OC Pizza

Application web OC Pizza

Dossier d'exploitation

Version 1.0.0

Auteur
Decroix Nicolas
Développeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
Decroix Nicolas	03/02/2020	Création du document	1.0.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application OC Pizza.

Ce document regroupe les procédures et opérations nécessaires à l'équipe technique pour pouvoir assurer une exploitation et une maintenance du système.

2.2 - Références

Pour de plus amples informations, se référer :

1. **DCT – 1.0.0** : Dossier de conception technique.
2. **DCF – 1.0.0** : Dossier de conception fonctionnelle.

3 - Pré-requis

3.1 - Système

3.1.1 - Serveur de Base de données

Le serveur de base de données Heroku PosgreSQL héberge le schéma de l'application web OC Pizza.

3.1.1.1 - Caractéristiques techniques

Le plan de la base de données Heroku utilise le plan « standard tier 0 » il dispose des caractéristiques suivantes :

- 4GB de mémoire RAM
- 64GB de limite de stockage
- Une limite de connexion de 120
- Prix de 50$ par mois

Le plan peut être modifié à tout moment en fonction des besoins et sans impact sur les données.

3.1.2 - Serveur Web

La plateforme Heroku reçoit l'application web mais il est nécessaire d'installer Gunicorn comme serveur web de production.

3.1.2.1 - Caractéristiques techniques

Serveur web Gunicorn :

- Prise en charge native du WSGI et Django.
- Une gestion automatique des procédures.
- Multiple configuration.
- Possibilité d'extension du serveur.

3.2 - Bases de données

Heroku support par défaut la version 11 du système de gestion de base de données relationnelle PostgreSQL. Il est aussi recommandé de mettre à jour la version du SGBD tous les 5 ans maximum sans quoi vous vous exposez à des dysfonctionnements de la base de données. Heroku vous informe un an au préalable lorsqu'une version n'est plus supportée pour que vous

puissiez migrer vers une version plus récente.

3.3 - Web-services

Les services web suivants doivent être accessibles et à jour :

- **Stripe :** version 2019-12-03, upgrades guide.
- **PayPal** : version 1.0 2019-03-22, release notes.
- **Heroku**.

4 - Procédure de déploiement

4.1 - Déploiement de l'Application Web

4.1.1 - Suivi de l'application avec Git

Il est nécessaire d'initialiser un dépôt local Git et d'y commit le code de l'application.

Suivez les instructions d'installation de Git du dossier de conception technique ou à ce lien.

Initialisation d'un dépôt Git :

```
Windows@DESKTOP-7DVVFF9 MINGW64 ~/desktop
$ cd ocpizza

Windows@DESKTOP-7DVVFF9 MINGW64 ~/desktop/ocpizza
$ git init
Initialized empty Git repository in C:/Users/Windows/Desktop/ocpizza/.git/

Windows@DESKTOP-7DVVFF9 MINGW64 ~/desktop/ocpizza (master)
$ git add .

Windows@DESKTOP-7DVVFF9 MINGW64 ~/desktop/ocpizza (master)
$ git commit -m "First commit"
[master (root-commit) 2e151a5] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.txt
```

Veillez à être sur la racine du projet ocpizza et non dans le sous-répertoire ocpizza/ocpizza.

4.1.2 - Création d'un remote Heroku

L'utilisation des commandes Heroku nécessitent l'interface en ligne de commande Heroku CLI disponible à ce lien ou en suivant les étapes d'installation du dossier de conception technique.

Nous commençons par créer une application sur Heroku :

```
$ heroku create ocpizza
```

Vous pouvez vérifier la création des remotes Git et Heroku avec la commande :

```
$ git remote -v
```

Et enfin envoyer le code sur Heroku :

```
$ git push heroku master
```

Vous pouvez vérifier la présence du site avec l'url résultant de la première commande create.

4.1.3 - Configuration de la base de données

Nous commençons par approvisionner notre application d'une base de données avec la commande :

```
$ heroku addons:create heroku-postgresql:<PLAN_NAME>
```

Remplacez <PLAN_NAME> par heroku-postgresql:standard-0.

Pour utiliser PostgreSQL comme base de données dans notre application nous allons avoir besoin du package psycopg2 :

```
$ pip install psycopg2-binary
```

Effectuer une connexion sur la base de données :

```
import os
import psycopg2

DATABASE_URL = os.environ['DATABASE_URL']

conn = psycopg2.connect(DATABASE_URL, sslmode='require')
```

Pour se connecter avec Django nous devons installer dj-database-url avec pip :

```
$ pip install dj-database-url
```

Et enfin ajouter le code qui suit dans le ficher settings.py afin de rendre la variable DATABASE_URL compréhensible par Django.

```
import dj_database_url
DATABASES['default'] = dj_database_url.config(conn_max_age=600, ssl_require=True)
```

Nous pouvons migrer la base de données sur la plateforme Heroku :

```
$ heroku run python manage.py migrate
Running `python manage.py migrate` attached to terminal... up, run.1059
Synchronizing apps without migrations:
  Creating tables...
```

Chargez les données avec la commande manage.py dumpdata si les données sont déjà présentes dans la base de données : **$ git run python manage.py dumpdata**

Si vous disposez d'un fichier JSON ou YAML vous pouvez vous en servir avec la commande loaddata à condition qu'il soit présent dans le dossier fixtures de l'application :

$ git run python manage.py <Nom du fichier>

Création d'un utilisateur administrateur : **$ heroku run manage.py creatsuperuser**

Remplir les champs Username, Email address et Password :

```
Username: admin
```

```
Email address: admin@example.com
```

```
Password: **********
Password (again): *********
Superuser created successfully.
```

4.1.4 - Envoie sur le serveur

L'envoie sur le serveur nécessite Gunicorn afin d'informer à Heroku la façon de lancer l'application ocpizza.

Dans le ficher Procfile à la racine de votre projet, ajoutez l'instruction :

- **web: gunicorn ocpizza:app**

Vous pouvez envoyer l'application sur le serveur : **$ git push heroku master**

4.1.5 - Environnement de l'application web

4.1.5.1 - Variables d'environnement

Nom	Obligatoire	Description	Données
DATABASE_URL	oui	URL d'accès de la base de données	**postgres://ajmwflxktovfnn:54fg4ndh4d74@ec2-23-23-86-179.compute-1.amazonaws.com:5432/d7ki8hbemu00vg**
STRIPE_SECRET_KEY	oui	Clé d'accès de l'API Stripe	**ea7107a800c54bb9d7aad3805bbc08ca**
PAYPAL_SECRET_KEY	oui	Clé d'accès de l'API PayPal	**126c77919a428a55f1c4955f216a29ba**
DJANGO_SETTINGS_MODULE	non	Chemin du fichier de réglages	**ocpizza/ocpizza/settings.py**

4.1.6 - Fichiers de configuration

4.1.6.1 - Settings.py

Ficher principale de configuration du projet Django. Il définit les réglages et variables d'environnement.

4.1.6.2 - __init__.py

Le fichier init identifie le dossier parent comme un paquet.

4.1.6.3 - Procfile

Permets à Heroku de connaître les commandes à exécuter lors du démarrage de l'application.

4.1.7 - Vérifications

Afin de vérifier le bon déploiement de l'application, naviguez dans le tableau de bord Heroku et dans le menu en haut à gauche, sélectionnez Production check. Heroku teste le bon déploiement de votre application et vous informe des erreurs si existantes.

5 - Procédure de démarrage / arrêt

5.1 - Application web

Pour démarrer votre application web il vous suffit de changer le nombre de dynos à minimum 1 en sachant que par défaut pendant le déploiement vous disposez déjà d'un dyno:

```
$ heroku ps:scale web=1 ocpizza
Scaling dynos... done, now running web at 1:Standard-1X
```

Pour arrêter l'application il suffit de mettre la valeur à zéro :

```
$ heroku ps:scale web=0 ocpizza
Scaling dynos... done, now running web at 0:Standard-1X
```

6 - Procédure de mise à jour

Les procédures de mise à jour nécessitent de mettre l'application hors ligne. Il est possible d'afficher une page de maintenance pour indiquer aux utilisateurs que le site est en maintenance.

6.1 - Mode maintenance

Pour customiser la page de maintenance afin d'indiquer aux utilisateurs le statut des serveurs suivez ce lien : Customize page.

6.1.1 - Gestion du mode

6.1.1.1 - Activer le mode maintenance

```
$ heroku maintenance:on
Enabling maintenance mode for myapp... done
```

6.1.1.2 - Désactiver le mode maintenance

```
$ heroku maintenance:off
Disabling maintenance mode for myapp... done
```

6.1.1.3 - Vérifier le statut du mode maintenance

```
$ heroku maintenance
off
```

6.2 - Base de données

Heroku dispose de deux méthodes de mise à jour :

1. pg:upgrade : Fonctionne sur tous les forfaits PostgreSql à l'exception du « hobby-tier »
2. pg:copy : Fonctionne sur tous les plans PostgreSQL, mais seulement sur les bases de données inférieurs à 10Gb et nécessite 3 minutes par GB de données.

La procédure qui suit concerne la commande pg:upgrade, pour la commande pg:copy suivre ce lien.

6.2.1 - Approvisionner une base de données de suivi

```
$ heroku addons:create heroku-postgresql:standard-0 --follow OC_PIZZA_DB_URL --app ocpizza
Adding heroku-postgresql:standard-0 to ocpizza... done, v71 ($50/mo)
Attached as OC_PIZZA_DB_FOLLOWER
Follower will become available for read-only queries when up-to-date
Use `heroku pg:wait` to track status

$ heroku pg:wait
Waiting for database OC_PIZZA_DB_FOLLOWER_URL... available
```

À partir de maintenant vous pouvez entrer en mode maintenance comme décrit en 6.1.

6.2.2 - Mise à jour de la base de données de suivi

Attendre la fin de tous les commits :

```
$ heroku pg:info --app ocpizza
=== OC_PIZZA_DB
Plan:        Standard 0
Status:      available
...
=== OC_PIZZA_DB_FOLLOWER
Plan:        Standard 0
Status:      available
...
Following:   OC_PIZZA_DB (DATABASE_URL)
Behind By:   0 commits
```

La valeur « Behind By » doit être à « 0 commits ».

Entrez la commande pg:upgrade pour mettre à jour la base de données de suivi :

```
$ heroku pg:upgrade OC_PIZZA_DB_FOLLOWER --app ocpizza
```

Promouvoir la base de données de suivi :

```
$ heroku pg:promote OC_PIZZA_DB_FOLLOWER
Promoting OC_PIZZA_DB_URL to DATABASE_URL... done
```

La base de données est à jour, vous pouvez arrêter le mode maintenance.

6.3 - Application web

La mise à jours de l'application s'effectue par un simple push depuis votre dépôt github qui contient le projet :

```
$ git push heroku master
Initializing repository, done.
updating 'refs/heads/master'
...
```

7 - Supervision/Monitoring

7.1 - Supervision de l'application web

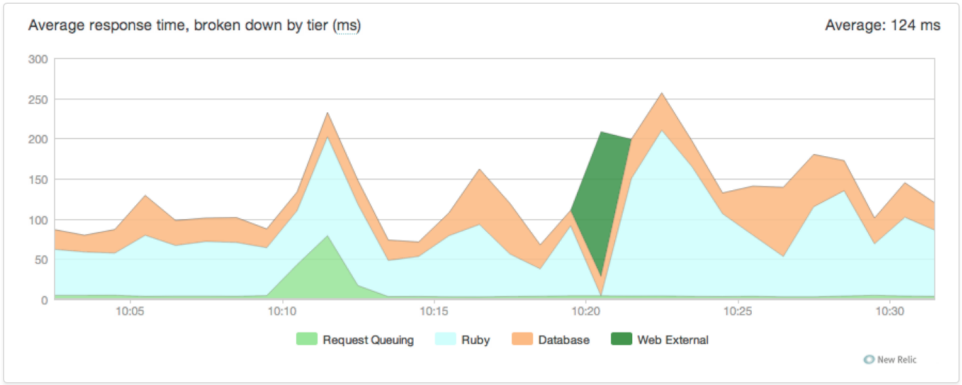
Afin de tester que l'application web est toujours fonctionnelle, Heroku dispose de l'outil de production check pour tester la configuration de l'application avec un certain nombre de critères.

Heroku dispose aussi d'une commande de supervision pour la base de données :

```
$ heroku pg:info -a ha-app
=== HEROKU_POSTGRESQL_CYAN_URL (DATABASE_URL)
Plan:               Standard 0
Status:             Available
Data Size:          6.4 MB
Tables:             0
PG Version:         9.4.1
Connections:        2/120
Fork/Follow:        Available
Rollback:           earliest from 2015-04-24 22:16 UTC
Created:            2015-04-24 22:14 UTC
Maintenance:        not required
Maintenance window: Mondays 20:30 to Tuesdays 00:30 UTC
```

Pour le monitoring nous disposons de deux outils :

1. **App monitoring** « *New Relic* » : qui nous indique des temps de réponse :



2. **Log monitoring** « *Papertrail* » : Service qui permet de créer des alertes en fonction des logs de notre application et de Heroku.

8 - Procédure de sauvegarde et restauration

8.1 - Procédure base de données

Pour la sauvegarde des données nous disposons de Heroku PGBackups qui nous permet d'effectuer les actions suivantes :

- Création de sauvegarde.
- Création d'une routine de sauvegarde.
- Téléchargement de sauvegarde.
- Vérification de l'état des sauvegardes.
- Suppression de sauvegarde.
- Restauration de sauvegarde.
- Copie de base de données.

Heroku nous permet d'effectuer 25 Backups. Si les 25 sauvegardes sont dépassées la plus ancienne est écrasée pour faire place à la nouvelle.

Documentation : Heroku PGBackups.

8.1.1 - Création d'une sauvegarde

Entrez la commande qui suit dans l'interface de commande Heroku :

```
$ heroku pg:backups:capture --app ocpizza
Hit Ctrl-C at any time to stop watching progress; the backup will
continue running. Stop a running backup with heroku pg:backups:cancel.
OC_PIZZA_DB (DATABASE_URL)  ----backup--->  b251

Running... done
```

En cas de problème vous pouvez annuler une sauvegarde :

```
$ heroku pg:backups:cancel

Canceled backup b252
```

8.1.2 - Programmer une sauvegarde régulière

8.1.2.1 - Programmer une sauvegarde

Sauvegarde programmée tous les jours à 2h00 :

```
$ heroku pg:backups:schedule DATABASE_URL --at '02:00 Europe/Paris' --app ocpizza
```

Remplacer les ‘ par des “ pour Windows.

Une sauvegarde régulière peut s’annuler sous certaines conditions :

- Si vous passez du forfait « hobby tier » à « production tier »
- Quand vous restaurez une sauvegarde sur une nouvelle base de données.
- Si vous mettez à jour votre base de données avec une « ollower database ».

8.1.2.2 - Vérifier la dernière Sauvegarde

```
$ heroku pg:backups:schedules --app ocpizza
Current backup schedules:
RED: daily at 2:00 (Europe/Paris)
```

8.1.2.3 - Arrêter une sauvegarde régulière

```
$ heroku pg:backups:unschedule DATABASE_URL --app ocpizza
```

8.1.3 - Télécharger une sauvegarde

Pour conserver une base de données en dehors des délais d’expiration de Heroku vous disposez de deux solutions.

8.1.3.1 - Généré une URL public de téléchargement

```
$ heroku pg:backups:url b001 --app ocpizza
The following URL will expire at 2015-04-07 18:35:50 +0000:
"http://s3.amazonaws.com/xkpgbackups/app1234567@heroku.com/b004.dump?AWSAccessKeyId=ABCD1234&Ex
```

L’URL reste disponible en public pendant 60 minutes avant expiration de cette dernière. Il est déconseillé d’utiliser une URL publique pour le transfert de données sensible.

8.1.3.2 - Télécharger la base de données directement

```
$ heroku pg:backups:download
```

8.1.4 - Suivi des sauvegarde

8.1.4.1 - Lister les sauvegardes

```
$ heroku pg:backups --app ocpizza
=== Backups
ID    Backup Time                Status                                Size    Database
----  -------------------------  ------------------------------------  ------  --------
b013  2015-03-18 19:03:16 +0000  Running                                       OCPIZZA
b011  2015-02-18 17:55:38 +0000  Finished 2015-02-18 17:55:39 +0000  1.9GB   OCPIZZA
b010  2015-02-17 19:14:43 +0000  Finished 2015-02-17 19:14:48 +0000  1.9GB   OCPIZZA
b004  2015-02-11 19:00:55 +0000  Finished 2015-02-17 19:14:48 +0000  1.9GB   OCPIZZA

==== Restores
ID    Restore Time               Status                                Size    Database
----  -------------------------  ------------------------------------  ------  --------
r002  2015-03-16 17:33:19 +0000  Finished 2015-03-16 17:33:19 +0000  1.9GB   OCPIZZA
r001  2015-03-15 12:13:44 +0000  Failed 2015-03-15 12:13:47 +0000    1.7GB   OCPIZZA
```

Nous disposons ici de la liste des sauvegardes disponibles à la restauration ainsi que les restaurations effectuées.

8.1.4.2 - Détail d'une sauvegarde

```
$ heroku pg:backups:info b017 --app ocpizza
=== Backup info: b017
Database: OC_PIZZA_DB
Started:  2013-06-24 17:11.28 UTC
Status:   Running
Type:     Manual
Size:     1.2GB
Schema:   0bff3ac
```

La commande ci-dessus demande les informations de la sauvegarde b017.

8.1.5 - Suppression d'une sauvegardes

```
$ heroku pg:backups:delete b101 --app ocpizza
Deleting b101... done
```

La suppression d'une sauvegarde s'effectue à l'aide de son Id. Toute sauvegarde supprimée ne peut être restaurée.

8.1.6 - Restaurer une base de données

Il est possible de restaurer une sauvegarde de trois manières différentes.

8.1.6.1 - Restauration à partir de l'URL de la base de données

```
$ heroku pg:backups:restore b101 DATABASE_URL --app ocpizza
```

B101 fait référence à l'identifiant unique de la sauvegarde à restaurer.

8.1.6.2 - Restauration depuis une autre base de données

```
$ heroku pg:backups:restore autreapp::b101 DATABASE_URL --app ocpizza
```

Autreapp fait référence à la base de données d'une autre application Heroku. La sauvegarde b101 de l'application autreapp sera restaurée dans notre application ocpizza.

8.1.6.3 - Restauration à partir d'une URL public

```
$ heroku pg:backups:restore 'https://s3.amazonaws.com/...' DATABASE_URL -a ocpizza
```

Remplacer les ' par des " pour Windows.

La commande pg:backups:restore ne permet pas d'effectuer une restauration partielle des données, la restauration d'une sauvegarde engendre la suppression de toutes les données présente sur celle-ci avant restauration.

8.1.6.4 - Copie des données entre base de données

```
$ heroku pg:copy OC_PIZZA_DB OC_PIZZA_DB_COPIE --app ocpizza
```

Les données de la base de données OC_PIZZA_DB seront copiées dans la base de données copie.

Pour effectuer la copie des données d'une application différente il est nécessaire de préfixer la

base de données source à l'aide du nom de l'application :

```
$ heroku pg:copy ocpizza::OC_PIZZA_DB NEW_OC_PIZZA_DB --app newocpizza
```

La base de données de l'application ocpizza est copiée dans la nouvelle application newocpizza.

8.2 - Procédure Application Web

Toutes modifications telles que le déploiement de code, changement des variables de configuration ou d'une extension engendrent la création d'une nouvelle release et d'un redémarrage de l'application. Il nous est ainsi possible de revenir à une version antérieure.

8.2.1 - Création d'une release

```
$ git push heroku master
...
-----> Compressing... done, 8.3MB
-----> Launching... done, v10
       http://severe-mountain-793.herokuapp.com deployed to Heroku
```

La version d'une release est indiquée par la lettre v suivie du numéro de version qui commence à 1 pour venir s'incrémenter à chaque nouvelle version.

8.2.2 - Lister les releases

```
$ heroku releases
Rel   Change                    By                  When
----  ----------------------    ----------          ----------
v52   Config add AWS_S3_KEY     jim@example.com     5 minutes ago
v51   Deploy de63889            stephan@example.com 7 minutes ago
v50   Deploy 7c35f77            stephan@example.com 3 hours ago
v49   Rollback to v46           joe@example.com     2010-09-12 15:32:17 -0700
```

La colonne « Change » indique la description des versions et lors de déploiements le hash dû commit Git déployé et indiqué.

Pour afficher le détail d'une release il suffit d'entrer la commande **heroku release:info** avec le numéro de version :

```
$ heroku releases:info v24
=== Release v24
Change:      Deploy 575bfa8
By:          jim@example.com
When:        6 hours ago
Addons:      deployhooks:email, releases:advanced
Config:      MY_CONFIG_VAR => 42
             RACK_ENV      => production
```

8.2.3 - Revenir à une release

La commande Heroku rollback permet de revenir à la version précédente sinon indiquer la version que vous souhaitez :

```
$ heroku rollback v40
Rolled back to v40
```

9 - Ressources

Python : https://docs.python.org/3/

Django : https://docs.djangoproject.com/en/3.0/

Django-paypal : https://django-paypal.readthedocs.io/en/stable/

Heroku : https://devcenter.heroku.com/categories/python-support

Stripe : https://stripe.com/docs/api

git : https://git-scm.com/doc

PostgreSQL : https://www.postgresql.org/docs/12/index.html

10 - GLOSSAIRE

Follower database	Base de données calquée sur une autre base de données pour donner un accès en lecture seulement aux données.
Release	Une realese fait référence à une version d'un logiciel.
Remote database	Base de données distante.