IUT de Colmar

SAE 23

Mettre en place une solution informatique pour l'entreprise

Marius Keltz
13/06/2024

SAÉ 23 : Mettre en place une solution informatique pour l'entreprise

1ère Partie : Création de la base de données externe au projet Django

Installation de SQL

Pour pouvoir créer une base de données, on va utiliser le service SQL qu'on peut télécharger sur ce lien. Ce logiciel permet de créer un serveur de bases de données qu'on va pouvoir utiliser pour ce projet. Lors de l'installation, faites attention lors de l'inscription d'un mot de passe root, il faudra vous en souvenir. Pour ma part, je vais utiliser comme mot de passe toto. Une fois l'installation terminée, lancez l'application MySQL Command Line Client et vous pouvez créer la base de données que vous souhaitez. Sur notre Github, vous pouvez retrouver le fichier .sql, il vous suffira de l'importer ou sinon vous pouvez utiliser un bloc note pour copier et coller.

Exemple de DataBase :

```
-- Créez la base de données

CREATE DATABASE IF NOT EXISTS <nom de la base de données>;

-- Utilisez la base de données

USE <nom de la base de données>;

-- Créez une table

CREATE TABLE <nom de la table 1>(
    id INT AUTO_INCREMENT PRIMARY KEY,
    pseudo VARCHAR(100) NOT NULL UNIQUE,
    nom VARCHAR(100) NOT NULL,
    prenom VARCHAR(100) NOT NULL,
    mail VARCHAR(254) NOT NULL UNIQUE,
    mdp VARCHAR(100) NOT NULL,
    type ENUM('professionnel', 'amateur') NOT NULL
);

-- Insérez des données dans la table

INSERT INTO filmographie_personne (pseudo, nom, prenom, mail, mdp, type) VALUES
    ('johndoe', 'Doe', 'John', 'johndoe@example.com', 'mdp123', 'amateur'),
    ('janedoe', 'Doe', 'Jane', 'janedoe@example.com', 'mdp456', 'professionnel'),
```

SAÉ 23 : Mettre en place une solution informatique pour l'entreprise

Mise en place de la base de données sur Django

Allez sur votre projet Django et dans le dossier de votre projet ou ce trouve votre fichier settings.py. Pour ma part, il se trouve dans SAE23/project_sae/project_sae/settings.py et il vous faudra y ajouter quelques lignes.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'filmoDBa',           #Nom de la base de données
        'USER': 'greg',               #Utilisateur de la base de données
        'PASSWORD': 'toto',           #Mot de passe de l'utilisateur
        'HOST': '192.168.0.72',       #Adresse Ip du Serveur MySQL
        'PORT': '3306',               #port utilisé par le serveur MySQL
    }
}
```

Il vous suffira par la suite de retourner sur Django pour mettre à la base de données avec ces commandes :

```
cd /SAE23/project_sae/
python manage .py makemigrations
python manage.py migrate
```

On peut maintenant relancer le serveur Django :

```
python manage.py runserver
```

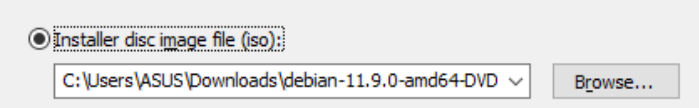
SAÉ 23 : Mettre en place une solution informatique pour l'entreprise

2ème Partie : Création de la relation Nginx, Gunicorn, Django et Mysql

Tout d'abord pour créer la relation entre Nginx, Gunicorn, Django et Mysql, il va nous falloir une machine virtuelle sous Debian 11 pour pouvoir héberger ces différents services. Pour pouvoir démarrer une machine virtuelle sur votre ordinateur, il va falloir activer la virtualisation.

Création de la machine virtuelle Debian 11 :

Nous allons récupérer l'iso de Debian 11 sur ce site et télécharger l'iso debian-11.9.0-amd64-DVD-1.iso. Personnellement pour la suite de la Sae, j'ai utilisé Vmware Workstation 16 Player comme logiciel de virtualisation. Vous allez créer une nouvelle machine avec l'iso que vous avez installée auparavant.



Normalement la distribution Debian 11 sera automatiquement détectée par le logiciel, on va créer un disque de 20 GB en Split virtual disk into mutiple files. Et maintenant que votre machine virtuelle est créée, vous pouvez augmenter les ressources allouées à la machine virtuelle en allant dans les settings, et activer le mode réseau Bridged qui sera nécessaire pour la suite. Maintenant vous n'avez plus qu'à lancer la machine et à suivre l'installation. Pour ma part, j'ai choisi d'installer sans interface graphique.

Une fois la machine virtuelle installée avec Debian 11, nous pouvons commencer la configuration des différents services. Durant toute la partie de la configuration, on sera en su et non en su- sauf exceptions.

Installations des services :

Nous allons commencer par mettre à jour les paquets puis installer tous les différents services qu'on va utiliser tel que Python3 pour avoir un environnement virtuel venv, python3 dev pour pouvoir exécuter des fichiers .py, postgresql pour la base de données Nginx pour pouvoir héberger le serveur web :

```
apt update
apt install python3-venv python3-dev libpq-dev postgresql postgresql-contrib nginx curl git sudo ufw
```

On va récupérer le projet Django qu'on a mis en ligne sur Github :

```
cd /home/toto
git clone https://github.com/Mariusklz/SAE23
cd SAE23
```

Création de la base de données sur la machine

On va créer une base de données dans la Vm, pour utiliser une base de données externe à Django. Pour faire cela, nous allons utiliser posqtgres installé auparavant, il permet de créer des bases de données psql.

```
sudo -u postgres psql
CREATE DATABASE filmographie;
CREATE USER toto WITH PASSWORD 'toto';
ALTER ROLE toto SET client_encoding TO 'utf8';
ALTER ROLE toto SET default_transaction_isolation TO 'read committed';
ALTER ROLE toto SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE filmographie TO toto;
\q
```

Ces commandes ont permis de créer une base de données nommée filmographie avec un utilisateur toto. De plus, elles ont permis de définir l'encodage de caractères par défaut à UTF-8, utilisé par Django, et de définir

aussi le schéma d'isolation des transactions par défaut à «read committed » ce qui permet de bloquer les lectures des transactions non validées.

Configuration de l'environnement virtuel et installation Django :

Pour installer l'environnement et Django nous allons utiliser python3 :

```
python3 -m venv venv
source venv/bin/activate
pip install django gunicorn psycopg2-binary pillow
```

Configuration de la base de données pour Django :

Dans le fichier project_sae/project_sae/settings.py, on va modifier quelques lignes pour voir le bon fonctionnement du serveur nginx ainsi que pour la base de données.

```
nano project_sae/project_sae/settings.py
```

```
ALLOWED_HOSTS = ['localhost','192.168.129.124']
```

L'adresse Ip est à modifier en fonction du réseau dans lequel on se trouve

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'filmographie',
        'USER': 'toto',
        'PASSWORD': 'toto',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Le nom, l'utilisateur, le mot de passe et l'host sont à modifier en fonction de la base de données créée auparavant. Le port par défaut utilisé par la Mysql est 3306.

Pour voir si ça fonctionne bien, on va faire une migration pour mettre à jour la base de données.

```
cd project_sae/
python3 manage.py migrate
python3 manage.py makemigrations
```

On va créer un super utilisateur pour administrer le projet :

```
python3 manage.py createsuperuser
```

Nous allons maintenant essayer de lancer le serveur Django pour voir s'il fonctionne correctement. Pour cela, il va d'abord falloir créer une exception de port pour le port 8000 :

```
sudo ufw allow 8000
python3 manage.py runserver 0.0.0.0:8000
```

Rendez-vous sur le site web avec comme url http://l'adresse_ip_du_serveur:8000 pour ma part ce sera http://192.168.129.124:8000.

Configuration de Gunicorn

Le but de Gunicorn est de pouvoir lancer le service Django, donc on va lancer le serveur à partir de Gunicorn :

```
gunicorn --bind 0.0.0.0:8000 project_sae.wsgi
```

Maintenant nous allons créer deux fichier, le fichier gunicorn.socket et le fichier gunicorn.service qui vont permettre de lancer Django automatiquement au démarrage de la Vm.

nano /etc/systemd/system/gunicorn.socket *nano /etc/systemd/system/gunicorn.service*

Gunicorn.socket

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target
```

Gunicorn.service

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target


[Service]
User=toto
Group=www-data
WorkingDirectory=/home/toto/SAE23/project_sae
ExecStart=/home/toto/SAE23/venv/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          project_sae.wsgi:application



[Install]
WantedBy=multi-user.target
```

Lancez Gunicorn et l'activer au démarrage du système :

systemctl start gunicorn.socket
systemctl enable gunicorn.socket

Vérification du lancement du service gunicorn.socket :

systemctl status gunicorn.socket
Vous devriez avoir la même chose que ceci :

```
● gunicorn.socket - gunicorn socket
     Loaded: loaded (/etc/systemd/system/gunicorn.socket; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-06-12 17:20:20 CEST; 1h 4min ago
   Triggers: ● gunicorn.service
     Listen: /run/gunicorn.sock (Stream)
     CGroup: /system.slice/gunicorn.socket
```

Configuration de Nginx :

On va créer un fichier dans le dossier /etc/nginx/sites-available/filmographie pour pouvoir ajouter le projet sur un site hébergé par Nginx, et ajoutez les lignes suivantes :

nano /etc/nginx/sites-available/filmographie

Ces lignes permettent de lancer le serveur sur le 80, qui est le port par défaut d'un site web. De plus, il donne l'accès au dossier static qui contient le fichier css du site web; sans cela, il n'arrive pas à le récupérer.

```
server {

    listen 80;
    server_name 'localhost';



    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/toto/SAE23/project_sae/filmographie/static/;

    }



    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }

}
```

On peut maintenant copier ce fichier et le mettre dans les sites enabled qui va pouvoir être hébergé par le service nginx :

sudo ln -s /etc/nginx/sites-available/filmographie /etc/nginx/sites-enabled

Vérifiez le fonctionnement de nginx, puis rédémarrez puis activez le service et désactivez l'exception créée au début sur le port 8000 pour donner l'accès complet à nginx.

systemctl restart nginx
systemctl enable nginx
sudo ufw delete allow 8000
sudo ufw allow 'Nginx Full'

S'il y a un problème lors du lancement du serveur nginx, faites ceci :

cd /etc/nginx/sites-enabled/
rm default
rm sites-enabledln

Par défaut Nginx crée des liens symboliques qui nous sont inutiles mais empêchent le lancement du serveur.

Enfin une fois tout ceci fait, il va falloir redémarrer la machine virtuelle pour voir si lorsqu'elle se lance, qu' elle lance les différents services et qu'on peut accéder au serveur Django sans problème.