Guide déploiement de l'application

Pré-requis

- Compte Fly.io
- Flyctl installé sur la machine
- Python et Pip installés
- PostgreSQL installé localement pour le développement
- La configuration de l'environnement de travail doit être fait en amont

1. Configuration Initiale

1.1. Configurer l'application Django

Créer un fichier .env pour les variables d'environnement
(Il était impossible pour moi de déployer en utilisant un .env ou os.getenv(). En local le fonctionnement était valide mais en production une erreur m'était renvoyée :
« django.db.utils.OperationalError: could not connect to server: No such file or directory Is the server running locally and accepting connections on Unix domain socket "/var/run/postgresql/.s.PGSQL.5432"? » et j'ai passé des heures à essayer de trouver une solution mais je n'ai pas eu le temps d'investiguer davantage le problème donc j'ai tout mis en clair ce qui m'a permis de déployer, mais j'ai push sur github une version sans mes credentials)

```
SECRET_KEY='votre_secret_key'
DB_NAME='nom_de_la_base'
DB_USER='utilisateur'
DB_PASSWORD='mot_de_passe'
DB_HOST='adresse_ip_du_serveur_postgres'
DB_PORT='5432'
```

Configurer les settings.py :

```
import environ
env = environ.Env()
environ.Env.read_env()
```

Il est aussi possible de la faire avec « os.getenv() ».

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': env('DB_NAME'),
        'USER': env('DB_USER'),
        'PASSWORD': env('DB_PASSWORD'),
        'HOST': env('DB_HOST'),
        'PORT': env('DB_PORT'),
    }
}
```

Collecter les fichiers statiques :

```
python manage.py collectstatic
```

Créer et appliquer les migrations :

```
python manage.py makemigrations python manage.py migrate
```

1.2. Créer et configurer la base de données PostgreSQL sur Fly.io

Créer une application PostgreSQL sur Fly.io :

```
flyctl apps create nom_app_postgres
```

Générer et sauvegarder le fichier de configuration fly.toml :

```
flyctl config save --app nom_ app_postgres
```

Déployer l'app fly postgres :

```
fly deploy . --app <pg-app-name> --image flyio/postgres:<major-version>
```

Configurer les accès dans pg_hba.conf (généralement dans C:\Program Files\PostgreSQL\16\data):

```
host all all adresse_ip_de_votre_serveur/32 md5 (ou scram-sha-256)
```

Configurer les adresses écoutées dans postgresql.conf :

```
listen_addresses = '*'
```

Redémarrer le serveur PostgreSQL :

Se rendre dans Service puis rechercher le serveur Postgres et faire un clic droit Redémarrer.

2. Déploiement de l'application Django

Installer le serveur Gunicorn :

```
python -m pip install gunicorn
```

Installer WhiteNoice :

```
python -m pip install whitenoise
```

Ajouter le middleware dans les settings.py :

```
INSTALLED_APPS = [
    ...
    'whitenoise.runserver_nostatic',
```

```
    'django.contrib.staticfiles',
    ...
]

MIDDLEWARE = [
    …
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    ...
]
```

Installer les dépendances nécessaires (à ajouter dans requirements.txt) :

```
pip freeze > requirements.txt
```

Lancer l'application Django sur Fly.io :

```
fly launch
```

Le launch fournira les credentials à sauvegarder pour fly postgres.

Déployer l'application:

```
flyctl deploy --app nom_de_votre_app_django
```

3. Configuration et Tests

3.1. Configurer les variables d'environnement

Définir les variables d'environnement pour PostgreSQL:

```
fly secrets set DATABASE_URL=postgres://example.com/mydb
```

Définir la clé secrète Django:

```
flyctl secrets set SECRET_KEY='votre_secret_key' --app nom_app_django
```

3.2. Vérification et Tests

Tester la connexion à la base de données :

```
psql -h nom_app_postgres.fly.dev -U nom_utilisateur -d db_nom
```

Accéder à l’application Django :

Visiter l'URL de l’application pour vérifier que tout fonctionne correctement.

4. Résolution des Problèmes

4.1. Erreurs courantes que j’ai pu avoir et solutions

Erreur 500 (Internal Server Error) :

- Vérifier les logs pour obtenir des informations détaillées.
- S’assurer que toutes les migrations sont appliquées.
- S’assurer que les fichiers statiques sont correctement collectés.
- Vérifier le pare feu si le port 5432 est bien configuré.

Problèmes de connexion à PostgreSQL :

- Vérifier les configurations dans pg_hba.conf et postgresql.conf.
- S’assurer que l'utilisateur et les mots de passe sont corrects.

Problèmes de variables d'environnement :

- Vérifier que le fichier .env est bien lu dans les settings de Django.
- S’assurer que les secrets sont correctement définis sur Fly.io.