DOSSIER D'EXPLOITATION

PAR

Quentin Lathière
Analyste/programmeur

https://github.com/Synkied/OC_Projet-9

NGL
https://ngl-it.io

14 rue du Japon, 75001 Paris – 06.00.11.22.33 – info@ngl.io
S.A.R.L. au capital de 5,00 € enregistrée au RCS – SIREN 000 111 222 – Code APE : 6202A

Versions

Auteur	Date	Description	Version
Quentin	09/04/2018	Création du document	1
Quentin	12/04/2018	Ajout informations mise à jour	1.1
Quentin	20/04/2018	Ajout précisions procédure de déploiement	1.2
Quentin	23/04/2018	Ajout schémas déploiement	1.3

Références

Pour de plus amples informations, se référer :

- **DC_OC-Pizza_Solution-technique_v01 ;**
- **DC_OC-Pizza_Gestion-fonctionnelle_v01 ;**
- **DC_OC-Pizza_PV_Livraison_v01.**

Table des matières

1. Objet du document

1.1 Objet

Ce document constitue le dossier d'exploitation de la plateforme web pour OC Pizza.

Ce document a pour objectif de présenter les différentes informations techniques permettant de comprendre la mise en en production de l'application et les différents procèdures liées à la bonne gestion de celle-ci.

2. Pré-requis

2.1 Système

2.1.1 Serveur de base de données

L'application est déployée sur **Heroku**, un serveur virtualisé aux **prix flexibles** suivant l'utilisation et les besoins de l'application. Plus d'informations ici : https://www.heroku.com/pricing

L'ensemble des modules de l'application est **géré et installé automatiquement** par Heroku lors du déploiement depuis GitHub, via Heroku GitHub Deploys.

Le projet est constitué de **deux interfaces** (client et employé), un **programme principal** (core) et une **base de données PostgreSQL**.
Chacune des deux interfaces de l'application est indépendante et dispose de sa propre instance Heroku (dyno) mais communique avec la même base de données (permettant un croisement simple des informations des utilisateurs). Pour une explication plus poussée, visiter : https://www.heroku.com/dynos.

2.1.2 Serveur web

Gunicorn fait office de serveur web pour l'application, comme recommandé par la documentation de Heroku : https://devcenter.heroku.com/articles/python-gunicorn. Ce serveur permet effectivement, contrairement au serveur inclu à Django, de faire t**ourner plusieurs processus Python** en même temps sur un même « dyno » Heroku.

2.2 Base de données

La base de données est une base PostgreSQL, comme recommandé par la plateforme Heroku. PostgreSQL a pour avantage d'être une base de données open source et n'appartenant pas à un grand groupe dont les intentions sont douteuses pour l'avenir. PostgreSQL est, et devrait rester gratuit à l'avenir (les donations sont cependanrt encouragées).

2.3 Web-services

Le serveur Heroku doit être maintenu en éveil.
Pour cela il suffit de posséder un abonnement payant auprès de Heroku.

3. Procédure de déploiement

3.1 Déploiement du paquet « core »

3.1.1 Installation

L'installation du paquet « core » repose sur un fichier **setup.py** créé par nos développeurs.
Un fichier **requirements.txt** est également requis pour l'installation des dépendances lors de la construction (build) des interfaces sur Heroku.

3.1.2 Variables d'environnement

Les variables d'environnement ci-dessous sont disponibles et reconnues par les différents fichiers du paquet.
La valeur de ces variables sur Heroku peut être vérifiée via la commande `heroku config`.

Nom	Description	Requis
ENV	Environnement de développement : PRODUCTION/ DEBUG	OUI
DATABASE_URL	Chemin vers l'URL de la base de données héber-gée par Heroku	OUI

3.1.3 Configurations

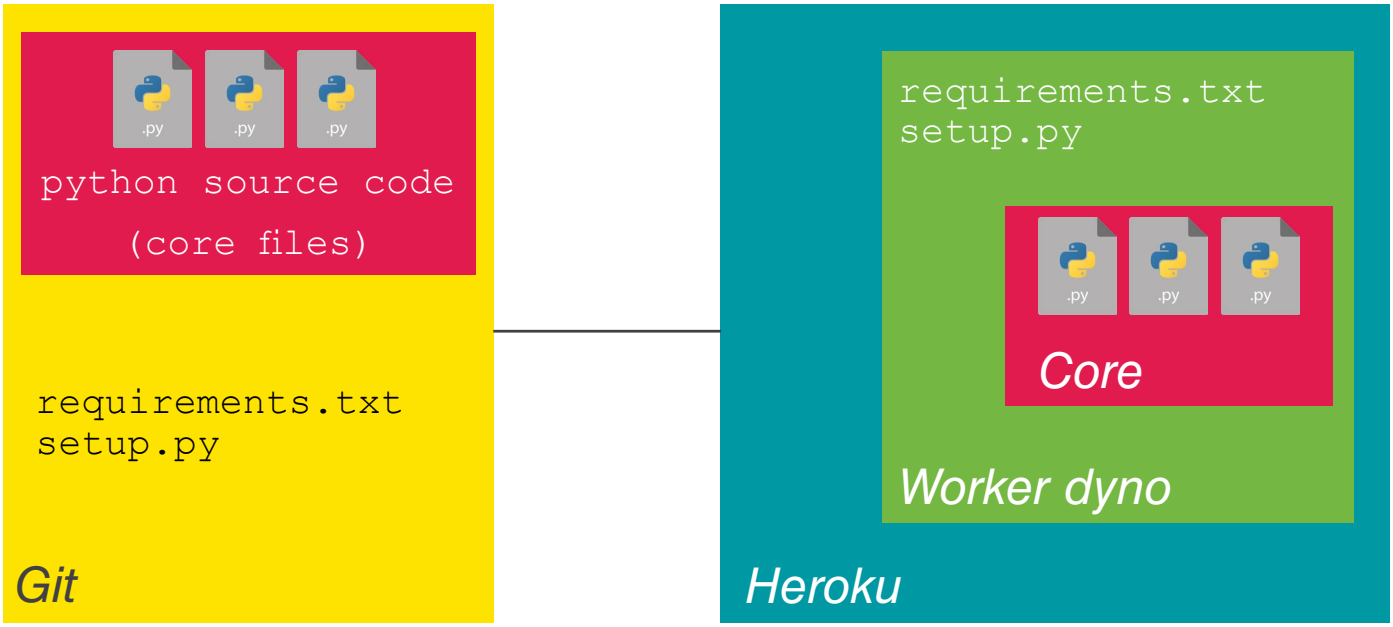
Heroku installe automatiquement les dépendances d'une application si celles-ci sont bien listées dans un fichier **requirements.txt, lors de la phase de construction (build) de l'application sur ses conteneurs.** Cela s'effectue lors du déploiement de l'application sur Heroku via GitHub.

Lorsque la construction de l'application a réussi, il suffit **d'éxecuter le fichier setup.py dans un terminal** (`python setup.py` ou `./setup.py`), afin de procéder à l'installation du paquet core sur Heroku.

3.1.4 Vérifications

Des tests peuvent être effectués grâce aux **tests unitaires associés au paquet**, via la commande `coverage run setup.py test`.

Une fois les tests effectués, un rapport de couverture peut être généré via la commande `coverage html`. Une couverture de tests optimale pour l'application est de de 90%.



Le déploiement du paquet core schématisé.

Pour en savoir + sur le fonctionnement d'Heroku :
https://devcenter.heroku.com/articles/how-heroku-works

3.2 Déploiement des interfaces

3.2.1 Installation

Les commandes liées aux intefaces sont toutes les commandes associées au fichier manage.py de Django. Ce fichier regroupe un pannel de commande utile au bon déploiement d'un projet Django. Une liste des commandes associées à ce fichier et leurs explications peut être trouvé ici : https://docs.djangoproject.com/fr/2.0/ref/django-admin/#available-commands

3.2.2 Variables d'environnement

Les variables d'environnement ci-dessous sont disponibles et reconnues par les différents fichiers des interfaces.
La valeur de ces variables sur Heroku peut être vérifiée via la commande `heroku config`.

Nom	Description	Requis
ENV	Environnement de développement : PRODUCTION/ DEBUG	OUI
CORE_PATH	Chemin vers le paquet core déployé sur Heroku	OUI
DJ_SECRET_KEY	Clé secrète utilisée par Django pour sécuriser l'application	OUI

3.2.3 Configurations

Un fichier Procfile est à créer pour chacun des deux paquets interface. Les fichiers Procfile sont des fichiers nécessaires à Heroku pour savoir quoi exécuter sur une instance. Pour en savoir + : https://devcenter.heroku.com/articles/procfile

Dans le cas de l'application OC Pizza, les fichiers Procfile doivent contenir une indication que le serveur web utilisé est gunicorn. Il suffit alors d'écrire au sein du Procfile :

```
web: gunicorn {{NOM_INTERFACE}}.wsgi
```

Où {{NOM_INTERFACE}} est le nom du fichier wsgi de l'interface (généralement le même nom que le nom du dossier dans lequel se trouve l'interface).

3.2.4 Vérifications

Comme pour le paquet core, des tests peuvent être effectués grâce aux **tests unitaires associés à l'interface.**

Mais tout d'abord, il est utile de savoir si Django se lance sans encombre lors du déploiement sur Heroku, via la commande :

```
python manage.py runserver
```

Si des bugs surviennent, leur nature sera affiché dans le terminal utilisé pour entrer la commande. Il sera alors nécessaire de les prendre en compte et régler les points problématiques.

Si aucun bug ne survient, il est possible de lancer les tests associée à l'interface, via la commande :

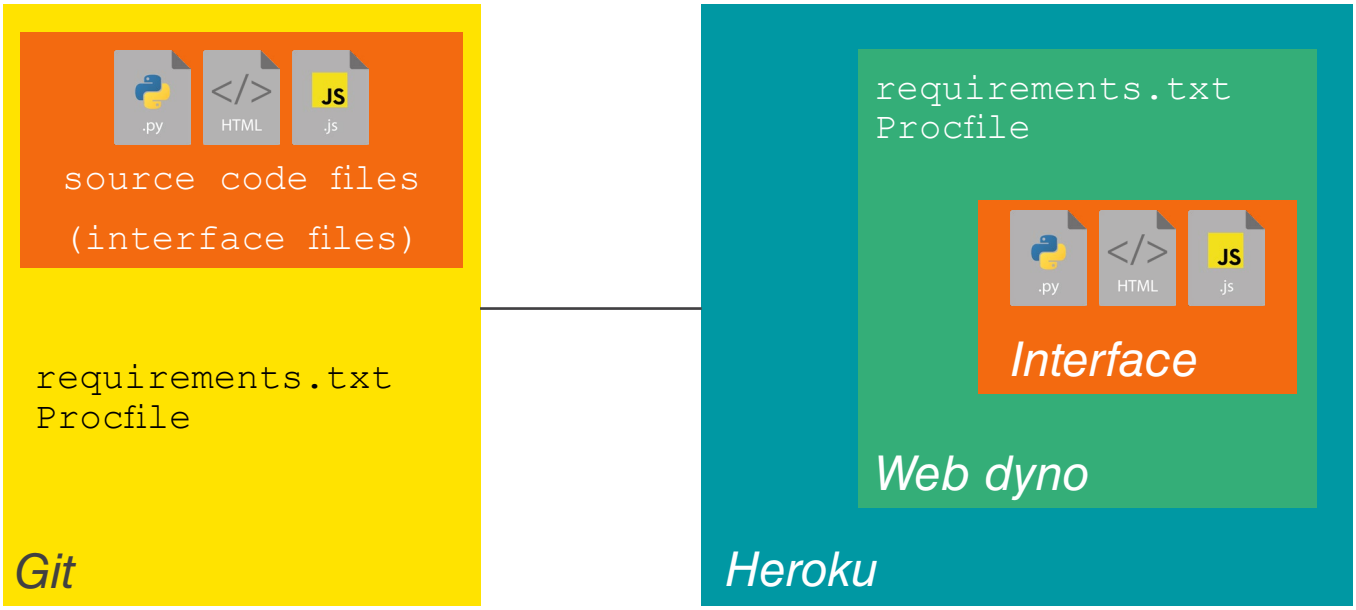
```
coverage run manage.py test
```

Puis, de générer un rapport HTML :

```
coverage html
```

Ou directement dans le terminal :

```
coverage report
```



Le déploiement d'une interface schématisé.

Pour en savoir + sur le fonctionnement d'Heroku :
https://devcenter.heroku.com/articles/how-heroku-works

4. Procédure de démarrage / arrêt

4.1 Cas général (base de données, application web)

L'application étant déployée sur Heroku, les procédures de démarrage et d'arrêt de celle-ci sont gérables via l'interface web fournie par Heroku (ou en ligne de commande).

Il est possible de gérer chaque dyno de l'application indépendamment les uns des autres.

Chaque dyno redémarre lors de l'installation d'un nouvel add-on, de changement des configurations de celui-ci ou lors de déploiement de nouveau code.
Les dyno redémarrent aussi au moins une fois par jour automatiquement afin de garder en bonne santé les applications déployées dessus.
Il est aussi possible de redémarrer un dyno de manière manuelle.

Pour en savoir + sur les dynos : https://devcenter.heroku.com/articles/dynos

5. Procédure de mise à jour

5.1 Base de données

La mise à jour de la structure de base de données devra se faire **manuellement**, après **concertation** avec l'équipe de NGL et **vérification** de la nécessité de celle-ci. Aucun procédure de mise à jour automatique n'est prévue pour l'application OC Pizza.

5.2 Application web

5.2.1 Design

Le design de l'application pourra être mis à jour après concertation avec l'équipe design de NGL.
Tout changement au delà du contrat initial occasionnera des frais supplémentaires.

5.2.2 Fonctionnalités

Les fonctionnalités de l'application ne seront communiquées à notre équipe technique qu'en cas de nécessité d'implémentation prouvée par OC Pizza, rapport à l'appui. Une fois la nouvelle fonctionnalité proposée à notre équipe technique, celle-ci jugera de son utilité et l'implémentera en conséquence, ou non.
Tout changement au delà du contrat initial occasionnera des frais supplémentaires.

NB : notre équipe technique pourra aussi être amenée à proposer des nouvelles fonctionnalités ou mises à jour suivant les dernières découvertes de failles ou mises à jour des dépendances de l'application.

6. Supervision / monitoring

6.1 Supervision de l'application web

Le monitoring de l'application est possible via le **dashboard** (interface web) mis à disposition par Heroku.

Cette supervision permet entre autre de diagnostiquer des bugs, mais aussi de connaître l'usage de l'application dans le temps.

Pour + d'informations sur la supervision via l'interface web Heroku :
https://devcenter.heroku.com/categories/monitoring-metrics

7. Procédure de sauvegarde et restauration

Nos développeurs ont conçu un script permettant la sauvegarde de la base de données à un temps donné (ou manuellement) afin de la répliquer en plusieurs endroits.

Par conséquent, la base de données est **sauvegardée toutes les nuits à 3h du matin** (afin d'éviter des conflits la journée) sur un serveur désigné par OC Pizza, et sur un serveur de NGL.

Aussi, Heroku effectue une **protection continue des bases de données** hébergées sur son site, voici un petit document expliquant comment :
https://devcenter.heroku.com/articles/heroku-postgres-data-safety-and-continuous-protection