Contact : Jean-Baptiste Baillet
jbbaillet@itconsultingdevelpment.com

Dossier d’exploitation

du Projet

OC Pizzas

Date de création **15/01/2022**

Date de modification
Echéance livraison

Table des matières

1-VERSIONS

Auteur	Date	Description	Version
Jean-Baptiste	**02/01/2022**	**Création du document**	**1**

2-INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier d’exploitation de l'application OCpizzas

L’Objectif du document est de fournir à l'équipe d'exploitation les informations dont elle a besoin pour pouvoir assurer une exploitation en règle du système et pouvoir réagir de manière appropriée lorsqu'un problème surgit.

2.2 - Références

Pour de plus amples informations, se référer :

1. **Ppizza1_01_specifications_techniques** : Dossier de conception technique de l'application OCPizza
2. **Ppizza1_01_specifications_fonctionnelles** : Dossier de conception fonctionnelle de l’application OCPizza

3-Pré-requis

3.1 – Système et serveurs :

L'hébergement de la solution OCpizza se fera sur un serveur dédié, acheté sur Digital Ocean, avec une configuration minimum de :

- 1 To de stockage

- 8 Go de RAM

Le serveur fonctionnera sous Ubuntu 20.04 LTS.

Il sera également nécessaire d'installer les dépendances suivantes pour faire fonctionner la solution :

- Python : version 3.9

- Django : version 3.2.7

- Gunicorn : version 20.1.0

- psycopg2 : version 2.9.3

- Nginx : version 1.20.0

- Supervisor : version 4.2.4

- Poftpd : version 1.3.7c

- sentry-sdk : version 1.5.1

3.2 - Bases de données

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

- **PostgreSQL : version 14**

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **API du service de paiement bancaire :**
- **API de géolocalisation :** Google Maps

4-PROCÉDURE DE DÉPLOIEMENT

4.1 - Configuration initiale du serveur avec Ubuntu 20.04

Besoin de connaître l'adresse publique du serveur, ainsi que le mot de passe ou la clef ssh.

4.1.1 - Connexion en tant que root

```
# ssh root@your_server_ip
```

4.1.2 - Création d'un nouvel utilisateur

```
# adduser ocpizzauser
```

4.1.3 - Octroi de privilèges administratifs

```
# usermod -aG sudo ocpizzauser
```

4.1.4 - Activation de l'accès externe pour votre utilisateur régulier

```
# ssh ocpizzauser@your_server_ip
```

4.2 – Configurer Django avec Postgres, Ngix et Gunicorn sur Ubuntu 20.04

4.2.1 - Installation des paquets à partir des référentiels Ubuntu

```
# sudo apt update
# sudo apt install python3-pip python3-dev libpq-dev postgresql postgresql-contrib nginx curl
# sudo apt-get install proftpd
```

4.2.2 - Création de la base de données et de l'utilisateur PostgreSQL

Lancer le script createDB.sql

4.2.3 - Création d'un environnement virtuel Python

```
# sudo -H pip3 install --upgrade pip
# sudo -H pip3 install virtualenv
# mkdir ~/projectOCpizza
# cd ~/projectOCpizza
# virtualenv .venv
# source .venv/bin/activate
```

4.3 – Déploiement de l'application web

4.3.1 – Télécharger l'application sur le serveur

Télécharger l'application dans le dossier *home*/ocpizzauser/projectOCpizza

4.3.2 - Variables d'environnement

Voici les variables d'environnement reconnues par les batches de l'application ocpizza :

Nom	Obligatoire	Description
SECRET_KEY	**oui**	**Permet à Django de sécuriser l'application**
ALLOWED_HOSTS	**oui**	**Liste des adresses du serveur pouvant être utilisées pour se connecter à l'instance Django**
DB_NAME	**oui**	**Nom de la base de données**
DB_USER	**oui**	**Utilisateur de la base de données**
DB_PASSWORD	**oui**	**Mot de passe de la base de données**
API_GOOGLE_KEY	**oui**	**Clef d'api de google map**
SENTRY_SDK	**oui**	**Dns de monitoring de Sentry**

Définir les variables d'environnement nécessaires en modifiant le fichier .env sur le serveur (en utilisant nano)

4.3.3 - Configuration

Voici les différents fichiers de configuration :

- **requirements.txt:** fichier de téléchargement des dépendances.
- **settings.py** : fichier de configuration de l'application…
- **migrate.py :** fichier de migration de la base de données
- .**env** : fichier contenant les variables d'environnement à importer dans le fichier settings.py

4.3.3.1 - Fichier requirements.txt

Il faut lancer le fichier en ligne de commande dans le terminal grâce à la commande :

```
# pip install -r requirements.txt
```

4.3.3.2 – Fichier migrate.py

```
# projectOCpizza/manage.py migrate
```

4.3.3.3 – Fichier .env

A modifier avec nano pour intégrer les variables environnement.

4.4 – Configuration de Gunicorn

Lancer Gunicorn avec la commande suivante :

```
# gunicorn ocpizza.wsgi
```

4.4.1 – Création de fichier de socket et de service systemd

```
# sudo nano /etc/systemd/system/gunicorn.socket
```

```
[Unit]
Description=gunicorn socket

[Socket]
ListenStream=/run/gunicorn.sock

[Install]
WantedBy=sockets.target
```

```
# sudo nano /etc/systemd/system/gunicorn.service
```

```
[Unit]
Description=gunicorn daemon
Requires=gunicorn.socket
After=network.target

[Service]
User=ocpizzauser
Group=www-data
WorkingDirectory=/home/ocpizzauser/ projectOCpizza
ExecStart=/home/ocpizzauser/projectOCpizza/.venv/bin/gunicorn \
          --access-logfile - \
          --workers 3 \
          --bind unix:/run/gunicorn.sock \
          ocpizza.wsgi:application

[Install]
WantedBy=multi-user.target
```

```
# sudo systemctl start gunicorn.socket
# sudo systemctl enable gunicorn.socket
```

4.5 - Configuration de Nginx

```
sudo nano /etc/nginx/sites-available/projectOCpizza
```

```
server {
    listen 80;
    server_name server_domain_or_IP;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/ocpizzauser/projectOCpizza;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

```
sudo ln -s /etc/nginx/sites-available/myproject /etc/nginx/sites-enabled
sudo systemctl restart nginx
```

5-PROCÉDURE DE DÉMARRAGE / ARRÊT

5.1 - Base de données

Pour lancer la base de données, utiliser la commande suivante

```
# sudo systemctl start postgresql
# sudo systemctl enable postgresql
```

Pour arrêter la base de données, utiliser la commande suivante

```
# sudo systemctl stop postgresql
```

5.3 - Application web

Pour démarrer l'application, lancer les commandes suivantes :

```
# sudo systemctl start gunicorn
# sudo systemctl enable gunicorn
# sudo systemctl start nginx
# sudo systemctl enable nginx
```

Pour arrêter l'application, utiliser la commande suivante :

```
# sudo systemctl stop gunicorn
```

6-PROCÉDURE DE MISE À JOUR

6.1 - Base de données

```
# sudo apt-get update
#sudo apt-get install  postgresql postgresql-contrib
```

6.3 - Application web

```
# sudo systemctl daemon-reload
# sudo systemctl restart gunicorn.socket gunicorn.service
```

7-SUPERVISION/MONITORING

7.1 - Supervision de l’application web

Afin de tester que l’application web est toujours fonctionnelle, il faudra utiliser Newrellic.

Commencer par vous créer un compte sur https://newrelic.com/fr

Puis suivre la procédure d’installation de l’agent python (https://docs.newrelic.com/docs/apm/agents/python-agent/installation/standard-python-agent-install/)

Pour le suivi des exceptions et des logs, il faudra installer Sentry.

Après la création d’un compte sur le site : https://sentry.io/

Commencer par installer la librairie sentry-sdk avec la commande suivante

```
pip install --upgrade sentry-sdk
```

Et modifier la variable environnement SENTRY_SDK dans le fichier .env

Puis relancer l’application afin que les changements soient pris en compte

```
# sudo systemctl daemon-reload
# sudo systemctl restart gunicorn.socket gunicorn.service
```

8 - GLOSSAIRE

Agent	**est un logiciel qui agit de façon autonome. C'est un programme qui accomplit des tâches à la manière d'un automate et en fonction de ce que lui a demandé son auteur.**
Artefacts	**désigne une entité utilisée ou produite pendant le cycle de développement d'un logiciel (code source, base de données, etc.)**
Log	**Fichier dont la mission principale consiste à stocker un historique des événements**
Newrelic	***Gestion des performances des applications SaaS pour les applications Ruby, PHP, .Net, Java, Python et Node.js*** **. New Relic est l'outil de performances d'applications Web tout-en-un qui vous permet de voir les performances de l'expérience de l'utilisateur final, via les serveurs, et jusqu'à la ligne de code de l'application**
Sentry	**Sentry est une plate-forme open source pour la productivité du flux de travail, agrégeant les erreurs de l'ensemble de la pile en temps réel.**