# Reconnect Pro

Runbooks opérationnels concernant Reconnect Pro.

- RP - Déploiement en pré-production
- RP - Correction des vulnérabilités
- RP - Migration des dossiers usagers

# RP - Déploiement en pré-production

*Runbook, Mai 2025. Workflow de déploiement en pré-production, de la PR validée jusqu'à l'exécution du script Deployer.*

> Certaines étapes de ce workflow sont **temporaires** et seront remplacées par un déclenchement à distance via **workflow_dispatch GitHub Actions** (prévu dans un prochain sprint tech).

---

## Étape 1 - Merger la PR vers `dev`

Une fois la PR approuvée par les pairs, effectuer le merge dans `dev` via GitHub.

---

## Étape 2 - Prévenir l'équipe sur Slack

Envoyer un message dans le salon **#équipe_technico_santé** pour signaler le déploiement imminent :

> :fusée: Déploiement de RP en PP en cours ...

---

## Étape 3 - Mettre à jour l'environnement local

Basculer sur la branche `dev`, la mettre à jour et synchroniser les dépendances :

```
git checkout dev
git pull
composer install
symfony console cache:clear
```

---

## Étape 4 - Modifier temporairement `deploy.php` *(temporaire, uniquement pour Reconnect Pro)*

> **Ne pas committer cette modification.** Cette étape est vouée à disparaître avec l'arrivée du workflow_dispatch.

Dans `deploy.php` à la racine du projet, localiser la tâche `deploy:restart_workers` et adapter les lignes commentées pour cibler la pré-production :

- **Commenter** la ligne `sudo systemctl restart messenger_worker@*.service`
- **Décommenter** les 3 lignes suivantes :
  - `sudo systemctl restart messenger_worker_async@*.service`
  - `sudo systemctl restart messenger_worker_dmp@*.service`
  - `sudo systemctl restart messenger_worker_export@*.service`

**Voir l'état attendu du bloc dans deploy.php**

```
task('deploy:restart_workers', function () {
// run('sudo systemctl restart messenger_worker@*.service');
    run('sudo systemctl restart messenger_worker_async@*.service');
    run('sudo systemctl restart messenger_worker_dmp@*.service');
    run('sudo systemctl restart messenger_worker_export@*.service');
});
```

---

# Étape 5 - Lancer le déploiement

Exécuter le script de déploiement via :

```
make deploy-preprod
```

---

# Étape 6 - Valider le déploiement et rétablir `deploy.php`

Poster un message de confirmation dans le salon **#équipe_technico_santé** :

> “ Déploiement de RP en PP :fusée:

Puis **rétablir** `deploy.php` dans son état d'origine :

- **Décommenter** la ligne `sudo systemctl restart messenger_worker@*.service`
- **Commenter** les 3 lignes pré-production :
  - `sudo systemctl restart messenger_worker_async@*.service`
  - `sudo systemctl restart messenger_worker_dmp@*.service`
  - `sudo systemctl restart messenger_worker_export@*.service`

> **Ne pas committer cette modification.** Le fichier doit rester dans son état production pour tout déploiement ultérieur.

**Voir l'état attendu du bloc rétabli**

```
task('deploy:restart_workers', function () {
    run('sudo systemctl restart messenger_worker@*.service');
// run('sudo systemctl restart messenger_worker_async@*.service');
// run('sudo systemctl restart messenger_worker_dmp@*.service');
// run('sudo systemctl restart messenger_worker_export@*.service');
});
```

# RP - Correction des vulnérabilités

*Runbook, Mai 2026. Workflow de correction des vulnérabilités de dépendances détectées par Dependabot.*

# Contexte

Les vulnérabilités sont détectées automatiquement par **Dependabot** et remontent chaque matin sur Slack. Ce document décrit la procédure complète de triage et correction, de l'alerte au merge.

> Dependabot est configuré en **alerts only,** il ne crée pas de Pull Requests automatiques. Les corrections sont appliquées manuellement selon ce workflow.

---

# Convention de nommage

## Branche Git

Format : security/<package>-patch-dep<N>-<N>

Exemples :

security/phpspreadsheet-patch-dep5-9

security/axios-patch-dep10-22

---

## Trello - Ticket de suivi

---

---

**Etiquette**: <projet>(RP) <prioritaire> <sécurité> et potentiellement <Non testable PO>.

**Titre** :

Format : Sécurité - Patch <package> <v.actuelle> → <v.cible> (dep N-N)

Exemples:

Securité - Patch phpspreadsheet 3.10.0 → 3.10.5 (dep #5-9) [1]

Securité - Patch axios 1.15.0 to 1.16.0 (dep 10-22) [1]

**Description** : liens vers les alertes de sécurité Dependabot

Exemple:

Fixes Dependabot alerts 19-22

19 - https://github.com/re-connect/pro/security/dependabot/19

20 - https://github.com/re-connect/pro/security/dependabot/20

21 - https://github.com/re-connect/pro/security/dependabot/21

22 - https://github.com/re-connect/pro/security/dependabot/22

Et bien sur, powers-up git pour les branches et les PRs.

---

# Pull Request

Toujours 2 PRs, une vers `main` et une vers `dev` et attribuer au PRs le label <security> pour une meilleur lisibilité.

**Titre :**

Format: security(deps-<écosystème>): upgrade <package> <v.actuelle> to <v.cible>

Exemples:

security(deps-composer): upgrade phpspreadsheet 3.10.0 to 3.10.5

security(deps-yarn): upgrade axios 1.15.0 to 1.16.0

**Description** :

Exemple:

Fixes Dependabot alerts 10-22

22 - https://github.com/re-connect/pro/security/dependabot/22

21 - https://github.com/re-connect/pro/security/dependabot/21

20 - https://github.com/re-connect/pro/security/dependabot/20

19 - https://github.com/re-connect/pro/security/dependabot/19

18 - https://github.com/re-connect/pro/security/dependabot/18

17 - https://github.com/re-connect/pro/security/dependabot/17

16 - https://github.com/re-connect/pro/security/dependabot/16

15 - https://github.com/re-connect/pro/security/dependabot/15

14 - https://github.com/re-connect/pro/security/dependabot/14

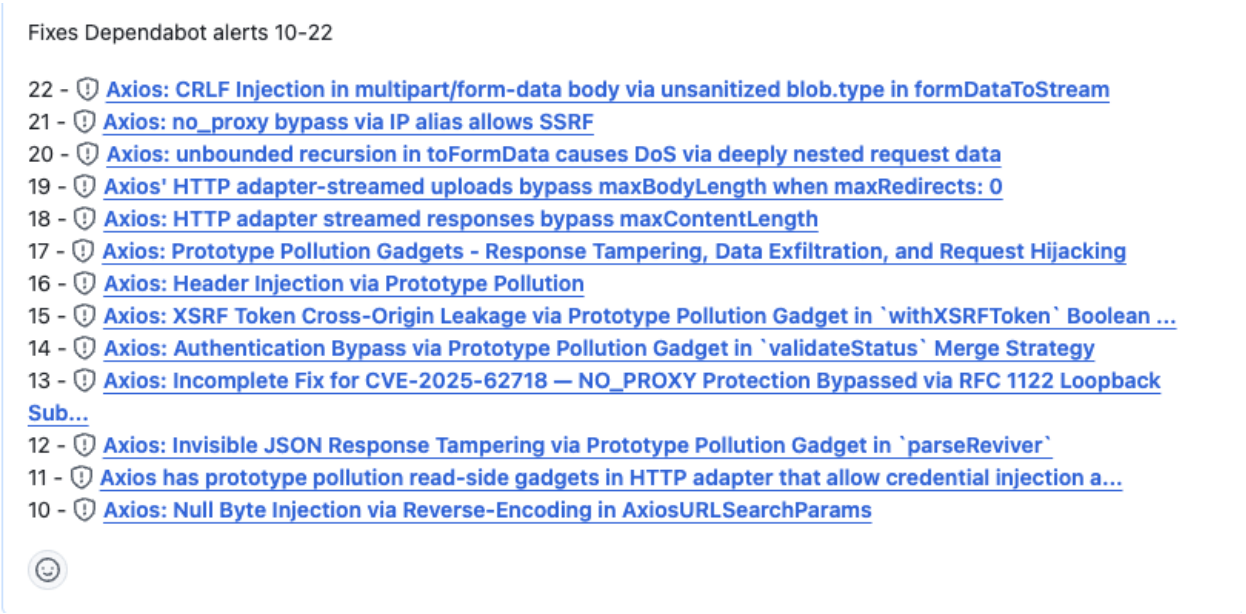
13 - https://github.com/re-connect/pro/security/dependabot/13

12 - https://github.com/re-connect/pro/security/dependabot/12

11 - https://github.com/re-connect/pro/security/dependabot/11

10 - https://github.com/re-connect/pro/security/dependabot/10

Preview:




---

# Procédure de triage

## 1. Évaluer le saut de version

| Saut | Risque | Action |
|---|---|---|
| patch `x.y.Z` | Nul | Update direct, pas de CHANGELOG à lire |
| minor `x.Y.z` | Faible | Lire le CHANGELOG, vérifier les releases notes |
| major `X.y.z` | Élevé | Lire le CHANGELOG, chercher une migration guide |

## 2. Grouper les alertes

- Même package + même écosystème → **un seul batch**
- Écosystèmes différents (npm / composer) → **branches séparées**
- Breaking change dans le batch → **commit séparé** dans la même branche

---

# Procédure de correction

# Composer

**Voir la procédure Composer**

1. Vérifier la version installée :

`composer show phpoffice/phpspreadsheet | grep versions`

2. S'assurer que `composer.json` utilise `^` et non une version exacte :

`"phpoffice/phpspreadsheet": "^3.10"`

3. Mettre à jour le package :

`composer update phpoffice/phpspreadsheet`

4. Valider :

`composer validate`

`composer audit`

5. Commiter :

`git add composer.json composer.lock`

`git commit -m "□□ security(deps-composer): upgrade <package> <v.actuelle> to <v.cible>"`

# npm / Yarn

**Voir la procédure Yarn**

1. Vérifier la version installée :

`yarn list axios`

2. Mettre à jour le package :

`yarn upgrade axios`

3. Valider :

`yarn audit`

4. Commiter :

`git add yarn.lock`

`git commit -m "□□ security(deps-npm): upgrade <package> <v.actuelle> to <v.cible>"`

> Si `yarn upgrade` monte en minor au lieu du patch indiqué par Dependabot, vérifier que la minor n'introduit pas de breaking change avant de commiter.

---

# Checklist avant merge

- CI verte
- `composer audit` ou `yarn audit` sans alerte bloquante
- 2 PRs ouvertes, `main` + `dev`
- Alertes Dependabot fermées après merge

# RP - Migration des dossiers usagers

*Runbook, Juin 2025. Workflow complet de traitement des tickets de migration de dossiers usagers sur Reconnect Pro.*

> Cette procédure s'applique à chaque ticket Trello de type « **Migration des dossiers usagers** ». Suivre les étapes dans l'ordre.

---

## La commande

La commande Symfony utilisée pour migrer les dossiers est :

```
php bin/console app:migrate-individuals-to-other-organization -o {ORIGINE_ID} -d {DESTINATION_ID} -f {CHAMP_MIGRATION} -t "{VALEUR_CIBLE}"
```

### Options disponibles

| Option | Nom long | Description | Exemple |
|---|---|---|---|
| `-o` | `--originOrganization` | ID du centre d'origine (celui qui perd des suivis) | `-o 4537` |
| `-d` | `--destinationOrganization` | ID du centre de destination (celui qui reçoit les suivis) | `-d 5644` |
| `-f` | `--targetFieldSlug` | Slug du champ personnalisé utilisé comme critère de migration | `-f migration-des-donnees` |
| `-t` | `--targetFieldValue` | Valeur du champ qui déclenche la migration (doit correspondre exactement à la valeur saisie dans le dossier) | `-t "CHU 77"` |

| | | | |
|---|---|---|---|
| `--noField` | - | Migrer *tous* les individus du centre d'origine, sans condition sur un champ. Incompatible avec `-f` / `-t`. | `--noField` |
| `--withDeleted` | - | Inclure les individus supprimés (soft-deleted) dans la migration | `--withDeleted` |
| `--withAccommodations` | - | Migrer également les hébergements liés aux individus migrés. Voir section dédiée ci-dessous. | `--withAccommodations` |
| `--dry-run` | - | Mode simulation : aucune donnée modifiée. **Toujours lancer en premier.** | `--dry-run` |

> **Logique de ciblage :** seuls les dossiers du centre d'origine dont le champ `-f` contient exactement la valeur `-t` sont migrés. Les dossiers avec le champ vide restent dans le centre initial. La valeur est recherchée d'abord sur le membre principal du foyer, puis sur les autres membres si non trouvée.

---

# Migration avec hébergements (`--withAccommodations`)

Lorsque l'option `--withAccommodations` est activée, la commande migre également les hébergements (`Accommodation`) liés aux individus migrés : l'hébergement est réaffecté au centre de destination.

> **Cas problématique - hébergement partagé :** si un hébergement contient à la fois des individus migrés et des individus *non* migrés (restant dans le centre d'origine), la commande le détecte et affiche un tableau d'alerte en dry-run. Ce cas nécessite une décision manuelle avant de lancer la migration réelle.

# Output dry-run avec hébergements

Le dry-run affiche, en plus du tableau des suivis, un récapitulatif des hébergements :

- Le nombre total d'hébergements qui seraient migrés
- Si des hébergements problématiques existent : un tableau listant pour chaque hébergement les individus migrés et les individus non migrés encore rattachés

Exemple de sortie en cas de problème :

Hébergements à migrer: 3

Problème: Hébergements avec individus non migrés rattachés

| Centre ID | Hébergement ID | Individus non migrés | Individus migrés |
|---|---|---|---|
| 4537 | 892 | 1042, 1078 | 1035 |

> En présence d'hébergements problématiques, **ne pas lancer la migration réelle** sans avoir tranché : soit on accepte de migrer l'hébergement (les individus non migrés se retrouveront sans hébergement dans le centre d'origine), soit on exclut `--withAccommodations` et on traite les hébergements manuellement.

## Comportement sans `--withAccommodations`

Par défaut (sans l'option), les individus migrés sont **décrochés de leur hébergement** avant migration, leur spot d'hébergement est réinitialisé. L'hébergement lui-même reste dans le centre d'origine.

---

# Procédure

## Étape 1 - Lire le ticket et construire les commandes

Le ticket Trello contient la liste des centres d'origine et leurs règles de migration. Pour chaque règle, construire une commande dédiée.

### Exemple de lecture d'un ticket

Ticket : *« Centre initial = France Fraternités - CHU 75 (ID 4537) - lorsque le champ = CHU 77 → vers France Fraternités - CHU 77 (ID 5644) »*

→ Commande correspondante (sans hébergements) :

```
php bin/console app:migrate-individuals-to-other-organization -o 4537 -d 5644 -f migration-des-donnees -t "CHU 77"
```

→ Commande correspondante (avec hébergements) :

```
php bin/console app:migrate-individuals-to-other-organization -o 4537 -d 5644 -f migration-des-donnees -t "CHU 77" --withAccommodations
```

> **Cas de croisement :** si deux centres d'origine s'échangent des suivis (ex : CHU 75 → CPH *et* CPH → CHU 75), traiter les deux groupes **séparément et séquentiellement,** terminer et vérifier le premier groupe avant de lancer le second. Ne jamais les lancer en parallèle.

# Étape 2 - Lancer les simulations (dry-run)

Avant toute migration réelle, lancer **toutes les commandes en mode `--dry-run`** sur la preprod pour vérifier les volumes. Ajouter `--withAccommodations` si le ticket concerne aussi les hébergements.

Exemple :

```
php bin/console app:migrate-individuals-to-other-organization -o 4537 -d 5644 -f migration-des-donnees -t "CHU 77" --dry-run
```

La commande affiche un tableau récapitulatif des suivis :

| Rôle | Centre ID | Centre Nom | Nb. suivis | Après migration |
|---|---|---|---|---|
| Origine | 4537 | France Fraternités - CHU 75 | 160 | 119 (- 41 migrés) |
| Destination | 5644 | France Fraternités - CHU 77 | 0 | 41 (+ 41 entrants) |

Si `--withAccommodations` est passé, un second bloc s'affiche avec le nombre d'hébergements impactés et les éventuels conflits. **Résoudre tous les conflits d'hébergement avant de continuer.**

Répéter pour chaque commande du ticket. Copier tous les outputs console.

# Étape 3 - Synthèse et validation PO

Une fois toutes les simulations effectuées (et les éventuels conflits d'hébergement résolus), transmettre les résultats au PO pour validation avant migration réelle. Deux formats possibles :

**Option A — Message Slack directe au PO**

Rédiger un message structuré avec les volumes par migration.

Centre d'origine : France Fraternités - CHU 75 (ID 4537) - 160 suivis au total

→ vers CHU 77 (ID 5644) : 41 migrés | reste 119
→ vers CPH (ID 4960) : 10 migrés | reste 150
→ vers DAHAR (ID 5634) : 19 migrés | reste 141
→ vers TEH (ID 5645) : 7 migrés | reste 153

Centre d'origine : France Fraternités - CPH (ID 4960) - 180 suivis au total

→ vers CHU 75 (ID 4537) : 21 migrés | reste 159
→ vers CHU 77 (ID 5644) : 37 migrés | reste 143
→ vers DAHAR (ID 5634) : 35 migrés | reste 145
→ vers TEH (ID 5645) : 17 migrés | reste 163

Hébergements : [X hébergements migrés — aucun conflit détecté / ou décrire les conflits et la décision prise]

**Option B - Excel de simulation**

Générer le fichier Excel de simulation nommé `simulation_migration_{NOM_CENTRE}.xlsx` et contient un tableau par centre d'origine avec décompte progressif.

Déposer le fichier dans le Drive à l'emplacement :

**Z-NEW RECONNECT > 4- Technique > Reconnect Pro > Migration des dossiers usagers**

Partager le lien Drive sur le ticket Trello et ping PO.

# Étape 4 - Mise à jour du ticket Trello

Une fois la validation PO reçue, coller dans le ticket Trello les commandes finales à exécuter **sans** `--dry-run` pour la preprod et la prod.

Format à utiliser dans le ticket :

Centre initial = [Nom centre] (ID XXXX)

Commande 1 :
php bin/console app:migrate-individuals-to-other-organization -o XXXX -d YYYY -f migration-des-donnees -t "VALEUR"

Commande 2 :
...

> Toujours exécuter et vérifier sur la **preprod en premier**, puis reproduire sur la prod. Ne jamais lancer directement en prod sans validation preprod.

> En cas de croisement entre deux centres, respecter l'ordre défini dans le ticket et attendre la fin du premier groupe avant de lancer le second, même en prod.