Documentation : notike-gestion.ltd

Ce guide explique comment installer MySQL sur un serveur web, créer des utilisateurs et importer une base de données, ainsi que comment configurer les restrictions d'accès avec iptables et mettre en place une sauvegarde automatique avec crontab. Il explique également comment automatiser l'initialisation et le déploiement du serveur web au démarrage avec un script init.d.

Installation Apache2

Cette section sera très brève puisque facile à mettre en application, notre serveur ne pas tourner sous nginx mais sous Apache2 il nous faudra donc au préalable l'installer sur notre debian qui servira de serveur web.

Voici les commandes à effectuer pour l'installer :

```
apt update && apt upgrade
apt install apache2
apt install php # si vous voulez faire supporter du php
```

Mise en place virtualhost

L'idée derrière cette implémentation est de rajouter une couche de protection supplémentaire à Apache en utilisant un virtualhost avec une restriction d'accès basée sur un nom de domaine local connu uniquement par les élèves et les

professeurs, il ne sera pas possible d'y accéder par l'ip du serveur web , pour activer la fonctionnalité permettant le virtualhost sur Apache2 il faut faire la commande suivante en superutilisateur:

```
a2enmod authz_host
```

Une fois cela fait, veuillez vous placer dans le répertoire /var/www/html/ et créez un répertoire "site" (dans notre cas) ce répertoire contiendra le projet django plus tard.

```
cd /var/www/html/; mkdir site
```

Une fois cela est fait nous pouvons commencer la mise en place du virtualhost.

Création du fichier configuration virtualhost

Dans un premier temps rendez vous à l'emplacement où se trouve les fichiers de configuration de Apache2 et puis créez un nouveau fichier configuration du nom de virtualhost.conf (en superutilisateur):

```
cd /etc/apache2/sites-available/; nano virtualhost.conf
```

Puis mettez-y les lignes de code suivantes:

```
<VirtualHost *:80>
    ServerName notike-gestion.ltd
    DocumentRoot /var/www/html/site
    #Redirect permanent / https://notike-gestion.ltd/
    <Directory /var/www/html/site>
        Require expr %{HTTP_HOST} == "notike-gestion.ltd"
    </Directory>
</VirtualHost>
```

Ce fichier configuration permet d'indiquer à Apache2 que si on essaie de communiquer avec le nom de domaine notike-gestion.ltd et que ça pointe vers le serveur Apache2 il doit afficher la ressource web contenu dans /var/www/html/site. J'ai rajouté une configuration supplémentaire qui souligne bien le fait que je veux que le nom de domaine soit écrit sans www ou préfixe devant.

Pour activer cette configuration il suffit d'éxécuter la commande suivante en superutilisateur et de reload le service Apache2 :

```
a2ensite virtualhost.conf
systemctl reload Apache2
```

Pensez-bien à rajouter dans le fichier /etc/hosts le nom de domaine local choisit pour qu'il cible le serveur

Création du fichier configuration redirection

Nous sommes qu'à mi-chemin de l'implémentation de la couche de protection, puisque malgré l'activation de cette règle, il est toujours possible d'accéder à la ressource "site" en entrant l'url suivante: http://{IP_site}/site/ , et ce comportement on souhaite

l'éviter puisque ça voudrait dire que toute personne possèdant l'ip du serveur web pourrait accéder à la ressource, et ce qu'on voulait à la base c'est que la ressource soit accessible uniquement depuis un nom de domaine local.

Pour ajouter cette fonctionnalité il faut créer un nouveau fichier de configuration au nom redirect.conf dans lequel se trouvera les lignes suivantes :

```
<VirtualHost *:80>
    ServerName 192.168.43.33

    RedirectMatch ^/$ https://www.facebook.com/

    <Location /site/>
        RewriteEngine On
        RewriteRule ^(.*)$ https://www.facebook.com/ [R=301,L]
    </Location>
</VirtualHost>
```

L'adresse IP dans servername correspond à l'adresse IP (de préférence statique) de votre serveur, mais dans le cas où votre réseau fonctionne en DHCP vous pourrez ultérieurement faire en sorte que l'ip soit automatiquement introduite à cette emplacement.

Et ce fichier configuration indique que si on essaie de communiquer avec le serveur web via l'ip directement : on se fait rediriger vers facebook avec un code redirection 301 . Plutôt radicale comme solution.

Puisqu'on utilise rewrite il faut penser à l'activer en superutilisateur avec la commande : a2enmod rewrite

Pour appliquer le fichier de configuration exécutez les commandes suivantes (en superutilisateur):

```
a2ensite redirect.conf
systemctl reload Apache2
```

💡 **Bien évidemment il était possible de regrouper les deux fichiers configurations en 1 seul, mais pour des raisons de compréhensions on a préféré les scinder selon leurs fonctions.**

Dernière Etape pour le virtualhost

Pensez-bien à mettre le fichiers hosts en faisant pointer le nom de domaine vers l'adresse IP du serveur, si vous ne le faites pas votre navigateur essaiera de résoudre le nom de domaine publiquement.

Voici un rappel de la syntaxe à utiliser dans le fichiers hosts :

127.0.0.1 monsite.com

Mise en place certicat TLS

Dans notre exemple on veut déployer un certficat TLS sur le site web local de notre application ayant comme nom de domaine : **notike-gestion.ltd** . Dans un premier temps avant mettre en place le certificat il faut activer l'extension ssl de apache2 avec la commande suivante (en superutilisateur):

```
a2enmod ssl
```

Une fois cela fait, veuillez créer un répertoire dans /var/www/ du nom **cert** par exemple afin de stocker les certificats dans un endroit accessible (l'emplacement de stockage peut être différent) :

```
cd /var/www/; mkdir cert; cd cert
```

Puis installez le paquet openssl qui nous servira pour créer le certificat.

Création du certificat TLS

1. Generate RSA

```
openssl genrsa -aes256 -out ca-key.pem 4096
```

Cette commande génère une clé RSA de 4096 bits avec un chiffrement AES 256 bits et la sauvegarde dans un fichier nommé "ca-key.pem". Il vous sera demandé de choisir une passphrase, pensez bien à la garder dans un fichier que vous nommerez passphrase.txt .

2. Generate a public CA Cert

```
openssl req -new -x509 -sha256 -days 365 -key ca-key.pem -out ca.pem
```

Cette commande crée un certificat d'autorité de certification (CA) public auto-signé au format X.509, valable pendant 365 jours, en utilisant la clé privée "ca-key.pem" précédemment généré. Le certificat est sauvegardé dans un fichier nommé "ca.pem".

Il vous sera ensuite demandé de remplir des informations relatives au certificat voici ce qu'on a mis dans notre cas :

```
root@debian-test:/var/www/cert# openssl req -new -x509 -sha256 -days 365 -key ca-key.pem -out ca.pem
Enter pass phrase for ca-key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Grand-Est
Locality Name (eg, city) []:Colmar
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Notike LTD
Organizational Unit Name (eg, section) []:RT
Common Name (e.g. server FQDN or YOUR name) []:admin
Email Address []:admin@localhost.com
root@debian-test:/var/www/cert#
```

3. Create a RSA key

```
openssl genrsa -out cert-key.pem 4096
```

Cette commande génère une clé RSA de 4096 bits et la sauvegarde dans un fichier nommé "cert-key.pem".

4. Create a Certificate Signing Request (CSR)

```
openssl req -new -sha256 -subj "/CN=notike-gestion" -key cert-key.pem -out cert.csr
```

Cette commande crée une demande de signature de certificat (CSR) avec le sujet "/CN=notike-gestion" et la clé privée "cert-key.pem". Le CSR est sauvegardé dans un fichier nommé "cert.csr".

5. Create a conf file

```
echo "subjectAltName=DNS:notike-gestion.ltd,IP:{IP}" >> extfile.cnf
```

Cette commande ajoute la ligne "subjectAltName=DNS:notike-gestion.ltd,IP:{IP}" au fichier "extfile.cnf". Cela spécifie les noms alternatifs du sujet dans le certificat. Il faut pas oublier de remplacer le champs IP par l'ip du serveur web.

6. Create the certificate

```
openssl x509 -req -sha256 -days 365 -in cert.csr -CA ca.pem -CAkey ca-key.pem -out cert.pem -extfile extfile.cnf -CAcreateserial
```

Cette commande crée le certificat en utilisant le CSR "cert.csr", la clé privée de l'autorité de certification "ca-key.pem" et le certificat de l'autorité de certification "ca.pem". Le certificat résultant est sauvegardé dans un fichier nommé "cert.pem" et les noms alternatifs du sujet sont spécifiés à partir du fichier "extfile.cnf". De plus, un fichier de série est créé pour suivre les numéros de série des certificats signés. Il sera demandé de spécifier le passphrase.

7. Fullchain

```
cat cert.pem > fullchain.pem; cat ca.pem >> fullchain.pem
```

Cette commande utilise la commande `cat` pour concaténer les contenus des fichiers `cert.pem` et `ca.pem` et les redirige vers un nouveau fichier nommé `fullchain.pem`.

- `cat cert.pem > fullchain.pem` copie le contenu du fichier `cert.pem` dans le fichier `fullchain.pem`. Le symbole `>` redirige la sortie vers un fichier, écrasant le contenu précédent ou créant un nouveau fichier s'il n'existe pas.
- `cat ca.pem >> fullchain.pem` concatène le contenu du fichier `ca.pem` à la fin du fichier `fullchain.pem`. Le symbole `>>` est utilisé pour ajouter la sortie à la fin d'un fichier existant sans écraser le contenu existant.

En exécutant ces deux commandes consécutivement, le fichier `fullchain.pem` sera créé avec le contenu combiné des certificats du serveur (`cert.pem`) et de l'autorité de certification (`ca.pem`). Cela permet d'obtenir un fichier contenant la chaîne complète de certificats pour une configuration appropriée et une vérification de la chaîne de confiance lors de l'établissement d'une connexion sécurisée.

Implémentation Apache2

Dans notre cas nous n'avons pas réellement de nom de domaine qui a été enregistré sur un service tel que namecheap, cependant en local nous pouvons en simuler un avec les fichiers hosts des systèmes d'opérations, en plus de cela Apache2 permet d'établir un servername qu'on peut appeler comme un nom de domaine. Il ne faut pas oublier d'activer depuis le compte superutilisateur l'extension Apache2 **authz_host (a2enmod authz_host)** permettant le déploiement des virtualhosts**.**

Dans un premier temps créez un fichier conf dans `/etc/apache2/sites-available/` portant le nom que vous souhaitez (doit se finir en .conf). Ce fichier contiendra à la fois la config relative au protocole http hebergé sur le port 80 et celle du protocole https hebergé sur le port 443.

Voici une configuration possible pour notre site :

```
<VirtualHost *:80>
    ServerName notike-gestion.ltd
    DocumentRoot /var/www/html/site
    Redirect permanent / https://notike-gestion.ltd/
    <Directory /var/www/html/site>
        Require expr %{HTTP_HOST} == "notike-gestion.ltd"
        # Autres configurations du dossier si nécessaire
    </Directory>
</VirtualHost>


<VirtualHost *:443>
    ServerName notike-gestion.ltd
    Protocols h2 http/1.1
    DocumentRoot /var/www/html/site
    RewriteEngine On
    <If "%{HTTP_HOST} == 'www.notike-gestion.ltd'">
    Redirect permanent / https://notike-gestion.ltd/
    </If>
    SSLEngine on
    SSLCertificateFile "/var/www/cert/fullchain.pem"
    SSLCertificateKeyFile "/var/www/cert/cert-key.pem"
    <Directory /var/www/html/site>
        Require expr %{HTTP_HOST} == "notike-gestion.ltd"
        # Autres configurations du dossier si nécessaire
    </Directory>
</VirtualHost>
```

Il faut juste remplacer /var/www/html/site par l'emplacement où se trouve votre site, en addition l'expression "require expr …" a été rajoutée puisqu'on voulait que le site puisse être atteint uniquement via son nom de domaine et non son IP.

une fois le fichier de configuration conf créé faites la commande suivante en superutilisateur pour charger la configuration apache et relancer le serveur :

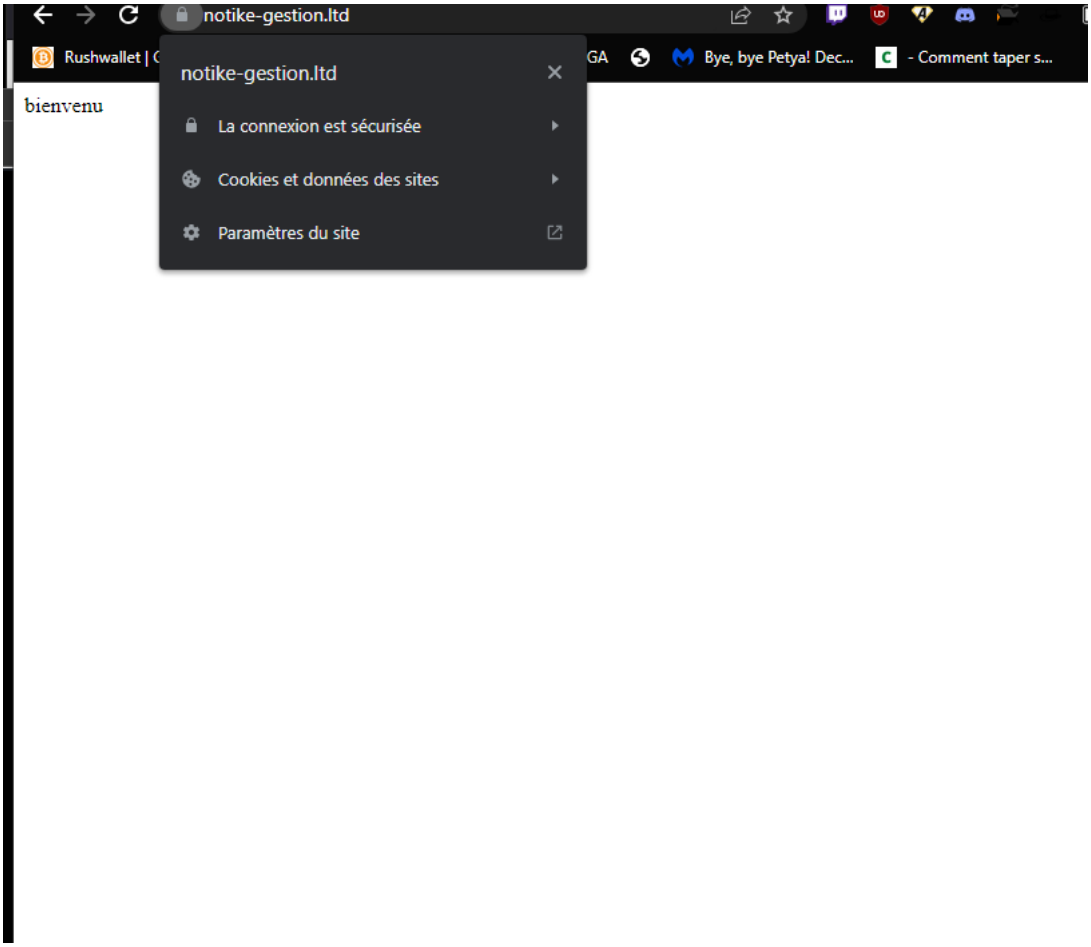
```
a2ensite {filename}.conf; systemctl restart apache2
```

Ajout certificat chez hôte

Nous en avons presque fini, cependant le fait que nous utilisons un nom de domaine local et non public, afin d'accéder à notre service web depuis le réseau local en https il faut installer le certificat, récupérer le fichier ca.pem et installez le sur l'hôte en tant que certificat reconnu et autorisé (en cliquant dessus sur windows).

En addition il faut ajouter le nom de domaine et l'ip du serveur dans le fichier hostname de l'hôte , se trouvant dans /etc/hosts pour linux et pour windows dans C:\Windows\System32\drivers\etc\.

Une fois que c'est fait vous pouvez enfin accéder à votre service préféré en tapant dans la barre de navigation l'url de site web :



Si vous préférez utiliser un script pour générer les certificats j'en ai un à disposition:

il faut juste penser à créer le fichier conf.txt

```
IP=$(ip addr show ens33 | grep -oP '(?<=inet\s)\d+(\.\d+){3}')

passphrase="24eefgegege9egegédde"
config_file="conf.txt"

country=$(sed -n '1p' "$config_file")
state=$(sed -n '2p' "$config_file")
locality=$(sed -n '3p' "$config_file")
organization=$(sed -n '4p' "$config_file")
organizational_unit=$(sed -n '5p' "$config_file")
common_name=$(sed -n '6p' "$config_file")
email_address=$(sed -n '7p' "$config_file")



openssl genrsa -aes256 -passout pass:"$passphrase" -out ca-key.pem 4096
echo "$passphrase" | openssl req -new -x509 -sha256 -days 365 -passin stdin -key ca-key.pem -out ca.pem \
-subj "/C=$country/ST=$state/L=$locality/O=$organization/OU=$organizational_unit/CN=$common_name/emailAddress=$email_address"
openssl genrsa -out cert-key.pem 4096
openssl req -new -sha256 -subj "/CN=notike-gestion" -key cert-key.pem -out cert.csr
echo "subjectAltName=DNS:notike-gestion.ltd,IP:$IP" >> extfile.cnf
openssl x509 -req -sha256 -passin pass:"$passphrase" -days 365 -in cert.csr -CA ca.pem -CAkey ca-key.pem -out cert.pem -extfile extfil
cat cert.pem > fullchain.pem; cat ca.pem >> fullchain.pem
```

https://github.com/ChristianLempa/cheat-sheets/blob/main/misc/ssl-certs.md

https://youtu.be/VH4gXcvkmOY

https://youtu.be/f9ZadlfSIDI

https://doc.owncloud.com/server/next/admin_manual/installation/letsencrypt/apache.html

Notike

Liaison Gunicorn à Django

Dans notre cas puisque l'application web devrait être utilisée hors production on ne va pas simplement relier Django et Apache2 sauvagement mais on va établir une passerelle entre les deux nommée Gunicorn. Gunicorn est un **serveur web HTTP WSGI** proposant du multiprocessing pour les requêtes entre le backend et le frontend. (Django et Apache2).

Pour des raisons de simplicité on va d'abord essayer de lier Gunicorn à Django mais avant toute chose pour éviter des complications futures avec le TLS on va modifier le script wsgi.py se trouvant dans les fichiers configurations du projet Django :

```
#application = get_wsgi_application()
django_app = get_wsgi_application()

def https_app(environ, start_response):
    environ["wsgi.url_scheme"] = "https"
    return django_app(environ, start_response)


application = https_app
```

Ces lignes sont utiles pour forcer le traffic via le https

Ensuite installer la dépendance python Gunicorn:

```
pip install gunicorn
```

Une fois que les choses ci-dessous sont effecutées on peut commencer à lier Gunicorn et Django.

Création de(s) script(s) configuration Gunicorn

Pour ne pas avoir à réécrire les mêmes commandes pour lier Gunicorn et Django, on va rédiger un petit script configuration python nommé **gunicorn_config.py** se trouvant à la racine du projet (là où se trouve manage.py), celui-ci permettra d'établir cette liaison lors de son éxécution :

```
from multiprocessing import cpu_count

bind = "127.0.0.1:8000"  # Adresse IP et port sur lesquels Gunicorn écoute les connexions
workers = cpu_count() * 2 + 1  # Nombre de processus de travail Gunicorn
```

Le script est assez clair , le bind est là où il écoute les connexions (par défaut django émet sur le port 8000) . Et workers correspond à la division des processus de travail Gunicorn.

Une fois cela effectué veuillez créer au même emplacement un script bash au nom de launch.sh contenant les lignes suivantes:

```
cd /var/www/html/site/
gunicorn gestionnaire_notes.wsgi:application --config gunicorn_config.py
```

La première ligne a été insérée dans un cas où on veut automatiser l'éxécution du script, sa présence est recommandée

Ce script exécute gunicorn sur le projet gestionnaire_notes en spécifiant la configuration gunicorn_config.py précédemment rédigée.

Une fois ce script rédigé veuillez l'éxécuter et accéder à votre site web pour vérifier que tout est fonctionnel.

Liaison Gunicorn à Apache2

Petite manipulation pour lier Gunicorn à Apache2

La liaision entre Gunicorn et le backend Django a été effectuée, mais maintenant il s'agirait de relier le frontend Apache2 avec cette même passerelle.

Avant de poursuivre veuillez activer les deux modules Apache suivant en superutilisateur :

```
a2enmod proxy
a2enmod proxy_http
```

Ces deux modules serviront lors de l'édition des fichiers configurations Apache2.

Une fois cela fait on peut lier Gunicorn et Apache2.

Liaison Gunicorn à Apache2 via fichier configuration.

Rendez vous à l'emplacement /etc/apache2/sites-available et éditez le fichier configuration virtualhost.conf en ajoutant ces lignes à la fin :

```
# Configurations pour Gunicorn
    ProxyPass / http://localhost:8000/
    ProxyPassReverse / http://localhost:8000/
    ProxyPreserveHost On
```

Enregistrez le fichier et c'est bon pas besoin de reload puisque le changement se fait en temps réel, vous pouvez tout de même le faire si vous le souhaitez.

- `ProxyPass / http://localhost:8000/` : Cette directive spécifie que toutes les requêtes qui correspondent à la racine du serveur (le chemin "/") doivent être redirigées vers l'adresse `http://localhost:8000/`. Cela signifie que toutes les requêtes HTTP seront envoyées à Gunicorn pour être traitées par votre application Django.
- `ProxyPassReverse / http://localhost:8000/` : Cette directive indique à Apache2 de modifier les en-têtes de réponse reçus de Gunicorn lors de la redirection inverse. Cela permet à Apache2 de renvoyer correctement les en-têtes de réponse à l'origine de la requête, en prenant en compte la redirection.
- `ProxyPreserveHost On` : Cette directive indique à Apache2 de préserver l'en-tête `Host` d'origine de la requête lors de la redirection vers Gunicorn. Cela garantit que Gunicorn reçoit la demande avec l'en-tête `Host` correct, ce qui est important

pour le routage approprié de l'application Django si vous utilisez des noms d'hôte virtuels (VirtualHosts) ou des configurations basées sur le nom d'hôte.

Voici le diagramme d’une requête du client vers le serveur web:

```
+--------------+
|   Client     |
+------+-------+
       |    1. Requête HTTP
       v
+--------------+
|   Apache2    |
+------+-------+
       |    2. ProxyPass
       v
+--------------+
|   Gunicorn   |
+------+-------+
       |    3. Transmission de la requête
       v
+--------------+
|   Django     |
+--------------+
       |    4. Traitement de la requête
       v
+--------------+
|   Django     |
|   (suite)    |
+--------------+
       |    5. Génération de la réponse
       v
+--------------+
|   Gunicorn   |
+------+-------+
       |    6. Renvoi de la réponse
       v
+--------------+
|   Apache2    |
+------+-------+
       |    7. Renvoi de la réponse
       v
+--------------+
|   Client     |
+--------------+
```

Installation mysql / importation BDD

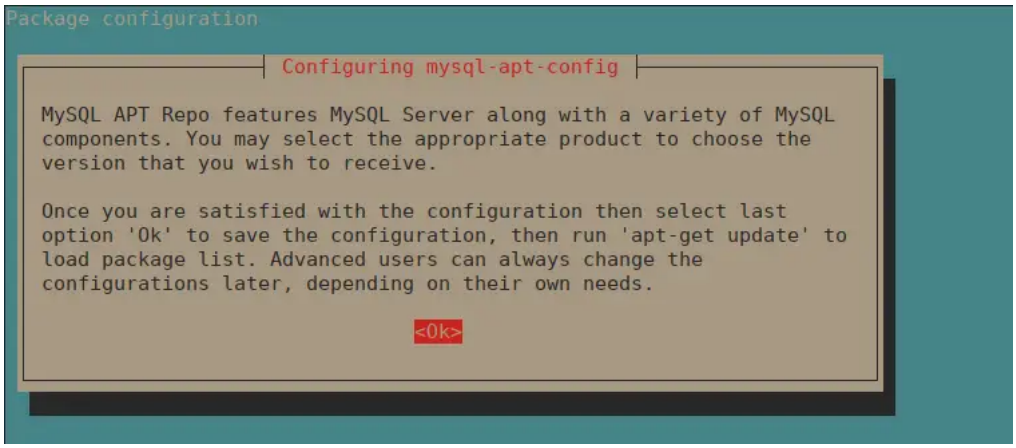
Notre projet web ne va évidemment pas remplir la base de donnée dans un fichier base de donnée se trouvant au répertoire racine du projet , il va le remplir via une connexion vers mysql avec l’utilisateur archiviste (dans notre cas root pour le développement).

Seulement par défaut, Debian l’OS de notre serveur n’a pas mysql d’installé et un simple apt install n’est pas suffisant pour l’installer.

Installation de mysql dev apt repository sur Debian

Pour installer le mysql dev apt repo sur debian il faut récupérer le fichier deb se trouvant dans le repo de mysql, entrez ces commandes pour initialiser l’installation :

```
wget https://repo.mysql.com/mysql-apt-config_0.8.25-1_all.deb
sudo dpkg -i mysql-apt-config_0.8.25-1_all.deb
```



Ne choisissez uniquement les options par défaut et faites entrer.

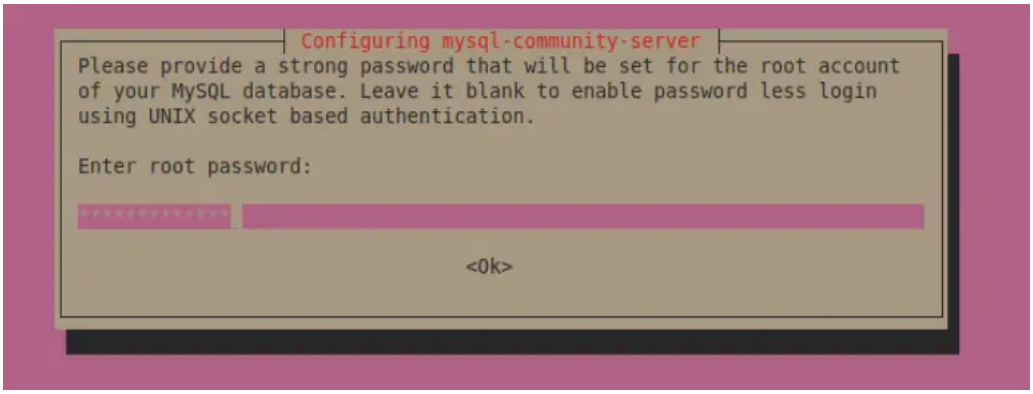
Vous pouvez désormais installer mysql-server

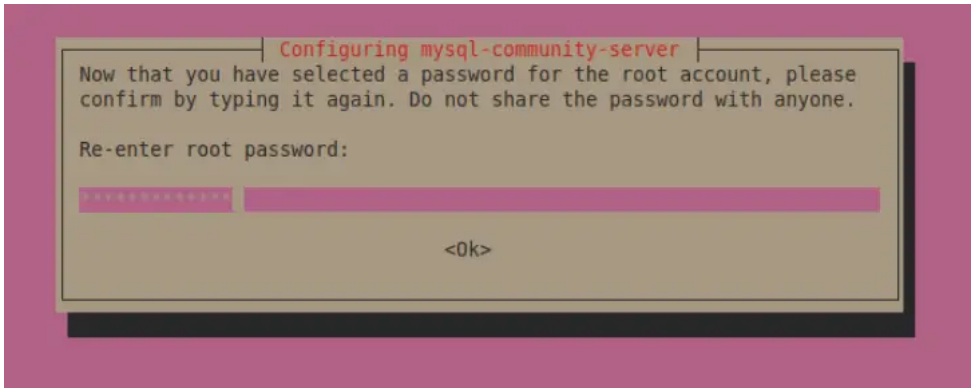
Installation de mysql server

Effectuez les commandes ci-dessous pour commencer l'installation:

```
sudo apt update
sudo apt install mysql-server
```

Créez votre mot de passe root de votre base de données (à garder précieusement)





Dans notre cas sélectionner Use Strong Password Encryption

Une fois cela fini vous avez officiellement installé mysql sur votre serveur web !

Création utilisateurs et importation de la base de données

Veuillez entrer dans le shell mysql avec la commande suivante :

```
mysql -u root -ppassword
```

Veuillez copier/coller ces commandes de création utilisateur, ce sont des utilisateurs utile au bon fonctionnement du site web avec scribe qui peut ajouter des inscriptions dans la base de donnée et archiviste qui peut dumper la base de donnée.

```
CREATE USER 'scribe'@'localhost' IDENTIFIED BY 'scribepasswd';
GRANT SELECT, INSERT, UPDATE, DELETE ON sae_23.* TO 'scribe'@'localhost';
CREATE USER 'archivist'@'localhost' IDENTIFIED BY 'arcpasswd';
GRANT SELECT, SHOW VIEW, EVENT, TRIGGER ON sae_23.* TO 'archivist'@'localhost';
FLUSH PRIVILEGES;
```

Ensuite pour l'importation de la base de donnée vous pouvez repasser en vue shell bash et effectuer la commande suivante :

```
mysql -u root -pmotdepasse sae_23 < backup_sae_23.sql
```

Une fois cela fait vous avez bien importé la base de donnée mysql.

iptables - restrictions d'accès

Lors du déploiement de notre solution web on souhaite dans l'idéal que les clients pouvant intéragir avec celui-ci ne communiquent uniquement avec Apache et les services relatifs au web, on veut éviter que les services de gestions tel que mysql peuvent être altérés.

Dans l'idéal on veut que nos configurations iptables soient persistantes et appliquées à chaque démarrage, il faudra donc faire ses étapes avant de commencer les règles d'accès:

```
sudo apt-get install iptables-persistent # pour la persistance
sudo mkdir /etc/iptables/
touch /etc/iptables/rules.v4 # contiendra les règles adresse IPv4
```

Une fois cela fait on peut commencer notre configuration

Configuration de iptables

Pour supprimer les règles actuelles de iptables veuillez entrer les commandes suivantes (en superutilisateur):

```
sudo iptables -F
sudo iptables -X
```

Ensuite pour éviter quelques conflits avec les redirections , entrées / sorties veuillez rentrer les commandes suivantes :

```
sudo iptables -P INPUT ACCEPT
sudo iptables -P FORWARD ACCEPT
sudo iptables -P OUTPUT ACCEPT
```

Maintenant, veuillez déterminer les services que vous souhaitez que leur accès soit restreint, une bonne manière de lister les services réseaux sur votre machine est de faire un netmapping :

```
nmap localhost

Starting Nmap 7.80 ( https://nmap.org ) at 2023-06-15 12:40 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 994 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
80/tcp   open     http
443/tcp  open     https
631/tcp  open     ipp
3306/tcp open     mysql
8000/tcp open     http-alt

Nmap done: 1 IP address (1 host up) scanned in 1.28 seconds
```

Dans notre cas il y a deux services qu'on souhaite restreindre l'accès depuis l'extérieur : ipp (car souvent vulnérable) et mysql (car contient nos bases de données).

Pour mettre en place ces règles veuillez entrer les commandes ci-dessous (en superutilisateur):

```
sudo iptables -A INPUT -p tcp -m tcp --dport 631 -j DROP
sudo iptables -A INPUT -p tcp -m tcp --dport 3306 -j DROP
```

Ces règles sont appliquées en temps réel vous pouvez donc déjà relancer un netmapping et vérifier que les services sont bien filtrés:

```
Starting Nmap 7.80 ( https://nmap.org ) at 2023-06-15 12:48 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 994 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
80/tcp   open     http
443/tcp  open     https
631/tcp  filtered ipp
3306/tcp filtered mysql
8000/tcp open     http-alt

Nmap done: 1 IP address (1 host up) scanned in 1.28 seconds
```

Maintenant il faut sauvegarder la configuration actuelle et la rendre persistante veuillez entrer les commandes suivantes:

```
sudo iptables-save > /etc/iptables/rules.v4
sudo systemctl enable netfilter-persistent
sudo systemctl start netfilter-persistent
```

Rédémarrez et vérifiez que tout fonctionne autant du côté d'iptables (avec iptables -L ou un nmap) que du côté serveur web.

Sur une machine différente du même réseau vous pouvez tester l'efficacité de vos règles, un scénario positif serait le suivant:

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -psomepassword -h 192.168.43.33
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 2003 (HY000): Can't connect to MySQL server on '192.168.43.33:3306' (10060)
```

crontab : sauvegarde/chiffrement base de données

Lors d'un déploiement d'une solution informatique qu'importe son origine, des soucis au niveau de la base de données peuvent survenir, dans ces situations il est souvent rassurant et appréciable d'avoir un service de sauvegarde tournant en arrière-plan pour pouvoir restaurer l'état de notre base de données dans le cas où il y a corruption/infection ou autres problèmes.

Pour que ce service tourne en arrière-plan nous avons opté pour l'utilisation de cron présent sur les OS Linux, celui-ci comme son alternative taskscheduler.exe sur windows permet de planifier des tâches en arrière-plan. Donc on créera un cronjob qui s'éxécutera toutes les heures qu'on fera cibler vers un script bash, ce script bash exportera la base de donnée en single transaction et la chiffrera avec openssh (pour plus de sécurités). Le single transaction permet de pouvoir exporter la BDD tout en préservant l'intéraction des clients avec la base de données.

Assurez-vous d'avoir openssl d'installé sur votre serveur, si c'est le cas nous pouvons commencer la planification de notre tâche de sauvegarde.

Nos backups vont s'effectuer dans un dossier spécial se trouvant dans /var/www/ veuillez le créer:

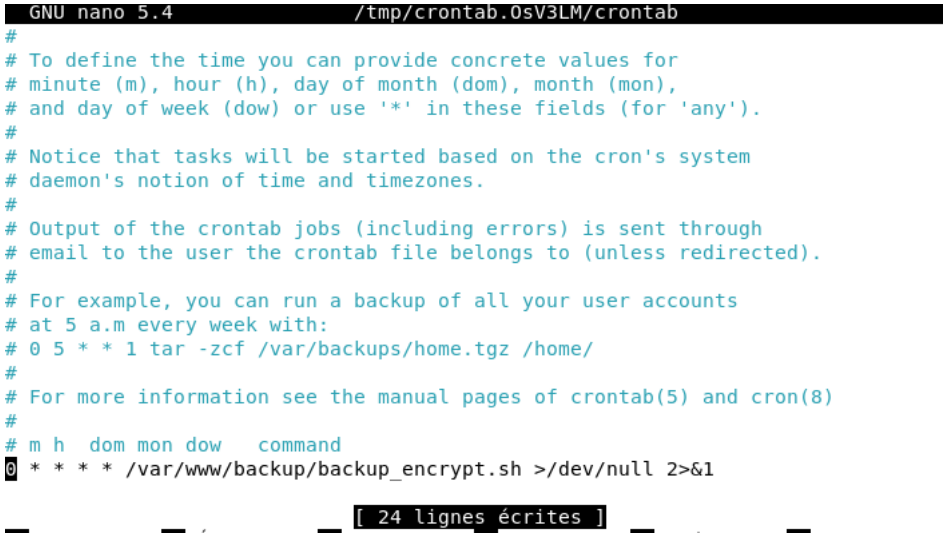
```
sudo mkdir /var/www/backup/
```

Planification de la sauvegarde

Si nous éxécutons la commande crontab -l avec notre utilisateur on peut constater qu'il n'y a aucune tâche créée avec notre utilisateur pour le moment, pour en créer une nouvelle veuillez entrer la commande suivante:

```
sudo crontab -e
(puis choisissez nano c'est beaucoup plus simple ! )
```

Vous tomberez sur le fichier configuration suivant veuillez le remplir de la même façon :



Notre tâche s'exécute toutes les heures , le 0 a été rajouté pour ne pas faire la sauvegarde toutes les minutes.

Pour enregistrer faites la combinaison de touche ctrl + o puis entrer (y pour yes pour enregistrer)

Création du script de sauvegarde

Nous avons utilisé dans cet exemple l'utilisateur root de mysql pour effectuer la sauvegarde mais dans un cas de production il faut utiliser l'utilisateur archiviste qui a été créé pour cela.

A l'emplacement /var/www/backup/ veuillez créer le script backup_encrypt.sh à l'aide de nano et entrer les lignes suivantes :

```
nano backup_encrypt.sh

#!/bin/bash

/usr/bin/mysqldump -u root -ptoto  -B sae_23 --skip-add-locks --lock-tables=false --single-transaction --quick --order-by-primary --tr
```

Ce script fait donc un mysqldump sur notre base de donnée sae_23 en ne verrouillant pas les tables pendant le dump et en utilisant le principe de single transaction. Une fois cela fait avec un pipe qui permet de rediriger la sortie standard de mysqldump dans l'entrée standard de openssl , on chiffre la base de donnée (avec salt) avec le mot de passe "somepass" et on sauvegarde le résultat à l'emplacement suivant /var/www/backup/encrypted_backup_name.sql.enc .

Il ne faut pas oublier de lui ajouter les droits d'exécutions : chmod +x backup_encrypt.sh

Encore une fois il est important de mentionner que l'utilisation de l'utilisateur root pour une telle manipulation est dérisoire et est dangereux, un utilisateur mysql comme archiviste pourrait parfaitement remplir cette tâche sans mettre en danger l'intégrite de la base de données.

Voici le script permettant de déchiffrer le dump de la base de données chiffrés :

```
#!/bin/bash

openssl aes-256-cbc -d -salt -in /var/www/backup/encrypted_backup_name.sql.enc -k somepass > /var/www/backup/decrypted_backup_name.sql
```

Pensez à mettre les droits d'exécutions avec chmod.

Initialisation et déploiement automatique au démarrage

L'intérêt de cette section est de mettre en place sur notre serveur un moyen efficace d'automatiquement démarrer les services et modifier les fichiers configurations à chaque démarrage du serveur pour gagner du temps à ne pas tout mettre en place à chaque utilisation nouvelle du serveur. Dans notre cas on utilisera init.d pour faire démarrer un script contenant les instructions nécessaires au fonctionnement du serveur web.

Dans une section relative à Apache2 j'ai soulevé un moyen de remplacer l'ip de servername automatiquement malgré le DHCP, nos manipulations que nous nous apprêtons à effectuer vont servir cette utilité.

Création du script d'initialisation

Dans un premier temps nous créer un dossier de travail qui va contenir notre script d'initialisation:

```
mkdir /var/www/init/
```

Puis on va créer et rédiger le script init.sh à l'aide de nano :

```
nano init.sh

#!/bin/bash

### BEGIN INIT INFO
# Provides:          Sae23
# Required-Start:    $local_fs $network
# Required-Stop:     $local_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Sae23 service
# Description:       Run Sae23 service
### END INIT INFO

IP=$(ip addr show ens33 | grep -oP '(?<=inet\s)\d+(\.\d+){3}') # pour le virtualhost
cat > /etc/apache2/sites-available/redirect.conf << EOF
<VirtualHost *:80>
    ServerName $IP

    RedirectMatch ^/$ https://www.facebook.com/
```

```
    <Location /site/>
        RewriteEngine On
        RewriteRule ^(.*)$ https://www.facebook.com/ [R=301,L]
    </Location>
</VirtualHost>
EOF


sudo -b bash /var/www/html/site/launch.sh # pour Django
bash /var/www/cert/conf.sh 1&>/dev/null  # pour le certificat TLS

exit 0
```

Ce script est indispensable pour l'automatisation/ la stabilisation et le dynamisme du serveur web puisque c'est celui-ci qui va configurer les différentes composantes du serveur : Apache2, Django(Gunicorn), Certificat TLS.

Le bloc init info a été ajouté pour que ce script soit exécuté après le script network puisqu'on souhaite récupérer l'IP de l'interface ens33 pour la substituer dans le fichier redirect.conf. Mais aussi pour l'identifier à l'aide sa description.

Le script exécute simplement les autres fichiers configuration pour déployer correctement le service web.

Les fichiers configurations ciblés et leurs emplacements dépendent uniquement de vos configurations précédentes.

Lien symbolique et mise à jour runlevel

Veuillez créer un lien symbolique du nom de votre choix dans le dossier /etc/init.d/ ciblant votre script d'initialisation :

```
ln -s /var/www/init/init.sh /etc/init.d/webinit
```

Une fois cela fait exécutez cette commande pour mettre à jour les startup directories :

```
update-rc.d webinit defaults
```

Vous pouvez désormais redémarrer votre serveur, ouvrir une votre page web sur votre client et constater que c'est comme si vous aviez configuré le serveur au préalable.