Déploiement de l'application

Cette documentation décrit les étapes que j'ai suivies pour déployer l'application Arcadia sur Fly.io. Elle couvre l'initialisation de la base de données PostgreSQL, l'exportation et l'importation des données, la configuration des secrets nécessaires, ainsi que l'utilisation d'un Dockerfile et la modification du fichier fly.toml.

Prérequis

Un compte Fly.io

Flyctl installé

PostgreSQL installé localement

Une application Node.js configurée

Étapes de déploiement

1. Inscription et installation de Fly.io

- **Je me suis inscrit sur fly.io.**

- **J'ai installé l'outil de ligne de commande Fly.io :**

  Copier le code :

  ```
  curl -L https://fly.io/install.sh | sh
  ```

- **Je me suis connecté à Fly.io depuis mon terminal :**

  Copier le code :

  ```
  flyctl auth login
  ```

2. Initialisation du projet sur Fly.io

- **J'ai initialisé une nouvelle application Fly.io dans le répertoire racine de mon projet Arcadia :**

  Copier le code :

  ```
  flyctl launch
  ```

  J'ai suivi les instructions pour configurer mon application, en choisissant une région proche de mes utilisateurs et en confirmant que j'avais déjà une configuration de déploiement.

3. Configuration de la base de données PostgreSQL sur Fly.io

- **J'ai créé une base de données PostgreSQL sur Fly.io nommée arcadia-db :**

  Copier le code :

  ```
  flyctl postgres create --name arcadia-db
  ```

- **J'ai exporté les données de ma base de données de développement locale arcadia :**

  Copier le code :

  ```
  pg_dump -U postgres -d arcadia -f backup.sql
  ```

- **J'ai importé les données dans la base de données PostgreSQL sur Fly.io. Pour cela, je me suis d'abord connecté à la base de données sur Fly.io :**

  Copier le code :

  ```
  flyctl postgres connect -a arcadia-db
  ```

- **Puis, dans le terminal interactif de la base de données, j'ai importé le fichier SQL :**

  Copier le code :

  ```
  psql -U postgres -d arcadia-db

  \i backup.sql
  ```

4. Configuration des secrets sur Fly.io

- **Pour que mon application puisse se connecter à la base de données et utiliser d'autres informations sensibles, j'ai configuré les secrets nécessaires :**

Copier le code :

flyctl secrets set DATABASE_URL=[URL_de_ma_base_de_données] API_KEY=[ma_clé_API]

- **J'ai ensuite vérifié que les secrets étaient correctement configurés :**

  Copier le code :

  flyctl secrets list -a arcadia

5. Utilisation du DockerfileFly.io m'a fourni un Dockerfile que j'ai utilisé pour containeriser mon application.

- **Voici le Dockerfile que j'ai utilisé :**

  dockerfile

  Copier le code :

```
# syntax = docker/dockerfile:1


# Adjust NODE_VERSION as desired
ARG NODE_VERSION=21.6.1
FROM node:${NODE_VERSION}-slim as base


LABEL fly_launch_runtime="Node.js"


# Node.js app lives here
WORKDIR /app


# Set production environment
ENV NODE_ENV="production"


# Throw-away build stage to reduce size of final image
FROM base as build
```

```
# Install packages needed to build node modules

RUN apt-get update -qq && \

    apt-get install --no-install-recommends -y build-essential node-gyp pkg-config
python-is-python3



# Install node modules

COPY --link package-lock.json package.json ./

RUN npm ci



# Copy application code

COPY --link . .



# Final stage for app image

FROM base



# Copy built application

COPY --from=build /app /app



# Start the server by default, this can be overwritten at runtime

EXPOSE 3000

CMD [ "node", "server.js" ]
```

6. Modification du fichier fly.toml

- **J'ai également modifié le fichier fly.toml pour ajouter un espace de stockage pour mes fichiers statiques. Voici le fichier fly.toml :**

  toml

  Copier le code :

```
# fly.toml app configuration file for brocelianzoo
# Generated on 2024-05-18T17:50:27+02:00

app = "brocelianzoo"
primary_region = "cdg"

[build]

[mounts]
  source = "app_data"
  destination = "/app/uploads"

[http_service]
  internal_port = 3000
  force_https = true
  auto_stop_machines = true
  auto_start_machines = true
  min_machines_running = 1
  processes = ["app"]

  [http_service.concurrency]
    type = "requests"
    soft_limit = 150
    hard_limit = 200

[[vm]]
  memory = "1gb"
```

cpu_kind = "shared"

cpus = 1

7. Déploiement de l'application

- **Enfin, j'ai déployé mon application sur Fly.io en utilisant la commande suivante :**

  Copier le code :

  flyctl deploy

  Cette commande a pris en charge le processus de build et de déploiement de mon application sur l'infrastructure de Fly.io.