Société : OC Pizza

-

Système de gestion informatique

Dossier d’exploitation

Table des matières

I. Cadrage du projet

I.1 Contexte de l'entreprise

La société OC Pizza est une jeune chaîne de pizzeria spécialisée dans la livraison ou la vente à emporter.
Les deux créateurs du groupe, Franck et Lola, souhaitent ouvrir 3 points de vente supplémentaires d'ici 6 mois, portant le nombre total de points de vente à 8.
Pour mener à bien ce développement, il est nécessaire au groupe de centraliser les données de ses points de vente, par l'intermédiaire d'un système informatique robuste. Cela permettra aux deux patrons d'avoir une vision globale du fonctionnement de chaque pizzeria, afin d'améliorer la gestion de la société.
De plus, ce système permettra d'améliorer l'efficacité des salariés en ayant un suivi en temps réel des livraisons.

I.2 Objectifs

Fournir à OC Pizza une solution clé en main, permettant aux clients d'effectuer leurs commandes en ligne, tout en centralisant les données du groupe dans un système de gestion d'entreprise.

I.3 Enjeux

- Rester compétitif face à la concurrence dense du milieu dans lequel évolue OC Pizza
- Augmenter la zone de chalandise de la société en proposant un site de vente en ligne
- Augmenter l'efficacité interne en améliorant la réactivité et la gestion des commandes
- Consolider les données du groupe pour avoir un meilleur pilotage d'entreprise

I.4 Objet

Le présent document est le dossier d'exploitation de l'application. Il se destine à l'équipe technique d'OC Pizza, afin d'aider à la procédure de déploiement de l'application.

II. Pré-requis

II.1 Système

II.1.1 Serveur web

L'application OC Pizza, ainsi que sa base de données seront hébergés sur le même serveur, auprès de l'hébergeur DigitalOcean.

II.1.1.1 Hébergement chez DigitalOcean

Commencer par louer un espace serveur (droplet) auprès de l'hébergeur DigitalOcean, avec un système d'exploitation UNIX (Ubuntu).

L'application étant à ses débuts, il est pertinent de se diriger vers une des premières offre d'hébergement, afin d'éviter de surcoûts supperflus.

Il sera possible, par la suite, d'évoluer vers une offre plus performante, en fonction de l'affluence de clients sur l'application. L'avantage des serveur cloud est que l'évolution vers un serveur plus performant se fait en seulement quelques clics à partir de l'interface web de DigitalOcean. Les performances du serveur actuel seront surveillées et contrôlées afin d'évaluer une nécessité d'évolution, se référer à la partie VI.1.

II.1.1.2 Activation du pare-feu

Pour limiter le type de connexions possibles au réseau, il est recommandé de configurer le pare-feu de DigitalOcean.

Pour se faire : *Networking → Firewalls → Create Firewall*

Type	**Protocol**	**Port range**	**Sources**
SSH	TCP	22	All IPV4 All IPV6
HTTP	TCP	80	All IPV4 All IPV6
HTTPS	TCP	443	All IPV4 All IPV6

Il faut penser à bien ajouter le nom du droplet pour que le pare-feu soit actif sur celui-ci.

II.1.1.3 Connexion au serveur

L'interaction avec le serveur se fait avec SSH, afin de sécuriser la communication.

Générer ses clés SSH en local : *ssh-keygen -t rsa*

Se connecter au serveur avec l'utilisateur *root*, crée par DigitalOcean : *root@IPSERVEUR*

Créer un utilisateur pour sécuriser le serveur : *adduser NOMUTILISATEUR*

Ajouter l'utilisateur au groupe *sudo* : *gpasswd -a NOMUTILISATEUR sudo*

Associer la clé publique à votre nouvel utilisateur :

- ➔ *su – NOMUTILISATEUR*
- ➔ *mkdir .ssh*
- ➔ *chmod 700 .ssh*
- ➔ *touch .ssh/authorized_keys*
- ➔ *vi .ssh/authorized_keys*
- ➔ Copier votre clé publique SSH (*cat id_rsa.pub*) dans le fichier, et sauvegarder
- ➔ *chmod 600 .ssh/authorized_keys*
- ➔ *exit*

Empêcher la connexion au serveur avec l'utilisateur *root* :

- ➔ *vi /etc/ssh/sshd_config*
- ➔ Passer le paramètre *PermitRootLogin* de *yes* à *no,* sauvegarder
- ➔ *service ssh reload*

II.1.1.4 Installer les librairies utiles au fonctionnement de l'application

L'application est écrite en langage Python 3.6.X et utilise le SGBD PostgreSQL 10.10, il est donc nécessaire d'installer ces librairies :

- ➔ *sudo apt-get update*
- ➔ *sudo apt-get install python3-pip python3-dev libpq-dev postgresql postgresql-contrib*

II.1.1.5 Télécharger l'application et créer l'environnement virtuel

Pour télécharger l'application sur le serveur, il suffit simplement de cloner le repository Github dans le répertoire de l'utilisateur : *git clone https://github.com/itcd/ocpizza-project.git*

Il faut maintenant créer l'environnement virtuel et installer les dépendances, se positionner dans le dossier du projet *ocpizza-project* :

- ➔ *sudo apt install virtualenv*
- ➔ *virtualenv env -p python3*

Activer l'environnement virtuel : *source env/bin/activate*

Installer les dépendances : *pip install -r requirements.txt*

II.1.2 Serveurs HTTP :

Nous utilisons 2 logiciels de serveur web pour transférer les requêtes de l'IP publique vers l'IP privée du serveur. Nginx est utilisé pour la version développement de l'application, et Gunicorn pour la production.

Installation :

- ➔ *sudo apt-get install nginx*
- ➔ *pip install gunicorn*

Pour contrôler le bon fonctionnement des processus sur le serveur nous utilisons Supervisor.

Installation : *sudo apt-get install supervisor*

II.2 Base de données

Il reste maintenant à créer la base de données, et à créer un utilisateur pour la gérer :

- ➔ *sudo -u postgres psql*
- ➔ *CREATE DATABASE ocpizza ;*
- ➔ *CREATE USER nomutilisateur WITH PASSWORD motdepasse ;*
- ➔ *ALTER ROLE nomutilisateur SET client_encoding TO 'utf8' ;*
- ➔ *ALTER ROLE nomutilisateur SET default_transaction_isolation TO 'read_committed' ;*
- ➔ *ALTER ROLE nomutilisateur SET timezone TO 'Europe/Paris' ;*
- ➔ *GRANT ALL PRIVILEGES ON DATABASE ocpizza TO nomutilisateur ;*

Il est maintenant nécessaire de renseigner les identifiants de l'administrateur de la base aux fichiers de configuration de l'application. Pour se faire, se placer à la racine du projet, et modifier :

- ➔ *vi OCPizza/settings/__init__.py (ici, ne pas renseigner le mot de passe)*
- ➔ *vi OCPizza/settings/production.py*

Migration des données, la commande suivante va remplir la base de données avec les données du dumps, et faire les migrations :

- ➔ *./manage.py populate_db*
- ➔ *./manage.py createsuperuser* (création de l'administrateur de l'application)

III. Déploiement de l’application web

III.1 Variables d’environnement

Par défaut, les réglages de l’application sont ceux de développement. Afin d’utiliser le paramétrages dédiés à la production, il faut définir une variable d’environnement *ENV == PRODUCTION*. Nous verrons comment la définir, grâce à Supervisor, dans la partie III.2.3.

Il est nécessaire d’utiliser cette variable d’environnement, car les configurations changent entre développement et production.

Pour générer une clé secrète Django, utiliser : *./manage.py gen_secret_key*

III.2 Fichiers de configurations

III.2.1 Module de configuration d’environnement

Les fichiers de configuration sont séparés en fonction de l’environnement. Ceux-ci se trouvent dans le module *settings* de l’application : *$home/user/ocpizza-project/OCPizza/settings/*

- ➔ Le fichier *__init__.py* contient toutes les configurations de l’application, et constitue les réglages pour l’environnement de développement. C’est le fichier qui est chargé par défaut sur le serveur.
- ➔ Le fichier *production.py* contient les paramètres dédiés à l’environnement de production. Notamment, une clé Django différente, voici une commande pour la générer : *./manage.py gen_secret_key*

  De plus, le mode DEBUG est désactivé dans cette configuration, car il ne doit pas être accessible en production.

  Ce fichier contient également le paramétrage pour Sentry (voir partie VI.2), ainsi que la configuration de la base de données.

III.2.2 Configuration de Nginx

Pour l’environnement de développement du projet, nous utilisons Nginx comme logiciel de gestion de serveur HTTP.

Il nécessite quelques configurations pour être fonctionnel :

- ➔ *sudo touch /etc/nginx/sites-available/ocpizza*
- ➔ *sudo ln -s /etc/nginx/sites-available/ocpizza /etc/nginx/sites-enabled*

Inscrire le code suivant dans */etc/nginx/sites-available/ocpizza :*

```
server {
   listen 80; server_name IPSERVEUR;
   root /home/USER/ocpizza-project/;
   location /static {
      alias /home/USER/ocpizza-project/OCPizza/staticfiles/;
   }
   location / {
      proxy_set_header Host $http_host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_redirect off;
      if (!-f $request_filename) {
         proxy_pass http://127.0.0.1:8000;
         break;
      }
   }
}
```

Redémarrer le service : *sudo service nginx reload*

Inscrire l'adresse IP publique du serveur dans le fichier *OCPizza/settings/__init__.py*, dans *ALLOWED-HOSTS*

Pour tester que le serveur Nginx fonctionne correctement, utiliser la commande :

- ➔ *./manage.py runserver*
- ➔ Se rendre sur un navigateur à l'adresse de l'IP publique

III.2.3 Configuration de Gunicorn et Supervisor

Le logiciel serveur web Gunicorn est installé, il est plus robuste que Nginx, et sera utilisé pour l'environnement de production.

Supervisor va permettre de contrôler que le service Gunicorn est bien actif, et le cas échéant va le relancer.

Téléchargement et configuration de Supervisor :

- ➔ sudo apt-get install supervisor
- ➔ sudo vi etc/supervisor/conf.d/ocpizza-gunicorn.conf

Ajouter les lignes de code suivantes au fichier crée :

[program:ocpizza-gunicorn]

*command = /home/**USER**/env/bin/gunicorn OCPizza.wsgi:application*

*user = **USER***

*directory = /home/**USER**/ocpizza-project*

autostart = true

autorestart = true

environment = DJANGO_SETTINGS_MODULE=«OCPizza.settings.production»

Lancer les processus :

- ➔ *sudo supervisorctl reread*
- ➔ *sudo supervisorctl update*

Commandes utiles :

- ➔ Lancer le serveur de production : *sudo supervisorctl start ocpizza-gunicorn*
- ➔ Arrêter le serveur de production : *sudo supervisorctl stop ocpizza-gunicorn*
- ➔ Redémarrer le serveur de production : *sudo supervisorctl restart ocpizza-gunicorn*
- ➔ Vérifier le fonctionnement du serveur gunicorn : *sudo supervisorctl status*

Pour tester que le serveur de production fonctionne bien, se rendre à l'adresse de l'IP publique après avoir démarré le serveur gunicorn avec la commande ci-dessus. (L'IP doit faire partie des *ALLOWED_HOSTS* de *OCPizza/settings/production.py)*

IV. Procédure de démarrage / arrêt

IV.1 Base de données

PostgreSQL tourne, de base, en tâche de fond.

Les commandes suivantes permettent tout de même de gérer le fonctionnement du serveur PostgreSQL :

- ➔ Démarrage de PostgreSQL : pg_ctl start
- ➔ Arrêt de PostgreSQL : pg_ctl stop
- ➔ Redémarrage de PostgreSQL : pg_ctl restart

IV.2 Application web

Dans le tableau ci-dessous se trouvent les commandes permettant de lancer, arrêter et redémarrer le serveur de l'application web en fonction des environnements :

Environnement virtuel	
Action	**Commande**
Activation	source env/bin/activate
Environnement de développement	
Action	**Commande**
Démarrage	./manage.py runserver
Arrêt	Ctrl + c
Environnement de production	
Action	**Commande**
Démarrage	*sudo supervisorctl start ocpizza-gunicorn*
Arrêt	*sudo supervisorctl stop ocpizza-gunicorn*
Redémarrage	*sudo supervisorctl restart ocpizza-gunicorn*

V. Procédure de mise à jour

V.1 Base de données

V.1.1 Ajout de produits

L'ajout de produits et/ou de recettes de pizzas se fait via l'interface de l'application. Il en va de même pour l'ajout de profils utilisateurs (magasins, livreurs, clients).

V.1.2 Modification du schéma de la base de données

Pour modifier la structure de la base de données il faut modifier les modèles Django relatifs. Pour modifier la structure des aliments et/ou recettes de pizzas :

➔ ocpizza-project/products/models.py

Puis, lancer les commandes :

➔ *./manage.py makemigrations*

➔ *./manage.py migrate*

Attention : il est fortement recommandé de sauvegarder la base de données avant de modifier la structure de celle-ci (voir partie VII.1).

V.2 Application web

Chez IT Consulting & Development nous mettons en place une procédure d'intégration continue, supervisée par Travis.

Ainsi, toute modification faite dans le but d'améliorer/corriger l'application OC Pizza devra se faire sur la branche *staging* du projet. Cette branche est soumis à validation de tests par Travis, après avoir push la modification sur GitHub. Cela permet de palier d'éventuels bugs qui pourraient intervenir au moment du merge à la branche principale.

Après le merge, il est nécessaire de relancer le serveur de production avec la commande dédiée (voir IV.2).

Travis utilise un fichier de configuration situé à l'emplacement suivant : *ocpizza-project/.travis.yml*

Il utilise un environnement dédiée pour fonctionner, situé ici :

ocpizza-project/OCPizza/settings/travis.py

VI. Monitoring

VI.1 Performances du serveur

Afin de surveiller les performances du serveur, et d'évaluer la nécessité d'évoluer vers une solution plus performante, nous mettons en place un monitoring des composants. Ce monitoring est un service proposé par DigitalOcean, compris dans l'offre de location du serveur.

Activer le monitoring DigitalOcean : *curl -sSL https://agent.digitalocean.com/install.sh | sh*

Il est maintenant possible d'accéder aux informations de performance du serveur dans l'espace d'administration de DigitalOcean.

Il est possible de mettre en place des alertes, afin d'être notifié lorsque une des performances atteint un seuil critique. Pour configurer ces alertes, cliquer sur *Monitoring → Create alert policy*

Nous conseillons généralement de surveiller les 4 composants suivants en priorité :

- ➔ *L'utilisation du processeur*
- ➔ *L'espace de stockage du disque dur*
- ➔ *L'utilisation de la mémoire vive (RAM)*
- ➔ *La capacité de la bande passante*

VI.2 Les logs de l'application

Pour surveiller les logs de l'application, et donc détecter d'éventuels bugs qui pourraient survenir, nous utilisons Sentry. Ce logiciel permet de faire remonter les alertes détectées dans les logs du serveur sur une interface web, afin de pouvoir en prendre connaissance et d'être notifié lorsqu'elles surviennent.

- ➔ Créer un compte sur Sentry.io
- ➔ Projects → Create project → Django
- ➔ *pip install –upgrade 'sentry-sdk==0.14.1'*

Ajouter le code suivant au fichier *ocpizza-project/OCPizza/settings/production.py* :

```
import sentry_sdk
from sentry_sdk.integrations.django import DjangoIntegration

sentry_sdk.init(
    dsn="https://3abcc05619744ee9a85c66eaadfcddc4@sentry.io/2520785",
    integrations=[DjangoIntegration()],

    # If you wish to associate users to errors (assuming you are using
    # django.contrib.auth) you may enable sending PII data.
    send_default_pii=True
```

Pour aller plus loin, il est possible de faire également remonter des informations sur les logs du serveur dans Sentry. Pour cela il est nécessaire d'utiliser le module logging de Django. Pour plu d'informations, se référer à : https://docs.djangoproject.com/fr/3.0/topics/logging/

VII. Procédure de sauvegarde / restauration

VII.1 Base de données

Avant toute modification de la base de données, nous vous conseillons d'effectuer une sauvegarde saine de celle-ci.

Sauvegarde des données actuelles :

➔ *./manage.py dumpdata > OCPizza/dumps/db_jj_mm_aa.json*

Restauration d'une version antérieure des données :

➔ *./manage.py loaddata OCPizza/dumps/db_jj_mm_aa.json*

Concernant la structure de la base de données, il est également possible de revenir à une version antérieure de celle-ci, car Django enregistre toutes les migrations faites sur l'application.

Commande pour revenir à une ancienne version de la structure de la base de données :

➔ *./manage.py migrate OCPizza* ***name_previous_migration***

VII.2 Application web

Pour le versioning de l'application nous utilisons le logiciel Git. Il permet de versionner l'application et de garder en mémoire les versions ultérieures de celle-ci. Ainsi, il est possible de revenir à une version antérieure du projet si nécessaire.

Git est un logiciel étant un logiciel très utilisé et documenté chez les développeurs, vous trouverez toutes les informations nécessaires sur : https://git-scm.com/doc

Pour rappel, la branche staging est la branche de travail utilisée dans le cadre du projet, il est indispensable de tester vos changements sur celle-ci afin de ne pas compromettre la branche principale du projet.