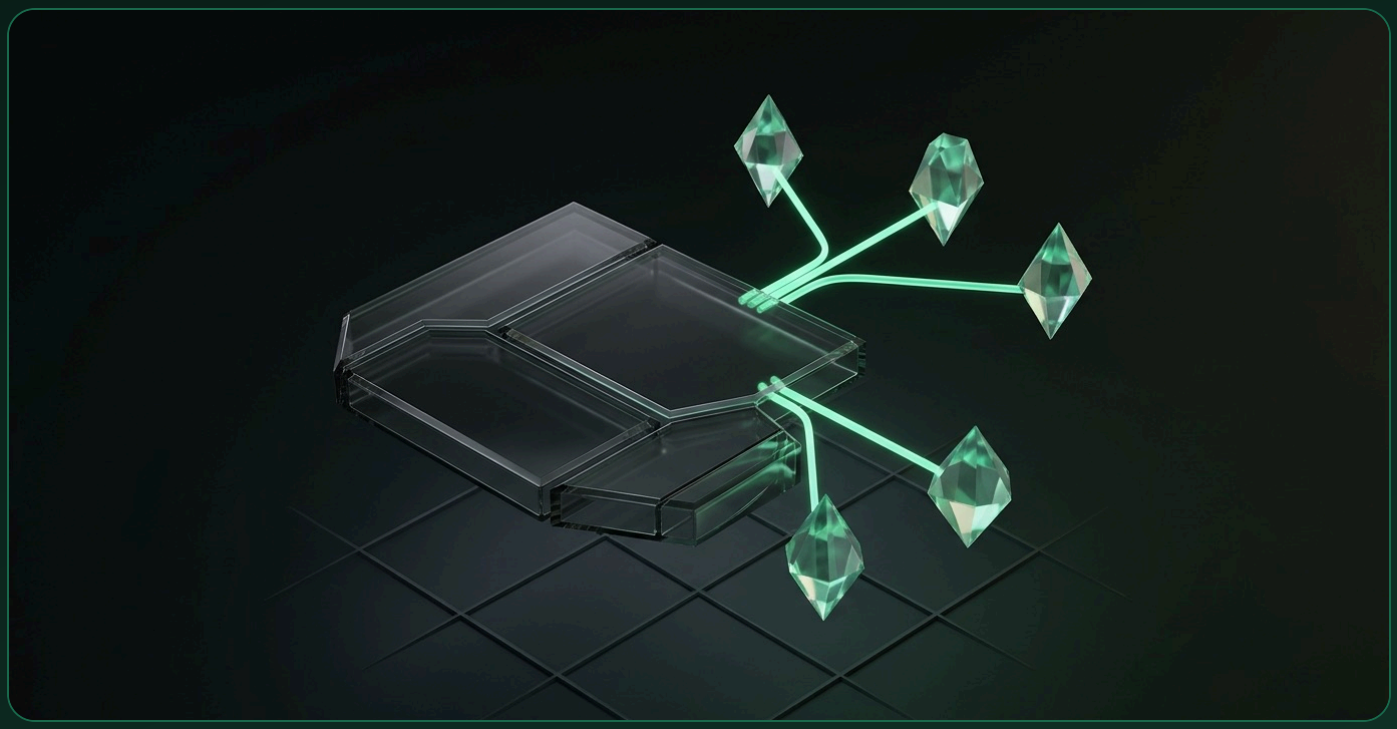
OPEN HARNESSES SERIES • OPENCODE TUI FIELD GUIDE

OpenCode TUI, Plan/Build Agents & Routing

Master the open-source, provider-agnostic terminal coding harness—switching cleanly between read-only Plan and full-access Build modes with `Tab`, routing 75+ cloud and local models in `opencode.json`, closing compiler loops via native LSP, and rewinding edits with Git-backed `/undo` snapshots.

- `Tab` Plan vs. Build Modes
- 75+ Provider Routing
- Native LSP Diagnostics
- Git Snapshot `/undo` & `/redo`
- `.opencode/agents/*.md`
- `/init`, `/compact` & `/share`



Zero Vendor Lock-In

Pair Claude, GPT, Gemini, DeepSeek, OpenCode Zen, and local Ollama models inside one unified keyboard-first TUI.

Compiler-Verified Edits

Built-in Language Server Protocol clients stream real type and syntax errors back to the agent automatically after every edit.

Fearless Git Time-Travel

Instant `/undo` and `/redo` revert both conversation turns and working-tree file changes together in milliseconds.

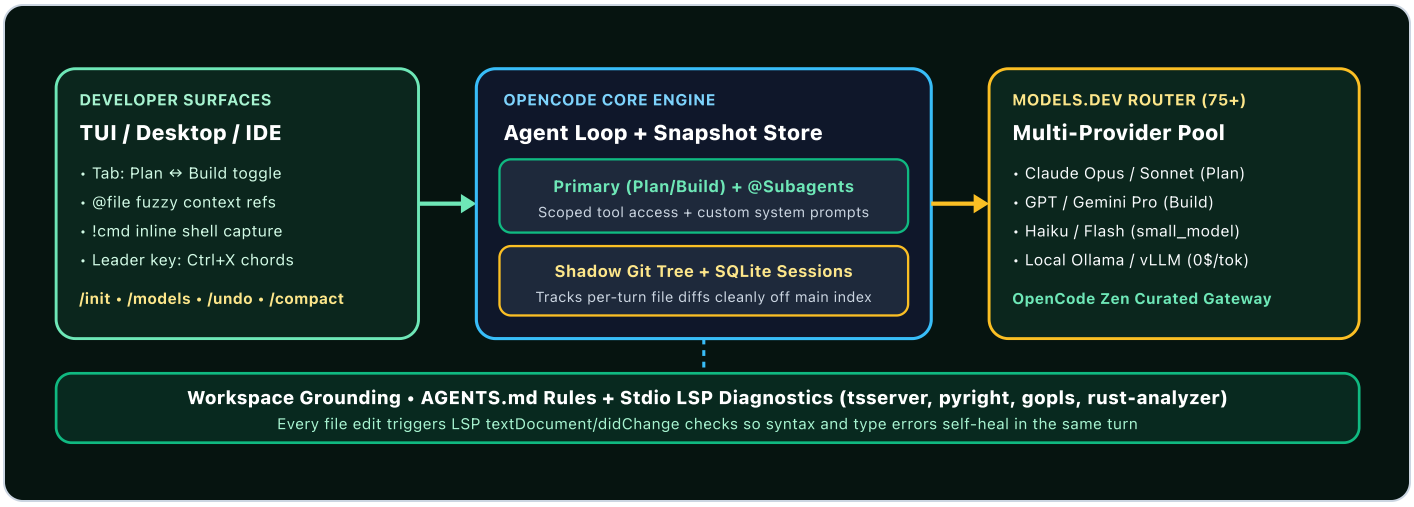
Why OpenCode: One Terminal Harness, Every Frontier & Local Model

Single-vendor coding CLIs tie your workflow, shortcuts, and team habits to one billing account. **OpenCode** decouples the **developer harness**—TUI ergonomics, file indexing, LSP diagnostics, session snapshots, and subagents—from the underlying model API so you can assign the sharpest brain to every task.

Quick Navigation

1. Agnostic Harness Mental Model p. 2
2. Install, Auth & Provider Matrix p. 3
3. Annotated OpenCode TUI Tour p. 4
4. Plan vs. Build Mode Mastery p. 5
5. Routing & Custom Markdown Agents p. 6
6. Troubleshooting & Pre-Flight p. 12
7. Native LSP & Context Hygiene p. 7
8. Git Snapshots (`/undo`) & Sessions p. 8
9. 5 Copy-Paste Production Recipes p. 9
10. Sharing, Privacy & Desktop Clients p. 10
11. Cross-Tool Synergy Pipeline p. 11

Architecture: Client-Server Core Driving TUI, LSP & 75+ Model Endpoints



Two-Tier Model Routing Cuts Spend by 50%+

Set a frontier reasoning model as your `plan` agent brain, a fast coding flagship for `build`, and a lightweight `small_model` in `opencode.json` for background session titles, summaries, and subagent exploration.

Installation, Credential Vault & Provider Strategy Matrix

Install OpenCode via native binary, Homebrew, or npm, link your preferred cloud subscriptions or local Ollama server via `/connect`, and run `/init` inside your repository to generate a tailored `AGENTS.md`.

1 60-Second Install Across macOS, Linux & WSL

```
# Official V2 install script or Homebrew / npm
curl -fsSL https://opencode.ai/v2/install | bash
brew install anomalyco/tap/opencode-v2
npm install -g @opencode/cli@latest

# Launch inside your repo & bootstrap AGENTS.md
cd ~/code/my-app && opencode
# Inside TUI: run /connect then /init
```

2 Authenticate Providers (`opencode auth`)

Run `/connect` inside the TUI or `opencode auth login` in your shell. Credentials live outside Git at `~/.local/share/opencode/auth.json`:

```
opencode auth login     # OAuth / API key wizard
opencode auth list      # Verify connected providers
opencode models         # List all resolved model IDs
```

Provider & Model Strategy Matrix

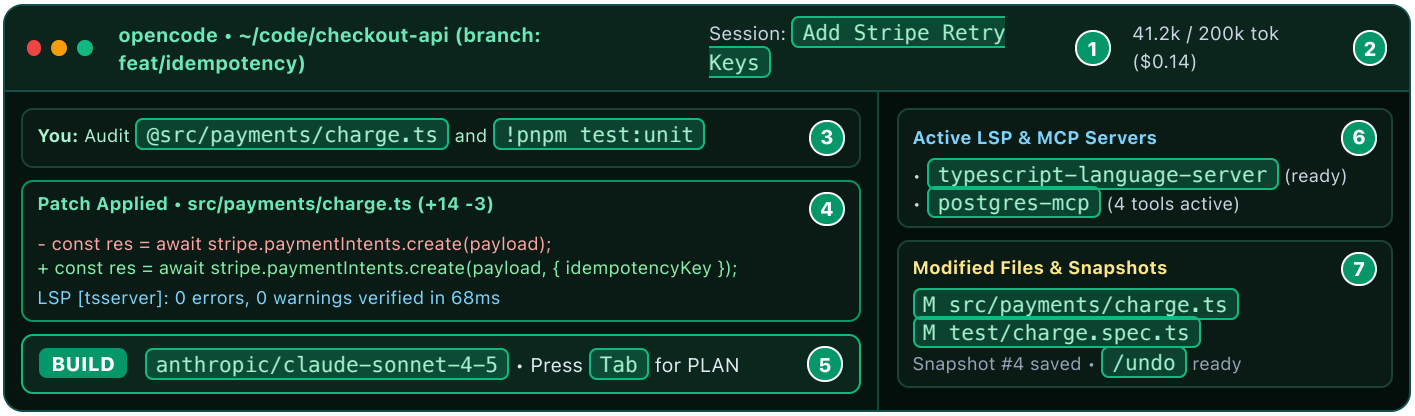
Provider Path	Example `model` String	Sweet Spot in OpenCode	Billing / Privacy Model
Anthropic / OpenAI	`anthropic/claude-sonnet-4-5` `openai/gpt-5`	Complex multi-file refactors, architectural `Plan` blueprints, deep debugging	Direct API key or supported Pro/Max OAuth flows
Google Gemini	`google/gemini-2.5-pro` `google/gemini-2.5-flash`	1M-token monorepo sweeps, large log analysis, ultra-fast `small_model`	AI Studio API key or Vertex AI enterprise IAM
OpenCode Zen	`opencode/claude-sonnet-4-5`	Zero-setup curated coding models benchmarked specifically for the OpenCode harness	Single unified pay-as-you-go balance via `/connect`
Local Ollama / vLLM	`ollama/qwen3-coder:30b`	Air-gapped codebases, offline flights, unlimited zero-marginal-cost edits	100% local hardware (set `num_ctx` ≥ `32768` !)

Config Precedence Order (Highest Wins)

OpenCode merges settings cleanly from general to specific: **1. Remote org defaults** → **2. Global** `~/.config/opencode/opencode.json` → **3. Env var** `OPENCODE_CONFIG` → **4. Project root** `opencode.json`. Commit project rules and model roles to Git while keeping API keys strictly in `auth.json` or `{env:VAR_NAME}` references.

Annotated Tour: Navigating the OpenCode TUI Like a Pro

OpenCode's keyboard-driven TUI keeps diffs, compiler diagnostics, token budgets, and active agent modes visible in one glance—controlled via quick slash commands and the `Ctrl+X` leader key.



Control Legend: Essential Zones & Leader Chords

1 **Session Header (`/sessions` or `Ctrl+X l`):** Jump between parallel task histories, rename threads, or fork a conversation cleanly.

2 **Live Context & Cost Meter:** Tracks exact input/cached tokens and dollar spend; run `/compact` before hitting 75% capacity.

3 `@` **File Fuzzy Search &** `!` **Shell Bar:** Type `@` to attach exact files/symbols or start a line with `!` to run a command and pipe stdout into context.

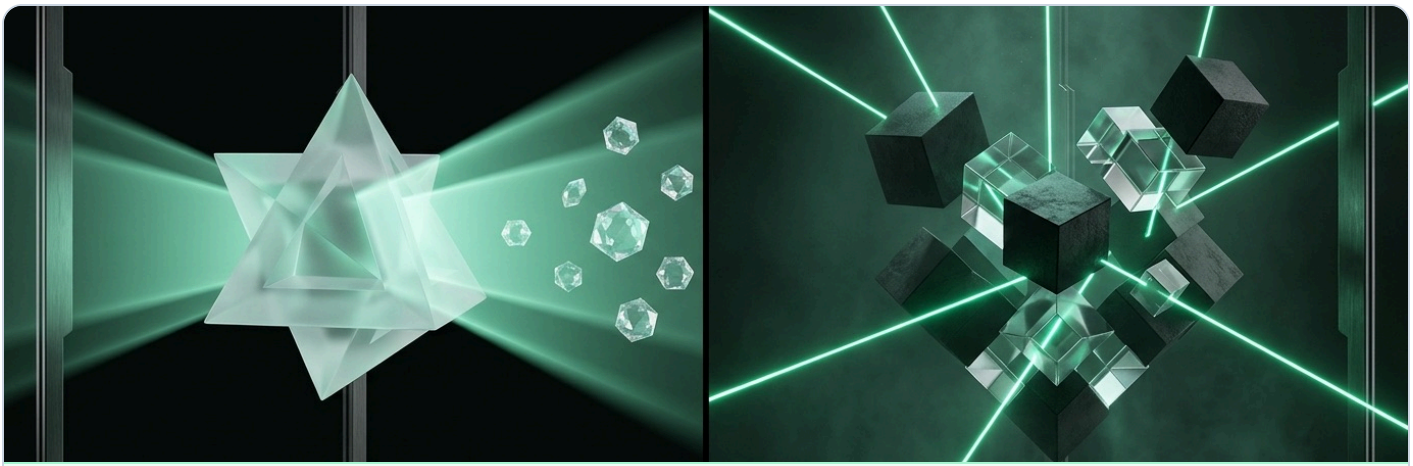
4 **Inline Diff + Automatic LSP Check:** Every edit renders unified color diffs and immediately reports compiler diagnostics from active language servers.

5 **Mode Badge & Model Selector (`Tab` / `Ctrl+X m`):** Press `Tab` to toggle **PLAN** ↔ **BUILD**, or `/models` to swap providers mid-thread.

6 **LSP/MCP Status &** 7 **Snapshot Tree:** Confirm healthy language servers and track every file touched since turn 1 for instant `/undo` rollback.

The `Tab` Key Habit: Separate Architecture From Mutation

The fastest way to wreck a codebase is letting an eager agent edit ten files before understanding how your auth middleware works. Pressing `Tab` switches OpenCode into **Plan Mode**—physically disabling file edits and destructive shell commands while the agent maps out a surgical blueprint.



Two Modes, One Keystroke (`Tab`): Inspect, search, and crystallize edge cases safely inside the read-only Plan frame (left) before flipping to Build mode (right) to execute verified multi-file patches and tests.

1 `PLAN` Mode (Read-Only Architect)

- **Hard Write Lock:** File creation, patch edits, and mutating bash tools are denied or gated by permission policy.
- **Deep Exploration:** Uses `grep`, `glob`, file reads, and `@explore` subagents to verify call sites before proposing changes.
- **Best Model Choice:** Route to high-reasoning models (`claude-opus-4-6`, `gpt-5`, or `gemini-2.5-pro`).

2 `BUILD` Mode (Full-Access Implementer)

- **Full Execution Loop:** Applies surgical patches, receives automatic LSP compiler feedback, and runs test suites.
- **Follows the Approved Plan:** Executes step by step against the blueprint already agreed upon in the same session context.
- **Best Model Choice:** Fast, high-throughput coding models (`claude-sonnet-4-5` or local `qwen3-coder`).

The 4-Step Plan → Review → Build Rhythm

```
1. Press [Tab] -> Confirm badge reads PLAN.
2. Prompt: "Trace @src/auth/session.ts and propose a 3-step plan to rotate refresh tokens."
3. Challenge assumptions ("Keep backward compatibility for mobile v2 tokens; skip DB migration").
4. Press [Tab] -> Badge flips to BUILD -> Prompt: "Execute steps 1-3 and run !pnpm test auth"
```

Multi-Model Routing & Markdown Specialist Subagents

Instead of paying flagship prices on every grep or code review, assign tailored models, temperatures, and tool boundaries per agent role in `opencode.json` or drop self-contained Markdown files into `.opencode/agents/`.

1. Role-Based Routing in `opencode.json`

```
{
  "$schema": "https://opencode.ai/config.json",
  "model": "anthropic/claude-sonnet-4-5",
  "small_model": "google/gemini-2.5-flash",
  "agent": {
    "plan": {
      "model": "openai/gpt-5",
      "temperature": 0.1
    },
    "build": {
      "model": "anthropic/claude-sonnet-4-5"
    }
  }
}
```

2. Custom Subagent (`.opencode/agents/sec.md`)

```
---
description: Audits diffs for OWASP & auth bugs
mode: subagent
model: google/gemini-2.5-pro
temperature: 0.1
tools:
  write: false
  edit: false
  bash: false
---
Review changed files for SQLi, SSRF, IDOR, and
leaked secrets. Cite file:line and minimal fix.
```

Invoking Subagents: Mention any subagent directly in chat (for example, `@sec audit my staged changes` or built-in `@explore find where webhook signatures are verified`), or let your primary agent delegate automatically based on its `description` frontmatter. Navigate between parent and child subagent threads anytime using `Ctrl+X Right` / `Left`.

Color-Coded Blueprint: High-Signal OpenCode TUI Prompt

[In PLAN mode] Ask @explore to map every caller of @src/billing/UsageMeter.ts and inspect recent test failures from !vitest run billing, draft a zero-downtime batching plan using our configured gpt-5 planner, then switch via Tab to BUILD, resolve any tsserver LSP diagnostics to zero, and run @sec before committing so we can /undo cleanly if edge cases fail.

Mode + Subagent | @File & !Shell Context | Model Role Routing | LSP Zero-Error Gate | Review & Snapshot Safety

Native LSP Loops, `AGENTS.md` & Token Compaction

Most terminal agents waste three turns running full builds just to discover a missing import. OpenCode spawns language servers directly behind the scenes, streaming instant compiler diagnostics into the agent turn the moment a file is written.

1 Built-In & Custom LSP Configuration

OpenCode auto-detects TypeScript, Python, Go, Rust, Elixir, Zig, C#, and Astro language servers. Customize or disable servers in `opencode.json`:

```
{
  "lsp": {
    "pyright": {
      "env": { "VIRTUAL_ENV": ".venv" }
    },
    "eslint": { "disabled": true }
  }
}
```

2 High-Signal `AGENTS.md` via `/init`

Running `/init` scans your build scripts, lint rules, and folder conventions to write `AGENTS.md` (also compatible with `CLAUDE.md` fallback):

- Pin exact single-file test commands (`pnpm vitest run path/to/file`).
- List architectural boundaries and forbidden directories.
- Load shared team docs via `"instructions": ["docs/arch.md"]`.

Context Window Hygiene: Staying Fast Past Turn 30

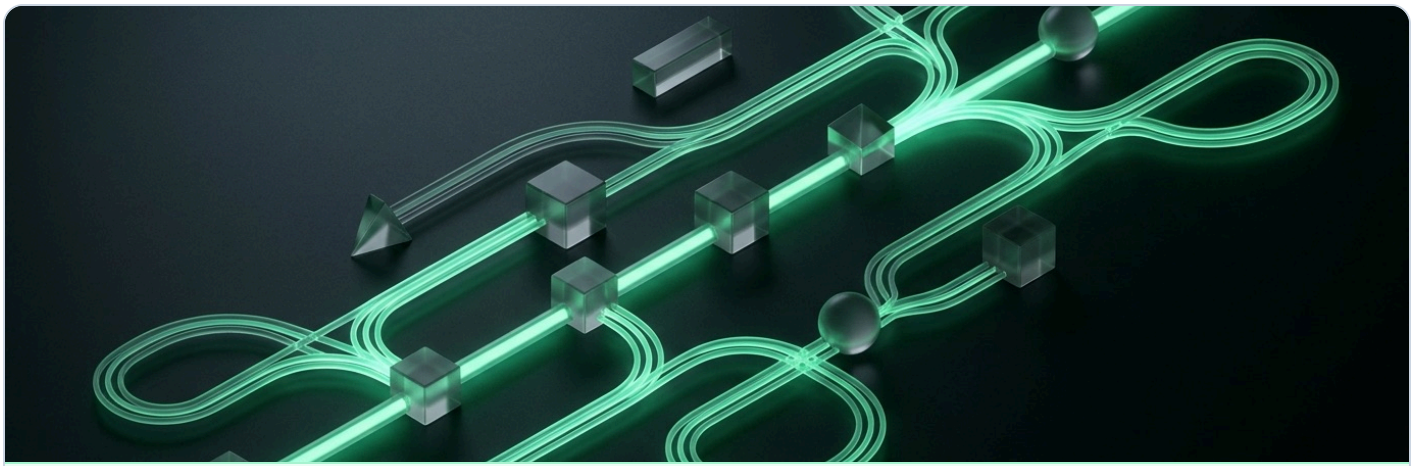
Command / Shortcut	What It Does Under the Hood	When to Trigger It
`/compact` (`Ctrl+X c`)	Summarizes prior turns, resolved tool logs, and key decisions into a tight handoff block	After finishing one feature slice or whenever token usage crosses ~65–75%
`/new` (`Ctrl+X n`)	Starts a brand-new clean conversation while preserving `AGENTS.md` project rules	Switching to an unrelated bug or subsystem where prior files add noise
`/editor` (`Ctrl+X e`)	Opens your `$EDITOR` (Neovim, VS Code, Zed) to compose multi-line specs comfortably	Drafting detailed acceptance criteria, pasted stack traces, or table schemas
Clipboard Image Paste	Drag-and-drop or paste UI screenshots straight into the TUI prompt box	Fixing CSS alignment bugs, mobile overflow, or matching a wireframe

Delegate Broad Searches to `@explore` to Protect Main Context

Reading 20 candidate files directly inside your main Build thread bloats every subsequent turn's prompt cache. Ask `@explore` to locate exact symbols first; only its concise `file:line` summary enters your main session.

Fearless Experimentation with `/undo` & `/redo` Snapshots

Traditional `git reset --hard` wipes your manual edits or pollutes commit history. OpenCode maintains a private shadow Git object store behind your repository so every turn is atomically reversible without touching your real Git staging index.



Coupled Code + Context Rewind: Running `/undo` rolls back both the workspace files and the bad conversation turn simultaneously, letting you refine your prompt and replay cleanly from a known-good checkpoint.

1 How `/undo` Beats Follow-Up Nagging

When an agent goes down the wrong architectural path, arguing with it in turn 6 leaves five turns of broken code polluting its context window. Instead:

1. Run `/undo` (`Ctrl+X u`) once or twice to restore exact pre-turn files; your original prompt pops right back into the input box.
2. Add the missing constraint or `@file` reference and hit `Enter`.
3. Changed your mind? Run `/redo` (`Ctrl+X r`) to restore the reverted turn and patches intact.

2 Comparing Two Models on One Bug

Benchmark two providers head-to-head on your real repo in under two minutes:

1. Let Model A attempt the fix and run `!pnpm test`.
2. Run `/undo` to rewind the working tree to its exact starting state.
3. Switch models with `/models` (`Ctrl+X m`) to Model B and submit the exact same pre-filled prompt.

Git Repository Required for Snapshot Tracking

OpenCode's `/undo` and `/redo` engine relies on your folder being initialized with `git init`. Always initialize Git in scratch prototypes before starting heavy multi-file Build loops.

5 Real-World Copy-Paste Recipes

Copy these five ready-to-use configurations and slash commands into your project to unlock hybrid cloud/local routing, custom slash commands, local Ollama 32K coding, and session continuation.

Recipe & Goal	Copy-Paste Snippet / Config	Expected Outcome
1. Cloud Planner + Local Ollama Coder	`{"agent":{"plan":{"model":"anthropic/claude-opus-4-6"},"build":{"model":"ollama/qwen3-coder:30b"}},"provider":{"ollama":{"npm":"@ai-sdk/openai-compatible","options":{"baseURL":"http://localhost:11434/v1"}}}}`	Uses frontier reasoning in `PLAN` mode and zero-cost local GPU tokens in `BUILD` mode.
2. Custom `/test-fix` Slash Command	Save `.opencode/commands/test-fix.md`: `---` `description: Run suite and fix failures` `---` `Failure output: !`pnpm test --run` — fix root cause for $ARGUMENTS without weakening assertions.`	Typing `/test-fix auth` executes tests first, injects stdout, and fixes the failing module.
3. Resume Latest Session From Shell	`opencode --continue` (or `opencode -s ses_01JXYZ...`)	Reopens your exact prior TUI thread, snapshot chain, and active model immediately.
4. Auto-Format Files After Every Edit	In `opencode.json`: `{"formatter":{"biome":{"command":["npx","@biomejs/biome","check","--write","$FILE"],"extensions":[".ts",".tsx",".json"]}}}`	Runs Biome automatically on every touched file so agent diffs always match repo style.
5. Export Full Session JSON Audit Log	`opencode session export ses_01JXYZ > session-audit.json`	Dumps complete prompts, tool calls, token counts, and diffs for team postmortems or evals.

Dynamic Shell Injection Inside Custom Commands

Inside `.opencode/commands/*.md` templates, wrapping a command in exclamation-backtick syntax runs that shell command locally (such as `git diff --staged` or test runners) and splices its fresh output straight into the prompt before sending it to the LLM.

Session Sharing Controls, Privacy Locks & IDE Companions

OpenCode makes code review walkthroughs effortless with one-command share links, while giving enterprise security teams a single config switch to lock sharing off completely across private repositories.

1 Three Session `share` Modes

Configure `"share"` in `opencode.json` to match your repo's confidentiality level:

- `"manual"` **(Default):** Nothing leaves your machine unless you explicitly run `/share` (`Ctrl+X s`); revoke anytime with `/unshare` .
- `"disabled"` **(Enterprise Lock):** Hard-blocks `/share` in proprietary repos so source snippets can never be published to a public URL.
- `"auto"` **:** Automatically creates shareable links for open-source PR walkthroughs.

2 Secret & Env Variable Substitution

- **Never Hardcode Keys:** Reference environment secrets in `opencode.json` with `"{env:LITELLM_API_KEY}"` or file contents via `"{file:~/.secrets/token}"` .
- **Block Sensitive Reads:** Add `.env*` , `*.pem` , and `secrets/` deny rules under `permission.read` so prompts never ingest raw credentials.
- **Zero-Telemetry Local Stack:** Pair `"share": "disabled"` with local Ollama or VPC Bedrock/Vertex endpoints for full data residency.

3 VS Code, Cursor & Neovim Companion

Run OpenCode inside your editor's integrated terminal pane: active file selections, line ranges, and diagnostic problems jump straight into your TUI prompt with a single shortcut while diffs update live in editor buffers.

4 Themes & TrueColor Terminal Tuning

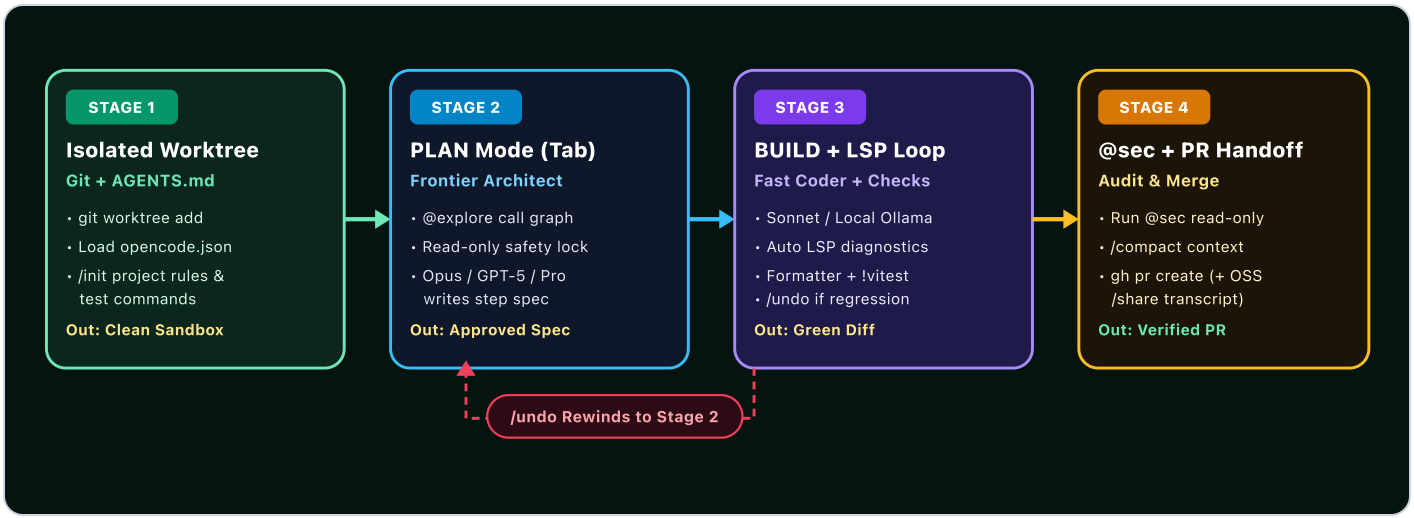
Run `/themes` (`Ctrl+X t`) and pick `system` or built-in palettes (Catppuccin, Tokyonight, Gruvbox, Rose Pine). Use a TrueColor terminal (Ghostty, Kitty, WezTerm, iTerm2, or Alacritty) for crisp diff highlights.

Commit `"share": "disabled"` on Day One in Private Client Repos

Adding `{"share": "disabled"}` to your repository's checked-in `opencode.json` protects every developer on the team from accidentally running `Ctrl+X s` during a late-night debugging session.

Cross-Tool Synergy: Worktrees + Plan/Build + LSP + PR Review

Combine isolated Git worktrees, multi-provider Plan/Build routing, native LSP verification, and shareable PR walkthroughs into a repeatable high-velocity engineering loop.



A Parallel Sessions Across Git Worktrees

Run three independent `opencode` TUI instances inside three `git worktree` directories (for example inside tmux or Ghostty splits) so each agent gets isolated shadow snapshots, ports, and LSP caches.

B Pairing With Part 2: Plugins, Tools & CI

Once your interactive TUI workflow is dialed in, turn to the companion handbook (*OpenCode Plugins, Custom Tools, Servers & CI*) to wire TypeScript hooks, headless `opencode serve`, and GitHub Actions automation.

Troubleshooting Clinic & 6-Point Pre-Flight Checklist

Fix the seven most frequent TUI, local-model, LSP, and snapshot hiccups in ten seconds flat, then run the 6-point checklist when onboarding any new repository.

Symptom / Issue	10-Second Root-Cause Fix
1. Local Ollama model forgets system tools or loops	Ollama defaults to a tiny `4096` token context window! Build a Modelfile with `PARAMETER num_ctx 32768` (or `65536`) before selecting it in OpenCode.
2. Agent refuses to edit files even when asked	Check the bottom-left mode badge—you are still in read-only `PLAN` mode! Press `Tab` once to flip the badge to `BUILD`.
3. `/undo` reports snapshot unavailable	The directory is not a Git repository. Run `git init` at the workspace root so OpenCode can initialize its shadow snapshot tree.
4. LSP diagnostics missing for Python or TypeScript	Language server binary isn't on `PATH` or wrong venv is active. Install `pyright` / `typescript-language-server` or pass `VIRTUAL_ENV` under `lsp` in `opencode.json`.
5. Newly added API key doesn't show models in `/models`	Run `/connect` inside the TUI, check `opencode auth list`, or run `opencode reload` before `opencode models`.
6. Diff colors look washed out inside tmux	Enable 24-bit TrueColor in `~/.tmux.conf` (`set -ga terminal-overrides ",xterm-256color:Tc"`) and select `/themes` → `system`.
7. Responses slow down & cost spikes after 40 turns	Long raw tool outputs are re-sent each turn. Run `/compact` (`Ctrl+X c`) to distill history or start a fresh thread with `/new`.

6-Point Pre-Flight Checklist for Every New Repo

- [ ] **Bootstrap `AGENTS.md`:** Ran `/init` and pinned fast single-file test and lint commands.
- [ ] **Split Model Roles:** Configured frontier `plan`, fast `build`, and cheap `small_model` in `opencode.json`.
- [ ] **Lock Sharing Policy:** Set `"share": "disabled"` in checked-in config for private or client repos.
- [ ] **Verify Active LSPs:** Confirmed green language server status on first file touch for instant error loops.
- [ ] **Plan Before Build:** Started complex tasks in `Tab` (`PLAN`) mode with `@explore` before mutating files.
- [ ] **Rewind, Don't Argue:** Used `/undo` on off-track turns and `/compact` between feature milestones.