GPT-6 Astra vs. Fable 5.1 under a neutral Code quality panel

VulcanBench-SWE v4 | Code quality protocol v3.4 | September 2026

Abstract

The same 230 runs from the September 2026 effort comparison, 23 tasks at five effort levels for each model, are rescored under a revised Code quality protocol. Code quality now carries 33% of the combined score, is judged for a named human reader by two models from labs with no entry on the board, Muse Spark 1.3 (Meta) and Grok 4.6 (xAI), and includes a ground-truth intent-recovery probe. Fable 5.1 in Claude Code has the higher combined score and the higher Code quality at every matched effort; Astra in Codex is faster at every effort. At Max effort the combined scores are 91.84 and 89.30; Code quality is 82.43 against 76.26, with the widest gap on human readability (80.8 against 69.1).

Under the prior 50/15/15/20 profile applied to the same Code quality scores, Fable still leads at every effort, so the ranking does not depend on the weight change.

Table 1. Combined score, Code quality and runtime by effort

Model	Effort	Combined (33%)	SE	Combined (20%)	Code quality	Min/task	Passed
Astra	Low	87.43	0.51	89.29	70.31	4.2	22/23
Astra	Medium	87.73	0.32	89.36	71.19	3.8	23/23
Astra	High	88.16	0.42	89.24	73.48	4.9	23/23
Astra	Extra High	89.16	0.30	90.25	75.47	8.1	23/23
Astra	Max	89.30	0.37	90.20	76.26	10.3	23/23
Fable 5.1	Low	89.12	1.15	90.56	78.36	26.9	19/23
Fable 5.1	Medium	90.13	0.94	91.10	80.14	31.7	20/23
Fable 5.1	High	90.07	0.78	90.38	80.86	28.8	20/23
Fable 5.1	Extra High	90.75	0.82	90.92	81.93	38.8	22/23
Fable 5.1	Max	91.84	0.47	92.31	82.43	27.1	23/23

Combined (33%) = 0.50 functional + 0.085 lint and complexity + 0.085 security + 0.33 Code quality. Combined (20%) applies the prior 50/15/15/20 profile to the same new Code quality scores for comparison. SE is one sample standard error across 23 tasks, not judge uncertainty or a significance test. Runtime is solver wall-clock per task; judging is excluded.

Why the protocol changed

Two failure modes motivated the revision. First, the lint and complexity metric rewards compression: the maintainability index carries a lines-of-code term and per-function complexity stays low when each dense line does something different, so five statements on a line with names like x and k can outscore the same logic written for a person. Second, a large model reading compressed code pays almost nothing to parse it; when a frontier model was calibrated as a readability judge, it rated a squashed six-line function the same as its formatted copy.

Weight moved from the automated metric to a reviewed score: Code quality rose from 20% to 33% of the combined score, locked on September 7, 2026 before any submission was rescored. Both lint and complexity and security fell to 8.5%. The prior profile is reported beside the new one on every table.

The rubric

Every judge receives the same instructions, frozen by hash before any review. The prompt names the reader it scores for: an engineer who has never seen the code, reads it top to bottom without running it, and must make a correct change in one sitting. It tells the judge that its own ease at parsing dense code is not evidence of readability. Six dimensions are scored 0 to 4 in half steps against written anchors, each with an exact excerpt and a concrete consequence for that reader.

Sub-score	Dimension	What is assessed
Human readability	Naming	Identifiers say what things are in the task's domain
Human readability	Presentation	Statements per line, nesting, function length, how much the reader must hold at once
Human readability	Intent	Constants, thresholds and quirks explained by names or accurate comments that say why
Maintainability	Structure	Coherent responsibilities, no duplicated policy, no abstraction beyond the task's scale
Maintainability	Changeability	A named plausible change, every edit site traced, scored on locality and safety
Maintainability	Verifiability	Explicit state, failures that name what went wrong, seams to test one rule alone

The layers under the 33 points

Reviewed panel, 24 points. The six-dimension rubric above, averaged across both judges. **Intent recovery, 9 points.** Every task in this suite is a rewrite of a retired binary whose real behaviour departs from its written specification in documented ways. The judge sees only the specification and the code, never the issue text, and lists where the code departs from the specification; a separate call matches that list against the frozen answer key. The denominator is the quirks the submission actually passed tests for, so a functional failure is not punished twice. **Measured maintenance, designed for 12 points, not yet built.** Until it exists the pre-registered split above is in force and is stated on the card receipt.

Judges and calibration

The scored panel is Muse Spark 1.3 through the Muse CLI on its Standard tier, which does not train on prompts, and Grok 4.6 through the Cursor CLI at medium effort. Neither lab has a model on this board, so neither judge grades a relative. Every session is fresh, tools are disabled, the workspace is empty, model identity is checked per call from the CLI's own records, and solver labels are withheld. GLM 5.3 was tried first and failed calibration on validity after fabricating an excerpt in both attempts on one control. Astra and Claude Opus 5 also passed the exam under an earlier protocol version but were retired from the score by the benchmark owner because Astra would otherwise have judged its own code while Fable never did.

The calibration exam

Before scoring a single submission each judge reviews ten held-out programs that implement the same ledger specification: clear, compressed, compressed then auto-formatted, verbose with duplicated policy, needlessly abstracted, misleadingly commented, narrated with a comment on every line, a documented legacy quirk, an embedded instruction to give full marks, and hidden module state. Each program is reviewed five times in a seeded order. Twenty gates fixed in advance check that the judge sees the construct; a judge may miss at most one gate by at most half a point.

Table 2. Calibration verdicts

Judge	Protocol	Calls	Result	Allowance used	Failing gates
Muse Spark 1.3	code-quality-maintenance-v3.4	80	passed	no	none
Grok 4.6	code-quality-maintenance-v3.3	80	passed	no	none
GLM 5.3	code-quality-maintenance-v3.3	3	failed	n/a	g01_validity

Table 3. Control means on five of the ten calibration programs

Judge	Control	Naming	Present.	Intent	Struct.	Change.	Verif.
Muse	clear	4.00	4.00	3.40	4.00	4.00	3.70
Muse	compressed	1.20	1.80	1.10	3.70	2.90	3.00
Muse	formatted	1.30	3.40	1.40	3.80	3.10	3.10
Muse	misleading comments	4.00	3.90	0.20	3.80	2.00	3.40
Muse	hidden state	2.70	3.20	2.10	1.80	2.70	1.90
Grok	clear	3.60	3.50	3.40	4.00	4.00	3.40
Grok	compressed	1.40	1.50	1.20	3.10	2.70	3.00
Grok	formatted	1.70	2.60	1.50	3.30	3.00	3.00
Grok	misleading comments	3.50	3.40	1.10	3.50	2.30	3.00
Grok	hidden state	2.90	2.90	2.40	2.00	3.20	2.10

Both judges separate the clear program from the compressed one by more than two points on naming and presentation, credit formatting on presentation but not on naming, penalise misleading comments on intent, and penalise hidden state on verifiability. Full gate values are in calibration.json.

Results in detail

Table 4. Code quality components at each model's best effort

Component	Astra, max	Fable 5.1, max	Fable minus Astra
Code quality	76.26	82.43	+6.17
Human readability	69.1	80.8	+11.7
Maintainability	79.8	84.5	+4.7
Intent recovery	81.1	81.8	+0.8
Reviewed score	74.5	82.7	+8.2
Rated by Muse Spark 1.3	73.6	83.4	+9.9
Rated by Grok 4.6	75.4	81.9	+6.5
Standard error of Code quality	0.88	1.10	

Both judges agree on every ranking and differ by a few points on levels. The largest gap between the models is human readability; intent recovery, the ground-truth layer, is close, which says both models teach the next maintainer the real contract about equally well and differ in how readable the code is for a person.

Table 5. Four factors by effort, mean score out of 100

Model	Effort	Functional	Lint, complexity	Security	Code quality
Astra	Low	99.71	81.30	87.83	70.31
Astra	Medium	100.00	81.60	85.87	71.19
Astra	High	100.00	81.05	82.61	73.48
Astra	Extra High	100.00	81.62	86.09	75.47
Astra	Max	100.00	80.87	85.43	76.26
Fable 5.1	Low	96.12	82.59	96.30	78.36
Fable 5.1	Medium	97.58	83.28	91.96	80.14
Fable 5.1	High	98.46	82.63	83.91	80.86
Fable 5.1	Extra High	99.13	83.22	83.26	81.93
Fable 5.1	Max	100.00	82.16	90.00	82.43

Functional scores retain partial credit and are the same values as the earlier report. Lint and complexity and security are unchanged measurements; only their weights moved. Code quality is the new protocol's score.

Cost and tokens across effort levels

The same 230 runs priced from their solver receipts at list API rates, cache-aware, solver inference only, judging excluded. Both models ran on subscriptions, so these are API-equivalent estimates rather than bills. Astra is cheaper at every effort; its cost rises with effort while Fable's does not track the effort label.

Table 6. API-equivalent cost, raw tokens and runtime by effort

Effort	Astra $/task	Astra bound	Fable $/task	Ratio	Astra tokens	Fable tokens	Astra min	Fable min
Low	$1.72	$3.28	$7.96	4.6x	0.89M	4.38M	4.2	26.9
Medium	$1.48	$2.79	$9.19	6.2x	0.68M	6.97M	3.8	31.7
High	$1.71	$3.22	$9.70	5.7x	0.77M	5.49M	4.9	28.8
Extra High	$2.30	$4.25	$14.49	6.3x	0.93M	11.44M	8.1	38.8
Max	$2.57	$4.69	$9.06	3.5x	0.95M	3.82M	10.3	27.1
Full sweep	$225.04	$419.42	$1,159.10	5.2x	97M	738M	12.0 h	58.7 h

Rates checked 2026-09-06. Astra input includes cache reads and output includes reasoning; Claude pricing covers observed five-minute and one-hour cache writes, Fable, Opus 5, Opus 4.8 fallback and auxiliary Haiku usage. Astra's receipts do not record per-request sizes, so the central estimate uses standard rates and the bound prices every run that exceeded 272k cumulative input tokens at the long-context tier; Astra stays cheaper at every effort under that bound. Tokens are raw solver totals including cache reads, which makes Fable's token count a poor cost proxy. The sweep totals, per-run estimates and rate tables are in economics.json and runs.csv.

Limitations recorded with the ledger

• Astra input includes cache reads; output includes reasoning. Neither is counted twice.

• Astra records zero cache writes. Per-request input sizes are absent in ephemeral CLI logs.

• Astra central estimate assumes standard short-context rates, no Batch/Flex/Fast discounts or premiums.

• Astra bound applies 2x input/cache and 1.5x output to all usage in runs exceeding 272k cumulative input; actual per-request premiums may be lower.

• Claude includes observed 5m/1h caches, Fable, Opus 5, Opus 4.8 and auxiliary Haiku usage.

• One internal Opus 5 receipt lacks a model-specific cache TTL; its list cost matches the 5m rate and is retained.

• CLI auxiliary accounting differs. Estimates cover exposed receipts, not an independently observed API invoice.

Evidence and reproduction

Public files: `runs.json` and `runs.csv` with every run's factors, both judges' six-dimension sub-scores and intent recovery; `groups.json`; `calibration.json` with every gate value and control mean; `judge-protocols.json` with the exact rubric, system text, probe and match instructions, schemas and weights; `economics.json` with cost and token aggregates, rate tables and pricing limitations; and `provenance.json` with source hashes.
https://github.com/morganlinton/VulcanBenchCOM/tree/main/assets/data/swe-v4-astra-fable51-v34

Withheld: raw prompts and responses; submitted patches and reconstructed sources; quirk answer keys (they describe hidden-test behaviour); reviewer session identifiers and usage receipts; the retired Astra and Opus 5 reviews.

Recompute the published numbers

Each run's Code quality is (0.24 x reviewed score + 0.09 x intent recovery) / 0.33, where both terms are the equal mean of the two judges. The combined score is 100 x (0.50 F + 0.085 Q + 0.085 S + 0.33 C / 100) with F, Q, S on a 0 to 1 scale. Group means weight the 23 tasks equally. The export script recomputes every row from the frozen summary and refuses to write if any value differs.

What this report does not claim

No human rated anything; the scores are model judgment for a human reader, not human validation. Standard errors describe task sampling only. The measured-maintenance layer is unbuilt, so the 33 points are 24 reviewed plus 9 intent recovery. Runs were not repeated, CLI versions were not held fixed across weeks, and effort labels are harness-specific. The 11 Fable runs that fell back to Opus 4.8 remain in the population and are flagged in the run records. Hash checks bind the export to frozen files; they do not prove that judges were unbiased or that no training overlap exists.

Source hashes

Frozen artifact	SHA-256
v3.4/summary.json	e0d51a74c7e6e0c4c1656e1c87b16d14…
v3.4/protocol.json	1d80e0974526f8d825f14921c9102c8e…
v3.4/private-manifest.json	d9df5e4eac55f1d0e3ec4ce904c76593…
v3.3/protocol.json	b82da5987bb555443c9006e51e5a4d55…
v3.3/calibration-grok.json	4671f64ed9acbab677a4dd6ff9c3af44…
v3.4/calibration-muse.json	d9b990248a5a553a2a5a7c68aadf128e…
api-equivalent-costs.json	7fa4c18e7c8045b74a9b06527d30c38a…

The protocol documents, controls, quirk-key format, runner and operator wrapper are in the VulcanBench repository under docs/judging and harness. The quirk answer keys themselves describe hidden-test behaviour and are not published.

Appendix. Code quality per task at Max effort

Task	Astra CQ	Fable CQ	Difference	Astra combined	Fable combined
blendcore	74.7	82.4	+7.7	88.46	92.87
cellarcore	72.6	78.8	+6.3	87.32	91.19
codeccore	82.5	86.2	+3.7	91.92	93.92
depotcore	72.4	66.1	-6.4	87.29	86.74
freightcore	82.6	90.2	+7.6	91.83	95.31
granarycore	69.1	78.1	+9.1	87.01	89.53
hedgecore	78.6	86.3	+7.7	90.52	94.00
lodgecore	74.6	85.0	+10.4	87.94	92.96
matchcore-order-book	78.0	85.6	+7.6	89.34	89.50
metercore	75.5	85.2	+9.8	88.51	93.71
pacecore	82.6	85.9	+3.4	91.90	93.93
paddockcore	66.6	77.3	+10.7	85.34	87.88
payrollcore	72.9	84.0	+11.1	88.73	91.35
qlite-store	81.1	81.8	+0.8	91.30	92.45
queuecore	77.4	76.4	-1.0	90.21	90.82
quotacore	78.0	81.1	+3.0	90.49	92.49
reflowcore	81.1	76.6	-4.4	90.62	91.03
schedcore	73.4	82.3	+8.9	88.84	90.63
settlecore	75.8	88.6	+12.9	88.12	94.77
snapcore	77.1	85.7	+8.6	90.02	90.10
stampcore	75.5	88.0	+12.4	89.69	94.65
tallycore	79.0	83.9	+4.9	90.66	91.86
vaultcore	73.3	80.5	+7.3	87.93	90.56

Per-task values from runs.json; Max effort for both models. Task names are shortened for width.

VulcanBench

September 2026

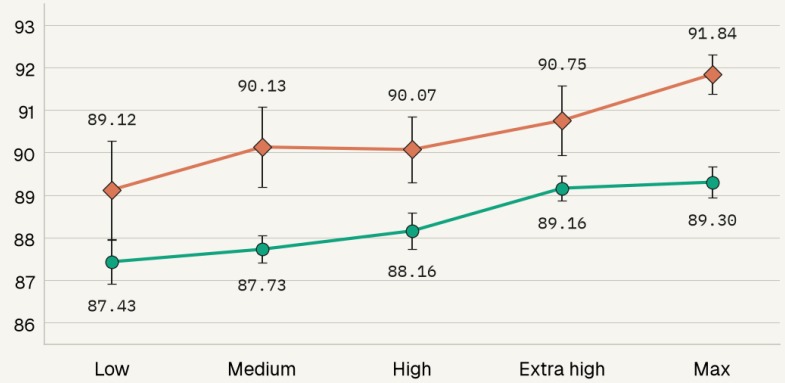
VulcanBench-SWE v4: Astra vs. Fable 5.1

Combined score and runtime at every effort level. 23 tasks per model per effort. Code quality judged by Muse Spark 1.3 and Grok 4.6.

● GPT-6 Astra · Codex ◆ Fable 5.1 · Claude Code n=23 at every effort

Combined score

/100 · higher is better · focused scale



Mean runtime

Minutes per task · lower is better

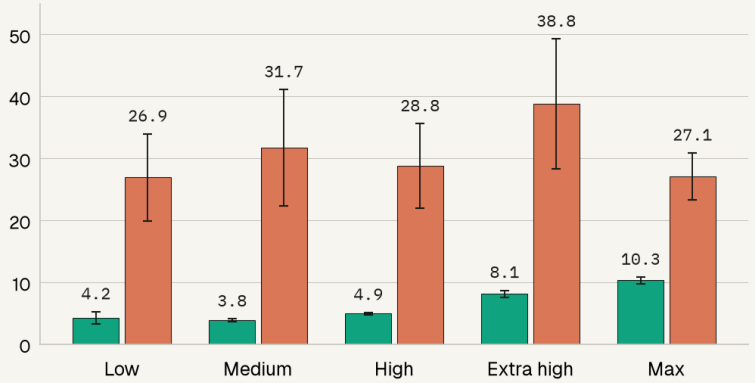


Table 1 | Code quality at each model's best effort

Component	GPT-6 Astra	Fable 5.1	Difference Fable minus Astra
Code quality	**76.26**	**82.43**	**+6.17**
Human readability	69.1	80.8	+11.7
Maintainability	79.8	84.5	+4.7
Intent recovery	81.1	81.8	+0.8
Rated by Muse Spark 1.3	73.6	83.4	+9.9
Rated by Grok 4.6	75.4	81.9	+6.5
Standard error of Code quality	0.88	1.10	

VulcanBench

September 2026

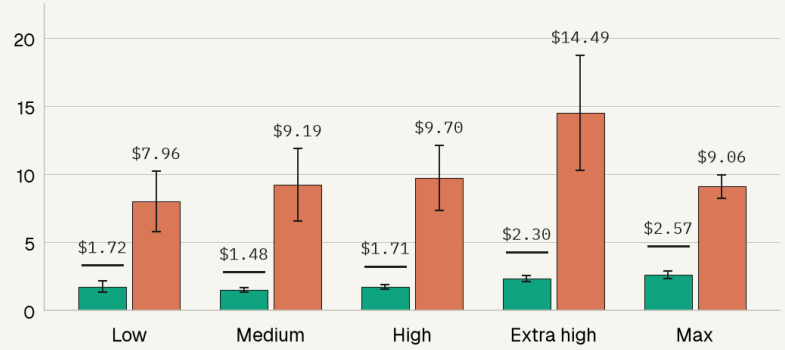
VulcanBench-SWE v4: Astra vs. Fable 5.1

API-equivalent cost and token use at every effort level. Same 230 runs as the score card; solver inference only, judging excluded.

● **GPT-6 Astra · Codex** ◆ **Fable 5.1 · Claude Code** n=23 at every effort

API-equivalent cost

USD per task at list prices · lower is better · ticks: Astra long-context bound



Tokens per task

Millions of raw tokens, cache reads included · lower is better

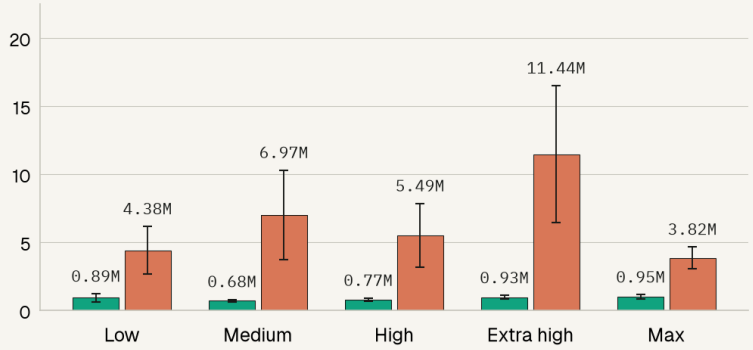


Table 1 | Cost and tokens at each effort level

Effort	Astra $/task	Fable $/task API-equivalent, list prices	Fable ÷ Astra cost ratio	Astra tokens	Fable tokens raw tokens per task, millions
Low	$1.72	$7.96	4.6×	0.89M	4.38M
Medium	$1.48	$9.19	6.2×	0.68M	6.97M
High	$1.71	$9.70	5.7×	0.77M	5.49M
Extra high	$2.30	$14.49	6.3×	0.93M	11.44M
Max	$2.57	$9.06	3.5×	0.95M	3.82M
Full sweep, 115 runs each	**$225**	**$1,159**	**5.2×**	**97M**	**738M**

USD at list prices checked 2026-09-06, cache-aware, solver inference only; subscription bills differ. Solver time for the sweep: Astra 12.0 h, Fable 58.7 h.

Astra request sizes are not logged: ticks mark a long-context upper bound ($419 for the sweep against $225 central). 11 of 115 Fable runs include Opus fallback usage, priced at its own rate.

Whiskers are one task standard error. Tokens are raw solver totals including cache reads; Fable's cached prefixes make its token count a poor cost proxy.