AI STACK SIGNAL

POSTED NOV 1, 2026

FREE PLAIN-ENGLISH GUIDE · ONE DEVELOPER'S REAL AI CODING SETUP

YOUR MOST EXPENSIVE AI SHOULDN'T BE THE ONE TYPING THE CODE

Developer Aayush Garg split his coding work between two AI models inside OpenCode, a free, open source coding agent. A strong paid model acts as the **boss**: it plans, writes the instructions, reads every change and runs the checks. A model running on his own hardware acts as the **worker**: it reads the code, makes the edits, runs the tests and fixes what breaks. This guide explains the setup in plain words, shows what makes it possible, lists his own warnings, and ends with our tips for trying a smaller version.

Boss

GPT-6.1 Sol

Plans, briefs, reads the diffs, runs the checks. Does not write the code.

Worker

GLM-5.3-Flash, self-hosted

Explores the code, edits, runs tests, fixes failures.

Key

One model per agent

OpenCode lets every agent use its own model, even from a different provider.

+

Try a smaller version

Our tips, his caveats, a simplified config, and every source.

01 THE IDEA IN PLAIN WORDS

PAY FOR JUDGMENT, NOT TYPING

AI coding tools spend most of their effort on **reading files, making edits and re-running tests**. That's a lot of tokens (the units AI usage is counted and billed in). Garg's idea: let the expensive model do only the parts that need judgment, and hand the busywork to a free model running on servers he already has. He did it "so that I can cut down my Claude Code and Codex usage."

The split

"Sol does the planning, writes the briefs, reads the diffs and runs the checks, while GLM reads the code, makes the edits, runs the tests and iterates on failures."

AAYUSH GARG · NOV 1, 2026

~60%
of tokens went to GLM on his measured toy task

70-75%
to GLM on bigger tasks, "In my experience" (not measured here)

4 GPUs
one H200 server runs the worker model

8-10
people can use that server at the same time

THE WORKER MODEL

"GLM-5.3-Flash is free, runs on a single 4-GPU H200 server." It's "an MoE model (320B total, only 18B active)," meaning only a small part of it works on each word, which keeps it fast. He serves it "with a 512K context window" and calls it "good enough for low and medium complexity tasks."

Why he can do this: "In my current job we have quite a few H200 GPU servers."

THE PAYOFF, IN HIS WORDS

"the plan limits (or the API bill) go to judgment and most of the tokens go to in-house GPUs."

And the boss earns its place: "Sol also catches things GLM misses."

No cost or speed numbers are given. The post doesn't publish dollars saved or tokens per second, so we don't either.

THE TOY EXAMPLE HE RAN

He asked for "a longest_word(text) function with tests" in a small Python project. The boss handed it to the worker, then ran "git diff, python3 -m unittest and git diff --check itself." It noticed a weak test where "hello and world tie at five letters," sent the work back, and "signed off once the 7 tests passed." Token count: "GLM with 104.6k tokens and Sol with 68.2k so even this toy task sent about 60% of the tokens to GLM."

02 WHO DOES WHAT THE FOUR AGENTS

In OpenCode an **agent** is an AI helper with its own instructions, permissions and model. A **subagent** is one the main agent can hand work to. Garg's setup has one boss and three helpers.

THE TEAM

AGENT	RUNS ON	JOB	IN HIS WORDS
orchestrator (the boss, default)	GPT-6.1 Sol	Plans the work, hands it out, checks the results, reports back. Never writes the code.	"Leads the work. Plans, delegates to explorer and implementer, verifies results, reports back. Does not implement."
explorer	GLM-5.3-Flash	Reads through the code to answer questions.	"Use `explorer` for any question that needs reading more than a couple of files."
implementer	GLM-5.3-Flash	Makes the edits, runs the tests, fixes failures.	Three rules: "stay in scope, stop and report both readings if the brief and the code disagree and never claim something passed that it did not run."
reviewer (optional)	GPT-6.1 Sol	A second opinion, only for risky work.	"Use `reviewer` only for changes that cut across several files or are risky."

HOUSE RULES FOR THE BOSS

"At most 3 GLM agents (explorer, implementer) at once; the GLM server is shared."

"After two failed rounds on one issue, ask me before fixing it yourself."

So the boss can't flood the shared server, and it can't quietly take over the typing (and the cost) when the worker struggles.

CLOSING THE BACK DOORS

OpenCode has built-in helpers called **general** and **explore**. He turned them off, because they "would run on the default model (Sol)," which would send cheap work to the expensive model.

Small background jobs, like naming a chat (title) and summarizing it (summary), also go to GLM.

✓ **Why the boss checks everything.** The worker is told never to claim a test passed that it didn't run, and the boss reruns the checks itself anyway. In the toy example, that's how the weak test got caught.

03 WHY OPENCODE

THE ONE FEATURE THAT MAKES IT WORK

"OpenCode is an open source AI coding agent. It's available as a terminal-based interface, desktop app, or IDE extension." For this setup, one feature matters most: **"every agent or subagent can have its own model from a different provider which is the one most important feature this setup needs."**

WHAT THE DOCS SAY

"Use the `model` config to override the model for this agent." Model names use "the format `provider/model-id`."

Leave it out and a subagent will "use the model of the primary agent that invoked the subagent," which is why he sets the model on every helper.

WHY NOT CLAUDE CODE OR CODEX

In Codex his subagent failed: "The 'coder-glm' model is not supported when using Codex with a ChatGPT account." In Claude Code, "a subagent can pick a different model but not a different endpoint." He names Pi as another option.

Bonus: "OpenCode can log in with a ChatGPT plan."

A SIMPLIFIED EXAMPLE CONFIG

SIMPLIFIED EXAMPLE

Modelled on his setup, cut down to show the idea. **This is not his exact file.** His lives at ~/.config/opencode/opencode.json, and his boss is set up in a separate orchestrator.md file. Check the OpenCode docs for every field before you use it.

```
{
  // the boss model, logged in through your plan
  "model": "openai/gpt-6.1-sol",
  "default_agent": "orchestrator",
  "provider": {
    "glm": {                                  // your own worker server
      "npm": "@ai-sdk/openai-compatible",
      "options": { "baseURL": "http://YOUR-SERVER/v1", "timeout": 900000 },
      "models": { "coder-glm": { "name": "GLM-5.3-Flash" } }
    }
  },
  "agent": {
    "general": { "disable": true },           // stop work falling back to the boss
    "explore": { "disable": true },
    "explorer":    { "mode": "subagent", "model": "glm/coder-glm" },
    "implementer": { "mode": "subagent", "model": "glm/coder-glm" },
    "reviewer":    { "mode": "subagent", "model": "openai/gpt-6.1-sol" }
  }
}
```

Each helper also needs a description and a prompt (its instructions); we left them out. Keep your API key out of the file: he reads his from a separate file. His note: "tool calls and streaming work."

! **Why a 15-minute timeout.** His worker server is shared, so a request "can wait in a queue for a few minutes when it is busy so the client timeout is 15 minutes" (900000 milliseconds).

04 THE HONEST LIMITS

READ THIS BEFORE YOU COPY IT

Garg is upfront about where this setup falls short. Here are his warnings, in his words.

1 There is no sandbox

"OpenCode does not sandbox the agent (SECURITY.md) unlike Codex (Seatbelt on macOS)." OpenCode's own security notes say its permissions are "not designed to provide security isolation." In plain words: the AI runs commands on your real computer. It happened to him: "in an early test my GLM implementer pip-installed pytest on its own."

2 So he made it ask first

His rules "ask before any shell command except read-only git, rg, builds, tests and lint." He also asks first on "> , .env, an absolute or home path, or ../" (writing to files, secret files, and paths outside the project). Order matters: "the last match wins."

3 Shared servers mean waiting

Requests can sit in a queue, so he set a 15-minute timeout and capped the boss at 3 worker agents at once.

4 The example is a toy

The only run he shows is a small Python function. His real sessions aren't shared, because "that is client work." The 70-75% figure is his experience, not a benchmark.

5 Not for every task

"I would not ideally switch to it for all tasks but it pairs really well with Sol (or maybe Opus 5.5)." Frontier models are "way ahead of their open-source counterparts for complex tasks."

! **Not covered in the post:** how GLM is actually served on the H200s. "How the model is deployed on the H200s is a separate post maybe for another day." Also not given: any dollar savings or speed figures.

05 TRY A SMALLER VERSION

NO H200 SERVER NEEDED OUR TIPS

Most people don't have a 4-GPU server. Garg says the model choice is open: "Nothing here depends on GLM. You can deploy this model or any other one, even something like Qwen3.8-27B on a single GPU." **Everything below is our own advice, not his.** In particular, using a cheaper hosted API as the worker is our idea; he only talks about running your own model.

1 **One paid boss, one cheaper worker**

Keep the strong model you already pay for as the boss. For the worker, use a smaller model on your own GPU, or (our idea) a cheaper model through any API that speaks the OpenAI-compatible format. Point the explorer and implementer at it.

2 **Give each worker its own files**

When two workers edit the same file, they undo each other. Hand each one a separate set of files. (His boss does this too: it gives the implementer briefs with separate file sets.)

3 **Cap how many run at once**

A small server or a rate-limited API chokes fast. Start with one or two workers at a time and raise it only if things stay quick.

4 **The boss reads the diff and reruns the checks**

Never trust a worker's "tests passed." Tell the boss to look at the actual changes (the diff) and run the tests itself before saying done.

5 **Approve shell commands yourself**

There's no sandbox, so set commands to "ask" and read each one. Better still, try it first in a throwaway copy of your project or a separate user account.

CHECK THE SPLIT

He measured his token share with this command. Run it after a few sessions to see whether the worker is really doing most of the work:

```
opencode stats --project . --models --cost
```

Note: a custom model shows $0.00 "only because no price is set for a custom model," so that number isn't real savings.

WHERE EVERY FACT COMES FROM

SOURCES

WHO SAYS WHAT

FACT	SOURCE
The split, the four agents, GLM-5.3-Flash on one 4-GPU H200 server, 320B/18B, 512K context, 8 to 10 users, token shares, toy example, limits of 3 agents and two failed rounds, 15-minute timeout, permission rules, no sandbox, Qwen3.8-27B, Codex and Claude Code limits	Aayush Garg, "Pairing GPT-6.1 Sol with a Local Model in OpenCode," Nov 1, 2026
What OpenCode is	OpenCode docs, intro page
Per-agent model, provider/model-id format, subagents inherit the caller's model	OpenCode docs, Agents page
No sandbox, permissions "not designed to provide security isolation"	OpenCode SECURITY.md, GitHub
Simplified config, "Our tips," cheaper-API worker idea	AI Stack Signal (our view)

THE FINE PRINT

- "70-75%" is the author's rough experience on bigger tasks. The only measured figure is about 60% on one toy task.
- No dollar savings or speed numbers were published, so we don't give any.
- The sample config is simplified and is not the author's exact file.
- The hardware is one server with 4 H200 GPUs, at the author's workplace.
- AI Stack Signal isn't connected to the author or to OpenCode.

LINKS

1. Aayush Garg, "Pairing GPT-6.1 Sol with a Local Model in OpenCode": aayushgarg.dev/posts/2026-11-01-gpt-local-model-opencode
2. OpenCode docs: opencode.ai/docs
3. OpenCode docs, Agents: opencode.ai/docs/agents
4. OpenCode SECURITY.md: github.com/anomalyco/opencode/blob/dev/SECURITY.md

news.aistacksignal.com/opencode

Facts checked against the sources above on November 1, 2026.