

Guide Complet - Workflow RAG Agent avec n8n

Aziza Neffati

Table des matières

1	Introduction	2
2	Pré-requis	3
3	Installation de Node.js	3
4	Installation de n8n	4
5	Démarrer n8n	4
5.1	Architecture générale	4
6	Importer un Workflow n8n	5
7	Partie 1 : Ingestion des Documents	5
7.1	Nœud 1.1 : Execute Workflow	5
7.2	Nœud 1.3 : Switch	7
7.3	Nœud 1.4 : Extract from File1	7
7.4	Nœud 1.5 : Convert to JSON	8
7.5	Nœud 1.6 : Qdrant Vector Store	9
7.5.1	Méthode1 : Création manuelle via la console Qdrant	9
7.5.2	Méthode 2 : Création directe du Vector Store via n8n	11
7.6	Nœud 1.7 : Default Data Loader	12
7.6.1	Configuration	12
7.6.2	Métadonnées	12
7.7	Token Splitter	13
7.7.1	Paramètres de configuration	13
7.7.2	Fonctionnement	13
8	Partie 2 : Chat et Réponses	15
8.1	Architecture	15
8.2	Nœud 2.1 : When chat message received	15
8.3	Nœud 2.2 : AI Agent	15
8.4	2.3. OpenAI Chat Model1	17
8.5	2.5. Qdrant Vector Store1	17
9	Conclusion	19

1 Introduction

Ce guide détaille les étapes pour installer et exécuter un workflow RAG (Retrieval-Augmented Generation) en local, en utilisant **n8n**. n8n est un outil d'automatisation sans code/low-code pour orchestrer des tâches telles que l'analyse de documents, les appels API, la gestion de fichiers, etc.

À propos du Retrieval-Augmented Generation (RAG)

Le **Retrieval-Augmented Generation (RAG)** est une architecture qui combine les avantages de la recherche d'information avec ceux de la génération de texte. Au lieu de s'appuyer uniquement sur les connaissances internes d'un modèle de langage, RAG les enrichit en récupérant des documents pertinents depuis une base externe (comme un magasin vectoriel), afin de générer des réponses plus précises et contextualisées.

Pourquoi utiliser le RAG ?

Les modèles de langage classiques peuvent parfois inventer des informations (halluciner) ou fournir des réponses obsolètes. RAG permet de surmonter ces limites en :

- **Ancrant les réponses dans des connaissances externes** : les documents récupérés apportent un soutien factuel
- **Améliorant la fiabilité et la traçabilité** : chaque réponse peut citer ses sources
- **Gérant des données spécifiques à un domaine** : RAG peut répondre à des questions basées sur votre propre base documentaire privée

Architecture générale d'un système RAG

Un système RAG typique comprend trois composants principaux :

1. **Retriever (extracteur)** : interroge une base vectorielle pour trouver les documents pertinents
2. **Reader/Generator (lecteur/générateur)** : utilise les documents récupérés pour générer une réponse
3. **Magasin vectoriel** : stocke les embeddings des documents pour une récupération rapide et précise (par ex. Qdrant, FAISS)

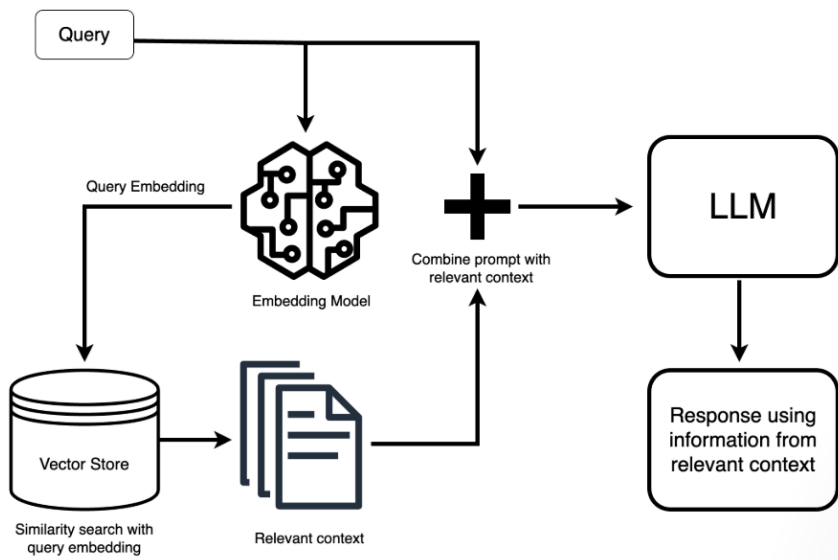


FIGURE 1 – Architecture typique d'un système RAG

Notre workflow RAG avec n8n

Dans ce guide, nous mettons en œuvre un système RAG personnalisé en utilisant :

- **n8n** : une plateforme d'automatisation no-code/low-code pour orchestrer le workflow
- **Qdrant** : un magasin vectoriel pour stocker et interroger les embeddings des documents
- **OpenAI** : pour extraire les embeddings et générer les réponses

Ce que fait notre RAG :

- Scanne et indexe automatiquement les documents depuis un dossier local
- Les divise en segments sémantiques et les stocke dans Qdrant
- Répond aux questions de l'utilisateur en temps réel grâce à OpenAI, en s'appuyant sur les documents indexés
- Fournit des réponses précises, contextualisées et accompagnées de sources citées

2 Pré-requis

Avant de commencer, assurez-vous d'avoir les éléments suivants installés sur votre machine :

- Node.js (version 18 ou supérieure)
- npm (inclus avec Node.js)
- Terminal ou Invite de commande

3 Installation de Node.js

1. Télécharger la version LTS sur <https://nodejs.org/>
2. Suivre les instructions d'installation

3. Vérifier l'installation avec les commandes :

```
node -v
npm -v
```

4 Installation de n8n

Dans le terminal :

```
npm install -g n8n
npx n8n
```

5 Démarrer n8n

Exécuter la commande :

```
n8n
```

Puis taper o pour l'ouvrir dans le navigateur. Par défaut, l'interface est disponible sur : <http://localhost:5678>

5.1 Architecture générale

- Partie Ingestion : Lire et indexer les documents
- Partie Chat : Interface conversationnelle avec recherche sémantique
- Base vectorielle : Qdrant

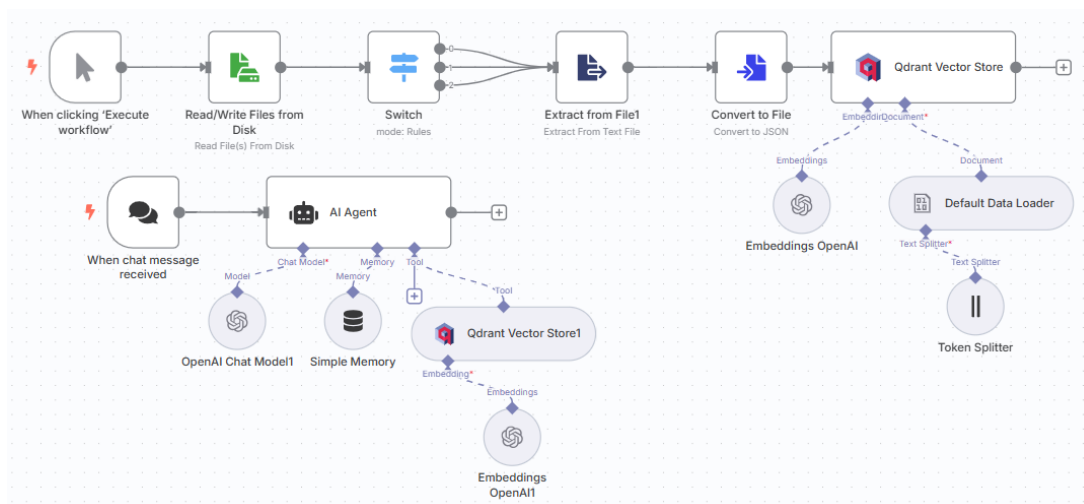


FIGURE 2 – Architecture générale du workflow RAG

6 Importer un Workflow n8n

1. Lancer n8n
2. Aller sur <http://localhost:5678>
3. Menu Workflows > + New Workflow
4. Cliquer sur > Import from File
5. Sélectionner le fichier .json
6. Save puis Execute Workflow

7 Partie 1 : Ingestion des Documents

Architecture des Nœuds

Execute Workflow → Read Files → Extract from file → Convert to JSON → Qdrant Vector Store → Default Data Loader → Token Splitter / Embeddings OpenAI

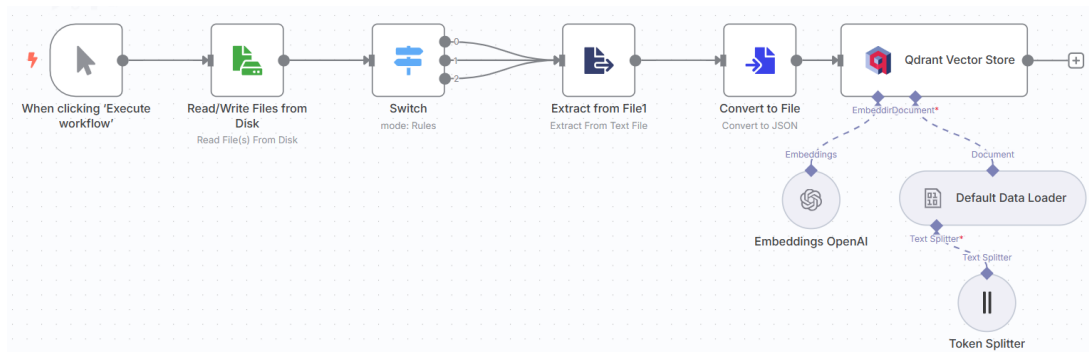


FIGURE 3 – Ingestion des Documents

Nœuds et Fonctions

7.1 Nœud 1.1 : Execute Workflow

Rôle : Déclenchement manuel du processus

Configuration :

- Type : Manual trigger
- Utilisation : Cliquer pour lancer l'indexation des documents

1.2 Node "Read/Write Files from Disk"

Rôle : Lecture récursive des fichiers depuis le système local

Configuration :

- **Operation** : Read Files
- **File(s) Selector** : D:/votre/dossier/**/*.{pdf,txt,docx,md}
- **Options** :
 - Include file metadata
 - Read file content as binary

Pattern glob expliqué :

- ****** : Parcours récursif des sous-dossiers
- ***** : Tous les fichiers
- **{pdf,txt,docx,md}** : Extensions spécifiques

FIGURE 4 – Noeud-Read files

Astuce – Accès aux sous-dossiers et fichiers dans une arborescence :

Pour accéder à l'ensemble des fichiers dans les sous-dossiers, ajoutez `/**/*.*` à la fin du chemin principal.

Exemple :

Pour lire tous les fichiers dans `C:/Users/acer laptop/Downloads/`, y compris ceux des sous-dossiers :

`C:/Users/acer laptop/Downloads/**/*.*`

Bonnes pratiques à suivre :

- **Chemins de fichiers valides** : Vérifiez que le fichier existe réellement à l'emplacement indiqué. Sinon, le nœud générera une erreur de lecture/écriture.
- **Utilisez le format absolu** :
 - `C:/Users/Downloads/Documents/data.csv`
 - `../data.csv`
- **Respect du format du chemin** : Utilisez toujours le slash `/` dans le chemin.
 - Mauvais : `C:\Users\acer laptop\Downloads\Commande.pdf`
 - Correct : `C:/Users/acer laptop/Downloads/Commande.pdf`
- **Nom du fichier** : Évitez les accents et caractères spéciaux dans les noms de fichiers/dossiers.

- **Droits d'accès** : Assurez-vous d'avoir les droits en lecture/écriture sur le fichier.
- **Fichiers multiples** : Si vous traitez plusieurs fichiers, ajoutez-les tous dans le champ File(s) selector (la plupart des outils acceptent la sélection multiple).
- **Type de fichier** : Vérifiez que le nœud prend en charge le format du fichier (ex. : PDF, CSV, JSON...).

7.2 Nœud 1.3 : Switch

Rôle : Redirige selon le type de fichier

Switch Execute step

Parameters Settings Docs

Mode
Rules

Routing Rules

fx `{{ $json.fileName.split('.').pop().toLowerCase() }}` A is equal to ✕
md

Rename Output ☐

fx `{{ $json.fileName.split('.').pop().toLowerCase() }}` A is equal to ✓
java

Rename Output ☐

fx `{{ $json.fileName.split('.').pop().toLowerCase() }}` A is equal to ✕
xml

FIGURE 5 – Nœud-Switch

7.3 Nœud 1.4 : Extract from File1

Rôle : Extraction du contenu textuel selon le type de fichier

Convert to File Execute step

Parameters **Settings** Docs

Operation
Convert to JSON

Mode
Each Item to Separate File

Put Output File in Field
data
The name of the output binary field to put the file in

Options

File Name
fx {{ \$json.fileName }}

Add option

FIGURE 7 – Noeud-Convert to Json

7.5 Nœud 1.6 : Qdrant Vector Store

Rôle : Stockage des embeddings dans la base vectorielle

7.5.1 Méthode1 : Création manuelle via la console Qdrant

Configuration Technique

- Étapes de configuration dans Qdrant :
 - Créer une clé API
 - Connectez-vous à votre compte Qdrant.
 - Accédez à la section API Keys et créez une clé API.

Aziza Neffati - Base Account Aziza Neffati

Clusters Create +

CLUSTER	CONFIGURATION	PROVIDER	STATUS	ACTIONS
FREE TIER test 7 Upgrade to a Paid Cluster	1 NODE Disk: 4GiB RAM: 1GiB vCPUs: 0.5	aws eu-central-1	HEALTHY	...

FIGURE 8 – API Key

- Objet groupé
 - Créer une collection en accédant à console et tapant le code
 - Nom de la collection : [à définir selon votre projet]

- Dimensions : 1536 (Compatible au modèle d'Open AI)
- Metric : cosine

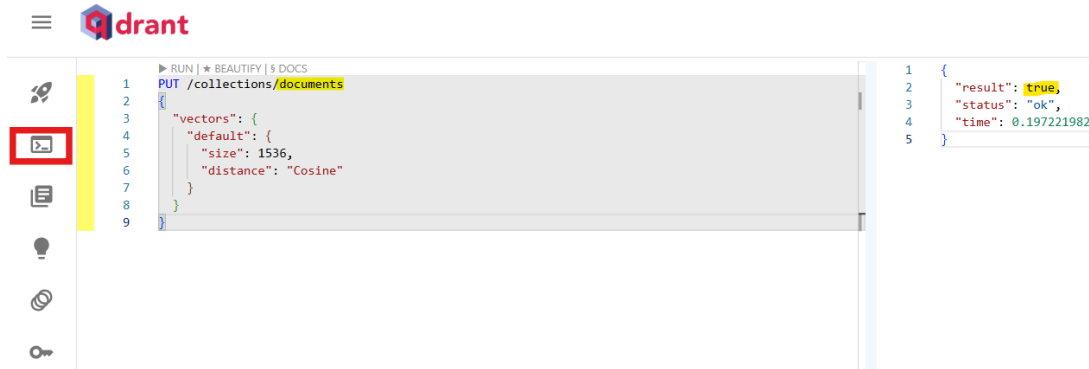


FIGURE 9 – Create Collection

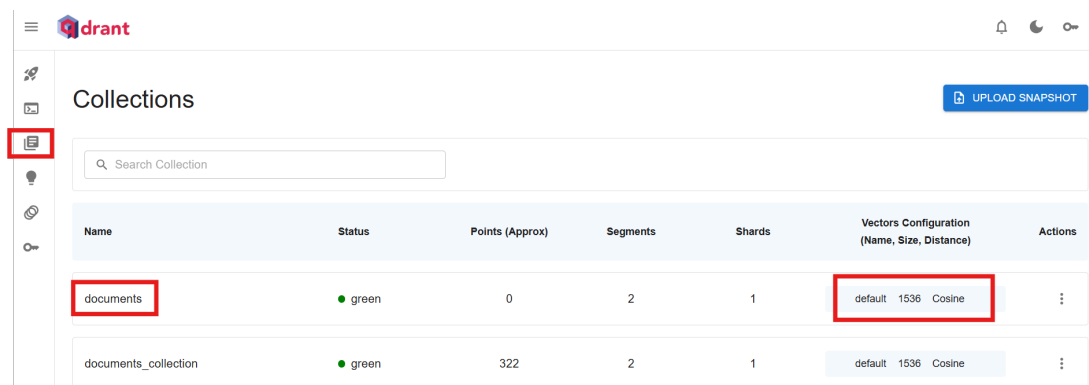


FIGURE 10 – List of collections

3. Configuration dans n8n

- Dans votre workflow n8n :
- Ajoutez un nœud Qdrant Vector Store.
- Dans les paramètres du nœud :
 - API Key : copiez-collez la clé API générée sur Qdrant.
 - Operation : Insert Document
 - Index : From List → sélectionnez le nom de votre Collection créé.
 - Batch Size : 200 (ou une autre valeur adaptée à la taille de vos données).
 - Metadata Fields :
 - file_name : {{\$json.fileName}}
 - file_path : {{\$json.filePath}}
 - chunk_id : {{\$json.chunkId}}
 - created_at : {{\$now}}

Une fois le workflow exécuté, vous verrez les chunks (morceaux) extraits de vos documents dans la collection sur Qdrant.

FIGURE 11 – Noeud-Qdrant Vector Store

7.5.2 Méthode 2 : Création directe du Vector Store via n8n

Objectif : Créer une nouvelle collection Qdrant (vector store) directement depuis n8n, sans passer par l’interface web de Qdrant.

Procédure Simplifiée

1. Pré-requis :
 - Disposer de votre clé API Qdrant (voir Méthode 1 pour la génération).
2. Dans le **premier noeud Qdrant Vector Store** de votre workflow :
 - Dans le champ **Qdrant Collection**, sélectionnez “**By ID**”.
 - Saisissez un nom pour votre nouvelle collection (vector store).
 - Assurez-vous que les paramètres suivants sont correctement définis :
 - **Operation** : Insert Document
 - **Batch Size** : selon vos données (par exemple, 200)
 - **Metadata Fields** :
 - `file_name` : `{{ $json.fileName }}`
 - `file_path` : `{{ $json.filePath }}`
 - `chunk_id` : `{{ $json.chunkId }}`
 - `created_at` : `{{ $now }}`
 - Cliquez sur **Exécuter le noeud**.
3. Résultat :
 - Le vector store sera automatiquement créé dans Qdrant à partir de ce premier envoi de données.
 - Dans les noeuds suivants (par exemple un noeud d’agent IA), vous pourrez simplement sélectionner ce vector store dans la liste déroulante.

Qdrant Vector Store Execute step

Parameters Settings Docs

Credential to connect with
QdrantApi account

Tip: Get a feel for vector stores in n8n with our [RAG starter template](#)

Operation Mode
Insert Documents

Qdrant Collection Fixed Expression
By ID Enter ID...

Embedding Batch Size
200

Options
No properties
Add Option

FIGURE 12 – Création d'un Vector Store via n8n (option By ID)

7.6 Nœud 1.7 : Default Data Loader

Rôle : Segmentation en chunks avec overlap des documents pour optimiser leur traitement dans la chaîne de workflow.

7.6.1 Configuration

- **Type de données** : JSON
- **Mode** : Load Specific Data
- **Source de données** : `{ $('Extract from File1').item.json.data }`
- **Text Splitting** : Custom

7.6.2 Métadonnées

Le nœud configure les métadonnées associées aux données chargées :

- **Name** : source
- **Value** : `{ $('Extract from File1').item.json.fileName }`

FIGURE 13 – Noeud-Default Data Loader

7.7 Token Splitter

Rôle : Le nœud **Token Splitter** est un composant de prétraitement essentiel qui permet de diviser de longs documents en segments plus petits et gérables. Cette segmentation est cruciale pour optimiser le traitement des données textuelles dans les pipelines de traitement du langage naturel.

7.7.1 Paramètres de configuration

Le Token Splitter dispose de deux paramètres principaux configurables :

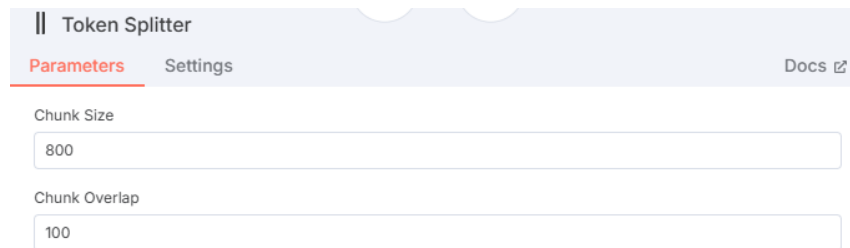
- **Chunk Size (Taille du segment)** : Définie à 800 tokens, cette valeur détermine la taille maximale de chaque segment généré. Cette configuration permet de maintenir un équilibre entre la cohérence contextuelle et les contraintes de traitement.
- **Chunk Overlap (Chevauchement)** : Configuré à 100 tokens, ce paramètre définit le nombre de tokens qui se chevauchent entre les segments consécutifs. Ce chevauchement garantit la préservation du contexte aux frontières des segments et évite la perte d'informations importantes.

7.7.2 Fonctionnement

Le processus de segmentation fonctionne selon le principe suivant :

1. Le texte d'entrée est analysé et divisé en tokens
2. Des segments de 800 tokens maximum sont créés

3. Chaque nouveau segment commence 700 tokens après le début du segment précédent (800 - 100 de chevauchement)
4. Cette approche garantit une continuité contextuelle entre les segments



The image shows a web interface for a 'Token Splitter'. At the top, there is a header bar with the title 'Token Splitter' and two tabs: 'Parameters' (which is active and underlined in red) and 'Settings'. To the right of the tabs is a link labeled 'Docs' with an external link icon. Below the header, there are two input fields. The first is labeled 'Chunk Size' and contains the value '800'. The second is labeled 'Chunk Overlap' and contains the value '100'.

FIGURE 14 – Noeud-Token Splitter

8 Partie 2 : Chat et Réponses

8.1 Architecture

Chat Message → AI Agent → OpenAI Chat Model → Simple Memory → Qdrant Vector Store → Embeddings OpenAI

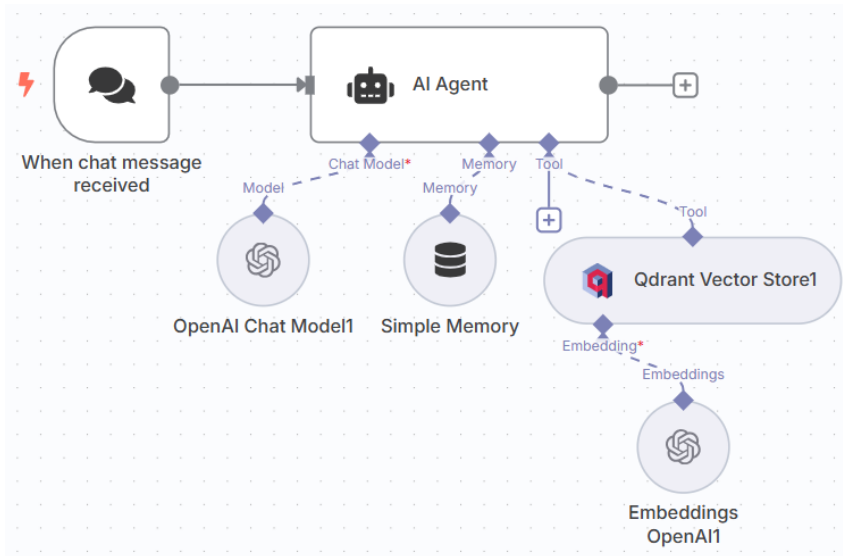


FIGURE 15 – Interface conversationnelle

Nœuds et Fonctions :

8.2 Nœud 2.1 : When chat message received

Rôle : Point de départ du workflow

Fonction : Déclenche le flux chaque fois qu'un utilisateur envoie un message dans le chat.

8.3 Nœud 2.2 : AI Agent

Rôle : Agent intelligent central

Fonction : Gère le traitement des messages entrants.

Utilise :

- Un modèle de langage pour comprendre et répondre.
- Une mémoire pour conserver le contexte des échanges.
- Un outil externe (Qdrant) pour enrichir ses réponses avec des données vectorielles.

Il combine :

- Chat Model → pour comprendre la requête
- Memory → pour garder le contexte du chat
- Tool → pour interagir avec des données vectorielles (Qdrant)

AI Agent Execute step Docs

Parameters Settings

Tip: Get a feel for agents with our quick [tutorial](#) or see an [example](#) of how this node works

Source for Prompt (User Message)
Connected Chat Trigger Node

Prompt (User Message)

```
fx {{ $json.chatInput }}
```

 bonjour parle moi du process du commit

Require Specific Output Format
☐

Options

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:
 Multi-document Analysis: For complex questions, search across ALL relevant documents
 Cross-referencing: When citing information, link related concepts from different documents

Add Option

FIGURE 16 – Noeud-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 :

- Multi-document Analysis : For complex questions, search across ALL relevant documents
- Cross-referencing : When citing information, link related concepts from different documents
- Process Flow : For process questions, show the complete flow across documents
- Documentation Integration : Connect different processes with their associated standards and guidelines

Mandatory Response Format :

Complete Answer [Provide comprehensive answer with cross-document insights]

Cross-Document Citations Process Flow :

Step 1 : [Document A] - [specific content]

Step 2 : [Document B] - [specific content]

Step 3 : [Document C] - [specific content]

Standards Referenced [List applicable standards from your organization's standard documents]

Source Metadata Primary Sources :

File : [file_name] | Chunk : [chunk_id] | Score : [score]

Supporting Sources :

File : [file_name] | Chunk : [chunk_id] | Score : [score]

Special Instructions for Process Questions :

- Always check process documentation, review procedures, and standard guidelines
- Show the relationship between different workflow steps and quality requirements
- Provide specific examples from the standards when relevant
- Cross-reference related procedures and dependencies

8.4 2.3. OpenAI Chat Model1

Rôle : Modèle de langage

Fonction : Fournit les capacités de compréhension et de génération de texte.

Spécificité : Utilise un modèle GPT-4o mini via OpenAI.

Relié à :

- AI Agent via Chat Model

2.4. Simple Memory

Rôle : Mémoire conversationnelle

Fonction : Enregistre l'historique des interactions pour fournir des réponses contextuelles.

Exemple : si l'utilisateur fait référence à un élément précédent, l'agent peut le retrouver.

Relié à :

- AI Agent via Memory

8.5 2.5. Qdrant Vector Store1

Rôle : Base de données vectorielle

Fonction : Permet de stocker et rechercher des embeddings (vecteurs) à grande vitesse.

Utilisé pour la RAG (Retrieval-Augmented Generation) :

L'agent cherche les documents les plus pertinents et s'en sert pour formuler une meilleure réponse.

Relié à :

- AI Agent via Tool
- Embeddings OpenAI1 (source des vecteurs)

Qdrant Vector Store1 Execute step

Parameters Settings Docs

Credential to connect with
QdrantApi account

Tip: Get a feel for vector stores in n8n with our [RAG starter template](#)

Operation Mode
Retrieve Documents (As Tool for AI Agent)

Description
Recherche dans la base de documents pour répondre aux questions

Qdrant Collection
From list documents_collection

Limit
4

Include Metadata
☒

Rerank Results
☐

Options
No properties
Add Option

FIGURE 17 – Noeud-Qdrant vector store

Paramètre	Valeur / Explication
Credential to connect with	QdrantApi account (compte configuré pour se connecter à Qdrant Cloud ou local)
Operation Mode	Retrieve Documents (As Tool for AI Agent) → ce mode permet à l'agent de questionner la base Qdrant
Description	Recherche dans la base de documents pour répondre aux questions
Qdrant Collection	documents_collection (nom de la collection d'embeddings dans Qdrant)
Limit	4 → limite à 4 documents pertinents retournés par requête
Include Metadata	Activé → les métadonnées (nom du fichier, ID de chunk, etc.) sont incluses dans les résultats
Rerank Results	Désactivé → les résultats ne sont pas reclassés par pertinence secondaire
Options	Aucun paramètre personnalisé ici

Table 1 : Configuration du nœud Qdrant Vector Store

9 Conclusion

Ce guide vous permet de mettre en place un agent RAG local à l'aide de n8n et Qdrant, pour indexer, rechercher et interroger intelligemment vos documents. Cette architecture est idéale pour tout projet de base documentaire augmentée par l'IA.