

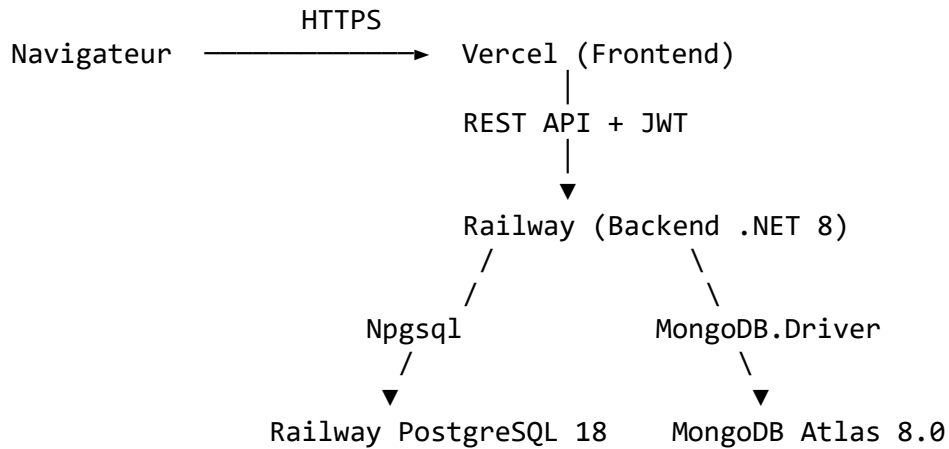
Dossier de Déploiement

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 (ci.yml).....	9
7.2. Protection de la branche main (pr-main.yml).....	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	https://www.fantasy-realm.com
Backend API	Railway	https://fantasyrealm-api-production.up.railway.app
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

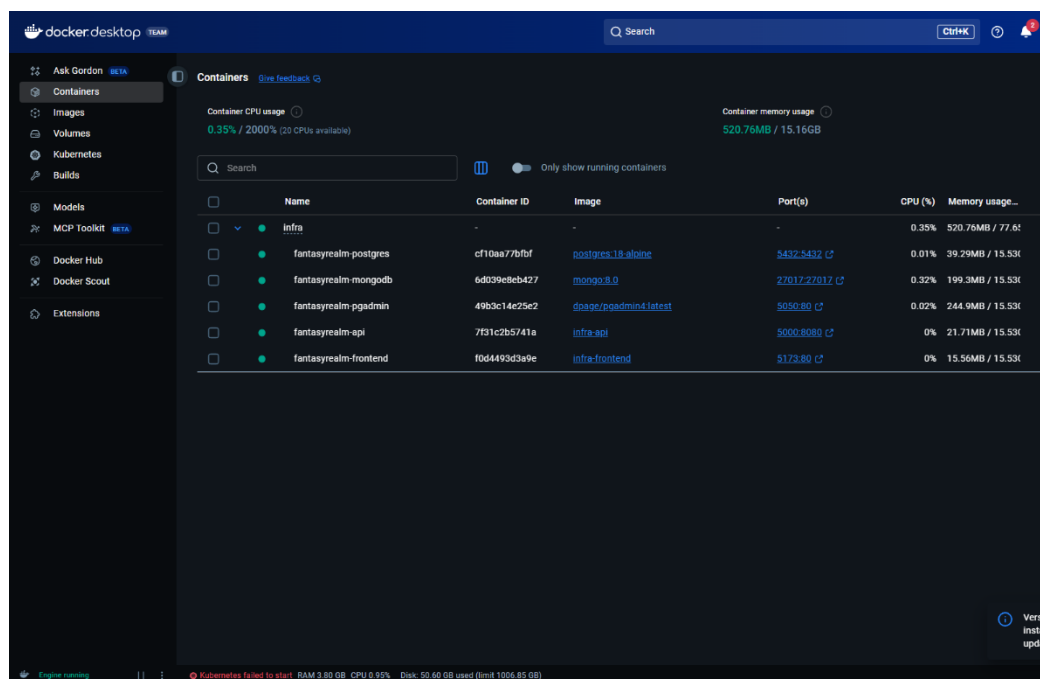
```
git clone https://github.com/pierrick-fonquerne/fantasyrealm-  
character-manager.git  
cd fantasyrealm-character-manager
```

2.3. Lancer l'environnement

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

Cela démarre 5 services :

Service	Port local	Description
postgres	5432	PostgreSQL 18 Alpine
mongodb	27017	MongoDB 8.0
pgadmin	5050	Interface d'administration PostgreSQL
api	5000	Backend .NET 8
frontend	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`

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

Option	Price	Description	Storage	RAM	vCPU
M10	\$0.09/hour	Dedicated cluster for development environments and low-traffic applications.	10 GB	2 GB	2 vCPUs
Flex	From \$0.011/hour (Up to \$30/month)	For development and testing, with on-demand burst capacity for unpredictable traffic.	5 GB	Shared	Shared
Free	Free	For learning and exploring MongoDB in a cloud environment.	512 MB	Shared	Shared

Free forever! Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Configurations

Name: You cannot change the name once the cluster is created.
fantasyrealmonline-cluster

Provider: ☒ AWS ☐ Google Cloud ☐ Azure

Region:

3.2. Configurer le réseau

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

3.3. Créer 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

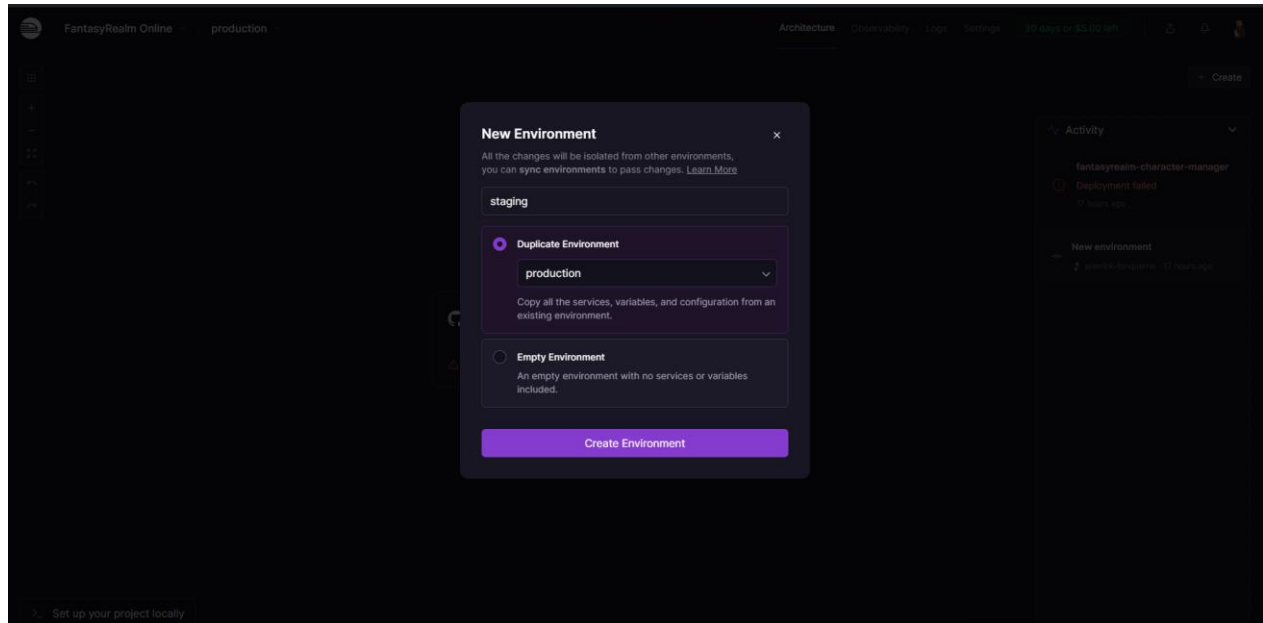
1. **Clusters** > **Connect** > **Connect your application**
2. Driver : **.NET/C#**
3. Copier la connection string :

`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.fantasy-realm.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.fantasy-realm.com

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

Project Settings
Go to Team Settings

Domains
Domains can be assigned to git branches, custom environments, and production.

fantasy-realm.com Production Refresh Edit

DNS Records Vercel DNS

The DNS records at your provider must match the following records to verify and connect your domain to Vercel.

Type	Name	Value
A	@	216.198.73

As part of a planned IP range expansion, you may notice new records above. The old records of `cname.vercel-dns.com` and `76.76.21.21` will continue to work but we recommend you use the new ones.

It might take some time for the DNS records to apply. [Learn More](#)

fantasyrealm-character-manager.vercel.app Production Refresh Edit

Valid Configuration

Project Settings
Go to Team Settings

Domains
Domains can be assigned to git branches, custom environments, and production.

fantasy-realm.com Production Refresh Edit

Generating SSL Certificate

fantasyrealm-character-manager.vercel.app Production Refresh Edit

Valid Configuration

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 :

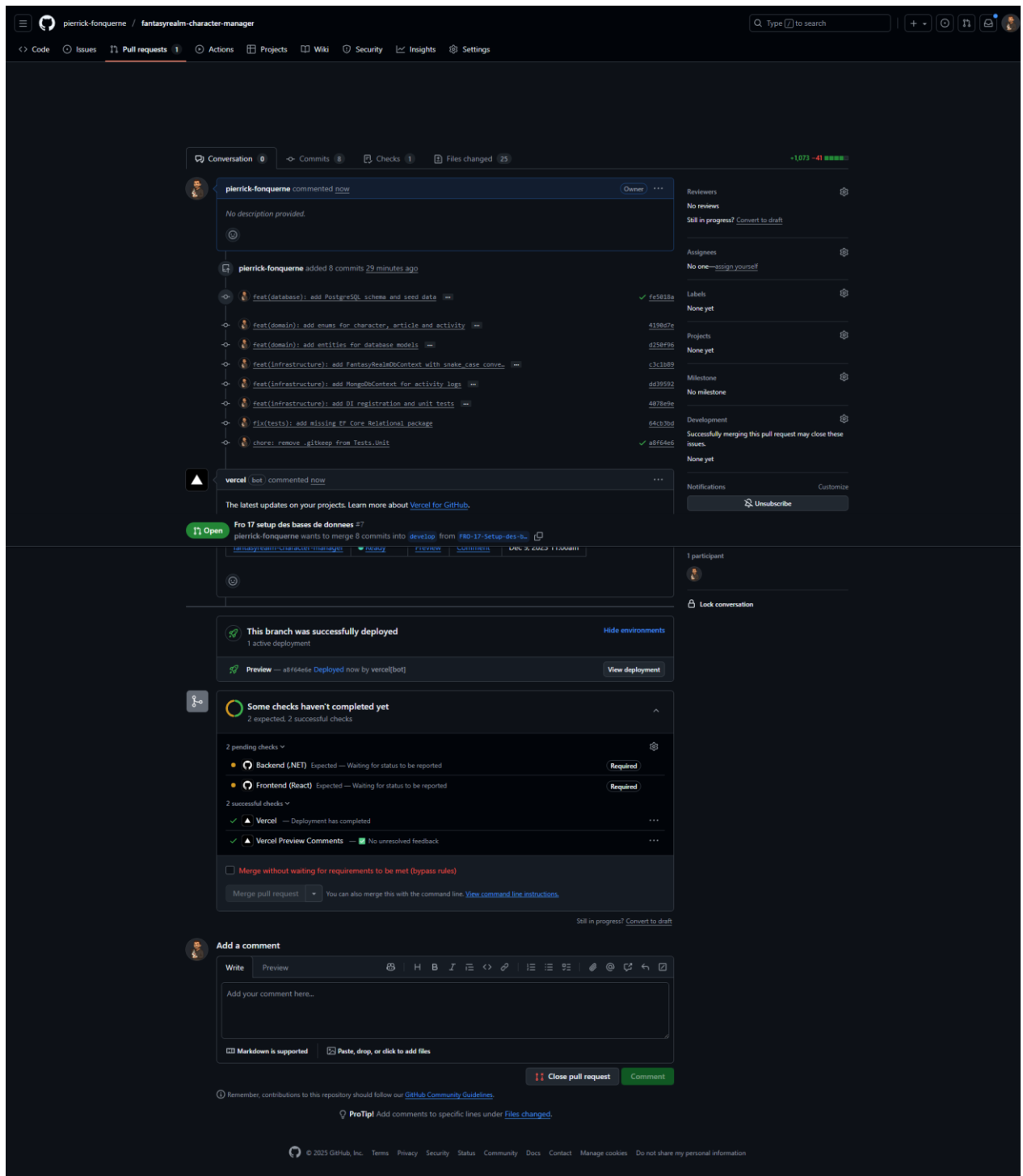
1. **Détection des changements** : identifie si le backend et/ou le frontend ont été modifiés
2. **Backend** (si modifié) :
 - Setup .NET 8
 - Restore, Build (Release), Test
3. **Frontend** (si modifié) :
 - Setup Node 20
 - npm ci, Lint, Type-check, Build, Test
4. **Vérification finale** : échoue si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



8. Environnements

Environnement

Branche Git

Frontend

Backend

Production

main

www.fantasy-

Railway

Environnement	Branche Git	Frontend	Backend
Recette	develop	realm.com staging.fantasy-realm.com (Vercel preview)	(production) Railway (staging)
Local	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.