AIR-GAPPED · ZERO-TRUST · HARDWARE-TETHERED

IRONGAP TECHNOLOGIES

www.iron-gap.com

# Vault-OS
# Technical Whitepaper

The architecture of a sovereign, on-premise enterprise AI knowledge vault — secure boot, encrypted RAG, role-bound inference, and the autonomous workflow & agent engine.

| DOCUMENT | PHASE | REVISION | CLASSIFICATION |
|---|---|---|---|
| Vault-OS Platform | 27.1.3 · Agent DAG | 1.0 | Confidential |

CONTENTS

# Table of Contents

SECTION 01

# 01 Executive Summary

Vault-OS is IronGap Technologies' sovereign AI platform: a self-contained, on-premise knowledge vault that runs large-language-model inference and retrieval-augmented generation entirely behind an air-gap, with **no internet egress by design**.

Where conventional "enterprise AI" ships sensitive corporate knowledge to third-party clouds, Vault-OS inverts the model. Documents, embeddings, chat history and audit records never leave the appliance. Every model — chat, vision, embedding — executes locally against on-host accelerators. The platform is engineered around four non-negotiable invariants that recur throughout this document:

INVARIANT 01 INVARIANT 02

### Air-Gapped

No outbound network path exists in the application logic. The former arbitrary-HTTP workflow node was removed outright; agents reach only the local inference engine and internal database.

### Zero-Trust

Every request re-validates identity, account status and token version against the database. Storage is ephemeral: ingested originals are securely destroyed after processing.

### Fail-Deadly & Hardware-Tethered

Missing secrets or an unrecognised hardware fingerprint halt the boot. The install is cryptographically bound to its silicon (TPM 2.0).

| 9 | 14 | 3 | 0 |
|---|---|---|---|
| CLEARANCE TIERS | WORKFLOW NODE TYPES | INFERENCE ENGINES | EGRESS PATHS |

This revision documents the platform through **Phase 27.1.3**, whose headline capability is the **Autonomous Workflow & Agent Engine** (Section 9): a directed-acyclic-graph runtime that lets cleared operators compose local LLM agents, vector search, sandboxed code and database actions into scheduled, event-driven automations — without ever breaching the air-gap.

SECTION 02

# 02 Design Philosophy & Threat Posture

Vault-OS assumes a hostile environment: a stolen drive, a cloned VM image, a terminated insider with a still-valid token, a compromised endpoint on the LAN. Each design decision answers a specific adversary.

## The Four Invariants in Practice

| PRINCIPLE | MECHANISM | ADVERSARY DEFEATED |
|---|---|---|
| **Air-Gapped** | Local-only inference (Ollama / vLLM / TensorRT); internal-only workflow nodes; no telemetry phone-home; CORS anchored to localhost/internal hosts only. | Data exfiltration; cloud-provider exposure |
| **Zero-Trust** | Per-request JWT verify + DB status + `token_version` check; RBAC-filtered retrieval; ephemeral dropzone storage; ledgered audit of every privileged action. | Replayed tokens; terminated insiders; lateral movement |
| **Fail-Deadly** | Absent `JWT_SECRET`, `AES_SECRET_KEY` or `AUDIT_PRIVATE_KEY` → `process.exit(1)`. Failed DB migration halts boot. Decorators fail *closed* on missing user rows. | Mis-configuration; silent degradation |
| **Hardware-Tethered** | SHA-256 fingerprint of machine UUID + MAC + TPM endorsement key, checked against `LICENSE.vault` before any service starts. | Drive theft; image cloning; unlicensed redeployment |

## Defense-in-Depth Layering

Controls are stacked so that the failure of any single layer does not collapse the security model. A request crossing the vault traverses, in order: hardware attestation → sealed-environment decryption → transport CORS/rate-limit → JWT signature → live account/token-version check → role decorator → resource-level RBAC filtering → cryptographic audit ledger.

> HONEST ENGINEERING
>
> Section 14 documents the genuinely open constraints (e.g. the `js_sandbox` node uses Node's `vm`, which is not a hard security boundary). IronGap treats candor about residual risk as part of the security model, not a marketing exception.

SECTION 03

# 03 System Architecture

Vault-OS is a three-tier appliance: a React command console, a Fastify control plane, and a Postgres + pgvector knowledge store, fronting a pluggable local inference fabric. All tiers are intended to co-reside on a single air-gapped host or an isolated internal segment.

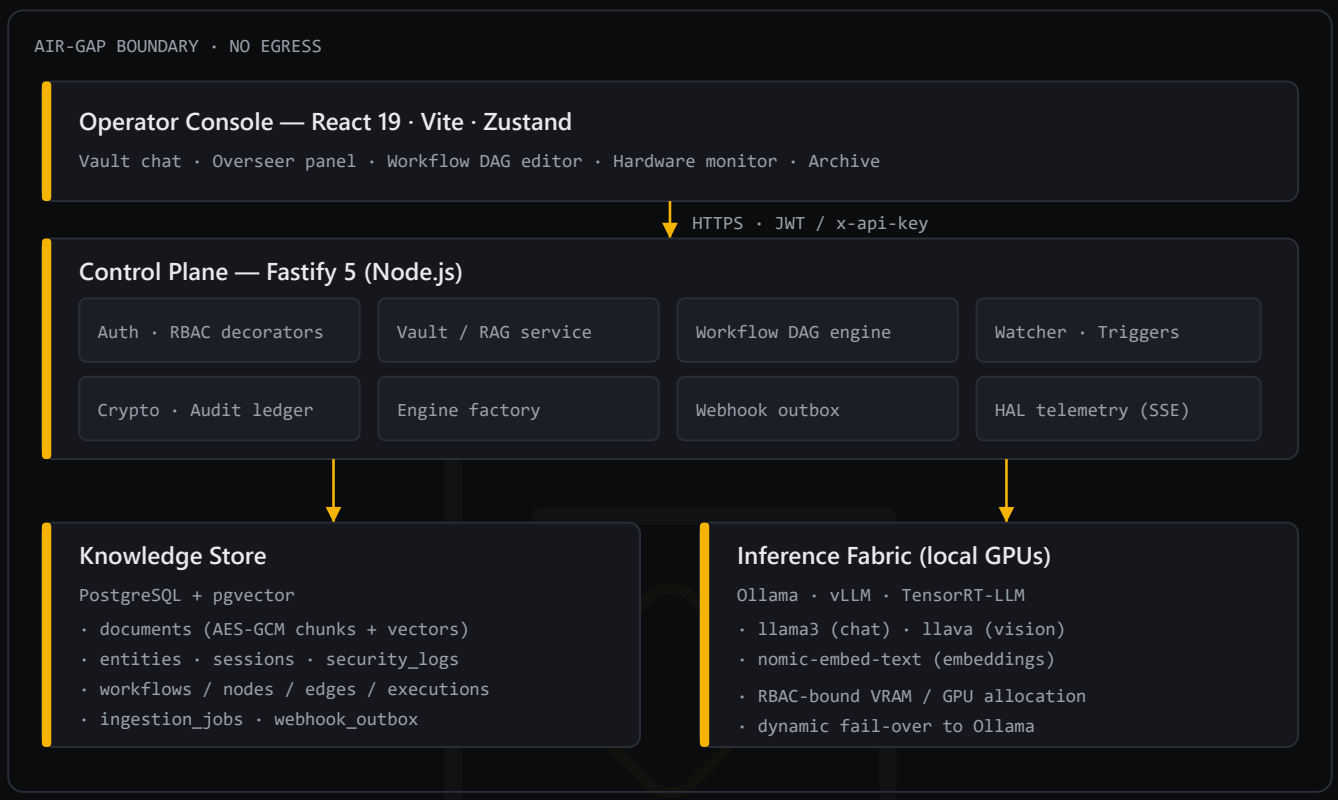



FIGURE 3.1 — VAULT-OS THREE-TIER TOPOLOGY INSIDE THE AIR-GAP BOUNDARY

## Technology Stack

| LAYER | TECHNOLOGY | ROLE |
|---|---|---|
| Control plane | Fastify 5 (Node.js), CommonJS | HTTP API, hooks, decorators |
| Console | React 19, Vite, Zustand, React-Flow | Operator UI & DAG editor |
| Datastore | PostgreSQL + pgvector (HNSW) | Vectors, metadata, ledger |
| Inference | Ollama / vLLM / TensorRT-LLM | Local chat · vision · embedding |
| Auth | @fastify/jwt, bcryptjs, otplib | Sessions, hashing, TOTP |
| Ingest | chokidar, worker_threads, mammoth/xlsx/pdf2json/textract | Watch & parse documents |
| Scheduling | node-cron, pg LISTEN/NOTIFY | Triggers & retention |
| Crypto | Node `crypto` (AES-256-GCM, scrypt, RSA/EC sign) | Sealing, at-rest, signing |
| Hardware | systeminformation, nvidia-smi, WMI/TPM | Telemetry & attestation |

SECTION 04

# 04 Secure Boot & Hardware Tethering

Vault-OS does not start as a normal Node process. The entrypoint `index.js` is a **sealed bootloader** that performs hardware attestation and decrypts the environment in memory before a single line of server code runs.


Any failure → FATAL halt. The vault never boots in a degraded state.

FIGURE 4.1 — SEALED BOOT CHAIN (INDEX.JS → SRC/SERVER.JS)

## Hardware Attestation & Silicon Lock

The Hardware Abstraction Layer ( `src/utils/hal.js` ) reads a platform-native machine identity — Win32 `ComputerSystem.UUID` , Linux `/sys/class/dmi/id/product_uuid` , or macOS `IOPlatformUUID` — combined with the primary MAC address. Where a **TPM 2.0** module is present, the endorsement-key hash (Windows) or PCR 0 reading (Linux `tpm2_pcrread` ) is folded into the fingerprint, anchoring the vault to that exact silicon. The composite string is SHA-256 hashed and compared against `LICENSE.vault` .

> ANTI-CLONE GUARANTEE
>
> A missing `LICENSE.vault` is **fatal** by default — provisioning a license for arbitrary hardware would defeat the tether. Sealing to new silicon is a deliberate, one-time act requiring `VAULT_PROVISION_LICENSE=true` . A stolen drive or cloned image therefore refuses to boot. If a TPM is present but its key cannot be read (insufficient privilege), the vault drops to an explicit software-lock fallback rather than silently weakening.

## Environment Sealing

Secrets are never stored in plaintext on a sealed appliance. The `seal` script encrypts `.env` into `.env.vault` using AES-256-GCM with a key derived from an operator master password via `scrypt` . The on-disk payload format is:

```
# .env.vault — authenticated, salted, single-line payload
SALT : IV : AUTHTAG : CIPHERTEXT   (all hex)

scryptSync(masterPassword, salt, 32) → aes-256-gcm key
decipher.setAuthTag(authTag)  // GCM integrity — tamper = decrypt failure = halt
```

For unattended/automated deployment, the master password may be supplied via `VAULT_PASSWORD` ; an unsealed development host falls back to a plaintext `.env` with a loud warning. A host-adaptation pass rewrites Docker service hostnames (e.g. `vault_db` → `127.0.0.1` ) when running outside a container.

SECTION 05

# 05 Cryptographic Subsystem

Three independent cryptographic domains protect the vault: the *sealed environment* (boot), *data at rest* (knowledge), and the *tamper-evident ledger* (audit). Each uses a distinct key, so compromise of one does not cascade.

## Encryption at Rest

Document chunks and chat messages are encrypted with **AES-256-GCM** ( `src/utils/crypto.js` ). The key is the SHA-256 digest of `AES_SECRET_KEY` — guaranteeing a valid 32-byte key regardless of the raw secret's length. Each record carries its own random 12-byte IV and 16-byte authentication tag, stored as `iv:authTag:ciphertext` (base64). On decrypt, IV/tag lengths are validated and GCM verification rejects any tampered ciphertext.

> DELIBERATE TRADE-OFF
>
> Vector **embeddings remain plaintext** — pgvector's HNSW index must perform cosine math directly on the float arrays. Only the human-readable chunk *content* is encrypted. This is logged at ingestion time as an explicit OPSEC acknowledgement, not an oversight.

## The Tamper-Evident Audit Ledger

Every privileged action (logins, model shifts, hardware-abuse attempts, workflow runs, burns) is written to `security_logs` through `logSecurityAction()` , which constructs a **hash chain**: each entry hashes `prevHash | user | action | details` with SHA-256, so any retroactive edit breaks every downstream link. Each hash is additionally **digitally signed** ( `AUDIT_PRIVATE_KEY` , PEM loaded from base64) and mirrored to an append-only physical file `logs/audit.log` .

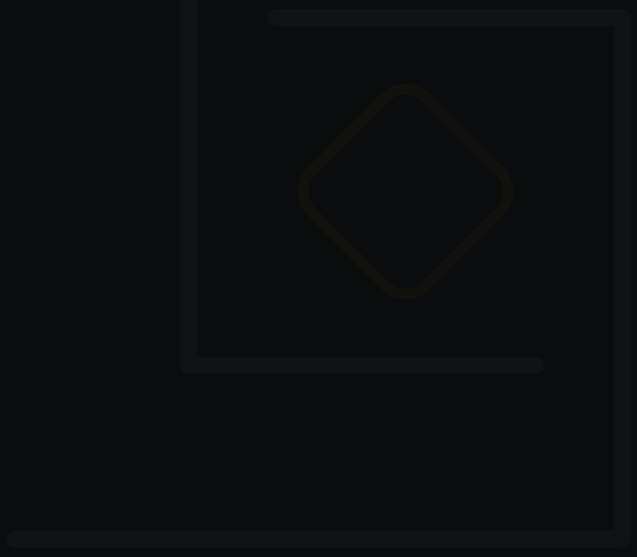
```
// Hash-chained, signed, advisory-locked ledger write
pg_advisory_xact_lock(1337)             // serialise concurrent writers
hash   = sha256(prevHash | user | action | details)
sig    = sign(hash, AUDIT_PRIVATE_KEY)
INSERT security_logs(...)  +  append logs/audit.log
emit  security.alert  → webhook outbox  // CRITICAL on BURN/FAILURE
```

A Postgres transaction-level advisory lock ( `1337` ) serialises writers so the chain can never fork under concurrency.

## ▸ Encrypted Backups & Recovery

The backup subsystem streams a `pg_dump` through AES-256-GCM under a separate `MASTER_RECOVERY_KEY` (a 32-byte hex key, length-validated), emitting a single sealed artefact prefixed with its IV and auth tag. Recovery key material is held only in the sealed environment and surfaced solely to elevated operators.

| KEY / SECRET | ALGORITHM | PROTECTS |
|---|---|---|
| `Master Password` | scrypt → AES-256-GCM | The sealed `.env.vault` |
| `AES_SECRET_KEY` | SHA-256 → AES-256-GCM | Chunks & chat at rest |
| `AUDIT_PRIVATE_KEY` | Asymmetric signature | Audit ledger integrity |
| `MASTER_RECOVERY_KEY` | AES-256-GCM | Encrypted DB backups |
| `JWT_SECRET` | HMAC-SHA256 (HS256) | Session tokens |



SECTION 06

# 06 Identity, Authentication & RBAC

Authentication is stateless in transport but stateful in trust: a signed JWT carries the claim, but every protected request re-confirms the holder still exists, is not terminated, and presents the current token version.

## Two-Factor Login with Pre-Auth Tickets

Login is a two-stage exchange. `POST /auth/verify` validates the password and, for TOTP-enrolled users, returns **not a session** but a single-purpose, 5-minute `scope:'totp_pending'` ticket bound to the username and token version. `POST /auth/totp-verify` exchanges a valid ticket plus a correct OTP for a real session. Crucially, **every authorization decorator rejects `totp_pending`-scoped tokens**, so a pre-auth ticket can never be replayed against a protected route.


FIGURE 6.1 — TWO-FACTOR LOGIN & PER-REQUEST ZERO-TRUST VERIFICATION

## Instant Global Revocation

Each user row carries a `token_version`. Every decorator compares the JWT's embedded version against the live DB value; incrementing it server-side **invalidates all outstanding sessions instantly**. Terminated accounts return `403`; a token whose subject no longer exists returns `401` (fail-closed) rather than retaining the access encoded in the still-valid JWT.

## Clearance Hierarchy & Decorators

Roles form a nine-tier lattice. Authorization is enforced by composable Fastify decorators, each of which first runs the full zero-trust identity check before testing role membership:

| DECORATOR | ADMITS |
|---|---|
| `authenticate` | Any live, non-terminated, current-version session |
| `requireManagerOrAbove` | Department_Head · Senior_Manager · VIP · Board · admin · Chairman |
| `requireAdmin` | admin · VIP · Board · Chairman |
| `requireBoardOrVIP` | VIP · Board · Chairman · admin |
| `requireChairman` | Chairman only |
| `requireNeuralClearance` | admin · Chairman — gates the entire workflow/agent engine |
| `verifyApiKey` | Machine-to-machine via `x-api-key: id.secret` (bcrypt) |

## Machine-to-Machine API Gateway

Enterprise deployments must let internal service accounts — ingestion pipelines, ETL jobs, line-of-business systems — feed the vault without a human in the loop. The API gateway serves this through keys of the form `id.raw_secret`; only a bcrypt hash of the secret is stored. Verification joins to the creator's user row so that terminating the human **disables every key they issued**. Keys are revocable, individually scoped, and track `last_used_at` for audit.

◆ AIR-GAP INTEGRITY — WHY THE GATEWAY IS NOT AN EGRESS

The gateway is **inbound-only and intra-perimeter**. It accepts calls that *arrive* at the vault from service accounts on the same isolated network segment; it never originates an outbound connection, opens no tunnel, and resolves no external host. A key authenticates *who is calling in* — it grants no internet path and no capability the air-gap forbids. Every gateway call traverses the identical zero-trust chain as a human session (status, revocation, RBAC, rate-limit) and is written to the signed ledger. The gateway therefore widens *who* may talk to the vault from inside the wire, never *where* the vault may talk to. Isolation, offline operation, and zero-trust are preserved without exception.

Role also governs **hardware entitlement** — see the engine control plane in Section 8, which binds each role to a permitted accelerator pool and a maximum VRAM ceiling via the RBAC resource matrix.

SECTION 07

# 07 Knowledge Ingestion Pipeline

Documents enter the vault through an autonomous, fault-tolerant pipeline that parses, enriches, vectorises and encrypts content — then **securely destroys the original**. Nothing transient lingers on disk.

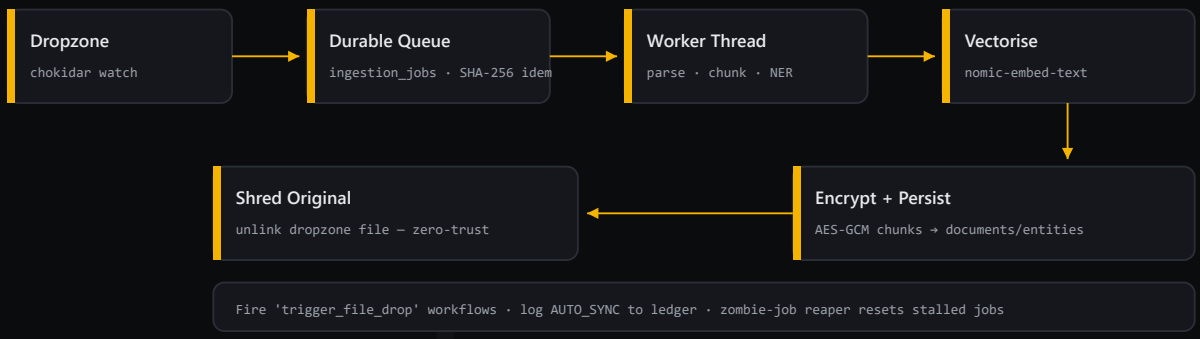



FIGURE 7.1 — AUTONOMOUS INGESTION: WATCH → QUEUE → ENRICH → ENCRYPT → DESTROY

## Autonomous Dropzone Watcher

The watcher ( `chokidar` ) monitors `vault_dropzone/` with write-stabilisation so partial uploads are never read mid-flight. New files are SHA-256 hashed and enqueued into a **durable Postgres queue** ( `ingestion_jobs` ). The hash provides **idempotency**: a file already processed successfully is discarded and securely deleted rather than re-ingested.

## Concurrency & Resilience

- **SKIP LOCKED dequeue** — workers claim jobs with `FOR UPDATE SKIP LOCKED` , allowing up to four concurrent workers with no double-processing races.
- **Worker-thread offload** — heavy parsing (PDF, DOCX, XLSX, images) runs in `worker_threads` so the event loop never blocks.
- **Zombie reaper** — jobs stuck in `processing` > 30 minutes are reset to `pending` and retried.
- **Graceful race handling** — a file deleted mid-hash resolves to a disposable ghost hash instead of crashing the watcher.

## Semantic Enrichment

Text is split with overlapping semantic chunking (≈1000 chars, 200 overlap). For each document the pipeline can attach an LLM-generated **executive summary** and extract **named entities** (PERSON / ORG / MONEY / DATE), stored in `entities` for the hybrid search path. Image drops are routed to the `llava` vision model, whose description is ingested as searchable text. Embeddings are generated with controlled concurrency (15-wide) and inserted in batched multi-row writes inside a single transaction.

> ZERO-TRUST STORAGE
>
> Once content is encrypted into Postgres, the physical dropzone file is `unlink` ed. The vault retains only the encrypted, access-controlled representation — never a loose plaintext copy.

SECTION 08

# 08 Retrieval & Inference Fabric

Answers are grounded in the vault's own documents through a hybrid retrieval pipeline, then generated by a pluggable local inference engine under strict, role-bound hardware governance.

## ▸ Hybrid RAG Retrieval

`searchSimilar()` fuses two signals. A fast lexical pass ( `ILIKE` over the entity table) finds exact/partial named-entity hits; a vector pass ranks document chunks by cosine distance ( `embedding <=> query` ). Direct entity hits receive a **+0.3 relevance boost**, after which results are decrypted, SHA-256 deduplicated to prevent context-window contamination, and truncated to the requested limit.

### RBAC-Filtered Retrieval

Every query is scoped to the caller's clearance. Non-privileged roles only retrieve chunks where `department` matches (or is `All` ) *and* the file's `clearance_tier` is at or below the user's tier; documents owned by *terminated* users are filtered out ("ghost-query" suppression). Privileged roles (Board / VIP / admin / Chairman) see the full corpus. The same lattice governs `/files` listing, downloads and entity aggregation.

## The Three Inference Engines

Vault-OS never hard-codes a model server. An `EngineFactory` exposes three interchangeable local back-ends behind one adapter interface ( `generateChat` , `generateVision` , `getModels` , `deployModel` ), so an operator can match the engine to the workload and the silicon without touching application code. All three execute *inside* the perimeter, on the organisation's own accelerators.

### 1 · Ollama — the always-on baseline

**How it serves:** GGUF-quantised models over a local REST surface ( `/api/generate` , `/api/chat` , `/api/embeddings` ) — covering mixed chat, vision ( `llava` ) and embeddings ( `nomic-embed-text` ) on a single node with minimal operational overhead. **Control levers:** per-request `num_gpu` / `num_ctx` sized from the caller's VRAM ceiling, and `keep_alive` (24 h) to pin weights in VRAM and amortise load. It is also the **universal fail-over target** for the other two engines.

### 2 · vLLM — high-throughput concurrent serving

**How it serves:** an OpenAI-compatible endpoint ( `/v1/chat/completions` , `/v1/embeddings` ) using PagedAttention and continuous batching to sustain hundreds of simultaneous operators at high aggregate tokens/sec — the workhorse for a busy enterprise floor. **Control levers:** a `max_tokens` budget derived from the role's VRAM ceiling, and the `X-vLLM-Routing-GPU` header that pins each request to the GPU(s) the role is entitled to.

### 3 · TensorRT-LLM — maximum single-node performance

**How it serves:** compiled, hardware-specific engines with multi-GPU **tensor parallelism** for the lowest achievable latency on dedicated, top-tier accelerators. **Control levers:** `tensor_parallel_constraints` enumerates exactly which GPUs a request may span, and the `X-TRT-Node-Routing` header carries the caller's role for node-level placement.

### ▸ Engine Control Plane — How Vault-OS Governs Them

Whichever engine is active, one governor mediates every call so behaviour, capacity and blast-radius stay under operator control:

- **Selection & override** — a global `ACTIVE_INFERENCE_ENGINE` (operator-set, audited as `GLOBAL_MODEL_SHIFT`) sets the default; individual calls may override it, while the globally pinned *Neural Core* model supersedes any client-requested model.
- **Hardware routing** — the RBAC resource matrix turns the caller's role into a permitted accelerator pool and a VRAM ceiling that becomes the engine's context/allocation budget. Hardware outside that entitlement is refused and logged as `UNAUTHORIZED_HARDWARE_ACCESS`.
- **Residency & reclaim** — `keep_alive` keeps hot models resident to cut cold-start latency; the `vram_purge` workflow node and `/system/models/purge` endpoint evict them on demand.
- **Dynamic fail-over** — on 5xx, timeout or out-of-memory the balancer transparently retries on local Ollama and stamps a `hardware_trace` onto the response, so operators see exactly which engine and GPU served each answer.
- **Model provisioning** — new weights are streamed to local disk (`uploadNeuralCore` → `vault_models/`) and deployed in place; there is no online registry pull.

◆ AIR-GAP INTEGRITY — THREE ENGINES, ZERO EGRESS

All three back-ends bind to loopback or internal hosts only (`OLLAMA_URL` / `VLLM_URL` / `TENSORRT_URL` resolve to `127.0.0.1` or the isolated segment). Models are loaded from files an operator places on disk — never fetched from an online hub. No engine has an outbound code path. Selecting or failing over between engines changes only *which local process does the math*, never whether a single token leaves the perimeter.

## ▸ Cognitive Load Balancer & Hardware RBAC

The resource matrix is the contract between identity and silicon. It binds each role band to an accelerator pool and a VRAM ceiling — sizing the context window and rejecting over-reach — so a junior analyst can never monopolise a reserved training-class GPU:

| ROLE BAND | ACCELERATOR POOL (DEPLOYMENT-CONFIGURABLE) | VRAM CEILING |
|---|---|---|
| `Chairman · admin · Board` | Tier-0 reserved pool — e.g. multi-GPU A100 / H100 80 GB, tensor-parallel | Unbounded |
| `VIP` | Priority pool — e.g. A100 / L40S 40 GB | 12 GB |
| `Analyst · Senior_Manager · Department_Head` | Shared inference pool — e.g. L4 / T4 class | 4 GB |
| `Employee · Standard` | Burst / integrated-GPU pool | 2 GB |

The accelerator identifiers shipped in the matrix are **deployment-configurable** — operators map each band onto their own fleet — but the mechanism is fixed: role → pool → VRAM-derived budget, enforced on every request and written to the immutable ledger. Out-of-entitlement requests are denied and recorded as `UNAUTHORIZED_HARDWARE_ACCESS`, and a failed primary engine triggers transparent fail-over to local Ollama.

## ▸ Generator–Critic Quality Loop

Optionally ( `use_critic` ), a first model draft is reviewed by a second "hyper-critical auditor" pass that hunts for hallucinations and omissions before returning the final answer — a self-contained, on-host quality gate that needs no external grader. Per-answer metrics (tokens/sec, latency, total tokens) are captured and stored with the encrypted message.

SECTION 09 · CURRENT PHASE 27.1.3

# 09 Autonomous Workflow & Agent Engine

Phase 27.1.3 elevates Vault-OS from a question-answering vault to an **autonomous orchestration platform**. Cleared operators visually compose directed acyclic graphs (DAGs) of agents, tools and conditions that execute on triggers — entirely inside the air-gap.

> CLEARANCE BOUNDARY
>
> The entire engine sits behind `requireNeuralClearance` — only **admin and Chairman** may author, execute, or inspect workflows. The error on violation is explicit: "*IronGap Protocol Violation. Only sysadmins can access the autonomous agent network.*"

## The DAG Execution Model

On execution, the engine loads the workflow's nodes and edges, then performs a **Kahn topological sort**; a graph that fails to fully sort is rejected as containing a cycle. Nodes run in dependency order against a shared, JSON-interpolated `executionState`. The run is wrapped in a configurable timeout (default 300s) and persisted step-by-step into the `agent_executions` ledger.

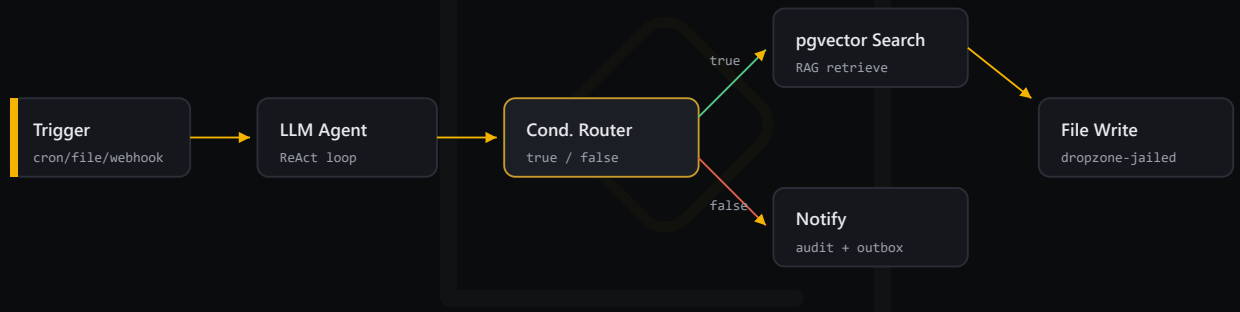



FIGURE 9.1 — AN EXAMPLE DAG: BRANCHING AGENT WORKFLOW WITH CONDITIONAL ROUTING

## Node Catalogue

| NODE | FUNCTION | SECURITY CONTROL |
|---|---|---|
| `trigger_*` | Manual, cron, webhook, file-drop, or pg-event entrypoints | Neural clearance to author |
| `llm_agent` / `agent_react` | ReAct reasoning loop with tools: `pgvector_search`, `internal_ner`, `final_answer` | 15-iteration loop cap (VRAM guard) |
| `pgvector_search` | Department-scoped vector retrieval | Owner department filter |
| `postgres_query` | Read-only data query | **SELECT-only**; users/api_keys/config tables denied; dept filter enforced |
| `summarize` / `extract_entities` | Local-LLM text transforms | Local engine only |
| `vision_analyzer` | llava image understanding | Path-checked input |
| `conditional_router` | Branches execution on a guarded expression | Character-whitelisted evaluator |
| `js_sandbox` | Inline JS transform | 2s timeout · see §14 caveat |
| `file_system_write` | Write artefact to dropzone | **Path-traversal jailed** to `vault_dropzone/` |
| `run_docker_container` | Execute whitelisted container | Image allow-list · `--network=none` · `spawnSync` argv (no shell) |
| `notify` · `vram_purge` | Internal alert / free model VRAM | No external dispatch |

### Execution Guardrails

- **Per-node retries with exponential back-off** ( `retries` , `retry_delay_ms` ) and conditional `skip_if` gating.
- **Branch pruning** — nodes whose every incoming edge is inactive (e.g. the untaken side of a router) are marked `skipped` rather than executed.
- **Air-gap enforcement** — the legacy arbitrary-HTTP node was deliberately removed; agents cannot originate outbound requests.
- **Immutable ledger** — deleting a workflow preserves its `agent_executions` history; every run emits `WORKFLOW_START/SUCCESS/FAILURE` to the signed audit chain.

### Triggering Surfaces

Workflows execute via four paths, all funnelled through the same engine: the authenticated `POST /workflows/:id/execute` ; an API-key webhook ( `POST /workflows/webhook/:id` , re-checking the key owner's neural clearance); the file-drop watcher; and the omnichannel `TriggerManager` (Section 10) covering cron schedules and Postgres event notifications.

◆ AIR-GAP INTEGRITY — TRIGGERS FIRE INWARD, NEVER OUTWARD

Every trigger is an **inbound or wholly internal** stimulus: the webhook is a call *received* from an internally-authenticated service account (API key + neural clearance), while cron, file-drop and Postgres `NOTIFY` originate entirely within the host. An agent's toolset is likewise confined to the local engine and the internal database — the arbitrary-HTTP node was removed precisely so no workflow can reach the outside. Automation amplifies what the vault does *to its own data*, never its reach beyond the wire.

SECTION 10

# 10 Eventing, Triggers & Telemetry

Vault-OS is event-driven internally while remaining silent externally. A transactional outbox, a database event bus and a live hardware-telemetry stream coordinate the platform without a single outbound connection.

## Transactional Outbox

Domain events are recorded via `emitEvent()` into `webhook_outbox` in the same logical flow as the action that produced them, guaranteeing at-least-once capture even across restarts. A polling dispatcher claims rows with `SKIP LOCKED`, processes up to four concurrently, and resets zombies after five minutes.

◆ AIR-GAP INTEGRITY — AN OUTBOX THAT STAYS IN

Despite the name, the outbox **transmits nothing outward in the air-gapped build**: the dispatcher *captures and logs* each event to the internal ledger and clears it. The "webhook" abstraction is retained only as a clean internal hook point for a future *signed, on-host* native dispatcher. There is no remote sink, no callback URL, and no network write — the pattern buys reliability and auditability, not connectivity.

## Omnichannel Triggers

| CHANNEL | MECHANISM | NODE TYPE |
|---|---|---|
| Schedule | node-cron (validated expressions) | trigger_cron |
| Database event | pg LISTEN/NOTIFY on `vault_os_events` | trigger_pg_event |
| File arrival | chokidar dropzone watcher | trigger_file_drop |
| Programmatic | API-key webhook endpoint | trigger_webhook |

## Live Hardware Telemetry (HAL)

The console's Hardware Monitor is fed by a Server-Sent-Events stream broadcasting every two seconds across the appliance's compute fleet. Telemetry is split into a cheap fast path (aggregate and per-core CPU load, RAM/swap — refreshed each tick) and a throttled slow path (the full GPU cluster via async `nvidia-smi`, disk volumes, network throughput and top processes — refreshed on an 8-second background cycle) so probing never stalls the event loop. Each accelerator is enriched with precise VRAM and utilisation figures; the stream cleans up automatically on client disconnect.

◆ AIR-GAP INTEGRITY — OBSERVABILITY WITHOUT PHONE-HOME

Telemetry is collected on the host and streamed **only** to the operator's authenticated console over the internal connection. Vault-OS ships no agent that reports to a vendor, no usage analytics, and no remote monitoring endpoint. Operations teams get full fleet visibility; no third party gets a single metric.

| 2s | 4 | 4 | 1337 |
|---|---|---|---|
| TELEMETRY CADENCE | OUTBOX WORKERS | TRIGGER CHANNELS | LEDGER ADVISORY LOCK |

SECTION 11

# 11 Data Lifecycle, Burn & Self-Update

## Retention

A nightly cron sweep ( `0 0 * * *` ) deletes chat sessions older than 30 days, cascading to their messages — bounding the standing footprint of conversational data. Document retention is governed separately by clearance and operator action.

## The Burn Protocol

IRREVERSIBLE

`DELETE /admin/burn` performs a NIST-style multi-pass shred of the dropzone, archive and model directories and purges the database — a deliberate scorched-earth response to physical compromise. It is presently reachable at VIP clearance; Section 14 flags the authority model (Chairman-only and/or a confirmation token) as a pending product decision.

## Zero-Downtime Self-Update

Patches are applied through a staged, two-person-friendly flow: `POST /system/update` (admin) stages a patch, and `POST /system/update/confirm` (**Chairman**) authorises it. The mechanism spawns a second server instance on port 3001 and transparently routes live traffic to it through an `onRequest` proxy hook keyed on `global.UPDATED_SERVER_URL`, achieving hot cut-over without dropping connections. Failed updates retain a downloadable diagnostic log.

◆ AIR-GAP INTEGRITY — UPDATES ARRIVE BY HAND, NOT BY WIRE

The vault never reaches out to a patch server. Update bundles are **staged on local disk by an operator** (sealed media / sneakernet) and applied from there; the hot cut-over proxy targets a sibling process on `127.0.0.1:3001`, so traffic never leaves the loopback interface. Patching is an internal, dual-authorised act — not an online auto-update.

ROADMAP

Because self-update copies staged files over the running root, it presents a privileged surface that IronGap intends to harden with cryptographic signing/verification of update bundles (Section 14).

## Self-Healing Schema

On boot the server runs idempotent migrations that converge the live schema — standardising column names, adding missing columns/tables (workflow DAG tables, ephemeral tickets, outbox, execution ledger), and extending the `user_role` enum. A migration failure triggers a fail-deadly halt rather than running against an inconsistent store.

SECTION 12

# 12 Operator Console (Frontend)

The console is a React 19 + Vite single-page application with Zustand state stores, designed as a dark, instrument-panel command surface for vault operators.

### Vault Chat & Sessions

RAG-grounded conversation with cited sources, optional incognito sessions, the generator–critic toggle, and auto-titled session history.

### Workflow DAG Editor

React-Flow canvas with typed handles, dynamic node configuration, Dagre auto-layout, and live execution-log streaming for the Phase 27.1.3 engine.

### Overseer Panel

Administrative cockpit — user & clearance management, API-key issuance, model/engine selection, security-log review and system diagnostics.

### Hardware Monitor

Live SSE dashboard of CPU, per-core load, RAM/swap, the GPU cluster, disks, network and top processes.

Shared UI primitives (Button, Card, Input, TextArea, Modal, Skeleton) and a layout shell (Header, Sidebar, Gateway login form, Archive table) keep the surface consistent. The store layer threads the TOTP pre-auth ticket through the two-stage login and holds the JWT for authenticated calls; a dedicated workflow store manages DAG editing state.

SECTION 13

# 13 API Surface Reference

All endpoints are served under `/api/v1`. The clearance column lists the decorator gating each route.

◆ AIR-GAP INTEGRITY — EVERY ROUTE IS AN INGRESS

The entire surface below is **inbound**: each route is something a client on the isolated network *calls into*, bound to loopback or the internal segment and CORS-anchored to it. Not one endpoint causes the vault to initiate an outbound connection — including `/gateway/ingest` and `/workflows/webhook/:id`, which receive data rather than send it. The API expands controlled, audited *access in*; it never opens a path *out*.

| M | ENDPOINT | CLEARANCE |
|---|---|---|
| POST | /auth/verify · /auth/totp-verify · /auth/reset-initial | Public (login) |
| POST | /auth/change-password · /auth/resolve | authenticate |
| PUT | /admin/users/:id | requireAdmin |
| POST | /admin/users/:id/generate-totp · /reset-password | requireAdmin |
| POST | /gateway/ingest | verifyApiKey |
| POST | /ingest · /upload | authenticate |
| GET | /files · /files/dropzone · /files/:id/download | authenticate (RBAC-filtered) |
| DELETE | /files/:id | requireAdmin |
| GET | /entities | requireAdmin |
| POST | /search · /ask | authenticate |
| GET/DELETE | /sessions · /sessions/:id/messages | authenticate |
| GET | /hardware · /models · /security-logs · /system/diagnostics | requireAdmin |
| GET/POST | /system/hardware/pool · /allocate | requireAdmin |
| GET/PUT/POST | /system/models/active · /upload · /purge · /system/engine | requireAdmin |
| GET/POST/DEL | /admin/api-keys ... | requireAdmin |
| GET | /system/backup · /backup/config · /recovery-key | requireAdmin |
| POST | /system/update | requireAdmin |
| POST | /system/update/confirm | requireChairman |
| DELETE | /admin/burn | requireBoardOrVIP |
| GET/POST/DEL | /workflows · /workflows/:id · /save · /:id/execute · /:id/executions | requireNeuralClearance |
| POST | /workflows/webhook/:id | verifyApiKey (+ neural owner) |
| — | /hr/* (human-resources module) | requireManagerOrAbove |

SECTION 14

# 14 Security Posture & Hardening Roadmap

IronGap publishes residual risk rather than concealing it. The controls below are **in force**; the constraints that follow are **known and tracked**.

## Controls in Force

HARDENED

- Hardware license is fail-deadly; no auto-mint on foreign silicon.
- Postgres bound to `127.0.0.1`, not `0.0.0.0`.
- CORS origin regex fully anchored against look-alike hosts.
- Auth decorators fail-closed on deleted users.
- TOTP bypass closed via single-purpose pre-auth tickets.

HARDENED

- `/files` RBAC leak closed — listing now clearance-scoped.
- Docker node command-injection removed (`spawnSync`, no shell).
- Seed-admin secret no longer logged; reset forced on first login.
- `postgres_query` node is SELECT-only with a credential-table denylist.
- Workflow file writes are path-traversal jailed to the dropzone.

## Known Constraints (Tracked)

| CONSTRAINT | STATUS & PLAN |
|---|---|
| OPEN `js_sandbox` uses Node `vm` | Not a true security boundary (escapable to the host). Admin-only, but slated for replacement with `isolated-vm` or removal. |
| OPEN Burn authority | Reachable at VIP via `requireBoardOrVIP` with no re-auth. Candidate for Chairman-only gating and a confirmation token. |
| REVIEW Self-update surface | Spawns a child server and copies staged files over root; to be hardened with signed, verified update bundles. |
| MINOR Legacy decrypt path | `decrypt()` returns input unchanged on malformed (pre-encryption) data — a fail-open legacy compatibility path to retire. |
| MINOR Dropzone listing | Raw pre-classification staging is visible to any authenticated user; to be scoped. |

POSTURE STATEMENT

Every item above is intentionally enumerated. The presence of a tracked-constraints register — not its absence — is the security signal: Vault-OS is engineered, audited, and iterated under the same zero-trust scrutiny it imposes on its users.

APPENDIX A

# A Schema, Environment & Glossary

## Core Data Model

| TABLE | PURPOSE |
| --- | --- |
| users | Identities, role, status, token_version, TOTP, reset flag |
| files / documents | File metadata; encrypted chunks + plaintext vectors + metadata |
| entities | Extracted PERSON/ORG/MONEY/DATE for hybrid search |
| sessions / chat_messages | Conversations; encrypted message bodies + metrics |
| security_logs | Hash-chained, signed audit ledger |
| ingestion_jobs | Durable ingest queue with hash idempotency |
| workflows / workflow_nodes / workflow_edges | Persisted DAG definitions with UI positions & handles |
| agent_executions | Immutable per-run execution state ledger |
| webhook_outbox | Transactional event outbox |
| api_keys / ephemeral_tickets / system_config | M2M keys; iframe handshake; runtime config |

## Required Environment Secrets

| VARIABLE | PURPOSE · FAIL-DEADLY? |
|---|---|
| JWT_SECRET | HS256 session signing · YES |
| AES_SECRET_KEY | At-rest encryption key · YES |
| AUDIT_PRIVATE_KEY | Audit-ledger signing (base64 PEM) · YES |
| MASTER_RECOVERY_KEY | Encrypted backups (32-byte hex) |
| VAULT_PASSWORD | Unattended unseal of .env.vault |
| VAULT_PROVISION_LICENSE | One-time license sealing to hardware |
| INITIAL_ADMIN_PASSWORD | Bootstrap admin secret (reset forced) |
| ACTIVE_INFERENCE_ENGINE / ACTIVE_LLM_MODEL | Engine & model selection |
| OLLAMA_URL / VLLM_URL / TENSORRT_URL | Local engine endpoints |
| DB_HOST / DB_PORT / DB_USER / DB_PASSWORD / DB_NAME | Postgres connection |

## Glossary

| | |
|---|---|
| Air-Gap | No network path between the vault and any external system. |
| Fail-Deadly | On ambiguity or missing trust material, halt — never degrade silently. |
| Silicon Lock | Binding the install to a TPM endorsement key / machine fingerprint. |
| Zero-Trust Storage | Ingested originals destroyed; only encrypted, access-controlled copies persist. |
| Neural Clearance | admin/Chairman-only gate over the autonomous agent engine. |
| Ghost Query | Retrieval of documents owned by terminated users — suppressed by RBAC. |
| Burn Protocol | Irreversible multi-pass destruction of vault data on compromise. |

IRONGAP TECHNOLOGIES

Vault-OS · Phase 27.1.3 · Technical Whitepaper · Revision 1.0

www.iron-gap.com · Confidential — Internal Distribution Only · © IronGap Technologies