OC Pizza

Application web - Site de commerce et de logistique

Dossier d'exploitation

Version 1.1

Auteur
Kilian Florin
Analyste-Programmeur

OCorp
OCorp.fr
7 Avenue du Segment Fault – 0153487963– contact@OCorp.fr
S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Paris – SIREN 999 999 999 – Code APE : 6202A

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Kilian Florin	03/04/2018	Création du document	1.0
Kilian Florin	26/04/2018	Mise à jour des informations global	1.1

2 - INTRODUCTION

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application OC Pizza…

Le document ci-présent à pour but d'être un support à l'installation et au déploiement de l'application web OC Pizza.

2.2 - Références

Pour de plus amples informations, se référer :

1. DCT – OC Pizza : Dossier de conception technique de l'application
2. DCF – OC Pizza : Dossier de conception fonctionnelle de l'application

3 - PRE-REQUIS

3.1 - Système

3.1.1 - Serveur Web

Serveur physique ou virtuel hébergeant l'application web.

3.1.1.1 - Caractéristiques techniques

CPU : 1 vCore à ~3.1 Ghz

RAM : ~2 Go

Stockage : 10 Go.

3.2 - Bases de données

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

- **PostgreSQL :** version 10

3.3 - Autres ressources

Les logiciels suivants doivent être accessibles et à jour :

- **Python :** version 3.5.0 ou +
- **Virtualenv :** version 15.0.0
- **Supervisor :** version 3.2.0-2 ou +
- **Nginx :** version 1.9.15 ou +
- **Git :** version 1:2.7.4 ou +

4 - PROCEDURE DE DEPLOIEMENT

4.1 - Déploiement de l'Application Web

4.1.1 - Environnement de l'application web

4.1.1.1 - Environnement virtuel

Afin d'installer l'application proprement, il est nécessaire de lui créer un environnement virtuel :

```
Virtualenv env -p python3
Source env/bin/activate
```

4.1.2 - Clone du repository sur GitHub

```
Git clone https://github.com/OcPizza/OcPizza.git
Cd ocpizza
```

4.1.3 - Installation des pré-requis

```
Pip install -r requirements.txt
```

4.1.4 - Configuration de l'application

4.1.4.1 - Fichier de configuration de l'application Django en production

Le répertoire de configuration de l'application en production doit être créé de la manière suivante :

```
Mkdir ocpizza_project/settings
```

Un fichier sera utilisé en substitue du fichier par default de configuration de l'application afin de répondre au besoin de la production, créer le de cette manière :

```
Touch ocpizza_project/settings/production.py
```

4.1.4.2 - Fichier production.py

Afin de configurer l'application, copier ceci en changeant les valeurs entre chevrons.

```
from . import *
import raven
import os

SECRET_KEY = '<Nouvelle clé secrète>'
DEBUG = False
ALLOWED_HOSTS = ['<IP du serveur>']

INSTALLED_APPS += (
    'raven.contrib.django.raven_compat',
)

LOGGING = {
    'version': 1,
    'disable_existing_loggers': True,
    'root': {
        'level': 'INFO', # WARNING by default. Change this to capture more than warnings.
        'handlers': ['sentry'],
    },
    'formatters': {
        'verbose': {
            'format': '%(levelname)s %(asctime)s %(module)s '
                      '%(process)d %(thread)d %(message)s'
        },
    },
    'handlers': {
        'sentry': {
            'level': 'INFO', # To capture more than ERROR, change to WARNING, INFO, etc.
            'class': 'raven.contrib.django.raven_compat.handlers.SentryHandler',
            'tags': {'custom-tag': 'x'},
        },
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
            'formatter': 'verbose'
        }
    },
    'loggers': {
        'django.db.backends': {
            'level': 'ERROR',
            'handlers': ['console'],
            'propagate': False,
        },
        'raven': {
            'level': 'DEBUG',
            'handlers': ['console'],
            'propagate': False,
        },
        'sentry.errors': {
            'level': 'DEBUG',
            'handlers': ['console'],
        },
    },
}

RAVEN_CONFIG = {
    'dsn': '<Liens de votre application sur sentry.io, voir comment dans la suite de ce document> '
    'release': raven.fetch_git_sha(os.path.dirname(os.pardir)),
}


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': '<Nom de la base de données créer  précédemment>',
        'USER': '<Nom de l'utilisateur de la base de données>',
        'PASSWORD': '<Mot de passe de l'utilisateur pgsql>',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

4.1.5 - Base de données

Afin d'autorisé votre application à utiliser PostgreSQL, suivez cette série d'instruction :

```
sudo -u postgres psql
# Vous entrez donc en l'interface de PostgresSQL
CREATE DATABASE ocpizza;
CREATE USER ocpizza WITH PASSWORD '<Votre choix de mot de passe>';
ALTER ROLE ocpizza SET client_encoding TO 'utf8';
ALTER ROLE ocpizza SET default_transaction_isolation TO 'read committed';
ALTER ROLE ocpizza SET timezone TO 'Europe/Paris';
GRANT ALL PRIVILEGES ON DATABASE ocpizza TO ocpizza;
\q
```

Avec ces informations, vous pouvez donc mettre à jours le fichier de configuration créer précédemment.

4.1.6 - Configuration de Nginx

Afin de permettre à Nginx de distribuer correctement les fichiers statiques, voici comment le configurer :

```
Cd /etc/nginx/
sudo touch sites-available/ocpizza
sudo ln -s /etc/nginx/sites-available/ocpizza /etc/nginx/sites-enabled
```

Enfin, changer le contenu de 'sites-available/ocpizza' pour :

```
server {

  listen 80; server_name 178.62.119.70;
  root /home/<votre utilisateur>/ocpizza/;

  location /static {
     alias /home/<votre utilisateur>/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;
     }
  }
```

Puis rechargez Nginx avec :

```
sudo service nginx reload
```

4.1.7 - Configuration de Supervisor

Créez le fichier de configuration relatif à notre application avec ceci :

```
sudo touch /etc/supervisor/conf.d/ocpizza-gunicorn.conf
```

Puis éditez le de la sorte :

```
[program:ocpizza-gunicorn]
command = /home/<Votre Utilisateur>/env/bin/gunicorn ocpizza_project.wsgi:application
user = <Votre utilisateur>
directory = /home/<Votre utilisateur>/ocpizza
autostart = true
autorestart = true
environment = DJANGO_SETTINGS_MODULE='ocpizza_project.settings.production'
stderr_logfile = /var/log/supervisor/ocpizza_stderr.log
stdout_logfile = /var/log/supervisor/ocpizza_stdout.log
```

4.1.8 - Installation de New Relic et Sentry.io

Afin de surveiller le bon fonctionnement de votre application, je vous redirige vers les guides d'installation des deux outils utilisé par notre application.

4.1.8.1 - New relic

https://docs.newrelic.com/docs/infrastructure/new-relic-infrastructure/installation/install-infrastructure-linux

4.1.8.2 - Sentry.io

Créez-vous simplement un compte ici : https://sentry.io/signup/

Enfin, créez une application et changer l'URL de la configuration de Raven dans notre fichier production.py par celui fourni par sentry.io.

5 - PROCEDURE DE DEMARRAGE / ARRET

5.1 - Base de données

```
# Pour arrêter
Sudo service postgresql stop
# Pour démarrer
Sudo service postgresql start
```

5.2 - Application web

```
# Pour arrêter
Sudo supervisorctl stop ocpizza-gunicorn
# Pour démarrer
Sudo supervisorctl start ocpizza-gunicorn
```

PROCEDURE DE MISE A JOUR

5.3 - Base de données

```
# Sauvegarder la base actuelle
Su – postgres
Pg_dumpall > dump.sql
Cp ~postgres/dump.sql ~
# Migrer
Apt-get remove postgresql-x.x.x
Apt-get install postgresql-x.x.x
# Restaurer
Su -postgres
Psql < dump.sql
```

5.4 - Tâche Crons

```
Python manage.py runcrons
```

5.5 - Application web

```
Git pull
```

6 - SUPERVISION/MONITORING

6.1 - Supervision de l'application web

6.1.1.1 - Vérification des ressources

https://infrastructure.newrelic.com/accounts/xxxxxxxx/hosts

6.1.1.2 - Vérification du fonctionnement de l'application

https://sentry.io/oc-pizza/oc-pizza/dashboard/

7 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

7.1 - Sauvegarde de la base de données

```
Python manage.py dumpdata > dump.json
```

7.2 - Restauration de la base de données

```
Python manage.py loaddata  dump.json
```