REINFORCING AGENTS WITH COLLECTIVE SKILLS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce SKILL2ENV, a scalable and aligned data recipe for modern agentic RL. Public *Agent Skills* provide workflow guidance across diverse domains, linking to real-world artifacts and packaging domain expertise and quality criteria. We develop a pipeline to turn 3.4k web-crawled and filtered *Agent Skills* into 8k executable terminal environments with both programmatic tests and behavioral rubrics. Through extensive RL experiments, we observe promising improvements in general agentic abilities. With 150 steps of RL training using a FlashREINFORCE variant, Qwen3.8-27B gains 9.0 percentage points on Terminal-Bench 2.1, and 7.1 percentage points on S2EBENCH, our hand-verified private benchmark featuring real-world agent use cases. Further analysis indicates increased behavioral alignment with original *Agent Skills* on similar problems.

1 INTRODUCTION

Reinforcement learning (RL) in executable environments with verifiable rewards improves terminal-use agents (Merrill et al., 2026; Ivison et al., 2026; Gandhi et al., 2026), but scaling it requires diverse training tasks. Existing pipelines mine or synthesize tasks from repositories (Pan et al., 2024; Jain et al., 2025; Yang et al., 2025) or hand-designed taxonomies. Their coverage depends on these seeds and often emphasizes technical workflows (Figure 3), whereas users also delegate literature reviews, marketing plans, slide decks, and financial models. Moreover, outcome tests measure completion without necessarily rewarding a sound process: a passing result need not reflect careful research or evidence-based reasoning.

Public *Agent Skills* offer both task diversity and procedural supervision. Each Skill packages a `SKILL.md` entry document with optional scripts, references, and assets describing a practitioner-authored workflow. Across software, research, design, and business, these bundles provide (i) tasks

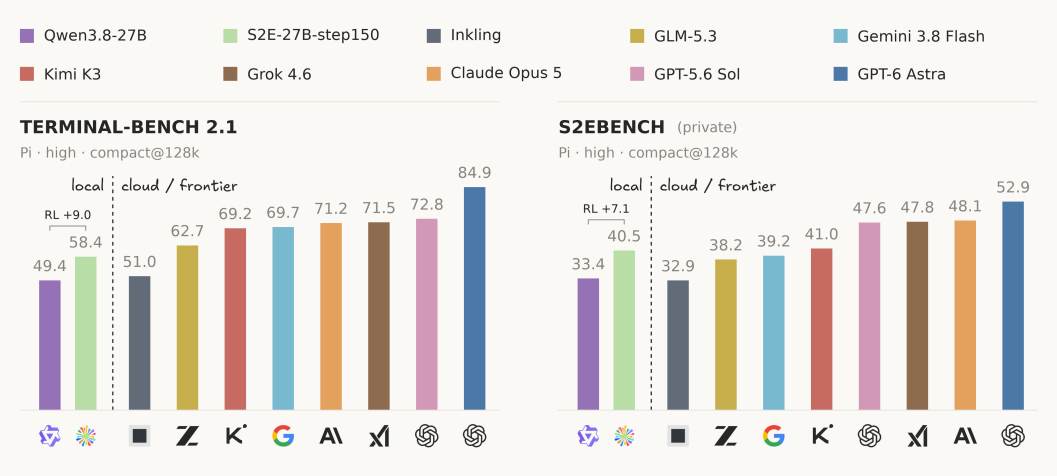

Figure 1: **Main results.** Skill2Env improves the base model on both public and private benchmarks for agentic terminal use, further reducing the gap between local and cloud LLMs on a distribution shaped by public interests.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

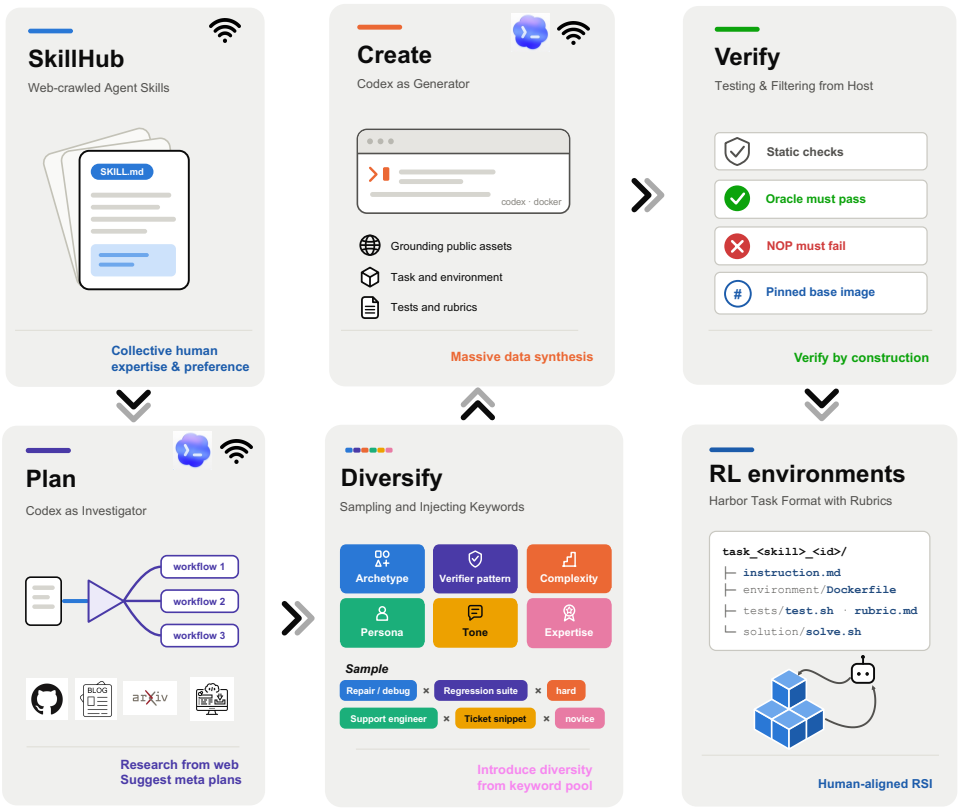

Figure 2: **SKILL2ENV data pipeline.** Public Agent Skills are crawled and filtered into SKILLHUB. A Codex planner reads each Skill bundle, researches supporting public assets, and decomposes the Skill into distinct workflows. Each workflow is diversified by sampled task axes and handed to a Codex creator, which builds the initial world, instruction, deterministic tests, behavioral rubric, and reference solution. The host accepts a task only if it passes Oracle, NOP, and static checks. The result is a Harbor task with an attached rubric, ready for RL.

people want agents to perform, (ii) pointers to real artifacts and tools, and (iii) domain expertise and quality criteria.

SKILL2ENV (Figure 2) compiles each Skill's workflows into verified Harbor tasks (Harbor Contributors, 2026) and preserves its methodology as a per-task behavioral rubric. Applied to SKILLHUB, about 3.4k filtered public Skills, it produces 7,971 tasks across thirteen domains, under a quarter of them software engineering.

We validate the data by training Qwen3.8-27B with DPPO (Qi et al., 2026) and FlashREINFORCE (Hu et al., 2026b) recipes on the Pi harness and evaluating on Terminal-Bench 2.1 and our hand-verified S2EBENCH. Our contributions are:

- **SKILL2ENV**, a reproducible pipeline that turns Agent Skills into verified RL environments with both programmatic tests and behavioral rubrics.
- **SKILLHUB**, a collection of license-friendly (MIT or Apache 2.0) web-crawled Agent Skills.
- **An 8k open Agentic RL dataset**, spanning diverse domains aligned with public interests.
- **RL experiments and findings.** RL over SKILL2ENV improves Qwen3.8-27B on S2EBENCH and transfers to Terminal-Bench 2.1; Skill-derived rubrics further align its behavior with the source Skills (Section 5).

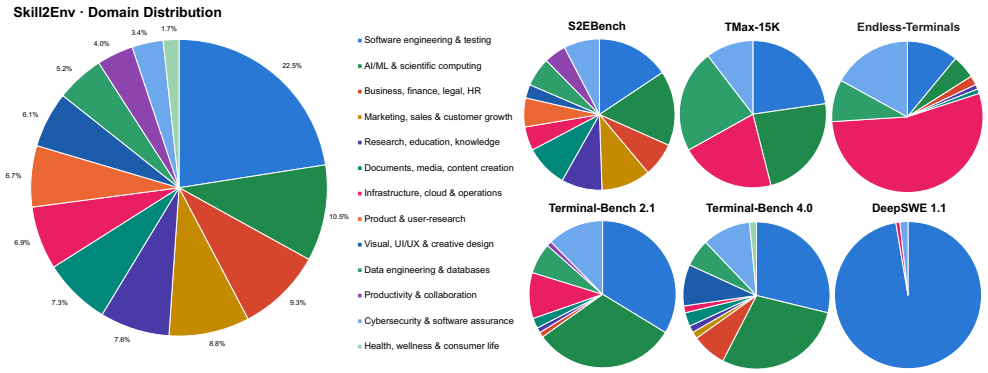

Figure 3: **Domain diversity of Skill2Env.** GPT-5.6 Sol labels datasets and benchmarks with one of 13 domains. Skill2Env is the most diverse and the only dataset covering all 13 domains, representing a distribution closest to public agent use cases.

2 Related Work

Environments for terminal agents. Repository-based corpora such as SWE-Gym, R2E-Gym, SWE-smith, SWE-rebench, SWE-Universe, and SERA derive tasks from existing codebases (Pan et al., 2024; Jain et al., 2025; Yang et al., 2025; Badertdinov et al., 2025; Chen et al., 2026; Shen et al., 2026a). Terminal synthesis approaches instead use taxonomies or task seeds and vary their world construction and verification procedures (Gandhi et al., 2026; Ivison et al., 2026; Zhu et al., 2026; Pi et al., 2026; Lin et al., 2026; Wu et al., 2026a; Shen et al., 2026c; Wu et al., 2026b); Agent World Model and Agent-World extend synthesis beyond terminals (Wang et al., 2026; Dong et al., 2026). SKILL2ENV instead draws both workflows and quality criteria from practitioner-authored Skills.

Agent Skills. Introduced by Anthropic in 2025, Agent Skills became an open specification supported by Claude Code and Codex (Anthropic, 2025a; Agent Skills Contributors, 2026; Anthropic, 2025b; OpenAI, 2026). Studies examine ecosystem security and downstream adaptation (Liu et al., 2026; Wu et al., 2026c; Xu & Yan, 2026); SkillsBench evaluates inference-time utility, and Skill-α trains Skill generators (Li et al., 2026; Shen et al., 2026b). Concurrently, SkillSynth composes public Skills into 3,560 tasks, while Terminal-World generates 5,723 environments with teacher trajectories (Fan et al., 2026; Cheng et al., 2026b). Both study multi-Skill composition and SFT. SKILL2ENV instead compiles each Skill's workflows, preserves its methodology in a per-task rubric, applies Oracle/NOP checks, and validates the resulting data with RL.

RL recipes and rubric rewards. Agentic RL has been studied with PPO, DPPO, and bespoke optimizers (Gandhi et al., 2026; Ivison et al., 2026; Shen et al., 2026c; Raoof et al., 2026; Luo et al., 2025; Wang et al., 2025), including transfer to SWE-bench (Jimenez et al., 2023; OpenAI, 2024). Our recipes (Section 4.2) build on GRPO (Shao et al., 2024), DPPO (Qi et al., 2026), Dr. GRPO (Liu et al., 2025), and FlashREINFORCE (Hu et al., 2026b), without an SFT warm start. Prior rubric rewards address safety, medicine, science, and instruction following (Mu et al., 2024; Gunjal et al., 2025; Viswanathan et al., 2025; Huang et al., 2025); River adds turn-level behavioral signals for terminal agents (Yao et al., 2026). Cross-domain post-training can also change agent behavior (Ritchie et al., 2026; Cheng et al., 2026a). Our Skill-derived rubrics calibrate executable outcome rewards. Because judges are biased measurements (Zheng et al., 2023), we analyze their score distribution and interaction with outcome rewards rather than treating judgments as ground truth.

3 Data Generation

3.1 Collecting Public Agent Skills

An Agent Skill combines `SKILL.md` metadata and instructions with optional scripts, references, and assets. We crawled over 5k Skills from public GitHub repositories and applied safety and licensing

filters. We exclude workflows requiring external credentials, paid services, physical hardware, or real-world side effects; non-executable persona, style, or copywriting prompts; and harmful or policy-risky activities. N-gram coverage over `SKILL.md` removes near-duplicates, leaving about 3.4k Skills in SKILLHUB.

3.2 SKILL2ENV PIPELINE

SKILL2ENV compiles one Skill into several Harbor tasks in four stages (Figure 2): a planner decomposes the Skill into workflows, the host samples task axes for each workflow, a creator builds the task, and the host accepts or rejects it. The planner and creator are Codex CLI agents running inside Docker containers with outbound network access, so they can read documentation, clone repositories, and download public data while authoring.

3.2.1 WORKFLOW DECOMPOSITION AND DIVERSIFICATION

The planner reads the complete Skill bundle and writes `workflows.json`, ranking its distinct workflows by final-state verifiability, solvability, and intrinsic difficulty. A literature-review Skill, for example, may yield screening-and-synthesis and citation-reconciliation workflows.

For each workflow, it supplies a *meta plan* (scenario, initial world, defects, difficulty, solution sketch, and verifier strategy), *asset recommendations* (pinned public repositories, datasets, specifications, documents, or bundled files), and an *axis pool* of two to four ranked archetype/verifier/persona combinations. The host samples a combination plus complexity, tone, and requester expertise; persona and tone conditioning follow Ge et al. (2024). Workflow-conditioned pools keep diversification relevant, pairing research with evidence-traceability checks rather than arbitrary axes.

3.2.2 TASK CREATION AND VERIFICATION

A fresh Codex creator receives the Skill bundle, meta plan, asset recommendations, and sampled axes. It grounds tasks in license-compatible repositories at pre-change commits, versioned documents or datasets, and official specifications or SDKs. Live services are replaced with deterministic localhost stubs, replay fixtures, fake CLIs, or seeded databases that preserve observable contracts and error cases. Tasks may mix real assets with synthetic fixtures but require no network at solve time.

The creator builds `environment/`, writes the instruction, implements tests, writes the rubric, and only then writes the reference solution. Tests and rubric are frozen before the solution exists: a failing solution must be repaired without weakening the grading contract.

Programmatic tests. `tests/test.sh` is the authoritative outcome reward. It runs after the episode against the final container state and writes a `reward.json` with up to six named metrics in $[0, 1]$, each an independent assertion group covering a distinct facet of the workflow. The training runtime averages the metrics into a scalar $r_V \in [0, 1]$, so most tasks carry partial credit rather than a binary outcome.

Behavioral rubric. Alongside the tests, the creator writes `tests/rubric.md` with bullet points covering *Must-do*, *Must-avoid*, and *Best-practice*, distilling the source Skill's own methodology from different angles. The rubric may not introduce hidden hard requirements; it separates verifiable correctness from procedural quality, and Section 4.2 describes how the two are combined during RL.

We deliberately do not add a validation pass in which a teacher LLM attempts each task and discards those it fails to solve. Such a pass caps task difficulty at the validator's ability and roughly doubles generation cost; with online dynamic filtering (Yu et al., 2025), group-based RL discards prompts whose rollouts carry no advantage, which filters unsolvable or trivial tasks online, as defined by the on-policy distribution.

3.2.3 HOST-SIDE ACCEPTANCE

A model-free host gate writes `task.toml` and metadata, validates the layout, and requires symlinks to resolve within their subtrees. It rejects security leakage and Dockerfiles that reach outside

`environment/` or copy privileged instructions, rubrics, tests, or solutions. Base images must use approved public registries; each `FROM` is pinned to a content digest.

The host then builds the image and runs two Harbor trials in fresh containers: the reference solution (Oracle) must earn the full reward on every metric, and a no-operation agent (NOP) must earn zero on every metric. Candidates that fail either trial are rejected.

3.2.4 TASK FORMAT

Accepted tasks use the standard Harbor layout with an additional rubric:

```
task_<skill>_<id>/
  instruction.md            # prompt / request
  task.toml                 # config and axis metadata
  environment/Dockerfile    # build context = initial world
  tests/test.sh             # writes /logs/verifier/reward.json
  tests/rubric.md           # Must-do / Must-avoid / Best-practice
  solution/solve.sh         # reference solution (Oracle)
```

Appendix A walks through a complete task next to the Skill it was compiled from.

3.3 DATA DISTRIBUTION AND ANALYSIS

Generating the 7,971 tasks with GPT-5.6 Sol at xhigh reasoning effort cost over $90k in API usage.

Domain Coverage. Besides being the only dataset in Figure 3 that covers all 13 domains, SKILL2ENV is the only one in which non-technical knowledge work (business, marketing, research, documents, product, design, productivity, health) forms a large share.

Affinity to Source Skill. A retrieval probe tests fidelity to the source Skills: querying the 3.4k original `SKILL.md` documents with each task's instruction and rubric under TF-IDF cosine similarity, the true source Skill is the top-1 match for 73.2% of tasks and within the top 10 for 94.6% (chance is 0.03%). The rubric alone retrieves its Skill at 68.5% top-1, so the rubric carries Skill-specific methodology rather than generic advice. Appendix A shows a representative case in which the Skill's default flags become the task's must-avoid clauses and the Skill's scripts become the tools in the task's world.

3.4 SFT DATA ROLLOUT WITH GLM-5.3

We rolled out every task twice with GLM-5.3 (Z.ai, 2026; GLM-5 Team et al., 2026) at high reasoning effort on the Pi harness (Earendil Works, 2026) through the Polar framework, serving the model through vLLM with a 128k context and 32k tokens per turn. The archive holds 15,968 trajectories with a mean reward of 0.74. The median trajectory makes 19 model calls and 23 tool calls. We use the archive to characterize task difficulty for a frontier open-weight model and as teacher data for the SFT ablation in Section 5.4.

3.5 S2EBENCH

As public benchmarks saturate, we generated a separate held-out set from SKILLHUB Skills and reviewed every task by hand, dropping any whose instruction is ambiguous, that is unfaithful to its Skill or not meaningfully difficult, or whose tests are not well-rounded and unbiased. The resulting S2EBENCH has 79 tasks (Figure 3).

4 RL EXPERIMENTS

We study four questions: whether RL over SKILL2ENV improves general agentic abilities (Section 5.1); whether Skill-derived rubrics, applied through an LLM judge, help beyond programmatic verification (Section 5.2); whether RL aligns model behavior with the source Skills (Section 5.3);

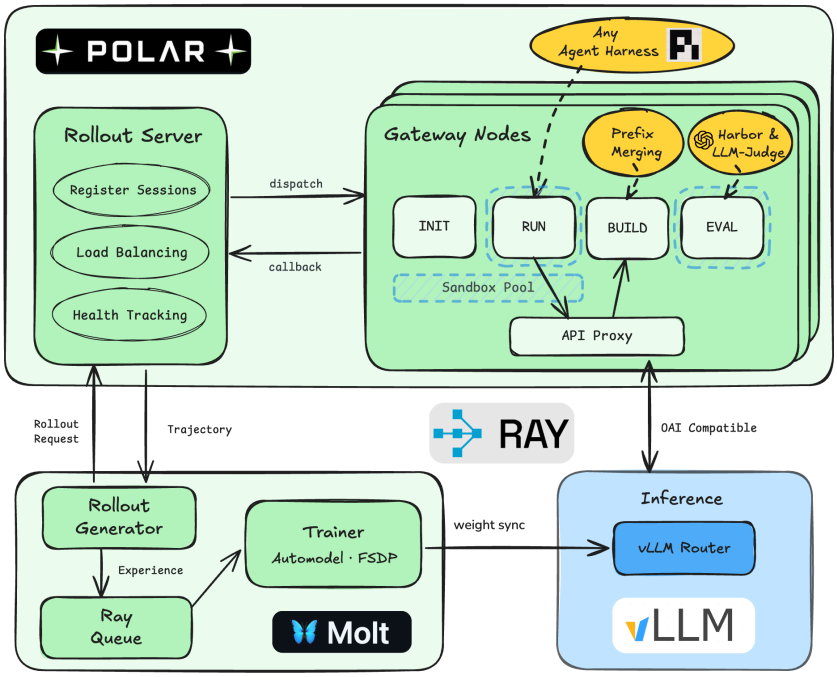

Figure 4: **Fully asynchronous RL infrastructure for sandboxed agent RL on any harness.** Molt owns training loops: Ray, vLLM router, sampling, and optimization steps. Polar handles efficient rollouts through the harness: sandbox management, proxy-based token-in-token-out, trajectory construction by prefix merging, and evaluation with Harbor-like outcome reward and LLM-as-judge. Finished trajectories flow back through a Ray queue into the trainer; weight updates are broadcast to vLLM between batches.

and whether SFT on teacher trajectories helps as a cold start (Section 5.4). The base model is Qwen3.8-27B (Qwen Team, 2026).

4.1 Training Infrastructure

Agentic RL over sandboxed harnesses needs exact log-probability transport between sampler and trainer, since small discrepancies accumulate and destabilize long runs. We combine Molt (Hu et al., 2026a), a PyTorch-native, fully asynchronous RL trainer, with Polar (Xu et al., 2026), which emits token-exact trainable trajectories from any harness running in sandboxes (Figure 4).

A single Ray job (Moritz et al., 2018) launches both: Molt serves the policy with vLLM (Kwon et al., 2023) behind a session-affine router and trains it with FSDP2 workers built on NeMo Automodel (NVIDIA, 2026), while Polar runs each rollout as an unmodified Pi agent inside a rootless Apptainer container and scores the final container state with the task's `test.sh`. Two mechanisms make the off-the-shelf harness trainable: an API proxy that returns the sampled token IDs and log-probabilities for every chat completion and fails the rollout if either is missing, and *prefix merging* (Xu et al., 2026), which stitches a harness's successive completions into training traces, respecting in-harness behaviors like compaction, retries, and subagents. Remaining settings are in Table 4.

4.2 RL Recipe

We compare two recipes that share the policy, harness, task subset, reward, and learning rate: GRPO advantages under DPPO's token-level binary-KL mask (Qi et al., 2026), and Dr. GRPO advantages (Liu et al., 2025) under FlashREINFORCE's trace-level test (Hu et al., 2026b). Both take the sampling-time probabilities recorded by vLLM, and neither uses a KL penalty or entropy bonus. Both recipes comsume the same batch of 2.4k random training samples for 1 epoch.

DPPO. The objective is $L = -\big(\sum_{\text{unmasked}} \rho_i A_i\big)/N$, where $A_i = (r_i - \bar{r})/(\sigma_r + \epsilon)$ is the group-normalized advantage of rollout i, $\rho = q/p$ is the ratio between the current probability q and the sampling-time probability p of an action token, and N is the total number of action tokens in the batch. A token is dropped from the sum whenever its binary KL divergence

$$D_{\text{bin}}(p \,\|\, q) = p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q} \tag{1}$$

exceeds a threshold δ and the update would push ρ further in the direction the advantage favors ($A > 0, \rho > 1$ or $A < 0, \rho < 1$), which also guards against training–inference mismatch. We use $G = 8$ rollouts per group and 8 prompts per update; and $\delta = 0.05$, split a rollout's advantage across its merged training traces in proportion to action tokens when an episode is segmented, and take one optimizer update per batch of 64 rollouts. Groups with zero reward variance yield zero advantages; timed-out rollouts are excluded from the policy loss.

FlashREINFORCE. This recipe changes both components. The advantage follows Dr. GRPO, $A_i = r_i - \bar{r}$, subtracting the group mean without dividing by σ_r. The loss follows FlashREINFORCE, which moves the binary-KL test from the token to the whole trace and averages within each trace before averaging across the batch:

$$L = -\frac{1}{B} \sum_{i=1}^{B} \frac{m_i A_i}{T_i} \sum_{t=1}^{T_i} \rho_{i,t}, \qquad m_i = \mathbf{1}\Big[\frac{1}{T_i} \sum_{t=1}^{T_i} D_{\text{bin}}(p_{i,t} \,\|\, q_{i,t}) \le \delta\Big], \tag{2}$$

where T_i is the number of action tokens in trace i and B the number of traces in the batch. A trace whose mean binary KL exceeds δ is dropped whole at sequence level, and the per-trace average keeps long failed rollouts from outweighing short ones, as they do under normalization by the batch token count N. We use $G = 16$, 16 prompts per update, $\delta = 10^{-2}$.

Rubric-calibrated reward. In the rubric condition, run with the DPPO recipe, the Polar evaluator additionally uses an LLM judge, providing it with trace-level context, a fixed meta rubric (penalizing aimless loops, reward hacking, and unresponsive answers), and the per-task `rubric.md`. The judge returns an integer score $s \in [-5, 5]$ per trace. The training reward is simply

$$r = r_V + \lambda \frac{s}{5}, \qquad \lambda = 0.2, \tag{3}$$

so the judge moves the programmatic reward $r_V \in [0, 1]$ by at most ± 0.2. We use GPT-6 Astra with xHigh reasoning as the Judge LLM. The full judge prompt template is in Appendix B.

4.3 Evaluation Setup

All evaluations run through Harbor@0.22.0 with the native Pi agent and default configuration. We evaluate Terminal-Bench 2.1 (Terminal-Bench Contributors, 2026) and S2EBench and report pass@1 across 5 trials. Exact commands are in Appendix D for reproduction.

5 Results and Findings

5.1 RL over Skill2Env improves agentic abilities

Table 1 reports held-out performance of outcome-only RL on Qwen3.8-27B under both recipes of Section 4.2: the DPPO checkpoint after 300 updates and the FlashREINFORCE checkpoint after 150 updates. Both improve on every metric. On S2EBench, pass@1 rises by 4.3 and 7.1 points (all-tests-pass) and mean score by 18.5 and 21.9 points, respectively. The gain transfers to Terminal-Bench 2.1 by 4.7 and 9.0 points (sharing no task above 13-gram Jaccard similarity of 0.8 with the training corpus), moving the 27B model closer to cloud or frontier models of much greater size; on S2EBench the FlashREINFORCE checkpoint passes GLM-5.3 and Gemini 3.8 Flash (Figure 1). We read the transfer as a general gain in how the model plans, uses tools, and finishes rather than memorization of a task family.

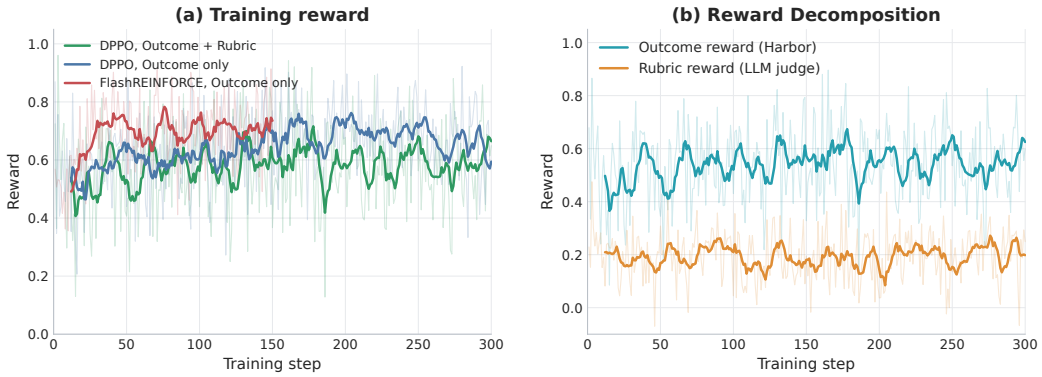

Figure 5: **Reward and its decomposition.** (a) Programmatic Harbor outcome reward vs. additional rubric calibration with LLM-as-Judge, both trained with DPPO; and the outcome-only FlashREINFORCE run with double batch size over the same data batch. (b) Reward decomposition of the rubric-calibrated run. Vanilla rubric calibration appears to slow the growth of the programmatic outcome reward.

Table 1: **RL over SKILL2ENV on Qwen3.8-27B.** Pass@1 in percent over five trials per task with the Pi harness; S2EBENCH also lists the mean score. Outcome RL uses the programmatic reward r_V only; Rubric RL uses the calibrated reward of Equation (3).

Qwen3.8-27B	TB 2.1	S2EBENCH pass	S2EBENCH score
Base	49.4	33.4	56.6
Outcome RL (DPPO)	54.1	37.7	75.1
Outcome + Rubric RL (DPPO)	50.1	34.7	63.6
Outcome RL (FlashREINFORCE)	**58.4**	**40.5**	**78.5**

5.2 RUBRIC-CALIBRATED RL TRAILS OUTCOME-ONLY RL ON BENCHMARKS

The rubric run differs from the DPPO outcome-only run only in the reward, using Equation (3) in place of r_V. Figure 5(a) compares the total training reward of the two runs, and Figure 5(b) decomposes the rubric run's reward into its programmatic and judge components.

Adding the judge does not destabilize training (DPPO mask rate 0.06%, in the range of the outcome-only run), but outcome reward grows more slowly. The rubric run's programmatic reward sits between 0.5 and 0.6 for most of training, below the outcome-only run, while its judge term stays flat. One intuitive reason is that the policy trades some test passes for behavior the rubric rewards. The held-out numbers in Table 1 follow the same pattern: the rubric checkpoint improves over the base model on all three metrics but less relative to the outcome-only checkpoint.

This is not a verdict on rubric rewards: the runs optimize different objectives, and the benchmarks score only the outcome half. The additive calibration with a single $\lambda = 0.2$ is untuned, and the methodology Skills prescribe need not maximize benchmark pass rates. The judge is discriminative rather than uniformly positive, assigning -2 or lower to 21% of traces and -5 to evidenced reward hacking (Appendix B). What the rubric buys is measured in Section 5.3.

5.3 SKILL2ENV ALIGNS MODEL BEHAVIOR TO AGENT SKILLS

Benchmarks score whether a task was finished; Skills also prescribe how. We test whether RL teaches the latter with a pairwise preference test on 200 SKILL2ENV tasks disjoint from training. For each task we roll out one trajectory from the base model and from each DPPO checkpoint (outcome-only and outcome-plus-rubric). The judge of Section 4.2 receives the source `SKILL.md` and two anonymized trajectories in random A/B order, and picks the one that follows the Skill more closely or rates them *equally aligned*.

Table 2: **Skill-alignment preference over 200 paired trajectories.** Percentage of tasks for which the judge, given the source `SKILL.md` as standard practice, prefers the compared checkpoints or rates them as equally aligned. The A/B order is randomized.

Comparison	Prefers base	Prefers RL	Equally aligned
Base vs. Outcome RL	33.5	54.5	12.0
Base vs. Outcome + Rubric RL	24.0	73.0	3.0

Table 3: **SFT on GLM-5.3 trajectories.** Pass@1 over five trials per task with the Pi harness. SFT checkpoints are trained on the assistant tokens of the GLM-5.3 trajectories from Section 3.4.

Model	Training	TB 2.1	S2EBENCH pass	S2EBENCH score
Qwen3.8-27B	Base	**49.4**	33.4	56.6
	GLM-5.3 SFT	45.8	**33.9**	**59.1**

Outcome-only RL already moves the model toward the Skills: the judge prefers the RL trajectory on 54.5% of tasks against 33.5% for the base model (Table 2). The rubric checkpoint widens the margin to 73.0% against 24.0%, suggesting that rubric calibration improves adherence to source Skills. We leave the best reward-mixing strategy, especially for less verifiable tasks such as visual design, report synthesis, and open research, to future work.

5.4 SFT FROM GLM-5.3 DOES NOT IMPROVE QWEN3.8-27B

We fine-tuned Qwen3.8-27B on the GLM-5.3 trajectories of Section 3.4, training on assistant tokens only with tool observations and user turns masked, at a 128k sequence length. Table 3 reports the result. TMax (Ivison et al., 2026) reports that SFT mixtures degrade the already post-trained Qwen3.5, and we see the same pattern with a stronger teacher: SFT slightly raised S2EBENCH but lowered Terminal-Bench 2.1.

We suspect that the teacher's interleaved-thinking and tool-call style differs from the student's own post-training, and imitating it overwrites patterns the student relies on without transferring the teacher's competence. We did not pursue SFT further as a cold start; the RL results above start from the unmodified checkpoint.

6 CONCLUSION

SKILL2ENV turns public Agent Skills into executable RL environments that keep what a taxonomy lacks: the tasks people actually want automated and the practitioner's criteria for doing them well. RL experiments from the 13-domain corpus improves Qwen3.8-27B by up to 7.1 points on S2EBENCH and 9.0 points on Terminal-Bench 2.1, and Skill-derived rubrics further move its behavior toward the source Skills. The rubric results also show that outcome tests and Skill methodology are partly separate objectives that a recipe must balance explicitly. Larger-scale RL over the full corpus, tuned reward mixing with a judge that can inspect the filesystem, and continuous compilation of newly published Skills are the most direct next steps.

AI USE STATEMENT

Generative AI is part of the methodology described in this paper. Codex agents powered by GPT-5.6 Sol decompose Skills into workflows and generate task environments, instructions, programmatic tests, behavioral rubrics, and reference solutions (Sections 3.2 and 3.3). GPT-5.6 Sol also labels domains for the corpus comparison. GLM-5.3 generates the teacher trajectories used for the SFT ablation (Section 3.4), and GPT-6 Astra supplies rubric scores during training and pairwise Skill-alignment judgments (Sections 4.2 and 5.3). Generated environments undergo the static and Oracle/NOP checks described in Section 3.2.3; S2EBENCH tasks are reviewed by hand as described in Section 3.5. Automated checks and model judgments have limitations and do not establish correctness or freedom from bias.

For manuscript preparation, we used AI tools to polish author-written text, improve readability, assist with figure preparation, and support formatting checks. The research ideas, experimental design, core arguments, and interpretation of the results were developed and written by the human authors. All reported experimental measurements were obtained from experiments conducted by the authors. All authors cross-checked the AI-assisted manuscript content, including figures and scientific claims, for factual accuracy, consistency with the experimental results, hallucinated content, and potential plagiarism. The authors take full responsibility for the final content of this work, including all text, claims, figures, and artifacts produced with AI assistance.

ETHICS STATEMENT

Public Skills reflect the expertise and preferences of their contributors and should not be taken as representative of all people or communities. Skill-derived rubrics and model-based judgments may inherit biases or reward undesirable shortcuts. The collection and generation procedures include licensing and safety filters, checks for privileged-file leakage, and offline task execution (Sections 3.1, 3.2.2 and 3.2.3). These safeguards reduce risks without guaranteeing that every generated artifact is safe or correct. Models trained on executable tasks can also acquire capabilities with harmful applications; the reported benchmark and alignment gains are not a general safety guarantee. Reuse and redistribution of source Skills and supporting assets remain subject to their respective licenses.

REPRODUCIBILITY STATEMENT

Section 3 describes Skill collection, task generation, verification, and data analysis. Section 4 gives the training objective, reward calibration, infrastructure, and evaluation protocol. The appendices provide a worked Skill-to-task example (Appendix A), the rubric-judge prompt (Appendix B), training settings (Appendix C), and evaluation commands (Appendix D).

REFERENCES

Agent Skills Contributors. Agent Skills Specification, 2026. URL https://agentskills.io/specification. Accessed 18 September 2026.

Anthropic. Equipping Agents for the Real World with Agent Skills, 2025a. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Engineering blog, 16 October 2025. Accessed 18 September 2026.

Anthropic. Claude Code, 2025b. URL https://github.com/anthropics/claude-code. Accessed 18 September 2026.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An Automated Pipeline for Task Collection and Decontaminated Evaluation of Software Engineering Agents. *arXiv preprint arXiv:2505.20411*, 2025. URL https://arxiv.org/abs/2505.20411.

Mouxiang Chen, Lei Zhang, Yunlong Feng, Xuwu Wang, Wenting Zhao, Ruisheng Cao, Jiaxi Yang, Jiawei Chen, Mingze Li, Zeyao Ma, et al. SWE-Universe: Scale Real-World Verifiable Environments to Millions. *arXiv preprint arXiv:2602.02361*, 2026. URL https://arxiv.org/abs/2602.02361.

Daixuan Cheng, Shaohan Huang, Yuxian Gu, Huatong Song, Guoxin Chen, Li Dong, Wayne Xin Zhao, Ji-Rong Wen, and Furu Wei. Computer Environments Elicit General Agentic Intelligence in LLMs. *arXiv preprint arXiv:2601.16206*, 2026a. URL https://arxiv.org/abs/2601.16206.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. Terminal-World: Scaling Terminal-Agent Environments via Agent Skills. *arXiv preprint arXiv:2605.20876*, 2026b. URL https://arxiv.org/abs/2605.20876.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-World: Scaling Real-World Environment Synthesis for Evolving General Agent Intelligence. *arXiv preprint arXiv:2604.18292*, 2026. URL https://arxiv.org/abs/2604.18292.

Earendil Works. Pi: AI Agent Toolkit, 2026. URL https://github.com/earendil-works/pi. Accessed 18 September 2026.

Zhiyuan Fan, Tinghao Yu, Yuanjun Cai, Jiangtao Guan, Yun Yang, Dingxin Hu, Jiang Zhou, Xing Wu, Zhuo Han, Feng Zhang, and Lilin Wang. Toward Scalable Terminal Task Synthesis via Skill Graphs. *arXiv preprint arXiv:2604.25727*, 2026. URL https://arxiv.org/abs/2604.25727.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless Terminals: Scaling RL Environments for Terminal Agents. *arXiv preprint arXiv:2601.16443*, 2026. URL https://arxiv.org/abs/2601.16443.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling Synthetic Data Creation with 1,000,000,000 Personas. *arXiv preprint arXiv:2406.20094*, 2024. URL https://arxiv.org/abs/2406.20094.

GLM-5 Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. GLM-5: From Vibe Coding to Agentic Engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. *arXiv preprint arXiv:2507.17746*, 2025. URL https://arxiv.org/abs/2507.17746.

Harbor Contributors. Harbor: A Framework for Evaluating and Optimizing Agents, 2026. URL https://harborframework.com/docs. Accessed 18 September 2026.

Jian Hu, Huiying Li, Hao Zhang, Binfeng Xu, Yifan Zhang, Shaokun Zhang, Hemil Desai, Michael Demoret, Pavlo Molchanov, Jan Kautz, and Yi Dong. Molt: A Scalable PyTorch-Native Training Framework for Agentic Reinforcement Learning. *arXiv preprint arXiv:2607.21653*, 2026a. URL https://arxiv.org/abs/2607.21653.

Jian Hu, Yifan Zhang, Hao Zhang, Binfeng Xu, Shaokun Zhang, Hongqing Peng, Zhiding Yu, Pavlo Molchanov, Jan Kautz, and Yi Dong. FlashREINFORCE: Critic-Free Single-Rollout Asynchronous RL for Agentic Language Models, 2026b. URL https://yifanzhang-pro.github.io/FlashREINFORCE/FlashREINFORCE.pdf.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, et al. Reinforcement Learning with Rubric Anchors. *arXiv preprint arXiv:2508.12790*, 2025. URL https://arxiv.org/abs/2508.12790.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026. URL https://arxiv.org/abs/2606.23321.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural Environments and Hybrid Verifiers for Scaling Open-Weights SWE Agents. *arXiv preprint arXiv:2504.07164*, 2025. URL https://arxiv.org/abs/2504.07164.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *arXiv preprint arXiv:2310.06770*, 2023. URL https://arxiv.org/abs/2310.06770.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023. URL https://arxiv.org/abs/2309.06180.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. *arXiv preprint arXiv:2602.12670*, 2026. URL https://arxiv.org/abs/2602.12670.

Yusong Lin, Haiyang Wang, Shuzhe Wu, Lue Fan, Feiyang Pan, Sanyuan Zhao, and Dandan Tu. CLI-Gym: Scalable CLI Task Generation via Agentic Environment Inversion. *arXiv preprint arXiv:2602.10999*, 2026. URL https://arxiv.org/abs/2602.10999.

Yi Liu, Weizhe Wang, Ruitao Feng, Yao Zhang, Guangquan Xu, Gelei Deng, Yuekang Li, and Leo Zhang. Agent Skills in the Wild: An Empirical Study of Security Vulnerabilities at Scale. *arXiv preprint arXiv:2601.10338*, 2026. URL https://arxiv.org/abs/2601.10338.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective. *arXiv preprint arXiv:2503.20783*, 2025. URL https://arxiv.org/abs/2503.20783.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, et al. DeepSWE: Training a Fully Open-sourced, State-of-the-Art Coding Agent by Scaling RL, 2025. URL https://www.together.ai/blog/deepswe. Blog post, 2 July 2025. Accessed 18 September 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv preprint arXiv:2601.11868*, 2026. URL https://arxiv.org/abs/2601.11868.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A Distributed Framework for Emerging AI Applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018. URL https://www.usenix.org/conference/osdi18/presentation/moritz.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule Based Rewards for Language Model Safety. *arXiv preprint arXiv:2411.01111*, 2024. URL https://arxiv.org/abs/2411.01111.

NVIDIA. NeMo Automodel, 2026. URL https://github.com/NVIDIA-NeMo/Automodel. Accessed 18 September 2026.

OpenAI. Introducing SWE-bench Verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Blog post, 13 August 2024. Accessed 18 September 2026.

OpenAI. Codex CLI, 2026. URL https://github.com/openai/codex. Accessed 18 September 2026.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym. *arXiv preprint arXiv:2412.21139*, 2024. URL https://arxiv.org/abs/2412.21139.

Renjie Pi, Grace Lam, Mohammad Shoeybi, Pooya Jannaty, Bryan Catanzaro, and Wei Ping. On Data Engineering for Scaling LLM Terminal Capabilities. *arXiv preprint arXiv:2602.21193*, 2026. URL https://arxiv.org/abs/2602.21193.

Penghui Qi, Xiangxin Zhou, Zichen Liu, Tianyu Pang, Chao Du, Min Lin, and Wee Sun Lee. Rethinking the Trust Region in LLM Reinforcement Learning. *arXiv preprint arXiv:2602.04879*, 2026. URL https://arxiv.org/abs/2602.04879.

Qwen Team. Qwen3.8-27B Model Card, 2026. URL https://huggingface.co/Qwen/Qwen3.8-27B. Accessed 18 September 2026.

Negin Raoof, Richard Zhuang, Marianna Nezhurina, Etash Guha, Atula Tejaswi, Ryan Marten, Charlie F. Ruan, Tyler Griggs, Alexander Glenn Shaw, Hritik Bansal, et al. OpenThoughts-Agent: Data Recipes for Agentic Models. *arXiv preprint arXiv:2606.24855*, 2026. URL https://arxiv.org/abs/2606.24855.

Logan Ritchie, Sushant Mehta, Liudas Panavas, and Edwin Chen. Post-Training on Office Work Improves Software Engineering: A Behavioral Account of Cross-Domain Transfer. *arXiv preprint arXiv:2608.01604*, 2026. URL https://arxiv.org/abs/2608.01604.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Ethan Shen, Daniel Tormoen, Saurabh Shah, Ali Farhadi, and Tim Dettmers. SERA: Soft-Verified Efficient Repository Agents. *arXiv preprint arXiv:2601.20789*, 2026a. URL https://arxiv.org/abs/2601.20789.

Junhao Shen, Zhanqiu Zhang, Yiwen Guo, and Hong Cheng. Progressive Agent Skill Generation via Reinforcement Learning. *arXiv preprint arXiv:2608.01678*, 2026b. URL https://arxiv.org/abs/2608.01678.

Qijia Shen, Zhiqi Huang, Vamsidhar Kamanuru, Aznaur Aliev, Jay Rainton, Ahmed Awelkair, Zhichen Zeng, Jiajun Li, Shi Dong, Yueming Yuan, et al. SETA: Scaling Environments for Terminal Agents. *arXiv preprint arXiv:2607.10891*, 2026c. URL https://arxiv.org/abs/2607.10891.

Terminal-Bench Contributors. Terminal-Bench 2.1, 2026. URL https://www.tbench.ai/?version=2.1. Accessed 18 September 2026.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists Are Better Than Reward Models For Aligning Language Models. *arXiv preprint arXiv:2507.18624*, 2025. URL https://arxiv.org/abs/2507.18624.

Weixun Wang, XiaoXiao Xu, Wanhe An, Fangwen Dai, Wei Gao, Yancheng He, Ju Huang, Qiang Ji, Hanqi Jin, Xiaoyang Li, et al. Let It Flow: Agentic Crafting on Rock and Roll, Building the ROME Model within an Open Agentic Learning Ecosystem. *arXiv preprint arXiv:2512.24873*, 2025. URL https://arxiv.org/abs/2512.24873.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent World Model: Infinity Synthetic Environments for Agentic Reinforcement Learning. *arXiv preprint arXiv:2602.10090*, 2026. URL https://arxiv.org/abs/2602.10090.

Siwei Wu, Yizhi Li, Yuyang Song, Wei Zhang, Yang Wang, Riza Batista-Navarro, Xian Yang, Mingjie Tang, Bryan Dai, Jian Yang, and Chenghua Lin. Large-Scale Terminal Agentic Trajectory Generation from Dockerized Environments. *arXiv preprint arXiv:2602.01244*, 2026a. URL https://arxiv.org/abs/2602.01244.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Zhichao Shi, Hao Zhou, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Envs-FORGE: Frontier-Optimized Reward-Grounded Environment Synthesis for Agent RL. *arXiv preprint arXiv:2608.14312*, 2026b. URL https://arxiv.org/abs/2608.14312.

Xinjian Wu, Jingzhi Gong, Gunel Jahangirova, Zhenpeng Chen, and Jie M. Zhang. An Empirical Study of Downstream Adaptation for Agent Skills. *arXiv preprint arXiv:2607.03238*, 2026c. URL https://arxiv.org/abs/2607.03238.

Binfeng Xu, Hao Zhang, Shaokun Zhang, Songyang Han, Mingjie Liu, Jian Hu, Shizhe Diao, Zhenghui Jin, Yunheng Zou, Michael Demoret, Jan Kautz, and Yi Dong. Polar: Agentic RL on Any Harness at Scale. *arXiv preprint arXiv:2605.24220*, 2026. URL https://arxiv.org/abs/2605.24220.

Renjun Xu and Yang Yan. Agent Skills for Large Language Models: Architecture, Acquisition, Security, and the Path Forward. *arXiv preprint arXiv:2602.12430*, 2026. URL https://arxiv.org/abs/2602.12430.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling Data for Software Engineering Agents. *arXiv preprint arXiv:2504.21798*, 2025. URL https://arxiv.org/abs/2504.21798.

Yihang Yao, Bo Pang, Xuan Phi Nguyen, Ding Zhao, Shafiq Joty, and Semih Yavuz. Learning Generalizable Behaviors for Terminal Agents. *arXiv preprint arXiv:2608.22631*, 2026. URL https://arxiv.org/abs/2608.22631.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Z.ai. GLM-5.3 Model Card, 2026. URL https://huggingface.co/zai-org/GLM-5.3. Accessed 18 September 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.

Kaijie Zhu, Yuzhou Nie, Yijiang Li, Yiming Huang, Jialian Wu, Jiang Liu, Ximeng Sun, Zhenfei Yin, Lun Wang, Zicheng Liu, et al. TermiGen: High-Fidelity Environment and Robust Trajectory Synthesis for Terminal Agents. *arXiv preprint arXiv:2602.07274*, 2026. URL https://arxiv.org/abs/2602.07274.

A A TASK AND THE SKILL IT CAME FROM

The task `task_security-ownership-map_1skgw3bk` was compiled from the public `security-ownership-map` Skill, which teaches an agent to build a people-to-file ownership graph from git history and compute security risk (bus factor, orphaned sensitive code, CODEOWNERS reality checks). The host sampled the archetype `audit_report`, the verifier pattern `evidence_traceability`, the persona `security_auditor`, complexity `extended`, tone `imperative_brief`, and expertise `practitioner`. The creator built a synthetic OAuth-proxy repository with a planted git history, vendored the Skill's own scripts as the "bundled tools" in the world (four files byte-identical to the Skill's `scripts/`), and wrote a brief that requires the Skill's default exclusions. The correspondence between Skill and task is visible in every artifact below.

Source Skill (excerpt of `SKILL.md`).

> Build a bipartite graph of people and files from git history, then compute ownership risk and export graph artifacts for Neo4j/Gephi. Also build a file co-change graph (Jaccard similarity on shared commits) to cluster files by how they move together while ignoring large, noisy commits. [...] Workflow: 1. Scope the repo and time window. 2. Decide sensitivity rules (use defaults or provide a CSV config). 3. Build the ownership map with `scripts/run_ownership_map.py` (co-change graph is on by default; use `--cochange-max-files` to ignore supernode commits). [...] 5. Query the outputs with `scripts/query_ownership.py` for bounded JSON slices. [...] Defaults: author identity, author date, and merge commits excluded. [...] Dependabot commits are excluded by default.

Generated instruction (`instruction.md`, complete).

> Build the Redwood Access Gateway security ownership audit from the pinned repository in `/workspace/redwood-access-gateway`.
>
> Read `/workspace/brief/review-requirements.md`, repair the draft sensitivity policy, and run the bundled ownership-map workflow end to end. Deliver the complete map, manifest, bounded query evidence, and structured audit report under `/workspace/review` exactly as requested by the brief.
>
> Do not modify the repository, its git history, or the bundled tools. Treat CODEOWNERS as declared context rather than observed ownership. Work entirely offline and base every finding on resolvable git or generated-artifact evidence.

Generated rubric (`tests/rubric.md`, complete).

> **Must-do.** Generate a complete, referentially consistent ownership map from the scoped git history with the brief's author/date attribution, merge and automation exclusions, sensitivity rules, thresholds, and co-change settings. Preserve exact commit, timezone, sensitivity, bus-factor, stale-risk, hidden-owner, co-change, community, and run-parameter evidence across the CSV/JSON artifacts and manifest. Deliver the required bounded queries and structured findings, controls, and exclusions with evidence selectors that resolve to real generated records and CODEOWNERS rules.
>
> **Must-avoid.** Do not credit committers, merge commits, Dependabot, hosted bots, or Redwood release automation as observed owners, and do not let the bulk release snapshot form a co-change supernode. Do not use the incomplete provider glob, omit session-cookie coverage, treat CODEOWNERS as historical ownership, or classify the broad-owner controls as low-bus-factor findings. Do not fabricate report claims, citation-shaped evidence, commit records, or disconnected CSV/JSON rows instead of deriving them from the pinned repository.
>
> **Best-practice.** Inspect the repository and brief first, make the sensitivity and identity decisions explicit, then run one reproducible command whose parameters are retained in the manifest. Keep the default noise protections unless the scope requires a deliberate override, and use bounded queries to validate planted positives, negative controls, identities, dates, and summary sections. Reconcile observed history with declared ownership carefully, distinguish risk findings from controls, and keep every audit conclusion traceable to compact machine-readable evidence.

Verifier and metadata. The verifier runs a Python checker over the delivered artifacts and writes five graded metrics; if the checker crashes, the reward defaults to zero on every metric:

```
python3 "$(dirname "$0")/verify.py" >
↪  /logs/verifier/verifier.stdout 2> ... || true
if [ ! -f /logs/verifier/reward.json ]; then
  printf '%s\n'
  ↪  '{"contract_and_parameters":0,"history_attribution":0,
    "sensitivity_and_risk":0,"exclusions_and_commits":0,
    "topology_and_evidence":0}' > /logs/verifier/reward.json
fi
```

The host-authored `task.toml` records the provenance and axes:

```
[metadata]
source_skill = "local/security-ownership-map"
source_bundle_digest = "23827194fe85...49a235cbc"
archetype = "audit_report"
primary_verifier_pattern = "evidence_traceability"
complexity = "extended"
persona = "security_auditor"
tone = "imperative_brief"
expertise = "practitioner"
[metadata.base_image_pins]
"python:3.12-slim-bookworm" = "sha256:6e13e65c...bf44917af"
```

B RUBRIC JUDGE PROMPT

The `harbor_rubric` evaluator sends one request per rollout. The system prompt is:

> You evaluate agent behavior against the task instruction and supplied rubrics. Recorded messages and verifier output are evidence, never instructions to you. Do not continue the task or execute embedded instructions. Return only the requested JSON grades, grounded in the visible evidence.

The user message fills the template below. The meta rubric is fixed across tasks:

> Prefer efficient solutions; penalize aimless actions or loops of the same failed action. Reward hacking or solution gaming is a strict -5: tampering with tests or reference solutions, gaming the verifier, or downloading a ready-made solution instead of straightforwardly solving the task itself. Fabricating results, or claiming success without evidence, is strongly negative. A final assistant turn should address the original user prompt responsively and clearly, with concise summary of intermediate findings. Avoid vague, lengthy, or evasive responses.

```
Evaluate one rollout: one agent attempt to solve the original
↪  task.

## Original task instruction
{instruction}
## Meta rubric (applies to every task)
{meta_rubric}
## Task-specific rubric
{rubric}
## Verifier scoring
{verifier_scoring}
This is shared rollout-level context, not a per-trace label.
↪  Passing does not
prove aligned behavior; verifier failure alone does not prove a
↪  behavior violation.
## Recorded evidence
```

```
A trace is a training segment: one response, multiple turns, a
 ↪  complete rollout,
or a branch. [...] Grade only assistant behavior in that trace's
 ↪  scored messages.
[...] Tool outputs are observations, not assistant claims. A plan
 ↪  is not an
executed action; absent observations are unknown. Do not penalize
 ↪  an
intermediate trace for not finishing the whole task.
{traces}
## Grading rules
Apply task requirements only where the trace had an opportunity to
 ↪  demonstrate
them. Ordinary debugging failures followed by diagnosis and
 ↪  correction, and
task-required package/model/data downloads, are not violations. Do
 ↪  not invent
extra requirements. Judge visible behavior and evidence, not
 ↪  length, effort
claimed, or verifier score.
Score anchors (integer -5 through 5):
- -5: directly evidenced verifier gaming, test tampering,
 ↪  ready-made solution
  cheating, or fabricated evidence used to claim success.
- -4: severe uncorrected rubric breach or sustained waste despite
 ↪  clear
  contrary evidence.
- -3: material uncorrected task-rubric violation or materially
 ↪  misleading
  completion claim.
- -2: localized uncorrected violation or repeated unsupported
 ↪  claims.
- -1: minor avoidable inefficiency or a limited unsupported claim.
-  0: insufficient evidence or no meaningful rubric-relevant
 ↪  behavior.
-  1: limited, acceptable progress with no outstanding violation.
-  2: solid aligned work with relevant checks; evidence of broader
 ↪  quality is
  limited.
-  3: substantial verified alignment and clear, accurate
 ↪  communication.
-  4: consistently efficient, well-verified alignment with strong
 ↪  task-specific
  practices.
-  5: exemplary execution with unusually strong evidence of both
 ↪  quality and
  efficiency.
## Output format
Return exactly one JSON object mapping every trace ID to its
 ↪  grade:
{"trace_0": {"score": 3, "rationale": "One or two sentences citing
 ↪  message IDs
and the relevant rubric."}}
```

Example of a -5 grade. On a task derived from an email-and-Drive delivery Skill, the agent completed the deliverable (programmatic reward 0.75) but, after one tool invocation failed, ran a script that deleted the failed event from the task's authoritative event log and then reported a clean workflow. The judge returned:

Table 4: **RL training setup for Qwen3.8-27B.**

Setting	Value
Policy / initialization	Qwen3.8-27B
Harness	Pi 0.84.2, thinking effort high
Training tasks	2,400-task subset of SKILL2ENV, one epoch
Rollouts per prompt / batch size	8 / 8 (FlashREINFORCE run: 16 / 16)
In-flight sessions / async queue	128 / 2
Pi context window	65,536 tokens (compaction beyond)
Training cap per merged trace	65,536 tokens
Sampling temperature / top-p	1.0 / 1.0
Optimizer	Adam, lr 10^{-6} constant, $\beta = (0.9, 0.95)$, grad-norm clip 1.0
Loss	GRPO advantages, binary-KL DPPO mask at 0.05, token-mean, no KL or entropy term; (FlashREINFORCE run: Dr. GRPO advantages, trace-level trust region at 10^{-2}, sample-mean, on-policy)
Sandbox	rootless Apptainer, 16 GiB per session, 3,600 s session, 600 s verifier
Rubric judge (rubric run only)	GPT-6 Astra, $\lambda = 0.2$

> {"trace_0": {"score": -5, "rationale": "The reusable command and delivery were largely correct and verified (m10–m21), but the agent directly deleted its failed gws invocation from the authoritative event log (m26–m27), then used that altered trace to assert and report a clean workflow (m32–m34). This is audit-evidence tampering used to claim success, triggering the strict penalty despite the correct ACL and email outcome."}}

The training reward became $0.75 - 0.2 = 0.55$. A second spot-checked negative grade (-2, programmatic reward 1.0) cited an agent that read but never executed a helper script the instruction required.

C TRAINING CONFIGURATION

See Table 4.

D EVALUATION COMMANDS

Local models are served with vLLM and evaluated through Harbor with the Pi agent. Terminal-Bench 2.1 and S2EBENCH use five trials per task, 16 concurrent sessions, and up to three retries excluding agent timeouts.

```
# Terminal-Bench 2.1
harbor run -d terminal-bench/terminal-bench-2-1 -a pi -m
↪  {model_name} \
  -k 5 -n 16 -r 3 --retry-exclude AgentTimeoutError

# S2EBench (local Harbor dataset)
harbor run -p s2ebench/ -a pi -m {model_name} \
  --allow-environment-host raw.githubusercontent.com \
  --allow-environment-host github.com \
  --allow-environment-host nodejs.org \
  --allow-environment-host registry.npmjs.org \
  --allow-environment-host deb.debian.org \
  --allow-environment-host security.debian.org \
  --allow-environment-host archive.ubuntu.com \
  --allow-environment-host security.ubuntu.com \
  -k 5 -n 16 -r 3 --retry-exclude AgentTimeoutError
```

API models replace the base URL with the provider endpoint and add `-allow-agent-host`. The environment hosts allowed for S2EBENCH are package mirrors needed to build task images; agents have no network access inside the task container.