

OC PIZZA

CENTRALISER LE SYSTEME INFORMATIQUE POUR L'ENSEMBLE DES RESTAURANTS

avec
OC SOFT

Dossier d'exploitation

Version 1.0

Auteur
Jean-Maxime GUTH
Développeur

TABLE DES MATIERES

1 -Versions	3
2 -Glossaire	4
3 -Introduction	5
3.1 -Objet du document	5
3.2 -Références.....	5
4 -Prérequis	6
4.1 -Système	6
4.1.1 -Serveur de base de données	6
4.1.2 -Serveur web	6
4.1.3 -Serveur d'application	6
4.2 -Web services	7
5 -Procédure de déploiement.....	8
5.1 -Déploiement de l'application.....	8
5.1.1 -Environnement de l'application.....	8
5.1.2 -Installation des services	8
5.2 -Configuration de l'application.....	9
5.2.1 -Fichiers de configurations	9
5.2.2 -Variables d'environnement	9
5.3 -Déploiement de la base de données	10
5.3.1 -Prérequis	10
5.3.2 -Commande pour créer la base de données.....	11
6 -Procédure de démarrage et arrêt.....	12
6.1 -Serveur d'application GUNICORN	12
6.2 -Serveur Web NGINX.....	12
6.3 -Base de données POSTGRESQL.....	13
7 -Procédure de mise à jour	14
7.1 -Serveur d'application.....	14
7.2 -Base de données	14
8 -Supervision Monitoring.....	15
8.1 -DigitalOcean pour la charge du serveur	15
8.2 -Sentry pour les logs et erreurs	15
9 -Procédure de sauvegarde et de restauration.....	16

1 - VERSIONS

Auteur	Date	Description	Version
JM GUTH	16/12/2021	Création du document	1.0

2 - GLOSSAIRE

Tâche CRON	Cron est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiée à l'avance, ou selon un cycle défini à l'avance.
-------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3 - INTRODUCTION

3.1 - Objet du document

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

Objectif du document :

L'objectif de ce document est de détailler les différentes caractéristiques du système nécessaire à l'exécution de l'application.

Il décrit également les procédures de déploiements et commandes de l'application ainsi que les outils utilisés pour l'exploitation de l'application.

3.2 - Références

Ce document découle du précédent document « **dossier de conception fonctionnelle OCSOFT V1.0** » validé par le client « **OC PIZZA** » le 22/12/2020 et remis à jour le 23/12/2021 (V2)

Il a été complété par :

- Le **dossier de conception technique** crée le 16/03/2021 (V1) et remis à jour le 23/12/2021 (V2)
- Le **PV de livraison** établi le 31/12/2021

4 - PRE-REQUIS

4.1 - Système

L'ensemble de l'application web sera hébergée sur un serveur de type IAAS (Infrastructure As A Service) fournit par DigitalOcean ayant les caractéristiques techniques suivantes :

Caractéristiques	Option souscrite
OS	Linux / Ubuntu version 20.04
Mémoire RAM	32 GB
SSD	600 GB
Nombre de CPU	8

Prix mensuel de l'offre souscrite : **250 EUR / mois**

Adresse IP : **127.666.999.696**

4.1.1 Serveur de base de données :

Le serveur de base de données utilisera **PostgreSQL**, les caractéristiques sont les suivantes :

- Version 12 et supérieure
- Une base de données db_ocpizza

4.1.2 Serveur web :

Le serveur de base de données utilisera **Nginx**, les caractéristiques sont les suivantes :

- Version 1.2 et supérieure

Le firewall sera configuré pour autoriser Nginx en connexions entrantes et sortantes.

3.1.3 - Serveur d'application

Le serveur d'application utilisé sera **Gunicorn**. Il sera surveillé par l'outil **Supervisor** afin d'assurer le démarrage et le redémarrage automatique de l'application. Un fichier de configuration de l'outil Supervisor sera créé dans le répertoire suivant :
/etc/supervisor/conf.d/ocpizza-gunicorn.conf.



- Version 20.1.0 et supérieur

4.2 - Web-services

Les web services suivants seront utilisés :

Pour les transactions financières:

- **Stripe**
- **Paypal**

5 - PROCEDURE DE DEPLOIEMENT

5.1 - Déploiement de l'application

5.1.1 - Environnement de l'application

Répertoire de projet : **ocsoft**

Utilisateur : **admin**

Se connecter au serveur en SSH :

```
$ ssh admin_ocp@127.666.999.696
```

Changer de répertoire et se placer dans le répertoire du projet :

```
$ cd /home/ocsoft
```

Cloner le projet du Github de it-consulting :

```
$ git clone https://github.com/it-consult/Oc_pizza
```

Activer l'environnement virtuel pipenv:

```
$ pipenv shell
```

Installer les paquets python :

```
$ pipenv install
```

Collecter les fichiers statiques :

```
$ python3 manage.py collectstatic
```

5.1.2 - Installation des services

Installation Nginx :

```
$ sudo apt-get install nginx
```

Installation Supervisor:

```
$ sudo apt-get install supervisor
```

5.2 - Configuration de l'application

5.2.1 - Fichiers de configuration

Chemin du fichier	Rôle
home/ocsoft/oc_pizza/settings/__init__.py	Configuration de l'application DJANGO
home/ocsoft/oc_pizza/settings/production.py	Configuration des spécificités de production (Crée sur le serveur uniquement, ignoré par Github)
/etc/nginx/sites-available/oc_pizza/	Fichier de configuration NGINX
/etc/supervisor/conf.d/P8_pur_beurre-unicorn.conf	Fichier de configuration SUPERVISOR

5.2.2 - Variables d'environnement

Les variables d'environnement sont stocké dans un fichier .env à la racine du projet.

Variable	Obligatoire
SECRET_KEY	Oui
DB_PASSWORD	Oui

5.3 - Déploiement de la base de données

5.3.1 – Prérequis

Conditions :

- Environnement virtuel activé (commande : « pipenv shell »)
- Directives à jours dans Supervisor pour l'environnement de production. (Point A)
- Configuration ok dans production.py (Point B)

A) Vérifier cette ligne dans le fichier de configuration de SUPERVISOR qu'elle correspond bien à ceci :

```
environment=DJANGO_SETTINGS_MODULE='oc_pizza.settings.production'
```

B) Vérifier production.py correspond à ceci:

```
from . import *

SECRET_KEY = os.environ.get('SECRET_KEY'),
DEBUG = False
ALLOWED_HOSTS = ['178.62.117.192']

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'db_ocpizza',
        'USER': 'admin',
        'PASSWORD': os.environ.get('DB_PASSWORD'),
        'HOST': '',
        'PORT': '5432',
    }
}
```

5.3.1 – Commandes pour créer la base de données.

Etape 1) Créer les tables de la base de données :

```
$ python3 manage.py makemigrations  
$ python3 manage.py migrate
```

Etape 2) Remplir la base de données :

```
$ python3 manage.py popdatabase
```

Note :

La commande « popdatabase » est une commande créée par It-consulting et n'est pas une commande DJANGO standard. Elle exécute un script se trouvant dans « home/ocsoft/oc_pizza/management/commands/popdatabase » et se charge d'insérer les données dans les tables de la base de données.

Les données ont été fournies par le client OC PIZZA en format CSV.

6 - PROCEDURE DE DEMARRAGE /ARRET

6.1 - Serveur d'application GUNICORN

Supervisor gère de manière automatique le démarrage de Unicorn.

Commandes manuelles (si nécessaire) :

Démarrer le serveur :

```
$ sudo supervisorctl start ocpizza-gunicorn
```

Vérifier le statut du processus :

```
$ sudo supervisorctl status ocpizza-gunicorn
```

Arrêter le serveur:

```
$ sudo supervisorctl stop ocpizza-gunicorn
```

Mise à jour par suite d'une modification du fichier de configuration :

```
$ sudo supervisorctl reread
```

```
$ sudo supervisorctl update
```

6.2 - Serveur Web NGINX

Démarrer le serveur :

```
$ sudo service nginx start
```

Vérifier le statut du processus :

```
$ sudo service nginx status
```

Arrêter le serveur :

```
$ sudo service nginx stop
```

Mise à jour par suite d'une modification du fichier de configuration :

```
$ sudo service nginx reload
```

6.3 - Base de données POSTGRESQL

Démarrer PostgreSQL

```
$ sudo systemctl start postgresql
```

Arrêter PostgreSQL

```
$ sudo systemctl start postgresql
```

7 - PROCEDURE DE MISE A JOUR

7.1 - Serveur d'application

Mise à jour des fichiers source de l'application depuis le dépôt GitHub :

```
$ git pull origin main
```

Mise à jour des fichiers statiques :

```
$ python3 manage.py collectstatic
```

Consulter les services pouvant être mis à jour :

```
$ sudo apt-get update
```

Mettre à jour les paquets :

```
$ sudo apt-get upgrade
```

7.2 - Base de données

Mise à jour de la base de données :

```
$ python3 manage.py migrate
```

8 - SUPERVISION / MONITORING

8.1 - DigitalOcean pour la charge du serveur

Les outils de supervision du serveur se font depuis l'espace client de DigitalOcean dans la partie Monitoring du dashboard :

Update resource alert

Select metric & set threshold

CPU	is above	70,00	%	30 min
-----	----------	-------	---	--------

Select Droplets or Tags

Alerting requires installation of the DigitalOcean metrics agent on your Droplets. When selecting all Droplets or tags, make sure the Droplets have the agent installed. [Get instructions](#) on how to install the metrics agent. The agent is pre-installed on Kubernetes worker nodes. To maintain resource alerts with worker node recycling, tags are recommended over node names.

P10-PurBeurre-deploy	Please type a Droplet or Tag Name
----------------------	-----------------------------------

Nous avons paramétré les alertes suivantes :

- Charge CPU > 70%

Pour vous connecter veuillez utiliser les identifiants et lien fournis avec le PV de livraison

8.2 - Sentry pour les logs et erreurs

La surveillance des logs et des erreurs est assurée par Sentry
Le tableau de bord Sentry est accessible via <https://sentry.io>

Pour vous connecter veuillez utiliser les identifiants et lien fournis avec le PV de livraison

Sentry est installé dans les librairies de l'application Django.

La commande de l'installation est :

```
$ pipenv install --upgrade sentry_sdk
```

La configuration de Sentry se fait dans production.py :

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

sentry_sdk.init(
    dsn="https://699854613165446126316484651264@o0.ingest.sentry.io/0",
    integrations=[DjangoIntegration()],
    traces_sample_rate=0.3,
    send_default_pii=True,
)
```

Des loggers ont été paramétrés dans l'ensemble du code de l'application pour recevoir différents logs. (Logs applicatifs, logs systèmes, logs de base de données, logs de trafic HTTP(s))

Ces logs permettront de :

- Voir et surveiller les événements des utilisateurs.
- Recevoir des informations sur les différentes erreurs et les fonctionnalités concernées

Exemple de log pour une recherche dans une vue de recherche :

```
Import logging
logging.info('New search', exc_info=True, extra={
    'request': request,
})
```

D'autres types de logs peuvent être paramétrés et sont visibles dans le code, on retrouvera notamment :

```
import logging
logging.debug("args")
logging.info("args")
logging.error("args")
logging.exception("args")
```

Pour ajouter des nouveaux logs, veuillez nous contacter.

9 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

Une tâche CRON effectue une sauvegarde journalière de la base de données tous les jours à 02:00 AM

Les sauvegardes se trouvent dans le dossier :
home/ocsoft/oc_pizza/backups/db_ocpizza

Restauration de la base de données :

```
$ pg_restore -d ocpizza_db /home/ocsoft/oc_pizza/backup/oc_pizza_db.bak
```

NOTES ET REMARQUES

Sujet	Remarques / observations	Page	Auteur

Sujet	Remarques / observations	Page	Auteur