Documentation de Déploiement EcoRide

1. Stratégie de Déploiement

Architecture de Déploiement Cible

Internet

↓

[CDN/Edge] ← Frontend (Vercel)

↓

[Load Balancer] ← Backend API (Railway)

↓

[Bases de Données Railway]

├── PostgreSQL (Railway Database)

└── MongoDB (Railway Database)

Environnements

1. **Développement** : Local avec Docker
2. **Test/Staging** : Branches de feature déployées automatiquement
3. **Production** : Branche main déployée après validation

2. Préparation au Déploiement

Checklist Pré-Déploiement

Backend

- [ ] Variables d'environnement configurées
- [ ] Migrations de base de données appliquées
- [ ] Tests unitaires et d'intégration passants
- [ ] Logs et monitoring configurés
- [ ] Sécurité renforcée (CORS, rate limiting, validation)

- [ ] Build de production optimisé

Frontend

- [ ] Variables d'environnement de production configurées
- [ ] Build optimisé (minification, tree-shaking)
- [ ] Assets optimisés (images, fonts)
- [ ] Meta tags SEO configurés
- [ ] Tests e2e passants
- [ ] Performance auditée (Lighthouse)

Configuration des Variables d'Environnement

Backend Production (.env.production)

Base de données Railway

DATABASE_URL="postgresql://postgres:password@railway-db-host:5432/database"

MONGODB_URI="mongodb://mongo:password@railway-mongo-host:27017/ecoride"

Authentification

JWT_SECRET="super-secret-production-key-256-bits"

Serveur

PORT=3000

NODE_ENV=production

CORS

FRONTEND_URL="https://ecoride-app.vercel.app"

Email

SMTP_HOST=smtp.gmail.com

SMTP_PORT=587

SMTP_USER=noreply@ecoride.fr

SMTP_PASS=app-password-gmail

Monitoring

SENTRY_DSN="https://your-sentry-dsn"

Frontend Production (.env.production)

VITE_API_URL=https://ecoride-api.up.railway.app/api

VITE_APP_VERSION=1.0.0

VITE_ENVIRONMENT=production

3. Configuration des Bases de Données sur Railway

PostgreSQL avec Railway

Étapes de Configuration

1. **Création du Service PostgreSQL**
    - Se connecter sur railway.app
    - Créer un nouveau projet "EcoRide"
    - Ajouter un service PostgreSQL depuis le catalog

Configuration de la Base

Railway génère automatiquement les credentials

Récupérer les variables d'environnement depuis l'interface Railway

2.

Variables Générées par Railway

PGHOST=containers-us-west-xxx.railway.app

PGPORT=5432

PGDATABASE=railway

PGUSER=postgres

PGPASSWORD=xxx-xxx-xxx

DATABASE_URL=postgresql://postgres:xxx@containers-us-west-xxx.railway.app:5432/railway

3.

Application des Migrations

Utiliser la DATABASE_URL fournie par Railway

npx prisma migrate deploy

npx prisma generate

4.

MongoDB avec Railway

Configuration MongoDB sur Railway

1. **Ajout du Service MongoDB**
    - Dans le même projet Railway
    - Ajouter MongoDB depuis le catalog des services
    - Railway configure automatiquement le service

Variables d'Environnement MongoDB

MONGOHOST=containers-us-west-xxx.railway.app

MONGOPORT=27017

MONGODATABASE=railway

MONGOUSER=mongo

MONGOPASSWORD=xxx-xxx-xxx

MONGODB_URI=mongodb://mongo:xxx@containers-us-west-xxx.railway.app:27017/railway

2.

Configuration de Connexion

// Configuration MongoDB avec Railway

```
const mongoUri = process.env.MONGODB_URI || 'mongodb://localhost:27017/ecoride';


mongoose.connect(mongoUri, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
```

3.

4. Déploiement du Backend sur Railway

Configuration Railway Backend

1. **Création du Service Web**
    - Dans le projet Railway existant
    - Connecter le dépôt GitHub
    - Sélectionner la branche main
    - Root Directory : ecoride-server (si applicable)

Configuration Build Railway

```
 // package.json - scripts nécessaires
{
  "scripts": {
    "build": "tsc",
    "start": "node dist/index.js",
    "dev": "nodemon src/index.ts"
  }
}
```

2.
3. **Variables d'Environnement Railway**
    - Configurer via l'interface Railway
    - Les variables de bases de données sont automatiquement injectées

Ajouter manuellement :

NODE_ENV=productionJWT_SECRET=${{Postgres.JWT_SECRET}}FRONTEND_URL=https://ecoride-app.vercel.app

-

Configuration Dockerfile (Optionnel)

```
# Railway peut utiliser un Dockerfile custom si nécessaire

FROM node:20-alpine


WORKDIR /app


COPY package*.json ./

RUN npm ci --only=production


COPY . .

RUN npm run build


EXPOSE 3000


CMD ["npm", "start"]
```

Script de Déploiement Railway

```
#!/bin/bash

# deploy-railway.sh


echo "🚀 Déploiement Backend EcoRide sur Railway"
```

```
# Installation des dépendances

echo "📦 Installation des dépendances..."

npm ci --only=production


# Build de l'application

echo "🏗️ Build de l'application..."

npm run build


# Application des migrations (Railway injecte automatiquement DATABASE_URL)

echo "🗄️ Application des migrations..."

npx prisma migrate deploy


# Génération du client Prisma

echo "🔧 Génération du client Prisma..."

npx prisma generate


echo "✅ Prêt pour déploiement Railway!"
```

5. Déploiement du Frontend

Vercel (Recommandé)

Configuration Vercel

1. **Import du Projet**
    - Connecter le dépôt GitHub
    - Sélectionner ecoride-client comme Root Directory

Configuration Build

```
// vercel.json
{
  "buildCommand": "npm run build",
  "outputDirectory": "dist",
  "framework": "vite",
  "rewrites": [
    {
      "source": "/(.*)",
      "destination": "/index.html"
    }
  ]
}
```

2.

Variables d'Environnement Vercel

```
# Via CLI Vercel ou interface web
VITE_API_URL=https://ecoride-backend.up.railway.app/api
```

3.

6. Configuration CI/CD avec GitHub Actions

Workflow Backend (.github/workflows/backend.yml)

```
name: Backend CI/CD

on:
  push:
```

```
    branches: [ main, develop ]
    paths: [ 'ecoride-server/**' ]
  pull_request:
    branches: [ main ]
    paths: [ 'ecoride-server/**' ]

jobs:
  test:
    runs-on: ubuntu-latest

    services:
      postgres:
        image: postgres:15
        env:
          POSTGRES_PASSWORD: test
          POSTGRES_DB: ecoride_test
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5
        ports:
          - 5432:5432

    steps:
```

```
    - uses: actions/checkout@v3

    - name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '20'
        cache: 'npm'
        cache-dependency-path: ecoride-server/package-lock.json

    - name: Install dependencies
      working-directory: ./ecoride-server
      run: npm ci

    - name: Run tests
      working-directory: ./ecoride-server
      run: npm test
      env:
        DATABASE_URL: postgresql://postgres:test@localhost:5432/ecoride_test
        JWT_SECRET: test-secret

    - name: Build
      working-directory: ./ecoride-server
      run: npm run build

  deploy:
```

```
needs: test

runs-on: ubuntu-latest

if: github.ref == 'refs/heads/main'


steps:

- name: Deploy to Railway

  run: |

    # Railway déploie automatiquement via GitHub integration

    echo "✅ Déploiement automatique Railway déclenché"
```

7. Avantages de Railway pour EcoRide

Avantages Spécifiques

1. **Simplicité de Configuration**
    - Bases de données provisionnées en un clic
    - Variables d'environnement automatiquement générées
    - Pas de configuration manuelle complexe
2. **Intégration Seamless**
    - Backend et bases de données dans le même projet
    - Réseau privé entre services
    - Monitoring intégré
3. **Développeur-Friendly**
    - CLI Railway pour développement local
    - Logs en temps réel
    - Rollbacks faciles
4. **Scaling Automatique**
    - Auto-scaling selon la charge
    - Pas de configuration serveur manuelle

Configuration CLI Railway (Développement)

```
# Installation du CLI Railway

npm install -g @railway/cli


# Login

railway login


# Lier le projet local

railway link


# Développement local avec variables de production

railway run npm run dev


# Déploiement direct depuis le CLI

railway up
```

8. Monitoring sur Railway

Dashboard Intégré

- **Métriques** : CPU, RAM, requêtes/sec
- **Logs** : Logs applicatifs en temps réel
- **Santé** : Status des services et bases de données

Configuration Health Check

```
// ecoride-server/src/routes/health.routes.ts

import { Router } from 'express';

import { PrismaClient } from '@prisma/client';
```

```
const router = Router();
const prisma = new PrismaClient();

router.get('/health', async (req, res) => {
  try {
    // Vérifier connexion PostgreSQL
    await prisma.$queryRaw`SELECT 1`;

    // Vérifier connexion MongoDB (si utilisé)
    // await mongoose.connection.db.admin().ping();

    res.status(200).json({
      status: 'healthy',
      timestamp: new Date().toISOString(),
      version: process.env.RAILWAY_REPLICA_ID || '1.0.0',
      environment: process.env.RAILWAY_ENVIRONMENT || 'production'
    });
  } catch (error) {
    res.status(500).json({
      status: 'unhealthy',
      error: error.message
    });
  }
});
```

export default router;

9. Processus de Déploiement avec Railway

Déploiement Automatique

1. **Configuration Auto-Deploy**

    - Railway surveille automatiquement la branche main
    - Push vers main → déploiement automatique
    - Rollback en un clic depuis l'interface

Variables d'Environnement

Via l'interface Railway ou CLI

railway variables set NODE_ENV=production

railway variables set JWT_SECRET=your-jwt-secret

2.

Déploiement Manuel (si nécessaire)

Forcer un nouveau déploiement

railway up --detach

Ou redéployer le dernier commit

railway redeploy

3.

10. Sauvegarde avec Railway

Sauvegardes Automatiques

- **PostgreSQL** : Sauvegardes quotidiennes automatiques
- **MongoDB** : Snapshots réguliers
- **Rétention** : 7 jours pour le plan gratuit, plus pour les plans payants

Export de Données

Export PostgreSQL depuis Railway

railway connect postgres

Puis utiliser pg_dump

Ou via tunnel

railway run pg_dump $DATABASE_URL > backup.sql

11. URLs de Production

Avec Railway, vos URLs de production seront :

- **Backend API** : https://ecoride-backend.up.railway.app
- **Frontend** : https://ecoride-app.vercel.app
- **PostgreSQL** : containers-us-west-xxx.railway.app:5432
- **MongoDB** : containers-us-west-xxx.railway.app:27017

12. Coûts et Scaling

Plan Railway

- **Développement** : Plan gratuit (500h/mois)
- **Production** : Plan Pro recommandé pour usage continu
- **Bases de données** : Incluses dans les plans Railway

Cette configuration Railway offre une solution intégrée et simple pour déployer l'application EcoRide avec toutes ses dépendances.