Jumeau numérique · Sous-section

Vectoriser ton coffre avec QMD (optionnel)

Installer QMD pour faire de la recherche sémantique dans tout ton vault — retrouver n'importe quelle info en moins d'une seconde.

Source : https://prisme.one/plateforme/ia-juillet-2026/semaine-1-vectoriser-coffre

---

Vectoriser ton coffre avec QMD

Vectoriser son coffre avec QMD

LOOM

Vectoriser son coffre avec QMD

https://www.loom.com/share/dab0cef6d37c4bc98a550b72b5808abf

Walkthrough : pourquoi vectoriser, install, et comment l'IA fouille dans toutes tes notes d'un coup.

> **Cette leçon est optionnelle.** Ton système fonctionne parfaitement sans cette étape. Mais si tu as un gros coffre (100+ notes) et que tu veux que ton IA puisse fouiller dans toutes tes notes d'un coup, c'est un add-on très puissant.

> **Tu veux aller plus loin que QMD ?** Si tu as des PDFs, des EPUBs, des docs Word, ou des données à partager en équipe, la section **"Aller plus loin : le paysage RAG complet"** en bas de cette leçon présente les autres options.

Le problème

Quand l'IA cherche une info dans ton coffre, elle ouvre des fichiers un par un et les lit en entier. Avec un petit coffre, ça marche. Mais quand tu as des centaines de notes :

- Il ne peut lire que **10 à 15 notes** à la fois
- Il gaspille des tokens en lisant des notes entières pour trouver 5% d'info utile
- Il cherche par mots-clés exacts : s'il ne connaît pas le mot, il ne trouve rien

La solution : la recherche sémantique

QMD transforme toutes tes notes en **coordonnées mathématiques** (on appelle ça des "vecteurs"). Résultat : ton IA peut chercher par **sens**, pas juste par mots-clés, dans toutes tes notes en moins d'une seconde.

Sans QMD
Recherche : "gérer mon stress"
Techniques de relaxation pour la prise de parole
✕ Aucun résultat (mots différents)
Avec QMD
Recherche : "gérer mon stress"
Techniques de relaxation pour la prise de parole
✓ Trouvé ! (même sens)

Exemple : tu cherches "comment gérer mon stress avant une présentation". Sans QMD, ton IA ne trouve rien si aucune note ne contient ces mots. Avec QMD, elle trouve ta note "Techniques de relaxation pour la prise de parole" parce qu'elle comprend que c'est la même idée.

Avant vs après

Sans QMD	Avec QMD
L'IA lit 10-15 notes entières	L'IA interroge toutes tes notes
Recherche par mots-clés uniquement	Recherche par sens + mots-clés
Gaspillage de tokens	~95% de tokens économisés
Lent sur gros coffres	Résultats en moins d'une seconde

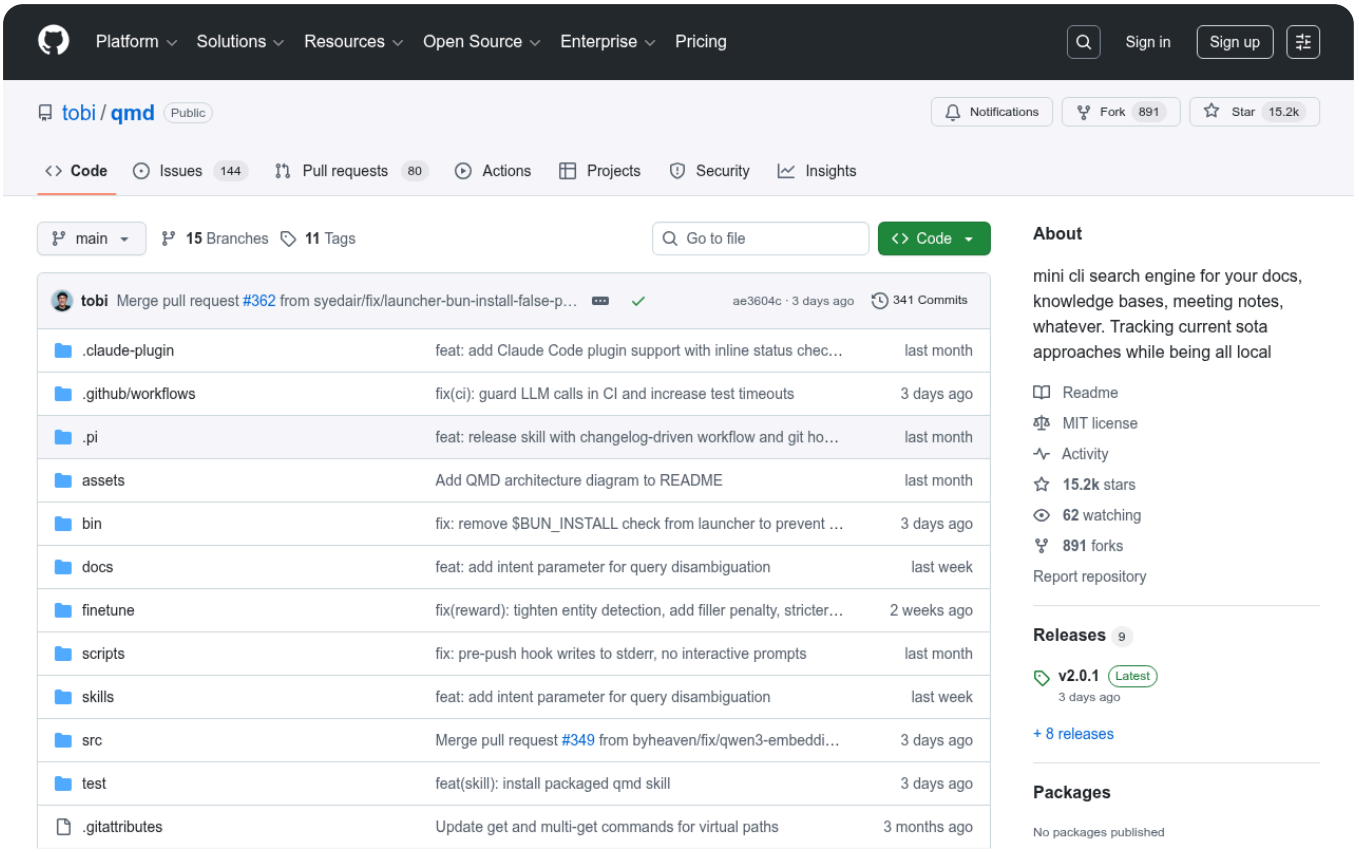
Prérequis

QMD fait tourner de petits modèles d'IA localement (~2 Go). Il faut une machine correcte :

- **RAM :** 8 Go minimum (16 Go recommandé)
- **Processeur :** Mac M1+ ou Intel/AMD récent
- **OS :** macOS ou Linux (Windows via WSL2)

Si ta machine est trop limitée : pas de panique. Tu pourras toujours l'installer plus tard ou sur ton serveur (semaine 3).

Installation



Ouvre OpenCode dans ton vault et donne-lui ce prompt :

```
J'aimerais que tu m'installes QMD (https://github.com/tobi/qmd)
pour
vectoriser mon coffre Obsidian et pouvoir faire de la recherche
sémantique.

Installe QMD, ajoute mon vault comme collection, lance
```

```
l'indexation,
et configure le MCP pour que tu puisses l'utiliser directement.

Mon vault Obsidian est dans : [CHEMIN DE TON VAULT]
```

Remplace `[CHEMIN DE TON VAULT]` par le chemin réel de ton coffre (par exemple `/Users/tonnom/Documents/MonVault`).

OpenCode va :

1. Installer QMD
2. Ajouter ton vault comme collection
3. Lancer l'indexation (quelques minutes la première fois)
4. Configurer le MCP
5. Tester avec une recherche

Utiliser

Une fois installé, tu n'as rien de spécial à faire. Quand tu poses une question, ton IA utilise QMD automatiquement pour chercher dans tes notes.

```
Retrouve ce que j'ai écrit sur la gestion de projet
```

```
Quel est l'état de mon index QMD ?
```

Maintenir l'index à jour

Quand tu ajoutes ou modifies des notes, il faut mettre à jour l'index. Tu peux soit le faire toi-même (`qmd embed` dans le terminal), soit demander à OpenCode :

```
Mets à jour l'index QMD de mon vault
```

QMD ne re-indexe que les fichiers modifiés : c'est rapide.

Aller plus loin : le paysage RAG complet

QMD couvre 90% des besoins pour un vault de notes Markdown. Mais il a des limites : il ne gère pas les PDFs, les EPUBs, les fichiers Word, et ce n'est pas conçu pour être partagé en équipe.

Voici une carte du paysage pour que tu saches quoi utiliser selon ta situation.

Quand QMD ne suffit plus

Situation
Problème avec QMD
Solution recommandée
Gros coffre de notes Markdown
Aucun : c'est fait pour ça
QMD ✓
PDFs, EPUBs, Word, thèses
Ne lit que le Markdown
Docling → QMD ou Qdrant
Données partagées en équipe
Index local seulement
Qdrant sur serveur
Relations entre concepts (graphe)
Recherche vectorielle simple
EdgeQuake (Graph-RAG)
Tout en un (recherche + chat + reco)
Pas d'interface utilisateur
Trieve
PDFs uniquement, usage desktop
Pas d'interface graphique
DocuMind

Les briques : comprendre le pipeline RAG

Un système RAG (Retrieval-Augmented Generation) se compose de plusieurs briques. QMD les fait toutes en une, pratique mais limité. Si tu veux aller plus loin, tu peux assembler les briques toi-même.

Étape 1 : Parsing
Convertir les documents en texte
Docling (IBM) : PDF, EPUB, Word, PowerPoint, images

Étape 2 : Chunking
Découper en morceaux intelligents
Chunkr : parsing + chunking en un service cloud
Étape 3 : Vectorisation
Transformer en vecteurs
QMD (local), Qdrant (serveur), pg_vectorize (PostgreSQL)
Étape 4 : Retrieval
Chercher et injecter dans le contexte de l'IA
MCP QMD, MCP Qdrant, API Trieve

QMD fait les étapes 3 et 4 pour du Markdown. Pour du PDF ou de l'EPUB, il faut d'abord passer par l'étape 1 (Docling) avant.

Les options en détail

QMD : Le point d'entrée (ce qu'on vient d'installer)

Ce que c'est : moteur de recherche sémantique local, conçu pour les notes Markdown. **Idéal pour :** ton vault Obsidian, uniquement des fichiers `.md`. **Limite principale :** ne gère pas les PDFs, EPUBs, fichiers Word. **Souveraineté :** 100% local, rien ne quitte ta machine.

Docling : Le convertisseur universel de documents

Ce que c'est : un SDK open-source créé par IBM (maintenant sous LF AI) qui convertit n'importe quel format de document en texte structuré exploitable par un RAG.

Formats supportés : PDF, EPUB, Word (.docx), PowerPoint (.pptx), images avec OCR, HTML, Markdown.

Le cas d'usage type : tu as des thèses en PDF, des rapports Word, des livres en EPUB. Tu les passes dans Docling, il ressort du Markdown propre. Tu ingères ce Markdown dans QMD (ou Qdrant).

```
# Installer Docling
pip install docling

# Convertir un PDF en Markdown
docling ma-these.pdf --output ma-these.md
```

```
# Convertir un dossier entier
docling /dossier/avec/pdfs/ --output /dossier/markdown/
```

Ensuite, tu ajoutes le dossier Markdown à ton index QMD normalement.

> **Astuce :** Si tu as des EPUBs à vectoriser, la solution la plus simple reste Docling → Markdown → QMD. Pas besoin de solution RAG plus complexe.

Qdrant : La base vectorielle sur serveur

Ce que c'est : une base de données vectorielle haute performance, open-source, déployable sur ton propre serveur.

Différence avec QMD : QMD est un index local sur ta machine. Qdrant est une base de données à part entière : plusieurs personnes peuvent l'interroger en même temps, depuis n'importe quelle machine.

Idéal pour :

- Vectoriser des documents d'équipe accessibles à tous
- Des volumes importants (milliers de documents)
- Intégrer la recherche sémantique dans une app ou un site

Setup typique : Qdrant tourne sur ton serveur (via Docker), ton interface IA l'interroge via son MCP.

```
# Lancer Qdrant avec Docker
docker run -p 6333:6333 qdrant/qdrant
```

> **Chez Perspectives :** Alexandre (dans la discussion) utilise Qdrant sur son serveur pour indexer ses cours en ligne et des livres de biologie. C'est le bon choix dès que tu veux partager l'index avec d'autres personnes.

---

pg_vectorize : Si tu as déjà PostgreSQL

Ce que c'est : une extension PostgreSQL qui ajoute la vectorisation directement dans ta base de données.

Idéal pour : tu as déjà une app avec une base PostgreSQL (Supabase, NocoDB, etc.) et tu veux ajouter de la recherche sémantique sans déployer un nouveau service.

Avantage : tout reste dans ta base de données existante. Pas de service supplémentaire à gérer.

EdgeQuake : Graph-RAG pour les cas complexes

Ce que c'est : un système RAG de type "graphe de connaissances", écrit en Rust pour la performance.

Différence avec un RAG classique : un RAG vectoriel cherche "les chunks les plus proches sémantiquement". Un Graph-RAG comprend aussi les **relations entre les concepts** : il peut répondre à des questions comme "quels sont les liens entre le concept A et le concept B dans ma base de connaissance ?"

Idéal pour : des corpus très riches où les connexions entre idées comptent autant que les idées elles-mêmes (bases de connaissance expert, documentation technique dense).

À retenir : c'est la solution la plus puissante, mais aussi la plus complexe à mettre en place. À explorer quand QMD ou Qdrant ne suffisent plus.

> **Stack recommandée par Emmanuel Salomon pour des documents experts :** Docling (parsing) → EdgeQuake (Graph-RAG) → PostgreSQL.

Trieve : La plateforme RAG clé en main

Ce que c'est : une plateforme full-stack qui combine recherche vectorielle, recommandations et interface de chat. Open-source et auto-hébergeable.

Idéal pour : tu veux construire un moteur de recherche ou un chatbot basé sur tes documents, avec une interface utilisateur incluse, sans assembler toi-même les briques.

Différence avec Qdrant : Qdrant est une brique (la base vectorielle). Trieve est un produit complet avec recherche, suggestions, analytics, et API prête à l'emploi.

DocuMind : Pour les PDFs en mode desktop

Ce que c'est : une application desktop (Mac, Windows, Linux) qui permet de faire du RAG sur des PDFs directement depuis ton ordinateur, sans serveur.

Idéal pour : tu travailles principalement avec des PDFs et tu veux une interface graphique simple, sans toucher à un terminal.

Limite : application desktop uniquement, pas de partage en équipe.

Quel chemin prendre ?

Uniquement des notes Markdown
→ **QMD** : c'est déjà fait dans cette leçon.
PDFs, EPUBs, Word à vectoriser
→ **Docling** pour convertir en Markdown, puis **QMD** pour indexer. Pipeline en 2 commandes.
Partager l'index avec une équipe
→ **Qdrant** sur ton serveur. Partagé, persistant, accessible depuis n'importe quelle machine.
Application ou site avec recherche sémantique
→ **Trieve** (plateforme complète) ou **Qdrant** + ton propre front.
Expertise métier dense avec relations entre concepts
→ **Docling → EdgeQuake → PostgreSQL**. La stack la plus puissante, la plus complexe.

- [ ] Ma machine a la config minimum (8 Go RAM, Mac M1+ ou équivalent)
- [ ] J'ai demandé à OpenCode d'installer QMD
- [ ] L'indexation est terminée
- [ ] J'ai testé une recherche dans OpenCode via QMD
- [ ] (Optionnel) J'ai converti mes PDFs/EPUBs avec Docling et indexé le résultat dans QMD
- [ ] (Optionnel) J'ai déployé Qdrant sur mon serveur pour un index partagé en équipe

Vidéos

LOOM

Vectoriser son coffre avec QMD

https://www.loom.com/share/dab0cef6d37c4bc98a550b72b5808abf

Assets

- Repo GitHub de QMD : moteur de recherche local pour tes documents

---

Extraction verbatim depuis Prisme One · https://prisme.one/plateforme/ia-juillet-2026/semaine-1-vectoriser-coffre