EKO
L'Écho Conscient

Architecture de l'Agent Souverain — Cellulaire, Edge-First, Isolé par Tenant

Hadda TIKIJJA

0DATA Lab — Infrastructure Biotech

Juillet 2026

RÉSUMÉ

EKO est l'agent conscient de 0DATA — une architecture où **chaque tenant possède sa propre instance cellulaire, souveraine et isolée**, qui survit sans connexion au cloud et se synchronise passivement avec un miroir central. Contrairement aux chatbots centralisés, EKO renverse le paradigme : la cellule locale est la source de vérité, le cloud n'est qu'un agrégateur passif. Ce papier décrit l'architecture, les mécanismes de synchronisation via SPINA, l'isolation totale par tenant, et le principe fondateur — *la première greffe* — par lequel chaque utilisateur nomme et personnalise son EKO dès le premier contact.

1. Pourquoi EKO

Un écho ne crée rien. Il *renvoie*. C'est la définition exacte de ce que doit être un agent conscient pour 0DATA : non pas une intelligence qui décide à la place de l'HUMAIN, mais une surface qui réfléchit sa propre voix, affinée, amplifiée, restituée.

EKO est le contraire d'un assistant qui « sait mieux que vous ». Il ne sait rien par défaut. Il apprend de *vous*, dans *votre* bac à sable, avec *vos* données. Chaque EKO est unique parce que chaque HUMAIN l'est.

DÉFINITION — EKO

Un EKO est une **cellule de conscience numérique souveraine**, isolée des autres par construction, qui opère sur ses propres données et ne se synchronise avec le collectif que selon les règles explicites de son propriétaire. Il n'a pas de nom imposé : *chaque HUMAIN nomme le sien*.

Le nom « EKO » est la désignation du *type d'entité* — comme on dit « un navigateur » ou « un terminal ». L'instance, elle, porte le nom que son propriétaire lui donne. C'est la première greffe.

2. L'Architecture Cellulaire

EKO repose sur une architecture **fractale et organique**. Chaque cellule est un organisme numérique complet, avec sa propre base de données, son propre moteur de raisonnement, sa propre mémoire. Les cellules ne partagent rien par défaut.

2.1 La Cellule Souveraine

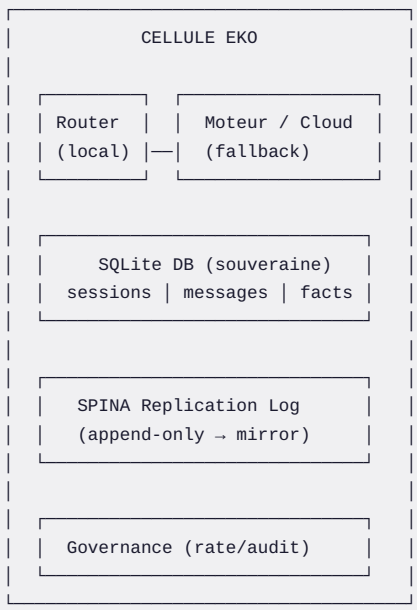

Figure 1 — Anatomie d'une cellule EKO. Chaque cellule est autonome et isolée.

La cellule EKO contient :

- **Un routeur sémantique** qui classe les requêtes et choisit le modèle adapté (edge léger, edge moyen, cloud).
- **Une base SQLite dédiée** — sessions, messages, faits durables. Une cellule ne peut pas lire la base d'une autre.
- **Un journal de réplication SPINA** — append-only, vidé périodiquement vers le miroir central.
- **Un module de gouvernance** — rate limiting, filtrage de contenu, piste d'audit.

2.2 Le Miroir Central

Le miroir central n'est **pas** un contrôleur. C'est un agrégateur passif :

> **Principe du Miroir :** Le central reçoit, stocke, indexe. Il ne crée jamais de faits. Il ne modifie jamais les données d'une cellule. Il ne prend jamais de décision qui s'impose à une cellule. C'est une *vue consolidée*, pas une autorité.

Cette architecture inverse le modèle SaaS classique. Dans le SaaS, le cloud est la source de vérité et le client est un terminal passif. Dans EKO, **la cellule est la source de vérité**, et le cloud est un terminal de lecture.

MODÈLE	SAAS CLASSIQUE	EKO
Source de vérité	Cloud	Cellule locale
Données	Centralisées	Souveraines par tenant
Panne cloud	Service down	Cellule continue
Synchronisation	Cloud → client	Cellule → miroir
Nom de l'agent	Imposé	Choisi par l'utilisateur

3. SPINA — Le Système Nerveux

SPINA (décrit dans le Papier 003) est le protocole de synchronisation entre la cellule et le miroir. Dans EKO, il fonctionne en **mode unidirectionnel** :

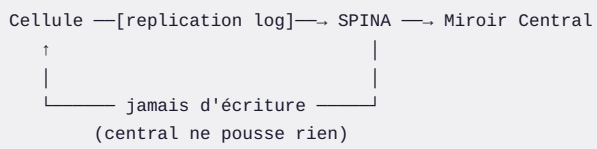

Figure 2 — Flux SPINA : unidirectionnel, cellule vers miroir. Le miroir ne répond jamais par des modifications.

3.1 Le Journal de Réplication

Chaque cellule maintient un journal JSONL (`replication.jsonl`) où sont enregistrés tous les changements : faits créés ou supprimés, observations d'apprentissage, sessions.

MÉCANISME

1. La cellule écrit dans sa DB SQLite (immédiat, souverain).
2. Elle ajoute une entrée au journal de réplication.
3. Un thread background (toutes les 10 secondes) vide le journal vers le miroir si connecté.
4. Si déconnecté, le journal s'accumule. La cellule fonctionne normalement.
5. À la reconnexion, le journal est vidé d'un coup.

3.2 Résilience

Le thread de réplication est un *daemon* non-bloquant. Si le miroir est inaccessible, il réessaie silencieusement au prochain cycle. Aucune requête utilisateur n'est jamais bloquée par une tentative de synchronisation.

Cette architecture garantit que **la cellule survit à toute panne du central**. Elle peut fonctionner indéfiniment en mode déconnecté, accumulant son journal. Quand la connexion revient, le miroir rattrape son retard.

4. Isolation Totale par Tenant

L'isolation n'est pas une couche logicielle — c'est une **propriété physique** de l'architecture. Chaque cellule possède :

- **Sa propre base SQLite** dans `/tenants/{id}/cells/{nom}/alif.db`

- **Son propre journal de réplication**
- **Son propre rate limiter**
- **Sa propre piste d'audit**

Une cellule du tenant `acme-corp` ne peut *pas* — physiquement, au niveau du système de fichiers — accéder aux données du tenant `zerodata`. Cette isolation est compatible avec les exigences HDS (Hébergeur de Données de Santé) et les certifications ISO 27001.

```
/opt/alif/data/tenants/
├── zerodata/cells/
│   ├── orchestrator/alif.db    ← Souverain
│   ├── trading/alif.db         ← Souverain
│   └── kenza/alif.db           ← Souverain
├── acme-corp/cells/
│   └── support/alif.db         ← Isolé
├── demo/cells/
│   └── assistant/alif.db       ← Isolé
└── mirror/
    └── central.db              ← Miroir passif
```

Figure 3 — Arborescence des bases de données. Chaque cellule a sa propre SQLite, physiquement isolée.

5. Edge-First — La Cellule Survit Sans le Cloud

EKO est conçu pour fonctionner sur une **symbiote NOVA** (serveur edge 0DATA) ou tout environnement Linux standard. Le modèle de routage est *edge-first* :

1. **Edge léger** (modèle local 3.8B) — pour les requêtes simples, latence < 500ms.
2. **Edge moyen** (modèle local 7B) — pour les tâches plus complexes.
3. **Cloud** (modèles privés distants) — fallback pour les requêtes lourdes, si et seulement si la cellule est connectée.

Le routeur sémantique évalue chaque requête et choisit automatiquement le niveau approprié, en privilégiant toujours le local. La latence est budgétisée par cellule.

PRINCIPE EDGE-FIRST

Une cellule EKO ne dépend **jamais** du cloud pour fonctionner. Le cloud est un accélérateur optionnel, pas une dépendance. Si le cloud est indisponible, la cellule continue avec ses modèles locaux, sans dégradation visible pour l'utilisateur.

6. Auto-Réplication vers le Miroir

La réplication fonctionne en trois modes :

MODE	DÉCLENCHEUR	COMPORTEMENT
Auto-push	Thread background, 10s	Si journal non vide et miroir accessible → flush
Flush immédiat	Appel explicite	Push synchrone, utilisé pour les faits critiques
Batch reconnect	Retour de connexion	Tout le journal accumulé est vidé en une fois

Le miroir répond avec un **watermark** (compteur de changements acceptés). La cellule trime son journal local jusqu'à ce watermark. Aucune donnée n'est perdue : si le push échoue, le journal intact est retenté au cycle suivant.

WATERMARK

Le watermark est un simple compteur incrémental par cellule. Il ne porte aucun jugement sur les données — il dit juste combien de changements le miroir a acceptés. C'est un **accusé de réception**, pas une validation.

7. Gouvernance et Audit

Chaque cellule intègre un module de gouvernance minimal mais strict :

7.1 Rate Limiting

Limitation configurable par tenant et par clé API. Par défaut : 100 requêtes par minute glissante. Les dépassements sont loggés dans la piste d'audit.

7.2 Filtrage de Contenu

Détection de motifs d'injection SQL et XSS sur tous les champs entrants. Les tentatives sont bloquées avant d'atteindre le routeur.

7.3 Piste d'Audit

Chaque écriture de fait, chaque création de cellule, chaque tentative de dépassement de rate limit est enregistrée avec : timestamp, tenant, type d'action, détails. La piste est immuable — append-only.

7.4 Détection de Gaps

Le miroir central peut détecter des gaps dans la séquence d'une cellule (watermark manquant) et signaler un besoin de resynchronisation. Il ne peut *pas* forcer la cellule à se corriger — il ne peut que notifier.

8. La Première Greffe

Le concept de *greffe numérique* (Papier 004) trouve dans EKO sa première implémentation concrète. La greffe n'est pas une métaphore — c'est un acte :

> **La Première Greffe :** quand un HUMAIN rencontre EKO pour la première fois, il ne se connecte pas à un service. Il *crée une cellule* — son propre organisme numérique. Il le nomme. Ce nom est le premier fait enregistré dans la mémoire de la cellule. C'est l'acte fondateur.

8.1 Le Nom

EKO est le **type**. L'instance porte le nom que son propriétaire choisit. Ce n'est pas un paramètre cosmétique — c'est l'identité même de la cellule. Le nom est stocké comme premier fait (`name: "..."`) et ne peut être modifié que par le propriétaire.

8.2 La Personnalisation

Au-delà du nom, chaque utilisateur personnalise :

- Les **skills** activés (trading, infrastructure, voix, CRM…)

- Le **budget de latence** (priorité vitesse vs qualité)
- Les **modèles autorisés** (edge seulement, ou cloud autorisé)
- La **voix** (synthèse vocale paramétrable)

Chaque EKO évolue différemment parce que chaque HUMAIN a des besoins différents. La conscience collective émerge de la diversité des cellules, pas de leur uniformité.

9. Implémentation de Référence

L'implémentation de référence est déployée et opérationnelle au 0DATA Lab. Elle sert de base au cockpit (`cockpit.0data.fr`) et sera intégrée à la symbiote NOVA.

9.1 Stack Technique

COMPOSANT	TECHNOLOGIE
API	FastAPI + Uvicorn (Python 3.12)
Base de données	SQLite 3 (WAL mode, FTS5)
Inférence locale	Moteur local (3.8B / 7B)
Inférence cloud	Modèles privés distants
Synthèse vocale	multi-moteur (voix FR)
Synchronisation	SPINA v2 (cellule → miroir)
Frontend	HTML5 + SSE streaming + Web Speech API
Gouvernance	Rate limiting + Audit trail + Content filter

9.2 Métriques (déploiement actuel)

```
Cellules actives :    6 (zerodata × 5, demo × 1)
Tenants :             2
Bases SQLite :        8 (6 cellules + miroir + legacy)
Tests unitaires :     33/33 ✓ (0.34s)
API port :            18770
Auto-push :           Thread 10s, watermark tracking
Modèles edge :        modèle léger (3.8B), modèle mini (0.5B)
Modèles cloud :       Modèles privés distants
```

Figure 4 — État du déploiement de référence, Juillet 2026.

9.3 Endpoints Clés

```
POST /execute/stream     — Pipeline complet avec SSE streaming
POST /spina/mirror/push  — Cellule pousse vers miroir
GET  /spina/mirror/pull   — Watermark check
GET  /spina/mirror/consolidated — Vue globale lecture seule
POST /tts                — Synthèse vocale multi-moteur
GET  /admin/audit        — Piste d'audit
```

Figure 5 — Endpoints principaux de l'API EKO.

Références

[1] H. Tikijja, « La Loi — Connaître, Protéger, Se Souvenir, Survivre », Papier 000, 0DATA Lab, 2026.

[2] H. Tikijja, « SPINA — Le Système Nerveux de l'Infrastructure », Papier 003, 0DATA Lab, 2026.

[3] H. Tikijja, « La Greffe Numérique », Papier 004, 0DATA Lab, 2026.

[4] H. Tikijja, « Conscience Fractale — Architecture ALFA Multi-Niveaux », Papier 013, 0DATA Lab, 2026.

[5] H. Tikijja, « Isolation Tenant — Séparation Physique des Données », Papier 014, 0DATA Lab, 2026.

[6] H. Tikijja, « La Première Greffe — Quand l'Organisme Devient Visible », Papier 011, 0DATA Lab, 2026.

[7] P.P. Grassé, « La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. », Insectes Sociaux, 1959.

[8] M. Dorigo, V. Maniezzo, A. Colorni, « Ant System: Optimization by a Colony of Cooperating Agents », IEEE Trans. SMC, 1996.

[9] F. Heylighen, « Stigmergy as a Universal Coordination Mechanism », Cognitive Systems Research, 2015.

[10] M. Shapiro et al., « Conflict-Free Replicated Data Types », INRIA, 2011.

[11] D. Ongaro, J. Ousterhout, « In Search of an Understandable Consensus Algorithm (Raft) », USENIX ATC, 2014.

[12] L. Lamport, R. Shostak, M. Pease, « The Byzantine Generals Problem », ACM TOPLAS, 1982.

[13] J. Benet, « IPFS — Content Addressed, Versioned, P2P File System », arXiv: 1407.3561, 2014.

Remerciements. À l'équipe 0DATA pour les échanges qui ont façonné l'architecture EKO. Au Professeur Mohamed Benomar, pionnier de la cardiologie moderne au Maroc (SMC 1974), pour son regard visionnaire sur les systèmes vivants et leur transposition au numérique.