2 mémoires pour des IAs qui n'oublient plus

Live Memory + Graph Memory — Architecture mémoire pour agents IA

Christophe Lesur — Cloud Temple

Open Source | Apache 2.0 | MCP Protocol

Un LLM, c'est quoi au fond ?

Pour ceux qui viennent du réseau, c'est simple

$$f(x) = y$$

prompt → modèle (200 milliards de paramètres) → réponse

C'est une **fonction mathématique**. Rien de plus.

- 📥 Une **entrée** — votre prompt + contexte
- ⚙️ Un **calcul** — multiplication de matrices
- 📤 Une **sortie** — la réponse générée

Comme un serveur web qui traite une requête HTTP :

c'est 100% stateless.

Aucune mémoire entre deux appels.
Zéro. Nada.

💡 Un concept familier pour les gens du réseau !

L'amnésie en action

Ce qui se passe à chaque nouvelle session

🟢 Session 1 — 14h00

L'agent a analysé votre infrastructure, compris les dépendances réseau, pris des décisions d'architecture, documenté les flux, écrit du code...

→ **Productif, contextualisé, pertinent**

🔴 Session 2 — Le lendemain

"Bonjour ! Comment puis-je vous aider ?"

→ **Tout est perdu. Il repart de zéro.**

*C'est comme si votre **meilleur ingénieur réseau** avait un **reset total** chaque matin.*
*Et si vous avez 3 agents IA sur le même projet ? **Chacun dans sa bulle.** Aucune coordination.*

Comment fonctionne *notre* mémoire ?

Le cerveau humain a résolu ce problème depuis des millions d'années

🔴 Mémoire de travail

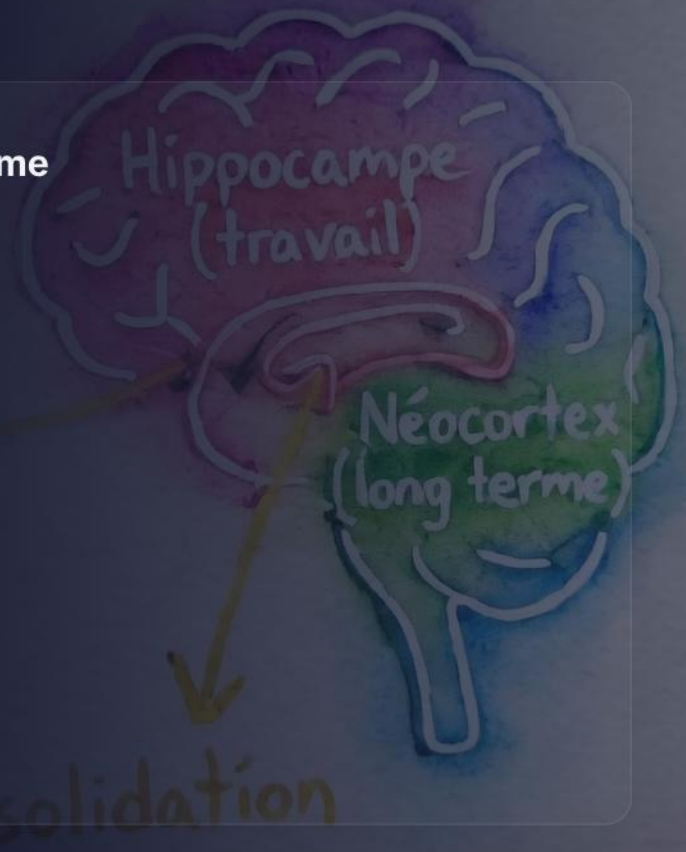
Hippocampe

- ⚡ Rapide, volatile
- 📝 Notes mentales en cours
- 🔄 Capacité limitée (~7 éléments)
- ⏱️ Durée : minutes à heures

🟢 Mémoire long terme

Néocortex

- 🏗️ Structurée, durable
- 🔗 Réseau de concepts reliés
- ♾️ Capacité quasi-illimitée
- ⏱️ Durée : jours à années



*Le sommeil **consolide** : il trie les souvenirs de la journée et les intègre dans la mémoire long terme.*
C'est exactement ce que fait notre architecture.

Notre réponse : 2 mémoires

Inspirées directement de l'architecture cognitive humaine

	🔴 LIVE MEMORY	🟢 GRAPH MEMORY
Rôle	Mémoire de travail partagée	Base de connaissances permanente
Durée	Session / projet	Permanent
Contenu	Notes live + Bank Markdown	Entités + Relations + Embeddings
Stack	S3 seul (pas de BDD !)	Neo4j + Qdrant + S3
Analogie	Tableau blanc + cahier de projet	Bibliothèque avec index

📖 Réf : Tran et al., 2025 — *"Multi-Agent Collaboration Mechanisms: A Survey of LLMs"* (arXiv:2501.06322)

🔴 Live Memory

Le tableau blanc partagé — chaque agent écrit des notes atomiques en temps réel

🤖 Agent Cline

`observation` Build OK, 0 erreurs

`decision` REST plutôt que gRPC

`todo` Tester l'auth OAuth2

🤖 Agent Claude

`observation` Latence API > 200ms

`question` Redis ou Memcached ?

`progress` Module auth terminé

🤖 Agent QA

`issue` Fuite mémoire en charge

`insight` Pattern retry efficace

`observation` 95% couverture tests

Architecture : Append-Only sur S3

📦 **Pas de base de données** — juste du S3

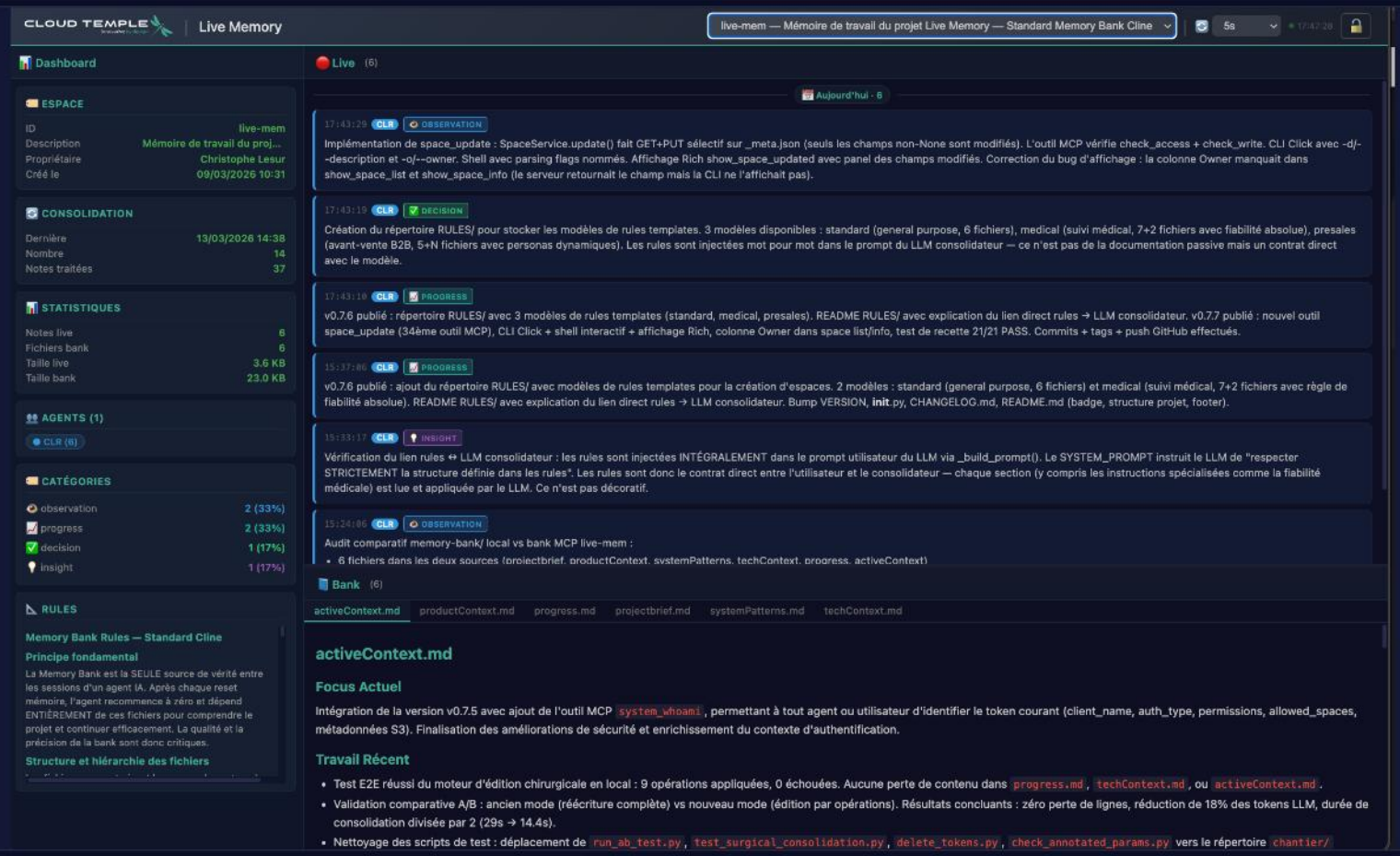
🔒 **Pas de conflit** — 1 note = 1 fichier unique horodaté

🐜 **Stigmergie** — communication indirecte via l'environnement

*Comme les fourmis qui communiquent par les **phéromones**, pas par la parole. Chaque agent laisse des traces que les autres lisent.*

L'interface Live Memory en action

Dashboard + Timeline live + Memory Bank — tout le contexte d'un projet en un coup d'œil



📊 Dashboard (agents, stats, rules) · 🔴 Timeline live (notes groupées par date) · 📘 Bank Viewer (Markdown consolidé) · 🔄 Auto-refresh intelligent

🧠 La consolidation LLM

Du chaos à la structure — le "sommeil" de notre architecture

📝 **Notes brutes**
47 notes · 3 agents

→

🧠 **LLM**
Rules + Notes + Bank

→

📘 **Bank mise à jour**
Édition chirurgicale

→

🗑️ **Notes purgées**
Intégrées → supprimées

❌ Approche naïve

Le LLM **réécrit tout** le fichier à chaque consolidation.

Syndrome **"photocopie de photocopie"** :

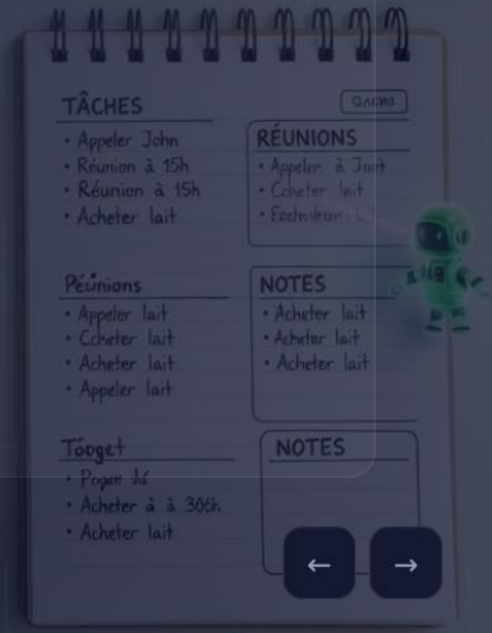
Dégradation progressive de l'information à chaque passe.

✅ Notre approche : édition chirurgicale

Le LLM produit des **opérations d'édition par section Markdown**.

Ce qui n'est pas touché reste **identique bit-à-bit**.

Zéro perte d'information. ~50% tokens en moins.



Les **Rules** sont le contrat immuable : elles définissent la structure de la bank. Le LLM s'y conforme systématiquement.

📐 Les Rules en action

Le contrat immuable qui structure la mémoire — défini une fois, respecté toujours

Exemple : Rules "Standard Dev"

```
## Structure de la Memory Bank

projectbrief.md (fondation)
├── productContext.md  pourquoi le projet existe
├── systemPatterns.md  architecture & patterns
└── techContext.md     stack technique & setup
    └── activeContext.md focus actuel
        └── progress.md     journal d'avancement

## Mapping notes → fichiers
observation → activeContext.md
decision    → systemPatterns.md
progress    → progress.md
issue       → progress.md (problèmes connus)
insight     → systemPatterns.md
```

4 templates fournis

🖥️ Standard Dev
6 fichiers — projets logiciels, architecture, code

📚 Écriture de livre
6 fichiers — narratif, chapitres, style, compteur de mots

🏥 Suivi médical
7+2 fichiers — patients, traitements, fiabilité absolue

💼 Avant-vente B2B
5+N fichiers — personas dynamiques, qualification, offre

> *Les rules sont **immuables après création**. Elles garantissent que la mémoire garde toujours la même structure, quel que soit le nombre de consolidations.*

🟢 Graph Memory

La bibliothèque intelligente — un graphe de connaissances, pas une boîte de chunks

Les ontologies : le vocabulaire métier qui guide l'extraction

⚖️ Legal	☁️ Cloud	🔧 Managed Svc	💼 Presales	📋 General
22 entités · 22 relations	27 entités · 19 relations	20 entités · 16 relations	28 entités · 30 relations	24 entités · 22 relations

> *L'ontologie dit au LLM quoi chercher dans les documents. Pas de relations génériques `RELATED_TO` : des types explicites comme `DEFINES`, `APPLIES_TO`, `REFERENCES`.*

⚖️ L'ontologie en action

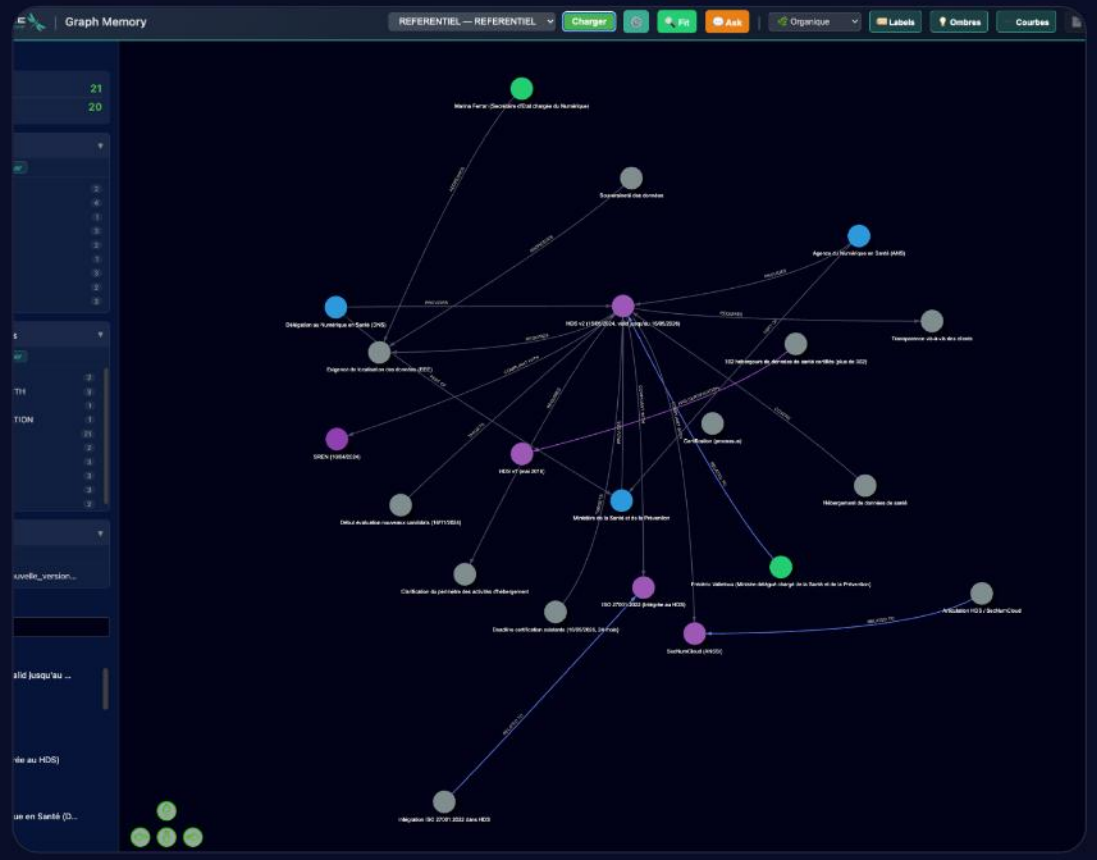
Exemple : ontologie "Legal" pour l'analyse de contrats

```
entity_types:
  - Party         # "Cloud Temple SAS", "TechCorp"
  - Clause        # "Article 12 - Réversibilité"
  - Amount        # "8 500 EUR HT/mois"
  - Duration      # "36 mois (durée initiale)"
  - SLA           # "99.95% disponibilité"
  - Certification # "SecNumCloud", "HDS"
  - Obligation    # "Garantie de disponibilité"
  ... (22 types au total)

relation_types:
  - PARTY_TO      # Client → Contrat
  - HAS_AMOUNT    # Prestation → 50 000 EUR
  - HAS_SLA       # Service → 99.95%
  - GUARANTEES    # Cloud Temple → SecNumCloud
  - OBLIGATES     # Clause → Client
  - GOVERNED_BY   # Traitement → RGPD
  ... (22 types au total)
```

> *Le LLM reçoit l'ontologie + le document → extrait **exactement** les entités et relations définies. Pas de bruit, pas d'hallucination structurelle.*

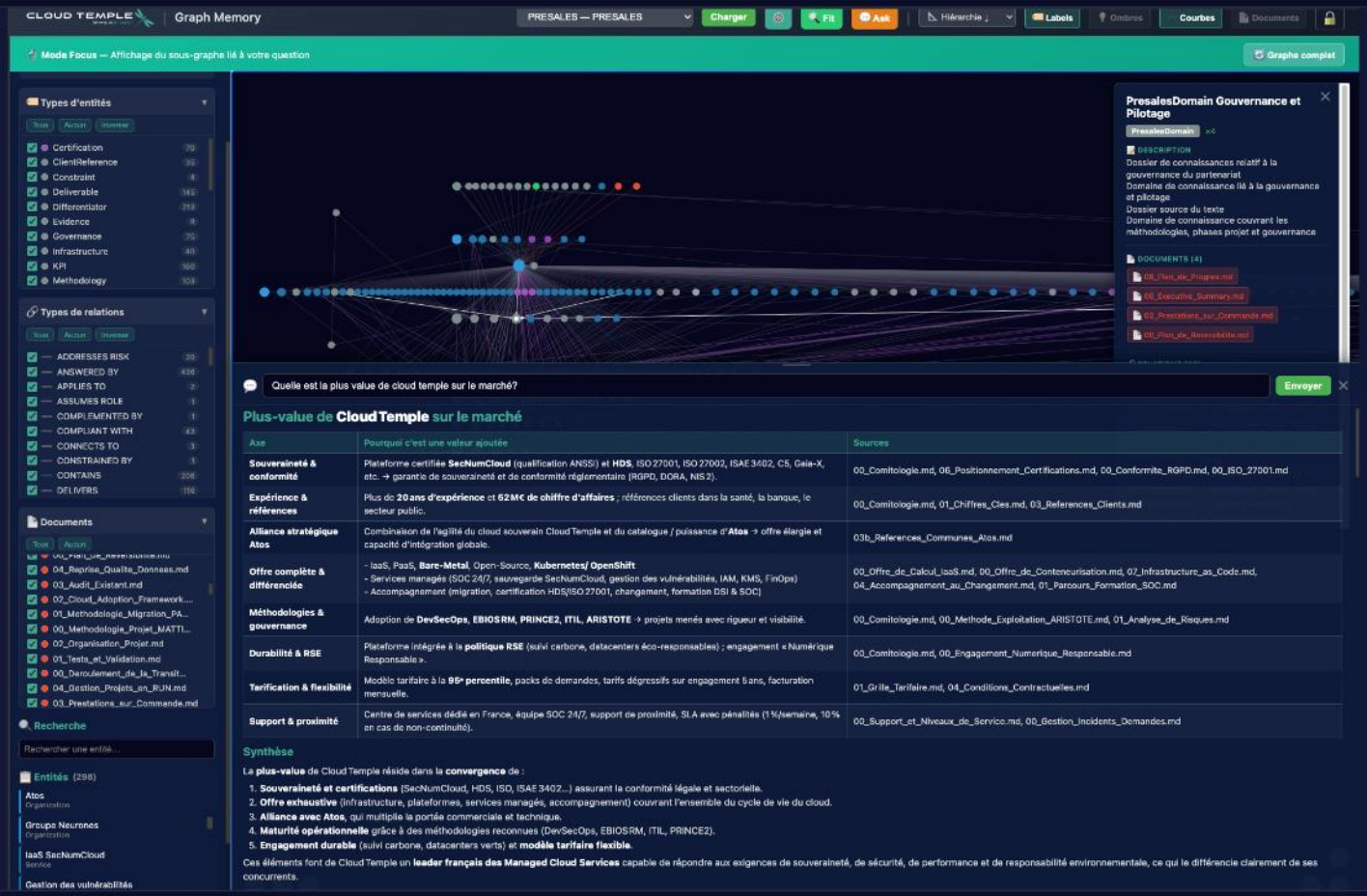
Résultat : le graphe de connaissances



Interface Graph Memory — Graphe interactif avec filtrage par types d'entités et de relations

💬 Interroger le graphe en langage naturel

Le panneau ASK — question → réponse LLM avec citations des documents sources



🗣️ Question en langage naturel | 📊 Réponse Markdown (tableaux, listes, code) | 📄 Citations des documents sources | 🔬 Mode Focus : sous-graphe pertinent

Pourquoi un graphe ?

Graph-First vs RAG vectoriel classique

RAG vectoriel classique

- 📦 Document → chunks → embeddings
- 🔍 Recherche par similitude cosinus
- ❓ Chunks **anonymes**
- 🎯 Approximatif (voisinage sémantique)
- ↔️ Recherche unidirectionnelle

Comme chercher dans Google
avec juste des mots-clés

VS

Graph-First (notre approche)

- 🔗 Document → **entités + relations**
- 🗺️ Navigation dans un graphe structuré
- 📄 Chaque entité liée à son **document source**
- 🎯 Précis (relations explicites, typées)
- 🔀 Navigation **multi-hop**

Comme naviguer Wikipedia :
de lien en lien, avec contexte

Graph-Guided RAG : le meilleur des deux mondes

❶ **Graphe**
Identifie entités + documents

→

❷ **Qdrant**
Chunks DANS ces documents

→

❸ **LLM**
Réponse avec citations

Le graphe **cible**, le vectoriel **précise**. Pas de bruit.

🌉 Le pont entre les deux

De la mémoire de travail à la mémoire permanente

⚡ Écrire vite

Notes live : **~50ms**

Consolidation : **~15 secondes**

→ Zéro friction pour les agents

🏛️ Capitaliser durablement

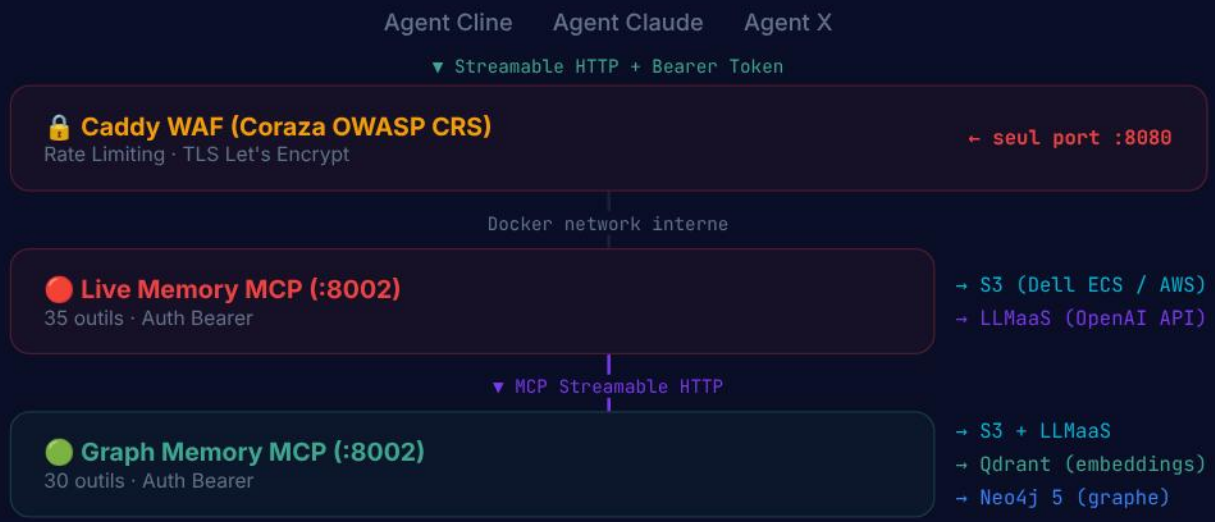
Graph push : **~30s/fichier**

Interrogation en langage naturel

→ Les connaissances survivent aux projets

🏗️ Architecture technique

Vue réseau — ce que vous aimez voir



🔒 WAF OWASP CRS | 🔑 Auth Bearer Token | 📡 MCP Streamable HTTP | 🐳 Docker non-root | 📦 S3 source of truth

MCP (Model Context Protocol) = la norme émergente pour les outils d'IA (Anthropic, 2024).
Comme REST pour les humains, **MCP pour les agents.**

Des IAs qui capitalisent

❌ Avant	✅ Après
Amnésie totale entre sessions	Mémoire de travail partagée
Agents isolés, sans coordination	Collaboration multi-agents
Connaissances perdues	Graphe de connaissances permanent
RAG approximatif	Q&A précis avec sources

Open Source · Apache 2.0 · Prêt pour la production

`github.com/Cloud-Temple/live-memory`
`github.com/Cloud-Temple/graph-memory`

Christophe Lesur — Cloud Temple