OC PIZZA

Boutique en ligne
&
Gestionnaire centralisé

Dossier d'exploitation

Version 1.0

Auteur
Stephen AOGOLO
Analyste-programmeur

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Stephen AOGOLO	14/01/2021	Création du document	1.0

2 - INTRODUCTION

2.1 - Objet du document

Ce dossier d'exploitation (***version 1.0***) fait suite au dossier de spécifications technique (***version 2.0***).

L'objectif de ce dossier d'exploitation se partage en deux parties.

Dans un premier temps, ce dossier reprend et décrit techniquement le contenu vital du système. Par la suite il renseigne sur les procédures nominales d'installation, d'utilisation et de maintenance du système. Ce document présente et décrit donc le déploiement du produit **OC PIZZA – Boutique en ligne & Gestionnaire centralisé**.

Les points suivants seront donc traités afin de garantir le meilleur niveau d'information quant aux manipulations post-production :

- La description technique des sous-systèmes
- La procédure nominale du déploiement
- La procédure nominale de mise en route et d'extinction
- La procédure nominale de mise à jour
- Les actions et méthodes nominales de supervision
- Les actions et méthodes nominales de sauvegarde
- Les actions et méthodes nominales de restauration

*Il est important de noter que la mention « **nominale** » fait référence à la manière standard et recommandé d'effectuer ces opérations. Dans une version future du système et donc un futur dossier d'exploitation, les procédures, les actions et les méthodes « **d'urgences** » seront disponibles afin d'intervenir dans des cas particuliers de dépannage du système.*

2.2 - Références

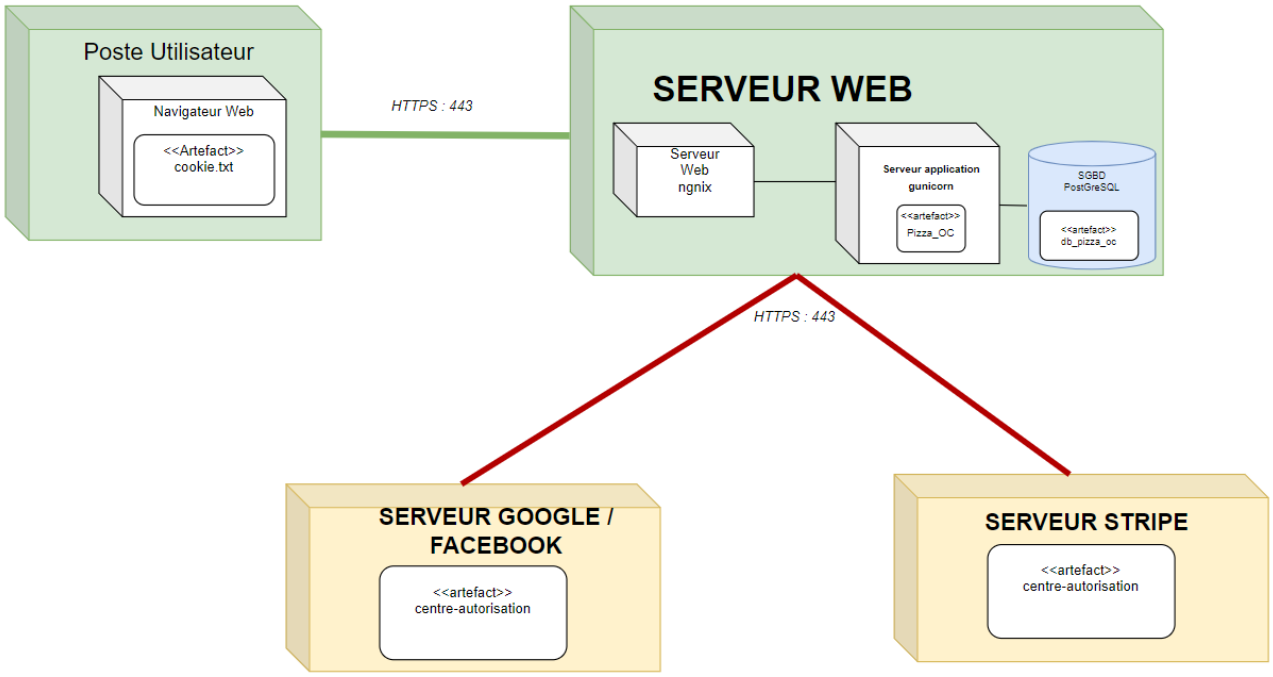
Pour de plus amples informations, merci de se référer aux documents suivants :

REFERENCES	VERSIONS
Dossier de conception fonctionnelle	1.4
Dossier de conception technique	2.0
Procès-verbal de livraison	1.0

3 - PREREQUIS

3.1 - Description du système

L'ensemble des composants du système seront hébergés chez DIGITAL OCEAN. Ils occuperont un emplacement virtuel. Ci-dessous, le diagramme de déploiement, extrait du dossier de conception.



Ce diagramme reprend donc l'ensemble des composants principaux du système à déployer.

PS : L'artefact « cookie.txt » sera installé automatiquement après la première connexion de l'utilisateur. Il n'est donc pas concerné par les procédures de déploiement, de démarrage et de maintenance.

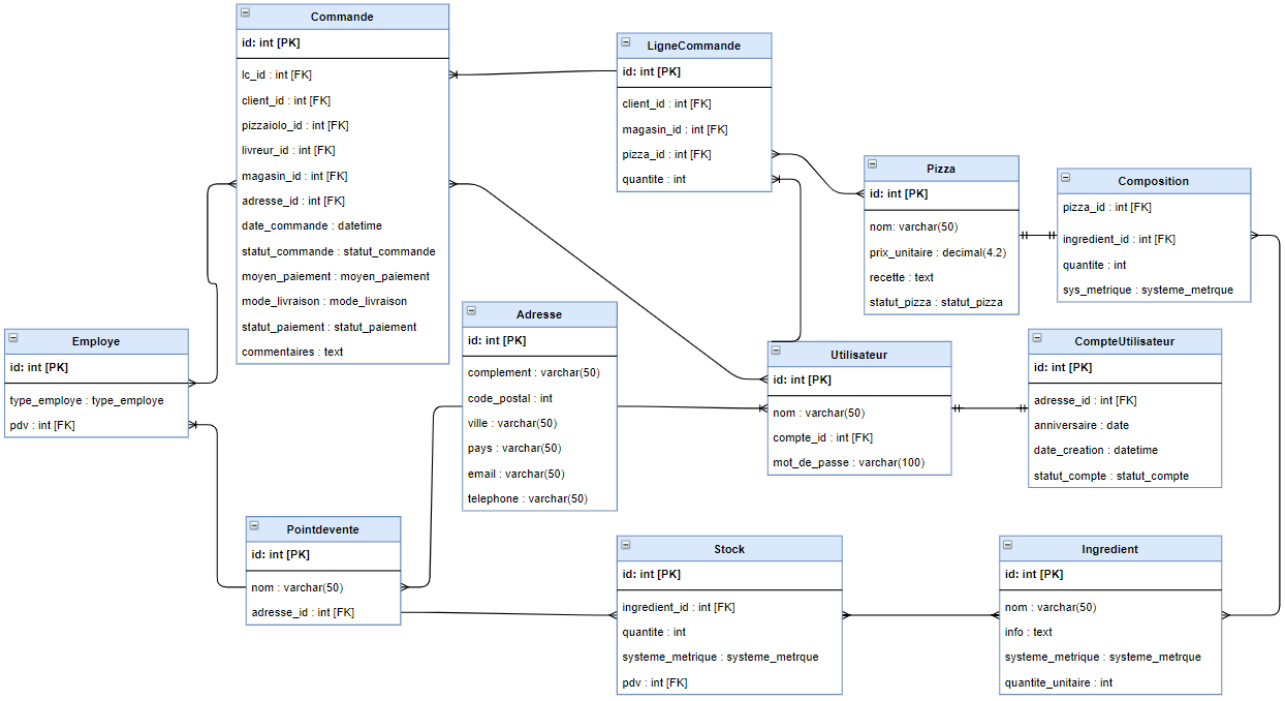
3.2 - Serveur de Base de données

Le serveur de base de données utilisé est de type **Postgresql**.

Le tableau ci-dessous reprend la description technique du serveur de base de données.

SERVEUR BASE DE DONNEES		
NOM	*VERSION*	*DESCRIPTION*
PostgreSQL	13	PostgreSQL est un système de gestion de bases de données relationnelles objet fondé sur POSTGRES, Version 4.2. Il supporte une grande partie du standard SQL tout en offrant de nombreuses fonctionnalités modernes.

Pour rappel, la base de données du site web est mise en forme selon le modèle physique de données suivant :



Pour de plus amples informations liées aux différents composants de la base de données, veuillez vous référer au ***dossier de conception technique***, à la partie ***3.2 - diagramme de classe***.

3.3 - Serveur Web

Le serveur web est de type **NGNIX**. Il est le support du serveur d'application web de type **Gunicorn**.

Le tableau ci-dessous reprend la description technique du serveur web.

SERVEUR WEB		
NOM	*VERSION*	*DESCRIPTION*
Ngnix	1.18.0	Ce serveur traite les demandes du client WEB avant de les transmettre au WSGI. Il est aussi sollicité dans le sens inverse, ce qui lui confère un rôle de passerelle entre les terminaux WEB et le serveur d'applications. NGNIX possède une bonne relation avec GUNICORN en termes de déploiement et de maintenance.

Le tableau ci-dessous reprend la description technique du serveur d'application web.

SERVEUR D'APPLICATION		
NOM	*VERSION*	*DESCRIPTION*
Gunicorn	20.0.4	Le WSGI assure la mise en forme du flux HTTP, provenant du serveur WEB, en un flux adapté à la solution applicative. En plus de sa fonction de passerelle, ce serveur fourni des services standards pour améliorer le développement, le fonctionnement et la maintenance de la solution. Il améliore également la portabilité de la solution applicative.

3.4 - Web-services

Les web-services sont des composants externes à la solution qui apportent des fonctionnalités supplémentaires au système. Elles sont implémentées via leur API respective.

Pour de plus d'informations sur le rôle et la mise en place des web-services de la solution, consultez le ***dossier de conception technique*** *à la partie* ***Règles de gestion Architecture Technique****.*

Le système utilise trois web-services classés en deux groupes :

- Un web-service monétique : Stripe
- Deux web-services d'authentification : Google Sign-in et Facebook Login

Le tableau ci-dessous reprend la description technique du web-service monétique.

WEB-SERVICE MONETIQUE		
NOM	*VERSION*	*DESCRIPTION*
Stripe	2020.08.27	L'API STRIPE permet d'intégrer l'accès aux fonctionnalités transactionelles de la plateforme.

Le tableau ci-dessous reprend la description technique des web-services d'authentification.

WEB-SERVICES D'AUTHENTIFICATION		
NOM	*VERSION*	*DESCRIPTION*
Google Sign-In	Oauth 2.0	L'API GOOGLE SIGN-IN permet d'intégrer les mécanismes d'authentification des utilisateurs ayant un compte GOOGLE.
Facebook Login	2018.07.02	L'API FACEBOOK LOGIN permet d'intégrer les mécanismes d'authentification des utilisateurs ayant un compte FACEBOOK.

4 - PROCEDURE DE DEPLOIEMENT

La procédure de déploiement suivante doit être respectée, afin de garantir le bon déploiement du site web **OC PIZZA**.

L'étape initiale

Avant de débuter le déploiement, il faut tout d'abord se rendre sur la machine d'hébergement, préalablement choisie. Dans le cas de ce dossier d'exploitation, une machine d'hébergement publique DIGITAL OCEAN. Il est donc essentiel de détenir l'adresse IP publique de la machine d'hébergement afin de s'y connecter et d'y effectuer l'ensemble des procédures à venir. Pour plus d'information sur cette étape initiale, merci de consulter ***https://www.digitalocean.com/docs/getting-started/*** .

Pour se rendre sur la machine d'hébergement, il faut y accéder via une connexion SSH en ligne de commandes, comme suit :

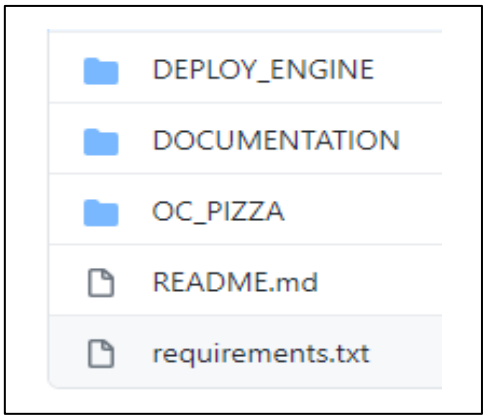
```
>ssh root@IPpublicaddress
```

Les acteurs du déploiement

A l'état initial, l'ensemble de la solution se trouve sur GITHUB à l'adresse suivante : ***https://github.com/StephenAOGOLO/P9_Documentez_votre_systeme_de_gestion_de_pizzeria.git*** .

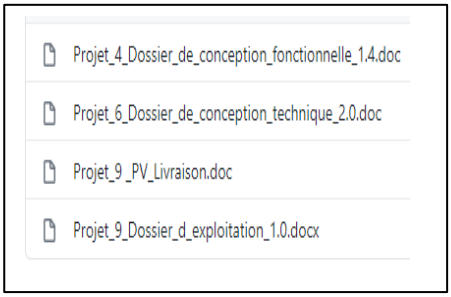
A cette adresse se trouve les éléments suivants :

SOLUTION

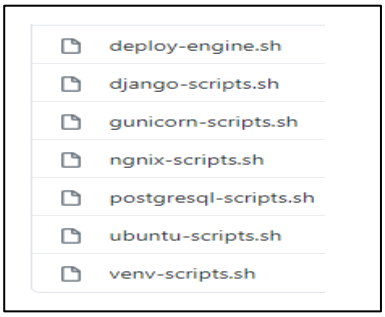


Le répertoire « DEPLOY_ENGINE » contient les scripts de déploiement de la solution.
Le répertoire « DOCUMENTATION » contient l'ensemble des dossiers liés à la conception du projet OC PIZZA.
Le répertoire « OC_PIZZA » est le projet DJANGO de la solution, soit, le site web à déployer.
Le fichier « README.md » reprend la description générale du projet et conseille sur son mode utilisation.
Le fichier « requirements.txt » est un fichier utilisé par les scripts « DEPLOY_ENGINE ».

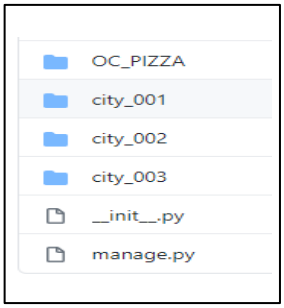
DOCUMENTATION



DEPLOY ENGINE



OC PIZZA



Les étapes de déploiement

Le déploiement du site web est entièrement géré par les éléments du répertoire « DEPLOY_ENGINE ».

Le site web, de type DJANGO, sera installé sur une machine publique de type UBUNTU. Des composants UNIX-UBUNTU seront donc installés pour prendre en charge et servir les applications DJANGO.

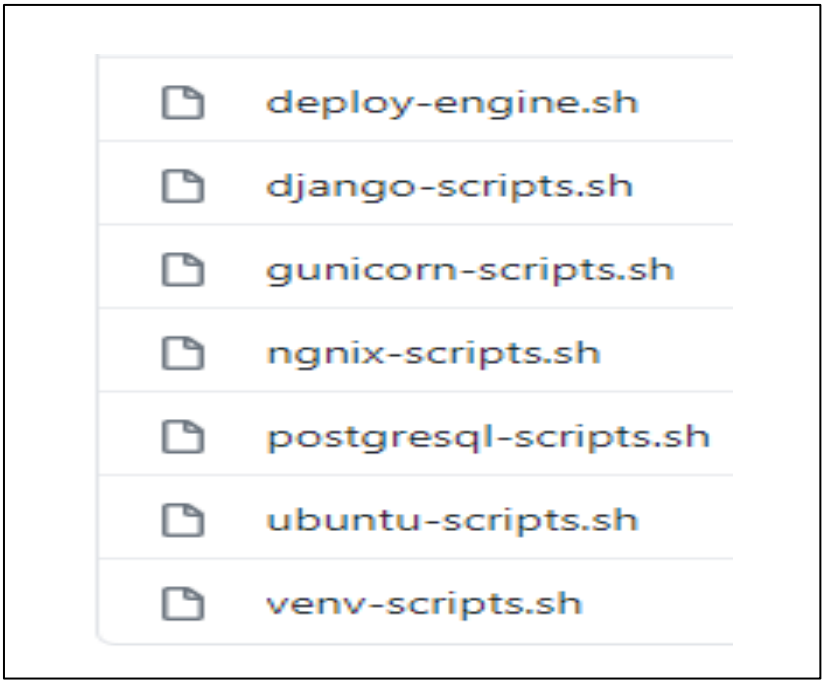
La base de données POSTGRESQL sera configurée afin de gérer les données applicatives.

Le serveur d'application GUNICORN sera installé et configuré pour s'interfacer avec les applications du site.

Enfin, le serveur web NGNIX sera installé et configuré pour gérer la communication entre le service GUNICORN et internet, et ainsi, profiter des fonctionnalités de sécurité et de performance.

Ci-dessous, le contenu du répertoire « DEPLOY_ENGINE ».

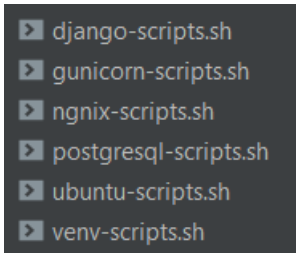
DEPLOY_ENGINE



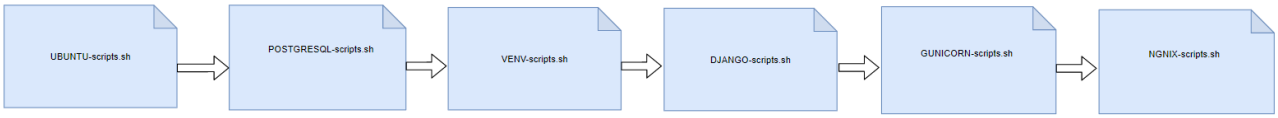
4.1 - Descriptions des artefacts

Le déploiement du site web **OC PIZZA** est semi-automatique. Il s'appuie sur l'exécution de plusieurs scripts partiellement rédigés nécessitant des précisions complémentaires.

Il y a six scripts à configurer avant qu'elles ne soient utilisées pour le déploiement semi-automatique du site :

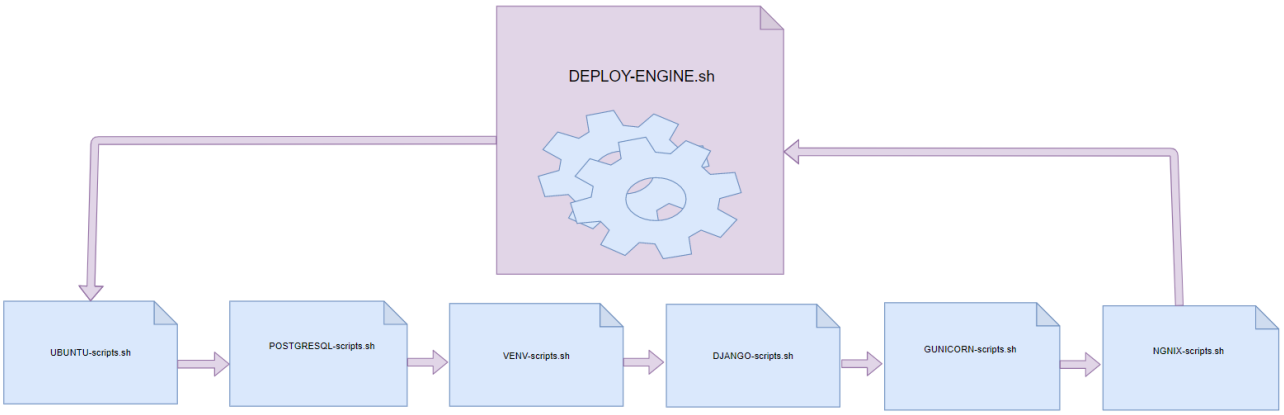


Ces artefacts seront exécutés dans l'ordre précis illustré ci-dessous :



Chaque artefact est une étape de déploiement. Chacun contient les opérations d'installation, de configuration et de lancement. Avant l'exécution de ces scripts, il est important de les édités afin de corriger et confirmer leur contenu.

L'exécution des six scripts cités précédemment, est gérée par un gestionnaire de déploiement « deploy-engine.sh ». Il assure le séquencement de l'installation générale des étapes de déploiement.



4.2 - Configuration des scripts

Sur votre machine locale, récupérez l'ensemble de la solution actuellement disponible sur GITHUB :

```
>git init
>git remote add origin https://github.com/StephenAOGOLO/P9_Documentez_votre_systeme_de_gestion_de_pizzeria.git
```

Editez les scripts suivants afin d'y ajouter vos informations personnalisées :

4.2.1.1 - Script ubuntu-scripts.sh

```
# Ubuntu Configuration

# Adding an Ubuntu user
adduser ubuntu_user

# Updating the Ubuntu user rules
usermod -aG sudo ubuntu_user

# Adding all the project needed packages
sudo apt update
sudo apt install python3-pip python3-dev libpq-dev postgresql postgresql-contrib nginx curl
```

- Ajout d'un utilisateur *ubuntu*.
- Ajout de l'utilisateur au groupe administrateur.
- Mise à jour du système de paquets *ubuntu*.
- Installation des paquets relatifs à *python, postgresql, ngnix et système de requêtage web*.

4.2.1.2 - Script postgresql-scripts.sh

```
# Database configuration

# Opening the POSTGRESQL console session
sudo -u postgres psql

# Creating the main website database
CREATE DATABASE db_oc_pizza;

# Creating a database user
CREATE USER the_user WITH PASSWORD 'password';

# Configuring the database user rules
ALTER ROLE the_user SET client_encoding TO 'utf8';
ALTER ROLE the_user SET default_transaction_isolation TO 'read committed';
ALTER ROLE the_user SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE db_oc_pizza TO the_user;

# Quitting console session
\q
```

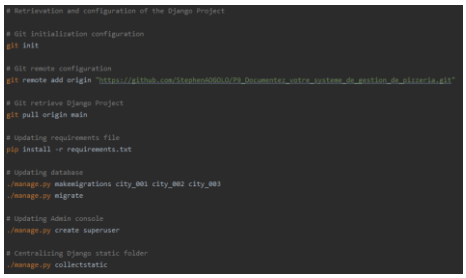
- Ouverture de la console *postgres*.
- Création de la base de données *db_oc_pizza*.
- Création d'un utilisateur.
- Paramétrage de l'encodage, d'isolation des transactions et du fuseau horaire lié à l'utilisateur.
- Fermeture de la console *postgres*.

4.2.1.3 - Script venv-scripts.sh

```
# Creation of Virtual Environment

# Initial step - Switch to Ubuntu User
sudo ubuntu_user

# Installation of pip and virtualenv
sudo -H pip3 install --upgrade pip
sudo -H pip3 install virtualenv

# Creation and Activation of a virtual environment
mkdir ~/OC_PIZZA
cd ~/OC_PIZZA
virtualenv oc_pizza_venv
source oc_pizza_venv/bin/activate
```

- Changement d'utilisateur *ubuntu*.
- Mise à jour du système de paquets *pip-python (rattaché au répertoire /home)*.
- Installation du paquet *virtualenv (rattaché au répertoire /home)*.
- Création et ouverture d'un répertoire *OC_PIZZA (rattaché au répertoire personnel de l'utilisateur)*.
- Création d'un environnement virtuel.
- Activation de l'environnement virtuel.

4.2.1.4 - Script django-scripts.sh



```
# Retrievation and configuration of the Django Project

# Git initialization configuration
git init

# Git remote configuration
git remote add origin "https://github.com/StephenADOLO/P9_Documentez_votre_systeme_de_gestion_de_pizzeria.git"

# Git retrieve Django Project
git pull origin main

# Updating requirements file
pip install -r requirements.txt

# Updating database
./manage.py makemigrations city_001 city_002 city_003
./manage.py migrate

# Updating Admin console
./manage.py create superuser

# Centralizing Django static folder
./manage.py collectstatic
```

- Initialisation du gestionnaire de versions *git*.
- Affectation du projet distant à la branche *origin*.
- Récupération du projet distant
- Installation des paquets présents dans *requirements.txt*
- Migration de la base de données du site web.
- Création d'un compte administrateur lié au site web.
- Centralisation des ressources statiques du site web.

4.2.1.5 - Script gunicorn-scripts.sh

```
# Configuring Gunicorn
gunicorn --bind 0.0.0.0:8000 OC_PIZZA.wsgi

# Deactivating the Virtual Environment
deactivate

# Creating and Editing Gunicorn service file
sudo cp /conf_files/gunicorn.service /etc/systemd/system/gunicorn.service

# Starting Gunicorn service
sudo systemctl start gunicorn

# Enabling Gunicorn service
sudo systemctl enable gunicorn

# Checking Gunicorn service status
sudo systemctl status gunicorn

# Checking Gunicorn service logs
sudo journalctl -u gunicorn
```

- Ouverture de l'accès du site web.
- Désactivation de l'environnement virtuel.
- Configuration du service *gunicorn*.
- Démarrage et activation du service *gunicorn*.
- Consultation de l'état du service gunicorn.

4.2.1.6 - Script ngnix-scripts.sh

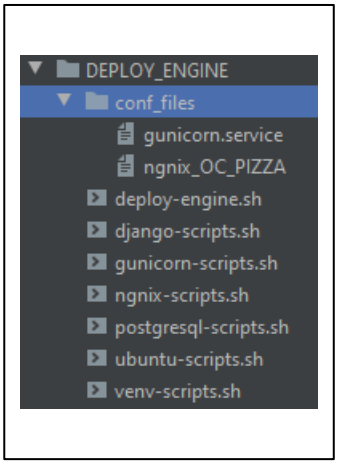
```
# Configuring Ngnix

# Creating and Editing Ngnix configuration file
sudo cp /conf_files/ngnix_OC_PIZZA /etc/nginx/sites-available/OC_PIZZA

# Enabling the nginx site
sudo ln -s /etc/nginx/sites-available/OC_PIZZA /etc/nginx/sites-enabled

# Testing Ngnix configuration
sudo nginx -t

# Restarting Ngnix Service
sudo systemctl restart nginx

# Opening full access to the website
sudo ufw allow 'Nginx Full'
```

- Configuration et test du service *ngnix*.
- Redémarrage du service *ngnix*.
- Ouverture de l'accès publique du site web.

4.3 - Configuration des fichiers

Dans le répertoire « DEPLOY_ENGINE », se trouve le second répertoire « conf_files ».

Il contient les deux fichiers de configuration suivants :



gunicorn.service
- Fichier de configuration du service GUNICORN

ngnix_OC_PIZZA
- Fichier de configuration du service NGNIX

Ces fichiers doivent également être modifiés avant l'exécution du gestionnaire de déploiement.

4.3.1.1 - Fichier gunicorn.service

Ce fichier contient les informations suivantes :

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

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

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

- La section *Unit* permet de paramétrer les données d'initialisation du service.
- La section *Service* permet de paramétrer les données relatives à l'utilisateur et aux emplacements physiques et stratégiques du site web.
- La section *Install* permet d'activer le lancement du service lors du démarrage du service *multi-user*.

4.3.1.2 - Fichier ngnix_OC_PIZZA

Ce fichier contient les informations suivantes :

```
server {
    listen 80;
    server_name OC PIZZA;

    location = /static/oc_pizza.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/ubuntu_user/OC_PIZZA;
    }

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

- Ouverture du port *80*.
- Affectation du nom de domaine (*Adresse IP le cas échéant*).
- Paramétrage de redirection du flux web vers *guinicorn*.

4.4 - Exécution du déploiement

La configuration des scripts est à présent terminée. Le lancement du déploiement peut alors débuter.

Lancez le gestionnaire de déploiement :

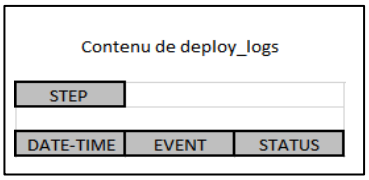
```
>source DEPLOY_ENGINE/deploy-engine.sh
```

4.5 - Vérifications du déploiement

À la suite de l'exécution du gestionnaire de déploiement, le fichier « deploy_logs » sera automatiquement généré. Ce fichier informe sur les sous-étapes de déploiement, autrement dit, les évènements principaux des six étapes de déploiement.

Ci-dessous, le fichier généré après le bon déploiement du site.

```
# Step 1 : UBUNTU
yyyy-mm-dd_hh-mm-ss-mmm : Ubuntu-user-creation > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Packages-installation > CORRECT

# Step 2 : POSTGRESQL DataBase
yyyy-mm-dd_hh-mm-ss-mmm : Database-creation > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Psql-user-creation > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : User-association > CORRECT

# Step 3 : VIRTUAL ENVIRONMENT
yyyy-mm-dd_hh-mm-ss-mmm : Virtual-environment-installation > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Virtual-environment-activation > CORRECT

# Step 4 : DJANGO PROJECT
yyyy-mm-dd_hh-mm-ss-mmm : Project-downloading > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Database-upgrading > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Ressources-access-formatting > CORRECT

# Step 5 : GUNICORN
yyyy-mm-dd_hh-mm-ss-mmm : Gunicorn-service-file-configuration > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Gunicorn-service-activation > CORRECT

# Step 6 : NGNIX
yyyy-mm-dd_hh-mm-ss-mmm : Ngnix-service-file-configuration > CORRECT
yyyy-mm-dd_hh-mm-ss-mmm : Ngnix-service-activation > CORRECT

# Final step : Launch the website
yyyy-mm-dd_hh-mm-ss-mmm : Website-connection > CORRECT
```



STEP	Etape de déploiement	Renseigne sur le nom de l'étape de déploiement.
DATE-TIME	Date et heure d'exécution	Renseigne sur l'instant d'exécution d'un évènement principal. Le format est de type Année-Mois-Jour_Heure-Minutes-Secondes-Centièmes
EVENT	Evenement ou sous-étape de déploiement	Renseigne sur le nom de l'évènement principal.
STATUS	Etat final de réalisation	Renseigne sur l'état final de l'évènement principal. CORRECT : La configuration est cohérente et l'installation a été réalisée avec succès. ERROR : La configuration n'est pas cohérente et/ou l'installation n'a pas été réalisée avec succès.

5 - PROCEDURE DE DEMARRAGE / ARRET

Le déploiement du site web a été réalisé avec succès, grâce au gestionnaire de déploiement.

Dans l'exemple de ce dossier d'exploitation, le site web est disponible à l'adresse suivante :

https://www.ocpizza.fr

Ainsi, les applications web et la base de données du site, sont opérationnelles et en cours d'exécution.

Certaines opérations de maintenance nécessitent une désactivation du site web. Afin de permettre ces maintenances particulières, des interventions manuelles sont requises pour arrêter et redémarrer le site web. Ces opérations se concentre sur le processus des services GUNICORN et NGNIX.

5.1 - Désactivation du site web

Pour désactiver correctement le site web, il faut exécuter la suite de commandes ci-dessous :

```
>ssh root@IPpublicaddress
>cd ~/OC_PIZZA
>sudo systemctl stop ngnix
>sudo systemctl status ngnix
>sudo systemctl stop gunicorn
>sudo systemctl status gunicorn
```

- Connexion *ssh* vers le serveur d'hébergement.
- Ouverture du répertoire *OC_PIZZA*.
- Mise en arrêt du service *ngnix*
- Consultation de l'état du service *ngnix*.
- Mise en arrêt du service *gunicorn*.
- Consultation de l'état du service *gunicorn*.

A la suite de ces opérations, le site web s'est désactivé correctement. Des interventions de maintenance peuvent être effectuées en toute sécurité.

5.2 - Activation du site web

Pour activer correctement le site web, il faut exécuter la suite de commandes ci-dessous :

```
>ssh root@IPpublicaddress
>cd ~/OC_PIZZA
>sudo systemctl start gunicorn
>sudo systemctl status gunicorn
>sudo ngnix -t
>sudo systemctl start ngnix
>sudo systemctl status ngnix
>sudo ufw allow 'Ngnix Full'
```

- Connexion *ssh* vers le serveur d'hébergement.
- Ouverture du répertoire *OC_PIZZA*.
- Mise en marche du service *gunicorn*.
- Consultation de l'état du service *gunicorn*.
- Mise en marche du service *ngnix*.
- Consultation de l'état du service *ngnix*.
- Ouverture de l'accès publique du site web.

A la suite de ces opérations, le site web s'est activé correctement. Son utilisation peut débuter en

toute sécurité.

5.3 - Rafraichissement du site web

Dans certains cas, la réactivation ou le rechargement du site web suffisent.

5.3.1 - Réactivation du site web

```
>ssh root@IPpublicaddress
>cd ~/OC_PIZZA
>sudo systemctl restart gunicorn
>sudo systemctl restart ngnix
```

- Connexion *ssh* vers le serveur d'hébergement.
- Ouverture du répertoire *OC_PIZZA*.
- Redémarrage du service *gunicorn*.
- Redémarrage du service *ngnix*.

5.3.2 - Rechargement du site web

```
>ssh root@IPpublicaddress
>cd ~/OC_PIZZA
>sudo systemctl reload gunicorn
```

- Connexion *ssh* vers le serveur d'hébergement.
- Ouverture du répertoire *OC_PIZZA*.
- Rechargement du service *gunicorn*.

5.3.3 - Les cas de maintenances manuelles

Durant la vie d'hébergement du site web, des interventions de maintenance manuelles pourront être effectuées.

Toutes interventions manuelles nécessitent une procédure de maintenance spécifique. Dans le tableau ci-dessous, sont répertoriés les cas d'interventions manuels les plus fréquents. Ces derniers sont couplés aux procédures qui doivent être respectées pour assurer le bon fonctionnement du site.

CAS DE MAINTENANCE MANUELLE	PROCEDURE DE MAINTENANCE MANUELLE
Modification de la base de données	Désactivation du site web > Modification souhaitée > Activation du site web
Modification du code source	Désactivation du site web > Modification souhaitée > Activation du site web
Modification des templates	Désactivation du site web > Modification souhaitée > Activation du site web
Modification du fichier gunicorn.service	Modification souhaitée > Rechargement du site web
Modification du fichier ngnix_OC_PIZZA	Modification souhaitée > Réactivation du site web

6 - PROCEDURE DE MISE A JOUR

Il existe deux moyens pour mettre à jour le site web :

La mise à jour catégorique

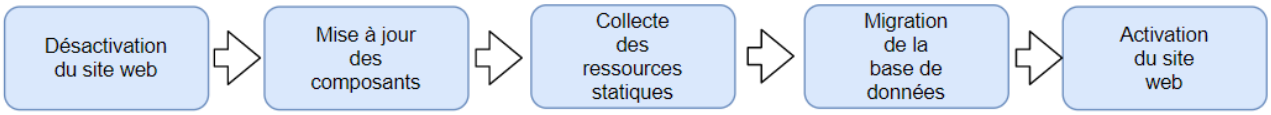
La mise à jour catégorique permet de mettre à jour la plupart des ressources du site web, tel que le code source, le contenu visuel ou encore la structure de la base de données. Cette est une opération non périodique, qui doit être préparée avant d'être exécutée. C'est donc un processus semi-automatique qui est déclenché manuellement.

La mise à jour cyclique

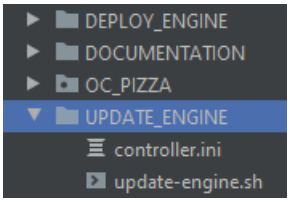
La mise à jour cyclique est un processus automatique qui démarre dès l'activation du site web. Il permet de mettre à jour périodiquement le contenu de la base de données, et non sa structure. Les données collectées durant ce processus proviennent de la base de données de production.

6.1 - Mise à jour catégorique

A l'instar de la procédure de déploiement, la procédure de mise à jour catégorique est semi-automatisée. Elle respecte des étapes de mise à jour bien précis :



Elle s'appuie sur l'exécution d'un script nommé « update-engine.sh », disponible dans le répertoire « UPDATE_ENGINE ».



6.1.1 - Description des artefacts

Le répertoire « UPDATE_ENGINE » contient donc deux artefacts :

- update-engine.sh
- controller.ini

update-engine.sh

```
# Stop Ngnix Service
sudo systemctl stop nginx

# Stop Gunicorn Service
sudo systemctl stop gunicorn

# Updating light and heavy components
./manage.py update_lhc

# Centralizing Django static folder
./manage.py collectstatic

# Updating database
./manage.py makemigrations city_001 city_002 city_003
./manage.py migrate

# Start Gunicorn Service
sudo systemctl start gunicorn

# Start Ngnix Service
sudo systemctl start nginx

# Opening full access to the website
sudo ufw allow 'Nginx Full'
```

Ce script permet de séquencer les étapes de mise à jour.

Controller.ini

```
# light_components
pizza_tokyo.png = OC_PIZZA.city_002.static.png
soda_framboise.jpg = OC_PIZZA.city_003.static.jpg
# heavy_components
historic_2.6.hmtl = OC_PIZZA.city_001.templates.city_001.historic.html
historic_2.6.py = OC_PIZZA.city_001.historic.py
models_2.6.py = OC_PIZZA.city_001.models.py
```

Ce fichier de configuration regroupe les informations d'adressage, relatives aux composants à mettre à jour.

6.1.2 - Configuration du script update-engine.sh

```
# Stop Ngnix Service
sudo systemctl stop nginx

# Stop Gunicorn Service
sudo systemctl stop gunicorn

# Updating light and heavy components
./manage.py update_lhc

# Centralizing Django static folder
./manage.py collectstatic

# Updating database
./manage.py makemigrations city_001 city_002 city_003
./manage.py migrate

# Start Gunicorn Service
sudo systemctl start gunicorn

# Start Ngnix Service
sudo systemctl start nginx

# Opening full access to the website
sudo ufw allow 'Nginx Full'
```

- Arrêt du service *ngnix*.
- Arrêt du service *guinicorn*.
- Exécution de la mise à jour.
- Centralisation des ressources statiques du site web.
- Migration de la base de données.
- Mise en marche du service *gunicorn*.
- Mise en marche du service *ngnix*.
- Ouverture de l'accès publique du site web.

6.1.3 - Configuration du fichier controller.ini

Les « light components » et les « heavy components » sont les composants ajoutés durant la mise à jour. Dans ce fichier, ces composants sont valorisés par leur adresse de destination.

```
# light_components
pizza_tokyo.png = OC_PIZZA.city_002.static.png
soda_framboise.jpg = OC_PIZZA.city_003.static.jpg
# heavy_components
historic_2.6.hmtl = OC_PIZZA.city_001.templates.city_001.historic.html
historic_2.6.py = OC_PIZZA.city_001.historic.py
models_2.6.py = OC_PIZZA.city_001.models.py
```

Les « Light components » ou composants légers définissent tous les éléments comportant les extensions ci-contre.

COMPOSANTS LEGERS
.ico
.jpg
.png
.pdf
.gif
...

Les « Heavy components » ou composants lourds définissent tous les éléments comportant les extensions ci-contre.

COMPOSANTS LOURDS
.py
.html
.css
.js
.json
.ini
.psql
...

6.1.4 - Exécution de la mise à jour catégorique

Le projet DJANGO du site web possède une procédure de mise à jour personnalisée et développée par nos soins.

6.1.5 - Vérification de la mise à jour catégorique

A la fin de la mise à jour automatique. On peut vérifier la prise en compte des nouvelles modifications.

6.2 - Mise à jour cyclique

La mise à jour cyclique est un processus intégré au site web via un module DJANGO :

- DJANGO-CRONTAB

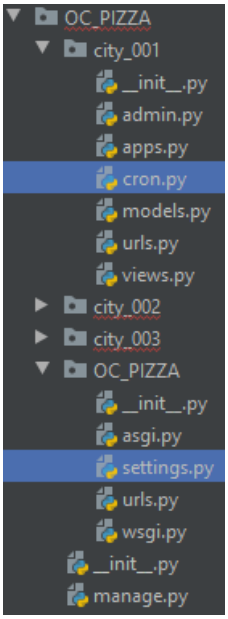
Conformément à sa documentation, ce module a été implémenté afin de s'affranchir de toutes interventions humaines, éventuellement utiles à la mise en marche du processus.

Toutefois, il est possible de régler la périodicité de cette opération.

6.2.1 - Description des artefacts

A titre d'information, l'exercice du module « DJANGO-CRONTAB » occupe deux types d'artefacts :

- **settings.py** – Module de paramétrage global du site web. Il contient notamment le réglage de périodicité.
- **cron.py** – Module dédié à l'action de mise à jour. Il est situé dans chaque application du projet DJANGO.



6.2.2 - Configuration de settings.py

Les images ci-dessous illustrent la couverture rédactionnelle du module « DAJNGO-CRONTAB ».

settings.py

```
INSTALLED_APPS = [
    'django_crontab',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dbbackup',
    'city_001',
    'city_002',
    'city_003',
]
```

```
CRONJOBS = [
    ('0 6 * * *', 'city001.cron.cyclic_update'),
    ('0 6 * * *', 'city002.cron.cyclic_update'),
    ('0 6 * * *', 'city003.cron.cyclic_update'),
```

cron.py

```
from django.core.management import call_command

def cyclic_update():
    try:
        call_command('update_ocpizza')
    except:
        pass
```

C'est au niveau du module « settings.py », dans la partie « CRONJOBS » que s'opère le réglage de la périodicité. Par défaut, la mise à jour a lieu tous les jours à 6h du matin.

6.2.3 - Réglage de la périodicité

Le réglage de la périodicité entre dans le cadre des cas de maintenances manuelles.

Avant de modifier le contenu du « CRONJOBS », assurez-vous de respecter la procédure de maintenance manuelle dédié aux modifications de code source.

CAS DE MAINTENANCE MANUELLE	PROCEDURE DE MAINTENANCE MANUELLE
Modification du code source	Désactivation du site web > Modification souhaitée > Activation du site web

Le « CRONJOBS » se compose essentiellement de deux parties. La première permet de préciser la périodicité des actions à mener. La seconde renseigne sur le type d'action à réaliser.

Dans notre cas, il existe plusieurs actions à réaliser car ce site web possèdent plusieurs applications, une action par application.

Afin de modifier uniquement la périodicité, veillez à apporter des modifications uniquement sur les premières parties du « CRONJOBS ».

« CRONJOBS » reprend les paramétrages et opérations standards du planificateur de tâches « CRON ».

Afin de parfaire le réglage de la périodicité, merci de consulter la documentation CRON UBUNTU.

6.2.4 - Vérification de la périodicité

A l'activation du site web, on peut vérifier la prise en compte des nouvelles modifications.

7 - SUPERVISION/MONITORING

Le service de surveillance du site web **OC PIZZA** n'a pas été intégré dans la procédure de déploiement semi-automatisée. Il est donc recommandé d'installer un tel service après le déploiement du site.

Nous recommandons trois services de surveillance pour garantir la meilleure observation quant à la surveillance du site :

- Interface DIGITAL OCEAN
- SUPERVISOR
- SENTRY

Interface DIGITAL OCEAN

Le site web OC PIZZA étant hébergé chez DIGITAL OCEAN, il peut bénéficier d'une interface de monitoring permettant de suivre son activité.

SENTRY

SENTRY est une plateforme web permettant de concentrer et afficher les évènements et erreurs prélevés depuis un site web.

SUPERVISOR

SUPERVISOR permet de centraliser la gestion des processus internes et externes qui sont utiles au fonctionnement du site web. Ce service pourra notamment être utilisé pour activer et désactiver GUNICORN.

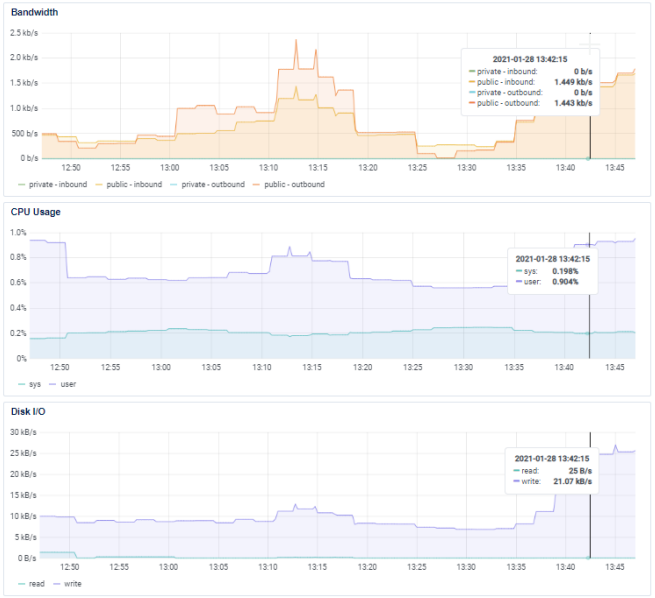
7.1 - Interface DIGITAL OCEAN

Afin de surveiller graphiquement l'activité du site web, veuillez suivre les instructions suivantes :

```
>ssh root@IPpublicaddress
```

```
>curl -sSL https://repos.insightsdigitalocean.com/install.sh
```

Vous pouvez désormais surveiller l'activité du site web en vous rendant sur le droplet associé à OC PIZZA. Dans la section « Graphiques », se trouvent divers graphes informant sur le CPU, l'occupation mémoire ou encore la bande-passante utilisée.



Vous pouvez également aiguiser la personnalisation de cette interface et ajouter des fonctionnalités de surveillance supplémentaires. Pour de plus amples informations, consulter la documentation proposée par DIGITAL OCEAN.

7.2 - Sentry

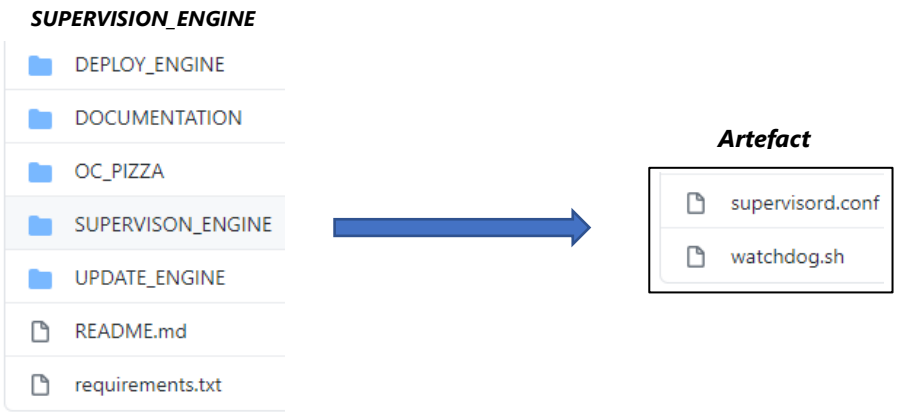
La plateforme SENTRY permet donc de centraliser les évènements et erreurs du site. Nous avons déjà préconfiguré OC PIZZA afin qu'il puisse se connecter à SENTRY.

```python
from pathlib import Path
import OC_PIZZA.the_secrets as tst
import sentry_sdk
from sentry_sdk.integrations.django import DjangoIntegration
import os

sentry_sdk.init(
    dsn="https://PublicKey@sentry.io/0",
    integrations=[DjangoIntegration()]
)
```

7.3 - Description des artefacts SUPERVISION_ENGINE

La procédure de mise en place de la surveillance est automatisée. Cependant, une intervention est nécessaire avant son exécution.

Les éléments de surveillance se trouve dans le répertoire « SUPERVISON_ENGINE ». Ce répertoire contient par défaut deux artefacts :



Supervisord.conf

Ce fichier permet de configurer les entités du service SUPERVISOR, à savoir *supervisord* et *supervisorctl,* mais aussi le service GUNICORN, qui lui est associé au site web. Il permet également de rediriger le rapport d'exécution « fichier log », vers un emplacement souhaité.

Watchdog.sh

Ce script permet de commander les services souhaités en passant par le service SUPERVISOR. Il prend donc en compte le fichier de configuration « supervisord.conf ».

```
#!/bin/bash
# Sequential execution
set -e

# supervisorctl access
supervisorctl -c SUPERVISON_ENGINE/supervisord.conf $@
```

7.4 - Configuration du fichier supervisord.conf

```
[inet_http_server]
port=127.0.0.1:9001

[supervisord]
user=root
logfile=/tmp/supervisord.log
pidfile=/tmp/supervisord.pid

[supervisorctl]
serverurl=http://127.0.0.1:9001
username=user
password=password

[program:oc_pizza-gunicorn]
command=/home/ubuntu_user/OC_PIZZA/oc_pizza_venv/gunicorn OC_PIZZA.wsgi:application --bind 0.0.0.0:80
directory=/OC_PIZZA/OC_PIZZA/
autorestart=true
redirect_stderr=true
stdout_logfile=/home/ubuntu_user/SUPERVISION_ENGINE/log/watchdog.log
```

- La section *inet_http_server* permet de paramétrer le serveur http et son port d'écoute.
- La section *supervisord* est lié au paramétrage des processus propres de la supervision. Il précise l'utilisateur UNIX qui sera utilisé pour l'éxécution des tâches, l'emplacement dédié à la création du journal d'activité, autrement dit, le rapport d'activité ou fichier log. Et enfin, l'emplacement du fichier renseignant le PID du processus.
- La section *supervisorctl* reprend les éléments nécessaires à la connexion au service SUPERVISORD, l'url à utiliser pour accéder au serveur de supervision ainsi que l'utilisateur et mot de passe.
- La section *program :oc_pizza-gunicorn* permet de créer une configuration groupée. Dans notre cas, cette configuration est centrée sur l'application GUNICORN du site web. Il reprend donc les précisions utiles au bon démarrage de l'application. Il permet de lancer et relancer l'application à la suite d'un crash. Enfin, il alimente le journal d'activité *watchdog.log*.

7.5 - Exécution de Supervisor

Une fois le fichier « supervisord.conf » correctement configuré, l'exécution du service peut avoir lieu en suivant les étapes ci-dessous :

```
>ssh root@IPpublicaddress
```

```
>cd ~/OC_PIZZA
```

```
>source oc_pizza_venv/bin/activate
```

```
>supervisord -c SUPERVISION_ENGINE/supervisord.conf
```

- Connexion *ssh* vers le serveur d'hébergement.
- Ouverture du répertoire *OC_PIZZA*.
- Activation de l'environnement virtuel
- Mise en marche du service *supervisord* couplé au paramétrage *supervisord.conf*

7.6 - Utilisation de Supervisor

```
>./SUPERVISION_ENGINE/watchdog.sh stop gunicorn
```

```
./SUPERVISION_ENGINE/watchdog.sh status gunicorn
```

```
>./SUPERVISION_ENGINE/watchdog.sh start gunicorn
```

```
>cat /SUPERVISION_ENGINE/log/watchdog.log
```

Exemple d'utilisation du service SUPERVISOR :

- Arrêt du service *gunicorn*
- Consultation de l'état du service *gunicorn*
- Mise en marche du service *gunicorn*
- Consultation des évènements gérés par *supervisor*

8 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

Dans cet ensemble de procédures, des éléments automatiques ont été implémentés afin de faciliter certaines opérations. Cependant, d'autres opérations nécessitent des interventions manuelles.

La sauvegarde et la restauration s'appuie sur le module « DJANGO-DBBACKUP ».

Son intégration au site OC PIZZA respecte les indications portées dans sa documentation.

8.1 - Procédure de sauvegarde automatique

La procédure de sauvegarde est automatisée. Elle est gérée par le processus de mise à jour cyclique.

Pour rappel, la mise à jour cyclique est un processus intégré au site web via le module « DJANGO-CRONTAB ».

Ce processus remplie également la fonction de sauvegarde automatique du site. Par défaut, nous avons fixé la sauvegarde automatique à **6h du matin tous les premiers du mois**.

Pour modifier cette périodicité, veuillez vous référer à la partie 6.2.3 – Réglage de la périodicité.

Attention, la procédure de sauvegarde concerne uniquement les fonctions « cyclic_backup », appelées dans « CRONJOBS ».

```
CRONJOBS = [
    ('0 6 * * *', 'city001.cron.cyclic_update'),
    ('0 6 * * *', 'city002.cron.cyclic_update'),
    ('0 6 * * *', 'city003.cron.cyclic_update'),
    ('0 6 1 * *', 'city001.cron.cyclic_backup'),
    ('0 6 1 * *', 'city002.cron.cyclic_backup'),
    ('0 6 1 * *', 'city003.cron.cyclic_backup'),
]
```

```
def cyclic_backup():
    try:
        call_command('backup_ocpizza')
    except:
        pass
```

8.2 - Procédure de sauvegarde manuelle

Dans certain cas, il est utile de sauvegarder manuellement la base de données. Pour ce faire, la procédure de maintenance manuelle, présente ci-dessous, doit être respectée :

CAS DE MAINTENANCE MANUELLE	PROCEDURE DE MAINTENANCE MANUELLE
Modification de la base de données	Désactivation du site web > Modification souhaitée > Activation du site web

```
>./manage.py dbbackup
```

Mise en route de la procédure de sauvegarde

8.3 - Procédure de restauration manuelle

Cette opération entre dans le cadre des procédures de maintenance manuelle. Il est donc recommandé de suivre les instructions suivantes :

CAS DE MAINTENANCE MANUELLE	PROCEDURE DE MAINTENANCE MANUELLE
Modification de la base de données	Désactivation du site web > Modification souhaitée > Activation du site web

```
>./manage.py dbrestore
```

Mise en route de la procédure de restauration

9 - GLOSSAIRE

API	Sigle anglais signifiant « Application Programming Interface », interface de programmation applicative en français. C'est une librairie de classes, fonctions et méthodes qui proposent un service WEB particulier.
BATCH	Terme anglais utilisé en informatique. C'est un proche synonyme de « script informatique ». C'est donc un fichier exécutable par une machine qui contient des instructions rédigées par un humain. Selon sa rédaction, il peut à son tour demander l'exécution d'un ou plusieurs autres scripts, on parle donc de « batch ou script en cascade ».
COOKIE	Fichier renfermant des informations liées aux données d'un utilisateur. Ces informations sont utilisées pour améliorer la navigation de l'utilisateur en question.
DIGITAL OCEAN	Fournisseur d'infrastructure cloud américain, hébergeur américain en d'autres termes dont le siège est basé à New York. La couverture en centre données occupe le monde entier. Cet hébergeur propose la pluspart formule standard d'hébergement, du physique au virtuel.
GITHUB	Service web d'hébergement permettant de gérer le versionnage de logiciel. Il est aussi utilisé pour les demandes de fonctionnalités et le suivi des bugs lié à un projet logiciel.
SSH	Sigle anglais signifiant « Secure Shell ». C'est un protocole de communication permettant de sécuriser une connexion entre deux hôtes informatiques.
WSGI	Sigle anglais signifiant « Web Server Gateway Interface ». C'est un type de serveur, voire une spécification qui définit une interface entre des serveurs et des applications web pour le langage Python.