

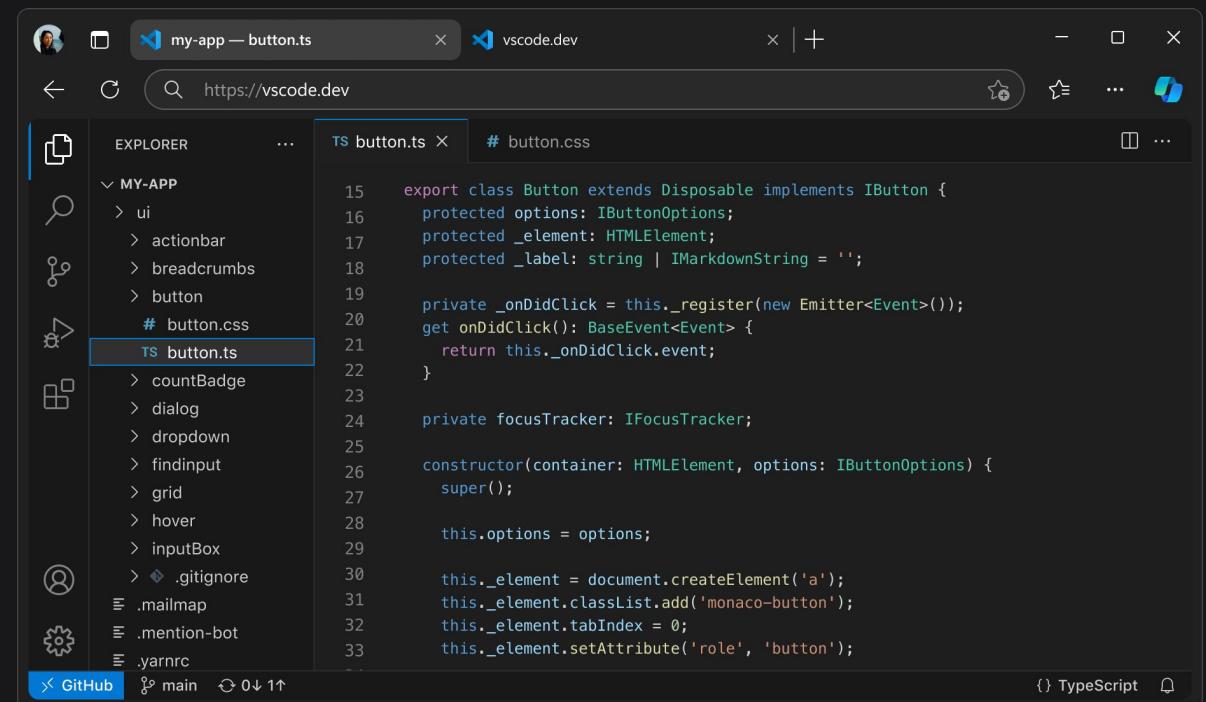
Open Source Agentic CLI Interfaces

Defining the Paradigm

⌨️ The "Writer" Model

Legacy IDEs

- › **Action:** User types, AI suggests.
- › **Scope:** Single file buffer.
- › **Loop:** Human reads → Human types.



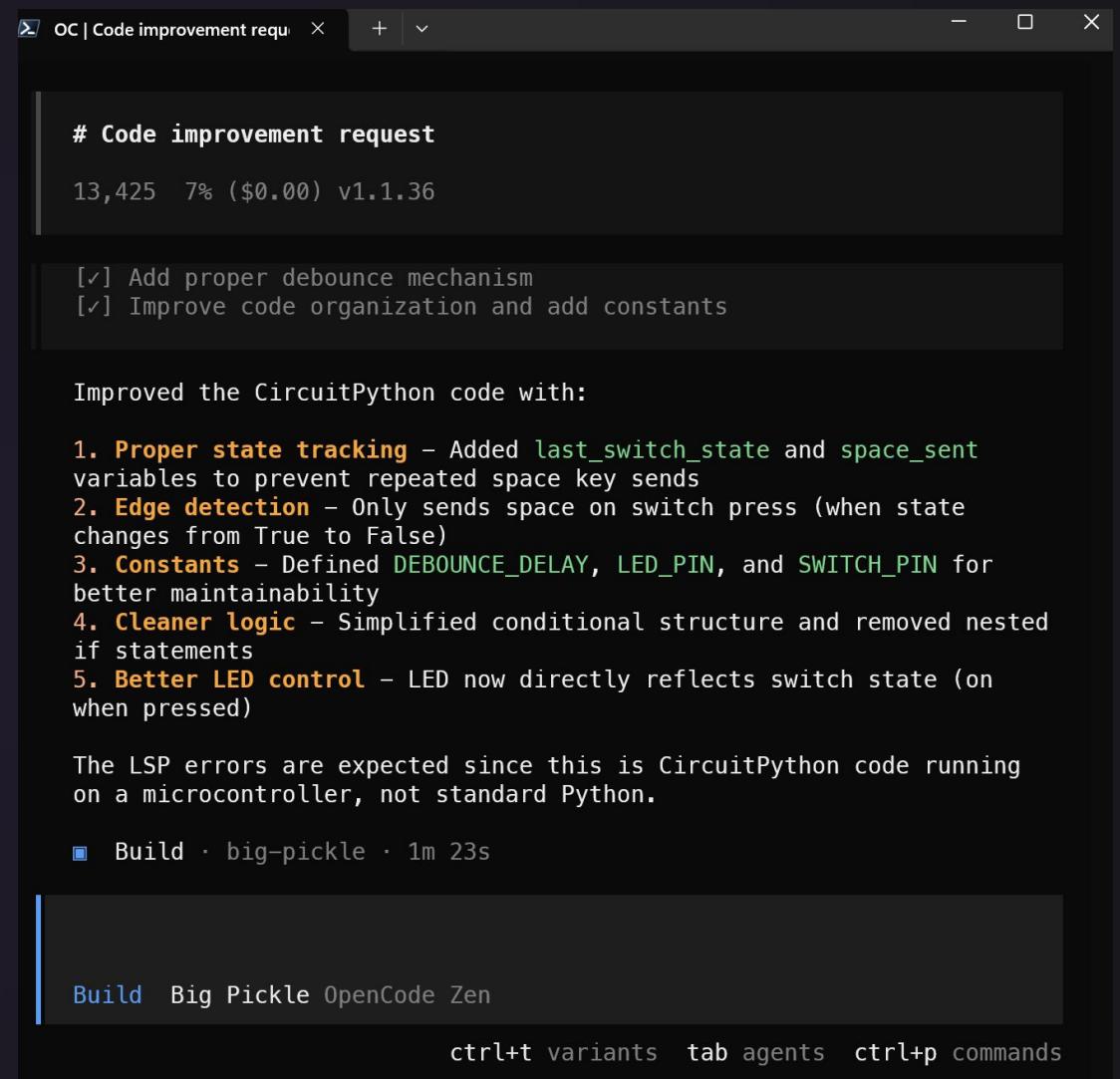
```
15 export class Button extends Disposable implements IButton {
16   protected options: IButtonOptions;
17   protected _element: HTMLElement;
18   protected _label: string | IMarkdownString = '';
19
20   private _onDidClick = this._register(new Emitter<Event>());
21   get onDidClick(): BaseEvent<Event> {
22     return this._onDidClick.event;
23   }
24
25   private focusTracker: IFocusTracker;
26
27   constructor(container: HTMLElement, options: IButtonOptions) {
28     super();
29
30     this.options = options;
31
32     this._element = document.createElement('a');
33     this._element.classList.add('monaco-button');
```

Defining the Paradigm

🤖 The "Architect" Model

Agentic CLIs

- > **Action:** User Prompts, Agent Acts.
- > **Scope:** Entire Repo & Terminal.
- > **Loop:** Plan → Act → Verify.



The screenshot shows a terminal window titled "OC | Code improvement requ..." with the following content:

```
# Code improvement request
13,425 7% ($0.00) v1.1.36

[✓] Add proper debounce mechanism
[✓] Improve code organization and add constants

Improved the CircuitPython code with:
1. Proper state tracking – Added last_switch_state and space_sent variables to prevent repeated space key sends
2. Edge detection – Only sends space on switch press (when state changes from True to False)
3. Constants – Defined DEBOUNCE_DELAY, LED_PIN, and SWITCH_PIN for better maintainability
4. Cleaner logic – Simplified conditional structure and removed nested if statements
5. Better LED control – LED now directly reflects switch state (on when pressed)

The LSP errors are expected since this is CircuitPython code running on a microcontroller, not standard Python.

Build · big-pickle · 1m 23s
```

At the bottom of the terminal window, there is a footer with the following text:

Build Big Pickle OpenCode Zen
ctrl+t variants tab agents ctrl+p commands

| The "Human-in-the-Loop" Workflow



1. Define

You define the **Goal** using natural language, not the implementation code.

"Refactor the login module to use OAuth2."



2. Execute

Agent enters an **Autonomous Loop**:
Reading files, running shell cmd's, and self-correcting.



3. Review

You review the **Outcome** (diffs). You verify logic, not syntax.

Proprietary Constraints

Gemini CLI • Claude Code • Copilot

These tools utilize a "Black Box" architecture that introduces significant risks for engineering teams.

- **Model Lock-in:** Bound to a single provider's roadmap.
- **Data Egress:** Intellectual Property leaves your machine.
- **Opacity:** Logic and System Prompts are hidden.



OpenCode

Model-Agnostic, Open-Source

The Hybrid Stack

Decoupling the **Control Plane** (Logic) from the **Inference Plane** (LLM).



Application

OpenCode CLI

Runtime & Tool definitions



Local

Inference
Ollama / vLLM

Llama, Mistral ...



Cloud

Inference
API Gateway

Any provider

| The Strategic Advantage

Future Proofing

New, better models are released weekly.

OpenCode lets you swap the backend instantly—use GPT-5.2 today, Claude 5 tomorrow, or a local Llama model to save costs.

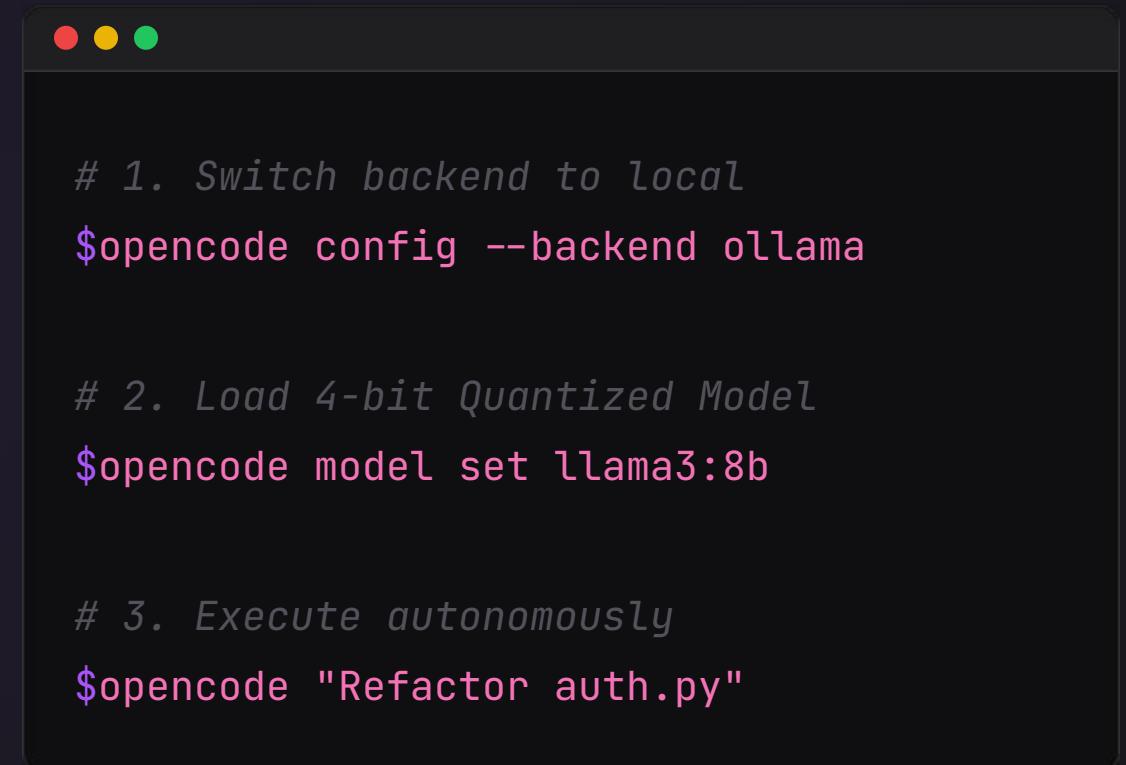
Rapid Evolution

Proprietary tools update on the vendor's schedule. Open source tools update at the speed of the community. New features and integrations are added by developers like you daily.

1. Local Execution (Ollama)

🛡 Why Local?

- **Sovereignty:** Code never leaves localhost.
Mandatory for NDA/HIPAA.
- **Air-Gapped:** Functional without internet.
- **Latency:** No network RTT. Limited only by VRAM.

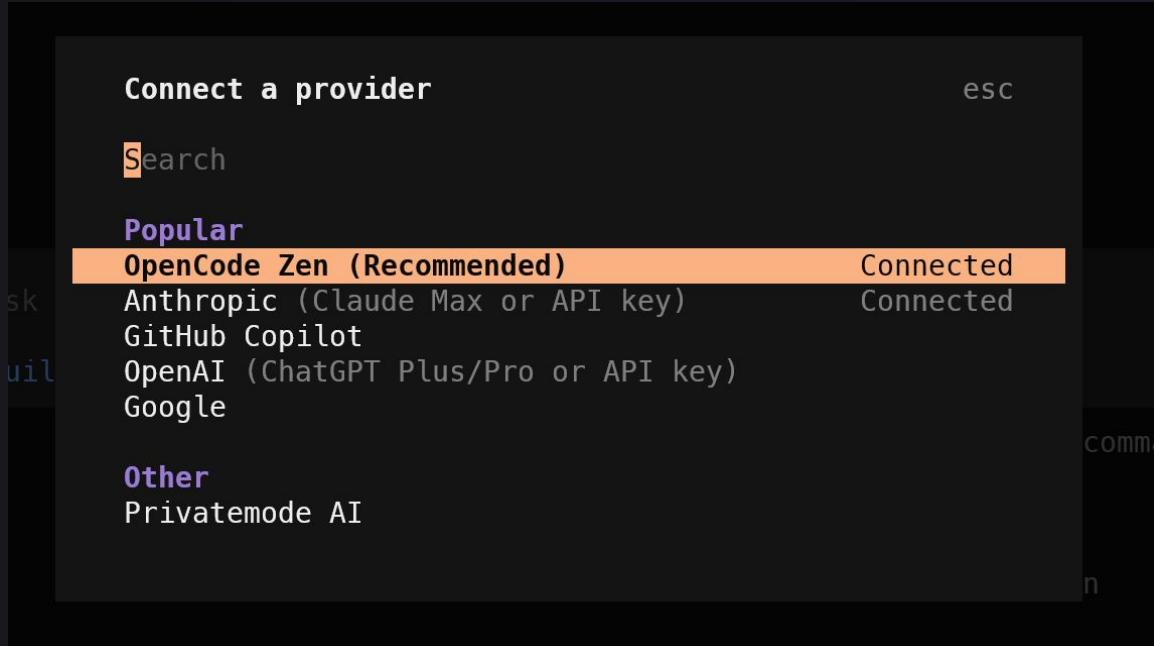


```
# 1. Switch backend to local
$opencode config --backend ollama

# 2. Load 4-bit Quantized Model
$opencode model set llama3:8b

# 3. Execute autonomously
$opencode "Refactor auth.py"
```

2. Direct API Integration



🔑 Bring Your Own Key (BYOK)

When you need SOTA reasoning power, swap the engine, keep the interface.

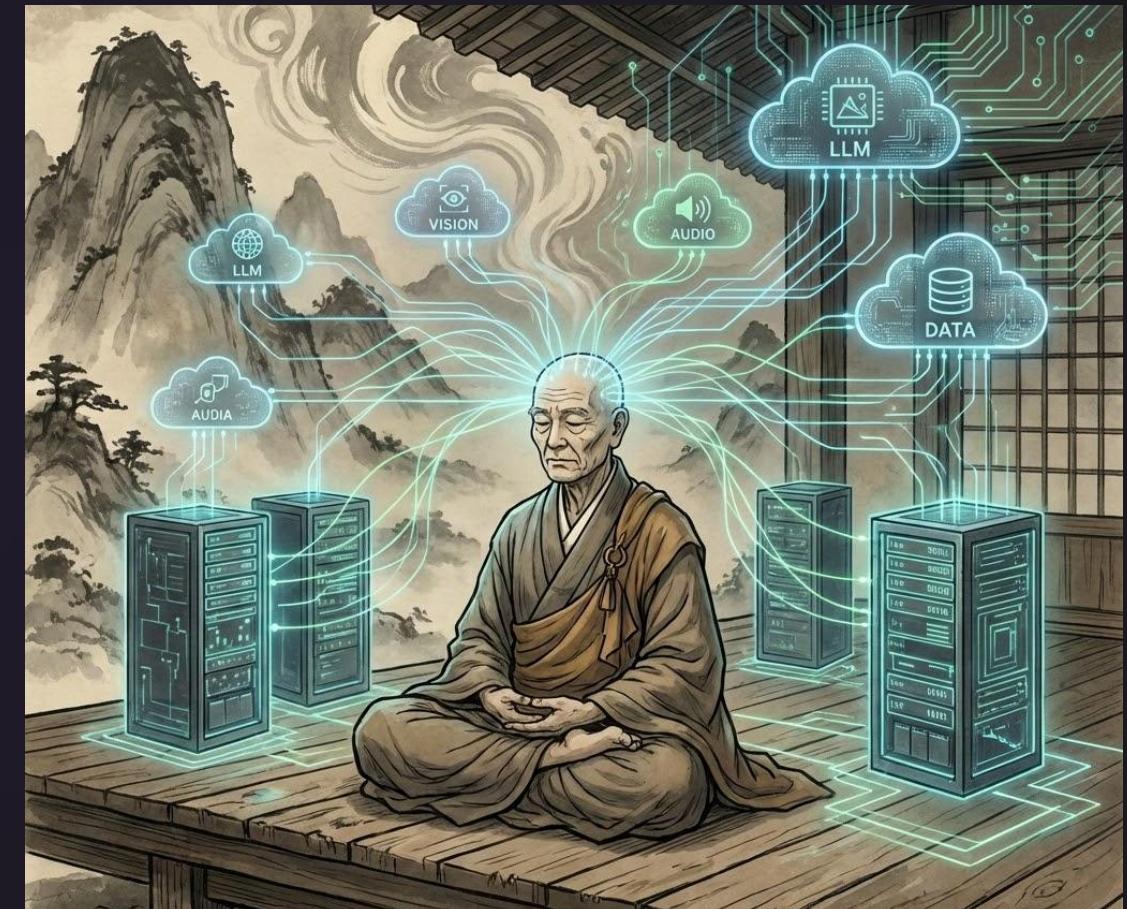
- **Multi-Provider:** Switch between tens of providers and their models.
- **Direct Billing:** Pay provider directly. No markup.

| 3. OpenCode ZEN Service

☰ The Aggregation Layer

A unified gateway (similar to OpenRouter) for instant access to any model.

- › **Single Billing:** One wallet for all vendors.
- › **Usage-Based:** Pay per token. No monthly SaaS seats.
- › **Sometimes free!** New models in testing are usually free during the testing period.



AGENTIC CLI COMPARISON: Proprietary vs. Open Source

FEATURE	PROPRIETARY AGENTIC CLIs (e.g., Claude Code)	OPEN SOURCE AGENTIC CLIs (e.g., OpenCode)
Control & Sovereignty	Vendor-Controlled, Black Box Execution	User-Controlled, Full Execution Visibility
Model Flexibility	Locked to Single Vendor Models	Agnostic, Swap Any Model (Local/Cloud)
Data Privacy	Data Egress to Vendor Cloud	Private (Local) or Controlled Egress
Cost Structure	Subscription / Bundled Pricing	Pay-per-Token (Direct) / Free (Local)
Vendor Lock-in	High, Tied to Ecosystem Roadmap	None, Future-Proof Architecture
Transparency	Opaque System Prompts & Logic	Transparent, Open Logic & Prompts

Comparison based on general architectural differences.

The Future is Agnostic

**"The most dangerous architectural decision is
coupling your workflow to a single model provider."**
