Lien trello : https://trello.com/b/59qHIEHT/projet10

Lien GitHub: https://github.com/yannhamdi/projet8

Lien Application : 167.99.143.67

Dans le cadre de ce projet, nous avons comme instructions de déployez l’application du projet8 sur un serveur ainsi que d’effectuer quelques différentes tâches telles que suivre notre projet par le biais de « Travis », suivre les logs de l’application en utilisant « sentry » et finalement suivre les performances de notre site grâce à une fonctionnalité disponible sur digital ocean.

**DIFFICULTES :**

Ce que j’ai trouvé difficile sur ce projet du fait de sa nouveauté est le fait de travailler en parallèle sur le serveur et en local. Ainsi que de préparer tous les fichiers de configuration pour le bon fonctionnement de l’application en local et en production. J’ai éprouvé quelques difficultés pour modifier les fichiers settings et séparer les environnements.

**SOLUTIONS :**

Les cours de ce projet étaient relativement bien ficelés et les explications de configuration m’ont donc guidées de manière efficace vers le bon fonctionnement de l’application en production.

J’ai donc pour ce projet suivi minutieusement les différentes instructions du cours chapitre par chapitre et je dois avouer que ce projet s’est plutôt bien déroulé.

Je vais donc vous expliquer quelles ont été les étapes pour ce projet :

* Digital Océan
* Nginx
* Gunicorn et Supervisor
* Separation des environnements
* Travis
* Suivi performance sur Digital Océan
* Sentry
* Cron

**DIGITAL OCEAN :**

J’ai donc utiliser l’IAS « digital ocean » pour déployer mon application. J’ai créé un nouveau Droplet, qui va nous fournir une adresse IP qui nous permettra d’accéder à notre application à distance. C’est donc la création d’un serveur qui gérera notre application.

Puis en se connectant en ssh sur le serveur, nous installons les librairies dont nous avons besoin par le biais de git, nous clonons le projet que nous souhaitons déployer afin d’y déposer le projet sur le serveur. Nous installons l’environnement virtuel, ainsi que les requirements.

Nous effectuons la collection des fichiers statiques, nous effectuons les migrations, créons notre base de données sur le serveur.

Une fois, la configuration du serveur effective, nous crèons un nouvel utilisateur ssh et lui donnons tous les droits. Il ne reste plus qu’à configurer Nginx et Supervisor.

**NGINX :**

Nous devons à présent installer le serveur web Nginx afin de gérer tout trafic arrivant sur notre serveur. Nous devons effectuer quelques configurations dans le but de donner les instructions nécessaires à Nginx pour le bon déroulement de prise en charge de ce trafic.

**GUNICORN ET SUPERVISOR :**

Nous utiliserons le serveur HHTP python Gunicorn par le biais du module WSGI inclut dans Django, nous lancons le serveur Gunicorn de manière assez simple, une ligne de commande sur notre serveur et nous constatons que notre site est fonctionnel à l’adresse IP correspondante.

Nous sommes confronté tout de même à un autre souci, que ferions nous dans le cas ou le serveur est saturé et s’arrette de maniere inopinée, nous devrions relancer le serveur à la main, pas toujours évident de surveiller tout cela.

C’est pourquoi, nous allons installer supervisor qui fera cette tâche de surveiller notre serveur et le relancer au besoin. Nous devons bien sûr configurer un fichier qui donnera les instructions à supervisor pour relancer le serveur puis nous lancons supervisor. Nous rendons compte que tout fonctionne à présent.

**SEPARATIONS DES ENVIRONNEMENTS :**

A présent tout fonctionne mais il serait préférable de séparer les fichiers settings de configuration pour des raisons de sécurités. Nous créons donc 2 fichiers succints de configuration. Un en production qui ne sera pas visible sur notre git grâce gitignore et un fichier settings en local.

De cette manière, nous protégeons notre application en production.

**TRAVIS :**Nous allons utiliser un outil d’intégration continu afin de surveiller notre repository dans le cas de modification de projet. Nous mettons donc en place l’outil travis qui à chaque push relancera les tests et s’asssurera que tous les tests sont toujours effectifs et donc qu’il n’y aura aucune incidence sur le bon fonctionnement de notre application. Nous configurons donc un fichier .travis.yml qui lancera les tests automatiquement.

**SUIVI PERFORMANCE SUR DIGITAL OCEAN**

Nous allons présent nous assurer que les performances de notre serveur sont suffisantes pour notre application. Et Digital Ocean propopse une option de surveillance de ces performances. Nous executons donc une ligne de commande sur le serveur afin d’activer la surveillance.

Puis, nous renseignons une adresse mail afin d’être averti dans le cas d’une surtilisation des performances.

**SENTRY :**

Nous allons ensuite utiliser Sentry afin de surveiller les logs de l’application, sentry nous indiquera toutes les erreurs rencontrées sur notre application. Pour cela, nous ecrivons directement du code sur notre fichier settings.

**CRON :**

Une fois, notre application fonctionnelle, il nous faut la déployer sur Heroku. Le déploiement est relativement facile, la seule vrai subtilité exporter nos bases de données vers le serveur Heroku. Une fois les bases de données exportées, nous déployons notre application sur heroku.

L’application est enfin en production.

Je souhaite partager mon ressenti pour ce projet, j’ai trouvé la partie backend relativement facile, Django est relativement facile à prendre en main et le tuto officielle est très bien expliqué et bien détaillé.

Avec ce projet, j’ai découvert que j’avais énormément de lacunes au niveau front-end, il y beaucoup d’élément à gérer positionnement, aspect responsive. Je me rends compte que je dois peaufiner mes compétences au niveau CSS et Bootstrsap. Je me rassure en me disant que ce n’est que le deuxième projet que l’on prépare au niveau front-end alors que le backend, nous avons déjà eu un parcours bien complet.