Claude Code in VS Code: How to use AI for efficient multi-file programming

This guide explains how to use Claude Code in VS Code, an AI tool that can understand codebases, edit multiple files, and perform automated testing.

Not all AI programming tools are the same. Currently, they can be divided into two distinct groups. On one side are the familiar autocomplete tools, responding line by line as you type. On the other side are systems capable of understanding the entire codebase, independently planning, editing multiple files, and running tests before producing results.

Anthropic positions Claude Code in the second category. It is an 'agentic coding' tool – meaning it not only suggests but can directly participate in the software development process.

The Claude Code extension in Visual Studio Code brings this entire workflow directly into the IDE, instead of requiring you to operate through the terminal as before.

How Claude Code works in VS Code

This extension is essentially a graphical user interface layer that runs on top of Claude Code's CLI. The CLI is already integrated during installation, so no further configuration is required.

When activated, the system will run an internal MCP server to connect the IDE and Claude. This allows the agent to read the open file, understand the current context, and display changes as diffs directly within VS Code.

The interaction is also quite intuitive. Users simply send a prompt, Claude reads the relevant files, suggests changes, and displays them as a side-by-side comparison. Importantly, all changes must be approved before being written to the code.

How to install Claude Code in VS Code

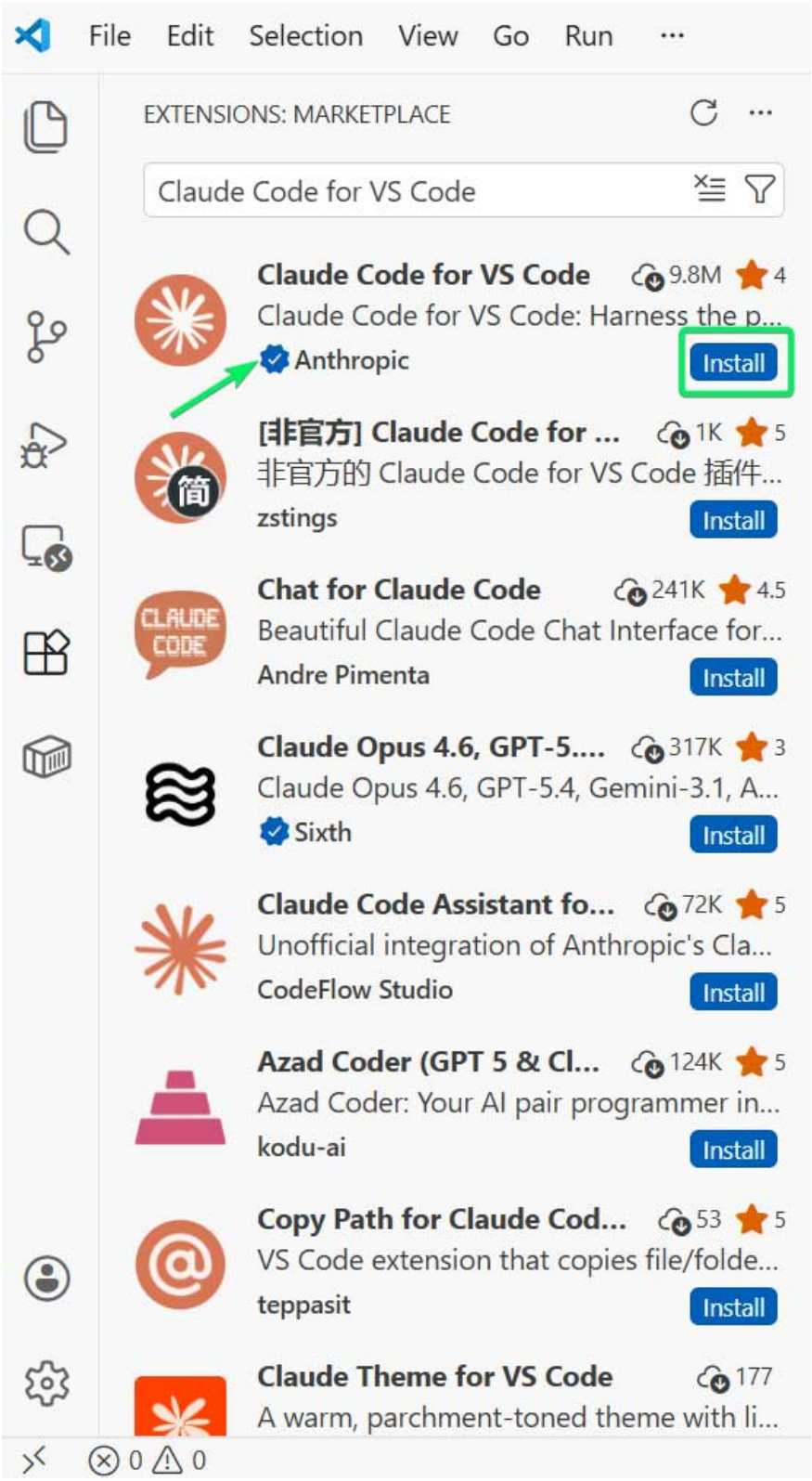
To begin, ensure you have VS Code version 1.98 or later and an Anthropic account. No Node.js installation is required if using the official version.

The installation process is similar to other extensions. After installation, open a file in your project, log in to your account, and the Claude icon will appear in the editor interface.

Here are the installation instructions:

1. Press **Cmd+Shift+X** on a Mac or **Ctrl+Shift+X** on Windows and Linux to open the Extensions window.
2. Search for " **Claude Code** " and install the extension released by Anthropic.

One thing to note is that Claude Code is not included in the free plan. Users need at least a Pro plan to use it, and if used frequently, higher plans would be more suitable due to usage limits.



CLAUDE.md: Important 'project memory'

After installation, creating the CLAUDE.md file in the root directory is almost mandatory if you want to optimize performance.

This is where information about:

1. system architecture
2. coding conventions
3. build and test methods
4. Patterns that should or shouldn't be used

Claude will read this file at the start of each session, reducing the need for repeated explanations. Users can use commands `/init`to have the agent automatically create a draft based on the existing codebase, and then modify it accordingly.

What can Claude Code do?

Unlike typical AI coding tools, Claude Code operates at the project level.

Users can describe a feature using natural language, and Claude will automatically plan the implementation and modify related files. When code needs to be understood, simply call the file or folder name, and the agent will automatically find and analyze it.

A major strength lies in its ability to handle multiple files. Each change is displayed as a diff so users can review it before applying it.

Additionally, Claude can write tests, run tests, and debug within the same session. With debugging, it can directly read errors in the IDE without needing to copy and paste.

Claude Code supports various modes depending on the desired level of control. In default mode, all actions require confirmation. When switched to plan mode, Claude analyzes the codebase and creates a detailed plan for the user to review before execution.

For those familiar with the workflow, auto-accept can be used to skip the step of confirming each change. There's even an auto mode – where the system automatically decides on an action based on an internal classifier, although this feature is currently still in the experimental stage.

The most significant difference between Claude Code in VS Code and the terminal lies in the review experience.

In the terminal, reading diffs is often quite difficult, especially with large changes. But in VS Code, each file has its own tab, making it easy to inspect each change. This is especially important when refactoring a large system. Claude can rename functions or modify logic across the entire project, while the user can visually inspect each part individually.

Debugging and refactoring: a new approach.

In debugging, Claude doesn't replace traditional debuggers, but it handles common errors very well through log reading and context analysis.

Users can provide the error, or let Claude read it from the Problems panel. The agent will then find the cause and suggest a fix in the form of a diff.

With refactoring, Claude is particularly useful when you need to change many files at once, such as transferring libraries or changing the architecture. However, its effectiveness depends heavily on the quality of the CLAUDE.md file.

Comparison with workflow terminal

The VS Code extension provides a more intuitive experience with a diff viewer, plan mode, and the ability to manage multiple sessions.

However, the terminal still has advantages in terms of automation capabilities, support for bash commands, pipelines, and advanced MCP server configuration.

In practice, many developers use a combination of both: VS Code for tasks requiring thorough review, and the terminal for quick and automated tasks.

Comparison with GitHub Copilot

One point of confusion is that Claude in VS Code can appear in several forms, including integration via GitHub Copilot.

However, Copilot and Claude Code serve two different purposes. Copilot focuses on autocomplete and inline support, while Claude Code is geared towards handling large, multi-file tasks with a clear plan.

Therefore, many programmers use both in parallel: Copilot to speed up code writing, and Claude Code to handle more complex problems.

Limitations to note

Despite its power, Claude Code still has certain limitations.

Currently, change approval only applies to files, not to individual segments. Additionally, usage limits are a problem if used continuously for extended periods. On Windows, errors can sometimes occur due to file conflicts when VS Code and Claude access the same file. Furthermore, with long sessions, context compression can cause the agent to 'forget' previous decisions, so proactive management using commands like `` is necessary `/compact`.

Claude Code in VS Code isn't just a smarter autocomplete tool, but a completely different approach to AI programming. Instead of assisting with each line of code, it participates directly in the entire process: from planning, editing, testing to review.

In the context of the ever-evolving coding agent landscape, the greatest value lies not in telling the AI ??what to do, but in the ability to control and evaluate what it has done. And that's precisely what Claude Code is trying to address.

and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.