Compte Rendu de l’Installation et de la Configuration d’Ollama et OpenWebUI

---

1. Installation et configuration d’Ollama sur le premier serveur

1.1. Installation d’Ollama

Sur le premier serveur, Ollama a été installé pour exécuter des modèles d’intelligence artificielle en local.

- **Téléchargement et installation d’Ollama** :

  ```
  curl -fsSL https://ollama.com/install.sh | sh
  ```

  Cela installe Ollama sur le serveur.
- **Vérification de l’installation** :

  ```
  ollama --version
  ```

1.2. Déploiement du modèle Mistral 8x7 sur Ollama

- **Téléchargement et ajout du modèle Mistral 8x7** :

  ```
  ollama pull mistral:8x7
  ```

  Ce modèle est désormais disponible localement pour être utilisé par Ollama.
- **Test du modèle pour vérifier son bon fonctionnement** :

  ```
  ollama run mistral:8x7 "Bonjour, peux-tu me donner un résumé de tes capacités ?"
  ```

Le premier serveur est maintenant prêt à fournir des services IA basés sur Mistral 8x7 via Ollama.

---

2. Installation et configuration d’OpenWebUI sur un second serveur

---

1. Prérequis du Système

Avant de commencer la configuration de l'environnement de développement local pour Open WebUI, il est important de vérifier que ton système remplit les prérequis nécessaires :

- **Système d'exploitation** : Linux (ou WSL sur Windows) ou macOS
- **Version Python** : 3.11 ou supérieure
- **Version Node.js** : 22.10 ou supérieure

---

2. Mise en Place du Dépôt et Installation des Composants Frontend et Backend

2.1. Clonage du Dépôt GitHub

Commence par cloner le dépôt **Open WebUI** depuis GitHub et déplace-toi dans le répertoire du projet :

```
git clone https://github.com/open-webui/open-webui.git
cd open-webui
```

2.2. Installation du Frontend

2.2.1. Création du Fichier `.env`

Crée un fichier `.env` pour configurer les variables d'environnement nécessaires pour le frontend :

```
cp -RPp .env.example .env
```

2.2.2. Installation des Dépendances Frontend

Ensuite, installe les dépendances frontend avec **npm** :

```
npm install
```

2.2.3. Lancement du Serveur Frontend

Pour démarrer le serveur frontend en mode développement, utilise la commande suivante :

```
npm run dev
```

Le frontend sera disponible à l'adresse suivante :
http://localhost:5173

2.3. Installation du Backend

2.3.1. Accéder au Répertoire Backend

Navigue vers le répertoire du backend pour commencer la configuration de l'environnement :

```
cd backend
```

2.3.2. Pourquoi utiliser Conda ?

Conda est un gestionnaire de paquets et un gestionnaire d'environnements virtuels qui permet de :

- **Isoler les environnements** : Conda permet de créer des environnements virtuels distincts pour chaque projet, garantissant ainsi que les dépendances spécifiques à un projet ne sont pas mélangées avec celles d'autres projets.
- **Facilité de gestion des dépendances** : Conda simplifie l'installation et la gestion des paquets nécessaires au backend, comme **Python** et ses bibliothèques.
- **Reproductibilité** : En utilisant Conda, tu peux facilement partager et reproduire l'environnement de développement exact sur d'autres machines en partageant simplement un fichier `environment.yml`.

2.3.3. Configuration de l'Environnement avec Conda

Pour configurer l'environnement pour le backend avec Conda, crée un nouvel environnement virtuel dédié au projet :

```
conda create --name open-webui python=3.11
conda activate open-webui
```

Cette commande crée et active un environnement Conda avec Python 3.11. Cela permet d'isoler les dépendances Python et d'éviter tout conflit avec d'autres projets ou environnements.

2.3.4. Installation des Dépendances Backend

Une fois l'environnement activé, installe les dépendances du backend via **pip** :

```
pip install -r requirements.txt -U
```

2.3.5. Lancement du Serveur Backend

Pour démarrer le serveur backend en mode développement, exécute le script `dev.sh` :

```
sh dev.sh
```

Le backend sera accessible à l'adresse suivante :
http://localhost:8080/docs (pour consulter la documentation de l'API).

---

3. Dépannage Courant

3.1. Erreur Fatale : Reached Heap Limit

Si tu rencontres des erreurs liées à la mémoire, tu peux augmenter la taille du tas mémoire allouée à **Node.js** dans le fichier **Dockerfile** :

```
ENV NODE_OPTIONS=--max-old-space-size=4096
```

Cela permettra d'allouer 4 Go de RAM à Node.js. Assure-toi que ta machine a suffisamment de mémoire disponible.

3.2. Conflits de Ports

Vérifie qu'aucune autre application n'utilise les ports **8080** (pour le backend) ou **5173** (pour le frontend).

3.3. Le Rechargement Automatique (Hot Reload) ne Fonctionne Pas

Assure-toi que le mode de surveillance est activé sur le **frontend** et le **backend** pour permettre un rechargement automatique des modifications sans redémarrer les serveurs manuellement.

---

4. Contribuer à Open WebUI

4.1. Flux de travail local

- **Commits réguliers** : Commits tes changements fréquemment pour garder une trace de l'évolution du projet.
- **Synchronisation avec la branche principale** : Avant de pousser des modifications, assure-toi de récupérer les dernières mises à jour du dépôt :

```
git pull origin main
```

- **Tests** : Avant de pousser des modifications sur la branche principale, exécute des tests pour garantir que ton code ne casse rien :

```
npm run test
```

---

4. Mise en place de l'authentification LDAP pour les utilisateurs

4.1. Installation des outils LDAP

- Installation des paquets nécessaires pour l'intégration LDAP :

```
apt install -y ldap-utils libpam-ldap libnss-ldap
```

4.2. Configuration d'OpenWebUI pour LDAP

- **Modification du fichier de configuration pour activer LDAP** :

```
{
  "auth": {
    "ldap": {
      "enabled": true,
      "server": "ldap://adresse-du-serveur-ldap",
      "base_dn": "dc=entreprise,dc=com",
      "bind_dn": "cn=admin,dc=entreprise,dc=com",
      "bind_password": "motdepasse",
      "user_filter": "(uid={username})"
    }
  }
}
```

  Ce fichier permet à OpenWebUI d'authentifier les utilisateurs en fonction des identifiants présents dans l'annuaire LDAP de l'entreprise.

4.3. Tests et validation

- **Test de connexion avec un utilisateur LDAP** :
  - Tentative de connexion sur OpenWebUI avec un compte existant dans LDAP.
  - Vérification que l'utilisateur est bien authentifié et peut accéder à l'interface.

---

5. État final et points de vérification

Composant	Statut	Remarque
Ollama installé	✅ Oui	Mistral 8x7 est fonctionnel
OpenWebUI installé	✅ Oui	Backend Python et frontend React opérationnels
Ollama intégré à OpenWebUI	✅ Oui	OpenWebUI envoie bien des requêtes à Ollama
Authentification LDAP	✅ Oui	Connexion des utilisateurs via LDAP réussie

---

6. Prochaines étapes

- **Optimisation des performances d'Ollama** en configurant la mémoire et les ressources CPU/GPU.
- **Mise en place d'un certificat SSL** pour sécuriser les connexions.
- **Ajout d'une gestion des rôles utilisateurs** dans OpenWebUI via LDAP pour différencier les droits d'accès.