

HorizonSWE: A Pilot of Repository-Level Coding-Agent Evaluation

Ayush Singh · Flamki Labs · Preliminary technical report

Preliminary report. This document records a budget-constrained pilot. It is not peer reviewed, a validated public benchmark release, or evidence of general model superiority.

Abstract

We implement a private evaluation pipeline that combines model inference through Kaggle’s Model Proxy with isolated Linux execution and deterministic patch grading. The planned matrix contains eight model aliases and five repository-level tasks, with one attempt per cell. At report generation, 40 attempts had finished, 8 submissions had test-grading outcomes, and 0 submissions passed both required test groups. Access errors, missing responses, resource limits, and incomplete runs are retained as separate outcomes. These observations support inspection of the pipeline and its failure modes; the sample and execution limits do not support a general model ranking.

1. Motivation and scope

Repository-level evaluation requires interpreting a task in the context of an existing codebase and checking an implementation against executable criteria. SWE-bench established an influential issue-resolution formulation of this problem [1]. This pilot follows the general repository-plus-task pattern but uses a separate private task collection. No equivalence to SWE-bench scores or other commercial benchmarks is claimed.

2. Task preparation

The pilot contains five Python-focused tasks across three upstream repositories, selected before model results from a larger private pool. Task themes are tool registry lifecycle, bundle installation, configuration overlays, event inspection, and per-step runtime configuration. The task selection is purposive rather than random. Each task has a pinned repository snapshot, prompt, reference patch, task-specific tests, and containerized validation environment. Reference validation was completed locally before model runs: 5/5 reference-patch controls passed, while 5/5 comment-only controls failed. These controls test the grading pipeline; they

are not a human performance baseline or proof of exhaustive coverage. This is internal qualification; independent task-authoring and publication-readiness review remain outstanding.

Only pristine upstream source and the task prompt are supplied to the model. Reference patches, hidden test additions, grader logs, and evaluator containers are withheld. Source archives are hash checked before extraction. Archive traversal, symlinks, and special files are rejected by the handoff loader.

3. Agent protocol

The frozen protocol is `retrieval-pilot-v4`. Models receive a shared tool set for listing directories, searching source, reading source, replacing a unique substring, creating files, running commands, and finishing an attempt. Native tool calls or explicit JSON actions are used according to endpoint compatibility. Persistent changes are restricted to approved Python source and documentation paths. Command execution takes place in fresh offline containers containing the pristine source and current edits; command-side filesystem changes are discarded.

Each attempt permits up to 40 turns and 16,384 output tokens per request, including provider-charged reasoning tokens. Native tools request low reasoning effort except GPT nano, which uses none because its endpoint rejected the low setting with tools. DeepSeek and Granite use a single JSON action per response with provider-default reasoning after native endpoint errors. These declared interface differences limit comparability. Temperature remains at provider defaults. The hard content budget is 96,000 characters, excluding opaque transport IDs. Above 80,000 characters or 26 messages, complete oldest tool exchanges are removed while preserving the task and recent context. Directory listings are shallow, search results are bounded, and reads default to 80 lines with an 8,000-character output cap. Each command is limited to 60 seconds, each request to 180 seconds, and generation has a 20-minute loop-level limit. Budget checks can stop an attempt before these limits.

Containers have no network, host-directory mounts, or model credentials, and have CPU, memory, process and capability restrictions. Grading occurs separately after generation. There is no hidden-test feedback loop. All earlier smoke and v1-v3 diagnostic runs are excluded from this matrix. The retrieval revision was made after diagnostic runs reached context limits before useful edits; those results were not selected for headline scores.

4. Models, access and cost

The evaluated identifiers are the aliases exposed by Kaggle's authenticated local development proxy, not its entire hosted model catalog [2,3]. Kaggle auth issued an endpoint whose hostname includes a staging designation. The exact downstream provider identity and alias revision therefore need independent confirmation before public capability claims. A daily allowance of

\$10 equivalent governs inference; the daily spend reported in the companion dashboard includes earlier debugging runs. Per-response usage metadata is preserved privately and aggregate cost fields are exported when available.

5. Grading and outcomes

A patch must satisfy path restrictions and apply to a fresh evaluator. The frozen grading policy requires both the existing-test group and task-specific-test group to pass. Exact case identity and approved skip policy are checked. A passing test suite demonstrates only the tested behavior, not exhaustive correctness. Empty patches, execution errors, and budget-limited attempts remain visible. Every nonempty patch remaining at an inference-credit cutoff receives the same local grading after the batch; its budget-limited generation status is retained in the data. This post-grading rule was established before any final-protocol attempt reached the credit cutoff.

Model alias	Pass / graded	No patch	Error / unresolved	Missing / budget
anthropic/claude-sonnet-5@default	0/1	3	0	1
google/gemini-3-flash-preview	0/1	3	0	1
openai/gpt-oss-120b	0/0	0	5	0
qwen/qwen3-next-80b-a3b-instruct	0/0	1	3	1
deepseek-ai/deepseek-r1-0528	0/1	1	2	1
openai/gpt-5.4-nano-2026-03-17	0/1	3	0	1
google/gemini-3.1-flash-lite-preview	0/4	0	0	1
ibm/granite-4.0-h-small	0/0	3	2	0

These are counts, not a ranked leaderboard. Missing or unresolved runs must not be silently converted to zero-scoring model failures or omitted when comparing coverage. The complete cell-level results are supplied in `results.csv` and `results.json`.

5.1 Additional catalog checks

A separate remaining-budget exploratory extension performs small access/cost probes before attempting FL-006 with additional catalog models. This compact protocol permits 20 turns and 2,048 output tokens per request, so its outcomes are not comparable to the core protocol. Probes are not coding evaluations. Full attempts require the observed effective probe token rates to remain within the core cost guard with twofold headroom; unverified pricing is skipped. At report generation, 36 of 36 candidate models had access checks, 2 had coding attempts, and 0

submissions were graded. State: `catalog_processed`. These one-task checks are not pooled with the core matrix; details are in `catalog-results.csv`.

6. Threats to validity

The five-task sample is small, selected, and limited to three repositories. The same tasks informed harness debugging; no held-out harness-development split was used. These outcomes describe models running within this specific scaffold and resource budget. A single attempt per cell cannot characterize stochastic variability. Different provider implementations of reasoning settings, alias updates, output caps, and transport behavior can affect results. Public upstream code may have appeared in training data; contamination has not been ruled out. Hidden tests can be incomplete or overconstrained, and no independent full-solution validation is claimed. Execution restrictions may disadvantage solutions requiring unavailable services. Manual review of successful patches and broader semantic review are required before publication.

7. Reproducibility and data availability

Internal records include source, harness and patch hashes, model identifiers, container image IDs, tool traces, usage metadata, generation timing and test outputs. The agent source checksum is `4311b3830f28e6f81a1fd1fff589a9fe7a943e203529d4c5619f6e7af1ceab1b`. Model aliases and private task access limit independent replay. Credential renewal was repaired between attempts without changing model inputs, tool behavior, limits, or grading; the affected attempt list and before/after adapter hashes are recorded in the JSON export. A provider reasoning-prefix parsing issue was also normalized after an ungraded DeepSeek diagnostic; that diagnostic was excluded and rerun, with the adapter revision recorded. This review package intentionally excludes prompts, test content, reference solutions, generated patches, credentials and repository snapshots.

8. Conclusion

This pilot provides an auditable implementation and a record of observed outcomes under a fixed constrained protocol. It does not yet establish a validated long-horizon benchmark or a reliable model ranking. The next research step is independent task review, verified model revisions, repeated runs, and a larger, deliberately sampled task set.

References

[1] Jimenez et al. [SWE-bench: Can Language Models Resolve Real-World GitHub Issues?](#) 2023/ICLR 2024.

[2] Kaggle. [Kaggle Benchmarks quick start.](#)

[3] Kaggle. [Benchmark CLI and Model Proxy documentation.](#)