



MNEMOLITE

PostgreSQL-Native Cognitive Memory

v1.3.0 • 100% Local • Zero API Keys

🤨 L'ENFER DE L'AMNÉSIE IA (2025)

💬 *"Désolé, je ne me souviens pas de notre conversation d'hier..."*

📁 LE COÛT CACHÉ DE L'AMNÉSIE:

Coût GPT-4 Turbo	\$1-2 / conversation longue <small>(20 échanges)</small>
Frustration Utilisateurs	77% trouvent chatbots frustrants <small>(2024)</small>
Abandon Clients	91% partent sans se plaindre <small>(UX research)</small>
Context Window Growth	30× par an depuis 2023 <small>(Epoch AI)</small>

🔥 LE VRAI PROBLÈME:

- ▶ Context windows limités (même Gemini 10M tokens) → Memory inflation → coûts explosifs
- ▶ Dégradation contextuelle → Info critique perdue → UX catastrophique
- ▶ "Bigger context" ≠ Solution → Besoin d'une VRAIE mémoire persistante

💡 Et si la solution n'était PAS un context window plus grand,
mais une mémoire persistante PostgreSQL ?



LE PROBLÈME DES SOLUTIONS EXISTANTES

PINECONE, WEAVIATE, MILVUS

- ▶ ✗ Cloud-only ou self-hosting complexe
- ▶ ✗ Coûts récurrents (\$25-500+/mois)
- ▶ ✗ Latence réseau (50-200ms)
- ▶ ✗ Dépendance OpenAI (\$\$\$)

SOLUTIONS DIY (REDIS + SQLITE + ...)

- ▶ ✗ Stack technique fragmentée
- ▶ ✗ Maintenance cauchemardesque
- ▶ ✗ Scalabilité limitée
- ▶ ✗ Pas de graph natif



Besoin: Solution simple, locale, performante, unifiée



ET SI POSTGRESQL POUVAIT TOUT FAIRE ?

✓ **Vector DB** → pgvector

✓ **Time-series** → pg_partman

✓ **Queue** → pgmq

✓ **Graph** → CTE récursifs

✓ **Embeddings** → Sentence-Transformers

✓ **Local** → 100% privacy



MNEMOLITE = POSTGRESQL + PGVECTOR + LOCAL AI

Zéro dépendance externe • Zéro coût cloud • 100% privacy



POURQUOI MNEMOLITE GAGNE MAINTENANT EN 2025

⚡ PGVECTOR 0.8.0 - RÉVOLUTION

Sortie: 30 octobre 2024

- ▶ **9.4× Plus Rapide** - 123ms → 13ms (scans itératifs)
- ▶ **Surpasse les DB Spécialisées** - Performance compétitive vs Weaviate/Qdrant
- ▶ **50% Stockage** - type halfvec (2-byte floats)
- ▶ **Pas de Surfiltre** - Modes relaxed/strict



EXPLOSION ADOPTION ENTERPRISE

- ▶ **Supabase** - valorisation \$2B (avril 2025)
- ▶ **Neon** - acquis par Databricks (\$1B, mai 2025)
- ▶ **Databricks + Snowflake** - acquisitions startups PG
- ▶ **60-80% Réduction Coûts** vs DB dédiées

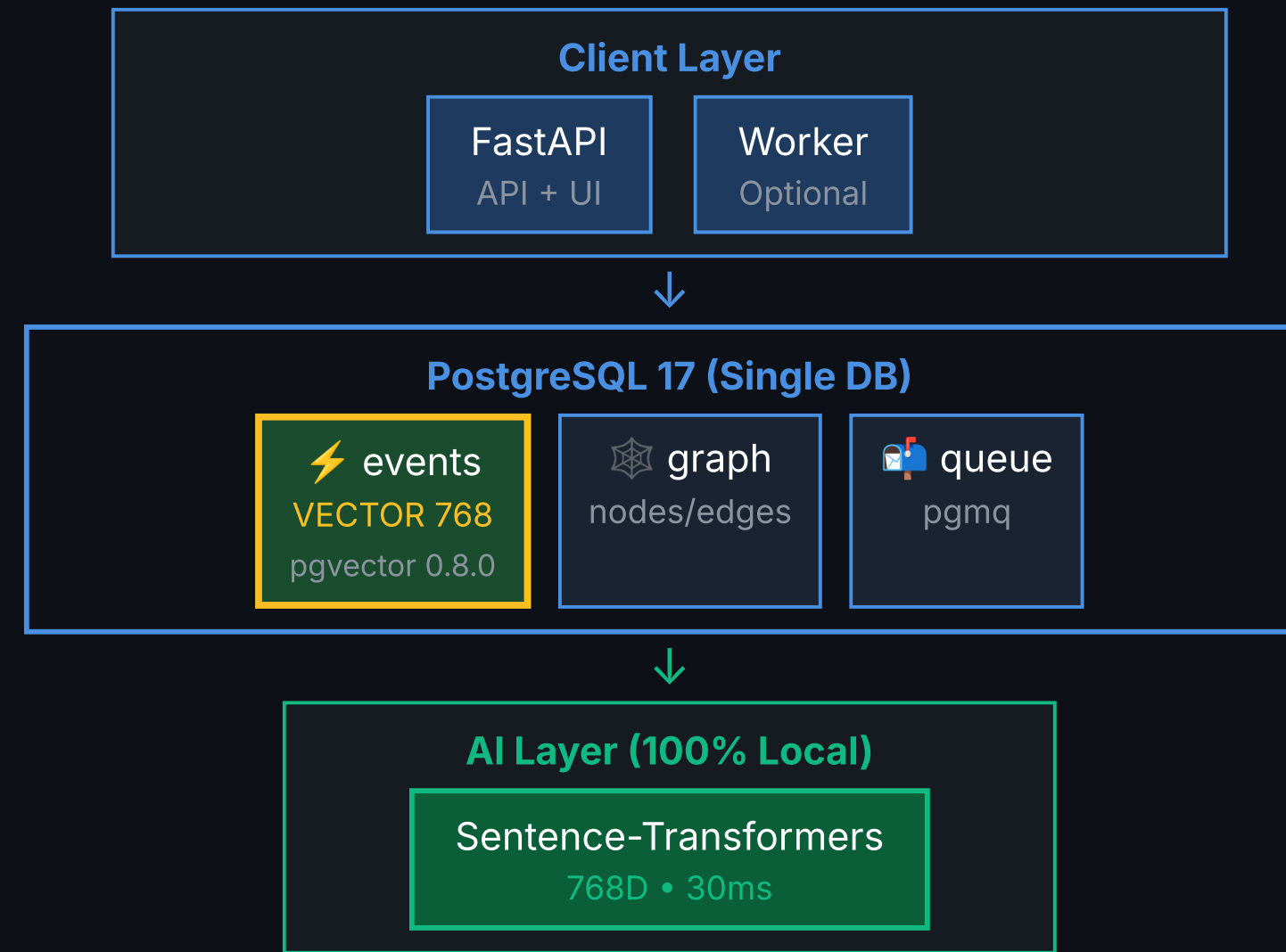


PostgreSQL est MAINTENANT le standard pour la recherche vectorielle

MnemoLite surfe cette vague **PARFAITEMENT**



ARCHITECTURE MNEMOLITE V1.3.0



⚡ **pgvector 0.8.0**
9.4× plus rapide

🎯 **Single DB**
Vector + Graph + Queue

🏠 **100% Local**
Zero external APIs

🎯 **2 containers (3 avec worker) • 1 database • 0 external APIs**



TECH STACK 2025 - 100% LOCAL

BACKEND

- ▶ FastAPI 0.111+
- ▶ SQLAlchemy 2.0+
- ▶ asyncpg
- ▶ Pydantic v2

DATABASE

- ▶ PostgreSQL 17
- ▶ pgvector 0.8.0
- ▶ pg_partman
- ▶ pgmq

AI / EMBEDDINGS

- ▶ Sentence-Transformers
- ▶ nomic-embed-text-v1.5
- ▶ 768D, Apache 2.0
- ▶ ~30ms, 50-100 emb/s



EMBEDDINGS 100% LOCAUX VS OPENAI

Coût	✗ ~\$300/mois	✓ \$0
Latency	✗ 150ms	✓ ~30ms
Privacy	✗ Cloud	✓ 100% local



ROI: \$3,600/an économisés



ÉVOLUTIONS 2025

Nomic v2 (Q2 2025)
→ Upgrade path ready

Alternatives:

- BGE-M3 (multi-lingual)
 - E5 (Microsoft)
 - MiniLM (compact)



Tout open-source • Tout local • Zéro vendor lock-in



UI V4.0 - DESIGN SCADA INDUSTRIEL

PRINCIPES SCADA

- ▶ Zero border-radius
- ▶ Ultra dark palette
- ▶ Border-left accents
- ▶ Compact spacing
- ▶ Fast transitions (80ms)
- ▶ High info density

ARCHITECTURE MODULAIRE

- ▶ 17 modules CSS
- ▶ 6 modules JavaScript
- ▶ 0 ligne inline JS
- ▶ HTMX 2.0
- ▶ Full ARIA



Dashboard



Search



Graph



Monitoring



DÉMARRAGE RAPIDE (< 2 MINUTES)

```
$ git clone https://github.com/giak/MnemoLite.git
$ cd MnemoLite
$ cp .env.example .env
$ make up
```

✓ Services démarrés:

- mnemo-postgres (PostgreSQL 17)
- mnemo-api (FastAPI @ :8001)

🌐 <http://localhost:8001/ui/>

CONFIGURATION (.ENV)

- ▶ POSTGRES_PASSWORD: votre_mot_de_passe
- ▶ EMBEDDING_MODEL: nomic-ai/nomic-embed-text-v1.5
- ▶ ENVIRONMENT: development|production





🔑 **Aucune clé API requise. Zéro configuration cloud.**



DEMO: DASHBOARD & RECHERCHE HYBRIDE

 Dashboard: `localhost:8001/ui/`

DASHBOARD LIVE

1.  Période "24h" → HTMX update
2.  Filtrer projet → instantané
3.  Event card → modal JSON
4.  Network: zéro reload

 **HTMX 2.0: Updates partiels HTML**

 Search: `localhost:8001/ui/search`

RECHERCHE SÉMANTIQUE

Vector: `"integration tests failing"`

Project: Expanse

Category: bug

Threshold: 0.8 (**strict**)

→ 8 events en ~9ms ⚡

Threshold: 0.8 (strict) | 1.0 (balanced) | 1.2 (relaxed)

 **Deux démos clés: UI moderne (HTMX) + Recherche puissante (hybrid search)**



DEMO: GRAPH DE CONNAISSANCES



LIVE: <http://localhost:8001/ui/graph>

INTERACTIONS LIVE:

1. 🗺️ Changer layout: cose → circle → grid
2. 🔍 Filtrer par type: désactiver "concepts"
3. 🖱️ Cliquer node → sidebar détails
4. 💡 Hover → tooltip infos



5 layouts



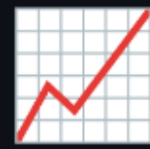
Color-coding



Zoom controls



Minimap



DEMO: MONITORING DASHBOARD



LIVE: <http://localhost:8001/ui/monitoring>



TIMELINE (30 DAYS)

ECharts line chart - Events/day



DISTRIBUTION

By Type • By Project (Pie charts)



CRITICAL EVENTS

Last 24h alerts (ERROR, CRITICAL)



AUTO-REFRESH

30s (pause when hidden - Visibility API)

API REST - OPENAPI 3.1

 Swagger UI: <http://localhost:8001/docs>

EVENTS V1

- ▶ **POST** /v1/events
- ▶ **GET** /v1/events/{id}
- ▶ **GET** /v1/events/

SEARCH V1

- ▶ **GET** /v1/search/
vector_query, filter_metadata,
ts_start, ts_end, distance_threshold

HEALTH

- ▶ **GET** /health
- ▶ **GET** /readiness
- ▶ **GET** /metrics

⚡ PERFORMANCE EXCEPTIONNELLE (PGVECTOR 0.8.0)

MNEMOLITE (PG17 + PGVECTOR 0.8.0)

Hybrid Search	11ms P50: 7ms
Vector Search	12ms P50: 8ms
Metadata + Time	3ms P50: 2ms

🎯 **<15ms P95: ATTEINT** ✅

COMPARAISON 2025 (BENCHMARKS RÉELS)

pgvector 0.8.0	9.4× plus rapide (vs 0.7.4)
vs Weaviate	243% plus rapide (Timescale)
vs Pinecone	pgvectorscale surpasse

📈 Throughput: **3-5× en production**

💡 SECRET SAUCE PGVECTOR 0.8.0:

- ▶ **Scans itératifs** → Pas de surfiltre
- ▶ **halfvec type** → 50% stockage économisé
- ▶ **HNSW + JSONB** → Combo PostgreSQL natif
- ▶ **Optimisation query plan** → Estimation coûts

📊 Environnement: Local machine, ~50k events, non-partitioned
Performance scalée avec partitioning @500k+

✓ TESTS & QUALITÉ

178

Tests Totaux

100%

Pass Rate






~13s

Duration

~87%

Coverage

TYPES DE TESTS:

- ▶  Repository: Event, Memory
- ▶  Services: Embedding, Search
- ▶  Routes: Events, Search, Health, UI
- ▶  Database: Schema, indexes
- ▶  Integration: Semantic, hybrid

 **CI/CD Ready: pytest-asyncio, Docker-based, Makefile**



DÉPLOIEMENT PRODUCTION

```
# docker-compose.yml (Production)
services:
  postgres:
    image: pgvector/pgvector:pg17
    volumes: ./data:/var/lib/postgresql
    restart: always

  api:
    build: ./api
    environment:
      - ENVIRONMENT=production
      - LOG_LEVEL=INFO
```



SÉCURITÉ

- ▶ PostgreSQL auth
- ▶ API CORS config
- ▶ Env variables (.env)



MONITORING

- ▶ /health endpoint
- ▶ /metrics (Prometheus)
- ▶ Structured logging



SCALING

- ▶ Current: <500k events
- ▶ @500k: pg_partman
- ▶ Future: INT8 quant



MNEMOLITE VS CONCURRENTS 2025

Critère	Pinecone	Weaviate	Qdrant	ChromaDB	MnemoLite
Coût	>\$500/mo @scale	\$25-153/mo cloud	\$102/mo AWS	FREE	\$0
P95 Latency	50-200ms	30-100ms	20-50ms	Varies	<15ms
Setup	Easy	Medium	Medium	pip install	make up
Local Deploy	✗	✓	✓	✓	✓
Scaling	Cloud Auto	Horizontal	Horizontal	Single-node	<500k optimal
Graph Support	✗	Partial	✗	✗	✓ Native
Lock-in	High	Medium	Medium	Zero	Zero

vs ChromaDB: Production-ready (vs single-node) + Graph

vs Qdrant: Simpler (PostgreSQL) + \$0 vs \$102/mo

vs Pinecone: 100% local + \$0 vs >\$500/mo

Sweet Spot: 10k-500k events, local, privacy



MNEMOLITE: MÉMOIRE COGNITIVE SIMPLIFIÉE



Agents IA

Mémoire persistante



Testing

Agent frameworks



RAG

Documentation, notes



Enterprise

Compliance, on-prem



Production-ready (v1.3.0)



Performance (<15ms P95)



100% local, \$0 cloud



Setup 2 min (Docker)



Open-source (MIT)



178 tests ✓



DÉMARRER MAINTENANT

```
$ git clone https://github.com/giak/MnemoLite.git
$ cd MnemoLite && make up
$ open http://localhost:8001/ui/
```



Docs: README.md • GUIDE_DEMARRAGE.md |  **GitHub:** github.com/giak/MnemoLite |  **License:** MIT

★ **Sweet Spot: 10k-500k events • local • privacy-critical**

? QUESTIONS ?