

AI Agent Readiness Audit

Prepared for Sample Open-Source Codebase Inc

roam-code · Cranot

2026-05-05

Contents

Executive Summary	1
1. Repository Overview	3
1.1 Architecture map	3
2. Health Scorecard	4
3. Top Risk Findings	5
4. Dead Code	6
5. Ownership & Bus-Factor	7
6. Test Coverage Gaps	8
7. Suggested CLAUDE.md / AGENTS.md Drop-in	9
8. Suggested CI Gates	10
9. 30 / 60 / 90 Day Fix Roadmap	10
Appendix A — Methodology	12
Appendix B — Reproducing this report	12

SAMPLE. This audit was generated from a real open-source Python codebase (~3,174 files, ~15,603 symbols, single-author contribution history). In a real client engagement, author names and identifying paths are anonymized in the deliverable. The structure, metrics, and rendering quality below match exactly what a paying customer receives.

Executive Summary

This codebase is in **healthy shape overall** (composite score 88/100), with debt concentrated in two predictable hotspots: 27 CRITICAL findings split between *bottlenecks* (15 files that single-thread a disproportionate amount of git churn) and *god-components* (12 modules carrying too much

responsibility). Five files account for most of the recent churn-complexity-fan-in mass; the top one alone carries 2,998 commits.

Recommended 30-day move: address the 5 danger-zone files in Section 3, starting with `output/sarif.py` and `commands/cmd_dead.py`. Their combined refactor is conservatively estimated at **60–80 engineer-hours** and unblocks a cascade of secondary improvements visible in Sections 4 and 5.

Bus-factor risk is HIGH in 53 of 57 directories (Section 5) — a single contributor accounts for >90% of churn in most of the codebase. This is the largest structural risk to the 30-day plan; cross-training should run in parallel with technical work.

Audit verdict: AUDIT — pressures: 5 danger-zone file(s)

Metric	Value
Composite health score	88 / 100
Total estimated debt	2247.4 h
Dead-symbol candidates	380
Danger-zone files	5
Imported test coverage	n/a
Public API surface	1278 symbols
Files / symbols indexed	3174 / 15603

1. Repository Overview

- **Project:** roam-code
- **Files / symbols indexed:** 3174 / 15603
- **Languages (top 5):** yaml, python, markdown, json, html
- **Top dependencies:** src, tests
- **Top-level structure:** rules/ (2492), src/ (346), tests/ (250), benchmarks/ (26), docs/ (21), ./ (15), .github/ (12), dev/ (7)
- **Detected conventions:** functions=snake_case, classes=PascalCase, methods=snake_case

1.1 Architecture map

- **Map verdict:** map: 10 directories, 3174 files, 15603 symbols, deepest: rules
- **Inferred edges (call / import):** 15674
- **Entry points (top 6):** docs/site/app.js, src/roam/__init__.py, src/roam/__main__.py, src/roam/analysis/__init__.py, src/roam/ask/__init__.py, src/roam/attest/__init__.py
- **Top symbols by PageRank:** cli (function), index_in_process (function), git_init (function), cli_runner (function), indexed_project (function)
- **Architectural metrics:** actionable cycles: 1 · tangle ratio: 0.000 · propagation cost: 0.00

2. Health Scorecard

The following findings come from **roam health**. CRITICAL items block ship; WARNINGS are next-quarter work; INFO is hygiene.

Composite health score: 88 / 100 — *Healthy codebase (88/100)* — 27 critical issues, focus: *bottlenecks* **Issue mix:** CRITICAL: 27, WARNING: 10, INFO: 47 · **Issue count:** 66 · **Actionable cycles:** 1

Category severities:

Category	CRITICAL	WARNING	INFO
bottlenecks	15	0	0
cycles	0	1	0
god_components	12	9	29
layer_violations	0	0	0

3. Top Risk Findings

Files ranked by `danger_score` — a composite of churn, complexity, and structural fan-in. These are the architectural pressure points where AI-generated PRs are most likely to introduce regressions.

#	File	Score	Churn	Complexity	Fan-in
1	<code>src/roam/output/sarif.py</code>	1.97	2998	22	16
2	<code>src/roam/commands/cmd_dead.py</code>	1.68	3362	25	8
3	<code>src/roam/languages/generic_lang.py</code>	1.49	2332	33	6
4	<code>src/roam/commands/cmd_adversarial.py</code>	1.30	2122	20	7
5	<code>src/roam/commands/cmd_duplicates.py</code>	1.25	1508	30	6

4. Dead Code

Symbols that no other code references, grouped by verdict (**SAFE** to remove, **REVIEW** manually, **INTENTIONAL** keep).

424 dead export(s): 78 safe, 302 review, 44 intentional

Bucket	Count
SAFE to remove	78
REVIEW manually	302
INTENTIONAL (keep)	44
Test-only	296
Scaffolding	1

Total dead lines of code: 10,877 · **Estimated removal effort:** 346 h · **Auditor-actionable:** 380 (SAFE + REVIEW)

Run `roam dead --detail --json` against the repo for the full per-symbol list (it is omitted here for size). The detail envelope is reproducible from the same git SHA.

5. Ownership & Bus-Factor

Files with concentrated ownership (a single active maintainer or none) — the highest-risk files for an unexpected departure or extended absence.

bus factor 1 (min), 53 high-risk, 57 single-owner modules, top risk: tests/

Team profile: single-author · **Concentrated dirs:** 57 / 57 · **HIGH-risk dirs:** 53 · **Critical-entropy dirs:** 55

Directory	Risk	Bus-factor	Primary author	Share	Stale?
tests/	HIGH	1	CosmoHac	99%	no
src/roam/command	HIGH	1	CosmoHac	100%	no
src/roam/language	HIGH	1	CosmoHac	94%	no
src/roam/index	HIGH	1	CosmoHac	98%	no
src/roam/	HIGH	1	CosmoHac	94%	no
src/roam/catalog	HIGH	1	CosmoHac	100%	no
src/roam/graph	HIGH	1	CosmoHac	100%	no
docs/site/	HIGH	1	CosmoHac	100%	no
rules/community/	Security/	1	CosmoHac	100%	no
./	HIGH	1	CosmoHac	98%	no
rules/community/	Correctness/	1	CosmoHac	100%	no
rules/community/	style/language-pack/	1	CosmoHac	100%	no
src/roam/output	HIGH	1	CosmoHac	100%	no
src/roam/search	HIGH	1	CosmoHac	100%	no
src/roam/rules	HIGH	1	CosmoHac	100%	no

6. Test Coverage Gaps

- **Total test files indexed:** 251
- **Unit / integration / e2e / smoke split:** 0 / 0 / 1 / 1
- **Imported test coverage (heuristic):** n/a
- **Verdict:** MOSTLY-UNSTRUCTURED — 249 of 251 test files have no kind hint (only 2 classified)

7. Suggested CLAUDE.md / AGENTS.md Drop-in

The block below is auto-generated from `roam describe --agent-prompt` and is ready to commit to your repo. It tells AI agents (Claude Code, Cursor, Codex CLI, Gemini CLI, Aider, Continue) how to navigate this codebase using `roam`.

Project: roam-code (3174 files, 15603 symbols, yaml, python, markdown, json, html)

Stack: src, tests

Conventions: functions=snake_case, classes=PascalCase, methods=snake_case

Structure: rules/ (2492), src/ (346), tests/ (250), benchmarks/ (26), docs/ (21), ./ (15), .git

8. Suggested CI Gates

Based on the findings above, the following gates would have caught most of this quarter’s structural regressions before merge:

1. **roam pr-risk --gate 8** — Block PRs that touch more than 8 files or cross more than 3 architecture layers. Pick the threshold at the current p90 of merged PRs so existing velocity isn’t disrupted; tighten as the codebase quality improves.
2. **roam health --gate health_score:85** — Fail CI if a PR drops the composite health score below 85. The current 88 gives a 3-point buffer; tighten to 90 once the 30-day cleanup (Section 9 phase 1) completes.
3. **roam dead --strict --threshold safe:80** — Fail CI if SAFE-bucket dead exports exceed 80 (current: 78). Forces incremental cleanup as new dead code appears rather than letting it accumulate quarter over quarter.
4. **roam clones --persist** as a nightly job — Surfaces clone-class drift (not currently measured). Pair with **roam critique** on PR diff to catch new clones before merge.
5. **Pin roam-code version in CI** — `pip install roam-code==12.25` so health-score deltas are comparable across PRs over time. Bump intentionally on a quarterly cadence.

Each gate maps to a specific finding above; do not adopt all 5 at once — phase them in over the 30/60/90 roadmap (Section 9) to avoid CI thrash.

9. 30 / 60 / 90 Day Fix Roadmap

Days 0–30 — Surface-level wins, no architectural change required.

- Delete the 78 SAFE-bucket dead exports surfaced in Section 4. Pure deletion, no logic changes, immediately visible in PR review velocity.
- Wire up CI gates 1–3 from Section 8 at *current* thresholds (do not tighten yet; the goal in this phase is to stop new regressions, not catch up on backlog).
- Address the top 3 danger-zone files (Section 3) one per week:
 - `output/sarif.py` — split into smaller modules per output format.
 - `commands/cmd_dead.py` — extract bucket-classification logic into its own helper.
 - `languages/generic_lang.py` — replace ad-hoc parsing with the shared LanguageExtractor pattern used by other extractors.

Estimated effort: 60–80 engineer-hours. Recommended owner: 1 senior eng, 2 days/week.

Days 31–60 — Structural refactors that need design review.

- Resolve the single actionable cycle surfaced in Section 2 (`cycles` row in the category-severity table).
- Split the 12 CRITICAL god-components flagged in Section 2’s `god_components` row. These are conversation-required because they affect every contributor’s mental model. Run a 60-90 min architecture-review meeting per split; record the reasoning so the rule can be encoded in Section 8 next phase.
- Begin cross-training plan (Section 5): identify the 3 highest-risk directories and rotate one new contributor per directory per week.

Estimated effort: 100–140 engineer-hours, plus ~5 hours/week of architectural review meetings.

Days 61–90 — Encode learnings as enforceable rules.

- Convert the architectural decisions from Days 31–60 into custom `.roam/rules.yml` entries. Each rule prevents the same drift from reappearing.
- Tighten CI gate thresholds from Section 8: bump `health_score` gate from 85 \rightarrow 90, drop `dead --strict` SAFE threshold from 80 \rightarrow 60.
- Continue cross-training (Section 5) — by Day 90, expect each high-risk directory to have at least 2 active contributors (bus-factor 2).

By Day 90, expect health score above 92, dead-export count below 50, and zero CRITICAL bottleneck findings.

Appendix A — Methodology

This audit was produced with **roam-code** (Apache 2.0, <https://github.com/Cranot/roam-code>). All analysis runs locally on the auditor’s machine; no client code is transmitted to any third-party service.

Pipeline. `roam init` indexes the codebase into a local SQLite graph (symbols, edges, file relationships, git stats). `roam audit --json` then chains `health`, `debt`, `dead`, `test-pyramid`, `api`, `stats`, and `hotspots --danger` into a single envelope. Sections 1, 1.1, 5, and 7 of this report come from `roam describe`, `roam map`, and `roam bus-factor`, run separately.

Determinism. Every metric in this report is reproducible from the same git SHA. There is no LLM-derived content (no probabilistic ranking, no AI summaries) — re-running the pipeline at the same SHA produces byte-identical output.

Roam version: roam, version 12.22

Data handling. Per SOW Section 2, the repo is deleted within 14 days of delivery; access logs available on request.

Appendix B — Reproducing this report

```
# 1. Index the target repo
cd /path/to/target && roam init

# 2. Render auto-content
python templates/audit-report/render.py \
    --client "Sample Open-Source Codebase Inc" \
    --date "2026-05-05" \
    --repo /path/to/target \
    --output audit-report.md

# 3. Fill narrative sections by hand, then convert to PDF
pandoc audit-report.md -o audit-report.pdf \
    --template eisvogel --listings --toc
```