Async Archive-Review Agent - Scoping & Durability

1 - The review loop, run as a fleet (the agent loop, pointed at a whole archive)

```
Archive -> work queue -> N parallel workers, each: perceive -> reason -> act -> observe -> write
Fan-out: 1,000 workers drain the line 1,000x faster.   No clock, no single prompt.
Confident verdicts -> catalog;  low-confidence / severe -> human review queue (escalate by exception).
```

2 - The cost funnel (cheapest tools first; the expensive reader sees only what survives)

```
Cheap filter        shot-change + transcript; drop dead air     (runs on everything)
Keyframe sampler    pick the few frames per shot that matter     (low cost)
Reader (VLM)        describe the sampled frames                  (HIGH cost - keep it last)
Policy / judge      description -> structured verdict + score    (medium)
Catalog / escalate  store the confident; send the rest to people (low)
```

3 - The no-clock economics (nothing is waiting, so optimize cost, not speed)

```
Batch APIs: OpenAI / Anthropic / Gemini all 50% off for a 24-hour turnaround. Real-time can't use them.
Funnel math: 18,000 frames/video -> ~40 sampled = ~450x less reader work; then batch halves what's left.
Spot / preemptible compute: cheap and interruptible - safe ONLY if the job survives interruption (see 4).
```

4 - The durability spine (a crash at item 80,000 resumes at 80,001, never restarts)

```
CHECKPOINT    record each done item before taking the next.
IDEMPOTENCY   key = video-N + model + policy version; skip what's already done (or you double-bill).
RETRY + DLQ   retry transient fails; a poison item -> dead-letter queue -> one human, not the whole job.
ORCHESTRATE   durable engine (e.g. Temporal) replays its event log and resumes mid-job.
```

5 - Scoping checklist (ask these before you build, buy, or wrap)

- [ ] Is there a cheap-filter + sampler funnel, so the expensive reader sees ~tens of frames, not every frame?
- [ ] Are the heavy model calls routed through a batch API (50% off) because nothing is waiting?
- [ ] Does the job checkpoint progress, so a crash resumes at the next item instead of restarting?
- [ ] Does every work unit carry an idempotency key (video + model + policy version) to stop double-processing?
- [ ] Do failures retry, with a dead-letter queue isolating poison items for human inspection?
- [ ] Are verdicts versioned, so re-processing after a model/policy change is partial, not a full re-run?
- [ ] Do only low-confidence and high-severity items go to humans, with a random sample of confident ones spot-checked?
- [ ] Is the reason for each verdict stored (evidence + policy clause), to satisfy moderation-transparency law?