



Dossier d'exploitation

Version 1.0

David Barat
Dev. Python



TABLE DES MATIÈRES

1. Versions	3
2. Introduction	4
2.1. Objet du document	4
2.2. Références	4
3. Pré-requis	5
3.1. Système	5
a. Serveur de base de données & serveur web	5
a.1. Caractéristiques techniques	5
3.2. Création Base de données	6
4. Procédure de déploiement	7
4.1. Déploiement de l'application web	7
a. Arborescence applicative	7
b. Configuration Nginx	7
c. Configuration Gunicorn & Supervisor	8
d. Peuplement base de données	9
4.2. Déploiement des Batches	10
a. Arborescence applicative	10
b. Configuration de la crontab	11
5. Procédure de démarrage / Arrêt	12
5.1. La base données	12
5.2. Application web ocpizza	12
6. Procédure de mise à jour	13
6.1. La base données	13
6.2. Batches	13
6.3. Application web	13
7. Supervision / Monitoring	15
7.1. Supervision de l'application web	15
a. Supervision disque	15
b. Supervision RAM & CPU	15
c. Supervision process Nginx	15
d. Supervision process PostgreSql	16
7.2. Monitoring	16
a. Nginx	16
b. Log applicative populate_db.log	17
c. Sentry	17

1. VERSIONS

Auteur	Date	Description	Version
DBA	04/01/2021	Création du document	1.0



2. INTRODUCTION

2.1. Objet du document

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

Les éléments du présent dossier découlent :

- du document de description du besoin client (OC-Pizza-description-de-notre-besoin.pdf)

2.2. Références



3. PRÉ-REQUIS

3.1. Système

Il faut dans un premier temps créer votre utilisateur unix, tapez la commande suivante:

```
adduser user
```

Puis lui accorder les droits d'élévation sudo:

```
gpasswd -a user sudo
```

Voici ci-dessous un tableau récapitulatif version produit

Produit	Version
Django	3.1.1
Nginx	1.18.0
Supervisor	4.1.0
PostgreSQL	12.5

a. Serveur de base de données & serveur web

Pour les besoins du projet nous avons installé le serveur web et la base de données sur le même serveur.

a.1. Caractéristiques techniques

La distribution utilisée est une Ubuntu 20.04 disposant 1 Vcpu et de 2Go de RAM, sur laquelle les packages suivants doivent être installés, utilisez les commandes suivantes:

```
sudo apt-get install nginx,  
sudo apt-get install supervisor,  
sudo apt-get install python3-pip python3-dev
```



```
sudo apt-get install libpq-dev  
sudo apt-get install postgresql postgresql-contrib  
sudo apt-get install virtualenv
```

Puis nous allons créer l'environnement virtual python django version 3.1.1:

```
virtualenv django -p python3
```

3.2.Création Base de données

La base de données est une PostgreSQL **12.5**. Il faut créer au préalable le schéma ocpizza ainsi que le user :

```
CREATE DATABASE ocpizza;  
CREATE USER user WITH PASSWORD 'yourpassword';
```

Nous recommandons un mot de passe respectant les points suivants:

- au moins 8 caractères,
- des majuscules et minuscules,
- des caractères alphanumériques.



4. PROCÉDURE DE DÉPLOIEMENT

4.1. Déploiement de l'application web

a. Arborescence applicative

Dans un premier temps s'assurer d'avoir accès au dépôt git source https://github.com/davidbarat/P6_OCPizza.

Effectuer un git clone dans le \$HOME de votre user crée au \$3.1 :

```
cd $HOME  
git clone https://github.com/davidbarat/P6_OCPizza
```

Vérifier que l'arborescence ocpizza a bien été créée. Puis installez les modules python nécessaires à l'application ocpizza grâce à la commande suivante :

```
cd $HOME  
. django/bin/activate  
pip install -r ocpizza/requirements.txt
```

b. Configuration Nginx

Sous /etc/nginx/sites-available, effectuer les commandes suivantes:

```
sudo vim ocpizza
```

Et coller le bloc suivant dans le fichier ocpizza:

```
server {  
  
    listen 80; server_name 127.0.0.1;  
    root /home/user/repertoire_applicatif/;
```



```
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;  
}
```

Remplacer l'adresse IP **127.0.0.1** par l'adresse IP de votre serveur ou l'url de l'application. Puis afin de finaliser la configuration Nginx, taper les commandes suivantes :

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

Lancer la commande `sudo nginx -t` afin de vérifier la syntaxe des fichiers de configuration nginx, voici le retour si la configuration est correcte:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

c. Configuration Gunicorn & Supervisor

Créer le fichier `ocpizza-gunicorn.conf` sous `/etc/supervisor/conf.d/`

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

```
[program:ocpizza-gunicorn]  
# command = /home/user/env/bin/gunicorn ocpizza_project.wsgi:application  
command=/home/user/django/bin/newrelic-admin run-program /home/user/django/bin/gunicorn  
ocpizza_project.wsgi:application  
user = user  
directory = /home/user/ocpizza  
autostart = true  
autorestart = true
```




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

Une fois le fichier créé, taper les commandes suivantes afin de prendre en compte la nouvelle configuration:

```
sudo supervisorctl reread
ocpizza-gunicorn: available
sudo supervisorctl update
ocpizza-gunicorn: added process group
```

d. Peuplement base de données

Modifier le fichier `production.py` sous `/home/user/ocpizza/ocpizzat/settings` section `DATABASES`, avec les informations de votre environnement :

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql",
        "NAME": "ocpizza", # le nom de notre base de donnees créée précédemment
        "USER": "david", # attention : remplacez par votre nom d'utilisateur
        "PASSWORD": « yourpassword », # attention: remplacer par votre password
        "HOST": "",
        "PORT": "5432",
    }
}
```

Puis on exécute les migrations:

```
cd $HOME/user/ocpizza
. django/bin/activate
./ocpizza/manage.py migrate
```

Finalement on initie la base de données avec des données:



```
/ocpizza/manage.py loaddata ocpizza/store/dumps/store.json
```

Si la commande a réussi, elle vous renvoie le nombre d'object créés:

```
Installed 39 object(s) from 1 fixture(s)
```

4.2.Déploiement des Batches

a. Arborescence applicative

Afin de réceptionner les batchs applicatifs, veuillez créer sous \$HOME/user les répertoires suivants :

- script,
- log,
- conf

Puis créer sous \$HOME/user/script/ le script *populate.ksh* avec le contenu ci-dessous:

```
#!/bin/sh  
cd /home/user  
. django/bin/activate  
cd /home/user/ocpizza; ./manage.py populate_db
```

Puis taper la commande suivante:

```
chmod 755 $HOME/user/script/populate.ksh
```



Puis créer le fichier configuration.ini dans le répertoire \$HOME/user/conf et copier le contenu suivant, en remplaçant user par le user unix créé au \$3.1:

```
# paramètres de configuration du serveur  
[settings]  
log_path : /home/user/log/
```

Enfin modifier le répertoire de configuration dans le fichier populate_db.py, remplacer user par le user unix créé au \$3.1:

```
parser.read('/home/user/conf/configuration.ini')
```

b. Configuration de la crontab

Le script *populate.ksh* est à lancer via la crontab, rajouter la ligne suivante:

```
0 3 * * 1 /home/user/script/populate.ksh
```



5. PROCÉDURE DE DÉMARRAGE / ARRÊT

En cas de redémarrage total de l'application, il faut d'abord arrêter l'application web ocpizza puis la base de données.

Une fois ces actions effectuées, démarrer la base de données et ensuite démarrer l'application ocpizza.

5.1. La base données

Pour arrêter la base de données, il faut taper les commandes suivantes:

```
sudo systemctl stop postgresql
```

Afin de la démarrer:

```
sudo systemctl start postgresql
```

5.2. Application web ocpizza

Pour arrêter l'application web copiez, utilisez les commandes suivantes:

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

Afin de la démarrer:

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



6. PROCÉDURE DE MISE À JOUR

6.1. La base données

Afin de modifier la base de données ocpizza, effectuer un git pull du repository:

```
git pull https://github.com/davidbarat/P6\_OCPizza
```

Puis taper les commandes suivantes:

```
cd $HOME/user/  
./django/bin/activate  
./ocpizza/manage.py makemigrations search  
./manage.py migrate
```

6.2. Batches

En cas de relivraison du script /home/david/script/populate.ksh, récupérer le nouveau script dans le repository https://github.com/davidbarat/P6_OCPizza et effectuer un annule et remplace. Bien vérifier les droits d'exécution, sinon exécutez la commande suivante:

```
chmod 755 $HOME/user/script/populate.ksh
```

6.3. Application web

Sauvegardez dans un premier temps le fichier production.py, ce fichier contient vos paramètres de votre environnement:

```
/home/user/ocpizza/ocpizza_project/settings/production.py
```



Ensuite récupérer le contenu du repository https://github.com/davidbarat/P6_OCPizza et le copier dans votre arborescence application \$HOME/user. Une fois copié, remplacer le fichier **production.py** par le fichier précédemment sauvegardé.

7. SUPERVISION / MONITORING

7.1. Supervision de l'application web

Voici ce que nous recommandons en terme de supervision d'infrastructure.

a. Supervision disque

Le filesystem hébergeant l'application web doit être surveillé, un Warning doit être envoyé à 70% d'utilisation du filesystem puis une erreur Critique à 90% d'utilisation.

b. Supervision RAM & CPU

L'utilisation de la RAM et la CPU doivent être surveillés. Ainsi il convient d'envoyer une alerte lors q'utilisation de la RAM ou de la CPU dépasse les 80% d'utilisation pendant un laps de 10 minutes.

c. Supervision process Nginx

Le process Nginx doit être surveillé. Les logs du process Nginx se situent dans l'arborescence suivante :

```
/var/log/nginx
```

Il y a deux logs différents, les access.log et les error.log. Les access donnent des informations sur la provenance des appels http, les codes retour http ainsi que les ressources du site appelées, exemple de log:

```
91.175.205.59 - - [16/Jan/2021:20:55:50 +0000] "GET /search/assets/img/favicon.ico HTTP/1.1" 404 5638 "http://167.99.212.10/search/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:84.0) Gecko/20100101 Firefox/84.0"
```

L'error.log indique les erreurs rencontrées par le serveur Nginx. Il est possible de modifier la verbosité de celle-ci en modifiant le paramètre suivant dans le fichier /etc/nginx/



nginx.conf en ajoutant debug comme ci-dessous:

```
error_log /var/log/nginx/error.log debug;
```

Il faut ensuite redémarrer le serveur web pour prise en compte:

```
sudo systemctl restart nginx
```

d. Supervision process PostgreSQL

Le process postgre doit être surveillé. Les logs se situent dans l'arborescence suivante:

```
/var/log/postgresql
```

7.2.Monitoring

a. Nginx

Vérifier le fichier /etc/logrotate.d/nginx, si le fichier est non présent, copier/coller le contenu suivant:

```
/var/log/nginx/*.log {  
    daily  
    missingok  
    rotate 14  
    compress  
    delaycompress  
    notifempty  
    create 0640 www-data adm
```




```
sharedscripts
prerotate
    if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
        run-parts /etc/logrotate.d/httpd-prerotate; \
    fi \
endscript
postrotate
    invoke-rc.d nginx rotate >/dev/null 2>&1
endscript
}
```

b. Log applicative populate_db.log

Le script populate_db.py comporte un logger, la rotation est effectuée lors du lancement de celui-ci :

```
logger = logging.getLogger("Rotating Log")
logger.setLevel(logging.DEBUG)
```

```
handler = RotatingFileHandler(
    path + 'populate_db.log',
    maxBytes=2000,
    backupCount=5)
logger.addHandler(handler)
```

c. Sentry

Dans ce paragraphe nous allons configurer et utiliser **Sentry**. Ceci est à titre d'exemple mais nous vous recommandons fortement d'utiliser un outil de monitoring et de suivi de d'erreurs applicatives.

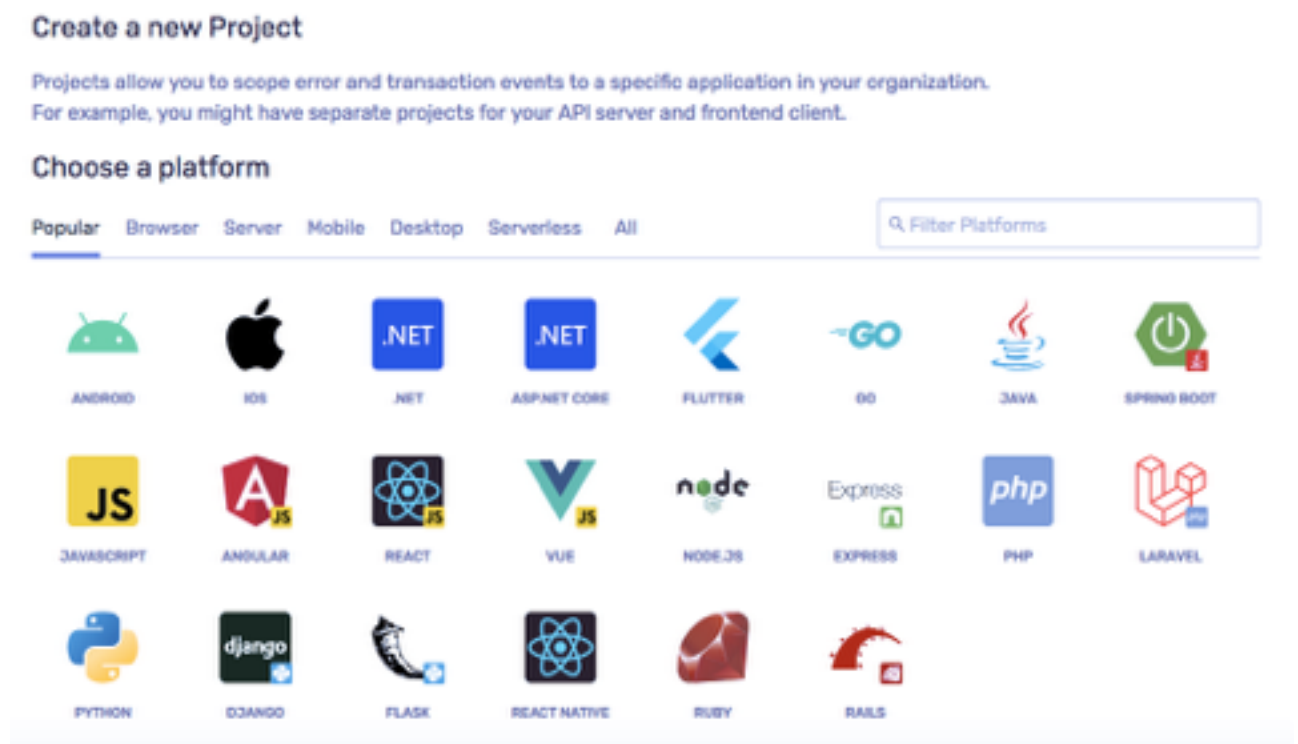


Dans un premier temps, veuillez vous créer un compte sur le site **Sentry** :

<https://sentry.io/auth/login/>

Il est possible de se logger avec son compte GitHub ou de se créer un compte dédié.


Une fois son compte créé, il faut créer un projet puis sélectionner Django :



Puis cliquez sur 'Create Project' :

Give your project a name

Project name Team

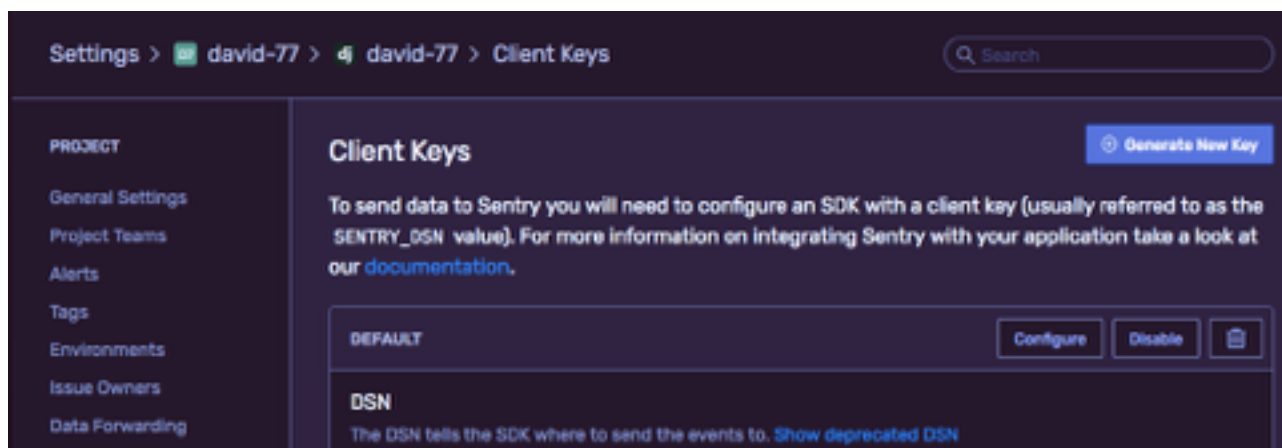
 Django #david + Create Project



Puis modifier le fichier production.py sous settings, il faut remplacer votre key **Sentry** à la section RAVEN_CONFIG -> dsn:

```
RAVEN_CONFIG = {  
    'dsn': 'https://', # A remplacer par votre key Sentry!!  
    'release': raven.fetch_git_sha(os.path.dirname(os.pardir)),  
}
```

Pour récupérer cette key, il faut aller sur le site **Sentry** dans Settings > Votre équipe > Votre projet > Client Keys (DSN) :



Maintenant retournez sur la page d'accueil et cliquez sur 'Issues', **Sentry** vous a listé les différentes problématiques de votre projet (la copie d'écran suivante est donnée à titre indicatif) :

