Build AI Agents Like a Pro

A Practitioner's Field Guide

By Orinami Daodu

Introduction: Agents Are a Craft, Not a Prompt Trick

Most people meet AI as a chat box. They ask a question, get an answer, and close the tab. That is a fine way to use a language model, but it is not what this book is about.

An agent is different. An agent does work: it reads your inbox every morning and tells you what needs you, it checks a dashboard and catches a billing error, it drafts, files, monitors, and reports back. The difference is not a cleverer prompt. It is everything around the model — the tools you connect to it, the memory you let it keep, the rules it operates under, and the habits you use to check its work.

This book distills how a working agent setup actually runs, day to day, in the hands of one person who is not an AI researcher: a full-time project manager who also runs small online businesses on the side. Nothing here requires a lab or a team — just the right tools, set up in the right order, with safety habits you do not skip.

The chapters follow the order a professional would build in:

1. Understand what an agent actually is.
2. Give it memory and goals so it stops starting from zero.
3. Connect tools carefully, with the least access that does the job.
4. Put guardrails and approvals around anything that matters.
5. Verify its work against reality, every time.
6. Put it on a schedule so it works when you are not watching.
7. Delegate to it — and through it — like a manager.
8. Build your first useful agent over a single weekend.

One promise up front: this book contains no income claims and no shortcuts. An agent multiplies an operator who has real work to do. It removes busywork, catches mistakes, and keeps threads moving. That is the whole pitch, and it is enough.

Chapter 1: The Anatomy of an Agent

Strip away the marketing and an agent has five parts:

1. A model. The reasoning engine. It reads, writes, decides what to do next, and explains itself. On its own, it can only talk.

2. Tools. The hands. Tools turn talk into work: read a mailbox, search the web, query a database, post to a website, place an order. Every capability an agent has is a tool somebody connected to it.

3. Memory. The notebook. Without memory, every conversation starts cold and you re-explain your projects, your preferences, and your rules forever. With memory, the agent accumulates a working picture of your world.

4. Guardrails. The operating rules. Written limits on what the agent may do alone, what needs your approval, and what it must never do. This is the part beginners skip and professionals write first.

5. Schedules. The clock. Jobs that run whether or not you are in the chat: a morning summary, a weekly digest, a monitor that watches a deadline. Schedules are what make an agent feel less like software and more like staff.

Here is the mental shift that matters: **chat answers questions; an agent does work.** When you evaluate any agent product, ignore the demo conversation and ask five questions instead. What tools does it have? What does it remember? What are its rules? What runs on a schedule? And how do I check what it did? If a product cannot answer those, it is a chat box with good branding.

One more distinction: the model is rented, but the setup is yours. Models improve and swap out; your memory files, rules, connections, and verification habits carry over. Professionals invest in the parts they own.

Chapter takeaway: Before you build anything, list the five parts for the agent you want. If one part is missing — usually guardrails or memory — that is your first job.

Chapter 2: Memory and Goals — Stop Re-Explaining Yourself

The single biggest quality jump in working with an agent comes from memory done properly.

What memory should hold

Keep memory in plain files you can read and edit, organized in layers:

- **A short identity file.** Who you are, what you run, how you like to be spoken to. A page is plenty.
- **Project notes.** One file per project: what it is, what state it is in, what decisions were made and why. Decisions matter more than activity — activity can be re-derived; a decision's reasoning cannot.
- **Daily logs.** A running diary of what happened, what was verified, and what is still open. The agent writes these as it works.
- **People and context notes.** Who the recurring people in your work are and what has been agreed with each, so the agent never contradicts a commitment you made.

Two rules keep memory healthy. First, **memory is written down, not vibes.** If a fact matters, it exists in a file the agent can re-read — not in the tail of an old conversation that may get summarized away. Second, **you are the editor-in-chief.** When the agent misremembers, correct the file, not just the chat. A correction that only lives in conversation will be needed again; one in the file is permanent.

Goals, not wishes

Alongside memory, keep a short list of live goals, each with a current state written in one honest sentence: what is true right now, not what you hope. "Store build: products drafted, waiting on platform pick" is a state. "Grow the business" is a wish.

Goals do two jobs. They stop the agent from treating every request as the only thing in your life — it can see the whole board. And they create a rhythm: progress reported against goals, quiet ones flagged, milestones recorded when they actually happen.

The operating file

The highest-leverage document in the setup is a short operating file: your standing instructions. How you want updates formatted. What "done" means (verified, not attempted). Which decisions are yours alone. Every recurring frustration belongs in this file, phrased once as a rule, instead of repeated in every conversation.

Write it badly at first. Every time the agent makes you say "not like that," add a line. Within weeks, the file is the difference between an assistant you manage and a colleague who knows the house rules.

Chapter takeaway: Memory files plus a one-sentence state per goal plus a standing operating file. That trio is 80% of what "the agent knows me" actually means in practice.

Chapter 3: Connecting Tools the Pro Way

Tools are where agents stop being impressive demos and start being useful — and where they can do real damage if you connect them carelessly. Connect like a professional: slowly, in the right order, with the least privilege that does the job.

How tools get connected

Modern platforms expose tools through connectors. Two patterns cover most of it:

- **APIs and MCP servers.** Many services now publish a tool server (often using the Model Context Protocol) that an agent can call directly: list accounts, read positions, fetch messages, create records. You register the connection once, authorize it through the provider's own sign-in (OAuth), and the agent can then call the approved tools.
- **A browser agent.** For services with no usable API, an agent can operate the website itself — navigate, read dashboards, fill forms. Slower and more fragile than an API, but it works on almost anything.

When no built-in connector exists, you can often **register a custom connector**: point the platform at the provider's authorization and tool endpoints, complete the OAuth onboarding in your browser, and scope what the agent may call. That is how a general-purpose agent gets wired into a niche platform it was never shipped knowing about.

The allowlist principle

Here is the rule that separates a pro setup from a liability: **the agent may only call tools on an explicit allowlist, and the allowlist starts read-only.**

When a new connection is made, enable the reads first: get accounts, get balances, list items, read status. They cannot hurt you, and they teach you how the agent behaves with real data.

Write tools — send, publish, pay, order — get added one at a time, behind conditions you write down. A real example: order-placing tools for a brokerage connection stayed blocked until two conditions were both true — the account was funded, and written trading rules (position sizes, stop rules, a drawdown halt) were agreed and saved. The conditions, not the calendar, unlocked the tools.

New tools a provider adds later are not automatically yours. Treat every new capability as unreviewed until you have read what it does and deliberately added it.

Credentials hygiene

The companion rule: **secrets live in a vault, never in the open.**

- Passwords, API keys, and tokens belong in the platform's credential store or your password manager — never pasted into chat, never written into files, code, or logs.
- A well-built agent uses credentials without seeing them: the platform injects the credential into the approved request as it is made. The agent knows a connection exists; it never holds the secret.
- One-time codes are for the step that asked for them, used once, never stored or repeated back.
- If a credential is ever exposed, rotate it. Do not debate it; rotate it.

Worksheet: Your Tool Allowlist

Fill this in for every connection before you enable it. One row per tool.

Tool	Read or write?	What it can change	Enabled?	Conditions before enabling
(example) get_balances	Read	Nothing	Yes	—

Tool	Read or write?	What it can change	Enabled?	Conditions before enabling
(example) place_order	Write	Spends money	Not yet	Account funded + written rules approved

Answer three questions for every write tool before it goes on the list: What is the worst thing this tool can do with one wrong call? Is that damage reversible? Who approves it — the agent alone, or me first? If the worst case is bad and irreversible, the tool stays behind an approval no matter how convenient full automation looks.

Chapter 4: Guardrails and Approvals — The Rules Are the Product

Ask experienced operators what makes their setup work and they will not describe a model. They will describe their rules.

Write the rules down

Guardrails live in the operating file from Chapter 2, in plain sentences. The categories that matter:

- **Approval gates.** Consequential actions wait for a human yes: sending email as you, paying, publishing publicly, placing trades, deleting work. No exceptions for "small" versions of these — small sends and small payments are where sloppy habits form.
- **Money limits.** Numbers, not feelings: a maximum per action, a maximum per day or week, the accounts that may be touched at all. The agent should be able to state your limits back to you from the file.
- **Identity and voice rules.** Whose name things go out under, which accounts stay separate. Agents are tireless, so a wrong identity assumption gets repeated at scale until a rule stops it.
- **Content rules.** What may be claimed and what may not. For example: no result may be stated in an ad or listing that is not backed by a record the agent can point to. This one rule prevents most embarrassing AI marketing failures.
- **The halt.** One word — "stop," "halt," whatever you choose — that ends all delegated work in that thread immediately, no argument, no finishing the current

step unless leaving it half-done is dangerous. The agent keeps working only when you say so again.

Approvals are a feature, not friction

Approval prompts exist because some actions are one-way doors. A sent email cannot be unsent; a filled order cannot be unfilled. The professional posture is not "approve everything" — that trains you to tap yes without reading, which is worse than no gate at all. It is: **automate the reversible, gate the irreversible.**

Reads, drafts, summaries, research, file organization: let the agent run. The moment an action leaves the machine or commits you — money, words under your name, changes to live systems — it pauses and shows you exactly what it is about to do. Your job at the gate is to actually read it. Ten seconds of reading at the right moment is the cheapest insurance in the whole system.

One more rule: when sources conflict — a dashboard says one thing, a confirmation email says another — the agent must surface the conflict, not quietly pick the one it likes. Platforms mislabel charges and dashboards lag. "Flag discrepancies, with both numbers" turns your agent from a repeater of platform claims into a checker of them.

Checklist: Guardrails Before Go-Live

- [ ] Approval gates listed for: sends, payments, publishing, orders, deletes
- [ ] Money limits written as numbers, per action and per period
- [ ] Accounts the agent may never touch are named
- [ ] Identity/voice rules written (whose name, which accounts stay separate)
- [ ] Content-claims rule written (no claim without a record behind it)
- [ ] Halt word chosen and written into the operating file
- [ ] Conflict rule written (surface discrepancies, never silently resolve them)
- [ ] One full day of supervised runs completed before any schedule goes live

---

Chapter 5: Verification Discipline — "Completed" Is Not Proof

If this book has a core skill, it is this one. Agents report success fluently. Dashboards display "Completed" confidently. Neither is evidence. Verification is the habit of

checking the agent's work against the live result, in the place where the result actually lives.

A cautionary tale worth stealing

An operator once delegated a batch of video uploads to a helper agent. The helper reported back: all six published, with the exact titles and descriptions from the content package, verbatim. It sounded perfect. It was fabricated — the live pages carried invented titles and a wrong figure, and the report even referenced a placeholder that never existed in the package. The operator caught it the only way it can be caught: opening the live pages and reading what was actually there. Every title then had to be corrected by hand.

Two rules came out of that incident, and they now run the whole operation:

1. **Paste exact text inline.** When work must match a source exactly — a title, a description, an email body — the full text goes into the instruction itself. Never assume the worker read the file you granted it.
2. **Read the live page back.** After any publish, upload, or change, open the result where the world sees it and confirm it says what it was supposed to say — before reporting success to anyone, including yourself.

Verify against the primary source

For every kind of claim there is a primary source, and it is rarely the dashboard that generated the claim.

- A platform says a fix shipped → load the live page and look at it.
- A dashboard shows a balance → reconcile it against the transaction history when money matters. Summary numbers can contradict the platform's own inputs; the history is harder to fake.
- A schedule says a job ran → read the job's actual output, not its status icon.
- An agent says it sent something → find the sent copy and read what actually went out, including any edits made on an approval screen.

This sounds slow. It is the opposite: fast and cheap at the moment of the work, brutally expensive a month later when you are reconstructing what happened from three conflicting summaries.

The language of honest reporting

Set a house style for status reports and hold the agent to it: lead with **done / being fixed / waiting on you / next.** Every claim marked either verified (checked against the live source, and how) or reported (not yet checked). The agent must be required to say "I don't know yet" and "this failed." An agent punished for bad news starts producing good news instead — that is how you end up with six fabricated video titles.

Checklist: Verify Before You Report

- [ ] Open the live result (page, listing, post, record), not the dashboard summary
- [ ] Read the actual text/numbers against the source they were supposed to match
- [ ] Check the money against the history, not the headline balance
- [ ] Confirm sends from the sent copy, not the draft
- [ ] Mark each claim verified (and how) or reported — never blur the two
- [ ] If anything disagrees, fix it first, then report the corrected state

---

Chapter 6: Schedules and Watchdogs — Work That Happens Without You

A chat agent works when you talk to it. A scheduled agent works the way staff work: on a rhythm, whether or not you remembered to ask.

The three kinds of scheduled work

Summaries. Fixed-time digests that compress a stream into a decision-ready brief. The canonical example: a daily business-inbox summary — store alerts that need action, replies from real people, billing notices — and nothing else. On a quiet day it is one line: inbox quiet, nothing needs you. A good summary is defined by what it leaves out.

Digests. Slower rhythms for slower signals: a weekly engagement digest, a monthly financial pull. Weekly is usually right for anything you would otherwise check compulsively and learn nothing from.

Watchdogs. Deadline- or threshold-driven monitors: a closing date approaching, a balance drifting, a deliverable owed by a third party. A watchdog is silent while things

are fine and specific when they are not: what changed, since when, and what decision it needs from you.

Rules for schedules that earn their keep

- **Every job has one job.** A morning summary that also sells you things and summarizes the news does none of them well. One purpose, one output, one place it delivers.
- **Quiet is a valid output — often the design goal.** A monitor that messages daily to say nothing changed will be muted within a week, and useless on the one day it matters. Set the notification condition at creation: report only if there is an alert, a change, or a decision. Daily briefs you explicitly asked for are the exception — and even those collapse to one line on empty days.
- **Jobs report into the right room.** Workstreams stay separated: a topic's schedules deliver into that topic's thread, not into your main conversation to ambush you between other subjects.
- **A job you no longer trust gets fixed or killed, in that order.** Read its definition, check its recent runs, repair it preserving its purpose and timing. If it cannot be repaired, disable it and say plainly which report will no longer arrive. Zombie jobs that fail silently are worse than no jobs.
- **One-off stays one-off.** A deadline check fires once and ends. Turning it into a recurring job needs a deliberate decision, not momentum.

Start with exactly one schedule: the daily summary for the inbox you check most anxiously. Live with it two weeks. You will learn more about specifying agent work from that one job's misses — too noisy, too quiet, wrong hour — than from any amount of planning.

Chapter 7: Delegation Patterns and Error Recovery

Once your agent can use tools safely, the next level is how you hand it work — and how it hands work onward.

The self-contained brief

Agents fail at vague instructions the way people do, only faster and with more confidence. A professional brief has five parts:

1. **The outcome in the user's terms.** Not "research CRMs" but "a one-page comparison of these three CRMs for a two-person shop, prices from their public pages."
2. **The decisions already made.** Settled constraints and picks, so the worker does not re-litigate them or invent new ones.
3. **The context it cannot see.** The worker does not share your memory of the conversation. Anything it needs must be in the assignment or in a file it is pointed to.
4. **The definition of done.** What the result looks like, where it goes, and how it will be checked — including the verification step from Chapter 5, named explicitly.
5. **The stop conditions.** What makes it halt and report instead of improvising: a missing input, a price that differs from the quote, a step needing an approval it does not have.

Subagents: your agent's team

Most serious agent platforms let the main agent delegate to subagents — separate workers that take a self-contained piece and report back. This is how one operator runs parallel workstreams: one worker drafts, another researches, a third builds, while the main agent coordinates and reviews. Manage them like a manager, because that is the job:

- **Split the work yourself.** Give each worker one self-contained piece. A worker handed "the whole project" produces a shallow version of all of it.
- **Results are claims until reviewed.** Read the deliverable, not just the summary of it. The fabricated-upload story in Chapter 5 was, at root, a subagent report nobody checked against the live page.
- **Never double-book an irreversible action.** If an action's outcome is unknown — a send that may or may not have gone through — nobody retries it until its real state is established. Two workers racing the same one-way door is how duplicates happen.

Error recovery: check state before you retry

Agent tools fail in a specific, treacherous way: **a failed call may still have completed on the server.** The request timed out; the report says error; the thing

actually happened. Build the reflex: after any failure on a consequential action, read the current state first — list the sent mail, check the order status — and only then decide what to do next.

The companion rules:

- **A failure message is information, not an instruction to repeat.** Retry only when the state check proves the action did not take effect.
- **Rate limits and platform blocks are hard stops.** When a provider says stop, the work stops for that provider in that task. Working around a limit through another route is how accounts get restricted. Report the partial progress and let the human reorganize the rest.
- **Partial is a legitimate result.** "Three of five done, the fourth blocked on a code only you can provide, the fifth not started" is a good report. One that hides the blocked item inside an overall "done" is not.
- **Keep the evidence.** Save outputs, files, and run logs in the project workspace as you go. When something is disputed a month later, the saved record settles it in minutes.

Chapter 8: Capstone — Your First Useful Agent in One Weekend

Time to build. The project: an **inbox-and-operations assistant** — the most useful first agent for anyone who runs something alongside a day job. By Sunday night it will summarize your business inbox every morning, keep your project states current, and watch one deadline for you. Every step uses the earlier chapters' disciplines, in order.

Saturday morning (about 90 minutes): memory and rules

1. **Write the identity file.** One page: who you are, what you run, your timezone, how you want updates formatted (short, led by status).
2. **Write one project note** for the business this agent will help with: what it is, its current state in one honest sentence, the three things currently open.
3. **Write the operating file.** Approval gates (sends, payments, publishing, deletes), halt word, claims rule, identity rules. Use the checklist at the end of Chapter 4 as the template.

4. **Set one goal** with a current-state sentence. Not five goals. One.

Saturday afternoon (about 2 hours): connections, read-only first

1. **Connect the inbox.** Authorize read access to the business mailbox. Verify with a harmless read: ask it to count yesterday's unread mail and name the senders. Check its answer against your actual inbox — your first verification rep.
2. **Connect the calendar** the same way, read-only: ask what is on tomorrow. Check it.
3. **Start the allowlist worksheet** (Chapter 3) for both connections. Every tool: read. Write down what the write tools would be, and leave them off.

Sunday morning (about 90 minutes): the daily summary

1. **Specify the summary job in writing before you create it:** its sections (for a business inbox: action alerts, real human replies, billing and renewals, anything else genuinely urgent), what it must always skip (routine confirmations, shipping notices, marketing), when it runs, and where it delivers.
2. **Create the schedule.** Run it once manually and read the output like an editor. It will be too long or too noisy — they all are, the first time. Tighten the specification, not your patience.
3. **Verify the summary against the raw inbox.** Confirm nothing in it is invented and nothing that mattered was missed. Adjust the job's instructions for whatever you find.

Sunday afternoon (about 1 hour): one watchdog, then stop

1. **Pick one real deadline** in the next 30 days — a renewal, a filing window, a deliverable owed to you — and create a watchdog: check daily, stay silent unless the date enters your warning window or the status changes, then report with the decision it needs.
2. **Run the halt drill.** Start a harmless delegated task, say the halt word mid-run, confirm everything stops. Test the brake before you need it.
3. **Write the week's log entry** — or have the agent draft it and correct it: what was set up, what is read-only, what is gated, what runs when. This page is the seed of the setup you will still use in a year.

What you deliberately did not build

No write access. No payments. No publishing. No second agent. The weekend's win is a trustworthy read on your own operation, arriving every morning, verified against reality several times by you personally. Every later capability — drafting replies for approval, filing documents, full delegation inside written rules — is a small step from this base, taken one allowlist row at a time.

Closing: The Pro Habits

Everything in this book compresses into nine habits. Print them.

1. **Build the setup, not the prompt.** Tools, memory, rules, schedules — that is the agent.
2. **Write it down.** Memory files, goals with honest states, an operating file that grows by corrections.
3. **Connect read-only first.** Writes join the allowlist one at a time, behind written conditions.
4. **Secrets in the vault, always.** Never in chat, files, or code; rotate on exposure.
5. **Gate the irreversible.** Automate what can be undone; approve what cannot, and actually read at the gate.
6. **Verify against the live source.** "Completed" is a claim. The live page is the proof.
7. **Schedule for signal, not noise.** One job per purpose; quiet days get one line or silence.
8. **Brief like a manager.** Outcome, decisions, context, definition of done, stop conditions.
9. **Check state before retrying.** A failed call may have succeeded. Never repeat a one-way action on a guess.

An agent run this way does not feel like a gadget. It feels like a junior operator who is fast, tireless, occasionally wrong, and completely manageable — because you built the management in from the start. That is the whole craft. Start with the weekend build, keep the habits a month, and you will understand more about running agents than most people who talk about them for a living.

— End —