

# Mini-rapport G6 — DevOps

## Docker / Compose / Sécurité / Environnement / Réseaux

Projet : SDID Energy Anomaly 2025–2026

3 février 2026

### 1 Contexte et rôle du groupe G6

Le groupe G6 est responsable de la mise en place de l'environnement d'exécution du projet via **Docker** et **Docker Compose**. L'objectif est d'assurer un déploiement **répétable**, **isolé** et **sécurisé**, en standardisant la configuration réseau, les variables d'environnement et la persistance des données.

### 2 Objectifs

- Créer un **Compose isolé** dédié au socle PostgreSQL du projet.
- Gérer correctement les **variables d'environnement** (fichier `.env` local + `.env.example` versionné).
- Mettre en place une **configuration réseau** maîtrisée (réseau backend, option interne).
- Assurer la **persistance** de la base via un **volume Docker**.
- Ajouter des **mesures de sécurité** (ex. `no-new-privileges`).
- Fournir des **commandes de lancement** claires et une structure prête à intégrer les services des autres groupes.

### 3 Livrables (G6)

#### 3.1 Structure

Les fichiers G6 sont regroupés dans le dossier :

- `G6-devops/docker-compose.yml`
- `.env.example`
- (optionnel) `config/postgres_security.sql` (script d'initialisation)

#### 3.2 Docker Compose PostgreSQL

Le service db utilise `postgres:15-alpine` avec :

- **env\_file** + variables `POSTGRES_DB`, `POSTGRES_USER`, `POSTGRES_PASSWORD`
- **ports** : mapping externe (ex. `5434:5432`) pour éviter les conflits
- **volume** : persistance `/var/lib/postgresql/data`
- **healthcheck** : `pg_isready` pour valider l'état du service
- **security\_opt** : `no-new-privileges:true`

### 3.3 Gestion des secrets et environnement

- Le fichier `.env` est **local** (non versionné).
- Le fichier `.env.example` est versionné pour guider l'équipe.
- Le `.gitignore` est ajusté pour éviter de pousser des données sensibles.

## 4 Exécution et vérification

### 4.1 Lancement de la base (G6 seulement)

Depuis la racine du projet :

```
docker compose --project-name sdid_g6 -f .\G6-devops\docker-compose.yml up -d
```

### 4.2 Vérifier l'état

```
docker compose --project-name sdid_g6 -f .\G6-devops\docker-compose.yml ps
```

### 4.3 Test de connexion PostgreSQL (à l'intérieur du conteneur)

```
docker exec -it sdid_g6-db-1 sh -lc 'psql -U $POSTGRES_USER -d $POSTGRES_DB -c "select 1;"'
```

## 5 Résultat

Le socle DevOps est fonctionnel : PostgreSQL démarre, devient **healthy**, conserve les données via volume, et la configuration est pilotée par `.env`. La base est prête pour l'intégration des services (ingestion, analyse, dashboard) via profils Compose.

## 6 Bonnes pratiques retenues

- Isolation par `--project-name` pour éviter les conflits de conteneurs.
- Réseau backend dédié et possibilité de mode interne.
- Sécurité minimale `no-new-privileges`.
- `.env.example` pour partager la configuration sans exposer les secrets.