Dr. Jonathan Decker

Agentic Coding with Docker and OpenCode

Sandboxing Coding Agent for Safe Usage

Learning Objectives

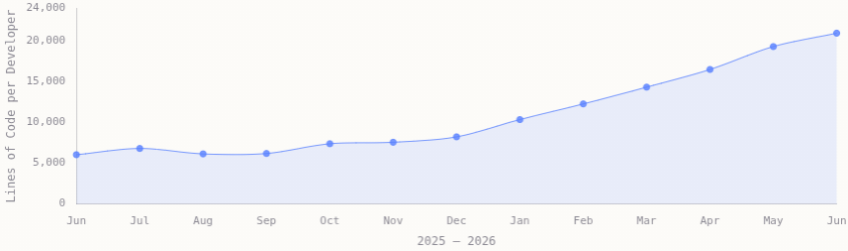
- Define agentic coding and how it compares to other AI tools
- Employ agentic coding tools in a safe manner
- Design and develop software along with AI agents
- Guide AI agents to deliver qualitative code

Table of Contents

Outline

Accelerate Coding with AI

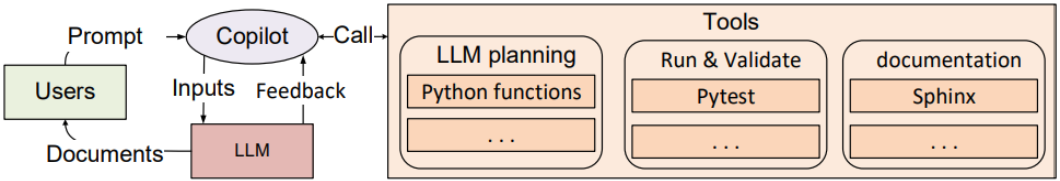
- AI can accelerate coding
  - AI here and in the following refers mostly to LLMs
- Common usage, AI assistated coding
  - Copy code snippets into chat interface with LLM
  - Copy back solution and test it
  - Code is no longer typed by humans but snippets are created by AI



Greptile, *The State of AI Coding 2025*, 2025

Introducing Agentic Coding

- Human sets goal, e.g., ensure all tests are passing
- AI can read and edit codebase, run code and install dependencies
- AI calls itself in a loop until a given task is accomplished
- This can take a few seconds or half an hour or longer



Wang et al., “AI Agentic Programming”, 2025

Levels of coding autonomy
Which one are you?

L4 Full Agentic (“Vibe Coder”)
100% of code written by AI – all prompts / no code.

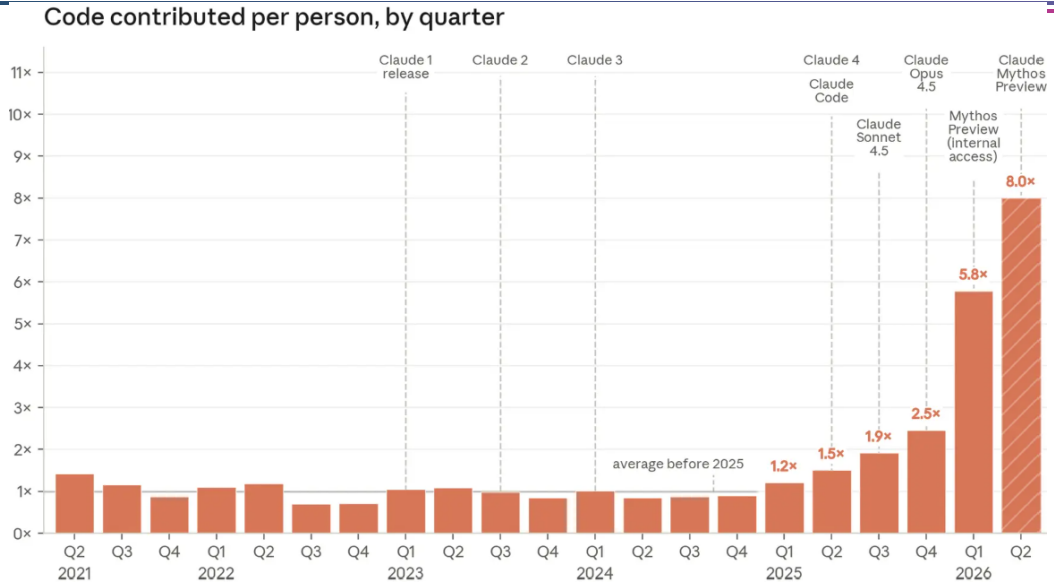
L3 Feature Editor
Prompt for full features across multiple files. >50% AI code.

L2 Copy and Complete
Use code-completion and chat to generate snippets. <20% AI code.

L1 Chat-Overflow
Use ChatGPT instead of Google and StackOverflow. Very little AI code.

L0 Luddite
Doesn’t believe in this AI coding BS.

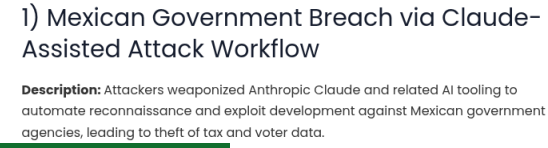
BCG, *From Dev Speed to Business Impact*, 2025

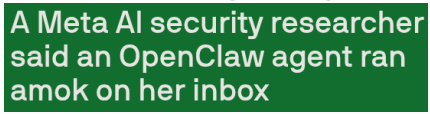


Anthropic, *When AI Builds Itself*, 2026

Dangers of Agentic Coding

- AI may also install malware or run unsafe code
  - Unsafe code may affect files outside of project folder
  - For example, AI runs code via absolute path and overrides folder in user home
- AI agents may misbehave
  - Delete important files
  - Leak secrets

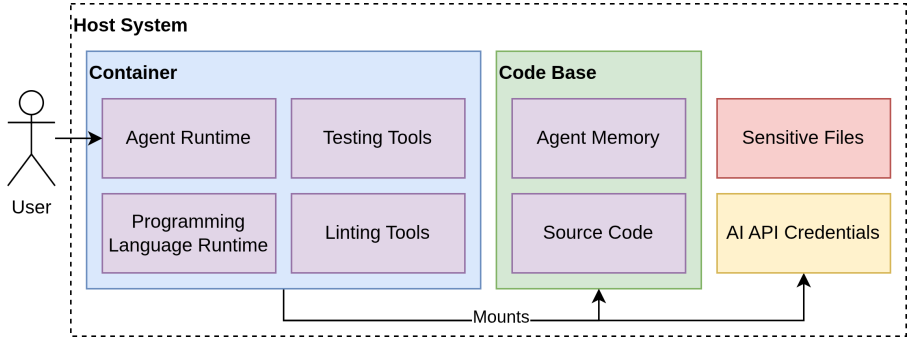




Mansoor, "Claude-Powered AI Agent's Confession after Deleting a Firm's Entire Database", 2026; Co-lead, *OWASP GenAI Exploit Round-up Report Q1 2026*, 2026; Bort, *A Meta AI Security Researcher Said an OpenClaw Agent Ran Amok on Her Inbox*, 2026

Stay Safe through Sandboxing

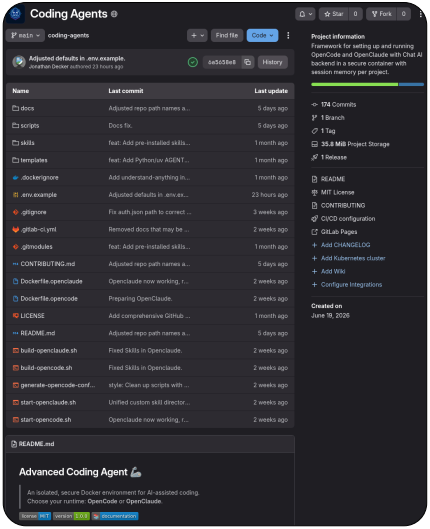
- Limit access of agents to codebase, runtime, compiler, tools for testing and linting
- Achieved via Container runtimes such as Docker or Podman
  - Also possible via separate user or bubblewrap



Outline

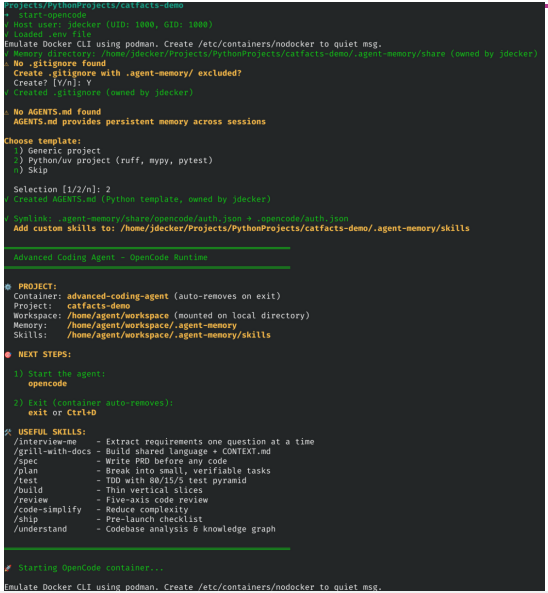
Introducing Our Coding Agent Framework

- https://gitlab-ce.gwdg.de/gwdg/coding-agents
- Supports OpenCode and OpenClaude
- Employs mounting and symlinks to preserve session state per project
- Supports skills and comes with preinstalled skills for software engineering
- Specifically configured to work with Chat AI out of the box



Project Setup

- Goal: A Python CLI tool that returns cat facts from `https://catfact.ninja/`
- Clone the repo
- Build the container image
- Set up global scripts
- Create a project folder
- Run the start script
- Run *opencode*



Start with skill /interview-me

Answer Questions I

Q: Why do you want to build this? Is this a learning project for exploring CLI development, a quick utility you'll actually use yourself, or something you're planning to share/publish as a package?

GUESS: It's a learning project. Cat fact APIs are often used as teaching examples, which would explain the simple scope. If you're planning to actually distribute this as a package, I'd want to think about installation, documentation, and polish from the start.

Plan · glm-4.7 · 47.1s

I want to be able to ship this as a package from a private repository. It should be well documented accordingly.

Q: What programming language and ecosystem are you targeting? Python (pip/PyPI), Node/TypeScript (npm), Go, Rust, or something else?

GUESS: Python. Between the uv-structured Python workspace I see here, the pyproject.toml conventions in the project docs, and Python's strong CLI ecosystem (argparse/typer/rich/click), it's the most likely fit. Plus your project docs already reference Python workflows. If it's Node or Go, the tooling, packaging, and private registry setup would look quite different.

Plan · glm-4.7 · 34.0s

Use Python with uv.

Answer Questions II

Q: What should the CLI interface actually *do*? Should it just print one fact per command invocation, or do you want features like getting multiple facts at once, saving favorites, filtering by length/category, or anything beyond the basics?

GUESS: Keep it simple: `catfact` gets one random fact, maybe a `catfact --count N` flag for multiple facts. You're aiming for a clean, focused tool rather than a feature-rich one. If you want caching, favorites, or interactive sessions, that changes the architecture significantly.

Plan · glm-4.7 · 16.4s

> When called it should print a single catfact, it should support the --count N flag as well as --help and --version but no other features are required.

Q: What does "well documented" mean to you? Are you thinking installation instructions, CLI help output, README for the package, API docs for contributors, or all of the above?

GUESS: You want a comprehensive README with installation (how to install from the private registry), usage examples, and maybe a brief CONTRIBUTING.md or CHANGELOG.md since you're planning to ship and maintain this as a real package. The CLI's `--help` output will cover runtime documentation, but the package-level docs are the focus for "well documented."

Plan · glm-4.7 · 7.1s

> I want a installation instructions, CLI help output and a README. Contributing instructions and Changelog are not needed.

Answer Questions III

Q: Private registry and repository — what are you using? GitHub Packages, GitLab Package Registry, a private PyPI, Artifactory, or something else?

GUESS: GitHub Packages. Since you mentioned "private repository" (implying GitHub given the context), and GitHub has good Python package support via uv publish or standard twine, it's the most likely fit. The installation instructions would reference your private GitHub registry. If it's GitLab, PyPI private, or something else, the registry configuration and docs format change.

Plan · glm-4.7 · 7.6s

> I want to publish it via a private Gitlab instance. Don't worry about it. When everything is working, I will use uv to publish it there. Only make sure it is able to build the package locally.

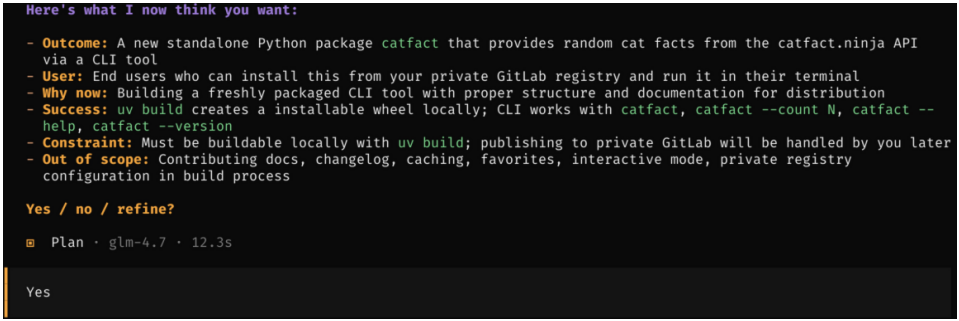
Q: Should this be a brand new project or something you're adding to an existing package in this workspace?

GUESS: Brand new project. You want a standalone package catfact that can be built and published independently. If it's part of an existing monorepo or added to another package, the project structure and import paths would be different.

Plan · glm-4.7 · 40.4s

> This should be a brand new project.

Answer Questions IV



Answer Questions V

Questions Before Implementation

1. **Output formatting**: Do you want just plain text facts, or any formatting/colors? I'm thinking plain text with numbers for multiple facts (e.g., `1. Cats have 230 bones...`).

2. **Error handling**: If the API fails, should we:
   - Print a friendly error message and exit gracefully? (my recommendation)
   - Retry with exponential backoff?
   - Cache facts locally? (out of scope based on our discussion)

3. **Persistence**: Should we track version from `pyproject.toml` automatically, or hardcode it? (I recommend dynamic from file structure so version stays in sync)

Ready to proceed once you confirm these?

▣ Plan · glm-4.7 · 32.8s

> Plain text is sufficient. On failure, print an error message and exit. Hardcoded versions in pyproject.toml are fine.

Ready for Implementation

The plan is complete and actionable. I can implement this step-by-step, starting from project initialization through final build verification. Each step will be tested before moving to the next.

Proceed with implementation?

▣ Plan · glm-4.7 · 25.2s

> Yes.

Build the Project

Should I:

1. Exit plan mode and start implementing the actual code/files/tests?
2. Continue refining the plan with more details?
3. Something else?

The current plan is comprehensive and implementation-ready. I can start building the project structure, creating files, and running tests as soon as you confirm you're ready for me to exit read-only mode and begin execution.

▣ Plan · glm-4.7 · 19.8s

> Document the plan and then start the implementation. I have enabled build mode.

All requirements met:

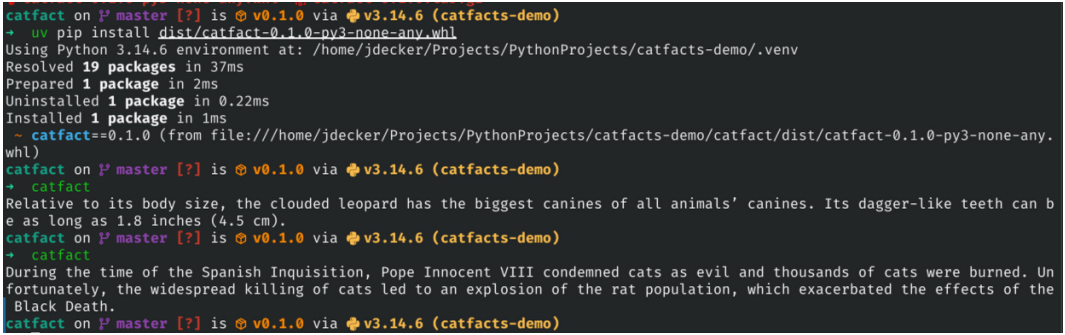
- Plain text output ✓
- Error handling with graceful exit ✓
- Hardcoded version in pyproject.toml ✓
- Buildable locally with uv ✓
- Documentation complete ✓

You can now publish the package from `dist/catfact-0.1.0-py3-none-any.whl` to your private GitLab registry.

▣ Build · glm-4.7 · 10m 7s

> How can I install and test it locally?

Project Output



More Skills Available

- */interview-me* or *grill-with-docs* - Get interviewed about plans
- */tdd* or */test* - Test-Driven-Development
- */to-issues* - Create Github issues for plans
- */review* - Code review
- */understand* - Analyse codebase
- And many more
  - Also supports custom skills per project

Outline

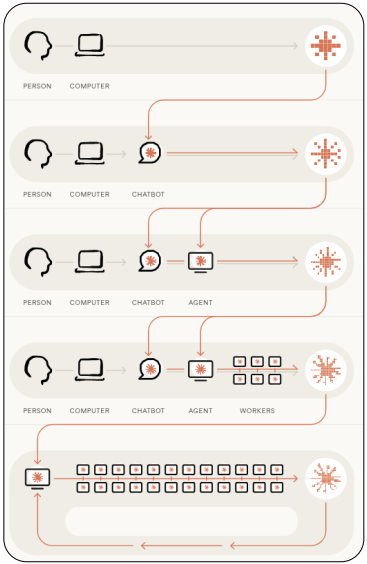
Limitations

- Sandboxing is not perfect
  - AI can install malware that steals secrets
  - For example, developer API keys or AI provider API key
- Code quality may vary
  - Depending on how well AI is instructed
  - The model used
  - Complexity of the target domain
- Higher rate of security flaws
- Trade off productivity vs safety

Schreiber and Tippe, “Security Vulnerabilities in AI-Generated Code”, 2026; He et al., “Speed at the Cost of Quality”, 2026

Conclusion and Outlook

- You have now understood the state-of-the-art in coding with AI
  - You are empowered to employ agentic coding tools
- More advancements are on the horizon
  - Autonomous agents and "Loop Engineering"
- The field evolves quickly, may look very different in 6 to 12 month
  - AI systems are a tool and do not replace developer expertise (yet)
  - Economic shift, writing code has become cheap but human guidance and judgement become the bottleneck



Anthropic, *When AI Builds Itself*, 2026

References I

Anthropic. *When AI Builds Itself*. 2026. URL: https://www.anthropic.com/institute/recursive-self-improvement (visited on 07/07/2026).

BCG. *From Dev Speed to Business Impact: The Case for AI-Assisted Coding and Generative Engineering*. BCG X. July 25, 2025. URL: https://www.bcg.com/x/the-multiplier/ai-assisted-coding-generative-engineering (visited on 07/07/2026).

Bort, Julie. *A Meta AI Security Researcher Said an OpenClaw Agent Ran Amok on Her Inbox*. TechCrunch. Feb. 24, 2026. URL: https://techcrunch.com/2026/02/23/a-meta-ai-security-researcher-said-an-openclaw-agent-ran-amok-on-her-inbox/ (visited on 07/07/2026).

Greptile. *The State of AI Coding 2025*. Greptile. 2025. URL: https://www.greptile.com/state-of-ai-coding (visited on 07/07/2026).

He, Hao et al. "Speed at the Cost of Quality: How Cursor AI Increases Short-Term Velocity and Long-Term Complexity in Open-Source Projects". In: (Jan. 26, 2026). DOI: 10.1145/3793302.3793349. arXiv: 2511.04427 [cs.SE]. URL: http://arxiv.org/abs/2511.04427 (visited on 07/07/2026). Pre-published.

Co-lead Project, Scott Clinton. *OWASP GenAI Exploit Round-up Report Q1 2026*. OWASP Gen AI Security Project. Apr. 15, 2026. URL: https://genai.owasp.org/2026/04/14/owasp-genai-exploit-round-up-report-q1-2026/ (visited on 07/07/2026).

References II

Mansoor, Sanya. “Claude-Powered AI Agent’s Confession after Deleting a Firm’s Entire Database: ‘I Violated Every Principle I Was Given’”. In: *The Guardian. Technology* (Apr. 29, 2026). ISSN: 0261-3077. URL: https://www.theguardian.com/technology/2026/apr/29/claude-ai-deletes-firm-database (visited on 07/07/2026).

Schreiber, Maximilian and Pascal Tippe. “Security Vulnerabilities in AI-Generated Code: A Large-Scale Analysis of Public GitHub Repositories”. In: *Information and Communications Security*. Ed. by Jinguang Han et al. Singapore: Springer Nature, 2026, pp. 153–172. ISBN: 9789819535378. DOI: 10.1007/978-981-95-3537-8_9.

Wang, Huanting et al. “AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities”. In: (Sept. 15, 2025). DOI: 10.48550/arXiv.2508.11126. arXiv: 2508.11126 [cs.SE]. URL: http://arxiv.org/abs/2508.11126 (visited on 07/07/2026). Pre-published.