

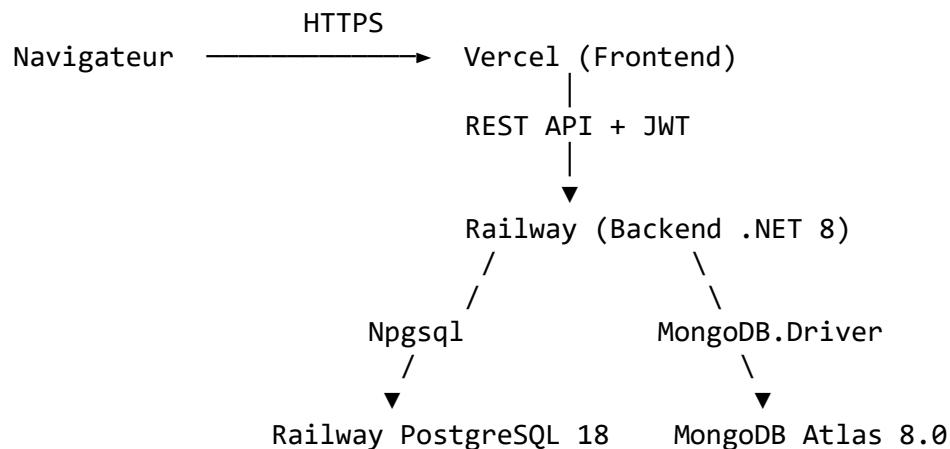
# Dossier de Déploiement

## Sommaire

1. Architecture de déploiement.....	3
2. Installation locale (Docker Compose) .....	3
2.1. Pré-requis .....	3
2.2. Cloner le repository.....	3
2.3. Lancer l'environnement.....	3
2.4. Accès.....	4
2.5. Arrêter l'environnement.....	4
3. Déploiement MongoDB Atlas .....	5
3.1. Créer un cluster .....	5
3.2. Configurer le réseau.....	5
3.3. Créer un utilisateur base de données .....	5
3.4. Récupérer la connection string .....	5
4. Déploiement Railway (Backend + PostgreSQL) .....	6
4.1. Créer le projet.....	6
4.2. Ajouter PostgreSQL.....	6
4.3. Configurer le service Backend.....	6
4.4. Variables d'environnement.....	6
4.5. Domaine personnalisé (optionnel) .....	7
5. Déploiement Vercel (Frontend) .....	7
5.1. Connecter le repository .....	7
5.2. Configuration.....	7
5.3. Variables d'environnement .....	7
5.4. Domaine personnalisé .....	7
6. Configuration Brevo (Emails).....	9
6.1. Créer un compte .....	9
6.2. Pourquoi Brevo ?.....	9
7. CI/CD — GitHub Actions .....	9
7.1. Pipeline CI ( <code>ci.yml</code> ).....	9
7.2. Protection de la branche main ( <code>pr-main.yml</code> ).....	9

7.3. Déploiement automatique .....	9
8. Environnements .....	10
9. Vérification post-déploiement.....	11
Checklist .....	11
Tests de santé .....	11
10. Rollback .....	11
Railway.....	11
Vercel.....	11
Base de données.....	11

## 1. Architecture de déploiement



Service	Plateforme	URL
Frontend	Vercel	<a href="https://www.fantasy-realm.com">https://www.fantasy-realm.com</a>
Backend API	Railway	<a href="https://fantasyrealm-api-production.up.railway.app">https://fantasyrealm-api-production.up.railway.app</a>
BDD PostgreSQL	Railway	Interne (connexion via variable Railway)
BDD MongoDB	MongoDB Atlas	Cluster cloud (région Europe)
Emails	Brevo	API transactionnelle

## 2. Installation locale (Docker Compose)

### 2.1. Pré-requis

- Docker Desktop installé
- Git

### 2.2. Cloner le repository

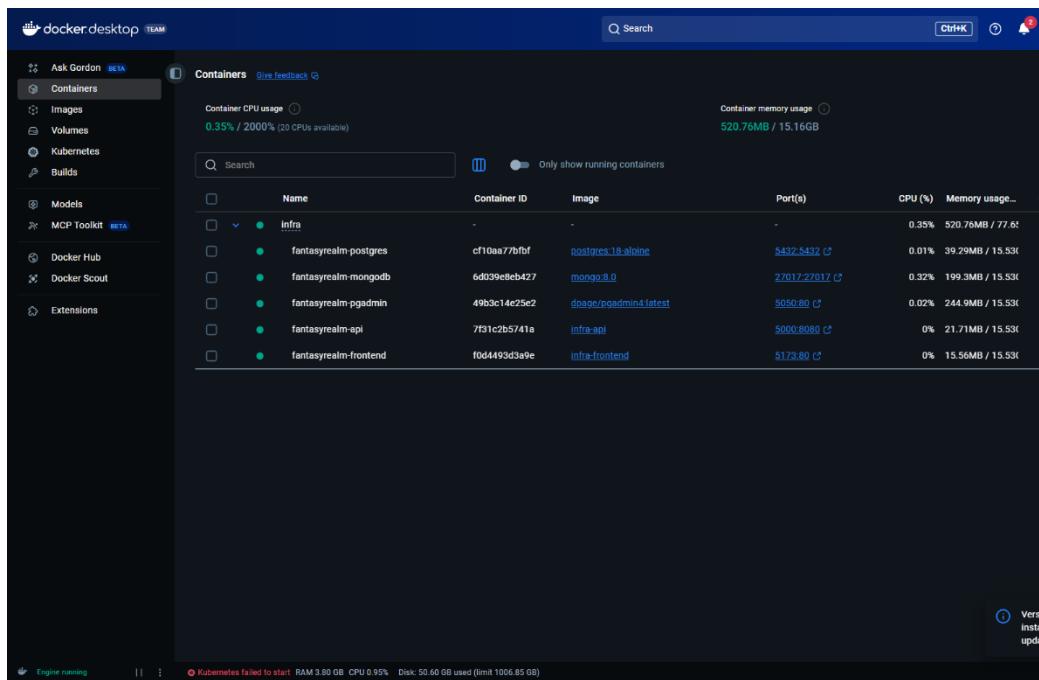
```
git clone https://github.com/pierrick-fonquerne/fantasyrealm-character-manager.git
cd fantasyrealm-character-manager
```

### 2.3. Lancer l'environnement

```
cd infra
docker compose up -d
```

Cela démarre 5 services :

Service	Port local	Description
postgre s	5432	PostgreSQL 18 Alpine
mongo db	27017	MongoDB 8.0
pgadmi n	5050	Interface d'administration PostgreSQL
api	5000	Backend .NET 8
fronten d	5173	Frontend React (Vite)



## 2.4. Accès

- Frontend : <http://localhost:5173>
- API : <http://localhost:5000>
- PgAdmin : <http://localhost:5050>
- Swagger API : <http://localhost:5000/swagger>

## 2.5. Arrêter l'environnement

```
docker compose down
```

Pour supprimer les volumes (réinitialiser les bases) :

```
docker compose down -v
```

## 3. Déploiement MongoDB Atlas

### 3.1. Créer un cluster

1. Se connecter sur <https://www.mongodb.com/atlas>
2. Créer un cluster **Shared** (gratuit)
3. Région : Europe (Paris)
4. Nom du cluster : fantasyrealmonline-cluster

The screenshot shows the MongoDB Atlas 'Deploy your cluster' interface. It displays three deployment options:

- M10**: \$0.09/hour. Dedicated cluster for development environments and low-traffic applications.
- Flex**: From \$0.01/hour. Up to \$30/month. For development and testing, with on-demand burst capacity for unpredictable traffic.
- Free**: For learning and exploring MongoDB in a cloud environment.

A note below the Free option states: "Free forever! Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime."

**Configurations**  
Name: fantasyrealmonline-cluster  
Provider: aws (selected)  
Region: (dropdown menu)  
Quick setup:  
 Automate security setup  
 Preload sample dataset

I'll do this later   Go to Advanced Configuration   Create Deployment

### 3.2. Configurer le réseau

1. **Network Access** > Add IP Address
2. Sélectionner **Allow Access from Anywhere** (0.0.0.0/0)
  - Nécessaire car Railway utilise des adresses IP dynamiques

### 3.3. Crée un utilisateur base de données

1. **Database Access** > Add New Database User
2. Username : fantasyrealm\_app
3. Password : générer un mot de passe fort
4. Role : readWriteAnyDatabase

### 3.4. Récupérer la connection string

1. **Clusters** > **Connect** > **Connect your application**
2. Driver : **.NET/C#**
3. Copier la connection string :

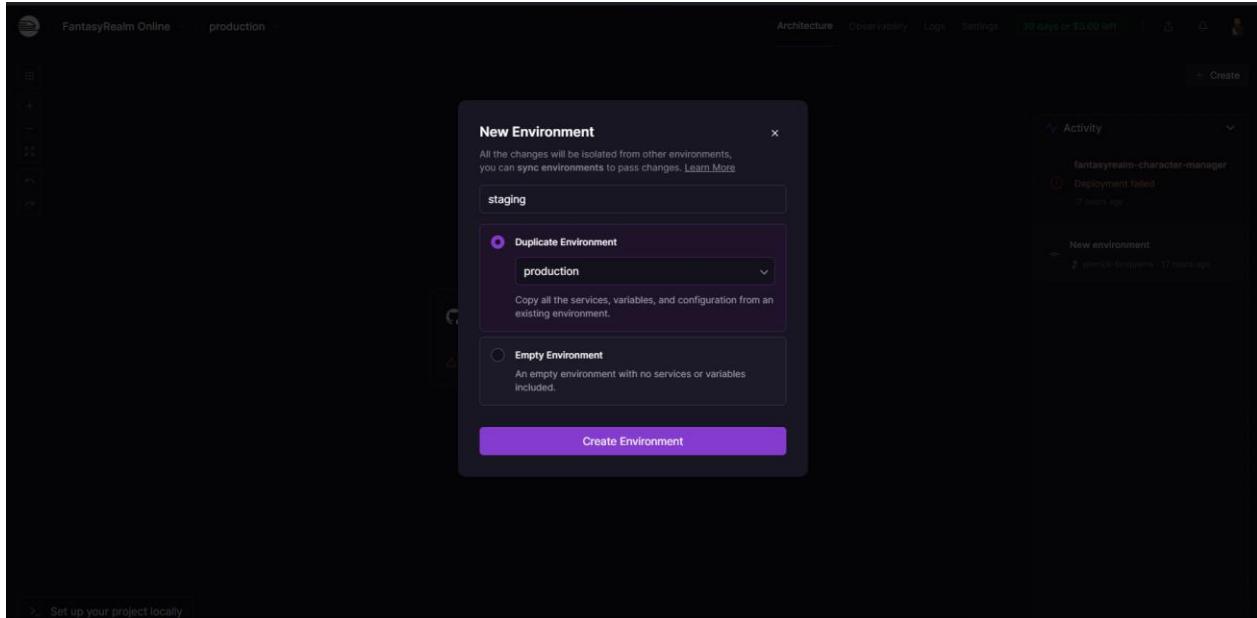
```
mongodb+srv://fantasyrealm_app:<password>@fantasyrealm.xxxxx.mongodb.net/fantasyrealm?retryWrites=true&w=majority
```

---

## 4. Déploiement Railway (Backend + PostgreSQL)

### 4.1. Créer le projet

1. Se connecter sur <https://railway.app> avec GitHub
2. **New Project > Deploy from GitHub repo**
3. Sélectionner le repository `fantasyrealm-character-manager`



### 4.2. Ajouter PostgreSQL

1. Dans le projet Railway > **+ New > Database > PostgreSQL**
2. Les variables de connexion sont générées automatiquement

### 4.3. Configurer le service Backend

**Settings :**

- Root Directory : `src/backend`
- Build Command : `dotnet publish src/FantasyRealm.Api -c Release -o out`
- Start Command : `cd out && dotnet FantasyRealm.Api.dll`

### 4.4. Variables d'environnement

Dans Railway > **Variables**, configurer :

```
# ASP.NET Core
ASPNETCORE_ENVIRONMENT=Production
ASPNETCORE_URLS=http://+:$PORT
```

```

# PostgreSQL (variable référencée Railway)
ConnectionStrings_PostgreSQL=${{Postgres.DATABASE_URL}}


# MongoDB Atlas
ConnectionStrings_MongoDB=mongodb+srv://user:pass@cluster.mongodb.net/fantasyrealm


# JWT
JwtSettings_Key=<clé-secrète-minimum-32-caractères>
JwtSettings_Issuer=FantasyRealm
JwtSettings_Audience=FantasyRealmUsers
JwtSettings_ExpireMinutes=60


# Brevo (emails transactionnels)
Brevo_ApiKey=<clé-api-brevo>
Brevo_SenderEmail=noreply@fantasy-realm.com
Brevo_SenderName=FantasyRealm
Brevo_AdminEmail=admin@fantasy-realm.com


# CORS
AllowedOrigins=https://www.fantasyrealm.com,https://*.vercel.app

```

#### 4.5. Domaine personnalisé (optionnel)

1. **Settings > Networking > Generate Domain**
  2. Résultat : fantasyrealm-api.up.railway.app
- 

### 5. Déploiement Vercel (Frontend)

#### 5.1. Connecter le repository

1. Se connecter sur <https://vercel.com> avec GitHub
2. **Import Project** > Sélectionner le repo

#### 5.2. Configuration

- **Root Directory** : src/frontend
- **Framework Preset** : Vite
- **Build Command** : npm run build
- **Output Directory** : dist

#### 5.3. Variables d'environnement

VITE\_API\_URL=https://fantasyrealm-api.up.railway.app/api

#### 5.4. Domaine personnalisé

1. **Settings > Domains**
2. Ajouter le domaine www.fantasyrealm.com

### 3. Configurer les DNS chez OVH (CNAME vers cname.vercel-dns.com)

The screenshot shows the 'Project Settings' page for a project named 'fantasyrealm-character-manager'. On the left, a sidebar lists various settings: General, Build and Deployment, Domains, Environments, Git, Integrations, Deployment Protection, Functions, Caches, Cron Jobs, Microfrontends, Project Members, Drains, Security, Connectivity, and Advanced. The 'Domains' section is currently selected. In the main area, there is a search bar and a table for managing domains. The first row shows 'fantasy-realm.com' with a status of 'Invalid Configuration' and a note about an IP range expansion. The second row shows 'fantasyrealm-character-manager.vercel.app' with a status of 'Valid Configuration'. Both rows have 'Production' and 'Edit' buttons.

This screenshot is from the same 'Project Settings' page as the previous one, but it shows a different state for the 'fantasy-realm.com' domain. The status is now 'Generating SSL Certificate'. The rest of the interface and the list of settings on the left remain the same.

---

## 6. Configuration Brevo (Emails)

### 6.1. Créer un compte

1. Se connecter sur <https://app.brevo.com>
2. Créer une clé API : **Settings > SMTP & API > API Keys**

### 6.2. Pourquoi Brevo ?

Railway bloque les ports SMTP (25, 465, 587) sur le plan gratuit. L'envoi d'emails via SMTP classique (OVH, Gmail) est impossible depuis Railway. Brevo propose une API HTTP pour l'envoi transactionnel, contournant cette limitation.

---

## 7. CI/CD — GitHub Actions

### 7.1. Pipeline CI (`ci.yml`)

Déclenché sur chaque push et pull request vers `main` et `develop`.

**Jobs :**

1. **Détection des changements** : identifie si le backend et/ou le frontend ont été modifiés
2. **Backend** (si modifié) :
  - o Setup .NET 8
  - o Restore, Build (Release), Test
3. **Frontend** (si modifié) :
  - o Setup Node 20
  - o npm ci, Lint, Type-check, Build, Test
4. **Vérification finale** : échoue si un des jobs a échoué

### 7.2. Protection de la branche main (`pr-main.yml`)

Vérifie que seule la branche `develop` peut être mergée dans `main`. Les PR provenant d'autres branches sont rejetées automatiquement.

### 7.3. Déploiement automatique

- **Railway** : déploie automatiquement à chaque push sur la branche configurée (`main` pour la production, `develop` pour la recette)
- **Vercel** : déploie automatiquement à chaque push. Les branches de feature génèrent des "preview deployments"

The screenshot shows a GitHub Pull Request page for the repository "fantasyrealm-character-manager". The pull request has been merged successfully, indicated by the green checkmark icon and the message "This branch was successfully deployed". The deployment status shows "1 active deployment" and "Deployed now by vercel[bot]". The "Checks" section indicates that some checks haven't completed yet, with two pending checks: "Backend (.NET)" and "Frontend (React)". Both are marked as "Expected" and "Waiting for status to be reported". There are also two successful checks: "Vercel" and "Vercel Preview Comments", both marked as "Completed". The "Comments" section shows a single comment from "vercel bot" about the latest updates. The "Files changed" section lists 25 files, mostly related to database schema and seed data, activity logs, and unit tests. The "Reviewers" and "Assignees" sections are empty. Notifications are customized, and there is an "Unsubscribe" button. The bottom of the page includes a comment input field, a "Comment" button, and a note about GitHub Community Guidelines.

## 8. Environnements

Environnement	Branche Git	Frontend	Backend
Production	main	www.fantasy-	Railway

Environnement	Branche Git	Frontend	Backend
<b>Recette</b>	develop	realm.com staging.fantasy-realm.com (Vercel preview)	(production) Railway (staging)
<b>Local</b>	XX-nom-de-la-fonctionnalite	localhost:5173	localhost:5000 (Docker)

---

## 9. Vérification post-déploiement

### Checklist

- Frontend accessible sur l'URL de production
- API répond sur /health
- Swagger accessible en mode Development
- Connexion PostgreSQL fonctionnelle (inscription/connexion OK)
- Connexion MongoDB fonctionnelle (logs d'activité OK)
- HTTPS actif sur les deux services
- CORS configuré pour le domaine frontend
- Envoi d'emails fonctionnel (inscription, modération)
- Compte admin accessible

### Tests de santé

```
# Test health endpoint
curl https://fantasyrealm-api.up.railway.app/health
```

---

## 10. Rollback

### Railway

1. **Deployments** > Sélectionner un déploiement précédent
2. Cliquer sur **Redeploy**

### Vercel

1. **Deployments** > Sélectionner un déploiement précédent
2. ... > **Promote to Production**

### Base de données

Railway effectue des sauvegardes automatiques de PostgreSQL. En cas de besoin, restaurer depuis la console Railway.