DEPLOIEMENT D'APPLICATION

I) INTRODUCTION

Bonjour,

Dans ce PDF intitulé « Déploiement d'application », nous allons voir comment faire pour déployer notre site web effectué durant la SAE23 : Mettre en place une solution informatique pour l'entreprise.

Pour ce faire nous allons utiliser une machine virtuelle utilisant Debian, la version importe peu cependant c'est toujours mieux d'avoir la dernière version 😊.
Pour faire fonctionner Debian il faut un hyperviseur de machine, dans ce cas précis nous allons utiliser VMware qui est le logiciel que nous avons utilisé toute l'année à l'IUT.

Bien évidemment vous pouvez utiliser d'autre alternatifs comme VMware ou Ubuntu etc…

Une fois votre machine virtuelle et bien installé vous allez vous connectez en NAT ou bien en BRIDGE (pont), sans cela vous ne pourrez pas lancer votre application.

Par la suite il vous faudra installer des paquets pour que le déploiement se passe correctement, vous devrez mettre à jour vos paquets également, cela peut vous être très utile ! 👍

II) INSTALLATION DES PAQUETS

La première chose à faire est de mettre à jour ses paquets avec la commande suivante, bien évidemment il faudra se mettre en super-user pour faire les prochaines commandes. On ne peut pas mettre à jour ou installer un paquet sans être en sudo


```
root@toto-VirtualBox:/home/toto# apt update
```

Les prochains paquets à installer sont les suivants :

```
root@toto:~# apt install python3-pip apache2 libapache2-mod-wsgi.py3
```

On peut installer plusieurs paquets en une seule commande ce qui est plutôt pratique, nous allons installer « pip » premièrement, c'est le paquet qui va nous permettre d'installer python.

Apache2 est un serveur HTTP, donc un serveur web, quand on installe notre VM (Virtual machine) nous pouvons cocher l'option serveur web pour préinstaller le paquet. Vous pouvez également utiliser nginx pour le déploiement de votre application.

Le dernier paquet « libapache2-mod-wsgi.py3 » sert au déploiement de l'application quand on créer un environnement virtuel il faut obligatoirement le fichier wsgi. C'est ce qui va nous rediriger vers le site.

Exemple :

```
<Directory /path/to/mysite.com/mysite>
<Files wsgi.py>
Require all granted
</Files>
</Directory>
```

Les deux prochains paquets seront indispensables :

```
root@toto:~# apt install git ufw -y
```

Nous installons git car plus tard nous allons faire un « git clone », ce qui va nous permettre de récupérer le répertoire de notre SAE 😉.

Ufw va nous permettre d'autoconfigurer le pare-feu et de gérer les IPTABLES.

Gunicorn sera également utile, par la suite nous allons créer des fichiers unicorn

III) MISE EN PLACE

Nous voilà fin prêt à démarrer ! Nous allons directement commencer par clone le GitHub sur notre VM

```
root@toto:~# git clone https://github.com/julesbrt/SAE23-gestion-absences
```

Il vous faudra taper la commande git clone suivi de votre lien du répertoire créé auparavant.

L'installation de l'environnement virtuel se fera avec la commande pip3 car nous travaillons avec une version de python en 3. XX.

```
root@toto:~# pip3 install virtualenv
```

Par la suite, dirigez vous dans votre projet avec la commande « cd ».

Désormais, vous pouvez travailler dans votre environnement virtuel avec la commande suivante

```
root@toto:~/SAE23-gestion-absences/django/sae23# source sae23/bin/activate
(sae23) root@toto:~/SAE23-gestion-absences/django/sae23#
```

On peut bien voir que nous travaillions dans l'environnement virtuel

Bien évidemment n'oubliez pas d'installer Django en faisant un « pip install django » .

IV) LES DETAILS

Ayant utilisé une base de données SQL il faudra installer ce paquet, installez la version qui vous plaît, mais faites qu'elle soit récente.

```
(sae23) root@toto:~/SAE23-gestion-absences/django/sae23# pip install mysqlclien
==2.0.0
```

Il faut également installez mariadb

```
(sae23) root@toto:~/SAE23-gestion-absences/django/sae23# apt install mariadb-ser
ver libmariadb-dev
```

Faites également un pip install mysql.

Ensuite, allez dans le fichier settings.py de votre projet. Cherchez « ALLOWED_HOST » et mettez votre adresse IP en plus de l'adresse local.

Voici un exemple.

```
ALLOWED_HOSTS = ['192.168.X.X', '127.0.0.1']
```

Il faudra faire la migration car nous avons effectué des changements, à chaque changement que vous avez fait, veuillez faire les deux prochaines commandes.

```
./manage.py makemigrations
./manage.py migrate
```

Voilà vos migrations bien effectuées !

Pour avoir les permissions sur la base de données et pouvoir l'utiliser correctement faites cette commande :

```
mysql -u root -p'toto' < /home/toto/SAE23-gestion-absences/django/absences.sql
```

V) GUNICORN ET APACHE

Nous arrivons presque à la fin de notre installation ! il reste plus qu'a paramétrer apache2 et gunicorn.

Créer un fichier qui permettra de paramétrer le servie unicorn voici la configuration du fichier :

############## nano /etc/systemd/system/gunicorn.service

[Unit]

Description=gunicorn daemon

Requires=gunicorn.socket

After=network.target

[Service]

User=toto

Group=www-data

WorkingDirectory=/root/SAE23-gestion-absences/django/sae23

ExecStart root/SAE23-gestion-absences/django/sae23/.venv/bin/gunicorn --access-logfile - --workers 3 --bind unix:/run/gunicorn.sock sae23.wsgi:application

[Install]

WantedBy=multi-user.target ############

Bien évidemment je me suis aidé d'Internet pour trouver ces commandes.

Démarrez votre system gunicorn et rendez le active en faisant les commandes :

```
systemctl start gunicorn.socket systemctl enable gunicorn.socket
```

Maintenant, il faut configurer apache 2

Il faudra faire presque la même manipulation qu'avant sauf avec un autre dossier :

Nano /etc/apache2/sites-available/sae23

server {

listen 80;

```
    server_name 192.168.X.X;
    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        /root/SAE23-gestion-absences/django/sae23;
    }
    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

Redémarrez votre serveur apache2.

Nous allons maintenant utiliser ufw, faites les commandes suivantes.

```
ufw enable
```

```
ufw allow 80/tcp
```
On met « 80 » car c'est le port HTTP

VI) CONCLUSION

Vous pouvez désormais lancez votre serveur dans toutes circonstances avec la commande « python manage.py runserver 0.0.0.0 :8000 ». 😬

Mon ressenti sur le déploiement d'application est assez mitigé. J'aime bien utiliser Linux, cependant faire ceci m'était totalement inconnu pour moi, c'était une première. Ce fut difficile pour moi ayant des compétences minimes en Linux pour faire ça, cependant, ce fut très intéressant !

VII) SOURCES

https://github.com/ldsvrn/SAE23-TraficAerien/blob/main/server_install.sh

https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-apache-and-mod_wsgi-on-debian-8

https://www.youtube.com/watch?v=YnrgBeIRtvo&t=200s

MERCI