : TTY=pts/0 ; PWD=/home/vboxuser ; USER=root ; COMMA
ND=/usr/bin/cat /var/log/auth.log
2025-01-04T21:52:23.360744+01:00 Server sudo: pam_unix(sudo:session): session opened for user root(uid=0) b
y (uid=1000)
vboxuser@Server:~$
```

EXERCICE 4.7 :

Après avoir été banni par *Fail2ban* suite à plusieurs tentatives de connexion échouées, l'accès au serveur a été rétabli de la manière suivante :

Débannissement de l'adresse IP avec :

```
vboxuser@Server:~$ sudo fail2ban-client set sshd unbanip 192.168.0.3
1
```

les fichiers de journalisation ont été vidés avec :

sudo truncate -s 0 /var/log/auth.log

sudo truncate -s 0 /var/log/fail2ban.log

```
vboxuser@Server:~$ sudo cat /var/log/auth.log
2025-01-04T21:52:16.771233+01:00 Server sudo: pam_unix(sudo:session): session closed for user root
2025-01-04T21:52:23.341063+01:00 Server sudo: vboxuser : TTY=pts/0 ; PWD=/home/vboxuser ; USER=root ; COMMA
ND=/usr/bin/cat /var/log/auth.log
2025-01-04T21:52:23.360744+01:00 Server sudo: pam_unix(sudo:session): session opened for user root(uid=0) b
y (uid=1000)
vboxuser@Server:~$
```

Le service *Fail2ban* a été redémarré pour appliquer les changements. Le serveur est désormais de nouveau accessible, et la sécurité SSH reste renforcée par la protection de Fail2ban.

Apache/PHP/Mysql/Wordpress

EXERCICE 4.8 :

Dans cet exercice, un serveur *Apache* avec le support PHP a été installé dans un conteneur *Docker* basé sur l'image debian. L'image a été obtenue avec la commande :

docker pull debian

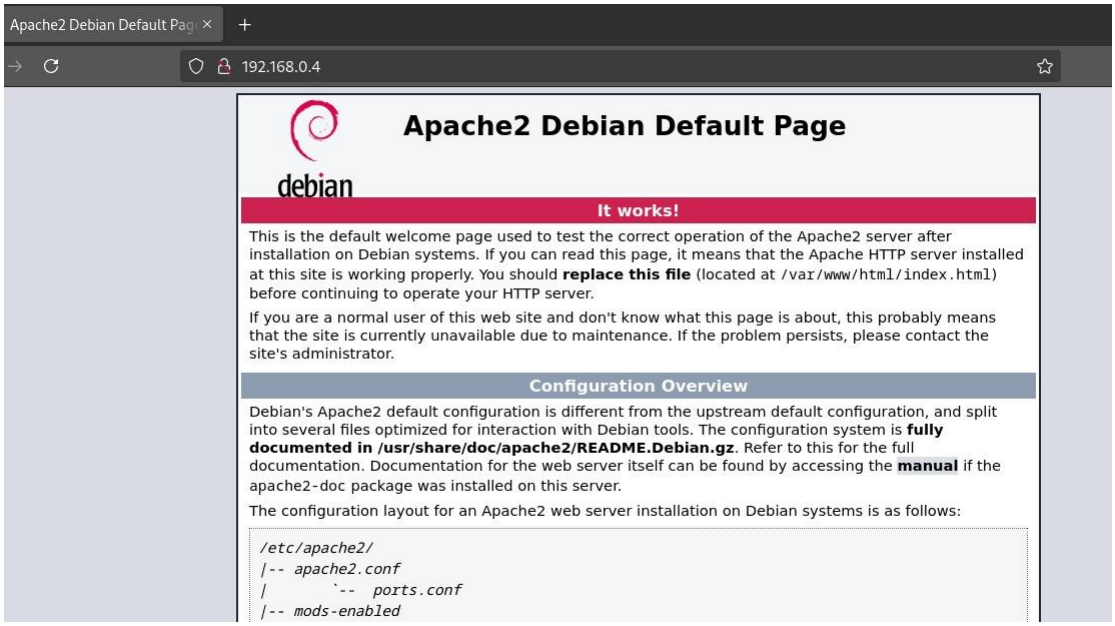
docker run -it --name apache-php-container debian bash

Les services *Apache* et PHP ont ensuite été installés et démarrés. Une page de test PHP a été créée et rendue accessible en exposant le port 8080 de l'hôte vers le port 80 du conteneur :

docker run -d -p 8080:80 apache-php-image

```
vboxuser@Server:~$ docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED        STATUS
                  PORTS                                NAMES
cd3747895f51   debian:latest   "/bin/bash -c 'while…"   18 hours ago   Exited (2
55) 5 hours ago   0.0.0.0:80->80/tcp, :::80->80/tcp   wordpress-debian
vboxuser@Server:~$
```

Le pare-feu a été configuré pour autoriser le trafic entrant vers le port 80 à travers le routeur avec *iptables*. L'installation a été validée en accédant à la page depuis un navigateur web.



EXERCICE 4.9 :

Ensuite, nous avons installé le serveur MySQL avec :
sudo apt install mysql-server -y
Puis nous avons démarré le service et exécuté le script de sécurisation :
sudo mysql_secure_installation

EXERCICE 4.10 :

Pour sécuriser le serveur MySQL, le script **`mysql_secure_installation`** a été exécuté. Celui-ci a permis :

La suppression des comptes utilisateurs anonymes.

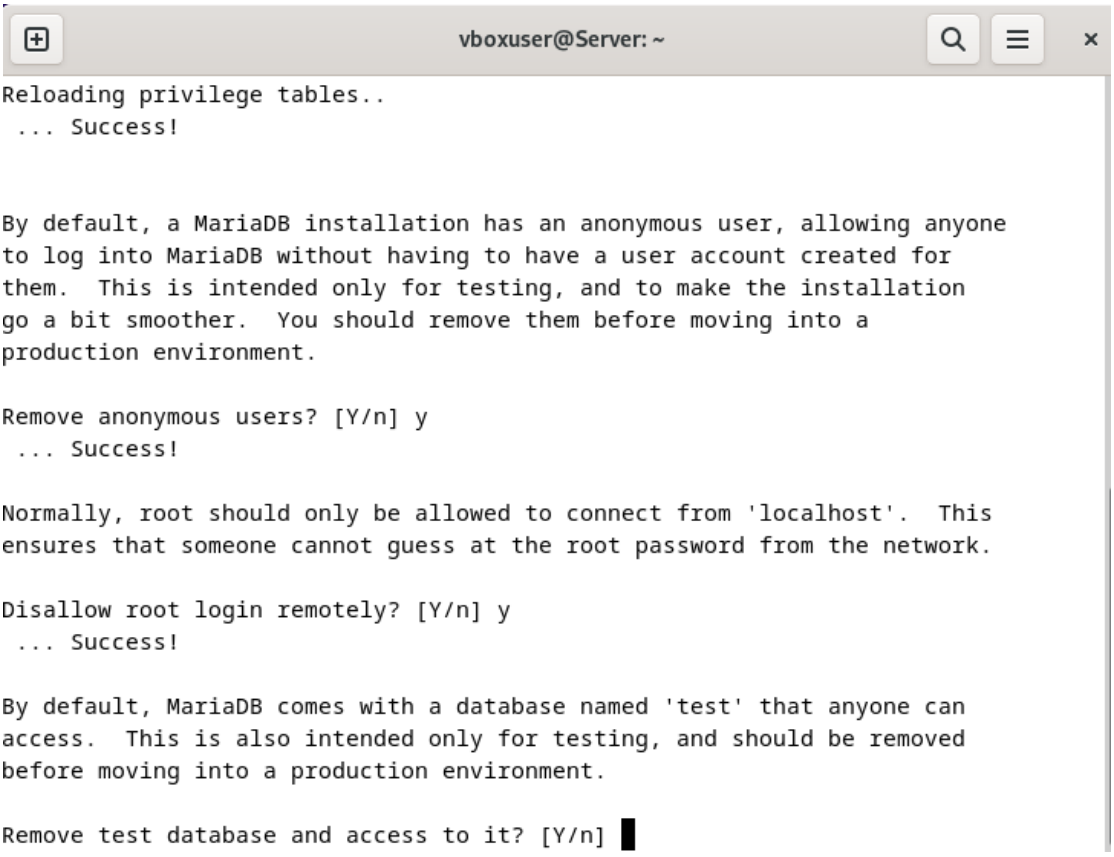
La désactivation de l'accès root depuis l'extérieur.

La suppression de la base de données `test` créée par défaut.

Le rechargement des privilèges pour appliquer les changements.

Cette configuration réduit les risques d'accès non autorisé à la base de données et garantit une meilleure protection du serveur MySQL.

```
vboxuser@Server:~$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n]
```



EXERCICE 4.11 :

Pour faciliter la gestion de la base de données MySQL, l'application web phpMyAdmin a été déployée dans un conteneur Docker lié au conteneur MySQL. L'installation a été réalisée avec la commande :

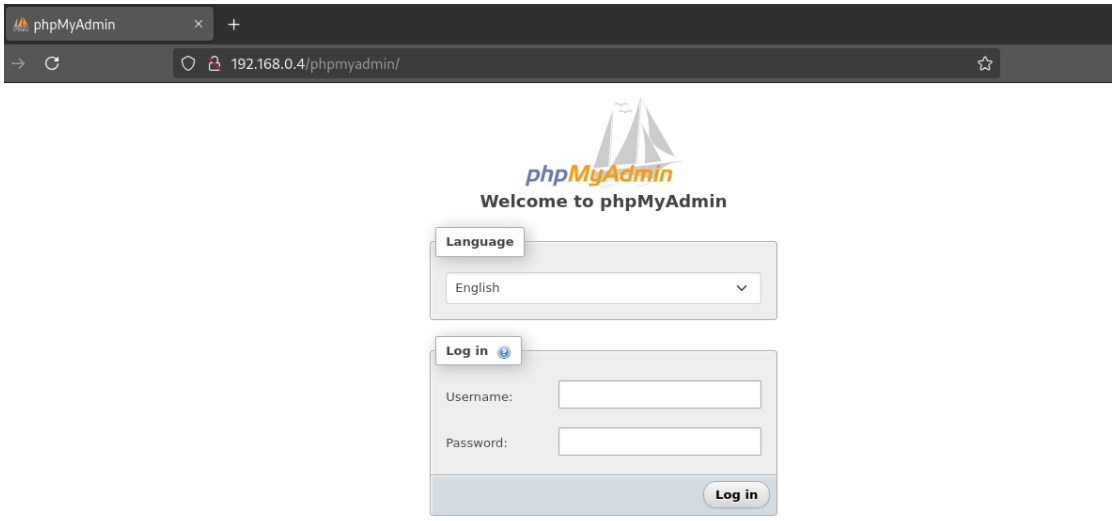
docker run -d --name phpmyadmin-container \

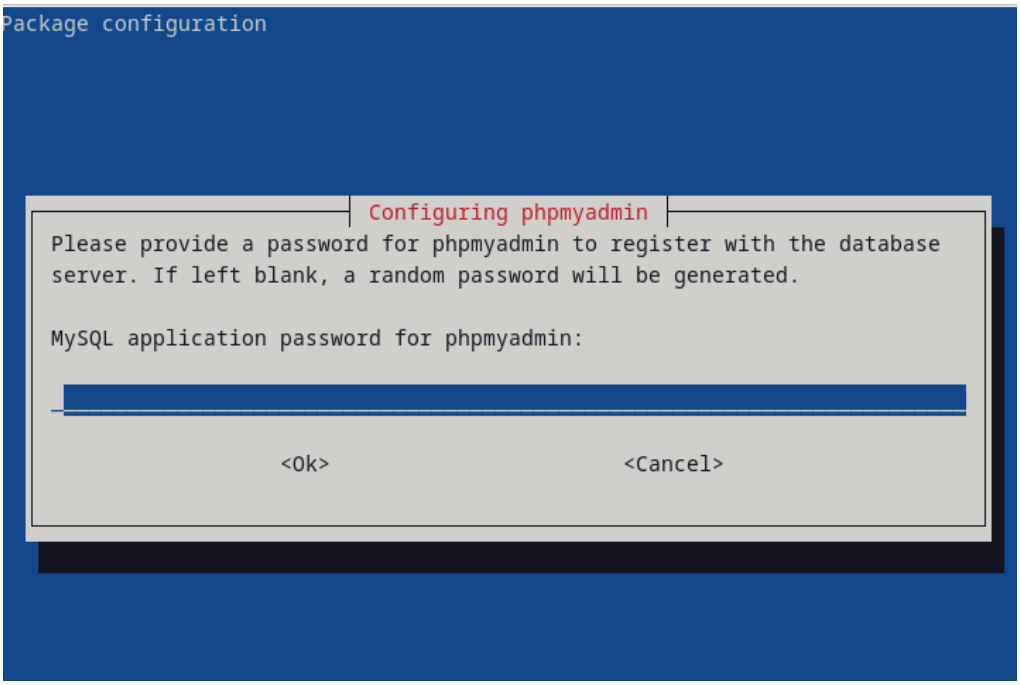
**-e PMA_HOST=mysql-container **

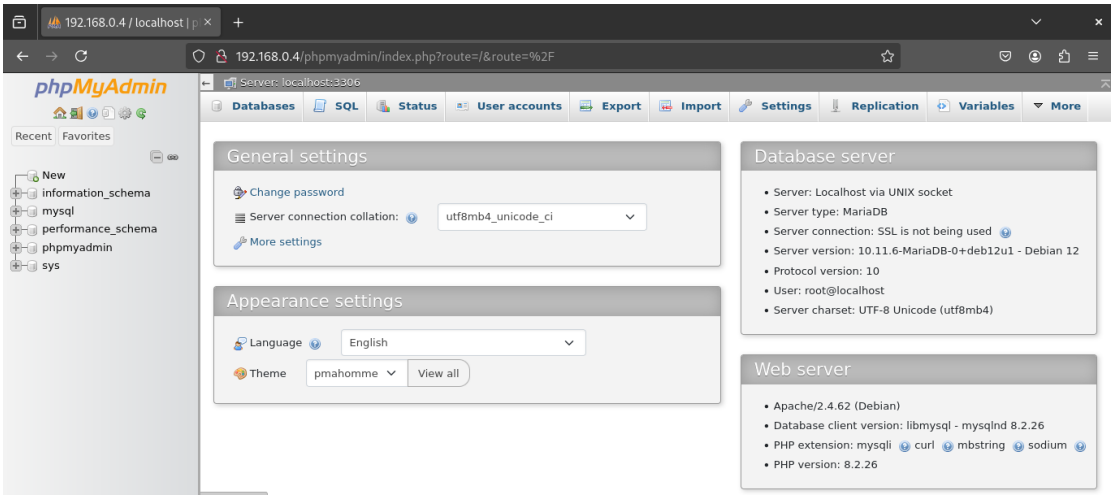
**-p 8081:80 **

Phpmyadmin

Le port 8081 a été exposé et la redirection a été configurée via *iptables*. L'accès à l'interface web a été testé avec succès depuis un web







EXERCICE 4.12 :

Nous avons téléchargé et installé WordPress dans le répertoire d'Apache :

```
cd /var/www/html

sudo wget https://wordpress.org/latest.tar.gz

sudo tar -xzvf latest.tar.gz

sudo mv wordpress/* .

sudo rm -r wordpress latest.tar.gz
```

Nous avons configuré les permissions nécessaires :

```
sudo chown -R www-data:www-data /var/www/html

sudo chmod -R 755 /var/www/html
```

Ensuite, nous avons configuré le fichier wp-config.php avec les informations de connexion à la base de données MySQL :

vboxuser@Server: /var/www/html

```
 GNU nano 7.2                    wp-config.php *
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'root' );

/** Database password */
define( 'DB_PASSWORD', 'changeme' );

/** Database hostname */
define( 'DB_HOST', '192.168.0.4' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Titre du site WordPress

Identifiant admin

Les identifiants ne peuvent utiliser que des caractères alphanumériques, des espaces, des tirets bas ("_"), des traits d'union ("-"), des points et le symbole @.

Mot de passe changeme_wordpress — Masquer

Moyenne

Important : Vous aurez besoin de ce mot de passe pour vous connecter. Pensez à le stocker dans un lieu sûr.

Votre e-mail rochdi.dardor.edu@gmail.com

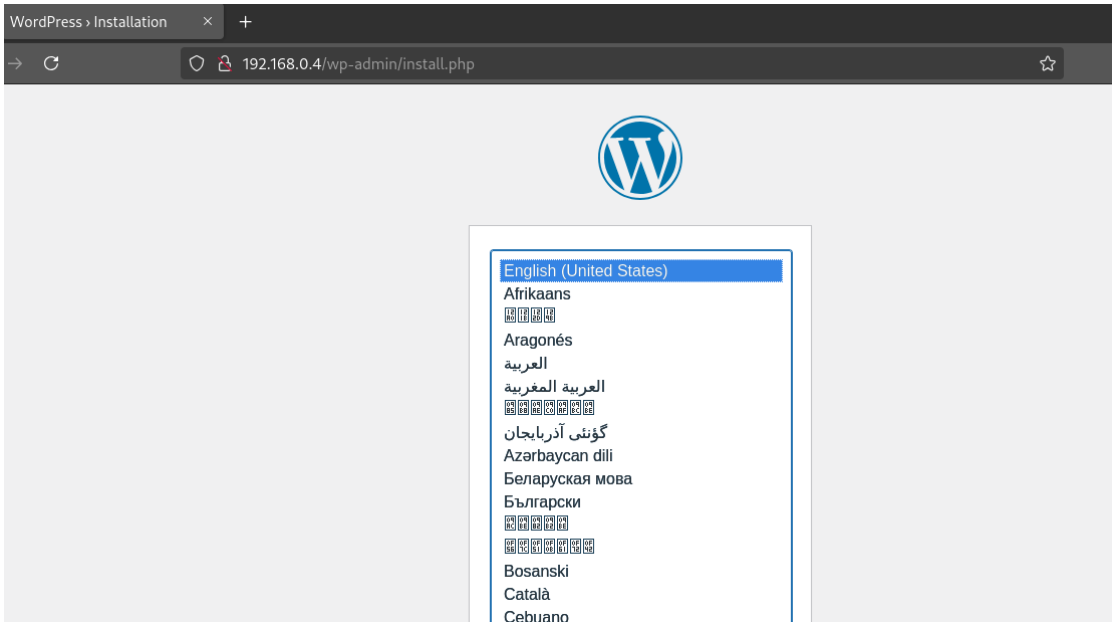
Vérifiez bien cette adresse e-mail avant de continuer.

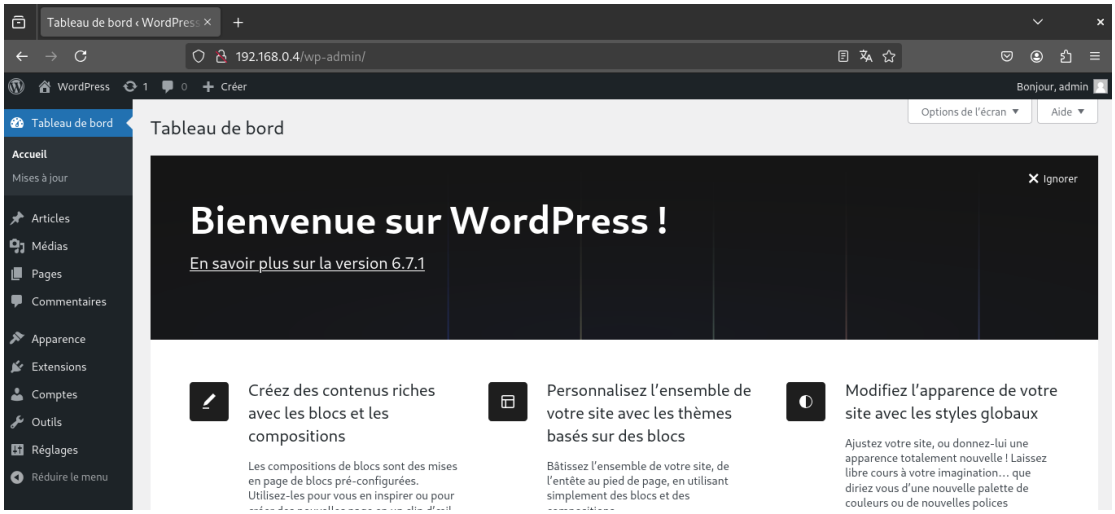
Visibilité par les moteurs de recherche

☑ Demander aux moteurs de recherche de ne pas indexer ce site

Certains moteurs de recherche peuvent décider de l'indexer malgré tout.

Installer WordPress





EXERCICE 4.13 :

Pour sécuriser l'accès au serveur WordPress, un certificat SSL auto-signé a été généré et configuré. La génération du certificat a été réalisée avec :

sudo a2enmod ssl

sudo a2ensite default-ssl

sudo systemctl reload apache2

Nous avons ensuite créé un certificat auto-signé :

**sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 **

**-keyout /etc/ssl/private/apache-selfsigned.key **

-out /etc/ssl/certs/apache-selfsigned.crt

Enfin, nous avons configuré Apache pour utiliser ce certificat.

```
vboxuser@Server:/var/www/html$ sudo systemctl restart apache2
vboxuser@Server:/var/www/html$ sudo a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
vboxuser@Server:/var/www/html$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create s
elf-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
vboxuser@Server:/var/www/html$ systemctl restart apache2
vboxuser@Server:/var/www/html$
```

EXERCICE 4.14 :

Un certificat auto-signé est un certificat SSL généré par l'administrateur du serveur lui-même, sans l'intervention d'une autorité de certification (CA) reconnue. Bien qu'il permette de chiffrer les échanges entre le client et le serveur, plusieurs risques de sécurité y sont associés :

Absence de vérification par une autorité de confiance :

Un certificat auto-signé n'est pas validé par une CA (Autorité de Certification) reconnue.

Les navigateurs web ne peuvent pas vérifier l'authenticité du certificat, ce qui entraîne l'affichage d'un avertissement de sécurité.

Vulnérabilité aux attaques de type MITM (Man-in-the-Middle) :

Un attaquant pourrait générer un certificat auto-signé similaire et l'utiliser pour intercepter le trafic.

Un certificat émis par une CA reconnue garantit l'identité du serveur et réduit ce risque.

Expérience utilisateur dégradée :

Les navigateurs affichent un avertissement de sécurité indiquant que le certificat ne peut pas être vérifié.

Cela peut dissuader les utilisateurs d'accéder au site web ou les inciter à ignorer d'autres avertissements critiques.

Usage recommandé :

Les certificats auto-signés conviennent pour des environnements de test ou des intranets fermés.

Pour un site public ou professionnel, il est recommandé d'utiliser un certificat émis par une autorité reconnue, comme *Let's Encrypt*.

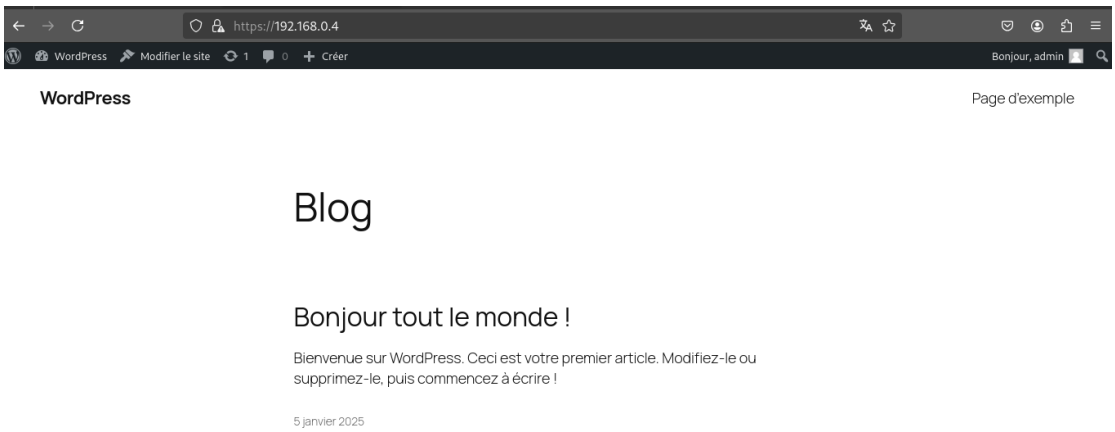
Recommandation :

Pour un environnement de production, l'utilisation de **Let's Encrypt** est fortement recommandée. Il permet de :

Obtenir des certificats gratuits et reconnus par la majorité des navigateurs.

Automatiser le renouvellement du certificat.

Renforcer la crédibilité et la sécurité du site web



Serveur NFS

Dans cette section, nous avons configuré un serveur NFS afin de partager les dossiers utilisateurs sur le réseau interne de l'entreprise. L'objectif est de permettre aux employés d'accéder à leurs données depuis n'importe quelle station de travail connectée au réseau.

EXERCICE 4.15 :

1. Installation et configuration du serveur NFS
   Nous avons installé le paquet nécessaire au serveur NFS avec la commande suivante :

   sudo apt update
   sudo apt install nfs-kernel-server -y

2. Création du répertoire à partager
   Un répertoire destiné à contenir les données des utilisateurs a été créé :

   sudo mkdir -p /srv/nfs/shared

3. Définition des permissions
   Nous avons défini les permissions nécessaires pour permettre l'accès aux utilisateurs :

   sudo chown nobody:nogroup /srv/nfs/shared
   sudo chmod 755 /srv/nfs/shared

4. Configuration des exports
   Le fichier de configuration des exports a été modifié pour partager le répertoire :

   sudo nano /etc/exports

Contenu ajouté au fichier :

/srv/nfs/shared 192.168.0.0/24(rw,sync,no_subtree_check)

La commande suivante a été utilisée pour appliquer la nouvelle configuration :

sudo exportfs -a

5. Redémarrage du service NFS
   Pour que les modifications prennent effet, nous avons redémarré le service NFS : **sudo systemctl restart nfs-kernel-server**

6. Test sur la machine cliente
   Sur une machine cliente, nous avons installé le paquet nécessaire : **sudo apt install nfs-common -y**

   Nous avons ensuite monté le dossier partagé :

   sudo mkdir -p /mnt/nfs_shared

   sudo mount 192.168.0.4:/srv/nfs/shared /mnt/nfs_shared

Enfin, nous avons vérifié que le montage était correct :

df -h

Capture d'écran :

```
root@Client1:~# sudo mkdir -p /mnt/nfs_shared
root@Client1:~# sudo mount 192.168.0.4:/srv/nfs/shared /mnt/nfs_shared
root@Client1:~# df -h
Filesystem                   Size  Used Avail Use% Mounted on
udev                         953M     0  953M   0% /dev
tmpfs                        197M  1.3M  196M   1% /run
/dev/sda1                    8.9G  5.7G  2.7G  69% /
tmpfs                        984M     0  984M   0% /dev/shm
tmpfs                        5.0M  8.0K  5.0M   1% /run/lock
192.168.0.1:/srv/nfs/home    8.9G  5.6G  2.8G  67% /mnt/nfs_home
tmpfs                        197M   96K  197M   1% /run/user/1000
192.168.0.4:/srv/nfs/shared  8.9G  7.2G  1.2G  87% /mnt/nfs_shared
root@Client1:~#
```

EXERCICE 4.16 :

La configuration mise en place permet aux utilisateurs d'accéder au répertoire partagé via le réseau interne. Toutefois, cette solution présente certaines limitations en matière de sécurité des droits utilisateurs :

- Problème de mapping des utilisateurs
  NFS repose sur le système d'UID (identifiant utilisateur unique) pour gérer les droits d'accès. Si les utilisateurs sur le serveur et les clients n'ont pas les mêmes UID, les permissions peuvent être mal appliquées. Par conséquent, les restrictions d'accès aux fichiers ne sont pas nécessairement toujours vérifiées correctement.

- Accès anonyme
  Dans notre configuration, nous avons utilisé l'utilisateur « nobody » pour les accès NFS anonymes. Cela signifie que tous les accès non authentifiés sont mappés sur cet utilisateur, limitant ainsi les risques d'accès non autorisé à des fichiers sensibles.

- Sécurisation des communications
  Par défaut, NFS ne chiffre pas les communications entre le client et le serveur, ce qui pourrait permettre à un attaquant sur le réseau de lire ou de modifier les

données échangées. Pour pallier ce problème, il serait préférable d'utiliser un tunnel SSH ou un VPN pour sécuriser les échanges.

En conclusion, bien que NFS facilite le partage de fichiers au sein de l'entreprise, il est important de s'assurer que les utilisateurs ont des UID synchronisés entre le serveur et les clients, et de mettre en place des solutions pour sécuriser les communications.

Jenkins

EXERCICE 4.17 :

Nous avons téléchargé et installé Apache Tomcat :

wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.98/bin/apache-tomcat-9.0.98.tar.gz

tar -xzvf apache-tomcat-9.0.98.tar.gz

sudo mv apache-tomcat-9.0.98 /opt/tomcat9

Nous avons configuré les permissions :

sudo chown -R $(whoami):$(whoami) /opt/tomcat9

chmod +x /opt/tomcat9/bin/*.sh

Nous avons ensuite démarré Tomcat :

/opt/tomcat9/bin/startup.sh

Nous avons téléchargé l'archive jenkins.war et l'avons placée dans le répertoire webapps de Tomcat :

wget http://updates.jenkins-ci.org/download/war/latest/jenkins.war

sudo mv jenkins.war /opt/tomcat9/webapps/

Enfin, nous avons redémarré Tomcat pour que Jenkins soit déployé :

/opt/tomcat9/bin/shutdown.sh

/opt/tomcat9/bin/startup.sh

Capture d'écran :