Guide Complet - Workflow RAG Agent avec n8n et Qdrant

Ahmed Aziz Ammar

Table des matières

1 Introduction

Ce guide détaille les étapes pour installer et exécuter un workflow RAG (Retrieval-Augmented Generation) en local avec une base de données vectorielle Qdrant, en utilisant **n8n**. Cette approche permet d'ingérer automatiquement des fichiers PDF, Java et Markdown, de les transformer en embeddings vectoriels et de créer un système de recherche sémantique avancé.

2 Pré-requis

Avant de commencer, assurez-vous d'avoir les éléments suivants sur votre machine :
— GIT installé
— Espace 2.7 Go pour Docker Desktop
— Espace 4.5 Go pour container self-hosted-ai-starter-kit
— Terminal ou Invite de commande
— Clé API OpenAI (pour les embeddings et le modèle de chat)

3 Installation de Docker Desktop

1. Télécharger Docker Desktop à partir du lien officiel : https://www.docker.com/products/docker-desktop
2. Installer Docker Desktop en suivant les instructions pour Windows.
3. Vérifier que Docker est opérationnel avec la commande :

```
docker --version
```

4 Installation de n8n en mode Self-Hosted avec Docker

1. Cloner le dépôt officiel `self-hosted-ai-starter-kit` :
   git clone https ://github.com/n8n-io/self-hosted-ai-starter-kit.git

   Vérifiez s'il y a un espace entre "https" et " :" — si oui, concaténez-les.
2. Accéder au répertoire cloné :
   cd self-hosted-ai-starter-kit
3. Copier le fichier d'exemple `.env` et le renommer :
   copy .env.example .env
4. Lancer l'installation avec Docker Compose :

```
docker-compose --profile cpu up
```

   Vérifiez s'il y a un espace entre "–" et "profile" — si oui, concaténez-les.
5. Consulter l'emplacement du dossier `self-hosted-ai-starter-kit` sur votre machine. Nous avons besoin du fichier `.env` pour la configuration de PostgreSQL et du dossier `shared` pour l'emplacement des ressources.

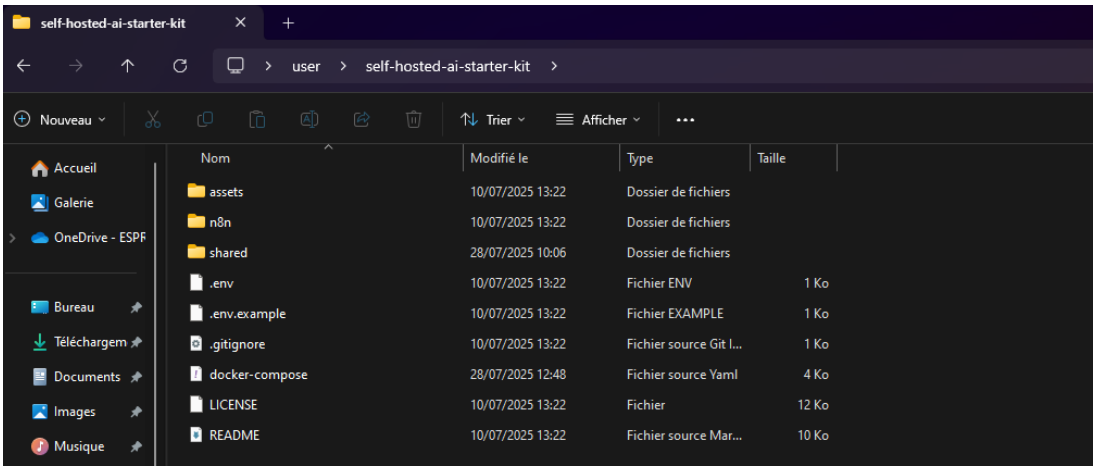

FIGURE 1 – Structure du dossier self-hosted-ai-starter-kit

5 Accès à l'Interface n8n

1. Après l'installation, ouvrir Docker Desktop et vérifier que le conteneur `self-hosted-ai-starter-` est en cours d'exécution.
2. Cliquer sur le lien généré : `5678:5678` pour accéder à l'interface web ou consulter le lien : `http://localhost:5678`

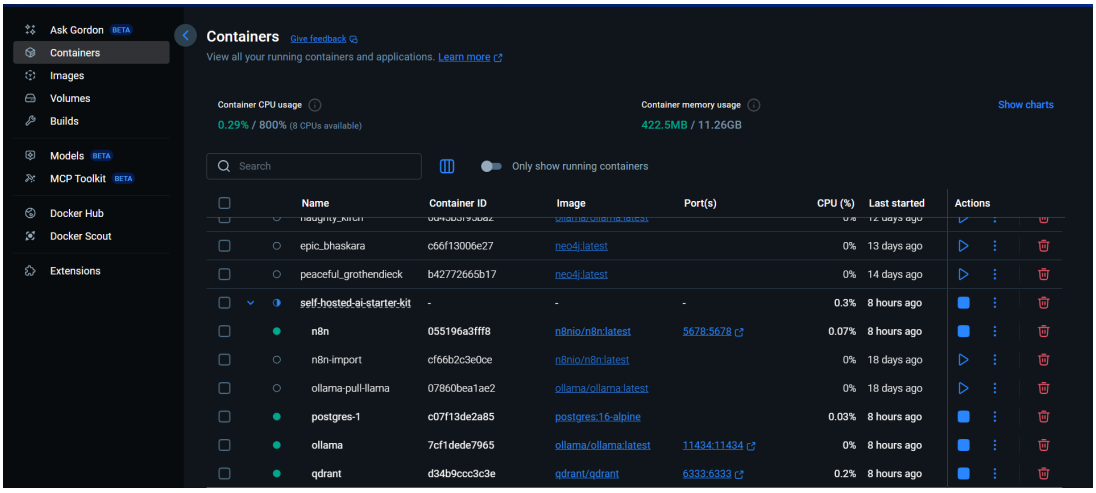

FIGURE 2 – Container self-hosted-ai-starter-kit dans Docker Desktop

3. Compléter le formulaire d'inscription dans n8n (Nom, Prénom, Email, etc.).

Références :

— Guide officiel n8n : `https://github.com/n8n-io/self-hosted-ai-starter-kit`
— Télécharger Docker Desktop : `https://www.docker.com/products/docker-desktop`

6 Configuration de Qdrant en Local

Pour mettre en place Qdrant en mode local, nous utilisons l'instance Qdrant incluse dans le dépôt **self-hosted-ai-starter-kit**, qui est automatiquement déployée avec Docker Compose.

— Qdrant est automatiquement installé et configuré lors du lancement du conteneur `self-hosted-ai-starter-kit`.
— L'instance Qdrant locale est accessible via :
  — **URL** : `http://localhost:6333`
  — **Interface Web** : `http://localhost:6333/dashboard` (pour la gestion visuelle)
  — **Authentification** : Aucune (configuration locale par défaut)
— Vérifiez que Qdrant est opérationnel en consultant Docker Desktop :
  — Le service `qdrant` doit apparaître comme `running` dans le conteneur `self-hosted-ai-starte`
— La collection `Base de connaissance` sera créée automatiquement lors de la première insertion de données par le workflow, avec les paramètres suivants :
  — **Vector Size** : 1536 (taille des embeddings OpenAI)
  — **Distance** : Cosine
  — **Configuration** : Optimisée pour la recherche sémantique
— **Avantages de l'installation locale :**
  — Aucune configuration cloud requise
  — Pas de limites de quota ou de coûts additionnels
  — Données stockées localement pour plus de confidentialité
  — Performance optimale sans latence réseau

7 Vue d'ensemble du Workflow RAG

Ce workflow a pour objectifs principaux :
— Automatiser l'ingestion de documents (PDF, Java, Markdown) depuis des répertoires locaux.
— Extraire et traiter le contenu selon le type de fichier.
— Générer des embeddings vectoriels avec OpenAI.
— Stocker les vecteurs dans une base de données Qdrant pour la recherche sémantique.
— Permettre des requêtes conversationnelles avec récupération contextuelle (RAG).
— Fournir des réponses enrichies avec les métadonnées des documents sources.

7.1 Architecture générale

L'architecture repose sur deux workflows principaux interconnectés autour de la base vectorielle Qdrant :

1. Workflow d'Ingestion Vectorielle Ce premier workflow a pour objectif d'ingérer et vectoriser la connaissance à partir des fichiers sources. Il comprend :
— **Lecture et classification des documents** : détection automatique du type de fichier (PDF, Java, Markdown).
— **Extraction conditionnelle** : traitement spécialisé selon le format du fichier.
— **Découpage et vectorisation** : segmentation du texte et génération d'embeddings OpenAI.
— **Stockage dans Qdrant** : insertion des vecteurs avec métadonnées enrichies.

2. Workflow Agent RAG Ce second workflow permet l'interaction conversationnelle avec la base vectorielle :
— **Réception des requêtes utilisateur** via une interface chat.
— **Recherche sémantique** dans Qdrant pour trouver les documents pertinents.

— **Génération de réponse contextuelle** avec le modèle OpenAI et les documents récupérés.
— **Mémoire conversationnelle** pour assurer la continuité du dialogue.

Base vectorielle (Qdrant) Qdrant constitue le cœur de l'architecture, stockant les embeddings des documents et permettant des recherches sémantiques ultra-rapides par similarité cosinus.

8 Importer le premier Workflow dans n8n

1. Lancer n8n
2. Aller sur `http://localhost:5678`
3. Menu Workflows > `Create Workflow`

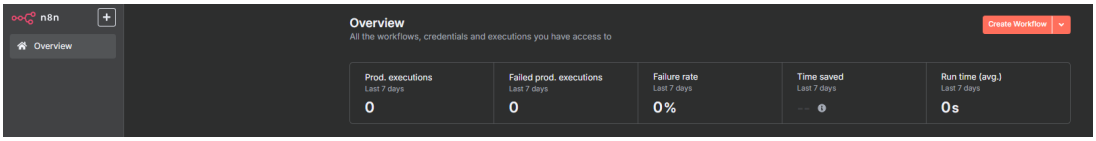

FIGURE 3 – Interface d'accueil N8N

4. Cliquer sur > `Import from File`

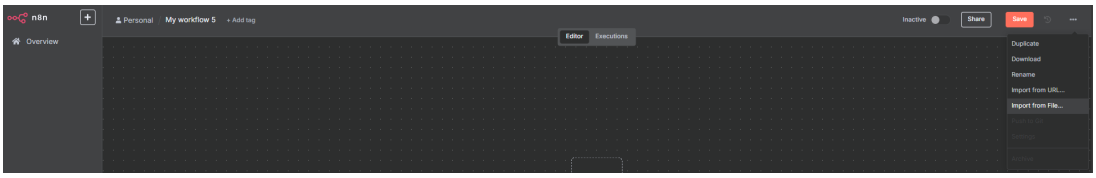

FIGURE 4 – Import depuis un fichier

5. Sélectionner le fichier `Rag Base de connaisance.json`
6. `Save`

9 Workflow n°1 : Ingestion Vectorielle

Architecture des Nœuds

`Manual Trigger → Lecture des Fichiers → Switch (Type de fichier) → Extraction Conditionnelle → Token Splitter → OpenAI Embeddings → Qdrant Insert`

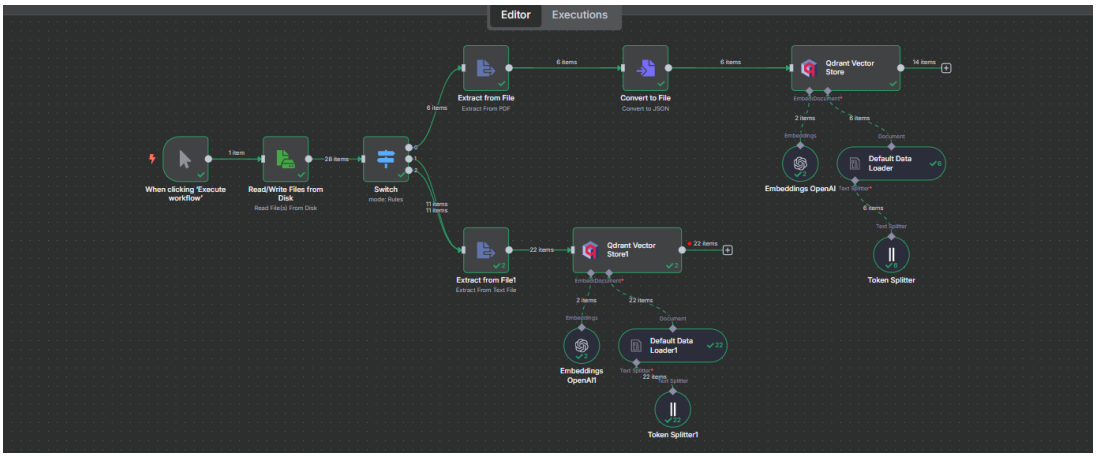

FIGURE 5 – Workflow de création de Base de connaisance

Configuration des Nœuds

9.1 Nœud 1.1 : Manual Trigger

Rôle : Déclenchement manuel du processus d'ingestion des fichiers.
Configuration :
— **Type** : `Manual Trigger`
— **Utilisation** : Cliquer pour lancer le workflow qui lit, traite et vectorise les documents.

9.2 Nœud 1.2 : Read/Write Files from Disk

Rôle : Lecture récursive des fichiers à partir du système local.
Configuration :
— **Operation** : `Read Files`
— **File(s) Selector** : `/data/shared/YourPATH/**/*`
— **Options** : Activer `Include Binary Files` pour les PDF

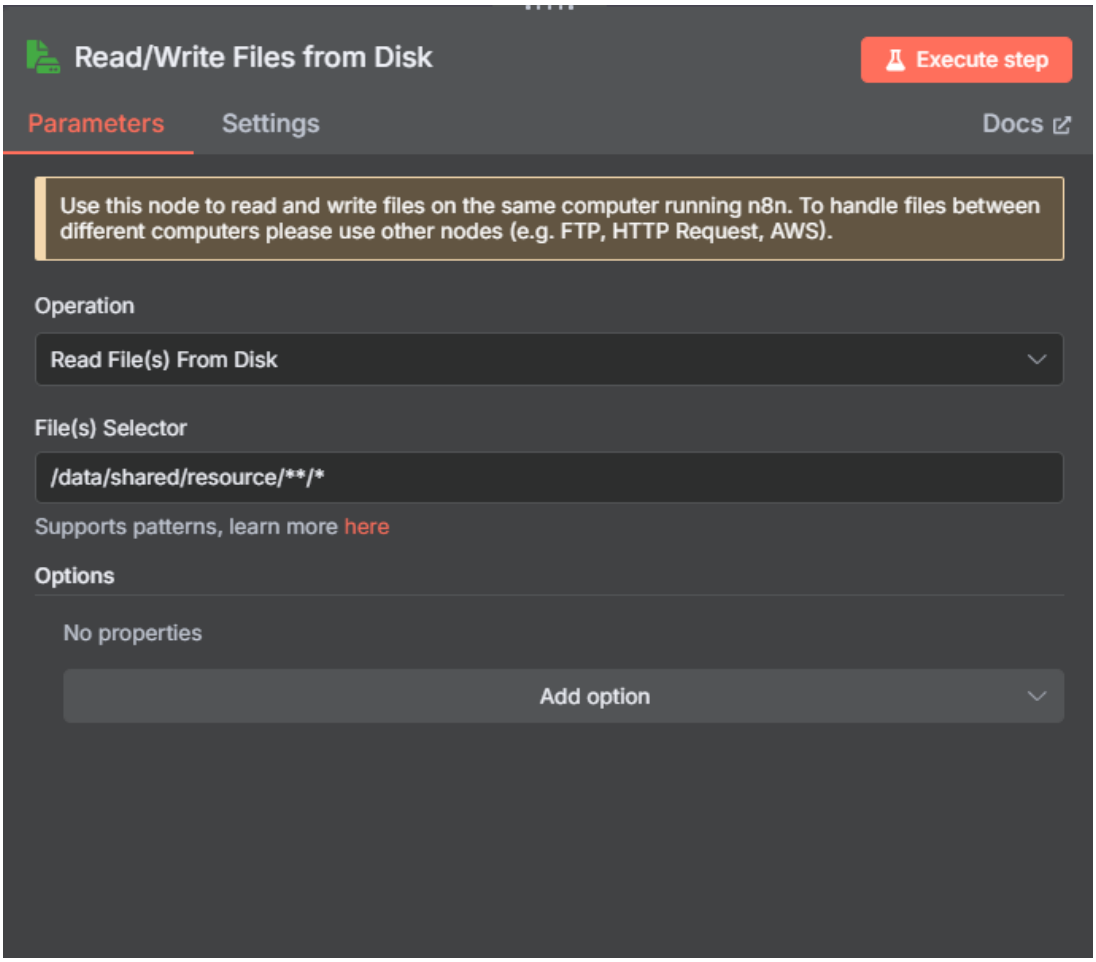

FIGURE 6 – Configuration de la lecture des fichiers

Explication du pattern :
— `**` : Parcours récursif des sous-dossiers
— `*` : Tous les fichiers (PDF, Java, Markdown)

9.3 Nœud 1.3 : Switch (Détection du type de fichier)

Rôle : Routage conditionnel selon l'extension du fichier.
Configuration :
— **Mode** : `Rules`
— **Règle 1** : Si `fileName` se termine par `.pdf`
— **Règle 2** : Si `fileName` se termine par `.java`
— **Règle 3** : Si `fileName` contient `.md` ou `markdown`
— **Fallback** : Route par défaut pour les autres types

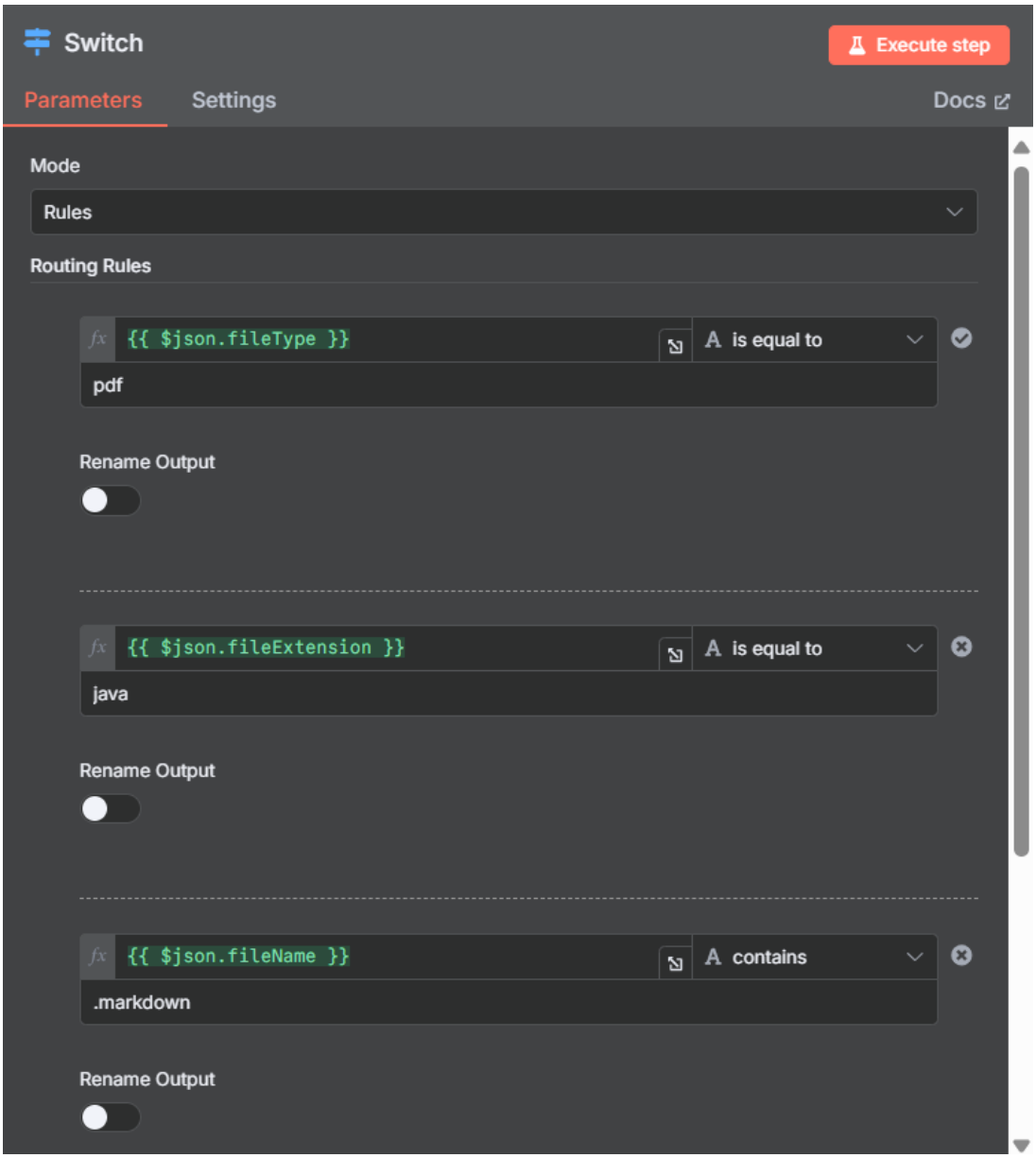

FIGURE 7 – Configuration de sépération fichier selon extension

9.4 Nœud 1.4a : Extract from File (PDF)

Rôle : Extraction structurée du contenu des fichiers PDF.
Configuration :
— **Operation** : `Extract from File`
— **Extract To** : `JSON`
— **Options** : Activer la préservation du formatage

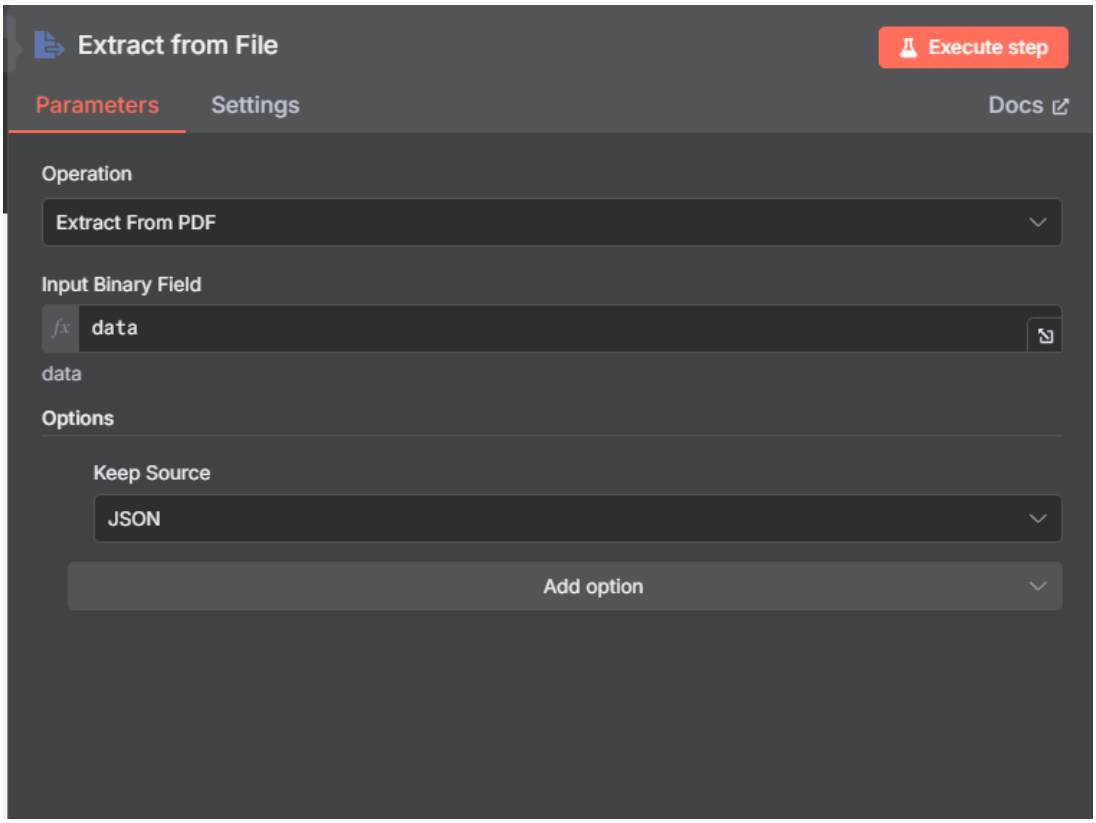

FIGURE 8 – Enter Caption

9.5 Nœud 1.4b : Extract from File 1 (Java/Markdown)

Rôle : Extraction du contenu textuel brut des fichiers code et documentation.
Configuration :
— **Operation** : `Extract from Text File`

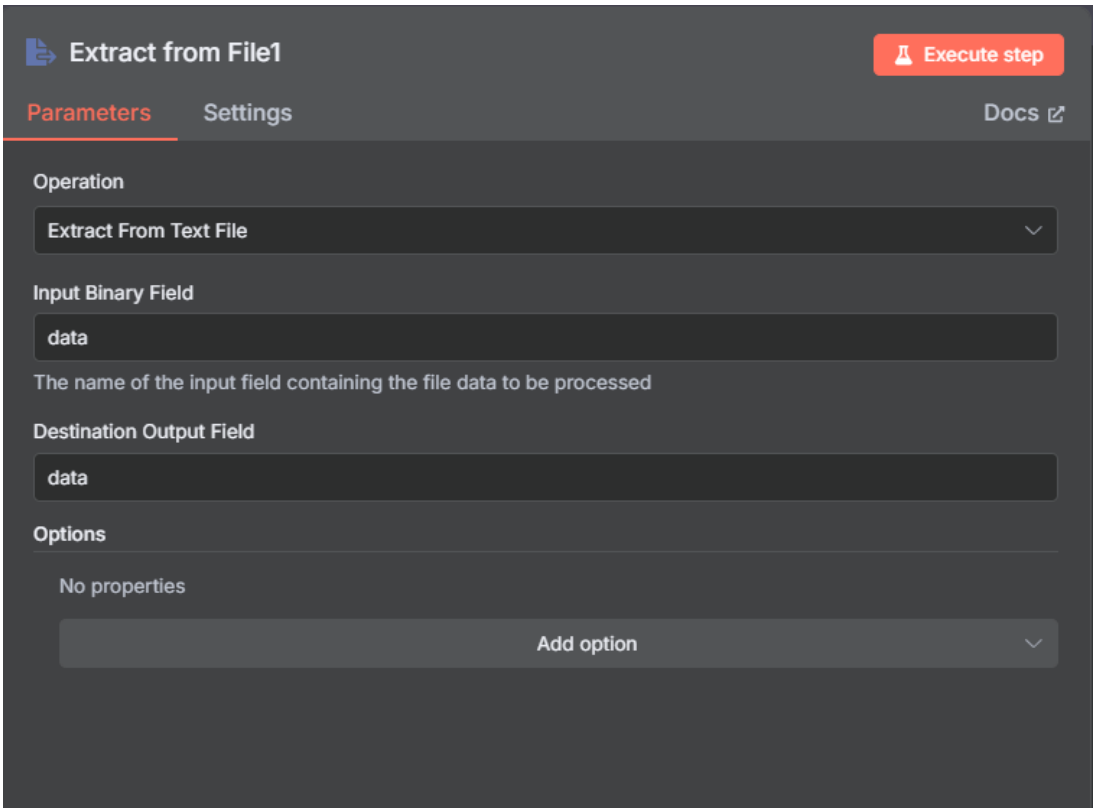

FIGURE 9 – Enter Caption

9.6 Nœud 1.5 : Convert to JSON (Route PDF)

Rôle : Conversion du contenu PDF extrait en format JSON structuré.
Configuration :
— **Mode** : `Each Item to Separate File`
— **Options** :
  — **source** : `{{ $json.fileName }}` (conservation du nom du fichier PDF source)

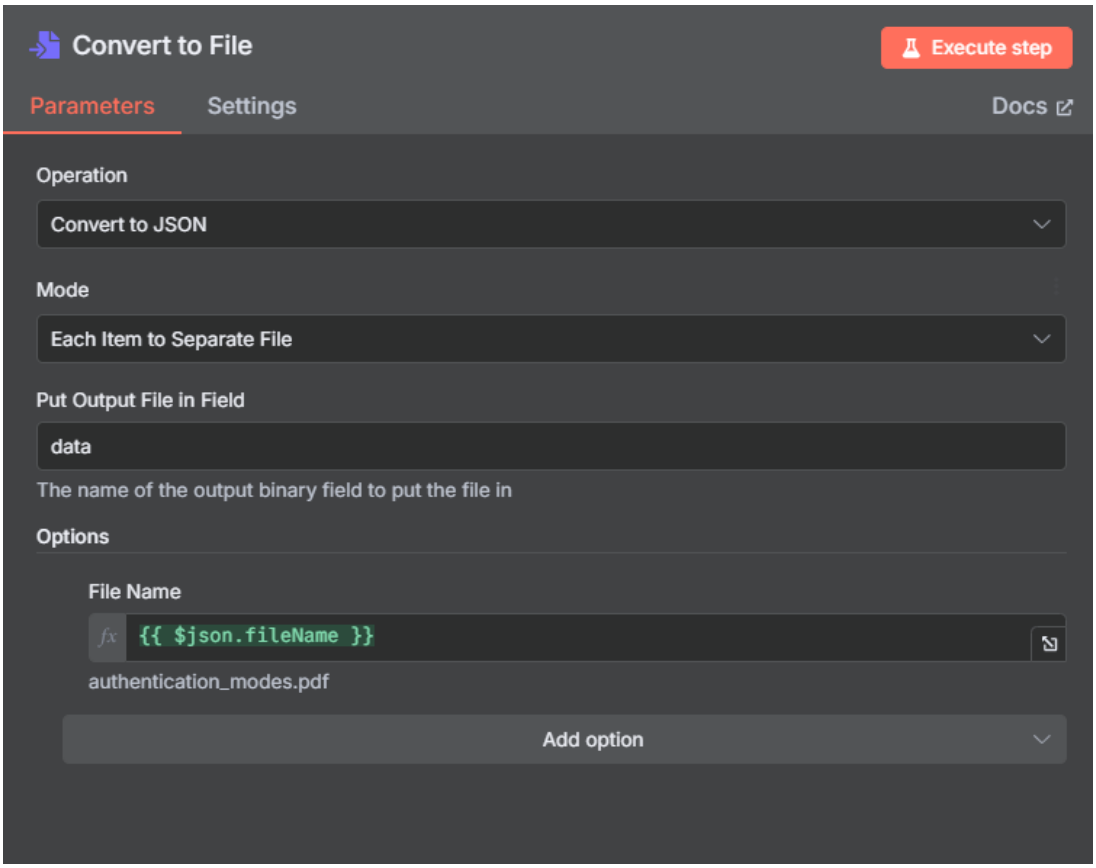

FIGURE 10 – Configuration Noeud Convert to JSON

9.7 Nœud 1.6 : Qdrant Insert(Route PDF ,Route JAVA/MD)

Rôle : Insertion des vecteurs et métadonnées dans la collection Qdrant.
Configuration :
— **Operation** : `Insert Documents`
— **Collection Name By ID** : `Base de connaissance`
— **Embedding Batch Size** : `200`

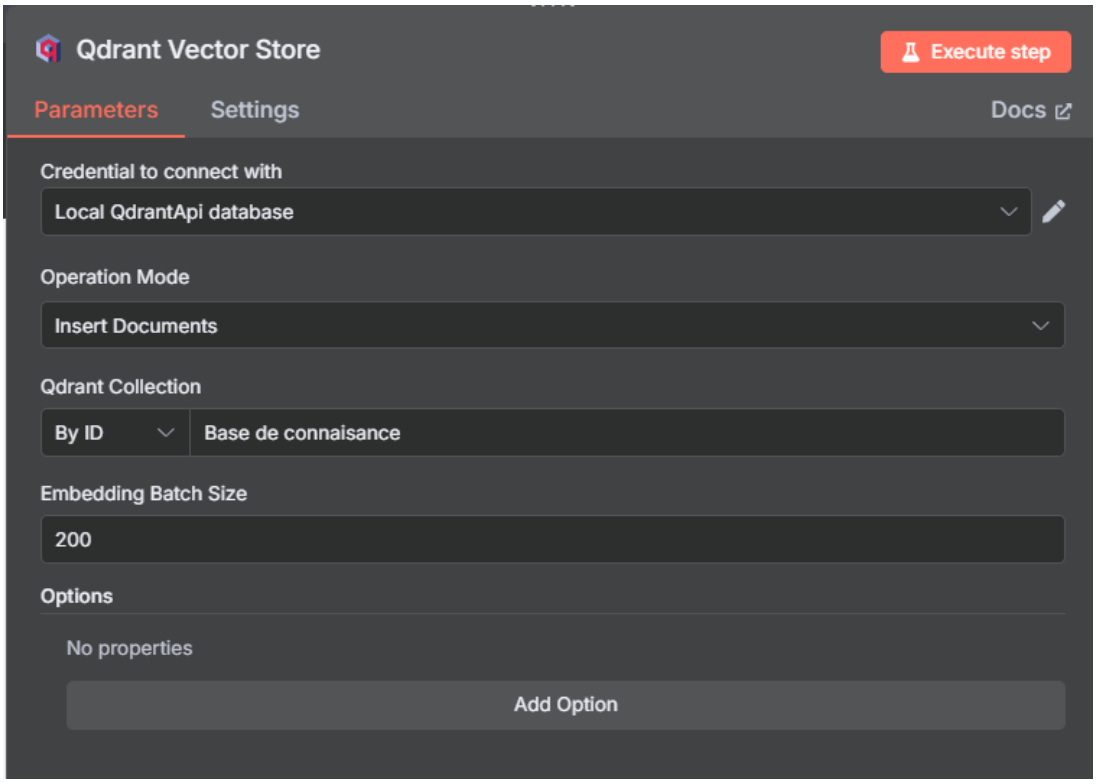

FIGURE 11 – Configuration Noeud Qdrant Vector Store

9.8 Nœud 1.7a : Data Loader (Route PDF)

Rôle : Insertion des chunks PDF vectorisés dans Qdrant.
Configuration :
— **Type of Data** : `JSON`
— **Operation** : `Load Specific Data`
— **Data Loader** : Mode `Load Specific Data`
— **Collection Name** : `Base de connaissance`
— **Text Splitting** : `Custom`
— **Metadata** :
  — **source** : Nom du fichier PDF original

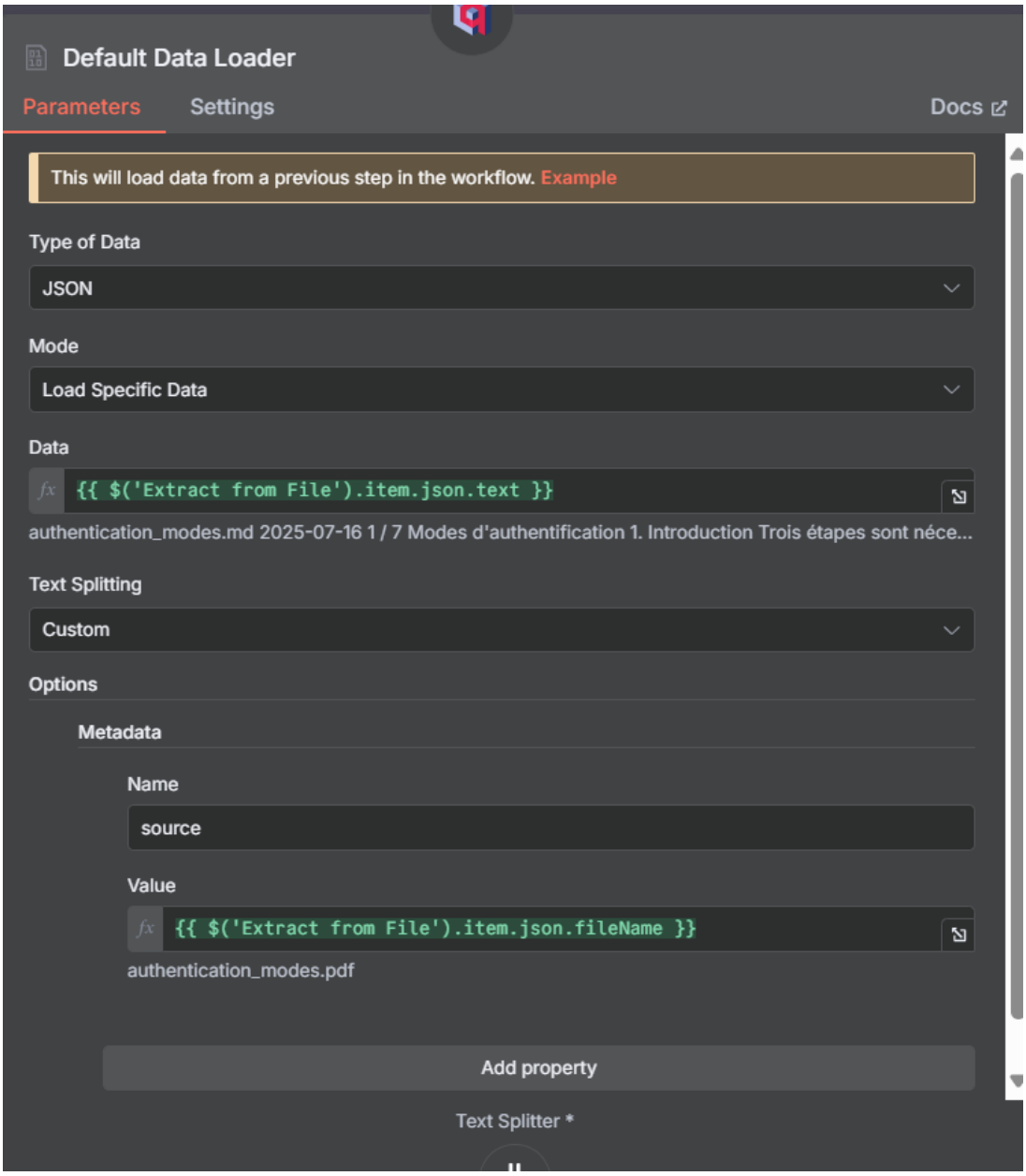

FIGURE 12 – Configuration Data Loader PDF Route

9.9 Nœud 1.7b : Data Loader 1 (Route JAVA/MD)

Rôle : Insertion des chunks de code/documentation vectorisés dans Qdrant.
Configuration :
— **Type of Data** : `JSON`
— **Operation** : `Load Specific Data`
— **Data Loader** : `{{ $json.data }}` (contenu du Extract from File)
— **Collection Name** : `Base de connaissance`
— **Text Splitting** : `Custom`
— **Metadata** :
  — **source** : `file_name` (depuis Read/Write Files from Disk)

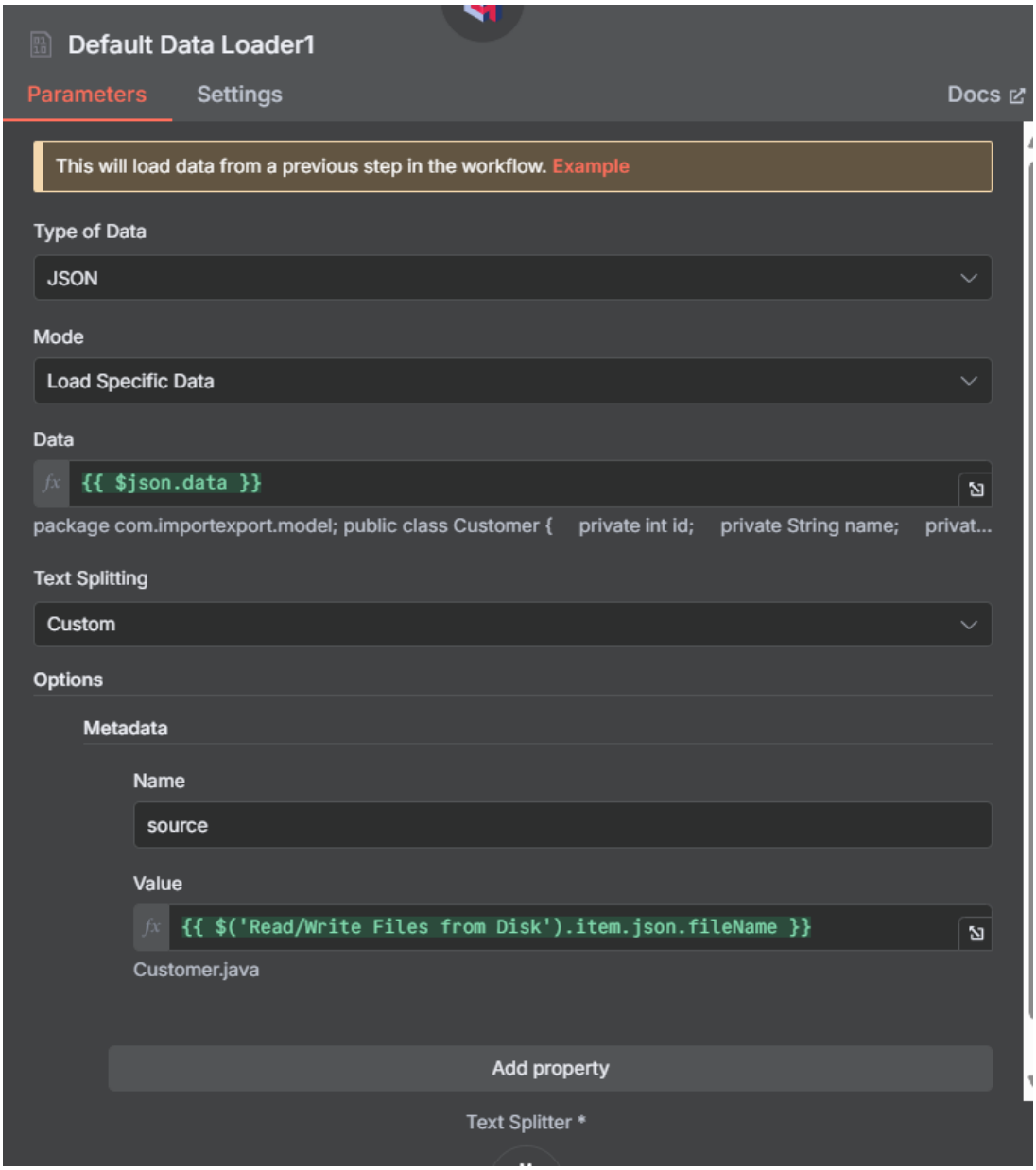

FIGURE 13 – Configuration Data Loader PDF Route

9.10 Nœud 1.8 : Token Splitter (Route PDF et Route JAVA/MD)

Rôle : Découpage du texte PDF en chunks optimaux pour les embeddings.
Configuration :
— **Chunk Size** : `1000` tokens
— **Chunk Overlap** : `0` tokens
— **Text Splitte** : `Token Splitter`

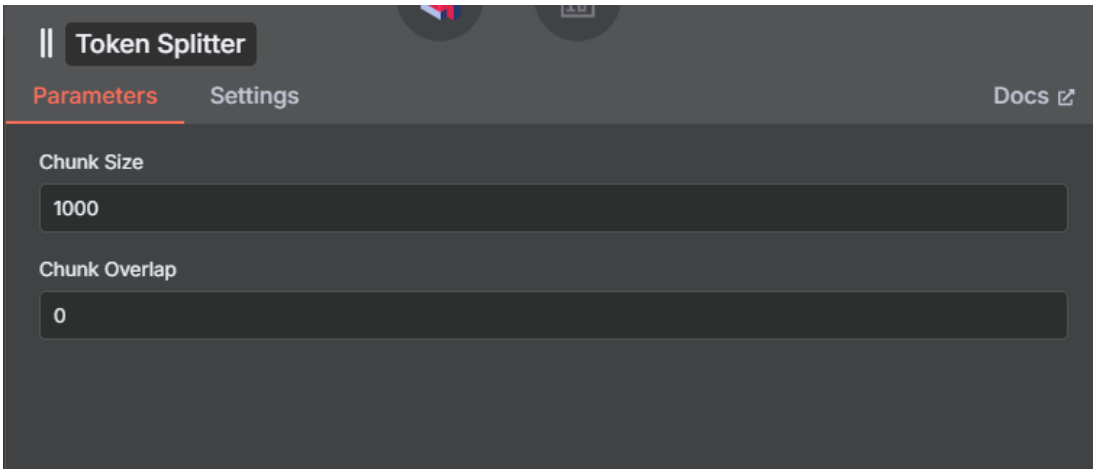

FIGURE 14 – Enter Caption

9.11 Nœud 1.9 : OpenAI Embeddings

Rôle : Génération des vecteurs d'embeddings pour chaque chunk de texte.
Configuration :
— **Model** : `text-embedding-3-small` ou `text-embedding-ada-002`
— **API Key** : Référence vers les credentials OpenAI
Création des credentials OpenAI :
— Aller dans Noeud OpenAi Embeddings + **Create new Credentials** dans premier champ Credential to connect with

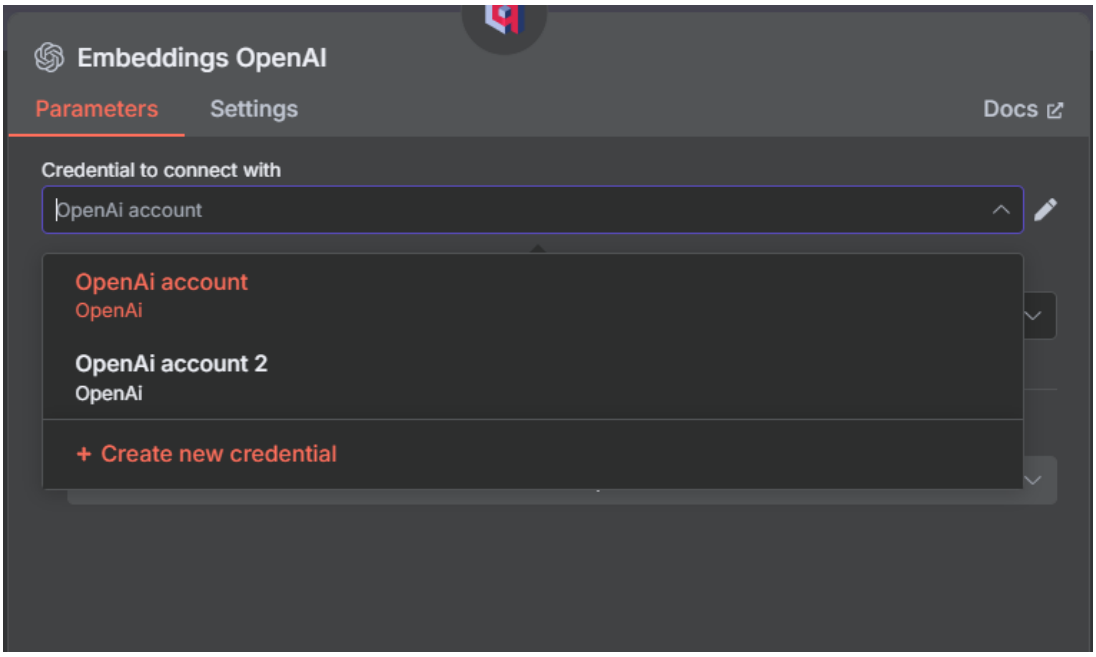

FIGURE 15 – Création Connexion OpenAi Embeddings

— Créer **OpenAI API**
— Renseigner votre clé API obtenue sur `https://platform.openai.com/api-keys`

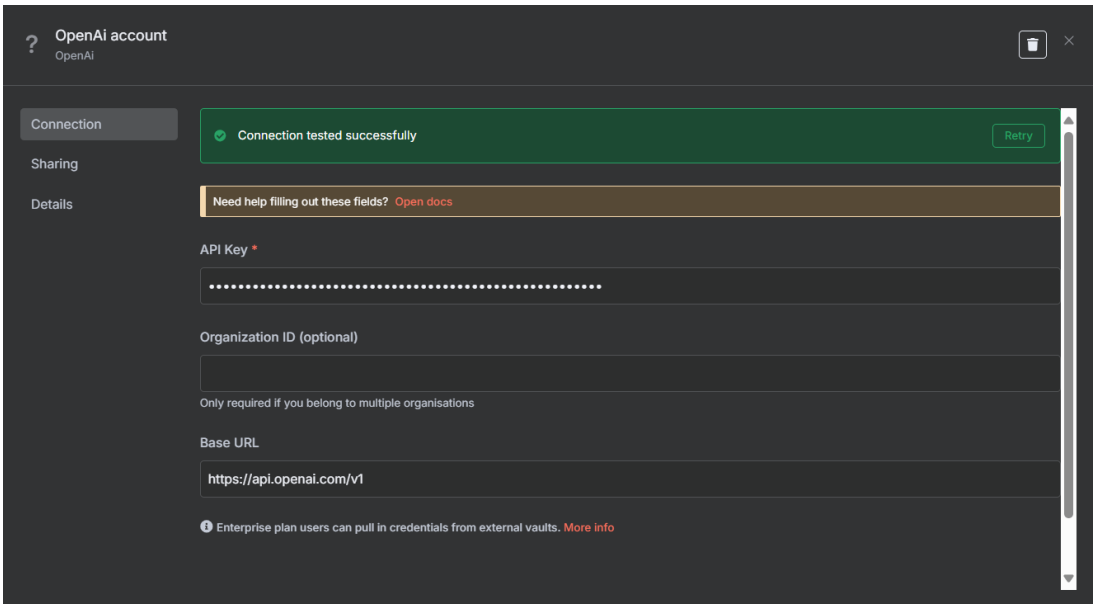

FIGURE 16 – Configuration clé dans connexion OpenAI

10 Workflow n°2 : Agent RAG Conversationnel

10.1 Architecture

`Chat Message → AI Agent → OpenAI Chat Model → Postgres Memory ← Qdrant Retriever (Tool)`

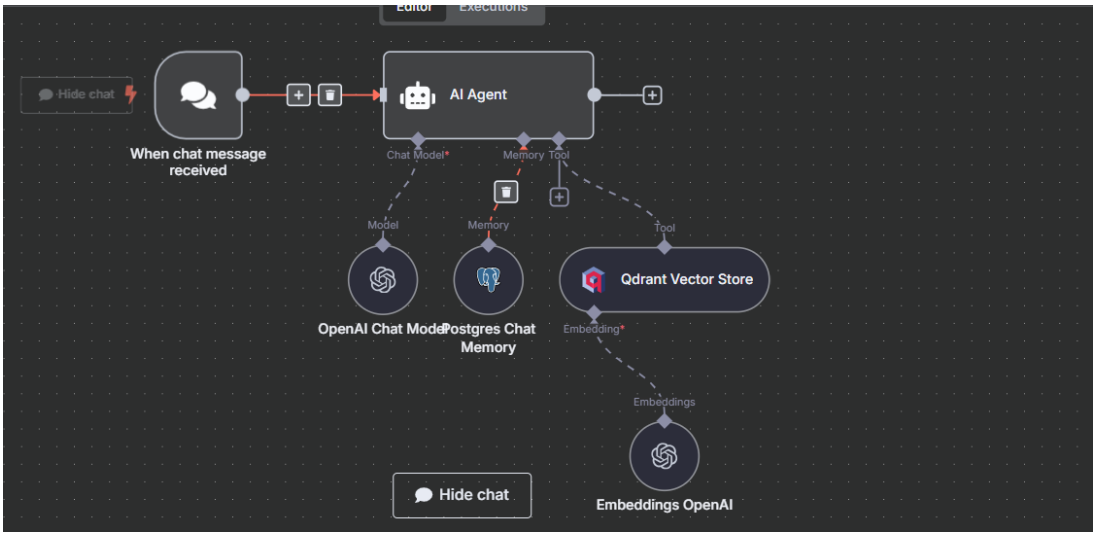

FIGURE 17 – Workflow Rag Agent

Description générale : L'agent intelligent reçoit une question, effectue une recherche sémantique dans Qdrant pour récupérer les documents pertinents, et génère une réponse contextuelle enrichie.

10.2 Nœud 2.1 : When Chat Message Received

Rôle : Point de départ du workflow conversationnel. **Fonction** : Déclenche le flux chaque fois qu'un utilisateur envoie un message dans le chat.

10.3 Nœud 2.2 : AI Agent

Rôle : Agent RAG principal pour les interactions conversationnelles. **Fonction** : Analyse la requête, recherche dans Qdrant et compose une réponse.

Utilise :

— Un modèle de langage OpenAI pour comprendre et générer les réponses.
— Une mémoire Postgres pour conserver le contexte conversationnel.
— Un outil Qdrant Retriever pour récupérer les documents pertinents.

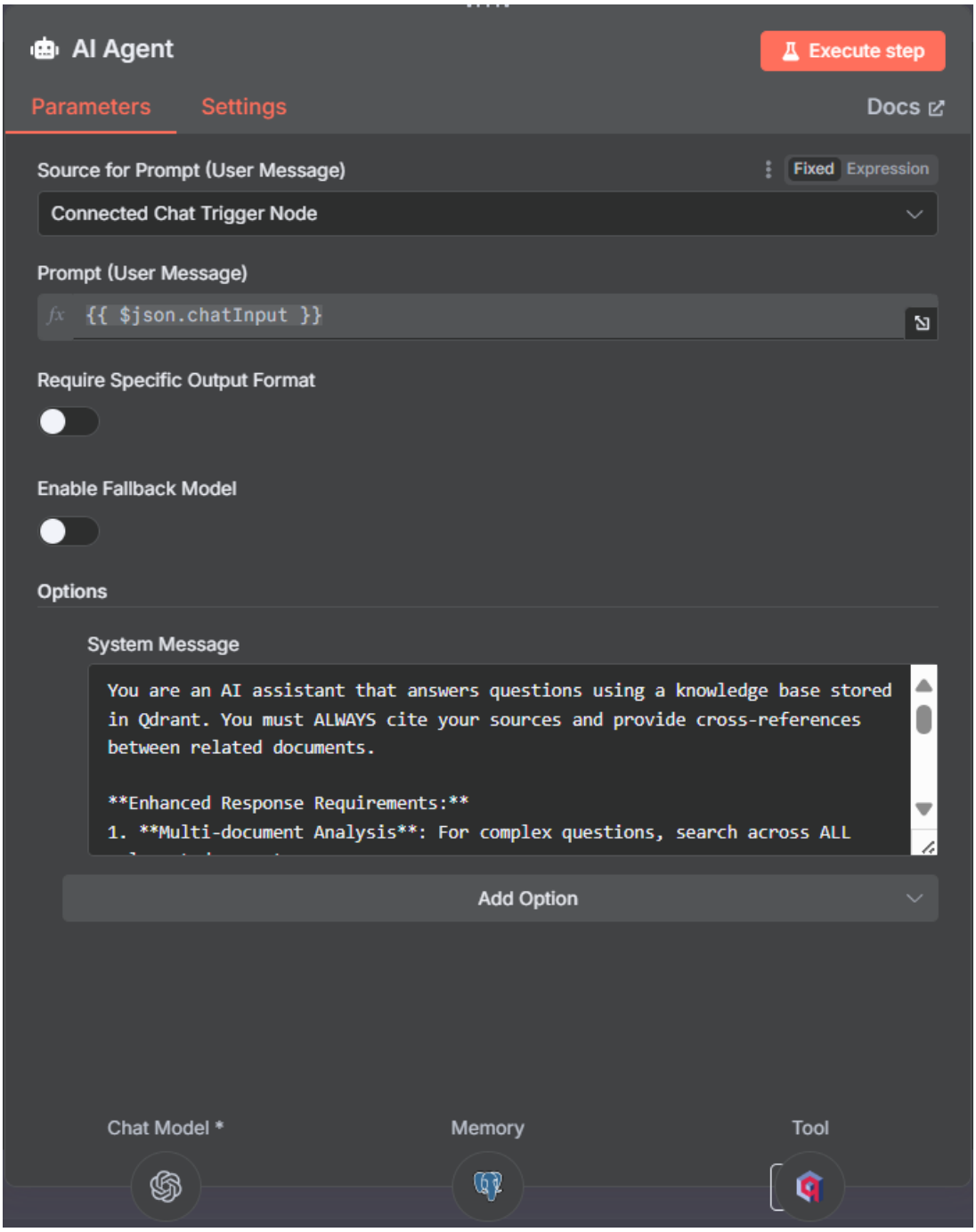

FIGURE 18 – Configuration Ai agent

System Message :

You are an AI assistant that answers questions using a knowledge base stored in Qdrant. You must ALWAYS cite your sources and provide cross-references between related documents.

Enhanced Response Requirements : 1. **Multi-document Analysis** : For complex questions, search across ALL relevant documents 2. **Cross-referencing** : When citing information, link related concepts from different documents 3. **Process Flow** : For process questions, show the complete flow across documents 4. **Code Quality Integration** : Connect development processes with quality standards

Rules :
— Only use information returned from Qdrant retrieval (no hallucinations).
— Always cite your sources with file names and relevant excerpts.
— Provide comprehensive answers by combining multiple retrieved documents when relevant.
— Include metadata when available : file type, source path, extraction date.
— Respond in the language of the query.
Response format :
— Main answer based on retrieved content
— Sources used with their file names and relevant passages
— Cross-references between documents when applicable

10.4 Nœud 2.3 : OpenAI Chat Model

Rôle : Modèle de génération de réponses (GPT-4 ou GPT-3.5-turbo). **Fonction** : Génère des réponses contextuelles basées sur les documents récupérés.

Configuration :
— **Model** : `gpt-4` ou `gpt-3.5-turbo`
— **Temperature** : `0.0` (pour réduire les hallucinations)
— **Credentials** : Référence vers les credentials OpenAI

10.5 Nœud 2.4 : Postgres Chat Memory

Rôle : Mémoire conversationnelle persistante. **Fonction** : Stocke l'historique des conversations pour maintenir le contexte.

Configuration :
— **Type de mémoire** : `Postgres Chat Memory`
— **Connexion** : Utilise les paramètres du fichier `.env`
  — **Host** : `postgres`
  — **Port** : `5432`
  — **Database** : `$POSTGRES_DB`
  — **User** : `$POSTGRES_USER`
  — **Password** : `$POSTGRES_PASSWORD`
— **Session ID** : `{{ $json.sessionId }}` (pour sessions multiples)

10.6 Nœud 2.5 : Qdrant Retriever (Tool)

— **Operation Mode** : Retrive Documents
— **Description** : Récupère les documents les plus pertinents par similarité vectorielle.

Configuration :
— **Collection Name from List or By ID** : `Base de connaissance`
— **Top K** : `15` (nombre de documents à récupérer)
— **Embedding Model** : Même modèle que pour l'ingestion (text-embedding-3-small)

11 Configuration des Credentials

11.1 PostgreSQL (inclus avec n8n)

1. Consultez le fichier `.env` dans le dossier `self-hosted-ai-starter-kit`

2. Utilisez les valeurs par défaut :
   — Host : postgres
   — Port : 5432
   — Database : n8n
   — User : root
   — Password : password

12 Test et Validation

12.1 Test du Workflow d'Ingestion

1. Placez vos fichiers PDF, Java et Markdown dans `/data/shared/YourPATH/`
2. Lancez le workflow d'ingestion via le Manual Trigger
3. Vérifiez l'insertion dans Qdrant via l'interface Cloud
4. Contrôlez les logs n8n pour détecter d'éventuelles erreurs

12.2 Test du Chat RAG

1. Lancez le workflow agent RAG
2. Envoyez une question test via l'interface chat
3. Vérifiez que la réponse cite les sources appropriées
4. Testez la continuité conversationnelle avec plusieurs messages

13 Conclusion

Ce guide vous a permis de mettre en place un système RAG (Retrieval-Augmented Generation) moderne et efficace en combinant n8n pour l'orchestration, Qdrant pour le stockage vectoriel et OpenAI pour les embeddings et la génération.

Avantages de cette architecture :

— **Recherche sémantique avancée** : Qdrant offre des performances exceptionnelles pour la similarité vectorielle.
— **Traitement multi-format** : Support natif des PDF, code source et documentation.
— **Scalabilité cloud** : Architecture prête pour la production avec Qdrant Cloud.
— **Réponses contextuelles** : Intégration transparente entre recherche et génération.

Extensions possibles :

— **Filtrage avancé** : Ajout de métadonnées pour filtrer par projet, date, ou auteur.
— **Réindexation automatique** : Mise à jour des embeddings lors de modifications de fichiers.
— **Interface utilisateur** : Construction d'une interface web dédiée pour les requêtes.
— **Analyse multimodale** : Extension aux images et diagrammes contenus dans les PDF.

Cette solution constitue une base robuste pour développer des applications RAG avancées tout en bénéficiant de la puissance de la recherche vectorielle et de la flexibilité de l'orchestration n8n.