

The operating policy

Codex presents reasoning as a dial. Turning that dial upward sounds safer: give the model more time, get better work. It also means longer waits and greater usage. Without evidence, a team can spend more on every task without knowing whether the result changed.

The benchmark was built to answer an everyday business question: **which Codex setting should a team use before it knows whether a task is hard?**

Our answer is a starting policy, not a promise:

Situation	Start here	Why
Ordinary bounded work with clear checks	Any available GPT-5.6 model at medium	Medium matched the higher settings on every valid common task while using less time and fewer tokens.
Difficult evaluation, code review, or adversarial checking	Sol high as a scout	Sol high is a reasonable hypothesis for careful evaluation, but it did not beat medium in this benchmark.
A task where deeper reasoning itself is the unresolved decision	Compare the smallest relevant pair	Use xhigh or max only when the outcome could change a real routing choice.
A task that divides into independent workstreams	Consider desktop Ultra separately	Our controlled Ultra runs disabled delegation, so they did not test the complete desktop experience.
Responsive frontend work	Choose no model from this benchmark	The original responsive-page test had a defective visual check. Use deterministic desktop and mobile verification instead.

For engineering and operations leaders, the value is straightforward. Starting at medium can reduce waiting time and usage without giving up measured quality on work like the tasks we tested. The savings become meaningful when a team runs many agent tasks. The policy also prevents the opposite mistake: using a cheaper setting on a high-cost failure where more scrutiny may be justified.

The model setting is only one part of the decision. A fast answer that fails the acceptance checks is expensive. A slower answer that adds no useful quality is also

expensive. The unit that matters is **cost per accepted result**, including review and correction, not raw token use alone.

What we tested

We compared three GPT-5.6 models available through Codex with ChatGPT subscription access:

- **Luna**
- **Terra**
- **Sol**

For a fair common comparison, every model received the same frozen task, prompt, tools, fixture, and acceptance checks at five reasoning levels:

Desktop wording	Runtime value	Plain-language meaning in this study
Light	low	Less reasoning, usually faster
Medium	medium	Balanced starting level
High	high	More reasoning for harder work
Extra High	xhigh	A larger reasoning allowance
Not shown in the Sol desktop picker	max	The deepest single-agent setting exposed by the tested runtime
Ultra	ultra	Maximum reasoning plus automatic task delegation in the desktop product

The runtime exposed `max` for Luna, Terra, and Sol, even though the Sol desktop picker did not show it. Ultra was exposed only for Terra and Sol, so we kept it out of the common ranking and evaluated it separately.

This distinction matters. `max` and `ultra` are not two names for the same thing. Max describes reasoning depth for one agent. Ultra combines maximum reasoning with automatic delegation. To keep tool access equal, our benchmark disabled delegation. The Ultra results therefore describe a controlled single-agent profile, not the full desktop Ultra experience.

The two tracks

The benchmark had two separate tracks.

The **environment-specific track** used frozen tasks shaped like work in our own operating environment. It covered precise evidence retrieval, ambiguous diagnosis, safe work in a shared Git checkout, bounded implementation, code review, failure recovery, policy synthesis, and frontend work.

The **public-generalizable track** used self-contained synthetic fixtures. It covered familiar coding-agent work such as a mechanical edit, localized bug fix, multi-file feature, repository exploration, code review, test generation, operational diagnosis, and responsive frontend implementation.

The private track asks whether a route is useful in our environment. The public track asks whether another user can reproduce the task without our private repositories or knowledge. Neither track represents every form of software work.

This is a different question from our [cross-provider visual benchmark](#). That study compared how six frontier models shaped information and decisions for one fictional product. This study compares only OpenAI's GPT-5.6 Codex family and asks how model choice and reasoning effort affect accepted coding work.

How the 294 runs were counted

- 240 common first-pass rows: 16 tasks across the three models and five common reasoning levels.
- 24 targeted repeats: only cells that could change a practical decision were repeated.
- 30 Ultra rows: Terra and Sol on a separate 15-task appendix.

Every decision-bearing run recorded the requested and observed model and reasoning level. There were no silent model or effort substitutions in the retained rows. The runs used Codex app CLI 0.144.2, ChatGPT subscription OAuth, a pinned tool profile, isolated worker state, and serial execution.

What the results actually say

The first result looked simple. Sol led the original common pass. The careful result was different.

Two task families had invalid evaluation rules and had to be removed from model routing conclusions. After excluding those rows, the valid common view was:

Model	Accepted valid rows	What this supports
Luna	69/70	Strong performance on these tasks, with one repeated low-effort failure
Terra	70/70	Complete acceptance on these tasks
Sol	70/70	Complete acceptance on these tasks

Terra and Sol tied. Luna differed only on one exact low-effort implementation task. That is not enough to name a general winner.

The more important finding is that the suite was mostly too easy for these models. When several options pass almost everything, the benchmark confirms a competence floor but stops helping with selection. A perfect score can mean the model is excellent. It can also mean the test reached its ceiling.

Medium matched the higher settings

Across the valid common rows, reasoning effort produced this result:

Reasoning level	Accepted Rows		Time per accepted Input tokens per accepted result	per accepted result
Low	41	42	31.88 seconds	53,024
Medium	42	42	35.61 seconds	54,478
High	42	42	37.54 seconds	58,591
Extra High	42	42	47.17 seconds	66,298
Max	42	42	57.11 seconds	72,143

Medium, high, xhigh, and max all passed 42/42. Medium used less time and fewer input tokens than the higher settings. On this exact task set, paying for more reasoning did not buy more accepted work.

That does **not** prove that medium reasons as deeply as max or that hard unseen tasks will behave the same way. It establishes a narrower point: the additional reasoning allowance did not improve the outcome that these acceptance checks could observe.

Low was fastest and used the fewest tokens, but it failed one task. That failure was not a one-off. Luna low failed the exact multi-file implementation task on all three attempts. Terra low passed it once in three attempts, while Sol low passed twice.

Low may be attractive for speed, but the evidence does not support using it blindly for that exact task shape.

Ultra cost more without improving accepted quality

Ultra could not be compared across all three models because Luna did not expose it. We therefore compared Terra and Sol against their own high and max settings on the same valid tasks.

Model and setting	Accepted	Mean wall time	Input tokens
Terra high	14/14	34.69 seconds	779,434
Terra max	14/14	56.73 seconds	1,070,116
Terra ultra	14/14	73.65 seconds	1,465,027
Sol high	14/14	37.09 seconds	779,752
Sol max	14/14	58.49 seconds	943,980
Sol ultra	14/14	86.39 seconds	1,590,421

All six cells passed every valid matched task. Ultra took longer and consumed more input tokens. It did not earn that extra cost on this suite.

The untested possibility is important: automatic delegation may help when a large task genuinely divides into independent streams. Our controlled profile did not test that product experience. The valid conclusion is therefore not "Ultra never helps." It is "Ultra did not help on these controlled tasks."

Two benchmark failures changed the story

The benchmark itself failed in two useful ways. Correcting those failures changed the apparent result, which is why they belong in the main story.

One task asked for five stages but graded four

The public workflow task called `PUB4` required five stages in its written task card. Its automated check expected four semantic stages. A model could follow the instructions and still fail the grader.

All 15 `PUB4` quality rows were excluded. We did not quietly rewrite the scores. The historical outputs remain preserved, but they cannot support a model or reasoning-level claim.

The responsive-page check confused valid layouts with failures

The original responsive frontend task, `PUB8`, tried to infer a two-column desktop layout by inspecting direct children in the page structure. A valid implementation with an extra semantic wrapper could render correctly and still fail. Padding could also be mistaken for a column.

A later sensitivity check changed the apparent aggregate result from $7/33$ under the old rule to $28/33$ under a prototype repair. That large change shows that the checker was driving much of the result.

All 35 historical `PUB8` rows were quarantined from routing evidence. We built a separately versioned replacement, `PUB8V2`, with explicit semantic hooks and independent checks for desktop structure, mobile stacking, overflow, readable content, and implementation evidence. It passed 14 calibration cases and one independently constructed instrumentation smoke. No model has been measured on it yet.

The business lesson is broader than frontend testing. An automated score is useful only when the check measures the property the decision depends on. A precise but invalid checker can produce confident, repeatable, and wrong conclusions.

What each audience should take away

Engineering leaders

Start bounded work at medium and require tests, diffs, screenshots, or other external proof. Reserve higher reasoning for tasks where failure is costly or where medium has already shown a concrete limitation.

This recommendation applies to work resembling the tested tasks. It does not establish a default for security-critical systems, production migrations, large-scale architecture, or unfamiliar repositories with weak tests.

Product and operations teams

Do not equate a higher reasoning label with a better business outcome. Measure the time and human effort required to reach an accepted result. A cheaper first attempt that creates review work may cost more overall. A slower setting that changes nothing may also be waste.

The benchmark measured task acceptance, latency, and exposed token use. It did not measure long-term maintenance cost, customer impact, or the value of a production deployment.

Developers using Codex every day

Use medium as the starting point, then escalate deliberately:

1. Write down what must be true when the task is finished.
2. Run the task at medium.
3. Check the result with evidence outside the model's own explanation.
4. Move to high or above only when the first attempt exposes a real reasoning gap or the cost of failure justifies more scrutiny.

For frontend work, inspect deterministic desktop and mobile screenshots. Do not use the historical `PUB8` results to choose a model.

Researchers and benchmark builders

Separate model failure from test failure. Preserve raw results, but version a repaired task instead of rewriting history. Repeat only cells that could change a decision. When every lane passes, build a more discriminating task only if a real routing question needs it.

These practices should generalize to many model evaluations. The exact scores and routing advice do not.

What we still do not know

This study leaves several questions open:

- whether Sol high outperforms medium on naturally difficult repository work
- whether full desktop Ultra delegation helps tasks that split cleanly into independent streams
- which model is best for responsive frontend implementation under the new `PUB8V2` check
- whether the same behavior appears through paid API access
- whether future Codex versions, tools, or model updates change the result
- how the models compare on long-lived production work where maintainability appears weeks later

Those unknowns are not defects to hide. They define where the evidence stops.

The decision in one minute

1. **Start at medium.** It matched every higher common reasoning level on the valid tasks while using less time and fewer tokens.
2. **Do not name a universal model winner.** Luna, Terra, and Sol were nearly indistinguishable after invalid tasks were removed.
3. **Use Sol high as a scout, not a default winner.** It is a reasonable lane for difficult evaluation and review, but it did not beat medium here.
4. **Gate xhigh, max, and ultra.** Use them when a named task risk or blocked decision can justify the additional cost.
5. **Verify the work externally.** Tests, browser checks, and evidence matter more than the label in the model picker.
6. **Audit the benchmark when a result looks surprising.** Two defective checks materially changed this study's first interpretation.

The next useful experiment is not another broad sweep. It is one naturally occurring hard task with a clear contract, a preselected model and effort, and an outcome that can be checked independently. Only then can a new measured row resolve a real decision instead of adding volume.

What we learned when we tested Astra next

In a separate September 2026 follow-up, GPT-6 Astra at low, medium and high completed two bounded tasks: implementing an in-memory storage adapter and diagnosing paired failure reports. **All six submissions passed** deterministic checks and a fresh-context, model-assisted blind review. That review was not independent human verification.

There was only one attempt per task and setting. A tie this small does not prove that the settings are equally capable; the tasks may simply be unable to distinguish them. These were different tasks and a different runtime from the GPT-5.6 study. The six observations are separate from the original 294 runs and do not establish an Astra-versus-GPT-5.6 ranking or a general low-effort default.

The more actionable finding came from the execution trace. The high-effort implementation attempt tried extra verification, but the shell blocked it before the checks could run. Those assertions did not fail: they never ran. A later non-measured

smoke check verified a temporary-file configuration repair in our local runner. It was an environment repair, not another model success.

Before paying for more reasoning, check what actually failed. A broken tool or test needs repair, not a higher setting. We stopped the six-row comparison without repeats, as planned. Further testing should address a specific unresolved task, not turn this small tie into a universal claim.

Method and scope

The benchmark was completed in July 2026 through Codex ChatGPT-subscription OAuth using Codex app CLI 0.144.2. It used frozen private and public task fixtures, a common core-tool profile, isolated execution state, and requested versus observed identity checks. The retained evidence contains 294 measured runs. Fifty historical rows are not current routing evidence: 15 from the contradictory `PUB4` task and 35 from the quarantined `PUB8` visual check.

The corrected common policy view contains 210 eligible rows. It applies only to the tested model versions, reasoning levels, tools, fixtures, and July 2026 runtime. It does not establish paid API behavior, universal coding ability, production safety, future model behavior, or a global model ranking.

The public benchmark package has been prepared for reproducibility but has not been published. Private fixtures and raw traces remain local because they can contain environment-specific paths and context.

Product semantics were checked against OpenAI's [GPT-5.6 model guidance](#) and [GPT-5.6 announcement](#), accessed 1 August 2026. OpenAI describes `max` as additional time for exploration, checking, and revision. It describes `ultra` as a multi-agent setting that coordinates four agents in parallel by default. Product availability and behavior can change, so current users should check the latest documentation and their own runtime.