Guide d'Installation et d'Exploitation — Jira Issue Extractor

Prérequis

Matériel / infrastructure

Ressource	Minimum	Recommandé (production)
CPU	2 vCPU	4 vCPU
RAM	2 Go	4 Go
Stockage	20 Go	50 Go+ (selon volume d'exports)
OS	Linux x86_64 / macOS	Linux x86_64

Logiciels requis

Logiciel	Version minimale	Notes
Docker	24.x	
Docker Compose	v2.x (`docker compose`)	
Accès réseau sortant	—	Vers l'instance Jira cible

Déploiement rapide (mode complet)

1. Récupérer le code source

```
git clone https://vincent@bitbucket.smartview.fr/scm/svsw/jira-plugin-issue-extractor.git
cd jira-plugin-issue-extractor
```

2. Créer le fichier d'environnement

```
cp .env.example .env
```

Éditez `.env` avec les valeurs appropriées (voir section Variables d'environnement).

3. Construire et démarrer

```
docker compose up -d --build
```

L'application est disponible sur `http://localhost` (ou le port configuré via `FRONTEND_PORT`).

4. Première connexion

Connectez-vous avec le compte administrateur créé automatiquement au démarrage :

- **Email** : valeur de `BOOTSTRAP_ADMIN_EMAIL` (défaut : `admin@example.com`)
- **Mot de passe** : valeur de `BOOTSTRAP_ADMIN_PASSWORD` (défaut : `admin`)

  Changez le mot de passe immédiatement après la première connexion.

Variables d'environnement

Backend (`backend`)

Variable	Défaut	Description
`DATABASE_URL`	`postgresql+asyncpg://...`	URL de connexion PostgreSQL
`JWT_SECRET_KEY`	`change-me-in-production`	Clé secrète JWT — **à changer impérativement**
`JWT_EXPIRE_HOURS`	`8`	Durée de vie des tokens JWT (heures)
`JIRA_EXTRACTOR_REDIS_URL`	_(vide)_	URL Redis (`redis://redis:6379/0`) — optionnel
`JIRA_EXTRACTOR_MINIO_ENDPOINT`	_(vide)_	Endpoint MinIO (ex. `minio:9000`) — optionnel
`JIRA_EXTRACTOR_MINIO_ACCESS_KEY`	_(vide)_	Clé d'accès MinIO
`JIRA_EXTRACTOR_MINIO_SECRET_KEY`	_(vide)_	Clé secrète MinIO
`JIRA_EXTRACTOR_MINIO_BUCKET`	`jira-exports`	Nom du bucket MinIO
`JIRA_EXTRACTOR_MINIO_SECURE`	`false`	TLS vers MinIO (`true`/`false`)
`JIRA_EXTRACTOR_ENABLE_SCHEDULER`	`false`	Active le scheduler dans le processus backend
`SMTP_HOST`	_(vide)_	Serveur SMTP (optionnel)

Variable	Défaut	Description
`SMTP_PORT`	`587`	Port SMTP
`SMTP_USER`	_(vide)_	Identifiant SMTP
`SMTP_PASSWORD`	_(vide)_	Mot de passe SMTP
`SMTP_FROM`	`noreply@jira-extractor.local`	Adresse expéditeur
`SMTP_USE_TLS`	`true`	Utiliser STARTTLS
`APP_BASE_URL`	`http://localhost:8080`	URL publique de l'application (liens dans les emails)
`BOOTSTRAP_ADMIN_EMAIL`	`admin@example.com`	Email du compte admin initial
`BOOTSTRAP_ADMIN_PASSWORD`	`admin`	Mot de passe du compte admin initial

PostgreSQL (`postgres`)

Variable	Défaut	Description
`POSTGRES_DB`	`jira_extractor`	Nom de la base de données
`POSTGRES_USER`	`jira_extractor`	Utilisateur PostgreSQL
`POSTGRES_PASSWORD`	`jira_extractor`	Mot de passe PostgreSQL

MinIO (`minio`)

Variable	Description
`MINIO_ROOT_USER`	Compte root MinIO
`MINIO_ROOT_PASSWORD`	Mot de passe root MinIO

Frontend (`frontend`)

Variable	Défaut	Description
`FRONTEND_PORT`	`80`	Port exposé sur l'hôte

Modes de déploiement

Mode complet (6 conteneurs) — recommandé en production

```
services: frontend, backend, scheduler, postgres, redis, minio
```

- Redis assure la propagation des événements SSE entre le backend et le scheduler.
- MinIO stocke les exports de manière persistante et partagée.
- Le scheduler s'exécute dans un conteneur isolé.

Mode simplifié (3 conteneurs)

```
services: frontend, backend, postgres
```

- Sans Redis : le SSE fonctionne via `asyncio.Queue` (mono-instance uniquement).
- Sans MinIO : les exports sont stockés sur le volume Docker `exports_data`.
- Le scheduler intégré au backend peut être activé via `JIRA_EXTRACTOR_ENABLE_SCHEDULER=true`.

Utilisez le fichier `docker-compose.simple.yml` si disponible, ou commentez les services `redis`, `minio` et `scheduler` dans `docker-compose.yml`.

Déploiement en mode développement

```
# Backend
cd backend
pip install -r requirements.txt
uvicorn backend.main:app --reload --port 8080

# Frontend (dans un autre terminal)
cd frontend
npm install
npm run dev  # Proxy vers localhost:8080 configuré dans vite.config.js
```

La base de données SQLite est utilisée automatiquement si `DATABASE_URL` n'est pas définie.

Mise à jour

```
git pull
docker compose build --no-cache backend frontend
docker compose up -d
```

> *Le flag `--no-cache` est recommandé pour s'assurer que les nouvelles versions du code source sont bien prises en compte.*

Les migrations de base de données sont exécutées automatiquement au démarrage du backend via Alembic.

Gestion des données

Volumes Docker

Volume	Contenu	Service
`postgres_data`	Base de données PostgreSQL	`postgres`
`minio_data`	Exports stockés sur MinIO	`minio`
`exports_data`	Exports locaux (sans MinIO)	`backend`

Sauvegarde PostgreSQL

```
# Dump
docker compose exec postgres pg_dump -U jira_extractor jira_extractor > backup_$(date +%Y%m%d).sql

# Restauration
docker compose exec -T postgres psql -U jira_extractor jira_extractor < backup_20260101.sql
```

Sauvegarde MinIO

Utilisez l'outil `mc` (MinIO Client) ou la console d'administration MinIO (`http://localhost:9001`) pour exporter le contenu du bucket.

```
mc alias set local http://localhost:9000 <access_key> <secret_key>
mc cp --recursive local/jira-exports ./backup-minio/
```

Surveillance et logs

Consulter les logs en temps réel

```
# Tous les services
docker compose logs -f

# Un service spécifique
docker compose logs -f backend
docker compose logs -f scheduler
```

Vérifier l'état des conteneurs

```
docker compose ps
```

Redémarrer un service

```
docker compose restart backend
```

Résolution de problèmes courants

L'application ne répond pas sur le port 80

1. Vérifiez que le conteneur `frontend` est bien démarré : `docker compose ps`
2. Vérifiez qu'aucun autre service n'utilise le port : `lsof -i :80`
3. Consultez les logs nginx : `docker compose logs frontend`

Le backend ne démarre pas

```
docker compose logs backend
```

Causes fréquentes :

- `DATABASE_URL` incorrect (mauvais host, port, mot de passe)
- PostgreSQL pas encore prêt au démarrage — Docker Compose attend normalement via `healthcheck`, mais un `docker compose restart backend` peut suffire

Les exports ne sont pas stockés sur MinIO

Vérifiez que :

- `JIRA_EXTRACTOR_MINIO_ENDPOINT` est défini et que MinIO est démarré
- Le bucket existe (créé automatiquement au démarrage normalement)
- Les credentials MinIO correspondent entre les variables backend et la configuration MinIO

Consultez les logs backend pour des messages `[minio]`.

Les emails d'invitation ne sont pas reçus

Si `SMTP_HOST` est vide, l'URL d'invitation est loguée dans la console backend :

```
[EMAIL] To: user@example.com
[EMAIL] Subject: Invitation à Jira Extractor
[EMAIL] Invite URL: http://localhost/?token=xxxx&purpose=invite
```

Si SMTP est configuré, vérifiez les logs backend pour des erreurs SMTP.

Un export affiche une erreur "route non trouvée" (404)

Après une mise à jour, reconstruisez les images avec `--no-cache` :

```
docker compose build --no-cache backend
docker compose up -d backend
```

Sécurité en production

Checklist minimale avant exposition publique

- [ ] Changer `JWT_SECRET_KEY` pour une valeur longue et aléatoire (`openssl rand -hex 32`)
- [ ] Changer le mot de passe du compte admin bootstrap (`BOOTSTRAP_ADMIN_PASSWORD`)
- [ ] Configurer HTTPS via un reverse proxy (nginx, Traefik, Caddy)
- [ ] Restreindre les ports exposés (PostgreSQL 5432, Redis 6379, MinIO 9000/9001 ne doivent pas être accessibles publiquement)
- [ ] Configurer SMTP pour les emails transactionnels
- [ ] Définir `APP_BASE_URL` avec l'URL publique HTTPS
- [ ] Configurer des sauvegardes régulières du volume PostgreSQL

Exemple de configuration nginx reverse proxy (HTTPS)

```
server {
    listen 443 ssl;
    server_name jira-extractor.example.com;

    ssl_certificate /etc/ssl/certs/jira-extractor.crt;
    ssl_certificate_key /etc/ssl/private/jira-extractor.key;

    location / {
        proxy_pass http://localhost:8081;  # Adapter au port FRONTEND_PORT
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # SSE — désactiver le buffering
        proxy_buffering off;
        proxy_read_timeout 3600s;
    }
}

server {
    listen 80;
    server_name jira-extractor.example.com;
    return 301 https://$host$request_uri;
```

```
}
```

Désinstallation

```
# Arrêter et supprimer les conteneurs
docker compose down

# Supprimer également les volumes (ATTENTION : supprime toutes les données)
docker compose down -v

# Supprimer les images
docker compose down --rmi all
```