Sandrine SUIRE 27/11/19

suiresandrine1@gmail.com

PROJET N°10 : Déployez votre application sur un serveur comme un pro !

github : https://github.com/sandrinesuire/project_10.git

trello : https://trello.com/b/L1O4W28w

site : http://209.97.190.27/substitute/searching

Installation

Clonage et mise à jour

Le projet de plateforme nutella existait déjà sur mon espace Github, je l'ai cloné sur mon disque local, puis remis en mode de fonctionnement dévelop car j'avais déjà modifié les paramètres de fonctionnement pour l'hébergé sur Heroku.

Intégration continu

La mise en place de l'intégration continu avec Travis a été très simple et rapide. Par contre, n'ayant pas encore séparé les fichiers de configuration de l'application, j'ai fait beaucoup de micro commits pour tenter de paramétrer correctement le fichier de Travis ainsi que le travis.yml. Travis me permettra donc de simuler le serveur de production en local et ainsi de vérifier que tous les tests de l'application passent avec succès.

Configuration d'un serveur mutualisé

La souscription à un serveur mutualisé est plutôt simple : créer un compte (digital océan), acheté l'espace de stockage en choisissant les paramètres souhaités. Là où les choses se complexifient un peu, c'est la connection en ssh, fournir une clé publique/privée, créer un nouvel utilisateur et lui donner les accès nécessaire. J'ai trouvé cette partie un peu plus complexe, mais plus j'avance dans ma formation et plus je trouve le travail avec la console attirant, nous donnant accès à ce qui se passe au plus profond de la machine.

Télécharger l'application sur le serveur

Avant de cloner l'application, j'ai fais un update de la machine, puis installé les applications python et postgresql nécessaire. J'ai ensuite, cloner mon application, installé et activé mon virtual environment.

Création de la base de données

J'ai créé ma base de données postgres selon les paramètres que j'ai indiqué dans mon fichier de settings production.py de django, créé l'utilisateur postgres correspondant et granté ses droits sur la nouvelle base.

Installation et paramétrage de NGINX

J'ai installé NGINX et créer son fichier de paramétrage, qui lui permettra d'écouter le port 80 de mon serveur et de rediriger ces requêtes vers le port 8000 du localhost de ma machine serveur.

Je lui indique également le répertoire pour servir les statics demandés par l'url /static, et enfin créer un lien symbolique dans site-enabled.

Installation et paramétrage de Supervisor

Afin de relancer le serveur gunicorn lorsqu'il s'arrête, j'ai installé supervisor. J'en ai profité pour installer newrelic qui me permettra de suivre la bonne santé de mon application python. J'ai créé le fichier de configuration pour le serveur gunicorn dans le dossier supervisor avec les paramètres gunicorn et les variables d'environnement nécessaire au démarrage de gunicorn.

Mise à jour hebdomadaire de la base de données

Afin de mettre à jour la base de données périodiquement j'ai créer une tâche cron qui exécutera un script bas de manière périodique. Ce script bash activera l'environnement virtuel dans le dossier de l'application, settera le fichier de settings de production, ce qui lui permettra de récupérer les paramètres de la base de données de production, et d'accéder aux commandes de django, exécutant donc une commande django qui récupèrera les données depuis l'api Openfoodfacts et les mettra à jour dans la base de données.

Installation et paramétrage de Sentry

Afin de suivre les erreurs de l'application, j'ai installé raven et paramétré Sentry dans les settings de production.

Démarrage

Pour démarrer mon application, j'ai créé et versionné un script de démarrage que je pourrais lancer depuis mon serveur. Ce script gère les variables d'environnement, l'installation des requirements, la collecte des fichiers statiques, les migrations et le démarrage des services supervisor, gunicorn et postgresql (qui est automatiquement démarré normalement lors de son installation).

Mis à jour

De Même que pour le démarrage, j'ai créé un script qui gère les variables d'environnement, l'installation complémentaire des requirements, la collecte des nouveaux fichiers statiques, les migrations et le redémarrage des services supervisor, gunicorn et postgresql.

Arrêt

J'ai créé enfin un script qui permet d'arrêter les différents services.

Monitoring

Depuis digital océan et newrelic, j'ai analysé les graphiques mis à disposition et construits les graphiques manquants à l'analyse de l'état de santé de mon serveur et de mon application.