Affine

AFFiNE est une plateforme de travail tout-en-un qui fusionne les documents, les tableaux blancs et les bases de données.

- Mise à jour (AFFiNE Self-Hosted)
- Configuration Affine AI (OpenAI Override)

Mise à jour (AFFiNE Self-Hosted)

? Étape 0 — Sauvegarde (très important)

Avant toute chose :

- Sauvegarde ton fichier `compose.yml`
- Sauvegarde tes données importantes (par exemple : volumes Docker, fichiers de config, base de données, etc.)

> En cas de souci pendant la mise à jour, tu pourras revenir en arrière sans perte.

? Étape 1 — Télécharger le nouveau `compose.yml`

1. Va sur la page officielle pour récupérer le **dernier fichier `compose.yml`**.
2. Remplace ton fichier existant (`compose.yml`) par celui que tu viens de télécharger.

⚠ **Attention** :
Si tu avais modifié ton ancien fichier (`ports`, `volumes`, variables d'environnement, etc.), **copie ces changements** dans le nouveau fichier.

? Étape 2 — Récupérer la dernière image Docker

Dans le terminal (dans le dossier où se trouve ton `compose.yml`) :

```
docker compose -f compose.yml pull
```

> Cette commande va télécharger la nouvelle version de l'image AFFiNE depuis Docker Hub.

□□ Étape 3 — Redémarrer le service

Toujours dans le terminal :

```
docker compose -f compose.yml up
```

>

> Cela va arrêter l’ancienne version (si elle tourne) et démarrer AFFiNE avec la **dernière version**.

? Mise à jour terminée !

Tu as maintenant la **dernière version de AFFiNE** en local, avec tous tes réglages conservés (si tu les as bien remis dans le nouveau fichier).

Configuration Affine AI (OpenAI Override)

1. Contexte et Problématique

Par défaut, les versions self-hosted d'Affine (v0.18+) sont configurées pour utiliser des modèles Google Gemini spécifiques. L'objectif est de forcer l'utilisation d'OpenAI (GPT-4o) pour tous les scénarios IA en contournant les limitations hardcodées du backend.

2. Configuration Infrastructure (Docker Compose)

La variable d'environnement `selfhosted=true` est **obligatoire** pour débloquer le menu d'administration IA et autoriser les API tiers.

Modification du fichier `docker-compose.yml`

Modifier le service `affine` pour ajouter les variables et fixer le montage du fichier de configuration dans un répertoire neutre (`/opt`) pour éviter les problèmes de permissions `root` vs `node`.

```
services:
  affine:
    image: ghcr.io/toeverything/affine:stable
    environment:
      # ... variables existantes (DB, Redis) ...
      - AFFINE_COPILOT_ENABLED=true

      # CRITIQUE : Active le support des API tiers et l'interface Admin AI
      - selfhosted=true

      # Force le chemin de configuration (évite les erreurs de dossier home)
      - AFFINE_CONFIG_PATH=/opt/affine_config.json

    volumes:
      - ${UPLOAD_LOCATION}:/root/.affine/self-host/storage
```

```
      # Montage dans /opt pour contourner les restrictions de permissions /root
      - ${CONFIG_LOCATION}:/opt/affine_config.json
```

> *Note : Assurez-vous que `${CONFIG_LOCATION}` dans le fichier `.env` pointe bien vers votre fichier json sur l'hôte.*

3. Configuration Initiale (Fichier JSON)

Ce fichier sert uniquement à déclarer le fournisseur. La configuration fine des modèles se fera via l'interface web.

Fichier sur l'hôte : `config.json`

```
{
  "copilot": {
    "enabled": true,
    "providers": {
      "openai": {
        "apiKey": "sk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
      }
    },
    "embedding": {
      "provider": "openai",
      "model": "text-embedding-3-small",
      "apiKey": "sk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
    }
  }
}
```

> *Note : Laisser les permissions du fichier à `644` pour qu'il soit lisible par le conteneur.*

4. Configuration Finale (Interface Web Admin)

C'est la méthode recommandée ("Plan B") pour assurer la persistance et mapper correctement les modèles internes d'Affine vers OpenAI.

1. Se connecter à l'instance Affine.

2. Accéder à l'URL : `https://votre-domaine-affine.com/admin` (ou via le menu **Settings > Admin Panel**).
3. Aller dans l'onglet **AI** (ou Intelligence).
4. Activer l'option **Enable Copilot**.

Mapping des Scénarios (Surcharge)

Dans la section de configuration des scénarios (Custom Models), coller le JSON suivant. **Important :** Cette configuration redirige toutes les requêtes internes (qui demandent par défaut `gemini-2.5-flash`) vers `gpt-4o`.

```
{
  "scenarios": {
    "chat": "gpt-4o",
    "chat_stream": "gpt-4o",
    "writing": "gpt-4o",
    "translation": "gpt-4o",
    "summary": "gpt-4o",
    "brainstorm": "gpt-4o",
    "coding": "gpt-4o",
    "check_grammar": "gpt-4o",
    "quick_decision_making": "gpt-4o",
    "quick_text_generation": "gpt-4o",
    "complex_text_generation": "gpt-4o",
    "polish_and_summarize": "gpt-4o",
    "audio_transcribing": "whisper-1",
    "image": "dall-e-3"
  },
  "override_enabled": true
}
```

Vérification OpenAI Provider

Juste en dessous, dans la section **OpenAI Provider**, vérifier que la clé API est bien détectée (masquée). Si nécessaire, forcer la configuration :

```
{
  "apiKey": "sk-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

5. Dépannage Rapide

Si l'IA ne répond pas (Erreur 500 ou "No provider available") :

1. Vérifier les logs : `docker compose logs -f affine`
2. Si erreur `ZodError` : Le fichier JSON est mal formé ou le modèle retourné par l'API n'est pas une liste valide.
3. Si erreur `Permission denied` : Vérifier que le montage Docker n'est pas dans `/root` côté conteneur.
4. Si les menus Admin sont invisibles : Vérifier que `selfhosted=true` est bien chargé dans les variables d'environnement.