```
$ ./begin-fast-takeoff
```

How to build autonomous AI agents

Brian Kelleher

Founder and CEO of Microdoc

microdoc.io

briankelleher.ie

Favourite AI coding tool?

Hands up!

Claude code?

Hands up!

Codex?

Hands up!

Cursor?

Hands up!

Factory AI?

Hands up!

Conductor?

Hands up!

Gemini CLI?

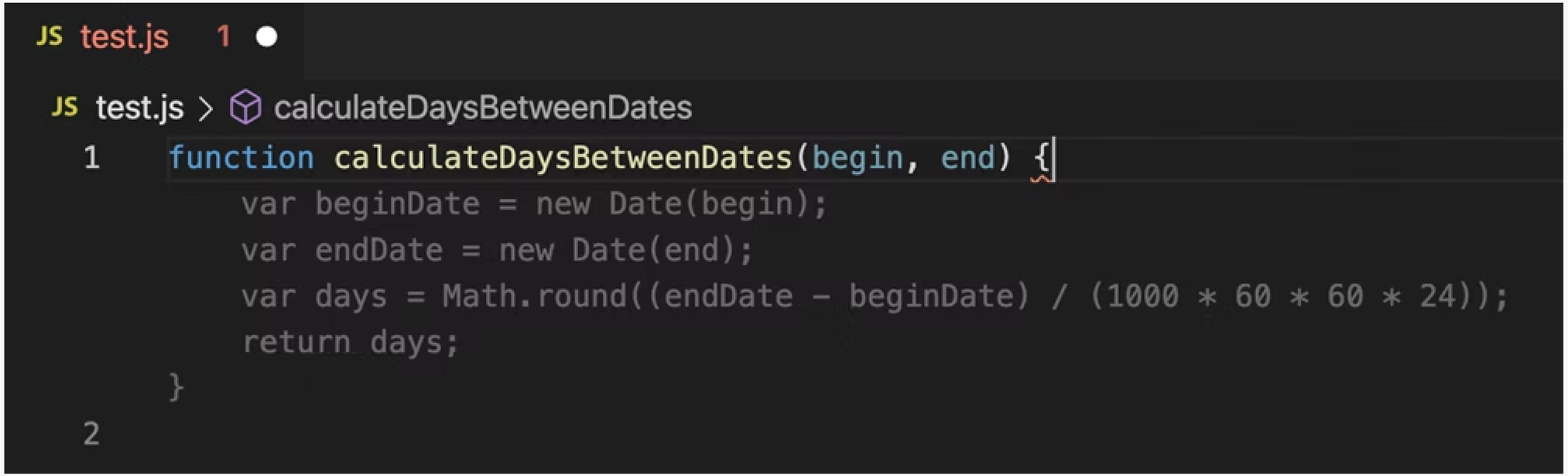
(You’re braver than me if you do)

Goal: Build autonomous AI agents that ship production code

Supabase
@supabase

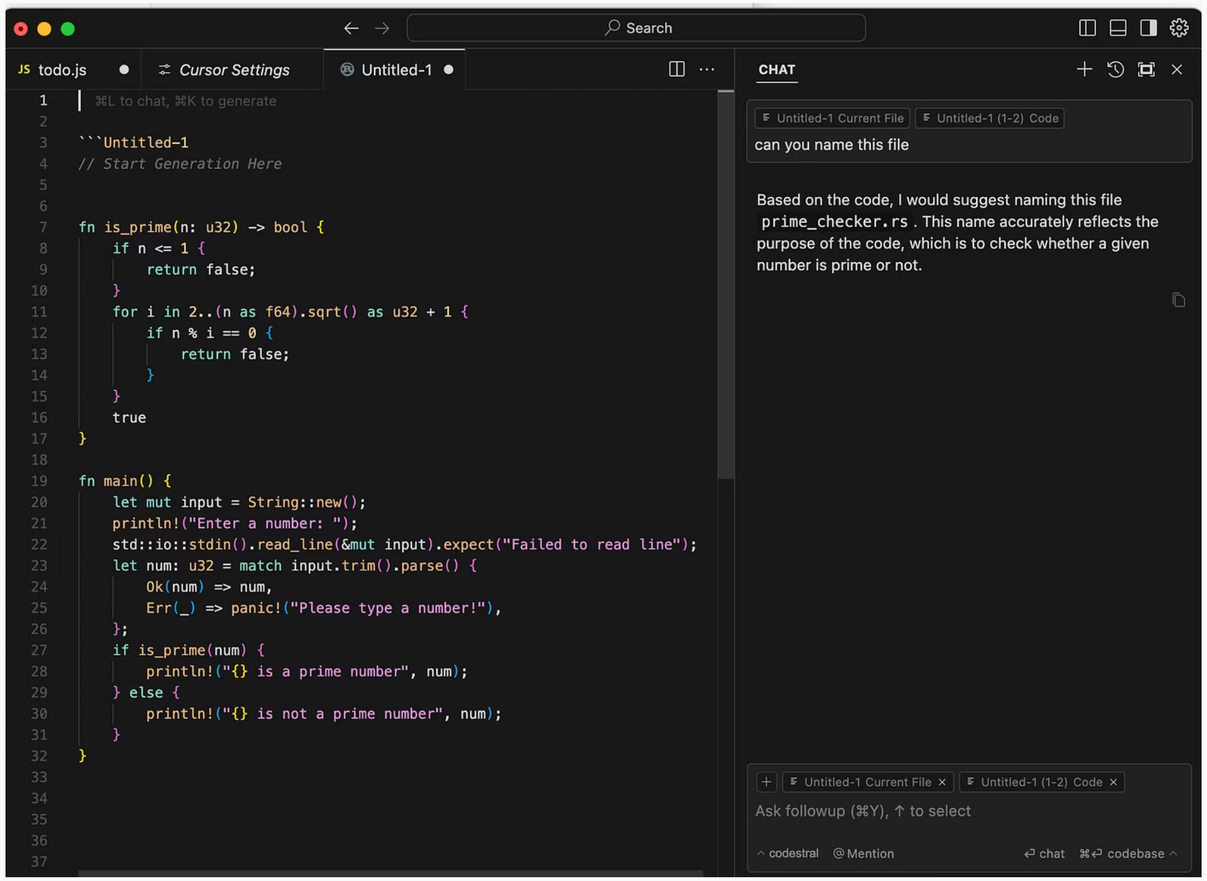
Supervising while Cursor deletes files at random

Evolution of AI agents



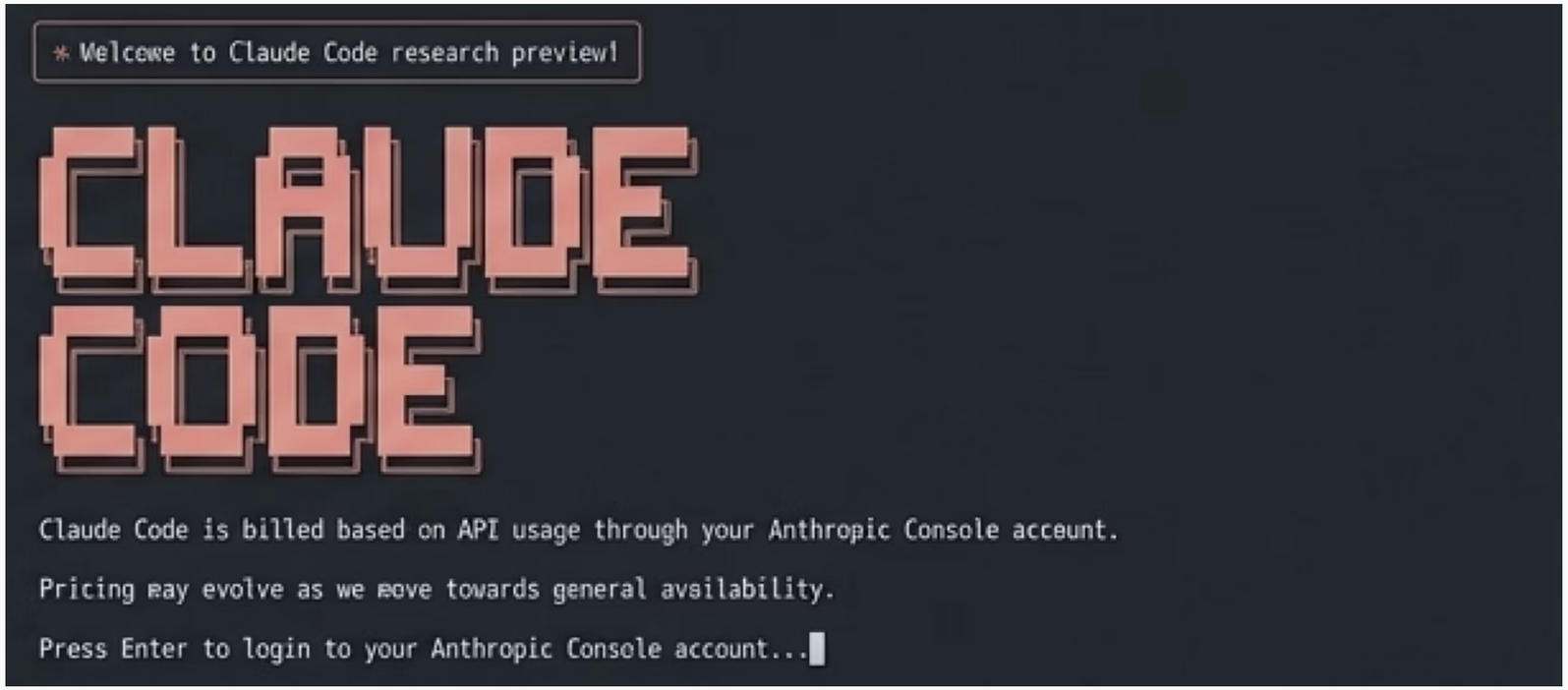
Github copilot, circa 2023

Evolution of AI agents



Cursor, circa 2024

Evolution of AI agents



Claude Code, circa February 2025

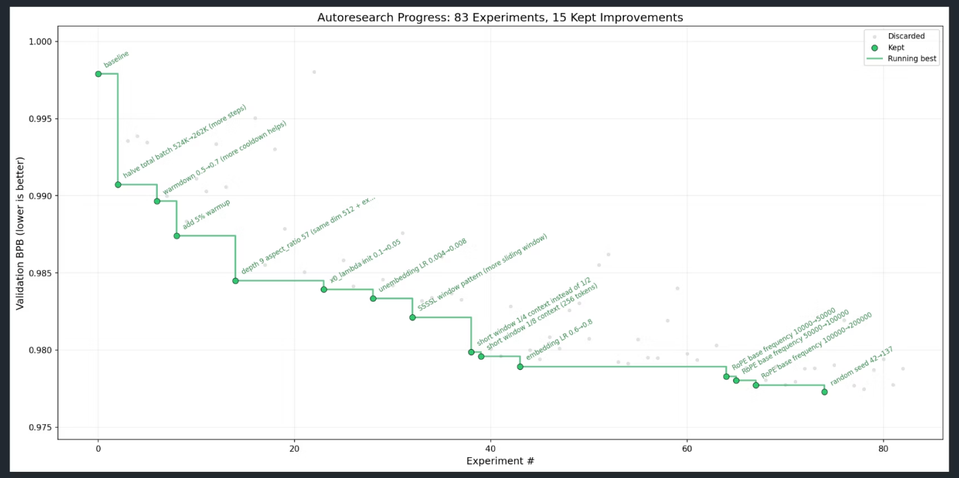
Evolution of AI agents

OpenClaw (Clawdbot), circa November 2025

Where are we now?

README

autoresearch



One day, frontier AI research used to be done by meat computers in between eating, sleeping, having other fun, and synchronizing once in a while using sound wave interconnect in the ritual of "group meeting". That era is long gone. Research is now entirely the domain of autonomous swarms of AI agents running across compute cluster megastructures in the skies. The agents claim that we are now in the 10,205th generation of the code base, in any case no one could tell if that's right or wrong as the "code" is now a self-modifying binary that has grown beyond human comprehension. This repo is the story of how it all began. -@karpathy, March 2026.

Andrej Karpathy

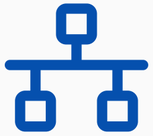
https://github.com/karpathy/autoresearch

What makes autoresearch so good?

Ability for the Agent to check its work

Work over long time horizons

Places for it to plan/scratchpad

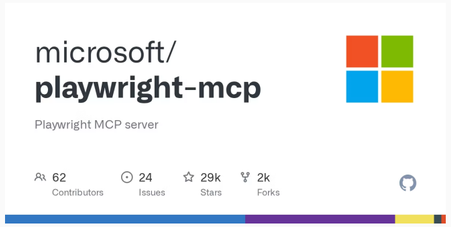


Infrastructure/tooling to support long autonomous runs

IMPORTANT!

1. Ability for the Agent to check its work

Playwright MCP



“Take a screenshot to see if the UX makes sense”

Unit tests/Type checks (CI tooling)

pytest

: my[py]

External Data Sources/Validation

DATADOG

facebook Ads

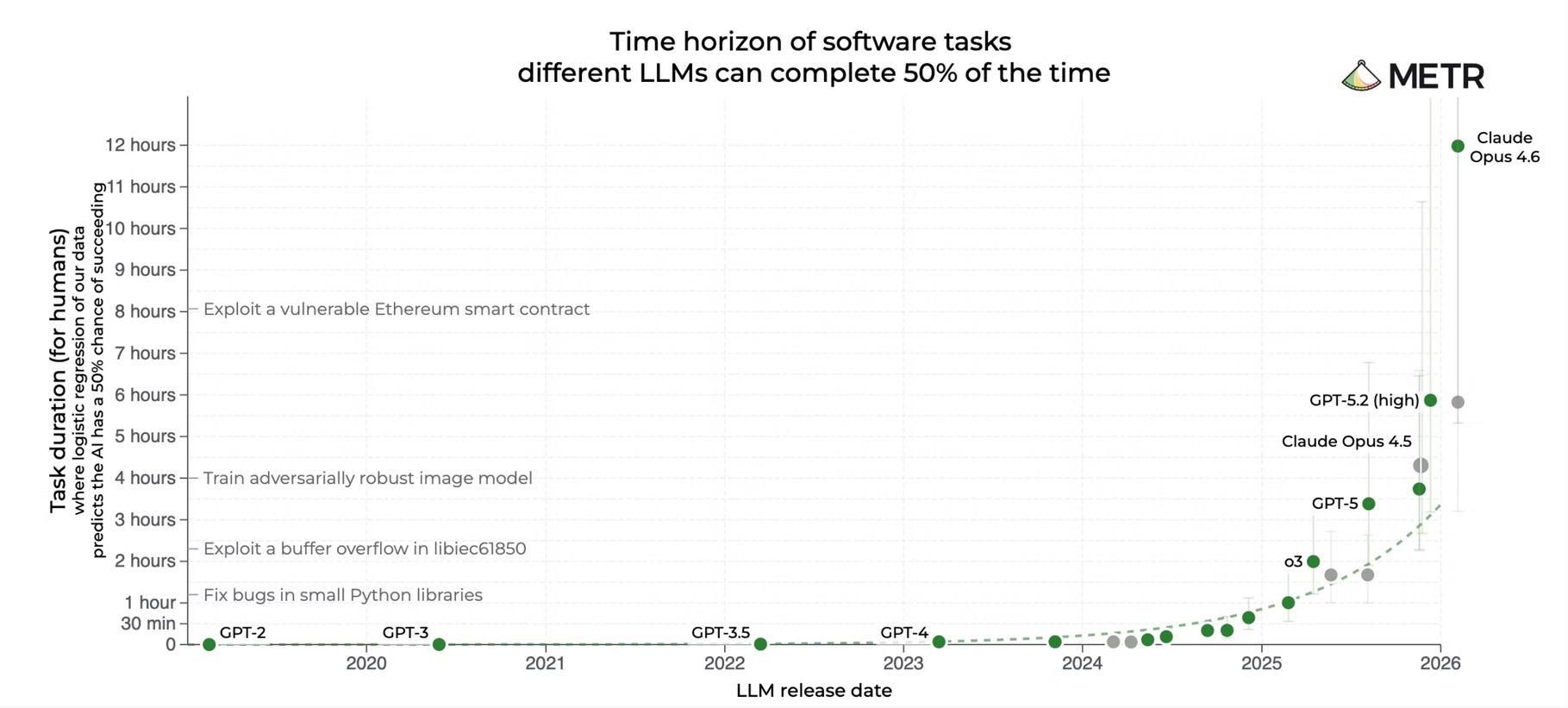
Never MCP, always CLI

Integration/E2E tests

Playwright

```
$ ./call-live-api “Foo Bar”
```

2.Work over long time horizons



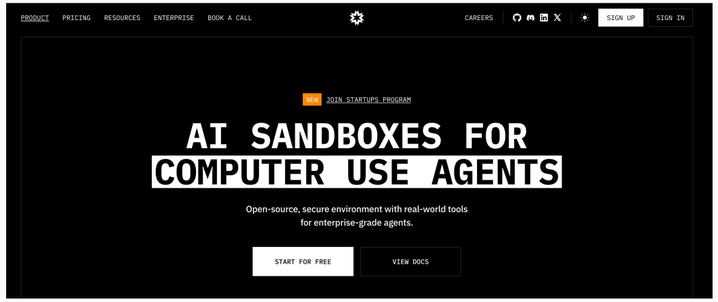
Inflection point: 5th Feb, 2026, release of gpt-5.3-codex

3. Scratchpad/plan

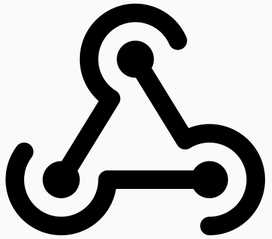
- Keeps them consistent within sessions and across sessions
- Allows for retrospective analysis
- Self-evolving/improving (e.g. on the basis of a fix, put in place documentation that makes it easy to avoid this bug in the future)

Give agents a filesystem and let them run bash!

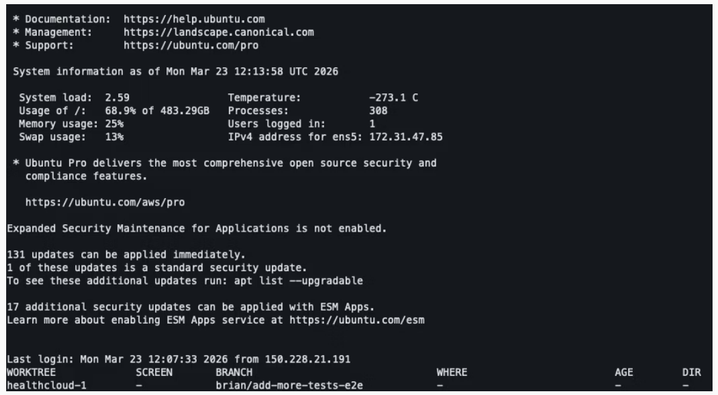
4. Infrastructure/tooling



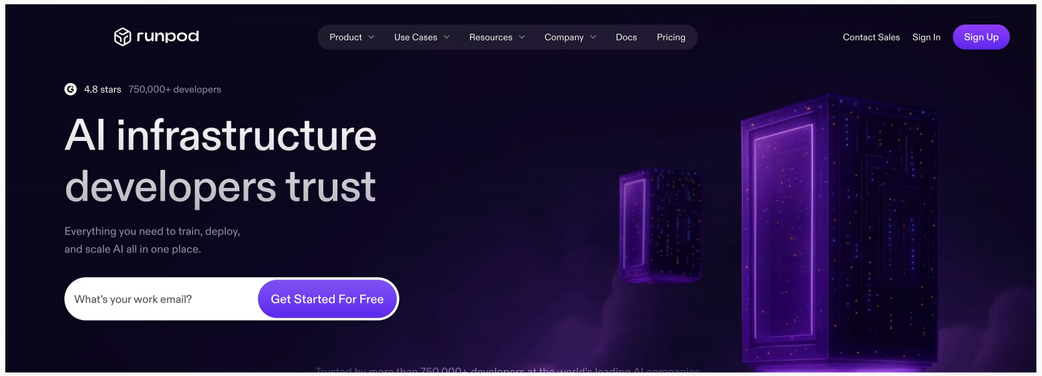
Managed virtualised cloud sandboxes



Webhooks and event-driven agent flows



Cloud ec2 instances



Tooling for AI to run its own experiments

Note: Agents change how we should develop

- Optimise for skimmability
- Let go of code perfectionism. Let it go.
- Parallelise aggressively
- Stop planning, and start prompting. Skip project management and just ship

How do we put all of these things into practice in a new codebase?

AI-native Codebase specs:

- React with Typescript
- Postgres on Supabase
- Monorepo
- Vercel deployment
- Drizzle ORM

Improve tooling

Add automated coding

Improve verification

Improve documentation

Self improving loop

Optimise code review

1. Improve tooling

Always use CLIs (MCPs bloat context)

See this paper on context rot: https://research.trychroma.com/context-rot

Install CLIs:

Type checking/linting

Very strict tsconfig

2. Event-driven coding

Use webhooks for event-driven coding

Example workflow #1

PR opened → AI code review → GH webhook fires for new comment

Lightweight endpoint receives GH webhook request and spawns a codex cloud instance

Fix this comment “`There are 4 lint e…`” then commit/push to the branch

2. Event-driven coding

Use webhooks for event-driven coding

Example workflow #2

Error detected → Sentry webhook fires for new issue

Lightweight endpoint receives sentry webhook request and spawns a codex cloud instance

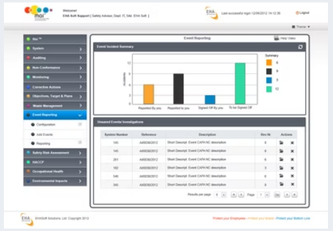
Fix this error `<traceback.xml.stri…` then create a PR for this change

3. Improve verification

Playwright, development API keys

Playwright

“Let me test this page to ensure it works properly. I’ll record a video”



Vercel

“I’m deploying to vercel to check that the CSRF issue is resolved”

zero bounce

“I’m running a test email through the zerobounce implementation”

4. Improve documentation

Give the agent a workspace

Setup project script: https://briankelleher.ie/projects/setup-project

```
projects/
└── active/
    └── YYYY_MM_DD_/
        ├── .gitignore
        ├── README.md
        ├── USER_STORY.md
        ├── REQUIREMENTS.md
        ├── DESIGN.md
        ├── VALIDATION_STRATEGY.md
        ├── STATUS_UPDATES.md
        ├── PROGRESS_TRACKER.md
        ├── RETROSPECTIVE.md
        ├── scripts/
        │   └── README.md
        ├── docs/
        ├── tests/
        └── spikes/
```

Improves consistency within session as well as from session to session

5. Self-improving loop

Agent improves over time

“What was the root cause of this bug? How can we update AGENTS.md so we avoid this bug in the future?”

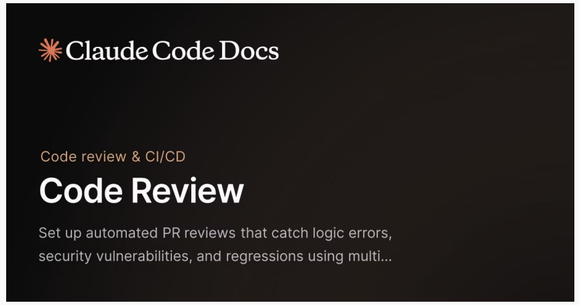
“Is there a check that we can do to prevent a similar regression in the future?”

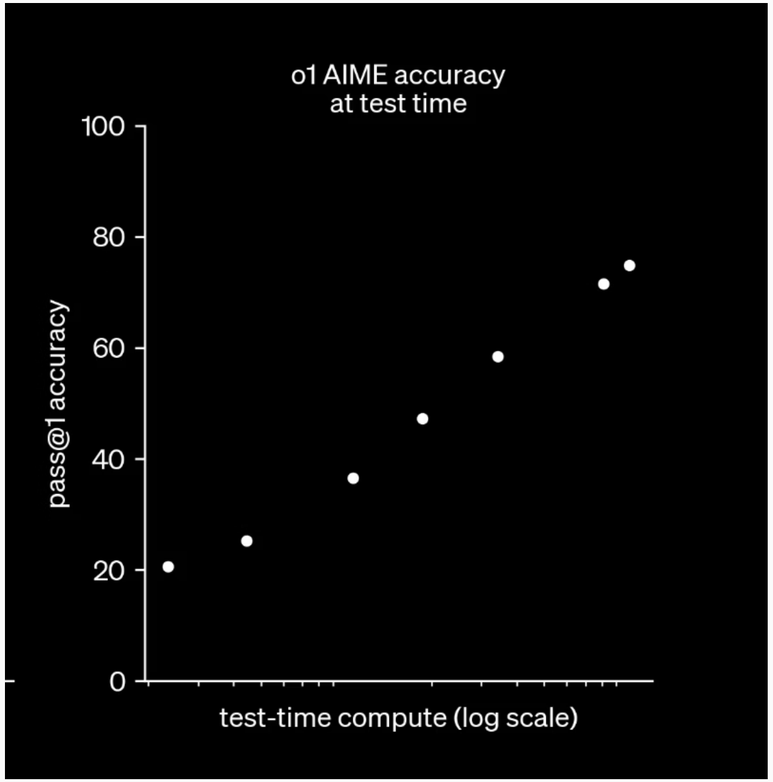
6. Optimise code review

Use several coding agents for review

CodeRabbit

greptile





Scaling inference time compute

```
$ ./predict-future
```

Try Pitch

Questions?

Brief demo of ec2 setup

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It’s free)