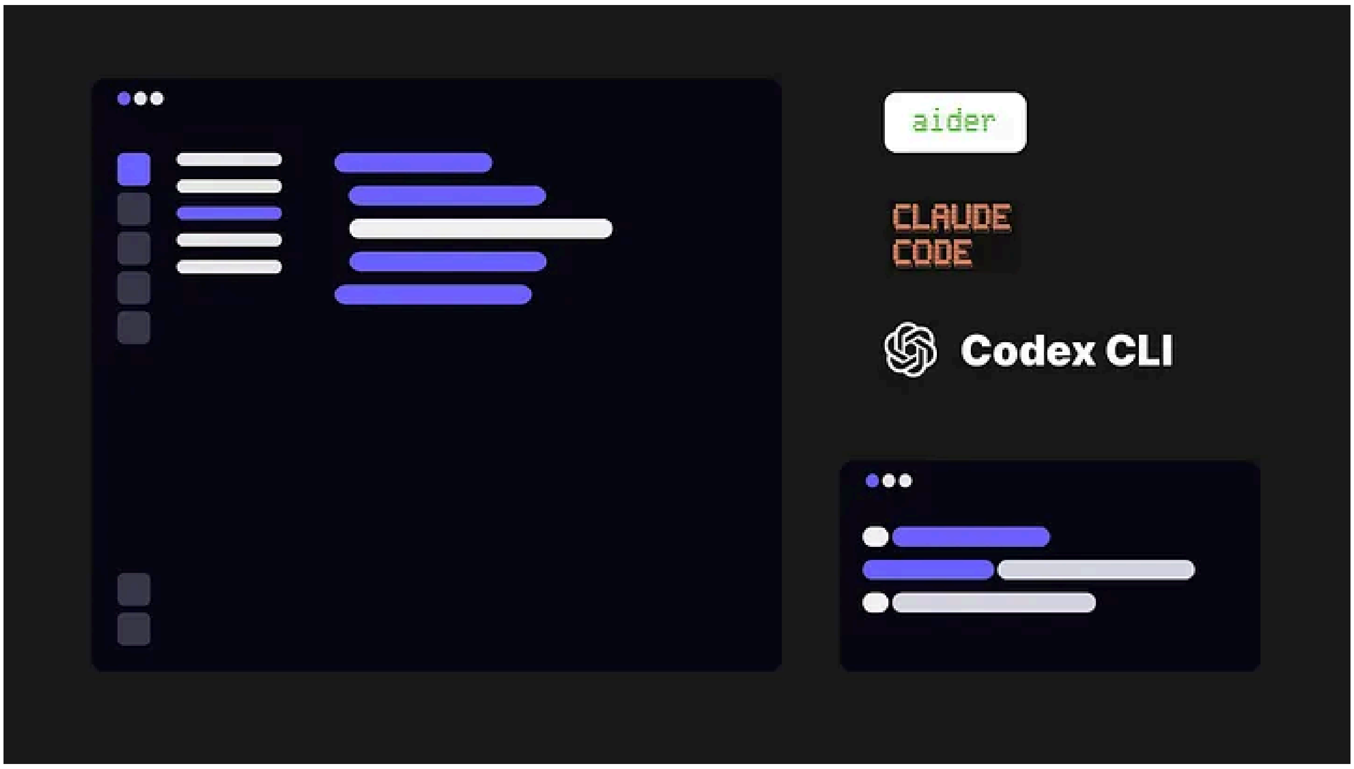
Compare the Top 5 Agentic CLI Coding Tools

14 min read · Jun 6, 2025

Amos Gyamfi Follow

Discover how CLI-based AI coding tools let you work directly with LLMs and agents from your Terminal. Use them to improve code, debug and fix errors, and collaborate on a Git project without opening an IDE.



Agentic AI coding tools vary in how they help you write, debug, and ship code.

Some, like Lovable and Bolt, allow developers to build web and mobile apps using prompts quickly. Others, like Cursor and Windsurf, provide developers with a fully AI-featured IDE to solve engineering problems.

Generally, AI coding platforms can be categorized into the following:

- **CLI-Based Agents**: Interact with AI agents through the command line using Aider, Claude Code, Codex CLI, Gemini CLI, and Warp.
- **AI Code Editors**: Interact with agents through GitHub Copilot, Cursor, and Windsurf.
- **Vibe Coding**: Build web and mobile applications with prompts using Bolt, Lovable, v0, Replit, Firebase Studio, and more.
- **AI Teammate**: A collaborative AI teammate for engineering teams. Examples include Devin and Genie by Cosine.

This article focuses on the first group: CLI-based coding platforms. You'll explore the **top five** tools, compare their capabilities, and learn how to get started with each one.

What Is a CLI Coding Tool?

Think of a CLI-based AI coding tool as any LLM like Claude, OpenAI models, or Gemini in your Terminal. This category consists of closed and open-source tools that allow developers to work on engineering projects directly by accessing coding agents from model providers like Anthropic, OpenAI, xAI, and Google.

To understand how CLI tools differ, consider how IDE-based agents like Cursor work. You pick the agent you want to use in your project and add a prompt to begin interacting with it. Cursor then presents a UI to accept, reject, and review the agent's changes based on your prompt.

In contrast, CLI coding tools streamline that experience. You run commands directly through the Terminal at the root of your project. After the agent analyzes your code, it asks yes/no questions about the task without leaving the Terminal.

CLI-Based Coding Platforms vs. AI-First Code Editors

CLI-based coding platforms use a minimal interface to communicate with AI models. Since these tools have minimal UI, you get things done faster. There are no chunky interfaces for confirming changes to your source code when communicating through the command line.

By contrast, interacting with agents via AI code editors is sometimes difficult due to clunky interfaces. For example, accepting code sections and reviewing the entire code may be confusing in Cursor since it involves several clicks and UI changes.

Although developers can use both categories to work on large projects, command-line coding tools provide raw and low-level access to LLMs. They do not offer a graphical UI, so there's a steeper learning curve and more control.

Many developers do not consider the pricing and cost per session aspects of using these agentic coding tools, whether working alone or in a team. It is recommended that the pricing for using the model on each platform be checked before starting a project.

Although Gemini CLI, Codex CLI, and Aider are open-source and free to use, using Claude Code, for example, can be expensive.

If you do not regularly check the cost of running an agent, it can cost you about 6x Cursor's monthly cost of $20.

CLI Coding Tools Use Cases Across Teams

CLI-based agentic tools have many application areas, from engineering to research teams. They work well across both small and large code bases.

Use cases include:

- **Codebase reviews**: Answer questions, fix bugs, commit code changes, and create pull requests.
- **Vibe coding**: CLI coding tools excel at vibe coding, like traditional AI code editors. They make it quicker to skim, review, or supervise generated code easily.
- **Repetitive engineering tasks**: Automate the repetitive and boring development tasks teams often procrastinate.
- **Productivity booster for ICs**: Help individual contributors on your team increase their coding productivity by about 10x.
- **Finance**: Use these tools for non-coding functions like editing and improving data workflows, or even parsing CSVs for finance or operations teams.
- **Data teams**: Build production-ready dashboard apps for visualizing information.

- **Infrastructure teams**: Perform effective security analysis, approvals, and reviews.
- **Marketing:** Evaluate the performance of Ads and campaigns and suggest improvements.
- **DevRel:** Write, edit, review, and manage documentation and technical tutorials in a GitHub repo.
- **Product design team**: Perform visual edits using prompts and manage the state of designs.

The 5 Leading Command-Line Coding Platforms

As the number of AI coding editors increases weekly, several agentic coding tools for the command line may emerge soon. Still, as of this writing, three platforms stand out for their adoption, capabilities, and developer experience.

The next sections break down what each one offers and how to get started.

What is Aider?

Using Aider Chat

Think of Aider as an AI pair engineer in your command-line utility. Aider lets developers select the LLMs they prefer when creating a new project or working with an existing codebase. Unlike Claude Code in this category, it is open-source and has the most installs, GitHub stars, and more than 135 contributors.

Key Features of Aider

Being open-source, Aider has evolved to support many features that suit small and large code bases.

- **Get usage information:** Aider displays information about the sent and received tokens and the cost of your messages after each session. This feature helps you unlock which models are cheaper for your projects.
- **Local and cloud LLMs support:** Access state-of-the-art models like Claude 4 Opus and Sonnet, OpenAI o3, DeepSeek-R1–0528, and more.
- **Auto Git support:** Aider automatically commits your changes to GitHub.
- **Access via an IDE:** Use Aider through the integrated Terminal of IDEs like Cursor or GitHub Copilot.
- **Voice mode:** Instead of writing instructions to perform tasks in Aider, use your voice as input.
- **Linting support:** Automatically lint and test your code after changes.
- **Image and web contexts:** Use screenshots and web content to provide context when working with Aider.
- **Instant code changes:** Aider shows code changes directly in the Terminal and syncs them instantly in your IDE.

Get Started With Aider

Aider is available on Pypi and provides a simple installation. To start a new project or work with an existing one, use the following commands to install Aider and configure your preferred LLM. Check out its user guide to discover the various commands you can use for your projects.

```
python -m pip install aider-install
aider-install

# Change directory into your codebase
```

```
cd /to/your/project

# DeepSeek
aider --model deepseek --api-key deepseek=<key>

# Claude 3.7 Sonnet
aider --model sonnet --api-key anthropic=<key>

# o3-mini
aider --model o3-mini --api-key openai=<key>
```

The sample project below uses OpenAI's `o3-mini` model to generate a SwiftUI code for an existing codebase.

Generating SwiftUI Code with Aider

What is Claude Code?

Claude Code is a Terminal tool from Anthropic designed for agentic coding to help developers work faster using Claude models. It works across several environments, codebases, and languages.

Unlike Aider and Codex CLI, Claude Code is closed-source. The agent can modify files, fix errors, execute test commands in your codebase, and integrate them with GitHub Actions for automated pull request management and code reviews. Since it understands any codebase, you do not need to provide files and directories as context for your project.

It also has powerful built-in tools like `WebSearch`, `WebFetch`, and `MultiEdit` for editing your codebase. To ensure enhanced security, using some of these tools requires configuring the permissions of your working environment.

Claude Code also has a command-line SDK for integration with applications. With the CLI-based SDK, you can build AI systems and assistants that use the capabilities of Claude models.

Key Features of Claude Code

Claude Code stands out with its privacy and security by default and its ability to seamlessly integrate with enterprise applications. It makes it easy to manage security to safeguard your workflow and working environment. You can also define and specify robust permissions for tool access to restrict who can access what.

Integrate LLMs fast! Our UI components are perfect for any AI chatbot interface right out of the box. Try them today and launch tomorrow!

Top features of Claude Code include:

- **Connect third-party APIs**: Claude Code projects can securely integrate enterprises with applications and deploy them on Amazon Bedrock and Vertex AI. It also provides secure, built-in tools for user management for large teams.
- **Code privacy and security**: Your prompts and private data stay with your Anthropic's API dashboard only and are inaccessible to third-party servers.
- **Prompt injection prevention**: It provides ready-made tools to prevent any code base of enterprise teams from prompt injection, a technique used by attackers to manipulate an LLM's instruction.
- **Tracing and usage monitoring**: Claude Code supports OpenTelemetry, allowing you to observe and trace the performance of selected models and monitor their usage.
- **CLI commands**: Work effectively with Claude Code using CLI flags and slash commands such as `--model`, `/cost`, and `/review`. To learn more, check out the Claude Code guide.
- **Vim mode**: Easily switch to Vim mode (`/vim`) for Vim keybindings and command modes. You can also use the `/configure` slash command to set up your Vim mode.
- **Connect your favorite IDE**: Claude Code provides developers seamless integration with popular IDEs such as VS Code, Cursor, Windsurf, PyCharm, WebStorm, and IntelliJ for improved workflows.
- **Instant launch**: Press `cmd + esc` and `ctrl + esc` to quickly launch Claude Code from your Terminal.
- **Built-in memory**: Claude Code can keep information about your sessions, commands, and style guides in its memory. This feature helps it remember your project preferences and actions for faster workflow.

Get Started With Claude Code

Like the other CLI coding tools, Claude Code makes it easy to start and create your first project. However, unlike Aider, a Python package, Claude Code's installation requires Node 18+.

To start, run this command to install and authenticate it.

```
npm install -g @anthropic-ai/claude-code
```

The following preview demonstrates Claude Code's usage when writing a test.

Using Claude Code

Using Claude Code in a large team can be expensive if several people use it to contribute to a codebase. To monitor a team's usage closely, you can limit Claude Code Workflow spending. You can learn more about it in this Anthropic guide.

Use Claude Code With Non-Anthropic Models

Due to the incredible coding capabilities, developers love Claude models like 4 Opus and Sonnet, and even previous ones like 3.5 and 3.7 Sonnet.

Engineering teams and indie devs normally complain about these models because of their high running costs. As highlighted in some sections of this article, running 4 Opus or Sonnet in Claude Code can cost you 6x the monthly cost of Cursor. For this reason, you and your team may want to use Claude Code with non-Anthropic models like Gemini 2.5 Pro, which is much cheaper but has coding performance on par with the SOTA models of Anthropic.

Luckily, you can utilize Claude Bridge, allowing you to use Claude Code with Germini, OpenAI, xAI, and Ollama models.

Using gpt-4.1 with Claude Code

The example above uses Claude Code with OpenAI's **gpt-4.1** model.

What is Codex CLI?

Codex CLI is an experimental, lightweight, and fully open-source coding agent that lives and pairs programs with you in the Terminal.

You can collaborate with it to read and modify your source code and run tests using closed and open-source models. It uses the reasoning capabilities of OpenAI's o3 and o4-mini models and integrates them seamlessly into your local environment. However, you can bring your preferred models, such as Gemini 2.5 Pro, Claude 4 Opus, and Sonnet.

Key Features of Codex CLI

- **Easy setup**: Once your OpenAI API key is ready, you can start using Codex CLI by running only two commands.
- **Approval modes**: Decide how you want an agent to read and write files with approval modes such as Suggest, Auto Edit, and Full Auto. Like the other platforms, it can accept and approve changes safely and securely.
- **Multimodal input**: Start a conversation and implement features by passing low-fidelity sketches, diagrams, and images to your model.
- **Configure codebase standards**: When using Codex CLI, add an `AGENTS.md` file to the root of your project or in your repo. Like `README.md`, you can define in this file how an agent should navigate your codebase, testing commands, projects' style guide, standard practices, etc.
- **Security and permissions**: Codex CLI safeguards your applications against prompt injection and refuses malicious requests. With its built-in security, you

can define different levels of autonomy for an agent using the `--approval-mode` flag.

- **Low-latency code Q&A**: It uses `codex-1` as the default model, an optimized version of `o4-mini` that assists with faster code editing and workflows. You can also use the model via the OpenAI API (`codex-mini-latest`).
- **Sign in with ChatGPT**: Codex CLI supports logging in with ChatGPT. This feature lets you connect your developer account without manual API token generation and setup.

Get Started With Codex CLI

Similar to the other command-line AI coding platforms, creating your first project with Codex CLI requires running only two commands. Ensure you have Node.js 22 or newer (LTS recommended) installed. Globally install the tool and store your API key as an environment variable.

```
npm install -g @openai/codex
export OPENAI_API_KEY="your-api-key-here"
```

Now, you can use Codex CLI for your projects by running `codex` in your Terminal. The preview below uses Codex CLI to commit changes to a technical tutorial, create a new branch, and push it to GitHub.

Using Codex CLI

This pull request shows that the tutorial changes have been successfully pushed to GitHub by Codex CLI.

What is Gemini CLI?

Gemini CLI

Although Gemini CLI is late to the game, it makes up for it with one significant edge: it's open-source and can be accessed via your Terminal. The CLI agent understands your codebase and has the tools to navigate a large engineering project.

Its key features include the following:

- **Project automation**: Use it to automate complex rebases, pull requests, and GitHub issues.
- **Context window**: Use Gemini's one million context window to query the Terminal agent and work with large code bases.
- **MCP support**: Using different MCP tools and servers to extend the agent's capabilities.
- **App generation**: Like the other CLI coding tools, it can generate code for an app's UI and functionalities for different platforms based on images and sketches.

Gemini CLI Quick Start

To start with Gemini CLI, visit this repo and follow the steps. You can also install it with this command:

`npm install -g @google/gemini-cli gemini`

Before running the above command in your shell, ensure you have Node 18 or a later version on your machine. Once you run the command above, you can select a theme and follow a few other instructions to complete the installation. To start working with Gemini CLI, you should see a Terminal interface similar to this image.

Gemini CLI

As seen in the above image, you can instruct the Gemini CLI agent to provide a summary about updates to a specified repo. Watch this video to get up and running with Gemini CLI in under three minutes.

Get started with Gemini CLI

What is Warp?

Warp is one of developers' favorite Terminal utilities. Version 2.0 introduces support for agentic coding. You can use multiple agents to work on a large code base.

Its key features include the following:

- **Context:** Provide agents with MCP tools, rules, Warp Drive, and multi-repo databases to achieve accurate and satisfactory results.
- **Agent control**: Configure agents with different levels of autonomy.
- **Secure and enterprise-ready**: With Warp, no third-party model provider trains on your data. This feature makes it excellent for agentic CLI coding in an [enterprise](https://www.warp.dev/enterprise) setting.
- **Multi-agent support**: Run multiple AI agents on a large project in parallel.

Like the other platforms, you can install Warp with a single command:

`brew install — cask warp`

Once installed successfully, you can begin with Warp for agentic development in the Terminal.

Warp agentic development

What to Consider When Using CLI Coding Platforms

CLI coding assistants can help solve engineering tasks easily by giving developers raw access to an AI model without interacting with a UI. However, they also have tradeoffs.

Watch for Hidden Costs

Some tools, like Aider and Codex CLI, are open-source and free. Others, like Claude Code, can get expensive fast. Each prompt interaction consumes tokens, and daily use can easily exceed $6 per developer. In team settings, that adds up quickly. To track and monitor your costs, you should always check out your Anthropic dashboard's usage history. Additionally, you could consider setting a limit on your workspace spending for your Claude Code workspace.

Learning Curve and UX Tradeoffs

If you are not a power CLI user, interacting with these agents via the Terminal may look dull and intimidating. Unlike AI IDEs like Cursor, CLI platforms offer no graphical interface for organizing files or visualizing agent suggestions. That can make managing extensive engineering projects, where context is crucial, more challenging.

If that's a concern, consider Cursor for managing large projects instead. For example, Cursor easily tags files and folders for a precise response.

Set Your Project Up for Success

To get the best out of these Terminal coding assistants, always start your projects with `AGENTS.md` or `Claude.md` files when using Codex CLI or Claude Code.

You can use them to specify things like:

- Test instructions.
- Core files.
- Code styling.
- Guidelines.
- Other important information you want an agent to remember in the files.

Whenever you send a query, the coding assistant retrieves the information in the files to guide its operations and help it respond with accurate results.

The Future of CLI Coding Platforms

Command-line AI coding tools like Aider, Claude Code, and Codex CLI help developers work faster and improve workflows with task delegation and automation. They are direct alternatives to leading AI code editors such as VS Code and Cursor, and they open a new landscape for engineers to collaborate with AI agents. As engineering teams and indie developers gradually adopt these tools, you should prepare for a future where agentic coding is done primarily within your Terminal.

As these platforms evolve, expect stronger integration with third-party APIs, improved security features, and support for real-time interaction, possibly even voice collaboration. With better enterprise-grade controls and broader model support, CLI agents are poised to become an essential part of the modern dev stack.

. . .

Originally published at https://getstream.io.

AI Agents Llm Artificial Intelligence Programming

Follow

Written by Amos Gyamfi

1.5K followers · 122 following

Developer Advocate @getstream.io | AI, Python & Swift/UI Content Creator

Responses (4)

Bgerby

What are your thoughts?

S. Kong he
Jul 9

Thanks for the detailed comparison of CLI-based coding tools! I found the breakdown of Aider's features, like its support for local and cloud LLMs and auto Git integration, particularly insightful. It's great to see how these tools can streamline coding tasks directly from the terminal.

Reply

Raf Turek
Jun 19

For Claude Code you can use Your payed subscription pro or max instead of API.
Max is almost unlimited pro it's just to have a taste and lower cost instead of payed API You use your conversation limit like in normal chat with Claude
Definitely... more

Reply

Joseph U.
Jun 12

If you are on the pro or max plan you won't be paying extra for daily use of Claude Code beyond your monthly plan.

1 reply Reply

See all responses

More from Amos Gyamfi

Amos Gyamfi

Build Your First Voice and Video Call AI Agent in Python

In this step-by-step quick-start guide, we will use Vision Agents, build and run a real-time voice AI agent that joins a video/audio call...

Oct 18 24

Amos Gyamfi

Top 5 Realtime Speech-to-Speech APIs and Libraries To Build Voice Agents

Building an AI app for real-time audio and video streaming over WebRTC or WebSockets is complex. Think turn-taking, noise cancellation, and...

Oct 8 25

Amos Gyamfi

Foundation Models Framework: Get Started With On-Device AI in Xcode 26

Apple announced the Foundation Models Framework (FMF) during WWDC25. This tutorial will guide you in quickly interacting with the model to...

Jun 15 32 2

Amos Gyamfi

The Top 7 MCP-Supported AI Frameworks

Create AI apps with Python and Typescript frameworks that leverage MCP servers to provide context to LLMs.

Apr 1 1.2K 23

See all from Amos Gyamfi

Recommended from Medium

In CodeToDeploy by TechToFit - Master Your Life with Tech

I Tried Google's New AI Agents. It's a Gold Rush.

This is one of those times.

Oct 10 172 3

Riccardo Tartaglia

5 Essential MCP Servers Every Developer Should Know

I’ve been experimenting with Model Context Protocol servers for a few months now, and I have to say, they’ve changed the way I work.

Oct 11 26 3

In AI Software Engineer by Joe Njenga

Why Claude Weekly Limits Are Making Everyone Angry (And $100/Month Plan Will Not Save You)

Yesterday, I finally hit my weekly Claude limit, and I wasn't surprised, since I see dozens of other users online going crazy over these...

5d ago 119 20

In Dare To Be Better by Max Petrusenko

Claude Skills: The $3 Automation Secret That’s Making Enterprise Teams Look Like Wizards

How a simple folder is replacing $50K consultants and saving companies literal days of work

Oct 17 283 4

In Stackademic by Somendradev

Top Open-Source Tools to Supercharge Your Coding Workflow

There’s something magical about open-source tools — built by developers, for developers. They’re not just free; they often outshine many...

Oct 11 35

In Towards Deep Learning by Sumit Pandey

Why Everyone Will Want DGX Spark on Their Desk — Yes, Everyone

I just saw this picture today and was amazed, I've been waiting for this moment for a long time. (No, it's not Elon.) It's that tiny...

Oct 14 26 9

See more recommendations