ChatGPT

AGENTS.md & Agent-Native Development (2025)

Introduction

Artificial-intelligence coding agents, such as OpenAI **Codex**, Google **Gemini**, **Sourcegraph Amp**, **Factory** and **Cursor**, have moved beyond autocomplete to perform multi-step tasks—reading repositories, running tests, applying style rules and preparing pull requests. Each tool originally shipped with its own configuration file (`CLAUDE.md`, `.cursor/rules`, `GEMINI.md`, etc.), which fragmented instructions across a project and created maintenance headaches. During summer 2025, major vendors—including OpenAI, Google and Sourcegraph—agreed on **AGENTS.md**, a simple Markdown file that lives alongside your repository and serves as a universal instruction manual for AI coding agents [1].

The specification allows teams to express build steps, test commands, code style rules, PR workflows, security notes and other conventions in a single, machine-readable document. AI assistants read the nearest `AGENTS.md` file (root-level or nested) and follow its instructions; deeper files override root-level defaults [2]. Because the format is plain Markdown, human contributors can also read and review these instructions. This report summarizes how research labs, open-source projects and individual developers are using `AGENTS.md` in 2025, provides examples of prompts and commands for popular agents such as Codex, Jules and Cursor, and highlights best-practice repository structures.

Why AGENTS.md matters

Traditional `README.md` files are written for human readers; they often omit the gritty details AI agents need, such as exact commands or folder-specific conventions. The **Everyday AI** article explains that prior to `AGENTS.md` each tool required a different rule file, leading to duplication and confusion; by unifying the format, `AGENTS.md` removes the need to copy instructions between `claude.md`, `.cursorrules`, `gemini.md` and other tool-specific files [3]. The article notes that OpenAI, Google and Cursor collaborated to create one file that works with nearly every AI coding tool [3].

The **Factory AI** news post further emphasises the benefits: `AGENTS.md` captures operational details (dependency installation, build and test commands, style rules, validation steps, security concerns) in a lightweight format and serves as a neutral standard across languages and frameworks [4]. Teams can add root-level and nested `AGENTS.md` files; tools look for the nearest file, allowing monorepos to have package-specific instructions [5]. The working group designing the standard aims to replace vendor-specific files and make it easy for agents to behave reliably across projects [6].

Adoption by AI labs and industry players

OpenAI Codex and the AGENTS.md ecosystem

OpenAI's **Codex** CLI uses `AGENTS.md` to control how the agent navigates your project. According to the Codex documentation, when the agent runs a task it reads `AGENTS.md` files in three places: a user-level file (`~/.codex/AGENTS.md`), the repository root, and the current working directory. These files are merged top-down, with more specific ones overriding earlier instructions [7]. The file describes build commands, test commands, linting rules and other checks; if programmatic checks are specified, the agent must run them and ensure they pass [2]. Codex CLI supports interactive sessions (`codex`) and one-shot commands (`codex "explain this codebase"`); full-auto mode (`--approval-mode full-auto`) allows the agent to run tasks without stopping for approval [8].

Google Jules

Google's **Jules** assistant automatically looks for an `AGENTS.md` file in the root of a repository; the docs say the file describes agents or tools within the codebase and helps Jules generate better plans and completions [9]. The **Jules Awesome List** on GitHub contains curated prompts categorized by everyday developer tasks (adding a test suite, generating mock data, scaffolding a new API), debugging, documentation, testing, package management, AI-native tasks (analyzing technical debt, clustering functions) and tasks that start from scratch (bootstrapping a Python or Express app) [10] [11]. These examples illustrate the types of tasks an agent can perform when guided by `AGENTS.md`.

Factory AI

Factory is part of the industry working group that standardised `AGENTS.md`. Its documentation describes the file as a briefing packet for AI coding agents, containing build and test instructions, architectural patterns, external services, environment variables and domain-specific vocabulary [12]. Factory emphasises that one `AGENTS.md` works across many agents—including Factory Droid, Cursor, Aider, Gemini CLI, Jules, Codex, Zed and Phoenix [13]. The file discovery hierarchy is `./AGENTS.md` (current directory), then parent directories up to the root, then any `AGENTS.md` in subfolders, with a personal override at `~/.config/AGENTS.md`; the closest file takes precedence [14].

Sourcegraph and AGENT.md

Sourcegraph proposed an alternative specification called **AGENT.md** (singular). The spec aims to unify multiple config files across tools and includes migration commands for renaming legacy files to `AGENT.md` [15]. The spec describes hierarchical support (global, root-level and subdirectory files), recommended content (project overview, commands, style guidelines, testing, security) and an example file structure [16] [17]. While AGENT.md competes with the plural `AGENTS.md`, many projects maintain both via symlinks for backward compatibility.

Cursor CLI

Cursor—a popular AI code editor—released a CLI that brings its programming agents directly into the terminal. An installation command (`curl https://cursor.com/install -fsS | bash`) installs the CLI [18], and the CLI is started with `cursor-agent` [19]. A blog post highlights that the CLI supports model-agnostic workflows (GPT-5, Gemini, Claude, etc.), real-time agent customization and automation, and integration with local, container and remote server environments [20]. The CLI can run headlessly and in parallel with IDEs, enabling flexible integration. A simple script example from the Cursor website uses `cursor-agent -p --force --output-format text` with a prompt to review recent code changes and write the output to `review.txt` [21]. Early reviews note that the CLI is still maturing and lacks some features compared with Claude Code [22], but adoption is growing.

Other tools and ecosystem

- **Octofriend**: The **synthetic-lab/octofriend** repository includes an `OCTO.md` file that instructs Octo's agent how to build and test a TypeScript project. The rules mention that the project may take up to 20 seconds to build via `npx tsc` and to use `type` aliases instead of interfaces [23]. The repository also states that Octo will search for `OCTO.md`, `CLAUDE.md` and `AGENTS.md` in that order and merge instructions across directories [24].
- **Kode (shareAI-lab)**: The Kode README notes that the tool supports the `AGENTS.md` protocol and can generate or update `AGENTS.md` automatically using structured documentation prompts [25]. It explains that prefixing prompts with `#` generates documentation that is appended to `AGENTS.md` [26].
- **Youtu-Agent**: Tencent's Youtu-Agent framework includes an *agents.md* file (lowercase) describing multi-agent paradigms. While it uses the same extension, it is not directly related to the AGENTS.md standard but illustrates how labs adopt similar naming conventions [27].

GitHub projects using AGENTS.md

Repository / Project	Use of AGENTS.md	Notable Features	Evidence
attogram/agents	Contains a root-level `AGENTS.md` that instructs all AI assistants to refer to `HUMANS.md` and to platform-specific agent files; emphasises that AI should assist with development while adhering to code standards [28].	The repository defines custom agent personas (e.g., `agents/jules.md`) with operational loops (explore, plan, execute, verify), file-modification protocol, link creation protocol and debug mode rules [29] [30]. It also provides `HUMANS.md` guidance for human collaborators [31] and `platforms/python.md` advice on virtual environments, code quality (black, flake8, pre-commit hooks) and testing [32].	Real example of custom agent instructions.

Repository / Project	Use of AGENTS.md	Notable Features	Evidence
openai/codex	The repository includes its own `AGENTS.md` and `codex-cli/README.md` that describe how Codex CLI uses the file. The CLI merges `AGENTS.md` from `~/.codex`, repo root and current directory, runs build and test commands, and passes programmatic checks [7] [2].	Quickstart commands: `npm install -g @openai/codex`; run `codex` interactively; pass a prompt (`codex \"explain this codebase\"`), or use full-auto mode with `--approval-mode full-auto` [8]. The CLI supports other providers via `--provider` and loads environment variables from `.env` [33].	Official docs for Codex CLI.
factory.ai examples	Factory's docs provide templates for `AGENTS.md` in Node + React monorepos. The example lists commands for installing dependencies (`pnpm install`), starting the dev server (`pnpm dev`), running tests (`pnpm test`), and building for production (`pnpm build`), along with code style (TypeScript strict mode, single quotes), testing frameworks (Vitest, Playwright), and PR instructions [34].	Shows how to structure `AGENTS.md` in modern monorepos and emphasises using additional `AGENTS.md` files in subpackages [5].	Practical template for developers.
synthetic-lab/ octofriend	Uses `OCTO.md` instead of `AGENTS.md`; instructs the agent to wait for long builds and prefer `type` aliases [23].	Highlights that some tools still use proprietary files but often merge with `AGENTS.md` [24].	Illustrates transitional state of ecosystem.
shareAI-lab/ Kode	Supports the `AGENTS.md` protocol and can generate documentation automatically. Using prompts prefixed with `#` appends instructions to `AGENTS.md` [26].	Integrates documentation generation into workflow; demonstrates how labs extend the standard.	Example of innovation on top of the specification.

Repository structure & typical AGENTS.md sections

Although the `AGENTS.md` format has no required schema, most files follow common patterns:

1. **Project overview and architecture**: A brief description of the tech stack, modules, and high-level design. In the Factory example, the overview notes that the project is a Next.js monorepo with front- and back-end apps [35].
2. **Build and command instructions**: Precise commands to install dependencies, build the project, start a dev server and run tests. For example, `pnpm install`, `pnpm dev`, `pnpm test`, `pnpm build` [36].
3. **Code style and conventions**: Rules for languages, variable naming (e.g., camelCase vs. snake_case), preferred patterns (functional components), linter configurations and formatting tools. The Factory template specifies TypeScript strict mode and single quotes [37]. The `attogram/agents` repository's `platforms/python.md` lists using `black` and `flake8`, pre-commit hooks, list comprehensions and pytest [32].
4. **Testing and validation**: Instructions to run unit tests (e.g., Jest, Vitest, pytest), end-to-end tests (Playwright), and lint checks. `AGENTS.md` may include programmatic checks that must pass before the agent can consider the task complete [38].
5. **Pull request guidelines**: Title formats (e.g., `[project] Title`), evidence required (screenshots, test output), and steps such as running `pnpm lint` and `pnpm test` before committing [39].
6. **Security and environment**: Notes about environment variables (use `.env.template`), handling secrets, network restrictions and data boundaries. Factory emphasises including security concerns and environment variables [4], while `attogram`'s instructions remind agents to ask for sensitive environment values rather than guessing [32].
7. **Additional sections**: Architecture diagrams, domain vocabulary, link creation protocols, file-modification protocols, and debug modes (as seen in the Jules persona file) [29] [30].

Prompt examples for AI coding agents

Everyday developer tasks (from Jules Awesome List)

- **Refactor or implement features:** "Add an Express.js API endpoint to `api/user.ts` that returns the profile for a given user ID" or "Refactor `UserService` to use async/await and add proper error handling."
- **Testing:** "Add a comprehensive Jest test suite for `src/validator.ts` covering edge cases" or "Set up Playwright tests for the `/login` page."
- **Documentation:** "Generate a README section explaining how to run the new analytics pipeline and update `AGENTS.md` accordingly."
- **Debugging:** "Investigate why the `checkout` function throws an exception when the cart is empty and fix the issue."
- **Infrastructure:** "Create a GitHub Actions workflow that runs `npm test` and `pnpm lint` on every pull request."

Codex CLI examples

- **Interactive session:** `codex` — launches a chat-driven interface where you can ask the agent to perform tasks like scaffolding a feature, refactoring code or writing tests.

- **One-shot execution:** `codex "explain this codebase to me"` — instructs Codex to summarise the repository's structure and architecture [40].
- **Full-auto mode:** `codex --approval-mode full-auto "create the fanciest todo-list app"` — Codex writes code, runs tests and commits changes without requiring intermediate approvals [41].
- **Provider selection:** `codex --provider gemini "add JWT authentication to the API"` — uses Gemini as the underlying model (if configured) [33].

Cursor CLI examples

- **Install and run:** `curl https://cursor.com/install -fsS | bash` followed by `cursor-agent` to start the agent [19].
- **Model selection:** In the interactive CLI, use commands like `/model gpt-5` or `/model sonnet-4.1` to switch models [42].
- **Scripted code review:** Use `cursor-agent -p --force --output-format text "Review the recent code changes and provide feedback on code quality and security"` to generate a text file with review comments [21].

Discussion and future considerations

The adoption of `AGENTS.md` has accelerated in 2025, with 20k+ projects already using the format [1]. Standardizing on a single file reduces cognitive load for developers and allows AI coding agents to behave predictably across tools. However, several open questions remain:

1. **Interoperability vs. fragmentation:** While `AGENTS.md` unifies instructions, some projects still use alternative files (`AGENT.md`, `CLAUDE.md`, `OCTO.md`). Maintaining symlinks or scripts to merge multiple rule files can be cumbersome [43]. Stakeholders should monitor whether the industry converges on `AGENTS.md` or continues to support multiple formats for backward compatibility.
2. **Security and guardrails:** As agents gain the ability to run arbitrary commands, the instructions in `AGENTS.md` must be audited carefully. Factory's docs recommend including security policies and validation steps [4]. Cursor's CLI article warns that users should consider security safeguards, because the tool can modify code and the software is still evolving [44]. Future standards may need to enforce stricter sandboxing or permission models.
3. **Testing reliability:** Programmatic checks defined in `AGENTS.md` give agents an objective measure of success, but they also increase run times and may require network access. Research labs could explore ways to optimize test suites for agentic workflows or to provide incremental evaluation.
4. **AI model selection:** Tools like Cursor and Codex now support multiple models (GPT-5, Claude, Gemini). `AGENTS.md` could include hints or constraints about which models perform best for different tasks, enabling agents to choose the optimal model dynamically. This area is still experimental.
5. **Future of documentation generation:** Kode's ability to auto-generate `AGENTS.md` sections from structured prompts [26] hints at a future where agents maintain their own instructions. Further research could investigate how to keep these auto-generated docs accurate and aligned with human intent.

Conclusion

`AGENTS.md` has emerged as the de facto standard for guiding AI coding agents. By centralizing build commands, testing protocols, style conventions, pull-request rules and security notes into a single file, teams can ensure that agents behave predictably and follow the same standards as human contributors. Adoption by major players such as OpenAI Codex, Google Jules, Factory AI and Cursor demonstrates the momentum behind this standard. Real-world projects like **attogram/agents**, **Kode** and **Octofriend** offer concrete examples of customizing agent behaviors, while articles from **Factory**, **Everyday AI** and **MLQ.ai** highlight industry collaboration and evolving tooling. As the ecosystem matures, stakeholders should continue refining `AGENTS.md` contents, strengthening security guardrails and exploring advanced features such as auto-generated documentation and model-specific instructions.

---

1 AGENTS.md: A new file is taking over GitHub - AI ML etc.
https://www.aimletc.com/agents-md/

2 38 Introducing Codex | OpenAI
https://openai.com/index/introducing-codex/

3 All about "AGENTS.md". AI Agents That Actually Get Your... | by Manpreet Singh | Everyday AI | Sep, 2025 | Medium
https://medium.com/everyday-ai/all-about-agents-md-972fde65d641

4 5 6 34 35 36 37 39 Factory joins AGENTS.md collaboration with OpenAI | Factory.ai
https://factory.ai/news/agents-md

7 raw.githubusercontent.com
https://raw.githubusercontent.com/openai/codex/main/docs/getting-started.md

8 33 40 41 raw.githubusercontent.com
https://raw.githubusercontent.com/openai/codex/main/codex-cli/README.md

9 Getting started | Jules
https://jules.google/docs

10 11 raw.githubusercontent.com
https://raw.githubusercontent.com/google-labs-code/jules-awesome-list/refs/heads/main/README.md

12 13 14 AGENTS.md - Factory Documentation
https://docs.factory.ai/cli/configuration/agents-md

15 16 17 raw.githubusercontent.com
https://raw.githubusercontent.com/agentmd/agent.md/refs/heads/main/README.md

18 21 42 CLI · Cursor
https://cursor.com/cli

19 22 Cursor Agent CLI
https://aiengineerguide.com/blog/cursor-agent-cli/

20 44 MLQ.ai | Stocks
https://mlq.ai/news/cursor-releases-major-cli-version-bringing-ai-programming-directly-to-the-terminal/

[23] raw.githubusercontent.com
https://raw.githubusercontent.com/synthetic-lab/octofriend/main/OCTO.md

[24] GitHub - synthetic-lab/octofriend: An open-source coding helper. Very friendly!
https://github.com/synthetic-lab/octofriend

[25] [26] GitHub - shareAI-lab/Kode: Like Claude Code, but Koding with DeepSeek V3.1, Kimi2, GLM4.5, Qwen Coder etc.（welcome to use Kode to summit PR)
https://github.com/shareAI-lab/Kode

[27] Introduction - Youtu-Agent
https://tencentcloudadp.github.io/youtu-agent/

[28] raw.githubusercontent.com
https://raw.githubusercontent.com/attogram/agents/refs/heads/main/AGENTS.md

[29] [30] raw.githubusercontent.com
https://raw.githubusercontent.com/attogram/agents/main/agents/jules.md

[31] raw.githubusercontent.com
https://raw.githubusercontent.com/attogram/agents/main/HUMANS.md

[32] raw.githubusercontent.com
https://raw.githubusercontent.com/attogram/agents/main/platforms/python.md

[43] One Source of Truth for AI Instructions: single AGENTS.md - Kaushik Gopal's Website
https://kau.sh/blog/agents-md/