PORTFOLIO TECHNIQUE

Mise à niveau et sécurisation d’un coffre Vaultwarden auto-hébergé

Diagnostic d’une incompatibilité avec les extensions Bitwarden, sauvegarde contrôlée, migration Docker et validation du service.

Vaultwarden 1.35.8 → 1.37.1

PROBLÈME	MÉTHODE	STATUT
Extension : NetworkError	**Backup + contrôle SQLite**	**Serveur 1.37.1 healthy**

Adel Sadek

Administration systèmes / infrastructure - 12 août 2026

1. Contexte et diagnostic

Le coffre Vaultwarden auto-hébergé restait accessible depuis l'interface Web : je pouvais me connecter, consulter les entrées et afficher les mots de passe. En revanche, les extensions Bitwarden installées dans plusieurs navigateurs renvoyaient systématiquement « NetworkError when attempting to fetch resource ».

> **Symptôme clé**
>
> Le service Web fonctionnait, mais les clients navigateurs échouaient. Ce contraste orientait le diagnostic vers une incompatibilité de version côté API/client plutôt que vers une perte de données ou une panne générale du serveur.

Constat technique

Le conteneur Docker exécutait Vaultwarden 1.35.8 avec Web-Vault 2026.3.1. L'image avait été créée environ trois mois plus tôt sous le tag vaultwarden/server:latest. Le tag « latest » n'entraîne pas une mise à jour automatique d'un conteneur déjà créé : il faut récupérer la nouvelle image puis recréer le service.

Vérification de version

```
docker exec vaultwarden /vaultwarden --version
# Vaultwarden 1.35.8
# Web-Vault 2026.3.1
```

Hypothèse retenue

La version serveur était devenue trop ancienne pour les extensions Bitwarden récentes. La stratégie choisie a donc été une mise à niveau contrôlée vers Vaultwarden 1.37.1, avec sauvegarde complète et procédure de retour arrière avant toute migration.

> **Principe de sécurité**
>
> Je ne mets jamais à jour un gestionnaire de mots de passe sans pouvoir revenir à l'état précédent. Je protège à la fois l'image Docker utilisée et les données persistantes avant de démarrer la nouvelle version.

2. Préparation et sauvegarde

L’objectif de cette phase est de rendre l’opération réversible. Chaque commande est exécutée dans un ordre précis afin de limiter le risque sur le coffre.

1

Conserver l’ancienne image Docker

Pourquoi : Le tag de secours permet de garder exactement l’image 1.35.8 même après le téléchargement du nouveau latest.

```
docker tag vaultwarden/server:latest vaultwarden/server:pre-update-1.35.8
```

Résultat : Les tags latest et pre-update-1.35.8 pointaient alors vers le même IMAGE ID 137d93f83384.

2

Identifier le stockage persistant

Pourquoi : Je vérifie le montage /data pour savoir où se trouvent réellement la base et les fichiers du coffre.

```
docker inspect vaultwarden --format '{{range .Mounts}}{{.Source}} -> {{.Destination}}
{{println}}{{end}}'
```

Résultat : /root/vault/data -> /data

3

Arrêter proprement Vaultwarden

Pourquoi : Une base SQLite ne doit pas être copiée pendant qu’elle est modifiée. L’arrêt réduit le risque d’obtenir une sauvegarde incohérente.

```
cd /root/vault && docker compose stop vaultwarden
```

Résultat : Conteneur arrêté proprement en quelques secondes.

4

Créer une archive complète

Pourquoi : Je sauvegarde le docker-compose.yml et l’intégralité du dossier data dans une archive datée.

```
tar -czf /root/vault-backup-before-update-$(date +%Y%m%d-%H%M%S).tar.gz /root/vault
```

Résultat : Archive créée : 4,6 Mo.

3. Contrôles avant migration

Une sauvegarde n’est utile que si elle contient les fichiers attendus et si la base est saine. Je contrôle donc le contenu de l’archive avant de télécharger la nouvelle image.

5

Vérifier la présence de la base

Pourquoi : Je confirme que l’archive contient db.sqlite3 et ses fichiers auxiliaires.

```
tar -tzf /root/vault-backup-before-update-20260812-013623.tar.gz | grep 'db.sqlite3'
```

Résultat : db.sqlite3, db.sqlite3-wal et db.sqlite3-shm présents dans la sauvegarde.

6

Contrôler l’intégrité SQLite

Pourquoi : PRAGMA integrity_check vérifie la cohérence interne de la base sans en modifier le contenu.

```
sqlite3 /root/vault/data/db.sqlite3 "PRAGMA integrity_check;"
```

Résultat : Réponse : ok.

4. Mise à niveau de Vaultwarden

7

Télécharger la nouvelle image

Pourquoi : docker compose pull récupère la nouvelle image sans démarrer le service et sans toucher au volume de données.

```
docker compose pull vaultwarden
```

Résultat : Nouvelle image téléchargée avec succès.

8

Contrôler la version téléchargée

Pourquoi : Je vérifie la version de l’image avant de l’exécuter afin d’éviter une mise à jour non maîtrisée.

```
docker inspect vaultwarden/server:latest --format '{{ index .Config.Labels
"org.opencontainers.image.version" }}'
```

Résultat : Version détectée : 1.37.1.

9

Démarrer le nouveau conteneur

Pourquoi : Docker recrée le service avec l’image 1.37.1 tout en remontant le même /root/vault/data dans /data.

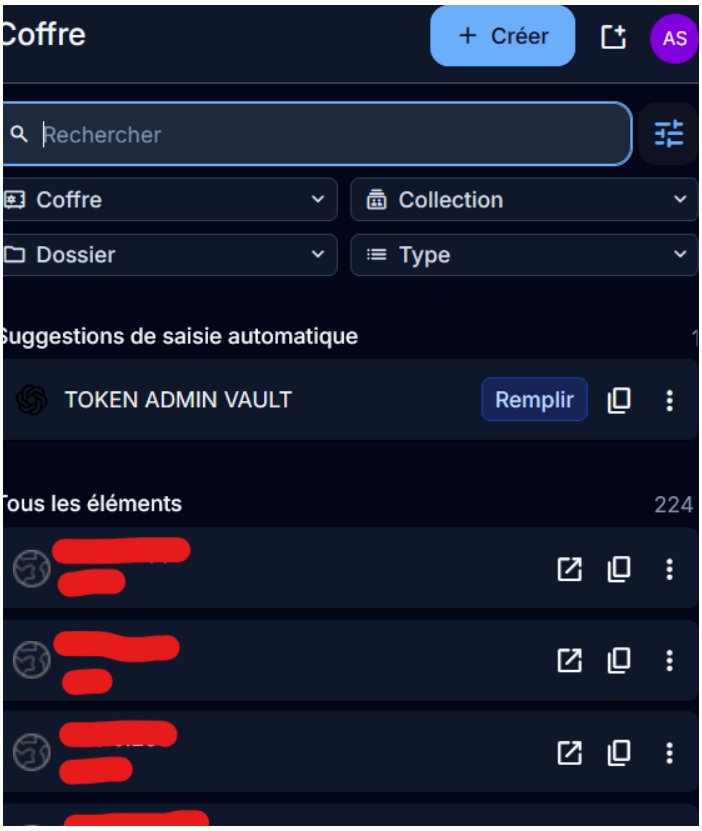
```
docker compose up -d vaultwarden
```

Résultat : Conteneur démarré avec succès.

5. Validation après mise à jour

Après le démarrage, je valide successivement la santé du conteneur, la version exécutée, les logs puis l'accès au coffre Web. Cette validation par couches évite de conclure trop vite qu'une migration est réussie.

État Docker	Up ... (healthy)
Version serveur	Vaultwarden 1.37.1
Web Vault	2026.6.4
Logs	Démarrage Rocket sans ERROR ni panic



Capture de validation du coffre Web après migration (éléments volontairement anonymisés).

> **Résultat obtenu**
>
> Le service Vaultwarden a été migré de 1.35.8 vers 1.37.1, le conteneur est healthy, la base est accessible et le coffre Web fonctionne après la migration.

6. Bilan et bonnes pratiques retenues

Cette intervention montre qu'une panne visible côté client ne signifie pas nécessairement que le serveur ou les données sont indisponibles. Le diagnostic a été guidé par la différence de comportement entre le Web Vault et les extensions, puis sécurisé par une procédure de mise à niveau réversible.

Ce que je retiens

- Toujours distinguer l'état du service Web, de l'API et des clients avant de conclure à une panne générale.
- Un conteneur utilisant le tag latest ne se met pas automatiquement à jour : il faut pull puis recréer le service.
- Avant une migration Vaultwarden, conserver l'ancienne image et sauvegarder le volume /data hors du conteneur.
- Pour SQLite, arrêter le service avant la copie et vérifier la base avec PRAGMA integrity_check.
- Valider une migration par plusieurs contrôles : version, healthcheck, logs, accès fonctionnel et tests clients.

Améliorations de sécurité à prévoir

Les logs signalent deux points qui ne bloquent pas le fonctionnement mais qui doivent être traités lors d'une prochaine maintenance : remplacer les anciennes variables SMTP_SSL / SMTP_EXPLICIT_TLS par SMTP_SECURITY et remplacer l'ADMIN_TOKEN stocké en clair par une chaîne Argon2 PHC. Ces actions seront réalisées séparément afin de ne pas mélanger une correction fonctionnelle avec un changement de configuration de sécurité.

> **Statut à la fin de cette intervention**
>
> Vaultwarden 1.37.1 est exécuté en production, le conteneur est healthy et le coffre Web est validé. Le test final des extensions Bitwarden est à réaliser après reconnexion au serveur auto-hébergé.

Document réalisé par

Adel Sadek

Administration systèmes / Infrastructure / Cybersécurité