OC Pizza

oc_pizza

Dossier d'exploitation

Version 1.1

Auteur
Nathan
Développeur

Table des matières

1 - Versions

Auteur	Date	Description	Version
Nathan	12/07/20	Création du document	v1.0
Nathan	13/07/20	Ajustement du format	v1.1

2 - Introduction

2.1 - Objet du document

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

Objectif du document est de décrire en détails les différents élément de l'exploitation et du déploiement de l'application.

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 - Pré-requis

3.1 - Système

3.1.1 - Serveur Web

Serveur physique l'application web sur Digital Ocean.

3.1.1.1 - Caractéristiques techniques

CPU : 2 vCPUs

RAM : 8GB

Stockage : 25 GB (SSD disk)

Transfer : 4TB

3.2 - Bases de données

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

- **PostgreSQL :** version 9.5

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

- **Supervisor :** version 4.2.0 ou +
- **Gunicorn :** version 20.0.4 ou +
- **Nginx :** version 1.18 ou +

3.4 - Autres Ressources

Les logiciels suivants doivent être accessibles et à jour :

- **Python :** version 3.8 ou +
- **Virtualenv :** version 20.0.26
- **Git :** version 2.27.0 ou +

4 - Procédure de déploiement

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 venv -p python3
source venv/bin/activate
```

4.1.2 - Installation de l'application Django

4.1.2.1 - Clone du repository sur GitHub

```
git clone https://github.com/M0l42/oc_pizza
cd oc_pizza
```

4.1.2.2 - Installation des pré-requis

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

4.1.3 - Configuration de l'application django

4.1.3.1 - Fichier de paramètre dédié à la production

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

```
Mkdir oc_pizza_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 oc_pizza_project/settings/production.py
```

4.1.3.2 - Fichier production.py

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

```
from . import *
import raven

SECRET_KEY = '<votre_cle_secrete>'
DEBUG = False
ALLOWED_HOSTS = ['<adresse_ip>']

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql', # on utilise l'adaptateur postgresql
        'NAME': 'oc_pizza', # le nom de notre base de données créée précédemment
        'USER': '<utilisateur>',
        'PASSWORD': '<mot_de_passe',
        'PORT': '5432',
    }
}

INSTALLED_APPS += [
    'raven.contrib.django.raven_compat',
]


RAVEN_CONFIG = {
    'dsn': 'https://somethingverylong@sentry.io/216272', # caution replace by your own!!
    # If you are using git, you can also automatically configure the
    # release based on the git info.
    'release': raven.fetch_git_sha(os.path.dirname(os.pardir)),
}

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'],
            'propagate': False,
        },
    },
}
```

4.1.4 - Création et configuration de la base de donnée

Afin d'autorisé votre application à utiliser PostgreSQL, suivez cette série d'instruction en changeant les valeurs entre chevrons:

```
sudo -u postgres psql

# Vous entrez donc en l'interface de Postgres

# Créez une nouvelle base de données

postgres=# CREATE DATABASE  ocpizza;

CREATE DATABASE

# Créez un l'utilisateur

postgres=# CREATE USER <utilisateur> WITH PASSWORD '<mot_de_passe>';

CREATE ROLE

# Modification conseillé par la documentation de Django

postgres=# ALTER ROLE <utilisateur> SET client_encoding TO 'utf8';

ALTER ROLE

postgres=# ALTER ROLE <utilisateur> SET default_transaction_isolation TO 'read committed';

ALTER ROLE

postgres=# ALTER ROLE <utilisateur> SET timezone TO 'Europe/Paris';

ALTER ROLE

# Quittez l'interface de Postgres

\q
```

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

Une fois le fichier de configuration mis à jours, allé dans le dossier contenant le fichier manage.py :

```
python manage.py migrate
```

4.1.5 - 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/oc_pizza
sudo ln -s /etc/nginx/sites-available/oc_pizza /etc/nginx/sites-enabled
```

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

```
server {

   listen 80; server_name <adress_ip>;
   root /home/<utilisateur>/oc_pizza/;

   location /static {
        alias /home/<utilisateur>/oc_pizza/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
```

4.1.6 - Configuration de Gunicorn et Supervisor

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

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

Puis éditez le de la sorte :

```
[program:oc_pizza-gunicorn]
command = /home/<utilisateur>/oc_pizza/venv/bin/gunicorn
oc_pizza_project.wsgi:application
user = directory = /home/<utilisateur>/oc_pizza
autostart = true
autorestart = true
environment = DJANGO_SETTINGS_MODULE='oc_pizza_project.settings.production'
stderr_logfile = /var/log/supervisor/oc_pizza_stderr.log
stdout_logfile = /var/log/supervisor/oc_pizza_stdout.log
```

4.1.7 - Sentry

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.

4.1.8 - Activer le Monitoring sur Digital Ocean

Installez l'agent de Digital Ocean sur votre serveur :

```
curl -sSL https://agent.digitalocean.com/install.sh | sh
```

Puis allez dans l'interface d'administration de Digital Ocean. Dans votre *droplet* vous avez déjà accès à certaines d'information.

Cliquez sur "Monitoring" puis sur "Create alert policy". Comme vous pouvez le constater, vous avez le choix !

En général, nous surveillons les éléments suivants :

- le CPU et la RAM ne sont pas trop utilisés ;
- la bande passante actuelle est suffisante ;
- le nombre de requêtes n'excède pas la capacité du serveur.

5 - Procédure de démarrage / arrêt

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

6 - Procédure de mise à jour

6.1 - 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

6.2 - Tâche Crons

Afficher le contenue du ficher crontab :

```
crontab -l
```

Modifier le contenue du ficher crontab :

```
crontab -e
```

Syntaxe à respecter :

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sa
t
# |  |  |  |  |
# *  *  *  *  *  user command to be executed
```

6.3 - Application web

Vérifiez que vous avez bien commit tout vos changement puis vous tapez :

```
git pull
```

7 - Supervision/Monitoring

7.1 - Supervision de l'application web

7.1.1 - Vérification du fonctionnement de l'application

7.1.1.1 - Sentry :

Vous avez accez à vos alerte et à vos logs dans la section du projet dans votre compte que vous avez créez plus tôt.

7.1.1.2 - Digtal Ocean :

Dans l'interface d'administration de Digital Ocean vous allez avoir accès à toutes les informations lié au serveur :

- **CPU**, ou processeur en français, est le composant qui exécute les instructions des programmes. Ici vous pouvez visualiser le pourcentage de puissance utilisé.
- **Memory** représente la mémoire vive utilisée.
- **Disk I/O** *(disk input/output)* est la quantité de données lues ou écrites sur le disque dur du serveur par seconde.
- **Disk usage** est la quantité de données stockées dans le disque dur du droplet.
- **Bandwidth** (bande passante) représente la quantité de données transférée entre le droplet et des ressources externes.
- **Top processes** affiche les processus les plus exigeants en terme de mémoire vive (Memory) et de processeur (CPU).

Vous allez aussi recevoir des e-mails dépendament des alertes que vous allez configurer.

8 - Procédure de sauvegarde et restauration

8.1 - Sauvegarde de la base de données

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

8.2 - Restauration de la base de donnée

```
python manage.py loaddata dump.json
```

9 - Glossaire

Monitoring	Le ***monitoring*** est l'anglicisme du terme surveillance et définit la mesure d'une activité.
Cron	Cron est un programme disponible sur les systèmes de type Unix (Linux, Mac Osx ...) permettant de planifier des taches régulières.