# **Projet 10 : Deploiement De La Plateforme Web : PurBeurre**

### Site internet déployé.

## Choix technologiques et étapes d'installation

- Création d'une Virtual Machnine (ec2) avec aws d'amazon.
- Installation d'Ubuntu 18.04 sur la VM.
- Mise à jour des paquets , installation de python et postgres.
- Création de l'environnement virtuel sudo apt install virtualenv puis virtualenv env -p python3
- Clonage des données de l'application git clone
   https://github.com/ivan-fr/oc\_projet\_10.git Installation
   des dépendances : pip install -r requirements.txt
- Création de base de données : \$sudo -u postgres createdb
   purbeurre . Création de l'utilisateur \$sudo -u postgres
   createuser --interactive
- Ajout des variables d'environnement (
   DJANGO\_SETTINGS\_MODULE) au chargement de la session utilisateur ubuntu.

```
export DJANGO\_SETTINGS\_MODULE="oc_projet_8.settings.prod
uction"
```

## Création de la configuration de production

Créer un fichier "production.py" dans oc\_projet\_10/oc\_projet\_8/settings/production.py

```
from . import *
SECRET KEY = 'your secret key'
DEBUG = False
ALLOWED HOSTS = [your website adress]
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql psycopg2'
        "NAME": "name",
        "USER": "user",
        "PASSWORD": "your password",
        "HOST": "localhost",
        "PORT": "5432",
}
```

# Installation du serveur web : nginx

```
sudo apt-get install nginx
```

```
La configuration se trouve dans /etc/nginx/sites-available/
Création d'un fichier de configuration touch /etc/nginx/sites-
available/oc_projet_8

Edition et ecriture avec vim sudo vi /etc/nginx/sites-
available/oc_projet_8
```

```
server {
    listen 80; server name 18.188.245.231;
    root /home/ubuntu/oc_projet_10/;
    location /static {
        alias /home/ubuntu/oc projet 10/staticfiles/;
    location / {
        proxy set header Host $http host;
        proxy set header X-Forwarded-For $proxy add x forw
arded for;
        proxy redirect off;
        if (!-f $request filename) {
            proxy pass http://127.0.0.1:8000;
            break;
    }
```

Création du lien vers /etc/nginx/sites\_enabled pour activer le vhost

/etc/nginx\$ sudo ln -s /etc/nginx/sites-available/oc\_proje
t\_8 /etc/nginx/sites-enabled

Nous rechargeons la configuration du serveur

\$sudo service nginx reload

### Installation de supervisor

Supervisor est un système qui permet de contrôler des processus dans un environnement Linux. Ici nous l'utilisons pour contrôler et redémarrer au besoin Gunicorn (serveur HTTP Python qui utilise les spécifications WSGI, utilisé par Django).

sudo apt-get install supervisor

Ajout d'un nouveau processus dans la configuration de supervisor

\$cd /etc/supervisor/conf.d/

/etc/supervisor/conf.d\$ sudo vi oc projet 10-gunicorn.conf

Voici la configuration mise en place

```
[program:oc_projet_10-gunicorn]
command = /home/ubuntu/env/bin/gunicorn oc_projet_8.wsgi:a
pplication
user = ubuntu
directory = /home/ubuntu/oc_projet_10
autostart = true
autorestart = true
environment = DJANGO_SETTINGS_MODULE='oc_projet_8.settings
.production'
```

La commande supervisorct met à dispose plusieurs fonctionalités liés à supervisor.

Ici nous disons à supervisor de prendre en considération les changements de configuration et d'ajouter le processus.

```
$sudo supervisorctl reread
oc_projet_10-gunicorn: available
$sudo supervisorctl update
oc_projet_10-gunicorn: added process group
```

### Affiche le statut

```
$sudo supervisorctl status
oc_projet_10-gunicorn RUNNING pid 126, upti
me 15:27:20
```

### Redémarre le processus

```
$sudo supervisorctl restart oc_projet_10-gunicorn
oc_projet_10-gunicorn: stopped
oc_projet_10-gunicorn: started
```

### **Installation de Travis**

Travis est un service d'automatisation. Travis crée un environnement équivalent à celui de notre application et y fait tourner les tests.

Nous l'interfaçons avec notre dépôt Github. Il surveille dans notre cas la branche staging où désormais toutes les modifications de l'application seront faites.

### **Configuration de Travis**

Création d'un compte.

Ajout sur la dashboard de Travis de notre dépôt Github:

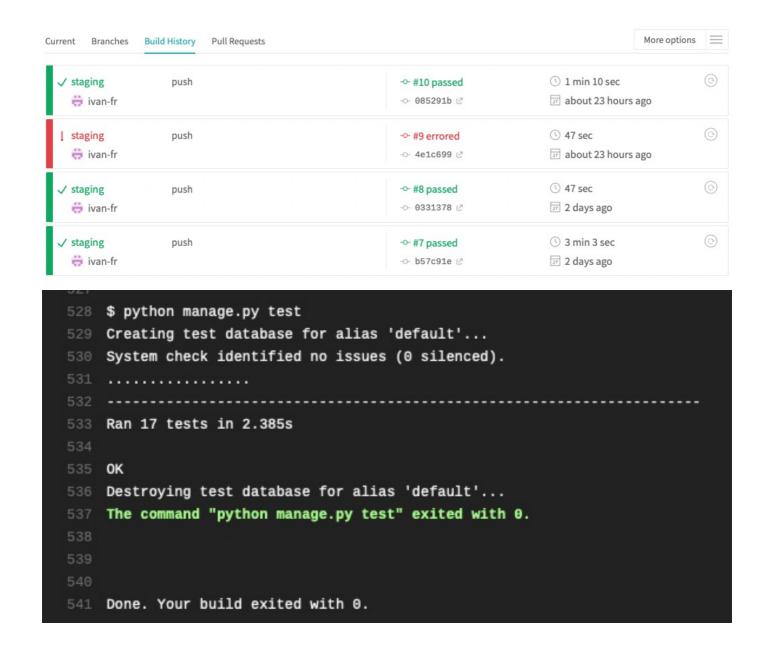


Sur notre environnement local , au même niveau que manage.py , nous créons un nouveau fichier travis.yml

language: python

```
python:
  - "3.6.2"
before script:
  - psql -c 'create database travis_ci_test;' -U postgres
  - psql -c 'create database test;' -U postgres
  - psql -c "CREATE USER besevic WITH PASSWORD 'unpassword
';" -U postgres
  - psql -c "ALTER USER besevic CREATEDB;" -U postgres
  - pip install -r requirements.txt
branches:
  only:
  - staging
env: DJANGO_SETTINGS_MODULE="oc_projet_8.settings.travis"
services:
  - postgresql
script:
  - python manage.py makemigrations
  - python manage.py migrate
  - python manage.py test
```

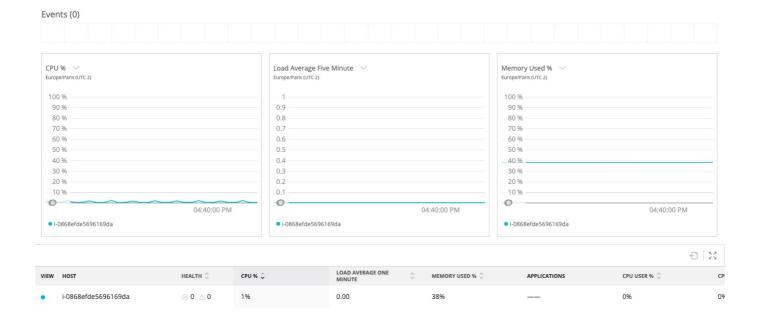
## Une vision de l'historique de notre activité:



# Monitorer le serveur : mise en place de Newrelic

Newrelic Infrastructure offre la possibilité de monitorer notre serveur : Etat de la CPU , de la mémoire vive, le load average etc.

- Création d'un compte sur New Relic
- Sur le site Infrastructure/All Host/Add host



### Suivre les instructions:

- Mise en place d'une clé de licence dans un fichier de configuration Newrelic
- Installation de Newrelic avec cette nouvelle lincence: charger le dépôt d'installation de Newrelic, et installation du paquet.
- Dorénavant le monitoring de notre système peut se surveiller depuis la plateforme New Relic :



## Surveillance de l'application Django oc\_projet\_10 : Sentry

 Création d'un compte puis création d'un projet Django nommé oc\_projet\_10.

Nous devons suivre ses instructions:

Dans oc projet 10 sur le serveur linux nous installons sentry:

```
$pip install --upgrade sentry-sdk==0.7
```

Nous devons également rajouté ces quelques lignes à notre fichier de configuration settings/production.py sur le serveur linux.

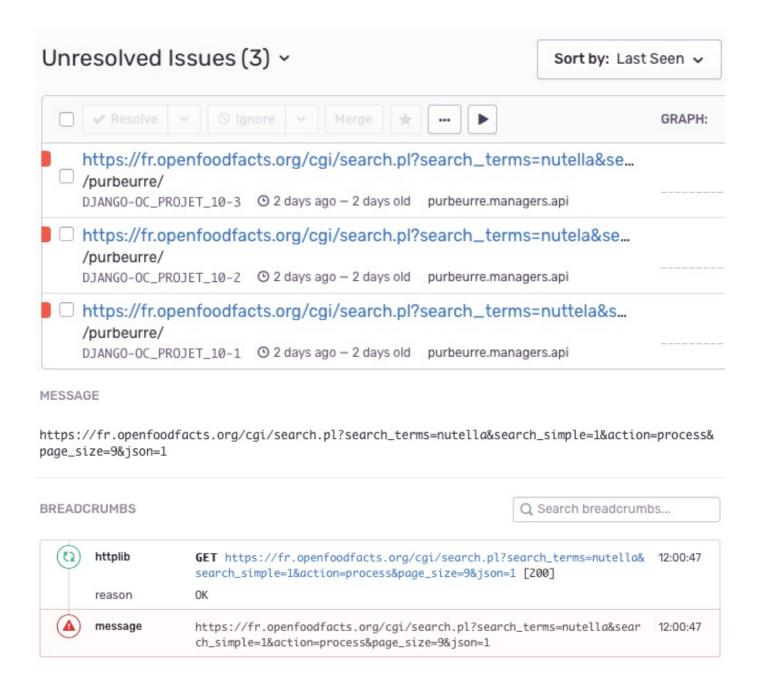
```
import sentry_sdk
from sentry_sdk.integrations.django import DjangoIntegrati
on

sentry_sdk.init(
    dsn="https://43e3b76493e047e48ba64ef49acde99a@sentry.i
    o/1389825",
    integrations=[DjangoIntegration()]
)
```

Dorénavant les erreurs de l'application seront reportées sur sentry.

### Exemples d'erreurs rapportées

 Erreur générée volontairement grâce au module logging dans api.py de l'application django se trouvant dans oc\_projet\_10 sur le serveur linux:



## Tâche planifiée

### **Généralités**

Notre serveur virtuel oc\_projet\_10 peut vouloir mettre à jour sa base de données régulièrement avec des nouvelles données importées d'OpenFoodFacts.

### Programme mis en oeuvre

Le programme est une tâche personnalisée Django : elle se lance donc avec la commande suivante:

```
(env)$python manage.py update_db
```

Nous récucupérons la liste de tout les produits enregistrer dans la base de données et la comparons avec le produit correspondant dans openfoodfacts.

Extrait de code:

```
with transaction.atomic():

    products_db = Product.objects.all()
    product_count = products_db.count()

    self.stdout.write(self.style.SUCCESS(
        "il y a " + str(product_count) + " produits."
    ))

    for i, product_db in enumerate(products_db, start=
1):
        product_from_api = ApiManager.get_product(product_db.bar_code,
        with
```

```
_clean=True)
```

L'intégralité du code se trouve ici.

### Sauvegarde des logs

Comme le programme renvoie un certain nombre d'informations, il est intéressant de sauver les logs de son déroulement.

C'est pour cela que, dans la tâche planifiée, nous créons aussi un fichier de logs update\_db.log qui se trouve dans ~/var/log/oc\_projet\_10/

Création du fichier de sauvegarde :

```
/$ mkdir /var/log/oc_projet_10
/$ touch /var/log/oc_projet_10/update_db.log
```

Changement des droits d'accès :

```
:/var/log/oc_projet_10# chown ubuntu update_db.log
```

### Script shell et Cron job

```
$ cd /usr/local/bin
$ sudo -s
$ vim update_database.sh
```

Dans update database.sh j'écris :

```
#!/bin/bash
source /home/ubuntu/env/bin/activate
export DJANGO_SETTINGS_MODULE="oc_projet_8.settings.produc
tion"
python /home/ubuntu/oc_projet_10/manage.py update_db > /va
r/log/oc_projet_10/update_db.log
deactivate
exit
```

Pour pouvoir éxecuter python manage.py update db

Ensuite je rends le script exécutable :

```
/usr/local/bin# chmod +x update_db.sh
```

Je crée la tâche planifiée :

```
$crontab -e
```

j'ajoute la ligne suivante dans l'éditeur qui vient de s'ouvrir.

```
00 00 * * 0 /usr/local/bin/update_db.sh
```

La tâche s'exécutera tous les dimanches à minuit.

On rafraichit le cron via la commande ci-dessous pour être sûr que la tâche s'éxecute.

sudo service cron restart