Déploiement EcoRide sur VPS

Objectif du document

Décrire la démarche réelle et les étapes exécutées pour déployer l'application EcoRide sur un VPS Hostinger avec Docker Compose, puis resserrer l'exposition au web (ne laisser public que le serveur web).

1) Contexte et architecture de déploiement

Architecture en conteneurs

EcoRide est déployée sous forme de services Docker :

- nginx : serveur web + reverse proxy vers PHP-FPM
- php : application Symfony (PHP-FPM)
- postgresql : base de données relationnelle
- mongodb : journal d'événements (NoSQL)
- mailhog : serveur SMTP de test + interface web

Le fichier d'orchestration : docker-compose.yml

Ports exposés initialement (configuration de départ)

- Application : http://<IP_VPS>:8080
- Mailhog (interface) : http://<IP_VPS>:8025
- SMTP Mailhog : <IP_VPS>:1025
- PostgreSQL : <IP_VPS>:5432
- MongoDB : <IP_VPS>:27017

Remarque importante (réalité du choix)

Exposer PostgreSQL/MongoDB publiquement n'est pas nécessaire au fonctionnement. Cette exposition a été conservée au départ pour simplifier les tests, puis j'ai prévu (et détaillé) la fermeture via Docker Compose + pare-feu.

2) Pré-requis sur le VPS

- Ubuntu 24.04 LTS
- Accès SSH (compte root au départ)

- Paquets : git, docker.io, docker-compose-v2

3) Installation des outils sur le VPS

Connexion SSH (root)

```
ssh root@<IP_VPS>
```

Installation

```
apt update
apt install -y docker.io docker-compose-v2 git
```

Vérifications

```
docker --version
docker compose version
```

Résultat attendu : les commandes répondent avec une version.

4) Création d'un utilisateur dédié au déploiement

Pourquoi ?

Éviter de déployer et manipuler l'application en root.

Étapes

```
adduser --disabled-password --gecos "" ecoride
usermod -aG docker ecoride
mkdir -p /srv/ecoride
chown -R ecoride:ecoride /srv/ecoride
```

Basculer sur l'utilisateur

```
su - ecoride
```

5) Accès GitHub depuis le VPS (clé SSH)

Générer une clé (utilisateur ecoride)

```
ssh-keygen -t ed25519 -C "deploiement-ecoride" -f ~/.ssh/id_ed25519
```

Afficher la clé publique

```
cat ~/.ssh/id_ed25519.pub
```

Ajouter la clé dans GitHub

Chemin : Settings → SSH and GPG keys → New SSH key

Tester la connexion

```
ssh -T git@github.com
```

Résultat attendu : message de réussite d'authentification (sans shell).

6) Récupération du projet (clone)

Cloner dans le répertoire de déploiement

```
cd /srv/ecoride
git clone git@github.com:rebecca-roussel/ecoride.git ecoride
cd ecoride
```

Se placer sur la branche utilisée

```
git checkout develop
git status -sb
```

7) Configuration des variables d'environnement (VPS)

7.1 Fichier .env (UID/GID)

Le service PHP utilise :

user: "${UID}:${GID}"

Sans .env, Docker Compose affiche :

WARN The "UID" variable is not set. Defaulting to a blank string.

Fixer UID/GID sur le VPS :

```
cd /srv/ecoride/ecoride
printf "UID=%s\nGID=%s\n" "$(id -u)" "$(id -g)" > .env
cat .env
```

Exemple attendu :

```
UID=1001
GID=1001
```

7.2 Fichier .env.docker

Ce fichier est utilisé par php et postgresql.

Points importants côté VPS :

- DEFAULT_URI doit pointer vers l'IP publique du VPS + port 8080
- APP_SECRET doit être différent de la valeur locale

Exemple (extraits) :

```
APP_ENV=dev
APP_DEBUG=1
APP_SECRET=...
APP_TIMEZONE=Europe/Paris
DEFAULT_URI=http://<IP_VPS>:8080
POSTGRES_DB=ecoride
POSTGRES_USER=ecoride
POSTGRES_PASSWORD=ecoride
POSTGRES_HOST=postgresql
POSTGRES_PORT=5432
DATABASE_URL=pgsql://ecoride:ecoride@postgresql:5432/ecoride
MONGO_URI=mongodb://mongodb:27017
MONGO_DB=ecoride_journal
MONGO_COLLECTION=journal_evenements
```

Générer un secret sans PHP :

```
openssl rand -hex 32
```

8) Lancement des conteneurs

Démarrage

```
cd /srv/ecoride/ecoride
docker compose up -d
docker compose ps
```

Résultat attendu : nginx, php, postgresql, mongodb, mailhog en Up.

9) Configuration Nginx (Symfony)

Le conteneur Nginx est configuré via :

docker/nginx/default.conf

Points clés (fonctionnels + sécurité) :

- Racine web : root /var/www/html/public;
- Routage Symfony : try_files $uri /index.php$is_args$args;
- Exécution PHP uniquement via index.php (réduction de surface d'attaque)
- Blocage des autres .php + fichiers cachés (.env, .git, etc.)
- Cache 7 jours sur les fichiers statiques

10) Vérifier que l'application répond

Test depuis le VPS (local)

```
curl -I http://localhost:8080 | head -n 30
```

Résultat attendu : HTTP/1.1 200 OK

Test depuis navigateur

- Application : http://<IP_VPS>:8080
- Mailhog : http://<IP_VPS>:8025

11) Initialisation de la base de données (scripts SQL)

Où sont les scripts ?

Dans docs/sql/

Méthode retenue (robuste)

L'exécution directe -f docs/sql/... peut échouer selon le contexte (chemins/volume).

Méthode fiable : piper le contenu vers psql dans le conteneur.

Depuis /srv/ecoride/ecoride :

```
cat docs/sql/01_schema.sql | docker compose exec -T postgresql psql -U
ecoride -d ecoride -v ON_ERROR_STOP=1
cat docs/sql/02_donnees_demo.sql | docker compose exec -T postgresql psql
-U ecoride -d ecoride -v ON_ERROR_STOP=1
cat docs/sql/03_requetes_verification.sql | docker compose exec -T
postgresql psql -U ecoride -d ecoride -v ON_ERROR_STOP=1
```

Messages du type "trigger … does not exist, skipping" : normal.

Le script supprime d'abord certains triggers s'ils existent, puis les recrée.

Vérifications rapides

```
docker compose exec -T postgresql psql -U ecoride -d ecoride -c "SELECT
count(*) AS nb_utilisateurs FROM utilisateur;"
docker compose exec -T postgresql psql -U ecoride -d ecoride -c "SELECT
count(*) AS nb_covoiturages FROM covoiturage;"
```

12) Commandes utiles (exploitation)

Logs

```
docker compose logs -n 100 --no-log-prefix nginx
docker compose logs -n 100 --no-log-prefix php
docker compose logs -n 100 --no-log-prefix postgresql
```

Redémarrer

```
docker compose restart
```

Arrêter

```
docker compose down
```

Arrêter + supprimer les volumes (réinitialisation complète des données)

```
docker compose down -v
```

Sécurisation : réduire l'exposition au web (étapes appliquées)

Principe visé

- Exposé au web : uniquement le serveur web nginx (80/443, ou 8080 temporaire)
- Privé : PostgreSQL, MongoDB, Mailhog → accessibles seulement dans le réseau Docker
- Pare-feu : n'autoriser que 22 (SSH) + 80/443 (web) (+ 8080 tant que nécessaire)

Étape 1 — Fermer les services internes dans docker-compose.yml

Sur le VPS :

```
cd /srv/ecoride/ecoride
nano docker-compose.yml
```

À retirer (selon mes blocs existants) :

- PostgreSQL : supprimer la section ports: "5432:5432"
- MongoDB : supprimer la section ports: "27017:27017"

- Mailhog : supprimer aussi :
  - ports: "1025:1025"
  - ports: "8025:8025"

Conséquence volontaire :

- Mailhog reste utilisable en interne : PHP y accède via mailhog:1025
- mais l'interface web 8025 n'est plus publique

Redémarrage :

```
cd /srv/ecoride/ecoride
docker compose up -d
docker compose ps
```

Étape 2 — Vérifier que seul nginx écoute vers l'extérieur

```
ss -lntp | grep -E ':8080|:8025|:1025|:5432|:27017' || true
```

Attendu : 8080 oui, tout le reste non.

Étape 3 — Activer le pare-feu UFW (configuration réalisée)

Vérifier l'état :

```
ufw status
```

Configurer :

```
ufw default deny incoming
ufw default allow outgoing
ufw allow OpenSSH
ufw allow 80/tcp
ufw allow 443/tcp
ufw allow 8080/tcp
ufw enable
ufw status verbose
```

Pourquoi autoriser 8080 ?

Parce que nginx est publié en 8080 à ce stade.

Quand nginx passera en 80/443, je pourrai retirer 8080.

Étape 4 — Vérifier que tout reste fonctionnel

Test web (VPS)

```
curl -I http://localhost:8080 | head -n 12
```

Test base de données (via réseau Docker)

```
docker compose exec -T postgresql psql -U ecoride -d ecoride -c "SELECT
now();"
```

Attendu : les deux commandes répondent correctement.

Étape 5 — Passage du domaine vers le VPS (DNS Hostinger)

Constat côté VPS : domaine non résolu au départ

Au début, une requête DNS renvoyait NXDOMAIN (domaine non trouvé) :

```
dig eco-ride.fr A
```

Résultat observé : status: NXDOMAIN

Configuration effectuée chez Hostinger

Dans la zone DNS Hostinger :

- A : eco-ride.fr → IP du VPS 72.61.161.107
- CNAME : www → eco-ride.fr

Vérification en interrogeant les serveurs DNS Hostinger

```
dig @ns1.dns-parking.com eco-ride.fr A +noall +answer
dig @ns2.dns-parking.com eco-ride.fr A +noall +answer
dig @ns1.dns-parking.com www.eco-ride.fr CNAME +noall +answer
```

Résultat attendu (observé) :

- eco-ride.fr → 72.61.161.107
- www.eco-ride.fr → CNAME eco-ride.fr

Vérification via des résolveurs publics

```
dig @1.1.1.1 eco-ride.fr A +noall +answer
dig @8.8.8.8 eco-ride.fr A +noall +answer
dig @1.1.1.1 www.eco-ride.fr CNAME +noall +answer
```

Résultat attendu (observé) identique.

Vérification HTTP sur le domaine

```
curl -I http://eco-ride.fr | head -n 12
curl -I http://www.eco-ride.fr | head -n 12
```

Étape 6 — Passer l'application de 8080 vers 80/443

Pourquoi ?

- 8080 est utile en test, mais un site web public doit écouter en 80 (HTTP) et 443 (HTTPS).
- Ça simplifie aussi le pare-feu (UFW) et l'accès utilisateur.

Action : modification docker-compose.yml

Dans le service nginx, le mapping devient :

- 80:80
- 443:443

Puis redémarrage ciblé :

```
cd /srv/ecoride/ecoride
docker compose config > /dev/null && echo "OK compose" || echo "ERREUR
compose"
docker compose up -d --force-recreate nginx
docker compose ps
```

Vérification : ports réellement exposés

```
ss -lntp | grep -E ':80|:443' || true
docker ps --format 'table {{.Names}}\t{{.Ports}}'
```

Résultat attendu : nginx publié en 80/443.

Étape 7 — Mise en place HTTPS Let's Encrypt (Certbot en conteneur)

Principe

- Nginx doit pouvoir servir un fichier temporaire dans : /.well-known/acme-challenge/
- Certbot utilise le mode -webroot pour prouver la propriété du domaine.
- Les certificats sont stockés dans un volume Docker letsencrypt.

Pré-requis côté Nginx : route ACME

Dans docker/nginx/default.conf, un bloc permet de servir les challenges :

location ^~ /.well-known/acme-challenge/ { ... }

Le point important : ne pas bloquer ce chemin avec la règle "fichiers cachés".

Génération du certificat

```
cd /srv/ecoride/ecoride

docker compose run --rm certbot certonly \
  --webroot -w /var/www/certbot \
  -d eco-ride.fr -d www.eco-ride.fr \
  --email rebecca.roussel10@gmail.com \
  --agree-tos --no-eff-email
```

Résultat attendu (observé) :

- certificat créé dans /etc/letsencrypt/live/eco-ride.fr/
- date d'expiration affichée (ex: 2026-05-19)

Activation TLS dans Nginx

Dans docker/nginx/default.conf :

- un bloc server écoute sur 443 ssl;
- il référence :

```
ssl_certificate     /etc/letsencrypt/live/eco-ride.fr/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/eco-ride.fr/privkey.pem;
```

Puis :

```
docker compose exec -T nginx nginx -t
docker compose exec -T nginx nginx -s reload
```

Vérifications HTTPS + redirection

```
curl -I http://eco-ride.fr | head -n 20
curl -I https://eco-ride.fr | head -n 20

curl -I http://www.eco-ride.fr | head -n 20
curl -I https://www.eco-ride.fr | head -n 20
```

Résultat attendu (observé) :

- HTTP → 301 vers HTTPS
- HTTPS → 200 (souvent en HTTP/2)

Étape 8 — Pare-feu UFW : n'autoriser que SSH + Web

Objectif

- Autoriser : 22, 80, 443
- Refuser le reste par défaut

Mise en place (root)

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow OpenSSH
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
sudo ufw status verbose
```

Résultat attendu :

```
sudo ufw status numbered
```

Doit montrer uniquement OpenSSH, 80, 443 (IPv4 + IPv6).

Vérification "écoute réseau"

```
sudo ss -lntp | grep -E ':(22|80|443)\b' || true
sudo ss -lntp | grep -E ':(5432|27017|8025|1025)\b' || true
```

Attendu : uniquement 22/80/443 visibles, rien sur 5432/27017/8025/1025.

Étape 9 — Renouvellement automatique du certificat

Test manuel (dry-run)

```
cd /srv/ecoride/ecoride
docker compose run --rm certbot renew --dry-run
```

Résultat attendu : all simulated renewals succeeded.

Cron root (renouvellement + reload nginx)

Éditer la crontab root :

```
sudo crontab -e
```

Ajouter (exécution quotidienne à 03:15) :

```
15 3 * * * cd /srv/ecoride/ecoride && docker compose run --rm certbot
renew --quiet && docker compose kill -s HUP nginx
```

Étape 10 — Contrôles finaux (checklist)

- Site accessible : https://eco-ride.fr et https://www.eco-ride.fr
- HTTP redirige vers HTTPS (301)
- docker compose ps : tous les services nécessaires sont Up
- ufw status : seulement 22/80/443 autorisés
- ss -lntp : seulement 22/80/443 en écoute publique
- DB initialisée (9 utilisateurs, 7 covoiturages) via scripts SQL