

"Harness" is the word everyone uses and nobody defines.

1 · MODEL

The brain

DeepSeek V4.1-Flash, GLM-5.3, Claude Sonnet 5.5. This is what you rent per token. It reads text, writes text, calls tools. It has **no memory, no files, no shell** on its own.

2 · HARNESS

The agent loop

Claude Code, OpenCode, Cline, dsh, ZCode. It decides what to ask the model, runs the tools, reads the errors, loops. Same model in a better harness = visibly better code.

3 · SURFACE

Where you sit

Terminal TUI, desktop app, VS Code panel. Cosmetic relative to the first two — but the reason people pick ZCode over a CLI, and it's a real reason.

The insight that saves money: models are now commodities you can swap per-request. The harness is the thing you actually learn and keep. That's why this deck compares harnesses, not just models.

Four names, four different things.

NAME	WHAT IT IS	COST	VERDICT IN ONE LINE
DeepSeek Harness dsh	DeepSeek's own MIT-licensed agent runtime — web UI, TUI, headless, SDK	free software + tokens	Real and official, but a <u>developer preview</u> — infrastructure, not a product
GLM Coding Plan Z.ai / Zhipu	A flat-rate subscription to GLM-5.3 that plugs into ~20 existing tools	\$18/mo (₹1,710)	Predictable cost, if you can live with two models and credit buckets
ZCode	A desktop app (Electron) — file tree, terminal, preview, review, mobile remote	app free + plan/key	The friendliest on-ramp; locks you to GLM routing
OpenCode	Open-source MIT agent — TUI + desktop + IDE extensions, 75+ providers	free + BYO key	The one to actually keep. Any model, your key, no lock-in

Not the official one: DeepSeek Engineer (Doriandarko, ~2.2k stars) is a third-party Python CLI, still on V3-era naming. Nothing to do with DeepSeek the company. If someone tells you it's "the DeepSeek harness", they're out of date.

DeepSeek does ship a harness. It's open source, MIT, and explicitly unstable.

- **Open-sourced 13 Aug 2026** , now in public preview — github.com/deepseek-ai/deepseek-harness
- **Five profiles:** `web` , `tui` , `headless` , `sdk` , `acp` (the last speaks to editors like Zed)
- **Desktop app** for macOS + Windows (v0.2 preview); Linux lives in the CLI
- **No first-party VS Code / JetBrains extension.** Marketplace hits are community-built — quality unverified

From their own README, in caps: "developer preview... **THERE WILL BE COMPATIBILITY-BREAKING CHANGES.**" Use it to learn, not to run your business on.

```
# the whole install
node -v          # must be ^22.19 or >=24 — it HANGS on 20
npx --yes @deepseek-ai/dsh --version
npx @deepseek-ai/dsh web      # → 127.0.0.1:3080

# one-shot, no UI
npx @deepseek-ai/dsh --profile headless \
  "summarise this repo and list its packages"

# see the plugin tree (the party trick)
npx @deepseek-ai/dsh web --dump-config | head -40
```

The silent Node-20 hang is the single most common "dsh is broken" report. Check `node -v` first, always.

Two models. One of them is a trap.

	DEEPSEEK-FLASH V4.1-FLASH-1017	DEEPSEEK-V4-PRO V4-PRO-0813
Context / max output	1M / 384K	1M / 384K
Vision	yes	no
Input · cache miss (off-peak / peak)	\$0.15 / \$0.30 per Mtok	\$0.66 / \$1.32
Input · cache hit	\$0.003 / \$0.006	\$0.022 / \$0.044
Output	\$0.60 / \$1.20	\$1.98 / \$3.96
Concurrency	2,500	500

OFF-PEAK = HALF PRICE

Peak is 06:30–09:30 & 11:30–15:30 IST

Mon–Fri only, in UTC terms 01–04 and 06–10. Which means peak lands right in an Indian afternoon — code in the evening and pay 50%.

CACHING IS AUTOMATIC

~50× cheaper on a hit

No code change. Agent sessions reuse prefixes constantly, so real bills sit far below the miss price — if your harness keeps context stable.

THE PRO TRAP

4.4× the price, worse at agents

On independent Terminal-Bench 4.0: Flash **31.8%**, V4-Pro **14.1%**. Paying more for the "bigger" model makes agentic coding worse.

Cheapest credible agentic model. Also the most verbose one measured.

WHAT IT'S GENUINELY GOOD AT

- **Cost per solved task.** Terminal-Bench 4.0: **\$0.54** per test vs \$13.20 (Claude Opus 5.5) and \$9.58 (GPT-6 Astra) — an 18–24× gap
- **1M context, 384K output** — you can paste a real repo in
- **Index 39, #10 of 121** on Artificial Analysis, ~181 tok/s
- **MIT weights** — you can self-host later if you ever want to

WHAT WILL ANNOY YOU

- **Verbosity.** 250M output tokens on that index — the highest of 47 models measured (median 140M). Output is 4× input price, so verbosity is your bill
- **Instruction drift on long files** — the 1M window is real, but quality fades well before you fill it. Keep files small; give it fresh context
- **Anthropic-layer gaps:** `cache_control` ignored, document blocks unsupported, MCP tool blocks unsupported — Claude-Code + MCP workflows degrade
- **Vendor benchmark caution:** DeepSeek's own 42.3% uses the Claude Code harness; the independent 31.8% uses Mini-SWE-agent. Both true, different claims

Practical read: the model is excellent value. DeepSeek's own harness is not the way to use it yet — run `deepseek - flash` inside a mature harness (OpenCode, Claude Code, Cline) and you get 80% of the value at 5% of the cost.

Ten commands, end to end — including the plumbing that trips people up.

```
# 1 node must be ^22.19 or >=24 (20 hangs silently)
node -v

# 2 key from platform.deepseek.com/api_keys
export DEEPSEEK_API_KEY=sk-...

# 3 smoke test without installing anything
npx --yes @deepseek-ai/dsh --version

# 4 boot the web UI
npx @deepseek-ai/dsh web          # 127.0.0.1:3080

# 5 raw API sanity check
curl -s https://api.deepseek.com/chat/completions \
  -H "Authorization: Bearer $DEEPSEEK_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"model": "deepseek-flash", "messages": [{"role": "user", "content": "hi"}]}'
```

```
# 6 check balance — catches 402 before it bites
curl -s https://api.deepseek.com/user/balance \
  -H "Authorization: Bearer $DEEPSEEK_API_KEY"

# 7 same model inside Claude Code
npm i -g @anthropic-ai/claude-code
export ANTHROPIC_BASE_URL=https://api.deepseek.com/anthropic
export ANTHROPIC_AUTH_TOKEN=$DEEPSEEK_API_KEY
export ANTHROPIC_MODEL="deepseek-flash[1m]"
claude

# 8 or inside OpenCode (cleaner, no env vars)
opencode          # then: /connect → deepseek → key
```

Rate limits are concurrency, not RPM: 2,500 in flight on Flash, 500 on Pro. Over the line you get **429**; empty balance is **402**. No tiered plans, no monthly minimum.

\$18/month for two models and a credit budget. Read the buckets.

TIER	MONTHLY	ANNUAL	CREDITS / 5H	/ WEEK
Lite	\$18 ₹1,710	\$12.6	2,000	10,000
Pro	\$80 ₹7,600	\$56	12,000	60,000
Max	\$168 ₹15,960	\$117.6	28,000	140,000

Changed 30 Jul 2026: plans moved from prompts per 5 hours to **credits**. Every "1,200 prompts/month" review you'll find is describing a plan nobody can buy anymore.

- Credits are **token-weighted** : $(in \times w + cached \times w + out \times w) / 10,000$
- GLM-5.3 weights: **6.9** input · 1.7 cached · **24 output** — output dominates
- Two caps at once: a 5-hour bucket and a weekly bucket
- **Weekends cost 50%** all day; peak is Mon–Fri 14:00–18:00 SGT

MODELS ON THE PLAN

GLM-5.3 (text) + GLM-5.3-Flash (multimodal)

Both **1M context, 128K output**. Requests for older IDs route automatically: 5.2/5.1 → 5.3, 4.7 → Flash. Flash carries the vision.

The quota cliff: when credits run out you wait for the reset. There is no overflow onto your cash balance, no top-up, no overage billing. Your agent stops mid-task.

Economics: \$18 ≈ ₹1,710 flat. That's more than pay-as-you-go DeepSeek Flash at light use, and cheaper than it at heavy use. The flat plan is insurance against a surprise bill — and a bet you'll stay under the cap.

Three base URLs. Pick the wrong one and it silently bills your cash balance instead.

TOOL YOU'RE WIRING UP	BASE URL	AUTH VAR STYLE
Claude Code, Goose, any Anthropic-shaped client	https://api.z.ai/api/anthropic	ANTHROPIC_BASE_URL + ANTHROPIC_AUTH_TOKEN
OpenCode, Cline, Roo, Codex, Cursor, Crush...	https://api.z.ai/api/coding/paas/v4	API key field (OpenAI-compatible)
OpenAI Responses API	https://api.z.ai/api/v1	API key field

A coding-plan key is not an API key. They're separate credentials — the plan key lives under [Personal Coding Plan](#) → [Plan Overview](#). Mixing them gives 1113 Insufficient Balance or quietly spends real credit.

Supported tools only. Using the plan outside the ~20 supported harnesses is a policy violation, and in practice the endpoint just won't bill the plan. Check the docs list before assuming it works.

```
# Claude Code - ~/.claude/settings.json
{
  "env": {
    "ANTHROPIC_BASE_URL": "https://api.z.ai/api/anthropic",
    "ANTHROPIC_AUTH_TOKEN": "<coding-plan-key>"
  }
}

# optional one-liner that writes this for you
npx @z_ai/coding-helper
```

A desktop app, not a CLI. The friendliest on-ramp — with a vendor lock.

- **An "ADE"** (agentic development environment) — Electron desktop app: `.dmg / .exe / AppImage / deb / rpm`
- **The app is free.** You bring a Coding Plan or any Anthropic/OpenAI-compatible key
- **Built-in:** file tree, terminal, git commit, live browser preview, review-diff flow, long-running "Goals", mobile remote, Telegram bot control
- **No plan? 5-day trial** at 8M tokens/day (3M GLM-5.3 + 5M Flash), then bring a key

Why a beginner should consider it: preview + review + commit in one window means you see what the agent changed before it hits your repo. That's the habit worth learning.

```
# install
macOS    drag ZCode.app → /Applications
          # Gatekeeper: xattr -dr com.apple.quarantine ZCode.app
Windows  run the .exe installer
Linux    chmod +x ZCode-*.AppImage    # needs libfuse2
          # or the .deb / .rpm
```

Two real gotchas: (1) **custom models default to ~200K context** unless the model id ends in `[ 1m ]` — you paid for 1M and get a fifth of it; (2) Linux/WSL login can fail on the `zcode://` URL handler.

Fine for learning the workflow. Less ideal as your permanent setup, because model routing is GLM-first and the app is desktop-only.

Open source, MIT, bring your own key. The one I'd actually install.

```
# install (pick one)
curl -fsSL https://opencode.ai/install | bash
npm i -g opencode-ai
brew install anomalyco/tap/opencode
choco install opencode          # windows (WSL recommended)

# point it at a provider
opencode
/connect          # → deepseek | z.ai coding plan | anthropic | openrouter
/models           # pick; switch per-session
```

Keys land in `~/.local/share/opencode/auth.json`. 75+ providers via models.dev — the same harness works whether you're on a ₹500/month token budget or a corporate key.

```
// ~/.config/opencode/opencode.json – any base URL
{
  "$schema": "https://opencode.ai/config.json",
  "model": "deepseek/deepseek-flash",
  "provider": {
    "zai": {
      "options": { "baseUrl": "https://api.z.ai/api/anthropic" }
    }
  }
}
```

- **Build / Plan agents** — Tab switches; Plan won't touch files
- **Subagents** for search/explore, `@` -mention files
- **LSP diagnostics** (`"lsp": true`) — real type errors, not vibes
- `/init` writes `AGENTS.md` ; `/share` gives a public link
- **VS Code / Cursor** extension, Cmd+Esc

Watch the versions: two generations coexist and both answer to the opencode command — brew and Arch's package give V2, the npm package may not. Pin deliberately, and check `opencode --version` after installing.

What each one costs at the entry tier — and what it locks you into.

	GLM CODING PLAN (LITE)	ZCODE	OPENCODE
Entry cost	\$18/mo ≈ ₹1,710	\$0 app; needs a plan (~\$18+) or key	\$0 + your API spend
Models	GLM-5.3 + Flash only	GLM-5.3/Flash, or your own key	Anything you plug in
Surface	Whatever tool you wire it to	Desktop GUI	TUI + desktop + IDE
Fail mode	Hard stop at quota, wait for reset	Vendor routing lock, desktop only	Version confusion, BYO-key setup
Best fit	Predictable flat cost, heavy use	Beginners who want a GUI	Anyone who wants to learn once

CHEAPEST TO START

OpenCode + DeepSeek Flash

₹0 to install. ₹1,200–2,000/month at ~2M input tokens/day off-peak, less with caching. Nothing prepaid, nothing to cancel.

MOST PREDICTABLE

GLM Coding Plan Lite

₹1,710 flat, and you cannot get a surprise bill — the trade is a hard wall mid-task and only two models.

BEST FOR LEARNING

ZCode + trial, then decide

Five free days at 8M tokens/day is enough to find out whether you like agentic coding before paying anything.

RECOMMENDATION

If I were starting from zero tomorrow, in India, on a student budget.

- **Week 1** — OpenCode + a ₹500 DeepSeek top-up. Learn the loop: `/init` , Plan agent, review every diff before accepting. Cost: ₹500, refundable as unused credit.
- **Week 2** — Add `AGENTS.md` to each project (OpenCode's `/init` writes it). This is the single biggest quality jump, and it's free.
- **Week 3** — Try ZCode's free 5-day trial only to see whether you prefer GUI over TUI. Uninstall if not.
- **Only then** — if your monthly token spend crosses ~₹1,700 and you want it fixed, buy GLM Lite. Not before.

The rule that carries over from the main deck: start on the free tier, understand exactly what the wall is, and only then pay — for the specific wall you hit.

SANITY CHECK

What this actually costs

OpenCode	₹0
DeepSeek tokens, light month	≈ ₹500
DeepSeek tokens, heavy month	≈ ₹2,400
GLM Lite (optional ceiling)	₹1,710
Hosting (main deck)	₹0

Total for the whole stack: **under ₹2,500/month** while learning. Domain is the only fixed cost.

Cost model: 2M input + 0.2M output tokens/day, off-peak, no cache reuse — DeepSeek Flash pricing as of 24 Oct 2026. Heavy month assumes 5M input/day with mixed peak.

Three things to take away, and one thing to do tonight.

01

Harness first, model second

Install a good agent loop once (OpenCode), then swap models freely. Your skills live in the harness and in `AGENTS.md` — not in the vendor's app.

02

Flat plans don't fail like free tiers

A quota doesn't degrade, it stops. Nothing falls back to your card. Know your reset clock before you start a long task.

03

The expensive model is often worse

For agentic coding, DeepSeek's own cheaper Flash beats V4-Pro on Terminal-Bench — at a quarter of the price. Always check the agentic benchmark, not the leaderboard vibe.

Tonight: install OpenCode, top up ₹500, run `/init` on the same project he deploys in the main session, and let the agent write the `AGENTS.md`. Two free wins, zero commitment.

Both decks, one principle: free tier → know the wall → pay only for the wall you actually hit. It works for hosting, and it works for tokens.