

## Job 01

### Faire les trois VM serveur:

ne pas oublier dans chaque VM de modifier le `/etc/sources.list` ainsi que de mettre les outils sudo (`apt install sudo`) en **ROOT** et de vérifier que le compte utilisateur a reçu les droits sudo (`whoami -sudo`) [pour revenir en **ROOT** sans redémarrer la machine ( **SU-**)]

### En priorité installer sur chaque VM

#### 1. Installation de Avahi-daemon :

Installez Avahi sur votre VM FTP ainsi que sur les autres VMs où vous souhaitez **pouvoir accéder au serveur FTP, SQL et Web par son nom.**

```
sudo apt update
sudo apt install avahi-daemon
```

#### 2. Démarrer et vérifier le statut d'Avahi :

Après l'installation, démarrez **Avahi** et vérifiez qu'il est bien actif :

```
sudo systemctl start avahi-daemon
sudo systemctl enable avahi-daemon
sudo systemctl status avahi-daemon
```

### 3. Accéder à la VM FTP via son nom :

Par défaut, le nom de votre machine est basé sur le nom d'hôte. Vous pouvez vérifier le nom d'hôte de votre VM FTP :

```
hostname
```

```
root@laurent-client:~# hostname  
laurent-client  
root@laurent-client:~# _
```

Désormais, sur les autres machines de votre réseau, vous pouvez accéder à la **VM FTP** via son nom d'hôte suivi de **.local** :

```
root@laurent-client:~# ping laurent-ftp.local  
PING laurent-ftp.local (10.10.76.112) 56(84) bytes of data.  
64 bytes from 10.10.76.112: icmp_seq=1 ttl=64 time=0.430 ms
```

```
root@laurent-client:~# ping laurent-sql.local  
PING laurent-sql.local (10.10.76.154) 56(84) bytes of data.  
64 bytes from 10.10.76.154: icmp_seq=1 ttl=64 time=1.62 ms
```

```
root@laurent-client:~# ping laurent-web.local  
PING laurent-web.local (10.10.76.155) 56(84) bytes of data.  
64 bytes from 10.10.76.155: icmp_seq=1 ttl=64 time=0.850 ms
```

Tout ping avec le nom complet de la machine et lors de changement d'ip Avahi-daemon se charge de toujours faire le rapport ip et nom complet machine

## 1er VM WEB

Création de la **VM laurent\_web**

**ne pas oubliez dans chaque VM de modifier le /etc/sources.list ainsi que de mettre les outils sudo (apt install sudo ) en ROOT et de vérifier que le compte utilisateur a reçu les droits sudo (whoami -sudo) pour revenir en ROOT sans redémarrer la machine ( SU-)**

Permet de tester le ping par le nom complet de la machine (remplace l'ip par le nom de la machine):

**apt install avahi-daemon**

puis

tapez : **ping laurent-web.local**

```
root@laurentweb:~# ping laurent-web.local
PING laurent-web.local (192.168.147.130) 56(84) bytes of data.
64 bytes from 192.168.147.130 (192.168.147.130): icmp_seq=1 ttl=64 time=0.026 ms
64 bytes from 192.168.147.130 (192.168.147.130): icmp_seq=2 ttl=64 time=0.036 ms
```

## Installation outils pour la VM WEB:

Installation de NGINX

1. **Mise à jour du système** : Avant d'installer NGINX, il est recommandé de mettre à jour votre système pour s'assurer que tous les paquets sont à jour.

```
sudo apt update && sudo apt upgrade -y
```

**2. Installation de NGINX :** Utilisez le gestionnaire de paquets **apt** pour installer NGINX. (version freeze sous debian 12).

```
sudo apt install nginx -y
```

**3. Démarrer NGINX :** Après l'installation, démarrez le service NGINX.

```
sudo systemctl start nginx
```

**4. Activer NGINX au démarrage :** Pour que **NGINX** démarre automatiquement au démarrage de la machine, activez le service.

```
sudo systemctl enable nginx
```

## 2eme VM SQL

Création de la **VM laurent\_sql**

**ne pas oubliez dans chaque VM de modifier le /etc/sources.list**

**ainsi que de mettre les outils sudo (apt install sudo ) en ROOT**

**et de vérifier que le compte utilisateur a reçu les droits sudo (whoami -sudo)**

**pour revenir en ROOT sans redémarrer la machine ( SU-)**

Permet de tester le ping par le nom complet de la machine (remplace l'ip par le nom de la machine):

**apt install avahi-daemon**

puis

tapez : **ping laurent-sql.local**

```
root@laurent-sql:~# ping laurent-sql.local
PING laurent-sql.local (192.168.147.132) 56(84) bytes of data.
64 bytes from 192.168.147.132 (192.168.147.132): icmp_seq=1 ttl=64 time=0.033 ms
64 bytes from 192.168.147.132 (192.168.147.132): icmp_seq=2 ttl=64 time=0.039 ms
```

# Installation de l'outils MariaDB :

## 1. Mise à jour des paquets

Avant d'installer quoi que ce soit, il est recommandé de mettre à jour les paquets :

```
sudo apt update && sudo apt upgrade -y
```

## 2. Installation de MariaDB

Ensuite, installe MariaDB en exécutant la commande suivante :

```
sudo apt install mariadb-server -y
```

## 3. Sécurisation de l'installation

MariaDB fournit un script de sécurité pour renforcer l'installation par défaut :

```
sudo mysql_secure_installation
```

Tu seras invité à :

- Définir un mot de passe root
- Supprimer les utilisateurs anonymes
- Désactiver l'accès root à distance
- Supprimer la base de données de test
- Recharger les privilèges

## 4. Démarrage et activation de MariaDB

Assure-toi que MariaDB démarre automatiquement à chaque démarrage :

```
sudo systemctl start mariadb  
sudo systemctl enable mariadb
```

## 3ème VM FTP

Création de la **VM laurent\_ftp**

**ne pas oublier dans chaque VM de modifier le /etc/sources.list ainsi que de mettre les outils sudo (apt install sudo ) en ROOT et de vérifier que le compte utilisateur a reçu les droits sudo (whoami -sudo) pour revenir en ROOT sans redémarrer la machine ( SU-)**

Permet de tester le ping par le nom complet de la machine (remplace l'ip par le nom de la machine):

**apt install avahi-daemon**

puis

tapez : **ping laurent-ftp.local**

```
Last login: Mon Sep 30 08:38:11 CEST 2024 on tty1  
root@laurent-client:~# ping laurent-ftp.local  
PING laurent-ftp.local (10.10.76.112) 56(84) bytes of data.  
64 bytes from 10.10.76.112: icmp_seq=1 ttl=64 time=0.793 ms
```

## Installation de l'outils Proftpd :

Installer proftpd :

```
apt install proftpd -y
```

On modifie ensuite le fichier /etc/proftpd/proftpd.conf

```
root@ftp:~# nano /etc/proftpd/proftpd.conf
```

Et on y ajoute ceci :

```
<Global>
RootLogin off
</Global>

DefaultRoot ~
```

On y crée ensuite le compte monitor :

```
adduser monitor
usermod -aG sudo monitor
```

Et on le configure pour qu'il se connecte avec des clés SSH :

```
mkdir /home/monitor/.ssh
chmod 700 /home/monitor/.ssh
touch /home/monitor/.ssh/authorized_keys
chmod 600 /home/monitor/.ssh/authorized_keys
```

Après on désactive l'accès root en SSH :

```
root@ftp:~# nano /etc/ssh/sshd_config
```

```
GNU nano 7.2 /etc/ssh/sshd config *
# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication. Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin prohibit-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
UsePAM yes
PermitRootLogin no
PasswordAuthentication no
#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
```

On configure ensuite le pare-feu en y autorisant uniquement le port FTP (21) et le port SSH (22)

```
ufw allow 21/tcp
ufw allow 22/tcp
ufw enable
```

---

## Job 02

### Faire la VM Client:

Création de la **VM laurent-client**

Permet de tester le ping par le nom complet de la machine (remplace l'ip par le nom de la machine):

**apt install avahi-daemon**

puis

tapez : **ping laurent-client.local**

### Configuration sécurité SSH de la VM WEB, FTP, SQL et client:

**Mettre les serveurs en accès ssh libre avec mot de passe pour envoyer les clés sinon y aura un résultat failed**

**Création du compte monitor :**

- Créez un utilisateur nommé **monitor** sur chaque VM :

```
root@laurent-web:/etc/ssh# adduser monitor_
```

```
root@laurent-web:/etc/ssh# usermod -aGsudo monitor_
```

**Configurez l'authentification par clé SSH :**



- Créez le répertoire SSH pour l'utilisateur **monitor** :

```
root@laurent-web:/etc/ssh# mkdir /home/monitor/.ssh  
root@laurent-web:/etc/ssh#
```

on lui donne les droits:

```
root@laurent-web:/etc/ssh# chmod 700 /home/monitor/.ssh  
root@laurent-web:/etc/ssh#
```

pour se faire démarrer les VM puis de la cliente créer les clés ssh public et privé avec cette commande:

```
ssh-keygen -t rsa
```

Clé publique vers la VM WEB, FTP, SQL

```
ssh-copy-id utilisateur@ip_de_vm
```

Exemple pour le compte **monitor**

```
monitor@laurent-ftp:~$ ssh monitor@10.10.67.146
```

**Configurer l'authentification par clé SSH :**

```
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
```

```
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#ChallengeResponseAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication. Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin prohibit-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
UsePAM yes

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
```

```
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem          sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#      X11Forwarding no
#      AllowTcpForwarding no
#      PermitTTY no
#      ForceCommand cvs server
```

## Désactivation du ROOT en SSH :

Modifiez le fichier de configuration SSH **/etc/ssh/sshd\_config** pour interdire la connexion SSH au compte root.

```
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Redémarrez le service SSH

```
root@laurent-web:/etc/ssh# systemctl restart sshd
```

# Installation de python, les outils MySQL/MariaDB (pas le serveur), client FTP

1. Installer **Python** : Tu peux installer **Python** en exécutant la commande suivante:

```
sudo apt update
sudo apt install python3 python3-pip
```

version :

```
root@laurent-client:~# python3 --version
Python 3.11.2
```

2. Installer les outils MySQL/MariaDB (client uniquement) : Pour installer le client MariaDB

```
sudo apt install mariadb-client
```

version :

```
root@laurent-client:/home/laurent-client# mariadb --version
mariadb Ver 15.1 Distrib 10.11.6-MariaDB, for debian-linux-gnu (x86_64) using EditLine wrapper
```

3. installer un client FTP : Pour installer un client FTP

```
sudo apt install ftp
```

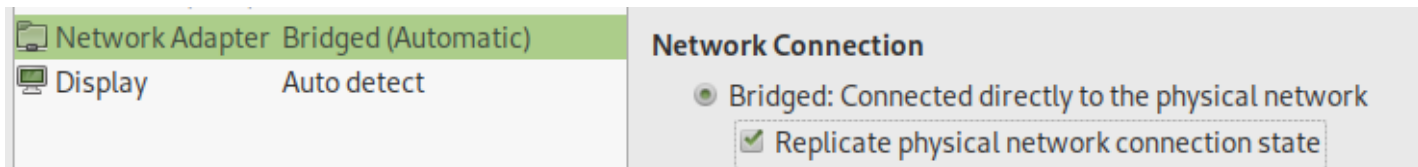
version :

```
root@laurent-client:~# dpkg -l | grep ftp
ii  ftp                20210827-4      all          dummy transitional package for tftp
ii  openssh-sftp-server 1:9.2p1-2+deb12u3 amd64        secure shell (SSH) sftp server module, for SFTP access from remote machines
ii  tftp                20210827-4+b1   amd64        enhanced ftp client
```

# Placer les VM sur le réseau la plateforme

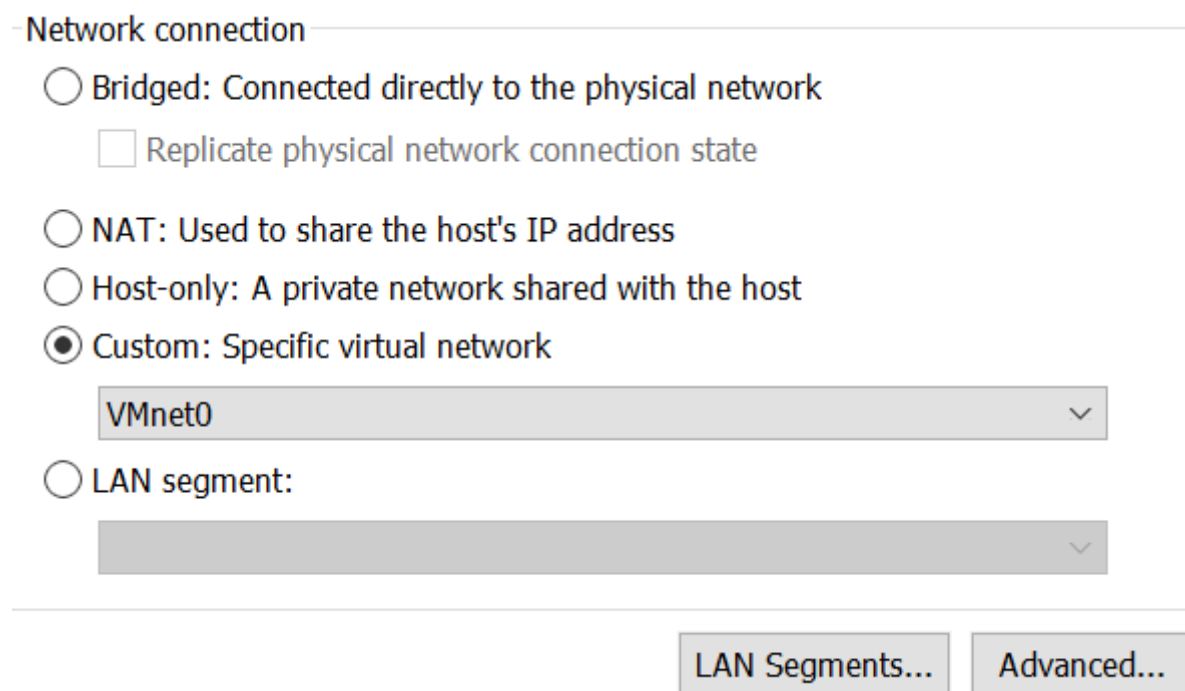
## Nota pour installer la VM sur le réseau La Plateforme

Pour ajouter la VM au réseau La Plateforme, il faut soit aller sur VMWare et mettre le réseau de la VM en bridge en cochant ces deux cases :



soit (si on a un dongle Wifi notamment) aller dans : Edit - Virtual Network Editor et créer un réseau en Bridge qui utilise la carte réseau connectée à la Plateforme (ici en l'occurrence, le dongle Wifi) :

Après il suffit de mettre la carte réseau de la VM en custom et en choisissant le réseau bridge qu'on a créer :



Vérifier que les VM sont sur le réseau de “la plateforme” avec la commande IP a :

### VM FTP

```
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:46:c8:5a brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.76.112/16 brd 10.10.255.255 scope global dynamic ens32
```

## VM SQL

```
valid_ifn forever preferred_ifn forever
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:a2:b6:a9 brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.76.154/16 brd 10.10.255.255 scope global dynamic ens32
    link-local 168.0.0.100
```

## VM WEB

```
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:a8:0d:0a brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    inet 10.10.76.155/16 brd 10.10.255.255 scope global dynamic ens32
    link-local 168.0.0.100
```

## VM CLIENT

```
valid_ifn forever preferred_ifn forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:e8:29:d0 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 10.10.83.33/16 brd 10.10.255.255 scope global dynamic ens33
    link-local 168.0.0.100
```

---

## Job 03:

faire un script **“ssh\_login.py”**, qui depuis la VM client , va se connecter sur un des serveurs on choisir FTP et lancer une commande shell (par exemple «df» ou «ls» ou autre).

## VM Client:

**le script se situe vers =>cd /home/laurent-client**

### Création du script avec Nano:

```
# Utilisation du script avec la fonction principale
if __name__ == "__main__":
    # Informations d'authentification pour le serveur SSH
    hostname = 'laurent-ftp.local' # Remplacez par l'adresse IP de votre serveur
    username = 'monitor' # Remplacez par le nom d'utilisateur SSH

    # Saisie sécurisée du mot de passe
```

donnez les droits au script avec : **CHMOD +X (nom du script.py)**

lancement du script avec la commande :

```
root@laurent-client:/home/laurent-client# ./ssh_login.py _
```

Demande mot de passe SSH: **MONITOR**

```
root@laurent-client:/home/laurent-client# ./ssh_login.py
Entrez le mot de passe SSH : _
```

Affichage de la commande **df -h** : cela affiche

```
Résultat de la commande 'df -h':
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
udev                441M      0  441M   0% /dev
tmpfs                93M    652K   92M   1% /run
/dev/sda1            6,9G    1,8G  4,8G  28% /
tmpfs                463M      0  463M   0% /dev/shm
tmpfs                5,0M      0   5,0M   0% /run/lock
tmpfs                93M      0   93M   0% /run/user/1001
```

La commande **df -h** est utilisée dans les systèmes Unix et Linux pour afficher des informations sur l'utilisation de l'espace disque. Voici ce que fait chaque partie de la commande :

- **df** : C'est l'acronyme de "disk filesystem". Cette commande permet d'afficher l'espace disque utilisé et disponible sur les systèmes de fichiers montés.
  - **-h** : C'est une option qui signifie "human-readable" (lisible par l'homme). Elle permet d'afficher les tailles en unités facilement compréhensibles, comme Ko (kilo octets), Mo (mégaoctets), Go (gigaoctets), etc.
-



## Job 04:

En utilisant le script précédent, vous devez faire un script **“ssh\_login\_sudo.py”**, qui depuis la VM, va se connecter sur un des serveurs et lancer une commande shell, cette fois en mode **«sudo»**.

## VM Client:

**le script se situe vers =>cd /home/laurent-client**

### Création du script avec Nano:

```
if __name__ == "__main__":
    # Informations d'authentification pour le serveur SSH
    hostname = 'laurent-ftp.local' # Remplace par l'adresse IP de ton serveur
    username = 'monitor' # Remplace par le nom d'utilisateur SSH

    # Saisie sécurisée du mot de passe SSH
    ssh_password = getpass("Entrez le mot de passe SSH : ")

    # Saisie sécurisée du mot de passe sudo
    sudo_password = getpass("Entrez le mot de passe sudo : ")

    # Commande shell à exécuter avec sudo
    command_to_execute = "df -h" # Exemple de commande à exécuter avec sudo

    # Appel de la fonction pour exécuter la commande via SSH avec sudo
    ssh_execute_command_sudo(hostname, username, ssh_password, sudo_password, command_to_execute)
```

donnez les droits au script avec : **CHMOD +X (nom du script.py)**

### lancement du script avec la commande :

```
laurent-client@laurent-client:~$ ./ssh_login_sudo.py _
```

Demande mot de passe SSH: **MONITOR**

```
Entrez le mot de passe SSH :
```

Demande mot de passe SUDO: **MONITOR**

```
Entrez le mot de passe SSH :
Entrez le mot de passe sudo :
```

```
Entrez le mot de passe SSH :  
Entrez le mot de passe sudo :  
Connexion à laurent-ftp.local en tant que monitor...
```

**Affichage de la commande `ls -al`** : cela affiche

```
laurent-client@laurent-client:~$ ./ssh_login_sudo.py  
Entrez le mot de passe SSH :  
Entrez le mot de passe sudo :  
Connexion à laurent-ftp.local en tant que monitor...  
Connexion réussie.  
Résultat de la commande 'ls -a' avec sudo:  
[sudo] Mot de passe de monitor : .  
  
..  
.bash_history  
.bash_logout  
.bashrc  
.local  
.profile  
readme.txt  
.ssh  
.sudo_as_admin_successful
```

- **ls** : C'est une commande qui signifie "list" (lister). Elle est utilisée pour afficher le contenu d'un répertoire.
- **-a** : C'est une option qui signifie "all" (tout). Elle permet d'inclure les fichiers et répertoires cachés dans la sortie. Les fichiers cachés sous Linux sont ceux dont le nom commence par un point (.), comme **.bashrc**.
- **-l** : C'est une option qui signifie "long format" (format long).

---

## **Job 05 :**

Maintenant que l'on peut lancer des commandes shell en «**sudo**», on va se connecter sur le serveur **MariaDB**.

Vous devez faire un script "**ssh\_mariadb.py**", pour vérifier que vous avez bien accès au serveur **MariaDB**.

**le script se situe vers =>cd /home/laurent-client**

le premier script sert à voir les bases de données de Mariadb stocké sur le serveur **laurent-sql.local**:

**[script pure pour utiliser en copie ]**

**copier les caractères qui sont en vert !!!!**

**GNU nano 7.2**

**ssh\_sql\_base.py**

```
#!/usr/bin/env python3
```

```
import paramiko
```

```
from getpass import getpass
```

```
def ssh_execute_sql_command(hostname, username, ssh_password, sql_password):
```

```
    # Création d'un objet SSHClient
```

```
    ssh = paramiko.SSHClient()
```

```
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
    try:
```

```
        # Connexion SSH avec le mot de passe SSH
```

```
        print(f"Connexion à {hostname} en tant que {username}...")
```

```
        ssh.connect(hostname, username=username, password=ssh_password)
```

```
        print("Connexion réussie.")
```

```
    # Préparation de la commande pour exécuter une commande SQL via MySQL
```

```
    command_to_execute = f"mysql -u monitor -p'{sql_password}' -h laurent-sql.local -e  
'SHOW DATABASES;'"
```

```
    # Exécution de la commande
```

```
    stdin, stdout, stderr = ssh.exec_command(command_to_execute)
```

### # Lecture de la sortie et des erreurs

```
output = stdout.read().decode('utf-8')  
error_output = stderr.read().decode('utf-8')
```

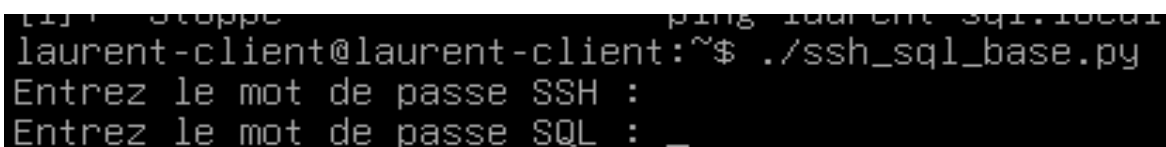
### # Affichage de la sortie de la commande

```
if output:  
    print(f"Résultat de la commande SQL:\n{output}")  
  
if error_output:  
    print(f"Erreur de la commande SQL:\n{error_output}")  
  
except paramiko.AuthenticationException:  
    print("Erreur d'authentification. Veuillez vérifier vos informations d'identification.")  
  
except paramiko.SSHException as e:  
    print(f"Erreur SSH: (e)")
```

donnez les droits au script avec : **CHMOD +X (nom du script.py)**

**le script se situe vers =>cd /home/laurent-client**

**lancement du script avec la commande : ./ssh\_sql\_base.py**



```
laurent-client@laurent-client:~$ ./ssh_sql_base.py  
Entrez le mot de passe SSH :  
Entrez le mot de passe SQL : _
```

tentative de connexion au compte **laurent-sql.local** puis

connexion au compte **monitor** de Mariadb

```
laurent-client@laurent-client:~$ ./ssh_sql_base.py
Entrez le mot de passe SSH :
Entrez le mot de passe SQL :
Connexion à laurent-sql.local en tant que monitor...
```

puis connexion et affichage de toutes les base de données:

```
laurent-client@laurent-client:~$ ./ssh_sql_base.py
Entrez le mot de passe SSH :
Entrez le mot de passe SQL :
Connexion à laurent-sql.local en tant que monitor...
Connexion réussie.
Résultat de la commande SQL:
Database
information_schema
mysql
performance_schema
sys
testdb
```

le deuxième script sert à se connecter a la console Mariadb à distance qui lui est stocké sur le serveur **laurent-sql.local**:

[script pure pour utiliser en copie ]

le script se situe vers =>cd /home/laurent-client

copier les caractères qui sont en vert !!!!

GNU nano 7.2

ssh\_mysql.py

```
#!/usr/bin/env python3
```

```
import paramiko
```

```
from getpass import getpass
```

```
def open_mariadb_console(hostname, username, ssh_password, sql_username,
sql_password):
```

```
    # Création d'un objet SSHClient
```

```
ssh = paramiko.SSHClient()
```

```
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
try:
```

### **# Connexion SSH**

```
print(f'Connexion à {hostname} en tant que {username}...')
```

```
ssh.connect(hostname, username=username, password=ssh_password)
```

```
print("Connexion réussie.")
```

### **# Ouvre une session interactive pour MariaDB**

```
console_command = f"mysql -u {sql_username} -p{sql_password}"
```

```
channel = ssh.invoke_shell()
```

### **# Envoi de la commande pour ouvrir la console MariaDB**

```
channel.send(console_command + '\n')
```

### **# Session interactive avec l'utilisateur**

```
while True:
```

### **# Vérifie s'il y a des données disponibles**

```
if channel.recv_ready():
```

#### **# Lecture et affichage des données de la session MariaDB**

```
    output = channel.recv(1024).decode('utf-8')
```

```
    print(output, end="")
```

### **# Saisie des commandes depuis l'utilisateur**

```
user_input = input()
```

```
if user_input.lower() in ['exit', 'quit']:
```

```
print("Fermeture de la session MariaDB.")
```

```
break
```

donnez les droits au script avec : **CHMOD +X (nom du script.py)**

lancement du script avec la commande :

```
ssh_login.py ssh_login_sudo.py ssh_mariadb.py ssh_sql_base.py
laurent-client@laurent-client:~$ ./ssh_mariadb.py
Entrez le mot de passe SSH :
Entrez le mot de passe MariaDB : _
```

tentative de connexion au compte **laurent-sql.local** puis

connexion au compte **monitor** de Mariadb

```
laurent-client@laurent-client:~$ ./ssh_mariadb.py
Entrez le mot de passe SSH :
Entrez le mot de passe MariaDB :
Connexion à laurent-sql.local en tant que monitor...
```

puis connexion et affichage de la console et on peut administrer une base de données et un utilisateur:

```
laurent-client@laurent-client:~$ ./ssh_mariadb.py
Entrez le mot de passe SSH :
Entrez le mot de passe MariaDB :
Connexion à laurent-sql.local en tant que monitor...
Connexion réussie.

Linux laurent-sql 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Oct  4 14:45:49 2024 from 10.10.87.113
mysql -u monitor -pmonitor
monitor@laurent-sql:~$ mysql -u monitor -pmonitor
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> _
```

## Job 06 :

Récupérer dans les fichiers log de MariaDB/MySQL, les tentatives d'accès depuis un compte / mot de passe incorrect, pour cela faites plusieurs accès avec des utilisateurs/MdP qui ne sont pas bons.

On va stocker ces « accès » dans votre base de données, avec le nom du compte utilisé pour l'accès, la date/heure, et l'IP du poste ayant tenté l'accès. Vous devez écrire le script "ssh\_mysql\_error.py"

### IMPORTANT !!!!! Zone de stockage des logs pour Mariadb :

Les logs d'erreurs de MariaDB sont stockés dans un fichier, dont l'emplacement peut varier selon la configuration du serveur.

Par défaut, sous les distributions **Linux** (comme **Debian**, Ubuntu, CentOS), les fichiers de logs d'erreurs de **MariaDB** sont généralement stockés dans **/var/log/mysql/** ou **/var/log/**, mais l'emplacement exact **dépend** de la **configuration** de **MariaDB**.

**nota:** (Les logs de MariaDB enregistrent plusieurs types d'événements, mais ils sont principalement centrés sur l'activité au niveau de la base de données elle-même).

Pour vérifier l'emplacement du fichier des erreurs de log:

L'emplacement du fichier de log des erreurs peut être modifié dans le fichier de configuration **Mariadb.conf.d**

Voici comment le vérifier :

Ouvrez le fichier de configuration de **Mariadb.conf.d** :

Sur Debian/Ubuntu :

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Sur RedHat/CentOS :

```
sudo nano /etc/my.cnf.d/server.cnf
```

dans le fichier **50-server.cnf** : Recherchez l'option **log\_error** Elle vous indiquera l'emplacement exact du fichier de logs d'erreurs ou faire un autre chemin :



```
[mysqld]  
log_error = /var/log/mysql/error.log
```

**nota :** la ligne ne sera pas activé faut retirer “#” pour passer de commentaire à activer la ligne de commande

---

Vérifier que mariadb à l’install à créer ou pas le dossier mysql ainsi que le fichier du stockage des logs qui se nommera “error.log” si absent le crée :

le Dossier **mysql** est manquant le crée :

```
sudo mkdir -p /var/log/mysql  
sudo chown mysql:mysql /var/log/mysql  
sudo chmod 750 /var/log/mysql
```

**750** veut dire :

**7** : Pour le propriétaire

**5** : Pour le groupe

**0** : Pour les autres utilisateurs

Chiffre	Permission	Description	Notation Symbolique
0	---	Aucun droit	---
1	--x	Exécution uniquement	--x
2	-w-	Écriture uniquement	-w-
3	-wx	Écriture et exécution	-wx
4	r--	Lecture uniquement	r--
5	r-x	Lecture et exécution	r-x
6	rw-	Lecture et écriture	rw-
7	rwX	Lecture, écriture et exécution	rwX

Dossier **mysql** crée:

```
monitor@laurent-sql:/var/log$ ls
alternatives.log  dpkg.log      lastlog
alternatives.log.1  dpkg.log.1    mysql
apt               faillog       private
btmp              installer     README
btmp.1            journal      _runit
```

On va créer le Fichier **“error.log”**

```
sudo touch /var/log/mysql/error.log
sudo chown mysql:mysql /var/log/mysql/error.log
sudo chmod 640 /var/log/mysql/error.log
```

**640** veut dire :

**6** : Pour le propriétaire

**4** : Pour le groupe

**0** : Pour les autres utilisateurs

Chiffre	Permission	Description	Notation Symbolique
0	---	Aucun droit	---
1	--x	Exécution uniquement	--x
2	-w-	Écriture uniquement	-w-
3	-wx	Écriture et exécution	-wx
4	r--	Lecture uniquement	r--
5	r-x	Lecture et exécution	r-x
6	rw-	Lecture et écriture	rw-
7	rwX	Lecture, écriture et exécution	rwX

---

Le Fichier **“error.log”** est créé et les droits **640** ne permettent pas au super utilisateurs comme au utilisateurs d'accéder au contenu du fichier qui stock les logs par mesure de protection donc on peut y avoir accès qu'en root.

```
root@laurent-sql:/var/log/mysql# ls
error.log
```

puis redémarrer Mariadb + verif Status :

```
sudo systemctl restart mariadb
```

```
sudo systemctl status mariadb
```

Générer une erreur dans MariaDB pour tester :

```
mysql -u fakeuser -p
```

et une requête SQL invalide :

```
SELECT * FROM nonexistent_table;
```

créer des erreurs dans le fichier “error.log” pour vérifier qu’il fonctionne comme il se doit!

lire le fichier **/var/log/mysql/error.log**.

```
sudo cat /var/log/mysql/error.log
```

```
root@laurent-sql:/var/log/mysql# cat /var/log/mysql/error.log
```

```
2024-10-05 10:42:59 0 [Note] Starting MariaDB 10.11.6-MariaDB-0+deb12u1 source revision as process 1632
2024-10-05 10:43:00 0 [Note] InnoDB: Compressed tables use zlib 1.2.13
2024-10-05 10:43:00 0 [Note] InnoDB: Number of transaction pools: 1
2024-10-05 10:43:00 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2024-10-05 10:43:00 0 [Note] InnoDB: Using liburing
2024-10-05 10:43:00 0 [Note] InnoDB: Initializing buffer pool, total size = 128.000MiB, chunk size = 2.000MiB
2024-10-05 10:43:00 0 [Note] InnoDB: Completed initialization of buffer pool
2024-10-05 10:43:00 0 [Note] InnoDB: File system buffers for log disabled (block size=512 bytes)
2024-10-05 10:43:00 0 [Note] InnoDB: End of log at LSN=47108
2024-10-05 10:43:00 0 [Note] InnoDB: 128 rollback segments are active.
2024-10-05 10:43:00 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing the file full; Please wait ...
2024-10-05 10:43:00 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
2024-10-05 10:43:00 0 [Note] InnoDB: log sequence number 47108; transaction id 14
2024-10-05 10:43:00 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
2024-10-05 10:43:00 0 [Note] Plugin 'FEEDBACK' is disabled.
2024-10-05 10:43:00 0 [Warning] You need to use --log-bin to make --expire-logs-days or --binlog-expire-logs-seconds work.
2024-10-05 10:43:00 0 [Note] InnoDB: Buffer pool(s) load completed at 241005 10:43:00
2024-10-05 10:43:00 0 [Note] Server socket created on IP: '0.0.0.0'.
2024-10-05 10:43:00 0 [Note] /usr/sbin/mariabdb: ready for connections.
Version: '10.11.6-MariaDB-0+deb12u1' socket: '/run/mysqld/mysqld.sock' port: 3306 Debian 12
2024-10-05 10:43:40 31 [Warning] Access denied for user 'fakeuser'@'localhost'
2024-10-05 10:44:31 32 [Warning] Access denied for user 'monitor'@'localhost' (using password: NO)
2024-10-05 11:29:25 34 [Warning] Access denied for user 'laurent'@'localhost' (using password: YES)
2024-10-05 11:29:40 35 [Warning] Access denied for user 'thierryveutseco'@'localhost' (using password: YES)
2024-10-05 11:30:01 36 [Warning] Access denied for user 'amandineabandonnepas'@'localhost' (using password: YES)
2024-10-05 11:30:47 38 [Warning] Access denied for user 'monitor'@'localhost' (using password: YES)
```

On peut vérifier que les erreurs sont bien sauvegardés dans le fichier **“error.log”**

**Maintenant on peut récupérer les logs avec assurance de ne pas se tromper de fichier !**

2/ On va stocker ces « accès » dans votre base de données, avec le nom du compte utilisé pour l'accès, la date/heure, et l'IP du poste ayant tenté l'accès. Vous devez écrire le script **“ssh\_mysql\_error.py”**

**[script pure pour utiliser en copie ]**

**le script se situe vers =>cd /home/laurent-client**

**copier les caractères qui sont en vert !!!!**

**GNU nano 7.2**

**ssh\_mysql\_error.py**

```
#!/usr/bin/env python3
```

```
import paramiko
```

```
import pymysql
```

```
import re
```

```
from getpass import getpass
```

```
def open_mariadb_console(hostname, username, ssh_password, sql_username,  
sql_password):
```

```
    # Création d'un objet SSHClient
```

```
    ssh = paramiko.SSHClient()
```

```
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
    try:
```

```
        # Connexion SSH
```

```
        print(f"Connexion à {hostname} en tant que {username}...")
```

```
        ssh.connect(hostname, username=username, password=ssh_password)
```

```
        print("Connexion réussie.")
```

## **# Ouvre une session interactive pour MariaDB**

```
console_command = f"mysql -u {sql_username} -p{sql_password}"  
  
channel = ssh.invoke_shell()
```

## **# Lecture du fichier de logs d'erreurs**

```
error_log_file = "/var/log/mysql/error.log"  
  
command = f"cat {error_log_file}"  
  
stdin, stdout, stderr = ssh.exec_command(command)  
  
error_log_output = stdout.read().decode('utf-8')
```

## **# Regex pour extraire les tentatives de connexion échouées**

```
pattern = r"\[(.*?)\] .*Access denied for user '(.*?)'@(.*?)"
```

## **# Connexion à la base de données MariaDB pour stocker les logs**

```
connection = pymysql.connect(  
  
host=hostname,  
  
user=sql_username,  
  
password=sql_password  
  
)
```

```
try:
```

```
with connection.cursor() as cursor:
```

### **# Création de la base de données si elle n'existe pas**

```
create_database_query = "CREATE DATABASE IF NOT EXISTS logs_db"  
  
cursor.execute(create_database_query)
```

### **# Sélection de la base de données**

```
cursor.execute("USE logs_db")
```

### **# Création de la table s'il n'existe pas**

```
create_table_query = """  
CREATE TABLE IF NOT EXISTS login_attempts (  
id INT AUTO_INCREMENT PRIMARY KEY,
```

```
username VARCHAR(50),  
ip_address VARCHAR(50),  
attempt_time DATETIME  
)  
"""
```

```
cursor.execute(create_table_query)
```

### **# Extraire les informations des tentatives de connexion échouées**

```
for match in re.finditer(pattern, error_log_output):  
attempt_time, username, ip_address = match.groups()
```

### **# Insérer les informations dans la table login\_attempts**

```
insert_query = """  
INSERT INTO login_attempts (username, ip_address, attempt_time)  
VALUES (%s, %s, %s)  
"""
```

```
cursor.execute(insert_query, (username, ip_address, attempt_time))
```

### **# Commit les changements dans la base de données**

```
connection.commit()
```

### **# Afficher le contenu de la table login\_attempts**

```
cursor.execute("SELECT * FROM login_attempts")
```

```
results = cursor.fetchall()
```

```
print("\nTentatives de connexion enregistrées :")
```

```
for row in results:
```

```
print(f"ID: {row[0]}, Username: {row[1]}, IP Address: {row[2]}, Attempt Time: {row[3]}")
```

### **# Afficher les bases de données**

```
print("\nBases de données existantes :")
```

```
cursor.execute("SHOW DATABASES")
```

```
db_results = cursor.fetchall()
```

```
for db in db_results:
```

```
print(f"- {db[0]}")
```

### **# Afficher les tables dans la base de données logs\_db**

```
cursor.execute("USE logs_db")
```

```
print("\nTables dans la base de données 'logs_db' :")
```

```
cursor.execute("SHOW TABLES")
```

```
tables = cursor.fetchall()
```

```
for table in tables:
```

```
print(f"- {table[0]}")
```

```
finally:
```

```
connection.close() # Assurez-vous que la connexion à la base de données est  
fermée
```

## **# Session interactive avec l'utilisateur**

```
while True:
```

## **# Vérifie s'il y a des données disponibles**

```
if channel.recv_ready():
```

```
    output = channel.recv(1024).decode('utf-8')
```

```
        print(output, end="")
```

## **# Saisie des commandes depuis l'utilisateur**

```
    user_input = input()
```

```
    if user_input.lower() in ['exit', 'quit']:
```

```
        print("Fermeture de la session MariaDB.")
```

```
        break
```

## **# Envoie la commande à la console MariaDB**

```
    channel.send(user_input + '\n')
```

```
except Exception as e:
```

```
    print(f"Erreur : {e}")
```

```
finally:
```

```
    ssh.close() # Assurez-vous que la connexion SSH est fermée
```

```
if __name__ == "__main__":
```

```
    ssh_host = "laurent-sql.local" # Remplace par le bon hostname ou IP
```

```
    ssh_user = "monitor"
```

```
    ssh_password = getpass("Entrez le mot de passe SSH : ")
```

```
    sql_username = "monitor"
```



```
sql_password = getpass("Entrez le mot de passe SQL : ")
```

```
open_mariadb_console(ssh_host, ssh_user, ssh_password, sql_username,  
sql_password)
```

donnez les droits au script avec : **CHMOD +X (nom du script.py)**

lancement du script avec la commande :

```
laurent-client@laurent-client:~$ ./ssh_mysql_error.py
```

connexion au compte **laurent-sql.local (SSH)**

connexion au compte **monitor** du serveur et de l'appli Mariadb

```
laurent-client@laurent-client:~$ ./ssh_mysql_error.py  
Entrez le mot de passe SSH :  
Entrez le mot de passe SQL :  
Connexion à laurent-sql.local en tant que monitor...  
Connexion réussie.  
  
Tentatives de connexion enregistrées :  
  
Bases de données existantes :  
- information_schema  
- logs_db  
- mysql  
- performance_schema  
- sys  
- testdb  
  
Tables dans la base de données 'logs_db' :  
- login_attempts  
Linux laurent-sql 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64
```

puis on peut voir la création de la base **logs\_db** puis récupération des logs du fichiers “**error.log**” dans la base **logs\_db**

**on a récupéré les logs pour les stocker dans la base de mariadb de manière à pouvoir les utiliser ou les protéger contre la suppression ou le vol des logs du serveur!!**

## [pour lire les logs dans Mariadb tapez:](#)

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| ftp_logs_db |
| information_schema |
| lo |
| logs_db |
| mysql |
| performance_schema |
| sys |
| testdb |
| testlo |
+-----+
9 rows in set (0,001 sec)

MariaDB [(none)]> use ftp_logs_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [ftp_logs_db]> SELECT * FROM ftp_logs_db;
ERROR 1146 (42S02): Table 'ftp_logs_db.ftp_logs_db' doesn't exist
MariaDB [ftp_logs_db]> DESCRIBE ftp_log_db;
ERROR 1146 (42S02): Table 'ftp_logs_db.ftp_log_db' doesn't exist
MariaDB [ftp_logs_db]> show tables;
+-----+
| Tables_in_ftp_logs_db |
+-----+
| ftp_login_attempts |
+-----+
1 row in set (0,000 sec)

MariaDB [ftp_logs_db]> SELECT * FROM ftp_login_attempts;
+----+-----+-----+-----+
| id | username | attempt_time | ip_address |
+----+-----+-----+-----+
| 1 | test_user | 2024-10-05 17:26:49 | 192.168.1.10 |
+----+-----+-----+-----+
1 row in set (0,000 sec)
```

## [Job 07 :](#)

installer **FileZilla** sur votre machine perso et accéder au serveur laurent-ftp et depuis la **VM-client** en CLI vers le serveur laurent-ftp

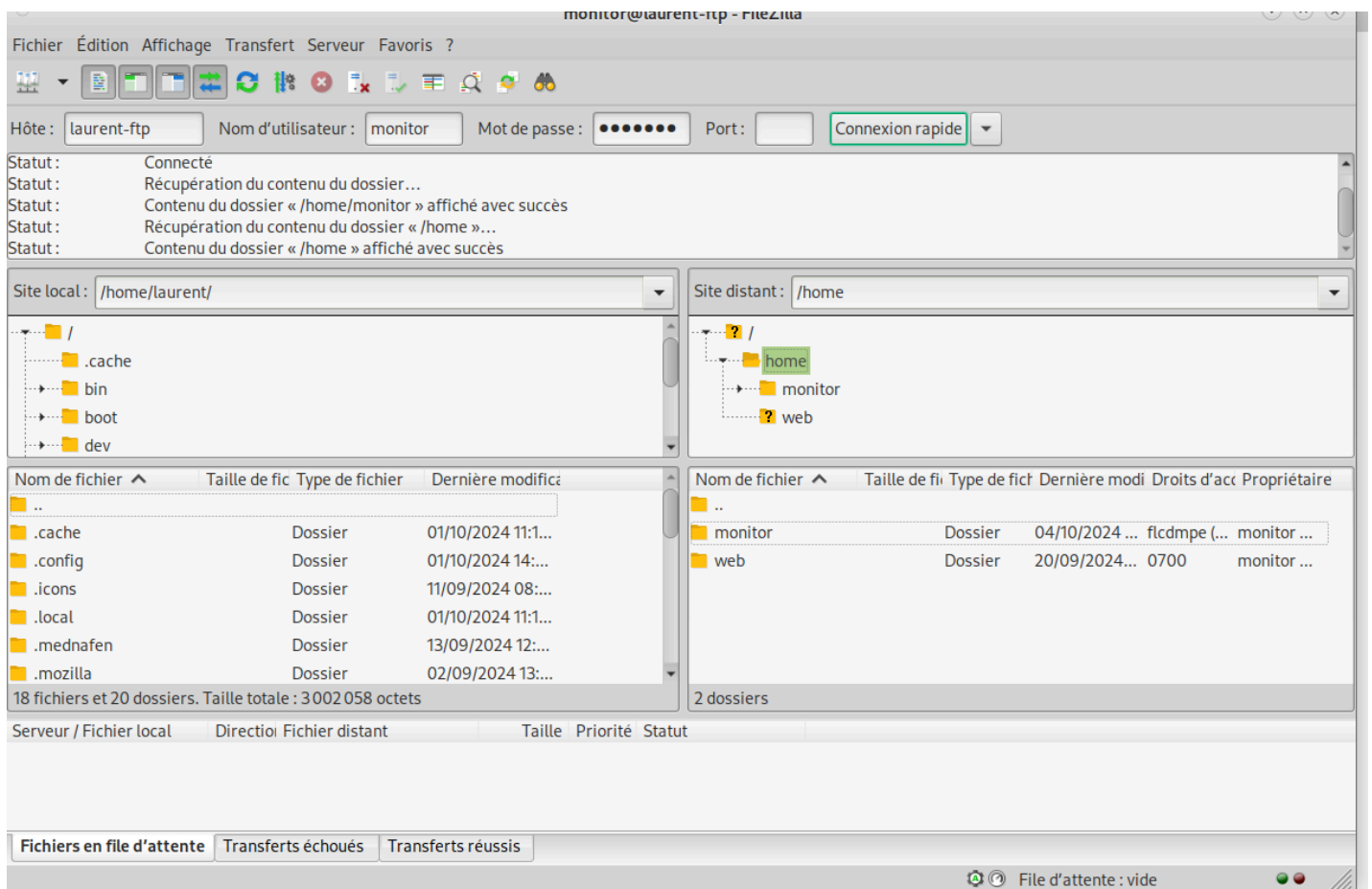
Installation de **FILEZILLA** sur PC perso:

**FileZilla** existe sous deux formes distinctes :

1. **FileZilla Client** : C'est le logiciel utilisé pour se connecter à un serveur FTP ou SFTP et transférer des fichiers. Il est principalement destiné aux utilisateurs qui veulent gérer et échanger des fichiers avec un serveur distant.
2. **FileZilla Server** : C'est un serveur FTP/SFTP qui permet à d'autres utilisateurs de se connecter à ton système pour échanger des fichiers. Cette version est utilisée pour héberger des fichiers et permettre à des clients FTP de se connecter à ton serveur.

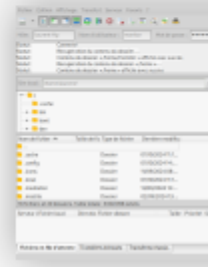
## Connexion par FILEZILLA:

connexion au serveur **laurent-ftp** avec compte **monitor** et mot de passe **monitor**



## connexion mode CLI :

```
laurent-client@laurent-client:~$ ftp laurent-ftp
Connected to laurent-ftp.home.
220 ProFTPD Server (Debian) [::ffff:192.168.1.79]
Name (laurent-ftp:laurent-client): monitor
331 Mot de passe requis pour monitor
Password:
230 Utilisateur monitor authentifié
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> █
```



## LOGS erreur ProFTPD :

- Les logs d'erreur de ProFTPD sont souvent configurés dans le fichier de configuration principal **/etc/proftpd/proftpd.conf**.

Pour vérifier ou modifier l'emplacement des logs, regarde les lignes suivantes dans le fichier de configuration (**/etc/proftpd/proftpd.conf**) :

- Voici les emplacements par défaut :

```
TransferLog /var/log/proftpd/xferlog
SystemLog /var/log/proftpd/proftpd.log
```

## lecture des logs dans la vm ftp

```
monitor@laurent-ftp:/var/log$ sudo cat /var/log/proftpd/proftpd.log
[sudo] Mot de passe de monitor :
2024-10-04 08:47:38,296 laurent-ftp proftpd[781] laurent-ftp: ProFTPD 1.3.8 (stable) (built Tue Jan 9 2024 21:52:35 UTC) standalone mode STARTUP
2024-10-04 09:49:36,013 laurent-ftp proftpd[781] laurent-ftp: ProFTPD killed (signal 15)
2024-10-04 09:49:36,013 laurent-ftp proftpd[781] laurent-ftp: ProFTPD 1.3.8 standalone mode SHUTDOWN
2024-10-04 09:50:16,343 laurent-ftp proftpd[719] laurent-ftp: ProFTPD 1.3.8 (stable) (built Tue Jan 9 2024 21:52:35 UTC) standalone mode STARTUP
2024-10-04 10:00:48,790 laurent-ftp proftpd[719] laurent-ftp: ProFTPD killed (signal 15)
2024-10-04 10:00:48,791 laurent-ftp proftpd[719] laurent-ftp: ProFTPD 1.3.8 standalone mode SHUTDOWN
2024-10-04 10:01:32,639 laurent-ftp proftpd[705] laurent-ftp: ProFTPD 1.3.8 (stable) (built Tue Jan 9 2024 21:52:35 UTC) standalone mode STARTUP
2024-10-04 12:27:05,994 laurent-ftp proftpd[705] laurent-ftp: ProFTPD killed (signal 15)
2024-10-04 12:27:05,995 laurent-ftp proftpd[705] laurent-ftp: ProFTPD 1.3.8 standalone mode SHUTDOWN
2024-10-05 16:03:17,504 laurent-ftp proftpd[707] laurent-ftp: ProFTPD 1.3.8 (stable) (built Tue Jan 9 2024 21:52:35 UTC) standalone mode STARTUP
2024-10-05 16:24:07,333 laurent-ftp proftpd[806] laurent-ftp (debian-1.home[192.168.1.73]): FTP no transfer timeout, disconnected
2024-10-05 16:26:47,077 laurent-ftp proftpd[807] laurent-ftp (debian-1.home[192.168.1.77]): FTP no transfer timeout, disconnected
```

Écrire le script **“ssh\_ftp\_error.py”**, pour qu’il récupère les logs d’erreur et les écrire dans votre base de données.

**[script pure pour utiliser en copie ]**

**le script se situe vers =>cd /home/laurent-client**

**copier les caractères qui sont en vert !!!!**

**GNU nano 7.2**

**ssh\_ftp\_error.py**

```
import ftplib
```

```
import pymysql
```

```
import re
```

```
from getpass import getpass
```

```
def connect_ftp_and_log_attempts(ftp_host, ftp_username, ftp_password, sql_host,  
sql_username, sql_password):
```

```
    try:
```

```
        # Connexion au serveur FTP
```

```
        print(f"Connexion au serveur FTP {ftp_host} en tant que {ftp_username}...")
```

```
        ftp = ftplib.FTP(ftp_host)
```

```
        ftp.login(ftp_username, ftp_password)
```

```
        print("Connexion FTP réussie.")
```

```
        # Lire le fichier de logs du serveur FTP (sous ProFTPD, par exemple)
```

```
        log_file_path = "/var/log/proftpd/proftpd.log" # Le chemin peut varier selon la  
configuration
```

```
        # Essayons de télécharger le fichier de logs
```

```
        with open("ftp_error_log.txt", "wb") as f:
```

```
            ftp.retrbinary(f"RETR {log_file_path}", f.write)
```

```
            print("Fichier de logs téléchargé avec succès.")
```

```
        # Lire le contenu du fichier de logs
```

```
        with open("ftp_error_log.txt", "r") as f:
```

```
            log_data = f.read()
```

## **# Regex pour extraire les tentatives de connexion échouées**

```
pattern = r"\[(.*?)\] .*no such user '(.*?)'"
```

## **# Connexion à la base de données MariaDB pour stocker les logs**

```
connection = pymysql.connect(  
    host=sql_host, # La VM laurent-sql.local  
    user=sql_username,  
    password=sql_password  
)
```

try:

with connection.cursor() as cursor:

### **# Création de la base de données si elle n'existe pas**

```
create_database_query = "CREATE DATABASE IF NOT EXISTS ftp_logs_db"  
cursor.execute(create_database_query)
```

### **# Sélection de la base de données**

```
cursor.execute("USE ftp_logs_db")
```

### **# Création de la table si elle n'existe pas**

```
create_table_query = """  
  
CREATE TABLE IF NOT EXISTS ftp_login_attempts (  
  
    id INT AUTO_INCREMENT PRIMARY KEY,  
  
    username VARCHAR(50),  
  
    attempt_time DATETIME  
  
)
```

```
"""
```

```
cursor.execute(create_table_query)
```

### **# Extraire les informations des tentatives de connexion échouées**

```
for match in re.finditer(pattern, log_data):
```

```
    attempt_time, username = match.groups()
```

### **# Insérer les informations dans la table ftp\_login\_attempts**

```
insert_query = """
```

```
INSERT INTO ftp_login_attempts (username, attempt_time)
```

```
VALUES (%s, %s)
```

```
"""
```

```
cursor.execute(insert_query, (username, attempt_time))
```

### **# Commit les changements dans la base de données**

```
connection.commit()
```

### **# Afficher le contenu de la table ftp\_login\_attempts**

```
cursor.execute("SELECT * FROM ftp_login_attempts")
```

```
results = cursor.fetchall()
```

```
print("\nTentatives de connexion FTP enregistrées :")
```

```
for row in results:
```

```
    print(f"ID: {row[0]}, Username: {row[1]}, Attempt Time: {row[2]}")
```

```
finally:
```

```
connection.close() # Fermer la connexion à la base de données
```

## # Déconnexion du serveur FTP

```
ftp.quit()

except Exception as e:

    print(f"Erreur : {e}")

if __name__ == "__main__":

    ftp_host = "laurent-ftp.local" # Remplacer par le bon hostname ou IP

    ftp_user = "monitor"

    ftp_password = getpass("Entrez le mot de passe FTP : ")

    sql_host = "laurent-sql.local" # Adresse de la VM SQL

    sql_username = "monitor" # Assurez-vous que cet utilisateur existe dans la base de
données SQL

    sql_password = getpass("Entrez le mot de passe SQL : ")

    connect_ftp_and_log_attempts(ftp_host, ftp_user, ftp_password, sql_host,
sql_username, sql_password)
```

lors de l'exécution du script si ya erreur de violation de fichier faut changer les droits d'accès au fichier des logs:

```
laurent-client@laurent-client:~$ python3 ssh_ftp_error.py
Entrez le mot de passe FTP :
Entrez le mot de passe SQL :
Connexion au serveur FTP laurent-ftp.local en tant que monitor...
Connexion FTP réussie.
Erreur : 550 /var/log/proftpd/proftpd.log: Permission non accordée
```

modifiez les droits



```
monitor@laurent-ftp:/var/log$ su -
Mot de passe :
root@laurent-ftp:~# sudo chmod 644 /var/log/proftpd/proftpd.log
```

**6** : Pour le propriétaire

**4** : Pour le groupe

**4** : Pour les autres utilisateurs

Chiffre	Permission	Description	Notation Symbolique
0	---	Aucun droit	---
1	--x	Exécution uniquement	--x
2	-w-	Écriture uniquement	-w-
3	-wx	Écriture et exécution	-wx
4	r--	Lecture uniquement	r--
5	r-x	Lecture et exécution	r-x
6	rw-	Lecture et écriture	rw-
7	rwX	Lecture, écriture et exécution	rwX

```
laurent-client@laurent-client:~$ python3 ssh_ftp_error.py
Entrez le mot de passe FTP :
Entrez le mot de passe SQL :
Connexion au serveur FTP laurent-ftp.local en tant que monitor...
Connexion FTP réussie.
Fichier de logs téléchargé avec succès.

Tentatives de connexion FTP enregistrées :
```

la base de données **ftp\_logs\_db** et bien créé dans mariadb sur le serveur sql:

```
MariaDB [(none)]> show database;
+-----+
| Database |
+-----+
| ftp_logs_db |
| information_schema |
| lo |
| logs_db |
| mysql |
| performance_schema |
| sys |
| testdb |
| testlo |
+-----+
9 rows in set (0,007 sec)

MariaDB [(none)]>
```

```

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| ftp_logs_db |
| information_schema |
| lo |
| logs_db |
| mysql |
| performance_schema |
| sys |
| testdb |
| testlo |
+-----+
9 rows in set (0,001 sec)

MariaDB [(none)]> use ftp_logs_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [ftp_logs_db]> SELECT * FROM ftp_logs_db;
ERROR 1146 (42S02): Table 'ftp_logs_db.ftp_logs_db' doesn't exist
MariaDB [ftp_logs_db]> DESCRIBE ftp_log_db;
ERROR 1146 (42S02): Table 'ftp_logs_db.ftp_log_db' doesn't exist
MariaDB [ftp_logs_db]> show tables;
+-----+
| Tables_in_ftp_logs_db |
+-----+
| ftp_login_attempts |
+-----+
1 row in set (0,000 sec)

MariaDB [ftp_logs_db]> SELECT * FROM ftp_login_attempts;
+----+-----+-----+-----+
| id | username | attempt_time | ip_address |
+----+-----+-----+-----+
| 1 | test_user | 2024-10-05 17:26:49 | 192.168.1.10 |
+----+-----+-----+-----+
1 row in set (0,000 sec)

```

---

## Job 08:

Écrire le script **“ssh\_web\_error.py”**, pour récupérer les logs d’erreurs et les écrire dans votre base de données.

