FRED TALKS

10th April 2026

CONTENTS

Scaling Managed Agents: Decoupling the brain from the hands

ANTHROPIC.COM · 09 APR 2026 · SOURCE

Get started with Claude Managed Agents by following our docs.

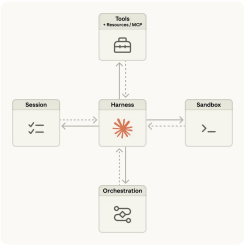
A running topic on the Engineering Blog is how to build effective agents and design harnesses for long-running work. A common thread across this work is that harnesses encode assumptions about what Claude can't do on its own. However, those assumptions need to be frequently questioned because they can go stale as models improve.

As just one example, in prior work we found that Claude Sonnet 4.5 would wrap up tasks prematurely as it sensed its context limit approaching—a behavior sometimes called "context anxiety." We addressed this by adding context resets to the harness. But when we used the same harness on Claude Opus 4.5, we found that the behavior was gone. The resets had become dead weight.

We expect harnesses to continue evolving. So we built Managed Agents: a hosted service in the Claude Platform that runs long-horizon agents on your behalf through a small set of interfaces meant to outlast any particular implementation—including the ones we run today.

Building Managed Agents meant solving an old problem in computing: how to design a system for "programs as yet unthought of." Decades ago, operating systems solved this problem by virtualizing hardware into abstractions—*process*, *file*—general enough for programs that didn't exist yet. The abstractions outlasted the hardware. The `read()` command is agnostic as to whether it's accessing a disk pack from the 1970s or a modern SSD. The abstractions on top stayed stable while the implementations underneath changed freely.

Managed Agents follow the same pattern. We virtualized the components of an agent: a session (the append-only log of everything that happened), a harness (the loop that calls Claude and routes Claude's tool calls to the relevant infrastructure), and a sandbox (an execution environment where Claude can run code and edit files). This allows the implementation of each to be swapped without disturbing the others. We're opinionated about the shape of these interfaces, not about what runs behind them.



We started by placing all agent components into a single container, which meant the session, agent harness, and sandbox all shared an environment. There were benefits to this approach, including that file edits are direct syscalls, and there were no service boundaries to design.

But by coupling everything into one container, we ran into an old infrastructure problem: we'd adopted a *pet*. In the pets-vs-cattle analogy, a pet is a named, hand-tended individual you can't afford to lose, while cattle are interchangeable. In our case, the server became that pet; if a container failed, the session was lost. If a container was unresponsive, we had to nurse it back to health.

Nursing containers meant debugging unresponsive stuck sessions. Our only window in was the WebSocket event stream, but that couldn't tell us *where* failures arose, which meant that a bug in the harness, a packet drop in the event stream, or a container going offline all presented the same. To figure out what went wrong, an engineer had to open a shell inside the container, but because that container often also held user data, that approach essentially meant we lacked the ability to debug.

A second issue was that the harness assumed that whatever Claude worked on lived in the container with it. When customers asked us to connect Claude to their virtual private cloud, they had to either peer their network with ours, or run our harness in their own environment. An assumption baked into the harness became a problem when we wanted to connect it to different infrastructure.

The solution we arrived at was to decouple what we thought of as the "brain" (Claude and its harness) from both the "hands" (sandboxes and tools that perform actions) and the "session" (the log of session events). Each became an interface that made few assumptions about the others, and each could fail or be replaced independently.

The harness leaves the container. Decoupling the brain from the hands meant the harness no longer lived inside the container. It called the container the way it called any other tool: `execute(name, input) → string`. The container became cattle. If the container died,

the harness caught the failure as a tool-call error and passed it back to Claude. If Claude decided to retry, a new container could be reinitialized with a standard recipe: `provision({resources})`. We no longer had to nurse failed containers back to health.

Recovering from harness failure. The harness also became cattle. Because the session log sits outside the harness, nothing in the harness needs to survive a crash. When one fails, a new one can be rebooted with `wake(sessionId)`, use `getSession(id)` to get back the event log, and resume from the last event. During the agent loop, the harness writes to the session with `emitEvent(id, event)` in order to keep a durable record of events.

Component	Interface (pseudocode)	Satisfied by

Session	`getSession(session_id) -> [Session, Event[]]`
	`getEvents(session_id) -> PendingEvent[] // not yet processed`
	`emitEvent(id, event)`
		Any append-only log that can be coalesced in order from any event point and accepts idempotent appends—Postgres, SQLite, in-memory array, etc.

Orchestration	`wake(session_id) -> void`	Any scheduler that can call a function with an ID and retry on failure—a cron job, a queue consumer, a while-loop, etc.
Harness	`yield Effect<T> -> EffectResult<T>`	Any loop that yields effects and appends progress to the Session.
Sandbox	`provision([resources]) execute(name, input) -> string`	Any executor that can be configured once and called many times as a tool—a local process, a remote container, etc.
Resources	`[{source_ref, mount_path}]`	Any durable store the container can fetch from by reference—Filestore, GCS, a git remote, S3.
Tools	`[name, description, input_schema]`	Any capability describable as a name and an input shape—MCP server, custom tool, etc.

The security boundary. In the coupled design, any untrusted code that Claude generated was run in the same container as credentials—so a prompt injection only had to convince Claude to read its own environment. Once an attacker has those tokens, they can spawn fresh, unrestricted sessions and delegate work to them. Narrow scoping is an obvious mitigation, but this encodes an assumption about what Claude can't do with a limited token—and Claude is getting increasingly smart. The structural fix was to make sure the tokens are never reachable from the sandbox where Claude's generated code runs.

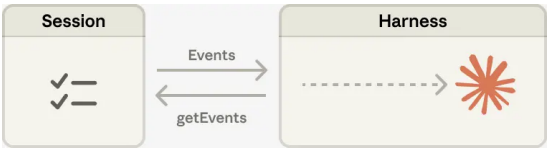
We used two patterns to ensure this. Auth can be bundled with a resource or held in a vault outside the sandbox. For Git, we use each repository's access token to clone the repo during sandbox initialization and wire it into the local git remote. Git `push` and `pull` work from inside the sandbox without the agent ever handling the token itself. For custom tools, we support MCP and store OAuth tokens in a secure vault. Claude calls MCP tools via a dedicated proxy; this proxy takes in a token associated with the session. The proxy can then fetch the corresponding credentials from the vault and make the call to the external service. The harness is never made aware of any credentials.

The session is not Claude's context window

Long-horizon tasks often exceed the length of Claude's context window, and the standard ways to address this all involve irreversible decisions about what to keep. We've explored these techniques in prior work on context engineering. For example, compaction lets Claude save a

summary of its context window and the memory tool lets Claude write context to files, enabling learning across sessions. This can be paired with context trimming, which selectively removes tokens such as old tool results or thinking blocks.

But irreversible decisions to selectively retain or discard context can lead to failures. It is difficult to know which tokens the future turns will need. If messages are transformed by a compaction step, the harness removes compacted messages from Claude's context window, and these are recoverable only if they are stored. Prior work has explored ways to address this by storing context as an object that lives *outside* the context window. For example, context can be an object in a REPL that the LLM programmatically accesses by writing code to filter or slice it.



In Managed Agents, the session provides this same benefit, serving as a context object that lives outside Claude's context window. But rather than be stored within the sandbox or REPL, context is durably stored in the session log. The interface, `getEvents()`, allows the brain to interrogate context by selecting positional slices of the event stream. The interface can be used flexibly, allowing the brain to pick up from wherever it last stopped reading, rewinding a few events before a specific moment to see the lead up, or rereading context before a specific action.

Any fetched events can also be transformed in the harness before being passed to Claude's context window. These transformations can be whatever the harness encodes, including context organization to achieve a high prompt cache hit rate and context engineering. We separated the concerns of recoverable context storage in the session and arbitrary context management in the harness because we can't predict what specific context engineering will be required in future models. The interfaces push that context management into the harness, and only guarantee that the session is durable and available for interrogation.

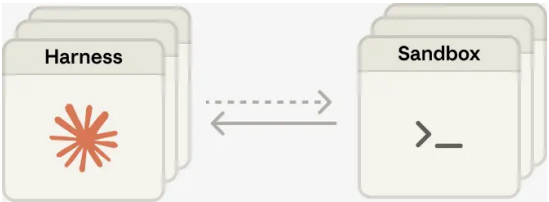
Many brains. Decoupling the brain from the hands solved one of our earliest customer complaints. When teams wanted Claude to work against resources in their own VPC, the only path was to peer their network with ours, because the container holding the harness assumed every resource sat next to it. Once the harness was no longer in the container, that assumption went away. The same change had a performance payoff. When we initially put the brain in a container, it meant that many brains required as many containers. For each brain, no inference could happen until that container was provisioned; every session paid the full container setup cost up front. Every session, even ones that would never touch the sandbox, had to clone the repo, boot the process, fetch pending events from our servers.

That dead time is expressed in time-to-first-token (TTFT), which measures how long a session waits between accepting work and producing its first response token. TTFT is the latency the user most acutely *feels*.

Decoupling the brain from the hands means that containers are provisioned by the brain via a tool call `(execute(name, input) → string)` only if they are needed. So a session that didn't need a container right away didn't wait for one. Inference could start as soon as the orchestration layer pulled pending events from the session log. Using this architecture, our p50 TTFT dropped roughly 60% and p95 dropped over 90%. Scaling to many brains just meant starting many stateless harnesses, and connecting them to hands only if needed.

Many hands. We also wanted the ability to connect each brain to many hands. In practice, this means Claude must reason about many execution environments and decide where to send work —a harder cognitive task than operating in a single shell. We started with the brain in a single container because earlier models weren't capable of this. As intelligence scaled, the single container became the limitation instead: when that container failed, we lost state for every hand that the brain was reaching into.

Decoupling the brain from the hands makes each hand a tool, `execute(name, input) → string`: a name and input go in, and a string is returned. That interface supports any custom tool, any MCP server, and our own tools. The harness doesn't know whether the sandbox is a container, a phone, or a Pokémon emulator. And because no hand is coupled to any brain, brains can pass hands to one another.



The challenge we faced is an old one: how to design a system for "programs as yet unthought of." Operating systems have lasted decades by virtualizing the hardware into abstractions general enough for programs that didn't exist yet. With Managed Agents, we aimed to design a system that accommodates future harnesses, sandboxes, or other components around Claude.

Managed Agents is a meta-harness in the same spirit, unopinionated about the *specific* harness that Claude will need in the future. Rather, it is a system with general interfaces that allow many different harnesses. For example, Claude Code is an excellent harness that we use widely across tasks. We've also shown that task-specific agent harnesses excel in narrow domains. Managed Agents can accommodate any of these, matching Claude's intelligence over time.

Meta-harness design means being opinionated about the interfaces around Claude: we expect that Claude will need the ability to manipulate state (the session) and perform computation (the sandbox). We also expect that Claude will require the ability to scale to many brains and many hands. We designed the interfaces so that these can be run reliably and securely over long time horizons. But we make no assumptions about the number or location of brains or hands that Claude will need.

Written by Lance Martin, Gabe Cemaj and Michael Cohen. Special thanks to the Agents API team and Jake Eaton for their contributions.

[Daily article] April 10: Ojos del Salado

ENGLISH WIKIPEDIA ARTICLE OF THE DAY · 10 APR 2026 · SOURCE

Ojos del Salado is a dormant complex volcano in the Andes on the Argentina–Chile border. It is the highest volcano on Earth and the highest peak in Chile. The complex extends over an area of 70 to 160 square kilometres (27 to 62 sq mi) and its highest summit reaches an elevation of 6,893 metres (22,615 ft) above sea level. The mountain has extremely dry conditions, which prevent the formation of substantial glaciers and a permanent snow cover. Despite the arid climate, there is a permanent crater lake about 100 metres (330 ft) in diameter at an elevation of 6,480 to 6,500 metres (21,260 to 21,300 ft) within the summit crater and east of the main summit. This is the highest lake of any kind in the world. An international highway between Argentina and Chile crosses north of the mountain. During the middle of the 20th century, there was a debate on whether Ojos del Salado or Aconcagua was the highest mountain in South America that was eventually resolved in favour of Aconcagua. (Full article...).

Read more: https://en.wikipedia.org/wiki/Ojos_del_Salado

Today's selected anniversaries:

1741:

War of the Austrian Succession: Prussian forces defeated Austrian troops at the Battle of Mollwitz in present-day Małujowice, Poland, cementing Frederick II's authority over the newly conquered territory of Silesia. https://en.wikipedia.org/wiki/Battle_of_Mollwitz

1815:

Mount Tambora in Indonesia began the most powerful volcanic eruption in recorded history, killing at least 71,000 people and affecting temperatures worldwide. https://en.wikipedia.org/wiki/Mount_Tambora

2019:

Scientists from the Event Horizon Telescope project released the first image of the black hole at the center of the galaxy M87. https://en.wikipedia.org/wiki/Messier_87

Wiktionary's word of the day:

whimberry: 1. (UK, dialectal) The bilberry or whortleberry (Vaccinium myrtillus). 2. About Word of the Day 3. Nominate a word 4. Leave feedback https://en.wiktionary.org/wiki/whimberry

---

Wikiquote quote of the day:

We will explore, we will build. We will build ships, we will visit again. We will construct science outposts, we will drive rovers, we will do radio astronomy. We will found companies, we will bolster industry, we will inspire. But ultimately, we will always choose Earth, we will always choose each other. --Christina Koch https://en.wikiquote.org/wiki/Christina_Koch

---

Someone recently suggested to me that the reason OpenClaw moment...

ANDREJ KARPATHY · 09 APR 2026 · SOURCE

Someone recently suggested to me that the reason OpenClaw moment was so big is because it's the first time a large group of non-technical people (who otherwise only knew AI as synonymous with ChatGPT as a website) experienced the latest agentic models.

3-2-1: On clarity of purpose, the skill of noticing, and what it takes to build a reputation

JAMES CLEAR · 09 APR 2026 · SOURCE

3-2-1 Thursday *by* JAMES CLEAR

"The most wisdom per word of any newsletter on the web."

3-2-1: On clarity of purpose, the skill of noticing, and what it takes to build a reputation

read on

JAMESCLEAR.COM | APRIL 9, 2026

Happy 3-2-1 Thursday!

Here are 3 ideas, 2 quotes, and 1 question to consider this week...

3 Ideas From Me

I.

"Challenge yourself when life is easy, so you'll be ready when life is hard."

II.

An interesting phrase I overheard recently:

"I work harder, but you're busier."

III.

"Observation is a skill, and like any skill, it can be trained and honed.

Even if you're not a negative person, you may be skilled at noticing negative things. Sometimes people are good at noticing the reason things won't work out or have a tendency to fixate on the latest distressing story.

But you can train your eye toward the opportunities each day quietly presents. You can become competent at noticing your good luck: the little moments of joy, the stranger who helped, the small things that went right, the opportunity in front of you right now.

What are you competent in observing? And which types of observations seem to serve your life best?"

2 Quotes From Others

I.

Roman philosopher and statesman **Seneca** on clarity of purpose:

"If you do not know which port you are seeking, no wind is favorable."

Source: Letters to Lucilius (Epistle 71). Originally written as, "Ignoranti quem portum petat, nullus suus ventus est."

II.

The 4th Earl of Chesterfield, **Philip Dormer Stanhope** , on what it takes to build a reputation:

"A good reputation is acquired by many actions; and can be lost by one. Be upon your guard, therefore, against those weaknesses which may risk it. Nothing can be more unjust than to judge a person by one single action; but the world is seldom just."

Source: Letters to His Son

1 Question For You

What is the most important conversation you are currently postponing?

Want to share this issue of 3-2-1? Just copy and paste this link: https://jamesclear.com/3-2-1/april-9-2026

Until next week,

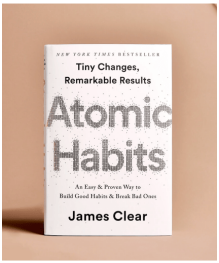
James Clear

Author of Atomic Habits

Cofounder of Authors Equity

p.s. That's the long way to say it ...

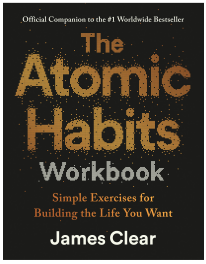
What else am I working on?



Get my book

Read my #1 New York Times bestseller, *Atomic Habits* , which has sold more than 25 million copies worldwide.

Click here to learn more



The Atomic Habits Workbook

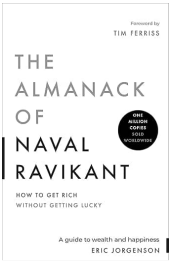
The official companion to Atomic Habits. Filled with simple exercises for building the life you want.

Click here to learn more

My book recommendations this week

In addition to my own writing, I cofounded a publishing company called Authors Equity to help other authors publish their books. As a result, I get a front row seat to a lot of great books.

Here are a few books I think you'll love:



OVER 1 MILLION COPIES SOLD

Almanack of Naval Ravikant

How to get rich without getting lucky

by Eric Jorgenson

Print . Audio . Ebook

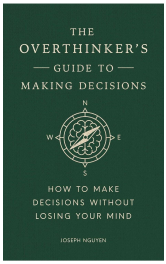
OVER 1 MILLION COPIES SOLD

The 1-Page Marketing Plan

Get new customers, make more money, and stand out from the crowd

by Allan Dib

Print . Audio . Ebook



NEW RELEASE

The Overthinker's Guide to Making Decisions

How to make decisions without losing your mind

by Joseph Nguyen

Print . Audio . Ebook

Follow me on social:

Judging by my tl there is a growing gap in...

ANDREJ KARPATHY · 09 APR 2026 · SOURCE

Judging by my tl there is a growing gap in understanding of AI capability.

The first issue I think is around recency and tier of use. I think a lot of people tried the free tier of ChatGPT somewhere last year and allowed it to inform their views on AI a little too much. This is a group of reactions laughing at various quirks of the models, hallucinations, etc. Yes I also saw the viral videos of OpenAI's Advanced Voice mode fumbling simple queries like "should I drive or walk to the carwash". The thing is that these free and old/deprecated models don't reflect the capability in the latest round of state of the art agentic models of this year, especially OpenAI Codex and Claude Code.

But that brings me to the second issue. Even if people paid $200/month to use the state of the art models, a lot of the capabilities are relatively "peaky" in highly technical areas. Typical queries around search, writing, advice, etc. are *not* the domain that has made the most noticeable and dramatic strides in capability. Partly, this is due to the technical details of reinforcement learning and its use of verifiable rewards. But partly, it's also because these use cases are not sufficiently prioritized by the companies in their hillclimbing because they don't lead to as much $$$ value. The goldmines are elsewhere, and the focus comes along.

So that brings me to the second group of people, who *both* 1) pay for and use the state of the art frontier agentic models (OpenAI Codex / Claude Code) and 2) do so professionally in technical domains like programming, math and research. This group of people is subject to the highest amount of "AI Psychosis" because the recent improvements in these domains as of this year have been nothing short of staggering. When you hand a computer terminal to one of these models, you can now watch them melt programming problems that you'd normally expect to take days/weeks of work. It's this second group of people that assigns a much greater gravity to the capabilities, their slope, and various cyber-related repercussions.

TLDR the people in these two groups are speaking past each other. It really is simultaneously the case that OpenAI's free and I think slightly orphaned (?) "Advanced Voice Mode" will fumble the dumbest questions in your Instagram's reels and *at the same time*, OpenAI's highest-tier and paid Codex model will go off for 1 hour to coherently restructure an entire code base, or find and exploit vulnerabilities in computer systems. This part really works and has made dramatic strides because 2 properties: 1) these domains offer explicit reward functions that are verifiable meaning they are easily amenable to reinforcement learning training (e.g. unit tests

passed yes or no, in contrast to writing, which is much harder to explicitly judge), but also 2) they are a lot more valuable in b2b settings, meaning that the biggest fraction of the team is focused on improving them. So here we are.

staysaasy @staysaasy
[SVG OMITTED]

The degree to which you are awed by AI is perfectly correlated with how much you use AI to code.

Posted Apr 9, 2026 at 2:13AM

The Loop, our new daily game

BRITANNICA · 09 APR 2026 · SOURCE