

OpenCode Go vs Command Code GOAT: Which \$10 AI Coding Plan Wins in 2026?

TechRounder PDF Edition

Live article:

<https://www.techrounder.com/ai/opencode-go-vs-command-code-goat-which-10-ai-coding-plan-wins-in-2026/>

By Vipin PG | Published August 19, 2026 | Updated October 7, 2026 | Format: Comparison | 15 min read

Bottom line

OpenCode Go and Command Code GOAT are both \$10-a-month AI coding subscriptions, but they're not identical. OpenCode Go gives you \$60 of usage across 19 open-source coding models and works with any editor through an open, provider-neutral agent. Command Code GOAT gives you \$70 of usage across 37 open and closed models, including Grok, GPT, and Gemini, inside its own built-in coding harness, and yes, it does include full API access. Command Code GOAT is the better raw-value pick for most models, while OpenCode Go wins on flexibility and on a few specific models like Qwen and MiMo.

If you've spent any time in coding subreddits or dev Twitter lately, you've probably seen two names being thrown around a lot: OpenCode Go and Command Code GOAT. Both cost roughly \$10 a month. Both promise to hand you a stack of powerful AI coding models for less than the price of a couple of coffees. And both have loud, opinionated fan bases who insist their pick is the smarter buy. We dug through the official documentation of both platforms, checked their current pricing pages, and went through real developer discussions to see how these two actually hold up against each other in practice. This guide walks through everything you need to know, pricing, models, usage limits, API access, privacy, and how real users feel about each one, in plain language, so you can decide which plan (if either) is right for you. A quick heads-up before we dive in: subscription AI plans like these change their pricing, limits, and model line-ups fairly often. The details below were checked directly against each company's official documentation, but it's always worth a quick look at their pricing pages before you subscribe, just in case something has shifted.

The Short Answer

If you don't have time to read the whole thing, here's the gist: Command Code GOAT gives you more raw usage for your \$10, a bigger model catalog, a bigger monthly credit pool, and access to a few closed models like Grok and GPT that OpenCode Go doesn't offer. OpenCode Go wins on openness and flexibility, it's built on an open-source coding agent, it plugs into almost any editor or tool you already use, and for a handful of specific models (like Qwen and MiMo) it actually gives you more usage than GOAT does. Neither one is really built for someone who wants to run AI agents non-stop, all day, every day. If that's your use case, you'll outgrow both plans quickly and should look at a heavier \$20+ tier instead. But for everyday coding help, writing functions, fixing bugs, generating tests, scaffolding new features, either plan is genuinely good value, and the "right" one depends on which models you actually plan to use.

What Exactly Are These Two Plans?

Before comparing numbers, it helps to understand what you're actually buying, because these aren't quite the same kind of product.

OpenCode Go, in plain English

OpenCode is a free, open-source coding agent, think of it as a smart assistant that lives in your terminal and can read your code, write new code, run commands, and fix bugs, all guided by an AI model. By itself, OpenCode doesn't come with any AI models built in. You have to connect it to one, whether that's your own Anthropic or OpenAI key, a local model, or anything else. OpenCode Go is a subscription that plugs directly into that same OpenCode agent (or into any other AI tool, really) and hands you access to a curated set of open-source coding models for a flat monthly fee. It's built by Anomaly, the team also known for the Serverless Stack (SST) project, so it has real open-source roots.

Command Code GOAT, in plain English

Command Code is a separate, terminal-based coding agent with its own built-in help, the software layer that decides how the AI reads your files, plans changes, runs tests, and reports back to you. The GOAT plan is Command Code's \$10 subscription tier, and it bundles a large pool of usage credits that work across a wide catalog of both open and closed AI models, all inside Command Code's own CLI (or through its API). Where OpenCode leans into being an open, swappable piece of a bigger toolchain, Command Code leans into being a complete, self-contained coding assistant with a heavily engineered help behind it.

Price and Value at a Glance

What you're comparing: Monthly price | OpenCode Go: \$5 for the first month, then \$10/month | Command Code GOAT: \$10/month, plus a small card processing fee

What you're comparing: Total included credit value | OpenCode Go: Up to \$60/month | Command Code GOAT: Up to \$70/month (can stretch past \$100 with active provider deals)

What you're comparing: 5-hour rolling usage limit | OpenCode Go: \$12 | Command Code GOAT: \$14

What you're comparing: 7-day (weekly) usage limit | OpenCode Go: \$30 | Command Code GOAT: \$35

What you're comparing: Number of models included | OpenCode Go: 19 open-source coding models | Command Code GOAT: 37 models (open and closed), out of 55+ across all Command Code plans

What you're comparing: API access included | OpenCode Go: Yes | Command Code GOAT: Yes

What you're comparing: Underlying agent | OpenCode Go: Open-source (OpenCode); works with any editor or AI agent | Command Code GOAT: Command Code's own proprietary CLI help

What you're comparing: One-click "enforce privacy" switch | OpenCode Go: Not available plan-wide; privacy varies by model | Command Code GOAT: Yes, via a single ZDR (zero data retention) flag

At first glance, GOAT looks like the clear winner just because \$70 is bigger than \$60. But that headline number is misleading on its own, because neither company hands out that value evenly across every model. We'll get into the real, model-by-model numbers a bit further down, they tell a much more interesting story.

Does Command Code GOAT Actually Have API Access? Clearing Up a Common Mix-Up

This is one of the most common points of confusion, so it's worth settling clearly: yes, the Command Code GOAT plan does include full API access. The idea that only OpenCode Go offers a usable API while GOAT is locked to its own app is outdated and incorrect. Here's where the confusion comes from. Command Code has a separate \$1 entry-level "Go" plan (not to be confused with OpenCode Go, the naming overlap is genuinely unfortunate) that does restrict API access. But once you move up to the \$10 GOAT tier, you get a proper, production-ready API with OpenAI-compatible and Anthropic-compatible endpoints, authenticated with a bearer token, and metered against the same GOAT credits you'd use in the CLI. Command Code's own GOAT plan documentation spells this out directly: every plan except that entry-level \$1 tier includes API access. In practical terms, this means you can point any OpenAI-compatible or Anthropic-compatible tool, a CI pipeline, a custom script, another coding agent entirely, at Command Code's Provider API and have it draw from your GOAT subscription, exactly the way OpenCode Go works as a provider inside OpenCode. If you want the technical details on request formats and authentication headers, Command Code's Provider API documentation covers it thoroughly. So if API access was the deciding factor in your comparison, you can cross it off the list, both plans genuinely offer it.

Every Model You Can Use on Each Plan

This is where the two plans really start to diverge. OpenCode Go sticks to a smaller, carefully tested roster. Command Code GOAT throws the doors open much wider.

OpenCode Go's model line-up

OpenCode Go currently gives you access to 19 models, all open-source or open-weight, spanning several major AI labs:

- DeepSeek: V4 Pro and V4 Flash
- Zhipu AI (GLM): GLM-5.3, GLM-5.2, and GLM-5.1
- Moonshot AI (Kimi): Kimi K3, Kimi K2.7 Code, and Kimi K2.6
- Alibaba (Qwen): Qwen3.8 Max, Qwen3.7 Max, Qwen3.7 Plus, and Qwen3.6 Plus
- MiniMax: M3 and M2.7
- Xiaomi (MiMo): MiMo-V2.5 and MiMo-V2.5-Pro
- Tencent: Hy3
- Two closed-model exceptions: Grok 4.5 and GPT 5.6 Luna

The full, current list, along with any changes OpenCode makes as it tests new models, is always kept up to date on OpenCode's official Go documentation page.

Command Code GOAT's model line-up

GOAT includes 37 models out of Command Code's full 55-model catalog (the rest need the pricier Pro or Max plans). Interestingly, nearly every model on OpenCode Go's list also shows up here, plus quite a few extras:

- Everything OpenCode Go offers, the same DeepSeek, GLM, Kimi, Qwen, MiniMax, MiMo, Tencent, Grok, and GPT models
- Google: Gemini 3.7 Flash, currently discounted 50%
- xAI: Grok 4.6, in addition to Grok 4.5
- OpenAI: GPT-5.6 Sol, in addition to GPT-5.6 Luna
- NVIDIA: Nemotron 3 Ultra

- A handful of smaller/newer labs: Step (3.7 Flash, 3.5 Flash), Inkling and Inkling Small, and Laguna S 2.1 (currently free while capacity lasts)
- "Speed" variants like GLM-5.2 Fast and Kimi K2.7 Code HighSpeed for lower-latency responses

So the practical takeaway is simple: if model variety and access to closed-source options matter to you, GOAT is the broader net. If you only care about a handful of open models and don't need the extras, the two plans overlap almost completely, which makes the actual usage limits (below) the more important factor.

How the Usage Limits Actually Work

Neither plan gives you a flat "X requests per day" number. Instead, both measure your usage in dollar value, based on how much the underlying model actually costs to run. That's a slightly unusual system, so here's what it means in practice. Every model has its own per-token price. A cheap, efficient model like MiMo-V2.5 costs a fraction of a cent per request, so your monthly allowance stretches to well over 100,000 requests. A more expensive, "smarter" model like Kimi K3 costs far more per request, so the same dollar allowance only buys you a few hundred requests a month. This is why you'll sometimes see wildly different "requests per month" numbers for models on the exact same plan, it's not a bug, it's just reflecting real compute cost. Both platforms structure their limits around three time windows:

- A 5-hour rolling window, the closest thing either plan has to an "hourly" limit. It's not a fixed clock-hour; it starts counting from your first request and resets five hours later.
- A 7-day (weekly) rolling window, smooths out usage across a work week so one busy day doesn't wipe out your whole month.
- A monthly ceiling, the hard cap for your billing cycle, resetting when you're billed again.

OpenCode Go's windows are \$12 (5-hour), \$30 (weekly), and \$60 (monthly). Command Code GOAT's are \$14 (5-hour), \$35 (weekly), and \$70 (monthly). If you run out mid-session on either plan, you're not completely stuck: OpenCode Go lets you fall back to free models or top up a separate pay-as-you-go balance, while Command Code GOAT lets you buy extra pay-as-you-go credits on the spot, which roll over and never expire.

Real Numbers: How Far Does Your \$10 Actually Go?

This is the part most comparisons skip, and it's honestly the most useful one. Since both companies publish estimated request counts for each model (based on a typical coding-agent request pattern), we can line them up side by side. The table below shows estimated requests per month for models that appear on both plans.

Model: DeepSeek V4 Flash | OpenCode Go (est. requests/month): 37,800 | Command Code GOAT (est. requests/month): 91,200 | Better value on: GOAT (about 2.4×more)

Model: DeepSeek V4 Pro | OpenCode Go (est. requests/month): 5,200 | Command Code GOAT (est. requests/month): 9,880 | Better value on: GOAT (about 1.9×more)

Model: MiMo V2.5 | OpenCode Go (est. requests/month): 150,400 | Command Code GOAT (est. requests/month): 97,400 | Better value on: OpenCode Go

Model: MiMo V2.5 Pro | OpenCode Go (est. requests/month): 16,300 | Command Code GOAT (est. requests/month): 28,500 | Better value on: GOAT (about 1.75×more)

Model: Qwen 3.7 Plus | OpenCode Go (est. requests/month): 21,600 | Command Code GOAT (est. requests/month): 7,110 | Better value on: OpenCode Go (about 3×more)

Model: Qwen 3.6 Plus | OpenCode Go (est. requests/month): 16,300 | Command Code GOAT (est. requests/month): 5,500 | Better value on: OpenCode Go (about 3×more)

Model: Kimi K2.7 Code | OpenCode Go (est. requests/month): 6,750 | Command Code GOAT (est. requests/month): 5,420 | Better value on: OpenCode Go

Model: MiniMax M3 | OpenCode Go (est. requests/month): 16,000 | Command Code GOAT (est. requests/month): 13,900 | Better value on: OpenCode Go

Model: Kimi K3 | OpenCode Go (est. requests/month): 490 | Command Code GOAT (est. requests/month): 980 | Better value on: GOAT (about 2×more)

Model: GLM-5.2 | OpenCode Go (est. requests/month): 4,300 | Command Code GOAT (est. requests/month): 4,740 | Better value on: GOAT (slightly)

Model: GPT-5.6 Luna | OpenCode Go (est. requests/month): 10,250 | Command Code GOAT (est. requests/month): 14,800 | Better value on: GOAT (about 1.4×more)

Model: Tencent Hy3 | OpenCode Go (est. requests/month): 21,500 | Command Code GOAT (est. requests/month): 35,400 | Better value on: GOAT (about 1.65×more)

Notice the pattern: it's genuinely a mixed bag. Command Code GOAT has a clear edge on DeepSeek, Kimi K3, GPT-5.6 Luna, and Tencent Hy3. OpenCode Go pulls ahead on Qwen (by a wide margin, roughly 3×more requests) and comes out slightly ahead on MiMo V2.5, Kimi K2.7 Code, and MiniMax M3. This is exactly why comparing "\$60 vs \$70" on its own doesn't tell the real story, the smarter approach is to check the model you'll actually use the most, and pick the plan that stretches furthest for that specific model. One more thing worth knowing: DeepSeek's pricing (on both platforms) changes depending on the time of day. Peak hours are 1-4 AM and 6-10 AM UTC, and requests cost roughly double during those windows. If your work schedule lets you shift heavy DeepSeek usage outside those hours, your monthly allowance will go noticeably further.

Privacy and Data: What Happens to Your Code

If you're working on client code, proprietary business logic, or anything under an NDA, this section matters more than the pricing table does. OpenCode Go publishes a model-by-model breakdown rather than one blanket privacy promise, and it's genuinely useful for that reason. Most of the open models on Go, the GLM, Kimi, Qwen, MiniMax, MiMo, and DeepSeek routes, are marked as not used for training, with zero-day data retention. The two closed-model exceptions, Grok 4.5 and GPT 5.6 Luna, are also not used for training, but their upstream providers keep abuse-monitoring logs for up to 30 days. It's worth noting that DeepSeek's zero-retention agreement is renewed monthly rather than being a permanent guarantee, so it's the kind of detail worth double-checking every so often. Command Code takes a different approach: instead of a model-by-model table, it offers a single switch. Setting the environment variable 'CMD_ZDR=1' in the CLI (or sending the 'x-cmd-zdr: 1' header through the API) forces every request onto a zero-data-retention route. If a particular model doesn't have a compliant route available at that moment, the request simply fails instead of silently sending your code through a route that might retain it. Without that flag switched on, Command Code's general privacy policy allows request content to be retained for up to 30 days. In short: OpenCode Go gives you more transparency about exactly what's happening per model, while Command Code GOAT gives you a simpler, blunter "just make everything private" switch. Neither is strictly better, it depends on whether you want granular detail or a one-click guarantee.

The Developer Experience: help, Editors, and Workflow

Beyond the numbers, the two products genuinely feel different to actually use. OpenCode's biggest strength is that it isn't trying to lock you into anything. Because the underlying agent is open-source, you can mix OpenCode Go with your own API keys from other providers, run local models alongside it, connect MCP servers, and use it inside VS Code and VS Code-derived editors (Cursor, Windsurf, VSCodium) through its editor integration, or inside any editor that supports the Agent Client Protocol (ACP). It also publishes a genuinely detailed list of supported programming languages through its Language Server Protocol integrations, which is more transparency than most competitors offer. Command Code leans much harder into being a complete, opinionated package. Its CLI includes a dedicated planning/review stage before it touches your code, session checkpoints you can roll back to, background tasks, git worktree support, and a feature called "Taste" that learns your coding preferences and style over time so you spend less effort repeating instructions. It officially supports VS Code, Cursor, and Windsurf, and the company has said it plans to open-source the CLI itself down the line, though as with any roadmap item, that's worth checking on before you factor it into your decision. If you like the idea of one tool that does everything out of the box, Command Code's help will likely feel more complete. If you'd rather keep full control over your toolchain and treat the AI model as just one swappable piece, OpenCode's open architecture is the more natural fit.

What Real Developers Are Saying

Numbers on a pricing page only tell half the story, so it's worth looking at how people actually feel after using these plans day to day. The most common praise for OpenCode Go is that it's excellent value when AI is used as a coding assistant rather than an all-day autonomous worker, plenty of developers report it comfortably handles routine tasks like scaffolding, test writing, and everyday bug fixes. The most common complaint is the opposite: heavy, all-day agentic sessions can chew through the 5-hour or weekly limit faster than the "\$10/month" framing suggests, especially on pricier models like Kimi or Grok. There have also been periods where OpenCode adjusted DeepSeek's allowances, and some users who were leaning heavily on DeepSeek felt that change sharply, which pushed a portion of that crowd to look at Command Code GOAT instead. On the Command Code GOAT side, the recurring theme is that its limits simply feel more generous in daily use, particularly for DeepSeek-heavy workflows, and its more structured help (the plan/review step, checkpoints, and built-in test/debug workflows) appeals to developers who want more guardrails around what the AI is allowed to do. The most common criticism is around trust and marketing, some users have pushed back on how the "7x your money" framing is presented, feeling it oversells the everyday experience even while agreeing the underlying value is genuinely solid. A smaller number of users have also noted that, because it's a newer plan, there's simply less long-term, independent track record to lean on compared to OpenCode's larger and older community. A pattern worth mentioning: a fair number of developers don't treat this as an either/or decision at all. Some run OpenCode Go for everyday, flexible work and keep Command Code GOAT in their back pocket specifically for DeepSeek-heavy or high-volume sprints, since \$20 total across both plans is still cheaper than most single \$20+ subscriptions.

Other \$10-ish Plans Worth Knowing About

OpenCode Go and Command Code GOAT aren't the only games in town at this price point. A few alternatives are worth a quick look before you commit:

Plan: GitHub Copilot Pro | Price: \$10/month | What makes it different: Unlimited inline code completions plus a smaller monthly credit pool (around \$15) for chat, agent mode, and code review. Best if you live inside VS Code, Visual Studio, or JetBrains and mainly want autocomplete with occasional AI chat, rather than a standalone multi-model gateway. Full details are on GitHub's own Copilot pricing page.

Plan: Zed Pro | Price: \$10/month | What makes it different: Unlimited AI-powered edit predictions inside the Zed code editor, plus a small monthly token allowance for chat and agent features. A strong pick if you're open to switching your entire editor, but it isn't a portable API-style plan the way OpenCode Go and Command Code GOAT are.

Plan: Free agents + your own API key (Cline, Aider, Kilo Code) | Price: Variable, pay-as-you-go | What makes it different: Maximum control, you pick the model, you pay only for what you use, no subscription markup. The trade-off is no built-in spending cap and no subsidized bulk pricing, so costs can climb fast if you're not watching usage closely.

Most other dedicated coding-model subscriptions, think GLM's own coding plan or MiniMax's token plan, now start closer to \$18-\$20/month, which is part of why OpenCode Go and Command Code GOAT stand out as unusually aggressive at the \$10 mark.

So, Which One Should You Actually Choose?

Here's a practical way to decide, based on how you actually work rather than the marketing copy:

- Pick OpenCode Go if: you want an open-source, provider-neutral tool you can bend to your own workflow; you plan to lean on Qwen, MiniMax, or MiMo models; you already use other AI provider keys and want to combine everything under one agent; or you want the clearest, most transparent privacy documentation before choosing a model.
- Pick Command Code GOAT if: you want the broadest model catalog for the same \$10, including a few closed-source extras; your work leans heavily on DeepSeek, Kimi K3, GPT-5.6, or Tencent Hy3; you'd rather have a fully built-out help with planning, checkpoints, and test workflows out of the box; or you want a single, simple switch to enforce zero data retention on every request.
- Consider both if: your budget allows \$20 total and your work genuinely spans multiple model families, plenty of developers do exactly this and treat the two subscriptions as complementary rather than competing.
- Consider neither if: you're running autonomous agents non-stop throughout the day. Both plans are built around moderate, iterative coding sessions, and heavy round-the-clock use will burn through even the generous 5-hour and weekly windows quickly. In that case, a higher \$20+ tier from either provider (or a pay-as-you-go API budget) will serve you better.

Frequently Asked Questions

Does OpenCode Go work with any coding tool, or only with OpenCode? It's designed to plug into OpenCode, but because it exposes standard OpenAI-compatible and Anthropic-compatible API endpoints, it can technically be used with any tool that speaks those formats. Can I use Command Code GOAT outside of the Command Code CLI? Yes. The GOAT plan includes full Provider API access, so you can call it from your own scripts, CI pipelines, or other agents, not just the official CLI. Which plan gives more usage overall? Command Code GOAT's headline monthly value (\$70) is higher than OpenCode Go's (\$60), but the actual advantage flips depending on the model, OpenCode Go pulls ahead on Qwen and MiMo, while GOAT pulls ahead on DeepSeek, Kimi K3, and GPT-5.6. Is either plan good enough for full-day, non-stop AI coding? Not really. Both are built around the assumption of moderate, assistant-style usage. Continuous, all-day autonomous agent work will exhaust either plan's rolling limits faster than a \$10/month price tag implies. Do I need to worry about my code being used to train these AI models? Most models on both platforms are documented as not used for training. OpenCode Go publishes a detailed model-by-model retention table, while Command Code offers a single enforceable zero-data-retention flag. It's worth checking the current documentation for the specific model you plan to use, since retention policies can change. Can I switch between plans easily? Yes, both are simple monthly subscriptions you can cancel at any time, there's no long-term contract locking you in on either side. Whichever way you lean, the good news is that at \$10 a month, trying one out and switching later if it doesn't fit your workflow isn't a big financial risk. Given how quickly both platforms are adding models and adjusting limits, it's worth revisiting this comparison every few months to make sure you're still on the better deal for how you actually code.

References

1. commandcode.ai - docs / plans - <https://commandcode.ai/docs/plans/goat>
2. commandcode.ai - docs / provider - <https://commandcode.ai/docs/provider>
3. opencode.ai - docs / go - <https://opencode.ai/docs/go/>
4. github.com - features / copilot - <https://github.com/features/copilot/plans>