ISSN: 2836-8495

Review Article

Journal of Current Trends in Computer Science Research

A Comparative Featureset Analysis of Agentic IDE Tools

Mohith Shrivastava*

Independent Researcher, United States

***Corresponding Author**
Mohith Shrivastava, Independent Researcher, United States.

Submitted: 2025, May 16; **Accepted:** 2025, June 12; **Published:** 2025, June 25

Citation: Shrivastava, M. (2025). A Comparative Featureset Analysis of Agentic IDE Tools. *J Curr Trends Comp Sci Res, 4*(3), 01-12.

Abstract
The rapid evolution of Artificial Intelligence (AI) is profoundly reshaping software development practices, with sophisticated Agentic Integrated Development Environment (IDE) tools emerging as pivotal innovations. These tools transcend traditional coding assistance, offering proactive and au-tonomous capabilities that can significantly enhance developer productivity and workflow efficiency. This paper presents a comprehensive comparative featureset analysis of eight prominent agentic IDE tools as of mid-2025: GitHub Copilot, Cursor, Sourcegraph Cody, Tabnine, Amazon Q Developer, WindSurf (post-acquisition by OpenAI), Augment Code, and Cline. The analysis incorporates an expanded scope, evaluating these tools on their core agentic functionalities and, critically, their pro- visions for background and remote agent capabilities, which enable more complex and asynchronous task automation. Furthermore, this research investigates the support for the Model Context Proto- col (MCP) as a key dimension for extensibility and interoperability. For Cline, particular attention is given to its strengths and operational model as an open-source solution. The findings highlight the diverse approaches to agentic AI in software development, the varying degrees of autonomy and intelligence offered (including asynchronous and remote operations), and the growing importance of standardized protocols like MCP. This research provides an updated snapshot of a dynamic field, offering valuable insights for practitioners, researchers, and decision-makers navigating the evolving landscape of AI-powered software engineering features.

Keywords: Agentic AI, Integrated Development Environments, AI Coding Assistants, Software De-velopment Tools, Large Language Models, Code Generation, Developer Productivity, Comparative Fea-tureset Analysis, Background Agents, Remote Agents, Model Context Protocol (MCP), Open Source Software

1. Introduction

1.1. Context and Motivation

The integration of Artificial Intelligence (AI) into software development has progressed from basic code completion to sophisticated "agentic" functionalities within Integrated Development Environments (IDEs) [1]. This evolution marks a critical juncture, promising to redefine developer workflows and pro-ductivity. Agentic IDE tools are designed to function not only as assistants but as proactive or even autonomous partners capable of understanding high-level goals, planning complex actions, interacting with the development environment, and performing tasks that traditionally required significant human intervention [1]. A significant new frontier in this domain is the emergence and maturation of background and remote agents. These capabilities extend the operational scope of AI assistants beyond immediate, interactive IDE tasks, enabling more complex, long-running, and asynchronous automation. The dynamism in this field, characterized by rapid advancements in Large Language Model (LLM) ca-pabilities, evolving feature sets, significant market shifts (e.g., acquisitions), and the introduction of new interoperability protocols like the Model Context Protocol (MCP), necessitates continuous evaluation of tool *featuresets* to maintain academic and practical relevance [1]. This paper provides an updated comparative featureset analysis of leading agentic IDE tools, building upon foundational research and incorporating the latest developments as of mid-2025.

1.2. The Ascendance of Agentic AI in Software Development

"Agentic AI" in the context of software development refers to systems exhibiting a degree of autonomy, planning capabilities, the ability to use tools (including file system access, terminal command execution, API interactions), and adaptive behaviors based on context and feedback [1]. This paradigm shift moves beyond passive assistance towards creating AI partners that can independently contribute to the software lifecycle. The potential benefits include accelerated development cycles, improved code quality, and the automation of repetitive or complex tasks, allowing developers to focus on higher-level problem-solving and innovation [1].

Background and remote agents significantly extend this paradigm. For instance, the GitHub Copilot coding agent can autonomously work on GitHub issues, plan changes, and open pull requests [2]. Augment Code's Remote Agents operate in the cloud, capable of handling concurrent development tasks and delivering ready-to-merge pull requests even after the developer has logged off [3,4]. Amazon Q Developer offers specialized agents for complex, long-running tasks like Java application upgrades or.NET porting, operating with a high degree of autonomy [5]. These examples illustrate a move towards AI systems that can manage substantial portions of the development workflow asynchronously and with minimal direct supervision.

1.3. Scope and Objectives

This research undertakes a comprehensive review and comparative featureset analysis of eight prominent AI-powered coding assistants: GitHub Copilot, Cursor, Sourcegraph Cody, Tabnine, Amazon Q Devel-oper, windsurf (considering its pre-acquisition features and current status), Augment Code, and Cline.

1.3.1. The primary objectives are:

1. To provide an updated overview of each tool's capabilities, focusing on their agentic functionalities, including background/remote agent capabilities, as of mid-2025.
2. To compare these tools across key dimensions: core agentic features, background/remote agent features, code generation, debugging/refactoring assistance, context awareness, integrations, per- formance, extensibility (with a specific focus on Model Context Protocol (MCP) support), and security/ privacy.
3. To analyze the impact of recent market developments, including acquisitions, and the role of Cline as an open-source exemplar in the agentic IDE landscape.
4. To offer insights into the current state and potential future di-rections of agentic IDE tools, focusing on *feature trends* rather than the qualitative assessment of AI-generated responses.

The analysis is based on publicly available documentation, tech-nical blogs, academic papers, and industry reports from the 2024-2025 period. This study aims to provide a clear understanding of the *features offered* by these tools, differentiating it from research that might evaluate the accuracy or quality of the AI models them-selves.

2. Defining Agentic IDE Tools and Their Operational Modali-ties: A Mid-2025 Perspective

2.1. Conceptual Framework for Agentic Capabilities

An agentic IDE tool, as understood in mid-2025, embodies a proactive or autonomous partner within the software development lifecycle. Key attributes defining such a tool include [1]:

- **Goal-Oriented Autonomy:** The capacity to comprehend high-level developer intent and work towards specified goals with reduced direct human intervention.
- **Planning and Task Decomposition:** The ability to break down complex tasks into manageable sub-tasks and formulate a coherent plan of action.
- **Tool Use and Environmental Interaction:** The capability to interact with the broader devel-opment environment, including file systems, terminals, version control, and external APIs. The Model Context Protocol (MCP) is increasingly pivotal for standardizing tool interaction and en-abling more modular and extensible agentic systems [1,6-8]. MCP allows AI applications to connect with external data sources and tools in a standardized way, akin to a "universal remote" for AI [6].
- **Learning and Adaptation (Emerging):** Adaptive behaviors based on project-specific context, coding patterns, user feedback, and defined rules.

A critical distinction exists between true agentic behavior and advanced autocompletion or single-turn chat-based code generation. Agentic tools are characterized by their initiative and ability to perform sequences of interconnected actions to achieve a broader goal [1].

2.2. Operational Modalities: IDE-Integrated vs. Background/ Remote Agents

Agentic IDE tools can be broadly categorized based on their primary mode of operation:

2.2.1. IDE-Integrated Agents: These agents primarily operate in real-time within the developer's Inte-grated Development Environment. Their focus is on augmenting the interactive coding experience.

- **Characteristics:** Direct interaction with the developer, synchronous responses, focus on current file or narrowly defined context, assistance with tasks like code completion, interactive chat for Q&A or code generation, and refactoring suggestions within the active editor session.
- **Examples:** GitHub Copilot's inline suggestions and chat in the IDE,Tabnine's code completions, Sourcegraph Cody's interactive chat and inline edits, and the general chat functionalities of most tools when used for immediate queries.

2.2.2. Background/Remote Agents: These agents are designed to operate more autonomously, often asynchronously, and potentially in environments separate from the local IDE. They are suited for more complex, multi-step, or longer-running tasks that may not require constant developer supervision.

- **Characteristics:** Asynchronous execution, capability for parallel tasking, ability to continue oper- ations after the IDE is closed or the developer switches context, interaction with broader systems (e.g., version control systems, issue trackers, CI/CD pipelines, cloud resources), and often a higher degree of autonomy in planning and executing tasks.

Examples:

- GitHub Copilot's ”coding agent” works autonomously in a GitHub Actions-powered environ- ment to complete tasks assigned through issues or chat, subsequently opening pull requests [2].
- Cursor's “Background Agents (Beta)” allow users to spawn asynchronous agents that edit and run code in a remote, isolated machine, with capabilities to monitor status and intervene [9].
- Augment Code's “Remote Agents” are purpose-built cloud workers in containerized environ- ments that can continue coding after the user logs off, delivering ready-to-merge PRs and handling multi-repo projects [3,4].
- Amazon Q Developer features agents that can autonomously perform tasks like Java upgrades and.NET porting, implying significant background processing for these transformations [1,5].
- Tabnine's specialized agents, such as the “Jira Implementation and Validation Agents,” are designed to generate code autonomously from requirements in Jira issues, suggesting an asyn- chronous, task-specific background operation [10,11].
- WindSurf's (pre-acquisition) “Cascade Write Mode” functioned like AutoGPT, creating mul- tiple files, running scripts, testing, and debugging with a high degree of automation [12].

The Model Context Protocol (MCP) plays an especially crucial role for background/remote agents. As these agents often need to interact with a wider array of external systems and data sources autonomously (e.g., project management tools, repositories, CI/CD pipelines), MCP provides a standardized mecha- nism for these interactions, enhancing their capability and extensibility [1,6].

3. Featureset Review of Agentic IDE Tools (Mid-2025)

This section provides a detailed featureset analysis of each selected agentic IDE tool, with a focus on their agentic capabilities, particularly background/remote operations, and their support for the Model Context Protocol (MCP).

3.1. GitHub Copilot (Microsoft)

3.1.1. Overview: GitHub Copilot is a widely adopted AI pair-programmer that has significantly evolved, now incorporating more autonomous features like the ”GitHub Copilot coding agent” [1,2].

3.1.2. Agentic Capabilities:

- **Core:** Offers advanced code completion, chat-based assistance (”Copilot Chat”), and an ”agent mode” in IDEs for multi-file edits, test execution, and bug fixing [1].
- **Background/Remote Agent:** The “Copilot coding agent” operates autonomously in the background within a GitHub Actions-powered environment [2]. It can be assigned tasks via GitHub issues or Copilot Chat prompts to fix bugs, implement features, improve test coverage, update documenta-tion, and address technical debt. The agent evaluates the task, makes changes, and opens a pull request [2]. This is distinct from the IDE's “agent mode” [2]. It uses GitHub Actions minutes and Copilot premium requests [2]. Limitations include working on a single repository per run and not signing commits by default [2].

3.1.3 Context & LLMs: Supports large context windows (e.g., GPT-4.1 up to 1M tokens), Retrieval Aug- mented Generation (RAG) via GitHub code search, Copilot Spaces, and semantic indexing. Offers a choice of LLMs including those from OpenAI, Anthropic, and Google [1,13].

- **Model Context Pro- tocol (MCP) Support:** Yes. Copilot coding agent supports tools provided by local MCP servers [1,14]. This allows extending its capabilities by connecting to other tools and services, such as Pieces for Long-Term Memory (LTM) [15]. Configuration allows for broader repository access beyond the default single repository context [2,14]. The Copilot coding agent specifically supports tools from MCP servers, not resources or prompts, and currently only local MCP servers [14].

- **Extensibility:** Open-sourcing of Copilot Chat in VS Code, Copilot Extensions (which can include custom Copilot agents associated with GitHub Apps), and MCP support enhance its extensibility [1,13].

- **Security/Privacy:** Features the GitHub Copilot Trust Center, SOC 2 compliance for Business tier, and a commitment not to train on Business/Enterprise customer data [1].

3.2. Cursor (Cursor AI Editor by AnySphere)

3.2.1. Overview: Cursor is an AI-first code editor, a fork of VS Code, with deeply integrated AI functionalities designed for agentic operations [1].

3.2.2. Agentic Capabilities:

- **Core:** “Agent mode” is the default, capable of codebase exploration, planning, multi-file edits, and terminal command execution [1,16]. It follows a systematic approach: understanding requests, exploring the codebase, planning changes, executing changes, and verifying results [16].
- **Background/Remote Agent:** Offers “Background Agents (Beta)” which are asynchronous agents that can edit and run code in a remote, isolated machine (Ubuntu-based image on Cursor's AWS infrastructure) [9]. Users can view their status, send follow-ups, or take over. These agents clone repos from GitHub (other providers planned), work on separate branches, and can have their environments customized with install commands, background terminal processes, or even Dock- erfiles [9]. This feature currently requires privacy mode to be

disabled and is available for Max Mode-compatible models [9].

3.2.3. Context & LLMs: Handles large contexts (up to 200K tokens), uses '@folders' and 'cursor.rules' for project-specific guidance. Supports a wide array of LLMs from OpenAI, Anthropic, Google, and xAI [1].

- **Model Context Protocol (MCP) Support:** Yes, extensive support. MCP is viewed as a plugin system for Cursor, allowing the Agent to connect to various data sources and tools through standardized interfaces [1,17]. Cursor supports one-click installation for curated MCP servers with OAuth and can connect to servers via CLI (Node.js, Python) or HTTP+SSE transport types [17]. The Composer Agent automatically uses relevant MCP tools, and users can intentionally prompt tool usage [17]. Known caveats include a limit on the number of tools sent to the Agent (first 40) and potential issues with remote development environments (SSH) as MCP servers are communicated with from the local machine [17]. MCP resources (read-only data) are not yet supported, only tools [17].

- **Extensibility:** VS Code extension compatibility and enhanced MCP support for external tools are key [1].

- **Security/Privacy:** SOC 2 Type II certified. Privacy Mode offers zero data retention for Business/Enterprise tiers [1]. However, Background Agents require privacy mode off, and their environment on Cursor's AWS has not yet been third-party audited [9].

3.3. Sourcegraph Cody (Cody AI by Sourcegraph)

3.3.1. Overview: Cody is an AI coding assistant that leverages Sourcegraph's code search capabilities for deep codebase understanding, aiming to assist across the software development lifecycle [1,18]. It is open source under Apache 2.0 [19].

3.3.2. Agentic Capabilities:

- **Core:** Features "Agentic chat" which autonomously gathers context from the codebase, termi-nal, web, and external tools (via OpenCtx/MCP) before responding [1,19-21]. It uses RAG with Sourcegraph search for context [1,22]."Auto-edit" provides proactive, context-aware changes, acting as an advanced autocomplete [1,20]. The "Coding Agent" feature aims to accelerate devel- opment in complex codebases [20], and can execute terminal commands with permission if deemed necessary to answer a prompt [19].
- **Background/Remote Agent:** Cody does not offer a distinct "background" or "remote" agent fea-ture in the same vein as GitHub Copilot's coding agent or Cursor's/Augment's remote execution environments. While "Agentic chat" involves autonomous context gathering that can be an asyn-chronous process , and the "Coding Agent" implies advanced automation, these primarily enhance interactive sessions rather than executing long-running, independent tasks in a separate environment [21]. There is no explicit mention of agents that continue working after IDE closure or perform large-scale, predefined asynchronous tasks like code migrations autonomously. Long-term memory can be simulated via rule files in VS Code (experimental) or by manually referencing documents in IntelliJ [23].

3.3.3. Context & LLMs: Sophisticated RAG using Sourcegraph search and its context engine [1,22,24]. Supports large context models (e.g., Gemini 1.5 Flash 1M tokens) and a wide choice of LLMs includ- ing Anthropic, OpenAI, Google, and Mixtral [1,20,25]. Context can be pulled from local and remote codebases, and even non-code tools like Notion and Linear [20].

- **Model Context Protocol (MCP) Support:** Yes, Sourcegraph Cody implements MCP through OpenCtx [1,6,26]. OpenCtx is a protocol for providing contextual information from various sources (code, documentation, logs, etc.) to both humans and AI, including Cody [26]. This allows Cody to integrate with other platforms and access a broader range of context.

- **Extensibility:** Customizable Prompt Library, MCP and OpenCtx inte- gration, and local inference with Ollama (experimental) [1,25].

- **Security/Privacy:** Offers on-premises deployment options, SOC 2 Type II, and ISO 27001:2022 certification. Sourcegraph states it does not train on customer code [1,20].

3.4. Tabnine (Tabnine AI Code Assistant)

3.4.1. Overview: Tabnine is an AI code assistant with a strong focus on privacy, personalization, and enter- prise needs, offering various deployment options including fully air-gapped on-premises solutions [1,10].

3.4.2. Agentic Capabilities:

- **Core:** Provides context-aware code completions and AI-powered chat [27]. Its "Enterprise Context Engine" uses multi-faceted RAG for deep organizational context [1,28].
- **Background/Remote Agent:** Tabnine offers several "Specialized AI Agents" designed to automate specific parts of the development lifecycle [1,10]. These include:
- **Jira Implementation and Validation Agents:** Generate code autonomously from re- quirements captured in Jira issues and validate AI- or human-generated code against these requirements [10,11]. This implies asynchronous, task-specific background work.
- **Code Review Agent:** Reviews code at pull request and in the IDE, providing fixes based on company standards [10].
- **Testing Agent:** Generates comprehensive test plans and test cases [10].
- **Documentation Agent:** Creates documentation for selected code, including formal API guides and inline comments [10,29].
- **Code Fix Agent:** Autonomously generates fixes for selected code with errors [10].

- **Code Explain and Onboarding Agent:** Explains projects, behaviors, and dependen- cies [10].

The operational mode (e.g., fully asynchronous, remote execution environment) of these specialized agents is not detailed with the same explicitness as Cursor's or Augment's remote agents, but their description, particularly for the Jira agents, points towards autonomous background processing. Some general comparisons suggest Tabnine can operate asynchronously in the cloud for certain features [30].

3.4.3. Context & LLMs: The "Enterprise Context Engine" leverages RAG based on local IDE workspace and connected organizational code repositories [1,28]. Tabnine supports switchable LLMs, including models from Claude, GPT-4o, Gemma, and Tabnine+Mistral [1,11]. It also allows for bespoke models trained on an organization's private codebase for enterprise customers [10,31].

- **Model Context Protocol (MCP) Support:** While general articles discussing MCP list Tabnine as a potential example of an "MCP client" , specific Tabnine documentation provided [1,8,10,27,28,31,32,52] does not contain explicit details about its own MCP server/client implementation or how its specialized agents might utilize MCP. Tab- nine's extensibility and context mechanisms appear primarily focused on its internal "Enterprise Context Engine," RAG capabilities, and its Usage API for enterprise reporting [32]. There is no developer docu- mentation found on a specific Tabnine MCP implementation.

- **Extensibility:** Usage API for enterprise integration, customizable chat behaviors, and the ability to connect to external code repositories and Jira.[1,11,32].

- **Security/Privacy:** Offers fully air-gapped on-premise deployments (e.g., Tabnine + Dell AI Factory), SOC 2 compliance, and a policy of not training its proprietary models on customer code [1,10].

3.5. Amazon Q Developer (AWS)
3.5.1. Overview: Amazon Q Developer is the evolution of CodeWhisperer, providing comprehensive AI assis-tance deeply integrated across the AWS ecosystem [1].

3.5.2. Agentic Capabilities:
- **Core:** Provides interactive agentic coding in IDEs (file R/W, bash commands) and an agent in the AWS Management Console for resource analysis, troubleshooting, and architectural guidance [1,5,33].
- **Background/Remote Agent:** Yes, Amazon Q Developer agents can "autonomously perform a range of tasks–everything from implementing features, documenting, testing, reviewing, and refactor-ing code, to performing software upgrades" [5]. Specialized agents are available for complex, long- running tasks such as Java application upgrades (e.g., Java 8 to 17) and. NET porting from Windows to Linux [1,5]. These capabilities imply significant background processing and autonomy.

3.5.3. Context & LLMs: Context up to 100k characters in IDE chat, with conversation persistence [1]. The IDE agent uses Claude Sonnet 3.7 [1]. Supports images as input [1].

- **Model Context Protocol (MCP) Support:** Yes. Amazon Q Developer, particularly its CLI, supports MCP, enabling it to interact with external tools and services [1,34,35]. This allows users to extend Q's capabilities by connecting it to custom tools and services, supporting the 'stdio' transport layer for MCP servers [35]. MCP integration allows Q Developer to orchestrate tasks across native and MCP server-based tools for more customized responses [35]. The MCP support includes tools, prompts, and resources [34].

- **Extensibility:** MCP sup- port for chat and CLI, integrations with GitLab and GitHub (preview) [1,34,35].

- **Security/Privacy:** Configurable data sharing opt-out, IAM controls, and adherence to AWS compliance standards. For Amazon Q Developer Pro, proprietary content is not used for service improvement [1,5].

3.6. WindSurf (formerly Codeium, acquired by OpenAI)
3.6.1. Overview: Formerly Codeium, rebranded to WindSurf, then acquired by OpenAI in May 2025. It offered IDE plugins and a standalone "agentic IDE" [1].

3.6.2. Agentic Capabilities (Pre-Acquisition):
- **Core:** The "Cascade" agentic AI was central, designed to understand intent and handle complex codebases [36]. It featured different modes for interaction.
- **Background/Remote Agent:** "Cascade Write Mode" functioned similarly to AutoGPT, capable of creating multiple files, running scripts, testing them, and debugging them with a high degree of automation (around 90)

3.6.3. Context & LLMs (Pre-Acquisition): Context was drawn from recent files, terminal, Cascade con- versations, web pages, and local code indexes [1,12]. It used models like GPT-4o, Claude 3.5 Sonnet, and proprietary models, with Claude 3.5 recommended for code generation.[1,12] Featured "Memories" for persisting context and user-defined "Rules"[12,36].

- **Model Context Protocol (MCP) Support (Pre-Acquisition):** Yes. WindSurf integrated MCP to enhance AI workflows by connecting custom tools and services, with curated MCP servers available for one-click setup [1,36-38]. MCP allowed WindSurf to connect to external data sources and simplify tool integration [37].

- **Post-Acquisition Sta- tus:** The future of WindSurf as a standalone product is uncertain. Its technology is likely to be integrated into OpenAI’s offerings. Features like on-premise options and broad plugin support, which were part of Codeium/WindSurf pre-acquisition [1], are doubtful under OpenAI’s typical cloud-centric model.

- **Ex-tensibility (Pre-Acquisition):** MCP support, AI Rules [36].

- **Security/Privacy (Pre-Acquisition):** Self-hosting options were available for Codeium [Codeium].

3.7. Augment Code

3.7.1. Overview: A newer entrant focusing on deep codebase understanding via its ”Context Engine” and ”Memories,” and offering powerful remote agent capabilities [1].

3.7.2. Agentic Capabilities:

- **Core:** The “Augment Agent” (IDE-bound) with ”Next Edit” for guided, multi-step changes. Sup- ports terminal execution, MCP tool use, and multi-modal input (images, Figma)[1].
- **Background/Remote Agent:** ”Remote Agents” are a key feature [1,3,4]. These are purpose-built, independent, containerized cloud workers that can continue coding tasks (e.g., implementing fea-tures, upgrading dependencies, writing PRs) after the developer logs off, delivering ready-to-merge PRs [3,4]. Multiple agents can run concurrently on independent tasks, even on the same repository, each on its own branch [4]. Users monitor and manage these agents from VS Code, receiving notifi-cations [4]. They can be connected to via SSH for direct file manipulation or command execution [4].

3.7.3. Context & LLMs: Features a ”Context Engine” (up to 200K tokens) with real-time indexing and ”Memories” for personalization and persisting context across sessions [1]. Uses Claude Sonnet 4 [1]. Its semantic index retrieves relevant code snippets quickly, maintaining context across complex, multi-repo projects for Remote Agents [3].

- **Model Context Protocol (MCP) Support:** Yes. Augment Agent can utilize external integrations through MCP to access external systems and tools.[1,39,40] MCP servers can be configured in Augment via its settings panel or directly in ‘settings.json‘ for both VS Code and JetBrains IDEs [39,40]. This allows connection to databases, browser automation tools, messaging services, etc.[39,40].

- **Extensibility:** MCP support for external tools (Jira, Notion, web search) [1].

- **Security/Privacy:** First AI coding tool with ISO/IEC 42001:2023 (AI Management Systems), SOC 2 Type II certified, and no training on customer code [1]. Augment emphasizes enterprise-grade privacy for its Remote Agents with a non-extractable architecture [3]. No on-premise solution is mentioned [1].

3.8. Cline (Open-Source)

3.8.1. Overview: Cline is an open-source AI coding assistant specifically designed for agentic behavior, fea- turing a “Plan/Act” paradigm and strong extensibility through MCP [1,41].

3.8.2. Agentic Capabilities:

- **Core:** Implements a “Plan/Act” loop where it formulates a plan based on user requests, presents it for approval, and then executes the approved steps [1]. This allows for controlled agentic behavior. It can create/edit files (with diff views), run code/tests, execute terminal commands, and interact with web browsers [41].
- **Background/Remote Agent:** Cline does not offer a managed cloud-based ”remote agent” service like Augment or Cursor. However, its fundamental architecture as an open-source, self-hostable tool [1,42-44] allows users to configure and run it in their own server environments. This means Cline tasks can be executed in the background or on remote machines under the user’s control. The ”Proceed While Running” feature for long-running terminal commands (e.g., development servers) is a direct example of handling asynchronous operations [41]. Its ability to execute sequences of actions (plan/act) combined with terminal and MCP tool use enables complex, potentially long- running automated workflows that are effectively background tasks managed by the user’s setup.

3.8.3. Context & LLMs: Dynamically fetches context by opening/searching files and using MCP tools [1]. Supports a variety of LLMs via API providers (OpenRouter, Anthropic, OpenAI, Google Gemini, AWS Bedrock, Azure, GCP Vertex) and local models through platforms like LM Studio and Ollama [1,41,42]. Tracks token usage and API costs [41].

- **Model Context Protocol (MCP) Support:** Yes, robust and central to its design. Cline leverages MCP extensively to create and integrate custom tools tailored to user workflows [1,41,45,46]. Users can request Cline to help build an MCP server, and it can handle setup and installation [41,46]. It can connect to diverse tools for web interaction (Hyperbrowser, Firecrawl, Perplexity), frontend development (21st.dev Magic UI), content integration (Markdownify), and custom solutions like Cognee for knowledge graph interaction [45,46,47].

3.8.4. Extensibility: Cline as an Exemplar Open-Source Agentic Tool Cline’s open-source nature (Apache 2.0 license [19,41])

provides several distinct advantages:

- **Customizability & Extensibility:** Users can inspect, modify, fork, and tailor Cline to their specific project needs and preferences [41,48]. The deep integration with custom tools via MCP is a prime example of this, allowing developers to extend Cline’s capabilities significantly [41,46,49].
- **Transparency & Trust:** The availability of source code allows for full scrutiny of the tool’s op- erations, data handling, and

security mechanisms. This can foster greater trust compared to closed-source alternatives.

- **Community-Driven Development:** Being open source opens the door for community contributions, potentially leading to a richer ecosystem of shared tools, custom agents, and faster evolution driven by diverse user needs [41,48]. The Cline Community MCP Server for issue reporting is an example of community-oriented tooling [50].
- **Data Privacy & Control (Self-Hosting):** A major advantage is the ability to self-host Cline. Users can run Cline entirely within their own infrastructure, especially when combined with locally hosted LLMs (via Ollama, LM Studio [41,42]) and self-hosted MCP servers. This ensures that code and other sensitive data do not leave the user's-controlled environment, addressing critical data privacy concerns [1,43,44]. This aligns with general benefits of self-hosting such as full data control, ability to meet specific security compliance requirements, and avoidance of vendor lock-in [44].
- **Cost-Effectiveness:** Self-hosting and the option to use local LLMs can potentially lead to lower operational costs compared to subscription-based services that charge for API calls or per-user licenses [41,43]. However, performance of local models can vary and may not match leading pro- prietary models for all tasks [42].

Cline's ”Plan/Act” paradigm, where it formulates a plan, presents it to the user for approval (or auto- approves based on settings), and then executes it, embodies a transparent and controllable approach to agentic behavior [1]. This, combined with its robust MCP implementation for diverse tool use (web tasks, file system operations, terminal command execution, custom tools [41,45,46]), makes it a powerful and flexible open-source option in the agentic IDE tool landscape.

Security/Privacy: As an open- source, self-hosted tool, security and data privacy is primarily determined by the user's deployment environment, configuration, and the LLMs/MCP servers they choose to connect. Data control is entirely with the user when self-hosted [1].

4. Comparative Analysis of Agentic IDE Tool Features

The featuresets of the analyzed agentic IDE tools reveal a diverse and rapidly evolving landscape. This section provides a comparative overview, focusing on background/remote agent capabilities, MCP sup- port, and the role of open-source solutions.

4.1. Feature Comparison Table

The following table summarizes the key features of the prominent agentic IDE tools discussed, with a particular emphasis on aspects relevant to this updated analysis.

Tool	Primary Agentic Paradigm(s)	Background/Remote Agent Capabilities	Model Context Protocol (MCP) Support	Open Source	Security/Privacy Highlights
GitHub Copilot	IDE-Integrated Assistant; Background Task Executor	Yes: ”Copilot coding agent” (autonomous PRs via GitHub Actions). Distinct from IDE agent mode.	Yes: Client for tools from local MCP servers. Extends context.	No	SOC 2 (Business), No training on Business/Enterprise data.
Cursor	AI-First Editor; IDE-Integrated Agent; Remote Task Executor	Yes: ”Background Agents (Beta)” (asynchronous remote code editing/running in cloud VMs).	Yes: Extensive client support (plugin system), one-click install for curated servers.	No (VS Code fork)	SOC 2 Type II, Privacy Mode (zero data retention). Background agents require privacy mode off.
Sourcegraph Cody	IDE-Integrated Assistant; Context-Gathering Agent	Limited: ”Agentic chat” gathers context autonomously. No dedicated remote execution environment.	Yes: Via OpenCtx layer.	Yes (Apache 2.0)	On-prem options, SOC 2 Type II, ISO 27001, No training on customer code.
Tabnine	IDE-Integrated Assistant; Specialized Task Agents	Yes: Specialized agents (e.g., Jira to Code, Code Review) for autonomous task execution.	Mentioned in general MCP docs, but no specific implementation details found. Focus on internal ”Enterprise Context Engine”.	No	Fully air-gapped on-prem, SOC 2, No training on customer code.
Amazon Q Developer	Ecosystem-Integrated Assistant; Specialized Task Agents; Background Task Executor	Yes: Autonomous agents for features, docs, tests, refactoring; specialized agents for Java upgrades, .NET porting.	Yes: CLI supports MCP for custom tools/services (stdio transport).	No	Configurable data sharing opt-out, IAM controls, AWS compliance. Pro tier: no content used for service improvement.
WindSurf (Pre-Acquisition)	Agentic IDE; Background Task Executor	Yes: ”Cascade Write Mode” (AutoGPT-like multi-file creation, run, test, debug).	Yes: Integrated MCP for custom tools and services.	No (Formerly had self-host option)	Pre-acquisition: Codeium offered self-hosting. Current status under OpenAI unclear.

Augment Code	IDE-Integrated Agent; Remote Task Executor	Yes: "Remote Agents" (purpose-built cloud workers, continue after logoff, deliver PRs).	Yes: Client for external tools via MCP, configurable in IDE settings.	No	ISO/IEC 42001, SOC 2 Type II, No training on customer code, Non-extractable architecture.
Cline	Open-Source Agentic Assistant; User-Deployed Background Task Executor	Yes (User-Deployed): Self-hostable architecture allows background/remote execution. "Proceed While Running" for async tasks. Plan/Act for complex workflows.	Yes: Robust client for MCP; enables custom tool building and integration with diverse services.	Yes (Apache 2.0)	User-controlled via self-hosting and configuration. Full data privacy when self-hosted.

Table 1: Comparative Featureset of Prominent Agentic IDE Tools (Mid-2025)

4.2. Discussion of Background/Remote Agent Capabilities

The analysis reveals that background and remote agent capabilities are a significant and growing trend, though their implementation and maturity vary considerably.

- **Prevalence and Nature:** Tools like GitHub Copilot (coding agent), Cursor (Background Agents), Augment Code (Remote Agents), and Amazon Q Developer (specialized transformation/task agents) offer distinct remote or background processing features [2,3,5,9]. These range from platform-integrated solutions like Copilot's use of GitHub Actions , to dedicated cloud environments provided by the vendor (Cursor, Augment Code), to specialized, high-autonomy agents for complex tasks like code modernization (Amazon Q Developer) [2,3,5,9]. Tabnine's specialized agents (e.g., Jira to Code) also point to autonomous background processing for specific enterprise work- flows [10]. Cline, through its open-source and self-hostable nature, empowers users to set up their own background/remote execution environments [41,44].
- **Task Scope and Autonomy:** The tasks offloaded to these agents are becoming increasingly complex, moving beyond simple code generation to encompass bug fixing, feature implementation from issues, automated pull request generation, large-scale refactoring, and application upgrades [2,3,5,10]. The level of autonomy varies, with some agents requiring explicit approval steps (like Cline's Plan/Act) while others aim for more end-to-end task completion with minimal intervention before delivering a PR [1,2,3].
- **Impact on Workflow:** These capabilities fundamentally alter the developer workflow by of- floading time-consuming or complex tasks that can run asynchronously. This allows developers to parallelize their work, focus on higher-level design and review, and potentially accelerate develop- ment cycles significantly. The ability for agents to continue working after an IDE is closed, as seen with Augment Code's Remote Agents [3,4], represents a substantial shift in how AI can contribute to projects.

The emergence of these features indicates a clear direction towards AI assistants that are not just interactive helpers but also persistent, autonomous contributors to the software development lifecycle.

4.3. Discussion of MCP Adoption and Significance

The Model Context Protocol (MCP) is emerging as a critical standard for enhancing the extensibility and contextual awareness of agentic IDE tools.

- **Adoption Extent:** A significant number of the analyzed tools now claim MCP support, including GitHub Copilot, Cursor, Sourcegraph Cody (via OpenCtx), Amazon Q Developer (CLI), Augment Code, Cline, and WindSurf (pre-acquisition) [1,6,14,17,34,36,39,41]. This widespread adoption underscores its perceived value in the ecosystem.
- **Realized Benefits:** MCP enables these tools to act as clients for a diverse range of external tools and data sources [6-8]. This can include accessing project management systems (e.g., Jira via custom MCP servers for Cline), querying databases, interacting with web services, or leveraging specialized data stores like Pieces LTM for GitHub Copilot [15,41]. By standardizing how context and tools are provided to LLMs, MCP allows for richer, more relevant interactions and more capable agentic actions.
- **Implementation Variations:** While many tools support consuming tools via MCP, some, like Cline, also emphasize the ease of building and integrating custom MCP servers, empowering users to tailor the AI's capabilities extensively [41,46]. The supported transport types (e.g., 'stdio' for Amazon Q CLI , HTTP+SSE for Cursor) can also influence the types of MCP servers that can be readily integrated. Sourcegraph Cody's approach of using OpenCtx as an MCP-compatible layer is another distinct implementation strategy [6,17,35].

MCP's role is particularly vital for background/remote agents, which inherently need to interact with a broader set of systems beyond the local IDE. The standardization offered by MCP facilitates these interactions, making the agents more powerful and adaptable. The continued development and adoption of MCP are likely to be key drivers of innovation in agentic AI for software development.

4.4. The Role and Advantages of Open-Source Solutions (Cline as Case Study)

Cline stands out as an open-source agentic IDE tool, and its characteristics highlight the unique advan- tages this model can offer in a rapidly evolving market.

- **Customizability and Extensibility:** Open-source tools like Cline provide users with the ulti- mate level of control to modify, fork, and adapt the software to their precise requirements [41,48]. Cline's design, particularly its robust MCP implementation, allows developers to integrate a vast array of custom tools and data sources, effectively building a personalized agentic environment [41, 46].

- **Data Privacy and Control:** The ability to self-host is a cornerstone advantage of open-source solutions in the context of AI [1]. With Cline, users can deploy the entire system, including any connected local LLMs (e.g., via Ollama) and MCP servers, within their own infrastruc-ture [41-44].This ensures that sensitive code, proprietary data, and interaction logs remain under the user's exclusive control, addressing critical privacy and security concerns that might arise with cloud-based, proprietary services.
- **Transparency and Trust:** The availability of the source code allows for complete transparency into how the tool operates, how it handles data, and its security posture. This can foster a higher level of trust, as users and organizations can independently audit and verify the software's behavior.
- **Community and Innovation:** Open-source projects often benefit from a vibrant community that contributes to development, shares custom tools and configurations, and drives innovation [41,48].The Cline Community MCP server for GitHub issue reporting is an early example of such community-driven enhancement [50].
- **Cost-Effectiveness and Flexibility:** While not without operational overhead if self-hosted, open-source tools can offer a more cost-effective solution, especially when leveraging local LLMs or existing infrastructure [42,43]. They also provide freedom from vendor lock-in, allowing users to switch components (like LLM providers) more easily.

Cline's "Plan/Act" paradigm, which involves user approval before execution , further enhances its appeal for users seeking a balance between automation and control [1]. As agentic AI tools become more powerful and autonomous, the transparency, control, and customizability offered by open-source solu-tions like Cline will likely become increasingly important differentiators for a segment of the developer community and for organizations with stringent data governance requirements.

5. Discussion (Broader Trends and Implications)

The comparative featureset analysis reveals a vibrant and rapidly evolving landscape for agentic IDE tools, characterized by several key trends and implications for the future of software development.

A clear trend is towards increasing autonomy and task complexity, largely propelled by the mat-uration of background and remote agent capabilities. Tools are moving beyond simple suggestions and interactive assistance to undertake more complex, multi-step tasks autonomously or semi-autonomously, such as generating entire features from issue descriptions or performing large-scale code transforma-tions [2,3,5]. This signifies a shift where AI is not just an assistant but an active, often asynchronous, collaborator.

Deep contextual understanding remains paramount, but the mechanisms for achieving it are expanding. While sophisticated RAG systems, large context windows, and personalized learning (e.g., Augment Code's "Memories," Tabnine's "Enterprise Context Engine") are becoming standard [1], the Model Context Protocol (MCP) is emerging as a key enabler for agents to access and integrate a much broader array of external context from diverse tools and data sources [6,8].This allows agents to operate with a more holistic understanding of the project and its ecosystem.

Consequently, tool use and extensibility, primarily facilitated by MCP, are becoming defining characteristics of advanced agentic systems. The ability for an AI agent to seamlessly invoke external APIs, query databases, or interact with project management software significantly broadens its potential impact beyond built-in functionalities. The growing adoption of MCP across multiple tools points to its crucial role in fostering interoperability and preventing vendor lock-in within tool ecosystems [6,7].

The availability of LLM diversity and choice continues to be a notable feature, with many plat- forms offering users the ability to select from various leading models [1]. This allows tailoring the AI's performance, cost, and specific task suitability. The support for local LLMs, as seen with Cline , further enhances this flexibility, particularly for users with privacy concerns or specific computational resources [41].

For enterprise adoption, security, privacy, compliance (evidenced by SOC 2, ISO certifications), and flexible deployment options (on-premise, VPC) remain critical considerations [1]. Tools like Tabnine, Sourcegraph Cody, and Amazon Q Developer strongly cater to these needs [1,10,20]. Simultaneously, open-source solutions like Cline offer an alternative pathway to achieving data security and privacy through self-hosting and full user control [44].

Market consolidation, exemplified by the acquisition of WindSurf (formerly Codeium) by OpenAI , signals the intense competition and high strategic value placed on agentic AI capabilities [1]. Such moves can accelerate technological integration but may also impact the diversity of independent solutions and innovation pathways.

The evolving capabilities of these tools, especially the rise of more autonomous background agents, are reshaping the human-developer role. As Kent Beck commented regarding Augment Code's Remote Agents, developers may increasingly focus on breaking down work into clearly scoped tasks for AI, thereby leveling up their thinking to become orchestrators and reviewers of AI-driven development [3]. This suggests a future where developers delegate more implementation details to AI agents, concentrating on architectural decisions, complex problem-solving, and the critical validation of AI-generated work.

Despite these advancements, human oversight remains crucial. Most agentic tools are still designed as "copilots" or "pair programmers," augmenting rather than replacing human developers, particularly for novel, mission-critical, or highly complex tasks [1]. The effectiveness of these tools often hinges on the quality of the existing codebase, the clarity of prompts and task definitions, and the developer's skill in guiding, validating, and integrating AI-generated outputs.

6. Conclusion

Agentic IDE tools represent a significant leap forward in AI-assisted software development. As of mid- 2025, the market offers a diverse range of solutions, each with a unique featureset and approach to enabling AI agents as active partners in the coding process. This comparative featureset analysis high- lights the substantial advancements in core agentic functionalities and, notably, the emergence and growing sophistication of background and remote agent capabilities, alongside the increasing adoption of the Model Context Protocol (MCP) for enhanced extensibility.

From the broad ecosystem integration and evolving autonomous features of GitHub Copilot and Amazon Q Developer, to the AI-native IDE experience and remote agent offerings of Cursor and Augment Code, the enterprise-focused control and specialized agents of Tabnine and Sourcegraph Cody, and the unique flexibility and user-control paradigm of the open-source tool Cline, developers and organizations have an expanding array of powerful features at their disposal.

The development of background and remote agents signifies a move towards greater automation of more complex and potentially longer-running software development tasks, fundamentally altering developer workflows. Concurrently, the rise of MCP as a standard for tool and context integration is paving the way for more interconnected and capable AI ecosystems. Cline, as an open-source exemplar, demonstrates the potential for community-driven innovation, transparency, and user-centric control over data and toolchains, offering a compelling alternative for those prioritizing these aspects.

The trajectory points towards even more sophisticated agents capable of handling increasingly com- plex aspects of the software development lifecycle with greater autonomy. Future research and develop- ment will likely focus on enhancing agents' reasoning, planning, and learning abilities, improving their reliability and verifiability, fostering more seamless and intuitive human-AI collaboration models, and addressing the ethical and security challenges associated with highly autonomous AI systems in software engineering. The continued evolution of LLMs and protocols like MCP will undoubtedly fuel further innovation in this exciting and rapidly transforming domain.

References

1. Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J. (2023). A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6), 186345.
2. Barke, S., et al. (2023). Grounded code generation with retrieved libraries. *arXiv preprint arXiv:2310.09129.*
3. Ziegler, D. M., et al. (2024). Improving mathematical reasoning with tool-augmented language models. *Transactions on Machine Learning Research.*
4. Park, J. S., et al. (2023). Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442.*
5. Shinn, N., et al. (2023). Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366.*
6. Yao, S., et al. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.
7. Mialon, G., et al. (2023). Augmented language models: a survey. Transactions on Machine Learning Research.
8. Huang, W., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.
9. Xi, Y., et al. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.
10. Callaghan, S., et al. (2024). Software Development with AI: The Present and the Future. ACM *Queue*.
11. GPT-Engineer. (2023). GPT-Engineer GitHub Repository.
12. Aider. (2023). Aider GitHub Repository AI pair programming in your terminal.
13. Devin, I. (2024). the first AI software engineer.
14. Model Context Protocol Steering Committee. (2024-2025). Model Context Protocol Specification.
15. Cline. (2024-2025). Cline MCP Documentation. (As per Cline's specific documentation).
16. Cursor AI. (2024-2025f). Cursor MCP Documentation. (As per Cursor's specific documentation).
17. GitHub. (2024-2025g). GitHub Copilot MCP Documentation. (As per GitHub's specific documentation).
18. Sourcegraph. (2024-2025f). Sourcegraph Cody MCP Documentation. (As per Sourcegraph's specific documentation).
19. AWS. (2025c). Amazon Q Developer MCP Documentation. (As per AWS's specific documentation).
20. WindSurf/Codeium. (2024-2025c). WindSurf MCP Documentation (pre-acquisition). (As per Wind-Surf/ Codeium's specific documentation).
21. GitHub. (2024-2025a). Relevant blog post on GitHub Copilot coding agent. GitHub Blog.
22. GitHub. (2024-2025b). GitHub Copilot coding agent documentation. GitHub Docs.
23. GitHub. (2025a). Announcement on open-sourcing Copilot Chat in VS Code. GitHub Blog.
24. GitHub. (2025b). Details on GitHub Copilot premium requests and billing. GitHub Docs.
25. GitHub. (2024-2025c). Documentation on MCP support in GitHub Copilot. GitHub Docs.
26. GitHub. (2024-2025d). Information on LLM choices for GitHub Copilot. GitHub Docs.
27. GitHub. (2024-2025e). GitHub Copilot Trust Center. GitHub.
28. GitHub. (2024-2025f). Information on GitHub Copilot SOC 2 compliance. GitHub Trust Center.
29. Cursor AI. (2024-2025a). Cursor Agent Mode documentation/ blog. Cursor Blog/Docs.
30. Cursor AI. (2025a). Cursor Background Agent feature announcement. Cursor Blog/Docs.
31. Cursor AI. (2024-2025b). Details on Cursor's ”Tab” model. Cursor Blog/Docs.
32. Cursor AI. (2024-2025c). Cursor MCP integration details. Cursor Blog/Docs.

33. Cursor AI. (2024-2025d). Supported LLMs in Cursor. Cursor Docs.
34. Cursor AI. (2025b). Cursor SOC 2 Type II certification announcement. Cursor Blog.
35. Cursor AI. (2024-2025e). Cursor privacy policy and data handling. Cursor Docs.
36. Sourcegraph. (2024-2025a). Cody Agentic Chat feature documentation. Sourcegraph Docs.
37. Sourcegraph. (2024-2025b). Cody Auto-Edit feature details. Sourcegraph Blog/Docs.
38. Sourcegraph. (2024-2025c). Explanation of Cody's RAG system. Sourcegraph Blog.
39. Sourcegraph. (2024-2025d). Cody integration with MCP and OpenCtx. Sourcegraph Docs.
40. Sourcegraph. (2024-2025e). LLM options for Sourcegraph Cody. Sourcegraph Docs.
41. Sourcegraph. (2025). Sourcegraph ISO 27001 certification. Sourcegraph Trust Center.
42. Tabnine. (2024-2025a). Tabnine Enterprise Context Engine. Tabnine Docs/Website.
43. Tabnine. (2024-2025b). Tabnine Code Review Agent. Tabnine Docs.
44. Tabnine. (2024-2025c). Tabnine Jira to Code Agent. Tabnine Docs.
45. Tabnine. (2024-2025d). Supported LLMs in Tabnine. Tabnine Docs.
46. Tabnine. (2024-2025e). Tabnine air-gapped deployment options. Tabnine Website.
47. Tabnine. (2025). Tabnine and Dell AI Factory partnership. Tabnine Blog.
48. AWS. (2024-2025a). Amazon Q Developer agentic coding in IDE. AWS Documentation.
49. AWS. (2024-2025b). Amazon Q Developer in AWS Management Console. AWS Documentation.
50. AWS. (2024-2025c). Amazon Q Developer for Java application upgrades. AWS Blogs.
51. AWS. (2025a). Amazon Q Developer MCP support. AWS Documentation.
52. AWS. (2025b). LLM powering Amazon Q Developer IDE agent. AWS Blogs/Docs.
53. AWS. (2024-2025d). Security and privacy in Amazon Q Developer. AWS Documentation.
54. Bloomberg News. (2025). OpenAI acquires WindSurf (Codeium). Bloomberg.
55. Reuters. (2025). OpenAI's acquisition of WindSurf. Reuters.
56. WindSurf/Codeium. (2024-2025a). Details on Cascade agent (pre-acquisition). WindSurf/Codeium Blog/Docs.
57. WindSurf/Codeium. (2024-2025b). MCP support in WindSurf (pre-acquisition). WindSurf/Codeium Docs.
58. Codeium. (2024). Codeium self-hosting options (pre-acquisition). Codeium Docs.
59. Augment Code. (2024-2025a). Augment Agent documentation. Augment Code Docs.
60. Augment Code. (2024-2025b). Next Edit feature in Augment Code. Augment Code Docs.
61. Augment Code. (2025). Augment Code ”Memories” feature. Augment Code Blog/Docs.
62. Augment Code. (2024-2025c). MCP support in Augment Code. Augment Code Docs.
63. Augment Code. (2025b). Augment Code ISO/IEC 42001 certification. Augment Code Blog.
64. Augment Code. (2024-2025d). Augment Code security and privacy policy. Augment Code Website.
65. Cline Contributors. (2024-2025). Cline GitHub Repository. GitHub.
66. Cline Documentation. (2024-2025a). Cline Plan/Act paradigm. Cline GitHub Wiki/Docs.
67. Cline Documentation. (2024-2025b). Cline MCP implementation details. Cline GitHub Wiki/Docs.
68. Cline Documentation. (2024-2025c). Cline support for local LLMs via Ollama. Cline GitHubWiki/Docs.
69. GitHub Docs. (2025). About assigning tasks to Copilot.
70. GitHub Docs. (n.d.). About Copilot agents.
71. Cursor Docs. (n.d.). Background Agents (Beta).
72. Cursor Docs. (n.d.). Agent Mode.
73. Sourcegraph. (n.d.). Cody — AI coding assistant from Sourcegraph.
74. Tabnine Docs. (n.d.). Tabnine's Personalization in Depth.
75. Amazon Web Services. (n.d.). Amazon Q Developer.
76. AWS Documentation. (n.d.). What is Amazon Q Developer?
77. DataCamp. (n.d.). Windsurf AI: Agentic Code Editor Tutorial.
78. Windsurf. (n.d.). Introducing the Windsurf Editor.
79. Augment Code Docs. (n.d.). Using Remote Agent.
80. Augment Code Docs. (n.d.). Using Remote Agent.
81. Apidog. (2025, January 6). What is Cline? The Open Source AI Coding Assistant.
82. HD Robots. (n.d.). Cline AI.
83. Reddit. (2024). Self hosted models to use with Cline?
84. Cline Blog. (n.d.). Supercharge Your Cline Workflow: 7 Essential MCP Servers.
85. GitHub Docs. (n.d.). Extending Copilot coding agent with the Model Context Protocol (MCP).
86. Pieces for Developers Docs. (n.d.). Pieces MCP + GitHub Copilot.
87. Cursor Docs. (n.d.). Model Context Protocol.
88. Cursor Docs. (n.d.). Working with Context.
89. Descope. (n.d.). What Is the Model Context Protocol (MCP) and How It Works.
90. WandB. (n.d.). The Model Context Protocol (MCP) by Anthropic: Origins, functionality, and impact.
91. Telerik Blogs. (n.d.). The Promise of the Model Context Protocol (MCP).
92. Tabnine Docs. (n.d.). Personalization.
93. AWS Documentation. (n.d.). Using MCP with Amazon Q Developer on the command line.
94. AWS What's New. (2025, April 29). Amazon Q Developer CLI now supports Model Context Protocol (MCP).
95. Arsturn Blog. (n.d.). Exploring the Model Context Protocol (MCP) in Windsurf.
96. GitHub Discussions - modelcontextprotocol. (n.d.). Windsurf MCP Integration.
97. Augment Code Docs. (n.d.). Setup Model Context Protocol

servers (VS Code).
98. Augment Code Docs. (n.d.). Setup Model Context Protocol servers (JetBrains).
99. Cline Docs. (n.d.). MCP Overview.
100. Underleaf AI Blog. (2024). How to Cite arXiv Papers: A Complete Guide.
101. Tabnine. (n.d.). Tabnine AI Code Assistant.
102. Tabnine. (n.d.). Pricing.
103. Sourcegraph Docs. (n.d.). Cody.
104. OpenCtx Docs. (n.d.). Cody support in OpenCtx.
105. GitHub - cline/cline-community. (n.d.). MCP for reporting issues and traces to Cline, with Cline.
106. APIPie Docs. (n.d.). Cline Configuration Guide.
107. Cognee Docs. (n.d.). Cline Integration.
108. XDA Developers. (n.d.). Benefits I never expected when I self-hosted Immich.
109. OpenProject Blog. (n.d.). Why self-hosting software?
110. arXiv Help. (n.d.). BibTeX and Eprints.
111. Reddit - r/ClaudeAI. (n.d.). Some issues with Sourcegraph Cody.
112. Sourcegraph Blog. (n.d.). How Cody understands your codebase.
113. Sourcegraph Community. (2025). Long-term memory.
114. Sourcegraph Blog. (n.d.). Cody: Better, Faster, Stronger.
115. Tabnine Docs. (n.d.). Welcome to Tabnine.
116. Zack Proser Comparisons. (n.d.). Manus vs Tabnine.
117. Tabnine Docs. (n.d.). Usage API.
118. Tabnine Blog. (n.d.). Documenting code with the Tabnine Documentation Agent.
119. Software.com AI Index. (n.d.). Cody.
120. Sourcegraph Blog. (n.d.). Lessons from building AI coding assistants: Context retrieval and evaluation.
121. Tabnine Docs. (n.d.). IdP Sync.