Déploiement

- Pipeline CI/CD (Gitea)
- Reverse proxy (Nginx Proxy Manager)
- Rollback
- Déploiement Unraid

Pipeline CI/CD (Gitea)

Pipeline CI/CD (Gitea)

> *Dernière mise à jour : 2026-05-10*

Workflow `.gitea/workflows/build.yml`

87 lignes. Trigger : `push` sur `main`.

Job `test`

- **Runner** : `ubuntu-latest`
- **Container** : `node:22-bookworm-slim`
- **Pourquoi container custom** : Alpine (musl libc) ne supporte pas les prebuilts Node fournis par `actions/setup-node` (glibc). `better-sqlite3` n'a pas de prebuilt musl. → Debian Slim + Node 22 + tous les prebuilts.

Étapes

```
- Checkout code
- Backend : npm ci → npm run build (tsc) → npm test (vitest --passWithNoTests)
- Frontend : npm ci → npm run build (vue-tsc + vite) → npm test
```

Si l'un des steps fail, le job suivant (build) ne tourne pas.

Job `build` (depends: test)

- **Runner** : `ubuntu-latest`

Étapes

```
- Setup Docker Buildx
- Login registry:
    server: vars.REGISTRY_URL  # gitea.thymon.fr
    username: secrets.REGISTRY_USER
    password: secrets.REGISTRY_PASSWORD
- Metadata: tags `latest` + short SHA (`docker/metadata-action`)
- Build & push:
    - context: .
    - file: ./Dockerfile (multi-stage)
    - push: true
    - tags: gitea.thymon.fr/thymon/boardgame-referee:latest + :sha-XXXXXXX
    - cache-from / cache-to: type=registry (réutilise les couches précédentes)
```

Tags d'image

- **`latest`** : dernier commit sur main (utilisé en prod pour pull auto)
- **`sha-<7chars>`** : tag immuable par commit (utilisé pour rollback)

Variables Gitea

Vars (pas secret)

- `REGISTRY_URL` = `gitea.thymon.fr`

Secrets

- `REGISTRY_USER` = compte Gitea avec write sur le package registry
- `REGISTRY_PASSWORD` = token Gitea ou password

Configurés dans Gitea UI : Repo → Settings → Secrets → Actions secrets.

Local : reproduire le build CI

```
# Backend
npm ci
npm run build
npm test

# Frontend
cd frontend
npm ci
npm run build
npm test
```

Si tout passe en local mais foire en CI, c'est probablement un souci d'environnement (Node version, Alpine vs Debian, etc.). Toujours utiliser Node 22 en local.

Déclencheur manuel

Pas configuré actuellement (`workflow_dispatch` absent). Pour relancer un build sans push :

1. Soit faire un commit vide : `git commit --allow-empty -m "chore: rebuild"`
2. Soit re-run le job depuis l'UI Gitea (Actions → workflow run → Re-run jobs)

Latence typique

- Job `test` : ~3-5 min (npm ci + tsc + vitest)
- Job `build` : ~5-10 min (Docker multi-stage build + push)

Total : ~10-15 min par push sur main.

Pas d'environnement de staging

Push direct sur `main` → image `latest` → tu pulls sur Unraid. Pas de branche `develop`, pas de staging URL.

Si tu veux ajouter un staging :

1. Créer une branche `staging`
2. Étendre `build.yml` avec un trigger `push: branches: [main, staging]`
3. Tag différencié : `staging-latest` vs `latest`
4. Container Unraid `boardgame-referee-staging` séparé

Reverse proxy (Nginx Proxy Manager)

Reverse proxy (Nginx Proxy Manager)

> *Dernière mise à jour : 2026-05-10*

URL prod

- **Public** : https://rules.thymon.fr
- **LAN direct** : `http://192.168.10.100:3000` (utile pour debug si NPM tombe — dans la limite du `CORS_ORIGIN` configuré)

Config NPM

Côté UI NPM (pas dans le repo) :

- **Domain** : `rules.thymon.fr`
- **Forward Hostname / IP** : `boardgame-referee` (nom du container)
- **Forward Port** : `3000`
- **SSL** : Let's Encrypt (renew auto)
- **Force SSL** : oui (HTTP redirige vers HTTPS)
- **HSTS** : oui (dans Advanced)

NPM est sur le même réseau Docker `proxy` que le container app, donc il joint le container par son nom DNS interne.

`TRUST_PROXY=true`

Variable d'env importante : `TRUST_PROXY=true` dans `.env` prod. Hono fait alors confiance aux headers `X-Forwarded-*` de NPM :

- `X-Forwarded-For` → IP réelle du client (utilisé par le rate-limiter login)
- `X-Forwarded-Proto` → `https` (utilisé pour `Set-Cookie Secure`)
- `X-Forwarded-Host` → host original (`rules.thymon.fr`)

Heartbeats SSE 8s

⚠ Critique : les endpoints streamSSE (`/api/ask/stream`, `/api/admin/games/:id/sync-cards`, `/api/games/ingest` SSE) émettent un event `{ type: 'heartbeat' }` toutes les 8s.

Sans ça, le `proxy_read_timeout 60s` par défaut de Nginx fermerait la connexion alors que le backend travaille encore (Claude streamant lentement, par exemple). L'utilisateur verrait une fausse "network error" alors que la réponse arrive bien en base.

Côté frontend, `useEventStream` filtre déjà ces events via `skipHeartbeat`.

Si tu changes le `proxy_read_timeout` côté NPM (par ex. à 5min), tu peux espacer les heartbeats — mais 8s est un bon défaut robuste.

Pas de Helmet Hono

Tous les headers sécurité (HSTS, CSP, X-Frame-Options, X-Content-Type-Options, Referrer-Policy) sont gérés par NPM en amont via Advanced → Custom Nginx Configuration. Évite la duplication / contradiction.

Si tu veux les voir / vérifier : `curl -I https://rules.thymon.fr` doit montrer :

```
Strict-Transport-Security: max-age=...
Content-Security-Policy: ...
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Referrer-Policy: strict-origin-when-cross-origin
```

Backup config NPM

NPM stocke sa config dans `/mnt/user/appdata/nginx-proxy-manager/data/nginx/proxy_host/`. Backup régulier ces fichiers + `database.sqlite` au cas où le container NPM se corrompt.

Si NPM tombe

Le container `app` reste up et écoute sur `:3000` interne. Tu peux y accéder direct via LAN avec :

```
curl http://192.168.10.100:3000/api/health
```

Pour que le frontend fonctionne sans NPM en accès LAN, il faut que `CORS_ORIGIN` autorise l'IP/CIDR LAN (`192.168.10.0/24`).

Rollback

Rollback

> *Dernière mise à jour : 2026-05-10*

Stratégie

La CI Gitea push 2 tags par build :

- `latest` (mutable, suit toujours le dernier commit `main`)
- `sha-<7chars>` (immuable)

Pour revenir à une version antérieure : pull et utilise un tag `sha-<7chars>`.

Étapes

1. Identifier le commit cible

Sur Gitea : `gitea.thymon.fr/thymon/boardgame-referee/commits/branch/main`. Note le SHA court (7 premiers chars) du commit auquel tu veux revenir.

Ou en CLI :

```
git log --oneline -20
# 2ff9b71 security(ssh): isolate Claude SSH on dedicated 'oracle' user
# 95229af perf(rag): parallelize Qdrant searches + adopt withTimeout
# ...
```

2. Modifier docker-compose.yml

```
 services:
   app:
-    image: gitea.thymon.fr/thymon/boardgame-referee:latest
+    image: gitea.thymon.fr/thymon/boardgame-referee:sha-95229af
```

3. Pull + recreate

```
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml pull app
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml up -d --force-recreate app
```

4. Vérifier

```
docker logs -f --tail 50 boardgame-referee
curl https://rules.thymon.fr/api/health
```

Rollback BDD ?

⚠ Si la version cible a un schéma SQLite différent, rollback de l'image **ne rollback pas la BDD**. Si tu as joué une migration `0011_xxx.sql` puis tu rollback à une version qui ne la connaît pas :

- Au mieux : la migration est vue comme déjà appliquée (Drizzle skip) et l'app fonctionne mais avec des colonnes en trop dans la table → généralement pas de souci si la requête select * ne dépend pas du schéma exact.
- Au pire : du code récent référençait une nouvelle colonne que l'ancienne version ne lit pas → erreur silencieuse ou crash.

Backup SQLite avant migration :

```
docker exec boardgame-referee sqlite3 /app/data/database.db ".backup /app/data/database.db.bak.$(date +%F)"
# Puis avant rollback, restore :
docker exec boardgame-referee cp /app/data/database.db.bak.<date> /app/data/database.db
docker compose ... restart app
```

Rollback Qdrant ?

Qdrant ne fait pas de migrations — les payloads sont schemaless. Donc un rollback de l'image n'impacte pas Qdrant.

Si tu as ré-ingéré des PDFs avec une nouvelle version du chunker (qui produit des chunks différents), rollback de l'image ne rollback pas les chunks. Si nécessaire :

```
# Snapshot Qdrant AVANT migration (recommandé)
curl -X POST http://192.168.10.100:6333/collections/rules_<slug>/snapshots
# (renvoie un nom de snapshot)

# Restore après rollback
curl -X PUT http://192.168.10.100:6333/collections/rules_<slug>/snapshots/recover \
  -H "Content-Type: application/json" \
  -d '{"location": "<snapshot-name>"}'
```

Rollback NPM config ?

NPM versionne sa propre config dans `database.sqlite` (interne). Si tu as cassé une route ou l'TLS, restaure depuis ton backup `/mnt/user/appdata/nginx-proxy-manager/`.

En pratique

Pour boardgame-referee, le risque "rollback obligatoire" est faible :

- Pas de données utilisateur critiques (ce que tu perds = des questions/réponses RAG, pas des transactions)
- Schéma SQLite stable depuis quelques migrations
- Qdrant peut être ré-ingéré si besoin

Donc rollback simple = changer le tag d'image dans `docker-compose.yml` et redémarrer.

Déploiement Unraid

Déploiement Unraid

> *Dernière mise à jour : 2026-05-10*

Architecture des containers

```
unraid/
├── Container `boardgame-referee` (image gitea.thymon.fr/thymon/boardgame-referee:latest)
├── Container `qdrant` (image qdrant/qdrant)
├── Container TEI (text-embeddings-inference, GPU RTX 3060)
└── Container TEI Reranker (text-embeddings-reranker, GPU RTX 3060)
```

VM externe : `oracle` (Claude Code CLI), accédée via SSH.

docker-compose.yml (prod)

Fichier : `/mnt/user/appdata/boardgame-referee/docker-compose.yml`

Service `app`

```
app:
  image: gitea.thymon.fr/thymon/boardgame-referee:latest
  # Pas de port exposé sur l'hôte (reverse proxy NPM)
  env_file: .env
  volumes:
    - /mnt/user/appdata/boardgame-referee/data:/app/data
    - /mnt/user/appdata/boardgame-referee/pdfs:/app/pdfs
```

```
    - /mnt/user/appdata/boardgame-referee/ssh:/app/ssh:ro
  networks:
    - proxy   # réseau externe NPM
  healthcheck:
    test: ["CMD", "wget", "-qO-", "http://localhost:3000/api/health"]
    interval: 30s
    timeout: 5s
    retries: 3
    start_period: 10s
  restart: unless-stopped
```

Service `qdrant`

```
qdrant:
  image: qdrant/qdrant
  networks:
    - proxy
  volumes:
    - /mnt/user/appdata/boardgame-referee/qdrant:/qdrant/storage
  restart: unless-stopped
```

Réseau

`proxy` : réseau Docker externe, partagé avec NPM. Permet à NPM de joindre le container `app` sans exposer son port sur l'hôte.

Volumes (`/mnt/user/appdata/boardgame-referee/`)

Sous-dossier	Container	Mount	Contenu
`data/`	app	`/app/data`	SQLite DB, caches JSON (OCR, contextual, conflicts), data sources cartes, logs

Sous-dossier	Container	Mount	Contenu
`pdfs/`	app	`/app/pdfs`	PDFs uploadés + PNG rendus
`ssh/`	app	`/app/ssh:ro`	Clé privée ed25519 oracle (RO)
`qdrant/`	qdrant	`/qdrant/storage`	Collections Qdrant

Tous sur le pool ZFS / parity Unraid (selon ta config).

Variables d'env (UI Unraid)

Configurées via le template XML `unraid/boardgame-referee.xml` :

- L'admin Unraid voit chaque var avec son label, type, défaut, et description
- ⚠ **Toute nouvelle var dans `src/config.ts` doit être ajoutée au XML**, sinon l'UI Unraid ne la propose pas

Templates XML inclus

Fichier	Container
`unraid/boardgame-referee.xml`	App principale
`unraid/text-embeddings-inference.xml`	TEI bge-m3
`unraid/text-embeddings-reranker.xml`	TEI reranker

Mise à jour en prod

```
# 1. Pull la nouvelle image (CI a déjà push)
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml pull app

# 2. Recreate le container avec la nouvelle image
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml up -d --force-
recreate app

# 3. Tail les logs pour vérifier que ça boot bien
docker logs -f --tail 50 boardgame-referee
```

Ou via l'UI Unraid : Apps → boardgame-referee → Update / Force Update.

Healthcheck

Docker healthcheck : `GET /api/health` toutes les 30s. Si 3 échecs consécutifs → container marqué unhealthy. Tu peux configurer un script Unraid (ou Uptime Kuma) qui restart le container automatiquement sur unhealthy, mais actuellement c'est manuel.

Logs

- **Stdout / stderr Docker** : `docker logs boardgame-referee`
- **Fichier persistant** : `/app/data/logs/server.log` (rotation 50 Mo / 30j via `entrypoint.sh`)

Entrypoint

`entrypoint.sh` (dans l'image Docker) fait :

1. Fix des permissions sur `/app/data` + `/app/pdfs` (gosu PUID:PGID configurable)
2. Rotation log 50 Mo si dépassé
3. Purge archives > 30 jours
4. Lance `node dist/index.js` avec `tee -a /app/data/logs/server.log`

Pas de scaling horizontal

Single instance. SQLite + sessions en mémoire + `setTimeout` cron in-process empêchent le scaling. Si un jour tu veux plus :

- SQLite → Postgres
- Sessions in-memory → table SQL ou Redis
- `setTimeout` → BullMQ ou équivalent