

KWIKILA Care — Guide de déploiement local en RDC (souveraineté des données)

Objectif : héberger KWIKILA Care sur un datacenter situé en République Démocratique du Congo (ex. Raxio Data Centres Kinshasa, Vodacom Business Cloud, Africa Data Centres Lubumbashi), au lieu du cloud Emergent, afin de répondre aux exigences de localisation des données de santé fixées par le Ministère de la Santé Publique.

1. Ce que ce paquet change

Avant (hébergement Emergent)	Après (hébergement local RDC)
Stockage des pièces jointes patients sur « Emergent Object Storage » (cloud géré, situé hors RDC)	Stockage sur MinIO , serveur S3-compatible installé dans le même datacenter que l'application
Base de données MongoDB gérée par Emergent	MongoDB installé localement, données jamais transmises à l'étranger
Certificat SSL géré par Emergent	Certificat SSL automatique (Let's Encrypt) géré par Caddy , inclus dans ce paquet
Déploiement par plateforme Emergent	Déploiement par Docker Compose , reproductible sur n'importe quel serveur Linux

Le code applicatif (FastAPI + React + MongoDB) est **strictement identique**. Seul le composant de stockage des fichiers bascule de « Emergent Object Storage » vers « MinIO » via la variable d'environnement `STORAGE_BACKEND=minio` (déjà câblée dans `backend/core.py`).

2. Pré-requis matériels et réseau

- Un serveur (VM ou dédié) chez un hébergeur **situé en RDC** :
- Minimum recommandé pour la phase pilote (3 hôpitaux) : 4 vCPU, 8 Go RAM, 100 Go disque SSD.
- Système d'exploitation : Ubuntu Server 22.04 LTS ou 24.04 LTS.
- Un nom de domaine pointant vers l'adresse IP publique du serveur (ex. `kwikilacare.cd` ou sous-domaine `app.kwikilacare.cd`), avec un enregistrement DNS de type **A**.
- Ports ouverts sur le pare-feu : **80** (HTTP, redirection) et **443** (HTTPS).
- Accès SSH administrateur au serveur.

3. Installation de Docker et Docker Compose

```
# Sur le serveur Ubuntu, en tant qu'administrateur (sudo)
curl -fsSL https://get.docker.com | sh
```

```
sudo usermod -aG docker $USER
# Se reconnecter (ou : newgrp docker) pour que le groupe prenne effet
docker --version
docker compose version
```

4. Récupération du code

```
git clone <URL_DU_DEPOT_GIT_KWIKILA_CARE> kwikila-care
cd kwikila-care
```

Le code est téléchargeable via la fonctionnalité « **Save to GitHub** » de l'interface Emergent, qui pousse l'intégralité du dépôt (avec historique Git) vers votre compte GitHub.

5. Configuration des variables d'environnement

5.1 Méthode automatisée (recommandée)

```
./scripts/setup_local_rdc.sh
```

Ce script crée `.env.docker` et `backend/.env.docker` à partir des modèles, génère automatiquement `JWT_SECRET` (256 bits), les identifiants MinIO et un mot de passe fondateur forts (`openssl rand`), et vous demande votre nom de domaine. Le mot de passe fondateur généré s'affiche une seule fois à l'écran : **notez-le immédiatement**.

Complétez ensuite manuellement dans `backend/.env.docker` les clés qui ne peuvent pas être générées automatiquement : `RESEND_API_KEY`, `PAWAPAY_TOKEN` (si Mobile Money activé), `CORS_ORIGINS`.

5.2 Méthode manuelle (si vous préférez tout contrôler)

Fichier racine (`.env.docker`)

```
cp .env.docker.example .env.docker
nano .env.docker
```

Renseigner :

- `DOMAIN` : votre nom de domaine (ex. `kwikilacare.cd`)
- `MINIO_ROOT_USER` / `MINIO_ROOT_PASSWORD` : identifiants forts pour le stockage (générer avec `openssl rand -hex 16`)
- `PUBLIC_BACKEND_URL` : laisser **vide** (le frontend et le backend sont servis sous le même domaine via Caddy)

Fichier backend (`backend/.env.docker`)

```
cp backend/.env.docker.example backend/.env.docker
nano backend/.env.docker
```

Renseigner au minimum :

- `JWT_SECRET` : générer avec `openssl rand -hex 64`
- `CORS_ORIGINS` : votre domaine (`https://kwikilacare.cd,https://www.kwikilacare.cd`)
- `RESEND_API_KEY` : clé de votre compte Resend (emails transactionnels — rappels, factures)
- `PAWAPAY_TOKEN` et `PAWAPAY_ENABLED=true` : si vous activez les paiements Mobile Money en production
- `FOUNDER_EMAIL` / `FOUNDER_INITIAL_PASSWORD` : identifiants du compte fondateur, à changer après le premier login

6. Démarrage de la plateforme

```
docker compose --env-file .env.docker up -d --build
```

Cette commande construit les images (backend, frontend) et démarre 5 conteneurs :

```
mongo, minio, backend, frontend, caddy.
```

Vérifier l'état :

```
docker compose ps
docker compose logs -f backend
```

Caddy obtient automatiquement un certificat SSL pour votre domaine dès que le DNS pointe correctement vers le serveur (peut prendre 1 à 2 minutes au premier démarrage).

Accéder à la plateforme : `https://votre-domaine.cd`

7. Migration des pièces jointes existantes (Emergent -> MinIO)

Si des hôpitaux utilisent déjà la version Emergent avec des pièces jointes stockées sur Emergent Object Storage, un script de migration **automatisé** est fourni avant la bascule définitive :

```
export MONGO_URL="<URL de la base de données existante (Emergent)>"
export DB_NAME="mavimpi_care"
export EMERGENT_LLM_KEY="<clé LLM Emergent existante>"
export MINIO_ENDPOINT="http://<ip-ou-domaine-serveur-local>:9000"
export MINIO_ACCESS_KEY="<MINIO_ROOT_USER défini dans .env.docker>"
export MINIO_SECRET_KEY="<MINIO_ROOT_PASSWORD défini dans .env.docker>"
export MINIO_BUCKET="kwikila-care"

python3 scripts/migrate_attachments_to_minio.py
```

Le script parcourt tous les patients ayant des pièces jointes, télécharge chaque fichier depuis l'ancien stockage Emergent et le re-téléverse dans MinIO sous le même chemin (`stored_name`), sans rien modifier dans MongoDB. Un résumé (réussites/échecs) s'affiche à la fin. **Ne désactivez l'ancien stockage Emergent qu'après avoir vérifié 0 échec.**

8. Sauvegardes

Un script de sauvegarde automatisée est fourni : `scripts/backup_rdc.sh`.

```
# Sauvegarde manuelle (MongoDB + fichiers MinIO), dossier de sortie au choix
./scripts/backup_rdc.sh /mnt/backups-externes
```

Pour une sauvegarde quotidienne automatique (recommandé), ajouter à la crontab du serveur (`crontab -e`) :

```
0 2 * * * /chemin/vers/kwikila-care/scripts/backup_rdc.sh /mnt/backups-externes >> /var/log/kwikila-backup
```

- Le script exporte MongoDB (`mongodump`) et archive le volume MinIO (`minio.tar.gz`) dans un sous-dossier horodaté.
- Conservation automatique des 14 derniers jours (les sauvegardes plus anciennes sont supprimées).
- **Important** : copiez régulièrement le dossier de sortie vers un support **hors du serveur** (disque externe, autre datacenter, ou service de stockage froid) pour vous protéger d'une panne matérielle complète.
- La plateforme envoie également, en interne, un **export CSV hebdomadaire par email** à chaque hôpital (fonctionnalité existante, indépendante du mode d'hébergement).

9. Conformité ARPTC et cadre réglementaire

Ce paquet de déploiement répond aux volets suivants :

- **Localisation des données de santé en RDC** : MongoDB et MinIO tournent exclusivement sur le serveur situé en RDC ; aucune donnée patient ne transite par un cloud étranger.
- **Agrément ARPTC (Autorité de Régulation de la Poste et des Télécommunications)** : requis pour l'exploitation commerciale d'un service numérique de santé grand public en RDC. Démarches à engager séparément auprès de l'ARPTC (dossier administratif, pas technique) — voir le budget mis à jour dans `DOSSIER_TECHNIQUE_FINANCIER_PILOTE.md` (poste « Agrément ARPTC »).
- **Convention avec le Ministère de la Santé** : voir `CONVENTION_CADRE_HOPITAL_PARTENAIRE.md` et `DOSSIER_PARTENARIAT_SECRETARIAT_GENERAL_SANTE.md`.

10. Mise à jour de la plateforme

```
cd kwikila-care
git pull
docker compose --env-file .env.docker up -d --build
```

Les données (MongoDB, MinIO) sont conservées dans des volumes Docker persistants (`mongo_data` , `minio_data`) : elles ne sont **jamais supprimées** par une mise à jour.

11. Support

Pour toute question technique sur ce paquet de déploiement, contacter l'équipe KWIKILA Care : contact@kwikilacare.com