MAGELLAN AI · DEPLOYMENT REFERENCE

Audit & Trust

How the Magellan audit ledger is deployed, what guarantees it gives an auditor, and what an operator does when something looks wrong.

Document version	1.0.0
Applies to	magellan-data ≥ 1.4.0
Owner	Magellan Platform Engineering
Last reviewed	2026-04-19
Review cadence	Quarterly, alongside Compliance Officer attestation
Classification	Customer-shareable

1. Topology

1.1 Co-location decision

The audit ledger writer ships **inside the `magellan-data` container**. It is not a separate microservice. The reasons, in priority order:

1. **Hash-chain integrity is a property of the data, not the process.** A reviewer verifies `H(n) == sha256(H(n-1) || canonical(entry_n))` against the stored rows. Where the bytes were assembled is irrelevant to that check.
2. **Synchronous network hops add failure modes the ledger cannot afford.** A missing entry is worse than a slow one. Co-location removes the "write crossed a network and was lost" class of bug.
3. **RBAC is enforced at the database, not at the service boundary.** Splitting processes does not strengthen role separation; the seven Magellan roles map to Postgres roles and RLS policies, which a separate container would not change.

A separate container is justified only when (a) `magellan-data` gains a less-trusted code path (e.g. customer plugins), (b) a FedRAMP boundary explicitly requires a separate authorization decision, or (c) per-tenant KMS isolation is required for entry signing. None of these apply at v1.

1.2 Logical layout

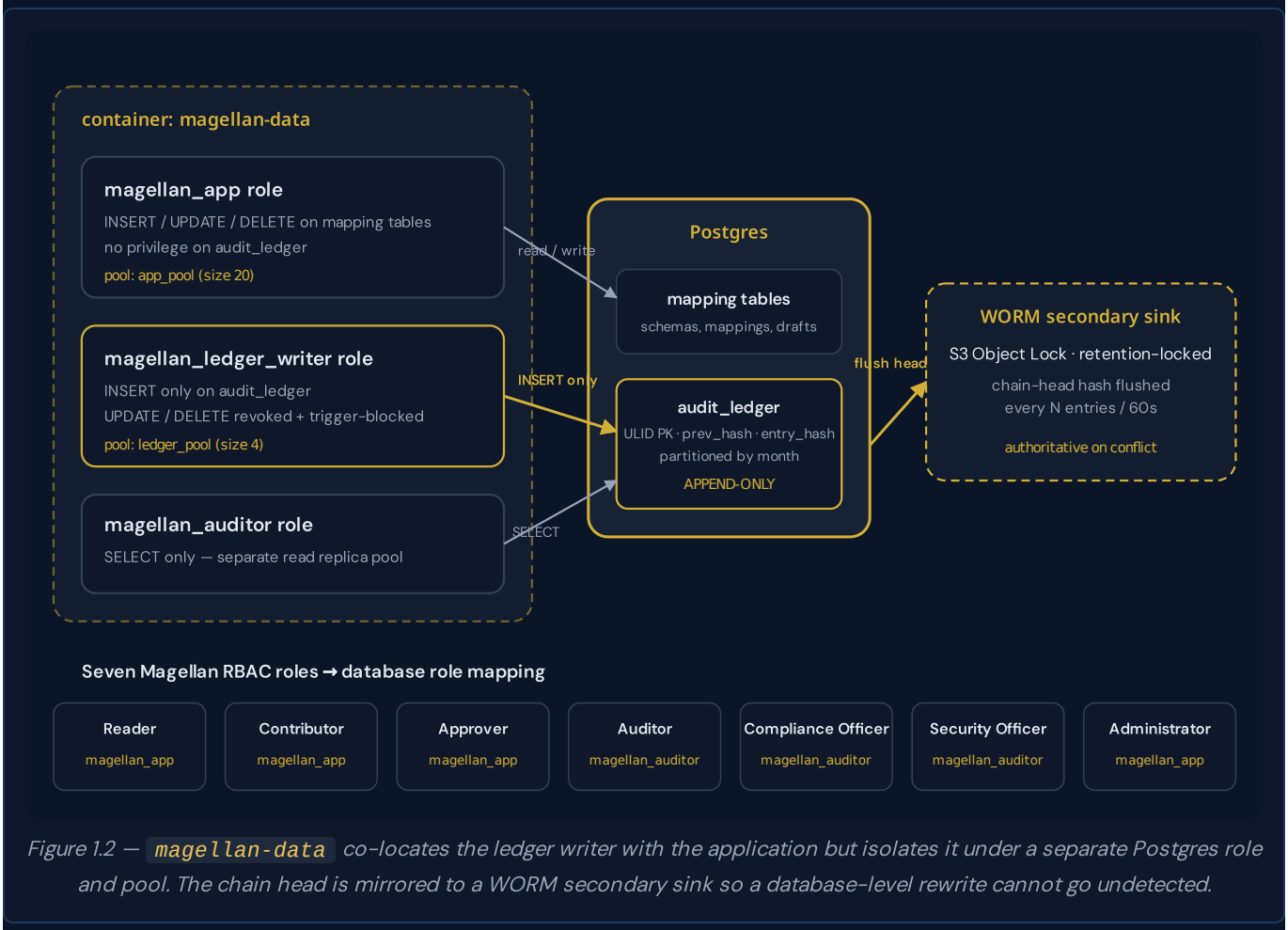


Figure 1.2 — `magellan-data` co-locates the ledger writer with the application but isolates it under a separate Postgres role and pool. The chain head is mirrored to a WORM secondary sink so a database-level rewrite cannot go undetected.

The diagram is also embedded in the white paper (`/white-paper#audit-trust`) via `src/components/AuditTopologyDiagram.tsx` . The two should stay in sync.

2. Postgres roles

Four database roles; the seventh row is documented for completeness even though no container connects under it directly.

Role	Privileges on `audit_ledger`	Privileges on mapping tables	Used by
`magellan_app`	none	`SELECT, INSERT, UPDATE, DELETE`	Application hot path (Reader, Contributor, Approver, Administrator UI sessions)
`magellan_ledger_writer`	`INSERT` only	none	Audit module inside `magellan-data`
`magellan_auditor`	`SELECT` only	`SELECT` only	Auditor and Compliance Officer dashboards (read replica)
`magellan_admin` (DDL)	DDL only — no DML	DDL only — no DML	Migrations; never holds an open application connection

`magellan_app` cannot write to `audit_ledger` even by mistake; the privilege does not exist. `magellan_ledger_writer` cannot read mapping tables; it does not need to. This separation is what an auditor will look for first, and it is the single most important property of the deployment.

2.1 Connection pools

Two distinct connection pools inside `magellan-data`:

Pool	Role	Size	Statement timeout
`app_pool`	`magellan_app`	20	30 s
`ledger_pool`	`magellan_ledger_writer`	4	5 s
`read_pool`	`magellan_auditor`	8	60 s

`ledger_pool` is intentionally small. The audit module batches at the application layer; if the pool is saturated, that is a signal of a bug, not a reason to grow it. A heavy auditor scan never starves the hot mapping path because `read_pool` lives on the read replica.

2.2 RBAC mapping (Magellan → Postgres)

The seven Magellan RBAC roles map onto the database roles as follows:

Magellan role	Database role	Notes
Reader	`magellan_app`	RLS limits visible mappings by tenant
Contributor	`magellan_app`	Adds draft/edit privileges via RLS
Approver	`magellan_app`	Adds release-signing privileges via RLS
Auditor	`magellan_auditor`	Read-only across tenant scope
Compliance Officer	`magellan_auditor`	Same as Auditor + can trigger attestation export
Security Officer	`magellan_auditor`	Same as Auditor + access to security event view
Administrator	`magellan_app`	Tenant configuration only — never DDL

Role-at-time-of-action is preserved by stamping the Magellan role name into each ledger entry at write time. The Postgres role is the *enforcement* mechanism; the Magellan role is the *recorded* fact.

3. Append-only enforcement

Three independent mechanisms, in defense-in-depth order. All three must be in place. Removing any one is a deployment defect.

3.1 Privilege revocation

```
REVOKE UPDATE, DELETE, TRUNCATE ON public.audit_ledger FROM PUBLIC;
REVOKE UPDATE, DELETE, TRUNCATE ON public.audit_ledger FROM magellan_app;
GRANT  INSERT                   ON public.audit_ledger TO   magellan_ledger_writer;
GRANT  SELECT                   ON public.audit_ledger TO   magellan_auditor;
```

3.2 Trigger-level enforcement (belt and suspenders)

Privileges can be granted by an operator with `SUPERUSER`. The trigger cannot.

```
CREATE OR REPLACE FUNCTION public.audit_ledger_block_mutation()
RETURNS trigger
LANGUAGE plpgsql
AS $$
BEGIN
  RAISE EXCEPTION 'audit_ledger is append-only (TG_OP=%)', TG_OP
    USING ERRCODE = 'insufficient_privilege';
END;
$$;

CREATE TRIGGER audit_ledger_no_update
  BEFORE UPDATE OR DELETE OR TRUNCATE
  ON public.audit_ledger
  FOR EACH STATEMENT
  EXECUTE FUNCTION public.audit_ledger_block_mutation();
```

3.3 Monthly partitioning

`audit_ledger` is range-partitioned by `entry_ulid` month. Each partition keeps its own first/last hash so a sub-chain can be verified without reading the entire history. Old partitions are detachable for archival without breaking the chain — the detached partition retains its first/last hash record.

```
CREATE TABLE public.audit_ledger (
  entry_ulid    char(26) PRIMARY KEY,
  prev_hash     bytea    NOT NULL,
  entry_hash    bytea    NOT NULL,
  actor_id      uuid     NOT NULL,
  magellan_role text     NOT NULL,          -- role-at-time-of-action
  entry_type    text     NOT NULL,          -- SchemaIngested, MappingDrafted, ...
  payload       jsonb    NOT NULL,
  written_at    timestamptz NOT NULL DEFAULT now()
) PARTITION BY RANGE (entry_ulid);
```

> *The PK is the ULID itself, not a surrogate. The 48-bit timestamp prefix is the chronological order. There is no separate `created_at` index because the PK already is one.*

4. WORM chain-head sink

A second, write-once store holds the current chain head so a database-level rewrite cannot go undetected. The application can never overwrite the WORM record.

4.1 Sink target

Setting	Value
Backend	S3 with Object Lock in compliance mode
Retention	7 years (matches HIPAA §164.316(b)(2)(i))
Object key	`audit/chain-head/<tenant>/<yyyy>/<mm>/<dd>/<ulid>.json`
Object body	`{ "as_of": "<ts>", "tail_ulid": "...", "head_hash": "<hex>", "partition": "<yyyy_mm>" }`
Encryption	SSE-KMS, customer-managed key
Versioning	Enabled (required for Object Lock)

4.2 Flush policy

The audit module flushes the chain head:

- every **N entries** (default `N = 500`), or
- every **60 seconds**, whichever comes first, and
- always **on graceful shutdown** of the container.

Configured via:

Env var	Default	Purpose
`LEDGER_WORM_SINK_URL`	—	`s3://bucket/prefix` of the WORM target
`LEDGER_WORM_KMS_KEY_ID`	—	KMS key ARN for SSE-KMS
`LEDGER_CHAIN_FLUSH_INTERVAL_S`	`60`	Time-based flush ceiling
`LEDGER_CHAIN_FLUSH_BATCH`	`500`	Entry-count flush ceiling
`LEDGER_WORM_REQUIRED`	`true`	If true, container fails health if last flush > 5 × interval ago

4.3 Conflict semantics

If the database's computed chain head disagrees with the most recent WORM record covering the same partition, **the WORM head is authoritative** and the divergence is treated as a P1 incident. See §6.3.

5. Backup, restore, retention

1. Postgres physical backups (PITR) run nightly. Backup verification **includes** running the chain-integrity check against the restored instance (§6.1) before the backup is marked good.
2. WORM partitions are immutable for the retention period; they are not included in the Postgres backup set because they are independently durable in S3 and locked.
3. On restore, the chain-head reconciliation step (§6.3) runs **before** the restored instance accepts any application traffic. A restored database that disagrees with WORM is quarantined.
4. Detached monthly partitions are exported to the same WORM bucket under `audit/partitions/<tenant>/<yyyy_mm>.parquet.gz` along with their first/last hash record. They remain verifiable without the live database.

6. Operator runbook

6.1 Verify chain integrity end-to-end

A single SQL query, suitable for a cron probe:

```
WITH ordered AS (
  SELECT entry_ulid,
         prev_hash,
         entry_hash,
         lag(entry_hash) OVER (ORDER BY entry_ulid) AS expected_prev
  FROM public.audit_ledger
)
SELECT entry_ulid
FROM   ordered
WHERE  expected_prev IS NOT NULL
  AND  expected_prev <> prev_hash
LIMIT  10;
```

Empty result = chain intact. Any row returned = corruption; go to §6.3.

6.2 Verify WORM head agrees with database

```
# 1. Read the most recent WORM head for the current partition
aws s3api list-objects-v2 \
  --bucket "$LEDGER_WORM_BUCKET" \
  --prefix "audit/chain-head/$TENANT/$(date +%Y/%m)/" \
  --query 'reverse(sort_by(Contents, &LastModified))[0].Key' \
  --output text

# 2. Compare the head_hash field to the live database
psql -At -c "
  SELECT encode(entry_hash, 'hex')
  FROM   public.audit_ledger
  ORDER  BY entry_ulid DESC
  LIMIT  1;
"
```

The two values must be identical for every entry covered by the WORM record.

6.3 Chain-head divergence (P1)

Symptoms: §6.1 returns rows, or §6.2 shows mismatch.

1. **Stop writes immediately.** Set the container to readiness=false. Drain `ledger_pool` so no new entries land while you investigate.
2. **Snapshot.** `pg_dump` of `audit_ledger`, plus a copy of the divergent WORM record, into the incident bucket.

3. **Identify the breakpoint.** Run §6.1; the first returned `entry_ulid` is where the chain broke. Pull `pg_audit` / WAL around that timestamp.
4. **Reconcile.** The WORM head is authoritative. The database is rebuilt from the most recent backup whose chain verifies (§6.1) plus any re-derivable entries; entries that cannot be re-derived are documented as lost in the incident report.
5. **Disclose.** If the divergence affected entries that were part of a completed quarterly attestation, notify the Compliance Officer; the attestation may need to be re-issued.

6.4 Rotating the WORM KMS key

KMS key rotation is supported transparently for new objects. Old objects remain readable under the previous key version for the lifetime of the retention period. Document the rotation event itself as a `LedgerKeyRotated` entry in the audit ledger.

7. Configuration reference

Env var	Required	Default	Purpose
`MAGELLAN_DB_URL_APP`	yes	—	DSN for `magellan_app` role
`MAGELLAN_DB_URL_LEDGER`	yes	—	DSN for `magellan_ledger_writer` role
`MAGELLAN_DB_URL_AUDITOR`	yes	—	DSN for `magellan_auditor` role (read replica)
`LEDGER_WORM_SINK_URL`	yes	—	`s3://bucket/prefix` for chain-head sink
`LEDGER_WORM_KMS_KEY_ID`	yes	—	KMS key ARN for SSE-KMS
`LEDGER_CHAIN_FLUSH_INTERVAL_S`	no	`60`	Max seconds between chain-head flushes
`LEDGER_CHAIN_FLUSH_BATCH`	no	`500`	Max entries between chain-head flushes
`LEDGER_WORM_REQUIRED`	no	`true`	Fail health if WORM flush stale > 5× interval
`LEDGER_HASH_ALGO`	no	`sha256`	Reserved for future migration; do not change in v1

8. Regulator crosswalk

For each control, the deployment artifact that satisfies it.

8.1 HIPAA Security Rule

Citation	Control	Satisfied by
§164.312(b)	Audit controls — record and examine activity	`audit_ledger` table (§3); chain integrity check (§6.1)
§164.312(c)(1)	Integrity — protect ePHI from improper alteration	Append-only enforcement (§3.1, §3.2); WORM secondary sink (§4)
§164.312(c)(2)	Mechanism to authenticate ePHI	Per-entry SHA-256 hash; chain-head signature in WORM (§4.1)
§164.308(a)(1)(ii)(D)	Information system activity review	Auditor read pool + dashboards under `magellan_auditor` (§2.1, §2.2)
§164.308(a)(2)	Assigned security responsibility	Magellan role mapping records who held authority at time of action (§2.2)
§164.308(a)(3)(ii)(A)	Authorization and/or supervision	Approver release signing — `ApproverReleaseSigned` ledger entry
§164.308(a)(4)(ii)(B)	Access authorization	RLS policies under `magellan_app`; role separation (§2)
§164.310(d)(2)(iv)	Data backup and storage	Backup verification includes chain integrity check (§5)
§164.316(b)(2)(i)	Time limit — retain documentation 6 years	WORM Object Lock retention set to 7 years (§4.1)

8.2 21 CFR Part 11 — Electronic records / electronic signatures

Citation	Control	Satisfied by
§11.10(a)	Validation of systems	Restore-time chain verification (§5.3); operator runbook (§6)
§11.10(b)	Generation of accurate copies	Deterministic attestation export bundle
§11.10(c)	Protection of records	Privilege revocation + trigger (§3.1, §3.2); WORM sink (§4)
§11.10(e)	Secure, computer-generated, time-stamped audit trail	ULID-keyed entries (§3.3 schema); WORM head signature (§4)
§11.10(g)	Authority checks	Magellan role recorded in entry; database role enforced (§2.2)
§11.10(j)	Holding individuals accountable	Role-at-time-of-action (§2.2); two-component approval signatures
§11.50	Signature manifestations	`ApproverReleaseSigned` entry includes signer, intent, datetime
§11.70	Signature/record linking	Signature payload references the exact ledger ULID it approves
§11.200(a)(1)	Two distinct components for non-biometric e-signatures	Approver auth: identity factor + signing intent factor

9. Versioning

This document follows semantic versioning. Backwards-incompatible changes to the deployment contract (env vars renamed, role names changed, schema column removed) require a major bump and a migration note in the corresponding `magellan-data` release.

Version	Date	Change
1.0.0	2026-04-19	Initial publication