

Copilots

There are a number of AI tools that can improve the developer experience. This article will discuss tooling that is available as well as advice on when it might be appropriate to use such tooling and **how to attribute AI-generated code.**

AI-Assisted Coding Tools

GitHub Copilot Extension

GitHub Copilot is an extension that adds AI-assisted coding capabilities to an IDE. For instance, the [GitHub Copilot for VS Code extension](#) can be installed from the VS Code Marketplace. It requires a GitHub account to use, and usage is limited without a subscription. For more information about what IDEs are supported, what languages are supported, cost, features, etc., please check out the information on [Copilot](#) and [Copilot for Business](#).

As of late 2025, some example use cases for GitHub Copilot include:

- **Code Completion.** Copilot can provide code completion suggestions as you type. If you want Copilot to write something useful for you, try writing a comment that describes what your code is going to do - it can often take it from there.
- **Write Documentation.** For example, the above paragraph was written using Copilot.

- **Write Unit Tests.** Given that setup and assertions are often consistent across unit tests, Copilot tends to be very accurate.
- **Unblock.** It is often hard to start writing when staring at a blank page. Copilot can fill the space with something that may or may not be what you ultimately want to do, but it can help get you in the right headspace.
- **Explain.** Copilot can explain what the code is doing in natural language.
- **Generate Tests.** Copilot can generate unit tests for your code.

You can also chat with Copilot to ask it to edit your code, including across multiple files. Beyond just writing code, consider using chat to:

- **Build SQL Indexes.** Given a query, Copilot can generate a SQL index that will improve the performance of the query.
- **Write Regular Expressions.** These are notoriously difficult to write, but Copilot can generate them for you if you give some sample input and describe what you want to extract.
- **Improve and Validate.** If you are unsure of the implications of writing code a particular way, you can ask questions about it. For instance, you might ask if there is a way to write the code that is more performant or uses less memory. Once it gives you an opinion, you can ask it to provide documentation validating that assertion.
- **Write Code.** Given prompting by the developer it can write code that you can one-click deploy into existing or new files.
- **Debug.** Copilot can analyze your code and propose solutions to fix bugs.

You can also use configuration files and extensions to customize and extend Copilot's capabilities. For example:

- **Custom instruction files.** You can create [custom instructions files](#) that provide Copilot with additional context about your project, such as [coding conventions, preferred libraries, or specific patterns you want it to follow](#).
- **Prompt files.** You can create [prompt files](#) to store reusable prompts or templates that can be easily accessed from the Copilot chat interface.
- **MCP Servers.** You can use [MCP servers](#) to expand the capabilities of Copilot by [providing it access to external tools or services, such as connecting to a project management system \(such as Jira or Azure DevOps\) or custom APIs \(such as the Azure resource manager API\)](#).

GitHub Copilot Coding Agent

The [GitHub Copilot Coding Agent](#) is an AI software development agent that can be assigned to work on issues within the GitHub work management system such as fixing bugs or implementing new features. [Once an issue has been assigned to the Copilot Agent, the bot analyzes the work item, creates a branch, starts a virtual environment to execute and test the code iteratively, authors commits, and opens a pull request for review.](#)

ChatGPT / Copilot Chat

For coding, generic AI chat tools such as ChatGPT and Copilot chat are less useful, but they still have their place. GitHub Copilot will only answer "questions about coding" and its interpretation of that rule can be a little restrictive. Some cases for using ChatGPT, Copilot chat, or M365 Researcher include:

- **Write Documentation.** Copilot can write documentation, but using ChatGPT or Copilot chat, you can expand your documentation to include

business information, use-cases, additional context, etc.

- **Change Perspective.** ChatGPT can impersonate a persona or even a system and answer questions from that perspective. For example, you can ask it to explain what a particular piece of code does from the perspective of a user. You might have ChatGPT imagine it is a database administrator and ask it to explain how to improve a particular query.

Prompt Engineering

Chat AI tools are only as good as the prompts you give them. The quality and appropriateness of the output can vary greatly depending on the prompt. In addition, many of these tools restrict the number of prompts you can send in a given amount of time. To learn more about prompt engineering, you might review some [background open source prompt engineering documentation](#) and how it applies to [crafting prompts for Copilot chat](#).

Considerations

If you make use of AI tools, it is important to understand how the data (including private or commercial code) might be used by the system. You can read more about how GitHub copilot handles your data and code at the [GitHub Copilot Privacy FAQ](#) and how to [manage Copilot policies](#). Many organizations have policies that restrict the use of AI tools when working with proprietary code or data, so it is important to check for applicable policies before using these tools.

Attributing AI-Assisted Code Authorship

When using AI tools to generate code, it can be beneficial to maintain transparency about authorship for accountability, code review, and auditing

purposes. This can be done easily by using [Git trailers](#) that append structured metadata to the end of commit messages.

Agentic Code Authorship

When using agentic AI tools to generate code, such as the GitHub Coding Agent feature where a GitHub issue (such as a story or task) can be assigned to a software engineering bot that creates a branch, writes commits, and opens a pull request, authorship is typically attributed using tool-specific mechanisms that follow conventions similar to pair programming. For example, the GitHub Coding Agent creates its commits using the agent as an author with the supervising person acknowledged using the `Co-authored-by` trailer in the commit message. This allows AI-generated code to be attributed to the agent while still giving credit (and accountability) to the person who supervised or initiated the task.

IDE Assistant Attribution

When using an AI coding assistant inside an IDE, such as the GitHub Copilot extension in Visual Studio Code, consider adopting a team convention to provide attribution in cases when the commit consists of primarily AI-generated code. This might occur when:

- More than 50% of the lines in the commit were generated by AI
- The AI provided the core logic or algorithmic approach
- Substantial code blocks were accepted from AI suggestions

This can be done by appending one or more custom trailers in the commit message, such as:

```
Assistant-model: GPT-4o
```

Because most Git tooling expects `Co-authored-by` trailers to be formatted as email addresses, you should use a different trailer key to avoid confusion and to distinguish authorship from assistance.

Trailers can be added manually at the end of a commit message, or by using the `git commit` command with the `--trailer` option:

```
git commit --message "Implement feature" --trailer "Assistant-  
model: GPT-4o"
```

Trailers can be displayed using the `pretty formats` option to `git log` command. For example, for a formatted history showing the hash, author name, and assistant models used for each commit:

```
git log --color --pretty=format:"%C(yellow)%h%C(reset)  
%C(blue)%an%C(reset) [%C(magenta)%(trailers:key=Assistant-  
model,valueonly=true,separator=%x2C)%C(reset)] %s%C(bold  
cyan)%d%C(reset)"
```

```
2100e6c Author [GPT-4.1,Claude Sonnet 4] Test commit 4 (HEAD ->  
work-item-8)  
7120221 Author [GPT-4.1] Test commit 3  
ea03d91 Author [] Test commit 2  
f93fd8e Author [GPT-4o] Test commit 1  
dde0159 Copilot [] Test work item (#7) (origin/main,  
origin/HEAD)
```