Florent

FoodPlanning

Dossier d'exploitation

Version 1.0

Auteur
Florent GRENAILLE
Développeur Python

Table des matières

1 - Versions

Auteur	Date	Description	Version
Florent	29/11/2013	Création du document	1.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application FoodPlanning.

L'objectif de ce document est de montrer la procédure de déploiement (mise en production) de l'application.

2.2 - Références

Pour de plus amples informations, se référer :

1. **DCT - FoodPlanning** : Dossier de conception technique de l'application.
2. **DCF – FoodPlanning** : Dossier de conceptoin fonctionnelle de l'application.

3 - Pré-requis

3.1 - Système

Pour une question de coût, les bases de données et l'application seront hébergés pour ma part sur le même serveur. (Serveur loué chez **A**mazon**W**eb**S**ervices)

Vous pouvez bien sûr, faire comme bon vous sembles.

3.1.1 - Serveur Web

Serveur physique ou virtuel hébergeant l'application web (Django) et les bases de données (PostgreSQL et Redis).

3.1.1.1 - Caractéristiques techniques

- Système d'exploitation : Ubuntu server 18.04 / Debian 9 / Raspbian Lite latest
- Processeur 1 cœur
- Minimum 1Go de mémoire vive (RAM)
- Une bonne connexion internet (minimum 1Mo/s)
- Stockage minimum 16 Go

3.2 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour :

- **PostgreSQL :** version 12.1
- **Redis :** version 5

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **Python :** version 3.7
- **Django :** version 2.2
- **Nginx :** version stable 1.16

4 - Procédure de déploiement

4.1 - Déploiement de l'application Web

4.1.1 - Mise à jour et préparation du serveur

Avant d'installer l'application web, il faut vérifier que le serveur est à jour et installer les prérequis. Pour ce faire voici connectez-vous au serveur web via SSH et effectuer les commandes suivantes.

```
$ sudo apt update && sudo apt upgrade -y
$ sudo apt install python3.7 git python-software-properties software-properties-common python3-pip
$ python3 -m pip install pip --upgrade --user
$ python3 -m pip install pipenv --user
```

Une fois, cela fait, redémarrer le serveur afin d'appliquer les mises à jour.

4.1.2 - Installer et configurer les bases de données

Pour que l'application puisse fonctionner, elle a besoin de deux bases de données. Une pour stocker toutes les données des utilisateurs (PostgreSQL) et une autre pour stocker les tâches d'envoi de SMS (Redis).

Pour ma part, je vais installer ses deux bases de données dans des containers Docker afin d'isoler proprement mes bases de données. Vous êtes libre de faire pareil ou pas.

Installation de Docker :

```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
$ sudo apt update
$ apt-cache policy docker-ce
$ sudo apt install docker-ce
```

Mise en place du container Docker PostgreSQL :

```
$ sudo docker pull postgres
$ sudo docker run --name some-postgres --restart always -e POSTGRES_PASSWORD=secretpassword -e POSTGRES_USER=userdb -e POSTGRES_DB=foodplanning_db -p 5432:5432 -d postgres
```

À vous de définir un mot de passe et un nom d'utilisateur pour la base de données.

Mise en place du container Docker Redis :

```
$ sudo docker pull redis
$ sudo docker run --name some-redis --restart always -p 6379:6379 -d redis
```

4.1.3 - Installation de l'application web et de ces dépendances

Pour installer l'application web, il faut la cloner sur le serveur.

```
$ cd ~
$ git clone https://github.com/riderflo85/foodplanning.git
```

Pour que l'application puisse fonctionner correctement, elle a besoins de package externe comme Django et Gunicorn par exemple. La procédure d'installation est très simple grâce à l'environnement virtuel *Pipenv* (installé précédemment) qui permet d'isoler les dépendances uniquement pour cette application.

```
$ cd ~/foodplanning/
$ pipenv install
```

4.1.3.1 - Variables d'environnement

Voici les variables d'environnement reconnues par l'application FoodPlanning :

Nom	Obligatoire	Description
DJANGO_SETTINGS_MODULE	Oui	Fichier de configuration de Django pour l'application FoodPlanning
SECRET_KEY	Oui	Clé secrète
HOST_DB	Oui	Nom d'hôte de la base de données (PostgreSQL)
USER_DB	Oui	Utilisateur de la base de données (PostgreSQL)
PWD_DB	Oui	Mot de passe de l'utilisateur de la base de données (PostgreSQL)
NAME_DB	Oui	Nom de la base de données (PostgreSQL)
PORT_DB	Oui	Port de la base de données (PostgreSQL)
LOGIN_CALLR	Oui	Nom d'utilisateur du compte Callr
PWD_CALLR	Oui	Mot de passe de l'utilisateur du compte Callr
IP	Oui	Adresse IP ou nom de domaine du serveur

Pour les notifications par SMS veuillez vous créer un compte sur https://www.callr.com/ (https://www.callr.com/#signup) afin de bénéficier des services de l'API Callr.

Définissez les variables d'environnement nécessaires comme ceci...

clé=valeur
exemple: *SECRET_KEY=ma_clé_secrète*

Les variables d'environnement doivent être inscrites dans un fichier *.env* à la racine de l'application.

```
$ cd ~/foodplanning/
$ nano .env
```

Voici le contenu du fichier *.env* :

```
DJANGO_SETTINGS_MODULE=foodplanning.settings.production
SECRET_KEY=new_secret_key
HOST_DB=localhost
USER_DB=user_db
PWD_DB=password_user_db
NAME_DB=foodplanning_db
PORT_DB=5432
LOGIN_CALLR=login_api_callr
PWD_CALLR=password_api_callr
IP=XXX.XXX.XXX.XXX ou NOM_DE_DOMAINE
```

Pour la variable « *SECRET_KEY* », indiquer le résultat que vous donne le script « *new_secret_key.py* ».

```
$ python3 new_secret_key.py
```

Vous aurez un résultat similaire à cela :

```
gnio7DxZSjYWNUG12R66Lnam
```

Pour les variables « *USER_DB* » et « *PWD_DB* », se sont les informations que vous avez renseignées lors de la création du container Docker PostgreSQL.

Pour les variables « *LOGIN_CALLR* » et « *PWD_CALLR* », se sont les informations de votre compte Callr.

Pour la variable « IP », indiquer l'adresse IP ou le nom de domaine de votre serveur.

4.1.4 - Configuration de l'application

4.1.4.1 - Collecter les fichiers statiques

Afin de faire fonctionner correctement l'application web, il faut réunir tous les fichiers statiques au même endroit pour que le serveur web (Nginx) puisse les servir aux clients.

On commence par créer un dossier qui recevra les fichiers :

```
$ cd ~/foodplanning/
$ mkdir foodplanning/settings/static
```

Activer l'environnement virtuel :

```
$ pipenv shell
```

Note : Pour quitter l'environnement virtuel vous pouvez faire CTRL+d ou la commande « *exit* ».

Ensuite, on réunit tous les fichiers statiques dans ce dossier :

```
$ ./manage.py collectstatic
```

4.1.5 - Compléter la base de données

La base de données est pour le moment vide, elle ne contient aucune table ni aucune données. L'application a besoin de différentes tables pour pouvoir fonctionner. Pour cela, il faut exécuter les migrations.

```
$ pipenv shell
$ ./manage.py migrate
$ ./manage.py pull_dishs
```

4.1.6 - Serveur web Nginx

Le serveur web utilisé pour traiter les requêtes des clients est Nginx.

4.1.6.1 - Installation de Nginx

Pour installer Nginx sur le serveur il faut ajouter le dépôt officiel de Nginx à la liste du système d'exploitation du serveur (Ubuntu Server pour ma part).

```
$ sudo add-apt-repository ppa:nginx/stable
$ sudo apt update
```

Ensuite on peut installer Nginx.

```
$ sudo apt install nginx
```

4.1.6.2 - Configuration de Nginx

Pour configurer Nginx, il faut créer un ficher et un lien symbolique de se fichier, dans le dossier d'installation de Nginx.

Le nouveau fichier :

```
$ sudo nano /etc/nginx/sites-available/foodplanning
```

Voici le contenu du nouveau fichier de configuration :

```
server {
    listen 80;
    server_name XXX.XXX.XXX.XXX;
    root /home/ubuntu/foodplanning/;

    location /static {
       alias /home/ubuntu/foodplanning/foodplanning/settings/staticfiles/;
    }

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-for $proxy_add_x_forwarded_for;
        proxy_redirect off;
        if (!-f $request_filename) {
```

```
            proxy_pass http://127.0.0.1:8000;
            break;
        }
    }
}
```

Remplacer « XXX.XXX.XXX.XXX » par l'adresse IP ou le nom de domaine de votre serveur.

Remplacer « ubuntu » par l'utilisateur système de votre serveur.

Activation de la nouvelle règle (nouveau fichier créer) par la création d'un lien symbolique.

```
$ sudo ln /etc/nginx/sites-available/foodplanning /etc/nginx/sites-enabled/
$ sudo rm /etc/nginx/sites-enabled/default
```

4.1.7 - Configuration de la supervision de l'application web

Afin de démarrer automatiquement l'application web, il est nécessaire de la superviser avec un système de supervision. Celui utilisé pour le projet, est, Supervisord.

4.1.7.1 - Installation de Supervisord

Installation depuis les dépôts du serveur.

```
$ sudo apt install supervisor
```

4.1.7.2 - Configuration de Supervisord

Pour que Supervisord, supervise l'application, il faut créer un fichier de configuration.

```
$ sudo nano /etc/supervisor/conf.d/foodplanning.conf
```

Voici son contenu :

```
[program:foodplanning-gunicorn]
command = /home/ubuntu/.local/share/virtualenvs/foodplanning-7pVMH_2r/bin/gunicorn
foodplanning.wsgi:application
user = ubuntu
directory = /home/ubuntu/foodplanning
autostart = true
autorestart = true
environment = DJANGO_SETTINGS_MODULE="foodplanning.settings.production",…
```

Pour la « *command* » vous devez indiquer le chemin vers le binaire de *gunicorn* qui est stocké dans votre environnement virtuel (créé avec *Pipenv*). Pour l'obtenir, lancer l'environnement virtuel.

```
$ cd ~/foodplanning/
$ pipenv shell
```

Vous aurez une résultat similaire à celui-ci :

```
/home/ubuntu/.local/share/virtualenvs/foodplanning-f_SnTg5j/bin/activate
```

Remplacer « activate » par « gunicorn foodplanning.wsgi:application »

Remplacer « ubuntu » par l'utilisateur système de votre serveur.

Pour les variables d'environnement (« environment »), vous devez spécifier toutes les variables

du fichier *.env* en les **séparant par des virgules et sans espace**. Pour la valeur de la variable, vous devez rajouter des guillemets comme dans l'exemple ci-dessus.

4.1.7.3 - Lancement de Supervisord

Commande de lancement de Supervisord :

```
$ sudo supervisorctl reread
$ sudo supervisorctl update
$ sudo supervisorctl status
```

4.1.8 - Configuration

Voici les différents fichiers de configuration :

- **foodplanning** : fichier de configuration qui définit les règles pour le serveur web (Nginx)
- **foodplanning.conf** : fichier de configuration pour le lancement automatique et la supervision (avec Supervisor)

4.1.8.1 - Fichier foodplanning

Ce fichier est le fichier qui contient les règles pour le serveur web Nginx. Grâce à ces règles Nginx sait où rediriger les requêtes effectuer par les utilisateurs et où chercher les fichiers statiques à servir aux utilisateurs.

4.1.8.2 - Fichier foodplanning.conf

Ce fichier est le fichier de configuration de Supervisor. Il contient les directives que Supervision doit suivre (emplacement de l'application, commande de démarrage, variables d'environnement...).

4.1.9 - Ressources

Pour plus d'information sur la configuration de Supervisor et de Nginx, consultez la documentation ci-dessous.

Supervisord: http://supervisord.org/index.html

Nginx : https://docs.nginx.com/nginx/

4.1.10 - Vérifications

Afin de vérifier la bonne configuration de Supervisor et de Nginx, faire ceci :

Nginx :

```
$ sudo systemctl status nginx
```

Doit renvoyer ceci :

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-07-03 13:55:46 UTC; 2 weeks 0 days ago
     Docs: man:nginx(8)
 Main PID: 3223 (nginx)
    Tasks: 2 (limit: 1152)
   CGroup: /system.slice/nginx.service
           ├─3223 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           └─3226 nginx: worker process
```

Supervisord :

```
$ sudo supervisorctl status
```

Doit renvoyer ceci :

```
foodplanning-gunicorn            RUNNING   pid 8119, uptime 0:35:32
```

Note: le *PID* sera différent

4.2 - Sécurisation du serveur

Les cyberattaques sont de plus en plus nombreuses et peuvent faire de gros dommages. Pour limiter cela, il est fortement recommandé de sécurisé son serveur web. Il existe beaucoup de ressources sur internet pour pouvoir effectuer cette tache correctement.

Voici quelques liens :

Comment sécurisé Nginx avec Let's Encrypt sur Ubuntu 18.04 :

https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-18-04

Comment protéger un serveur Nginx avec Fail2Ban :

https://www.digitalocean.com/community/tutorials/how-to-protect-an-nginx-server-with-fail2ban-on-ubuntu-14-04?fbclid=IwAR1MqbhTGAHg6UjYvlmBIDA8yheefgIkoq2QK_PhX3soef7mId__NkrtTDc

5 - Procédure de démarrage / arrêt

5.1 - Base de données

5.1.1 - Démarrage de la base de données

Le démarrage de la base de données se fait automatiquement lors du démarrage du serveur.

Pour en être sûr faire ceci.

```
$ sudo docker start some-postgres some-redis
```

5.1.2 - Arrêt de la base de données

L'arrêt de la base de données se fait automatiquement lors de l'arrêt du serveur.

Pour l'arrêter manuellement faire ceci.

```
$ sudo docker stop some-postgres some-redis
```

5.2 - Application web

Le serveur web (Nginx et Supervisor) démarre automatiquement lors du lancement du serveur.

Pour un démarrage manuel faire ceci.

```
$ sudo systemctl start nginx
$ sudo systemctl status nginx
$ sudo supervisorctl reload
$ sudo supervisorctl status
```

5.2.1 - Arrêt du serveur web

Le serveur web (Nginx et Supervisor) s'arrête automatiquement lors de l'arrêt du serveur.

Pour un arrêt manuel faire ceci.

```
$ sudo systemctl stop nginx
$ sudo systemctl status nginx
$ sudo supervisorctl stop all
$ sudo supervisorctl status
```

6 - Procédure de mise à jour

6.1 - Application web

Pour mettre à jour le serveur web vous pouvez exécuter ces commandes.

```
$ sudo apt update
$ sudo apt upgrade -y
```

Pour mettre à jour les dépendances de l'application web exécuter ces commandes.

```
$ cd /home/user/foodplanning/
$ pipenv shell
$ pipenv update
$ sudo supervisorctl reload
$ sudo supervisorctl status
```

7 - Supervision/Monitoring

7.1 - Supervision de l'application web

Afin de tester que l'application web est toujours fonctionnelles, faire ceci.

```
$ cd /home/user/foodplaning
$ pipenv run ./manage.test
$ sudo supervisorctl status
```

Si possible vous pouvez monitorer l'application avec Sentry. Cette solution n'est pas pré configuré car vous devez créer votre compte pour pouvoir associé un identifiant à l'application.

Sentry permet un suivi des erreurs qui aide les développeurs à surveiller et à réparer les pannes en temps réel. Itérer en permanence. Augmenter l'efficacité. Améliorer l'expérience utilisateur.

7.1.1 - Ressources

Sentry : https://sentry.io/welcome/

Création de compte : https://sentry.io/signup/

Documentation de Sentry pour Python : https://docs.sentry.io/error-reporting/quickstart/?platform=python

Commencez par vous créez un compte et suivez les étapes indiquées.

8 - Procédure de sauvegarde et restauration

8.1 - Procédure de sauvegarde de la base de données

La sauvegarde est une procédure de sécurité qui doit être effectuer régulièrement afin de prévenir toute perte de données et de permettre une restauration rapide du système et des données.

Pour sauvegarder la base de données et éviter de perdre vos données faire ceci.

```
$ cd /home/user/foodplanning
$ pipenv run ./manage.py dumpdata --indent 2 > save_database.json
```

8.2 - Procédure de restauration de la base de données

En cas de conflits, de soucis ou de dommages sur les données de votre base de données, vous devez procéder à la restauration de votre base de données. Grâce au sauvegardes régulière la restauration des données est simple.

Exécuter ces commandes sur le serveur web :

```
$ cd /home/user/foodplanning
$ pipenv run ./manage.py loaddata save_database.json
```

9 - Automatisation de tâches manuelles

Certaines tâche précédemment expliquées, peuvent être automatisées et exécutées régulièrement de façon autonome.

L'outil utilisé est ***cron***. Il est installé de base sur les systèmes Linux et permet d'exécuter des commandes définit par l'utilisateur à intervalle régulier.

9.1 - Configuration de cron pour le serveur web

Pour configurer ***cron*** vous devez effectuer ces commandes sur le serveur web :

```
$ cd /home/user
$ crontab -e
```

À la fin du fichier ajoutez les lignes suivantes :

```
0 0 * * 7 sudo apt-get update && sudo apt-get upgrade -y
30 0 * * 7 cd /home/user/foodplanning && pipenv update && sudo supervisorctl reload
0 0 * * * cd /home/user/foodplanning && pipenv run ./manage.py dumpdata --indent 2 > save_database.json
```

Pour le serveur web, ***cron*** se chargera de la mise à jour du système, de la mise à jour de l'application et de la sauvegarde de la base de données.

9.2 - Ressources

Pour plus d'informations sur ***cron***, consultez les liens suivants :

https://doc.ubuntu-fr.org/cron

https://fr.wikipedia.org/wiki/Cron

https://www.linuxtricks.fr/wiki/cron-et-crontab-le-planificateur-de-taches