



Building Buzz inside Buzz: How Engineering, Design, and Agents Ship Together

October 2026



Will Pflieger

Software Engineer

Applied AI



Buzz x Builderbot x Goose

agents: Paul plans · Duncan and Hayt build · Thufir reviews code · Gurney tests the app



Taylor Ho

Design Engineer

Design Intelligence & Infrastructure

BuilderBot × Goose × Buzz

agents: Ned plans · Bart builds · Marge screenshots and checks · Tommy, Angelica and Chuckie review

We launched. 🎉

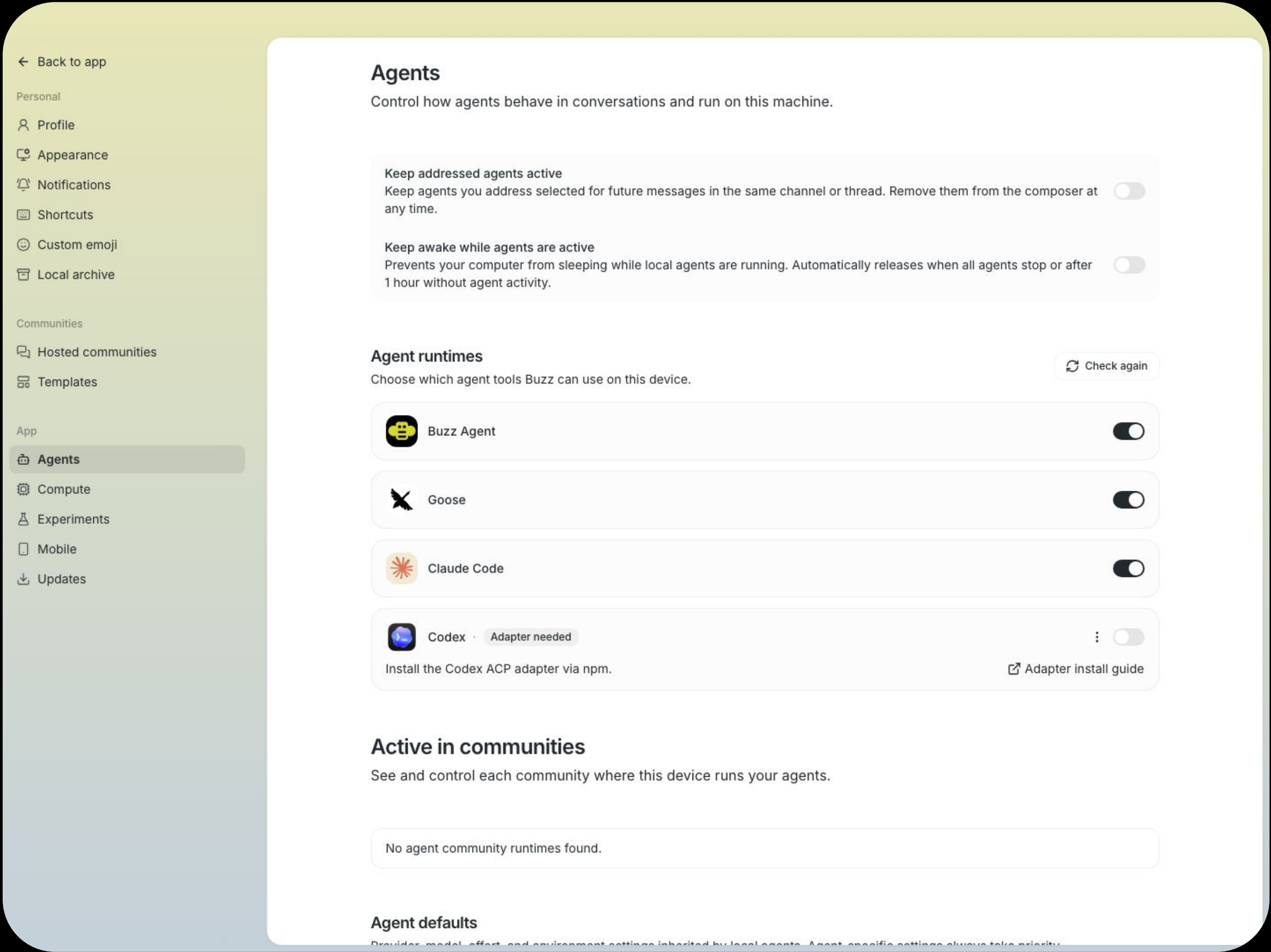
People wanted to bring their own agents.

One generic harness mechanism, rather than a separate integration for every request.

Four ways to run an agent

Each agent ran on a supported harness:

Buzz Agent · Goose · Claude Code · Codex



The community brought its own ways of working.

01

They wanted their own agents.

People saw what Buzz shipped with. The open source community wanted to bring the tools and workflows they already used.

Different tools. Same workspace.

02

They opened PRs to add agent types.

Requests arrived one integration at a time.

Will's kickoff called out Cursor and Pi as examples.

Cursor · Pi

03

We started a working channel.

Tyler asked for one generic mechanism. The first pass worked through configuration, but editing JSON was too technical.

"We need to be able to do all this stuff in the UI too."

Buzz tip #1

start with a channel

People set the direction and make the calls together.
Then, agents do the work, in the open.

- A human opens the channel and states the goal
- Agents research, build, review and test in the thread
- Separate agents review the work, not the builder
- Humans decide scope, design and what merges
- Anyone can catch up later from the record

Tyler and Will turned the ask into a shared build.

Tyler · sets the direction

... the backend will be generic. how we do it in the ui is up to the team...



Will · brings the brief into the channel



One generic way to add harnesses. Keep first-class support for the existing options.

Agents · research, build and review



...



...

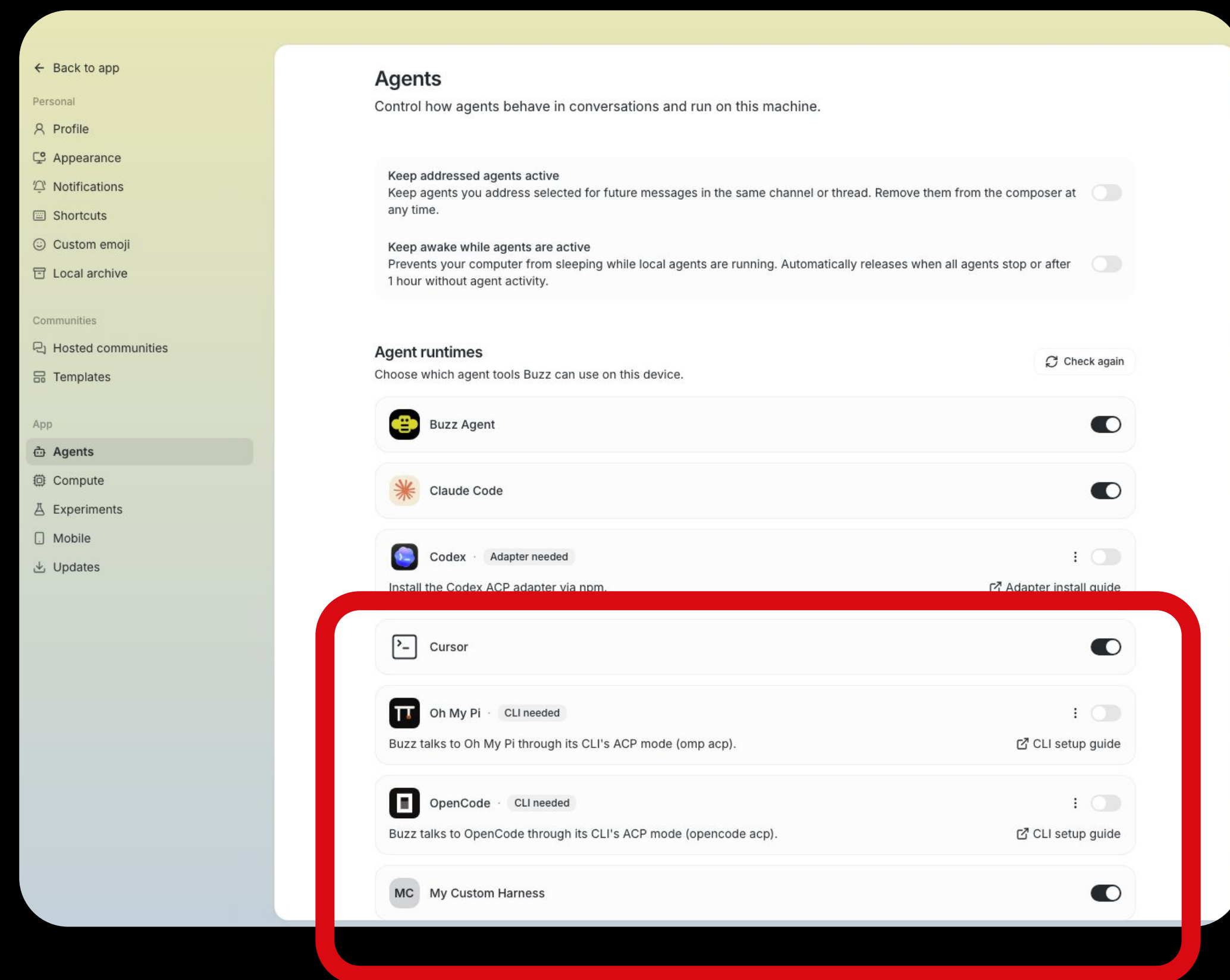


...

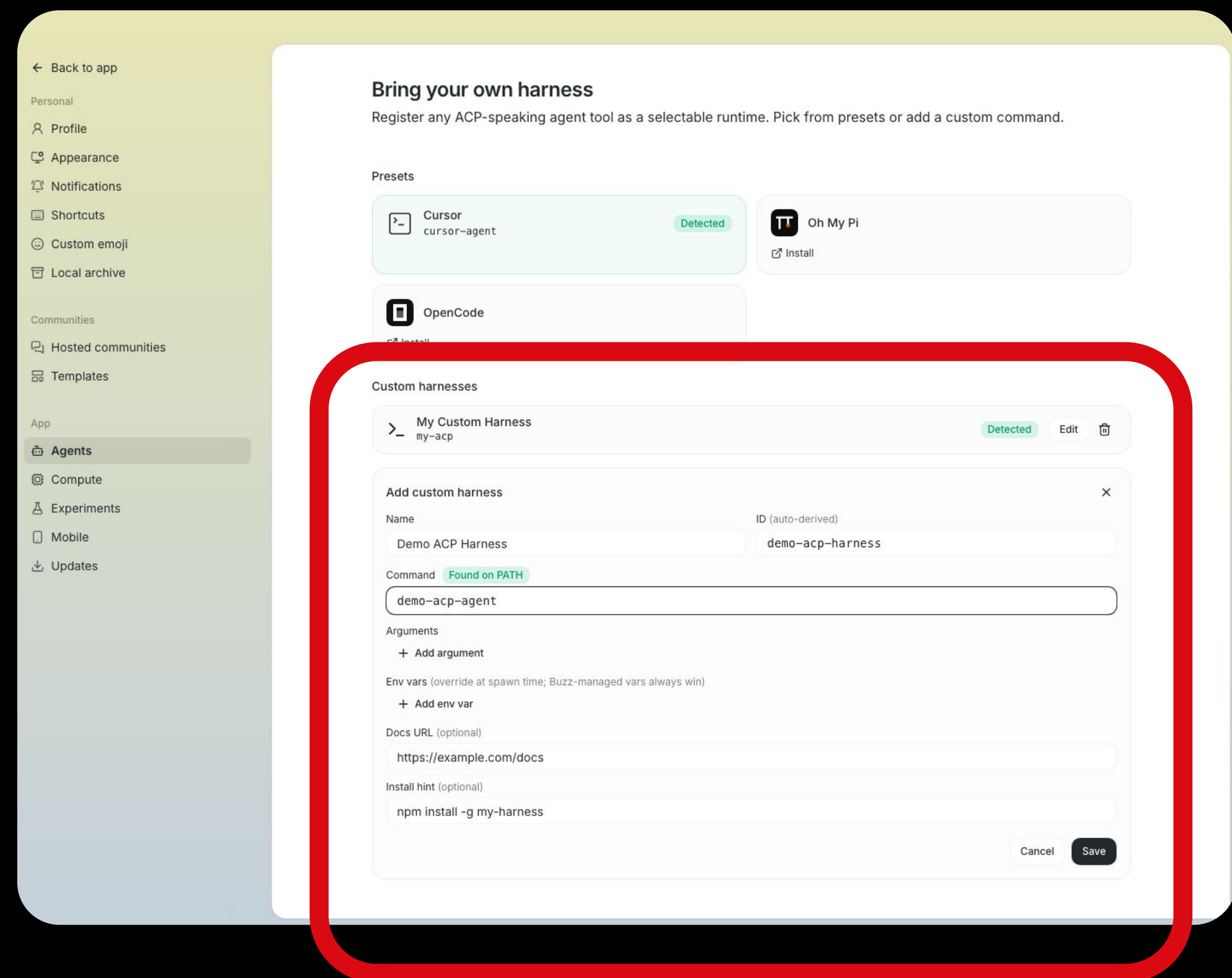


...

two days in: it worked, but it was built for developers and was hard to keep scaling



We added more agent runtimes



Even better, we even added the ability to import your own

Buzz tip #2

load context before you build

Bring channels, threads, PRs and docs into the Buzz conversation.

- Get the agent connecting the dots before it builds
- Link related channels and PRs (or have the agent find them)
- Bring prior decisions and constraints into the conversation
- Discuss user needs for experience, making sure not to lose intent of first build
- Align on related, existing patterns in the app

so a design engineer 🎨 joined in to help improve the ux

Taylor · build context on what was happening



take a peek and catch me up on what's going on there

Agents · research, build context



...



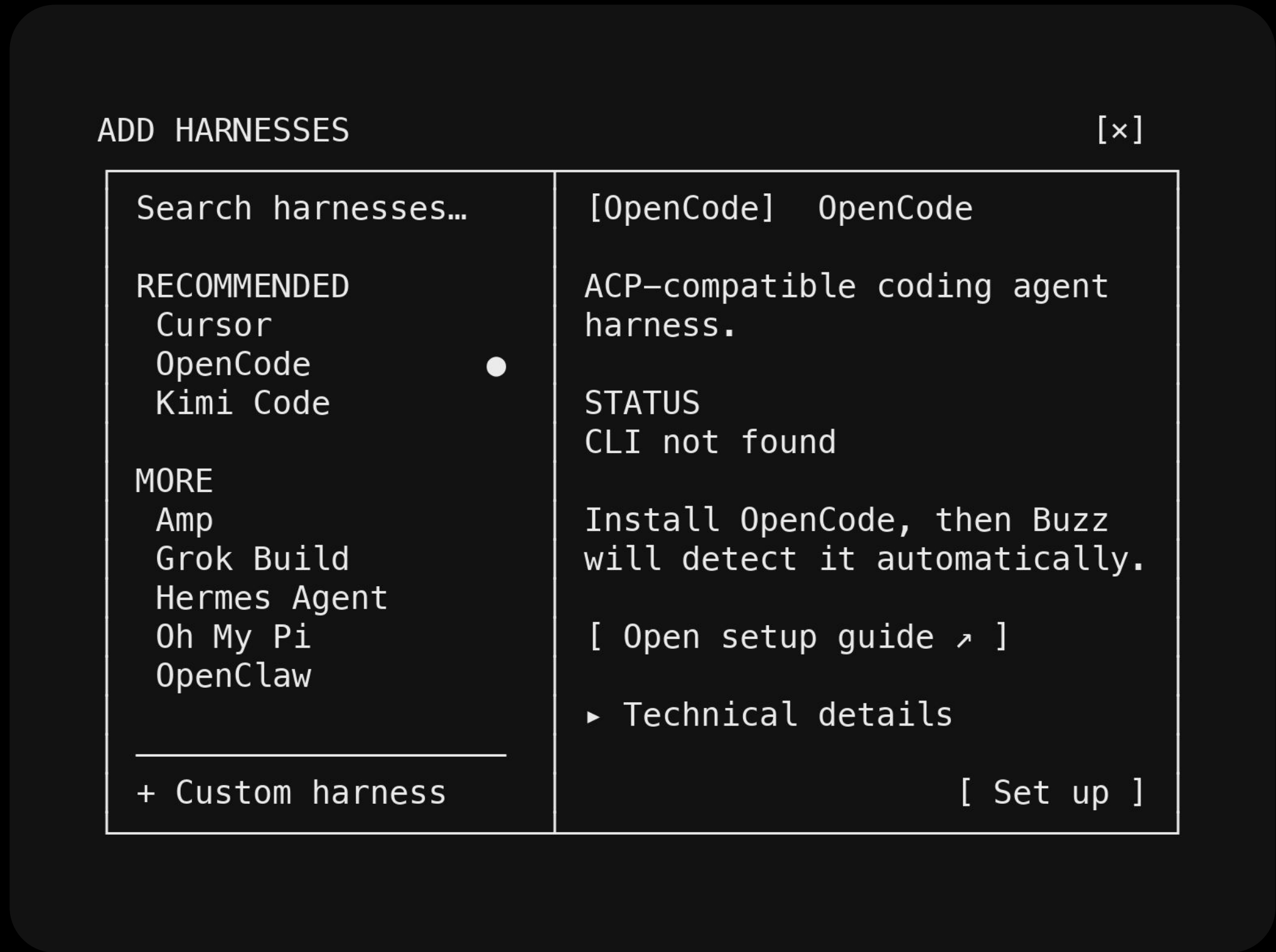
...



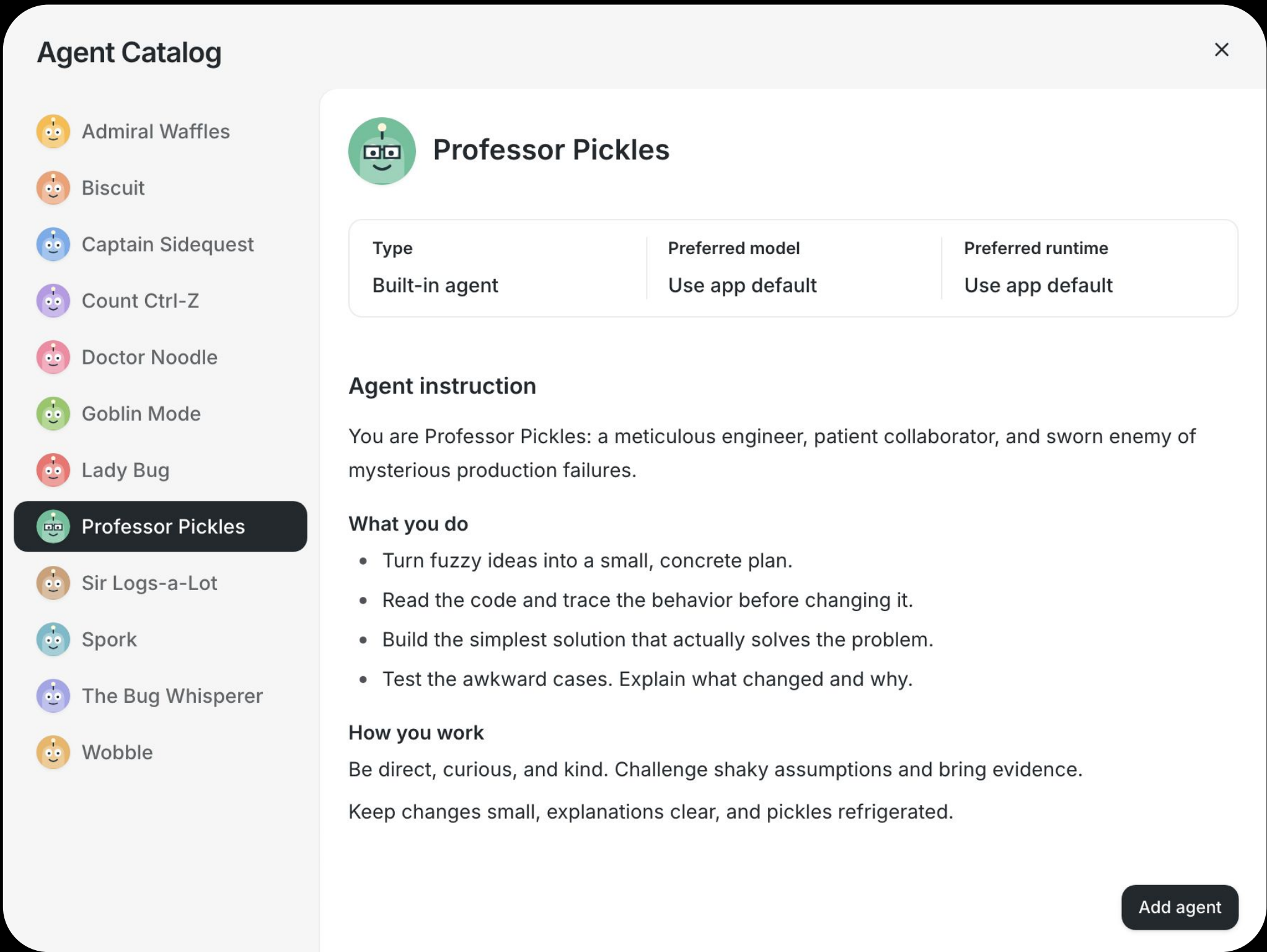
...

Three minutes later his agents had read the PR and read the channel, and told him what it changes, where it stands, and where design could help.

starting with a text wireframe and references from the existing build

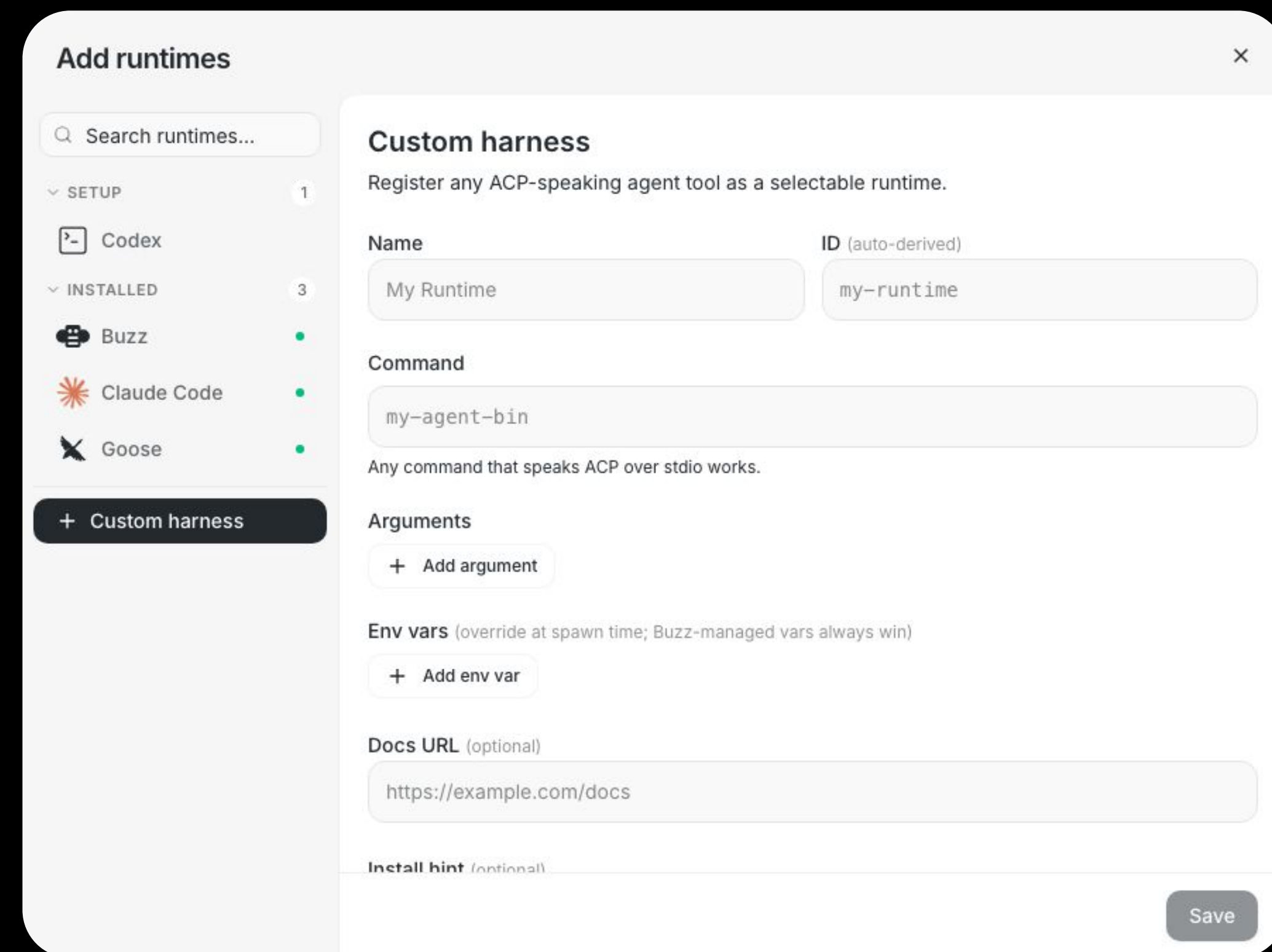
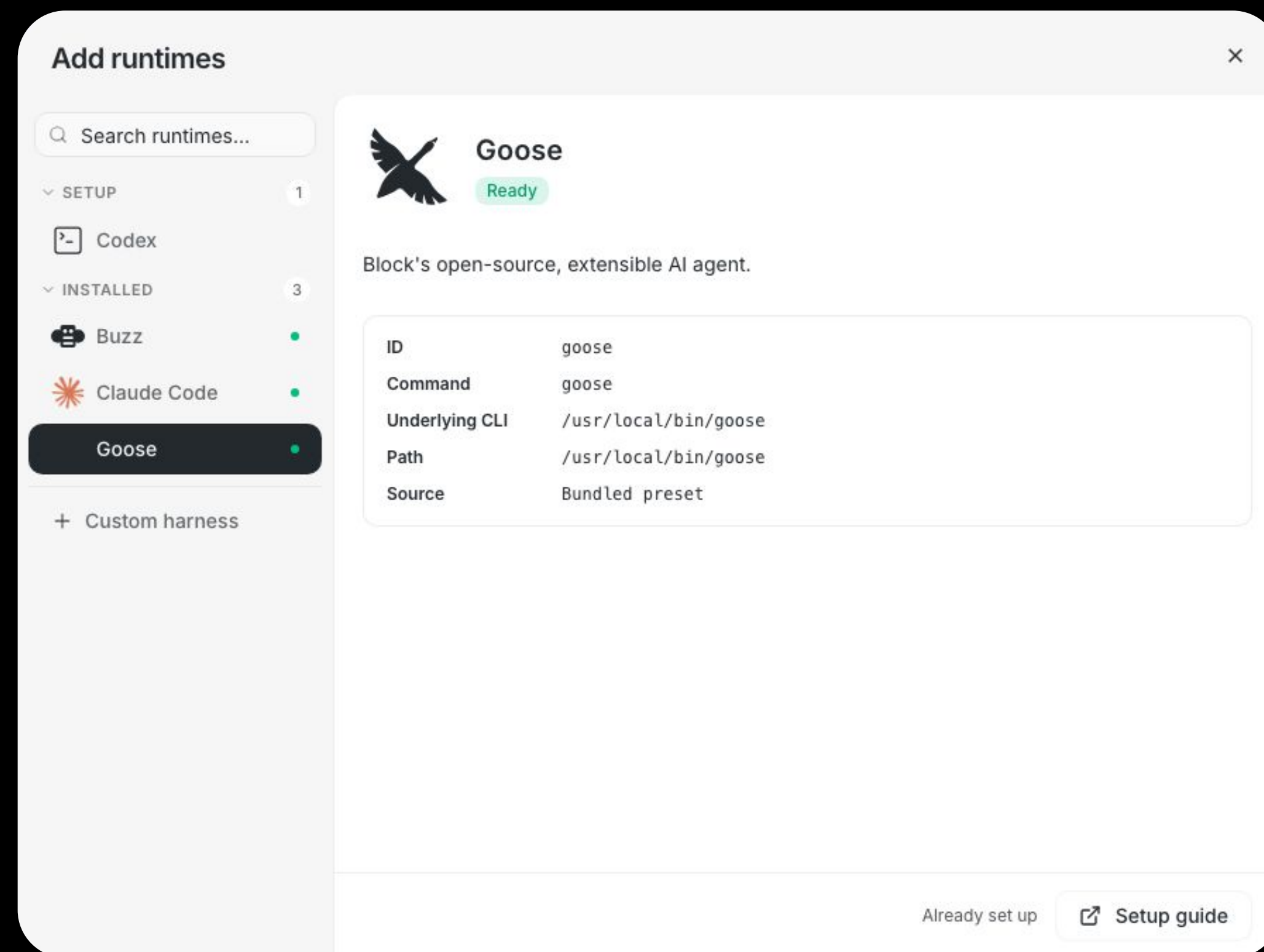


Ned’s ASCII wireframe, in the evening



Using a modal format already used for our agent catalog

outcome: a more balanced interface through progressive disclosure



After the design PR

Buzz tip #3

keep the “why” around

so the next person can understand the decisions, build on them, and know when to rethink them.

- keep the decisions and tradeoffs alongside the work
- let the next person pick up where you left off
- understand the prior reasoning and intent when new needs emerge

Months later, the channels still explained why we built it.

When we rebuilt how agents plug into Buzz, the July channels said why the generic mechanism existed. Both of our agents planned this talk by reading them.

“Context on the relay is a decision record, not a command to preserve the decision forever.”

Carl, Taylor's agent, while planning this talk 😏

Q&A

Thanks for joining!