OC Pizza

OPENCLASSROOMS

O.C. PIZZA

Projet Origanware

Dossier d'exploitation

Version 2.1

Auteur
Cédric Joseph
Développeur

OC Pizza OPENCLASSROOMS

TABLE DES MATIÈRES

1 - Versions

Auteur	Date	Description	Version
Cédric Joseph	23/10/2020	Création du document	1.4

2 - Introduction

2.1 -Objet du document

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

Il permettra à l'utilisateur final de prendre en main l'application, de la maintenir et de la faire évoluer.

2.2 -Références

Pour de plus amples informations, se référer :

1. **DCT -** : Dossier de conception technique de l'application
2. **DCF -** : Dossier de conception fonctionnelle

3 - Pré-requis

3.1 -Système

3.1.1 - Serveur de Base de données

Serveur de base de données Postgresql hébergeant le schéma ocpizza_database

3.1.1.1 - Caractéristiques techniques

Image: Ubuntu 18.04.3 (LTS) x64

Size: 1 vCPUs - 1GB / 25GB Disk

Prix : $5/mois

Region : LON1

IPv4 : 209.97.178.92

3.1.2 - Serveur Web

Serveur sur lequel est installé le serveur HTTP NGinx

3.1.2.1 - Caractéristiques techniques

Image: Ubuntu 18.04.3 (LTS) x64

Size: 1 vCPUs - 1GB / 25GB Disk

Prix : $5/mois

Region : LON1

IPv4 : 209.97.178.93

3.1.3 - Serveur d'application

Serveur qui permet de communiquer avec le code Django. Il contient le code source.

3.1.3.1 - Caractéristiques techniques

Image: Ubuntu 18.04.3 (LTS) x64

Size: 1 vCPUs - 1GB / 25GB Disk

Prix : $5/mois

Region : LON1

IPv4 : 209.97.178.91...
Installations de Python 3.6 et de pip3

3.2 -Bases de données

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

ocpizza_database.sql : version 2.0

3.3 -Web-services

Les web services suivants doivent être accessibles et à jour :

- Stripe
- Google Maps

3.4 -Autres Ressources

- Python 3 en tant que langage par défaut. Par défaut c'est Python 2 qui est sélectionné.

- sudo apt install python3.6
- `sudo update-alternatives --set /usr/bin/python python /usr/bin/python3.6`
-
- Vérifications :

  python --version

  python3 -V

- Outil de monitoring Digital Ocean (fournisseur du cloud): https://cloud.digitalocean.com/droplets/194336491/graphs?i=b47414&period=hour

  Installer le client de DigitalOcean (en tant qu'utilisateur linux):

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

- Service de monitoring Sentry:

  Sentry est un tableau de bord qui vous permet de visualiser ce qui se passe dans l'application Django. Il faut
  - un compte (https://sentry.io/signup/) que nous vous fournissons
  - puis installer la librairie raven (*$ pip install raven*)
  - enfin lancer la commande *$ sudo supervisorctl restart disquaire-gunicorn*

  Nous vous fournissons les configurations

4 - Procédure de déploiement

4.1 -Déploiement du serveur de base de données

4.1.1 - DataSources

Les accès aux bases de données doivent se configurer à l'aide du fichier.

$home/origanware/oc_database.sql

Pour l'obtenir il est nécessaire de cloner le dépôt git
https://github.com/ocdatabases/ocpizza/origanware.git

4.1.2 - Installation

sudo apt-get -y install postgresql
Toutes les opérations d'administration se font, au départ, avec l'utilisateur *postgres*.

1) La première chose à faire sera de créer un nouvel utilisateur, mais pour ce faire, il faut se connecter au moins une fois en tant qu'utilisateur postgres. Pour devenir postgres et faire les opérations d'administration qui suivent, utilisez sudo :

$ sudo -i -u postgres

Password: le vôtre.

2) Désormais, l'invite de commande doit mentionner que vous êtes actif en tant que postgres. Pour lancer l'invite de commande SQL de PostgreSQL, tapez simplement : *psql*

3) Créer la base de données
postgres=# CREATE DATABASE *ocpizza_database*

4) Protéger le mot de passe de l'utilisateur (le ENCRYPTED permet l'utilisation de md5 dans le pg_hba.conf) :
postgres=# ALTER USER postgres WITH ENCRYPTED PASSWORD 'mon_mot_de_passe';

5) Importer le script de création de base de données:

psql -U postgres -d ocpizza_database < ocpizza_database.sql

4.1.3 - Configuration

4.1.3.1 - Fichier pg_hba.conf

On le trouve ici : /etc/postgresql/9.1/main/pg_hba.conf
On y accède en tant qu'utilisateur linux.

L'authentification du client est contrôlée par un fichier de configuration, qui s'appelle traditionnellement pg_hba.conf et est stocké dans le répertoire de données du cluster de base de données. (HBA signifie authentification basée sur l'hôte.) Un fichier pg_hba.conf par défaut est installé lorsque le répertoire de données est initialisé par initdb .

Le format général du fichier pg_hba.conf est un ensemble d'enregistrements, un par ligne. Les lignes vides sont ignorées, comme tout texte après le caractère de commentaire # .

Chaque enregistrement spécifie un type de connexion, une plage d'adresses IP client (le cas échéant pour le type de connexion), un nom de base de données, un nom d'utilisateur et la méthode d'authentification à utiliser pour les connexions correspondant à ces paramètres. Si aucun enregistrement ne correspond, l'accès est refusé.

4.1.4 - Ressources

- Documentation Postgresql pour Ubuntu
- Pg_hba documentation

4.1.5 - Vérifications

Base de données :

- Exécuter en tant qu'utilisateur *postgres* la requête *select datname from pg_database ;*

4.2 - Déploiement du serveur Web

4.2.1 - Installation

Tout d'abord créons un user *pizzadmin* :

- *adduser pizzadmin*
- *usermod -aG sudo pizzadmin*
- *su pizzadmin*

L'ensemble des commandes qui suivent seront exécutées en tant que *pizzadmin*
Installation : *sudo apt-get install nginx*
La configuration par défaut de Nginx est située dans *etc/nginx/sites-enabled/*

1) Création d'un fichier de configuration
Par convention, les fichiers de configuration sont regroupés dans le dossier sites-enabled. Ils sont tous pris en compte de la même manière.
Ouvrez-le : vous y découvrirez un fichier *default*

Un lien symbolique fait référence à un fichier enregistré ailleurs dans le système. Ainsi, si vous demandez à ouvrir *nginx/sites-enabled/default*, le système ouvrira nginx/sites-available/default.

Vous devez donc :

- créer un nouveau fichier dans *sites-available* ;
- ajouter un lien symbolique dans sites-enabled grâce à la commande *ln*.
- *sudo touch sites-available/origanware*

pizzadmin@origanware:/etc/nginx/$ sudo ln -s /etc/nginx/sites-available/origanware /etc/nginx/sites-enabled
Le chemin doit partir de la racine du système et non du répertoire courant.

Servir l'application Django :

Éditer *nginx/sites-available/disquaire*

```
server {

        listen 80; server_name 209.97.178.91;
        root /home/origanware/;

        location / {
           proxy_set_header Host $http_host;
           proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
           proxy_redirect off;
           proxy_pass http://127.0.0.1:8000;
        }
}
```

Lancer pizzadmin@origanware:/etc/nginx/$ *sudo service nginx reload*

4.2.2 - Configuration

supervisord.log : fichier de configuration des logs supervisors logs

4.2.3 - Ressources

- Documentation de Supervisor

4.2.4 - Vérifications

- Etat du serveur web: *sudo supervisorctl status*

4.3 -Déploiement du serveur applicatif

4.3.1 - Artefacts

Le code source sur github est accessible en clonant un dépôt git:
https://github.com/ocpizza/origanware.git

4.3.2 - Installations

Les commandes qui suivent seront lancées en tant qu'utilisateur linux.

1) Installer le serveur d'application: *pip install gunicorn*

La racine du projet se trouve à */home/origanware/*
Se rendre dans */home/origanware/origanware/* pour lancer *origanware.wsgi:application*

2) Installer Supervisor un système qui permet de contrôler un ensemble de processus vivant dans un environnement UNIX: *sudo apt-get install supervisor*

Tout comme Nginx, Supervisor lit un fichier de configuration situé dans etc/supervisor.
Chaque fichier finissant en .conf et situé dans le chemin
/etc/supervisor/conf.d
représente un processus qui est surveillé par Supervisor. Créez-en un nouveau :
$ sudo vi /etc/supervisor/conf.d/disquaire-gunicorn.conf

À l'intérieur, ajoutez les commande à exécuter pour démarrer Gunicorn :

origanware-gunicorn.conf

command=/home/origanware/env/bin/gunicorn origanware_project.wsgi:application
autostart = true
autorestart = true

environment = ENV="PRODUCTION"
L'option *autostart=true* indique à Supervisor que vous souhaitez lancer ce processus au premier démarrage du serveur.
autorestart=true que la commande doit être exécutée si le processus est interrompu.

Gunicorn a été installé dans un environnement virtuel, c'est pourquoi vous devez indiquer le chemin absolu qui mène à l'exécutable (le fichier qui lance le programme).
En l'occurrence : */home/origanware/env/bin/gunicorn*.

3) Lancer Supervisor
$ sudo supervisorctl reread
origanware-gunicorn: available

$ sudo supervisorctl update
origanware-gunicorn: added process group

4.3.1- Configurations

4.3.2.1 - Fichier production.py

Fichier qui contient les configurations Django pour l'environnement de production.

À la racine du projet se trouve un module *settings* qui contient ce fichier ainsi que la clé secrète du projet.

4.3.3 - Environnement de l'application web

4.3.3.1 - Variables d'environnement

Le serveur d'application Gunicorn doit être exécuté avec la variable d'environnement *gunicorn* définie au démarrage.
$ gunicorn --env DJANGO_SETTINGS_MODULE=origanware.settings origanware.origanware.wsgi
Elle est nécessaire afin de récupérer le répertoire contenant les fichiers de configuration de l'application :

-Dcom.ocpizza.apps.conf=$home_application_conf_directory

INFO : il ne faut pas mettre de « / » à la fin de la valeur de la variable et ne pas utiliser d'espace dans le chemin.

4.3.4 - Répertoire de configuration applicatif

Le répertoire de configuration applicatif doit être créé sur le système de fichier et définit de la façon suivante :

$home/origanware

... fichiers de configuration... :

$home/origanware/requirements.txt
$home/origanware/Pipfile
$home/origanware/manage.py
$home/origanware/.travis.yml
$home/origanware/settings/production.py
$home/origanware/settings/travis.py

...

4.3.5 - Ressources

- Documentation Gunicorn

4.3.6 - Vérifications

Etat du serveur d'application:

- *sudo systemctl status gunicorn*

5 - Procédure de démarrage / arrêt

Les commandes suivantes seront utilisées:

- 'sudo power off' pour éteindre
- 'sudo reboot' pour redémarrer
- Pour démarrer il faut passer par l'interface de Digital Ocean qui permet aussi l'arrêt et le redémarrage.
- Le serveur de bases de données sera le premier à être démarré puis arrêté
- Ordre de démarrage : 1. BDD 2.applicatif 3.Nginx
- Ordre d'arrêt inverse

5.1 -Base de données

- *systemctl start postgresql*
- *pg_ctl start*
- *systemctl stop postgresql*
- *sudo service postgresql restart*

5.2 -Serveur applicatif

- *pkill gunicorn* arrête tous les daemons gunicorn
- *sudo supervisorctl restart all* pour redémarrer
- *sudo supervisorctl stop all* pour arrêter

5.3 - Serveur web,

Exécuter en fonction du besoin :

- *sudo systemctl start nginx*
- *sudo systemctl restart nginx*
- *sudo systemctl stop nginx*

7 - Procédure de mise à jour

7.1 -Base de données

Lancer en tant qu'utilisateur linux la commande *python manage.py fill_db* dans le dossier racine du projet. Il va exécuter le script de création de la base de données à partir du fichier *ocpizza_database.sql*.

- Premier remplissage de la base données :
  psql -U postgres -d oc_pizza_database -a -f oc_pizzadatabase.sql

- Exécuter : *sudo apt-get update puis upgrade*

7.2 -Serveur Web

Exécuter : *sudo apt-get update*

7.3 -Serveur Django

Exécuter: *sudo apt-get update*

8 - Supervision/Monitoring

8.1 -Supervision de l'application web

Afin de tester que l'application web est toujours fonctionnelles:

- Consulter le tableau de bord Sentry et celui de DigitalOcean
- En ligne de commande, en tant qu'utilisateur linux, exécuter *sudo supervisorctl status*.

9 - Procédure de sauvegarde et restauration

Une tâche cron exporte la base de données toutes les 30 minutes sur le serveur de bases de données. Il faut être utilisateur linux pour lancer les commandes ci-dessous:

- Copiez et collez le texte suivant dans le fichier *.pgpass* . Remplacer le nom de base de données par le nom de la base de données que vous souhaitez sauvegarder, *et le nom d'utilisateur* avec un utilisateur ayant accès à la base de données, et mot de passe avec le mot de passe de l'utilisateur de la base de données:

  #hostname: port: base de données: nom d'utilisateur: mot de passe

  localhost: 5432:nom de base de données:postgres:mot de passe

- Enregistrez les modifications dans le fichier .pgpass et quittez l'éditeur de texte.
- À l'invite de commandes, tapez : ***chmod 600 .pgpass***
- Créez une tâche cron (commande ***crontab -e***) qui exécute la commande suivante :

pg_dump --no-password -U dbusername nom de base de données > /chemin/backup.pgsql

Remplacer *dbusername* avec l'utilisateur qui a accès à la base de données, remplacez nom de base de données avec le nom de la base de données que vous souhaitez sauvegarder et remplacez chemin avec le chemin où vous souhaitez stocker le fichier de sauvegarde de la base de données.

Cet exemple utilise *backup.pgsql* pour le nom de fichier de la sauvegarde, mais vous pouvez utiliser le nom de fichier de votre choix.

Exemple de tâche cron :

0,30 * * * * pg_dump --no-password -U postgres ocpizza_database > /home/backups/backup.pgsql

10 - GLOSSAIRE

Déploiement	Fait mettre en fonctionnement un service en le rendant accessible
Monitoring	Surveillance continue de l'état d'un service