OC Pizza

IT Consulting & Development|

# OC Pizza

# SauceTomapp

Dossier d'exploitation

Version 1.0.0

**Auteur**
Louise Zanier-Roumieux
*Analyste-programmeuse*

# Table des matières

# 1 - Versions

| Auteur | Date | Description | Version |
|---|---|---|---|
| Louise Z. | 02/05/2020 | Création du document | 1.0 |
| | | | |
| | | | |
| | | | |

# 2 - Introduction

## 2.1 - **Objet du document**

Le présent document constitue le dossier d'exploitation de l'application **SauceTomapp**.

L'objectif de ce document est de présenter à l'équipe technique d'OC Pizza les informations essentielles à l'installation et l'utilisation de l'application.

## 2.2 - **Références**

Pour de plus amples informations, se référer :

1. **DCF – 1.0** : Dossier de conception fonctionnelle de l'application
2. **DCT – 1.0** : Dossier de conception technique de l'application

# 3 - Pré-requis

## 3.1 - Système

### *3.1.1 - Serveur de Base de données*

Le serveur de base de données est un conteneur **Docker**, basé sur l'image **postgres**. Dans la configuration actuelle, il tourne sur la même machine que le serveur Web, mais il sera possible de changer cette caractéristique dans le cas d'un agrandissement, les deux parties étant bien compartimentalisées.

### *3.1.2 - Serveur Web*

Le serveur Web consiste d'un conteneur **Docker**, à partir d'une image créée pour l'application, qui contient le projet **Django**, toutes ses dépendances Python, ainsi qu'un serveur **Nginx**. Contenir tout cela dans un conteneur permettra de faciliter l'agrandissement de l'application en pouvant lancer d'autres conteneurs sur la machine sans avoir de conflits de ports.

**Nginx** permettra également de servir les fichiers statiques du projet, puisque **Django** ne le fait pas en production.

Pour l'instant, le port 80 du conteneur sera lié au port 80 de la machine, mais il sera facile de changer cet état de fait s'il venait à falloir agrandir l'application.

## 3.2 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour :

- **PostgreSQL** : On utilise le schéma **ocpizza**, livré avec le projet.

## 3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **API Google Maps** : Il sera nécessaire de créer un compte afin d'avoir une clé API, et de gérer le paiement.
- **Gateway bancaire** : Il doit être là pour gérer les paiements en ligne.

# 4 - Procédure de déploiement

## 4.1 - Déploiement de l'application Web

### *4.1.1 - Contenu de l'application*

L'application **SauceTomapp** est un dépôt Git contenant les fichiers suivants :

- **Dockerfile** : Le fichier décrivant l'image **Docker**, qui vise à contenir l'application et son serveur.
- **docker-compose.yml** : Le fichier décrivant la composition de l'image **Docker** de l'application et de celle de la base de données, ainsi que leurs interactions, permettant d'installer les deux en même temps.
- **fabfile.py** : Le fichier décrivant les tâches nécessaires au déploiement, permettant de les exécuter facilement.
- **.env** : Le fichier contenant les variables d'environnement nécessaires au fonctionnement de l'application
- **requirements.txt** : Le fichier contenant toutes les dépendances du projet **Django**, ainsi que leurs versions.
- **manage.py** : Le script permettant de lancer l'application en mode développement, ou de lancer des utilitaires pour gérer l'application
- **saucetomapp** : Le dossier contenant les données du projet, comme les paramètres de l'application.
- **users** : Le dossier contenant l'application de gestion des utilisateurs.
- **orders** : Le dossier contenant l'application de gestion des commandes.
- **inventory** : Le dossier contenant l'application de gestion de l'inventaire.
- **templates** : Le dossier contenant les templates de base de l'application, afin d'offrir une expérience uniforme.
- **static** : Le dossier contenant les fichiers statiques communs à toutes les applications.

Pour déployer l'application, il suffit de lancer la tâche :

```
fab -H ocpizza@ocpizza.com deploy
```

La tâche se chargera de copier toutes les données aux endroits nécessaires, pourvu que l'utilisateur **ocpizza** ait les droits nécessaires sur le dossier **/var/www/ocpizza** (droits de lecture et d'écriture) et fasse partie du groupe **wheel**.

### *4.1.2 - Variables d'environnement*

Voici les variables d'environnement reconnues par le script de déploiement de l'application, et chargées avec elle subséquemment :

| Nom | Obligatoire | Description |
|---|---|---|
| SECRET_KEY | oui | La clé secrète de l'application, dont elle a besoin pour générer certaines données. Elle doit être générée par un générateur aléatoire cryptographique. |
| GMAPS_API_KEY | Oui | La clé d'API de Google Maps, préalablement enregistrée. |
| POSTGRES_PASSWORD | Oui | Le mot de passe de la base de données, il est la seule donnée obligatoire liée à la base de données. |
| POSTGRES_USER | Non | L'utilisateur de la base de données qui va être créé, le défaut est **postgres**. |
| POSTGRES_DB | Non | Le nom de la base de données créée. S'il n'est pas précisé, le défaut est de prendre la même valeur que **POSTGRES_USER**. |

Définissez les variables d'environnement nécessaires en remplissant les champs nécessares dans le fichier **.env**.

### *4.1.3 - Configuration*

Voici les différents fichiers de configuration :

- **.env** : La plupart de la configuration se fait dans ce fichier. En temps normal, vous n'aurez besoin de modifier que ce fichier.
- **saucetomapp/settings/production.py** : S'il faut modifier des paramètres de **Django**, c'est ce fichier qu'il faudra modifier. Il hérite des paramètres communs de l'application et permet de remplacer les valeurs de base.

### *4.1.4 - Vérifications*

Le retour de la tâche **Fabric** devrait établir avec quasi-certitude que l'application est bien déployée. Un message affichant «  deployment succeeded  » devrait être affiché à la fin, après d'autres indiquant les différentes étapes du déploiement.

## 4.2 - **Déploiement de la base de données**

Les étapes effectuées durant le déploiement de l'application devraient avoir déployés la base de données, mais pour une première utilisation il reste trois tâches à effectuer.

### *4.2.1 - Migrations de la base de données*

Pour créer le schéma, pour la première utilisation, ou le mettre à jour, pour les versions suivantes, il faut lancer une migration de la base de données. Cela se fait très simplement en lançant :

```
fab -H ocpizza@ocpizza.com migrate
```

Le script devrait indiquer la progression de la migration ainsi que son succès final.

### *4.2.2 - Création d'une pizzeria*

Pour créer une pizzeria, ce qui ne peut être fait depuis l'interface, il suffit de lancer la tâche :

```
fab -H ocpizza@ocpizza.com create_pizzeria
```

Le script demandera un nom de pizzeria, puis une localisation. Il sera possible de rentrer l'adresse, et le script fera un appel à Google Maps pour avoir des informations de localisation plus précise.

### *4.2.3 - Création d'un utilisateur initial*

Pour créer un utilisateur initial de la base de données, afin d'ensuite pouvoir créer les autres employés, il faut lancer une nouvelle tâche :

```
fab -H ocpizza@ocpizza.com create_user
```

Le script vous demandera les informations de base de cet utilisateur, soit son nom d'utilisateur, son nom, son adresse mail, son numéro de téléphone, et un mot de passe, et lui donnera un rôle de gestionnaire de point de vente. Il sera aussi demandé à quelle pizzeria l'associer, il est donc important qu'au moins une pizzeria soit créé, autrement le script refusera de s'exécuter.

### *4.2.4 - Vérifications*

Afin de vérifier le bon déploiement de ces données, il suffit de tenter de se connecter à l'application avec l'utilisateur créé.

# 5 - Procédure de démarrage / arrêt

Une procédure globale existe pour arrêter ou relancer tout les conteneurs :

```
fab -H ocpizza@ocpizza.com stop_everything
fab -H ocpizza@ocpizza.com restart_everything
```

Ces deux commandes lancent respectivement l'arrêt et le redémarrage de tout les conteneurs, c'est à dire ceux de l'application et de la base de données.

Il est également possible de relancer uniquement les conteneurs que l'on souhaite, ou de les arrêter. À noter par contre que si l'on stoppe uniquement le conteneur de la base de données, l'application ne pourra pas fonctionner.

```
fab -H ocpizza@ocpizza.com stop_application
fab -H ocpizza@ocpizza.com restart_application
fab -H ocpizza@ocpizza.com stop_database
fab -H ocpizza@ocpizza.com restart_database
```

Ces commandes font la même chose que celles décrites au dessus, mais celles préfixées par **application** ne concernent que le conteneur de l'application, et celles préfixées par **database** ne concernent que celui de la base de données.

# 6 - Procédure de mise à jour

Pour mettre à jour l'application, il suffit de lancer la tâche :

```
fab -H ocpizza@ocpizza.com update
```

Le script se chargera des mêmes tâches que pour la tâche **deploy**, mais s'assurera de garder les données, et de lancer une migration du schéma de la base de données. L'application sera indisponible pendant le temps (court, normalement) de déploiement de la mise à jour.

# 7 - Supervision/Monitoring

## 7.1 - **Supervision de l’application web**

Pour surveiller le fonctionnement de l’application Web, vous pouvez utiliser les métriques de l’hébergeur du serveur utilisé. Il devrait contenir un indicateur :

- L’utilisation du processeur
- L’occupation de la mémoire vive
- L’occupation de l’espace de stockage

Nous vous conseillons de surveiller ces données, afin de planifier un éventuel changement de serveur si les limites étaient presque atteintes.

# 8 - Procédure de sauvegarde et restauration

Il est aussi prévu dans le script de déploiement des tâches pour sauvegarder et restaurer la base de données. Les commandes à lancer sont les suivantes, en remplaçant **<fichier_de_sauvegarde>** par le réel nom du fichier de sauvegarde :

```
fab -H ocpizza@ocpizza.com save_data > <fichier_de_sauvegarde>
cat <fichier_de_sauvegarde> | fab -H ocpizza@ocpizza.com load_data
```

Le script vérifiera notamment que les données fournies sont intègres, avec un système de checksum et de signature. Il n'est pas possible de modifier ce fichier de sauvegarde à la main sans que le résultat ne le rende irrestorable. Notons aussi que malgré ces précautions, le fichier n'est pas chiffré, et nous vous recommendont vivement de le stocker dans un endroit sûr.

# 9 - Glossaire

| | |
|---|---|
| | |
| | |