Développement d'Applications Python – Projet 10
« Déployez votre application sur un serveur comme un pro ! »

1. Lien vers le code source du projet sur GitHub

https://github.com/etienne86/oc_p10-pur_beurre

2. Reformulation succincte du sujet

L'objectif de ce projet est d'utiliser le principe de l'intégration continue, le déploiement sur un serveur de production, le suivi (monitoring) du serveur et de l'application, ainsi que la mise en place d'une tâche régulière automatisée : ces quatre aspects ont attrait à de nouvelles compétences dans le cadre de la progression dans le parcours DA Python.

En particulier, le projet 10 fait suite au déploiement d'une application Django, réalisé avec Heroku dans le cadre du projet 08.

L'application déployée est accessible en suivant ce lien : http://178.62.10.30/.

3. Gestion du projet

Depuis le projet 04, j'utilise la méthodologie agile pour planifier au mieux mon travail. Voici le tableau de suivi que j'ai utilisé pour piloter le projet 10 :

https://trello.com/b/YmSrQjOa/e-barbier-oc-da-py-p10-pur-beurre

4. Déroulement du projet

4.1. Intégration continue

Pour travailler avec l'intégration continue dans ce projet, j'ai choisi l'outil Travis. J'ai trouvé cette méthode très vertueuse puisqu'elle exécutait notamment le plan de tests à chaque nouvelle version de mon code sur une branche dédiée. Cela permet donc de limiter les problèmes potentiels, en ne mettant en production que du code qui passe ces tests. Cela a été très agréable à utiliser, et je compte désormais l'utiliser pour tous mes prochains projets.

Cependant, j'ai connu quelques difficultés lors de la configuration de Travis, notamment sur deux points :

- Mes tests fonctionnels avec *selenium* utilisant le driver *geckodriver*, j'ai essayé d'installer ce driver directement sur le serveur, sans succès pour le bon fonctionnement, avant de trouver une solution plus pérenne avec le fait d'indiquer le paquet *firefox-geckodriver* dans la rubrique *addons* de la configuration de Travis
- De plus, j'ai cherché longuement comment passer deux variables d'environnement en parallèle dans le fichier de configuration de Travis, puisque cela générait initialement deux configurations différentes au lieu d'une seule

4.2. Déploiement sur un serveur de production

J'ai choisi d'héberger mon application sur un serveur Ubuntu 18.04 de DigitalOcean. La prise en main de l'outil s'est avérée assez aisée, notamment avec les explications données dans le cours recommandé dans le cadre de ce projet.

J'ai utilisé le serveur web Nginx pour ce projet. Afin d'utiliser un réel serveur de production, j'ai utilisé Gunicorn, piloté avec Supervisor. L'utilisation et la configuration de ces trois services n'ont pas posé de problème.

4.3. Monitoring

Pour surveiller l'activité du serveur, j'ai utilisé l'agent fourni par DigitalOcean, afin de surveiller une utilisation éventuellement trop importante de la mémoire vive ou de la capacité des processeurs.

Pour surveiller l'activité de l'application Django, j'ai utilisé Sentry qui permet de remonter des erreurs, avertissements et autres messages d'information : cela m'a permis de revenir sur le sujet des logs avec Python que je n'avais pas utilisé depuis très longtemps, c'est donc une très bonne chose !

4.4. Mise en place d'une tâche régulière automatisée

De même que les trois points précédents, la mise en place d'une tâche régulière automatisée avec Cron était quelque chose de nouveau pour moi. La prise en main de cet outil n'a pas posé de problème particulier. Au contraire, c'est le code avec Python qui a été plus compliqué que prévu, que ce soit pour la commande de mise à jour de la base de données à mettre en place, mais aussi les tests unitaires relatifs à cette nouvelle fonctionnalité, en particulier l'écriture d'un mock. En effet, la méthode à imiter était incluse dans un *context manager*, il a donc fallu écrire des méthodes spéciales __*enter*__ et __*exit*__ pour pouvoir y parvenir. Même s'il n'est pas agréable de consacrer plus de temps que prévu à l'implémentation et au test d'une fonctionnalité, je retiens ici que j'ai pu apprendre de nouvelles choses en Python, en particulier en comprenant mieux le principe du *context manager*.

4.5. Mise à jour du plan de tests

Afin de compléter le projet, j'ai également mis à jour le plan de tests initié au projet 08, avec les tests unitaires supplémentaires écrits dans le cadre de ce projet 10.

5. Retour d'expérience

J'ai trouvé ce projet très intéressant, différent des deux projets précédents qui étaient orientés sur le développement web (ils visaient à découvrir les frameworks Flask et Django, ainsi que le front-end avec HTML, CSS et Javascript). J'ai trouvé que ce projet, permettant une initiation et une découverte de l'administration système, avait finalement une connotation DevOps, même si cela est naturellement limité par le fait que j'ai travaillé seul, et pas dans le cadre d'une équipe DevOps.