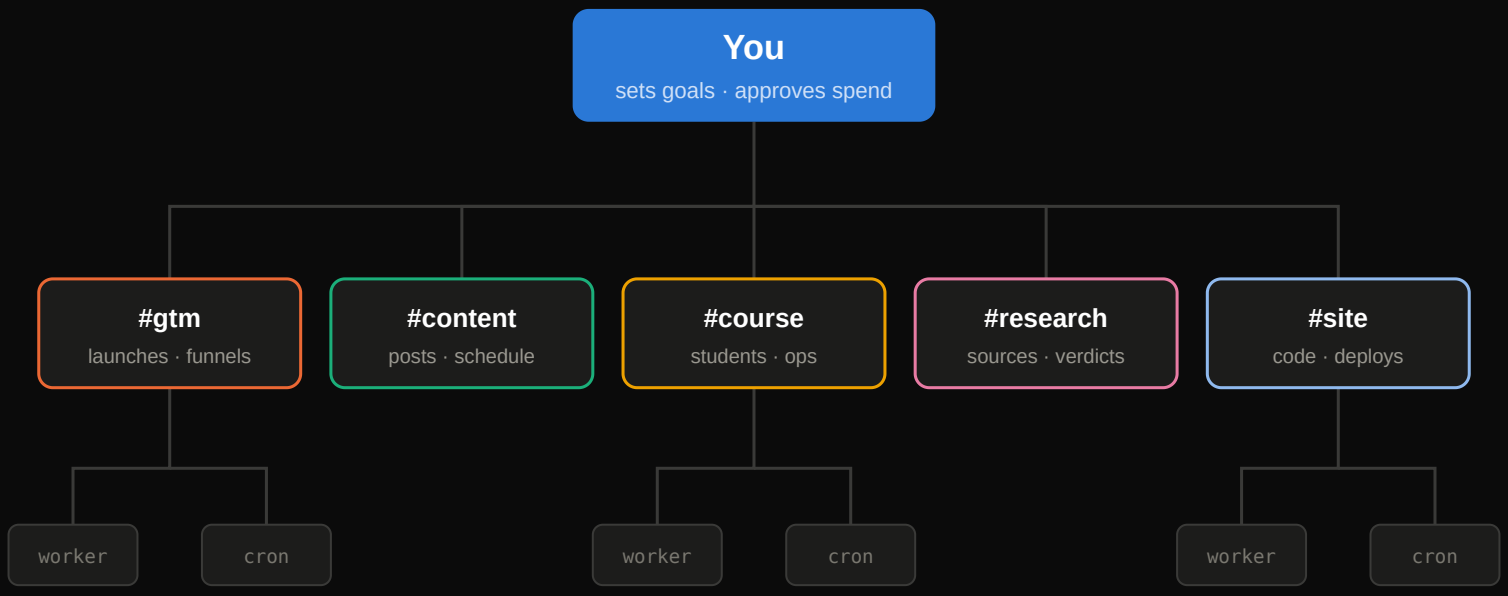


FREE GUIDE

The Definitive Guide to Hiring AI Agents as Employees

How I run a one-person company with more than thirty AI agents: the job descriptions, desks, decision rights, reports and reviews that make them worth trusting.

Who this is for: founders, operators and professionals who already use AI every day and want it to *own work* instead of waiting for prompts. No code required to start. 13 pages, a 25-minute read, and a 7-day plan at the end.

Nathan Wang · AI Superpower

course-ai.app · 2026 edition

CONTENTS

A field guide, not an essay

Read it once front to back. After that, jump to the chapter for the problem in front of you.

HOW TO USE THIS GUIDE

Every chapter ends with something you can copy

A template, a rule or a checklist. The ideas work with any agent stack: a chat assistant with projects, a coding agent in a terminal, or a full agent gateway. I name the tools I use once, in chapter 2, and after that I talk about the method, because tools change every few months and the method hasn't.

00 · THE SHIFT

Tools wait. Employees own.

Most people use AI like a very smart hammer. The work only moves when they pick it up. That caps what AI can do for you at the number of hours you spend prompting.

I teach AI to working professionals, I ship my own products, and I run all of it as one person. Behind that are more than thirty AI agents. Each has one job and one channel where I talk to it. One runs go-to-market. One runs content and the posting schedule. One runs my course operations. One does research and refuses to state anything it can't source. One writes and deploys my website.

None of them is smarter than the chat assistant you already use. They run on the same models. What's different is how they're set up. Each one has a job description, a place to work, written rules about what it may decide alone, a fixed format for reporting, a memory it maintains, and a review process that turns my corrections into permanent rules.

That's the whole secret. An agent becomes an employee when you manage it like one.

What changes when you make the shift

	AI AS A TOOL	AI AS AN EMPLOYEE
Who starts the work	You, every time	A goal, a schedule, or an event
Context	Re-explained every chat	Lives in its files, loaded every time it starts
Mistakes	Fixed in that one chat, then repeated	Become a rule it follows from then on
Output	A draft you finish	A finished, checked result, or a clear ask
Your job	Operator	Manager: goals, approvals, reviews

THE HONEST PART

Agent employees fail in very human ways

They say "done" when a step is only started. They report a number without checking what it measures. They repeat an action that already happened. Most of this guide is about the management systems that catch those failures, because the failures are what stop people from trusting agents with real work.

01 · HIRING

Hire for a job, not a task

A task ends when the chat ends. A job keeps going. Before you set anything up, write the role the way you'd write it for a human hire.

The first mistake I made was building one "do everything" assistant. It knew a little about everything and nothing well. It mixed up products, forgot which audience it was writing for, and every answer began with ten minutes of re-explaining. Splitting it into narrow roles fixed more than any prompt trick did.

The test for a good role: you can name what it owns, what it doesn't own, and what "done" looks like, each in one sentence. If you can't, the role is too broad. Split it.

Role card · template (fill in, save as IDENTITY.md)

Name	GTM
Owns	Launch plans, growth tests, funnels, distribution copy for my products
Does NOT own	Building the product. Sending anything to a real person without my OK
Done looks like	A shipped, checked result with a link, or one clear question for me
Voice	Short, direct, answer first. Opinionated, with one reason
Hands off to	#content for posting, #site for code, #research for fact checks
Scoreboard	Emails captured, paid conversions. Not likes, not followers

Three rules for splitting roles

- **Split by outcome, not by skill.** "Grow the newsletter" is a role. "Write well" is not. Every agent can write.
- **One owner per surface.** If two agents can post to the same account, one day both will. Give posting to one of them and have the others hand it off.
- **The "Does NOT own" line matters most.** It's what stops an agent from happily doing another agent's job badly.

START HERE

Your first hire is the job you do most often and like least

For most people that's a weekly report, inbox triage, a content calendar, or research before meetings. Choose work that repeats and has a clear finish line. Leave "be my strategist" for later.

02 · ONBOARDING

Give every agent a desk

An employee without a desk keeps turning up in your office asking where things are. Give each agent a place that's only its own: one channel to talk in and one folder to work from.

My setup: each agent has its own Discord channel, and that channel is the only place I talk to it. Behind it sits a folder the agent reads every time it starts. I run this on OpenClaw, an open-source agent gateway, with Claude as the model. You can build the same thing with a coding agent pointed at a folder, or a chat assistant's "project" feature. What matters is having the same files every time.

The six files on every desk

FILE	WHAT'S IN IT	WHO WRITES IT
`IDENTITY.md`	The role card from chapter 1	You, once
`SOUL.md`	How it works: principles, decision rights, what "good" means	You, then refined
`TOOLS.md`	What it can reach and how: accounts, scripts, which route is allowed for which action	You and the agent
`MEMORY.md`	An index of what it has learned about you and the business	The agent
`HEARTBEAT.md`	Its standing routine: what to check each time it's woken	You
`IMPROVEMENT.md`	Rules made from past mistakes (chapter 7)	The review loop

Why one channel per agent

Context never leaks between jobs. The go-to-market agent never sees half a thread about a code bug. And when you scroll back, each channel reads like a work log for one job.

Why plain files

You can read them, edit them, and track changes. When an agent behaves oddly, the cause is almost always a line in one of these files, and you can go and look at it.

COMMON MISTAKE

Writing the rules only in chat

"From now on, always do X" typed in a chat lasts until that conversation is summarized or restarted. If a rule matters, it goes in a file. If it's only in chat, the agent will eventually lose it.

03 · AUTHORITY

Write down decision rights

An agent that asks permission for everything is slow. One that asks for nothing will eventually email your investor by mistake. Choose the line on purpose and write it down.

I use a four-level ladder. Every agent's SOUL file says which rung it can act on without me, and the list of things that always need my OK is short and explicit.

L0 · ASK
You decide
New spend. Anything that reaches a real person. Deletes. Public posts not already approved.

L1 · PROPOSE
It drafts, you approve
Outreach copy, pricing changes, a new campaign plan.

L2 · ACT + REPORT
It acts, then tells you
Fixing its own bugs, correcting data, reversible edits, internal docs.

L3 · SILENT
It just does it
Routine checks that found nothing. No news isn't news.

The two gates I never delegate

Money and people. Any new spend, and anything that reaches a real human (an email, a DM, a friend request, a post), waits for my explicit OK. Everything else that can be undone, the agent decides.

One refinement makes this workable: **an approval covers a defined batch, not a whole conversation.** When I approve "these five replies," the agent sends those five. When those people write back, that's new input and needs a new OK. Without that line, one "yes" slowly turns into permission for everything.

Decision rights · paste into SOUL.md

Act alone	Anything reversible and inside your role. Decide, do it, report the result
Always ask	New spend · messages to real people · deleting data · publishing to a new public surface
How to ask	Show the exact recipient and the exact text, then wait. One OK = one defined batch
Disagree	If my instruction looks wrong, say so first, with one concrete reason. Then do what I decide

WHY "DISAGREE" IS IN THERE

You're hiring for judgment, not obedience. The most valuable thing my agents do is stop me, with evidence, before I spend money on a plan built on an old assumption.

04 · REPORTING

Demand a report you can trust

The most dangerous word an agent can say is "done." Half the time it means "I started it." Fix the format, and fixing the behavior follows.

Every status report from my agents follows one contract. It's the single change that did the most to make me trust them.

1. **Verdict first, in bold.** Line one is the answer. "Build passed." "Fixed, not deployed yet." "No, that won't work."
2. **Three states, never blurred.** *Implemented* (exists, with a link). *Still blocked* (and whose move it is). *Not yet verified* (what it didn't see working).
3. **One "Next" line, and it's never the agent's own promise.** It's either my action plus the exact reply that unblocks it, or a scheduled check with a time.
4. **Short.** About 120 words. The reasoning is available if I ask.
5. **Evidence as pointers.** "Checked: the live page, the database row." Not a story.

BEFORE

"Great news! I've updated the landing page with the new pricing and committed the changes. I also took the opportunity to improve the header. Everything should be working now. Let me know if you'd like any changes! I'll keep monitoring."

AFTER

Pricing is live on the site.

Implemented: new prices, commit a1b2c3, live page checked.

Not verified: checkout with a real card.

Next: reply `test card` and I'll run a $1 purchase and refund it.

"Committed" is not "working"

Look at what "Before" actually claims: it saved a change. It doesn't claim the page loads, the price shows, or checkout works, and "should be working" covers the gap. Train your agents to say only what they *observed*. If they didn't load the live page, it's "not verified," however sure they feel.

THE "I'LL KEEP MONITORING" TRAP

Most agents stop running when their reply ends. "I'll report back when it finishes" is a promise nothing will keep, unless a scheduled wake-up actually exists. Ban the phrase unless the schedule is real. Chapter 6 covers how to set one up.

05 · MEMORY

Memory is a filing cabinet, not a diary

An agent that remembers everything remembers nothing useful. Give it three drawers, and put each fact in exactly one of them.

About you

Preferences, decisions, projects, people, context

`memory/`

"Nathan approves spend, never outreach, by default"

About the world

Products, companies, markets, facts checked against sources

`wiki/`

"Search Partner clicks convert differently. Segment them."

How to work

Rules and lessons from corrections and past failures

`IMPROVEMENT.md`

"A queued email is not a delivered email. Check."

Five filing rules

- **One fact per file,** with a one-line summary at the top. The agent reads the index and opens only what's relevant.
- **Record the "why" with every rule.** A rule without its reason gets applied in the wrong places, or dropped the first time it's inconvenient.
- **Write down surprises, not routine.** "The report says 6,245 members, the real number is 745" is worth keeping. "Posted today's update" isn't.
- **Dates are absolute.** "Last week" means nothing three months later. Write 2026-09-14.
- **Contradictions get escalated, not settled quietly.** If a new fact conflicts with an old one, the agent asks you which is true instead of picking one.

THE QUIET FAILURE

Memory is a claim from the past, not the current state

"The pricing page was fixed" was true when it was written. Teach agents to check memory against the live system before acting on it or repeating it to you. Old memory stated confidently is how a stale fact reaches your customers.

06 · AUTONOMY

Work that happens while you sleep

An employee who only works while you're watching isn't really an employee. Give each agent a routine it runs without you, and a real way to come back to unfinished work.

Two kinds of standing work

Heartbeats · "check your area"

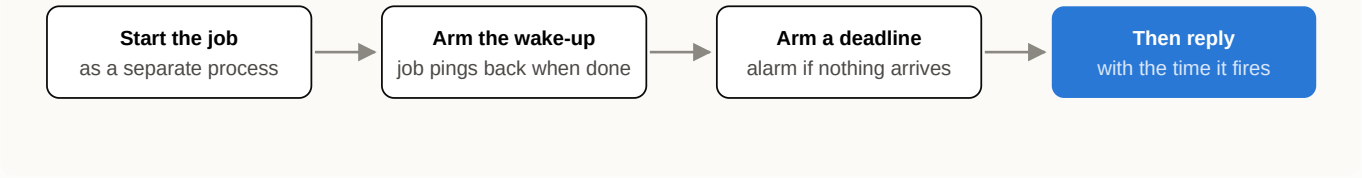
A scheduled wake-up that reads HEARTBEAT.md and runs its checklist: new signups? a failed job? a reply waiting on a draft? If nothing needs attention, it stays silent. Silence is the correct output most of the time.

Scheduled jobs · "do this at 8am"

A specific job on a specific schedule: the Monday newsletter draft, the Thursday community digest, a nightly metrics pull. Each job's instructions have to be complete, because it starts with no memory of your conversation.

The rule that makes this work: arrange the wake-up

Long work (a video render, a big import, a deploy) outlasts the agent's reply. Once the reply is sent, the agent is gone. Nothing brings it back unless something was set up to.



Failures should wake the agent, not you. If a job fails, its first move is to bring the agent back to diagnose and fix it, not to send you an error log. You get a message only when there's a verified result or a decision that's genuinely yours.

A REAL ONE

The re-run that would have double-posted

One of my agents was scheduling twelve social posts when it hit a time limit. Most of the posts were already in place. Running the job again from the start would have published them twice. The rule now: after a timeout, **check what already happened, then redo only what's missing.**

07 · IMPROVEMENT

Performance reviews, every night

If you correct a human twice, you expect them to learn. Most AI setups forget the correction as soon as the chat ends. Build the loop that makes corrections stick.

Every night a review job reads each agent's day: where I corrected it, where a tool call failed, where it retried the same thing three times. It groups repeats together and drafts a rule for each one. The important step is the gate: a rule is installed only if it would have prevented the failure it came from.

What a good rule looks like

Example rule from my own setup

"Queued" is not "delivered" (n=4). The email service returns 200 "Queued" for messages it later drops. Before reporting a send as done, check for a delivery event for that message. If there isn't one, say "sent, delivery not confirmed."

Why it's good: it names the trigger, the reason, and the exact behavior to replace it with. The n=4 records how often the mistake happened, so rules seen rarely can be retired.

Your part is five minutes a week

- **Correct in writing, with the reason.** "Wrong, because the live page still shows the old price" teaches. "Wrong" doesn't.
- **Say it when something went right, too.** Otherwise the loop only learns what to avoid and the agent gets timid.
- **Read the weekly digest** of new rules and delete any you disagree with. The loop proposes; you still set the culture.

08 · FAILURE MODES

Five ways agent employees fail

I've hit every one of these. None of them comes from a weak model. Each comes from a missing management system, so each has a system-level fix.

FAILURE	WHAT IT LOOKS LIKE	THE GUARD
1 · Fake done	"Deployed!" The code was saved; nobody loaded the live page.	The three-state report (ch. 4). "Done" means observed working, with a link.
2 · Silent failure	A background job dies at 2am. You find out on Friday, when you go looking.	Every job reports its own failure and wakes its agent. Every long job has a deadline alarm.
3 · Double action	The same email sent twice, the same post published twice, after a retry.	Before any retry, check what already happened. One owner per surface.
4 · Stale memory	States something as current that was fixed, or broke, three weeks ago.	Memory is a pointer, not proof. Check the live system before acting or repeating it.
5 · Confident wrong number	A metric that's real but means something other than what the report says.	Ask what produced the number and what else it includes. Split totals before comparing.

Number 5 is the one that costs money

An ad channel of mine once showed a cost per signup that was a fraction of every other channel. The number was accurate. The conclusion was wrong: most of those signups came from low-cost regions and partner sites that didn't match my audience. The agent reported the total without asking what was in it.

The same pattern showed up elsewhere. A dashboard showed 6,245 community members when the real figure was 745. An email API returned "success" for messages that were never delivered. A speed test reported "no problem" because the test itself couldn't detect one.

> THE FIX THAT COVERS ALL OF THESE
>
> **Prove the instrument can see the problem before trusting "no problem"**
>
> Before an agent reports "nothing is wrong," it should show that the check *would have caught* a known problem. A test that can't fail proves nothing. Put this in every agent's SOUL file. It's the most valuable line I've written.

Your first AI employee in 7 days

You don't need thirty agents. You need one that you trust. Build that one properly, and the next ten take an afternoon each.

DAY 1	**Pick the job.** The repeating task you like least and can define "done" for. Fill in the role card (ch. 1).
DAY 2	**Set up the desk.** One channel or project, one folder, the six files. Start with IDENTITY and SOUL; the rest can be three lines each.
DAY 3	**Write decision rights.** Paste the block from chapter 3. List the tools it may use, and which actions need your OK.
DAY 4	**Do the job together.** Run it once by hand, watching. Every time you correct it, write the correction and the reason into its files.
DAY 5	**Enforce the report.** Add the three-state contract (ch. 4). Reject any report that says "done" without a link or evidence.
DAY 6	**Put it on a schedule.** One scheduled run of the job. Check that a failure actually reaches you.
DAY 7	**First review.** Read the week's corrections. Turn every one you made twice into a rule in IMPROVEMENT.md.

Launch checklist: is it an employee yet?

- [ ] It owns one job, and you can say in one sentence what it doesn't own.
- [ ] It starts every run by reading its own files, not your re-explanation.
- [ ] It can't spend money or contact a real person without your OK.
- [ ] Its reports start with a verdict and separate done / blocked / unverified.
- [ ] It runs at least once a week without you starting it.
- [ ] A failed run reaches you (or wakes the agent) within the hour, without you checking.
- [ ] Yesterday's correction is in a file, so it won't happen again tomorrow.

10 · WHERE TO GO NEXT

You don't have to build this alone

Everything in this guide is free and complete. If you'd rather build it with other people, and with me looking at your actual setup, here's where to find me.

BUILD IT LIVE

AI Superpower · the cohort on Maven

Four weeks, live sessions and workshops. You build your own AI-powered app and agent, from landing page to a live agent on your own domain, with a working result at the end of each session instead of more slides. **https://maven.com/ai-nate/ai-superpower**

KEEP LEARNING, FREE

The AI Superpower community

The free account you made to download this also opens the community: what I'm building, what broke, and what I changed, posted as it happens, plus a free starter course. **https://course-ai.app/community**

ANOTHER FREE GUIDE

The KV Cache, from zero

A 26-page plain-English explainer on the memory every AI company is now paying for: what it is, why it outgrows the model, and why the storage industry cares. **https://course-ai.app/projects/kv-cache**

About me

I'm Nathan Wang. I build AI products and teach AI to working professionals. More than 50,000 students have taken my courses on Udemy, LinkedIn Learning and CCTalk. Everything in this guide comes from how I run my own work, including the mistakes.

> SHARE IT
> If this helped, send a friend the page instead of the PDF, so they get their own copy and future updates: https://course-ai.app/projects/hire-ai-agents