

Guide Complet - Workflow Graph RAG Agent avec n8n

Ahmed Aziz Ammar

Table des matières

1	Introduction	3
2	Pré-requis	3
3	Installation de Docker Desktop	3
4	Installation de n8n en mode Self-Hosted avec Docker	3
5	Accès à l'Interface n8n	4
6	Configuration de Neo4j en Cloud	4
7	Installation de Neo4j dans N8N	6
8	Vue d'ensemble du Workflow Graph RAG	7
8.1	Architecture générale	8
9	Importer le premier Workflow dans n8n	9
10	Workflow n°1 : Graph Database Workflow	9
10.1	Nœud 1.1 : Execute Workflow	10
10.2	Nœud 1.2 : Read/Write Files from Disk	10
10.3	Nœud 1.3 : Extract from File	11
10.4	Nœud 1.4 : Edit Fields1	11
10.5	Nœud 1.5 : Execute Command	12
10.6	Nœud Code 1.6 : Extract All informations (Important)	13
10.7	Nœud Code 1.7 : Nettoyage Bad Format	13
10.8	Nœud 1.8 : Edit Fields	14
10.9	Important : Création de la connexion avec le nœud Neo4j	15
10.10	Remplir les champs de connexion	16
10.11	Nœud 1.9 : Create Database (Neo4j)	17
10.12	Nœuds relationnels : Pairs Sans Doublons → AI Agent → Pairs bidirectionnelle → ExecuteQuery graphDb	18
11	Appuiez sur button Execute workflow et	23
12	Importer le deuxième Workflow dans n8n	24

13 Workflow n°2 : Graph RAG Agent	24
13.1 Architecture	24
13.2 Nœud 2.1 : When Chat Message Received	25
13.3 Nœud 2.2 : AI Agent	25
13.4 Nœud 2.3 : OpenAI Chat Model	27
13.5 Nœud 2.4 : Postgres Chat Memory	28
13.6 Nœud 2.5 : ExecuteQuery graphDb in Neo4j	29
14 Tester le Graph RAG	30
15 Conclusion	31

1 Introduction

Ce guide détaille les étapes pour installer et exécuter un workflow Graph RAG en local avec une base de données de graphes en Cloud, 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.

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

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.

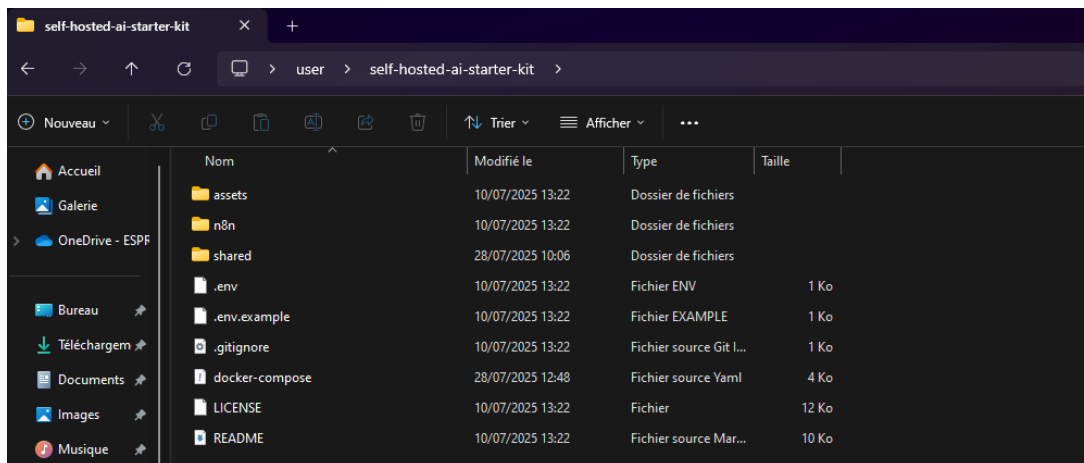


FIGURE 1 – Structure du dossier self-hosted-ai-starter-kit

5 Accès à l'Interface n8n

1. Après l'installation, ouvrir Docker Desktop et vérifier que le conteneur `self-hosted-ai-starter-kit` 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>

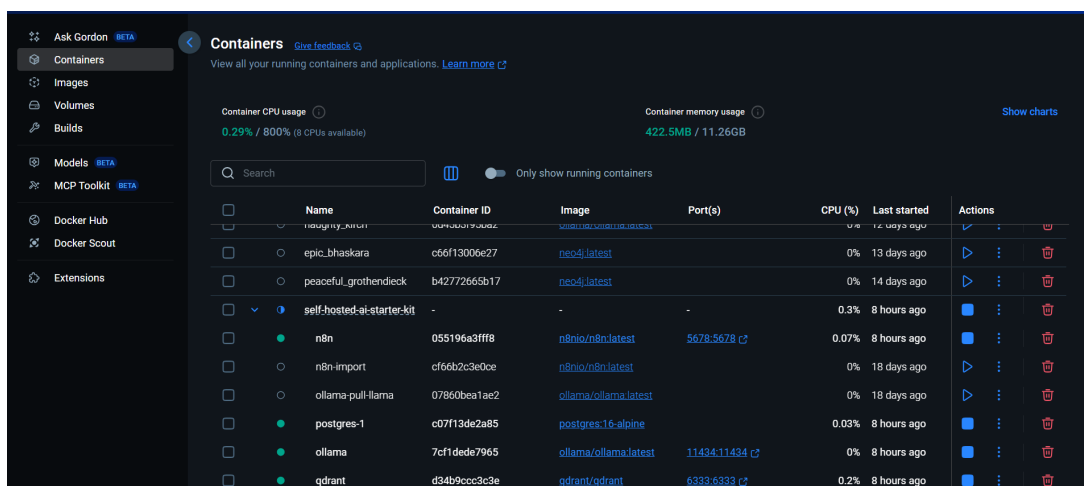


FIGURE 2 – Container self-hosted-ai-starter-kit dans Docker Desktop

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 Neo4j en Cloud

Pour mettre en place Neo4j en mode Cloud, nous utilisons la plateforme **Neo4j AuraDB**, qui permet d'héberger des graphes de données et d'y accéder via une API sécurisée.

- Rendez-vous sur le site officiel : <https://console.neo4j.io/>.
- Connectez-vous à l'aide de votre compte Gmail ou Outlook.
- Une fois connecté, vous serez redirigé vers une interface vous demandant de télécharger un fichier contenant vos identifiants de connexion :
 - **NEO4J_USERNAME**
 - **NEO4J_PASSWORD**
 - **NEO4J_URI**
- Téléchargez ce fichier et conservez-le précieusement. Attendez quelques minutes pour la création de l'instance gratuite (sous le nom "free instance" dans Neo4j Aura).

```
# Wait 60 seconds before connecting using these details, or login to https://console.neo4j.io to validate the Aura Instance is available
NEO4J_URI=neo4j+s://bf4fb4b6.databases.neo4j.io
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=wEaHt8y9mhN1uU9VtNkLmXMaWxwMJdm-pFddyYS8w
NEO4J_DATABASE=neo4j
AURA_INSTANCEID=bf4fb4b6
AURA_INSTANCENAME=trial instance
```

FIGURE 3 – Fichier d'identifiants de connexion Neo4j AuraDB

- Ensuite, vous serez automatiquement redirigé vers l'interface de gestion Neo4j Aura.

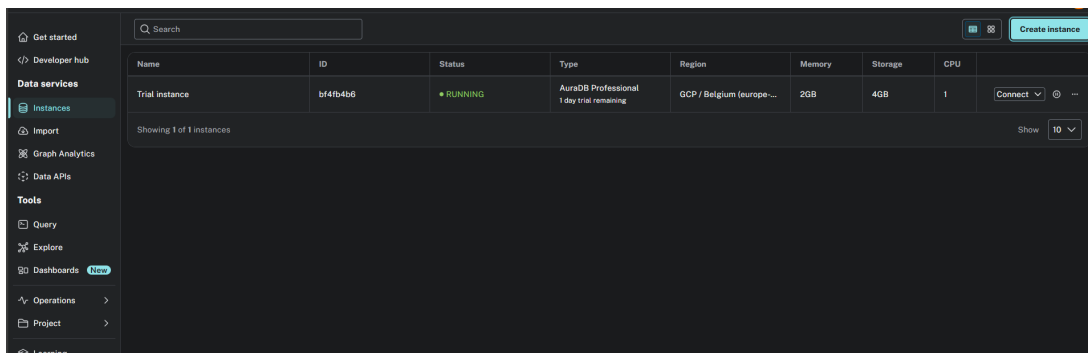


FIGURE 4 – Interface Neo4j Aura

- Appuyez sur Connect et choisissez Query. Vous serez automatiquement dirigé vers l'interface Cypher Query.

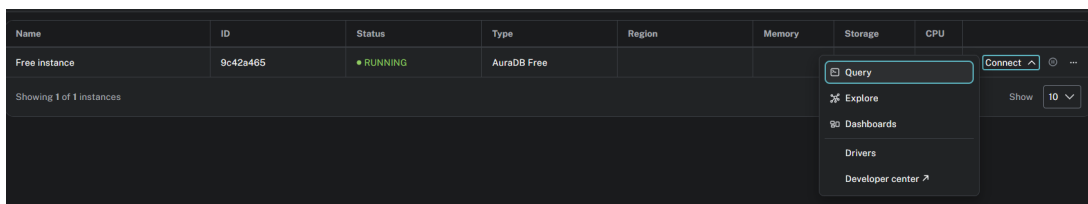


FIGURE 5 – Options de connexion Neo4j

- Vous pouvez exécuter des commandes Cypher pour visualiser les nœuds et leurs relations.
- **Exemple de requête :**
`MATCH (n:YourGraphDatabaseName) RETURN n LIMIT 50;`

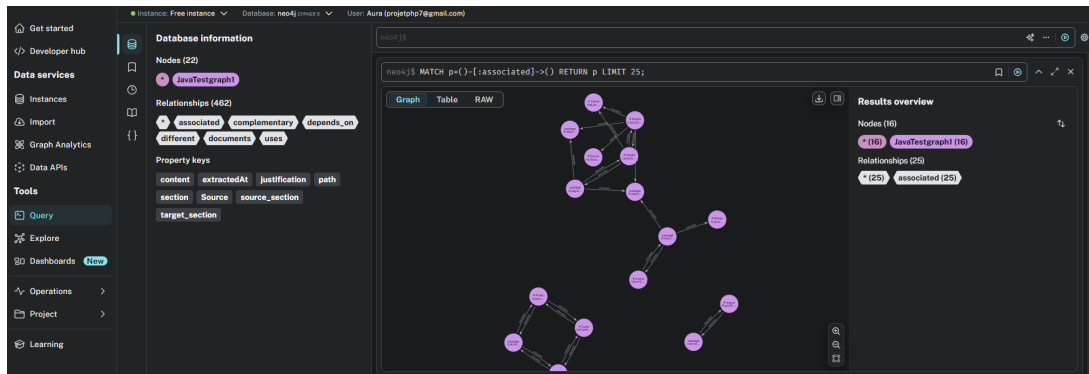


FIGURE 6 – Visualisation des relations entre les nœuds dans Neo4j

7 Installation de Neo4j dans N8N

Pour intégrer Neo4j dans N8N, suivez les étapes ci-dessous :

1. Accédez aux **Paramètres (Settings)** de N8N.
2. Sélectionnez l'onglet **Community Nodes**.
3. Cliquez sur le bouton **Installer un nœud (Install)**.
4. Dans le champ de recherche, saisissez : `n8n-nodes-neo4j`.
5. Cochez le paquet trouvé, puis cliquez sur **Installer**.

Captures d'écran illustratives

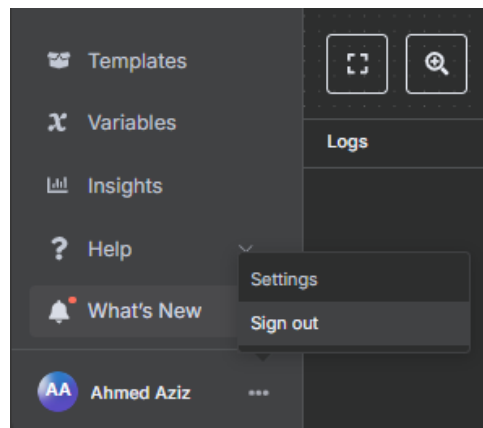


FIGURE 7 – Menu Settings dans N8N

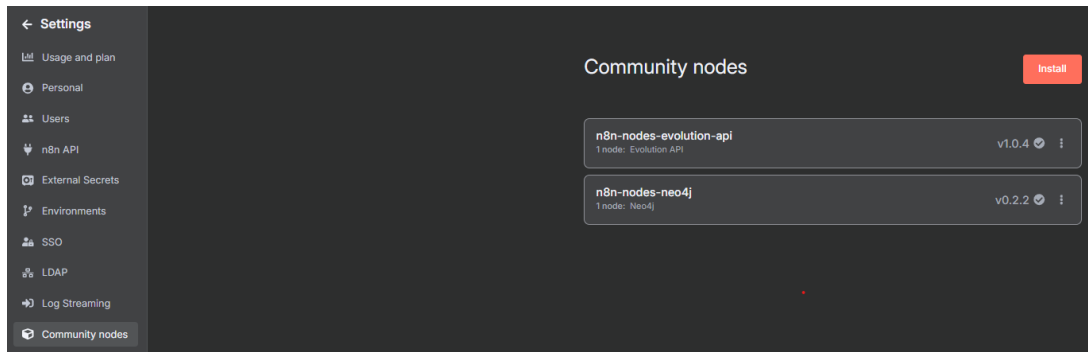


FIGURE 8 – Onglet Community Nodes pour l’installation d’extensions

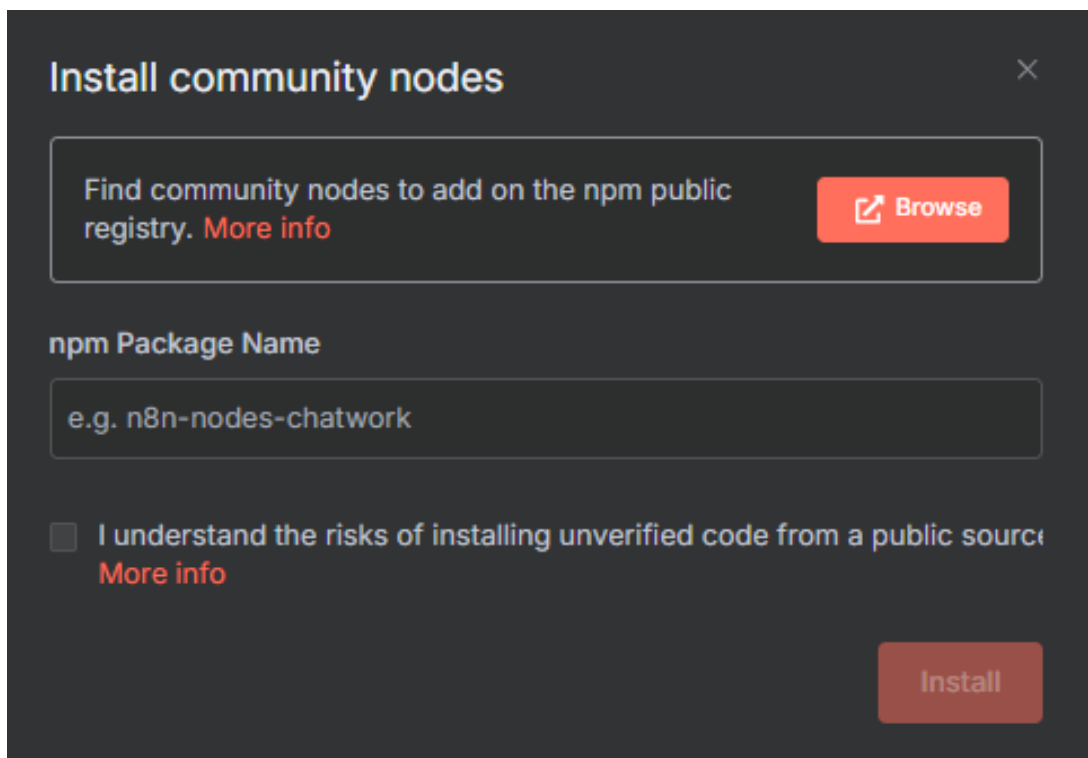


FIGURE 9 – Installation du nœud Neo4j (n8n-nodes-neo4j)

8 Vue d’ensemble du Workflow Graph RAG

Ces deux workflows ont pour objectifs principaux :

- Automatiser l’ingestion de documents (Java, Markdown) depuis des répertoires locaux.
- Indexer et stocker ces documents sous forme de nœuds dans une base de graphes Neo4j hébergée en cloud.
- Générer automatiquement des relations sémantiques entre les documents (par exemple : *documente*, *complémentaire*, *différent*).
- Permettre des requêtes avancées et des réponses synthétiques basées sur le graphe (approche **Graph RAG**).
- Fournir des réponses justifiées avec les métadonnées des documents sources (titre, section, date d’extraction, extrait de contenu).

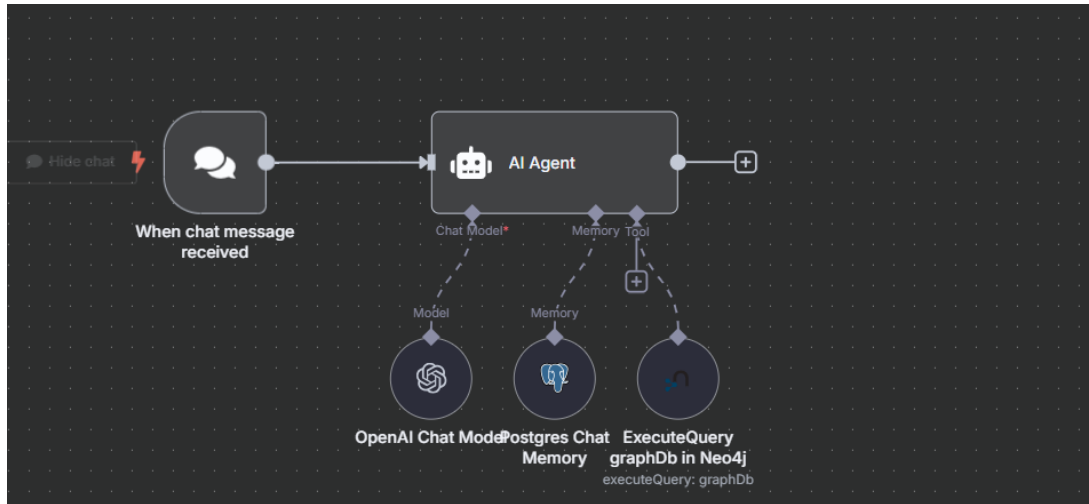


FIGURE 10 – Workflow de l’agent Graph RAG

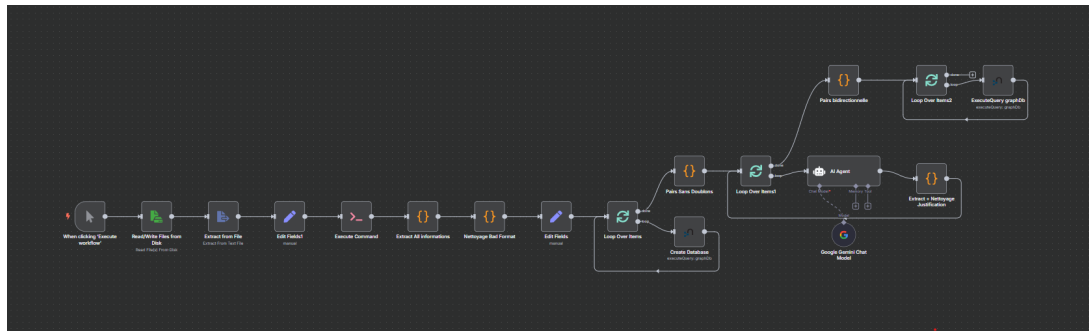


FIGURE 11 – Workflow d’ingestion et de configuration de la base de graphes Neo4j

8.1 Architecture générale

L’architecture repose sur deux workflows principaux interconnectés autour de la base de graphes Neo4j :

1. Workflow de Construction du Graphe Ce premier workflow a pour objectif d’ingérer et structurer la connaissance à partir des fichiers sources (Java, Markdown). Il comprend :

- **Lecture et extraction des documents** : récupération du contenu et des méta-données des fichiers.
- **Nettoyage et normalisation** : préparation des données pour éviter les erreurs d’insertion.
- **Création des nœuds et relations dans Neo4j** : insertion des fichiers sous forme de nœuds enrichis et génération automatique des relations sémantiques entre eux.

2. Workflow Agent Graph RAG Ce second workflow permet l’interaction conversationnelle avec la base :

- **Réception des requêtes utilisateur** via une interface chat.
- **Récupération des informations pertinentes** en interrogeant Neo4j à l’aide d’un agent RAG.
- **Raisonnement multi-documents** pour fournir une réponse synthétique basée sur les fichiers et leurs relations, sans hallucination.

— **Mémoire conversationnelle** pour assurer la continuité du dialogue.

Base de connaissances (Neo4j) Neo4j constitue le cœur de l'architecture, organisant les documents et leurs liens sous forme de graphes, permettant des recherches avancées et un raisonnement contextuel.

9 Importer le premier Workflow dans n8n

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

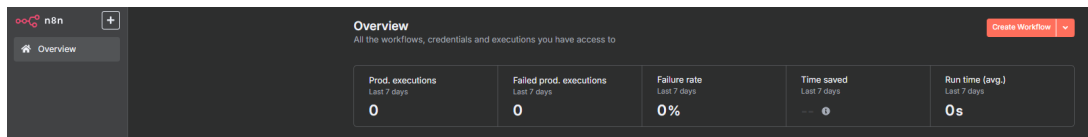


FIGURE 12 – Interface d'accueil N8N

4. Cliquer sur > Import from File

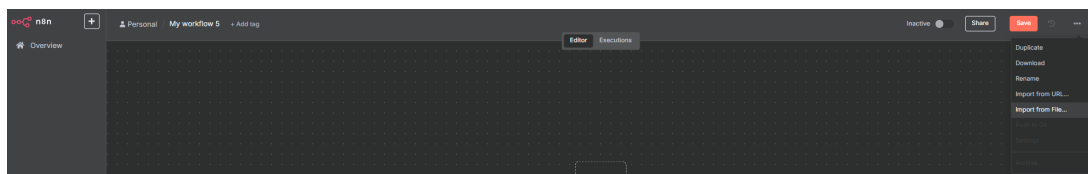


FIGURE 13 – Import depuis un fichier

5. Sélectionner le fichier Graph Database Resources: JAVA+MD.json
6. Save

10 Workflow n°1 : Graph Database Workflow

Architecture des Nœuds

Manual Trigger → Lecture des Fichiers → Extraction du Contenu → Nettoyage et Normalisation → Génération des Paires → Analyse des Relations par IA → Insertion dans Neo4j (Nœuds + Relations)

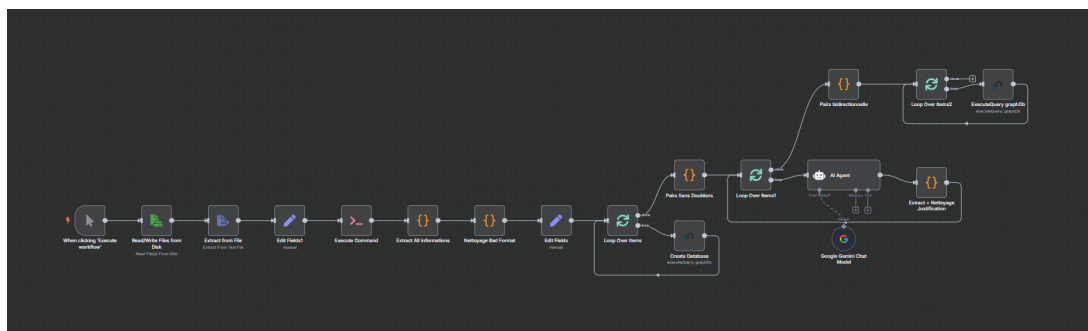


FIGURE 14 – Workflow de création du graphe dans Neo4j

Configuration des Nœuds

10.1 Nœud 1.1 : Execute Workflow

Rôle : Déclenchement manuel du processus d'indexation des fichiers et de génération des relations.

Configuration :

- **Type :** Manual Trigger
- **Utilisation :** Cliquer pour lancer le workflow qui lit les fichiers, nettoie le contenu, et insère les données dans Neo4j.

10.2 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 :** Pour lire tous les fichiers y compris dans les sous-dossiers avec n8n installé avec docker consulter le dossier self-hosted-ai-starter-kit et ajouter vos ressources dans dossier shared : /data/shared/YourPATH/**/*.{JAVA,md,markdown}

Explication du pattern glob :

- **** :** Parcours récursif des sous-dossiers
- ***** : Tous les fichiers
- **{JAVA,md,markdown} :** Extensions ciblées

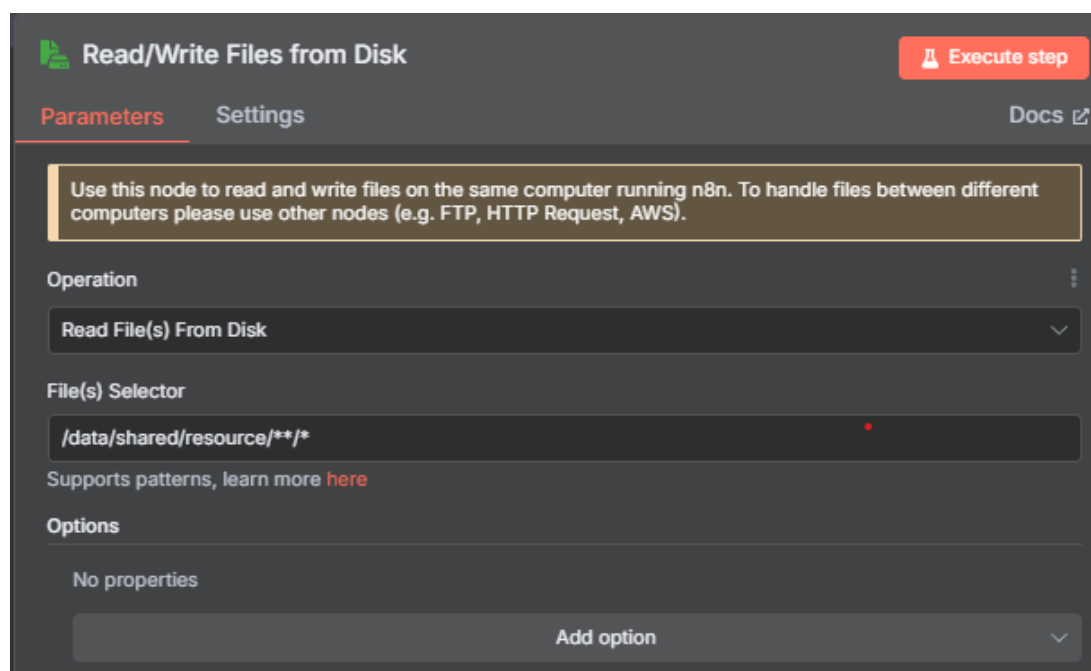
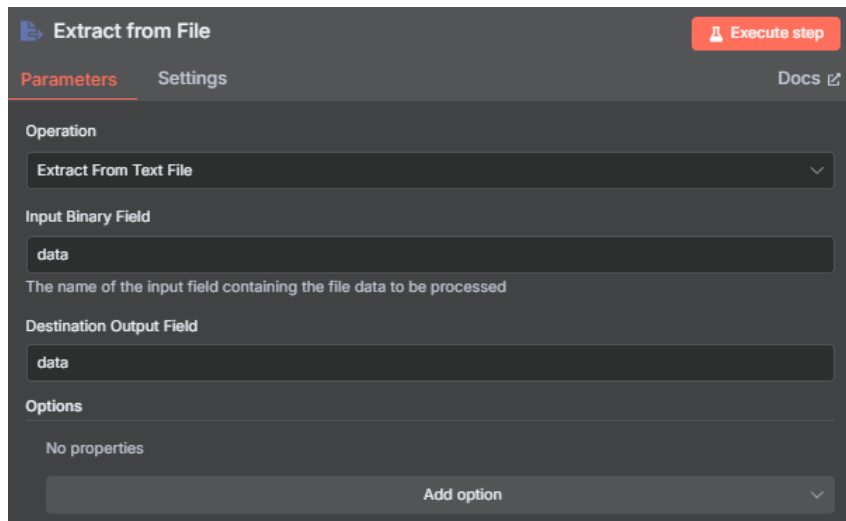


FIGURE 15 – Configuration de la lecture des fichiers

Les Resources dans Path dans la capture sont JAVA et MD pour cela il n'y a pas de .{JAVA,md,markdown}

10.3 Nœud 1.3 : Extract from File

Rôle : Extraction du contenu brut des fichiers (TXT, MD, Markdown, Java).



The screenshot shows the configuration interface for the 'Extract from File' node. It has a dark theme. At the top, there's a title bar with a file icon, the text 'Extract from File', and a red 'Execute step' button. Below the title bar are two tabs: 'Parameters' (active) and 'Settings'. A 'Docs' link with an external icon is on the right. The main configuration area is divided into sections: 'Operation' with a dropdown menu set to 'Extract From Text File'; 'Input Binary Field' with a text input containing 'data' and a description 'The name of the input field containing the file data to be processed'; 'Destination Output Field' with a text input containing 'data'; and 'Options' with the text 'No properties' and an 'Add option' button with a dropdown arrow.

FIGURE 16 – Configuration de l'extraction de fichiers

10.4 Nœud 1.4 : Edit Fields1

Rôle : Normalisation des champs extraits pour créer des propriétés comme `file_name` et `content`.

Edit Fields1 Execute step

Parameters Settings Docs

Mode
Manual Mapping

Fields to Set

file_name	A String	= {{ \$('Read/Write Files from Disk').item.json.fileName }}
content	A String	= {{ \$json.data }}

Drag input fields here or Add Field

Include Other Input Fields ☐

Options
No properties
Add option

FIGURE 17 – Configuration de l'édition des champs

10.5 Nœud 1.5 : Execute Command

Rôle : Liste tous les chemins de fichiers via la commande shell :

```
find /data/shared/YourPATH/*
```

Execute Command Execute step

Parameters Settings Docs

Execute Once ☒

Command
find /data/shared/resource/*

FIGURE 18 – Configuration de l'exécution de commande shell

10.6 Nœud Code 1.6 : Extract All informations (Important)

Rôle : Associe chaque chemin trouvé avec les métadonnées et le contenu extrait pour préparer les données en vue de leur insertion dans Neo4j.

ATTENTION : Il est impératif de ****modifier la ligne 32 du code**** pour spécifier le ****chemin racine de votre dossier contenant les ressources****. Sans cette modification, aucune donnée ne sera liée correctement dans la base de graphes.

Exemple :

```
const relativePath = fullPath.replace('/data/shared/resource/', '');
```

Astuce : Ce chemin doit pointer vers votre répertoire **/data/shared/** car il s'agit du volume de n8n avec docker.

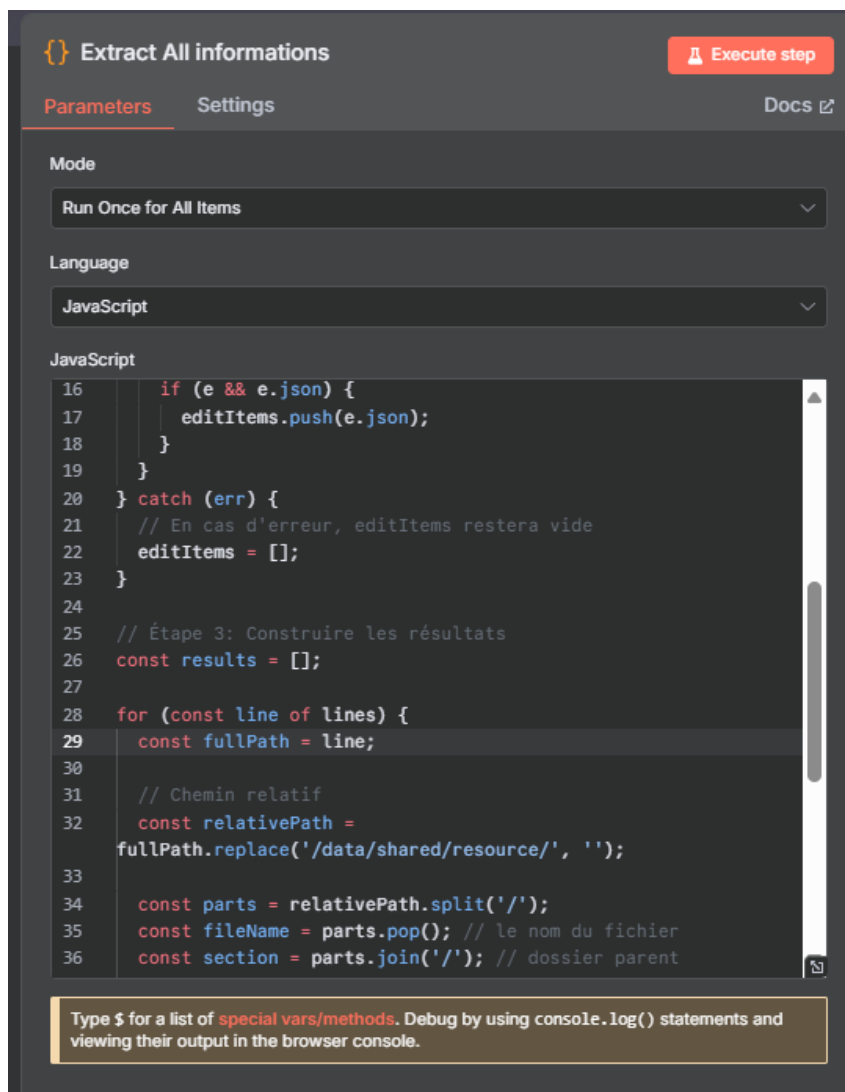


FIGURE 19 – Code d'extraction des informations

10.7 Nœud Code 1.7 : Nettoyage Bad Format

Rôle : Nettoyage du contenu extrait pour enlever les caractères indésirables (“”, retours à la ligne, etc.) et échapper les caractères avant insertion dans Neo4j.

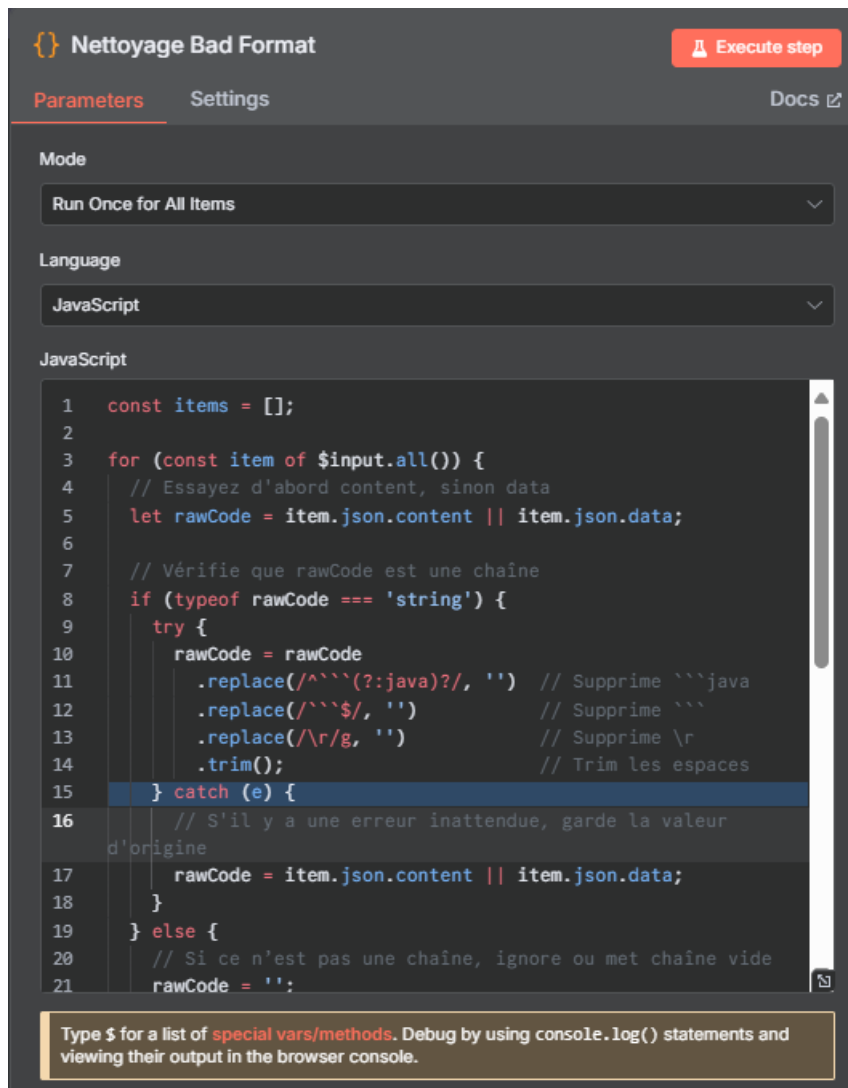


FIGURE 20 – Code de nettoyage du format

10.8 Nœud 1.8 : Edit Fields

Rôle : Prépare les propriétés finales pour Neo4j (nom du fichier, section, chemin, contenu nettoyé).

The screenshot shows the 'Edit Fields' configuration window with the 'Parameters' tab selected. The 'Mode' is set to 'Manual Mapping'. Under 'Fields to Set', five fields are configured:

Field Name	Value
file_name	{{ \$json.file }}
section	{{ \$json.section }}
PATH	{{ \$json.path }}
content	{{ \$json.cleanedCode }}
extractedAT	{{ \$json.extractedAt }}

At the bottom, there is a dashed box containing the text 'Drag input fields here or **Add Field**'.

FIGURE 21 – Configuration finale des champs

10.9 Important : Création de la connexion avec le nœud Neo4j

Dans le premier nœud Neo4j, allez dans le premier champ + **Create new Credential**

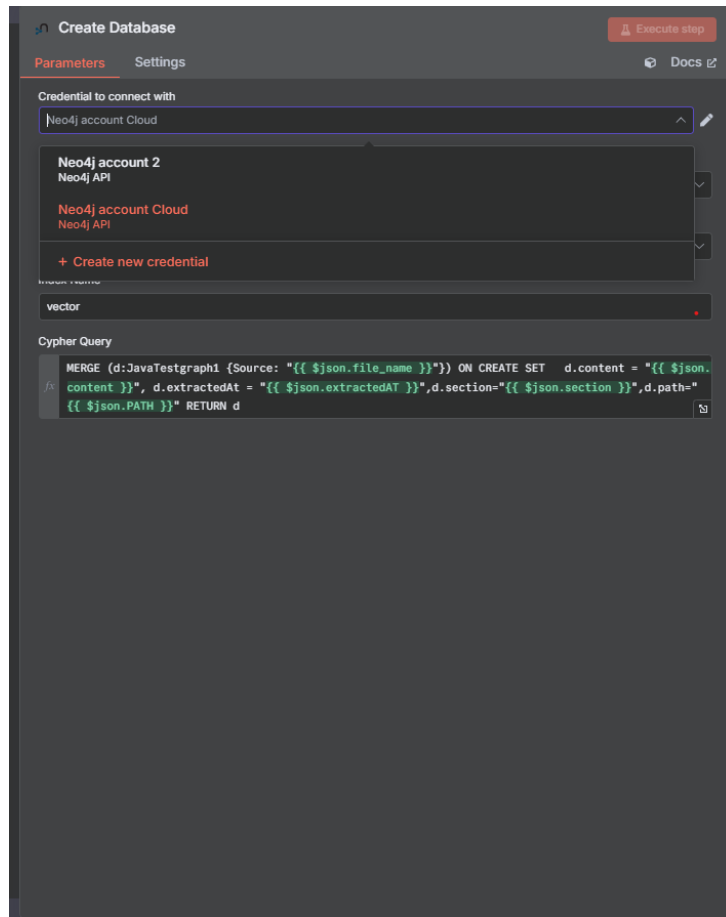


FIGURE 22 – Création d'un identifiant d'accès à Neo4j dans N8N

10.10 Remplir les champs de connexion

Renseignez les champs suivants avec les valeurs récupérées depuis le fichier téléchargé lors de la création de l'instance gratuite dans Neo4j Aura :

- **Host** : NEO4J_URI (exemple : neo4j+s://xxxx.databases.neo4j.io)
- **Username** : NEO4J_USERNAME (généralement neo4j)
- **Password** : NEO4J_PASSWORD (généré automatiquement lors de la création)
- **Database** : NEO4J_DATABASE (souvent neo4j)

Neo4j account Cloud
Neo4j API

Need help filling out these fields? [Open docs](#)

Connection

Sharing

Details

Connection URI *

neo4j+s://9c42a465.databases.neo4j.io

Username *

neo4j

Password *

.....

Database *

neo4j

Enterprise plan users can pull in credentials from external vaults. [More info](#)

FIGURE 23 – Configuration de la connexion à Neo4j dans N8N

Important : Une fois l'identifiant créé, il est automatiquement utilisé dans les autres nœuds Neo4j et le même principe s'applique aux autres nœuds de N8N.

10.11 Nœud 1.9 : Create Database (Neo4j)

Rôle : Insère les nœuds (fichiers) dans la base Neo4j avec les propriétés suivantes :

- **Source :** Nom du fichier
- **content :** Contenu nettoyé
- **section :** Dossier parent
- **path :** Chemin complet

Requête Cypher :

```
MERGE (d:graphDatabaseName {Source: "{{ $json.file_name }}"})
ON CREATE SET d.content = "{{ $json.content }}",
              d.extractedAt = "{{ $json.extractedAT }}",
              d.section = "{{ $json.section }}",
              d.path = "{{ $json.PATH }}"

RETURN d
```

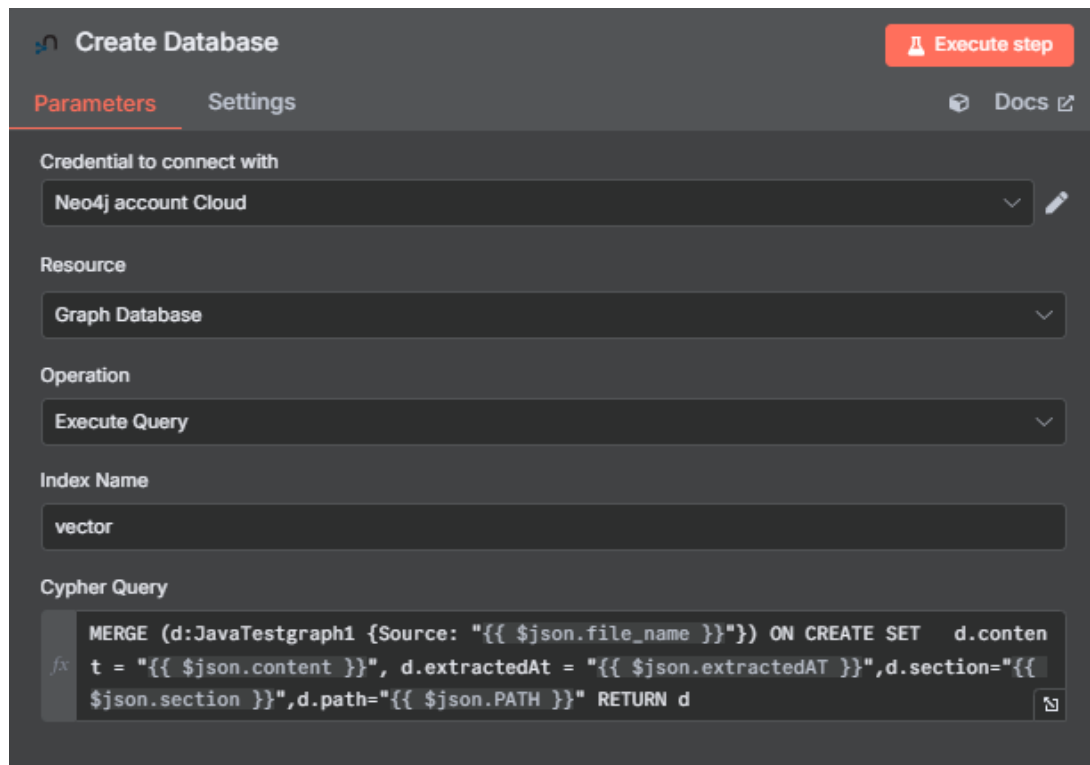


FIGURE 24 – Nœud Execute Query pour la création des nœuds

10.12 Nœuds relationnels : Pairs Sans Doublons → AI Agent → Pairs bidirectionnelle → ExecuteQuery graphDb

Rôle global : Construire les relations sémantiques entre fichiers à partir de leurs contenus et sections, et les insérer dans Neo4j.

Chaîne de traitement :

1. **Pairs Sans Doublons (nœud Code)** Génère toutes les paires (A, B) sans répétition (pas de $A \rightarrow A$ et pas de doublons inversés $B \rightarrow A$). Chaque paire inclut :
 - **source** : {file, path, section, content}
 - **target** : {file, path, section, content}

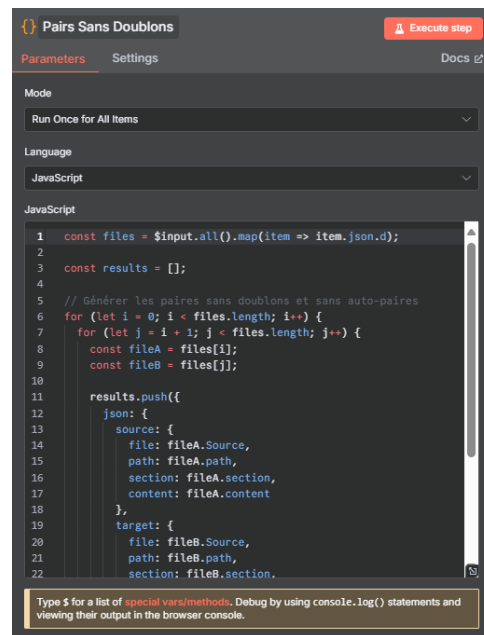


FIGURE 25 – Code de génération des paires sans doublons

2. **AI Agent (LangChain)** Analyse chaque paire (*source*, *target*) pour déterminer :
 - *type* : Type de relation (complementary, uses, documents, associated, etc.)
 - *justification* : Explication concise de la relation
 - *source_section* et *target_section*
 - *source* et *target* (noms des fichiers)
 - *extractedAt* : Horodatage de génération

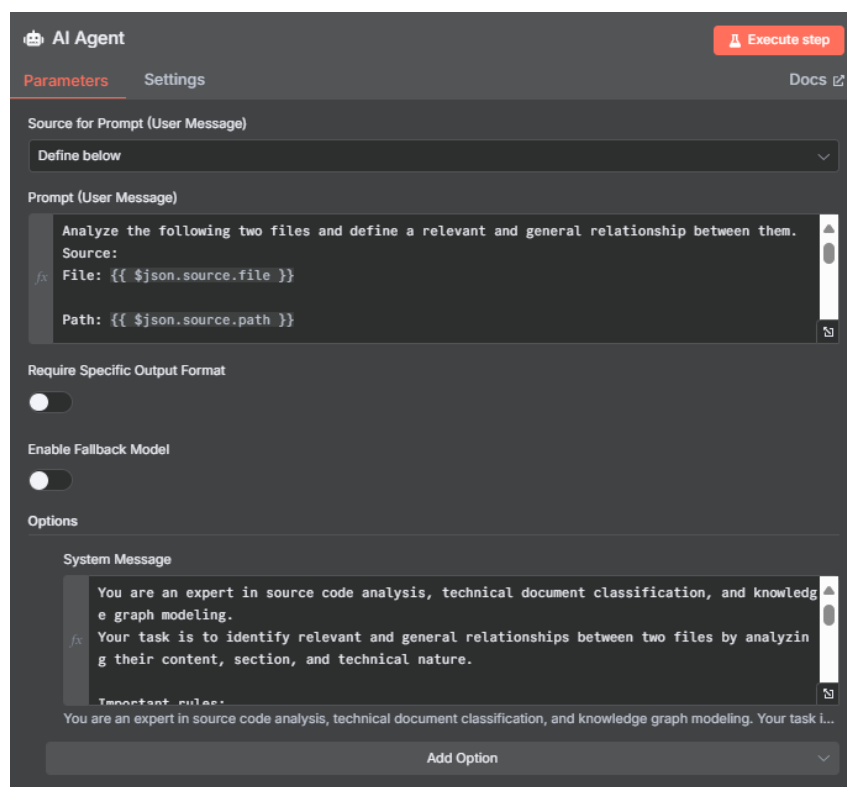


FIGURE 26 – Configuration de l'agent IA pour l'analyse des relations

3. Chat Model Gemini Flash 2.5 (Configuration) — Création de connexion

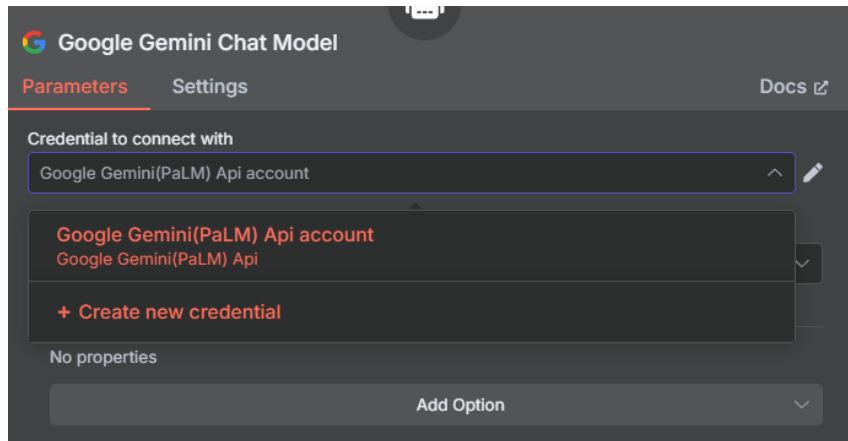


FIGURE 27 – Création d'un identifiant d'accès à Google Gemini dans N8N

4. Remplir le champ API KEY par la valeur de clé générée par Google AI Studio <https://aistudio.google.com/app/apikey>

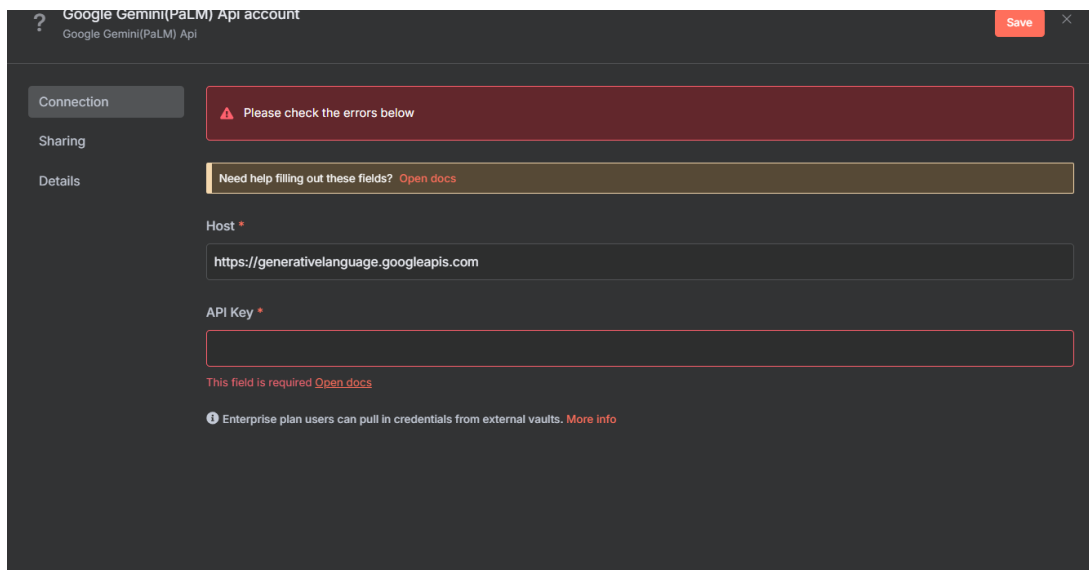


FIGURE 28 – Configuration de la connexion Chat Model Gemini Google

5. **Extract + Nettoyage Justification (nœud Code)** Ce nœud a deux fonctions principales :

- Extraction** : Transforme la sortie brute du modèle IA (un seul bloc JSON encapsulé dans du texte) en un objet structuré contenant :
 - **source** et **target** : noms des fichiers
 - **source_section** et **target_section** : sections correspondantes
 - **relationshipType** : type de relation déterminé par le modèle IA
 - **justification** : justification nettoyée
 - **extractedAt** : horodatage
- Nettoyage** : Supprime les caractères spéciaux, échappe les guillemets, et formate la justification pour une insertion sécurisée dans Neo4j.

```

1 // Fonction de nettoyage pour Neo4j
2 function cleanTextForCypher(text) {
3   if (!text) return "";
4
5   return text
6     .replace(/\\/g, '\\\\') // Échapper \
7     .replace(/"/g, '\\"') // Échapper "
8     .replace(/'/g, '\\\'') // Échapper '
9     .replace(/\n/g, ' ') // Remplacer retour ligne par
    espace
10    .replace(/\r/g, ' ') // Remplacer retour chariot
11    .replace(/\t/g, ' ') // Remplacer tabulation
12    .replace(/[^\x00-\x1f\x7f]/g, '') // Supprimer caractères
    de contrôle
13    .replace(/\s+/g, ' ') // Réduire les espaces
    multiples
14    .trim();
15 }
16
17 // Entrée
18 const raw = $input.first().json.output || "";
19

```

Type \$ for a list of special vars/methods. Debug by using console.log() statements and viewing their output in the browser console.

FIGURE 29 – Code d’extraction et de nettoyage des justifications

6. **Pairs bidirectionnelle (nœud Code)** À partir de la sortie du modèle IA pour (A, B) , génère automatiquement une relation inverse (B, A) pour garantir la bidirectionnalité. Chaque relation inverse conserve :
- **relationshipType** : identique à l’original
 - **justification** : identique ou annotée comme relation inversée

```

1 const inputItems = $input.all();
2
3 const outputItems = inputItems.flatMap(item => {
4   const {
5     source,
6     target,
7     source_section,
8     target_section,
9     relationshipType,
10    justification
11  } = item.json;
12
13  return [
14    // Relation normale
15    {
16      json: {
17        source,
18        target,
19        source_section,
20        target_section,
21        relationshipType,
22        justification

```

Type \$ for a list of special vars/methods. Debug by using console.log() statements and viewing their output in the browser console.

FIGURE 30 – Code de génération des paires bidirectionnelles

7. **ExecuteQuery graphDb (Neo4j)** Insère les relations entre nœuds existants (créés lors de l'étape précédente d'insertion des fichiers) via la procédure :

```
MATCH (src:YourGraphDatabaseName {Source: "{{ $json.source }}"})
MATCH (tgt:YourGraphDatabaseName {Source: "{{ $json.target }}"})
WITH src, tgt,
    "{{ $json.relationshipType }}" AS relType,
    "{{ $json.justification }}" AS justification,
    "{{ $json.source_section }}" AS sourceSection,
    "{{ $json.target_section }}" AS targetSection
WHERE src IS NOT NULL AND tgt IS NOT NULL
CALL apoc.create.relationship(
    src,
    relType,
    {
        justification: justification,
        source_section: sourceSection,
        target_section: targetSection
    },
    tgt
)
YIELD rel
RETURN src, rel, tgt
```

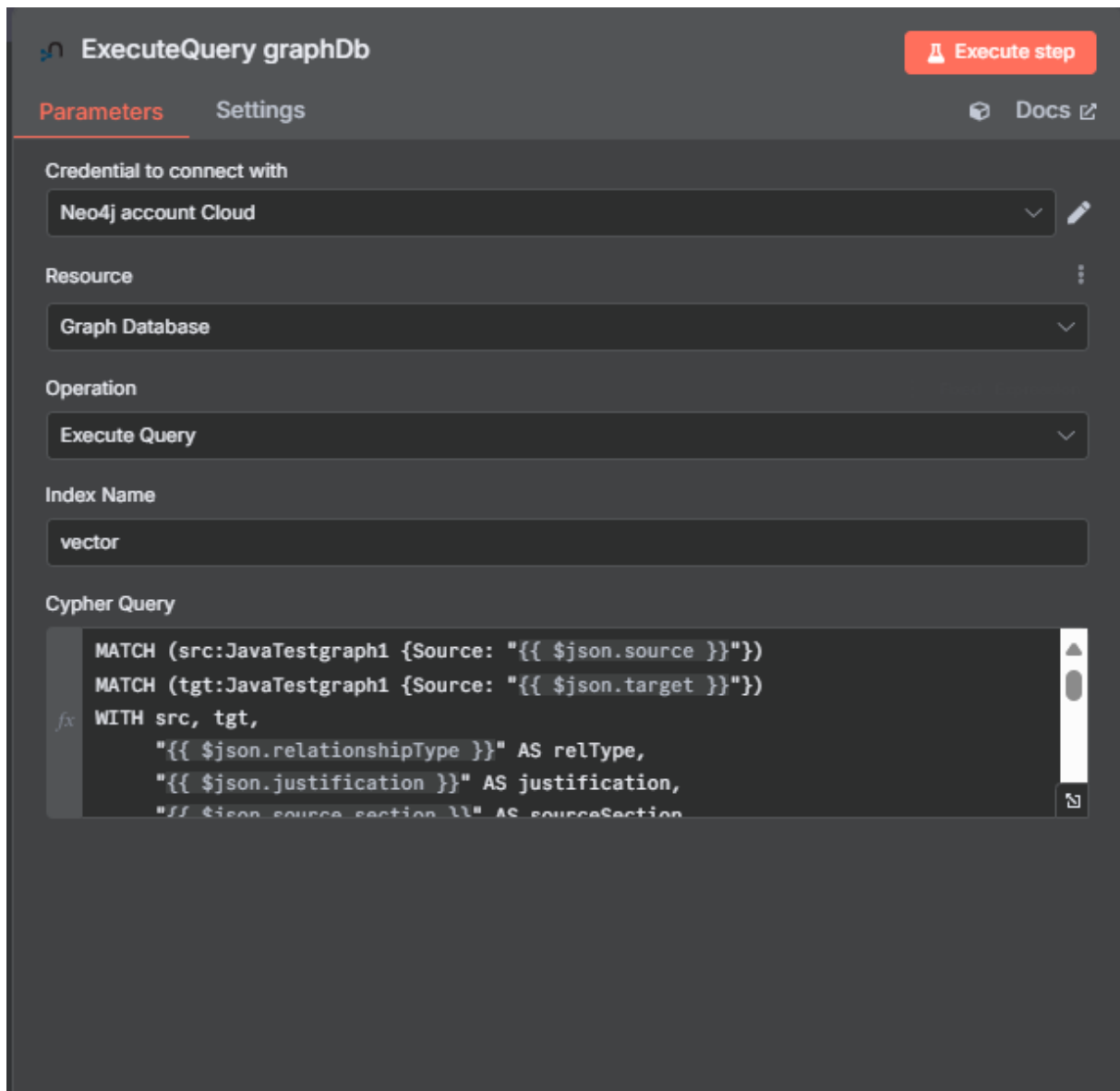


FIGURE 31 – Configuration du nœud Execute Query pour créer les relations

Résumé des données insérées :

- source → Nom du fichier source
- target → Nom du fichier cible
- relationshipType → Type général de la relation
- justification → Raisonnement du modèle IA
- source_section, target_section → Sections correspondantes
- extractedAt → Date de génération

11 Appuiez sur button Execute workflow et

!!Important Attendez le workflow jusqu'à l'exécution termine elle peut prendre assez de temps selon le nombre de vos Ressources

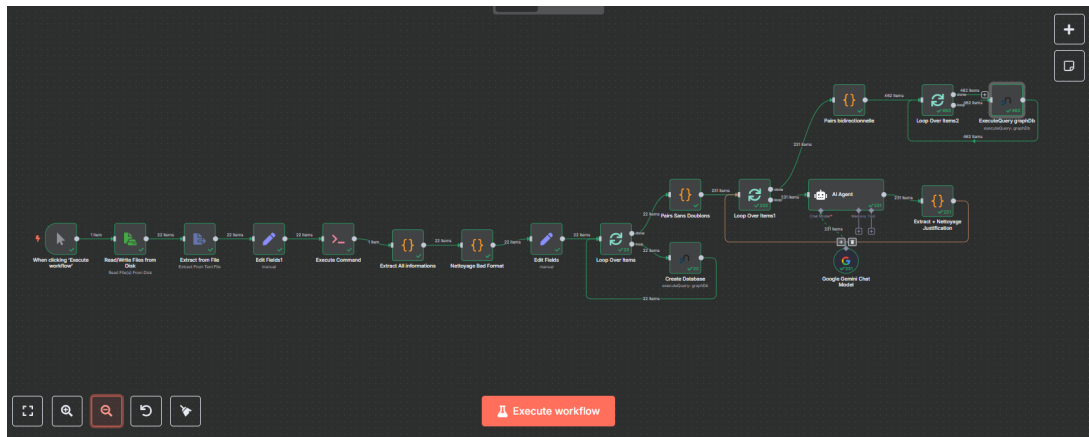


FIGURE 32 – Graph Database Workflow exécuté

12 Importer le deuxième Workflow dans n8n

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

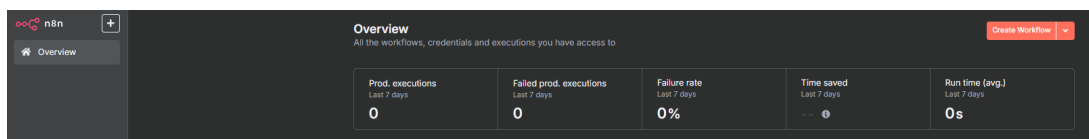


FIGURE 33 – Interface d'accueil N8N

4. Cliquer sur > Import from File

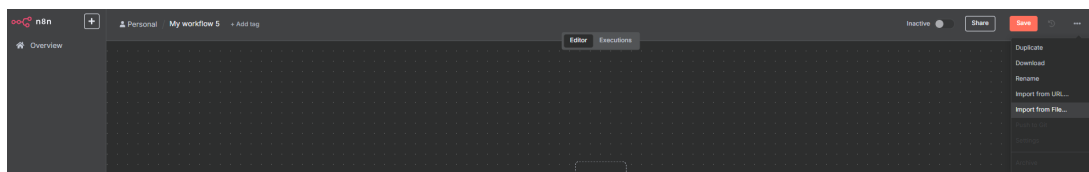


FIGURE 34 – Import depuis un fichier

5. Sélectionner le fichier Graph RAG AGENT new resource JAVA,MD.json
6. Save

13 Workflow n°2 : Graph RAG Agent

13.1 Architecture

Chat Message → AI Agent → OpenAI Chat Model → Postgres Memory → Neo4j Graph Database (Tool)

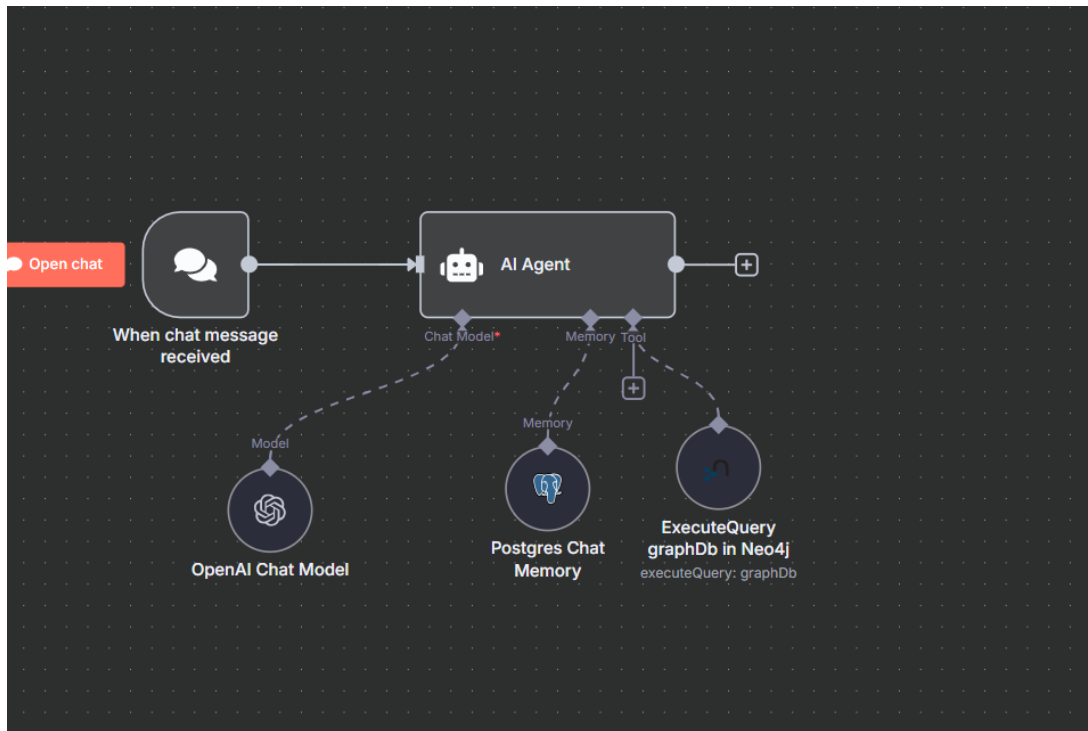


FIGURE 35 – Interface conversationnelle pour l’agent Graph RAG basé sur Neo4j

Description générale : L’agent intelligent (AI Agent) reçoit une question, interroge la base de graphes Neo4j pour récupérer les fichiers Java et documentation liés, et génère une réponse enrichie en expliquant les relations entre les fichiers.

13.2 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.

13.3 Nœud 2.2 : AI Agent

Rôle : Agent Graph RAG principal. **Fonction :** Analyse la requête utilisateur, interagit avec Neo4j et compose une réponse.

Utilise :

- Un modèle de langage pour comprendre et générer la réponse.
- Une mémoire (Postgres) pour conserver le contexte des conversations.
- Un outil Neo4j pour récupérer les fichiers et leurs relations.

Combinaison :

- **Chat Model** → compréhension et génération
- **Memory** → contexte conversationnel
- **Tool (Neo4j)** → recherche des nœuds et relations pertinentes

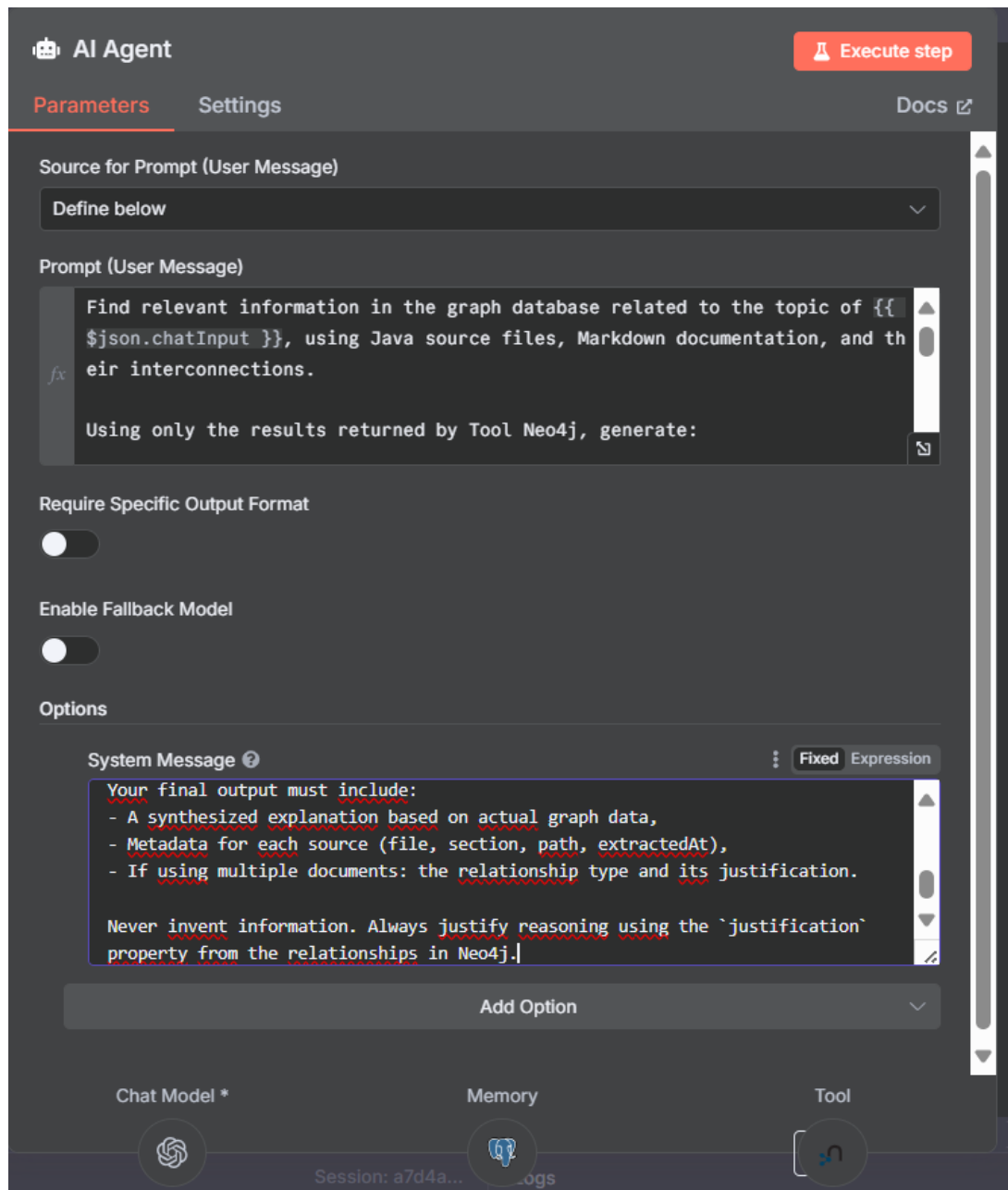


FIGURE 36 – Configuration de l'agent AI connecté à Neo4j, Postgres et OpenAI

System Message :

You are a Graph RAG AI agent specialized in analyzing Java source files and their related Markdown documentation stored in a Neo4j knowledge graph.

Your main tool is the Neo4j database (Tool Neo4j), which you must query to extract relevant files and their semantic relationships.

Rules :

- Only use data returned from Neo4j (no hallucinations).
- Prioritize relationships : **documente, complémentaire**.
- Include metadata : file name, section, path, extractedAt, snippet of content.
- Respond in the language of the query.

Format attendu :

- Réponse synthétique et raisonnée.

- Liste des fichiers utilisés avec leur relation et justification.
- Citations multi-documents si nécessaire.

13.4 Nœud 2.3 : OpenAI Chat Model

Rôle : Chat GPT 4o mini **Fonction** : Génère la réponse basée sur les données du graphe. Relié à :

- AI Agent via Chat Model.

Configuration :

- Ajoutez l'option *Sampling Temperature* à 0,0 afin de réduire le taux d'hallucination de l'agent IA à zéro.
- Créer new Credentiel

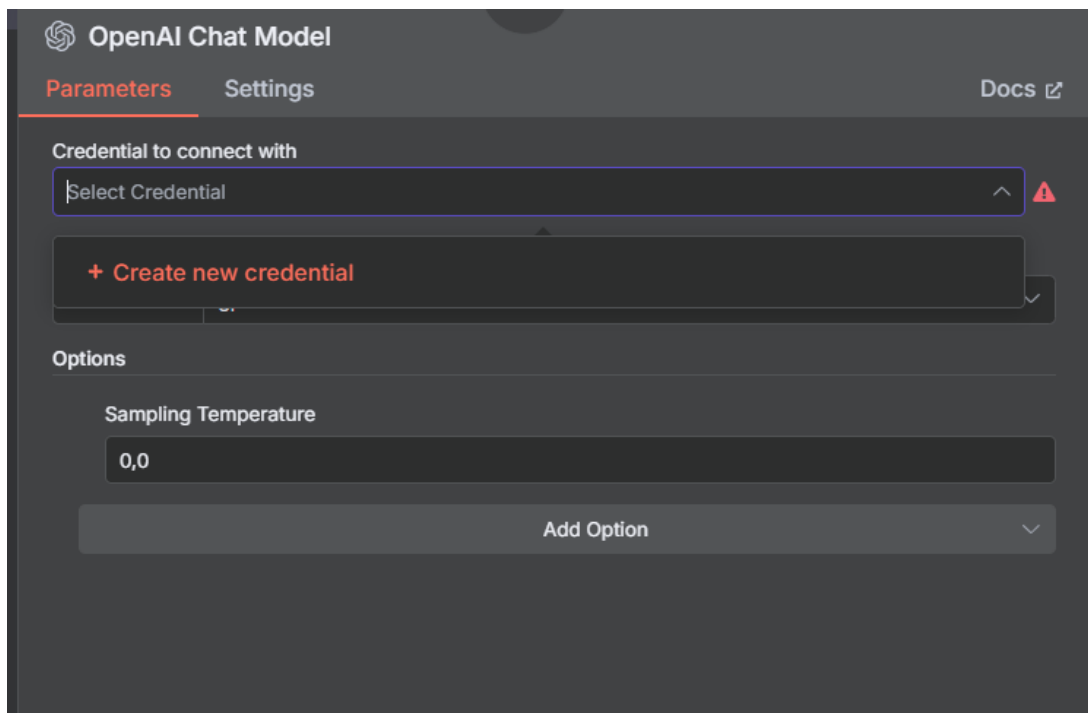


FIGURE 37 – Création de Connexion avec OpenAI

- Remplir les champs de connexion : Obtenir Clé Open AI (payant) par le créer dans lien platform.openai.com/docs/quickstart

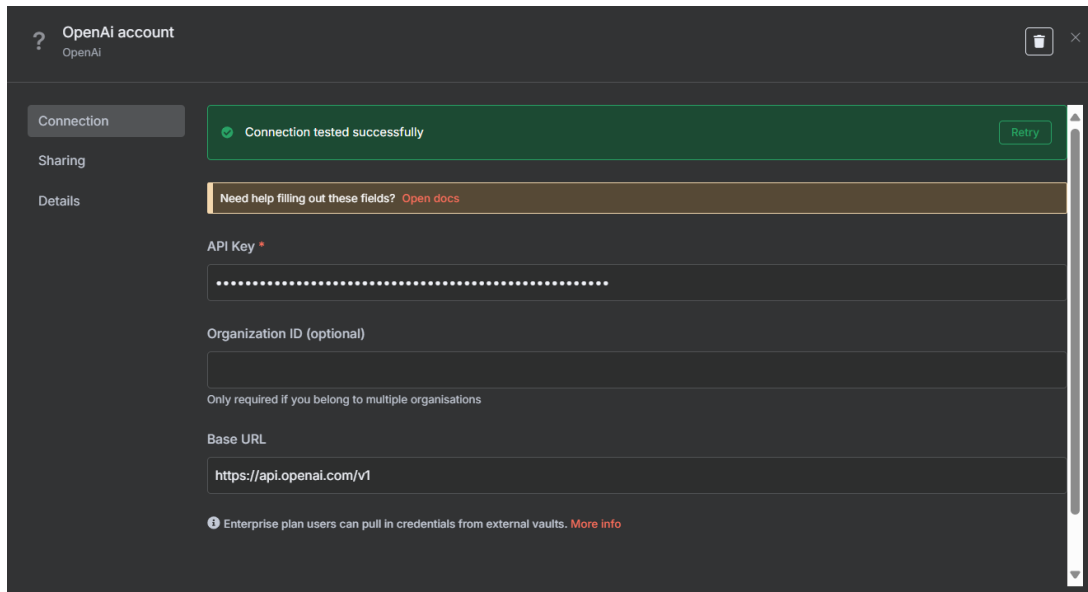


FIGURE 38 – Champs de Connexion OPEN AI

13.5 Nœud 2.4 : Postgres Chat Memory

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

Configuration :

- **Type de mémoire** : Postgres
- **Connexion** : Requiert la création d'un identifiant d'accès à la base de données PostgreSQL (Credential)
- **Paramètres à renseigner** lors de la configuration de l'identifiant :
- Ces variables sont définies dans le fichier `.env` situé dans le dossier `self-hosted-ai-starter-kit`
 - **Host** : postgres (nom du service dans `docker-compose.yml`)
 - **Port** : 5432 (port par défaut de PostgreSQL)
 - **Database** : `$POSTGRES_DB`
 - **User** : `$POSTGRES_USER`
 - **Password** : `$POSTGRES_PASSWORD`

FIGURE 39 – Configuration de la connexion PostgreSQL

— Exemple de contenu du fichier `.env` :

```
POSTGRES_USER=root
POSTGRES_PASSWORD=password
POSTGRES_DB=n8n

N8N_ENCRYPTION_KEY=super-secret-key
N8N_USER_MANAGEMENT_JWT_SECRET=even-more-secret
N8N_DEFAULT_BINARY_DATA_MODE=filesystem

# For Mac users running OLLAMA locally
# See https://github.com/n8n-io/self-hosted-ai-starter-kit?tab=readme-ov-file#for-mac--apple-silicon-users
# OLLAMA_HOST=host.docker.internal:11434
```

FIGURE 40 – Contenu par défaut du fichier `.env`

Connexion : Ce nœud est relié à l’agent conversationnel (AI Agent) via l’option Memory.

13.6 Nœud 2.5 : ExecuteQuery graphDb in Neo4j

Rôle : Interroger la base Neo4j pour extraire les fichiers et leurs relations. **Fonction** : Exécute la requête Cypher pour obtenir les nœuds et relations.

Requête utilisée :

```
MATCH (n:YourGraphDatabaseName) OPTIONAL MATCH (n)-[r]->(m) OPTIONAL MATCH (m)-[r2]->
```

The screenshot shows the 'ExecuteQuery graphDb in Neo4j' configuration window. It has a dark theme and includes tabs for 'Parameters' and 'Settings'. A red 'Execute step' button is in the top right. The 'Parameters' tab is active, showing several configuration fields:

- Credential to connect with:** A dropdown menu set to 'Neo4j account Cloud'.
- Tool Description:** A dropdown menu set to 'Set Automatically'.
- Resource:** A dropdown menu set to 'Graph Database'.
- Operation:** A dropdown menu set to 'Execute Query'.
- Index Name:** A text input field containing 'vector'.
- Cypher Query:** A text area containing the query:


```
MATCH (n:JavaTestgraph1) OPTIONAL MATCH (n)-[r]->(m) OPTIONAL MATCH (m)-[r2]
->(n) RETURN n, r, m, r2
```

Below the Cypher Query field, a preview of the query is shown: `MATCH (n:JavaTestgraph1) OPTIONAL MATCH (n)-[r]->(m) OPTIONAL MATCH (m)-[r2]->(n) RETURN n, r, ...`

FIGURE 41 – Configuration du nœud Neo4j pour l’exécution de requêtes

14 Tester le Graph RAG

Après avoir terminé la configuration, vous pouvez tester le fonctionnement du Graph RAG en saisissant votre requête dans l’interface n8n, puis en l’envoyant pour obtenir une réponse.

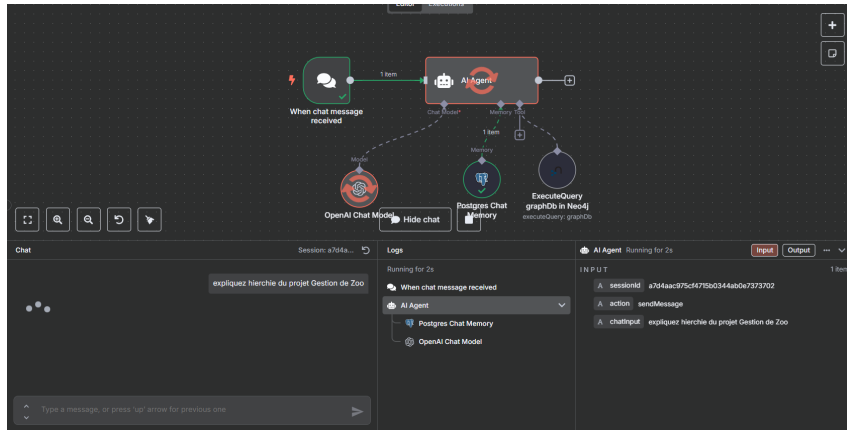


FIGURE 42 – Exemple d’envoi de requête via n8n pour interroger le Graph RAG

15 Conclusion

Ce guide vous a permis de mettre en place un système Graph RAG (Retrieval-Augmented Generation) complet en combinant les outils **n8n** pour l’orchestration, **Neo4j** pour la gestion de la base de données de graphes, et des modèles d’IA avancés pour l’analyse sémantique et la génération de réponses.

Cette architecture offre plusieurs avantages majeurs :

Avantages de l’approche Graph RAG :

- **Contextualisation avancée** : Les relations sémantiques entre documents permettent une compréhension plus profonde du contexte et des interconnexions.
- **Réponses enrichies** : Les réponses générées s’appuient sur les relations entre fichiers, offrant une perspective multi-documents cohérente.
- **Traçabilité complète** : Chaque réponse est accompagnée des métadonnées des sources (fichier, section, justification des relations).
- **Évolutivité** : L’architecture modulaire permet d’ajouter facilement de nouveaux types de documents et de relations.

Applications pratiques :

- **Documentation intelligente** : Analyse automatique de bases de code Java avec leur documentation Markdown associée.
- **Assistant de développement** : Aide à la compréhension des relations entre composants logiciels.
- **Système de veille technologique** : Analyse des liens entre différents documents techniques et de recherche.
- **Base de connaissances d’entreprise** : Organisation et interrogation intelligente de la documentation interne.

Extensibilité future : Le système peut être étendu pour inclure d’autres types de fichiers (PDF, Word, Excel), d’autres bases de données de graphes, ou intégrer des modèles d’IA spécialisés selon les besoins spécifiques de votre organisation.

Cette solution constitue une base solide pour développer des applications d’intelligence artificielle avancées tout en gardant le contrôle total sur vos données et en bénéficiant de la puissance du raisonnement basé sur les graphes de connaissances.