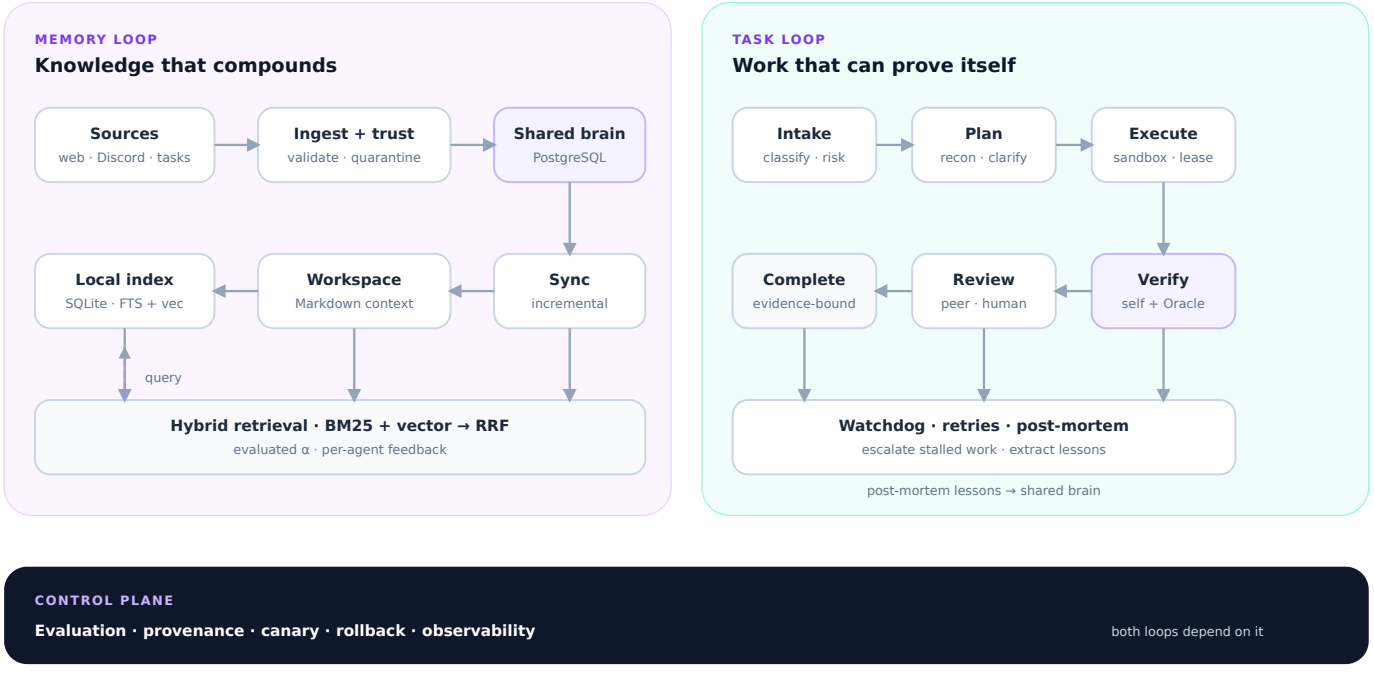
OPEN REFERENCE · V1.0

A self-learning AI harness you can actually trust.

A practical architecture for agent systems that remember, improve and prove their work. Distilled from an operating 8-agent agency and hardened with evaluation, provenance, idempotency and observability.

Reference design · not a product · adapt the trade-offs to your domain

3 memory layers	**8** workflow phases
10 task states	**1** rule: evaluate first



Two loops, one control plane. Learning without verification is drift; verification without memory is repetition.

Two loops, one system

The memory loop makes agents more useful over time. The task loop makes their output verifiable. Neither is safe without the other.

MEMORY

A three-layer memory that learns without losing trust

Workspace files, a fast local hybrid index and a shared global brain — with provenance, evaluation and decay built in.

- BM25 + vector retrieval
- Provenance and quarantine
- Strategy evolution with rollback

Read the design →

WORKFLOW

A task lifecycle that can prove it is done

Classification, recon, retries, peer review, an independent Oracle and DAG orchestration — with idempotent states and evidence-bound completion.

- 10 states, explicit transitions
- Independent Oracle
- Watchdog and post-mortem

Read the design →

Design principles

These are the constraints I would not remove. They are the difference between a demo that appears to learn and a system you can operate.

Evaluate before autonomy

A self-improving system without a holdout set learns to look better, not to be better. Every prompt, strategy and retrieval weight change must pass an offline gate.

Memory is a trust boundary

Web pages, Discord, attachments and peer agents can poison memory. Store provenance, quarantine unverified knowledge and never clone a strategy without evaluation.

Exactly once, not “usually once”

Tasks, retries, payments and deployments need idempotency keys, leases and terminal-state guards. Cron overlap is a bug, not an edge case.

Evidence, not claims

“Completed” must mean reproducible evidence: command, output hash, artifact, timestamp and the verifier that checked it. Oracles should not trust the agent’s summary.

Humans at risk gates

Human approval should depend on reversibility, spend, data access and blast radius — not only task complexity.

Observe the loop

Track recall, precision, stale memory, retry rate, Oracle pass rate, cost per task and drift. If you cannot see the loop, you cannot improve it.

What I changed from the original design

The original decks describe a strong loop. This reference adds the parts that make it safe to run unattended.

- ✓ **Evaluation harness**
  Golden tasks, holdouts, regression suites and canaries before any α, strategy or model change.
- ✓ **Provenance & trust labels**
  Raw → verified → trusted, with source, author, timestamp and a quarantine path.
- ✓ **Incremental memory sync**
  Event-driven updates plus a reconciliation sweep; version embeddings instead of re-embedding everything.
- ✓ **Idempotent workflow**
  Leases, transition guards, dependency-cycle checks and replay-safe retries.
- ✓ **Evidence contract**
  Oracle re-checks executable evidence; stale or self-supplied proof cannot pass.

✓ **Operational visibility**

SLOs and dashboards for every loop: retrieval, memory, retries, Oracle, cost and human gates.

If you are building this

A minimum implementation order that avoids the common traps.

1. Write the task state machine and transition table before code.
2. Define a memory record schema with provenance and trust level.
3. Build the evaluation set before the evolution engine.
4. Make every scheduled job idempotent and lock-protected.
5. Log retrieval, feedback, strategy changes and Oracle evidence.
6. Add backup/restore and a disaster-recovery drill.

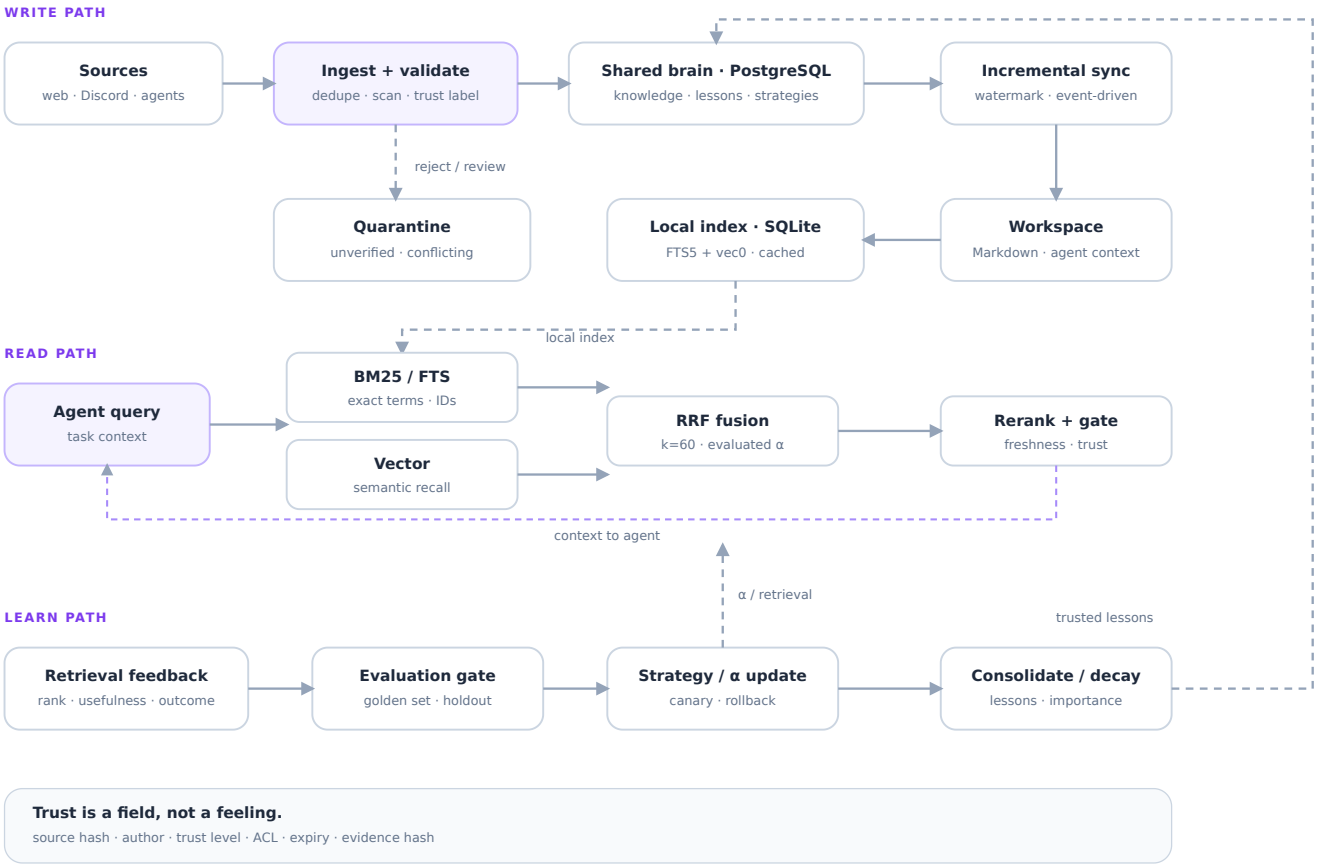
Use it as a starting point, not a blueprint. The right number of states, agents and memory layers depends on your risk, volume and team. Keep the loops; question the constants.

MEMORY LOOP

Memory that improves the agent — without quietly poisoning it.

A three-layer memory system: readable workspace files, a fast local hybrid index and a shared global brain. The additions that matter are provenance, incremental sync and an evaluation gate on everything that claims to be an improvement.

TL;DR — Keep the three layers. Make writes transactional and attributable. Retrieve with BM25 + vectors. Evaluate every strategy and retrieval-weight change on a holdout set. Never let an agent's own score be the only evidence that it got better.



The write path is governed; the read path is hybrid; the learn path is evaluated before it changes behaviour.

The three layers

Each layer has a different job. Do not collapse them into one database.

01 Workspace memory

`~/.openclaw/workspace-{agent}/memory/`

Human-readable Markdown, materialised from the shared brain. This is the agent's context window: knowledge digest, peer knowledge, strategies, reflections and lessons.

Markdown **per agent** **reviewable**

02 Local hybrid index

`~/.openclaw/memory/{agent}.sqlite`

Fast, offline retrieval with FTS5 for exact terms and a vector table for semantic recall. Embeds once and caches by content hash.

SQLite **FTS5 + vec0** **no network**

03 Shared global brain

`PostgreSQL · schema: agency`

Durable, cross-agent knowledge: knowledge, lessons, strategies, reflections, scores, competency and retrieval feedback. The source of truth for learning.

pgvector **tsvector** **7 tables**

Add a control plane. Ingestion, consolidation, decay, evolution and transfer are jobs, not tables. Put them behind a queue with locks, retries and audit logs.

The memory loop

A write is not finished when it is stored. It is finished when it is attributable, retrievable and evaluated.

1 Ingest
Research or task output becomes a candidate record with source, author and timestamp.

2 Validate
Deduplicate, classify and score trust. Unverified content goes to quarantine.

3 Persist
Write to PostgreSQL with embedding + tsvector; keep the event log.

4 Sync
Incremental materialisation to workspace Markdown, then reindex only changed chunks.

5 Retrieve
Local or global hybrid search: BM25 + vector, fused with RRF.

6 Learn
Log results and feedback; consolidate lessons; evaluate before any strategy or α change.

Hybrid retrieval

BM25 catches exact identifiers, error codes and names that embeddings blur. Vectors catch conceptual matches. Fuse both; do not choose one.

Fusion

```
score = (1 - α) / (k + rank_bm25)
      + α     / (k + rank_vector)

k = 60
α = 0.5 initially, clamped [0.2, 0.8]
α is per agent and changes only after
an offline evaluation gate.
```

When to use which index

- **Local SQLite** — current task context, private workspace, offline.
- **Global PostgreSQL** — peer knowledge, durable lessons, cross-agent strategies.
- **Both** — merge candidates, then rerank with a small model or deterministic rules.

Do not tune α from clicks alone. “Was helpful” is sparse and biased. Use a labelled query set, measure recall@k and answer quality, and ship α changes as canaries.

Schema that supports trust

The original seven tables are a good base. Add provenance and trust to every knowledge record.

TABLE	PURPOSE
`agency.knowledge`	Durable facts and research with embeddings and full-text index.
`agency.lessons`	Trigger → action lessons extracted from incidents and reviews.
`agency.strategy_registry`	Versioned strategy prompts, scores, parent/child lineage.
`agency.reflections`	Post-task analysis and proposed rewrites.
`agency.score_ledger`	Immutable score events — not a mutable average.
`agency.agent_competency`	Per-agent, per-skill tier and evidence.
`agency.retrieval_feedback`	Query, result, rank and usefulness — the evaluation signal.

```
-- minimum trust fields
source_url, source_hash, author_agent, created_at,
trust_level: raw | verified | trusted,
trust_reason, expires_at, acl_scope
```

Consolidation, decay and forgetting

Decay keeps memory lean, but forgetting must be deliberate and auditable.

CLASS	DEFAULT	RULE
Permanent	Identity, soul, core memory	Never auto-archive; human pin only.
Short-term	Daily logs	14 days, importance-weighted; archived, not deleted.
Long-term	Reports, audits, digests	90 days, then summary retained.

Importance multiplies the effective age, but keep a hard retention floor and an archive. A production incident at 2× importance should not become immortal by accident.

Strategy evolution with a brake

UCB selection and reflection are useful. They are not evidence on their own.

1. **Register**
2. **Select**
3. **Execute**
4. **Score**
5. **Reflect**
6. **Evolve**

- **Minimum sample:** do not promote or clone a strategy on fewer than N scored tasks.
- **Holdout gate:** a new strategy must beat its parent on a frozen evaluation set.
- **Canary:** route a small share of live tasks; automatic rollback on regression.
- **Diversity:** cosine similarity alone is not enough; require task-level diversity too.
- **Lineage:** every strategy stores parent, evidence, score history and why it was retired.

What to measure

METRIC	QUESTION IT ANSWERS
Recall@k	Are the right memories retrieved for a known query set?
Precision / noise	How much irrelevant context enters the window?
Staleness	Median age of retrieved memory; drift from source.
Poison rate	Quarantined or rejected records per 1,000 ingests.
Strategy delta	Win rate of a new strategy versus its parent on a holdout set.
Cost / query	Embedding, vector and rerank cost per successful retrieval.

Failure modes

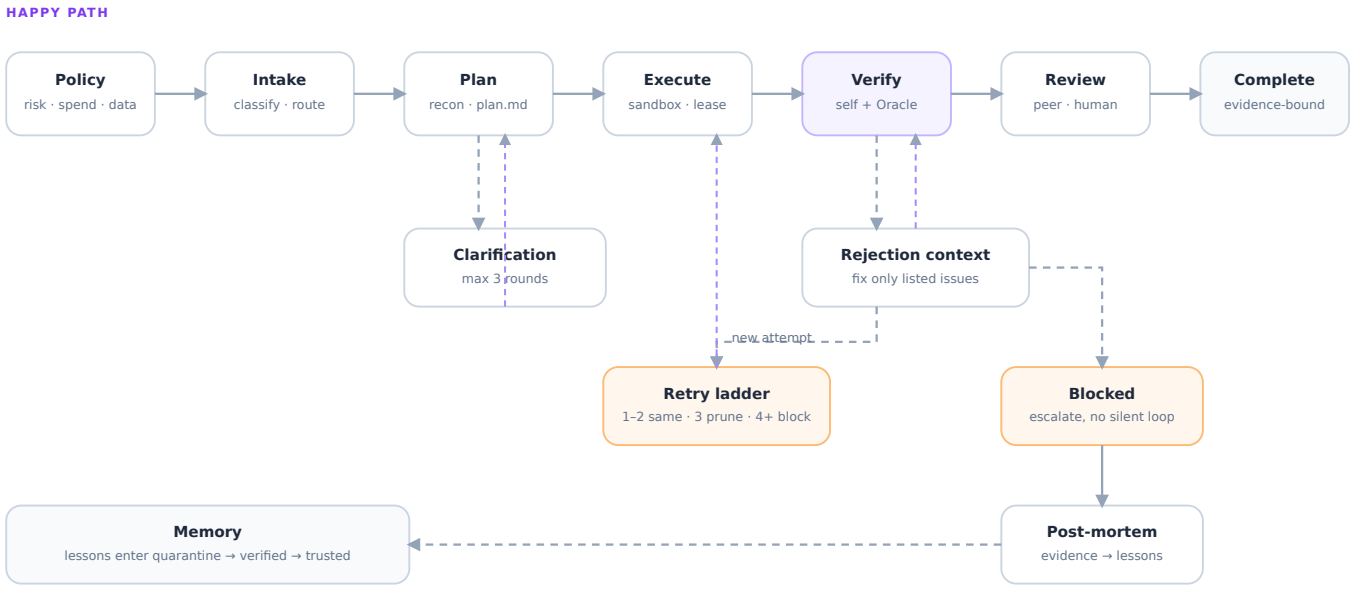
FAILURE	CAUSE	FIX
Stale memory	Six-hour full sync; no watermark	Event-driven updates + reconciliation; surface freshness.
Poisoned knowledge	Any agent can ingest web/Discord text	Provenance, trust level, quarantine, injection checks.
Runaway evolution	Self-scored strategies; no holdout	Offline evaluation gate, canary, minimum sample, rollback.
Index drift	SQLite rebuilt from changed files	Content hashes, incremental index, embedding version.
Privacy leak	Shared brain across agents	ACL by agent/tenant, PII scan, retention policy.

TASK LOOP

A workflow that can prove the task is done.

Ten states, eight phases plus a policy gate, and one rule: completion is evidence, not a claim. This reference keeps the original lifecycle and adds the invariants that make it safe to run unattended.

TL;DR — Keep the state machine and the Oracle. Add idempotent transitions, a real evidence contract, risk-based human gates and evaluation before a strategy is promoted.



Verification is not a formality: self-audit and Oracle are independent checks; evidence determines completion.

Eight phases plus a policy gate. Clarification is bounded, retries are structured, and blocked is a real state.

The lifecycle at a glance

Eight phases plus a policy gate. Self-audit and the Oracle are parallel checks, not a chain; the Oracle is mandatory for promise mode and high-risk audit mode.

00

Intake & policy

Classify type and complexity, attach risk, spend and data-access policy. Pick a strategy with enough evidence.

01

Recon & plan

Inspect the workspace, images and memory. Write plan.md with steps, risks, dependencies and a verification plan.

02

Execute

Run with a scoped sandbox and a heartbeat. Every tool call that mutates state is idempotent.

03

Self-audit

Check acceptance criteria against the plan. Produce structured evidence, not prose only.

04

Oracle

Independent, clean-context verifier re-runs or attests the evidence. No access to the agent’s session history.

05

Peer review

Blind, rotated reviewer checks quality and risk. Same-pool peers are not enough for critical work.

06

Human gate

Approval depends on reversibility, spend, data access and blast radius — not just complexity.

07

Orchestrate

DAG dependencies, leases, cycle detection and workload-aware routing.

08

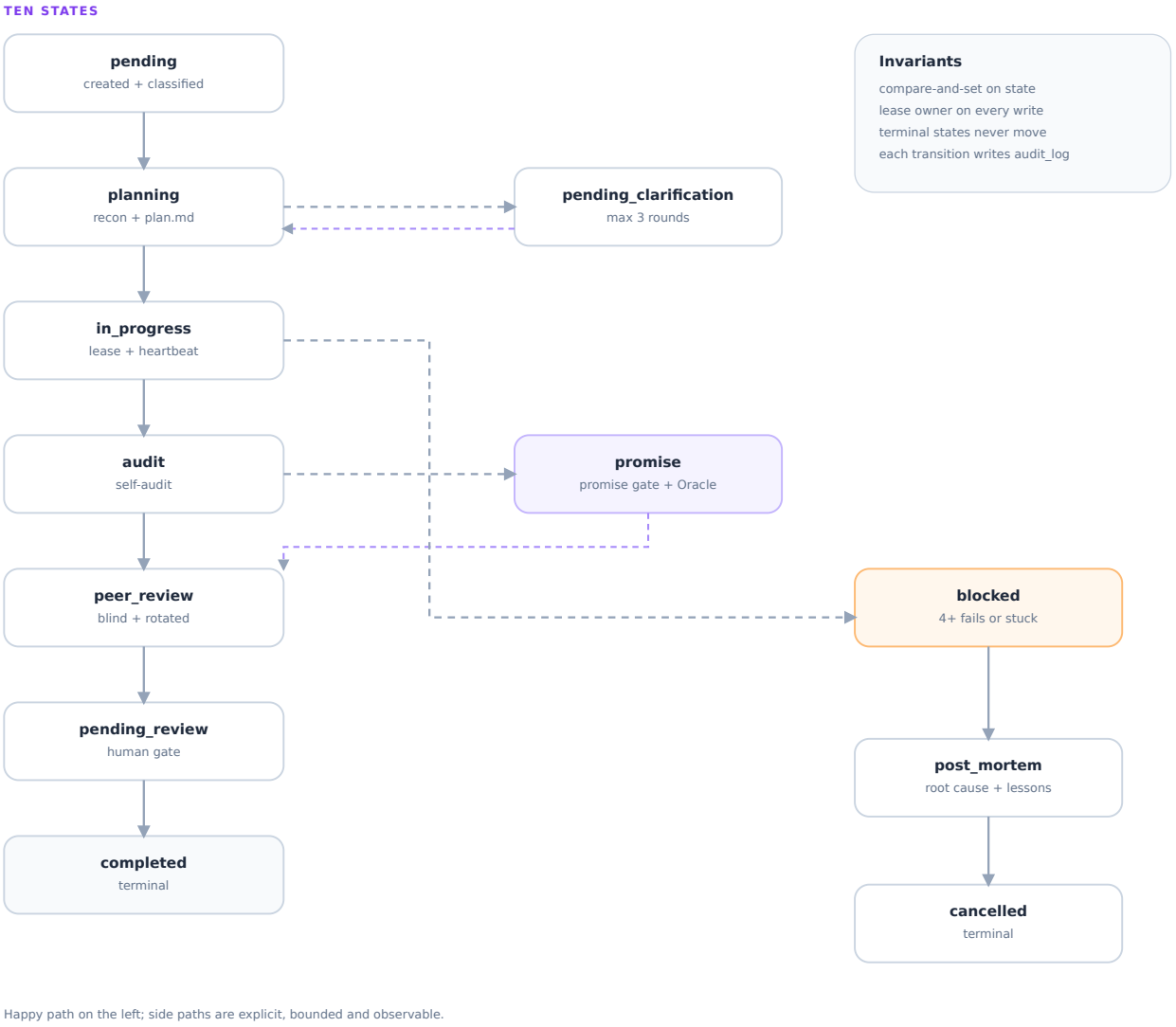
Watchdog & post-mortem

Detect stalls vs waiting, escalate by tier, preserve evidence and extract structured lessons.

State machine

Every transition needs an owner, a guard and an audit event. Terminal states never transition again.

STATE	ENTERED WHEN	MEANING
`pending`	created + classified	Task is valid and has an owner.
`planning`	complex task	Recon and plan are in progress.
`pending_clarification`	gaps found	A specific question is waiting; max 3 rounds.
`in_progress`	plan approved / simple	Agent is executing with a lease.
`audit`	execution complete	Self-audit is running.
`promise`	promise-mode completion	Promise gate and Oracle are running.
`peer_review`	audit passed	Independent peer checks the result.
`pending_review`	peer approved	Human gate is waiting.
`completed`	all gates passed	Terminal; write the score and evals.
`blocked`	stuck / 4+ fails	Needs escalation; never silently retried.



Happy path on the left; side paths are explicit, bounded and observable.

Ten states, guarded transitions. “Complete” is a terminal state, not an optimistic message.

```
transition(task, from, to):
  assert task.state == from
  assert guard(from, to)              # owner, lease, evidence
  update tasks set state = to
    where id = task.id and state = from  # compare-and-set
  append audit_log(task, from, to, actor, evidence_hash)
```

Rejection context and retry

A retry without structured feedback is just another attempt at the same mistake.

FAILURE	ACTION	CONTEXT POLICY
1–2	Same session, continuation prompt with rejection-context.json.	Fix only listed issues.
3	Prune context, switch strategy, fresh session.	Keep task brief, rejection context and new strategy only.
4+	Block and escalate to the human owner.	Dead-letter with full evidence; no silent loop.

rejection-context.json should list: failed criteria, severity, what passed, suggested approach, previous attempt summaries and the evidence hash. The next agent gets this — not the whole failed conversation.

Evidence contract

An Oracle with clean context is only as good as the evidence it is given.

FIELD	REQUIREMENT
Command	The exact command or tool call, with its working directory and environment version.
Output	Exit code, stdout/stderr hashes and the relevant excerpt.
Artifact	Hash, path, size and timestamp of every produced file.
Check	The acceptance criterion it satisfies and who verified it.
Nonce	A unique task/attempt identity so stale evidence cannot be replayed.

```
{
  "task_id": "TSK-042",
  "attempt": 2,
  "nonce": "9f3c…",
  "checks": [
    {"criterion": "service returns 200",
     "command": "curl -fsS https://…/health",
     "exit": 0, "output_hash": "sha256:…",
     "verified_at": "2026-10-26T08:00:00Z"}
  ],
  "artifacts": [{"path": "…", "sha256": "…"}]
}
```

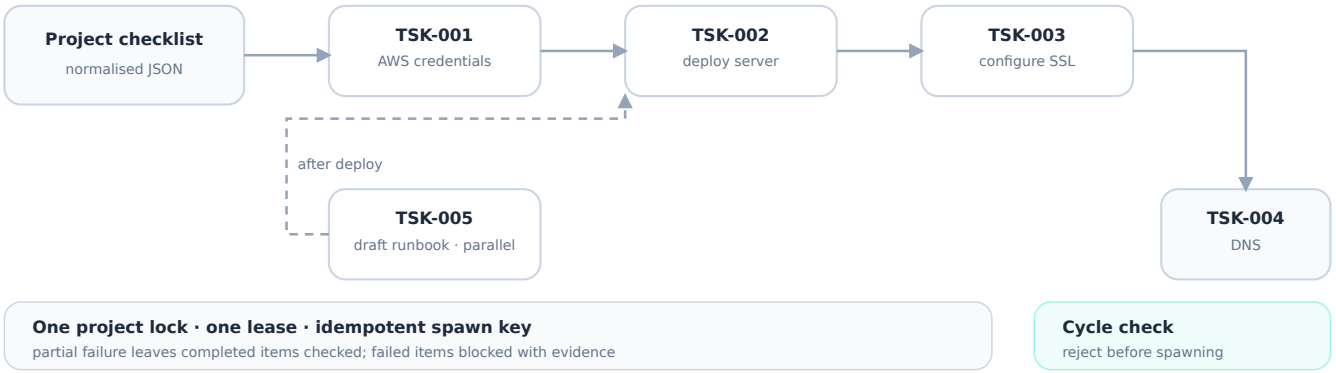
HTTP 200 is not proof. The Oracle must bind the check to this attempt, re-run it where possible, and reject stale artifacts, unverifiable screenshots and self-reported logs.

Human gates by risk

Complexity alone is the wrong trigger for a human.

RISK	EXAMPLES	GATE
Low	Draft, research, formatting	Auto-approve after self-audit plus sample review.
Medium	Publishing, customer-visible content	Peer review; human spot-check.
High	Deploy, spend, delete, credentials	Human approval with evidence and rollback plan.
Critical	Money movement, legal, personal data	Two-person rule; Oracle plus human.

DAG orchestration



Dependencies are data, not a mental model. The orchestrator claims a lock, checks cycles and spawns once.

- **One project, one lock.** The orchestrator claims a project with a lease before spawning.
- **Cycle detection** before execution; a dependency graph with a cycle is a configuration error.
- **Idempotent spawn:** checklist item + project + attempt is the unique key.
- **Workload-aware routing** uses live leases, not stale counters.
- **Partial failure** leaves completed items checked and failed items blocked with evidence.

Watchdog and post-mortem

Watchdog

- Heartbeat from the acting agent; lease expiry, not just “no status change”.
- Distinguish **waiting on a human** from **stuck**.
- Tiered escalation with a deadline at each tier.
- Budget guard: time and token ceilings per task.

Post-mortem

- Evidence first: rejection context, decision trace, score history.
- LLM proposes; the evaluation set decides.
- Lessons are structured: trigger, action, evidence, confidence.
- Only evaluated lessons enter the shared brain as trusted.

What to measure

METRIC	QUESTION
Lead time	Created → completed, p50/p95, split by task type.
First-pass rate	Completed without retry or human reject.
Oracle catch rate	Failures found by the Oracle but missed by self-audit.
Retry cost	Tokens and time spent per successful task.
Human load	Approvals per day and time-to-decision.
Post-mortem action rate	Lessons that changed a strategy and passed evaluation.

Failure modes

FAILURE	CAUSE	FIX
Double completion	Watchdog + agent finish at the same time	Terminal-state guard, idempotency key, compare-and-set.
Context-loop retry	Agent repeats the same failed approach	Structured rejection, strategy switch, fresh session at 3rd failure.
Oracle theatre	Agent supplies its own “evidence”	Evidence contract, nonce, independent re-run, signed tool output.
DAG race	Two orchestrators spawn the same task	Project lock, lease, cycle check, idempotent spawn key.
Peer collusion	Same-pool reviewers approve each other	Blind review, rotation, independent quality sample.
Zombie task	Status stuck with no heartbeat	Heartbeat + lease expiry; distinguish waiting from stuck.