Notes sur Flask/Gazouillis (flgaz) :

*Alan Pierre, EISI 20.2 CS*

Découverte des outils :

Le site Pythonanywhere.com permet de mettre en place et héberger des applications python directement sur le cloud, cela a quelques avantages, comme la possibilité de déployer très rapidement son application, ou encore de ne pas se gêner avec certains paramètres pouvant être présents avec des applications sur machine statique. (DNS, Redirection des ports, etc.)

Flask, quant à lui est un Framework de développement web fonctionnant sous Python. Il est utilisé pour effectuer des déploiements d’applications rapides et permet facilement d’indiquer le layout de l’application, qui prennent la forme de « @app.route() ».

Autre que Pythonanywhere, beaucoup d’autres sites utilisent Flask. Cela inclue des sites informatiques, tels que RedHat, Fedora, Ansible, etc mais aussi d’autres sites tels que Airbnb, Patreon, Uber ou encore Reddit. Au total, environ 1500 entreprises se servent de Flask.

Description des actions réalisées :

La première étape consistait à créer un repository afin de stocker nos fichiers nécessaires à la bonne mise en place de l’application. Pour cela, nous avons créé un compte sur Github et par la même occasion, un [repository public](https://github.com/Surnist/flask-gazouilles).

Une fois ce repository crée, il fallait d’abord créer une console bash et créer un environnement virtuel dans lequel nous allions pull les fichiers que nous venions de mettre dans notre repository :

mkvirtualenv --python=/usr/bin/python3.6 my-virtualenv

pip install flask

workon my-virtualenv

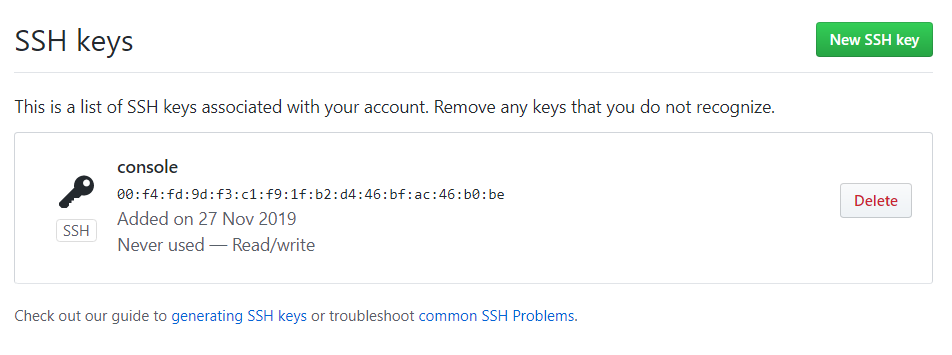
Avant d’importer cependant, il fallait créer une clef SSH afin d’être reconnu comme étant capable d’importer depuis notre repo, en l’ajoutant à notre profile Github.

ssh-keygen -t rsa -b 4096 -C « email@example.com »

eval $(ssh-agent -s)

cat ~/ssh/id\_rsa.pub #copier la clef à ce moment là

Une fois la clé récupérée, il faut aller dans les paramêtres de Github et ajouter la clef sous « SSH and GPG Keys » :



Puis, nous pouvons, sur notre machine locale, importer le projet :

git init # initialisation du git

git add app.py gazouilles.csv templates/ # ajout des fichiers

git remote add origin <https://github.com/Surnist/flask-gazouilles.git>

git push -u origin master # push vers le repo

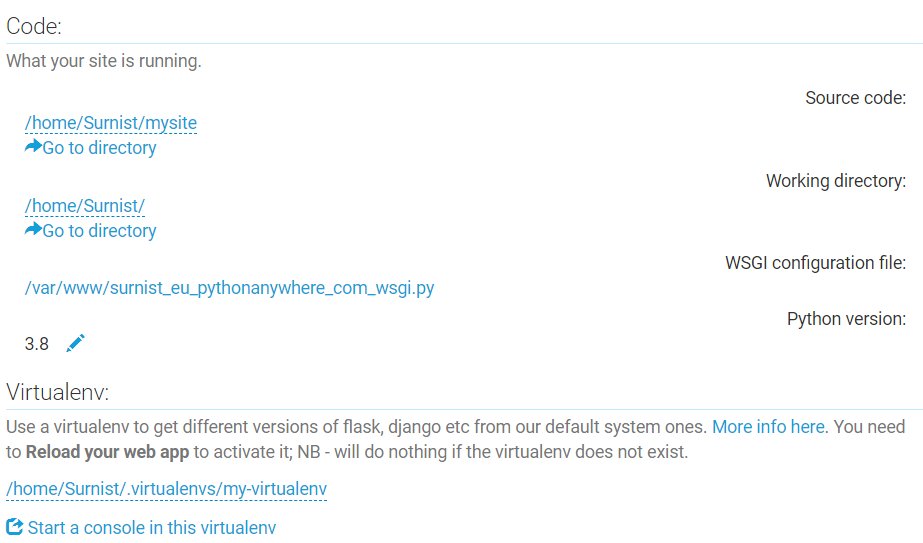
Une fois cela effectué, les fichiers se trouvaient sur le repo. Inversement, sur notre console bash, pour les importer, nous avons utilisé ces commandes :

git init # initialisation du git

git remote origin master [git@github.com:Surnist/flask-gazouilles.git](mailto:git@github.com:Surnist/flask-gazouilles.git)

git pull origin master

Les scripts désormais importés sur notre machine, il nous restait à configurer notre web app dans Pythonanywhere. Sur le Dashboard dans « Web App ». Il fallait modifier dans cette page les options dans « Code », à savoir le Source Code, Working Directory, WSGI Configuration file et le Virtualenv en suivant les emplacements de l’application.



Cela fait, il suffit de reload l’application et de s’y connecter.

Les problèmes sur l’application ont été rencontrés lors de la configuration sur le fichier WSGI, où j’ai cru qu’il fallait directement modifier le code au lieu de changer le fichier sur Pythonanywhere. Autrement, aucun autre problème n’a été remarqué.

Réflexions sur le projet :

Dans son état initial, il est impossible de changer de page dans le projet sans connaitre au préalable son chemin. Comme une application est supposée être utilisée par les clients, il est impensable de laisser l’application comme telle. Aussi, l’application n’est pas à l’abri de quoi que ce soit par rapport à ce qui est écrit sur le site.

Il est possible que des insultes ou autres propos racistes ou homophobes se retrouvent sur la timeline et que les personnes se trouvant sur la timeline voient ces insultes sans pouvoir y faire quelque chose.