Muse Spark 1.2 & Muse Code

Evaluation Methodology

We evaluate Muse Spark 1.2 with Muse Code on Terminal-Bench 2.1 (Stanford × Laude Institute) and DeepSWE v1.1 (Datacurve), alongside results on GDPVal-AA v2 (Artificial Analysis), MCP Atlas (Scale AI), and Meta Internal Coding Bench.

Overall Methodology

- **Models and agents.** Where comparable results are available, we compare Muse Spark 1.2 with Muse Spark 1.1, Grok 4.5, Claude Opus 5, GPT-5.6 Terra, Gemini 3.6 Flash, and Kimi K3. For Terminal-Bench 2.1 and DeepSWE 1.1, each model is evaluated with its selected agent product: Muse Code for Muse Spark 1.2, mini-swe-agent for Muse Spark 1.1, Grok Build for Grok, Claude Code for Opus, Codex for GPT, Antigravity for Gemini, and Kimi Code for Kimi. We use the maximum available reasoning strength for each model: xhigh reasoning effort for Muse Spark 1.2 and Muse Spark 1.1, high for Grok and Gemini, and max for Opus, GPT, and Kimi. Muse Spark 1.2 is served through the Meta Model API. GDPVal-AA v2 and MCP Atlas use the benchmark providers' agent harnesses and therefore are not CLI agent comparisons.
- **Evaluation framework.** Our Terminal-Bench 2.1 and DeepSWE 1.1 runs use an internal agent evaluation framework. Each task attempt runs the selected agent and model in an isolated Daytona cloud sandbox. We use the official benchmark datasets or faithfully convert them to Harbor format. Each benchmark's verifier grades the final result. GDPVal-AA v2 results come from Artificial Analysis, and MCP Atlas results come from Scale AI. Meta Internal Coding Bench uses a separate internal agentic harness and dedicated grading containers.
- **Agentic evaluations for third-party models:** The results represent our best-effort evaluations of third-party models. Where applicable, we use a common evaluation framework and align benchmark settings as closely as is practical. We note that our evaluation setup (e.g., agent tools and system prompts) may not be specifically tuned for proprietary third-party models. Therefore, the results may not reflect these models' best performance when used in environments tailored to their specific strengths.

Terminal benchmarks

Terminal-Bench 2.1: Terminal-Bench 2.1 evaluates agents on tasks completed in a terminal environment. The Terminal-Bench paper describes the preceding 2.0 release. Our evaluation uses all 89 tasks in the official 2.1 release. Each attempt runs the selected agent in an isolated Daytona sandbox. The task's executable verifier then evaluates the final container state. We report the average task success rate (pass@1) across five attempts.

Software engineering

DeepSWE v1.1: Datacurve's DeepSWE paper defines 113 tasks across 91 repositories and five languages: TypeScript, Go, Python, JavaScript, and Rust. Each task has a handwritten functional verifier and regression checks. We use the Harbor-format dataset and run tasks in isolated Daytona cloud sandboxes. We attempt to follow Pier v0.3.0 as closely as possible, the benchmark's official runner. The agent's final patch is applied to a pristine checkout in a fresh verifier container. External internet access is blocked during rollout and grading, and only the model endpoint remains reachable. A task passes only when its functional verifier and regression checks both succeed. The official leaderboard uses mini-swe-agent for every model. Our evaluation instead uses each model's selected agent product and therefore is not harness-identical to the leaderboard. We report the average task success rate (pass@1) across five attempts.

Professional tasks

GDPVal-AA v2: GDPVal-AA v2 evaluates agents on 220 real-world professional tasks from OpenAI's GDPval dataset. The tasks span 44 occupations and nine major U.S. industries and require deliverables such as documents, spreadsheets, slide decks, diagrams, and reports. Models operate in Artificial Analysis's Stirrup agentic harness with shell access and web browsing. For each task, an LLM judge compares anonymized deliverables head-to-head. The primary metric is the Elo rating derived from these blind pairwise comparisons, with the human baseline anchored at 1,000. We use results produced by Artificial Analysis.

MCP tool use

MCP Atlas: MCP Atlas evaluates agents on 1,000 human-authored tasks across 36 real Model Context Protocol servers and 220 tools. The public and private splits each contain 500 tasks. Tasks run against containerized MCP servers with a controlled set of target and distractor tools. An LLM judge assigns each ground-truth claim a score: 1 if fulfilled, 0.5 if partially fulfilled, and 0 if not fulfilled. A task's coverage is the mean of its claim scores, and the task passes when coverage is at least 0.75. The primary metric is pass

rate, the percentage of tasks that pass. We use the result produced by Scale AI with the benchmark's own agent harness and scoring pipeline.

Internal coding

Meta Internal Coding Bench: Meta Internal Coding Bench consists of 440 tasks sourced from Meta's internal codebase. The tasks are derived from real internal pull requests and cover bug fixes, feature development, refactoring, code cleanup, and other software engineering work. Internet access is disabled, and each task runs in an isolated sandbox using an internal agentic harness. Submissions are compiled and evaluated against unit tests in dedicated grading containers. We use two attempts per task. For each model, we compute each task's success rate across its attempts and then average those task-level rates. The primary metric is percent resolved (pass@1).